Scaling Up Without Blowing Up

Douglas Thain
University of Notre Dame
The Cooperative Computing Lab
University of Notre Dame

http://www.nd.edu/~ccl
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 release open source software for large scale distributed computing.

http://www.nd.edu/~ccl
Good News:
Computing is Plentiful
CPU Utilization for the Last Week

404855  (51%) CPU-Hours Unused
328960  (41%) CPU-Hours Used by Condor
58935   (7%)   CPU-Hours Used by Owner
792750  (100%) CPU-Hours Total
Superclusters by the Hour

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

By Jon Brodkin | Published a day ago

Our Collaborators
The Cooperative Computing Tools

**Makeflow**

Makeflow is a workflow system for parallel and distributed computing that uses a language very similar to Make. Using Makeflow, you can write simple scripts that easily execute on hundreds or thousands of machines.

**Parrot**

Parrot is a transparent user-level virtual filesystem that allows any ordinary program to be attached to many different remote storage systems, including HDFS, iRODS, Chirp, and FTP.

**Work Queue**

Work Queue is a system and library for creating and managing scalable master-worker style programs that scale up to thousands of machines on clusters, clouds, and grids. Work Queue programs are easy to write in C, Python or Perl.

**Chirp**

Chirp is a personal user-level distributed filesystem that allows unprivileged users to share space securely, efficiently, and conveniently. When combined with Parrot, Chirp allows users to create custom wide-area distributed filesystems.

http://www.nd.edu/~ccl
Example: Adaptive Weighted Ensemble

1: Start with Standard MD Simulation

2: Research Goal: Generate Network of States

3: Build AWE Logic Using Work Queue Library

4: Run on 2000+ CPUs/GPUs at Notre Dame, Stanford, and Amazon

5: New State Transition Discovered!

Badi Abdul-Wahid, Li Yu, Dinesh Rajan, Haoyun Feng, Eric Darve, Douglas Thain, Jesus A. Izaguirre,
Folding Proteins at 500 ns/hour with Work Queue, IEEE e-Science Conference, 2012
The Ideal Picture

\[
\text{X 1000}
\]
What actually happens:

1 TB

128 GB

GPU

3M files of 1K each

X 1000
Some reasonable questions:

• Will this workload at all on machine X?
• How many workloads can I run simultaneously without running out of storage space?
• Did this workload actually behave as expected when run on a new machine?
• How is run X different from run Y?
• If my workload wasn’t able to run on this machine, where can I run it?
End users have no idea what resources their applications actually need.

and...

Computer systems are terrible at describing their capabilities and limits.

and...

They don’t know when to say NO.
dV/dt: Accelerating the Rate of Progress Towards Extreme Scale Collaborative Science

Miron Livny (UW), Ewa Deelman (USC/ISI), Douglas Thain (ND), Frank Wuerthwein (UCSD), Bill Allcock (ANL)

... make it easier for scientists to conduct large-scale computational tasks that use the power of computing resources they do not own to process data they did not collect with applications they did not develop ...
Stages of Resource Management

• **Estimate** the application resource needs
• **Find** the appropriate computing resources
• **Acquire** those resources
• **Deploy** applications and data on the resources
• **Manage** applications and resources during run.

• Can we do it for **one task**?
• How about an app composed of **many tasks**?
Categories of Applications

Concurrent Workloads

Static Workloads

Regular Graphs

Irregular Graphs

Dynamic Workloads

while( more work to do)
{
    foreach work unit {
        t = create_task();
        submit_task(t);
    }
    t = wait_for_task();
    process_result(t);
}
Bioinformatics Portal Generates Workflows for Makeflow

- BLAST (Small): 17 sub-tasks, ~4h on 17 nodes
- BWA: 825 sub-tasks, ~27m on 100 nodes
- SHRIMP: 5080 sub-tasks, ~3h on 200 nodes
239 Workflows

- Each site in the input map corresponds to one workflow
- Each workflow has:
  - 820,000 tasks

MPI codes ~ 12,000 CPU hours, Post Processing 2,000 CPU hours
Data footprint ~ 800GB

Pegasus managed workflows

Southern California Earthquake Center

CyberShake PSHA Workflow

- **Description**
  - Builders ask seismologists: “What will the peak ground motion be at my new building in the next 50 years?”
  - Seismologists answer this question using Probabilistic Seismic Hazard Analysis (PSHA)
Task Characterization/Execution

• Understand the resource needs of a task
• Establish expected values and limits for task resource consumption
• Launch tasks on the correct resources
• Monitor task execution and resource consumption, interrupt tasks that reach limits
• Possibly re-launch task on different resources
Data Collection and Modeling

