

Computational grids

Summary

What types of applications will grids be used for?

There are many different purposes of using computational grids and many different problems that need different underlying technology. Those different approaches can be classified into five classes, which are distributed supercomputing, high throughput, on demand, data intensive and collaborative computing.

Who will use computational grids?

There are many different groups in society that could benefit from a grid. Communities like government, health maintenance, science collaborations and many others all need the ability to share data and CPU power. Since there are so many areas that grids could be used in, we don't expect to see only one grid architecture, but many different.

What is involved in building a grid?

This totally depends on what the grid is going to be used for, giving a single answer is just not possible. We divide the grids into four main groups ordered by scale. End systems, Clusters, Intranets and Internets

What approaches are needed to develop computational grids?

One needs to divide the computational grid development in at least three levels. Those levels are like the protocol layers that build up the World Wide Web today. Where each layer is standardised so it becomes easy to develop new application. The developers of the layers can be grid developers, tool developers and application developers.

What is needed for computational grids to become a service everyone uses?

The computational grid development needs to be standardised to make it robust, effective and easy to use. Creating possibilities for new applications to be produced effectively and cheap.

Introduction to computational grids

Some people say that computational grids are on its way to become just as natural as electricity. End users just use the computational power, but don't know (or care) how it works. Allowing people, without any knowledge, to share advanced things like CPU time, storage and algorithms just by plugging in their computer-device forces the service to be simple. Things like how the technology works and where the resources are located must be hidden from the end users. Later on we discuss more about why this has not yet been implemented.

Many research projects requires lot of CPU time, some requires a lot of memory and some projects need the ability to communicate in real time. Today super computers are not enough to solve those needs. They don't have the capacity, even if they did, it would no be economical justifiable to use these resources.

Computational grids are the solution to all these problems and many more. They offer a convenient way to connect many devices (e.g., processors, memory and IO-devices) so that end users is able, if allowed, to use all devices computational powers combined for a certain amount of time. For example

if a researcher needs to make some very CPU consuming calculations, he could occasionally borrow CPU-time from a grid to a much lower cost than borrow the time from a super computer. A grid could be created in all environments where end users have a computer with memory and CPU. Computers are often in idle state and over time they only use 5 % of their capacity. Hence there is a lot of computational power going to waste, set up as a computational grid each end user would be able enjoy a lot of computational power that otherwise would have gone to waste.

What types of applications will grids be used for?

Different problems must be tackled in different ways. Some problems are easy to convert into sub problems while other must use other underlying technologies and advanced algorithms. Also the reason behind using grids can differ widely. There are five application classes for computational grids, which will be discussed separately.

Distributed supercomputing – is the first application class and it is used to tackle problems that needs much more computational performance than any supercomputer can give. The problems they solve are distributed interactive simulation, high-resolution chemistry simulation, climate modelling and much more. Here the approach for computational grids to work is firstly to coschedule expensive resources. Because supercomputers are very expensive and often used by different groups the simulation must be planned well to work. Secondly those problems can be hard to split to subprocesses because they are not always independent. Much effort must therefore be done to construct advanced algorithms that split the problems into subprocesses, to be solved in different supercomputers. Those algorithms must also be tolerant to latency since the supercomputers are not at the same geographical location, distance creates flaws in communication. Finally protocols must be constructed to make the different systems to work properly together.

High-throughput computing - is used for problems that are loosely coupled in contradiction for the problems in distributed supercomputers. Those problems are therefore much easier to split up into subprocesses, which can be independently solved by thousands of ordinary personal computers. For example AMD used thousands of their computers to design their K6 and K7 processors.

On-demand computing - are used to tackle short-term use of resources. The main reason behind on-demand, is to save costs by sharing resources. The resources can be advanced programs to solve some task, devices like sensors and computer power. It works because every user does not need the resources all the time. The challenge to make this to work is to get a large group or groups of people to share their resources. There must therefore be good scheduling. This people may not want to share their work with others so the grid must be secure against intrusion and not leak information. The system must be fault tolerance otherwise people cannot do their job, which will cost the company in less effectiveness and also lead to complain. There must also be some payment system so that those who use a resource pay for it.

Data-intensive computation - is used to synthesize very much data which is distributed geographically in databases. For example does high-energy experiments produce petabytes of data every year. To store so much data it must be distributed to different locations. There are also a lot of scientists, which need to get hold of some data around the world. Here the main challenge is the scheduling of high volumes of data through different levels of hierarchy.

The fifth application is *collaborative computing* which mainly is used to make it possible for people

around to work and interact in real-time with each other. They are often structured in virtual shared spaces, where they also share resources and data, which also are the main issue in on-demand and data-intensive grid application. But here the main challenge is to make it possible for people to interact in real-time without disturbance.

There is as we can see a lot of different reasons and problems for using computational grids. Which require different technical approaches. It will therefore require much effort to standardise the grid technology to fit every application.

Who will use computational grids?

