

A “Possible Future” for the Grid

Geoff Coulson and ...

Computing Department, Lancaster University, Lancaster, LA1 4YR
geoff@comp.lancs.ac.uk

1. Introduction

The various other papers in this special issue paint a nicely representative picture of current research issues in grid computing. In this article, our purpose is different; rather than focus on the present or short-term future, we want to articulate a vision of what grid computing *might* be like in around 10 to 15 years time. This ‘potential future’ is necessarily partial and influenced by our personal interests and background. For example, while acknowledging its considerable importance for the future, we omit consideration of the semantic grid [Semantic,06]. We also largely leave aside security issues. Our emphasis, in keeping with our research interests, is on the architecture of the ‘low-level’ grid infrastructure in terms of computer systems, their interconnectivity, and their system level software.

The motivation for offering this vision is to counterbalance, in a small way, what we see as an over-emphasis in grid research on short-term development. This is caused largely by the funding-driven necessity of developing deployable applications while keeping up with the ever-moving target of grid middleware standards wars. While pragmatic application development is indeed important and necessary, we feel that the long-term health of grid computing requires a complementary thread of architectural thinking that is unconstrained by short-term pragmatic imperatives.

2. A “Possible Future” for the Grid

Let us begin with a working definition of grid computing which is broad enough to encompass all the possible developments we sketch out below. We see grids as (potentially global) distributed computing infrastructures that lease certain kinds of resources to users. These resources are often computational, but can equally well encompass storage, connectivity, data, sensors, actuators, or human presence. The resources offered by grids are pooled from offerings made by a variety of contributing parties (ranging from individual users to virtual organisations) who may wish to impose conditions on the useage of their resources and to dynamically donate and withdraw them as they see fit. On the basis of this pool, grids build an integrated, resilient, flexible, and secure resource leasing service. In particular, they enable users to discover and obtain resources both by pre-booking and on demand, and offer a range of guarantees on leased resource in terms of ‘ilities’ such as quantity, quality,

availability, security, integrity, dependability, etc. These are offered on a sliding scale according to factors such as service level agreements, user priority levels, or willingness to pay (whether in hard or notional currency).

We start the description of our ‘possible future’ at the ‘bottom’ level of devices and networks. Whereas the focus of present-day grids is on relatively highly-powered and centralised compute nodes connected using high-speed networks, future grids will be far more heterogeneous and *pervasive*. For a start they will seamlessly encompass the types of system exemplified by SET@home [Korpela,01]—i.e. systems that exploit lower-powered and less well-connected compute resources that are, however, available in significantly large numbers. In addition, future grids will encompass *mobile devices* such as PDAs and mobile phones (or a future integrations of these currently separate technologies). These devices, interconnected with various types of wireless networks, will be important in the context of collaborative resource useage carried out by ad-hoc task-oriented groups of people such as rescue workers, military forces, sports crowds, or travellers on trains or aircraft. Such groups will want to call on grid-provided resources both ‘locally’ (e.g. to access specialised peripherals in the local area) and globally (e.g. to execute compute-intensive jobs on behalf of the group; see below).

Beyond this, *sensor networks* (employing a diverse range of both wired and wireless networks including 802.11, 802.15.4, Bluetooth, microwave links, mains power networks, etc.) will increasingly be integrated into grids. Sensor networks have the potential to radically extend the reach of grids to encompass real-time connection to the environment. In particular, they offer the potential to track and predict environmental phenomena such as weather and climate; geological phenomena; the spread of fire, flood or pollution; or the movement of people or animals. In many cases, the outputs from sensor networks will be closely linked to large-scale computations running on ‘traditional’ grid-based computational resources. In other cases, however, as with the mobile computing scenarios mentioned above, parallel and distributed computation may also be carried out ‘locally’ using the nodes of the sensor network as a grid or sub-grid. An example of this from our research at Lancaster is the use of networked sensor nodes to compute the flow rate of a river, based on an analysis of video images of the movement of flotsam on the river’s surface [Hughes,06]. This particular computation *must* be carried out locally because the sensor network’s connectivity (GSM-based) to the wider grid, although sufficient to carry other sensor data such as pressure and salinity, is insufficient to support the data rates necessary to ship raw video data offsite.

