Grid Computing, Clusters and Security

Ruediger Berlich
Ursula Epting
Jos Van Wezel

Table of Contents

Introduction

In the computing world there are many areas where technical development can't keep up with the demand for computational resources. Sometimes, workarounds used to overcome such deficiencies gain a life on their own and become the basis for new developments. As an example, modern particle physics experiments, such as the upcoming LHC experiments [2] at CERN/Switzerland or the BaBar experiment at SLAC (Stanford, USA) [3] will, over the years, produce more data than can be realistically processed and stored in one location, even when using sophisticated cluster architectures. Predictions for the data production of the four LHC experiments are in the range of one Petabyte per experiment per year, or altogether a data rate of 40 GBit/s.

What's more, as the experiments evolve and particle accelerators become more sophisticated, the predicted growth in data production over the years far exceeds the predicted growth of computing power. The latter is described by Moore's Law, according to which the processing power doubles every 18 months. So a local cluster of a given size won't be able to keep up with processing this data, even if it is constantly being upgraded to the newest technology. In such a situation, one has but two choices: One can try to find additional monetary resources to frequently increase the computing and storage power in the location where the data originates. Or, one can try to use distributed computing- and storage-resources already available in participating institutions - particle physics experiments are international by design. Single countries today cannot afford the huge cost involved in building and maintaining particle detectors and accelerators any more. While, from a technical perspective, maintaining and administering local computing resources is of course preferable over distributed approaches, it becomes immediately clear that, in times of tight budgets, using available distributed resources is the only possible solution to the challenges imposed by modern particle physics. It should be pointed out that the need to examine more data than can be realistically stored and processed in a single location is not particular to particle physics.

Thus, an additional cost saving effect stems from the possibility to share distributed computing resources not only among physicists, but also with other research disciplines and business ventures.

In the following we will present an overview of Grid Computing. Security and authentication in the Grid environment are then discussed in more detail on the example of GridKa, the Grid Computing Centre being built at Forschungszentrum Karlsruhe in conjunction with particle physics initiatives such as LHC/CERN or BaBar/Stanford/USA. Finally, the design and setup of GridKa is described in the section "Towards the 1000 nodes".

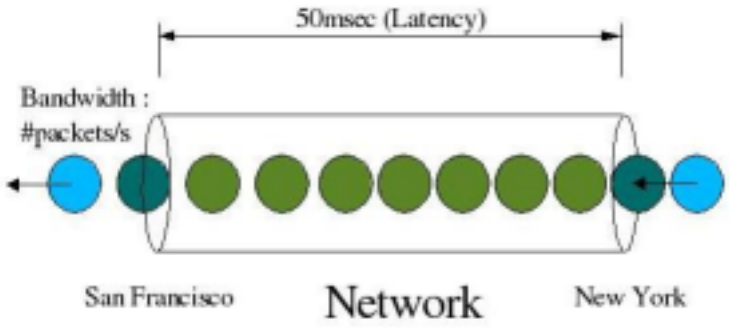
The Vision

The lack of suitable, standardised solutions for distributed, large-scale computation has sparked off a new research discipline, called Grid computing. The vision behind this new approach was first put forward by Ian Foster and Carl Kesselman in their book "The Grid - Blueprint for a new computing infrastructure"[4]. In short, it could be described as "Computing Power from a Plug in the wall". In the end, one shouldn't need to care for the location where data is being processed. Really important are speed, safety, and reproducability of results. It is this obvious analogy to the electrical power grid that has also given the name to Grid computing: You do not need to know where electrical power is being produced. All that is important to you is that it is delivered to your home in a reliable way.

Ian Foster, one of the Grid visionaries, while giving a talk at the Sun booth during Supercomputing 2001.

