



MapReduce: Simplified Data Processing on Large Clusters

Peter Pietzuch

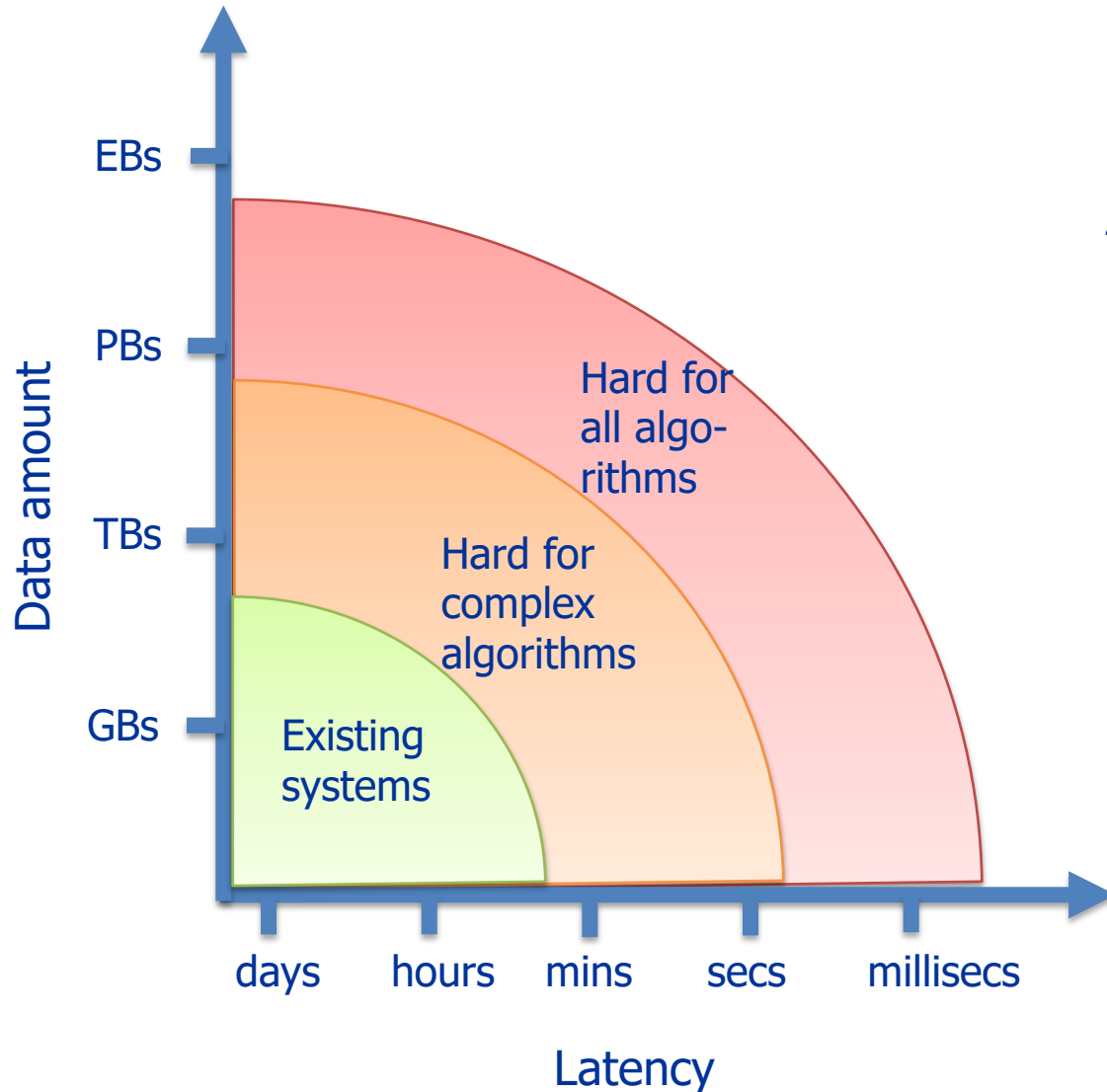
prp@doc.ic.ac.uk

Department of Computing
Imperial College London

<http://lsds.doc.ic.ac.uk>

Design Space: Big Data Processing

Volume and Velocity



Algorithmic complexity

- Arbitrary data transformation
- Iterative algorithms
- Large state as part of computation

MapReduce Overview

Mainstream “Big Data” analytics framework

- E.g. PageRank (web search indexing)
- Made popular by Google in 2004
- Open source version (Hadoop) by Yahoo/Apache



