信息科学与技术学院
School of Information Science and Technology

CS 110
Computer Architecture
Warehouse Scale Computing (WSC)

Instructors: Siting Liu & Yuan Xiao
Course website: https://faculty.sist.shanghaitech.edu.cn/liust/courses/CS110.html
School of Information Science and Technology (SIST)
ShanghaiTech University

2026/6/11

Administratives

- Final exam, June 25th 8am-10am; you can bring **3**-page A4-sized double-sided cheat sheet, **handwritten** only! (Teaching center 102/202); the whole course will be covered. No electronic devices (no smart watches, no calculators, no phones, etc.)
- All the labs have been released! ✌️
- Project 3 ddl today!
- Project 4 released, ddl June 18th. **Will be checked during lab sessions June 18th/22nd/23rd. Remember to return the board!** ✌️
- HW 7 ddl today!
- Discussion June 12th on *Final Review*.

Parallelism Overview

Today's Lecture

Software | *Hardware*

- Parallel Requests
  Assigned to computer
  e.g., Search "CS110"
- Parallel Threads
  Assigned to core
  e.g., Lookup, Ads
- Parallel Instructions
  >1 instruction @ one time
  e.g., 5 pipelined instructions
- Parallel Data
  >1 data item @ one time
  e.g., Add of 4 pairs of words
- Hardware descriptions
  All gates @ one time
- Programming Languages

Harness Parallelism & Achieve High Performance

Warehouse Scale Computer

Smart Phone

Google's WSC

Containers in WSCs

Inside WSC

Inside container

Array, Rack, Server

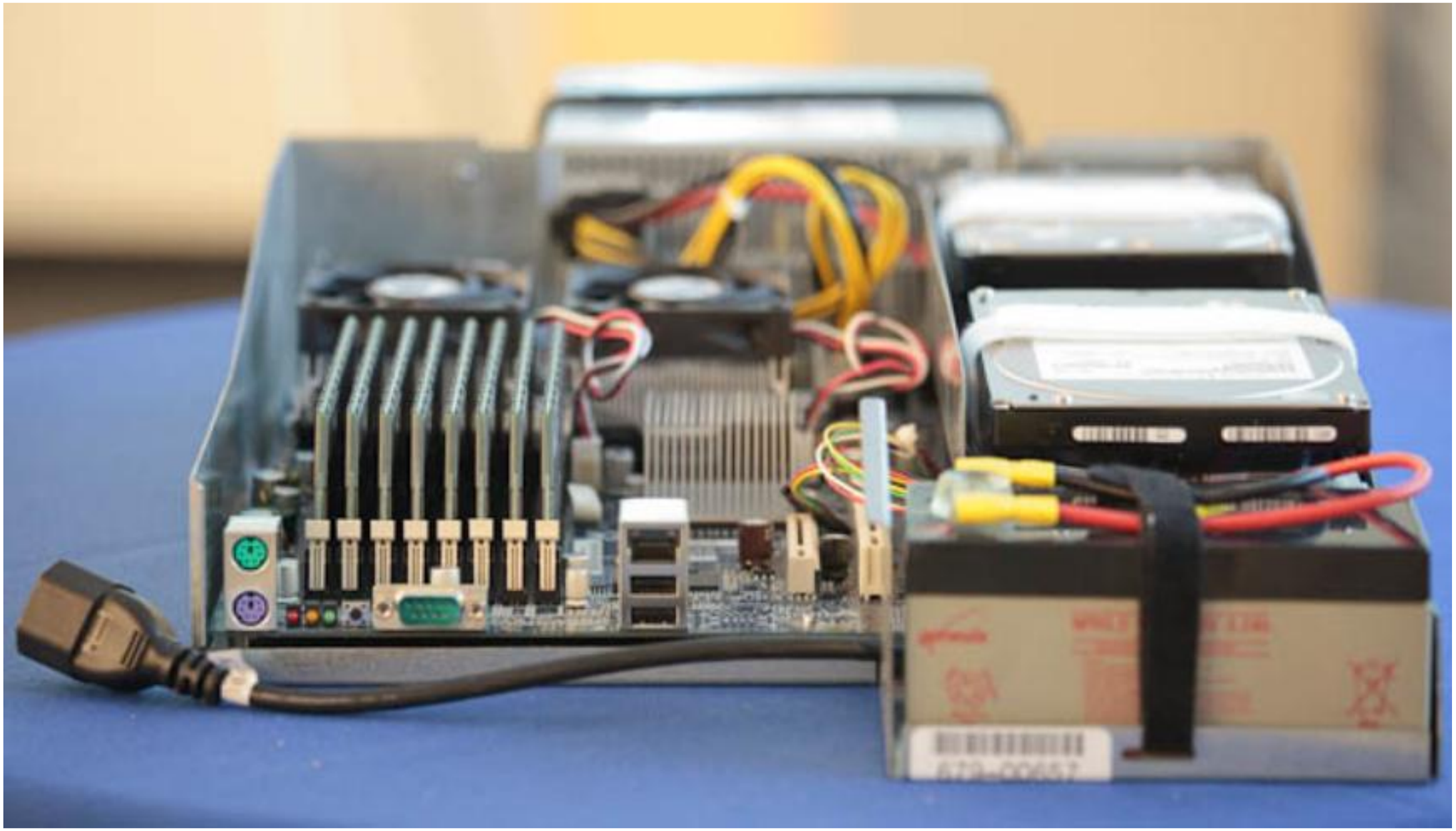
A Giant Computer

- Sunway TaihuLight

系统峰值性能	125.436PFlops
实测持续运算性能	93.015PFlops
处理器型号	"申威26010" 众核处理器
整机处理器个数	40960个
实整机处理器核数	10649600个
系统总内存	1310720 GB
操作系统	Raise Linux
编程语言	C、C++、Fortran
并行语言及环境	MPI、OpenMP、OpenACC等
SSD存储	230TB
在线存储	10PB，带宽288GB/s
近线存储	10PB，带宽32GB/s

http://www.nsccwx.cn/swsource/5d2fe23624364f0351459262

Google Server Internals



Open Compute Project

- Share designs of data center products
  - Facebook, Intel, Nokia, Google, Apple, Microsoft, Seagate Technology, Dell, Cisco, Goldman Sachs, Lenovo, …
- Design and enable the delivery of the most efficient server, storage and data center hardware designs for scalable computing.
- Openly sharing ideas, specifications and other intellectual property is the key to maximizing innovation and reducing operational complexity
- All Facebook Data Centers are 100% OCP

Warehouse-Scale Computers

- Datacenter
  - Collection of 10,000 to 100,000 servers
  - Networks connecting them together
- **Single gigantic** machine
- Very large applications (Internet service): search, email, video sharing, social networking
- Very high availability
- "…WSCs are no less worthy of the expertise of computer systems architects than any other class of machines" Barroso and Hoelzle, 2009

Unique to WSCs

- Ample Parallelism
  - Request-level Parallelism: e.g., web search
  - Data-level Parallelism: e.g., LLM training
- Scale and its Opportunities/Problems
  - Scale of economy: low per-unit cost
  - Cloud computing: rent computing power with low costs (e.g., AWS)
  - High # of failures
    - e.g.: 4 disks/server, annual failure rate: 4% WSC of 50,000 servers: 1 disk fail/hour

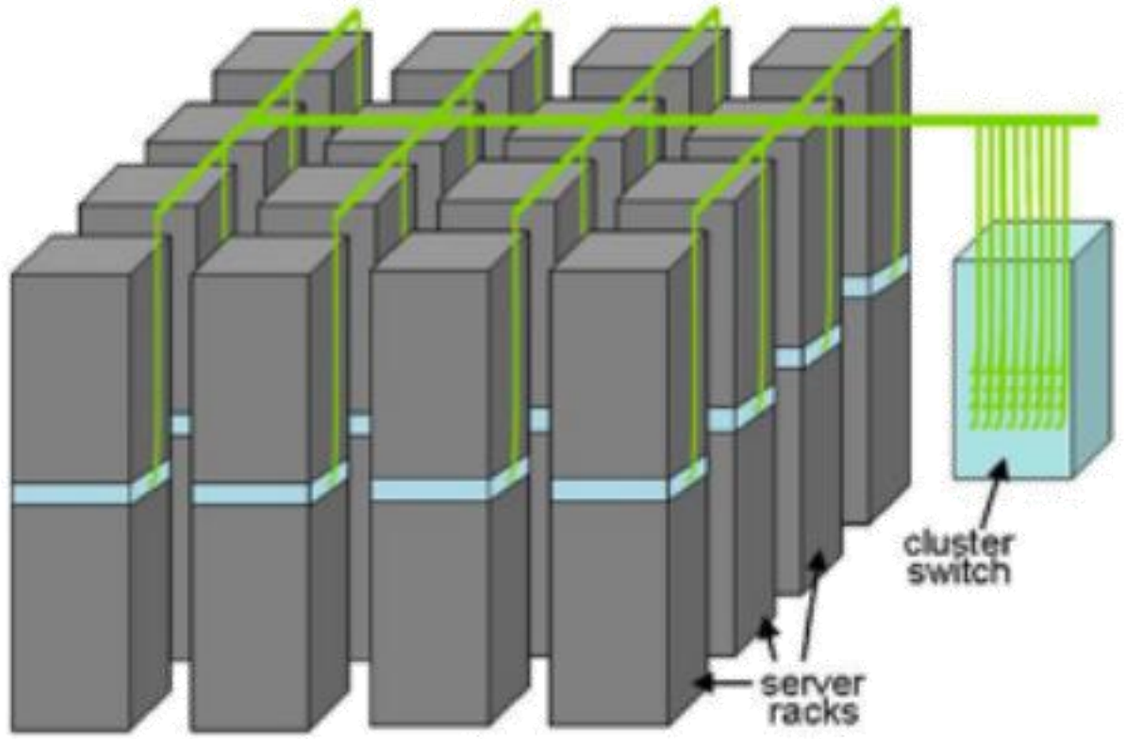
$$\frac{50000 \times 4 \times 4\%}{365 \times 24} \approx 0.913$$

