Building a Virtualized Desktop Grid

Eric Sedore
essedore@syr.edu
Why create a desktop grid?

- One prong of an three pronged strategy to enhance research infrastructure on campus (physical hosting, HTC grid, private research cloud)
- Create a common, no cost (to them), resource pool for research community - especially beneficial for researchers with limited access to compute resources
- Attract faculty/researchers
- Leverage an existing resource
- Use as a seed to work toward critical mass in the research community
Goals

- Create Condor pool sizeable enough for “significant” computational work (initial success = 2000 concurrent cores)
- Create and deploy grid infrastructure rapidly (6 months)
- Secure and low impact enough to run on any machine on campus
- Create a adaptive research environment (virtualization)
- Simple for distributed desktop administrators to add computers to grid
  - Automated methods for detecting/enabling Intel-VT (for hypervisor)
  - Automated hypervisor deployment
Integration of Existing Components

- Condor
- VirtualBox
- Windows 7 (64 bit)
- TCL / FreeWrap – Condor VM Catapult (glue)
- AD – Group Policy Preference
Typical Challenges introducing the Grid (FUD)

- **Security**
  - You want to use “my” computer?
  - Where does my research data go?

- **Technical**
  - Hypervisor / VM Management
  - Scalability
  - After you put “the grid” on my computer…

- **Governance**
  - Who gets access to “my” resources?
  - How does the scheduling work?
Security
Security on the client

- Grid processes run as a non-privileged user
- Virtualization to abstract research environment / interaction
- VM’s on the local drive are encrypted at all times – (using certificate of non-privileged user)
  - Local cached repository and when running in a slot
  - Utilize Windows 7 encrypted file system
  - Allows grid work on machines with end users as local administrators
- To-do – create a signature to ensure researcher (and admins) that the VM started is “approved” and has not been modified (i.e. not modified to be a botnet)
Securing/Protecting the Infrastructure

- Create an isolated private 10.x.x.x. network via VPN tunnels (pfSense and OpenVPN)
- Limit bandwidth for each research VM to protect against a network DOS
- Research VM’s NAT’d on desktops
- Other standard protections – Firewalls, ACL’s
Public Network

OpenVPN End-Point (pfSense) / FW / Router

Research VM's

ITS-SL6-LSCSOFT

10.x.x.x network

Bottleneck for higher bandwidth jobs

Condor Infrastructure Roles

Condor Submit Server
Condor VM Coordinator (CMVC)

- Condor’s VM “agent” on the desktop
- Manage distribution of local virtual machine repository
- Manage encryption of virtual machines
- Runs as non-privileged user – reduces adoption barriers
- Pseudo Scheduler
  - Rudimentary logic for when to allow grid activity
  - Windows specific – is there a user logged in?
Why did you write CVMC?

- Runs as non-privileged user (and needs windows profile)
- Mistrust in a 3rd party agent (condor client) on all campus desktops – especially when turned over to the research community – even with the strong sandbox controls in condor
- Utilizes built-in MS Task Scheduler for idle detection – no processes running in user’s context for activity detection
- VM repository management
- Encryption
- It seemed so simple when I started…
Job Configuration

- Requirements = (TARGET.vm_name == "its-u11-boinc-20120415") && (TARGET.Arch == "X86_64") && (TARGET.OpSys == "LINUX") && (TARGET.Disk >= DiskUsage) && ((TARGET.Memory * 1024) >= ImageSize) && ((RequestMemory * 1024) >= ImageSize) && (TARGET.HasFileTransfer)

- ClassAd addition
  - vm_name = "its-u11-boinc-20120415"

- CVMC Uses vm_name ClassAd to determine which VM to launch

- Jobs without vm_name can use running VM’s (assuming the requirements match) – but they won’t startup new VM’s
Technical Challenges

- Host resource starvation
  - Leave memory for the host OS
  - Memory controls on jobs (within Condor)
- Unique combination of approaches implementing Condor
  - CVMC / Web service
  - VM distribution
  - Build custom VM’s based on job needs vs. scavenging existing operating system configurations
- Hypervisor expects to have an interactive session environment (windows profile)
- Reinventing the wheel on occasion
How do you “ensure” low impact?

