Adaptable Distance-Based Decision-Making Support in Dynamic Cross-Grid Environment

Julien Gossa, Jean-Marc Pierson, Lionel Brunie

Adaptable distance-based decision-making support in dynamic cross-grid environment

Julien Gossa[1], Jean-Marc Pierson[2], and Lionel Brunie[1]

[1] LIRIS INSA-Lyon, UMR5205 F-69621, France,
{firstname}.{lastname}@liris.cnrs.fr.
[2] IRIT University Paul Sabatier UMR5505 F-31062, France,
{lastname}@liris.cnrs.fr.

Abstract. The grid environment presents numerous opportunities for business applications as well as for scientific ones. Nevertheless the current trends seem to lead to several independent specialized grids in opposition to the early visions of one generic world wide grid. In such a cross-grid context, the environment might be harder to manipulate whereas more decisions must be handled from user-side. Our proposal is a distance-based decision-making support designed to be usable, adaptable and accurate. Our main contribution is to ensure the profitability of classical monitoring solutions by improving their usability. Our approach is illustrated and validated with experiments in a real grid environment. [3]

1 Introduction

The term Grid Islands has been proposed by GridBus project [1] to describe the situation of several grid solutions cohabiting without any collaboration. In their paper, De Assunao and al. analyze the lack of interoperability between grid islands and the necessity of transparent and secure interlinks to build a large World Wide Grid. Our vision of grid evolution is slightly different: Each specialized domain has so specific needs and constraints that it seems very difficult to satisfy them in a generic environment. Consequently, the perspective of several specialized and adapted grids is more realistic. In such a context the users interact with several grids middlewares. This interaction is allowed by Service Oriented Architectures, as WSRF-based grids are now usable with the standard software equipment. Many proposals already address the security requirement. But few works are interesting in the requirement of a usable decision-making support. It presents numerous challenges and requirements that must be mandatory addressed, as illustrated in the next section.

1.1 Use Case

The figure 1 shows a cross-grid environment composed of nine sites distributed throughout France. One VO is composed of two grids: $Grid1$ and $Grid2$. One

[3] This work is supported by the French ministry of research within the ACI "masse de données" (Grid for Geno-Medicine Project)

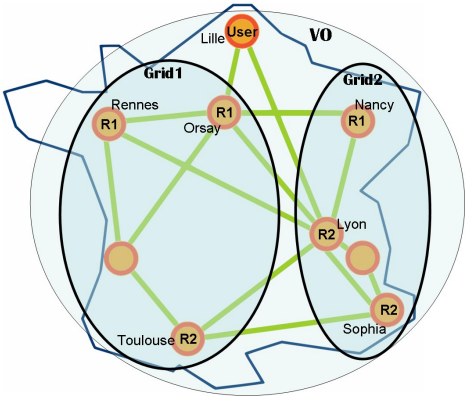


Fig. 1. Example of user decision-making in cross-grid context

member of this grid is a biologist located at Lille. He has to compute the signature of a medical image. This classical task implies generally two resources: $R1$ an image normalization service and $R2$ a signature computation service. These are replicated on both grids. He knows the size of initial image, the size of the intermediate one and the size of signature. Moreover, he knows that signature computation is CPU expensive.

The task can be submitted either to $Grid1$ or to $Grid2$. Assuming that grid middlewares are able to automatically compose resources, $R1$ and $R2$ are linked automatically. Another alternative is to handle the composition from user side and to use $R1$ and $R2$ from different grids. This last possibility seems less efficient as the user must retrieve the output from $R1$ and resend it to $R2$.

To make this decision, the user has access to the monitoring information about the infrastructure: CPU speed and load of each resource, latency and bandwidth between the resources... But using these data is difficult as the task implies both computation and communication with different magnitude: While the normalization is mainly communication expensive, the signature computation is mainly CPU expensive. Moreover the possible combinations of resources are numerous and consequently confusing. Unfortunately in a cross-grid environment the users can not rely on the classical grid decision-making units, such as resource brokers or schedulers, as they are not designed to collaborate. Some decisions must be handled from the user side.

In such a case, the user needs to rely on a usable decision-making support.

