



Cloud Computing

COMP / ECPE 293A

Google

MapReduce

Jeffrey Dean and Sanjay Ghemawat, “MapReduce: simplified data processing on large clusters”, In *Proceedings of the 6th conference on Symposium on Operating Systems Design & Implementation (OSDI'04)*, Berkeley, CA, USA, 2004

Imagine You're Google



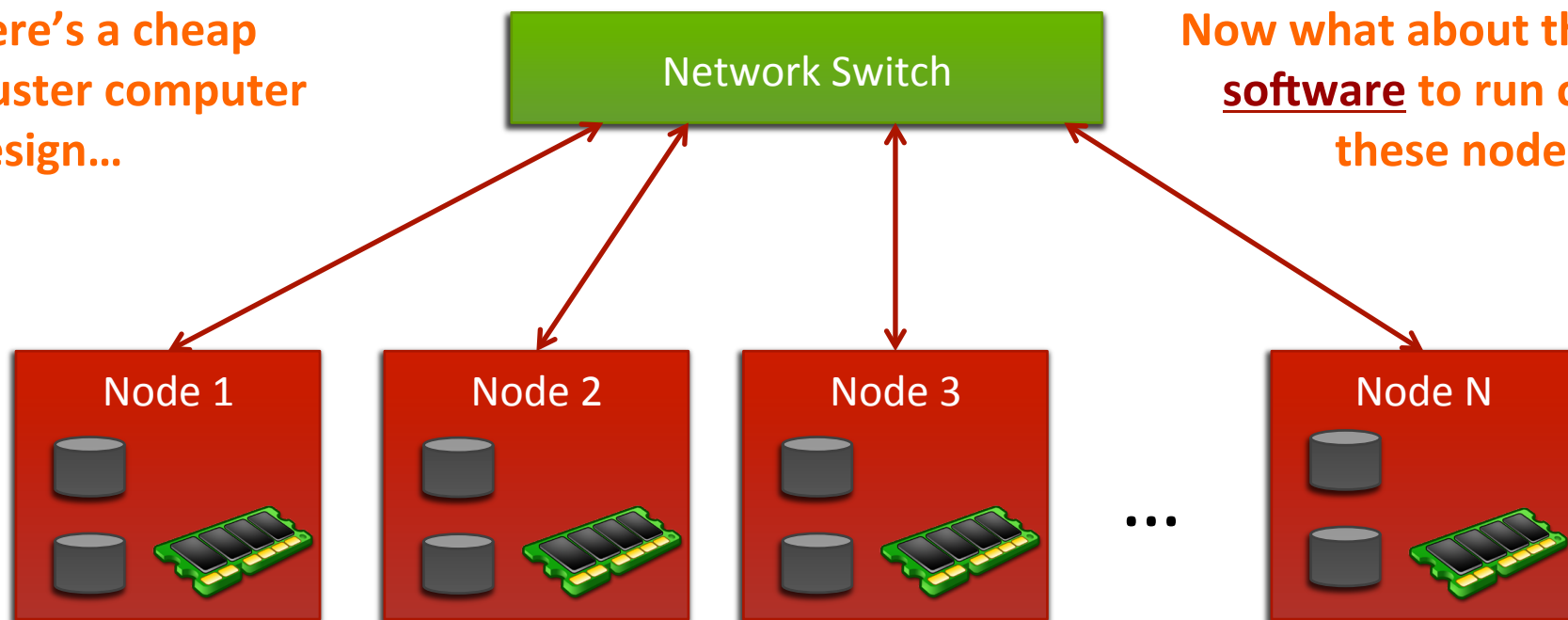
- Yesterday's job: Download web pages
 - 40 billion web pages x 40kb per page = 1.4PB of data



- Today's job: Make sense of the data!
 - Imagine the data is stored on one (hypothetical) hard drive
 - Typical hard drive read bandwidth: 100MB/sec
 - **~6 months for one computer to read in the full data set**
- **Obvious conclusion**
 - **This problem needs to be parallelized**
 - **Let's build a cluster computer!**

Imagine You're Google

Here's a cheap cluster computer design...



Now what about the software to run on these nodes?