- When no one is logged in CVMC will allow grid load regardless of the time
- When a user is logged in CVMC will kill grid load at 7 AM and not allow it to run again until 5 PM (regardless if the machine is idle)
- Leave the OS memory (512MB-1GB) so it does not page out key OS components (using a simple memory allocation method)
- Do not cache VM disks – will keep OS from filling its memory cache with VM I/O traffic
Keep OS from Caching VM I/O
# Syracuse University Condor Pool Machine Statistics for Week

![Graph](image)

**Graph Note:** The Y-axis is number of machines, the X-axis is time. When graph finishes updating, press “Configure” to view different architectures or time data. Also, you can use the mouse to draw a rectangle on the graph and then press “Zoom In”. Press “Reset” to reset view the data after “Configure” or when done zooming. Nighttime shows up as graph background in gray.

<table>
<thead>
<tr>
<th>Arch</th>
<th>Owner Average</th>
<th>Condor Average</th>
<th>Idle Average</th>
<th>Backfill Average</th>
<th>Drained Average</th>
<th>Owner Peak</th>
<th>Condor Peak</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Total</strong></td>
<td>0.0 (0.0%)</td>
<td>2811.9 (98.2%)</td>
<td>44.2 (1.8%)</td>
<td>0.0 (0.0%)</td>
<td>0.0 (0.0%)</td>
<td>0 (0%)</td>
<td>4038 (100%)</td>
</tr>
<tr>
<td><strong>X86_64/LINUX</strong></td>
<td>0.0 (0.0%)</td>
<td>2811.9 (98.2%)</td>
<td>44.2 (1.8%)</td>
<td>0.0 (0.0%)</td>
<td>0.0 (0.0%)</td>
<td>0 (0%)</td>
<td>4038 (100%)</td>
</tr>
</tbody>
</table>
Next Steps

- Grow the research community – depth and diversity
- Increase pool size – ~12,000 cores which are eligible
- Infrastructure Scalability
  - Condor (tuning/sizing)
  - Network / Storage (NFS – Parrot / Chirp)
Solving the Data Transfer Problem

- Born from an unfinished side-project 7+ years ago.
- Goal: maximize the compute resources available to LIGO’s search for gravitational waves
  - More cycles == a better search.
- Problem: huge input data, impractical to move w/job.
- How to...
  - Run on other LIGO Data Grid sites without a shared filesystem?
  - Run on clusters outside the LIGO Data Grid lacking LIGO data?

*Tools to get the job done: ihope, GLUE, Pegasus, Condor Checkpointing, and Condor-C.*

*People: Kayleigh Bohémier, Duncan Brown, Peter Couvares. Help from SU ITS, Pegasus Team, Condor Team*
Idea: Cross-Pool Checkpoint Migration

- Condor_compiled (checkpointable) jobs.
- Jobs start on a LIGO pool with local data.
- Jobs read in data and pre-process.
- Jobs call checkpoint_and_exit().
- Pegasus workflow treats checkpoint image as output, and provides it as “input” to a second Condor-C job.
- Condor-C job transfers and executes standalone checkpoint on remote pool, and transfers results back.
Devil in the Details

- Condor `checkpoint_and_exit()` caused the job to exit with SIGUSR2, so we needed to catch that and treat it as success.
- Standalone checkpoint images didn’t like to restart in a different cwd, even if they shouldn’t care, so we had to binary edit each checkpoint image to replace the hard-coded `/path/to/cwd` with `.////////////`
  - Will be fixed in Condor 7.8?
- Pegasus needed minor mods to support Condor-C “grid” jobs w/Condor file transfer
  - Fixed for next Pegasus release.
Move jobs that do not require input files on the SUGAR cluster to the remote campus cluster.