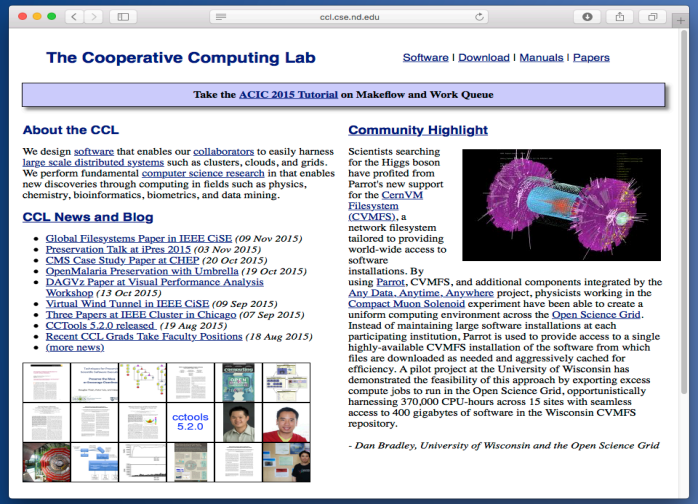
UNIVERSITY OF NOTRE DAME

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



ccl.cse.nd.edu

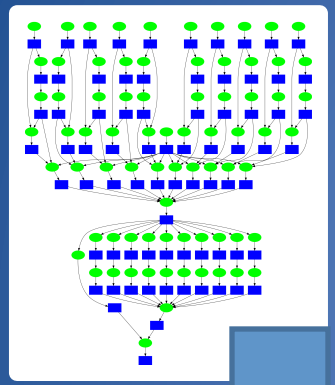
We *collaborate with people* who have large scale computing problems in science, engineering, and other fields.

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.

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.

Makeflow

Local | HTCondor | SLURM | Work Queue | Amazon

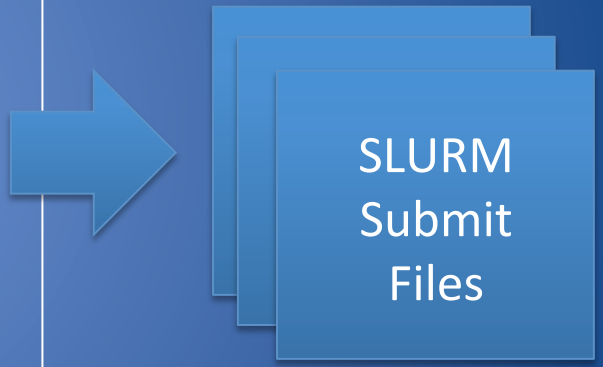
http://ccl.cse.nd.edu/software/makeflow

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
. . .
```



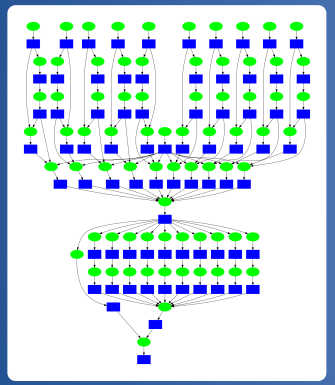
JX (JSON + Expressions)

