Reduction of Workflow Resource Consumption Using a Density-based Clustering Model

Qimin Zhang, Nathaniel Kremer-Herman, Benjamin Tovar, and Douglas Thain

WORKS Workshop at Supercomputing 2018
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.

ccl.cse.nd.edu
Makeflow = Make + Workflow

- Provides portability across batch systems.
- Enables parallelism (but not too much!)
- Fault tolerance at multiple scales.
- Data and resource management.
- Transactional semantics for job execution.

http://ccl.cse.nd.edu/software/makeflow
Classic Make Language

MEMORY=8
CORES=4
DISK=32

output.5.txt: simulate.py
  /simulate.py –n 5 > output.5.txt

output.6.txt: simulate.py
  /simulate.py –n 6 > output.6.txt

...
JX (JSON + Expressions)

```json
{
    "command": "./simulate.py -n "+N+" > output."+N+".txt",
    "inputs": [ "simulate.py" ],
    "outputs": [ "output."+N+".txt" ],
    "resources": {
        "memory": 8,
        "cores": 16,
        "disk": 128
    }
} for N in range(1,1000)

...
Makeflow Shapes a Workflow

- Millions of Tasks
- Concurrency and Policy Control
- Make Flow
- Cluster or Cloud
- Precise Cleanup
- Transaction Log
- Performance Monitoring
Example: Species Distribution Modeling

Full Workflow: 12,500 species
x 15 climate scenarios
x 6 experiments
x 500 MB per projection
= 1.1M jobs, 72TB of output

Small Example: 10 species x 10 expts
How many resources should be requested for each job?
It is an important decision, but it is hard to determine.
The Job Sizing Problem

How should one select the resources (cores, memory, disk) required by each job in a workflow?

A high throughput workflow:
- hundreds of thousands of jobs;
- complex distributions of resource consumption.

Too small: the job may fail due to resource exhaustion.
Too large: the job may succeed, but resources will be wasted, fewer jobs will run, and throughput will be reduced.
The Job Sizing Problem

- 32 GB RAM
- 16 CPUs
- Machine Allocation

Jobs and Fragmentation:
- Internal Fragmentation
- External Fragmentation
Job Sizing is a Client's Problem (Not the Facility's Problem.)

User/WMS selects allocation:
- Too big? Wasted resources.
- Too small? Job fails, retry.

Facility places the allocation:
- Too big? Get paid.
- Too small? Still get paid!

Facility handles scheduling (placement) but not the job sizing problem.
Other Approaches and Their Limitations

Static Resource Allocation:
- An expert user sets the maximum resources used for all tasks.
- Easy to implement.
- Ignores different kinds of tasks.
- Ignores distribution of tasks.
- Expert knowledge is hard to come by, and may be incorrect.

Dynamic Resource Allocation:
- Resources are allocated as historical data becomes available.
- Harder to implement.
- User labels the different kinds of tasks.
- Takes into account distribution of tasks.
- Failures while distributions are learned.
Clustering Based Approach

- Max resources are set according to available historical data.
- Easier to implement.
- Different types of tasks can be learned without user intervention.
- Takes into account distribution of tasks.
- Needs apriori historical data.
- History may not represent the future.
Our approach can be applied when users need to:

1. Run the workflow multiple times.
2. Run a small batch of workflow before running the whole workflow.
A resource monitor tool is used to generate resource usage reports.

A density-based clustering model is applied to discover hidden clusters.

An improved resource allocation strategy is given by calculating the maximum resources of each cluster.

Run the remainder (or next) with the improved resource allocation strategy.
Density-based Clustering Model

Advantages of Density-based Clustering:
- Clusters found by density can be any shape;
- The number of clusters is not required to be specified a priori.

An Example: Clustering Result of BWA-GATK Workflow
Parameters of the Clustering Model

Features:
- cores, memory, disk, and wall-time

Minimal samples:
- the minimal number of samples in a cluster
- set to be 1 to ensure each job have its cluster label

Weights: \((w)\)
- represent the importance of features (or axes)
- weights of cores, memory, disk and wall-time were set to be 0.1, 0.7, 0.1, 0.1 respectively
Parameters of the Clustering Model

**Distance:**
- represents inherent similarity between two points
- The distance \( D(X, Y) \) between two points \( X = (x_1, x_2, x_3, x_4) \) and \( Y = (y_1, y_2, y_3, y_4) \) is defined as \((p=2)\):
  \[
  D(X, Y) = \left( \sum_{i=1}^{4} w \times \left| x_i - y_i \right|^p \right)^{1/p}
  \]

**eps:**
- The maximum distance between two samples for them to be considered as in the same cluster
- The value of \( eps \) should be modified in different workflows to achieve better performance.
Experiments on Scientific Workflows

Lifemapper  SHRIMP  BWA  BWA-GATK

https://github.com/cooperative-computing-lab/makeflow-examples
Lifemapper-Workflow

Estimated number of clusters: 3
SHRIMP-Workflow

Estimated number of clusters: 3
<table>
<thead>
<tr>
<th>Lifemapper</th>
<th>Job</th>
<th>Resource</th>
<th>Time (s)</th>
<th>Memory (GB.min)</th>
<th>Disk (GB.min)</th>
<th>Time saving</th>
<th>Memory saving (GB.min)</th>
<th>Disk saving (GB.min)</th>
<th>Cores saving</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1&lt;sup&gt;st&lt;/sup&gt;</td>
