Towards efficient resource allocation on scientific grids

Vers une allocation efficace des ressources des grilles scientifiques

---

Xavier Gréhant
Isabelle Demeure
Sverre Jarp

2008D001

2008

Département Informatique et Réseaux
Groupe S3 : Systèmes, Logiciels, Services

École Nationale Supérieure des Télécommunications

Towards efficient resource allocation on scientific grids

Vers une allocation efficace des ressources des grilles scientifiques

Xavier Gréhant[12] Isabelle Demeure[2]
Sverre Jarp[1]

[1] CERN openlab, Geneva, Switzerland
[2] ENST, Paris, France

Contents

Abstract

In this report we propose a pragmatic synthesis of the different ways scientific grids carry out computing resource allocation. We analyze the systems in place today, their emergence and their structures. Beyond the different middleware distributions, services, protocols and standards, we draw a simple picture: Application-agnostic infrastructures built for communication, authentication and agreements, provide initial access to computing resources. More complex allocation is managed by independent systems that temporarily infiltrate grid nodes on behalf of applications or a federation of users. We derive the allocation opportunities and constraints that make the case of scientific grids specific among computer systems.

Ce rapport est une synthèse pragmatique des différentes manières d'allouer les ressources de calcul des grilles scientifiques. Nous analysons les systèmes en place aujourd'hui, leur émergence et leurs structures. Au delà des différentes distributions intergicielles, des services, protocoles et standards, nous dressons un tableau simple: les infrastructures généralistes déployées pour la communication, l'authentification et les accords de services fournissent un accès initial au ressources. Une allocation plus fine est réalisée par des systèmes indépendants qui infiltrent temporairement les nœuds d'exécution pour les relier à des applications ou des fédérations d'utilisateurs. Nous dérivons les opportunités et contraintes de ces systèmes qui font des grilles scientifique un cas spécfique dans l'allocation des ressources de calcul.

Chapter 1

Introduction

1.1 Down-to-earth analysis

Allocating resource for scientific computations is the purpose of scientific grids. Literature abounds in presenting what systems should do ideally but lacks a concrete description of the mechanisms actually implemented [Sto07]. In this report we carry an analysis of such mechanisms. We consider the constraints specific to scientific grids, how these constraints shaped resource allocation systems, and the design of allocation algorithms.

The first question that arises is: *In terms of computing resource allocation, what is specific about scientific grids as opposed to other computer systems?*

1.2 Scope

Definition 1. Grids coordinate *resource sharing and problem solving in dynamic, multi-institutional virtual organizations [Fos01].*

Definition 2. ***Virtual Organizations*** *(VOs) enable disparate groups of organizations and/or individuals to share resources in a controlled fashion, so that members may collaborate to achieve a shared goal [Fos01].*

More precisely grids emerged from *metacomputing* [FK97].

Definition 3. ***Metacomputing*** *is the seamless application of geographically-separated distributed computing resources to user applications [Wei98].*

In grids these applications are *large-scale, resource-intensive* [BFH03], and *each task could require computing resources that are distributed geographically and come from several administrative domains.* [CCHJ05].

Definition 4. In the following, we call ***grid*** *a metacomputing system whose resources span independent administrative domains and are used by independent collaborations.*

In scientific grids the *administrative domains* are mostly academia or other public-funded institutions that voluntarily offer a certain amount of their computing resources to scientific projects (the user *collaborations*).

Definition 5. *A **grid site** is a set of grid nodes under the same administrative domain and controlled by the same resource allocation mechanisms and policies.*

In particular grid sites define their own set of user priorities.

Our study excludes a number of related computer systems:

- The term *grid* is often abusively used for **clusters** inside a single organization. It is a simpler case than the one of grids addressed here.
- Our study does not directly take into consideration **general overlay networks** and **testbeds** such as PlanetLab which are not meant to run *computations* but to experiment the deployment of *networked services* [Fiu06]. Such infrastructures can be used to deploy services destined to scientific grids but also any other kind of service.
- **Desktop grids** like SETI@home are a particular case of scientific grids that group together *single nodes* belonging to *individuals*, instead of *computing sites* from different *organizations*. In our discussion we do not specifically target desktop grids, although they are also meant to run scientific computations [And03]. However the reader familiar with desktop grids may understand that the allocation mechanism by infiltration described in the last section also fits in their context.

1.3 Outline

The fact that grid resources are administered by multiple independent organizations introduces a number of challenges: security, user identification, information flow, seamless resource integration, central monitoring, etc. Among these, resource allocation is considered as yet another concern. Traditional systems provide scientific collaborations with access to grid resources, but do not allocate resources efficiently.

The remainder of the paper is organized as follows. In a first section we describe the way grid resource allocation is implemented traditionally: from job submission to local batch systems towards job submission to grids. We then assess the degrees of freedom of resource allocation algorithms in these infrastructures. Finally we consider grafting allocation systems: we observe their different implementations, their common model, and the new frame they define for allocation algorithms.

Chapter 2

Stressing traditional models: a survey

Despite ideals of ubiquitous resource presence inspired by power grids, computing grids emerged by implementing simple batch job submission, thus reviving with a different scale the history of computing systems [Cer94].

In this section we survey major grid projects. Their computational workload is introduced, the notion of job is defined, and grids are related to local batch systems. We then present resource allocation problems scientists may face. These notions help to understand how the independence of grid sites limited their aptitude for solving these allocation problems, which is developed in the next section.

2.1 Scientific grids on stage

A few grid projects scale to tens of thousands of nodes. Efforts to build grids started around year 2000. The infrastructure itself consists in hardware to build and link computer centers maintained mostly in public institutions, and software to operate hardware resources at the local and global level. Infrastructures currently in production rely on public hardware resource and integrate open-source and academic software distributions.

EGEE

Enabling Grids for E-science has sites mainly in Europe, but also in Taiwan and Korea. It has over 41,000 CPUs from 240 sites [ABD+04]. EGEE took in 2004 over the work of the European **DataGrid** project started in 2000 [Rud01] and the infrastructure of the **LCG**, the Computing Grid launched in 2001 for the Large Hadron Collider, CERN particle accelerator in Switzerland [BBB+05].

EGEE is supported by the European Commission and more than 90 organizations from over 30 countries. It integrates infrastructures like **GridPP**,

funded by the UK government through the Science and Technology Facilities Council, which provides 9000 processors [tGC06].

OSG

The *Open Science Grid*, started in 2005, funded by U.S. LHC software and computing programs, the National Science Foundation (NSF), and the U.S. Department of Energy. It continued **Grid3**, started in 2003 [Ave07].

TeraGrid

TeraGrid started in 2001 with funds from the NSF to establish a Distributed Terascale Facility (**DTF**). It includes collaboration from 9 major national computer centers in the U.S. It provides 250 teraflops of computing capacity and plans to integrate a petaflop system in 2009 [Pen02].

NorduGrid

In 2001 the NORDUNet2 program funded NorduGrid to build a grid for countries in northern Europe. The NORDUNet2 program aimed to respond to the "American challenge" of the Next Generation Initiative (NGI) and Internet2 (I2). NorduGrid provides around 5,000 CPUs over 50 sites [EGK+07].

Naregi

The Japanese grid project *National Research Grid Initiative* started in 2003. It is deployed in beta on a 3,000 CPUs testbed and targets the PetaFLOPS in 2010 on national computer centers. The software development is done by private companies (Fujitsu, NEC, Hitachi, NTT). It is funded by the Ministry of Education, Culture, Sports, Science and Technology (MEXT) [Miu06].

Each of these grids is distinctive by the hardware resources integrated, hence by the organizations supplying these resources, by the scientific projects supported, and by the distributed software used (*aka* **middleware**).

These infrastructures must not be confused with software development projects like **Globus**[1] and **VDT**[2] which distribute consistent sets of components for grids and are active in the standardization effort [FKNT02]. Grids may or may not integrate some of these components in their middleware.

2.2 Applications

Scientific grids are concerned with *high throughput computing*, and most often with *distributed data analysis*.

[1] www.globus.org
[2] Virtual Data Toolkit: vdt.cs.wisc.edu

Definition 6. ***High Throughput Computing*** (HTC) is the area of computer systems concerned with *effective management and exploitation of all available computing resources* in *environments that can deliver large amounts of processing capacity over long periods of time [Con96].*

We note that HTC applies in presence of multiple tasks. As opposed to **high performance** where the concern is the number of operations per second, high throughput systems are typically concerned with the statistical distribution of the time perceived by users to run their computations.

Hard computational problems with divisible workloads may typically be submitted to high-throughput systems. The resolution of NUG30, a famous quadratic assignment problem, is an early example which pushed the study of metacomputing models [ABGL00, GLY00].

Definition 7. ***Distributed data analysis*** *is the analysis of large data sets which are best handled by distributed computation [MB03].*

From the angle of resource allocation, distributed data analysis is the area of high throughput computing in which computing resource allocation is influenced by data location.

Major grids were prominently pushed by the will to analyze unprecedented amounts of data, especially in the field of particle physics. In this discipline scientists search for interesting events in the vast amount of data generated by detectors [WDRT97, GCC+04, RSZ+06]. The experiments driven at CERN, the European Center for Nuclear Research, and Fermilab, its American counterpart, attract collaborations of physicists who represent most users and contributions to EGEE/LCG, OSG, TeraGrid and NorduGrid [Ter02, FPC+02].

However particle physics are not the unique applications, as shown by Naregi, essentially targeted at nanotechnologies and biotechnologies, and EGEE, which diversifies in a variety of disciplines, including *in silico* drug discovery [LSJ+06, BBH+06].

2.3 Workloads

Computationally intensive, *embarrassingly parallel* workloads are the most straightforward to process on grids [GMP06]. They belong to the class of *divisible* problems [LSV06].

