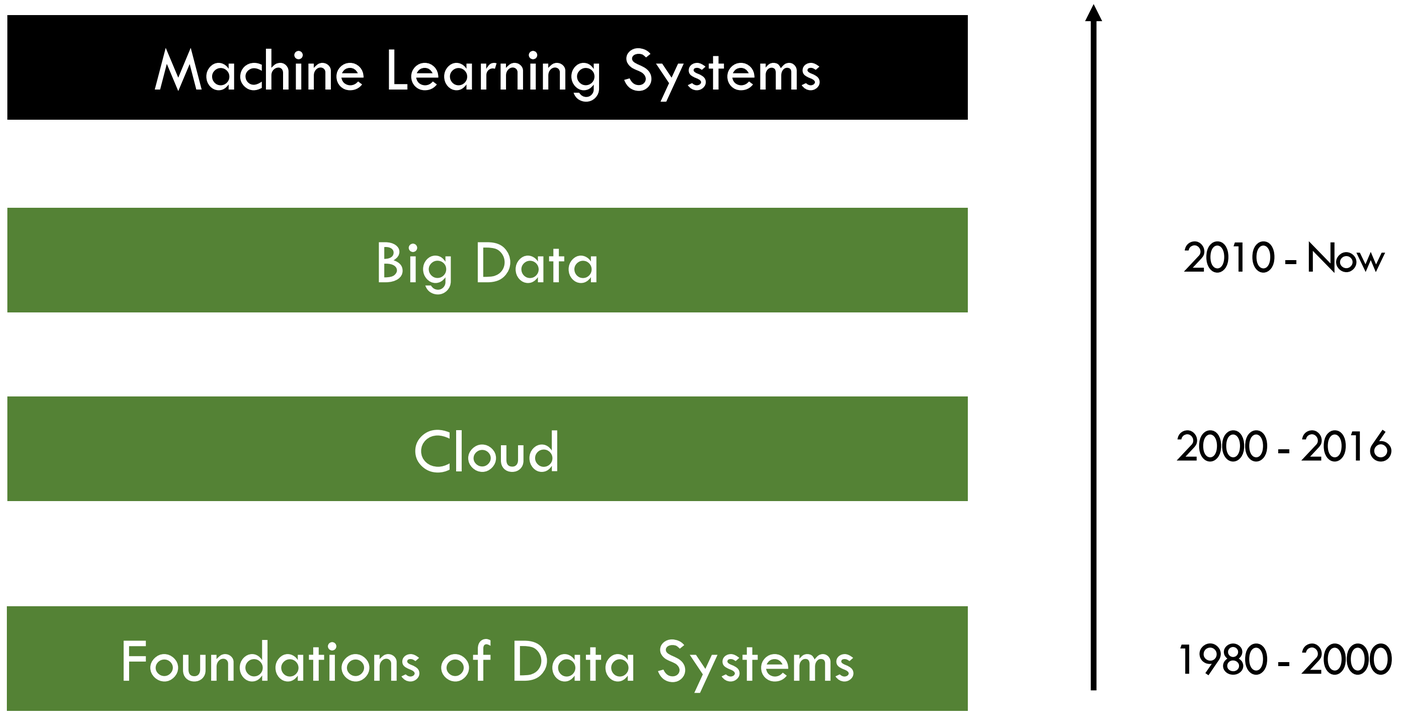
Where We Are


Machine Learning Systems
Big Data
Cloud
Foundations of Data Systems
2010 - Now
2000 - 2016
1980 - 2000

Logistics

- PA2 released
  - 2 Weeks to finish
- Please participate in guest talk discussion
  - Will initiate discussion threads after every guest talk
  - Develop an understand of technological trend is the most important outcome of this course
- Next guest talk: Ray by Stephanie Wang
- Reading summary
  - The 2 page limit is meant to reduce your workload, but if you want to go over 2 pages, just do it.

Logistics

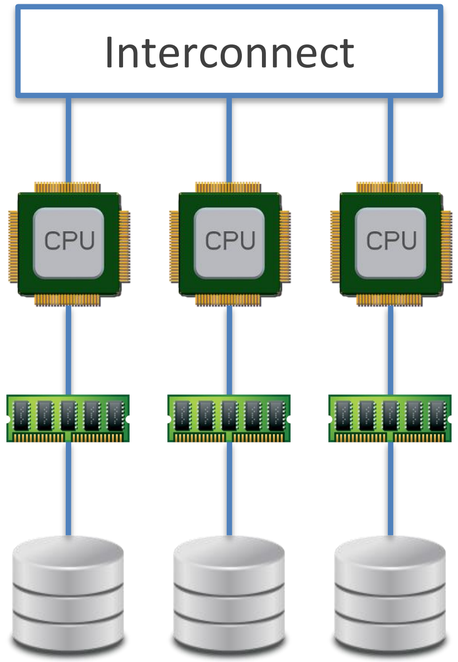
- PA3 will be released in end of week 8
- Scribe duties
  - The scribe scheduling was tentative
  - Please do scribe the day you signed up, even if the topic was changed from the schedule
  - If you really dislike the topic – write an email to instructors

Problems Distributed Systems Need to Solve

- **How to Distribute Data?**
  - **Replicate / Partition the data**
  - **Replicate -> redundancy -> fault tolerance**
  - **Partition -> scalability -> hot partition problem**
- How to Distribute Compute?
  - Batch processing
  - Streaming processing
- How to Coordinate/Synchronize?

Partitioning challenges

- How to partition and how to index?
- How to add or remove nodes?
- How to route the requests and execute queries?



