BatchMR: Improving MapReduce Performance for the Combination of Diverse Workloads in Hybrid Clusters

Song-Lei Luo[1,2,3], Heng Wu[1,2], Guo-Quan Wu [1], Wen-Bo Zhang[1]
[1] Technology Center of Software Engineering, Institute of Software, the Chinese Academy of Sciences, Beijing 100090
[2] State key Lab, Institute of Software, Chinese Academy of Sciences, Beijing 100190
[3] University of Chinese Academy of Sciences, Beijing, 100049

Abstract. Nowadays, many giant companies produce lots of large-scale data of stable size from the combination of different production workloads every day, which are to be processed for business goals. One typical solution is to process these data in a hybrid cluster, but it's difficult to balance between resources utilization and performance. We introduce a new scheduler BatchMR to analyze the incoming jobs to arrange the allocation (cluster type and cluster size) based on a cost-benefit model. Evaluations on a hybrid cluster with diverse workloads consisting of many current standard benchmarks demonstrate that BatchMR can achieve up to almost twice improvement in the job completion times, over the original random arrangements.

1 Introduction

Virtualized environments are becoming more and more important due to their reliability, manageability and overall performance, resulting in the current pervasive hybrid clusters naturally (a mix of physical and virtual clusters). Though virtualization overheads have continued to decrease with the introduction of virtualization-aware hardware, there still exists severe performance degradation when batch jobs are executed in a hybrid cluster, clearly demonstrated in our experiments (Section 2). But if we process all the batch jobs in a pure physical cluster, its resources utilization will be strongly affected resulting in lots of resources wasted. The throughput[1] of the physical cluster will be dramatically reduced, also proved in our experiments (Section 2). The result is even worse than that of the virtual clusters.

To make better use of the physical and virtual clusters to improve MapReduce resources utilization and performance, enhancing the batch jobs processing ability as a whole, placement of the batch jobs in hybrid clusters should be properly arranged. However, latest work [2] [3] [4] [5] [6] is focusing on improving the Hadoop performance on a pure environment, either in virtual or physical environment. Or they all [7] [8] [9] try to optimize the speculative execution algorithm in hybrid clusters while ignoring the placement between physical and virtual clusters, which directly underestimates the extra virtualization overheads on virtual clusters.

[1] Throughput in this paper refers to the average amount of batch jobs finished in a certain amount of time.

Nowadays lots of giant companies are producing lots of large-scale data from different combinations of diverse workloads every day. To convert the data for better statistical analysis, they are pouring the data to a hybrid cluster and retrieve the statistical result to better their application design, spam and fraud detection capacity and advertising opportunities and the like. As mentioned above, lots of works are focusing on handling the generalized situation ignoring the plain fact that the data here is always similar and stable in size and data type. To better handle a situation like this, this paper proposes a new scheduler called BatchMR for large-scale similar batch jobs in hybrid clusters. Different types of job will be labelled as different batch job. BatchMR analyze the coming batch jobs according to the job type. Based on a cost-benefit model, BatchMR will automatically arrange a specific allocation for each job, related to cluster type (physical or virtual) and cluster size. The BatchMR is composed of offline-training and online-learning phases. Since the types of the large-scale data are limited and the sizes are stable every day, an offline-training can nicely depict the features of all the batch jobs. Based on the offline-training we can easily build up our cost-benefit model which can be abstracted to a three dimensional dynamic programming problem. As for the new data type, all we need to do is to do offline training on that single new data type while retaining all the previous training data sets. As for the online-learning process, the data of a company is stable but if we just build a single static model and stay unchanged, one day the data will be hugely different from the first original data sets resulting in bad depiction. In that case, the cost-benefit model can be inaccurate for the outdated underlying data sets. So it's quite essential to have an online-learning mechanism to update the underlying data sets properly for better accuracy in model prediction.

