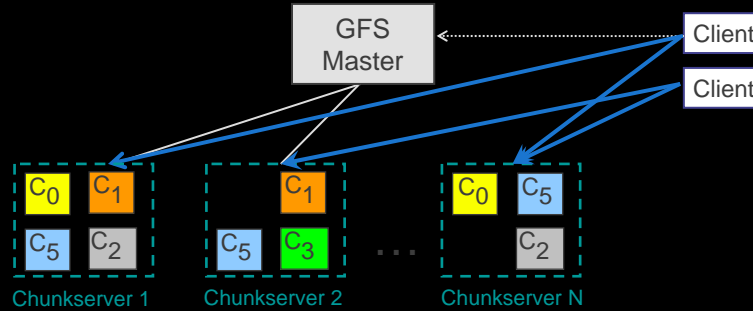




## GFS: Google File System

- Data replicated 3 times. Upon failure, software re-replicates.
- Master: Manages file metadata. Chunk size 64 MB.



<http://research.google.com/archive/gfs-sosp2003.pdf>

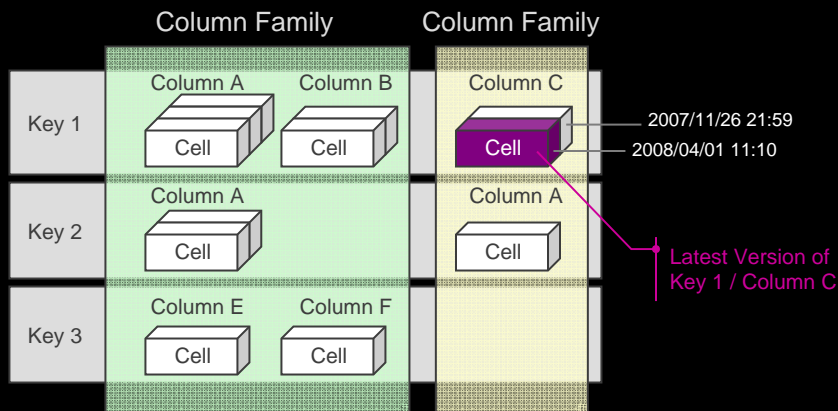


© 2009 Google, Inc. All rights reserved.

3

## Bigtable: A sparse, distributed, persistent, multidimensional, sorted Map

(RowKey, ColumnFamily:Column, Timestamp) → Value



<http://research.google.com/archive/bigtable-osdi06.pdf>



© 2009 Google, Inc. All rights reserved.

4

## Map-Reduce

- Computation expressed as Map / Group / Reduce

```
map(in_key, data) → list(key, value)
  (group output by key)
reduce(key, list(values)) → list(out_data)
```

- Well suited for "embarrassingly parallel" problems
- Open source implementation: Hadoop

<http://research.google.com/archive/mapreduce.html>



© 2009 Google, Inc. All rights reserved.

5

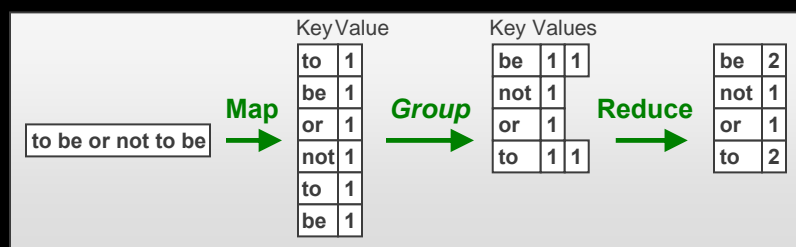
## Simple Example: Word Count

- Chose the key to get the most out of the framework
- Word count: word as the key

```
map(in_key, data):
  for word in data.split():
    output(word, 1)
reduce(key, list(values)):
  print key, len(values)
```

Value (not used here)

Key



© 2009 Google, Inc. All rights reserved.

6

## Log Processing: Sawzall

- Domain-specific language
- Describes processing of a single record
- Aggregation externalized into "tables"

```
count: table sum of int;
total: table sum of float;

number: float = string(input);

emit count <- 1;
emit total <- number;
```

```
3.5
2
1.25
-----
count[] = 3
total[] = 6.75
```

- "Programming in the first derivative"

<http://labs.google.com/papers/sawzall.html>

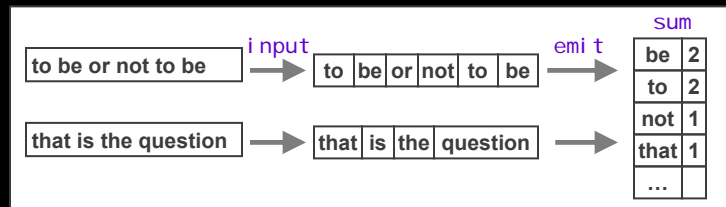


© 2009 Google, Inc. All rights reserved.

