Realistic simulation of the execution of applications deployed on large distributed systems with a focus on improving file management

Anchen Chai

N°d'ordre NNT :

THESE de DOCTORAT DE L'UNIVERSITE DE LYON
opérée au sein de
(INSA LYON)

Ecole Doctorale N° ED160
(Électronique, Électrotechnique, Automatique)

Spécialité/ discipline de doctorat :
Informatique

Soutenue publiquement le 14/01/2019, par :

Anchen CHAI

Titre de la thèse

Simulation réaliste de l'exécution des applications déployées sur des systèmes distribués avec un focus sur l'amélioration de la gestion des fichiers

Devant le jury composé de :

Prodan, Radu	Professeur	University of Klagenfurt	Rapporteur
Breton, Vincent	DR	CNRS IN2P2	Rapporteur
Genaud, Stéphane	Professeur	Université de Strasbourg	Examinateur
Marangozova-Martin, Vania	MCF, HDR	Université de Grenoble	Examinatrice
Benoit-cattin, Hugues	Professeur	INSA-LYON	Directeur de thèse
Suter, Frédéric	CR, HDR	CC-IN2P3	Co-directeur
Pop, Sorina	IR, docteur	CNRS	Co-directrice

Département FEDORA – INSA Lyon - Ecoles Doctorales – Quinquennal 2016-2020

SIGLE	ECOLE DOCTORALE	NOM ET COORDONNEES DU RESPONSABLE
CHIMIE	**CHIMIE DE LYON** **http://www.edchimie-lyon.fr** Sec. : Renée EL MELHEM Bât. Blaise PASCAL, 3e étage **secretariat@edchimie-lyon.fr** INSA : R. GOURDON	M. Stéphane DANIELE Institut de recherches sur la catalyse et l'environnement de Lyon IRCELYON-UMR 5256 Équipe CDFA 2 Avenue Albert EINSTEIN 69 626 Villeurbanne CEDEX **directeur@edchimie-lyon.fr**
E.E.A.	**ÉLECTRONIQUE, ÉLECTROTECHNIQUE, AUTOMATIQUE** **http://edeea.ec-lyon.fr** Sec. : M.C. HAVGOUDOUKIAN **ecole-doctorale.eea@ec-lyon.fr**	M. Gérard SCORLETTI École Centrale de Lyon 36 Avenue Guy DE COLLONGUE 69 134 Écully Tél : 04.72.18.60.97 Fax 04.78.43.37.17 **gerard.scorletti@ec-lyon.fr**
E2M2	**ÉVOLUTION, ÉCOSYSTÈME, MICROBIOLOGIE, MODÉLISATION** **http://e2m2.universite-lyon.fr** Sec. : Sylvie ROBERJOT Bât. Atrium, UCB Lyon 1 Tél : 04.72.44.83.62 INSA : H. CHARLES **secretariat.e2m2@univ-lyon1.fr**	M. Philippe NORMAND UMR 5557 Lab. d'Ecologie Microbienne Université Claude Bernard Lyon 1 Bâtiment Mendel 43, boulevard du 11 Novembre 1918 69 622 Villeurbanne CEDEX **philippe.normand@univ-lyon1.fr**
EDISS	**INTERDISCIPLINAIRE SCIENCES-SANTÉ** **http://www.ediss-lyon.fr** Sec. : Sylvie ROBERJOT Bât. Atrium, UCB Lyon 1 Tél : 04.72.44.83.62 INSA : M. LAGARDE **secretariat.ediss@univ-lyon1.fr**	Mme Emmanuelle CANET-SOULAS INSERM U1060, CarMeN lab, Univ. Lyon 1 Bâtiment IMBL 11 Avenue Jean CAPELLE INSA de Lyon 69 621 Villeurbanne Tél : 04.72.68.49.09 Fax : 04.72.68.49.16 **emmanuelle.canet@univ-lyon1.fr**
INFOMATHS	**INFORMATIQUE ET MATHÉMATIQUES** **http://edinfomaths.universite-lyon.fr** Sec. : Renée EL MELHEM Bât. Blaise PASCAL, 3e étage Tél : 04.72.43.80.46 Fax : 04.72.43.16.87 **infomaths@univ-lyon1.fr**	M. Luca ZAMBONI Bât. Braconnier 43 Boulevard du 11 novembre 1918 69 622 Villeurbanne CEDEX Tél : 04.26.23.45.52 **zamboni@maths.univ-lyon1.fr**
Matériaux	**MATÉRIAUX DE LYON** **http://ed34.universite-lyon.fr** Sec. : Marion COMBE Tél : 04.72.43.71.70 Fax : 04.72.43.87.12 Bât. Direction **ed.materiaux@insa-lyon.fr**	M. Jean-Yves BUFFIÈRE INSA de Lyon MATEIS - Bât. Saint-Exupéry 7 Avenue Jean CAPELLE 69 621 Villeurbanne CEDEX Tél : 04.72.43.71.70 Fax : 04.72.43.85.28 **jean-yves.buffiere@insa-lyon.fr**
MEGA	**MÉCANIQUE, ÉNERGÉTIQUE, GÉNIE CIVIL, ACOUSTIQUE** **http://edmega.universite-lyon.fr** Sec. : Marion COMBE Tél : 04.72.43.71.70 Fax : 04.72.43.87.12 Bât. Direction **mega@insa-lyon.fr**	M. Jocelyn BONJOUR INSA de Lyon Laboratoire CETHIL Bâtiment Sadi-Carnot 9, rue de la Physique 69 621 Villeurbanne CEDEX **jocelyn.bonjour@insa-lyon.fr**
ScSo	**ScSo*** **http://ed483.univ-lyon2.fr** Sec. : Viviane POLSINELLI Brigitte DUBOIS INSA : J.Y. TOUSSAINT Tél : 04.78.69.72.76 **viviane.polsinelli@univ-lyon2.fr**	M. Christian MONTES Université Lyon 2 86 Rue Pasteur 69 365 Lyon CEDEX 07 **christian.montes@univ-lyon2.fr**

*ScSo : Histoire, Géographie, Aménagement, Urbanisme, Archéologie, Science politique, Sociologie, Anthropologie

Acknowledgements

I would like to emphasize my acknowledgment to all the advisors, Prof. Hugues Benoit-cattin, Dr. Frédéric Suter, and Dr. Sorina Pop for guiding me during the preparation of this thesis. And I am particularly grateful for their countless hours devoted to proof-reading and correction of this thesis, as well as the previously published articles. They have made great efforts to point out my flaws all the time and encourage me to systematically pursue perfection.

Also, I want to thank Dr. Tristan Glatard for always giving me brilliant suggestions to my work and my articles during the last three years. Even though I never met him in person, I still learned enormously from all his articles, our email exchanges, and the video conferences.

Furthermore, I owe many appreciations to Prof. Radu Prodan and Prof. Vincent Breton for their willingness to be my reporters. Your precious time spent on reviewing my Ph.d manuscript and giving me valuable recommendations are deeply respected and admired.

And I feel fairly happy to have known these dear friends from CREATIS: Pierre-Antoine Ganaye, Sarah Leclerc, Noelie Debs, Fei GE, Yunyun SUN, and Hoai-Thu Nguyen. Thanks for your inspirations and help for my research, the Monday and Friday cake, the "soirée", especially for all the moments that we shared together.

Finally, I would like to thank my girlfriend Xiaoyi TIAN for her continuous support and priceless encourage throughout my thesis.

Abstract

Simulation is a powerful tool to study distributed systems. It allows researchers to evaluate different scenarios in a reproducible manner, which is hardly possible in real experiments. However, the realism of simulations is rarely investigated in the literature, leading to a questionable accuracy of the simulated metrics. In this context, the main aim of our work is to improve the realism of simulations with a focus on file transfer in a large distributed production system (i.e., the EGI federated e-Infrastructure (EGI)). Then, based on the findings obtained from realistic simulations, we can propose reliable recommendations to improve file management in the Virtual Imaging Platform (VIP).

In order to realistically reproduce certain behaviors of the real system in simulation, we need to obtain an inside view of it. Therefore, we collect and analyze a set of execution traces of one particular application executed on EGI via VIP. The realism of simulations is investigated with respect to two main aspects in this thesis: the simulator and the platform model.

Based on the knowledge obtained from traces, we design and implement a simulator to provide a simulated environment as close as possible to the real execution conditions for file transfers on EGI. A complete description of a realistic platform model is also built by leveraging the information registered in traces. The accuracy of our platform model is evaluated by confronting the simulation results with the ground truth of real transfers. Our proposed model is shown to largely outperform the state-of-the-art model to reproduce the real-life variability of file transfers on EGI.

Finally, we cross-evaluate different file replication strategies by simulations using an enhanced state-of-the-art model and our platform model built from traces. Simulation results highlight that the instantiation of the two models leads to different qualitative decisions of replication, even though they reflect a similar hierarchical network topology. Last but not least, we find that selecting sites hosting a large number of executed jobs to replicate files is a reliable recommendation to improve file management of VIP. In addition, adopting our proposed dynamic replication strategy can further reduce the duration of file transfers except for extreme cases (very poorly connected sites) that only our proposed platform model is able to capture.

Contents

Abstract **iv**

Contents **xiii**

Acronyms **xiv**

Introduction **1**

1 State-of-the-art **4**

1.1 Introduction 4

1.2 Distributed computing infrastructures 4

1.2.1 Grid computing 5

1.2.2 Cloud computing 6

1.3 Application deployment in distributed systems 6

1.3.1 Science gateways 7

1.3.2 Scientific workflows 8

1.3.3 Traces of workflow executions 9

1.3.4 Optimization of workflow execution 10

1.4 File replication 11

1.4.1 Classification for file replication 11

1.4.2 Requirements for file replication in production 12

1.4.3 Existing replication management in production systems 13

1.5 Simulation 16

1.5.1 Simulation tools 16

1.5.2 Platform models for file replication 18

1.6 Conclusion 20

2 Analyzing execution traces of one application deployed on a large distributed system **22**

2.1 Introduction 22

2.2 GATE application 23

2.3 Execution traces . . . 25
2.4 Characteristics of file transfers . . . 27
2.4.1 Coarse-grain analysis . . . 28
2.4.2 Fine-grain analysis . . . 32
2.5 Characteristics of workflow and job executions . . . 35
2.5.1 Queuing time durations . . . 35
2.5.2 Distribution of jobs . . . 36
2.5.3 Cumulative downloads for SEs . . . 38
2.6 Conclusion . . . 39

3 Realistic simulation of file transfers for applications deployed on distributed infrastructures 42
3.1 Introduction . . . 42
3.2 Execution of the GATE workflow on EGI . . . 43
3.2.1 GATE workflow in VIP . . . 43
3.2.2 Data management services in EGI . . . 44
3.2.3 Summary of the characteristics of the real system . . . 46
3.3 The SimGrid toolkit . . . 47
3.4 Simulator design . . . 48
3.4.1 Simulated services for data management on EGI . . . 49
3.4.2 Communication cost for file transfers . . . 50
3.4.3 Parameter injection . . . 51
3.5 Conclusion . . . 52

4 Realistic platform models for replaying real workflow executions 53
4.1 Introduction . . . 53
4.2 Build partial platform model from execution traces . . . 54
4.2.1 Baseline model . . . 54
4.2.2 Improvements based on execution traces . . . 55
4.3 Overall evaluation of our model . . . 65
4.3.1 Analysis of simulated transfer durations . . . 65
4.3.2 Analysis of errors . . . 67
4.3.3 Analysis of the root causes of large simulation errors . . . 68
4.4 Conclusion . . . 70

5 Towards a complete and realistic description of the Biomed VO platform 71
5.1 Introduction . . . 71
5.2 Aggregating network information from multiple traces . . . 73
5.3 Filling-in missing links in the merged platform . . . 76

5.3.1 Empirical model . . . 77
5.3.2 Machine learning model . . . 78
5.3.3 Evaluation . . . 80
5.4 Conclusion and discussion . . . 84

6 Evaluation of file replication strategies through realistic simulations 85
6.1 Introduction . . . 85
6.2 Replication strategies . . . 86
6.2.1 Dynamic replication strategy . . . 86
6.3 Simulation studies . . . 88
6.3.1 Platform Models . . . 88
6.3.2 Simulation scenarios . . . 88
6.4 Performance evaluation . . . 89
6.4.1 Impact of dynamic replication . . . 89
6.4.2 Impact of different prestaging lists on static replication . . . 92
6.4.3 Impact of platform model on replication decisions . . . 93
6.5 Recommendations for file replication in VIP on EGI . . . 95
6.6 Conclusion . . . 97

7 Conclusions and perspectives 98
7.1 Contributions . . . 98
7.2 Perspectives . . . 100
7.2.1 Improving the realism of file transfer simulations . . . 100
7.2.2 Improving the realism of the platform description of EGI . . . 101
7.2.3 Extending the current capacities of our simulator . . . 102

Bibliography 115

List of Figures

1.1 The complete life cycle for an application executed via VIP. 8
1.2 Common components in scientific workflows. Figure extracted from [Bharathi *et al.* (2008)]. 9
1.3 Grid model in OptorSim, illustrated from [Bell *et al.* (2003)]. 17
1.4 Multi-tier platform model. 18
1.5 Hybrid platform model. 19
1.6 General graph model. Figure 1.4, 1.5, 1.6 are from [Tos *et al.* (2015)]. . . . 19
1.7 A 3-level network hierarchical model. 20

2.2 GATE workflow including a computing phase, a merging phase, and associated file transfers. 25
2.3 Gantt chart view of 10 transfers from "srm-biomed.gridpp.rl.ac.uk" to "CIEMAT-LCG2". 29
2.4 Gantt chart view of 409 transfers to "CIEMAT-LCG2", including uploads and downloads in one workflow execution. Small points correspond to very short transfer duration. 30
2.5 Distributions of the experienced bandwidth by 11,033 release transfers according to different link categories. 31
2.6 Small local bandwidth for different SEs. Each bar corresponds to the bandwidths experienced by local transfers from a specific SE. 32
2.7 Distribution of release file transfer durations for a given SE (marsedpm.in2p3.fr) in a given workflow. 32
2.8 Gantt chart view of 18 release file transfer durations from the SE "sbgse1.in2p3.fr" to the site "INFN-PISA". Transfers belong to different clusters are in different colors. 33
2.9 Gantt chart view of 5 release file transfer durations for a given SE, a given site, and a given cluster. 34
2.10 Gantt chart view of release file transfer durations for a given SE, a given site, and a given cluster. 34

2.11 Distribution of executed jobs by site in studied workflows. 37
2.12 Cumulative downloads by SEs in studied workflows. 38
2.13 Cumulative number of downloads for SEs versus the number of jobs in the corresponding sites. 39

3.1 The complete life-cycle of the execution of the GATE application on EGI. . 43
3.2 Data management on EGI. Figure extracted from [Löschen and Müller-Pfefferkorn] 44
3.3 The process for file download on EGI. 45
3.4 Graphical representation of hierarchical platform in SimGrid, illustrated from [Bobelin *et al.* (2012)]. 48
3.5 Example of a file catalog. 50
3.6 Three parts of a simulated file transfer in our simulator. 51

4.4 Measured and simulated (with global or grouped average bandwidth for each SE) durations for the download of the release file by each job in the `fNUfzs` workflow instance. 59
4.5 Measured and simulated durations for the downloads of the release file from the `marsedpm.in2p3.fr` SE in the `fNUfzs` workflow instance (top). Simulated times are obtained with a single **average** bandwidth (SE) and distinct **average** bandwidths (SE-Site) for each site (bottom). 60
4.7 Measured and simulated durations for the downloads of the release file from the `marsedpm.in2p3.fr` SE in the `fNUfzs` workflow instance (top). Simulated times are obtained with a single **maximum** bandwidth (SE) and distinct **maximum** bandwidths (SE-Site) for each site (bottom). 61
4.8 Gantt chart view of the transfers of the release file from the `marsedpm.in2p3.fr` SE to worker nodes in the INFN-PISA site. 62
4.9 Measured and simulated durations for the downloads of the release file from the `marsedpm.in2p3.fr` SE in the `fNUfzs` workflow instance with and without correction of the maximum observed bandwidth. 63
4.10 Measured and simulated durations for the downloads of the release file from the `sbgse1.in2p3.fr` SE to the INFN-PISA site in the `LQz3XJ` workflow instance with and without distinction of the clusters. 64
4.12 Graphical summary of measured and simulated (with the 10G-SotA, Average, and Maximum models) file transfer durations (in seconds). 66
4.13 Cumulative Distributed Functions of the absolute logarithmic error achieved by the three platform models over the whole set of transfers of the release files. 67

4.14 Gantt chart view of 5 release file transfer durations for a given SE, a given site, and a given cluster. 69

4.15 Gantt chart view of release file transfer durations for a given SE, a given site, and a given cluster. 69

5.1 Three partial platform descriptions generated from execution traces of workflow 1 (WF1), workflow 2 (WF2), and workflow 3 (WF3). Each line corresponds to the network link between a site and a SE. 72

5.2 The merged platform description from the execution traces of workflow 1 (WF1), workflow 2 (WF2), and workflow 3 (WF3). Red line corresponds to the network link without any information from these traces. 72

5.3 Graphical summary of measured and simulated (with the Average_Merged, and Maximum_Merged models) release file transfer durations (in seconds). Each point corresponds to the duration of one simulated transfer. 75

5.4 Graphical summary of the Mean Absolute Error (MAE) for 20 validation tests of empirical, linear regression, and neural network model. 81

5.5 The validation test which gives us the largest error (0.26) for the neural network model. The solid line corresponds to the **ideal case** where the predicted bandwidth value is equal to the merged bandwidth for a given link. From top to bottom, panels correspond to the predictions by the empirical model (emp_pred) and the neural network model (nn_pred), respectively. . 82

5.6 The validation test which gives us the smallest error (0.034) for the neural network model. The solid line corresponds to the **ideal case** where the predicted bandwidth value is equal to the merged bandwidth for a given link. From top to bottom, panels correspond to the predictions by the empirical model (emp_pred) and the neural network model (nn_pred), respectively. . 83

6.1 Cumulative distribution of simulated file transfer durations with and without dynamic replication. Each line corresponds to a list of 3 SEs used for file pre-staging. The same 50 random prestaging lists are used in all four scenarios. 90

6.2 Cumulative number of downloads for SEs versus the number of jobs in the corresponding sites with and without dynamic replication. 91

6.3 Comparison of random, predefined, and the current production prestaging list without dynamic replication for two platform models 92

6.4 Cumulative distribution of simulated file transfer durations with dynamic replication for two platform models 94

6.5 Cumulative distribution of simulated file transfer durations without dynamic replication for two platform models. Best performance achieved by predefined or randomly selected lists is highlighted. 95
6.6 Comparison of the best and the worst prestaging with the current production prestaging for trace-based model with or without dynamic replication. 95

List of Tables

2.1 Summary of information recorded in traces. 26
2.2 Summary of the number of jobs for 60 workflows. 27
2.3 Statistics on the size of different files in 60 workflows. 27
2.4 Statistics of real transfer durations for different files (in seconds). 28
2.5 Statistics of queuing time for GATE jobs in collected traces. 35
2.6 Statistics of intra-site delay for GATE jobs in collected traces. 36
2.7 Statistics for the number of jobs per site. 36
2.8 Cumulative number of executed jobs in different countries in Biomed VO of EGI in 2017. Statistics extracted from traces and from the Dirac server. 37
2.9 Summary of findings or hypothesis from the analysis of file transfers. 40
2.10 Summary of characteristics extracted from the executions of workflows. . . 41
2.11 Summary of characteristics extracted from the executions of workflows. . . 41

3.1 Strategy for different parameters in simulator. 52

4.1 Statistics of measured and simulated (with the 10G-SotA, Average, and Maximum models) durations (in seconds) of the transfer of the GATE release files (> 121 MB). 66

5.1 Statistics of measured and simulated (with the 10G-SotA, Merged average, and merged maximum models) durations (in seconds) of the transfer of the GATE release files (> 121 MB). 75
5.2 Example of a data set with 4 numerical and 1 categorical variables. 79
5.3 The new data set after transforming categorical variable into dummy variables. 79
5.4 Statistic of MAE for 10 validation tests of neural network and linear regression model. 81

6.1 Cumulative number of different type of transfers with and without dynamic replication for 15 simulated workflows with 50 random prestaging lists. . . . 92
6.2 95%-confidence interval for the statistics of the simulated release transfers durations of 55 prestaging lists with and without dynamic replication 96

Acronyms

DAG Directed Acyclic Graph.

DCI Distributed computing infrastructure.

EGI EGI federated e-Infrastructure.

LFC Logical File Catalog.

SE Storage Element.

VIP Virtual Imaging Platform.

VO Virtual Organization.

Introduction

Distributed computing infrastructures (DCIs) such as grids and clouds are popular instruments for conducting scientific research nowadays. Computing resources from multiple administrative domains are coordinated to be used as a single coherent system. They thus enable scientists to achieve large computing tasks, which could not be executed only by using local resources. Many recent scientific experiments produce large amounts of data that need to be made accessible to large groups of geographically dispersed researchers. For instance, more than 150 Peta bytes of data generated by the ATLAS [Aad *et al.* (2008)] experiment on the Large Hadron Collider (LHC) [Fernandez *et al.* (2012)] are currently stored, distributed, and analyzed in World Wide LHC Computing Grid (WLCG) [Bonacorsi and Ferrari (2007)]. In the last decade, several projects have already demonstrated the ability of DCIs to support heavy scientific computations with a high throughput [Laure and Jones (2009), Romanus *et al.* (2012)].

To facilitate the utilization of the resources in DCIs, high-level interfaces, such as science gateways [Zhao *et al.* (2010)a, Glatard *et al.* (2013), Miller *et al.* (2015)], have emerged to hide the complexity of managing underlying resources and executing applications. Science gateways combine a set of services (e.g., authentication, data management, application deployment, etc) to deliver the computing and storage resources as transparently as possible for researchers to conduct large scientific experiments.

Meanwhile, different optimization strategies [Da Silva *et al.* (2013), Camarasu-Pop *et al.* (2013)c, Amoon (2013), Calheiros and Buyya (2014), Chettaoui and Charrada (2014), Poola *et al.* (2016)] have also been widely investigated for applications deployed on large DCIs to improve their performance and to consolidate their reliability. However, validating the impacts of optimizations for applications deployed on such infrastructures is a complex and challenging task, especially for production infrastructures which are operating platforms providing various services to support scientific research twenty four hours a day. Difficulties come from several aspects:

- real experiments of application execution are extremely time-consuming and costly;
- application executions are almost impossible to be reproduced in production systems due to uncontrollable conditions, e.g., network traffic, background workload;
- large number of validation experiments may harm the utilization of other users.

These aspects bring extra obstacles for researchers or developers to study and investigate the optimization of applications deployments on DCIs.

To cope with these obstacles, simulation has become a widely used method to study applications deployed on large DCIs. Simulation enables researchers to study the execution of applications in a variety of conditions with a complete control for factors which are uncontrollable in real systems. Simulation scenarios are also reproducible and much faster compared to real experiments. Numerous aspects concerning applications executed on DCIs are studied and evaluated by simulation in the literature [Camarasu-Pop *et al.* (2013)a, Mansouri and Dastghaibyfard (2013)a, Dayyani and Khayyambashi (2015), Camarasu-Pop *et al.* (2016), Glatard and Evans (2015), Barisits *et al.* (2016)].

Various simulation toolkits exist [Ostermann *et al.* (2010), Calheiros *et al.* (2011)a, Desprez and Rouzaud-Cornabas (2013), Kliazovich *et al.* (2012), Chen and Deelman (2012), Casanova *et al.* (2014)]. They provide basic functionalities and resource sharing models allowing users to build their own simulators based on these services. Despite the different implementations adopted by each toolkit, they all require an abstraction of several common components to simulate application executions:

- the hardware platform, e.g., the network topology, the space of storage elements, and the computing power of processors, etc.;
- the software services deployed on the platform, e.g., schedulers, file transfer service;
- the application itself, e.g., the workload characteristics.

Each component will have a serious impact on the simulation performance and the realism of simulation behavior for these components will directly decide the reliability of simulation results.

However, the accuracy of these simulation toolkits has rarely been evaluated and the configurations of simulations are often oversimplified in the literature. It may critically question the findings derived from simulation results. Even more badly, studies such as [Velho *et al.* (2013)] have shown that basic predefined models in widely-used simulation toolkits are flawed, which may lead to a false estimation of the simulated metrics.

In this context, the main challenge addressed by this thesis is to improve the realism of simulations targeting the execution of applications with a focus on file transfers on EGI. EGI is one of the largest production DCIs worldwide providing more than 850,000 logical CPUs and 650 PB of disk space in 2018. We evaluate the realism of our simulations based on the ground truth provided by the execution traces offered by VIP, which is a scientific gateway hosting more than 20 medical imaging applications with more than 1,000 registered users.

One possible usage of these simulations is to evaluate file management strategies for applications deployed on large DCIs. In this work, for example, we will use them to

propose reliable recommendations for data placement in VIP. The rest of the manuscript is organized as follows:

Chapter 1 presents the related work and current status in the field of distributed computing, including the existing infrastructures, the deployment of applications on distributed production infrastructures, the simulation tools for studying applications in distributed environments, and the optimization techniques for these applications. Among the numerous optimization strategies for applications in distributed environments, we will focus on file replication which is a widely used technique to optimize the file management.

Chapter 2 presents a thorough analysis of real execution traces of one particular application deployed on EGI. It helps us to derive the network characteristics of the underlying infrastructure and the information on file transfers during the executions of the application. Several issues related to the performance of file transfers are also identified, which allows us to propose relevant optimization to improve file management.

Chapter 3 presents our work to build a realistic simulator to provide a simulated environment as close as possible to the real execution conditions for file transfers during the execution of application deployed on large DCIs. We first identify several fundamental components for file transfers from the real system. Then the choice of simulation tools and the implementation design of the simulator are driven by our concern for the realism in simulated file transfers. This simulator is used to validate our proposed platform model in Chapter 4 and to evaluate file management strategies in Chapter 6.

Chapter 4. In this chapter, we build realistic ad-hoc platform models to replay the executions of the Geant4 Application for Tomographic Emission (GATE) on EGI. Starting from a simplified but widely used platform model, we propose incremental improvements to increase the accuracy of our file transfer simulations thanks to a thorough analysis of trace contents in Chapter 2. The overall improvement of these ad-hoc models is evaluated by confronting simulation results to the ground truth of actual executions registered in the execution traces. The work presented in this chapter were presented in the CCGrid conference [Chai *et al.* (2017)].

Chapter 5 presents our method to construct a complete platform description beyond the ad-hoc models proposed in Chapter 4 by aggregating multiple execution traces. Three predictive models are also proposed and evaluated to fill-in the bandwidth of missing links in the platform after the trace aggregation.

Chapter 6. In this chapter, we cross-evaluate different file management strategies for applications executed in a large DCI by simulations using an enhanced state-of-the-art platform description and the complete platform description built in Chapter 5. Reliable recommendations for improving file management are also derived from simulation results. Results presented in this chapter were presented at HeteroPar workshop in the Euro-Par conference [Chai *et al.* (2018)]. An extension of this work is in preparation for a journal.

Chapter 1

State-of-the-art

Abstract *This chapter presents the state-of-the art in the field of distributed computing, including the underlying infrastructures, the deployment of applications on distributed production infrastructures, the simulation tools for studying applications in distributed environments, and the optimization techniques for these applications. Among the numerous optimization strategies for applications in distributed environments, we focus on file replication which is a widely used technique to optimize data management.*

1.1 Introduction

Large distributed computing infrastructures have been successfully used to support heavy scientific experiments [Fernandez *et al.* (2012), Bird *et al.* (2014), Barisits *et al.* (2017)] over the last decade. Different infrastructures are presented in Section 1.2, including a specific classification for these large distributed infrastructures. In Section 1.3, we discuss different aspects of applications deployed on such infrastructures, including science gateways, scientific workflows, execution traces, and optimization strategies.

In section 1.4, we focus on file replication, which is a widely used technique to optimize data management in large distributed environments. We first present two common classification strategies for numerous file replication methods in the literature. Then we summarize several requirements for applying file replication in production systems. Finally, we investigate several replication strategies implemented for production usages.

Different simulation toolkits and platform models for evaluating file replication are discussed in section 1.5.

1.2 Distributed computing infrastructures

A distributed system is defined as a collection of heterogeneous networked computers which are combined and coordinated to achieve a common goal. Based on different

concepts, models, and technologies, distributed systems can be divided into different categories. Here, the infrastructures of two popular distributed systems are presented: Grid computing and Cloud computing.

1.2.1 Grid computing

Grid computing [Foster *et al.* (2001)] originated in academia in the mid 1990s with the popularity of the Internet and the availability of powerful computers and high-speed network technologies. It aimed at facilitating users to remotely utilize idle computing power within other computing centers when the local one is busy. One of the main strategies of grid computing is to use middleware to divide and apportion pieces of a program among several computers. The size of a grid may vary from computer workstations coordinated by network to large collaborations across many computing sites and data centers in different continents.

We can categorize grids into research and production infrastructures. Research infrastructures are experimental testbeds designed to conduct controllable experiments, usually in academia. Examples of such infrastructures are Emulab [Siaterlis *et al.* (2013)], PlanetLab [Kim *et al.* (2011)], Grid'5000 [Balouek *et al.* (2012)], and Future Grid [Von Laszewski *et al.* (2010)]. Production infrastructures, on the other hand, are operating platforms providing various services to support computing and data sharing for production purposes twenty four hours a day.

Grids can be further classified into High Performance Computing (HPC) and High Throughput Computing (HTC) infrastructures. HPC systems focus on the efficient execution of tightly-coupled tasks, while HTC systems focus on the efficient execution of a large number of independent tasks.

The Extreme Science and Engineering Digital Environment (XSEDE) [Towns *et al.* (2014)] and the Partnership for Advanced Computing in Europe (PRACE) [prace project] are the main HPC infrastructures in production. XSEDE is a single virtual system that scientists can use to interactively share computing resources, data, and expertise. A wide range of software is available on various XSEDE resources supporting diverse fields of study. PRACE aims at developing a distributed HPC infrastructure based on the national supercomputing centers in Europe.

The Open Science Grid (OSG) [Juve *et al.* (2013)] is a production grid used in the United States. It consists of computing and storage elements at over 100 individual sites spanning the United States. In Europe, one of the largest HTC production infrastructures is the EGI e-infrastructure [Kranzlmuller (2009)], which provides advanced computing services for research and innovation. EGI coordinates computing, storage, and network resources over 350 resource centers across more than 50 countries. In EGI, a set of computing resources (e.g., clusters, workstations, etc.) localized at a site is defined as a Computing

Element (CE) while a Storage Element (SE) provides uniform access to a set of data storage resources grouped together. In general, each CE defines a local SE which is either within or very close to the site. EGI currently hosts more than 200 Virtual Organizations (VOs) for communities with interests as diverse as Earth Science, Computer Science, Mathematics, Life Sciences, or High-Energy Physics. It provides more than 850,000 logical CPUs and 650 PB of disk space in 2018.

1.2.2 Cloud computing

In the last decade, cloud computing has become a buzzword. However, cloud computing is not a completely new concept. It is a specialized distributed computing paradigm and has intricate connections to grid computing. One definition for cloud computing can be found in [Foster *et al.* (2008)]:

> *A large-scale distributed computing paradigm that is driven by economies of scale, in which a pool of abstracted, virtualized, dynamically-scalable, managed computing power, storage, platforms, and services are delivered on demand to external customers over the Internet.*

A more detailed comparison between cloud and grid computing can be found in [Foster *et al.* (2008), Sadashiv and Kumar (2011)].

Similar to grid computing, cloud computing systems can be also categorized into production and research infrastructures. The Amazon Elastic Compute Cloud (EC2) [amazon ec2] is currently the most used cloud computing infrastructure. It provides Infrastructure as a Service (IaaS) at a scale that can accommodate different distributed production infrastructures. It also enables users to increase or reduce the number of virtual machines needed and charges them according to the size of the instances and the capacity used. Other commercial cloud computing infrastructures in production can be found in industry, e.g., Windows Azure cloud computing platform [windows azure] and Google Cloud platform [google cloud].

Cloud testbeds are also constructed for research purposes. A few examples of such infrastructures are the Virtual Computing Lab (VCL) [Averitt *et al.* (2007)], the Open Cloud testbed [Grossman *et al.* (2009)], the OpenCirrus [Avetisyan *et al.* (2010)], and the Chameleon cloud testbed [Mambretti *et al.* (2015)]. They enable researchers to test or validate new designs for cloud computing in controllable environments.

1.3 Application deployment in distributed systems

Although distributed computing infrastructures provide large amounts of computing and storage capacities, using them can sometimes be a complex work for scientists and

researchers. In this context, science gateways are emerging as high-level interfaces for users to facilitate the access to distributed infrastructures. One of their main features is allowing users to describe applications as abstract workflows. In this section, we present the state-of-the-art of existing science gateways and different widely used workflow models.

1.3.1 Science gateways

The concept of a science gateway is a community-specific set of tools, applications, and data collections that are integrated together via a web portal or a suite of applications, providing access to the computing and storage resources of different infrastructures. It combines a set of services, e.g., authentication, file transfer, and workload management tools, to deliver computing power as transparently as possible. Its responsibilities consist in monitoring the status of running experiments, killing misbehaving executions, and taking decisions to optimize the performance of applications on behalf of users. Science-gateways can be found in various scientific domains, such as climate, life-science, and medical imaging.

The Community Climate System Model (CCSM) portal [Zhao *et al.* (2010)a] provides a one-stop shop for creating, configuring, and running CCSM simulations as well as managing jobs and processing output data by using TeraGrid high performance computing resources. CIPRES [Miller *et al.* (2015)] is a web portal designed to provide researchers with transparent access to the fastest available community codes for inference of phylogenetic relationships. It allows more than 1,800 unique users to run jobs that required 2.5 million CPU hours. In the life-science field, WeNMR [Wassenaar *et al.* (2012)] is developed for bio-informatics applications. With over 450 registered users, WeNMR is one of the largest VO in the life-science community officially recognized by EGI.

VIP [Glatard *et al.* (2013)] is a widely used science gateway in the field of medical imaging. It currently hosts more than 20 medical imaging applications with more than 1,000 registered users. VIP is supported by the Biomed VO of EGI, which has access to about 65 computing sites world-wide, including more than 130 computing clusters and 5PB of storage distributed across 50 different SEs. The complete life cycle for the execution of an application in VIP is depicted in Figure 1.1. From the web portal, users are authenticated by their login and password (step 0). They can easily upload their input data onto the storage resources in the Biomed VO via VIP (step 1) and select the application that they want to execute (step 3). Each application integrated into VIP is described as a workflow. The workflow description is interpreted by the MOTEUR [Glatard *et al.* (2008)] workflow engine to generate jobs (step 4) that are submitted and scheduled to EGI by the DIRAC [Tsaregorodtsev *et al.* (2010)] workload management system (step 5-7). When the required resources are allocated, jobs will first download input files for the application and then execute the computing workload (step 8-9). After the execution,

results are automatically stored on a storage resource (step 10) and made available to VIP users through the web portal.

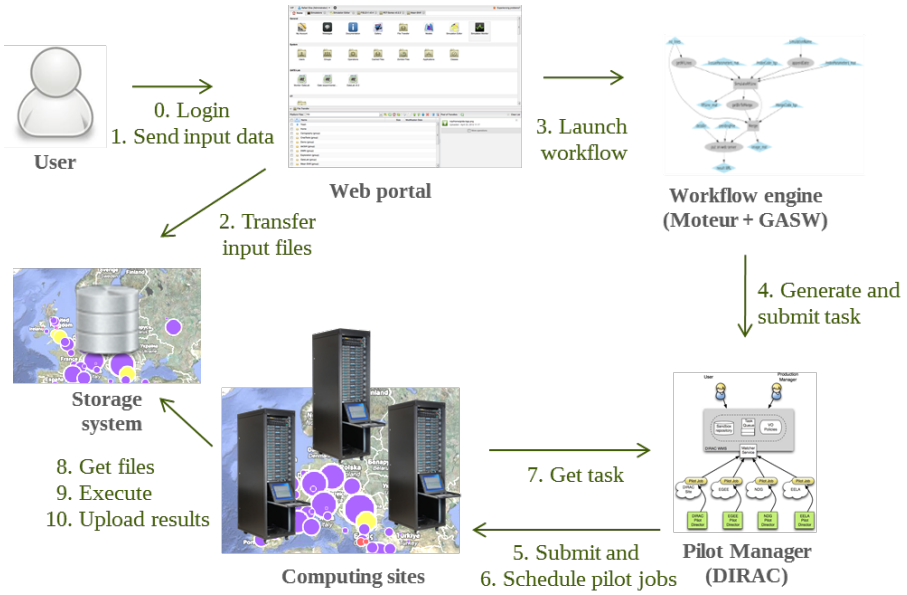

Figure 1.1: The complete life cycle for an application executed via VIP.

1.3.2 Scientific workflows

Applications deployed on large distributed infrastructures via science gateways are usually expressed as workflows. Scientific workflows are often abstracted as a Directed Acyclic Graph (DAG) to combine a series of phases, including data movement, analysis, computation, and final result combination. In DAG-based workflows, nodes represent computational jobs and edges represent data or logical dependencies between jobs.

For scientific workflows, different models exist depending on the design concept and the purposes of applications. Some basic structures or common components of them are illustrated in Figure 1.2. The simplest process for a workflow is a job analyzing input data and producing a result. This process can then be used as a basic component to construct a pipeline model, where each job requires the output data of its previous job to obtain the final result. Then depending on whether the workflow scatters or gathers data, we can distinguish three more basic models: data distribution model, data aggregation model, and data redistribution model.