Sanjay
Ghemawat

Jeff
Dean

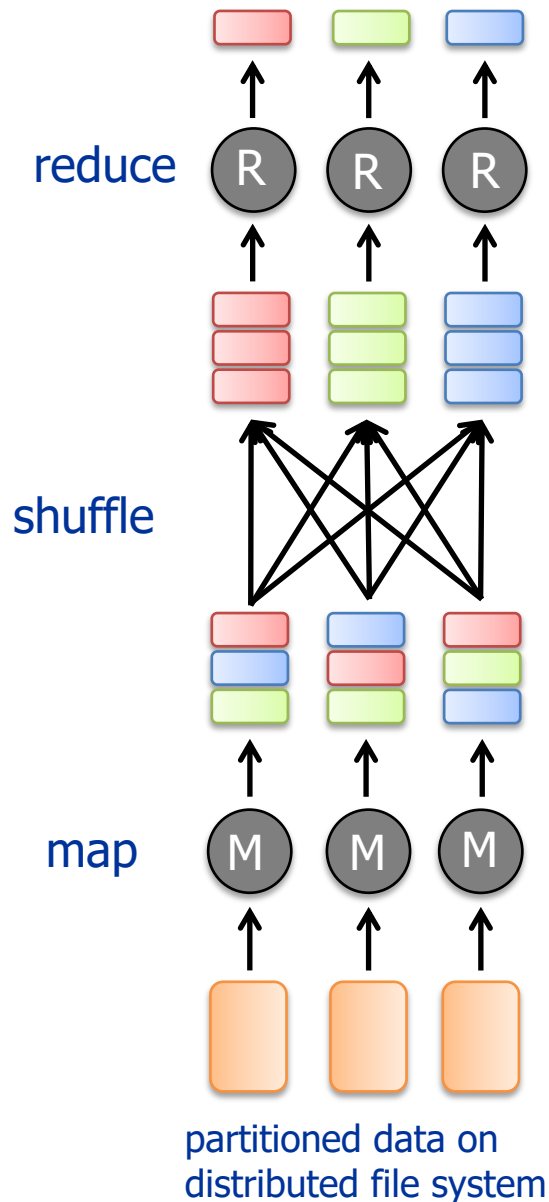
Benefits:

- Simple programming model
- Transparent parallelisation
- Fault-tolerant processing



\$2 billion market revenue (2013)

MapReduce: Distributed Dataflow



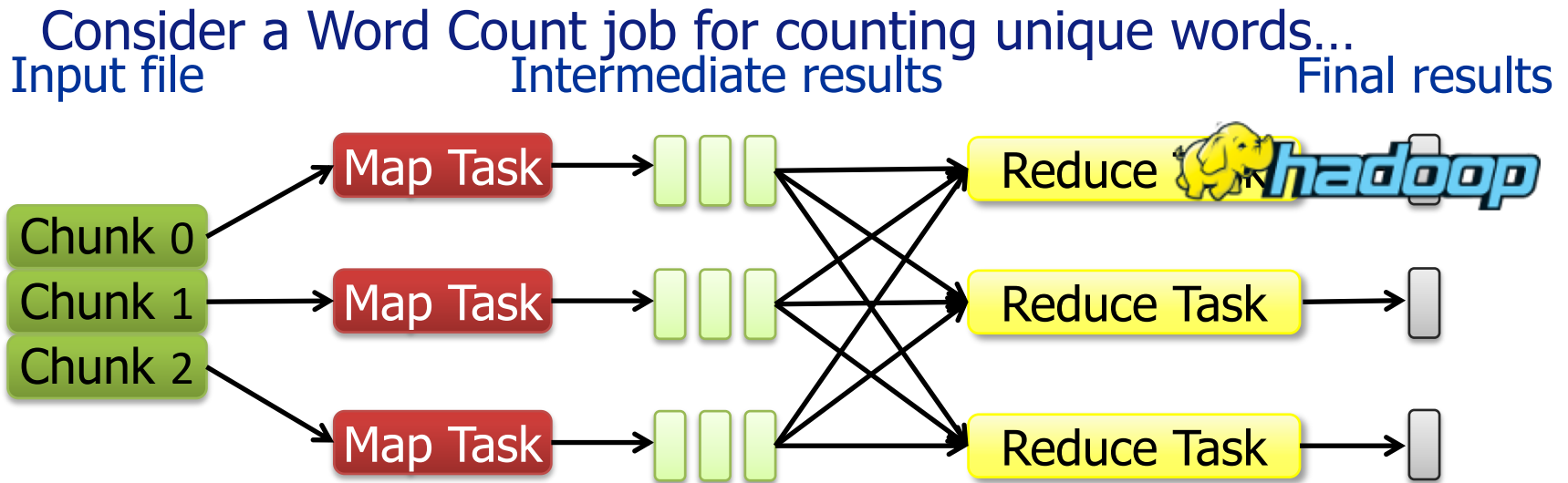
Data model: (key, value) pairs

Two processing functions:

