CS 795: Distributed Systems & Cloud Computing
Fall 2018

Lec 0: Course introduction

Yue Cheng

A brief intro

- Yue Cheng (http://cs.gmu.edu/~yuecheng)

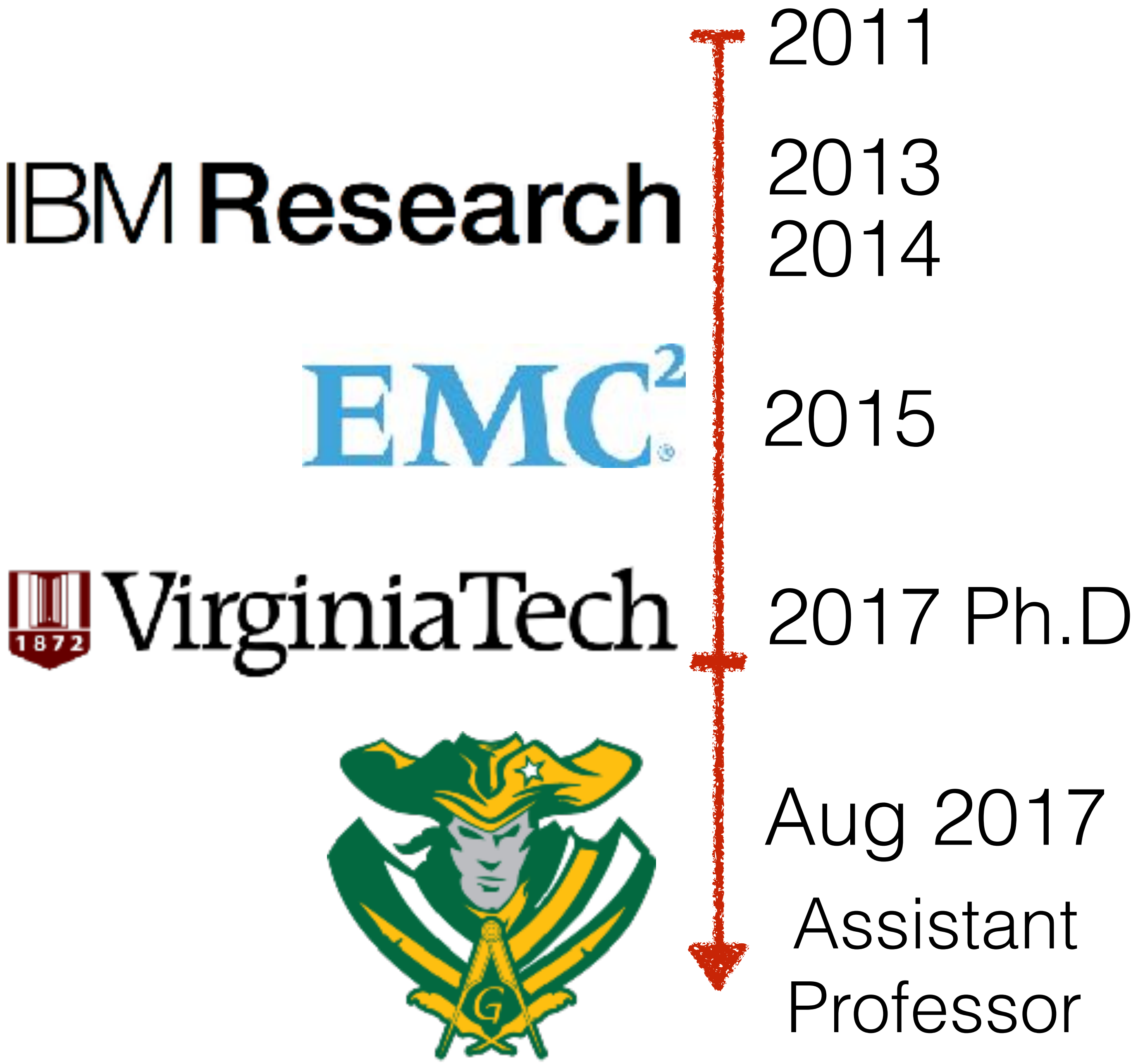
VirginiaTech 2017 Ph.D

Aug 2017

Assistant Professor

A brief intro

- Yue Cheng (http://cs.gmu.edu/~yuecheng)



2011

2013
2014

2015

2017 Ph.D

Aug 2017
Assistant
Professor

Research areas

- Computer systems
- Workload characterization
- Performance analysis

Research interests

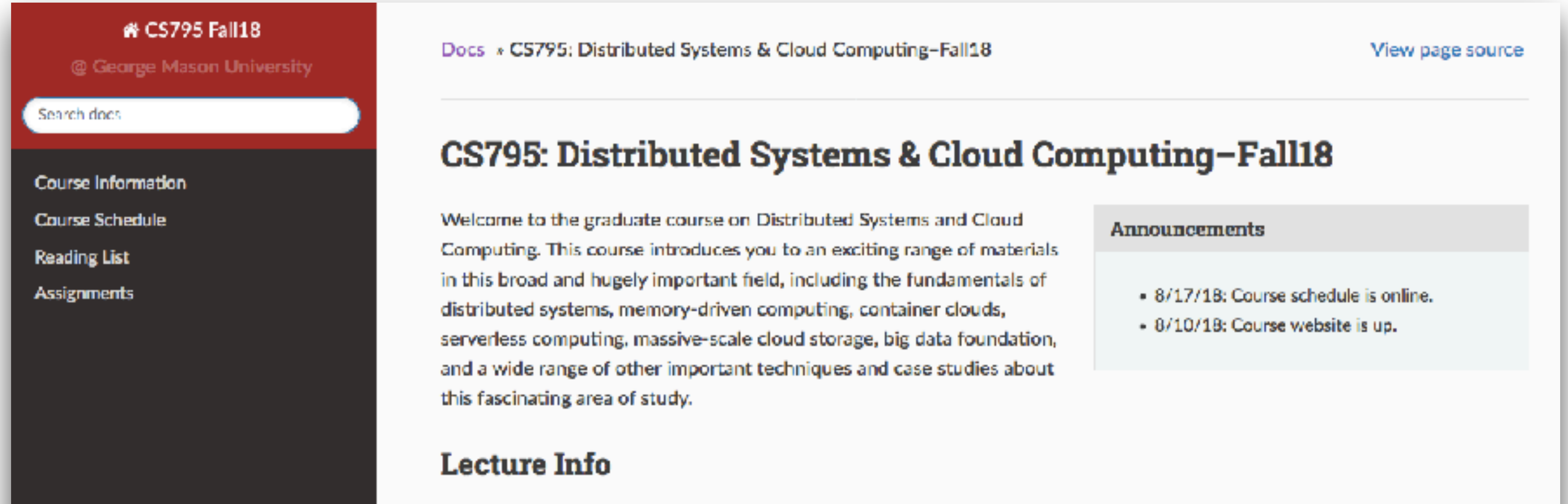
- Distributed & storage systems
- Serverless & cloud computing
- High performance computing (HPC)
- Internet of Things (IoT) & edge computing

Course info

- Meeting time: Wed 4:30 — 7:10pm
- Location: Peterson Hall 1113
- Office hours: Thur 2 — 4pm or by appointment, Engineering 5324

Course info

- Course website:
  - https://cs.gmu.edu/~yuecheng/teaching/cs795_fall18/index.html
  - Updated regularly throughout this semester
  - We will be using Piazza for communication and discussion



CS795: Distributed Systems & Cloud Computing–Fall18

Welcome to the graduate course on Distributed Systems and Cloud Computing. This course introduces you to an exciting range of materials in this broad and hugely important field, including the fundamentals of distributed systems, memory-driven computing, container clouds, serverless computing, massive-scale cloud storage, big data foundation, and a wide range of other important techniques and case studies about this fascinating area of study.

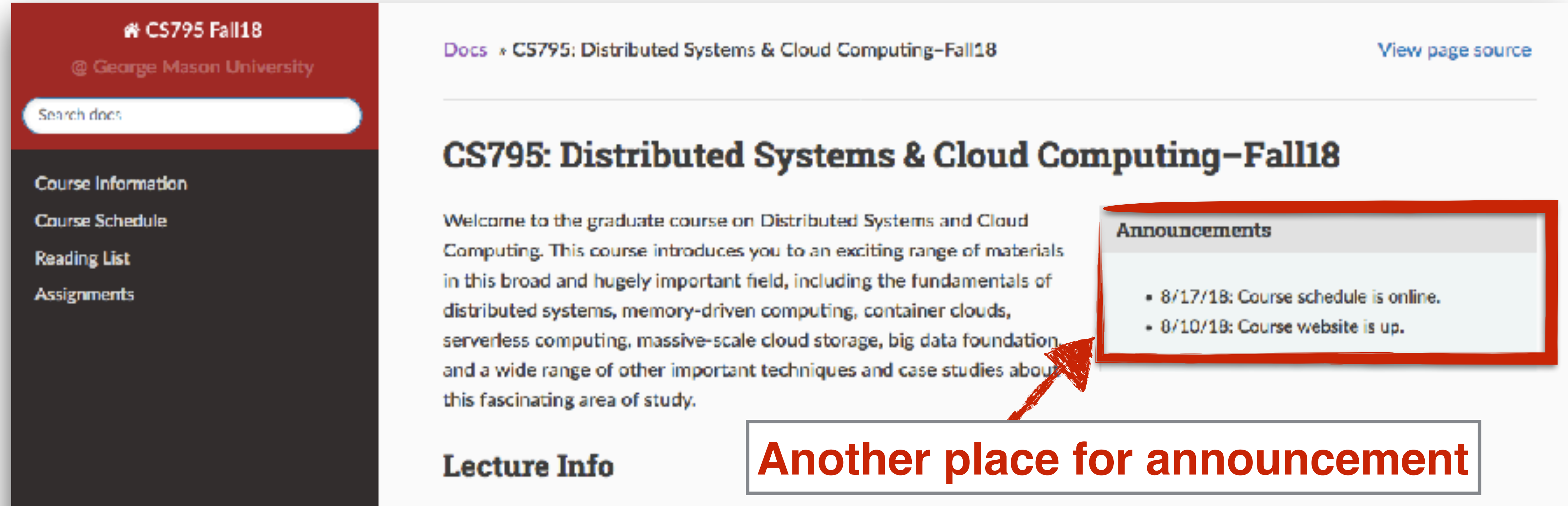
Announcements

- 8/17/18: Course schedule is online.
- 8/10/18: Course website is up.

Lecture Info

Course info

- Course website:
  - https://cs.gmu.edu/~yuecheng/teaching/cs795_fall18/index.html
  - Updated regularly throughout this semester
  - We will be using Piazza for communication and discussion



CS795: Distributed Systems & Cloud Computing–Fall18

Welcome to the graduate course on Distributed Systems and Cloud Computing. This course introduces you to an exciting range of materials in this broad and hugely important field, including the fundamentals of distributed systems, memory-driven computing, container clouds, serverless computing, massive-scale cloud storage, big data foundation, and a wide range of other important techniques and case studies about this fascinating area of study.

Announcements

- 8/17/18: Course schedule is online.
- 8/10/18: Course website is up.

Lecture Info

Another place for announcement

Prerequisites

- NO explicit prerequisite courses
- What I assume you already know:
  - Understanding of the undergrad-level operating systems concepts and principles
  - Programming skills in at least one of: **Python**, C/C++, Golang, Java
- Some familiarity with *NIX systems

Reading materials

- Primary reading is the assigned research papers
  - Look at the reading list on the class website
  - What would you like to present? What would you like to scribe? (later slides)

- NO textbook required
- One excellent source of OS-level knowledge & background (highly recommended)
  - Operating Systems: Three Easy Pieces, by Remzi H. Arpaci-Dusseau and Andrea C. Arpaci-Dusseau (v 0.92)
  - URL: http://pages.cs.wisc.edu/~remzi/OSTEP/

Course format

- Lectures + discussions
    - At the beginning of the semester: I give a few lectures covering fundamentals that you need to know
    - Starting from week 3 (tentatively): I give a short lecture (30+min) covering Why Today's Papers + 2 paper discussions (50-60min each) led by you
- Four student presentations
    - Two for assigned research papers (may +/-1 depending on class size and your motivation)
    - Two for research projects (1 proposal + 1 final)

Discussions

- Everyone reads the assigned papers (x2) before class
- Submit a brief evaluation form before each class
  - Proves that you’ve read the papers
  - Enable you to contribute to the discussion
- Each paper assigned to a scriber
  - Scriber writes reaction posts on Piazza after class

Paper evaluation form

- Submissions: Will send out a Google form for the two papers that you need to fill in (the Google form will close **10min** before the class)
- No late submissions will be accepted
- Instead, you will have **three wildcards**
  - Three dates on which you can skip evaluation forms without penalty
  - Need not be announced beforehand
- Contact instructor for exceptions in severe circumstances only

Paper evaluation form

- What **problem** does the paper attack? How does it relate to and improve upon previous work in its domain?
- What is the solution's **main idea**?
- Does the paper (or do you) identify any fundamental/hard **tradeoffs**?
- Write down **question/s** that you plan to bring up in the discussion

Your presentation

- Go through the reading list looking for papers from USENIX conferences
    - USENIX publishes all talks in form of **slides**, **audio**, and **video**
    - See how paper authors present
- Present research as if it were your own
    - Give background if necessary
- Evaluate research from your perspective
    - Add insights, criticism, etc.
    - In retrospect: Why did it succeed or fail?
- Be prepared to be interrupted
    - People ask questions in between

Scribe: Reaction post

- A 3-paragraph post for each paper on Piazza after each class
  - Due 11pm the day of the class
- Reaction post format
  - Summary (~3 lines)
    - What were the main doubts/questions raised in discussion?
  - Expanding (~3 lines)
    - What non-trivial technical aspects were missing in the presentation (but covered in paper)?
  - Brainstorming (~4 lines)
    - Any open interesting questions (e.g., unsolved challenges)? Extensions?
    - What possible applications could benefit from the paper?
    - Any flaws possibly (intentionally) hidden in the paper?

USENIX FAST'14 scribe report:
https://www.usenix.org/system/files/login/articles/1406_fast14_reports.pdf

Conference Reports

In this issue:

FAST '14: 12th USENIX Conference on File and Storage Technologies

February 17–20, 2014, San Jose, CA

Opening Remarks

Summarized by Rik Farrow

Bianca Schroeder (University of Toronto) opened this year's USENIX Conference on File and Storage Technologies (FAST '14) by telling us that we represented a record number of attendees for FAST. Additionally, 133 papers were submitted, with 24 accepted. That's also near the record number of submissions, 137, which was set in 2012. The acceptance rate was 18%, with 12 academic, three industry, and nine collaborations in the author lists. The 28 PC members together completed 500 reviews, and most visited Toronto in December for the PC meeting.

utilization. Existing garbage collectors, however, are expensive and scale poorly. They wait until a lot of free space is available (to amortize cleaning costs), which can require up to 5x over-utilization of memory. When the garbage collector does run, it can consume up to three seconds, which is slower than just resetting the system and rebuilding the RAM store from the backup log on disk.

The authors develop a new cleaning approach that avoids these problems. Because pointers in a file system are well-controlled, centrally stored, and have no circularities, it is possible to clean and copy incrementally (which would not work for a more general-purpose garbage-collection system). In the authors' approach, the cleaner continuously finds and cleans some segments with significant free space, reducing cleaning cost and improving utilization. Further, the authors distinguish between the main log, kept in expensive DRAM with high bandwidth (targeted at 90% utilization), and the backup log, stored on disk where capacity is cheap but bandwidth is lower (targeted at 50% utilization). They use a two-level approach in which one cleaner ("compaction") incrementally cleans one segment at a time in memory, while a second one ("combined cleaning") less frequently cleans across segments in both memory and disk. Both cleaners run in parallel to normal operations, with limited synchronization points to avoid interference with new writes.

Class participation

- Your participation is very important
- Usually as an indicator of how well you’ve got prepared
  - In-class discussions
  - After-class non-trivial response to reaction posts
- Don’t be shy

- Lack of participation may lead to a loss of as much as a letter grade

Homework assignments

- 2 coding homework assignments
  - Pick your partner: a team of at most 2 students
  - #1: Build a consistent cloud object store atop weakly consistent S3: Due on Sep 14
  - #2: Optimize your object store service: Due on Oct 5
- You will learn how to leverage off-the-shelf open-source frameworks and public cloud services to build useful services
  - Assemble microservices together
  - Enable useful tool
- To get you warmed up for the final research projects

Homework assignments

- 2 coding homework assignments
  - Pick your partner: a team of at most 2 students
  - #1: Build a consistent cloud object store atop weakly consistent S3: Due on Sep 14
  - #2: Optimize your object store service: Due on Oct 5
- You will learn how to leve
  - Assemble microservices together
- To get you warmed up for the final research projects

Please sign-up for aws educate

https://aws.amazon.com/education/awseducate/

Research-oriented term projects

- Investigate new ideas and solutions in a class research project
    - Define the problem
    - Execute the research
    - Write up and present your research
- Ideally, best projects will have the potential to become conference papers :)

Research projects: Steps

- I will distribute a list of projects (Week 5/6)
  - You can either choose one or come up with your own
- Pick your partner: a team of at most 2 students
- Milestones (tentative)
  - Project proposal presentation on Oct 17, proposal report due Oct 26
  - Project checkpoint report due Nov 16
  - Final presentation on Dec 5, final project report & src due Dec 14

Grading (tentative)

- Class participation necessary!
  - To get at least **5%**
    - 5+ Non-trivial response to reaction posts counts (Piazza tracks post stats)
    - In-class discussions
- HW & Proj
  - **START EARLY!!!**

Homework assignments	10+10%	Two
Paper evaluations & scribe	10%	~14 evaluation forms + 2 scribe reports
Paper presentations	15%	Two
Class participation	5%	Get involved
Research projects	50%	Substantial!

Why not introduce yourself?

Questions?

Warm-up & basics

What is Systems Research about?

- Manage resources
  - Memory, CPU, storage, network
  - Data (file systems, database systems, key-value stores)

- Provide abstractions to applications
  - Files
  - Processes, threads
  - Virtual machines (VMs), containers
  - …

What is Distributed Systems & Cloud Computing Research about?