Definition 8. *A problem of size N is* ***embarrassingly parallel*** *if it is "quite easy" to achieve a computational speedup of N without any interprocess communication [Har03].*

Definition 9. *A* ***divisible*** *task is a computation which can be divided with arbitrary granularity into independent parts solved in parallel by distributed computers [BDM99].*

More precisely, distributed data analysis processed on grids present *partially data-parallel* workloads [HGLS86].

Definition 10. *A **partially data parallel** problem divides the input data into a number of completely independent parts. The same computation is undertaken on each part. It may require pre and post processing and redundant computations to avoid communication [Har03].*

For instance a paving may be extracted from satellite images for independent analysis of the elements [Zha02]. In particle physics, the thousands of tracks detected at the occasion of collisions are bunched in dozens for analysis and for each bunch.

2.4 Job: the element of a computation

Definition 11. *A **computational job** is a uniquely identifiable task, or a number of tasks running under a workflow system, which may involve the execution of one or more processes or computer programs [SAB+05].*

For clarity we consider in the following that a job may run on a single computer at a given time; a task that simultaneously runs on several computers is a set of jobs. In practice a divisible load is divided into jobs before submission to a grid and is never re-arranged after submission [GLMR07].

Job requirements

A job carries requirements constraining the allocation in order to end up on an execution node with the proper operating system flavor and application software, and close to its data.

Specific **hardware** and **software** may be necessary for a job execution. If not already present on a grid node the software may be installed before a job is run.

In distributed data analysis the bulk of the data is found on the grid site. Jobs do not carry substantial **data**, and will probably never do so, since the transfer/processing time ratio is not decreasing with the progress of technology. Specialized components of the infrastructure handle the data distribution, and jobs are allocated close to their data if possible: in the absence of intelligent data distribution mechanisms this means close to data generators and their storage devices (e.g. detectors, telescopes).

Job load

Grids understand job requirements and route them accordingly to appropriate grid nodes. A general-purpose grid is not concerned, however, with how a computation should be divided into jobs. This is application-specific, and the responsibility of grid users. Their applications have integrated submission systems and software to run on grid nodes [Mac04].

Jobs' sizes have consequences on the throughput: the time to process a job must be long with regards to the time to queue in the grid, and the whole data

required by a job must be found on a single grid site or engage little transfer [GMP07].

2.5 Expanding local systems

By allocating jobs to nodes, scientific grids extend the model of local batch systems across grid sites.

Definition 12. *A **batch system** is an enterprise software application that is in charge of unattended background executions, commonly known for historical reasons as batch processing*[3].

Before the advent of grids, scientists would submit their jobs on their local cluster, inside their own laboratory. However local clusters are not sufficient for highly demanding applications. Thus grids bring together multiple clusters from different organizations with an infrastructure that acts like a *super* or *meta*-batch system:

> What distinguishes resource management in a Grid environment from these local systems is the fact that the managed resources span administrative domains [CFK04].

We distinguish between *queuing systems* and *Condor*-based systems. We can compare the former to subcontracting companies and the latter to recruitment agencies. They influenced the evolution of grids in different ways.

Queuing systems

Most batch systems essentially manipulate job queues. We mention the most prominent. They were evaluated by [CCF+94] in their early days.

LSF. *Load Sharing Facility* started with Utopia [ZZWD93], whose authors created Platform Computing, a company with now 360 employees. See [XLT+05] for an integration into a grid.

PBS. *Portable Batch System* is maintained by Altair, a 1200 employees corporation [Hum06]. It was originally developed at NASA since 1993 [Hum06]. BPS comes along with a separate scheduler (e.g. Torque or Maui [BHK+00]) or a language to write one to implement site specific allocation policies. In [BLT03] PBS is integrated in the framework of a particle physics collaboration for use in grids or local clusters.

SGE. *Sun Grid Engine* started as DQS (Distributed Queuing System) at Florida State University in 1993. In 2000, Sun acquired all rights and renamed the product. Its integration with EGEE is described in [BDG+07].

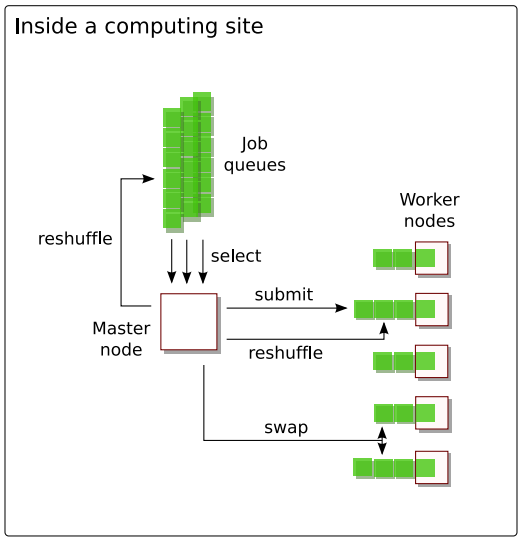

Figure 2.1: Generic batch system

Batch systems manage queues of jobs and submit jobs to available nodes, in an order optimizing performance, and relative to the site's internal precedence policies.

Figure 2.1 shows the standard features of a generic batch system. A batch system is composed with a master node and worker nodes. Jobs waiting to be scheduled are classified in different queues, depending on their priorities or their load characteristics (duration, memory consumption, required data, etc.). The scheduler running on the master node may reorder jobs within and between queues. When the load of worker nodes permits, the scheduler selects a job and allocates it to a worker node. when a job is allocated, it usually waits on a smaller queue in front of the assigned worker node. At that time, the scheduler can still reorder worker node queues or move jobs from one queue to another, in order to handle dynamic events like a priority change or a worker slowdown.

Condor: *CPU scavenging* and *matchmaking*

In addition to queue management, in order to handle *CPU scavenging*, Condor introduces the *matchmaking* mechanism.

Definition 13. ***CPU scavenging*** *means utilizing non-dedicated CPUs when primary users would otherwise leave them idle.*

CPU scavenging is primarily used in desktop grids where resources are taken from individuals computers [And03]. This is also the specificity of Condor

[3]From Wikipedia, Job scheduler (as of Nov. 8, 2007, 12:46 GMT).

among other batch systems. Focused on dynamic, heterogeneous pools of resource, Condor developed the matchmaking mechanism which became a critical component of grids.

Definition 14. ***Matchmaking*** *is the process of associating nodes and jobs into fitting couples.*

Figure 2.2 shows how this is done. A job submitted to Condor waits for a matching CPU to be idle. On the other hand, An idle node describes itself to be made available to a remote user. The match is good when the node has the resources, operating system flavor, application software and access to data required by the job to execute. Making the match is the responsibility of the **Matchmaker**, depicted on the figure.

In fact the job is not really forwarded to the matchmaker. It is kept on the user node by a *scheduler daemon*, **Schedd** on the figure. Instead of the job, the Schedd sends the job requirements in a file called **ClassAd**[4] in Condor terminology. The process is symmetric and from the idle worker node, the *starter daemon* (**Startd** on the figure) also sends a ClassAd advertising its resources [RLS98].

The **Collector** collects ClassAds, passes them to the Matchmaker. When a match is appropriate, the **Scheduler** notifies the corresponding Schedd with the address of its appointed Startd. The Schedd launches a process, the **Shadow**, that sends the job directly to the Startd. The Startd launches a process, the **Starter**, to take care of the job. Shadow and Starter stay connected while the job is running.

Like any other batch system, Condor is used inside grid sites. As we will see in the following sections, its derivatives are also used in other core places of grids and allocation systems [TTL02].

2.6 Scheduling problems

Grids have been built under the assumption that the input of their resource allocation system is an independent set of jobs. In general scheduling problems, however, resource allocation systems face other constraints such as co-allocation, priorities, job dependencies, interactivity, inter-process communication and migration.

- **Co-allocation.** The co-allocation of several jobs to the same node in order to improve resource utilization is not processed at the global grid level but left to the grid sites policies. The synchronous co-allocation of several nodes to a set of jobs to means the allocation of several nodes is a requirement for IPC and raises interesting challenges [CFK99].

- **Run-time priorities.** On a single machine run-time priorities are set by the CPU scheduler provided as part of the operating system: it dispatches

[4] from *Classified Advertisement*

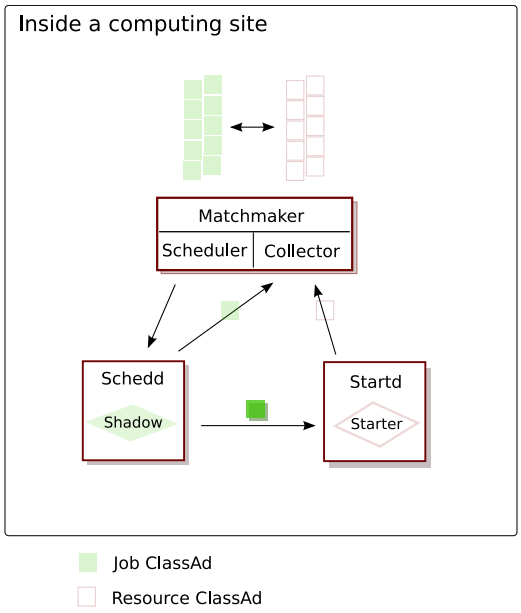

Figure 2.2: Condor

the computing capacity between different processes so that priorities are maintained while processes are running. At the scale of a computing center, resources can also be attributed in a dynamic manner, as is the case for networked services [CGR+06].

- **Dependencies**. Dependencies are expressed with Direct Acyclic Graphs (DAGs). DAGs occur when, in order to start, a job needs another job to be processed first. DAGs constrain the workflow [MFRW06]. When there is no proper DAG management mechanism in the infrastructure an alternative consists in tinkering with the submission interface: the application waits for the grid to return the result of a job before submitting the next one in the DAG.