We make a thorough evaluation of BatchMR on a set of several standard benchmarks (e.g., Grep, Sort, Pi, Kmeans and WordCount) in a hybrid cluster. The job set is composed of different types of jobs with stable sizes emulating the job combination of diverse workloads generated in a company. The hybrid cluster in the experiment consists of 10 physical and 40 virtual machines using Xen 4.6 and Hadoop 2.7. The results of our evaluation are promising: BatchMR can achieve almost twice improvement in the throughput of MapReduce jobs than that of the random arrangement on a physical cluster. And to demonstrate the importance of the online-learning process, we also run another set of comparison experiments.

In this paper, we made the following five contributions. First, we discover the severe performance degradation and inefficient resources utilization in similar batch jobs processed in the hybrid clusters by means of local experiments. Second, we propose an effective scheduling algorithm for a stable combination of diverse workloads to properly arrange the allocation for the batch jobs in a hybrid cluster. Third, we provide two simple options to retrieve the most significant sample sizes for different jobs to optimize the modeling performance. Fourth, we conducted representative comparison tests to demonstrate the necessity of online-learning process in BatchMR. Finally, we conduct several evaluation experiments and demonstrate that BatchMR can achieve up to almost twice improvement in the throughput of MapReduce jobs than that of the random arrangements on a pure physical cluster.

2 Motivation

There are enormous difference between physical and virtual clusters in resources utilization and MapReduce performance which has been a problem years ago [3] [6] [7] [12]. In this section, we are to present some experiment results to explain some details about that difference and its corresponding causes. In order to clarify the importance of the placement between physical and virtual clusters for large-scale batch jobs, we conduct two independent contrast experiments including performance comparison and resources utilization comparison. We have learned so much from the previous works [3] [10] [11] about the various challenges, benefits and tradeoffs involved in deploying Hadoop MapReduce in a virtual cluster. But there is not too much about how to place properly between physical and virtual clusters. To begin with, we set up 4 VMs on 10 PMs each. Each PM has a quad-core 64-bit 3.4GHz Intel processors, 16GB RAM and 500GB SATA disk drives, connected with 1 Gbps Ethernet. Each VM is configured with 1 vCPU and 1024 MB Memory. As a whole, we have 10 Dom0s [26] and 30 DomUs to allocate.

2.1 Performance Comparison

To demonstrate that some batch jobs have special preference for physical environments, we conducted the following experiments. In these experiments, we try to process three sets of I/O bound batch jobs in pure physical clusters against pure virtual clusters, whose computation and I/O resources will both be controlled by *cgroup* and *tc*. Set1 includes the following three standard benchmarks:

- Sort, sorting 20GB text data, which is a typical I/O-intensive batch job;
- Grep, searching a random regular expression in 10GB of text data, which is a typical I/O-intensive batch job;
- WordCount, computing words frequency in 10GB of text data, which is a typical I/O-intensive batch job.

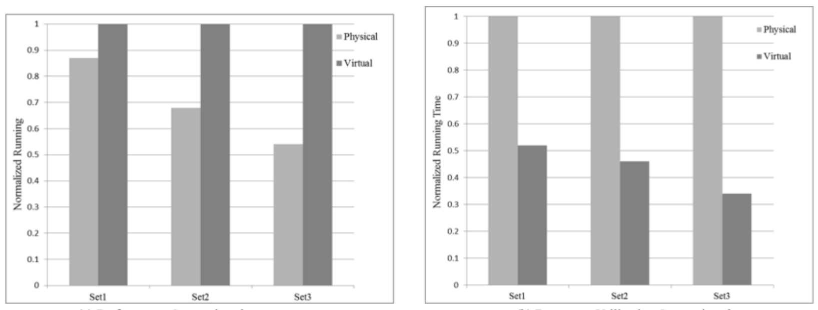
As for set2, the sizes are 40GB, 20GB and 20GB respectively, and set3 are 80GB, 40GB and 40GB respectively. Each set of batch jobs will be executed 3 times in virtual and physical environments, and we will take the average for accuracy. The Y-axis in figure 1 represents the normalized job completion times. It's clearly displayed that in set1 the performance in the physical cluster is about 75% better than that in the virtual cluster and as the size goes up, the performance gap is even larger; as for set3, it's about twice better. This demonstrates that some batch jobs will have special preference for the physical environments. In this case, physical clusters should be a better choice to ensure performance.

