Grads, Accounting and High-Performance Computing

Lennart Johnsson
Department of Computer Science,
Mathematics, and Electrical and Computer Engineering
University of Houston

June 2007

Abstract. The rapidly increasing capabilities of networks enable the construction of wide-area integrated environments for collaboration and sharing of instruments, data, and computational and storage resources. We will briefly present some current Grid infrastructure developments, some of the associated challenges, and some of our own research in regards to standards based non-intrusive Grid accounting, and techniques for design and implementation of scientific software libraries achieving portability across platforms with preservation of efficiency to the extent possible.

Introduction

During the last decade the concept of Grids has evolved from a concept to early production environments. Grids have become pervasive in that Grids in some form now exist in almost all networked nations. Grids as we know them today have evolved from the I-WAY [42,43] showcase of the vision of what computing in a networked world could be like. The I-WAY showcase was conceived by Tom DeFanti, Director of the Electronic Visualization Laboratory (EVL) at the University of Chicago, inventor of the CAVE technology and founder of the SIGGraph conference series, Larry Smarr, astrophysicist and Director of the National Center for Supercomputer Applications (NCSA), and Rick Stevens computer scientists at the Mathematics and Computer Sciences Division of Argonne National Laboratory, for the Supercomputing '95 conference and exhibit. The showcase involved 17 organizations, 11 computer networks across the US, and about 60 applications. The software development necessary to make applications work in the I-WAY distributed environment, was developed under the leadership of Ian Foster and Carl Kesselman, the well recognized leaders of the Globus Grid middleware.

The word Grid for this form of distributed computing was made because of the analogy made to Grids for energy production, transmission, distribution and consumption in which consumers (in today’s world with universally adopted standards) do not need to know anything about where the electricity is produced and transported to the point of usage, or how that process is managed. This vision for computing was already present among the founders of the ARPA-net. Leonard Kleinrock in 1969 stated “We will perhaps see the spread of ‘computer utilities’, which, like present electric and telephone utilities, will service individual homes and offices across the country”, and in fact even earlier than that. John McCarthy at the MIT 1961 Centennial stated “If computers of the kind I have advocated become the computers of the future, then computing may someday
be organized as a public utility just as the telephone system is a public utility... The computer utility could become the basis of a new and important industry.”

The main enabling factor for the emergence of Grids is the wide spread deployment of high-performance networks, i.e. local area, metropolitan area and wide are networks (LANs, MANs and WANs) that have capabilities similar to those of system area networks or even the networks in parallel computer clusters. This leveling of the communication capabilities has been observed by many and the impact on technology and working environments forecasted by many as well. For instance, George Gilder made the statement “When the network is as fast as the computer's internal links, the machine disintegrates across the net into a set of special purpose appliances” and Ian Foster and Rick Stevens observed that the improved communication capabilities eliminate the “tyranny of distance” (Foster, Stevens), and Thomas L Friedman’s published a book entitled “The World is Flat”[44]. In fact, the capabilities of communication in WANs in particular have increased exponentially at about double the rate of the familiar Moore’s law. Larry Roberts, one of the ARPAnet pioneers, about a decade ago illustrated this as shown below

Another often used illustration is one that appeared in Scientific American in January 2001.
Much of the improvement in WAN capabilities comes from the use of so-called Dense Wave Division Multiplexing (DWDM) technology that became possible once production quality lasers of different colors became available. That allowed a single strand of fiber to carry many concurrent light paths, thus increasing the capabilities several fold in addition to improvements in speed of controlling individual light paths. The idea of the DWDM technology is illustrated below.

![Wave Division Multiplexing (WDM)](image)

The impact on WANs is illustrated by the result of a 2004 survey [16].

![Number of Backbones Worldwide Operating at Various Gbps Rates by Year](image)

Source: Lennart Johnsson

In conclusion, for the foreseeable future communication capabilities are expected to improve at a higher rate than computing and storage capabilities, and thus the hardware technologies will continue to evolve in a direction that makes the creation of distributed computing, information and collaboration environments increasingly appealing and in many cases highly desirable because team members are geographically distributed, data sources are remote in reference to the processing capabilities, or to the users, or both.
Status of Grid Deployment

A few years ago some surveys of Grid deployment were made. Today, Grids have become so widespread that it is difficult to make a survey that would be complete and accurate. We choose to give a few examples with the intent being to show extent of deployment of some grid efforts as well as diversity.

