# Dynamic Task Scheduling and Data Collocation in Distributed Computing Systems

Daria Glushkova

Home University: Universitat Politècnica de Catalunya
Host University: Technische Universität Dresden
Supervisors: Prof. Alberto Abelló, Dr. Petar Jovanovic, Prof. Wolfgang Lehner
Period: December 2017 – December 2019

# Agenda

- Problem Context
- Thesis Objectives
- O1: MapReduce Performance Model for Hadoop 2.x
  - Background: Hadoop Architecture
  - Related Work: MapReduce Performance Models
  - Solution
  - Exerimental Setup & Evaluation
- O2: Performance Model for Spark
- Timeplan
- Publication Plan
- Other Activities

# Problem Context

- Performance Models

  Provide reasonable approximation of job response time at significantly lower cost than experimental evaluation of real setups.

  Essential for optimizing and can be helpful in critical decision making in workload management and resource capacity planning.

- Data Distribution

  The current distributed data storage systems do not consider the imposed workload during the data placement process in the cluster.

- Combining Data and Query Shipping

# Thesis objectives

**O1:** Build performance model to Hadoop 2.x.
**O2:** Create performance model to Spark.
**O3**: Find an appropriate effective way for workload aware data placement (data redistribution), which should increase the overall performance.
**O4**: Develop workload aware scheduling algorithms for scheduling jobs inside the cluster.

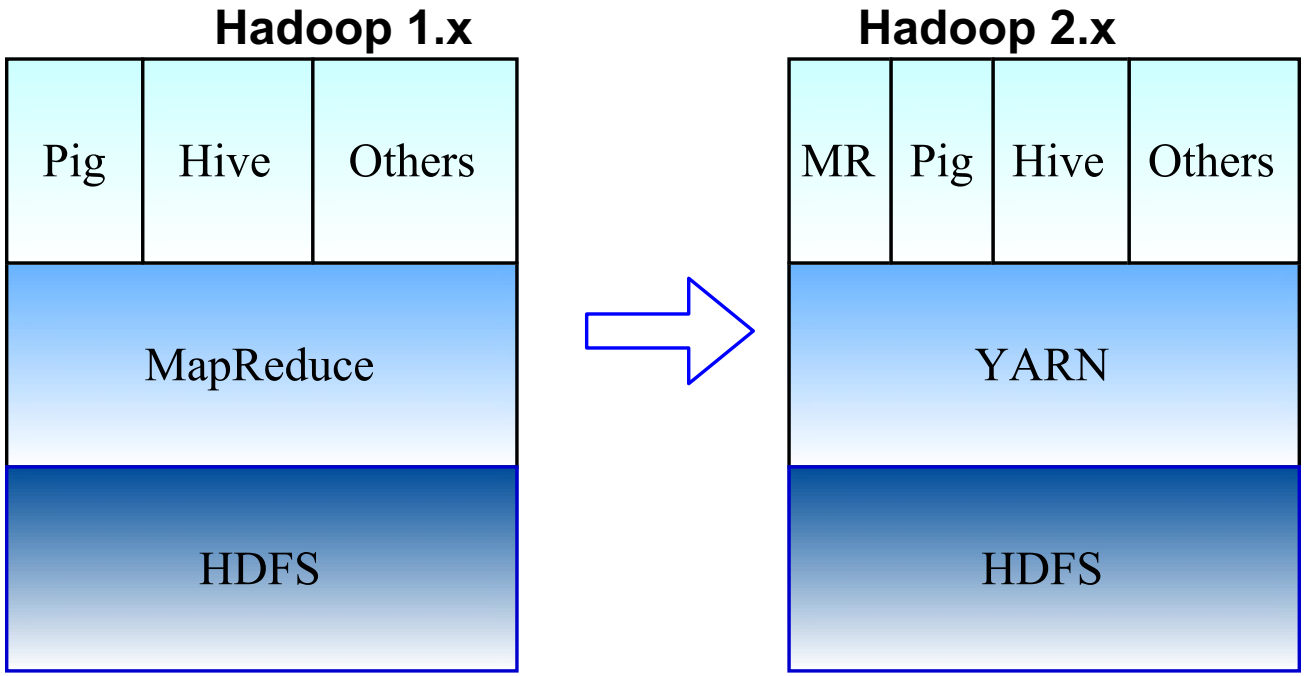
# O1: MapReduce Performance Model for Hadoop 2.x

The **objective** is to develop an efficient algorithm to estimate two measures of interest:

- The mean response time of individual tasks;
- The mean response time for a job.