The above factors lead to an overall picture of an increasingly *diverse* grid in terms of devices, inter-connecting networks and, indeed, applications. But what kind of system-level software is required to support such applications running over such infrastructures? It is pretty clear that present-day grid middleware such as Globus 2, OGSA, or the WSRF is inadequate to this task. This is due to factors such as: the high levels of heterogeneity expected in both end-systems and networking infrastructures; the large scale; the high complexity; the need for real-time interactive and multi-party collaboration; the multiple media-types; the QoS-sensitivity; and the need for dynamic reconfiguration in response to environmental change.

Future grid middleware will accordingly be based on quite different premises to the present generation. For one thing, it will be highly *configurable*, so that it can be executed on anything from a microcontroller-based sensor mote to a supercomputer. One way to provide such configurability is to structure *all* system-level functionality, including networking protocols, operating system functions, middleware, and even applications, in terms of dynamically composable software components (see e.g. [Bruneton,04], [Coulson,04] or [Furmento,02]). Note that such an architecture does not necessarily preclude the use of legacy software (e.g. operating systems) as these can be ‘wrapped’ as components. Such a component –based approach can provide a means of integrating functionality on a per-system, per-need, basis so that systems can ‘profiled’ by composing in or composing out different individual pieces of functionality as appropriate to the target system type, target device type, or target application. This largely comes about because component models are typically *reflective*—e.g. they make explicit the dependencies on components on other components, and can thus be dynamically deployed by external third-parties. Also, they are often programming language independent so they can be used to build systems from multi-language pieces.

As well as supporting static configurability, component-based structures also offer a natural basis for *dynamic reconfiguration* of system software by replacing or adding components at run time. In the future, once basic issues of integrity maintenance and security have been sufficiently addressed, we see these fine-grained component compositions as forming the ‘mechanism layer’ of a broader autonomic adaptation architecture that initiates reconfiguration operations on the basis of suitable triggers and policies.

All of the above is concerned with how grid middleware is internally structured in support of configurability and reconfigurability. But what is the basic *functionality* that will be offered by future grid middleware? We believe that this will be in terms of resource allocation and management (as is presently the case) but also in terms of *communication and interaction services*.

In the first area (i.e. resource allocation and management), we see a significant generalisation of the services offered by current resource management middleware. In the future, resource management middleware will not only deal with the ‘expanded’ notion of resource implied in the definition above, but will also manage resources in a more dynamic and fine-grained manner. Dynamicity is especially needed by the pervasive grid scenarios sketched above. If we are, say, tracking and predicting the progress of a wildfire by feeding sensor data into a large scale compute-intensive simulation [Grace,05] we need to be able to obtain compute resources *on demand* to support real-time computational steering of the simulation by live sensor data. This requires resource management middleware that is far more agile and responsive than present-day technologies [Cai,05]. This need for dynamicity also points to the need for *finer-grained* resource allocation. If a whole processor cannot be found when our wildfire simulation needs more resources, then maybe it is possible to find 25% of 4 processors—perhaps by downgrading the priorities of non-real-time computations that are currently running on them.

In the second area (i.e. communication and interaction services), we see future grid middleware offering a much wider range of *interaction paradigms*. Present-day middleware basically offers remote procedure call and maybe one-shot messaging; but in the future we will see the widespread deployment of a range of other interaction paradigms such as: RPC-with-QoS, messaging-with-QoS; multicast; group communication-with-QoS; publish-subscribe; tuple-spaces; peer-to-peer interaction; and media-streaming-with-QoS (we also see the increasingly-recognised workflow paradigm being integrated as just one more interaction paradigm). This increased richness and sophistication in interaction paradigms is driven by a range of factors such as the generalised notion of resource outlined in the above definition of grid computing (e.g. the need to interact with sensors), the integration of mobile computing and sensor networks into the grid, and an increasing emphasis on the grid being used in collaborative modalities as well as the ‘single-user batch’ model that we are currently familiar with.

The final aspect of our vision of future grid infrastructure software concerns the way in which functionality provided by grids will be made visible to users—i.e. the levels at which programming APIs are offered. Note first that the provision of programming APIs is largely *orthogonal* to the foregoing discussion—a range of styles of API (including present-day APIs based on the WSRF etc.) can equally well be provided on top of the above sketch of a pervasive grid offering a range of resource types and a range of means of interaction between the constituent pieces of a distributed computation.