```
{
  "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

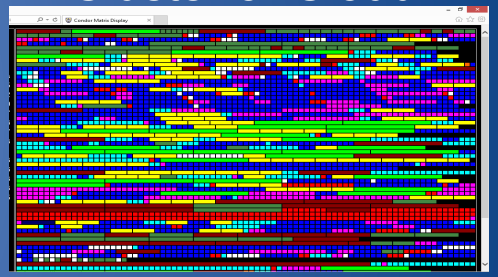
Concurrency
and Policy Control

Millions of Tasks



Make
Flow

Cluster or Cloud

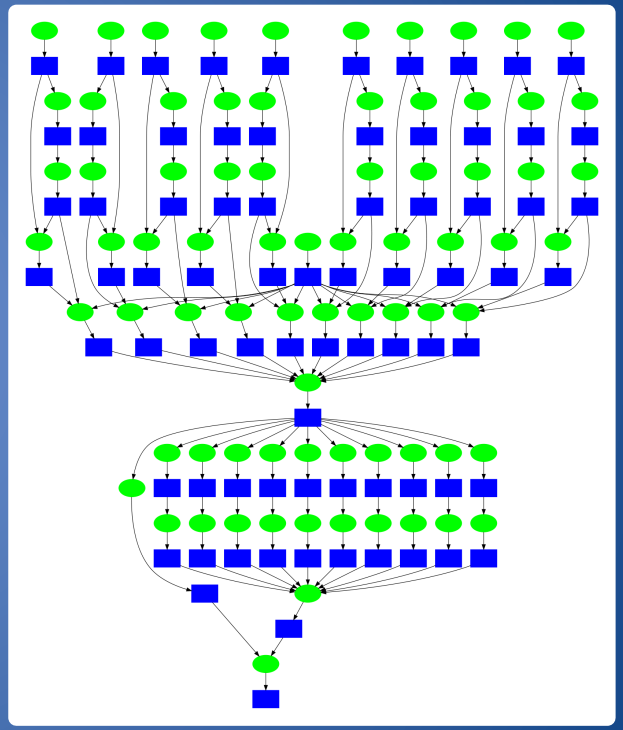


Precise
Cleanup

Transaction Log

Performance
Monitoring

Example: Species Distribution Modeling



Full Workflow: 12,500 species

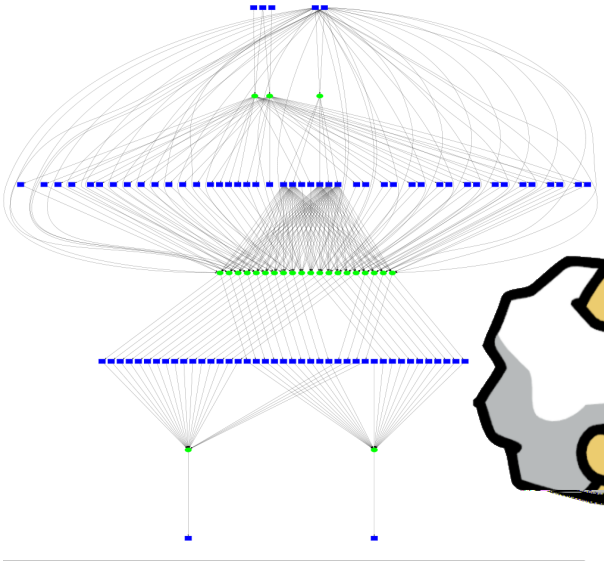
x 15 climate scenarios

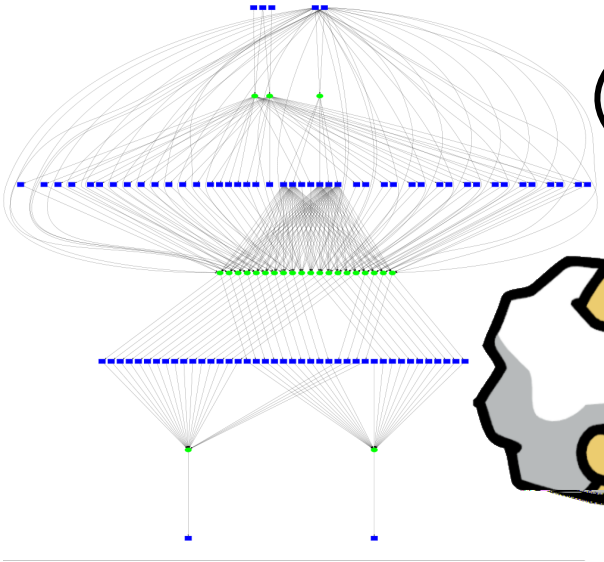
x 6 experiments

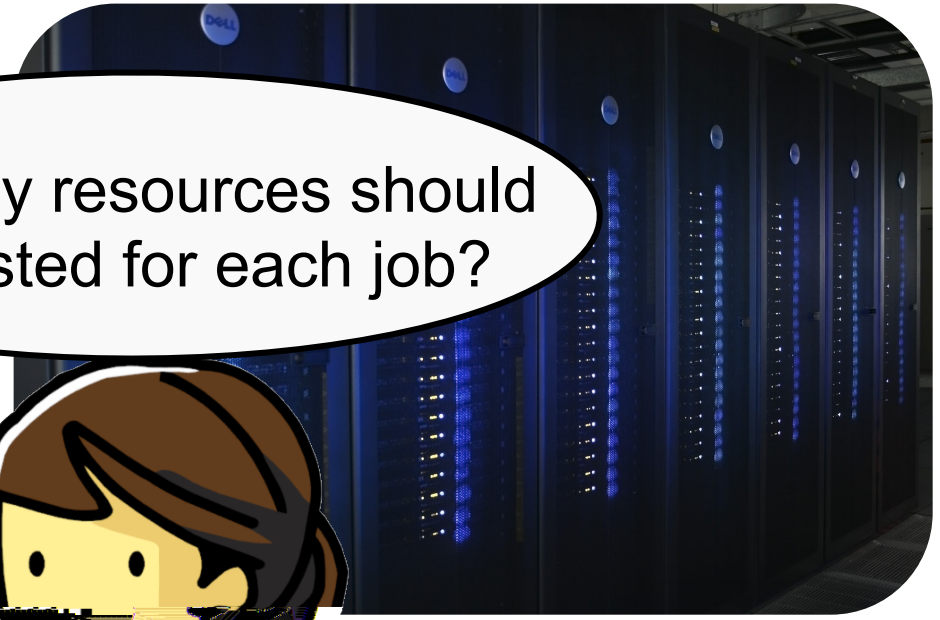
x 500 MB per projection

= 1.1M jobs, 72TB of output

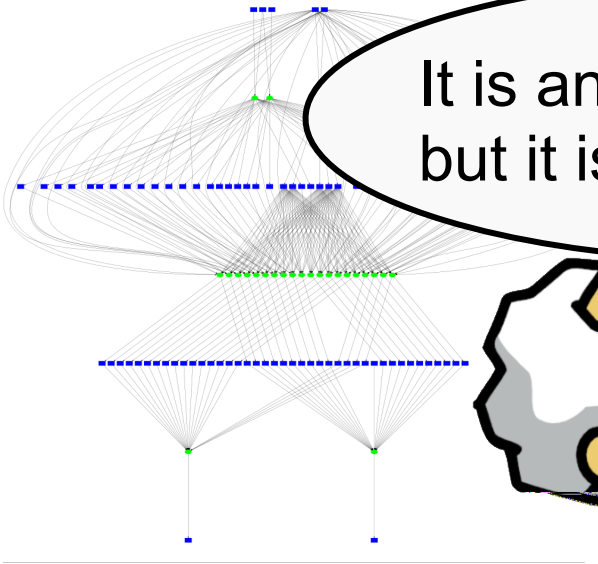
Small Example: 10 species x 10 expts







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?