- Commodity = **cheap** servers
 - Current price "sweet spot"

- Configuration
 - x86 CPUs
 - IDE (now SATA) hard drives
 - Gigabit Ethernet

Imagine You're Google

- Parallelizing programs across a cluster computer is **hard work** (*harder than for a multi-core CPU*)
 - One existing method: MPI (message passing interface)
 - Popular with supercomputer applications
 - API provides fine grained control over all communication between processes running on different nodes

- **Challenges**
 - Communication, coordination, data replication, recovery from failures, debugging, performance optimizing
 - **You need to do this for every program you write**



MapReduce Vision

Tradeoff: Limit the style of programs you can write, but provide automatic parallelization and infrastructure support in return!

MapReduce Vision

What You Do

- Write a program in a special way that can be trivially parallelized
 - **Literally write “map” and “reduce” functions!**

What the Framework Does

- **Distributes your data** across the cluster
 - Replication
- **Distributes processing** across the cluster
 - Scheduling jobs
- Manages **communication between nodes**
- Detects and **recovers from faults**
 - Re-runs failed tasks
- Provides monitoring and administration tools

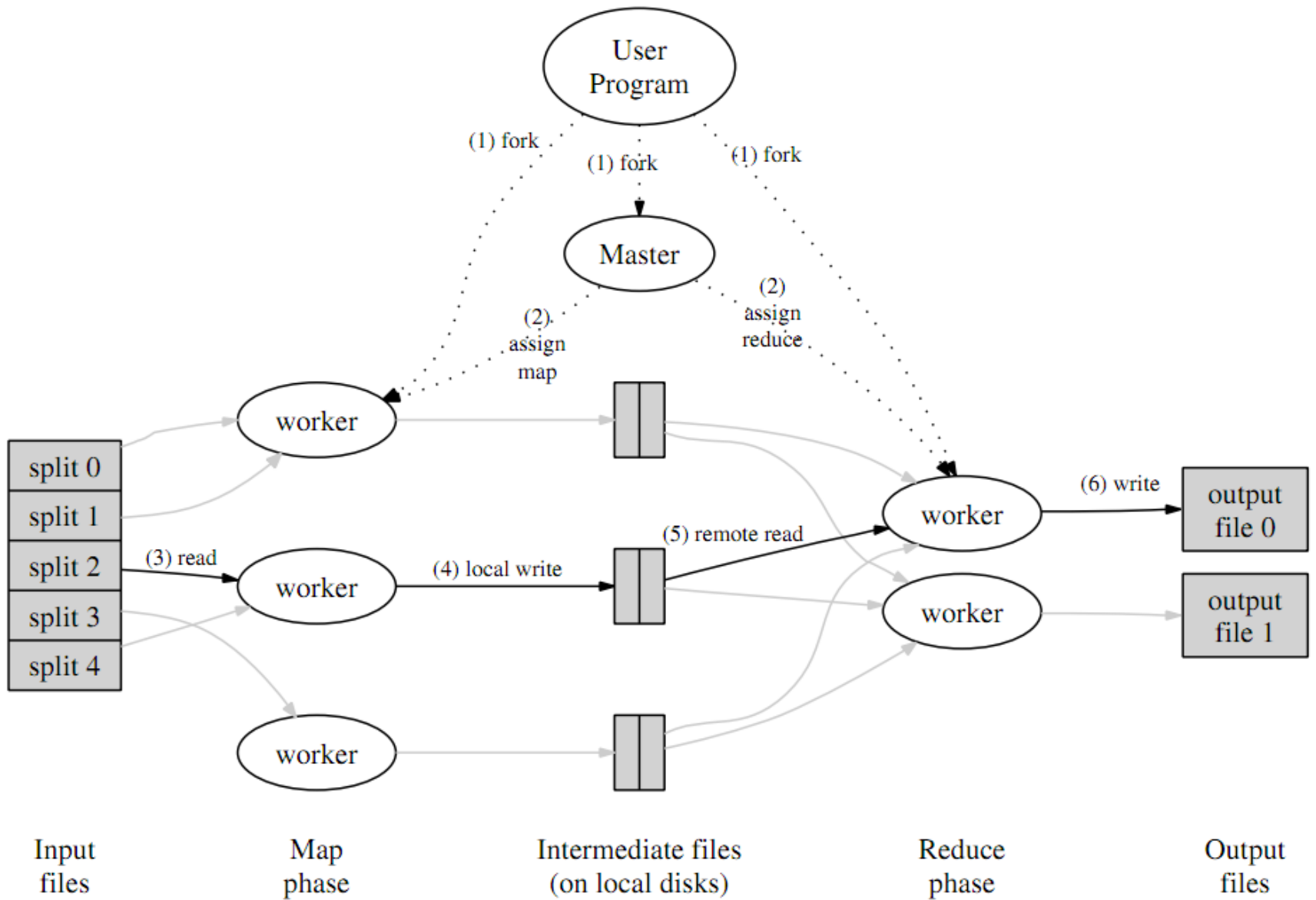


Figure 1: Execution overview

Count of URL Access Frequency From Web Logs (with URLs A-F)

Original input data:

A D F C E B B A D F E F E A B

Master splits data into M pieces:

A D F C E

B B A D F

E F E A B

Master starts M map tasks on the cluster.