In VIP, several workflows correspond to simple processes. This is the case for certain neuroimaging tools based on FSL [Smith *et al.*] and Freesurfer [Fischl (2012)], as well as for some tools developed at the CREATIS research lab, such as "RF Coil Characterization" [1] or "Super Resolution" [Van Reeth *et al.* (2015)]. Nevertheless, some VIP

[1] https://vip.creatis.insa-lyon.fr/documentation/mri_charact.html

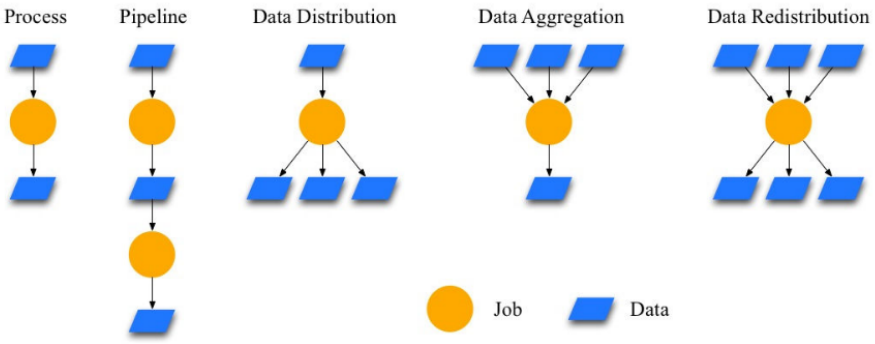

Figure 1.2: Common components in scientific workflows. Figure extracted from [Bharathi *et al.* (2008)].

applications are described with complex workflow models, allowing to manage data distribution, aggregation, loops and conditions. This is the case for the workflow describing the GATE application [Jan *et al.* (2004)]. GATE is a Geant4-based open-source software to perform nuclear medicine simulations, especially for TEP and SPECT imaging, as well for radiation therapy. As a Monte-Carlo simulator, one GATE execution can be easily split into multiple jobs. Each job produces a partial result, which is then merged into a final output. Therefore, the GATE workflow is a combination of the process and the data aggregation model. The GATE workflow can be deployed on EGI by two possible approaches: static and dynamic partitioning. The number of jobs for each execution will adapt to the computing duration estimated by users.

In this thesis, we will focus on the workflow of the GATE application, which is one of the most successful applications in VIP. More detail of the deployment of GATE workflow on EGI will be presented in Chapter 2.

1.3.3 Traces of workflow executions

During the execution of scientific workflows, diverse real-time metrics are monitored and registered by science gateways. These metrics are then generated and stored as execution traces. Such execution traces are important data sources for conducting off-line research work. They provide researchers rich information on the executed jobs to identify the characteristics of workflows, the computing and network resources to model the underlying platforms, and the resource utilization to investigate failure causes or predict the performance of workflow executions.

Numerous research works in distributed environments were based on trace analysis. For instance, authors in [Barisits *et al.* (2016)] proposed a hybrid simulation model for data grids based on the statistical analysis of traces. [Ferreira da Silva *et al.* (2015)] presented a practical machine learning method to predict job characteristics by conducting an analysis on workload traces. Authors in [Reiss *et al.* (2012)] studied the heterogeneity

and the dynamicity of the resource usage of Cloud by analyzing publicly available Google trace data. These traces consists of the monitored execution of applications over a month executed in ∼12,000 machine clusters. In [Iosup *et al.* (2011)], authors analyzed long-term performance traces from the two largest commercial clouds: Amazon cloud and Google cloud. They found that yearly and daily patterns were followed by half of the cloud services and the impact of the observed variability was assessed by trace-based simulation. In [Javadi *et al.* (2013)], a failure trace archive was created and a comparative analysis of failures in various distributed systems was presented. Authors also emphasized that different interpretations of the meaning of failure data can lead to different conclusions for failure modeling and job scheduling in distributed systems.

In this thesis, we also adopt the trace-analyzing method. The execution traces generated by VIP are collected to study the GATE application executed on EGI. By analyzing these traces, we are able to characterize file transfers, to model the hardware platform for the Biomed VO in EGI, and to identify the current issues for the application performance or the resource usages in VIP. The analysis of execution traces of the GATE application will be presented in Chapter 2.

1.3.4 Optimization of workflow execution

Large production distributed infrastructures, which coordinate large amounts of geographically distributed resources, can be more prone to errors than traditional HPC clusters or experimental testbeds [Montagnat *et al.* (2010), Alsoghayer and Djemame (2014), Carrión *et al.* (2015)]. In [Kondo *et al.* (2010)], authors report that Grids have a yearly resource availability of 70% or less. In [Ma *et al.* (2013)], authors mention that EGI has an average availability and reliability ranging from 84% to 96% for resources. It implies that improving the reliability of applications is as important as improving the performance and that the reliability is directly relevant to the performance of applications executed in production grids [Iosup *et al.* (2006)].

Different aspects of applications deployed on distributed environments can be optimized to improve the performance, to consolidate the reliability of workflow executions, or both. At application level, optimizations often concern two aspects: jobs and data.

Numerous strategies were proposed to provide fault tolerance for jobs, for instance, job checkpointing [Nazir *et al.* (2009), Chtepen *et al.* (2009), Cao *et al.* (2010), Amoon (2013)] and job replication [Ben-Yehuda *et al.* (2012), Calheiros and Buyya (2014), Poola *et al.* (2016)]. Checkpointing is a mechanism allowing to save the state of a running job so that the job can be resumed from the registered state in case of any fault. It prevents restarting the execution of applications from the very beginning and therefore can reduce the whole execution time of applications. Job replication consists in dispatching multiple replicas of a job across different resources and using the result from the first instance to complete.

Job replication is based on the assumption that the probability of a single resource failure is much higher than that of a simultaneous failure of multiple resources in distributed systems. It can thus achieve good performance even in the absence of information on computing resources.

Meanwhile, file replication [Vrbsky *et al.* (2010), Yang *et al.* (2010), Bsoul *et al.* (2011), Andronikou *et al.* (2012), Chettaoui and Charrada (2014)] is the most widely used technique to optimize data management in distributed systems. It consists in replicating the same file on different storage resources. This allows to avoid one-point failure of data, thereby increasing data availability and fault tolerance. It can also ease the file transfer burden on the network between one computing site and one storage resource, therefore reducing data access latency and file transfer duration.

The optimization of jobs for applications deployed on EGI via VIP has already been studied and investigated. Authors proposed a self-healing mechanism to handle operational incidents through job replication in [Da Silva *et al.* (2013)]. The added value of adopting checkpointing in workflow executions has been evaluated by conducting real experiments on EGI in [Camarasu-Pop *et al.* (2013)c]. Following these works, we focus in this thesis on improving the data management for applications in VIP by file replication.

1.4 File replication

File management is a key component in large distributed environments. Efficient file transfers are critical to the performance of data-intensive applications, since a long file transfer may badly delay a given job during the execution of a workflow, change the schedule of subsequent jobs, and therefore impact the whole application execution time. Numerous file replication strategies were proposed to optimize file management in distributed systems and their implementations vastly vary depending on their optimizing metrics.

1.4.1 Classification for file replication

While replication strategies can be quite different, they have common features with respect to certain aspects. Different classification schemes for replication strategies were proposed in the literature.

Static vs. dynamic replication

The most general classification is static versus dynamic replication. In static replication [Ranganathan and Foster (2003), Loukopoulos and Ahmad (2004), Chervenak *et al.* (2007), Xiong *et al.* (2013)], decisions regarding to the replication strategy are made before launching the application and do not change during the execution. Static replication will

not add any decision and management overhead during the execution of applications and it can be a good choice for non-changing environments as static replication strategies are usually simple to implement. However, they are often inefficient in a dynamic environment such as large production grids.

On the other hand, decisions in dynamic replication [Chang and Chang (2008), Hanandeh *et al.* (2012), Mansouri and Dastghaibyfard (2013)b, Vashisht *et al.* (2014)] would adapt to changes of systems, e.g., storage capacity or network bandwidth. Replicas can be created on new nodes during the execution of an application and can be deleted when they are no longer required. Dynamic replication strategies often rely on information obtained at runtime, hence adding an extra overhead to the application execution time.

Centralized vs. decentralized replication

Another classification scheme for replication strategies is centralized versus decentralized. For centralized strategies [Wu *et al.* (2008), Pérez *et al.* (2010), Zhao *et al.* (2010)b, Sashi and Thanamani (2011), Andronikou *et al.* (2012)], a central authority is implemented to control all the aspects of data replication. It collects global metrics on the underlying systems and then propagates these information to different nodes. However, this central authority can easily become a single point of failure and a bottleneck when systems scale up.

In decentralized replication strategies [Abdullah *et al.* (2008), Mansouri and Dastghaibyfard (2012), Mansouri and Dastghaibyfard (2013)a, Chettaoui and Charrada (2014)], there is no central control mechanism. Hence, no single node can possess a complete view about the entirety of systems. Nodes need to take their own decisions regarding to replication. Decentralized replication is good for reliability and scalability as there is no single point of failure in the system but it may lead to excessive replications as no global information about the systems is monitored.

1.4.2 Requirements for file replication in production

Although numerous file replication strategies were proposed in the literature, they are rarely applied and implemented in real production infrastructures. Authors have already highlighted the gap between theoretical research and the implementation of replication strategies in production in [Ma *et al.* (2013)]. We summarize several restrictions hampering the integration of replication strategies, especially the automated and dynamic replication strategies, in large distributed production infrastructures.

Diverse requirements. Large production distributed infrastructures often support the execution of applications from diverse scientific fields. For instance, EGI distinguishes more than 200 VOs for communities with interests as diverse as Earth Science, Computer

Science, Mathematics, Life Sciences, or High-Energy Physics. Each community has its different applications with different characteristics and file access patterns. It is thus very difficult to provide a general and efficient file replication method to meet the requirements of all applications or users at the middleware level. Hence, most of the replication management is delegated to applications and most decisions are made by administrators or applications with an application-oriented strategy.

Non-clairvoyance. The information about where jobs will be executed, what is the current usage of storage resources, or the network throughput cannot (or are very hard to) be foreseen before execution time in production infrastructures. However, most theoretical works assume a complete clairvoyance about compute, storage, and network resources in the proposed strategies. At the application/user level, the middleware is considered as a black-box and only services at application-level are available to users. Even though certain services allow users to collect and monitor the resource usages in production infrastructures, it will add large extra overhead to the execution time of applications if the replication decisions strongly depend on such real-time information. Therefore, an applicable file replication strategy for production infrastructures should rely on as few information as possible about the system (e.g., network throughput, job queuing time, job execution time, etc.).

Replication cost. In large production infrastructures, storage resources are often shared by different groups of users and the available space is limited for each group. The number of replicas is thus limited for a given file [Ramakrishnan *et al.* (2007)]. However, most theoretical works only focus on reducing the file access time in order to increase the performance of file transfers. As a consequence, many theoretical studies favor strategies that always replicate [Ranganathan *et al.* (2002), Doğan (2009)] or create as many replicas as possible [Park *et al.* (2003), Sashi and Thanamani (2011)]. It badly questions the applicability of these strategies in production. Another cost related to increasing the number of replicas is the potential risk of increasing uncertainty and inability for the execution of an application, since large distributed production systems are prone to failures. A brittleness entropy metric was introduced in [Ma *et al.* (2015)] to describe the risk associated to file replication on unreliable SEs. Results show that limiting the number of storage elements involved in the execution of an application improves the execution reliability on EGI. Therefore, an applicable replication strategy should also take into account the replication costs by limiting the number of replicas.

1.4.3 Existing replication management in production systems

In this section, in addition to the replication management implemented in VIP, we investigate two other replication strategies that are adopted for production usage in large distributed infrastructures.

Replication in Pegasus

Pegasus [Deelman *et al.* (2015)] is a workflow management system, which enables user to map abstract workflow descriptions onto large distributed computing infrastructures. It also offers data management and job monitoring subsystems for workflow executions. In Pegasus, different data management services are proposed depending on the target cyber-infrastructures on which to execute workflows. Here, we only focus on the replication services provided by the data management system in Pegasus for computing grids.

Pegasus assumes that the datasets required by workflows have already been distributed across the infrastructure and the replica locations are registered in a Replica Catalog. Hence, no additional services related to replica creation are provided. Before the execution of a workflow, each job is explicitly mapped to the candidate execution sites specified by the user. This process is called *Site Selection* in Pegasus. Several site selection strategies are supported in Pegasus, for instance, random, round-robin, or Heterogeneous Earliest Finish Time (HEFT) [Topcuoglu *et al.* (2002)]. After this site selection process, Pegasus will have the complete information about where jobs will be mapped before the execution. It thus enables a special job (called *stage-in* job) to transfer the datasets required by the workflow from the locations specified in the Replica Catalog to storage resources close to the execution sites. If multiple locations are available for the same dataset, a variety of replica selection strategies are offered by Pegasus such as preferring the location that has a good bandwidth to the execution site, randomly selecting a replica, or using a user provided ranking. Due to site selection and data stage-in, jobs do not need to download the required files from remote sites during the execution, which can reduce the execution time of workflows.

Replication in Rucio

Rucio [Garonne *et al.* (2014)] is a distributed data management system implemented to support the ATLAS experiment [Aad *et al.* (2008)] which is one of the largest experiments at the Large Hadron Collider (LHC) at the European Organization for Nuclear Research (CERN). It manages the data produced by ATLAS with more than 150 Petabytes spread worldwide across 130 sites and provides users with the functionalities for data placement, data replication, and data deletion.

Replica creation in Rucio is based on user-defined replication rules for files. A replication rule may define the minimum number of replicas to be available on a list of SEs. It allows users to express their intention behind the replication request instead of defining a specific destination for replicating data. For instance, a user can replicate a file two times in France by defining a rule such as "copies=2&country=fr". Rucio will then select the appropriate two SEs in France depending on the current resource usages, such as the

available storage spaces or network bandwidth. Users can also associate a weight to SEs so the replica selection algorithm will choose the SE with respect to this value. Rucio offers various functionalities to ease the creation of file replicas. However, the decision of where to replicate is still the responsibility of users.

This rule-based replica creation strategy enables to spread data over the system to make them available for users. Besides this static strategy, a dynamic data placement strategy has recently been proposed and tested in a pre-production mode for ATLAS [Beermann *et al.* (2017)]. In this strategy, information from different sources are collected to decide if and where to create a new replica, e.g., the current available replicas for a given file from the Rucio database, bandwidth information from perfSonar [Tierney *et al.* (2009)], and file popularity (i.e., the daily access numbers for a file). This dynamic strategy consists in two main phases: deciding whether to create a new replica and where to create. The decision process for whether creating a new replica for a given file is described below:

- Check if a replica has already been created for this file in the past 24 hours. If yes, no new replica will be created; otherwise, continue.
- Check how many replicas already exist. If more than 4, no new replica; otherwise, continue.
- Check the popularity of the file in the last 7 days. If it has not been popular (i.e., the number of accesses is lower than a configurable threshold), no new replica; otherwise, create a new replica.

If all the requirements are met by the target file, a new replica will be created. The decision about where to replicate is based on site ranking:

- Check the network bandwidth for links between sites having existing replica and other sites.
- The possible destination sites are ranked based on available storage space, bandwidth, and the number of queuing transfer requests.
- Sites are down-ranked if a replica has been recently created there.

Then a new replica will be created for the site with highest rank if it has enough storage space for the target file. Recent results [Maier *et al.* (2018)] have shown that adopting this dynamic strategy can improve the data availability and lead to a better usage of the available disk space. Moreover, the completion time of a fraction of jobs is reduced after adopting this dynamic strategy.

Replication in VIP

On EGI, the Unified Middleware Distribution (UMD) [David *et al.* (2014)] is an integrated set of software components packaged for deployment as production services. Among them, the data management services allow users to upload files onto a SE, then replicate and register them in a File Catalog. However, the decisions about where to replicate files and how many replicas to create are left to the applications (users). During the execution of an application, jobs can use the replica selection service offered by the UMD to select file replicas according to their distance to the computing site, of which the algorithm will be detailed in Chapter 3.

The replica creation strategy currently implemented in VIP adopts a static strategy. The required data for a given application are asynchronously replicated to several remote sites before the application execution. This process is named *file prestaging*, which have been demonstrated to significantly reduce the execution time of applications deployed on large distributed infrastructures [Ranganathan and Foster (2003), Chervenak *et al.* (2007)]. However, the decision of where to prestage files relies on the experience and *a priori* knowledge of its administrators. For most of applications hosted by VIP, input files are automatically replicated to a static predefined list of 3 to 5 SEs chosen among the ones considered as stable, with a general good network connectivity, and sufficiently large amounts of available storage space (generally at least 500 GB). This list is updated when one of the SEs needs to be replaced, is in downtime, is full, or faces any other issue preventing its usage. The number of replicas also varies depending on the type and size of the files. Files larger than 500MB are usually replicated on the most available SEs.

1.5 Simulation

Large distributed infrastructures, such as Grids and Clouds, are complex, dynamic, and heterogeneous environments. Therefore, it is difficult to evaluate new prototypes or new strategies (e.g., scheduling algorithms or data management strategies) in a repeatable and controlled manner. Besides, a full-scale evaluation by real experiences implies interference with on-going executions, which is not encouraged in a production environment. Simulation is thus mandatory to test and evaluate complex scenarios for large distributed production infrastructures.

1.5.1 Simulation tools

Many simulators have been developed in the era of Grids construction. They helped researchers to test the performance of the design of middleware services, evaluate job scheduling algorithms, or assess new data management strategies. For instance, Optor-

Sim [Bell *et al.* (2003)] and ChicSim [Ranganathan and Foster (2002)] were designed to study file replication. In ChicSim, a grid model with 30 sites and two theoretical bandwidth values were used, while a model of The European DataGrid Testbed was adopted in OptorSim, which is shown in Figure 1.3. However, the development of both projects has been discontinued.

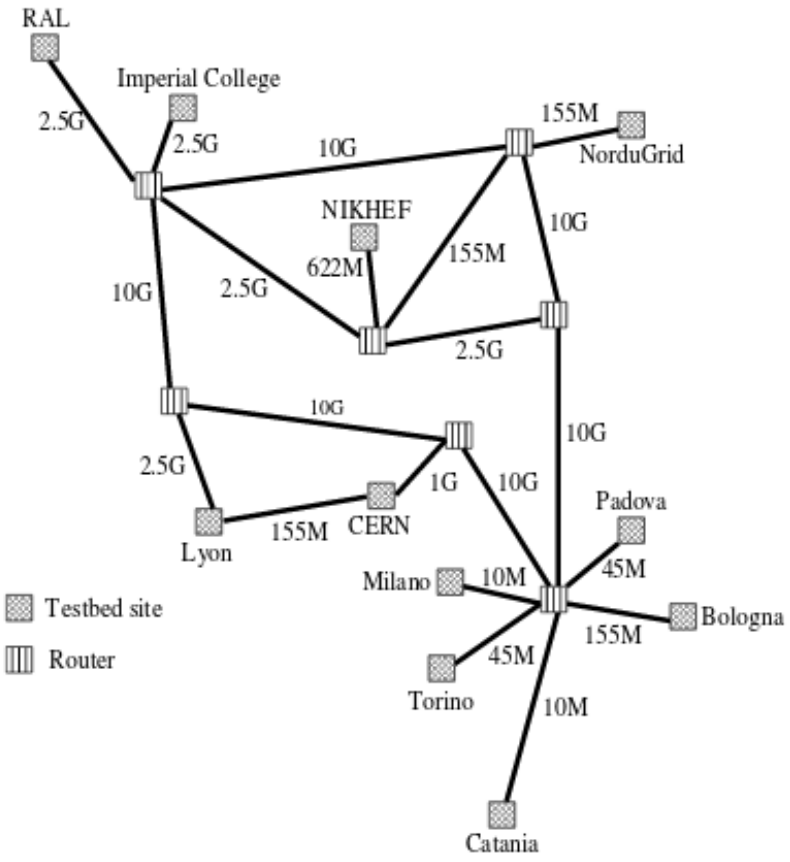

Figure 1.3: Grid model in OptorSim, illustrated from [Bell *et al.* (2003)].

SimGrid [Casanova *et al.* (2014)] and GridSim [Buyya *et al.* (2011)] are two simulation toolkits widely used in grid computing research. GridSim is a Java-based discrete-event simulation toolkit based on SimJava [Howell and McNab (1998)]. It offers the modeling of heterogeneous computational resources, different policies for job scheduling (e.g., time or space shared policy), and different network services. A variable size packet-level model is used for network communications. However, this packet-level model in GridSim has been shown to be false and inaccurate [Velho *et al.* (2013)].

The SimGrid toolkit provides various core functionalities for simulating distributed applications in heterogeneous distributed environments. It is based on fast and accurate fluid models for simulating network communication. This toolkit has been continuously developed for about 20 years. Storage simulation capacities [Lebre *et al.* (2015)] have been recently integrated into SimGrid. Besides, the validity of its analytical models was

widely investigated in [Velho and Legrand (2009), Velho *et al.* (2013)]. SimGrid can thus be conceived as a scientific instrument, which can be used as a reliable tool to simulate applications in distributed environments.

With the growth of popularity for cloud computing paradigm, numerous simulators have been proposed for cloud systems and applications. Many of cloud simulators are based on existing simulation toolkits. For instance, the Amazon Cloud simulator [Desprez and Rouzaud-Cornabas (2013)] is implemented on top of SimGrid. The APIs of AWS cloud (EC2 and S3) are replicated in this simulator. It allows users to evaluate executions of applications and resource provisioning algorithms. CloudSim [Calheiros *et al.* (2011)a] is a simulation tool built on the core engine of GridSim. It provides features to model data center, various instance types of virtual machines, different brokering policies, and the pay-as-you-go model. GreenCloud [Kliazovich *et al.* (2012)] is a simulator designed for energy-aware cloud computing systems. It is a packet-level simulator based on the network simulator ns-2 [Issariyakul and Hossain (2012)]. The energy consumption for each component of a data center (e.g., link, switch, gateway) and the consumption of network communications can be calculated in GreenCloud.

1.5.2 Platform models for file replication

Almost all the work done in the field of file replication for large distributed infrastructures has been evaluated and validated through simulation. OptorSim has been used for file replication studies in numerous works [Tu *et al.* (2008), REN *et al.* (2010), Challal and Bouabana-Tebibel (2010), Zhong *et al.* (2010), Mansouri and Dastghaibyfard (2012), Bai *et al.* (2013), Cui *et al.* (2015)] and these simulations are usually based on the platform model of the European DataGrid Testbed, which is shown in Figure 1.3. However, [Grace and Manimegalai (2014)] mention that it is very difficult to write configuration files and modify the parameters adapted to the requirements of users in OptorSim.

Besides the work done with OptorSim, three more platform models are widely used in the literature. The first one is multi-tier model, which is shown in Figure 1.4.

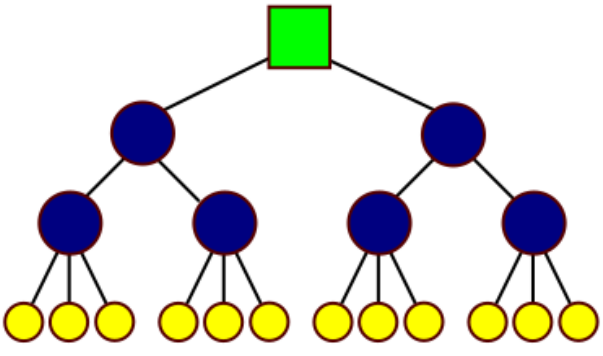

Figure 1.4: Multi-tier platform model.

This model was proposed in the GriPhyN project [Ranganathan and Foster (2001)]. The problem in this tree-like-structure is that a child node can only communicate with its immediate parent but not with other nodes, which is an invalid assumption for real grids.

The second one is called hybrid or sibling tree model [Lamehamedi *et al.* (2002)], which is shown in Figure 1.5. It extends the multi-tier model, allowing the communication among nodes in the same tier (siblings). Although this model can address some limitations of the multi-tier model, it still can hardly represent real production grids, in which there is no central node designated as a root node, and any node can be connected with any other node.

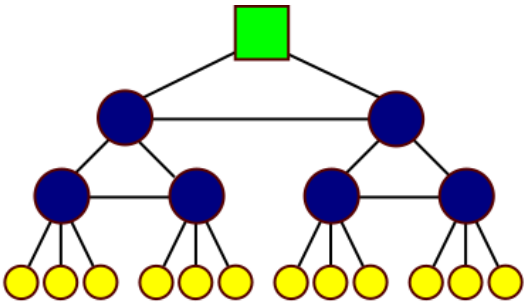

Figure 1.5: Hybrid platform model.

The last model is the general graph model, which is shown in Figure 1.6. A general graph model represents a complete network model, in which any node can be connected to any other node without any restriction as other models.

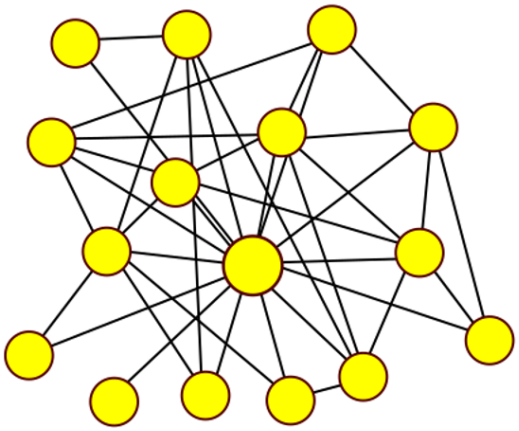

Figure 1.6: General graph model. Figure 1.4, 1.5, 1.6 are from [Tos *et al.* (2015)].

A 3-level network hierarchical model is the most used general graph model, in which the bandwidth connectivity increases from inter-region links to inter-LAN links, and to intra-LAN links [Park *et al.* (2003),Horri *et al.* (2008),Shorfuzzaman *et al.* (2010),Mansouri and Dastghaibyfard (2013)a,Dayyani and Khayyambashi (2015)]. As illustrated in Figure 1.7, nodes in local area are connected by intra-LAN links and then local areas are connected

by inter-LAN links. Finally, all nodes inside a network region are connected to another regions by inter-region links.

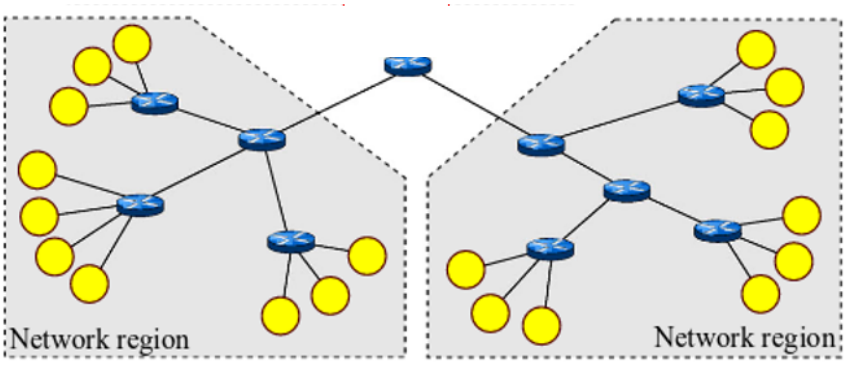

Figure 1.7: A 3-level network hierarchical model.

However, the configurations of platforms are often oversimplified. Almost all the simulation evaluations for replication strategies based on the 3-level model use a unique theoretical bandwidth for each link category. For instance, the configuration for bandwidth is 1GB/s for intra-LAN links, 100MB/s for inter-LAN links, and 10MB/s for inter-region links in [Horri *et al.* (2008)]; 1Gb/s for intra-LAN link, 100Mb/s for inter-LAN, and 10Mb/s for inter-region links in [Dayyani and Khayyambashi (2015)]. Although this 3-level network hierarchical model addresses the limitations of the multi-tier and hybrid models by adopting a general graph representation, we feel that the oversimplified configurations used in the literature can hardly capture the characteristics of heterogeneous production distributed infrastructures such as EGI. To the best of our knowledge, there exists no full and faithful description of EGI that has been built for simulation purposes. This is mainly due to the lack of detailed information on the configuration (number and performance of processing units, intra-site bandwidth, etc.) of its more than 350 distributed sites across 50 countries and on the inter-site network connections.

1.6 Conclusion

In this chapter, we presented the related work on applications deployed on large distributed systems, from the underlying infrastructures to the high-level optimization aspects. File management is one critical component influencing not only the performance but also the reliability of application executions in such large distributed environments. File replication is thus widely studied to optimize data management in the literature. Due to numerous restrictions and drawbacks of real experiences, the evaluation and validation of file replication strategies are usually done with the help of simulation. However, the platform models used in simulation are often oversimplified, which may lead to a questionable applicability for real production systems. Besides, we also identified the gap between the

theoretical research and the implementation in production of replication strategies, especially for dynamic replication strategies. Two replication strategies adopted in production systems have also been investigated.

Chapter 2

Analyzing execution traces of one application deployed on a large distributed system

Abstract *This chapter presents a thorough analysis of the execution traces of one particular application deployed on EGI, the largest European distributed infrastructure. It helps us to derive the network characteristics of the underlying infrastructure and information on file transfers during the execution of the application. Several issues related to the performance of file transfers are also identified, which allows us to propose relevant optimizations to improve file management for a real system such as VIP.*

2.1 Introduction

Trace analysis is a common approach in research on distributed systems. Fine-grained information can be found in execution traces, e.g., execution time line of jobs, resource utilization, or distribution of jobs. They can be used to identify workflow characteristics [Ostermann *et al.* (2008)a, Ostermann *et al.* (2008)b, Iosup and Epema (2011)], model the underlying platforms [Calheiros *et al.* (2011)b, Barisits *et al.* (2016)], predict job characteristics [Ferreira da Silva *et al.* (2015)], analyze performance variability [Iosup *et al.* (2011)], and investigate failure issues [Kondo *et al.* (2010), Javadi *et al.* (2013)].

As we focus on file management in this thesis, it is fundamental to understand the characteristics of file transfers and to identify the current issues concerning their performance on a production platform. In this chapter, we present our work on the analysis of the execution traces of one application deployed on EGI via VIP.

From the collected traces, we extract the characteristics of file transfers by analyzing the distribution of their duration. Understanding why there exits a large variability in

transfer durations allows us to derive the characteristics of the network topology of EGI. Besides, we also investigate other characteristics of workflow executions (i.e., queuing time, distribution of jobs, and the number of downloads of SEs.). These aspects have indeed a direct or indirect impact on file transfers.

Then in this chapter, the questions we want to address are:

- What knowledge of the network topology of EGI can be extracted from traces?
- What are the reasons for the observed large variability in file transfer durations?
- What knowledge can we extract from the information on job executions?
- What are the current issues that we can improve regarding file management in VIP?

The rest of the chapter is organized as follows. In Section 2.2, we first present the detail of the GATE application and its deployment on EGI. In Section 2.3, we present the collected execution traces of the GATE workflow and do a first analysis of the workflows, e.g., the size of downloaded files, the number of jobs, etc. File transfer durations are analyzed in Section 2.4 and different aspects of workflow executions influencing file transfer are studied in Section 2.5.

2.2 GATE application

The Geant4 Application for Tomographic Emission (GATE) is an advanced open-source software dedicated to Monte-Carlo simulations which plays an increasing role in medical imaging and radiotherapy research. GATE[1] currently supports simulations of Positron Emission Tomography (PET), Computed Tomography (CT), Optical Imaging, and Radiotherapy experiments. For these experiments, GATE can be used to help design and assess new imaging devices, and to optimize the acquisition and data processing protocols.

The simulation in a particle tracking Monte-Carlo application consists in the successive stochastic tracking through matter of a large set of individual particles and each particle has an initial set of properties (type, location, direction, energy, etc.). The simulation accuracy of Monte-Carlo approaches such as GATE depends on the number of simulated particles. However, more simulated particles means longer computing time. Therefore, typical radiotherapy simulations would take days or even weeks to complete using personal computers. Figure 2.1 illustrates a 3D whole-body F18-FDG PET scan simulated with GATE. This simulation requires approximately 4,000 CPU hours, which means that it will take about 5.3 months to finish by using one single CPU in a personal computer.

[1] http://www.opengatecollaboration.org/home

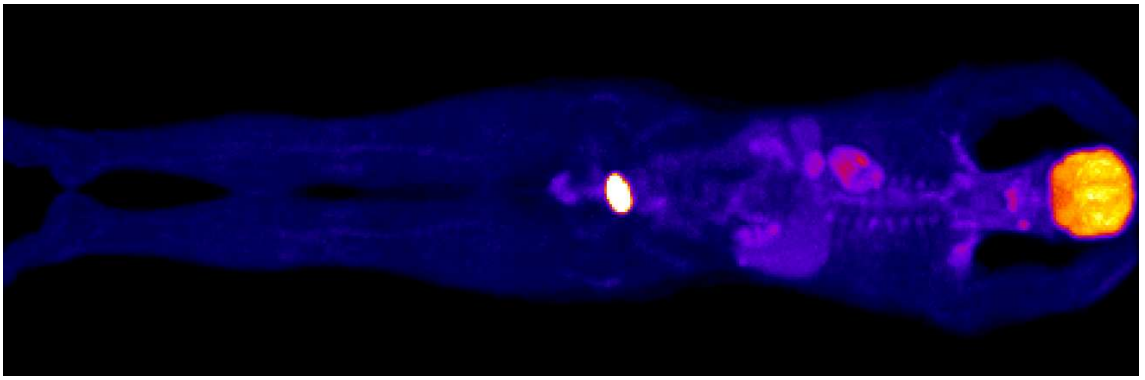

Figure 2.1: 3D whole-body F18-FDG PET scan simulated with GATE, representing approximately 4,000 CPU hours (5.3 months). Credits: IMNC-IN2P3 (CNRS UMR 8165)

In this context, different parallelization methods for Monte-Carlo simulations have been proposed to be executed on distributed environments. Authors presented their work of implementing a distributed version of GATE deployed on EGI in [Camarasu-Pop *et al.* (2010)]. They studied two approaches: static and dynamic partitioning of the simulation workload. The number of jobs for a GATE workflow instance can vary from 1 to 500, depending on the estimation of the execution time of simulation made by users. With a static partitioning, each job will compute a fix number of particles, which is predefined depending on the number of submitted jobs. Jobs will be resubmitted if any execution failure occurs. With the dynamic partitioning, each job keeps computing the tracking of particles until the desired total number is reached. Hence, the number of executed jobs may be less than the original submitted number if all particles are already computed by early or faster jobs.

Figure 2.2 shows the abstracted workflow of GATE deployed and executed on EGI via VIP. The GATE workflow is expressed using the GWENDIA language [Montagnat *et al.* (2009)] and managed by MOTEUR engine [Glatard *et al.* (2008)]. It consists of two main phases: computing and merging.

During the computing phase, a set of independent GATE jobs: (1) download the required input files; (2) execute a part of the computational workload; and (3) upload partial results on a SE upon completion. Once all the particles have been computed, the workflow enters a merging phase, where all the partial results are downloaded from different SEs by one or several Merge job(s) (4) and aggregated (5). The final result is then copied back to a SE (6). The data exchanges between jobs of the GATE workflow are file-based and require transfers over the network and through SEs, since the jobs may be executed in different nodes that do not share a common file system.

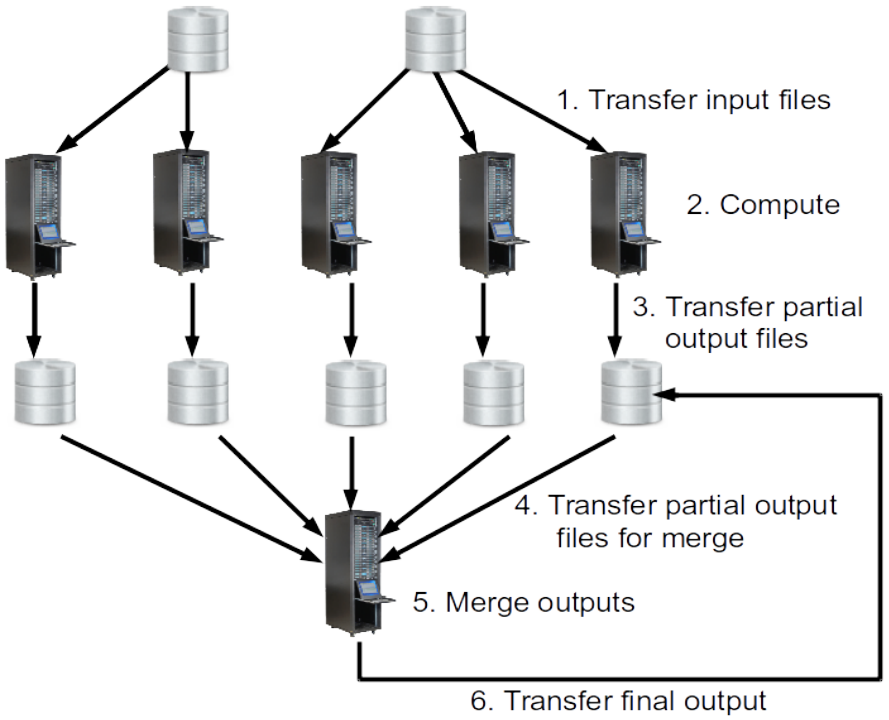

Figure 2.2: GATE workflow including a computing phase, a merging phase, and associated file transfers.

In step (1), GATE jobs need to download three files to enter the computing phase:

1. a wrapper script file automatically created by VIP;
2. a tarball file containing the GATE executable release along with all the required libraries needed for execution;
3. a tarball file containing the user-defined inputs, e.g., macro files describing the simulation, images, or ROOT files. The size can be quite different depending on each execution varying from several kB to hundreds of MB.

Merge jobs need to download an executable release file and the same user-defined input file as GATE jobs to merge all the partial results produced by the GATE jobs.

2.3 Execution traces

A very little set of traces may not represent the overall characteristics of the executions of applications and the underlying platform, while analyzing long-term execution traces in a fine-grained way can be an extremely time-consumming task. We thus decided to make a trade-off by collecting 60 execution traces of GATE workflows submitted by 14 distinct VIP users over a period of one month from September 8, 2015 to October 8, 2015.

