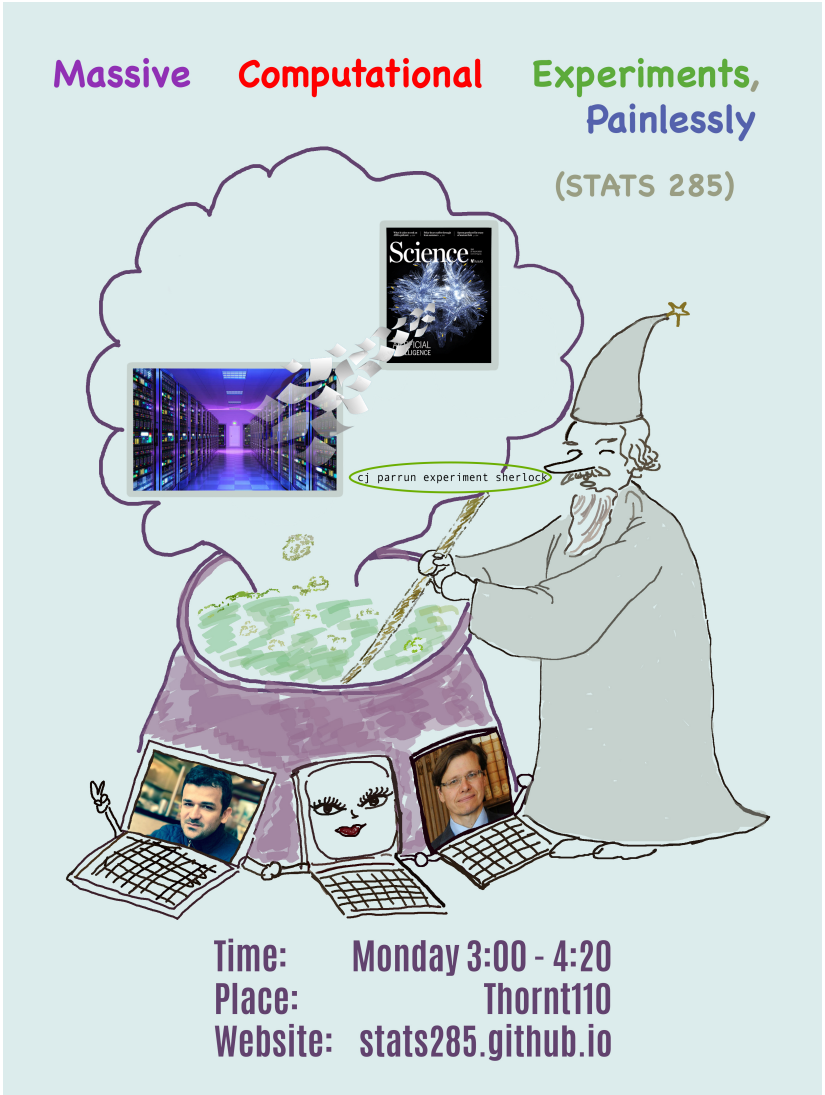
Lecture 02: The Cluster Scaling Revolution, and Clusterjob

H. Monajemi/DL. Donoho

Stats285, Stanford

Oct/02/2017

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.*

Disclaimer, 2: Tweets

This presentation contains several **fake tweets** *that can be easily created online and are included solely as a parody.*

No-one should take anything in tweets seriously.

Outline

Global Economy → Computing → Science

Stanford University

Explosion of Computational Resources

Cloud Paradigm:

- Billions of smart devices each drive queries to cloud servers
- Millions of business relying on cloud for all needs

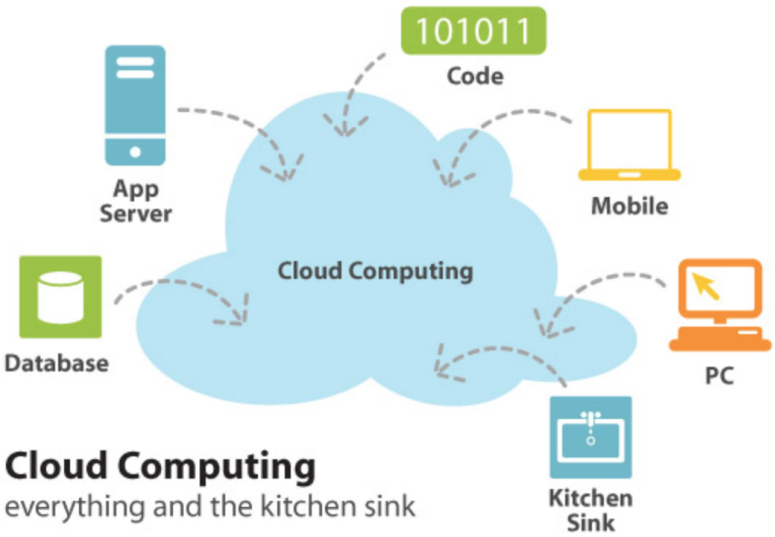
Symbiosis of cloud and economy is *lasting* and *disruptive*.

Cloud provides *any user* **same-day** delivery:

- Tens to hundreds of thousands of hours of CPU
- Pennies per CPU hour

Any user can consume *1 Million CPU hours* over a few days for a few $10K's.