Distributed Computing

Distributed Computing is not a new paradigm. But up until a few years ago, networks were too slow to allow efficient use of remote resources. But as the bandwidth of high-speed WAN's today even exceeds the bandwidth found in the internal links of commodity computers, it becomes inevitable that distributed computing is taken to a new level. It now becomes feasible to actually think of a set of computers coupled through a high-speed network as one large computational device. Or in other words: "The world is your computer". Of course there are still limiting factors. The "speed" of a network is a complex variable. It consists of the bandwidth (the number of bits you receive per second on one end of the network) and the latency (the amount of time it has taken these bits to travel from the source to the recipient. While you can today scale the bandwidth of a network connection to virtually any level - provided you can pay for it - there are physical limits to its latency.



Obviously, data cannot travel faster than the speed of light. So there is a lower limit to the amount of time needed to transfer the data, no matter how sophisticated your network hardware is. But since this data will have to pass repeaters and routers along the way, the actual latency will be much higher than the physical limit. E.g., the latency across the USA is in the range of 50 msec. Still, this is not a very high value. As a comparison, a modern IDE hard drive with 7200 RPM has mean access times of 8.5 msec. So while latency does form a limiting factor, and will continue to do so in the foreseeable future, network latency is already in the range of, say, the mean access times of old MFM hard drives.

Overcoming Latency

While latency is a limiting factor, there do exist some possibilities to reduce its effects. One example is a semi-commercial approach by Canada-based Canarie, Inc. [7], called the Wavelength Disk Drive (WDD). The idea is to use the network itself as a storage device. An optical network is used to form a loop, i.e., data stays in the network until it is removed by some interested party. As the data doesn't need to be transferred into the network anymore - the network is the storage device - and since it is, with a certain likelihood, already close to its recipient, access times to data are reduced. Unfortunately, the storage capacity of such a device is limited to a few Gigabytes. But it is still sufficient to allow the usage of Wavelength Disk Drives as cache-like devices. One could think of other ways to reduce latency, such as speculative copying of frequently used data, a technique often used in modern processors to overcome the speed difference between CPU-caches and main memory.

Requirements of distributed computing

Imagine you want to submit a compute job to the Grid. There are certain parameters you want to be sure of. First of all, you want to know that your job is submitted to a computer that fits the requirements of your program. Such requirements may include the processor type, local storage capacity for temporary files, amount of RAM and various other hardware parameters. Still, the idea behind Grid computing is that you do not need to know where your program is actually being executed. So instead of choosing a machine from a list, you need to describe the requirements of your program in a way that can be understood by some Grid component responsible for choosing the target machine. If you are handling sensitive data, you need to know that no unauthorised party can gain access to it. Likewise, the owner of the machine used to do your computation needs to know that you are using his hardware only in the way it is intended to. In short, there must be a trust relationship between the person who submits the job and the owner of the target machine. The complicated part is that these two people do not know each other and indeed should not need to have to interact in any way in order to allow the job submission.

Before program execution starts, any data needed by your program in order to do its job must be accessible to it from the target machine. Usually this means copying some data set over the network before transferring the program code. Alternatively, one could bring the program to the data rather than vice versa. During and after the calculation you need to get access to the output created by your program, so this information needs to be transported back to you. Last, but not least, you need to pay for the computing time you've used. The cost can vary from very tiny amounts of money to huge sums, but in any case there must be an accounting infrastructure across country and currency boundaries. Most of these requirements of Grid computing could probably be satisfied using existing tools. E.g., with a Virtual Private Network and batch submission systems such as PBS it would be possible to submit jobs on remote machines in a secure and reliable way. But while many tools

for distributed computing are available, they do not form - and, more importantly, do not intend to form - a homogeneous approach. So the task at hand now is the creation of a standardised software infrastructure suitable to the requirements of distributed computing. Collectively, efforts to create such an infrastructure are today often referred to as "Grid computing".