<td>153</td>
<td>8</td>
<td>8000</td>
<td>50000</td>
<td>1394.23</td>
<td>539.23</td>
<td>3370.22</td>
<td>compare with 1&lt;sup&gt;st&lt;/sup&gt; 84.83%</td>
<td>compare with 1&lt;sup&gt;st&lt;/sup&gt; 72.34%</td>
</tr>
<tr>
<td>2&lt;sup&gt;nd&lt;/sup&gt;</td>
<td>153</td>
<td>4</td>
<td>8000</td>
<td>50000</td>
<td>542.49</td>
<td>406.41</td>
<td>2540.03</td>
<td>compare with 2&lt;sup&gt;nd&lt;/sup&gt; 61.01%</td>
<td>compare with 2&lt;sup&gt;nd&lt;/sup&gt; 63.31%</td>
</tr>
<tr>
<td>3&lt;sup&gt;rd&lt;/sup&gt;</td>
<td>153</td>
<td>2</td>
<td>8000</td>
<td>50000</td>
<td>245.87</td>
<td>293.22</td>
<td>1832.64</td>
<td>compare with 3&lt;sup&gt;rd&lt;/sup&gt; 13.98%</td>
<td>compare with 3&lt;sup&gt;rd&lt;/sup&gt; 49.14%</td>
</tr>
<tr>
<td>After clustering</td>
<td>153</td>
<td>Cluster 1</td>
<td>100</td>
<td>3000</td>
<td>211.50</td>
<td>149.13</td>
<td>122.05</td>
<td>compare with 1&lt;sup&gt;st&lt;/sup&gt; 84.83%</td>
<td>compare with 1&lt;sup&gt;st&lt;/sup&gt; 72.34%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cluster 2</td>
<td>5000</td>
<td>30000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cluster 3</td>
<td>1500</td>
<td>3000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
SHRIMP, BWA, and BWA-GATK were run on an HTCondor batch system.

Compared to naive approach, the least resource saving of cores, memory, and disk are 50.00%, 91.14%, and 51.82%, respectively.
Second Order Effect:
Smaller Jobs are Easier to Schedule!

Larger resource requesting:
- harder to schedule within the cluster
- longer to wait in work queue

Smaller resource requesting:
- faster scheduling times
- faster to fulfill
Takeaways and Caveats

- Clustering-based resource allocation strategy significantly reduces resource consumption of scientific workflows.

- The density-based clustering model performs well in discovering hidden clusters or relationships between jobs.

- Caveat: User cannot avoid setting some knobs: Relative weights of resources. EPS parameter controlling max distance

- Caveat: Clustering requires some advance resource consumption. (How much pays off?)
Clustering: The Chicken or the Egg?

Unlabeled Workflow → Measure to Find Resources → Cluster Resources to find Groupings → Execute Workflow

Qimin Zhang, Nathaniel Kremer-Herman, Benjamin Tovar, and Douglas Thain, Reduction of Workflow Resource Consumption Using a Density-based Clustering Model, WORKS Workshop at Supercomputing, November 2018.
Clustering: The Chicken or the Egg?

User Provides Clusters (not resources) → Execute Workflow → Measure Resources of Clusters → Improve Predictions

But manual clusters might not be compact!

How to pick the first allocation?

Ben Tovar says:
Minimize probability of first attempt succeeding + fallback succeeding, weighted by resources.
- What is the minimal amount of useful a priori data?
  ▶ Simple categories provided by the user? Estimates of past resource consumption? Access to runs from "similar" workflows and users? (But what is similar?)

- Can we determine when clustering/predictions fail?
  ▶ Detecting one outlier is easy, but it's not practical to recluster at every step. What's the cost of doing this incrementally?

- Resource description is not entirely separate from scheduling!
  ▶ How much faster can I schedule a 32-core job than a 128...
Reduction of Workflow Resource Consumption Using a Density-based Clustering Model

Qimin Zhang, Nathaniel Kremer-Herman, Benjamin Tovar, and Douglas Thain

Support for this work is provided by an iSURE summer fellowship at Notre Dame. We thank Tim Shaffer and Kyle Sweeney, researchers at the University of Notre Dame, for assistance with Makeflow and the HTCondor distributed batch computing system.
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

- Preservation Talk at iPep 2015 (03 Nov 2015)
- CCL Case Study Paper at CHiP (20 Oct 2015)
- OpenMalaria Preservation with Umbrella (19 Oct 2015)
- Virtual Wind Tunnel in IEEE CSSE (09 Sep 2015)
- Three Papers at IEEE Cluster in Chicago (07 Sep 2015)
- CCL Tools 5.2.0 released (10 Aug 2015)
- Recent CCL Grids Take Faculty Positions (18 Aug 2015) (more news)

Community Highlight

Scientists searching for the Higgs boson have profited from Parrot's new support for the CondorX Filesystem (CVMS), a network filesystem tailored to providing world-wide access to software installations. By using Parrot, CVMS, and additional components integrated by theanyone, Anywhere, Anytime project, physicists working in the Compact Muon Solenoid experiment have been able to create a uniform computing environment across the Open Science Grid. Instead of maintaining large software installations at each participating institution, Parrot is used to provide access to a single highly-available CVMS installation of the software from which files are downloaded as needed and aggressively cached for efficiency. A pilot project at the University of Wisconsin has demonstrated the feasibility of this approach by exporting excess compute jobs to run in the Open Science Grid, opportunistically harnessing 370,000 CPU-hours across 35 sites with seamless access to 400 gigabytes of software in the Wisconsin CVMS repository.

- Dan Bradley, University of Wisconsin and the Open Science Grid

http://ccl.cse.nd.edu

@ProfThain