DOE Collaborative Projects Meeting
Sponsored by the U.S. Department of Energy
Office of Advanced Scientific Computing Research
September 23–25, 2019

Virtual Clusters
for Community Computation

Douglas Thain, University of Notre Dame
Rob Gardner, University of Chicago
John Hover, Brookhaven National Lab
Outline

● Intro to the Virtual Cluster Concept
● Self-service Provisioning of Middleware Across Heterogeneous Resources
● Dealing with Software Environments
● Clustering Middleware Examples
● New Applications and Configurations
● Thoughts and Lessons Learned
Introduction
VC3: A platform for provisioning cluster frameworks over heterogeneous resources for collaborative science teams
You have developed a complex workload which runs successfully at one site, perhaps your home university.

Now, you want to migrate and expand that application to national-scale infrastructure. And allow others to easily access and run similar workloads.

Leadership HPC Facility

Distributed Computing Facility

Commercial Cloud
200 nodes of 36 cores and 128GB RAM
100GB local disk per node
ATLAS software release and python 2.7
CentOS 7 Operating System
Running the HTCondor batch system.
Shared between Amir, Ye, and Max’s allocations

Concept: Virtual Cluster

Virtual Cluster Service

Virtual Cluster Head Node

Virtual Cluster Head Node

Virtual Cluster Head Node

Deploy Software

Leadership HPC Facility

Deploy Software

Distributed Computing Facility

Deploy Software

Commercial Cloud
Concept: Virtual Cluster

Share with collaborators.

SSH Access

Virtual Cluster Service

Virtual Cluster Head Node

Virtual Cluster Head Node

Deploy Software

Leadership HPC Facility

Deploy Software

Distributed Computing Facility

Deploy Software

Commercial Cloud

Virtual Cluster
VC3: Virtual Clusters for Community Computation

- **VC3** is an interactive service for creating/sharing/using virtual clusters.
- A virtual cluster consists of:
  - 1 x **head node** for interactive access to the cluster. (SSH, Jupyter,..)
  - N x **worker nodes** for executing your workload.
  - **Middleware** to manage the cluster. (HTCondor, Makeflow, Spark, ...)
  - **Application** software to do real work. (BLAST, CMSSW, etc...)
- A virtual cluster is created using:
  - Your standard accounts/credentials on existing facilities.
  - Plain ssh/qsub/srun access on each facility.
  - Container technology (if available) or user-level software builds (otherwise).
  - (No special privileges or admin access required on the facility.)
Self-Service Provisioning
login with a vc3 account
Institutional Identity (CI-Logon)

Log in to use VC3

Use your existing organizational login

- University of Notre Dame

Didn't find your organization? Then use Globus ID to sign in. (What's this?)

Continue

Globus uses CILogon to enable you to Log In from this organization. By clicking Continue, you agree to the CILogon privacy policy and you agree to share your username, email address, and affiliation with CILogon and Globus. You also agree for CILogon to issue a certificate that allows Globus to act on your behalf.

