Data I/O provision for Spark applications in a Mesos cluster

Nam Hoai Do, Tien Van Do, Xuan Thi Tran
Analysis, Design and Development of ICT systems (AddICT) Laboratory
Budapest University of Technology and Economics
Magyar tudósok körút 2, Budapest, Hungary

Lóránt Farkas, Csaba Rotter
Nokia, Bell Labs, Research
Köztelek utca 6, Budapest, Hungary

Agenda

- Motivation
- Technical Background
- Problem Investigation and Proposed Solution
- A proof of concept
- Conclusions
- Q & A

NOKIA

icdn 2016

More Users
One Computer

One user
One Computer

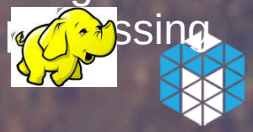
More Computers
One User or App

More Datacenters
More Users

Distributed Computing

Specialized HW
for special
pos

Commodity HW
BigData
ssing



Multitenant cloud
environment

Computer Usage History

Motivation

white box deployment

Shared with Operator and other vendors applications

SLA

Data Rate Guarantees

Multitenancy

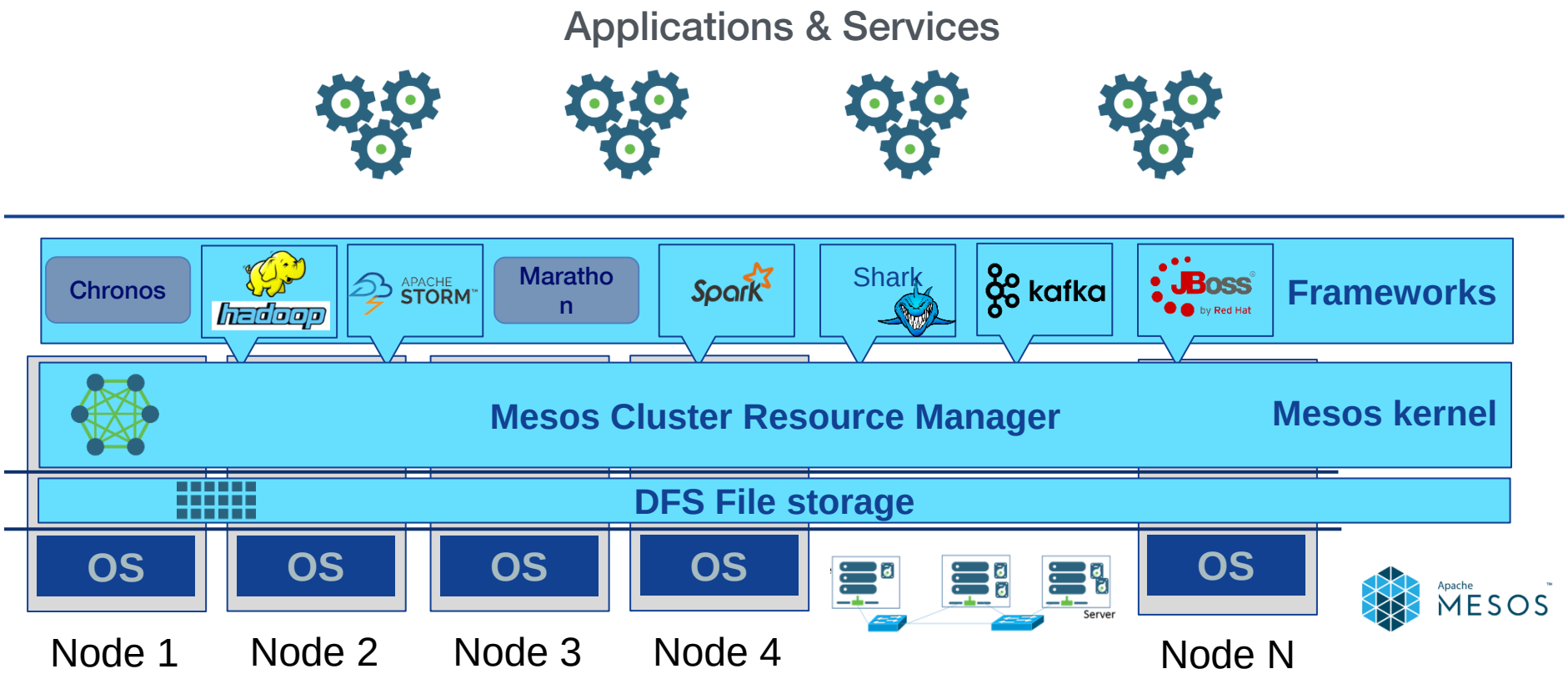
Current Limitations

Apache MESOS

Enforcing Disk and Network I/O guarantees.

- I/O contentions happen in production environments
- A need of data rate guarantee at Service Level Agreement
- Mesos supports CPUs, Memory, disk quota, outgoing traffic, port range
- Lack of I/O capacity

Technical background – Mesos



Main users:

