CS310 Operating System

Lecture 43 : Datacenter and Cloud Computing

Ravi Mittal

ravi.mittal@iitgoa.ac.in

Indian Institute of Technology, Goa

We will study

- Computer Evolution
- Why Datacenters and Cloud
- Datacenter and Cloud Computing
- What is inside a Datacenter?
- Requirements of Datacenters and Cloud
- Cloud Computing Services
- OS for Datacenters / Cloud

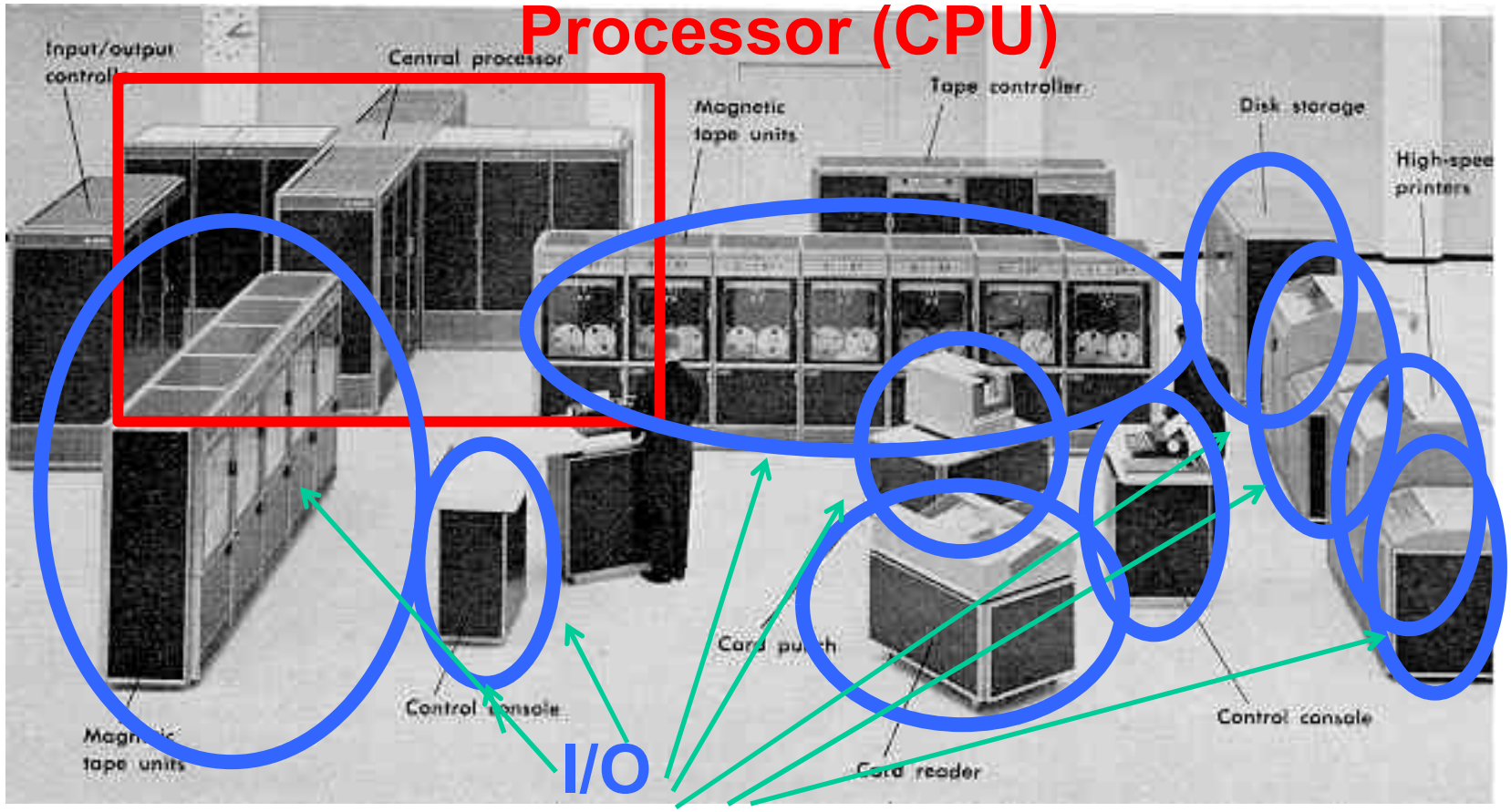
Acknowledgements

Contents of this presentation has been taken from various sources. Would like to thank original contributors

- Course 61c, Great Ideas in Computer Architecture, University of California, Berkeley, lecture Input/Output, Prof Asonovic and Prof Katz
- Course CS162, Operating Systems and Systems Programming, Lecture 25, Cloud Services and Cloud Infrastructure, Prof. Anthony D. Joseph, University of California, Berkeley, Year 2016

Computer Evolution

Computer Eras: Mainframe 1950s-60s



"Big Iron": IBM, UNIVAC, … build $1M computers for businesses ➔ COBOL, Fortran, timesharing OS

Minicomputer Eras: 1970s

Using integrated circuits, Digital, HP… build $10k computers for labs, universities ➔ C, UNIX OS

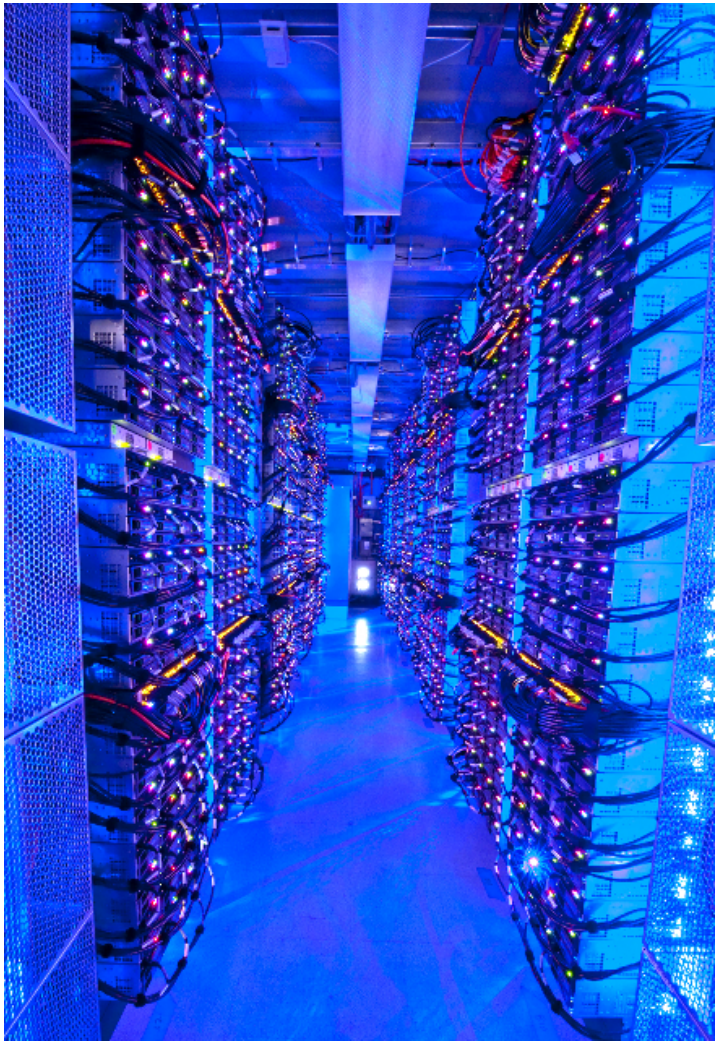
PC Era: Mid 1980s - Mid 2000s

Using microprocessors, Apple, IBM, … build $1k computer for 1 person ➔ Basic, Java, Windows OS

PostPC Era: Late 2000s - ??

Personal Mobile Devices (PMD): Relying on wireless networking, Apple, Nokia, … build $500 smartphone and tablet computers for individuals
➔ Objective C, Swift, Java, Android OS + iOS

Cloud Computing:
Using Local Area Networks, Amazon, Google, … build $200M Warehouse Scale Computers with 100,000 servers for Internet Services for PMDs

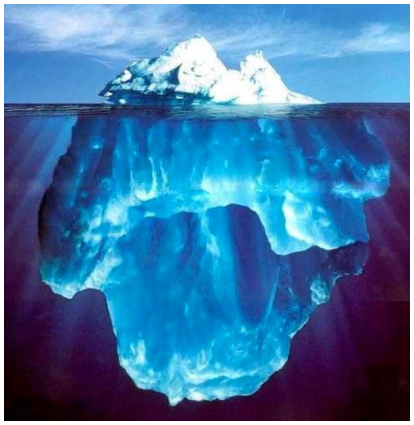


Ref: CS 61C, Prof Asonovic, Univ of California, Berkeley

Why Datacenters and Cloud?