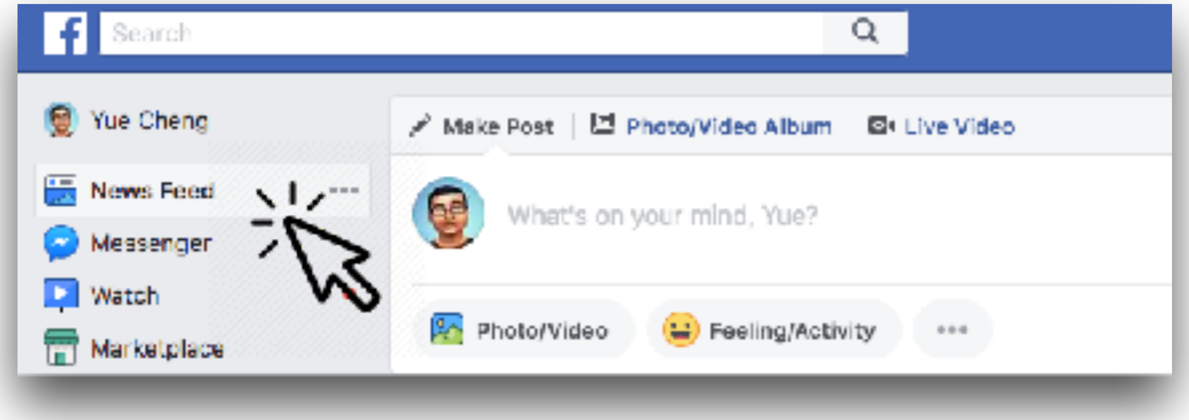
- So we are using a whole bunch of **Cloud-boosted** services everyday…

Dropbox example

- As a startup, stored all data on AWS initially
- Became so popular so quickly
  - Latest number of users: 500 Million
  - Overall amount of data stored: 500 Petabytes
- Seriously considered to move data out of the public cloud
- Cloud lock-in
  - Egress costs
- Now still part of its data services sitting atop AWS

Facebook example

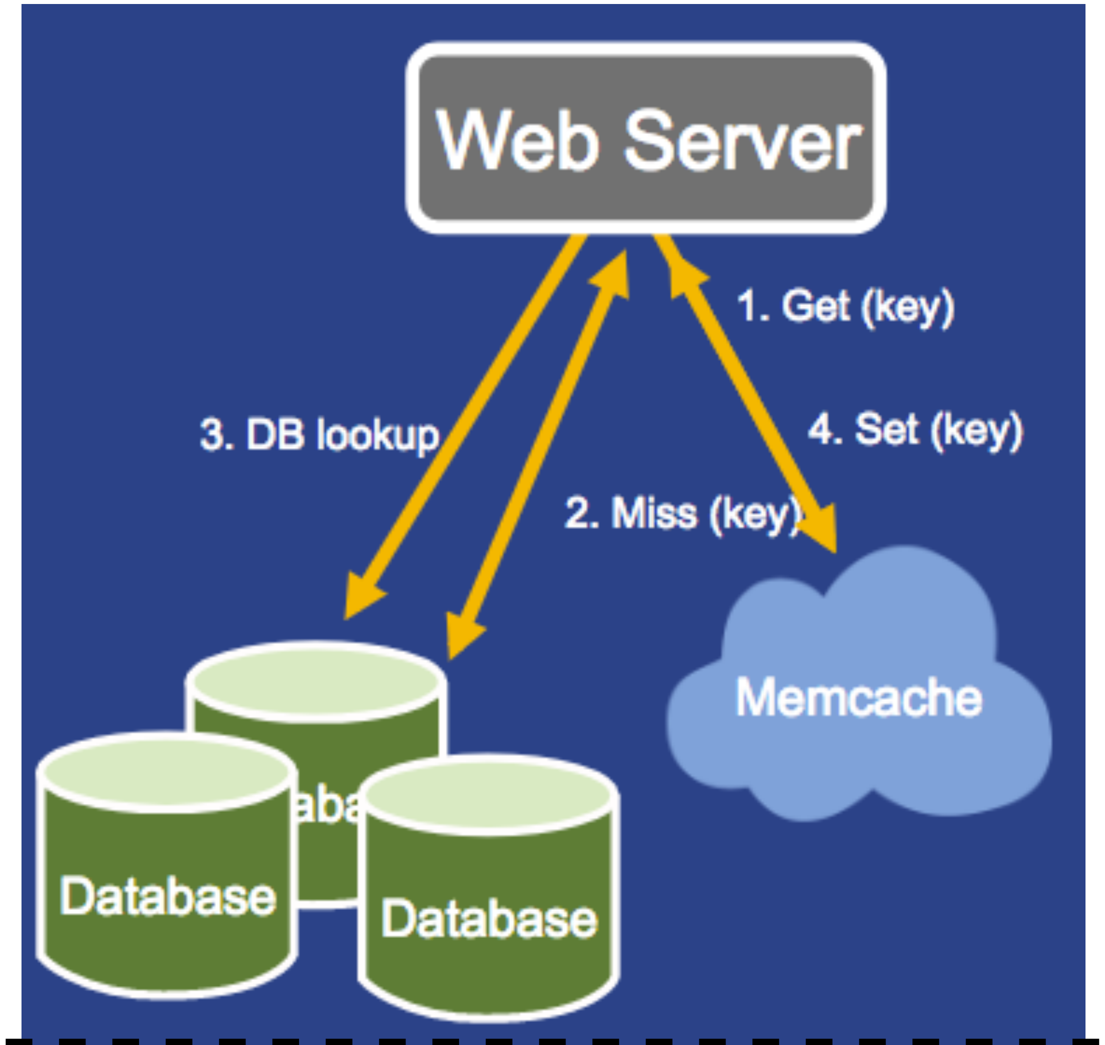
1. User clicks on a link



2. Browser sends requests to FB's front-end web servers

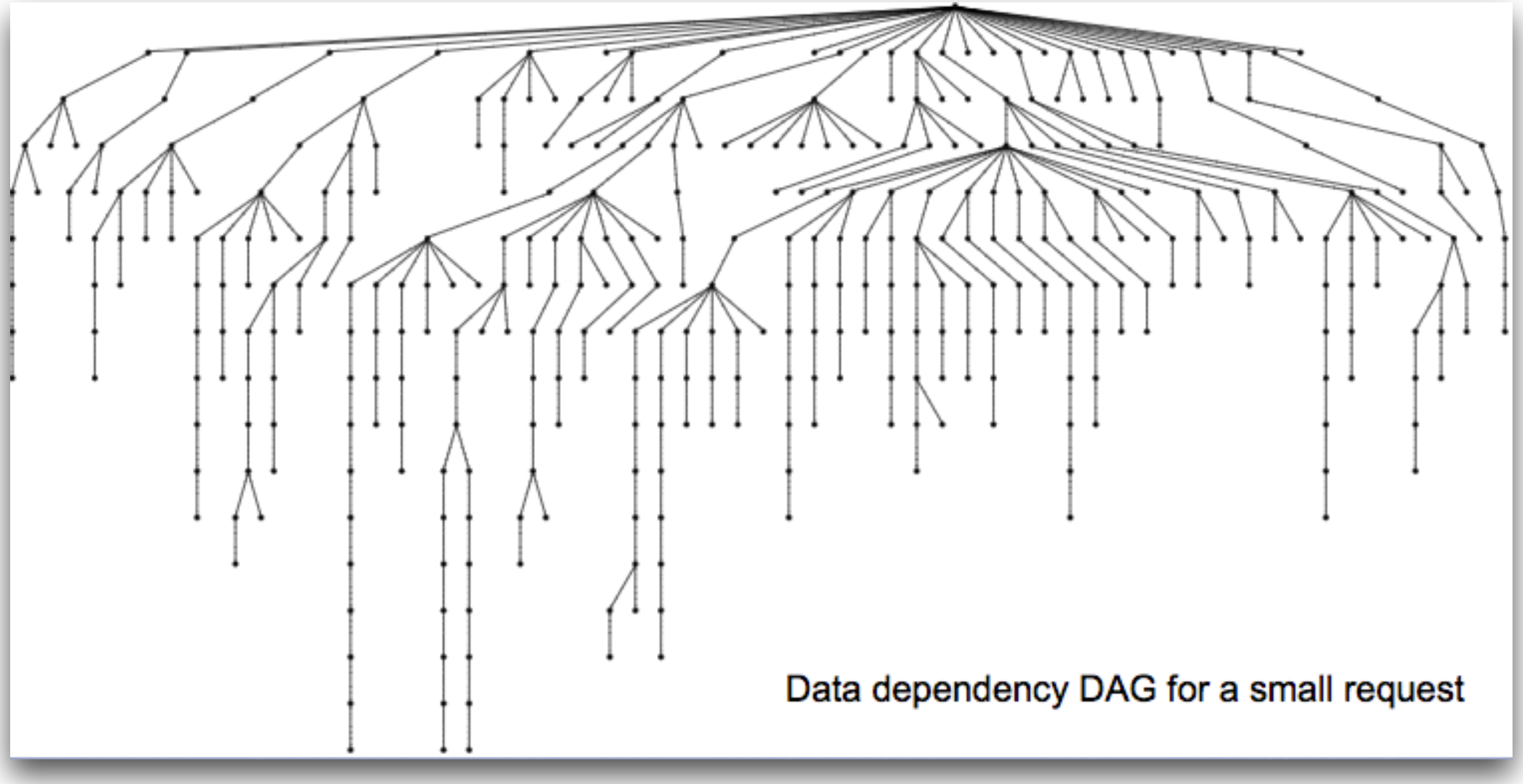
3. Web servers convert requests to:

- DB queries
- Memcache queries



Facebook example

- High fanout and multiple rounds of data fetching*

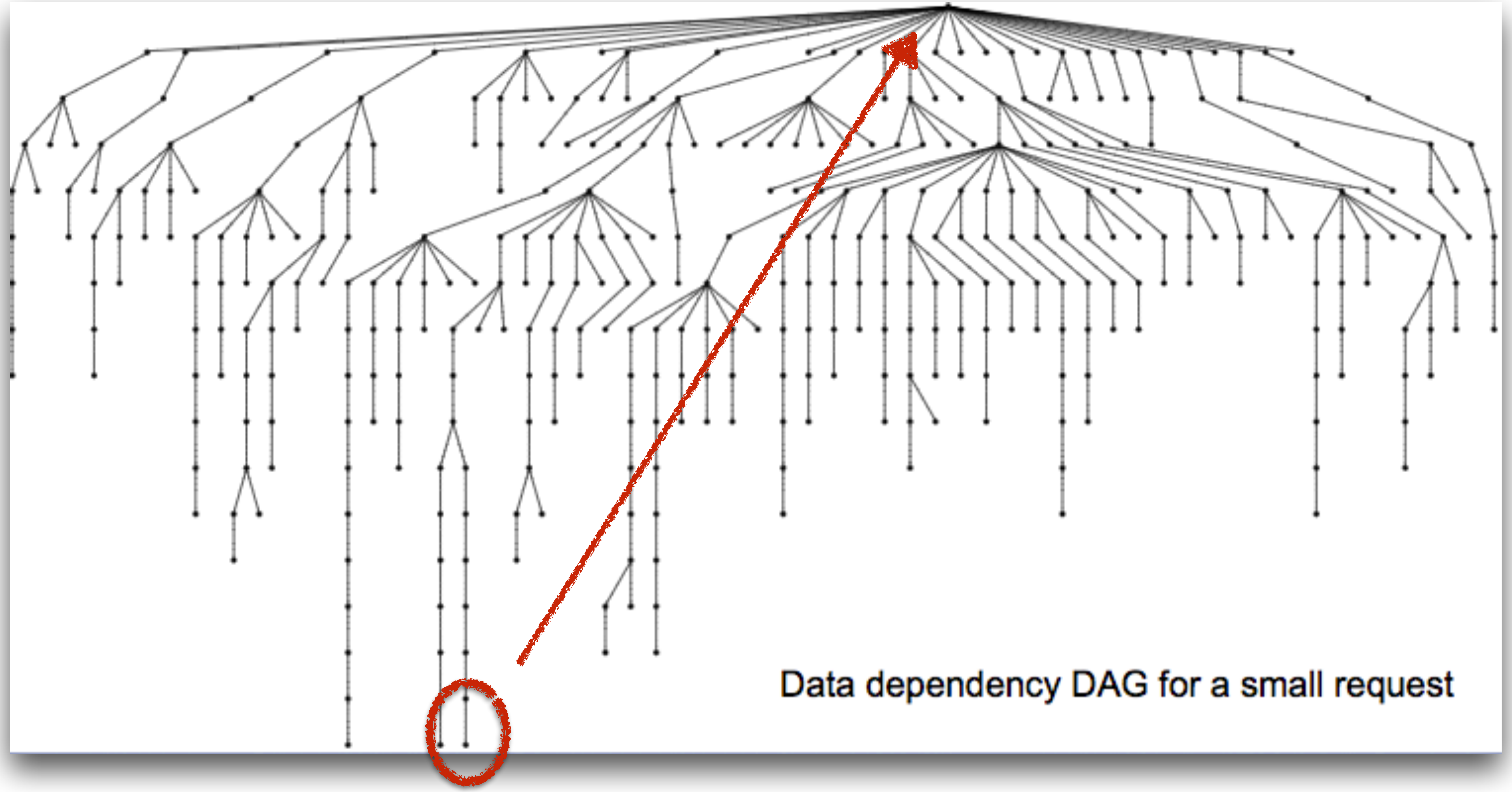


Data dependency DAG for a small request

*: Scaling Memcache at Facebook [NSDI'13]

Facebook example

- High fanout and multiple rounds of data fetching*



Data dependency DAG for a small request

*: Scaling Memcache at Facebook [NSDI'13]

Facebook example

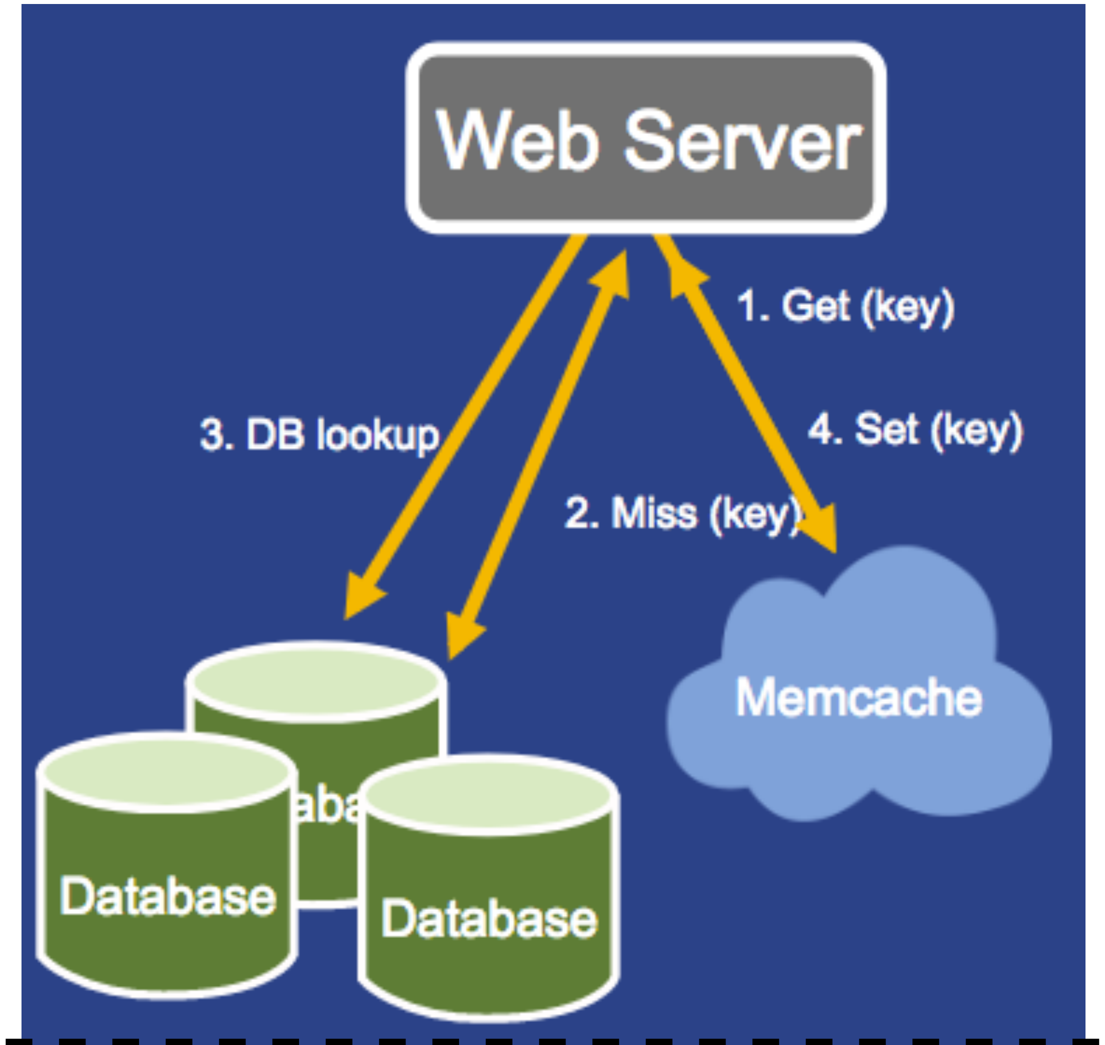
1. Click on a link

2. Browser sends requests to FB's front-end web servers

Our focus: To learn how backend *magic* manages resources

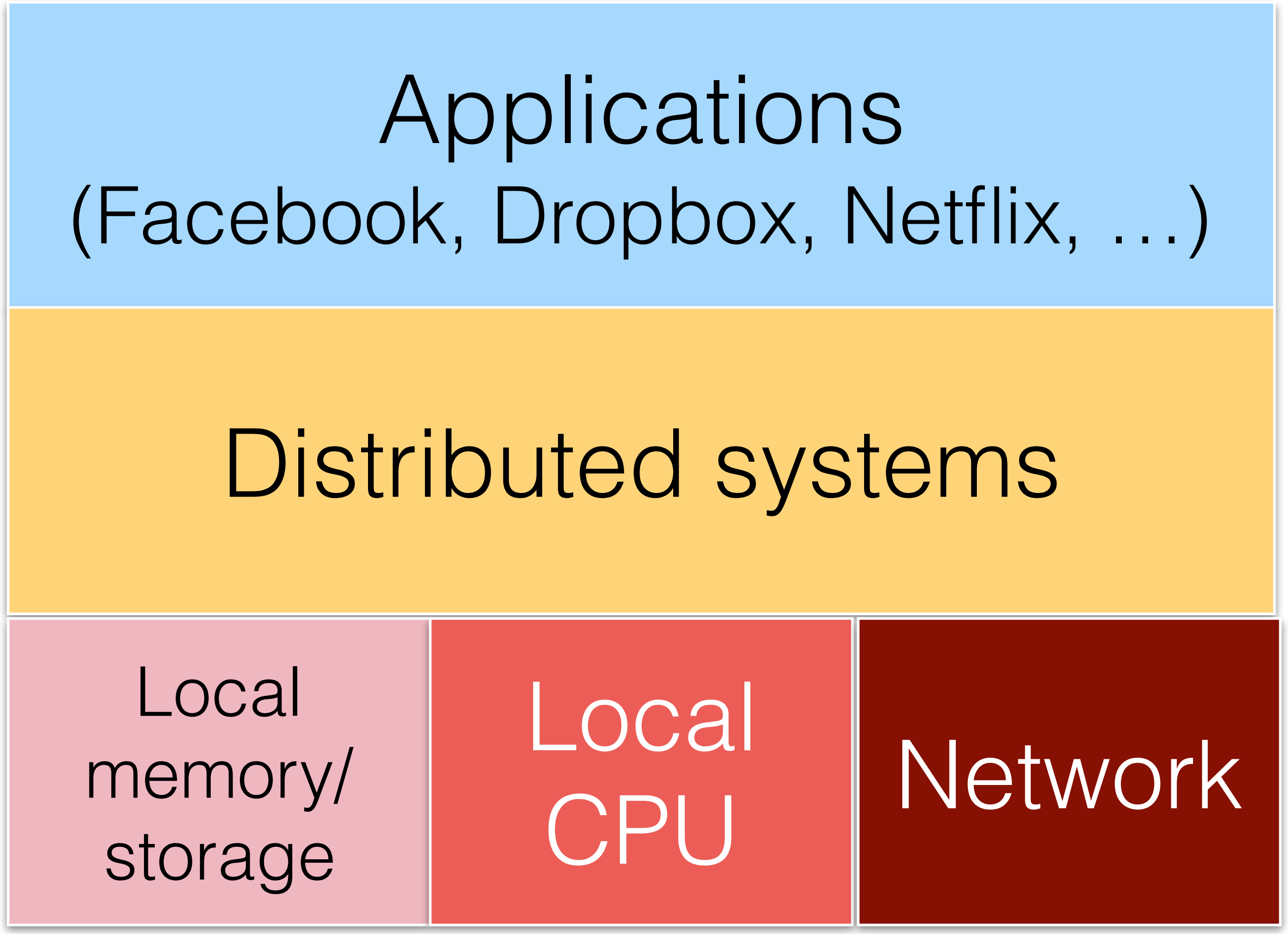
3. Web servers convert requests to:

- DB queries
- Memcache queries



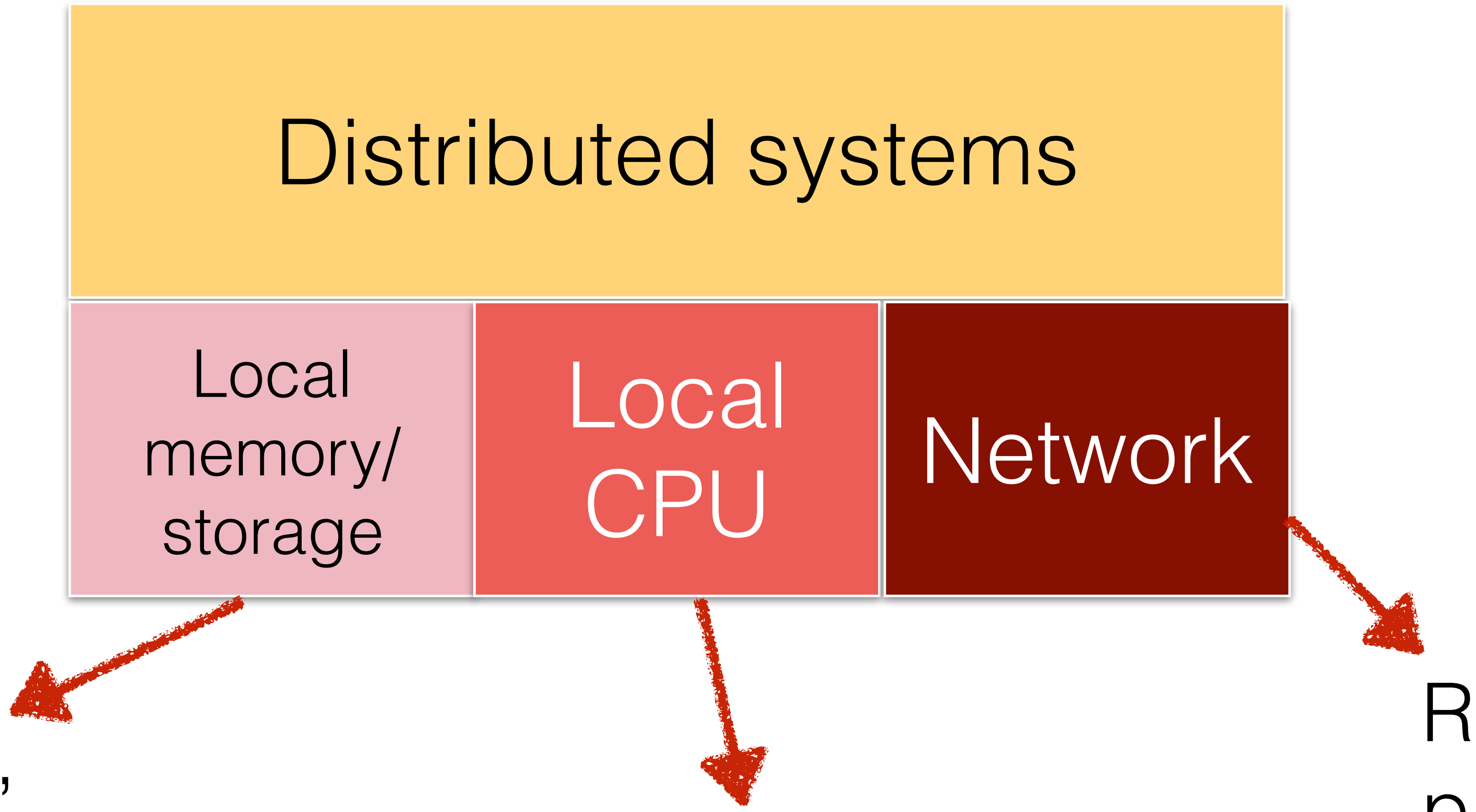
What is Distributed Systems & Cloud Computing Research about?