These workflows are different in terms of software release, input files, number of jobs, and execution times.

These collected traces correspond to the execution of 11,033 GATE jobs and 60 Merge jobs by compute nodes across 41 different computing sites, which represents 60% of the computing sites in Biomed VO. These jobs performed 54,777 file transfers using 65 different SEs, including 33,095 downloads and 11,033 uploads by GATE jobs; 10,589 downloads and 60 uploads by Merge jobs.

During the execution of a workflow, two main sources of information are produced by the jobs and recorded by the VIP server. They are summarized in Table 2.1.

Source	Information
Log files	• Log creation time, workflow name, job id • File transfers: source, destination, duration, etc. • Hardware: name, number of cores, processing speed
Database	• Timestamps for each phase of job execution

Table 2.1: Summary of information recorded in traces.

The first source is the structured log file produced by each job (GATE or Merge). It starts with a header that comprises a timestamp of the log creation, the name of the workflow and the job id. User information, including the name of the user and which VO the user belongs to, can also be found in this header. Information on the hardware on which the job is executed is gathered from different sources, for instance, the name of the compute node, the information on the network card, the number of cores and their processing speed (expressed in BogoMIPS), and information on memory capacity. For each transfer, jobs log its source, destination, file size, and duration. It is important to note that the duration value is given by the time lag between the moment when the job starts to download the file and the moment when the job finishes the download. It is a single value that includes the latency of a transfer request from middleware services, the I/O latency from storage systems, and the data transfer time spent in the network. In production systems, file transfers do not always succeed because of the network connectivity or issues related to the SE (e.g., maintenance shutdown or no more available space). For jobs with several retries before achieving a successful transfer, the logged durations even include the duration of the different attempts. However, we do not have detailed information to distinguish the exact time spent in these different phases (i.e., middleware latency, I/O latency, and network transfer time) for a file transfer.

The second source is a database filled with various information on each job. While some information is redundant with the contents of the log files (e.g., job id and status, workflow, worker node, and grid site names), it also contains a timestamp for each milestone on the

job execution time line (i.e., job creation, entering the queue, starting the download, computation, uploading results, and termination).

We developed the **log2sim** framework [Suter *et al.* (2016)] to parse the log files and the database produced by jobs in the GATE workflow. We produce three CSV files for each workflow, which facilitate us to conduct analysis in the rest of this chapter.

For all workflows in the collected traces, their sizes are summarized in Table 2.2. There are 5 small workflows composed of less than 10 jobs, 13 workflows have more than 10 but less than 25 jobs. 42 workflows consist of more than 100 jobs and the largest workflow is composed of 500 jobs.

number of jobs	<10	10 ∼ 25	100	>100
number of workflows	5	13	25	17

Table 2.2: Summary of the number of jobs for 60 workflows.

In the collected traces, four different software releases are used by GATE jobs. Their sizes are 121.5MB, 376.9MB, 463.7MB, and 501.8MB. The size of the wrapper file for GATE jobs is always around 73kB. A single 90MB release file is used by all Merge jobs. The size of the partial results produced by each GATE job and the user-defined inputs varies a lot from one workflow execution to another.

Table 2.3 summarizes the size of input files and partial results for GATE and Merge jobs. Input files vary from 4.9kB to 29MB with a median size of 20.8kB while partial results vary from 0.5kB to 90.1MB with a median size of 186kB.

	Min	1st Qu.	Median	Mean	3rd Qu.	Max
input file for gate/merge jobs	4.9kB	10.7kB	20.8kB	547kB	88.1kB	29MB
partial results for merge jobs	0.5kB	38.1kB	186kB	3.3MB	2.7MB	90.1MB

Table 2.3: Statistics on the size of different files in 60 workflows.

2.4 Characteristics of file transfers

After having presented the number of jobs and the size of downloaded files in the studied workflows, we conduct a more detailed analysis in this section. First, we investigate the global distribution of transfer durations for different file sizes. It allows us to obtain an overview of the file transfer durations in a production system. Then different characteristics of file transfers extracted from individual workflows will be illustrated. 80% of transfers are done by the GATE jobs and their durations will directly decide when jobs can enter the computing phase, we thus focus on the file transfers initiated by GATE jobs.

2.4.1 Coarse-grain analysis

Table 2.4 summarizes the statistics of transfer durations for different file transfers of all the GATE jobs in 60 execution traces.

	Min	1st Qu.	Median	Mean	3rd Qu.	Max
12-byte upload test	1.0	1.0	1.01	2.59	1.01	579.8
input file transfer	1.0	1.02	1.03	1.715	1.07	122.0
release file transfer	1.0	4.05	5.01	34.8	14.13	2423.1
wrapper file transfer	1.0	1.02	1.03	3.08	1.04	523.5

Table 2.4: Statistics of real transfer durations for different files (in seconds).

Before starting to download the required input files, GATE jobs first test whether they can reach their default SEs by doing a 12-byte upload test. As shown in the first line of Table 2.4, these 12-byte transfers required at least 1s and 75% of them are completed in less than 1.01s, with an average value of 2.59s. The durations of such small file transfers give us an idea of the minimal time required to perform data transfers, i.e., the minimal latency experienced by each file transfer in EGI is around one second. It allows us to estimate the latency of a transfer request, which will be used to instantiate the communication costs between application jobs and middleware service of EGI to reflect this latency in our simulator described in Chapter 3.

The maximum duration of a 12-byte upload test observed in the collected traces is 579.8s, which is abnormally long. By checking the corresponding execution log, we find this large value is given by a job that made two attempts to complete its upload test. This job first conducted a test to its local SE but it failed. Then it succeeded in testing a backup SE pre-defined by the VIP administrator. Several upload-retries might have been triggered while contacting the local SE before starting to use the backup SE. The total duration of all these retries is included in the 579.8 seconds. As explained before, we can not distinguish the exact time spent in retries from the actual data transfer over the network because of the lack of detailed information in the execution traces. But this additional overhead because of one transfer failure is still remarkable, which implies that the reliability of the SEs is a very important aspect that needs to be considered for file management in large production systems such as EGI.

For input file transfers, 75% of them complete in less than 1.07s with a average value of 1.71s (see the second line in Table 2.4). The maximum value (122s) is given by a download of the maximum size of input file (29MB) between a compute site and a SE that are both located in UK. Another download of this particular input file (29MB) between the same site and SE pair (there are only two such downloads among all traces) in the same workflow took 94.03s. No transfer failures were recorded for these downloads in the logs. We thus consider that these long transfers are due to a poor network connectivity between the site and the SE.

With a larger size for release file transfers, the variability in durations increases a lot. 75% of them complete in less than 14.13s with an average value of 34.8s (see the third line in Table 2.4). The maximum observed duration is 2423.1 seconds given by a download of a 501.8MB release file with no transfer failure recorded for this download in the log.

To understand whether this long duration is related to a poor network or related to other issues, we looked at 27 downloads of the 501.8MB release file between the same SE ("srm-biomed.gridpp.rl.ac.uk") and the same site ("CIEMAT-LCG2") as for the longest transfer in the same workflow. We find the minimum duration is only of 95.81s which is 25 times faster. Figure 2.3 depicts the potential concurrent downloads to the longest transfer.

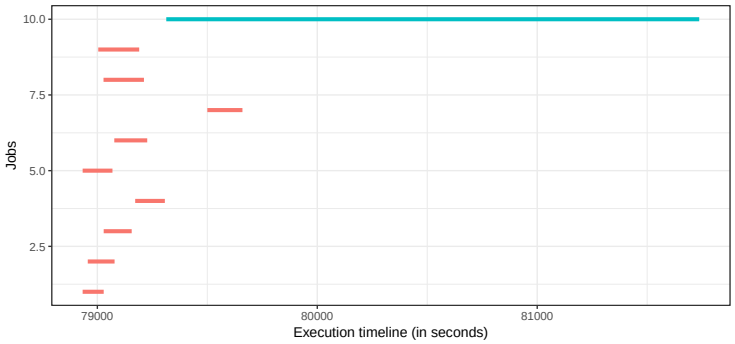

Figure 2.3: Gantt chart view of 10 transfers from "srm-biomed.gridpp.rl.ac.uk" to "CIEMAT-LCG2".

It shows that this large difference is neither due to concurrent transfers within the workflow execution, nor because of the external concurrent transfers from other applications since one download started just after the longest one but was much faster.

We then investigate 409 transfers from/to the site "CIEMAT-LCG2" in the workflow where the longest download is observed, which is presented in Figure 2.4.

We observe that there are a large number of transfers from/to this site before the longest download. We thus assume that this large number of concurrent transfers at the level of the compute site might have badly delayed the completion time of this particular transfer. Such a phenomenon should be reflected in a realistic model of EGI.

Regarding the wrapper file transfers, the maximum duration is 523.5 seconds (see the fourth line in Table 2.4). It is much longer than the average duration for a 73kB file transfer. In order to understand where these pathologically long downloads come from, we analyze the sources of wrapper transfers accomplished in more than 60s. There are 196 such transfers in total. We note that 195 among these transfers are from a specific SE ("ccsrm02.in2p3.fr"). The administrators of this SE confirmed that there was a scheduled maintenance shutdown of this SE during the period in which we collected traces. It might have led to plenty of pending transfer requests from/to this SE before or just after the

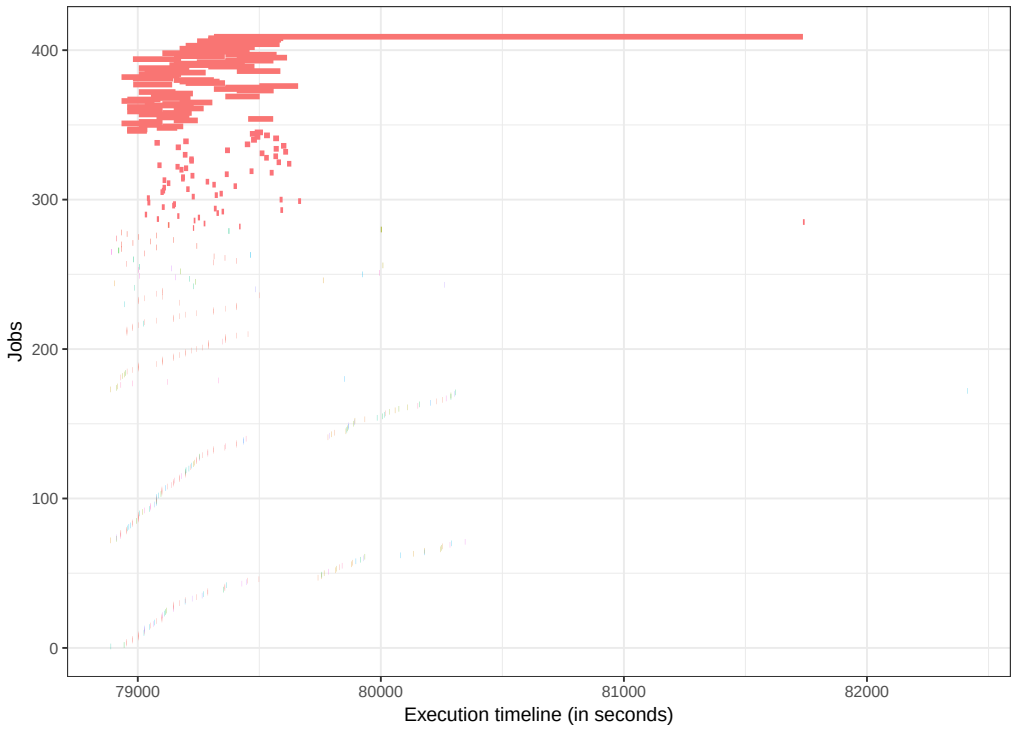

Figure 2.4: Gantt chart view of 409 transfers to "CIEMAT-LCG2", including uploads and downloads in one workflow execution. Small points correspond to very short transfer duration.

maintenance, which might have badly delayed the transfers. This problem that SEs can get more data than they can handle due to the changes in the systems, such as the downtime of storage, is also mentioned in [Barisits *et al.* (2017)].

By comparing the source and destination of each transfer, we distinguish three different categories for transfers/links to investigate the potential network hierarchy in EGI. Inspired by the three-level hierarchical network model in the literature, we consider:

- local transfer: the download source is the defined local SE for the computing site where the job is executed;
- intra-country transfer: the download source and computing site where job is executed are in the same country;
- inter-country transfer: the download source and computing site are not in the same country.

In order to compare transfers for all files, regardless of their sizes, we transform the transfer durations into bandwidth, i.e., the file size is divided by the transfer duration minus the latency (i.e., one second). The bandwidths derived from local transfers correspond to the intra-LAN links. If we consider each country as a region, then the bandwidths

derived from intra-country and inter-country transfers correspond to the inter-LAN and inter-region links in a three-level hierarchical model, respectively.

The distribution of bandwidth experienced by the 11,033 release file transfers according to transfer categories is shown in Figure 2.5, including 7,902 local transfers, 672 intra-country transfers, and 2,459 inter-country transfers.

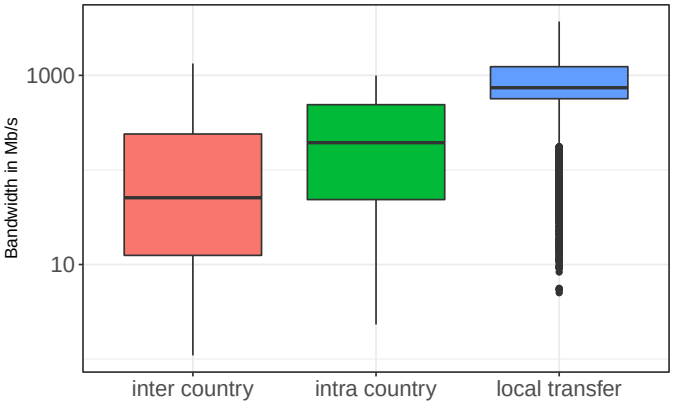

Figure 2.5: Distributions of the experienced bandwidth by 11,033 release transfers according to different link categories.

We observe that the overall bandwidths experienced by transfers significantly increase from inter-country link to local link. It confirms that the network hierarchy exists in EGI at some level. However, we can not conclude that the connectivity of local links is always better than others, which is shown by the observed outliers for local transfers. These observations imply that a three-level hierarchical model with limited bandwidths could be a good approximation for EGI only at a coarse level. The real production systems are more heterogeneous and complex than what a simplified hierarchical model can capture.

To understand where the long local transfers come from, we look at 247 local transfers with experienced bandwidths lower than 48.5 Mb/s (i.e., the first Quartile value of bandwidth for intra-country transfers). Figure 2.6 depicts these bandwidths according to the SEs from where the download is made.

We observe a large variability for the derived bandwidths even for a given SE. Note that we mix the downloads from different workflows in Figure 2.6, each workflow might thus experience different background network traffic conditions during its execution. It is thus difficult to determine whether the reason for this large variability is due to external network traffic or some other issues. In the next section, we focus on the analysis of transfers from an individual workflow execution, which can also guarantee that the size of release file is unique.

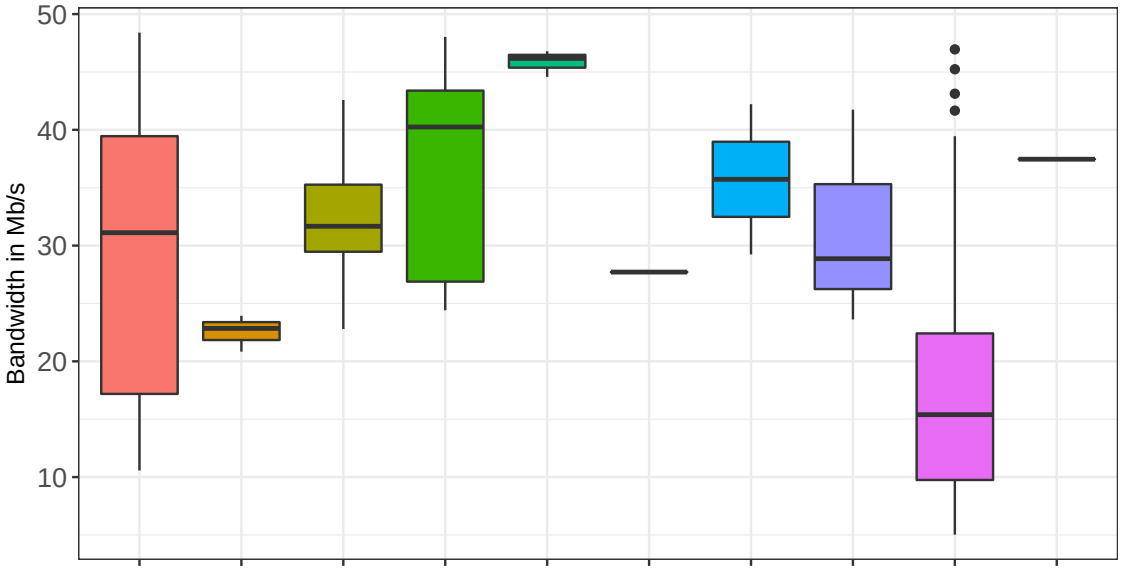

Figure 2.6: Small local bandwidth for different SEs. Each bar corresponds to the bandwidths experienced by local transfers from a specific SE.

2.4.2 Fine-grain analysis

During the analysis of the file transfers from all workflows, we identified several aspects that could explain the variability of transfer durations. They may be related to the network topology of the underlying system or only correspond to a transient issue on one specific SE. In what follows, we attempt to identify more characteristics of file transfers by focusing on individual workflow executions.

Figure 2.7 depicts the distribution of release transfer durations for a given SE (marsedpm.in2p3.fr) in one workflow execution. Each panel contains the downloads to a specific computing site.

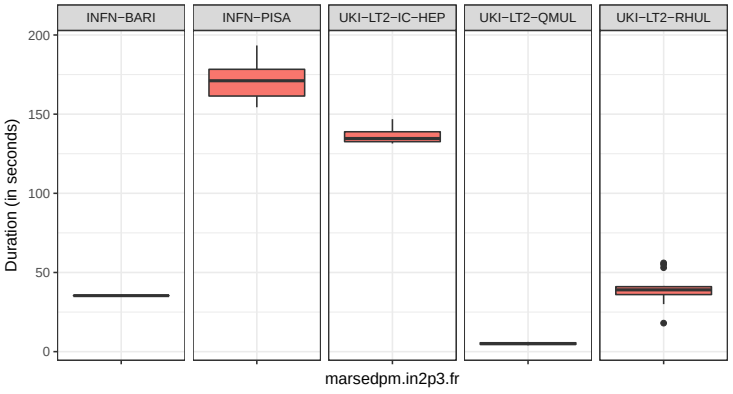

Figure 2.7: Distribution of release file transfer durations for a given SE (marsedpm.in2p3.fr) in a given workflow.

Clearly, we observe that download durations are quite different to different computing sites from a given SE and the variability decreases for a given computing site. It shows that the connectivity to different computing sites from a given SE is very heterogeneous in EGI.

In order to further investigate the characteristics of transfer durations for a given computing site, we zoom in on the downloads from a given SE ("sbgse1.in2p3.fr") to a given site ("INFN-PISA") in one workflow execution. Figure 2.8 presents the Gantt chart of the transfer durations to three different clusters inside this particular site.

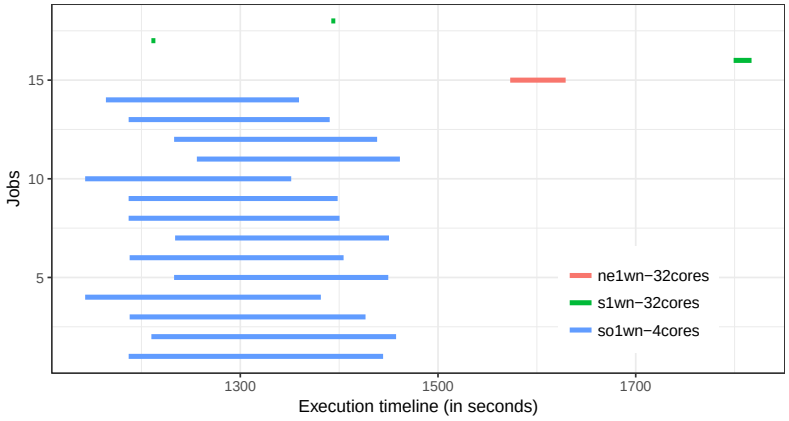

Figure 2.8: Gantt chart view of 18 release file transfer durations from the SE "sbgse1.in2p3.fr" to the site "INFN-PISA". Transfers belong to different clusters are in different colors.

We notice a large variability of download durations among different clusters. Even though the first two downloads for the "s1wn" cluster are concurrent to the downloads for the "so1wn" cluster in the execution time line, their durations are very different: around 200 seconds for the "so1wn" cluster but only 4 seconds for the "s1wn" cluster. It implies that compute nodes in different clusters do not share one common link inside a computing site.

Even when we group transfers by SE, computing site, and cluster for one file size, large difference of transfer durations can still be observed. Figure 2.9 presents the Gantt chart of the transfer durations for 5 jobs executed in the same cluster in a given computing site with a given SE in one workflow execution.

We can see that the first four files are transferred much faster (from 14s to 45s) than the fifth one (172s). The first four transfers are all finished before 200s while the fifth one begins after 400s. During the workflow execution, transfer 5 is neither impacted by other transfers from the same SE or to the same computing site. We thus assume that the reason for this particularly long transfer is the influence of external network load, which, however, is not observable in our execution traces.

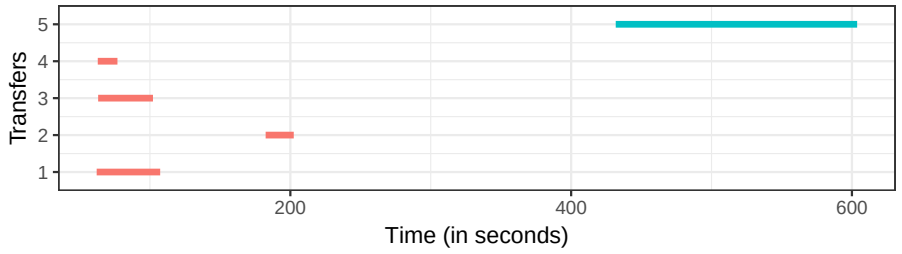

Figure 2.9: Gantt chart view of 5 release file transfer durations for a given SE, a given site, and a given cluster.

Another identified issue is shown in Figure 2.10. It presents the Gantt chart of the transfer durations for 5 jobs executed in the same cluster in a given computing site with a given SE in one workflow execution.

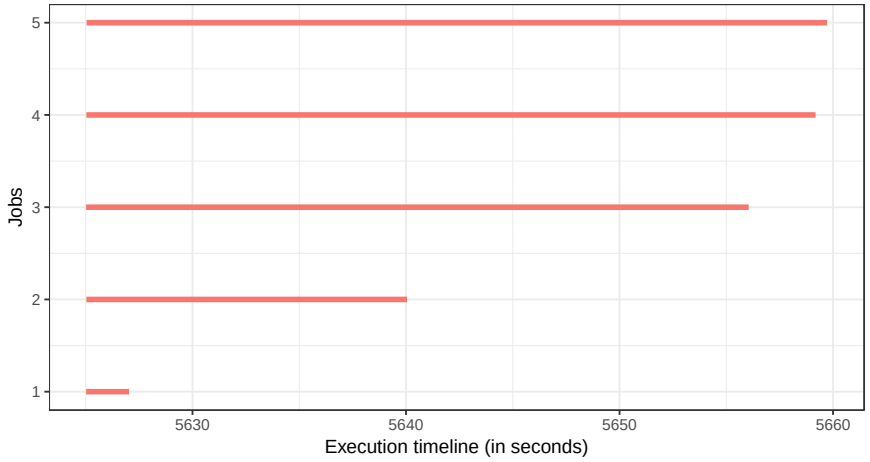

Figure 2.10: Gantt chart view of release file transfer durations for a given SE, a given site, and a given cluster.

We observe that these 5 transfers are simultaneous but their durations exponentially increase, from 2s to 35s. It is therefore not caused by external load. This phenomenon can be due to the local configurations of the SE that limit the maximum number of concurrent transfer and trigger some "timeout and retry" mechanisms. This variability is then considered as a special configuration of SEs, but not as a characteristic of file transfer. It should thus be captured by the simulated services of SE in simulator, but not in the platform model. Such information obtained by analyzing file transfers from individual workflows allows us to derive more detailed characteristics of the network topology, which will be used to build a fine-grain platform model for the Biomed VO in EGI presented in Chapter 4.

2.5 Characteristics of workflow and job executions

In this section, we analyze the characteristics of workflow executions, such as the queuing time of jobs, the distribution of jobs in sites and in countries, and the number of downloads by SEs. They can have a non-negligible impact on file transfers.

2.5.1 Queuing time durations

In large distributed systems, application jobs often suffer a delay between their submissions and their executions. This delay is called queuing time and includes the resource allocation time, job scheduling time, and other overheads introduced by different middleware services. Queuing time determines when jobs can start to download their required files and thus impact the network sharing for a given link. As a consequence, the global performance of file transfer for applications is indirectly influenced by this delay. The statistics of queuing time durations for all GATE jobs are summarized by Table 2.5.

	Min	1st Qu.	Median	Mean	3rd Qu.	Max
queuing time (in sec)	0	814	2315	6293	6490	62295

Table 2.5: Statistics of queuing time for GATE jobs in collected traces.

It shows that half of the jobs suffer a delay of more than 30 minutes with an average of 104 minutes. Such a large variability in queuing time has also been observed in other EGI VOs [Mościcki *et al.* (2011)]. The maximum queuing duration (17 hours) is observed for a workflow composed of 5 jobs executed in a single site. Although these jobs suffered a large queuing time, the delay between the first job and the last job in this site was not that large, i.e., 130 seconds. This observation shows that jobs may suffer a very long queuing time but the intra-site delay could be much smaller. The execution delay inside a site for a given workflow is a more meaningful metric than the general queuing duration to characterize the workflow execution on EGI.

Therefore we introduce another parameter to quantify the execution delay inside a site, i.e., the time delay between the first and subsequent jobs starting to execute in a given computing site for a given workflow. This delay is computed as: $(D_i^j - D_1^j)$, where D_1^j is the time of the first job starting to execute in a given site j, D_i^j is the starting time of the i^{th} job in site j during one workflow execution. As shown in Table 2.6, 75% of the jobs start to execute more than 91 seconds later compared to the first job in a given site. 50% of the jobs have a time delay of more than 569 seconds. The maximum intra-site delay is 78,312 seconds (21.7 hours) due to a failed job that was resubmitted 21 hours after the workflow execution. This particular job badly delayed the completion time of the workflow execution, which emphasizes the importance of the resource reliability for application performance in large distributed systems.

	Min	1st Qu.	Median	Mean	3rd Qu.	Max
intra-site delay (in sec)	0	91	569	2924	2312	78312

Table 2.6: Statistics of intra-site delay for GATE jobs in collected traces.

Based on these observations, we find that jobs suffer less intra-site execution delay than general queuing time delay. But still, this intra-site delay is not negligible. Given the characteristics of GATE workflows, i.e., all GATE jobs require the same input files, we believe that we can effectively use this intra-site delay to improve file management. For instance, we could make the first job to copy the required files onto its local SE. If this copy finishes in less time than the intra-site delay, all subsequent jobs will then be able to directly benefit of local transfers.

2.5.2 Distribution of jobs

The distribution of jobs can also have an impact on file transfer during the execution. For instance, if jobs are uniformly distributed over different computing sites or countries, the network resources from different domains might be equally exploited. Conversely, if the distribution of jobs is badly imbalanced, then some network resources may be overloaded leading to severe congestion.

Table 2.7 presents the statistics of the number of jobs per site in a given workflow execution. We distinguish small (< 100 jobs) and large (≥ 100 jobs) workflows. 75% of the sites executed more than 3 jobs and half of the sites executed more than 9 jobs during one execution for large workflows while half of the sites executed more than 3 jobs for small workflows. In general more than one job will be executed in a computing site during one execution, especially for large workflows.

	Min	1st Qu.	Median	Mean	3rd Qu.	Max
large workflow	1	3	9	27	26	493
small workflow	1	1	3	4	5	24

Table 2.7: Statistics for the number of jobs per site.

Figure 2.11 depicts the cumulative number of jobs by computing site. Each bar is colored by the country to which the computing site belongs and all countries are sorted in descending order of their cumulative number of jobs. We observe that jobs are not uniformly distributed over both computing sites and countries. One computing site in UK executed 2,214 jobs while other sites may only execute one job. Almost 90% of jobs were executed in four countries: the UK (32.5%), Netherlands (24.5%), France (20.8%), and Italy (11.1%). To be more statistically significant, we collected more recent information from the DIRAC [Tsaregorodtsev *et al.* (2010)] server which is the job scheduler used by VIP. The cumulative number of executed jobs within the Biomed VO for three months, six months, and one year in 2017 are summarized in Table 2.8.

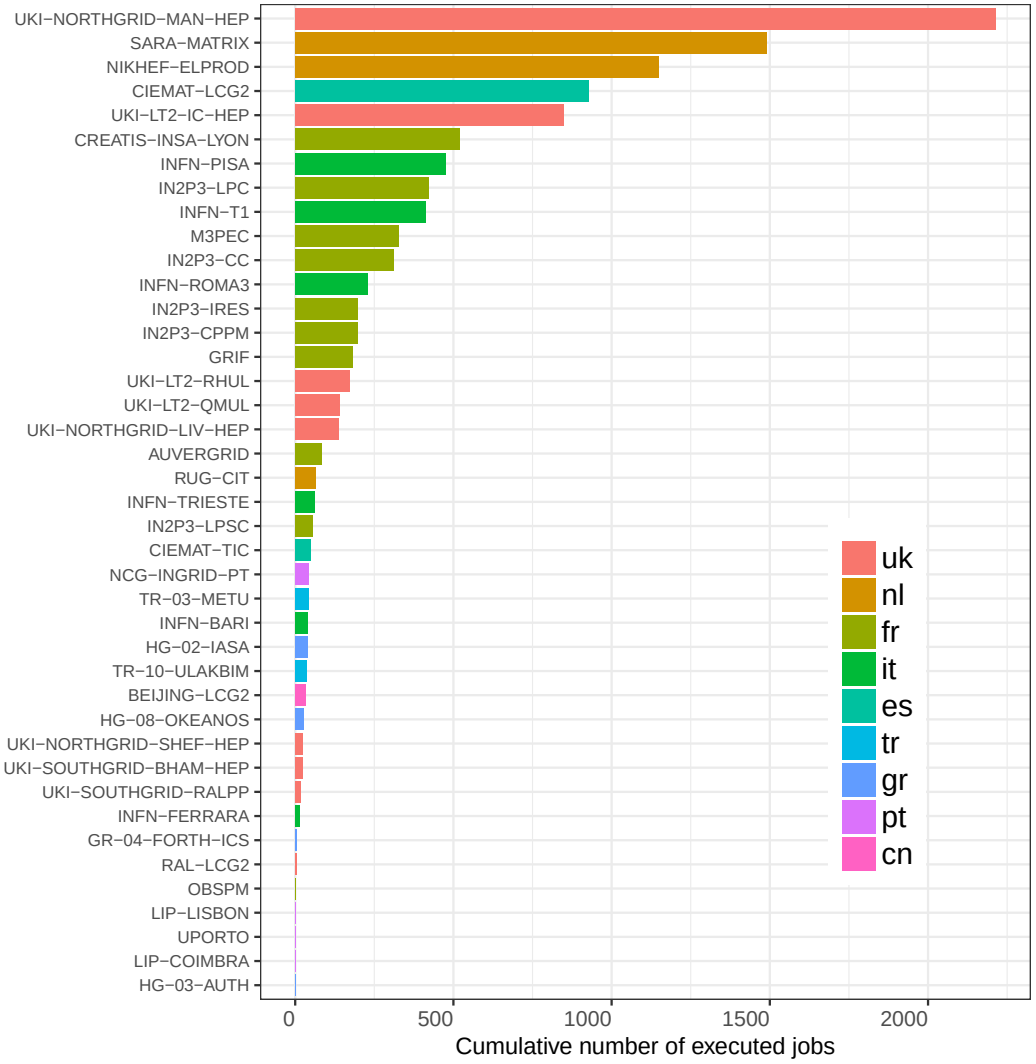

Figure 2.11: Distribution of executed jobs by site in studied workflows.

	Netherlands	UK	France	Italy	Others
In studied workflows	2,705(24%)	3,584(32%)	2,301(20%)	1,230(11%)	1,213(11%)
17/10 - 17/12	50,788(35%)	38,199(26%)	30,017(20%)	12,204(8.4%)	13,818(9.5%)
17/07 - 17/12	101,400(30%)	87,900(26%)	85,100(25%)	28,900(8.5%)	33,700(10%)
17/01 - 17/12	231,200(27%)	217,400(25%)	227,400(27%)	72,300(8.6%)	92,700(11%)

Table 2.8: Cumulative number of executed jobs in different countries in Biomed VO of EGI in 2017. Statistics extracted from traces and from the Dirac server.

We note that the distribution of the executed jobs monitored by DIRAC is coherent with the observation made from our collected workflows. About 80% of the jobs were executed in UK, Netherlands, and France. This large heterogeneity in the distribution of jobs allows for an optimization of file transfer by improving the placement of the required files. For instance, we could copy the targeted files onto SEs in or near to the countries where the most jobs are executed.

2.5.3 Cumulative downloads for SEs

Finally, we investigate the number of downloads per SE in the collected traces, which can help us to identify potential performance issues related to file transfers that come from the utilization of network resources.

Figure 2.12 depicts the cumulative number of downloads per SE. We can observe that the downloads are not equally distributed across SEs. The most used SE ("marsedpm.in2p3.fr") contributed to 7,548 downloads for jobs, while several SEs were rarely used. This large imbalance implies an inefficient network usage, which may penalize the performance of workflow executions.

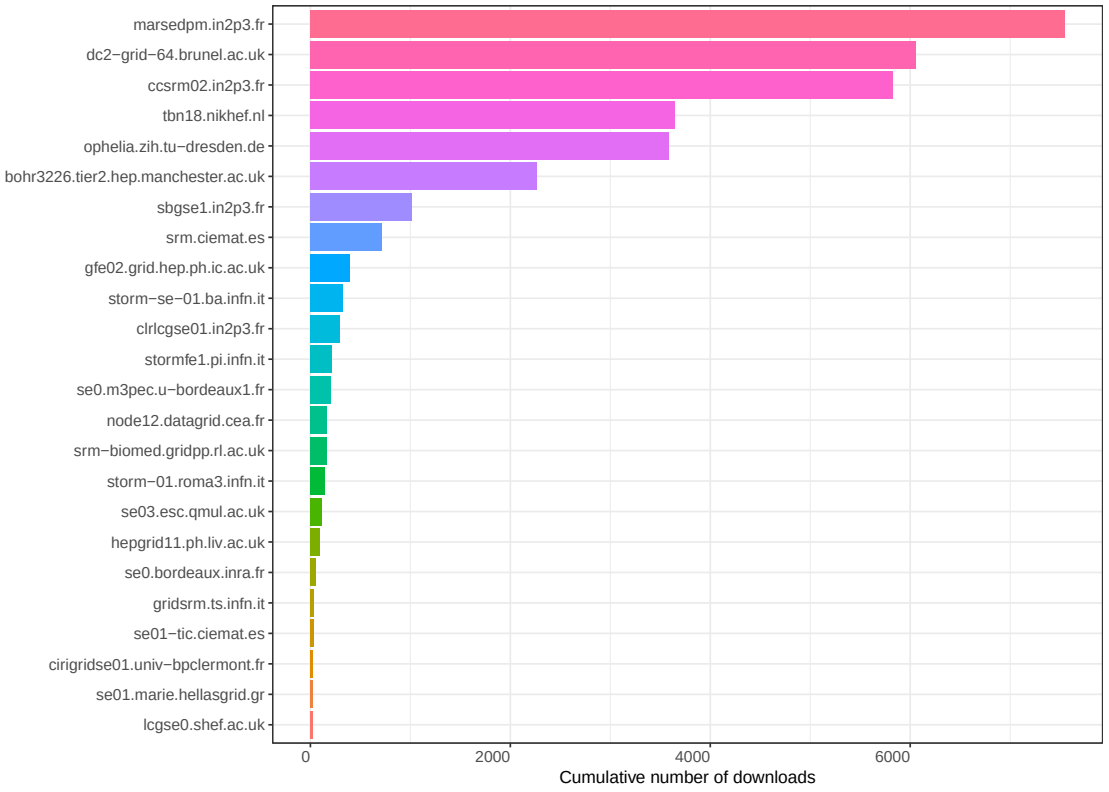

Figure 2.12: Cumulative downloads by SEs in studied workflows.

Another remarkable issue is the imbalanced usage between the storage and the compute resources for a site, which is shown in Figure 2.13. We compared the number of downloads for a SE and the number of jobs executed in the computing site associating this SE as the local one. We notice that there is a large difference between the number of downloads and the number of jobs for most SEs. Even badly, there were only 179 jobs executed in the site associating "marsedpm.in2p3.fr" as local SE despite the 7,548 downloads from this SE, which means that most of these downloads are either national or inter-country transfers.

