FutureGRID : A Program for long-term research into GRID systems architecture

J A Crowcroft[†], S M Hand[†], T L Harris[†], A J Herbert[◇], M A Parker[‡] and I A Pratt[†]

[†] University of Cambridge Computer Laboratory, Cambridge, UK
[◇] Microsoft Research, Cambridge, UK
[‡] Cambridge e-Science Centre

August 7, 2003

Abstract

This is a project to carry out research into long-term GRID architecture, in the University of Cambridge Computer Laboratory and the Cambridge eScience Center, with support from the Microsoft Research Laboratory, Cambridge.

It is part of a larger vision for future systems architectures for public computing platforms, including both scientific GRID and commodity level computing such as games, peer2peer computing and storage services and so forth, based on work in the laboratories in recent years into massively scaleable distributed systems for storage, computation, content distribution and collaboration[26].

Good architecture arises from harvesting best practice. For example the ANSA project led by Andrew Herbert, now of Microsoft, was formed in 1985 to harvest the early experience of research into LAN-based distributed computing. ANSA paved the way towards the CORBA standard for enterprise application integration. From the research results of the time the ANSA project synthesized a platform, ANSAware, which was used by others to build distributed applications. The design of ANSAware was a challenge in integration and spawned research in its own right. Experience from the community of ANSAware developers and ongoing research results were fed back into the design iterations of the platform ensuring it remained state of the art. Grid computing is currently at a similar threshold to that faced by LAN-based distributed computing in 1985: promising research results point the way, many groups are building ambitious applications, but there is a lack of a suitable architecture to pull it all together. It is our belief that coupling the four areas of investigation cited in this project we can develop a future Grid architecture that points the way forward from the first steps taken with the introduction of the Open Grid Services Architecture (OGSA). Our target for large scale exploitation is in the timeframe of five to ten years out.

Emergent Architecture

There are a number of specific techniques that we believe are likely to have important implications for GRID architecture, and we are pursuing their development now, so that the local (Cambridge, and broader UK) community can benefit soonest:

1. Concrete resource accounting and management
2. Use of Distributed Hash Tables and other peer-to-peer techniques
3. Spread-Spectrum Computing
4. Design of self organising systems

Recent research results in these areas, from the peer-to-peer community, large scale control systems theory, publish/subscribe and event notification systems, and self-organising systems theory is very promising. We anticipate that several orders of magnitude growth in the size of typical distributed applications is feasible, with lower management costs than previous systems designs. Searchable storage systems of thousands of petabytes, with "10 9s" availability should not be impossible; high availability, high performance distributed computations for tasks that have traditionally been hard to decompose; timely notification of events and content update are on our agenda; finally, a replacement of the collaborative frameworks for synchronous and asynchronous computer supported working is needed, and we believe we have an approach that can accommodate this very nicely.

Four corners of the Program

It is too early to specify an architecture on a *tabula rasa*. Instead, we choose to explore the systems design space through four projects, which exploit the skills and track record of the two laboratories, and see how the gradual identification of common elements leads to an emergent understanding of the overall requirements.

In particular, there are four application areas which motivate our study, and for which we will demonstrate prototype GRID services:

1. Massively scalable middleware for collaborative virtual communities. This combines the experience with implementing the multicast tools that make up the access grid, and already involving MSR and the Computer Lab, but replacing the multicast IP substrate with P2P systems based on MSR work, including Scribe and Pastry.
2. Advanced resource location mechanisms. This extends recent results on Content Addressable Systems to include multiple criteria for resource location.
3. Automatic replication and distribution of s/w components. Here we will exploit self organisation and redundancy coding concepts to their full.
4. Global data storage and publishing systems. This takes advantage of work adding persistence and efficient update to Pastry and other P2P storage systems, already in collaboration between CL and MSR.

The next four sections cover the details of the four related, but non inter-dependant projects which form the basis of this part of a larger programme of work on Future GRID Architectures. At the end of the work that constitutes these four projects, we will be in an excellent position to make a fundamental contribution to the vision for the Future GRID Architecture, and this is discussed in the final section.