- Operation Cost Count
  - Longer life time (>10 years)
  - **Cost of equipment purchases << cost of ownership**

WSC Architecture

1U Server:
8 cores,
16 GB DRAM,
4x1 TB disk

Rack:
40-80 severs,
Local Ethernet (1-10Gbps) switch



Array (aka cluster):
16-32 racks
Expensive switch
(10X bandwidth 100x cost)

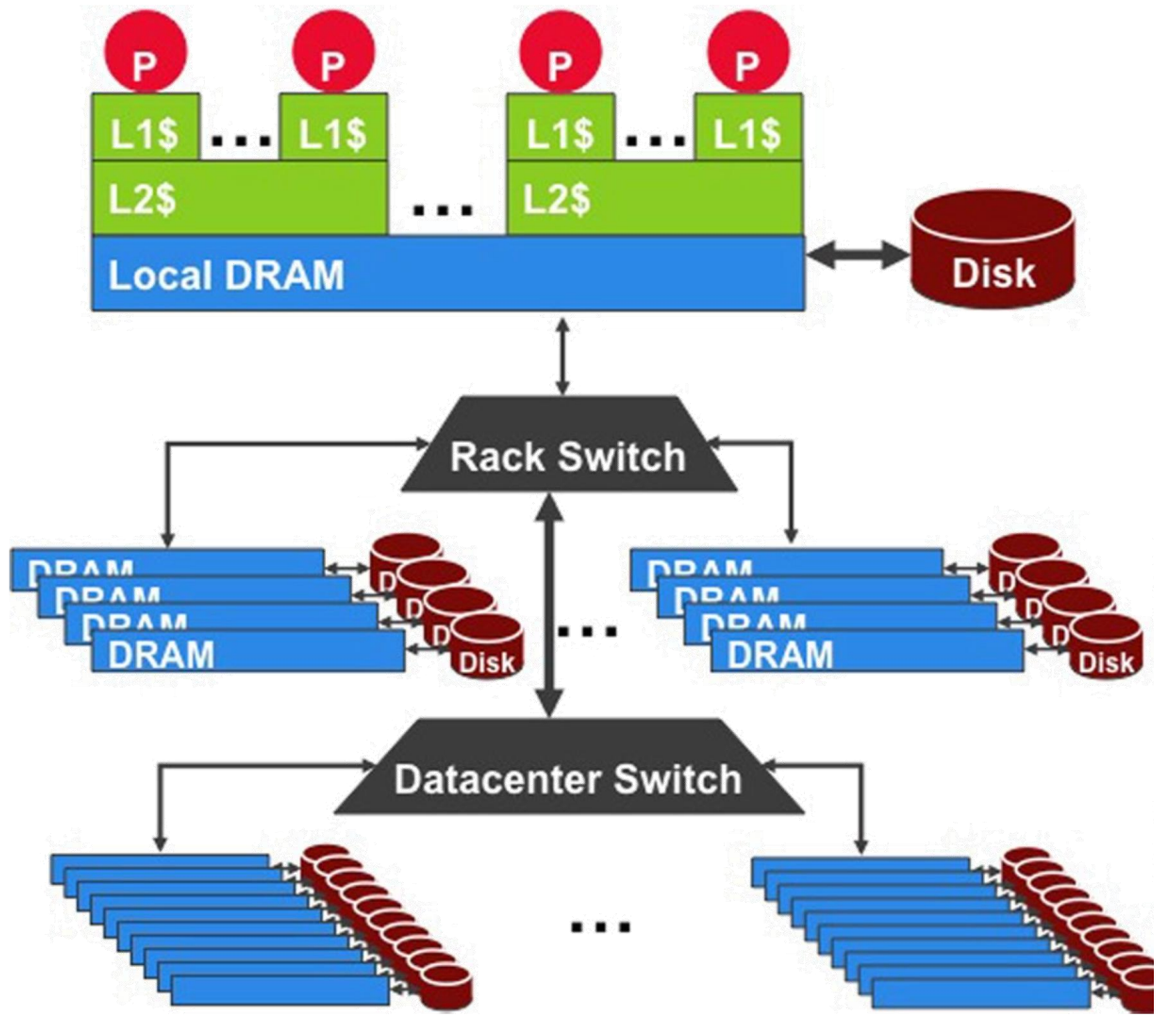
WSC Storage Hierarchy

- 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, 0.1us, 20GB/s
Disk: 2TB, 10^4us, 200MB/s

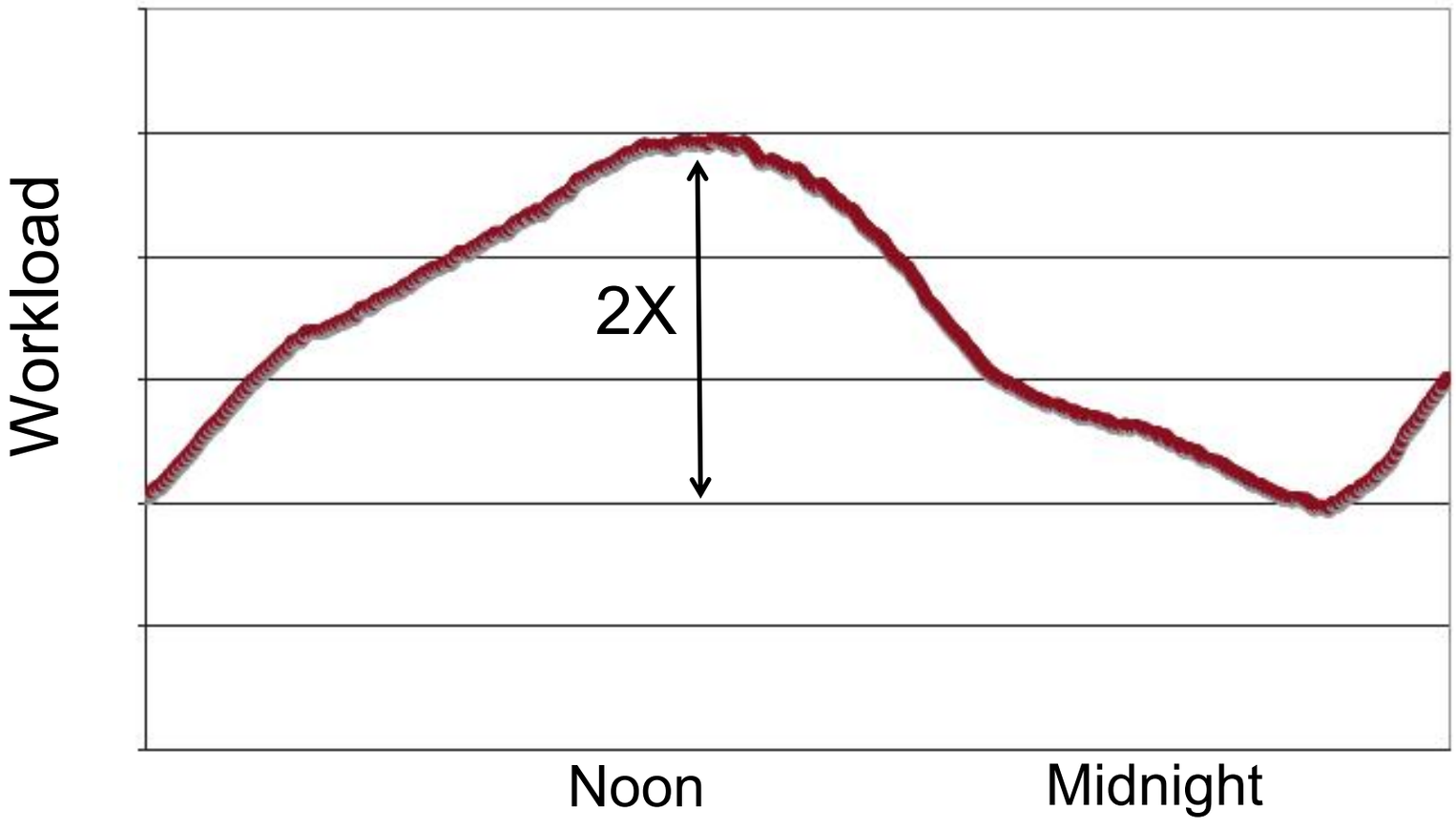
Rack(80 severs):
DRAM: 1TB, 300us, 100MB/s
Disk: 160TB, 11ms, 100MB/s