From an global point of view, 21.7% (11,901) of the transfers are local, while 33.6% (18,438) are intra-country transfers, and 44.6% (24,438) are inter-country transfers. This

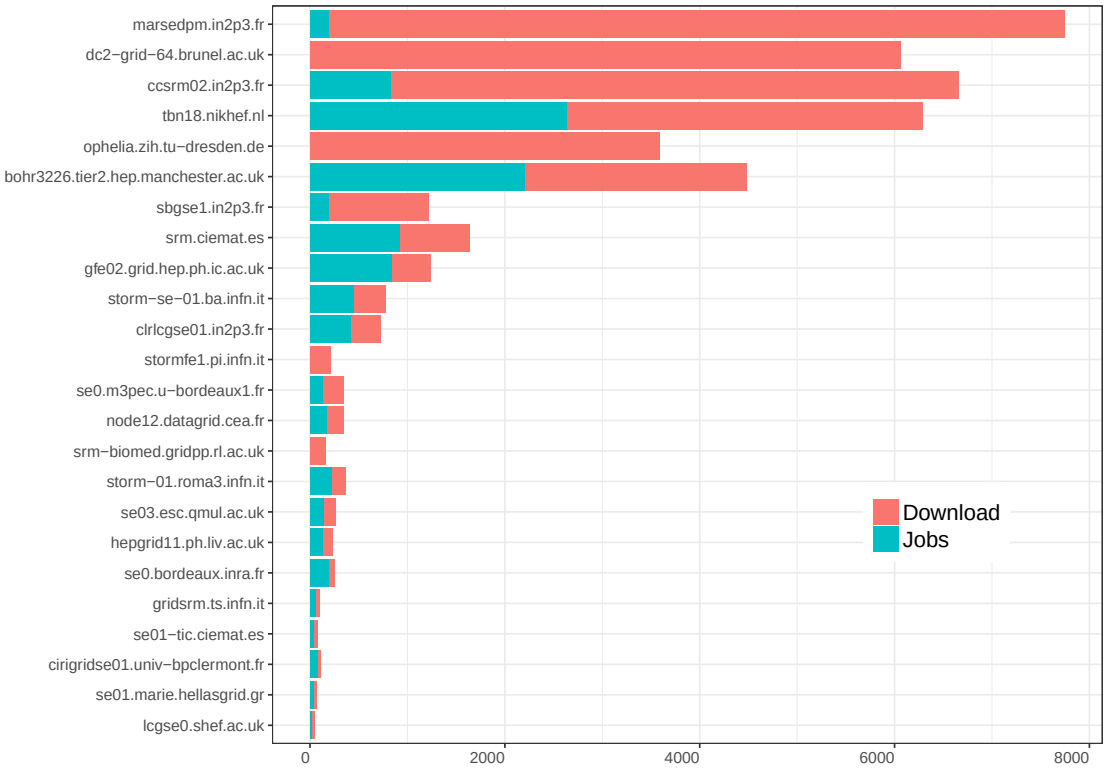

Figure 2.13: Cumulative number of downloads for SEs versus the number of jobs in the corresponding sites.

large number of distant (i.e., intra-country or inter-country) downloads not only badly influences the global performance of file transfers, but also implies an imbalanced usage between storage and compute resources in a site. These issues can be improved by adopting a more efficient file management strategy at the application level, which further emphasizes the necessity of the work proposed in this thesis.

2.6 Conclusion

In this chapter, we presented an analysis of execution traces collected from a production platform at a fine-grain level. This analysis has been done from two different angles: file transfer durations and characteristics of workflow executions. By investigating the durations of file transfers at a coarse and fine-grain way, we identified several aspects that could explain the large variability in file transfer durations observed in these traces. They may relate to the network topology of the underlying system or only to a transient issue on a specific SE. These findings are summarized in Table 2.9.

By studying the workflow executions, we extracted some characteristics which may have an impact on file transfers and also identified several improvable aspects related to file management. They are summarized in Table 2.10.

The non-negligible intra-site execution delay could be used to optimize the file placement during workflow execution, while the large heterogeneity of the distribution of jobs by site or by country enables the optimization of the file placement before executing workflows. Regarding the utilization of SEs, the collected traces show a suboptimal file management. The possible optimizations for these issues are shown in Table 2.11.

All the knowledge obtained in this chapter will help us to:

- feed the simulator presented in Chapter 3;
- model the targeted platform in Chapter 4;
- predict the bandwidth of missing links in Chapter 5;
- propose a dynamic replication method to optimize the performance of file transfers in Chapter 6.

Source	Findings/Hypothesis
File transfer durations	• latency for transfers in EGI is one second • the maximum number of concurrent transfers for SEs is limited • possible causes for extremely long durations: transfer failures, SEs shutdown, external network charge, etc.
Network topology	• network hierarchy exists, but more complex than a 3-level model • connectivity to different computing sites from a given SE is very heterogeneous • compute nodes in different clusters may not share one common link inside a computing site

Table 2.9: Summary of findings or hypothesis from the analysis of file transfers.

Source	Characteristics
Queuing time	• large variability of queuing time for jobs in Biomed VO • less intra-site delay but still non-negligible
Distribution of jobs	• The distribution is different not only among computing sites, but also at the level of countries • UK, Netherlands, and France are the 3 top countries with most executed jobs in Biomed VO
Utilization of SEs	• The utilization of SEs is not balanced • Most transfers are either intra-country or inter-country transfers for the overcharged SEs

Table 2.10: Summary of characteristics extracted from the executions of workflows.

Observation	Possible optimization
Intra-site execution delay	• Copy required files onto the local SE by the first job executed in a computing site
Heterogeneity of the distribution of jobs	• Choose SEs in the top 3 countries to host files
Heterogeneity of the utilization of SEs	• Choose the local SE of the computing site with most executed jobs to host files

Table 2.11: Summary of characteristics extracted from the executions of workflows.

Chapter 3

Realistic simulation of file transfers for applications deployed on distributed infrastructures

Abstract *This chapter presents our work to build a realistic simulator*[1]*. Our aim is to provide a simulated environment as close as possible to the real execution conditions for file transfers during the execution of applications deployed on large distributed infrastructures. We first identify several fundamental components for file transfers from real systems. Then the choice of simulation tools and the implementation design of the simulator are driven by our concern for the realism of simulated file transfers.*

3.1 Introduction

Simulation has been widely used in the research on distributed systems. It allows researchers not only to evaluate new prototypes or new strategies in a repeatable and controlled manner, but also enables full-scale evaluations in an acceptable time range. Simulation also provides access to variables that could hardly be monitored in real systems, such as network congestion.

Numerous simulation toolkits exist in the literature [Calheiros *et al.* (2011)a, Kliazovich *et al.* (2012), Casanova *et al.* (2014), Cai *et al.* (2017)]. They often offer core functionalities to simulate applications executed in distributed environments. Based on these core functionalities, users can develop their own simulators for general utilization [Chen and Deelman (2012), Desprez and Rouzaud-Cornabas (2013)] or only for studying one specific application [Camarasu-Pop *et al.* (2013)b, Camarasu-Pop *et al.* (2016)]. A simulator usually builds on models of the hardware platform, the software services deployed on the platform, and a mapping of these services on the hardware resources.

[1] The code of simulator is available at: "http://github.com/frs69wq/VIPSimulator"

Our aim in this chapter is to build a realistic simulator allowing us to accurately simulate file transfers during the execution of medical imaging workflows deployed on EGI. To reach this goal, we first investigate the execution of the GATE workflow to **identify** the relevant components (e.g., services related to transfers and algorithms used to select file replica) used in production systems. Then we can **abstract** them as close as possible to the reality and **implement** them in our simulator.

We choose to build our simulator upon the SimGrid toolkit [Casanova *et al.* (2014)], which is a scientific project with a 20-year development and that offers accurate network and computing resource models [Velho and Legrand (2009)]. SimGrid allows researchers to conduct large scale simulations in a reasonable time without losing the accuracy of simulation results.

The rest of the chapter is organized as follows. We investigate the real execution of the GATE workflow on EGI in Section 3.2. It allows us to identify the fundamental components to realistically simulate file transfers. More details on the SimGrid toolkit will be given in Section 3.3. The design and the implementation of our simulator will be presented in Section 3.4.

3.2 Execution of the GATE workflow on EGI

3.2.1 GATE workflow in VIP

Figure 3.1 presents the complete life-cycle of the execution of the GATE application on EGI. The MOTEUR [Glatard *et al.* (2008)] workflow engine, which is hosted on the VIP server, works as a central authority to control the executions of GATE or Merge jobs based on message exchanges.

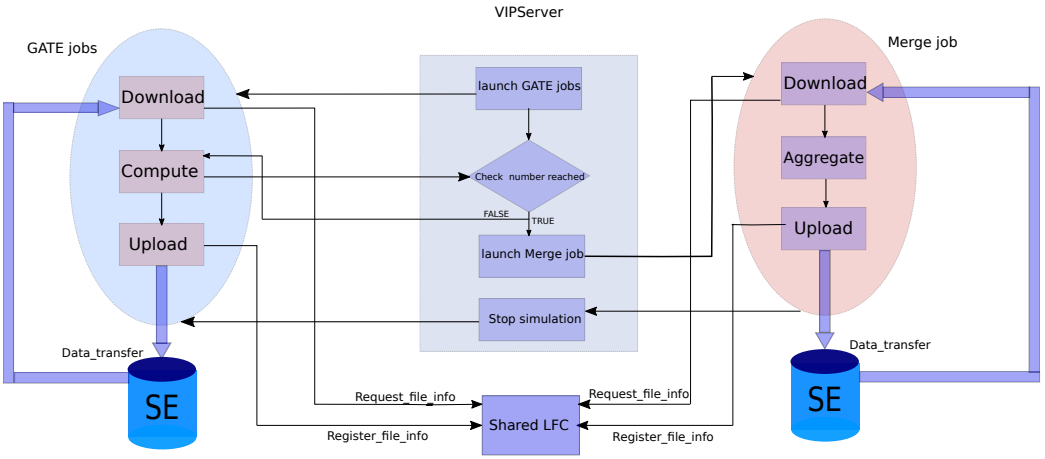

Figure 3.1: The complete life-cycle of the execution of the GATE application on EGI.

Once the compute resources have been acquired, the submitted GATE jobs will begin to execute the three phases, i.e., download the required files, compute the workload, and upload their partial results. When the total number of particles has been reached, MOTEUR will launch the execution of the Merge job which first downloads all the partial results uploaded by GATE jobs and then aggregates them. The final result is uploaded onto a SE on EGI after the aggregation. Finally, MOTEUR stops all the GATE jobs and the execution of workflow is accomplished.

The downloading of input files and uploading of partial results require transfers over the network and through different SEs on EGI. Several high-level data management clients are provided by the EGI middleware, which hide the complexity of the underlying distributed infrastructure. For instance, VIP leverages the *lcg-util* client which provides simple commands for users to download or upload files on EGI. In order to realistically simulate the behavior of these services, we need to take a deeper look at the complete process of a file transfer on EGI.

3.2.2 Data management services in EGI

Figure 3.2 shows the general data management on EGI. Files are stored on disks or tapes at different sites. Storage resources are managed by Storage Resource Managers (SRM) [Donno *et al.* (2008)], which provides a homogeneous interface and allows to list all files stored in a given SE. SRM is also in charge of translating Storage URLs (SURL) to the actual file location on disks or tapes and to Transfer URLs (TURL). These TURLs will finally be used to physically access a file on EGI. At the top level, a Logical File Catalog (LFC), which may have several distributed copies in the production system, contains the logical filenames (LFN) of physical files that may have one or multiple replicas stored on several SEs and are registered by SURLs.

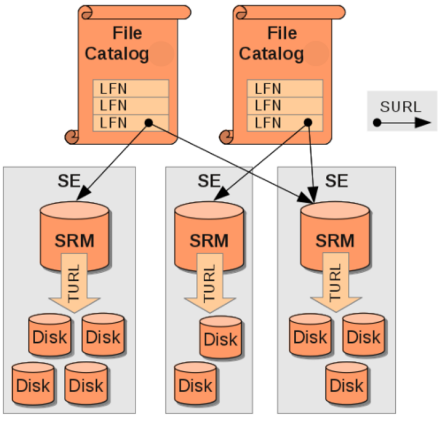

Figure 3.2: Data management on EGI. Figure extracted from [Löschen and Müller-Pfefferkorn]

As shown in Figure 3.3, four communications are done among the Worker Node (WN), which is the computing node inside a site where the jobs are finally executed at, and middleware services before a file is actually transferred over the network. When a job executed in a WN attempts to download a file by the using corresponding service from the *lcg-util* client, the client first requests the LFC to retrieve all the SURLs for the replicas of the target file. It automatically selects a SURL if multiple replicas are available for the given file. Then the client contacts the SRM of the selected SE to get the TURL of the requested file. Finally, the client downloads the file by its TURL. Conversely, to upload a file, the client sends the file to a SE and then notifies the LFC to register the LFN if it does not exist or to add the SURL of this SE as a new available replica for existing files.

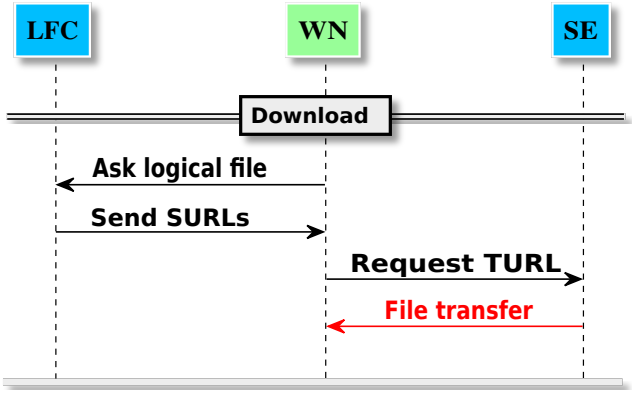

Figure 3.3: The process for file download on EGI.

The algorithm to automatically choose a replica (i.e., the SURL of a SE) for a file download used by the *lcg-util* on EGI is composed of two steps. First, a sorted list for all available replicas is constructed by lfc_fillsurls (described in Algorithm 1). The order is set according to the distance to the computing site where jobs execute, that is, first the SE local to the computing site (line 3 in Algorithm 1), then the same country as the job execution (line 5 in Algorithm 1), and in last resort, randomly among all remaining replicas ((line 7 in Algorithm 1)). This order allows jobs to benefit of the higher bandwidths from a closer SE, which was also observed and verified from the execution traces in Chapter 2.

Replicas in this sorted list are then selected by testing their availabilities ((line 6 in Algorithm 2)). If the first replica is not available, the client will continue to test the next replica in a round-robin manner until an available replica is found to download the file ((line 7-11 in Algorithm 2)). Several possible reasons may cause the unavailability of a replica in the real system. For instance, the SRM may fail to list the files in the corresponding SE, the file status maybe NULL in the SE, or the SRM may fail to transform a SURL into a TURL, etc.

Algorithm 1: lfc_fillsurls

Input: list of available replicas: L
Output: sorted list of replicas: SL

```
for each replica r in L do
    if r is local SE of the job then
        add r in the first place of SL;
    else r is from same country of this job
        randomly add r after the local SE in SL;
    else
        randomly add r after all replicas from the same country in SL
    end
end
```

Algorithm 2: The algorithm of replica selection implemented in *lcg-util*.

```
rep_list = get_replica_list_from_LFC();
sorted_rep_list = lfc_fillsurls(); #get a sorted list of replicas based on Algorithm 1
bool flag = false;
while !flag do
    current_replica = sorted_rep_list.current_replica();
    flag = Test_availability(current_replica);
    if flag then
        copy_file(current_replica);
    else
        current_replica = sorted_rep_list.next_replica();
    end
end
```

3.2.3 Summary of the characteristics of the real system

From the middleware services for a file transfer described above, we can identify several fundamental components to enable the simulation of file transfers on EGI :

- the workflow engine hosted on VIP server, which is a central authority to launch jobs, monitor the computed particles, and stop the simulation;
- the middleware services for data management (i.e., the functionalities of SEs and LFC) to enable the interaction processes of a file transfer;
- the jobs executed at different sites to request file transfers.

The implementation of these fundamental components is a mandatory step towards the simulation of file transfers on EGI. However, to ensure the realism, several aspects need to be taken into account.

The first aspect is the communication among jobs, SEs, and LFC. In the production system, these communications are not costless and they determine the minimum duration of a file transfer occurring on EGI.

The second aspect is the replica selection service. It decides the source for each download if no explicit SE is imposed. This decision directly impacts the sharing of network links within one workflow execution. Replicating the exact algorithm in simulation enables us to better reproduce the distribution of network traffic over links. It can also allow us to replay the simulation scenario corresponding to the current production conditions.

The last aspect is the life-cycle of jobs. It is another key part for simulating file transfers. For instance, the queuing time determines when jobs can start to download the input files and the compute duration will decide when jobs begin to upload their partial results. Similar queuing time for jobs in a same site will probably cause more concurrent transfer requests to/from a given SE, while large variable queuing time may lead to sequential transfer requests. The simulated transfer durations can be quite different with different patterns of transfer requests. It is thus important to correctly simulate the complete life-cycle of jobs, especially the starting point of each phase.

Besides all aspects above, we also needs an accurate network model, which enables us to correctly simulate the duration of each transfer given a precise description for the network of the underlying system (i.e., EGI).

3.3 The SimGrid toolkit

The SimGrid tookit is our best choice to accurately simulate file transfers. It provides core functionalities for simulating distributed applications in heterogeneous distributed environments. This tool is based on fast and accurate fluid models for network, which allows for large-scale simulations with thousands of file transfers and gives a certain confidence in the realism of the simulated communication time [Velho and Legrand (2009)]. The fluid network model abstracts the individual packets of an end-to-end communication into a flow, which is characterized by a bandwidth value or data transfer rate. It also takes into account network contention when multiple communication flows are present. Further (in)validation studies were conducted to ensure the accuracy of this network modeling in SimGrid [Velho *et al.* (2013)], which have been rarely done for other simulation toolkits.

Several programming interfaces are offered in SimGrid, for instance, S4U as the core API for simulations with Concurrent Sequential Processes (CSP) that interact by exchanging messages, SMPI for MPI simulations, and SimDag for DAGs of parallel tasks. A SimGrid task is composed by an amount of computing load and an amount of bytes to transfer over network. The total duration of a task consists in the transfer duration decided by the bandwidth of the route between its sender and receiver and the processing time of a computing load determined by the computing speed of the worker node.

A platform model is another important component for simulating distributed applications. It usually describes the characteristics of the compute resources and network

resources. It also describes how resources are grouped (e.g., in clusters, data centers, etc.) and interconnected through a network topology. SimGrid allows users to decouple the platform from the applications and exploits a hierarchical representation [Bobelin *et al.* (2012)] to describe a platform by decomposing it into networking zones. A zone can contain several other zones allowing to have more than one topology model in the platform. In a SimGrid zone, we can define:

- host: a single hardware resource, with a computing speed and a number of cores;
- cluster: a certain number of hosts interconnected by a local network;
- link: the network connection between hosts or zones with a bandwidth and a latency. Different bandwidth sharing models can be given for each link;
- router: an entity to declare which link is connected to this point;
- route: the explicit path between two hosts, clusters, or zones.

An example of hierarchical platform in SimGrid is illustrated in Figure 3.4. Circles represent hosts and squares represent network routers. Bold lines represent communication links. The zone "AS2" models the core of a national network interconnecting a small flat cluster (AS4) and a larger hierarchical cluster (AS5), a subset of a LAN (AS6), and a set of peers scattered around the world (AS7).

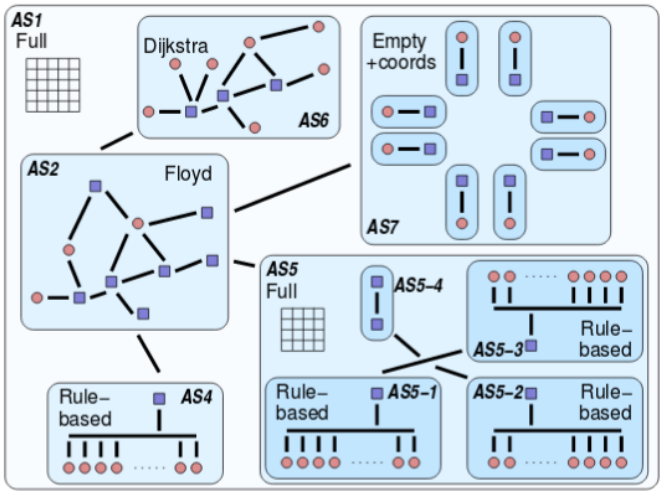

Figure 3.4: Graphical representation of hierarchical platform in SimGrid, illustrated from [Bobelin *et al.* (2012)].

3.4 Simulator design

A first simulator of the execution of GATE has been designed and used in [Camarasu-Pop *et al.* (2013)a]. However, the simulation of file transfers was simplified because the

focus was on the compute and merge phases of the GATE application. Authors chose a single file size of 15MB, which had a simulated median transfer time similar to the real transfers. Moreover, only one replica per file was simulated.

In our simulator, we attempt to replay all file transfers during the execution of GATE workflow on EGI. These transfers may have different file sizes and multiple file replicas scattered onto different SEs with respect to the execution instances. The middleware services related to data management are designed as close as possible to the actual behavior in the production system. As our final aim is to improve the file management at the application level, dynamically and correctly simulating the behavior of job scheduling and resource allocation in real system is not our focus in this work. We thus assume that the jobs composing the workflow have already been submitted and that the compute resources have already been acquired. Besides, we only consider the normal situations where all the partial results files produced by GATE jobs are downloaded by the Merge job and no particular failure happens (file transfer or job execution).

Similar to previous work, the VIP server is abstracted as a centralized point to control the execution of jobs. It determines when to launch the execution of simulated jobs (GATE or Merge) and when to stop them. We choose to simulate the interactions between VIP server and jobs by message exchanges with the same cost as the communication between jobs and middleware services, which will be explained in section 3.4.2. The main messages handled by the simulated VIP server are job connection, the number of computed particles by GATE jobs, and job disconnection. Based on this control-message mechanism, the complete logical life-cycle of the GATE application, as shown in Figure 3.1, is implemented and simulated. Hereafter, we will detail our additional efforts to improve the realism of the simulation of file transfers.

3.4.1 Simulated services for data management on EGI

SEs are designed to process two types of transfer requests from jobs: download request and upload request (with or without timeout). To reflect the fact that SEs can receive and process multiple simultaneous requests from jobs, each simulated SE is implemented with a parameter defining the maximum number of simultaneous requests it can receive. This value is currently set to 500, which corresponds to the maximum number of jobs in a GATE workflow. This value ensures that no unwanted contention is introduced by the simulator. However, it gives us the opportunity to simulate one of the phenomena highlighted in Section 2.4.2.

We can effectively further improve the realism for simulated SEs by investigating the configurations of different SEs to find out the general number of parallel requests accepted by each SE on EGI. Simulated SEs are assumed to have a large number of cores (48) to ensure that several simultaneous requests can be processed at same speed.

The Simulated **LFC** allows jobs to register files, list all registered files, and list all replicas for a given file. It also enables users to specify the replica locations for files registered in LFC before simulation. This information is stored in a file which is attached as an argument of the simulated LFC service, following the format: file name, size, `<se_1:se_2:....se_n>`. Such file is generated from the execution trace for each workflow. An example of the replica locations for a GATE workflow on EGI is illustrated in Figure 3.5.

```
inputs/gate/gate.sh.tar.gz,73039,tbn18.nikhef.nl:ccsrm02.in2p3.fr:marsedpm.in2p3.fr
inputs/gate/release_Gate7.1_all.tar.gz,501843312,bohr3226.tier2.hep.manchester.ac.uk:srm-
biomed.gridpp.rl.ac.uk:sbgse1.in2p3.fr:ccsrm02.in2p3.fr:marsedpm.in2p3.fr
inputs/gate/file-27434930370786.zip,6901,dc2-grid-64.brunel.ac.uk
inputs/merge/merge.sh.tar.gz,90104427,marsedpm.in2p3.fr
inputs/merge/file-27434930370786.zip,6901,dc2-grid-64.brunel.ac.uk
```

Figure 3.5: Example of a file catalog.

Each line corresponds to one file registered in LFC. The first line, for example, points to the "gate.sh.tar.gz" file, whose logical path is "inputs/gate/". This file has a size of 73,039 bytes and has been replicated on three SEs on EGI: "tbn18.nikhef.nl", "ccsrm02.in2p3.fr", and "marsedpm.in2p3.fr".

At the beginning of the simulation, the simulated LFC parses this predefined file catalog to retrieve the replica locations for each file. The file information (e.g., file name, the size) will also be stored at the simulated SEs which host a replica for the file. When SEs receive a download request for a file from jobs, they will generate a "SimGrid task" with the amount of bytes corresponding to the size of requesting file.

The **replica selection** service is implemented according to the exact algorithm shown in Algorithm 2. The SURL of each SE is represented by the SE name. Since we do not simulate any transfer failures, the first replica in the head of the sorted list is always available (line 6 in Algorithm 2) and it is always selected as the source to download the corresponding file. However, if a timeout is imposed and the simulated transfer is not accomplished before the timeout value, it will be systematically killed and the next replica in the sorted list will be attempted (line 10 in Algorithm 2).

3.4.2 Communication cost for file transfers

In Chapter 2, we found that the minimum duration of transfers on EGI was one second based on the analysis of the real transfer durations. We decided to reflect this minimum duration in our simulator by decomposing it into 900 milliseconds as communication cost and 100 milliseconds as network latency.

We equally decompose the 900 milliseconds into four messages, corresponding to Figure 3.3, and each message thus costs 225 milliseconds. These communication messages are simulated by "SimGrid tasks". As the computing speed for service nodes (i.e., SEs

and LFC) is defined as 5GFlop/s according to what we found in execution logs for the compute nodes, we therefore instantiated these "SimGrid tasks" with a computing load of 1.125GFlop (5GFlop/s × 0.225s) and a negligible amount of bytes to transfer.

The remaining 100 milliseconds are divided into four times 25 milliseconds representing the network latency for each of the messages. Indeed, each message goes through four links: two private links, the common backbone interconnecting worker nodes in a site, and the link that connects a site to a given SE. We decide to instantiate 5 milliseconds as latency for private links and 7.5 milliseconds for the two other links.

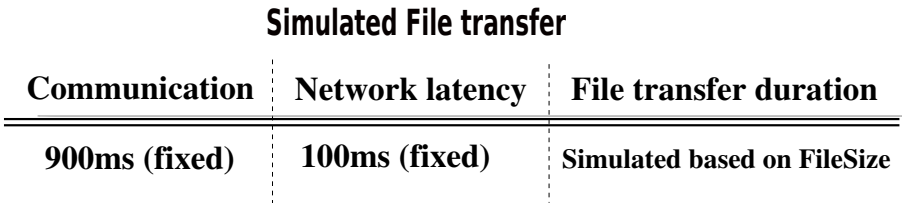

Figure 3.6: Three parts of a simulated file transfer in our simulator.

Finally, as shown in Figure 3.6, the duration of a simulated file transfer in our simulator consists of: (i) the simulated communication costs (i.e., 900 ms), (ii) the network latency across the routes (i.e., 100 ms), and (iii) the processing time of the "SimGrid task" generated by the SE. If a timeout is used, the total duration is the cumulative time of all transfer attempts.

3.4.3 Parameter injection

Given the complexity of the whole real system, it is difficult to correctly simulate all parameters since it would require to establish accurate models for all of them and also to validate them, which can be even more challenging. To better estimate the starting point of each phase in the life-cycle of jobs, we distinguish two strategies to decide the values of parameters in our simulator: inject or estimate.

Injecting a parameter for simulating the execution of a workflow means that the value of this parameter is directly extracted from the execution traces for the given workflow, while estimating a parameter means that the value is computed by a predefined model during the simulation execution. This distinction gives us more flexibility to design simulation scenarios focusing on the parameters that we want to investigate without losing realism for the other parameters.

The simulation strategy chosen for the different parameters related to file transfers are summarized in Table 3.1. The queuing time, compute durations, and upload file size for GATE or Merge jobs are always injected in our simulator. These are the parameters that we do not plan to investigate in the current study. We want these values to be as precise as possible in order to avoid additional bias for studying file transfers. The durations of

file transfer are simulated by using the different network sharing models defined in the SimGrid toolkit. The download sources for jobs are either imposed to match the same SEs in real execution or decided by the simulated replica selection service, according to the purpose of the simulation scenarios.

Parameter	Strategy
queuing time	• always injected
compute durations	• always injected
upload file size for jobs	• always injected
file transfer durations	• always estimated by network models in SimGrid
file replica for each transfer	• injected (the same as in real transfer) • estimated by the replica selection service

Table 3.1: Strategy for different parameters in simulator.

Our long-term aim is to construct a realistic simulator which can capture the actual behavior of the real production systems (i.e., VIP and EGI), instead of a theoretical simulator. Therefore, each time we change the simulation strategy of a parameter from *inject* to *estimate*, it requires to build an appropriate model and more importantly, to validate its performance by comparing the obtained results with the contents of real traces.

3.5 Conclusion

In this chapter, we presented our simulator built upon the SimGrid toolkit to study file management scenarios for applications deployed on EGI via VIP. We considered the specific workflow of GATE, which is representative of the deployment process of many other applications managed by VIP. We simulated the relevant components and the complete life-cycle of workflow executions by assuming that the jobs composing the workflow have already been created and that the compute resources have been acquired.

In order to improve the accuracy of the simulation of file transfers, we designed and abstracted the actual behavior of the simulated services as follows:

- Abstracting a file transfer into four messages among different components;
- Instantiating the cost of control messages based on the transfer latency derived from execution traces;
- Implementing the same replica selection algorithm as in the EGI middleware;
- Injecting the exact value of important parameters related to workflow executions.

In the next chapter, we will present how to construct realistic models of EGI to replay the executions of GATE workflow with a focus on file transfers by using our simulator. These ad-hoc models will be evaluated by comparing the simulated file transfers to the measured file transfers registered in the execution traces of GATE workflow.

Chapter 4

Realistic platform models for replaying real workflow executions

Abstract. *In this chapter*[1]*, we build realistic ad-hoc platform models to replay the executions of the GATE workflow on EGI. Starting from a simplified but widely used platform model, we propose incremental improvements to increase the accuracy of file transfer simulation thanks to a thorough analysis of both trace contents and simulation results. The overall improvement of these ad-hoc models is evaluated by confronting the simulation results to the ground truth of the actual executions registered in the execution traces. Results show that our proposed model largely outperforms the state-of-the art model to reproduce the real-life variability of file transfer durations.*

4.1 Introduction

A platform model is a critical component for the simulation of applications deployed on distributed infrastructures. The realism of the platform model strongly impacts the level of confidence one can have in simulation results, especially if the evaluation metrics are directly influenced by the platform model, such as file transfer durations which depend on the network topology and network bandwidths. The accuracy of the platform model is also key to the applicability of simulation findings to real production systems since different platform models may lead to different qualitative assessment for the same optimization strategy.

As we have pointed out in Chapter 1, the configurations of the platform models used to conduct simulations are often simplified in the literature. For instance, the network topology of Grids is simplified by a multi-tier [Ranganathan and Foster (2001), Rasool

[1]Results described in this chapter have been published in [Chai *et al.* (2017)]: Chai, Anchen, et al. "Modeling Distributed Platforms from Application Traces for Realistic File Transfer Simulation." Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing. IEEE Press, 2017.

et al. (2009), Lee *et al.* (2012)] or a sibling tree model [Lamehamedi *et al.* (2002), Fadaie and Rahmani (2012)]. Moreover, the network bandwidths are usually instantiated by a limited number of theoretical values [Shorfuzzaman *et al.* (2010), Camarasu-Pop *et al.* (2013)b, Dayyani and Khayyambashi (2015), Gupta *et al.* (2017)].

In this context, we propose a realistic network topology and an accurate bandwidth instantiation methods for modeling our target platform (i.e., the Biomed VO of EGI). Based on the information found in each individual execution trace, we generate ad-hoc platform models to replay the execution of each workflow. The realism of our platform models will be evaluated by confronting the simulation results to the ground truth of the actual executions registered in the traces. To ensure that the simulated file transfers occur between the same hosts as in the real execution, we inject the file replica involved in the corresponding real transfers as parameters for jobs in our simulator.

The rest of the chapter is organized as follows. In Section 4.2, we detail all the improvements based on observations from execution traces for constructing the realistic network topology and accurate bandwidth instantiation methods. The overall evaluation of our proposed model will be presented in Section 4.3.

4.2 Build partial platform model from execution traces

In this section, we detail incremental improvements to the platform model to increase the accuracy of file transfer simulation thanks to a thorough analysis of both trace contents and simulation results. In order to evaluate the impact of our proposed improvements, we first introduce a widely used model in literature which is considered as a baseline model for our work. Then the impact of each improvement for the accuracy of simulated file transfers are illustrated on a particular case extracted from the execution traces collected in Chapter 2.

4.2.1 Baseline model

The quality of file transfer simulation is mainly impacted by two main features of a platform model: the interconnection topology and the instantiation of network link bandwidth. To define a baseline for our study, we consider a first platform model that only minimally relies on the execution traces for two important features, i.e., the characteristics of the computing nodes (e.g., name or processing speed) extracted from the execution traces and their hierarchical organization in clusters and sites.

Without information on the interconnection topology coming from the traces, we have to assume a simple and uniform connectivity among the compute and storage nodes that compose the distributed platform. A way to model such a platform is to connect each SE to all the computing entities through a single backbone as shown in Figure 4.1.

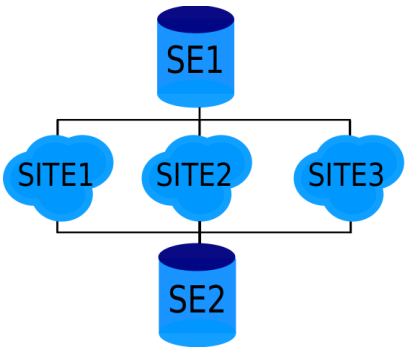

Figure 4.1: Graphical representation of the network topology for baseline model.

To instantiate different types of network links, we use the nominal bandwidth value of the network card for the compute nodes, which is typically of *1 Gb/s*. This information is extracted from the traces, but does not depend on a specific run. For the storage elements, we consider an interconnection bandwidth of *10 Gb/s*. The rationale is that SEs usually are large disk bays, hence with a better connectivity than simple compute nodes to ensure a good throughput. The latency for links is set according to the instantiation explained in Section 3.4.2 of Chapter 3 (i.e., 5 milliseconds for private links and 7.5 milliseconds for the external links). Such a model can be considered as state-of-the-art as it has been used for the simulation of similar workflows running on the same distributed platform [Camarasu-Pop *et al.* (2013)b, Camarasu-Pop *et al.* (2016)], but in a context where the focus was not on file transfers.

Figure 4.2 presents the file transfer durations (in seconds) of two of the input files downloaded by each of the hundred jobs composing one specific workflow instance, as logged in the traces and simulated with the aforementioned model. The respective sizes of these two files are 73 kB (wrapper file) and 121.5 MB (GATE release file). These transfers are representative of all the other file transfers for this workflow, but also of all the other studied workflows.

This figure illustrates two major drawbacks of relying on a "theoretical" bandwidth of 10 Gb/s to instantiate the platform model. First, the duration of the file transfers are globally and severely underestimated. For the release file, the simulated durations are in the [1.186s; 2.437s] range, while the measured transfer times are in the [2s; 223.5s] range. Similar differences occur for the wrapper file. Second, and more importantly, this model fails to reproduce the variability because it does not properly capture the competition for resources between certain transfers due to inaccurate transfer time estimations.

4.2.2 Improvements based on execution traces

To address the two issues raised by a uniform bandwidth instantiation of the link that connects a SE to the rest of the platform to a nominal value of 10 Gb/s, we exploit the

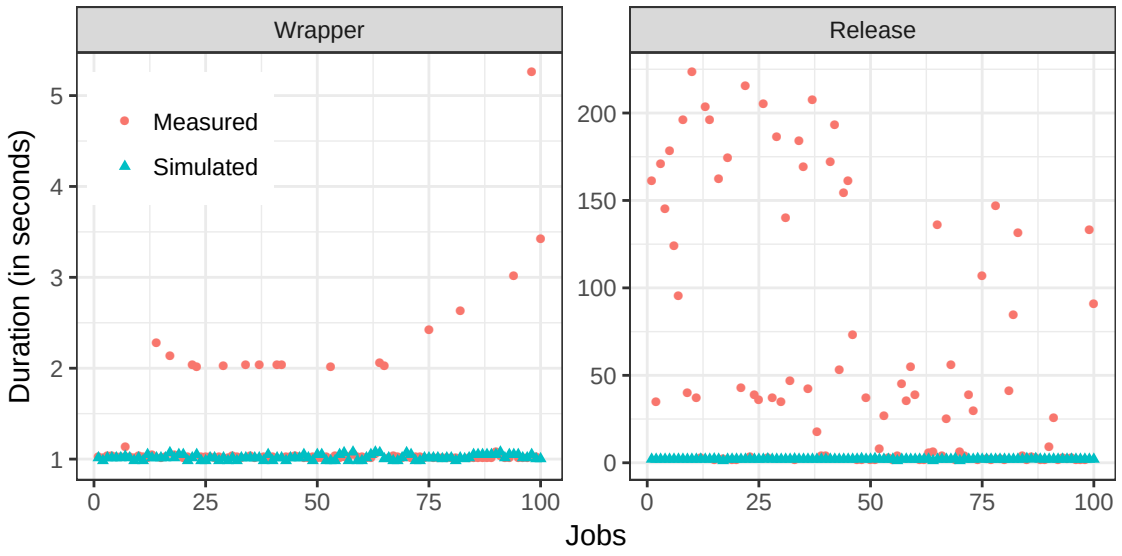

Figure 4.2: Measured[2] and simulated (with a 10 Gb/s link per SE) transfer durations for two input files downloaded by each job in the `fNUfzs` workflow instance.

contents of the execution traces. Our aim is at using values that are more representative of the actual execution conditions. We thus incrementally improve the baseline model by leveraging the traces.

Aggregation methods for instantiate links

We consider two common-sense instantiation methods, which have their respective pros and cons, to decide the bandwidth of each link. First, we compute the **inter-quartile average** (i.e., only data between the first and third quartiles is used) value over all the observed transfers to/from a given SE. Using average values is likely to be more realistic when it comes to reproduce the behavior of a single workflow execution. It reflects the network connectivity as experienced by the application. Inter-quartile average also allows to reduce the bias caused by outliers (abnormally long transfers). The sharing of network resources by concurrent transfers is thus directly captured in the model and not handled by the simulation kernel. The drawback of this average-based approach is that the resulting instantiation is limited to replay the file transfers in the workflows used to produce the average value and it will be less significant to simulate other workflow executions since they may experience different network connectivity due to different external traffic.

