Parallelization of Littlewood-Richardson Coefficients Computation and its Integration into the BonjourGrid Meta-Desktop Grid Middleware

Heithem Abbes, Franck Butelle, Christophe Cérin

Parallelization of Littlewood-Richardson Coefficients Computation and its Integration into the BonjourGrid Meta-Desktop Grid Middleware

Heithem Abbes Research Unit UTIC, ESSTT, University of Tunis, Tunisia
Franck Butelle LIPN/UMR 7030 — CNRS/Université Paris 13, France
Christophe Cérin LIPN/UMR 7030 — CNRS/Université Paris 13, France

Abstract

The aim of this paper is to show how to parallelize a compute intensive application in mathematics (Group Theory) for an institutional Desktop Grid platform coordinated by a meta-grid middleware named BonjourGrid. The paper is twofold: first of all, it shows how to parallelize a sequential program for a multicore CPU which participates in the computation and second it demonstrates the effort for launching multiple instances of the solutions for the mathematical problem with the BonjourGrid middleware. BonjourGrid is a fully decentralized Desktop Grid middleware. The main results of the paper are: a) an efficient multi-threaded version of a sequential program to compute Littlewood-Richardson coefficients, namely the Multi-LR program and b) a proof of concept, centered around the user needs, for the BonjourGrid middleware dedicated to coordinate multiple instances of programsfor Desktop Grids and with the help of Multi-LR. In this paper, the scientific work consists in starting from a model for the solution of a compute intensive problem in mathematics, to incorporate the concrete model into a middleware and running it on commodity PCs platform managed by an innovative meta Desktop Grid middleware.

Keywords Incorporation of a threaded application with Desktop Grid infrastructures, Desktop Grid models, Modeling, Simulation, Emulation of large-scale environments, Resource management and scheduling.

1 Introduction and motivations

Desktop Grid (DG) have been successfully used to address large applications with significant computational requirements, including search for extraterrestrial intelligence (SETI@Home [22]), global climate predication (Climatprediction.net [21]), and cosmic rays study (XtremWeb [10, 16]). While the success of these applications demonstrates the potential of Desktop Grid, existing systems are often centralized and suffer from relying always on an administrative staff who guarantees the continuous running of the master. Moreover, although, in practical, the crash of the master is not frequent and replication techniques can solve this problem when it occurs, we still believe in the need to decentralized approaches. We justify this by the fact that since all institutions have not the same financial means to guarantee high levels of equipments robustness, the community should offer solutions to institutions which have not sufficient facilities to avoid the central element crash for instance. Thus, we believe in the importance of collaborative and decentralized solutions for the setting of Desktop Grid middleware.

In this context, Abbes, Cérin and Jemni have proposed a novel approach, called BonjourGrid [7, 8, 8], which enables to establish a specific execution environment for each user. Indeed, BonjourGrid constructs, dynamically and in a decentralized way, a Computing Element (CE – a CE is a set of workers managed by one master, or an instance of a local Desktop Grid middleware) when a user needs to run an application. BonjourGrid orchestrates multiple instances of CEs in a decentralized manner. Our approach does not rely on a unique static central element in the whole system, since we dedicate a temporary central element for each running application, in a dynamic way.

Furthermore, it is important to note that BonjourGrid is not only a decentralized approach to orchestrate and coordinate local DG (Desktop Grid) but it is also a system which is able, contrarily to classical DG, to construct specific execution environment on-demand (based on any combination of XtremWeb, Boinc, Condor middleware), in a decentralized, dynamic and autonomous way. This is the originality of the BonjourGrid system. We consider that it is a novel approach, a step forward regarding Desktop Grid systems.

The application that we investigate in this paper to explain our approach is a compute intensive applicationin the field of Group Theory. To our knowledge, there has been no attempt in the past to derive a multi-threaded solution of the computation of Littlewood–Richardson coefficients for desktop grid, especially for those based on multi-core processors. We give an original and efficient solution to this difficult problem because it is hard to predict on the fly the space where the solution is located in as we will see later in the paper.

Therefore, the paper is mainly organized according to two central discussions: first of all the threading of the computation of LR coefficients and second the integration of the solution into BonjourGrid. The typical use case underlying our work is the following: a mathematician needs to guess the property of some numbers. Usually it uses the PCs in its institution to ’compute’ the properties of the objects with the sequential implementation of Schur1 while colleagues work on others problems. He requires also that it could be done with a minimal effort for him (minimum of deployment, minimum of line codes to launch the application) because he is not an expert in grid middleware. This user belongs to the community of Boinc users, so he wants to use this middleware whereas one of his colleague belongs to the Condor community and he wants to use Condor. Our solution allows any user to run the multi-threaded release of the Schur application that we introduce in this paper on its favorite middleware.

We do not fully describe the adaptation of the application on the BonjourGrid infrastructure because it is out of the scope and will require too much space. BonjourGrid is a large project consisting in coordinating, simultaneously major desktop grid middleware such as Condor, Boinc and XtremWeb instances.

This idea, while closely related to the concept of "Infrastructure on demand" concept central to the Cloud Computing paradigm and thus not completely new, may help in widening the spread of the Desktop Grid approach. The article strives to demonstrate the feasibility of the solution using the application from mathematics namely the computation of LR coefficients because we still need to validate BonjourGrid through ’real applications’ and not emulation as we have already done. We choose the LR coefficients application because it is used by people in our laboratory, and because it is a central piece in the Schur package that is a tool to ’guess’ properties of polynomials for instance.

Again, the main result of this paper is not about a new Desktop Grid middleware because in this case we need also to address, for the sake of completeness, the problems of volatility (host churn) and heterogeneity but it is about a multithreaded code for the computation of LR coefficients. The ultimate goal is not to beat records in speed but to demonstrate that BonjourGrid is able to coordinate multiple instances of the same application on multiple instances of an innovative grid middleware deployed over an institution and also that a non experimented user may use this new environment.

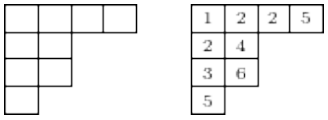
2 A computation problem in Group Theory

2.1 Mathematical background

In order to be self content we need some standard definitions and vocabulary from group theory [2]. A **partition** of a positive integer n is a way of writing n as a sum of non increasing strictly positive integers. For example $\lambda = (4,2,2,1)$ and $\mu = (2,1)$ are partitions of $n = 9$ and $n' = 3$ respectively. We write $\lambda \vdash n$ and $\mu \vdash n'$ or $|\lambda| = n$ and $|\mu| = n'$.