2.2 Resources Utilization Comparison

To better demonstrate that the virtual clusters will be more efficient in resources utilization, three sets of CPU bound batch jobs will be processed in a pure virtual cluster taking up all the 30 DomUs in the 10 physical machines against a pure physical cluster composed of 10 Dom0s in the same 10 physical machines. We have tried three different sets of batch jobs similarly. Set1 insists of:

- Kmeans, clustering 5GB data, which is a typical CPU-intensive batch job;
- Pi, estimating Pi (one hundred thousand points), which is a typical CPU-intensive batch job.

As for set2, the sizes are 10GB and 1 million. Set3 are 20GB and 1 billion respectively. The process is exactly the same as the previous series of experiments. It's clearly presented that the set1 processed in virtual cluster is about twice faster than that in physical and as the size goes up, as for set3, the difference is even about thrice faster. In physical clusters, if the resources is occupied by one single job then it cannot be reused, a typical resources waste which will never happen in virtual clusters where the mappers and reducers will have more CPU cycles and as a direct result more number of map and reduce tasks can be launched to utilize the available idle CPU.



(a) Performance Comparison between Physical and Virtual Clusters

(b) Resources Utilization Comparison between Physical and Virtual Clusters

Fig. 1. Comparisons in Performance and Resources Utilization

2.3 Problem statement

These experiments clarify it clearly that proper placement is non-trivial. If these jobs are always quite the same, the accumulative difference can then be huge. According to these experiments, virtualization overheads in virtual clusters can be ignored in some batch jobs, which can be used to promote resources utilization. Besides as Sharma [12] points out that when the network communication overheads become the main resource bottleneck, it will become the major factor that determines the overall performance. This again reminds us that even the batch jobs might have access to more computation resources, the whole performance might not as good as we expect. All in all, in a given hybrid cluster, it's essential to find out the optimal allocation arrangements (related to cluster type and cluster size) for each job in a set of workloads to achieve the optimal time cost as a whole.

3 BatchMR Design

We are trying to improve the Hadoop performance while improving the resources utilization in hybrid clusters. According to our experiments, it's non-trivial to select the proper cluster type for the batch job since some jobs might encounter severe degradation in virtual clusters but can be executed much better in physical. Besides, when the degradation is inherited by the latter similar workloads, the negative influence will be spread and magnified dramatically. As a straightforward result, we design a new scheduler called BatchMR focusing on scheduling for consecutive similar large-scale batch jobs in a given hybrid cluster. As the figure 2 shows, firstly BatchMR will classify the incoming batch jobs based on the cost-benefit model, retrieved from the Rule Engine. Secondly, to keep the Rule Engine updated all the time, the BatchMR will do online learning to update the modeling data sets accordingly.

