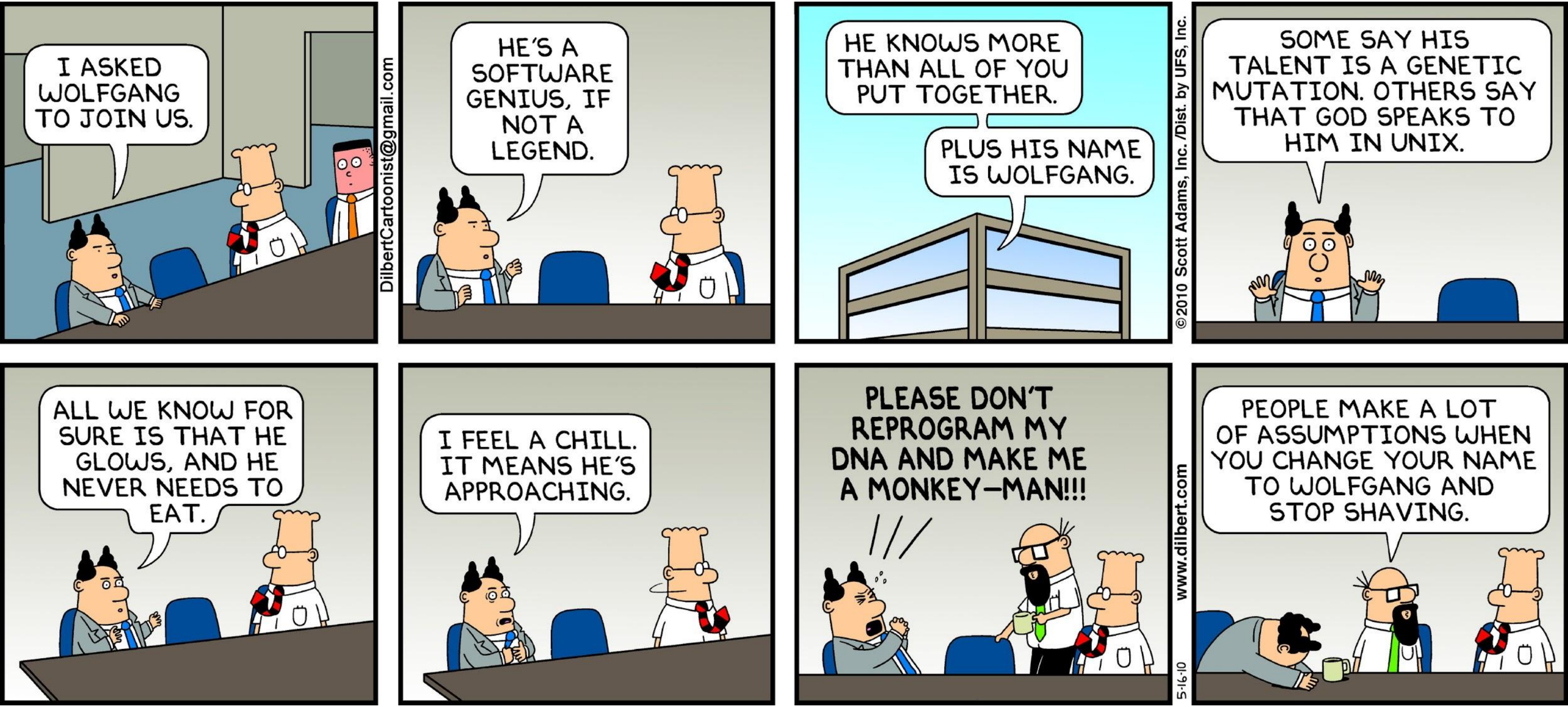
MapReduce

Wolfgang Richter
wolf@cs.cmu.edu

whoami



I ASKED WOLFGANG TO JOIN US.
DilbertCartoonist@gmail.com
HE'S A SOFTWARE GENIUS, IF NOT A LEGEND.
HE KNOWS MORE THAN ALL OF YOU PUT TOGETHER.
PLUS HIS NAME IS WOLFGANG.
©2010 Scott Adams, Inc. /Dist. by UFS, Inc.
SOME SAY HIS TALENT IS A GENETIC MUTATION. OTHERS SAY THAT GOD SPEAKS TO HIM IN UNIX.
ALL WE KNOW FOR SURE IS THAT HE GLOWS, AND HE NEVER NEEDS TO EAT.
I FEEL A CHILL. IT MEANS HE'S APPROACHING.
PLEASE DON'T REPROGRAM MY DNA AND MAKE ME A MONKEY-MAN!!!
www.dilbert.com
5-16-10
PEOPLE MAKE A LOT OF ASSUMPTIONS WHEN YOU CHANGE YOUR NAME TO WOLFGANG AND STOP SHAVING.