Estimate the response time for the whole workload.

# Background: Hadoop Architecture




**Dynamic resource allocation!**

# Related Work: MapReduce Performance Models

Two groups of approaches:

- **Static**

  Do not take into account the queuing delay due to contention at shared resources.

  - Herodotou [1]; - ARIA [2]; - TETRIS [3]

- **Dynamic**

  Do not consider the synchronization delays due to precedence constraints among tasks that cooperate in the same job (map and reduce phases).

  Two techniques:

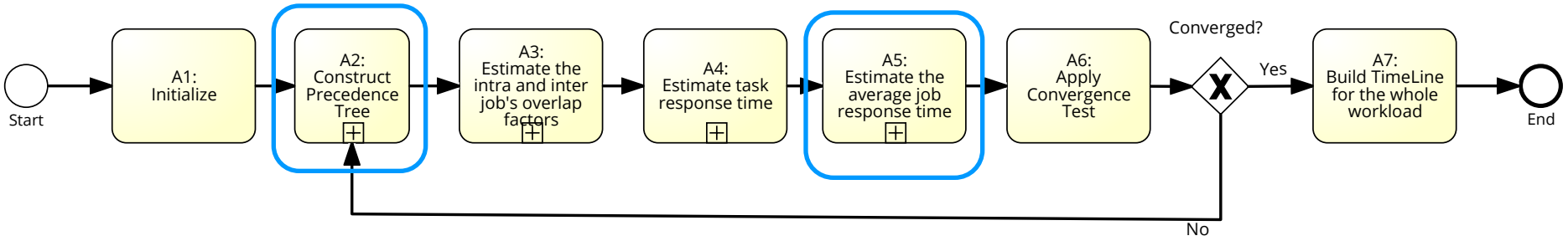
  - Mean Value Analysis (MVA)
  - Markov Chains

  - Vianna et al.[4]

Common limitation: Use a fixed amount of slots per map and reduce tasks within one node.

# Solution: General Schema

**Main Challenge**: How to adapt the existing model to Hadoop 2.x




Vianna et al.[4]

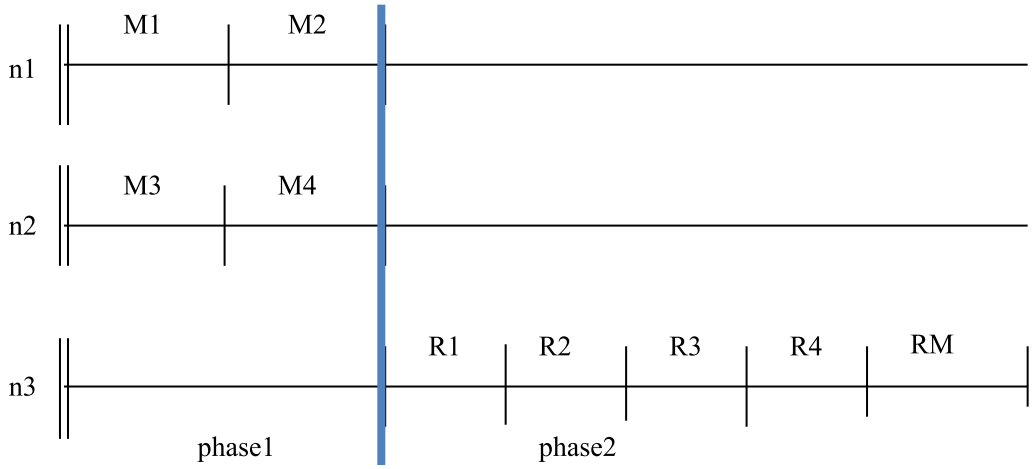
# A1: Initialization

a) Using sample techniques - taking the average of task response time from job profile.
b) Obtain from existing cost models that can capture unit costs of map and reduce tasks (e.g.,Herodotou's cost model [1])