- Those applications/services are the Abstractions & Interfaces. So what is under the hood in the backend?



What is Distributed Systems & Cloud Computing Research about?

- At single-node level, OS manages all local resources
  - OS is essentially a mini distributed system manager

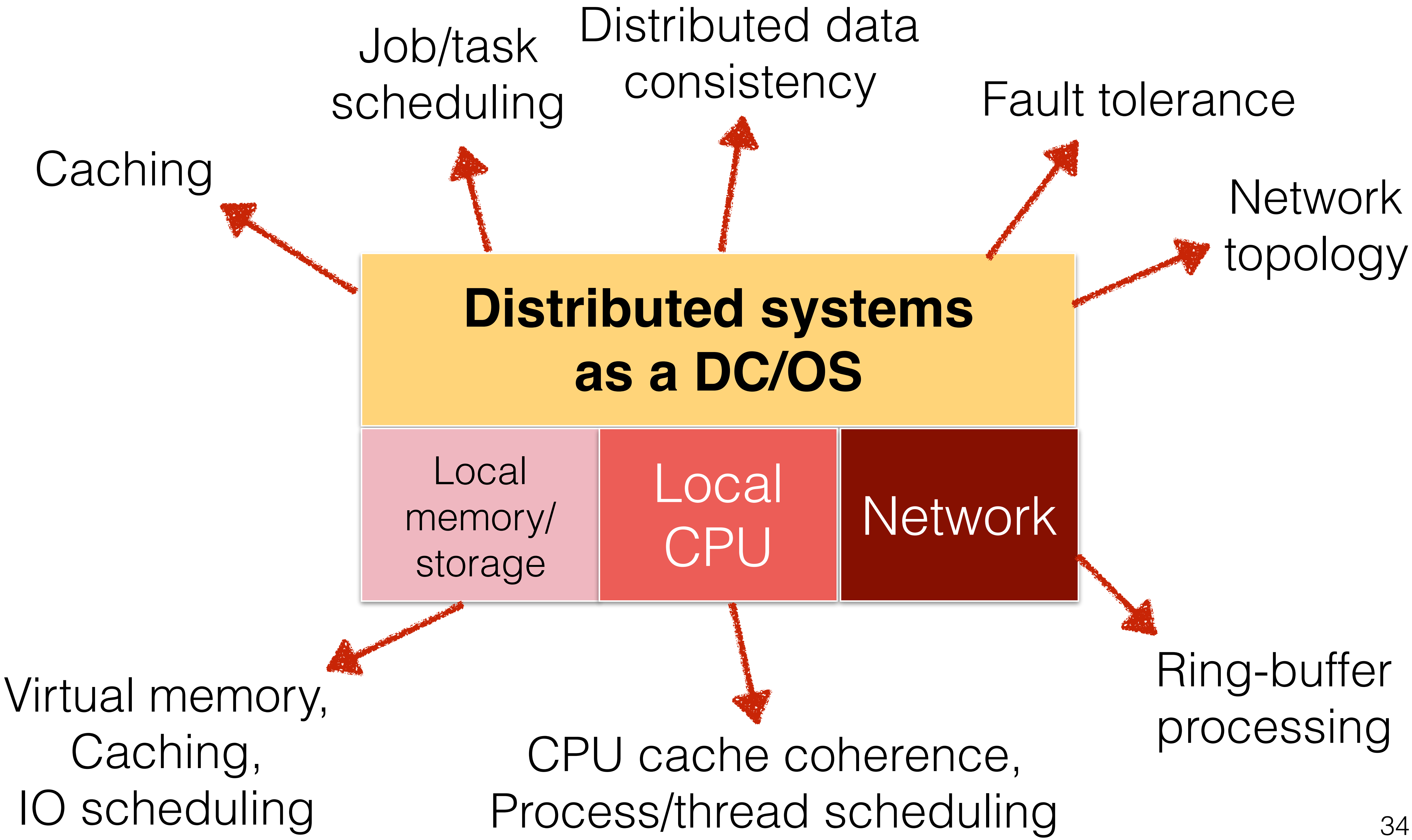


Virtual memory, Caching, IO scheduling

CPU cache coherence, Process/thread scheduling

Ring-buffer processing

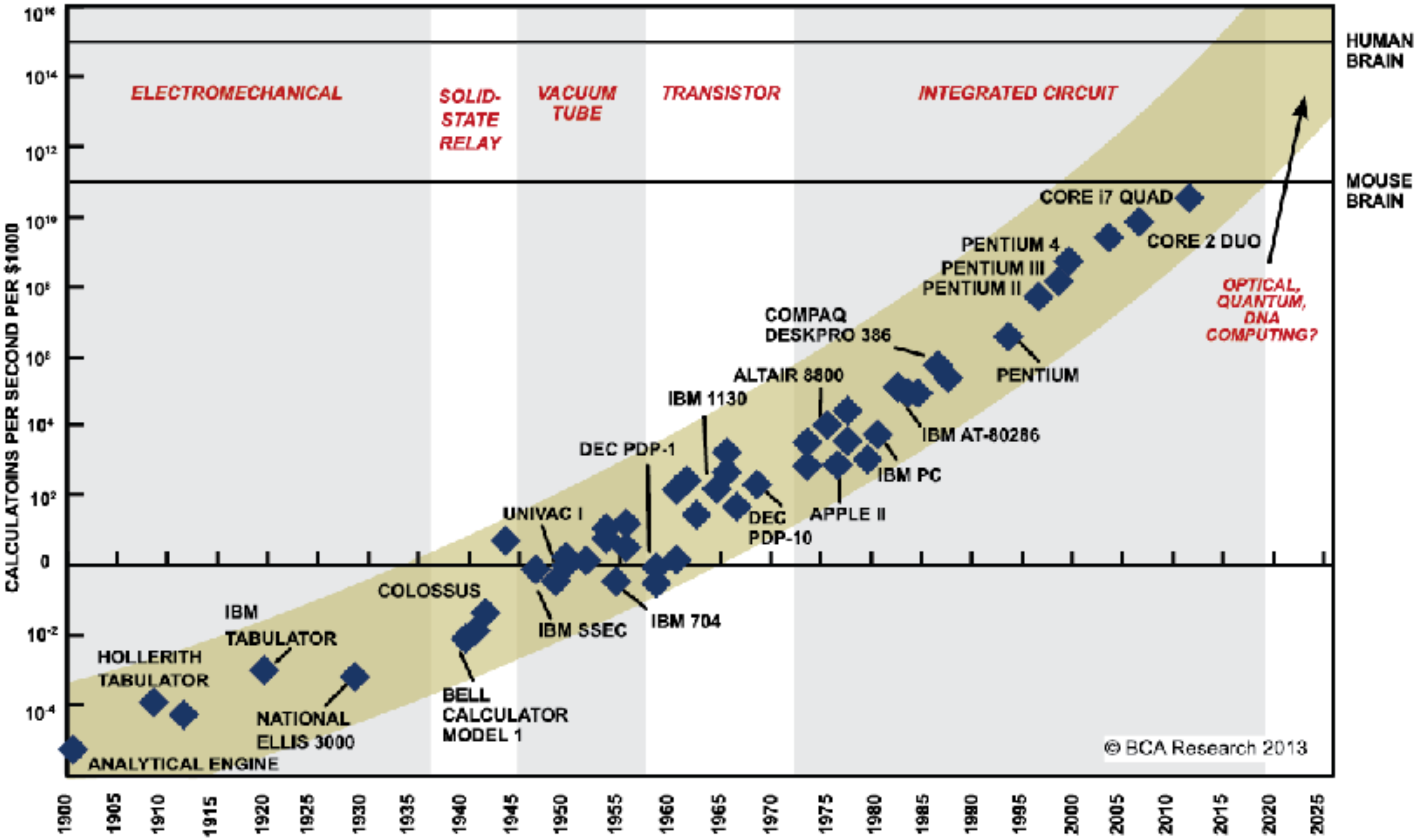
What is Distributed Systems & Cloud Computing Research about?



Caching
Job/task scheduling
Distributed data consistency
Fault tolerance
Network topology
Distributed systems as a DC/OS
Local memory/ storage
Local CPU
Network
Virtual memory, Caching, IO scheduling
CPU cache coherence, Process/thread scheduling
Ring-buffer processing

Exciting times in Distributed Systems & Cloud Computing Research

- Moore’s law ending —> many challenges



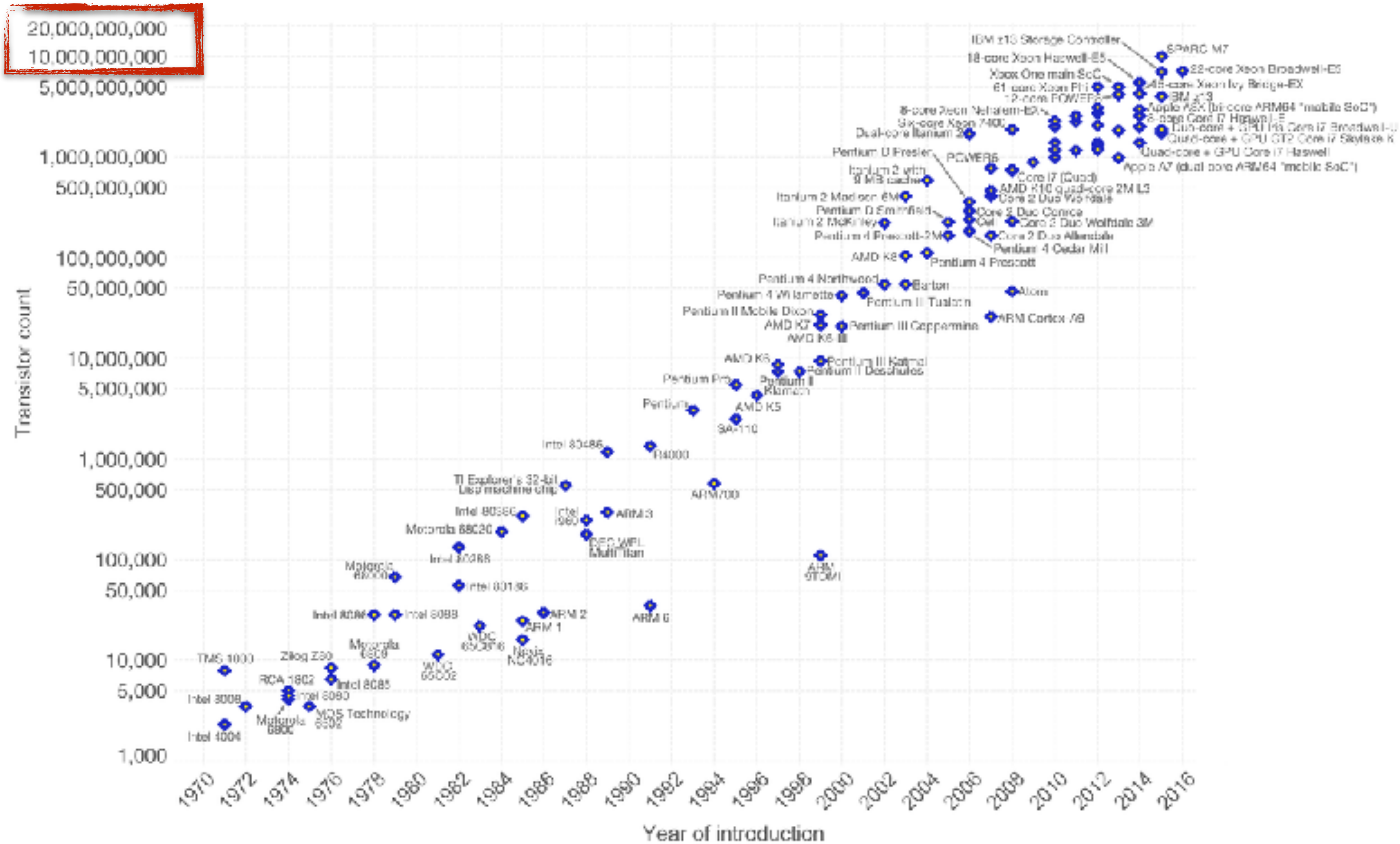
SOURCE: RAY KURZWEIL. "THE SINGULARITY IS NEAR: WHEN HUMANS TRANSCEND BIOLOGY", P.67, *THE VIKING PRESS*, 2006. DATAPOINTS BETWEEN 2000 AND 2012 REPRESENT BCA ESTIMATES.

Exciting times in Distributed Systems & Cloud Computing Research

- Moore's law ending —> many challenges

Moore's Law – The number of transistors on integrated circuit chips (1971-2016) Our World in Data

Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years. This advancement is important as other aspects of technological progress – such as processing speed or the price of electronic products – are strongly linked to Moore's law.

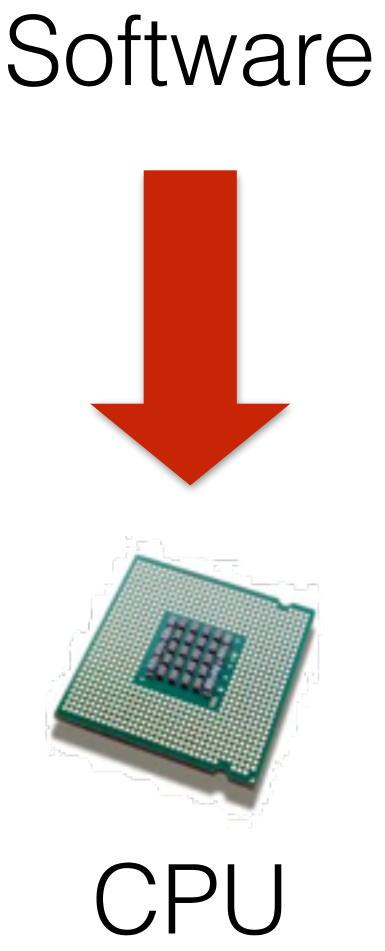


Data source: Wikipedia (https://en.wikipedia.org/wiki/Transistor_count)
The data visualization is available at OurWorldinData.org. There you find more visualizations and research on this topic.

Exciting times in Distributed Systems & Cloud Computing Research

- Many-core machines
  - Amazon’s X1 instances: 128 scores and 2TB DRAM
- Large-scale distributed systems maturing, but many challenges remain
- Specialized hardware
  - [GP]GPUs, FPGAs, etc.
- New memory technologies: Non-Volatile Memories (NVMs)
  - 3D XPoint

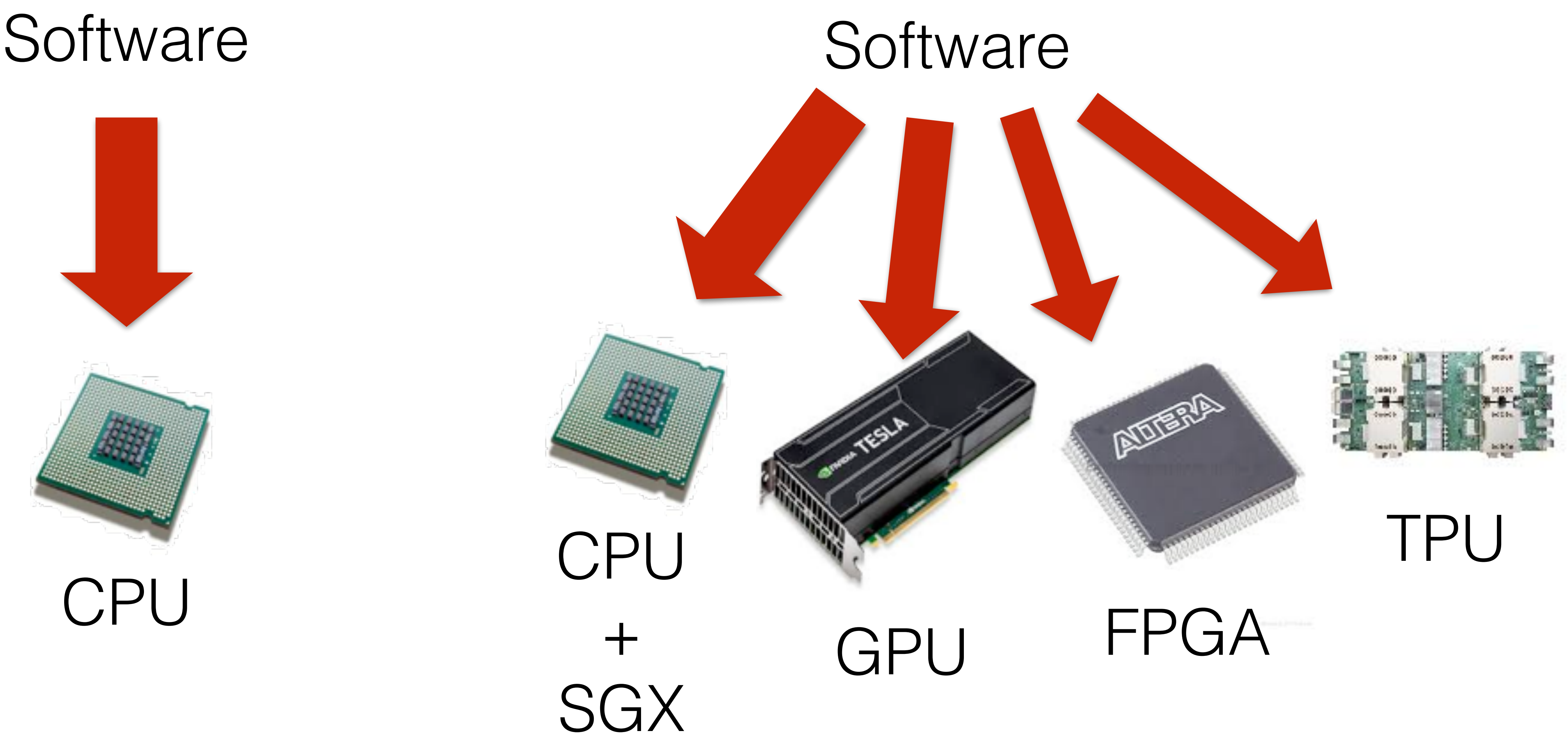
Increased complexity — Computation



Software

CPU

Increased complexity — Computation



Software

CPU

Software

CPU
+
SGX

GPU

FPGA

TPU

Increased complexity — Memory

2015

L1/L2 cache	~1ns
L3 cache	~10ns
Main memory	~100ns / ~80GB/s / ~100GB
NAND SSD	~100us / ~10GB/s / ~1TB
Fast HDD	~10ms / ~100MB/s / ~10TB