MapReduce for Big Data Processing

- September 2007, Google
  - 2.2+ million MR jobs
  - 403 TiB input data (almost 0.5 PiB)
  - 34 TiB intermediate data
  - 14 TiB output data
  - Averaged 394 machines assigned to each job

  - 3,800 LoC C++ → 700 LoC C++

MapReduce Wins Terasort

- 2008, won TeraSort Benchmark
  - 910 nodes
  - Sorted 10 billion records, 1 terabyte of data
  - 209 seconds (3.48 minutes)
- 2013, 100 terabytes, 72 minutes
- MapReduce is important across the industry

https://hadoop.apache.org/docs/current/api/org/apache/hadoop/examples/terasort/package-summary.html

Outline

1. Building Distributed Systems
2. Let’s Simplify: map and reduce
    a. My Sister’s Engagement
3. MapReduce: Optimizations
4. MapReduce: Not for Everyone

Building Distributed Systems

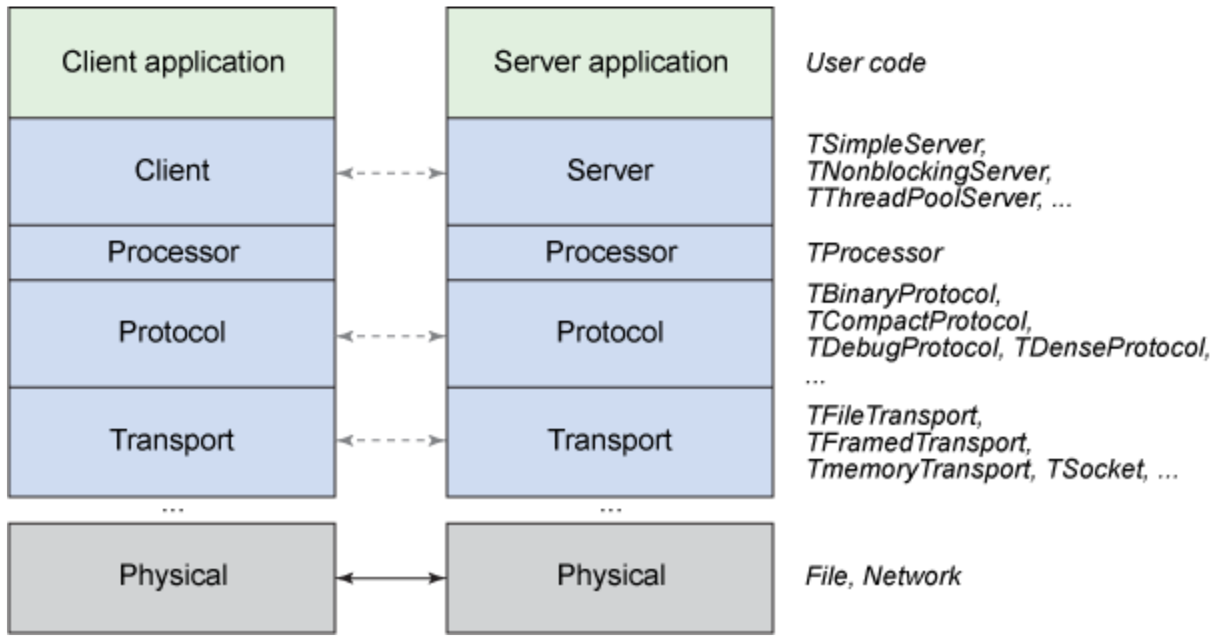
- Is REALLY hard

- But, many workloads share common patterns

Why is it hard?

Why are they so hard?

