Networking and Information Technology
Research and Development

The government seeks individual input; attendees/participants may provide individual advice only.

Middleware and Grid Interagency Coordination (MAGIC) Meeting Minutes
April 4, 2018, 12-2 pm
NCO, 490 L'Enfant Plaza, Ste. 811
Washington, D.C. 20024

Participants (*In-Person Participants)

Bryan Biegel	NCO	Ron Payne	UIUC
Brian Bockelman	UNL	Jared Punzel	ANL
Richard Carlson	DOE/SC	Don Riley	UMD
Michael Crusoe	CWL/Octiog	Stefan Robila	NSF
Kaushik De	UTA	Gonzalo Rodrigo	LBNL
Yaniv Donenfeld	AWS	Sonia Sachs	DOE/SC
Dave Godlove	Sylabs	Alan Sill	TTU
Chris Hoge	OpenStack	Derek Simmel	PSC
Derek Jensen	ANL	Adam Simpson	ORNL
Padma Krishnaswamy	FCC	Adam Soroka	Smithsonian
Joyce Lee	NCO	Kevin Thompson	NSF
Miron Livny	UW-Madison	Ralph Wachter	NSF
Paul Love	NCO	Amy Walton	NSF
JP Navarro	ANL	Jack Wells	ORNL
Sanjay Padhi	AWS	Frank Wurthwein	UCSD/OSG
Philip Papadopoulos	Comet	Wei Yang	SLAC/Stanford
Valerio Pascucci	UOU		

Proceedings
This meeting was chaired by Rich Carlson (DOE/SC).

Speaker Series: Resource providers' server and system preparation for containers

- **Comet:** Philip Papadopoulos, Chief Technology Officer, San Diego Supercomputer Center (SDSC)
- **Open Science Grid:** Frank Wurthwein, Director, Open Science Grid (OSG); Professor of Physics (UCSD); HTC Lead (SDSC)
- **CMS/Large Hadron Collider:** Brian Bockelman, Research Assistant Professor, Computer Science and Engineering Department, University of Nebraska-Lincoln
- **OpenStack:** Chris Hoge, Senior Strategic Program Manager, OpenStack Foundation
- **Leadership Computing Facilities:**
  - Adam Simpson, HPC Support Specialist and Programmer, Oak Ridge Leadership Computing Facility (OLCF)
  - Derek Jenson, HPC Systems Administrator, Argonne Leadership Computing Facility (ALCF)
  - Jared Punzel, HPC Systems Administrator, ALCF
- **Amazon Web Services:** Yaniv Donenfeld, Business Development Manager, Container Services, Amazon Web Services

Speaker Presentations

***Comet Virtual Clusters and Containers* - Philip Papadopoulos**

Comet: NSF funded XSEDE resource (Gateways to Discovery: CyberInfrastructure for the Long Tail of Science)

Overall Network Architecture & Perspective (Slide 3-4)

- Infiniband to Ethernet bridging (Infiniband 2.8 TB bridging to large 1040 x 100GB Ethernet fabric)
- Complex software stacks, but virtualization gives some responsibility and control to users
- Mission: Run usable computation and data-centric facility for national scientific-computing user base; not intended to be a generic cloud provider.

Virtualization: First XSEDE resource to propose and support high performance virtual clusters. (Slides 5 - 6)

- Not target end users, rather users who can handle clusters (system admin, etc) supporting users.
- Administrators can use their administrative tools without learning a new cloud software stack.
- Single Root I/O Virtualization (SR-IOV): gives latency (within 90-95% of raw hardware latency) and 100% bandwidth efficiency compared to raw hardware. Impacts some applications but not many.
- Supports Singularity-based containers, which replaces their executable.

User Perspective of Virtual Cluster (Slides 7 – 8)

- Power on/off of virtual nodes are scheduled in batch system.
- Retain disk state in virtual cluster nodes between boots.
- Overlay policy: Virtual compute nodes completely overlay Comet cluster nodes. See 2 virtual clusters with private networks completely isolated from each other, so compute nodes can PXE-boot over those networks. Owners can start with ISO image and install from ground up.
- Bare Metal "experience": Cluster owners not have to learn how to use virtual hardware
  - Can install from bootable ISO image; subordinate nodes can PXE boot and compute nodes retain disk state. Allows them to bring their entire tool chain with them.

