Optimal Capacity Allocation for executing MapReduce Jobs in Cloud Systems

www.modaclouds.eu

M. Malekimajd, A. M. Rizzi, D. Ardagna, M. Ciavotta, M. Passacantando, and A. Movaghar

Agenda

- Introduction

- Problem
  - Problem statement
  - Optimization problem

- Experimental results
  - Design of experiments
  - Accuracy of the time bounds

- Conclusions

MODAClouds

Introduction

MapReduce has attracted interest because:

- analyzes large amounts of unstructured data
- overtakes the scalability level that can be achieved by traditional data warehouses and business intelligence technologies

Cloud Computing paradigm is also emerging

- Pay-per-use approach
- Scalability
- Almost infinite capacity
- Supports perfectly Big Data solution

IDC estimates that by 2020, nearly 40% of Big Data analyses will be supported by public cloud.

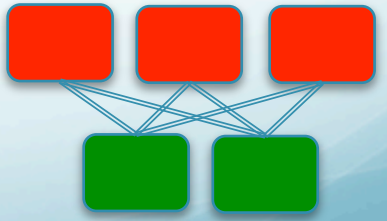
MODAClouds

Introduction: MapReduce

MapReduce: A general **algorithm**, and is prevalent in **functional programming** languages, which supports the notion of **map** and **reduce functions**

MapReduce Architecture provides:

- **Automatic** parallelization & distribution
- **Fault tolerance**
- Task **scheduling**
- **Monitoring** & status updates



MODAClouds

Introduction: Challenges

Scheduling

- Order and amount of resource allocated to a job
- Job ordering defines which job should be processed next
- Scheduling policy decides how many map/reduce slots should be allocated to the current job

MODAClouds

MR Schedulers

- First In First Out (FIFO) scheduler
- Fair scheduler
- Capacity scheduler
- Other ad-hoc schedulers

MODAClouds

Fair Scheduler

- Guarantees that short jobs finish in reasonable time
- Jobs organized into pools
- Guaranteed minimum shares
- Preemption
- Limit the number of concurrent running jobs/tasks

MODAClouds

Agenda

- Introduction
- Problem
  - Problem statement
  - Optimization problem
- Experimental results
  - Design of experiments
  - Accuracy of the time bounds
- Conclusions

Problem statement

Shared Hadoop 2.x system

- Fair or capacity scheduler
- Serving a set $\mathcal{U}$ of users
- Each user requests the execution of a job from class $\mathcal{J}_i$ of jobs having a similar execution profile
- Each class $\mathcal{J}_i$ is executed on s_M^i Map slots and on s_R^i Reduce slots with a concurrency degree of h_i (i.e., h_i jobs with profile $\mathcal{J}_i$ are executed concurrently)
- Admission control mechanism bounding the number of concurrent jobs $H_i^{low} \leq h_i \leq H_i^{up}$
- Set of (soft) deadlines D_i

MODAClouds

Time bounds

- Map

$$T^{low}_{M^i} = \frac{N^i_M \, M^i_{avg} \, h_i}{S^i_M}$$

$$T^{up}_{M^i} = \frac{(N^i_M M^i_{avg} - 2M^i_{max})h_i}{S^i_M} + 2M^i_{max}$$

- Typical Shuffle

$$T^{low}_{Sh^i} = (\frac{N^i_R \, h_i}{S^i_R} - 1)Sh^i_{avg}$$

$$T^{up}_{Sh^i} = \frac{(N^i_R Sh^i_{avg} - 2Sh^i_{max})h_i}{S^i_M} + 2Sh^i_{max}$$

- Reduce

$$T^{low}_{R^i} = \frac{N^i_R \, R^i_{avg} \, h_i}{S^i_R}$$

$$T^{up}_{R^i} = \frac{(N^i_R R^i_{avg} - 2R^i_{max})h_i}{S^i_R} + 2R^i_{max}$$

MODAClouds

Time bounds

Putting all together:

$$T_i^{low} = T_{M^i}^{low} + Sh_{avg}^{1(J_i)} + T_{Sh^i}^{low} + T_{R^i}^{low}$$
$$T_i^{up} = T_{M^i}^{up} + Sh_{max}^{1(J_i)} + T_{Sh^i}^{up} + T_{R^i}^{up}$$

