

Computing Techniques for Big Data



IEEE

June 20, 2013

© Gayn B. Winters, Ph.D.

gaynwinters@ieee.org

www.convergeio.com

www.xreholdings.com

www.computer.org/cc

Hadoop was created by Doug Cutting who named it after his son's toy elephant. Hadoop and the logo above are owned and maintained by the Apache Software Foundation. For info on "free" licensing and code, see www.apache.org .

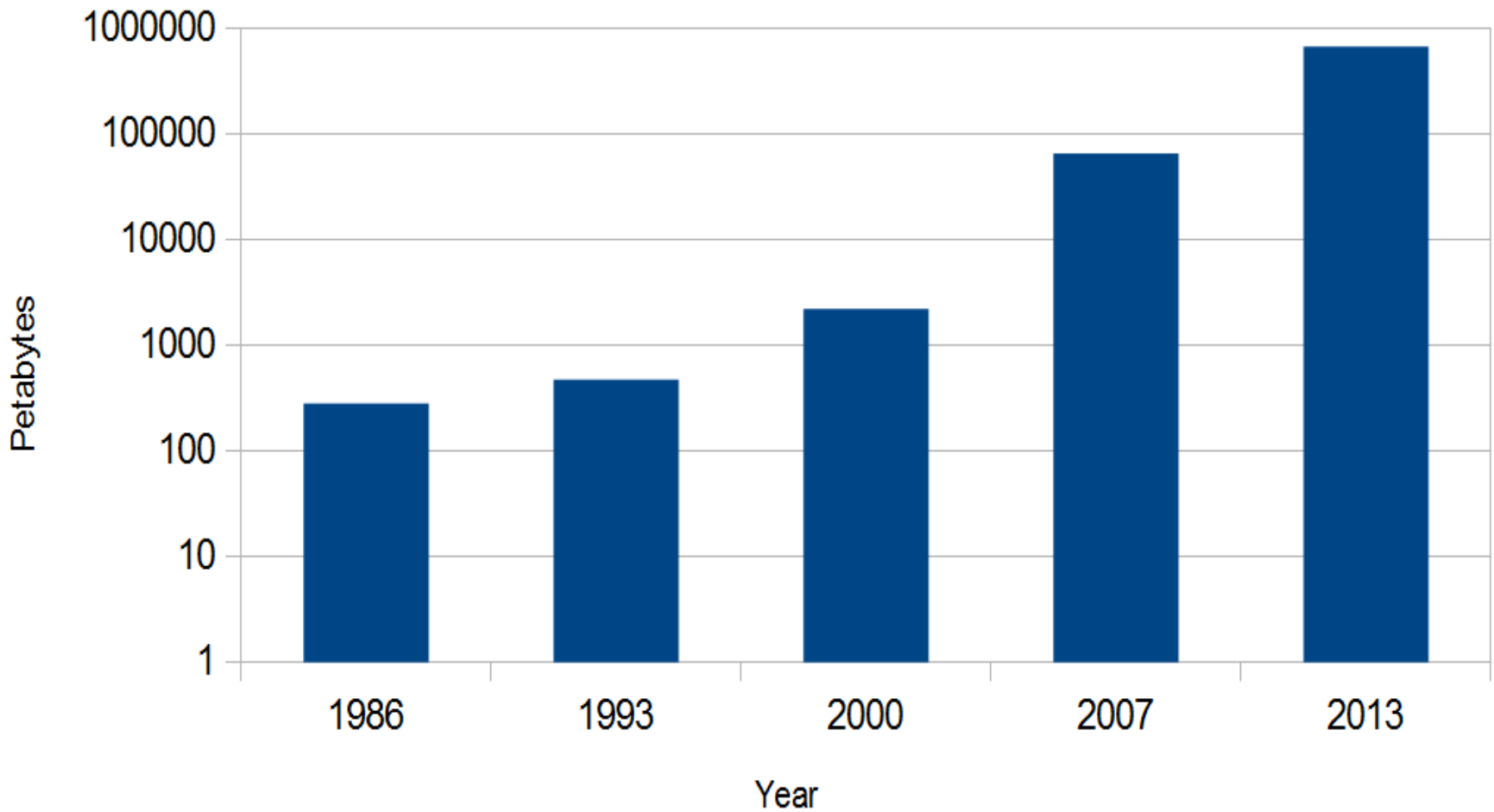
Multiples of bytes

- kilobyte (KB) 10^3 ← **Actually 1024**
- megabyte (MB) 10^6 **1024²**
- gigabyte (GB) 10^9 **1024³, etc.**
- terabyte (TB) 10^{12}
- petabyte (PB) 10^{15} ← **Library of
Congress
~ 10 PB**
- exabyte (EB) 10^{18}
- zettabyte (ZB) 10^{21} ← **1024⁷**
- yottabyte (YB) 10^{24} ← **=1.2*10²⁴**

Big Data

- “Big Data” = Exceeds in quantity, transfer speed, and/or data type variety. (3V's)
- Big Data worth \$100B growing 10% per year
- As of June 2012, ~2.5 Exabytes (10^{18}) of business data are created DAILY
- Twitter receives 12TB of data a day
- Modern airline eng: 20TB/hr performance data
- NASA Sq. Kilometer Array: 100's of TB/sec = 10's of Exabytes/day
- Machines dominate data generation!!!

Internet Transmission Growth



Moving Data to the Cloud will make this transmission ramp even steeper!

What do you do ...?

(Tipping Points)

- When your data size exceeds what can be put on or handled by one machine?
- When your daily calculations take longer than a day?
- When your fancy compute systems have terrible ROA? E.g. 10% utilization
- When the Availability (= %time fully functional) of your computer system is unacceptable?



BIG DATA



— GOT IT...

NOW WHAT?

Cloud Computing

- Elastically and Independently provision:
 - Compute resources
 - Storage
 - Network capacity: access and internal
 - Resource Pooling and Load Balancing
- Only pay for what you use; includes transmission, maintenance, etc.
- Security and Reliability “Guaranteed”, but ...
- Vendors: Amazon, IBM, Google, Rackspace, Microsoft, HP...

Cloud Computing NIST Definitions

- IaaS = Infrastructure as a Service
- PaaS = Platform as a Service
- SaaS = Software as a service
-
-
- Problem: These boundaries are blurring, and vendors are moving around. Some vendors just don't fit any of these definitions.

Private and Internal Clouds

- Use open source infrastructure (Eucalyptus, OpenNebula, Nimbula, OpenStack, Platform, Enomaly, Deltacloud, KVM)
- House computers on client site
- Increases asset utilization
- Market is growing!



Private Cloud Advantages

- Cost Reduction (viewed via charge-back)
- Efficiency (ROA) via virtualization, thin provisioning, snapshots, deduplication, backup
- Security and Regulatory Conformance
- Performance
 - Base CPU, memory, speed to storage
 - High IO/s
 - Low latency
 - High Availability

Hybrid Cloud Advantages

- Increased Flexibility
 - e.g. Intel will never put IP on a public cloud, but runs many applications there
 - Excellent for load balancing when loads peak.
- Cost balancing
 - Put peaky applications on public cloud
 - Put controlled applications on private part
- Product development
 - Public: non production development and testing
 - Private: controlled production, private benefits

Public vs. Private Cloud Costs

- Public cloud vendors add profit margin
- Public cloud vendors benefit from economies of scale
 - Equipment: volume discounts
 - Energy: can negotiate discounts and can locate their data centers where energy costs are low and air conditioning isn't much needed.
- Public clouds get better equipment utilization from better load leveling.

Vendors of “Private Cloud in a Box”

- HP CloudSystem
- Dell Virtual Integrated Systems
- Cisco's Cloud Infrastructure
- IBM's CloudBurst
- BMC Cloud Lifecycle Management (sw only)
- Coud.com's CloudStack (sw only)
- Microsoft's Hyper-V (sw only)
- VMware's vCloud. (sw only)

New Cloud Vendors

What's the Message?

- Tier3 (mix)
- Nutanix (Storage)
- DataCore (Storage)
- CloudByte (Storage)
- ScaleIO (Storage) (*)
- Exablox (Storage)
- SolidFire (SSDs)
- Veeam (Backup)
- DataON (Box Cluster)
- SoftLayer (IaaS) (*)
- Savvis (IaaS) (*)
- Permabit (Tools)
- Pure (SSDs)
- Virsto (Storage)

Cloud Workload Elasticity

- Seasonal (retail, sports, tax, school)
- Batch (resource intensive: payroll, billing, HPC, order processing, backup)
- Mixed/complex/random (analytics, news, marketing)
- Rapid growth (exponential or hockey stick: difficult to provision)
- Transient (workload goes to zero at times: engineering, QA, training, disaster recovery)
- Non-conforming (doesn't fit billing models)

Low Latency Applications

- Trading, esp. arbitrage
 - Info Week: “A 1 millisecond advantage in trading can be worth \$100 Million per year.”
- broadcasting market data, publishing volatilities, reconfiguring connections and performing other time-critical tasks in trading
- Connecting news to trades. Weather forecast in Brazil affect futures' price of coffee.
- Telecom voice and video quality
- Droop: handshake slow speed slowing down throughput on fiber channels.