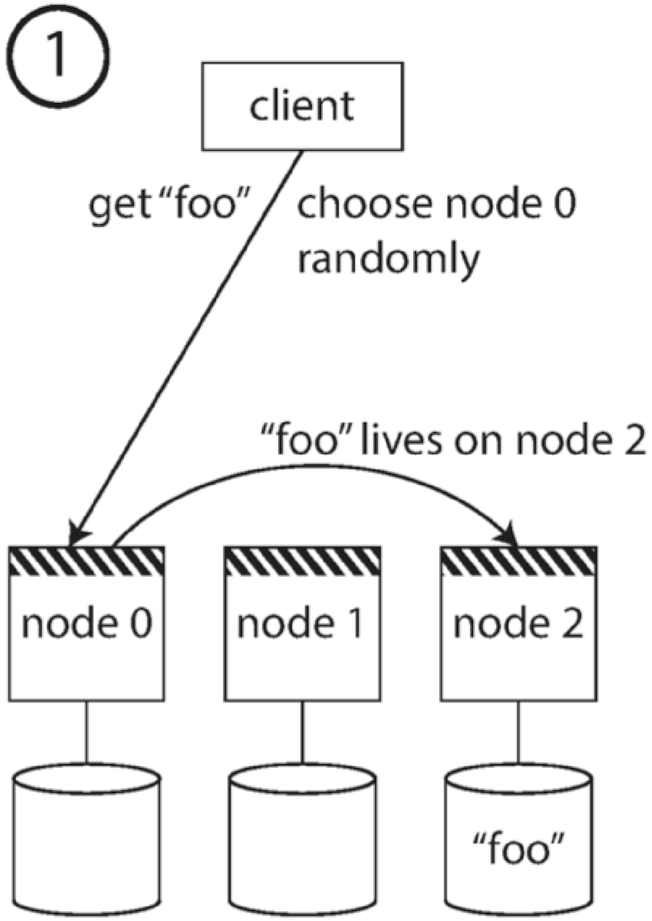
Shared-Nothing Parallelism

Routing paradigm #1

- Contact any node (e.g., a round-robin load balancer),
- If the node has the data copy, respond.
- If not, forward, receives the reply, and passes the reply along to the client.

Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications

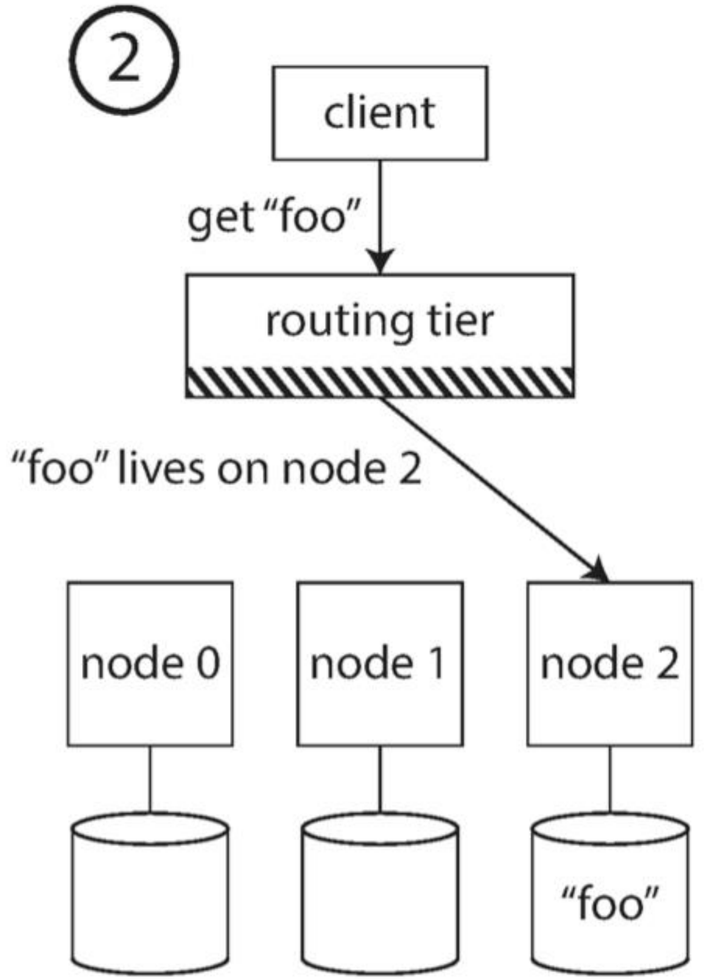
Ion Stoica; Robert Morris, David Karger, M. Frans Kaashoek, Hari Balakrishnan†
MIT Laboratory for Computer Science
chord@lcs.mit.edu
http://pdos.lcs.mit.edu/chord/



▨ = the knowledge of which partition is assigned to which node

Routing paradigm #2

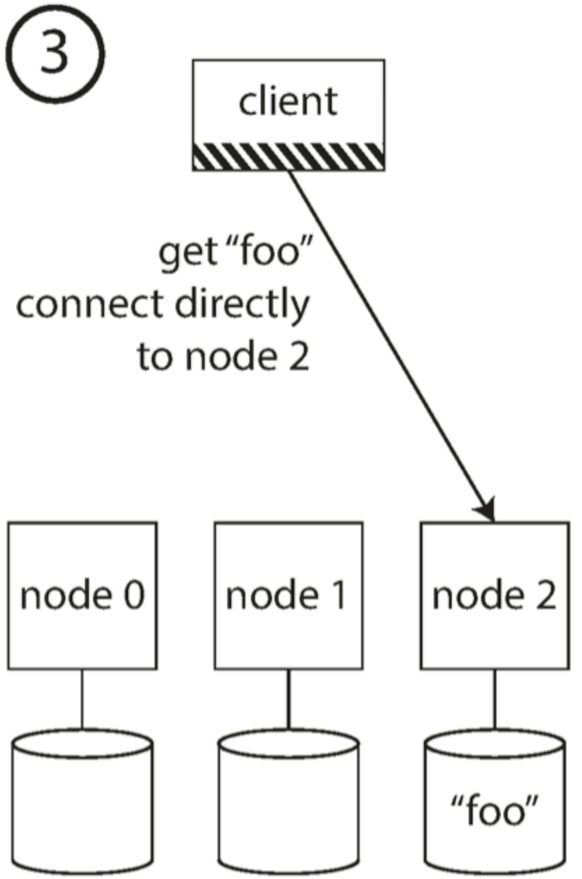
- Send all requests to a routing tier first.
- The routing tier forward all the requests.



▨ = the knowledge of which partition is assigned to which node

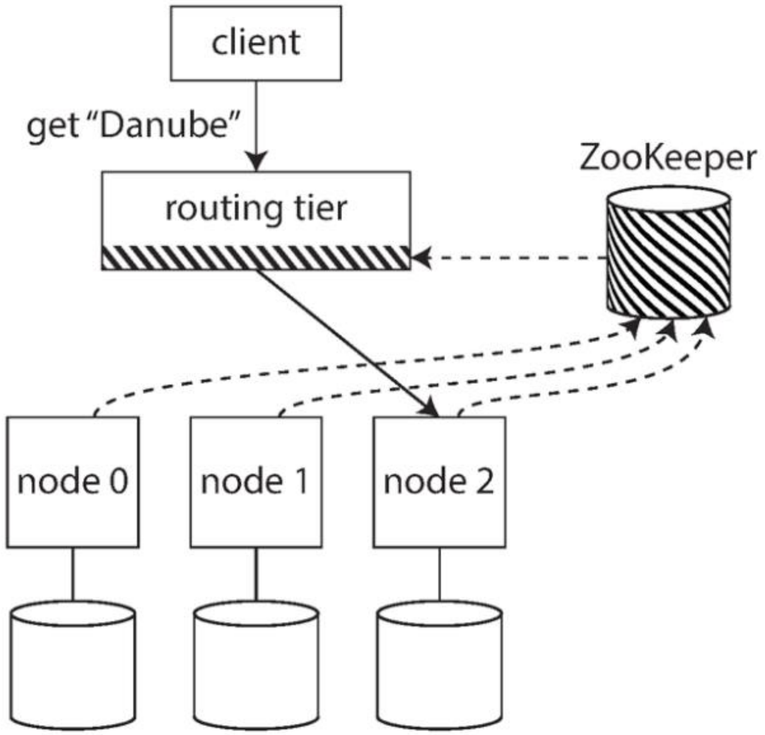
Routing paradigm #3

- The client is aware of the partitioning and the assignment of partitions to nodes.
- No intermediary.



