High Performance Computing How-To

Joseph Paul Cohen

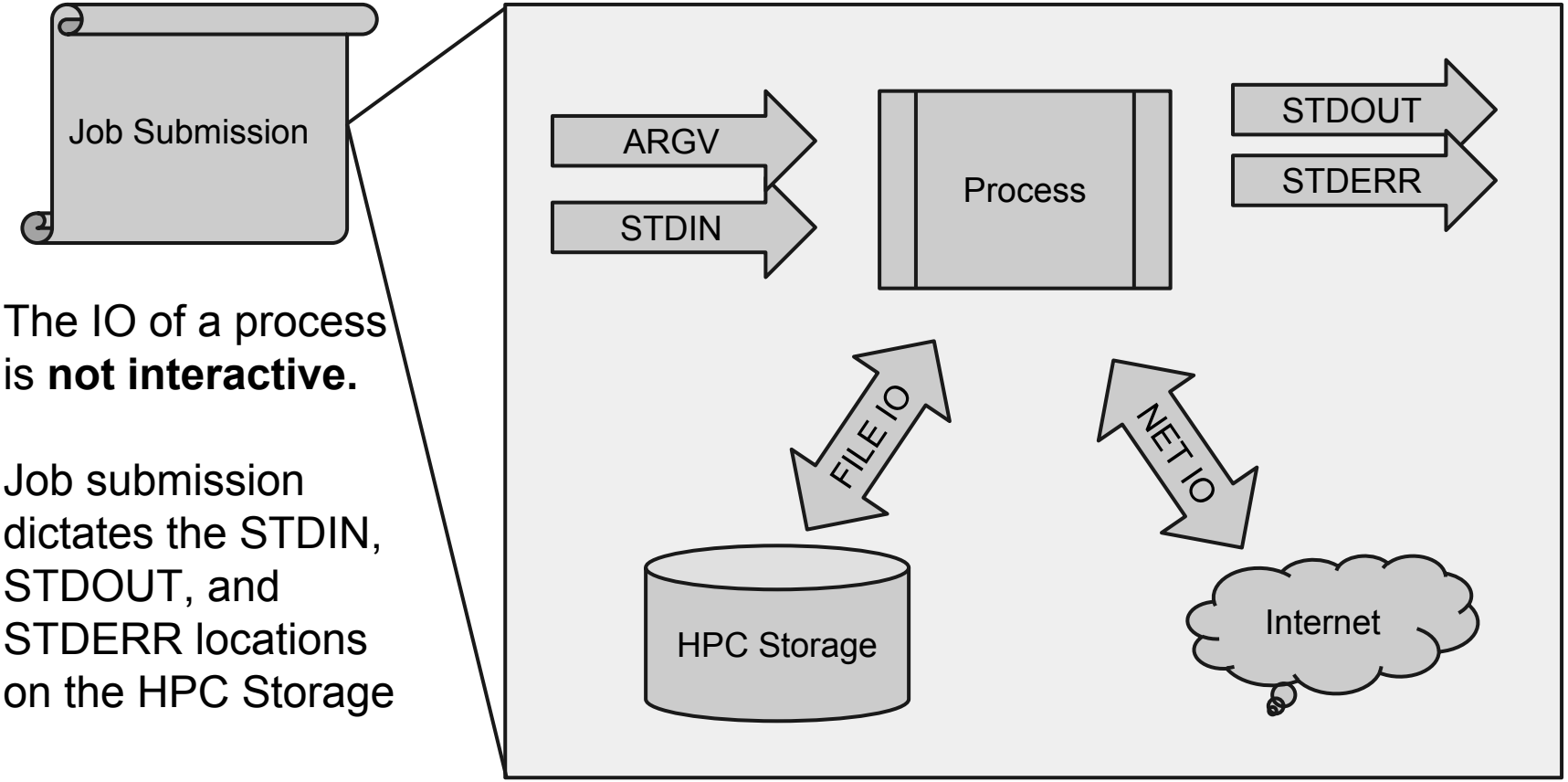
Abstract

This talk discusses how HPC is used and how it is different from typical interactive programs. I discuss job descriptions and scheduling. It also includes two entry level hands on examples. One, in Python, simple divides up work and the other, in Java, uses many cores at once to compute even faster.

Do you really need HPC? What are you trying to do?