The probably largest Grid effort is EGEE, enabling Grid for e-Science. As of May 2007, the EGEE production Grid had over 35,000 CPUs at 237 sites in 45 countries [11]. The rapid acceptance of the environment used in EGEE is well illustrated in the graphs below showing a 12-fold increase in CPUs in less than three years, and about a 6-fold increase in the number of sites over the same time period. The number of jobs is also growing at a good rate. The 98,000 jobs/day corresponds to about 40% of available resources. In addition to High-Energy Physics application areas using EGEE include Astrophysics, Computational Chemistry, Earth Sciences, Financial modeling, Fusion physics, Geophysics, Life Sciences, Multi-media and Material Sciences.

The US NSF funded TeraGrid [13] has a fairly limited number of sites. Initially it was focused on NCSA and SDSC for production resources and ANL and Caltech for R&D type systems and efforts. TeraGrid has since added a few sites (e.g., Indiana University, Purdue University, PSC, TACC and ORNL) to comprise seven universities and two government laboratories. In terms of the number if CPUs it is about half the size of the EGEE production Grid, and so is the number of users, about 10,200 for EGEE in February 2007 and about 4,500 for TeraGrid.

Dane Skow (dds@uchicago.edu) February 2007
DEISA, Distributed European Infrastructure for Supercomputer Applications [45], is like TeraGrid focused on creating a distributed infrastructure primarily for users of High-End systems. DEISA at the moment comprises 11 sites in Europe with a capability that by the end of 2007 is expected to reach 400 TF.

The Open Science Grid, OSG [14], is a joint effort of NSF and DoE with a focus very similar to EGEE, i.e. providing an infrastructure for the Large Hadron Collider (LHC) Computing Grid is the primary driving application, but not exclusively. Thus, both EGEE and OSG are primarily focused on an infrastructure with resources suitable for throughput computing. The OSG and EGEE sites are shown below together with a snapshot of the OSG usage.

Of other Grid efforts with international scope PRAGMA (Pacific Rim Applications and Grid Middleware Assembly) [15] has a large number of participating institutions. In late 2005 there were about 600 CPUs in total available at 19 sites. The sites indeed to cover the Pacific Rim as shown below.

Of national Grid efforts the Japanese Naregi [17,32] effort should be mentioned because it both is establishing a Grid in Japan that has diverse resources, i.e., both what often is referred to as capability and capacity computing, and aims to develop and deploy commercial quality middleware.
There are many other Grid efforts that are providing useful services to scientists and engineers, such as M-Grid [18] (Material Science) in Finland, EMBRACE Grid [19] (bioinformatics), BIRN [20], and caBIG [21] to mention a few domain specific Grids, or national Grids such as the UK National Grid Service [22], the German D-Grid [23], Baltic Grid [24], the Italian INFN Grid [25], Hellas Grid (Greece), SEE-Grid [26], the K-Grid of South Korea [27], the National Grid Singapore [28], etc. The examples given should hopefully be sufficient though to convince that Grid technologies have matured to the point that useful services are now being provided to some groups.

Challenges

To create a Grid environment phases many challenges. Grids for research and educational communities typically include resources and users from several administrative domains. These domains are rarely uniform but often heterogeneous in almost every aspect: hardware, operating systems and other forms of system software, account management and user policies, systems administration, application software, etc. Thus, in creating Grids both policy and technical issues in regards to security and interoperability has to be addressed. Standards become a critical issue, like in the creation of the internet and in creating global communications infrastructures. Another set of issues are related to application software and data. Applications should ideally run on any platform in a Grid, i.e., on a wide range of architectures and operating systems, and better still run as efficiently as possible on the various platforms. For data, format and semantics are crucial to sharing of information. This is clearly not unique to Grids, but through Grids sharing should be simplified and thus the usage of data can be expected to grow both within communities and across communities, and thus the pressure on data access being ubiquitous is likely to increase significantly.

