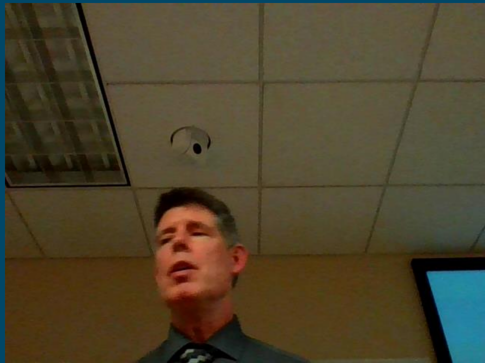
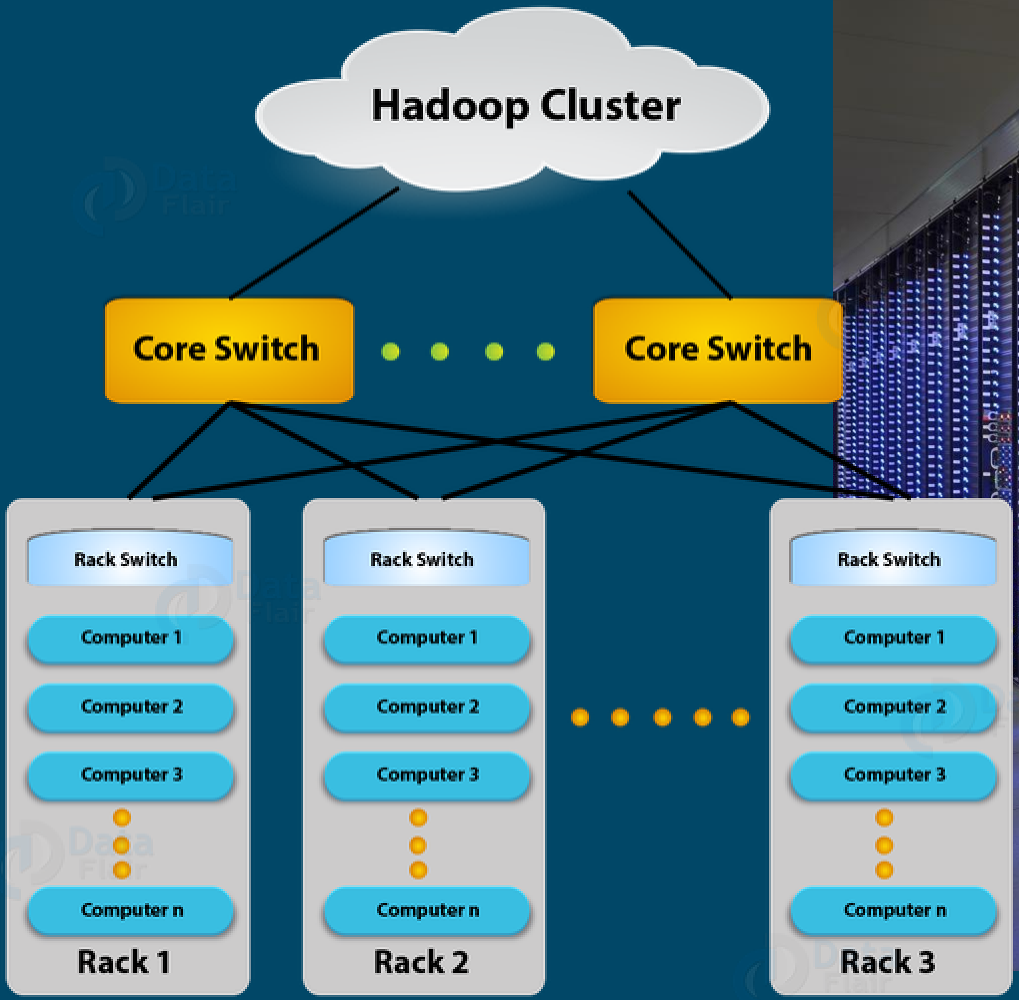
Cluster Networking Concepts

Bandwidth, latency, etc.