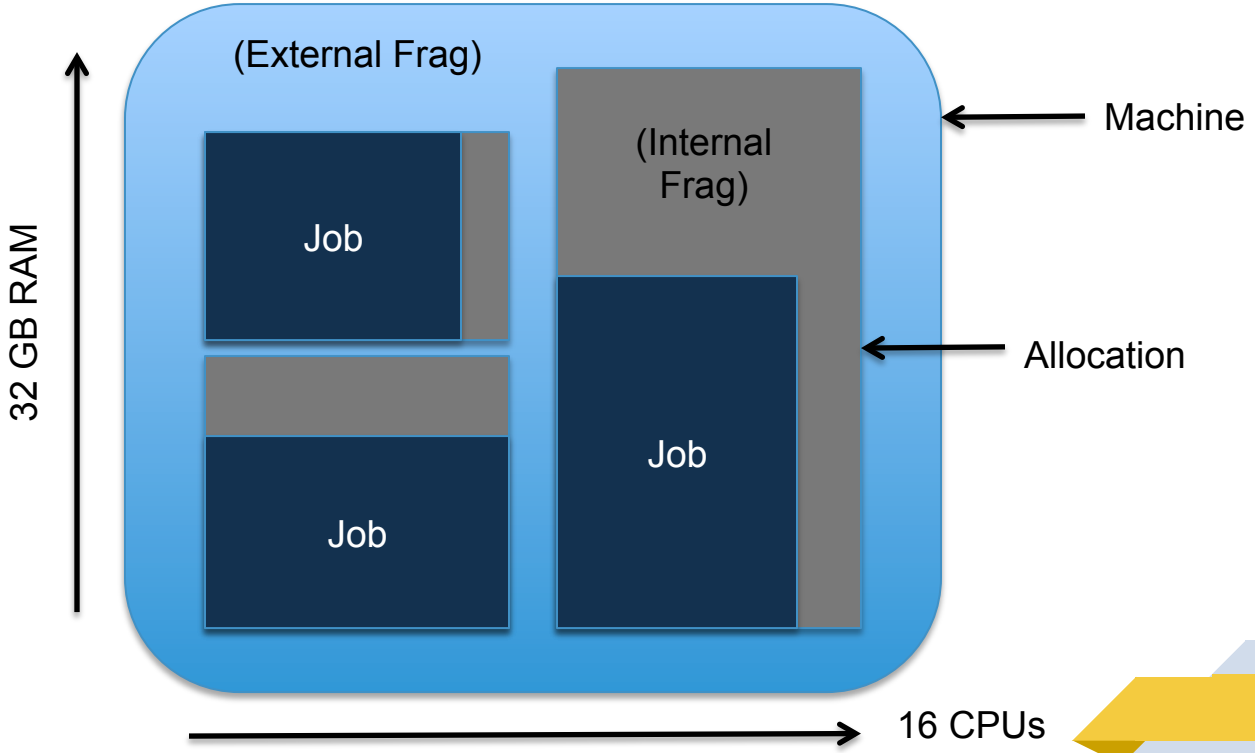
A high throughput workflow:

- 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



Job Sizing is a Client's Problem (Not the Facility's Problem.)

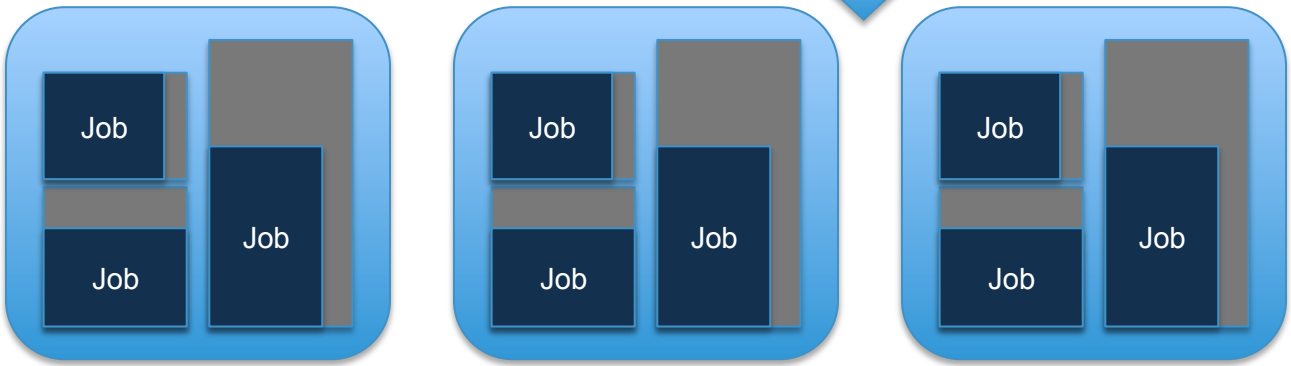
User/WMS selects allocation:
Too big? Wasted resources.
Too small? Job fails, retry.

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

Facility handles scheduling (placement) but not the job sizing problem.

User

Facility



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.

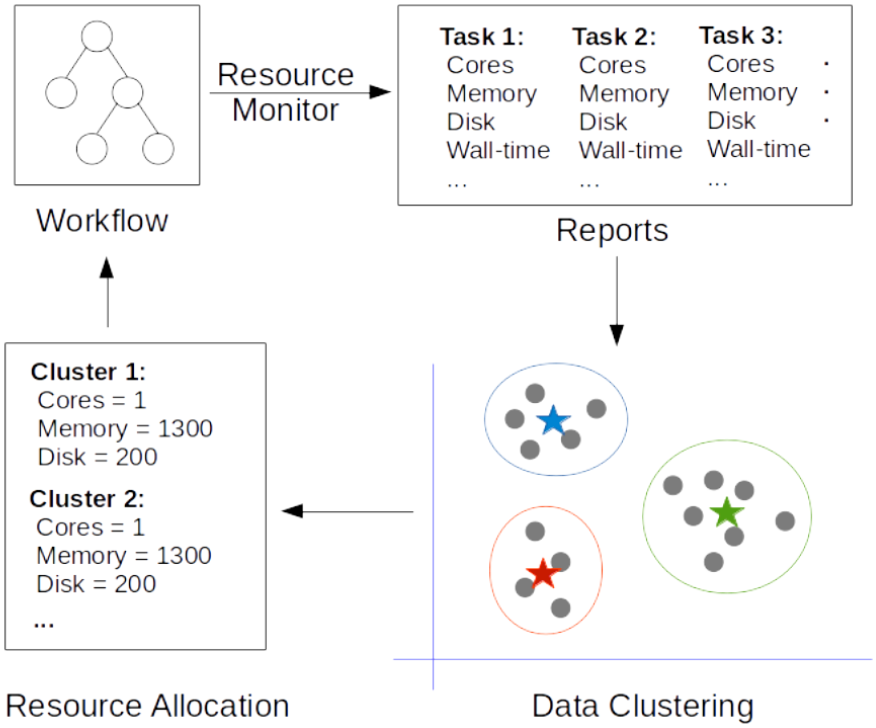
Dynamic Resource Allocation:

- 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.