After some algebra:

$$T_i^{low} = A_i^{low} \frac{h_i}{s_M^i} + B_i^{low} \frac{h_i}{s_R^i} + C_i^{low}$$
$$T_i^{up} = A_i^{up} \frac{h_i}{s_M^i} + B_i^{up} \frac{h_i}{s_R^i} + C_i^{up}$$

Decision variables

MODAClouds

Problem statement

Cloud environment hosting Hadoop

- provides *on-demand* and *reserved* VMs
- each VM supports c_M^i Map and c_R^i Reduce concurrent tasks for each $J_i \in \mathcal{J}_i$

The overall execution cost is:

$$\delta d + \rho r + \sum_{i=1}^{c} p_i (H_i^{up} - h_i)$$

Decision variables

MODAClouds

Problem formulation P0

The optimization problem can be defined as follows:

$$(P0) \qquad \min \ \delta d + \rho r - \sum_{i \in \mathcal{U}} p_i \, h_i$$

Subject to:

$$\frac{A_i \, h_i}{s_M^i} + \frac{B_i \, h_i}{s_R^i} + E_i \leq 0, \quad \forall i \in \mathcal{U}$$

$$r \leq \bar{r}$$

$$\sum_{i \in \mathcal{U}} \left(\frac{s_M^i}{c_M^i} + \frac{s_R^i}{c_R^i} \right) \leq r + d$$

$$H_i^{low} \leq h_i \leq H_i^{up}, \quad \forall i \in \mathcal{U}$$

$$r \geq 0$$

$$d \geq 0$$

$$s_M^i \geq 0, \quad \forall i \in \mathcal{U}$$

$$s_R^i \geq 0, \quad \forall i \in \mathcal{U}$$

Constraints on the deadlines (not convex)

Total number of VMs

MODAClouds

Problem formlation P1

Convexification: $\Psi_i = 1/h_i$

$$(P1) \qquad min\ \delta d + \rho r - \sum_{i \in \mathcal{U}} \frac{p_i}{\Psi_i}$$

Subject to:

$$\begin{aligned}
\frac{A_i}{s_M^i \Psi_i} + \frac{B_i}{s_R^i \Psi_i} + E_i \leq 0, \quad & \forall i \in \mathcal{U}, \\
r \leq \bar{r}, & \\
\sum_{i \in \mathcal{U}} \left(\frac{s_M^i}{c_M^i} + \frac{s_R^i}{c_R^i} \right) \leq r + d, & \\
\Psi_i - \Psi_i^{up} \leq 0, \quad & \forall i \in \mathcal{U}, \\
-\Psi_i + \Psi_i^{low} \leq 0, \quad & \forall i \in \mathcal{U}, \\
r \geq 0, & \\
d \geq 0, & \\
s_M^i \geq 0, \quad & \forall i \in \mathcal{U}, \\
s_R^i \geq 0, \quad & \forall i \in \mathcal{U},
\end{aligned}$$

MODAClouds

Problem formulation P2

Theorem

In any optimal solution of problem (P1) the number of slots to be allocated to job class $\mathcal{J}_i$, s_M^i and s_R^i, can be evaluated as follows:

$$s_M^i = -\frac{1}{E_i \Psi_i}\left(\sqrt{\frac{A_i\, B_i\, c_M^i}{c_R^i}} + A_i\right), \tag{1}$$

$$s_R^i = -\frac{1}{E_i \Psi_i}\left(\sqrt{\frac{A_i\, B_i\, c_R^i}{c_M^i}} + B_i\right). \tag{2}$$

- Tip: The problem is convex and Slater's condition holds, then we can apply KKT and Slackness conditions for optimality

