

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

H. Monajemi/DL. Donoho

Stats285, Stanford

Oct/09/2018

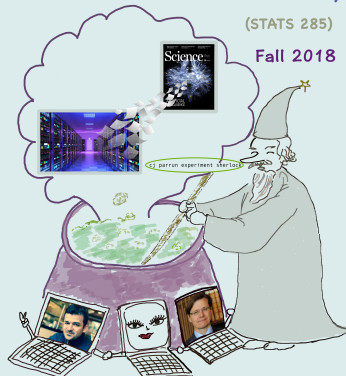


Stanford University

## Massive Computational Experiments, Painlessly

(STATS 285)

Fall 2018



Time: Tuesday 3:00-4:20 PM

Place: Thornt 110

Website: [stats285.github.io](https://stats285.github.io)



Stanford University

*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.*



- 1 Announcements!
- 2 Review of the Past Lectures!
- 3 Automation in Data Science
- 4 Why Can Cluster Computing Seem Painful?
- 5 How Can We Make Cluster Computing Less Painful?
- 6 CJ: An Open-source EMS



## Meet Hackathon Mentors



Vardan Papyan



[Alon Kipnis](#)



Yaniv Romano



Morteza Mardani



Pete Mohanty



Sara Hooshangi

## Time and Venue

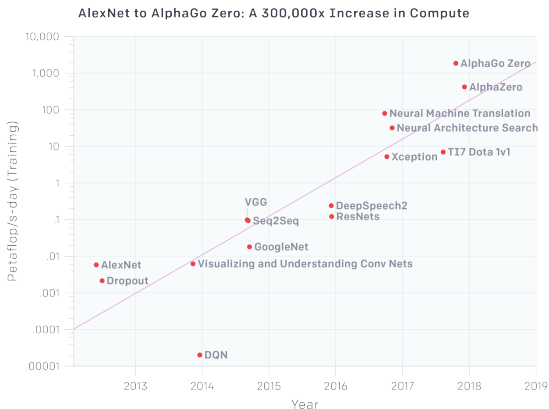
The hackathon starts at 8:00 AM on Nov 17 and ends at midnight on Nov 18. The event takes place at [Wallenberg Hall-Building 160](#).



- **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/>)



- 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



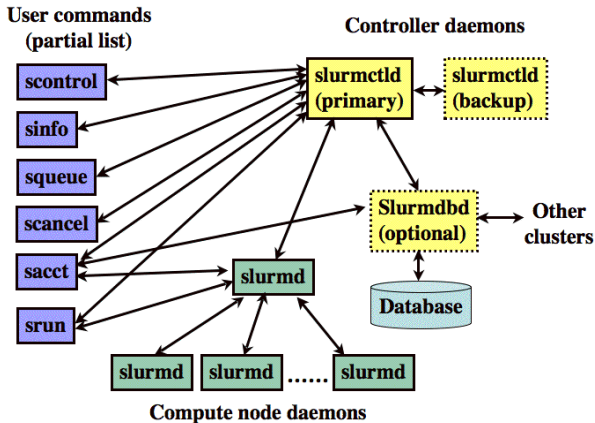
- Simple **L**inux **U**tility **R**esource **M**anagement
- Used by many clusters today



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



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



SLURM architecture



## **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



- 1 Announcements!
- 2 Review of the Past Lectures!
- 3 Automation in Data Science**
- 4 Why Can Cluster Computing Seem Painful?
- 5 How Can We Make Cluster Computing Less Painful?
- 6 CJ: An Open-source EMS



CTF

More Tweak  $\longrightarrow$  More Win



CTF

More Tweak  $\longrightarrow$  More Win

Data Science

More Experiments  $\longrightarrow$  More Discovery



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



To critics of automation in 1961:

- proper automation → **more study** of the data
- automation of known procedure → **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



- 1 Announcements!
- 2 Review of the Past Lectures!
- 3 Automation in Data Science
- 4 Why Can Cluster Computing Seem Painful?**
- 5 How Can We Make Cluster Computing Less Painful?
- 6 CJ: An Open-source EMS



You need to learn the  
job schedulers

For python, you just  
use

```
$ pip install --user  
to install
```

But I just want to  
run my experiments,  
Do I really have to  
go through all of  
this?

## 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



Stanford University

# Many clusters, many systems, many policies!

**Using the OSG**

**OSG Client Software**

All of the following sections will focus on job and data management on the OSG in order to build a successful grid application. These will all assume that you have access to (an account on) an OSG Submit host provided by your VO or have the [OSG client software installed](#).