Clustering Based Approach



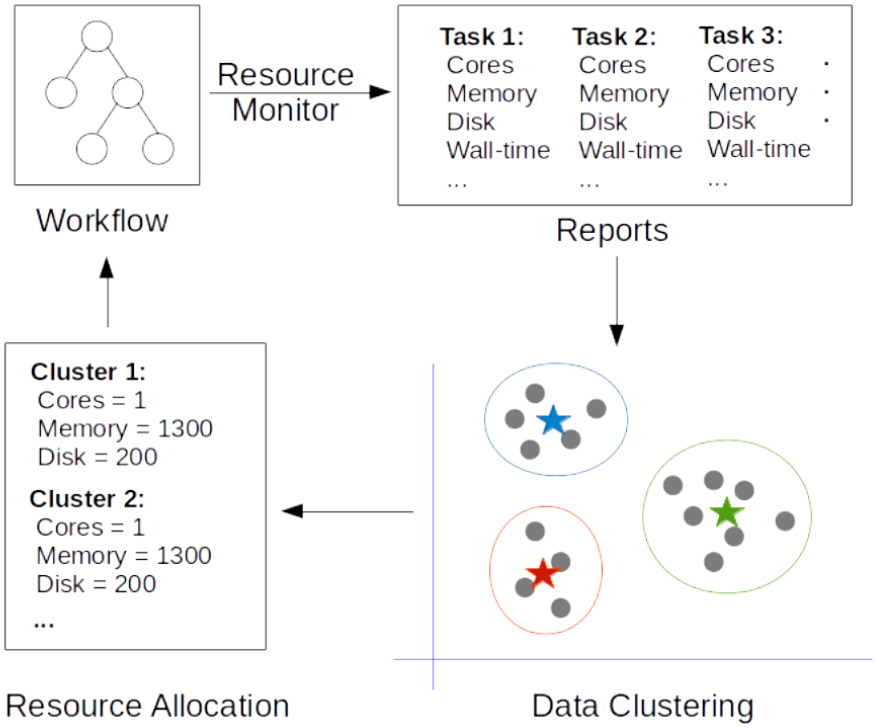
Our approach can be applied

When users need to:

1. Run the workflow multiple times.

2. Run a small batch of workflow before running the whole workflow.

Clustering Based Approach



A *resource monitor* tool is used to generate resource usage reports.

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

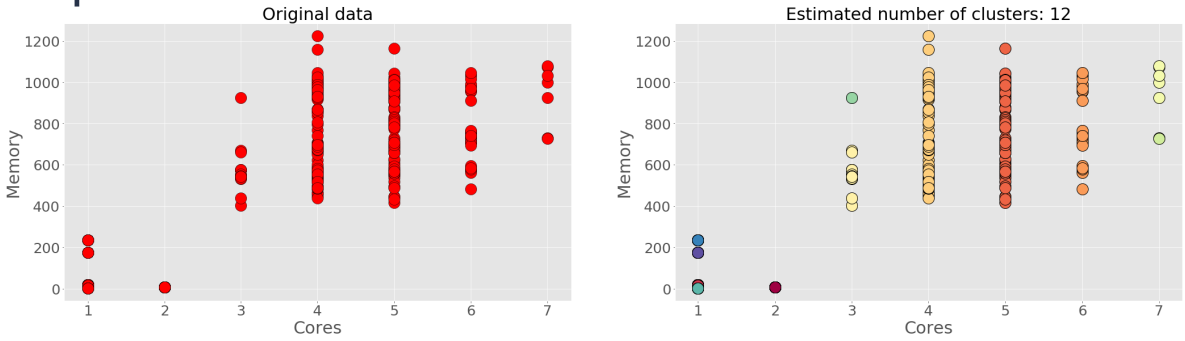
An improved resource allocation strategy is given by calculating the maximum resources of each cluster.

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.



(a) Before Clustering

(b) After Clustering

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 sample 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 inferent 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):

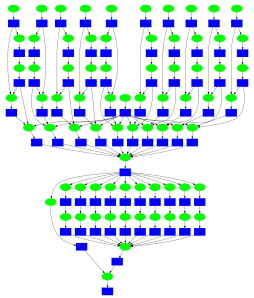
$$D(X,Y)=\left(\sum_{i=1}^{4} w\times\left|x_i-y_i\right|^{p}\right)^{(1/p)}$$

eps:

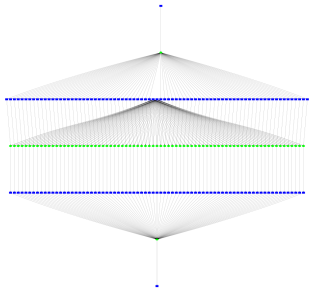
- The maximum distance between two samples for them to be considered as in the same cluster
- The value of *eps* should be modified in different workflows to achieve better performance.