The collection of software used to create Grids is typically referred to as middleware because it is layered on top of traditional system software, but provide services to applications whether those are end-user applications or applications used for monitoring, accounting or auditing. Globus [29], Legion [30] and Unicore [31] (Uniform Interface to
Computing Resources) were well known early efforts establishing middleware, with Globus and Unicore and derivates thereof today being the dominant middlewares.

Because of the central role of interoperability for Grids the community developing middleware engaged heavily in establishing standards. The community founded the US Grid Forum and the European Grid Forum that merged to form the Global Grid Forum, that intern merged with the Enterprise Grid Alliance to form the Open Grid Forum [34]. The Grid standardization effort initiated by the research community has been taking a service oriented approach in recent years and adopting and extending web services standards to accommodate the notion of “state”. The Open Grid Services Architecture (OGSA) [35] is a result of this development. The OGSA services can be grouped into information services, resource management services, execution management services, data services, self-management services and security services as illustrated below.

Of the challenges in Grid computing we will briefly describe our efforts in regards to accounting and in regards to application code portability with preservation of execution efficiency.
**Grid Accounting**

In Grids, management of rights and privileges for users is simplified by creating so-called Virtual Organizations (VOs) [36] with most rights and privileges being common for all members of a VO. A user can belong to more than one VO. Different VOs can have access rights to different sets of resources within a Grid. Our approach to Grid accounting, SGAS [1,4], is Web services based, recognizes VOs and is non-intrusive, i.e., the system is designed with the notion that different administrative domains may have their own accounting procedures as well as user policies. The role of SGAS is shown below:

SGAS has three main services: the bank, logging and usage tracking service (LUTS), and job account reservation manager (JARM). Their relationship is shown below:

The bank service manages resource quota on a VO and user basis. A bank account is created for each VO that has resources to spend for usage of resources recognizing the VO. The VO manager can add users allowed to submit jobs whose resource usage can be charged to the VO account. The bank service can be queried for available funds in an account, and *holds* of parts of the funds can be issued and then charged. At resource reservation time the bank is queried to assure that sufficient funds are available to execute the job on a feasible resource selected by a resource selection manager. If sufficient funds are available a hold is created for the required funds. On successful termination of the execution the actual usage/cost is charged to the account and the hold released. The bank...
service is implemented as three Web Services Resource Framework (WSRF) [37] compliant services: Bank service, Account and Hold. The bank service is used in creating new accounts, while the account service is used in managing an existing account. The hold service is used to manage successful requests for services. The created hold belongs to the requesting resource. The relationship between the components of the bank are shown below where operations are shown as methods and properties as attributes.

The Job Account Reservation Manager (JARM) integrates local resources into the VO-wide accounting context. The globally scoped VO accounts are enforced collectively by the JARMs which intercept job submissions through a pre-execution call-out from the Grid submission software. The call-out is handled by the JARM reservation manager that 1) finds an account for the submitter by searching the bank if not explicitly provided by the job, 2) estimate the job cost based on provided estimate of execution time on a particular type of resource, 3) checks whether the job is compliant with local policies for execution, and 4) acquires a hold on the account corresponding to estimated cost, possibly with some margin. Holds are time limited to assure that holds are released if jobs fail to execute to completion in a reasonable amount of time. When the job finishes, the workload manager makes a post-execution call-out to JARM, which notifies the commit manager of job completion. The commit manager then 1) collects usage data for the job in the environment-specific format, 2) transforms it into a standard OGF usage record and reports it to the LUTS, and 3) based on job usage, calculates the actual job cost and charges the VO account. Any residual hold amount is released.