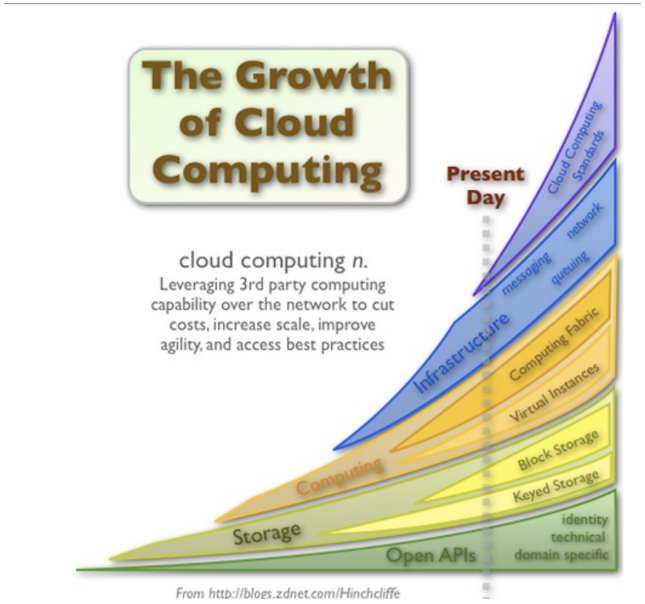
Cloud is all-purpose



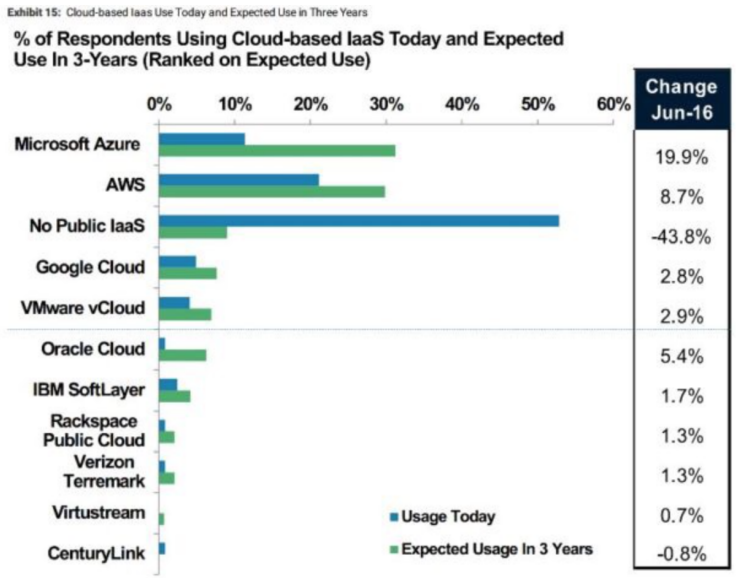
Many Uses

Widespread Advocacy

Services offered by Cloud Expanding

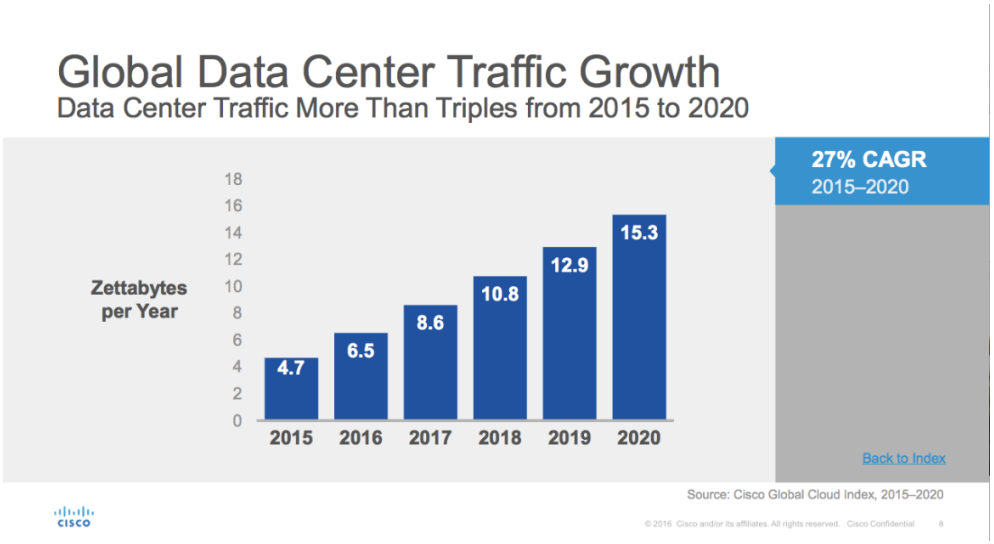


Proliferating Services

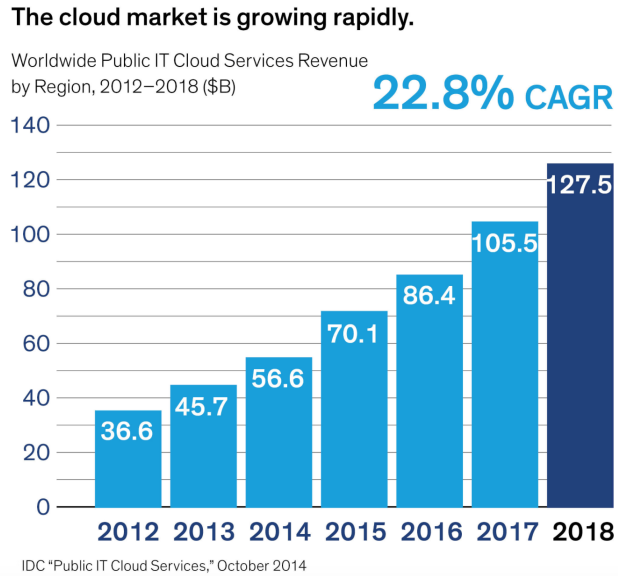


Businesses Expect Growth

Global Data Center Growth

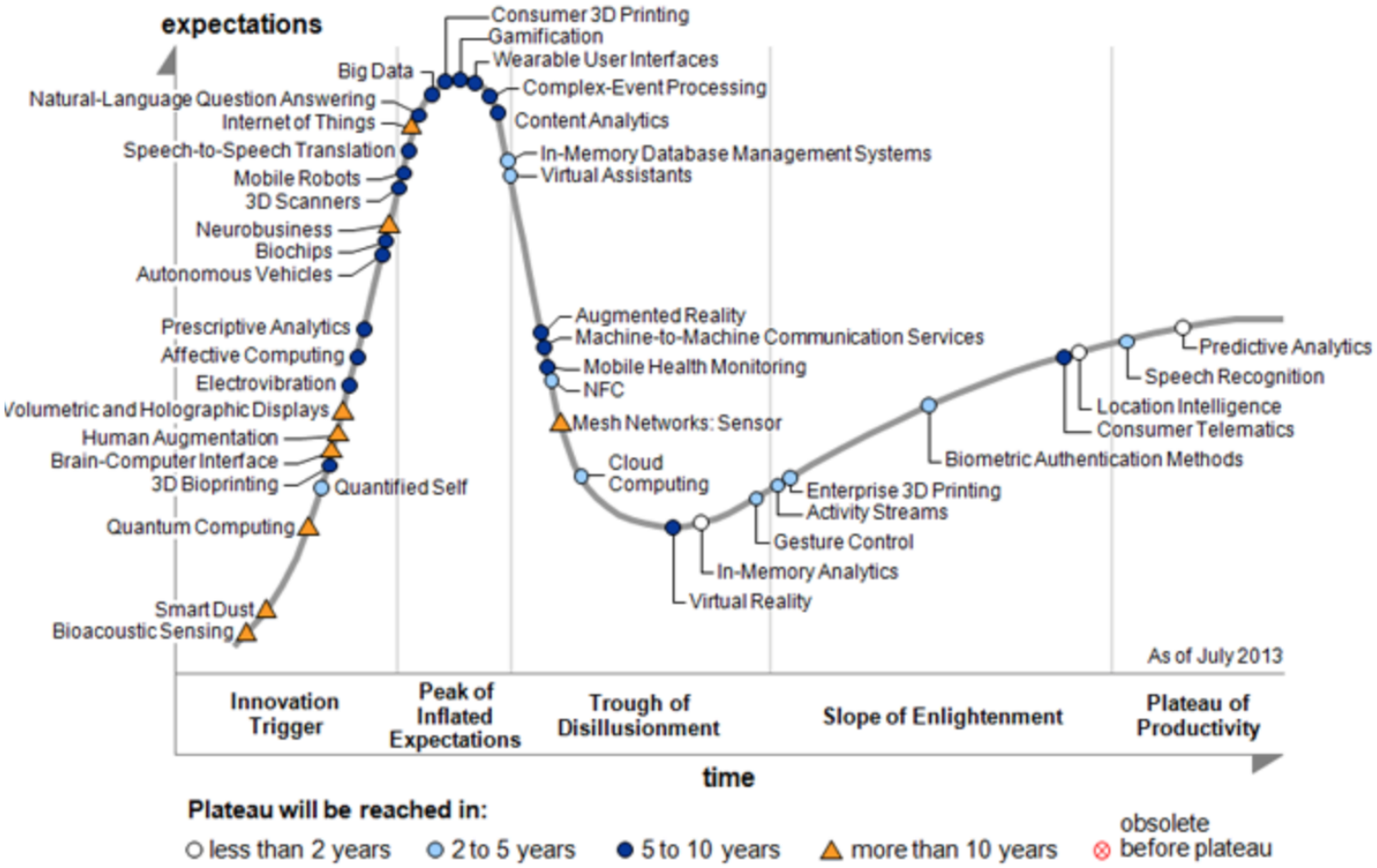


Traffic

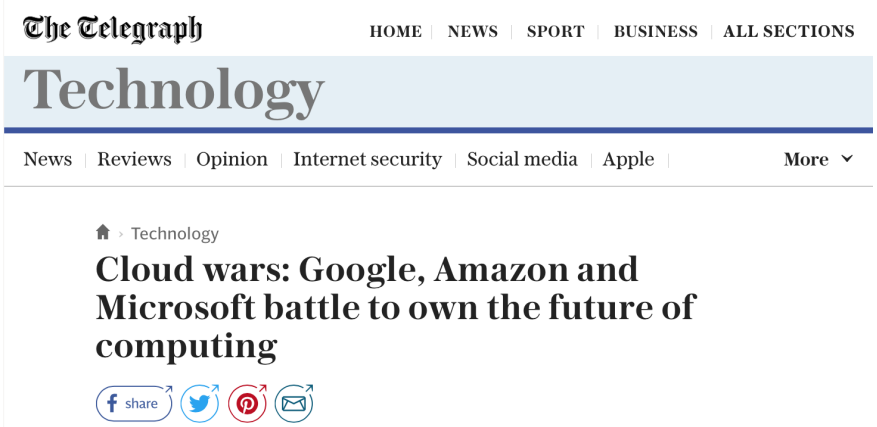


Revenue

Cloud Computing past 'Hype Hump'



