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

Middleware and Grid Interagency Coordination (MAGIC) Meeting Minutes
March 7, 2018, 12-2 pm
NCO, 490 L'Enfant Plaza, Ste. 811
Washington, D.C. 20024

Participants (*In-Person Participants)

Gonzalo Alvarez	LBL	Brian Lyles	ORNL
Ashely Barker	ORNL	David Martin	ANL
Bryan Biegel	NCO	Gilberto Pastorello	LBL
Laura Biven	DOE/SC	Sergey Panitkin	BNL
Ben Brown	DOE/SC	Dula Parkinson	LBL
Rich Carlson	DOE/SC	Rajiv Ramnath	NSF
Vipul Chaudhary	NSF	Don Riley	UMD
Luca Cinquini	NASA	Sonia Sachs	DOE/SC
James Clark	LIGO	Scott Sellars	NSF
Peter Couvares	LIGO	Matyas Selmici	UW-Madison
Michael Crusoe	CWL/Octiog	Alan Sill	TTU
Dave Godlove	Sylabs	Adam Soroka	Smithsonian
Dan Gunter	LBNL	Panagiotis Spentzouris	FNAL
Jason Kincl	ORNL	Kevin Thompson	NSF
Joyce Lee*	NCO	Matteo Turilli	Rutgers
Miron Livny	UW-Madison	Wei Yang	Stanford/SLAC

Proceedings
This meeting was chaired by Rich Carlson (DOE/SC) and Rajiv Ramnath (NSF).

Speaker Series: Containerization & Virtualization Technologies: scientific community's adoption and/or use of the containerization and virtualization technologies.

- **ALS Science**: Dula Parkinson, Research Scientist, Experimental Systems Group, Lawrence Berkeley National Laboratory
- **ESGF and DREAM**: Luca Cinquini, Data Scientist, National Aeronautics and Space Administration Jet Propulsion Laboratory
- **HEPCloud**: Panagiotis Spentzouris, Head, Scientific Computing Division, Office of the CIO, Fermi National Accelerator Laboratory
- **KBase**: Michael R. Crusoe, Lead and co-founder, Common Workflow Language, a project of the Software Freedom Conservancy and CEO, Octiog, Inc.
- **US ATLAS**: Wei Yang, Ph.D. Information Systems Specialist, Stanford University/SLAC National Accelerator Laboratory
- **LIGO:** James Clark, Laser Interferometer Gravitational-Wave Observatory (LIGO)Research Scientist at Georgia Institute of Technology

***Synchrotrons and Supercomputers* - Dula Parkinson**
Synchrotron**:** Biggest light sources in U.S. (Slides 2-3):

- Advanced Light Source (ALS),
- Stanford Synchrotron Radiation Lightsource (SSRL)
- National Synchrotron Light Source (NSLS)
- Advanced Photon Source (APS)
- NSLS-II

Beamlines: Multiple beamlines for each light source; ALS has 40 beamlines (Slide 4)

User Statistics (Slides 5 - 7)

- Light sources are user facilities; 50-100 users on site, 1 hour to 10 days
- Variety of disciplines and experiments (e.g., mouse imaging to fuel cells, historical artifacts)
- New User: Daily, more than 20% are new
- Beamline (Slide 8): would like to have camera connected to computer, but insufficient for computing, storage and processing. Move computers to facility-level and other computing centers

Thoughts about container use (Slide 8)

- Increasing need for more and different computer capabilities
- Uptime is critical – need feedback during experiment to conduct it correctly.
  - Access to database/portal so users can make decisions about experiments
  - need different resources at different times to match experiment needs
- Build once, deploy (Slides 9 -10)
  - Different computers running different instruments; want to hook them into same system. E.g., Wish to deploy data ingestion system at many instruments
  - For analysis and visualization - so users can take containers and easily run analysis or visualization wherever they have access.
  - Prototype superfacility concepts: ALS, Spot suite built with LBL, partnered with ESnet for networking; National Energy Research Scientific Computing Center (NERSC) for HPC and storage; Camera for algorithm development; multi-year effort
- Collecting data at ALS beamline and transferred to NERSC for pre-processing and some to TITAN for simulations (run to submit data, brought back to NERSC and presented through web portal)
- NERSC Shifter and Spin: working on running experiments (Slide 12)

Xi-CAM Analysis Interface (Slide 11): Allows graphical interface for interactive data processing.

