MapReduce for Big Data with a DHT

Master’s Thesis in Software Systems

Oliver Zihler

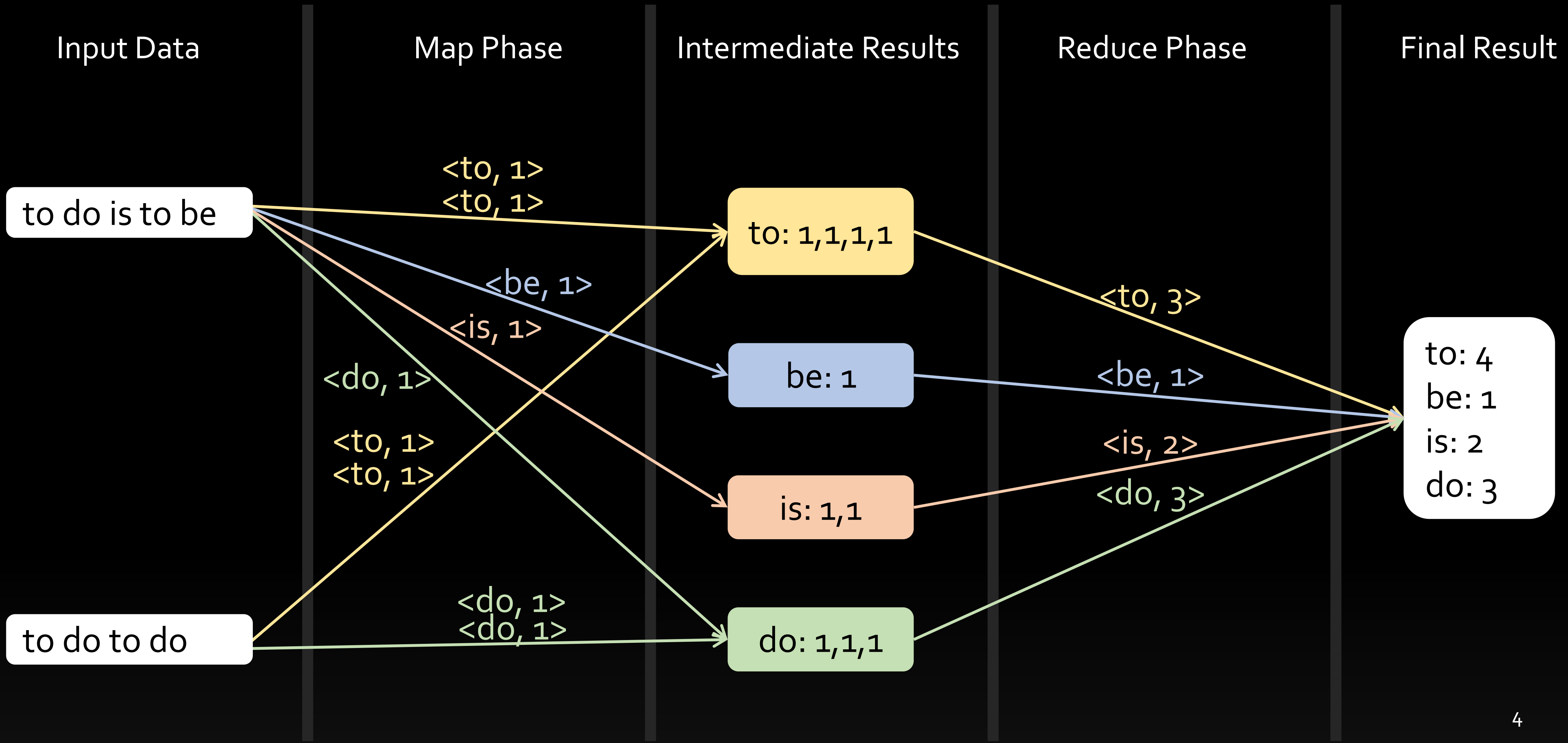
Overview

- What?
- Why?
- Challenges
- System Design & Implementation
- Remaining work
- Demo (?)
- Questions & Inputs?

What?

- MapReduce on TomP2P
- Evaluation using different datasets
- Comparison to Apache Hadoop (and possibly CloudFIT) regarding
  - Execution speed
  - Reliability

MapReduce Basic Wordcount Example



Why?

- **Exploring** (new) **possibilities** and **testing** the **feasibility** of implementing MapReduce on TomP2P with a distributed hash table as main storage facility
- **Exploring** ways to increase **reliability** compared to e.g. Hadoop
- Personal Goal: working and dealing with challenges in distributed systems, deepening understanding of both MapReduce and P2P

Challenges

- **Basic Challenges**
  - How to implement a fault-tolerant MR system on a P2P overlay network?

- **Communication Challenges**
  - How should nodes communicate with each other to finish execution faster?

- **Data Storage Challenges**
  - How to avoid data corruption in case a node fails?

