
20 pt Gil Sans
32 pt Gil Sans
48 pt Gil Sans
64 pt Gil Sans
84 pt Gil Sans
16:9 TEST SLIDE

OCCUPY THE CLOUD

Distributed computing for the 99%

Eric Jonas
Postdoctoral Researcher
jonas@eecs.berkeley.edu
@stochastician

riselab UC Berkeley

Berkeley **C**enter for **C**omputational **I**maging

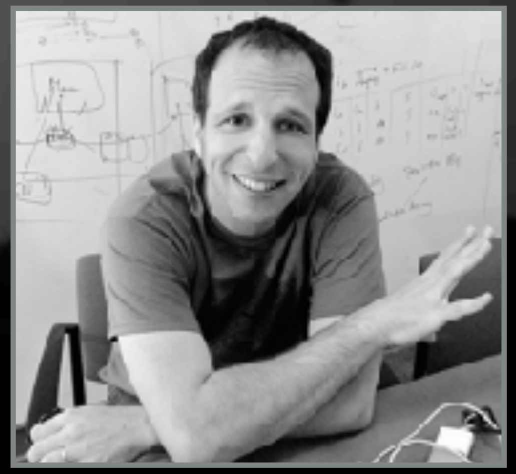

Qifan Pu


Shivaram Venkataraman


Vaishaal Shankar

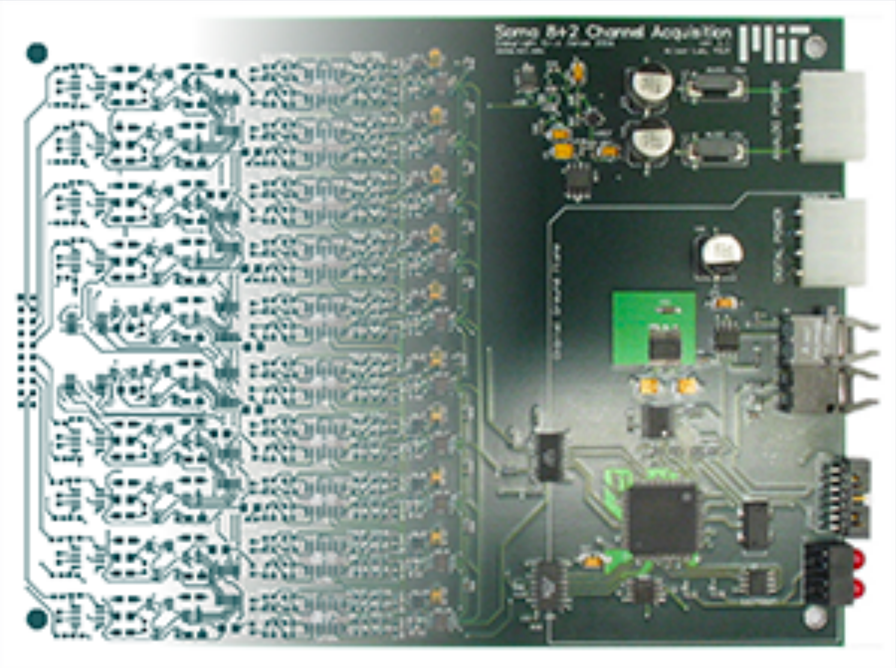

Allan Peng

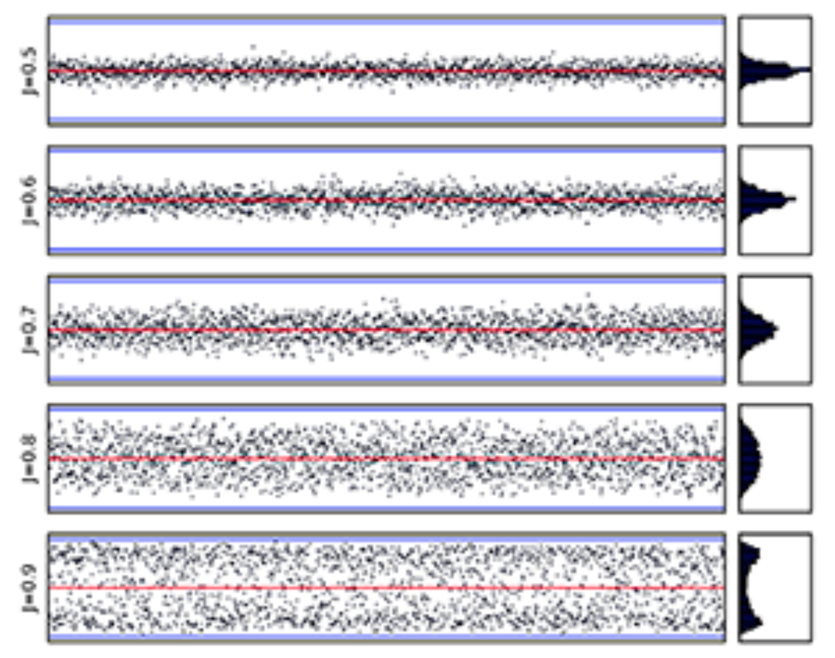

Ion Stoica

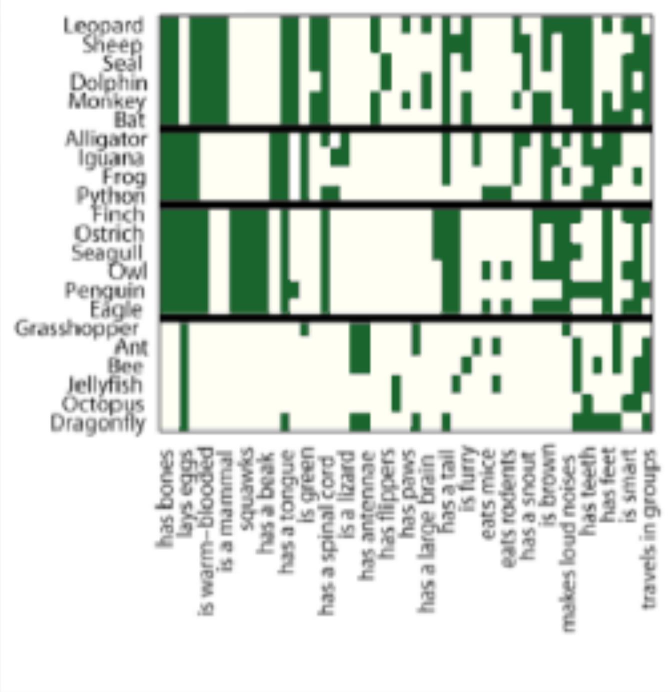

Ben Recht

ONCE UPON A TIME…

(my Markovian life decisions)

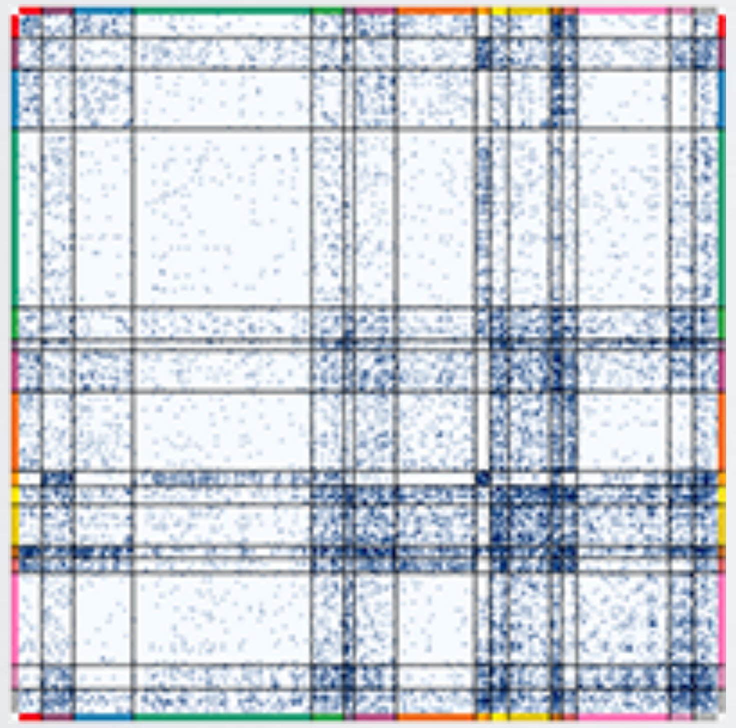


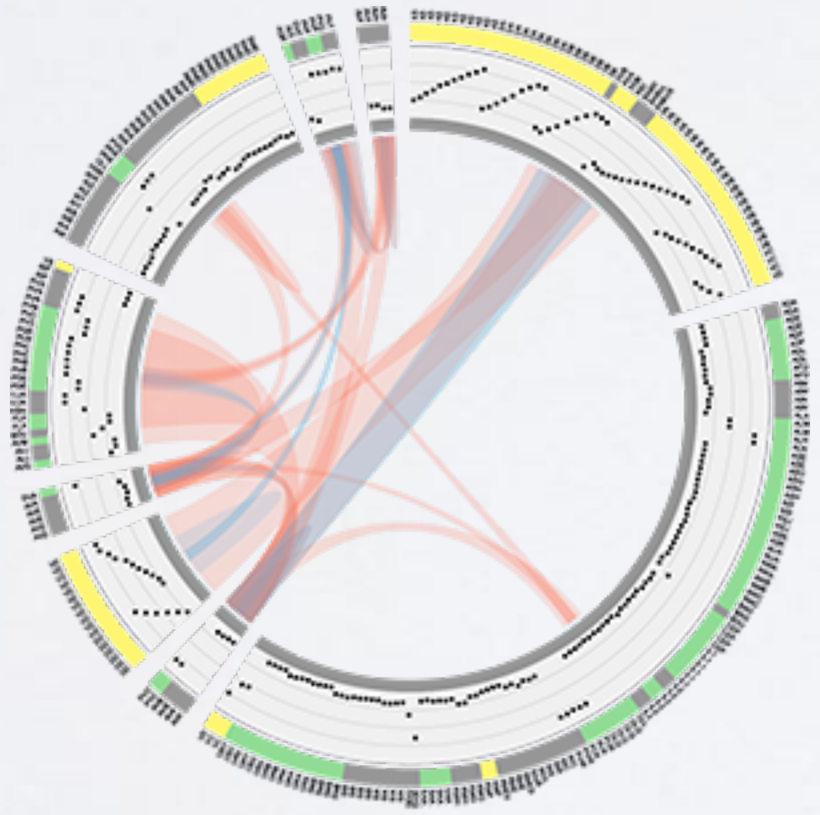


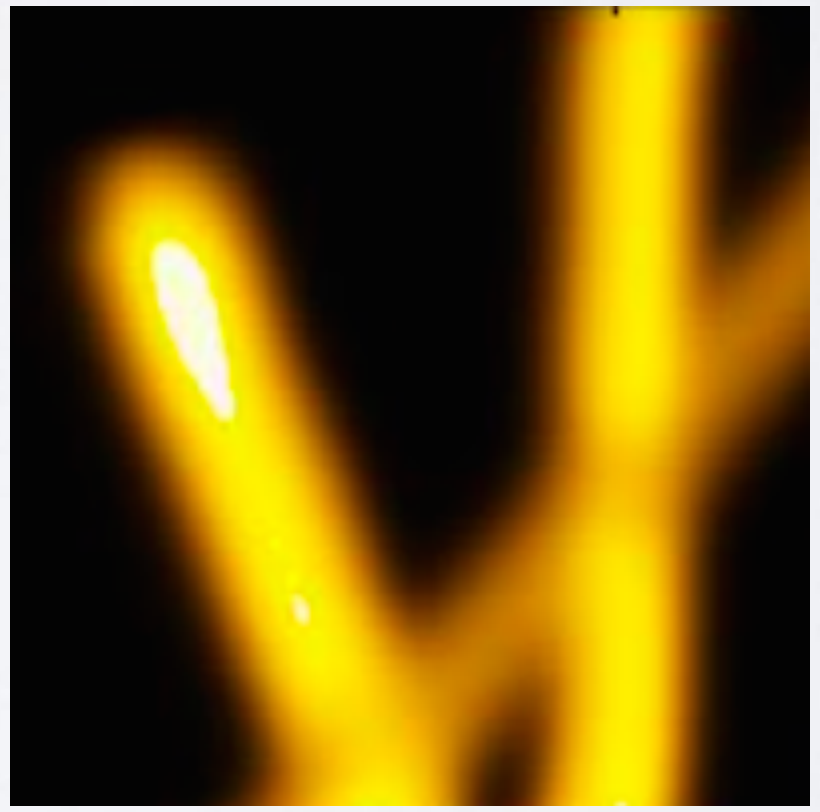


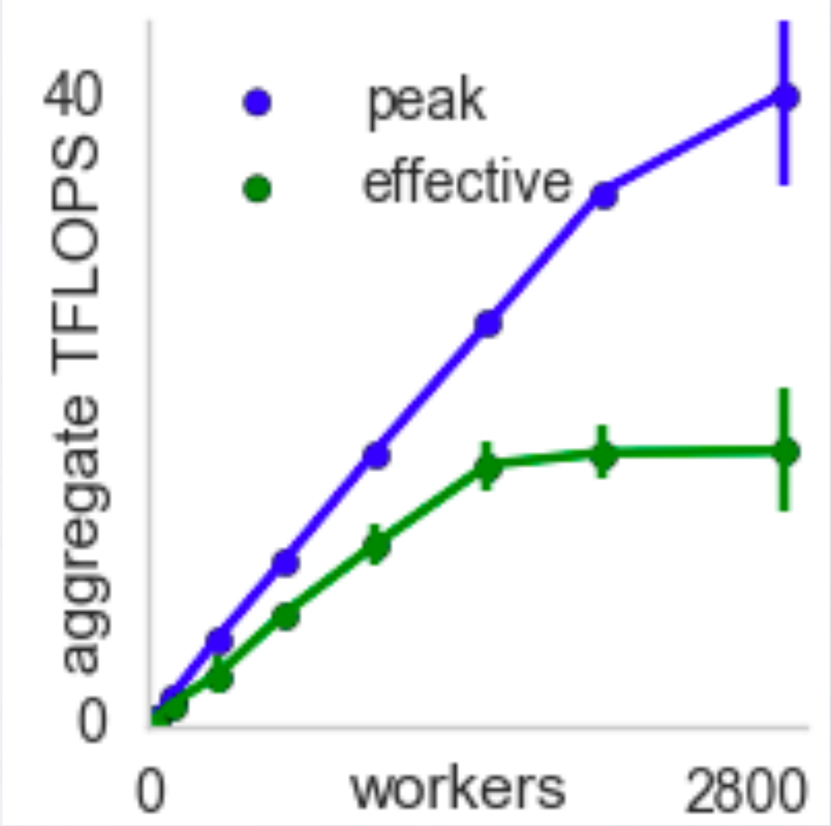
p(k)
prior knowledge, inc.

salesforce









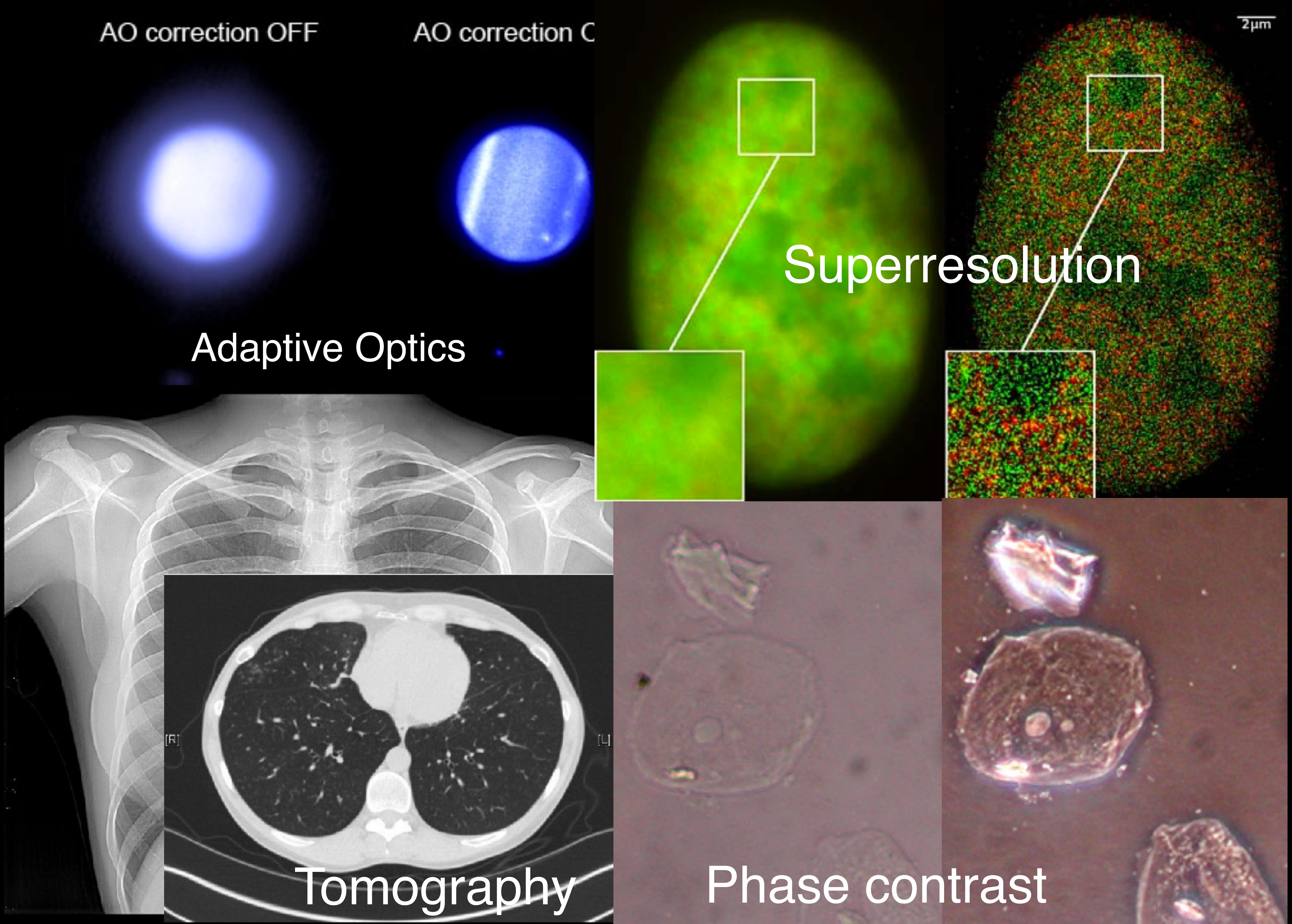
Grad school was embarrassing(-ly parallel)

“I hate computers”

–Eric Jonas, 2017

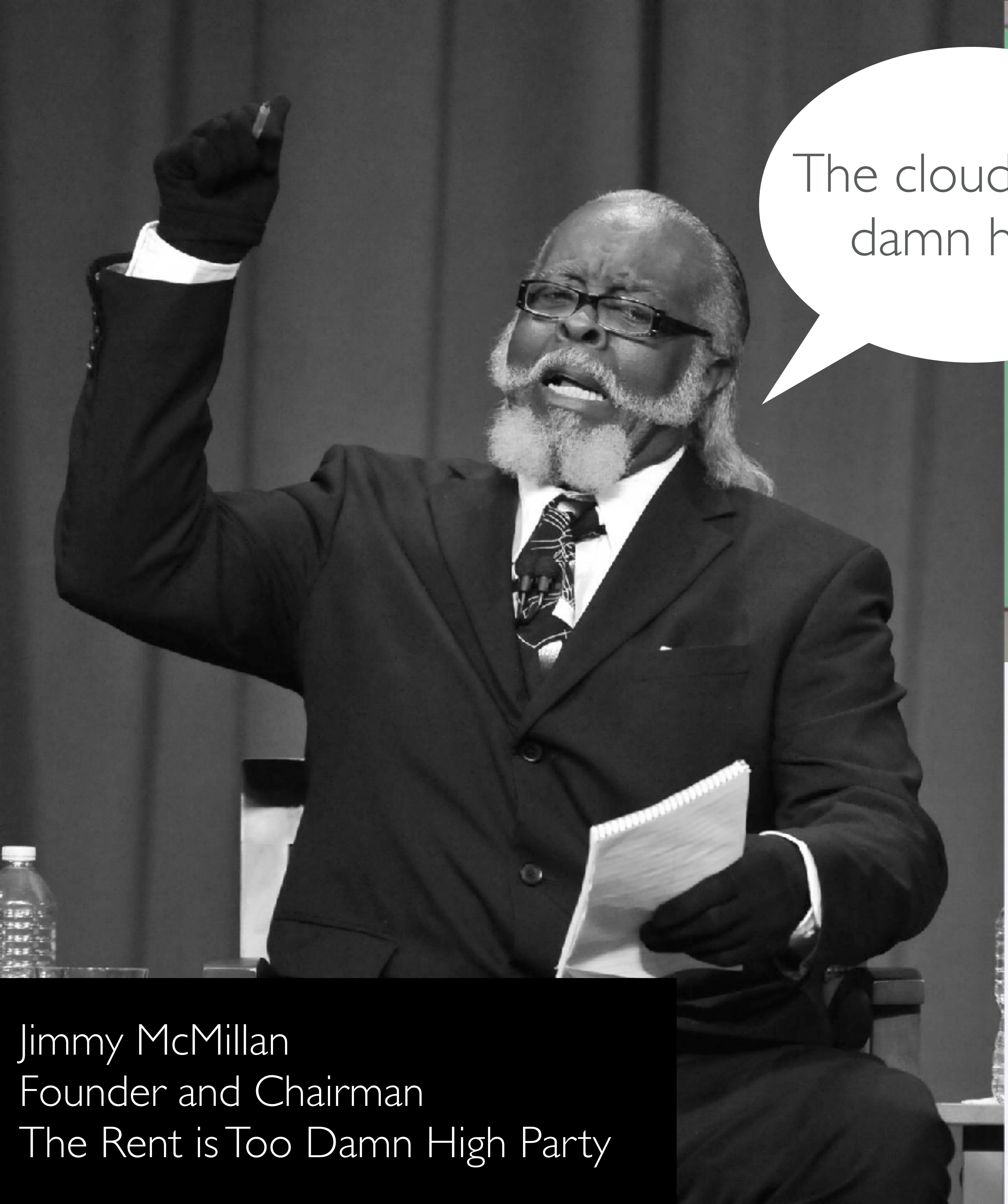
I'm interested in how computer science and machine learning can improve instrumentation and measurement

