A Control Approach for Performance of Big Data Systems

Mihaly Berekmeri, Damian Serrano, Sara Bouchenak, Nicolas Marchand, Bogdan Robu

GIPSA - LIG - Grenoble University, France

The structure of the presentation

1. **Introduction**
   – Big Data MapReduce
   – State of the art
2. **Experimental setup**
   – Sensors / Actuators
   – MRBS
3. **Control**
   – Our model
   – Control architecture
   – Control examples
4. **Conclusions and Future Work**

Big Data, Big Problems

Problem:

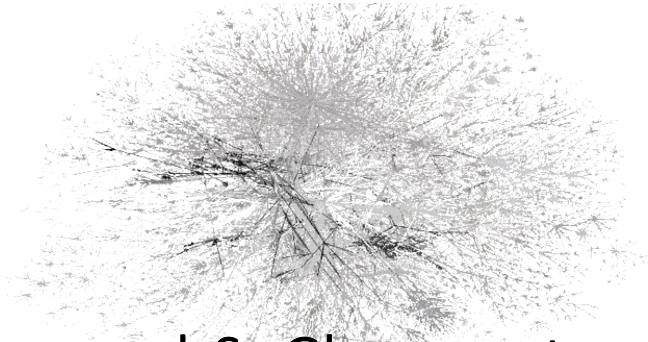
Vast amounts of data generated daily

- Facebook:
  - 1.11 x 10^9 active users, 50% log in daily
  - 3.2 x 10^9 likes and comments/day
  - > 100 clusters (largest has > 100PB, 200 million files)
- CERN's LHC: Up to 1 PB/s during experiments

How do we store it? How do we process it?

Ben Chams - Fotolia

MapReduce



Programming model introduced by J. Dean and S. Ghemawat (Google) in 2004 as a PaaS paradigm -> large scale distributed data processing on clusters of commodity computers

Automatic features: data partitioning and replication, task scheduling, fault tolerance

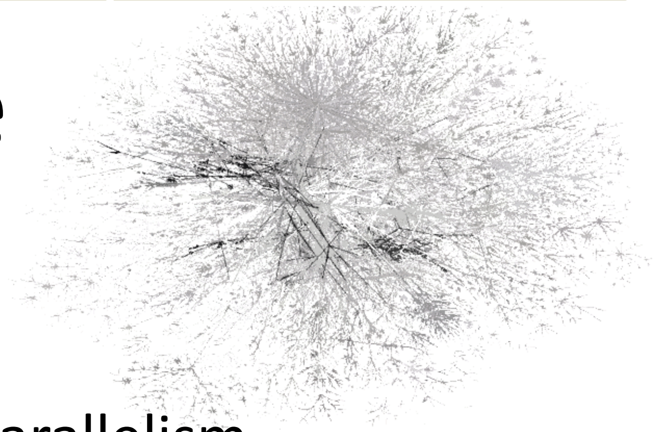
Used by the biggest companies :

Amazon, eBay, Facebook, LinkedIn, Twitter, Yahoo, Microsoft...

Wide range of applications :

log analysis, data mining, web search engines, scientific computing, business intelligence,…

MapReduce



- **Advantages:**
  - Hides many of the complexities of parallelism
  - Usage simplicity and great scalability
- **Challenges**:
  - Difficult to provision for MR, when faced with a changing workload
  - Complex architecture, many points of contention: CPU, IO, network skews, failures, node homogeneity problems

→ assuring SLA performance objectives poses considerable challenges

State of the Art

- Existing models
  - predict the steady state response of MapReduce jobs and do not capture system dynamics

  → not suitable for control using control theory
  - assume that every job is running in a isolated virtual cluster

  → don't deal with concurrent job executions, unlikely in real life scenarios

For modeling, we've essentially started from scratch.

State of the Art

- Existing controls
  - **Focus on static, off-line configuration** optimization for dead-line assurance

  → not robust enough
  - **Dedicated cluster or job priorities**

  → bad performance for jobs not bounded by latency constraints
  - **Job level controllers**, improving on fair scheduler: off-line profile, online adjustment based on job progress

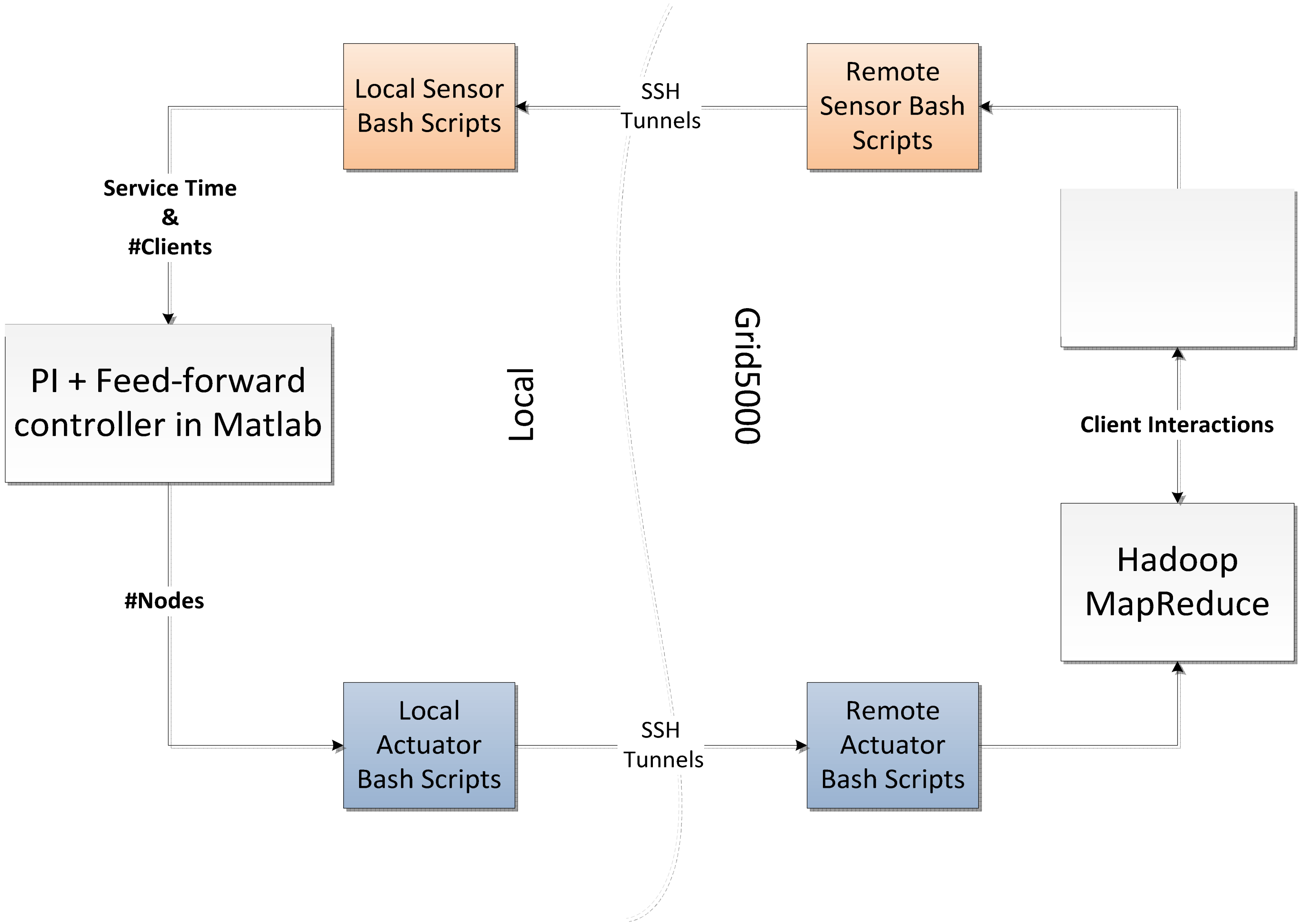
Objectives

- Develop a **dynamical** model for a concurrent MapReduce workload -> holistic, scalable approach
- Develop a test framework for control strategies
- Propose control strategies that assure SLA compliance