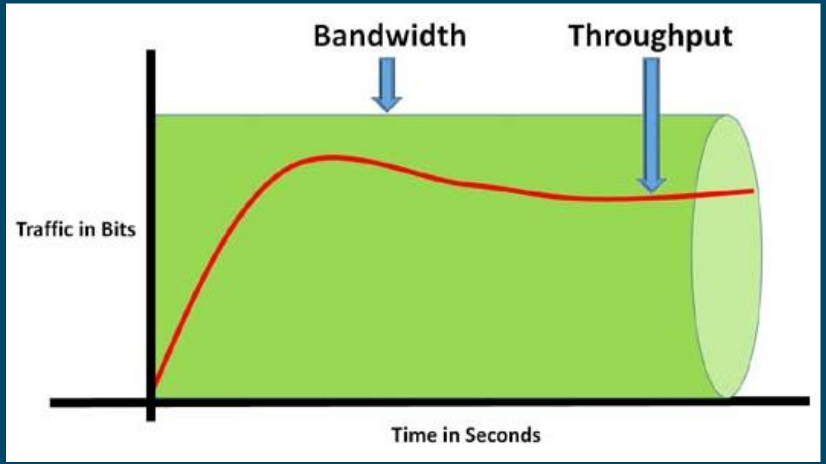


Computing cluster network

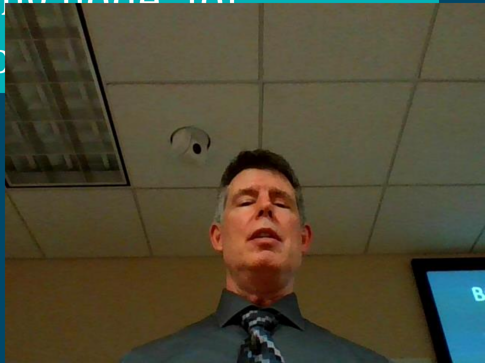


Bandwidth & throughput

- **Bandwidth** is the maximum amount of data that can be transmitted in a period. Measured as bits per second (bps)
- **Throughput** is the actual number of bits that flows through a network connection in a period
  - Can also measure job throughput—number of jobs processed per period
- Why a difference?
  - Contention is when two more processors attempt to access them same resource (e.g., network, disk drive, memory); called network congestion when due to heavy network usage
  - Resource processing limitation (CPU, RAM) of network devices