= the knowledge of which partition is assigned to which node

ZooKeeper

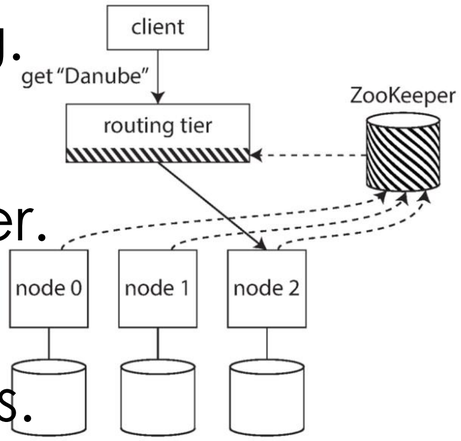


Key range	Partition	Node	IP address
A-ak — Bayes	partition 0	node 0	10.20.30.100
Bayeu — Ceanothus	partition 1	node 1	10.20.30.101
Ceara — Deluc	partition 2	node 2	10.20.30.102
Delusion — Frenssen	partition 3	node 0	10.20.30.100
Freon — Holderlin	partition 4	node 1	10.20.30.101
Holderness — Krasnoje	partition 5	node 2	10.20.30.102
Krasnokamsk — Menadra	partition 6	node 0	10.20.30.100
Menage — Ottawa	partition 7	node 1	10.20.30.101
Otter — Rethimnon	partition 8	node 2	10.20.30.102
Reti — Solovets	partition 9	node 0	10.20.30.100
Solovyov — Truck	partition 10	node 1	10.20.30.101
Trudeau — Zywiec	partition 11	node 2	10.20.30.102

= the knowledge of which partition is assigned to which node

ZooKeeper

- Each node registers itself in ZooKeeper.
- ZooKeeper maintains the mapping.
- Other actors (different in three paradigms) subscribe to ZooKeeper.
- Whenever the partition mapping changes, ZooKeeper notifies actors.



Key range	Partition	Node	IP address
A-ak — Bayes	partition 0	node 0	10.20.30.100
Bayeu — Ceanothus	partition 1	node 1	10.20.30.101
Ceara — Deluc	partition 2	node 2	10.20.30.102
Delusion — Frenssen	partition 3	node 0	10.20.30.100
Freon — Holderlin	partition 4	node 1	10.20.30.101
Holderness — Krasnoje	partition 5	node 2	10.20.30.102
Krasnokamsk — Menadra	partition 6	node 0	10.20.30.100
Menage — Ottawa	partition 7	node 1	10.20.30.101
Otter — Rethimnon	partition 8	node 2	10.20.30.102
Reti — Solovets	partition 9	node 0	10.20.30.100
Solovyov — Truck	partition 10	node 1	10.20.30.101
Trudeau — Zywiec	partition 11	node 2	10.20.30.102

= the knowledge of which partition is assigned to which node

Takeaway

- The benefits of Partitioning and Replication.
- The challenges of Partitioning and Replication.
- The tradeoffs of different strategies.
- Replication: single-leader, multiple-leader, leaderless
- Partition: Key range, hash, hybrid.
  - Partition rebalancing strategies: fixed, dynamic
  - Partition routing

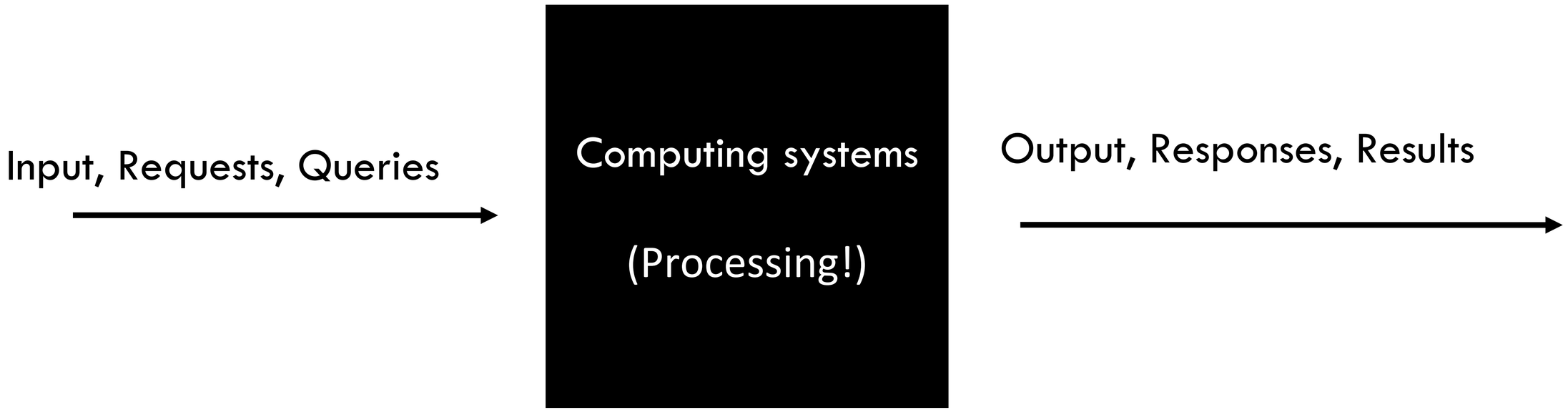
Problems Distributed Systems Need to Solve

- How to Distribute Data?
  - Replicate / Partition the data
  - Replicate -> redundancy -> fault tolerance
  - Partition -> scalability
- **How to Distribute Compute?**
  - Batch processing
  - Streaming processing
- How to Coordinate/Synchronize?

Today's topic: Batch Processing

- Overview
- IO & Unix pipes
- MapReduce
- Beyond MapReduce

Basic Computing System Paradigm



Processing latency

feedback cycle time

direct manipulation
no visible lag

Interactive data science!