Latency in Fiber Optics

- Light in a vacuum, 3.33 microseconds per km.
- Light in fiber, ~5.0 microseconds per km.
- LA to NYC 2462 mi (3961 km), but driving ~fiber is 2778 (4469 km)
- Light LA to NYC = 22.3 milliseconds + relay
- Light NYC to Bermuda = (1245km) 6.2 millisecs

Latency Requirements

- Trading Floor – current speeds
- Switch hardware
 - Zeptonics, a Sydney-based financial technology firm, said clients had begun testing its trading switch, which routes messages inside high-speed data centers in 130 nanoseconds.
 - Cisco Nexus 3016 switch can get the last number or letter of a 1,024-character message onto a wire in 1.17 microseconds, or millionths of a second. The last number or letter of a 256-character message makes it onto the wire in 950 nanoseconds, or billionths of a second.



Moving to Another Cloud Tough Problem

- Both need standard VM image formats
- Deal with different cloud APIs
- Moving data: formats, frequency of exchange
- Networking differences: DNS, load balancing, database configurations,
-
- IEEE standards may help ... eventually

Backup: Tape vs. D2D

- 2013: cost 1.5 TB LTO-5 tape about \$50
- 2013: cost 4.0 TB disk about \$80
- Even if $\text{cost}(\text{TB of disk}) < \text{cost}(\text{TB of tape})$ there are other expenses:
 - Backup servers and Networking (more for disk)
 - Storage Management, Backup Software, Labor, training, processes (replace for disk)
- RAID makes disk more reliable than tape
 - Increases cost
 - Erasure codes equalize reliability
- Tape problems: time, reliability, aging, etc.
- Tape utilization going down $< 10\%/year$

Backup: Cloud?

- T1 line: 1.5Mbps = 127 Gb/day
- Moving 1TB = 1024×10 Gb takes 80.6 days
- High Speed Cable 10 Mbps = 847 Gb/day and moving 1TB takes 12.1 days.
- De-duplication and compression will help by factors 8:1 and 22:1. Encryption will make things worse, however.

CRN's Coolest 100 Cloud Vendors 2013

- **20 Platform and Development** (AppFog; Cloudera, Cloud Foundry; MapR, Hortonworks, RedHat...)
- **20 Storage and Data Center** (Apple, Carbonite, Box, BitCasa, Dropbox, EMC, Hitachi, VMware...)
- **20 Infrastructure** (Amazon, AT&T, Cisco, Dell, Eucalyptus, HP, Rackspace, IBM, Verizon, CA, Joyent, Nebula, Google...)
- **20 Applications & Software** (Citrix, Google, Intuit, Microsoft, Oracle, Quest, Salesforce.com, SAP, Taleo...)
- **20 Security** (CA, Check Point, IBM, McAfee, NTT, Symantec, Trend Micro, CipherCloud, ...)

And, ..., many cool startups...

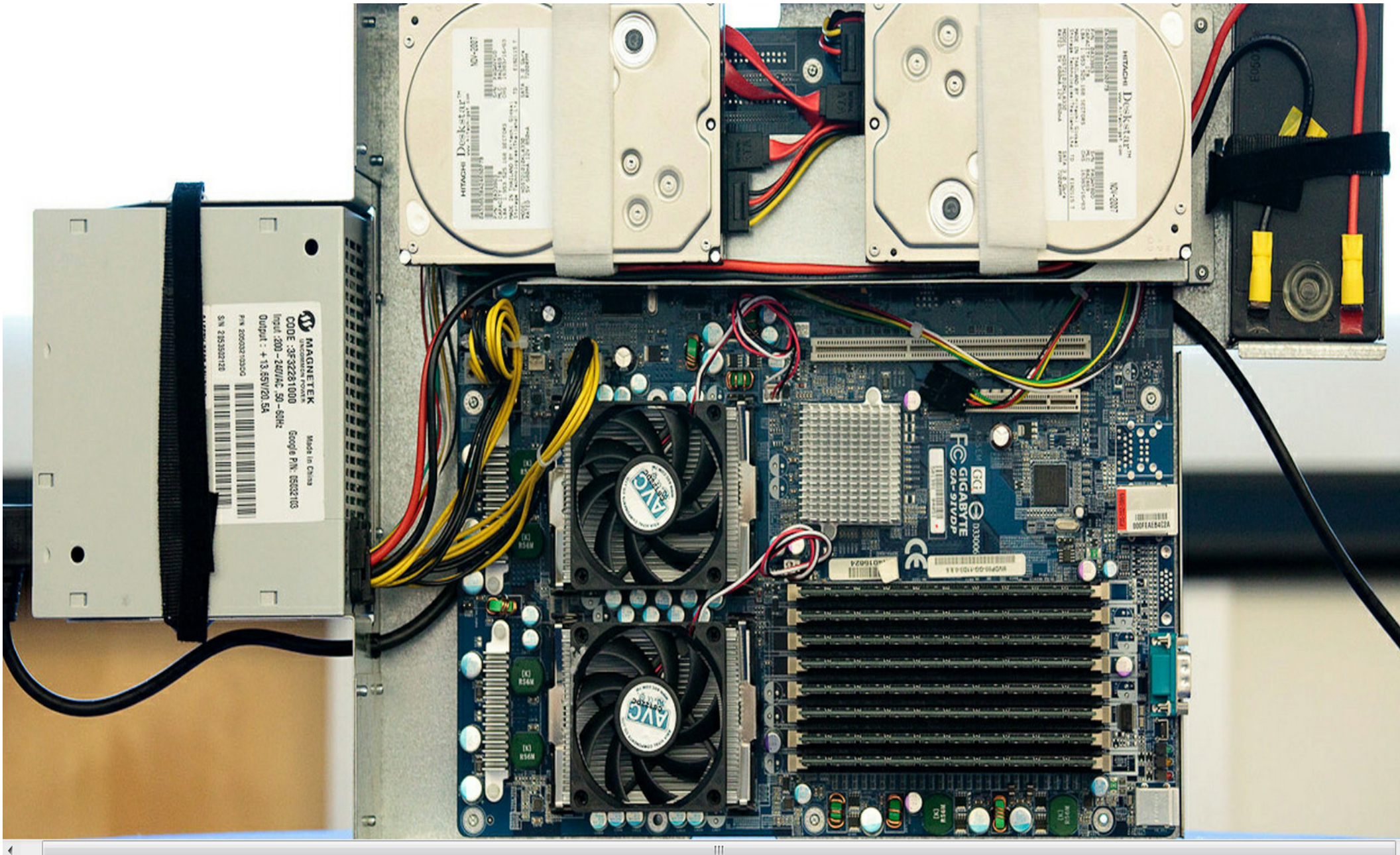
Cloud Consulting Ideas

- Key Problem: Pro's and Con's of moving to the cloud. Cost Analysis. Private vs. Public.
- Choice of vendor(s) – match business models
- Are Availability requirements met? 99.9%?
- Avoiding platform lock-in (IEEE standards)
- Transition challenges and costs
- Future Big Data needs? If yes, ask many questions, e.g. can you run on real rather than virtual machines?
- Performance Issues – IOPS, Latency, etc.

Data Center Terms for Big Data

- Rack mounted server (2-4 core 8GHz CPU, 16-32 GB DRAM, 2-4 1-4TB Disks)
- Rack (40-80 servers, 1 40/10 GB/s Ethernet Switch)
- Cluster (30-40 Racks, 1 100/40 GB/s Switch)
- Data Center (100's of Clusters, large routers)
 - Google has interesting patents + 1 data center description
 - Facebook has published their “green” data center specifications and designs