$\text{map}(k_1, v_1) \rightarrow \text{list}(k_2, v_2)$

$\text{reduce}(k_2, \text{list}(v_2)) \rightarrow \text{list}(v_3)$

Example: Wordcount



Map: Processes input data and generates (key, value) pairs

- e.g. {(cat, 5), (dog, 7), (elephant, 9)}

Shuffle: Distributes intermediate pairs to reduce tasks

- e.g. all words starting with 'A' to reducer 1, those with 'B' to reducer 2

Reduce: Aggregates all values associated to each key

- e.g. sum all values for word "cat", all values for word "dog"

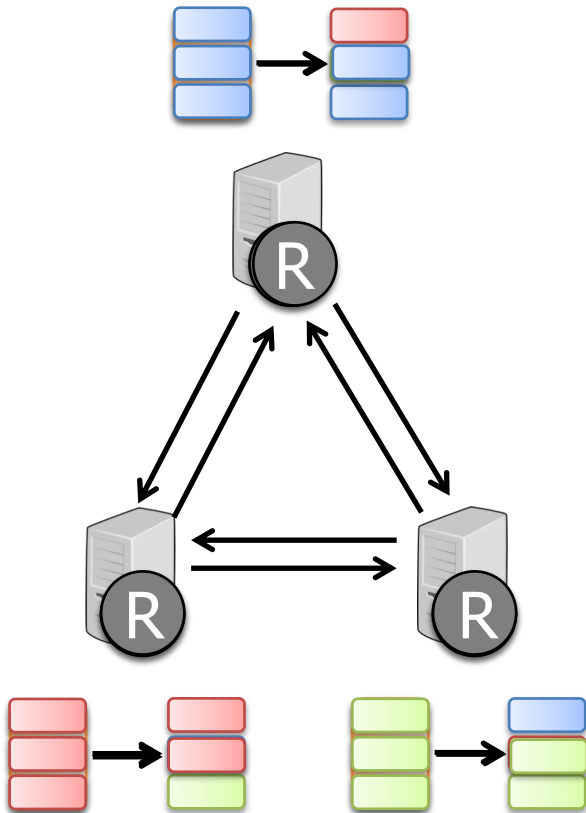
MapReduce Execution Model

Map/reduce tasks **scheduled** across cluster nodes

- Locality-aware scheduler

Intermediate results **persisted** to local disks

- Final results of MapReduce job stored in GFS



Failure Recovery

1. Task failure: restart task

- Map tasks fetch data from GFS
- Reduce tasks fetch intermediate results from local disks

2. Node failure: restart tasks on new node

- Need to re-run all tasks because intermediate results are lost

Speculative Execution

MapReduce jobs dominated by slowest task

Speculative execution:

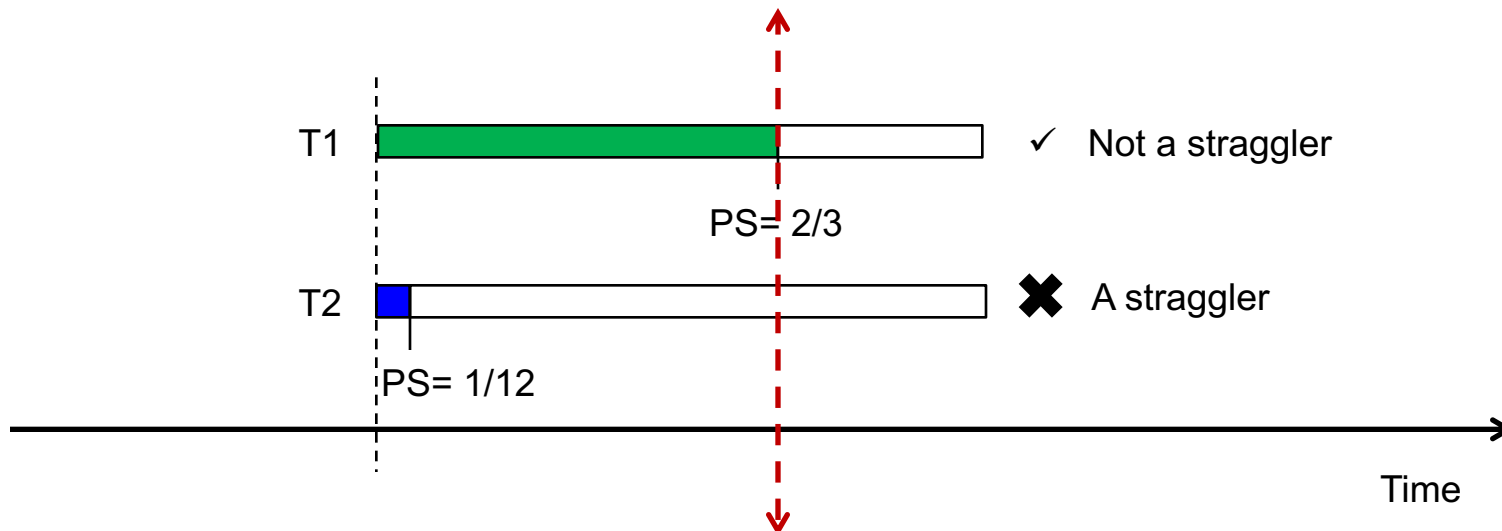
MapReduce attempts to locate slow tasks (stragglers) and run redundant (speculative) tasks