Technical background – Mesos

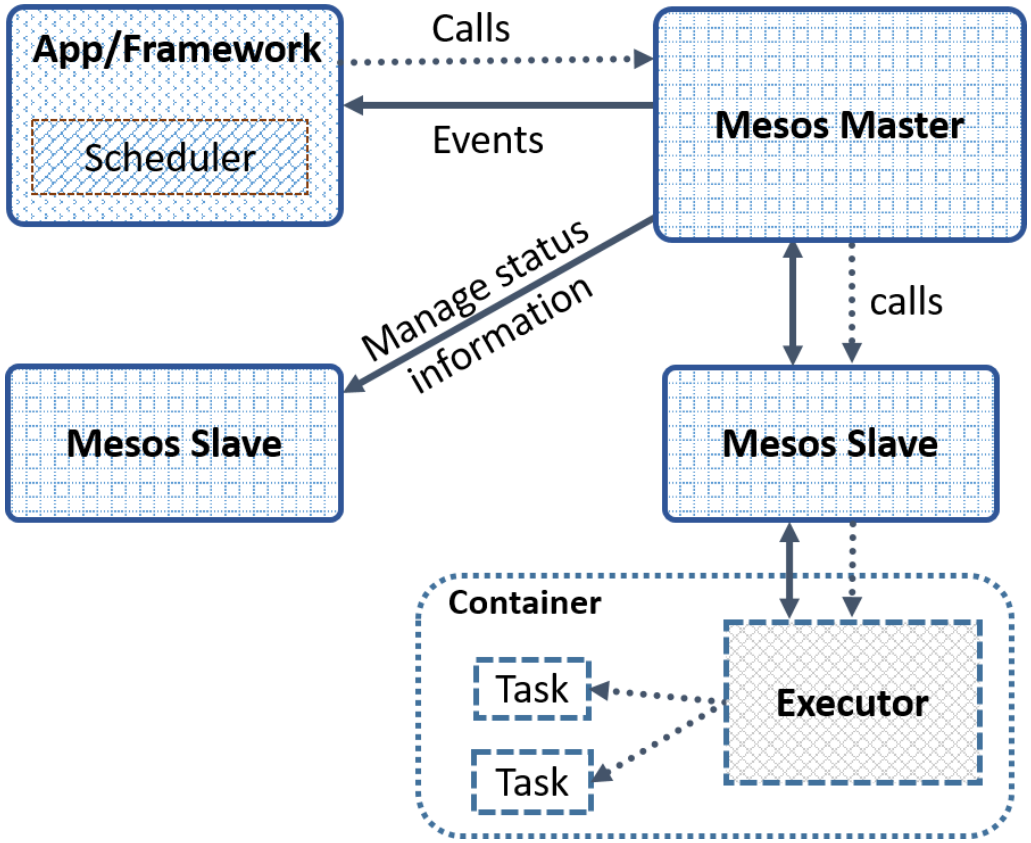
Master/slave architecture

- Mesos master manages and offers resources to frameworks

- Resource allocation: decisions made by frameworks

- New resources: simply defined in Mesos slaves

- Containers run on slave nodes to execute application tasks



Technical background – Spark

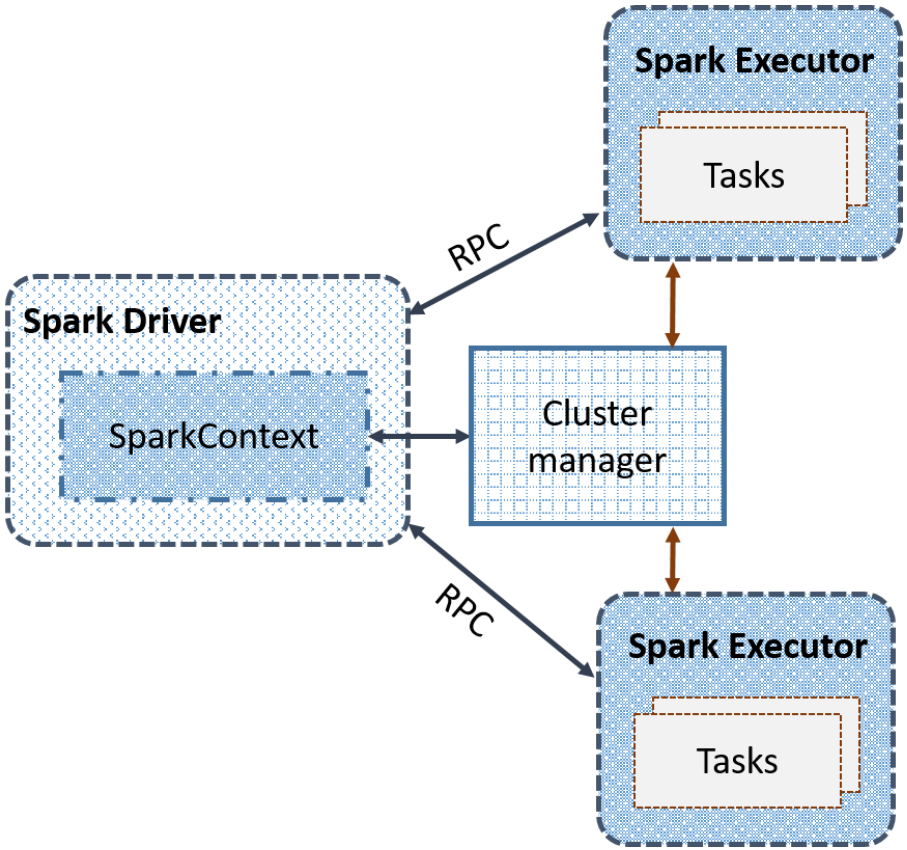
Apache Spark™ is a fast and general engine for large-scale data processing.

100x faster in memory
10x faster in disk
