Introduction to Makeflow and Work Queue

Prof. Douglas Thain, University of Notre Dame

http://www.nd.edu/~dthain
dthain@nd.edu
@ProfThain
Lots of information here:
http://ccl.cse.nd.edu

The Cooperative Computing Lab

About the CCL
We design software that enables our collaborators to easily harness large scale distributed systems such as clusters, clouds, and grids. We perform fundamental computer science research in that enables new discoveries through computing in fields such as physics, chemistry, bioinformatics, biometrics, and data mining.

CCL News and Blog
- Virtual Wind Tunnel in IEEE CiSE (09 Sep 2015)
- Three Papers at IEEE Cluster in Chicago (07 Sep 2015)
- CCTools 5.2.0 released (18 Aug 2015)
- Recent CCL Grad Takes Faculty Positions (18 Aug 2015)
- CMS Analysis on 10K Cores Using Lobster (14 Aug 2015)
- CCTools 5.1.0 released (16 Jul 2015)
- CCTools 5.0.0 released (07 Jul 2015)
- Preservation Framework for Computational Reproducibility at ICCS 2015 (01 Jul 2015)

Community Highlight
ForceBalance is an open source software tool for creating accurate force fields for molecular mechanics simulation using flexible combinations of reference data from experimental measurements and theoretical calculations. These force fields are used to simulate the dynamics and physical properties of molecules in chemistry and biochemistry.

The Work Queue framework gives ForceBalance the ability to distribute computationally intensive components of a force field optimization calculation in a highly flexible way. For example, each optimization cycle launched by ForceBalance may require running 50 molecular dynamics simulations, each of which may take 10-20 hours on a high end NVIDIA GPU. While GPU computing resources are available, it is rare to find 50 available GPU nodes on any single supercomputer or HPC cluster. With Work Queue, it is possible to distribute the simulations across several HPC clusters, including the Certainty HPC cluster at Stanford, the Keeneland GPU cluster managed by Georgia Tech and Oak Ridge National Laboratories, and the Stampede supercomputer managed by the University of Texas. This makes it possible to run many simulations in parallel and complete the high level optimization in weeks instead of years.

- Lee-Ping Wang, Stanford University
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.

http://www.nd.edu/~ccl
Science Depends on Computing!

AGTCCGTACGATGCTATTAGCGAGCGTGA...
## Users of Condor Cycles

### Notre Dame Condor Status

<table>
<thead>
<tr>
<th>User</th>
<th>Slots</th>
<th>Cores</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="mailto:mzh4u@nd.edu">mzh4u@nd.edu</a></td>
<td>1465</td>
<td>1465</td>
</tr>
<tr>
<td><a href="mailto:awoodard@nd.edu">awoodard@nd.edu</a></td>
<td>67</td>
<td>1072</td>
</tr>
<tr>
<td><a href="mailto:jsarro@nd.edu">jsarro@nd.edu</a></td>
<td>170</td>
<td>680</td>
</tr>
<tr>
<td><a href="mailto:khuang3@nd.edu">khuang3@nd.edu</a></td>
<td>527</td>
<td>527</td>
</tr>
<tr>
<td><a href="mailto:hhatami@nd.edu">hhatami@nd.edu</a></td>
<td>426</td>
<td>426</td>
</tr>
<tr>
<td><a href="mailto:cswert1@nd.edu">cswert1@nd.edu</a></td>
<td>287</td>
<td>287</td>
</tr>
<tr>
<td><a href="mailto:klannon@nd.edu">klannon@nd.edu</a></td>
<td>123</td>
<td>123</td>
</tr>
<tr>
<td>Unclaimed</td>
<td>89</td>
<td>443</td>
</tr>
<tr>
<td>Matched</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>Preempting</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Owner</td>
<td>55</td>
<td>573</td>
</tr>
<tr>
<td>Total</td>
<td>3213</td>
<td>5605</td>
</tr>
</tbody>
</table>

### Display Options

- **Sort:** users, machines
- **Show:** users, states
- **Size:** bigger, smaller
- **Scale:** none, cores
Superclusters at Amazon

$1,279-per-hour, 30,000-core cluster built on Amazon EC2 cloud

By Jon Brodkin | Published a day ago

I can get as many machines on the cloud/grid as I want!

How do I organize my application to run on those machines?
The Cooperative Computing Tools
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, Cygwin, FreeBSD, ...
- Interoperates with many distributed computing systems.
  - Condor, SGE, 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 file system.
  - Chirp – A user-level distributed filesystem.

http://ccl.cse.nd.edu/software
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.

http://ccl.cse.nd.edu/software/software/makeflow
Makeflow Syntax

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

out.txt : calib.dat in.dat sim.exe
./sim.exe –p 50 in.data > out.txt
You must state all the files needed by the command.
sims.mf

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
How to run a Makeflow

• Run a workflow locally, using multiple cores:
  – makeflow -T local sims.mf

• Run the workflow on Torque:
  – makeflow -T torque sims.mf

• Run the workflow on Condor:
  – makeflow -T condor sims.mf

• Run the workflow on SLURM:
  – makeflow -T slurm sims.mf
You should see this:

% makeflow -T local sims.mf
parsing sims.mf...
checking sims.mf for consistency...
sims.mf has 3 rules.
starting workflow....
submitting job: ./sim.exe -p 30 in.data > out.30
submitted job 2035
submitting job: ./sim.exe -p 20 in.data > out.20
submitted job 2036
submitting job: ./sim.exe -p 10 in.data > out.10
submitted job 2037
job 2035 completed
job 2036 completed
job 2037 completed
nothing left to do.
If you do the same thing twice:

% makeflow -T local sims.mf
parsing sims.mf...
checking sims.mf for consistency...
sims.mf has 3 rules.
recovering from log file sims.mf.makeflowlog...
starting workflow....
nothing left to do.