Array(30 racks):
DRAM: 30TB, 500us, 10MB/s
Disk: 4.80PB, 12ms, 10MB/s



Workload Variation

- Online service: Peak usage 2X off-peak



Impact on WSC software

- Latency, bandwidth → Performance
  - Independent data set within an array
  - Locality of access within server or rack
- High failure rate → Reliability, Availability
  - Preventing failures is expensive
  - Cope with failures gracefully
- Varying workloads → Scalability, Availability
  - Scale up and down gracefully
- More challenging than software for single computers!

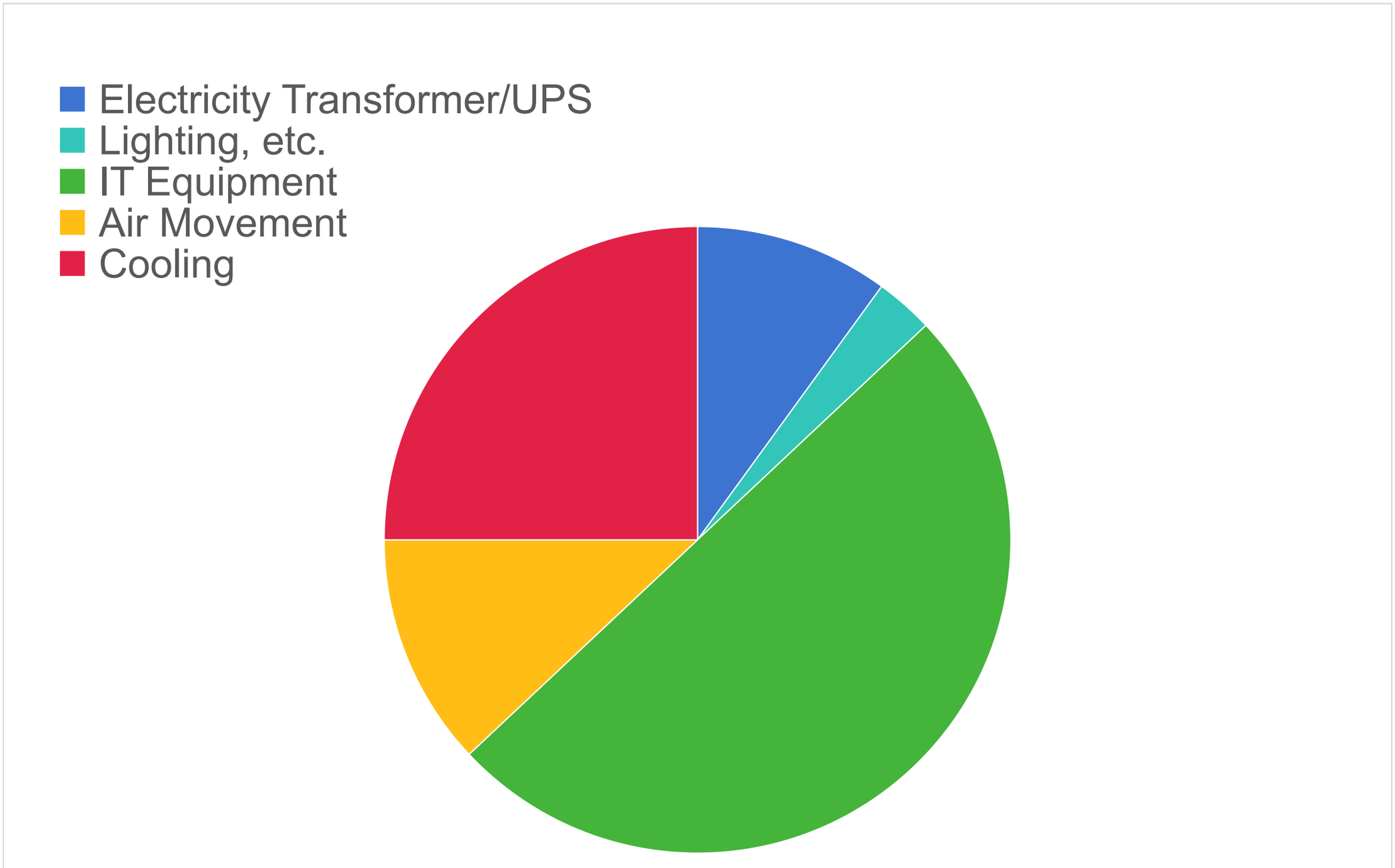
Power Usage Effectiveness

- Energy efficiency
  - Primary concern in the design of WSC
  - Important component of the total cost of ownership
- Power Usage Effectiveness (PUE):

$$\frac{\text{Total Building Power}}{\text{IT Equipment Power}}$$

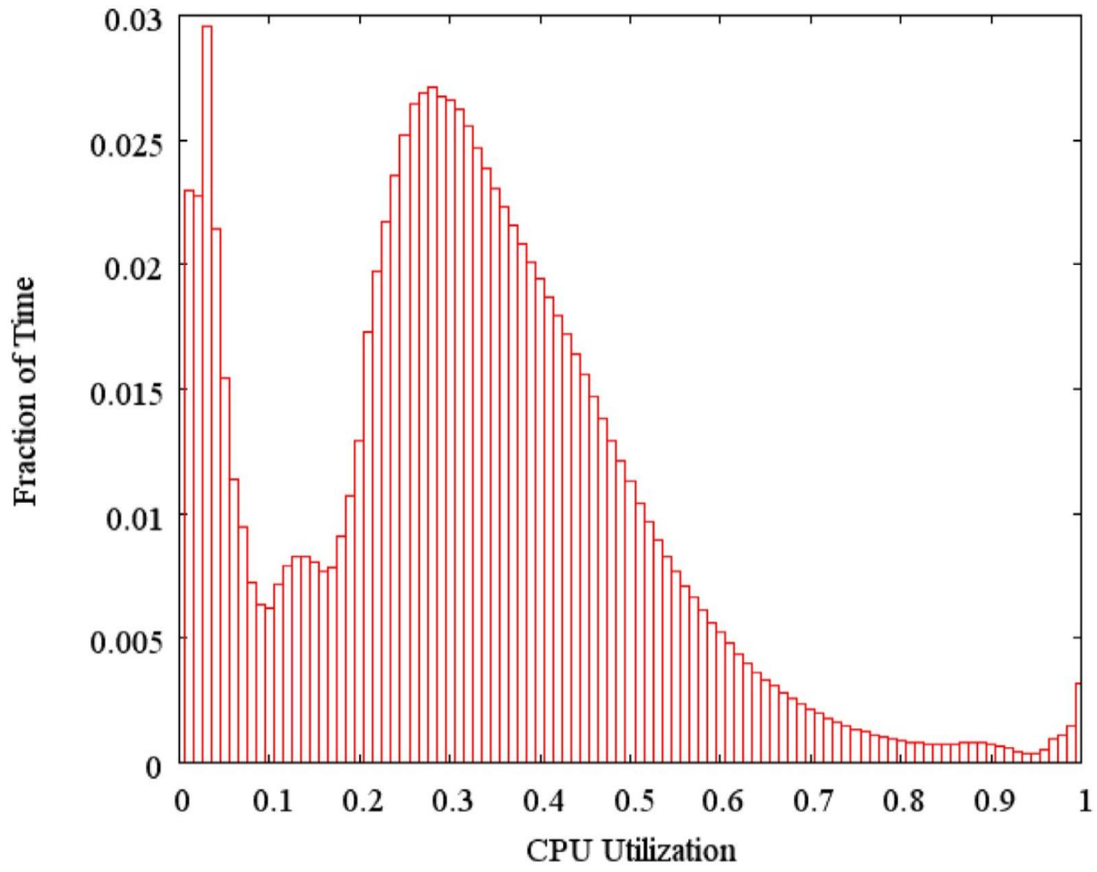
  - A power efficiency measure for WSC
  - Not considering efficiency of servers, networking
  - Perfection = 1.0
  - Google WSC's PUE = 1.2