Online system: handle request ASAP

turn-taking
minutes to seconds

Stream processing systems
(near-real-time systems)

batch-processing
hours or overnight

Batch processing systems
(Offline systems)

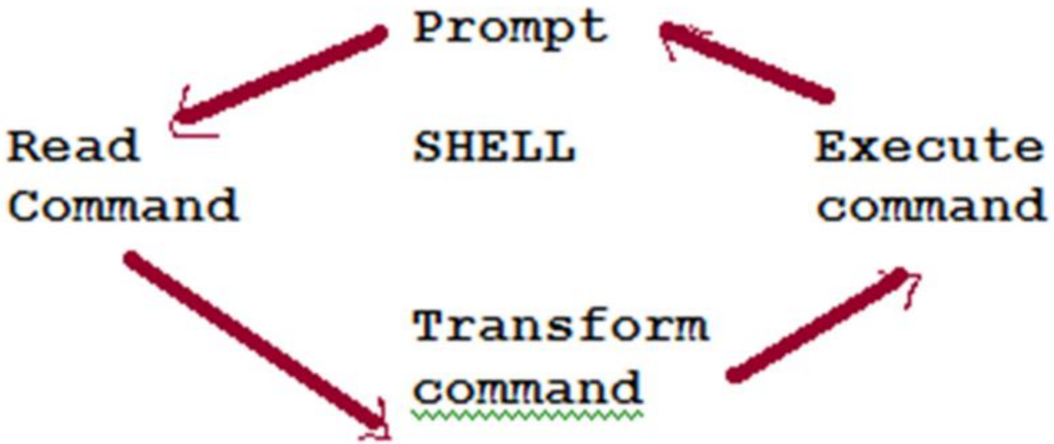
Figure from https://publishup.uni-potsdam.de/opus4-ubp/frontdoor/deliver/index/docId/10090/file/mueller_diss.pdf