- **Interactive jobs.** While the job is running, it communicates with the user and needs user input to proceed. This case requires a communication path between the user and the execution node. It requires that the job starts straight at submission because the user is waiting. Interaction with the user induces long interrupts. Condor provides a library to support interactive jobs, as long as a connexion between the user and worker nodes is possible.

- **Inter-Process Communication (IPC).** Jobs are not really independent and need to communicate at runtime. This requires a communication

path with low latency between the nodes where jobs are executed, and a synchronous execution.

- **Process migration.** On a single system image, processes are constantly migrated from one node to another where they find the resources they need. Condor and LSF enable process migration provided the code of the application linked with a specific library. For most systems, job migration means that a job is moved from one queue to another instead of migrated at runtime [MDP+00]. Grids currently do not handle job migration: if a job, possibly after many hours, overpasses its lease period on its node, it has to be resubmitted and starts again from scratch on another node.

These problems may well be solved on local clusters. In grids, additional constraints make their resolution more complex: *latency between sites is inherently high; access to data is not uniform; the infrastructure development is separated from the development of its applications; and grid sites are independent administrative domains.* Each of these four constraints would require proper investigation. In the following we focus on the grid sites' administrative independence and the serious consequence of this constraint on resource allocation on grids.

Chapter 3

Disruptions to resource allocation

In this section we explain how the independence of resource providers determined a *de facto* choice for a grid allocation model.

Before addressing resource allocation, grids are shaped by other concerns that result from the independence of grid sites: the need for a site to authenticate grid users, apply its own policies, account its resource heterogeneity and integrate its own systems. These determinants suffice to draw a simple picture of the level of freedom left to resource allocation algorithms in grids.

3.1 Authentication at the site gatekeeper

Grid resource allocation is disrupted by the separation between grid sites and the grid broker.

In a batch queuing system (fig. 2.1), the user submits a job to the master, which in turn, submits to the worker. In general-purpose grids, the job flow is analogous but more complex.

Definition 15. A **Grid resource broker** is an service with which *the end users interact* and *that performs resource discovery, scheduling, and the processing of application jobs on the distributed Grid resources [ABK+04].*

The grid broker chooses an appropriate grid site and delegates job allocation to its *gatekeeper*.

Definition 16. *The **gatekeeper** is a process which exists before any request is submitted. When the gatekeeper receives an allocation request from a client, it mutually authenticates with the client, maps the requester to a local user, starts a job manager on the local host as the local user, and passes the allocation arguments to the newly created job manager [Gra].*

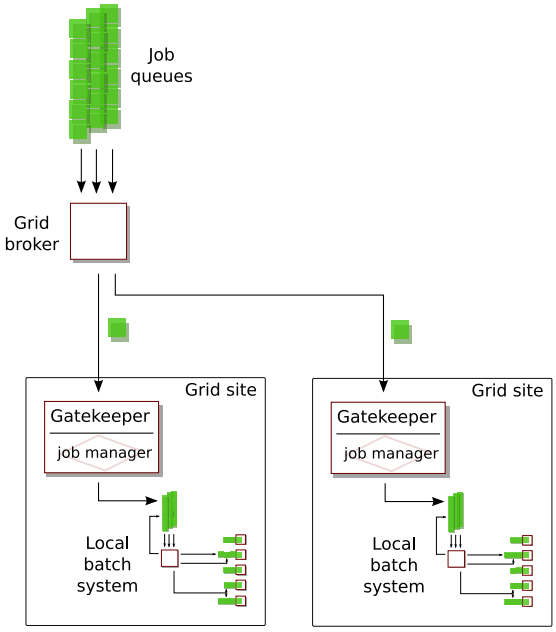

Figure 3.1: Delegation of job submission to Globus Resource Allocation Manager (GRAM)

Grid sites authenticate jobs because they only accept jobs submitted by trusted users, and provide differentiated *service levels* to different users. Job allocation is simply delegated by the grid broker to the grid site via the site gatekeeper because this is the simplest way to let grid sites authenticate jobs and enforce their own access policies.

Definition 17. ***Service Level Agreements*** *(SLA)* implement *Commitments and assurances* and *determine the contract between the user and the service provider stating the expectations and obligations that exist between the two [PDD05].*

GRAM

In most infrastructures, authentication and delegation is done by *Globus Resource Allocation Manager* (GRAM), illustrated on figure 3.1 [Fos06]. Jobs queue at the **Grid broker** which acts like the master node of a local queuing batch system (see fig. 2.1). The broker never submits to a worker node. Instead it finds an appropriate site and submits a job to its **Gatekeeper**, along with a certificate to authenticate the owner of the job. If the gatekeeper refuses the job it sends it back to the broker. If it accepts it, it starts a **job manager** process. The job manager submits the job to the local batch system and collects job status information.

As a consequence of this necessary separation between sites and global bro-

kers, no single component in a grid controls resource allocation from job submission to end node assignment, and the set of grid nodes is not considered as a whole: a node is never compared for assignment with another node from another site.

Execution nodes are disconnected from the grid broker and from each other across grid sites. They are also disconnected from the users.

3.2 User and job disconnected

A few batch systems allow interactive jobs and runtime job migration using a direct connection between user and execution node. In grids, since job submission is delegated to sites and execution nodes are not directly accessible from outside of their site, this connection is lost.

From Condor to Condor-G

In Condor, daemons take responsibility of a job on both user node (the *shadow*) and execution node (the *starter*). This is mentioned in section 2.5 and depicted on figure 2.2. These daemons maintain a connexion between each other through which users may control their jobs at runtime. Provided the job's code is linked with a specific Condor library, the starter can checkpoint it while it is running, and send checkpoints to the shadow, possibly for migration on another execution node.

This connexion, runtime management and migration was lost with Condor-G, that the Condor team produced in 2001 in order to integrate delegation to grid sites [FTF+02]. The *G* originated from *Globus* because Condor-G was initially intended to support GRAM, but was turned to *Grid* in the end to denote the generality of the mechanism (fig. 3.2).

By comparing fig. 3.2 with fig. 2.2, we note that the **Grid broker** has taken over the **Schedd** that would be run by the user in Condor. This means that the user fully delegates job submission and all potential management to the broker. The **Grid manager** is the process responsible for the job and launched by the schedd. The grid manager must submit the job to the **Gatekeeper** of the **Grid site**. As on fig. 3.1, the gatekeeper decides to accept or return the job to the broker. If it accepts the job, it forwards it to the **Startd**, which deploys a **Job manager** process. The job manager, in turn, submits to the grid site's **Local batch system** of choice, which can be other than Condor. In recent versions of Condor-G, the **Matchmaker** can be used to select an appropriate site.

Grid manager and Job manager in Condor-G are the counterparts of Shadow and Starter in Condor. They have reduced functionality because they only submit jobs and do not manage them at runtime.

The example of gLite

Condor-G illustrates the separation between a user and her running job. In practice Condor-G can be integrated into more complex submission chains. This

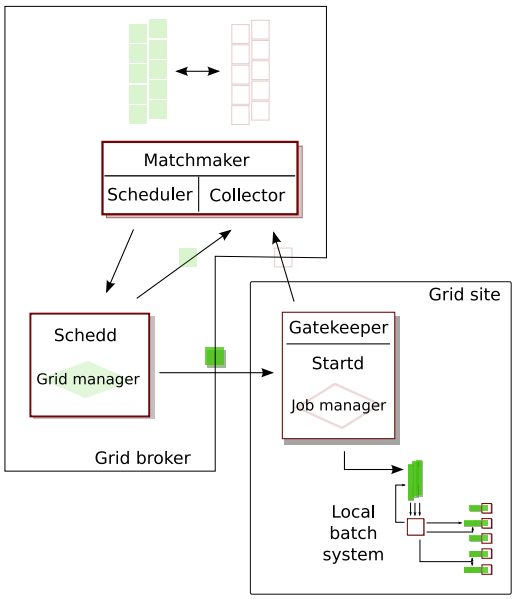

Figure 3.2: Condor-G

is the case in gLite, the LCG/EGEE middleware distribution [Lit07]. To handle its heavy load, gLite replicates its broker, and uses a trick to know to which instance the job status should be communicated back. Once a site is selected to run a job, a gLite broker submits a Condor Schedd to this site. Once running on the site the Schedd sends a simple ClassAd to the gLite broker to request its associated real job. The *Schedd* then forwards the job to the local batch system. It knows which replica of the grid broker sent the job and can communicate back the job status. In this case the Condor matchmaker is not used to select the appropriate site but only to wait for the request of a single Schedd.

Once it is accepted that in the end, allocation capabilities are not enhanced by intricate job submission chains, simple replacement systems are developed. An alternative deployment of gLite proposes two simple java servlets, ICE and CREAM[1], that implement the practical functionality of a gLite grid broker (ICE) and interface a local batch system on a grid site (CREAM) [And06]. ICE and CREAM provide only the access to the different sites and no solution to specific allocation problems, but no less than other grid implementations.

We describe these infrastructures' capabilities in the remaining of part 3 on disruptions to resource allocation, after more details on the site selection process.

[1]Computing Resource Execution And Management

3.3 Site selection

From the previous section we understand that a grid broker acts as a gate to the different sites and does not match jobs against end nodes for allocation. Instead, it matches job requirements against site resource advertisements, similarly to Condor matchmaking process between a job and a node.

Job description

When a a job is submitted to a grid, it carries along a description of hardware and software flavor and configuration, resources and data that it expects to find on the execution node. This is done in a format chosen by the infrastructure. gLite uses Condor ClassAds (see section 2.5), renamed **JDL** (*Job Description Language*). NorduGrid middleware, ARC[2], has its own format called **RSF** (*Resource Selection Language*). The Open Grid Forum, a standardization consortium assembling representatives from industry and academia, recommends another variant: **JSDL** (*Job Submission Description Language*) [SAB+05].