- Common to collect data set and have interactive feedback (adjust parameters and see its impact on images before launching a big job)
- Exploring remote data and remote execution

***The Magic of Dreams* (*Earth System Grid Federation (ESGF) and Distributed Resources for ESGF Advanced Managemen*t (*DREAM*)) - Luca Cinquini**

ESGF Overview

- International collaboration of climate centers building global infrastructure for managing, sharing and providing access to climate data. Includes about 30 data centers worldwide. R&D 100 Award.
- Looking at containers to address next infrastructure
- Hosts one of the most important data collections in the world for climate change research
  - Coupled Model Intercomparison Project (CMIP): Most significant project
  - Compiles assessment reports on climate change for world governments. Preparing to serve the next generation, CMIP3, (25-40 PB data needing replication at different centers)

Why Containers?

- Complex software stack: composition of many applications reduced by normal Open Source communities (Tomcat, Solr) and specific applications for climate field. Can be deployed on single host/server or on data node (just serve data, no other functions).
- Installation driven by gigantic shell script creates problems with understanding, upgrading, etc.

DREAM Project (Distributed Resources for the ESGF Advanced Management)

- Goal: build next generation ESGF, new infrastructure and new architecture that is more modular, scalable, easy to install, and applicable to other domains (climate, hydrology, biology, etc.).
- Strategy: Modularize ESGF component and have each talk to each other through RESTful APIs.
- Implementation: Containerization brings more tools to achieve this goal. Use full stack of publicly available containerization technologies to design new, micro-services based architecture
  - Docker, Docker Compose, Docker Swarm, Docker Stack (containerization technologies slide)
  - Will support Kubernetes - becoming the main orchestration engine in Container community
    - Helm: to configure, package, and deploy Kubernetes resources (Kubernetes add-on)
    - OpenShift Origin: to provide higher level platform for managing Kubernetes clusters
  - Prototype architecture: ESGF/Docker Architecture with Docker Swarm; each service has become a Docker container. (See Slide of containers running on 5 node cluster on AWS)
    - Docker Swarm is flexible - e.g., if provision cluster of 5 different machines, can assign service to machine based on metadata labels (1 can be data node, etc)
    - Advantages (see slide): solves current problems regarding installation, etc

Lessons Learned

Can be difficult to move legacy application to container architecture

- Data storage – In container architecture, containers and pods are moved from host to host
  - Need to code into the configuration to ensure that the same data is always read
- IP addresses and host names – In container architecture, IPs change, so need to use Docker, Kubernetes, to configure end-point for containers to reference each other
- Services/APIs may not be stateless – problem when use data from a previous request

Steep learning curve for application developers and system administers

- Difficult to run multiple Docker containers and to run applications as pods in Kubernetes cluster

Best Practices & Conclusion (see last slides)

- If using, completely commit to containers so everyone generates certificates the same way
- Release complete ESGF/Docker architecture by end of 2018
- Run operational sites with ESGF/Docker architecture (On Premise or Cloud) by next year

***Container Technology for HEPCloud* - Panagiotis Spentzouris with Burt Holzman, Anthony Tiradani**

HEP user community (Slide 2)

- National and international distributed collaborations (supporting around 15 experiments and projects: largest experiment (CMS) at CERN; largest project (DUNE) – 1000 users)
- HEP has "Standard" software stack; but different versions and usually limited resources/expertise

HEPCloud: concept for elastically expanding the Fermilab facility

- Experiments need massive computing resources (e.g., CMS use about 150k cores) (Slide 3)

- Resource utilization pattern of peaks and valleys
- Need elastic provision of resources (e.g., if potential for discovery exists)
- Agnostic - could be used by other SC programs with similar needs

HEPCloud Architecture – (Slide 4) High view of architecture: Decision engine is most important element

Current HEPCloud concept of deployment (Slide 5) - In project phase; move to operations at end of CY2018

- HP condor for workload management; variety of specific software layers and packages for monitoring, interfaces
- Use cases run on different resources: AWS, Google, NERSC (Slide 6)

Containers in HEPCloud (Slide 7)

- Needed to decouple user environment facility resource environment
- Current deployment driven by security requirements of different resource providers and ability to stand up on different resources; and HEPCloud support of services to provide container technologies and capabilities to users
- Current and near-term future compute resources targets identified in slide. (e.g., LCF, but issues with access models and containers)

Current implementation (Slides 8 -11)

