

# Evolutionary Pressures on the TeraGrid and the Larger Grid/Distributed Computing Community

Craig A. Lee  
February 18, 2008

## I. Introduction

This document serves as an effort to catalog the evolutionary pressures on the general grid and distributed computing communities. Assuming that the TeraGrid wants to grow and evolve beyond its current form and user base, these general evolutionary pressures should be applicable. For each of the areas discussed in this document, however, the TeraGrid leadership must evaluate and decide where the applicability is higher or lower. In all cases, though, this should help identify and comprehend the developments and pressures on the entire community.

The rest of this document has the following structure:

- Usage Models
- System Properties
- Strategic Directions
- Final Observations

## II. Usage Models

What is meant here by *usage model* to identify the key goals and methods for how to most effectively engage and serve the user constituency and stakeholders.

\* *Make it easy to use.*

For grids -- or any technology -- to be truly successful, users should not have to know or care that they are actually using a grid. It should fade into the background. (See section on "cloud computing" below.)

\* *Engage end-users, i.e., end-user scientists.*

I'm sure this has been discussed endlessly in TeraGrid but it's true for every organization that is trying to deliver value. Beyond building the basic infrastructure, there has to be genuine adoption by non-computer-scientists and non-computational scientists.

\* *Support a spectrum of spectrum of usage models.*

End-users want simple tools that give them good results and are "worth the trouble". To this end, there must be a spectrum of how users use the infrastructure, based on what services and capabilities they actually need for their project and goals. For simple, experimental prototyping, Web 2.0 mash-ups may be useful. Web 2.0 has no direct support for discovery, security, and other "traditional" grid capabilities, but for many

projects, these are unnecessary "overheads" that diminish the returns and can make a system "more trouble than it's worth". When experimental prototypes "grow-up", however, there should be a path of evolution whereby additional capabilities can be added without having to rearchitect the system from the ground up. How can better security be added when needed? How can catalogs be federated, when needed, that were not intended to be so? This path of evolution should eventually lead a full-featured, traditional, open-ended grid that we all have envisioned for many years.

Hence, there should be a suite of tools that can be used, either separately or in any combination, by users to build just as much of a "grid" as they need, and no more. This notion of a spectrum of usage models can be applied to many functional areas in grids. (Doing an in-depth survey on each of these is beyond the scope of this document, but hopefully the reader will get the idea.):

- *Security.* Basic SSH keys to VO management systems.
- *Cataloging and Discovery -- Information Services.* Many catalog systems exist and others getting adopted in various communities, e.g., eBRIM. There needs to be some notion of data interoperability, or semantic interoperability, across various communities to enable catalog or registry federation.
- *Data Movement and Management.* See Cataloging and Discovery.
- *Job Submission and Transaction Processing.* Job submission may actually be one of the more mature aspects of grid computing. The HPC Basic Profile is getting vendor adoption. Tools like SAGA and Grid-RPC also provide basic APIs. Transaction processing is something that has not been associated with traditional grids.
- *Programming and Workflow Management.* Simple programming tools may actually go a long way for addressing basic workflow management.
- *User Interfaces and Environments.* Clearly domain-specific portals and the like should make it easier for end-users. An interesting example from Dave DeRoure is the researcher who builds a web page front end for a canned workflow where the web page URL can be emailed to a user who only has to specify input files and push one button to initiate the workflow. This is an attractive level of simplicity. Clearly this requires that a lot of stuff be pre-configured and some flexibility is lost, but again, simplicity is valuable.
- *Policy: Definition, Monitoring and Enforcement.* This is something that end-users may never want to think about but as distributed systems get larger and more complex, they will be managed through policy mechanisms rather than by direct intervention. End-users may want to express the management of their data and processing through policy. I think systems such as iRODS will get wider usage, and not just for archival data management. Coupling a policy rule engine

with user-definable services is a powerful approach that can be applied in many different areas.

- *Intelligence.* Policy rule engines are actually an example of the application of "intelligence". Making systems more flexible means that less and less information is "compiled-in" or hard-coded, and must be discovered at run-time. Also, the global state of a large, distributed system will not be knowable. A specific site in a large system will only have knowledge within some "local" region, and that information will be unreliable to some degree. Autonomic or some other inference engines will be necessary to manage such systems.

From an architectural viewpoint, supporting a range of usage models could mean providing both an installed infrastructure with a full set of well-defined services, and also something akin to a cloud. One view of cloud computing is that a cloud computer is an ad hoc grid on the back-end but with a simple, well-defined API that essentially virtualizes all resources for the end-user. Virtualization, and contextualization for the management of sets of virtual machines, promises to be a powerful architectural approach whereby system utilization can be maximized, while end-users do not have to worry about maintaining their own physical resources.

This can make adopting and using a grid very simple since none of the administrative or maintenance issues have to be addressed by the end-user. The downside, of course, is that as architectural information is abstracted away (virtualized), many performance issues may arise that the end-user cannot control nor manage. This trade-off between flexibility and performance, however, is fundamental to all systems, and for some segment of users, this will be a non-issue. The issue facing TeraGrid to understand their "market demographics" and what "flavors" of distributed computing can be effectively supported that means the broadest needs.

### **III. System Properties**

OGF recently conducted a Stakeholder Survey and the following items ranked high on the list of desired system properties:

- Scalability
- Interorganization Collaboration
- Availability
- Application Development
- Maturity
- Integration
- Management
- Cost

While these properties may seem like they are "universally good", TeraGrid may nonetheless wish to consider and rank them according to priority when considering where to invest resources in further development.

## IV. Strategic Directions

### *\* Be Stakeholder-Driven.*

I'm sure this has also been endlessly discussed. In the academic and research communities, it is natural to pursue broad, fundamental technical issues associated with distributed infrastructures, middleware, and applications. When serving a set of stakeholders, however, all development should be driven by stakeholder needs and requirements, rather than general computer science research goals. I'm sure the question has been posed, "Who are our real stakeholders?". The end-users? The funding agencies? The system builders? Are these groups only acting as "proxies" for the *real* stakeholders? How an organization connects with its real stakeholders is an endless question. "Build it and they will come" does not usually work. Outreach and engagement must be a continual process.

### *\* Natural Adjacencies.*

All projects start with a target audience. As projects progress, however, so does the larger "landscape" of requirements and capabilities. The original audience may be not the only demographic anymore. Hence, it is important to look for *natural adjacencies* to the currently perceived constituency.

In the context of OGF, a collaboration has been started with the Open Geospatial Consortium. OGC has a large community around geospatial data that use computers, but are not primarily computer scientists who focus on computing infrastructures. Hopefully this is a natural adjacency for OGF since OGC has defined a number of adopted standards for cataloging and presenting geospatial data, but want to integrate those tools w/ distributed resource management capabilities, e.g., grids. If TeraGrid can identify a similar application domain and can use the infrastructure built, perhaps synergy can be created.

## V. Final Observations

Many of the topics mentioned here could very well have an architectural impact on the future TeraGrid. Virtualization/Contextualization/Cloud computing could possibly be in this category. Other topics, like multicore, may certainly be incorporated but will probably have less of an architectural impact that end-users would see or care about. Likewise, green IT or green grid is growing in importance, but may not be the most important system property that TeraGrid should pursue -- simplicity, ease-of-use, and connecting with end-users may be the most important issue.

Clearly, much more time could be spent in drilling down into specific for each of the comments made here. I leave this to the TeraGrid team to rank these comments and allocate resources accordingly for further study.