Inverse Problems	Signal processing
Compressed Sensing	Computational Imaging

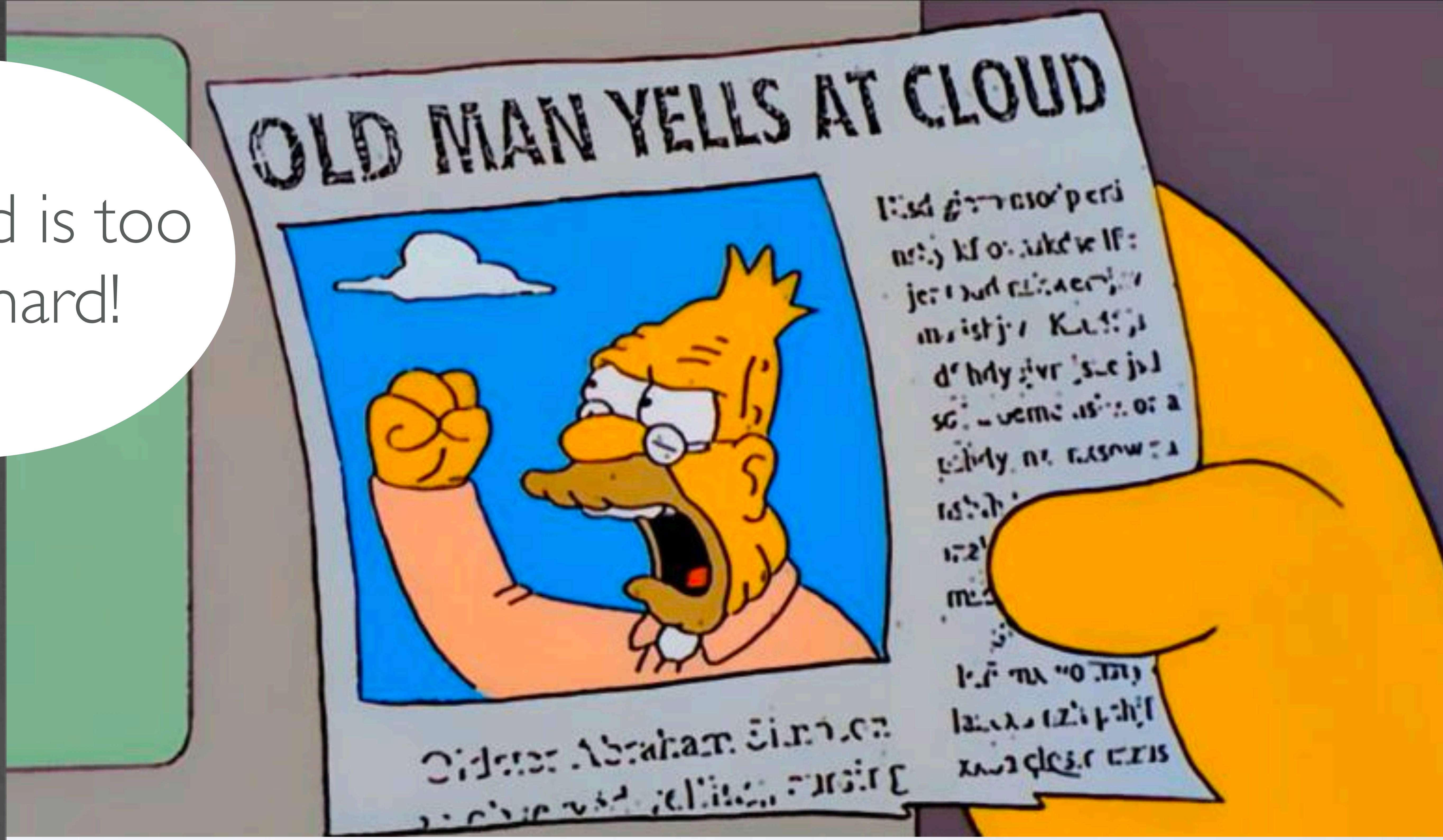

AO correction OFF
Adaptive Optics
2µm
Superresolution
[R]
[L]
Tomography
Phase contrast

PREVIOUSLY, AT COMP IMAGING LUNCH

Why is there no "cloud button"?



Jimmy McMillan
Founder and Chairman
The Rent is Too Damn High Party



Less than half of the graduate students in our group have ever written a Spark or Hadoop job

EC2Instances.info Easy Amazon EC2 Instance Comparison

EC2 RDS

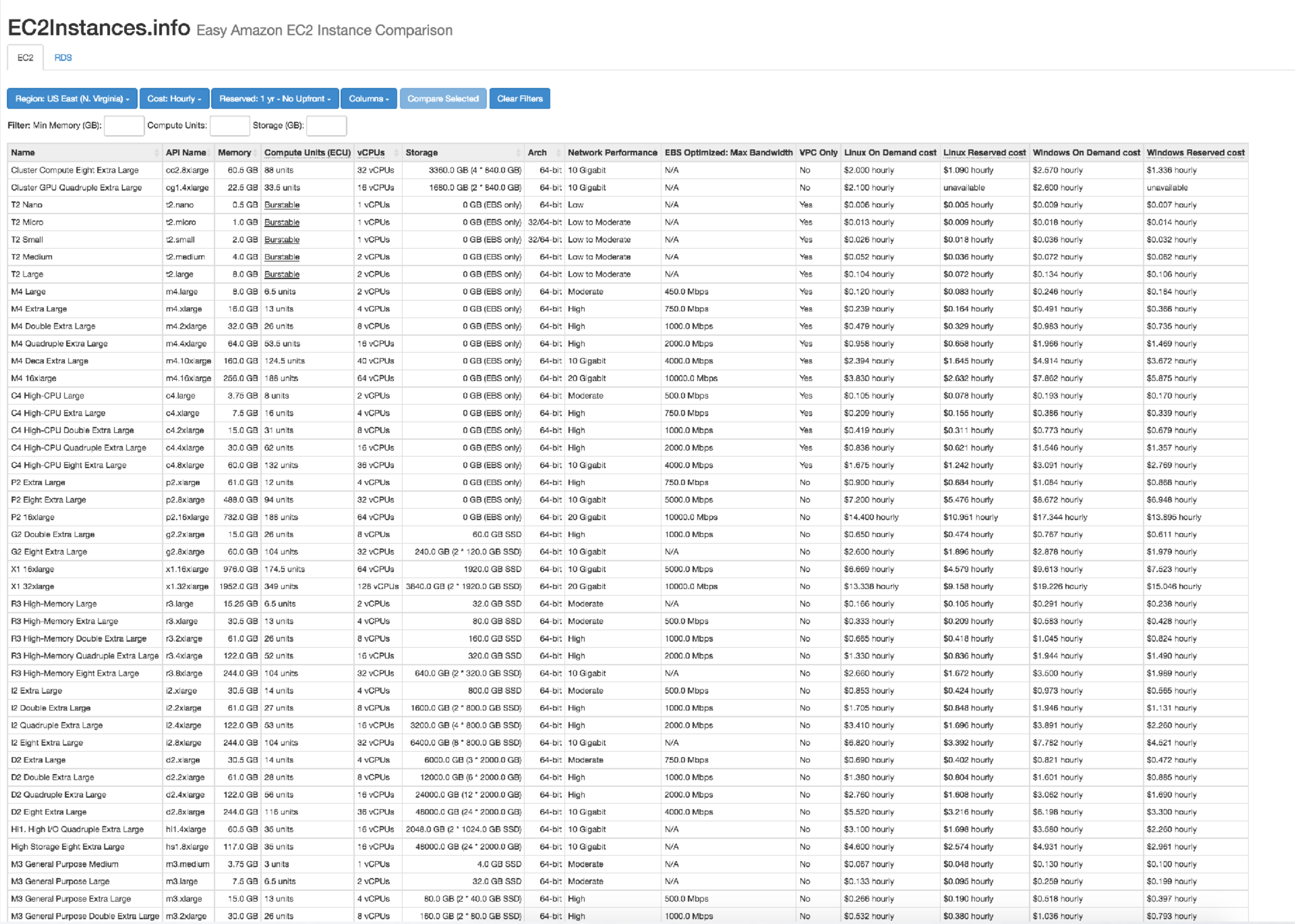
Region: US East (N. Virginia) - Cost: Hourly - Reserved: 1 yr - No Upfront - Columns - Compare Selected Clear Filters

Filter: Min Memory (GB): Compute Units: Storage (GB):

Name	API Name	Memory	Compute Units (ECU)	vCPUs	Storage	Arch	Network Performance	EBS Optimized: Max Bandwidth	VPC Only	Linux On Demand cost	Linux Reserved cost	Windows On Demand cost	Windows Reserved cost
Cluster Compute Eight Extra Large	cc2.8xlarge	60.5 GB	88 units	32 vCPUs	3360.0 GB (4 * 840.0 GB)	64-bit	10 Gigabit	N/A	No	$2.000 hourly	$1.090 hourly	$2.570 hourly	$1.336 hourly
Cluster GPU Quadruple Extra Large	cg1.4xlarge	22.5 GB	33.5 units	16 vCPUs	1680.0 GB (2 * 840.0 GB)	64-bit	10 Gigabit	N/A	No	$2.100 hourly	unavailable	$2.600 hourly	unavailable
T2 Nano	t2.nano	0.5 GB	Burstable	1 vCPUs	0 GB (EBS only)	64-bit	Low	N/A	Yes	$0.006 hourly	$0.005 hourly	$0.009 hourly	$0.007 hourly
T2 Micro	t2.micro	1.0 GB	Burstable	1 vCPUs	0 GB (EBS only)	32/64-bit	Low to Moderate	N/A	Yes	$0.013 hourly	$0.009 hourly	$0.018 hourly	$0.014 hourly
T2 Small	t2.small	2.0 GB	Burstable	1 vCPUs	0 GB (EBS only)	32/64-bit	Low to Moderate	N/A	Yes	$0.026 hourly	$0.018 hourly	$0.036 hourly	$0.032 hourly
T2 Medium	t2.medium	4.0 GB	Burstable	2 vCPUs	0 GB (EBS only)	64-bit	Low to Moderate	N/A	Yes	$0.052 hourly	$0.036 hourly	$0.072 hourly	$0.062 hourly
T2 Large	t2.large	8.0 GB	Burstable	2 vCPUs	0 GB (EBS only)	64-bit	Low to Moderate	N/A	Yes	$0.104 hourly	$0.072 hourly	$0.134 hourly	$0.106 hourly
M4 Large	m4.large	8.0 GB	6.5 units	2 vCPUs	0 GB (EBS only)	64-bit	Moderate	450.0 Mbps	Yes	$0.120 hourly	$0.083 hourly	$0.246 hourly	$0.184 hourly
M4 Extra Large	m4.xlarge	16.0 GB	13 units	4 vCPUs	0 GB (EBS only)	64-bit	High	750.0 Mbps	Yes	$0.239 hourly	$0.164 hourly	$0.491 hourly	$0.366 hourly
M4 Double Extra Large	m4.2xlarge	32.0 GB	26 units	8 vCPUs	0 GB (EBS only)	64-bit	High	1000.0 Mbps	Yes	$0.479 hourly	$0.329 hourly	$0.983 hourly	$0.735 hourly
M4 Quadruple Extra Large	m4.4xlarge	64.0 GB	53.5 units	16 vCPUs	0 GB (EBS only)	64-bit	High	2000.0 Mbps	Yes	$0.958 hourly	$0.658 hourly	$1.966 hourly	$1.469 hourly
M4 Deca Extra Large	m4.10xlarge	160.0 GB	124.5 units	40 vCPUs	0 GB (EBS only)	64-bit	10 Gigabit	4000.0 Mbps	Yes	$2.394 hourly	$1.645 hourly	$4.914 hourly	$3.672 hourly
M4 16xlarge	m4.16xlarge	256.0 GB	188 units	64 vCPUs	0 GB (EBS only)	64-bit	20 Gigabit	10000.0 Mbps	Yes	$3.830 hourly	$2.632 hourly	$7.862 hourly	$5.875 hourly
C4 High-CPU Large	c4.large	3.75 GB	8 units	2 vCPUs	0 GB (EBS only)	64-bit	Moderate	500.0 Mbps	Yes	$0.105 hourly	$0.078 hourly	$0.193 hourly	$0.170 hourly
C4 High-CPU Extra Large	c4.xlarge	7.5 GB	16 units	4 vCPUs	0 GB (EBS only)	64-bit	High	750.0 Mbps	Yes	$0.209 hourly	$0.155 hourly	$0.386 hourly	$0.339 hourly
C4 High-CPU Double Extra Large	c4.2xlarge	15.0 GB	31 units	8 vCPUs	0 GB (EBS only)	64-bit	High	1000.0 Mbps	Yes	$0.419 hourly	$0.311 hourly	$0.773 hourly	$0.679 hourly
C4 High-CPU Quadruple Extra Large	c4.4xlarge	30.0 GB	62 units	16 vCPUs	0 GB (EBS only)	64-bit	High	2000.0 Mbps	Yes	$0.838 hourly	$0.621 hourly	$1.546 hourly	$1.357 hourly
C4 High-CPU Eight Extra Large	c4.8xlarge	60.0 GB	132 units	36 vCPUs	0 GB (EBS only)	64-bit	10 Gigabit	4000.0 Mbps	Yes	$1.675 hourly	$1.242 hourly	$3.091 hourly	$2.769 hourly
P2 Extra Large	p2.xlarge	61.0 GB	12 units	4 vCPUs	0 GB (EBS only)	64-bit	High	750.0 Mbps	No	$0.900 hourly	$0.684 hourly	$1.084 hourly	$0.868 hourly
P2 Eight Extra Large	p2.8xlarge	488.0 GB	94 units	32 vCPUs	0 GB (EBS only)	64-bit	10 Gigabit	5000.0 Mbps	No	$7.200 hourly	$5.476 hourly	$8.672 hourly	$6.948 hourly
P2 16xlarge	p2.16xlarge	732.0 GB	188 units	64 vCPUs	0 GB (EBS only)	64-bit	20 Gigabit	10000.0 Mbps	No	$14.400 hourly	$10.951 hourly	$17.344 hourly	$13.895 hourly
G2 Double Extra Large	g2.2xlarge	15.0 GB	26 units	8 vCPUs	60.0 GB SSD	64-bit	High	1000.0 Mbps	No	$0.650 hourly	$0.474 hourly	$0.767 hourly	$0.611 hourly
G2 Eight Extra Large	g2.8xlarge	60.0 GB	104 units	32 vCPUs	240.0 GB (2 * 120.0 GB SSD)	64-bit	10 Gigabit	N/A	No	$2.600 hourly	$1.896 hourly	$2.878 hourly	$1.979 hourly
X1 16xlarge	x1.16xlarge	976.0 GB	174.5 units	64 vCPUs	1920.0 GB SSD	64-bit	10 Gigabit	5000.0 Mbps	No	$6.669 hourly	$4.579 hourly	$9.613 hourly	$7.523 hourly
X1 32xlarge	x1.32xlarge	1952.0 GB	349 units	128 vCPUs	3840.0 GB (2 * 1920.0 GB SSD)	64-bit	20 Gigabit	10000.0 Mbps	No	$13.338 hourly	$9.158 hourly	$19.226 hourly	$15.046 hourly
R3 High-Memory Large	r3.large	15.25 GB	6.5 units	2 vCPUs	32.0 GB SSD	64-bit	Moderate	N/A	No	$0.166 hourly	$0.105 hourly	$0.291 hourly	$0.238 hourly
R3 High-Memory Extra Large	r3.xlarge	30.5 GB	13 units	4 vCPUs	80.0 GB SSD	64-bit	Moderate	500.0 Mbps	No	$0.333 hourly	$0.209 hourly	$0.583 hourly	$0.428 hourly
R3 High-Memory Double Extra Large	r3.2xlarge	61.0 GB	26 units	8 vCPUs	160.0 GB SSD	64-bit	High	1000.0 Mbps	No	$0.665 hourly	$0.418 hourly	$1.045 hourly	$0.824 hourly
R3 High-Memory Quadruple Extra Large	r3.4xlarge	122.0 GB	52 units	16 vCPUs	320.0 GB SSD	64-bit	High	2000.0 Mbps	No	$1.330 hourly	$0.836 hourly	$1.944 hourly	$1.490 hourly
R3 High-Memory Eight Extra Large	r3.8xlarge	244.0 GB	104 units	32 vCPUs	640.0 GB (2 * 320.0 GB SSD)	64-bit	10 Gigabit	N/A	No	$2.660 hourly	$1.672 hourly	$3.500 hourly	$1.989 hourly
I2 Extra Large	i2.xlarge	30.5 GB	14 units	4 vCPUs	800.0 GB SSD	64-bit	Moderate	500.0 Mbps	No	$0.853 hourly	$0.424 hourly	$0.973 hourly	$0.565 hourly
I2 Double Extra Large	i2.2xlarge	61.0 GB	27 units	8 vCPUs	1600.0 GB (2 * 800.0 GB SSD)	64-bit	High	1000.0 Mbps	No	$1.705 hourly	$0.848 hourly	$1.946 hourly	$1.131 hourly
I2 Quadruple Extra Large	i2.4xlarge	122.0 GB	53 units	16 vCPUs	3200.0 GB (4 * 800.0 GB SSD)	64-bit	High	2000.0 Mbps	No	$3.410 hourly	$1.696 hourly	$3.891 hourly	$2.260 hourly

#THECLOUDISTOODAMNHARD

- What type? what instance? What base image?
- How many to spin up? What price? spot?
- wait, Wait, WAIT oh god
- now what? DEVOPS



EC2Instances.info Easy Amazon EC2 Instance Comparison

EC2 RDS

Region: US East (N. Virginia) - | Cost: Hourly - | Reserved: 1 yr - No Upfront - | Columns - | Compare Selected | Clear Filters

Filter: Min Memory (GB): Compute Units: Storage (GB):

