Feedback Autonomic Provisioning for guaranteeing performance (and reliability) - application to Big Data Systems

Bogdan Robu
bogdan.robu@gipsa-lab.fr

HIPEAC - HPES Workshop Amsterdam 19-21.01.2015

Context of the work presented here

- Joint work with **Mihaly Berekmeri, Nicolas Marchand** (Control Theory, GIPSA-lab) and **Sara Bouchenak, Damian Serrano** (Computer Science, INSA Lyon)
- Work supported by LabEx PERSYVAL-Lab

Why putting control theory in computer science?

I want my system to be autonomous, i want him to recover if something goes wrong, i want him to be robust ... while minimizing some resource usage.

- How it is done nowadays?
  - feeback + using upper and lower thresholds for adding / removing resources, machine learning (assume homogeneity)
- But what can happen ... ?
  - instability (or very slow response)
  - continuous oscillations along the objective (jitter)
  - ... and what about adding additional delays in the output?

Why putting control theory in computer science?

I want my system to be autonomous, i want him to recover if something goes wrong, i want him to be robust ... while minimizing some resource usage.

- How it is done nowadays?
  - feeback + using upper and lower thresholds for adding / removing resources, machine learning (assume homogeneity)
- But what can happen ... ?
  - instability (or very slow response)
  - continuous oscillations along the objective (jitter)
  - ... and what about adding additional delays in the output?

gipsa-lab

Why putting control theory in computer science?

I want my system to be autonomous, i want him to recover if something goes wrong, i want him to be robust ... while minimizing some resource usage.

- How it is done nowadays?
  - feeback + using upper and lower thresholds for adding / removing resources, machine learning (assume homogeneity)
- But what can happen ... ?
  - instability (or very slow response)
  - continuous oscillations along the objective (jitter)
  - ... and what about adding additional delays in the output?

gipsa-lab

UNIVERSITE JOSEPH FOURIER SCIENCES. TECHNOLOGIE. SANTÉ

Motivation to put control theory in computer science

- Dealing with the dynamics: time is crucial
- Mathematical tools to "control" a system
- By "control", we mean being able to
  - define a control objective
  - define control actions accordingly
  - **guarantee** performances of the controlled system
    - despite errors
    - despite perturbations
    - Facing everything that is unknown
    - Guarantee stability
- Nowadays control theory is everywhere...
  - automotive, robotics, energy (grids, production, etc.), microelectronics, etc.
- ...except maybe in computer science

CHALLENGING DIFFICULTIES

- Language difficulties
  - things with the same name do not mean same thing
- No physics behind algorithms, applications, services, etc.

CHALLENGING DIFFICULTIES

- Language difficulties
  - things with the same name do not mean same thing
- No physics behind algorithms, applications, services, etc.

Challenging difficulties

- Language difficulties
  - things with the same name do not mean same thing
- No physics behind algorithms, applications, services, etc.
  - Building models is critical and unusual
    - How do i put the system in the control theory normal form $\frac{dx}{dt} = f(x,u)$?
    - Control, outputs, sensors, etc. can disappear with a system update
    - Evolution of a system can be discontinuous (robustness issue)
    - No "tiredness", only crashes
  - Model must capture main behavior BUT
    - if too precise ☞ too complex
    - if too complex ☞ inefficient for control (not robust)
    - Model for control is not classical modeling
  - **Requires much more interaction than usual sciences !**

Structure of the presentation

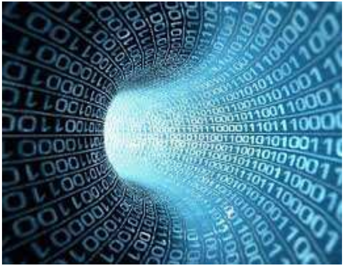
Outline of the talk :

- Application of control to Big Data Clouds : motivation
- Modeling the Big Data MR system
- Controlling the MR system
- Conclusions and perspectives

Big Data - Big Science

Big Data : very big amount of unstructured data !

- "90% of world's data generated over last two years" (Science Daily, May 22, 2013)
- "The volume of business data worldwide, across all companies, doubles every 1.2 years" (eMarketer. October 2013)



(One)Answer : MapReduce

Developed by Google in 2008, it provides a parallel processing model and associated implementation to process huge amount of data.

- Advantages of MapReduce:
  - Hides many of the complexities of parallelism
  - Usage simplicity, scalablity and fault-tolerance
- Challenges of MapReduce:
  - Difficult to provision when faced with a changing workload
  - Complex architecture, node homogeneity problems, many points of contention: CPU, IO, network skews, failures,

Modeling

How to have an idea about the I/O behavior of MapReduce?

How do I translate this behavior into some kind of equation?

Difficulties with (modeling) MapReduce (MR) :

- the performance of MR systems varies from one version to another :-(;
- existing models predict only the steady state response of MRjobs (That's already something !) and do not capture system dynamics :-(;
- existing models assume that every job is running in a isolated virtual cluster.