Resource description

Sites, on the other hand, advertise their resources. They do not publish specific information for every node but instead group their nodes into *computing elements* [Chi04].

Definition 18. *As a common abstraction, the **Computing Element** refers to the characteristics, resource set and policies of a single queue of the underlying management system.*

*At the Grid level, computing capabilities appear as **Computing Elements** (each being a set of job slots to which policies and status information are associated) that are reachable from a specific network endpoint [ABD+07].*

The description of a computing element includes information about the flavor, configuration, resource of its nodes, along with the number of nodes, overall load, and access control rules. The same configuration is maintained on all nodes of a computing element, so that a job description matches a computing element as a whole, and the grid broker submits jobs to the intent of a computing element, and not to each node independently. Therefore each computing element has its own batch system. A site usually contains one or a few computing elements.

Computing elements can be advertised according to different models: the **GLUE** schema, which stands for *Grid Laboratory Uniform Environment* is the most used by early grids. GLUE is specialized in grid resources and its specifications extend our discussion with insightful details [ABD+07]. The *Common Information Model* (**CIM**) is a general schema recommended by DMTF[3] and used as an alternative notably in Naregi. Different schemes may be converted

[2] Advanced Resource Connector

[3] Distributed Management Task Force: www.dmtf.org

into ClassAds in the grid broker for processing by Condor matchmaker (e.g. in OSG Resource Selection Service (ReSS) [BGK+03]).

Among suitable computing elements, the grid broker submits to the least loaded. Each computing element typically updates its information every five to fifteen minutes, which leaves little space for dynamic allocation.

Bypassing site selection

Users have the option to skip queues at the broker and submit directly to a computing element known to be appropriate. Jobs may also be submitted to a local batch system which redirects them to the grid broker only if no appropriate node is found locally. This is known as the **flocking mechanism** [TTL02].

The grid broker is the top-level component in grid resource allocation. It handles relatively static, bulk resource information. The next level is the local systems. How do their capabilities scale to cross-organization scheduling?

3.4 Intersecting the capabilities of local systems

A grid broker heads a variety of local batch systems. Communication is seamless but skims allocation capabilities.

There are two ways to integrate different implementations in the same infrastructure: standard interfaces or translators.

Standards

In this first approach, components on both sides of a communication implement a standard interface[4] [Fos05]. Once an agreement is reached concerning the generic interface between two well-defined components, and once all flavors of these components implement this interface, functionality is not lost along the path for communication problems [FKNT02].

It is however difficult to agree on functionality. There is not yet even a clear agreement with regards to the components involved in a grid. To reach such agreements, not only must standards integrate current practice, they also need to predict evolutions.

Translators

The second approach is pragmatic. Before standard interfaces are implemented for all components in use, these components have to communicate anyways. Instructions from the grid broker, forwarded to the site, must be understood by the variety of local batch systems available.

[4]Standardization organizations involved in grid computing include OGF (Open Grid Forum: ogf.org), IETF (Internet Engineering Task Force: ietf.org), OASIS (Organization for the Advancement of Structured Information Standards: oasis-open.org), DMTF (Distributed Management Task Force: dmtf.org).

Thus comes the **GAHP** (*Grid ASCII Helper Protocol*). GAHP is a translation protocol originally developed as part of Globus Toolkit [Fos06]. It translates instructions from the grid broker to various implementations of a site's gatekeeper (Globus, Condor, gLite, etc) and from the site's job manager[5] to the local batch system [NYI+05].

Unfortunately the vocabulary of translators intersects the capabilities of the systems they interface. With a few variants is is reduced to: **submit** to submit a job, **cancel** to cancel a job submission and **status** to get the status of a job submission [Reb05].

In this section we noticed that grid resources are located under different administrative domains. As a consequence resource allocation is delegated to clusters instead of grid-wide, and the management capabilities left to the user or a central allocation system are simplistic [NLJ+05]. Advanced allocation strategies must look at the problem from a different angle, which we do in the next section.

[5] the process to which a job is delegated on the site, see section 3.2.

Chapter 4

User-driven allocation

4.1 Motivation

Grids follow the application-agnostic model of a *super*-batch system with reduced functionality. This model lacks opportunities for allocation *problem-solving* and *performance*.

Problem-solving. In practice, users or applications may have requirements susceptible to affect resource allocation: DAGs, interactivity, IPC, real-time prioritization, etc. these demands are not covered by grids. As a consequence, only a restricted class of applications can be handled directly: independent jobs resulting from embarrassingly parallel, divisible workload.

Performance. In these favorable cases, specific job profiles may be worth taking into account for efficient allocation. For example users may know that some of their jobs are network intensive and others CPU-intensive. In this scenario it would not decrease the performance to co-allocate a network-bound job with a CPU-bound job on the same node and thus save half of the nodes. Such opportunities are lost by relying on the grid for the allocation [XZQ00].

From previous section we conclude that the provision of grid resources is made inflexible by side concerns subsequent to the independence of grid sites. Grids do not have the freedom left to implement all particular scheduling strategies that arise from application requirements and optimization opportunities.

> The field of performance analysis and optimization for Grid applications is still in its infancy [RMdL04].

A practical solution resides in independent allocation systems implemented by users or application portals, grafted to the grid.

In the following we define their common model, their implementations, their benefits in presence of large federations of collaborative users; we then derive the new constraints for resource allocation methods in these environments.

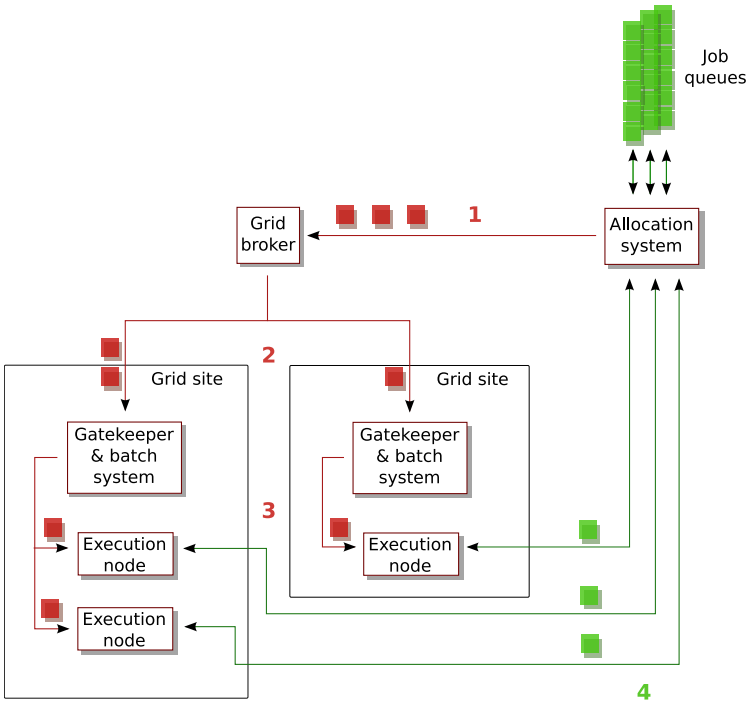

Figure 4.1: Generic infiltrating system

4.2 Infiltrating resources

Independent allocation systems may take temporary control over grid nodes, as explained in the following.

Definition 19. *We call* ***infiltration system*** *a system that submits monitors to grids under the form of jobs, and submits actual jobs directly to these monitors when they are running on grid nodes.*

An infiltration system is described on figure 4.1. The central component submits monitors to the grid broker through the standard grid job submission mechanism (1): the **Grid broker** delegates submission to the site **Gatekeeper** (2) which forwards it to the **batch system**, and the batch system assigns an **Execution node** (3). Once running on the execution node, the monitor waits for actual job submissions from the **Allocation system** (4).

Definition 20. *In the following we call* ***direct submission*** *a job submission to a running monitor by an infiltration system, as opposed to* ***standard submission****: a job submission to a grid broker or a computing element.*

The infiltration mechanism allows:

- **No submission delay.** Direct submission results to consecutive execution.
- **Runtime communication.** A connection or a message initiated by the monitor can cross the site's firewall.

  The monitor communicates information about the node and the job, the allocation system replies with scheduling decisions, and the application or user may interact with the job at runtime.

4.3 Allocation by applications

Direct control necessary for fine grid resource allocation is possible by infiltrating resources accessed through grids.

Applications benefit from infiltration

Performance gains and problem solving capability justify that applications implement their infiltration system.

Performance. Applications benefit from the absence of delay between job submission and execution, after the start-up time corresponding to the submission of monitors. To the opposite, by directly submitting jobs to EGEE, half of them wait for more than five minutes before execution, and 5% wait for more than 15 minutes [GLMR07].

Problem solving. By directly interacting with end nodes, application-specific problems (see section 2.6) may be addressed: Job dependencies, interaction with the user, interaction between jobs, real-time control, priorities; Besides strict resource allocation, fault tolerance may be provided [J.T06].

Development. An application ported to a grid generates jobs and submits them, infiltrating grid nodes does not require much additional effort, which may still be eased by general-purpose frameworks.

DIANE, an infiltration framework

DIANE[1] is a framework that implements infiltration logic for applications [Mos03]. The project started in 2002 to port CERN applications to the DataGrid, the ancestor of LCG/EGEE. It has also been used notably for gene sequencing [MHS+04], and for one of the pilot applications of EGEE, *in-silico* drug discovery against malaria and bird flu [LHC+06].

DIANE dynamically balances jobs across grid nodes in order to minimize the time before the last job completes and the application returns: it keeps queues in front of each execution node and re-assigns jobs to different queues. Re-assignment is triggered when new monitors start execution and contact the

[1] DIstributed ANalysis Environment

framework or when the job progress on a node is too slow [J.T06].