The World Wide Grid

There is a striking resemblance of Grid computing to the World Wide Web. The Web was started at CERN in late 1990 by Tim Berners-Lee as a means of efficient information exchange between physicists all over the world [5]. Grid computing in its current form aims at providing a means for efficient exchange of computing power and storage capacities between scientists and commercial entities, and, just like the Web, it owes many of its current features to work done by computer scientists at CERN. Just like it was the case in the beginning of the World Wide Web, there are currently many special purpose Grids, which usually use the Internet for data transportation. It'll take its time until these Grids grow together and form a World Wide Grid, but the ultimate goal is a global, standardised infrastructure for transparent execution of compute jobs across network boundaries. Mind you, we are not talking about every-day jobs here like, for example, spell checking of text documents. The tasks likely to be executed over a Grid in the medium-term future will be huge analysis jobs like weather forecasts or simulation of particle decays. As described in the beginning, local clusters or workstations have become insufficient for such large-scale computation.

Data-Grids vs. Computing-Grids

So far we have offered the needs of particle physics as an explanation, why Grid Computing was started. Distributed data processing requires parallelisation. A typical particle physics analysis requires the analysis of millions of collisions of particles (or in short "events"). Usually, when processing one event, there is no need to have any information about the processing of another event. The analysis of a given set of events is sometimes called an "embarassingly parallel" problem, as one only has to run the same analysis with a portion of the dataset on more than one compute node, and collect and assemble the results in the end. This is the typical situation found in Data-Grids, i.e. Grid environments tailored to the needs of the processing of huge data samples. In comparison, a Computing-Grid deals with the execution of parallel algorithms rather than the distributed processing of huge amounts of data. A Computing-Grid could thus be described as a "Super-Cluster", assembled from local clusters and single machines all over the world. As we have described above, latency is the limiting factor for parallel computation. Usually parallel algorithms need to exchange information between participating compute nodes during the computation, so the degree of parallelisation and thus the speedup depend on the amount of information they need to exchange. One could argue that, for this reason, Grid techniques are more suitable to Data-Grids than Computing-Grids, as the number of possible applications tolerant to the comparatively large latencies is limited. But we have also seen that the distinction between local machines and the Grid becomes more and more blurred. Latency is today much less of an issue than it was 5 years ago and there are interesting new developments in the field of latency tolerant algorithms. So while the majority of Grid applications can be expected to be of the data Grid type, Computing-Grids will soon start to play an important role as well.

Hardware-Infrastructure

It shouldn't come as a surprise that some of the main initiatives related to Grid computing deal with the formation of high-speed networks and the provision of large clusters.

Here are two of the more well-known international efforts: Geant [8] is a four year project, set up by a consortium of 27 European national research and education networks, to form a fast, pan-European network infrastructure. It incorporates nine circuits operating at speeds of 10 Gbit/s plus eleven others running at 2.5 Gbit/s. The TeraGrid [10] is an effort to build and deploy the world's largest and fastest distributed infrastructure for open scientific research. When completed, the TeraGrid will include 20 teraflops of Linux Cluster computing power distributed at the five sites (Argonne National Lab,California Institute of Technology, National Centre for Supercomputing Applicatons, San Diego Supercomputer Centre and the Pittsburgh Supercomputer Centre), facilities capable of managing and storing nearly one petabyte of data, high-resolution visualization environments, and toolkits for Grid computing. These components will be tightly integrated and connected through a network that will operate at 40 gigabits per second.

Locally, initiatives such as GridKa (see further below) provide the computing power needed to make Grid Computing a reality.

Software-Infrastructure

In Grid computing, the link between applications and the physical Grid infrastructure is provided by a "middleware". It is the middleware's task to address most of the requirements of distributed computing mentioned above. Due to the huge variety of projects, the following list can only be a subset of the middleware packages used in science: The most common software component in Grid computing today is the Globus toolkit [9]. Cactus [11] is a higher-level middleware targeted more at Computing-Grid projects than Data-Grids. Legion [13] falls into the same category as Globus, but aims more at generating the illusion of a single, distributed computer. Unicore allows for seamless inter-operation of supercomputers over a WAN. The Sun Grid Engine is a commercial Grid middleware, incorporating many of the features of a load leveller

Traditionally, in local compute clusters the task of sending information from one node to another has been handled by the Message Passing Interface. There is an implementation of MPI, called MPICH-G [19] that uses the Globus toolkit for authentication. It doesn't fall into the category of middleware, but should nevertheless be mentioned here due to its importance in Computing-Grids. Basically it allows treating a Grid environment like a local cluster, albeit with a larger latency.

