Apache Hadoop YARN: Yet Another Resource Negotiator

Presenters: Carlo Curino & Chris Douglas

Microsoft Berkeley UNIVERSITY OF CALIFORNIA

Vinod Kumar Vavilapalli
Arun C Murthy
Sharad Agarwal
Mahadev Konar
Robert Evans
Thomas Graves
Jason Lowe
Hitesh Shah
Siddharth Seth
Bikas Saha
Owen O'Malley
Sanjay Radia
Benjamin Reed
Eric Baldeschwieler

Apache Hadoop YARN: Yet Another Resource Negotiator

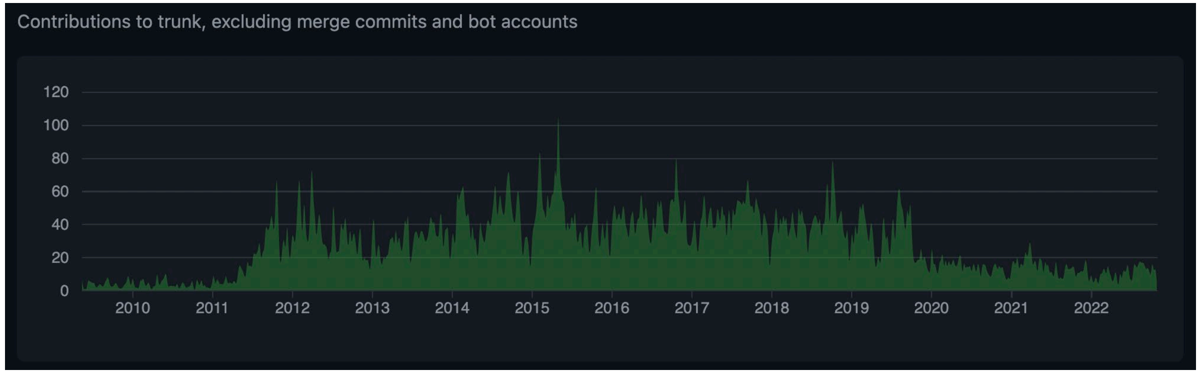
Vinod Kumar Vavilapalli[h] Arun C Murthy[h] Chris Douglas[m] Sharad Agarwal[i]
Mahadev Konar[h] Robert Evans[y] Thomas Graves[y] Jason Lowe[y] Hitesh Shah[h]
Siddharth Seth[h] Bikas Saha[h] Carlo Curino[m] Owen O'Malley[h] Sanjay Radia[h]
Benjamin Reed[f] Eric Baldeschwieler[h]

shared and accessed. This broad adoption and ubiquitous usage has stretched the initial design well beyond its intended target, exposing two key shortcomings: 1) tight coupling of a specific programming model with the resource management infrastructure, forcing developers to abuse the MapReduce programming model, and 2) centralized handling of jobs' control flow, which resulted in endless scalability concerns for the scheduler.

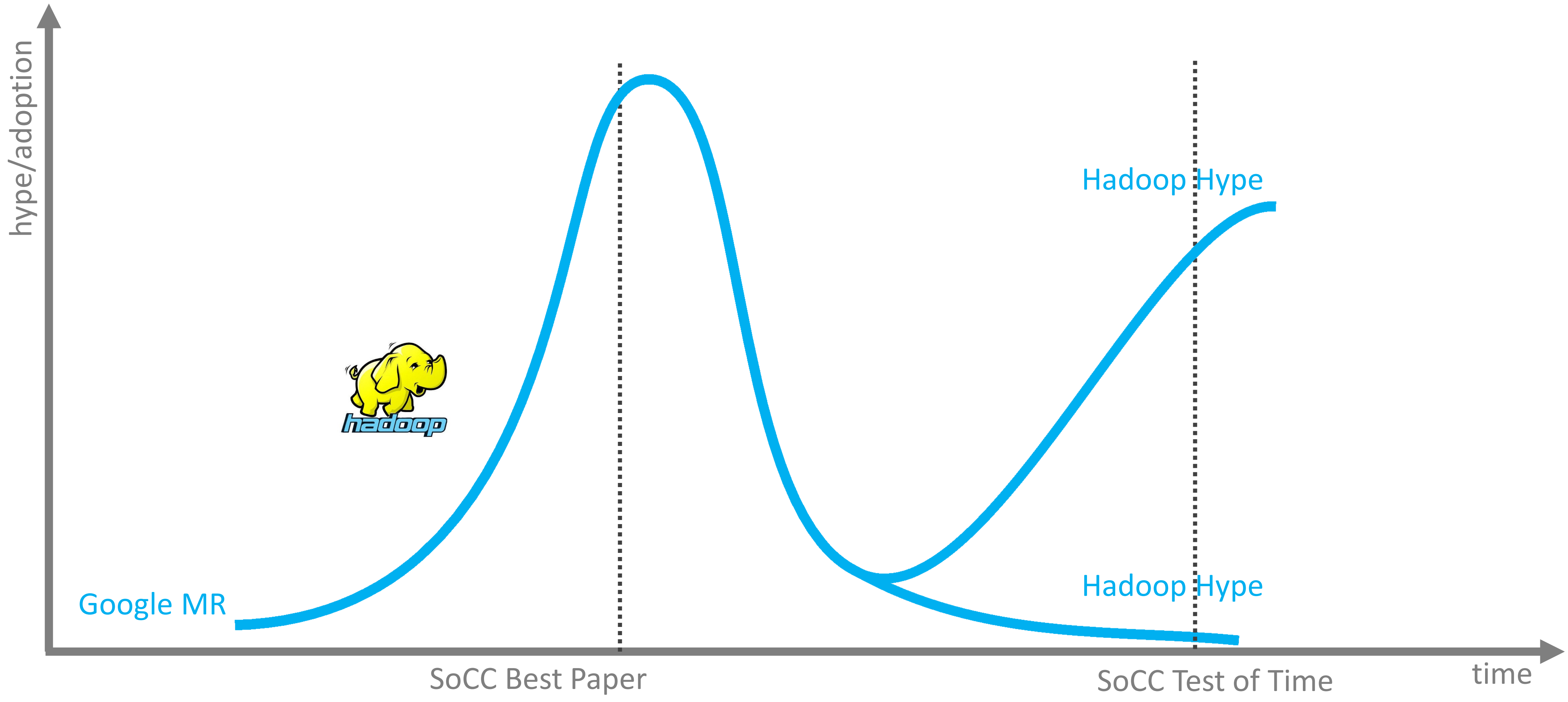
In this paper, we summarize the design, development, and current state of deployment of the next generation of Hadoop's compute platform: *YARN*. The new architecture we introduced decouples the programming model from the resource management infrastructure, and delegates many scheduling functions (e.g., task fault-tolerance) to per-application components. We provide experimental evidence demonstrating the improvements we made, confirm improved efficiency by reporting the experience of running YARN on production environments (including 100% of Yahoo! grids), and confirm the flexibility claims by discussing the porting of several

Apache Hadoop began as one of many open-source implementations of MapReduce [12], focused on tackling the unprecedented scale required to index web crawls. Its execution architecture was tuned for this use case, focusing on strong fault tolerance for massive, data-intensive computations. In many large web companies and startups, Hadoop clusters are the common place where operational data are stored and processed.

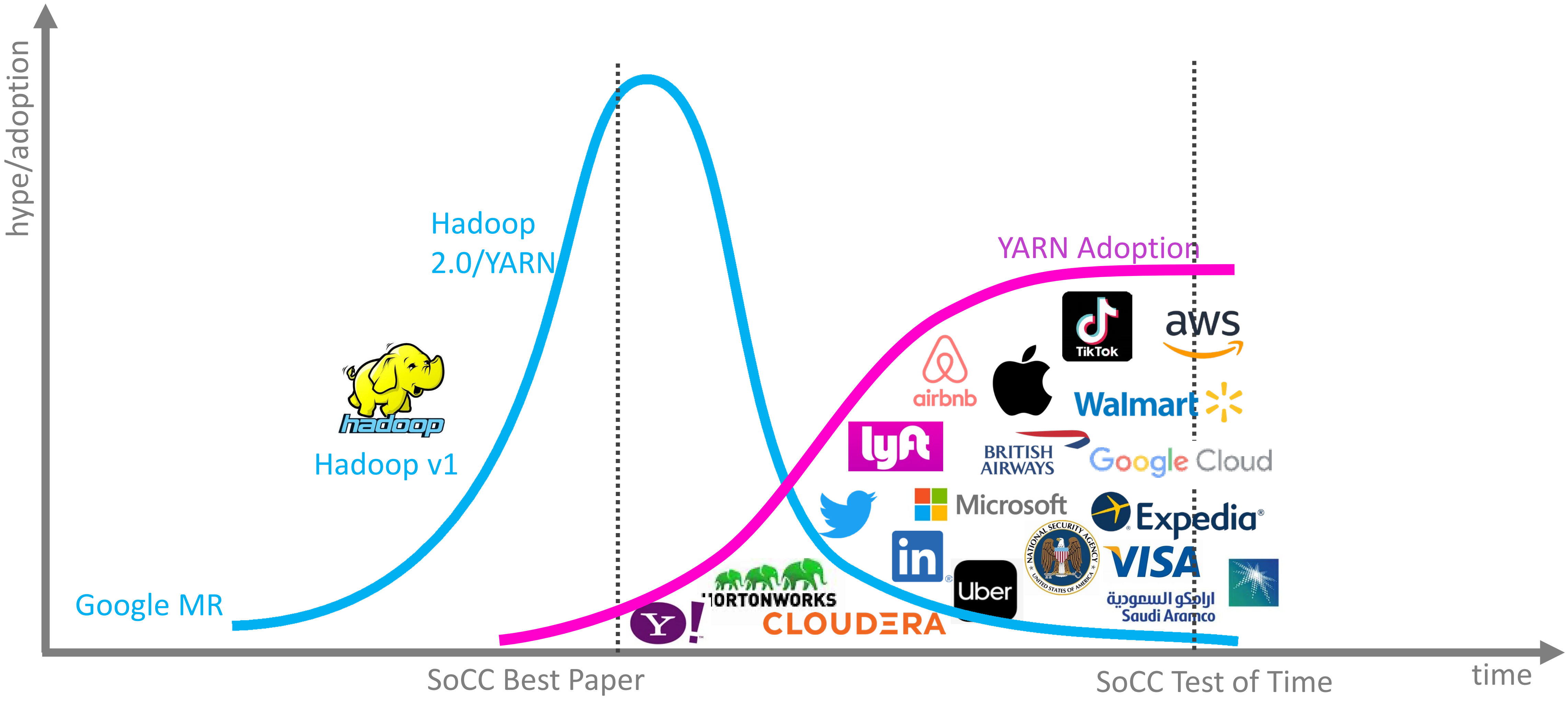
More importantly, it became *the* place within an organization where engineers and researchers have instantaneous and almost unrestricted access to vast amounts of computational resources and troves of company data. This is both a cause of Hadoop's success and also its biggest curse, as the public of developers extended the MapReduce programming model beyond the capabilities of the cluster management substrate. A common pattern submits "map-only" jobs to spawn arbitrary processes in the cluster. Examples of (ab)uses include forking web servers and gang-scheduled computation of iterative workloads. Developers, in order to leverage the



