Cloud+X: Exploring Asynchronous Concurrent Applications

Authors: Allen McPherson, James Ahrens, Christopher Mitchell

Date: 4/11/2012

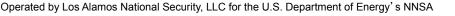
LA-UR-12-20511

Slides for: LA-UR-12-10472

Abstract:

We present a position paper that argues for the use of so-called "web" or "cloud" technologies as a platform for experimenting with asynchronous concurrent applications. We argue that it is unrealistic to ask developers to write their own code for load balancing, scheduling, fault-tolerance, etc. Instead we believe that exascale systems must provide a runtime system that provides these services. We advocate the use of cloud technologies as an experimental platform for designing requirements and APIs for these runtime systems.







Cloud+X: Exploring Asynchronous Concurrent Applications

Allen McPherson James Ahrens Christopher Mitchell

Los Alamos National Laboratory



UNCLASSIFIED LA-UR-12-20511

Slide 2

Operated by Los Alamos National Security, LLC for the U.S. Department of Energy's NNSA



Building Applications: Now and Near Future

- Today's apps are built with...
 - C, C++, FORTRAN
 - MPI
 - Job scheduler
- Tomorrow's apps built with...
 - C, C++, FORTRAN
 - <u>MPI+X</u>
 - X is CUDA, OpenMP, OpenCL, Cilk+, TBB, etc.
 - Job scheduler
- Application developer responsible for everything
 - Load balancing
 - Communication
 - Fault tolerance (checkpoint/restart)
 - Visualization, analysis, etc.



UNCLASSIFIED LA-UR-12-20511



Slide 3

Runtime/Middleware: The Missing Piece

- It's unreasonable to expect app developer to do everything
- System services should provide support for...
 - Dynamic load balancing (fork and kill processes on the fly)
 - Dynamic communication linkages...
 - Dynamic response to faults and power management
 - In situ visualization, analysis, tools...
- We believe a runtime or middleware system must provide this
 - Large effort to build
 - Vendors? Academia? Open community?
 - Before it's built...
 - What are the requirements
 - What are the appropriate APIs (allow innovation underneath)





Use Cloud+X vs. MPI+X

• We know how to do X

• Optimize for accelerator

We know how to do MPI

• And there is MPI-3 to help

But these do not cover all bases

- Need to experiment to define functionality and APIs
- Require an experimental platform that enables...
 - Rapid prototyping
 - Dynamic evolution of prototypes
- Require an experimental platform that provides...
 - Multitude of "service" functions that enable construction of dynamic apps
- We think "cloud" or "web" technology is experimental platform



UNCLASSIFIED LA-UR-12-20511

Slide 5

Cloud as Experimental Platform

We DO NOT advocate cloud as production platform

- Cloud components can be replaced with HPC technology
 - E.g. TCP messaging with MPI
- Cloud components can be optimized
 - If the functionality is there, often an engineering effort to make fast

Potentially useful cloud "services"

- Messaging (map communication and data translation to machine)
- Concurrent, asynchronous programming languages (rapid prototyping)
- Discovery and scheduling (map tasks to machine dynamically)
- NoSQL databases (complex data models, journaling, viz and analysis)

X is still X

- Integrate accelerated kernels as components
- Define interfaces between components and runtime system





Proposed Research Tasks

Gather potential experimental technologies

- Define functionality (e.g. service discovery)
- Select "best" candidates for prototyping

Evaluate other "HPC Cloud" efforts

- As pointers to useful technology
- As pointers to potentially show stopping issues

Develop open source proxies

• Dynamic, multi-scale, fault-tolerant, etc. <u>Vehicles for collaboration</u>.

Extract relevant abstractions

- Develop APIs for useful runtime
- Continuously iterate and refine

Identify issues that prevent use of cloud technology for production

• Evaluate potential remediation approaches and tradeoffs





Cloud+X: Very Early Efforts

Erlang for dynamic process creation/teardown

- Chris Mitchell @ LANL
- "Shock" through 2D square medium
- If shock intersects square, refine (ala quad-tree) and start 4 new processes
- Continue recursively until threshold (e.g. number of available nodes) reached
- Single node now, moving to cluster
 - Erlang process/message semantics same off-node as on-node

Magellan

- ASCR (NERSC) effort to evaluate cloud for science
- Huge report we need to read and evaluate carefully
- One conclusion (which we've seen in prior efforts)
 - Cloud communication mechanisms too slow (fixable with engineering \$\$?)
- They were looking to use cloud for production, we look to it for experimentation



UNCLASSIFIED LA-UR-12-20511

Slide 8