The power of DIANE's load balancing system resides in the heavy workload of large, embarrassingly parallel applications: job liquidity is sufficient to gain substantially by coordinating their allocation. Comparable or greater liquidity can be reached by centralizing job submissions in a scientific collaboration.

4.4 Allocation by collaborations

Scientific collaborations already provide an 'access card' to grids. In addition they may provide their own centralized allocation systems.

Virtual Organizations

Let us recall the definition of Virtual Organization (def. 2) by [Fos01]: a *group of organizations and/or individuals* who *share resources in a controlled fashion, so that members may collaborate to achieve a shared goal.*

In practice the scientific collaborations that consume grid resources are distinct from the institutions that provide resources to grids. The term virtual organisation almost always refers to federations of grid users and not resource providers. We adopt this restriction in the remainder of the paper.

In particle physics, A VO corresponds to a collaboration that builds a detector and analyzes its data.

- At Fermilab, in the area of Chicago, **CDF**[2] and **D0**[3] are two VOs that study the results of Protons-Antiprotons collisions, scheduled to analyze data until 2009. **MINOS**[4] analyzes Neutrino oscillations.
- At SLAC, *Stanford Linear Accelerator Center*, **BaBar**[5] analyzes the violation of charge and parity (CP) symmetry in the decays of B mesons.
- At CERN, the European Center for Nuclear Research in Geneva, four VOs are finishing to build detectors and preparing for data analysis for the coming years. **Atlas**[6] and **CMS**[7] are two general-purpose detectors to analyze proton-proton and heavy ions collisions. **Alice**[8] studies Pb-Pb collisions generating a quark-gluon plasma as in the early universe, and **LHCb**[9] studies collisions of baryons containing the Beauty quark for CP violation measurements and rare decays observations.

[2] *Collider Detector at Fermilab.* www-cdf.fnal.gov
[3] www-d0.fnal.gov
[4] *Main Injector Neutrino Oscillator Search.* www-numi.fnal.gov
[5] $B\bar{B}$. www-public.slac.stanford.edu/babar
[6] *A Toroidal LHC ApparatuS.* altas.ch
[7] *Compact Muon Solenoid.* cms.cern.ch
[8] *Large Ion Collider Experiment.* aliceinfo.cern.ch
[9] *Large Hadron Collider beauty.* lhcb.web.cern.ch

EGEE supports many other scientific collaborations. For example, **BIOMED** is a VO divided in 3 sectors: Medical Imaging, Bio-informatics and Drug Discovery. Major applications of the area are ported to EGEE, with integrated job generation and submission [LHC+04], and domain-specific portals are provided [GSM+07].

Grids originated inside particle physics VOs (see section 2.1). In resulting general-purpose grids the VO name is now a group identifier for a grid user, which lets grid sites define bulk resource supply contracts and authenticate job owners [Fos01]. In these infrastructures VOs do not control the hardware but they may control the resource allocation by infiltration.

Benefits

There are several reasons why the infiltration model is well suited to VOs: VOs share the efforts; they have homogeneous resource usage; their members collaborate and their large number leverage optimization opportunities.

- **De-multiplied effort.** In large VOs most job submissions to grids are naturally delegated to a few members. The gap is narrow between centralized skills to manage grid jobs and the development of a single allocation system for the whole VO.
- **Homogeneity.** Jobs from the same VO have similar resource requirements: they expect the same software to run on execution nodes, and their profile follow a certain regularity.
- **Cooperation.** VO members collaborate for a shared goal, so a global allocation strategy can be implemented instead of a competition between multiple users. For example a job may be reasonably delayed to run another more important for the group or for the overall allocation performance.
- **Liquidity.** Finally, VOs may be large. For example 1900 physicists collaborate in Atlas. The number of users and jobs, and thus the number of grid nodes controlled by a VO, may gives sufficient liquidity to justify an allocation mechanism and raise opportunities for performance optimization.

Infiltration mechanism

VOs request the highest possible/useful number of grid nodes and maximize the computing throughput on these nodes.

Grid nodes are requested by submitting monitors as described in section 4.2. Usually a job can run on a grid node until its termination. A monitor pretends to be a job that never terminates. However there are maximum job lengths set by site policies, typically 48 or 72 hours.

For this time period, monitors get actual jobs from the VO and control their execution. If jobs appear to the grid as if they run for days, actual jobs last for

a few minutes to a few hours. They bypass queues of the traditional grid submission flow to be directly allocated to grid nodes following a strategy defined for the benefit of the collaboration.

VO-specific infiltration systems are relatively recent and few. Despite their performance they are not widely identified in the literature. We briefly survey them below.

4.5 Sudden success of an old Condor mechanism.

Specialized in CPU scavenging (def 13), Condor was designed to operate execution nodes with transient availability. Therefore it was well suited to serve as the basis for a generic infiltration system. The following is a description of Condor infiltration mechanism and Condor-based infiltration systems.

Definition 21. *The Condor-G **GlideIn** mechanism is the use of grid protocols to dynamically create a Condor pool out of grid resources by "gliding-in" Condor daemons to the remote resource* (adapted from [FTF+02]).

Portable shell scripts called **glideIns** are submitted to a grid and, once running on end nodes, launch processes equivalent to Condor startd's[10] connecting to a central Condor matchmaker. The GildeIn mechanism is available in Condor since version 6.1 (year 2001).

Only recently a few allocation systems equipped with a Condor matchmaker started to systematically send glideIns for subsequent direct submissions to grid nodes. Once glideIns are running on worker nodes, direct job submissions are processed like in a local condor system (figure 2.2).

Communication

Direct communication between the user's shadow and the glideIn's starter may be instantiated by the starter because grid nodes and grid sites normally have outbound connectivity.

To bridge incoming messages from the central allocation system to glideIn's running on execution nodes, Condor provides a proxy: the GCB (*Generic Connexion Broker*) [BSK05]. Grid sites may run this proxy on a special node where VO administrators have direct access.

glideCAF

The Central Analysis Farm (CAF) of CDF, the particle physics VO and Fermilab experiment, was extended in 2005 to use grid resources with glideCAF, which integrates direct submission to Condor glideIns [BHL+06].

[10] See section 2.5

Cronus

An individual initiative in Atlas gradually gained momentum and led to Cronus in 2006 [PW07]. Cronus allocates a substantial part of Atlas jobs and controls a dynamic pool of about 8500 CPUs infiltrated through EGEE, OSG and NorduGrid.

In addition to short-circuiting grid submission delays, Cronus manages *data distribution* and *load takeover*.

- **Data distribution.** Considering that Atlas jobs do not consume network bandwidth, Cronus glideIns download data in the background from major storage systems. Future jobs do not wait for their data to be downloaded. Instead they are executed where their data already is.

  NorduGrid already plans in advance data downloads before jobs are scheduled for execution, but EGEE does not: jobs start execution by requesting the data and stay idle until the download is complete. Cronus saves this idle time.

- **Load takeover.** Cronus lets glide-ins take over the jobs of others whose lease is about to expire. This saves from 80% of the jobs failures observed on standard grid submission.

glideinWMS

glideinWMS is a project started in 2007 by US CMS, the American part of the CMS collaboration. It extends glideCAF and Cronus information system [Sfi07].

4.6 An evolution of VO strategies

Infiltration naturally emerged from the same VOs that built the cornerstones of general purpose grids. CMS was the last CERN VO to start working on infiltartion systems. Agents were introduced in *AliEn* (Alice) for node control and configuration. They leveraged in *DIRAC* (LHCb) for high throughput, and *Panda* (Atlas) followed.

AliEn: controlling resources

AliEn, *Alice Environment*, has been since 2001 the distributed data analysis environment of Alice, a CERN detector and collaboration [BPS03, SAB+03, SBP03]. It is written in Perl. In 2004 it served as a basis for the general-purpose gLite middleware (LCG/EGEE) and kept evolving besides for Alice [BPSGO04, LHA+04]. AliEn was the first infrastructure where resources (computing elements) trigger job submission based on their real-time state. While job submission in gLite follows the model described in section 2, AliEn is based on *job agents*.

Definition 22. ***Job Agent:*** *Web Service allowing users to interact with the running job, send a signal or inspect the output. Prior to job execution, the Job Agent can automatically install the software packages required by the job [BPSGO04].*

With AliEn, jobs wait in Alice central queue until they are requested by job agents on grid sites where they can be processed: Local batch systems receive jobs only for consecutive processing. General-purpose grids have not managed to integrate this mechanism with the management of multiple VOs: grid sites receive jobs without having requested them and thus let them queue again until a node is available.

DIRAC: improving throughput

DIRAC, *Distributed Infrastructure with Remote Agent Control*, started in 2003 for LHCb, another CERN detector and collaboration [vHCF+03, TGSR04]. It implemented the model of AliEn in Python. In 2004, DIRAC made explicit the benefits of direct submission for performance by infiltrating grid nodes with *pilots agents* while complying with the security rules of European grids [PSP06].

Definition 23. ***Pilot agent:*** *process submitted to a grid site that initiates an outgoing connection to a central allocation system and requests a job whenever the corresponding resource is free (adapted from [PT06]).*

Instead of submitting jobs to the local batch system, pilot agents are submitted themselves and advertise the node where they land for direct submissions. By doing so DIRAC improves the throughput at the VO level without altering the competition between VOs at the grid level [PT06]. AliEn's job agents also integrated this mechanism.

In late 2005, the US Department of Energy funded a project, **Panda**, based on DIRAC to manage Atlas jobs on OSG [WLrW06, Nil07].

Allocation mechanism

In AliEn or DIRAC, jobs are stored in a central server in a number of queues. When a monitor (Job agent for AliEn or Pilot agent for DIRAC) requests a job, the whole job list may be scanned to select the most appropriate job according to the monitor's information on the node. In practice a queue is selected, known to contain the most appropriate class of jobs, and one of the first jobs found on that queue is sent.