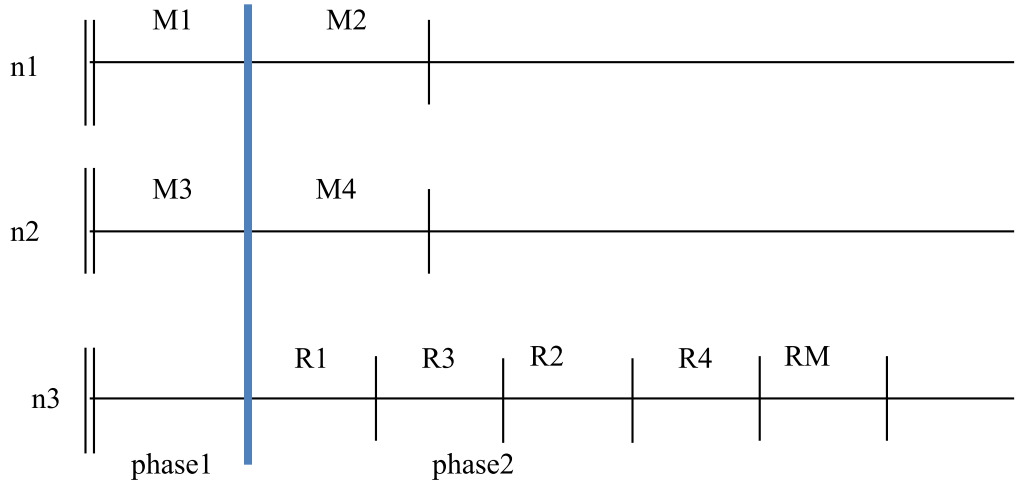
# A2: Building precedence tree

## Timeline construction

a) without slow start




b) with slow start




Resource request object

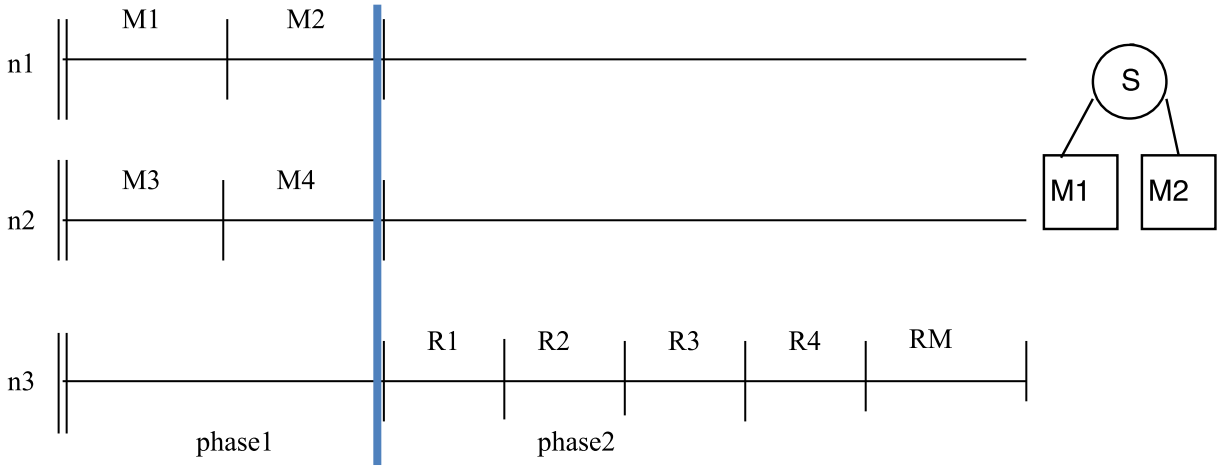
| Number of containers | Priority | Size | Locality constraints | Task type |
|---|---|---|---|---|
| 2 | 20 | x | n1 | map |
| 2 | 20 | x | n2 | map |
| 1 | 10 | x | * | reduce |