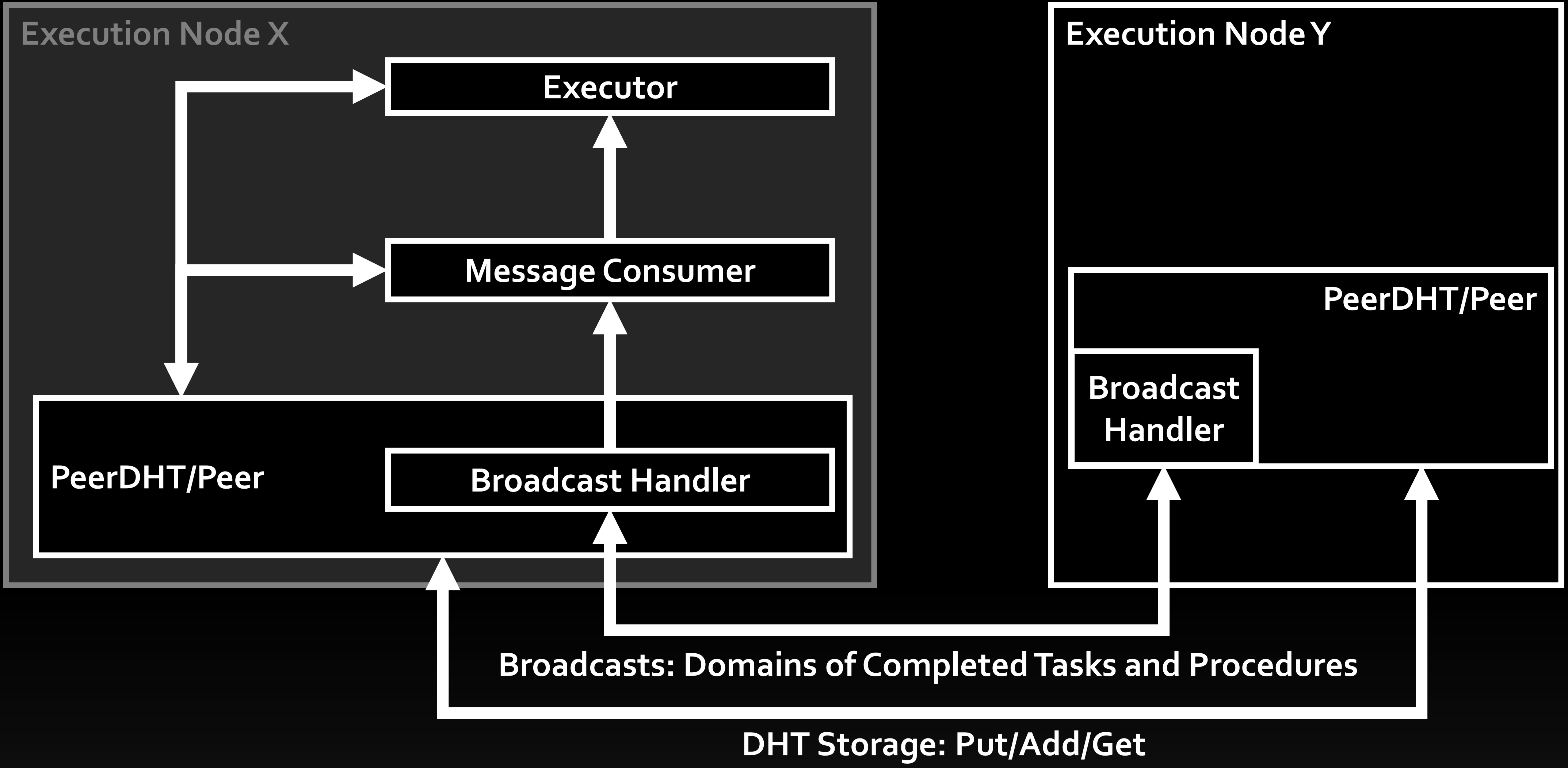
Addressing Challenges

- **Basic Challenges**
  - Assume every node may fail: every node executes everything
  - Results from other nodes are a nice-to-have but not a must-have!
- **Communication Challenges**
  - Only inform other nodes about completed executions
  - Do not inform nodes about executions in progress or failed ones
  - Use broadcasts for loose coupling between nodes
- **Data Storage Challenges**
  - Distinguish different result sets by storing each node’s results into an own part of the DHT (domain)
  - Only send locations of data via broadcasts, no actual data

System Design Ideas

- Abstract Map and Reduce Functions to «**Procedures**» for simplicity
- A MapReduce **job** is a set of **procedures** and **input data** for each procedure
- Divide input data for every procedure into **tasks** and **execute all tasks** on every node
- Store data from each **execution node**, **procedure**, and **task** in an own domain in the DHT (using TomP2P's domain key feature)
- **Broadcast only domains of completed tasks and procedures**, not the actual data, to other nodes
  - if a domain is not received, execution simply takes longer
  - If a node crashes **before** sending the broadcast, nothing happens
  - If a node crashes **after** sending the broadcast, its data may still be used by other nodes as it is safely stored within the DHT

Current Implementation



Remaining Work

- **Larger implementation endevours**
  - Correct scheduling and multithreading
  - Data cleanup for discarded and intermediate data
  - Possible (unknown) adaptions will become necessary during evaluation

- **Evaluation**
  - Testing behaviour with multiple jobs, task executions, procedure executions
  - Correct interaction with multiple nodes/computers
  - Datasets: evaluation of execution time and possible problems with increasing dataset size
  - Comparison to other MapReduce implementations (Hadoop, CloudFIT?)

Demo (?)

- Not sure yet…

Questions & Inputs?