The **Ferrers diagram** F^λ associated to a partition $\lambda = (\lambda_1,\lambda_2,\ldots,\lambda_p)$ consists of $|\lambda| = n$ boxes, arranged in $l(\lambda) = p$ left-justified rows of lengths $\lambda_1, \lambda_2, \ldots, \lambda_p$. Rows in F^λ are oriented downwards. F^λ is called the shape of λ.

A **semi-standard Young tableau** of shape F^λ (*SSY* T^λ) is a numbering of the boxes of F^λ with entries from $\{1,2,\ldots,n\}$, weakly increasing across rows and strictly increasing down columns. A tableau is **standard** (*SY* T^λ) if all its entries are different. Skew tableaux are defined in an analogous way. For example, for $\lambda = (4,2,2,1)$ and $\mu = (2,1)$, F^λ and *SSY* T^λ are as follows:



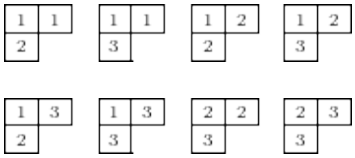
A **Symmetric Function** is a function which is symmetric or invariant under permutation of its variables.

$$f(x_{\sigma(1)}, x_{\sigma(2)}, \cdots, x_{\sigma(n)}) = f(x_1, x_2, \cdots, x_n)$$

where σ is any permutation of the symmetric group S_n. The **Schur function** s_λ is the symmetric function defined as:

$$s_\lambda(x) = \sum_{T \in \mathrm{Tab}(\lambda)} x^T$$

where Tab(λ) is the set of all tableaux of shape λ and x^T is the product of the x_i for all i appearing in the tableau T. For example, for $\mu = (2,1)$ there are 8 semi-standard tableaux of shape μ using $\{1,2,3\}$:



The associated Schur function is therefore:

$$\begin{aligned} s_{21}(x_1,x_2,x_3) &= x_1^2x_2 + x_1^2x_3 + x_1x_2^2 + x_1x_2x_3 \\ &\quad + x_1x_2x_3 + x_1x_3^2 + x_2^2x_3 + x_2x_3^2 \end{aligned}$$

The **Littlewood-Richardson coefficients** $c_{\lambda\mu}{}^\nu$ are defined as the structure constants for the multiplication in the basis of the Schur functions. So if $\lambda \vdash n$ and $\mu \vdash m$:

$$s_\lambda s_\mu = \sum_{\nu \vdash n+m} c^\nu_{\lambda\mu} s_\nu$$

Example:

$$\begin{aligned} s_{4221}s_{21} &= s_{6321} + s_{6222} + s_{62211} + s_{5421} + s_{5331} \\ &\quad + 2s_{5322} + 2s_{53211} + s_{4431} + s_{4422} \\ &\quad + s_{44211} + 2s_{52221} + s_{4332} + 2s_{43221} \\ &\quad + s_{522111} + s_{43311} + s_{432111} + s_{42222} \\ &\quad + s_{422211} \end{aligned}$$

Thus $c_{4221,21}{}^{5322} = c_{4221,21}{}^{43221} = 2$ and

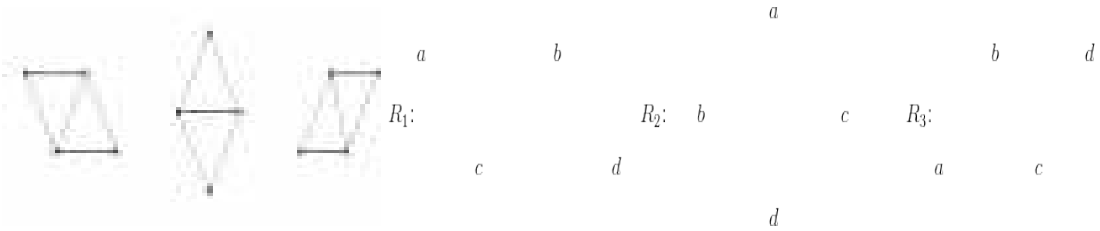
$c_{4221,21}{}^{6321} = c_{4221,21}{}^{422211} = 1$.

2.2 The hives model

Littlewood-Richardson coefficients $c_{\lambda\mu}^{\ \nu}$ have a polynomial growth with respect to the dilatation factor $N \in \mathbb{N}$:

$$c_{N\lambda,N\mu}^{N\nu} = P_{\lambda\mu}^{\nu}(N) \quad ; \quad P_{\lambda\mu}^{\nu}(0) = 1$$

where $P_{\lambda\mu}^{\ \nu}$ is a polynomial in N with non negative rational coefficients depending on λ, μ and ν. This was first conjectured in [3]. A partial proof (existence of $P_{\lambda\mu}^{\ \nu}$ and rationality of coefficients) was given in [1]. Those polynomials [3] are obtained considering a model known as the hive model. An ***n*-integer-hive** is a triangular array of non negative integers a_j^i with $0 \leq i,j \leq n$ where neighboring entries define three distinct types of rhombus, each with its own constraint condition.



In each case, with the labeling as shown, the hive condition takes the form:

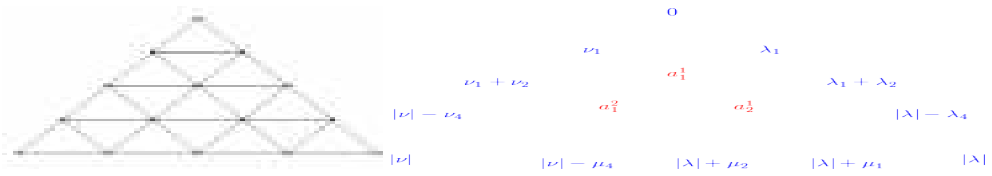
$$b + c \geq a + d \quad (1)$$

A ***LR*-hives** [3] is an integer hive satisfying the hive condition (1) for all its constituent rhombic of type R_1, R_2 and R_3, with border labels determined by the partitions λ, μ and ν, such that:

$$\begin{cases} a_0^0 = 0 \ ; \ a_j^j = \lambda_1 + \lambda_2 + \cdots + \lambda_j \\ \qquad\qquad\qquad\qquad \forall j \in \{1, \cdots, n\} \\ a_0^i = \nu_1 + \nu_2 + \cdots + \nu_i \\ \qquad\qquad\qquad\qquad \forall i \in \{1, \cdots, n\} \\ a_{n-k}^k = a_n^0 + \mu_1 + \mu_2 + \cdots + \mu_k \\ \qquad\qquad\qquad\qquad \forall k \in \{1, \cdots, n\} \end{cases}$$

with $l(\lambda)$, $l(\mu)$, $l(\nu) \leq n$ and $|\lambda| + |\mu| = |\nu|$.