Google's Server



Google Server Backside

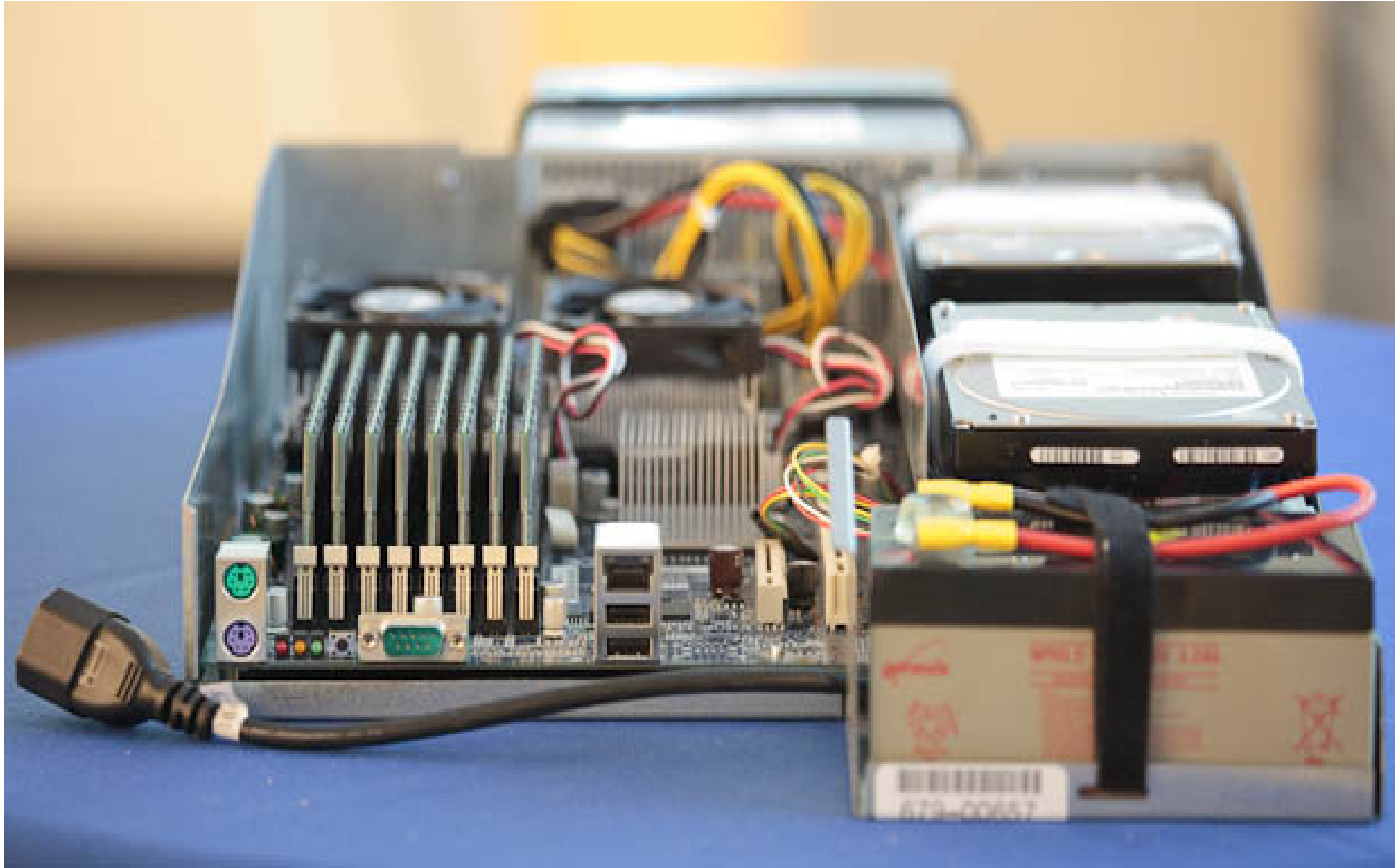


Photo by Stephen Shankland of CNET

Google Server Details

- 2 CPUs
- 2 Hard Drives
- Custom 12v Power Supply
 - Low cost, peak capacity, high efficiency
- Gigabyte MB converts 12v to lower
- 12v Battery Backup (99.9% efficiency vs 95%)
- 8 sticks DRAM
- Ethernet, USB, and other ports
- 2U = 3.5" high

Google's data center at The Dalles, OR



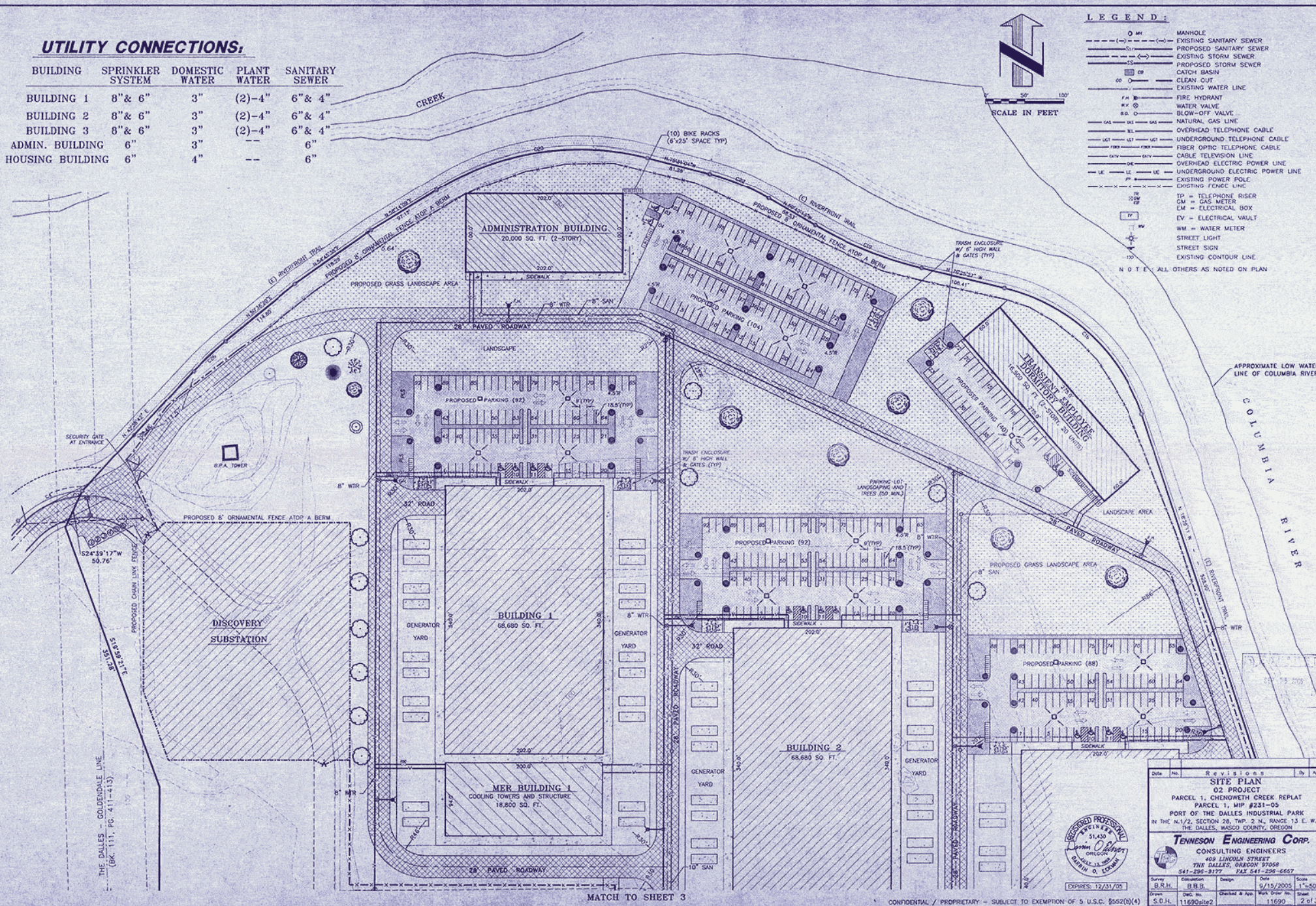
UTILITY CONNECTIONS.

BUILDING	SPRINKLER SYSTEM	DOMESTIC WATER	PLANT WATER	SANITARY SEWER
BUILDING 1	8" & 6"	3"	(2)-4"	6" & 4"
BUILDING 2	8" & 6"	3"	(2)-4"	6" & 4"
BUILDING 3	8" & 6"	3"	(2)-4"	6" & 4"
ADMIN. BUILDING	6"	3"	--	6"
HOUSING BUILDING	6"	4"	--	6"

LEGEND:

- MH MANHOLE
- EXISTING SANITARY SEWER
- PROPOSED SANITARY SEWER
- EXISTING STORM SEWER
- PROPOSED STORM SEWER
- CB CATCH BASIN
- CLEAN OUT
- EXISTING WATER LINE
- F.H. FIRE HYDRANT
- W.V. WATER VALVE
- B.O.V. BLOW-OFF VALVE
- NATURAL GAS LINE
- TEL OVERHEAD TELEPHONE CABLE
- UT UNDERGROUND TELEPHONE CABLE
- FOC FIBER OPTIC TELEPHONE CABLE
- CABLE TELEVISION LINE
- OVERHEAD ELECTRIC POWER LINE
- UNDERGROUND ELECTRIC POWER LINE
- EXISTING POWER POLE
- EXISTING FENCE LINE
- TP TELEPHONE RISER
- GM GAS METER
- EM ELECTRICAL BOX
- EV ELECTRICAL VAULT
- WM WATER METER
- SL STREET LIGHT
- SS STREET SIGN
- EXISTING CONTOUR LINE

NOTE: ALL OTHERS AS NOTED ON PLAN



MATCH TO SHEET 3

Date	Rev.	REVISIONS	By	App.
		SITE PLAN		
		02 PROJECT		
		PARCEL 1, CHENOWETH CREEK REPLAT		
		PARCEL 1, MIP #231-05		
		PORT OF THE DALLES INDUSTRIAL PARK		
		IN THE N1/2, SECTION 28, TWP. 2 N., RANGE 13 E. W.M.		
		THE DALLES, WASCOCO COUNTY, OREGON		
TENNESON ENGINEERING CORP.				
CONSULTING ENGINEERS				
409 LINCOLN STREET				
THE DALLES, OREGON 97058				
541-296-9177 FAX 541-296-6657				
Survey	Calculation	Design	Date	Scale
B.R.H.	B.R.H.	Checked & App.	5/15/2005	1"=50'
Drawn	Dist. No.	Sheet	Work Order No.	Sheet
S.D.H.	11690a2	11690	2 of 3	

EXPIRES: 12/31/05

CONFIDENTIAL / PROPRIETARY - SUBJECT TO EXEMPTION OF 5 U.S.C. §552(b)(4)

NSA Data Center, Bluffdale, Utah



NSA's Bumblehive

- <http://nsa.gov1.info/utah-data-center/>
- 1-1.5 million square feet, \$1.2B, ~5 zettabytes
- Water treatment, 60 diesel generators (3 day fuel supply) for 100% backup, 65 megawatts, 1.5 million gal of water per day to cool.
- \$20M/year to maintain
- Next datacenter in Ft. Meade, Md.