- FNAL"local" resources: Not ideal, but provide short menu of options for Docker containers, consulting for Singularity
  - Diagram: stack of facility services; Worker Node includes CVMFS service (distributed, read only file system that can leverage to distribute specific libraries and software packages needed for user jobs (Slide 8)
- Commercial clouds (Slide 9)
  - Diagram: HEPCloud will build Facility services; CVMFS (part of worker node)
  - HEPCloud is working with AWS
  - Goal: operations at end of CY2018; current solutions are temporary

- NERSC (Slide 10)
  - Send Docker image to NERSC, converted to Shifter; no CVMSF (facility policy)
  - Considering expanding scope of edge services/Squid services, (may help with software distribution and with the flexibility of our overall container implementation)
- LCF – Possible Solution? (Slide 11)
  - Assumes there will be edge services allowing some fixes to be available on worker node
  - Not expect CVMFS; Singularity would be useful

Implementation needs moving forward (Slide 12)

- Resource providers use different technologies and configurations for the same technologies
- Rule uniformity or, at least, tools/services for seamless and transparent transitions between different instantiations with minimum overhead
- HEP still needs to fully develop system services, distribution service and administration/configuration services (current efforts)
- Investigating available tools/services provided by the different container technologies to help achieve this goal

***Common Workflow Language (CWL): Portable workflow creation and deployment with the CWL Standards* - Michael R. Crusoe**

CWL – standard for describing Command Line tools and data analysis workflows made from those tools

- Many implementations, e.g., Reference implementation – allows for portable workflows
- Why have standards around workflows
  - Creates good ecosystem
  - Researchers use different data analysis platforms. Not standardize on a single system, but standardize in a way of moving between systems
  - Creating wiki of existing research workflow systems (over 200 currently)

<u>Moving from legacy workflow to standardized approach</u>

- Bioinformatics approach: European Bioinformatics Institute has 9522 lines of Python, BASH and Perl code (data analysis of logic mixed with operational details) which were converted into 2560 lines of CWL description (i.e., can visualize and manage it; put on GitHub)
- Pros of standardized approach to workflows: 1) data analysis – choices and parameters; 2) how will we run it optimally (look at data locality, match resources to right shape of compute)

<u>CWL Use Example</u>

- Gijs Molenaar converted radio astronomy pipelines to CWL version
- Enabled to run on Dutch national supercomputer; Used singularity processing

<u>CWL standards</u>

- Originated from bioinformatics because data analysis workflow has different components/sources. Extended to hydrology, high energy physics and beyond

<u>REANA system</u>

- Young system focusing on last mile data analysis
- Has CWL support and Kubernetes based

<u>CWL model</u>

- POSIX Command Line tools connected to each other; modularity is helpful for resource matchmaking
- Data model has built in identifiers; helps with understanding data locality
- Contains a lot of data information in model available to workflow system that can optimize round
- Open standards; run publicly

<u>Containers</u>

- Container v 1.0 has built in support for Docker software containers
- Reference runner has support for running Docker containers using Docker, Singularity, uDocker or x-Docker runtimes
- CWL tool descriptions can also contain more generic software requirements; can use to matchmake if using other software distribution methods

<u>Future version of CWL standard</u> (2.0 or 3.0) will switch from "Docker" image format for software containers to Open Container Initiative image format standard

- Recent developments: IBM Spectrum Computing will add native CWL support to LSF by end of Q2
  - Base runner will be Open Source project

- Toil workflow system has gained HTCondor support. Expected to be a new version of the standards

Atlas Computing Model - Wei Yang

Atlas: LHC experiment at CERN (Slide 3)

- Adopted Grid computing model (highly distributed model): works, but high operational cost
- Containers: essential to standardize environment, reduce operational cost, utilize HPC resources

Containers on Grid and as a Service (Slide 4)

- Recent Slate project
  - Uses Kubernetes to manage/deploy containerized services
  - Includes Grid CE whose interface allows external users to submit jobs to our batch system
- Future monetary analytic platform at CERN (began building)
  - Will use Kubernetes and Containerizer search to enable analytics on monetary data
  - Likely to use with Kubernetes for a deployment strategy
- Individual containerized services in environments that we run
  - Choice of Docker or Singularity
  - Single Atlas software release environment that people can build is relatively small; give to software developer for continued development

Container for Scientific computing on Grid sites (Slide 5)

Historically, ran jobs directly on batch nodes. Need heavy customization per site for ATLAS batch jobs.

- Want to reduce work for scientists.
- Can customize onto Singularity container. Prefer Singularity due to privilege concern (Docker runs as root).