For example, for $n = 4$, *LR*-hives will be sketched as:



The Littlewood-Richardson coefficient $c_{\lambda\mu}^{\ \nu}$ is the number of *LR*-hives with border labeled as above by λ, μ and ν. So $c_{\lambda\mu}^{\ \nu}$ is the number of triples $(a_1^{\ 1}, a_1^{\ 2}, a_2^{\ 1})$ satisfying (1) for all rhombic of type R_1, R_2 and R_3. That is the number of integer solutions to the system of inequalities obtained by writing (1) for all rhombic. The degree of the polynomial $P_{\lambda\mu}^{\ \nu}(N)$ is always bounded by $(n-1)(n-2)/2$, the number of

interior points of the hive, but if any of the partitions, λ, μ or v has repeated parts then it is possible to use the hive inequalities to improve this bound with some $maxDeg < (n-1)(n-2)/2$. We compute $c_{N\lambda,N\mu}{}^{Nv}$ for any $N \in \mathbb{N}$ with N increasing, and compute the interpolating polynomial for the points $(N, c_{N\lambda,N\mu}{}^{Nv})$. The computation is stopped when two successive interpolating polynomials are equal. As stated by theory, actually the number of computed coefficients is often less than $(n-1)(n-2)/2$.

2.3 A parallel computation

The aim of this section is to explain the different ways we compute the LR coefficients in parallel. Two kinds of parallelization is under concern in this article which are the parallelization on a single machine (multicore CPU) and the parallelization at the grid level. The goal is to replace the current sequential code provided with Schur2 by a multi-threaded one able to be integrated into a DG middleware. Schur is recognized to be a powerful and efficient tool for calculating properties of Lie groups and symmetric functions. Developing a parallel version of Schur is challenging and we restrict our study to the computation of LR coefficients in parallel.

2.3.1 Parallelization on a single machine: the Multi-LR code

The ways we are computing the Littlewood-Richardson coefficients allows us to split the enumeration process of the feasible space in several parts. To do this, we use the Pthread API, and divide the enumeration process in two pieces:

1. We assume that only n threads are allowed for the process. The first step of the enumeration is to divide the feasible space in n parts. Since the enumeration space is made of a product of intervals, we divide the first interval in $min(nb\_thread, interval\_size)$ of the same size, and for each part we attach a thread.
2. Each thread uses an optimized method to explore the feasible space and returns the number of solutions. Once the enumeration is done for all the threads, the main thread sends the result to the user.

2.3.2 Parallelization using a Desktop Grid

The computation of LR polynomials can be achieved using the multiple machine on a local network. The idea is to compute the Littlewood-Richardson coefficients on various machines for a set of dilatation coefficients and to collect the results to build the interpolating polynomial. The time needed to compute one LR coefficient depends on the partitions given to the program and on the dilatation coefficient.

The four input parameters (λ,μ,v,N) are given to the program, where $N \in \mathbb{N}$ is the dilatation coefficient. For each computer on the network, a unique dilatation coefficient is given. The result of each computation is the Littlewood-Richardson coefficient $c_{N\lambda,N\mu}{}^{Nv}$. Thus many interpolation points $(N, c_{N\lambda,N\mu}{}^{Nv})$ are computed. Each time a new interpolation point is produced, a new interpolating polynomial is computed using all the values available. Thus, a sequence of polynomials is built:

$$P_{1\lambda\mu}{}^{v}(N), P_{2\lambda\mu}{}^{v}(N), \ldots, P_{i\lambda\mu}{}^{v}(N), \ldots$$

The computation is stopped when the sequence becomes stationary. That means that $P_{i\lambda\mu}{}^{v} = P_{(i+1)\lambda\mu}{}^{v}$ for some i and it holds for any j greater than i.

The complexity of the algorithm used to compute each Littlewood-Richardson coefficient is

$$C(N) = \left(\prod_{k=1}^{m} |I_k| \right) N^m$$

where m is the number of intervals in the hive, $|I_k|$ is the number of elements in the interval I_k and N is the dilatation coefficient. Thus, the time needed to compute a Littlewood-Richardson coefficient has a polynomial growth of order m (see figure 1). This means that for high values of m, the time needed to compute the coefficient $N+1$ will be greater than the time needed to compute all the coefficients up to N. In other words, for large values of m, the time needed for the whole computation (all coefficients) is more or less the time needed to compute the last one.

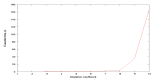


Figure 1: Computation time with respect to the dilatation coefficient. The partitions are $\lambda = (5,3,2,2,1)$, $\mu = (4,3,2,2,1)$ and $\nu = (7,5,4,4,3,2)$.