Experiments on Scientific Workflows

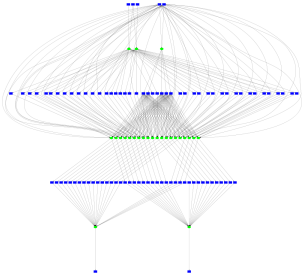
Lifemapper



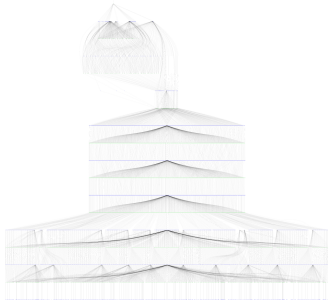
SHRIMP

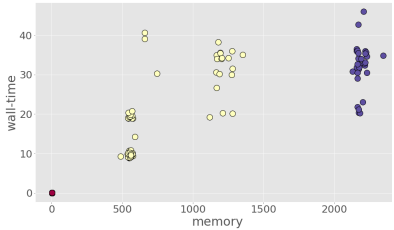


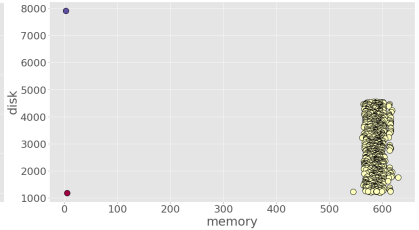
BWA

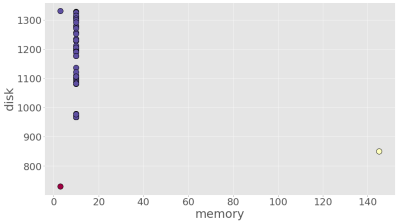


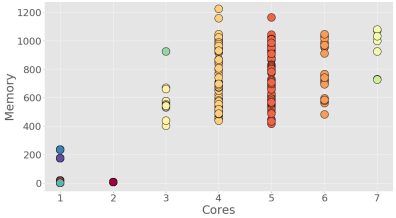
BWA-GATK





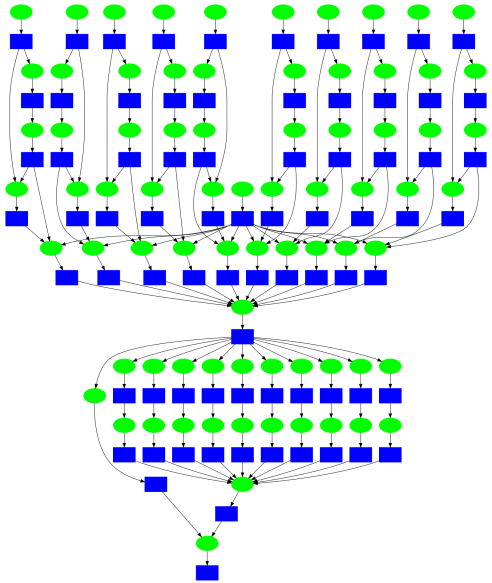


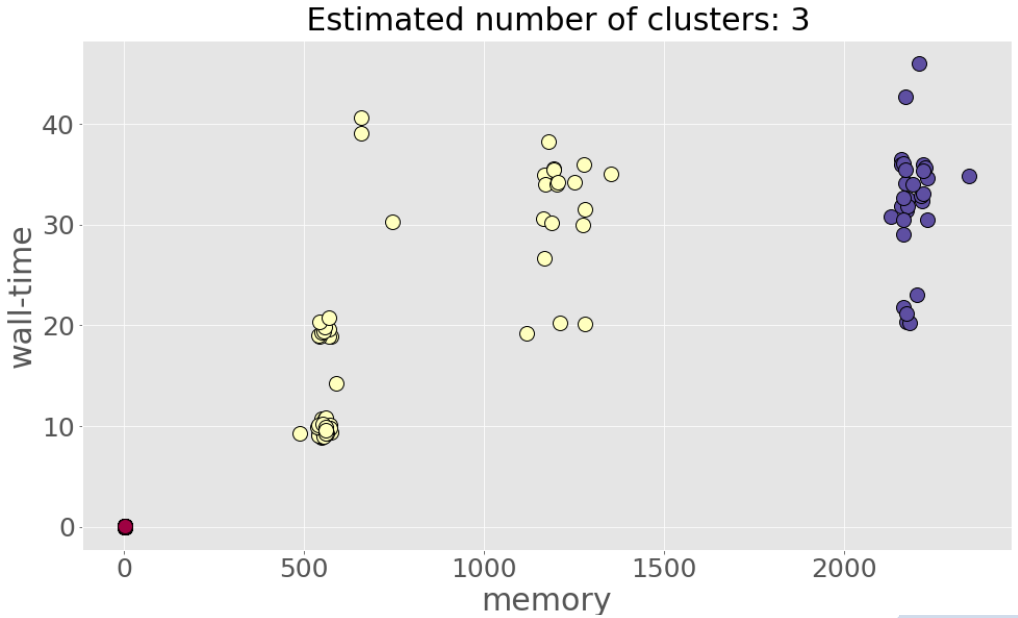




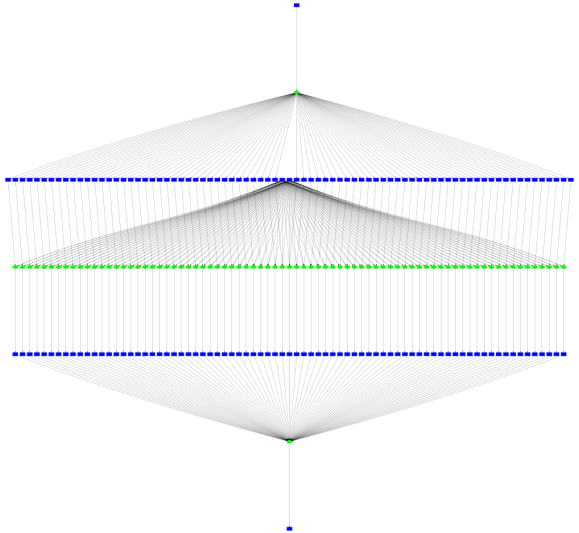
https://github.com/cooperative-computing-lab/makeflow-examples

