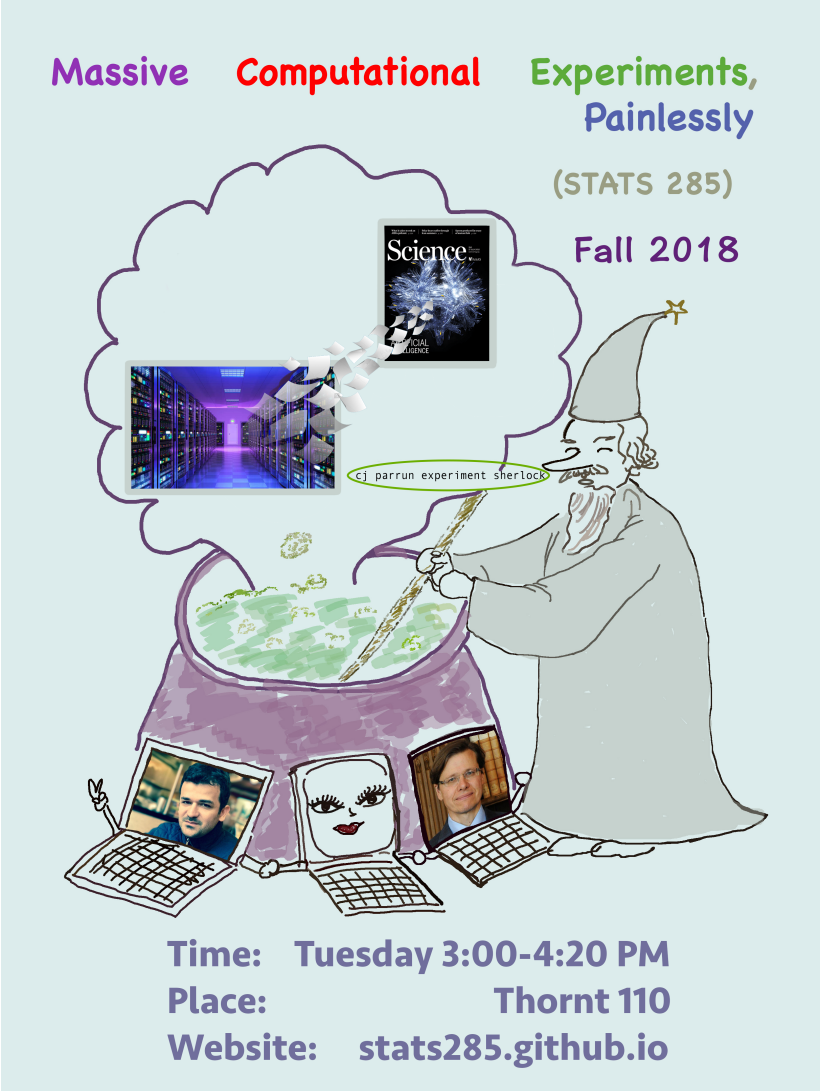
ClusterJob: An Open-source Experiment Management System for Data Science

H. Monajemi/DL. Donoho

Stats285, Stanford

Oct/09/2018

Stanford University

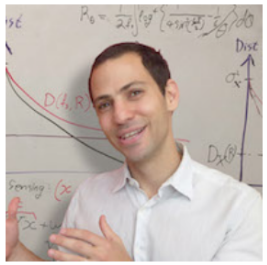


Disclaimer, 1: Google Images

This document contains images obtained by routine Google Images searches. Some of these images may perhaps be copyright. They are included here for educational noncommercial purposes and are considered to be covered by the doctrine of **Fair Use**. *In any event they are easily available from Google Images. It's not feasible to give full scholarly credit to the creators of these images. We hope they can be satisfied with the positive role they are playing in the educational process.*