Consideration:

- Implementations evolve rapidly, to be relevant, remain agnostic to implementation

Experimental setup



Sensors & Actuators

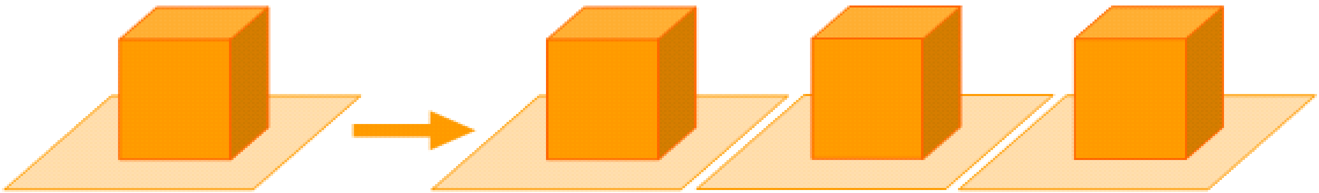
- Linux Bash scripts: shell scripts are widely used in the UNIX world.
  - excellent for speeding up repetitive tasks
  - they can be as simple as a set of commands, or they can orchestrate complex tasks.
- Client/Server Java application

Sensors

- **Problem**: most metrics are not readily available online -> systems not conceived with online measurements in mind
- Non-intrusive approach -> process software logs files online
- Metrics: average performance, availability, throughput in the last time window
- SED, AWK -> Powerful tools to analyze log files

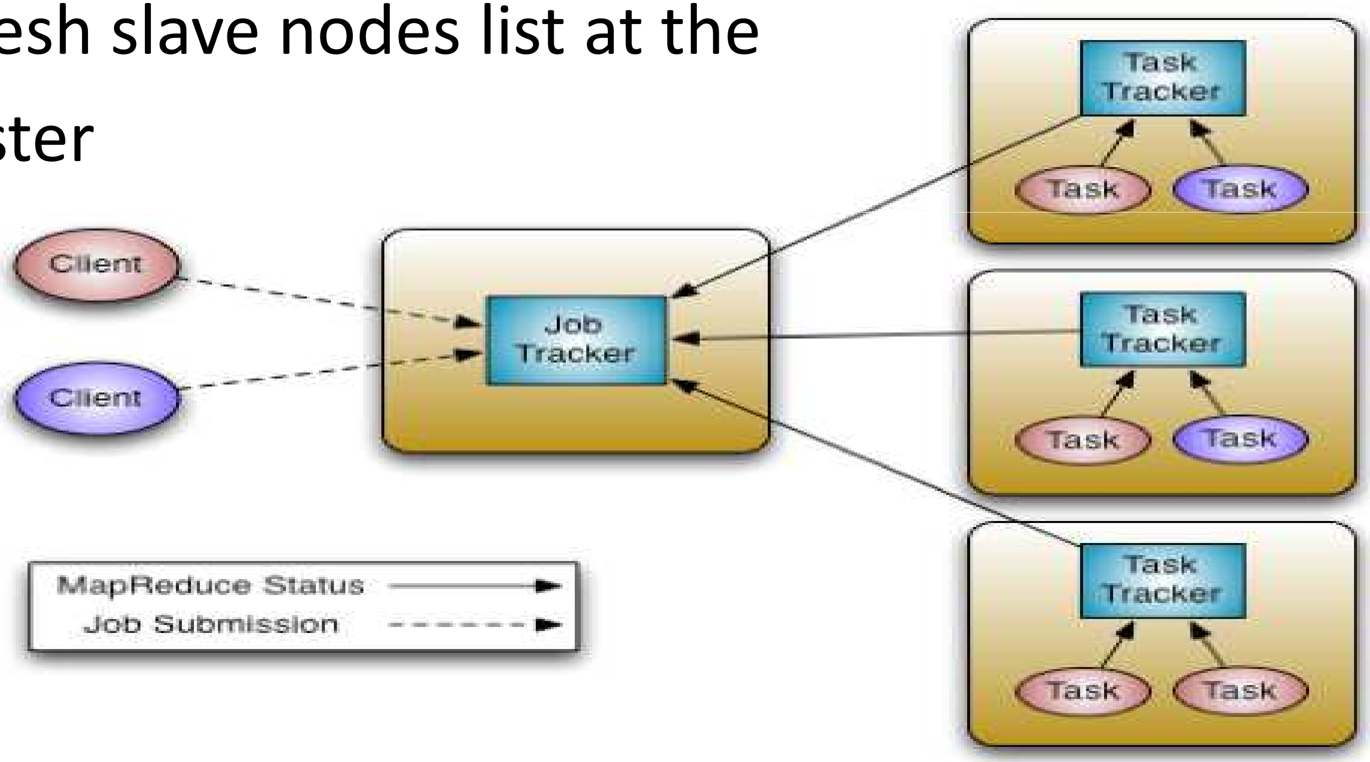
Actuators

- The choice of control inputs out of Hadoop's many parameters (more than 170) is not straightforward.
- Software implementations changing rapidly
  -> remain implementation agnostic
- Number of Mappers and Reducers
- Horizontal scaling: changing the number of nodes



Actuators

- Scripts that start up slave node services
- Refresh slave nodes list at the master