1 Massive Scaling Collaborative Environments through P2P

1.1 Background

This part of the project is about using P2P techniques to build massively scalable, reliable distributed systems support for online Collaborative virtual environments.

It is rooted in much deep background in the lab in such tools in the past using IP Multicast (the Access Grid tools were partly built one of us) and in P2P.

This work (described in detail at `http://www-mice.cs.ucl.ac.uk/multimedia/software/`) formed the basis of the MICE Mbone tools, now largely in use as Access Grid software. Many problems persist with the systems design that these tools rest on, which are recognised by the community, not the least being the limited deployment of native IP multicast, and the unreliability and problems of accounting and carrying out access control for interdomain multicast.

Several UK GRID projects have ambitions in this space, albeit largely at higher levels (e.g. MyGRID, Discovery Net, etc). The Equator IRC (which the PI is involved in) has also been extended to look at Advanced GRID Interfaces for Environmental e-Science, and CoAKTinG, GRID enabled knowledge services. However, these are at the human-computer interface, and knowledge engineering levels. They rest on rather shaky network foundations at the moment.

Recent work at MSR and in the Computer Laboratory has addressed the way that application layer multicast and p2p may be effectively and efficiently combined, at least in simulation.

This work involves extended the approach to applications and testbeds and obtaining real performance experimentation results.

1.2 Observations on P2P

MSR was a key player in developing Pastry. A related system, CAN [25] achieves efficient organisation of content distribution over a set of distributed nodes, without any distinguished node. Both systems have been extended to offer multicast services, through Scribe and CANcast [28]. Work at CMU developed End Systems Multicast (ESM) [7] which was then used to build collaborative tools on an experimental basis [6].

Meanwhile, work between the Computer Lab and MSR also moved forward on transport protocols assuming the presence of native IP multicast, including work on Fcast for file dissemination, and PGM which can be used for event notification as well as media tools. However, these tend to assume some degree of novel IP router support, which is not without problems.

1.3 Proposal

To carry out the work of porting Access Grid transport protocols and applications such as fcast, PGM, and vic and rat, over Scribe/Pastry, we propose the following work plan:

- Deploy Xenoservers [26] at Access Grid sites (all e-Science Centres).

- Port Pastry and Scribe to Xenoserver with accounting and resource management of network usage! (multicast congestion control work with Jim Gemmell, closed group multicast, pricing with Peter Key etc etc).

- Design suitable API based on WSDL/OGSA (Mainline GRID GGF liason here!) SOAP, etc.

- Port application set (vic, rat, fcast, powerpoint) to new infrastructure, including gateways to legacy "native" multicast versions. Note that such a gateway is also dual function since it is also a BRIDGE for unicast IP native tools!

- User trials

1.4 Value and Evaluation

Value - we develop a standard massively scalable middleware with the largest software provider in the collaborative virtual community space.

Evaluation - of middleware on UK-wide platform requiring no input from network provider (it is important to remove this potentially fatal dependence, in our experience in other projects in the last decade).

We will carry out a performance comparison of end-system p2p only multicast with native solutions. (We expect performance to be slightly lower in terms of use of network bandwidth, but much better in terms of scalable deployment).

2 Multi-dimensional peer-to-peer search systems

2.1 Background

Recent research on peer-to-peer systems has developed a number of mechanisms for implementing distributed look-up algorithms – for example Pastry [27], Chord [30] and Tapestry [32].

These systems differ from one another in the exact approach taken but, in outline, each manages a large global key space within which identifiers representing peers and data are both located. The system exposes operations such as *lookup(k)* and *store(k,v)* on key-value pairs within this space.

A peer manages a section of the key space around its own identifier, maintaining key-value mappings for data in that region. Schemes based on Plaxton trees are usually used for routing lookup and storage requests between nodes [23], forming the peers into an overlay network in which $O(\lg n)$ links are traversed to access any key in a system of n nodes. For performance the peers may track the underlying network topology and use this to influence how the overlay is constructed. Peers maintain additional information for robustness and to support the dynamic entry and exit of nodes.