Name	API Name	Memory	Compute Units (ECU)	vCPUs	Storage	Arch	Network Performance	EBS Optimized: Max Bandwidth	VPC Only	Linux On Demand cost	Linux Reserved cost	Windows On Demand cost	Windows Reserved cost
Cluster Compute Eight Extra Large	cc2.8xlarge	60.5 GB	88 units	32 vCPUs	3360.0 GB (4 * 840.0 GB)	64-bit	10 Gigabit	N/A	No	$2.000 hourly	$1.090 hourly	$2.570 hourly	$1.336 hourly
Cluster GPU Quadruple Extra Large	cg1.4xlarge	22.5 GB	33.5 units	16 vCPUs	1680.0 GB (2 * 840.0 GB)	64-bit	10 Gigabit	N/A	No	$2.100 hourly	unavailable	$2.600 hourly	unavailable
T2 Nano	t2.nano	0.5 GB	Burstable	1 vCPUs	0 GB (EBS only)	64-bit	Low	N/A	Yes	$0.006 hourly	$0.005 hourly	$0.009 hourly	$0.007 hourly
T2 Micro	t2.micro	1.0 GB	Burstable	1 vCPUs	0 GB (EBS only)	32/64-bit	Low to Moderate	N/A	Yes	$0.013 hourly	$0.009 hourly	$0.018 hourly	$0.014 hourly
T2 Small	t2.small	2.0 GB	Burstable	1 vCPUs	0 GB (EBS only)	32/64-bit	Low to Moderate	N/A	Yes	$0.026 hourly	$0.018 hourly	$0.036 hourly	$0.032 hourly
T2 Medium	t2.medium	4.0 GB	Burstable	2 vCPUs	0 GB (EBS only)	64-bit	Low to Moderate	N/A	Yes	$0.052 hourly	$0.036 hourly	$0.072 hourly	$0.062 hourly
T2 Large	t2.large	8.0 GB	Burstable	2 vCPUs	0 GB (EBS only)	64-bit	Low to Moderate	N/A	Yes	$0.104 hourly	$0.072 hourly	$0.134 hourly	$0.106 hourly
M4 Large	m4.large	8.0 GB	6.5 units	2 vCPUs	0 GB (EBS only)	64-bit	Moderate	450.0 Mbps	Yes	$0.120 hourly	$0.083 hourly	$0.246 hourly	$0.184 hourly
M4 Extra Large	m4.xlarge	16.0 GB	13 units	4 vCPUs	0 GB (EBS only)	64-bit	High	750.0 Mbps	Yes	$0.239 hourly	$0.164 hourly	$0.491 hourly	$0.366 hourly
M4 Double Extra Large	m4.2xlarge	32.0 GB	26 units	8 vCPUs	0 GB (EBS only)	64-bit	High	1000.0 Mbps	Yes	$0.479 hourly	$0.329 hourly	$0.983 hourly	$0.735 hourly
M4 Quadruple Extra Large	m4.4xlarge	64.0 GB	53.5 units	16 vCPUs	0 GB (EBS only)	64-bit	High	2000.0 Mbps	Yes	$0.958 hourly	$0.658 hourly	$1.966 hourly	$1.469 hourly
M4 Deca Extra Large	m4.10xlarge	160.0 GB	124.5 units	40 vCPUs	0 GB (EBS only)	64-bit	10 Gigabit	4000.0 Mbps	Yes	$2.394 hourly	$1.645 hourly	$4.914 hourly	$3.672 hourly
M4 16xlarge	m4.16xlarge	256.0 GB	188 units	64 vCPUs	0 GB (EBS only)	64-bit	20 Gigabit	10000.0 Mbps	Yes	$3.830 hourly	$2.632 hourly	$7.862 hourly	$5.875 hourly
C4 High-CPU Large	c4.large	3.75 GB	8 units	2 vCPUs	0 GB (EBS only)	64-bit	Moderate	500.0 Mbps	Yes	$0.105 hourly	$0.078 hourly	$0.193 hourly	$0.170 hourly
C4 High-CPU Extra Large	c4.xlarge	7.5 GB	16 units	4 vCPUs	0 GB (EBS only)	64-bit	High	750.0 Mbps	Yes	$0.209 hourly	$0.155 hourly	$0.386 hourly	$0.339 hourly
C4 High-CPU Double Extra Large	c4.2xlarge	15.0 GB	31 units	8 vCPUs	0 GB (EBS only)	64-bit	High	1000.0 Mbps	Yes	$0.419 hourly	$0.311 hourly	$0.773 hourly	$0.679 hourly
C4 High-CPU Quadruple Extra Large	c4.4xlarge	30.0 GB	62 units	16 vCPUs	0 GB (EBS only)	64-bit	High	2000.0 Mbps	Yes	$0.838 hourly	$0.621 hourly	$1.546 hourly	$1.357 hourly
C4 High-CPU Eight Extra Large	c4.8xlarge	60.0 GB	132 units	36 vCPUs	0 GB (EBS only)	64-bit	10 Gigabit	4000.0 Mbps	Yes	$1.675 hourly	$1.242 hourly	$3.091 hourly	$2.769 hourly
P2 Extra Large	p2.xlarge	61.0 GB	12 units	4 vCPUs	0 GB (EBS only)	64-bit	High	750.0 Mbps	No	$0.900 hourly	$0.684 hourly	$1.084 hourly	$0.868 hourly
P2 Eight Extra Large	p2.8xlarge	488.0 GB	94 units	32 vCPUs	0 GB (EBS only)	64-bit	10 Gigabit	5000.0 Mbps	No	$7.200 hourly	$5.476 hourly	$8.672 hourly	$6.948 hourly
P2 16xlarge	p2.16xlarge	732.0 GB	188 units	64 vCPUs	0 GB (EBS only)	64-bit	20 Gigabit	10000.0 Mbps	No	$14.400 hourly	$10.951 hourly	$17.344 hourly	$13.895 hourly
G2 Double Extra Large	g2.2xlarge	15.0 GB	26 units	8 vCPUs	60.0 GB SSD	64-bit	High	1000.0 Mbps	No	$0.650 hourly	$0.474 hourly	$0.767 hourly	$0.611 hourly
G2 Eight Extra Large	g2.8xlarge	60.0 GB	104 units	32 vCPUs	240.0 GB (2 * 120.0 GB SSD)	64-bit	10 Gigabit	N/A	No	$2.600 hourly	$1.896 hourly	$2.878 hourly	$1.979 hourly
X1 16xlarge	x1.16xlarge	976.0 GB	174.5 units	64 vCPUs	1920.0 GB SSD	64-bit	10 Gigabit	5000.0 Mbps	No	$6.669 hourly	$4.579 hourly	$9.613 hourly	$7.523 hourly
X1 32xlarge	x1.32xlarge	1952.0 GB	349 units	128 vCPUs	3840.0 GB (2 * 1920.0 GB SSD)	64-bit	20 Gigabit	10000.0 Mbps	No	$13.338 hourly	$9.158 hourly	$19.226 hourly	$15.046 hourly
R3 High-Memory Large	r3.large	15.25 GB	6.5 units	2 vCPUs	32.0 GB SSD	64-bit	Moderate	N/A	No	$0.166 hourly	$0.105 hourly	$0.291 hourly	$0.238 hourly
R3 High-Memory Extra Large	r3.xlarge	30.5 GB	13 units	4 vCPUs	80.0 GB SSD	64-bit	Moderate	500.0 Mbps	No	$0.333 hourly	$0.209 hourly	$0.583 hourly	$0.428 hourly
R3 High-Memory Double Extra Large	r3.2xlarge	61.0 GB	26 units	8 vCPUs	160.0 GB SSD	64-bit	High	1000.0 Mbps	No	$0.665 hourly	$0.418 hourly	$1.045 hourly	$0.824 hourly
R3 High-Memory Quadruple Extra Large	r3.4xlarge	122.0 GB	52 units	16 vCPUs	320.0 GB SSD	64-bit	High	2000.0 Mbps	No	$1.330 hourly	$0.836 hourly	$1.944 hourly	$1.490 hourly
R3 High-Memory Eight Extra Large	r3.8xlarge	244.0 GB	104 units	32 vCPUs	640.0 GB (2 * 320.0 GB SSD)	64-bit	10 Gigabit	N/A	No	$2.660 hourly	$1.672 hourly	$3.500 hourly	$1.989 hourly
I2 Extra Large	i2.xlarge	30.5 GB	14 units	4 vCPUs	800.0 GB SSD	64-bit	Moderate	500.0 Mbps	No	$0.853 hourly	$0.424 hourly	$0.973 hourly	$0.565 hourly
I2 Double Extra Large	i2.2xlarge	61.0 GB	27 units	8 vCPUs	1600.0 GB (2 * 800.0 GB SSD)	64-bit	High	1000.0 Mbps	No	$1.705 hourly	$0.848 hourly	$1.946 hourly	$1.131 hourly
I2 Quadruple Extra Large	i2.4xlarge	122.0 GB	53 units	16 vCPUs	3200.0 GB (4 * 800.0 GB SSD)	64-bit	High	2000.0 Mbps	No	$3.410 hourly	$1.696 hourly	$3.891 hourly	$2.260 hourly
I2 Eight Extra Large	i2.8xlarge	244.0 GB	104 units	32 vCPUs	6400.0 GB (8 * 800.0 GB SSD)	64-bit	10 Gigabit	N/A	No	$6.820 hourly	$3.392 hourly	$7.782 hourly	$4.521 hourly
D2 Extra Large	d2.xlarge	30.5 GB	14 units	4 vCPUs	6000.0 GB (3 * 2000.0 GB)	64-bit	Moderate	750.0 Mbps	No	$0.690 hourly	$0.402 hourly	$0.821 hourly	$0.472 hourly
D2 Double Extra Large	d2.2xlarge	61.0 GB	28 units	8 vCPUs	12000.0 GB (6 * 2000.0 GB)	64-bit	High	1000.0 Mbps	No	$1.380 hourly	$0.804 hourly	$1.601 hourly	$0.885 hourly
D2 Quadruple Extra Large	d2.4xlarge	122.0 GB	56 units	16 vCPUs	24000.0 GB (12 * 2000.0 GB)	64-bit	High	2000.0 Mbps	No	$2.760 hourly	$1.608 hourly	$3.062 hourly	$1.690 hourly
D2 Eight Extra Large	d2.8xlarge	244.0 GB	116 units	36 vCPUs	48000.0 GB (24 * 2000.0 GB)	64-bit	10 Gigabit	4000.0 Mbps	No	$5.520 hourly	$3.216 hourly	$6.198 hourly	$3.300 hourly
HI1. High I/O Quadruple Extra Large	hi1.4xlarge	60.5 GB	35 units	16 vCPUs	2048.0 GB (2 * 1024.0 GB SSD)	64-bit	10 Gigabit	N/A	No	$3.100 hourly	$1.698 hourly	$3.580 hourly	$2.260 hourly
High Storage Eight Extra Large	hs1.8xlarge	117.0 GB	35 units	16 vCPUs	48000.0 GB (24 * 2000.0 GB)	64-bit	10 Gigabit	N/A	No	$4.600 hourly	$2.574 hourly	$4.931 hourly	$2.961 hourly
M3 General Purpose Medium	m3.medium	3.75 GB	3 units	1 vCPUs	4.0 GB SSD	64-bit	Moderate	N/A	No	$0.067 hourly	$0.048 hourly	$0.130 hourly	$0.100 hourly
M3 General Purpose Large	m3.large	7.5 GB	6.5 units	2 vCPUs	32.0 GB SSD	64-bit	Moderate	N/A	No	$0.133 hourly	$0.095 hourly	$0.259 hourly	$0.199 hourly
M3 General Purpose Extra Large	m3.xlarge	15.0 GB	13 units	4 vCPUs	80.0 GB (2 * 40.0 GB SSD)	64-bit	High	500.0 Mbps	No	$0.266 hourly	$0.190 hourly	$0.518 hourly	$0.397 hourly
M3 General Purpose Double Extra Large	m3.2xlarge	30.0 GB	26 units	8 vCPUs	160.0 GB (2 * 80.0 GB SSD)	64-bit	High	1000.0 Mbps	No	$0.532 hourly	$0.380 hourly	$1.036 hourly	$0.793 hourly

WHAT DO WE WANT?

1. Very little overhead for setup once someone has an AWS account. In particular, no persistent overhead -- you don't have to keep a large (expensive) cluster up and you don't have to wait 10+ min for a cluster to come up

WHAT DO WE WANT?

2. As close to zero overhead for users as possible
In particular, **anyone who can write python** should be able to invoke it through a reasonable interface. It should support all legacy code

WHAT DO WE WANT?

3. Target jobs that run in the **minutes-or-more regime**.

WHAT DO WE WANT?

4. I don't want to run a service.

That is, I personally don't want to offer the front-end for other people to use, rather, I want to directly pay AWS.

WHAT DO WE WANT?

5. It has to be from a **cloud player that's likely to give out an academic grant** -- AWS, Google, MS Azure.

There are startups in this space that might build cool technology, but often don't want to be paid in AWS research credits.

ORIGINAL DESIGN GOALS

1. **Very little overhead for setup** once someone has an AWS account. In particular, no persistent overhead -- you don't have to keep a large (expensive) cluster up and you don't have to wait 10+ min for a cluster to come up

2. As close to zero overhead for users as possible -- in particular, **anyone who can write python** should be able to invoke it through a reasonable interface.

3. Target jobs that run in the **minutes-or-more regime**.

4. **I don't want to run a service**. That is, I personally don't want to offer the front-end for other people to use, rather, I want to directly pay AWS.

5. It has to be from a **cloud player that's likely to give out an academic grant** -- AWS, Google, Azure. There are startups in this space that might build cool technology, but often don't want to be paid in AWS research credits.

servers

"I hate ~~computers~~"

–Eric Jonas, 2017

(condor)

"Most wrens are small and rather inconspicuous, except for their loud and often complex songs."

WHAT IS PYWREN

Research	Tool
Exploiting **real-time** elastic **execution**	Building a "**cloud button**"
How do systems change when you have **real-time** access to 10,000 stateless cores in <1 sec?	How can we bring the benefits of elastic compute to underserved audiences?

THE API

The most important primitive:

```
map(function, data)
```

and… th

```
def myfunc(x):
   return x + 1

futures = pwex.map(myfunc,

print pywren.get_all_resul

[2, 3, 4]
```

```
import pywren
import numpy as np

def addone(x):
    return x + 1

wrenexec = pywren.default_executor()
xlist = np.arange(10)
futures = wrenexec.map(addone, xlist)

print [f.result() for f in futures]
```

The output is as expected:

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

THE API

The most important primitive:

`map(function, data)`

and… that's mostly it

```
import pywren
import numpy as np

def addone(x):
    return x + 1

wrenexec = pywren.default_executor()
xlist = np.arange(10)
futures = wrenexec.map(addone, xlist)

print [f.result() for f in futures]
```

The output is as expected:

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

AWS Lambda

Run code without thinking about servers.
Pay for only the compute time you consume.

+

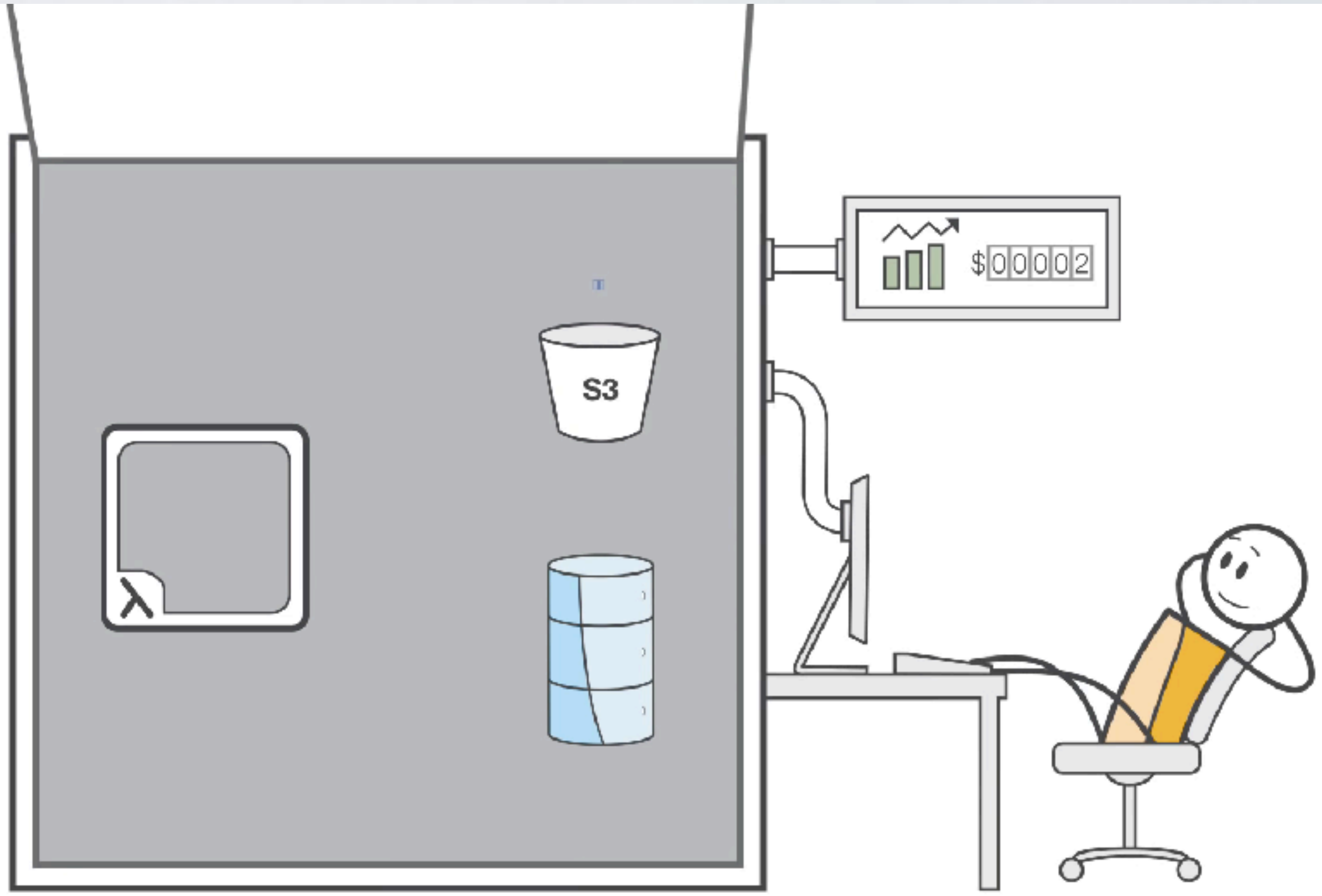
Amazon S3

Object storage built to store and retrieve any amount of data from anywhere

ANACONDA®

AWS LAMBDA

- 300 seconds
  single-core (AVX2)
- 512 MB in /tmp
- 1.5GB RAM
- Python, Java, Node



Google Cloud Platform

CLOUD FUNCTIONS ALPHA

A serverless platform for building event-based microservices

Microsoft Azure

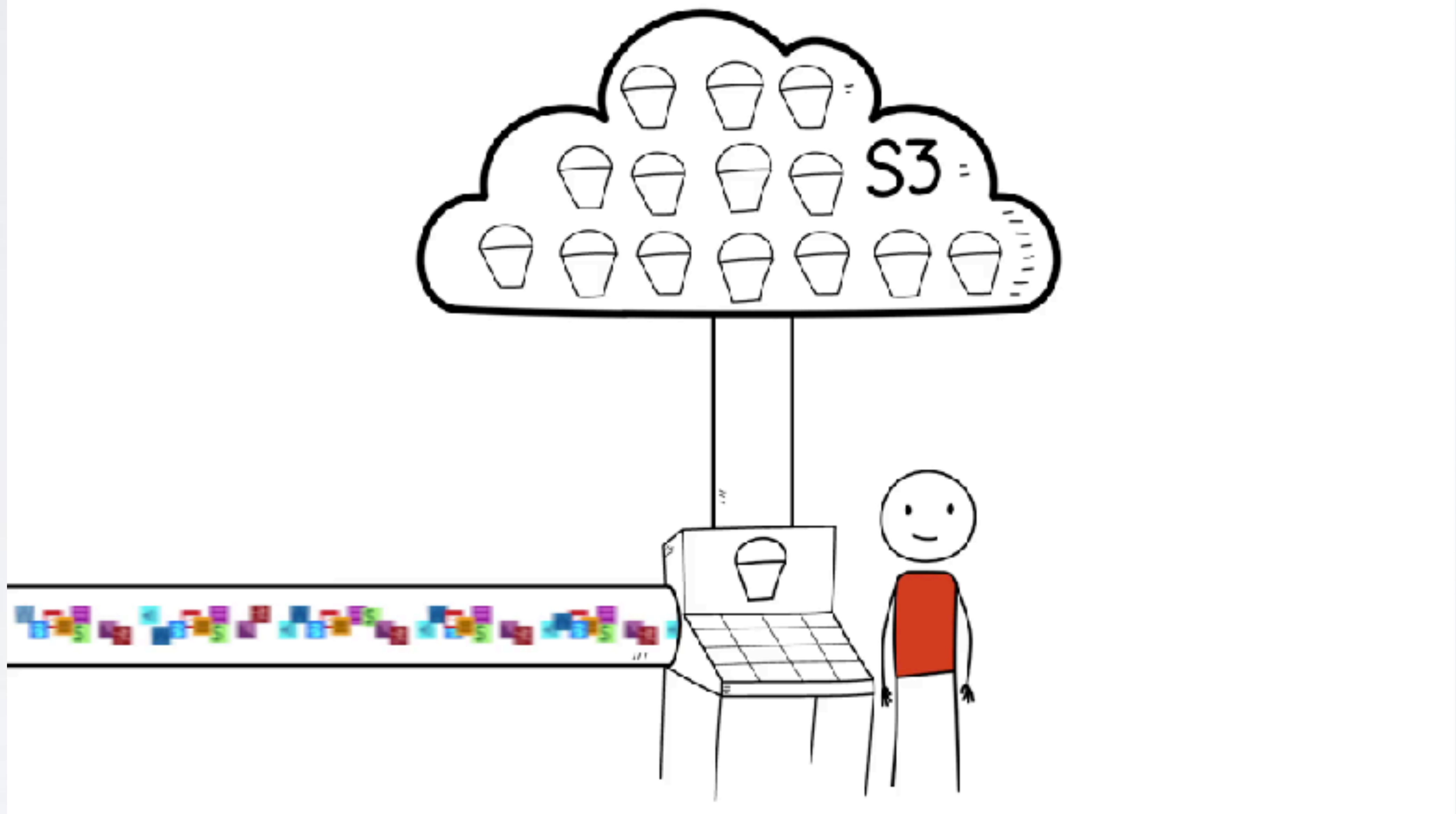
Azure Functions

Process events with a serverless code architecture

AMAZON S3

Simple Storage Service

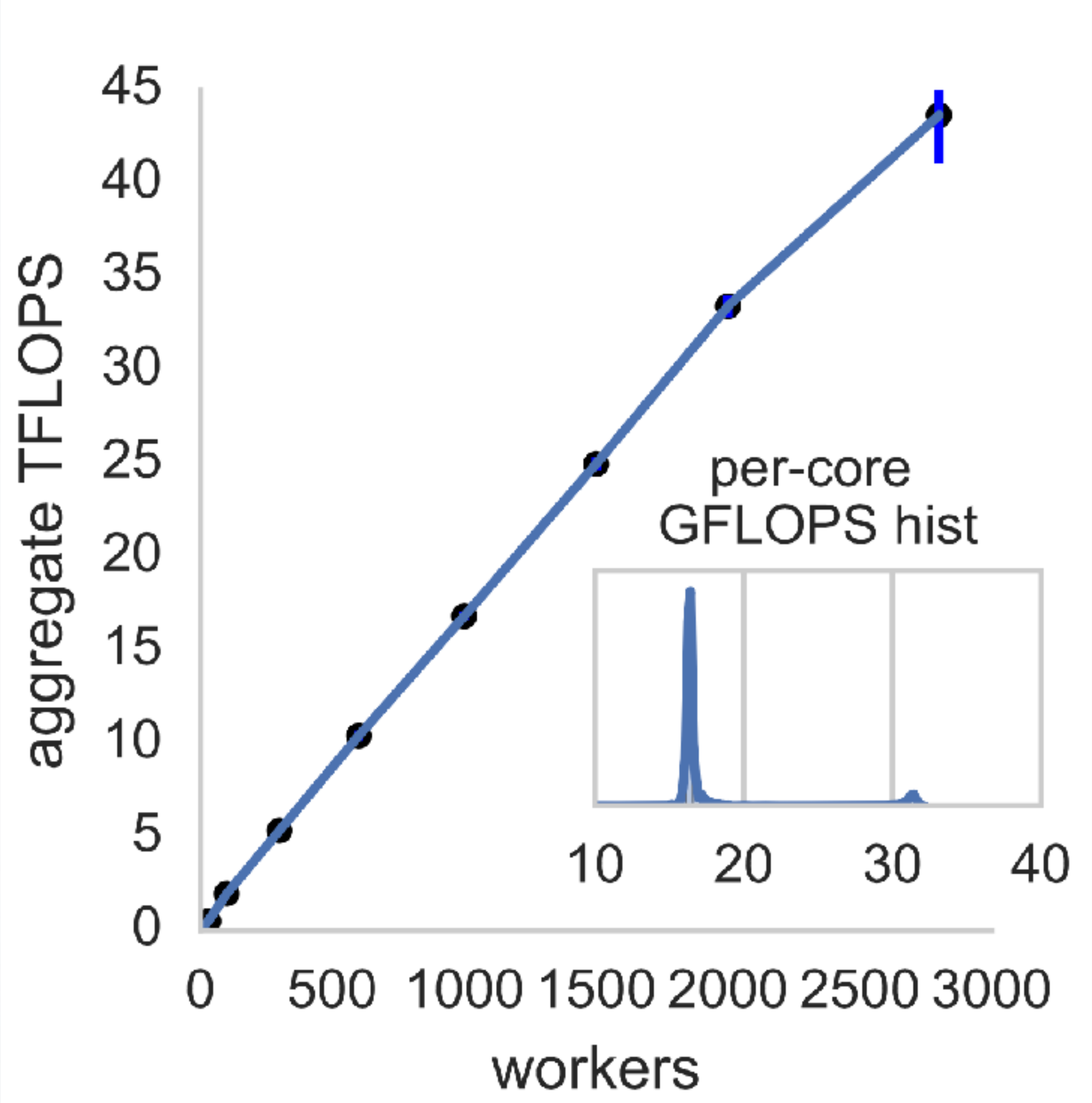
- **What is an object store?**
  - A place to put binary data
  - Look data up by a path
  - That's basically it