There is the scientist doing CPU intensive calculations, but there are also many other groups that would benefit from the concept of a computational grid.

Government – is a relatively small community that would enjoy the benefits of a grid system in areas like disaster response, national defence and research. Research such as environment change and environmental clean up, is really CPU demanding and is probably in every governments best interest. A national grid could also serve as a “computational reserve” that could be used in time of crisis (e.g., calculating the effects of an earthquake). *National grids* has the disadvantage of distributing resources, this could be really hard.

Health Maintenance – could truly experience the power of grids. By connecting all the computers and machines (e.g., MRI machines and CAT scanners) in a hospital. Hospital personnel could perform operations like advanced computer aided diagnosis on mammograms on their personal computer. Life-critical applications like telerobotic surgery and cardiac monitoring could use advanced algorithms to perform a better result. The so-called *private grids* could be used in many institutions providing better performance. Private grids has the disadvantage of combining life-critical computations with other less important computations, it also has the need to integrate many low-cost technologies.

A material science collaboratory – consists of people all around the world in desperate need of sharing research data, applications, CPU time and other tools for research. This kind of grid is called a *virtual* one and is characterised by a central unifying focus, dynamic membership and a lack of central management.

Computational market economy – consist of end user systems connected with broadband connections. The *public grid* could be used to form communities of diverse interests, like financial modelling, graphics rendering and online gaming. Today applications like this exist large-scale in only in certain areas (e.g., search for extraterrestrial life and search for prime numbers). It is hard to convince people to give computational power away, but in the future we will probably see much more applications of this kind.

As we can see there is quite many different areas where grid system could be of use. Therefore we expect to see not only one grid architecture, but many. Making it a hard topic to fully cover and grasp.

What is involved in building a grid?

A grid system has to satisfy many different users with alternating needs. Therefore there is no simple answer to what is involved in building a grid architecture, it depends on what the system is going to be used for. Even though it seems highly unlikely that there will be only one grid architecture, it is still possible to identify some basic services that most grids will implement.

To decide what kind of services is needed in computational grids we look at basic services provided on traditional conventional computers. We assume that the services in traditional computers that have been effective for decades will also be desirable in computational grids. Grid systems also need new services, but we claim that this is due to the new more complex wide area environment. So what kind of services is needed? There has to be an *authentication* process, to establish the identity of the current user. An *authorization* process that decides what kind of *processes* the current user is allowed to create. Each process consists of threads executing in an own address space and each process must also be able to *communicate* with other processes. Each process acts on the creator's behalf to acquire resources (e.g. writing to disks and occupy memory). *Scheduling* that is described earlier on must be taken care of and so must *accounting* mechanisms that keep track of resource allocations. All of these, and many more, are basic services that a grid system has to implement.

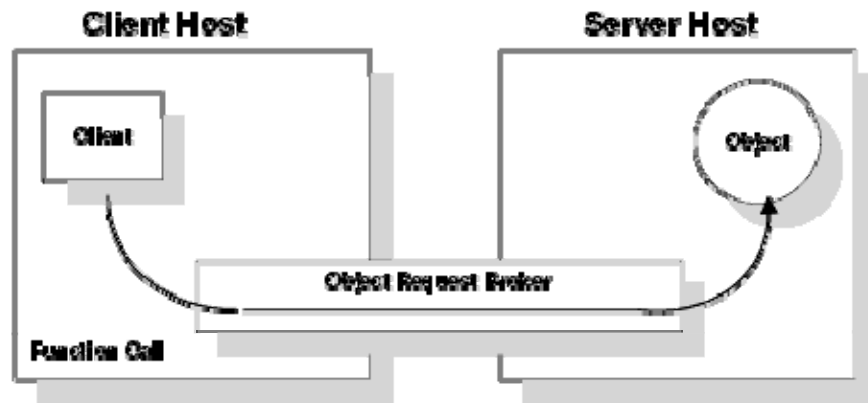
One major driver for the techniques used to implement grid services is scale. As scale increases so does the systems complexity. Hence we apply scale as a starting point to make comparisons between different systems offering the basic services. We divide the systems into 4 different groups, beginning with "end systems" and finishing off with "Internet systems".

End systems – are individual systems like ordinary computers. Characteristic traits for end systems are small scale, high degree of homogeneity and integration. The basic services are provided by the operating system that has absolute control over all resources in the computer. The integrated nature of these systems leads to high performance and effective compilers that let the user create high performance applications with relatively little effort.

Clusters – a collection of computers that is connected by a high speed local area network. A cluster is also a homogeneous entity, they differ from end systems primarily in the way that each computer has a separate configuration. Computers are controlled by a single administrator, having full control over all systems. Clusters introduce complicating factors like increased scale (many computers), making things like algorithms for resource management and control functions a must. A cluster also has reduced integration giving the disadvantage of reduced performance in areas like communication. To build a grid like this you have to consider that you cannot use the same uniform address space as you can in end systems. One approach to solve this problem is to have a logical shared memory, having software translate between local and global addresses. Systems like these are generally called DSM (Distributed Shared Memory) systems. In low performance applications we can allow user-level software to implement DSM and use TCP/IP for messaging. But if high performance is necessary fundamental changes must be made, new low-level messaging systems must be invented/used and things like coscheduling must be implemented. Not going further into this, we can conclude that the complexity of new approaches and services tends to increase proportionally with performance.