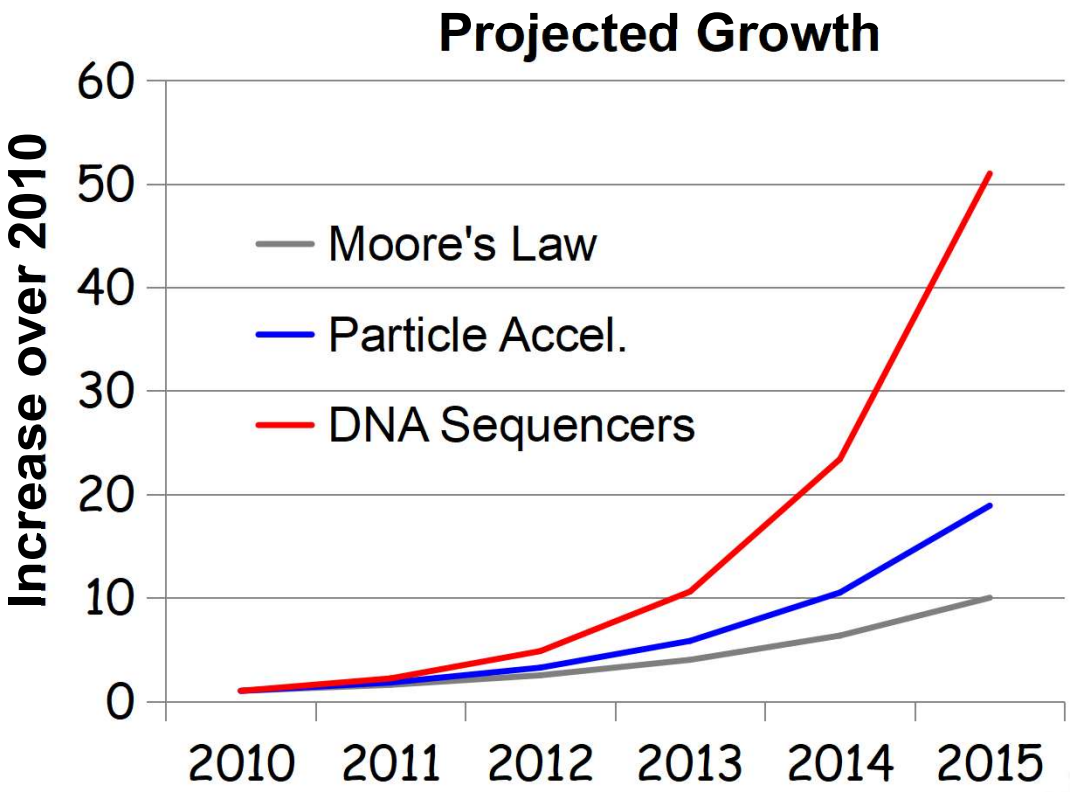
Data Deluge

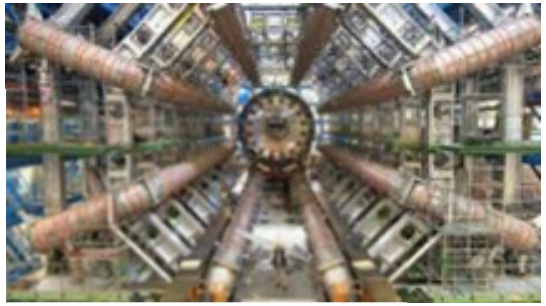
- Billions of users connected through the net
  - WWW, FB, twitter, cell phones, …
  - 80% of the data on FB was produced last year
  - FB building Exabyte ($2^{60} \cong 10^{18}$) data centers



- It's all happening online – could record every:
  - Click, ad impression, billing event, server request, transaction, network messages, fault, fast forward, pause, skip, …
- User Generated Content (Web & Mobile)
  - Facebook, Instagram, TripAdvisor, Twitter, YouTube, …

Data Grows Faster than Moore's Law





Proliferation of Demanding Applications

- These application demand: high computing power, high memory, high availability, and low access time
- Cloud email: Gmail, Yahoo mail, etc
- Search
- On-demand Video services
- Electronic transactions
- Social Networking
- High resolution – gaming and other applications

Ref: CS 61C, Prof Asonovic, Univ of California, Berkeley

Requirement of On Demand Computing

- A user may want to do a huge amount of computing for a short period
  - And pay only for that period
  - On-Demand Instances let you pay for compute capacity by the hour or second (minimum of 60 seconds) with no long-term commitments (from AWS website)
- Today's cloud provider can provide thousands of computer to anyone for a short period
- Corporates need not bother to manage and maintain their own farms of servers and expensive IT team
  - Reduced Infrastructure cost
  - Reduced Human cost
  - No maintenance and updates at all
  - High security

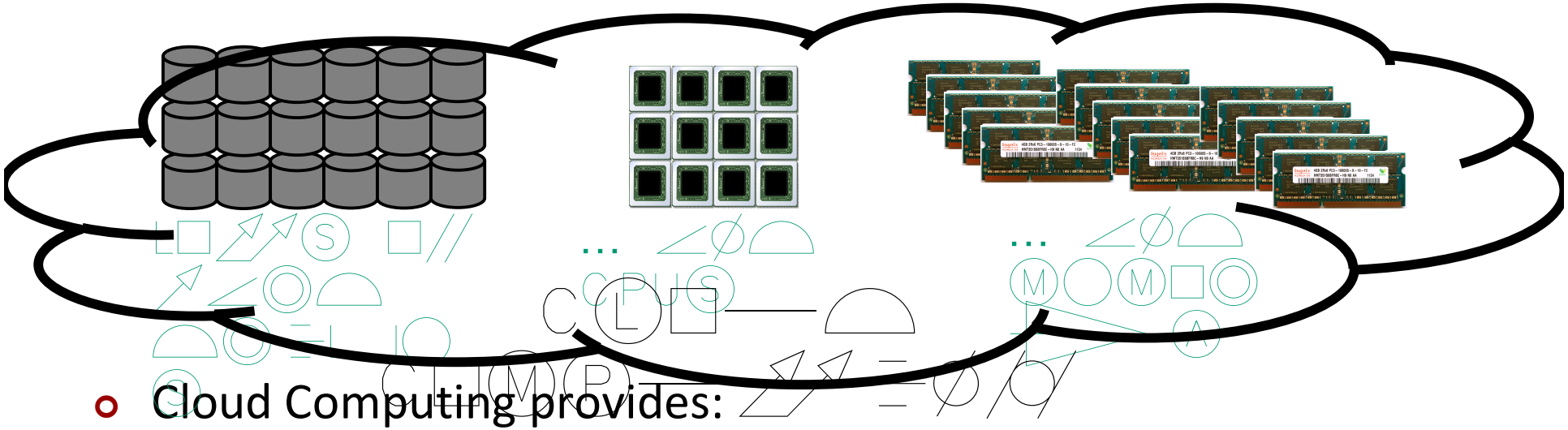
Ref: CS 61C, Prof Asonovic, Univ of California, Berkeley

Solving the Mismatch

- Computers not getting faster, and we are drowning in data
  - How to resolve the dilemma?
- New applications demand huge computing resources: CPU Cycles, Memory
- On demand computing Model
- Solution adopted by web-scale companies
  - Go massively *distributed* and *parallel*

The Big Data Solution: Cloud Computing

- One machine can not process or even store all the data
- Solution: distribute data over cluster of cheap machines



- Cloud Computing provides:
  - Illusion of infinite resources
  - Short-term, on-demand resource allocation
  - Can be much less expensive than owning computers
  - Access to latest technologies (SSDs, GPUs, …)

Why Cloud Computing Now?

- Extremely large datacenters (10,000's of ***commodity*** PCs)
- Economy of scale: 5-7x cheaper than provisioning a medium-sized (1000 servers) facility
- More pervasive broadband Internet so can access remote computers efficiently
- Commoditization of hardware and software

Ref: CS 61C, Prof Asonovic, Univ of California, Berkeley

November 2019 AWS Instances & Prices
aws.amazon.com/ec2/pricing/on-demand

Instance	Per Hour	$ Ratio to Small	EC2 Compute Unit (integer)	Virtual Cores (vCPU)	Memory (GiB)	Disk (GiB)
Standard Small (t3.small)	$0.021	1	Variable	2	2	EBS
Standard Large (t3.large)	$0.083	4	Variable	2	8	EBS
Standard 2x Extra Large (t3.2xlarge)	$0.333	16	Variable	8	32	EBS
High-Mem Large (r5.large)	$0.140	6.7	9	2	16	EBS
High-Mem Double Xlarge (r5.2xlarge)	$0.504	24	38	8	64	EBS
High-Mem 24x Large (r5.24xlarge)	$6.048	288	347	96	768	EBS
High-CPU Large (c5.large)	$0.085	4	9	2	4	EBS
High-CPU 18x Large (c5.18xlarge)	$3.060	146	281	72	144	EBS

- Closest computer in WSC example is Standard 2X Extra Large
- At these low rates, Amazon EC2 can make money! (even utilized 50% time)
- EBS = Elastic Block Store (SSD=$0.12/GiB-month, HDD=$0.054/GiB-month)
- Each also comes with dedicated attached SSD if you choose & pay for that

Ref: CS 61C, Prof Asonovic, Univ of California, Berkeley

Datacenter and Cloud Computing

ARE YOU SURE THIS IS HOW WE GET DATA INTO THE CLOUD?
IT

Datacenter, Cloud, WSC

- Datacenter: Refers to hardware software Infrastructure consisting of thousands of networked servers
  - Compute, storage, and networking elements working in concert
  - A datacenter can be viewed as a very large size Computer: Warehouse Scale Computer (WSC)
  - Datacenter can be private to an organization or Public Datacenter

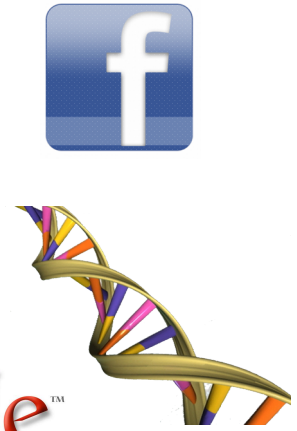
- Cloud: A cloud or a cloud data service is a virtual version of a datacenter – located somewhere away from your company's physical premises – that provides you computing, storage services