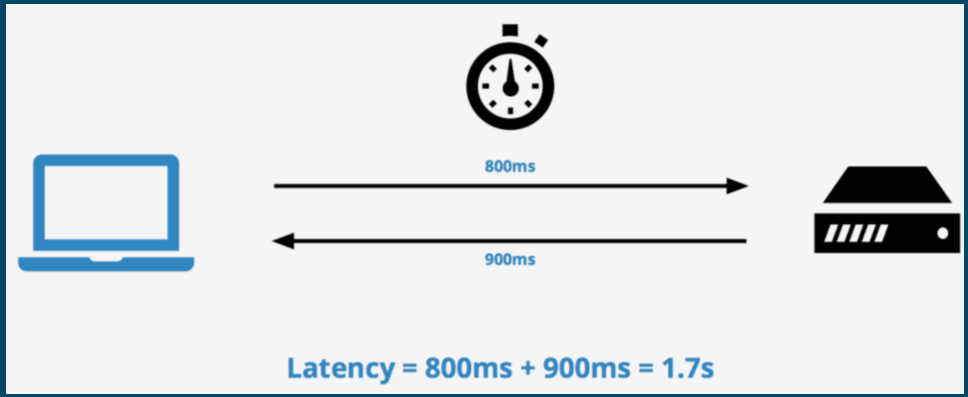


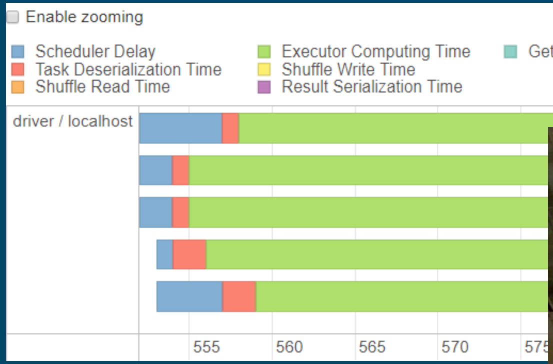
Cluster congestion often because of data transfer among node; for example, a large jo



Latency & response time

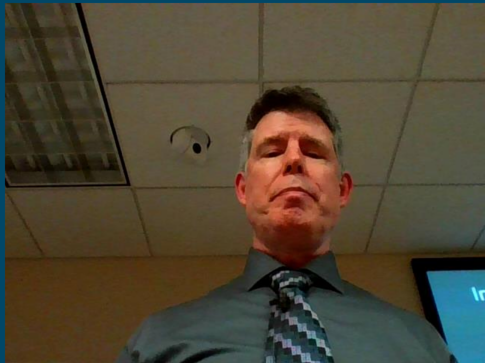
- **Network latency** is the time it takes for a packet of data to get from one designated point to another
- **Response time** is the amount of time it takes to complete a job from the time it is submitted
  - Response time = 1 / throughput
- Both increase with cluster contention (network & processing)





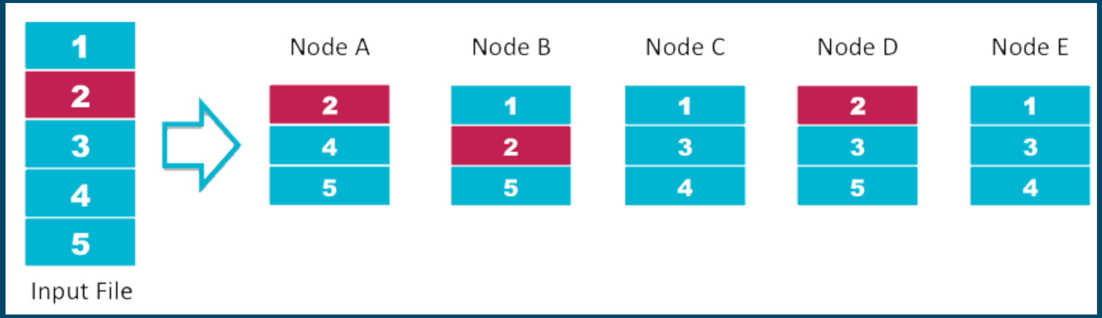
Increasing the cluster speed

- Vertical Scaling (**scale-up**)
  - adding more processors and RAM, buying a more expensive and robust server
- Horizontal scaling (**scale out**)
  - adding more nodes to a system

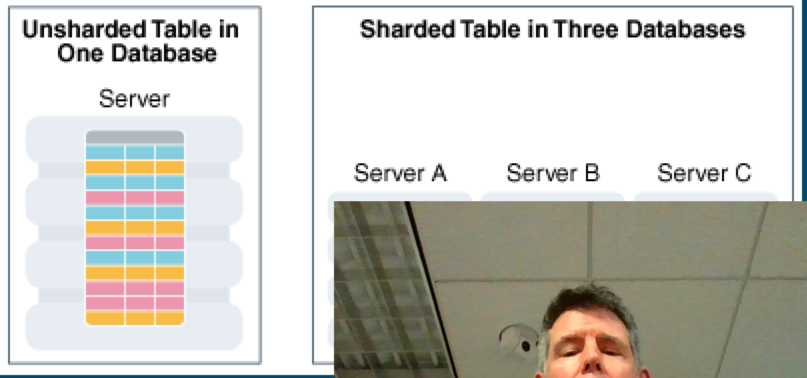


Increasing the data processing speed

- **Data Replication**
  - storing data in multiple nodes, improving availability, dependability,