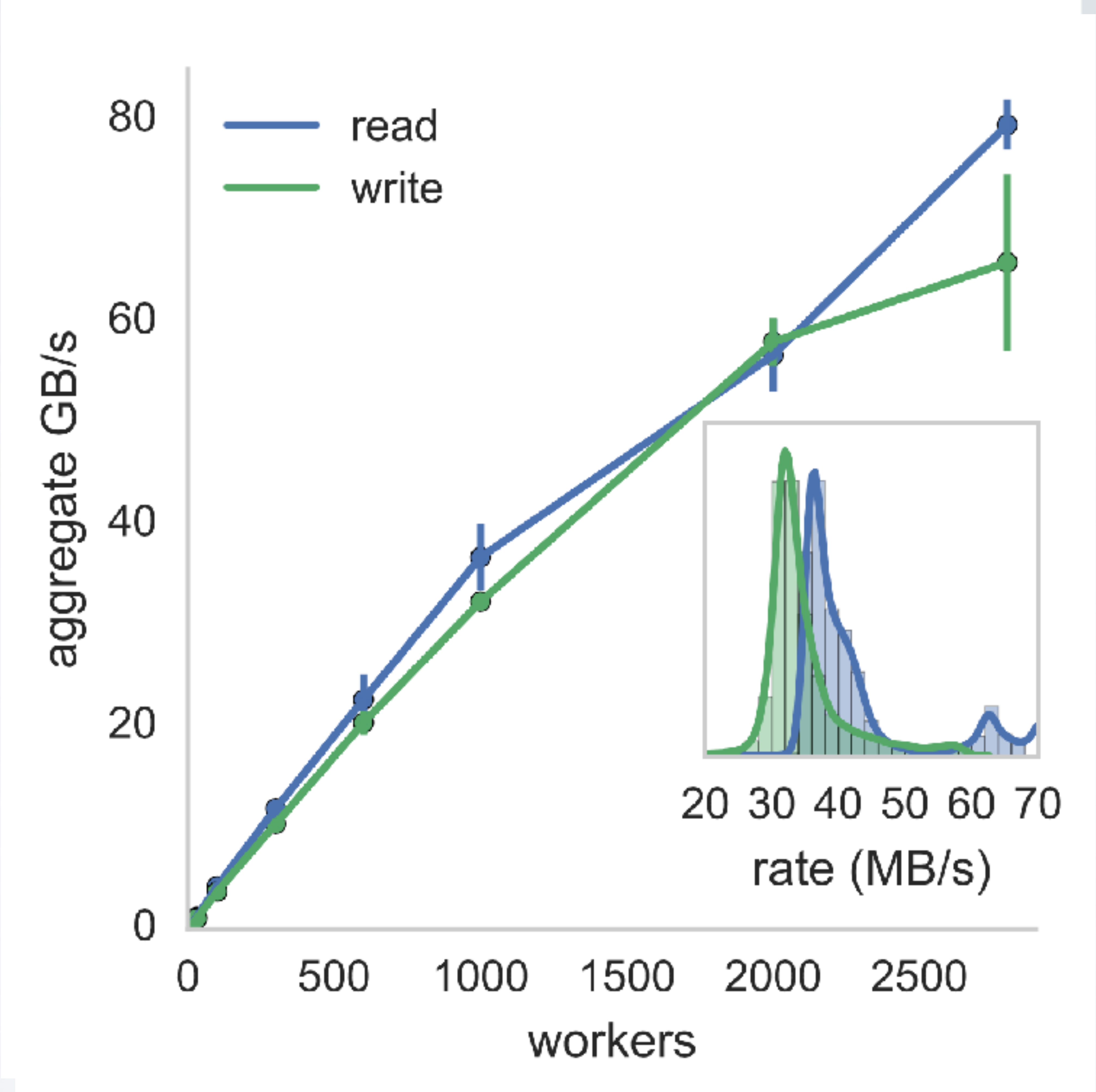
Unlike a regular filesystem there is no support for multiple read/write to a file, or writing parts of a file, or…

PYWREN SCALABILITY

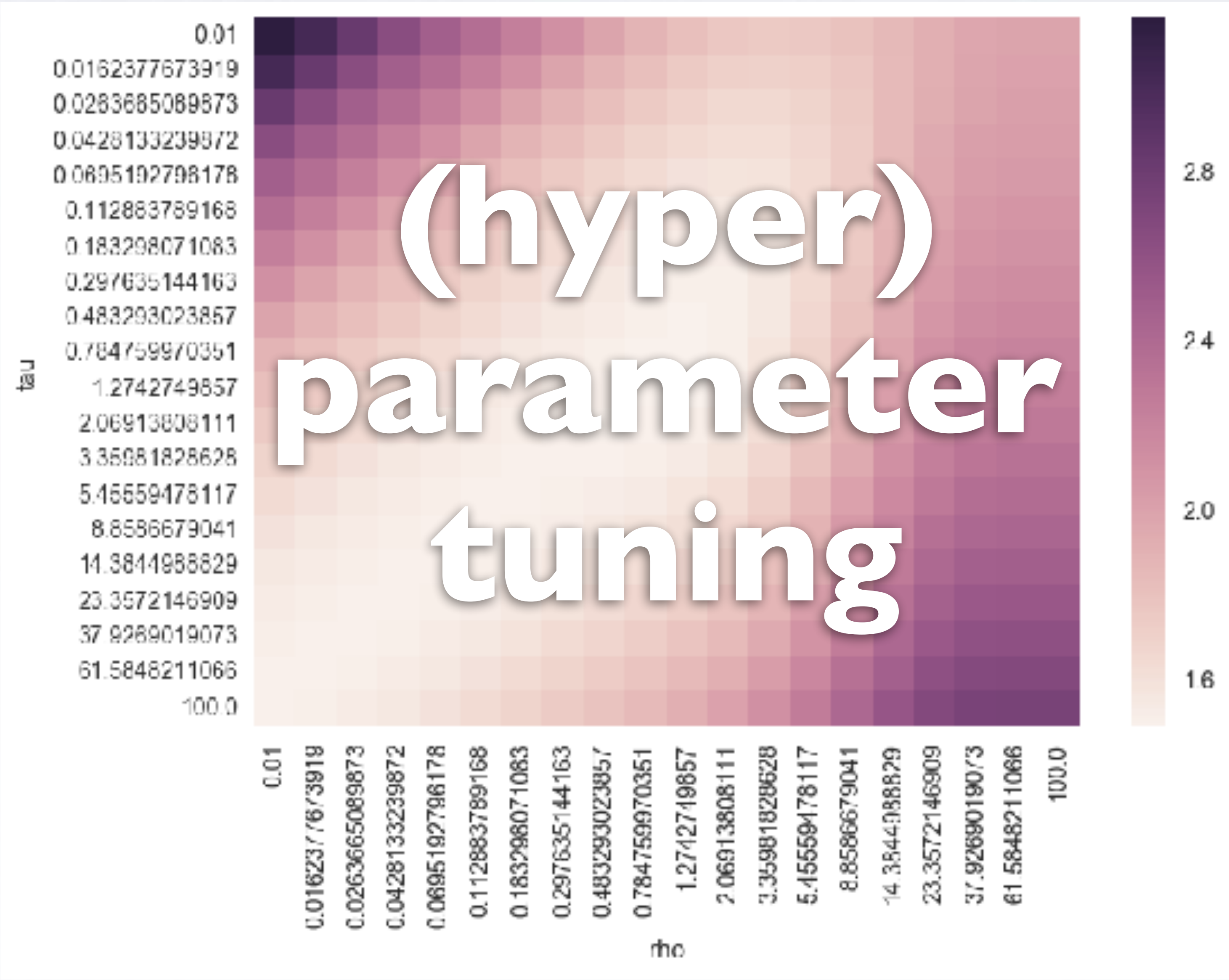
Compute

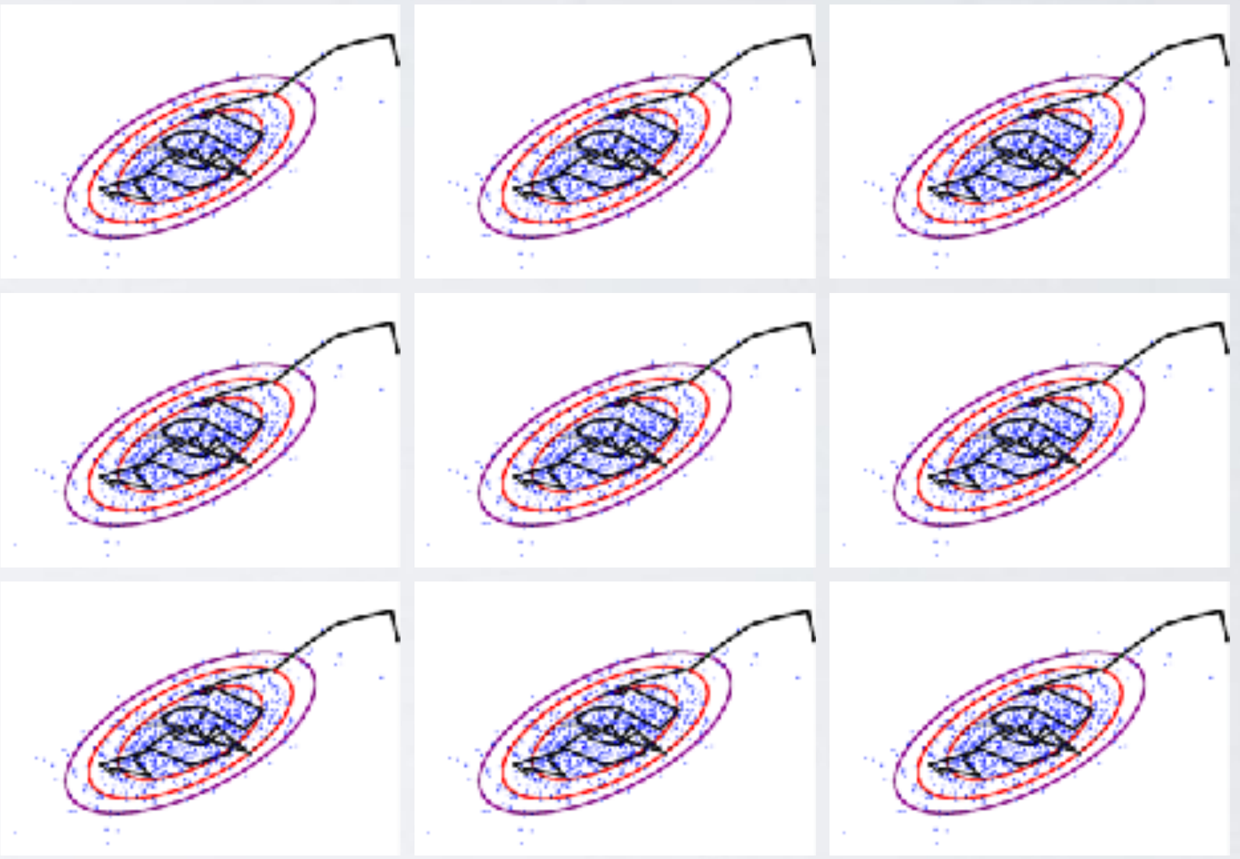


Data



YOU CAN DO A LOT OF WORK WITH MAP!



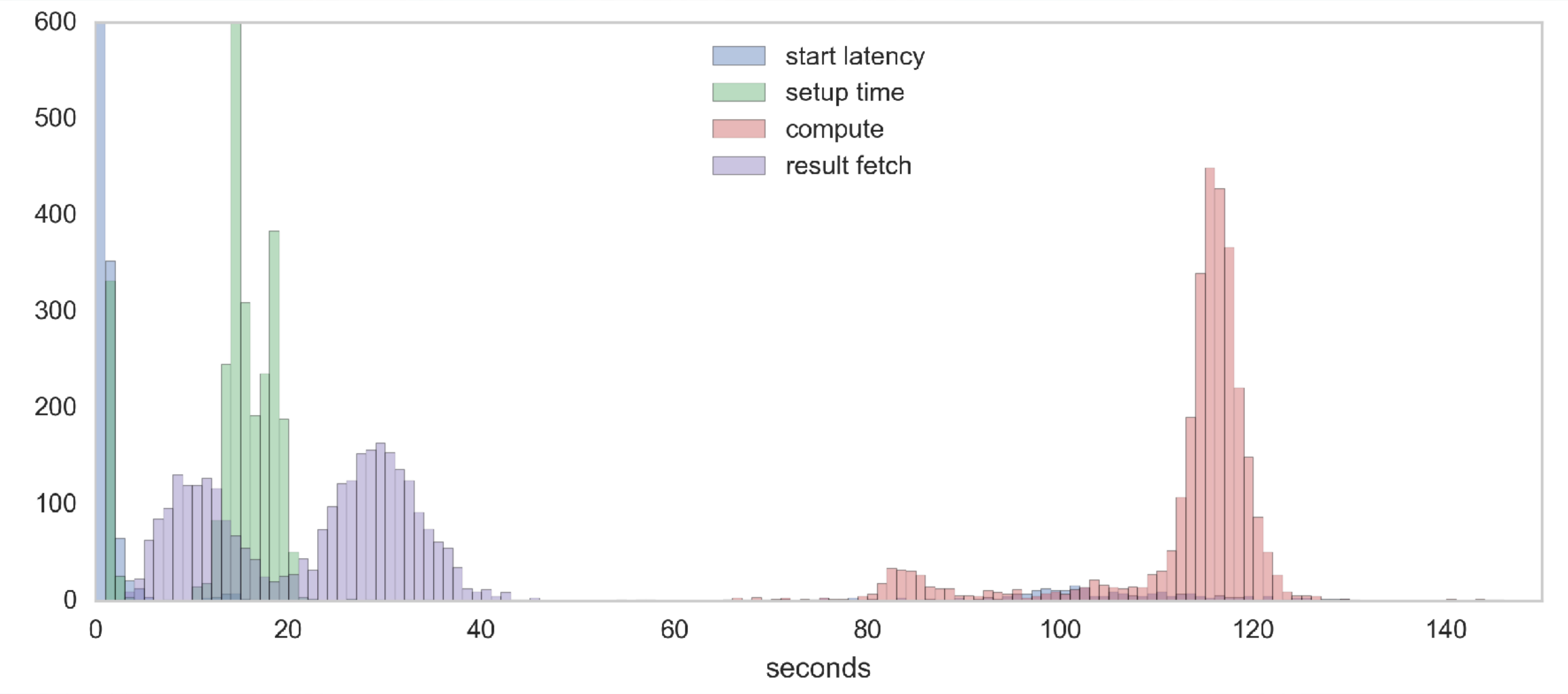


Scalable Simulation

Preprocess 1.4M images from
IMAGENET

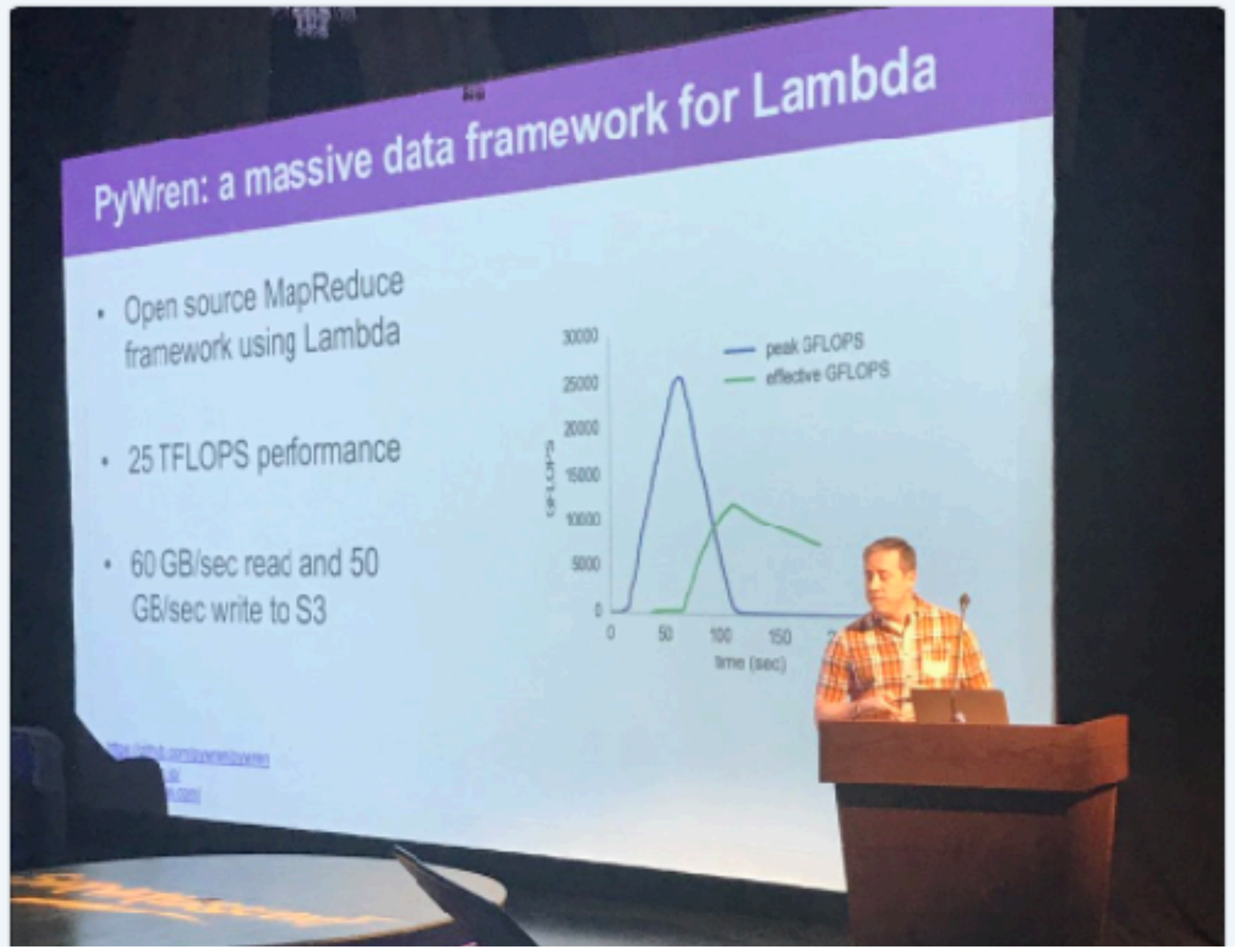
Compute GIST
image descriptor
(some random
python code off
the internet)

Host submit
Lambda Start
Setup done, Job start
Job Done
Results returned
Stragglers
host submit
job start
setup done
job done
results returned
job
3000
2900
2300
2200
2100
2000
1900
1800
1700
1600
1500
1400
1300
1200
1100
1000
900
800
700
600
500
400
300
200
100
0
0
50
100
150
200
250
wallclock time (sec)


600
500
400
300
200
100
0
start latency
setup time
compute
result fetch
0
20
40
60
80
100
120
140
seconds

INTEREST!