As argued in a recent paper [Brebner,06] there are *two* main styles of API that are currently emerging in grid computing. The first of these is a *service-centric* style in which the services offered by the grid are directly offered as first class entities (e.g. as Web Services). The second is a *resource-centric* style in which the grid exposes ‘container’-based APIs into which user applications can be deployed and executed. Our view is that the future grid will employ both of these styles. Here, however, we will focus on the resource-centric style as we guess that this will become increasingly

prevalent. In the future, we see the notion of a ‘container’ being radically generalised to cover everything from a JVM to a Xenoserver [Kotsovinos,04] to a cluster to a Virtual Organisation, and the kinds of entity (component) that can be deployed into a container being similarly generalised. Furthermore, containers could be organised hierarchically so that a single uniform container model and API is exposed at every level. Containers will serve both as resource providers and as security scopes.

3. Conclusions

We do not know if the vision outlined above will bear much or any resemblance to what we will actually see in 10 to 15 years. However, we hope it represents at least a coherent sketch of one possible future. If more space were available it would be interesting to flesh out the proposed vision and also (in collaboration with others, perhaps) to consider the areas that we have explicitly omitted from consideration— e.g. resource discovery through approaches such as semantic web and ontologies; the crucial issue of security; and the provision of a market-place for grid resources, etc.). At the least we hope we have provided a sufficiently plausible ‘straw man’, the burning of which might spark the emergence of productive thinking about the longer term development of grid computing.

References

- [Semantic,06] Definition of the Semantic Grid, http://en.wikipedia.org/wiki/Semantic_Grid, 2006.
- [Korpela,01] Korpela, E., “SETI@home - Massively Distributed Computing for SETI”, Computing in Science & Engineering, pp78-83, January 2001.
- [Hughes,06] Hughes, D., Greenwood, P., Coulson, G., Blair, G., Pappenberger, F., Smith, P., Beven, K., “GridStix: Supporting Flood Prediction using Embedded Hardware and Next Generation Grid Middleware”, 4th IEEE International Workshop on Mobile Distributed Computing (MDC 2006), co-located with WoWMoM, 2006.
- [Bruneton,04] Bruneton, E., Coupaye, T., Leclerc, M., Quema, V., Stefani, J-B, “An Open Component Model and its Support in Java”, 7th Intl Symp. on Component-Based Software Engineering (ICSE-CBSE7), Edinburgh, Scotland, May 2004.
- [Coulson,04] Coulson, G., Blair, G.S., Grace, P., Joolia, A., Lee, K., Ueyama, J., “A Component Model for Building Systems Software”, Proc. IASTED Software Engineering and Applications (SEA’04), Cambridge, MA, USA, Nov 2004.
- [Furmento,02] Furmento, N., Mayer, A., McGough, S., Newhouse, S., Field, T., Darlington, J., “ICENI: Optimisation of Component Applications within a Grid Environment”, Journal of Parallel Computing, Feb 2002.
- [Grace,05] Grace, P., Coulson, G., Blair, G.S., Porter, B., “Deep Middleware for the Divergent Grid”, Proc. IFIP/ACM/USENIX Middleware 2005, Grenoble, France, November 2005.
- [Cai,05] Cai, W., Coulson, G., Grace, P., Blair, G.S., Mathy, L., Yeung, W.K., “The Gridkit Distributed Resource Management Framework”, Proc. European Grid

Conference, Science Park Amsterdam, The Netherlands, February 14 -16 2005, published in LNCS Vol 3470, *Advances in Grid Computing*, ed. Sloot, Hoekstra, Priol, Reinefeld, Bubak, http://dx.doi.org/10.1007/11508380_80, pp786-796, June 2005.

[Brebner,06] Brebner, P., W. Emmerich, W., “Two Ways to Grid: The contribution of Open Grid Services Architecture (OGSA) Mechanisms to Service-centric and Resource-centric lifecycles”, *Journal of Grid Computing*, 2006.

[Kotsovinos,04] Kotsovinos, E., Moreton, T., Pratt, I., Ross, R., Fraser, K., Hand, S., Harris, T., “Global-scale Service Deployment in the XenoServer Platform”, *Proc. 1st Workshop on Real, Large Distributed Systems (WORLDS '04)*, San Francisco, USA, December 2004.