Facebook Data Center NC



facebook

Forest City, NC
April 2, 2012

FRC 1

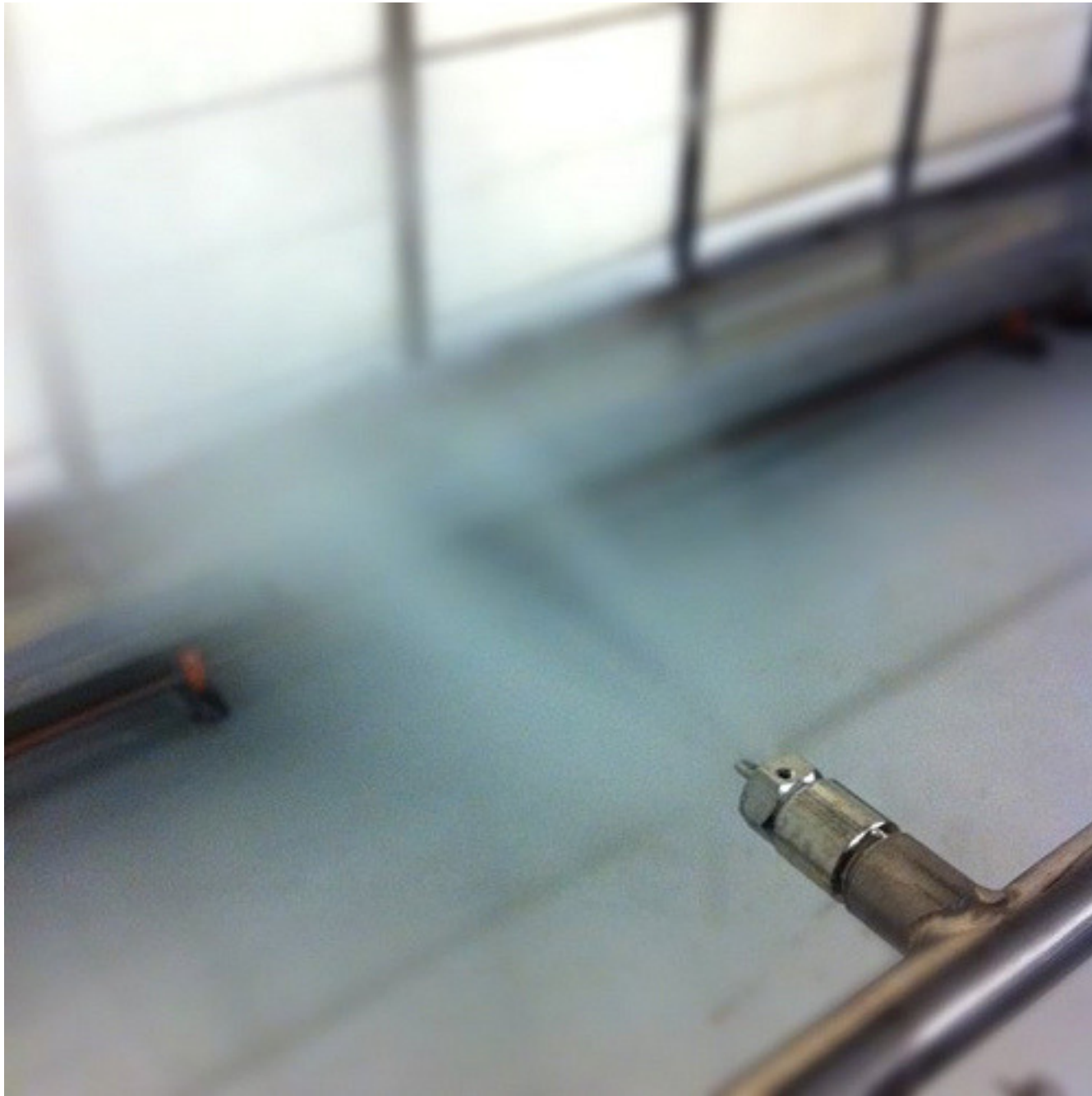
AerialPhotoPros.com

An aerial view of Facebook's new data center in Forest City, North Carolina. *Image: Courtesy Facebook*