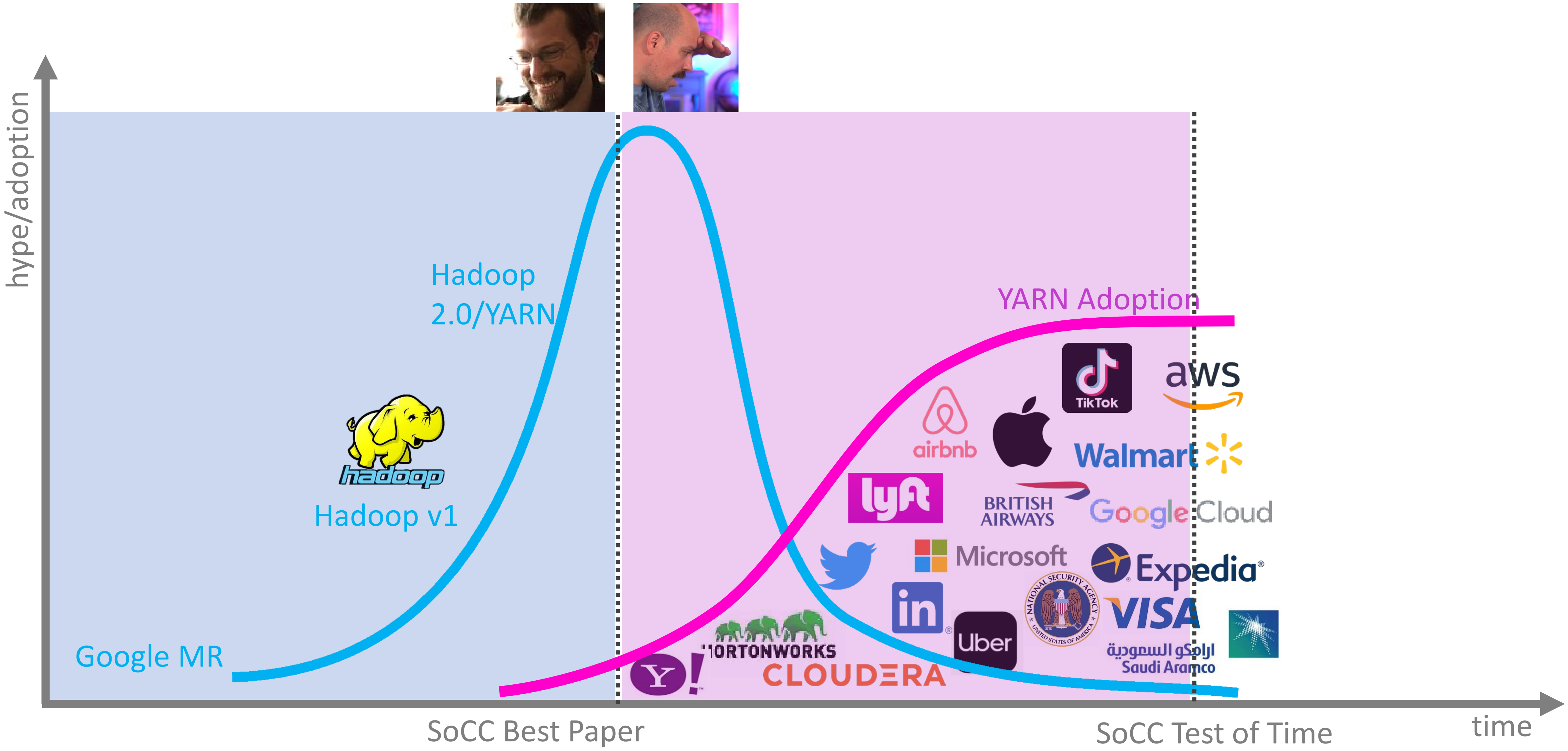
Test of ~~Time~~ Timing Award



Apache Hadoop YARN Hype/Adoption Cycle

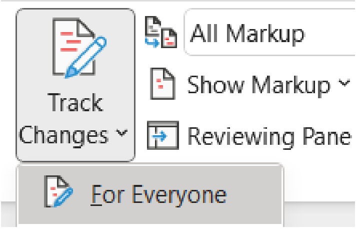


Apache Hadoop YARN Hype/Adoption Cycle



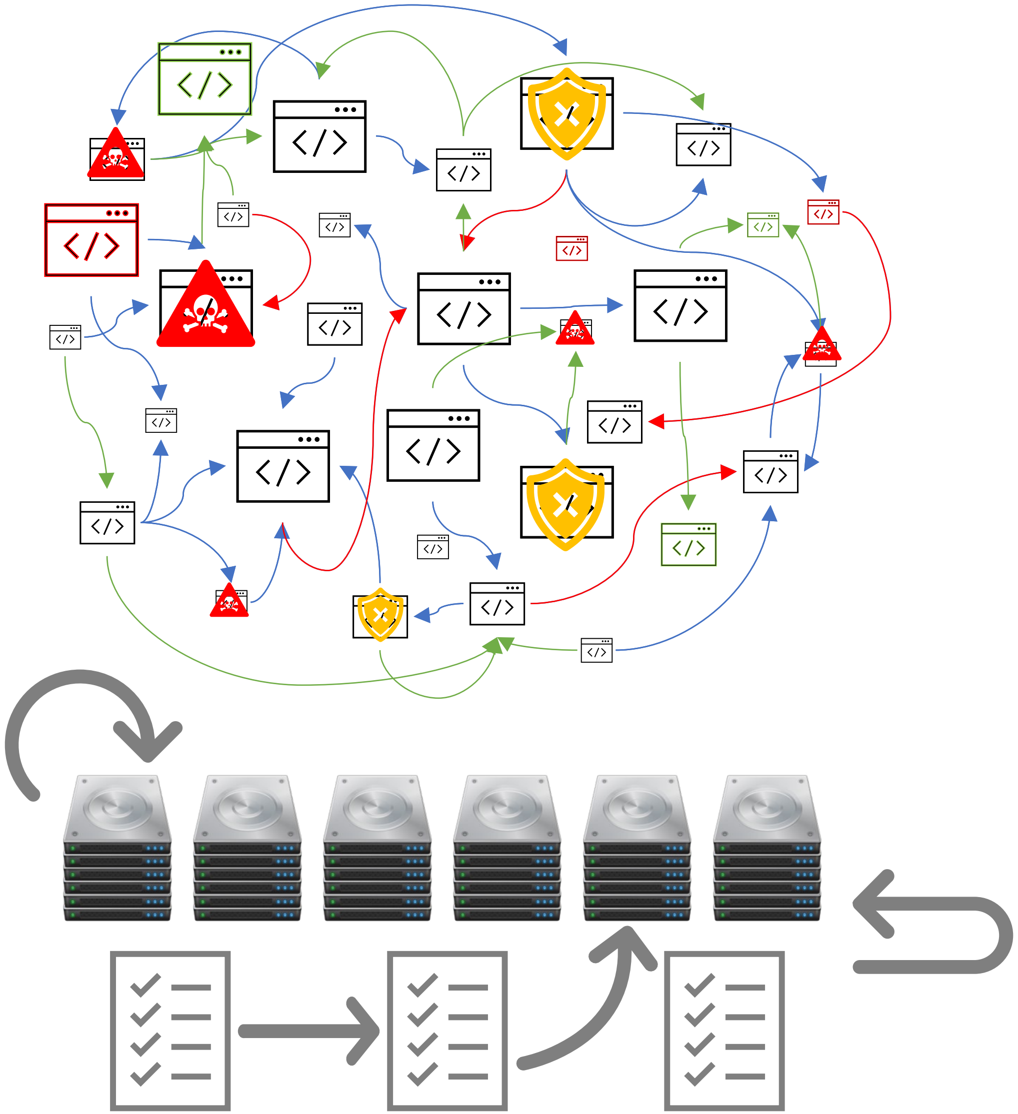
Scaling with the Web (2000s)