Outline

Meet Hackathon Mentors

		
Vardan Papyan	Alon Kipnis	Yaniv Romano
		
Morteza Mardani	Pete Mohanty	Sara Hooshangi

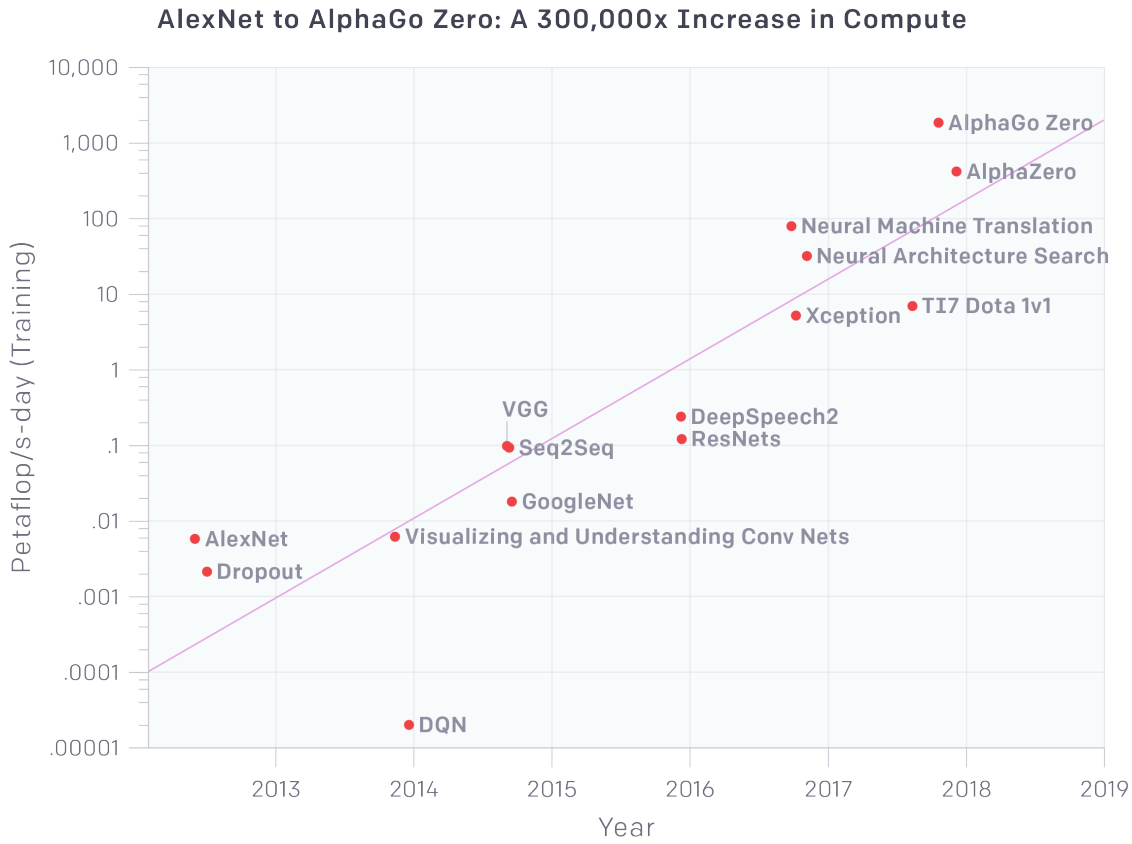
Time and Venue

The hackathon starts at 8:00 AM on Nov 17 and ends at midnight on Nov 18. The event takes place at Wallenberg Hall-Building 160.

Lecture01: Revolution is here!

- **massive** computational resources created in the past decade
- Publicly available at a reasonable cost (CPUs, GPUs, TPUs)
- Near-unlimited quantities on-demand (for a price)
- Expansion by factors of 1000's in immediate computing capacity when job is *trivially parallelizable*

Computing power spawns rapid scientific progress



Amount of available compute doubles every 3.5 month

($300K \uparrow$ since 2012)

source: OpenAI

(`https://blog.openai.com/ai-and-compute/`)

Embrace the change

- Powerful computers easily accessible everywhere,
- Cloud can make scientists very powerful
- Requires change in research process and individual habits:
  - **Psychological change** and rethinking of scientific values
  - **Pose** bold research **hypotheses** to settle computationally
  - **Design massive computing experiments**
  - **Adopt** an Experiment Management System (EMS)
  - **Raise money** to pay for cloud-based computing
  - *Push Button*
- We describe one EMS today: **ClusterJob (CJ)**
  - In daily use at Stanford
  - Developed by Yours Truly.

Lecture 2: Cluster Computing

- **cluster**: A collection of compute nodes (servers)
  - *node* (IP address)
    - *sockets* (typically 2-4)
      - *cores* (10 core/chip on Sherlock)
- **job** : a unit of work/execution comprised of **tasks**/**steps**
  - a job can use one or several cores (CPUs)
- **job scheduler**: application that controls execution of jobs
  - a.k.a. batch scheduling, cluster management system, workload automation, batch queue system (BQS)
  - examples: Portable Batch System(PBS), Sun Grid Engine (SGE), HTCondor, SLURM Workload Manager, Apache Mesos
- **job queue**: a data structure of jobs to run used by BQS