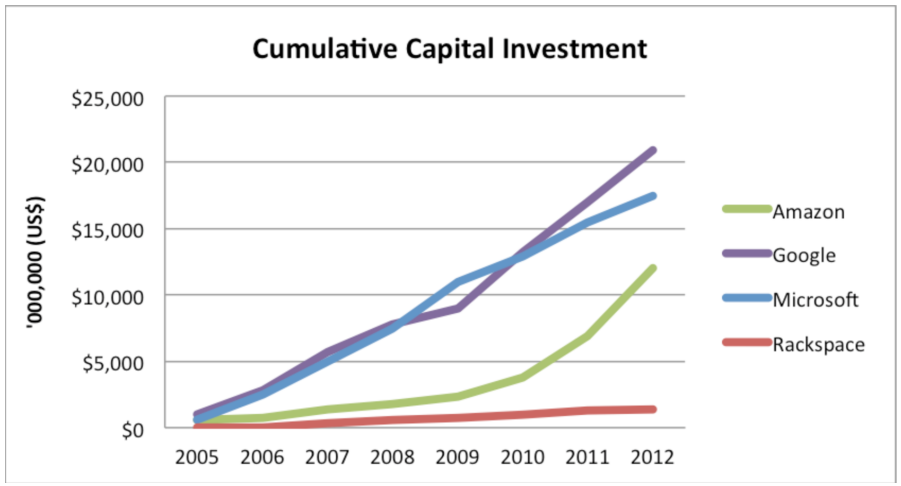
Cloud Wars



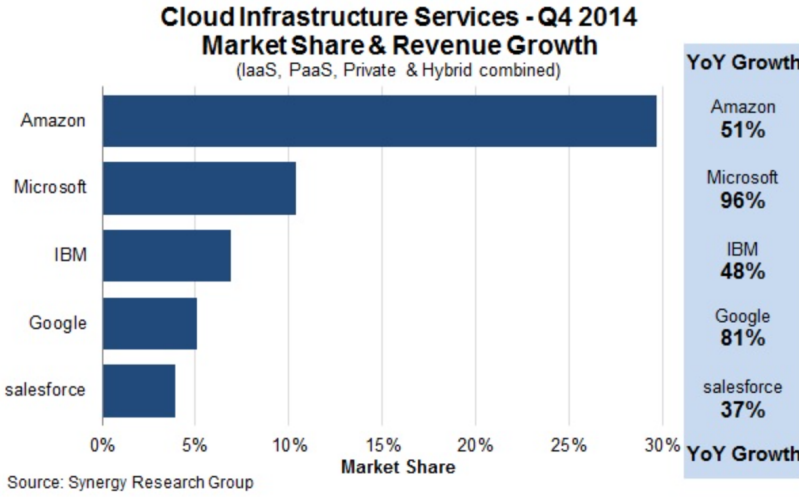
Media

Symbols

Big Three Invest and Profit



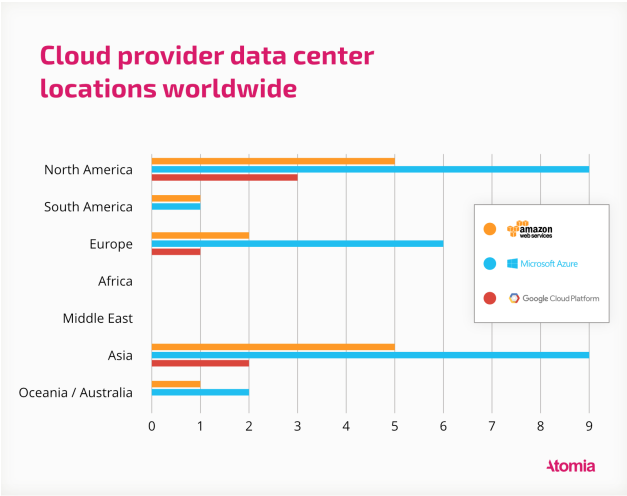
Traffic



Revenue

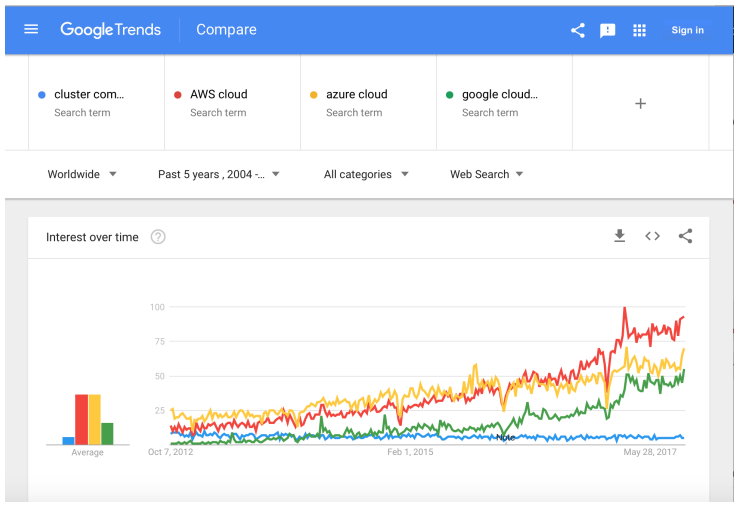
Big Three Have Global Reach

Map

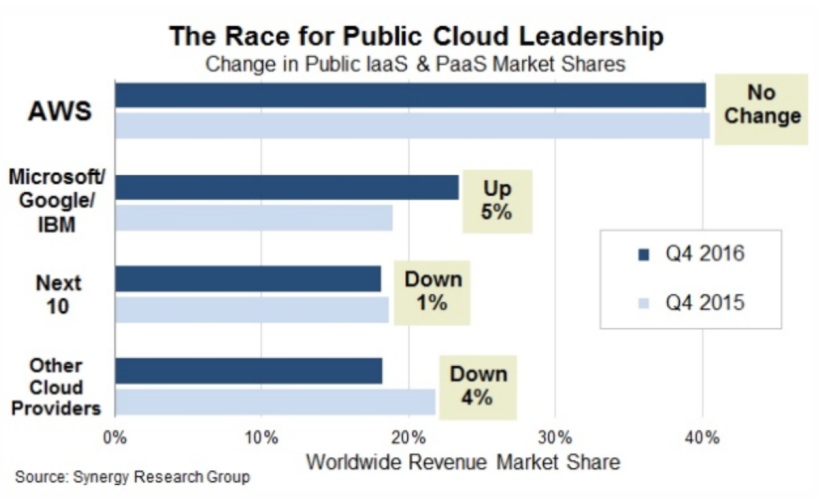


Breakdown

Big Three Have Buzz

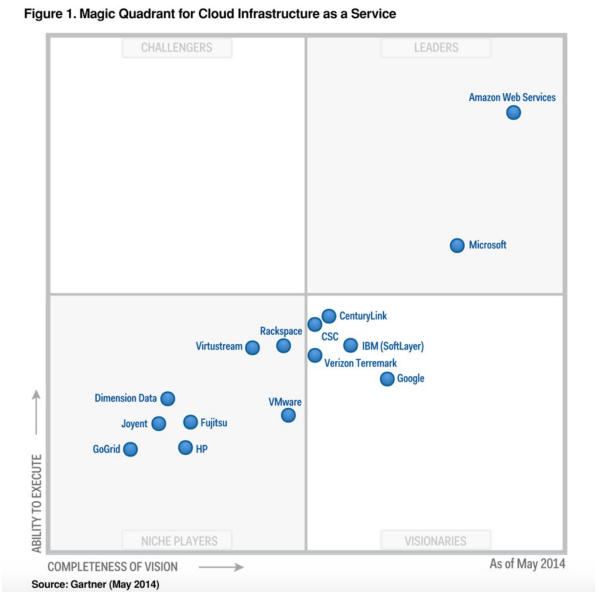


Google Trends

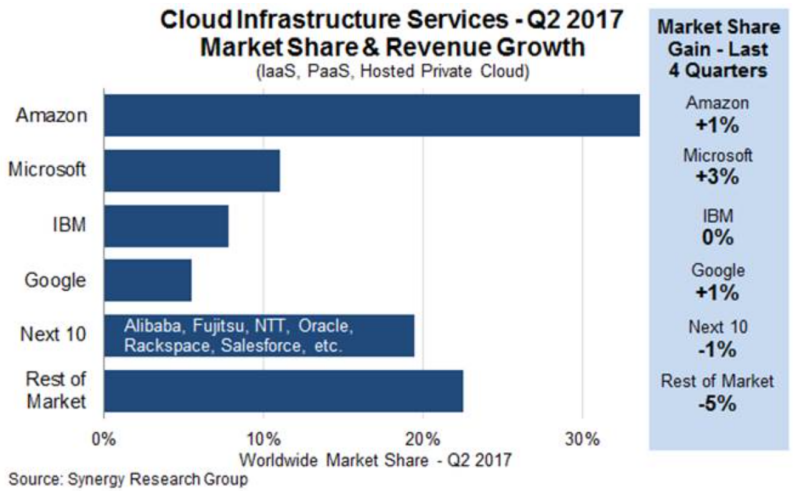


Market Share Growth

AWS Solidly Leading



Gartner Magic Quadrant



Market Share

AWS is world's largest computer

AWS placed its data centers across 33 availability zones within 12 regions worldwide. Each availability zone has at least one data center (some have as many as six) that has redundant power for stability, networking and connectivity. In each data center, there are between 50,000 to 80,000 servers with up to 102 Tbps bandwidth.

If you assume an average of three data centers per zone and 65,000 servers per data center, you will end up having 6.4 million servers worldwide. For those of you who care about availability and performance of their applications in the cloud, the huge computing capacity of AWS ensures higher fault tolerance and low latency.

Cloud Prices Approx $0.03/(GB-Hour)

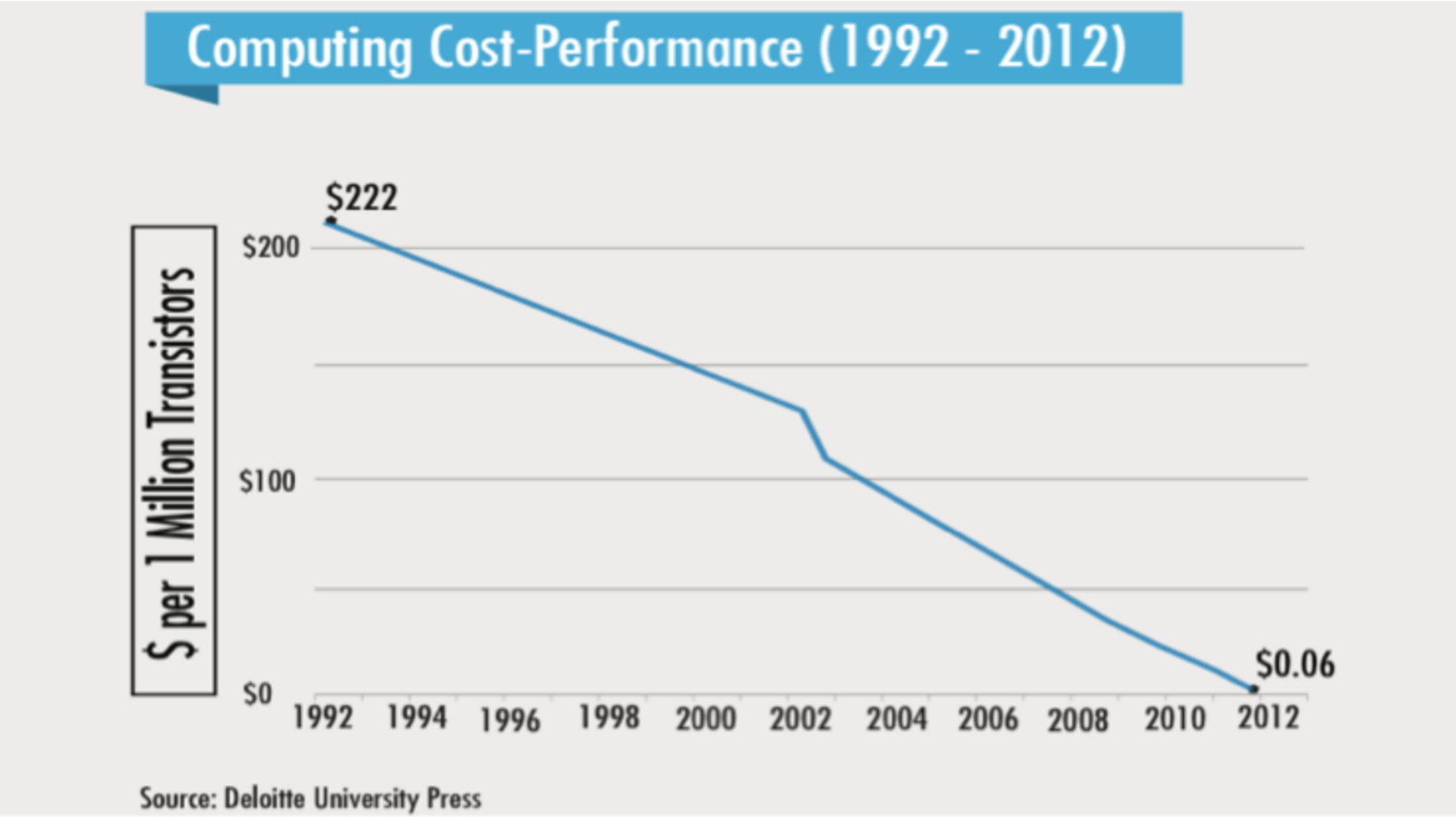
AWS vs. Azure vs. Google On-Demand Prices

Resource Type (us-east, Linux)	AWS Instance	Azure Instance	Google Instance	AWS OD Hourly	Azure OD Hourly	Google OD Hourly	AWS /GB RAM	Azure /GB RAM	Google /GB RAM
Standard 2 vCPU w SSD	m3.large	D2 v2	n1-standard-2	$0.133	$0.114	$0.212	$0.017	$0.016	$0.028
Highmem 2 vCPU w SSD	r3.large	D11 v2	n1-highmem-2	$0.166	$0.149	$0.238	$0.011	$0.011	$0.018
Highcpu 2 vCPU w SSD	c3.large	F2	n1-highcpu-2	$0.105	$0.099	$0.188	$0.028	$0.025	$0.104
Standard 2 vCPU no SSD	m4.large	D2 v2	n1-standard-2	$0.108	$0.114	$0.100	$0.014	$0.016	$0.013
Highmem 2 vCPU no SSD	r4.large	D11 v2	n1-highmem-2	$0.133	$0.149	$0.126	$0.009	$0.011	$0.010
Highcpu 2 vCPU no SSD	c4.large	F2	n1-highcpu-2	$0.105	$0.099	$0.076	$0.027	$0.025	$0.042