4.7 Specific constraints

As opposed to the use of dedicated clusters, the infiltration strategy presents specific constraints for the deployment of systems like AliEn, DIRAC, Condor and glideCAF.

Convoluted communication. A connection can be instantiated only from the inside of a grid site: by monitors connecting to the infiltration system. The infiltration system cannot instantiate a connection with a monitor, nor a monitor with another monitor on another site.

In practice however, each grid site provides a dedicated machine, the **VO box**, for direct access to VO administrators. The VO box may be used as a proxy (e.g. Condor GCB: section 4.5). But the VO box is considered a hack that grids and sites temporarily accept to bypass a number of their known limitations until a better solution is found.

In general the remaining possible communication is asynchronous, via messages passing and polling by monitors.

Limited node control. Monitors are simple jobs from the viewpoint of the grid site. At the moment they are unprivileged though they are owned by VO administrators. Virtual machine deployments (See for example [GPJ+07, KDF04]) is a possible solution to give VOs more control on their environments from the operating system flavor to the software, but is not used on grid execution nodes at the time of writing [FPC+02].

Infiltration systems project the model of a local cluster to the wide area, with a few differences: new constraints also apply to their allocation algorithms as opposed to allocation on a local cluster.

Transient node availability. Nodes leave the pool when their maximum lease period expires. Jobs (and hence monitors) are notified before. Other nodes come in the pool when a newly submitted monitor starts execution. Infiltrated resource is a pool of blinking nodes.

Latency. Both data access and communication between peers of a distributed allocation system are affected. Problems arise that were not present in the local area context: data-driven allocation and local knowledge of the system state (figure 4.2).

Scientific grids used through infiltration convert the cross-organization barrier into these few constraints. Research in resource allocation that did not take this barrier into account may still be readily adapted, e.g. [HSLL00].

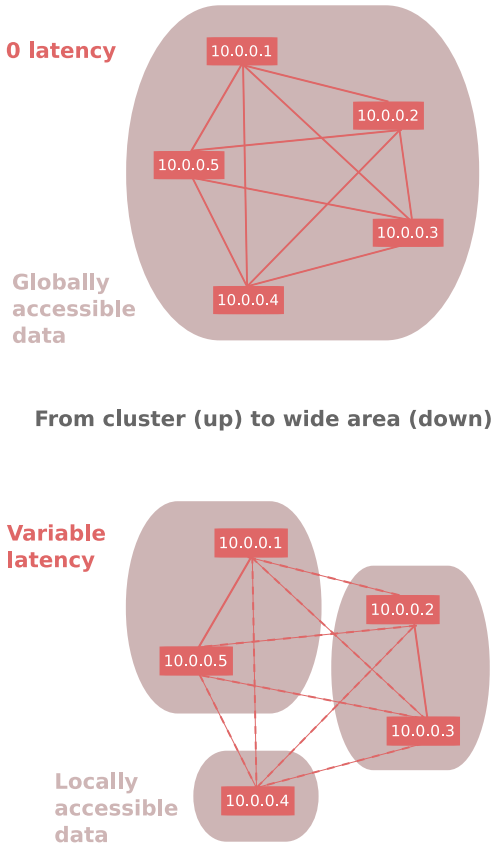

Figure 4.2: Latency constraint on wide area.

Chapter 5

Conclusion

In this survey we gave an overview of practical resource allocation in scientific grids. We considered a grid as a world-wide collection of resources located across a variety of computing sites under independent administrative domains. Grids route computations from a variety of applications to these resources according to capabilities and contracts while trying to limit obtrusiveness to local policies and impact on security.

Large federations of users generate enough liquidity to graft efficient allocation mechanisms based on their own needs and workflows. They do so by submitting monitors to grids which, once running on end nodes, report to a central allocation system. Grids are thus partitioned into dynamic pools controlled at the user level by these federations. Direct submission to end nodes and runtime control enables fine-grained, dynamic foreign resource allocation. This mechanism realizes the move from local resource allocation across different applications, to usage-specific allocation without *a priori* knowledge of resource topology.

Since allocation is centralized and direct, throughput optimization does not need to wait for more grid standards. Boundaries between resources from different institutions are overpassed.

Chapter 6

Acknowledgments

This work was supported by HP Labs. Thanks to Peter Toft (HP) for his supervision.

Thanks to Sanjay Padhi (UoW, Atlas), Predrag Buncic (CERN, Alice), Maarten Litmaath (CERN, EGEE), Jakub Moscicki (CERN, LHCb, Atlas) and Stuart Paterson (CERN, LHCb) for sharing their expertise and showing that resource allocation in grids is emerging as a fascinating research aera.

Bibliography

[ABD+04] P. Andreetto, Valentina Borgia, A. Dorigo, A. Gianelle, M. Mordacchini, M. Sgaravatto, L. Zangrando, S. Andreozzi, V. Ciaschini, C. Di Giusto, F. Giacomini, V. Medici, E. Ronchieri, V. Venturi, G. Avellino, S. Beco, A. Maraschini, F. Pacini, A. Guarise, and G. Patania. Practical approaches to grid workload and resource management in the egee project. In *CHEP '04: Proceedings of the Conference on Computing in High Energy and Nuclear Physics*, volume 2, pages 899–902, Interlaken, Switzerland, 09 2004.

[ABD+07] Sergio Andreozzi, Stephen Burke, Flavia Donno, Laurence Field, Steve Fisher, Jens Jensen, Balazs Konya, Maarten Litmaath, Marco Mambelli, Jennifer M. Schopf, Matt Viljoen, Antony Wilson, and Riccardo Zappi. Glue schema specification version 1.3. glueschema.forge.cnaf.infn.it/Spec/V13, 16 Jan 2007.

[ABGL00] K. M. Anstreicher, N. W. Brixius, J.-P. Goux, and J. Linderoth. Solving large quadratic assignment problems on computational grids. Technical report, MetaNEOS project, Iowa City, Iowa 52242, 2000.

[ABK+04] Parvin Asadzadeh, Rajkumar Buyya, Chun Ling Kei, Deepa Nayar, and Srikumar Venugopal. Global grids and software toolkits: A study of four grid middleware technologies. Technical Report GRIDS-TR-2004-5, Grid Computing and Distributed Systems Laboratory, University of Melbourne, Australia, July 1 2004.

[And03] David P. Anderson. Public computing: Reconnecting people to science. In *Proceedings of the Conference on Shared Knowledge and the Web*, Residencia de Estudiantes, Madrid, Spain, Nov. 17-19 2003.

[And06] P. Andreetto. Cream: A simple, grid-accessible, job management system for local computational resources. In *Proceedings of CHEP'06*, Mumbay, India, February 2006.

[Ave07] Paul Avery. Open science grid: Building and sustaining general cyberinfrastructure using a collaborative approach. In *Cyberin-*

frastructure for Collaboration and Innovation, number CSD5052, June 2007.

[BBB+05] I. Bird, K. Bos, N. Brook, D. Duellmann, C. Eck, I. Fisk, D. Foster, B. Gibbard, C. Grandi, F. Grey, J. Harvey, A. Heiss, F. Hemmer, S. Jarp, R. Jones, D. Kelsey, J. Knobloch, M. Lamanna, H. Marten, P. Mato Vila, F. Ould-Saada, B. Panzer-Steindel, L. Perini, L. Robertson, Y. Schutz, U. Schwickerath, J. Shiers, and T. Wenaus. Lcg technical design report. Technical report, CERN, 06 2005.

[BBH+06] R. Barbera, V. Breton, F. Harris, M. Lamanna, C. Loomis, and J. Montagnat. *Second revision of EGEE Application Migration Progress Report*. EGEE, 04 2006.

[BDG+07] Goncalo Borges, Mario David, J Gomes, Carlos Fernandez, Javier Lopez Cacheiro, Pablo Rey Mayo, Alvaro Simon Garcia, Dave Kant, and Keith Sephton. Sun Grid Engine, a new scheduler for EGEE middleware. In *IBERGRID - Iberian Grid Infrastructure Conference*, May 2007.

[BDM99] Jacek BaImageewicz, Maciej Drozdowski, and Mariusz Markiewicz. Divisible task scheduling concept and verification. *Parallel Computing*, 25(1):87–98, January 1999.

[BFH03] Fran Berman, Geoffrey Fox, and Tony Hey. The grid: past, present, future. In Fran Berman, Geoffrey Fox, and Tony Hey, editors, *Grid Computing Making the Global Infrastructure a Reality*. John Wiley & Sons, 2003.

[BGK+03] A. Baranovski, G. Garzoglio, A. Kreymer, L. Lueking, V. Murthi, P. Mhashikar, F. Ratnikov, A. Roy, T. Rockwell, S. Stonjek T. Tannenbaum, I. Terekhov, R. Walker, and F. Wuerthwein. Management of grid jobs and information within samgrid. In *Proceedings of UK e-Science All Hands Conference*, Nottingham, UK, September 2003.

[BHK+00] Brett Bode, David M. Halstead, Ricky Kendall, Zhou Lei, and David Jackson. The portable batch scheduler and the maui scheduler on linux clusters. In *Proceedings of the 4th Annual Showcase & Conference (LINUX-00)*, pages 217–224, Berkeley, CA, October 10–14 2000. The USENIX Association.

[BHL+06] Stefano Belforte, Shih-Chieh Hsu, Elliot Lipeles, Matthew Norman, Frank Wu thwein, Donatella Lucchesi, Subir Sarkar, and Igor Sfiligoi. Glidecaf: A late binding approach to the grid. In *Proceedings of CHEP'06*, 2006.

[BLT03] V. Berten, L.Goossens, and Chun L. Tan. Atlas commander: an atlas production tool. In *Proceedings of Computing in High Energy and Nuclear Physics*, La Jolla, California, 24-28 March 2003.