Facebook Air Filter

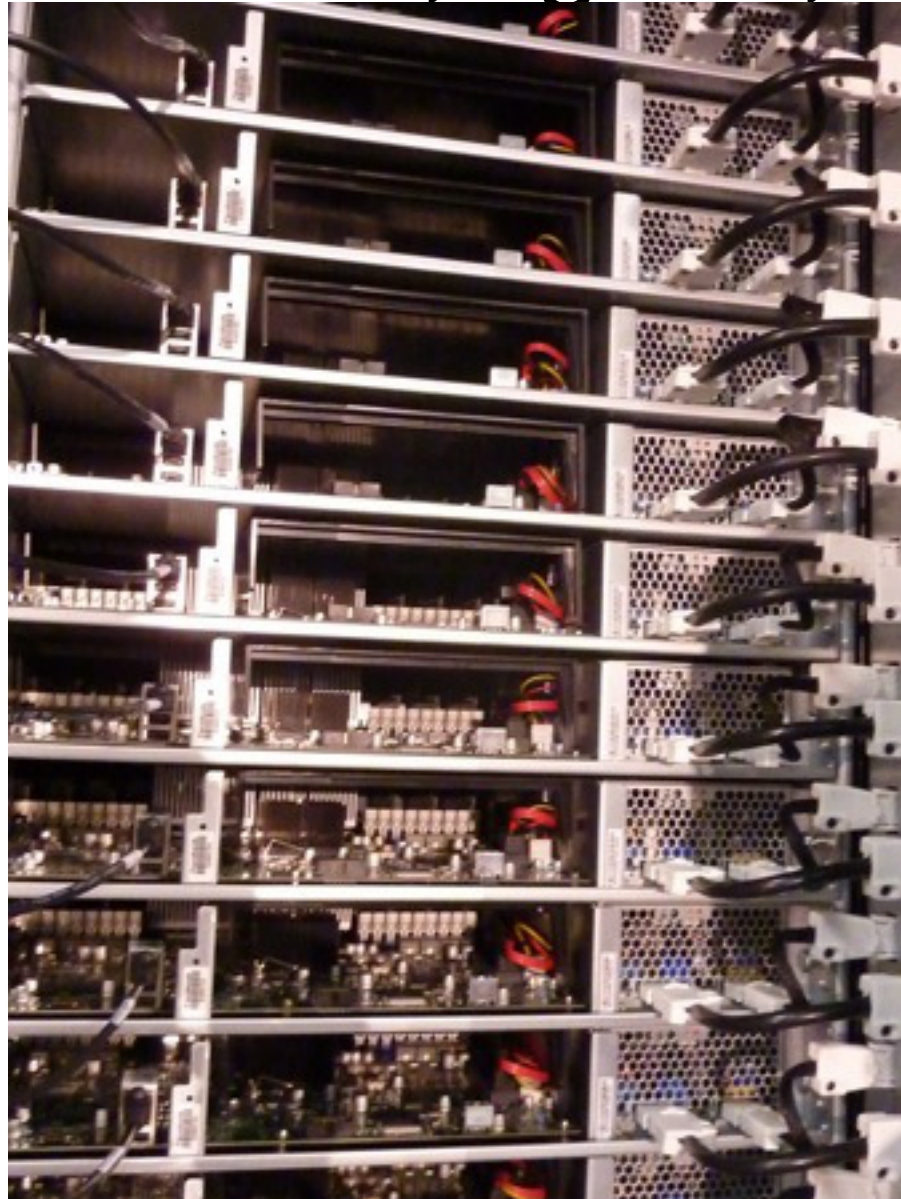


Facebook Just Add Water



Facebook Rack of Servers

1.5U, no big UPS, no AC, tall heat sinks, no tools, lighter, cheaper



OpenCompute.org

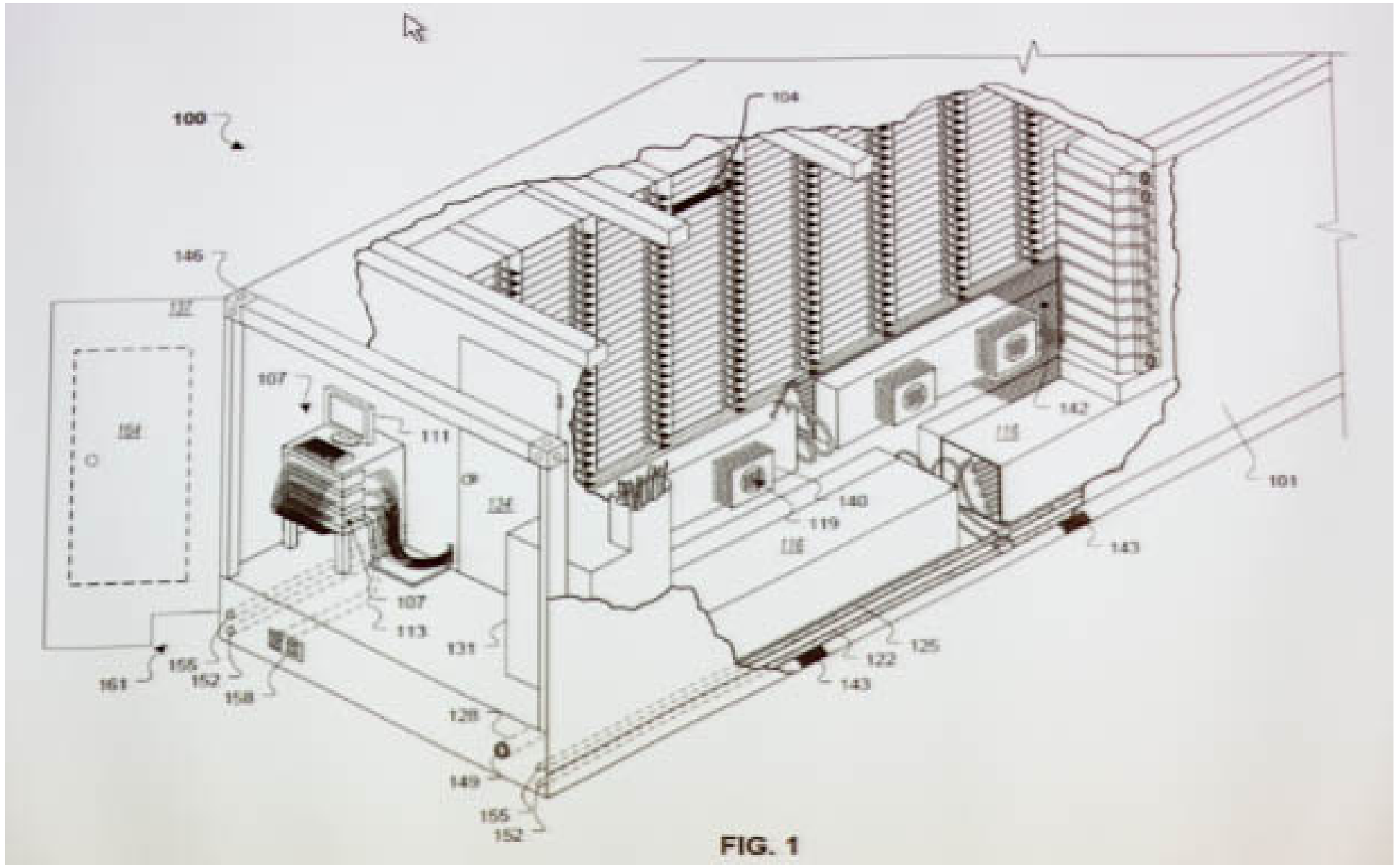
(From Facebook's Specifications)

- Claims/Goals for a datacenter
 - 38% more efficient
 - 34% less expensive
- Open hardware designs (images.google.com)
 - Rack (OU = 1.5U) 21"
 - Server
 - Block storage
 - Virtual IO
 - Hardware management
 - Certification

Tiered Storage

- Big Memory
- SSDs on PCIe bus (cache and perm)
- Local HDDs (mix PCIe, SAS, ...)
- Net attached SSDs, HDDs
- Local backup (no DR)
- Remote HDDs
- Remote tape

Google's 1AAA Container (1160 servers, many/datacenter)



Raised Floors in Google's Containers



eBay Data/Compute Servers

- 1U with 64 bit CentOS (use Enterprise Red Hat on “main” servers)
- 2 quad core machines
- 48 GB RAM
- 1 Gig Ethernet for nodes
- 12-24 TB storage
- A rack has 38-42 servers
- Rack switch has uplink of 40Gb/s to the core switches



**FRUSTRATION
AHEAD**



Google: Failures/Year/Cluster

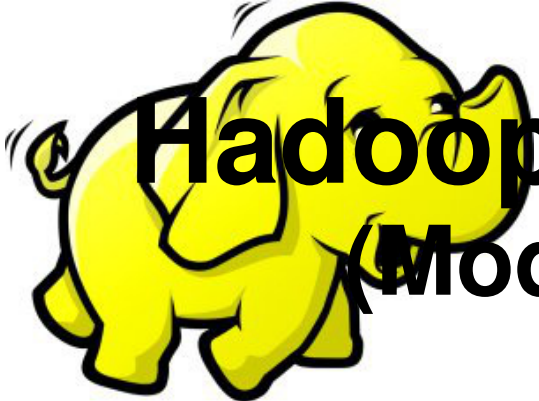
- ~0.5 heat problems (pwr down most eq < 5 min, 2 day MTTR)
- ~1 PDU failure (500-1000 machines gone, ~6 hrs MTTR)
- ~1 rack move (~500-1000 machines down, ~6 hrs MTTR)
- ~1 network rewiring (rolling ~5% down over 2 day span)
- ~20 rack failures (40-80 machines disappear, 1-6 hrs MTTR)
- ~6 racks see 50% packet loss (?? MTTR)
- ~8 net maint failures (30 min connectivity loss MTTR)
- ~12 router reloads (2-3 min MTTR, web and clusters)
- ~3 router failures (web access 1 hr MTTR)
- ~1000 machine failures; ~many thousands disk failures
- Many minor, hw and sw problems; many Internet problems

Don't Yell; Deal With It!



Technology to address Big Data

- Distributed File System for very large files, distributed redundantly over many machines
- MapReduce computing environment provides distribution and fault tolerance
- Integrate your RDBMS!!! The above technology will **not** solve traditional problems for which a RDBMS does well!!!
- Additional infrastructure (Lots new here!)



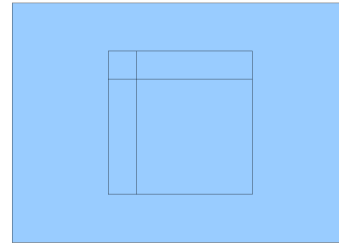
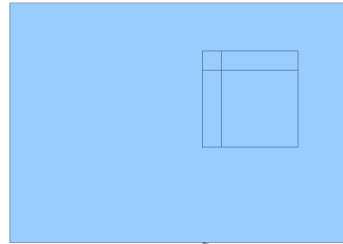
Hadoop Distributed File System (Modeled after Google's GFS)

- Divide (big) file into large “chunks” 64-128MB.
- Replicate each chunk at least 3 times, storing each chunk as a linuxfile on a data server. Usually put 2 in the same rack.
- Optimize for reads and for record appends.
- Before every read, checksum the data.
- Maintain a master name server to keep track of metadata, incl. chunk locations, replicas,...
- Clients cache metadata and read/write directly to the data servers.