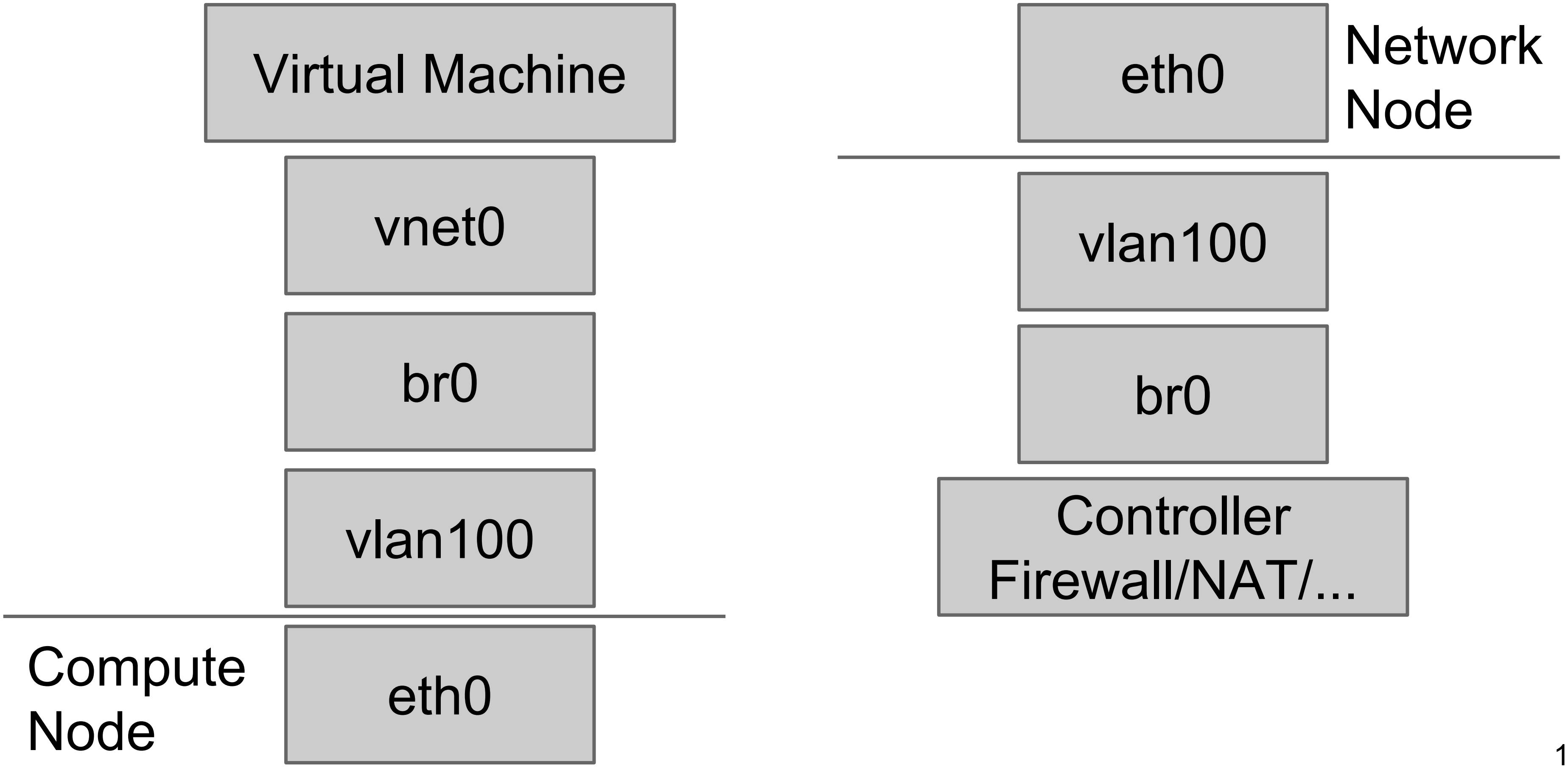
- Network Communication
  - Serialization and marshaling of data
  - RPC mechanisms
  - Fault-tolerance
  - Bandwidth limitations
  - Latency limitations


Client application
Server application
User code
Client
Server
TSimpleServer,
TNonblockingServer,
TThreadPoolServer, ...
Processor
Processor
TProcessor
Protocol
Protocol
TBinaryProtocol,
TCompactProtocol,
TDebugProtocol, TDenseProtocol,
...
Transport
Transport
TFileTransport,
TFramedTransport,
TmemoryTransport, TSocket, ...
...
...
Physical
Physical
File, Network

Why are they so hard?