2 methods to start Singularity container

1) Directly hook container with batch system.
   a. Already in container when jobs land, but need to do for every use case- not scalable (BNL).
2) Future route: payload starts container:
   a. Job lands on batch system. Find Singularity software and start container when needed.
   b. Applications take care of bind mount etc.

Containers on HPCs - address 2 problems on HPCs (Slide 6)

1) HPCs typically overwhelmed by Metadata IO, open and small file IO.
2) HPCs lack CVMFS (POSIX file system with global replications).
   a. As it replicates files only when used, it is a useful tool designed for our environment.
   b. ATLAS depends on CVMFS to distribute its software environments.
      i. Problem: CVMFS requires FUSE, which most do not wish to require.
   c. We initially build software environment on HPC shared filesystems.
      i. Problem: Costly to install/maintain and in different environments
3) Solution: Containers (different types: Shifter, Singularity)

Solution: FAT Container on HPCS (Slides 7-8)

- Put all software and operating system into a container (red line scales well)
- LBL experiments (Yellow line– software in Cori burst buffer; does not scale well)
- Put OS and CVMFS in container:

    - Allows software distribution without having CVMFS on HPC side, so same software environment. 400GB image can be mounted/loaded as fast as a 400 MB image
    - Run up to 1000 nodes (limit on shared file system)
    - With container, scale up to 3000 nodes (not reach limit on shared file system)

ORNL similar studies on Titan – pre-Container use (Slide 9)

- When one job runs, thousands of files opened and closed at start and end of job, does not scale well
- In a container- file opens/closes within container; there is no impact on performance (Slide 10)

Summary (Slide 11)

- ATLAS is working on using Docker/Kubernetes to deploy services
- Experimenting on various uses of Singularity containers in grid sites
- In HPC environments, use FAT container to distribute our software in a standard way and allow system to reduce IO to shared file systems, to enable scaling jobs on HPC centers

***Containing BayesWave: Deploying LIGO analyses on the OSG* - James Clark**

Background: Gravitational waves were first detected in 2015; birth of gravitational wave astronomy

Data analysis problem (Slide 4)

- Observational data: noisy strain time series from each detector
- Transient detection problem: signals with known and unknown morphology
- Computational problems in transient analyses
    - Performing Monte Carlo simulations requires meaningful statistical significance
    - Bayesian source characterization challenges, depend if signal is known or unknown
        - Wish to run parallel jobs to characterize and validate of source characterization simulations and background studies

LIGO Compute Resources (Slides 5-6)

- 2 runtime software environments for and gravitational wave analyses: LIGO Data Grid (traditionally) and, more recently, Open Science Grid (17%)
- LDG sites: computational and data storage resources with grid computing middleware create coherent LIGO data analysis environment; users manually partition workflows across sites

LIGO Analysis and OSG (began to access in 2015) (Slide 7)

- Data access challenges: used to input data and instrument observations provided at the sites.
    - OSG- collection of geographically distributed OSG sites that host LIGO data locally; otherwise StashCache and CVMFS as extensions; publish data to CVMSF repository.
- Software availability: LDG sites provide compute and data storage.
    - Currently, Docker images deployed automatically to CVMFS repository via gitlab

Case Study – BayesWave (Slide 8)

- Evaluations on waveform reconstructions and hypothesis of GW data
- Significant if able to leverage additional compute resources into a dedicated search analysis (with background and foreground estimations)

Containing BayesWave (Slide 9)

- If new user logged into LIGO dedicated clusters and pull data, resource code would make and build natively in LDG environment. Main dependency is LIGO algorithm
- Along with deployment of containers, LIGO started maintaining Docker images, including a nightly build of LAL Suite (see link), which BayesWave support developer pulls and add BayesWave specific commands to do job

Build & Deploy (Slide 10)

- Continuous integration script living in LIGO list of gitlab repositories (BayesWave should be publicly viewable). Use gitlab CI script to build BayesWave and Docker container

BayesWave and Singularity (Slide 11): Running BayesWave in singularity container via wrapper script

- Need to figure out how to bind CVMFS data repository to accessible directory inside container
- Running this analysis is seamless between LIGO resources, OSG and local work stations

Summary (Slide12, See also supplementary slides)

- Gravitational wave astronomy is increasingly resource-hungry
- Dedicated LIGO resources need to make more efficient use of resources such as OSG.
- Containers will be a game changer; Each user within the gravitational wave community would deploy their own containers to OSG. More expansion may become possible

Next MAGIC meeting: April 4, 2018 at the National Coordination Office, 490 L'Enfant Plaza, Suite 8001