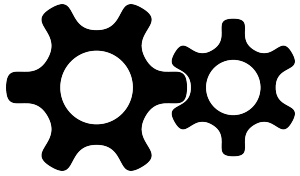
Pufferbench: Evaluating and Optimizing Malleability of Distributed Storage

Nathanaël Cheriere, Matthieu Dorier, Gabriel Antoniu

PDSW-DISCS 2018, Dallas

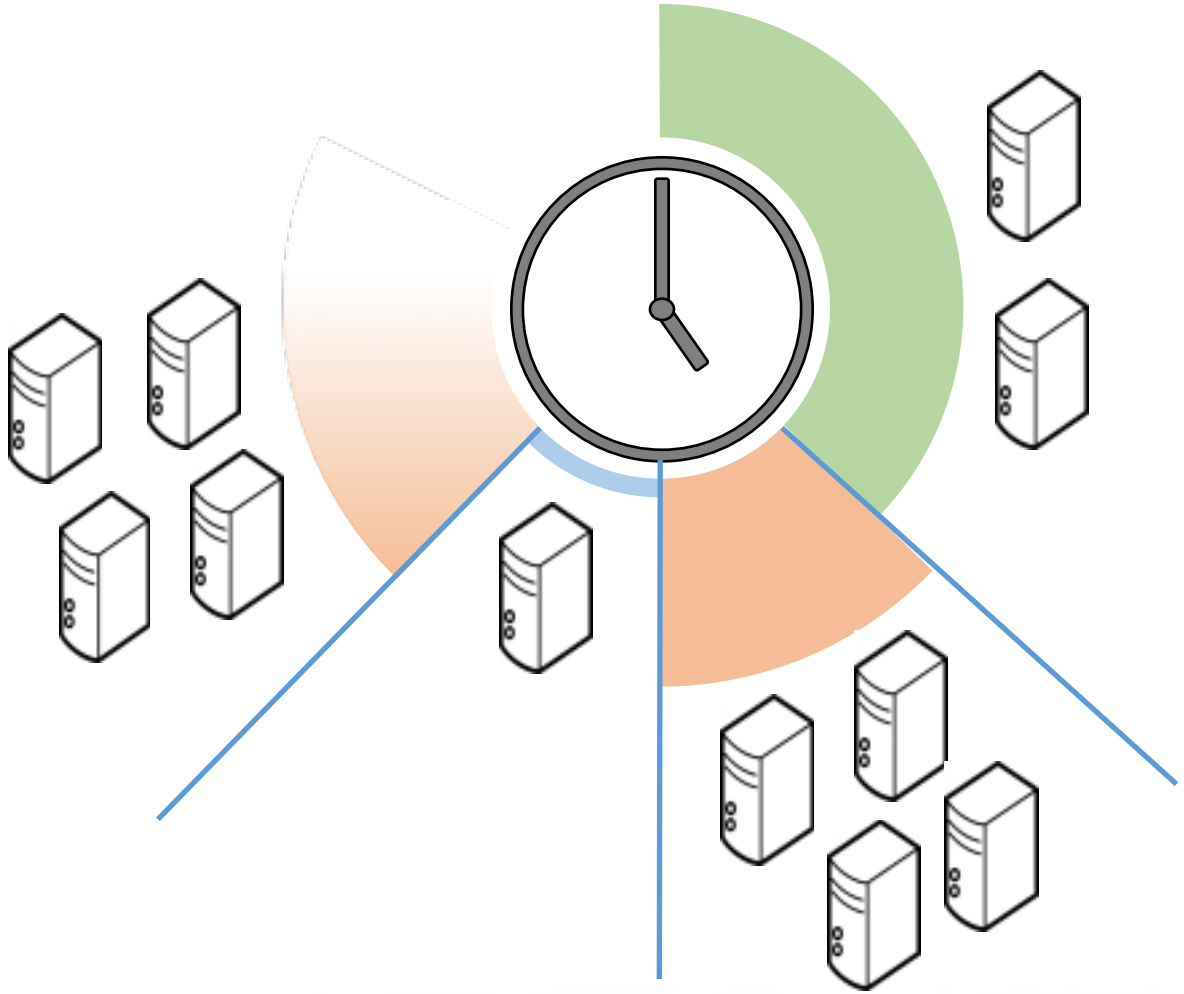
Data is everywhere



High variety of applications

High variety of needs

Resource requirements vary in time



Day/night cycles

Weekly cycles

L	M	M	J	V	S	D
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	1	2	3	4

Workflows

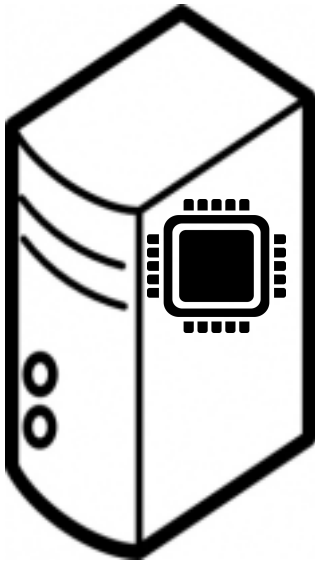
Dynamically adjust the amount of resources?

Why?

- Satisfy resource requirements
  - Peaks
  - Low
- Avoid idle nodes

✓Save money

✓Save energy

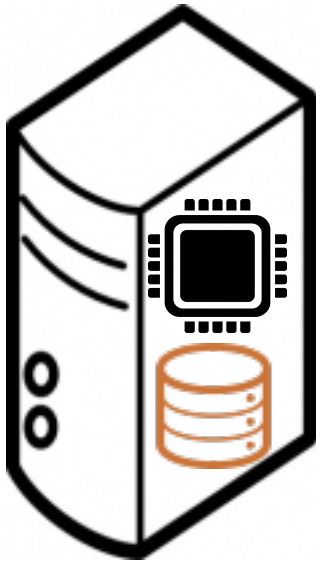


✓Computing resources malleability

Problem:

What about task/data colocation?

