Introduction to Big Data

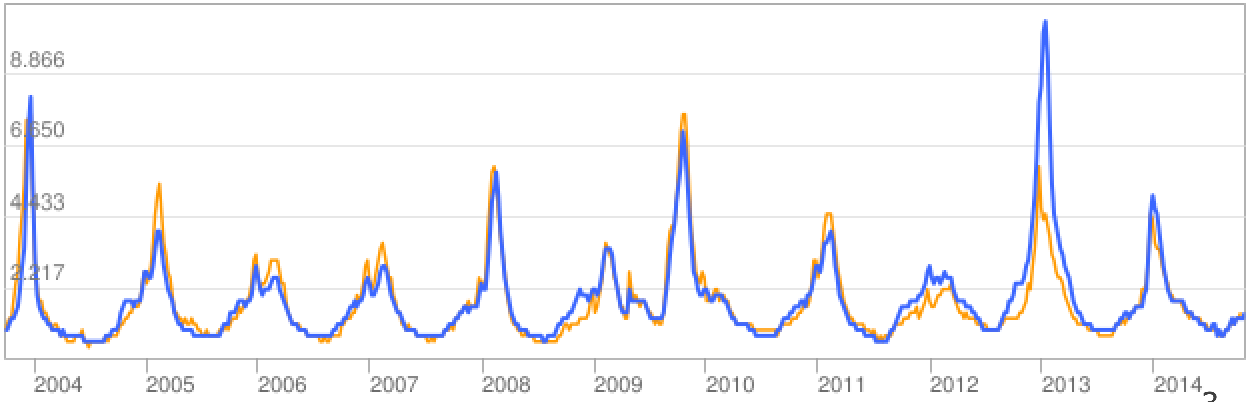
Big data


big data
information
search
petabytes
storage
internet
management
analysis
science
large
software
future
research
sets
new
billion
technology
volume
exabytes
size
based
processing
decision
development
approaches
million

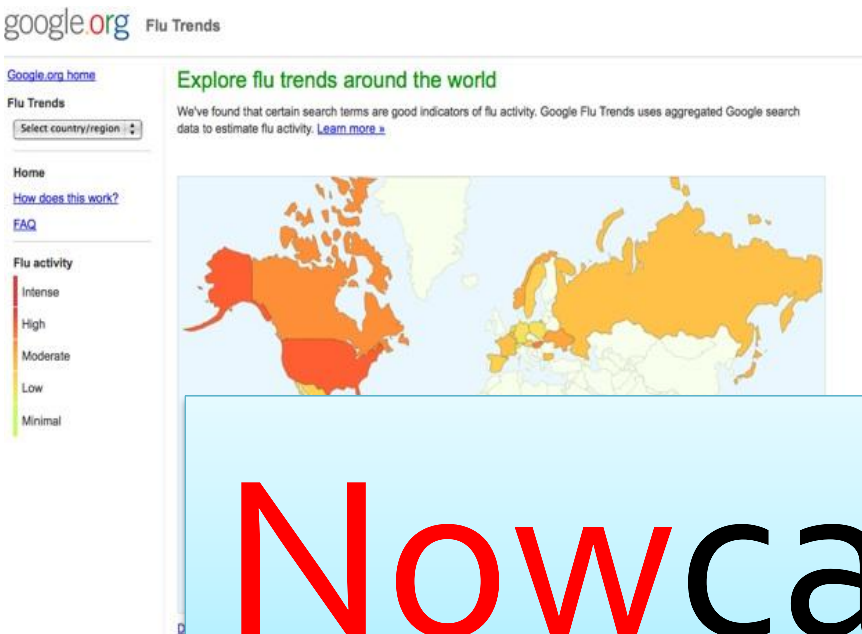

BIG DATA
STORAGE
ANALYTICS
TECHNOLOGIES
INFORMATION
COMPLEX
RESEARCH
EXAMPLES
PETABYTES
DATABASES
SHARED
LARGE
PROCESSING
RECORDS
SEARCH
SYSTEMS
PARALLEL
INCLUDE
TIME
HUNDREDS
NEEDED
SAN
INTERNET
MANAGEMENT
SETS
SIZE
SOFTWARE
NETWORKS
SOCIAL
CAPTURE
DISTRIBUTED
CITATION
EVERY
CURRENT
TERABYTES