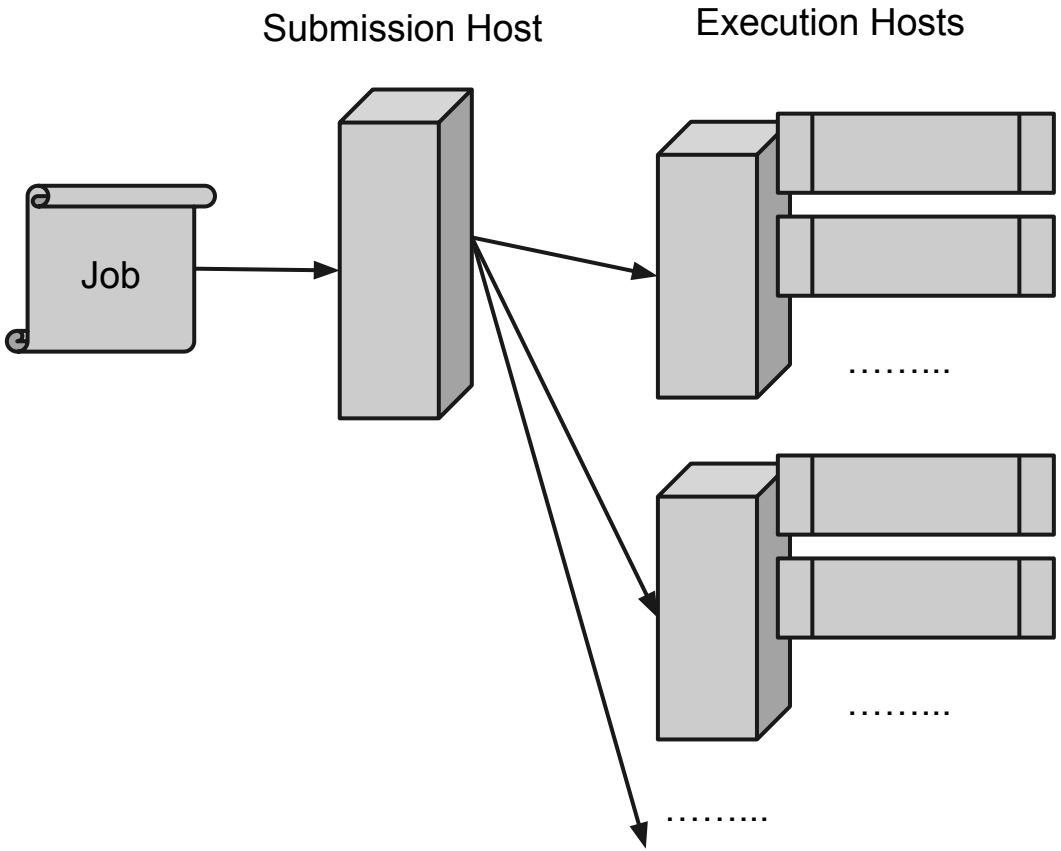
1. Analyse data?
    a. Data won't fit in memory? Does it need to?
    b. Can process locally but it's slow?
2. Analyse an Algorithm?
    a. Need to vary parameters?
3. Visualize data?
    a. Need to process the data to plot it?

Input/Output Overview



The IO of a process is **not interactive.**

Job submission dictates the STDIN, STDOUT, and STDERR locations on the HPC Storage

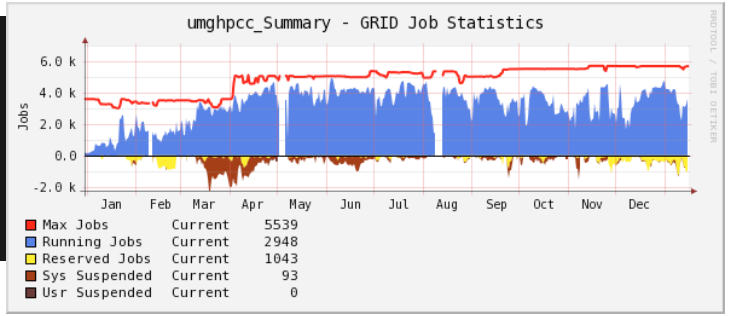


Each job runs on one core (or many) of a machine in the cluster.

You are responsible for keeping your process within the memory and cpu limits you specify.