Modeling

How to have an idea about the I/O behavior of MapReduce?

How do I translate this behavior into some kind of equation?

Difficulties with (modeling) MapReduce (MR) :

- the performance of MR systems varies from one version to another :-(;
- existing models predict only the steady state response of MRjobs (That's already something !) and do not capture system dynamics :-(;
- existing models assume that every job is running in a isolated virtual cluster.

Modeling

How to have an idea about the I/O behavior of MapReduce?

How do I translate this behavior into some kind of equation?

Difficulties with (modeling) MapReduce (MR) :

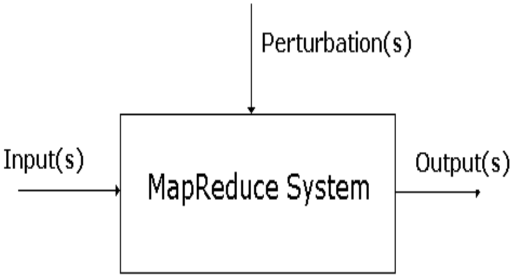
- the performance of MR systems varies from one version to another :-(;
- existing models predict only the steady state response of MRjobs (That's already something !) and do not capture system dynamics :-(;
- existing models assume that every job is running in a isolated virtual cluster.

From where to start?

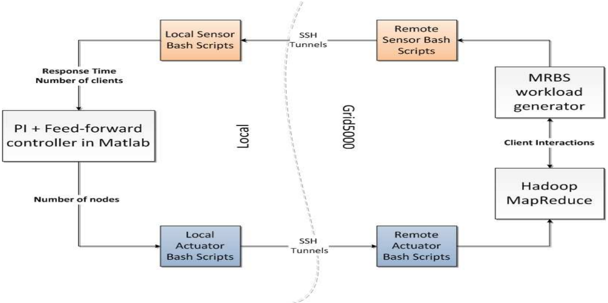
The model needs to be implementation agnostic!

1. Choosing the control inputs:
   - **number of nodes**.
2. What about other things that i can not influence?
   - **number of clients** (disturbance).
3. Choosing the outputs
   - **response time (the time it takes for a client interaction to execute)**.

And this brings us to something like this ...



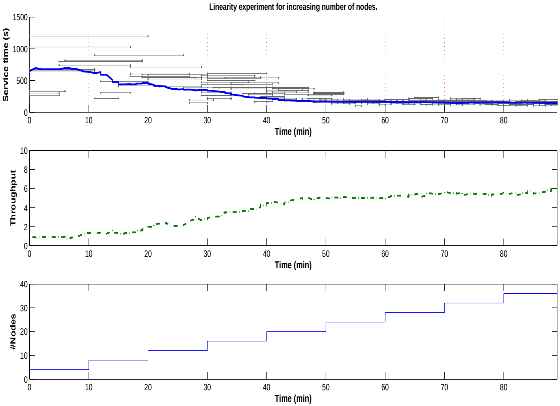
or if we zoom out ... the experimental setup



After the inputs and outputs let's find the input-output relation (*model*)

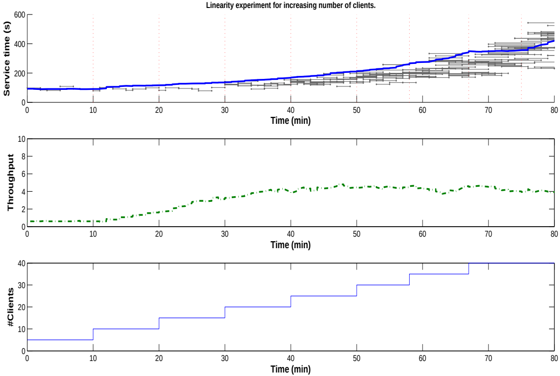
Observing system behavior while nodes are increasing

- number of clients is kept constant = 10.



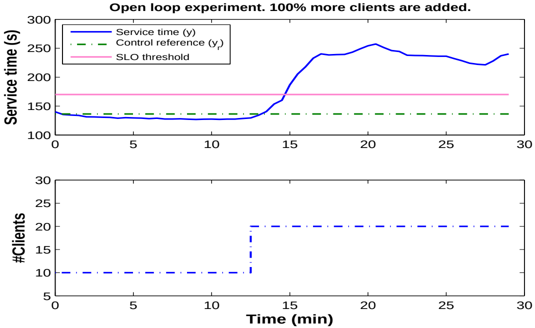
Observing system behavior while clients are increasing

- number of nodes is kept constant = 20.

