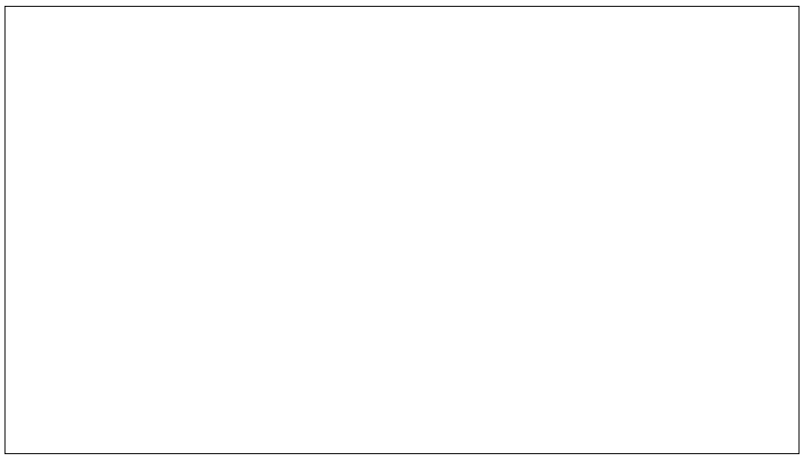
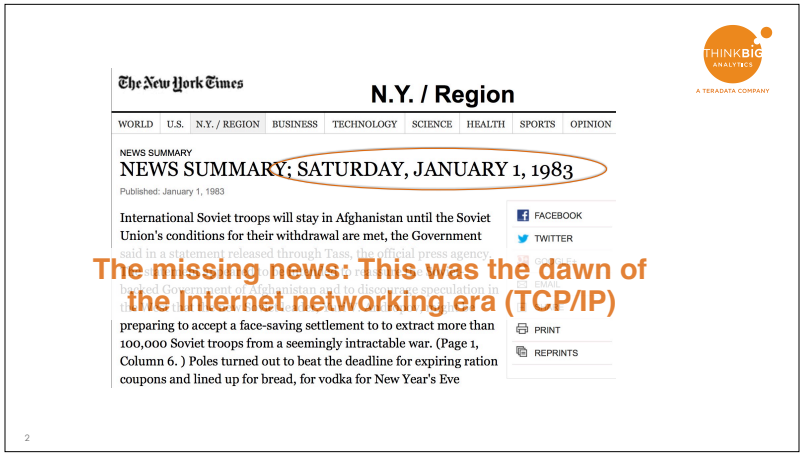
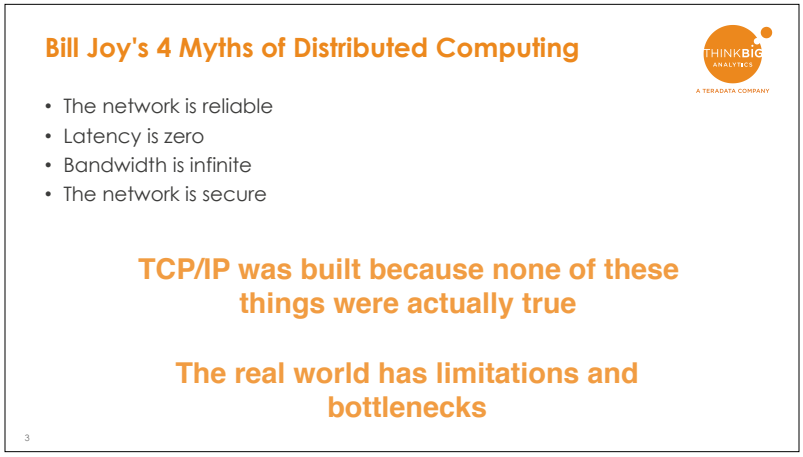


This is intentionally a blank slide. It provides us an opportunity to ask a question to kick off the presentation.

What was the importance of January 1, 1983? Do you know?