Grid Overview

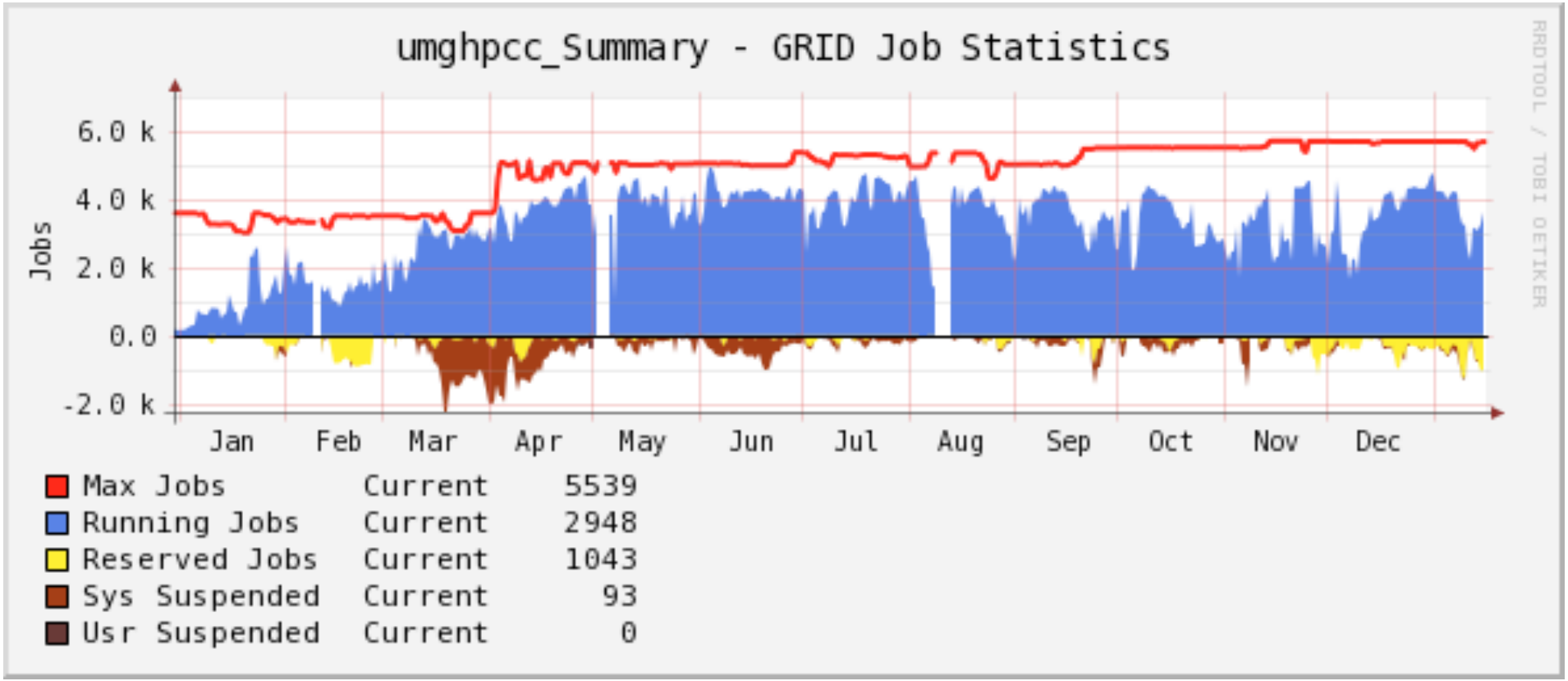
Job Scheduling



Jobs are encapsulated so they run modularly.

A queue can be filled with 1000's of jobs that take 10 hours each running only 30 at a time.

A queue can be filled with 1000's of jobs that take 20 minutes running all at once.



MGHPCC Cacti server statistics

Process Limits

Memory Default: 1G per core

CPU Default: 1 core

- As you request more CPUs, memory request will also go up.
- High limits can slow down scheduling. Free machines may have low specs. Don't wait for no reason!

System Differences

- Shared disk storage vs independent storage
- Job schedulers (bsub,qsub,condor_q)
- Max size of storage (maybe scratch space)

Varying Parameters

First Challenge

GET THE CODE

git clone https://github.com/ieee8023/hpc-demo

In folder: fibonacci

fib.py

```
import sys

def F(n):
    if n == 0: return 0
    elif n == 1: return 1
    else: return F(n-1)+F(n-2)

i = sys.argv[1]

print "#," + i + "," + str(F(int(i)))
```

Toy Problem (fibonacci sequence)

We want to evaluate this code from 1-100

How to split?

runJobs.sh

```
for i in `seq 1 40`;
do
    python fib.py $i
done
```

seq examples:

```
$ seq 1 3
1
2
3

$ seq 5 10 30
5
15
25
```

Running from the command line without cluster

Lets throw computers at it!?

BSUB Job Submission File

```
#BSUB -q short  # which queue (long or short)

#BSUB -n 1 # to request a number of cores

#BSUB -R rusage[mem=2000] # to specify the amount of memory required per slot, default is 1G

#BSUB -W 4:00 # how much Wall Clock (time) this job needs in Hours:Seconds, default is 60 minutes

……..
```

Sample BSUB script (MGHPCC)

BSUB Job Submission File

```
#BSUB -J demo[1] #name and number of copies of this job to run. Here 1 time. demo[5] would be 5 times.

#Set where logs go %J is job id and %I is instance of it
#BSUB -o "logs/%J.%I.out"
#BSUB -e "logs/%J.%I.err"

# execute program with argument
python fib.py 5
```

Sample BSUB script (MGHPCC)

Running jobs

BSUB wants the job script to be piped in STDIN

$bsub < job.bsub

This is done from a submission host. You should not run jobs on the submission host.

run.bsub

```
bsub << EndOfMessage
#BSUB -q short
….. add BSUB args
#BSUB -e "logs/%J.%I.err"

python fib.py $1   ← here we use the first CLI arg

EndOfMessage
```

Sample BSUB script (MGHPCC)

runJobs.sh

```
for i in `seq 1 40`;
do
    sh run.bsub $i
done
```

Modify runJobs.sh to run on cluster

```
$ sh runJobs.sh
Job <2413367> is submitted to queue <short>.
Job <2413368> is submitted to queue <short>.
Job <2413369> is submitted to queue <short>.
Job <2413370> is submitted to queue <short>.
Job <2413371> is submitted to queue <short>.
.............
```

Run script to start jobs

Is your job running?