1.2 Grid Decision-Making Support

The decisions in grid environment concern all grid resources: data, databases, services, hosts... And all users: end-users selecting resources to submit tasks, administrators planning deployment, developers sizing software...

In SOA the number of services is meant to dramatically increase. In this context, the optimization of the whole architecture relies on the optimization of

each service. Consequently the decision-making support will be needed at each level of the architecture and must be adaptable to several contexts.

Several aspects must be taken into account: (1) *The infrastructure topology and condition*: communication, computation and storage capabilities... (2) *The characteristics of the task*: communication, computation and storage needs... And (3) *the objectives to be achieved*: End-user response delay, load balancing, optimization of financial costs...

Moreover, decision-making supports have some mandatory requirements:

Usability, Accuracy, and Profitability : They must be available under the best conditions and the provided information must be as close as possible of user concerns, otherwise it will not be used. Furthermore, the ratio profit/cost must be satisfactory.

Adaptability, Flexibility and Extensibility : They must be suitable whatever the user needs are or the infrastructures evolution is.

Scalability : They must be able to handle world-scale decisions as easily as simple ones, particularly from the user point of view.

This article is organized as follows. First in Section 2 we present the existing decision-making supports provided by grid environments and their limitations. Second we describe our proposal in Section 3 and analyse some experimentations in Section 4. Finally, we conclude with perspectives in Section 5.

2 Related Work

Decision-making in existing grid environment is mainly supported by monitoring systems. The identification of relevant metrics for grid environment and measurement methods has been made by the Network Measurements Working Group of the Global Grid Forum in [2]. Recent developments in grid infrastructure have lead to effective tools providing these measurements, such as the Monitoring and Discovery Service of Globus [3], R-GMA [4], and SCALEA-G [5]. Most of them are based on the Grid Monitoring Architecture (GMA). Another approach is adopted by the Network Weather Service [6]. It is able to capture the condition of both network and hosts. It can provide the raw measurements of the classical metrics as well as forecasts.

An alternative to monitoring system providing several raw metric measurements, is the concept of *distance*. Several Distance Vector protocols (RIP, IGRP, EIGRP, OSPF,...) have been implemented for routing in packet-switched networks. Here, the concept of *distance* is mostly the number of hops between two end-points. Since then, this concept has often been reused for very different purposes such as data management, network topology discovering, resource brokering, nodes clustering, etc. We do think that the popularity of this concept comes from its similarity with our real world. So it constitutes a precious help in the understanding of the network and in the elaboration of decision-making processes. The Distance Map Services (DMS) such as IDMaps in [7] and Global Network Positioning (GPN) [8] aim at providing an estimation of the distance

between all the hosts of a network while minimizing the number of measurements. Most of them are limited to the latency which is the easiest and most inexpensive metric to measure.

Both monitoring systems and DMS aim at efficiency and scalability. They are too low-level to be really usable as a full support for decision-making: Their users have to deal with raw metrics such as latencies or CPU loads which are far from their actual concerns. In grid environment, the tasks and goals are more complex. Furthermore, while DMS provide too limited information, monitoring systems are highly resource expensive. Producing and providing the monitoring information consume the monitored resource as well as communication resources. It might be a waste if the monitoring data are not fully exploited.

Consequently, they are inadequate to fully support decision-making in grid environment. There is a need for an advanced decision-making support, improving the usability while achieving a good adaptability and accuracy.

3 NDS: The Network Distance Service

Our proposal is a distance-based decision support. It provides distances adaptable to any given *task*. We call *task* an interaction between hosts of a network. Generally, it corresponds to the invocation of one service. But it may be more basic tasks such as data retrieval or storage. Such distances are meant to be more usable than raw metrics and more relevant than the distance provided by Distance Map Services. Basically, they are based on the composition of different raw metrics provided by external monitoring systems. This computation is embedded in a Web Service developed with Globus Toolkit 4: the Network Distance Service.

3.1 Metric Model

According to the GGF NM-WG in [2], *a metric is a quantity related to the performance and reliability of the Internet*. We call *measures* the actual values related to metrics.

