# Issues in Building Agent-Based Computational Grids

Omer F. Rana
Department of Computer Science
Cardiff University
Newport Road, Cardiff CF24 3XF, UK

Luc Moreau
Department of Electronics and Computer Science
University of Southampton
Southampton SO17 1BJ UK

**Abstract**

We emphasise and briefly review existing infrastructure required to realise the Computational Grid, and define such Grid with reference to Knowledge and Information Grids. We then propose an agent-based approach for the Computational Grid, which is centered on providing "services" for managing resources.

## 1 Introduction

> *The "Grid" is an emerging infrastructure that connects multiple regional and national grids to create a universal source of computing power — the work "Grid" was chosen by analogy with the electric power grid, which provides pervasive access to power. We believe that by providing pervasive, dependable, consistent and inexpensive access to advanced computational capabilities, databases, sensors, and people, computational grids will have a transforming effect similar to the electric power grid, allowing new classes of applications to emerge.*
>
> *Ian Foster and Carl Kesselman [14]*

Since the publication of the Grid Manisfesto [14], research activity has dramatically increased in this area. The term "Grid" has become a reference to a large scale pervasive infrastructure into which hardware or software components can be plugged, and which permits easy configuration and creation of new functionality from existing components. The Grid is therefore the underlying infrastructure that enables new information services to be defined, activated, supported and managed in a uniform way.

Research in "Grids" has become an area of active interest, with communities such as the Grid Forum [1] in the US, the E-Grid [2] forum in Europe, and the Asia-Pacific Grid Forum in Australia and Japan, working to unify common interest across the world.

Due to the very generic nature of the Grid, the term "Grid" has started to have different meanings for different people. An organisation based on three layers has been proposed [5], and we will define the *computational grid*, the *information grid* and the *knowledge grid*.

By their ability to adapt to their environments, agents can provide solutions to the very dynamic services required by each of these layers. In this paper, we focus on the Computational Grid, for which we identify existing approaches, we discuss their limitations and suggest research directions for agent-based solutions.

# 2 A Three Layer Model

A three layer model for the Grid infrastructure was described by Jeffery [5] in a strategy document earlier this year. This model has been adopted widely by various research communities in the US and Europe, and we now describe it.

The *Computational Grid*, the lower layer, is primarily concerned with large-scale pooling of computational and data resources. (Alternatively, this lower layer is called the *Data Grid*.) Such pooling requires significant shared infrastructure to enable the monitoring and control of resources in the resulting ensemble. The Computational Grid generalises ideas undertaken in early work on Metacomputing [11], concerned with creating a giant computational environment out of a distributed collection of files, databases, computers, scientific instruments and devices.

The *Information Grid* constitutes the middle layer, allowing uniform access to heterogeneous information sources and providing commonly used services running on distributed computational resources. Uniform access to information sources relies on metadata to describe information and to help integrating heterogeneous sources. The granularity of the offered services can vary, from subroutine or method calls to complete applications. Hence, in scientific computing, services can include the availability of specialised numerical solvers, such as matrix solvers and partial differential equation solvers, to complete scientific codes for applications such as wheather forecast and molecular or fluid dynamics. In commercial computing, services can be statistical routines based on existing software libraries, such as SPSS or SAS, or prediction services which offer coarser grained functionality, such as database profiling or visualisation services. In hypermedia applications, services can be multimedia content analysis algorithms or hyper-link servers [21]. Services can therefore be offered by individual providers or by corporations; they may be specialised for specific applications, such as genomic databases, or general purpose, such as numerical libraries.

The *Knowledge Grid* is the top most layer and provides specialised services which can look for patterns in existing data repositories, and manage information services. The knowledge grid is aimed at creating new, value added services which cannot be defined as a single service: they involve an aggregation of many different types of services, providing a correlation between data sets generated by different services, or combining results of existing services in novel ways.

It is intended that each of these layers provide services to various applications, ranging from support for mobile devices, to large scale single applications such as modelling protein folding and concurrent engineering.