2.4 Experimental Results

2.4.1 Performance gain on a single machine

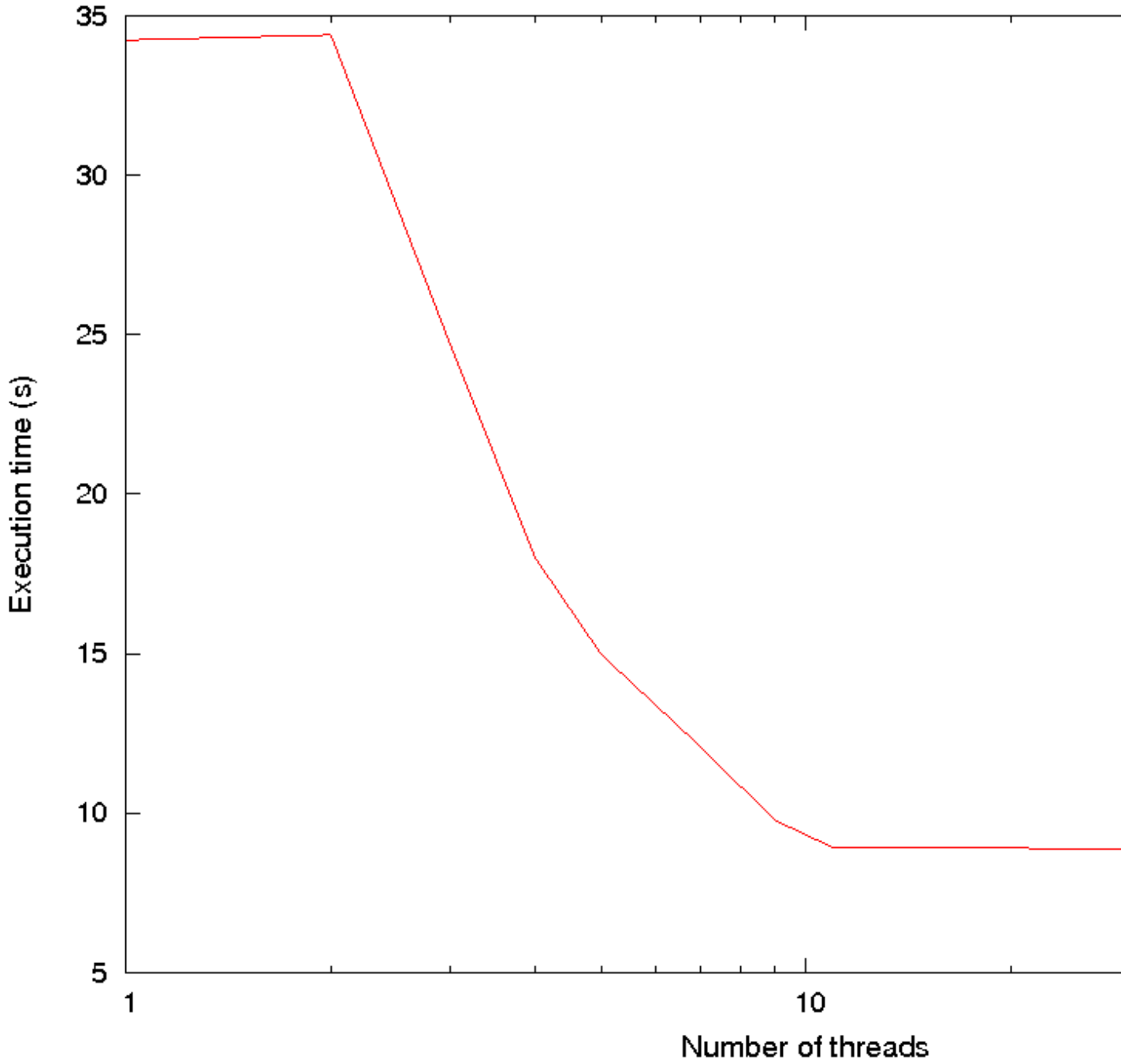
The use of threads to compute the Littlewood-Richardson coefficients leads to improvement in the execution time. To compare the gain in performance, we use a multi-core processor. In the remainder of the section, computation are done on a Bi-AMD Opteron dual core at 2.8GHz, which means that 4 cores were available for the computation.

The computation of Littlewood-Richardson coefficients was done for various partitions and variable maximum number of threads. The first remark is that the gain in time depends on the first interval (which is divided to create the threads) and on the symmetry of the problem. For example if the first interval is a single value, only one thread is launched, and the time needed for the computation is the same than without threads.

The "symmetry" of the feasible space is also important for the enumeration. To understand this, the algorithm has to be explained again. As said before, the enumeration method used by each thread is optimized, so that the whole feasible space is not searched. Each step of the algorithm corresponds to one point of the hive. For each feasible value of a point, the feasible space is computed again so that useless values are not tried. Therefore, the number of feasible values tried by each thread depends on the interval it received from the first step, and some threads are faster than other. This phenomenon can be observed on Figure 2, which shows the results for the partitions $(\lambda = (5,5,3,2,1,1)$, $\mu = (6,6,4,2,1)$ and $\nu = (6,6,6,5,5,3,2,2,1))$ and $(\lambda = (7,6,5,4)$, $\mu = (7,7,7,4)$, $\nu = (12,8,8,7,6,4,2))$. The computation time for the dilatation coefficient $N = 7$ does not decrease when the number of threads increases.

However, the majority of the practical cases shows that threading the program is useful. The computation time is divided by a factor of 3.5, and nearly four in thebest cases, depending on the "symmetry" of the problem. In fact, we could think that the results should be similar for any number of threads greater than 4. But as we can see on the first diagram of figure 2, the time needed for the computation is stationary for a number of threads greater than 10. This is explained by the fact that for less than 10 threads, one of the threads is slower than the others because the number of values it deals with is bigger than for the other threads.

For a very large number of threads, the performance mainly depends on the operating system’s scheduler. We are using the Linux scheduler, and even with a great number of threads (over 200 threads), the results are impressive.