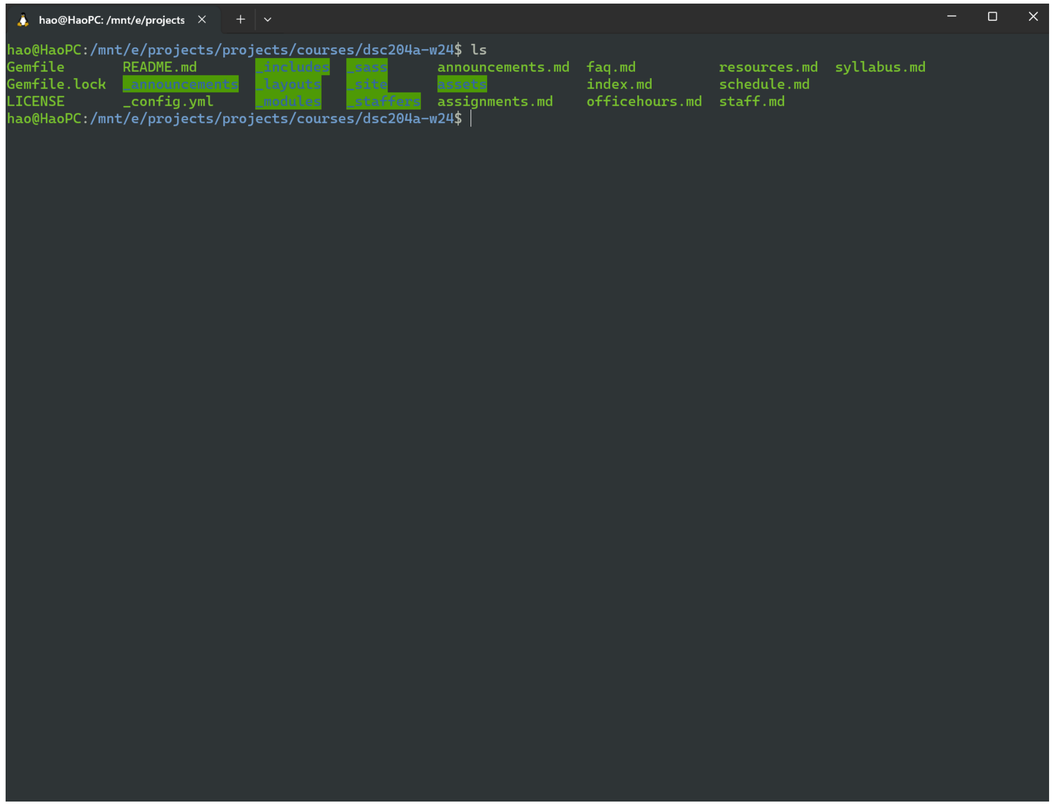
Today's topic: Batch Processing

- Overview
- IO & Unix pipes
- MapReduce
- Beyond MapReduce

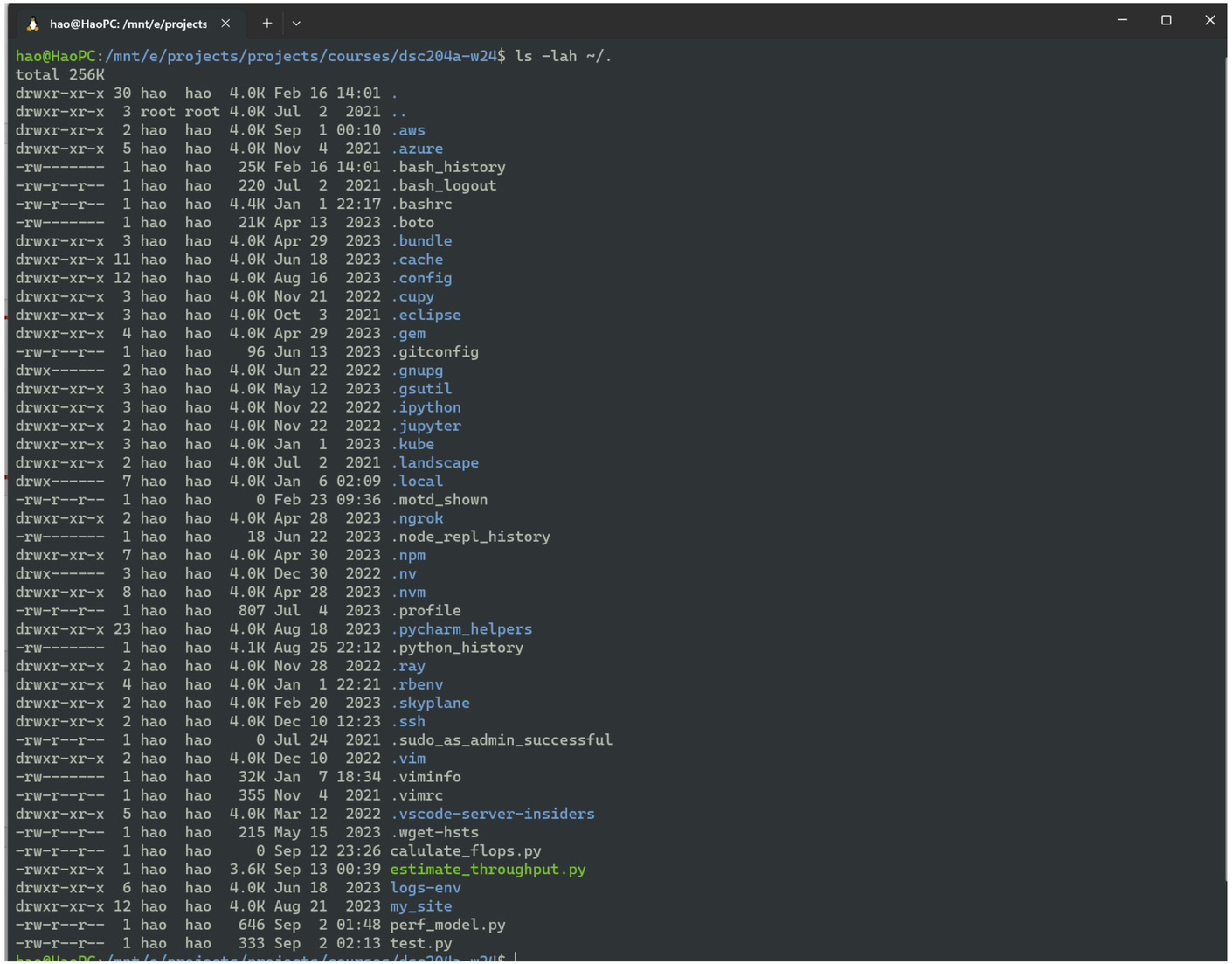
The shell

A command line interpreter that provides the interface to Unix OS.





Shell example



“.****rc”

Run commands

Useful shell commands

- Shell already has a collection of rich commands
  - Some Useful commands
    - uptime, cut, date, cat, finger, hexdump, man, md5sum, quota,
    - mkdir, rmdir, rm, mv, du, df, find, cp, chmod, cd
    - uname, zip, unzip, gzip, tar
    - tr, sed, sort, uniq, ascii
    - Type “man command” to read about shell commands

What do these shell commands do?

- cat dups.txt | sort | uniq
- cat dups.txt | sort -V | uniq
- cat dups.txt | sort -V | uniq > outfile.txt
- tr "a" "e" < z.txt
- cat z.txt | tr a e

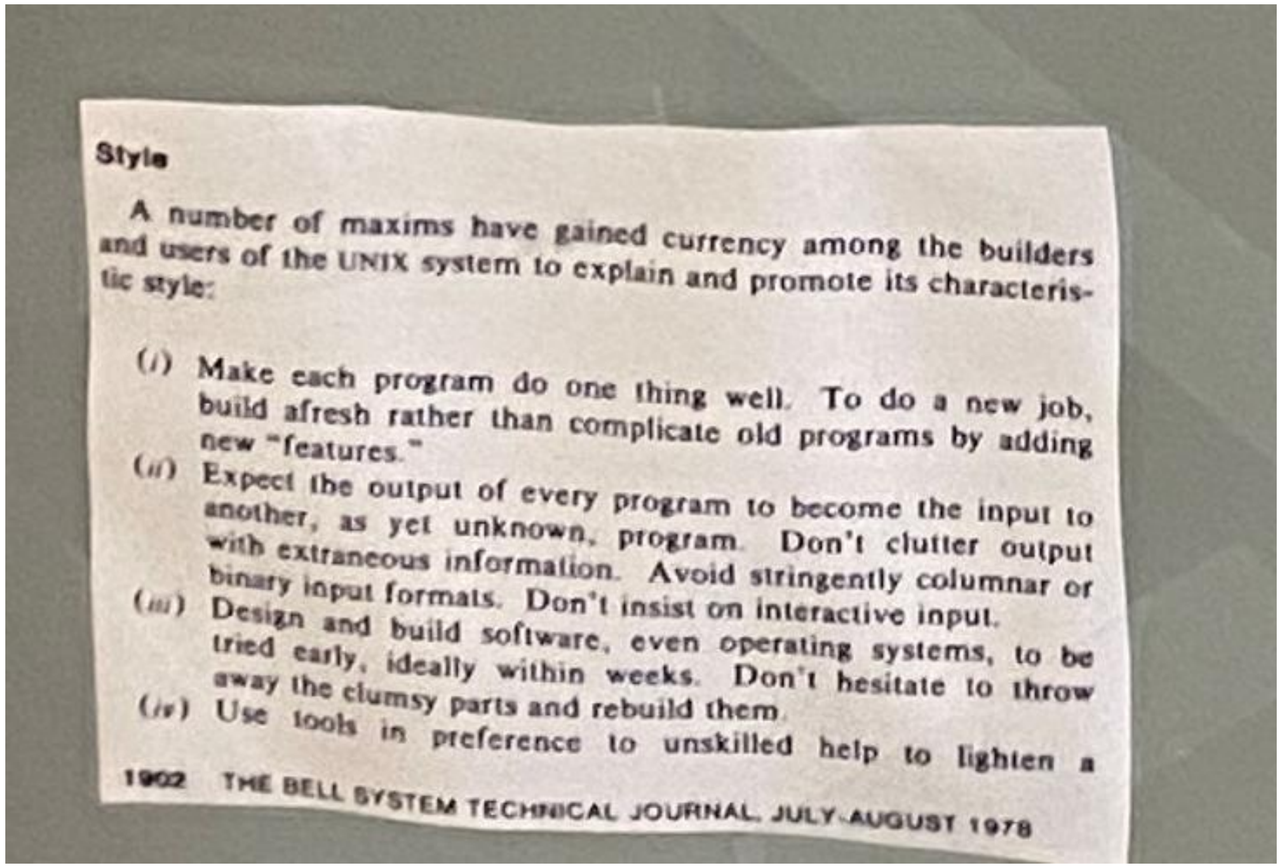
Batch processing with Unix Tools

```
cat /var/log/nginx/access.log |  ❶
  awk '{print $7}' |             ❷
  sort             |             ❸
  uniq -c          |             ❹
  sort -r -n       |             ❺
  head -n 5                      ❻
```

```
4189 /favicon.ico
3631 /2013/05/24/improving-security-of-ssh-private-keys.html
2124 /2012/12/05/schema-evolution-in-avro-protocol-buffers-thrift.html
1369 /
 915 /css/typography.css
```

- Read the log file.
- Split each line into fields by white space, output only the 7th element (requested URL).
- Alphabetically sort
- Filter out repeated lines.
- Sort it again based on the line number (-n)
- Out put the first five lines.