Virtual Cluster Projects & Containers (Slide 9):

- Projects
  - Open Science Grid (OSG): extend UCSD Physics Department Tier 2 cluster with Comet
  - Nautilus Hypercluster: runs Linux Kernel 4.0. Can run preferred kernel; running containers inside their own virtual cluster.
  - Lake Ecology/Freshwater quality modeling – uses overlay network and iPOP (connect into larger scale HTCondor pool)
- Singularity – allows folks to run their applications in standard queuing system

Other Container Systems/Orchestration? (Slide 10)

- Docker Native: moving to Singularity because of need to be "sudo" and uncomfortable with ease of access to open public container repositories
- Shifter/Singularity: better usability and fit within HPC space

***Containers on OSG* - Frank Wurthwein**

Background (Slides 2-3)

- OSG empowers researchers to use compute and data resources across institutional boundaries nationally and internationally. Integration is key to success, as science is a team sport.
- OSG provides global integration across commercial and academic computing but respects local ownership and control (hardware owner decides what is done with it)

- Business model: provides knowledge and software infrastructure to empower scientists to work with their home institutions for long term sustainability

Integrating Compute and Storage Clusters (Slide 4)

- Approximately 200,000 Intel x86 cores used by approximately 400 projects across 36 fields of science (about 80% of this scale is running containers).

OSG Computing Integration (Slide 5)

- Supports allocations on National Supercomputer (e.g., Comet)
- Supports sharing on: collaborator's cluster, national shared clusters and purchasing on the commercial cloud. User has access point and can define what resources to use and access.

Caching Service (Slide 6)

Started last year; Currently 500 TB/week delivered to wide range of customers (in addition to customers stored resources)

Container Background (Slide 7)

Distributed High Throughput Computing is about making applications portable (run anywhere)
Containers: latest technology allowing people to port and move applications; eliminate high heterogeneity

OSG Overlay Batch System (Slide 8)

OSG: comprised of 60+ institutions allowing access to their cluster. As each has a different policy, it would be impractical to submit user jobs (queued in 60 different places).

- Instead, OSG uses the batch system: virtualizes all resources into 1 global batch system and deploys from there. The batch system OSG submits may run in a container and the application submitted to a batch system may again run in a container. At user's option to use the container.
- OSG produces the appearance of homogeneity for the users in a heterogeneous environment.
- Users should not be system administrators.

Containers on OSG (Slide 9 -13)

- Provides Singularity containers for user to build upon; users derive singularity containers from Docker containers. Gives users a starting point that works.
- GPUs on the grid: 4 out of 60 institutions provide GPU access via OSG (Slide 11-12)
  - Introduces heterogeneity explosion to science community, so created singularity images
  - Number of GPUs per slot set by the site; currently not much support for multi-GPU slot
- Complexity: try to hide from scientists. Knowing Condor Solutions is sufficient.

***Moving CMS to a Container-based Infrastructure* - Brian Bockelman**

Scope: How we use container in terms of side software and batch systems and address 1 CMS experiment. (Slide 3-4)

Background

- CMS is general purpose detector in LHC. Complex physics problem becomes complex computing problem. Need HPC hardware near detector to quickly filter out >99.99% of data resulting from uninteresting events. Remaining -0.001% of events still require hundreds of millions of computing hours to reconstruct. Result: multi-billion-CPU/hour annually.
- Traditional solution: distributed computing because often cannot find all resources at a single site to address entire problem. Express as high throughput computing problem (number of events able to process per year).

- OSG: resource used by LHC community; provides common platform

2 Challenges: Portable Environment (slide 5)

1) CMS experiment has complex software base. Made 2 attempts at making portable:
- Install software manually, but impossible to do integrity checking
- CERN VM File System: custom file system that limits scope to support HEP community

2) Overlay system limited regarding pilot job. But not much separation from payload
- Singularity Solution works well (Slides 10-14)
- Multiple layers: Container layer can be outside or inside pilot. Attempted to isolate payloads, but some sites do both. Process tree starts at site batch system level to user job.

Different Views:
- System administrator (Slide 12),
- Within pilot, some batch systems do not see the batch system from pilot (Slide 13).
- From user job, no visibility into host batch system or where pilot doing (Slide 14)

Singularity and CMS: Start running workflow in 2017; 90% adoption by end of March 2018.

Pitfalls (Slide 18)
- Shared filesystems tricky – deciding what shared filesystem gets exposed into the container
  - Now all payloads have same UID, so potentially more work on separation techniques on file systems
- Linux kernels have bugs, so had to do workarounds
- Spawn a new container? Has container launched appropriately?

Lessons Learned (Slide 19)
- Fast transition
- Must adopt systems to new things
- Run in privilege mode: security bugs in Singularity;
- In unprivileged mode: subject to kernel security bugs

Looking Forward (Slide 20)
- Portability: how to move software stack into container and manage large containers or many per site
- 5-10 years out, community will push bounds of how much data we can take. As community is in the process of "bending resource curve" into our budgets, there will be an opportunity to diversify location for running computing workflows. Hopefully, containers will remain part of the solution for more compute resources, sites, and types of providers.