Grid Initiatives

Again this can only be an incomplete list. The Global Grid Forum [14] acts as a standardising body in Grid computing, similar to the Internet Engineering Task Force (IETF). Between two and three meetings per year aim at providing a forum for the discussion of new technical developments. The European DataGrid project [15] (EDG) was initially started with the needs of particle physics experiments in mind, but today incorporates many other research disciplines, including genome research. The project, which is funded by the European Union, aims at setting up a "computational and data-intensive grid of resources for the analysis of data coming from scientific exploration". Like the name suggests, the EDG is purely targeted at Data-Grid applications. Part of the EDG project is a software package that provides brokerage of computing resources on top of Globus, which still requires information about the target machine and thus acts more like a global batch-submission system. The CrossGrid [16] aims at providing an application framework for interactive computation on top of the infrastructure provided by the EDG. The Grid Physics Network, or in short GriPhyN [17], is the American counterpart to the EDG. It is mostly aimed at particle physics experiments. LCG aims at providing a virtual computing environment for the upcoming LHC experiments (CERN). It collaborates closely with Geant, EDG, GriPhyN and other Grid initiatives. GridPP [18] "is a collaboration of particle physicists and computer scientists from the UK and CERN, who are building a Grid for particle physics".

Grid Computing and Linux

We have seen that there is a huge variety of Grid computing projects. Open research benefits from open platforms. As an example, it can be expected that, should Grid computing be successful, the middleware will become part of the Operating System. In an ideal Grid environment, users and authors of software packages should need to know as little as possible about Grid computing, just like today an ordinary user doesn't need to know anything about networking in order to send an email. This is nowadays often referred to by the term "Invisible Computing". In order to develop an integrated, Grid-aware OS the free availability of the source code is mandatory. This makes Linux (and other Open Source Operating Systems) the ideal platform for Grid computing. Linux already has a strong position in Grid research, owing to the fact that Grid computing inherits many of its features and ideas from clustering. Linux is strong in this area.

Having discussed the general idea of Grid Computing, we now want to take a closer look at security issues.

Security and Authentication in the Grid Environment of GridKa

The idea of building up a World Wide Grid is a particularly big challenge for security. How can a local site administrator be sure that an alleged scientist from a remote institution - let's say in Canada, who is requesting access to local data archives or compute nodes in Germany really is the person he pretends to be ? And how can a scientist in Italy be sure, that she is really communicating with the right data server, where she wants to save her results of the latest experiments? Apart from the need to build up this two-way trust relationship in technical means, there is a lot of organisational work to be done to establish a wide-spread security infrastructure, which is accepted by all involved institutions in the world.

Within the environment of GridKa at the Forschungszentrum Karlsruhe the Grid Security Infrastruture (GSI), which is implemented in the Globus Software Toolkit, is being used. GSI offers secure single sign-on and preserves the site control over access policies and local security. It provides its own secure versions of common applications, such as FTP (grid-ftp), and a programming interface for creating secure, Grid-enabled applications. See [9] for further information on GSI and the Globus Toolkit.

GSI is based on asymmetric cryptography used in a "Public Key Infrastructure" (PKI). Asymmetric cryptography allows users to communicate securely without the need for a prior confidential channel to exchange an encryption key. Exploiting features of a specific class of mathematical challenges that are easy to create, but virtually impossible to hack (like factorizing large prime numbers), and end-entities generate a complementary set of keys: a "private key" that will be kept secret and a "public key", that is distributed to the world. Data sets which are encrypted with the public key can only be decrypted with the private key (and vice versa). This way data confidentiality, message integrity and non-repudiation can be achieved between two halves of the key pair.

