UNIVERSITY OF CAMBRIDGE

WEDGE: A Wide-area Efficiently Distributed Graph Engine

Presenter: Karthik Nilakant

Supervisor: Dr Eiko Yoneki

MSN ‘12

13^3h April 2012

Motivation

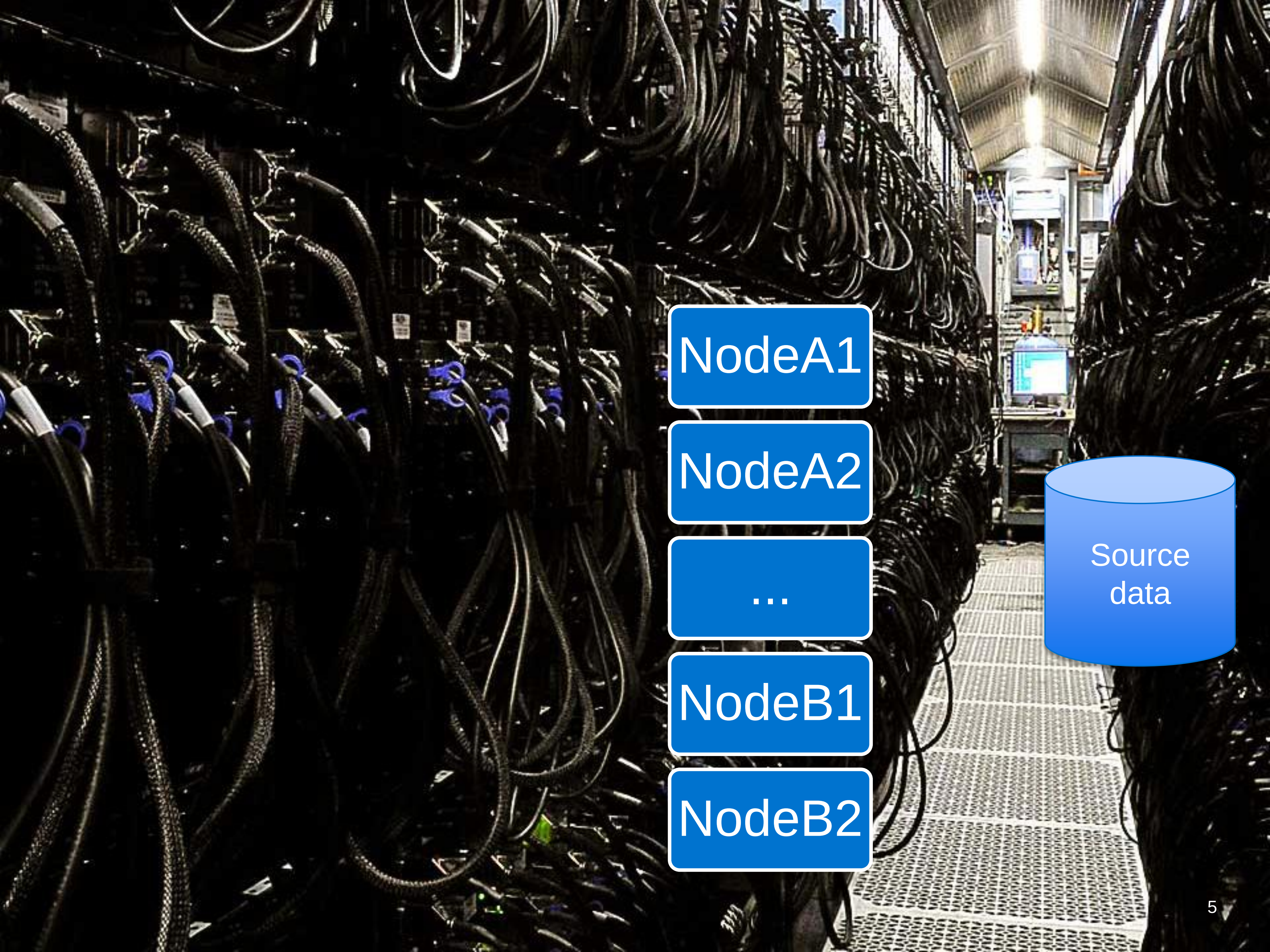
"Wide area" ***There are a variety of use cases for geographic dispersion of data:*** ***•Data may be generated globally*** ***•Want to take advantage of regional price differences in cloud computing services*** ***•Energy saving*** ***•Hybrid clouds*** ***HOWEVER current systems focus on single datacentre environments***	***"Graph processing"*** ***Many large datasets are graph-structured – e.g. The web graph, social networks, transport / traffic, recommendation networks, bioinformatic data.*** ***Users have a need to process this data in various ways (searching, ranking, clustering, etc).*** ***HOWEVER, the predominant map-reduce model is not well-suited to this.***

