BatchMR: Improving MapReduce Performance and Resources Utilization in Heterogeneous Environments

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, lots of large-scale batch jobs are processed in heterogeneous environments, a mix of physical and virtual clusters, where resources utilization seems better but actually performance suffers from severe degradation. We introduce a new 2-phase scheduler BatchMR, trying to make better use of the advantages of hybrid clusters to improve resources utilization and Hadoop performance. BatchMR consists of a Placer and a Scheduler. First, BatchMR Placer classifies incoming MapReduce jobs to guide placement between physical and virtual clusters based on the virtualization overheads. Second, BatchMR Scheduler adjusts Hadoop speculative execution mechanism for workloads in heterogeneous environments. Evaluations on a hybrid cluster with diverse large-scale workloads consisting of many current standard benchmarks demonstrate that BatchMR can achieve up to almost twice improvement in the finish times, over original Hadoop on the pure physical cluster.

1 Introduction

Virtualized environments are becoming more and more important due to their reliability, manageability and overall performance. As a direct result, virtual machines either in public or private data centers have become a critical option, resulting in heterogeneous environments naturally, actually a hybrid of physical and virtual clusters. After systematic analysis accomplished by Google [1], it's reported that the virtualization overheads to be around 5% for computation and 15% for I/O workloads even when ignoring resources competition from other virtual machines within the same host. Though these virtualization overheads have continued to decrease with the introduction of virtualization-aware hardware, there exists severe performance degradation when batch jobs are executed in a heterogeneous environment, which is clearly demonstrated in our experiments. But if we process all the batch jobs in a pure physical cluster, its resources utilization will be strongly affected. The throughput[1] of the physical cluster will be dramatically reduced, also proved in our experiments. The result is even worse than that of processing all the jobs in the virtual clusters.

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

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 heterogeneous environments 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 heterogeneous environments while ignoring the placement between physical and virtual clusters, which directly underestimates the extra virtualization overheads on virtual clusters. To further improve resources utilization and performance, virtual clusters should be utilized as much as possible. However, when some jobs are much better in physical environments, physical clusters instead then should be used to meet their preference. If this placement is good enough, we can process more batch jobs, improving the parallel computing capability of the whole physical cluster.

Furthermore, in our experiments, we found out that the current speculative execution algorithm in Hadoop is not quite suitable for the heterogeneous environments which will cause severe performance degradation just as [7] points out. Speculative execution in Hadoop is to automatically back up the slow tasks (far slower than average and always called stragglers) on another machine to make the whole batch job finished faster. Without this automatic scheduling mechanism of speculative execution, a batch job would be as slow as the abnormal tasks. It's been proved that speculative execution can improve job response times (by 44% by Google [1]). But when it comes to heterogeneous environments, its performance is badly affected [7] [8] [9].

This paper proposes a 2-layer or 2-phase scheduler called BatchMR, composed of BatchMR Placer and BatchMR Scheduler, for large-scale MapReduce batch jobs in heterogeneous environments. In the first phase, BatchMR Placer classifies incoming MapReduce batch jobs based on the estimated virtualization overheads and automatically places the MapReduce batch jobs between physical and virtual clusters. In the second phase, BatchMR Scheduler adopts a new speculative execution algorithm optimizing the original homogeneity-based algorithm in Hadoop to schedule tasks within the cluster.

We also make a thorough evaluation of BatchMR on several standard benchmarks (e.g., Grep, Sort, Pi, Kmeans and WordCount) in a heterogeneous environment, which consists of 10 physical and 40 virtual machines using Xen and Hadoop with diverse workloads. The results of our evaluation are promising: BatchMR can achieve almost twice improvement in the throughput of MapReduce jobs than that of the original Hadoop on a pure physical cluster.

In this paper, we made the following four contributions. First, we discover the performance degradation and inefficient resources utilization in the heterogeneous environments by means of experiments. Second, we propose an effective classification mechanism to properly guide the placement of batch jobs in physical and virtual clusters. Third, we customize the Hadoop speculative execution algorithm to further improve Hadoop performance as well as resources utilization. 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 original Hadoop 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 separate contrast experiments including performance comparison and resources utilization comparison. There will be detailed explanation and related graphs to better explain the specifics and always there will be appropriate summaries to point out the critical points of the experiments on their purposes and conclusions. 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. We can then dynamically adjust the amount of the physical and virtual clusters depending on the different requirements of each experiment.

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.