Where Data Center Power Goes



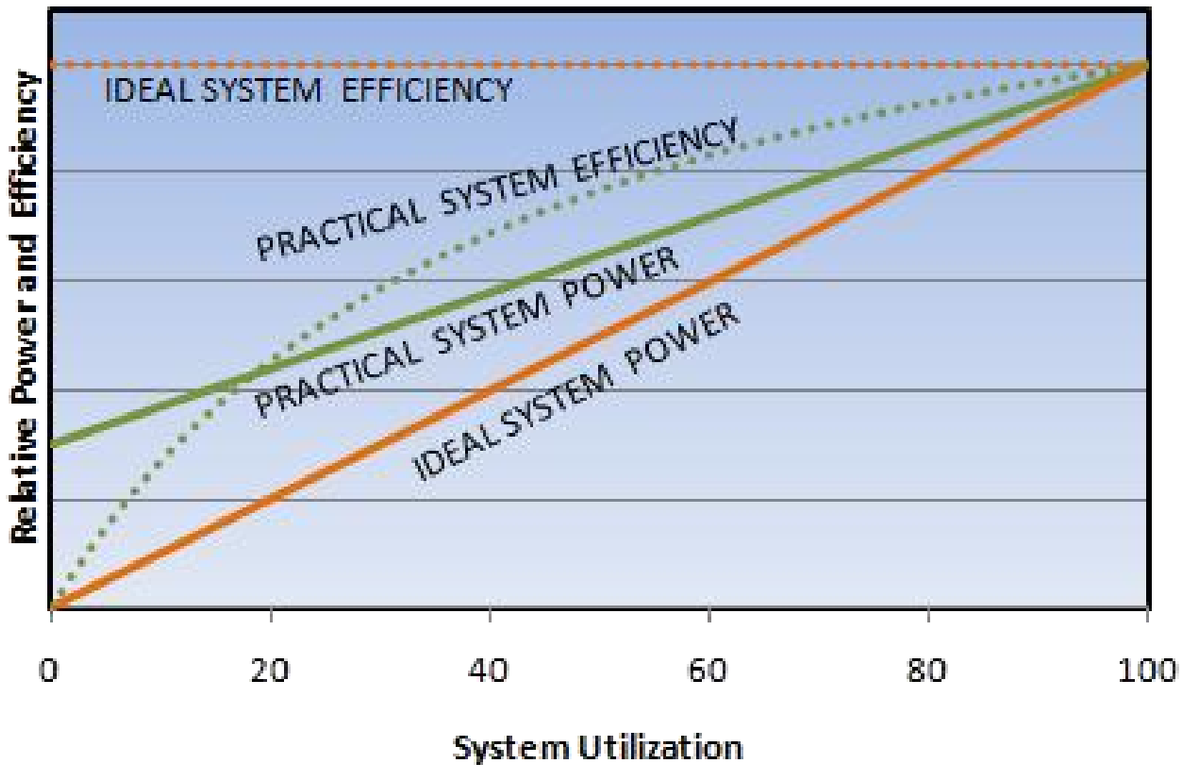
Load Profile of WSCs

- Average CPU utilization of 5,000 Google servers, 6 month period
- Servers rarely idle or fully utilized, operating most of the time at 10% to 50% of their maximum utilization



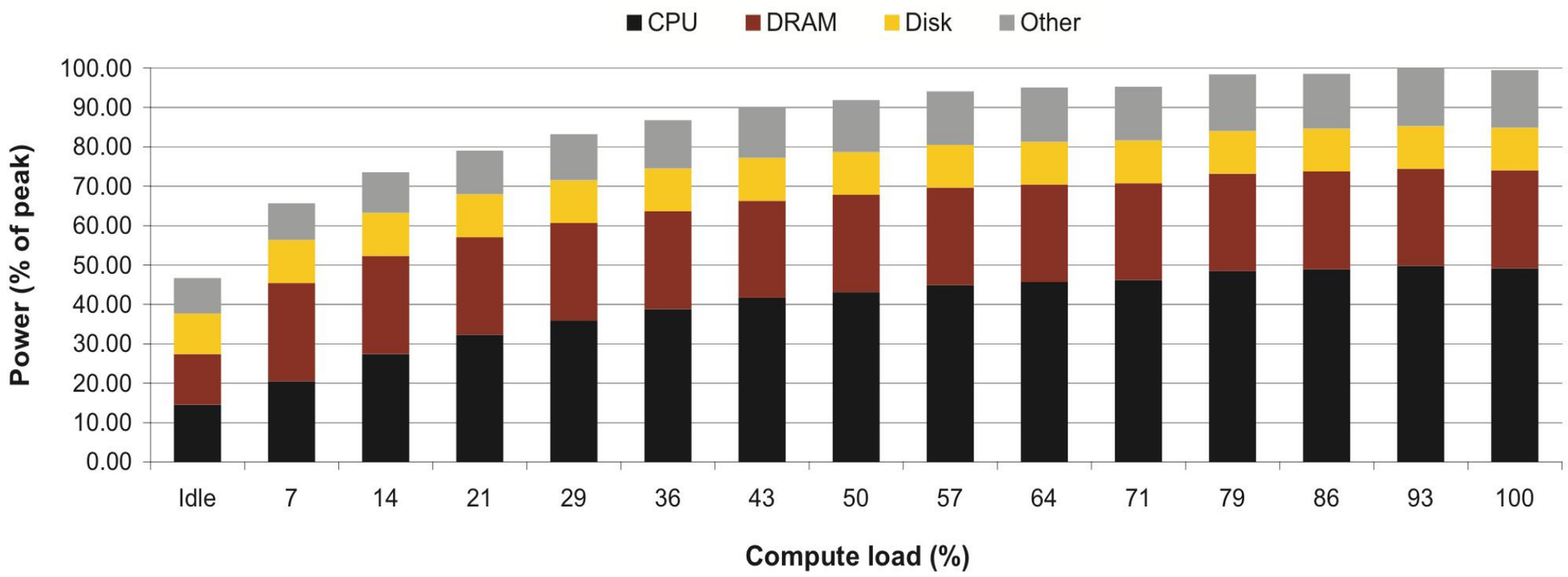
Energy-Proportional Computing: Design Goal of WSC

- Energy = Power x Time
- Efficiency = Computation/Energy
- Desire:
  - Consume almost no power when idle (Doing nothing well)
  - Gradually consume more power as the activity level increases



Cause of Poor Energy Proportionality

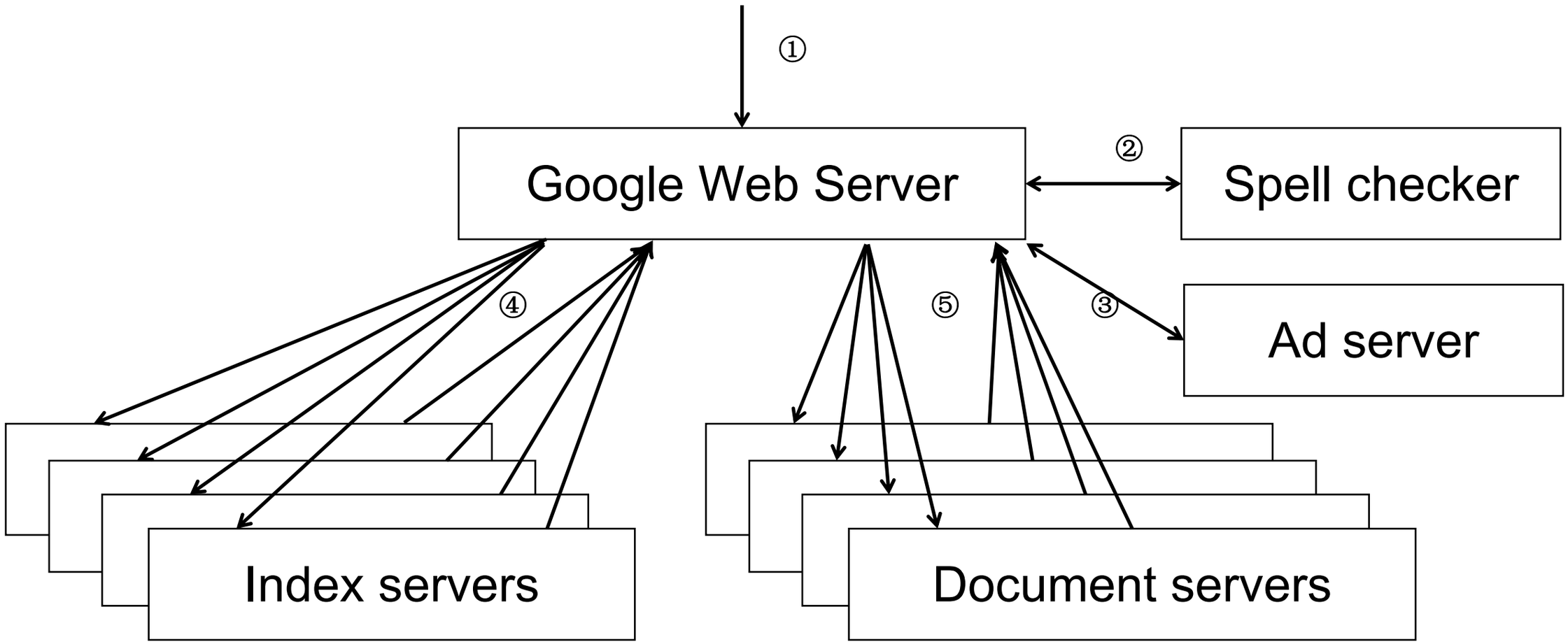
- DRAM, disks, networking: 70% at idle!
- Need to improve the energy efficiency of peripherals