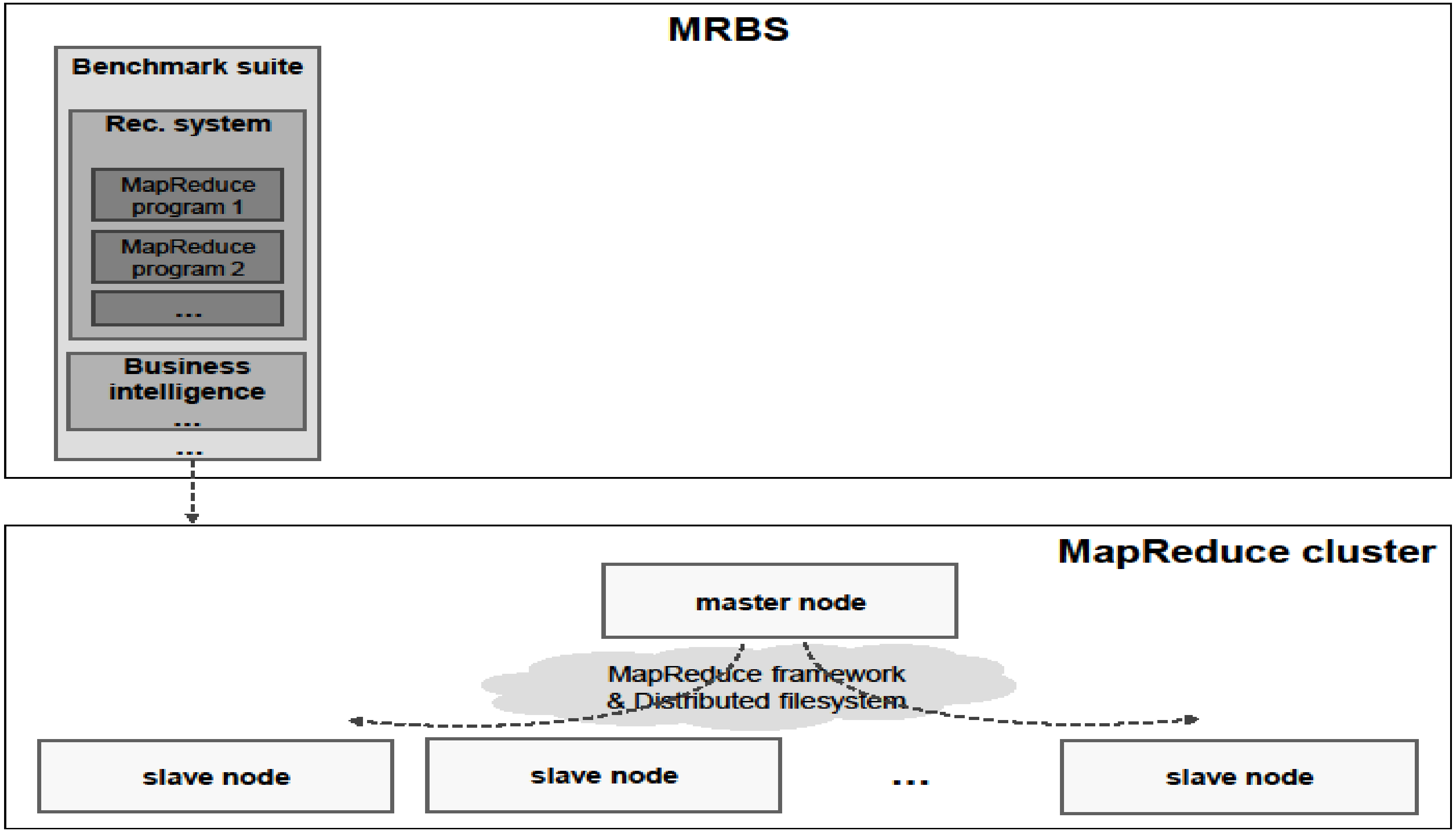
MRBS

- the MapReduce Benchmark Suite (MRBS) developed by Sangroya et al. (2012)
- is a performance and dependability benchmark suite for MapReduce systems.
- most previous evaluations used micro-benchmarks

Advantages:

- representative of fully distributed, concurrent applications
- provide realistic multiuser workloads
- dependability benchmarking

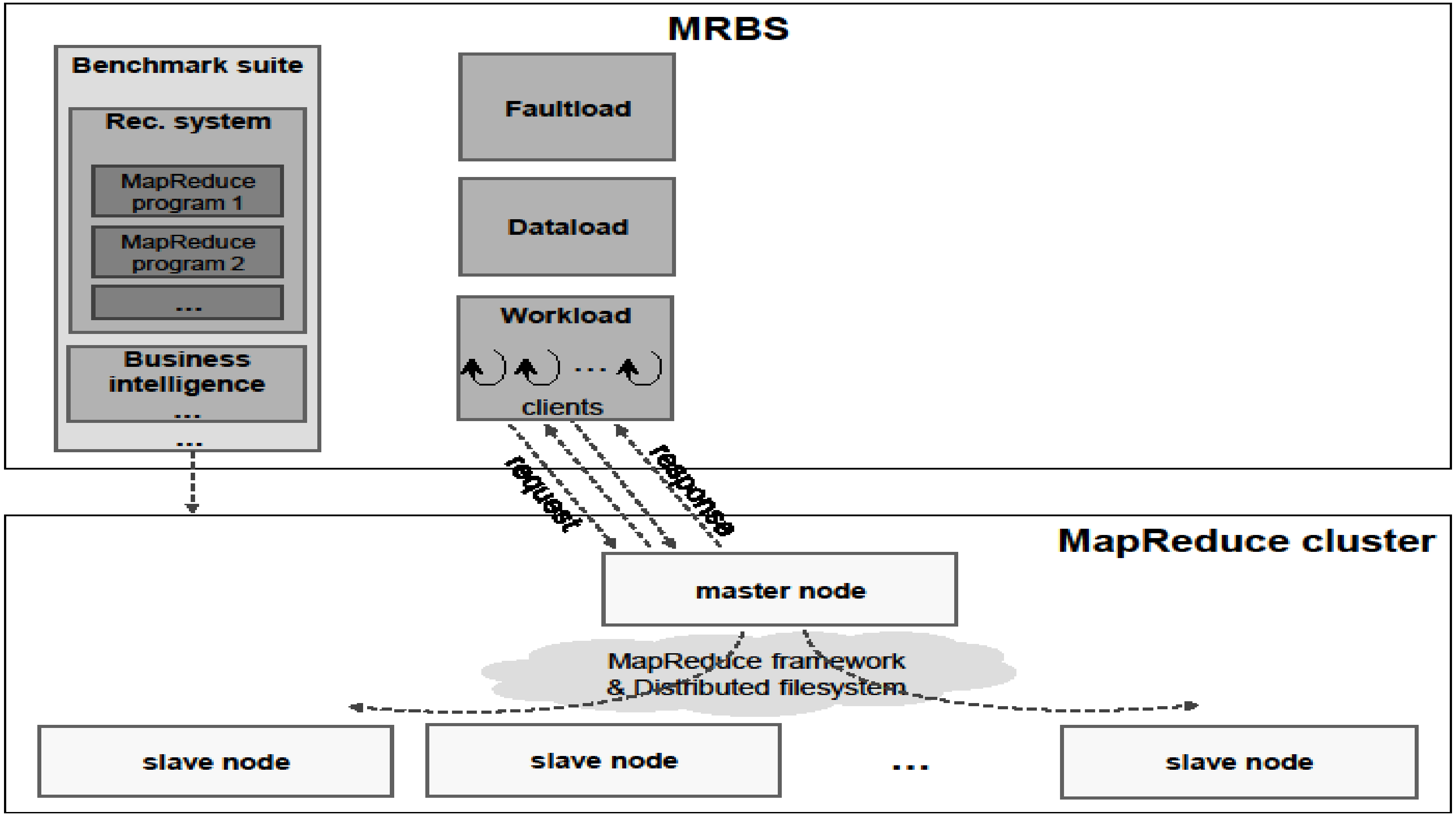
MRBS


MRBS
Benchmark suite
Rec. system
MapReduce program 1
MapReduce program 2
...
Business intelligence
...
...
MapReduce cluster
master node
MapReduce framework & Distributed filesystem
slave node
slave node
...
slave node