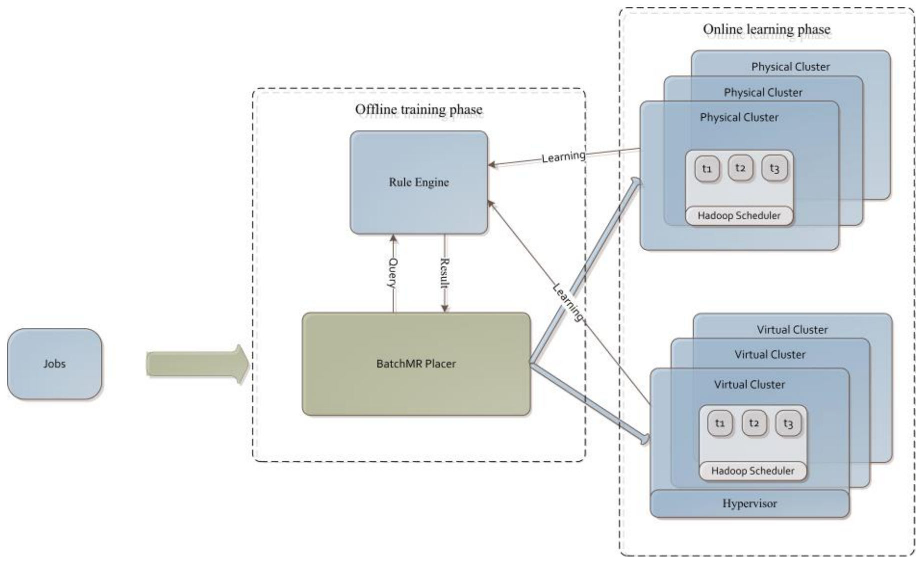


Fig. 2. Overview of BatchMR.

3.1 Offline training

BatchMR is to classify the batch jobs within the workloads and automatically dispatch the jobs to physical or virtual clusters in certain cluster size. Our objective in this phase is to make full use of physical and virtual clusters altogether trying to improve performance and resources utilization as a whole avoiding too much notorious virtualization overheads and resources waste.

Before Rule Engine can be actually utilized, we have to do offline training first. Since the types of jobs are limited, the objective of scheduling is to take less time to complete the set of workloads in a given hybrid cluster. Then the problem can be further converted to a dynamic programming problem. Since each batch job in different cluster sizes will take different time and the relationship between the job completion time and cluster size is negative correlation [12]; and also according to our previous experiments, the job completion time is quite different in different clusters (physical and virtual clusters). Each batch job in the workloads can be depicted as the table 1 and table 2 after training. The job completion time is determined by two major features: cluster size and cluster type. Each job is located either in physical clusters or virtual clusters and once the cluster size is specified, the completion time is determined.

Table 1. JCTs in Physical Cluster with Different Cluster Sizes

	$size_{p\ 0}$	$size_{p\ 1}$	$size_{p\ 2}$	$size_{p\ 4}$	$size_{p\ 5}$	…
Physical	$time_{p\ 0}$	$time_{p\ 1}$	$time_{p\ 2}$	$time_{p\ 4}$	$time_{p\ 5}$	…

Table 2. JCTs in Physical Cluster with Different Cluster Sizes

	$size_{v\ 0}$	$size_{v\ 1}$	$size_{v\ 2}$	$size_{v\ 4}$	$size_{v\ 5}$	…
Virtual	$time_{v\ 0}$	$time_{v\ 1}$	$time_{v\ 2}$	$time_{v\ 4}$	$time_{v\ 5}$	…

In a restricted environment, the actual problem naturally becomes this: we should select the best cluster type and cluster size combination for each batch job in the hybrid cluster to make the whole job completion times the least. Since each batch job performs differently in different cluster sizes and cluster types and if we directly turn to the dynamic programming technique trying each possible cluster size without any preceding pruning, the problem can be hard to solve within an acceptable time. The workloads are always similar and batch job types are predefined, so the ideal cluster size should be known before the training by working experience or systematic analysis [23]; as a straight-forward result, we could manually select the most representative cluster sizes for each batch job to reduce the scale of the problem. We should run the batch job in two different cluster types and in different cluster sizes and at last we will have two time cost tables for each job just as the table 1 and table 2 present above. And the problem finally is converted to retrieve the least time under the limited resources for a set of batch jobs within the workloads.

$$Time_{least} = \sum_{i=0}^{m}(time_{p\_i} + time_{v\_i}) \quad \sum_{i=0}^{m} size_{p\_i} <= P, \sum_{i=0}^{m} size_{v\_i} <= V \qquad (1)$$

