Intro to MapReduce

Mark Stetzer
mark.stetzer@chacha.com
@stetzer
August 23, 2011
ChaCha

Introductions
(aka, who is this guy?)

- Director of Engineering at ChaCha
- Created ChaCha's first Hadoop cluster
- Interested in large-scale data collection & analysis
- At ChaCha, my code powers things like our user profile system, guide matching, and sending SMS messages

Outline

- What is MapReduce?
- How does it work?
- Tips & Tricks?
- Use Cases
- Demo

Obligatory Quote

Every two days now we create as much information as we did from the dawn of civilization up until 2003...That's something like five exabytes of data

-Eric Schmidt

What is MapReduce?

- Programming model meant to simplify data processing on large clusters
- You specify a *map* and a *reduce* function; the framework does the rest
- Programs are inherently parallel; horizontal scalability becomes simple

What is MapReduce?

- Unstructured data, and "semistructured" data are great fits for processing via MapReduce, as opposed to an RDBMS
- Since scaling to process large amounts of data is "easy", we should be able to store a ton of data for processing

How does it work?

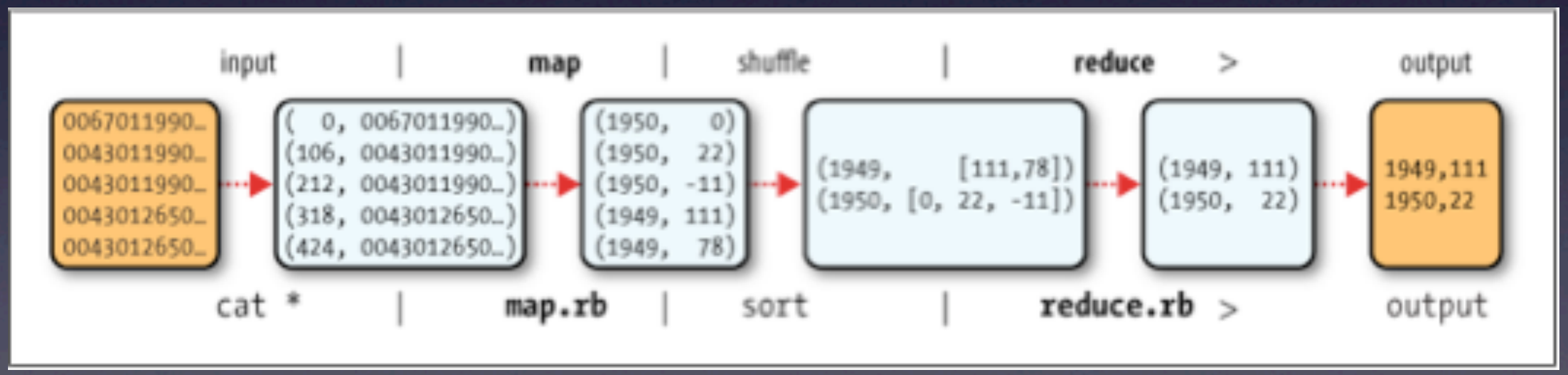
- Hadoop uses HDFS to store data on local disc across large clusters
- This data is kept in large blocks to optimize reading
- Map tasks on the large blocks are run local to their respective data nodes

How does it work?

(input) <k1, v1> -> map -> <k2, v2> -> combine -> <k2, v2> -> reduce -> <k3, v3> (output)

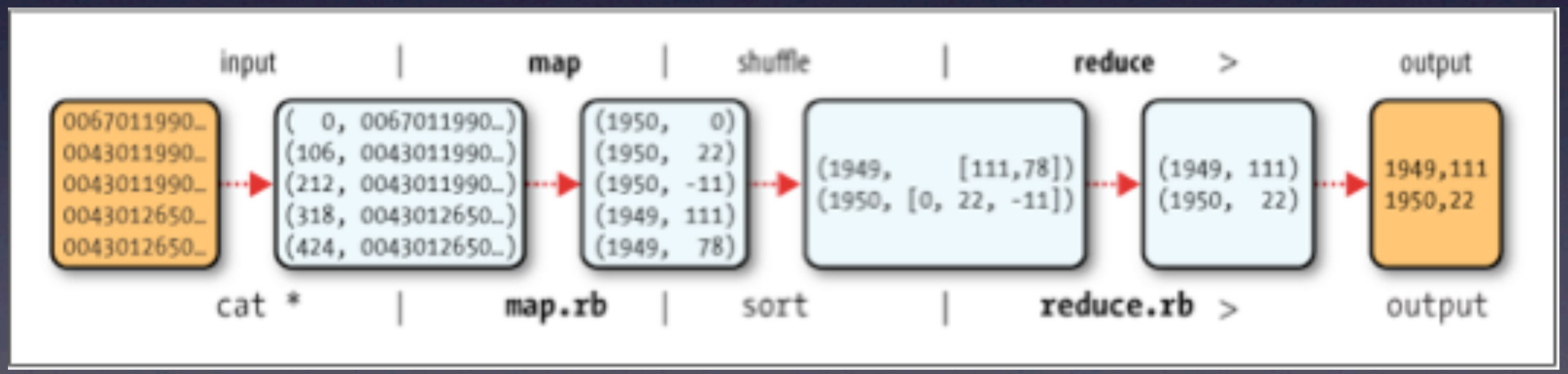
How does it work?