Map task is not divided into phases;
Reduce task:
(shuffle+sort) - shuffle subtask
(final sort+reduce function) – merge subtask

# A2: Building precedence tree

Captures the execution flow of the job using two types of primitive operators:

- P (parallel)
- S (sequential)

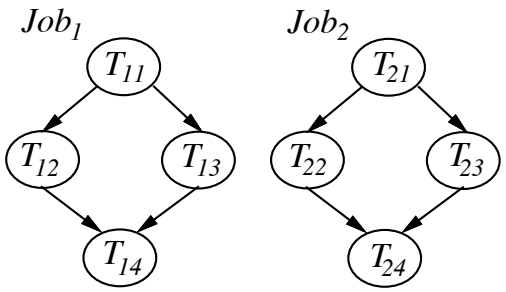


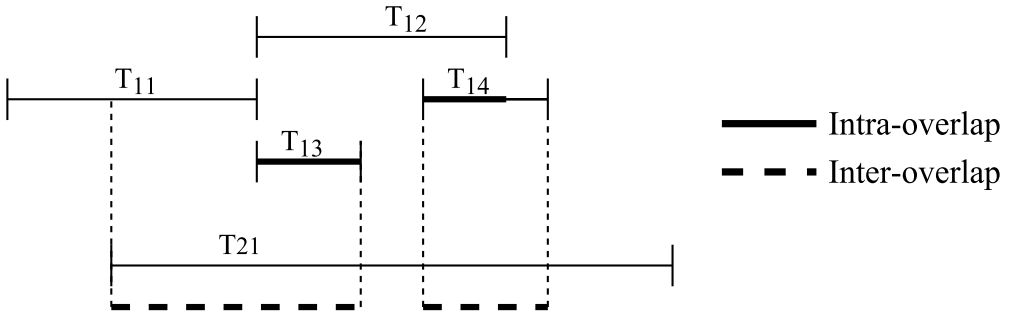
# A3: Estimation of the Intra- and Inter-job overlaps factors

For a system with multiple classes of tasks the queuing delay of task *i* class due to class *j* task is directly proportional to their overlaps.

Two types of overlap factors:

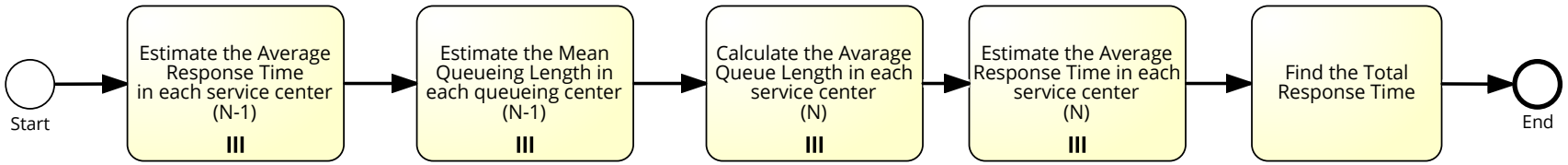
- Intra-job
- Inter-job



# First Approach (AMVA): A4: Estimation of task response time

To solve the queueing network models we apply Mean Value Analysis (MVA) [7]. MVA is based on the relation between the mean waiting time and the mean queue size of a system with one job less



# A5 Average Job Response Time Estimation

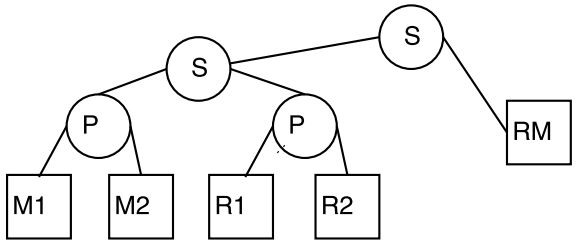
There are 3 alternative approaches to estimate the job response time:

- Tripathi-based

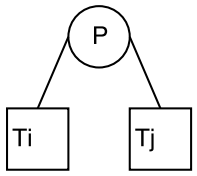
  Assumption: task durations have Erlang or Hyperexponential distributions.