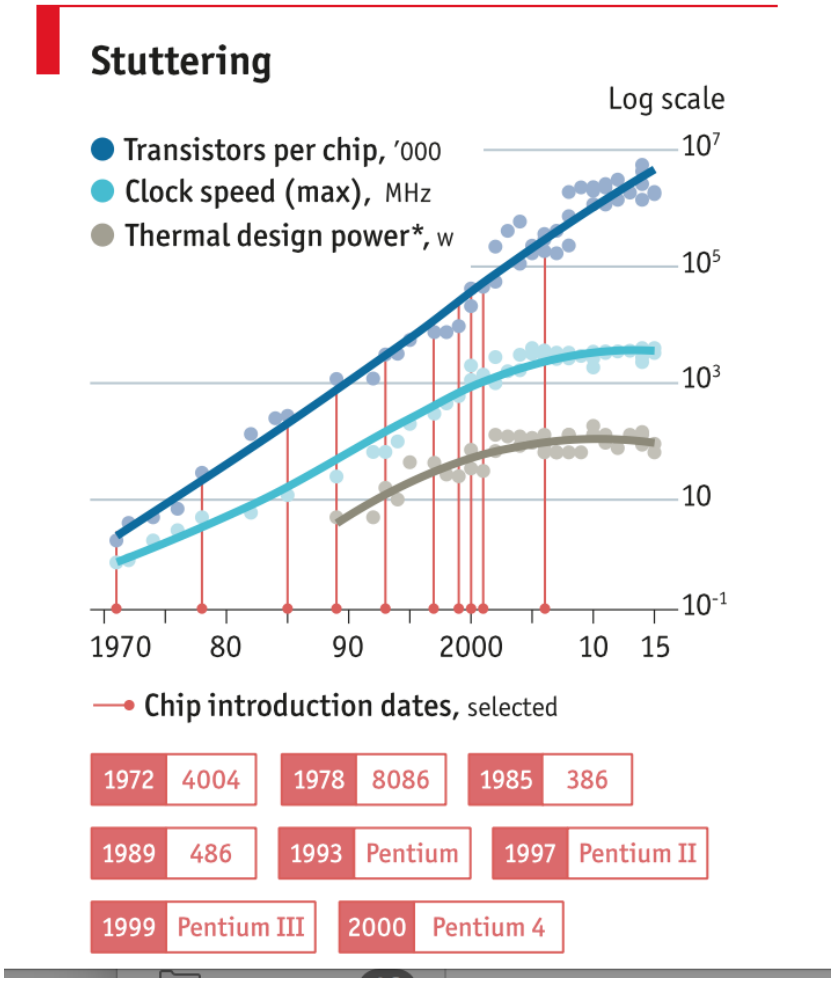
As of Dec 2, 2016

Source: RightScale

Traditional Story of Moore's Law: Inexorable Progress



Recent Story of Moore's Law: Stagnation

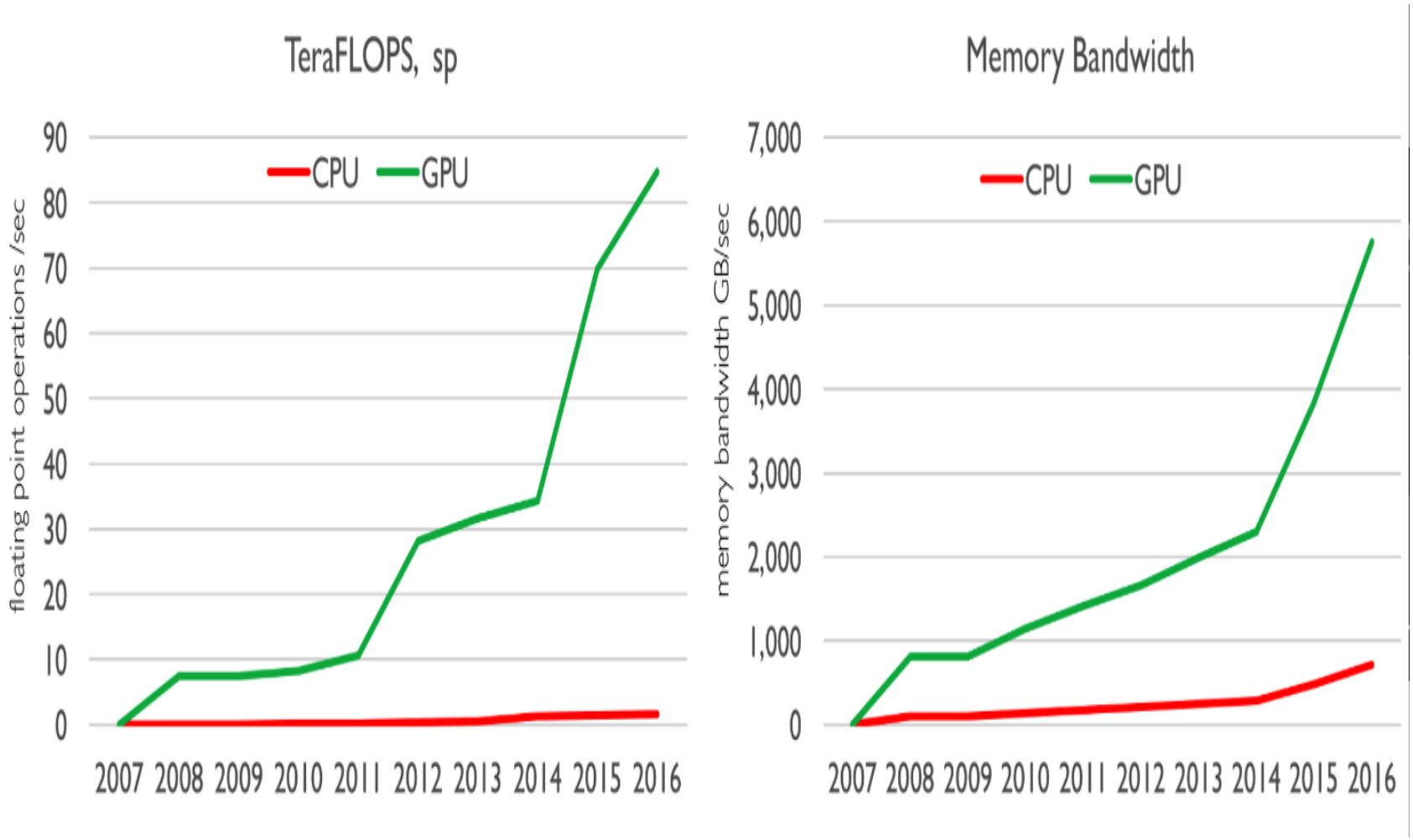


The result is a consensus among Silicon Valleys experts that **Moore's law is near its end**. *"From an economic standpoint,* **Moores law is dead***", says Linley Gwennap, who runs a Silicon Valley analysis firm. Dario Gil, IBM's head of research and development, is similarly frank: "I would say categorically that* **the future of computing cannot just be Moores law any more**." *Bob Colwell, a former chip designer at Intel, thinks the industry may be able to get down to chips whose components are just five nanometres apart by the early 2020s but "you'll struggle to persuade me that they'll get much further than that.*

Guardian

`https://www.theguardian.com/technology/2017/jan/26/vanishing-point-rise-invisible-computer`

GPU's may give temporary respite



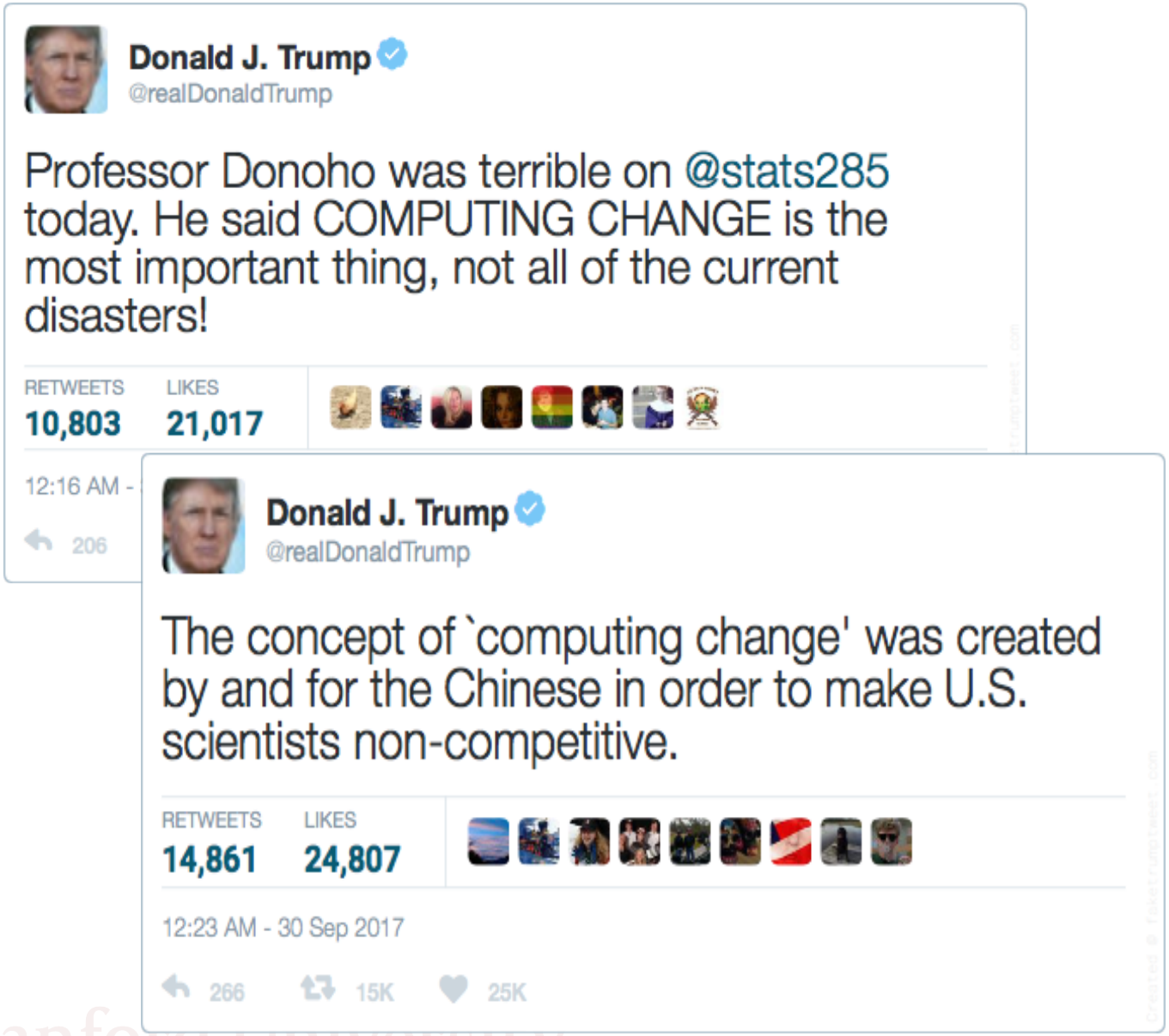
GPU performance forecast to saturate by 2020.
AWS is offering GPU's