Or

Sign in with Google

Sign in with ORCID iD
# Curated Resources

<table>
<thead>
<tr>
<th>Name</th>
<th>Organization</th>
<th>Description</th>
<th>Cores</th>
<th>Memory</th>
<th>Storage</th>
<th>Native OS</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cori</td>
<td>National Energy Research Scientific Computing Center (NERSC)</td>
<td>Cori Supercomputer at NERSC</td>
<td>32</td>
<td>4000 MB</td>
<td>10000 MB</td>
<td>susev12</td>
<td>Shifter</td>
</tr>
<tr>
<td>MWT2</td>
<td>Midwest Tier 2</td>
<td>ATLAS Midwest Tier 2 Center job gateway (UChicago)</td>
<td>4</td>
<td>1000 MB</td>
<td>1000 MB</td>
<td>scientificlinuxv6.9</td>
<td>N/A</td>
</tr>
<tr>
<td>Midway</td>
<td>University of Chicago Research Computing Center (RCC)</td>
<td>Midway cluster at the University of Chicago Research Computing Center (RCC)</td>
<td>64</td>
<td>4000 MB</td>
<td>10000 MB</td>
<td>scientificlinuxv6.7</td>
<td>N/A</td>
</tr>
<tr>
<td>Stampede 2</td>
<td>Texas Advanced Computing Center (TACC)</td>
<td>Stampede 2 Super Computer</td>
<td>96</td>
<td>2000 MB</td>
<td>10000 MB</td>
<td>centosv7.4</td>
<td>Singularity</td>
</tr>
<tr>
<td>CoreOS</td>
<td>University of Chicago</td>
<td>CoreOS Cluster</td>
<td>4</td>
<td>1000 MB</td>
<td>1000 MB</td>
<td>scientificlinuxv6.9</td>
<td>Singularity</td>
</tr>
<tr>
<td>UCT3</td>
<td>University of Chicago</td>
<td>UChicago ATLAS Tier 3</td>
<td>4</td>
<td>1000 MB</td>
<td>1000 MB</td>
<td>scientificlinuxv6.9</td>
<td>N/A</td>
</tr>
<tr>
<td>ND CCL</td>
<td>University of Notre Dame Cooperative Computing Lab</td>
<td>ND-CCL login name</td>
<td>4</td>
<td>1000 MB</td>
<td>10000 MB</td>
<td>redhatv7</td>
<td>Singularity</td>
</tr>
<tr>
<td>Bridges</td>
<td>Pittsburgh Supercomputing Center</td>
<td>Bridges Supercomputer at PSC</td>
<td>28</td>
<td>4000 MB</td>
<td>35000 MB</td>
<td>centosv7.3</td>
<td>Singularity</td>
</tr>
<tr>
<td>VC3 Test Pool</td>
<td>VC3</td>
<td>VC3 Test Pool</td>
<td>4</td>
<td>1000 MB</td>
<td>1000 MB</td>
<td>centosv6.9</td>
<td>N/A</td>
</tr>
<tr>
<td>UCLA Hoffman2</td>
<td>University of California, Los Angeles</td>
<td>UCLA Hoffman2</td>
<td>6</td>
<td>1000 MB</td>
<td>10000 MB</td>
<td>centosv6.9</td>
<td>N/A</td>
</tr>
<tr>
<td>OSG Connect</td>
<td>Open Science Grid</td>
<td>Open Science Grid (SL7)</td>
<td>4</td>
<td>1000 MB</td>
<td>1000 MB</td>
<td>Unknown</td>
<td>N/A</td>
</tr>
</tbody>
</table>
Allocations

Step 1: Log Into Resource

In a terminal type:

```
ssh btwvar@cclvm05.crc.nd.edu
```

Step 2: Access Resource

Enter your password for `cclvm05.crc.nd.edu` for access

Step 3: Add Allocation SSH Public Key to Resource

Once the SSH key is generated below, click ‘Copy to Clipboard’ and paste the following line into your SSH session. You will only need to do this once per allocation.

```
<Your generated SSH key here>
```

Step 4: Validate Allocation
# Projects

## Project Profiles

<table>
<thead>
<tr>
<th>Name</th>
<th>Members</th>
<th>Allocations</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vc3-team</td>
<td>Benjamin Tovar (Owner) - <a href="mailto:btovar@nd.edu">btovar@nd.edu</a></td>
<td>btovar-ndccl</td>
<td>Currently no description</td>
</tr>
<tr>
<td></td>
<td>Lincoln Bryant (UChicago)</td>
<td>kurtado-osgconnect</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Jeremy Van (UChicago)</td>
<td>lincolnrb-midway</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Robert Gardner (UChicago)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Kenyi Hurtado (University of Notre Dame)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>btovar</td>
<td>Benjamin Tovar (Owner) - <a href="mailto:btovar@nd.edu">btovar@nd.edu</a></td>
<td>btovar-ndccl</td>
<td>Currently no description</td>
</tr>
<tr>
<td></td>
<td>Benjamin Tovar (University of Notre Dame)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Launching a Virtual Cluster

<table>
<thead>
<tr>
<th>VIRTUAL CLUSTER NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>my-virtual-cluster</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CLUSTER TEMPLATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>lincoln-b-tcondor-10-workers</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ENVIRONMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>btovar-oasis-osg</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ALLOCATIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nothing selected</td>
</tr>
</tbody>
</table>

Select Allocations for Virtual Cluster

- Select All
- Deselect All

- btovar-ndccl
- khurtado-osgconnect
- lincolnb-midway

Workers will have this environment installed.