35
30
25
20
15
10
5
Execution time (s)
1
10
Number of threads

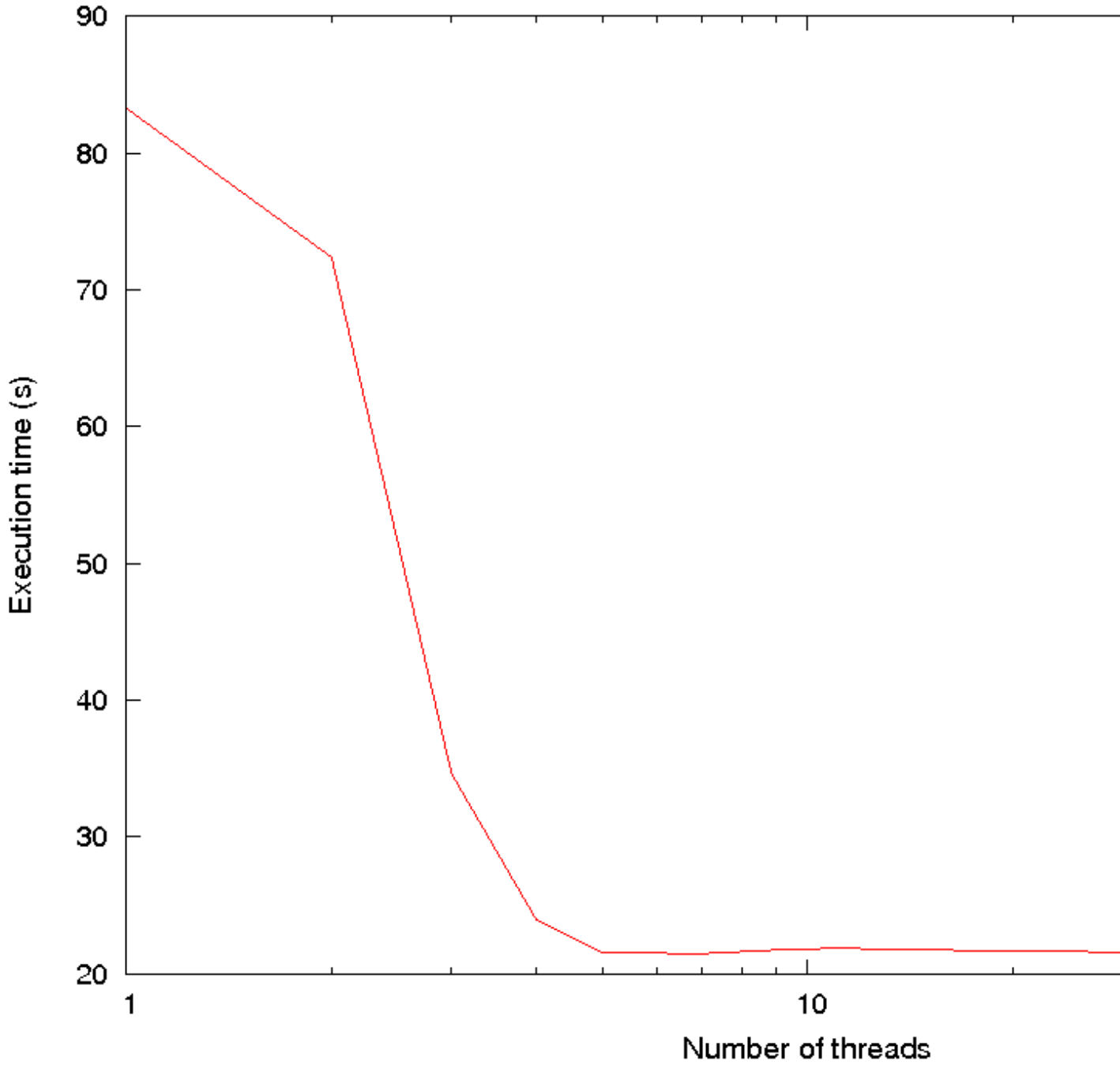


Figure 2: Evolution of the computation time needed with the number of threads allowed for partitions.

---

3 Porting the parallel code on a Desktop Grid platform

3.1 Introduction to BonjourGrid

BonjourGrid is a meta Desktop Grid middleware meaning that it is able to instantiate multiple Desktop Grid middleware in the same infrastructure. As referenced in many other works [12, 13], this kind of environment is called Institutional Desktop Grid or Enterprise Desktop Grid (e.g located in the same institution). The principle of the proposed approach is to create, dynamically and in a decentralized way, a specific execution environment for each user to execute any type of applications without any system administrator intervention. An environment do not affect another one if it fails.

This section aims at demonstrating the potential of BonjourGrid and it serves as a proof of concept. The

BonjourGrid protocol has been validated in [7, 8, 9] but the goal was to show that BonjourGrid is able to behave "like" a job scheduler. Here we are deploying a real application on top of BonjourGrid. Moreover, in this section we do not evaluate the application performance on the Grid, but we explain, from a user point of view, what is the work to accomplish to run a code on top of the middleware.

Details about BonjourGrid follow and the software is available on sourceForge3 . Each user, behind its desktop machine in his office, can submit an application. BonjourGrid deploys a master (coordinator), locally on the user machine, and requests for participants (workers). Negotiations to select them should now take place. Using a publish/subscribe infrastructure, each machine publishes its state (idle, worker or master) when changes occur as well as information about its local load or its “utilization cost”, in order to provide useful metrics for the selection of participants by the BonjourGrid middleware. Under these assumptions, the master can select a subset of workers nodes according to a strategy that could balance the “power” of the node and the “price” of its use. The master and the set of selected workers build the Computing Element (CE) which will execute, manage and control the user application. When a CE finishes the application, its master becomes free, returns in idle state and releases all workers to return also to the idle state. When no application is submitted, all machines are in the idle state.

The main technologies used in BonjourGrid are publish/subscribe systems [17] and ZeroConf protocol [4]. In a previous work [5], we tackled the following questions: Can a system based on publish/subscribe protocols be “powerful”? Can it be scalable? What is the response time to publish a service? What is the discovery time of a service? We carried out experiments on various protocols such as Bonjour protocol, on the GdX node of Grid5000 [20] platform using more than 300 machines (AMD Opterons connected with a 1Gb/s network). Measures show that Bonjour is reliable and very powerful in resources discovery. Indeed, Bonjour discovers more than 300 services published simultaneously in less than 2 seconds with 0% of loss. These evaluations support and justify our choice to use Bonjour as a basis for the discovery protocol in BonjourGrid system. Moreover, we believe that using an existing and tested protocol for the industry (Bonjour is available on Macintosh machines), to evaluate its usability in Desktop Grid environment is, in itself, of great value and very useful for the community.

The advantages in using a publication/subscription (Pub-Sub) [14] mechanisms come from the asynchronous nature of the paradigm. The Pub-Sub paradigm is an asynchronous mode for communicating between entities. Some users, namely the subscribers or clients or consumers, express and record their interests under the form or subscriptions, and are notified later by another event produced by other users, namely the producers).

This communication mode is thus multipoint, anonymous and implicit. It is a multipoint mode (one-to-many or many-to-many) because events are sent to the set of clients that have declared an interest into the topic. It is an anonymous mode because the provider does not know the identity of clients. It is an implicit mode because the clients are determined by the subscriptions and not explicitly by the providers.

It is also known that this asynchronous communicating mode allows spatial deco upling (the interacting entities do not know each other), and time decoupling (the interacting entities do not need to participate at the same time). This total decoupling between the production and the consumption of services increases the scalability by eliminating many sorts of explicit dependencies between participating entities. Eliminating dependencies reduces the coordination needs and consequently the synchronizations between entities. These advantages make the communicating infrastructure well suited to the management of distributed systems and simplify the development of a middleware for the coordination of DGs.

The key idea of BonjourGrid is to rely on existing Institutional Desktop Grid middleware, and to orchestrate and coordinate multiple instances, i.e multiple CEs, through a publish/subscribe system (see figure 3). Each CE will be owned by the user who has started the master on his machine. Then this CE is responsible for the execution of one or many applications for the same user. As shown in figure 3, in the user level, a user A (resp. B) deploys his application on his machine and the execution seems to be local. Level 1 (middleware layer) shows that, actually, a CE with 4 (resp. 5) workers has been dynamically created, specifically for the user A (resp. B). Level 0 shows that all machines are interconnected and under the availability of any user.