Increased complexity — Memory

2015

L1/L2 cache	~1ns
L3 cache	~10ns
Main memory	~100ns / ~80GB/s / ~100GB
NAND SSD	~100us / ~10GB/s / ~1TB
Fast HDD	~10ms / ~100MB/s / ~10TB

2020

L1/L2 cache	~1ns
L3 cache	~10ns
High-bandwidth memory	~10-100ns / ~1TB/s / ~10GB
Main memory	~100ns / ~80GB/s / ~100GB
NVM (3D XPoint)	~1us / ~10GB/s / ~1TB
NAND SSD	~100us / ~10GB/s / ~10TB
Fast HDD	~10ms / ~100MB/s / ~100TB

Increased complexity — More and more choices — **Decision Paralysis!**

Basic tier: A0, A1, A2, A3, A4
Optimized Compute : D1, D2, D3, D4, D11, D12, D13
D1v2, D2v2, D3v2, D11v2,…
Latest CPUs: G1, G2, G3, …
Network Optimized: A8, A9
Compute Intensive: A10, A11,…

Microsoft Azure

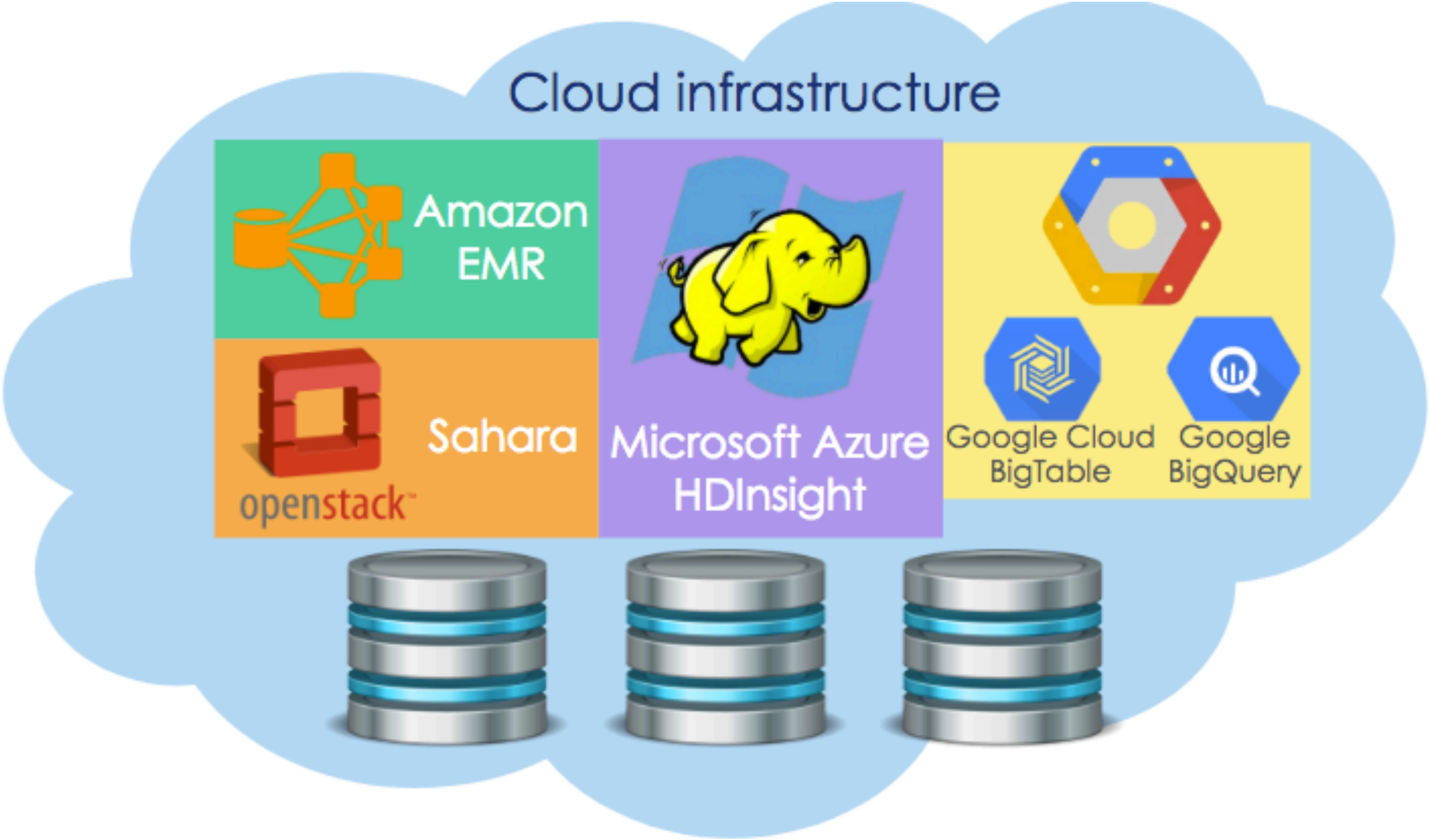
n1-standard-1, ns1-standard-2, ns1-standard-4, ns1-standard-8, ns1-standard-16, ns1highmem-2, ns1-highmem-4, ns1-highmem-8, n1-highcpu-2, n1-highcpu-4, n1-highcpu-8, n1-highcpu-16, n1-highcpu-32, f1-micro, g1-small…

Google Cloud Engine

t2.nano, t2.micro, t2.small
m4.large, m4.xlarge, m4.2xlarge, m4.4xlarge, m3.medium,
c4.large, c4.xlarge, c4.2xlarge,
c3.large, c3.xlarge, c3.4xlarge,
r3.large, r3.xlarge, r3.4xlarge,
i2.2xlarge, i2.4xlarge, d2.xlarge
d2.2xlarge, d2.4xlarge,…

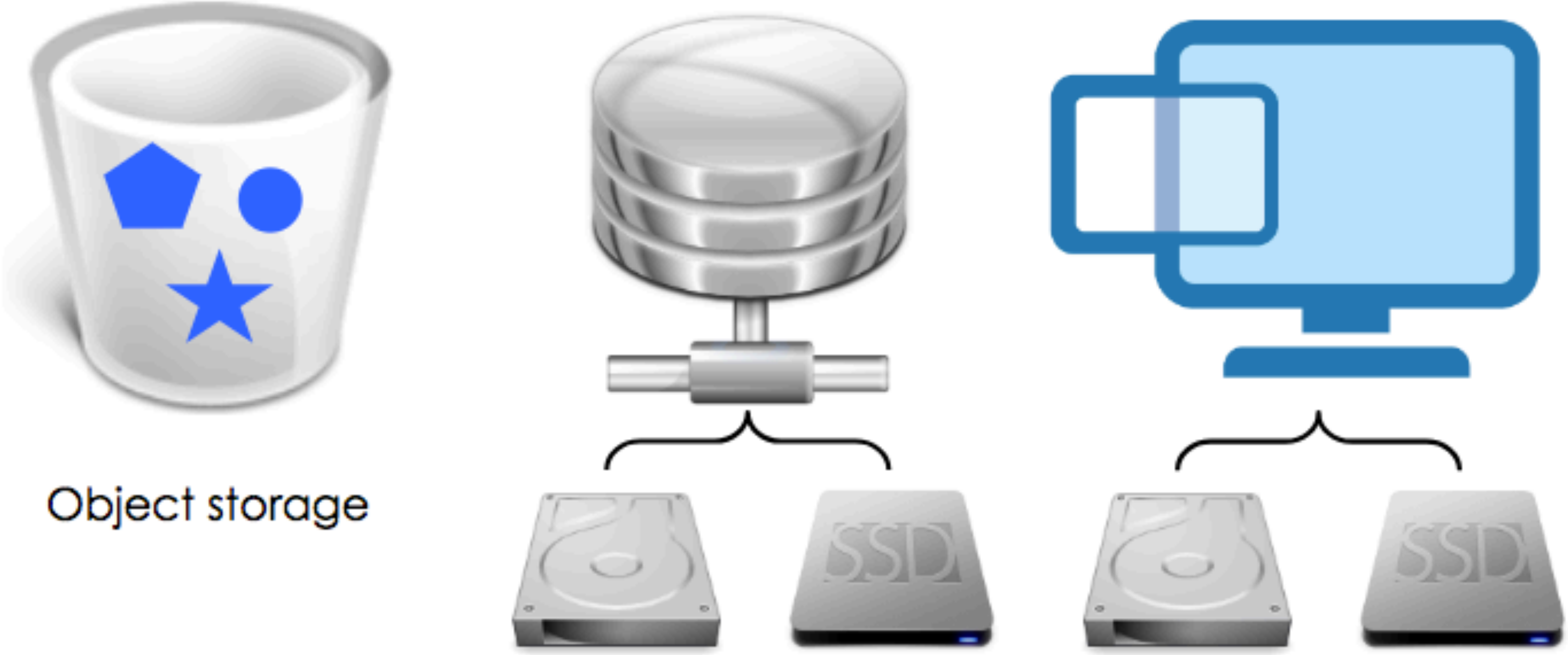
AWS EC2

Case study: Choices of cloud storage for big data analytics



*: CAST: Tiering Storage for Data Analytics in the Cloud [HPDC'15]

Vast variety of cloud storage services



Object storage

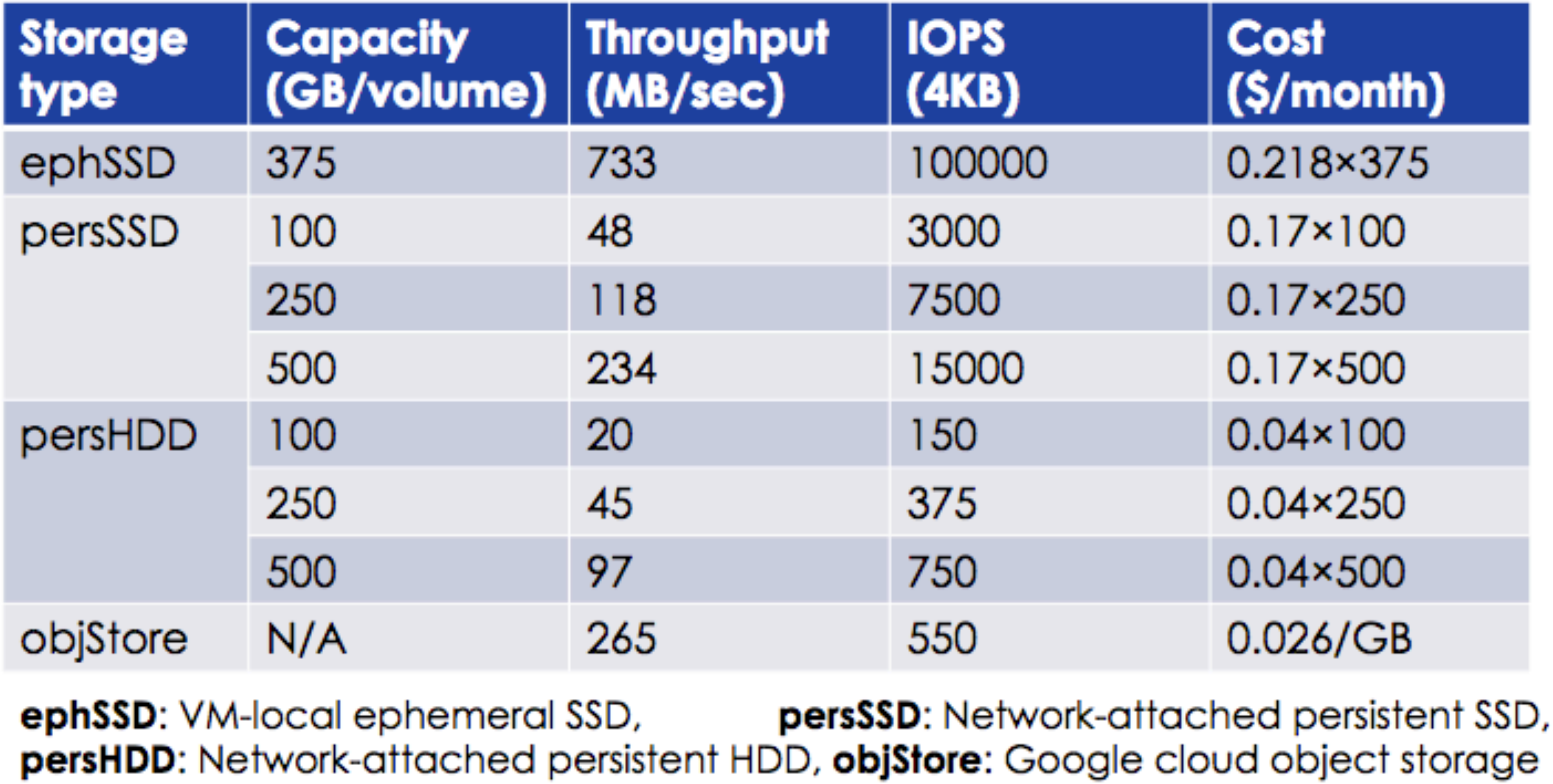
Network-attached block storage

VM-local ephemeral storage

amazon web services

Google Cloud Platform

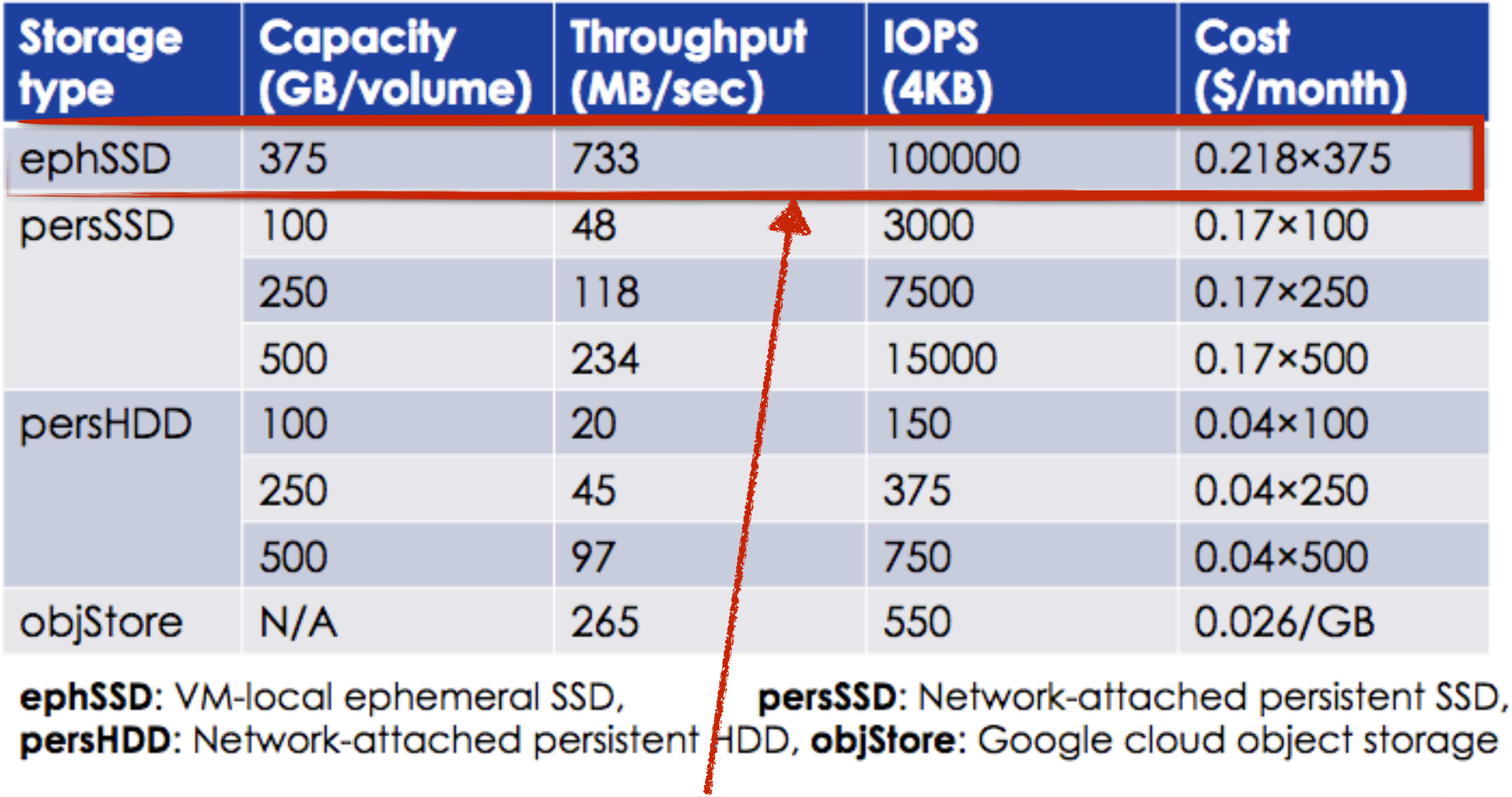
Multi-dimensional heterogeneity in cloud storage services



Storage type	Capacity (GB/volume)	Throughput (MB/sec)	IOPS (4KB)	Cost ($/month)
ephSSD	375	733	100000	0.218×375
persSSD	100	48	3000	0.17×100
	250	118	7500	0.17×250
	500	234	15000	0.17×500
persHDD	100	20	150	0.04×100
	250	45	375	0.04×250
	500	97	750	0.04×500
objStore	N/A	265	550	0.026/GB

ephSSD: VM-local ephemeral SSD, **persSSD**: Network-attached persistent SSD, **persHDD**: Network-attached persistent HDD, **objStore:** Google cloud object storage

Multi-dimensional heterogeneity in cloud storage services

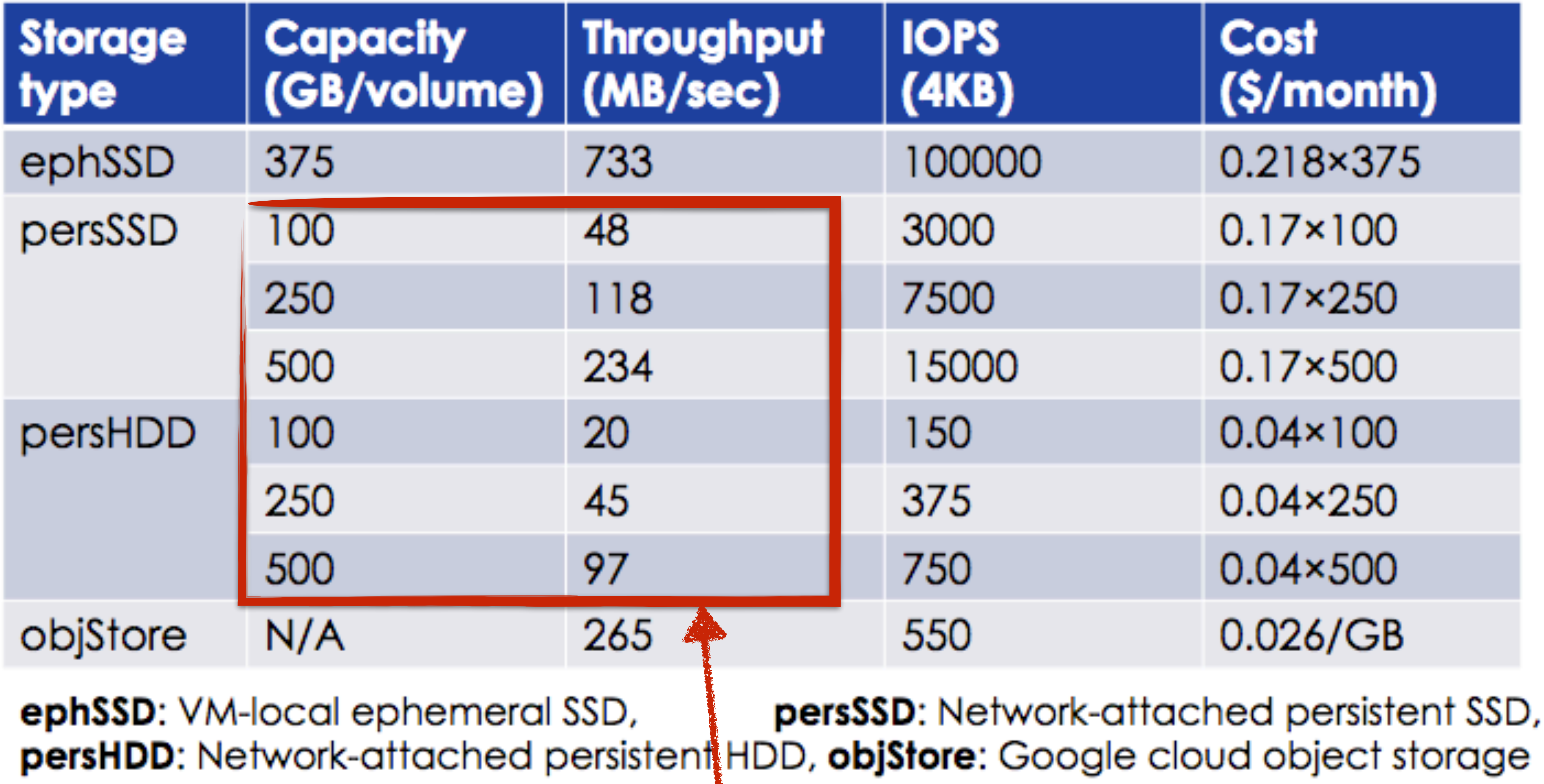


Storage type	Capacity (GB/volume)	Throughput (MB/sec)	IOPS (4KB)	Cost ($/month)
ephSSD	375	733	100000	0.218×375
persSSD	100	48	3000	0.17×100
	250	118	7500	0.17×250
	500	234	15000	0.17×500
persHDD	100	20	150	0.04×100
	250	45	375	0.04×250
	500	97	750	0.04×500
objStore	N/A	265	550	0.026/GB

ephSSD: VM-local ephemeral SSD, **persSSD**: Network-attached persistent SSD, **persHDD**: Network-attached persistent HDD, **objStore**: Google cloud object storage

Ephemeral SSDs offer best perf but without data persistence!

