Integrating Remote Cloud and Local HPC Resources

Angel Pizarro
Institute for Translational Medicine and Therapeutics
Perelman School of Medicine
University of Pennsylvania
Acknowledgements

• Los Jefes:
  – Garret FitzGerald
  – John Hogenesch
  – Junhyong Kim
  – Jim Eberwine

• Mi Equipo:
  – Anand Srinivasan
  – Katharina Hayer
  – Mike DeLaurentis
  – Dimitra Sarantopoulou

• El Dinero:
  – UL1RR024134 (CTSA)
  – PA Department of Health
  – UPENN
The Problem

- Not enough compute
- No where to put computational infrastructure
- Attracting IT talent easier said than done
- No academic institution does this as well as mega-corps
Cloud to the rescue!

- “Magical land of endless compute!”™
- Amazon Web Services
  - UPENN strategic partnership
- Initial usage caps are easily lifted on request
  - Went from 40 to 300 in 2 days
- Then how to integrate?
Integration to Local Resources

• I have no immediate and easy answer for you

• My 2¢:
  – Start with separate resources
  – Provide a robust transport mechanism
  – Stabilize both resources
  – Closely monitor usage patterns of both
  – THEN AND ONLY THEN start thinking about tight integration
The Cloudy Choices Before Us

• Pay someone to provide a solution
• Managed multi-tenant environments
  – Hosted provider agreements
    • E.g. POD or other non-root accessible resources
  – Set-contract VPS & managed hosting
    • Assumes administrative rights on resources
    • Can be “bear metal” dedicated servers (RackSpace)
  – IaaS providers
    • AWS, RackSpace Cloud, etc.
• [Un]managed single-tenant environments
  – IaaS where “users” request and administer resources
Managed Multi-tenant Environments

- Recreating current HPC environments on AWS EC2
- Known management and execution tools
- EC2 is “just different enough” to make your life a huge pain
- Costs are no longer fixed and amortized
  - chargebacks are going to be different (and variable)
Single-tenant Managed Environments

- Bootstrapped Single Purpose Clusters (SPC™)
- Automation is critical
  - Permanent resources have a different management style, allow certain tradeoffs that are less palatable with cloud resources
- Able to tune SPC’s for each business process
  - Instance type, how many, execution engine, storage strategy, etc.
- Let’s look at an example: RNA-Seq analysis
Algorithm: RNA-Seq Unified Mapper (RUM)

Closer look at RUM workflow

- 10MM 100bp paired end simulated data
- Lots of IO
  - 75% writes
- Essentially a map-reduce workflow
- 30 “chunks”
Orchestration via StarCluster

- Python command line tool to configure and launch single-tenant clusters on AWS
<<< Configuring cluster took 5.672 mins
<<< Starting cluster took 6.576 mins
Our StarCluster Plugins

• Extend StarCluster’s bootstrapping procedure
• GridEngine Tweaks
  – Alter the number of slots on the master
  – Enable h_vmem on execution hosts
  – Enable exclusive reservation of hosts
• RAID0 Ephemeral storage
  – Formats all ephemeral disks into a single BTRFS volume
• GlusterFS on ephemeral storage
  – Parallel shared file system
  – Uses above to get massive single-namespace parallel filesystem
• PVFS2 on ephemeral storage
  – Alternate parallel shared file system
  – Built for high I/O workflows

https://github.com/PGFI/StarClusterPlugins
Experiment: Effect of a Shared File System on Run Time

- **NFS from the master host**
  - Native to StarCluster default strategy, on EBS

- **GlusterFS**
  - FUSE-based, slower than kernel modules
  - All nodes on cluster join their ephemeral storage as one distributed GlusterFS volume

- **PVFS2**
  - Kernel module shunts requests via a pvfs2-client daemon
  - Distributed striped volumes across ephemeral storage
PVFS2 Results

• Killed PVFS2 after 232 minutes
  – Master process looked for files and aggressively cleaned up after itself
  – Restarted each “chunk” analysis
  – Your algorithm may work better
  – There are tuning parameters that allow more file system consistency
  – Can tune the data server and metadata server layout configuration
PVFS2 Profile Data
(something is not right)
GlusterFS Results

• Completed in 112 minutes
  – 10MM paired end RNASeq data ~ $7.50 to align
  – VERY CLEAN SIMULATED DATA

• We’ve tried using 2 dedicated GlusterFS file servers to service the cluster, and that failed badly.
  – NFS time outs, lots of EBS => $$$

• Much better performance to use it as scratch space on ephemeral drives
  – Also cheap, since it utilizes ephemeral drives
GlusterFS Profile Data
NFS from the master node

• Finished in 91 minutes
• Not much faster than GlusterFS

• We have seen it fail hard under heavy loads
  – 10-15 servers, 90 processes
• Unless you RAID, limited to 1TB volumes
• Probably best to use local scratch space on nodes, copy back final results to NFS space
NFS Profile Data
Conclusions

• Profile your algorithms, tune Single Purpose Clusters (SPC™) for CPU and IO

• **COMPLETELY** automate the process of bootstrapping SPC’s
  – Automate the process of bringing them up and bringing them down

• Once you have achieved “set it and forget it” status, treat it as a single algorithm/service that gets integrated with local resources