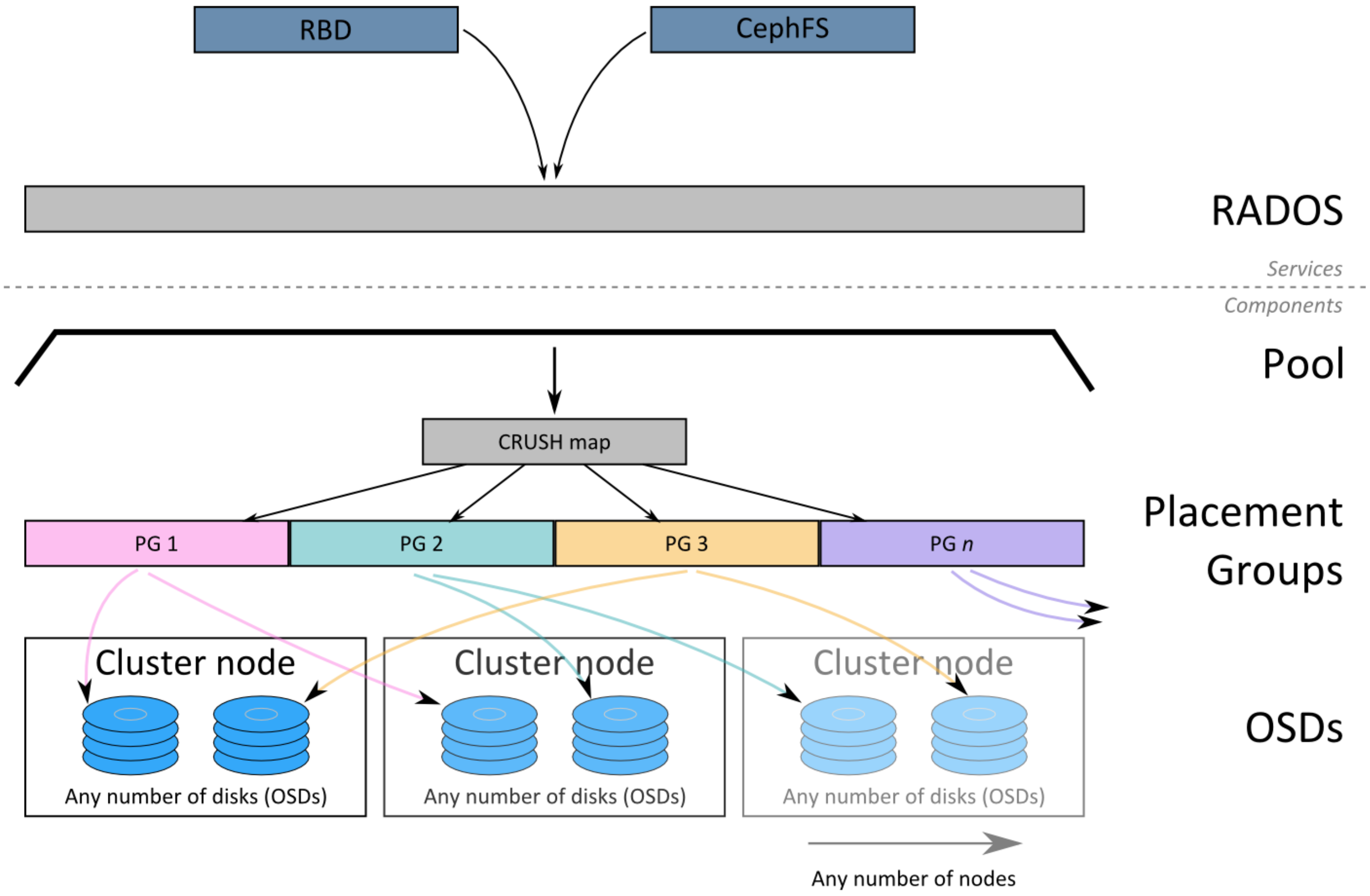
- Problem Diagnosis
  - Log collection via network
  - Streaming log analytics
- Security
  - Audits
  - Updates
  - Intrusion detection
  - Virus scanning
  - Firewalls

“Simple” Cloud Example



Why are they so hard?

- Storage Scalability
  - Cross-node data sharing
  - Storage fault tolerance (RAID, etc.)
  - Unifying file namespace across many file systems
  - Extreme high read/write bandwidth requirements
  - Backup solutions and replication


RBD
CephFS
RADOS
Services
Components
Pool
CRUSH map
PG 1
PG 2
PG 3
PG n
Placement Groups
Cluster node
Any number of disks (OSDs)
Cluster node
Any number of disks (OSDs)
Cluster node
Any number of disks (OSDs)
OSDs
Any number of nodes

Why are they so hard?

- Environment Issues
  - Cross-OS cooperation (at least kernel versions)
  - Heterogeneous hardware
  - Library versions
  - Bugs in various parts of the stack (firmware, OS, …)