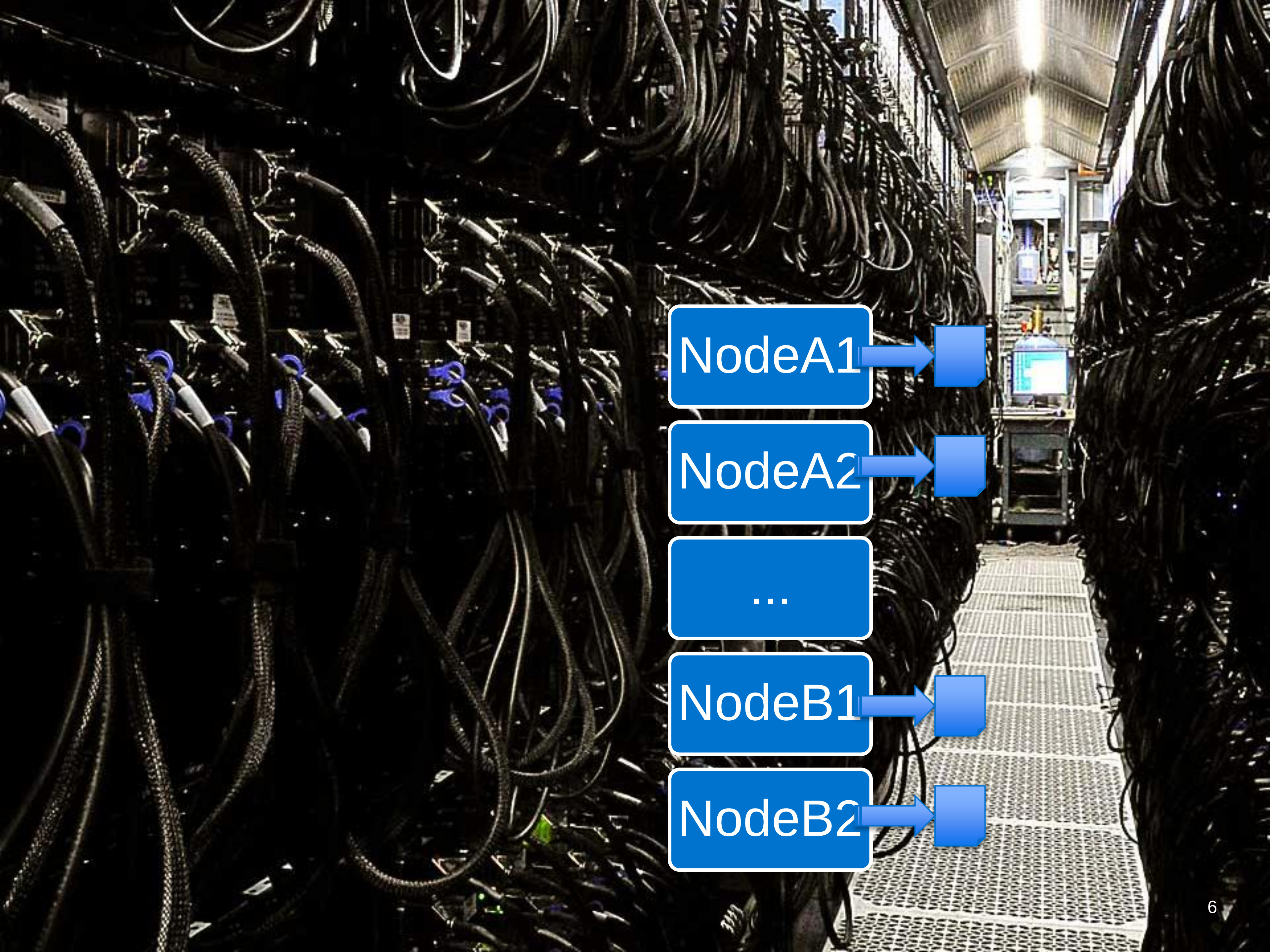
The Data Locality Challenge

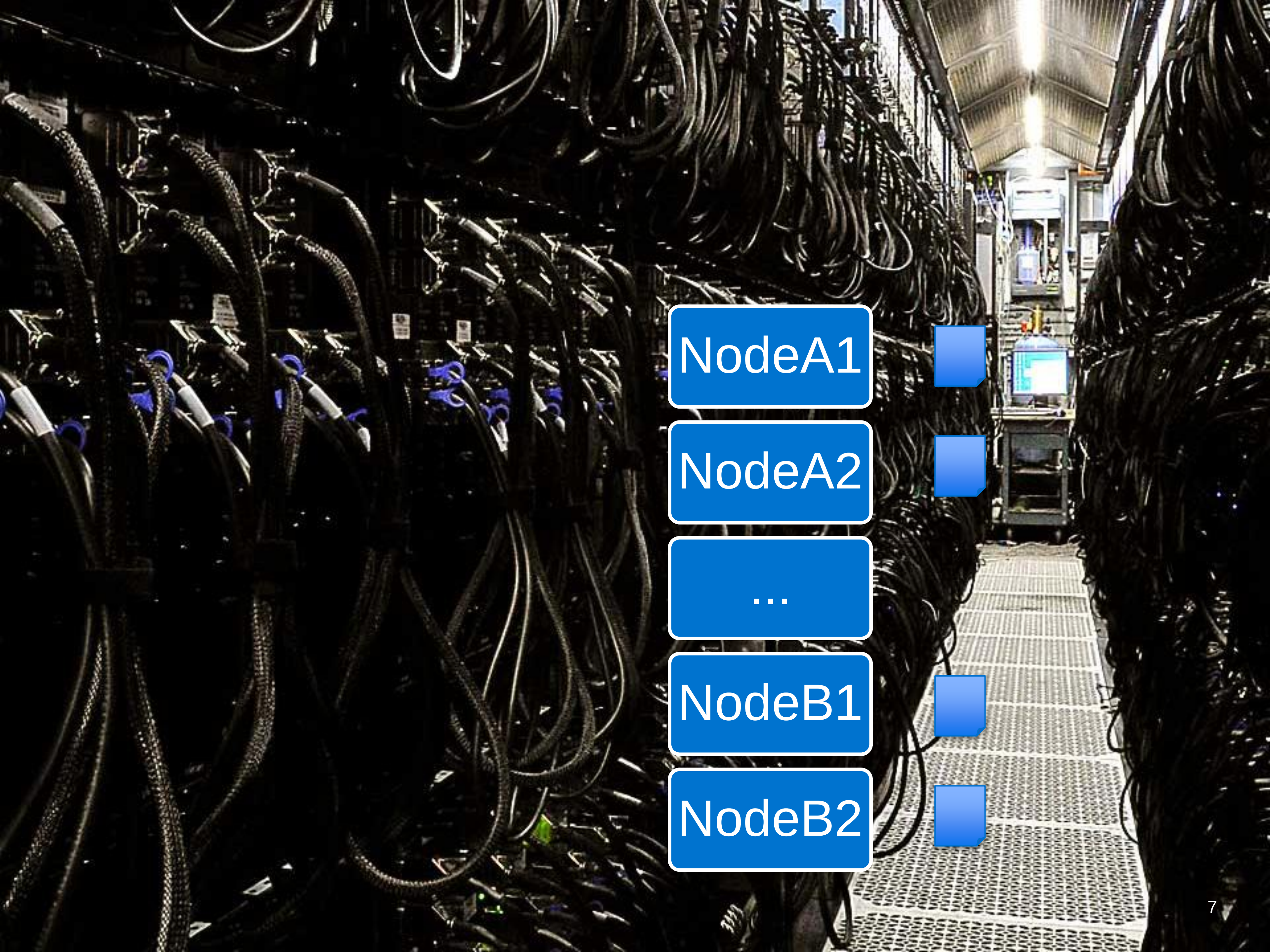
- “Data-parallelism” is at the core of processing frameworks such as Hadoop.
  - We parallelise by splitting up data and processing with independent tasks. This is usually a local operation.
- The challenge arises from the need to collate the results (“reduction”). In the map-reduce model, this requires shuffling.