Multi-dimensional heterogeneity in cloud storage services

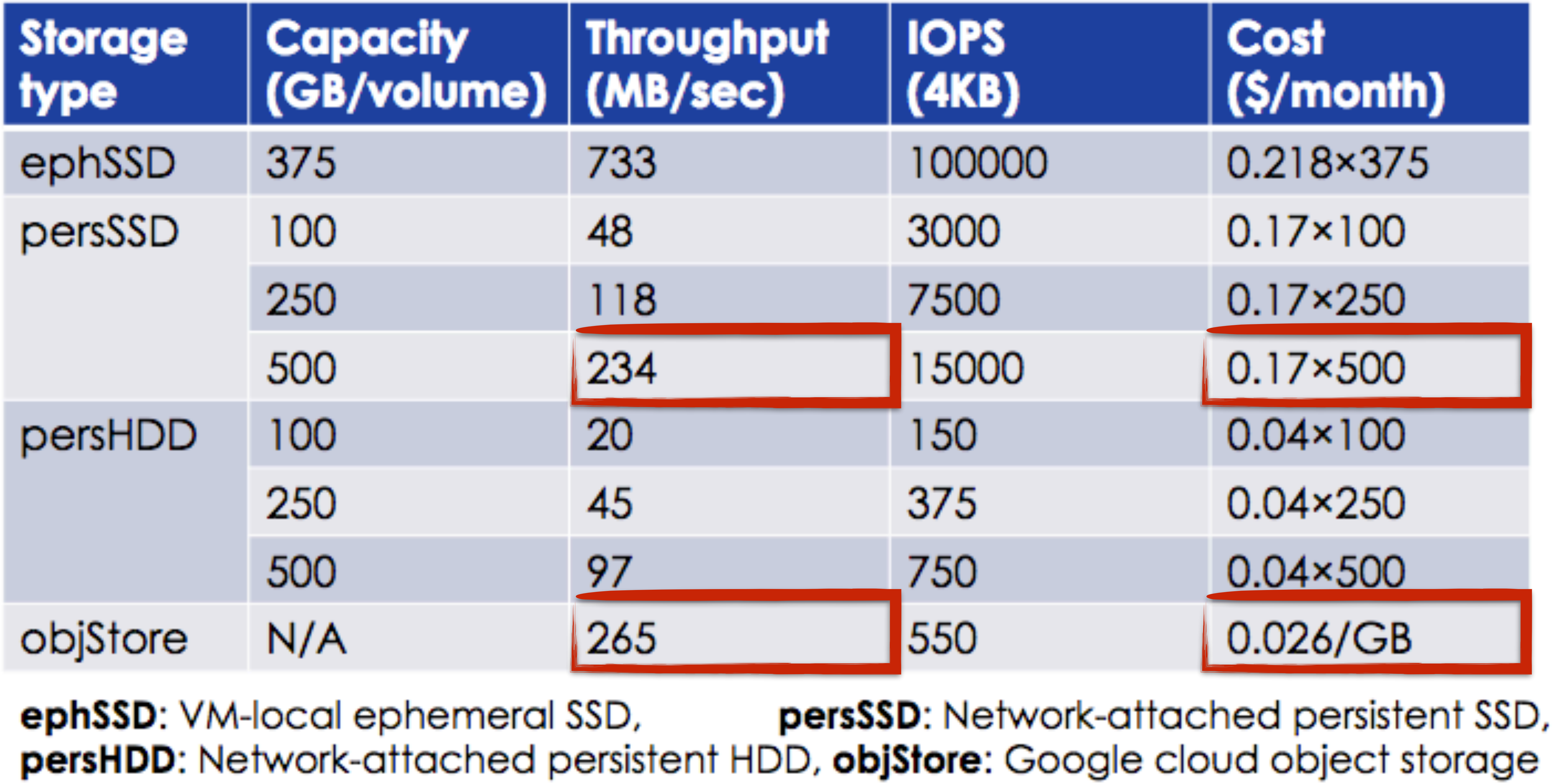


Storage type	Capacity (GB/volume)	Throughput (MB/sec)	IOPS (4KB)	Cost ($/month)
ephSSD	375	733	100000	0.218×375
persSSD	100	48	3000	0.17×100
	250	118	7500	0.17×250
	500	234	15000	0.17×500
persHDD	100	20	150	0.04×100
	250	45	375	0.04×250
	500	97	750	0.04×500
objStore	N/A	265	550	0.026/GB

ephSSD: VM-local ephemeral SSD, **persSSD**: Network-attached persistent SSD, **persHDD**: Network-attached persistent HDD, **objStore**: Google cloud object storage

Perf of the network-attached EBS (block storage) depends on size of the volume!

Multi-dimensional heterogeneity in cloud storage services

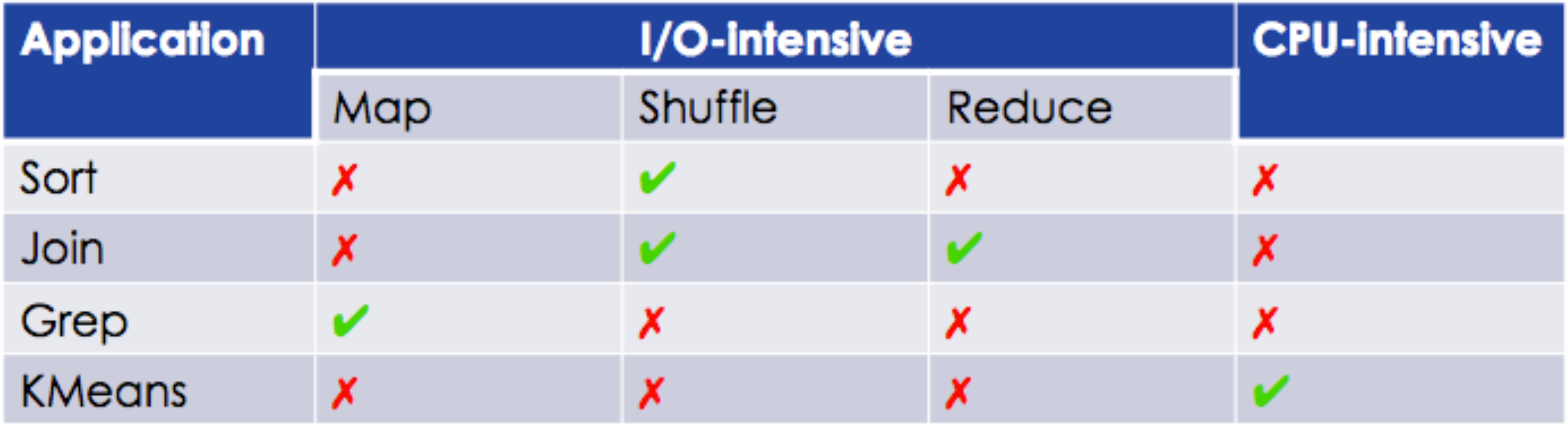


Storage type	Capacity (GB/volume)	Throughput (MB/sec)	IOPS (4KB)	Cost ($/month)
ephSSD	375	733	100000	0.218×375
persSSD	100	48	3000	0.17×100
	250	118	7500	0.17×250
	500	234	15000	0.17×500
persHDD	100	20	150	0.04×100
	250	45	375	0.04×250
	500	97	750	0.04×500
objStore	N/A	265	550	0.026/GB

ephSSD: VM-local ephemeral SSD, **persSSD**: Network-attached persistent SSD, **persHDD**: Network-attached persistent HDD, **objStore**: Google cloud object storage

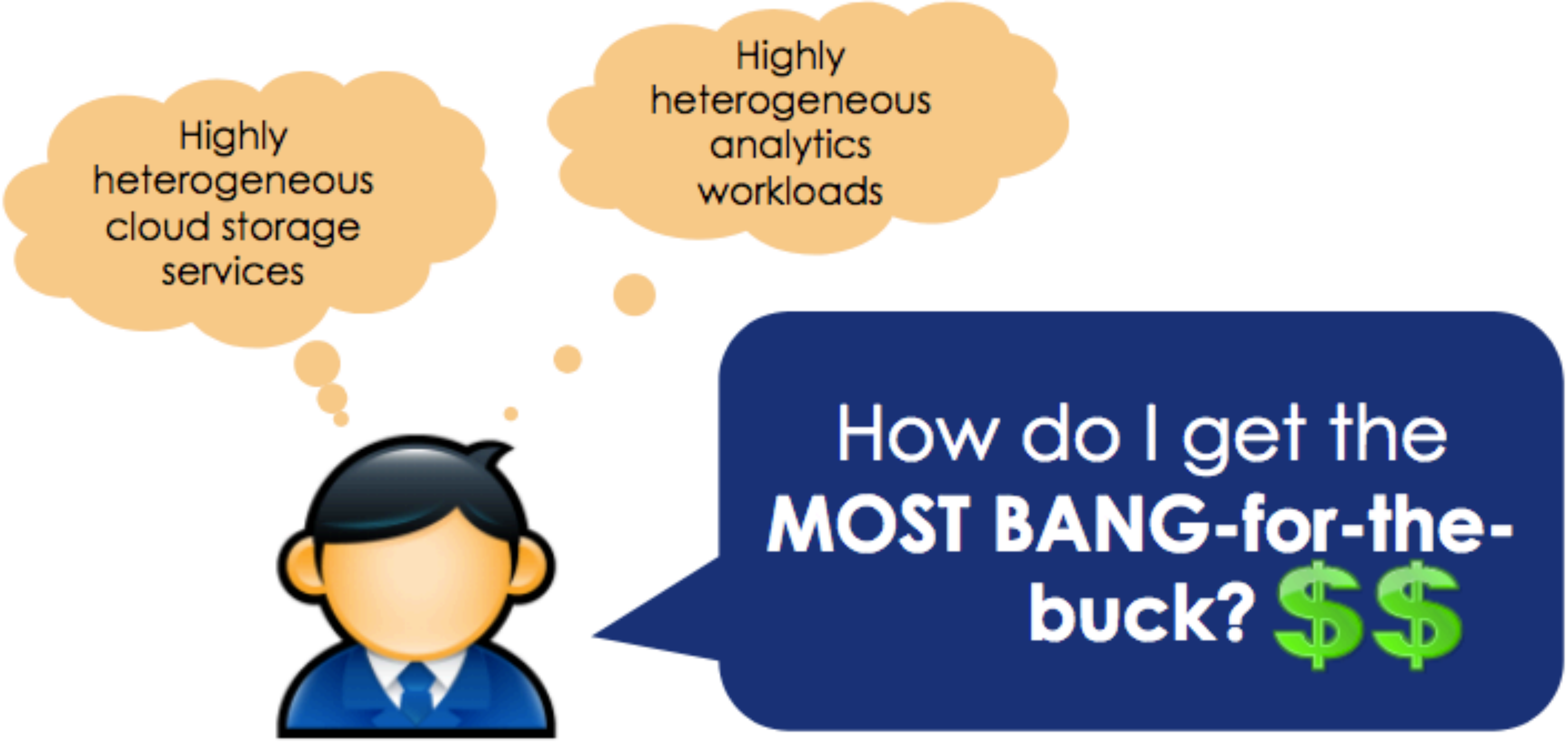
Object store provides the cheapest service but offers comparable sequential IO throughput!

To make things worse: Heterogeneity in data analytics workloads



Application	I/O-Intensive			CPU-Intensive
	Map	Shuffle	Reduce	
Sort	✗	✓	✗	✗
Join	✗	✓	✓	✗
Grep	✓	✗	✗	✗
KMeans	✗	✗	✗	✓

Decision paralysis…



Highly heterogeneous cloud storage services

Highly heterogeneous analytics workloads

How do I get the **MOST BANG-for-the-buck?** $$

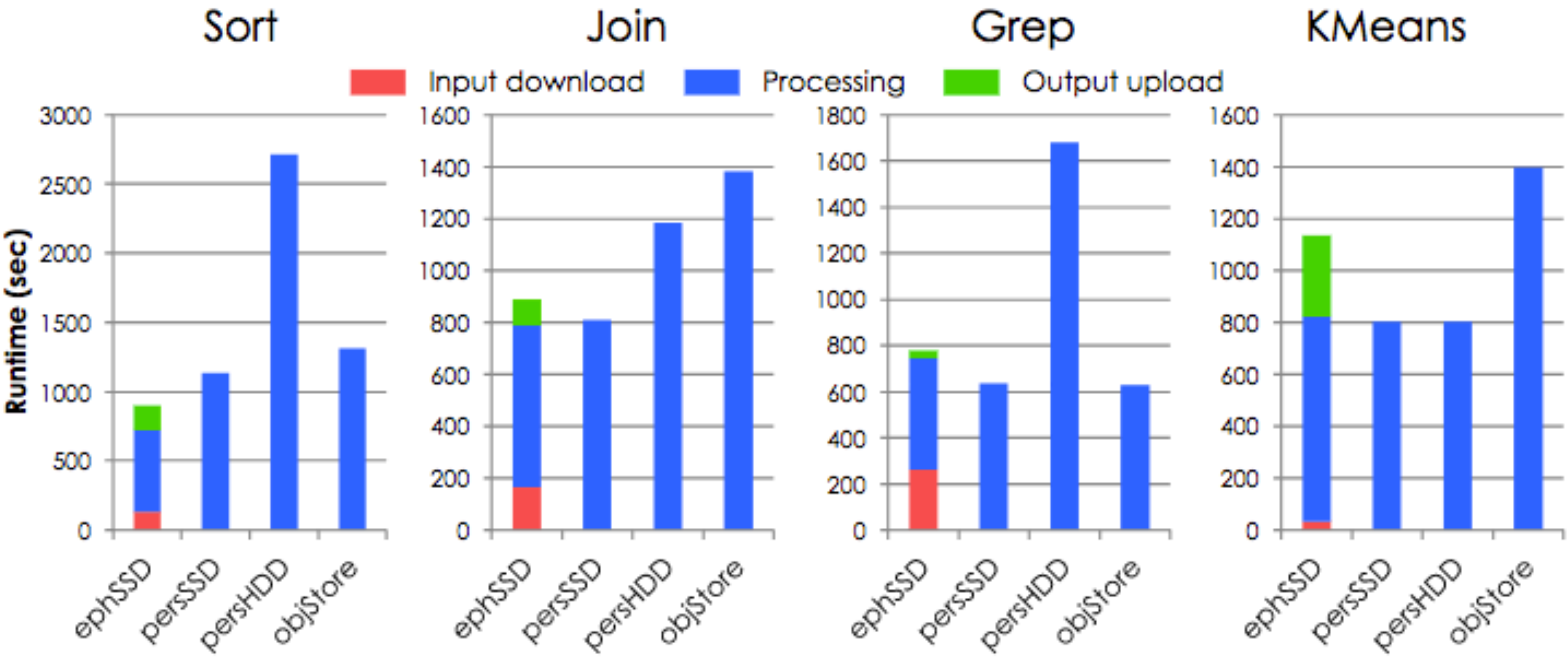
Cloud tenant

What to do?

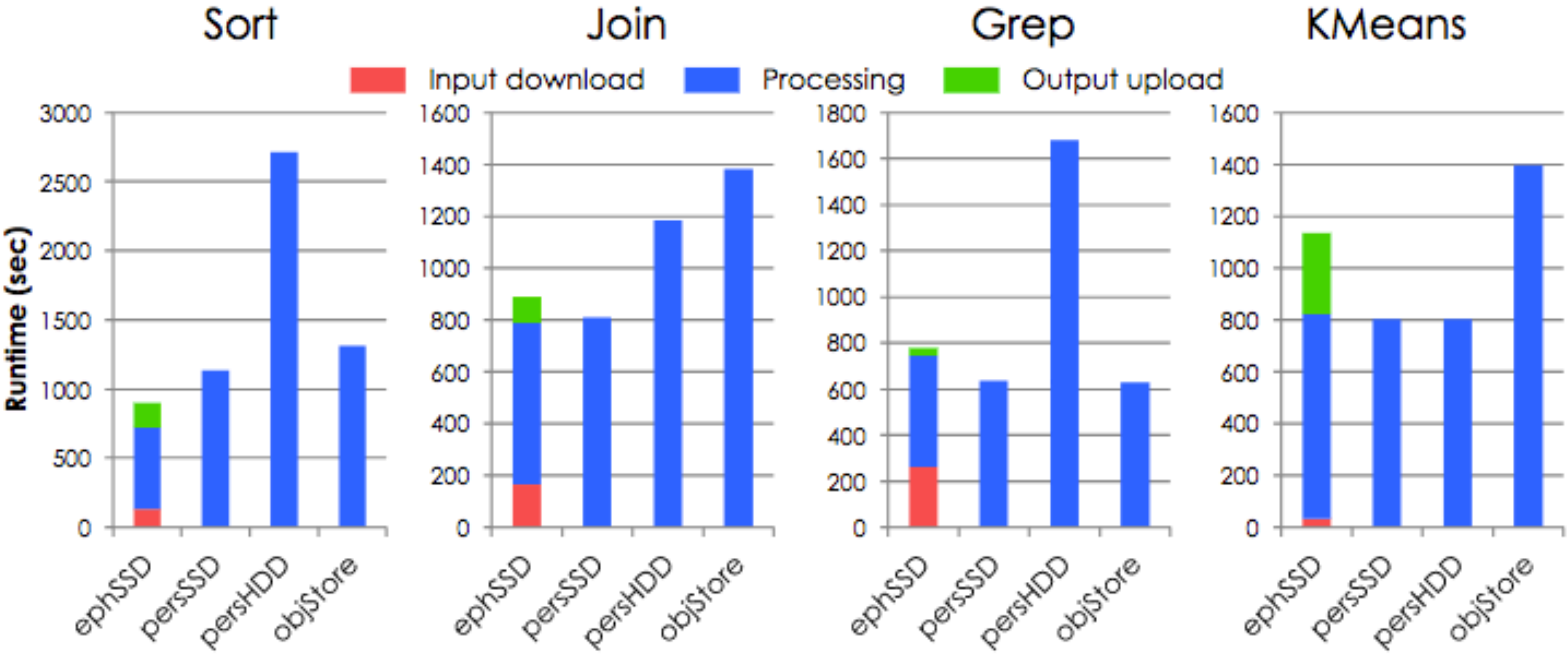
- Not much is known about how current data analytics workloads perform on various cloud storage services