[BPS03] Predrag Buncic, Andreas J. Peters, and Pablo Saiz. The alien system, status and perspectives. In *Proceedings of Computing in High-Energy Physics*, 2003.

[BPSGO04] Predrag Buncic, A. J. Peters, P. Saiz, and J.F. Grosse-Oetringhaus. The architecture of the alien system. In *Proceedings of CHEP'2004*, Interlaken, Switzerland, 09 2004.

[BSK05] Bruce Beckles, Sechang Son, and John Kewley. Current methods for negotiating firewalls for the condor system. In *Proceedings of the 4th UK e-Science All Hands Meeting 2005*, Nottingham, UK, September 2005.

[CCF+94] K. Castagnera, D. Cheng, R. Fatoohi, E. Hook, B. Kramer, C. Manning, J. Musch, C. Niggley, W. Saphir, D. Sheppard, M. Smith, I. Stockdale, S. Welch, R. Williams, and D. Yip. Nas experiences with a prototype cluster of workstations. In *Supercomputing '94: Proceedings of the 1994 ACM/IEEE conference on Supercomputing*, pages 410–419, New York, NY, USA, 1994. ACM Press.

[CCHJ05] Peter V. Coveney, Jonathan Chin, Matthew J. Harvey, and Shantenu Jha. Scientific grid computing: The first generation. *Computing in Science and Engg.*, 7(5):24–32, 2005.

[Cer94] Paul E. Ceruzzi. From batch to interactive: The evolution of computing systems, 1957-1969. In *IFIP Congress (2)*, pages 279–284, 1994.

[CFK99] Karl Czajkowski, Ian T. Foster, and Carl Kesselman. Resource co-allocation in computational grids. In *Proceedings Eighth IEEE International Symposium on High Performance Distributed Computing (8th HPDC'99)*, Redondo Beach, California, USA, March 22 1999. IEEE Computer Society.

[CFK04] Karl Czaijkowski, Ian Foster, and Carl Kesselman. *The Grid 2. Resource and Service Management*, chapter 18, pages 259–283. Morgan Kaufman, 2nd edition, 2004.

[CGR+06] L. Cherkasova, D. Gupta, E. Ryabinkin, R. Kurakin, V. Dobretsov, and A. Vahdat. Optimizing grid site manager performance with virtual machines. In *Proc. of the 3rd USENIX Workshop on Real Large Distributed Systems (WORLDS '06)*, Seattle, 11 2006.

[Chi04] Andrew A. Chien. *The Grid 2. Computing Elements*, chapter 28, pages 567–591. Morgan Kaufman, 2nd edition, 2004.

[Con96] Condor. High throughput computing. www.cs.wisc.edu/condor/htc.html, 1996.

[EGK+07] M. Ellert, M. Gronager, A. Konstantinov, B. Kónya, J. Lindemann, I. Livenson, J. L. Nielsen, M. Niinimäki, O. Smirnova, and A. Wäänänen. Advanced resource connector middleware for lightweight computational grids. *Future Gener. Comput. Syst.*, 23(2):219–240, 2007.

[Fiu06] Marc E. Fiuczynski. Planetlab: overview, history, and future directions. *Operating Systems Review*, 40(1):6–10, 2006.

[FK97] Ian Foster and Carl Kesselman. Globus: A metacomputing infrastructure toolkit. *Intl J. Supercomputer Applications*, 11(2):115–128, 1997.

[FKNT02] I. Foster, C. Kesselman, J. Nick, and S. Tuecke. The physiology of the grid: An open grid services architecture for distributed systems integration, 2002.

[Fos01] Ian T. Foster. The anatomy of the grid: Enabling scalable virtual organizations. In *Euro-Par '01: Proceedings of the 7th International Euro-Par Conference Manchester on Parallel Processing*, pages 1–4, London, UK, 2001. Springer-Verlag.

[Fos05] Ian T. Foster. Service oriented science. *Science*, 308(5723):214–217, 05 2005.

[Fos06] Ian T. Foster. Globus toolkit version 4: Software for service-oriented systems. In *Proceedings of FIP International Conference on Network and Parallel Computing*, volume LNCS 3779, pages 2–13. Springer-Verlag, 2006.

[FPC+02] F.Carminati, P.Cerello, C.Grandi, E.Van Herwijnen, O.Smirnova, and J.Templon. Lhc grid computing project: Common use cases for a hep common application layer. Technical report, HEPCAL, 11 2002.

[FTF+02] James Frey, Todd Tannenbaum, Ian Foster, Miron Livny, and Steve Tuecke. Condor-G: A computation management agent for multi-institutional grids. *Cluster Computing*, 5:237–246, 2002.

[GCC+04] Greg Graham, Richard Cavanaugh, Peter Couvares, Alan De Smet, and Miron Livny. *The Grid 2. Distributed Data Analysis: Federated Computing for High-Energy Physics*, chapter 10, pages 136–145. Morgan Kaufman, 2nd edition, 2004.

[GLMR07] Tristan Glatard, Diane Lingrand, Johan Montagnat, and Michel Riveill. Impact of the execution context on Grid job performances. In *International Workshop on Context-Awareness and Mobility in*

Grid Computing (WCAMG'07), pages 713–718, Rio de Janeiro, May 2007. IEEE.

[GLY00] Jean-Pierre Goux, Jeff Linderoth, and Michael Yoder. Metacomputing and the master-worker paradigm. Technical report, Argonne National Labs, October 17 2000.

[GMP06] Tristan Glatard, Johan Montagnat, and Xavier Pennec. Probabilistic and dynamic optimization of job partitioning on a grid infrastructure. In *14th euromicro conference on Parallel, Distributed and network-based Processing (PDP06)*, pages 231–238, Montbéliard-Sochaux, France, February 2006.

[GMP07] Tristan Glatard, Johan Montagnat, and Xavier Pennec. Optimizing jobs timeouts on clusters and production grids. In *International Symposium on Cluster Computing and the Grid (CCGrid'07)*, pages 100–107, Rio de Janeiro, May 2007. IEEE.

[GPJ+07] Xavier Grehant, Olivier Pernet, Sverre Jarp, Isabelle Demeure, and Peter Toft. Automatic xen management with smartfrog to supply on-demand computing resource. In *VHPC '07: Proceedings of the Workshop on Virtualization in High-Performance Cluster and Grid Computing*, LNCS. Springer, 08 2007.

[Gra] *Globus Resource Allocation Manager (GRAM) 1.6 documentation.*

[GSM+07] Tristan Glatard, Gergely Sipos, Johan Montagnat, Zoltán Farkas, and Péter Kacsuk. *Workflow Level Parametric Study Support by MOTEUR and the P-GRADE Portal*, chapter 18, pages 279–299. Springer-Verlag, 2007.

[Har03] Aaron Harwood. *Networks and Parallel Processing Complexity.* Melbourne School of Engineering, Department of Computer Science and Software Engineering, 2003.

[HGLS86] W. Daniel Hillis and Jr. Guy L. Steele. Data parallel algorithms. *Communications of the ACM*, 29(12):1170–1183, 1986.

[HSLL00] Elisa Heymann, Miquel A. Senar, Emilio Luque, and Miron Livny. Adaptive scheduling for master-worker applications on the computational grid. In Rajkumar Buyya and Mark Baker, editors, *Proceedings of Grid Computing - GRID 2000, First IEEE/ACM International Workshop*, volume 1971 of *Lecture Notes in Computer Science*, pages 214–227, Bangalore, India, December 2000. Springer.

[Hum06] Michael Humphrey. Altair's PBS - altair's PBS professional update. In *SC*, page 28. ACM Press, 2006.

[J.T06] J.T.Moscicki. Efficient job handling in the grid: short deadline, interactivity, fault tolerance and parallelism. In *EGEE User Forum*, Geneva, Switzerland, March 2006. CERN.

[KDF04] Katarzyna Keahey, Karl Doering, and Ian Foster. From sandbox to playground: Dynamic virtual environments in the grid. In *GRID '04: Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing (GRID'04)*, pages 34–42, Washington, DC, USA, 2004. IEEE Computer Society.

[LHA+04] E. Laure, F. Hemmer, A. Aimar, M. Barroso, P. Buncic, A. Di Meglio, L. Guy, P. Kunszt, S. Beco, F. Pacini, F. Prelz, M. Sgaravatto, A. Edlund, O. Mulmo, D. Groep, S.M. Fisher, and M. Livny. Middleware for the next generation grid infrastructure. In *Proceedings of Computing in High Energy Physics*, Interlaken, Switzerland, 09 2004.

[LHC+04] L.Maigne, D. Hill, P. Calvat, V. Breton, R. Reuillon, D.Lazaro, Y. Legr, and D. Donnarieix. Parallelization of monte carlo simulations and submission to a grid environment. *Parallel Processing Letters journal*, 14(2):177–196, June 2004.

[LHC+06] Hurng-Chun Lee, Li-Yung Ho, Hsin-Yen Chen, Ying-Ta Wu, and Simon C. Lin. Efficient handling of large scale in-silico screening using diane. In *Poster in EGEE'06 Conference, Enabling Grids for E-Science*, Geneva, Switzerland, 2006.

[Lit07] Maarten Litmaath. glite job submission chain v.1.2. litmaath.home.cern.ch/ litmaath/UI-WMS-CE-WN, June 2007.

[LSJ+06] Hurng-Chun Lee, Jean Salzemann, Nicolas Jacq, Hsin-Yen Chen, Li-Yung Ho, Ivan Merelli, Luciano Milanesi, Vincent Breton, Simon C. Lin, and Ying-Ta Wu. Grid-enabled high-throughput in silico screening against influenza a neuraminidase. In *Proceedings of NETTAB 2006*, Santa Margherita di Pula, July 10-13 2006.