More Parallelism

- Request-Level Parallelism (RLP)
  - Hundreds of thousands of requests per sec.
  - Popular Internet services like web search, social networking, …
  - Such requests are largely independent
    - Often involve read-mostly databases
    - Rarely involve read-write sharing or synchronization across requests
  - Computation easily partitioned across different requests and even within a request

Google Query-Serving Architecture



Anatomy of a Web Search

Na Zha 2

All Images Videos News Short videos Forums Web More Tools

Any time
All results
Advanced Search

About 15,100,000 results (0.33s)

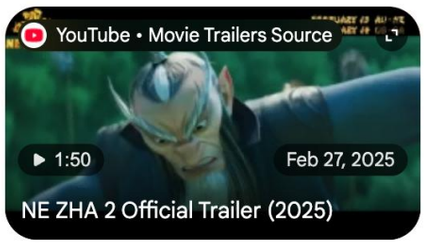
These are results for ***Ne*** Zha 2
Search instead for Na Zha 2

Ne Zha 2

Not Rated 2025 · Fantasy/Animation · 2h 24m

Overview Reviews Cast

3DVF
Ne Zha 2 Confirms the Phenomenon: We Saw the Chinese...
A Chinese phenomenon coming soon to America. Since its release in China, Ne…
3 weeks ago



Ratings
IMDb 8.1/10
Rotten Tomatoes 96%

Wikipedia
https://en.wikipedia.org › wiki › Ne_Zha_2

Ne Zha 2

Ne Zha 2 is a 2025 Chinese animated fantasy action adventure film written and directed by Jiaozi. The direct sequel to Ne Zha (2019), it is based on the ...

People also ask

Is Ne Zha 2 coming out?

Why is Ne Zha 2 so successful?

About

8.1/10	96%	63%
IMDb	Rotten Tomatoes	Metacritic

94% liked this movie
Google users

After the catastrophe, although the souls of Nezha and Aobing were saved, their bodies would soon be shattered. Taiyi Zhenren planned to use the seven-colored lotus to rebuild their bodies.

Release date: February 14, 2025 (USA)

Director: Yu Yang

Anatomy of a Web Search (cont'd)

- Search "Ne Zha 2"
  - Direct request to "closest" WSC;
  - Front-end load balancer directs request to one of many arrays (cluster of servers) within WSC;
  - Within array, select one of many Google Web Servers (e.g. GWS) to handle the request and compose the response pages;
  - GWS communicates with Index Servers to find documents that contains the search word, "Ne Zha 2";
  - Return document list with associated relevance score;

Anatomy of a Web Search (cont'd)

- In parallel, Ad system: run ad auction for bidders on search terms
- Use docids (Document IDs) to access indexed documents
- Compose the page
  - Result document extracts (with keyword in context) ordered by relevance score
  - Sponsored links (along the top) and advertisements (along the sides)

Anatomy of a Web Search (cont'd)

- Implementation strategy
  - Randomly distribute the entries
  - Make many copies of data (a.k.a. "replicas")
  - Load balance requests across replicas
- **Redundant copies** of indices and documents
  - Breaks up search hot spots, e.g., "Ne Zha 2"
  - Increases opportunities for **request-level parallelism**
  - Makes the system more **tolerant of failures**

Data-level Parallelism in WSC

- SIMD
  - Supports data-level parallelism in a single machine
  - Additional instructions & hardware
  - e.g., Matrix multiplication in memory
- DLP on WSC
  - Supports data-level parallelism across multiple machines
  - MapReduce & scalable file systems

Problem Statement

- How to process large amounts of raw data (crawled documents, request logs, …) every day to compute derived data (inverted indices, page popularity, …), when computation is conceptually simple but input data is large and distributed across 100s to 1000s of servers, so as to finish in reasonable time?
- Challenge: Parallelize computation, distribute data, tolerate faults without obscuring simple computation with complex code to deal with issues

Solution: MapReduce

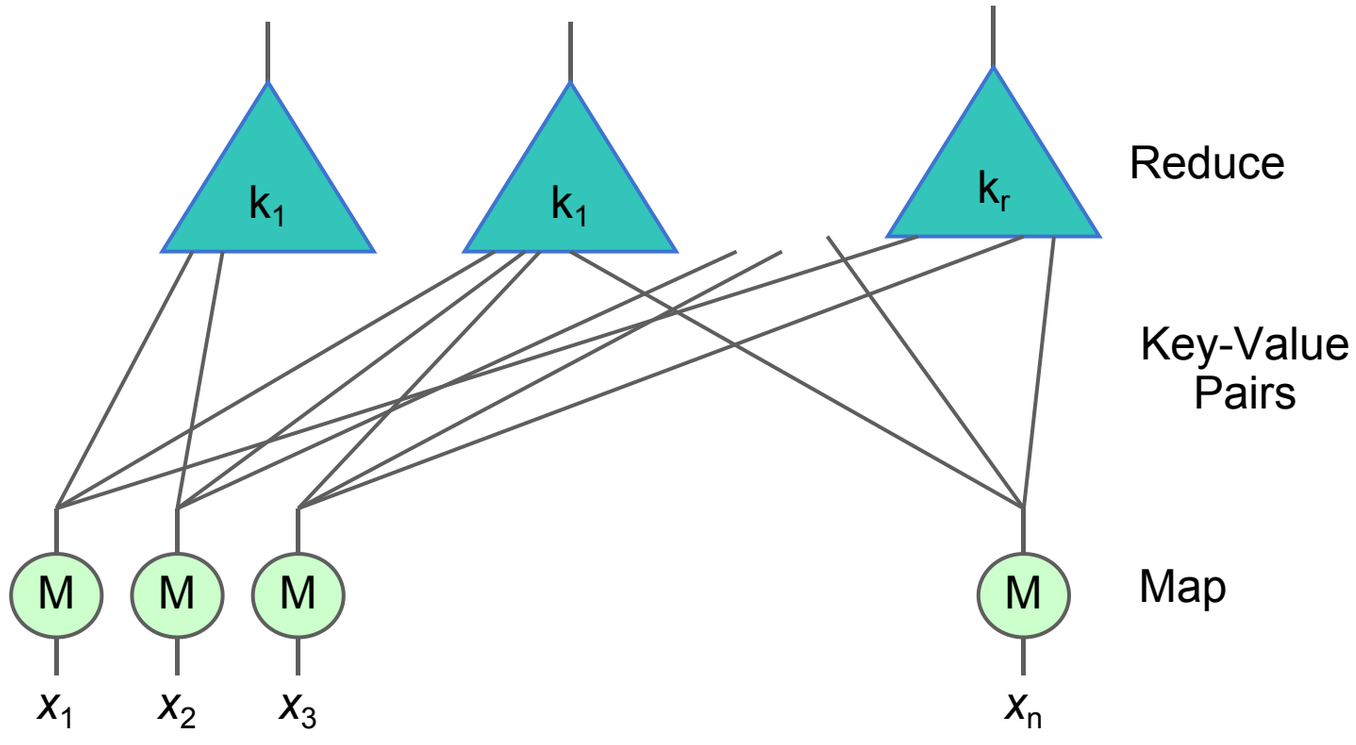
- Simple data-parallel programming model and implementation for processing large datasets
- Users specify the computation in terms of
  - a ***map*** function, and
  - a ***reduce*** function
- Underlying runtime system
  - Automatically parallelize the computation across large scale clusters of machines
  - Handles machine failure
  - Schedule inter-machine communication to make efficient use of the networks

MapReduce: Real Applications

- At Google
  - Index construction for Google Search
  - Article clustering for Google News
  - Statistical machine translation
  - For computing multi-layers street maps
- At Yahoo!
  - “Web map” powering Yahoo! Search
  - Spam detection for Yahoo! Mail
- At Facebook
  - Data mining
  - Ad optimization
  - Spam detection

MapReduce Programming Model

- Map computation across many objects
  - E.g., 10^{10} Internet web pages
- Aggregate results in many different ways
- System deals with issues of resource allocation & reliability