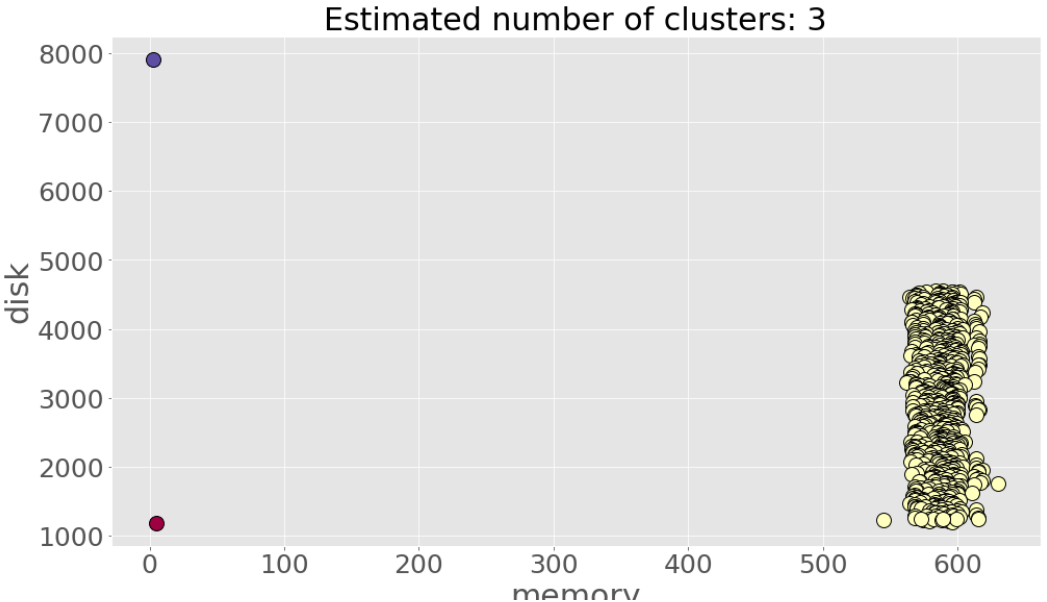
Lifemapper-Workflow



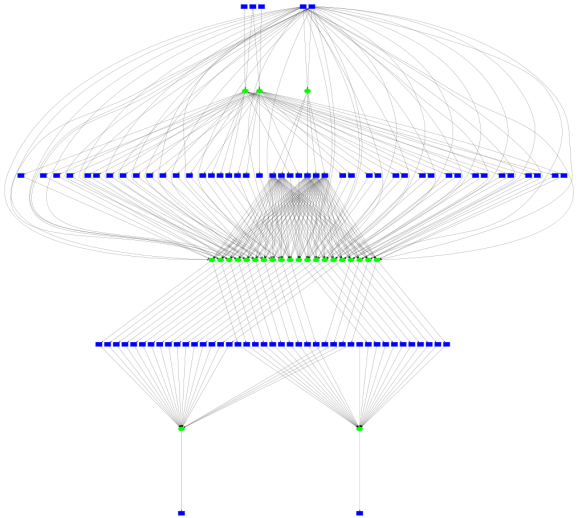


SHRIMP-Workflow

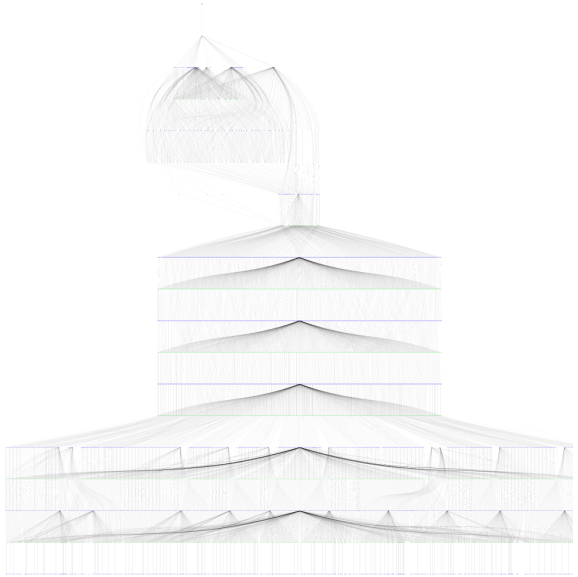


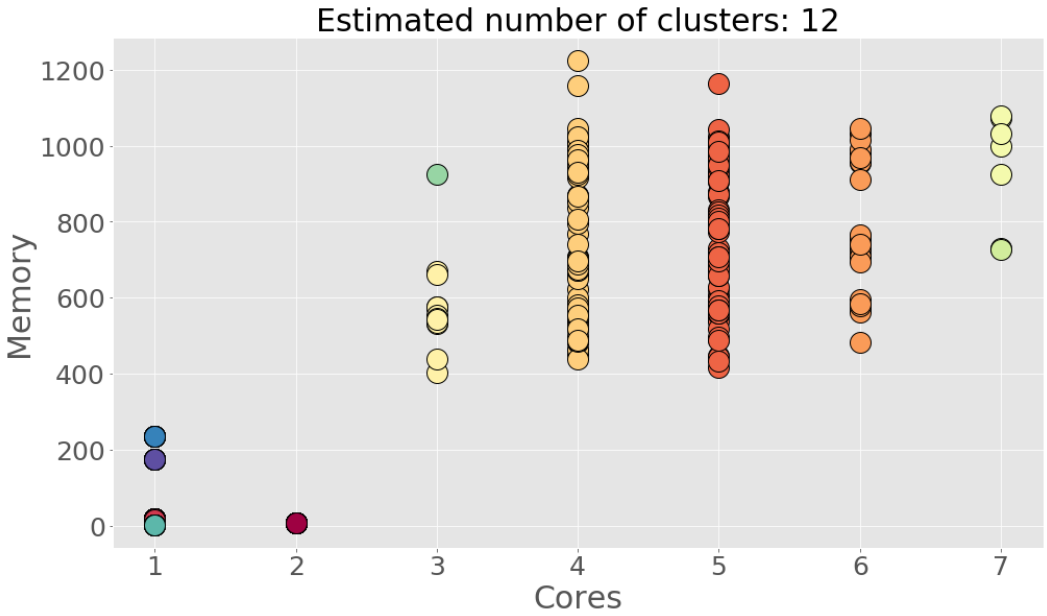


BWA-Workflow

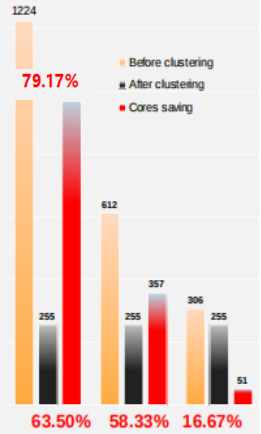


BWA-GATK-Workflow





Evaluation Results (Lifemapper)