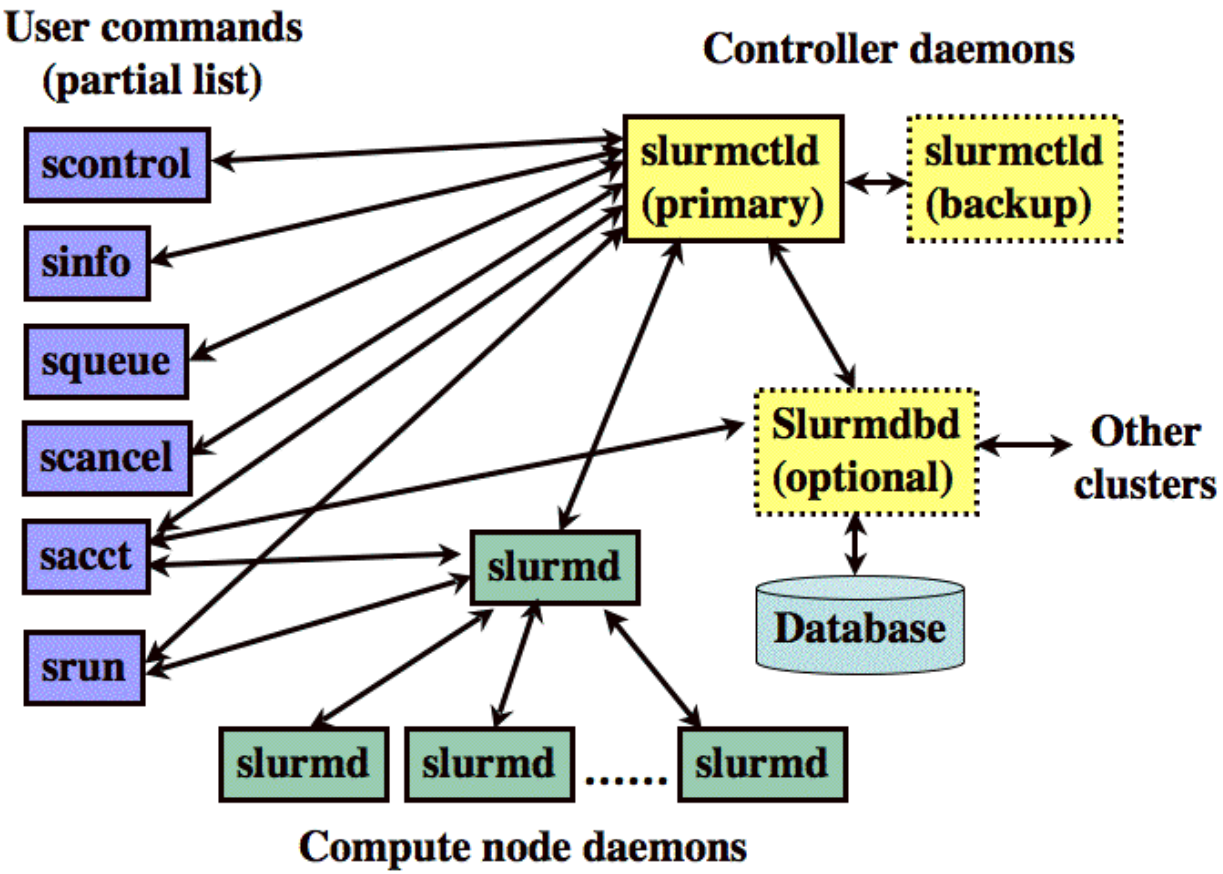
SLURM

- **S**imple **L**inux **U**tility **R**esource **M**anagement
- Used by many clusters today

Sunway-TaiHuLight Supercomputer (Wuxi,China), uses SLURM to manage $\approx 11M$ cores.

SLURM

- **S**imple **L**inux **U**tility **R**esource **M**anagement
- Used by many clusters today



SLURM architecture

Advice for using clusters

Bad habits:

- Repetitive interactive logging on to the cluster
- Manual copy of your code and data
- Manually using scheduler ($sbatch) each time

Good habits:

- Automating activities (or using EMS)
- Occasional logging on to the cluster

Outline

Stanford University

Data Science demands scaling

CTF

More Tweak $\longrightarrow$ More Win

Stanford University

Data Science demands scaling

CTF

More Tweak $\longrightarrow$ More Win

Data Science

More Experiments $\longrightarrow$ More Discovery

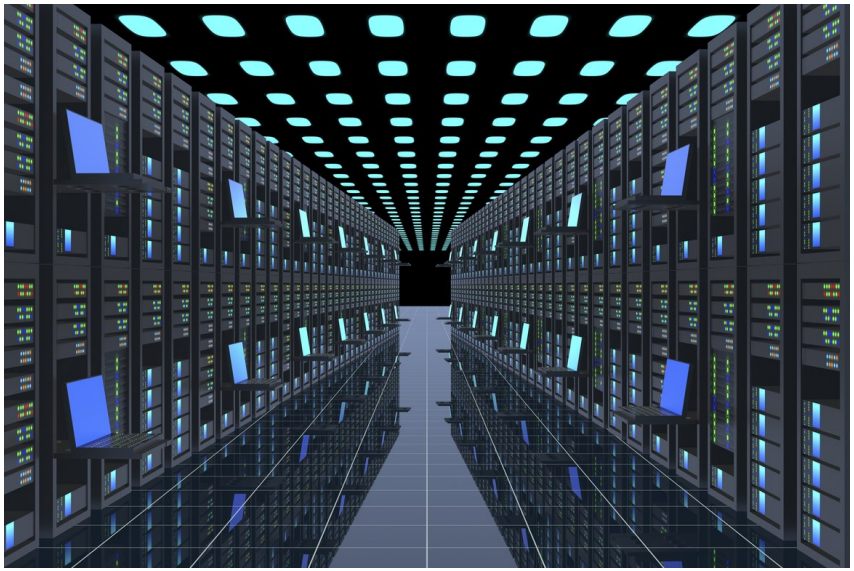
Tukey, FoDA 1961

Tukey *chose* automation in the 60's and 70's!

To critics of automation in 1961:

- proper automation $\rightarrow$ **more study** of the data
- automation of known procedure $\rightarrow$ **more new procedures**
- Much **easier to intercompare** automated procedures

Automation no longer a *choice*, but a *necessity*!



- Bigger datasets
- Need for powerful computers (to make sense of the data!)
- Complexity of the cloud
- Need for stacking and reproducibility

Outline

Stanford University

Learn schedulers and how to install software, etc!

Pros and Cons of HPC vs DIY/desktop

Pros-

- Your code/calculations are run on servers that are always on, networked and accessible from anywhere by anyone in your PI group (including off campus collaborators with basic SUNetIDs)
- High performance parallel file systems- fast i/o, 30TB of group Scratch storage on Sherlock
- Much more compute power, hundreds of CPUs, large memory servers up to 3TB of RAM
- Data sharing among research groups is easy, Globus for large data transfers
- Data in home directories are backed up (snapshotted) and replicated
- The job scheduler handles problems with hardware, hardware/nodes can fail but jobs do not, you launch your jobs and log off

Cons-

- Need to learn how to use a job scheduler and the Linux command line- aka "The Shell"
- Jobs go through a scheduler using the Fairshare algorithm since the system is shared by thousands of users; so you need to wait
- Sometimes you need to request and wait for software installs, you will not have the same permissions to change/modify the system as you do on your own laptop, desktop or cloud instance, however often users can install software themselves on Sherlock/Farmshare

Stanford University

Many clusters, many systems, many policies!

Changing working scripts, more work!

SUBMIT MULTIPLE JOBS AT ONCE WITH WRAP

The wrap feature of sbatch is very powerful. With it you can send any argumer commands run are inside the quotation marks after --wrap, for example, modu to create multi-line sbatch submissions based on a directory contents or any st matching to do this.