- Hope that speculative tasks finish before stragglers
- Only one copy of straggler allowed to be speculated

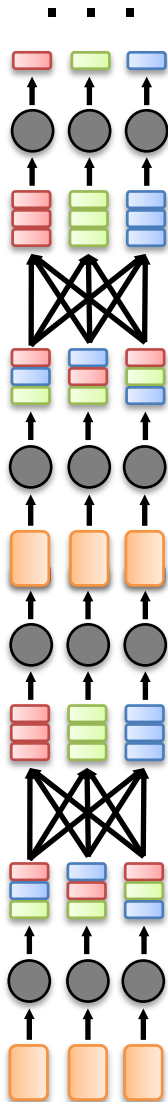
Locating Stragglers

How does Hadoop locate stragglers?

- Hadoop monitors each task progress using progress score between 0 and 1
- If task's progress score less than (average – 0.2) + task ran for at least 1 minute → mark as straggler



Support for Iteration



Iteration useful for many algorithms

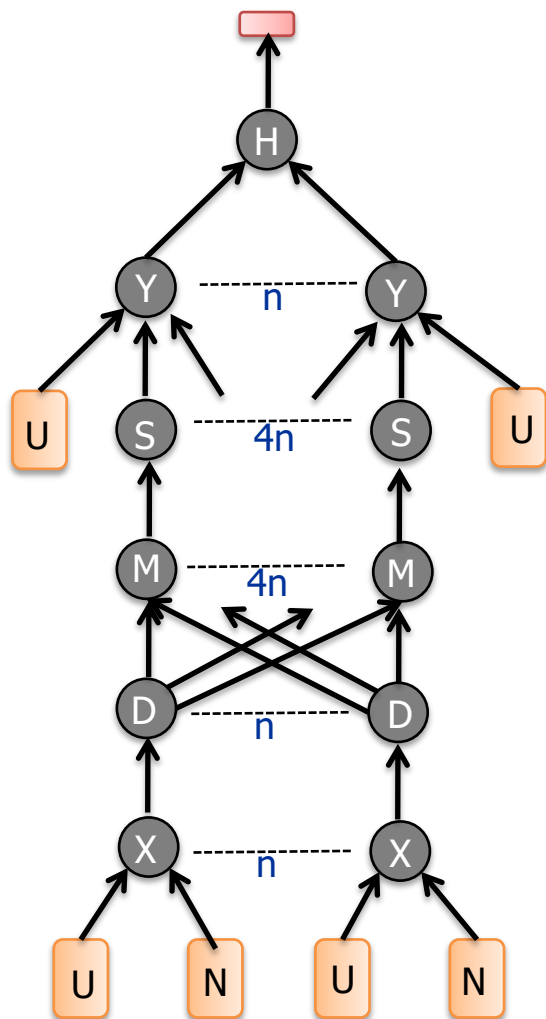
- Machine learning/data mining algorithm (pagerank, k-means, logistic regression, ...)

Loop unrolling: multiple MapReduce jobs

Challenge:

- Materialisation of intermediate results becomes expensive

Dryad: Dataflows as DAGs



Arbitrary functions as tasks

- Eg Joins, group by etc

Dataflow graph \rightarrow directed acyclic graph

Same model as Spark...

SkyServer query:
3-way join to find gravitational
lensing in astronomy data