- Largest, most dynamic dataset ever assembled

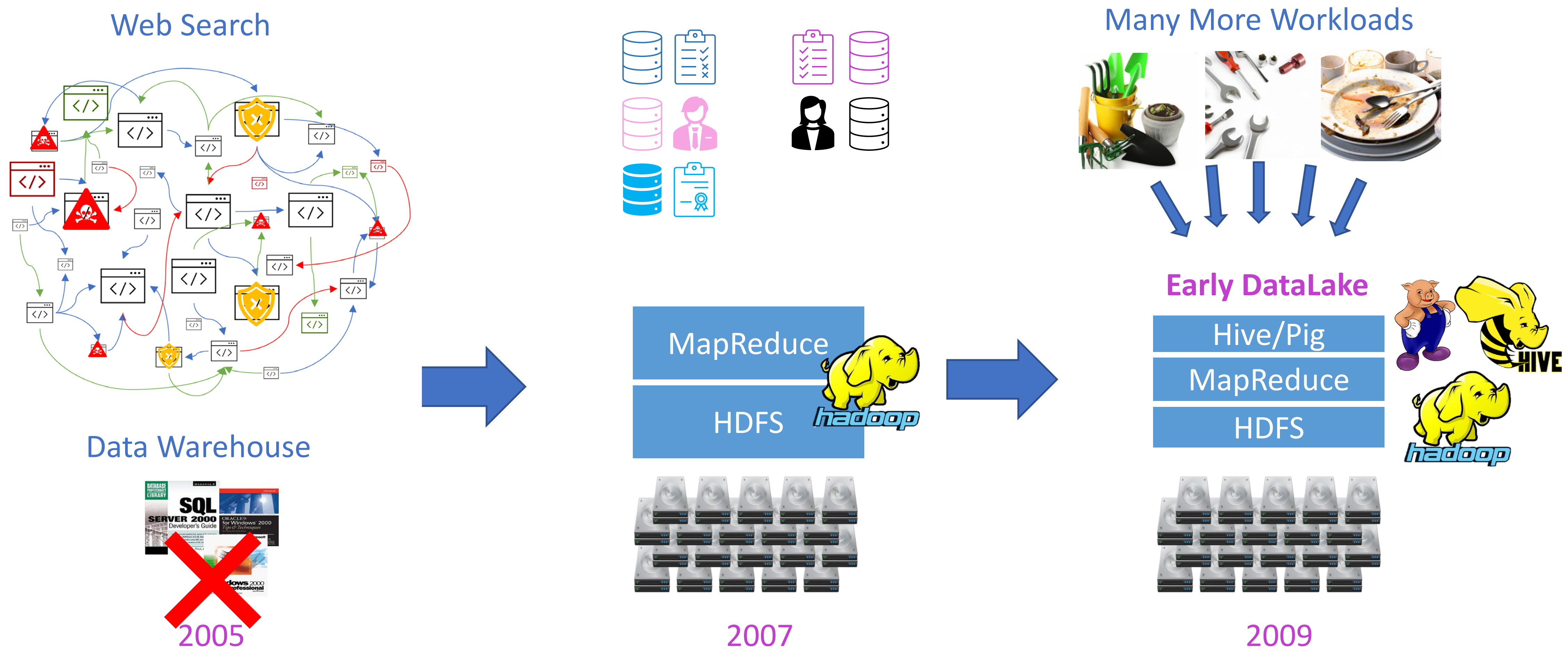
- 

- Annotation dimensions

- Metadata expand with base data



Data Silos to Data Lake



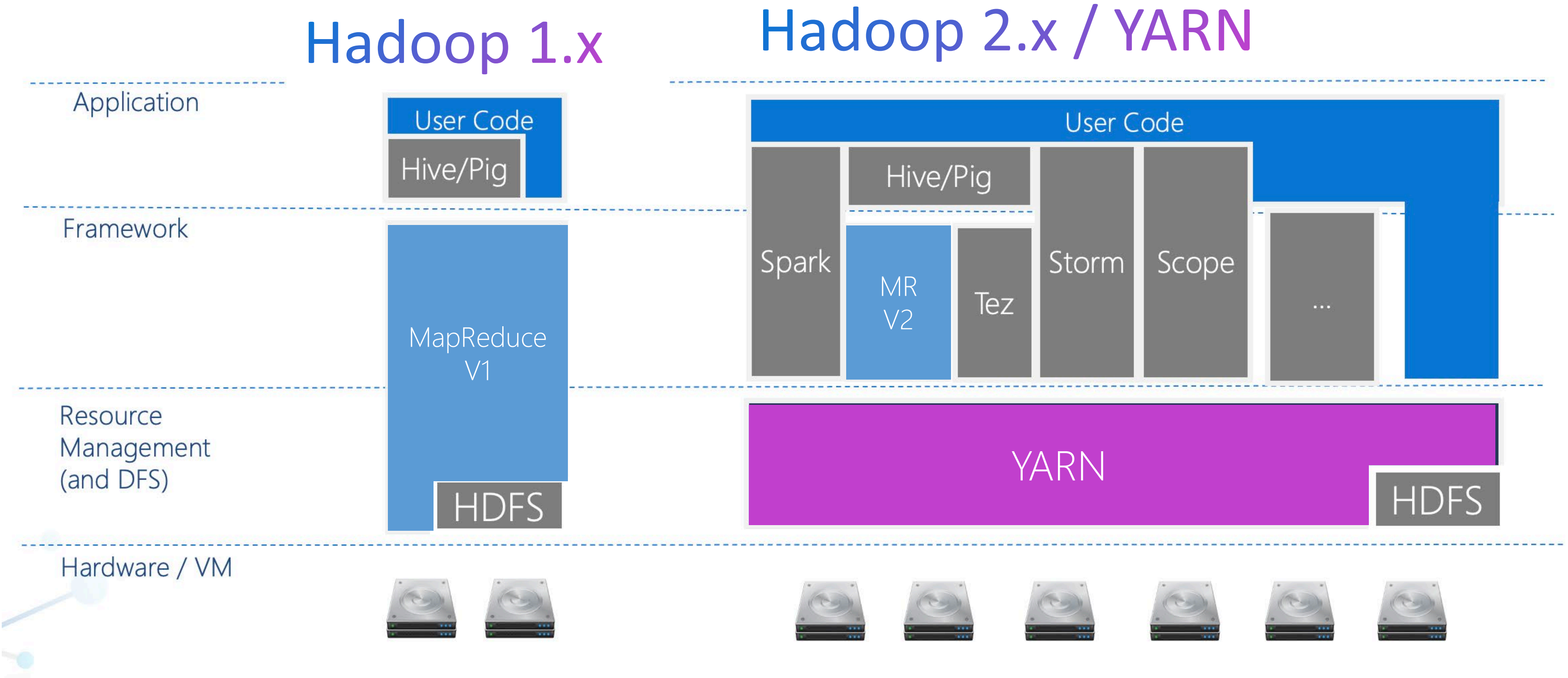
YARN+HDFS → Cluster Operating System

MapReduce was not enough

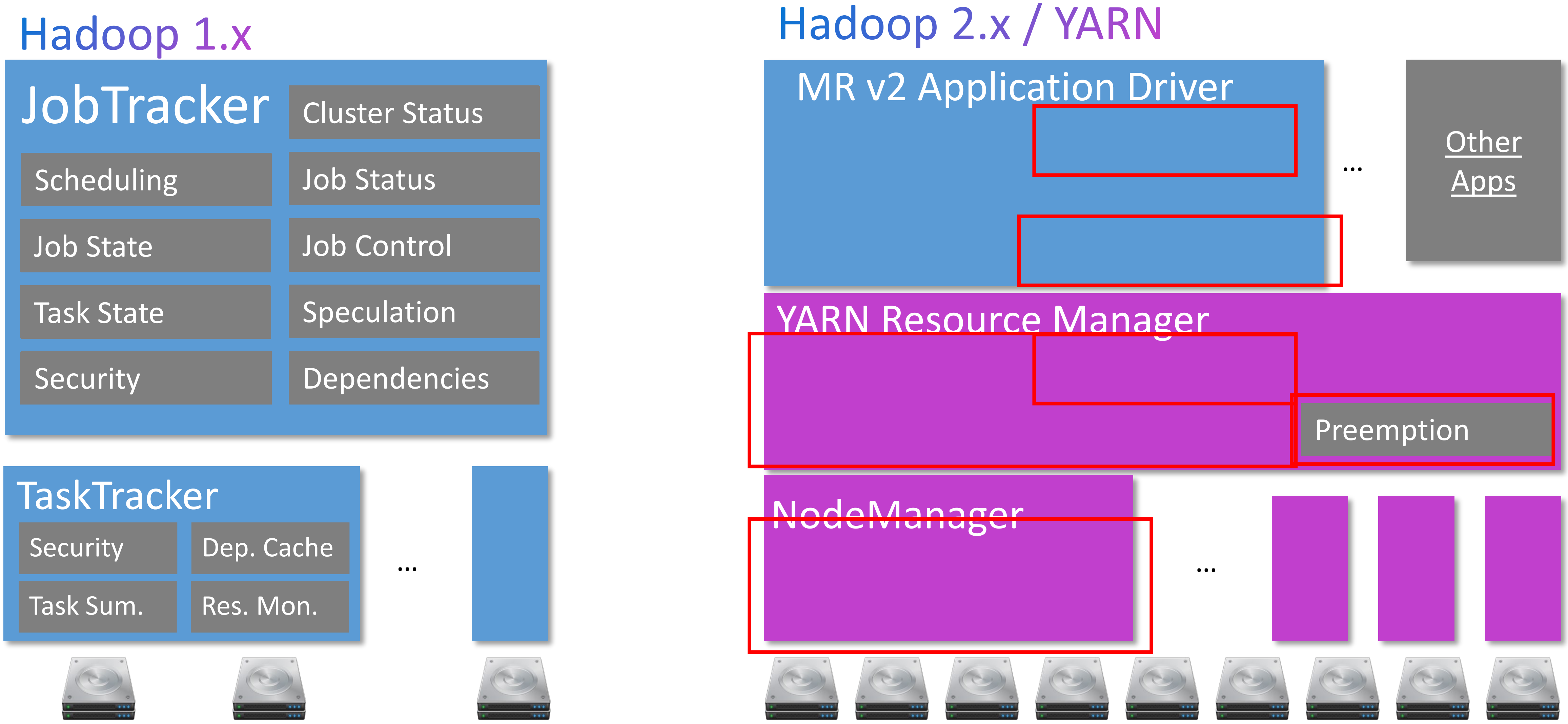
Rigid Programming model,
batch-oriented, low utilization

Decouple Resource Management from App

YARN handles Resource Management
MRv2, Spark, Scope,... are "apps" sharing a cluster



MapReduce to a Cluster OS (YARN)



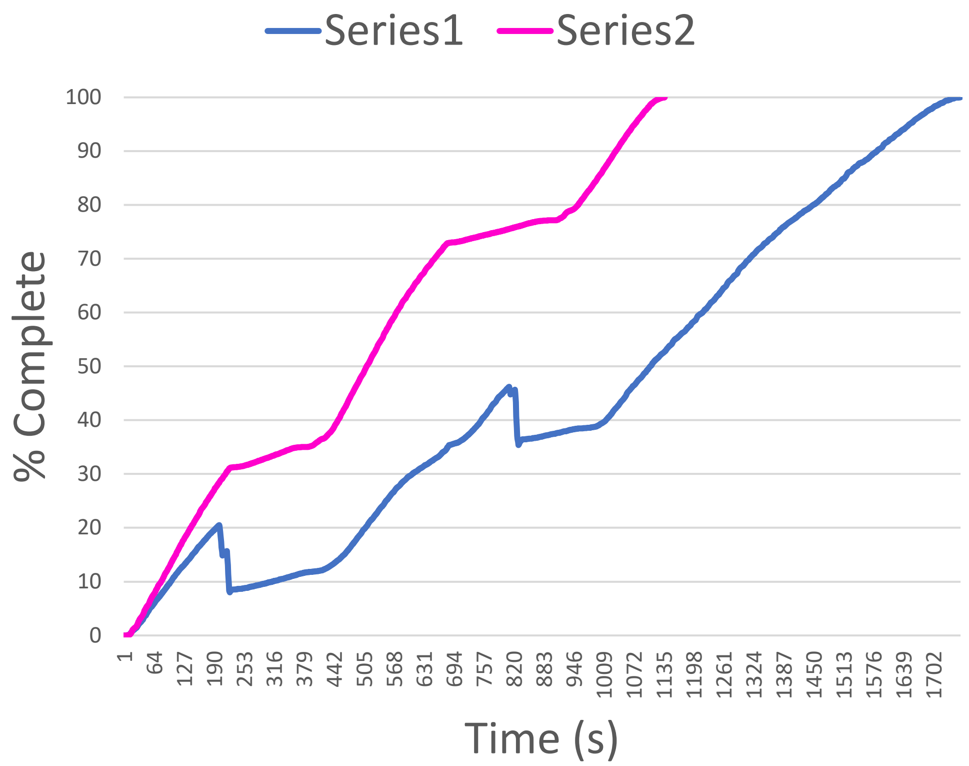
YARN Preemption: Why

OS Scheduling: bin packing over time

- Workload changes affect allocations
- Preemption to evict process from resources
- Fallow resources to absorb changes