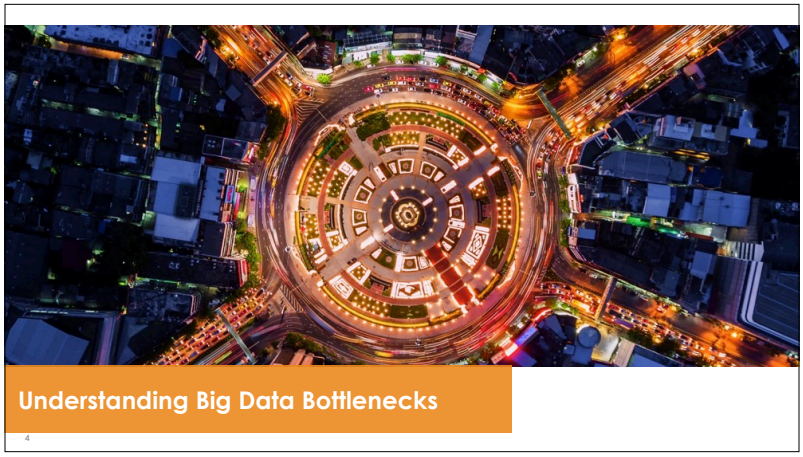


It was the dawn of the Internet networking era. It was when the ARPAnet was split from the MILnet, and when all the host computers were required to use a new protocol: TCP/IP to communicate. TCP/IP was a reinvention of networking that refuted Bill Joy's famous 4 myths (next slide)



Bill Joy's 4 myths were actually the guiding rules for networks before the 1980s. Prior to this, architects assumed that was that a sufficiently smart network could take care of all the errors might occur (i.e., the network was reliable), that they could ignore the latency of the network, they could assume they'd have all the bandwidth they needed, and they didn't need to worry about security because the network was "sealed off" from the bad things in the world.

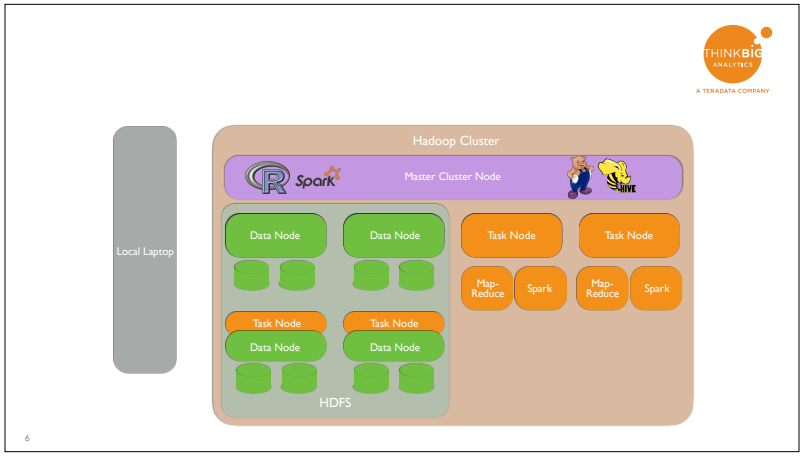
The reality of the world is that none of these things are true of internets. TCP/IP was built to deal with real networks, not ideal ones. The real world has limitations and bottlenecks.



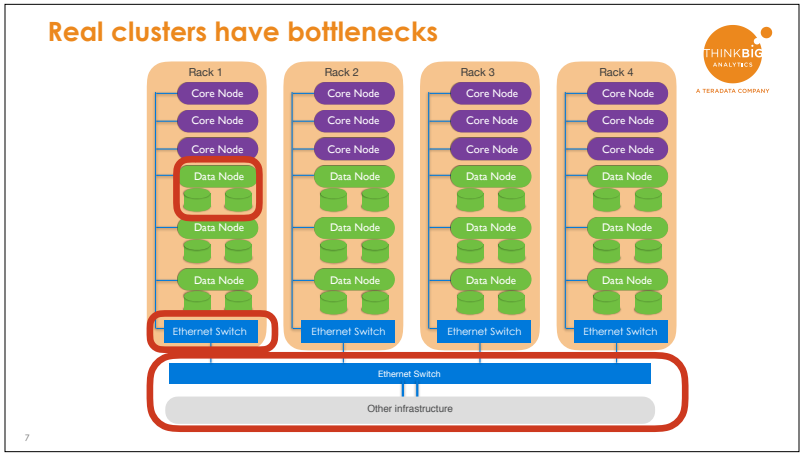
The same is true of Big Data. Hadoop exists in a real world of computing. The real world has limitations and bottlenecks. We need to understand these to be successful architects and developers.

Bottlenecks define the performance of clusters

- Every component of a cluster has performance limitations
- Your slowest components define what you can do
- Understanding speed bottlenecks can spell the difference between successful Hadoop projects and failed ones



Remember this diagram? This is a logical architecture of a cluster. However, there is also a physical architecture that is more complex.