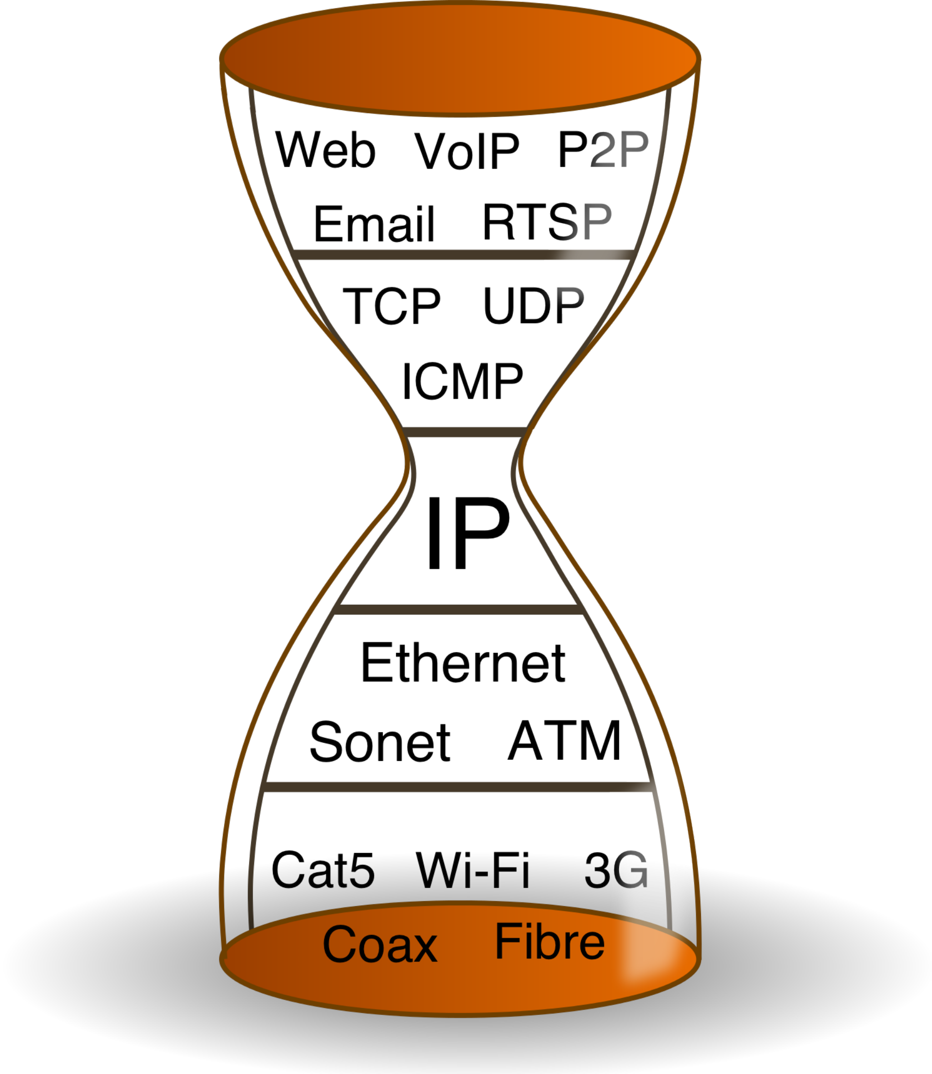
Number of machines	Platform	CPUs	Memory
6732	B	0.50	0.50
3863	B	0.50	0.25
1001	B	0.50	0.75
795	C	1.00	1.00
126	A	0.25	0.25
52	B	0.50	0.12
5	B	0.50	0.03
5	B	0.50	0.97
3	C	1.00	0.50
1	B	0.50	0.06

Source:
Google Cluster Traces
http://www.pdl.cmu.edu/PDL-FTP/CloudComputing/ISTC-CC-TR-12-101.pdf

How did we handle the Internet?

- Size: 4.47 billion indexed web pages
  - http://www.worldwidewebsize.com/
- 15 billion devices by 2015
  - http://www.forbes.com/sites/quora/2013/01/07/how-many-things-are-currently-connected-to-the-internet-of-things-iot/
- North Korea: 8 hosts (2012)
  - Still, a distributed system (source: CIA)

The Narrow Waist



MapReduce: ~~The~~ a DS Narrow Waist

- Many big data workloads have similarity

- Pattern
    - Extract/transform input data (10s TiB, PiB, EiB scale)
    - Sort/stage intermediate data (same scale)
    - Last computation, output final result

- Extract, Transform, Load (DB community)

Where Did MapReduce Come From?

- Google, OSDI 2004
  - Cited by 12,950
  - Even my future cousin-in-law (heart surgeon, UVa)

- As of September 2007:
  - 4,000+ node clusters
  - 2.2 million MR jobs, 73,000 per day
  - 403 TiB of data

Goals of MapReduce

- Minimize network traffic
- Load balance
- Easily parallelize/schedule tasks
- Fault-tolerant

What are map and reduce?

Let's simplify... with a `map` and a `reduce`

```
>>> map(lambda x: x, [1,2,3,4,5])
[1, 2, 3, 4, 5]


>>> reduce(lambda a,b: a + b, [1,2,3,4,5], 0)
15
```

Google's Version

map(k1, v1) → <k2, v2>

reduce(k2, [v2...]) → <k3, v3>

- MapReduce is a distributed run-time
  - Takes care of every distributed headache
  - Must fit problem to map + reduce paradigm

Why are map and reduce good abstractions for a DS?