Figure 3: BonjourGrid abstract layers

3.2 Implementation

BonjourGrid is entirely implemented using Python. We have used Bonjour-py package that offers a python interface to interact with Bonjour protocol. When a machine joins our system, information about its characteristics (i.e. MHZ, CPU, RAM, Hostname, IP address, average load) are automatically collected and stored in a Python dictionary. These informations are useful for the selection of the suitable machines that match the application requirements. For instance, one would like to select workers according to the CPU metric only; another one would like to select workers that have been being idle for a long time. In our system, we can easily plug a new policy in order to select workers. We can also imagine, in the future, to make an economic model based choice of the suitable policy; the coding effort is not important, thanks to the Python abstractions.

3.2.1 Running the XtremWeb services

In this paper, we choose to use XW (XtremWeb) as a middleware for the CE. To run or deactivate an XW service (coordinator or worker), the environment should be already installed. The procedure to install a XW-Coordinator, in particular, is not currently simple enough, so we have improved it. We would not like to make user spends his time in configuring and installing the various files and modules necessary to the XW installation. An installation procedure consists in installing a MySQL server, Java Development Kit, creating a specific database for XW, making several directories and configuring system files. Consequently, we set up an automatic installation of all the necessary packages. Such facilities were not included in the current distribution of XW.

3.3 Experiments and validation of BonjourGrid

From a user point of view, deploying an application on BonjourGrid starts with preparing the executable code and data that are archived in a compressed file. BonjourGrid enables the execution of applications with precedences between tasks. Indeed, the user can describe the precedences of a data flow graph using a XML description; this is not trivial, especially, for complex applications.

SDAD is the system of deployment that we have developed and it helps the user in describing the data flow graph of his application according to an XML syntax using a graphical interface (for simple applications, see figure 9) or an advanced wizard (for complex ones, see figure 10). It also helps user to put the different files of his applications (i.e. binary and data files) in the right path of his package. SDAD generates a compressed package ready to be deployed on BonjourGrid system. Now, the user can submit his application to BonjourGrid. He can specify the size of the computing element (CE) and the middleware (XtremWeb in this case study) to run the application. BonjourGrid, then, will construct the CE according to the criteria mentioned in the XML file.

To summarize, using the SDAD tool, the user can draw the task graph and put binary and data files in the suitable path in the application tree. Thereafter, SDAD will generate the XML description of the application which is used by XW and BonjourGrid.

As described in subsection 2.3, the first module of the application Hives can be divided in several parallel tasks. The outputs of these tasks are forwarded to the so called BuildInter module. Finally, the module Interp gives the interpolation. Figure 4 shows the data flow graph of the Hives application.

Figure 5 shows the first part of this file which illustrates the description of the three modules of the application (i.e, OS architecture, OS type, location of binary files…). Here, the user can provide several binary files for different architectures and OS types. The precedences between tasks are illustrated in the second part of the XML file shown in Figure 6. On Figure 6 we have simplified the XML file to make it easier to understand (we drawn only two parallel tasks for the Hives module).

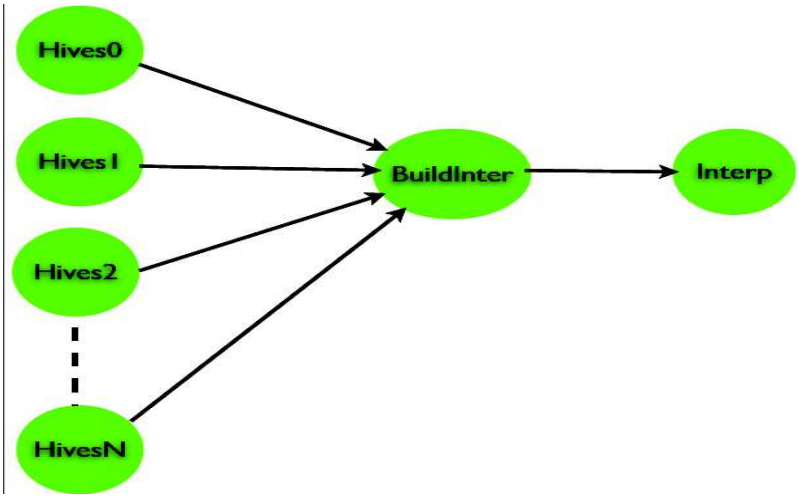


Figure 4: Description of the data flow graph of Hives composed with 3 modules; Hives, BuildInter and Interp

Figure 5: Description of the flow data graph of the application Hive

Figure 6: Description of the precedences between tasks in Hive application

Now, Hives is ready to be submitted to the BonjourGrid system. In the following, we illustrate snapshots picked out from a Hives execution using the Orsay node of Grid5000 [20]. Specifically for this demo, we are going to dissociate the CE building phase from the effective submission to illustrate the different steps. First, we initiate machines in idle state. We launch the coordinator, on any machine, to start the building phase of a suitable CE. Figure 7 shows the outputs of a construction of a CE with 2 workers (just for the sake of clarity of reading). Indeed, the coordinator is started on the `gdx-5` node, requiring two workers as shown in the figure 7. For that, the coordinator gdx-5 searches for idle machines which match the tasks requirements. Figure 7 shows that the coordinator gdx-5 discovers in this test two idle machines `gdx-9` and `gdx-17` and asks them to accept to work for it. Thereafter, the `gdx-5` coordinator receives two confirmations from `gdx-17` and `gdx-9` as depicted in figure 8. When the execution completes, we can invoke BonjourGrid to download the results.

Figure 7: Construction of a new CE with 2 workers

Figure 8: Confirmation of two idle machines to work for the master gdx-5

To summarize, the objective of this experiment was to present a use case of BonjourGrid using a real application and from the user point of view. Abbes, Cérin and Jemni have already done experiments to analyze the performance of BonjourGrid in [7, 8] and [5] but the scope of this work is to demonstrate how BonjourGrid can help users to construct, dynamically and without any intervention of a system administrator, their own environments to deploy and perform out a parallel application.