The Logging and Usage Tracking Service (LUTS) provides a Web service interface for 1) publishing of usage data in OGF’s Usage Record format, and 2) for query-based retrieval of usage data using the XPath [38] query language. LUTS off-loads the bank from storing detailed logging and auditing records. Those records can be queried by all stakeholders, e.g. for off-line accounting after the jobs have run, or for making allocation and authorization decisions based on previous history. The LUTS service interface is designed for the document-centric operations provided by the WS Resource Properties portTypes. SGAS uses a native XML-database (eXist) for persistent storage and
recoverability of usage records. The LUTS query expression evaluator, redirects the embedded XPath query to the back-end XML database, effectively exposing a database view through the service interface. The same mechanism is used to query account transaction logs.

The extension-points defined in the Usage Record specification together with the schema-agnostic LUTS storage of usage records (usage record documents are stored “as is” in the XML database) facilitate extensibility, allowing sites to publish custom usage record elements, without modifying the LUTS. The Logging and Usage Tracking Service allows off-line accounting after the jobs have run, and off-loads the bank from storing detailed logging and auditing records. The LUTS can be queried by all stakeholders as a means to making allocation and authorization decisions based on previous history.

SGAS offers a flexible and interoperable end-to-end security solution through the use of standard security mechanisms and a fine-grained, highly customizable authorization framework. Message-level security is supported via WS-Security and WS-SecureConversation and transport-level security via TLS. The authorization process is independent of the service requested, which allows run-time administration of XML-based authorization policies through a Web service interface, with plug-in support for different back-end authorization engines. The current back-end is based on the eXtensible Access Control Markup Language (XACML) [39] and allows fine-grained access policies to be set up for services and WS-Resources. SGAS uses credential delegation and single sign-on to allow JARM to transparently reserve allocation and charge usage on behalf of the user. Resource owners may configure a set of trusted banks with JARM in order to prevent users from supplying (through the job description) a bogus account in a fake bank. The bank does not need to be configured to explicitly trust resources, since resources always act on behalf of users (using delegated credentials). The security solution is illustrated below.

The SGAS has been adopted by Globus Toolkit version 4 (GT4) and is now also being considered by OGF for standardization and by Grid efforts such as EGEE and the Baltic Grid for deployment.

Source: Thomas Sandholm
In a Grid a resource broker should be able to find the best possible resource, according to some criteria, for an application. To make the feasible resource pool as large as possible it is important that the application can execute on a wide range of platforms, and efficiently so. Efficient execution on a platform often requires not only recompilation of the code for a particular platform, but some modification of algorithms or code. For well defined functions, such as mathematical software libraries, a few different approaches have been pursued. About 20 years ago adapting library functions to different degrees of parallelism and different data distributions was made in the Connection Machine Scientific Software Library (CMSSL) [40] by having several different algorithms and for each possibly several different implementations for a “high-level” function call. At runtime, when the degree of parallelism and data distributions were known a code that minimized the expected execution time was chosen transparently to the user based on performance models. Later in the ATLAS [46] project a brute-force approach was taken by benchmarking “all possible” blockings, loop interchanges etc for library functions and then selecting the one with the minimum run-time. A more sophisticated approach has been taken in the FFTW [8,9] and SPIRAL [7,10] efforts which went one step further than the approach used in CMSSL in that final code for execution was produced at run-time instead of being selected at run-time. But, as in CMSSL, the function interface makes the choice of algorithm and code for execution transparent to the user. The UHFFT [2,3,5,6] effort that we describe briefly here has the same overall concept as initiated with CMSSL, and the same high-level implementation strategy as FFTW, but differs in regards to some of the algorithms supported, and in optimization procedures and implementation language.

SPIRAL is a code generator system that generates optimized codes for specific size transforms. The UHFFT and FFTW systems have two phases, a library installation phase in which automatic code generation is used for highly optimized small size FFTs, and a run-time framework in which the small size FFTs are combined for sizes not directly covered. In the UHFFT the code generator generates straight line C code blocks called codelets, while FFTW use CAML [41]. A block diagram of UHFFT run-time is given below.
Each FFT problem is identified by a DFTI [47] descriptor, which describes various characteristics of the problem, i.e., size, precision, placement, input and output data type, number of threads etc. The Planner seeks to find an execution order, plan, that minimizes the execution time. Our current implementation supports two strategies for searching for the “best” plan. Both strategies use the dynamic programming to search the space of possible factorizations and algorithms. Context Sensitive Empirical Search empirically evaluates sub-plans. To avoid re-evaluation of identical sub-plans, a lookup table is maintained to store their performance. Context Free Hybrid Search reduces the cost of search is significantly, possibly at the expense of the quality of plans found. In this scheme, the cost of a sub-plan is estimated by empirically evaluating only the codelets that are encountered in a bottom up traversal.