Datacenters: Warehouse Scale Computers

- Massive scale datacenters: 10,000 to 100,000 servers + networks to connect them together
  - Emphasize cost-efficiency
  - Attention to power: distribution and cooling
  - (relatively) homogeneous hardware/software
- Can be viewed as **Single gigantic** machine (WSC)
- Offer very large applications (Internet services): search, social networking, video sharing
- Very highly available: < 1 hour down/year
  - Must cope with failures common at scale
- "…WSCs are no less worthy of the expertise of computer systems architects than any other class of machines" Barroso and Hoelzle 2009

Ref: CS 61C, Prof Asonovic, Univ of California, Berkeley

Types of Data Centers

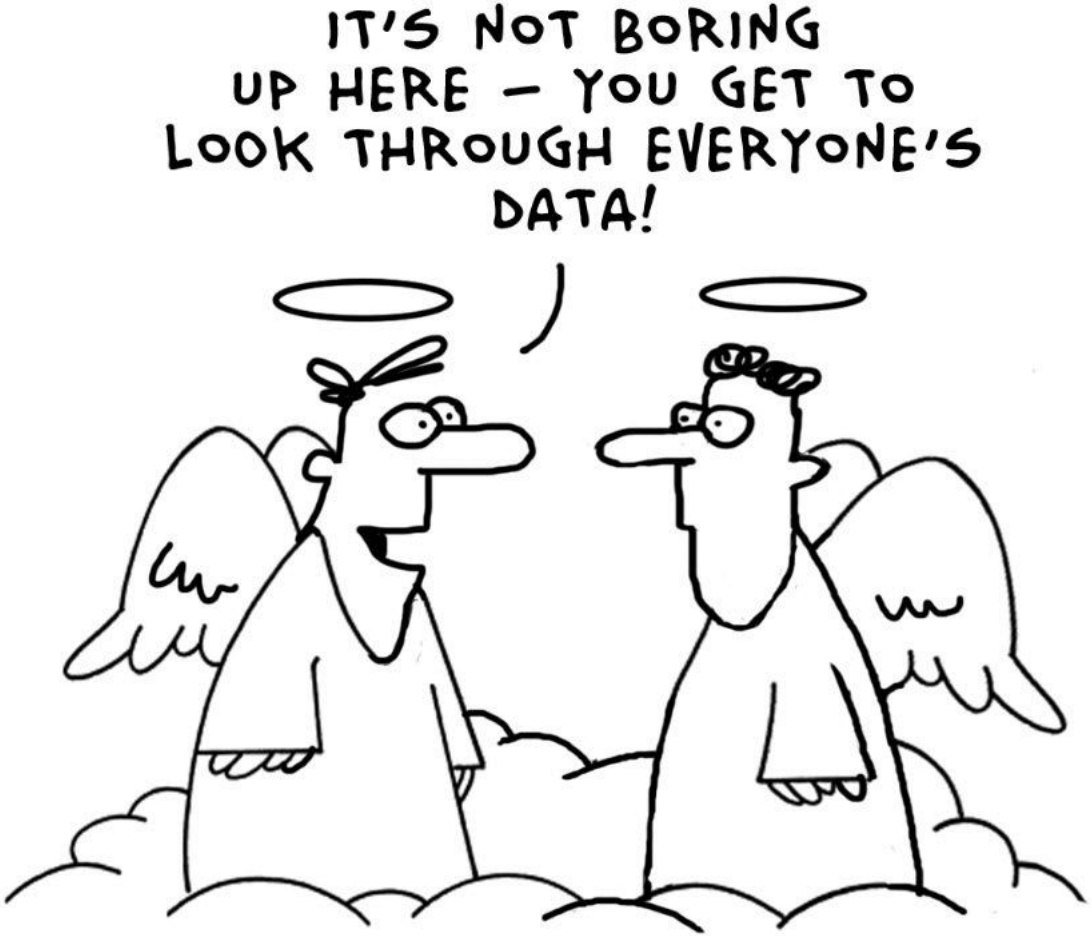


- Specialized data centers built for one big app
  - Social networking: Facebook
  - Web Search: Google, Bing
- "Cloud" data centers
  - Amazon EC2, Windows Azure
  - Google App Engine

Data Centers with 100,000+ Servers

Google’s Oregon Datacenter

What is inside a Datacenter?

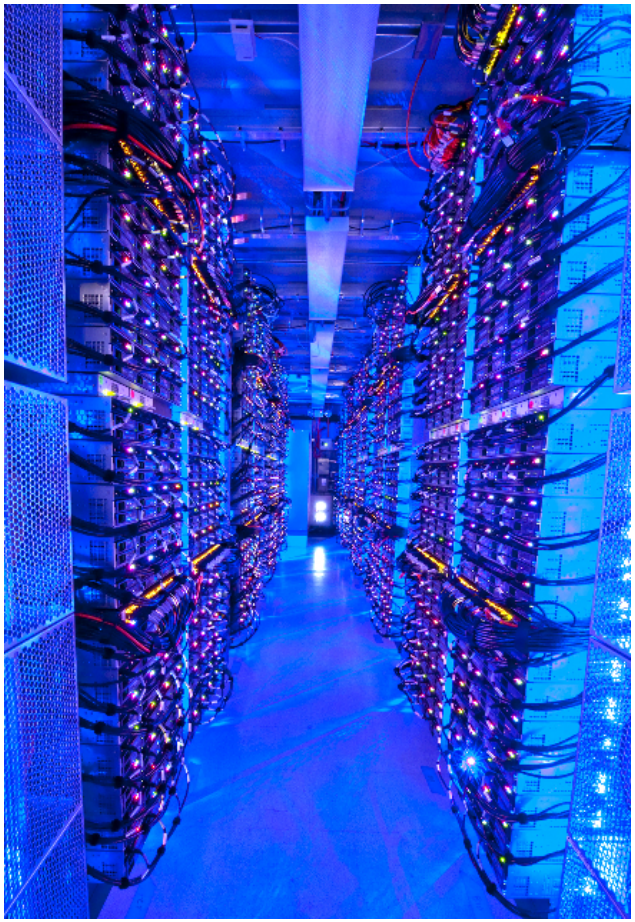

IT'S NOT BORING UP HERE – YOU GET TO LOOK THROUGH EVERYONE'S DATA!

SCALE!

Containers in Datacenter

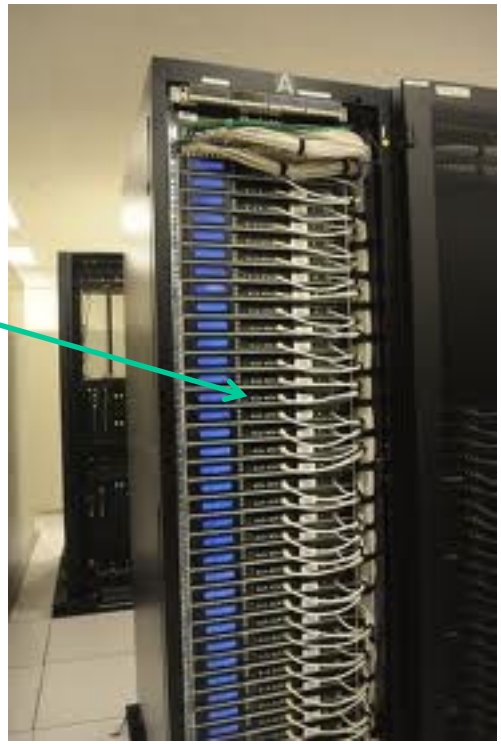
Inside WSC

Inside Container



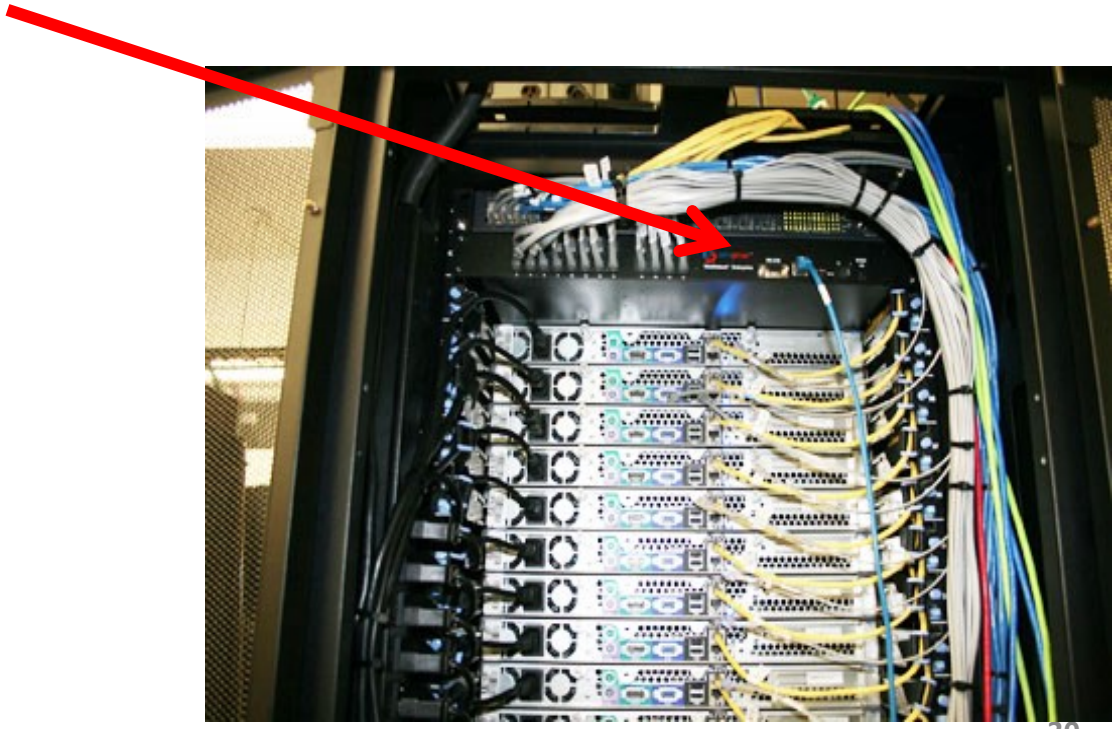
What goes into a datacenter ?