Conclusions

- Over the last ten years, massive computational resources have been created
- Publicly available to anyone for price 3 cents per GB per CPU hour
- Near-unlimited quantities (for a price)
- Expansion by factors of 1000's in immediate computing capacity when job is *trivially parallelizable*
- Traditional routes to enhanced performance are blocked.
- Welcome to the era of **Computing Change**

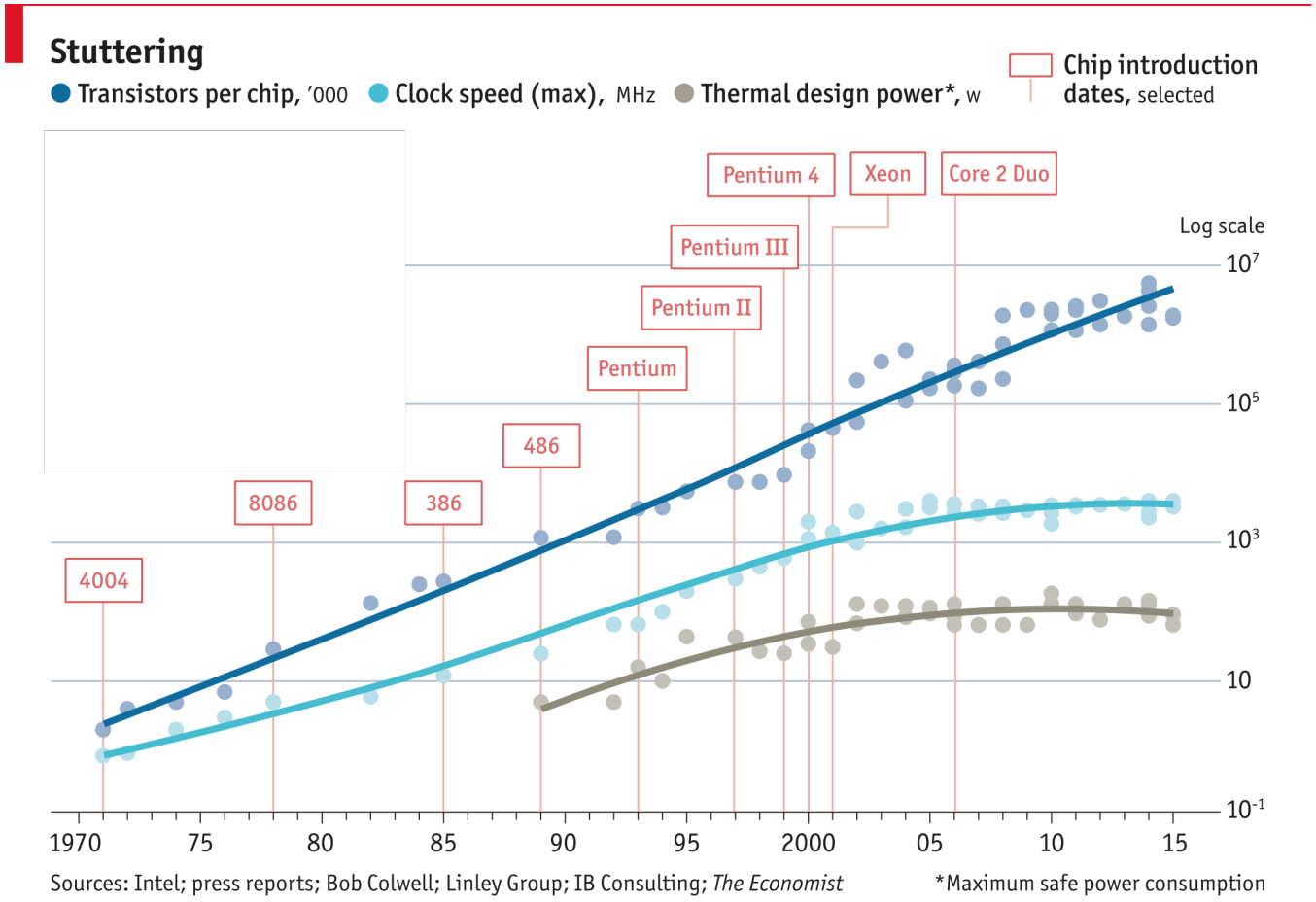
Computing Change Era!

Computing change skeptics



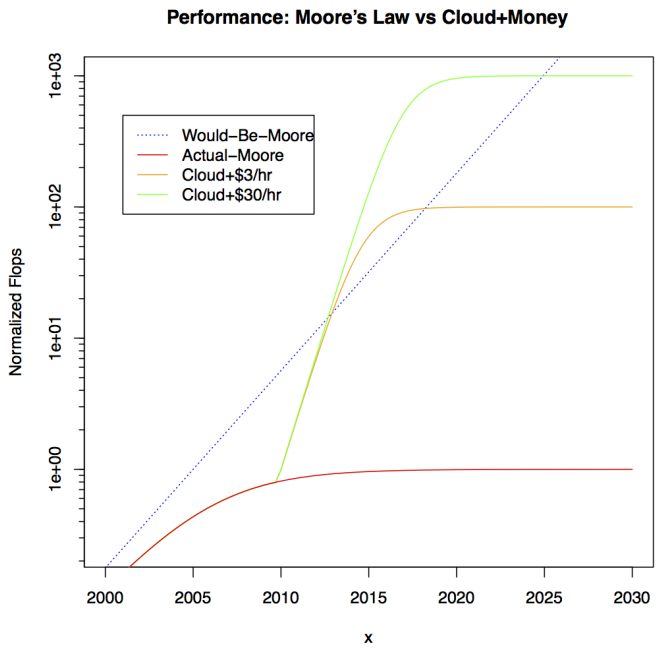
Computing Phase Transition - I

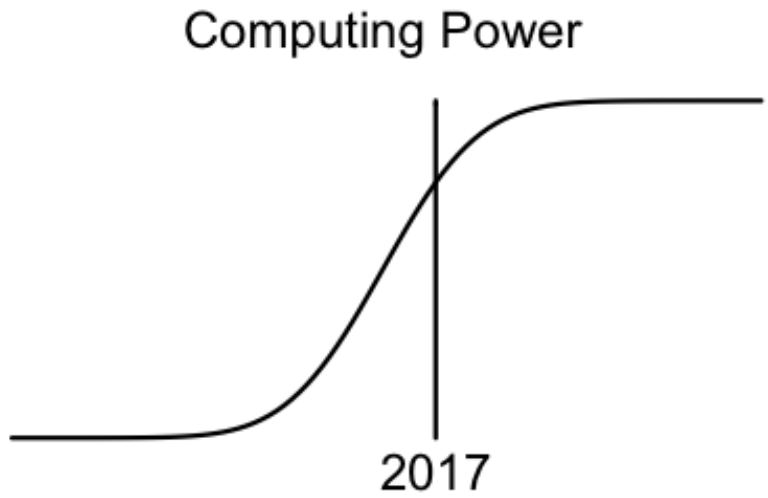
- Classic Moore's law predicts 32x increase from 2010-2020
- Moore's law no longer possible *physically* and *economically*



Computing Phase Transition - II

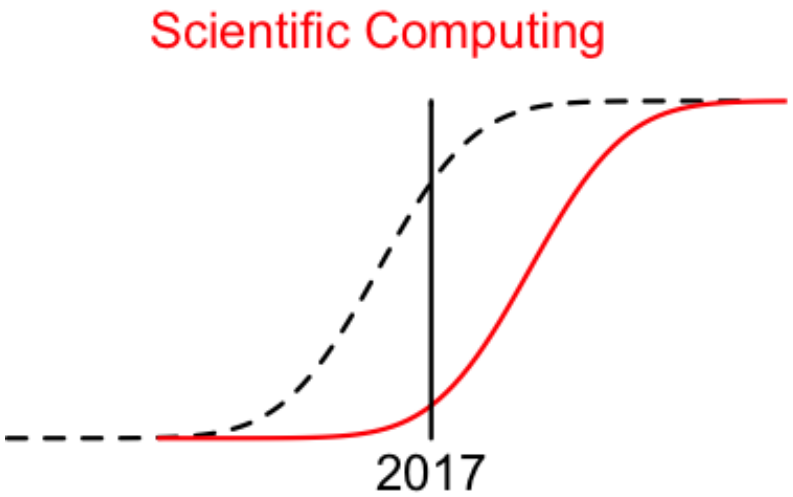
- Classic Moore's law predicts 32x increase from 2010-2020
- Moore's law no longer possible *physically* and *economically*
- . . . Today, computing power *easily* accessible ↑ 10,000x
  - \+ thanks to the symbiosis of cloud and economy!