Embarrassingly Parallel Applications

- A problem in which there is
  - little, to no effort to separate into many parallel tasks

- Many languages implement parallel versions
  - Python (multiprocessing module)
  - Haskell (parallel package)
  - Julia (pmap built-in)

Google’s Word Count: mapper

```
map(String key, String value):
  // key: document name
  // value: document contents
  for each word w in value:
    EmitIntermediate(w, “1”);
```

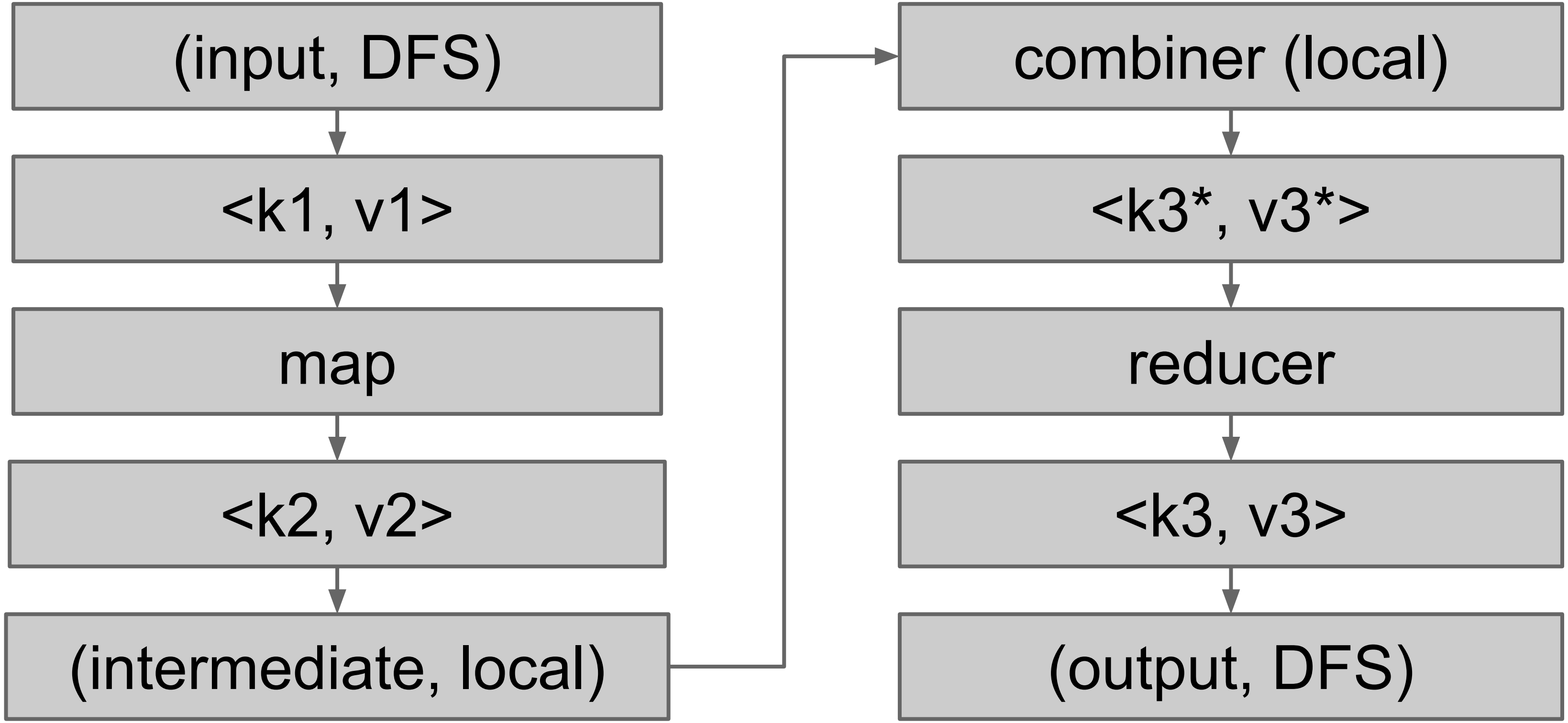
Google's Word Count: reducer

```
reduce(String key, Iterator values):
  // key: a word
  // values: a list of counts
  int result = 0;
  for each v in values:
    result += ParseInt(v);
  Emit(AsString(result));
```

MapReduce Assumptions

- Data in some sort of <k,v> record format
  - Or is easily transformed into such a format
- Data comes from a DFS
- Intermediate data stored locally
- Final data stored back in the DFS
- Jobs managed by a centralized scheduler