Shared cluster definition

Allocations available in this project
# Cluster Status

## My Virtual Clusters

<table>
<thead>
<tr>
<th>Name</th>
<th>State</th>
<th>Cluster Template</th>
<th>Workers</th>
<th>Head Node</th>
</tr>
</thead>
<tbody>
<tr>
<td>my-virtual-cluster</td>
<td>Running</td>
<td>lincolnb-hfcondor-10-workers</td>
<td>Requested: 10</td>
<td>128.135.158.187</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Running: 7</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Queued: 3</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Error: 0</td>
<td></td>
</tr>
</tbody>
</table>

All requested compute workers are running.
Workers from many sites

```
[btovar@btovar-my-virtual-cluster ~]$ ip addr | grep 128.135.158.187
  inet 128.135.158.187/25 brd 128.135.158.255 scope global dynamic eth0
[btovar@btovar-my-virtual-cluster ~]$ condor_status
```

<table>
<thead>
<tr>
<th>Name</th>
<th>OpSys</th>
<th>Arch</th>
<th>State</th>
<th>Activity</th>
<th>LoadAv</th>
<th>Mem</th>
<th>ActvtyTime</th>
</tr>
</thead>
<tbody>
<tr>
<td>slot1@<a href="mailto:glidein_21791@camd01.crc.nd.edu">glidein_21791@camd01.crc.nd.edu</a></td>
<td>LINUX</td>
<td>X86_64</td>
<td>Unclaimed Idle</td>
<td>5.120</td>
<td>4013</td>
<td>0:00</td>
<td>19:37</td>
</tr>
<tr>
<td>slot1@<a href="mailto:glidein_29106@camd01.crc.nd.edu">glidein_29106@camd01.crc.nd.edu</a></td>
<td>LINUX</td>
<td>X86_64</td>
<td>Unclaimed Idle</td>
<td>5.120</td>
<td>4013</td>
<td>0:00</td>
<td>19:37</td>
</tr>
<tr>
<td>slot1@<a href="mailto:glidein_91802@camd05.crc.nd.edu">glidein_91802@camd05.crc.nd.edu</a></td>
<td>LINUX</td>
<td>X86_64</td>
<td>Unclaimed Idle</td>
<td>5.260</td>
<td>4013</td>
<td>0:00</td>
<td>19:37</td>
</tr>
<tr>
<td>slot1@<a href="mailto:glidein_39133@iut2-c257.iu.edu">glidein_39133@iut2-c257.iu.edu</a></td>
<td>LINUX</td>
<td>X86_64</td>
<td>Unclaimed Idle</td>
<td>34.620</td>
<td>3223</td>
<td>0:00</td>
<td>19:48</td>
</tr>
<tr>
<td>slot1@<a href="mailto:glidein_61297@lnxfarm275.colorado.edu">glidein_61297@lnxfarm275.colorado.edu</a></td>
<td>LINUX</td>
<td>X86_64</td>
<td>Unclaimed Idle</td>
<td>6.990</td>
<td>3002</td>
<td>0:00</td>
<td>14:36</td>
</tr>
<tr>
<td>slot1@<a href="mailto:glidein_28373@midway091.rcc.local">glidein_28373@midway091.rcc.local</a></td>
<td>LINUX</td>
<td>X86_64</td>
<td>Unclaimed Idle</td>
<td>8.170</td>
<td>2013</td>
<td>0:00</td>
<td>19:36</td>
</tr>
<tr>
<td>slot1@<a href="mailto:glidein_71179@midway098.rcc.local">glidein_71179@midway098.rcc.local</a></td>
<td>LINUX</td>
<td>X86_64</td>
<td>Unclaimed Idle</td>
<td>7.480</td>
<td>2013</td>
<td>0:00</td>
<td>19:36</td>
</tr>
<tr>
<td>slot1@<a href="mailto:glidein_46364@midway260.rcc.local">glidein_46364@midway260.rcc.local</a></td>
<td>LINUX</td>
<td>X86_64</td>
<td>Unclaimed Idle</td>
<td>8.040</td>
<td>2013</td>
<td>0:00</td>
<td>19:35</td>
</tr>
<tr>
<td>slot1@<a href="mailto:glidein_39282@midway324.rcc.local">glidein_39282@midway324.rcc.local</a></td>
<td>LINUX</td>
<td>X86_64</td>
<td>Unclaimed Idle</td>
<td>8.750</td>
<td>2013</td>
<td>0:00</td>
<td>19:36</td>
</tr>
<tr>
<td>slot1@<a href="mailto:glidein_39133@uct2-c373.mwt2.org">glidein_39133@uct2-c373.mwt2.org</a></td>
<td>LINUX</td>
<td>X86_64</td>
<td>Unclaimed Idle</td>
<td>34.080</td>
<td>2415</td>
<td>0:00</td>
<td>19:33</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Machines</th>
<th>Owner</th>
<th>Claimed</th>
<th>Unclaimed</th>
<th>Matched</th>
<th>Preempting</th>
<th>Drain</th>
</tr>
</thead>
<tbody>
<tr>
<td>X86_64/LINUX</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Total</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
### VC3 Monitoring