A substantial part of the research effort dedicated to the Grid has concentrated on the Computational Grid. The primary reason is that the Physics community is an important "customer" of the Grid, which they use in order to process huge amounts of data, such as

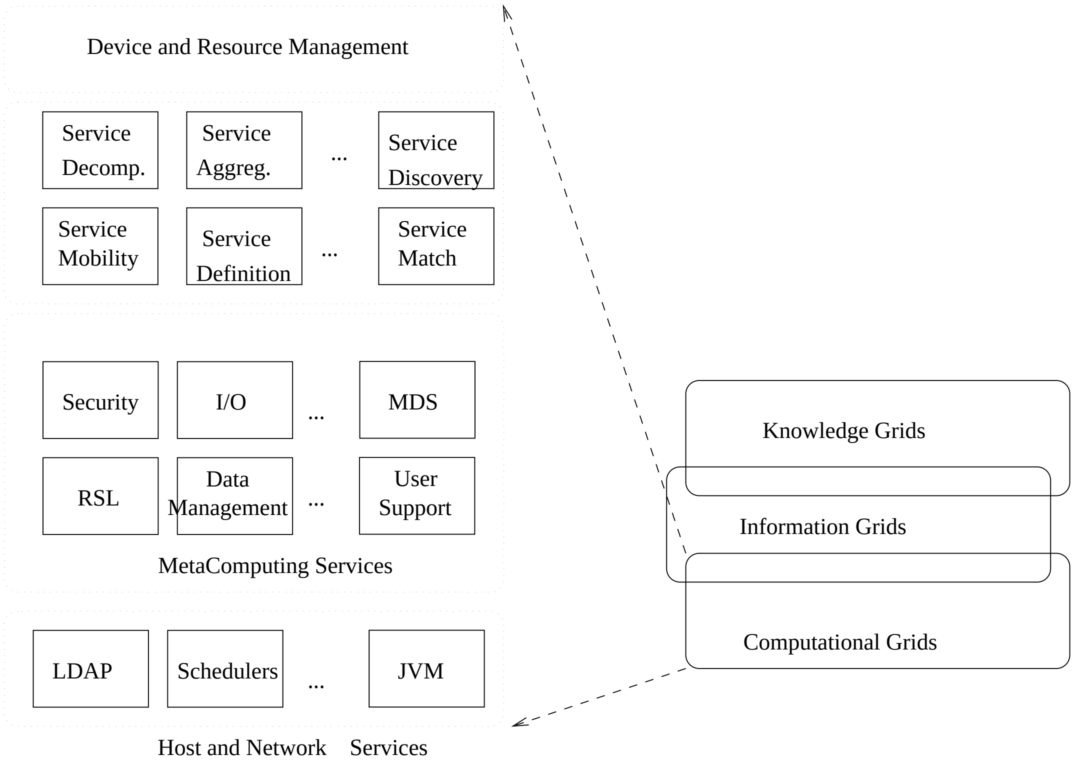


Figure 1: *Grids and the Computational Grid*

generated by the Large Hadron Collider (LHC) experiments to be undertaken by CERN in Switzerland. The data volumes are of the order of Petabytes per year, and must be distributed to scientists and engineers across the world.

However, initiatives such as the "Semantic Web" [3], or research in collaborative environments [22], clearly investigate fundamental issues that are relevant to the Information and Knowledge Grids.

Agents [28] can provide a useful abstraction at each of the three Grid layers. Indeed, by their ability to adapt to the prevailing circumstances, agents will provide services that are very dynamic and robust, and therefore suitable for a Grid environment. However, in this paper, we concentrate on the first layer, outlining how agents can support and extend existing computational infrastructures.

Our emphasis on the lower layer is primarily due to the lack of agreement about standards for services to be supported within Information and Knowledge Grids at present. The resource management problem has been investigated in detail, and a considerable research effort has been expended to create and support standards in this area (such as LDAP). However, what is lacking in current Computational Grid efforts, is the lack of service discovery and management, which an agent based paradigm can offer. We feel that a small improvement at this layer can have a significant impact on the higher level Information and Knowledge Grids; therefore, we show how agents can be used to support and extend the current Computational Grid infrastructure.

# 3 The Computational Grid

In this section, we identify some of the current applications of the Grid and derive some requirements. We discuss some of the services of the computational grid which have to meet these requirements.

## 3.1 Overview

The Computational Grid aims to provide the infrastructure for integrating computational resources to form a single virtual machine, or enable solving large scale problems that cannot be solved on a single system. Examples include distributed interactive simulation such as military simulations, and simulation of physical processes such as climate modelling. It may be useful to note that such resource integration may not lead to performance improvements, instead providing high throughput computing to improve resource utilisation. Applications that could benefit from this approach include the use of multiple distributed workstations to solve hard cryptographic or complex design problems.