Second, we determine the observed **maximum** value over all these transfers. This allows us to get an approximation of the nominal capacity of the network link. In that case, we let the simulation kernel determine how the network resources are shared among concurrent transfers. The main advantage of such an instantiation of the platform model

[2] For the sake of readability, a data point corresponding to a transfer of the wrapper file that lasts for 65 seconds is not displayed.

is that it can be reused beyond the simulated replay of the execution that led to its generation. It can also be aggregated, both spatially and temporally, with information obtained from other traces.

Figure 4.3 presents the same type of results as Figure 4.2, but here the simulation results obtained with a uniform 10 Gb/s instantiation have been replaced by those obtained with an average or maximum bandwidth value for each storage element. We use the same interconnection topology.

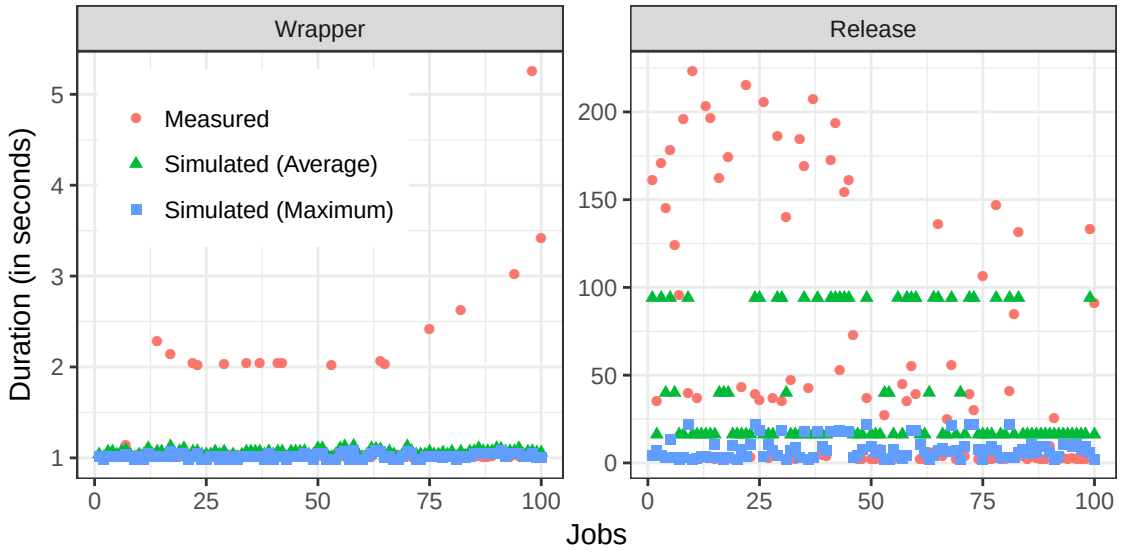

Figure 4.3: Measured[2] and simulated (with average or maximum bandwidth for each SE) transfer durations for two input files downloaded by each job in the `fNUfzs` workflow instance.

Using trace contents to instantiate the bandwidth values of the network links partially addresses the capture of the variability of file transfer duration, especially for the large release file. However, both the aggregation methods fail to be accurate. Averaging the observations tends to overestimate the transfer times (by a factor of 2 in average with a median of 1.5, and a maximum of 14.5), while using the maximum leads to an underestimation of at least a factor of 2 for half of the transfers. This denotes biases that require further investigation for the derived bandwidth for links.

Distinguishing transfer type and file size

We then looked at the distribution of the individual bandwidths derived from the file transfers that involve a given SE in a given workflow trace. We notice great difference in some cases, sometimes of more than two orders of magnitude. We identified three root causes to this variability. First, the transfer latency, which is one second (identified

[2] For the sake of readibility, a data point corresponding to a transfer of the wrapper file that lasts for 65 seconds is not displayed.

in Chapter 2), is negligible with regard to the total transfer time. However, for upload tests (i.e., 12-byte file transfers), this leads to unrealistically low bandwidth values, i.e., of less than a kilobit per second. Second, the limited precision of the timings logged in the traces leads to almost instantaneous transfer times for files of a few kilobytes. Then the derived bandwidth is unrealistically high, i.e., greater than 10 Gb/s. Third, we observed a sometimes large difference between the upload and download of the partial result files produced by the jobs. Each job uploads such a file to its local SE which is then downloaded by the merge job from the same SE. However, the concurrency conditions in which these transfers occur are different and impact the transfer times. The merge job downloads all the partial results in sequence while several worker nodes can upload their files to the same SE simultaneously. Consequently, depending on the direction of the transfer, i.e., to or from a given SE, the derived bandwidth may greatly differ.

All these observations are likely to negatively impact the computation of the average or maximum bandwidth of the link connecting a storage element to the rest of the platform.

To address these issues, we propose to group the transfers by type (i.e., upload test, each of the three input files, and both upload and download of partial result files) and direction (i.e., to or from a SE) before computing the average and maximum bandwidth values. Bandwidth estimations are then performed based on specific types for which the measured file transfer durations are assumed to be more reliable. For transfers from a SE, we favor that of the release file, whose larger size prevents to be hit by the timing precision. Similarly, for transfers to a SE, the upload of partial results is preferred over the more sensitive initial upload tests.

Figure 4.4 shows the impact of the average bandwidth computation method on the simulation of the transfer of the release file by jobs in the `fNUfzs` workflow instance. The transfer of this particular file is the one for which this improvement is the most noticeable, hence our focus on these results. The *Average (all)* data are the same as in Figure 4.3, while the *Average (grouped)* data refer to the distinction of transfer type and file size. In that case, the average bandwidth from a given SE to any compute site is computed based only on the transfers of the release file found in the traces. We also distinguish the SE from which the file has been downloaded in this figure.

The proposed improvement reduces the overestimation of the transfer duration. Simulated transfers, whose durations were twice as long in average (and up to 14.5 times longer) than the actual transfers and are now 1.2 times shorter in average (with a maximum overestimation by a factor 7.5). However, we can clearly see in Figure 4.4 that, for a given SE, the model still fails to capture the variability in transfer durations. To further polish our platform model, we focus on a specific SE in the next improvement.

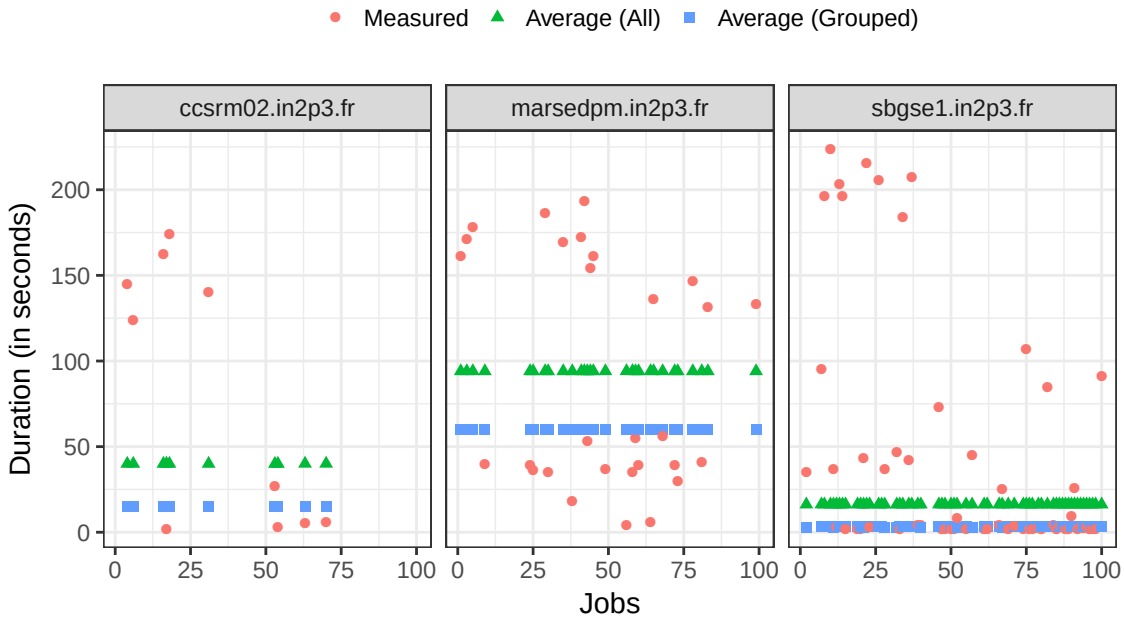

Figure 4.4: Measured and simulated (with global or grouped average bandwidth for each SE) durations for the download of the release file by each job in the `fNUfzs` workflow instance.

Distinguishing computing sites

From the fine-grained analysis of transfer durations in Chapter 2, we found that the connectivity to different computing sites from a given SE was very heterogeneous on EGI. Consequently, using one single backbone to connect each SE to all computing sites leads to an uniform transfer time to all the computing sites, even though the bandwidth of the backbone is computed by leveraging the trace contents (as in Figure 4.4).

The bottom part of Figure 4.5 shows an important variability in the average bandwidth when we distinguish the downloads of the release file from the `marsedpm.in2p3.fr` storage element in the `fNUfzs` workflow instance by *computing site*. The global average bandwidth for the SE, as computed in the previous section and depicted by a dashed line, is of 33.52 Mb/s. This value is a good approximation for the INFN-BARI and UKI-LT2-RHUL sites but an overestimation by a factor of around 5 for the INFN-PISA and UKI-LT2-IC-HEP sites and a clear underestimation for the UKI-LT2-QMUL site.

This large deviation to the global average requires to modify the network topology by separating the single backbone into distinguished links between a SE and the computing sites, which is depicted in Figure 4.6.

Simulation results with this new topology are presented in the top part of Figure 4.5. The "*By SE*" data corresponds to the middle panel of Figure 4.4 while the "*By SE-Site*" data relies on the distinct average value computed for each computing site. To improve

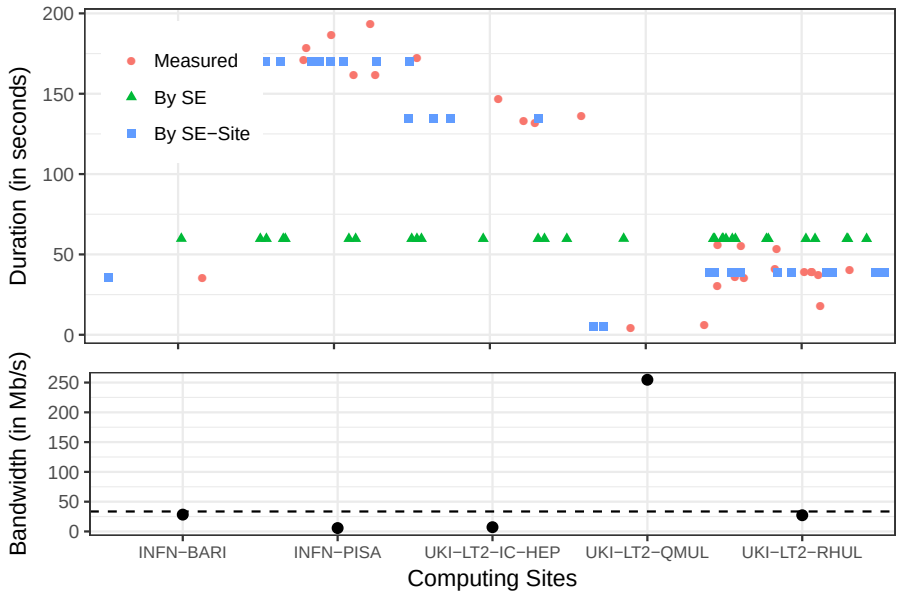

Figure 4.5: Measured and simulated durations for the downloads of the release file from the `marsedpm.in2p3.fr` SE in the `fNUfzs` workflow instance (top). Simulated times are obtained with a single **average** bandwidth (SE) and distinct **average** bandwidths (SE-Site) for each site (bottom).

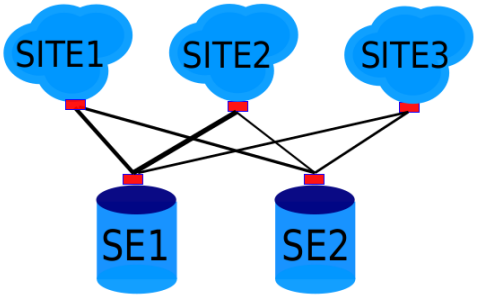

Figure 4.6: Graphical representation of the network topology after separating the single backbone. Red boxes represent the *limiter links*.

the readability of the figure, we introduce a horizontal jitter to separate data points corresponding to similar durations. We can observe that this improvement can significantly improve the accuracy of simulated transfer durations for the average-based model.

Now we make a similar analysis for the alternate approach based on the determination of the maximum observed bandwidth. As for the average, we see in Figure 4.7 (bottom) an important deviation from the global maximum value of 321 Mb/s for most of the computing sites. We also see in Figure 4.7 (top) that the use of a single value for all the sites also leads to similar simulated durations for all the transfers.

In this version of the model, it should be noted that distinguishing the links between a SE and the computing sites may bias the way the simulation kernel handles the sharing of

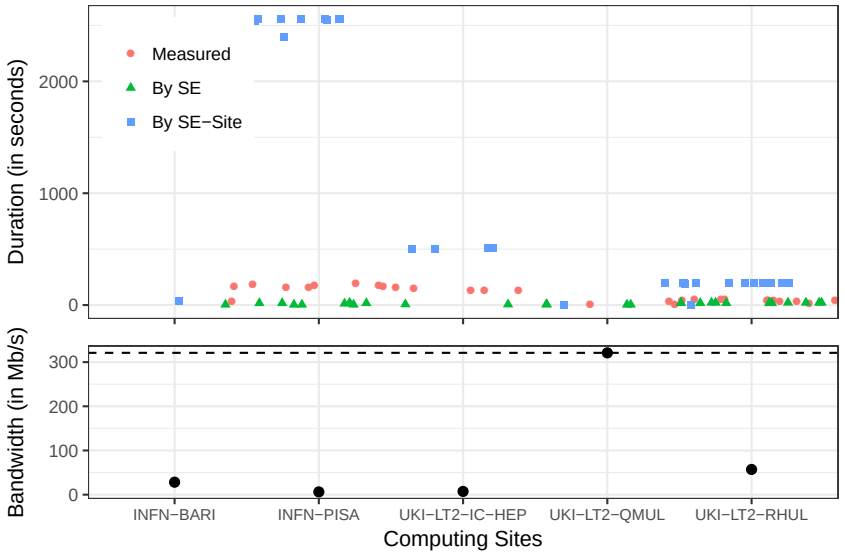

Figure 4.7: Measured and simulated durations for the downloads of the release file from the `marsedpm.in2p3.fr` SE in the `fNUfzs` workflow instance (top). Simulated times are obtained with a single **maximum** bandwidth (SE) and distinct **maximum** bandwidths (SE-Site) for each site (bottom).

the network resources. Indeed, separated links mean independent traffic and this can lead to a cumulative simulated bandwidth greater than what is observed from both the SE and computing site point of views. To prevent such a bias, we introduce the notion of *limiter* links whose bandwidths are determined by taking the maximum value observed over all the transfers to/from a SE or a given site (represented by red boxes in Figure 4.6).

Unlike the improvement observed for the average-based model, we see here that determining a distinct maximum bandwidth for each site dramatically degrades the quality of the simulation. This suggests that our estimation of the maximum value is biased and returns underestimations of the nominal bandwidths of the links between the storage element and the different computing sites.

More precisely, the modification of the model improves the simulation accuracy when there are only a few transfers from the SE to a computing site (i.e., for the INFN-BARI and UKI-LT2-QMUL sites). When there are more transfers from the SE to a computing site, the durations become, sometimes largely, overestimated. We thus assume that our model fails to capture an important phenomenon, that we identify and take into account in the next section.

Correcting the maximum bandwidth

Figure 4.8 presents the part of the Gantt chart of the execution of the `fNUfzs` workflow instance, when the nine downloads of the release file by worker nodes in the INFN-PISA site from the `marsedpm.in2p3.fr` SE take place.

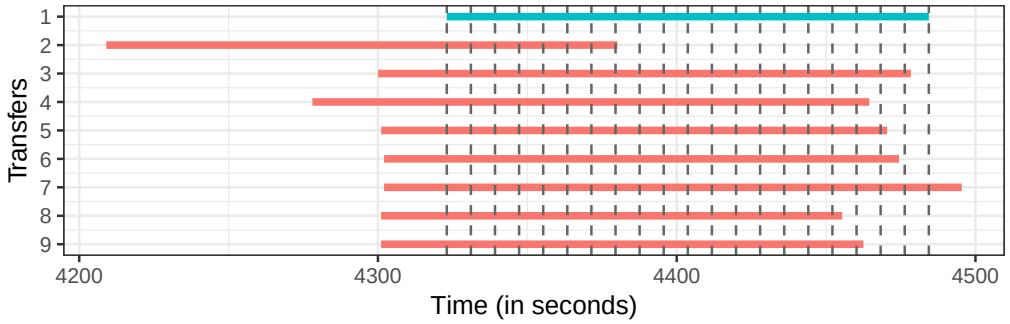

Figure 4.8: Gantt chart view of the transfers of the release file from the `marsedpm.in2p3.fr` SE to worker nodes in the INFN-PISA site.

We can see that almost all these transfers are simultaneous and of durations in the same range (from 154.4 to 193.4 seconds). As a consequence, the derived bandwidth of each individual transfer is impacted by the competition with the other transfers for a single shared network resource. In other words, the observed bandwidths correspond to the shares of the link capacity that are respectively allocated to each transfer, but they do not reflect the total capacity of the link itself.

To get a better estimate of the nominal capacity of the link, we thus have to first compute a correcting factor to the bandwidth derived from each transfer. The computation of this factor, during the generation of the platform model, somehow corresponds to the inverse of the computation made by the simulation kernel to assign a share of the network resource to a given transfer. We proceed as follows. First we sample the duration of the transfer in at most 50 uniform intervals. For instance, in Figure 4.8, we split the first transfer (in blue) in 20 intervals. For each of them, we estimate the number of concurrent transfers, hence how the network resource is shared. The first transfer will thus have only one ninth of the capacity of the link for the seven first interval but one third of it during the last interval. Then we compute the correcting factor as:

$$f = \frac{n}{\sum_{i=1}^{n} \frac{1}{ci}}, \tag{4.1}$$

where n is the number of intervals, and c_i the number of concurrent transfers in the i^{th} interval. In our example, we obtain a correcting factor of 20/(7/8+10/7+1/6+1/4+1/2) = 6.21 for the first transfer.

For each transfer, we compute three different correcting factors by estimating the concurrency for each SE, each site and each (SE, Site) couple. This allows us to correct the instantiation of the link between a SE and a site and of the limiter links.

Figure 4.9 shows the impact of this correction of the maximum observed bandwidth on the results presented in Figure 4.7. We can see that the important overestimation of the transfer durations disappears and the simulation accuracy is globally improved.

However, this model now leads to a moderate underestimation of the durations for two sites (INFN-PISA and UKI-LT2-RHUL), which is caused by different phenomena.

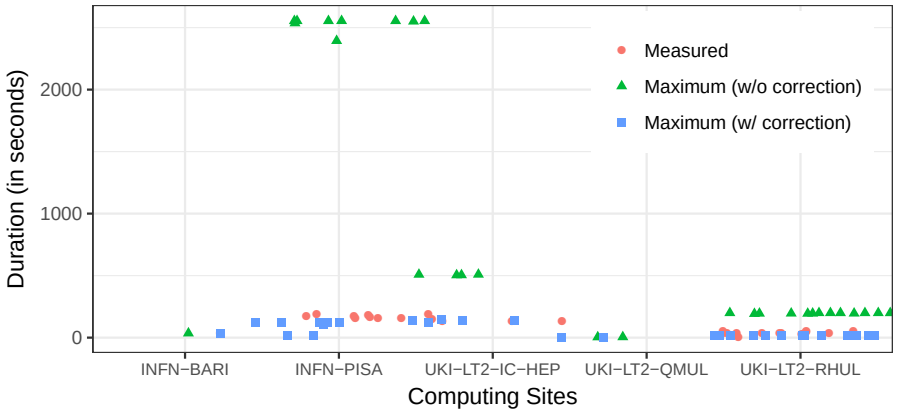

Figure 4.9: Measured and simulated durations for the downloads of the release file from the `marsedpm.in2p3.fr` SE in the `fNUfzs` workflow instance with and without correction of the maximum observed bandwidth.

For the INFN-PISA site, two transfers (second and fourth from top in Figure 4.8) start before a bulk of seven transfers. While these two transfers last for roughly the same time as the others in the real execution, they will experience less concurrency in simulation and will then be faster to complete (in around 20 seconds). More importantly they will end before the other transfers start, which in turn impacts the concurrency they experience and their simulated duration. This phenomenon corresponds to the external network load which has been identified in Chapter 2 and that our model currently fails to capture.

For the UKI-LT2-RHUL site, we observed that the transfer durations differ depending on which cluster in the site is involved. We propose to illustrate and address this phenomenon in the next section using a more glaring example.

Distinguishing clusters in sites

As we have found in Chapter 2 that compute nodes in different clusters might not share one common link inside a computing site, using a single bandwidth value for the site, be it the average or the maximum, still fails to capture that cluster-related difference in performance, and leads to inaccurate simulations (as shown by "By Site" in Figure 4.10).

To improve our models, we propose to differentiate the links that connect the different clusters in a site to the rest of the platform (see Figure 4.11). Note that the clustering of worker nodes is based only on information available in the trace, i.e., name, number of cores, and processing speed, and do not leverage information by the sites themselves. As for the previous improvements, we modify the way we aggregate the individual bandwidth values to take this distinction into account. We also compute specific correction factors for the model based on the maximum bandwidth.

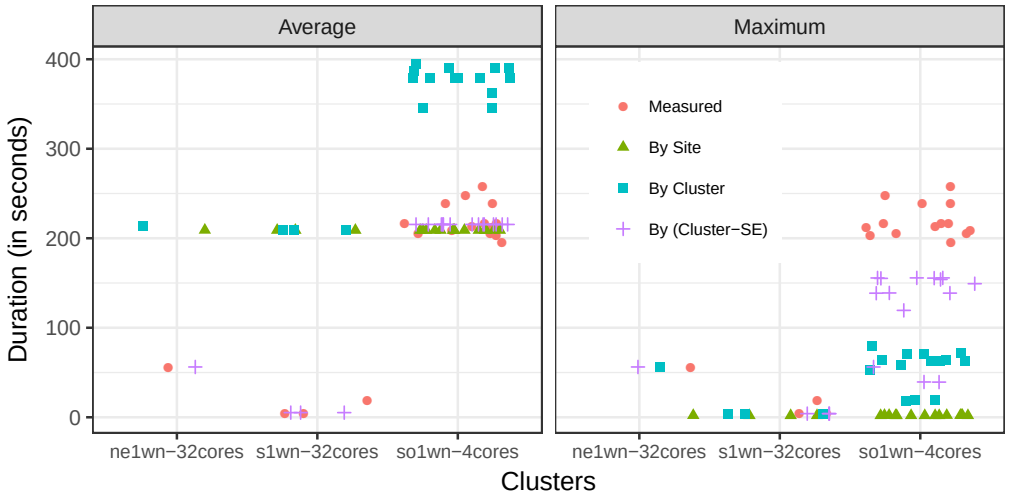

Figure 4.10: Measured and simulated durations for the downloads of the release file from the `sbgse1.in2p3.fr` SE to the INFN-PISA site in the `LQz3XJ` workflow instance with and without distinction of the clusters.

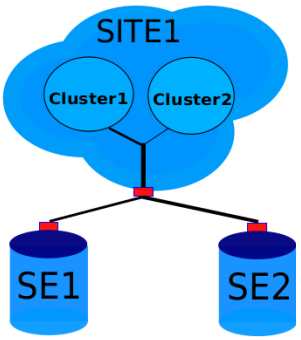

Figure 4.11: Graphical representation of the network topology by differentiating the links within a computing site. Red boxes represent the *limiter* links.

Unfortunately, this modification improves the accuracy of the simulations only partially. Figure 4.10 highlights two limitations with different causes but a common solution. For the average-based model, we observe an overestimation of the durations for the `s1wn-32cores` cluster. This is because the link between the SE and the site becomes a bottleneck. Indeed, the average bandwidth we compute for this link is hindered by the bad performance obtained for the `so1wn-4cores` cluster. Conversely, in the maximum-based model, we see an underestimation of the durations for the `so1wn-4cores` cluster. In this case it is the better bandwidth from another SE (not displayed in Figure 4.10) that defines the maximum observed bandwidth for this cluster. These two situations advocate for a common improvement that consists in differentiating the links not between a site and a SE, but between each individual cluster and a SE.

The results obtained with this final improvement are labeled as "*By Cluster-SE*" in Figure 4.10. We can see that, as expected, it completely solves the inaccuracy issue

for the average-based model. For the maximum-based model, the proposed distinction improves the accuracy only partially. This can be explained by some abnormally long transfers, whose causes have been summarized in Chapter 2, in the real execution that have a shorter simulated duration which in turn modifies the level of concurrency on the network resource. We can see three such transfers in Figure 4.10.

4.3 Overall evaluation of our model

After illustrating the improvement of each proposed modification on a particular case, we then evaluate our improved platform models by confronting simulation results obtained using these models to other simulation results using the *baseline* model and to the reference given by the real transfer durations extracted from traces. We consider 24 workflow instances, which match the current capabilities of our simulator presented in Chapter 3, out of the 60 collected workflows. We discard 31 workflows in which there exist particular failures (job or file transfer) and 5 workflows that have been severely impacted by the scheduled maintenance shutdown of a specific SE ("ccsrm02.in2p3.fr"), which was identified in Chapter 2.

4.3.1 Analysis of simulated transfer durations

We evaluate the simulated transfers obtained from three platform models: the *baseline* (named 10G-SotA), *Average* and *Maximum* model (our final proposed topology shown in Figure 4.11 by using different instantiation method). For each model, we simulate 5,316 download transfers. Among these, 1,772 files (one third) correspond to the GATE release file and are larger than 121 MB. The other files correspond to the GATE inputs and wrappers and are smaller than 130 kB. Figure 4.12 shows a summary of the measured and simulated transfer durations for these two sets of files.

For the GATE inputs and wrappers files, 80% of these 3,544 small file transfers last for less than 1.3 seconds in the real execution. The minimal incompressible delay imposed by control and network latency being of one second, there is no possible discrimination between the models and very few mis-estimations. We also observe 26 transfers (of the 73 kB wrapper file) whose durations are close to 64.5 seconds. The only common point to all these transfers is the storage element use to download this wrapper file. This highlights an issue with this specific SE rather than a modeling problem. However, it is interesting to note that both the *Average* and *Maximum* models are able to correctly reproduce one of this longer transfers. The proposed models are also able to partially capture the variability of 15% to 19% of the remaining transfers, while the *10G-SotA* model cannot.

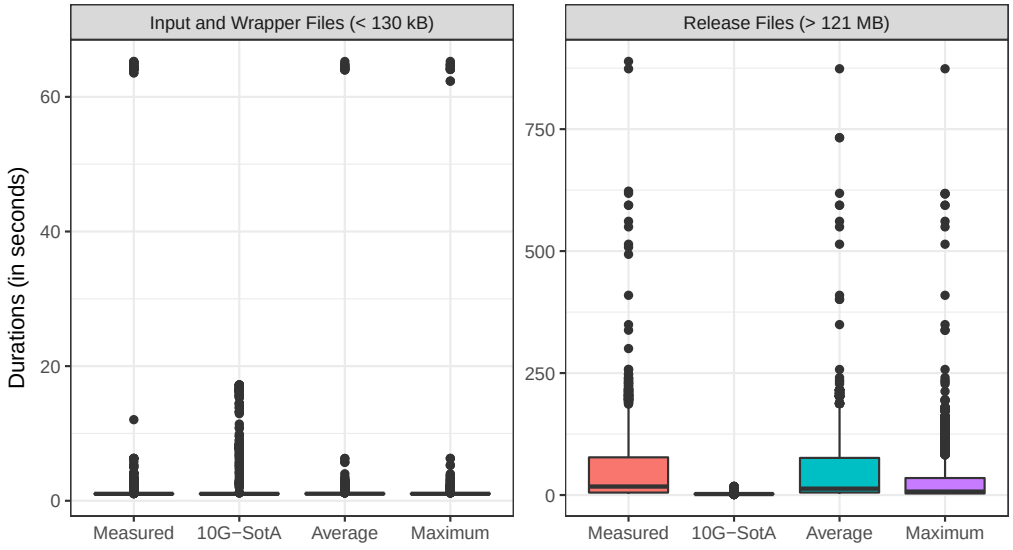

Figure 4.12: Graphical summary of measured and simulated (with the 10G-SotA, Average, and Maximum models) file transfer durations (in seconds).

In the remaining of this section, we focus our analysis on the transfers of the larger GATE release file. The longer durations allow us to get a better insight on the respective strengths and limitations of the different models.

We complete Figure 4.12 with Table 4.1 that gives the corresponding statistics for the transfers of the release file. The poor quality of the *10G-SotA* model is blatant: it largely underestimates all the transfer durations and cannot capture the variability that characterizes the real executions. The proposed *Average* and *Maximum* models, however, are able to reproduce that variability and correctly simulate the longest transfers. We also note a tendency of the *Maximum* platform model to underestimate the transfer durations, which can be explained by the fact that this models relies on estimations of the nominal bandwidths of the links that are less accurate than the average of the bandwidths as perceived by the application.

Table 4.1: Statistics of measured and simulated (with the 10G-SotA, Average, and Maximum models) durations (in seconds) of the transfer of the GATE release files (> 121 MB).

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
Measured	2.00	5.01	17.42	49.91	77.31	888.80
10G-SotA	1.19	2.02	2.03	2.55	2.15	11.52
Average	2.00	5.13	13.12	48.19	76.11	873.80
Maximum	1.98	3.21	7.44	31.72	35.09	873.80

4.3.2 Analysis of errors

Another way to assess the respective quality of the different platform models and to measure the benefits of the proposed trace-based approach in terms of simulation accuracy is to compare errors with respect to the real transfer times. The relative difference between simulation results and the reference of a real execution can be computed in many ways. For instance, the relative error is a standard error measure which gives a good indication on the precision of a simulation and whether it under- or overestimates the reality. However, this method has a certain number of disadvantages such as the fact that it is not symmetrical (the error lies in the interval]-∞, 1]) or the fact that the relative difference of bandwidths is different of the relative difference of transfer times, while there is a priori no reason to favor one over the other. As explained in [Velho and Legrand (2009)], an absolute logarithmic error solves these two issues and allows us to compare the different models more easily. We define the absolute logarithmic error as follows:

$$LogErr = |\log(R) - log(S)|, \tag{4.2}$$

where R is the real time and S the simulated time.

The absolute logarithmic error also allow us to apply additive aggregation operators such as the maximum or the mean, hence easing the comparison of the different models.

Figure 4.13 shows the Cumulative Distribution Functions of the absolute logarithmic errors obtained for each model. The maximum errors respectively are 6.09 for the *10G-SotA* model, 3.99 for the *Maximum* model, and 2.83 for the *Average* model.

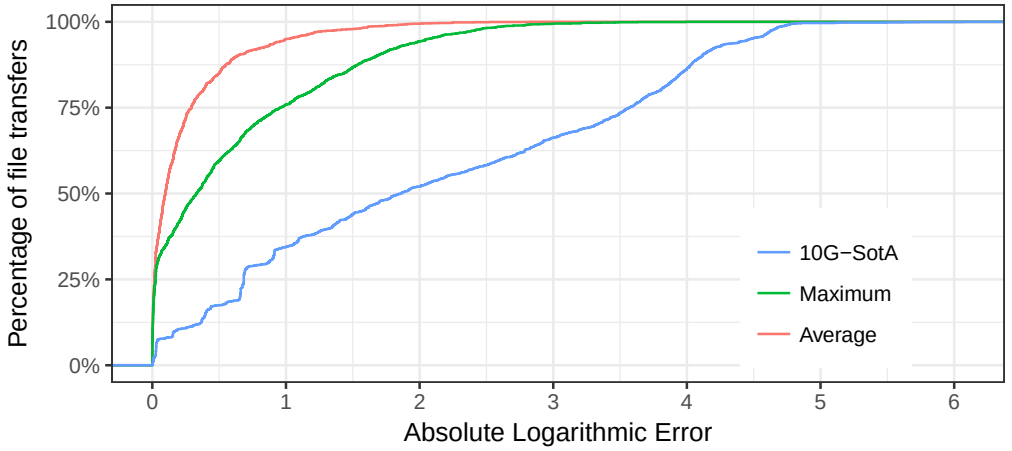

Figure 4.13: Cumulative Distributed Functions of the absolute logarithmic error achieved by the three platform models over the whole set of transfers of the release files.

This figure confirms the poor accuracy of the *10G-SotA* platform model that shows errors greater than two (i.e., relative error greater than 639%) for nearly half of the transfers. This corresponds to the systematic and large underestimations of the file transfer

durations made by this model that was illustrated by Figure 4.12.

As expected, the *Average* platform model leads to best results as the sharing of network resources by concurrent transfers is directly captured in the model which reflects the network connectivity as experienced by the application. This model is thus able to simulate 75% of the transfers with a logarithmic error smaller than 0.29 (relative error: 33.8%) and 92.9% of the transfers with an error smaller than 1. Finally, the mean logarithmic error is 0.23 (relative error: 25%).

The *Maximum* platform model also clearly outperforms the *10G-SotA* model but leads to greater errors than the *Average* model. The mean logarithmic error is 0.62 (relative error: 85%) and 75% of the transfers are simulated with a logarithmic error under 0.97 (relative error: 164%). This loss of accuracy is the cost of the higher potential for further studies of this model that offers a better reusability beyond the simulated replay of the execution that led to its generation. It also confirms the tendency of this model to be too optimistic in its determination of the bandwidth values, and thus to underestimates the transfer durations. A deeper analysis of the individual simulation results would be needed to determine whether the way to compute the bandwidth correction factor can be improved.

To summarize, our analysis of the distribution of the simulated transfer durations and of the logarithmic errors demonstrates that a correct modeling of the interconnection topology and a good instantiation of the characteristics of the network resources are key to the quality of simulations of file transfers on a large-scale distributed infrastructure. It also shows that leveraging the contents of execution traces is a sound approach that enables a pretty accurate simulated replay of a given execution with the *Average* model or even beyond with the *Maximum* model, with an affordable accuracy loss.

4.3.3 Analysis of the root causes of large simulation errors

While the obtained results are promising and greatly outperform the state-of-the-art model, they also include some large errors. In this section, we analyze such errors to determine whether they correspond to a modeling flaw or to phenomena that we cannot, or do not even want to, model. For instance, the cause behind the outlying durations for small file transfers does not have to be part of the platform model, but rather to be injected into the simulation as an additional parameter. After carefully analyzing the obtained simulation results from our platform model, we identify two source of high inaccuracy.

The first one is the **external network load**. As identified in Chapter 2, external load may lead to abnormally long transfers. We illustrate the identified case in Figure 4.14.

In the *Maximum* model, the bandwidth derives from transfer 2 and is, after correction, of 968 Mb/s. However, the actual bandwidth for transfer 5 is only of 23.5 Mb/s. This

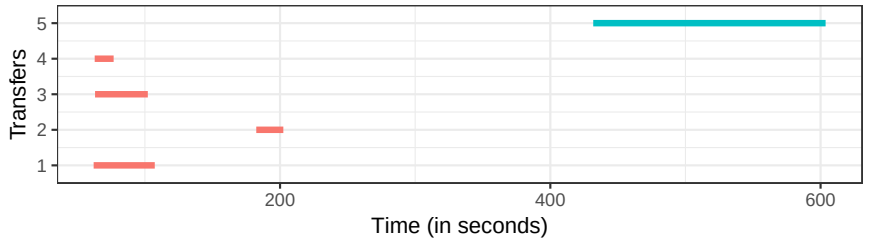

Figure 4.14: Gantt chart view of 5 release file transfer durations for a given SE, a given site, and a given cluster.

difference leads to a dramatic underestimation of the simulated duration (only 5.15s) and one of the largest errors. With the *Average* model, this also impacts the simulation accuracy by lowering the computed average (at 163 Mb/s). Again the great deviation from the actual bandwidth leads to an important error.

The second identified source of large errors corresponds to the observation of "**staircase**" phenomenon, i.e., the durations of simultaneous transfers exponentially increase (as shown in Figure 4.15).

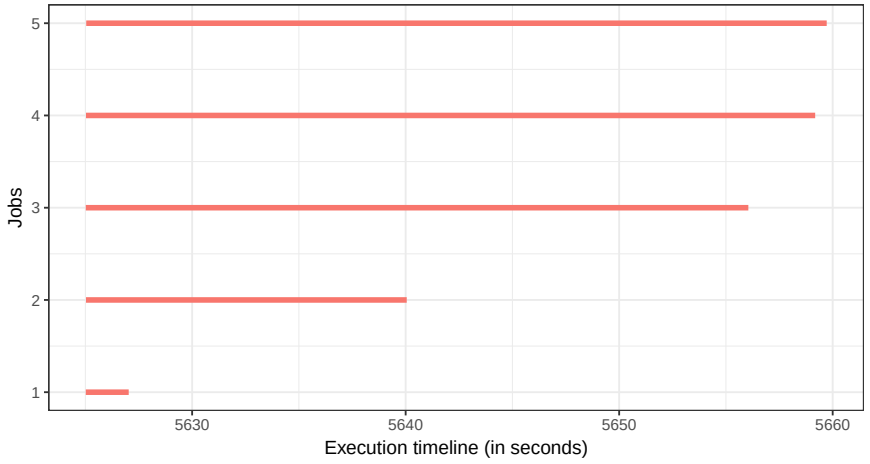

Figure 4.15: Gantt chart view of release file transfer durations for a given SE, a given site, and a given cluster.