  Knowing the distribution of tasks, we can determine the mean value for the root node, going from leafs to the top

- Task durations have normal distributions (Pearson's Criterion, 95% significance level).




- Fork/join-based




Execution time:

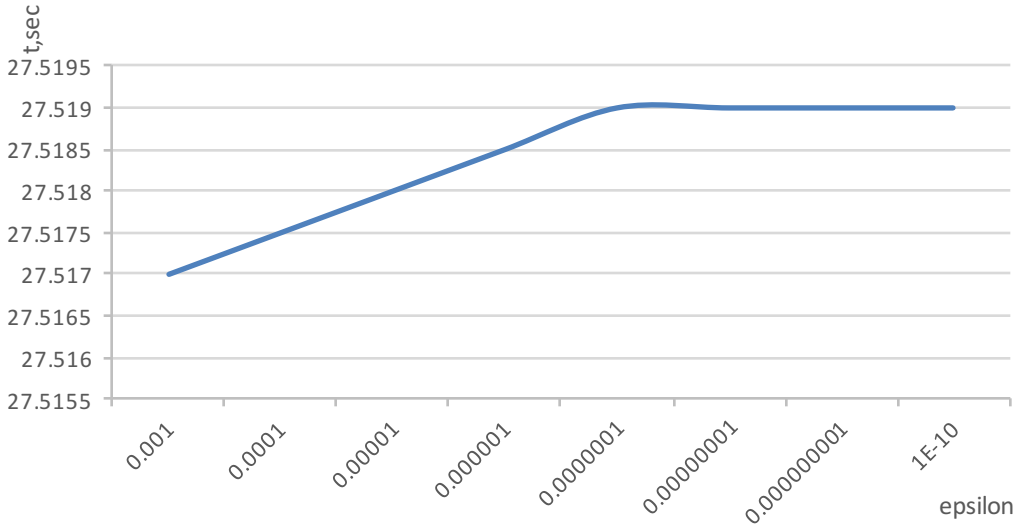
$$R_{ik} = H_k \cdot max(T_i, T_j),$$

where $H_k = \sum_{i=1}^{s} \frac{1}{i}$,

$s$ - is the number of child nodes