Google Flu trends

- February 2010
  - Google detected flu outbreak two weeks ahead of CDC data (Centers for Disease Control and Prevention – U.S.A)
  - Based on the analysis of Google search queries



Google Flu trends

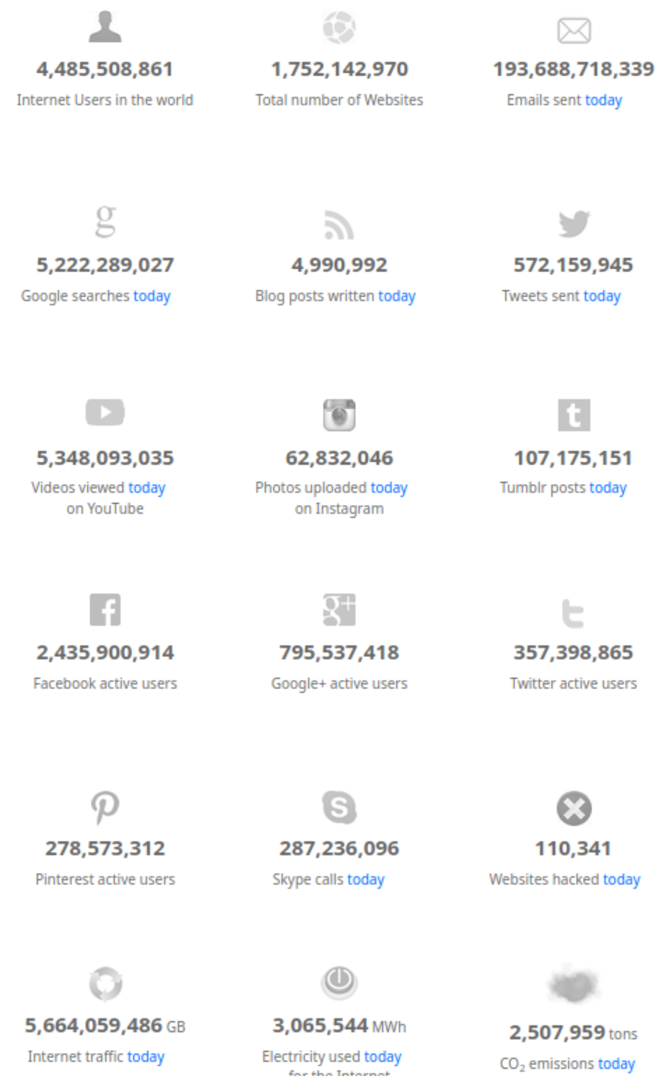


- February 2010
  - Google detected flu outbreak two weeks ahead of CDC data (Centers for Disease Control and
  - S.A)
  - alysis of
  - ueries

Nowcasting

Data on the Internet…

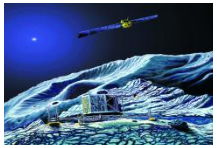
- Internet live stats
  - http://www.internetlivestats.com/



Who generates big data?

- User Generated Content (Web & Mobile)
  - E.g., Facebook, Instagram, Yelp, TripAdvisor, Twitter, YouTube

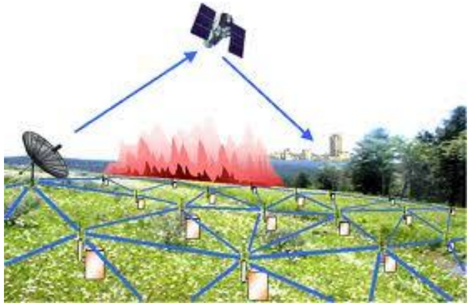
- Health and scientific computing

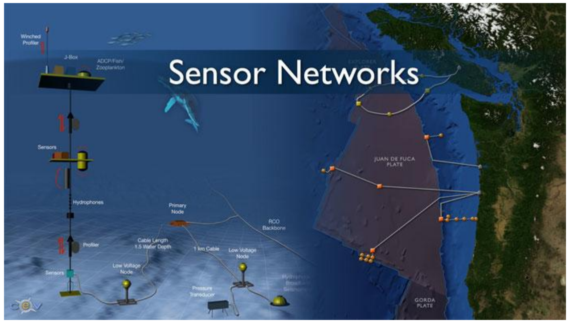


Who generates big data?