Cluster scheduling: dynamic bin packing

- (Temporarily) Failures cause reconfiguration
- Killing tasks loses progress



YARN Preemption: How

Diverse applications

Dependencies, costs opaque to RM

Announce need for resource revocation

Symmetric protocol for requests

Specific nodes, filter expression

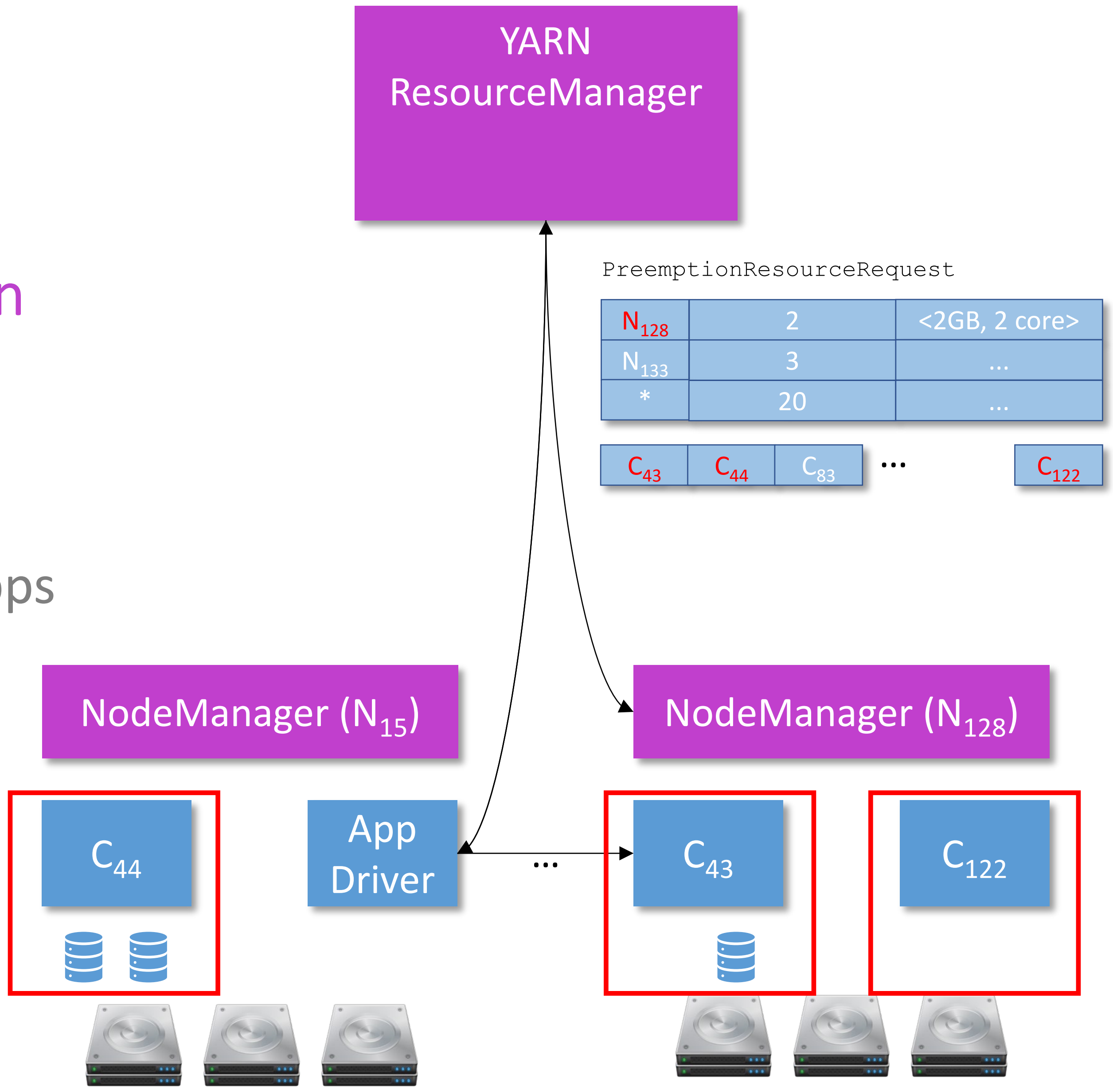
Apps select satisfying set of resources

System-enforced for non-cooperating apps

New capability in YARN

Agent to negotiate

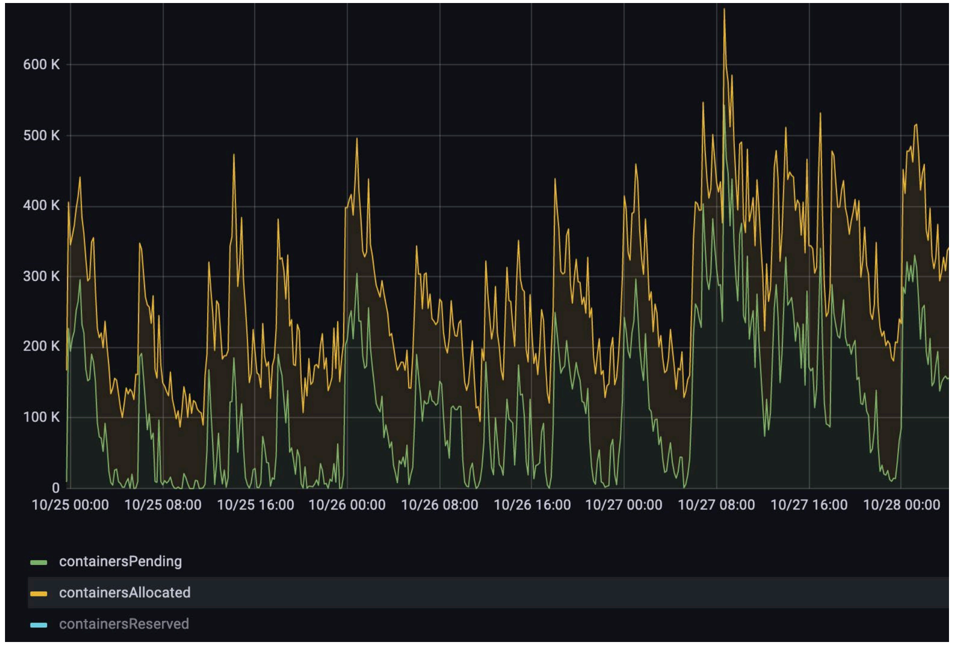
Capacity for application logic



YARN@in: Container Utilization (two weeks ago)

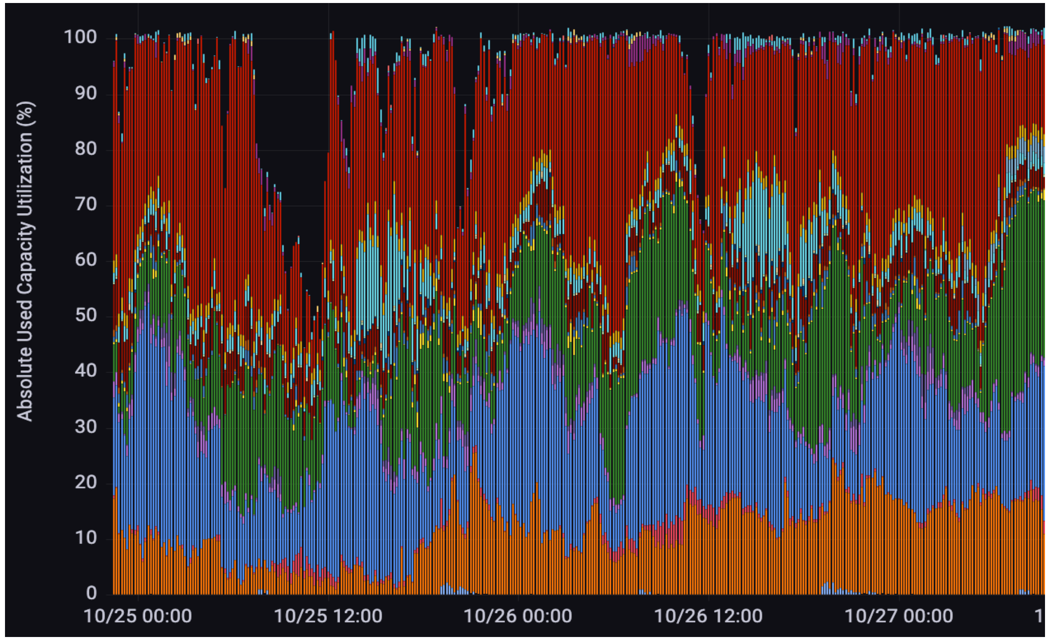
Very Diverse Workload

- Multiple frameworks
- Spark has most resources
- MR has most containers



Preemption -> High Utilization

- Many queues (per organization/user)
- 80-90% container utilization
- Clusters of 7-10k nodes



Screenshots courtesy of: Varun Saxena (LinkedIn)