For example, lets say you want to do something to all fastq files in a directory. matching the string pattern *.fastq. Then we toss that as an argument to sbat

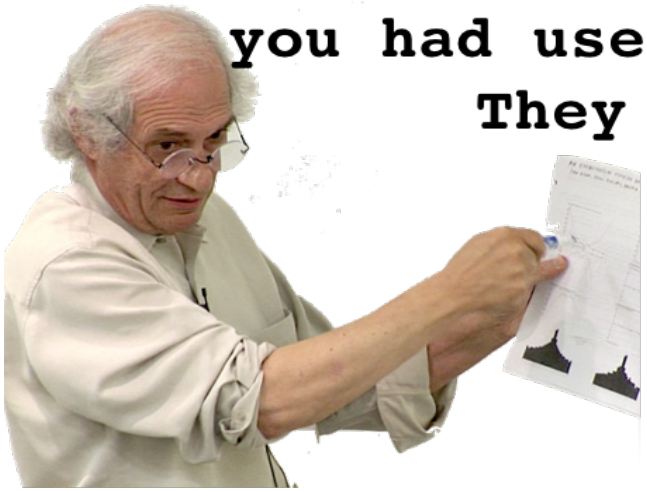
Create a shell script called wrap.sh:

```
#!/bin/sh
for FILE in *.fastq;
do  sbatch -p normal -t 10 --mem=200 --wrap="gzip ${FILE}"
sleep 1 # pause for 1 second so we don't overload the scheduler
done
```

My script runs just fine on my laptop. To run it in parallel on cluster, they say I have to change it and give parameters as command line args!!!

Manual tracking, irreproducibility and error!

Can you send me the code and parameters you had used to produce these results? They do not seem correct!



Oh, God! That was like 3 month ago. Since then, I ran a million more jobs. I can't seem to find it!

Outline

Stanford University

Rethink the way we do computational research: use EMS

What does a data science project involve?

Typically:

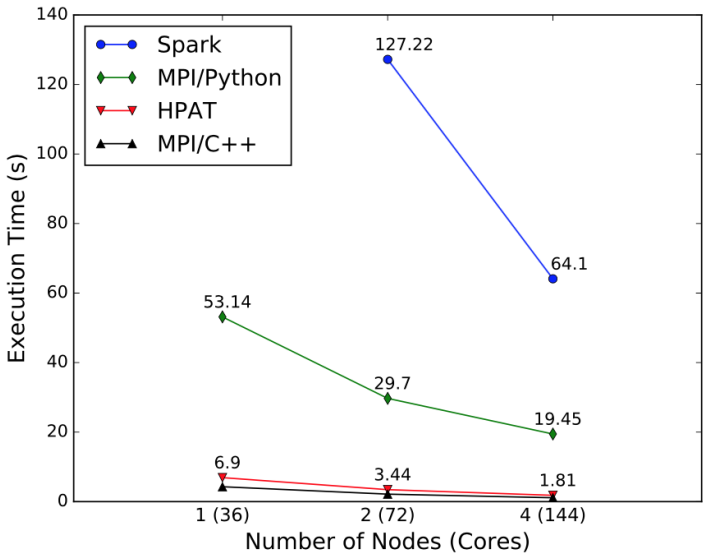
1. **Precise Specification** (define metric and parameters)
2. **Execution and management** of all the jobs
3. **Harvesting** of all the data generated by all the jobs
4. **Analysis** of the data
5. **Iterations** of steps (1-4)
6. **Reporting** of results.

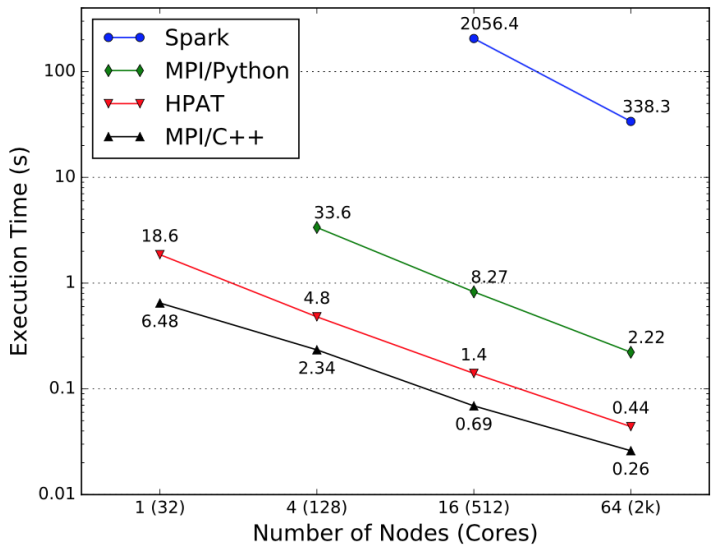
An Experiment Management System should automate and seamlessly integrate all these tasks

Desired features of an EMS

- **Simple**: the *right* level of abstraction!

Good example: Popularity of Spark though 59x slower than MPI!





(a) Scaling on Amazon AWS cloud (c4.8xlarge instances, 256M 10-feature samples, 20 iterations).

(b) Scaling on Cori supercomputer (1B 10-feature samples, 20 iterations). Please note the logarithmic scale.

Totoni et al. 2017, "A Case Against Tiny Tasks in Iterative Analytics"