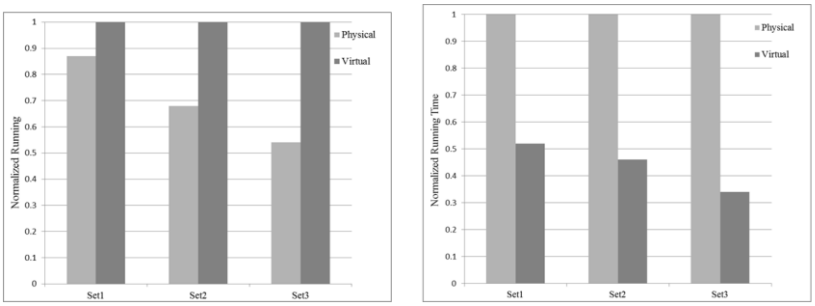
As for set2, the sizes are 40GB, 20GB and 20GB respectively, and set3 are 80GB, 40GB and 40GB respectively. Each set of batch jobs will be executed 3 times in virtual and physical environments, and we will take average for accuracy. The Y-axis in figure 1 represents the normalized running time. It's 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 even about twice better. This clearly exemplifies that the performance of I/O bound batch jobs in the physical environment can be much better than that in the virtual environment, which indirectly 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 against a pure physical cluster, whose computation and I/O resources will both be controlled by *cgroup* and *tc*. We have tried three different sets of batch jobs. 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 respectively. 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 average, the processing speed in virtual clusters is almost twice than that in pure physical cluster, which implies we can use the remaining time to process more jobs, plenty of resources wasted. 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 directly exemplify that proper placement is non-trivial especially for some batch jobs whose running time is quite different in different environments. According to these two sets of experiments, it's clearly demonstrated that virtualization overheads in virtual clusters can be ignored in some types of 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.

It's very important to classify the batch jobs and properly place them between physical and virtual clusters. Otherwise we might not only waste resources, but also not gain the performance we hoped.

3 BatchMR Design

We are trying to improve the Hadoop performance while improving the resources utilization in heterogeneous environments. According to our experiments, we think it's non-trivial to select the proper cluster for the right batch job since some jobs might encounter severe degradation in virtual clusters but can be executed much better in physical. Besides, according to pervious work [7] [8] [9], we found out that Hadoop will suffer from serious performance degradation in heterogeneous environments. As a straightforward result, we design a 2-phase scheduler called BatchMR focusing on scheduling for large-scale batch jobs in heterogeneous environments. This section mainly describes the overall architecture of BatchMR (shown in figure 2). In the first phase, BatchMR Placer will classify the incoming MapReduce batch jobs based on the estimated virtualization overheads. This estimation will be retrieved from the Rule Engine. However if the Rule Engine failed, the batch job will be put into the Sampling Area for a second estimation. Then according to the estimated overheads, BatchMR Placer will automatically guide the placement between physical and virtual clusters. In the second phase, the BatchMR Scheduler will adopt a more customized speculative execution algorithm to further improve Hadoop performance avoiding severe performance degradation in heterogeneous environments.

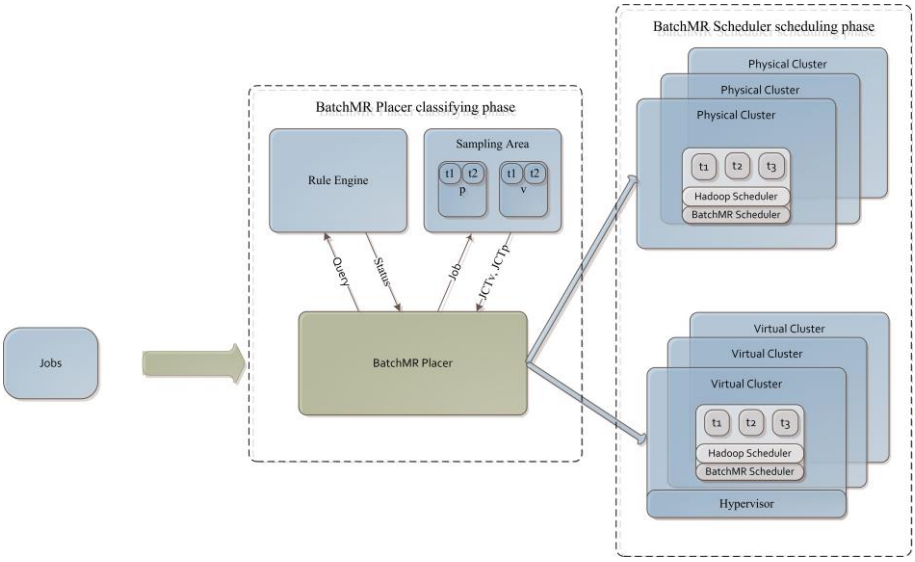


Fig. 2. Overview of BatchMR.