#### Virtual Cluster Size (queued)

<table>
<thead>
<tr>
<th>Time</th>
<th>Workers</th>
</tr>
</thead>
<tbody>
<tr>
<td>14:10</td>
<td>10</td>
</tr>
<tr>
<td>14:15</td>
<td>20</td>
</tr>
<tr>
<td>14:20</td>
<td>30</td>
</tr>
</tbody>
</table>

#### My Virtual Clusters

<table>
<thead>
<tr>
<th>Name</th>
<th>State</th>
<th>Cluster Template</th>
<th>Workers</th>
<th>Head Node</th>
</tr>
</thead>
<tbody>
<tr>
<td>ndt3_500</td>
<td>Running</td>
<td>khurtado-htcondor-500workers</td>
<td>100</td>
<td>128.135.158.178</td>
</tr>
</tbody>
</table>

Requesting 114 more compute workers.
Dealing with Software Environments
Deploying Software Environments

The **vc3-builder**, a command-line tool for deploying software environments on clusters.

```bash
vc3-builder
  --require-os centos:7
  --mount /scratch=/data
  --require /cvmfs
  --require python:2.7 -- myapp ...my args...
```

[https://github.com/vc3-project/vc3-builder](https://github.com/vc3-project/vc3-builder)
vc3-builder --require ncbi-blast

..Plan:  ncbi-blast => [, ]
..Try:   ncbi-blast => v2.2.28
....Plan:  perl => [v5.008, ]
....Try:   perl => v5.10.0
....could not add any source for: perl v5.010 => [v5.8.0, ]
....Try:   perl => v5.16.0
....could not add any source for: perl v5.016 => [v5.8.0, ]
....Try:   perl => v5.24.0
......Plan:  perl-vc3-modules => [v0.001.000, ]
......Try:   perl-vc3-modules => v0.1.0
......Success: perl-vc3-modules v0.1.0 => [v0.1.0, ]
....Success: perl v5.24.0 => [v5.8.0, ]
....Plan:  python => [v2.006, ]
....Try:   python => v2.6.0
....could not add any source for: python v2.006 => [v2.6.0, ]
....Try:   python => v2.7.12
......Plan:  openssl => [v1.000, ]
        
 bash$  which blastx
/tmp/test/vc3-root/x86_64/redhat6/ncbi-blast/v2.2.28/bin/
blastx

bash$ blastx –help
USAGE
  blastx [-h] [-help] [-import_search_strategy filename]
Problem: Long Build on Head Node

• Many facilities limit the amount of work that can be done on the head node, so as to maintain quality of service for everyone.