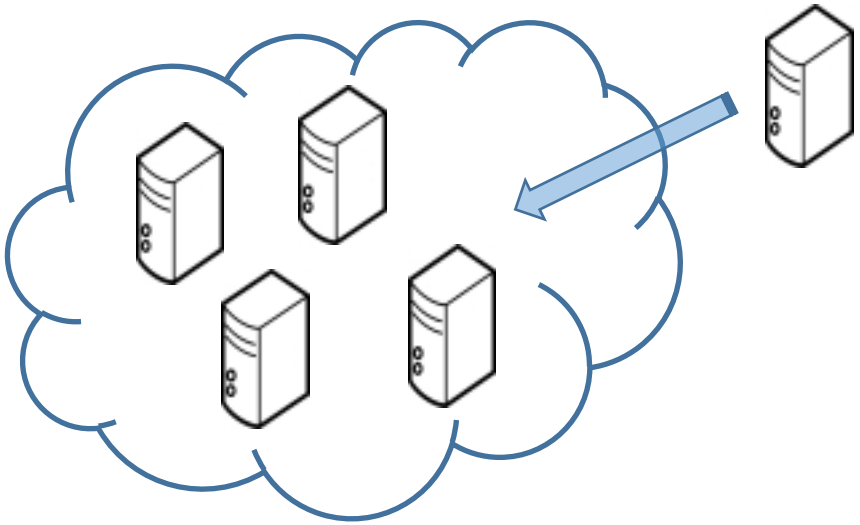
- Local data access
- Easy scalability



? Storage system malleability

Two operations:

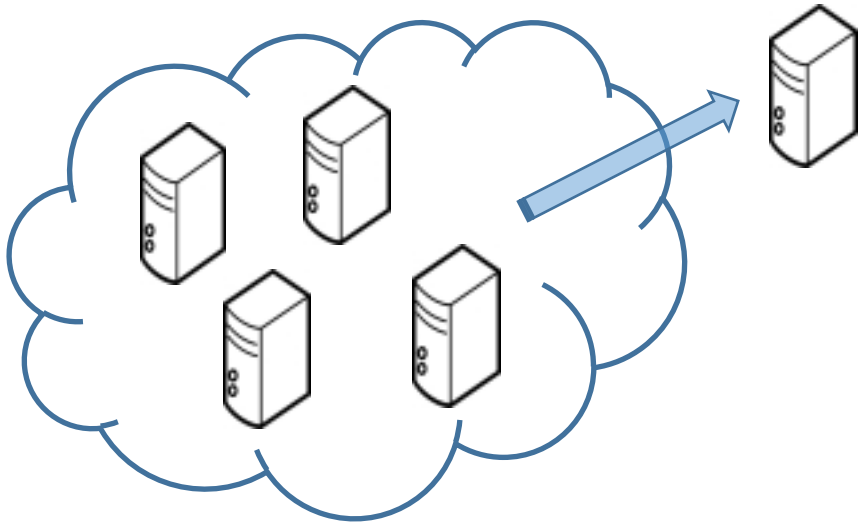
Commission



Constraints:

- No data losses
- Maintain fault tolerance
- Balance data

Decommission



Problems:

- Long data transfers

What is the duration of storage rescaling on a given platform?

- Previous works: lower bounds
  - Useful but unrealistic
  - Many simplifications

- Need a tool to measure it on real hardware

How fast can one scale down a distributed file system?, *N. Cheriere, G. Antoniu,* Bigdata 2017
A Lower Bound for the Commission Times in Replication-Based Distributed Storage Systems. *N. Cheriere, M. Dorier, G. Antoniu.* [Research Report – Submitted to JPDC] 2018

A benchmark: Pufferbench

Goals:

- Measure the duration of rescaling on a platform
- Serve as a quick prototyping testbed for rescaling mechanisms

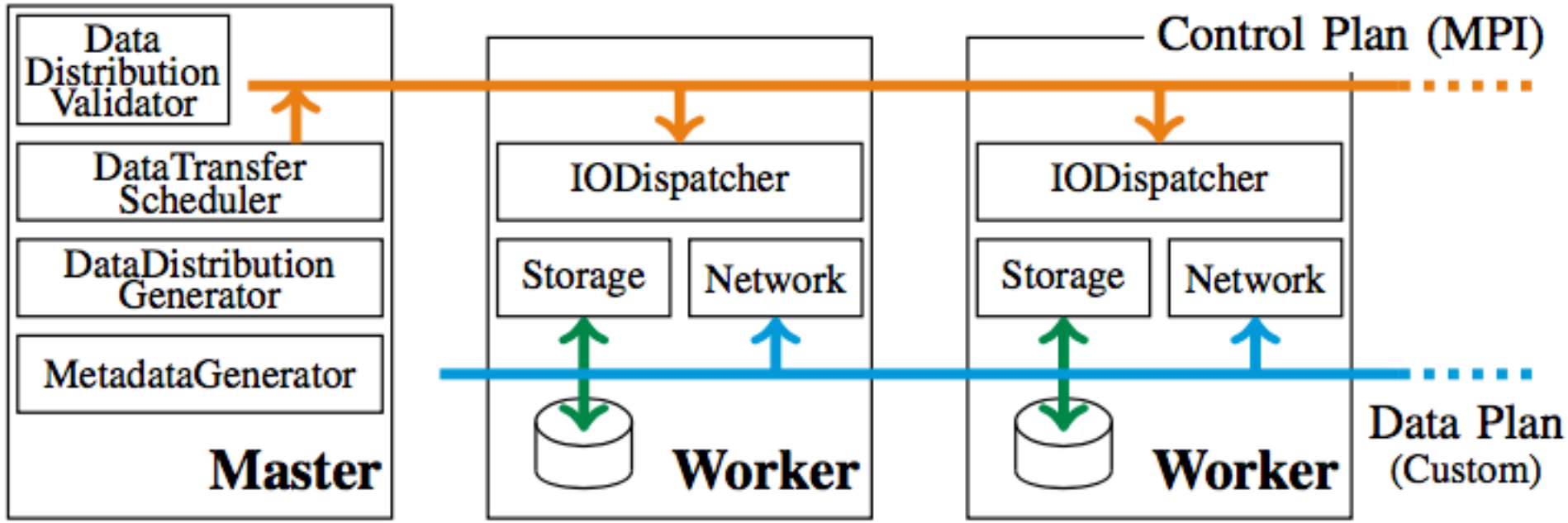
How:

- Do all I/Os that are needed by a rescaling

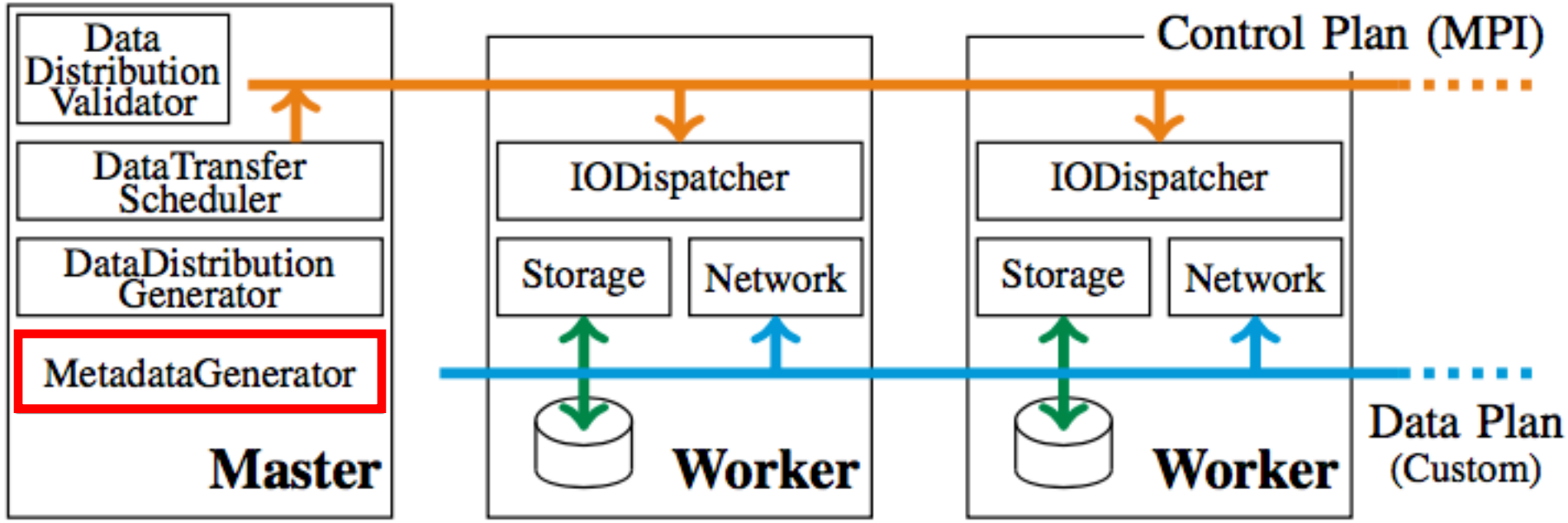
Main steps

1. Migration Planning
2. Data Generation
3. Execution
4. Statistics Aggregation

Software Architecture

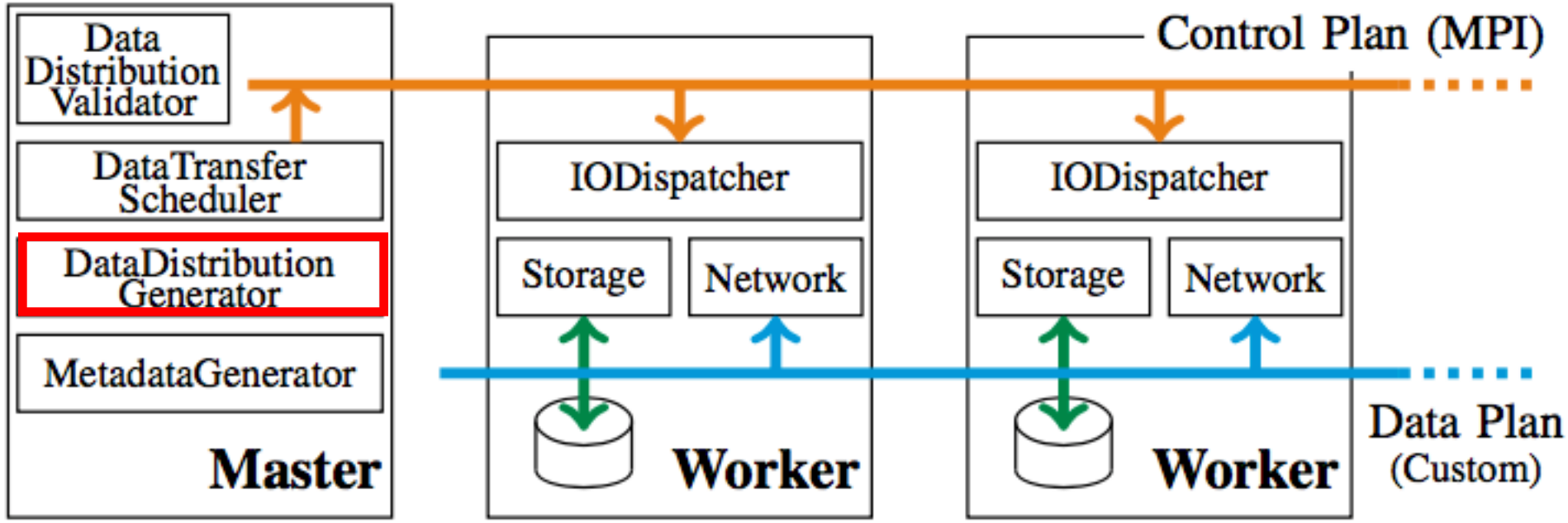


Software Architecture