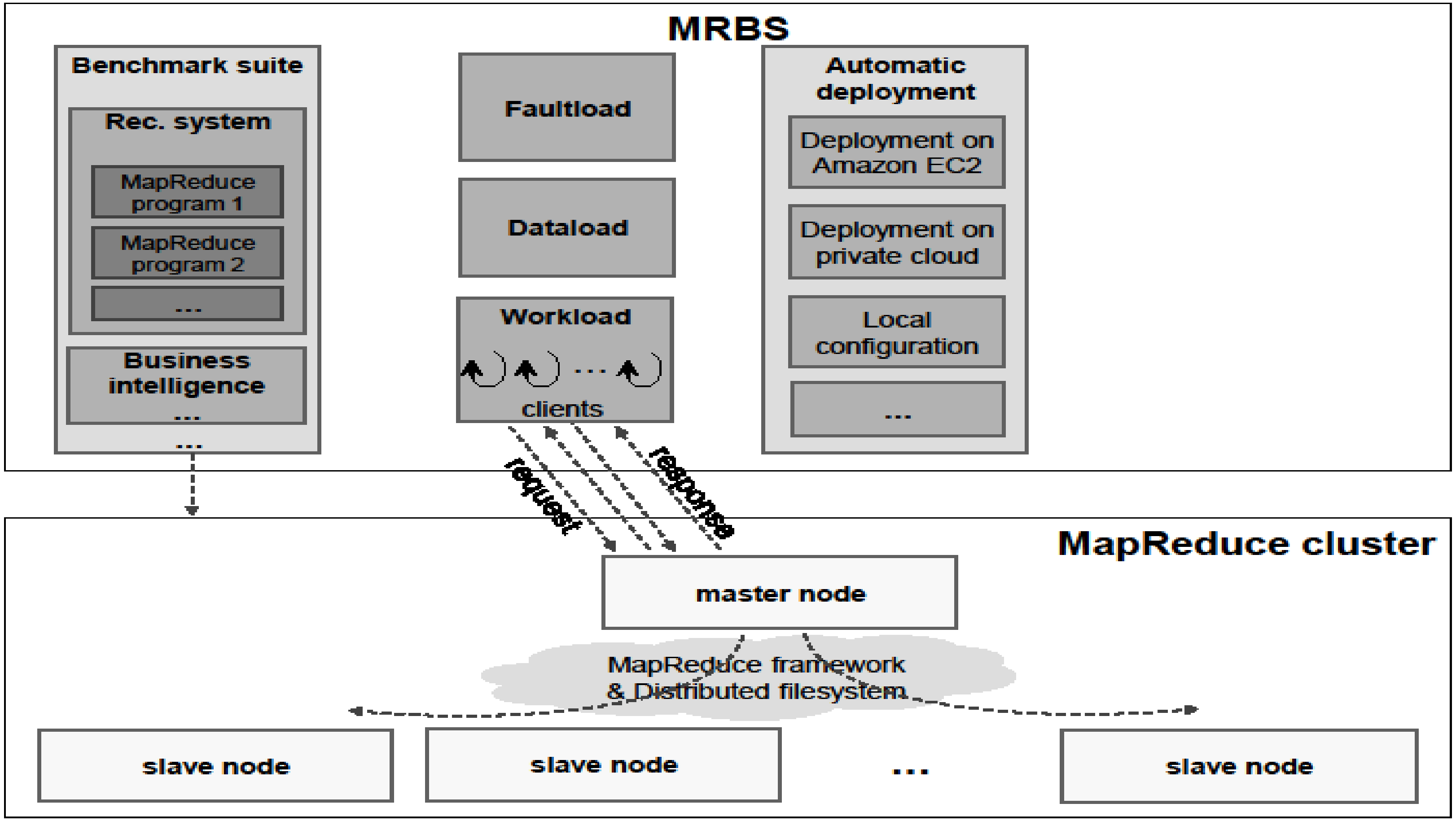
MRBS

Domain	Computation vs. data access	Dataload	Execution mode
Recommendation system	compute-oriented+	dataload 100,000 ratings, 1000 users, 1700 movies	interactive* / batch
		dataload+ 1 million ratings, 6000 users, 4000 movies	
		dataload++ 10 million ratings, 72,000 , 10,000 movies	
Business intelligence	data-oriented+	dataload 1GB	interactive* / batch
		dataload+ 10GB	
		dataload++ 100GB	
Bioinformatics	data-oriented / compute-oriented	dataload genomes of 2,000,000 to 3,000,000 DNA characters	interactive* / batch
Text processing	data-oriented / compute-oriented	dataload text files (1GB)	interactive / batch*
		dataload+ text files (10GB)	
		dataload++ text files (100GB)	
Data mining	data-oriented / compute-oriented	dataload 5000 documents, 5 newsgroups, 600 control charts	interactive / batch*
		dataload+ 10,000 documents, 10 newsgroups, 1200 control charts	
		dataload++ 20,000 documents, 20 newsgroups, 2400 control charts	