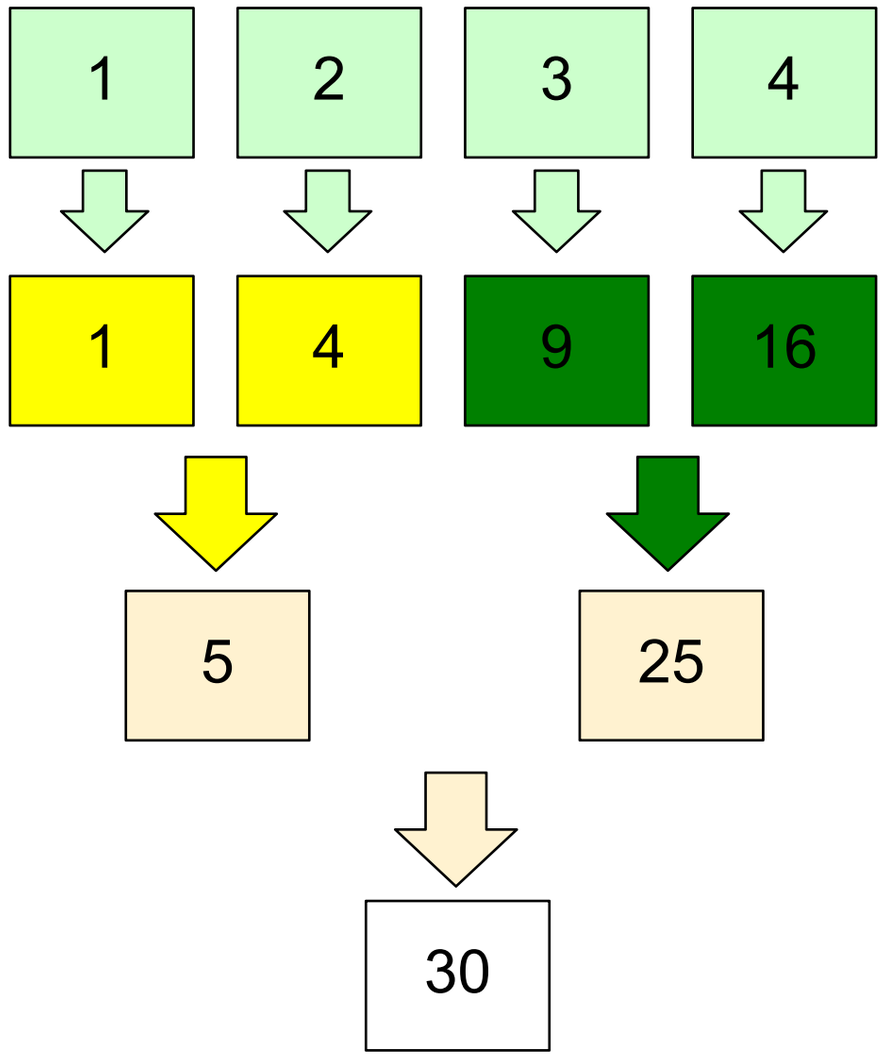
Inspiration: Map & Reduce Functions

- Calculate: $\sum_{i=1}^{4} n^2$

- In Python

```
from functools import reduce
A = [1, 2, 3, 4]
def square(x):
  return x * x
def sum(x, y):
  return x + y

reduce(sum,
  map(square, A))
```



MapReduce Programming Model

- **Map:** (in_key,in_value)→list(interm_key,interm_val)

```
map(in_key, in_val):
// DO WORK HERE
emit(interm_key, interm_val)
```

  - Slice data into “shards” or “splits” and distribute to workers
  - Compute set of intermediate key/value pairs

- **Reduce:**(interm_key,list(interm_value))→list(out_value)

```
reduce(interm_key, list(interm_val)):
// DO WORK HERE
emit(out_key, out_val)
```

  - Combines all intermediate values for a particular key
  - Produces a set of merged output values (usually just one)

MapReduce Word Count Example

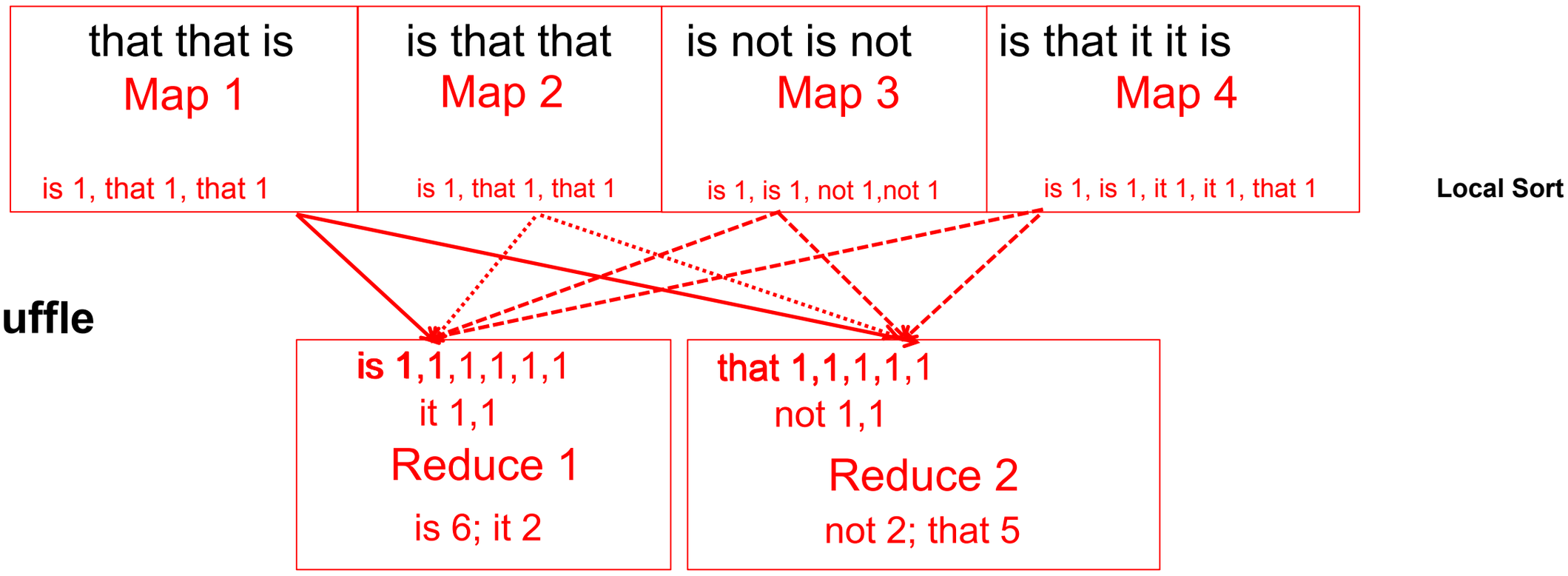
- **Distribute**

that that is Map 1 that 1, that 1, is 1 is 1, that 1, that 1	is that that Map 2 is 1, that 1, that 1 is 1, that 1, that 1	is not is not Map 3 is 1, not 1, is 1, not 1 is 1, is 1, not 1,not 1	is that it it is Map 4 is 1, that 1, it 1, it 1, is 1 is 1, is 1, it 1, it 1, that 1