Intranets – The main difference between an intranet and a cluster is that an Intranet introduces

heterogeneity into the system, it also presents the problem with separate administration giving rise to systems must negotiate conflicting policies (systems in an Intranet are assumed to be centrally administered). Another problem is the lack of global knowledge. It is impossible for any system to have global accurate knowledge about the different systems states. Centralized administration gives the advantage that it simplifies security and systems like Distributed Computing Environment (DCE), DCOM and CORBA could be successfully applied to Intranets. Programs in these systems do not generally create processes manually, but rather connect to “services” that “encapsulate” hardware resources.



Figur 1: Example in CORBA

Interactions occur via RPC (Remote Procedure Call) or remote method invocation, models that have become a standard in remote “function calls”.

Services like “wide area files system” technologies could also be applied successfully in an intranet, DFS (Distributes File System) is probably the most well known virtual file system. Systems like these introduce services that can be used to break the boundaries for secondary storage restricted by the operating system. E.g. offering petabytes of storage on network drive F:\ (Even though windows imposes restrictions on hard drive size not to be bigger than x bytes). Virtual file systems also allow for data to be duplicated, securing access and making the system more secure against data loss (e.g., in case of hard drive failure).

Due to the less secure environment, systems like Kerberos has evolved, offering security and a unified authentication structure throughout the Intranet.

Internets – is the most complicated system and is characterised by a lack of centralised control, large geographical distribution and international issues. In an Internet we cannot rely on the existence of a common scheduler and must therefore explore other alternatives. A common strategy to solve this problem is using a “scavenging” grid system. A system that enables, often idle, computers communicate with a kind of global scheduler, called management node. The management node applies jobs to computers that match the current jobs restraints. New technologies like Legion and Globus is being developed, treating hosts like objects in an ordinary object oriented fashion.

What approaches are needed to develop computational grids?

Today grids are developed independently and often in low level languages like in assembler. These

works are often expensive, hard to adapt to other applications as well as other grid-systems. They are also often fragile. The development of grids must be internationally standardised. The development must also be done in smaller modules, as with the different protocol layers that are the fundamentals of the Internet today. One can divide the developers into three classes, which are grid, tool, and application developers.

Grid developers - develop protocols and produce routine libraries. The challenge here is to produce a library of protocols which will work well with many underlying technologies (e.g., different types of networks). The library must also fulfil the many different requests from the tool developers, making it hard to give every different request best performance, while at the same time accommodating the different underlying technologies. There will therefore be a battle between generality and performance. It is very important to standardise all protocols so the tool developers know how they can implement their work.

Tool developers - concentrate on developing a system that will take care of the main things that must exist for using various applications. Security must be taken care of, things like authentication and confidentiality has to be implemented. They also develop methods for payments, which are very important in for example on-demand grids. Finally they also develop methods to find and organise resources and information. Which include communication, fault detection and many more things. Tool developers must adapt their protocols to fit the protocols developed by the grid developers and also keep in mind the requests from the application developers. Everything must be standardised so that the application developers easily can make use of the capabilities from the tool-layer. The tool developers must also inform the application developers which implementation can get higher or lower performance.

Finally there are the *application developers* which are supposed to use all the methods they need from the tool level to make specific application programs for the end users. Applications that are intended to solve hard problems for the end users. The challenge for the application developers is finding algorithms that divide a task into thousands of smaller tasks that can be handled separately and to make the tasks work efficient with the tool layer. For the end user it will only be important to solve a request while it is less important for the user to know how it works.

What is needed for computational grids to become a common service?

As was discussed earlier on there are a lot of possibilities and great advantages of having a computational grid infrastructure. But there are also many difficulties to overcome before a grid will be as natural to use as electricity. There are some requirements that must be fulfilled before peoples and companies will use it in a large scale. One of the parameters is that the service has to be dependable, which means that the service always comply with the expected and fundamental parameters as security, availability and confidentiality.

The second requirement is consistency, which means that every layer should be standardised. Standard interfaces make it simple and cheap to make new applications.

The hard thing is to balance high performance and homogeneity. Since different application need different underlying technology to work best, one need to encapsulate different technology approaches in the same standard. That is to create the illusion that it is homogeneous. But it is very hard to get the best performance for every application.

The last requirement for the grid is that it must be inexpensive, while at the same time deliver high

performance. Compared to electricity one end-user can use very little power for little money, although it comes from very expensive power plants. Making payments just for the power one consumes.

To make computational grids common there must be some influence from politics and organs that acts internationally to standardise technology.