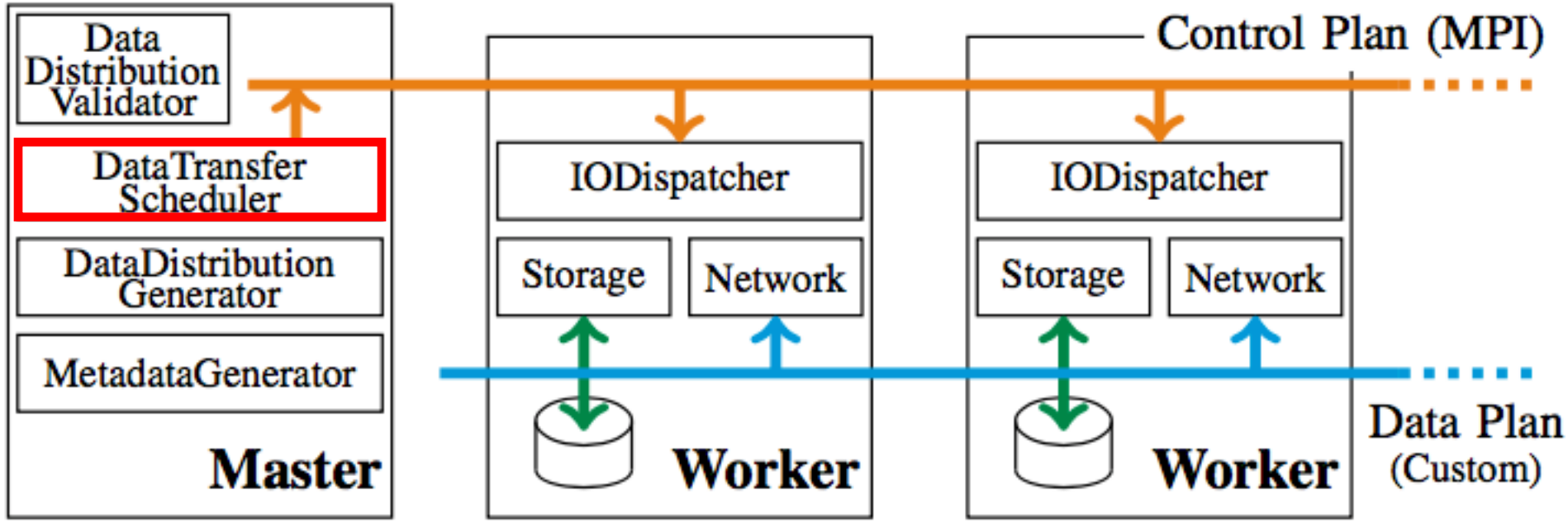
MetadataGenerator: Generate information about files on the storage (number,size)

Software Architecture



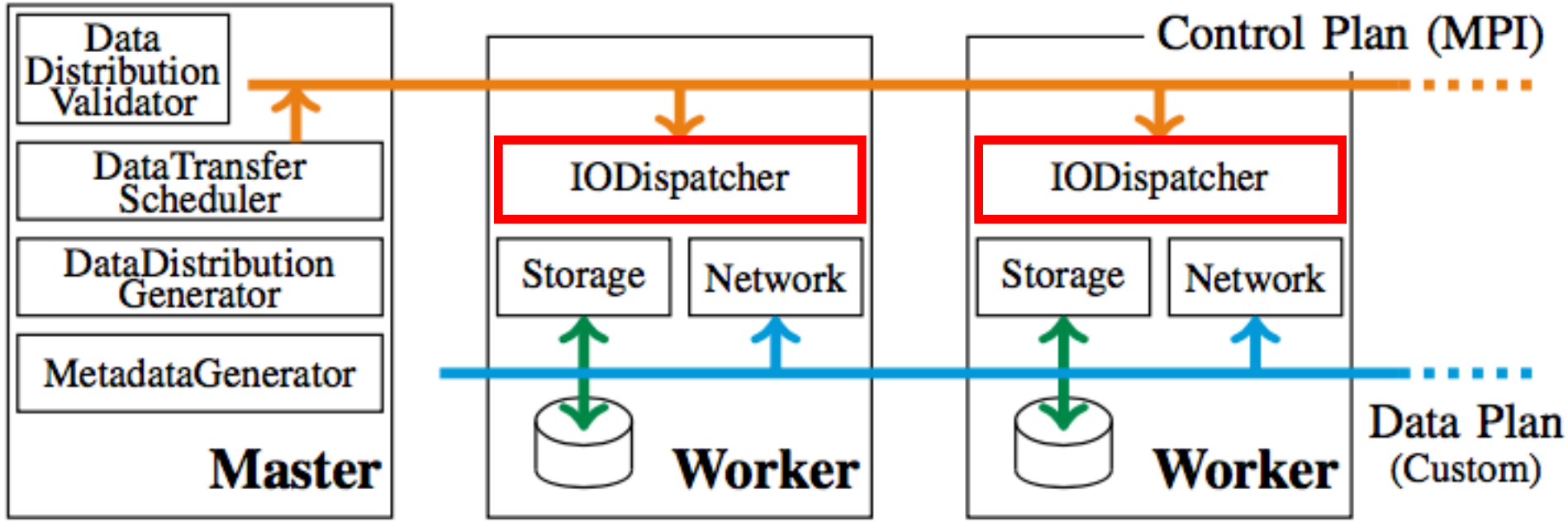
DataDistributionGenerator: Assign files to storage nodes