We note M the set of metrics. It includes the *bandwidth* (BW), the *latency* (L), the *CPU speed* ($CPUs$), and the *CPU availability* ($CPUa$). The measure of the network metric m from the host i to the host j is noted $m_{i,j}$. The measure of the host metric m for the host i is noted m_i.

NDS must be able to use the measures provided by any monitoring tools. Thus monitoring tools are accessed through command lines execution. Then, no wrapper has to be developed and new metrics and tools are integrated through a simple JNDI configuration file. The only requirement is that the monitoring tools must by queryable from the execution host of NDS.

3.2 Compound Metric Model

In order to come closer to user concerns and to improve its usability, NDS embeds two compound metrics related to the two basic kinds of grid task: data transfers or computations.

DTC for *Data Transfer Cost*. It assesses the cost to transfer a piece of data of size *data_size* from one host i to another j. According to [9], "the Raw Bandwidth model using NWS forecasts can be used effectively to rank alternative candidate schedules". Actually, we observe that data size does not influence only distance, but also relevance of the different metrics: Latency is the key factor about small size data, whereas Bandwidth is the key factor about large ones. Thus DTC includes the 3 RTT needed to open and close TCP/IP connections.

$$DTC_{i,j}(data\_size) = 3 \times (L_{i,j} + L_{j,i}) + \frac{data\_size}{BW_{i,j}}$$

CTC for *Computation Task Cost* assesses of the cost to execute *nb_cycles* CPU cycles on the host i. *nb_cycles* can be obtained either by calibration or compilation technics. Its automatic extraction will be investigated in future work.

$$CTC_i(nb\_cycles) = \frac{nb\_cycles}{CPUs_i \times CPUa_i}$$

One can note that theses compound metrics are rather basic. Some advanced characteristics are not taken into account, such as host architecture, OS, bus frequency, buffer size, scheduler configuration, protocol, MTU, TCP/IP configuration... Nevertheless, tools like NWS measure the metrics from the application layer. Consequently they already take all the influencing characteristics into account without having to identify them.

Moreover compound metrics are declared as easily as raw metrics: It is easy to refine them or to add new ones assessing of any goals. Even non-functional aspects like financial costs can be integrated as soon as the raw data are available.

3.3 NDS Queries

The NDS queries include three sections:

Set of hosts: The set of names of hosts involved in the decision. It can include several named subsets and is noted $\mathcal{H}$.

Task Properties Set: The set of task parameters involved in the distance computation which are not monitoring metrics. It is a set of named values noted TPS.

Distance Function: The real-valued function that gives the final value of the distance between two nodes. We note it df.
It is a nonlinear combination of monitoring metrics and values in TPS. This function is the core of distance computation. It must be relevant to the

aspects the user wants to assess. Generally its result must tend to zero when the performances tend to perfection, as distances generally represent costs.

The next section shows some examples of NDS queries in concrete use cases.

4 Experiments

The experiments are made on Grid5000 [10]. This experimental platform allows to reserve nodes to conduct distributed experiments. Its topology is shown in the Figure 1. Its network is high-performance (GB/s) and shared among all users. The nodes are not shared during reservation and heterogeneous architectures are represented (AMD and Intel, 32 bits and 64 bits, simple and double core). The raw metrics used by NDS in distance computations are provided by NWS.

Our approach is validated by comparing the execution time for all possible decisions. These times are retrieved with a Fake Service: The Fake Client sends *in* random bytes to the Fake Service; Then this one makes *comp* divisions of double typed variables; Finally, *out* random bytes are returned to either the client or another Fake Service. This allows to emulate a large range of resource. All presented times are means of 10 experiments.

4.1 Selection

The problem presented in section 1.1 is to decide which instances of $R1$ and $R2$ optimize the execution time. This problem can be solved by computing and sorting the distance representing the task performance. This distance must include $s1 = 10MB$ the size of the initial image, $s2 = 100KB$ the size of the intermediate image, $s3 = 1KB$ the size of the signature, $c1 = 100\ 000$ the number of CPU cycles needed by the normalization, and $c2 = 10\ 000\ 000$ the number of CPU cycles needed by the signature computation.