Desired features of an EMS

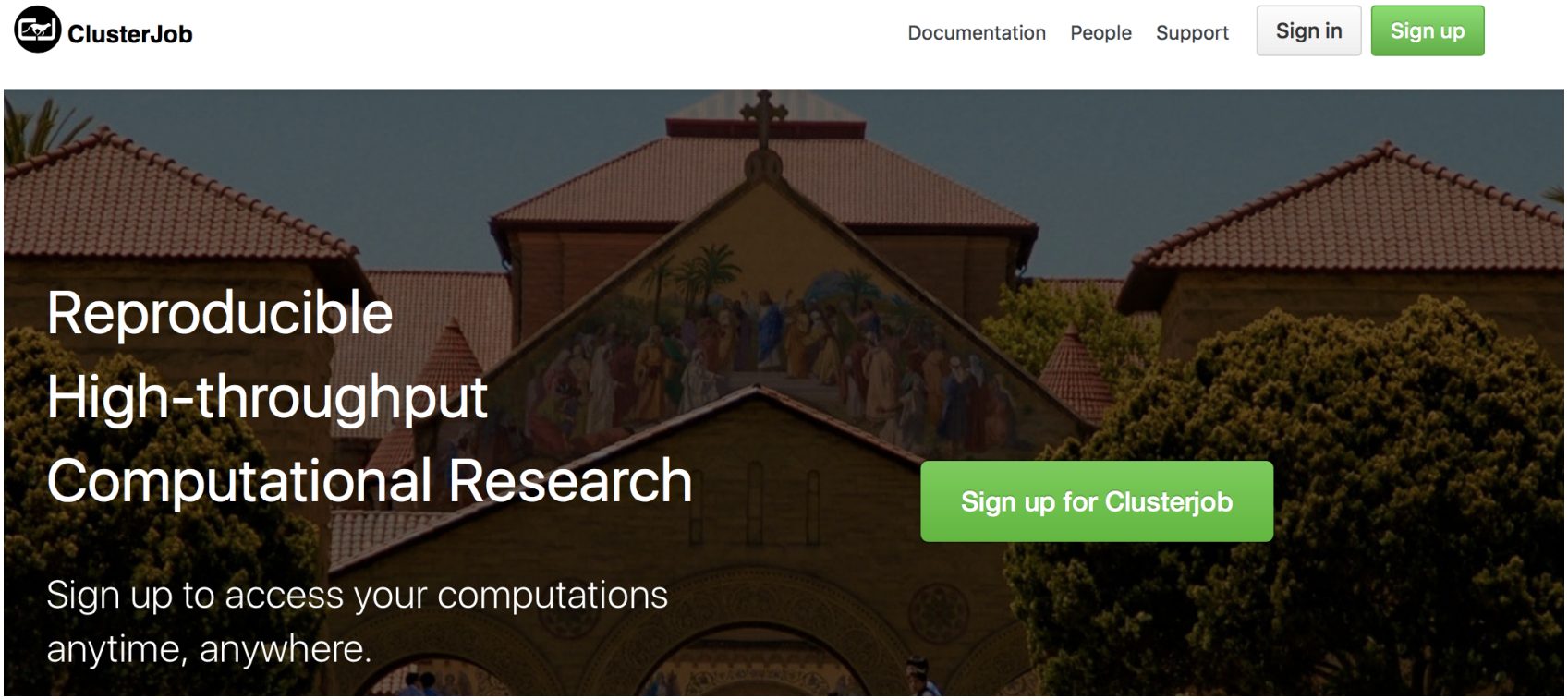
- **Simple**: the *right* level of abstraction!
- **Scalable**: push-button massive scaling-up of experiments
- **Reproducible**: all the tasks done in a reproducible way
- **Transparent**: easily be understood post facto

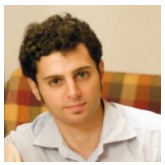
We will see next how we can build such a system