- Servers organized in racks
- Homogeneous – all servers/processors are of the same type

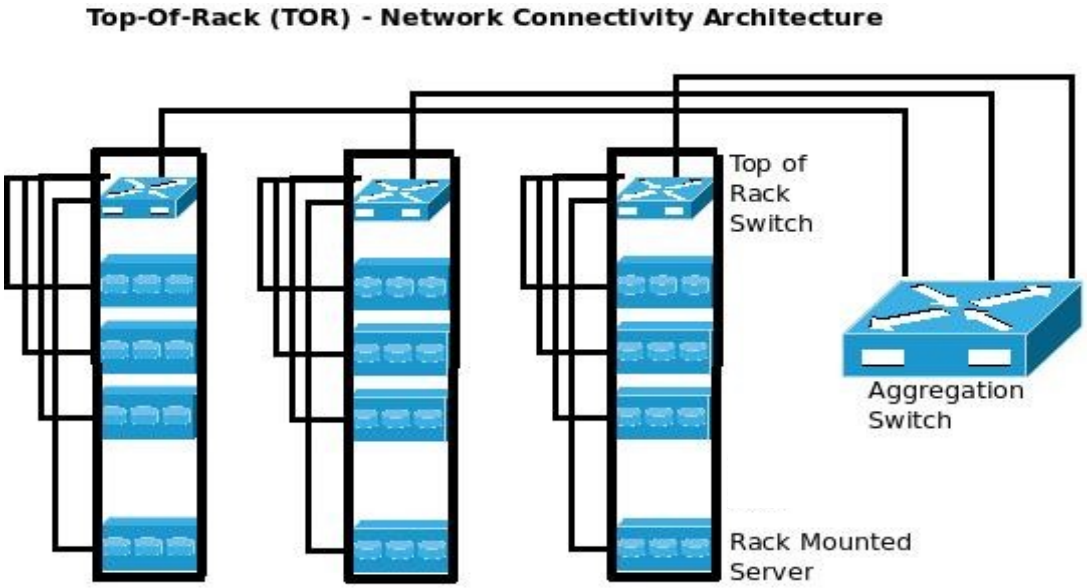


What goes into a datacenter ?

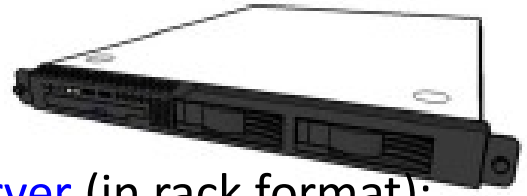
- Each rack has a `Top of Rack' (ToR) switch



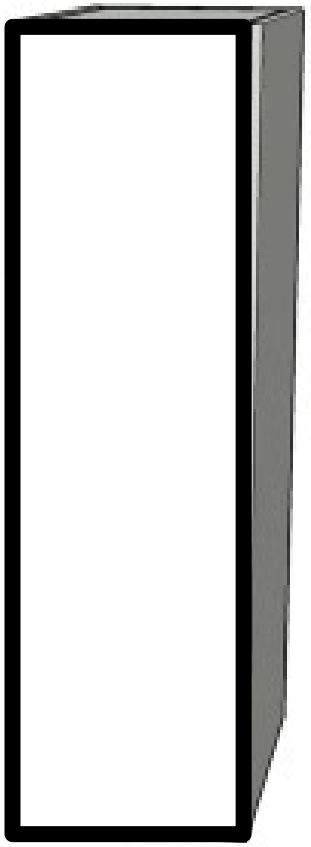
What goes into a datacenter ?

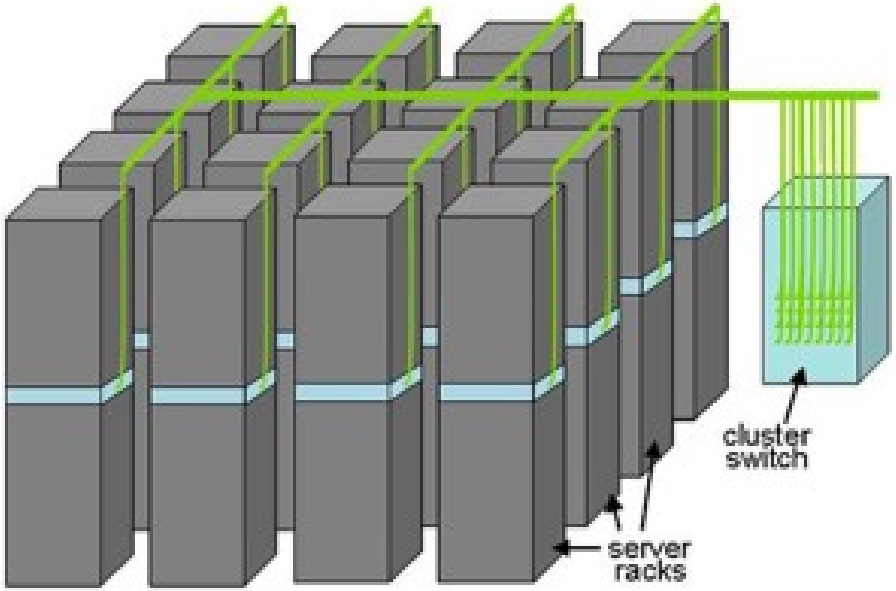


Equipment Inside a Datacenter



Server (in rack format):
1 ¾ inches high "1U",
x 19 inches x 16-20 inches:
8 cores, 16 GB DRAM, 4x1 TB disk





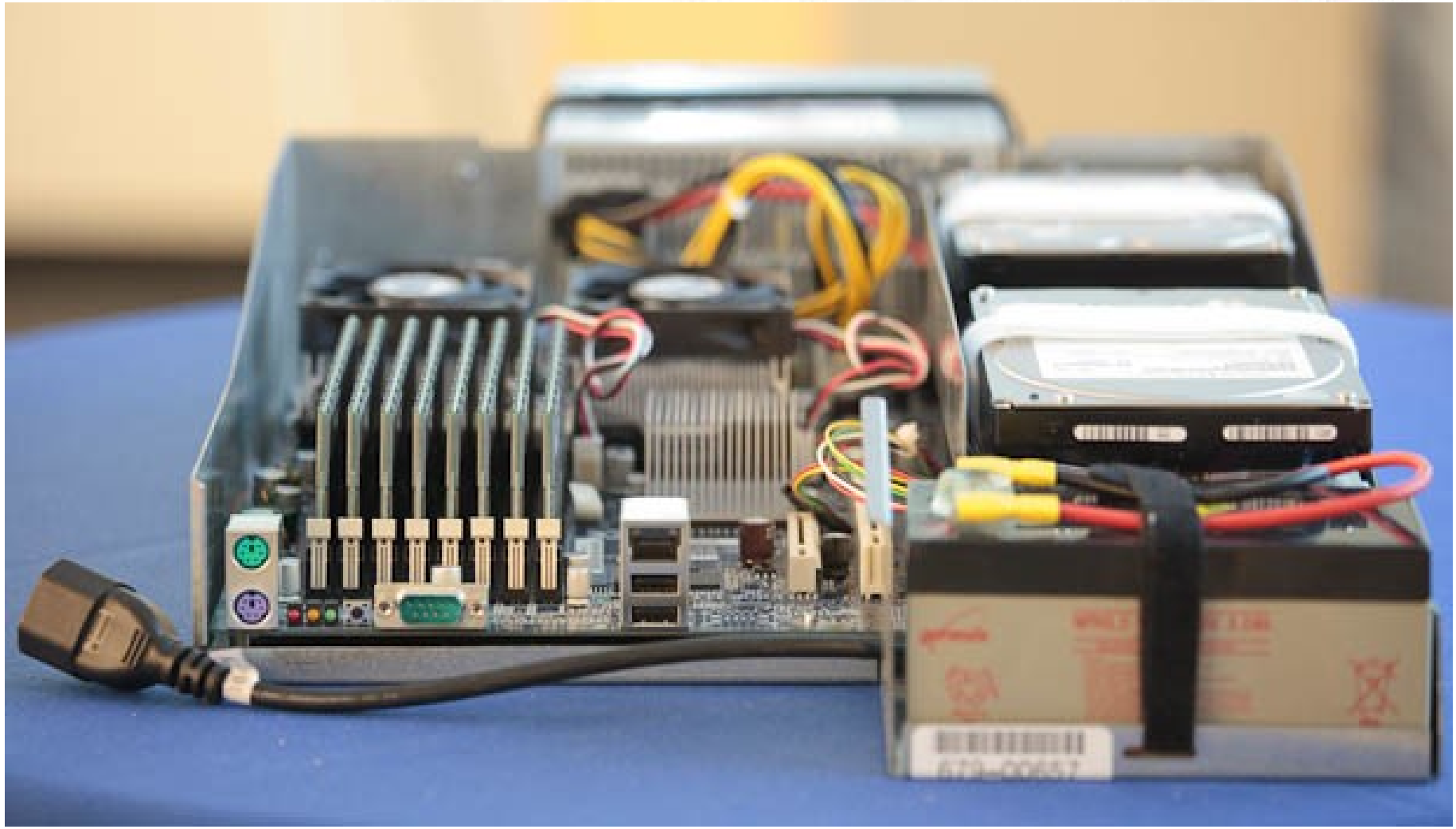
Array (aka cluster):
16-32 server racks + larger local area network switch