$Time_{least}$ is the optimal time cost of the whole workloads; $time_{p\_i}$ and $time_{v\_i}$ is the time the job would take in the specified cluster size $size_{p\_i}$ and $size_{v\_i}$ respectively (only one of them is non-zero since in our algorithm the job is either placed in physical or virtual cluster); similarly only one of the cluster sizes ($size_{p\_i}$ or $size_{v\_i}$) is non-zero. Since the whole cluster size in physical and virtual cluster is limited, so the restraint is the sum of the cluster size. And the dynamic programming equation can then be abstracted as follows:

$$f(i,p,v) = \min\{\begin{matrix} \min(f(i-1, p-size_i, v) + time_{p\_i}), p >= size_i \\ \min(f(i-1, p, v-size_i) + time_{v\_i}), v >= size_i \end{matrix} \quad (2)$$

For task i in a given hybrid cluster composed of p physical machines and v virtual machines, the optimal time cost $f(i,p,v)$ is determined by its previous task and its own time cost table which is the result of our offline training. There will be some exceptions in the previous cost $f(i-1,p-size_i,v)$ and $f(i-1,p,v-size_i)$ where some values can be missing, in which case the cost will be INT_MAX to indicate that there is no such possibility for the previous stage to arrange such an allocation for all the previous i-1 jobs. As for the specific allocation for each task when retrieving the optimal time cost, we could utilize a hash table $path(i,p,v)$ to record the allocation $(cluster\_type, cluster\_size)$ for each task i and easily we can retrieve the allocation for each task backwards. To ensure the path hash table is complete, we will try to adopt a bottom-up method to construct the 3-dimension array $f(n,P,V)$. The whole time complexity will be $O(nm^3)$ where n is the number of the batch jobs in the workloads while there are at most m types of cluster size.

As for the detailed working mechanism, when a set of workloads arrives, BatchMR will send a query to the Rule Engine. Rule Engine will determine the cluster type and cluster size for each job according to the job IDs in the workloads, based on the time cost tables generated in the offline training process. BatchMR then can determine how to schedule the job using which cluster type and how many machines. In a word, in this phase, we are trying to arrange all the batch jobs in the workloads properly to ensure the global least time cost in the given hybrid cluster.

3.2 Online learning

Though the workloads are stable in a short time, nevertheless the change will always be there, not only about the batch job itself but the hardware environment. To handle the changes smoothly, we will update the task time cost tables once we finish each set of workloads. Since in a set of workloads, each batch job will be scheduled with different cluster types and cluster sizes and result in different completion times, we will utilize EWMA (exponentially weighted moving average) [23] to update the time cost tables properly to ensure better accuracy of Rule Engine.

$$Z(t) = \alpha * Y(t) + (1-\alpha) * Z(t-1), 0 < \alpha < 1 \quad (3)$$

$Z(t)$ is the JCT (Job Completion Times) to be used to update the time cost table, while $Y(t)$ is the new JCT of the batch job in the current workloads; α here indicates how much weight we should put on the new JCT; we will take 0.3 for α which is an ideal choice according to our experiments.

4 Evaluation

There will be 4VMs in all 10 PMs each and the detailed hardware information has been discussed before. Under this circumstance, we can easily adjust the number of machines in virtual and physical cluster to perform evaluation on BatchMR. There will be three sets of batch jobs composed of Sort, Kmeans, Grep, WordCount and Pi which are all the representative benchmarks in Hadoop. Set1 includes:

- Sort, sorting 10GB text data, which is a typical I/O-intensive batch job;
- Kmeans, clustering 5GB data, which is a typical CPU-intensive batch job;
- Grep, searching a random regular expression in 10GB of text data, which is a typical I/O-intensive batch job;
- WordCount, computing words frequency in 10GB of text data, which is a typical I/O-intensive batch job;
- Pi, estimating Pi (one hundred thousand points), which is a typical CPU-intensive batch job.