2.2 Observations

The distributed look-up interface provided by existing peer-to-peer toolkits can form the basis of many data storage applications. For example, projects have set out to build distributed file systems such as PAST [12] (providing persistent and anonymous storage), Pastry [24] (providing mutability and decentralized namespace management) and Mnemosyne [13] (providing a steganographic file system). The existing systems can also support a one-dimensional 'nearest neighbor' lookup in the key space since messages concerning a key can be routed to the node whose own identifier is closest.

However, many problems that might benefit from a distributed implementation cannot easily be cast in terms of simple key lookup operations. As an example, and one that is of particular concern to the Grid community, consider the problem of matching the resource requirements of a particular computational job against the resources offered by a number of systems. Existing proposals (such as those surveyed by Krauter [18]) are generally based on building some form of *resource information database* that clients query in order to locate servers. Replication and caching can be configured to provide robustness and to exploit locality that may exist in accesses.

The desire for a *self-organizing* system, in which replication, caching and (ideally) load balancing are managed and configured automatically is one of the advantages claimed for well-designed peer-to-peer systems such as those outlined in Section 2.1.

2.3 The Work

We are researching techniques for casting more general search problems into peer-to-peer solutions, taking the matching of resource requirements and resource availability as a motivating example. Such systems stand to provide the same benefits to multi-dimensional search problems as existing peer-to-peer lookup systems do to single-dimensional key spaces.

To illustrate this, consider a multi-dimensional space in which server capabilities and job requirements reside – for instance with two dimensions corresponding to network locations (in the manner proposed by Ng and Zhang [20]), others to physical memory size, processor families, availability of specialised facilities, operating jurisdiction and the like. Jobs would specify a target location in this space, corresponding to their requirements and servers would specify a location corresponding to their facilities.

With this in mind a multi-dimensional search system could perform a nearest-neighbor search to identify a set of possible servers to use. In a peer-to-peer implementation, each peer would also be located in the multi-dimensional space and act as a 'broker' between the servers in that space and the job requirements being matched. These brokers would move within the space to perform load balancing. General sequential algorithms for searching metric spaces are well developed, for example those surveyed by Chávez [5].

Concretely, we are investigating:

- Techniques for managing distributed data structures whose representation is shared between a number of nodes as in a peer-to-peer system. Doing so in the face of node failures relies on self-stabilization [10], a field of which our existing work on non-blocking data structures forms a part [15, 16].

- Pastry enables dynamic cacheing and proxying since an application can be invoked each time a message is routed, also allowing resource accounting. Query paths can be reversed for dissemination, achieving excellent locality. Pastry looks after the state robustly.

- Mechanisms for decomposing the design of peer-to-peer systems into constituent parts – for example to separate the management of entry/exit requests of nodes from the management of the data structure being represented.

- The ability for such a system to operate at grid-scale levels and, in particular, the performance of an automatically self-organizing peer-to-peer system in comparison to a distributed directory.

We have reported on early results here in two publications in HPDC[29] and IPTPS[22].

3 Spread-Spectrum Computing Techniques

3.1 Background

In computational grid systems such as Globus, Legion and Condor a key challenge is deciding which resources on which nodes should be used for any given task. In the case where a single processor or portion thereof is required, this boils down to a resource matching problem and may be tackled by techniques suggested in proposal 2. In the more general case, a subset of the grid including many processors or portions is required. This problem is often referred to as the "co-allocation" problem.

Earlier systems such as the distributed queueing system (DQS), IBM's Load Leveller and NASAs portable batch system (PBS) focused on load balancing individual jobs. Indeed, work in this area goes back to the early 70's – a good taxonomy is of work until the late 1980's is given in [3]

On-line monitoring and prediction of resource usage is done using tools like the network weather service (NWS) [31]. Actual distribution and 'launching' of applications is done in a variety of ways, often by the grid resource management tool GRAM.

Current techniques to solve the co-allocation problem take a straightforward approach [19]: a set of candidate resource shares is identified, and then reservations are attempted. A "rollback" process may be instigated in cases where a reservation fails. Node failure later in the computation may or may not be handled.

Again, the use of Pastry techniques to keep statistics on usage and resources over time allows systems to be built which can make longer term decisions.