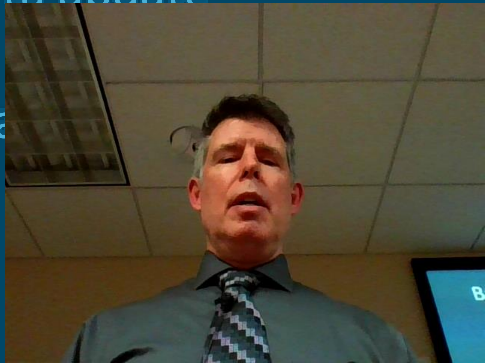


- **Database shard**
  - a partition of data (e.g., SQL rows), which may be stored on a node, improving availability, dependability, and processing time (through parallel processing)



Batch and Transaction Scaleup

- **Batch scaleup**
  - A single large job; typical of big data analysis and database
  - Use an *N*-times larger computer on *N*-times larger problem.
- **Transaction scaleup**
  - Numerous small queries submitted by independent users to a shared database; typical transaction processing and timesharing systems.
  - *N*-times as many users submitting requests (hence, *N*-times as many requests) to an *N*-times larger database, on an *N*-times larger computer.
- Big Data processing (Spark / Hadoop) is better with batch processing
  - Well-suited to parallel execution, but performs poorly with many clients and data update (and only recently addresses ACID transactions)
  - Note: batch process does allow for iterative algorithms, like K-means, which Spa (while Hadoop does not)



Factors limiting speedup

- Linear speed-up
  - Each added node decreases the processing time by one unit
- Speedup is often sublinear due to:
  - **Startup costs**
    - Node startup take time, which delays job computation
  - **Contention**
    - Nodes contend for shared resources (network, CPUs), and thus wait for processes to complete
  - **Skew**
    - Non-uniform data distribution (partitions) causes some tasks to com before others. The job is complete when the slowest task ends

Hadoop overhead example on small query

```
00:45:43,041 - Parsing command: select 'A' from dual where 1=1
00:45:44,184 - Starting command: select 'A' from dual where 1=1
00:45:45,232 - Connecting to ResourceManager (by client)
00:45:48,459 - Submitted application
00:45:52,148 - Created MRAppMaster
00:45:55,742 - Connecting to ResourceManager (by AM)
00:45:58,184 - ContainerLauncher - CONTAINER_REMOTE_LAUNCH
00:45:58,246 - Transitioned from ASSIGNED to RUNNING
00:46:01,195 - JVM given task
00:46:04,181 - Progress of TaskAttempt is : 0.0
00:46:04,595 - Progress of TaskAttempt is : 1.0
00:46:04,677 - Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 2.85 sec
00:46:06,820 - Ended Job
Time taken: 23.8 seconds, Fetched: 1 row(s)
```

- 75% Overhead!
  - 1 second to parse the query
  - 9 seconds to submitting the query and launching ApplicationMaster (00:45:43 – 00:45:52)
  - 6 seconds to initialize and launch the container for Map task (00:45:52 – 00:45:58)
  - 3 seconds to initialize JVM (00:45:58 – 00:46:01)
  - 6 seconds for actual M and cleanup (00:46:01
    - 6/23.8 = 25%