As for the set2, the sizes are 20GB, 10GB, 20GB, 20GB and 10 million respectively and the set3 are 40GB, 20GB, 40GB, 40GB and 1 billion respectively. These MapReduce jobs are chosen due to their popularity and representativeness of real MapReduce batch jobs.

Before the three sets of workloads are poured into the hybrid cluster, we first label them to identify different jobs and then do our offline training to construct the time cost tables. Since in our working environment, the virtual machines are more than the physical ones, so in our experiments the most representative cluster size is not the same. The following table is the time cost table for Sort in set1 (labelled as Sort1), selecting the most representative cluster sizes.

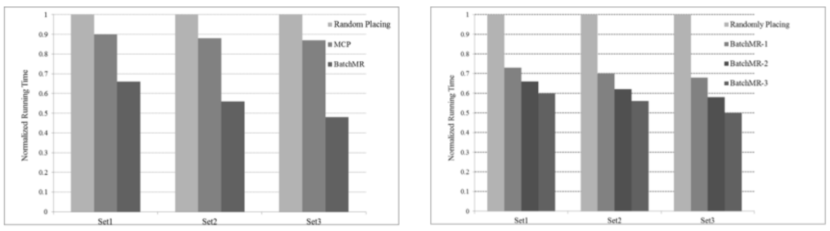
Table 3. Time Cost Table of Sort1 in Physical Cluster with Different Cluster Sizes

	3	4	5	6	7	8	9	10
P	42ms	39ms	30ms	30ms	28ms	26ms	28ms	30ms

Table 4. Time Cost Table of Sort1 in Virtual Cluster with Different Cluster Sizes

	6	8	10	12	14	16	18	20
V	51ms	48ms	44ms	43ms	48ms	48ms	52ms	50ms

After building up all the time cost tables for each job in set1, we first process the set1 without special scheduling and placing mechanism in the hybrid cluster, placing them randomly. Along with these pure random arrangements, we also conducted another experiment using MCP without special scheduling. Later on, we make the BatchMR to manage the placing arrangements according to the cost-benefit model based on the time cost tables in the same hybrid cluster with the same set of workloads, set1. The same set of workloads, set1 will be will be executed 3 times by random placing, MCP and BatchMR. We will take the average for better accuracy. To be more persuasive, we again run other similar tests in set2 and set3. The normalized running time is clearly presented in figure 3(a).



a. Comparison between Random Placing, MCP and BatchMR b. Comparison When Data Size Changes

Fig. 3. Comparison Experiments

Even compared with the MCP [9], the BatchMR still overpass its total time cost by about 80%, since the MCP is mainly focusing on handling the local optimal choice ignoring the overall influence of each allocation. It is clearly presented that as the size of the batch jobs goes up, the better performance the BatchMR presents (shown in figure 3(a)). This is due to the fact that larger jobs will run longer. If their placement is better, as the time moves on, the betterment will be displayed more distinctly. These experiments strongly exemplify that proper allocation for each job is quite essential to improve performance and resources utilization.

To further demonstrate the robustness of BatchMR, we design another set of testing sets which changes the set1, set2 and set3 about 10% in data size. Based on the original offline training, we will run these sets three times to check the stability of the BatchMR. As the figure 3(b) shows, the BatchMR is good enough to handle the fluctuation of data size by online learning. As the data size changes, gradually the BatchMR is coping with it more and more naturally.

5 Related Work

Yanpei Chen [19] even proposes bran-new workload suites for better MapReduce performance measurements. Sharma [12] also proposes a 2-phase scheduler considering the environments with interactive application reducing the interference from MapReduce jobs, while we are trying to improve the whole Hadoop performance and resources utilization in a hybrid cluster.