- So, **measure…**

Application performance

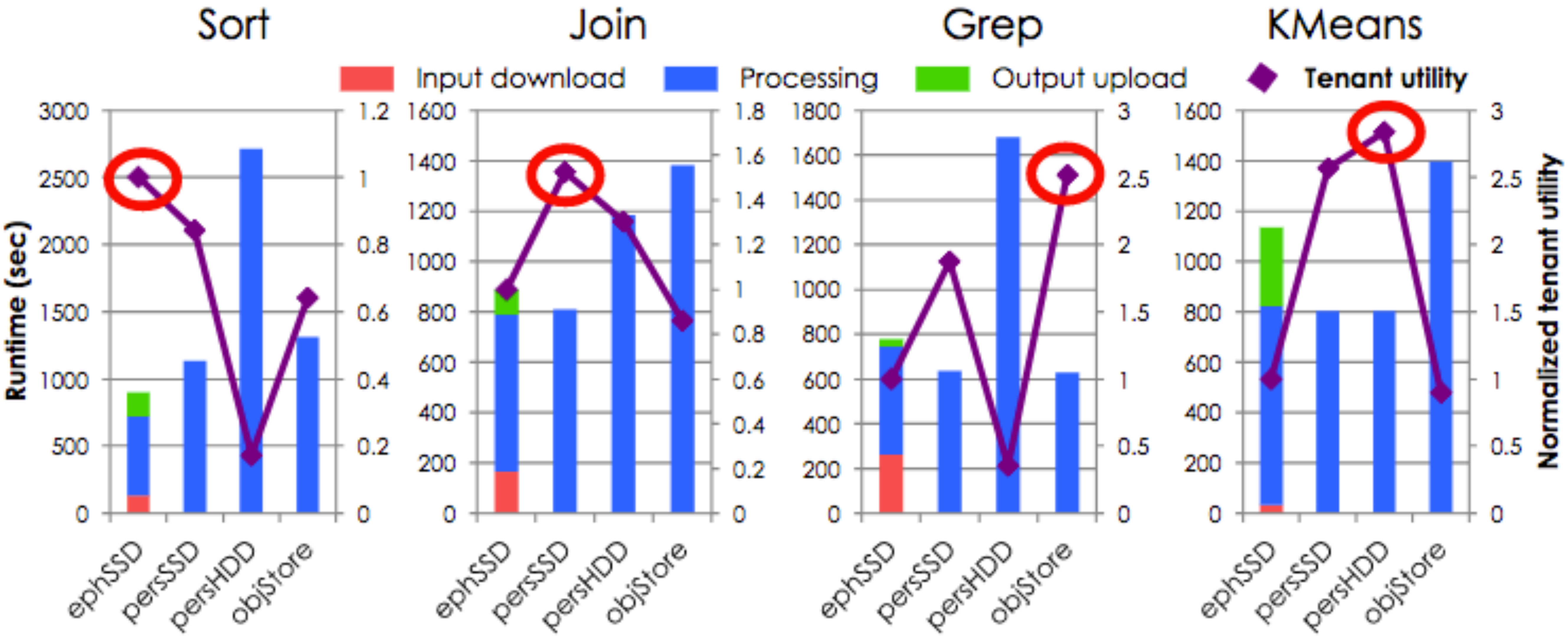


Application performance



No storage service is the best in terms of performance

Application performance/$cost

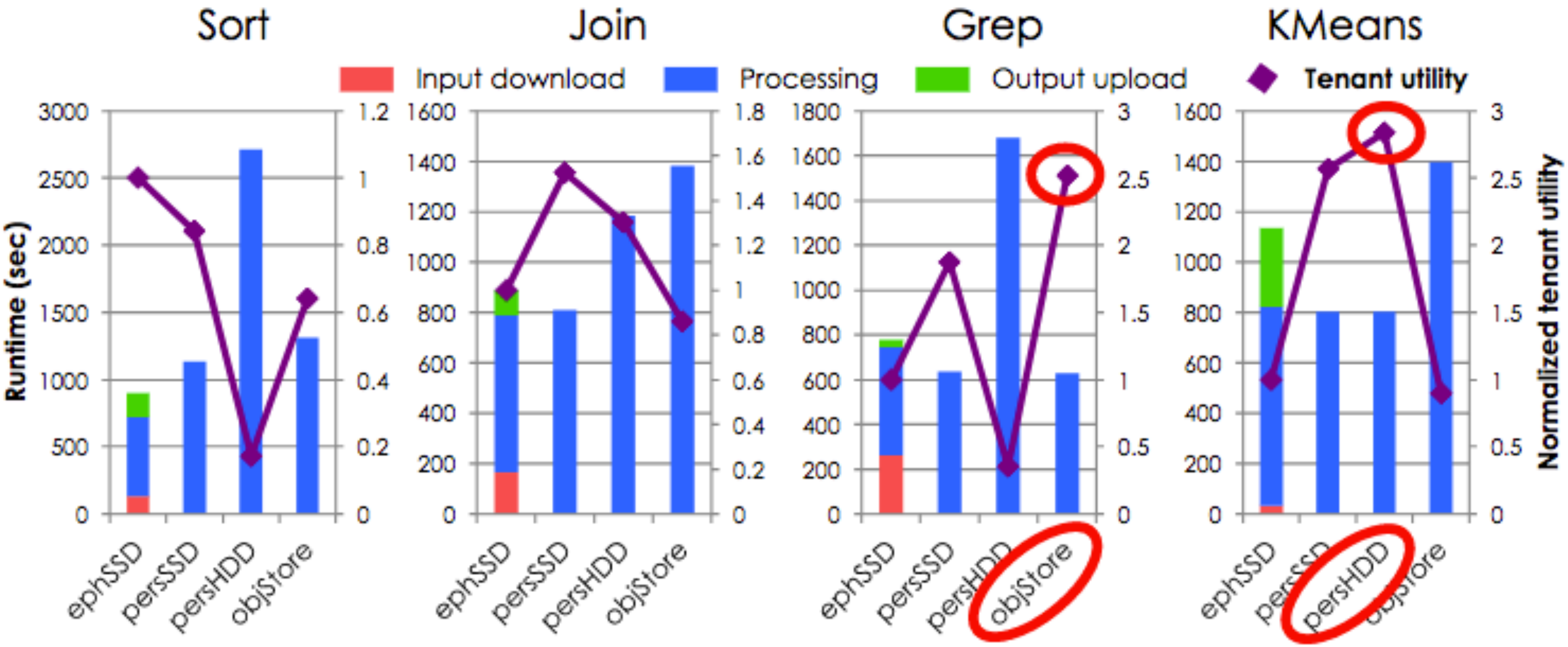


$$\text{Tenant utility} = \frac{1/T}{\$}$$

← Performance

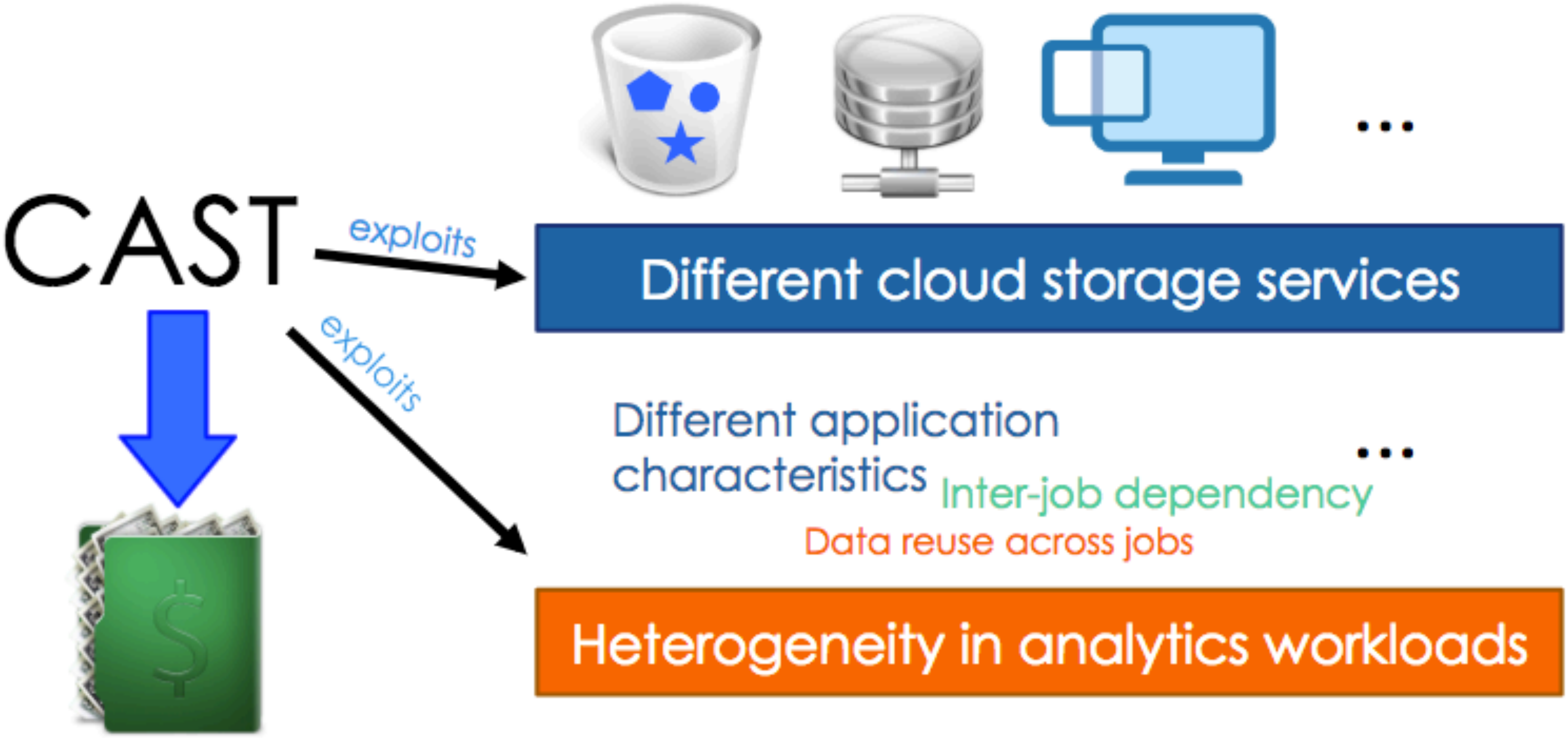
← $ cost

Application performance/$cost



Slower storage, in some case, may provide higher utility and comparable performance

So we built CAST: A **C**loud **A**nalytics **S**torage **T**iering framework



CAST

exploits

Different cloud storage services

exploits

Different application characteristics

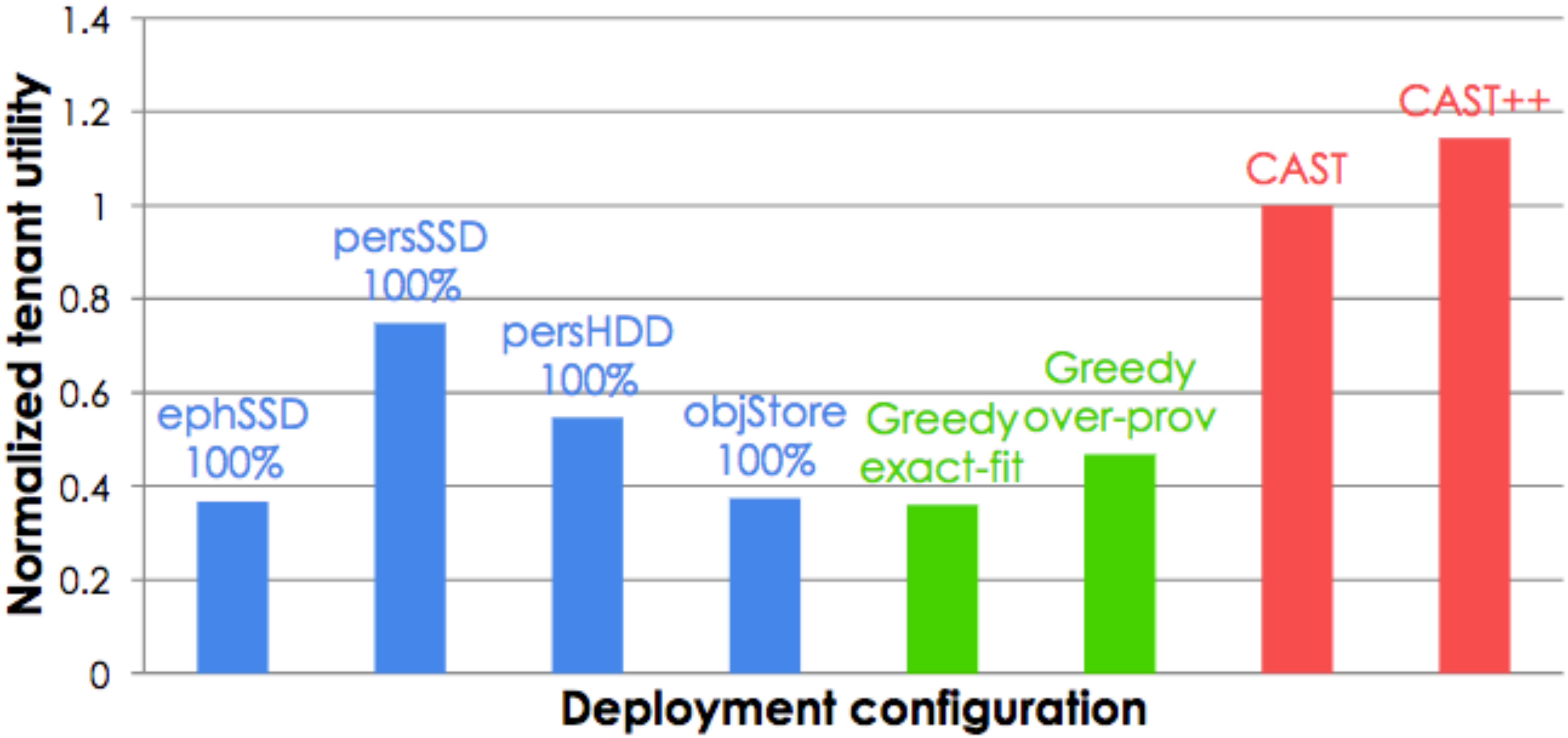
Inter-job dependency

Data reuse across jobs

…

Heterogeneity in analytics workloads

Some results



100-job Hadoop workload, simulating behaviors of Facebook's 3000-machine Hadoop clusters

Increased complexity — More and more requirements

Security

Quality of service

Scale

Latency

$ Cost

Ease of use

Throughput

Accuracy

Ease of management

Reliability

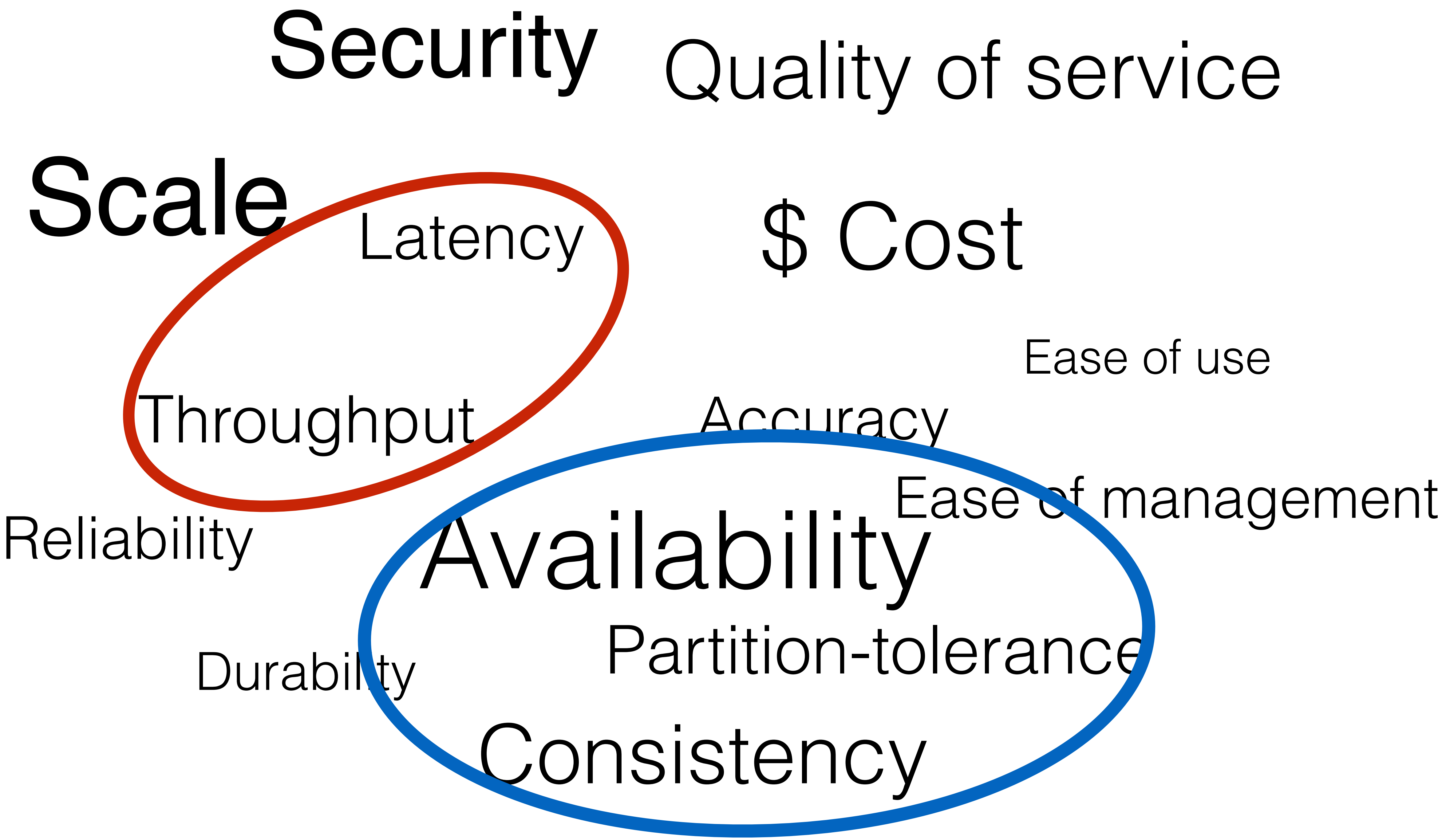
Availability

Partition-tolerance

Durability

Consistency

What are hard/fundamental tradeoffs?



Security

Quality of service

Scale

Latency

$ Cost

Ease of use

Throughput

Accuracy

Ease of management

Reliability

Availability

Partition-tolerance

Durability

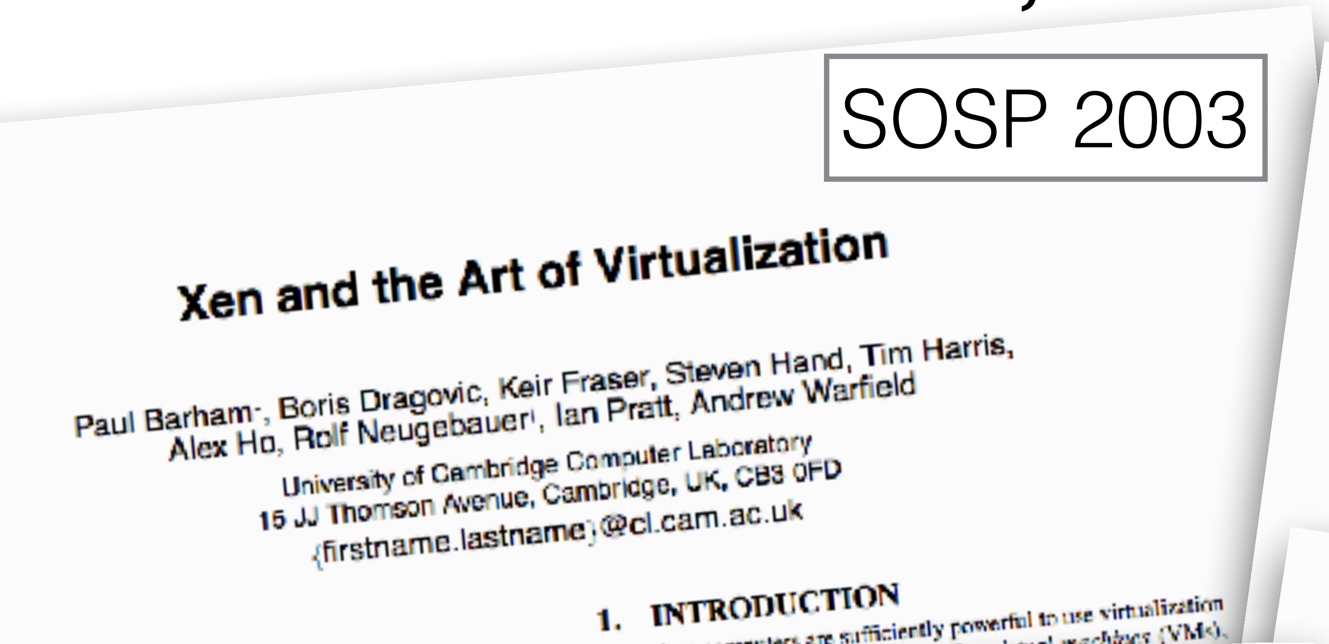
Consistency

Academia <—> Industry

- Top-tier systems research conferences
  - USENIX OSDI, ACM SOSP (bi-annually)
  - ACM EuroSys
  - USENIX NSDI, ACM SIGCOMM
  - USENIX ATC, USENIX FAST

Academia <—> Industry

- Top-tier systems research conferences
  - **USENIX OSDI, ACM SOSP (bi-annually)**