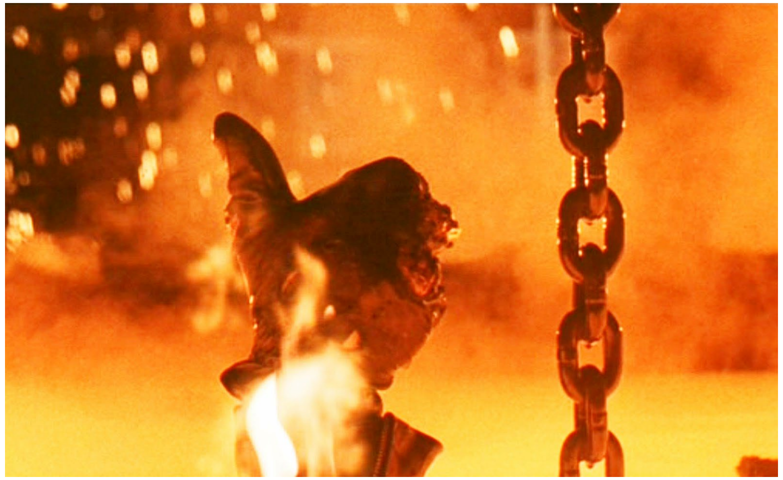
Second System: Lessons Learned

Existing system is your fiercest competitor

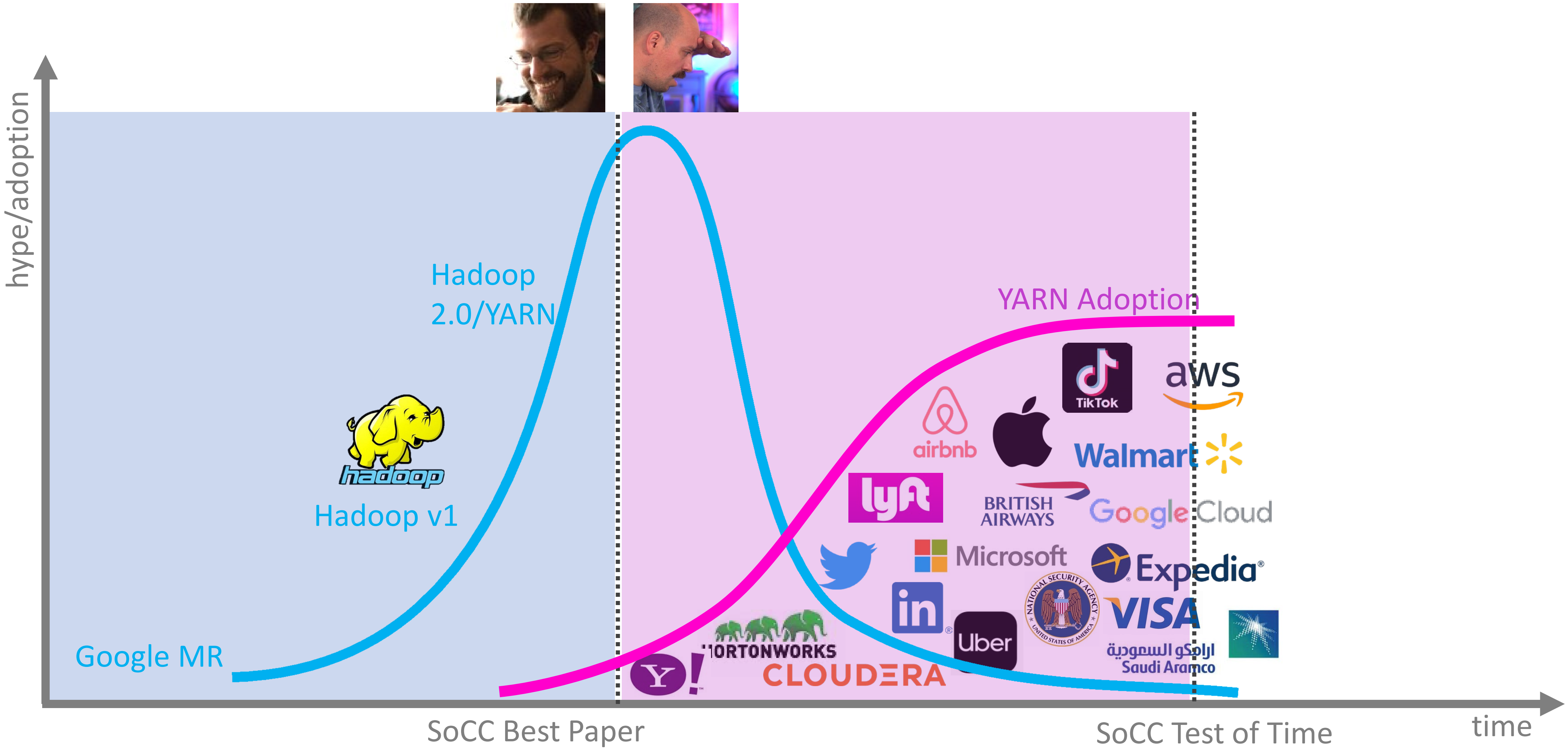
- Share its defaults to share its strength
- Same >> Improved: **until you're deployed**

Software is easier to change than people

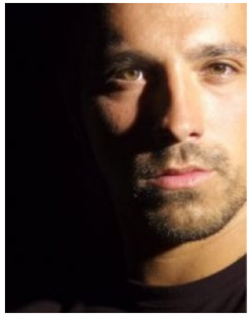
- Change harms adoption
- Start structural, not functional



Apache Hadoop YARN Hype/Adoption Cycle



My Vantage Point..



Carlo at CSAIL MIT (2009):
Stonebraker: "[MapReduce] is a giant step backward."

DBs are the best, SQL is the way!

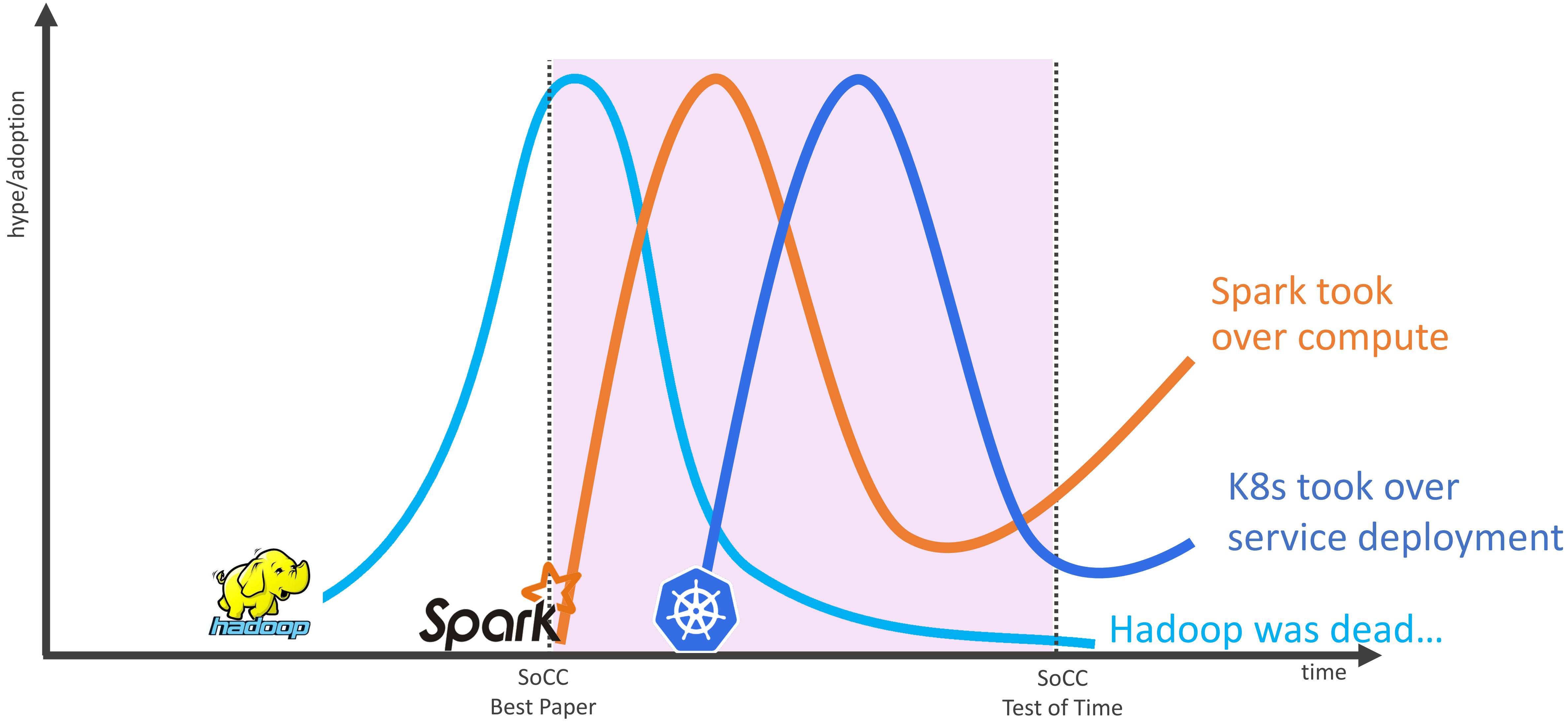
Carlo at Yahoo experience (2011):
Need to process some logs, two options:
*a) **Use MySQL:** Weeks of red-tape*
*b) **Learn Hadoop:** Thousands of servers with data preloaded*

The "DataLake" experience is Exhilarating!

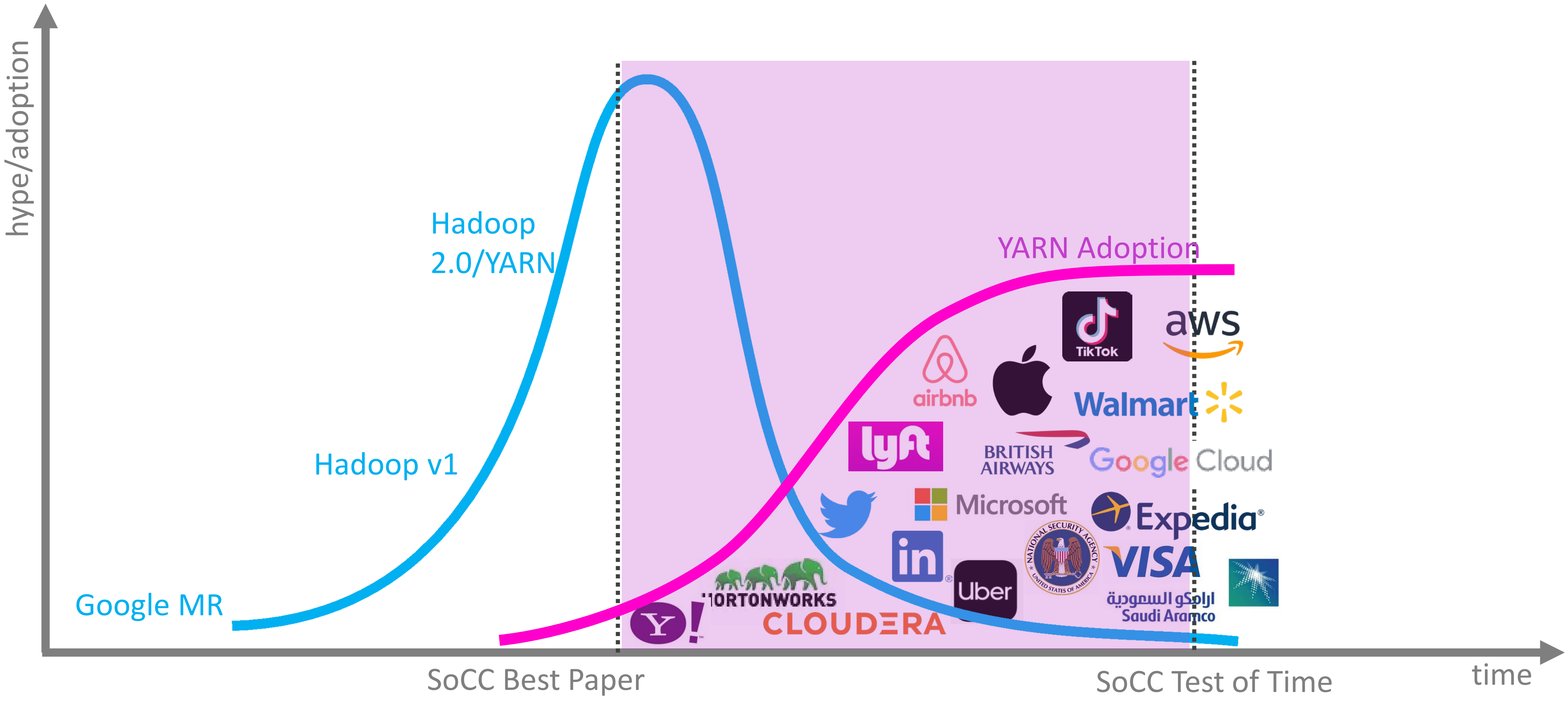
Carlo at Microsoft (hindsight 2020):
*Through OSS and careful API design. We **can** have it all (scale, efficiency, flexibility)!*

Make your sandbox open(-source) and friendly. Academics and other companies will come!