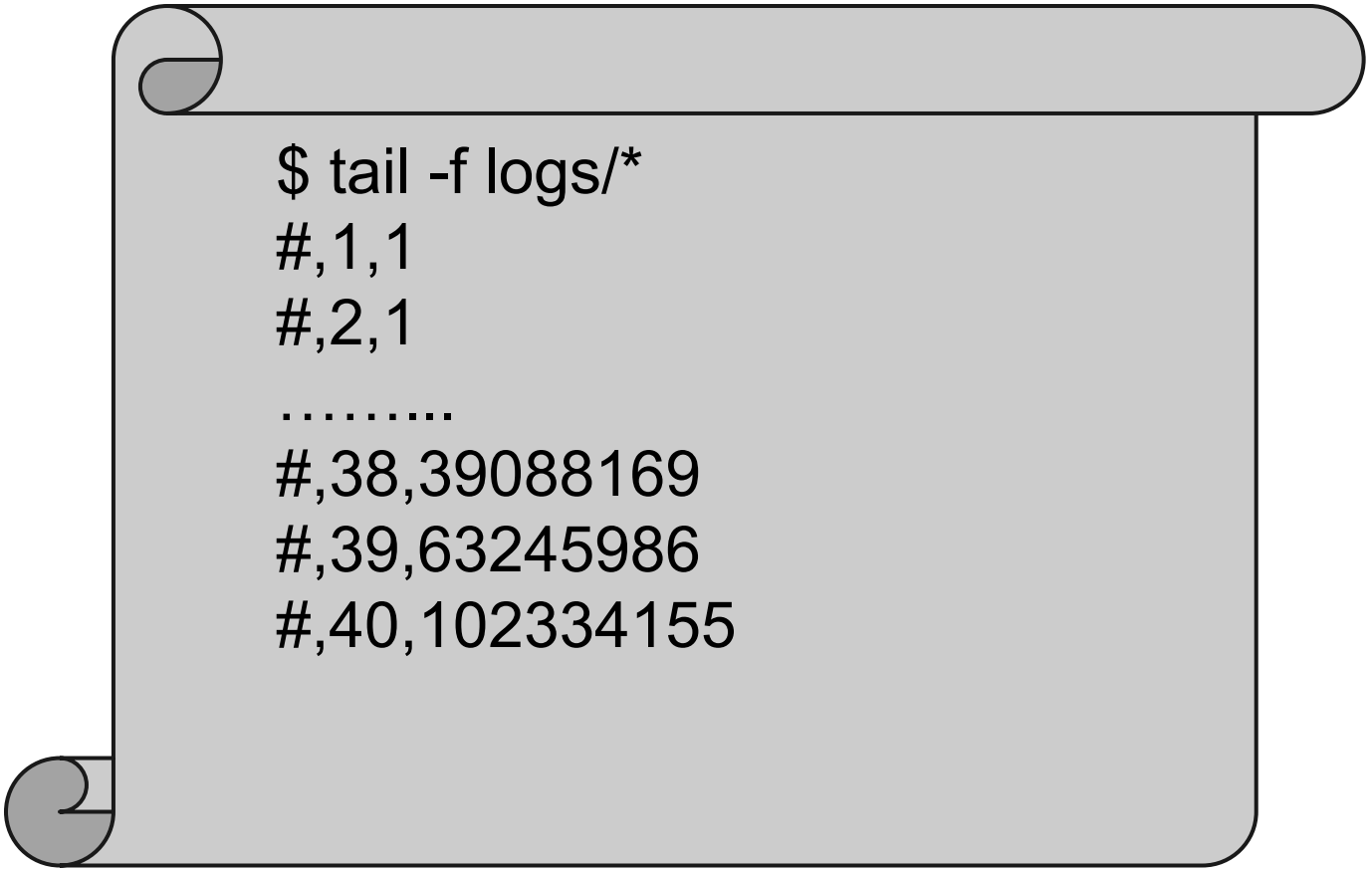
```
[jc93b@ghpcc06 demo]$ bjobs
JOBID   USER    STAT  QUEUE      FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
2413343 jc93b   RUN   short      ghpcc06     2*c23b07    demo38[1]  Feb 13 19:14
2413344 jc93b   RUN   short      ghpcc06     2*c23b07    demo39[1]  Feb 13 19:14
2413345 jc93b   RUN   short      ghpcc06     2*c23b07    demo40[1]  Feb 13 19:14
```

Do you want to stop it?