- In the map phase, we're pairing down the data we want to look at



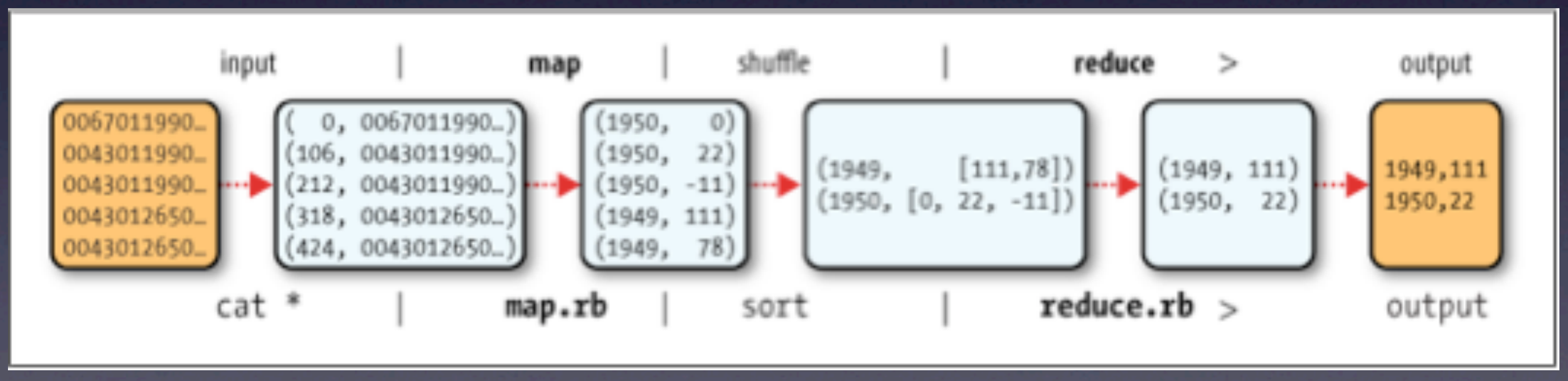
How does it work?

- Next comes the shuffle, where each map output is assigned to a reducer based on its key