A PKI is used to uniquely bind an identifier to a specific key together in a "certificate". The identifier can represent any entity on the Internet or a Grid service. Anyone wanting to communicate to another entity on the Grid can obtain their certificate and use the public key contained in it to send messages that can only be read by the original owner - who has the knowledge of the private key. But the sender must first be sure that the intended recipient is indeed the holder of this private key. Therefore a "third, trusted party", a Certification Authority (CA) issues a certificate to a user (or host or service) which contains the public key of the entity, the unique identifier (the subject of a certificate - for example the common name of a user or the Fully Qualified Hostname - FQHN) and some additional information, provided with the signature of the CA.

```
Example Extract:
  Certificate:
Data:
  Version: 3 (0x2)
  Serial Number: 1 (0x1)
  Signature Algorithm: md5WithRSAEncryption
  Issuer: C=DE, O=GermanGrid, CN=FZK-Grid-CA
  Validity
     Not Before: Dec  4 13:53:46 2001 GMT
     Not After : Dec  4 13:53:46 2002 GMT
  Subject: O=GermanGrid, OU=FZK, CN=Albert Einstein or CN=gridserver.fzk.de
    [...]
      RSA Public Key: (1024 bit)
   [...]
  -----BEGIN CERTIFICATE-----
  MIIEMTCCAxmgAwIBAgIBATANBgkqhkiG9w0BAQQFADA2MQswCQYDVQQGEwJERTER
  [....]
  -----END CERTIFICATE-----
```

Before a certificate is issued, users must go through a registration process, defined in a "Certification Policy and Certification Practice Statement" (CP/CPS). End-users requesting a certificate must present a passport or identity card to the Registration Authority (RA) of the CA in order to prove that the person is really who he or she pretends to be.

The software infrastructure of the Certification Authority at GridKa uses an OpenSSL software framework, it is hosted on a Linux System. The GermanGrid CA is operating since December 2001 and its CP and CPS are published on the web (see [24] for further information). Certificates are being issued to scientists from several German institutions related to national and international Grid-Projects. Each certificate has a lifetime for only one

year and can easily be "revoked"if any misuse is announced or the holder of the certificate changes his job. A minimum key-length of 1024 Bit is required.

Mainly within the coordination groups of CA-Managers from the European-Data-Grid (EDG) and CrossGrid-Project, trust-relationships between 18 international institutions were established. These institutions are: AUTH (Greece), CERN (Schweiz), CESNET (Czech Republik), CNRS (France), Canada Grid (Canada), DATAGrid-ES (Spain), DOE (United States), Grid-Ireland (Ireland), GridKa (Germany), II SAS (Slovakia), INFN (Italy), LIP (Portugal), NIKHEF (Netherlands), Nordic Countries (Denmark, Norway, Sweden, Finland), PSNC (Poland), Russian DataGrid (Russia), UCY (Cyprus), UK e-Science (United Kingdom). (For further information see [25]).

In practice this means that a holder of a valid certificate, issued by one of the 18 CA's is able to authenticate himself to all resources provided by these institutions. At this point a distinction between "Authentication" and "Authorization" has to be made. In the Grid Security Infrastructure authorization decisions are made at the local resource level. Individuals and institutions are united in so called "Virtual Organizations" (VOs). Entities authenticate themselves using their certificate and the ability to use their private key. An LDAP-based directory service retains a list of users (i.e. : certificate subject names) that are part of a VO, managed by a representative of the community. This list of users is further divided into groups, managed by one or more group administrators. Sites periodically retrieve a list of users from this directory and configure access to their resource according to the local security policy.

It seems that with the implementation of the Grid-Security Infrastructure a good basis for security in the World Wide Grid is given. But while asymmetric cryptographie in technical/mathematical view gives a high level of security, a few weak points remain :

- How well is the private key of a user protected?
- How are users dealing with their private keys?
- Are they trying to share certificates and passwords ?