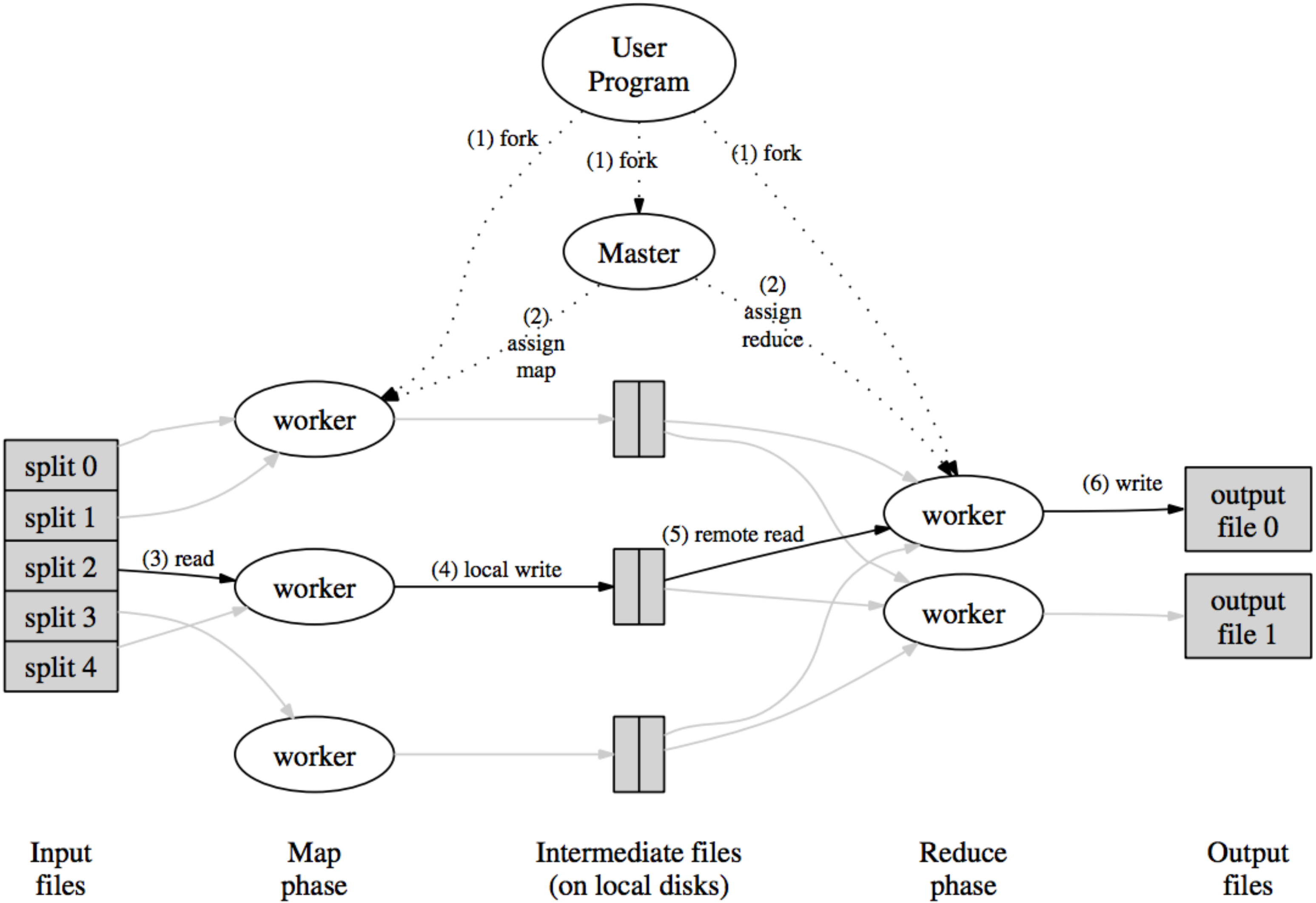
MapReduce Data Flow



How is data assigned to reducers?

- User-defined partition+sort function, or…
  - hash(k2) mod R
  - R → number of reducers
  - Increasing key order presented to reducer

- Tough to fix data skew problem
  - Zipfian distributed data set
  - “Lady Gaga” problem


User Program
(1) fork
(1) fork
(1) fork
Master
(2) assign map
(2) assign reduce
worker
worker
worker
worker
worker
split 0
split 1
split 2
split 3
split 4
(3) read
(4) local write
(5) remote read
(6) write
output file 0
output file 1
Input files
Map phase
Intermediate files (on local disks)
Reduce phase
Output files

What kind of API do we need from the DFS?

Any special assumptions?

Problems?

1. Movement of data to reducers
2. Crashing master or worker nodes
3. Stragglers: slow mappers or reducers
4. Bad records causing map/reduce failure
    a. Everyone has bugs right?