7 foot Rack: 40-80 servers + Ethernet local area network (1-10 Gbps) switch in middle ("rack switch")

Server, Rack, Array

Ref: CS 61C, Prof Asonovic, Univ of California, Berkeley

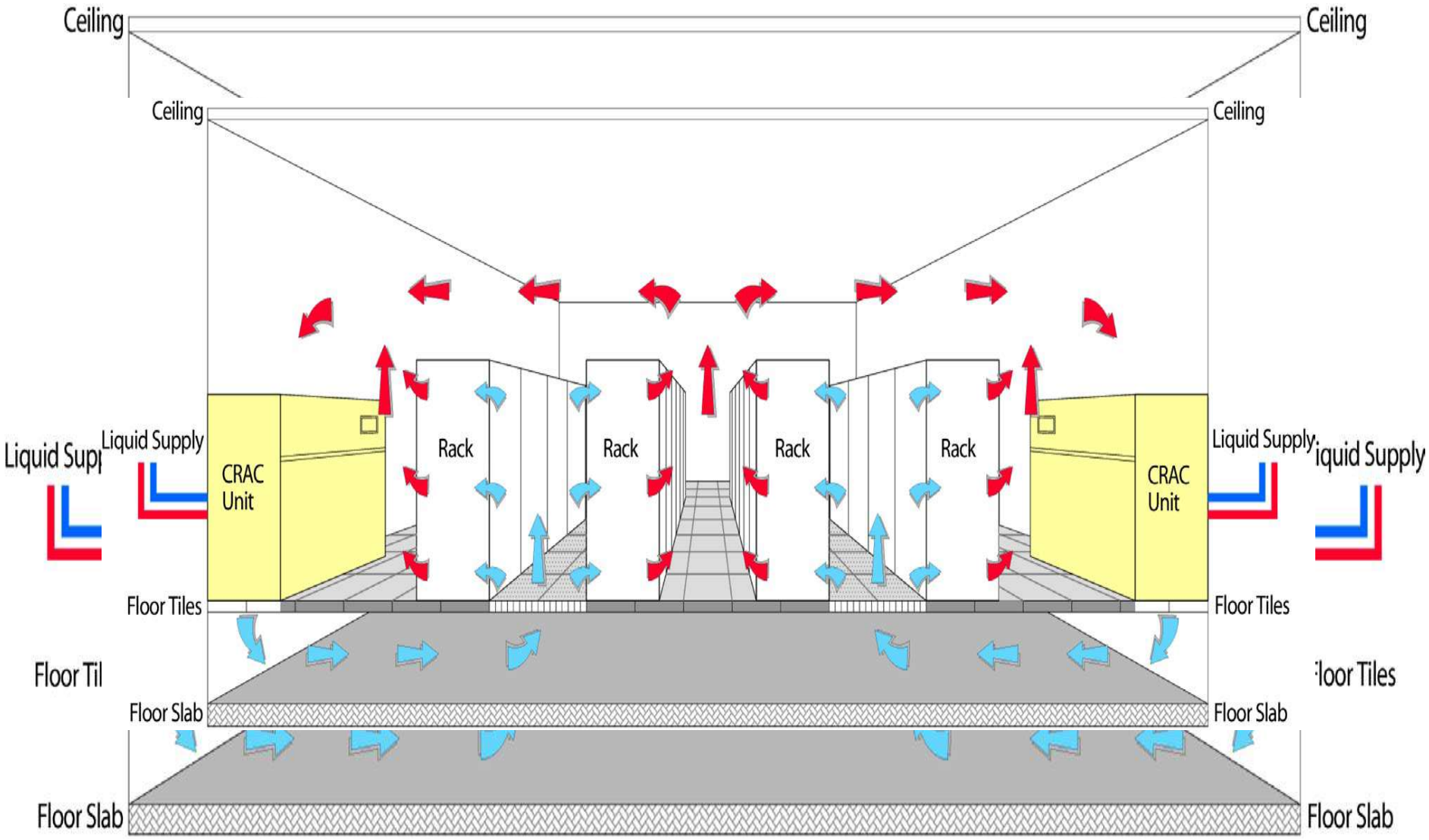
Google Server Internals



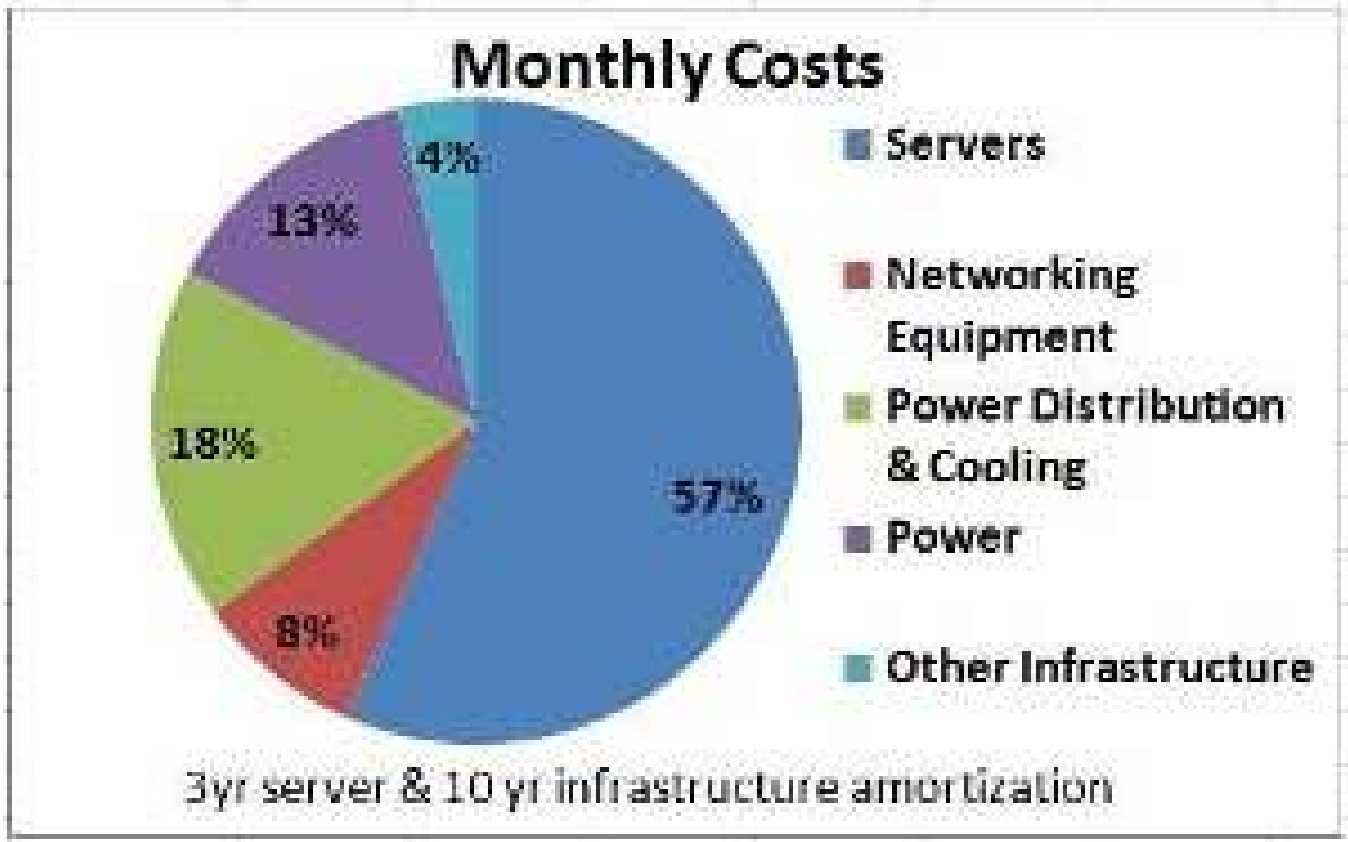
Ref: CS 61C, Prof Asonovic, Univ of California, Berkeley