Problem formulation P2

Final formulation

$$(P2) \qquad \min \delta d + \rho r - \sum_{i \in \mathcal{U}} p_i \, h_i$$

Subject to:

$$r \leq \bar{r}$$

$$\sum_{i \in \mathcal{U}} (\gamma_i^1 + \gamma_i^2) \, h_i \leq r + d$$

$$H_i^{low} \leq h_i \leq H_i^{up}, \forall i \in \mathcal{U}$$

$$r \geq 0$$

$$d \geq 0$$

Where

$$\gamma_i^1 = -\frac{1}{E_i \, c_R^i} \left(\sqrt{\frac{A_i \, B_i \, c_R^i}{c_M^i}} + B_i \right)$$

$$\gamma_i^2 = -\frac{1}{E_i \, c_M^i} \left(\sqrt{\frac{A_i \, B_i \, c_M^i}{c_R^i}} + A_i \right)$$

MODAClouds

Agenda

- Introduction
- Problem
  - Problem statement
  - Optimization problem
- Experimental results
  - Scalability analysis
  - Accuracy of the time bounds
- Conclusions

Experimental results

- Instances randomly generated.
- MapReduce application parameters we considered real log traces: Twitter, Sort, WikiTrends and WordCount.
- Amazon EC2 prices for VM hourly costs

Estimated ranges for MpReduce Cluster

Job Profile	
N_M^i	(70,700)
N_M^i	(32,64)
M_{max}^i (s)	(16,120)
$Sh_{max}^{typ^i}$ (s)	(30,150)
R_{max}^i (s)	(15,75)
$Sh_{max}^{1(i)}$ (s)	(10,30)
c_M^i, c_R^i	(1,4)
D_i (s)	(600,1200)
Cluster Scale	
H_i^{up} (¢)	(10,30)
Job Rejection Penalty	
p_i (¢)	(250,2500)
Cloud Instance Price	
ρ	(5,20)

MODAClouds

Scalability analysis

- VirtualBox virtual machine based on Ubuntu 12.04 server running on an Intel Xeon Nehalem dual socket quad-core system with 32 GB of RAM.
- CPLEX 12.0
- We varied the cardinality of the set $\mathcal{U}$ between 20 and 1,000 with step 20, and run each experiment ten times.
- The time required to solve the MILP problem is, on average, less than 0.08 seconds.
- The instances of maximum size (1,000 user classes) can be solved in less than 0.5 second in the worst case.

MODAClouds

Accuracy of the time bounds

- The aim is to compare the time bounds and against the execution times obtained via YARN SLS,
- One to one mapping between simulated and real times.
- We generated actual traces from from log traces available from Twitter, Sort, WikiTrends and WordCount (2 and 3 classes of jobs and with a random number of users for each class varying between 2 and 10).
- YARN SLS cannot reserve virtual slot to maps or reduce task or accurately simulate the shuffle phase.
- The formulae of the bounds had to be simplified.

MODAClouds

bounds

2 jobs analysis

- the gap between the upper bound and the jobs mean execution time is around 19% on average.
- the gap with respect to $m_i = (T_i^{low} + T_i^{up})/2$ is only 10% on average.

Twitter			Sort		
No. of users	T_1^{up} gap	m_1 gap	No. of users	T_2^{up} gap	m_2 gap
4	7.78%	1.24%	10	6.04%	0.61%
6	6.35%	-0.09%	8	8.83%	3.26%
5	18.53%	7.68%	4	19.79%	10.42%
4	16.10%	6.43%	6	12.79%	4.79%
8	6.70%	-11.98%	7	2.85%	-7.48%
3	17.04%	7.12%	7	14.24%	5.64%
6	4.65%	-9.80%	10	6.35%	-10.80%
6	2.26%	-5.07%	6	5.07%	-1.58%
9	0.49%	-4.94%	7	2.43%	-2.44%
4	8.24%	1.56%	10	5.28%	-0.45%

MODAClouds

bounds

3 jobs analysis

- the gap between the upper bound and the jobs mean execution time is around 11% on average,
- the gap with respect to $m_i = (T_i^{low} + T_i^{up})/2$ is only 5% on average.