Software Architecture



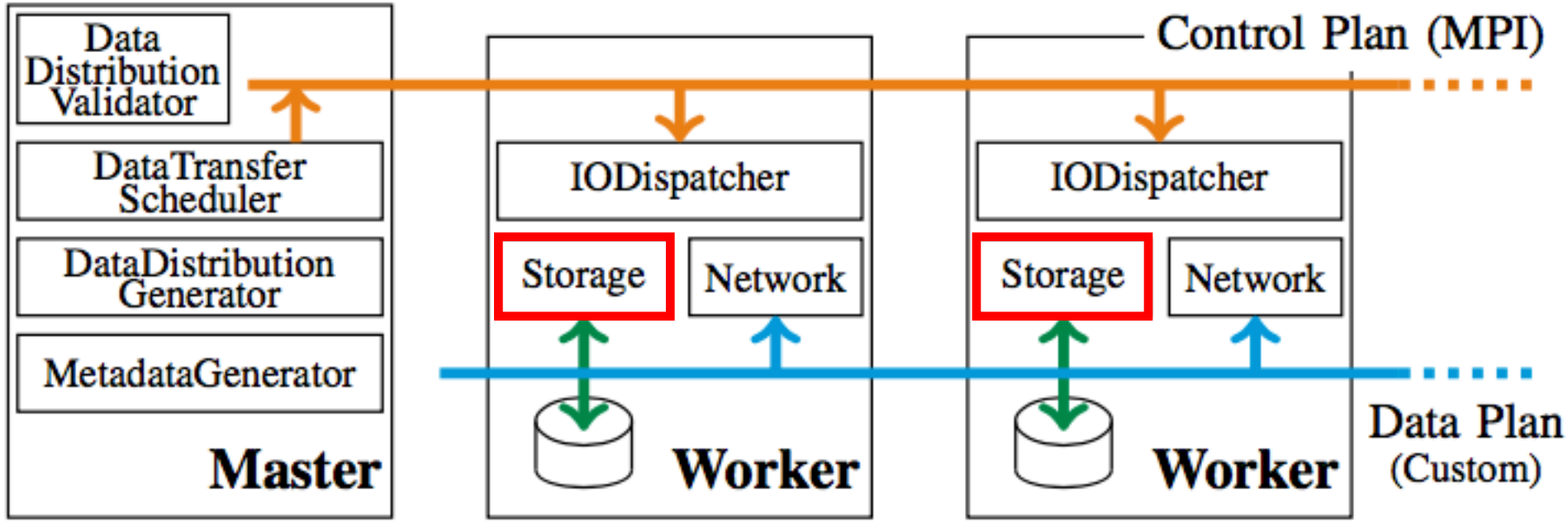
DataTransferScheduler: Compute data transfers needed for rescaling

Software Architecture



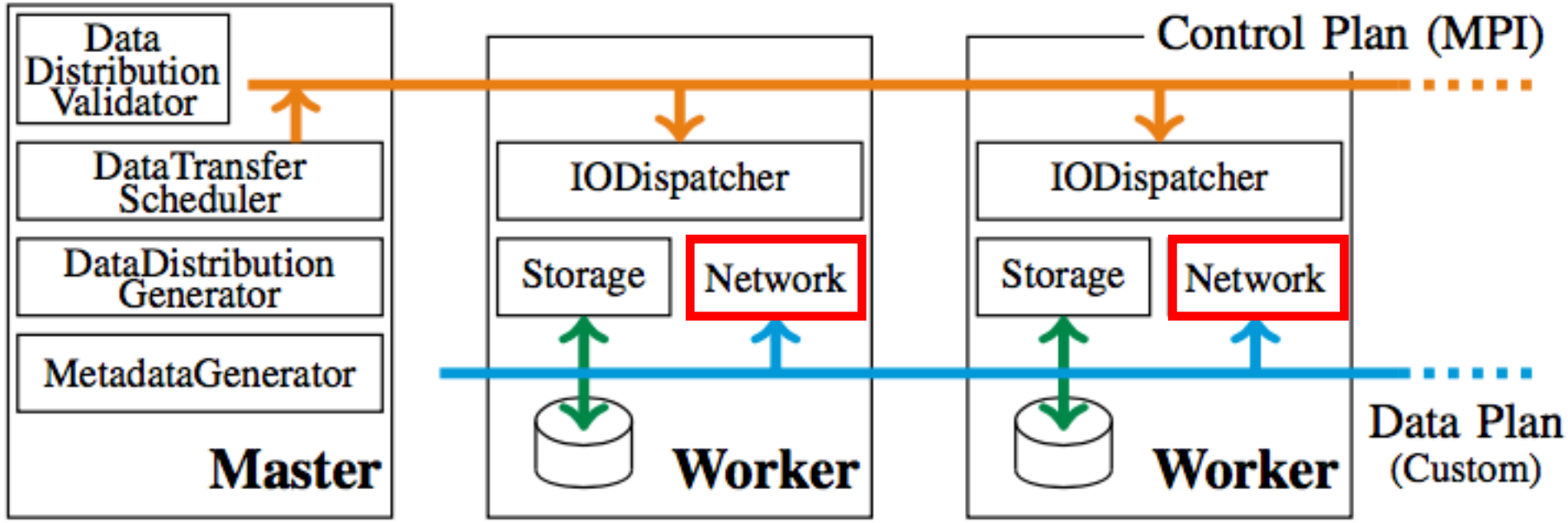
IODispatcher: Assign transfer instructions to storage and network

