Modeling and Control of MapReduce Systems
Mihaly Berekmeri, Sara Bouchenak, Bogdan Robu, Nicolas Marchand

To cite this version:
Mihaly Berekmeri, Sara Bouchenak, Bogdan Robu, Nicolas Marchand. Modeling and Control of MapReduce Systems. ComPAS 2013 - Conférence d’informatique en Parallélisme, Architecture et Système, Jan 2013, Grenoble, France. pp.n/c. hal-00940292

HAL Id: hal-00940292
https://hal.archives-ouvertes.fr/hal-00940292
Submitted on 31 Jan 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
I. Context and Problem Statement.

As our world is becoming ever more digitalized companies have more and more data to process. This steep increase of the unstructured data available is asking for a shift of perspective from the traditional database approach to an efficient distributed computing platform specifically designed for handling petabytes of unstructured information available in jobs such as: personalized advertising, advanced data mining or classification.

II. Cloud computing

Cloud computing, using the pay as you go model, offers an enticing new approach to deploying such task. The dominant new perspective emerging, in large scale data processing, is the programming model called MapReduce.

III. MapReduce

MapReduce is a programming model especially designed for large scale unstructured data processing. Its success lies in its simplicity, scalability and fault-tolerance. The runtime environments, which build upon the MapReduce framework, automatically take care of:
- data partitioning
- replication, data consistency
- load balancing and fault tolerance
- task distribution and scheduling

Only two functions have to be implemented: the Map function and the Reduce function. The Map function takes an input set of (key, value) pair and outputs an intermediate (key, value) pair. The MapReduce framework automatically groups all the values associated with the same keys and forwards it to a Reduce function. The Reduce function processes these values and usually gives as output a reduced set of values. Finally, from a user perspective, the MapReduce framework hides all the messy overhead of parallel computing and lets us focus on the task at hand.

IV. Current problems and development perspectives

Modeling issues: the models are specific to certain job metric, such as job finish time and don't consider latency, availability metrics.
- Most of the models don't account for any kind of faults.
- Randomness of such systems is not incorporated into current models.
- Ad-hoc configuration of MR framework poses many challenges.

SLA problems: a general framework for defining the terms of the Service Level Agreements, for MapReduce jobs, is needed.
- It is very difficult for the user to check that the constraints of the service level agreements are kept, difficult to differentiate between providers.
- Abusive jobs are becoming increasingly difficult to handle in virtualized environments where multiple jobs use the same cluster.

Control Insertion: novel control laws are needed that can enforce the Service Level Agreements, both performance and dependability metrics.
- More energy efficient control laws are desired.

V. Our goals

I. MapReduce Dynamic Model
- Transfer the problems into the control theory framework
- The model has to ease the definition of service level objectives
- Account for uncertainties
- Include faults

II. MapReduce Control Laws
- Provide guarantees in terms of performance, dependability and cost
- Optimize the system configuration
- Minimize deployment costs
- Handle multi-lateness
- Test it on real life systems, such as AmazonEc2, Grid5000