Makeflow keeps a log of operations, so it knows which jobs have been sent to the batch system, and which files have already been created.
Automatically clean outputs:

% makeflow --clean sims.mf
parsing sims.mf...
checking sims.mf for consistency...
sims.mf has 3 rules.
recovering from log file sims.mf.makeflowlog...
cleaning filesystem...

Note that you do *not* have to write a “clean” rule. Makeflow just figures it out for you.
Some more handy options:

- Limit the number of jobs running at once:
  --max-local #
  --max-remote #

- Retry jobs that have a tendency to fail:
  --retry-count=5

- Send email when the workflow is done:
  --email user@domain.com

- Monitor the resources consumed by each job:
  --monitor <output-dir>
Visualization with DOT

- `makeflow_viz -D example.mf > example.dot`
- `dot -T gif < example.dot > example.gif`

DOT and related tools:
http://www.graphviz.org
Makeflow Shapes a Workflow

Millions of Tasks

Concurrency Control

Make Flow

Precise Cleanup

Transaction Log

Performance Monitoring

Thousands of Nodes
Example: Biocompute Portal

My Data

Generate Makefile

Action

Run Workflow

Makeflow

Progress Bar

Transaction Log

Update Status

Submit Tasks

BLAST
SSAHA
SHRIMP
EST
MAKER

Condor Pool

Submit Tasks

Biocompute Portal

My Queue

BLAST
SSAHA
SHRIMP
EST
MAKER
Makeflow Applications
Makeflow + Work Queue
Makeflow can send jobs to one batch system at a time.

Makefile

Makeflow

Local Files and Programs

FutureGrid
Torque Cluster

Campus
Condor Pool

Private
Cluster

Public
Cloud Provider
Makeflow + Work Queue can harness multiple clusters at once.

Makefile

Makeflow

submit tasks

Local Files and Programs

Thousands of Workers in a Personal Cloud

condor_submit_workers

ssh

FutureGrid

Torque Cluster

Private Cluster

Public Cloud Provider

Campus Condor Pool
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.
Makeflow and Work Queue

First, start the Makeflow:
% makeflow –T wq  sims.mf
Could not create work queue on port 9123.

Whoops, try again:
% makeflow –T wq --port 0  sims.mf
Listening for workers on port 8374...

Start one worker and tell it where to find makeflow:
% work_queue_worker master.hostname.org 8374
Start 25 Workers in Batch System

Submit workers to Condor:
condor_submit_workers master.hostname.org 8374 25

Submit workers to SGE:
sge_submit_workers master.hostname.org 8374 25

Submit workers to Torque:
torque_submit_workers master.hostname.org 8374 25
Keeping track of port numbers gets old fast...
makeflow ...
-N myproject

work_queue_worker
-N myproject

Makeflow
(port 4057)

Worker

connect to
india:4057

advertise

Catalog

query

“myproject”
is at india:4057

query
Project Names

Start Makeflow with a project name:
% makeflow –T wq –N myproject  sims.mf
Listening for workers on port XYZ...

Start one worker:
% work_queue_worker -N myproject

Start many workers:
% torque_submit_workers -N myproject  5
<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 as not to hold resources idle.
Lots more information here:
http://ccl.cse.nd.edu

The Cooperative Computing Lab

About the CCL

We design software that enables our collaborators to easily harness large scale distributed systems such as clusters, clouds, and grids. We perform fundamental computer science research that enables new discoveries through computing in fields such as physics, chemistry, bioinformatics, biometrics, and data mining.

CCL News and Blog

- Virtual Wind Tunnel in IEEE CiSE (09 Sep 2015)
- Three Papers at IEEE Cluster in Chicago (07 Sep 2015)
- CCTools 5.2.0 released (18 Aug 2015)
- Recent CCL Grads Take Faculty Positions (18 Aug 2015)
- CMS Analysis on 10K Cores Using Lobster (14 Aug 2015)
- CCTools 5.1.0 released (16 Jul 2015)
- CCTools 5.0.0 released (07 Jul 2015)
- Preservation Framework for Computational Reproducibility at ICCS 2015 (01 Jul 2015)
- (more news)

Community Highlight

**ForceBalance** is an open source software tool for creating accurate force fields for molecular mechanics simulation using flexible combinations of reference data from experimental measurements and theoretical calculations. These force fields are used to simulate the dynamics and physical properties of molecules in chemistry and biochemistry.

The **Work Queue** framework gives ForceBalance the ability to distribute computationally intensive components of a force field optimization calculation in a highly flexible way. For example, each optimization cycle launched by ForceBalance may require running 50 molecular dynamics simulations, each of which may take 10-20 hours on a high end NVIDIA GPU. While GPU computing resources are available, it is rare to find 50 available GPU nodes on any single supercomputer or HPC cluster. With Work Queue, it is possible to distribute the simulations across several HPC clusters, including the Certainty HPC cluster at Stanford, the Keeneland GPU cluster managed by Georgia Tech and Oak Ridge National Laboratories, and the Stampede supercomputer managed by the University of Texas. This makes it possible to run many simulations in parallel and complete the high level optimization in weeks instead of years.

-Lee-Ping Wang, Stanford University

Research

- Papers
- Projects
- People
- Jobs
- REU

Software

- Download
- Manuals
- Makeflow
- Work Queue
- Parrot
- Chip
- SAND
- AWE

Community

- Forum
- Getting Help
- Highlights
- Annual Meeting
- Workshops
- For Developers

Operations

- Condor Display
- Condor Pool
- Hadoop Cluster
- Biocompute
- IXGrid
- Condor Log Analyzer
- Internal