- Log files
  - Web server log files, machine system log files

- Internet Of Things (IoT)
  - Sensor networks, RFIDs, smart meters





The Vs of big data

- The 3Vs of big data
  - **V**olume: scale of data
  - **V**ariety: different forms of data
  - **V**elocity: analysis of streaming data
- … but also
  - **V**eracity: uncertainty of data
  - **V**alue: exploit information provided by data

Big Data Frameworks: Motivations and Challenges

Data volumes

- The amount of data increases every day
- Some numbers (**~ 2012**):
  - Data processed by Google every day: 100+ PB
  - Data processed by Facebook every day: 10+ PB
- To analyze them, systems that scale with respect to the data volume are needed

Data volumes: Google Example

- Analyze **10 billion web pages**
- Average size of a webpage: **20KB**
- Size of the collection: 10 billion x 20KBs = **200TB**
- HDD hard disk read bandwidth: 150MB/sec
- Time needed to **read all web pages** (**without analyzing them**): more than **15 days**
- A single node architecture is not adequate

Data volumes: Google Example with SSD

- Analyze **10 billion web pages**
- Average size of a webpage: **20KB**
- Size of the collection: 10 billion x 20KBs = **200TB**
- SSD hard disk read bandwidth: 550MB/sec
- Time needed to **read all web pages** (**without analyzing them**): more than **4 days**
- A single node architecture is not adequate

Failures

- Failures are part of everyday life, especially in data center
  - A single server stays up for 3 years (~1000 days)
    - 10 servers → 1 failure every 100 days (~3 months)
    - 100 servers → 1 failure every 10 days
    - 1000 servers → 1 failure/day
- Sources of failures
  - Hardware/Software
  - Electrical, Cooling, ...
  - Unavailability of a resource due to overload

Failures

- LALN data [DSN 2006]
  - Data for 5000 machines, for 9 years
  - Hardware failures: 60%, Software: 20%, Network 5%
- DRAM error analysis [Sigmetrics 2009]
  - Data for 2.5 years
  - 8% of DIMMs affected by errors
- Disk drive failure analysis [FAST 2007]
  - Utilization and temperature major causes of failures

Failures

- Failure types
  - Permanent
    - E.g., Broken motherboard
  - Transient
    - E.g., Unavailability of a resource due to overload

Network bandwidth

- Network becomes the bottleneck if big amounts of data need to be exchanged between nodes/servers
  - Network bandwidth (in a data center): 10Gbps
  - Moving 10 TB from one server to another takes more than 2 hours
    $\rightarrow$ **Data** should be **moved across nodes only when** it is **indispensable**

