Introduction to
Makeflow and Work Queue

Nate Kremer-Herman
Blue Waters Webinar
March 22nd, 2017
The Cooperative Computing Lab

• We *collaborate with people* who have large scale computing problems in science, engineering, and other fields.

• We *operate computer systems* on the O(10,000) cores: clusters, clouds, grids.

• We *conduct computer science* research in the context of real people and problems.

• We *develop open source software* for large scale distributed computing.
Our Philosophy:

- Harness all the resources that are available: desktops, clusters, clouds, and grids.
- Make it easy to scale up from one desktop to national scale infrastructure.
- Provide familiar interfaces that make it easy to connect existing apps together.
- Allow portability across operating systems, storage systems, middleware...
- Make simple things easy, and complex things possible.
- No special privileges required.
A Quick Tour of the CCTools

- Open source, GNU General Public License.
- Compiles in 1-2 minutes, installs in $HOME.
- Runs on Linux, Solaris, MacOS, FreeBSD, …
- Interoperates with many distributed computing systems.
  - Condor, SGE, SLURM, TORQUE, Globus, iRODS, Hadoop…
- Components:
  - **Makeflow** – A portable workflow manager.
  - **Work Queue** – A lightweight distributed execution system.
  - All-Pairs / Wavefront / SAND – Specialized execution engines.
  - Parrot – A personal user-level virtual filesystem.
  - Chirp – A user-level distributed filesystem.
makeflow(1)

NAME
makeflow - workflow engine for executing distributed workflows

SYNOPSIS
makeflow [options] <dagfile>

DESCRIPTION
Makeflow is a workflow engine for distributed computing. It accepts a specification of a large amount of work to be performed, and runs it on remote machines in parallel where possible. In addition, Makeflow is fault-tolerant, so you can use it to coordinate very large tasks that may run for days or weeks in the face of failures. Makeflow is designed to be similar to Make, so if you can write a Makefile, then you can write a Makeflow.

You can run a Makeflow on your local machine to test it out. If you have a multi-core machine, then you can run multiple tasks simultaneously. If you have a Condor pool or a Sun Grid Engine batch system, then you can send your jobs there to run. If you don't already have a batch system, Makeflow comes with a system called Work Queue that will let you distribute the load across any collection of machines, large or small.

OPTIONS
When makeflow is run without arguments, it will attempt to execute the workflow specified by the Makeflow dagfile using the local execution engine.

Commands

-c, --clean Clean up: remove logfile and all targets.
-f, --summary-log <file>
Write summary of workflow to file.
Recap from Last Workflow Webinar

• What is a workflow?
  • A collection of things to do (tasks) to reach a final result.

• What are the parts of a task?
  • The thing we want to do (application to run), input to give that application, output we expect to get from that application.

• How can a workflow management system help me do my research?
  • Add automation, resource provisioning, task scheduling, data management, etc.

bluewaters.ncsa.illinois.edu/webinars/workflows/overview-of-scientific-workflows
Makeflow:
A Portable Workflow System
An Old Idea: Makefiles

part1 part2 part3: input.data split.py
./split.py input.data

.out1: part1 mysim.exe
./mysim.exe part1 > out1

.out2: part2 mysim.exe
./mysim.exe part2 > out2

.out3: part3 mysim.exe
./mysim.exe part3 > out3

.result: out1 out2 out3 join.py
./join.py out1 out2 out3 > result
Makeflow = Make + Workflow

- Provides portability across batch systems.
- Enable parallelism (but not too much!).
- Trickle out work to batch system.
- Fault tolerance at multiple scales.
- Data and resource management.
Makeflow Syntax

[output files] : [input files]  
[command to run]

out.txt : in.dat calib.dat sim.exe  
sim.exe -p 50 in.data > out.txt
You must state all the files needed by the command.
example.makeflow

out.10 : in.dat calib.dat sim.exe
        sim.exe -p 10 in.data > out.10

out.20 : in.dat calib.dat sim.exe
        sim.exe -p 20 in.data > out.20

out.30 : in.dat calib.dat sim.exe
        sim.exe -p 30 in.data > out.30
Sync Point - Questions?