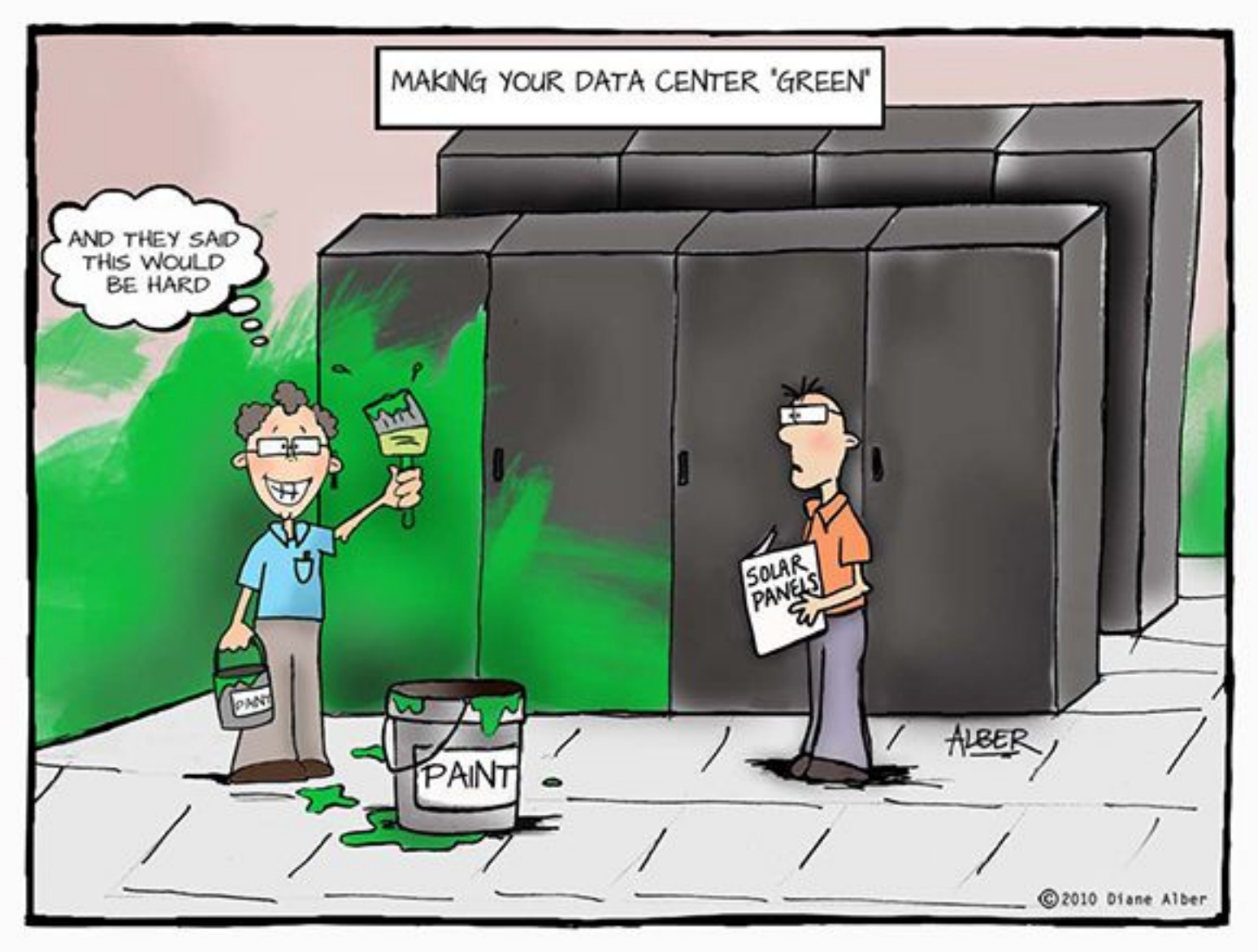
Warehouse-Scale Data Center: Cooling System

CRAC: Computer Room Air Conditioner



Datacenter Monthly Cost (amortized)




MAKING YOUR DATA CENTER 'GREEN'
AND THEY SAID THIS WOULD BE HARD
PAINT
PAINT
SOLAR PANELS
ALBER
©2010 Diane Alber

Requirements of Datacenters/Cloud

Datacenter Requirements

- Support for parallelism:
  - Support for *Data-Level Parallelism*
- High Availability and fault tolerance
  - Must cope with failures gracefully
  - *High failure rate impact Reliability Availability*
  - Resiliency: isolate failure of servers and storage
- Dynamic Management
  - Workload movement: move work to other locations
  - Scale up and down gracefully in response to varying demand
  - *Elasticity*
- Multi-tenancy
  - Multiple independent users share infrastructure
  - Security and resource isolation
  - SLAs on performance and reliability (sometimes)

Ref: CS 61C, Prof Asonovic, Univ of California, Berkeley

Cloud Distinguished by...

- Shared platform with illusion of isolation
  - Collocation with other tenants
  - Exploits technology of VMs and hypervisors
  - At best "fair" allocation of resources, but not true isolation
- Attraction of low-cost cycles
  - Economies of scale driving move to consolidation
  - Statistical multipl*exing to achieve high utilization/efficiency of resources*
- Elastic service
  - Pay for what you need, get more when you need it
  - Limited performance guarantees

Ref: CS 61C, Prof Asonovic, Univ of California, Berkeley

Datacenter Network Requirements

- Massive bisection bandwidth
- Ultra-Low latency (<10 microseconds)
  - Rate-control or packet scheduling?
  - Centralized or distributed?
- Congestion free operation / low queuing
- Effective handling of dynamic traffic
- High Burst Tolerance
  - Incast
- Managing resources across network and servers
  - Provisioning
  - Multi-tenant performance isolation
  - App-aware network scheduling (e.g. for big data)

Datacenter Network Requirements (more..)

- Traffic load balancing
- Support for VM migration
- Security in multi-tenant network

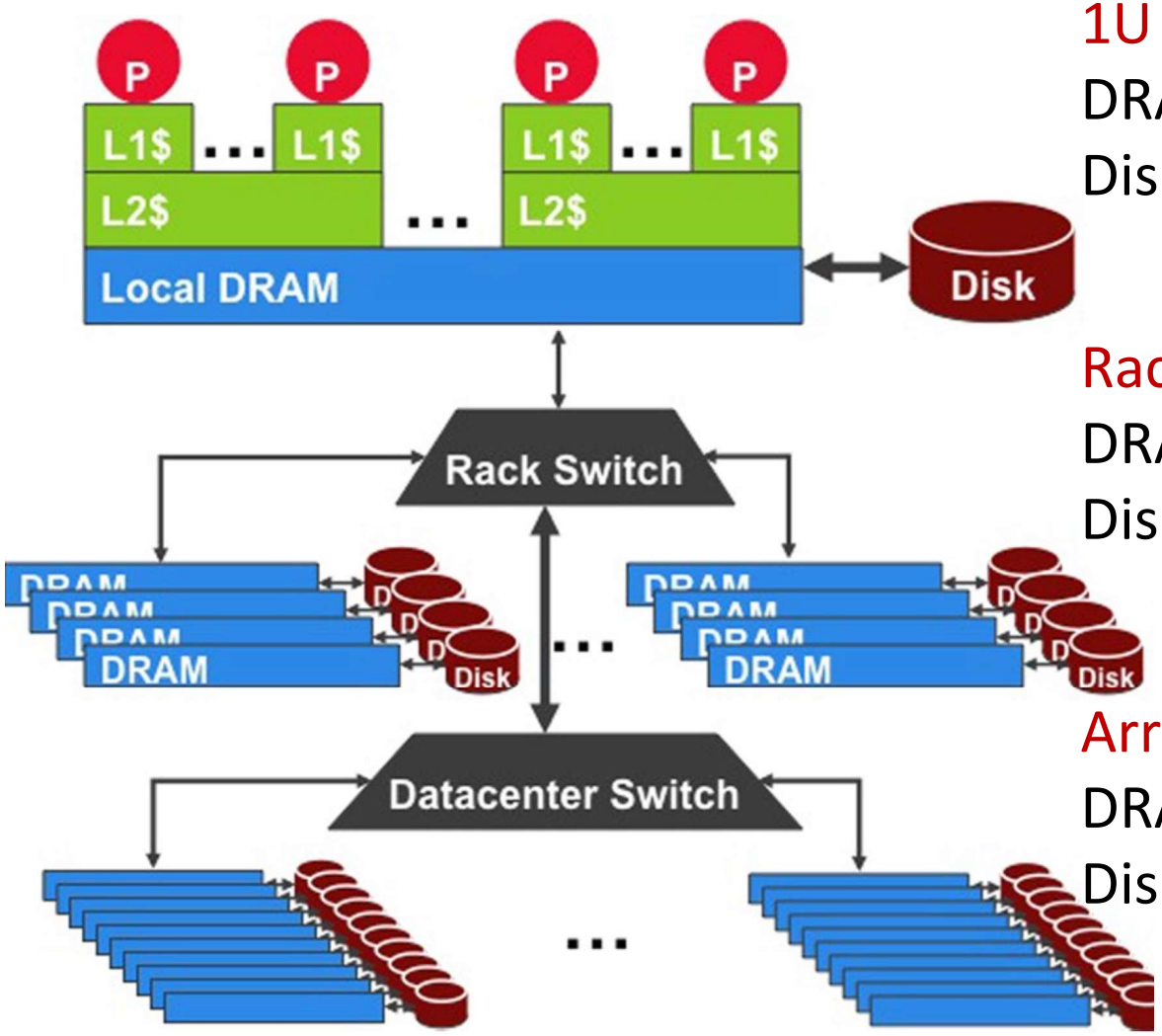
Need scalable solution

- improving efficiency, lowering cost is critical

→ *`scale out' solutions with commodity technologies*

Coping with Performance in Array

Lower latency to DRAM in another server than local disk
Higher bandwidth to local disk than to DRAM in another server



1U Server:

DRAM: 16GB, 100ns, 20GB/s

Disk: 2TB, 10ms, 200MB/s

Rack(80 servers):