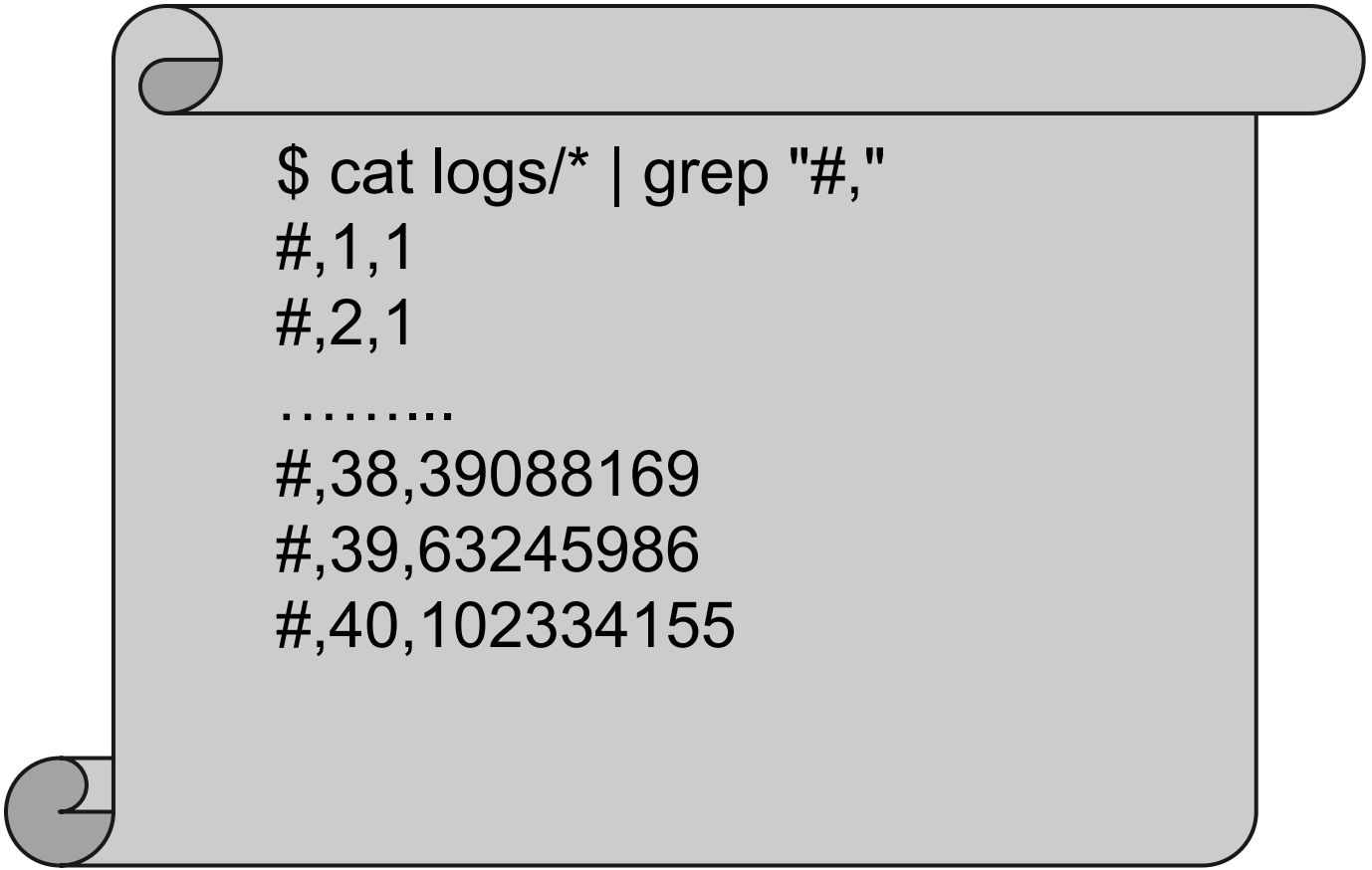
```
# kill job with id 2413343
[jc93b@ghpcc06 demo]$ bkill 2413343

# or just kill all the jobs
[jc93b@ghpcc06 demo]$ bkill 0
```

```
$ tail -f logs/2413379.1.*
==> logs/2413379.1.out <==
Sun Feb 15 14:06:08 EST 2015 start

==> logs/2413406.1.out <==
Lets calc!

==> logs/2413401.1.out <==
Done
```

Follow job progress



Follow all progress



Check Results

View results

SCP them back to yourself

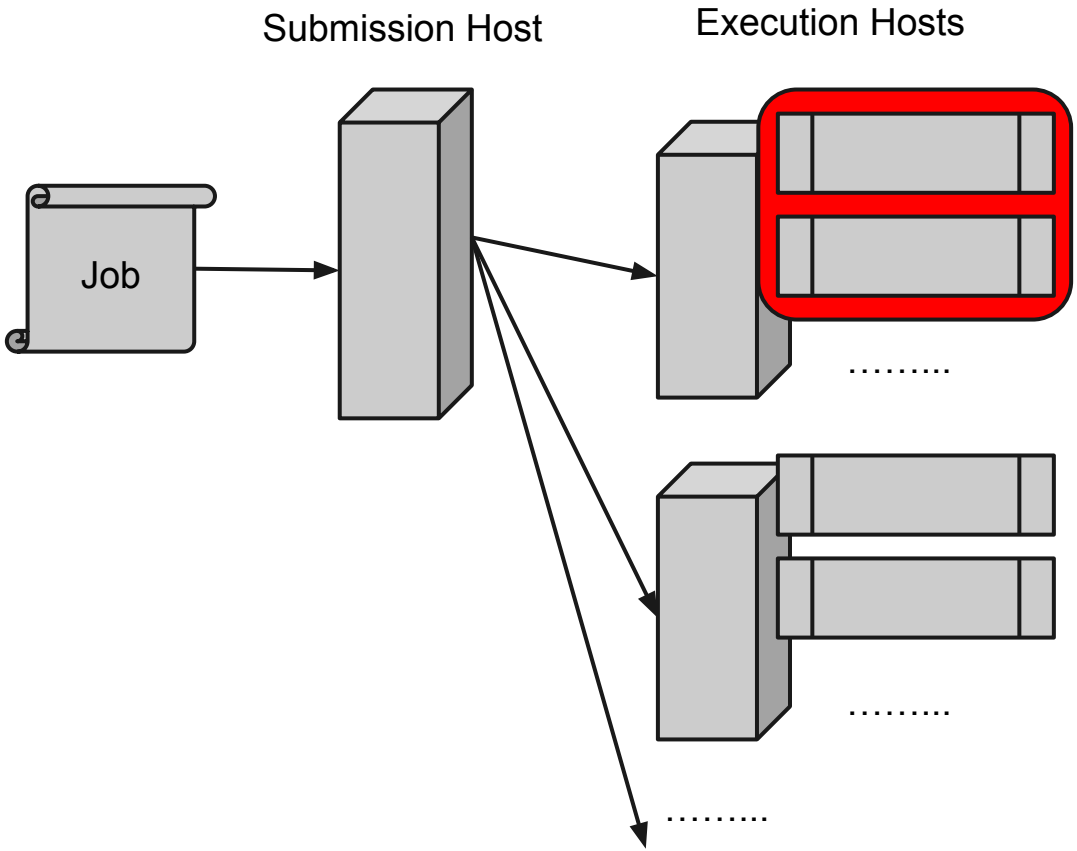
```
$cat logs/* | grep "#," > results.csv
$scp results.csv ieee8023@argus.cs.umb.edu:demo

$scp jc93b@ghpcc06.umassrc.org:results.csv .
```