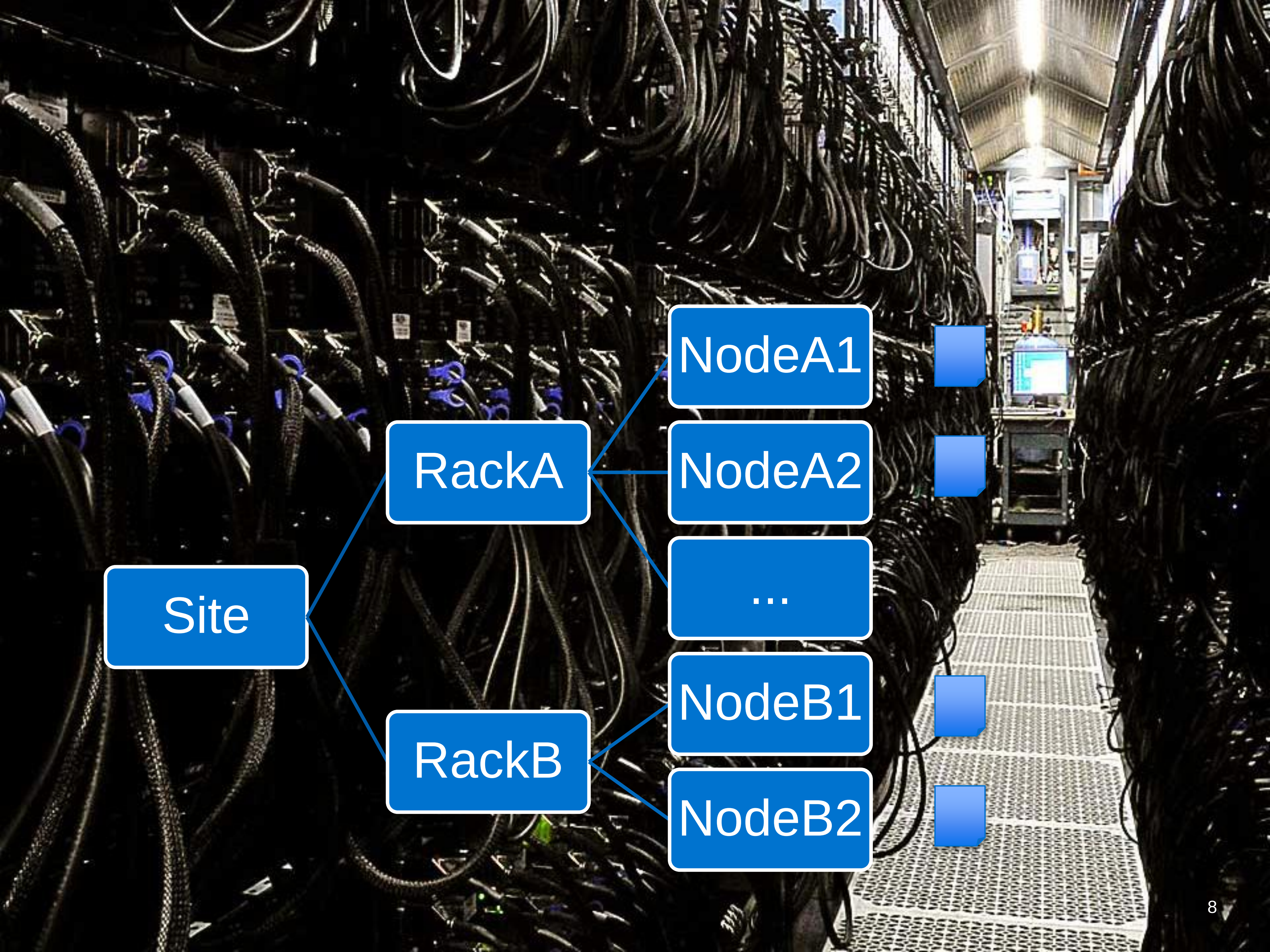
Brief overview of MapReduce

- map() takes an *input split* and produces zero or more *intermediate key-value pairs*.
- reduce() (or "fold") takes a set of intermediate key-values with a common key, and produces output.
- The framework arranges for source data to be distributed amongst a group of mappers, and then arranges for intermediate data to flow from mappers to one or more reducers


NodeA1
NodeA2
...
NodeB1
NodeB2
Source data


NodeA1
NodeA2
...
NodeB1
NodeB2


NodeA1
NodeA2
...
NodeB1
NodeB2

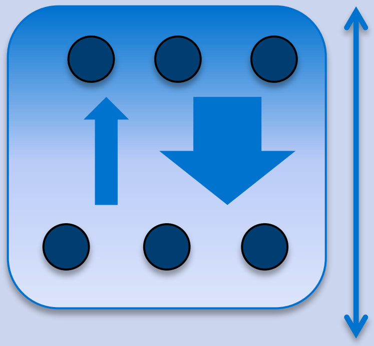


Site
RackA
RackB
NodeA1
NodeA2
...
NodeB1
NodeB2

Example

"Unbalanced Reduction"