MRBS

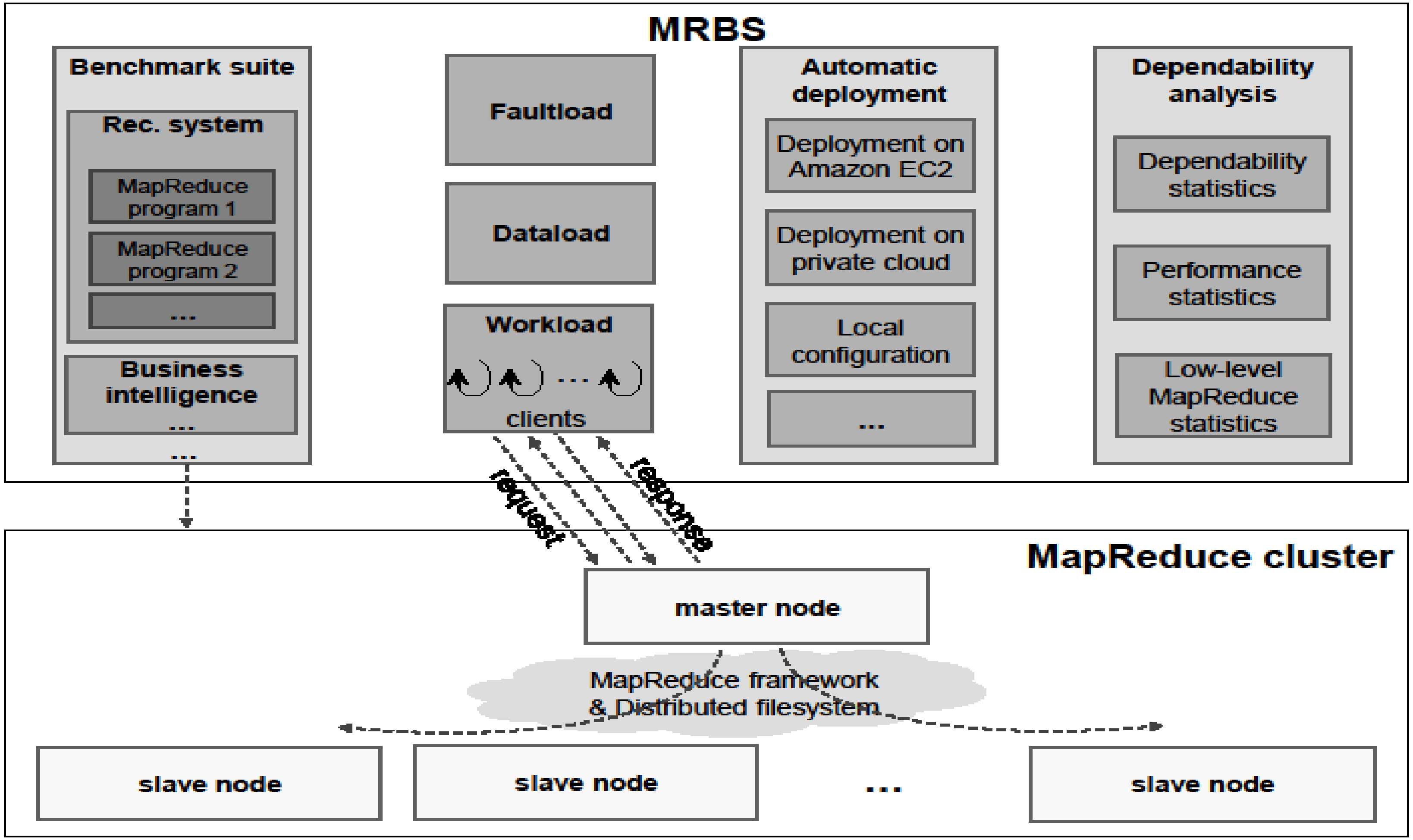

MRBS
Benchmark suite
Rec. system
MapReduce program 1
MapReduce program 2
...
Business intelligence
...
...
Faultload
Dataload
Workload
...
clients
request
response
MapReduce cluster
master node
MapReduce framework
& Distributed filesystem
slave node
slave node
...
slave node

MRBS



MRBS
Benchmark suite
Rec. system
MapReduce program 1
MapReduce program 2
...
Business intelligence
...
...
Faultload
Dataload
Workload
clients
Automatic deployment
Deployment on Amazon EC2
Deployment on private cloud
Local configuration
...
request
response
MapReduce cluster
master node
MapReduce framework & Distributed filesystem
slave node
slave node
...
slave node

MRBS


MRBS
Benchmark suite
Rec. system
MapReduce program 1
MapReduce program 2
...
Business intelligence
...
...
Faultload
Dataload
Workload
clients
Automatic deployment
Deployment on Amazon EC2
Deployment on private cloud
Local configuration
...
Dependability analysis
Dependability statistics
Performance statistics
Low-level MapReduce statistics
request
response
MapReduce cluster
master node
MapReduce framework & Distributed filesystem
slave node
slave node
...
slave node

Experimental setup

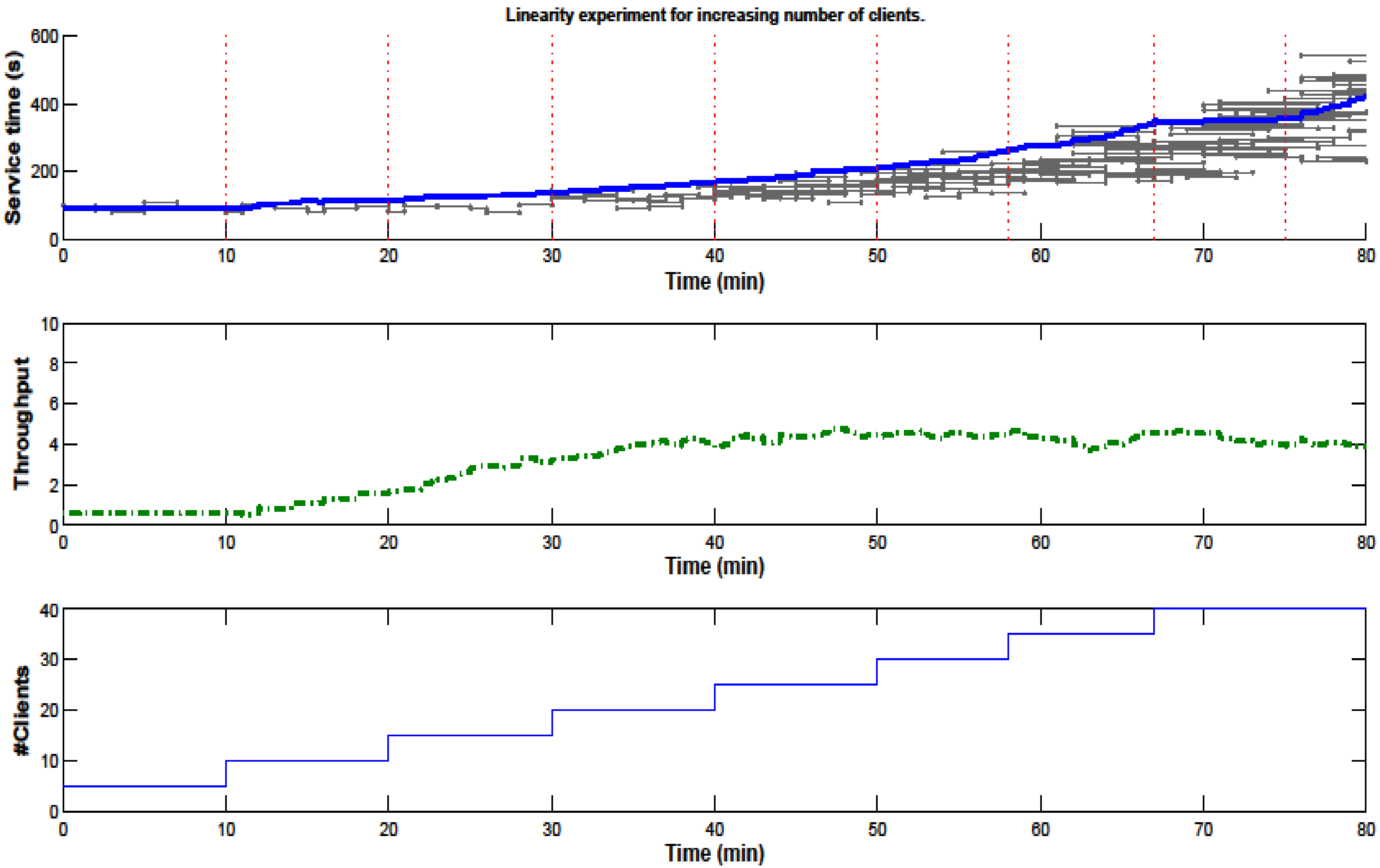
Cluster	CPU	Memory	Storage	Network
60 nodes Grid5000	4 cores/CPU Intel 2.53GHz	15GB	298GB	Infiniband 20G

- data intensive BI workload is selected as our workload
- BI benchmark consists of a decision support system for a wholesale supplier
- request emulate a typical business oriented query that processes a large amount of data (10GB)