The Computational Grid may also be used to provide on-demand computing and to meet short-term requirements for resources that cannot be cost-effectively or conveniently located locally. The resources may provide computation capabilities, or may include software and data repositories, specialised sensors and other devices. Unlike existing applications based on distributed supercomputing, these new application domains are often driven by cost-performance concerns rather than absolute performance. Particular challenges in these applications are the existence of dynamic resource requirements, and the potentially large population of users and resources involved; in particular, location, scheduling, code management, configuration, fault-tolerance and security deserve some investigation. To support some of these requirements, we define a service layer that can benefit from existing monitoring and management services within Metacomputing systems. Such a service layer can include (1) management services, such as locating devices, migrating jobs to devices, load balancing on devices etc, or (2) capability services, such as matrix solving or other linear algebra operations, running a molecular dynamics or fluid dynamics code etc. Hence, we couple management services currently available in Metacomputing systems, as illustrated in figure 1, with a layer that can aggregate and combine these with software libraries that offer a particular functionality for applications.

The Computational Grid may also provide support for data-intensive computing, in which the grid is used to synthesise new information from data maintained in geographically-distributed repositories, digital libraries, and databases. Challenges in this class of applications are the scheduling and configuration of complex, high-volume data flows through the network and multiple levels of processing. Collaborative and concurrent engineering provides one example of such design activities, enabling groups of users to collaborate within virtual environments. In many cases, these applications involve providing the participants with shared access to data and computational resources.

### 3.2 Computational Grid Components

Figure 1 provides our perspective on the Computational Grid, with reference to Information and Knowledge grids identified above. As illustrated, our model of Computational Grid comprises three layers of services: (1) host and network services, (2) meta computing services and (3) device and resource management. These layers are based on existing implementation infrastructures, such as Java/Jini [12] and Metacomputing systems such as Globus [9, 14] or Legion [10].

At the lowest level, we find a set of resources and local schedulers. Resources can be single processor machines managed by an operating system, or multiprocessor supercomputers managed by batch queueing systems such as LSF or Codine. Resources may also include mobile devices containing a Java Virtual Machine (JVM) or other embedded controllers, such as Digital Signal Processor (DSP) chips. Each individual resource or cluster is responsible for managing tasks locally through the use of a local scheduler.

The second layer is composed of a set of Metacomputing services relying on local resources and scheduler. In particular, the Metacomputing Directory Service (MDS) is responsible for translating device characteristics into a searchable tree, comprising device identifiers, parameters such as CPU speeds, local memory etc. Devices can be managed within administrative domains. Security services can also be provided at this level, offering authentication based on certificates, or lower level encryption techniques such as SSL.

At the third level, we then find higher level services, for service discovery, service advertisement and service management.

## 4 Agent Based Services

Grids use existing infrastructures wherever possible, especially if standardisation work is already in progress or completed. We review some of the technologies which could contribute to the Computational Grid, analyse their limitations, and suggest agent-oriented solutions.

### 4.1 Resource and Service Discovery

Resource discovery exists within many Metacomputing toolkits, such as Globus and Legion, and is primarily concerned with locating devices based on their IP address within a domain or sub-domain. The Lightweight Directory Access Protocol (LDAP) based hierarchical naming scheme has been widely used to register devices, enabling new devices to be registered, and available devices to be discovered at run time. Such a resource discovery service can be used by a resource management system to identify possible devices for task placement and execution.

Existing resource discovery services can only be used to locate available devices for executing tasks, and generally do not provide details of the software available on these devices. In some limited cases, software details are made available, but these are generally restricted to operating system information, such as the Condor system [13].

Hence, resource discovery *needs to be extended* with service discovery, whereby service level information is also included in a resource specification. An agent framework can be used to provide such service discovery, based on a Service Advertiser agent, a MatchMaker agent, and a Service Request agent [6, 7]. Service advertisements can range from methods or procedure calls, to more complex signatures specifying service types and categories. The match making process can also vary in complexity from exact matching of services, to matching based on category or parameters. Furthermore, agent frameworks provide various mechanisms by which components may be kept informed, such as asynchronous notifications or the publish/subscribe paradigm [24].