Local Sort

MapReduce Word Count Example

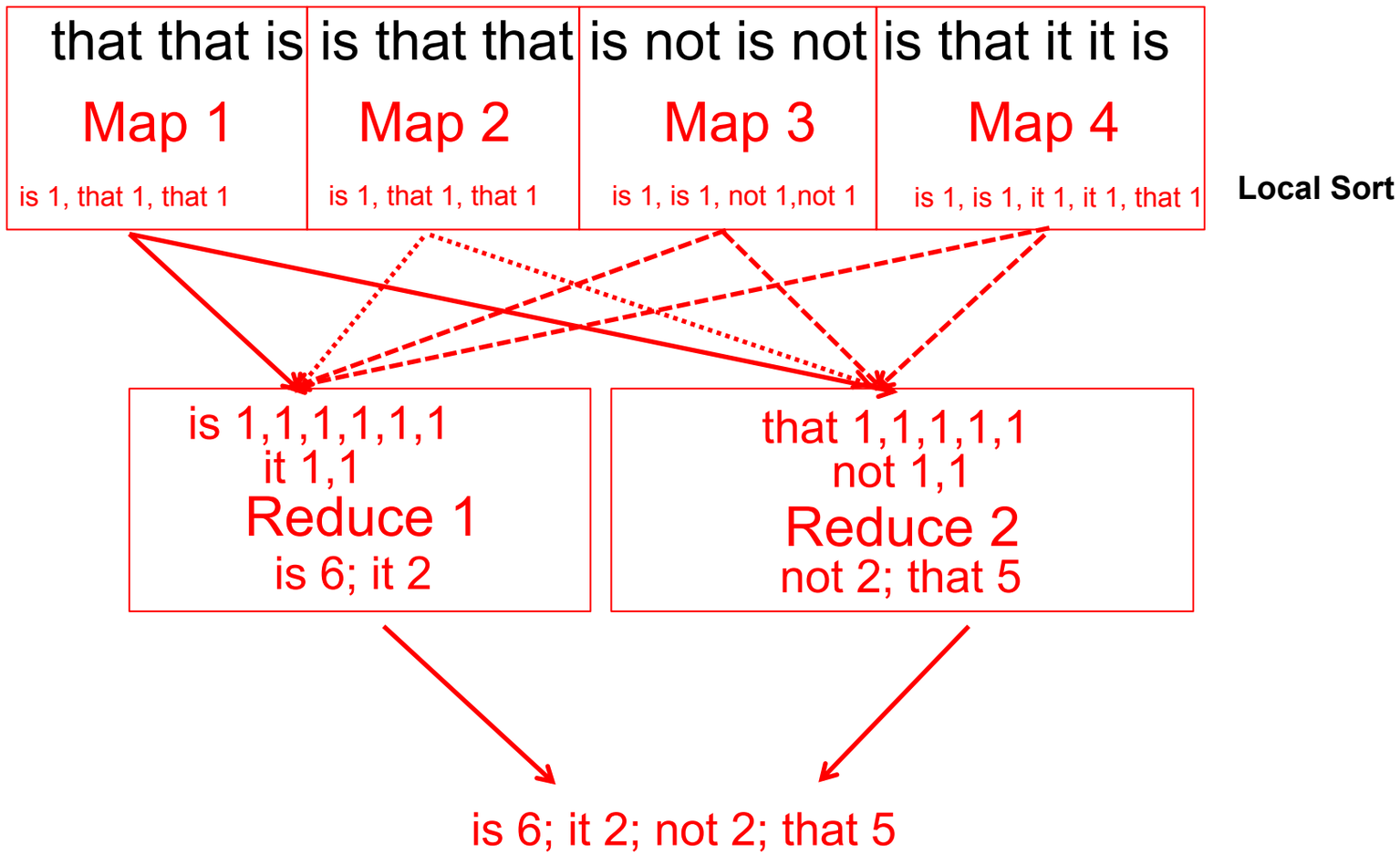
- **Distribute**



- **Shuffle**

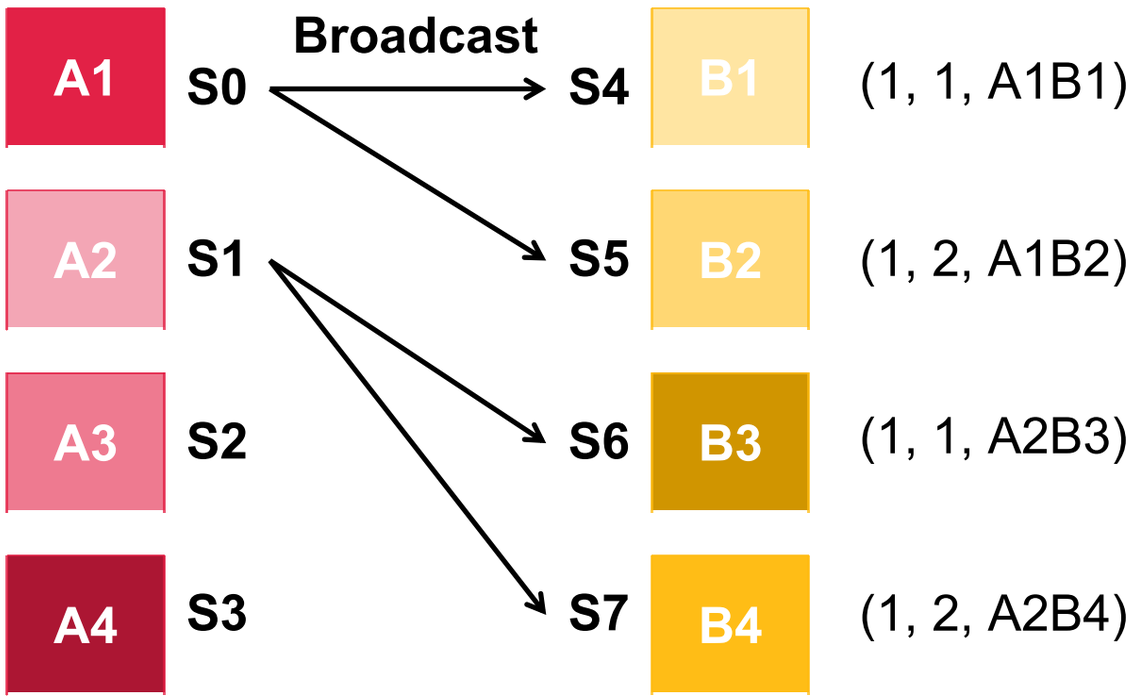
MapReduce Word Count Example

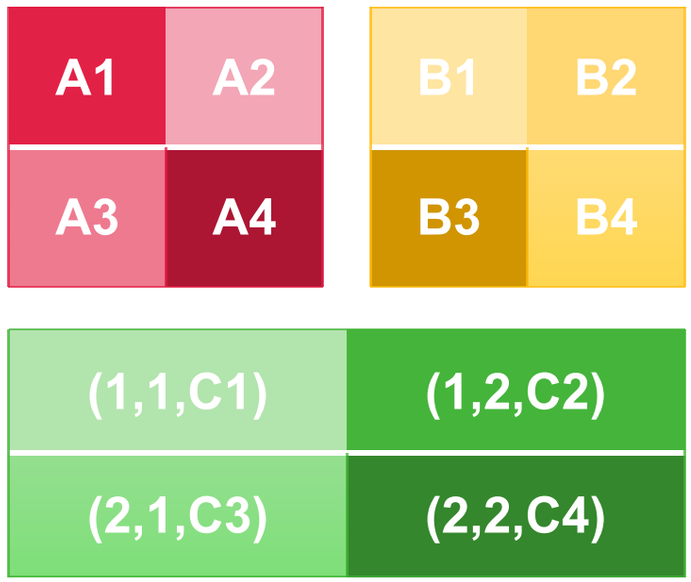
- **Distribute**
- **Shuffle**
- **Collect**