Software Architecture



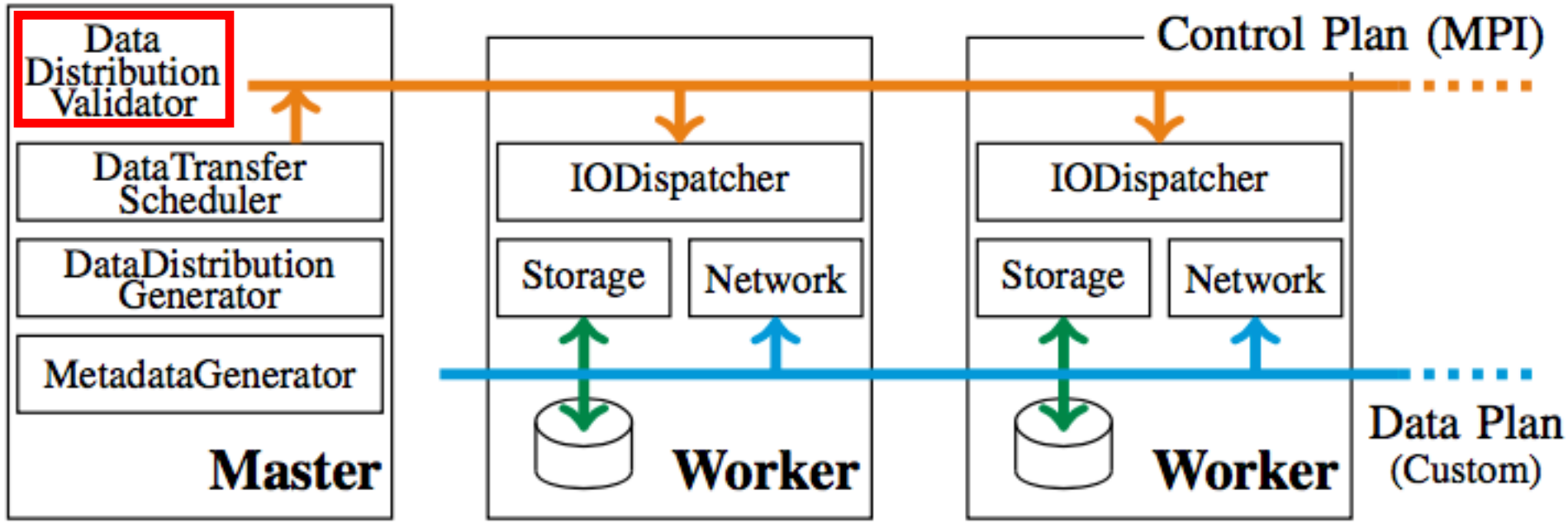
Storage: Interface with the storage devices

Software Architecture



Network: Exchange data between nodes

Software Architecture



DataDistributionValidator: Compute statistics about data placement (load, replication)

Validation

Hardware

- Up to 40 nodes
  - 16 cores, 2.4 GHz
  - 128 GB RAM
  - 558 GB disk
  - 10 Gbps ethernet

Comparison to lower bounds

Matching hypotheses:

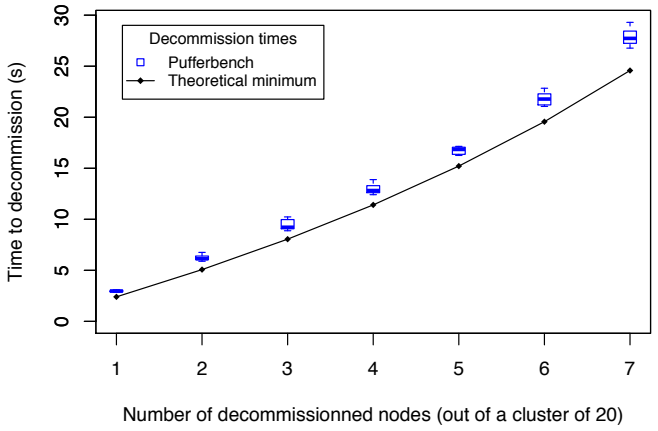
- Load balancing (50 GB per node)
- Uniform data distribution
- Data replication

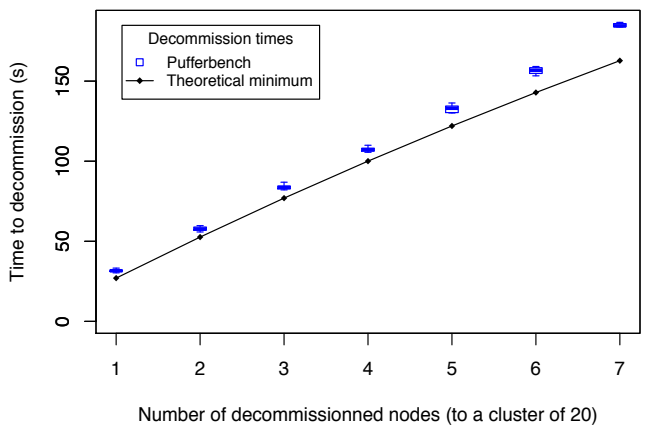
Differences:

- Hardware is not identical
- Storage has latency
- Network has latency and interferences

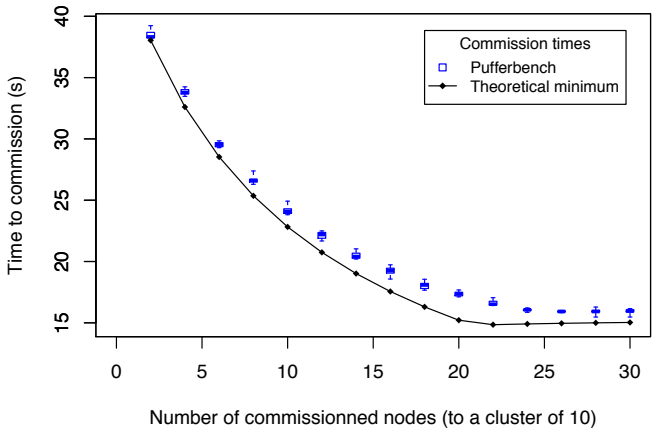
Pufferbench is close to lower bounds!

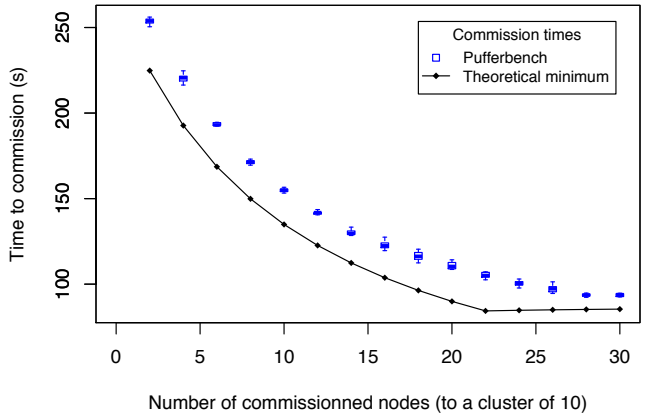
Decommission





Commission





In memory storage

On drive storage

Within 16% of lower bounds

Lower bounds are realistic

Use case: HDFS

Question: How fast can the rescaling in HDFS be?