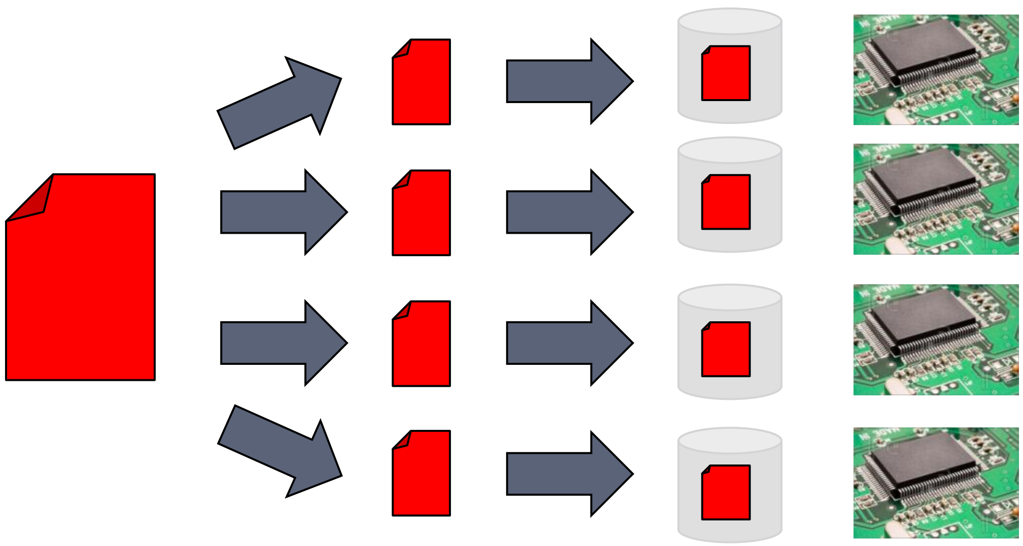
The solution

- **Transfer** the **processing power** and **code to** the **data**
- Usually, codes/programs are small (few MBs) → **Move code** (programs) and **computation** to data