Certificate Login

```
$ssh-keygen
$ssh-copy-id jc93b@ghpcc06.umassrc.org
$ssh -i id_ghpcc jc93b@ghpcc06.umassrc.org
```

Certificates allow quick login. Easy to share and revoke.

```
laptop$ cat id_ghpcc
-----BEGIN RSA PRIVATE KEY-----
YXNkYXNkZmFzZGZhc2RmYXNkZmFzZGZhc2RrZmpibmFza2R
mamJuYXNrZGpmYm53bGllamZoYglxbGl3ZWhmYmFsc2RoZm
….
-----END RSA PRIVATE KEY-----

ghpcc$ cat .ssh/authorized_keys
ssh-rsa AAAAB3NzaC1y…….
```



We can utilize multiple cores on a host at once.

This way we can share memory between threads.

Multiple Threads Sharing Memory

GET THE CODE

git clone https://github.com/ieee8023/hpc-demo

In folder: weka-research-computing

Add Java

In your ~/.bash_profile add this line:

module load jdk/1.7.0_25

Browse other modules with:

module avail

Evaluate Support Vector Machines

Using Weka, sharing data in memory

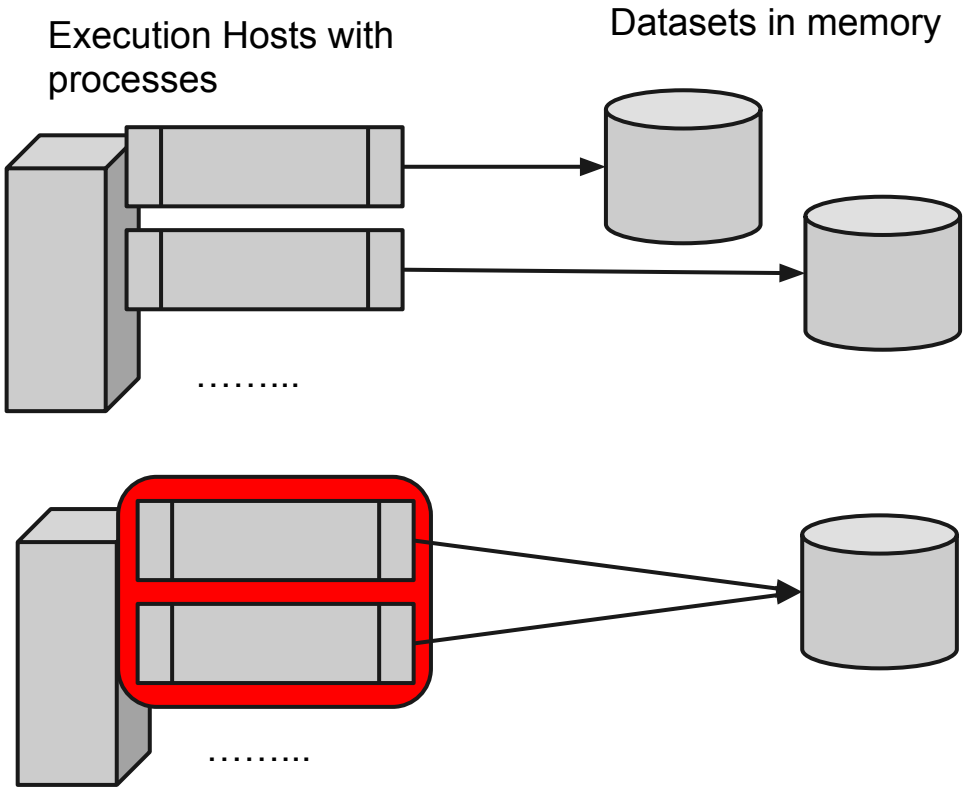
```
// Get an Instances object
Instances data = new Instances(....);

//Create an eval object and do cross-validation
Evaluation eval = new Evaluation(data);
eval.crossValidateModel(classifier, data, 5, new Random());

//calculate the F1-Score
double f1 = eval.weightedFMeasure();
```

Using Weka

Experiment Class

```
Experiment implements Runnable {

Experiment(  String label,
             String dataset,
             Instances instances,
             Classifier classifier,
             ThreadPoolExecutor es)

…..
```

Runnable Experiment object will allow to multithread



If loading the data into memory is costly then don't do it more than you have to.