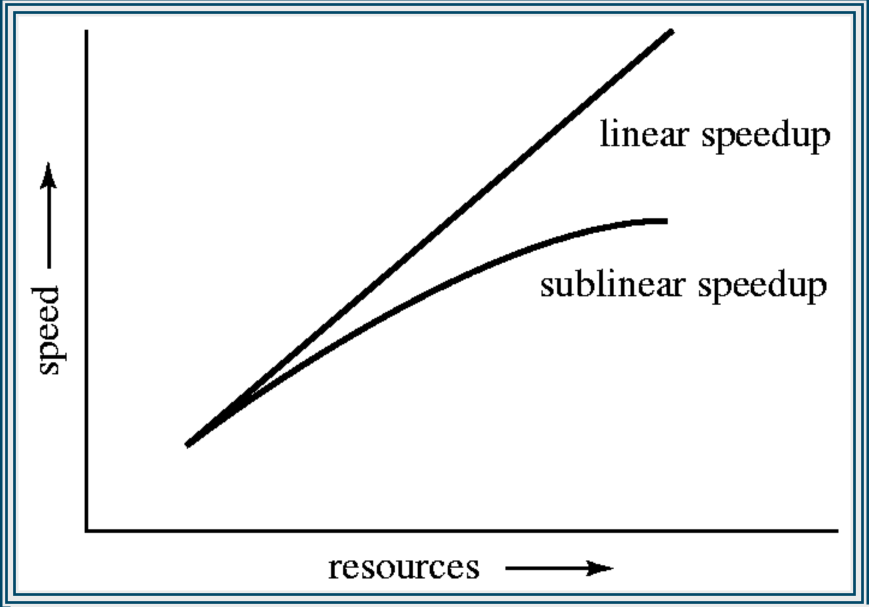
Parallel system performance measures

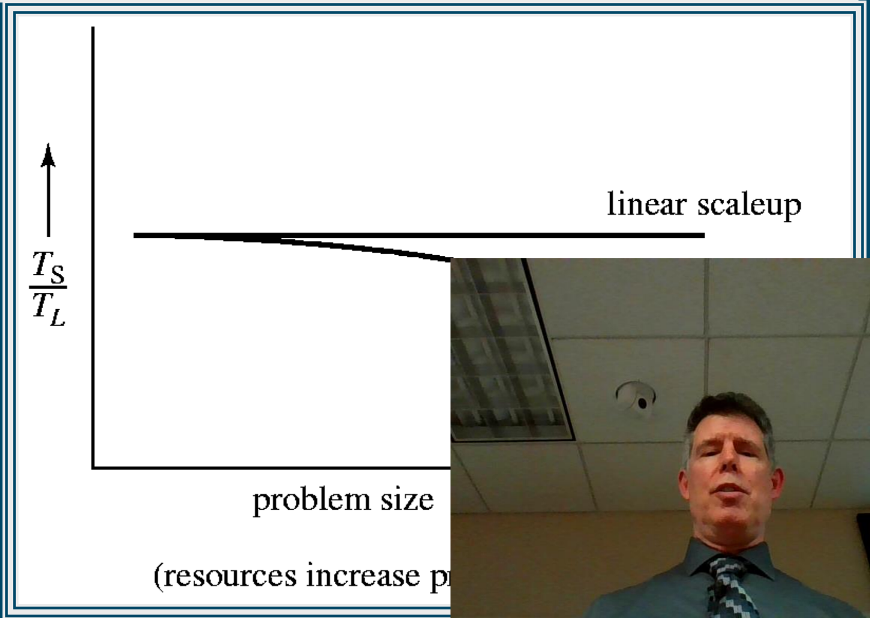
- **Speedup**
  - *small system elapsed time* / *large system elapsed time*   *100 min* / *50 min* = 2X

    linear

- **Scaleup**
  - Do things "at scale". Linear scaleup = 1
    - N-times larger system to perform n-times larger job
  - small system small problem elapsed time / big system big problem elapsed time
  - 10 CPU for 10 G data: 10 min / 100 CPU for 100G data: 11 min = 0.91 scaleup