3.2 Observations

The co-allocation algorithm described in Section 3.1 is expected to operate for all users regardless of their requirements or preferences. There is a danger here that a single notion of optimality may unintentionally be imposed upon the entire system. In particular, it may be that certain classes of user are willing to expend additional redundant resources in order to obtain more resilient or available service.

Redundancy in distributed systems has been used to provide increased availability, performance and reliability by techniques such as striping and mirroring [1], fast fail-over [21], and byzantine fault-tolerance [4]. However this is oriented toward collections of machines and devices which are fairly small (a few hundred machines at most) relative to modern wide-area distributed systems – in particular existing 'peer-to-peer' systems and emerging computational grids – which may have participant nodes numbering in the hundreds of thousands.

3.3 Workplan

We are investigating *spread spectrum computing* as a computation paradigm for the grid. In spread-spectrum computing a subset of a large number of distributed resources are selected according to some keyed pseudo-random process, using redundancy to remove the need to explicitly arbitrate usage between independent users. Although the selection is decentralised, if the candidate set is large enough and the pseudo-random procedure fairly uniform, we can expect relatively good load balancing. If the keys are good enough, the set of resources used by any particular client should be unpredictable and hence resilient to attack.

In terms of the co-allocation problem this means that a set of m candidate nodes are tried in parallel on the assumption that at least some n of them will succeed. Any node may reject an incoming request for any reason (e.g. overload, security policy). Providing at least n accept the request, the overall execution will complete correctly.

We believe such a scheme has a number of benefits. Firstly, it avoids a central notion of "optimality" and hence allows each individual user to choose their own trade-offs in terms of cost, reliability availability, etc. Secondly, tolerance to collisions effectively provides "soft capacity" – the resources of the entire system are shared automatically between the number of users. This second property also means that scalability is inherent.

Existing experience with using this techniques for *storage* have been promising [14], but applying the approach to general purpose computing is more challenging for a variety of reasons. It remains to be seen to how great an extent the desired benefits may be achieved in this domain.

Concretely, we are investigating:

- designing redundantly encoded parallel algorithms,
- efficiency of coding/partition functions,
- fuzzy distribution protocols, and
- programming language support.

A report on an initial strawman design for this was recently submitted to the HOTOS conference, entitled “An Operating System Symphony”.

4 Storage and distribution

Providing a common location-independent environment to applications is an essential tenet of Grid computing. A massively scalable and globally available storage system is a key component of this.

Ideally, such a system should make data highly available through replication across physically separate hosts (perhaps making use of information dispersion codes for storage efficiency). It should utilize aggressive caching so that data can be served locally and with low-latency from where it is currently being accessed, and automatically replicated to cope with varying demand such as ’flash crowds’. However, data updates should be propagated quickly, and some model of consistency enforced.

The Grid Storage Resource Broker [2] is a first attempt to provide a unified storage interface to Grid applications, but doesn’t provide the automatic caching and replication that is desired. Other distributed file systems such as AFS[17] address some of these issues, but can not be described as massively-scalable, and do not provide the automatic ‘hands-free’ management that is required.

The current popularity of peer-to-peer systems has led to significant work in this area, most notably PAST[11], CFS[9] and Freenet[8]. These systems employ Distributed Hash Tables (DHTs) as the underlying storage mechanism and can achieve good availability and caching.

However, current work has focused on write-once publication systems rather than fully-fledged file systems. We wish to investigate how these techniques can be extended to provide the functionality required by a Grid storage service, supporting the file system mutability that is missing from current systems. More complex distributed data structures (such as B* trees) may be required to allow efficient indexing and searching.

A further area we wish to investigate how a Grid storage service could be used to provide shared workspaces to allow ad-hoc collaboration between users.

In traditional file systems users belonging to a particular group or ACL have permission to create and modify files in shared spaces. Such schemes breakdown when there are large numbers of users, particularly when not all our necessarily fully trusted.

One potential solution is to allow each user to have their own personal ‘view’ of shared spaces, where modifications they make are performed in a copy-on-write fashion. This modified view can then be published and made available for other users to use as the basis of their own view, which may consist of the composition of the views of several such users in an overlaying fashion.

We envisage that ‘authorities’ on particular topics will emerge and over time be linked together to form a structure akin to a Google or Yahoo *directory*, that most users will choose to have as their own root view that they extend and customize as desired.