• Solution: Move the build jobs out to the cluster nodes. (Which may not have network connections.)

• Idea: Reduce the problem to something we already know how to do: Workflow! But how do we bootstrap the workflow software? With the builder!
Bootstrapping Workflows and Apps

- Software Recipes
- Upstream Sources
- Head Node
- Builder
- Makeflow
- Build Makeflow
- Build BLAST
- BLAST Task
Example Applications

Octave

Typically, 2-3x faster overall. But more importantly, filesystem-intensive jobs run on the cluster resources, not on the head node!
Controlling Cluster Size
VC3 Supports several mechanisms for setting the number of workers in a given virtual cluster.

- Set static size for cluster upon creation.
- Manually change cluster size via web interface. The provisioning factory automatically adds or removes worker jobs at resources.
- (For HTCondor cluster) Scale the cluster dynamically based on idle jobs on the head node.
# Current Resizing Cases

<table>
<thead>
<tr>
<th>Source of desired number of workers</th>
<th>Job Removal Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>User (via web interface)</td>
<td><strong>Peaceful = True</strong></td>
</tr>
<tr>
<td>Increasing:</td>
<td>Works as expected.</td>
</tr>
<tr>
<td>Decreasing:</td>
<td>Workers will terminate when they have finished current job. By default, the younger workers go away first.</td>
</tr>
<tr>
<td>Decreasing:</td>
<td>Running jobs may be killed. But cluster can be reduced rapidly.</td>
</tr>
<tr>
<td>VC3 (automatic, dynamic, based on user's job pressure on head node.)</td>
<td>Increasing: Works as expected.</td>
</tr>
<tr>
<td></td>
<td>Decreasing: Workers will terminate when they have finished current job. By default, the younger workers go away first.</td>
</tr>
</tbody>
</table>
Scaling Out Production Clusters

- Collaborations who have existing HTCondor pools can extend them by adding more worker nodes via VC3
- Add XSEDE resources, Open Science Grid, and campus HPC clusters
- End-users can transparently use additional resources
Building Tier 3s on top of campus resources

- Although different computing resources are often available at universities, meeting all requirements to deploy a valid Tier 3 able to run CMS workflows on the grid is not an easy task to achieve without the intervention of a system administrator with root access.
- VC3 allows the provisioning at user-level of:
  - The CERN File System (CVMFS) (via parrot)
  - The OSG grid environment on the worker nodes (via CVMFS)
  - Customized Operating Systems (via singularity)
- The OSG Compute Element (CE) is then integrated with the VC3 submit host, allowing the creation of a CMS Tier 3 using Notre Dame opportunistic campus resources without any root access level.

CMS Analysis activity per Tier Site

Note T3_US_VC3_NotreDame performing at the scale of the Tier 2s!
Examples of clustering middleware
Use Cases

- **HTCondor Pool**
  - Already demonstrated self-service, fully automated provisioning clusters, binding allocations into projects

- **Workflow Technologies**
  - Makeflow + Work Queue
  - Pegasus + HTCondor,
  - Parsl + iPyParallel

- **Spark**
  - Deployment of a Spark cluster for data analysis on top of existing schedulers

- **JupyterLab**
  - Popular interactive analytics environment – provisionable by VC3

- **SCAILFIN + REANA**
  - Complex set of REANA services deployed on minikube on head node + HTCondor on Stampede, Blue Waters and PSC.

- **KOTO**
  - Helping a new collaboration with no established resources to run on BNL HPC, KEK HPC + OSG HTC + campus resources.
HTCondor Cluster on VC3