Sort			WordCount			WikiTrends		
N.	T_1^{up} gap	m_1 gap	N.	T_2^{up} gap	m_2 gap	N.	T_3^{up} gap	m_3 gap
4	5.15%	-1.51%	4	6.28%	-2.33%	4	15.77%	9.73%
4	8.01%	-0.12%	3	9.48%	-0.97%	3	23.12%	15.46%
5	2.56%	-4.51%	2	2.48%	-6.51%	4	14.40%	7.90%
4	8.79%	-0.22%	2	9.02%	-2.38%	3	16.17%	8.19%
2	3.54%	-4.25%	3	7.03%	-3.18%	5	11.32%	4.39%
2	13.98%	5.07%	3	13.05%	1.18%	4	19.55%	11.28%
4	14.74%	6.60%	2	14.79%	3.80%	4	21.14%	13.55%
5	8.64%	1.60%	4	6.90%	-2.51%	2	14.52%	7.97%
5	0.91%	-5.64%	2	2.26%	-6.75%	4	9.64%	3.37%
4	11.02%	4.40%	4	7.84%	-0.93%	4	14.43%	8.42%

MODAClouds

Agenda

- Introduction
- Problem
  - Problem statement
  - Optimization problem
- Experimental results
  - Scalability analysis
  - Accuracy of the time bounds
- Conclusions

MODAClouds

Conclusions and Future Works

We proposed an optimization model able to minimize the execution costs of heterogeneous tasks in cloud based shared Hadoop clusters.

Our model is based on novel upper and lower bounds for MapReduce job execution time

Our method is fast and scalable (optimal solutions for systems up to 1,000 user classes found in less than 0.5 seconds).

The average execution time of MapReduce jobs obtained through simulations is within 14% of our bounds on average.

Future works:

- a distributed implementation able to exploit the YARN hierarchical architecture

MODAClouds

Thank You!

Questions?

MODAClouds

Introduction

A recent McKinsey analysis has shown how,

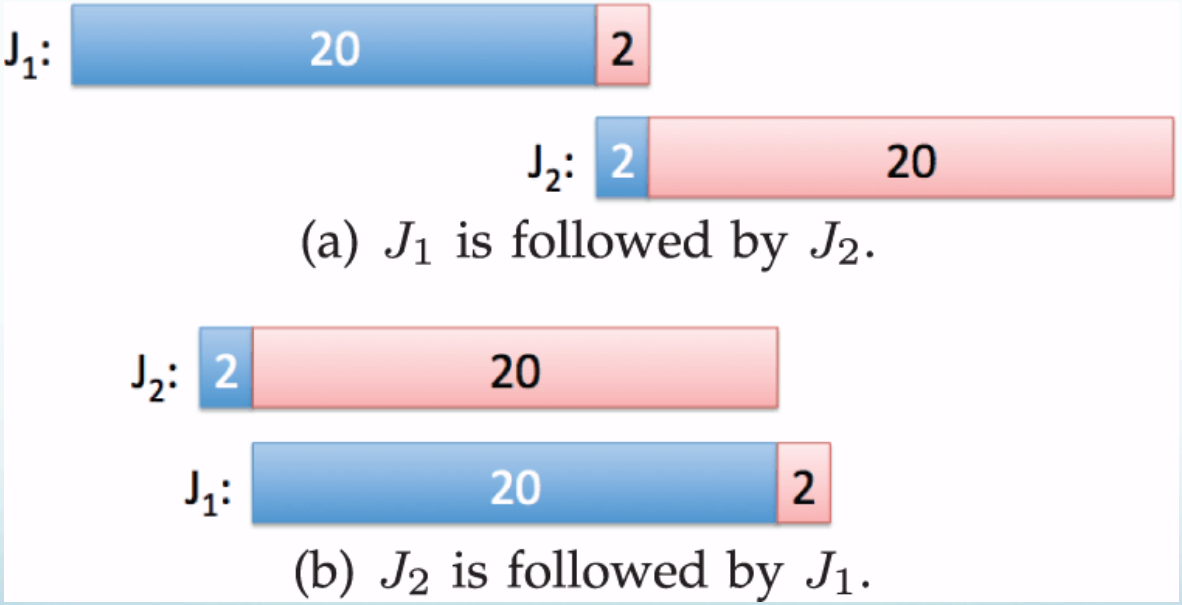
- Big Data could produce \$300 billion potential annual value to US health care
- Europe public sector could potentially reduce expenditure of administrative activities by 15--20\%.