**Jini** The Java programming language provides many features conducive to the development of the Computational Grid, most significant of which is the Jini API. The 'Lookup-Discovery' protocol in Jini [15], along with the 'Leasing', 'Transactions' and distributed 'Events' services can be used as underlying infrastructure for developing resource and service discovery, service management and migration, and service decomposition and delegation.

For instance, the Leasing service can be used to grant licensing periods, over which a user or a service can invoke another service. On the expiry of this period, access to the service is withdrawn, requiring the user or service to acquire a new lease. Such functionality in core infrastructure can be useful for building more complex services, such as match making and service advertising. Agent frameworks such as SoFAR [24] use such a concept of lease in the semantics of registration and notification.

However, issues such as stability, security and scalability are lacking in the Jini API at present, and these need to be addressed before it can be deployed within the Computational Grid. Furthermore, service discovery and matching in Jini is based on an exact syntax match between service request and service interface, and based on Java primitive or derived types. This may not be adequate when service requests must be decomposed, or delegated to other providers.

The Jini API can also be used in collaboration with the JavaSpaces API to provide distributed shared memory support for sharing services. JavaSpaces is based on the concept of tuple spaces in Linda, and enables service types to be grouped, and matched on type signatures. A similar service from IBM called TSpaces, also provides a similar functionality, and may be used in collaboration with Jini. This idea of tuple space has been reused to coordinate the activities of mobile agents [20].

## 4.2 Service Specification

A shared data model or framework is required to enable resources and services to be described. This is necessary to match service requests with service availability, for service decomposition, and for new services to be made available. The DARPA Agent Markup Language (DAML) [19] provides a constraint language to enable the specification of services, and offers a useful tool for encoding services. No standard data models or ontologies exist at present, however, for specifying resource capabilities and services. Various projects, such as Globus and Legion have defined their own specification languages, although these are restricted to specifying resource capability only, and do not tackle the more difficult task of defining services.

It is unlikely that a common ontology will emerge, suggesting that negotiation between service providers and service users is more likely to be the prevalent scenario. Dynamic ontology creation is important in this context, to enable users and providers to use common terms or themes to negotiate and propose services. An important consideration in this context is the time to reach agreement: negotiation protocols should be chosen in function of their cost and their impact on the scalability of the Computation Grid.

## 4.3 Metacomputing Frameworks

Metacomputing frameworks generally involve support for aggregating various aspects of resource management, such as scheduling, security, communication, resource location, resource allocation, process management, and data access. Resources can be tightly coupled parallel machines, or loosely coupled networks of workstations. Metacomputing frameworks do not provide local schedulers or resource managers, but enable multiple resource managers to be integrated at a higher level. In this way, local control and resource management policies can be respected, whilst enabling large scale applications to be executed on distributed computing infrastructure.

The Globus project [9, 14] is one example of a Metacomputing framework, and is based on the assumptions that, (1) grid architectures should provide basic services, but not prescribe particular programming models or higher-level architectures, (2) grid applications require services beyond those provided by today's commodity technologies.

Legion [10] provides another such environment, where a collection of workstations, vector and parallel machines connected by local area and larger-scale networks appears to the user as a single computer. Legion uses object-oriented design techniques to specify resource capability, their location and deployment information, and means of accessing them. The Legion architecture defines a complete object model that includes object abstractions for compute resources (called host objects), storage systems (called data vault objects), as well as other object classes. Users can use inheritance to specialise the behaviour of these objects to support specific requirements, as well as to develop new objects. The use of reflection (the representation of parts of the underlying system as objects that can be directly operated on to access and change system behaviour) is particularly important in Legion. For example, host objects represent Legion processors. One or more host objects run on each computing resources included in Legion. These objects create and manage processes for application-level Legion objects. Object classes invoke the operations of host objects to activate their instances on the computing resources that the host objects represent. Representing computing resources as Legion objects abstracts the heterogeneity of different host computing platforms, and allows resource owners to manage and control their resources within the context of the system.

Metacomputing frameworks available at present are primarily concerned with resource discovery and management. There is little or no support available to specify services. They provide useful infrastructure for building agent based service discovery and migration techniques.