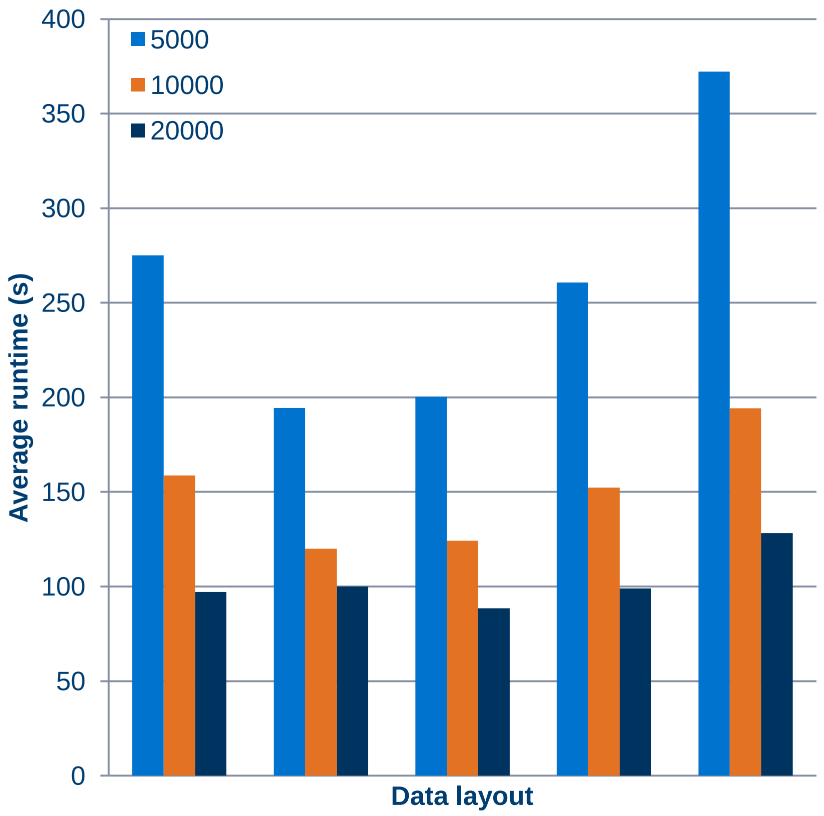
Increased traffic to remote reducers

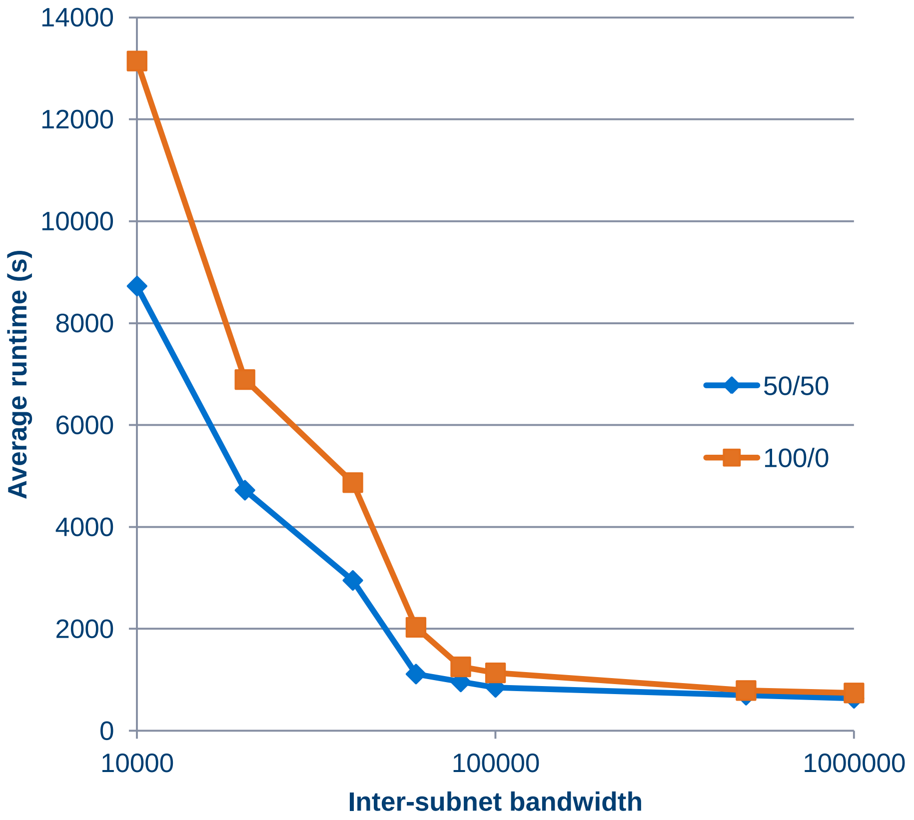


Uneven distribution

- Trade-off between work distribution and data locality
- How to partition key-values?
- *Programming constraint: all instances of a key map to single reducer*
- Attempts to improve locality may affect the functionality of the task