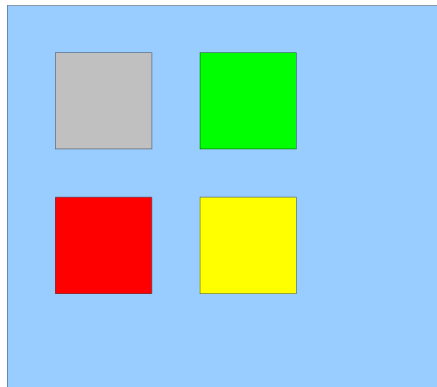
HDFS



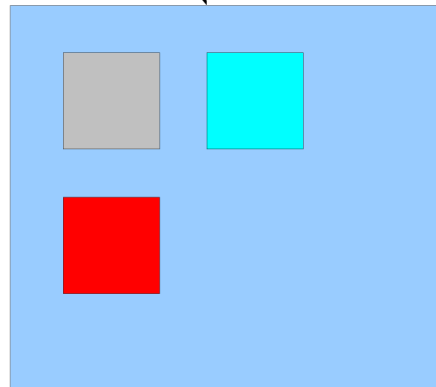
Client



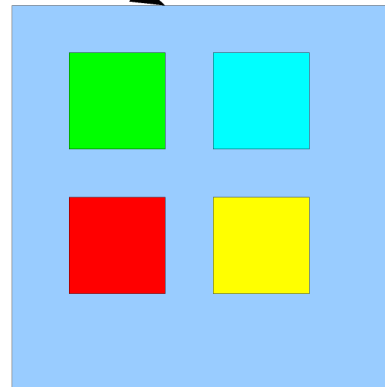
Name Master



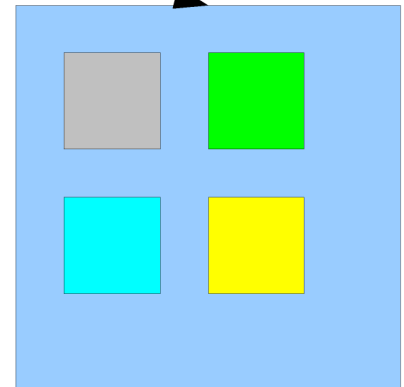
Chunk/Data
Server



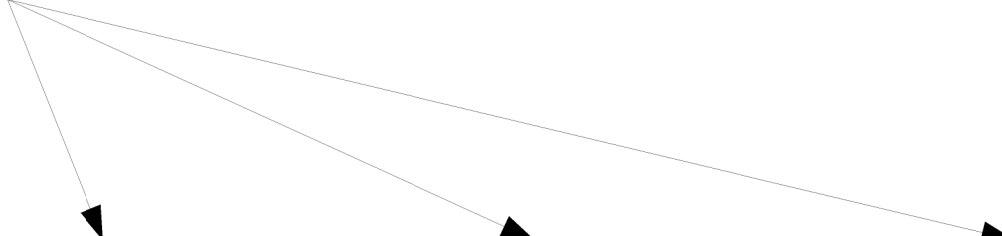
Chunk/Data
Server



Chunk/Data
Server



Chunk/Data
Server



HDFS/GFS Notes

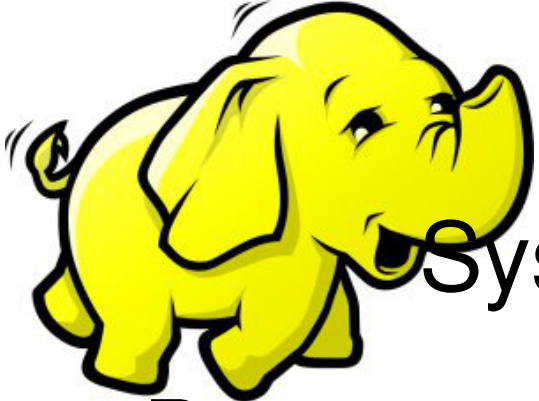
- A client interacts with master name server only to get file metadata, which it caches. Then it can interact directly with the data servers.
- Data servers update their metadata with the master every minute or so. (Heartbeat)
- Large chunk size reduces size of metadata, master-client interaction, and network overhead.
- Master name server is a single point of failure that is addressed with shadow servers and logs.
- Applications must optimize to the file system.

Critique of HDFS (or GFS)

- Single Name Node
 - Single point of failure (Cloudera and MapR are addressing this with failover strategies)
 - Namespace must fit in Name Node's memory: bounds the number of files (even if space exists on the data nodes)
- Replication strategy (3 copies) is expensive
 - Storage cost
 - Network bandwidth
- Inflexible striping strategy
- Heartbeat overhead

Goals to Distribute Applications

- Distribute the application across many machines
- Put the computations close to the data
- Near linear scaling n to N machines
- Handle failures (machine, storage, network)
- Monitor progress and deal with slow jobs

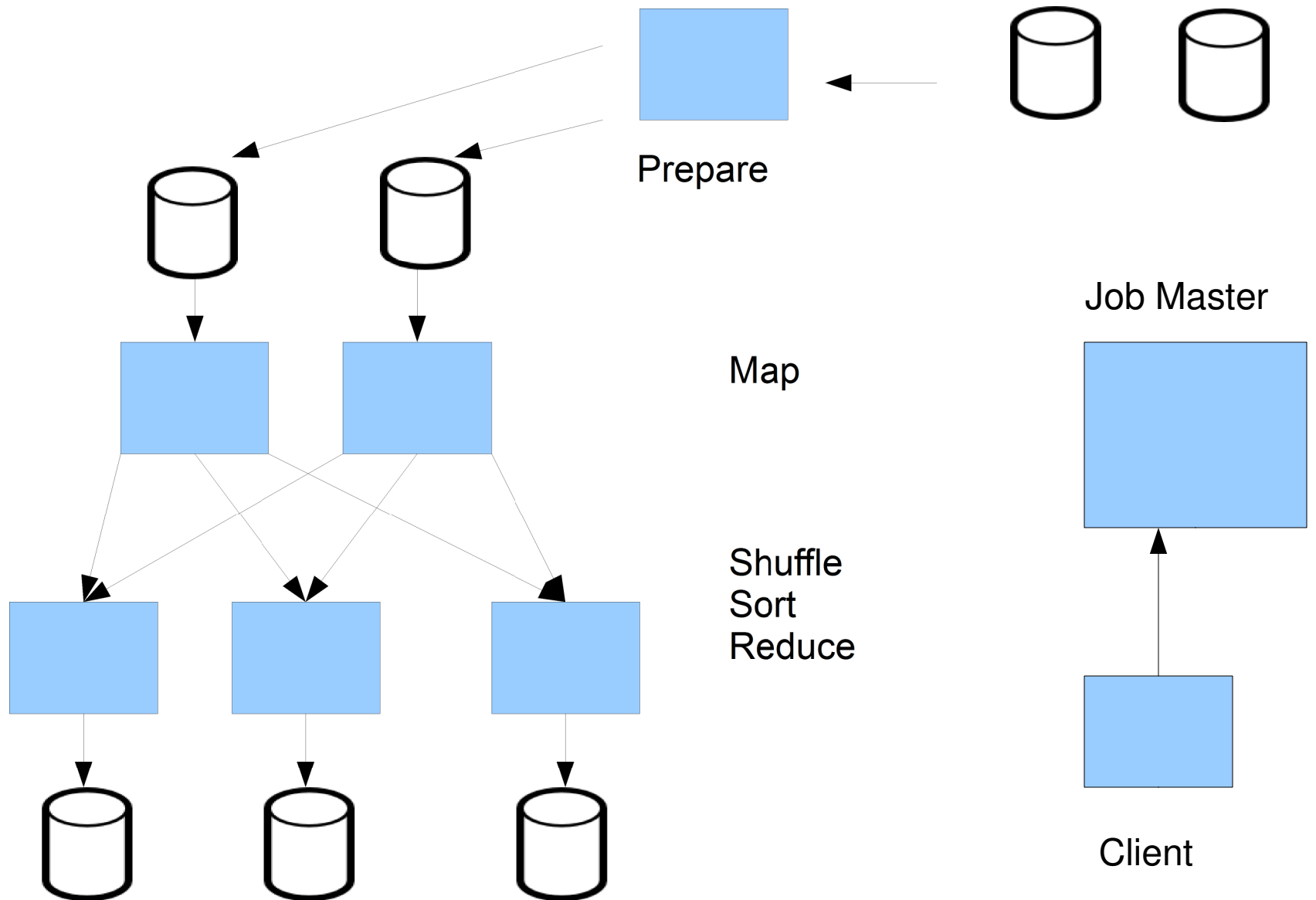


MapReduce

System to Process Big Files

- **Prepare** a Big File for input as (key1, value1) pairs into multiple Mapper jobs
- **Map**(key1, value1) generates intermediate (key2, value2) pairs and separates this output into R files for input into R parallel Reducers.
- When all mappers done, the R reducers read and **Sort** their input files.
- **Reduce**(key2, value2) does a data reduction processing stage. Output = R Big Files of (key3, value3) pairs (not sorted nor combined.)

MapReduce System



Immediate MR Benefits

- Automatic parallelization: No code changes to go from 10 to 1000 machines.
- Automatic load balancing
- Network and disk transfer optimizations
- Robust: Machine and disk failover

Job/Task/Attempt Structure

- Job
 - Multiple tasks: mapper tasks, reducer tasks, .
 - Attempt: a restart of a task
- Word Count across 20,000 files is one job
- Quality Factor: $m\%$ of data are processed.
- Speculative execution yields multiple attempts even if no failures. Job tracker will kill those attempts that haven't finished.
- Task Tracker runs on slave nodes

