The Importance of Data Locality in Distributed Computing Applications

Alex Szalay, Julian Bunn, Jim Gray, Ian Foster, Ioan Raicu

Current grid computing environments are primarily built to support large-scale batch computations, where turnaround may be measured in hours or days – their primary goal is not interactive data analysis. While these batch systems are necessary and highly useful for the repetitive '*pipeline processing*' of many large scientific collaborations, they are less useful for subsequent scientific analyses of higher level data products, usually performed by individual scientists or small groups. Such exploratory, interactive analyses require turnaround measured in minutes or seconds so that the scientist can focus, pose questions and get answers within one session. The databases, analysis tasks and visualization tasks involve hundreds of computers and terabytes of data. Of course this interactive access will not be achieved by magic – it requires new organizations of storage, networking and computing, new algorithms, and new tools.

As CPU cycles become cheaper and data sets double in size every year, the main challenge for a rapid turnaround is the location of the data relative to the available computational resources – moving the data repeatedly to distant CPUs is becoming the bottleneck. There are large differences in IO speeds from local disk storage to wide area networks. A single $10K server today can easily provide a GB/sec IO bandwidth, that requires a 10Gbit/sec network connection to transmit. We propose a system in which each 'node' (perhaps a small cluster of tightly coupled computers) has its own high speed local storage that functions as a smart data cache.

Interactive users measure a system by its *time-to-solution*: the time to go from hypothesis to results. The early steps might move some data from a slow long-term storage resource. But the analysis will quickly form a *working set* of data and applications that should be co-located in a high performance cluster of processors, storage, and applications.

A data and application scheduling system can observe the *workload* and recognize data and application locality. Repeated requests for the same services lead to a dynamic rearrangement of the data: the frequently called applications will have their data 'diffusing' into the grid, most residing in local, thus fast storage, and reach a near-optimal thermal equilibrium with their competitor processes for the resources. The process arbitrating data movement is aware of all relevant costs, which include data movement, computing, and starting and stopping applications.

Such an adaptive system can respond rapidly to small requests, in addition to the background batch processing applications. We believe that many of the necessary components to build a suitable system are already available: they just need to be connected following this new philosophy. The architecture requires aggressive use of data partitioning and replication among computational nodes, extensive use of indexing, pre-computation, computational caching, detailed monitoring of the system and immediate feedback (computational steering) so that execution plans can be modified. It also requires resource scheduling mechanisms that favor interactive uses.

We have identified a few possible scenarios from the anticipated use of the National Virtual Observatory data that are currently used to experiment with this approach. These include image stacking services, and fast parallel federations of large collections. These experiments are already telling us that, in contrast to traditional scheduling, we need to schedule not just individual jobs but workloads of many jobs.

The key to a successful system will be "*provisioning*", i.e., a process that decides how many resources to allocate to different workloads. It can run the stacking service on 1 CPU, or 100: the number to be allocated at any particular time will depend on the load (or expected load) and on other demands.