Virtual Cluster Service

Request VC

SSH to VC

Virtual Cluster Head Node

condor schedd

condor collector

condor negot’r

HTCondor Cluster

deploy software

Site A

SSH to VC

SLURM Cluster

deploy software

Site B

PBS Cluster

deploy software

Site C
Workflows on VC3

Virtual Cluster Service

VCS

Makeflow

Site A

HTCondor Cluster

Deploy Software

condor startd

condor schedd

condor collector

condor negot’r

Site B

SLURM Cluster

Deploy Software

condor startd

Site C

PBS Cluster

Deploy Software

condor startd

condor startd

condor startd

condor startd

condor startd

condor startd
Workflows on VC3

Virtual Cluster Service

Virtual Cluster Interface

Work Queue Master

Source Data

HTCondor Cluster

Deploy Software

Site A

$$$

Parsl

Makeflow

Site B

$$$

Site C

$$$

Work Queue

Master

Source Data

Site A

$$$

Site B

$$$

Site C

$$$

Work Queue

Master

Work Queue

Master

Work Queue

Master

Work Queue

Master

Work Queue

Master

Work Queue

Master
Spark Middleware Integration

- Apache Spark now a first-class supported middleware in VC3
- Spark master runs on the virtual cluster head-node
- Virtual cluster workers are spark slaves
  - Java JRE, spark, scala, pyspark are deployed as-needed by the target resource.
- A shared secret secures connection to the master from workers and application clients
- No shared filesystem, use of an IO driver is needed.
Spark Cluster on VC3

Virtual Cluster Service

Request VC

SSH to VC

Spark Master

XRootD Data Server

Virtual Cluster Head Node

HTCondor Cluster

Deploy Software

Site A

SLURM Cluster

Deploy Software

Site B

PBS Cluster

Deploy Software

Site C

Spark

Master

XRootD

Data

Server

condor startd

Spark Worker

Request VC

SSH to VC

No Global Shared Filesystem
Spark & CMS Analysis

- Late-stage custom analysis code processing a Mini-AOD file to produce a histogram.
- Application written in python, using pyspark
- Data IO with the spark-xrootd plugin
  - Automatically installed in headnode and workers
- Ran with 20 workers, 4 cores each.
- Maximum concurrency was 80 tasks.
Spark & CMS Analysis

Virtual Cluster: btovar-zpeak

Terminate Cluster

STATE OF VIRTUAL CLUSTER

Running

All requested compute workers are running.

Running Applications

<table>
<thead>
<tr>
<th>App ID</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>app.201809191230309-0000</td>
<td>Zpawk, MonoJed 68</td>
</tr>
<tr>
<td>app.201809191230309-0000</td>
<td>Zpawk, MonoJed 68</td>
</tr>
<tr>
<td>app.201809191230309-0000</td>
<td>Zpawk, MonoJed 68</td>
</tr>
<tr>
<td>app.201809191230309-0000</td>
<td>Zpawk, MonoJed 68</td>
</tr>
</tbody>
</table>

Status

BTMOVAR-ZPKEA
Cluster Framework

Requested 3
Running 2
Quoted 5
Error 2

Data Workers

39
Provisioning JupyterLab Notebooks

- JupyterLab-based head nodes now launchable in VC3 development branch
- Web-based notebook interface for Python
- Uses Globus identity plugin to login
- Integration with HTCondor and other middleware types
Provisioning JupyterLab Notebooks

Select items to perform actions on them.

- 07-Kalman-Filter-Math.ipynb
- Balancing Reactions.ipynb

Example: Hypergolic Reactions

Hypergolic reactions are reactions where the reactants spontaneously ignite. They are frequently used in space propulsion where it is desired to have a thruster that can be readily controlled over a range of operating conditions.
Credential Delegation:
- Globus Auth to VC3
- VC3 WWW to Jupyter
- VC3 WWW to Site SSH
- User to Jupyter
- Worker to Master
- Worker to XRootD
Each one a different technology!
SCAILFIN: Scalable CyberInfrastructure for Artificial Intelligence and Likelihood Free Inference

The SCAILFIN project aims to deploy artificial intelligence and likelihood-free inference techniques and software using scalable cyberinfrastructure (CI) that is developed to be integrated into existing CI elements, such as the REANA system, to work on HPC facilities.