The corresponding NDS query is:

$$\begin{cases} \mathcal{V} & = \{client = \{\text{node-36.lille}\}, \\ & \quad R1 = \{\text{gdx0039.orsay, grillon-20.nancy, helios51.sophia}\}, \\ & \quad R2 = \{\text{node-2.lyon, paravent74.rennes, node-25.toulouse}\}\} \\ TPS & = \{s1{=}10\ 000\ 000,\ s2{=}100\ 000,\ s3{=}1\ 000,\ c1{=}100\ 000,\ c2{=}10\ 000\ 000\} \\ df & = DTC_{client,R1}(s1) + CTC_{R1}(c2) \\ & \quad + (sameVO_{R1,R2})?DTC_{R1,R2}(s2) : DTC_{R2,client}(s2) + DTC_{client,R2}(s2) \\ & \quad + CTC_{R2}(c2) + DTC_{R2,client}(s3) \end{cases}$$

Where *client*, $R1$, and $R2$ into *df* represent the client host and the instances of resources declared in $\mathcal{V}$. In this example *df* represents the costs to achieve the whole task using $R1$ and $R2$. Please note the use of the special test $sameVO$ returning *true* if the given parameters are members of a same VO. It is integrated in NDS as a raw metric and implemented into a fake external tools that can be replaced by a real one if available. It shows the expressivity of NDS which

uses the library Java math Expression Parser (JEP [11]) to parse df. It supports classical mathematical operations and functions as well as Boolean tests and character strings.

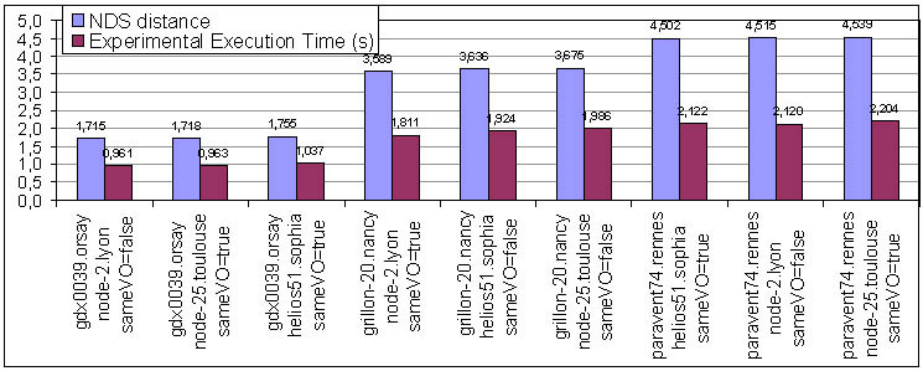


Fig. 2. Experimental execution times and NDS distances according to $R1$ and $R2$

The figure 2 shows both the experimental execution times (in seconds) and the distances given by NDS (in arbitrary units) according to the selection of $R1$ and $R2$. One can note that the 9 alternate possibilities are almost perfectly ranked by NDS and that the best solution was hard to guess as `gdx0039.orsay` and `node-2.lyon` are not in the same grid. Furthermore the important decision is actually the selection of $R1$. Indeed the results are sensibly equal whatever the selection of $R2$ is. This shows that first data transfer is the key factor of the efficiency of the whole execution, which was hard to guess too.

An important observation is that NDS distances are not exactly directly proportional to the real execution times. Consequently they can not be expressed in physical units like seconds and they are not previsions and must be used for ranking exclusively. Nevertheless, NDS distances can be used to support decisions in a wide range of applications, for instance in deployment tasks.

4.2 Deployment

Another problem is to decide how many instances of a resources must be deployed and where. In graph theory, this problem is called the *k-medians problem*:

Given a set V of points in a metric space endowed with a metric distance function df, and given a desired number k of resulting clusters, partition S into non-overlapping clusters $C_1, \ldots, C_k$ and determine their "centres"

$$\mu = \{\mu_1, \ldots, \mu_k\} \subset V \text{ so that } criterion = \sum_{i \in \mathcal{V}} min_{j=1}^{k}\ df(i, \mu_j) \text{ is minimized.}$$

In our scenario, k is the number of instances of the resource, while $\mu_1, \ldots, \mu_k$ are their optimal locations. We show in [12] how the distance produced by NDS

can be validated to be "metric distance". NDS embeds an algorithm to compute and compare the *criterion* of each possible solution. It was used to make decisions about the placement of $R1$ and $R2$.