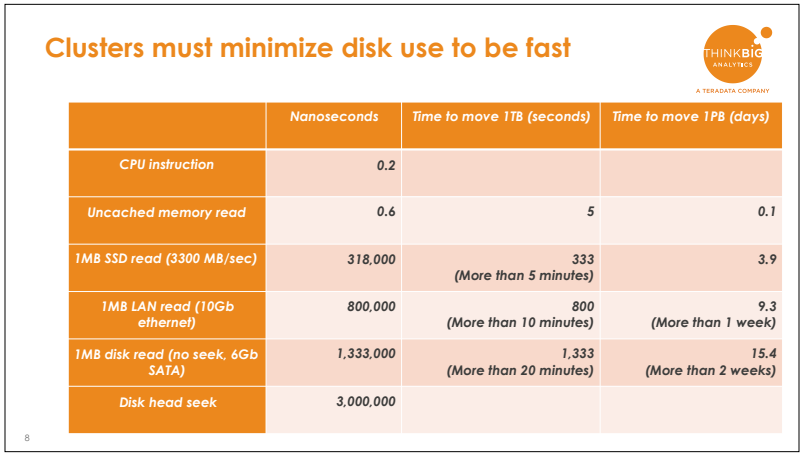
This is how a real cluster works.

We have core nodes <click>
and we have data nodes <click>
Those are tied together by an Ethernet <click>
And the nodes are all put in a rack <click>
The racks are then replicated <click>
And the racks are networked together by another larger switch which connects it to the rest of the infrastructure <click>

Now where are the bottlenecks?

Well, the obvious one is the disk. <click>
However, less obvious is the ethernet switch in the rack <click>
And most neglected of all is the ethernet switch that connects to the rest of the infrastructure. <click> Often it can be very difficult to find out what the bandwidth of that link is because it sometimes is managed by a different organization (networking versus IT).

So how can we sort out the effects of all these possible bottlenecks?



	Nanoseconds	Time to move 1TB (seconds)	Time to move 1PB (days)
CPU instruction	0.2		
Uncached memory read	0.6	5	0.1
1MB SSD read (3300 MB/sec)	318,000	333 (More than 5 minutes)	3.9
1MB LAN read (10Gb ethernet)	800,000	800 (More than 10 minutes)	9.3 (More than 1 week)
1MB disk read (no seek, 6Gb SATA)	1,333,000	1,333 (More than 20 minutes)	15.4 (More than 2 weeks)
Disk head seek	3,000,000		

Here are some constants that every programmer should know. We're going to begin by assuming a 5 GHz CPU, where each instruction takes 0.2 nanoseconds or 200 picoseconds. <click>

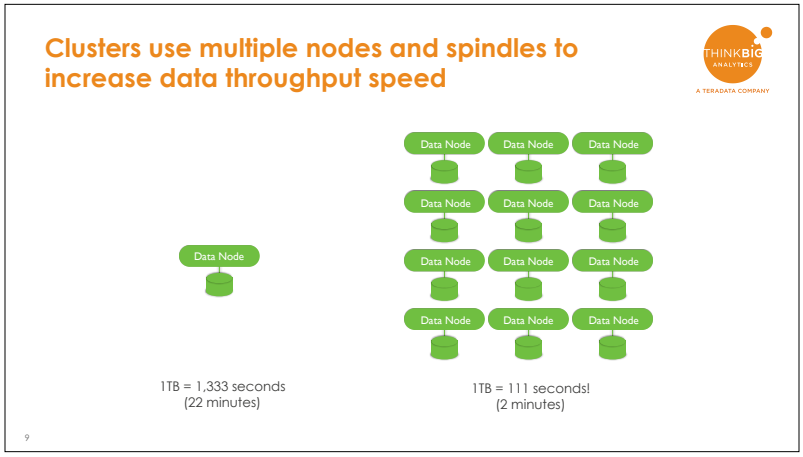
Unfortunately, our memory doesn't run as fast as our processor. It takes 0.6 nanoseconds to deliver a row of data. That means that it takes 5 seconds to read a Terabyte out of memory and more than 2 hours—about 0.1 days— to read a petabyte. And that's using RAM. <click>

Let's look at Solid State Drives or SSDs. Those are fast, right? Well, sort of. Your typical enterprise SSD reads data at about 3,300 MB/second. That means it takes 333 minutes—that's more than 5 minutes—to read a Terabyte from an SSD. It also means it takes nearly 4 days to read a Petabyte. Hmmm.

We spoke about Ethernet being a possible bottleneck earlier. A 1MB read on 10Gb Ethernet takes about 800 microseconds. That means it takes 800 seconds to read a TByte and more than a week to read a Petabyte over 10Gb Ethernet.