**Running Jobs**

- Finding OSG sites to run your jobs
- Testing your first resources
- Running jobs with Condo-G: Condo-G is the most common client tool for submitting patterns; each link builds upon the knowledge learned in the previous one.
  - Submitting a single, single job with Condo-G.
  - Submitting multiple, single jobs with Condo-G.
  - More complex Condo-G jobs.
  - Building workflows in Condo (external link). This link covers in-depth how to discuss vanilla universe jobs, but DAGMan works with grid universe jobs.
- Running jobs with glideWMS
  - How to connect a Condo-G submit file to use glideWMS: Condo-G job
  - Using the Condo Frontend: glideWMS Condo installation and tutorial
- The OSG job environment (Where can I install software? What's in the environment?)
- Using the OSG Matchmaker to run jobs. The OSG Match-Maker

**AWS Batch**

User Guide

Documentation - This Guide

- What Is AWS Batch?
- Setting Up
- Getting Started
- Jobs
  - Submitting a Job
  - Job States
  - Automated Job Retries
  - Job Definitions

**Google Cloud Platform**

Why Google Products Solutions Launcher

**Cloud Dataproc**

Product Overview Documentation

Quikstarters

- All Quikstarters
- Using the API Explorer
- Using the console
- Using the gcloud command-line tool

**How-to Guides**

- All How-to Guides
- Set up a project
- Create a cluster
- Submit a job
- Manage a cluster

**APIs & Reference**

All APIs & Reference

- REST

**TACC USER PORTAL**

HOME NEWS RESOURCES ALLOCATIONS DOCUMENTATION TRAINING CONSULTING ABOUT

ACCOUNT REQUEST PASSWORD RESET NEW USER INFORMATION CONTACT ACCOUNT PROFILE

- 09/18/17 Stampede 1's VMs sub-system is no longer available, and the VMs material in the Stampede 1 User Guide is now obsolete. We have begun the process of moving Stampede 1 VMs VMs nodes to Stampede2. See the [Stampede2 documentation](#) for more information.
- 08/01/17 The maximum number of nodes requestable in the development queue has been reduced. Jobs are now limited to four nodes. See [Stampede2 Production Queue](#) for other info.
- 08/21/16 This user guide has been updated substantially to reflect the new Knight Landing (KNC) Upgrade. Most of the older Knight Corner (KNC) experience content has been rewritten in the new [Stanford Knight Corner Technical Manual](#).
- 09/11/16 Multi-Factor Authentication (MFA) is now mandatory in order to access all TACC resources. Please see the [Multi-Factor Authentication \(MFA\) Manual](#) for assistance in setting up your account.

**Introduction**

TACC's Stampede system, generously funded by the National Science Foundation (NSF) through award ACI-1154872, entered production in January 2012 as a 6400+ node cluster of 2nd level Dell R710 server nodes. Nodes have been ES-Ready Edge host processes and the new Knight Corner (KNC) processor, the first generation of processors

**STANFORD UNIVERSITY | SHERLOCK CLUSTER**

**STANFORD UNIVERSITY | FARM SHARE**

**SLURMSubmit**

SLURM @ stanford supports a variety of job submission techniques. By accurately requesting the resources you need, it's important to understand that the more resources you request (CPU, RAM and especially nodes) the faster your job will be queued/partitioned and how many resources you have used in the past (roughly two weeks). This describes

**GridEngine**

We're using the Debian packages of "Sun Grid Engine" which sort of Open Grid Engine or Sun Grid Engine or Univa Grid Engine.

**CONTENTS (just)**

- 1 useful commands
- 2 documentation
- 3 email alerts
- 4 quotas
- 5 Grid Engine settings on farmshare
- 6.1 setting 6
- 6.2 spring thing 6
- 6.3 3rd file 6
- 6.4 making the test\_6
- 6.5 3rd file 6
- 6.6 3rd file 6
- 6.7 3rd file 6
- 6.8 3rd file 6
- 6.9 3rd file 6
- 6.10 3rd file 6

**CONTENTS (link)**

- 1 Batch Job Submission
- 2 Sample Batch Job
- 3 Submit multiple jobs at once with wrap
- 4 Interactive Jobs
- 5 Other ways to request resources
- 6 Monitoring your jobs
- 7 Controlling jobs
- 8 Job cancellation



Stanford University

## 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:

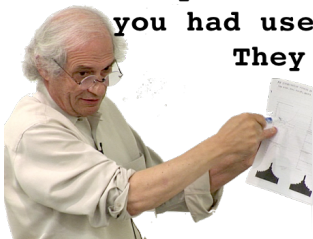
```
#!/bin/sh
for FILE in *.fastq;
do sbatch -p normal -t 10 --mem=200 --wrap="gzip ${FILE}"
sleep 1 # pause for 1 second so we don't overload the scheduler
done
```

**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!



Stanford University

- 1 Announcements!
- 2 Review of the Past Lectures!
- 3 Automation in Data Science
- 4 Why Can Cluster Computing Seem Painful?
- 5 How Can We Make Cluster Computing Less Painful?**
- 6 CJ: An Open-source EMS



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.