***OpenStack and Container Integrations: From Bare Metal to Applications with OpenStack and open source technologies* - Chris Hoge** (See Presentation for links to projects and resources)

OpenStack (Slide 3):
- Provides infrastructure as an abstraction. Delivers all the major components of compute networking and storage to users through a common API.
- Serves 2 main groups: 1) operator looking to simplify internal operations and 2) end user wishing to consume heterogenous resources through an abstract API that is consistent from region to region and cloud to cloud.

OpenStack Ironic: provisions bare metal machines; stand alone or integrated with OpenStack projects. Research cloud examples (Slides 4-6):
- Chameleon Cloud: infrastructure project implementing an experiment testbed for computer science. Comprises about 600 nodes. Ironic is meeting their needs.

- CERN: provides computing resources for LHC and other experiments (more than 160k cores). Currently deploying Ironic into production for bare metal management of machines. It is adding features to Ironic that help manage the machines' complete life cycle.

Container-Hosted OpenStack: Infrastructure containers have privileged system access and provide infrastructure support and management. Project examples (Slides 7-9):

- OpenStack Ansible (LXC with Ansible): OpenStack services deployed to LXC containers, primarily developed at Rackspace to support their private clouds.
- Kolla Ansible: Kolla image-based with sophisticated build system allowing elaborate configuration. Ongoing project with support for upgrades, cluster expansions and node maintenance, repair and removal.
- OpenStack Helm: Powerful and flexible new AT&T project featuring Kubernetes and Helm as the infrastructure layer. Leverages Kubernetes, which can take abstractions regarding applications management and apply them generically across both infrastructure and application layers.
    - OpenStack Loci (written to support OS Helm): Small containers without the extra applications not needed to deliver OpenStack services.

OpenStack Hosted Containers: Hosting container application frameworks on OpenStack (Slides 11-15)

- Zun: typically launches Docker containers. Working on support for other frameworks.
    - Backed by OpenStack Kuryr: Acts as plug in for Docker networking and replaces other network models available to Docker. If running neutron network overlay, can directly connect tenant separate and secure networks to the containers.
- OpenStack Magnum: fully hosted Kubernetes (or Docker Swarm). Enables a high level of security within Kubernetes cluster by providing tenant-isolation through networks and virtual machines, and machine isolation. Working on true tenant isolation through isolated machine reservations because of potential cache timing attacks in untrusted code within a cluster (leading to the discovery of secrets on shared hardware).
- Kubernetes Cloud Provider OpenStack: Collaboration between Kubernetes, OpenStack, and cloud providers (e.g., Azure, AWS and Google Compute) to ensure same interface and consistent documentation to facilitate moving workloads between cloud platforms. Hosted within the Kubernetes repository to provide support for Kubernetes running on top of OpenStack. Working on support for cluster autoscaling.
- Kata Containers: New (unofficial) OpenStack Foundation project with open source collaboration. Have the additional security of running containers as individual machines with the security guarantees of virtualization without giving up container run time APIs. Because they are running in a virtual machine, they are unsuitable for running infrastructure loads but good for delivering application workloads because guarantees certain isolation. Microsoft and Google are exploring Kata containers to deliver secure applications to containers within their cloud frameworks.

***Singularity: DOE LCF HPC container usage* - Adam Simpson, Derek Jenson, Jared Punzel**

OLCF & ANL Perspectives on Containers:

- OLCF- View containers as empowering users to better control the software stack. Many systems in production and development at ORNL cover a wide range of CPU architecture and software stacks; ORNL would like to run containers on them.
- ANL: Portability is top reason for container use.

Singularity (OLCF): Provides uniform interface for building and running containers (Slide 2)

- Ease of use: works well with old kernels, not just newest kernels.
    - ANL: no kernel issues because running Singularity in a heterogeneous environment

- Singularity file works well with existing infrastructure. Can share a container among members of project using standard file techniques
- Interoperates well with Docker: can convert docker image to Singularity format
- Security: no root-owned daemon processes required. Can't allow root access in HPC environment.
- Portability with no performance degradation.
- ANL: Lighter administrative load for Singularity compared to Shifter. Easier to deploy single binary.

Container Builder (OLCF) (Slide 3)

- Need root access to build container from recipe file.
- Deployed Container builder: allows users to build containers from production compute resources.

Building Containers: System-Specific Considerations (Slide 4)

- OLCF: Provides base container images with system-specific implementations (MPI, CUDA).
- ANL: Cray-specific information baked into base container for users to import and build from.