Actually speculative execution mechanism in MapReduce shares some ideas with "speculative execution" in distributed file systems [20], configuration management [21] which indeed inspires us much for further optimization in hybrid clusters. Mantri [8] proposes another speculative execution algorithm using the task's process bandwidth (processed data/time) to calculate the task's remaining time (data left/process bandwidth). The method will break down in the current Hadoop environments where pre-copy will speed up the copy phase while map tasks have not been finished and as a direct result, the bandwidth can fluctuate dramatically when reduce phase just finish its copying process. Chen Qi [9] proposes a more sophisticated speculative execution strategy named MCP to improve the Hadoop performance, but her strategy is only local optimal. There are also other methods to improve MapReduce performance: [16] trying to make better use of data-locality which is also a significant factor in our algorithm discussed earlier, [3] trying to adjust the whole MapReduce scheduling mechanism and structure to reduce network communication which is actually also trying to enhance the data-locality but its deficiency in scheduling is still obvious, besides it will not be easy to rebuild the whole MapReduce.

Compared to all the works mentioned above, our work is different considering the following three aspects. First, the target is quite specific: improve the performance (throughput) of MapReduce batch jobs and take better advantage of the current virtualization technology to improve resources utilization in a certain environment where are batch jobs are predefined and similar all the time. Second, the offline training and modeling process are simple and easy to be carried out, either by the working experience or the systematic tools [25], compared to the previous works [3] [9] [16]. Third, the scheduler BatchMR significantly improves the throughput of large-scale MapReduce batch jobs in a hybrid cluster while enhancing the resources utilization.

6 Conclusion

This paper presents a new scheduler BatchMR to handle the special situation where the workloads are always predefined and similar. BatchMR is used to improve the resources utilization and Hadoop performance in hybrid clusters, especially for the large-scale batch jobs. Before everything, there will be offline training to construct the time cost tables for all the predefined jobs in different cluster types and in different cluster sizes. Once a set of workloads comes, the BatchMR will analyze the batch jobs within; and according to a cost-benefit model, BatchMR will retrieve a set of allocations for each job from the Rule Engine and schedule them accordingly. To retain feasibility, BatchMR will utilize an online learning mechanism. Detailed evaluations on a hybrid cluster consisting of 10 physical and 40 virtual machines, with diverse workloads (a mixture of CPU and I/O bound batch jobs), demonstrate that BatchMR can achieve up to almost twice improvement in throughput of MapReduce batch jobs over the randomly scheduled arrangements on a pure physical cluster. And even when the workloads fluctuate, the BatchMR can also blend into the new situation by learning within 2 rounds.

References

[1] Dean J, Ghemawat S (2004) MapReduce: Simplified Data Processing on Large Clusters. In: Proc. OSDI - Symp. Oper. Syst. Des. Implement. USENIX, pp 137–149

[2] Park J, Lee D, Kim B, et al (2012) Locality-aware dynamic VM reconfiguration on MapReduce clouds. HPDC '12 Proc 21st Int Symp High-Performance Parallel Distrib Comput SE - HPDC '12 27–36. doi: doi: 10.1145/2287076.2287082

[3] Ibrahim S, Jin H, Cheng B, et al (2009) CLOUDLET: Towards MapReduce Implementation on Virtual Machines. Sci Technol 65–66. doi: 10.1145/1551609.1551624

[4] Cardosa M, Narang P, Chandra A, et al (2011) STEAMEngine: Driving MapReduce provisioning in the cloud. In: 2011 18th Int. Conf. High Perform. Comput. IEEE, pp 1–10.