Outline

Stanford University

clusterjob.org



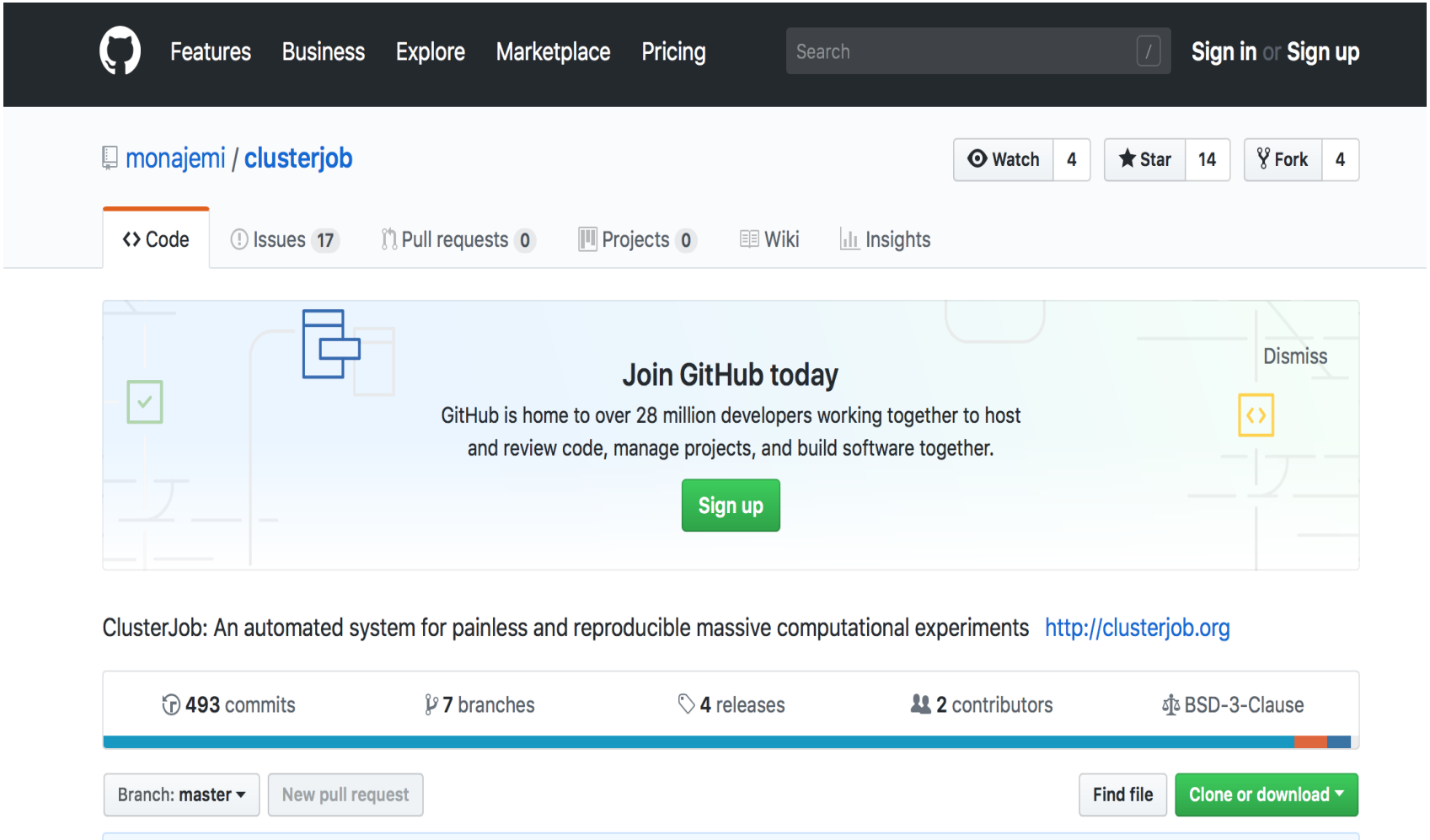


"This is how it [computation] should be done." – V. Morgenshtern

"Your software has made my life much easier." – C. Chang

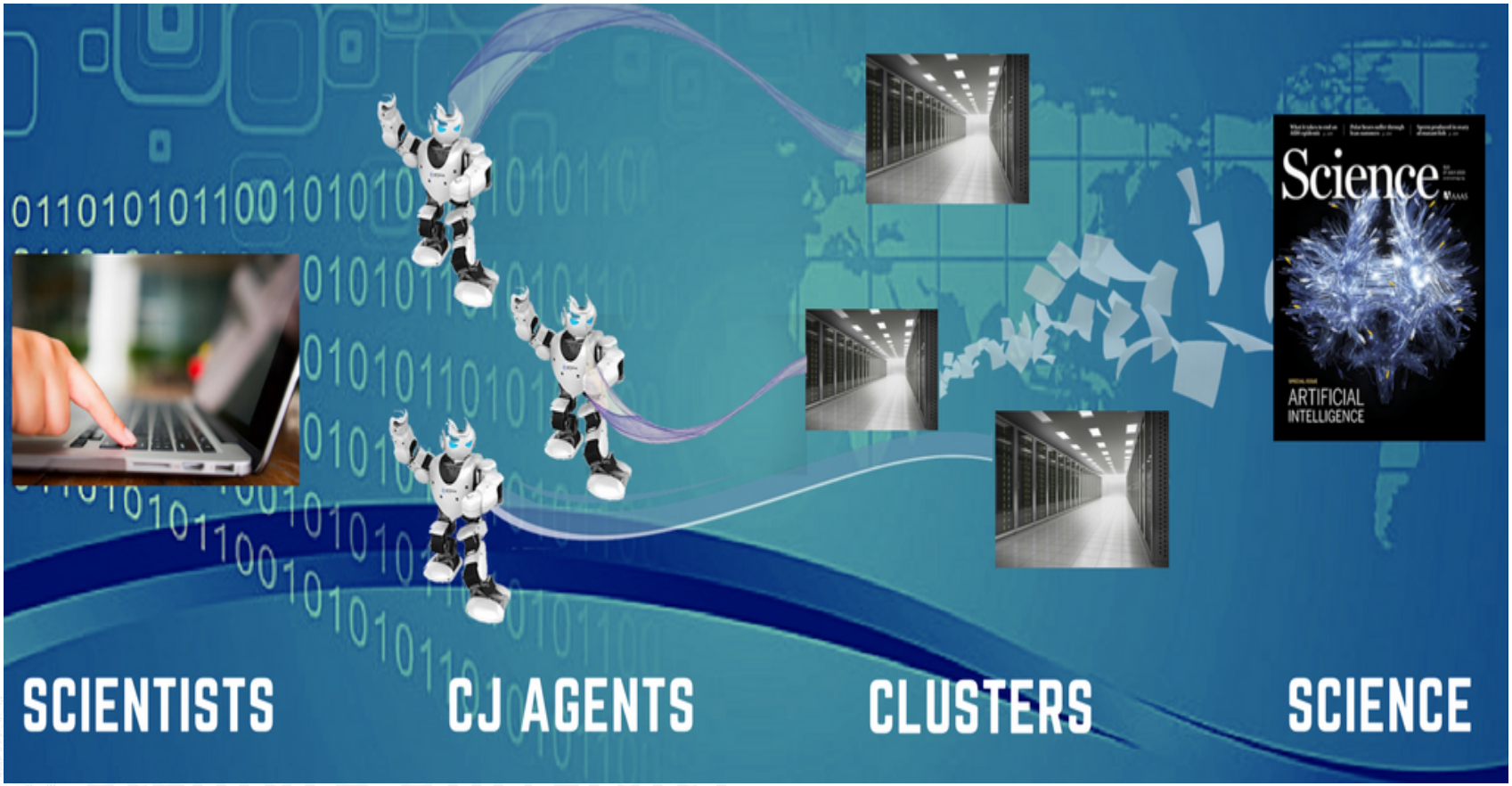
Stanford University

CJ on GitHub



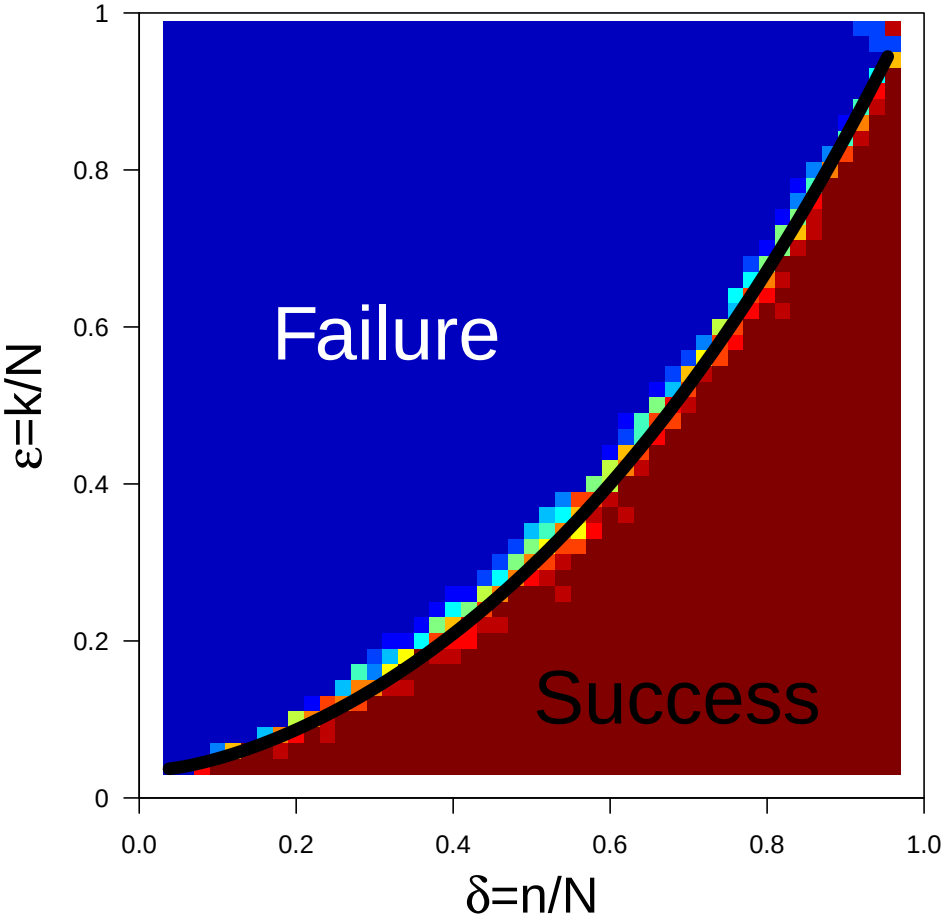
ClusterJob's model

- *Think experiments, implement in your language of choice*
- *Push a button, fire to your favorite cluster and forget*
- *Harvest, analyze and refine hypothesis*
- *Publish discovery with share reproducible CJ packages*



An example

Compressed Sensing Phase Transition Experiments:



How does CJ work, an example

Write a *simple and decipherable* MATLAB (Python, R) script:

```
% PT.m
% This test code calculates the
% probability of successful
% reconstruction in compressed sensing.
% Author: Hatef Monajemi Nov 1 2016

file = 'results.txt';
delta   = 0.1:.1:.9;
epsilon = 0.02:0.02:0.98;
for i = 1:length(delta)
for j = 1:length(epsilon)
  pr = computeProb(delta, epsilon);
  fid = fopen(file,'at');
  fprintf(fid, '%3.2f,%3.2f,%3.2f\n', ...
                 delta,epsilon,pr);
  fclose(fid)
end
end
```


Stanford University

How does CJ work, an example

- Submit 1 job:

```
$ cj run PT.m sherlock -dep bin -m "Test PT"
```

How does CJ work, an example

- Submit 441 separate jobs by a simple command

```
$ cj parrun PT.m sherlock -dep bin -m "Test PT"
```

```
CJmessage::initiating package 8ab7a5aa
CJmessage::parruning [PT.m] on [sherlock] with:
           alloc: -p donoho --time 48:00:00 --mem 8G
CJmessage::sending from: /Users/hatef/github_projects/CJ/clusterjob/example/MATLAB
CJmessage::LMOD module found on sherlock
CJmessage::Testing if module matlab/R2017a is available via LMOD:
           matlab/R2017a available.
CJmessage::Creating reproducible script(s) reproduce_PT.m
CJmessage::compressing files to propagate...
CJmessage::sending 2.19 MB to: sherlock2:/scratch/users/monajemi/CJRepo_Remote/PT
CJmessage::extracting package...