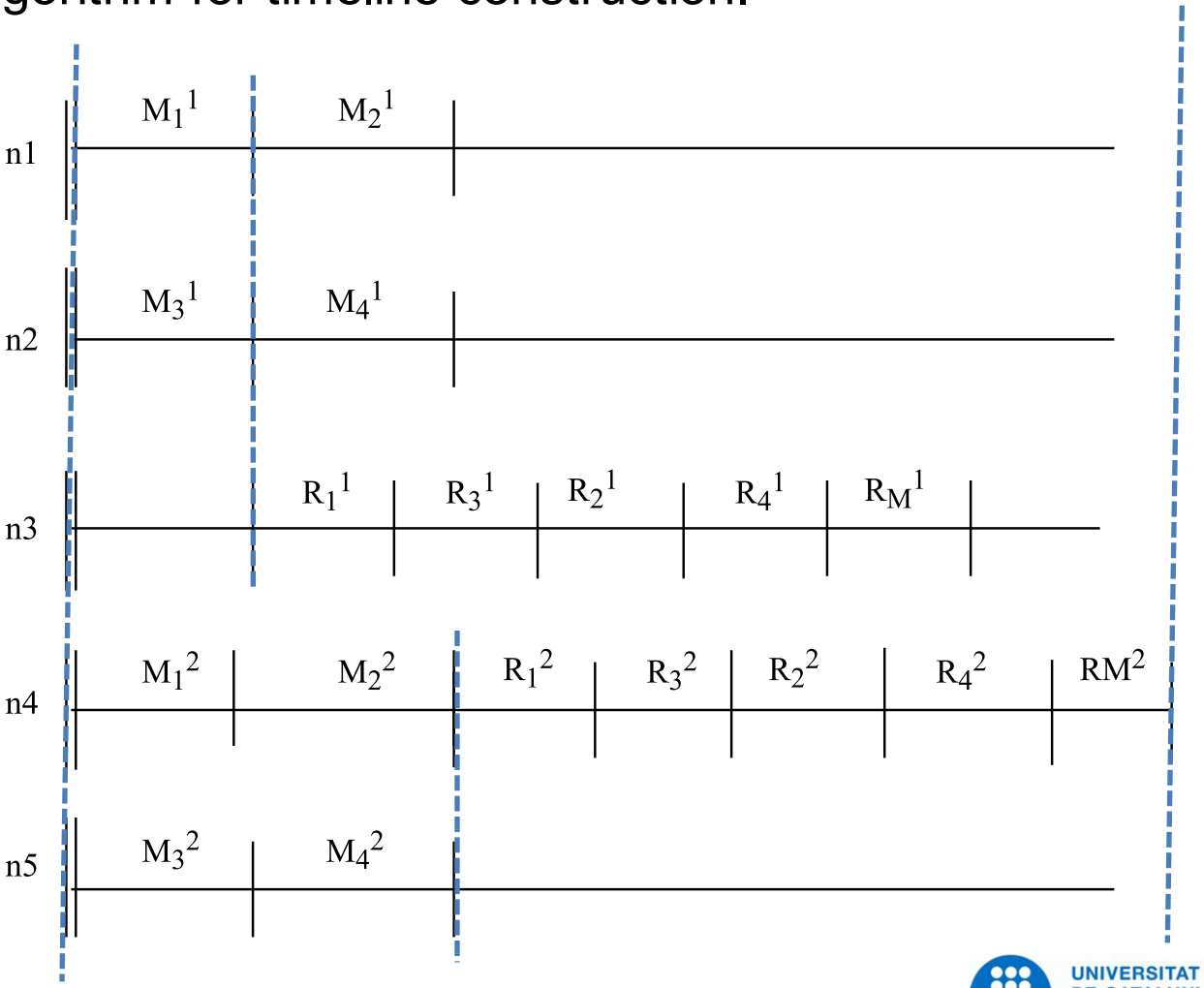
# A6: Applying convergence test

$$|R^{cur} - R^{prev}| < \epsilon$$



# A7: Building final timeline

Based on obtained estimations for task response time, apply the algorithm for timeline construction.




Workload: J1,J2
FIFO order

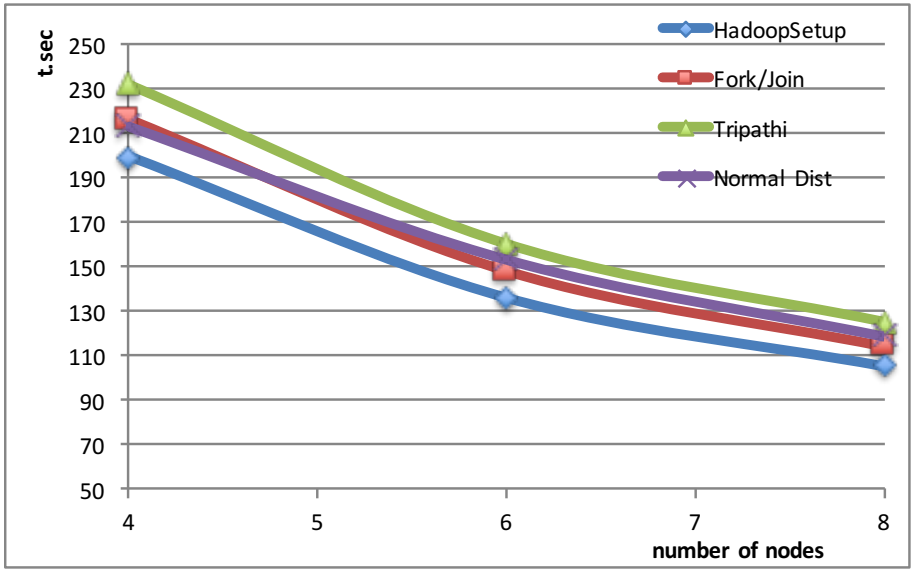
# Experimental setup

We performed a set of experiments analyzing the job response time in terms of the following parameters:

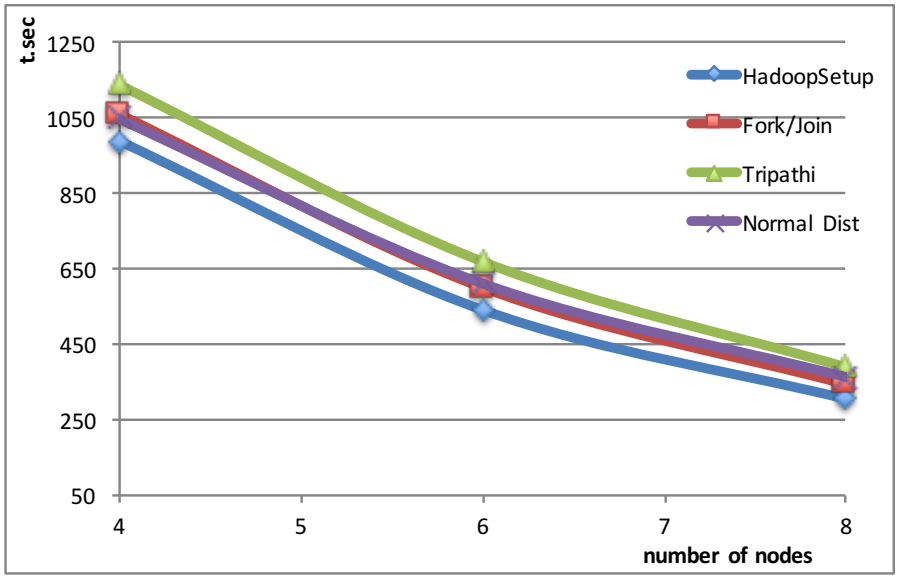
- number of nodes: 4,6,8;
- size of input data: 1GB, 5GB;
- number of jobs (wordcount, sort) that are executed simultaneously in the cluster: 1,2,3,4;
- Tripathi-based, Fork/Join-based algorithms.

# Evaluation

- Input: #5GB; Number of jobs #1




Input: #5GB; Number of jobs #4




Error for:
Fork/Joined based: 11%-14%
Normal Distr.: 12-15%
Tripathi based: 20-23%