DRAM: 1TB, 300us 100MB/s

Disk: 160TB, 11ms, 100MB/s

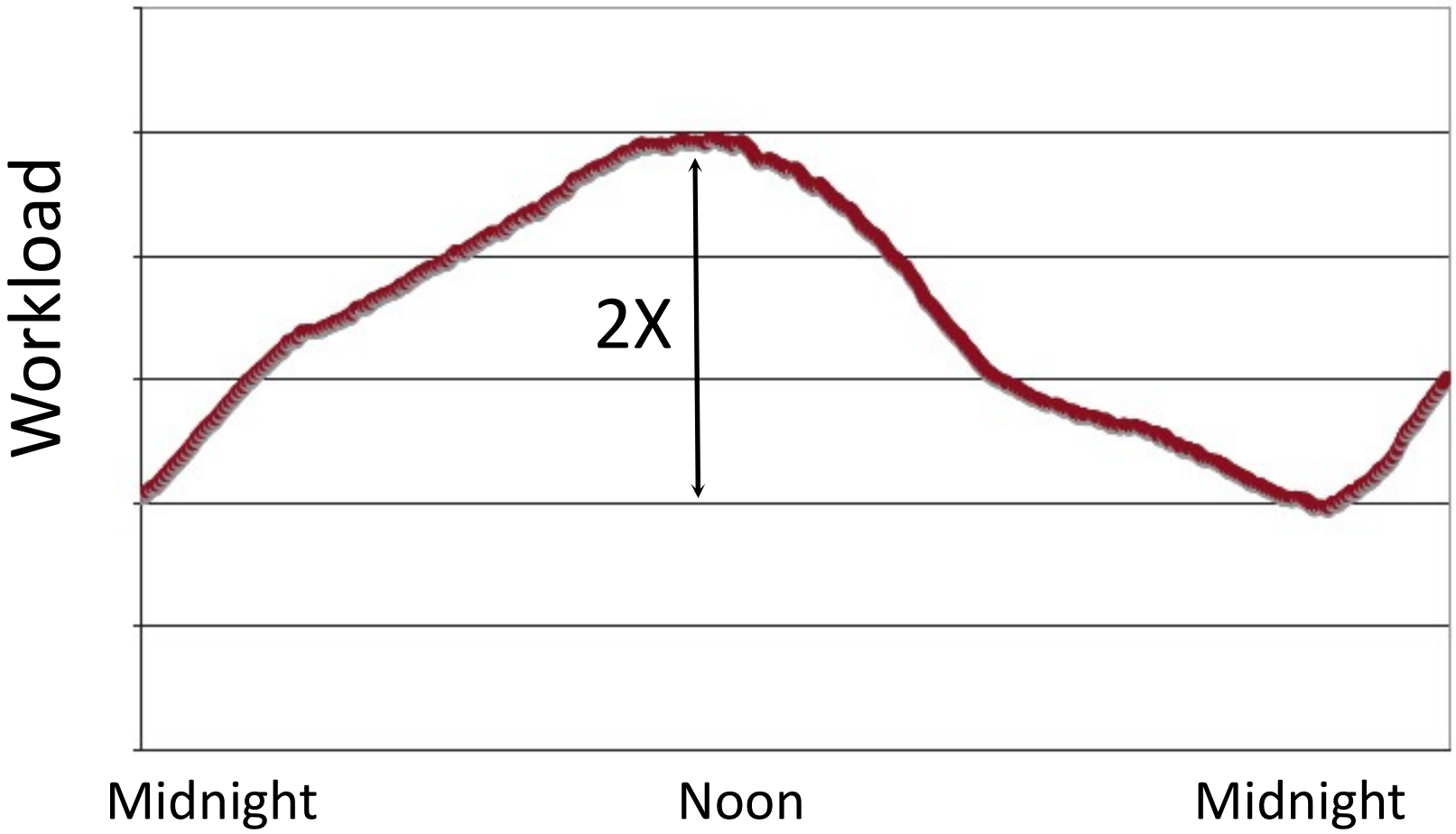
Array(30 racks):

DRAM: 30TB, 500us, 10MB/s

Disk: 4.80PB, 12ms, 10MB/s

Ref: CS 61C, Prof Asonovic, Univ of California, Berkeley

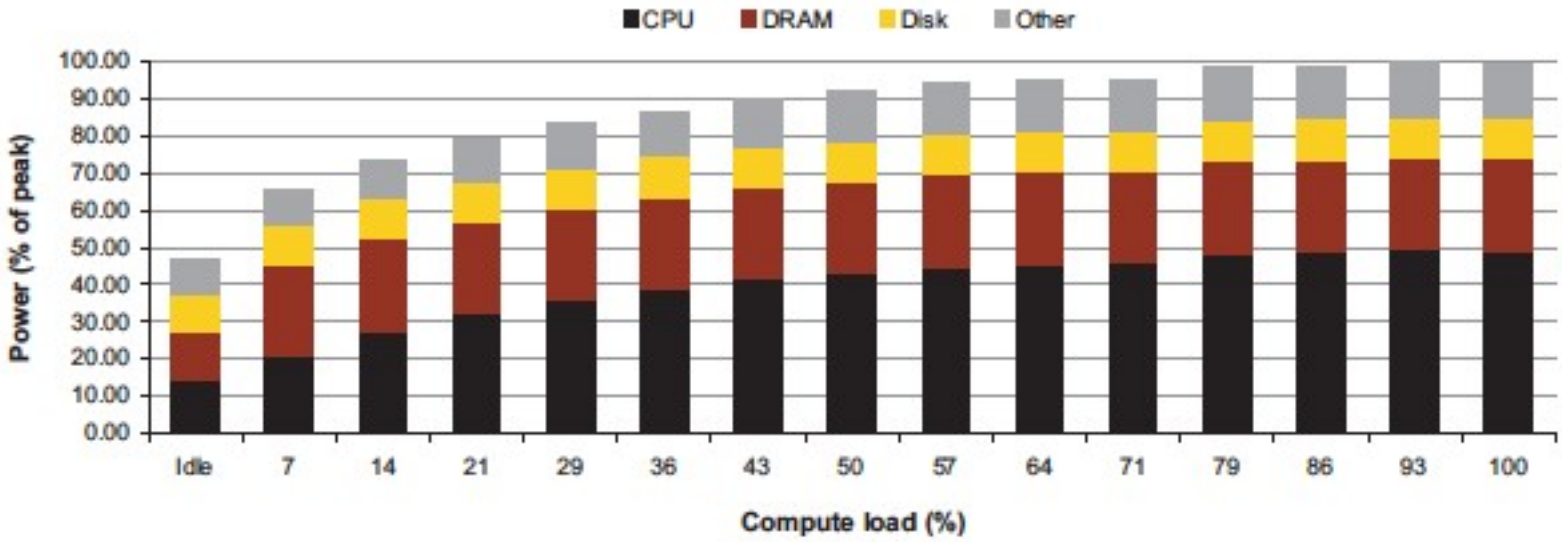
Coping with Workload Variation



Online service: Peak usage 2X off-peak

Ref: CS 61C, Prof Asonovic, Univ of California, Berkeley

Power vs. Server Utilization



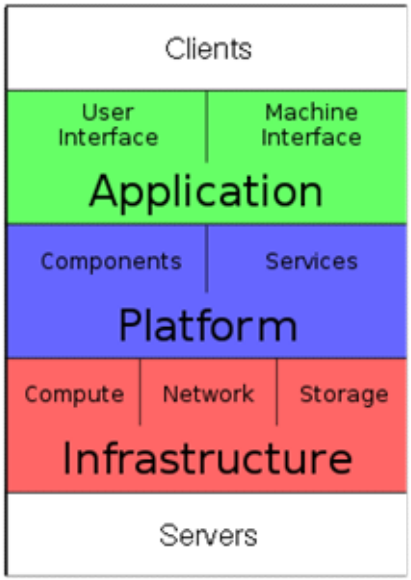
- Server power usage as load varies idle to 100%
- Uses ½ peak power when idle!
- Uses ⅔ peak power when 10% utilized! 90%@ 50%!
- Most servers in WSC utilized 10% to 50%
- Goal should be Energy-Proportionality:
  % peak load = % peak energy

Ref: CS 61C, Prof Asonovic, Univ of California, Berkeley

Cloud Computing Services

Cloud Services

- SaaS: deliver apps over Internet, eliminating need to install/run on customer's computers, simplifying maintenance and support
  - *E.g., Google Docs, Win Apps in the Cloud*
- PaaS: deliver computing "stack" as a service, using cloud infrastructure to implement apps. Deploy apps without cost/complexity of buying and managing underlying layers
  - *E.g., Hadoop on EC2, Apache Spark on GCP*
- IaaS: Rather than purchasing servers, software, data center space or net equipment, clients buy resources as an outsourced service. Billed on utility basis. Amount of resources consumed/cost reflect level of activity
  - *E.g., Amazon Elastic Compute Cloud, Google Compute Platform*



Cloud Computing Stack

Ref: CS 61C, Prof Asonovic, Univ of California, Berkeley

OS for Datacenters

Datacenter/Cloud Computing OS

- If the datacenter/cloud is the new computer
  - What is its **Operating System**?
  - Note that we are not talking about a host OS

- Could be equivalent in benefit as the LAMP stack was to the .com boom – every startup *secretly* implementing the same functionality!