Michael H. Oshita @ijin · Apr 28
PyWren - lambda map/reduce framework. 25TFLOPS!
github.com/pywren/pywren #ServerlessConf

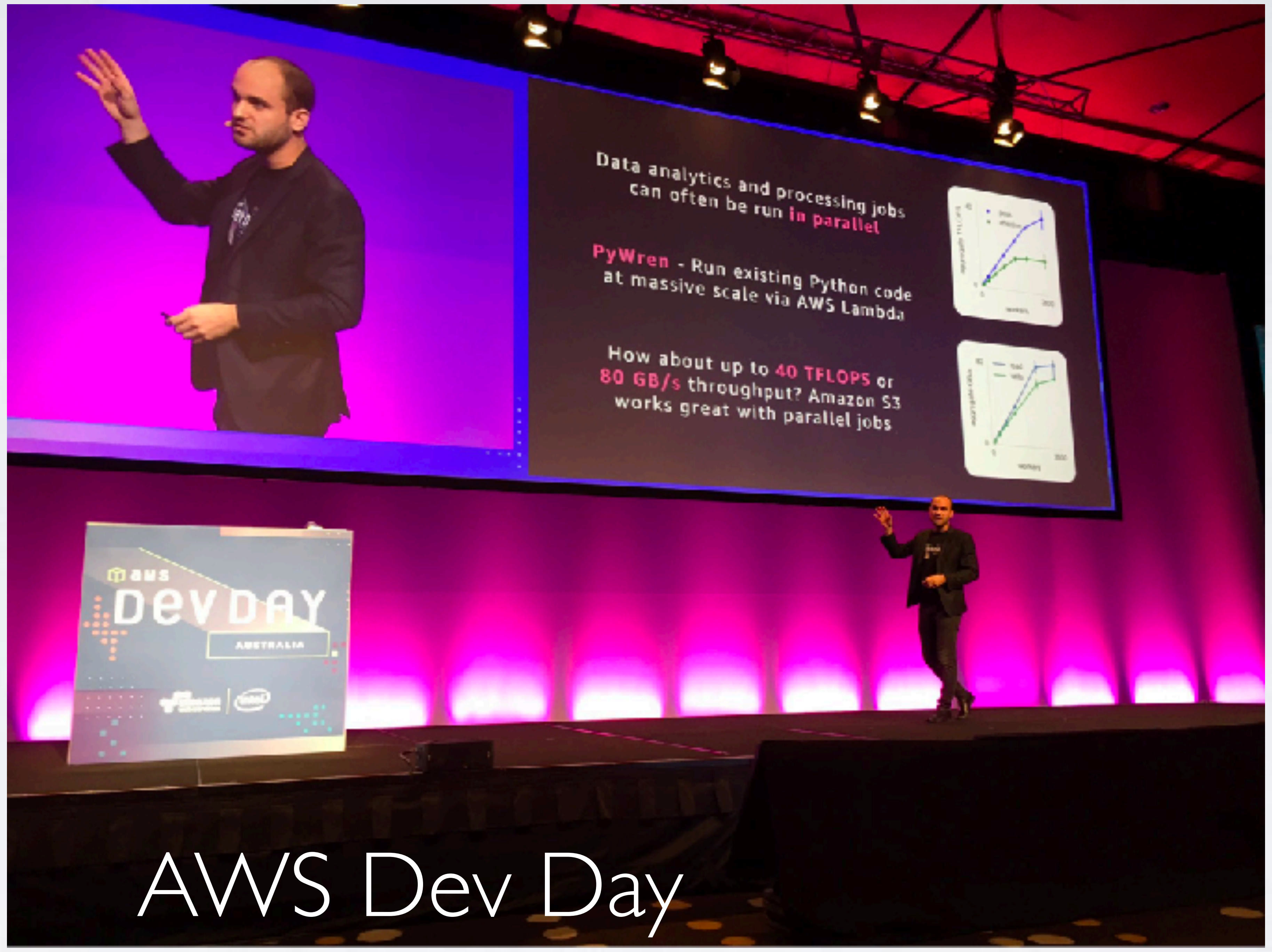


THENEWSTACK

EVENTS / TECHNOLOGY

With PyWren, AWS Lambda Finds an Unexpected Market in Scientific Computing

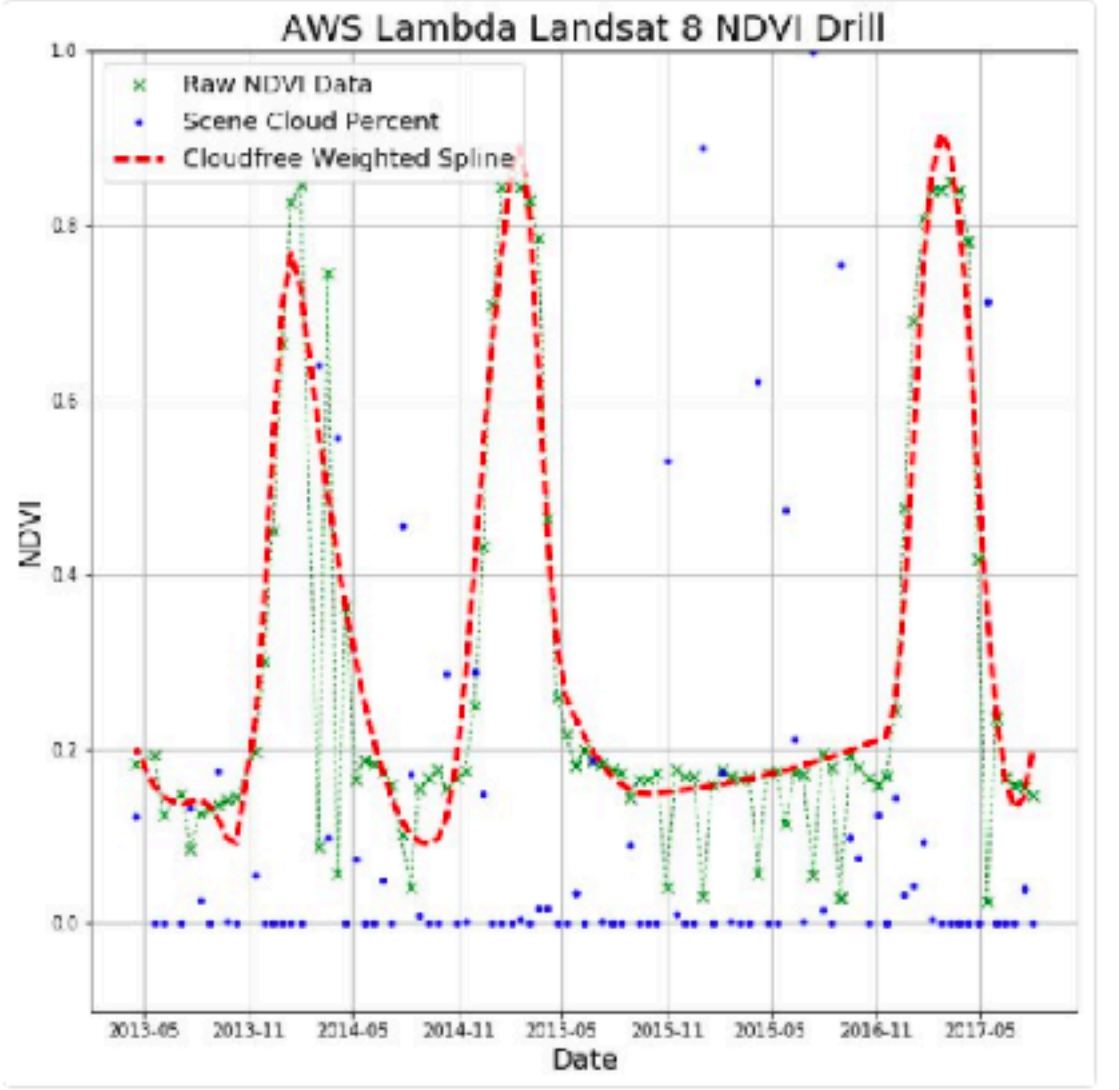
16 Feb 2017 10:26am, by Joab Jackson



AWS Dev Day

Peter Scarth @petescarth
Follow

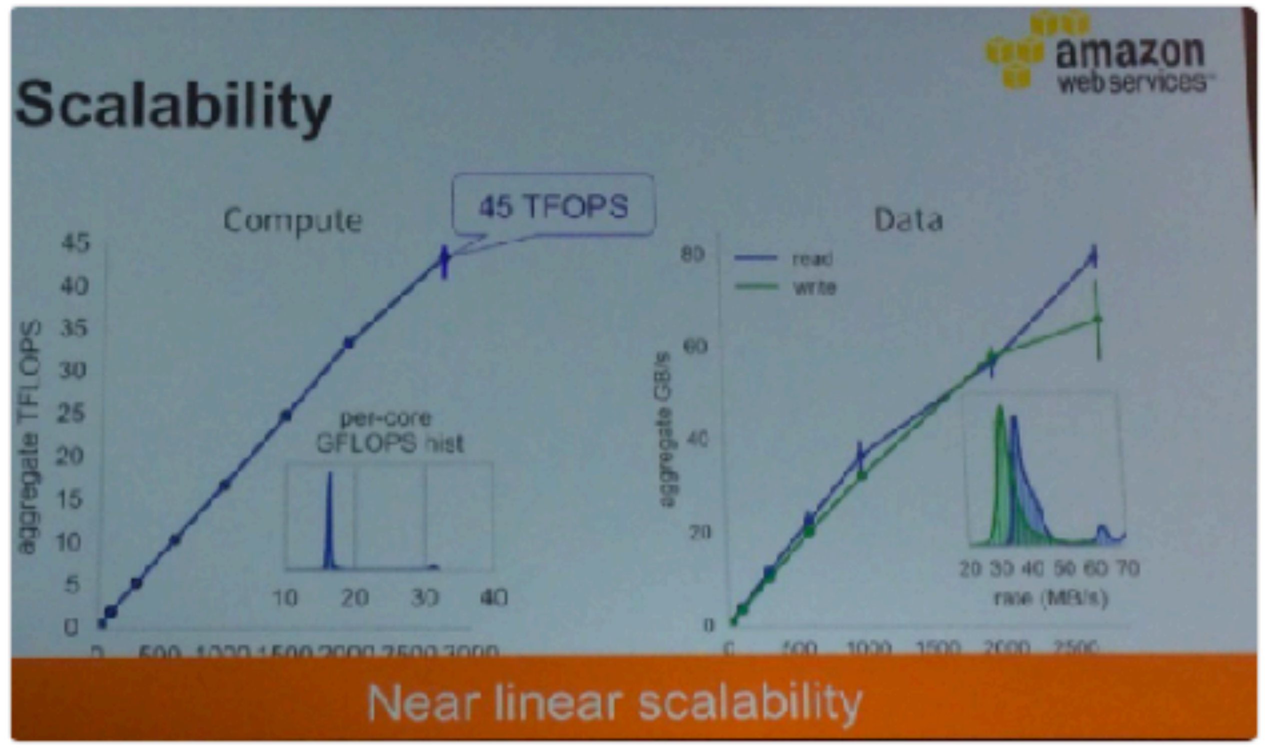
Today's little experiment - #Landsat8 time series extracted over cotton. #lambda + #pywren = #serverless query of 120 scenes in 60 seconds



1:17 AM - 13 Aug 2017

Dave Smith @DruidSmith
Follow

Wow, impressive scalability with #PyWren #AWSPSSummit



BL ABOUT PROJECTS BLOG

305 Million Solutions to The Black-Scholes Equation in 16 Minutes with AWS Lambda

Originally Posted: May 28, 2017

The research I'm working on involves estimating a firm's probability of default over a variety of time horizons using the Merton Distance to Default model. The dataset contains daily financial information for more than 24,000 firms over the past 30 years. Given that I am calculating the probability of default over five time horizons, applying the Merton model will require solving the Black-Scholes equation roughly 305 million times. Luckily, the model is easily parallelized because the only data needed for the model, aside from the risk-free rate, is firm specific. This post shows how the Python library Pywren can leverage AWS Lambda to run hundreds of models in parallel, achieving a **270x speed-up** over a quad-core i7-4770, with minimal changes to the simulation code. If you are interested in learning more about the model, see my post about implementing the model in Python.

one in a million @TearTheSky
Follow

サーバレスのトークを聞きにきてるけどFlaskだけ固有名詞で出たりPyWrenが出たり、スピーカーはPython推しなのかな？ PyWrenは科学計算フレームワークみたい。
aws.amazon.com/jp/blogs/news/ ...

Translate from Japanese

9:34 PM - 30 May 2017

Werner Vogels @Werner
Follow

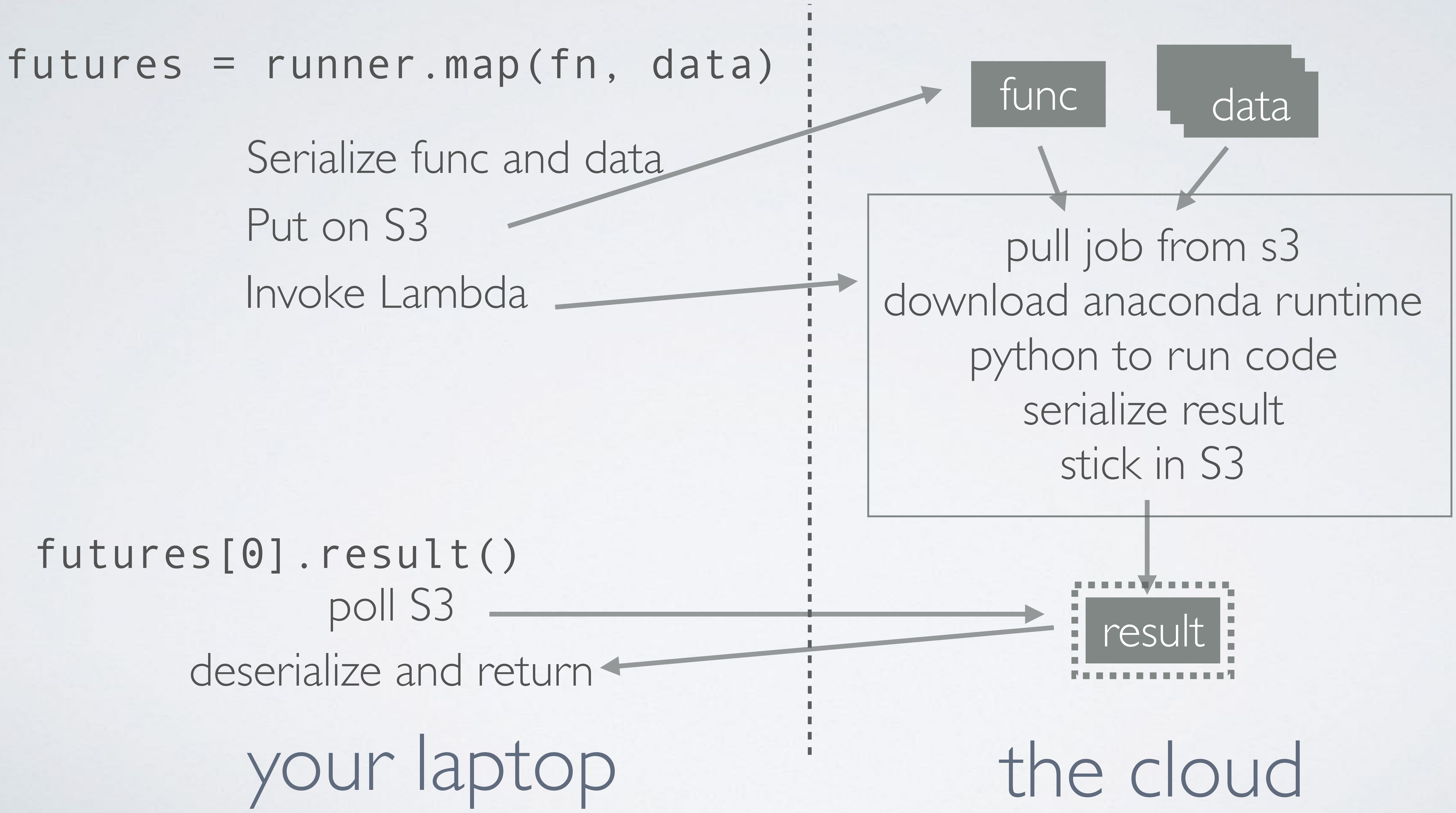
#Microservices and TerraFlops - Extracting 25 TFLOPS from #AWS #Lambda - @stochastician on the origin of #pywren ericjonas.com/pywren.html

SOCC ACM Symposium on Cloud Computing

Occupy the Cloud: Distributed Computing for the 99% [VISION]

Eric Jonas, Qifan Pu, Shivaram Venkataraman, Ion Stoica, Benjamin Recht (UC Berkeley)

HOW IT WORKS


futures = runner.map(fn, data)
Serialize func and data
Put on S3
Invoke Lambda
func
data
pull job from s3
download anaconda runtime
python to run code
serialize result
stick in S3
futures[0].result()
poll S3
deserialize and return
result
your laptop
the cloud

(Leptotyphlops carlae)