CJmessage::Submitting job(s)
CJmessage::441/441 job(s) submitted (10097772-10097786)
```

How does CJ work, an example

- Check status of jobs

```
$ cj state 8ab7a5aa

pid 8ab7a5aafa1b8232cc3da05a7814bed1d21dd0aa
remote_account: monajemi@sherlock.stanford.edu
1       10097772        COMPLETED
2       10097773        COMPLETED
3       10097774        COMPLETED
            .
            .
            .
441     10097786        RUNNING
```

How does CJ work, an example

- Retrieve information

```
$ cj log

pid 8ab7a5aafa1b8232cc3da05a7814bed1d21dd0aa
date: 2016-Oct-08  11:47:37  (GMT -07:00:00)
user: monajemi
agent: 2DCA5476-8197-11E6-B8C8-3A835C8A0BAC
account: monajemi@corn.stanford.edu
script: PT.m
initial_flag: parrun

          Test PT
```


Stanford University

How does CJ work, an example

- Sanity checks

```
$ cj sanity exists 8ab7a5aa
What file (e.g., results.txt | */results.txt)?
*/results.txt

✓ File 'results.txt' exists in all subPackages.
```

How does CJ work, an example

- Easily harvest results

```
$ cj reduce results.txt 8ab7a5aa
```

How does CJ work, an example

- . . . and many more functionalities

```
$ cj help
```

A look inside, core modules

CJ is written in Perl

clusterjob — -bash — 132×29

```
Hatefs-MacBook-Pro:clusterjob hatef$ ls
CJlog           LICENSE         cj_config       example         openall         ssh_config      test.sh
INSTALL         README.md       dep.pl          misc            src             ssh_config.bak  todo
Hatefs-MacBook-Pro:clusterjob hatef$ ls -1 src/CJ.*; ls -1 src/CJ/*
src/CJ.pl
src/CJ.pm
src/CJ/CJVars.pm
src/CJ/CJ_reduce.m
src/CJ/Get.pm
src/CJ/Install.pm
src/CJ/Matlab.pm
src/CJ/Python.pm
src/CJ/R.pm
src/CJ/Run.pm
src/CJ/Sanity.pm
src/CJ/Scripts.pm
src/CJ/Sync.pm
Hatefs-MacBook-Pro:clusterjob hatef$ ls src
CJ              CJ.pl           CJ.pm           external        sanity_checks tmp
Hatefs-MacBook-Pro:clusterjob hatef$
```

Configuring CJ - I

- Your CJID is unique
- Keep your CJKEY private (used for Firebase DB).