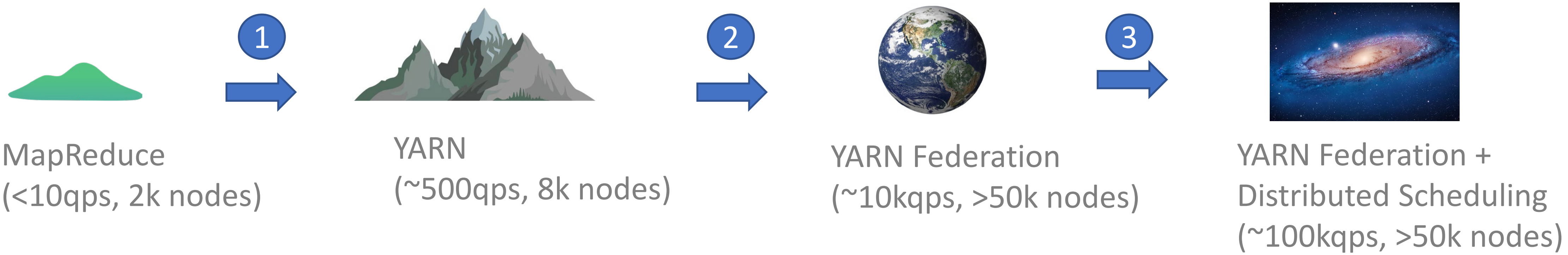
Apache Hadoop YARN Hype/Adoption Cycle



Apache Hadoop YARN Hype/Adoption Cycle



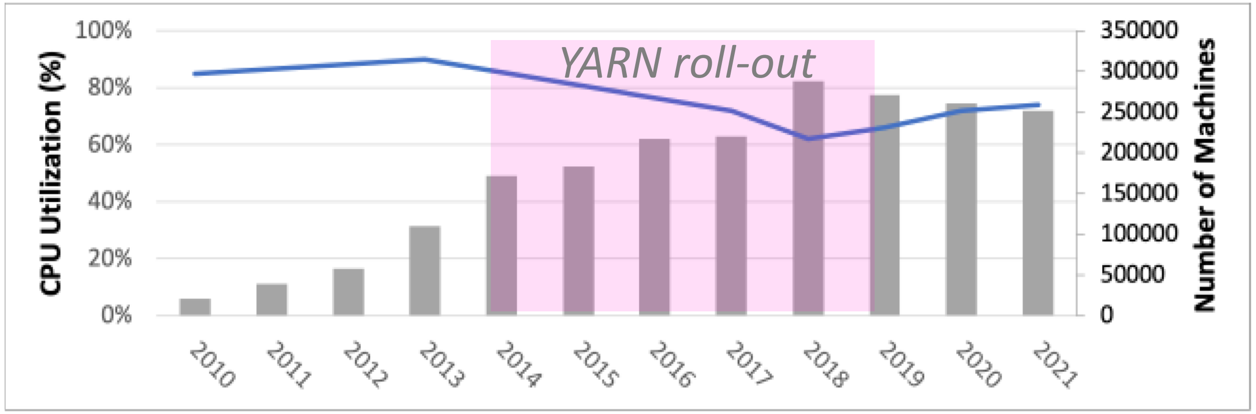
Why YARN adoption among Hyperscalers?



Scale and High Utilization

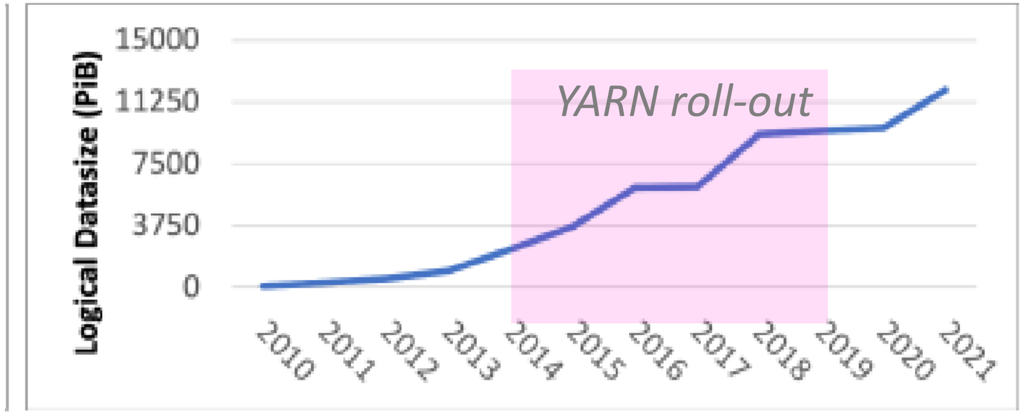
Many of the largest/busiest BigData clusters are YARN-based
How? We progressively “removed” responsibilities from central components

BigData at Microsoft

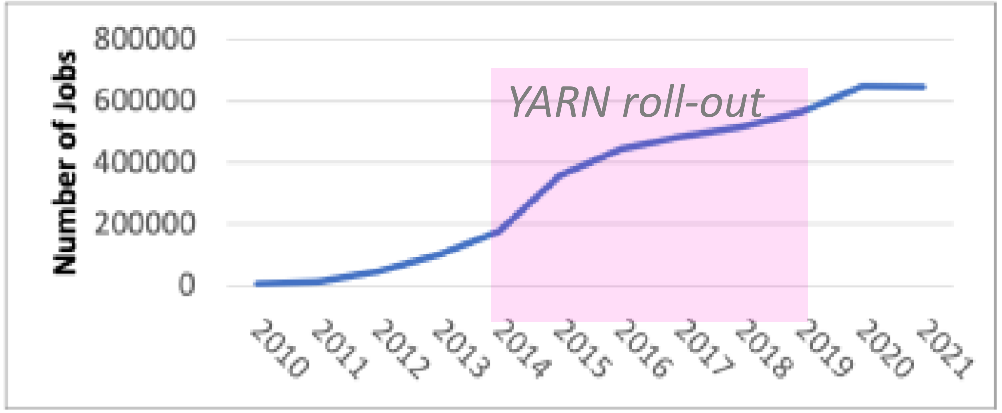


User: ~15k
Queue: >5k
Job: 600k / day
Task: 5 Billions / day
Cluster: >50K nodes
Data: >10Exabyte

Figure 5: CPU utilization over the years in Cosmos.



(b) Data size before compression/replication.



(c) Number of batch SCOPE jobs run per day.

* *Pictures from* "[The Cosmos Big Data Platform at Microsoft: Over a Decade of Progress and a Decade to Look Forward]"

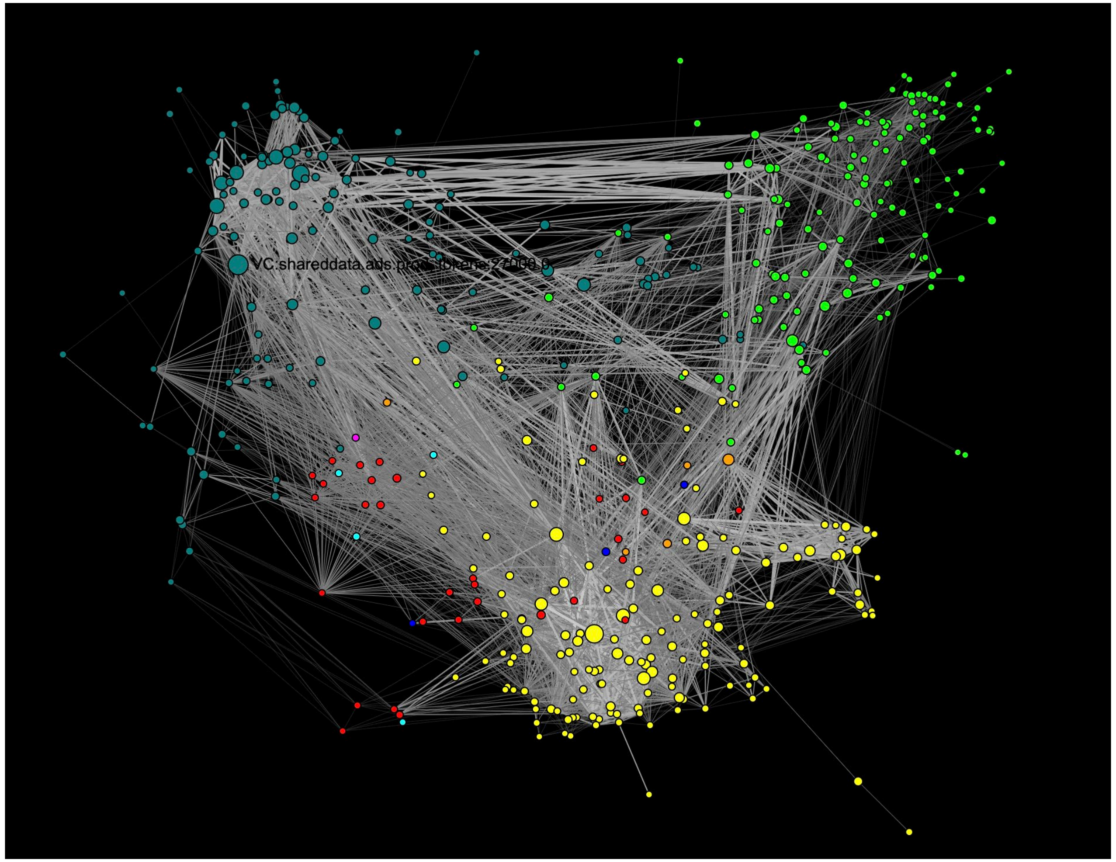
Why not partitioning to smaller clusters?