Effects of imbalance



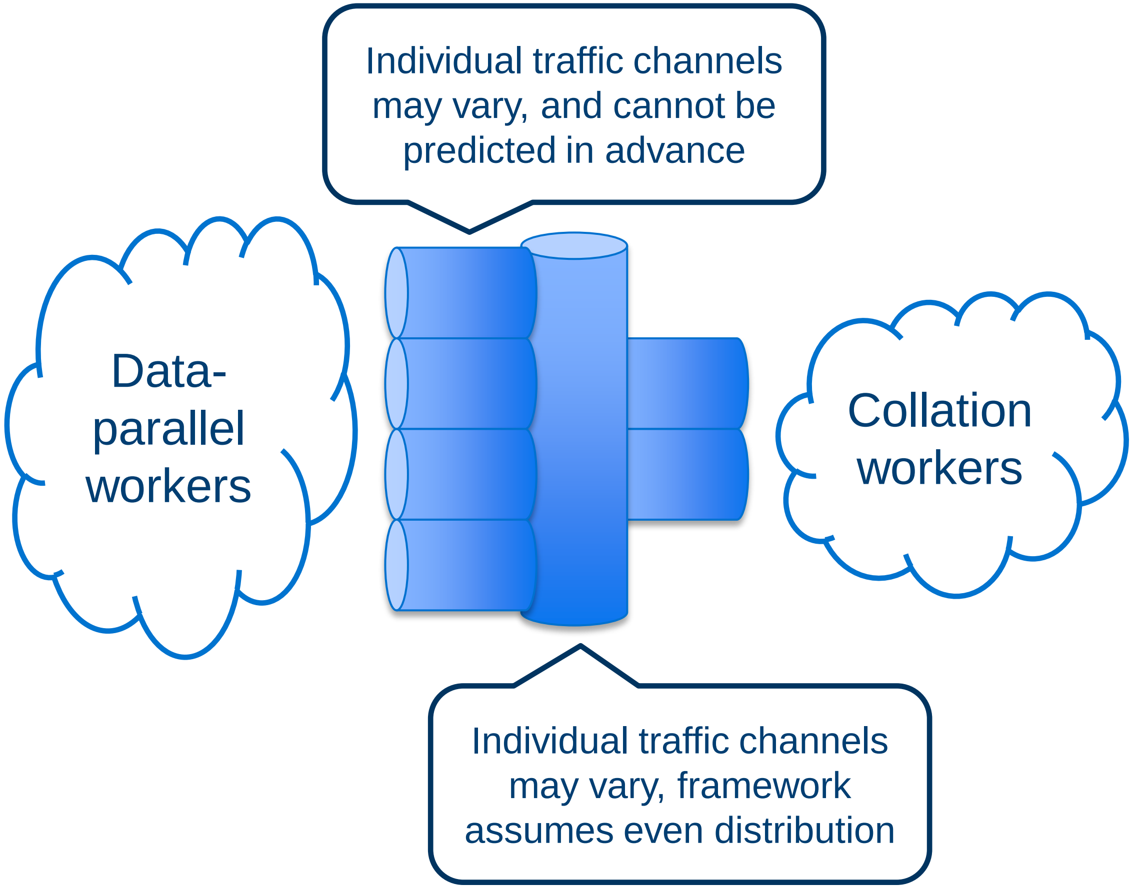


UNIVERSITY OF CAMBRIDGE

Current mitigation strategies

- Framework tuning
  - Problematic due to large parameter space; may be counterproductive (requires trial and error)
- Programmatic
  - Custom partitioning; custom combiners; domain-specific optimisation
- Reactive scheduling
  - Straggler detection (LATE etc)
  - Speculative execution

But the core problem remains...



Unwieldy for the wide-area: we can't identify individual traffic flows.