Configuring CJ - II

- Info of Clusters

clusterjob — -bash — 132×29

```
Hatefs-MacBook-Pro:clusterjob hatef$ cat ssh_config
[sherlock2]
Host            login.sherlock.stanford.edu
User            monajemi
Bqs             SLURM
Repo            /scratch/users/monajemi/CJRepo_Remote
MAT             matlab/R2017a
MATlib          ~/BPDN/CVX/cvx:~/mosek/7/toolbox/r2013a
Python          python/3.6
Pythonlib       pytorch:pandas:cuda80:scipy:matplotlib:torchvision:-c soumith
R               R/3.4.0
Rlib            ggplot2
Alloc           -p donoho --time 48:00:00 --mem 8G
[sherlock2]


[corn]
Host            corn.stanford.edu
User            monajemi
Bqs             SGE
Repo            /farmshare/user_data/monajemi/CJRepo_Remote
MAT             matlab/r2016b
MATlib          ~/BPDN/CVX/cvx:~/mosek/7/toolbox/r2013a
Python          python/3.4.3
Pythonlib       scipy:pytorch
[corn]


[rice]
```

What happens when you issue `parrun`?

- Pseudo code of PARRUN... Part I: Preparation

```
# build info of directories and package
my ($date,$ssh,$pid,$program_type,$localDir,$remoteDir)  = run_common($self);

# setup env
$self->setup_conda_venv($ssh) if($program_type eq 'python');
$self->setup_R_env($pid,$ssh) if ($program_type eq 'R');

# parse script out, find the loops, tags and ranges of indices
my $codeobj            = &CJ::CodeObj($self->{path},$self->{program},$self->{dep_folder});
my $parser             = $codeobj->parse();
my ($idx_tags,$ranges) = $codeobj->findIdxTagRange($parser,$self->{verbose});

# Check job is feasible
my $max_jobs = &CJ::max_jobs_allowed(...);
&CJ::err("Maximum jobs exceeded ...'') unless  ($max_jobs >= $totalJobs);


# build necessary submission scripts and reproducible code
$count = 0;
foreach my $loop (0..$nloops ){
  foreach my $i (0..$#idx_set ){
    $count++;
    &CJ::CodeObj("$localDir/$count",$program)->build_reproducible_script($runflag);
    &CJ::Scripts::make_par_shell_script($count,...);
    $master_script = &CJ::Scripts::make_master_script($master_script, $count, ...);
}}


# Compress and archive package
&CJ::system("tar -czf  $tarfile $pid/");
```

What happens when you issue `parrun`?

- Pseudo code of PARRUN... Part II: Firing up

```
# send package to cluster
&CJ::system("rsync -arvz  ${localDir}/${tarfile} $ssh->{account}:$remoteDir/");

# submit jobs
&CJ::system("ssh $ssh->{account} 'bash -l master.sh > $remoteDir/qsub.info")

# bring back submission info
&CJ::system("rsync -avz $ssh->{account}:$qsubfilepath  $info_dir")

# parse submission info
($job_ids,$errors) = &CJ::read_qsub($local_qsub_info_file);
$self->_checkSubmitSuccess($job_ids,$errors, ...);


# record run info
my $runinfo={
    pid           => ${pid},
    user          => ${CJID},
    ...
};


# save record locally and remote DB
&CJ::add_record($runinfo);
&CJ::write2firebase($pid,$runinfo, ...);
```

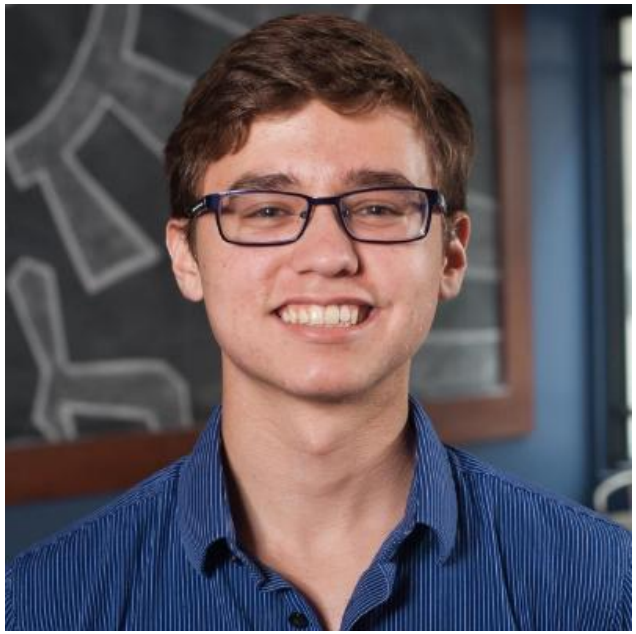
What about the data?

- There is a number of applications for data transfer:
  - scp
  - rsync (used by CJ)
  - Globus
  - bbcp (from SLAC)

- 'Comment-CJ' directive for data already on the cluster:

```
%CJ -s 'local-path' 'cluster-path'
#CJ -s 'local-path' 'cluster-path'
```

Future development: CJHub



Bekk Blando (CJ contributor, Clemson)

- CJHub is CJ's cloud storage
- will host all CJ packages with user consent.
- make it easy to share packages: `cj share PID user`
- automatically archives packages after run.

Summary

- We are experiencing a *computing phase transition*
- Scientists need to embrace and adapt to this change!
- EMS is a necessity for data science research.
- Open-source CJ is an example of EMS
- You can use CJ for Matlab, Python and R scripts.
- Contributions to CJ are welcomed!