Modeling challenges & Insights

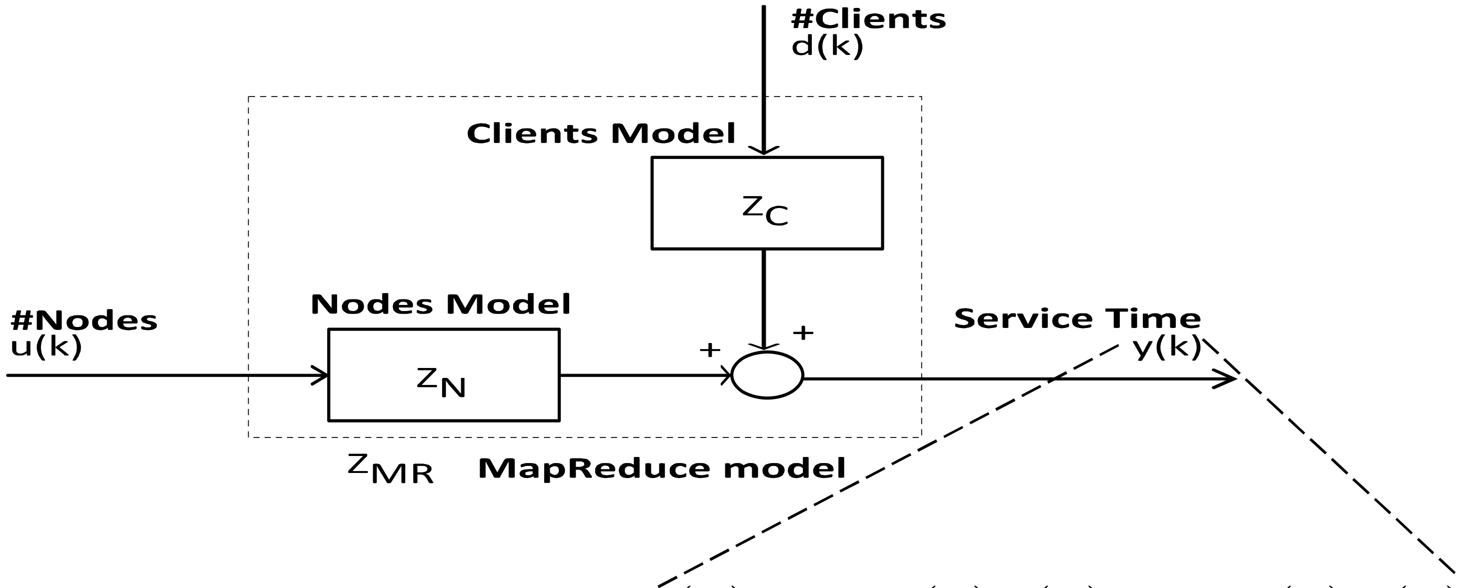
- **Capturing system dynamics**
  - our control objective is selected as keeping the average service time below a threshold in the last time window
- **Implementation agnostic:** parameters that have a high influence regardless of the MapReduce version used
- **Complex system architecture**
  - linearize around an operating point defined by a baseline number of nodes and clients
  - the point of full utilization is the set-point

Clients increasing



Model structure

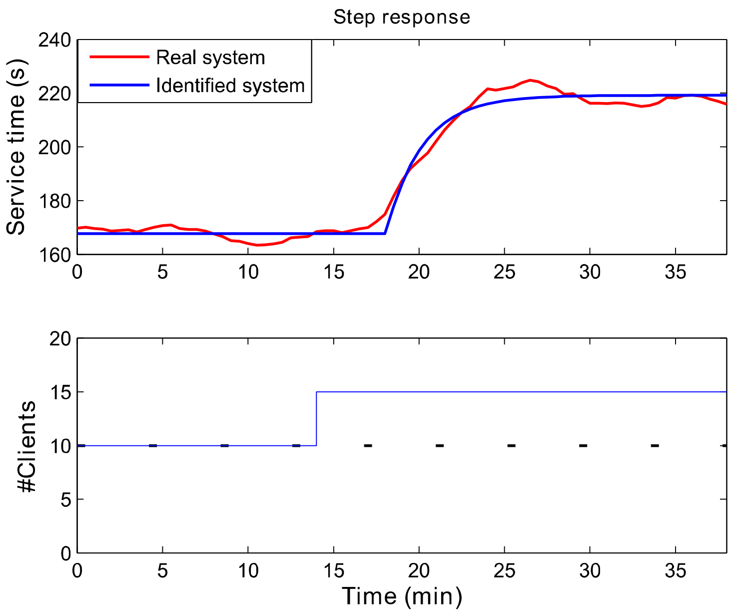
- grey-box modeling technique
- predicts MapReduce cluster performance, in our case average service time, based on the number of nodes and the number of clients

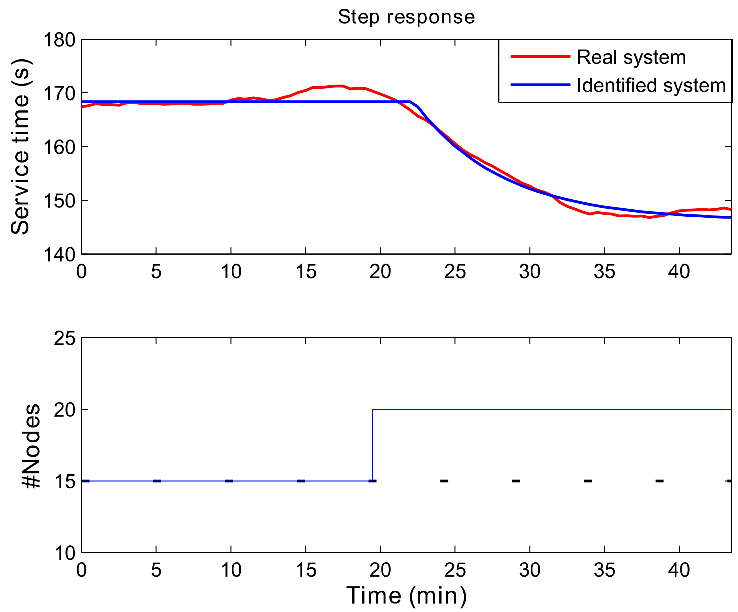


$$y(k) = Z_C(z)d(k) + Z_N(z)u(k)$$

Identification

- both of the models were identified using step response identification (prediction error estimation method)