Easy to use: Java, Scala, Python, R
Runs everywhere: on Hadoop, on Mesos standalone or cloud

- Spark Driver: Register with manager, allocate resources, launch tasks

- Spark Executors: execute tasks on allocated resources

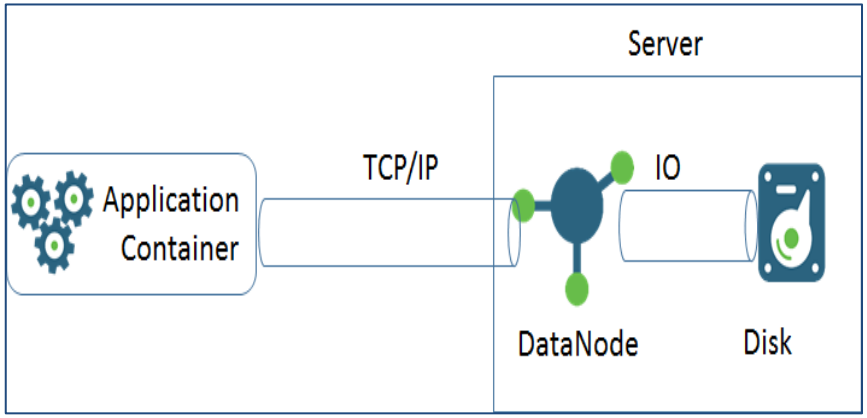
- Tasks process data stored in HDFS



Technical background – HDFS

HDFS

Files are splitted into blocks and stored in Datanodes.
Replication mechanism for failure tolerance.
The filesystem tree, the metadata are maintained by Namenode.