7

## Simple Example: Word Count

```
word_count: table sum[string] of int;
line = input;
words = split(line);
for each word in words:
  emit word_count[word] <- 1;
```



© 2009 Google, Inc. All rights reserved.

8

## Underlying Considerations

1. Sharding
2. Less is More
3. Expect Failure
4. Autonomy
5. Empower the Runtime
6. Favor Stateless
7. Separate Stateless from Stateful
8. Precision vs. Speed



*Conscious trade-offs*



© 2009 Google, Inc. All rights reserved.

9

## Sharding



Divide and Conquer

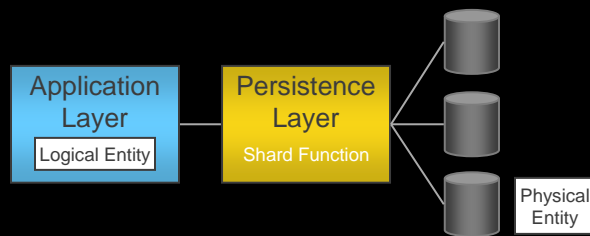


© 2009 Google, Inc. All rights reserved.

10

## Partition Data Across DB Instances

- Shard function, e.g. customer ID
- Hierarchies (one-way) work well
- Many-to-many relationships (two-way) difficult
- "Special Shard" / All shard queries



© 2009 Google, Inc. All rights reserved.

11

## Less is More



Parallelizable instead of Feature Bloat



© 2009 Google, Inc. All rights reserved.

12

## Bigtable, Not Bigdatabase

- **Less is More:**
  - No transactions
  - No schema
  - No foreign keys
  - No join
  - No relational algebra, Cartesian product, etc
  - No SQL
- Single row updates are atomic. Everything else is not.
- Only basic data types: string, counter, protocol buffer



© 2009 Google, Inc. All rights reserved.

13

## Expect Failure



Not If, But When



© 2009 Google, Inc. All rights reserved.

14

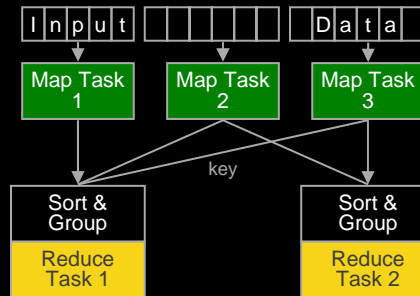
## Expect Failure: GFS & MapReduce

- **GFS:**

- Data replicated 3 times.
- Upon failure, software re-replicates.

- **MapReduce:**

- Restarts failed map / reduce workers
- Detects key/value pairs that cause crashes, skips
- Tougher to deal with: laggards



© 2009 Google, Inc. All rights reserved.

15

## Autonomy



Keep Going without Supervision



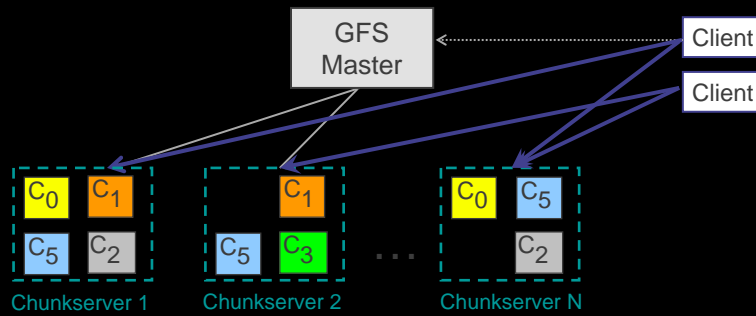
© 2009 Google, Inc. All rights reserved.

16



# GFS

- Direct communication between client and Chunk server
- Large Chunk size (64MB)



© 2009 Google, Inc. All rights reserved.