Individual very-large jobs (~16k servers)

Usability/Fragmentation advantages

DataLake Effect (visualized):

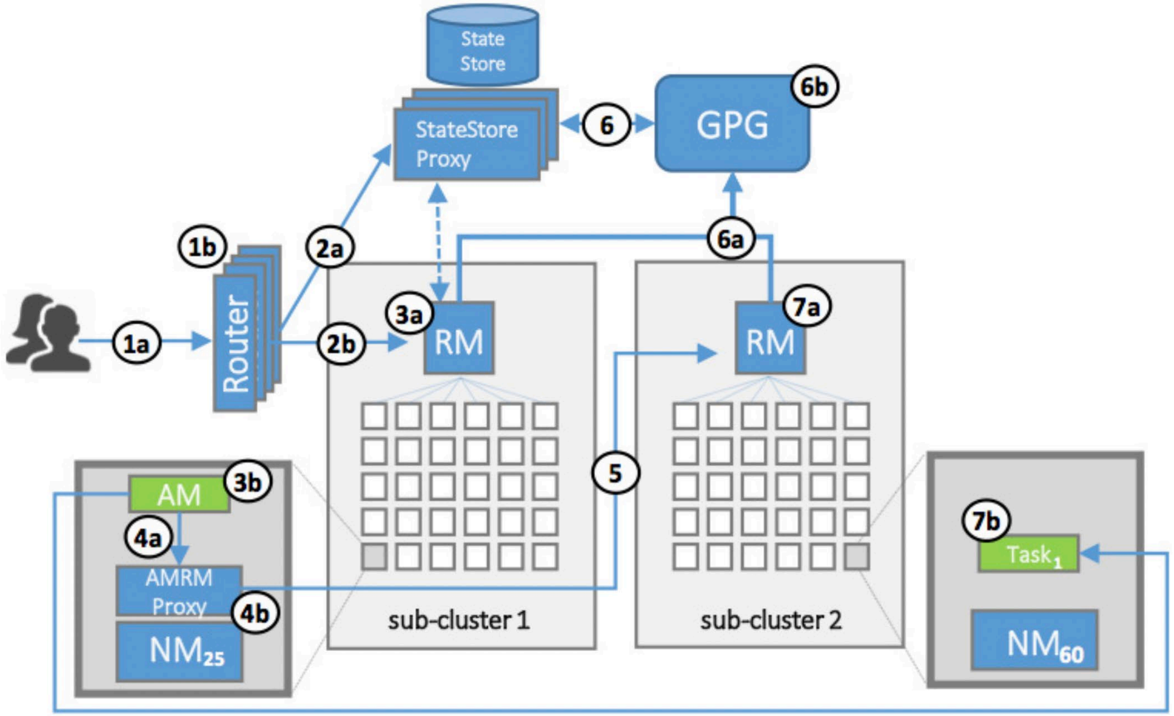
- Nodes represent 2k queues
- Edges represent “data sharing”



YARN Federation*

Illusion of single-cluster

- Proxy all interactions with the system
- Centrally define capacity policies
- Routers/RM/AMRMProxy choreograph enforcement
- AMRMProxy/RM coordinate to handle “placement” decisions



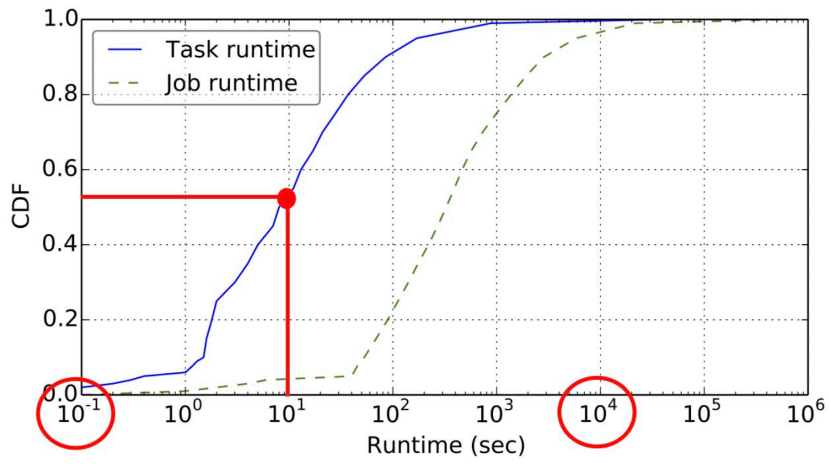
* Hydra: a federated resource manager for data-center scale analytics, NSDI 2019

YARN Distributed Scheduling*

Challenge

Scheduling rates at 50k nodes with short tasks

Heartbeat latencies waste resources (~10sec from free-to-used resources)



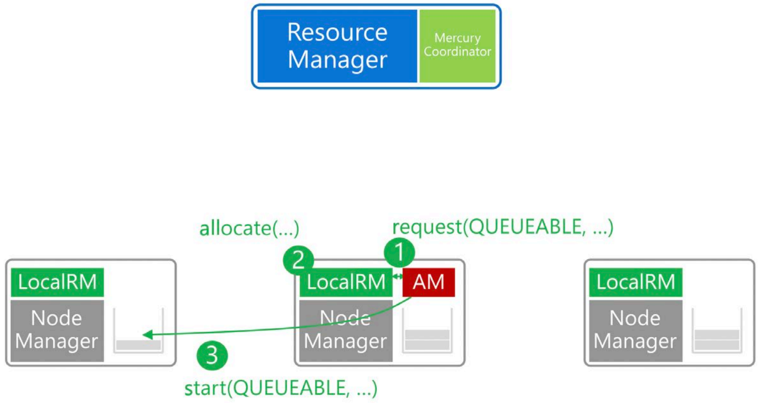
Key idea: Distributed Scheduling

Introduce Opportunistic Containers

Per-node LocalRM makes scheduling decisions

OC queue at NM and are locally preempted

Quotas/Queue priorities/Rebalancing

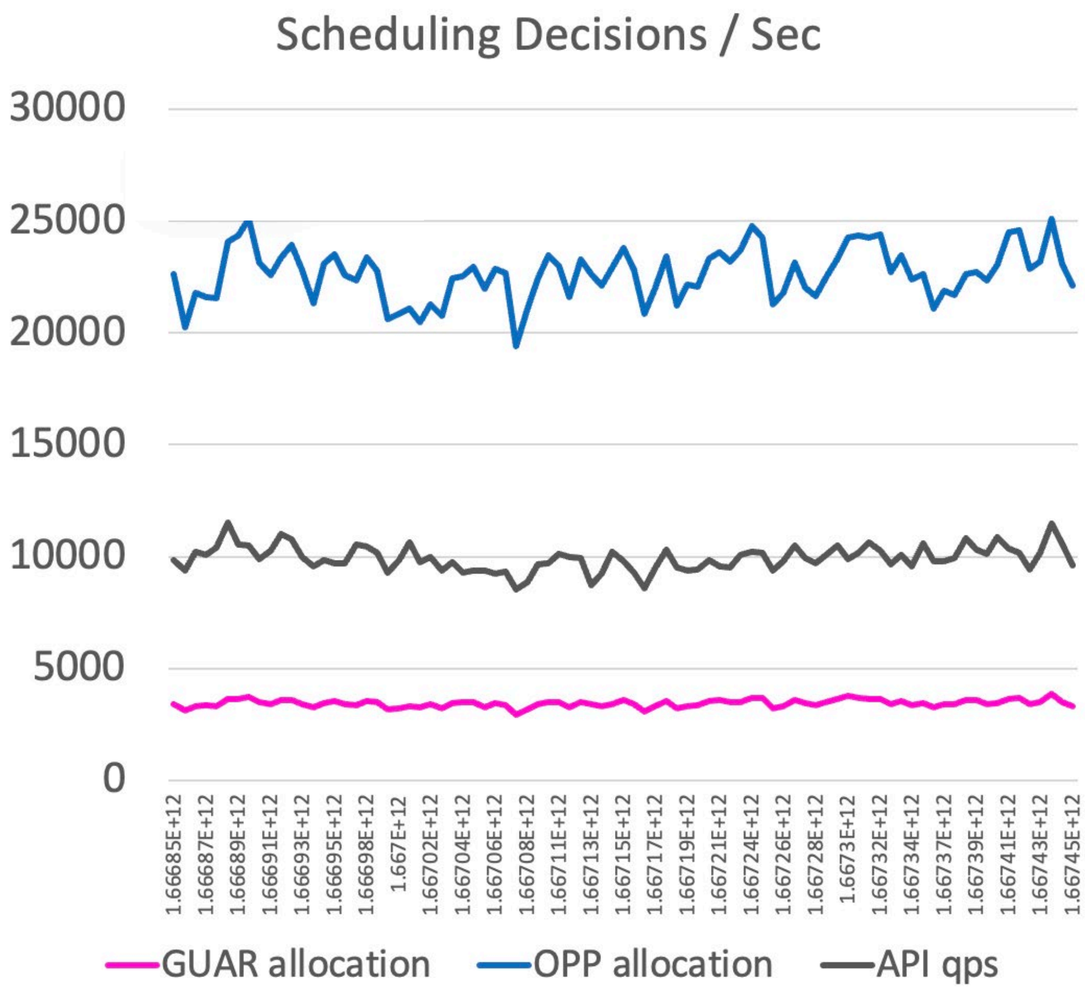


* Mercury: Hybrid Centralized and Distributed Scheduling in Large Shared Clusters, ATC 2015

YARN @ Microsoft (federation+OppCont)

Scheduling Pressure:

~40k qps average
>100k qps at peak
~50k queued OPP



Screenshots courtesy of Cosmos Team (Microsoft)

Why is YARN used for small-scale OnPrem/Cloud?

Battle-tested/non-differentiating functionalities

Scalable Scheduler, fault tolerance, library management, rolling-upgrades, load balancing, log aggregation, metrics gathering, node health, job cleanup, …

Cluster-OS + OSS APIs

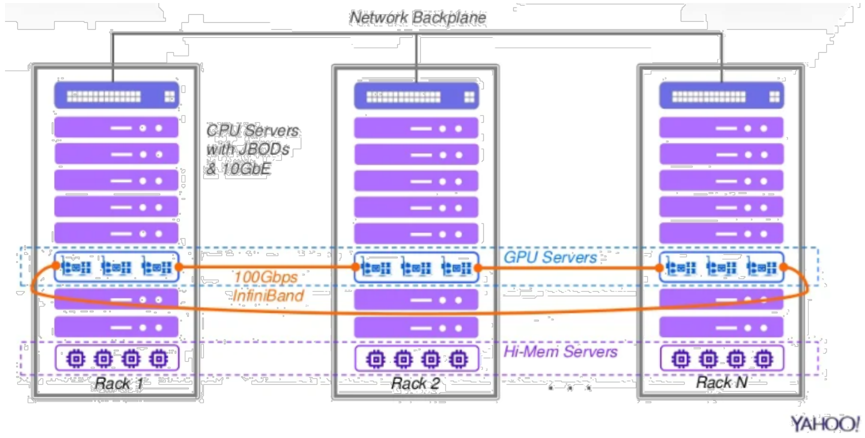
OSS APIs targeted by most “new” frameworks
Customizable deployments

YARN @ yahoo!: Example of Customization

Availability at 99.999%!!

Diverse set of workloads

Diverse HW (node-labels)



Screenshots courtesy of: Eric Badger (Yahoo!)

Conclusions

Origin Story:

Web-Index →BigData → DataLake → YARN

YARN Adoption:

Scalable/High Utilization Cluster OS

OSS APIs and Sturdy Core Features

Next Decade:

New powerful trends towards Cloud and SaaS → ?