Some of these issues will remain moot points with any security infrastructure that has to care for practical issues like usability and scalability alongside the mere security requirements. But work continues to combine these two contradicting requirements.

In the last part of this paper we will now have a look at the design and set-up of the cluster which, as part of GridKa, provides computing resources to authenticated and authorised users.

Towards the 1000 nodes - Large cluster design and operation with Linux

Constructing a cluster system starts with asking questions, such as

- What will it be used for ? (the programs that will run on it determine its construction)
- Are there many small programs or a few large ones?
- Can the programs work in parallel?
- Do running programs need to communicate with each other while running?
- Are they floating point or integer intensive ?
- How much memory is needed and should it be accessible throughout the cluster ?
- What about IO. How large are the programs and how much do they read and write ?
- More importantly, how do they read and or write?

• Are the programs available on a specific platform ?

• Is the source available ?

• Is it allowed to use the programs on more than one CPU at the same time ?

Linux has come a long way, but there are still many commercial programs that are not yet available or behave poorly on (Intel) Linux.

The choice to build a Linux cluster computer is appealing because of the lower costs of hardware and software. The hardware consists of commodity PC components and the software is largely available as Open Source. Still, there remain some challenging problems to be solved with this approach. The hardware is usually not as robust or multi purpose as professional server grade machines. Linux has limited support for hardware monitoring and almost no support for handling failed components. Even a cluster, with all its redundant compute nodes, needs reliable machines in some key positions (file servers, gateway nodes, administration nodes). Although Linux is easy to install from CDROM, installing 100+ nodes this way can be a tedious process. Centralized installation packages are scarce and have limited functionality. The hardware may be available and cheap, it is the administrative software that makes the in concert operation and maintenance of hundreds to thousands of nodes possible.

How large do you want your cluster to be? There are only relatively few installations with more than a few hundred nodes, so Linux has no sufficient track record in this area (has any other OS?). Apart from installation and monitoring there are issues regarding throughput and intercommunication scalability. If the task you have selected for your cluster implies some sort of memory sharing you need a product like MPI ("Message Passing Interface", see e.g. [27]) which implements a distributed memory environment in software. Messages are passed between nodes over ethernet or special low latency, read-expensive, hardware interfaces such as Myrinet or Infiniband.

One of the challenging problems of large clusters is to offer all nodes uniform access to online data. Jobs are scheduled to run on randomly selected nodes and need not be aware of the location of the data they need during execution. In fact it is observed at the GridKA cluster that many jobs read and write data from and to the same disk area because users start their work in batches. The job scheduler dispatches the jobs in a FIFO kind of way ("first in, first out") and users usually queue a batch of jobs in one session. One way of offering the data to the applications running on the compute nodes is by NFS. A single Linux NFS file server can sustain 30 to 40 MB/s on Gigabit Ethernet to one client. Jobs on clients that access the same data use the same server. Depending on the access pattern an NFS server can support 20 to 30 clients. It is easy to see that this does not scale to hundreds or even thousands of nodes.

The GridKA cluster in Karlsruhe is built to analyze data from high energy physics experiments. The analysis is characterized by reading, processing and writing files with a size anywhere between 100 and 2000 MB. The CPU

involvement is relatively low which means the jobs are mainly IO bound. When many nodes read many files at the same time, throughput can be improved by reading the data in parallel from N disks by N machines.

The public domain package PVFS (Parallel Virtual File System) makes it possible to turn file servers into IO nodes that each serve a slice of the data. The parallel file system driver on the client side (i.e. the compute nodes) assembles the slices into their original sequence. The slices or stripes of files are stored on several disks and accessing the file results in the simultaneous access to all disks, therefore multiplying the theoretical throughput with the number of stripes. Disks inside the cluster nodes can be joined to form one large parallel accessible storage device. The PVFS performs especially well when the cluster accesses the same file on several nodes at the same time. There are some drawbacks with PVFS and for this reason GridKA uses a commercial parallel file system known as GPFS.