Task Record
- RAM: 50M
- Disk: 1G
- CPU: 4 C

Records From Many Tasks
- RAM: 50M
- Disk: 1G
- CPU: 4 C

Task Profile
- RAM
  - min
  - typ
  - max

Workflow Schedule

Workflow Profile

Workflow Structure
Resource Monitor

**Log File:**

<table>
<thead>
<tr>
<th>wall_clock (useconds)</th>
<th>concurrent_processes</th>
<th>cpu_time (useconds)</th>
<th>virtual_memory (kB)</th>
<th>resident_memory (kB)</th>
<th>swap_memory (kB)</th>
<th>bytes_read</th>
<th>bytes_written</th>
</tr>
</thead>
<tbody>
<tr>
<td>1   1   0</td>
<td>1</td>
<td>8700</td>
<td>376</td>
<td>0</td>
<td>385024</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>2   5   20000</td>
<td>5</td>
<td>326368</td>
<td>6100</td>
<td>0</td>
<td>27381007</td>
<td>1474560</td>
<td></td>
</tr>
<tr>
<td>3   6   20000</td>
<td>6</td>
<td>394412</td>
<td>7468</td>
<td>0</td>
<td>29735839</td>
<td>1503232</td>
<td></td>
</tr>
<tr>
<td>4   8   60000</td>
<td>8</td>
<td>531468</td>
<td>14092</td>
<td>0</td>
<td>36917793</td>
<td>1503232</td>
<td></td>
</tr>
<tr>
<td>5   8   100000</td>
<td>8</td>
<td>532612</td>
<td>16256</td>
<td>0</td>
<td>39285593</td>
<td>1503232</td>
<td></td>
</tr>
</tbody>
</table>

% resource_monitor mysim.exe

**Summary File**

- **start:** 1367424802.676755
- **end:** 1367424881.236612
- **exit_type:** normal
- **exit_status:** 0
- **max_concurrent_processes:** 16
- **wall_time:** 78.559857
- **cpu_time:** 54.181762
- **virtual_memory:** 1051160
- **resident_memory:** 117604
- **swap_memory:** 0
- **bytes_read:** 4847233552
- **bytes_written:** 256950272
## Monitoring Strategies

<table>
<thead>
<tr>
<th>Indirect</th>
<th>Direct</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monitor how the world changes while the</td>
<td>Monitor what functions, and with which</td>
</tr>
<tr>
<td>process tree is alive.</td>
<td>arguments the process tree is calling.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Summaries</th>
<th>Snapshot</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>getrusage and times</td>
<td>Reading /proc and measuring disk at given intervals.</td>
<td>Linker wrapper to libc</td>
</tr>
<tr>
<td>Available only at the end of a process.</td>
<td>Blind while waiting for next interval.</td>
<td>Fragile to modifications of the environment, no statically linked processes.</td>
</tr>
</tbody>
</table>
while (more work to do) {
    foreach work unit {
        t = create_task();
        submit_task(t);
    }
    t = wait_for_task();
    process_result(t);
}
Resource Visualization of SHRiMP
Outliers Happen: BWA Example
Completing the Cycle

Allocate Resources
- CPU: 10s
- RAM: 16GB
- DISK: 100GB

Measurement and Enforcement

Exception Handling
Is it an outlier?

Historical Repository

Observed Resources
- CPU: 5s
- RAM: 15GB
- DISK: 90GB

P

min        typ        max
We can approach the question: Can it run on this particular machine? What machines could it run on?
Two jokes about computer scientists.
How do you tell the difference between a computer scientist and a real scientist?
What’s the difference between a **computer scientist** and an **engineer**?
Posters Preview

• Hierarchical Resource Management
  – Michael Albrecht

• Workflow Dependency Management
  – Casey Robinson

• Application Assembly Technologies
  – Peter Sempolinski
Acknowledgements

dV/dT Project PIs
- Bill Allcock (ALCF)
- Ewa Deelman (USC)
- Miron Livny (UW)
- Frank Weurthwein (UCSD)

CCL Staff
- Ben Tovar

CCL Graduate Students:
- Michael Albrecht
- Patrick Donnelly
- Dinesh Rajan
- Casey Robinson
- Peter Sempolinski
- Li Yu
The Cooperative Computing Lab

University of Notre Dame

http://www.nd.edu/~ccl

Makeflow

Makeflow is a workflow system for parallel and distributed computing that uses a language very similar to Make. Using Makeflow, you can write simple scripts that easily execute on hundreds or thousands of machines.

Work Queue

Work Queue is a system and library for creating and managing scalable master-worker style programs that scale up to thousands of machines on clusters, clouds, and grids. Work Queue programs are easy to write in C, Python or Perl.

Parrot

Parrot is a transparent user-level virtual filesystem that allows any ordinary program to be attached to many different remote storage systems, including HDFS, iRODS, Chirp, and FTP.

Chirp

Chirp is a personal user-level distributed filesystem that allows unprivileged users to share space securely, efficiently, and conveniently. When combined with Parrot, Chirp allows users to create custom wide-area distributed filesystems.