No modifications of HDFS

With Pufferbench:

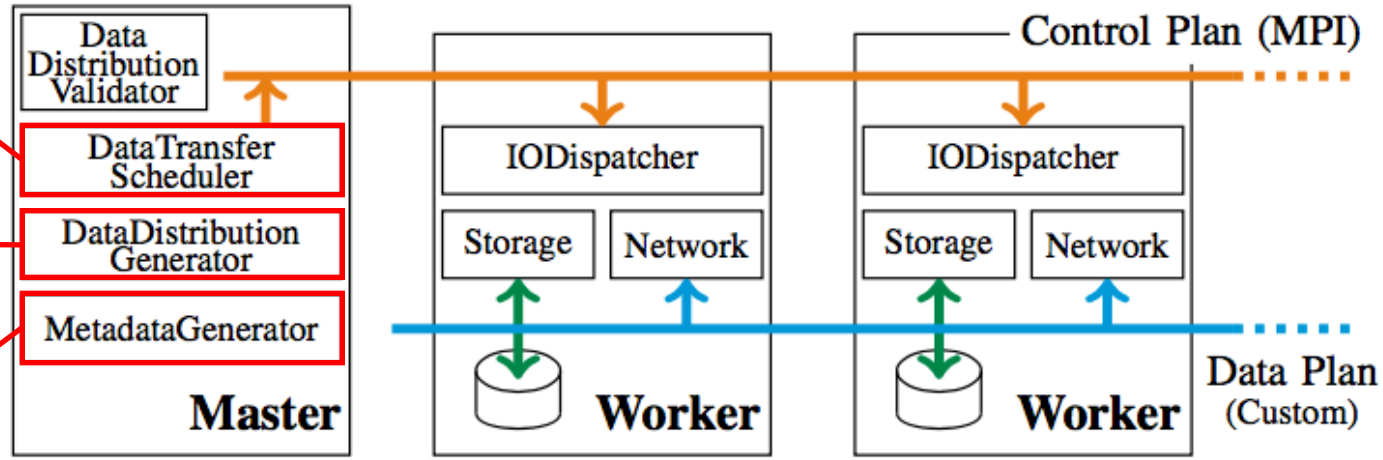
- Reproduce initial conditions
- Aim for same final data placement