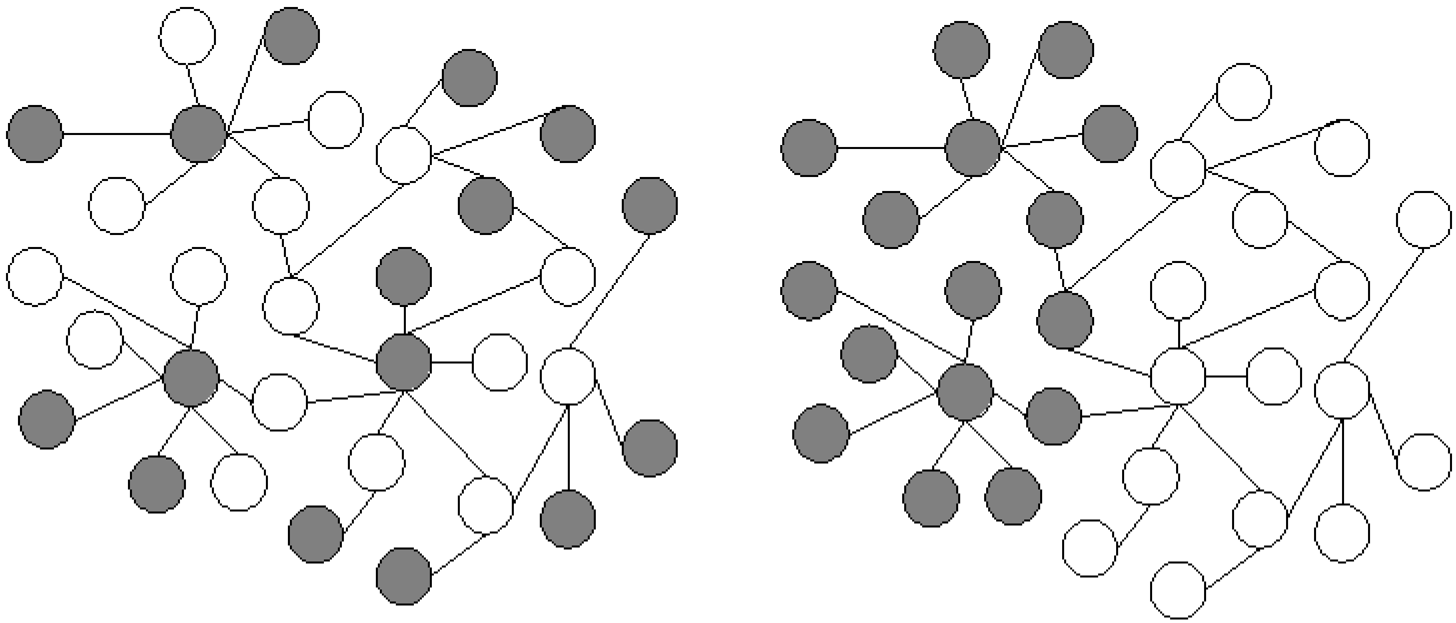
Unwieldy for graph processing: need to pass state between iterations

Bulk Synchronous Parallel

- BSP offers an alternative approach for graph processing (as used by Pregel)
- Step 1: For each vertex in the graph, we process local data in parallel.
- Step 2: Each vertex then generates updates to be passed to immediate neighbours in the graph.
- Step 3: We impose a synchronisation barrier to ensure all updates are propagated.
- Step 4: Iterate.

Insight: shuffling becomes message passing

- How can we minimise the amount of message passing in this model?
- Answer: re-partition the graph



Approaches to graph partitioning

- In HDFS-style systems, typically a background process is responsible for rebalancing data.
- In WEDGE, a similar process would operate on the graph, by identifying optimal partitions and re-organising data.
  - Could use minimal/balanced cut approaches (such as ParMETIS). These produce well-partitioned graphs, but are computationally expensive.
  - Community detection methods offer a heuristic approach based on modularity maximisation. Less overhead, but possibly lower yield.
  - Could offer a range of strategies to the user.

WEDGE: The broader framework

- Three desirable properties:
  - Algorithmic flexibility
    - CIEL offers the building blocks.
  - Partition-tolerance
    - Need to re-fashion CIEL to deal with the wide area
  - Locality optimisation (graph partitioning)
    - Need to integrated with existing resource-management capability

Summary

- There are a range of motivating factors for wide-area graph processing
- Hadoop is neither well-suited to graphs nor the wide area
- We need to explore alternative programming models – BSP is one possibility.
- We can optimise message passing in the BSP model by re-partitioning the graph.
- Community-structured partitions should also facilitate efficient graph processing in other programming models.