Unix philosophy



Style

A number of maxims have gained currency among the builders and users of the UNIX system to explain and promote its characteristic style:

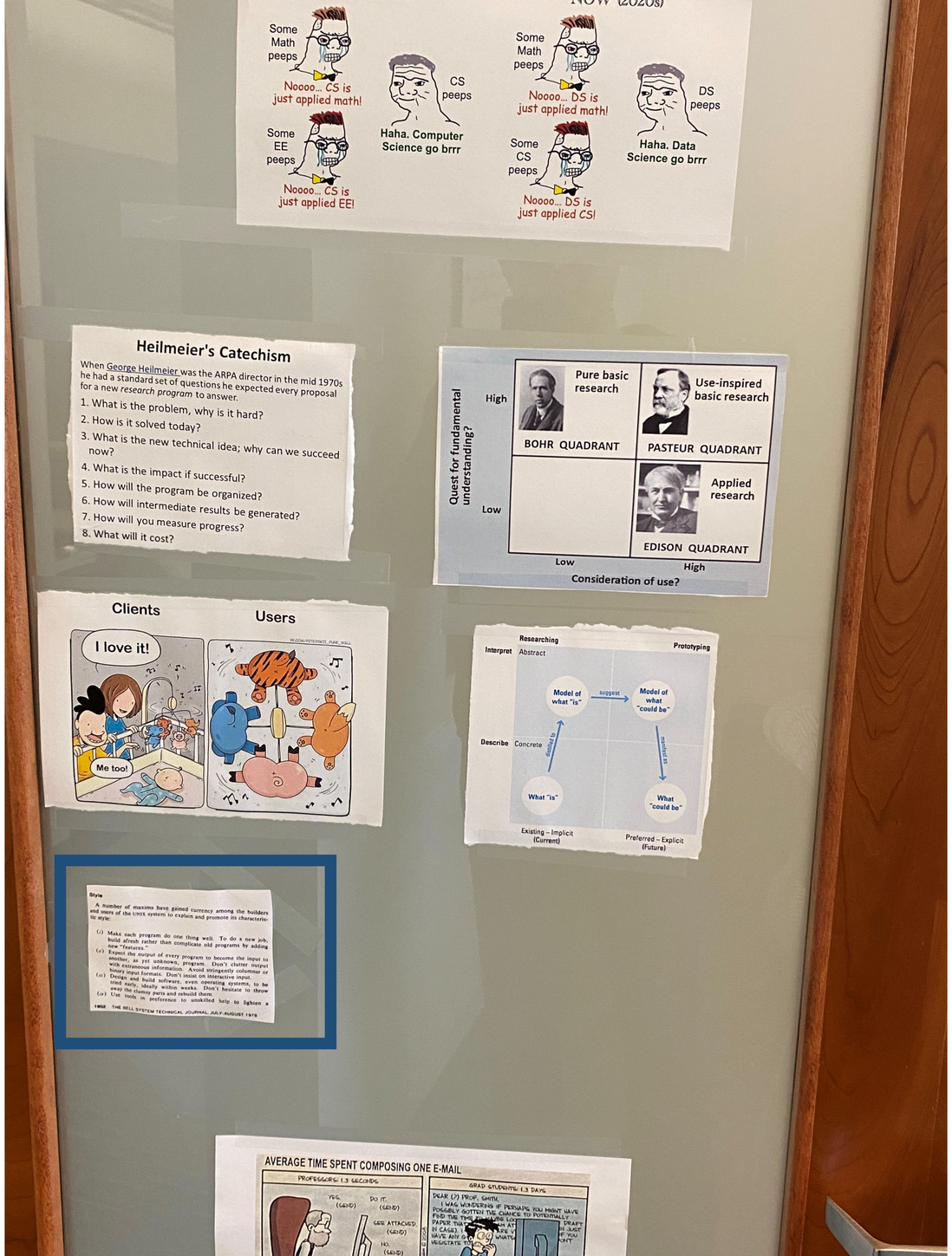
(*i*) Make each program do one thing well. To do a new job, build afresh rather than complicate old programs by adding new "features."

(*ii*) Expect the output of every program to become the input to another, as yet unknown, program. Don't clutter output with extraneous information. Avoid stringently columnar or binary input formats. Don't insist on interactive input.

(*iii*) Design and build software, even operating systems, to be tried early, ideally within weeks. Don't hesitate to throw away the clumsy parts and rebuild them.

(*iv*) Use tools in preference to unskilled help to lighten a

1902 THE BELL SYSTEM TECHNICAL JOURNAL, JULY-AUGUST 1978



Unix philosophy

- Make each program do one thing well. To do a new job, build afresh rather than complicate old programs by adding new “features”.
- Expect the output of every program to become the input to another, as yet unknown, program. Don’t clutter output with extraneous information. Avoid stringently columnar or binary input formats. Don’t insist on interactive input.
- Design and build software, even operating systems, to be tried early, ideally within weeks. Don’t hesitate to throw away the clumsy parts and rebuild them.
- Use tools in preference to unskilled help to lighten a programming task, even if you have to detour to build the tools and expect to throw some of them out after you’ve finished using them.

Unix philosophy

- Make each program do one thing well. To do a new job, build afresh rather than complicate old programs by adding new “features”.
- Expect the output of every program to become the input to another, as yet unknown, program. Don’t clutter output with extraneous information. Avoid stringently columnar or binary input formats. Don’t insist on interactive input.
- Design and build software, even operating systems, to be tried early, ideally within weeks. Don’t hesitate to throw away the clumsy parts and rebuild them.
- Use tools in preference to unskilled help to lighten a programming task, even if you have to detour to build the tools and expect to throw some of them out after you’ve finished using them.

Unix philosophy

- Make each program do one thing well. To do a new job, build afresh rather than complicate old programs by adding new “features”.
- Expect the output of every program to become the input to another, as yet unknown, program. Don’t clutter output with extraneous information. Avoid stringently columnar or binary input formats. Don’t insist on interactive input.
- Design and build software, even operating systems, to be tried early, ideally within weeks. Don’t hesitate to throw away the clumsy parts and rebuild them.
- Use tools in preference to unskilled help to lighten a programming task, even if you have to detour to build the tools and expect to throw some of them out after you’ve finished using them.

Unix philosophy

- Make each program do one thing well. To do a new job, build afresh rather than complicate old programs by adding new “features”.
- Expect the output of every program to become the input to another, as yet unknown, program. Don’t clutter output with extraneous information. Avoid stringently columnar or binary input formats. Don’t insist on interactive input.
- Design and build software, even operating systems, to be tried early, ideally within weeks. Don’t hesitate to throw away the clumsy parts and rebuild them.
- Use tools in preference to unskilled help to lighten a programming task, even if you have to detour to build the tools and expect to throw some of them out after you’ve finished using them.