5 Relationship of work with Evolution of Web Service and GRID Services

Looking at the four areas of work in retrospect over the last 20 years we can see a pattern of evolution that one might characterise as *punctuated equilibrium*: in classical distributed computation, we have moved from the Cambridge Distributed System through ANSA and CORBA to OGSA; in the area of collaborative tools we have moved from point-to-point, to multicast, to peer-to-peer; for directory information we have moved from grapevine through DNS to the LDAP systems; and in storage systems we have moved from remote file access of NFS and AFS, to the large scale SANs and p2p storage systems that are emerging now. This project takes a view that we are about to go through a rapid phase shift, and that while continuity for existing eScience services must be assured, the Computer Science community must prepare for the next step-shift into a new distribution paradigm.

In the immediate term, though, we note that there is a large body of work ongoing in the GRID community moving from Web Services through to the GRID. We do not anticipate being on a direct intercept with this for some time, although the working relationship with Microsoft Research may result in some influence on directions in their `.net` technology. MSR have a symmetrical view. However, we believe that our approaches are radical enough in structure that they are likely to be largely complimentary in any case.

We will bring ideas to the relevant standards groups when the time is ripe. One area of direct collaboration with the Cambridge e-Science centre will be through the peer-to-peer Access Grid worked described in section 1.3. The self-organising content distribution overlay network we propose will have clear application to the tele-medicine work ongoing at the centre. We envisage that opportunities for other direct interaction will emerge as the project progresses.

Code and other IP developed by the project will be owned by the University of Cambridge, but it is our intention to release it into the public domain under a BSD-style license.

References

[1] Thomas Anderson, Michael Dahlin, Jeanna Neefe, David Patterson, Drew Roselli, and Randolph Wang. Serverless network file systems. In *Proceedings of the 15th Symposium on Operating System Principles. ACM*, pages 109–126, Copper Mountain Resort, Colorado, December 1995.

[2] C. Baru, R. Moore, A. Rajasekar, and M. Wan. The sdsc storage resource broker, 1998.

[3] Thomas L. Casavant and Jon G. Kuhl. A taxonomy of scheduling in general-purpose distributed computing systems. *IEEE Transactions on Software Engineering*, 14(2):141–154, February 1988.

[4] Miguel Castro and Barbara Liskov. Practical Byzantine Fault Tolerance. In *Proceedings of the 3rd Symposium on Operating Systems Design and Implementation, Usenix Association*, New Orleans, LA, USA, February 1999. USENIX Association, Co-sponsored by IEEE TCOS and ACM SIGOPS.

[5] Edgar Chávez, Gonzalo Navarro, Ricardo Baeza-Yates, and José Luis Marroquin. Searching in metric spaces. *ACM Computing Surveys*, 33(3):273–321, September 2001.

[6] Y. Chu, S. Rao, S. Seshan, and H. Zhang. Enabling conferencing applications on the internet using an overlay multicast architecture. In *Proceedings of ACM SIGCOMM*, 2001.

[7] Y. Chu, S. Rao, and H. Zhang. A case for end system multicast. In *Proceedings of ACM SIGMETRICS*, pages 1–12, June 2000.

[8] Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong. Freenet: A distributed anonymous information storage and retrieval system. In *Workshop on Design Issues in Anonymity and Unobservability*, pages 46–66, 2000.

[9] F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica. Wide-area cooperative stroage with cfs. In *Proceedings of ACM SOSP 2001*, October 2001.

[10] Shlomi Dolev. *Self-Stabilization*. MIT Press, Cambridge, MA, 2000. Ben-Gurion University of the Negev, Israel.

[11] P. Druschel and A. Rowstron. Past: A large-scale, persistent peer-to-peer sotrage utility. In *Proceedings of HOTOS VIII*, May 2001.

[12] Peter Druschel and Antony Rowstron. PAST: A persistent and anonymous store. In *HotOS VIII*, May 2001.

[13] Steven Hand and Timothy Roscoe. Mnemosyne: peer-to-peer steganographic storage. In *Proceedings of the 1st International Workshop on Peer-to-Peer Systems*, March 2002.