- MapReduce programming model is the most prominent solution for Big Data applications
- Its open source implementation, Hadoop, is able to manage large datasets over either commodity clusters and high performance distributed topologies.

MODAClouds

Scheduling

- Challenges:
  - Delay for short jobs
  - Not efficient use

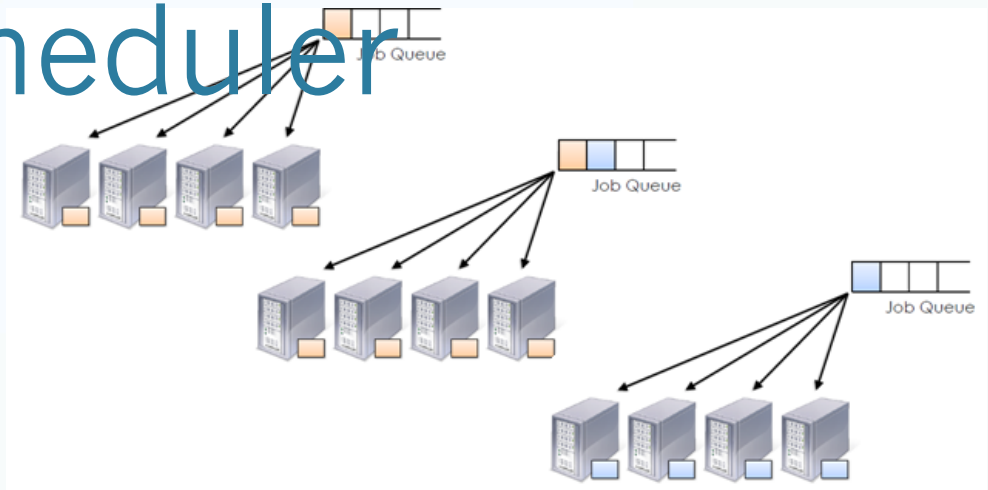


(a) J_1 is followed by J_2.

(b) J_2 is followed by J_1.

MODAClouds

FIFO Scheduler



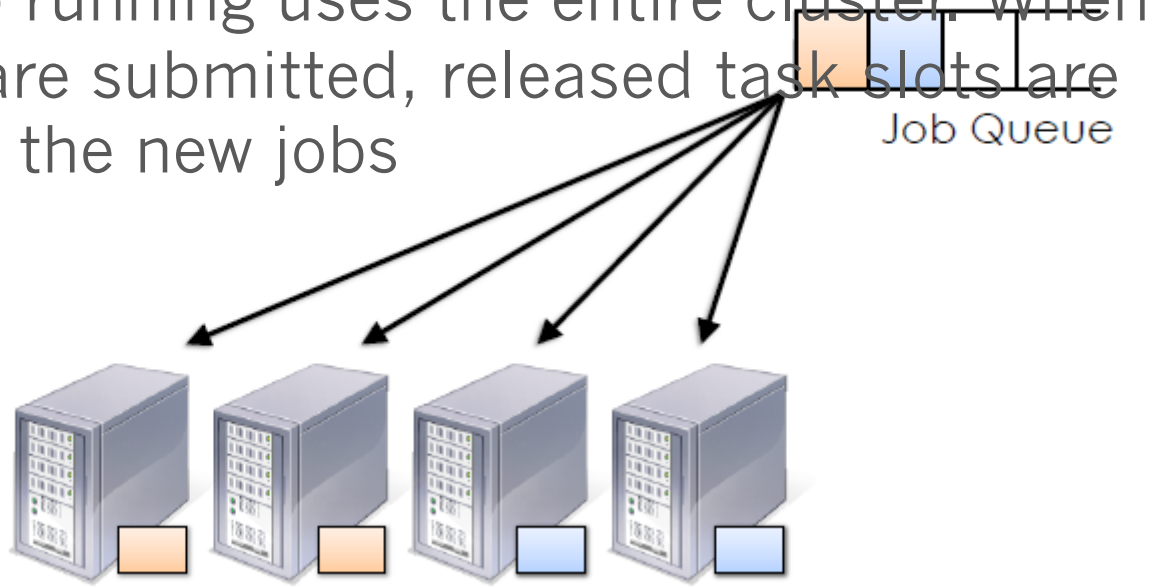
- Forms a queue of jobs
- In FIFO scheduling, a JobTracker pulled jobs from a work queue, oldest job first
- No concept of the priority or size of the job, but the approach was simple to implement and efficient.