Transparency and experimentation

- The input files to Unix commands are normally treated as immutable.
  - Run most commands without damaging the input files.
- You can end the pipeline at any point, pipe the output into less, and look at it to see if it has the expected form.
  - Great for debugging.
- You can write the output of one pipeline stage to a file and use that file as input to the next stage.
  - Restart process.

The biggest limitation of Unix tools is that they run only on a single machine — and that's where tools like Hadoop come in.

Today's topic: Batch Processing

- Overview
- IO & Unix pipes
- MapReduce
  - HDFS - infrastructure
  - Job execution
  - Programming models
  - Workflow
- Beyond MapReduce

Batch processing with Unix Tools

```
cat /var/log/nginx/access.log |  ❶
  awk '{print $7}' |             ❷
  sort             |             ❸
  uniq -c          |             ❹
  sort -r -n       |             ❺
  head -n 5                      ❻
```

```
4189 /favicon.ico
3631 /2013/05/24/improving-security-of-ssh-private-keys.html
2124 /2012/12/05/schema-evolution-in-avro-protocol-buffers-thrift.html
1369 /
 915 /css/typography.css
```

- Read the log file.
- Split each line into fields by white space, output only the 7th element (requested URL).
- Alphabetically sort
- Filter out repeated lines.
- Sort it again based on the line number (-n)
- Output the first five lines.

Today's topic: Batch Processing

- Overview
- IO & Unix pipes
- MapReduce
  - HDFS - infrastructure
  - Programming models
  - Job execution
  - Workflow
- Beyond MapReduce

Google Data Centers

- Dalles, Oregon
- Hydroelectric power @ 2¢ / KW Hr
- 50 Megawatts
- Enough to power 60,000 homes

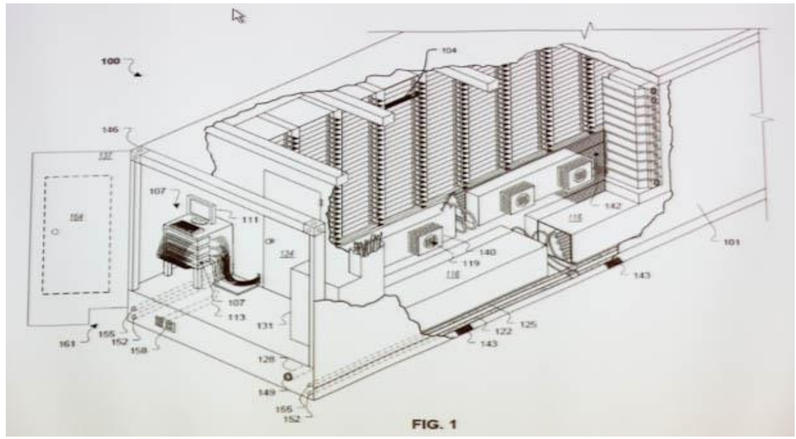
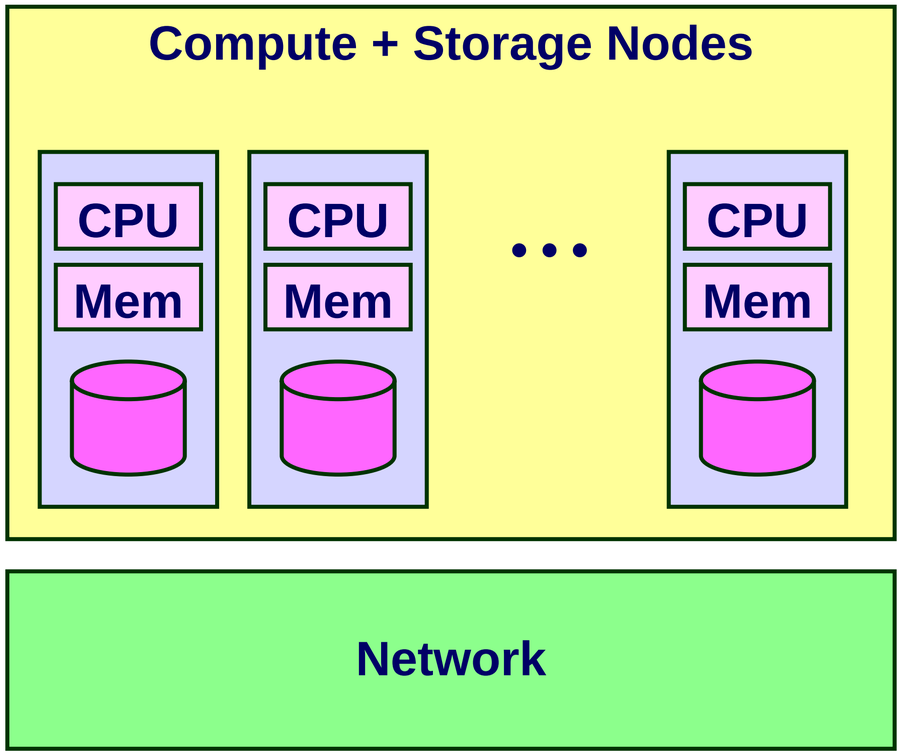


FIG. 1

- Engineered for maximum modularity & power efficiency
- Container: 1160 servers, 250KW
- Server: 2 disks, 2 processors

Typical Cluster Machine



Compute + Storage Nodes

- Medium-performance processors
- Modest memory
- 1-2 disks

Network

- Conventional Ethernet switches
  - 10 Gb/s within rack
  - 100 Gb/s across racks

Machines with Disks

Lots of storage for cheap

- 3 TB @ $150
  (5¢ / GB)
- Compare 2007:
  0.75 TB @ $266
  35¢ / GB

Roll over image to zoom in

WD Black 3TB Performance Desktop Hard Disk Drive - 7200 RPM SATA 6 Gb/s 64MB Cache 3.5 Inch - WD3003FZEX

by Western Digital

1,615 customer reviews | 28 answered questions

List Price: $249.99
Price: $149.99 Prime
You Save: $100.00 (40%)

In Stock.
Ships from and sold by Amazon.com. Gift-wrap available.