A different approach to parallel data access was developed in the high energy physics community. The data is first copied to several servers before delivering it to individual nodes. The number of copies can scale with the size of the cluster. CASTOR, developed at CERN/Geneva, and dCache, developed at DESY in Hamburg, have sophisticated algorithms to cache data on the server nodes and interact with background storage on tape. The biggest disadvantage of these systems is that data can only be accessed in read-only mode. There is no mechanism to guarantee cache coherency.

Common to all methods is a dedicated server for the metadata. The metadata server maps file names to the location of the slices (that together form the content of the file), stores file locks and file attributes and arbitrates concurrent write accesses. At the time one meta-data server per cluster of 100 nodes was sufficient. But 250 or 1000 nodes will swamp the meta-data server with requests. For example each stat() in the cluster goes directly to the meta-data server. The object storage project 'Lustre' works with a scalable number of meta-data servers.

Finally the newly constructed cluster goes into production. Your nodes are happily running jobs, which have high speed access to online storage. Then you receive a mail about jobs failing to run because /tmp is full. Another mail reports that the job scheduler is unable to find some compute nodes. It is time to set up a monitoring system. The monitoring system will report problems before the users of the cluster do and does this by checking the status of several key indicators on cluster nodes, e.g disk space, functionality of services like NFS, DNS, DHCP, network connections etc.

The public domain software Nagios is a framework for monitoring several computers at a time. It aggregates the status and readings of possibly hundreds of machines and in a series of orderly arranged web pages. It comes with ready-made plug-ins to monitor a range of services and allows extensive adaptation. For the GridKa cluster Nagios was adapted to do monitoring in two stages. At the lowest level the compute nodes, file servers or administrative clients are monitored by a Nagios first level collector. The first level collectors are monitored by the second level Nagios system which presents the cluster status to the administrator. This split mode filters out many false positive alarms, and, more importantly, is better scalable. One Nagios server now monitors 40 to 50 machines and the split mode makes it possible to monitor 40*40 to 50*50 machines.

Most software needed to operate a large cluster with Linux is still in an early stage of development. New approaches are tested and fine-tuned. As the clusters grow, more bottlenecks will become apparent that have to be dealt with. Linux needs the input and experience of large cluster installations to steer this development in the right direction. Many tools and mechanisms developed for Linux simply lack the scalability today because it is not needed for desktop machines or the few servers in the machine room. Really large clusters are pushing the Linux limits today. But frontiers, in the Linux arena, have always served as an incentive for further development in the days to come.

Conclusion and Outlook

It should have become clear by now that Grid computing has already come a long way. Still it can be considered to be "work in progress" and, in some ways, a buzz word. While, thanks to the involvement of many commercial and non-commercial institutions, it is safe to conclude that Grid computing will play an important role in the future, it is not yet possible to exactly determine all areas in which it will have an impact on society beyond its current strong-holds. Competing, but lower-scale, efforts such as .Net, Mono, .Gnu or OneNet might fulfil some of the roles "traditionally" assigned to Grid computing. Possibly even the naming conventions might change and "Grid computing" might become a catch-all term for everything related to distributed computating. The important point is that a lot of work is being invested in the moment in order to take distributed computing to the next level.