REANA is automatically deployed and integrated with HTCondor in VC3

REANA components are started via kubernetes(minikube)
Simulation-Based Likelihood Free Inference

Symbolically:

\[ r(x, z|\theta) \]

\[ t(x, z|\theta) \]

\[ \text{augmented data} \]

\[ \text{Regression to optimal estimator} \]

\[ \arg \min_{g} L[g] \]

\[ \hat{r}(x|\theta) \]

\[ \text{approximate likelihood ratio} \]

\[ \theta_j \]

\[ \theta_i \]

Simulation

Machine Learning

Inference

Estimation of optimal estimator lends itself to ML methods:

- Training data derived from simulations
- Can be guided by optimal sampling based on phase space density of generator, sensitivity to physics under study
reana-cluster - Simplified Diagram

(CERN Instance)

CERN OpenStack COE + Kubernetes

REANA Cluster / Service Components

- DB / SQLAlchemy
  TCP 5432

- Workflow Mon
  ZMQ Proxy
  TCP 8666/8667

- Msg Broker
  RabbitMQ
  TCP 5672, 15672

- Workflow Engine & Controller

REANA Cluster / Workers

- Yadage-run Docker -> Payload
- Yadage-run Docker -> Payload
- Yadage-run Docker -> Payload

Job Controller

W.E Yadage+ Serial

WDB
TCP 1984, 19840
SCAILFIN on Blue Waters via VC3

VC3 Headnode

REANA Components

- Reana-Job-Controller
- Condor Schedd collector/CCB

VC3 Factory

- Bosco (GSI-SSH)
- Torque

Internet

reverse connection from condor startd to CCB/collector.

BW Submit Node

MOM Node

VC3-glidein

Compute Node

Run Shifter Payload for REANA workflow step

aprun -b -- shifter...
Data is generated/emulated according to Standard Model expectations.

After processing, a statistical model involving both signal and control regions is built and the model is fitted against the observed data.

The signal sample is scaled down significantly to fit the data, which is expected since the data was emulated in accordance with a SM-only scenario.
KOTO experiment

High energy physics experiment at the J-PARC (Japan Proton Accelerator Research Complex) Facility in Japan - [http://koto.kek.jp/](http://koto.kek.jp/)

Experiment measures the decay rate of neutral K-mesons (kaons) into neutral pi-mesons and a pair of neutrinos
Requirements

Storage: Estimated storage for accommodating the MC and MC/Analysis portions of the pipeline for a full experiment cycle is in the order of 200 TB per experiment cycle.

Software stack: There are two applications running on the KEKCC machine that needed to be rebuilt on the OSG side under RHEL7. A KOTO GEANT package and an analysis collection of tools. Most of the effort so far is to provide the environment for the software to be built and run on remote OSG sites.

Submission scripts: KEKCC uses the LSF scheduler (bsub). Submission scripts needed to be modified to HTCondor.
VC3 job submission for KOTO

- OSG Connect (blue), UC (yellow), IRIS-HEP-SSL (orange)
- VC3 evenly distributes the load across resources
Working Middleware and Applications

- Various Bioinformatics Workflows
  - Makeflow + HTCondor + BWA, Shrimp, BLAST.
- Lobster CMS Data Analysis
  - Work Queue + Builder + CVMFS
- Lifemapper
  - Work Queue + Builder
- Spark CMS Data Analysis
- South Pole Telescope (SPT-3G) Analysis Framework
  - HTCondor Jobs + Docker/Shifter + CVMFS
- XENON1T Analysis Framework
  - Pegasus + HTCondor + CVMFS
- MAKER Bioinformatics Pipeline
  - Work Queue + Builder
- IceCube Simulation Framework
  - HTCondor
Thoughts and Lessons Learned
Successful Ideas that Worked

- Automated Provisioning of Cluster Frameworks Across Heterogeneous HPC Resources
- Automated Deployment of User-Level Software
  - (Build itself may require substantial cluster resources)
- Internal Architecture for Robust Service Deployment
  - (Surprisingly complex state machine.)
- Multi-Layer Configuration Enables Collaborations
  - Credentials go deep into the framework
Things That were Harder than Expected

● New Security Requirements (MFA) Spread Very Quickly
  ○ Working Idea: Let user login and “pull” the cluster blueprint to the local site where it can be deployed.