[5] H. Herodotou and S. Babu. (2011) Profiling, what-if analysis, and cost-based optimization of mapreduce programs. PVLDB, 4(11):1111–1122.
[6] Verma A, Cherkasova L, Campbell R (2011) ARIA: Automatic Resource Inference and Allocation for MapReduce Environments. 8th ACM Int Conf Auton Comput 235–244. doi: 10.1145/1998582.1998637
[7] Zaharia M, Konwinski A, Joseph A, et al (2008) Improving MapReduce Performance in Heterogeneous Environments. Osdi 29–42. doi: 10.1109/IPDPSW.2010.5470880
[8] Ananthanarayanan G, Kandula S, Greenberg A, et al (2010) Reining in the Outliers in Map-Reduce Clusters using Mantri. Proc 9th USENIX Conf Oper Syst Des Implement 1–16.
[9] Chen Q, Liu C, Xiao Z (2014) Improving mapreduce performance using smart speculative execution strategy. IEEE Trans Comput 63:954–967. doi: 10.1109/TC.2013.15
[10] Chiang RC, Huang HH (2011) TRACON: Interference-aware scheduling for data-intensive applications in virtualized environments. 2011 Int Conf High Perform Comput Networking, Storage Anal 1–12. doi: 10.1109/TPDS.2013.82
[11] Bu X, Rao J, Xu C (2013) Interference and locality-aware task scheduling for MapReduce applications in virtual clusters. Proc 22nd Int Symp High-performance parallel Distrib Comput 227. doi: 10.1145/2493123.2462904
[12] Sharma B, Wood T, Das CR (2013) HybridMR: A hierarchical MapReduce scheduler for hybrid data centers. Proc - Int Conf Distrib Comput Syst 102–111. doi: 10.1109/ICDCS.2013.31
[13] Cardosa M, Narang P, Chandra A, et al (2011) STEAMEngine: Driving MapReduce provisioning in the cloud. In: 2011 18th Int. Conf. High Perform. Comput. IEEE, pp 1–10.
[14] H. Herodotou and S. Babu. (2011) Profiling, what-if analysis, and cost-based optimization of mapreduce programs. PVLDB, 4(11):1111–1122.
[15] Verma A, Cherkasova L, Campbell R (2011) ARIA: Automatic Resource Inference and Allocation for MapReduce Environments. 8th ACM Int Conf Auton Comput 235–244. doi: 10.1145/1998582.1998637
[16] Park J, Lee D, Kim B, et al (2012) Locality-aware dynamic VM reconfiguration on MapReduce clouds. HPDC '12 Proc 21st Int Symp High-Performance Parallel Distrib Comput SE - HPDC '12 27–36. doi: doi: 10.1145/2287076.2287082
[17] Ibrahim S (2009) Evaluating MapReduce on Virtual Machines : The Hadoop Case Evaluating MapReduce on Virtual Machines : The Hadoop Case *. doi: 10.1007/978-3-642-10665-1
[18] Kang H, Chen Y, Wong JL, et al (2011) Enhancement of Xen's scheduler for MapReduce workloads. Proc 20th Int Symp High Perform Distrib Comput - HPDC '11 251. doi: 10.1145/1996130.1996164
[19] Chen Y, Ganapathi A, Griffith R, Katz R (2011) The case for evaluating mapreduce performance using workload suites. IEEE Int Work Model Anal Simul Comput Telecommun Syst - Proc 390–399. doi: 10.1109/MASCOTS.2011.12
[20] Nightingale EB, Chen PM, Flinn J (2006) Speculative execution in a distributed file system[C]. Sosp 361–392. doi: 10.1145/1189256.1189258
[21] Su Y-Y, Attariyan M, Flinn J (2007) AutoBash : Improving configuration management with operating system causality analysis. Proc twenty-first ACM SIGOPS Symp Oper Syst Princ 237–250. doi: http://doi.acm.org/10.1145/1323293.1294284
[22] https://en.wikipedia.org/wiki/EWMA_chart
[23] Babu S (2010) Towards Automatic Optimization of MapReduce Programs. 0–5.
[24] Herodotou H (2011) MapReduce Programming and Cost-based Optimization ? Crossing this Chasm with Starfish. Proc. VLDB Endow. 4:
[25] http://www.cs.duke.edu/starfish/publications.html
[26] https://wiki.xen.org/wiki/Dom0