SOSP 2003

Xen and the Art of Virtualization

Paul Barham*, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris,
Alex Ho, Rolf Neugebauer†, Ian Pratt, Andrew Warfield

University of Cambridge Computer Laboratory
15 JJ Thomson Avenue, Cambridge, UK, CB3 0FD
{firstname.lastname}@cl.cam.ac.uk

RACT

1. INTRODUCTION

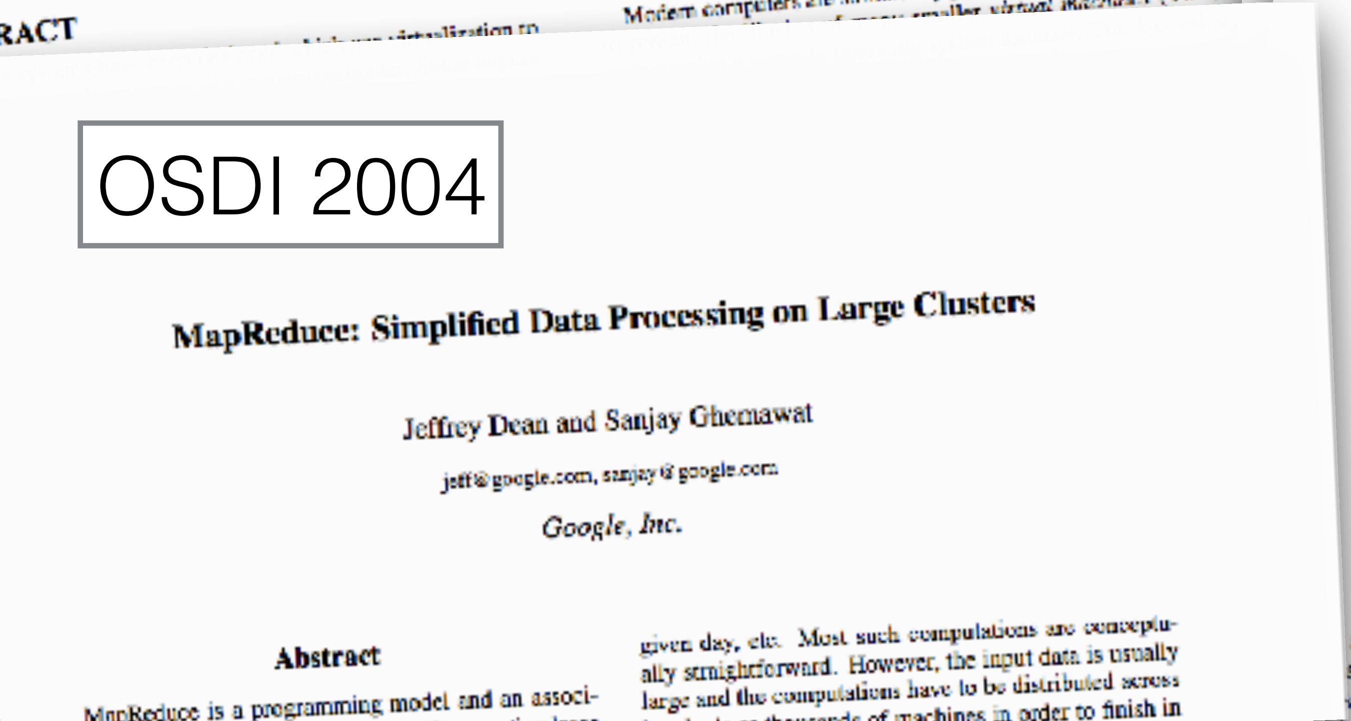
Modern computers are sufficiently powerful to use virtualization ... of many smaller virtual machines (VMs),

SOSP 2003

The Google File System

Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung

Google*



OSDI 2004

MapReduce: Simplified Data Processing on Large Clusters

Jeffrey Dean and Sanjay Ghemawat

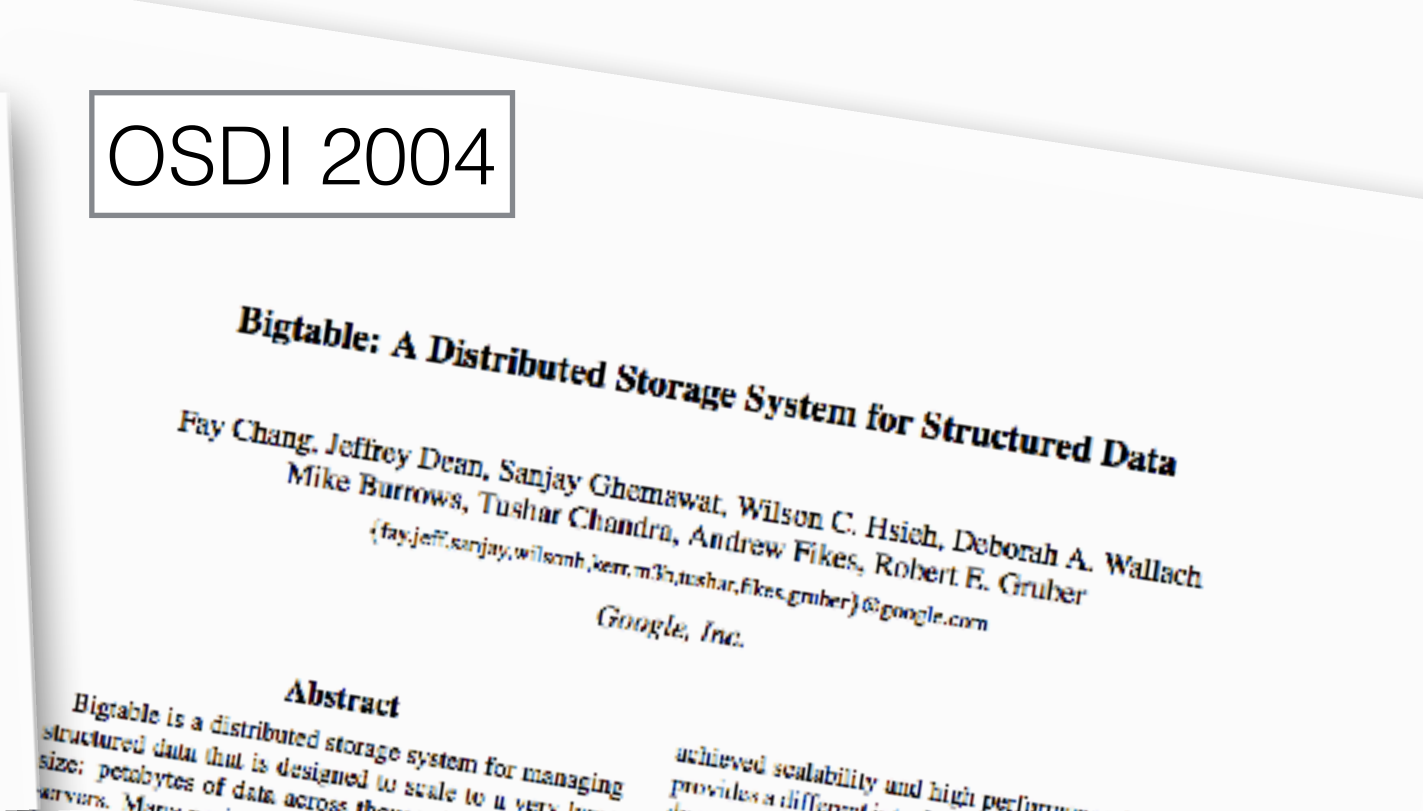
jeff@google.com, sanjay@google.com

Google, Inc.

Abstract

MapReduce is a programming model and an associ-

given day, etc. Most such computations are conceptually straightforward. However, the input data is usually large and the computations have to be distributed across thousands of machines in order to finish in



OSDI 2004

Bigtable: A Distributed Storage System for Structured Data

Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach
Mike Burrows, Tushar Chandra, Andrew Fikes, Robert E. Gruber

{fay,jeff,sanjay,wilsonh,kerr,m3b,tushar,fikes,gruber}@google.com

Google, Inc.

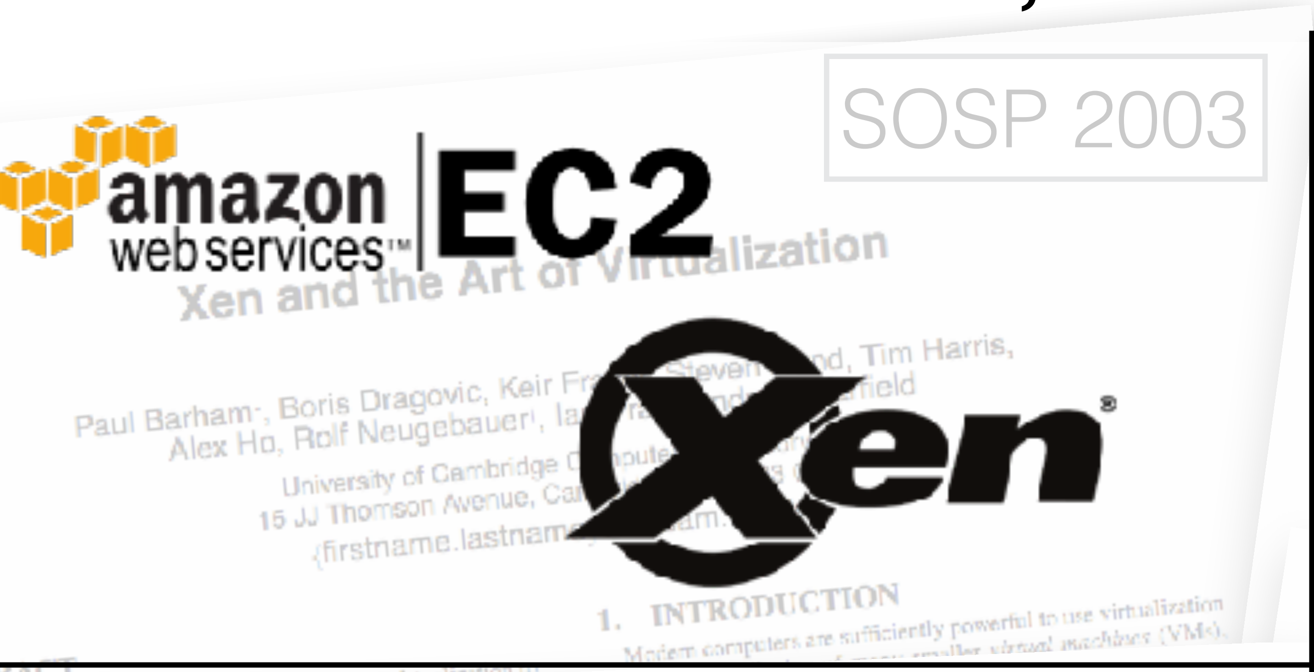
Abstract

Bigtable is a distributed storage system for managing structured data that is designed to scale to a very large size: petabytes of data across thousands of commodity servers. Many

achieved scalability and high performance ... provides a different interface

Academia <—> Industry

- Top-tier systems research conferences
  - **USENIX OSDI, ACM SOSP (bi-annually)**



Xen and the Art of Virtualization

Paul Barham, Boris Dragovic, Keir Fr... Steven ...nd, Tim Harris, Alex Ho, Rolf Neugebauer, Ia... ...field

University of Cambridge C...

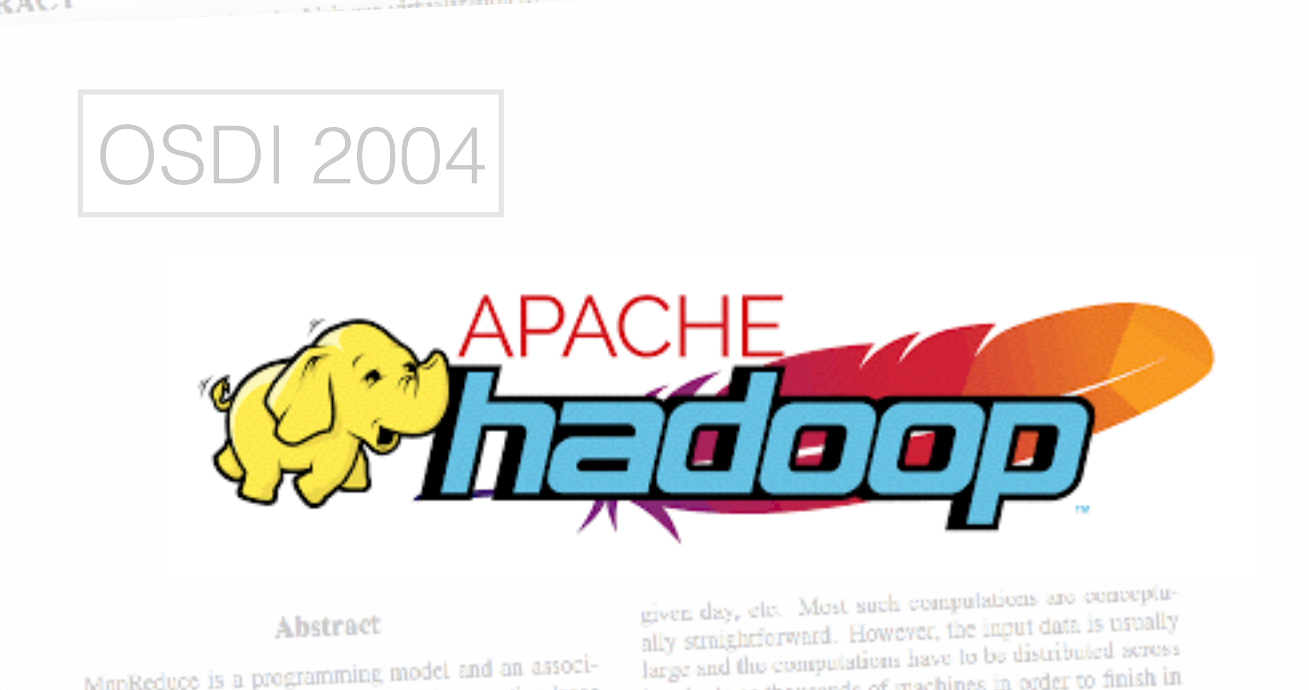
15 JJ Thomson Avenue, Ca...

{firstname.lastnam...

1. INTRODUCTION

Modern computers are sufficiently powerful to use virtualization ... smaller virtual machines (VMs),

Sanjay



Abstract

MapReduce is a programming model and an associ-

given day, etc. Most such computations are conceptually straightforward. However, the input data is usually large and the computations have to be distributed across ... thousands of machines in order to finish in

Bigtable: A Distributed Storage System for Structured

...y Chang, Jeffrey Dean, Sanjay Ghem...

Google, Inc.

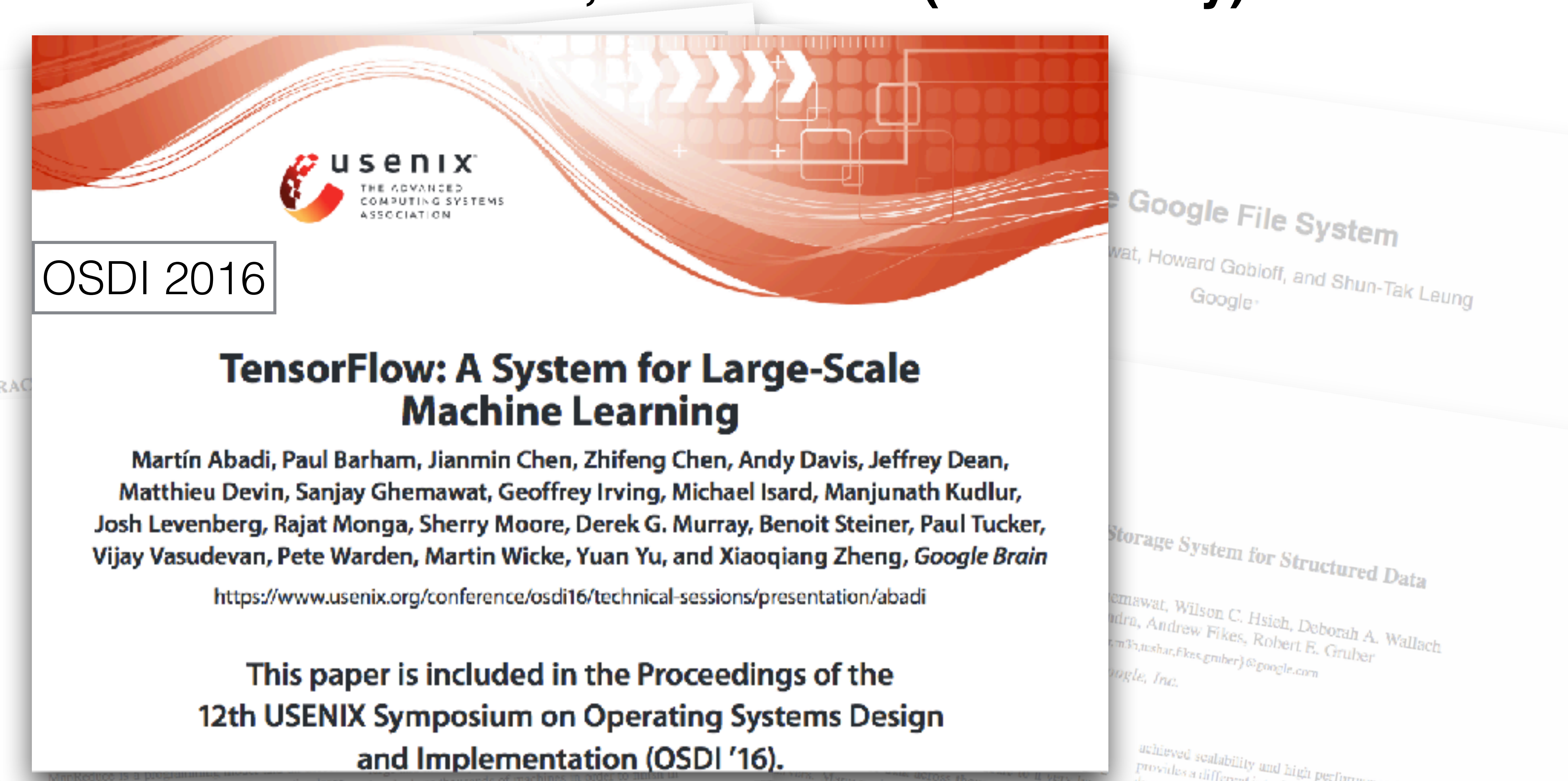
Abstract

Bigtable is a distributed storage system for managing structured data that is designed to scale to a very lar... size: petabytes of data across thou... servers. Man...

achieved scalability and high perform... provides a different...

Academia <—> Industry

- Top-tier systems research conferences
  - **USENIX OSDI, ACM SOSP (bi-annually)**



OSDI 2016

usenix
THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

TensorFlow: A System for Large-Scale Machine Learning

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng, *Google Brain*

https://www.usenix.org/conference/osdi16/technical-sessions/presentation/abadi

This paper is included in the Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI '16).

e Google File System

wat, Howard Gobioff, and Shun-Tak Leung

Google

Storage System for Structured Data

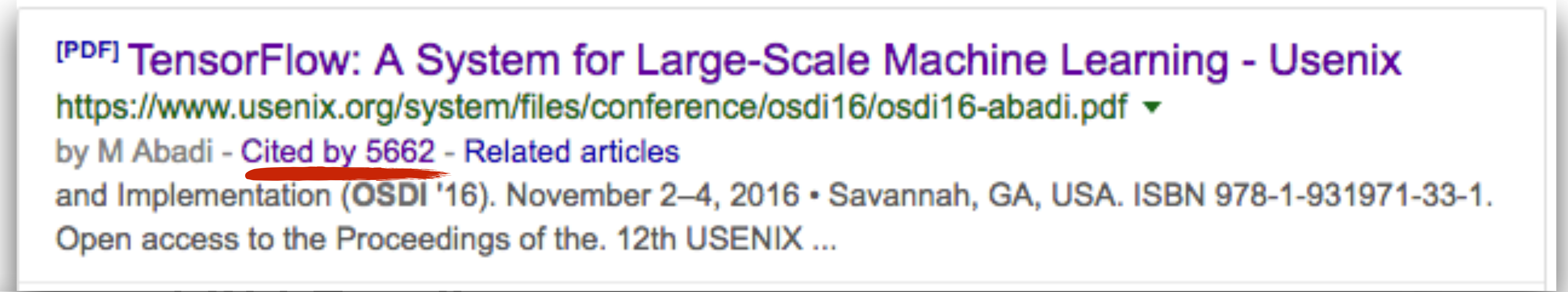
emawat, Wilson C. Hsieh, Deborah A. Wallach
dra, Andrew Fikes, Robert E. Gruber

ogle, Inc.