4 Related work on advanced Desktop Grid Architectures

Before concluding, we compare BonjourGrid with others systems. OurGrid [15] system avoids the centralized server by creating the notion of the home machine from which applications are submitted; the existence of several home machines reduces the impact of failures at the same time. Moreover, OurGrid provides an accounting model to assure a fair resources sharing in order to attract nodes to join the system. However, the originality of BonjourGrid comparing to OurGrid is that it supports distributed applications with precedence between tasks as well as Bag-of-Tasks (BOT), while OurGrid supports only BOT applications (BOT applications are independent divisible tasks). WaveGrid [6] is a P2P middleware which uses a timezone-aware overlay network to indicate when hosts have a large block of idle time. This system reinforces the idea of BonjourGrid concept since changing from a set of workers to another one depending on the time zone (Wave Grid) is analogous to the principle of creating a CE from an application to another one in BonjourGrid, and depending on users requirements.

Approaches based on publish/subscribe systems to coordinate or decentralise Desktop Grid infrastructures are not very numerous according to our knowledge. A similar project to our project is the Xgrid project [19]. In Xgrid system, each agent (or worker) makes itself available to a single controller. It receives computational tasks and returns the results of these computations to the controller. Hence, the whole architecture relies on asingle and static component which is the controller. Moreover, Xgrid runs only on MacOS systems. In contrast with Xgrid, in BonjourGrid, the coordinator is not static and is created in a dynamic way. Furthermore, BonjourGrid is more generic since it is possible to “plug” on it any computing system (XW, Boinc, Condor) while Xgrid has its own computing system. The key advantage of bonjourGrid is that the user is free to use his favorite desktop grid middleware.

5 Conclusion and future works

In this work, we have proposed a novel algorithm to compute Littlewood-Richardson coefficients (by the Hive method) in parallel and we have shown how to run the code on top of BonjourGrid. The aim of BonjourGrid is to orchestrate multi-instances of computing elements, in a decentralized manner. BonjourGrid creates, dynamically, a specific environment for the user to run his application. There is no need for a system administrator. Indeed, BonjourGrid is fully autonomous and decentralized. We have conducted several experimentations to show that BonjourGrid operates well with a real world application. Therefore BonjourGrid has shown its usefulness. Moreover, the deployment of the Hive code on BonjourGrid, demonstrates that BonjourGrid may help users to create several independent environments. Then, mathematicians, for instance, can create their Desktop Grid easily to run their parallel applications.

Several issues must be taken into account in our future work about meta-Desktop Grid middleware. The first issue is to build a fault-tolerant system for the coordinators. In fact, it is important to continue the execution of the application when the coordinator (user machine) fails (it is disconnected for instance). This issue has been solved and it is currently tested overand validated on Grid’5000 testbed. Condor, Boinc plugins have also been built and are currently tested with the Hive code, especially but not only. The second issue is the reservation of participation: in the current version, BonjourGrid allocates

available resources for a user without any reservation rules. Thus, BonjourGrid may allocate to a single user all the available resources. The third issue is to pass to a wide area network.

In order to be widely adopted by the community, we are currently working also on the following issues. First, BonjourGrid is still working on top of Bonjour which is a local-area network. The wide area-Bonjour4 is available but require some work to configure it. The goal is to deploy our protocol over the Internet and not over a local network. After installing Wide Area Bonjour, the experimental work could not really be about performance because they cannot be reproducible over Internet. The setting of a large platform of volunteers is more challenging and the experimental work would be to show that the system can handle large configurations of PC.

Another choice is to implement our protocol over another Pub-Sub tool. The target language may take the form of a popular language on the Internet such as XMPP5 . In fact, many libraries support the publish-subscribe mechanism are being deployed (e.g. Jetty), including the forthcoming HTML5 standard. The danger is reimplementing the protocol is to forget some implementation details that make the protocol correct. We are currently working on the specification of our coordination protocol by using Petri Net tools in order to isolate the main properties of our protocol but also the main knowhow in using Pub-sub systems. Petri Net is a formal tool to reasoning about the specification in order to check fundamental properties of the design such as safety ("nothing bad happens") for which the absence of deadlock is an example and liveness ("something good eventualy happens").

Any grid middleware should also support a security level. Even if you are sure that communications will not be decrypted you must ensure the authentication of the participants. Conventionally authentication in grids is via X509 certificates. In the case of BonjourGrid, we also need an orchestration of certificates to ensure for example that two Boinc coordinators can exchange participants. Who says X509 certificates also said VO (Virtual Organizations) that allow the federation of resources. The problem will be to manage various VOs.

We are also working on the adaptation of BonjourGrid principles for data management. The main challenge is to resume the protocol to coordinate data servers by introducing a new state in the Protocol to mean that a node can become a data server. This new state impacts the protocol with respect to fault tolerance. It will be necessary to revise the protocol to manage data backup and servers backup.

.... Faire aussi des perspectives sur le calcul des coeff... c.a.d des perspectives moins en rapport avec des problematiques architectural.

We hope that this work helps to understand what is a possible future for Desktop Grid middleware as well as the new efforts that users should make to use such systems.

References

[1] E. Rassart. *A polynomiality property for Littlewood-Richardson coefficients*, In J. Combinatorial Theory, Ser A 107, 2004, pp 161-179.

[2] I.G. MacDonald. *Symmetric functions a nd Hall Polynomials*, In 2nd ed., Clarendon Press, Oxford Science Publication, 1995.

[3] R. C. King, C. Tollu and F. Toumazet *Stretched Littlewood-Richardson and Kostka coefficients*, In CRM Proceedings and Lecture Notes, Vol. 34, Amer. Math. Soc, 2004, pp99-112.

[4] D. Steinberg and S. Cheshire. Zero Configuration Networking: The Definitive Guide. *O'Reilly Media, Inc*, 2005.

[5] Heithem Abbes and Jean-Chistophe Dubacq *Analysis of Peer-to-Peer Protocols Performance for Establishing a Decentralized Desktop Grid Middleware*, In EuroPar 2008/SGS Workshop, LNCS, 2008.

[6] D. Zhou and V. Lo. *WaveGrid: a scalable fast-turnaround heterogeneous peer-based Desktop Grid system*, In IPDPS'20, IEEE Computer Society, 2006.

[7] Heithem Abbes, Christophe Cérin and Mohamed Jemni. *BonjourGrid as a Decentralised Job Scheduler*, APSCC '08: Proceedings of the 2008 IEEE Asia-Pacific Services Computing Conference, 2008, IEEE Computer Society, isbn: 978-0-7695-3473-2

[8] Heithem Abbes, Christophe Cérin and Mohamed Jemni, *BonjourGrid: Orchestration of Multi-instances of Grid Middleware on Institutional Desktop Grid*, 3rd Workshop on Desktop Grid and Volunteer Computing Systems (PCGrid 2009), in conjunction with IPDPS 2009, May 29, Roma, Italy

[9] Heithem Abbes, Christophe Cérin, Mohamed Jemni and Walid Saad, *Fault tolerance based on the Publish-subscribe Paradigm for the BonjourGrid Middleware*, The 11th ACM/IEEE International Conference on Grid Computing (GRID 2010), Bruxelles, Oct 2010.