Want our runtime to include

NumPy

SciPy

Cython

Numba

scikit learn

pillow

Start **1205MB**

conda clean

977 MB

eliminate pkg

946 MB

Delete non-AVX2 MKL

670 MB

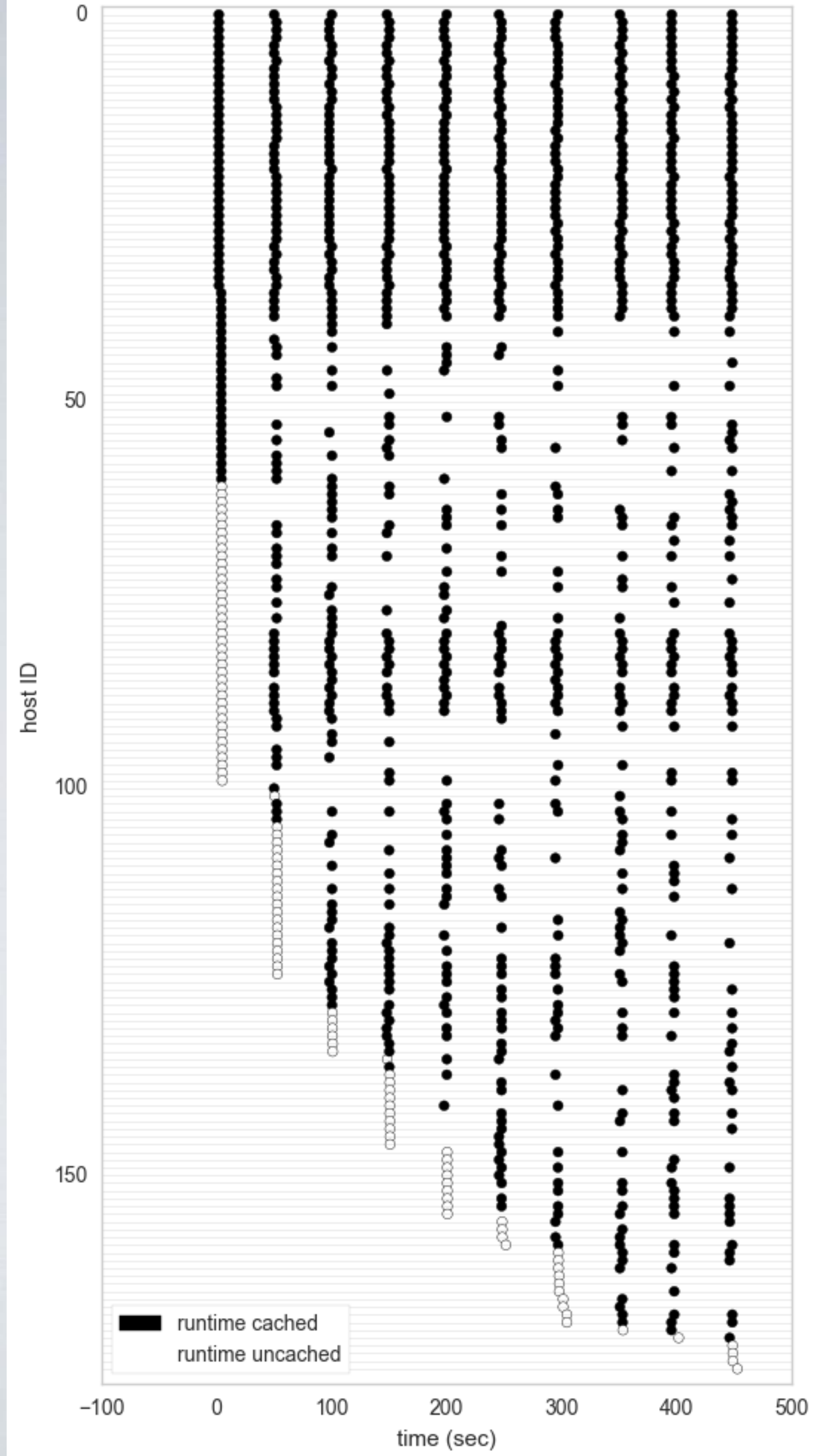
strip shared libs

510MB

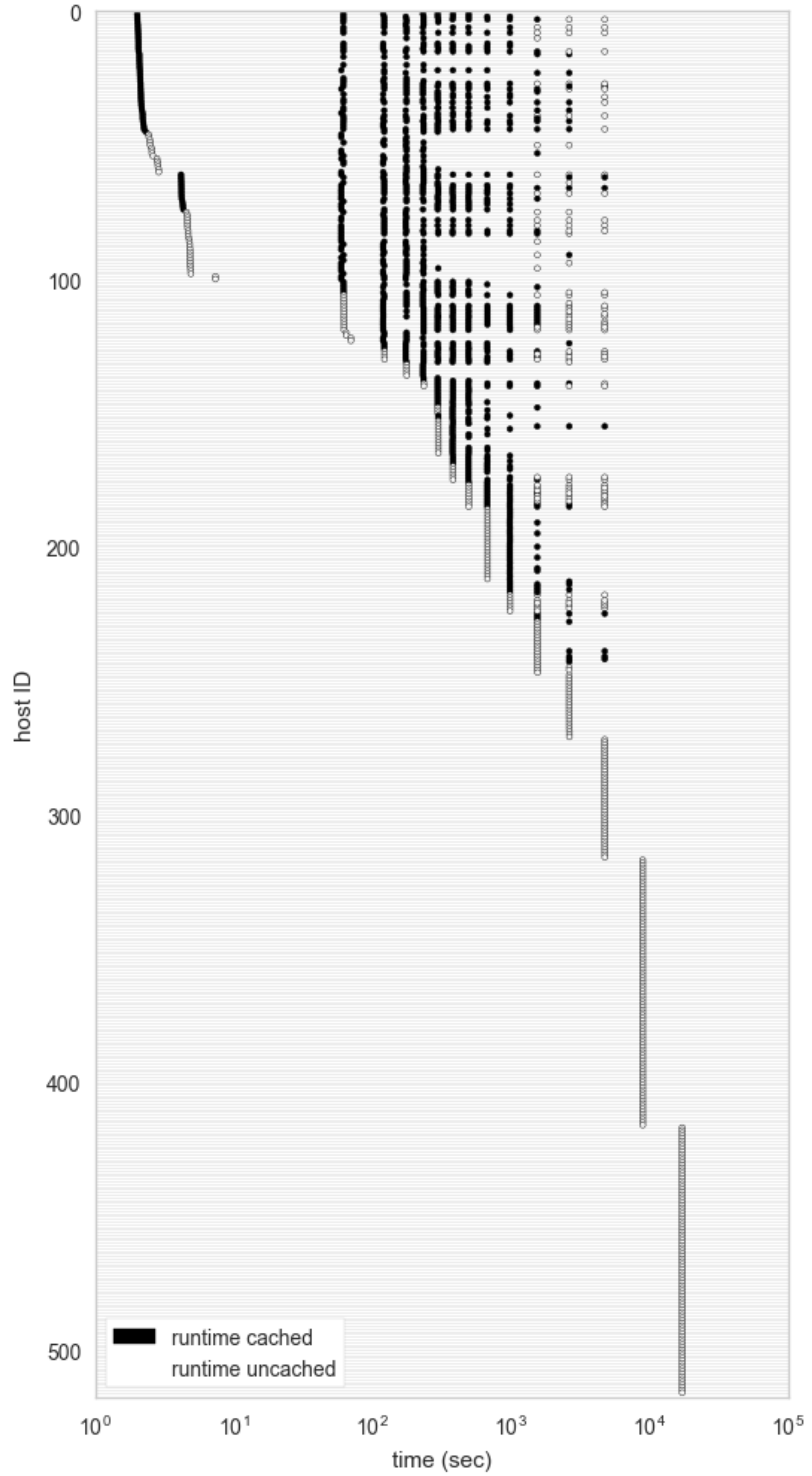
delete pyc

441MB

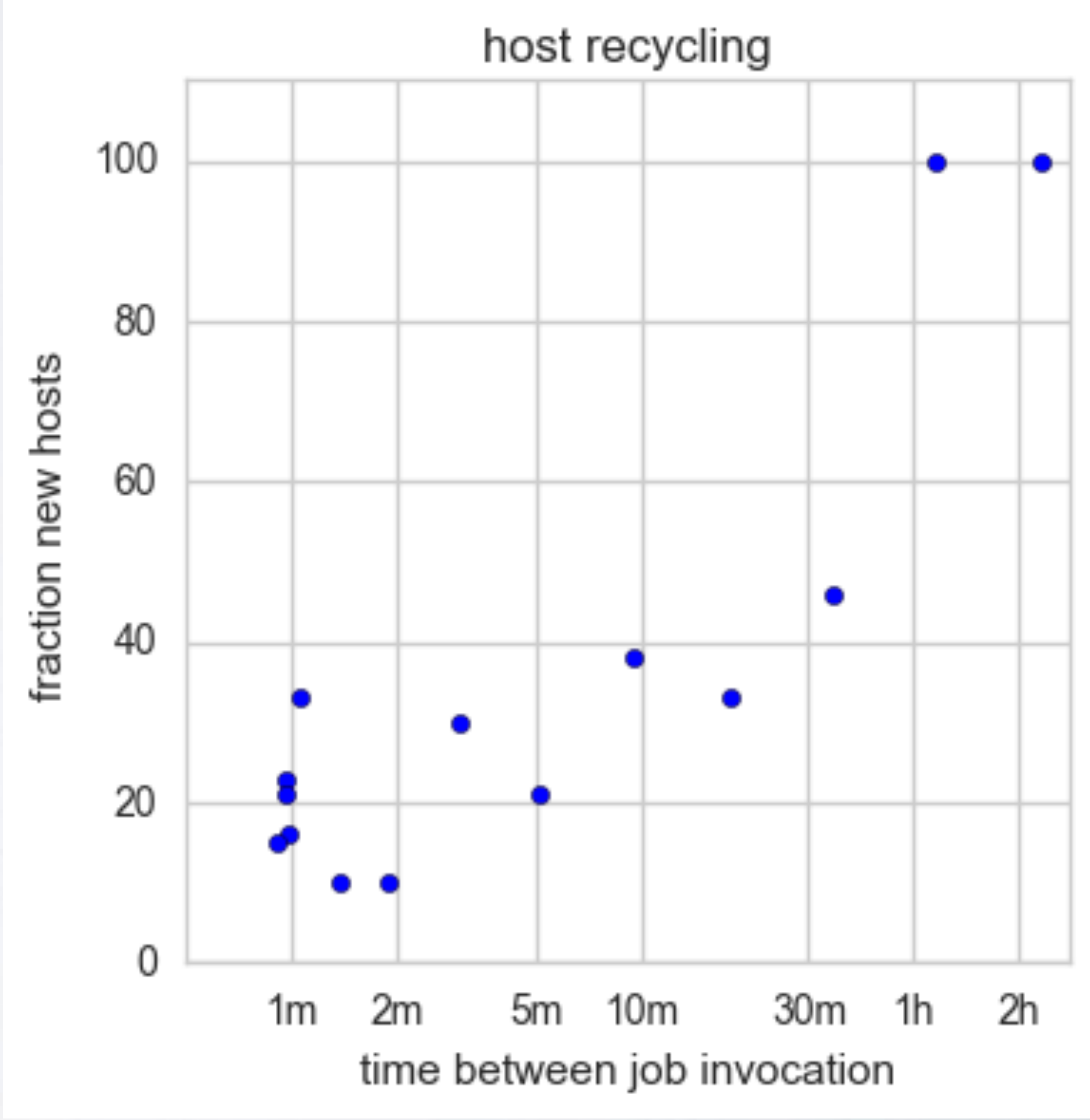
BEHIND
HOO



host ID
time (sec)
runtime cached
runtime uncached



host ID
time (sec)
runtime cached
runtime uncached



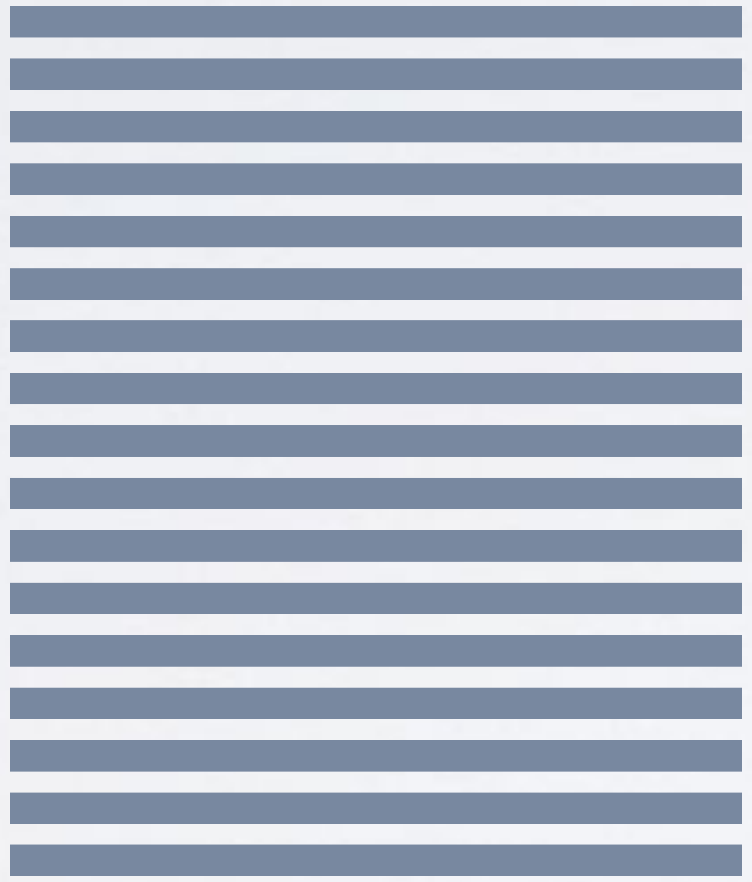
host recycling
fraction new hosts
time between job invocation
1m
2m
5m
10m
30m
1h
2h

MAP IS NOT ENOUGH?

A lot of data analytics looks like:

Data

ETL / preprocessing



featurization

machine learning

Great PyWren Fit

Distributed!
Scale! TensorFlow
Deep MLBase

“You can have a second computer when you’ve shown you know how to use the first one.”

–Paul Barnum, quoted in McSherry, 2015

scalable system	cores	twitter	uk-2007-05
Stratosphere [8]	16	950s	-
X-Stream [21]	16	1159s	-
Spark [10]	128	1784s	$\geq$ 8000s
Giraph [10]	128	200s	$\geq$ 8000s
GraphLab [10]	128	242s	714s
GraphX [10]	128	251s	800s
Single thread (SSD)	1	153s	417s

Table 3: Reported elapsed times for label propagation, compared with measured times for single-threaded label propagation from SSD.

scalable system	cores	twitter	uk-2007-05
GraphLab	128	249s	833s
GraphX	128	419s	462s
Vertex order (SSD)	1	300s	651s
Vertex order (RAM)	1	275s	-
Hilbert order (SSD)	1	242s	256s
Hilbert order (RAM)	1	110s	-

Table 4: Reported elapsed times for 20 PageRank iterations, compared with measured times for single-threaded implementations from SSD and from RAM. The single-threaded times use identical algorithms, but with different edge orders.

Scalability! But at what COST? Frank McSherry, Michael Isard, Derek G. Murray.
USENIX Hot Topics In Operating Systems, 2015

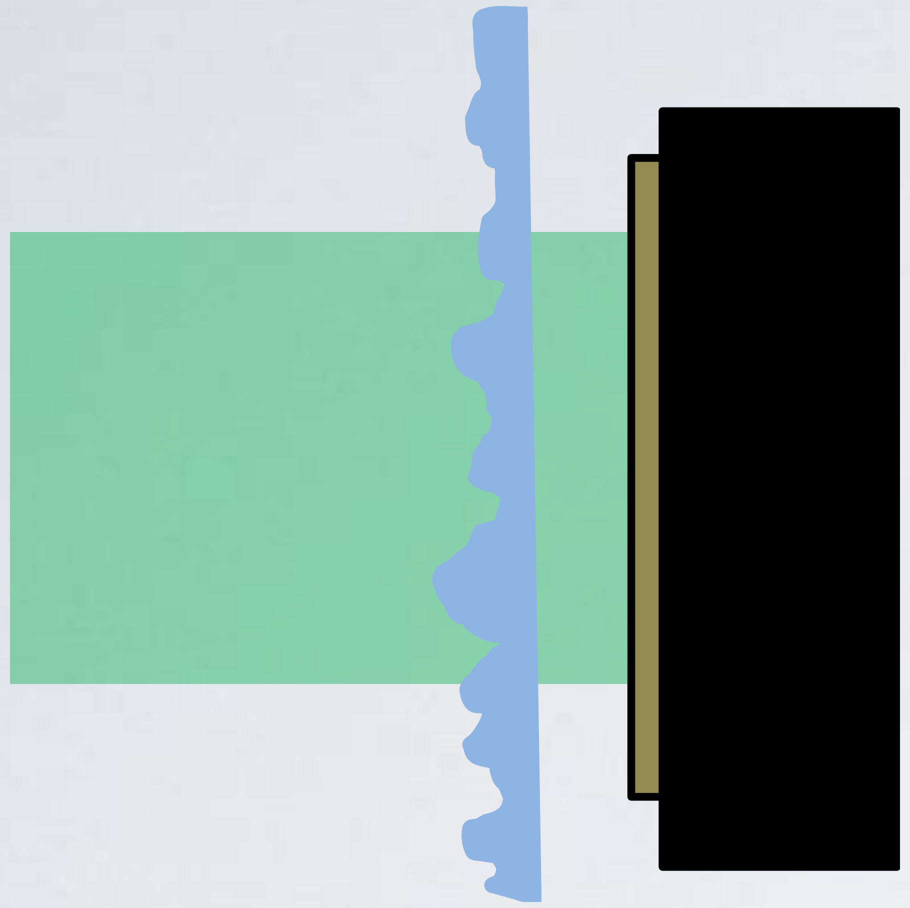
SINGLE-MACHINE REDUCE

	cores	RAM	COST
x1.32xlarge	64	2 TB	$14/hr
x1.16xlarge	32	1 TB	$7/hr
p2.16xlarge	32 + 16 GPUs	750 GB	$14/hr
r4.16xlarge	32	500 GB	$4/hr

```
futures = exec.map(function, data)

answer = exec.reduce(reduce_func, futures)
```


USING PYWREN
(my day job)

COMPUTATIONAL IMAGING



Hardware design

Take Image

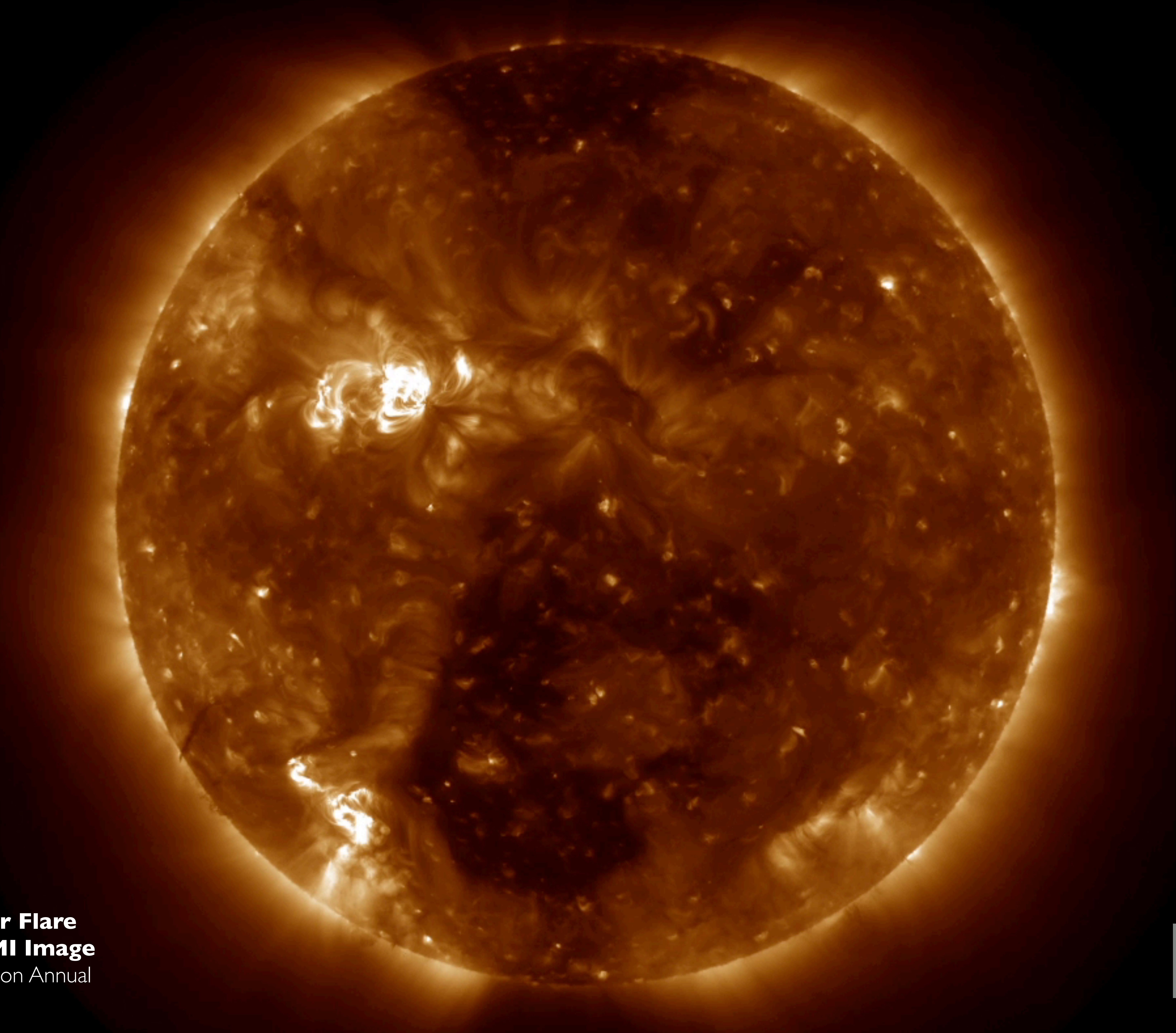
Processing

Success

Complex forward models

Large-scale solvers

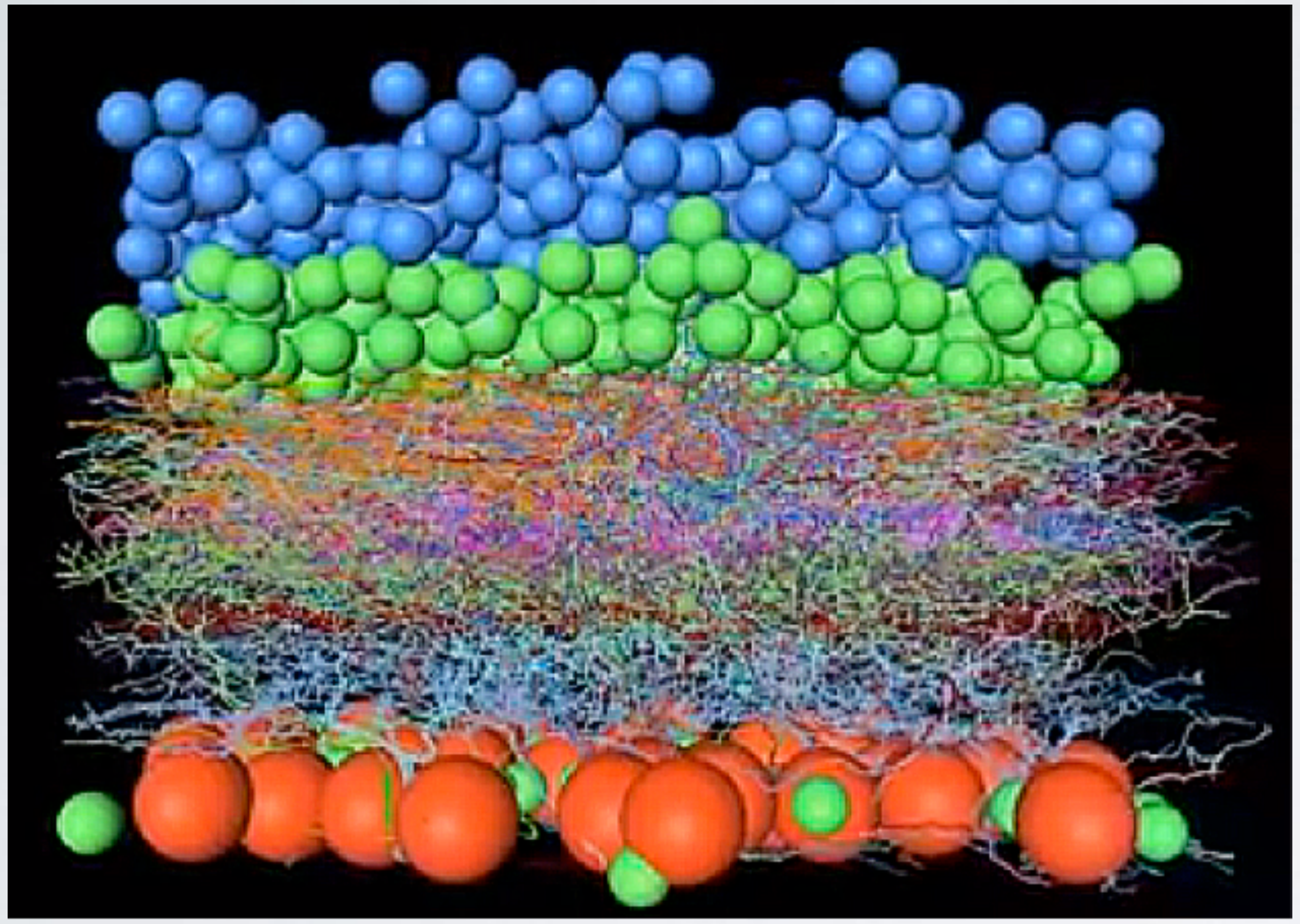
Nick Antipa, Sylvia Necula, Ren Ng, Laura Waller
"**Single-shot diffuser-encoded light field imaging.**" *Computational Photography (ICCP), 2016 IEEE International Conference on*. IEEE, 2016.