We assume that the resources clients have been identified on 19 nodes of 7 sites of Grid5000: `Lyon` (3 nodes), `Rennes` (2 nodes), `Orsay` (2 nodes), `Sophia-Antipolis` (4 nodes), `Toulouse` (2 nodes), `Nancy` (2 nodes) and `Lille` (1 nodes). Moreover we assume the client requests distribution uniform and the resource deployable on any of the 19 nodes.

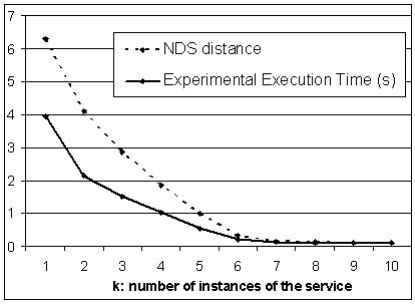
The corresponding NDS query is:

$$\begin{cases} \mathcal{V} & = \{client = \{\text{sagittaire-18.lyon, ... (all of the 19 hosts) }\}, \\ & \quad location = \{\text{sagittaire-18.lyon, ... (all of the 19 hosts)}\}\} \\ TPS & = \{in, out, comp\} \\ df & = DTC_{client,location}(in) + CTC_{location}(comp) + DTC_{location,client}(out) \end{cases}$$

The provided distances assesses a resource invocation from all nodes to all nodes with *in* bytes of input, *comp* CPU cycles of computation, and *out* bytes of output. It allows executing the k-medians algorithm and making the deployment decisions to optimize global resource access time, as stated below.

Deployment of $R1$: $TPS = \{in = 10\,000\,000, out = 100\,000, comp = 100\,000\}$

Fig. 3. Experimental execution times (in seconds) according to the deployment of $R1$



decision	time	accuracy	μ with $k = 2$
NDS	1.125	91.41%	gdx0077.orsay sagittaire-44.lyon
best	1.029	100.00%	node-25.toulouse node-2.lyon
median	1.249	82.36%	node-85.sophia gdx0039.orsay
worst	1.788	36.89%	paravent74.rennes grillon-39.nancy

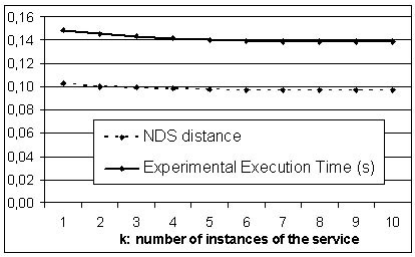
The Figure 3 shows k-medians algorithms results in two forms: First and according to k the best case of (1) the mean experimental execution times from all of the 19 nodes to their closest resource and (2) the distance-based k-server criteria. Obviously as $R1$ is communication expensive, deploying numerous instances allows to come closer to the clients and consequently to improve the performance. But NDS highlights two particular values $k = 2$ and $k = 6$ where the tangent line changes: The speed-up is very impressive from $k = 1$ to $k = 2$, good from $k = 2$ to $k = 6$, and null afterwards. This is perfectly assessed by

NDS and allows the user to decide how many instances he will deploy according to the gain and cost to add a new resource.

Second, the table shows the experimental result with $k = 2$ of four placements: recommended by NDS thanks to the k-medians algorithm, and experimental best, median and worst. The mean execution time from all the 19 nodes shows that the NDS recommendation achieves a good accuracy, especially regarding the median and worst decisions which might possibly be made by intuitive means.

Deployment of *R2*: $TPS = \{in = 100\ 000, out = 1\ 000, comp = 10\ 000\ 000\}$.

Fig. 4. Execution times (in seconds) according to the deployment of $R2$



decision	time	accuracy	μ with $k = 1$
NDS	0.140	100.00%	sagittaire-18.lyon
best	0.140	100.00%	sagittaire-18.lyon
median	0.166	84.49%	node-6.toulouse
worst	0.186	75,44%	grillon-39.nancy

In Figure 4 one can see that the replication of $R2$ does not lead to any speed-up, which is perfectly assessed by NDS. Indeed as $R2$ is mainly computation expensive, it does not need to be brought close to each client. Then one can decide to deploy only one instance. Moreover, the NDS placement recommendation is perfect while a average decision might lead to a loss of more than 15%.