Computing Phase Transition - III

- Classic Moore's law predicts 32x increase from 2010-2020
- Moore's law no longer possible *physically* and *economically*
- . . . Today, computing power *easily* accessible ↑ 1000x
- . . . Traditional laptop/desktop scientific computing will lag behind



Computing change real, Hot science emerging

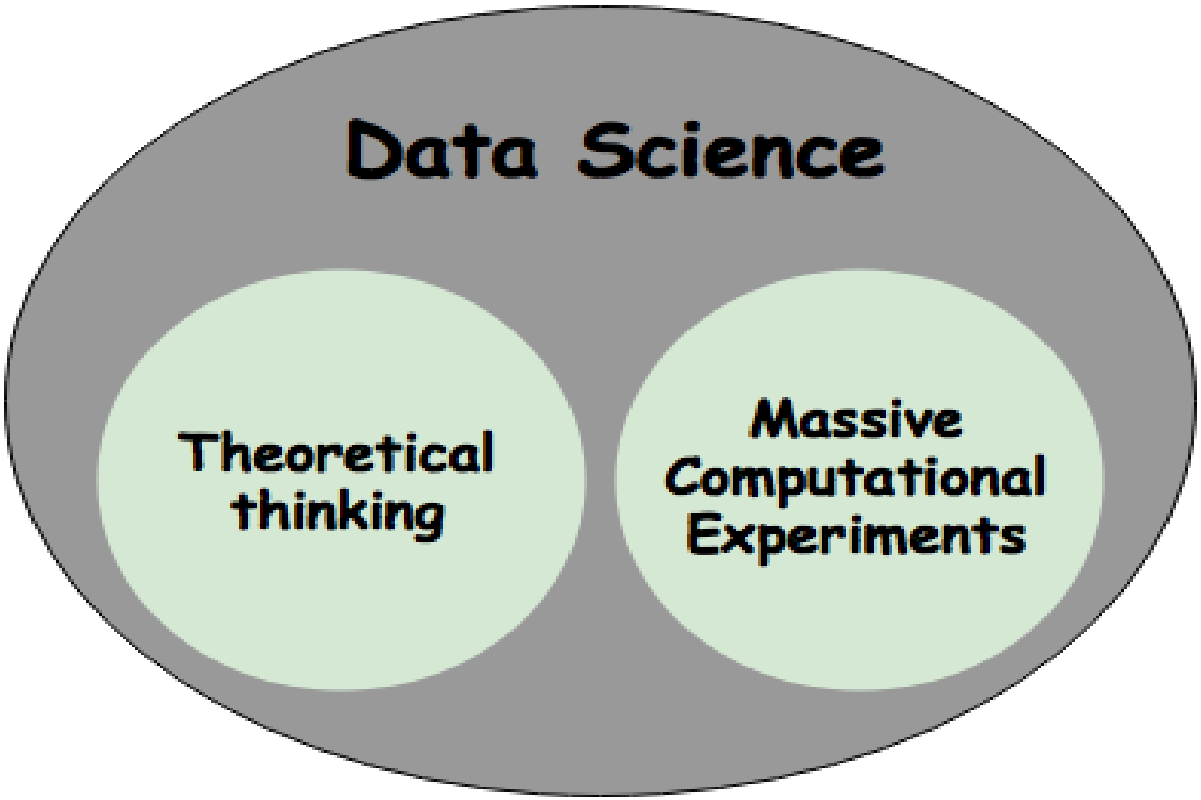
- Consumer desire to be online $\rightarrow$ IT enrichment
- Better IT $\rightarrow$ new ways to discover how things work (new science)

  > *"There is good authority for the prediction that within ten years a digital computer will be the world's chess champion, and that another will discover and prove an important new mathematical theorem" -- Harold J. Leavitt, Management in the 1980's (1958 article)*

- The new "hot science" can discover new things beyond the reach of theory!

What is this *New Hot Science?*

- it is *Data Science*!



- Data Science will transform Mathematics, Engineering, Medicine, Finance, Security, . . . *"literally all human life!*

Adapting to Computing Change

- Just as Climate Change demands adaptation,
- Computing Change demands adaptation:
  - **Psychological change** and rethinking of scientific values
  - **Pose** bold research **hypotheses** to settle computationally
  - **Design massive computing experiments**
  - **Adopt** painless computing **frameworks**
  - **Raise money** to pay for cloud-based computing
  - *Push Button*
- We describe one such framework today: **CJ**
  - In daily use at Stanford
  - Developed by Yours Truly.

Cluster Facilities at Stanford and Elsewhere

- Stanford offers cluster access through
  - Sherlock – 727 servers: 127 shared, 600 owned by faculty
  - Sherlock2 – new generation of Sherlock cluster
  - FarmShare – mostly for coursework and unsponsored research
- To gain access
  - Sponsoring faculty must email `research-computing-support@stanford.edu`
- Resources where you can learn more about clusters
  - Sherlock: `http://sherlock.stanford.edu`
  - FarmShare: `https://web.stanford.edu/group/farmshare/cgi-bin/wiki/`
  - XSEDE: `https://www.xsede.org`
  - TACC: `https://www.tacc.utexas.edu`
  - PSC: `https://www.psc.edu`
  - OSG: `https://www.opensciencegrid.org`

Outline

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

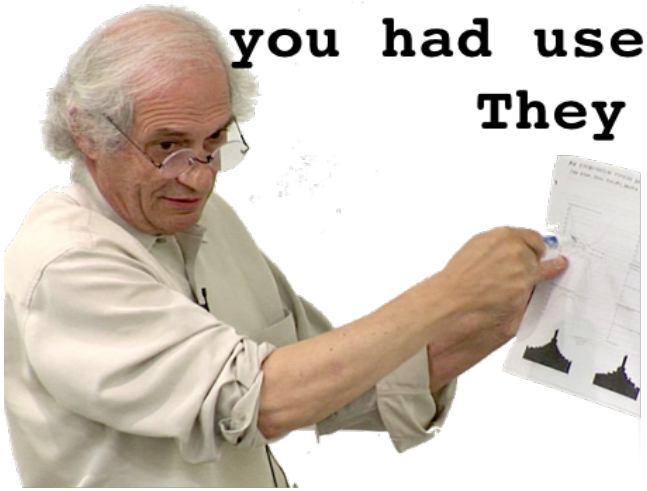
Create a shell script called wrap.sh:

```
#!/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

How to get rid of the pain?

We need to rethink the way we do computational experiments

What does an experiment involve?

In our telling, a computational experiment involves:

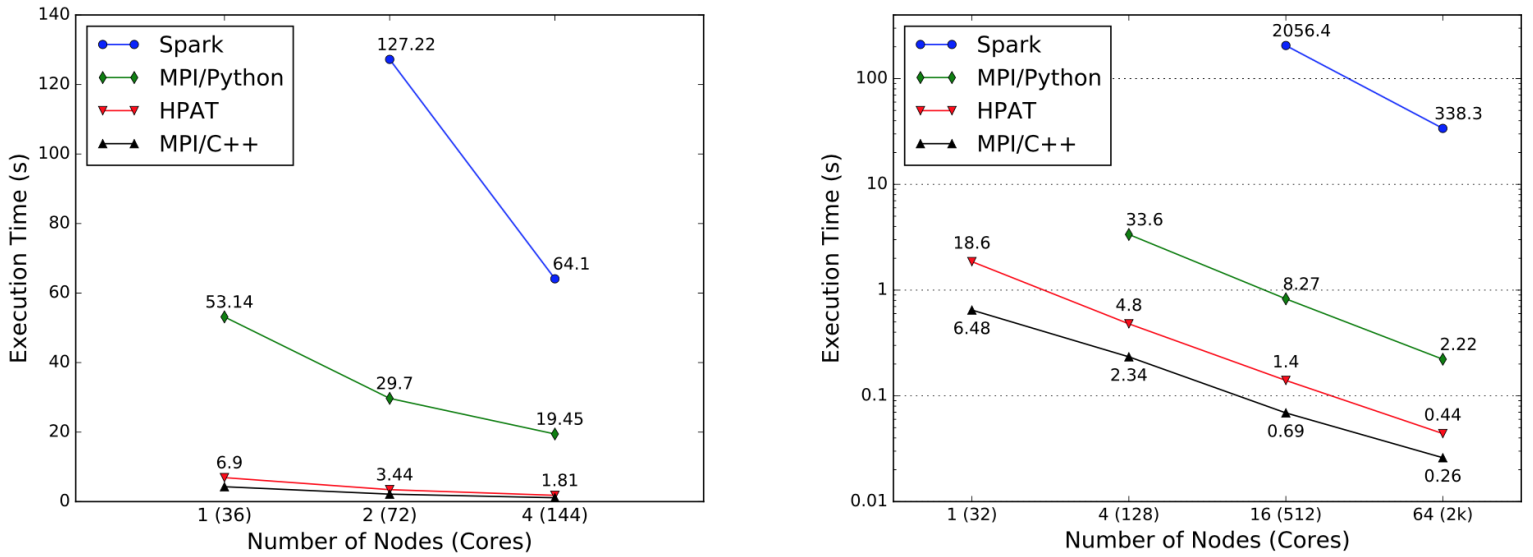
1. **Precise Specification** (define metric and parameters)
2. **Execution and management** of all the jobs
3. **Harvesting** of all the data generated by all the jobs
4. **Analysis** of the data
5. **Reporting** of results.

The painless computing paradigm should seamlessly integrate and automate all these 5 tasks

Desired features of a painless paradigm

- **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 a painless computing system

- **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 later how we can build such a system

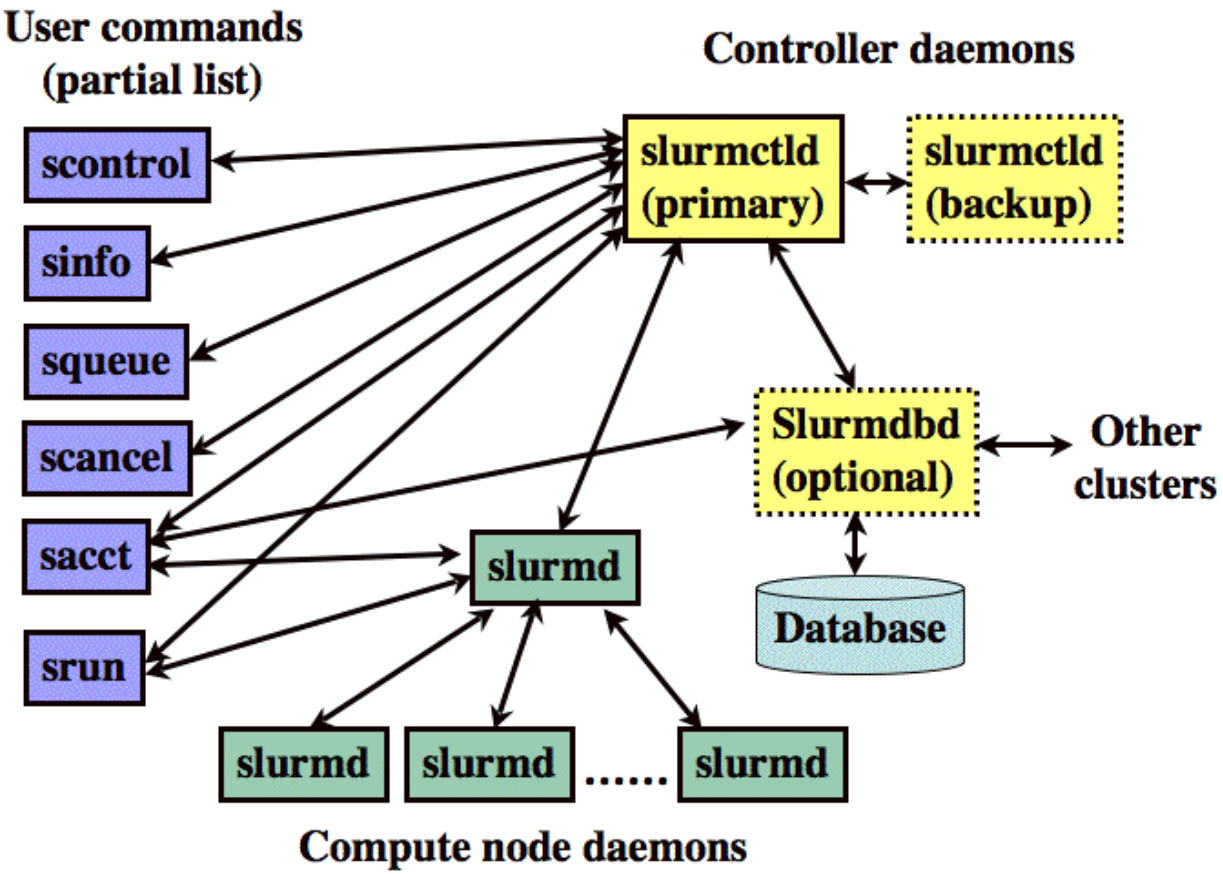
Outline

Some cluster terminology

- **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 Sherlock and Farmshare clusters at Stanford



SLURM architecture

Typical submission script