- Open source stack for a Web 2.0 company:
  - **L**inux OS
  - **A**pache web server
  - **M**ySQL, MariaDB or MongoDB DBMS
  - **P**HP, Perl, or Python languages for dynamic web pages

Classical Operating Systems

- Data sharing
  - Inter-Process Communication, RPC, files, pipes, …
- Programming Abstractions
  - Libraries (libc), system calls, …
- Multiplexing of resources
  - Scheduling, virtual memory, file allocation/protection, …

Datacenter/Cloud Operating System (self reading)

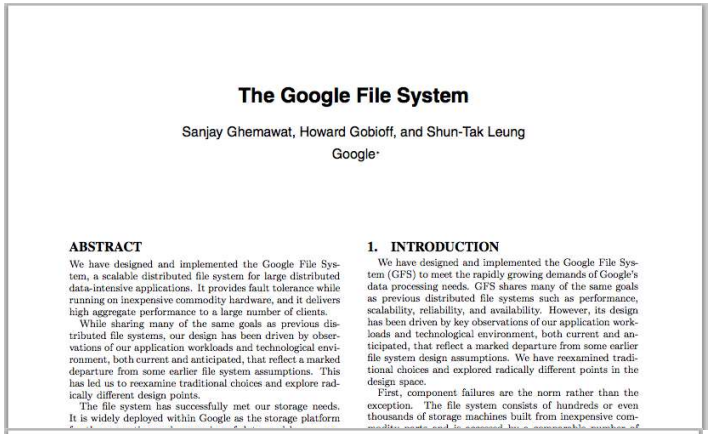
- Data sharing
  - Google File System, key/value stores
  - Apache project: Hadoop Distributed File System
- Programming Abstractions
  - Google MapReduce
  - Apache projects: Hadoop, Pig, Hive, Spark
- Multiplexing of resources
  - Apache projects: Mesos, YARN (MapReduce v2), ZooKeeper, BookKeeper, …

Google Cloud Infrastructure (self reading)

- Google File System (GFS), 2003
  - Distributed File System for entire cluster
  - Single namespace

- Google MapReduce (MR), 2004
  - Runs queries/jobs on data
  - Manages work distribution & fault-tolerance
  - Collocated with file system

- Apache open source versions:
  Hadoop DFS and Hadoop MR



The Google File System

Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung

Google·

ABSTRACT

We have designed and implemented the Google File System, a scalable distributed file system for large distributed data-intensive applications. It provides fault tolerance while running on inexpensive commodity hardware, and it delivers high aggregate performance to a large number of clients.

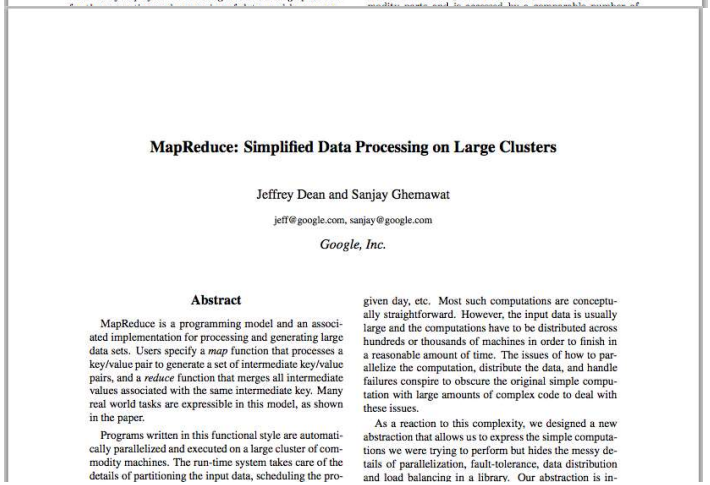
While sharing many of the same goals as previous distributed file systems, our design has been driven by observations of our application workloads and technological environment, both current and anticipated, that reflect a marked departure from some earlier file system assumptions. This has led us to reexamine traditional choices and explore radically different design points.

The file system has successfully met our storage needs. It is widely deployed within Google as the storage platform

1. INTRODUCTION

We have designed and implemented the Google File System (GFS) to meet the rapidly growing demands of Google's data processing needs. GFS shares many of the same goals as previous distributed file systems such as performance, scalability, reliability, and availability. However, its design has been driven by key observations of our application workloads and technological environment, both current and anticipated, that reflect a marked departure from some earlier file system design assumptions. We have reexamined traditional choices and explored radically different points in the design space.

First, component failures are the norm rather than the exception. The file system consists of hundreds or even thousands of storage machines built from inexpensive com-



MapReduce: Simplified Data Processing on Large Clusters

Jeffrey Dean and Sanjay Ghemawat

jeff@google.com, sanjay@google.com

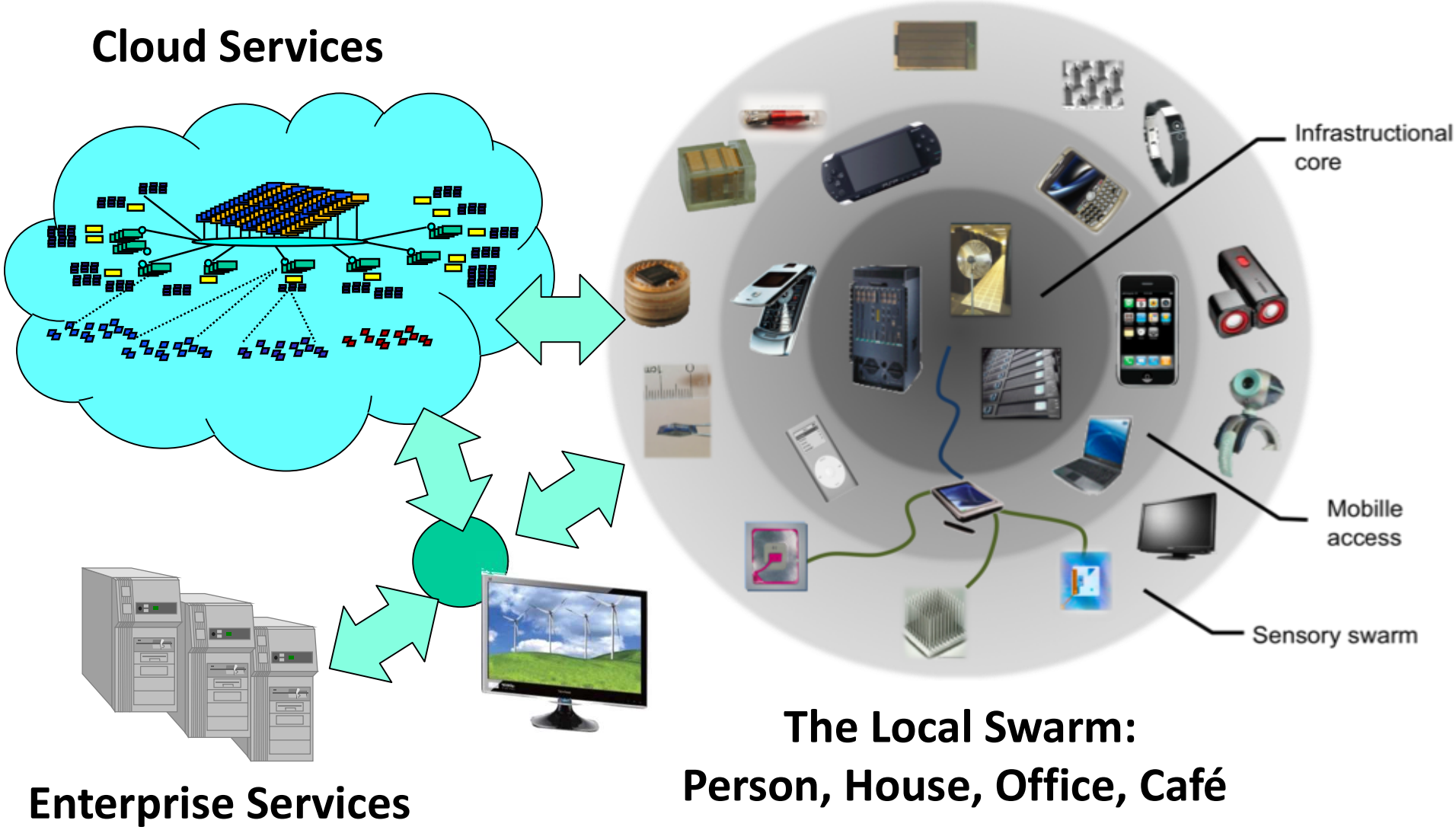
Google, Inc.

Abstract

MapReduce is a programming model and an associated implementation for processing and generating large data sets. Users specify a *map* function that processes a key/value pair to generate a set of intermediate key/value pairs, and a *reduce* function that merges all intermediate values associated with the same intermediate key. Many real world tasks are expressible in this model, as shown in the paper.

Programs written in this functional style are automatically parallelized and executed on a large cluster of commodity machines. The run-time system takes care of the details of partitioning the input data, scheduling the pro-

given day, etc. Most such computations are conceptually straightforward. However, the input data is usually large and the computations have to be distributed across hundreds or thousands of machines in order to finish in a reasonable amount of time. The issues of how to parallelize the computation, distribute the data, and handle failures conspire to obscure the original simple computation with large amounts of complex code to deal with these issues.

As a reaction to this complexity, we designed a new abstraction that allows us to express the simple computations we were trying to perform but hides the messy details of parallelization, fault-tolerance, data distribution and load balancing in a library. Our abstraction is in-

Summary: Cloud and Datacenters