An execution plan determines the codelets that will be used for the computation and also the order (schedule) in which they will be executed. A UHFFT plan is described in concise language using a grammar as shown below. It allows different algorithms to be mixed together to generate a high performance execution plan based on properties of the input vector and its factors. Indeed, by implementing a minimal set of rules, adaptive schedules can be constructed that suit different types of architectures.

<table>
<thead>
<tr>
<th>#</th>
<th>CFG Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-2</td>
<td>ROOT → MULTID _FFT</td>
</tr>
<tr>
<td>3-4</td>
<td>MULTID _FFT → FFT</td>
</tr>
<tr>
<td>5-6</td>
<td>NDFFT → NDFFT , FFT</td>
</tr>
<tr>
<td>7</td>
<td>SMFFFT → ( mр Z , MULTID _FFT ) Z</td>
</tr>
<tr>
<td>8-9</td>
<td>FFT → FFT nr MODULE</td>
</tr>
<tr>
<td>10-11</td>
<td>MODULE → CODELET</td>
</tr>
<tr>
<td>12</td>
<td>( rnder , FFT ) Z</td>
</tr>
<tr>
<td>13</td>
<td>ORDERED _FFT → ( inplaceZ , FFT )</td>
</tr>
<tr>
<td>14</td>
<td>( outplaceZ , FFT )</td>
</tr>
<tr>
<td>15</td>
<td>( pfaZ , PEAFFFT pfa , ROT _CODELET )</td>
</tr>
<tr>
<td>16</td>
<td>CODELET → n ∈ DFT codelets</td>
</tr>
<tr>
<td>17</td>
<td>ROT _CODELET → n ∈ Rotated DFT codelets</td>
</tr>
<tr>
<td>18</td>
<td>BLOCK → b Z ; Z</td>
</tr>
</tbody>
</table>

**FFT Schedule Specification Language (FSSL)**

The performance sensitivity to plan choice can be quite significant. We illustrate this for a codelet of size 16. As seen from the graph the performance difference depending on plan is more than a factor 10. For larger transforms the performance difference may not be quite as dramatic, but still significant as shown in the case of a transform of size 2520 on the same platform. Since the particular codelet implementation and selection and
execution order of codelets for best performance can be platform dependent for a specific size the objective of the adaptive approaches of UHFFT and FFTW are to remove the burden of this optimization from the user by automating it.

Another important aspect is the impact of strides. This is illustrated below for some by now old platforms, the Pentium 4, AMD Athlon and the PowerPC G4. In some further detail we show the stride significance for the Itanium-1, Itanium-2 and Ultrsparc.
Finally some comparisons with FFTW

Summary

We have presented a brief overview of current Grid efforts and some of the challenges in making Grids pervasive. We have also given a brief overview of two of our research efforts related to Grids, one in regards to distributed accounting across different administrative domains, one in regards achieving performance portability across platforms.

Acknowledgements

Much of this work has been carried out by graduate students, in particular Thomas Sandholm in regards to SGAS, and in regards to the UHFFT Ayaz Ali, Purvi Shah, Haiyang Teng, Fredrik Mwandia, and Rishad Mahasoom, and Research Scientist Dragan Mirkovic. The work has been supported by Intel Corp, NSF under the GrADS and VGrADS projects and the National Computational Science Alliance. The SGAS development was in part supported by the Swedish Research Council.

References


11. I Bird, EGEE User Forum, May 9 – 11, Manchester, UK


22. http://www.grid-support.ac.uk/


27. http://www.gridcenter.or.kr/


34. http://www.ogf.org/
38. “XML Path Language (XPath)”, http://www.w3.org/TR/xpath
45. http://www.deisa.org