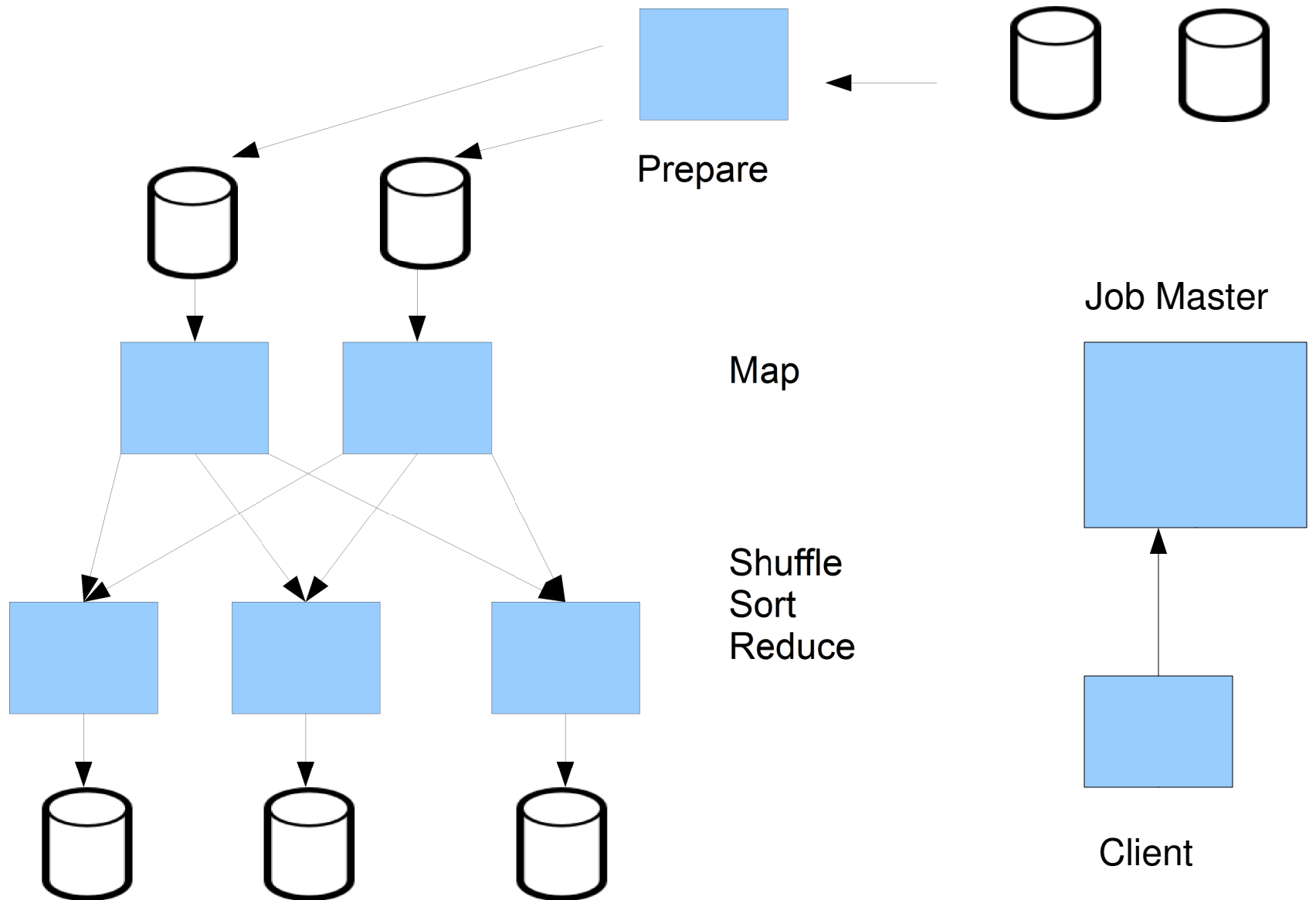
Properties of MapReduce

- Can read from multiple file systems
- Can use indices
- Often MR is faster than even loading the data into a DBMS
- Often algorithms are difficult or impossible in SQL
- Address startup overhead by keeping worker processes alive waiting for next MR run.

Interesting Google Applications of MR

- Inverting {outgoing links} to {incoming links}
- Efficient support of Google Queries
- Stitching together overlapping satellite images for Google Earth
- Rendering map-tile images of road segments for Google Maps
- Over 15,000+ MR applications; 100,000+ MR jobs/day

MapReduce System



Sort on Key

- Map(key, value) = (key, value) identity fcn
- Reduce (key, value) = (key,value) identity fcn
- With multiple reducers, Map needs to have a partitioner class or a hash function so that $k1 < k2$ implies $\text{hash}(k1) \leq \text{hash}(k2)$, an “ordered hash function”

String Search

- Map (docName, contents) \rightarrow (docName, Contents)
 - If string In Contents Then emit(docName, Contents)
- Reduce(docName, Contents) = identity
- Note Google(string) does a little more, it emits the line or two that contains string, and it orders them by “goodness”

Sorted String Search

- Suppose you want to search for a string in a list of documents, but you want the output sorted.
- Map(docName, Contents)
 - Is the identity, except it hashes the input into R output files with $d < d'$ implies $\text{hash}(d) \leq \text{hash}(d')$
 - emit(docName, Contents) to file(hash(docName)/R)
- Reduce(docName, Contents)
 - If string In Contents Then emit(docName, Contents)
- Exercise: Make more efficient

Time Series

- A *time series* is a sequence $(t_1, v_1), (t_2, v_2), \dots$ where $t_1 \leq t_2 \leq \dots$ are points in time.
- Medical, financial, image. Smart Grid data, ...
- Surprise: genome data (text representation.)
- TS programming languages (EPS, APL, R, ...)
- RDMS' put time series in a two column table; at best ok for small ts, but not for Big Data. Also SQL not so hot for ts. Ditto spreadsheets.

Moving Averages – Smoothing Data

- A *TS moving average* over time T at time t is the average of the v_i for t_i in $(t-T, t]$. Typically one takes t to be one of the t_i to guarantee at least one value to be averaged. Move t to repeat..
- Three day's average stock price for IBM (typically thousands of trades with actual quantity varying daily). Usually computed daily.

Example: Daily 3-day Average Stock Market Trades

- Multiple trade streams funneled to the mappers. Map=Identity hashes symbol into R buckets; sort (time, symbol) in R reducers:
- Reduce(time, symbol, quantity, price)
 - $\text{count3[*]} := \text{count2[*]} := \text{count1[*]} := \text{sum3[*]} := \text{sum2[*]} := \text{sum1[*]} := \text{day0} := 0$
 - $\text{Symbols3} := \text{Symbols2} := \text{Symbols1} := \{ \}$
 - Loop
 - Input(time, symbol, quantity, price)

Example 3day moving average continued

- If EOF Or day(time) > day0 Then - - flush what we've got
 - Begin For s In Symbols3 + Symbols2 + Symbols1 Do
 - Output (s, day(time), (sum1[s]+sum2[s]+sum3[s])/(count1[s]+count2[s]+count3[s]+1))
 - If EOF Then Return
 - day0 := day(time)
 - count3[*] := count2[*]; count2[*] := count1[*]
 - sum3[*] := sum2[*]; sum2[*] := sum1[*]
 - sum1[*] := count1[*] := 0
 - Symbols3 := Symbols2; Symbols2 := Symbols1
 - Symbols1 := { } End
- **Symbols1 += symbol - - set addition**
- **sum1[symbol] += quantity*price**
- **count1[symbol] += quantity**
- End Loop