[10] G. Fedak, C. Germain, V. Néri, and F. Cappello. *Xtremweb: A generic global computing system*, In Proceedings of IEEE Int. Symp. on Cluster Computing and the Grid, 2001.

[11] D. Thain and M. Livny. Condor and the grid. In France Berman, Anthony J.G. Hey, and Geoffrey Fox, Ed., *Grid Computing: Making The Global Infrastructure a Reality*. John Wiley, 2003.

[12] P. Domingues, A. Andrzejak and L. M. Silva. *Using Checkpointing to Enhance Turnaround Time on Institutional Desktop Grids.* In *e-Science, pp 73*, 2006.

[13] D. Kondo, A. A. Chien and H. Casanova. *Resource Management for Rapid Application Turnaround on Enterprise Desktop Grids*, In SC'04, IEEE Computer Society, pp 17, 2004.

[14] Patrick Th. Eugster, Pascal A. Felber, Rachid Guerraoui, and Anne-Marie Kermarrec. The many faces of publish/subscribe. ACM Comput. Surv., 35(2): 114-131, 2003.

[15] W. Cirne, F. Brasileiro, N. Andrade, L. Costa, A. Andrade, R. Novaes and M. Mowbray. *Labs of the World, Unite!!!* In Journal of Grid Computing, Springer, New York, USA, pp 225-246, 2006.

[16] F. Cappello, S. Djilali, G. Fedak, T. Herault, F. Magniette, V. Né ri and O. Lodygensky. *Computing on Large Scale Distributed Systems: XtremWeb Architecture,Programming Models, Security, Tests and Convergence with Grid*. In FGCS, volume 21, issue 3, pp. 417-437, March 2005.

[17] P. T. Eugster, P.A. Felber, R. Guerraoui, and A. Kermarrec. *The many faces of publish/subscribe*, In ACM Computing Surveys, 35(2): 114-131, June 2003.

[18] Bonjour protocol, http://developer.apple.com/networking/bonjour.

[19] Xgrid. http://gcd.udl.cat/upload/recerca/.

[20] Grid'5000. http://www.grid5000.fr.

[21] BOINC. http://boinc.berkeley.edu.

[22] Seti@home. http://setiathome.ssl.berkeley.edu.

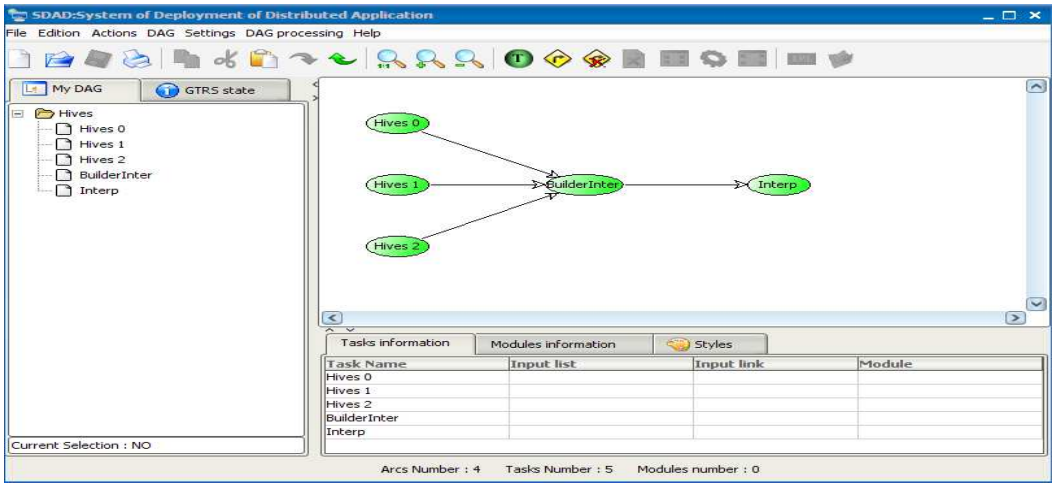


Figure 9: One snapshot of SDAD tool which depicts the graphical interface to draw the Hives data flow graph

---

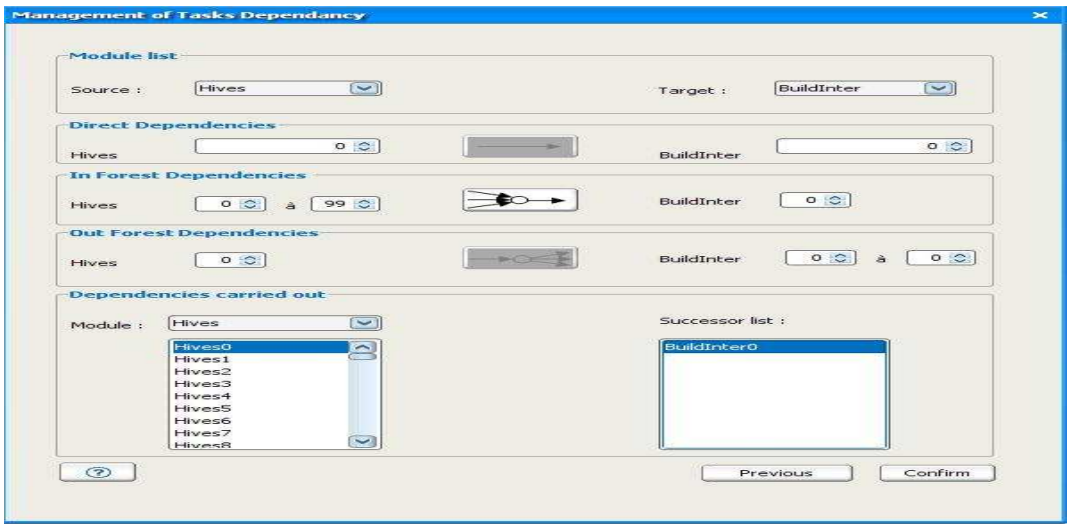


Figure 10: One snapshot of SDAD tool showing the advanced mode where the user needs 100 parallel tasks of the Hives module.