# O2: Performance Model for Spark

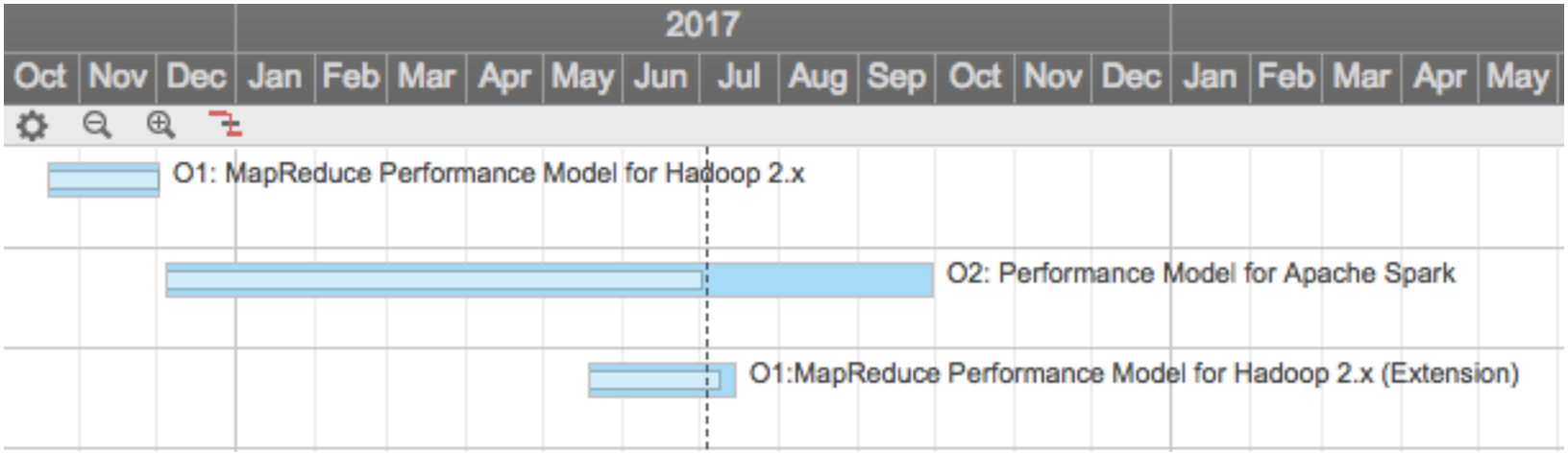
## Ongoing work

We need to adapt for Spark the precedence tree construction procedure taking into consideration 3 levels of Scheduling:

- Job Scheduling (FIFO, FAIR)
- Stage Scheduling (DAG Scheduler)
- Task Scheduling (Delay Scheduling)

# Timeplan

- **O1** completed
- **O2** ongoing



# Publication Plan

**Submitted:**

Submission of the paper “Performance Model for Hadoop 2.x" – DOLAP, 2017. Accepted.

**Scheduled**:
Submission of the paper about \Performance model for Hadoop 2.x" (Extended version) - Information Systems, 14th July, 2017.

**Planned**:
“Performance model for Apache Spark” DOLAP, November 18th

# Other Activities

**Teaching activities**

- Web Services Course: labs + supervision of projects

**Studying**

- Courses:
  - Self-service Business Intelligence
  - Foundations and recent trends on ontology engineering
- Spanish language courses

# Thank you for your time!

Questions & Discussion

# References (Hadoop)


[1] Herodotou, H. "Hadoop Performance Models", Technical Report, CS-2011-05 Computer Science Department Duke University, p. 19.

[2] A. Verma, L. Cherkasova, and R. H. Campbell. ARIA: automatic resource inference and allocation for mapreduce environments. In Proceedings of the 8th ACM international conference on Autonomic computing, pages 235-244. ACM, 2011

[3] Grandl, Robert, et al "Multi-resource packing for cluster schedulers", ACM SIGCOMM Computer Communication Review. Vol. 44. No. 4. ACM, 2014.

[4] E. Vianna, G. Comarela, T. Pontes, J. Almeida, V. Almeida, K. Wilkinson, H. Kuno, and U. Dayal. Analytical performance models for MapReduce workloads. International Journal of Parallel Programming, 41(4):495-525, 2013.

[5] M. Reiser and S. S. Lavenberg. Mean-value analysis of closed multichain queuing networks. Journal of the ACM (JACM), 27(2):313-322, 1980

[6] D.-R. Liang and S. K. Tripathi. On performance prediction of parallel computations with precedent constraints. IEEE Transactions on Parallel and Distributed Systems, 11(5):491-508, 2000.

# References (Spark)

[7] Wang K., Khan M. M. H. Performance prediction for apache spark platform, High Performance Computing and Communications (HPCC), 2015 IEEE 7th International Symposium on Cyberspace Safety and Security (CSS), 2015 IEEE

[8] Sidhanta S., Golab W., Mukhopadhyay S. OptEx: A Deadline-Aware Cost Optimization Model for Spark, Cluster, Cloud and Grid Computing (CCGrid), 2016 16th IEEE/ACM International Symposium on. – IEEE, 2016. – C. 193-202.

[9] Gounaris A. et al. Dynamic Configuration of Partitioning in Spark Applications, IEEE Transactions on Parallel and Distributed Systems. – 2017.

[10] Specification Design Specification of Spark Cost-Based Optimization (SparkSQL) https://issues.apache.org/jira/secure/attachment/12823839/Spark_CBO_Design_Spec.pdf