THANK YOU

Presenters

Carlo Curino
Chris Douglas

Authors

Vinod Kumar Vavilapalli
Arun C Murthy
Chris Douglas
Sharad Agarwal
Mahadev Konar
Robert Evans
Thomas Graves
Jason Lowe
Hitesh Shah
Siddharth Seth
Bikas Saha
Carlo Curino
Owen O'Malley
Sanjay Radia
Benjamin Reed
Eric Baldeschwieler

Hadoop PMC

Akira Ajisaka
Arun C Murthy
Anu Engineer
Amareshwari Sriramadasu
Arpit Agarwal
Arun Suresh
Aaron T. Myers
Ayush Saxena
Bikas Saha
Billie Rinaldi
Bibin A Chundatt
Robert(Bobby) Evans
Brahma Reddy Battula
Brandon Li
Carlo Curino
Chris Douglas
Chen Liang
Colin Patrick McCabe
Chris Nauroth
Doug Cutting
Daryn Sharp
Devaraj Das
Devaraj K
Dhruba Borthakur
Eric Badger
Eli Collins
Enis Soztutar
Eric Payne
Giridharan Kesavan
Haibo Chen
Hairong Kuang
He Xiaoqiao
Hitesh Shah
Íñigo Goiri
Masatake Iwasaki
Jonathan Eagles
Jakob Homan
Jonathan Hung
Jian He
Jing Zhao
Jitendra Nath Pandey
Jason Lowe
John Zhuge
Junping Du
Karthik Kambatla
Kihwal Lee
Konstantinos Karanasos
Lei Xu
Mingliang Liu
Lokesh Jain
Luke Lu
Mahadev Konar
Matt Foley
Ming Ma
Mukul Kumar Singh
Naganarasimha G R
Nanda kumar
Nigel Daley
Owen O'Malley
Tsuyoshi Ozawa
Patrick Hunt
Raghu Angadi
Ray Chiang
Robert Kanter
Rohith Sharma K S
Sharad Agarwal
Konstantin Shvachko
Sammi Chen
Sandy Ryza
Shashikant Banerjee
Sangjin Lee
Szilard Nemeth
Sanjay Radia
Siddharth Seth
Michael Stack
Steve Loughran
Subru Krishnan
Sunil Govindan
Surendra Singh Lilhore
Suresh Srinivas
Tsz Wo (Nicholas) Sze
Takanobu Asanuma
Daniel Templeton
Thomas Graves
Todd Lipcon
Tom White
Alejandro Abdelnur
Uma Maheswara Rao G
Varun Saxena
Vinayakumar B
Vinod Kumar Vavilapalli
Varun Vasudev
Hemanth Yamijala
Andrew Wang
Wangda Tan
Haohui Mai
Weiwei Yang
Xuan Gong
Xiao Chen
Xiaoyu Yao
Yongjun Zhang
Yi Liu
Yiqun Lin
Yufei Gu
Zhe Zhang
Zhijie Shen
Zheng Shao
Zhankun Tang

Hadoop Committers

Akira Ajisaka
Andrzej Bialecki
Abhishek Modi
Arun C Murthy
Anubhav Dhoot
Anu Engineer
Ajay Kumar
Amareshwari Sriramadasu
Amar Ramesh Kamat
Arpit Agarwal
Arpit Gupta
Arun Suresh
Aaron T. Myers
Aravindan Vijayan
Allen Wittenauer
Ayush Saxena
Benoy Antony
Bharat Viswanadham
Bibin A Chundatt
Bikas Saha
Billie Rinaldi
Robert(Bobby) Evans
Boris Shkolnik
Botong Huang
Brahma Reddy Battula
Brandon Li
Benjamin Teke
Sean Busbey
Chris Douglas
Chen Liang
Colin Patrick McCabe
Chris Nauroth
Konstantin Boudnik
Chandni Singh
Chris Trezzo
Carlo Curino
Doug Cutting
Daryn Sharp
Devaraj Das
Devaraj K
Dhruba Borthakur
Dinesh Chitlangia
David Mollitor
Kai Zheng
Eric Badger
Márton Elek
Eli Collins
Eric Payne
Enis Soztutar
Eric Yang
Hui Fei
Gabor Bota
Gautham Banasandra
Gera Shegalov
Giovanni Matteo Fumarola
Giridharan Kesavan
Li Lu
Hemanth Boyina
Hitesh Shah
Haibo Chen
Hairong Kuang
Hanisha Koneru
Harsh J
He Xiaoqiao
Íñigo Goiri
Ivan Mitic
Masatake Iwasaki
Jim Brennan
Jonathan Eagles
Jakob Homan
Jonathan Hung
Jian He
Jing Zhao
Jitendra Nath Pandey
Jason Lowe
Johan Oskarsson
Junping Du
John Zhuge
Karthik Kambatla
Kihwal Lee
Konstantinos Karanasos
Koji Noguchi
Kan Zhang
Lei Xu
Li Cheng
Mingliang Liu
Xun Liu
Lokesh Jain
Lohit Vijayarenu
Luke Lu
Larry McCay

Sean Mackrory
Mahadev Konar
Manoj Govindassamy
Matei Zaharia
Matthew Foley
Mayank Bansal
Dmytro Molkov
Mukund Madhugiri
Ming Ma
Mukul Kumar Singh
Naganarasimha G R
Nanda kumar
Nigel Daley
Nathan Roberts
Owen O'Malley
Tsuyoshi Ozawa
Prabhu Joseph
Andras Gyori
Rakesh Radhakrishnan
Raghu Angadi
Ramya Sunil
Ravi Prakash
Ray Chiang
Robert Kanter
Rohith Sharma K S
Roman Shaposhnik
Sammi Chen
Scott Chun-Yang Chen
Sharad Agarwal
Konstantin Shvachko
Sandy Ryza
Shashikant Banerjee
Siyao Meng
Sangjin Lee
Sanjay Radia
Sreekanth Ramakrishnan
Sidharta Seethana
Szilard Nemeth
Siddharth Seth
Steve Loughran
Subru Krishnan
Chao Sun
Sunil Govindan
Surendra Singh Lilhore
Suresh Srinivas
Siddharth Wagle
Miklos Szegedi
Tsz Wo (Nicholas) Sze
Tanping Wang
Tao Yang
Takanobu Asanuma
Christophe Taton
Daniel Templeton
Thomas Graves
Todd Lipcon
Tao Li
Tom White
Alejandro Abdelnur
Uma Maheswara Rao G
Varun Saxena
Vinayakumar B
Vinod Kumar Vavilapalli
Vivek Ratnavel Subramanian
Vrushali Channapattan
Varun Vasudev
Walter Su
Andrew Wang
Wangda Tan
Wei-Chiu Chuang
Wei Yan
Haohui Mai
Wilfred Spiegelenburg
Weiwei Yang
Xuan Gong
Xiao Chen
Erik Krogen
Xiaoyu Yao
Yufei Gu
Yongjun Zhang
Yi Liu
Yiqun Lin
Zengqiang Xu
Zac Zhou
Qi Zhu
Zhe Zhang
Zhijie Shen
Zheng Shao
Zhankun Tang
Zhihai Xu

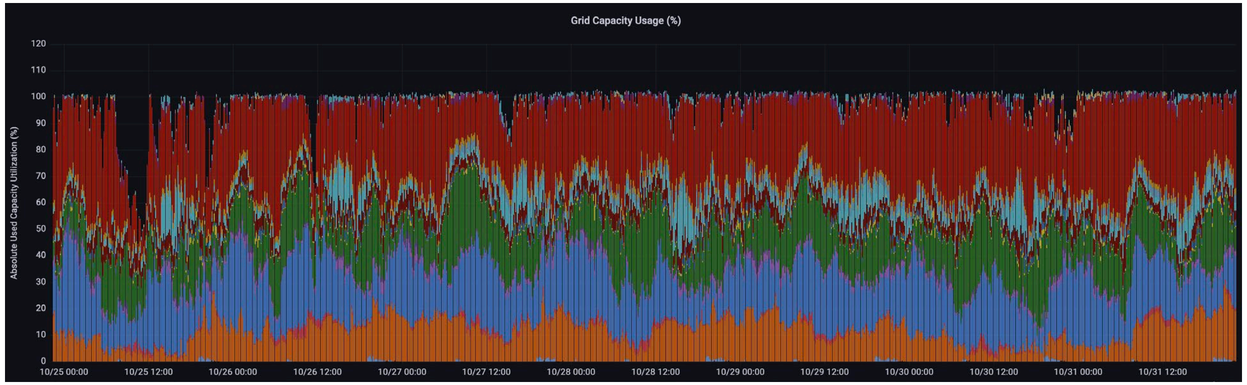
YARN@in: Container Utilization (two weeks ago)

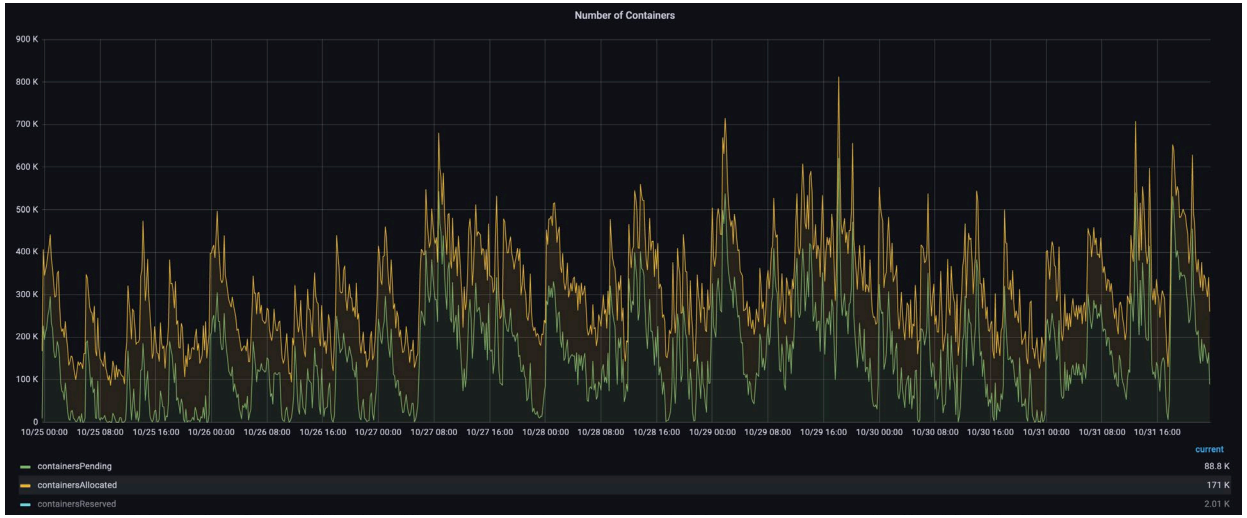
Cluster statistics

- 80-90% container utilization
- Average 20 containers per node (peak ~70)
- 31K nodes in clusters of 7-10.5K nodes
  - Recent, annual growth of 38% GB/Hr
  - Peak 1.1k containers allocated per second

Workload statistics

- Up to 1.1k container allocations per second
- Majority rsrc usage: Spark
- Majority of containers: MapReduce





Screenshots courtesy of: Varun Saxena (LinkedIn)