Time Series

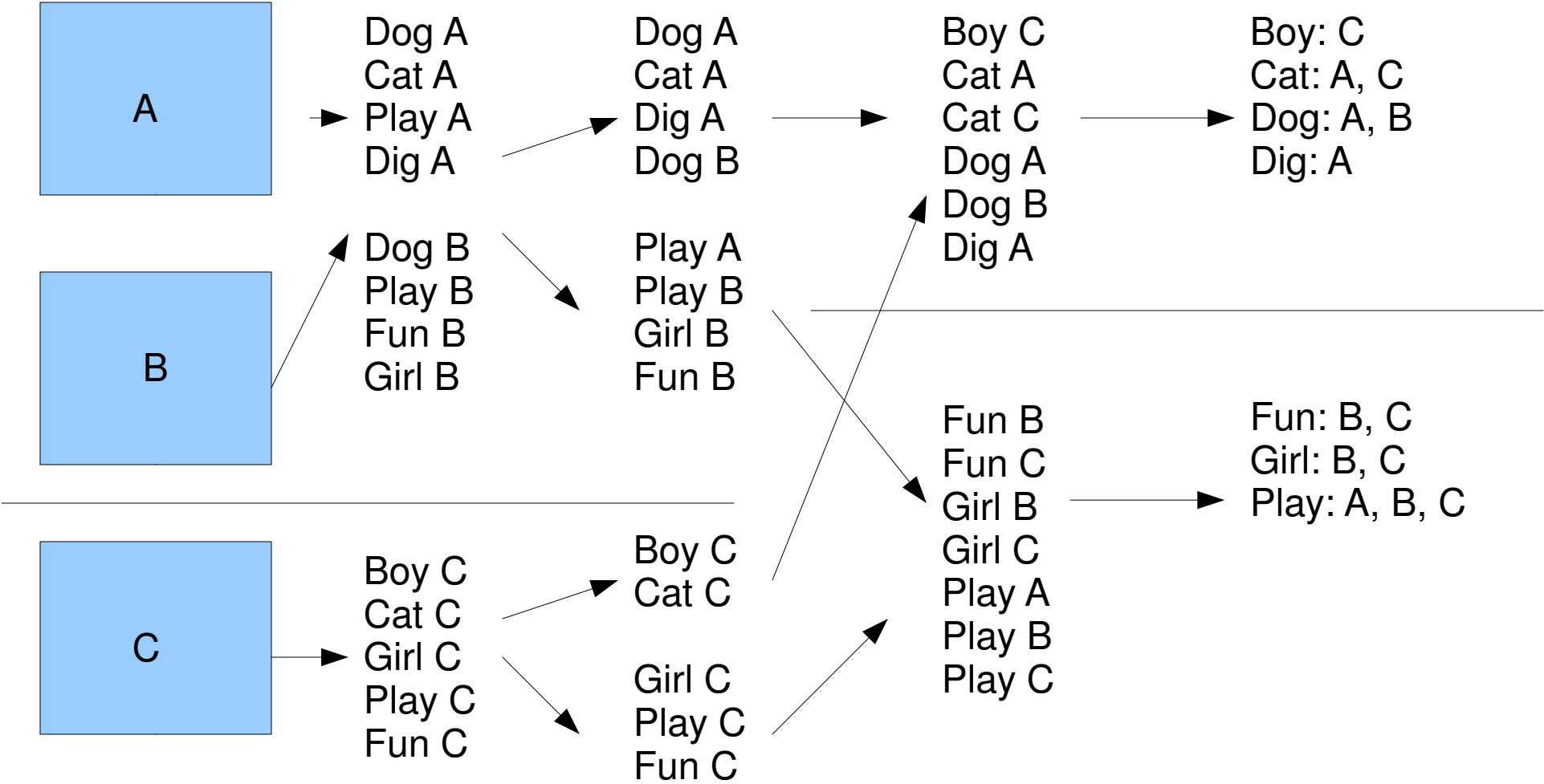
Lessons Learned

- Often the Mapper only hashes so that each reducer gets selected data.
- The sort is essential to the Reducer's algorithm.
- Standard programming idiom: Watch for EOF or change in a parameter. If yes, output accumulated results and reset.
- OK to store locally modest amount of data.
 - Watch for data scaling problems
 - Here, a larger $R = \# \text{Reducers}$ helps
- Hadoop will schedule if $R > \# \text{Machines}$ avail.

Index a Document

- `map(pageName, pageText)`
 - For word `w` In `pageText`
 - `Emit(w, pageName)` to `file(hash(w))`
- Use ordered hash to output to reducers
- `reduce(word, values)`
 - For each `pageName` in `values` Until word changes
Do `AddToOutputList(pageName)`
 - `Emit(word, “: ”, pageNameList)`

Example Word Index



More Lessons Learned

- Sometimes significant work can be divided between Map and Reduce (Word Count and Index parse for words; Select and Search evaluate the predicate).
- Sometimes local storage is needed (Index) be careful that this scales!
- Often self-identifying records are needed which need to be constructed early (Prepare).
- Next we'll see an example where multiple MR jobs are needed.

Multistage Pipeline

Term Frequency & Inverse Document Frequency

- t denotes term=word; d names a document in D
- $tf(t,d)$ = number times t occurs in d
- $wc(d)$ = number distinct terms in d
- $ndocs(t,D)$ = number of docs in D containing t
- $totdocs(D)$ = number of docs in D
- $idf(t,D) = \log(totdocs(D)/ ndocs(t,D))$
- $tfidf(t,d,D) = tf(t,d)*idf(t,D)$
- ASSUME t is in some d In D

tf*idf Job Pipeline

- Job 0 - compute $N = \text{totdocs}(D)$
- Job 1 - compute $\text{tf}(t,d)$, pass N as a parameter
- Job 2 - compute $\text{wc}(d)$ adding the $\text{tf}(t,d)$ over t . Pass N , output all $\text{tf}(t,d)$ under key “k tf ”, and output all $\text{wc}(d)$ under key “k wc ”.
- Job 3 - compute $\text{ndocs}(t,D)$ adding for each d , if $\text{tf}(t,d) > 0$ then 1 else 0. Pass N , output all $\text{tf}(t,d)$ and $\text{wc}(d)$ under key “k tf ”, and output $\text{ndocs}(t,D)$ under key “k ndocs ”.
- Job 4 - calculate, for each (t,d) both $\text{idf}(t,D) = \log(\text{totdocs}(D)/\text{ndocs}(t,D))$ and $\text{tfidf}(t,d,D)$ then $\text{output}((t,d),(\text{tf}(t,d),\text{wc}(d),\text{ndocs}(t,D),\text{idf}(t,d), \text{tfidf}(t,d,D)))$

Job 0 - totdocs(D)

- Prepare sets up many mappers and one reducer
- $\text{map}(d, 1) = (D, N)$
 - $v := 0$
 - Loop $\text{read}(d, n); v += 1$; End Loop
 - If EOF Then $\text{emit}(D, v)$
- $\text{reduce}(X, v) = N$ - - =total docs in D = $\text{totdocs}(D)$
 - $N := 0$
 - Loop $\text{read}(X, v); N += v$; End Loop
 - If EOF Then $\text{emit}(D, N)$ - - single element output

Job 1 – Term Frequency in Doc

- Job 1: Term/Word frequency and count
 - $\text{map}((d, \text{contents}, N) = ((t, d), 1, N)$
 - - - parse for t implicitly; $N = \text{totdocs}(D)$
 - - - hash so that each reducer gets all (t,d) 's together with no (t,d) splits across reducers
 - $\text{reduce}((t, d), v, N) = ((t, d), n, N)$ - - $n = \text{tf}(t,d)$
 - For each (t,d) add up all the $v=1$'s to get n
 - $\text{Emit}((t,d), n, N)$
 - Note the output (t,d) 's are unique and inclusive

Job 2: Word Counts for d in D

- $\text{Map}((t,d), n, N) = (d,(t,n,N))$ -- $n = \text{tf}(t,d)$, $N = |D|$
 - Hash the d's so that each reducer gets all the d values for the (t,d)
- $\text{Reduce}(d,(t, n, N)) = (k, (t, d, \text{tf}(t,d), \text{wc}(d), m, N))$
 - For each d, $v := 0$; Loop over t,
 - $k := \text{"ktf"}$; $m := \text{If } n > 0 \text{ Then } 1 \text{ Else } 0$
 - $\text{emit}(k, (t, d, n, 0, m, N))$ - - pass on the $n = \text{tf}(t,d)$ info & m
 - $v += n$ - - $n = \text{tf}(t,d)$ - - Note, no need to parse d again!
 - End Loop over t - - now $v = \text{wc}(d)$
 - $k := \text{"kwc"}$
 - $\text{emit}(k, (\text{""}, d, 0, v, 0, N))$
 - End Loop on d

Job 3 – $\text{ndocs}(t, D)$

- Pass each reducer output file back to a job3 mapper
- $\text{map}(k, (t, d, n, w, m, N)) = \text{identity}$
 - -if $k = \text{"ktf"}$, $n = \text{tf}(t, d)$ and $m = 1$ if $\text{tf}(t, d) > 0$ else 0 & $w = \text{wc}(d)$ if $k = \text{"kwc"}$ & $N = \text{totdocs}(D)$
- Again hash d so that no d 's are split across reducers, and sort so that "kwc" is before "ktf" and all d 's are together.
- $\text{reduce}(k, (t, d, n, w, m, N)) = (k, (t, d, n, w, v, N))$
 - If $k = \text{"kwc"}$ Then
 - $\text{WC} := w$
 - Skip to next record - - $\text{wc} = \text{wc}(d)$

Job 3 continued = ndocs(t,D)

- If k = “ktf” Then For each t, v:=0; Loop over d
 - emit(k, (t,d,n,wc,0,N)) - - Note wc is folded in!
 - v += m - - add all the m's to get ndocs(t,D)
 - End Loop over d
- k := “kndocs”
- emit(k, (t, “”, 0, 0, v, N)) - - v = ndocs(t,D)
- End For each t

Job 4 – Calculate $\text{idf}(t,D)$ and $\text{tf}*\text{idf}(t,d,D)$

- $\text{Map}(k,(t,d,n,w,v,N)) = \text{identity}$
 - Hash d's so the (t,d)'s aren't split across reducers; sort so that “kndocs” is first and all d's are together.
- $\text{Reduce}(k,(t,d,n,w,v,N))$
 - Loop
 - $\text{Read}(k, (t,d,n,w,v,N))$
 - If $k = \text{“kndocs”}$ Then $\text{ndocs} := v$; Skip - - $\text{ndocs}(t,D)$
 - If $k = \text{“ktf”}$ Then $\text{tf} := n$; $\text{idf} := \log(N/\text{ndocs})$; $\text{tfidf} := \text{tf}*\text{idf}$ - - $\text{tf}(t,d), \text{idf}(t,D), \text{ndocs}(t,D), \text{tfidf}(t,d,D)$
 - Output $((t, d), (\text{tf},\text{idf},\text{wc},\text{ndocs},N,\text{tfidf}))$
 - End Loop

Summary TF-IDF

- Several small jobs add up to a full algorithm
- Lots of code reuse is possible – stock classes exist for aggregation, identity, etc. Standard idioms for code.
- Job 0 is not necessary if $\text{card}(D)$ is known
- Needed self-identifying records
- There are other ways to do $\text{tf} \cdot \text{idf}$, e.g. using external files.
- Same techniques to iterate and converge.

More Topics

- The impact of SSDs
- The death of RAID?
 - Small v. disks
 - Erasure codes
- Backup
 - Compression, deduplications
 - Tape?
- Virtual Networks
- Virtual Datacenters

Thank You!

- Questions?
-
-
- Gayn Winters, Ph.D.
- gaynwinters@ieee.org
- 714-366-4296