[LSV06] Arnaud Legrand, Alan Su, and Frederic Vivien. Minimizing the stretch when scheduling flows of biological requests. In *SPAA '06: Proceedings of the eighteenth annual ACM symposium on Parallelism in algorithms and architectures*, pages 103–112, New York, NY, USA, 2006. ACM Press.

[Mac04] Marlon Machado. Enable existing applications for grid: Batch anywhere, independent concurrent batch, and parallel batch. Technical report, IBM, June 2004.

[MB03] Jeremiah Mans and David Bengali. Blueox: A java framework for distributed data analysis. In *Proceedings of Computing in High Energy Physics*, La Jolla, Ca, USA, March 2003.

[MDP+00] Dejan S. Milojíčić, Fred Douglis, Yves Paindaveine, Richard Wheeler, and Songnian Zhou. Process migration. *ACM Computing Surveys*, 32(3):241–299, 2000.

[MFRW06] Grzegorz Malewicz, Ian Foster, Arnold L. Rosenberg, and Michael Wilde. Tool for prioritizing dagman jobs and its evaluation. In *Proceedings of the 15th IEEE Intl. Symp. on High-Performance Distributed Computing (HPDC '06)*, pages 156–167, 2006.

[MHS+04] J.T. Moscicki, H.C.Lee, S.Guatelli, S.C. Lin, and M.G.Pia. Biomedical applications on the grid: Efficient management of parallel jobs. In *NSS*, Rome, Italy, October 2004. IEEE.

[Miu06] Kenichi Miura. Overview of japanese science grid project naregi. *Progress in Informatic*, 3(1349-8614):67–75, 2006.

[Mos03] J.T. Moscicki. Diane - distributed analysis environment for grid-enabled simulation and analysis of physics data. In *NSS*, Portland, Oregon, USA, October 2003. IEEE.

[Nil07] P. Nilsson. Experience from a pilot based system for atlas. In *Proceedings of Computing in High Energy Physics (CHEP'07)*, Victoria, Canada, 2007.

[NLJ+05] A. Nishandar, D. Levine, S. Jain, G. Garzoglio, and I. Terekhov. Extending the cluster-grid interface using batch system abstraction and idealization. In *Proceedings of Cluster Computing and Grid 2005 (CCGrid05)*, Cardiff, UK, May 2005.

[NYI+05] Hidemoto Nakada, Motohiro Yamada, Yasuyoshi Itou, Satoshi Matsuoka, Jaime Frey, and Yasumasa Nakano. Design and implementation of condor-unicore bridge. In *HPCASIA '05: Proceedings of the Eighth International Conference on High-Performance Computing in Asia-Pacific Region*, page 307, Washington, DC, USA, 2005. IEEE Computer Society.

[PDD05] James Padgett, Karim Djemame, and Peter Dew. Grid service level agreements combining resource reservation and predictive run-time adaptation. In *Proceedings of the UK e-Science All Hands Meeting*, Nottingham UK, September 19th - 22nd 2005.

[Pen02] Rob Pennington. Terascale clusters and the teragrid. In *Proceedings for High Performance Computing Asia*, pages 407–413, Dec 16-19 2002.

[PSP06] Stuart Paterson, P. Soler, and C. Parkes. *LHCb Distributed Data Analysis on the Computing Grid*. PhD thesis, University of Glasgow, 2006.

[PT06] S. Paterson and A. Tsaregorodtsev. Dirac infrastructure for distributed analysis. In *Proceedings of Computing in High Energy Physics*, Mumbai, India, February 2006.

[PW07] Sanjay Padhi and Rodney Walker. Cronus: A condor glide-in based atlas production executor. In *Proceedings of CHEP'07*, September 2007.

[Reb05] David Rebatto. Egee batch local ascii helper (blahp). In *HEPiX Meeting*, Karlsruhe, Germany, May 2005.

[RLS98] Rajesh Raman, Miron Livny, and Marvin Solomon. Matchmaking: Distributed resource management for high throughput computing. In *Proceedings of the Seventh IEEE International Symposium on High Performance Distributed Computing*, Chicago, IL, July 28-31 1998.

[RMdL04] Daniel A. Reed, Celso L. Mendes, and Charng da Lu. *The Grid 2. Application Tuning and Adaptation*, chapter 26, pages 513–532. Morgan Kaufman, 2nd edition, 2004.

[RSZ+06] A. Ramakrishnan, G. Singh, H. Zhao, E. Deelman, R. Sakellariou, and K. Vahi. Scheduling data-intensive workflows onto storage-constrained distributed resources. In *Proceedings of the Seventh IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2007)*, April 2006.

[Rud01] M. Ruda. Integrating grid tools to build a computing resource broker: activities of datagrid wp1. In *Proceedings of Computing in High Energy Physics (CHEP 2001)*, Beijing, September 3-7 2001.

[SAB+03] P. Saiz, L. Aphecetcheb, P. BunImageiImagea, R. PiskaImaged, J. E. Revsbeche, and V. Imageegod. Alien - alice environment on the grid. *Nucl. Instrum. Meth.*, A502:437–440, 2003.

[SAB+05] Andreas Savva, Ali Anjomshoaa, Fred Brisard, Michel Drescher, Donal Fellows, An Ly, Stephen McGough, and Darren Pulsipher. Job submission description language (jsdl) specification. http://forge.gridforum.org/projects/jsdl-wg, November 2005. GFD-R.056.

[SBP03] Pablo Saiz, Predrag Buncic, and Andreas J. Peters. Alien resource brokers. In *Proceedings of CHEP'03*, June 2003.

[Sfi07] Igor Sfiligoi. glideinwms - a generic pilot-based workload management system. In *Proceedings of Computing in High Energy Physics (CHEP'07)*, 2007.

[Sto07] Heinz Stockinger. Defining the grid: a snapshot on the current view. *J. Supercomput.*, 42(1):3–17, 2007.

[Ter02] I. Terekhov. Meta-computing at d0. *Nuclear Instruments and Methods in Physics Research (ACAT-02)*, 502/2-3(NIMA14225):402–406, June 2002.

[tGC06] the GridPP Collaboration. Gridpp: development of the uk computing grid for particle physics. *Journal of Physics G: Nuclear and Particle Physics*, 32:N1–N20, 2006.

[TGSR04] Andrei Tsaregorodtsev, Vincent Garonne, and Ian Stokes-Rees. Dirac: A scalable lightweight architecture for high throughput computing. In *GRID '04: Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing*, pages 19–25, Washington, DC, USA, 2004. IEEE Computer Society.

[TTL02] Douglas Thain, Todd Tannenbaum, and Miron Livny. Condor and the grid. In Fran Berman, Geoffrey Fox, and Tony Hey, editors, *Grid Computing: Making the Global Infrastructure a Reality*. John Wiley & Sons Inc., December 2002.

[vHCF+03] Eric van Herwijnen, Joel Closier, Markus Frank, Clara Gaspar, Francoise Loverre, Sebastien Ponce, Roberto Graciani Diaz, Domenico Galli, Umberto Marconi, Vincenzo Vagnoni, Nicholas Brook, K. Buckley, A.and Harrison, Michael Schmelling, Ulrik Egede, Andrei Tsaregorotsev, V. Garonne, B. Bogdanchikov, Ivan Korolko, Juan P. Washbrook, A.and Palacios, Sander Klous, Juan J. Saborido, Akram Khan, A. Pickford, A. Soroko, V. Romanovski, G.N. Patrick, Genady Kuznetsov, and Miriam Gandelman. Dirac - distributed infrastructure with remote agent control. In *Proceedings of Computing in High Energy Physics*, 2003.

[WDRT97] Philip M. Williams, Martyn C. Davies, Clive J. Roberts, and Saul J. B. Tendler. Data analysis using the internet: the world wide web scanning probe microscopy data analysis system. *Analyst*, 122:1001–1006, October 1997.

[Wei98] Jon B. Weissman. Metascheduling: A scheduling model for metacomputing systems. In *Proceedings of High Performance Distributed Computing (HPDC'98)*, pages 348–349, 1998.

[WLrW06] Torre Wenaus, Miron Livny, and rank Wrthwein. Preliminary plans for just-in-time workload management in the osg extensions program. Technical report, US Atlas, October 2006. based on SAP proposal of March 2006.

[XLT+05] Wei Xiaohui, Wilfred W. Li, Osamu Tatebe, Xu Gaochao, Hu Liang, and Ju Jiubin. *Parallel and Distributed Processing and Applications - Integrating Local Job Scheduler - LSF with Gfarm*, high-performance computing and architecture i 2A, pages 196–204.

Number 3758 in Lecture Notes In Computer Science. Springer, 2005.

[XZQ00] Xiao, Zhang, and Qu. Effective load sharing on heterogeneous networks of workstations. In *IPPS: 14th International Parallel Processing Symposium*, pages 431–438, Los Alamitos, May 1–5 2000. IEEE Computer Society Press.

[Zha02] Yong Zhao. Virtual galaxy clusters: An application of the griphyn virtual data toolkit to sloan digital sky survey data. Technical Report TR-2002-6, GriPhyN, 2002.

[ZZWD93] S. Zhou, X. Zheng, J. Wang, and P. Delisle. Utopia: a load sharing facility for large, heterogenous distributed computer systems. *Software: Practice And Experience*, 23(12):1305–1336, December 1993.

Dépôt légal : 2008 – 1er trimestre
Imprimé à l'Ecole Nationale Supérieure des Télécommunications – Paris
ISSN 0751-1345 ENST D (Paris) (France 1983-9999)

Ecole Nationale Supérieure des Télécommunications
Groupe des Ecoles des Télécommunications - membre de ParisTech
46, rue Barrault - 75634 Paris Cedex 13 - Tél. + 33 (0)1 45 81 77 77 - www.enst.fr
Département INFRES