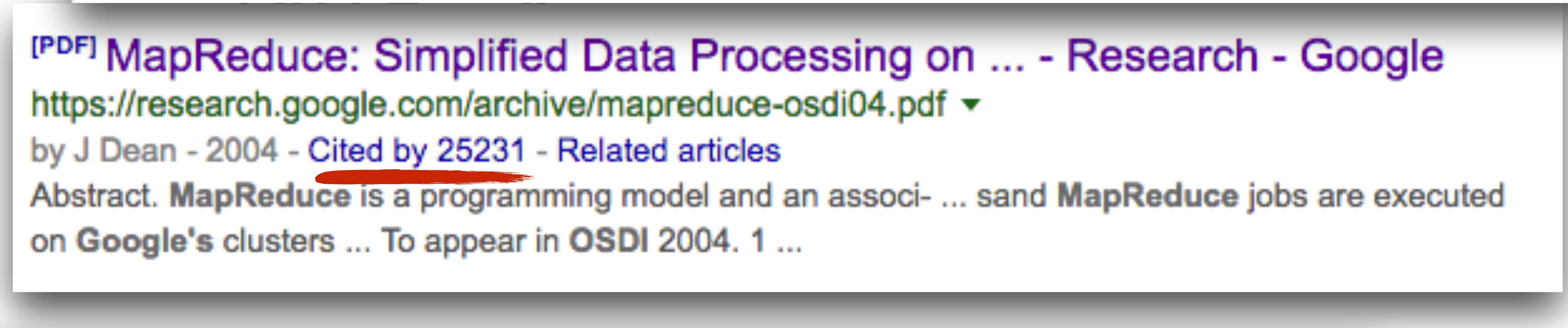
achieved scalability and high perform
provides a different

Academia <—> Industry

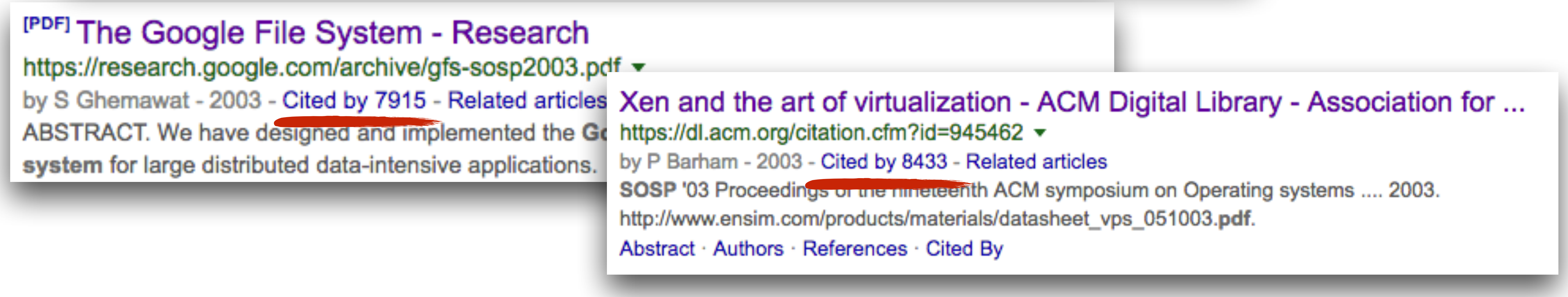
- Top-tier systems research conferences
    - **USENIX OSDI, ACM SOSP (bi-annually)**



[PDF] TensorFlow: A System for Large-Scale Machine Learning - Usenix
https://www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf
by M Abadi - Cited by 5662 - Related articles
and Implementation (**OSDI** '16). November 2–4, 2016 • Savannah, GA, USA. ISBN 978-1-931971-33-1.
Open access to the Proceedings of the. 12th USENIX ...



[PDF] MapReduce: Simplified Data Processing on ... - Research - Google
https://research.google.com/archive/mapreduce-osdi04.pdf
by J Dean - 2004 - Cited by 25231 - Related articles
Abstract. **MapReduce** is a programming model and an associ- ... sand **MapReduce** jobs are executed
on **Google's** clusters ... To appear in **OSDI** 2004. 1 ...

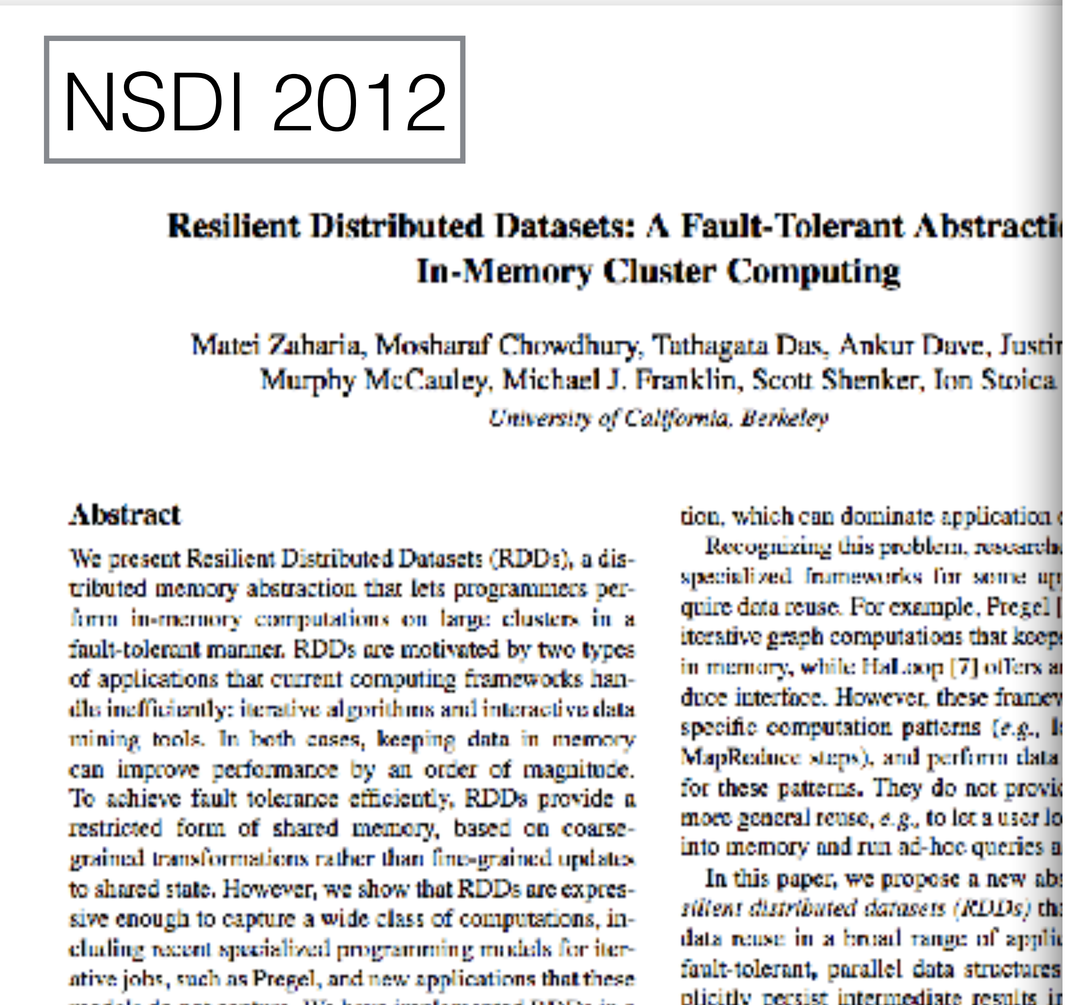


[PDF] The Google File System - Research
https://research.google.com/archive/gfs-sosp2003.pdf
by S Ghemawat - 2003 - Cited by 7915 - Related articles
ABSTRACT. We have designed and implemented the G
system for large distributed data-intensive applications.

Xen and the art of virtualization - ACM Digital Library - Association for ...
https://dl.acm.org/citation.cfm?id=945462
by P Barham - 2003 - Cited by 8433 - Related articles
SOSP '03 Proceedings of the nineteenth ACM symposium on Operating systems .... 2003.
http://www.ensim.com/products/materials/datasheet_vps_051003.**pdf**.
Abstract · Authors · References · Cited By

Academia <—> Industry

- Top-tier systems research conferences
  - USENIX OSDI, ACM SOSP (bi-annually)
  - **ACM EuroSys**
  - **USENIX NSDI**,



NSDI 2012

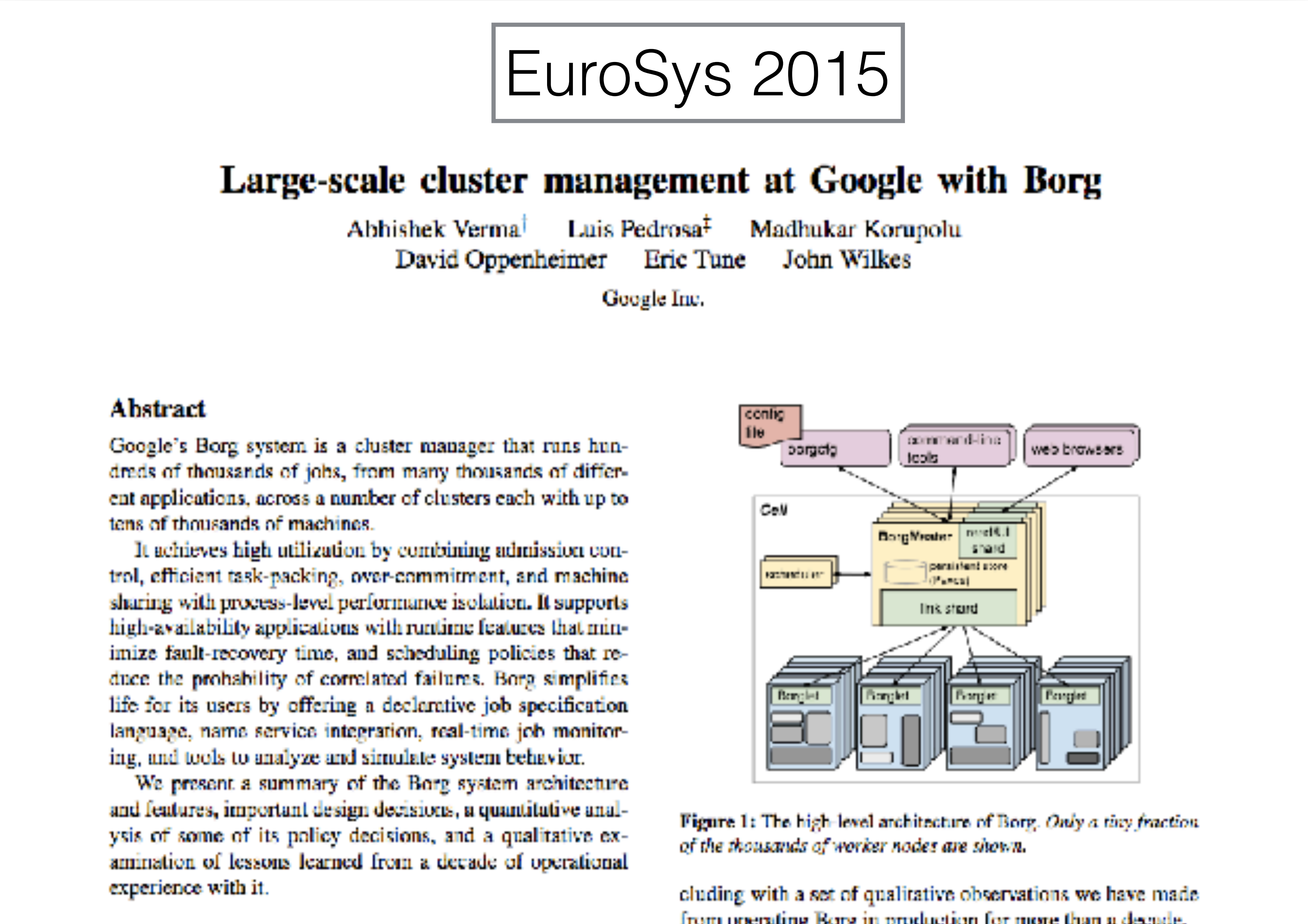
Resilient Distributed Datasets: A Fault-Tolerant Abstracti
In-Memory Cluster Computing

Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justi
Murphy McCauley, Michael J. Franklin, Scott Shenker, Ion Stoica
University of California, Berkeley

Abstract

We present Resilient Distributed Datasets (RDDs), a distributed memory abstraction that lets programmers perform in-memory computations on large clusters in a fault-tolerant manner. RDDs are motivated by two types of applications that current computing frameworks handle inefficiently: iterative algorithms and interactive data mining tools. In both cases, keeping data in memory can improve performance by an order of magnitude. To achieve fault tolerance efficiently, RDDs provide a restricted form of shared memory, based on coarse-grained transformations rather than fine-grained updates to shared state. However, we show that RDDs are expressive enough to capture a wide class of computations, including recent specialized programming models for iterative jobs, such as Pregel, and new applications that these

tion, which can dominate application
Recognizing this problem, research
specialized frameworks for some ap
quire data reuse. For example, Pregel
iterative graph computations that keep
in memory, while HaLoop [7] offers a
duce interface. However, these framew
specific computation patterns (*e.g.*, l
MapReduce steps), and perform data
for these patterns. They do not provi
more general reuse, *e.g.*, to let a user lo
into memory and run ad-hoc queries a
In this paper, we propose a new ab
silient distributed datasets (RDDs) th
data reuse in a broad range of applic
fault-tolerant, parallel data structures
plicitly persist intermediate results i



EuroSys 2015

Large-scale cluster management at Google with Borg

Abhishek Verma[†] Luis Pedrosa[‡] Madhukar Korupolu
David Oppenheimer Eric Tune John Wilkes
Google Inc.

Abstract

Google's Borg system is a cluster manager that runs hundreds of thousands of jobs, from many thousands of different applications, across a number of clusters each with up to tens of thousands of machines.

It achieves high utilization by combining admission control, efficient task-packing, over-commitment, and machine sharing with process-level performance isolation. It supports high-availability applications with runtime features that minimize fault-recovery time, and scheduling policies that reduce the probability of correlated failures. Borg simplifies life for its users by offering a declarative job specification language, name service integration, real-time job monitoring, and tools to analyze and simulate system behavior.

We present a summary of the Borg system architecture and features, important design decisions, a quantitative analysis of some of its policy decisions, and a qualitative examination of lessons learned from a decade of operational experience with it.

Figure 1: The high-level architecture of Borg. *Only a tiny fraction of the thousands of worker nodes are shown.*

cluding with a set of qualitative observations we have made from operating Borg in production for more than a decade.

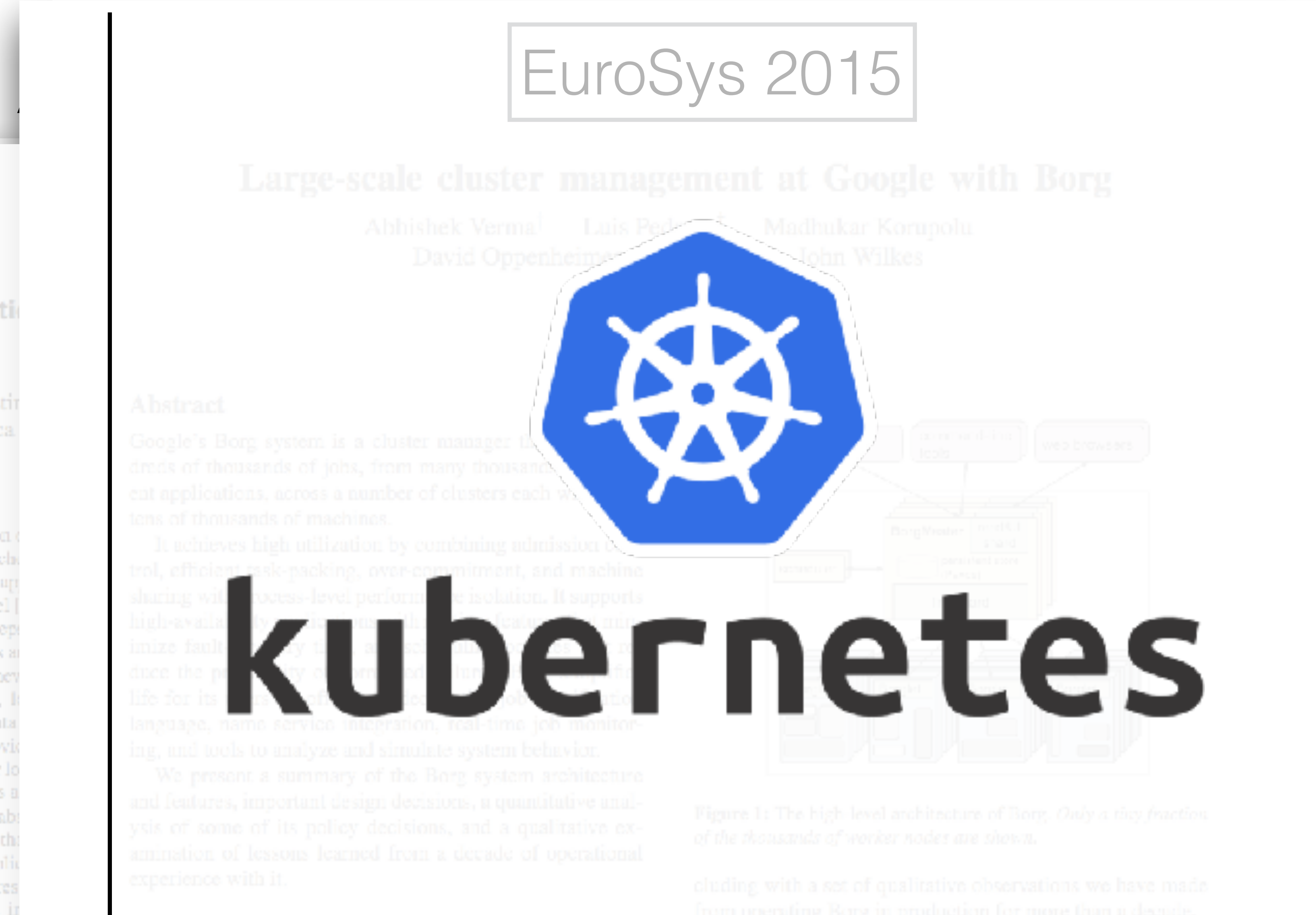
Academia <—> Industry

- Top-tier systems research conferences
  - USENIX OSDI, ACM SOSP (bi-annually)
  - **ACM EuroSys**
  - **USENIX NSDI**,

NSDI 2012

Resilient Distributed Datasets: A Fault-T nt Abstracti
In-Memory Cluster Con

Matei Zaharia, Mosharaf Chowdhury, Tath Ankur D Justi
Murphy McCauley, Michael J. Fr Scott Shenker Stoica



EuroSys 2015

Large-scale cluster management at Google with Borg

Abhishek Verma Luis Pedr Madhukar Korupolu
David Oppenheimer John Wilkes

Abstract

Figure 1: The high-level architecture of Borg. *Only a tiny fraction of the thousands of worker nodes are shown.*

Overview of topics

- Distributed systems foundation
    - Distributed consensus algorithms: 2PC, 3PC, Paxos, Raft
    - Data consistency: Strong consistency, Eventual consistency, causal consistency
- Cloud storage
    - Amazon Dynamo, Hybrid-cloud storage, Azure cloud storage, Google Spanner, LinkedIn Ambry object store, *BespoKV* **[Sneak peek: our must recent paper:)]**

Overview of topics

- Container-based virtualization
  - Container virtualization, container registry, IBM’s Docker registry workload characterization
- Serverless computing
  - Serverless architecture, current state, innovative applications

Overview of topics

- Distributed machine learning
  - Ray, Tensorflow, Parameter server

- Big data systems
  - Google MapReduce, Google File System (GFS), Google Bigtable, Hadoop YARN, Apache Spark

Overview of topics

- Cluster and datacenter resource management
  - Google Borg, Apache Mesos, Omega, Sparrow, Quasar, etc.
- Memory-driven computing
  - Distributed memory caching, RAMCloud, Facebook's Memcache
  - Memcached's load balancing, Tachyon's erasure coding based load balancing

Announcement

- Homework assignment #0 (0%):
    - A one-pager self-intro (get to know each other better :)
    - Piazza sign-up
    - Paper presentation sign-up (FCFS: will send out a Doodle link)
    - AWS Educate sign-up

- Next class
    - Distributed consensus algorithms
    - Data consistency
    - Take a look at all papers on website for next class
    - Release of homework assignment #1