Lifemapper	Job	Resource			Time (s)	Memory (GB.min)	Disk (GB.min)	Time saving	Memory saving (GB.min)	Disk saving (GB.min)	Cores saving
		cores	Memory (MB)	Disk (MB)							
1st	153	8	8000	50000	1394.23	539.23	3370.22				
2nd	153	4	8000	50000	542.49	406.41	2540.03	compare with 1st 84.83%	compare with 1st 72.34%	compare with 1st 96.68%	
3rd	153	2	8000	50000	245.87	293.22	1832.64				
After clustering	153	Cluster 1			211.50	149.13	122.05	compare with 2nd 61.01%	compare with 2nd 63.31%	compare with 2nd 95.59%	
		1	100	3000							
		Cluster 2						compare with3rd 13.98%	compare with 3rd 49.14%	compare with 3rd 93.89%	
		2	5000	30000							
		Cluster 3									
		2	1500	3000							

Evaluation Results

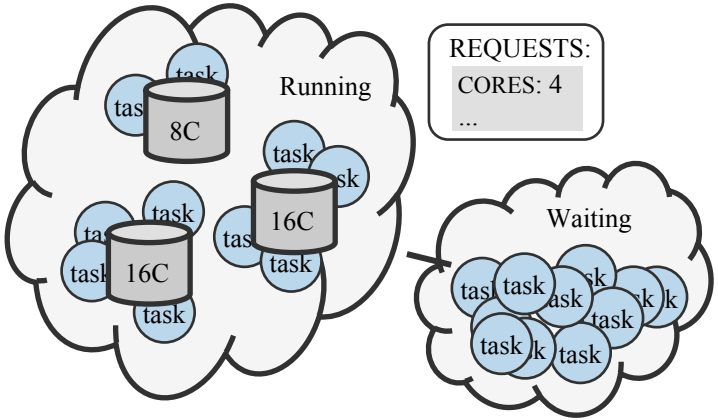
Workflow	Resource Saving		
	Cores	Memory (GB.min)	Disk (GB.min)
SHRIMP	50.00%	91.14%	92.10%
BWA	50.00%	95.44%	51.82%
BWA-GATK	59.10%	94.98%	94.81%

- 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!

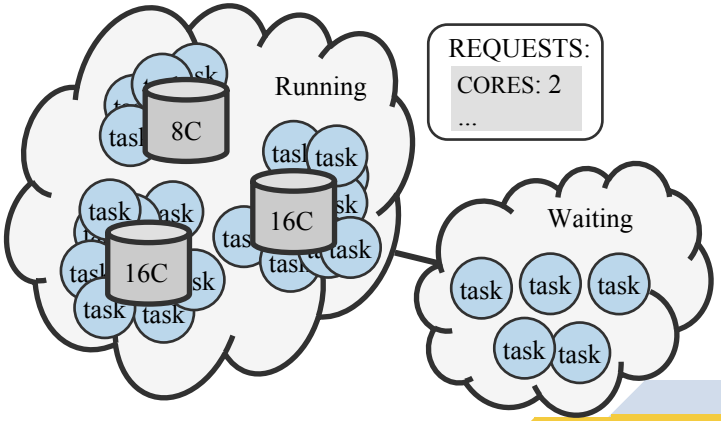
Larger resource requesting:

- harder to schedule within the cluster
- longer to wait in work queue



Smaller resource requesting:

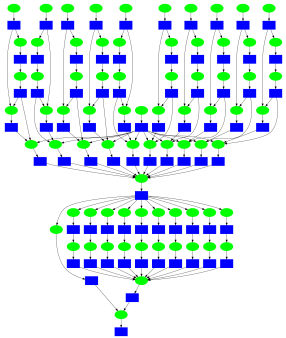
- faster scheduling times
- faster to fullfill



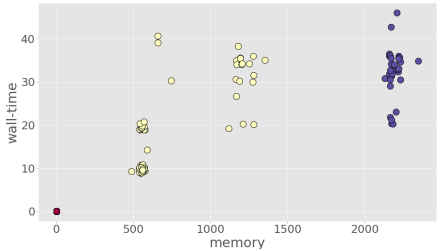
Takeaways and Caveats

- Clustering-based resource allocation strategy significantly reduces resouce 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?)

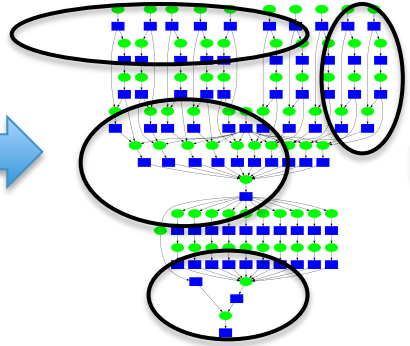
Clustering: The Chicken or the Egg?



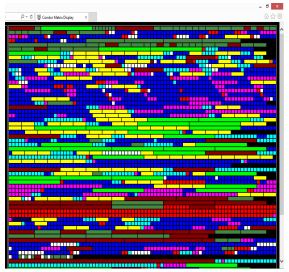
Unlabeled Workflow



Measure to Find Resources



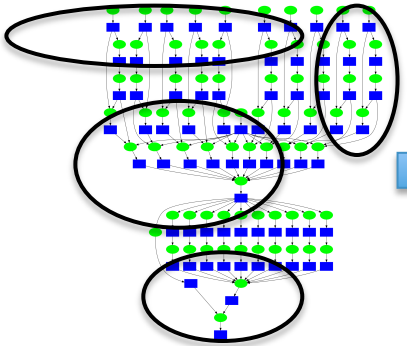
Cluster Resources to find Groupings



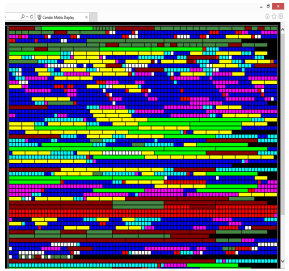
Execute Workflow

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.

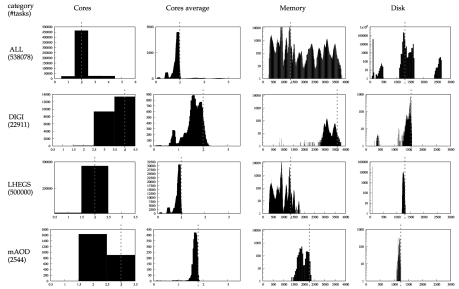
Clustering: The Chicken or the Egg?