Sharing Instances in memory

```
for (int i : new int[]{1,2,3,4,5})
	for(Instances instances : instancess){
		Experiment exp = new Experiment(
			"Test1",
			instances.relationName(),
			instances,
			new LibSVM(),
			es);

			// run exp directly with: exp.run();
			// run it with an executor with: es.execute(exp);
	}
```

Running multiple Experiments

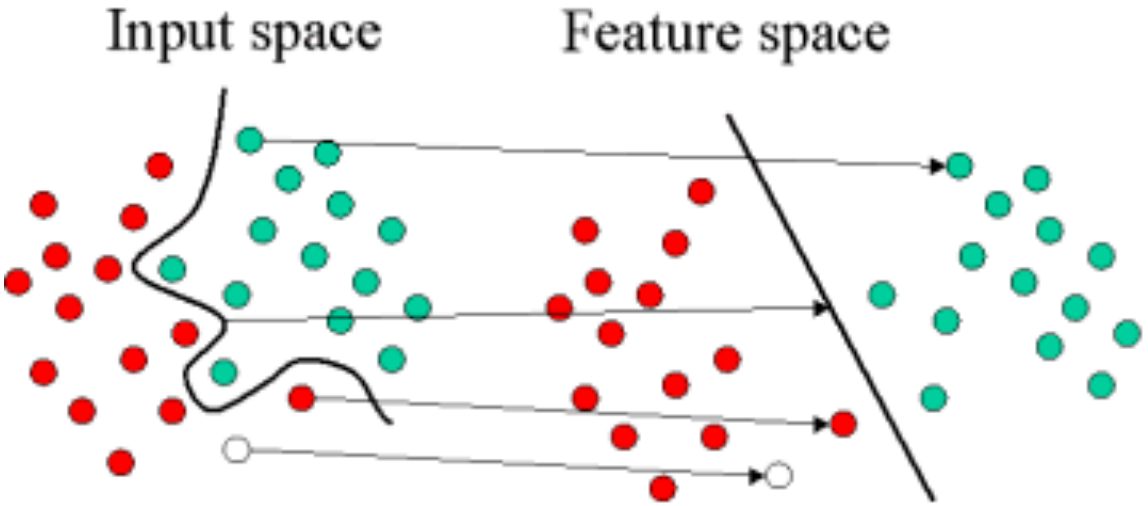
```
// make threadpool to multithread with limit (cores)
ThreadPoolExecutor es = (ThreadPoolExecutor) Executors.
newFixedThreadPool(cores);

//create Experiment and execute it right away
es.execute(new Experiment(.....));

//wait forever for all Experiments to finish
es.shutdown();
es.awaitTermination(9999, TimeUnit.DAYS);
```

run.bsub

```
#BSUB -q short  # which queue
#BSUB -n 5 # to request a number of cores
…

# we call run.sh with sh
sh run.sh $1

==========================================
run.sh:
java -Xmx4g -cp `sh getclasspath.sh`:classes joe.Experiment $@
```

Running with bsub

http://www.statsoft.com/textbook/graphics/SVMIntro3.gif



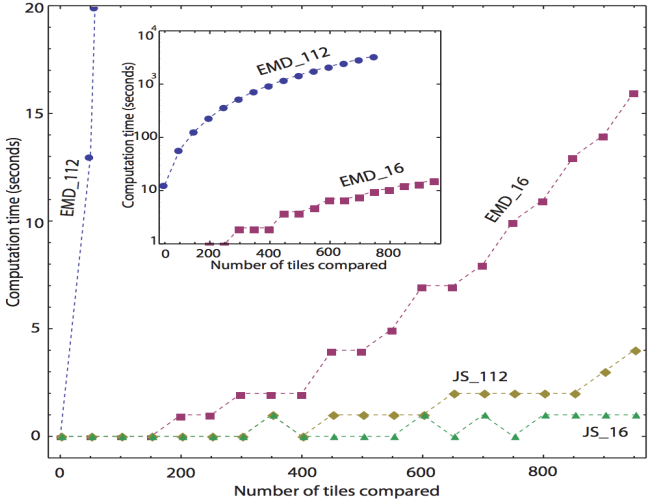
What results do you get for an SVM?

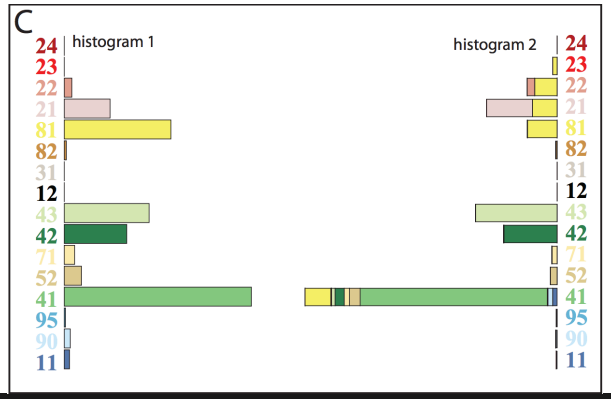
Challenges

- Add another dataset
- Vary the cross validation from 2-10
  - Plot the difference
- Compare different classifiers
  - NaiveBayes, J48, AdaBoostM1,RandomForest