```
$sbatch bashMain.sh
Submitted batch job 15831
```

```
#!/bin/bash
#SBATCH --mem=8G
#SBATCH --time=48:00:00
#SBATCH --partition donoho

echo starting job  $SLURM_JOBID

module load matlab/R2016b

matlab -nosplash -nodisplay <<HERE
run('universality.m');
quit;
HERE

echo ended job $SLURM_JOBID
```

Working with a cluster

Bad habits:

- Repetitive interactive logging on to the cluster
- Manual copy of your codes and script
- Manually using $sbatch each time

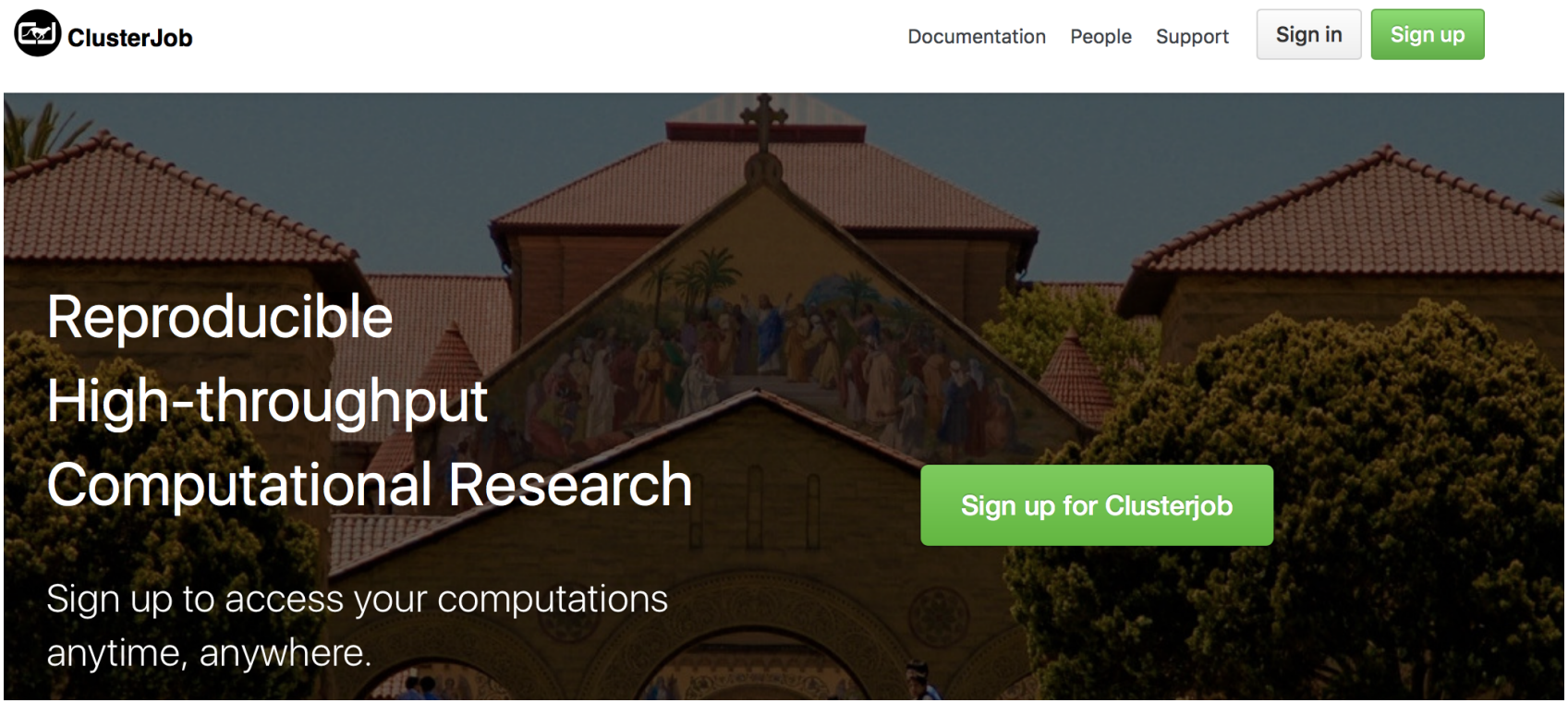
Good habits:

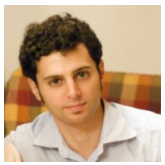
- Automating your activities
- Occasional logging on to the cluster

Let's see next how we can build an automation system

Outline

CJ stands for ClusterJob



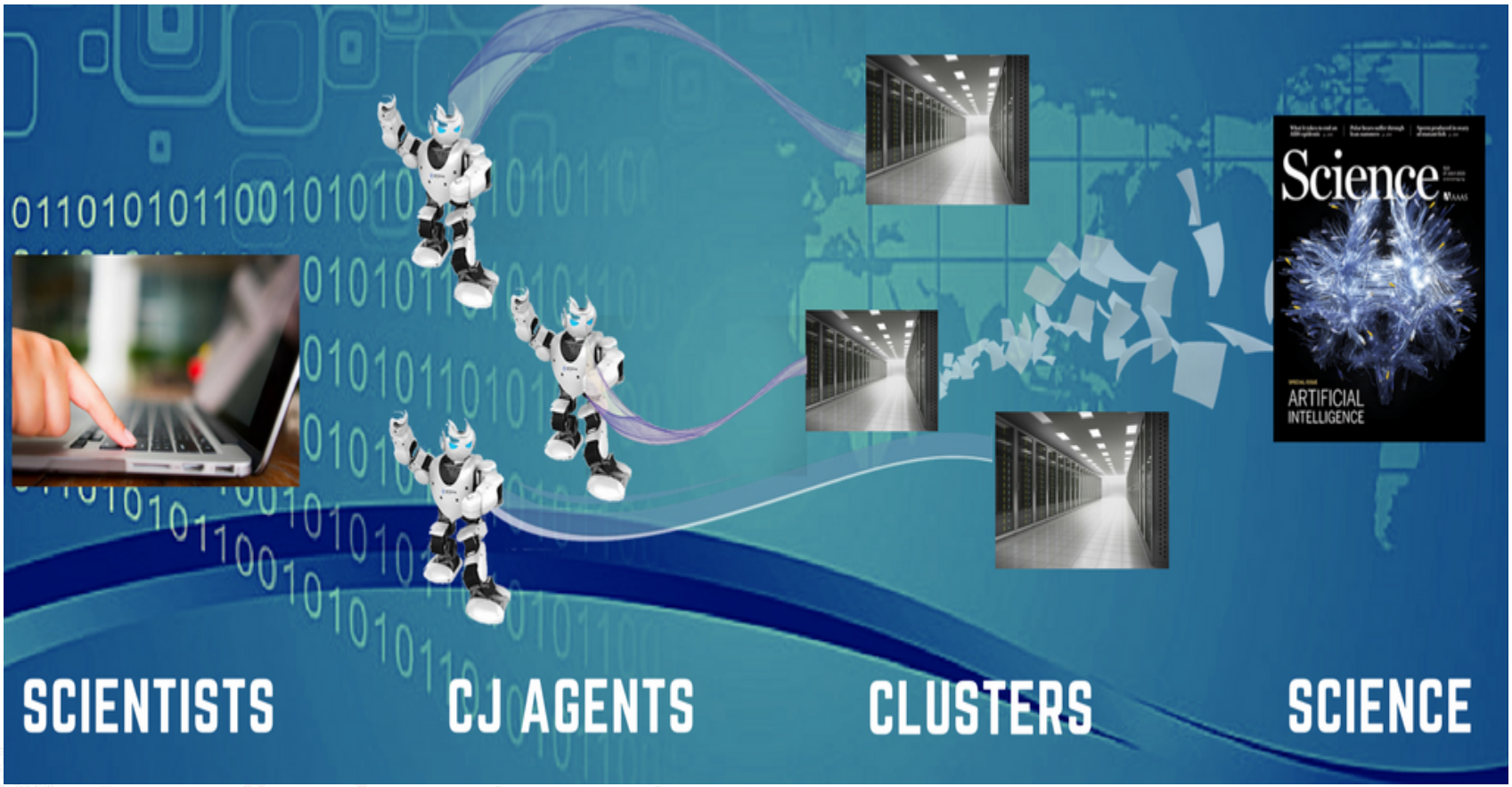


"This is how it [computation] should be done." – V. Morgenshtern

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

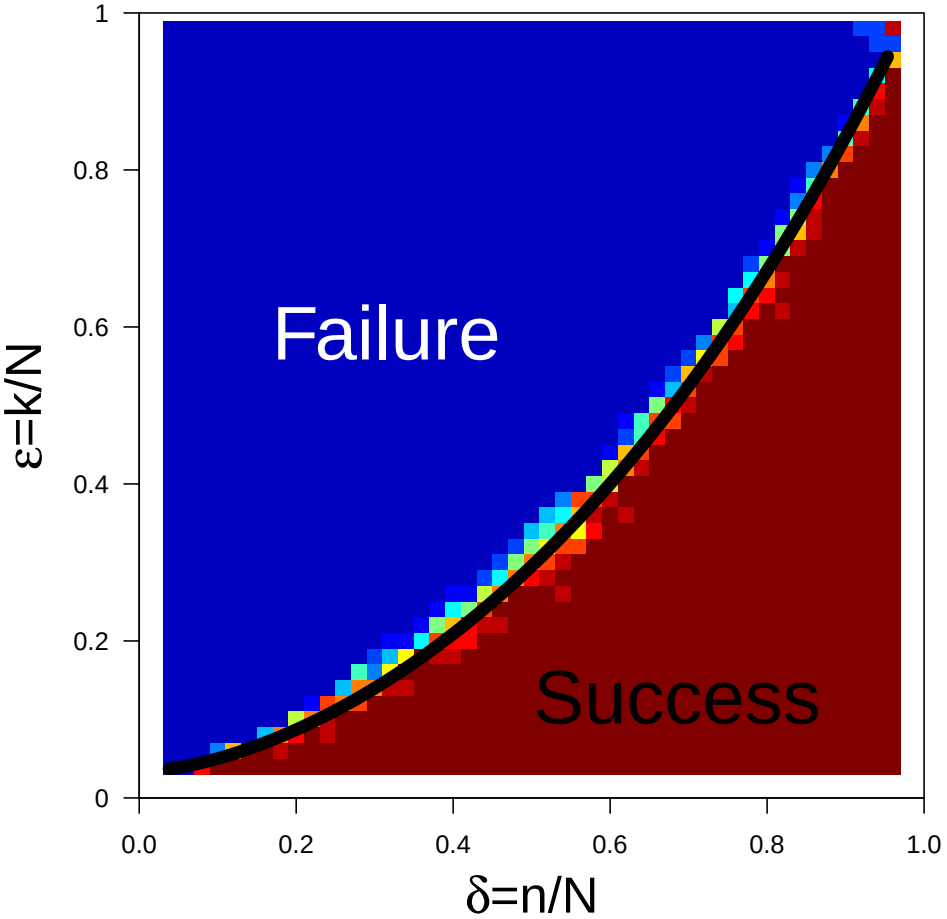
Our vision for CJ

- *push a button, fire and forget*
- *harvest, analyze and publish discovery*



How does CJ work, an example

Compressed Sensing Phase Transition Experiments:



How does CJ work, an example

Write a *simple and decipherable* MATLAB script:

```
% test.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
```

How does CJ work, an example

Let CJ handle the rest.

- Submit 441 separate jobs by a simple command

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

How does CJ work, an example

Let CJ handle the rest.

- 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

Let CJ handle the rest.

- 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: test.m
initial_flag: parrun

          Test PT
```

How does CJ work, an example

Let CJ handle the rest.

- Easily harvest results

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

How does CJ work, an example

Let CJ handle the rest.

- . . . and many more functionalities

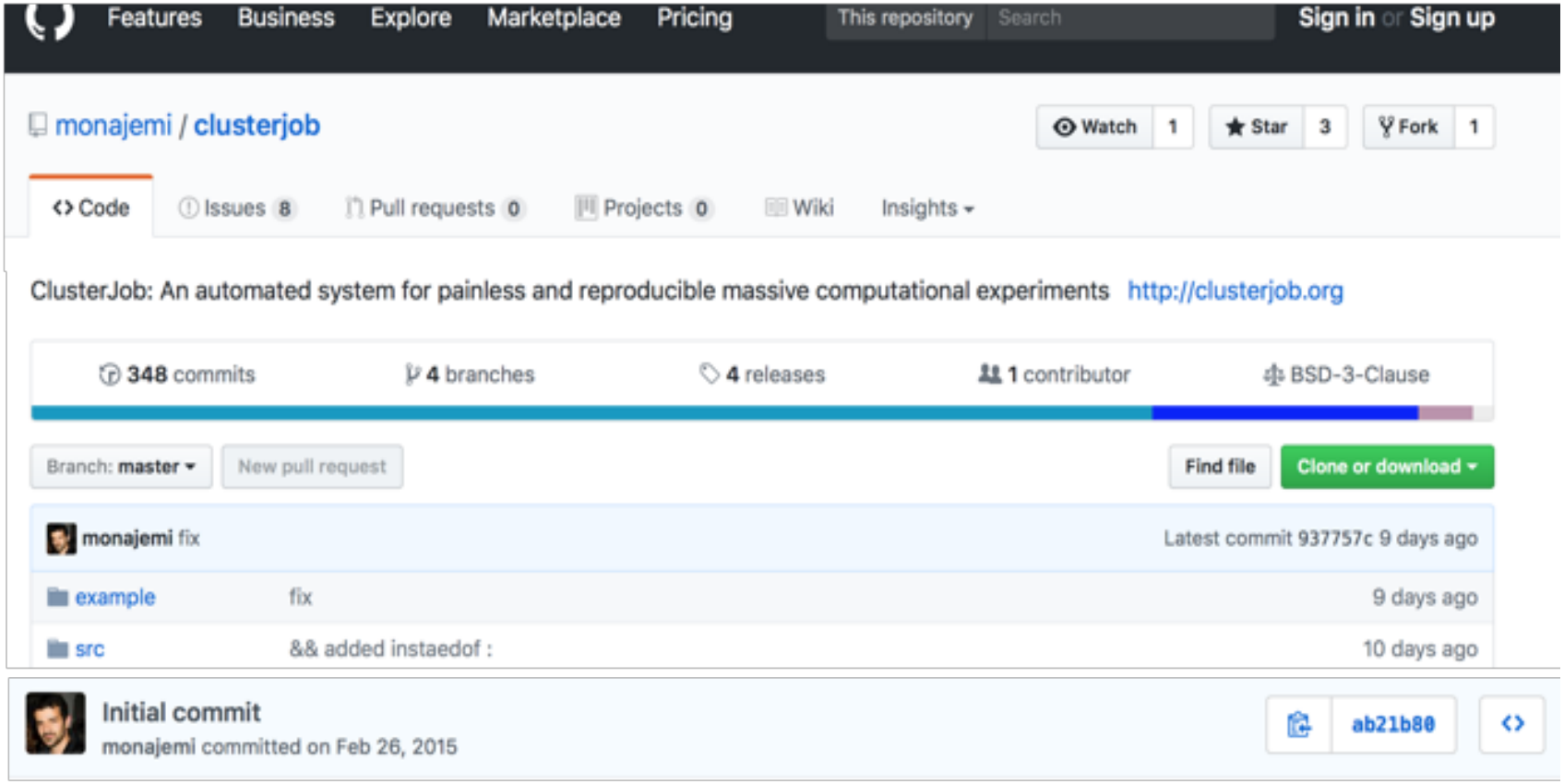
```
$ cj help
```

CJ demo

CJ project in more detail

Open source project:

`https://github.com/monajemi/clusterjob`



A look inside, core modules

CJ is written in Perl

clusterjob — -bash — 86×38

```
Hatefs-MacBook-Pro:clusterjob hatef$ ls .
CJlog       LICENSE     cj_config   example     ssh_config
INSTALL     README.md   dep.pl      src         todo
Hatefs-MacBook-Pro:clusterjob hatef$
Hatefs-MacBook-Pro:clusterjob hatef$
Hatefs-MacBook-Pro:clusterjob hatef$ ls src/
CJ              CJ.pl           CJ.pm           external        sanity_checks tmp
Hatefs-MacBook-Pro:clusterjob hatef$
Hatefs-MacBook-Pro:clusterjob hatef$
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/Scripts.pm
src/CJ/Sync.pm
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 — 96×35

```
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:torchvision:cuda80:scipy:matplotlib:torchvision:-c soumith
[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
[corn]


[rice]
Host            rice.stanford.edu
User            monajemi
```

What happens when you issue `parrun`?

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

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

# setup conda env for python
$self->setup_conda_venv($ssh) if($program_type eq 'python');

# 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'
```

Summary

- We are experiencing a *computing phase transition*
- Scientists need to adapt to this change by rethinking computing habits!
- CJ is an open-source software for *painless massive computing*
- You can use CJ for Matlab and Python jobs in your research.
- You can contribute to CJ project by forking it on GitHub.

Outline

1. 2017 Gear Up for Social Science Data Extravaganza
   Friday, October 27, 2017
   Get the full program at: bit.ly/slgearup
2. Homework

- *push a button, fire and forget*
- *harvest, analyze and publish breakthrough discovery*