• Several additional features to Makeflow which we do not have time to cover today (please take a look at our documentation).

• Categories and resource specification.

• Shared filesystems support.

• Container support (Docker and Singularity).

  ccl.cse.nd.edu/software/manuals/makeflow.html
Let’s work through a brief tutorial:

ccl.cse.nd.edu/software/tutorials/ncsatut17/makeflow-tutorial.php
Makeflow + Work Queue
Makefile

Makeflow

XSEDE Torque Cluster

Private Cluster

Campus Condor Pool

Public Cloud Provider

Local Files and Programs

makeflow –T torque

makeflow –T condor

???
Makeflow + Work Queue

- Makefile
- Makeflow
- Local Files and Programs
- ssh

Submit tasks to:
- torque_submit_workers
- condor_submit_workers

Workers in a Personal Cloud:
- XSE Torque Cluster
- Private Cluster
- Campus Condor Pool
- Public Cloud Provider

Thousands of Workers in a Personal Cloud
Advantages of Work Queue

• Harness multiple resources simultaneously.
• Hold on to cluster nodes to execute multiple tasks rapidly. (ms/task instead of min/task)
• Scale resources up and down as needed.
• Better management of data, with local caching for data intensive tasks.
• Matching of tasks to nodes with data.
Project Names

makeflow …
–N myproject

work_queue_worker
–N myproject

---

Makeflow (port 9050)

Catalog

Worker

connect to
workflow.iu:9050

advertise

query

work_queue_status

query

“myproject”
is at workflow.iu:9050
<table>
<thead>
<tr>
<th>PROJECT</th>
<th>NAME</th>
<th>PORT</th>
<th>WAITING</th>
<th>BUSY</th>
<th>COMPLETE</th>
<th>WORKERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>awe-fip35</td>
<td>fahnd04.crc.nd.edu</td>
<td>1024</td>
<td>719</td>
<td>1882</td>
<td>1206967</td>
<td>1882</td>
</tr>
<tr>
<td>hfeng-gromacs-10ps</td>
<td>lclsstor01.crc.nd.edu</td>
<td>1024</td>
<td>4980</td>
<td>0</td>
<td>1280240</td>
<td>111</td>
</tr>
<tr>
<td>hfeng2-ala5</td>
<td>lclsstor01.crc.nd.edu</td>
<td>1025</td>
<td>2404</td>
<td>140</td>
<td>1234514</td>
<td>140</td>
</tr>
<tr>
<td>forcebalance</td>
<td>leeping.Stanford.EDU</td>
<td>5817</td>
<td>1082</td>
<td>26</td>
<td>822</td>
<td>26</td>
</tr>
<tr>
<td>forcebalance</td>
<td>leeping.Stanford.EDU</td>
<td>9230</td>
<td>0</td>
<td>3</td>
<td>147</td>
<td>3</td>
</tr>
<tr>
<td>fg-tutorial</td>
<td>login1.futuregrid.tacc</td>
<td>1024</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Resilience and Fault Tolerance

• MF +WQ is fault tolerant in many different ways:
  • If Makeflow crashes (or is killed) at any point, it will recover by reading the transaction log and continue where it left off.
  • Makeflow keeps statistics on both network and task performance, so that excessively bad workers are avoided.
  • If a worker crashes, the master will detect the failure and restart the task elsewhere.
  • Workers can be added and removed at any time during the execution of the workflow.
  • Multiple masters with the same project name can be added and removed while the workers remain.
  • If the worker sits idle for too long (default 15m) it will exit, so it does not hold resources while idle.
Let’s return to the tutorial:

ccl.cse.nd.edu/software/tutorials/ncsatut17/makeflow-tutorial.php
Visit our website: ccl.cse.nd.edu

Follow us on Twitter: @ProfThain

Check out our blog: cclnd.blogspot.com

Makeflow examples:
github.com/cooperative-computing-lab/makeflow-examples