Pufferbench matching HDFS's rescaling

- Load balanced
- Mostly random

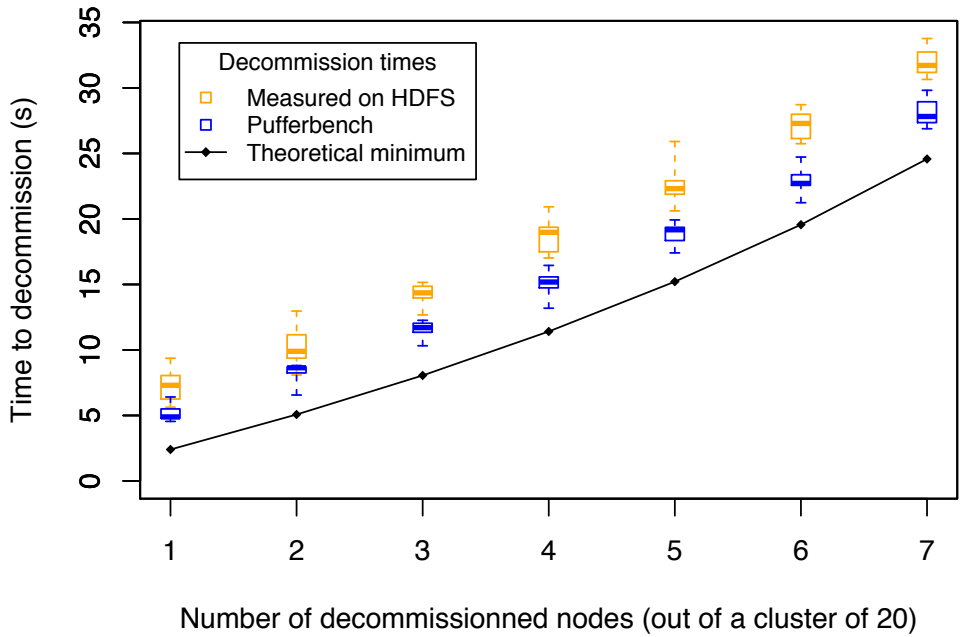
- Random placement
- Replicated 3 times

- Chunks of 128 MiB

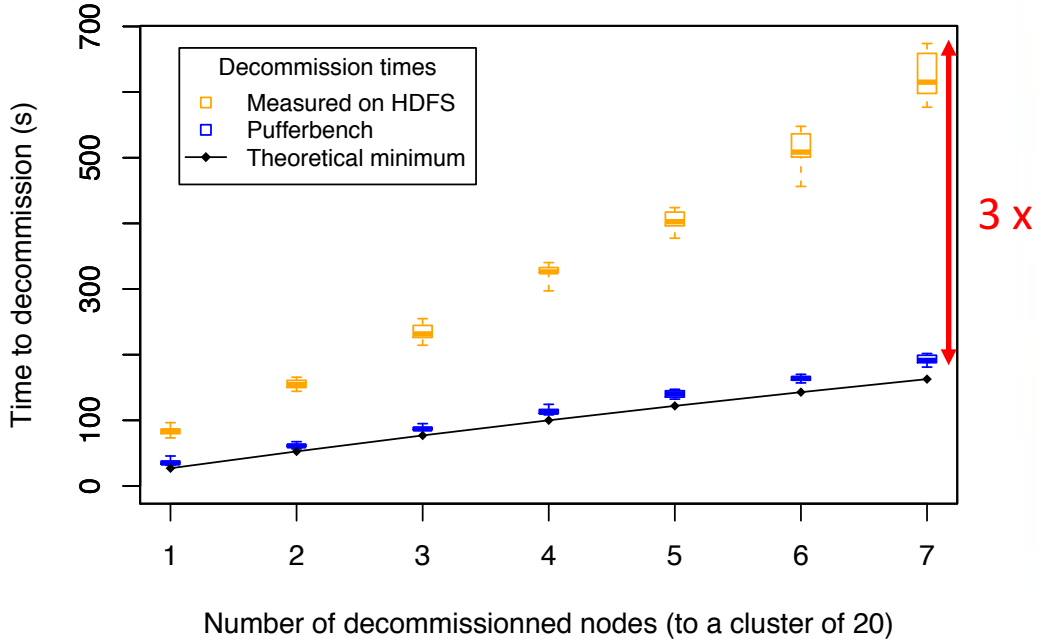


HDFS needs better disk I/Os

Commission



In memory storage

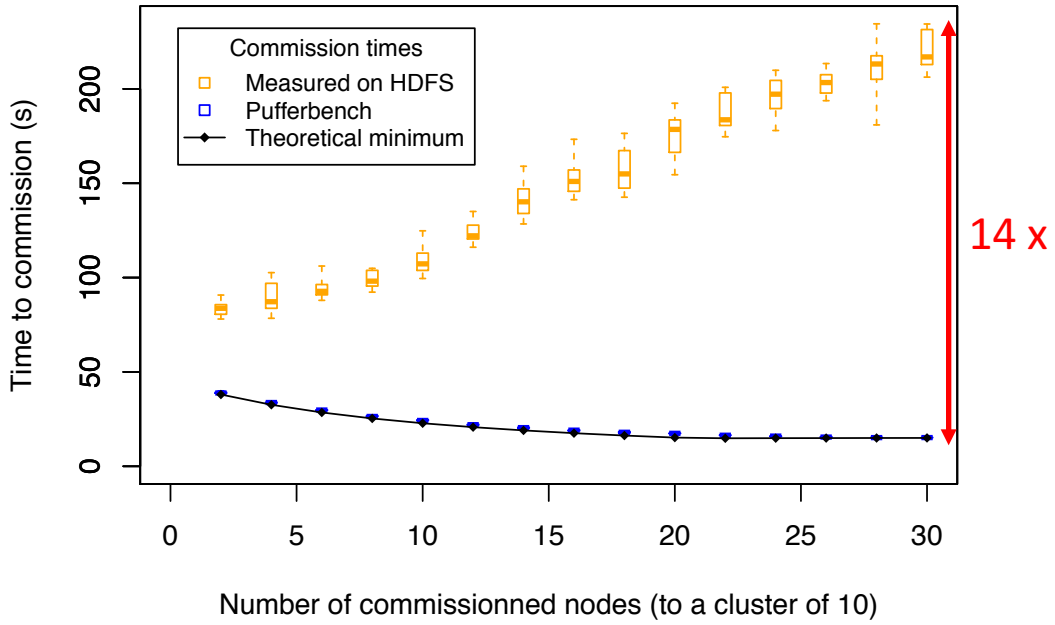


On drive storage

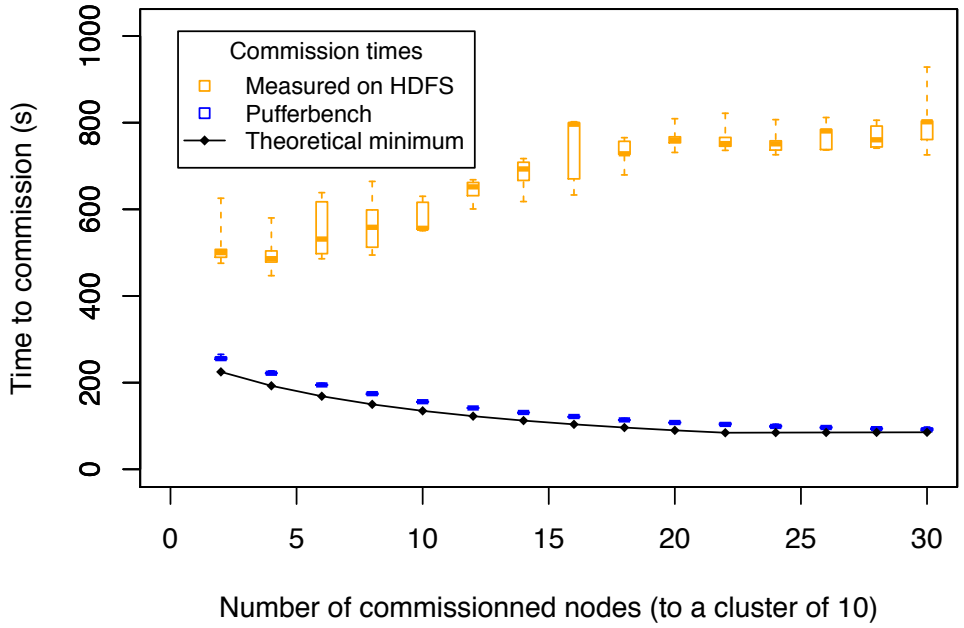
Improvement possible on disk access patterns

HDFS is far from optimal performances!

Commission



In memory storage



On drive storage

Improvement possible on algorithms, disk access patterns, pipelining

Setup duration

Setup overhead for the commission in memory:

- HDFS: 26 h
- Pufferbench: 53 min

Good for prototyping:

- Fast evaluation
- Light setup

To conclude

Pufferbench:

- Evaluate the viability of storage malleability on platforms
- Quickly prototype and evaluate rescaling mechanisms

Available at https://gitlab.inria.fr/Puffertools/Pufferbench

Can be installed with Spack

To conclude

Pufferbench:

- Evaluate the viability of storage malleability on platforms
- Quickly prototype and evaluate rescaling mechanisms

Available at https://gitlab.inria.fr/Puffertools/Pufferbench

Can be installed with Spack

Thank you! Questions?

nathanael.cheriere@irisa.fr