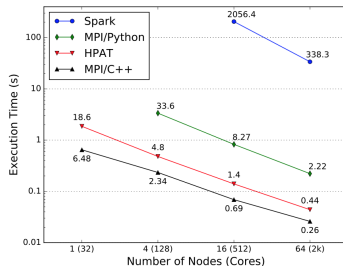
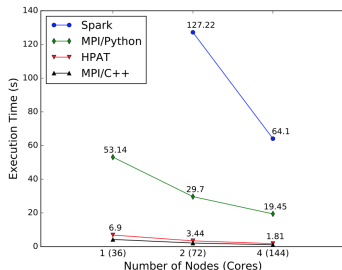
An Experiment Management System should automate and seamlessly integrate all these tasks



# 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- (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"

Stanford University



# Desired features of an EMS

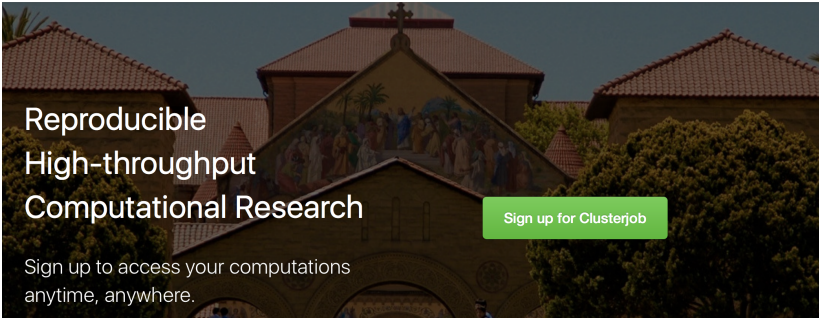
- **Simple:** the *right* level of abstraction!
- **Scalable:** push-button massive scaling-up of experiments
- **Reproducible:** all the tasks done in a reproducible way
- **Transparent:** easily be understood post facto

We will see next how we can build such a system



- 1 Announcements!
- 2 Review of the Past Lectures!
- 3 Automation in Data Science
- 4 Why Can Cluster Computing Seem Painful?
- 5 How Can We Make Cluster Computing Less Painful?
- 6 **CJ: An Open-source EMS**





## Reproducible High-throughput Computational Research

[Sign up for Clusterjob](#)

Sign up to access your computations  
anytime, anywhere.



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



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



Stanford University

The screenshot shows the GitHub repository page for `monajemi/clusterjob`. At the top, there is a navigation bar with links for Features, Business, Explore, Marketplace, and Pricing, along with a search bar and Sign in or Sign up buttons. Below the navigation bar, the repository name `monajemi/clusterjob` is displayed, followed by statistics: 4 Watchers, 14 Stars, and 4 Forks. A secondary navigation bar includes links for Code, Issues (17), Pull requests (0), Projects (0), Wiki, and Insights. A large banner for GitHub promotion is present, with the text "Join GitHub today" and "GitHub is home to over 28 million developers working together to host and review code, manage projects, and build software together." A green "Sign up" button is centered in the banner. Below the banner, the repository description reads: "ClusterJob: An automated system for painless and reproducible massive computational experiments" with a link to <http://clusterjob.org>. A statistics bar shows 493 commits, 7 branches, 4 releases, 2 contributors, and BSD-3-Clause license. At the bottom, there are buttons for "Branch: master", "New pull request", "Find file", and "Clone or download".

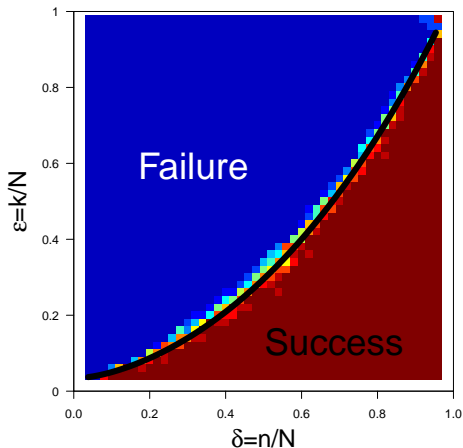


# ClusterJob's model

- *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*



## Compressed Sensing Phase Transition Experiments:





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 = 0.1:0.1:0.9;
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
```



# 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 8ab7a5aafal1b8232cc3da05a7814bed1d21dd0aa  
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
```



# 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 — 132x29
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).

```
clusterjob -- -bash -- 99x32
Hatefs-MacBook-Pro:clusterjob hatef$ cat cj_config
CJID    moosh
CJKEY   <YOUR_CJKEY>
SYNC_TYPE    manual
SYNC_INTERVAL 300
```



## • Info of Clusters

```
clusterjob — -bash — 132x29
Hatefs-MacBook-Pro:clusterjob hatef$ cat ssh_config
[sherlock2]
Host                login.sherlock.stanford.edu
User                monajemi
Bq                 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
Bq                 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 $starfile $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'
```





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.



- 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!