## 4.4 Mobile Agent Libraries

Mobile agents also provide a useful abstraction for supporting service migration and mobile services in the Computational Grid. Existing libraries such as Voyager (ObjectSpace) [4], Aglets (IBM Research) [8], D'Agents (Dartmouth College) etc, provide well defined calls to enable a service to be migrated, checkpointed, and run on remote machines. Virtually all of these libraries still rely on the existence of a "place" process on the remote host to receive in-coming code. No support is provided to bootstrap or initialise new hosts at run time into a cluster in any of these libraries.

Additional work is needed to enable mobile agents to be deployed within the Computational Grid, and ranges from support for managing "forwarders" during service migration [25], maintaining state consistency after migration, checkpointing the complete execution state, selective data migration after service migration, and mechanisms to deal with security, related both to the host receiving the mobile code, and the integrity of the mobile agent itself. All of these requirements will bring security, adaptability and robustness to mobile agent systems; their deployment in the Computational Grid is still conditional to the scalability of their supporting infrastructure.

We can identify common agent services in the context of the Computational Grid, which may be shared across applications. These services can be offered centrally and shared, or services may be federated, with one service in each resource cluster. The objective of defining such agent services is to identify common usage patterns across resources, which can vary in type and capability, and to combine expertise at a single place.

## 4.5 High-Level Multi-Agent Interactions

All the services described before would benefit of some agent-based approaches, but we reckon that the benefit of agents would further be visible in higher-level services that involve the collaboration of several nodes in the computational Grid.

Two forms of interactions between autonomous agents are generally distinguished. *Cooperation* (cooperative problem solving) is the process by which a group of agents choose to work together to achieve their goal [27]. *Negotiation* is the process by which a group of agents communicate with one another to try and come to a mutually acceptable conclusion [26].

In the general case, resource allocation can be seen as a problem involving the collaboration of several nodes, which have to agree on a satisfactory allocation of resources for the current problem and the prevailing circumstances. We can foresee a very dynamic market-based approach where idle nodes bid for computations to be started on or even migrated to themselves. A similar approach is adopted in the design of new generation mobile phone networks, where software agents are used to trade communication services, e.g. QoS [23]. Multi-agent high-level interactions such as auctions seem therefore worth investigating in the context of the Computational Grid.

### 4.6 Related Work

The DARPA CoABS (Control of Agent Based Systems) project [18] aims to develop agent systems for offering specialised services, such as component interaction managers, database wrappers, traders/brokers, to resource planners and interactions managers. The remit of the project is very wide, and a diverse set of requirements have been identified within the project. The approach proposed in this paper addresses a subset of the CoABS functionalities. The computational grid is more well defined, and contains a more precise set of requirements. There are also more well defined standards which operate at the level of the Computational Grid. However, as the CoABS project matures, it is likely to impact various aspects of the Computational Grid also.

# 5 Conclusion

Establishing and implementing Grids is an important undertaking, and offers new challenges in developing robust and scalable infrastructures. We have identified three types of Grids in this paper, a Computational Grid as core infrastructure for supporting higher level services in Information and Knowledge Grids.

So far, the Computational Grid has been considered primarily as resource specification and management. We have extended this notion to also include service discovery and specification; we have suggested the use of agents, as offering services within such a grid, and also as means to manage existing resources and integrate existing frameworks. We have presented infrastructure tools and frameworks that could be of benefit in establishing the Computational Grid, and we have proposed an agent based architecture for the Computational Grid.

# References


[1] The US Grid Forum, see Web site at: `http://www.gridforum.org`

[2] The European Grid Forum, see Web site at: `http://www.egrid.org`

[3] The Semantics Web `http://www.semanticweb.org`

[4] ObjectSpace. Voyager. `http://www.objectspace.com/`.

[5] Keith G Jeffery, "Knowledge, Information and Data", A briefing to the Office of Science and Technology, UK, February 2000. Available at: `http://www.itd.clrc.ac.uk/ActivityPublications/239`

[6] K. Sycara, J. Lu, and M. Klusch. "Interoperability among Heterogeneous Software Agents on the Internet", Technical report, Carnegie Mellon University, October 1998. Research report no. CMU-RI-TR-98-22.