In current simulations, with *Maximum* or *Average* model, simultaneous transfers from a given SE to jobs in a given cluster will have similar simulated durations, which therefore causes large errors compared to real transfer durations. However, both of these identified sources of high inaccuracy are not related to our platform model. The first one can be addressed by introducing a certain level of background network traffic for simulating the workflow execution, while the second one is related to the configuration of simulated SE in simulator which was explained in Chapter 3.

4.4 Conclusion

In this chapter, we presented our strategy to build realistic platform models to replay the execution of each individual workflow with a focus on file transfers on EGI. In order to minimize the bias for the simulated file transfers, we generate ad-hoc platform models for each workflow and inject the file replica involved in real transfers as parameters for jobs in our simulator built in Chapter 3.

We evaluated these platform models are evaluated by comparing the simulated and real transfer durations. Simulation results show that our proposed network topology and bandwidth instantiation methods are able to correctly reproduce real-life variability of file transfers on EGI, as opposed to the state-of-the-art platform model. Indeed, both the Maximum and Average models manage to correctly capture the distribution of the transfer durations. The analysis of the absolute logarithmic errors also shows that the proposed platform models clearly outperform the state-of-the-art model, which largely underestimates a vast majority of the transfer durations.

In the next chapter, we will present our method to build a more general and complete platform description based on the proposed network topology and bandwidth instantiation methods.

Chapter 5

Towards a complete and realistic description of the Biomed VO platform

Abstract. *This chapter presents our method to construct a complete platform description beyond the ad-hoc models proposed in Chapter 4 by aggregating multiple execution traces. We also propose a model to fill-in the bandwidth of missing links after the trace aggregation. Predicted bandwidths of missing links follow the network hierarchy and heterogeneity presented by known links. The complete platform description generated in this chapter will allow us to evaluate different "what-if" scenarios for file management of applications in Chapter 6.*

5.1 Introduction

In Chapter 4, we proposed and evaluated different network topologies and bandwidth instantiation methods for building realistic platform models for each individual workflow. These ad-hoc platform descriptions are very useful for replaying existing scenarios of file transfers. However, in order to evaluate different "what-if" scenarios for improving the data management of VIP, we need a more general and complete description in which any computing node can communicate with any SE. Therefore, our aim in this chapter is to build a complete description of the platform by leveraging information from multiple traces.

To better explain the problem, we illustrate it with an example. Let us take the traces from three workflows. As shown in Figure 5.1, we have three partial descriptions of the Biomed VO platform independently generated from the execution traces of workflow 1 (WF1), workflow 2 (WF2), and workflow 3 (WF3). Each description only contains partial

information about the network links. For instance, in workflow 1, jobs are executed in two computing sites (i.e., CE1 and CE2) and they use two storage elements (i.e., SE1 and SE2) to download the required files. It is to note that all jobs in CE1 only use SE1 while jobs in CE2 only use SE2. As a consequence, the partial description generated from WF1 only contains network information for links (CE1, SE1) and (CE2, SE2).

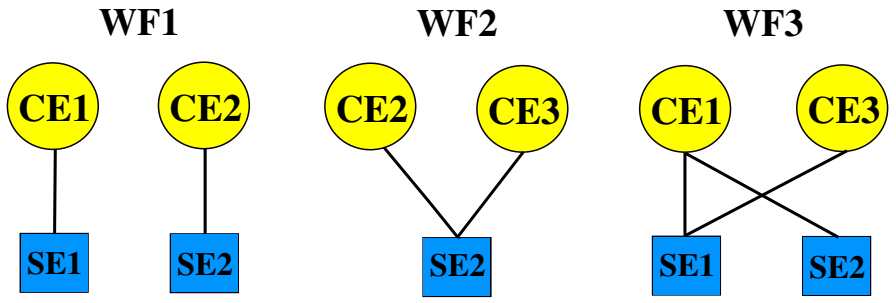

Figure 5.1: Three partial platform descriptions generated from execution traces of workflow 1 (WF1), workflow 2 (WF2), and workflow 3 (WF3). Each line corresponds to the network link between a site and a SE.

In order to obtain a larger description of the platform than those produced from each execution trace, we first merge these three partial descriptions. The merged platform from WF1, WF2, and WF3 is illustrated in Figure 5.2. In this merged platform, all distinct computing sites and SEs concerned by file transfers from all workflows are described. This aggregation of multiple traces allows us to fully exploit the information about network links. For instance, we can find information on the (CE1, SE2) link from WF3, which was not available in WF1.

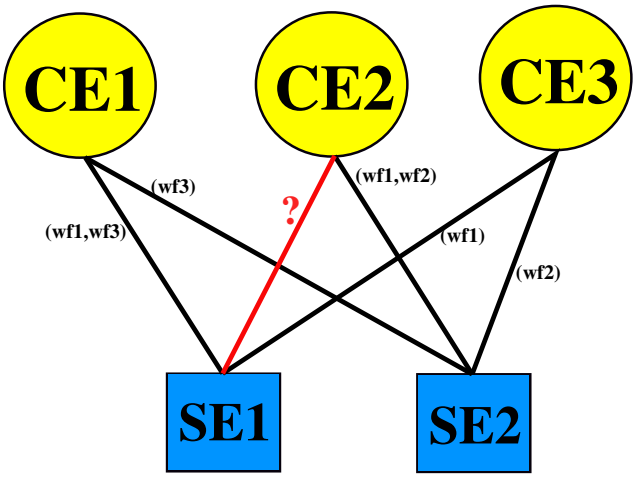

Figure 5.2: The merged platform description from the execution traces of workflow 1 (WF1), workflow 2 (WF2), and workflow 3 (WF3). Red line corresponds to the network link without any information from these traces.

However, two problems need to be addressed for this merged platform. First, we may find different information about the same network link in different traces. For instance, we have information for the (CE1, SE1) link in both WF1 and WF3. Hence, we need to determine how to instantiate the bandwidth for this link based on the information from two traces (wf1 and wf3). Second, even when aggregating multiple traces, we cannot guarantee that we will have sufficient information to generate a complete platform, with all possible link combinations. For instance, the information on the (CE2, SE1) link is not provided by any of these traces. As our aim is to generate a complete platform description, in which any computing node can communicate with any described SE, we need to fill-in all missing information in the merged platform.

In last chapter, we proposed to distinguish the network links of different clusters within a site and also the links of these clusters to the same SE. However, when we attempt to generate a complete platform description, the second distinction generates many missing links as we need to connect each cluster to all described SEs. Therefore, in this chapter, we decide to no longer distinguish the links of different clusters in the same site to the same SE. We focus on describing the network links for each pair of (CE, SE).

So in the rest of this chapter, we first address the problem of how to determine bandwidth instantiation for links with information from multiple traces in Section 5.2. Then different methods to fill-in missing links will be presented and evaluated in Section 5.3.

5.2 Aggregating network information from multiple traces

Trace aggregation enables us to combine multiple pieces of information from both a spatial and a temporal perspective in order to generate the description of a larger platform. Spatial aggregation allows us to describe more hardware resources (e.g., computing cluster, computing site, SE, etc.) and network links from all transfers with different sources and destinations. On the other hand, the temporal aggregation of traces helps us to determine the variability of several parameters (e.g., the bandwidth of links, the available storage space of SE, etc.) by leveraging time-related information. Here, in addition to spatially aggregate more information about hardware resources, we focus on determining the bandwidth instantiation for links with information from multiple traces. Integrating the temporal variability for bandwidths of network links is considered as a perspective.

Based on the proposed instantiation methods for bandwidths (i.e., inter-quartile average and maximum) in Chapter 4, we also consider two aggregation methods to exploit the information of the same network links from different traces: **Average_merged** and **Maximum_merged**. In the *Average_merged* method, we consider the aggregation of network information at the level of bandwidths that are already derived by inter-quartile average for each link in the partial descriptions generated in Chapter 4. We compute the average value for each network link by favoring the bandwidths derived from the transfer of

release file. If no bandwidth is derived from the release file transfer, we take the average of bandwidths derived from any other files. For instance, to determine the bandwidth of the (CE1, SE1) network link in Figure 5.2, we take the derived bandwidths of this link from WF1 and WF3. If these two bandwidths are all derived from release files, we compute their average value.

On the other hand, we aggregate network information at the level of file transfers in the *Maximum_merged* method. We assume that the maximum bandwidth observed from multiple traces for a given network link corresponds to a better approximation of its nominal capacity. Based on the numerous information recorded in the execution traces, we are able to retrieve the time stamp, which represents the global starting point of jobs, and the start time, which represents the relative starting point of jobs within the workflow execution. By combining these two pieces of information and the registered transfer durations, we are able to deduce the global starting time of each transfer. Then based on the bandwidth correcting method presented in Section 4.2.2 of Chapter 4, we compute a correcting factor for each transfer by taking into account all the measured transfers from either the same or different workflows. The bandwidth for a given link in the merged platform is thus the maximum derived value from all transfers going through this link.

By using these two methods, we generate the Average_Merged and Maximum_Merged platforms based on the 24 execution traces. Then we use these two merged platforms to reproduce the real file transfers as we did in Chapter 4. As we are always in the *reproducing* phase, we inject the file replica involved in the corresponding real transfers as parameters for jobs in our simulator to prevent from using missing links during the simulations.

Figure 5.3 shows a summary of the measured and simulated transfer durations of release files in 24 workflows with the two merged platforms. We observe that the *Average_Merged* model can correctly reproduce the overall distribution of the measured release file transfer durations, except for some large transfer durations. However, most of release file transfers are underestimated by the *Maximum_Merged* model. As our collected traces are over a period of one month, the external network traffic for file transfers occurring at different hours and days could be highly variable. The **Maximum_merged** method looks for the maximum bandwidth experienced by file transfers across multiple traces for a given link. We can thus easily derive a large bandwidth from one specific file transfer experiencing less network traffic for a given link, which will underestimate most of other file transfers without integrating the temporal variability of bandwidths. In the **Average_merged** method, we first compute the inter-quartile average of bandwidth for a given link in one workflow. This average value can reduce the impact of outliers (i.e., extremely short and long transfers). Then we take the average value of these inter-quartile bandwidths from each workflow for a given link. It is the simplest way to implicitly take into account the temporal network

information across different traces. The instantiations in the *Average_Merged* model can be considered as coarse estimations for the average available bandwidths of network links over a month experienced by all workflows. Therefore, this model can still reproduce the overall distribution of the measured file transfer durations.

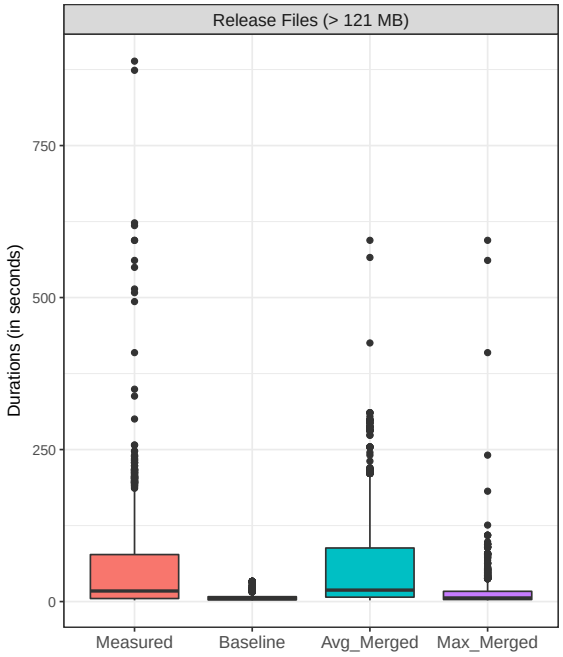

Figure 5.3: Graphical summary of measured and simulated (with the Average_Merged, and Maximum_Merged models) release file transfer durations (in seconds). Each point corresponds to the duration of one simulated transfer.

Table 5.1 gives the statistics for the transfer durations of the release file with merged platforms. We notice that the *Average_Merged* model tends to slightly overestimate most of the transfers, with a median value of 18.88s and a third quartile value of 88.14s compared to 17.42s and 77.31s for the actual measures, respectively. With the *Maximum_Merged* model, the third quartile value drops to 16.86s, which is 4.6 times shorter than what was measured. The impact of trace aggregation is thus much stronger for the *maximum* based model than *average* based model.

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
Measured	2.00	5.01	17.42	49.91	77.31	888.80
Average_Merged	2.05	7.26	18.88	54.98	88.14	594.18
Maximum_Merged	1.97	3.35	5.96	13.44	16.86	594.18
10G-SotA	1.19	2.02	2.03	2.55	2.15	11.52

Table 5.1: Statistics of measured and simulated (with the 10G-SotA, Merged average, and merged maximum models) durations (in seconds) of the transfer of the GATE release files ($>$ 121 MB).

Another remarkable aspect is that both merged models fail to reproduce the longest transfer (888.8s) which corresponds to a 121.5 MB release transfer between the "marsedpm.in2p3.fr" SE and a computing site in UK in the Njp5KF workflow instance. In the partial description built for this workflow, the derived bandwidth for the corresponding link was 1.3 Mb/s and 3.2 Mb/s in *Average* and *Maximum* model, respectively. However, after the aggregation of multiple traces, the bandwidth of this link is now 8.9 Mb/s and 331.4 Mb/s in *Average_Merged* and *Maximum_Merged* model, respectively. It shows that without integrating the temporal variability of bandwidths, the merged models will lose the information on extreme cases.

To summarize, the *Average_Merged* model can correctly reproduce the overall distribution of the measured release file transfer durations with a slight overestimation. The *Maximum_Merged* model allows for a better estimation of the nominal capacity of links by taking into account the network sharing across multiple workflow executions. However, without integrating the information of temporal variability of bandwidths, the *Maximum_Merged* model is likely to largely underestimate transfer durations. Moreover, extremely long transfers are not properly reproduced by both models. But still, these two merged platforms outperforms the state-of-the-art model (10G-SotA) to reproduce the real-life variability of transfer durations.

5.3 Filling-in missing links in the merged platform

Aggregation of multiple execution traces enables us to merge partial descriptions derived from each individual workflow. However, as illustrated in Figure 5.2, it does not allow us to generate a complete description due to two main reasons. First, each execution trace only contains partial network information and we can hardly find two traces holding completely complementary information. Therefore, aggregating traces will produce new network links for which no information can be found. Second, some links may never be used for a given computing site. As presented in Chapter 3, jobs favor file replica in local SE, then in the same country, and lastly in foreign countries according to the replica selection service of EGI. If a file replica is available in the same country as the computing site, jobs in this site will never download file from a replica in foreign country. Therefore, we can only extract very limited information of the inter-country links for the site even from multiple traces. Because of these two problems, we can not envisage to obtain a complete platform descriptions by simply collecting more traces.

So in this section, we will exploit the existing bandwidths of known links in the merged platform to estimate the missing bandwidths of the newly created links. Three predictive models are proposed. In order to study the prediction accuracy of each model, we first use each model to estimate the bandwidths of some known links based on existing information.

Then we compare these estimated bandwidths with the bandwidths obtained from the previous section for known links. The difference between the estimated and obtained bandwidths of known links is used as a metric to evaluate the prediction accuracy of each model.

5.3.1 Empirical model

On EGI, each site declares a SE as its local storage, which is usually in the same network domain as the computing site. If a site does not contribute with any storage resources to the Biomed VO, it will choose the closest SE with a general connectivity as good as the local one. Hence, our assumption is that the network connectivity of a computing site is similar to the connectivity of its local SE to other sites. Besides, there exists at least one measured bandwidth for each computing site because sites described in the merged platform executed at least one job and therefore we have at least one file transfer to each site. Based on these assumptions, we first attempt to predict the bandwidth of missing links at the level of each computing site. An empirical model is proposed to meet the following requirements:

- The measured bandwidth of links to/from a given site S_i should reflect its general network connectivity;
- For a given site S_i, its connectivity to different SEs should follow the network hierarchy observed in Chapter 2, i.e., a clear variation of bandwidths from local to national and inter-country links;
- The reliability for the estimation of bandwidth for site S_i should depend on the number of known links.

To determine the bandwidth of each missing link for a site S_i, we first classify all known links of this site into three categories (i.e., local, national, and inter-country). For each category c, we estimate the connectivity of a site S_i as the ratio between the median bandwidth of the known links to/from S_i and the median bandwidth of all the links in merged platform: $R_i^c = \widetilde{B_i^c}/\widetilde{B^c}$. Using the median value can reduce the impact of extremely good or bad bandwidths in a site and R_i^c represents the relative connectivity of S_i among all links for the category c. As we assume that the measured bandwidth of links can reflect the general network connectivity of one site, we then use these ratios R_i^c to predict the bandwidths of the missing links.

However, not all the sites have known links for each of the three categories. In order to be comparable among all sites, we need to combine the category ratios of each site into one metric. As we consider that if there are more known links for a category c in S_i, its ratio R_i^c will be more meaningful and reliable to represent the general connectivity of S_i,

we thus decide to weight each ratio by $|L_i^c|/|L_i|$, where $|L_i^c|$ is the number of known links for category c of site S_i and $|L_i|$ is the total number of known links of site S_i. The overall connectivity of S_i with regard to the rest of the platform is then estimated by the following weighted sum:

$$C_i = \sum_c \left(\frac{|L_i^c|}{|L_i|} . \frac{\widetilde{B_i^c}}{\widetilde{B^c}} \right). \tag{5.1}$$

Finally, the bandwidth of a missing link of category c to/from S_i is computed as the median bandwidth of all the links in this category in the merged platform times the overall connectivity: $\widetilde{B^c} \times C_i$. As we independently predict the missing links for each link category, the estimated bandwidths will follow the network hierarchy observed in Chapter 2.

5.3.2 Machine learning model

The empirical model has been designed to reflect the hierarchical topology of the platform and the overall connectivity of a site concerned by missing link(s). However, it has several limitations because of its strong hypotheses: (i) it is a simple linear model, (ii) the linearity depends on the number of known links in each category, and (iii) it does not take the connectivity of SEs into account. To address these issues, we decide to use machine learning techniques to construct more accurate models to predict the bandwidths of missing links based on the empirical model.

One consequence for not taking information on SEs into account in the empirical model is that the predicted bandwidths for the missing links of a given computing site will always be the same within a given link category. To address this problem, we need to integrate the network information of the corresponding SE for each link. However, as we have already distinguished the three link categories at the level of the computing site, we should not make the same distinction at the SE level as it may create redundant information. For instance, each site has only one local SE and therefore there exists only one local link for each site and each SE. We will obtain exactly the same information about this local link at the site and SE level. This situation can also happen for national and inter-country links when the network information is symmetric between a site and a SE, e.g., a site has only one foreign SE and this SE has only one inter-country link to this site. In order to avoid such redundant information, which may lead to over-fitting of machine learning models, we decide to consider one more attribute for each link: the general connectivity of the corresponding SE, which is represented by the median value of all bandwidth to/from this SE. Then we assume that the bandwidth B_{ij} of a link between $Site_i$ and SE_j can be expressed by:

$$B_{ij} = f(link\_category, Site_i, \widetilde{B_i^c}, \widetilde{B_j}) \tag{5.2}$$

where c represents the categories of network links (i.e., local, national, and inter-country), $\widetilde{B_i^c}$ represents the median value of known links to/from $Site_i$ for category c. $\widetilde{B_j}$ is the median value of known links to/from SE_j. $link\_category$ and $Site_i$ are two categorical variables, which represent the category and the site name of the target link, respectively. This time, instead of assuming a simple linearity depending on the number of known links among the attributes, we consider two models to fit our desired f: a **linear regression** model, which assumes a linear proportion among the attributes in f and a **neural network** model, which can capture more complex and non-linear relationship among these attributes.

However, categorical variables (i.e., link category and site name) cannot be directly used to tune parameters in either of these models. Hence, we need to first code these two variables as a set of dummy variables, which is a widely used coding technique in order to build a regression model for data sets containing categorical variables [Long *et al.* (2006),Iacobucci (2012)]. The idea of this coding technique is to use $J-1$ numerical values to present a categorical variable with J categories. For instance, as shown in Table 5.2, each link has five attributes, including four numerical variables and one categorical variable (i.e., the category for this link).

	V1	V2	V3	V4	Category
link1	value	value	value	value	local
link2	value	value	value	value	national
link3	value	value	value	value	inter-country

Table 5.2: Example of a data set with 4 numerical and 1 categorical variables.

After transforming the categorical variable into dummy variables, the new data set is shown in Table 5.3. Two extra attributes (i.e., local and national) are generated for links representing their link categories. The third category (i.e., inter-country) can be deduced from these two extra attributes, i.e., (local=0 & national=0) represents the category is inter-country for the link3.

	V1	V2	V3	V4	local	national
link1	value	value	value	value	1	0
link2	value	value	value	value	0	1
link3	value	value	value	value	0	0

Table 5.3: The new data set after transforming categorical variable into dummy variables.

Based on this coding technique, our final data set contains the following 37 attributes for each link. Three median values represents the measured bandwidths of different link categories of the computing site and the median bandwidth of the corresponding SE, two attributes represent the three link categories and 31 attributes present the 32 different computing sites defined in our merged platform. If no measured bandwidth exists for a

given link category of a site, we use the overall median values from all known links in this category to represent the overall connectivity represented in the merged platform.

Finally, for neural network, we consider a simple model of three layers, which is the minimum number of layers allowing to capture non-linear problems: (i) an input layer with 37 neurons which corresponds to the number of attributes for each link; (ii) one single hidden layer; and (iii) an output layer with one neuron. As there is no standard and general rules for choosing the optimal number of neurons in the hidden layer of a neural network model, we use the empirically-derived rules, i.e., the optimal size of the hidden layer is the mean of the number of neurons in the input and output layers. The number of neurons is thus set to 19 in the hidden layer of our neural network model.

5.3.3 Evaluation

In order to understand the prediction accuracy of each model, we compare the bandwidths estimated by each model with the bandwidths obtained in the merged platform. If a model can correctly estimate the existing bandwidths in the merged platform, we then consider it as a good predictive model to estimate the bandwidths of the missing links. Here, our evaluation is based upon the merged bandwidth values in the *Average_Merged* model (i.e., 420 known links) since it has been shown to give the best accuracy to reproduce file transfer durations.

For the empirical model, we can directly estimate the bandwidth of each known link by using Equation 5.1. However, for the linear regression and neural network models, we need to first train each model to find the proper values for parameters. Therefore, we split the 420 known links, each one consisting of the 37 attributes described above, into 85% (357) as training data and 15% (63) as testing data. We randomly split all links 20 times to investigate the prediction ability and the robustness of each model. We use the Mean Absolute Error (MAE) as an evaluation metric to quantitatively assess the overall predictions:

$$MAE = \frac{\sum_{i=1}^{n} |pred\_value - merged\_value|}{n}. \tag{5.3}$$

where pred_value and merged_value are the bandwidth predicted by proposed models and the merged bandwidth for a given known link in the *Average_Merged* platform description, respectively.

Figure 5.4 shows the summary of the MAE for 20 validation tests of three models. We observe that the linear regression model can reduce the maximum prediction error given by the empirical model but the overall MAE is only slightly decreased. The neural network model has the lowest overall prediction errors among the three models, which means that it has the best prediction accuracy.

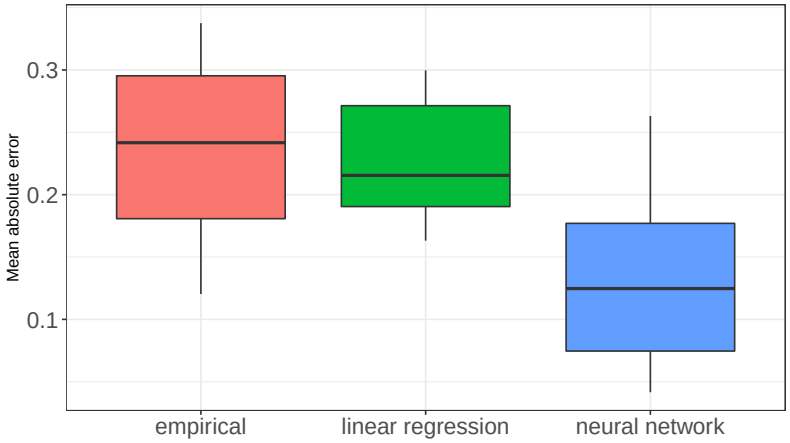

Figure 5.4: Graphical summary of the Mean Absolute Error (MAE) for 20 validation tests of empirical, linear regression, and neural network model.

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
Empirical model	0.12	0.18	0.23	0.24	0.29	0.34
Linear regression	0.16	0.19	0.21	0.22	0.27	0.29
Neural network	0.034	0.074	0.11	0.12	0.16	0.26

Table 5.4: Statistic of MAE for 10 validation tests of neural network and linear regression model.

As summarized in Table 5.4, the average MAE among 20 validation tests for the neural network model is 0.12 and the third quartile value is 0.16, which are significantly smaller than the other two models.

From these observations, we find that a neural network model globally fits f much better than a linear regression model. It means that the relationship among the attributes described in Equation 5.2 cannot be simply expressed by a linear formula. Therefore, the analysis hereafter will focus on the results provided by the neural network model.

Although the neural network model gives us the best overall prediction accuracy, the prediction error still varies a lot depending on the data splitting for training and testing. To understand where this difference comes from , we investigate the two validation tests which correspond to the worst and the best prediction errors of the neural network model.

Figure 5.5 depicts the comparison between the predicted bandwidths and the merged bandwidths of one validation test which gives us the largest error (0.26) for the neural network model. The solid line corresponds to the **ideal case** where the predicted bandwidth value is equal to the merged bandwidth for a given link.

Clearly, we observe in Figure 5.5 that this large prediction error mainly comes from the bad predictions for local links. As each site has only one local link, there is much less information for local links than national or inter-country links in the merged platform. Due

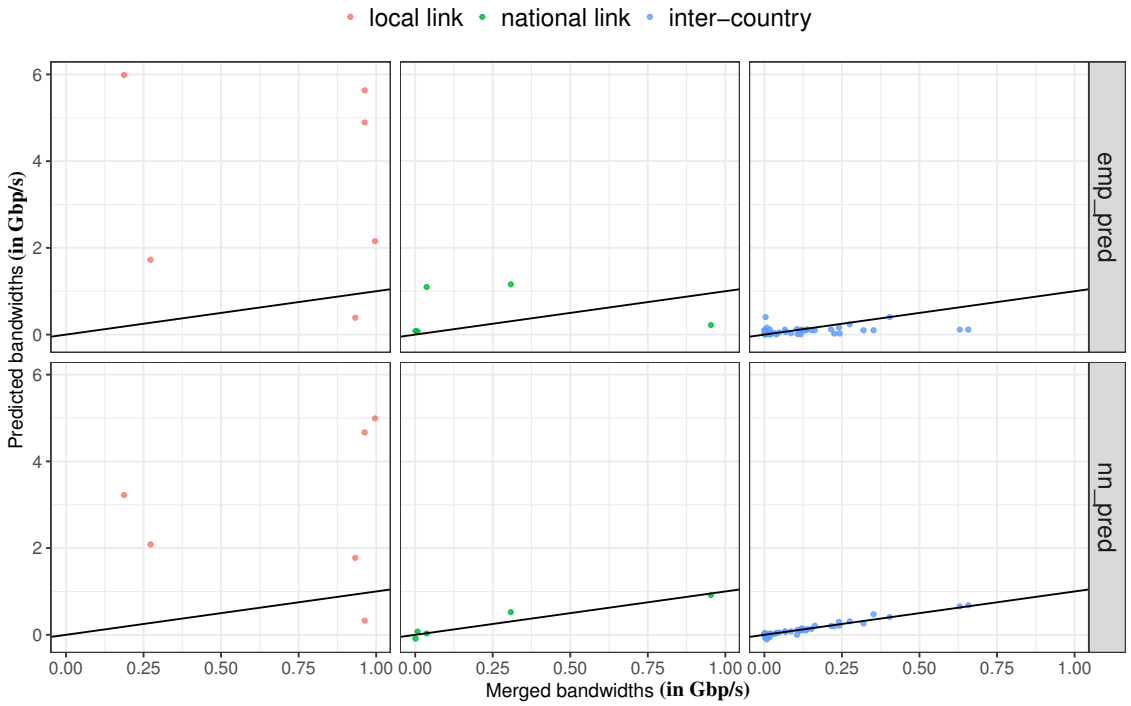

Figure 5.5: The validation test which gives us the largest error (0.26) for the neural network model. The solid line corresponds to the **ideal case** where the predicted bandwidth value is equal to the merged bandwidth for a given link. From top to bottom, panels correspond to the predictions by the empirical model (emp_pred) and the neural network model (nn_pred), respectively.

to this lack of comparable information for local links, almost all the large prediction errors come from local links for both models. It means that none of these models can correctly capture the relationship f for local links because of their unique value for each computing site. For national and inter-country links, the neural network model outperforms the empirical model to better predict the bandwidths for known links. It means that the neural network model can be a good candidate to predict the missing bandwidths of national and inter-country links in the merged platform.

The validation test corresponding to the smallest error (0.034) for the neural network model is shown in Figure 5.6. As we envisaged, this validation test corresponds to the case in which no local link is selected into the testing set and therefore gives the best prediction accuracy for the neural network model.

However, the bandwidths of several links are still predicted to negative values in this "best" validation test (i.e., red points below zero of the national link in the neural network model in Figure 5.6). Based on the analysis of the negative predictions in different tests, we find that this issue is due to the insufficient information for a given site. For instance, two computing sites in France (OBSPM and IN2P3-CC) do not have any measured bandwidths

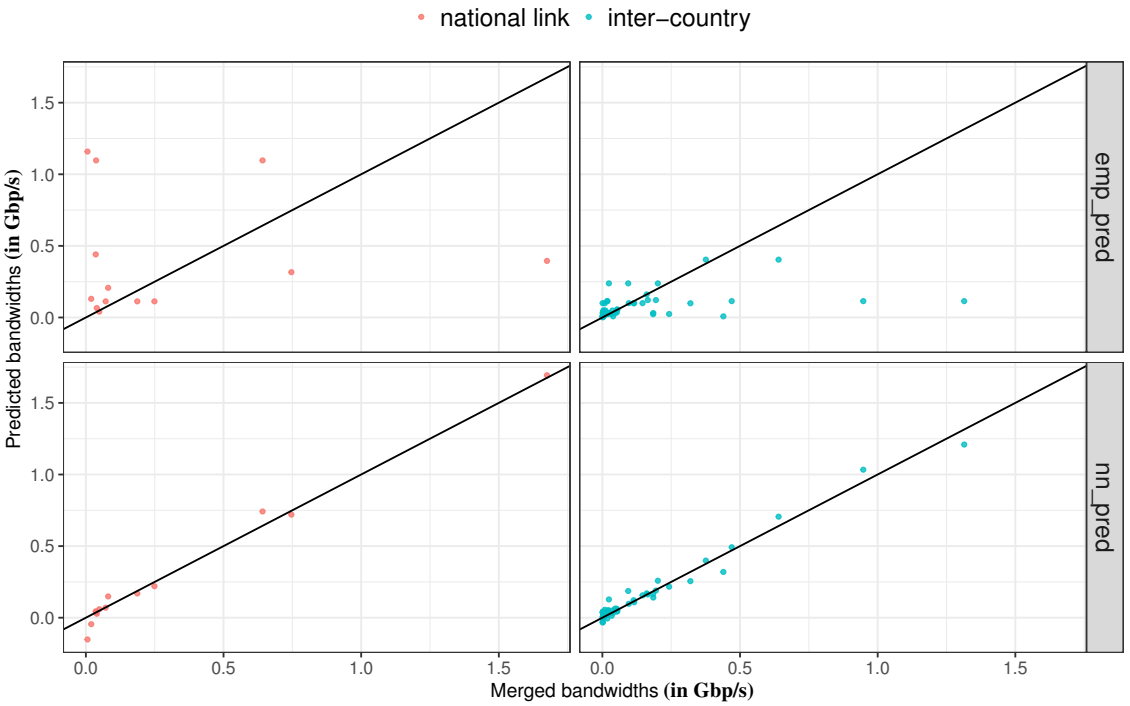

Figure 5.6: The validation test which gives us the smallest error (0.034) for the neural network model. The solid line corresponds to the **ideal case** where the predicted bandwidth value is equal to the merged bandwidth for a given link. From top to bottom, panels correspond to the predictions by the empirical model (emp_pred) and the neural network model (nn_pred), respectively.

for inter-country links. As explained before, input files are replicated onto a limited number of SEs on EGI. For the GATE workflow, whose execution traces are used, its input files are always replicated at least once on a SE in France. According to the replica selection rules described in Chapter 3, the replica in France is favored by these two French computing sites. Hence, only limited information for inter-country links of these two sites can be retrieved, even from multiple traces. Since no available information can be used to train our models, the predicted values for the inter-country links of these two sites are usually very small (close to 0) or even negative in the neural network model. This issue is not considered as a problem related to the training model but to the training data. It can be mitigated by aggregating traces from different workflows to find more available bandwidths for the sites with insufficient information.

Finally, we train the neural network model by all 420 known links to predict the bandwidths of 572 missing links in the *Average_Merged* platform. Among these 572 predictions, 27 are negative values and they all correspond to the case where the computing sites have insufficient information. 25 negative values are from the prediction of inter-country links for the computing site "OBSPM" in France. The remaining 2 negative predictions corre-

spond to national links for site "CIEMAT-LCG2" which has no measured national link and in "HG-08-OKEANOS" having only one measured national link. These negative predictions not only imply that the existing information in current Average_Merged platform is too sparse to train a perfect model for correctly predicting all missing links, but also harm the confidence for other positive predictions.

5.4 Conclusion and discussion

In this chapter, we have presented our method to build a complete platform model for our target infrastructure by leveraging information from multiple execution traces. By spatially aggregating the information from multiple traces, we are able to construct a merged platform with a broader set of hardware resources. We also proposed two methods to exploit the network information for the same links from multiple traces. Simulation results show that the Average_Merged platform can still reproduce the overall distribution of the real transfer durations while the Maximum_Merged platform underestimates most of file transfer durations. Although both models cannot properly capture the extremely long transfers, they still outperform the state-of-the-art model to reproduce the real-life variability of transfer durations.

To fill-in the bandwidths of remaining missing links in the merged platform, we proposed three predictive models based on the existing information. Then we evaluated the prediction accuracy of each model by comparing the predicted bandwidth values with the merged values of known links in the Average_Merged platform.

Machine learning is an interesting approach to construct a more accurate predictive model. The neural network model has the lowest prediction errors compared to the linear regression and empirical models. It can correctly predict most of the national and inter-country links. However, with a small data set (i.e., 420 known links), we can hardly train a perfect model to predict the bandwidths of all missing links in the Average_Merged platform. For several sites with a limited number of known links, the predicted bandwidths are even negative. We believe that this model can be further improved by adding more available information in the training data set.

Therefore, we decide to use the Average_Merged platform filled-in using the empirical model, which allows us to predict bandwidths respecting the overall network hierarchy, to investigate and evaluate different "what-if" simulation scenarios in the next chapter.

Chapter 6

Evaluation of file replication strategies through realistic simulations

Abstract. *In this chapter*[1]*, we propose a practical dynamic replication method to improve data management for applications executed on EGI via VIP. We cross-evaluate the impact of this strategy by simulation using two different platform models: an enhanced state-of-the-art platform description and the complete platform description built in Chapter 5. Simulation results show that the realism of the platform model is key to the evaluation process. They allow us to propose reliable recommendations to reduce the file transfer durations for applications hosted by VIP.*

6.1 Introduction

File replication to multiple storage resources is a common technique to optimize data management in large distributed systems. It reduces file transfer bottlenecks and increases file availability, with a great impact on the application execution time [Lamehamedi *et al.* (2002)]. Numerous file replication strategies were proposed and evaluated using simulations [Elghirani *et al.* (2008),Lei *et al.* (2008),Sato *et al.* (2008),Vrbsky *et al.* (2010),Yang *et al.* (2010),Bsoul *et al.* (2016)], focusing mostly on optimizing file transfer durations (average or total duration by job). However, as pointed out in Chapter 1, the used platform descriptions are often oversimplified, leading to a questionable accuracy of the simulated transfer durations.

[1]Results described in this chapter have been published in [Chai *et al.* (2018)]: Chai, Anchen, et al. "Evaluation through Realistic Simulations of File Replication Strategies for Large Heterogeneous Distributed Systems." Europar 2018-24th International European Conference on Parallel and Distributed Computing; Workshop HeteroPar 2018.

In this chapter, we propose a practical dynamic file replication method to improve the performance of file transfers and to address the issue of imbalanced resource usage identified in Chapter 2. We use two different platform models to evaluate the proposed file replication strategies: a three-level hierarchical model, representing the state-of-the-art platform and the model built from real execution traces in Chapter 5. As we focus on the file management for applications deployed on EGI by VIP and based on different simulation scenarios, we aim at answering the following questions:

- What is the impact of different replication strategies on file transfer durations and resource usages?
- Does the answer to the above question depend on the platform model?
- What would be reliable recommendations for data placement in VIP?

The rest of the chapter is organized as follows. In Section 6.2, we present the proposed dynamic file replication method. Different components of simulation scenarios are detailed in Section 6.3. Section 6.4 presents the evaluation and the analysis of the simulation results. Recommendations for the applications deployed on EGI via VIP are given in Section 6.5.

6.2 Replication strategies

6.2.1 Dynamic replication strategy

As presented in Chapter 1, the current replica creation strategy implemented in VIP is a static method, which chooses 3 to 5 stable SEs with sufficiently large amounts of available storage space. Dynamic replica creation is not adopted by VIP for the applications in production.

Given the large scale of distributed systems such as EGI, allowing thousands of independent jobs to be executed in parallel, we propose a dynamic replication strategy to further improve file placement during the execution of an application in addition to the current replica management strategy adopted by VIP (i.e., static replica creation and replica selection according to geographical distance).