Usage Examples

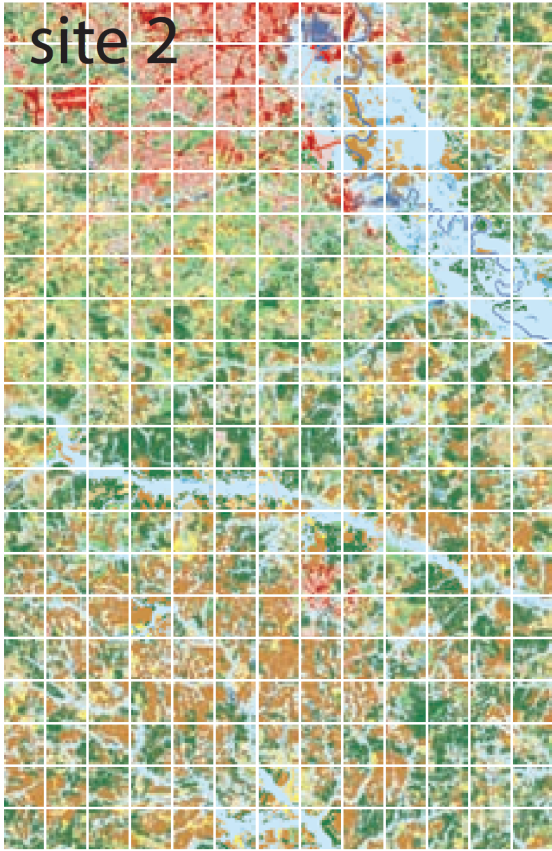
From my work



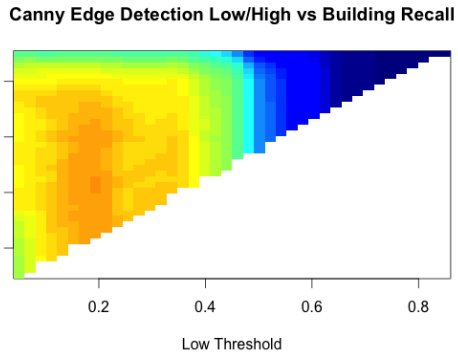


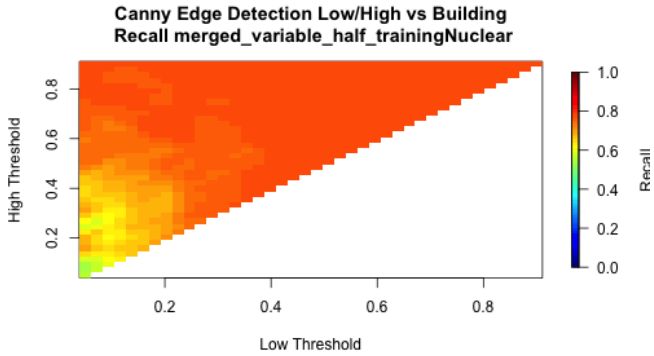
- open water(11)
- ice/snow(12)
- developed open space(21)
- developed low intensity(22)
- developed med. intensity(23)
- developed high intensity(24)
- barren land(31)
- deciduous forest(41)
- evergreen forest(42)
- mixed forest(43)
- shrub/scrub(52)
- grassland(71)
- pasture/hay(81)
- cultivated crops(82)
- woody wetlands(90)
- emergent wetlands(95)

For evaluation of a site a distance matrix consisting of all tiles is computed. To evaluate EMD_112 a grid must be used.



Evaluating National Land Cover (NLCD) Data





Finding optimal parameters for the entire pipeline is very expensive. ~4hr per set of parameters. To generate heatmaps must be done using a grid system.

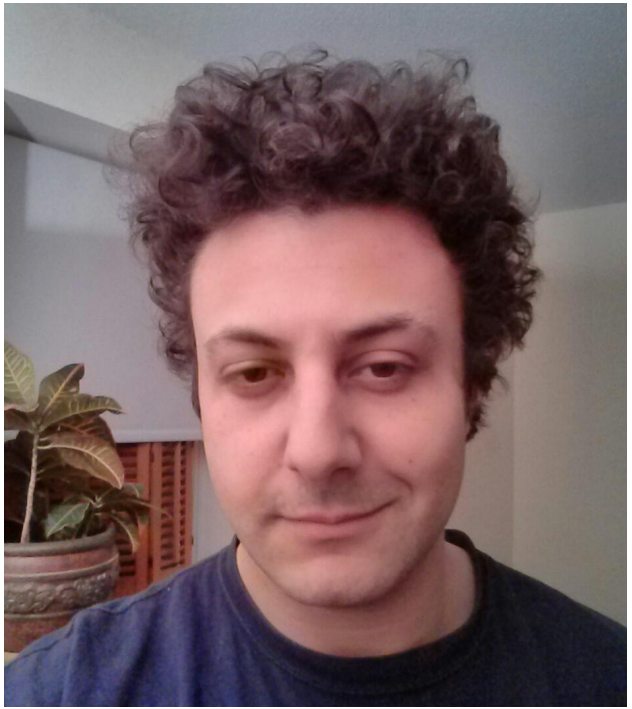
Evaluating Building Detection Code

Links

Wiki: http://wiki.umassrc.org/wiki/index.php/Main_Page

Request Access: https://ghpcc06.umassrc.org/hpc/index.php

Speaker



Joseph Paul Cohen

Email: joecohen@cs.umb.edu

National Science Foundation Graduate Fellow

Ph.D Candidate - Computer Science

University of Massachusetts Boston