MapReduce GEMM Example

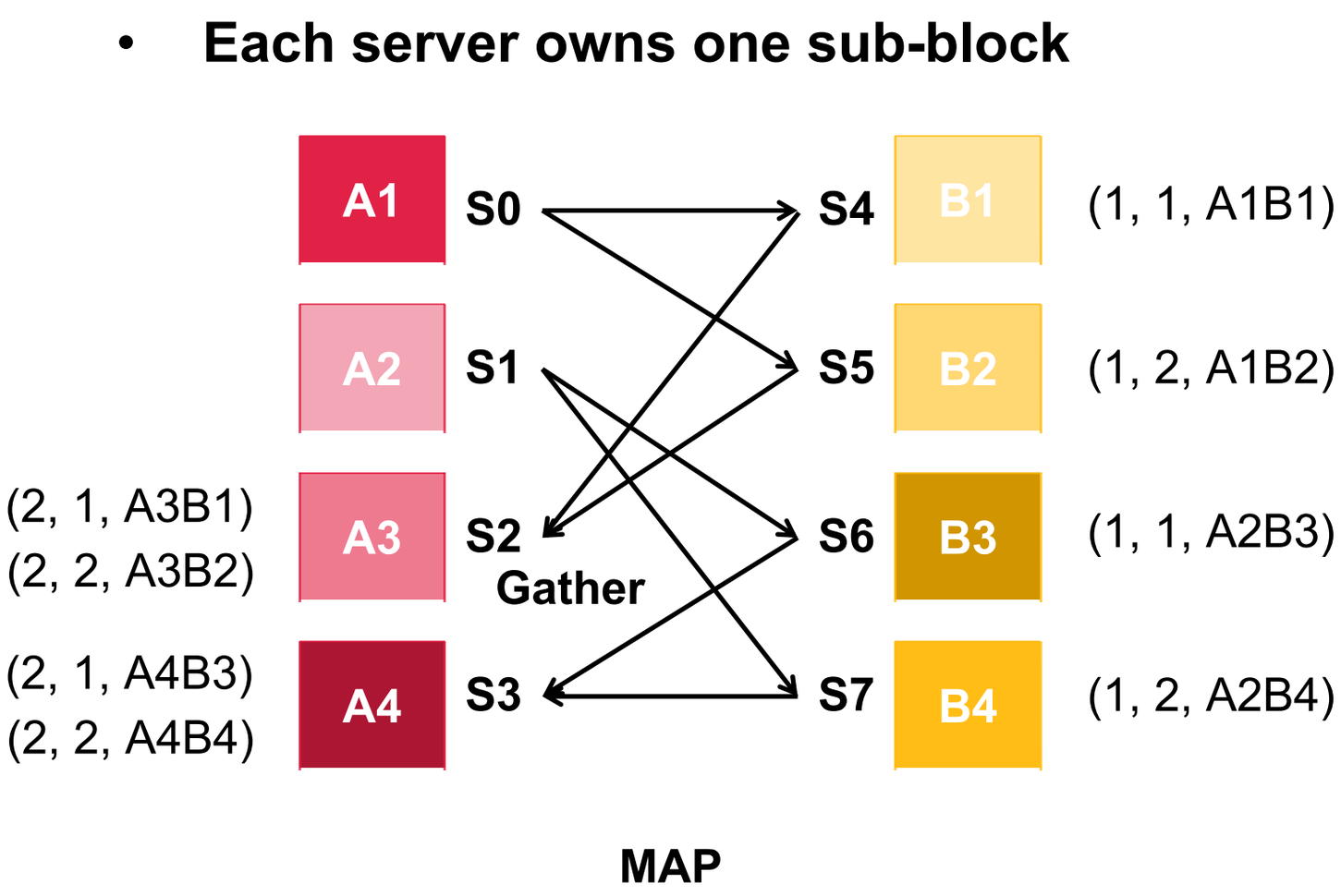
- **Each server owns one sub-block**

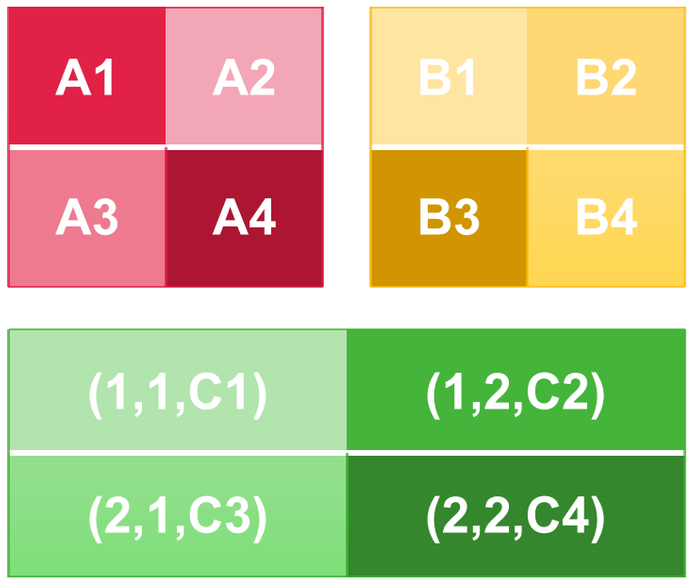




MapReduce GEMM Example

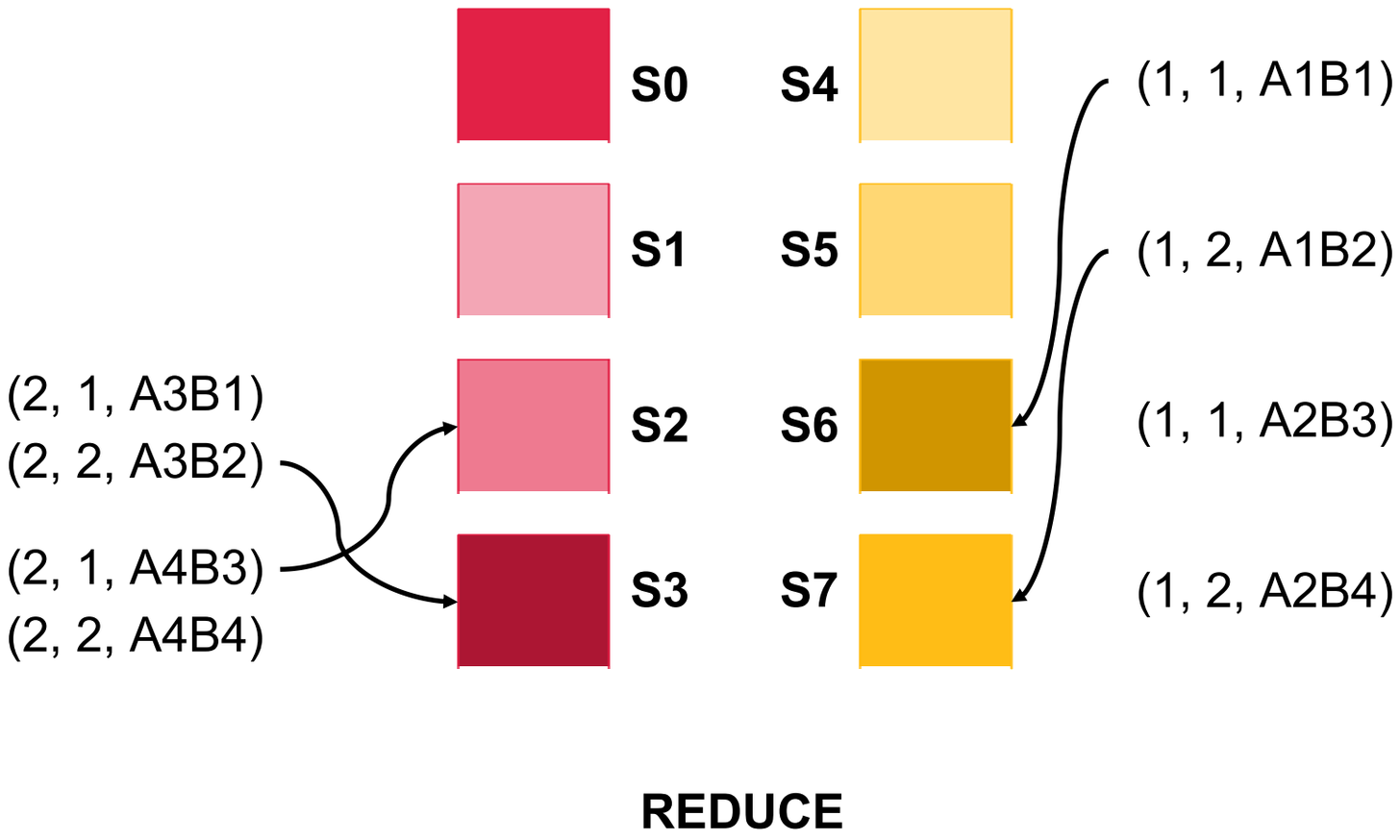
- **Each server owns one sub-block**

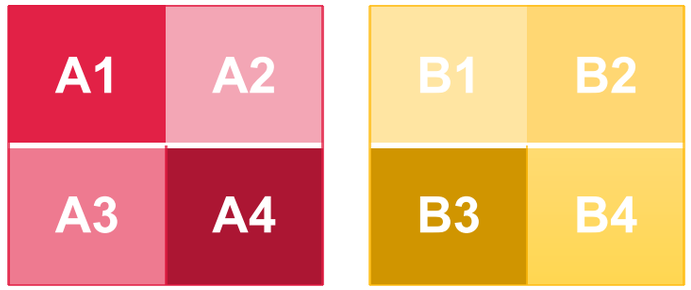


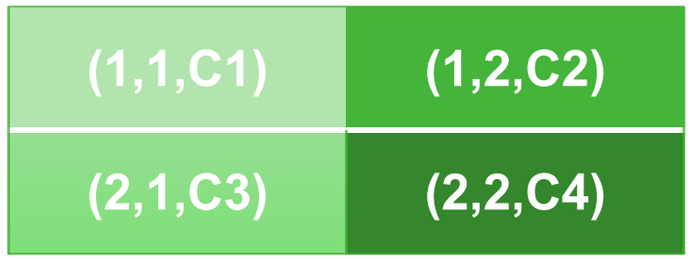


MapReduce GEMM Example

- **Each server owns one sub-block**







Big Data Framework: Hadoop & Spark

- Apache Hadoop
  - Open-source MapReduce Framework
  - Hadoop Distributed File System (HDFS)
  - Hadoop YARN Resource Management
  - MapReduce Java APIs
  - more than half of the Fortune 50 used Hadoop (2013)

- Apache Spark
  - Fast and general engine for large-scale data processing.
  - Running on HDFS
  - Provides Java, Scala, Python APIs for
    - Database
    - Machine learning
    - Graph algorithm

Conclusion

- Warehouse-Scale Computers (WSCs)
  - New class of computers
  - Scalability, energy efficiency, high failure rate
- Cloud Computing
  - Benefits of WSC computing for third parties
  - “Elastic” pay as you go resource allocation
- Request-Level Parallelism
  - High request volume, each largely independent of other
  - Use replication for better request throughput, availability
- MapReduce Data Parallelism
  - Map: Divide large data set into pieces for independent parallel processing
  - Reduce: Combine and process intermediate results to obtain final result
  - Hadoop, Spark