Our idea is inspired by the "cache hit" mechanism. The first job executed in a computing site downloads the file, then copies and registers it onto the local SE associated to this site. Then, the subsequent jobs scheduled in the same site can directly benefit of a local file transfer, hence optimizing the overall file transfer duration. This strategy is motivated by two observations extracted from execution traces on EGI in Chapter 2. First, when the workflow of an application consists of a large number of jobs, a given site will execute more than one job in general. Second, the intra-site delay between the first job and subsequent jobs in one computing site is highly variable. It means that if subsequent jobs have a

much longer intra-site delay compared to the duration of file uploading and registration by the first job, they can directly benefit of the local transfer without any extra delay. This method is implemented as a dynamic replica creation service in our simulator presented in Chapter 3. The detail is described in Algorithm 3.

Algorithm 3: Dynamic replication method

```
   Input: S: local SE, F: target file, T: Timeout, H: Threshold
 1 Patient_Time = 0, Nb_retry = 3, Retry_freq = 2s;
 2 bool Lock = False, status = False;
 3 bool F_exist_in_S = Check_F_exist_in_S();
 4 if F_exist_in_S then
 5 |   download(F, S); # download file from local SE
 6 else
 7 |   if Lock then
 8 |   |   while (Patient_Time < H) do
 9 |   |   |   sleep(Retry_freq) ;
10 |   |   |   F_exist_in_S = Check_F_exist_in_S() ;
11 |   |   |   Patient_Time+ = Retry_freq ;
12 |   |   end
13 |   |   if !F_exist_in_S then
14 |   |   |   download(F, lcg-util); # download file using replica selection service on EGI
15 |   |   else
16 |   |   |   download(F, S);
17 |   |   end
18 |   else
19 |   |   Lock = True;
20 |   |   Locked: # Begin of critical section
21 |   |   i = 0;
22 |   |   Sorted_list = lfc_fillsurls(); # sort replicas using the same rule as in the middleware
23 |   |   while (i < Nb_retry and status == Fail) do
24 |   |   |   # download F from Sorted_list[i] with timeout T
25 |   |   |   status = download(F, Sorted_list[i], T);
26 |   |   |   i ++ ;
27 |   |   end
28 |   |   if status == Success then
29 |   |   |   upload(F, S); # upload F onto S
30 |   |   else
31 |   |   |   download(F, lcg-util);
32 |   |   end
33 |   |   Unlock # End of critical section
34 |   end
35 end
```

In the implementation of this dynamic replication method, two parameters need to be configured. To reduce the impact of extremely long transfers [Glatard *et al.* (2007)], a timeout (i.e., T) is imposed for the first job in a site to download file from remote SE (line

25 in Algorithm 3). The second parameter is a threshold (i.e., H) on the maximum waiting time for subsequent jobs in a site (line 8 in Algorithm 3). It determines the maximum patient time of subsequent jobs to wait for the availability of required file in local SE. We choose to configure this parameter to the same value as the timeout for the first job in a site. It means that subsequent jobs will lose their patience if the first attempt of transfer is failed by the first job in their site.

The value of timeout and threshold is currently set to 110 seconds which corresponds to the third quartile of distant transfer (i.e., national or inter-country) durations of 121MB release file in collected execution traces. If the timeout expires, the transfer is canceled and a new attempt is made with another SE (line 23 to 27 in Algorithm 3). The maximum number of retries is currently configured to 3. If the transfer still fails after three retries, a fourth file transfer will be launched without timeout (line 28 to 32 in Algorithm 3)

6.3 Simulation studies

The long-term objective of this study is to optimize data placement for science gateways such as VIP using large scale distributed heterogeneous infrastructures such as EGI. To this end, we propose to evaluate different simulation scenarios fed with realistic information coming from execution traces using the simulator developed in Chapter 3. Hereafter we detail the different components of our designed simulations.

6.3.1 Platform Models

We consider two platform models. First, the **trace-based** complete platform description generated in Chapter 5. We choose the *Average_merged* model since it has been shown to give the best accuracy when simulating file transfers.

While this traced-based model is accurate, it is also complex to build. Therefore, we also consider a simpler model inspired from the state-of-the-art model: **a three-level hierarchical network model**. To better reflect the connectivity of the production system, we enhance it by using average bandwidth values derived from traces instead of the theoretical values proposed in the literature. We use 1.3 Gb/s for local links, 255Mb/s for national links, and 100 Mb/s for inter-country links. If simulation results are consistent between the two models, then the building simplicity of this three-level hierarchical platform makes it a good candidate for further studies.

6.3.2 Simulation scenarios

We evaluate two file replication strategies: file prestaging and the proposed dynamic replication strategy, which also includes file prestaging. In the file prestaging strategy, files are copied on three preselected SEs before the execution of the application. This

corresponds to the current replication strategy used by VIP. We will evaluate the impact of different prestaging lists on the performance of file transfers, with or without *a priori* information on the sites where jobs are executed.

We simulate the execution of 15 GATE workflows, each consisting of 100 jobs and downloading 121MB release file, to study the performance of file transfers with different replication strategies. Realistic information are extracted from execution traces and injected as parameters in our simulator (i.e., the queuing time and the execution site of jobs). The simulated replica selection service on EGI is used to determine the download source of files for each job during the simulation.

To determine the impact of SE selection for each platform model, we study three categories of prestaging lists:

- the current production setting, which corresponds to three SEs located in France;
- 50 randomly selected lists;
- four prestaging lists selected based on statistical information on the sites where the jobs of the 15 workflows were executed.

These four lists contain the local SEs of the three sites hosting the largest number of jobs located in one or different countries or three sites hosting no jobs at all located in one or different countries, respectively. We always fix the number of SEs used to prestage files to three to match the number of replicas currently used in production. The impact of the number of SEs is let out of the scope of the current study.

In total, we simulate 220 scenarios (2 strategies $\times$ 2 platform models $\times$ 55 prestaging lists) for each of the 15 workflows. They generate 330,000 (220 scenarios $\times$ 15 workflows $\times$ 100 jobs) simulated release file transfers which will be evaluated.

6.4 Performance evaluation

6.4.1 Impact of dynamic replication

We begin our evaluation by studying the cumulative distribution of the simulated durations of file transfers with and without dynamic replication. Each line in Fig. 6.1 corresponds to one list of 3 SEs used for file prestaging, using either the 3-level (top) or the trace-based (bottom) platform model. The same 50 random prestaging lists are used in all four scenarios.

For the 3-level model, we see that dynamic replication significantly decreases file transfer durations, as more jobs can download files from a local SE. Moreover, the performance does not depend on the SEs used for prestaging with a median duration of 5.1s and a maximum value of 32.3s. Without dynamic replication, the choice of the prestaging list

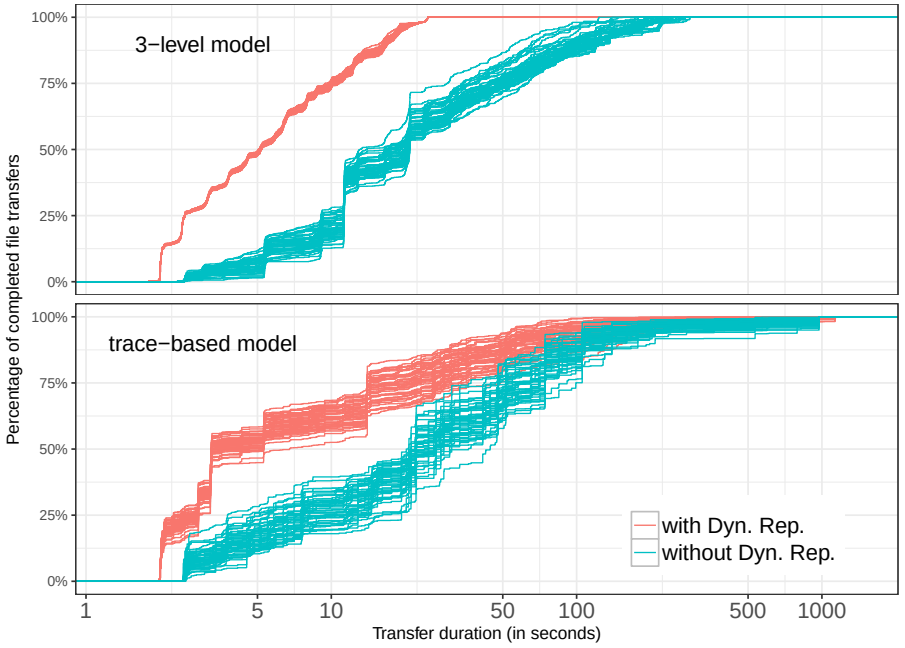

Figure 6.1: Cumulative distribution of simulated file transfer durations with and without dynamic replication. Each line corresponds to a list of 3 SEs used for file pre-staging. The same 50 random prestaging lists are used in all four scenarios.

has a stronger impact, leading to longer and more variable transfer durations. The median varies from 13s to 21s when utilizing different lists while the maximum varies from 123s to 290s.

For the trace-based model, we also see a reduction of file transfer durations when using dynamic replication, but the gap is less clear. Contrary to the 3-level model, the performance with dynamic replication varies more significantly depending on the prestaging list. As for the 3-level model, the choice of the prestaging list always has a strong impact on performance when there is no dynamic replication. The median duration varies from 20s to 44s while the longest duration is always 975s even with different prestaging lists. It is due to a special site, which has a poor connectivity (i.e., 1Mb/s) to most SEs, in the trace-based model. Therefore, the longest simulated transfer is always given by jobs in this site.

Besides the impact on the durations of file transfers, we are also interested in the impact on the resource usages brought by dynamic replication. Figure 6.2 compares the cumulative number of downloads for a SE and the number of jobs executed in the computing site associating this SE as the local one with and without dynamic replication. We study the resource usages during the executions of the 15 workflows with the 50 random prestaging lists. Here, we present the comparison from the simulation results with the trace-based model. The 3-level model gives us the similar comparison.

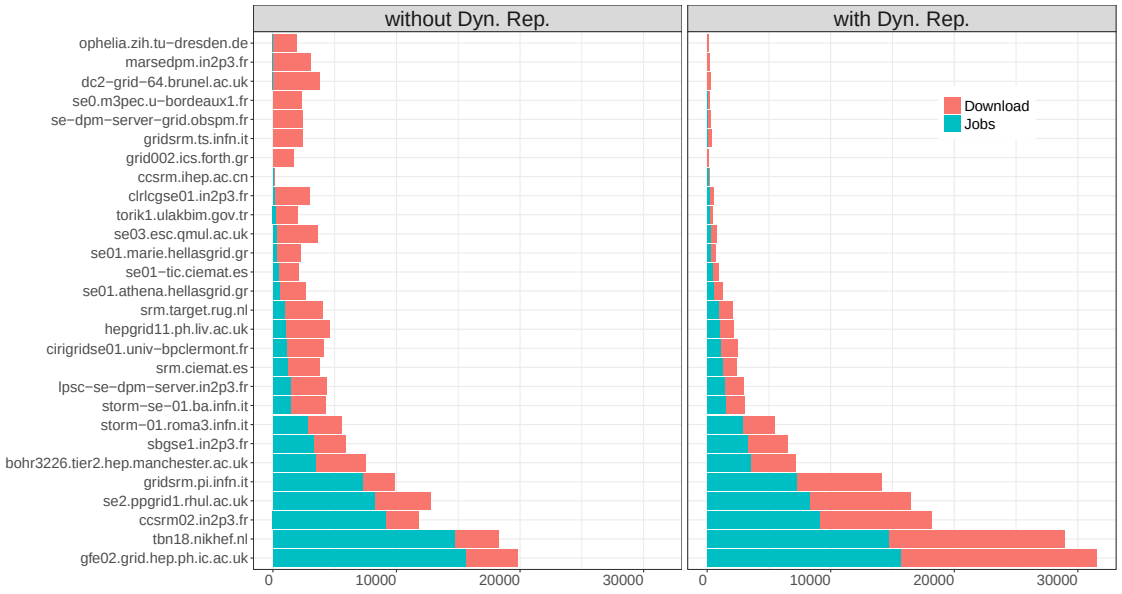

Figure 6.2: Cumulative number of downloads for SEs versus the number of jobs in the corresponding sites with and without dynamic replication.

We observe that without dynamic replication, the usage between the storage resources and the compute resources for a site is very imbalanced. It is consistent with the analysis of the execution traces presented in Chapter 2 since the scenarios without dynamic replication actually corresponds to the current replication management in VIP, i.e., three SEs to replicate files before the application execution. Without *a priori* information on where jobs will be executed, this issue can hardly be solved by selecting different SEs for file prestaging. By adopting dynamic replication, we can obtain a better balance between the number of downloads and the number of jobs for each site. Moreover, with dynamic replication, the number of downloads from computing sites with no jobs are much less than without dynamic replication, decreasing from 8,690 to 567.

Overall, as shown in Table 6.1, without dynamic replication, only 11% (8,452) of simulated transfers are local while 89% are distant transfers (i.e., intra-country and inter-country). With dynamic replication, the ratio of local transfer increases to 91% (68,395) due to the "cache-hit" mechanism. The remaining 9% are distant transfers: 6.6% (4,918) are from the first job executed in a computing site and 2.3% (1,687) are subsequent jobs which have reached the maximum threshold value before the completion of file registration by the first job. This comparison shows that our proposed dynamic replication can effectively address the issue of imbalanced usage between the storage resources and the compute resources for a site and also the inefficient network usages, which were identified from the analysis of execution traces in Chapter 2.

	local transfer	intra-country	inter-country
without Dynamic Replication	8,452	26,078	40,470
with Dynamic Replication	68,395	3,152	3,453

Table 6.1: Cumulative number of different type of transfers with and without dynamic replication for 15 simulated workflows with 50 random prestaging lists.

6.4.2 Impact of different prestaging lists on static replication

We saw that, globally, the choice of SEs used for prestaging mainly matters when there is no dynamic replication. To measure the impact of SE choice on file prestaging, we compare the 50 random prestaging lists, the 4 predefined lists and the current prestaging list used in production.

We identify the best and the worst prestaging among these 55 lists based on the median simulated file transfers duration. The performance corresponding to the current production prestaging list is also identified (named "prod prestaging"). It utilizes 3 SEs in France, chosen according to the criteria described in Chapter 1. Note that we only evaluate the impact of the prestaging list w.r.t. the file transfer duration. Other aspects taken into account by VIP administrators (e.g., reliability, availability and storage space of each SE) are left as future work. The comparisons for the 3-level hierarchical (top) and trace based-model (bottom) are shown in Fig. 6.3.

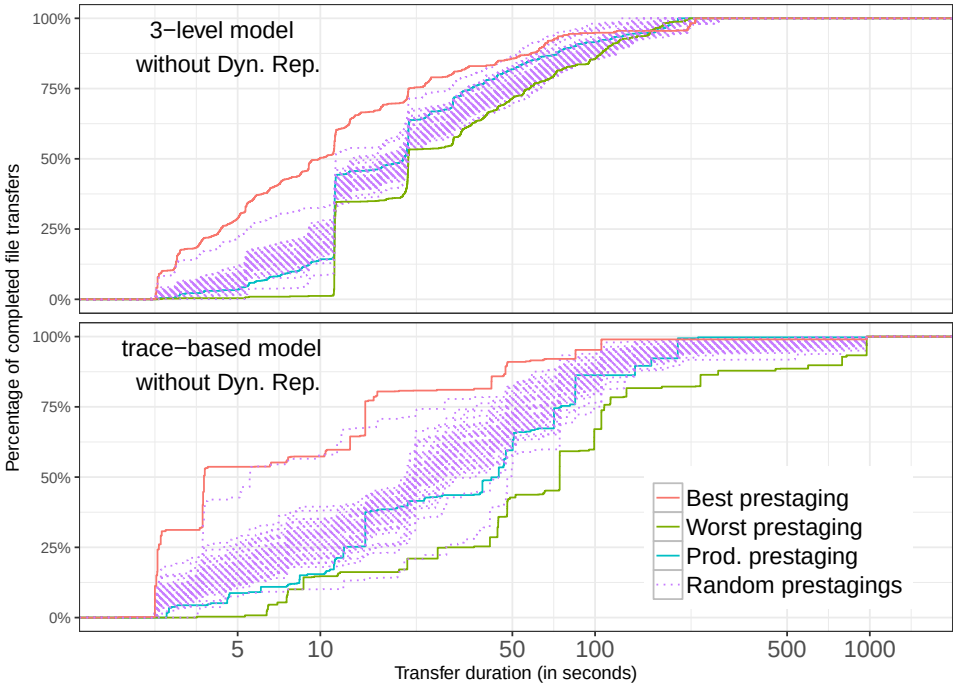

Figure 6.3: Comparison of random, predefined, and the current production prestaging list without dynamic replication for two platform models

For the 3-level model, the "best prestaging" corresponds to one of the predefined lists: three SEs associated to the sites hosting the largest number of jobs located in three different countries, i.e., UK, Netherlands, and France. By selecting the most used sites, most of the jobs can directly download files from their local SEs. Moreover, scattering file replicas in different countries can efficiently reduce the number of downloads from a foreign country. Conversely, the "worst prestaging" for the 3-level model is given by three SEs associated to sites that do not execute any job and are located in different countries. Thus, most of the jobs download files from a foreign country, which leads to the worst performance.

For the trace-based model, we find the exact same "best prestaging" and "worst prestaging" as for the 3-level model. It further validates the findings from the 3-level model. In Chapter 2, we saw that the distribution of jobs was very heterogeneous at the country level. Moreover, from both the information in collected traces and the historical information from the DIRAC [Tsaregorodtsev *et al.* (2010)] server, we found that UK, Netherlands, and France were the countries hosting the largest number of executed jobs (∼80%) in the Biomed VO. Prestaging files at these three countries allows to maximumly reduce the number of downloads from a SE located in foreign countries. Therefore, we believe that the best performance without dynamic replication is likely to be obtained by selecting the SEs in different countries hosting the largest cumulative number of executed jobs for both models.

It is also interesting to note in Fig. 6.3 that the performance of the prestaging currently used in production is quite different between the 3-level and trace-based models. In the former, SEs are equivalent in the sense that a single bandwidth value is used for all the links in each category (i.e., local, inter-country, and intra-country). Performance will then be better for lists with SEs close to the sites executing most of the jobs. In the latter, each link is unique and the use of close SEs alone cannot ensure the best performance. The "prod prestaging" list illustrates this. It corresponds to three SEs in France, close to sites that execute more than 16% (which is more than the average sites) of the total number of jobs. However, the general connectivity for these three SEs is worse than the average. This explains why the performance of the "prod prestaging" list is better than most of the randomly selected prestaging lists in the 3-level model and worse in the trace-based model. It also shows that different platform models can lead to different qualitative assessments for scenarios even with similar configurations.

6.4.3 Impact of platform model on replication decisions

Figure 6.4 compares the duration of file transfers when using dynamic replication for both models. We observe that dynamic replication leads to much more stable results in the 3-level model than in the trace-based model.

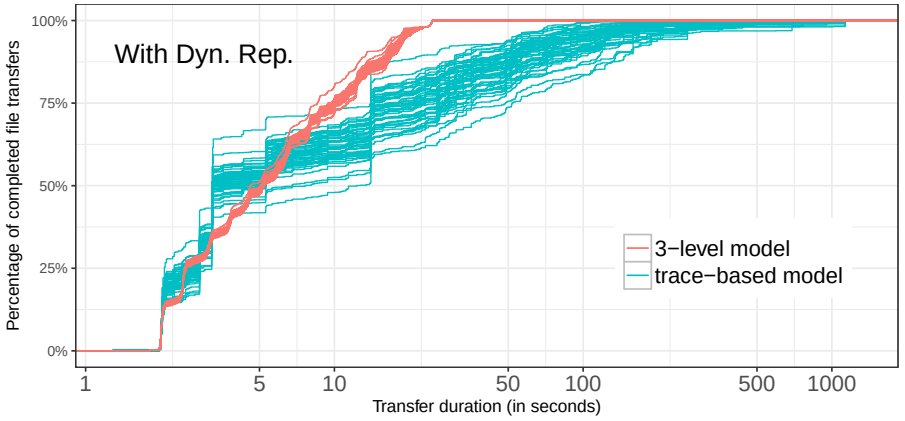

Figure 6.4: Cumulative distribution of simulated file transfer durations with dynamic replication for two platform models

In other words, in the 3-level model, a random selection of SEs to prestage files is enough. No improved SE selection strategy is required. However, for the trace-based model, we observe a greater variability which can be explained by the important heterogeneity in terms of network connectivity that is better captured by this model. With dynamic replication, the file transfer duration of the first job in a site will directly decide how many subsequent jobs can benefit from the local transfer. A very long duration may let more subsequent jobs reach the maximum patient time. Depending on where files are prestaged, the transfer duration of the first job in a site can be highly variable in the trace-based model. Therefore, even with dynamic replication, the performances still has a large variability in the trace-based model. In the 3-level model, as we only have one bandwidth value for each link category, the choice of the SEs in the prestaging list only have limited impact on the transfer duration of the first job in a site. The performances are hence much more stable.

Figure 6.5 compares the best performance achieved by predefined or randomly selected lists without dynamic replication for each model. As in simulation we have the complete *a priori* information (e.g., the distribution of executed jobs on computing sites or in countries) about the sites on which jobs are going to be executed, the best predefined prestaging list is always better than the best random list. Interestingly, we see that the gain is much larger in trace-based model. The more heterogeneous the platform is, the more important *a priori* information is to optimize file transfers.

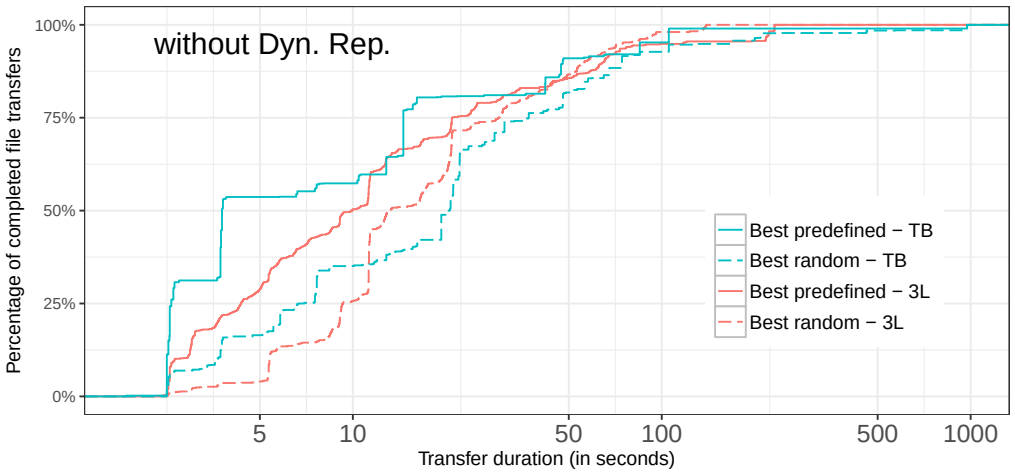

Figure 6.5: Cumulative distribution of simulated file transfer durations without dynamic replication for two platform models. Best performance achieved by predefined or randomly selected lists is highlighted.

6.5 Recommendations for file replication in VIP on EGI

As we have seen above, simulation results are not always consistent between the two models. We find that a larger variability exists in the trace-based model even with dynamic replication. The relative performance of the current production configuration also differs from a model to another. Consequently, recommendations for VIP need to be based on the results obtained with the trace-based model.

Figure 6.6 compares the best and worst performance (with or without dynamic replication) to the current production setting.

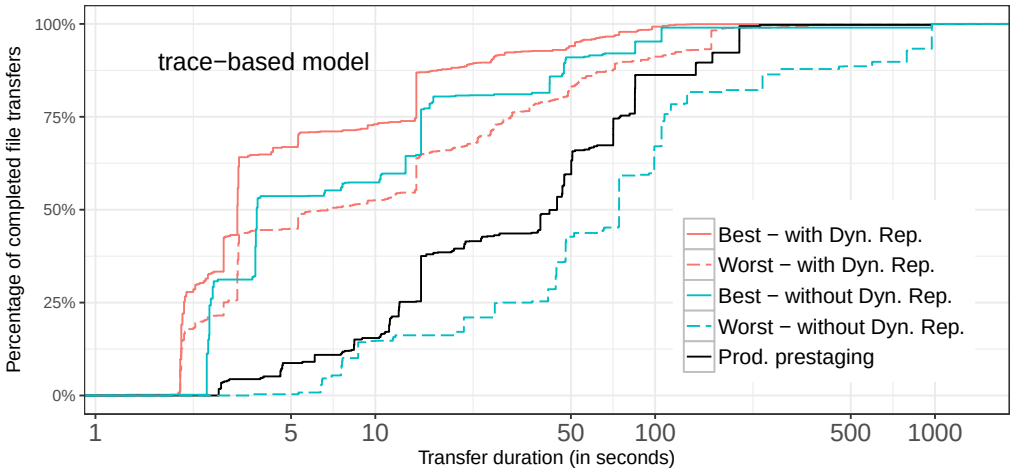

Figure 6.6: Comparison of the best and the worst prestaging with the current production prestaging for trace-based model with or without dynamic replication.

Without dynamic replication, a careful selection of the SEs used for file prestaging reduces file transfer times. Based on statistical information from the DIRAC server, we can decide which countries hosted largest cumulative number of jobs in a long-term point of view. However, it still requires *a priori* information on where jobs are going to be executed to reach the best performance without dynamic replication. For jobs submitted independently in large distributed systems such as EGI, we cannot know in advance where they will be executed. However, we could attempt to predict it by leveraging historical data on where the jobs have been running over a given period of time.

Dynamic replication always outperforms the current production configuration. To better quantify its gain, we computed in Table 6.2 the 95%-confidence interval for the statistics on the simulated transfer durations over the 55 studied prestaging lists. We conclude that with dynamic replication, there is a 95% chance that 75% of file transfers will be 2.5 times shorther than without, regardless of the selected prestaging list.

	1st Qu.	Median	Mean	3rd Qu.	Max
without Dyn. Rep.	[8.3;11.2]	[22.9;28.23]	[60.5;71.2]	[57.8;66.9]	[974.4;974.8]
with Dyn. Rep.	[2.6;2.7]	[3.5;4.3]	[25.6;30.7]	[19.8;24.7]	[1192.1;1301.4]

Table 6.2: 95%-confidence interval for the statistics of the simulated release transfers durations of 55 prestaging lists with and without dynamic replication

However, the longest transfer duration seems to be worse with dynamic replication, which is due to the timeout mechanism. As we have identified, the longest simulated transfer corresponds to a job executed on a site with poor connectivity to/from most SEs in the trace-based model. When using the dynamic replication, the timeout expires 3 times, hence adding an overhead of three times the timeout value (i.e., 110s). The total transfer time corresponds to the cumulative time of all transfer attempts (failed and successful). This effect could be mitigated with a timeout value that makes a trade-off between the longest acceptable transfer duration and this extra overhead caused by retries. It is important to note that such an extreme case cannot be evaluated with the 3-level model that does not reflect the heterogeneity of the actual infrastructure.

To summarize, we can conclude from our observations that dynamic replication can globally reduce the duration of file transfers except for extreme cases where multiple transfer timeouts are hit successively. Such cases are only captured by the trace-based platform model. As the benefits of dynamic replication come from the number of jobs that transfer files from a local SE thanks to the copy made by the first job, it may not be interesting for small workflows. Finally, implementing such a dynamic replication strategy in the production environment would require non-negligible development effort to ensure the correct handling of concurrent file access synchronization.

6.6 Conclusion

In this chapter, we presented an evaluation of file replication strategies by studying two platform models: a 3-level hierarchical model and a model built out of execution traces. The originality of our work consists in evaluating both the impact of the platform models and the impact of the replication strategies thanks to realistic simulations.

Results show that the estimated impact of a strategy can be quite different when the platform model changes. In other words, the conclusion drawn from one model cannot be automatically transferred to another. We show that the instantiation of the two models leads to different qualitative decisions, even though they reflect a similar hierarchical topology. It emphasizes the fact that the realism of the platform model is key to the evaluation process.

By comparing the results obtained with each model, we found that selecting the sites hosting a large number of executed jobs to prestage files is a safe recommendation to optimize data management in the production system. In addition, adopting dynamic replication can further reduce the duration of file transfers except for extreme cases (very poorly connected sites) that our realistic simulations were able to capture.

Chapter 7

Conclusions and perspectives

Simulation has often been considered as "literally costless" for research in distributed systems. However, the realism of simulation has rarely been investigated in the literature, as it requires serious scientific efforts which can be very costly.

In this thesis, we addressed the challenge of conducting realistic simulations of file transfers for applications executed in a large distributed production system. The realism of simulations has been investigated in two main aspects: the simulator and the platform model. To have an inside view of our target infrastructure EGI, we collected and analyzed a set of execution traces of the GATE application hosted by science gateway VIP. Based on the knowledge derived from traces, we designed and implemented a simulator allowing us to accurately simulate file transfers during GATE executions. Then we built realistic and fully connected platforms in order to replay the execution of GATE workflows and to evaluate different "what-if" scenarios for improving file management in science gateways such as VIP. Finally, we cross-evaluated different file replication strategies by simulations using two platform models and proposed reliable recommendations for applications deployed and executed on EGI via VIP.

In the remainder of this chapter, we first summarize the contributions of each chapter and then present particular perspectives identified along the development of this thesis. Some perspectives present the limitations of our work in this thesis, and some others open new research directions for future work.

7.1 Contributions

Chapter 2: Analysis of the execution traces of the GATE workflow. Based on the analysis of the file transfer durations, we found that the minimum latency of file transfers on EGI was around 1 second. It allows us to instantiate the cost of communications among simulated components (i.e., SE, LFC, etc.) in our simulator. We also confirmed that network hierarchy did exist in EGI, but was much more heterogeneous than a simple

three-level model. The characteristics of workflow executions (e.g., intra-site delay, imbalanced distribution of jobs, etc.) and the identified issues (e.g., imbalanced usage between compute and storage resources of a site) allowed us to propose potential improvements for file management in VIP.

Chapter 3: An ad-hoc realistic simulator of the GATE workflow. We implemented a simulator based upon the SimGrid toolkit to simulate file transfers during the executions of the GATE workflow. To improve the accuracy of the simulated durations of file transfers, we

- abstracted a file transfer into 4 messages among different components;
- instantiated the cost of control messages based on the transfer latency derived from execution traces;
- implemented the same algorithm of replica selection service as in the EGI middleware;
- injected the exact values of important parameters related to workflow executions.

This simulator was then used to conduct simulations in Chapters 4, 5, and 6.

Chapter 4: Replaying the execution of GATE workflow. We built realistic ad-hoc platform models to replay the executions of the GATE workflow with a focus on file transfers. The accuracy of our proposed network topology and bandwidth instantiation methods have been evaluated by comparing the simulation results with the ground truth of real transfers registered in the traces. Simulation results showed that our proposed models largely outperformed the widely used state-of-the-art model to capture the real-life variability of file transfers on EGI. This work was presented at the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid) conference [Chai *et al.* (2017)].

Chapter 5: Towards a complete description of the Biomed VO. In this chapter, we aggregate multiple traces to construct a more general and complete description of the Biomed VO platform based on the ad-hoc models built in Chapter 4. We also propose three predictive models to fill-in the bandwidth of missing links. The accuracy of models is evaluated by comparing the estimated bandwidths with the merged bandwidths for all known links. A complete description is generated with one predictive model, which is used as a realistic platform model of EGI to evaluate file replication strategies in [Chai *et al.* (2018)].

Chapter 6: Evaluating file replication strategies by realistic simulations. We cross-evaluate different file replication strategies using an enhanced three-level network hierarchical model and the trace-based platform description built in Chapter 5. Simulation results have shown that the estimated impact of a strategy can be quite different while

the two platform models reflect a similar hierarchical network topology. It emphasizes the importance of our work to build a realistic platform model of EGI in this thesis. By comparing the results obtained with each model, we found that selecting the sites hosting a large number of executed jobs to prestage files is a safe recommendation for VIP. In addition, adopting dynamic replication can further reduce the duration of file transfers except for extreme cases (very poorly connected sites). This work was presented at the HeteroPar 2018 workshop of the 24th International European Conference on Parallel and Distributed Computing (Euro-Par) conference [Chai *et al.* (2018)], and was awarded the "Best workshop paper on heterogeneous systems".

7.2 Perspectives

7.2.1 Improving the realism of file transfer simulations

To provide a more realistic simulation environment to simulate file transfers on large distributed systems, additional aspects need to be considered in our simulator proposed in **Chapter 3**. When taking into account each new aspect in our simulator, the difficulty remains to validate its simulation behavior against the real production system. Hereafter, we list some aspects that could be investigated in future work.

Transfer failures are quite common in large production distributed systems which are prone to errors. They may have a strong impact on the completion time of a file transfer in real executions of applications, as observed in Chapter 2. The SimGrid toolkit offers the "ON/OFF" attribute for simulated SEs, which allows us to define the availability of SEs along time to simulate transfer failures.

I/O latency is also an important aspect deciding the duration of file transfers in distributed systems, especially for transferring large files which may be split and stored on different disks or tapes. Storage APIs have been recently added in SimGrid toolkit. Models for different disks have been proposed and studied in [Lebre *et al.* (2015)]. These APIs will allow us to simulate the I/O latency of a file transfer on EGI.

Maximum number of concurrent transfers supported by each SE. This configurable parameter is not always the same in real systems, depending on the protocal used by SEs (e.g., dCache or DPM) and their local configurations on EGI. With the current design and implementation of our simulator, we can easily adjust this parameter by calibrating the maximum number of simultaneous requests that each simulated SE can process. However, it would require to collect more information about the general configuration of SEs.

7.2.2 Improving the realism of the platform description of EGI

Concurrency estimation for network links. In **Chapter 4**, a method was proposed to correct the estimation of the *nominal capacity* of network links, which depends on the starting points of transfers to estimate the number of concurrent transfers. However, only one single value is registered for the duration of file transfers in traces. This single value does not allow us to distinguish the real network transfer duration from other latencies. Therefore, the number of concurrent transfers could be mis-estimated with respect to the actual network usage, which can lead to a wrong estimation of the nominal capacity of bandwidths. In future work, we can effectively improve this method by leveraging more detailed information on transfer durations in traces or imposing more constraints in our definition of concurrent transfers to minimize the impact of inaccurate estimation of starting time-stamp. For example, we could define "concurrent transfers" as transfers with similar starting time-stamp or at least 80% of their durations are overlapped.

Temporal variability for bandwidths of network links. Building a fully connected platform description was mandatory to evaluate different file replication scenarios. In **Chapter 5**, we evaluated two methods to instantiate the bandwidths of network links with information from multiple traces. Simulation results have shown that the average-merged platform still reproduces the overall distribution of real transfer durations. However, this average-based approach is only valid to evaluate these specific workflows. In order to reach a more general description beyond *average* bandwidths, we need to take into account temporal information of bandwidth variability for network links. It consists in modeling either the distribution of the available bandwidth of links or the network traffic on EGI. However, it is challenging to find the best compromise between the realism and the level of abstraction. For instance, the most realistic way is to model the network traffic of each network link and then define the bandwidth availability along time in an "availability file" which is one potential attribute for links in SimGrid. This solution would offer us more realism to reflect the network traffic on EGI. However, it requires to study the bandwidth distribution and then produce an "availability file" for each network link. Another solution is to model the network traffic in a more coarse-grained level, for instance, by link categories (i.e., local, national, inter-country). In this case, we only need to generate three "availability files" no matter how many links will be finally defined in our platform. In [Glatard and Evans (2015)], authors proposed a site classification algorithm to address the numerical variability among computing sites in large distributed production systems. A similar approach could also be applied to investigate bandwidth variability of links in EGI and it will help us to decide the best abstraction level for modeling the bandwidth distribution of links. The idea is to use a classification method to group together network links with similar variability so that we can model their bandwidth distributions

and generate the "availability file" for each group.

Fill-in missing links. In **Chapter 5**, we used machine learning techniques to train models to capture the relationship between the bandwidth of network links and attributes extracted from the computing sites and SEs. Our cross validation has shown that the neural network approach can be used as a promising method to predict the bandwidths of national and inter-national links by leveraging known information in the merged platform. However, with currently a small set of known links (i.e., 420) in the merged platform, we cannot perfectly train a neural network model to correctly predict the bandwidths of all missing links. In future work, we would further improve the neural network model by aggregating more execution traces from different applications to obtain more measured bandwidths for the sites on EGI. Besides, additional efforts are also needed to find the optimal values for classical hyper-parameters (e.g., learning rate, number of neural in hidden layer, etc.) in a neural network model.

Another direction to find the information of bandwidth for missing links is to use network monitoring systems such as the perfSONAR [Campana *et al.* (2014)]. One long-term perfSONAR archive[1] allows us to retrieve numerous information about the network (e.g., the bandwidth, delay, packet loss, and packet traces) between perfSONAR hosts installed at different sites on EGI. However, several drawbacks have prevented us to use perfSONAR in this thesis:

- Not all computing sites in the Biomed VO of EGI currently support perfSONAR. We cannot find all necessary information for our platform.
- Available information in the perfSONAR archive are site-based. No detailed information is available for local links if the local SE belongs to the computing site.

After a large deployment of perfSONAR hosts on the sites in the Biomed VO, such a tool would be helpful to investigate the distribution of links and to model the network traffic in future work.

7.2.3 Extending the current capacities of our simulator

Simulation results in **Chapter 6** have shown that *a priori* information of the distribution of jobs is very important to improve the performance of file transfers in heterogeneous platforms. A bad decision for the prestaging list will lead to a worse performance than with the current production setting. Although our proposed dynamic replication method has been shown to be able to improve the performance with no need to have the *a priori* information, implementing such a dynamic strategy in the production environment would require a non-negligible development effort for the correct handling of concurrent

[1] psds.grid.iu.edu/esmond/perfsonar/archive

file access synchronization and human maintenance. One alternative is to do **dynamic planning** for applications, which is similar to the *site selection* process in Pegasus [Deelman *et al.* (2015)]. The idea is to explicitly select different computing sites for workflows before their executions. It allows us to obtain the complete information about where jobs will be executed and therefore to decide the best prestaging list. The replica creation problem is therefore transformed into estimating where will be the available computing resources. Dynamic planning will be beneficial if the latency waiting for available resources is much smaller than file transfers from remote SEs. The evaluation of such dynamic planning mechanism requires to extend our current simulator by correctly simulating more middleware services of EGI, e.g., resource allocation and job scheduling.