17

# Empower the Runtime



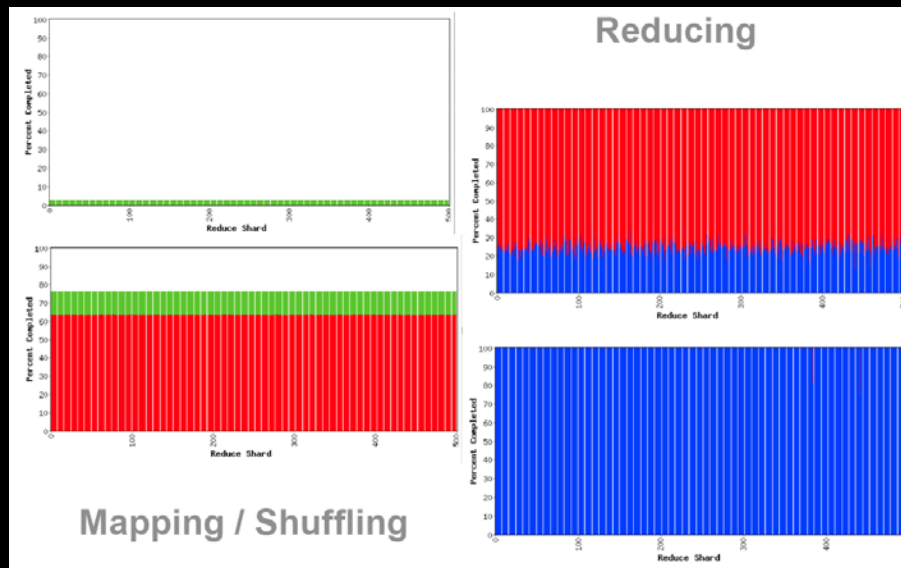
Free Flow Instead of Strict Rules



© 2009 Google, Inc. All rights reserved.

18

## MapReduce Execution



© 2009 Google, Inc. All rights reserved.

19

## Sawzall Quantifiers

- Descriptive as opposed to Prescriptive

```
function(word: string): bool {
  when(i: some int; word[i] != word[$-1-i])
  return false;
  return true;
};
```

- some / all / each



© 2009 Google, Inc. All rights reserved.

20

## Favor Stateless



Next Available over Lasting Relationships

 © 2009 Google, Inc. All rights reserved. 21

## MapReduce / Sawzall / Bigtable

- Map and Reduce step are stateless

```
map(in_key, data)
  → list(key, value)
```

```
reduce(key, list(values))
  → list(out_data)
```


- Sawzall views input data as set, not list
- Bigtable has set() operation vs. insert / update




© 2009 Google, Inc. All rights reserved.

22

## Separate Stateless From Stateful



The Real World is Rarely Stateless

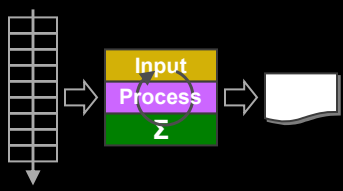
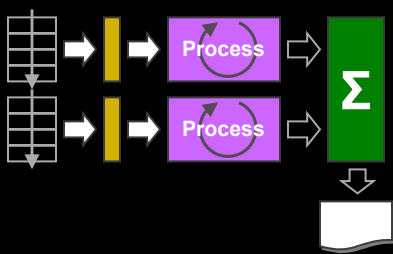
 © 2009 Google, Inc. All rights reserved. 23


## Sawzall

```
while (!eof(file)) {
  line = readline(file);
  words = split(line);
  for each word in words:
    map[word]++;
}
```


```
map: table sum[string] of int;
line = input;
words = split(line);
for each word in words:
  emit map[word] <- 1;
```

State!!

 © 2009 Google, Inc. All rights reserved. 24

## Precision vs. Speed



Faster *is* Better (in Software)

Google

© 2009 Google, Inc. All rights reserved.

25

## Trade precision for speed: Sawzall

- **Top() function:**  
Statistical samplings that record the `top N' data items.

```
// This type is for estimating the most common
// entries based on the CountSketch algorithm from
// "Finding Frequent Items in Data Streams",
// Moses Charikar, Kevin Chen and Martin
// Farach-Colton.
```

## It's all about trade-offs!

- **GFS**
  - Large chunk size (64MB)
  - Optimized for sequential read, not random access
- **Bigtable**
  - Optimized for read-intensive applications
  - Distributed, but not transactional
- **Sawzall**
  - Cannot detect duplicate rows



© 2009 Google, Inc. All rights reserved.

27