[14] Steven Hand and Timothy Roscoe. Spread spectrum storage with mnemosyne. In *FuDiCo 2002: International Workshop in Future Directions in Distributed Computing*, 2002.

[15] Timothy L. Harris. A pragmatic implementation of non-blocking linked lists. In *Distributed Computing, 15th International Conference*, volume 2180 of *Lecture Notes in Computer Science*, pages 300–314. Springer-Verlag, October 2001.

[16] Timothy L Harris, Keir Fraser, and Ian A Pratt. A practical multi-word compare-and-swap operation. In *Submitted for publication*, April 2002.

[17] J. Howard, S. Menees M. Kazar, D. Nichols, M. Satyanarayanan, R. Sidebotham, and M. West. Scale and performance in a distributed file system. *ACM Transactions on Computer Systems*, February 1988.

[18] Klaus Krauter, Rajkumar Buyya, and Muthucumaru Maheswaran. A taxonomy and survey of grid resource management systems for distributed computing. *Software Practice and Experience*, 32(2):135–164, February 2002.

[19] Chuang Liu, Lingyun Yang, Ian Foster, and Dave Angulo. Design and Evaluation of a Resource Selection Framework for Grid Applications, 2002. Submitted for publication.

[20] T S Eugene Ng and Hui Zhang. Predicting internet network distance with coordinates-based approaches. In *Proceedings of IEEE INFOCOM 2002*, 2002.

[21] Fernando Pedone and Svend Frolund. Pronto: A Fast Failover Mechanism for Off-the-Shelf Commercial Databases. Technical Report HPL-2000-96, HP Laboratories, July 2000.

[22] Marcelo Pias, Jon Crowcroft, Steve Wilbur, Tim Harris, and Saleem Bhatti. Lighthouses for scalable distributed location. In *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS '03)*, febuary 2003.

[23] C Greg Plaxton, Rajmohan Rajaraman, and Andrea W Richa. Accessing nearby copies of replicated objects in a distributed environment. In *Proceedings of ACM SPAA*, June 1997.

[24] Ian A Pratt, Timothy Moreton, and Timothy L Harris. Storage, mutability and naming in pasta. In *2002 International Workshop on Peer-to-Peer Computing (to appear)*, April 2002.

[25] S Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A Scalable Content-Addressable Network. In *Proceedings of ACM SIGCOMM 2001, San Diego, California, USA.*, August 2001.

[26] Dickon Reed, Ian Pratt, Paul Menage, Stephen Early, and Neil Stratford. Xenoservers: accounted execution of untrusted code. In *Proceedings of the fifth Workshop on Hot Topics in Operating Systems (HotOS-VII)*, 1999.

[27] Antony Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. *Lecture Notes in Computer Science*, 2218:329–350, 2001.

[28] Antony Rowstron, Anne-Marie Kermarrec, Miguel Castro, and Peter Druschel. Scribe: The design of a large-scale event notification infrastructure. In Jon Crowcroft and Markus Hofmann, editors, *Networked Group Communication, Third International COST264 Workshop (NGC'2001)*, volume 2233 of *Lecture Notes in Computer Science*, pages 30–43, November 2001.

[29] David Spence and Tim Harris. Xenosearch: Distributed resource discovery in the xenoserver open platform. In *Proceedings of the Twelfth IEEE International Symposium on High Performance Distributed Computing (HPDC-12)*, 2003.

[30] Ion Stoica, Robert Morris, David Karger, Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable Peer-To-Peer lookup service for internet applications. In Roch Guerin, editor, *Proceedings of the ACM SIGCOMM 2001 Conference (SIGCOMM-01)*, volume 31, 4 of *Computer Communication Review*, pages 149–160, New York, August 27–31 2001. ACM Press.

[31] R. Wolski, N. Spring, and J. Hayes. The network weather service: A distributed resource performance forecasting service for metacomputing. In *Journal of Future Generation Computer System 15(5/6):757–768*, 1999.

[32] Ben Y. Zhao, John Kubiatowicz, and Anthony D. Joseph. Tapestry: an infrastructure for fault-resilient wide-area location and routing. Technical Report UCB//CSD-01-1141, University of California at Berkeley, April 2001.