[7] Omer F. Rana, "Dynamic Resource Discovery through MatchMaking", Proceedings of High Performance Computing and Networking (HPCN). Springer Verlag. Amsterdam, May 2000

[8] Danny B. Lange and Mitsuru Ishima. *Programming and Deploying Java Mobile Agents with Aglets*. Addison-Wesley, 1998.

[9] Gregor von Laszewski, "CoG-Kits: A Bridge Between Commodity Distributed Computing and High-Performance Grid", Mathematics and Computer Science Division, Argonne National Laboratory, see Web site at: `http://www.globus.org`,2000.

[10] S. Chapin, J. Karpovich and A. Grimshaw, "The Legion Resource Management System", Proceedings of the 5th Workshop on Job Scheduling Strategies for Parallel Processing at IPDPD99, San Juan, Puerto Rico, April 1999.

[11] G. Fox and W. Furmanski, "Petaops and Exaops: Supercomputing on the Web", IEEE Internet Computing 1(2), March-April 1997.

[12] R. Ashri and M. Luck, "Paradigma: Agent Implementation through Jini", Proceedings of the Eleventh International Workshop on Database and Expert Systems Applications, IEEE Computer Society Press, 2000.

[13] R. Raman, M. Livny, and M. Solomon. "Matchmaking: Distributed Resource Management for High Throughput Computing", Proceedings of the Seventh IEEE International Symposium on High Performance Distributed Computing, July 1998.

[14] I. Foster and C. Kesselman (eds.), "The Grid : Blueprint for a New Computing Infrastructure", Morgan Kaufmann, 1999.

[15] W. Keith Edwards, "Core Jini", Addison Wesley, 1999.

[16] Omer F. Rana and David W. Walker, "Agent based Resource Integration in PSEs", Proceedings of HICSS34, January 2000. (to appear)

[17] Frank Manola, "Characterizing Computer-Related Grid Concepts", Technical Report, Object Services and Consulting, Inc, December 1998. See Web site at: `http://www.objs.com`.

[18] The CoABS DARPA Project, "Control of Agent Based Systems", see Web site at: `http://coabs.globalinfotek.com/`.

[19] The DARPA Agent Markup Language (DAML), see Web site at: `http://www.daml.org`.

[20] G. Cabri, L. Leonardi, and F. Zambonelli. Reactive Tuple Spaces for Mobile Agent Coordination . In *Proceedings of the 2nd International Workshop on Mobile Agents (MA'98)*, number 1477 in LNCS, 1998.

[21] David De Roure, Nigel Walker, and Leslie Carr. Investigating Link Service Infrastructures. In *Proceedings of the Hypertext Conference HT'00*, 2000.

[22] Dave DeRoure, Wendy Hall, Siegfried Reich, Aggelos Pikrakis, Gary Hill, and Mark Stairmand. An open architecture for supporting collaboration on the web. In *WET ICE '98 — IEEE Seventh International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, June 17-19, Stanford University, California, USA*, pages 90–95, 1998.

[23] Gwenael Le Bodic, Demessie Girma, James Irvine, and John Dunlop. Dynamic 3G Network Selection for Increasing the Competition in the Mobile Communication Market. in *Proc. IEEE Vehicular Technology*, Boston, MA, September 2000.

[24] Luc Moreau, Nick Gibbins, David DeRoure, Samhaa El-Beltagy, Wendy Hall, Gareth Hughes, Dan Joyce, Sanghee Kim, Danius Michaelides, Dave Millard, Sigi Reich, Robert Tansley, and Mark Weal. SoFAR with DIM Agents: An Agent Framework for Distributed Information Management. In *The Fifth International Conference and Exhibition on The Practical Application of Intelligent Agents and Multi-Agents*, pages 369–388, Manchester, UK, April 2000.

[25] Luc Moreau. Distributed Directory Service and Message Router for Mobile Agents. *Science of Computer Programming*, Accepted for publication.

[26] P. Faratin, C. Sierra, and N. R. Jennings. Negotiation decision functions for autonomous agents. *Int. Journal of Robotics and Autonomous Systems*, 24(3 - 4):159–182, 1998.

[27] M. J. Wooldridge and N. R. Jennings. Cooperative problem solving. *Journal of Logic and Computation*, 9(4):563–592, 1999.

[28] M. Wooldridge and N. R. Jennings. Intelligent Agents: Theory and Practice. *Knowledge Engineering Review*, 10(2), June 1995.