Cloud+X: Exploring Asynchronous Concurrent Applications

Allen McPherson, James Ahrens, Christopher Mitchell
Los Alamos National Laboratory

Position: *So-called "cloud" and "web" technologies should be used to explore new computational paradigms and develop runtime system requirements for next-generation exascale applications.*

As we approach exascale, and discuss programming models, it seems clear that there will not be one uber-language for building codes—instead there will be many that will need to be used in concert to develop useful applications. One oft-discussed approach to exascale development is MPI+X. This two-part model envisions using 'X' to accelerate code on local, heterogeneous node architectures and MPI to construct the overall application and communicate between the nodes running 'X'. Today, 'X' includes languages such as CUDA, OpenCL, Cilk+, TBB, HMPP, OpenMP, etc.—more will follow in the future.

We believe that, while this approach may be made to work, it is unreasonable to expect developers of complex multi-physics codes to schedule their computation onto the machine, dynamically load balance itself, respond to faults, throttle itself based on power conditions, integrate visualization and analysis, debug at exascale with global tools, etc. There will need to be some form of runtime system or "middleware" that assists application developers in construction of these complex applications.

In discussions with our peers, and with vendors, there is some agreement on this notion of a runtime and supporting APIs for architecting large-scale simulations. There is however, no agreement on who is going to build it! Before anything of production quality can be built there must be research to develop paradigms and APIs that the runtime would need to support to enable construction of these applications. We simply propose to *use the broad set of technologies available to "cloud" and "web" developers to accelerate this research*. It's important to note that, at this point, we **do not** advocate the use of these technologies for production codes—there will likely be performance issues that will need to be addressed (via updated, or new, implementations) for our specific domain of high-performance, large-scale scientific simulations.

Along with the use of cloud technology comes a change in computational paradigm. We will need to move to a more loosely coupled, concurrent, asynchronous approach as opposed to today's tightly coupled block synchronous approach. This move is likely necessary anyway, if only as a model for responding to faults. Existing cloud technologies support this model of computation. Some baseline technologies that could be used in this research include:

- Messaging services for communicating between parts of the computation and the system that controls and oversees the execution of the code (e.g. ZeroMQ).

- Programming languages that support concurrency, asynchrony, fault tolerance, hot code loading, moving code to data, etc. (e.g. Erlang).
- Discovery and scheduling services that identify resources in the cluster and orchestrate computations on those resources.
- NoSQL databases (e.g. Riak, MongoDB) that enable complex data models for in situ visualization and analysis. These databases can also be used to journal messages and support live checkpoint/restart. Note that we currently see these as runtime-only databases—other than visualization and analysis, they are unlikely to be backed by rotational storage.

We also believe that this must be a holistic approach. The paradigms, methods, and APIs resulting from the work should support all aspects of the computation including I/O, in situ visualization and analysis, etc. It is even possible to conceive that this approach may enable functionality such as domain specific performance analysis or debugging.

As a first step, we suggest the following research tasks:

1. Investigate technology space including NoSQL databases, messaging systems, programming languages, resource discovery services, etc. Search for other efforts that may also be targeting this approach to identify their successes or failures. In particular, we will carefully read the final report of the Magellan project as a baseline collection of information on previous investigation of cloud technology for scientific computing.
2. Develop a set of demonstration compact applications using these technologies and paradigms. Release as open source early and often.
3. Extract relevant abstractions and develop APIs that enable construction of applications. Continuously iterate with compact application development to apply and test these abstractions.
4. Identify critical performance issues with existing technology. Identify potential performance improvements and cost to achieve it.

Any initial successes of this approach should encourage a more broad community effort. Lacking a large investment, it seems unlikely that any one entity will develop the required runtime system—a collaborative effort across DoE labs, universities, and vendors would be required to transition this research to production.