Portable Resource Management for Data Intensive Workflows

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
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
A Familiar Problem

![Diagram of a network with 'X 100' indicating a significant increase in scale]
What actually happens:

1 TB

128 GB

GPU

3M files of 1K each

X 100
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 Toward 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 ...
dV/dt Project Approach

- Identify challenging applications.
- Develop a framework that allows to characterize the application needs, the resource availability, and plan for their use.

- Threads of Research:
  - High level planning algorithms.
  - Measurement, representation, analysis.
  - Resource allocation and enforcement.
  - Resources: Storage, networks, memory, cores...?
  - Evaluate on major DOE resources: OSG and ALCF.
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);
}

Static Workloads

Concurrent Workloads

Dynamic Workloads
Bioinformatics Portal Generates Workflows for Makeflow

Bioinformatic Workflow Examples:

- **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
**Periodograms: generate an atlas of extra-solar planets**

- Find extra-solar planets by
  - Wobbles in radial velocity of star, or
  - Dips in star’s intensity

210k light-curves released in July 2010

Apply 3 algorithms to each curve

3 different parameter sets

- 210K input, 630K output files
- 1 super-workflow
- 40 sub-workflows
- ~5,000 tasks per sub-workflow
- 210K tasks total

*Pegasus managed workflows*
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)

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
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

Workflow Schedule

Workflow Profile

Workflow Structure
% resource_monitor mysim.exe

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</td>
<td>1</td>
<td>0</td>
<td>8700</td>
<td>376</td>
<td>0</td>
<td>385024</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>20000</td>
<td>326368</td>
<td>6100</td>
<td>0</td>
<td>27381007</td>
<td>1474560</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>20000</td>
<td>394412</td>
<td>7468</td>
<td>0</td>
<td>29735839</td>
<td>1503232</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>60000</td>
<td>531468</td>
<td>14092</td>
<td>0</td>
<td>36917793</td>
<td>1503232</td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td>100000</td>
<td>532612</td>
<td>16256</td>
<td>0</td>
<td>39285593</td>
<td>1503232</td>
</tr>
</tbody>
</table>

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 process tree is alive.</td>
<td>Monitor what functions, and with which arguments the process tree is calling.</td>
</tr>
</tbody>
</table>

### Summaries
- **Indirect**: getrusage and times
  - Available only at the end of a process.
- **Direct**: Reading /proc and measuring disk at given intervals.
  - Blind while waiting for next interval.

### Events
- Linker wrapper to libc
  - Fragile to modifications of the environment, no statically linked processes.
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

Exception Handling
Is it an outlier?

Observed Resources

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

Measurement and Enforcement

Historical Repository

min typ max
Multi-Party Resource Management

WF Coord

SDN

Batch System

Storage Allocator
while( more work to do) {
    foreach work unit {
        t = create_task();
        submit_task(t);
    }
    t = wait_for_task();
    process_result(t);
}
Coming up soon in CCTools...

• Makeflow
  – Integration with resource management.
  – Built-in linker pulls in deps to make a portable package.

• Work Queue
  – Hierarchy, multi-slot workers, cluster caching.
  – Automatic scaling of workers with network capacity.

• Parrot
  – Integration with CVMFS for CMS and (almost?) ATLAS.
  – Continuous improvement of syscall support.

• Chirp
  – Support for HDFS as a storage backend.
  – Neat feature: search() system call.
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.