Global deployment results and discussion: We have tested the NDS recommendations with $\{in, out, comp\} \in \{\{1\ 000, 100\ 000, 10\ 000\ 000\}^3\}, \forall k \in [1, 19]$. They achieve a global mean accuracy of 94.26%.

The main limitation of NDS in deployment problems, is the necessity to fix the TPS values. This can be avoided by computing the integral of df over the values of TPS according to their distribution. Moreover, the change in speed-up can be highlighted by the derivative of the k-medians criterion with respect to k. This is a part of our future work.

The purpose of this experiment is to show the adaptability and usability of our service: A large variety of complex and concrete user problems can be accurately solved with actually simple queries. NDS does not directly address the scalability. Actually NDS can handle as many host as necessary, but the underlying monitoring system has to be exhaustive and thus might present scalability issues. Nevertheless, NDS improve the usability of this system and thus ensure its profitability.

5 Conclusion and Future Work

We have presented a novel distance-based decision-making support called the *Network Distance Service*. It is designed for dynamic SOA-based grids and is particularly useful in cross-grid environments where some decisions must be handled from user-side.

The *NDS* provides an uniform and high level access to monitoring information, making the expression of the need at same time easy and accurate. It allows the computation of distances adapted to any task in a wide range of application. Its accuracy and usability have been shown with concrete experiments for classical decision-making problems in a real platform. Our main contribution is to ensure the profitability of monitoring systems by improving the usability of the produced data. NDS can be used either by users through a JAVA GUI or by other services through WSDL API. Moreover NDS can easily solve complex problems thanks to embedded algorithms.

Finally, this work opens numerous perspectives on how integral and derivative of distances can be exploited and how parameters like task CPU cycles can be extracted and included into WSDM. Moreover, a scalable software architecture to deliver monitoring data in cross-grid environment must be studied.

References

1. de Assuncao, M.D., Buyya, R.: A case for the world wide grid. Technical Report GRIDS-TR-2006-1, GRIDS Laboratory, Mel bourne University, Australia (2006)
2. Lowekamp, B., Tierney, B., Cottrell, L., Hughes-Jones, R., Kielmann, T., Swany., M.: A hierarchy of network performance characteristics for grid applications and services. In: Global Grid Forum. (2004)
3. Globus_Alliance: (Monitoring and discovery service) http://www.globus.org/mds/.
4. Cooke, A., Gray, A., et al.: The relational grid monitoring architecture: Mediating information about the grid. (2004)
5. Truong, H.L., Fahringer, T.: Scalea-g: a unified monitoring and performance analysis system for the grid. **12**(4) (2004) 225–237
6. Wolski, R., Spring, N.T., Hayes, J.: The Network Weather Service: a distributed resource performance forecasting service for metacomputing. Future Generation Computer Systems **15**(5–6) (1999) 757–768
7. Francis, P., Jamin, S., Jin, C., Jin, Y., Raz, D., Shavitt, Y., Zhang, L.: IDMaps: A global Internet host distance estimation service. IEEE/ACM Transactions on Networking **9**(5) (2001) 525–540
8. Ng, T.S.E., Zhang, H.: Predicting internet network distance with coordinates-based approaches. Volume 1 of Proceedings IEEE INFOCOM 2002., (IEEE) 170–179
9. Faerman, M., Su, A., Wolski, R., Berman, F.: Adaptive performance prediction for distributed data-intensive applications. In: ACM/IEEE SC99 Conference on High Performance Networking and Computing, Portland, OR, USA (1999)
10. Grid 5000. https://www.grid5000.fr/.
11. JEP - Java Math Expression Parser. Singular Systems. www.singularsys.com/jep/.
12. Gossa, J.: Evaluation of network distances properties by nds, the network distance service. In: 3th International Workshop on Networks for Grid Applications (GRIDNETS'06), IEEE/Create-Net (2006)