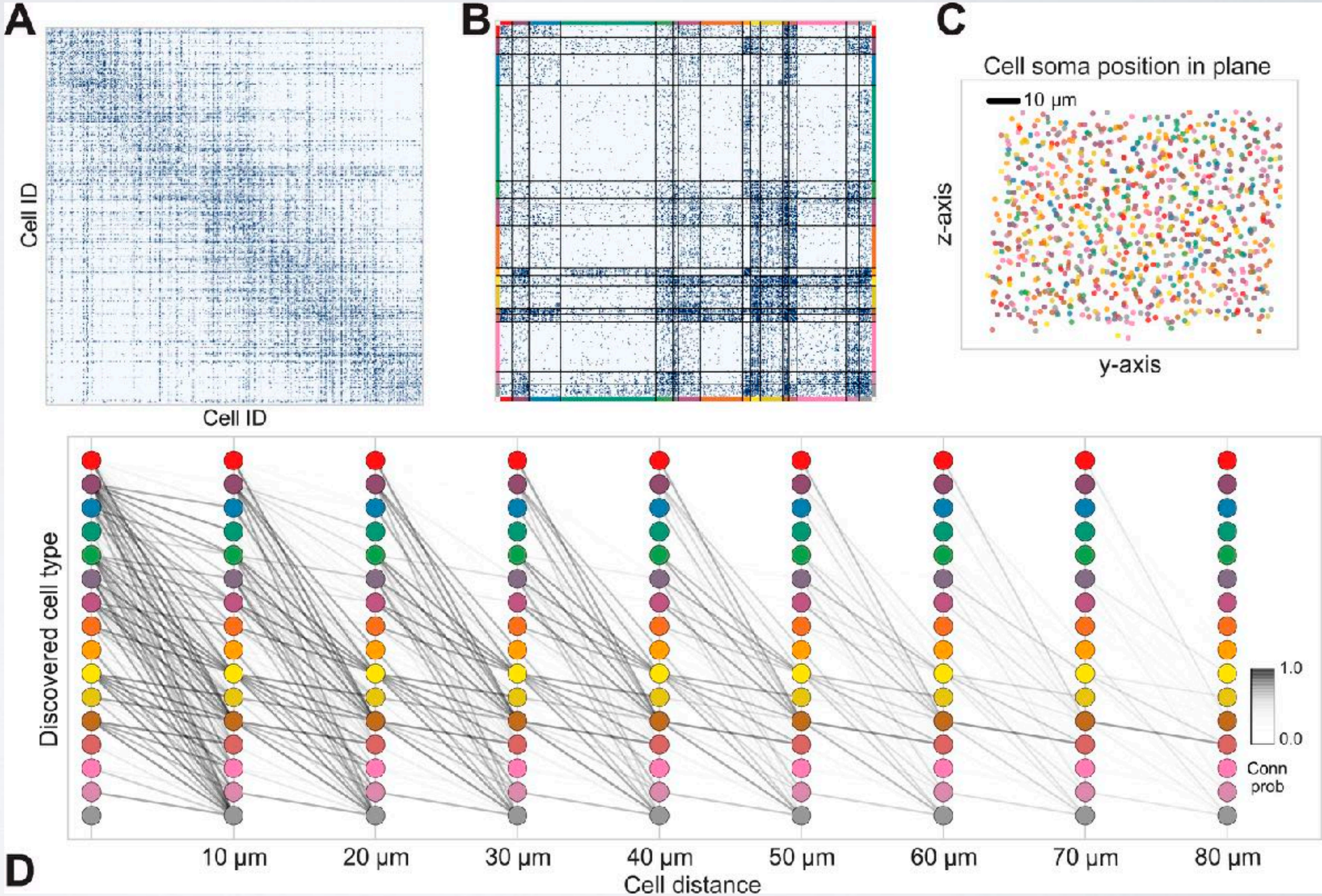


Jonas, Shankar, Bobra, Recht. **Solar Flare Prediction via AIA and HMI Image data.** American Geophysical Union Annual Meeting, 2016

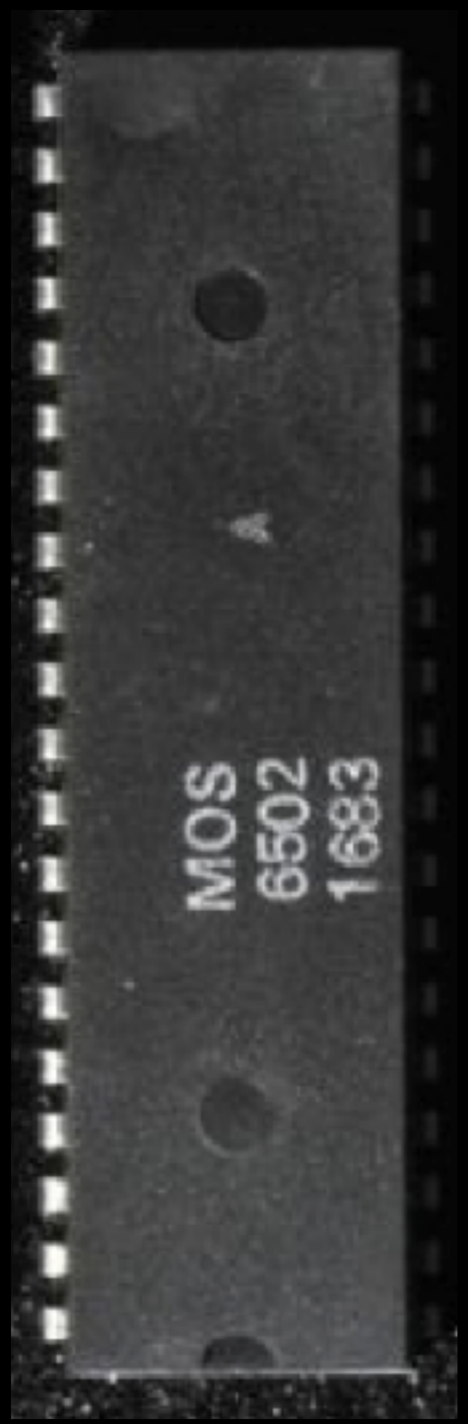
1.5 TB/day

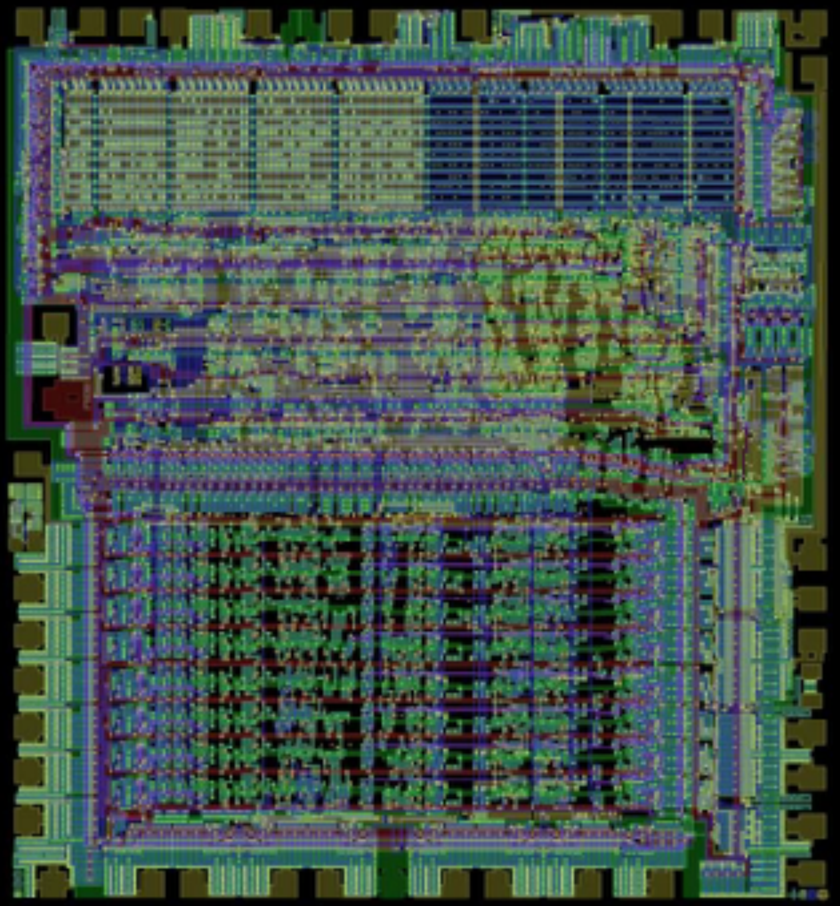
NEUROSCIENCE

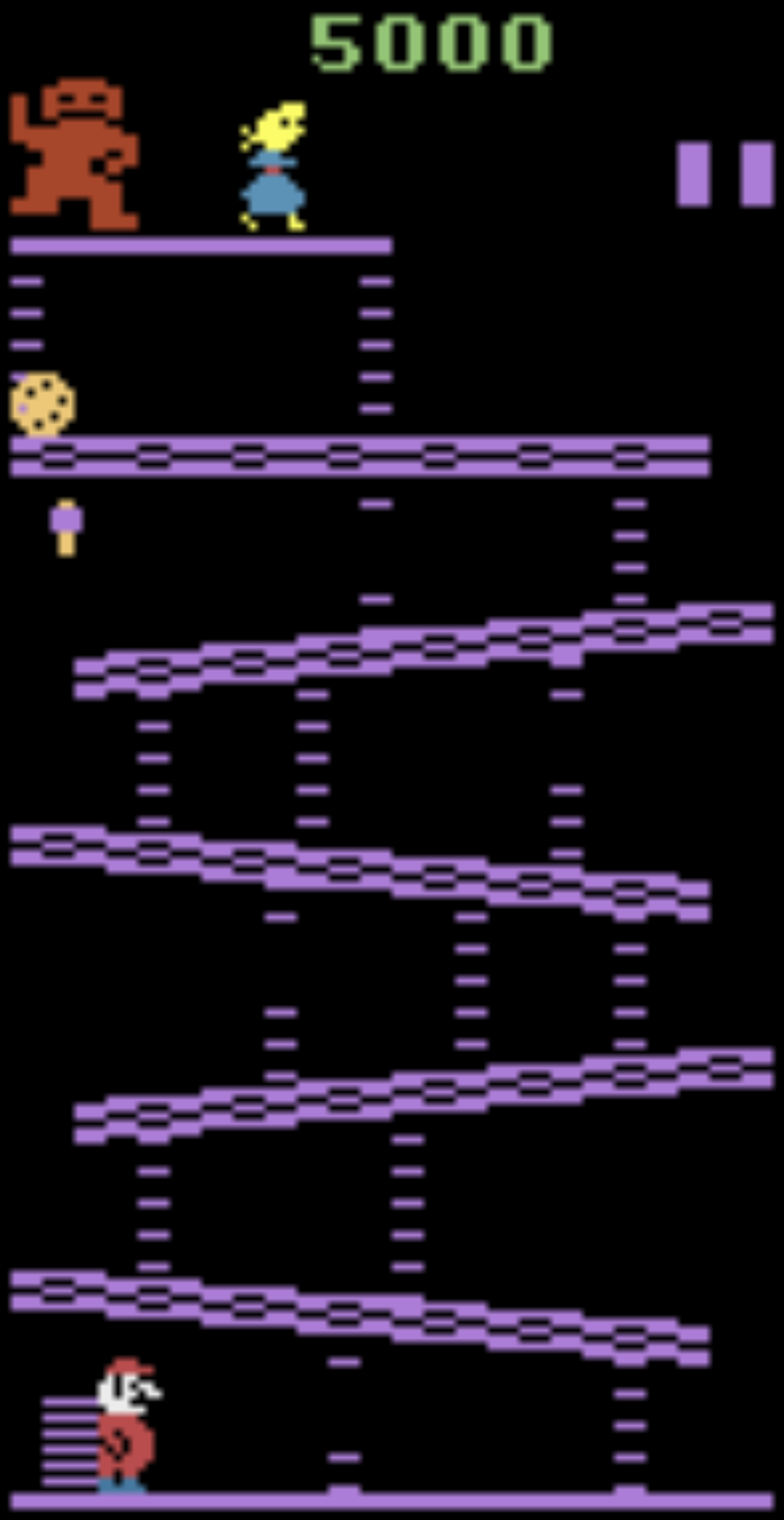


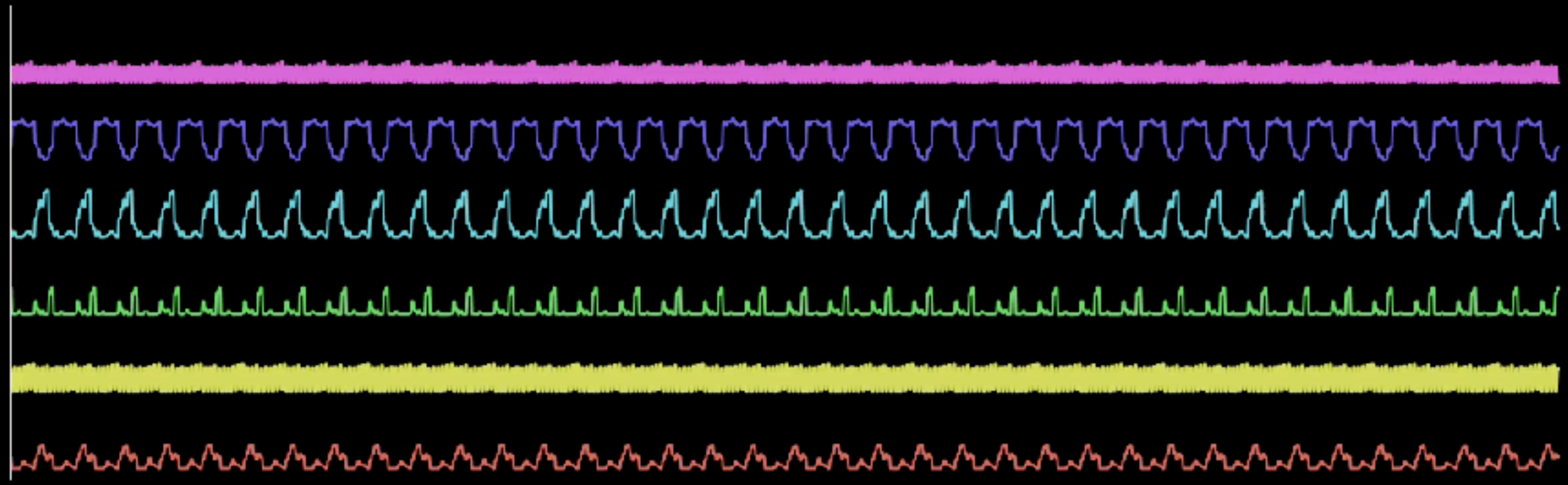


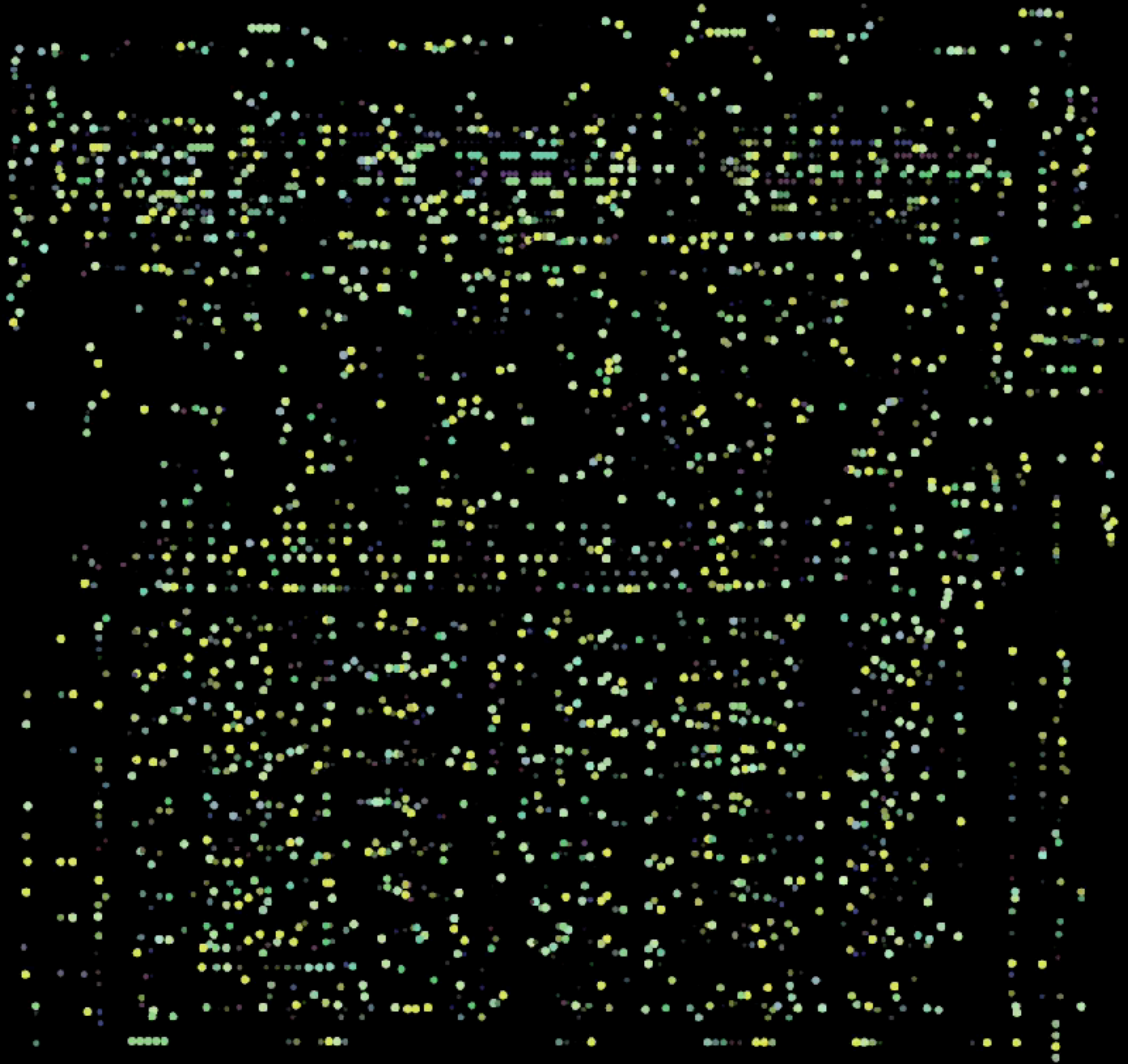
Eric Jonas and Konrad Kording.
Automatic discovery of cell types and microcircuitry from neural connectomics eLife, April 30 2015











Could a Neuroscientist understand a microprocessor?
Jonas, Kording. PLOS Computational Biology, 2017

CURRENT RESEARCH DIRECTIONS

CURRENT PYWREN RESEARCH

- Beyond PSPACE
- λPACK
- Towards Shuffle
- Comparison of Cloud Providers

Encoding, Fast and Slow:
Low-Latency Video Processing Using Thousands of Tiny Threads

Sadjad Fouladi, Riad S. Wahby, Brennan Shacklett,
Karthikeyan Vasuki Balasubramaniam, William Zeng, Rahul Bhalerao,
Anirudh Sivaraman, George Porter, Keith Winstein
Stanford University, University of California San Diego, Massachusetts Institute of Technology

NSDI '17

Wise Technology

Serverless Distributed Decision Forests with AWS Lambda

Posted by Joshua Bloom June 26, 2017

Within the Wise.io team in GE Digital, we have monthly "edu-hackdays" where the entire tech team spends the entire day trying to learn and implement new promising

Serverless λ Databases

risElab UC Berkeley

Johann Schleier-Smith
& Joe Hellerstein

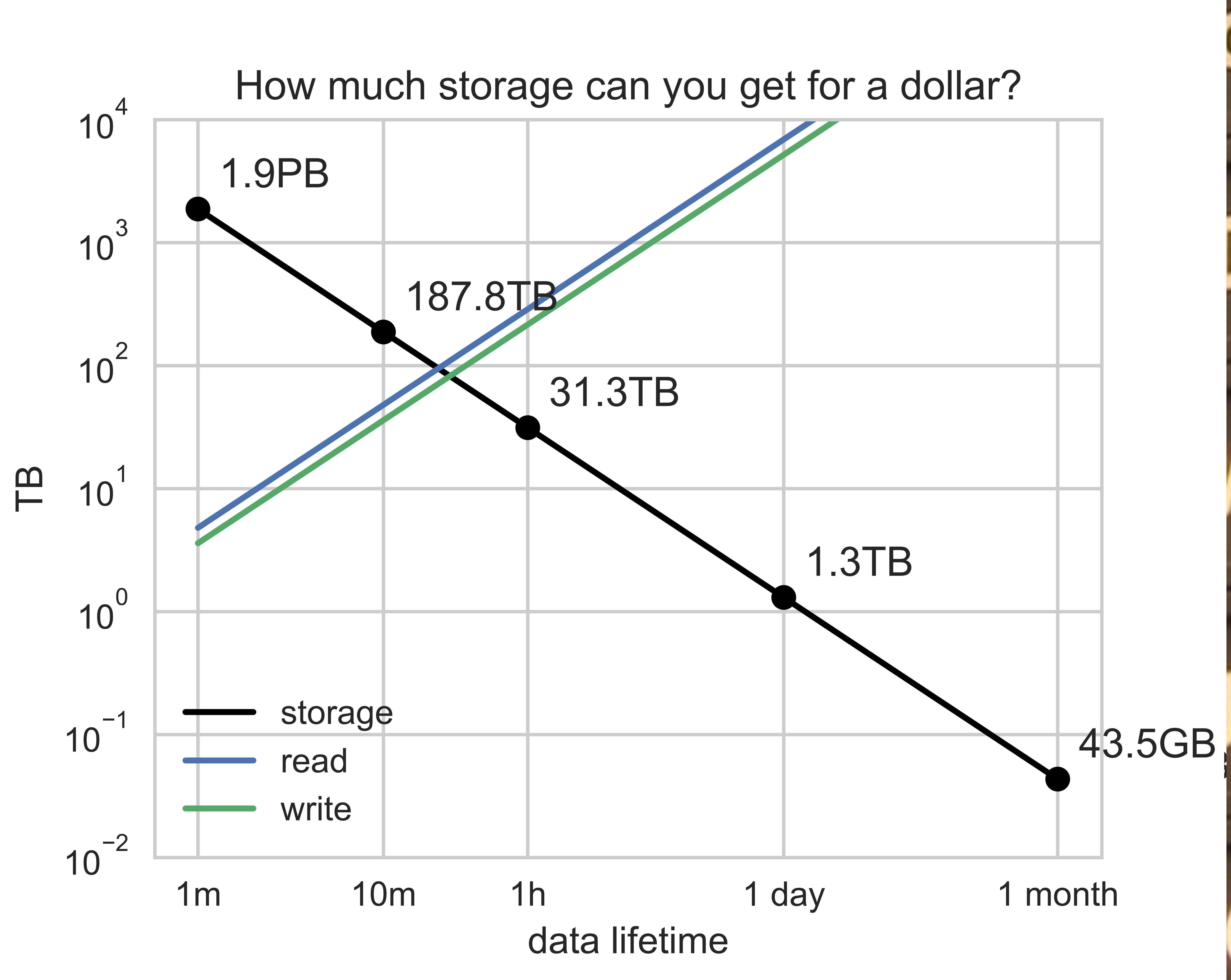
HOW EXPENSIVE IS **S3**?