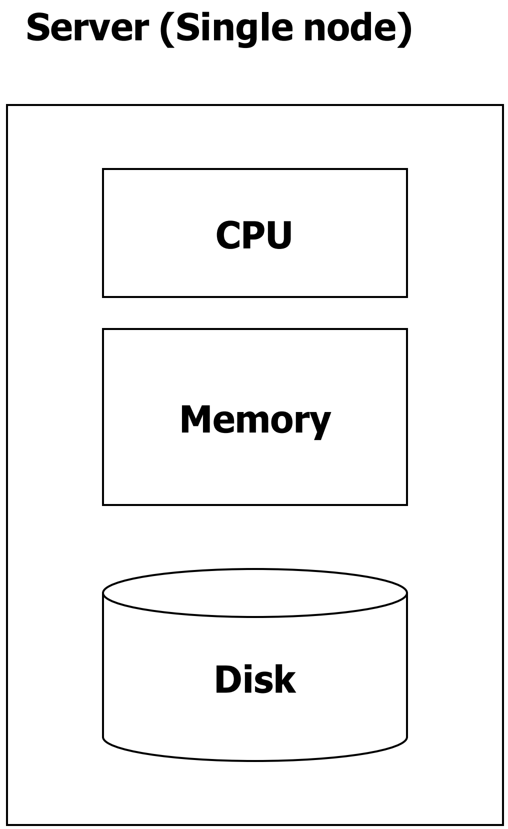
Data locality

The solution

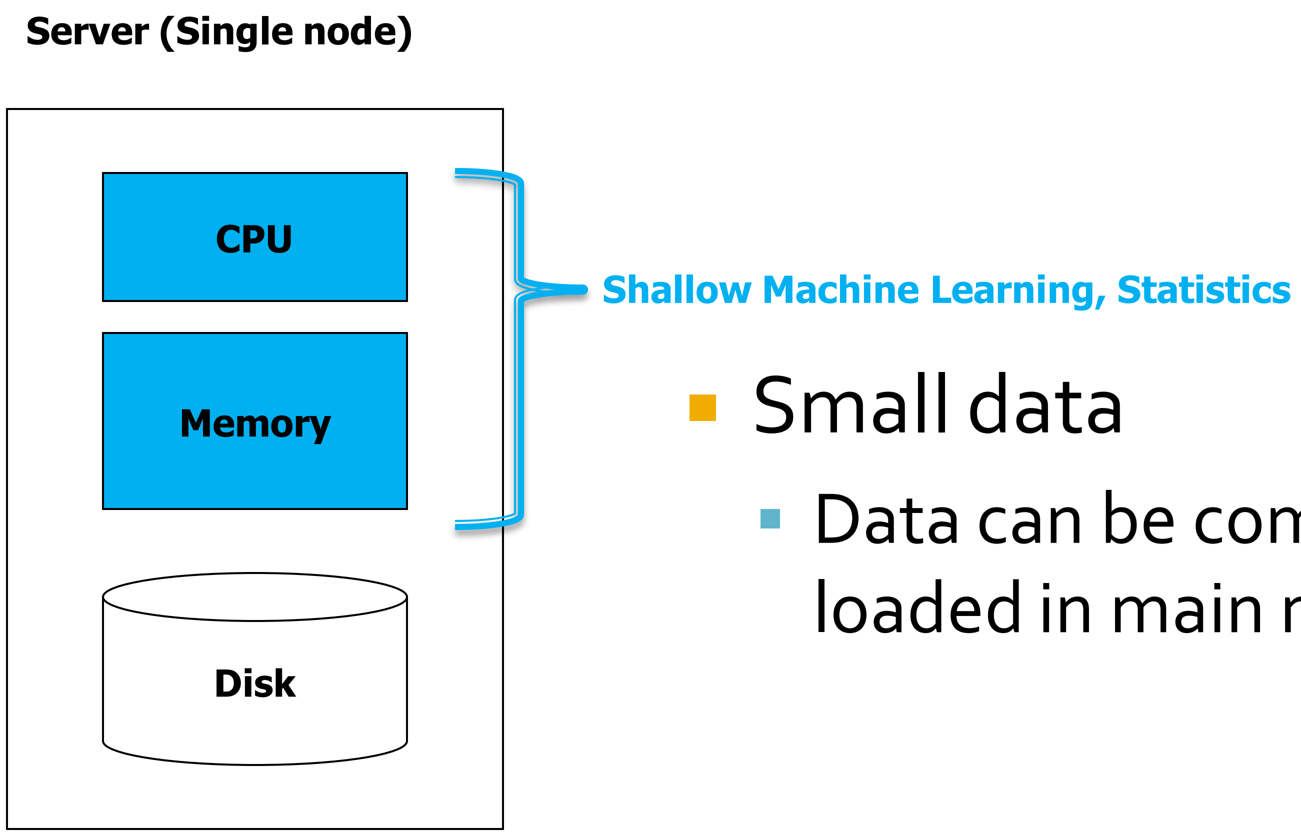
- **Multiple distributed disks**
  - Each one holding a portion of a large dataset
- **Process in parallel different** file **portions** from different disks