Pre-Reduce with Combiners

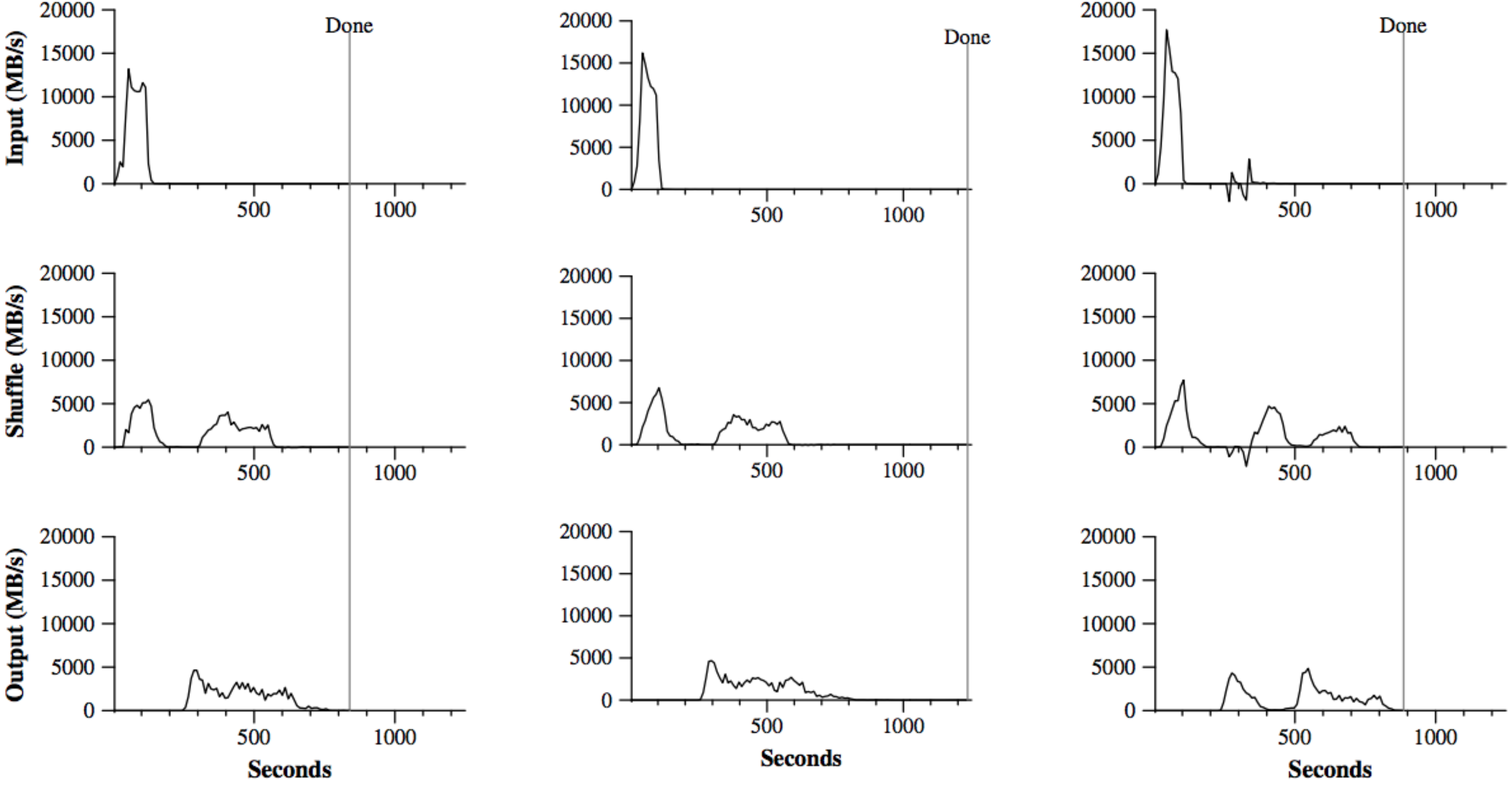
- Minimizes data movement
  - By an immediate “reduce” on intermediate data
  - Schedules compute at the data
- Requires reduce functions which
  - Are commutative and associative
  - Are deterministic

Crashes

- Master node
  - Restart MapReduce job
- Workers
  - Re-schedule unfinished maps or reduces
  - Re-schedule finished maps (local storage only)
  - Do nothing for finished reduces (DFS already)

Stragglers

- Arise from resource contention, failing HW…
- Solved with “backup jobs”
  - When close to completion, duplicate remaining tasks
  - Saves up to 44% wall time on certain jobs



(a) Normal execution

(b) No backup tasks

(c) 200 tasks killed

Skipping Bad Records

- MapReduce runtime installs signal handlers
  - When map/reduce crashes, send report to Master

- Given enough reports, the Master skips
  - Records never presented to map or reduce function

What is MapReduce bad at?

What is MapReduce Bad at?

- Graph problems + Some Machine Learning
  - GraphLab (CMU now commercial product)

- Iterative problems (more than 5-10 MR jobs)
  - Apache Spark (thanks UC Berkeley)

- Streaming problems
  - Apache Storm (thanks Twitter)

What is MapReduce Bad at?

- Exploratory Computation
  - Early termination of job
  - Examination of intermediate data/partial results
  - Low-latency computation
  - Arbitrary search primitives (more than map/reduce)

http://diamond.cs.cmu.edu/

Play around with MapReduce at home:

http://hadoop.apache.org/

or

http://aws.amazon.com/elasticmapreduce/

"...millions of EMR clusters per year"

wolf@cs.cmu.edu