(Taking dimensionality analysis seriously, or "beyond PSPACE")

Storage Pricing (varies by region)

Region: US West (Oregon)

	Standard Storage
First 50 TB / month	$0.023 per GB
Next 450 TB / month	$0.022 per GB
Over 500 TB / month	$0.021 per GB



- How do algorithms change when you have infinite memory (through a straw)
- Never discard intermediate information

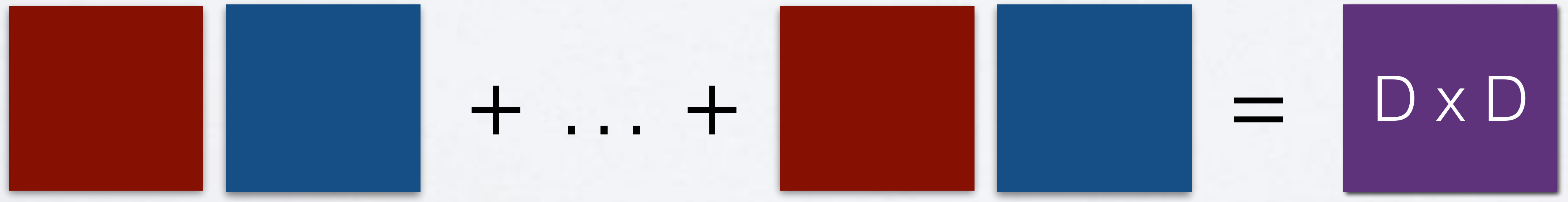
NumPyWren

Vaishaal Shankar

NumPyWren

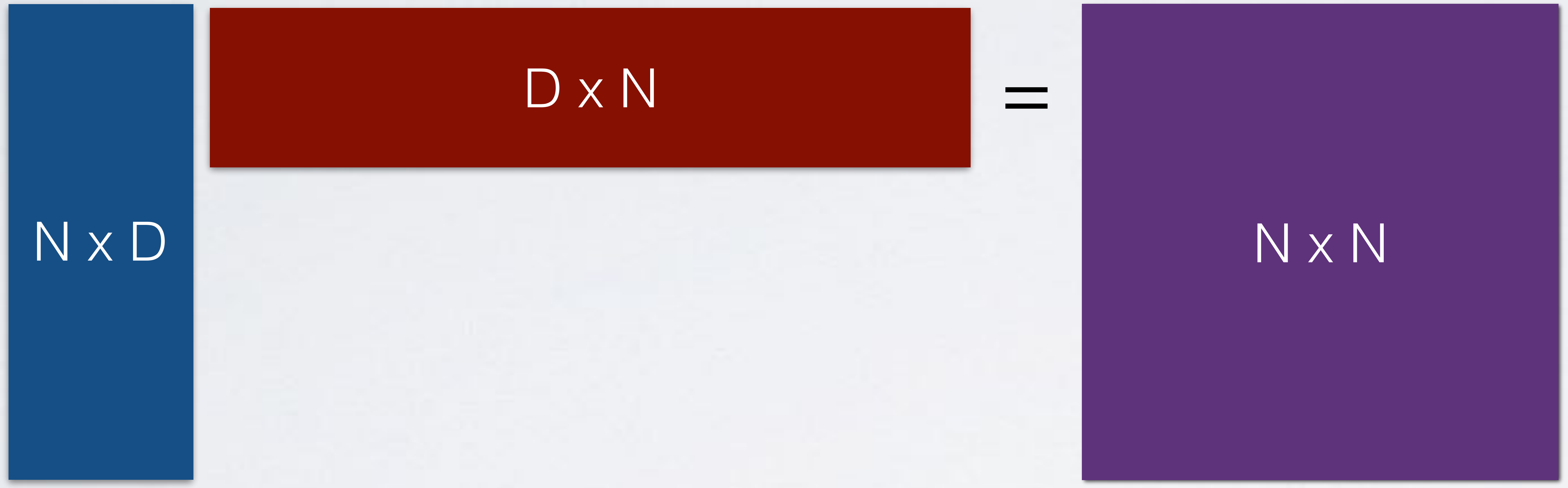
- That's a lot of SIMD cores!
- Parallel matrix multiplication is easy when output matrix is small

- Fits cleanly into map-reduce framework

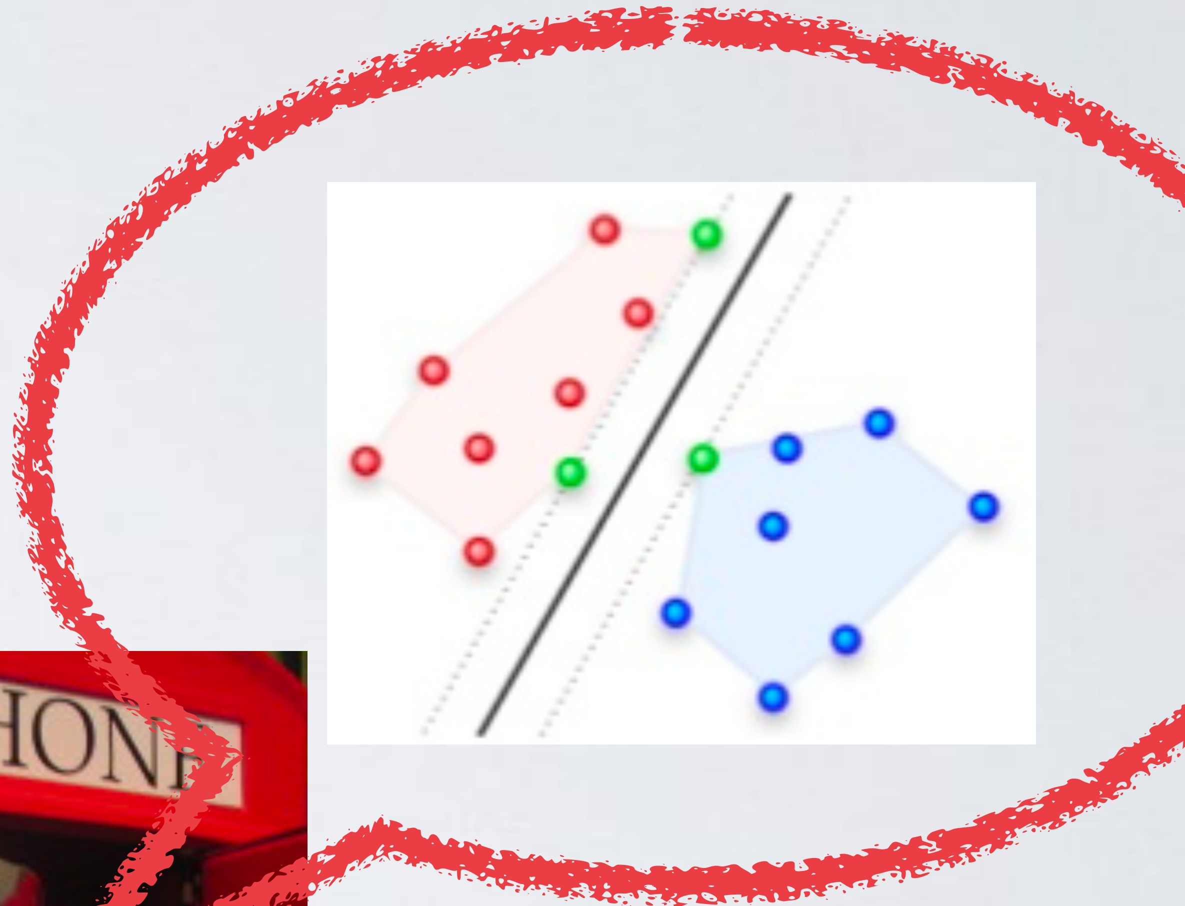


NumPyWren

- However when output matrix is very large it becomes very difficult or expensive to store in memory



- For example for N = 1e6 and D=1e4
  - D x D matrix of doubles is 800 Mb
  - N x N matrix of doubles is 8 TB
- Storing 8 TB in memory traditional cluster is expensive!



Keeping the kernel dream alive!

NumPyWren

- Solution: Use S3 to store matrices, stream blocks to Lambdas to compute output matrix in parallel

N	D	Lambdas	Runtime	Output Size
50000	784	225	192s	20 GB
50000	18432	225	271s	20 GB
1.2 Millon	4096	3000	1320s	11 TB
1.2 Million	18432	3000	2520s	11 TB

Iteration Interface

ITERATION INTERFACE

```
def myfunc(iter_pos, last_state, arg):
    if iter_pos == 0:
        return create_init_state(arg)
    else:
        return next_state(last_state, arg)
```

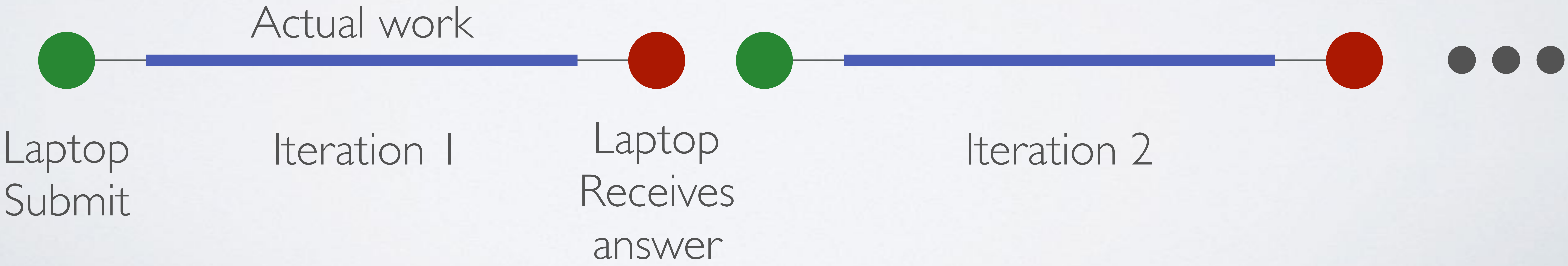
```
def grad_step(k, x_k, alpha):
    if k == 0:
        return np.zeros(N)
    else:
        return x_k + alpha * grad(x_k)
```

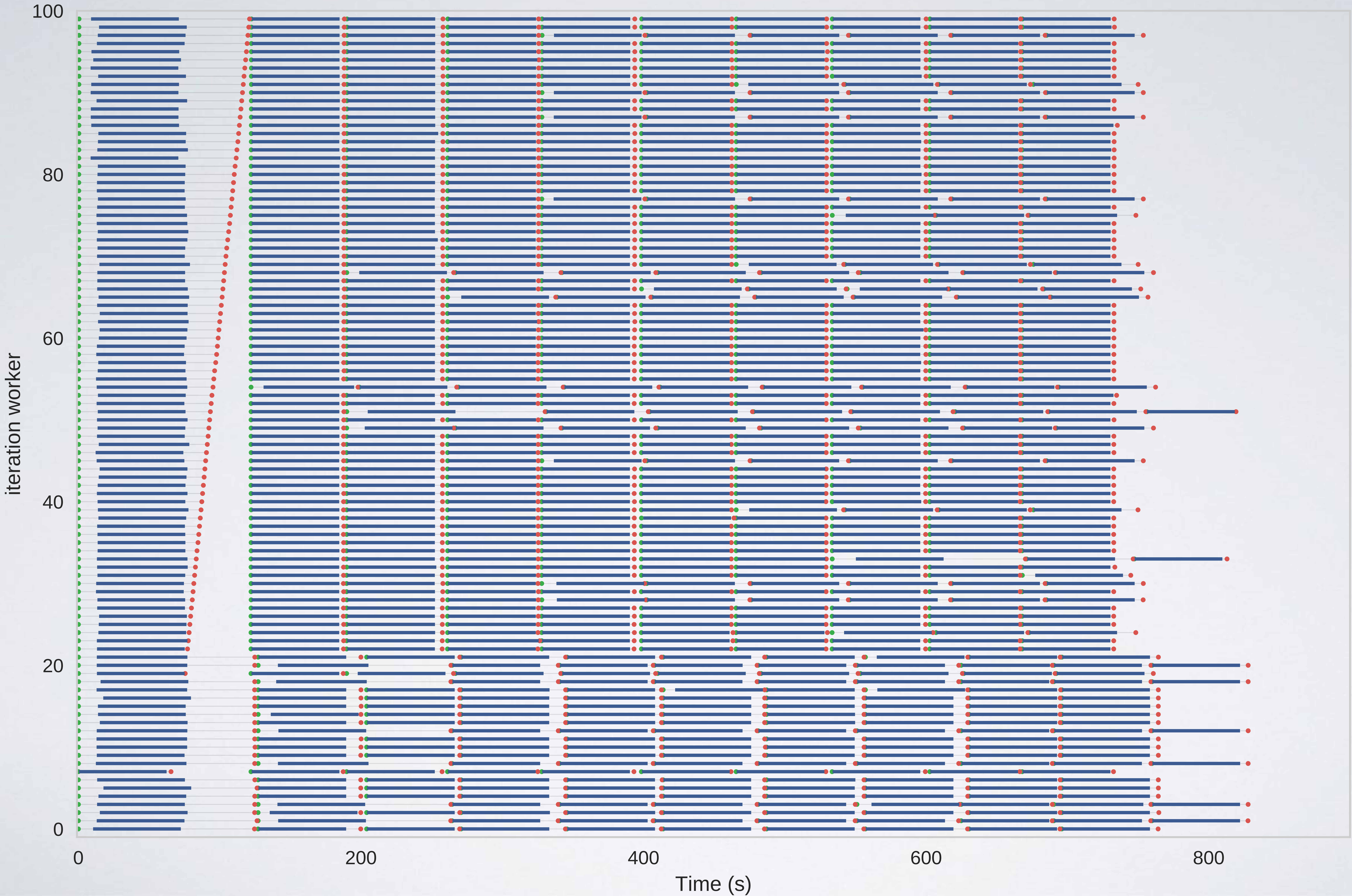
RUNNING THE EXECUTOR

```
wrenexec = pywren.default_executor()

with IterExec(wrenexec) as IE:
     ITER_NUMBER = 100
     ALPHAS = [0.001, 0.01, 0.1]

     iter_futures = IE.map(grad_step, ITER_NUMBER,ALPHAS)

     IE.wait_till_done(iter_futures)
```

SIMPLE EXAMPLE

```python
def offset_counter(k, x_k, offset):
    time.sleep(60)
    if k == 0:
        return offset
    else:
        return x_k + 1
```





100
80
60
40
20
0
iteration worker
0
200
400
600
800
Time (s)


total wallclock time
setup
lambda run time
Start time
compute time
Result return
For a job with 60s iterations
time (wallclock) efficiency
0.76
0.78
0.80
0.82
0.84
0
10
20
30
40
50
60
Total: 97.5%
cost efficiency
0.80
0.85
0.90
0.95
1.00
0
200
400
600
800

TUTORIAL EXERCISES

PyWren RISECamp, 2017

Welcome to the hands-on tutorial for PyWren.

This tutorial consists of a set of exercises that will have you working directly with PyWren:

- basic exercises that introduce you to PyWren APIs (covered in this notebook)
- data analysis on a wikipedia dataset (see analyze-wikipedia.ipynb)
- matrix multiplication with PyWren (see matrix-computations-advanced.ipynb)
- hyperparameter optimization (see hyperparameter-optimization.ipynb)

A couple of notes before you dive into the actual tutorials:

- To run a code cell: select the cell, click Cell -> Run Cells or use Ctrl + Enter.
- ***Execute*** indicates that the following code cell just works as given. Make sure to run them.
- ***Exercise*** indicates an incomplete/broken code cell. Modify the code to make them work.
- You can find solutions for the exercises here

Introduction to PyWren ¶

For this tutorial, we have already installed PyWren in the docker container where this jupyter notebook is running. PyWren provides command line tool that provides basic functionalities for creating AWS IAM roles, configuring PyWren environme deploying/updating Lambda functions, etc. We have also done that for you.
Before we go into the exercises, let's use the command line tool to test if PyWren works properly.

Execute the cell below ().
If PyWren is correctly installed, you should see `function returned: Hello world` after a few seconds.

pywren-intro.ipynb

PyWren RISECamp, 2017

Data Analytics with PyWren

In this section, we will use PyWren explore a dataset of Wikipedia records.

0. The Data

We've prepared an S3 bucket with 20GB of Wikipedia traffic statistics data obtained from http://aws.amazon.com/datasets/4182. T make the analysis more feasible for the short time you're here, we've shortened the dataset to three days worth of data (May 5 t May 7, 2009; roughly 20G and 329 million entries).

Let's take a look into the bucket with our dataset. We'll print a few files from a few files from our bucket.

Execute the code below to print out the names of the first 20 files.

In [ ]:

```
# These lines are only needed for the solutions.
import sys
sys.path.append("..")

# some libraries that are useful for this tutorial
from training import wikipedia_bucket, list_keys_with_prefix, read_from_s3

filenames = list_keys_with_prefix(wikipedia_bucket, "wikistats_20090505_restricted-01/")
for filename in filenames[:20]:
    print(filename)
```

analyze-wikipedia.ipynb

Hyperparameter optimization for machine learning

Many machine learning models have hyperparamters -- parameters that control some aspect of the model. The exact setting of these hyperparameters can dramatically impact the performance of your underlying model. Fortunately, most hyperparameters can be tried in parallel, making the task of *hyperparemter optimization* a great fit for PyWren.

Here we use a simple dataset included in scikit-learn to show how to do hyperparameter optimization across a number of different datasets, and a number of different cross-validations

In [5]:

```
%pylab inline
import pywren
import sklearn
import seaborn as sns
import itertools
import pandas as pd
from sklearn.model_selection import train_test_split
import sklearn.svm

from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline, Pipeline
```

Populating the interactive namespace from numpy and matplotlib

get the data

First we load in the data from scikit learn and examine it. Here we will be using an existing dataset of breast cancer tumor properties that's shipped with scikit-learn. This is a small binary classification problem, and the hyperparameter optimization we are doing here

hyperparameter-optimization.ipynb

Large Scale Matrix Computations

In this notebookwe will walk through some of the more advanced things you can achieve with PyWren. Namely using S3 as a backing store we will implement a nearest neighbor classifier algorithm.

In [ ]:

```
%pylab inline
import boto3
import cloudpickle
import itertools
import concurrent.futures as fs
import io
import numpy as np
import time
from importlib import reload
from sklearn import metrics
import pywren
import pywren.wrenconfig as wc
import itertools
from operator import itemgetter
import matrix
```

In [ ]:

```
DEFAULT_BUCKET = wc.default()['s3']['bucket']
```

1. Matrix Multiplication

One nice thing about PyWren is it allows users to integrate existing python libraries easily. For the following exercise, we are going to

matrix-computations-advanced.ipynb

OUR VISION

- Map for everyone
- Transparent language support
- Transparent elasticity
- Unlimited fast storage

THANK YOU!

pywren.io

Shivaram Venkataraman · Qifan Pu · Vaishaal Shankar · Allan Peng · Ion Stoica · Ben Recht