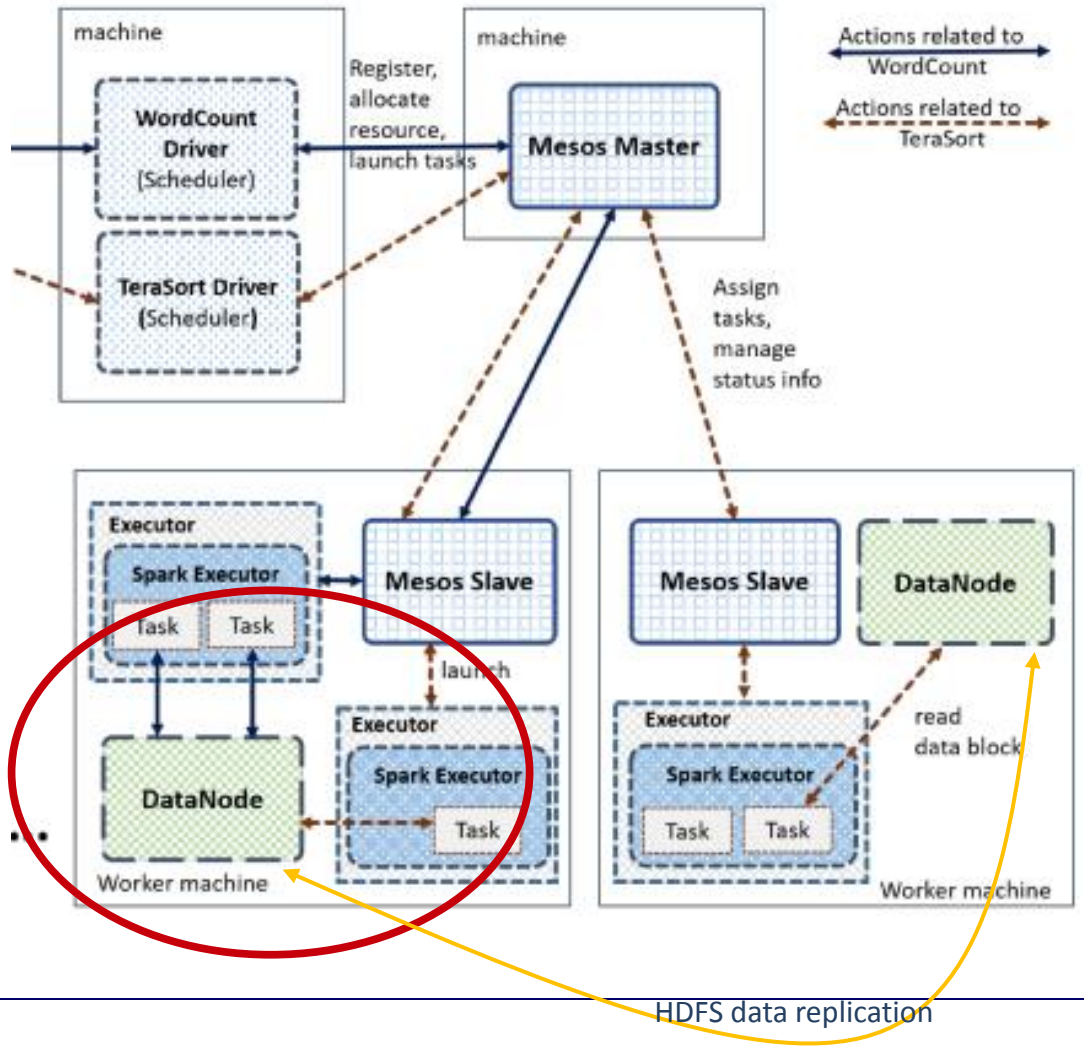
Data blocks are streamed from a chosen Datanote to a client through pipes:

TCP/IP pipe (through either a network or the loopback interface of a Datanode's machine) between an application and a DataNode.
a disk I/O pipe between DataNode and a certain disk.