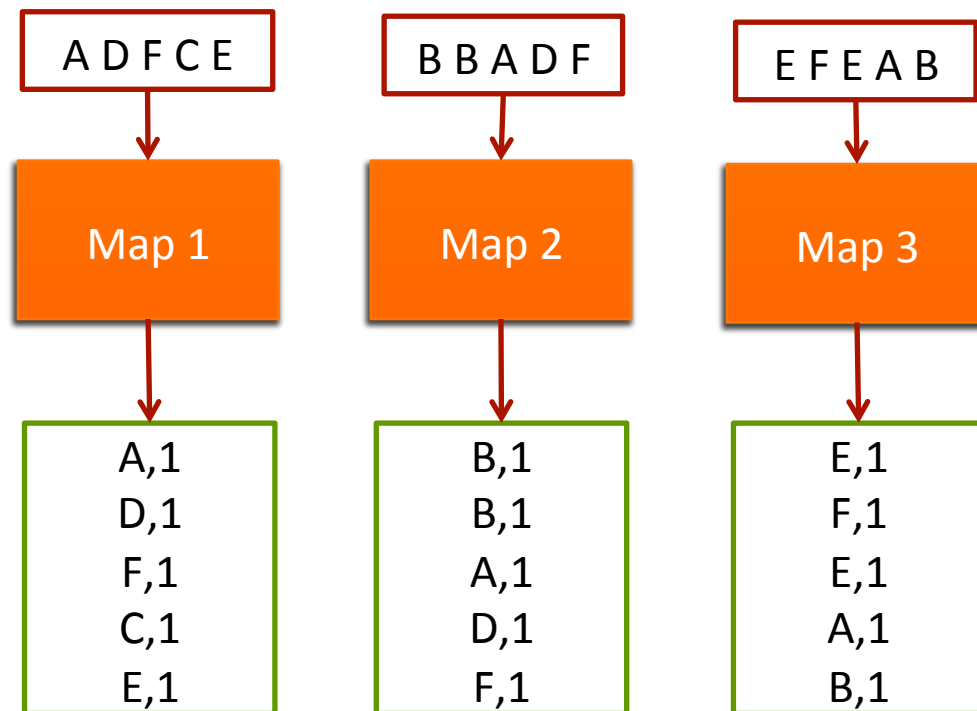
(Try to assign based on data locality. Some tasks might wait for idle nodes)

Each **map task**

- (1) Reads its own **data subset**
- (2) Parses the data into key/value pairs
- (3) Emits *intermediate* key/value pairs

Ideally, this is all on local data from the local disk! Nice and parallel...

For this program, each URL is emitted with a count of 1 (e.g. 1 hit for URL A)