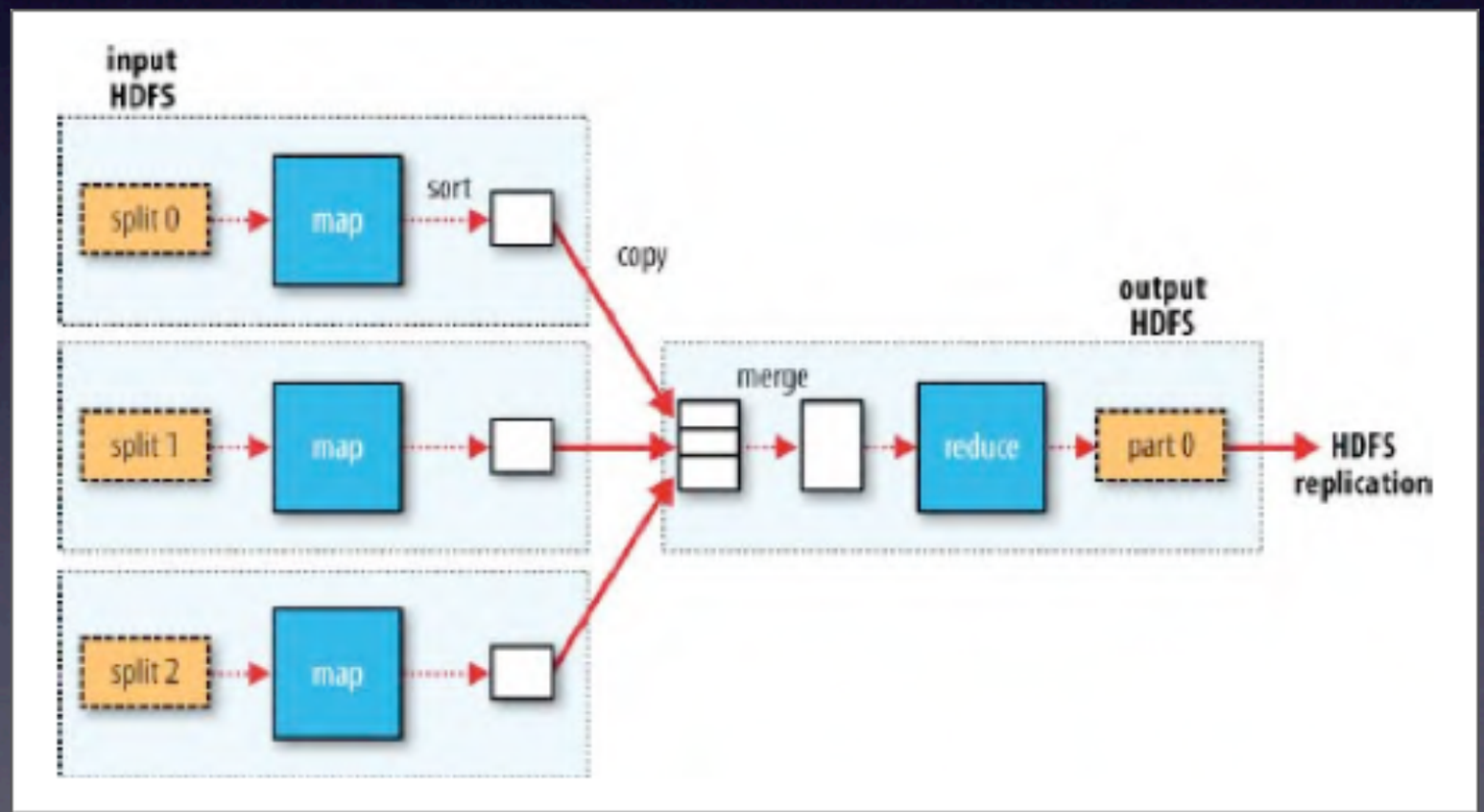
How does it work?

- Finally in the reduce phase, we copy map outputs to their respective reducer nodes, merge the outputs, and apply the *reduce* function to the output



How does it work?

- The output of the reduce is written directly to the output filesystem, typically HDFS



Tips & Tricks?

- In the weather example, isn't it non-optimal to send all temperature values through the shuffle? Couldn't each map output only the max for its given key?
- Specify a Combiner function; function may be called several times
- Pig calls this an Algebraic function

Tips & Tricks?

- What if one or more nodes are misbehaving? Do I have to wait forever for my job to complete?
- Speculative execution will attempt to launch a backup task when a task is running slower than expected
- Turned on by default; can be turned off (cluster-wide or by job)

Tips & Tricks?

- What if my map tasks have an expensive initialization time? What if I need to start a bunch of small map tasks?
- Task JVM reuse will run map tasks *sequentially* in the same JVM
- Can be a performance boost if HotSpot makes optimizations on long-running tasks

Tips & Tricks?

- What if I have job configuration data that is needed to run my tasks (e.g. list of search keywords)? Do I have to serialize them in my job configuration?
- The Distributed Cache can be used to copy data to tasks when they need the data; the cache data is only copied to a node once per job

Tips & Tricks?

- Do I have to store my data as raw text? Won't that use a TON of disk?
- HDFS supports compression at the filesystem level; codec options include zip, gzip, and bzip2
- If you use a splittable format (bzip2), large files can be compressed and still separated into separate map tasks

Use Cases

- Log file analysis probably most common
- NLP at large scale
- Machine learning at scale
- Full text search
- Projects like HBase allow for a distributed datastore capable of very high write volumes

Huzzah! Demo!

Questions?

- The floor is open!
- Also accepting of feedback!

Thanks!

- mark.stetzer@chacha.com
- @stetzer
- ChaCha is hiring engineers!