Want it Friday, Oct. 30? Order within **55 mins** and choose **Two-Day Shipping** at checkout. Details

Capacity: **3 TB**

1 TB	2 TB	3 TB	4 TB
$71.79 Prime	$116.99 Prime	$149.99 Prime	$198.00 Prime

5 TB	6 TB
$260.99	$298.91

Drawbacks

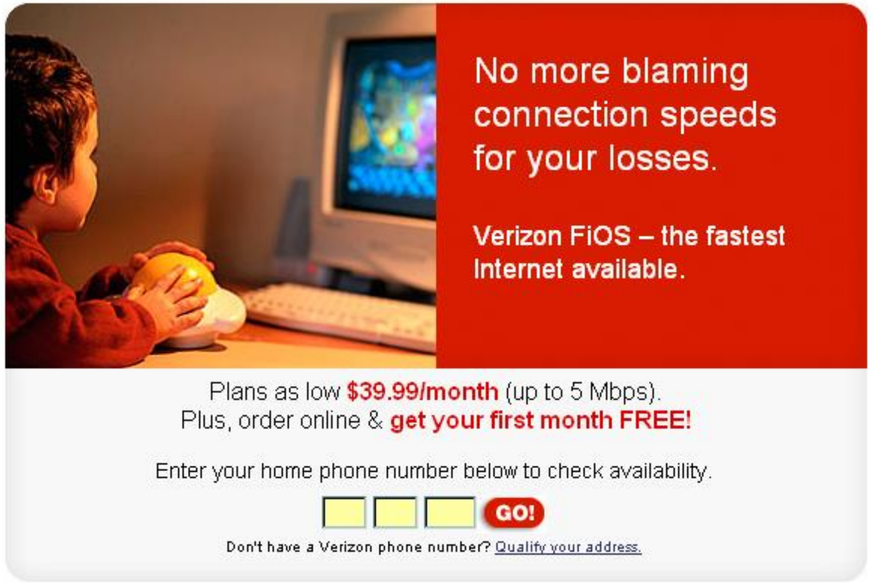
- Long and highly variable delays
- Not very reliable

Not included in HPC Nodes

Oceans of Data, Skinny Pipes

1 Terabyte

- Easy to store
- Hard to move



Seagate

Disks	MB / s	Time
Seagate Barracuda	115	2.3 hours
Seagate Cheetah	125	2.2 hours
Networks	**MB / s**	**Time**
Home Internet	< 0.625	> 18.5 days
Gigabit Ethernet	< 125	> 2.2 hours
PSC Teragrid Connection	< 3,750	> 4.4 minutes

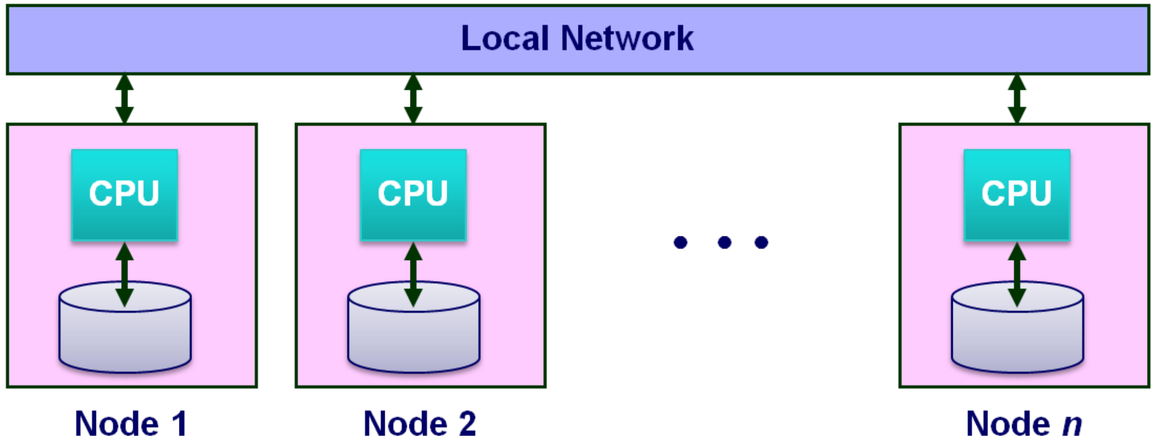
Data-Intensive System Challenge

For Computation That Accesses 1 TB in 5 minutes

- Data distributed over 100+ disks
  - Assuming uniform data partitioning
- Compute using 100+ processors
- Connected by gigabit Ethernet (or equivalent)

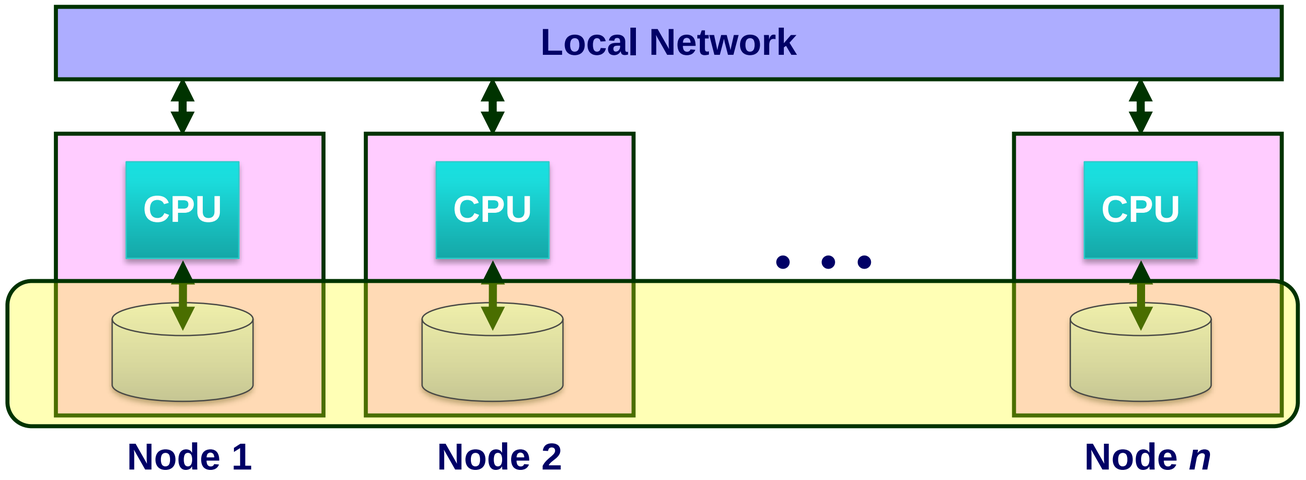
System Requirements

- Lots of disks
- Lots of processors
- Located in close proximity
  - Within reach of fast, local-area network



Hadoop Project

File system with files distributed across nodes



- Store multiple (typically 3 copies of each file)
  - If one node fails, data still available
- Logically, any node has access to any file
  - May need to fetch across network (ideally, leverage locality for perf.)

Map / Reduce programming environment

- Software manages execution of tasks on nodes