Finally, with more parameters and services taken into account in our simulator and a more complete platform description of EGI, we would offer a general and realistic tool for researchers and developers to test different optimization or deployment strategies for applications executed on large distributed production systems, which is our long-term aim for future work.

Bibliography

[Aad *et al.* (2008)] Aad, G., Butterworth, J., Thion, J., Bratzler, U., Ratoff, P., Nickerson, R., Seixas, J., Grabowska-Bold, I., Meisel, F., Lokwitz, S., *et al.* (2008). The atlas experiment at the cern large hadron collider. *Jinst*, 3:S08003.

[Abdullah *et al.* (2008)] Abdullah, A., Othman, M., Ibrahim, H., Sulaiman, M. N., and Othman, A. T. (2008). Decentralized replication strategies for p2p based scientific data grid. In *Information Technology, 2008. ITSim 2008. International Symposium on*, volume 3, pages 1–8. IEEE.

[Alsoghayer and Djemame (2014)] Alsoghayer, R. and Djemame, K. (2014). Resource failures risk assessment modelling in distributed environments. *Journal of Systems and Software*, 88:42–53.

[Amoon (2013)] Amoon, M. (2013). A job checkpointing system for computational grids. *Open Computer Science*, 3(1):17–26.

[Andronikou *et al.* (2012)] Andronikou, V., Mamouras, K., Tserpes, K., Kyriazis, D., and Varvarigou, T. (2012). Dynamic qos-aware data replication in grid environments based on data "importance". *Future Generation Computer Systems*, 28(3):544–553.

[Averitt *et al.* (2007)] Averitt, S., Bugaev, M., Peeler, A., Shaffer, H., Sills, E., Stein, S., Thompson, J., and Vouk, M. (2007). Virtual computing laboratory (vcl). In *Proceedings of the International Conference on the Virtual Computing Initiative*, pages 1–6.

[Avetisyan *et al.* (2010)] Avetisyan, A. I., Campbell, R., Gupta, I., Heath, M. T., Ko, S. Y., Ganger, G. R., Kozuch, M. A., O'Hallaron, D., Kunze, M., Kwan, T. T., *et al.* (2010). Open cirrus: A global cloud computing testbed. *Computer*, 43(4):35–43.

[Bai *et al.* (2013)] Bai, X., Jin, H., Liao, X., Shi, X., and Shao, Z. (2013). Rtrm: a response time-based replica management strategy for cloud storage system. In *International Conference on Grid and Pervasive Computing*, pages 124–133. Springer.

[Balouek *et al.* (2012)] Balouek, D., Amarie, A. C., Charrier, G., Desprez, F., Jeannot, E., Jeanvoine, E., Lèbre, A., Margery, D., Niclausse, N., Nussbaum, L., *et al.* (2012). Adding virtualization capabilities to the grid'5000 testbed. In *International Conference on Cloud Computing and Services Science*, pages 3–20. Springer.

[Barisits *et al.* (2016)] Barisits, M., Kühn, E., and Lassnig, M. (2016). A Hybrid Simulation Model for Data Grids. In *Proc. of 16th IEEE/ACM Intl. Symp. on Cluster, Cloud and Grid Computing*, pages 255–260.

[Barisits *et al.* (2017)] Barisits, M.-S., Lassnig, M., Beermann, T., Garonne, V., Javurek, T., and Serfon, C. (2017). Automatic rebalancing of data in atlas distributed data management. Technical report, ATL-COM-SOFT-2016-143.

[Beermann *et al.* (2017)] Beermann, T., Lassnig, M., Barisits, M., Serfon, C., Garonne, V., Collaboration, A., *et al.* (2017). C3po-a dynamic data placement agent for atlas distributed data management. In *Journal of Physics: Conference Series*, volume 898, page 062012. IOP Publishing.

[Bell *et al.* (2003)] Bell, W. H., Cameron, D. G., Millar, A. P., Capozza, L., Stockinger, K., and Zini, F. (2003). OptorSim - A Grid Simulator for Studying Dynamic Data Replication Strategies. *IJHPCA*, 17(4):403–416.

[Ben-Yehuda *et al.* (2012)] Ben-Yehuda, O. A., Schuster, A., Sharov, A., Silberstein, M., and Iosup, A. (2012). Expert: Pareto-efficient task replication on grids and a cloud. In *Parallel & Distributed Processing Symposium (IPDPS), 2012 IEEE 26th International*, pages 167–178. IEEE.

[Bharathi *et al.* (2008)] Bharathi, S., Chervenak, A., Deelman, E., Mehta, G., Su, M.-H., and Vahi, K. (2008). Characterization of scientific workflows. In *Workflows in Support of Large-Scale Science, 2008. WORKS 2008. Third Workshop on*, pages 1–10. IEEE.

[Bird *et al.* (2014)] Bird, I., Carminati, F., Mount, R., Panzer-Steindel, B., Harvey, J., Fisk, I., Kersevan, B., Clarke, P., Girone, M., Buncic, P., *et al.* (2014). Update of the computing models of the wlcg and the lhc experiments. Technical report.

[Bobelin *et al.* (2012)] Bobelin, L., Legrand, A., Márquez, D. A. G., Navarro, P., Quinson, M., Suter, F., and Thiéry, C. (2012). Scalable multi-purpose network representation for large scale distributed system simulation. In *Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012)*, pages 220–227. IEEE Computer Society.

[Bonacorsi and Ferrari (2007)] Bonacorsi, D. and Ferrari, T. (2007). Wlcg service challenges and tiered architecture in the lhc era. In *IFAE 2006*, pages 365–368. Springer.

[Bsoul *et al.* (2011)] Bsoul, M., Al-Khasawneh, A., Abdallah, E. E., and Kilani, Y. (2011). Enhanced fast spread replication strategy for data grid. *Journal of Network and Computer Applications*, 34(2):575–580.

[Bsoul *et al.* (2016)] Bsoul, M., Abdallah, A., Almakadmeh, K., and Tahat, N. (2016). A Round-Based Data Replication Strategy. *IEEE TPDS*, 27(1):31–39.

[Buyya *et al.* (2011)] Buyya, R., Ranjan, R., Broberg, J., and Dias de Assuncao, M. (2011). Gridsim: A grid simulation toolkit for resource modelling and application scheduling for parallel and distributed computing.

[Cai *et al.* (2017)] Cai, Z., Li, Q., and Li, X. (2017). Elasticsim: A toolkit for simulating workflows with cloud resource runtime auto-scaling and stochastic task execution times. *Journal of Grid Computing*, 15(2):257–272.

[Calheiros *et al.* (2011)a] Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A. F., and Buyya, R. (2011a). CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms. *SPE*, 41(1):23–50.

[Calheiros *et al.* (2011)b] Calheiros, R. N., Ranjan, R., and Buyya, R. (2011b). Virtual machine provisioning based on analytical performance and qos in cloud computing environments. In *Parallel processing (ICPP), 2011 international conference on*, pages 295–304. IEEE.

[Calheiros and Buyya (2014)] Calheiros, R. N. and Buyya, R. (2014). Meeting deadlines of scientific workflows in public clouds with tasks replication. *IEEE Transactions on Parallel and Distributed Systems*, 25(7):1787–1796.

[Camarasu-Pop *et al.* (2010)] Camarasu-Pop, S., Glatard, T., Mościcki, J. T., Benoit-Cattin, H., and Sarrut, D. (2010). Dynamic partitioning of gate monte-carlo simulations on egee. *Journal of Grid Computing*, 8(2):241–259.

[Camarasu-Pop *et al.* (2013)a] Camarasu-Pop, S., Glatard, T., and Benoit-Cattin, H. (2013a). Simulating application workflows and services deployed on the european grid infrastructure. In *Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on*, pages 18–25. IEEE.

[Camarasu-Pop *et al.* (2013)b] Camarasu-Pop, S., Glatard, T., and Benoit-Cattin, H. (2013b). Simulating Application Workflows and Services Deployed on the European Grid Infrastructure. In *Proceedings of the 13th IEEE/ACM International. Symposium on Cluster, Cloud, and Grid Computing*, pages 18–25.

[Camarasu-Pop *et al.* (2013)c] Camarasu-Pop, S., Glatard, T., Da Silva, R. F., Gueth, P., Sarrut, D., and Benoit-Cattin, H. (2013c). Monte carlo simulation on heterogeneous distributed systems: A computing framework with parallel merging and checkpointing strategies. *Future Generation Computer Systems*, 29(3):728–738.

[Camarasu-Pop *et al.* (2016)] Camarasu-Pop, S., Glatard, T., and Benoit-Cattin, H. (2016). Combining Analytical Modeling, Realistic Simulation and Real Experimentation for the Optimization of Monte-Carlo Applications on the European Grid Infrastructure. *FGCS*, 57:13–23.

[Campana *et al.* (2014)] Campana, S., Brown, A., Bonacorsi, D., Capone, V., De Girolamo, D., Casani, A. F., Flix, J., Forti, A., Gable, I., Gutsche, O., Hesnaux, A., Liu, S., Lopez Munoz, F., Magini, N., McKee, S., Mohammed, K., Rand, D., Reale, M., Roiser, S., Zielinski, M., and Zurawski, J. (2014). Deployment of a WLCG Network Monitoring Infrastructure Based on the perfSONAR-PS Technology. In *Proceedings of the 20th International Conference on Computing in High Energy and Nuclear Physics*, volume 513 of *Journal of Physics: Conference Series*, page 062008.

[Cao *et al.* (2010)] Cao, H., Jin, H., Wu, X., Wu, S., and Shi, X. (2010). Dagmap: efficient and dependable scheduling of dag workflow job in grid. *The Journal of supercomputing*, 51(2):201–223.

[Carrión *et al.* (2015)] Carrión, I. M., Huedo, E., and Llorente, I. M. (2015). Interoperating grid infrastructures with the gridway metascheduler. *Concurrency and Computation: Practice and Experience*, 27(9):2278–2290.

[Casanova *et al.* (2014)] Casanova, H., Giersch, A., Legrand, A., Quinson, M., and Suter, F. (2014). Versatile, Scalable, and Accurate Simulation of Distributed Applications and Platforms. *Journal of Parallel and Distributed Computing*, 74(10):2899–2917.

[Chai *et al.* (2017)] Chai, A., Bazm, M.-M., Camarasu-Pop, S., Glatard, T., Benoit-Cattin, H., and Suter, F. (2017). Modeling Distributed Platforms from Application Traces for Realistic File Transfer Simulation. In *Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pages 54–63.

[Chai *et al.* (2018)] Chai, A., Camarasu-Pop, S., Glatard, T., Benoit-Cattin, H., and Suter, F. (2018). Evaluation through realistic simulations of file replication strategies for large heterogeneous distributed systems. In *Europar 2018-24th International European Conference on Parallel and Distributed Computing; Workshop HeteroPar 2018*, pages in–press.

[Challal and Bouabana-Tebibel (2010)] Challal, Z. and Bouabana-Tebibel, T. (2010). A priori replica placement strategy in data grid. In *Machine and Web Intelligence (ICMWI), 2010 International Conference on*, pages 402–406. IEEE.

[Chang and Chang (2008)] Chang, R.-S. and Chang, H.-P. (2008). A dynamic data replication strategy using access-weights in data grids. *The Journal of Supercomputing*, 45(3):277–295.

[Chen and Deelman (2012)] Chen, W. and Deelman, E. (2012). Workflowsim: A toolkit for simulating scientific workflows in distributed environments. In *E-science (e-science), 2012 IEEE 8th International Conference on*, pages 1–8. IEEE.

[Chervenak *et al.* (2007)] Chervenak, A. *et al.* (2007). Data placement for scientific applications in distributed environments. In *Proceedings of the 8th IEEE/ACM International Conference on Grid Computing*, pages 267–274.

[Chettaoui and Charrada (2014)] Chettaoui, H. and Charrada, F. B. (2014). A new decentralized periodic replication strategy for dynamic data grids. *Scalable Computing: Practice and Experience*, 15(1):101–119.

[Chtepen *et al.* (2009)] Chtepen, M., Dhoedt, B., De Turck, F., Demeester, P., Claeys, F. H., and Vanrolleghem, P. A. (2009). Adaptive checkpointing in dynamic grids for uncertain job durations. In *Information Technology Interfaces, 2009. ITI'09. Proceedings of the ITI 2009 31st International Conference on*, pages 585–590. IEEE.

[Cui *et al.* (2015)] Cui, Z., Zhang, Z., *et al.* (2015). Based on the correlation of the file dynamic replication strategy in multi-tier data grid. *International Journal of Database Theory and Application*, 8(1):75–86.

[David *et al.* (2014)] David, M. *et al.* (2014). Validation of Grid Middleware for the European Grid Infrastructure. *Journal of Grid Computing*, 12(3):543–558.

[Dayyani and Khayyambashi (2015)] Dayyani, S. and Khayyambashi, M. (2015). RDT: A New Data Replication Algorithm for Hierarchical Data Grid. *International Journal of Computer Science and Engineering*, 3(7):186–197.

[Da Silva *et al.* (2013)] Da Silva, R. F., Glatard, T., and Desprez, F. (2013). Self-healing of workflow activity incidents on distributed computing infrastructures. *Future Generation Computer Systems*, 29(8):2284–2294.

[Deelman *et al.* (2015)] Deelman, E., Vahi, K., Juve, G., Rynge, M., Callaghan, S., Maechling, P. J., Mayani, R., Chen, W., da Silva, R. F., Livny, M., *et al.* (2015). Pegasus, a workflow management system for science automation. *Future Generation Computer Systems*, 46:17–35.

[Desprez and Rouzaud-Cornabas (2013)] Desprez, F. and Rouzaud-Cornabas, J. (2013). *SimGrid Cloud Broker: Simulating the Amazon AWS Cloud*. PhD thesis, INRIA.

[Donno *et al.* (2008)] Donno, F., Abadie, L., Badino, P., Baud, J., Corso, E., Witt, S., Fuhrmann, P., Gu, J., Koblitz, B., Lemaitre, S., *et al.* (2008). Storage resource manager version 2.2: design, implementation, and testing experience. In *Journal of Physics: Conference Series*, volume 119, page 062028. IOP Publishing.

[Doğan (2009)] Doğan, A. (2009). A study on performance of dynamic file replication algorithms for real-time file access in data grids. *Future Generation Computer Systems*, 25(8):829–839.

[Elghirani *et al.* (2008)] Elghirani, A., Subrata, R., and Zomaya, A. (2008). A Proactive Non-Cooperative Game-Theoretic Framework for Data Replication in Data Grids. In *Proceedings of the 8th IEEE International Symposium on Cluster Computing and the Grid*, pages 433–440.

[Fadaie and Rahmani (2012)] Fadaie, Z. and Rahmani, A. M. (2012). A new replica placement algorithm in data grid. *International Journal of Computer Science Issues (IJCSI)*, 9(2):491.

[Fernandez *et al.* (2012)] Fernandez, J. A., Adolphsen, C., Akay, A., Aksakal, H., Albacete, J., Alekhin, S., Allport, P., Andreev, V., Appleby, R., Arikan, E., *et al.* (2012). A large hadron electron collider at cernreport on the physics and design concepts for machine and detector. *Journal of Physics G: Nuclear and Particle Physics*, 39(7):075001.

[Ferreira da Silva *et al.* (2015)] Ferreira da Silva, R., Rynge, M., Juve, G., Sfiligoi, I., Deelman, E., Letts, J., Würthwein, F., and Livny, M. (2015). Characterizing a High Throughput Computing Workload: The Compact Muon Solenoid (CMS) Experiment at LHC. *Procedia Computer Science*, 51:39–48.

[Fischl (2012)] Fischl, B. (2012). Freesurfer. *Neuroimage*, 62(2):774–781.

[Foster *et al.* (2001)] Foster, I., Kesselman, C., and Tuecke, S. (2001). The anatomy of the grid: Enabling scalable virtual organizations. *The International Journal of High Performance Computing Applications*, 15(3):200–222.

[Foster *et al.* (2008)] Foster, I., Zhao, Y., Raicu, I., and Lu, S. (2008). Cloud computing and grid computing 360-degree compared. In *Grid Computing Environments Workshop, 2008. GCE'08*, pages 1–10. Ieee.

[Garonne *et al.* (2014)] Garonne, V., Vigne, R., Stewart, G., Barisits, M., Lassnig, M., Serfon, C., Goossens, L., Nairz, A., Collaboration, A., *et al.* (2014). Rucio–the next generation of large scale distributed system for atlas data management. In *Journal of Physics: Conference Series*, volume 513, page 042021. IOP Publishing.

[Glatard *et al.* (2007)] Glatard, T., Montagnat, J., and Pennec, X. (2007). Optimizing Jobs Timeouts on Clusters and Production Grids. In *Proceedings of the 7th IEEE International Symposium on Cluster Computing and the Grid*, pages 100–107.

[Glatard *et al.* (2008)] Glatard, T., Montagnat, J., Lingrand, D., and Pennec, X. (2008). Flexible and Efficient Workflow Deployment of Data-Intensive Applications On Grids With MOTEUR. *IJHPCA*, 22(3):347–360.

[Glatard *et al.* (2013)] Glatard, T., Lartizien, C., *et al.* (2013). A Virtual Imaging Platform for Multi-Modality Medical Image Simulation. *IEEE Trans. on Medical Imaging*, 32(1):110–118.

[Glatard and Evans (2015)] Glatard, T. and Evans, A. C. (2015). Classifications of computing sites to handle numerical variability. In *2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, pages 863–870. IEEE.

[Grace and Manimegalai (2014)] Grace, R. K. and Manimegalai, R. (2014). Dynamic replica placement and selection strategies in data grids—a comprehensive survey. *Journal of Parallel and Distributed Computing*, 74(2):2099–2108.

[Grossman *et al.* (2009)] Grossman, R., Gu, Y., Sabala, M., Bennet, C., Seidman, J., and Mambratti, J. (2009). The open cloud testbed: A wide area testbed for cloud computing utilizing high performance network services. *arXiv preprint arXiv:0907.4810*.

[Gupta *et al.* (2017)] Gupta, H. *et al.* (2017). iFogSim: A Toolkit for Modeling and Simulation of Resource Management Techniques in The Internet of Things, Edge and Fog Computing Environments. *Software: Practice and Experience*, 47(9):1275–1296.

[Hanandeh *et al.* (2012)] Hanandeh, F., Khazaaleh, M., Ibrahim, H., and Latip, R. (2012). Cfs: a new dynamic replication strategy for data grids. *Int. Arab J. Inf. Technol.*, 9(1):94–99.

[Horri *et al.* (2008)] Horri, A., Sepahvand, R., and Dastghaibyfard, G. (2008). A hierarchical scheduling and replication strategy. *Int J Comput Sci Netw Secur*, 8(8):30–35.

[Howell and McNab (1998)] Howell, F. and McNab, R. (1998). Simjava: A discrete event simulation library for java. *Simulation Series*, 30:51–56.

[Iacobucci (2012)] Iacobucci, D. (2012). Mediation analysis and categorical variables: The final frontier. *Journal of Consumer Psychology*, 22(4):582–594.

[Iosup *et al.* (2006)] Iosup, A., Jan, M., Sonmez, O., and Epema, D. (2006). *On the dynamic resources availability in grids*. PhD thesis, INRIA.

[Iosup *et al.* (2011)] Iosup, A., Yigitbasi, N., and Epema, D. (2011). On the performance variability of production cloud services. In *Cluster, Cloud and Grid Computing (CCGrid), 2011 11th IEEE/ACM International Symposium on*, pages 104–113. IEEE.

[Iosup and Epema (2011)] Iosup, A. and Epema, D. (2011). Grid computing workloads. *IEEE Internet Computing*, 15(2):19–26.

[Issariyakul and Hossain (2012)] Issariyakul, T. and Hossain, E. (2012). Introduction to network simulator 2 (ns2). In *Introduction to Network Simulator NS2*, pages 21–40. Springer.

[Jan *et al.* (2004)] Jan, S. *et al.* (2004). GATE: a Simulation Toolkit for PET and SPECT. *Physics in Medicine and Biology*, 49(19):4543.

[Javadi *et al.* (2013)] Javadi, B., Kondo, D., Iosup, A., and Epema, D. (2013). The failure trace archive: Enabling the comparison of failure measurements and models of distributed systems. *Journal of Parallel and Distributed Computing*, 73(8):1208–1223.

[Juve *et al.* (2013)] Juve, G., Rynge, M., Deelman, E., Vockler, J.-S., and Berriman, G. B. (2013). Comparing futuregrid, amazon ec2, and open science grid for scientific workflows. *Computing in Science & Engineering*, 15(4):20–29.

[Kim *et al.* (2011)] Kim, W., Roopakalu, A., Li, K. Y., and Pai, V. S. (2011). Understanding and characterizing planetlab resource usage for federated network testbeds. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, pages 515–532. ACM.

[Kliazovich *et al.* (2012)] Kliazovich, D., Bouvry, P., and Khan, S. U. (2012). Greencloud: a packet-level simulator of energy-aware cloud computing data centers. *The Journal of Supercomputing*, 62(3):1263–1283.

[Kondo *et al.* (2010)] Kondo, D., Javadi, B., Iosup, A., and Epema, D. (2010). The failure trace archive: Enabling comparative analysis of failures in diverse distributed systems. In *Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on*, pages 398–407. IEEE.

[Kranzlmuller (2009)] Kranzlmuller, D. (2009). The future european grid infrastructure—roadmap and challenges. In *Information Technology Interfaces, 2009. ITI'09. Proceedings of the ITI 2009 31st International Conference on*, pages 17–20. IEEE.

[Lamehamedi *et al.* (2002)] Lamehamedi, H. *et al.* (2002). Data Replication Strategies in Grid Environments. In *Proceedings of the 5th International Conference on Algorithms and Architectures for Parallel Processing*, pages 378–383.

[Laure and Jones (2009)] Laure, E. and Jones, B. (2009). Enabling grids for e-science: The egee project. *Grid computing: infrastructure, service, and applications*, page 55.

[Lebre *et al.* (2015)] Lebre, A., Legrand, A., Suter, F., and Veyre, P. (2015). Adding storage simulation capacities to the simgrid toolkit: Concepts, models, and api. In *Cluster, Cloud and Grid Computing (CCGrid), 2015 15th IEEE/ACM International Symposium on*, pages 251–260. IEEE.

[Lee *et al.* (2012)] Lee, M.-C., Leu, F.-Y., and Chen, Y.-p. (2012). Pfrf: An adaptive data replication algorithm based on star-topology data grids. *Future Generation Computer Systems*, 28(7):1045–1057.

[Lei *et al.* (2008)] Lei, M., Vrbsky, S., and Hong, X. (2008). An On-Line Replication Strategy to Increase Availability in Data Grids. *Future Generation Computing Systems*, 24(2):85–98.

[Long *et al.* (2006)] Long, S. J., Long, J. S., and Freese, J. (2006). *Regression models for categorical dependent variables using Stata*. Stata press.

[Loukopoulos and Ahmad (2004)] Loukopoulos, T. and Ahmad, I. (2004). Static and adaptive distributed data replication using genetic algorithms. *Journal of Parallel and Distributed Computing*, 64(11):1270–1285.

[Löschen and Müller-Pfefferkorn] Löschen, C. and Müller-Pfefferkorn, R. Providing grid data access on srm and lfc to unicore. *Schriften des Forschungszentrums Jülich IAS Series Volume 15*, page 95.

[Ma *et al.* (2013)] Ma, J., Liu, W., and Glatard, T. (2013). A classification of file placement and replication methods on grids. *Future Generation Computer Systems*, 29(6):1395–1406.

[Ma *et al.* (2015)] Ma, J., Liu, W., and Glatard, T. (2015). A stateful storage availability and entropy model to control storage distribution on grids. *Concurrency and Computation: Practice and Experience*, 27(1):164–171.

[Maier *et al.* (2018)] Maier, T., Beermann, T. A., Duckeck, G., Lassnig, M., Vukotic, I., and Legger, F. (2018). Performance and impact of dynamic data placement in atlas. Technical report, ATL-COM-SOFT-2018-047.

[Mambretti *et al.* (2015)] Mambretti, J., Chen, J., and Yeh, F. (2015). Next generation clouds, the chameleon cloud testbed, and software defined networking (sdn). In *2015 International Conference on Cloud Computing Research and Innovation (ICCCRI)*, pages 73–79. IEEE.

[Mansouri and Dastghaibyfard (2012)] Mansouri, N. and Dastghaibyfard, G. H. (2012). A dynamic replica management strategy in data grid. *Journal of network and computer applications*, 35(4):1297–1303.

[Mansouri and Dastghaibyfard (2013)a] Mansouri, N. and Dastghaibyfard, G. H. (2013a). Enhanced dynamic hierarchical replication and weighted scheduling strategy in data grid. *Journal of parallel and distributed computing*, 73(4):534–543.

[Mansouri and Dastghaibyfard (2013)b] Mansouri, N. and Dastghaibyfard, G. H. (2013b). Job scheduling and dynamic data replication in data grid environment. *The Journal of Supercomputing*, 64(1):204–225.

[Miller *et al.* (2015)] Miller, M. A., Schwartz, T., Pickett, B. E., He, S., Klem, E. B., Scheuermann, R. H., Passarotti, M., Kaufman, S., and O'Leary, M. A. (2015). A restful api for access to phylogenetic tools via the cipres science gateway. *Evolutionary Bioinformatics*, 11:EBO–S21501.

[Montagnat *et al.* (2009)] Montagnat, J., Isnard, B., Glatard, T., Maheshwari, K., and Fornarino, M. B. (2009). A data-driven workflow language for grids based on array programming principles. In *Proceedings of the 4th Workshop on Workflows in Support of Large-Scale Science*, page 7. ACM.

[Montagnat *et al.* (2010)] Montagnat, J., Glatard, T., Reimert, D., Maheshwari, K., Caron, E., and Desprez, F. (2010). Workflow-based comparison of two distributed computing infrastructures. In *Workflows in Support of Large-Scale Science (WORKS), 2010 5th Workshop on*, pages 1–10. IEEE.

[Mościcki *et al.* (2011)] Mościcki, J., Lamanna, M., Bubak, M., and Sloot, P. M. (2011). Processing moldable tasks on the grid: Late job binding with lightweight user-level overlay. *Future Generation Computer Systems*, 27(6):725–736.

[Nazir *et al.* (2009)] Nazir, B., Qureshi, K., and Manuel, P. (2009). Adaptive checkpointing strategy to tolerate faults in economy based grid. *The Journal of Supercomputing*, 50(1):1–18.

[Ostermann *et al.* (2008)a] Ostermann, S., Prodan, R., Fahringer, T., Iosup, A., and Epema, D. (2008a). A trace-based investigation of the characteristics of grid workflows. In *From Grids to Service and Pervasive Computing*, pages 191–203. Springer.

[Ostermann *et al.* (2008)b] Ostermann, S., Prodan, R., Fahringer, T., Iosup, R., and Epema, D. (2008b). On the characteristics of grid workflows. In *CoreGRID Symposium-Euro-Par*, volume 2008, pages 1–12.

[Ostermann *et al.* (2010)] Ostermann, S., Prodan, R., and Fahringer, T. (2010). Dynamic Cloud Provisioning for Scientific Grid Workflows. In *Proc. of the 11th ACM/IEEE International Conference on Grid Computing*, pages 97–104.

[Park *et al.* (2003)] Park, S.-M., Kim, J.-H., Ko, Y.-B., and Yoon, W.-S. (2003). Dynamic data grid replication strategy based on internet hierarchy. In *International Conference on Grid and Cooperative Computing*, pages 838–846. Springer.

[Poola *et al.* (2016)] Poola, D., Ramamohanarao, K., and Buyya, R. (2016). Enhancing reliability of workflow execution using task replication and spot instances. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 10(4):30.

[Pérez *et al.* (2010)] Pérez, J. M., García-Carballeira, F., Carretero, J., Calderón, A., and Fernández, J. (2010). Branch replication scheme: A new model for data replication in large scale data grids. *Future Generation Computer Systems*, 26(1):12–20.

[REN *et al.* (2010)] REN, X.-y., WANG, R.-c., and Qiang, K. (2010). Using optorsim to efficiently simulate replica placement strategies. *The Journal of China Universities of Posts and Telecommunications*, 17(1):111–119.

[Ramakrishnan *et al.* (2007)] Ramakrishnan, A., Singh, G., *et al.* (2007). Scheduling data-intensive workflows onto storage-constrained distributed resources. In *Cluster Computing and the Grid, 2007. CCGRID 2007. Seventh IEEE International Symposium on*, pages 401–409. IEEE.

[Ranganathan *et al.* (2002)] Ranganathan, K., Iamnitchi, A., and Foster, I. (2002). Improving data availability through dynamic model-driven replication in large peer-to-peer communities. In *Cluster Computing and the Grid, 2002. 2nd IEEE/ACM International Symposium on*, pages 376–376. IEEE.

[Ranganathan and Foster (2001)] Ranganathan, K. and Foster, I. (2001). Identifying dynamic replication strategies for a high-performance data grid. In *International Workshop on Grid Computing*, pages 75–86. Springer.

[Ranganathan and Foster (2002)] Ranganathan, K. and Foster, I. (2002). Decoupling Computation and Data Scheduling in Distributed Data-Intensive Applications. In *Proc. of the 11th IEEE Intl. Symp. on High Performance Distributed Computing*, pages 352–358.

[Ranganathan and Foster (2003)] Ranganathan, K. and Foster, I. (2003). Simulation studies of computation and data scheduling algorithms for data grids. *Journal of Grid computing*, 1(1):53–62.

[Rasool *et al.* (2009)] Rasool, Q., Li, J., and Zhang, S. (2009). Replica placement in multi-tier data grid. In *Dependable, Autonomic and Secure Computing, 2009. DASC'09. Eighth IEEE International Conference on*, pages 103–108. IEEE.

[Reiss *et al.* (2012)] Reiss, C., Tumanov, A., Ganger, G. R., Katz, R. H., and Kozuch, M. A. (2012). Heterogeneity and dynamicity of clouds at scale: Google trace analysis. In *Proceedings of the Third ACM Symposium on Cloud Computing*, page 7. ACM.

[Romanus *et al.* (2012)] Romanus, M., Mantha, P. K., McKenzie, M., Bishop, T. C., Gallichio, E., Merzky, A., El Khamra, Y., and Jha, S. (2012). The anatomy of successful ecss projects: lessons of supporting high-throughput high-performance ensembles on xsede. In *Proceedings of the 1st Conference of the Extreme Science and Engineering Discovery Environment: Bridging from the eXtreme to the campus and beyond*, page 46. ACM.

[Sadashiv and Kumar (2011)] Sadashiv, N. and Kumar, S. D. (2011). Cluster, grid and cloud computing: A detailed comparison. In *Computer Science & Education (ICCSE), 2011 6th International Conference on*, pages 477–482. IEEE.

[Sashi and Thanamani (2011)] Sashi, K. and Thanamani, A. S. (2011). Dynamic replication in a data grid using a modified bhr region based algorithm. *Future Generation Computer Systems*, 27(2):202–210.

[Sato *et al.* (2008)] Sato, H., Matsuoka, S., Endo, T., and Maruyama, N. (2008). Access-Pattern and Bandwidth Aware File Replication Algorithm in a Grid Environment. In *Proceedings of the 9th IEEE/ACM International Conference on Grid Computing*, pages 250–257.

[Shorfuzzaman *et al.* (2010)] Shorfuzzaman, M., Graham, P., and Eskicioglu, R. (2010). Adaptive Popularity-Driven Replica Placement in Hierarchical Data Grids. *The Journal of Supercomputing*, 51(3):374–392.

[Siaterlis *et al.* (2013)] Siaterlis, C., Garcia, A. P., and Genge, B. (2013). On the use of emulab testbeds for scientifically rigorous experiments. *IEEE Communications Surveys & Tutorials*, 15(2):929–942.

[Smith *et al.*] Smith, S., Woolrich, M., Behrens, T., Beckmann, C., Flitney, D., Jenkinson, M., Bannister, P., Clare, S., De Luca, M., Hansen, P., *et al.* Fmrib software library.

[Suter *et al.* (2016)] Suter, F., Chai, A., and Bazm, M. (2016). The Log2sim Framework. Available at: http://github.com/frs69wq/log2sim.

[Tierney *et al.* (2009)] Tierney, B., Metzger, J., Boote, J., Boyd, E., Brown, A., Carlson, R., Zekauskas, M., Zurawski, J., Swany, M., and Grigoriev, M. (2009). perfsonar: Instantiating a global network measurement framework. *SOSP Wksp. Real Overlays and Distrib. Sys.*

[Topcuoglu *et al.* (2002)] Topcuoglu, H., Hariri, S., and Wu, M.-y. (2002). Performance-effective and low-complexity task scheduling for heterogeneous computing. *IEEE transactions on parallel and distributed systems*, 13(3):260–274.

[Tos *et al.* (2015)] Tos, U., Mokadem, R., Hameurlain, A., Ayav, T., and Bora, S. (2015). Dynamic replication strategies in data grid systems: a survey. *The Journal of Supercomputing*, 71(11):4116–4140.

[Towns *et al.* (2014)] Towns, J., Cockerill, T., Dahan, M., Foster, I., Gaither, K., Grimshaw, A., Hazlewood, V., Lathrop, S., Lifka, D., Peterson, G. D., *et al.* (2014). Xsede: accelerating scientific discovery. *Computing in Science & Engineering*, 16(5):62–74.

[Tsaregorodtsev *et al.* (2010)] Tsaregorodtsev, A. *et al.* (2010). DIRAC3 – the New Generation of the LHCb Grid Software. *Journal of Physics: Conference Series*, 219(6):062029.

[Tu *et al.* (2008)] Tu, M., Li, P., Yen, I.-L., Thuraisingham, B., and Khan, L. (2008). Secure data objects replication in data grid. *IEEE Transactions on dependable and secure computing*, (1):50–64.

[Van Reeth *et al.* (2015)] Van Reeth, E., Soujanya, G., Sdika, M., Cervenansky, F., and Poh, C. (2015). Misosr: Medical image isotropic super-resolution reconstruction. *Midas Journal*.

[Vashisht *et al.* (2014)] Vashisht, P., Kumar, R., and Sharma, A. (2014). Efficient dynamic replication algorithm using agent for data grid. *The Scientific World Journal*, 2014.

[Velho *et al.* (2013)] Velho, P., Schnorr, L. M., Casanova, H., and Legrand, A. (2013). On the Validity of Flow-Level TCP Network Models for Grid and Cloud Simulations. *ACM TOMACS*, 23(4):23.

[Velho and Legrand (2009)] Velho, P. and Legrand, A. (2009). Accuracy Study and Improvement of Network Simulation in the SimGrid Framework. In *Proc. of the 2nd Intl. Conf. on Simulation Tools and Techniques for Communications, Networks and Systems*.

[Von Laszewski *et al.* (2010)] Von Laszewski, G., Fox, G. C., Wang, F., Younge, A. J., Kulshrestha, A., Pike, G. G., Smith, W., Voeckler, J., Figueiredo, R. J., Fortes, J., *et al.* (2010). Design of the futuregrid experiment management framework. In *Gateway Computing Environments Workshop (GCE), 2010*, pages 1–10. IEEE.

[Vrbsky *et al.* (2010)] Vrbsky, S., Lei, M., Smith, K., and Byrd, J. (2010). Data Replication and Power Consumption in Data Grids. In *Proceedings of the 2nd IEEE International Conference on Cloud Computing Technology and Science*, pages 288–295.

[Wassenaar *et al.* (2012)] Wassenaar, T. A., Van Dijk, M., Loureiro-Ferreira, N., Van Der Schot, G., De Vries, S. J., Schmitz, C., Van Der Zwan, J., Boelens, R., Giachetti, A., Ferella, L., *et al.* (2012). Wenmr: structural biology on the grid. *Journal of Grid Computing*, 10(4):743–767.

[Wu *et al.* (2008)] Wu, J.-J., Lin, Y.-F., and Liu, P. (2008). Optimal replica placement in hierarchical data grids with locality assurance. *Journal of Parallel and Distributed Computing*, 68(12):1517–1538.

[Xiong *et al.* (2013)] Xiong, F., ZHU, X.-x., HAN, J.-y., and WANG, R.-c. (2013). Qos-aware replica placement for data intensive applications. *The Journal of China Universities of Posts and Telecommunications*, 20(3):43–47.

[Yang *et al.* (2010)] Yang, C.-T., Fu, C.-P., and Hsu, C.-H. (2010). File Replication, Maintenance, and Consistency Management Services in Data Grids. *The Journal of Supercomputing*, 53(3):411–439.

[Zhao *et al.* (2010)a] Zhao, L., Lee, W., Song, C. X., Huber, M., and Goldner, A. (2010a). Bringing high performance climate modeling into the classroom. In *Proceedings of the 2010 TeraGrid Conference*, page 24. ACM.

[Zhao *et al.* (2010)b] Zhao, W., Xu, X., Wang, Z., Zhang, Y., and He, S. (2010b). A dynamic optimal replication strategy in data grid environment. In *Internet Technology and Applications, 2010 International Conference on*, pages 1–4. IEEE.

[Zhong *et al.* (2010)] Zhong, H., Zhang, Z., and Zhang, X. (2010). A dynamic replica management strategy based on data grid. In *Grid and Cooperative Computing (GCC), 2010 9th International Conference on*, pages 18–23. IEEE.

[amazon ec2] amazon ec2. https://aws.amazon.com/ec2.

[google cloud] google cloud. https://cloud.google.com/.

[prace project] prace project. http://www.prace-project.eu.

[windows azure] windows azure. https://azure.microsoft.com/en-us/.