Single-node architecture

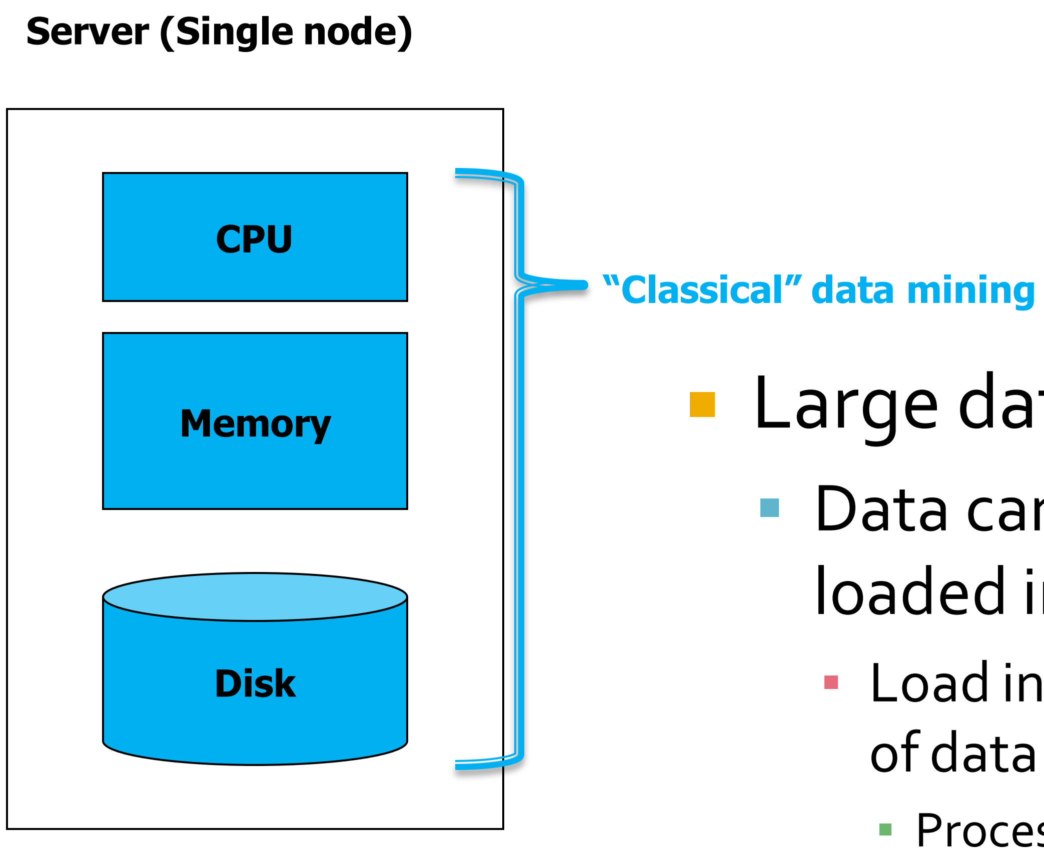


Single-node architecture



- Small data
  - Data can be completely loaded in main memory

Single-node architecture

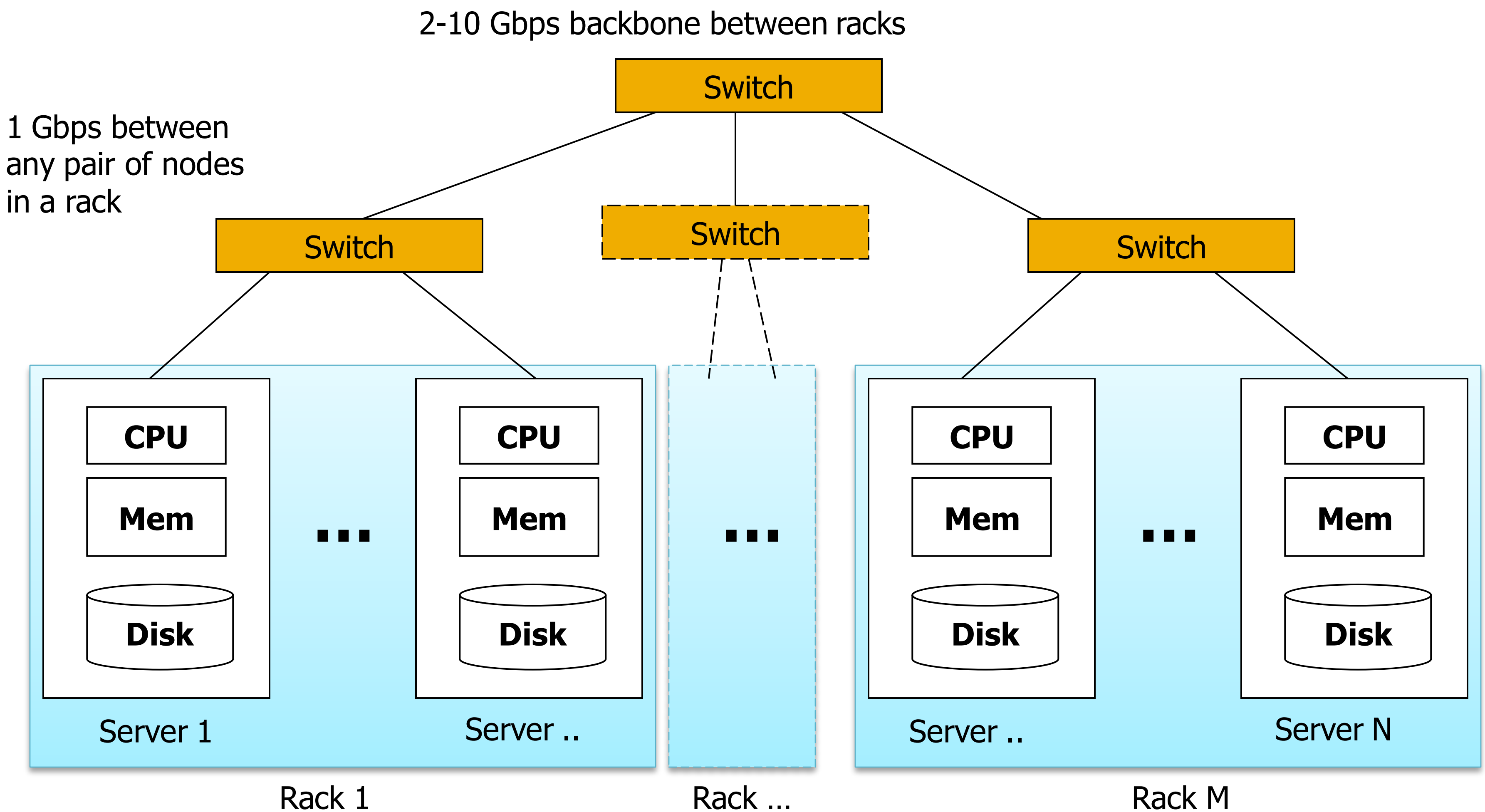


- Large data
  - Data can not be completely loaded in main memory
    - Load in main memory one chunk of data at a time
      - Process it and store some statistics
    - Combine statistics to compute the final result

Cluster Architecture

- Cluster of servers (data center)
  - Computation is distributed across servers
  - Data are stored/distributed across servers
- Standard architecture in the Big data context (**~ 2012**)
  - Cluster of commodity Linux nodes/servers
    - 32 GB of main memory per node
  - Gigabit Ethernet interconnection

Commodity Cluster Architecture



Each rack contains 16-64 nodes

Data center

Data center

Scalability

- Current systems must scale to address
  - The increasing amount of data to analyze
  - The increasing number of users to serve
- Two approaches are usually used to address scalability issues
  - Vertical scalability (scale up)
  - Horizontal scalability (scale out)

Scale up vs. Scale out

- Vertical scalability (scale up)
  - Add more power/resources (main memory, CPUs) to a single node (high-performing server)
    - Cost of super-computers is not linear with respect to their resources
- Horizontal scalability (scale out)
  - Add more nodes (commodity servers) to a system
    - The cost scales approximately linearly with respect to the number of added nodes
    - But data center efficiency is a difficult problem to solve

Scale up vs. Scale out

- For **data-intensive workloads**, a **large number of commodity servers** is **preferred** over a small number of high-performing servers
  - At the same cost, we can deploy a system that processes data more efficiently and is more fault-tolerant