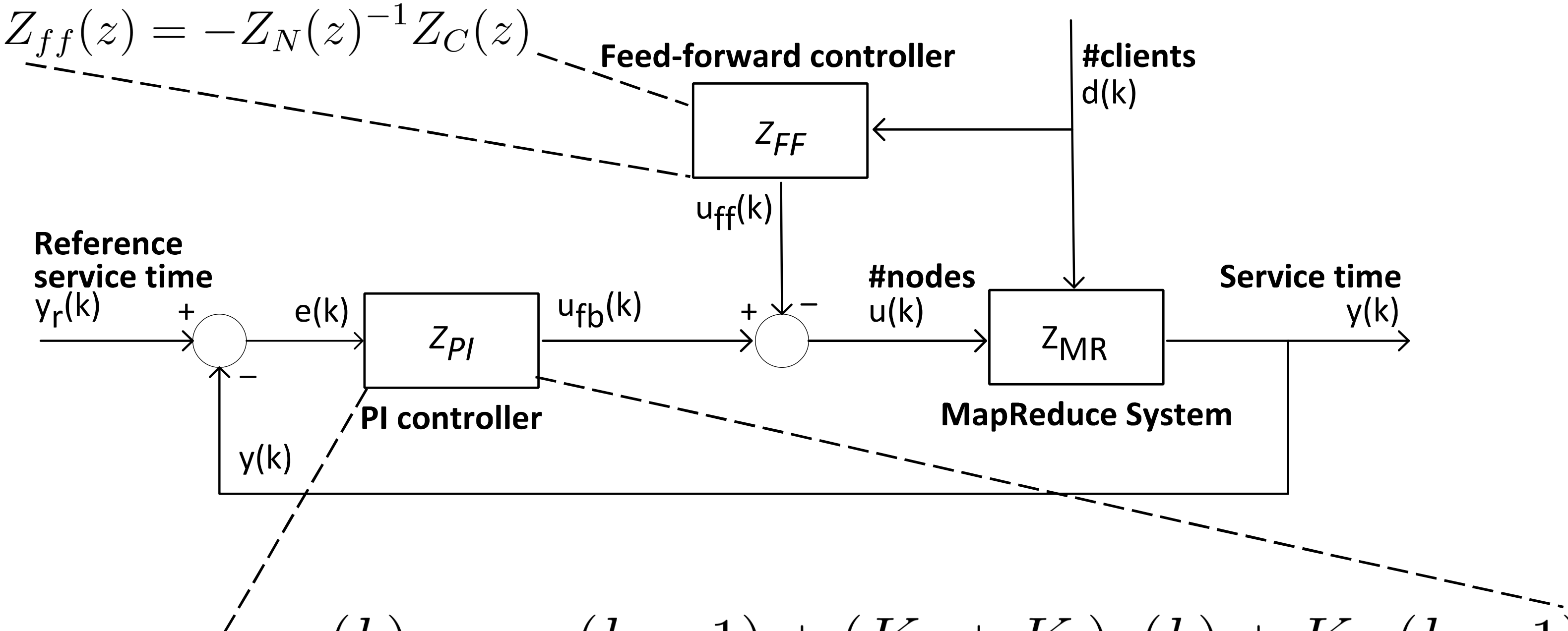
$$Z_C(z) = z^{-8}\frac{1.0716(z+1)}{z-0.7915}$$

$$Z_N(z) = z^{-5}\frac{-0.17951(z+1)}{z-0.919}$$

Control architecture

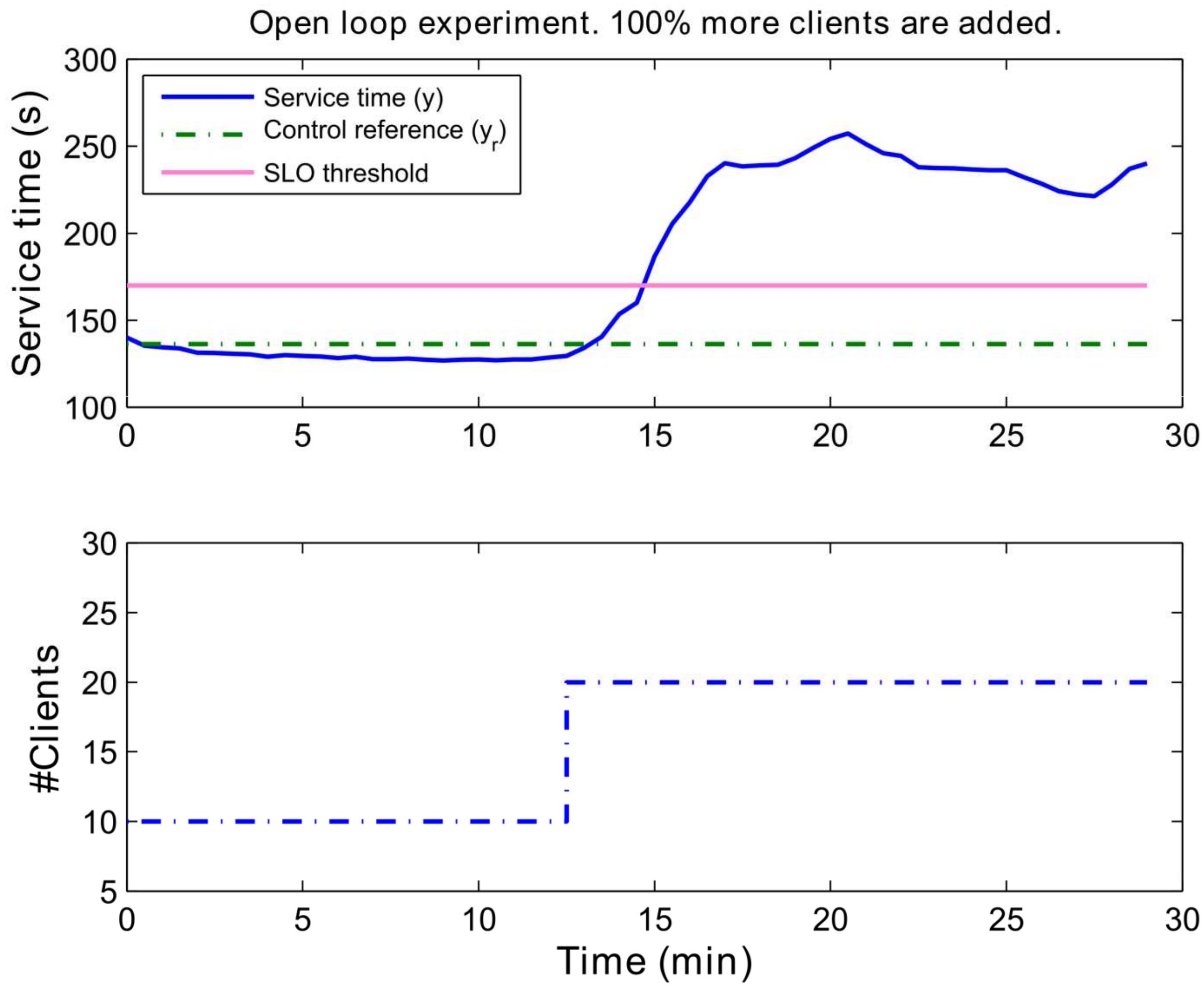
- Challenges:
  - large deadtime
  - as the system performance may very over time because of the many points of contention a robust controller is needed