User Provides Clusters
(not resources)



Execute Workflow

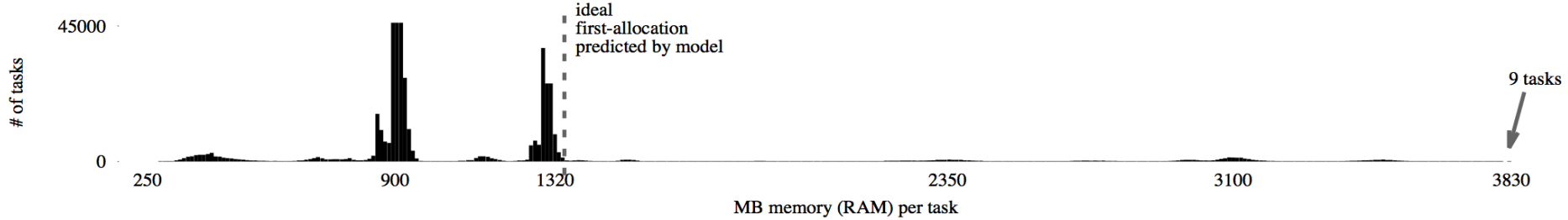


Measure Resources of Clusters

Improve Predictions

Benjamin Tovar, Rafael Ferreira da Silva, Gideon Juve, Ewa Deelman, William Allcock, Douglas Thain, and Miron Livny, **A Job Sizing Strategy for High-Throughput Scientific Workflows,** *IEEE Transactions on Parallel and Distributed Systems*, **29**(2), pages 240-253, February, 2018. DOI: 10.1109/TPDS.2017.2762310

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.

$$
\begin{aligned}
\mathrm{E}[\mathrm{waste}(\mathrm{r},\tau,a_1)] = \int_0^\infty \Bigg(& \underbrace{\int_0^{a_1} (a_1 - \mathrm{r})\tau p(\mathrm{r},\tau)\mathrm{d}\mathrm{r}}_{\text{first-allocation succeds}} \\
& + \underbrace{\int_{a_1}^{a_m} ((a_m + a_1 - \mathrm{r})\tau p(\mathrm{r},\tau)\mathrm{d}\mathrm{r}}_{\text{final allocation succeds}} \Bigg) \mathrm{d}\tau \\
= & a_1 \underbrace{\int_{a_1}^{a_m} \int_0^\infty \tau p(\mathrm{r},\tau)\mathrm{d}\tau\mathrm{d}\mathrm{r}}_{\text{mean wall-time for all tasks}} \\
& + a_m \int_{a_1}^{a_m} \underbrace{\int_0^\infty \tau p(\tau|\mathrm{r})\mathrm{d}\tau}_{\text{mean wall-time taks w. peak r}} \quad p(\mathrm{r})\mathrm{d}\mathrm{r} \\
& - \underbrace{\int_0^\infty \int_0^\infty \mathrm{r}\tau p(\mathrm{r},\tau)\mathrm{d}\tau\mathrm{d}\mathrm{r}}_{\text{used resources}},
\end{aligned}
$$

Big Picture Questions

- 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.

http://ccl.cse.nd.edu

ccl.cse.nd.edu

The Cooperative Computing Lab

Software | Download | Manuals | Papers

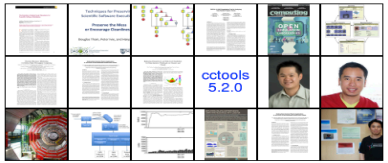
Take the ACIC 2015 Tutorial on Makeflow and Work Queue

About the CCL

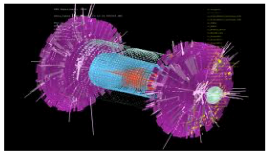
We design software that enables our collaborators to easily harness large scale distributed systems such as clusters, clouds, and grids. We perform fundamental computer science research in that enables new discoveries through computing in fields such as physics, chemistry, bioinformatics, biometrics,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

CCL News and Blog

- Global Filesystems Paper in IEEE CiSE *(09 Nov 2015)*
- Preservation Talk at iPres 2015 *(03 Nov 2015)*
- CMS Case Study Paper at CHEP *(20 Oct 2015)*
- OpenMalaria Preservation with Umbrella *(19 Oct 2015)*
- DAGVz Paper at Visual Performance Analysis Workshop *(13 Oct 2015)*
- Virtual Wind Tunnel in IEEE CiSE *(09 Sep 2015)*
- Three Papers at IEEE Cluster in Chicago *(07 Sep 2015)*
- CCTools 5.2.0 released *(19 Aug 2015)*
- Recent CCL Grads 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 CernVM Filesystem (CVMFS), a network filesystem tailored to providing world-wide access to software installations. By using Parrot, CVMFS, and additional components integrated by the Any Data, Anytime, Anywhere 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 CVMFS 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 15 sites with seamless access to 400 gigabytes of software in the Wisconsin CVMFS repository.

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

@ProfThain

Twitter, Inc.

Search Twitter | Have an account? Log in

TWEETS 28 | FOLLOWING 52 | FOLLOWERS 35 | LIKES 8

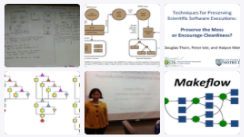
Follow

Douglas Thain
@ProfThain

Distributed computing for big data problems in science and engineering.

Notre Dame, IN
nd.edu/~dthain

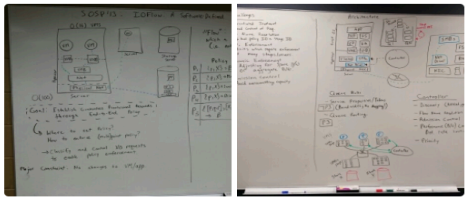
13 Photos and videos



Tweets | Tweets & replies | Photos & videos

Douglas Thain @ProfThain · Nov 10

My grad students now summarize research papers by preparing a whiteboard in advance. Much better than a slide deck!



New to Twitter?