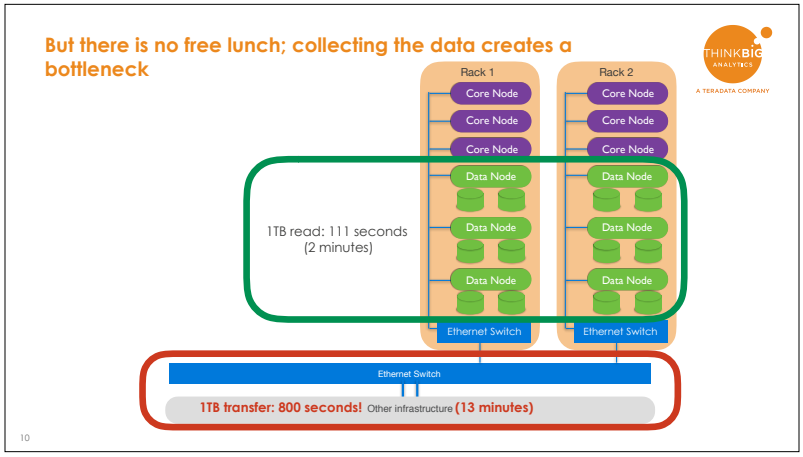
Disks are even slower, with our one Terabyte read taking more than 20 minutes and a PByte read taking more than 2 weeks.

And all of this assumes that we have no disk seeks. Disks impose even more delay to the process. It takes about 3 milliseconds to move a disk head. Suffice it to say, that seems like a

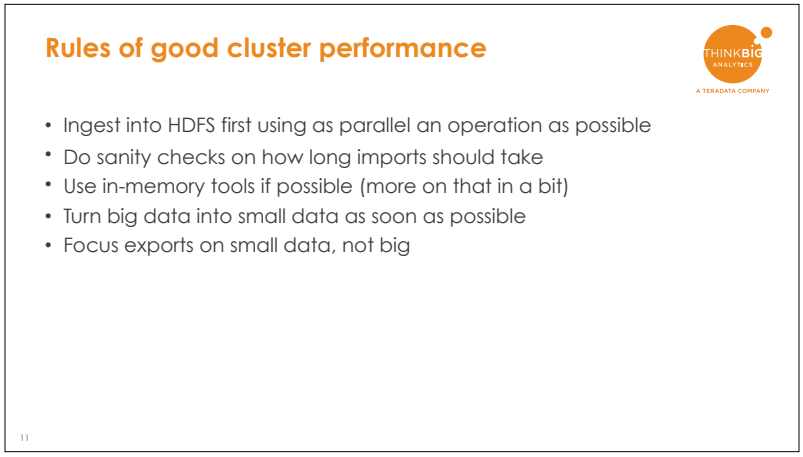


We should be able to speed up our terabyte read by sharing the data across multiple disks and then reading from multiple spindles at once. So if a single node takes 22 minutes to read a terabyte, it should only require 2 minutes for 12 data nodes to read that same terabyte (assuming it was appropriately distributed across those disks).

However....



Collecting that data, however, will require sending the information over the ethernet switch. Unfortunately, when we calculate how long that will take, that increases our time to read our terabyte back up to 13 minutes. Our bottleneck has moved from the disk to the ethernet.



Bottlenecks will define how fast our cluster programs will run. But there are some ways we can avoid those bottlenecks.

Ingest into HDFS first using as parallel an operation as possible. Use tools like hadoop df distcp, sqoop, and tdch to ingest data, not single-node tools like hdfs dfs get.
Do sanity checks on how long imports should take. If someone says they are ingesting a petabyte in a few hours, don't be afraid to ask questions about how they are doing that. Petabytes are more about days than minutes.
Use in-memory tools if possible (more on that in a bit). Memory is many orders of magnitude faster than disk. It would be awesome if we had a tool that focused on keeping data in memory instead of disk. And Spark is just that tool.
Turn big data into small data as soon as possible. The best way to make a program run fast is to filter or sample your data first before processing it. Very few algorithms work significantly better with a petabyte than with a gigabyte. Don't be shy about filtering and sampling.
Focus exports on small data, not big. Hadoop isn't about doing ETL, but really about finding the analytical needle in the Big Data haystack. The art of data science is turning Big Data into insight, which if it's going to be used for decision-making, has to be small. Do that in the cluster before you export your data.