Data I/O Problem Investigation

Applications may compete I/O with

- I/O of other running tasks
- I/O operations within HDFS



NOKIA

Data I/O Problem Investigation

Scenario 1: WordCount vs. TeraSort

Configurations:

- WordCount processes 3GB HDFS data with block size of 256MB
- TeraSort sorts 3 million records in HDFS with block size of 512MB
- Each app is configured to launch their tasks in a dedicated machine (which keeps their data blocks)
- All executors run up to three tasks in parallel.

Table 1. Performance of Applications without and with another application

Scenario	Read-Rate (MB/s)	Write-Rate (MB/s)	blkio_delay (ms)	Runtime (s)
WordCount alone	44.76	-	5.19	40.5
WordCount with TeraSort	22.18	-	13.08	**71.3**
TeraSort alone	36.67	69.38	8.51	99.8
TeraSort with WordCount	17.37	70.65	15.71	**133.2**

Data I/O Problem Investigation

Scenario 2: Spark applications vs. disk I/O stress activities

Disk I/O stress activities

- data is uploaded to HDFS or
- data reorganized in HDFS.

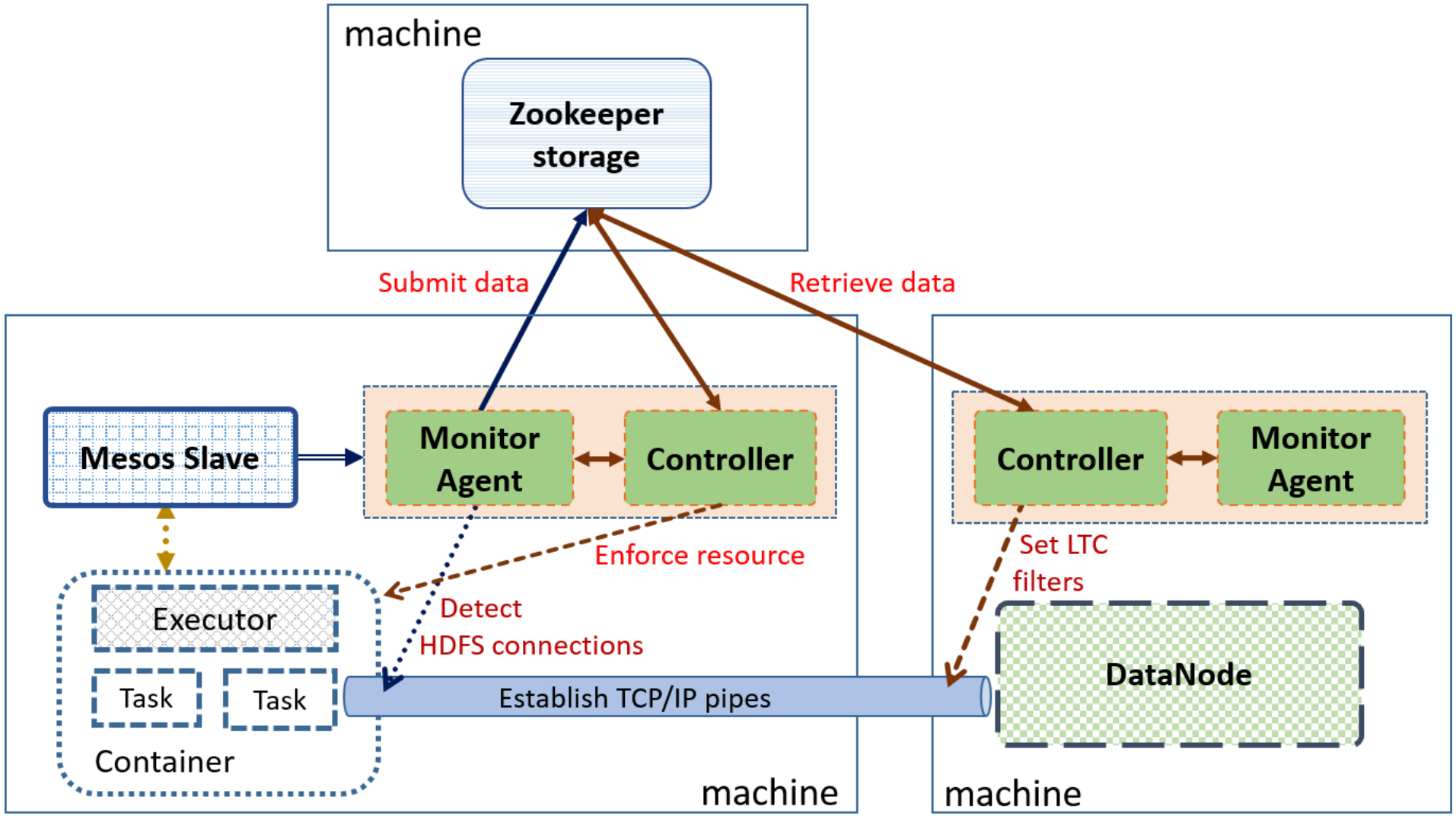
- emulated by:

- HDFS-reader & HDFS-writer: reading and writing HDFS data,
- Fio-reader and Fio-writer: process data in the disk drive shared with HDFS data blocks.

Table 2. Performance of WordCount without and with I/O stress activites

Scenario	Read-Rate (MB/s)	blkio_delay (ms)	Runtime (s)
WordCount alone	44.76	5.19	40.5
With Fio-reader	16.76	20.64	**90.8**
With Fio-writer	28.20	58.08	**143.6**
With HDFS-reader	34.46	8.25	**54.9**
With HDFS-writer	22.41	56.52	**162.1**

Proposed Solution: A control framework



Proposed Solution

Design of Proposed framework:

- a process (in each worker) with:

- **Monitor agent** to detect I/O resource usage and requirement information from Mesos slave and the established TCP/IP connections. Collected data is exchanged to adjacent or remote **Controllers.**

- **Controller** performs I/O enforcement on local containers or on a TCP/IP connection of a remote container based on retrieved data.

- a persistent storage (e.g. **Zookeeper**) for exchanging information among processes in different nodes.

Technical solutions:

* **CGroups block IO controller**:
- throttle I/O rate of read and write of a process group
- does not work on buffered write

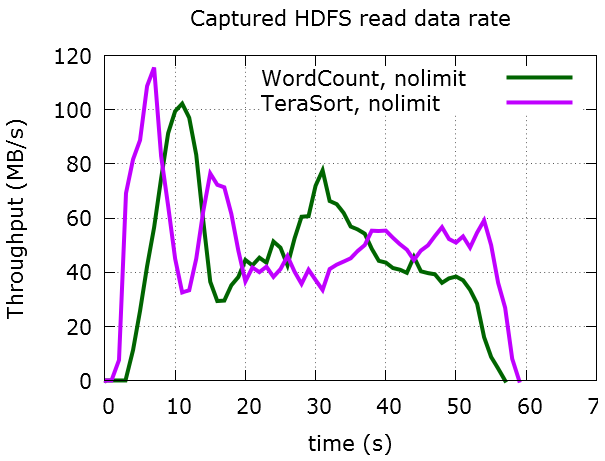
* **Linux Traffic Controller:**
- set filters to control data rate for a TCP/IP traffic.
- works only for *outgoing traffic*

=> rate of data write is set on containers, while rate of data read is set on DataNode side