MODAClouds

Fair Scheduler

- Assigning resources to jobs such that all jobs receive, on average, an equal share of resources over time

- A single job running uses the entire cluster. When other jobs are submitted, released task slots are assigned to the new jobs



MODAClouds

Capacity Scheduler

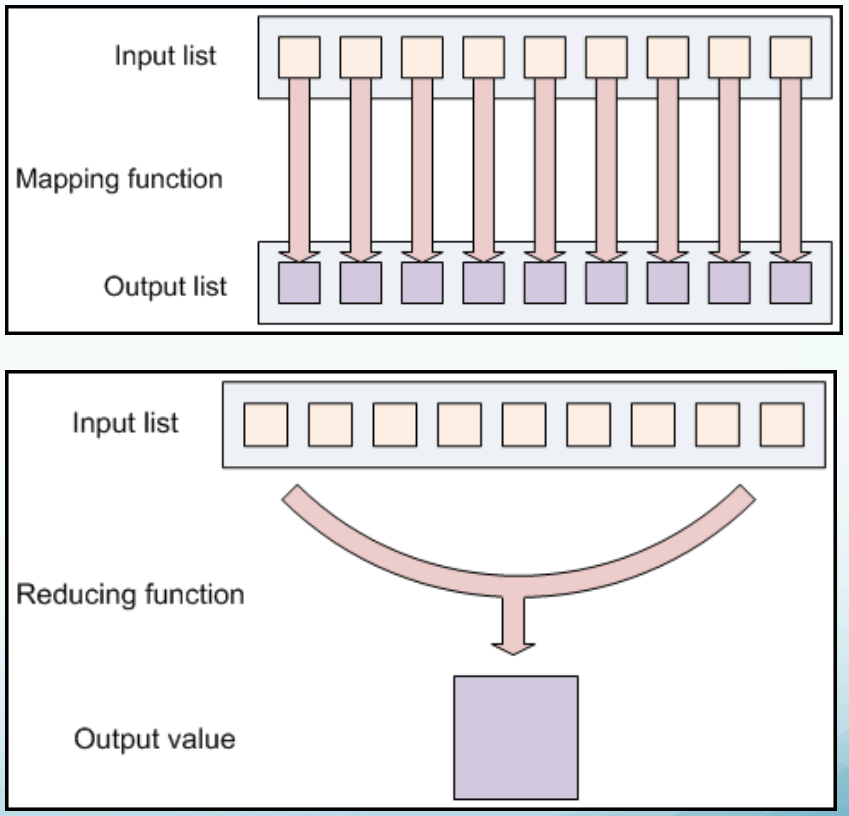
- The CapacityScheduler is designed to allow sharing a large cluster while giving each organization a minimum capacity guarantee

- Maximizing the throughput and the utilization of the cluster

Problem statement

- Job execution profile:
  - Average and Maximum Map execution time
    (M^i_{avg}, M^i_{max})
  - Average and Maximum Reduce execution time
    (R^i_{avg}, R^i_{max})
  - Average and Maximum Typical Shuffle execution time
    (Sh^i_{avg}, Sh^i_{max})
  - Average and Maximum First Shuffle execution time
    $(Sh^{1(J_i)}_{avg}, Sh^{1(J_i)}_{max})$

MODAClouds

Introduction: MapReduce

- Input: a **set** of **key/value pairs**
- User supplies two functions:
  - **map(k,v) → list(k1,v1)**
  - **reduce(k1, list(v1)) → v2**
- **(k1,v1)** is an **intermediate** key/value pair
- **Output** is the **set** of **(k1,v2)** pairs