- **Horizontal scalability** (**scale out**) is preferred for **big data** applications
  - But distributed computing is hard

    →New systems hiding the complexity of the distributed part of the problem to developers are needed

Cluster computing challenges

- Distributed programming is hard
  - Problem decomposition and parallelization
  - Task synchronization
- Task scheduling of distributed applications is critical
  - Assign tasks to nodes by trying to
    - Speed up the execution of the application
    - Exploit (almost) all the available resources
    - Reduce the impact of node failures

Cluster computing challenges

- Distributed data storage
  - How do we store data persistently on disk and keep it available if nodes can fail?
    - Redundancy is the solution, but it increases the complexity of the system
- Network bottleneck
  - Reduce the amount of data send through the network
    - Move computation and code to data

Cluster computing challenges

- Distributed computing is not a new topic
  - **HPC** (High-performance computing) ~1960
  - **Grid computing** ~1990
  - **Distributed databases** ~1990
- Hence, many solutions to the mentioned challenges are already available
- But we are now facing big data driven-problems

  → The former solutions are **not adequate to address big data** volumes

Big Data Challenges: A Summary

- The challenges:
  - Parallelization/Distributed computation
  - Distributed storage of large data sets (Terabytes, Petabytes, ..)
  - Node failure management
  - Network bottleneck
  - Diverse input format (data diversity & heterogeneity)

Typical Big Data Problem

- Typical Big Data Problem
  - Iterate over a big amount of records/objects
  - Extract something of interest from each record/object
  - Aggregate intermediate results
  - Generate final output/global result