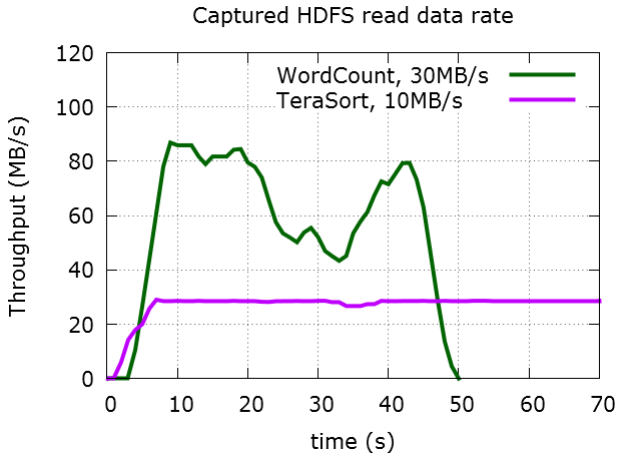
- controlling data rate on TCP/IP pipe affects on the disk I/O rate.

A proof-of-concept

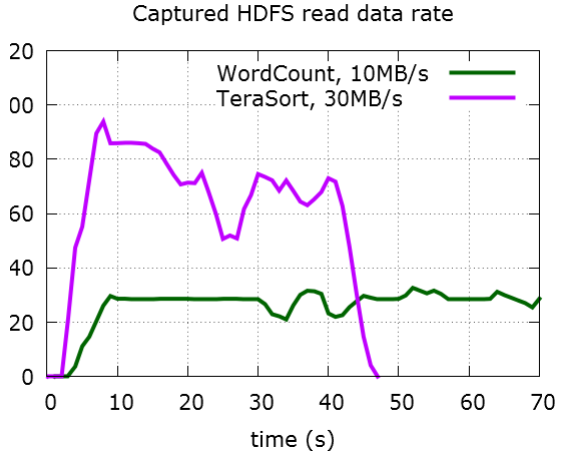
- **Experiment 1: Controling HDFS read data rates of Spark applications (WordCount and TeraSort)**
  - Both executors are in the same worker node
  - Each executor can run up 3 tasks in parallel
  - All tasks read data from the same DataNode



a) Without limiting rates



b) WordCount is preferred



c) TeraSort is preferred

NOKIA

A proof-of-concept

Experiment 2: Limiting the I/O rate of background applications

- WordCount and TeraSort executors can run up 3 tasks in parallel
- Executors of background applications can run only 1 task
- All tasks read data from the same DataNode/Worker machine
- I/O rate of background applications is controlled at 20 (MB/s)

WordCount		Avg. Read (MB/s)	blkio delay (ms)	Runtime (s)
Baseline		44.76	5.19	40.5
With uncontrolled I/O activities	+Fio-reader	16.76	20.64	**90.8**
	+Fio-writer	28.20	58.08	**143.6**
	+HDFS-reader	34.46	8.25	**54.9**
	+HDFS-writer	22.41	56.52	**162.1**
With controlled I/O activities	+Fio-reader/CGroup	40.51	6.41	**44.2**
	+Fio-writer / CGroup	28.01	59.71	**143.7**
	+HDFS-reader/LTC	40.39	6.37	**49.6**
	+HDFS-writer/LTC	35.85	7.24	**52.6**

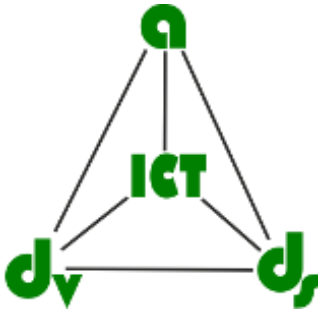
NOKIA

Conclusions

- We indicate I/O contention problems a shared cluster controlled by Mesos.
- Proposal of a control framework to
  - Take into account I/O capacity of worker nodes
    - Monitor I/O usage and detect TCP/IP connections related to data transfer.
  - Control I/O rate of applications based on an SLA
- The enforcement is performed at container level
- Data rate of applications can be properly controlled with our solution.

NOKIA

Thank you!



NOKIA