Seeking Supernovae in the Clouds: A Performance Study

Keith R. Jackson, Lavanya Ramakrishnan, Karl J. Runge, Rollin C. Thomas

Lawrence Berkeley National Laboratory
Why Do I Care About Supernovae?

• The rate of expansion of the universe is accelerating
  – Propelled by mysterious new physics dubbed “Dark Energy”
  – This discovery was made by studying the brightness of Type 1a supernovae at varying distances
Type 1a Supernovae

- White dwarf stars that accrete gas from a companion star
- Explode at a critical mass equal to 1.4x the mass of the sun
- Standard amount of fuel creates a “Standard Candle” that can be used to measure their distance
Nearby Supernova Factory (Snfactory)

- Experiment to develop Type Ia supernovae as tools to measure the expansion history of the Universe and explore the nature of Dark Energy
  - Largest data volume supernova search from 2004-2008
  - Over 600 spectroscopically-confirmed supernovae
  - Observed both optically and with a custom designed spectrograph
- Collaboration between:
  - LBNL, Yale in the US
  - LPNHE, IPNL and CRAL IN2P3/CNRS labs in France
Data Pipeline

• Custom data analysis codes
  – Run on a standard Linux cluster
    • Jobs submitted to a standard batch queue
  – Controlled by a series of Python scripts
    • Used for coordination
• Complex set of algorithms
  – digital filtering, Fourier transforms, full matrix inversions, and nonlinear function optimization
• Heavily dependent on external packages
  – CFITSIO, the GNU Scientific Library (GSL), scipy, numpy, BLAS, LAPACK
  – Process level parallel
    • Large numbers of serial codes with different inputs
Why Interested in the Cloud?

- Long lived scientific project with most software development upfront
- Using shared clusters at NERSC and CCIN2P3
  - Changes to the clusters necessitate drafting scientists into debugging and rewriting codes
    - Change from 32 to 64 bit OS
- Amazon AWS Cloud provides:
  - Control over OS versions
  - Ability to install packages
  - Root access and ability to use shared “group” account
  - Immunity to externally enforced OS or architecture changes
Amazon AWS Setup

- SNfactory assumes a traditional HPC cluster environment
  - Not an option to re-architect for the Cloud
  - Emulate the cluster environment in the Cloud
- Used c1.medium instance types
  - 2 virtual cores, 2.5 EC2 Compute Units each
  - Most cost effective for Snfactory codes
Data Placement Options

- Elastic Block Store (EBS)
  - Provides a block level storage device that can be attached to an EC2 instance
  - Can be shared amongst instances via NFS
- Simple Storage Service (S3)
  - Highly scalable object store
  - Get/Put key values
  - Simple REST interface
Experimental Setup

- Examine the data storage options to maximize performance
- Use the Linux `sar` command to collect performance data
- One night handled by a worker VM
  - ~370 files
  - ~2.7 GB of input data
  - ~9 GB of output data
- 80 core virtual cluster

<table>
<thead>
<tr>
<th>Input Data</th>
<th>Output Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>EBS via NFS</td>
<td>Local storage to EBS via NFS</td>
</tr>
<tr>
<td>Staged to local storage from EBS</td>
<td>Local storage to EBS via NFS</td>
</tr>
<tr>
<td>EBS via NFS</td>
<td>EBS via NFS</td>
</tr>
<tr>
<td>Staged to local storage from EBS</td>
<td>EBS via NFS</td>
</tr>
<tr>
<td>EBS via NFS</td>
<td>Local storage to S3</td>
</tr>
<tr>
<td>Staged to local storage from S3</td>
<td>Local storage to S3</td>
</tr>
</tbody>
</table>
Experiment 1

- Input from EBS via NFS
- Output to local storage and then staged to EBS via NFS after the processing is complete
NFS Server
Worker Node

- net-send/recv
- net-rev MB/s
- net-send MB/s
- disk-read MB/s
- disk-write 10MB/s
- system load

Wallclock Time (h)
Experiment 2

• Input data read from EBS via NFS
• Output data written directly to EBS via NFS
  – Interleave the I/O with data processing
NFS Server
Worker Node

![Charts showing network send/recv, disk read/write, and system load over wallclock time.](image)
Experiment 4

• Input data read from EBS via NFS
• Output written to local storage and then to S3
NFS Server

![Graph showing network traffic and system load over time](image)
Worker Node
Cost of Analyzing One Night of Data

Experiment Matrix

<table>
<thead>
<tr>
<th>Instance Cost/night (80)</th>
<th>Data Cost/night (80)</th>
<th>Data Cost/night (40)</th>
<th>Instance Cost/night (40)</th>
</tr>
</thead>
<tbody>
<tr>
<td>EBS-A1</td>
<td>EBS-A2</td>
<td>EBS-B1</td>
<td>EBS-B2</td>
</tr>
<tr>
<td>S3-A</td>
<td>S3-B</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Costs of Full Experiment and 1 Month Data Storage

Experiment Matrix

<table>
<thead>
<tr>
<th>Experiment Matrix</th>
<th>Data Cost per month (80)</th>
<th>Instance Cost (80)</th>
<th>Data Cost per month (40)</th>
<th>Instance Cost (40)</th>
</tr>
</thead>
<tbody>
<tr>
<td>EBS-A1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EBS-A2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EBS-B1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EBS-B2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S3-A</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S3-B</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Failures

• Saw a significant aggregate rate of failure during our testing
  – Unable to allocate all 80 cores was the most common
    • Need to architect your system to adapt to the number of actual resources acquired
      – Unable to access the “user-data” passed in at instantiation time for customization
      – Failure to boot properly
      – Failure to configure the network
      – Poor performing nodes
Conclusions

- Porting your scientific application to the Amazon AWS Cloud requires significant work
- Must architect your application to handle failure
- Benchmarking your application is essential
  - Establish what instance type is most cost effective for your application
    - Cheaper per/node cost does not always translate to the cheapest overall cost
  - Understand the cost/performance tradeoffs for the various storage options
- Directly porting existing scientific applications does not best utilize the features available in the AWS Cloud
Acknowledgements

• This work was funded in part by the Advanced Scientific Computing Research (ASCR) in the DOE Office of Science under contract number DE-C02-05CH112

• Amazon for providing AWS access

• The Magellan team at NERSC