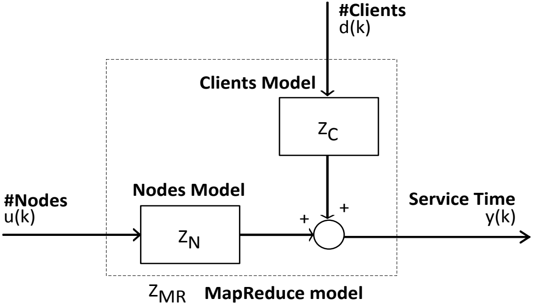


... and during time

Bursty increase in the number of clients (see 10 month log production of Yahoo's supercomputing cluster)



Proposed model structure



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

$$Z_C(z) = z^{-8}\frac{1.0716(z+1)}{z-0.7915} \qquad Z_N(z) = z^{-5}\frac{-0.17951(z+1)}{z-0.919}$$

Control architecture

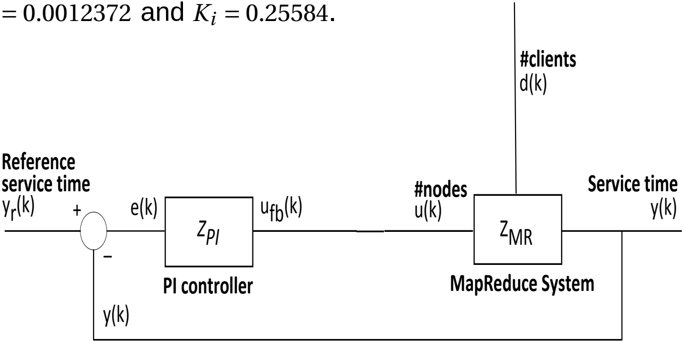
Two versions of control:

1. Relaxed performance Control with Minimal Resources
2. Strict performance Control

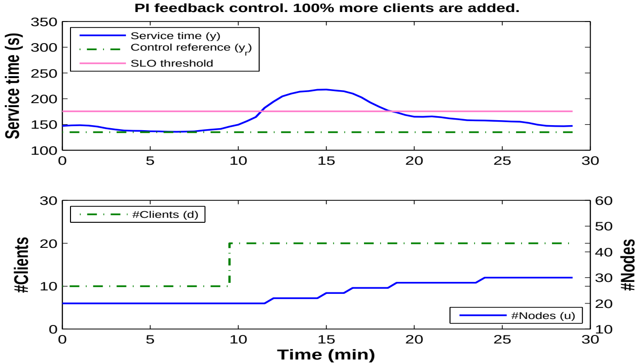
Relaxed performance Control with Minimal Resources

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

where $K_p = 0.0012372$ and $K_i = 0.25584$.



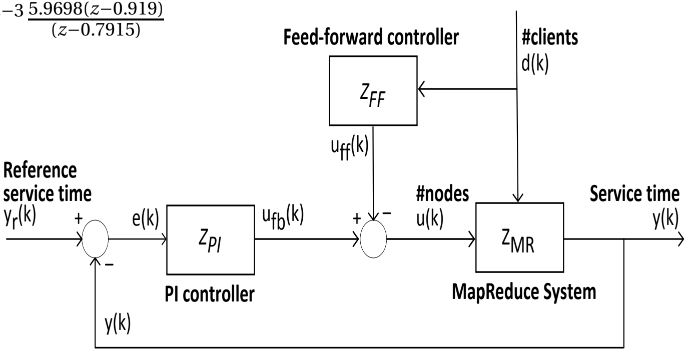
Relaxed performance - Control with Minimal Resources



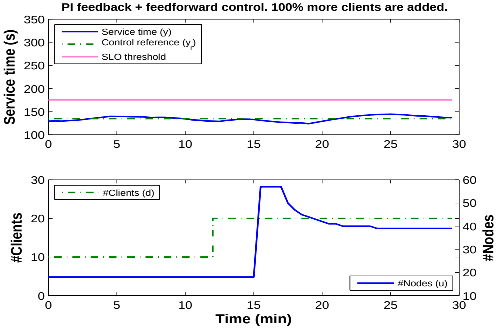
Strict performance Control

Take advantage of the fact that we can measure online the number of clients

$$Z_{ff}(z) = z^{-3}\frac{5.9698(z-0.919)}{(z-0.7915)}$$



Strict performance - Feedforward Control



Can we do better ?

Minimizing the number of quick changes in the control signal !

- adding and removing of resources takes considerable time and has **energetic and monetary cost** !

Solution given by : **Event-based control**

- A new control value is calculated only if the difference between the current error and last error value for which control was calculated is greater than this threshold e_{lim}

Control conclusions

Control is depending on the problem !

Conclusions and Future work

- design, implementation and evaluation of an algorithm for creating dynamic performance models for Big Data MapReduce systems.
- control: relaxed-minimal resource and strict performance constraints while minimizing resource usage.

Many ideas for future work in : optimization, predictive, adaptive control ...

Thank you !

gipsa-lab

UNIVERSITE JOSEPH FOURIER
SCIENCES. TECHNOLOGIE. SANTÉ