● Difficulty of Curating Site/Cluster Details due to Churn
  ○ Choice: Either invest more in active curation, or involve users in the configuration process.

● Debugging Complex Service Deployments
  ○ Working Idea: Test before using, expose stages of success to user, give concrete feedback for fixing.
Portable Ideas to Keep and Use Elsewhere

● Importance of the Head Node as Infrastructure
  ○ Allocatable, Personal, Shareable, Configurable

● Self-Service Deployment Model
  ○ Generalization of Infrastructure-as-Code:
    ○ OpSys + Software + Middleware + Entry Point

● A Model for Sharing Institutional Resources
  ○ Classic: Submit My Work to Remote Environments
  ○ VC3: Deploy My Environment on Remote Resources
Technology Evolution 2016–2020

Evolution from VMs to Containers:
- Swap out OpenStack in favor of Kubernetes, which is lightweight and better supports process automation.
- Docker is being supplanted by Singularity, and other container environments.

Evolution from credentials to capabilities:
- ssh-keygen -> Upload Pubkey -> RSA-SSH (Standalone)
- Globus Auth -> GSI Token -> GSI-SSH (DCDE Fed.)
- Globus Auth -> Web Tokens -> WebTokens-SSH (Web Native)
- SciTokens? (scitokens.org)
Towards Federated Ops with SLATE

- Remotely manage edge services at sites by **expert teams** from **trusted organizations**
- Deploy updates more quickly & introduce new services more easily
- Save time and effort for local site admins -- towards OSG NoOps
- Edge federation via lightweight server/client overlay using **Kubernetes**, the industry leading container orchestration platform

- Software catalog, with push button deploy using vetted **Helm** charts

```
$ slate instance list
$ slate instance delete <instance name>
$ slate app install --group atlas-xcache --cluster uchicago-prod --conf MWT2.yaml xcache
```
• **SLATE** - a value added K8s distribution
  o Support for CVMFS, ingress controller (multi-tenant, scoped privileges), Prometheus monitoring, **curated application catalog w/ Jenkins CI**

• Site security & policy conscious
  o SLATE works as an unprivileged user
  o Single entrypoint via institutional identity
  o Site owner controls group whitelists & service apps; **retains full control**

• With OSG, WLCG, trustedci.org & others working to establish a "CISO compliant" security posture and **new trust delegation model**

SLATE creates secrets and XCache deployment on cluster

XCache Container Download

Kubernetes objects instantiated

Pod starts up, registers itself in AGIS

A data caching network deployed in less than 20 minutes.

Upgrades are as simple as re-deploying.
VC3 and SLATE synergies

SLATE: federated service orchestration in the SciDMZ

Locate "pull model" VC3 services on site
Open Problems in Federated Systems

● Troubleshooting Across Systems
  ○ Many stakeholders, system churn, lack of evidence.
  ○ Troubleshooting posed as a distributed database query?

● State Management and Garbage Collection
  ○ Remote creation of containers for local sites, how long must they be kept around?

● Socio–Technical Understanding of Delegation
  ○ User X submits jobs to user Y’s account on site S. Ok?
  ○ Technical ability to represent non–local account?
Collaborators and Connections
VC3
Virtual Clusters for Community Computation

https://www.virtualclusters.org
@virtualclusters


Supported by the Department of Energy Office of Advanced Scientific Computing Research and Next Generation Networking Services, Solicitation DE-FOA0001344 (DDRM), Proposal 0000219942.
VC3 Funding and Team

Funded by DOE Office of Advanced Scientific Computing Research (ASCR) and NSF Next Generation Networking Services (NGNS)

PIs: Rob Gardner (UC), Douglas Thain (ND), and John Hover (BNL)

co-PIs: David Miller (UC), Paul Brenner (ND), Mike Hildreth (ND), Kevin Lannon (ND)

dev-team: Lincoln Bryant (UC), Benedikt Riedel (UC), Suchandra Thapa (UC), Jeremy Van (UC), Kenyi Hurtado Anampa (ND), Ben Tovar (ND), Jose Caballero Bejar (BNL).