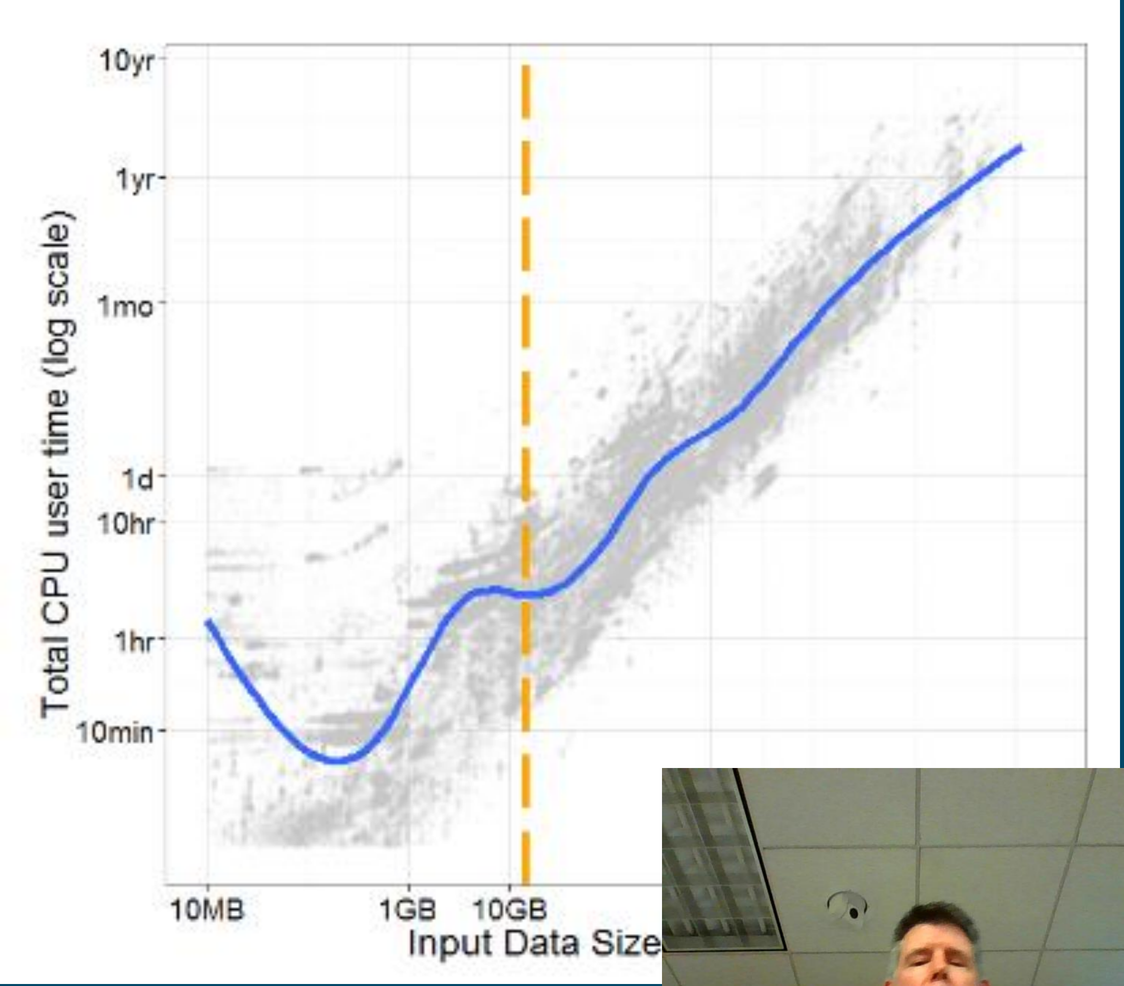
    sub-linear





Clusters don’t provide linear improvement

- Cluster study
  - MapReduce 32 cores, 512 GB
  - Initially, as job size increases time decreases
  - Eventually, time increases with job size, with slope sub-linear



Appuswamy, R., Gkantsidis, C., Narayanan, D., Hodson, O., & Rowstron, A. (2013, October). Scale-up vs scale-out for hadoop: Time to rethink?. In Proceedings of the 4th annual S
Computing (pp. 1-13).

Important to remember

- Know networking concepts
  - Bandwidth, throughput, contention, latency, response time, scale up/out, shard, replication
- Factors limiting linear improvement
  - Start up, contention, skew
- Clusters don’t provide linear improvement