References

1. The "Transtec Compendium" http://www.transtec.co.uk
2. Information about the Large Hadron Collider at CERN http://lhc-new-homepage.web.cern.ch
3. Information about the BaBar experiment http://www.slac.stanford.edu/BFROOT/
4. The Grid, Blueprint for a new computing infrastructure; Ian Foster and Carl Kesselman, Morgan Kaufmann, ISBN 1-55860-475-8
5. The web : how it all began http://public.web.cern.ch/Public/ACHIEVEMENTS/web.html [http://public.web.cern.ch/Public/ACHIEVEMENTS/web.html]
6. OGSA Whitepaper - The Open Grid Services Architecture http://www.globus.org/research/papers.html
7. Information about the Wavelength Disk Drive http://www.ccc.on.ca/wdd
8. Geant - a pan-european network infrastructure http://www.dante.net/geant/
9. The Globus toolkit http://www.globus.org
10. An effort to create the world's largest and fastest distributed infrastructure for open scientific research http://www.teragrid.org
11. The Cactus middleware http://www.cactuscode.org
12. The Unicore middleware http://www.unicore.org
13. The Legion middleware http://legion.virginia.edu
14. The Global Grid Forum http://www.gridforum.org
15. Homepage of the EU DataGrid http://eu-datagrid.web.cern.ch/eudatagrid/ [http://eu-datagrid.web.cern.ch/eudatagrid/]
16. Information about the CrossGrid project http://www.crossgrid.org/
17. The Grid Physics Network http://www.griphyn.org/
18. The Grid for UK particle physics http://www.gridpp.ac.uk/
19. Homepage of MPICH-G2 http://www.unix.mcs.anl.gov/mpi/mpich/
20. Randy Butler, Von Welch, Douglas Engert, Ian T. Foster, Steven Tuecke, John Volmer, Carl Kesselman: A National-Scale Authentication Infrastructur. IEEE Computer 33(12): 60-66 (2000)
21. EDG-WP6: Authentication and Authorization, http://www.dutchgrid.nl/DataGrid/security/WP6-D6.4-security-summary-draft-20020125-1.doc [http://www.dutchgrid.nl/DataGrid/security/WP6-D6.4-security-summary-draft-20020125-1.doc]
22. Mary R. Thompson and Doug Olson, LBNL, Robert Cowles, SLAC: Grid Certificate Policy WG, Grid Trust Moder for CA Signed Identity Certificates, April 2000
23. Fran Berman, Geoffrey C. Fox, Anthony J.G. Hey: Grid Computing, Making the Global Infrastructure a Reality, 2003
24. CP/CPS for GridKa http://www.gridka.fzk.de/ca/fzkcps.pdf
25. Trust relationships http://marianne.in2p3.fr/datagrid/ca/ca-table-ca.html [http://marianne.in2p3.fr/datagrid/ca/ca-table-ca.html] and http://grid.ifca.unican.es/crossgrid/wp4/ca

26. Parts of this paper have been taken from a paper submitted to the UKUUG 2002 Linux developers conference by Ruediger Berlich and Dr. Marcel Kunze, see http://www.ukuug.org/events/linux2002/

27. The MPI standard http://www-unix.mcs.anl.gov/mpi/

About the authors

After obtaining a masters degree in Physics in 1995, Ruediger Berlich started working for SuSE Linux AG. Having worked in various positions in the company, in 1999 he founded and led the UK office of SuSE together with a colleague. In August 2001 he left SuSE in order to start working in Particle Physics again, with a strong focus on Grid Computing.

Ursula Epting has studied mathematics, German language and literature at the university of Mannheim. After further training (MCSE, Novell, Unix) she started in December 2000 at Forschungszentrum Karlsruhe in the area of network security. Ursula maintains the certificate authority of GermanGrid, located at Forschungszentrum Karlsruhe. (ursula.epting@hik.fzk.de)

Jos van Wezel (jvw@hik.fzk.de) has been a system administrator and scientific programmer at the Vrije Universiteit, Amsterdam where he has been involved in the early Grid projects. Currently he works as storage expert for the GridKa cluster at Forschungszentrum Karlsruhe.

Thanks !

The authors would like to thank the department heads of GIS, GES and NINA at HIK/Forschungszentrum Karlsruhe (http://hikwww2.fzk.de/hik/), Prof. Koch of Ruhr-University Bochum (http://www.ep1.ruhr-uni-bochum.de), and the German Federal Ministry of Education and Research (http://www.bmbf.de) for their support.