$$Z_{ff}(z) = -Z_N(z)^{-1} Z_C(z)$$

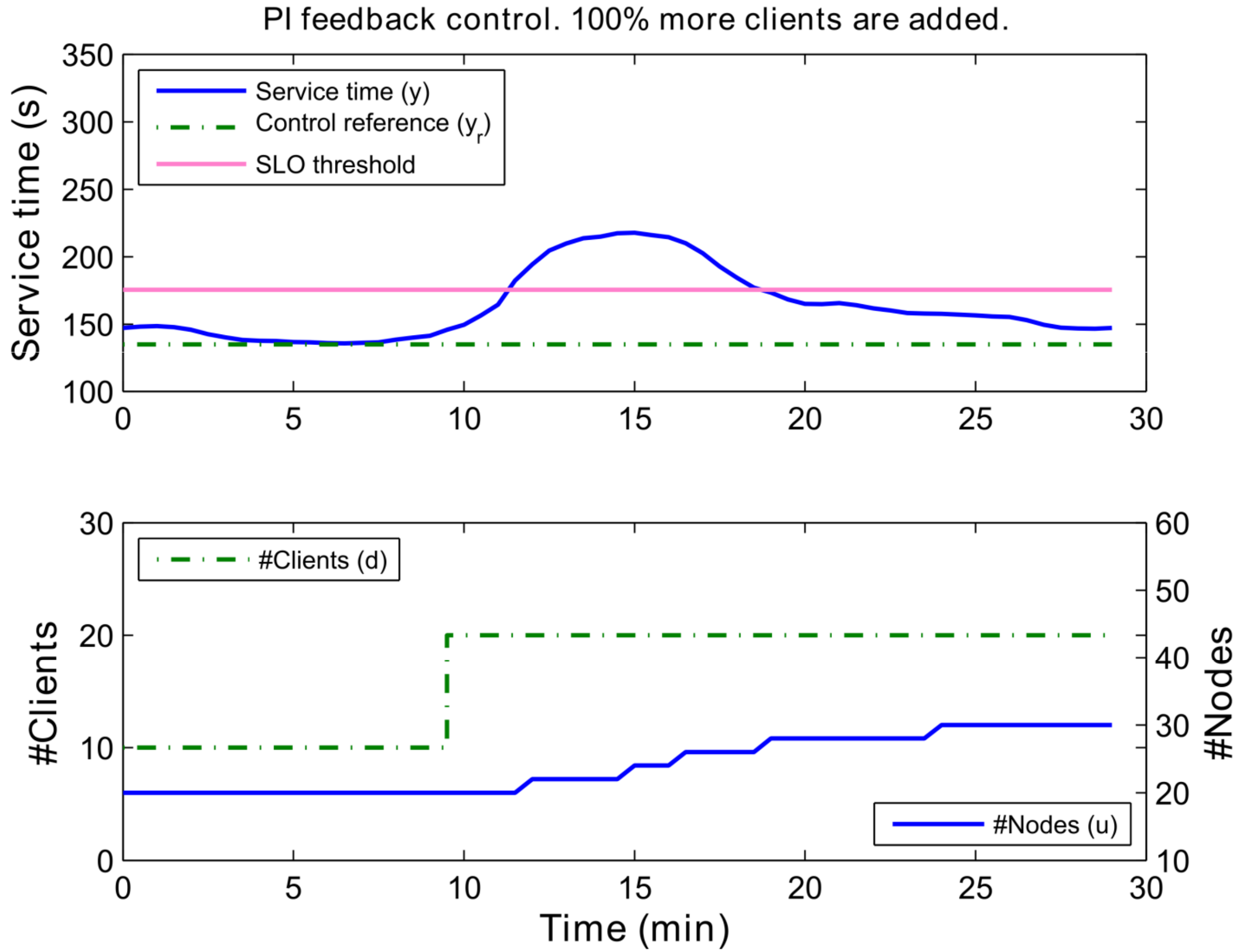


$$u_{fb}(k) = u_{fb}(k-1) + (K_p + K_i)e(k) + K_i e(k-1)$$

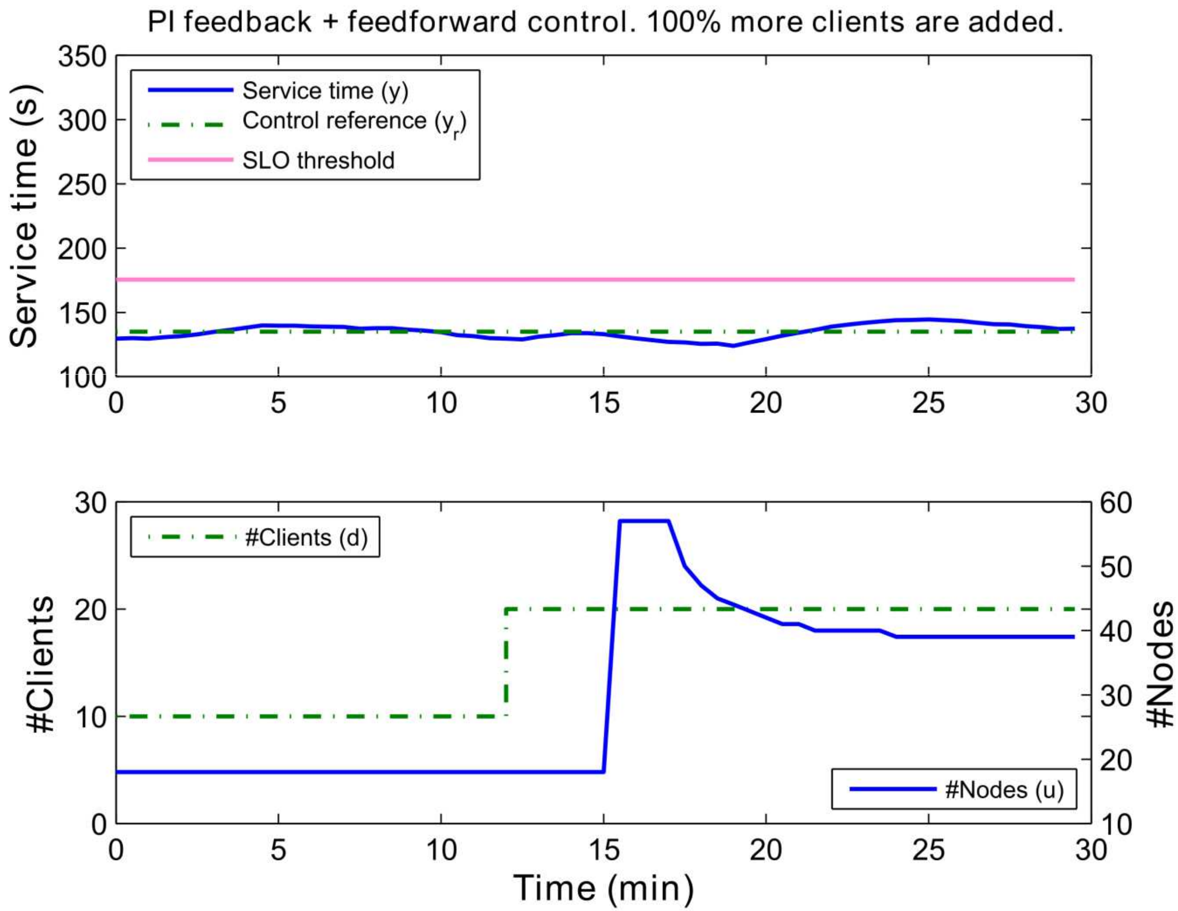
Baseline experiment



RELAXED PERFORMANCE – MINIMAL RESOURCE CONTROL



STRICT PERFORMANCE – PI + FEEDFORWARD CONTROL



Conclusions

- This paper presents:
  - **design, implementation and evaluation of the first dynamic model for MapReduce systems**
  - **development and successful implementation of a control framework for assuring service time constraints**
- The control architecture is implemented on a Hadoop cluster using a data intensive workload
- Our experiments show that the controllers are successful in keeping the SLA

Future Work

- Add other metrics to our model such as throughput, availability, reliability
- Improve upon our identification by making it online
- Minimize the number of changes in the control input. Other control techniques such as an event-based controller for example are being studied now
- Implementing the control framework in several on-line cloud frameworks, with more complex scenarios

Thank you for your attention!
Questions?