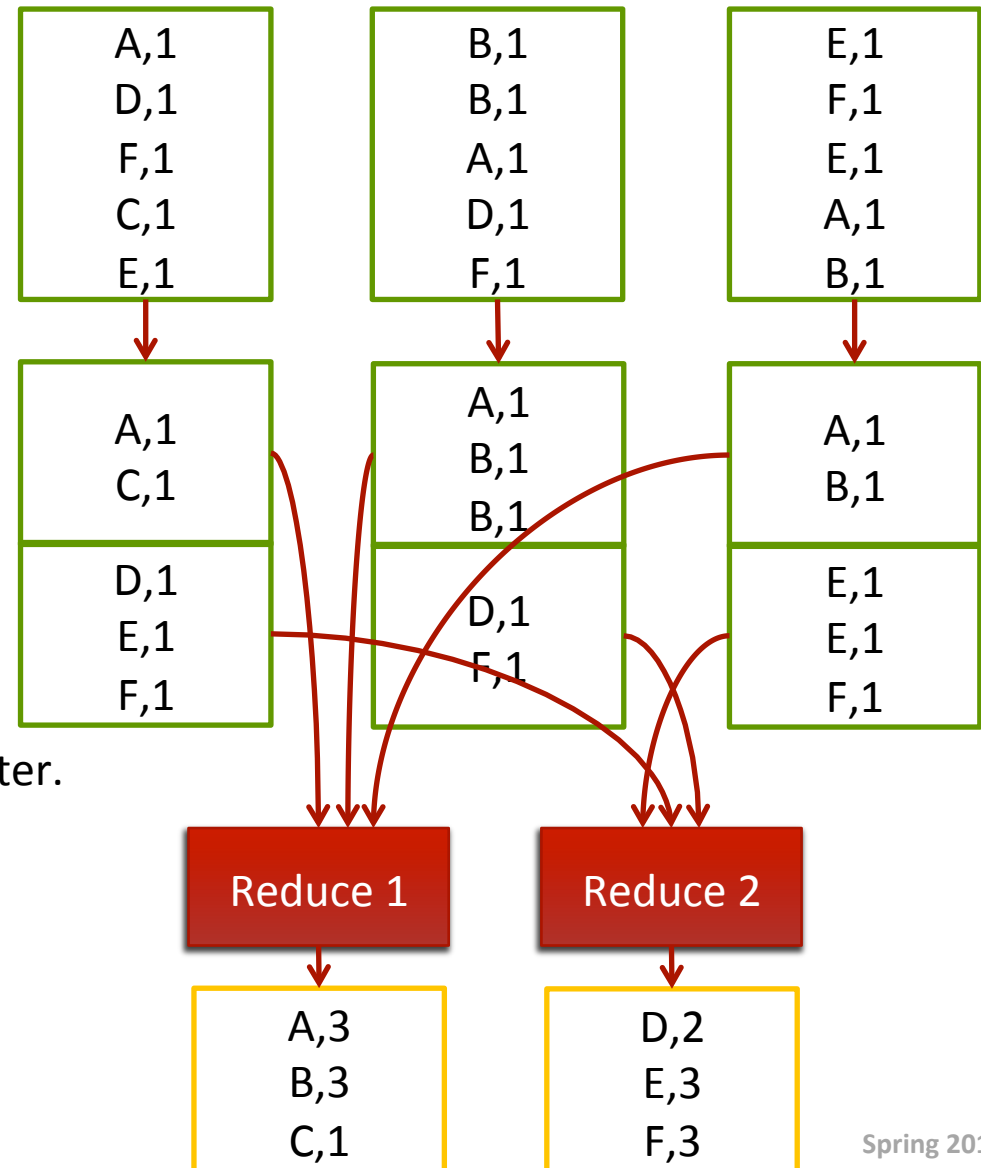
Count of URL Access Frequency From Web Logs (with URLs A-F)

Intermediate key/value pairs:

Intermediate pairs are partitioned into R partitions. Partitions are sorted by key order. (Either in RAM or on disk if huge) – *Framework does this for you!*

Master starts R Reduce tasks on the cluster. Reduce tasks pull in intermediate data (this is across the network)

For this program, Reduce tasks sum up total hits per each URL



Performance

- Skipping details for this talk...
 - *You may want to make a different decision depending on your paper selection...*
- Key question is **not**: Is this the absolute fastest we could perform this job?
- Rather, key question is: Is this the fastest way we can program the job with reasonable efficiency?
 - *We can always buy a few more computers!*

Importance of Locality

- Network *could* be a huge bottleneck in this type of system
- Goal: Process data that is stored *locally* (on your nodes' hard disk)
- Solution
 - Master knows how data is divided and tries to assign map tasks to the same node that stores data
 - Or at least nearby – same rack?
 - Minimizes network communication

Fault Tolerance

- Master probes the other nodes periodically
- Node failed?
 - Master re-assigns tasks to other nodes (which have replicated data)
- What if the master fails?

Optimizations

- Backup tasks
 - Don't wait for a handful of slow (buggy, failing) nodes – re-execute those tasks somewhere else!

- Skipping bad records
 - Many apps don't care if a few data points are corrupt in a data set measuring in the terabytes

I'm Not Google!

- Can people outside of Google benefit from MapReduce and related technologies?
- **Yes!** Open-source clone of MapReduce:



- Literally, some engineers read the Google paper and said “We can build that!”
- Now used at Yahoo, Facebook, eBay, Last.fm, ...

Discussion Questions

- What type of applications are suitable for MapReduce?
- What apps are unsuitable for MapReduce?
- What type of API would you like for these apps?
- Strengths and weaknesses of the paper?