Singularity HPC Jobs (Slide 6)
Scheduler agnostic. Very easy to integrate Singularity into users' batch job. Singularity exec replaces regular binary; easy to use with MPI commands as well.

Singularity Registry (Slides 7-8)

- Provides web-based central repository for Singularity images.
- ANL: Created own registry of images to users that are guaranteed to work on its systems. Easy to use with the "pull" subcommand.

***AWS Container Services* - Yaniv Donenfeld**
Background:

- Created Container services, Amazon ECS. Built to be reliable, scalable and decoupled in its operation/ distributed system, with no single point of failure. Customers are using containers at scale for a broad range of services (batch jobs, Machine Learning algorithms, etc); about 70,000 jobs/hour
- Significant Platform Growth: Weekly, hundreds of millions of containers on ECS, across millions of container instances; also used internally.
- Compliance: ensured platform complies with security standards (HIPAA, International Organization for standardization, etc.). Shared responsibility model.
- Service Level Agreement added for service availability: if choosing to run applications on Docker containers through services like ECS.

Goal: AWS as the best place to run containers regardless of the application. Provide all capabilities at the container, or task/quad level, that are available on traditional server scale.

- E.g., ECS can give granular access permissions for a given workload. Can decide level of permissions of your application accessing other services on AWS directly through containers.
- Provide means to autoscale applications based on containers.
- Load balancers (e.g., can forward traffic directly to containers)

Recently launched:

- Ability to integrate Amazon network stack (VPC) into users' running container. If forming a network between containers, can use standard IP addresses and security groups at a task level.
- Managed Service Discovery for ECS: Can register ECS services with their user-friendly name and AWS will manage it.

AWS Fargate: new technology allowing users to have server-less way of running docker containers

- Fargate provides customers with the means to abstract the infrastructure layer so they only manage and scale their applications.
- For typical clusters of resources made of few EC2 instances/VMs running several tasks/quads, users do not have to bring in actual servers into that cluster to run applications. They can stay at that abstraction, define, run and scale the services and applications based on their configuration.
- Configuration and pricing: based on CPU configuration and time running on system

Kubernetes

- Background: Working with CNCF to ensure addressing customer needs for AWS to run the control plane and other AWS services. Currently, 63% of community workflows worldwide are running on AWS. Customers prefer keeping the Open Source communities upstream.
- To facilitate running workflow on AWS, started to build Amazon EKS (Elastic container service for Kubernetes). General availability: later in CY2018.
- Provides controllable, highly available plane abstracted from the user. Work with same tooling and ecosystem that work with in Kubernetes; but AWS manages and scales it. User simply brings in worker nodes that are running the actual quads in your system.

Calico Project: network policy and overlay solution for Kubernetes

- Used to configure quad level access on your running Kubernetes clusters on EKS. Calico network policy APIs: If want to decide what ports or level of granular access your application provides.
- Plan to support every new version of Kubernetes. Currently running on 1.9. Will provide mechanisms to upgrade and self-patch every new version launched. AWS will facilitate upgrading worker node side and bring Cubelets to most recent and updated version.
- 2 parts: 1) Control plane (AWS manages and scales; upgrades automatically) and 2) Nodes running quads (user currently manages and upgrade using standard rolling upgrades). Going forward, AWS will provide a means to automate as part of cluster upgrade.
- Plan to support Fargate on EKS in the future. Customers will want to only address API level on Kubernetes.

Discussion

- OLCF Container builder availability: Currently, the container builder is not available to the community as it is specific to the center, but OLCF is looking into opening it up.
- NCAR containerized applications: Users are requesting NCAR Containerized applications which are difficult to build from scratch. There is nothing registered in the Singularity hub for atmospheric applications.
    - Many OLCF users are interested in packages like Tensorflow and making modifications to it. OLCF is maintaining a GitHub page for recipes files for popular packages. Users take it to a container builder to obtain or to modify/build a container. Useful to provide recipes for OLCF's user base. Repository contains based container definitions and Deep Learning packages (Tensorflow and PyTorch). https://github.com/OLCF /container-recipes
- Availability of Amazon Fargate specifications? Implemented as part of ECS service, but planning to implement EKS Fargate using community standards, but in design phase. Yaniv Donenfeld will connect you with AWS's product team if you have any questions.
- Open Container Initiative (OCI) standard and commitment from Singularity to implement the OCI image, registry and runtime specification. Thoughts?
    - Perhaps go to www.cncf.io, one of the most active community driven specification efforts.
    - OpenStack's KATA containers program is an OCI member and will adhere to the specification. Chris Hoge (chris@openstack.org)

Next MAGIC meeting: May 2, 2018 at the National Coordination Office, 490 L'Enfant Plaza, Suite 8001