3.1 Job placement with BatchMR Placer

BatchMR Placer is to classify the workloads and automatically dispatch the jobs to physical or virtual clusters. Our objective in this phase is to make full use of physical and virtual clusters altogether trying to improve performance and resources utilization avoiding too much notorious virtualization overheads. Described clearly in the **Job Placement Algorithm** below, when a MapReduce batch job arrives, BatchMR will send a query to the Rule Engine to determine how to dispatch the job. If the returned value is P_CLUSTER then BatchMR will directly guide the job to the physical cluster; if the returned status is V_CLUSTER then BatchMR Placer will guide the job to the virtual cluster; if the returned status is UNKNOWN then the BatchMR will send the job to the Sampling Area for a second decision. As for the potential cost from BatchMR Placer, since we are processing large-scale batch jobs and we just do querying and sampling, so compared to the whole comparably much longer running time, its cost can be ignorable, which is also clearly proved in Section 4 Evaluation. In a word, in this phase, we are trying to make sure only the suitable batch jobs can be executed in a virtual cluster, instead of batch jobs which are far better in the physical cluster but far worse in the virtual cluster.

Job Placement Algorithm

Input: Jobs along with their data sizes and cluster sizes; Rule Engine will return the placement decision according to the cluster size and data size of a batch job.

1. **For** each job J_i in Jobs **do**
2. status = Retrieve(Rule Engine, $cluster\_size_i$, $data\_size_i$)
3. **if** status == P_CLUSTER **then**
4. Place Ji on physical cluster.
5. **else if** status == V_CLUSTER **then**
6. Place Ji on virtual cluster.
7. **else if** status == UNKNOWN **then**
8. $JCT_{diff\_ratio} = (JCT_{v\_sampling} - JCT_{p\_sampling}) / JCT_{p\_sampling}$
9. // $JCT_{v\_sampling}$ and $JCT_{p\_sampling}$ are retrieved from the sampling area
10. // where only representative sample of the job will be executed.
11. **end if**
12. **if** $JCT_{diff\_ratio} > threshold_{diff\_ratio}$ **then**
13. Place Ji on physical cluster.
14. **else then**
15. Place Ji on virtual cluster.
16. **end if**
17. **end for**

There are three thresholds in the Job Placement Algorithm: $threshold_{cluster\_size}$, $threshold_{data\_size}$ and $threshold_{diff\_ratio}$ which all are used to adjust the scheduler for corresponding working environments and workloads. After our evaluation on tens of different cluster sizes and different data sizes, we found out that when $threshold_{cluster\_size}$, $threshold_{data\_size}$ (unit is GB) and $threshold_{diff\_ratio}$ are 20%, 15% and

40% respectively, the overall throughput of the batch jobs reaches its top. How to determine these values will be further discussed later in the current section.

As for JCT estimation technique used in Rule Engine, there are many previous works [12] [13] [14] [15] for us to refer to. We will care about map-reduce running time, the data size and the cluster size. Both of data size and cluster size can be the most predominant factors in the whole process [12]. Apart from those values, we also need to record the cluster type for classification and timestamp for the **Rule Engine Classification Algorithm**.

Rule Engine Classification Algorithm

Input: Job along with its data size and cluster size; DB contains JCT (Job Completions Times), timestamp, cluster size, data size and cluster type (physical or virtual).

1. cluster_size_range = cluster_size *[1-threshold$_{cluster_size}$, 1+threshold$_{cluster_size}$]
2. data_size_range = data_size *[1-threshold$_{data_size}$, 1+threshold$_{data_size}$]
3. data_set = Retrieve(DB, cluster_size_range, data_size_range)
4. **if** data_set is empty **then**
5. Return UNKNOWN.
6. **else then**
7. data_set$_v$ = Retrieve(data_set, virtual)
8. data_set$_p$ = Retrieve(data_set, physical)
9. **if** data_set$_v$ is empty **or** data_set$_p$ is empty **then**
10. Return UNKNOWN.
11. **else then**
12. JCT_v = runTimestampBasedEWMA (data_set$_v$)
13. JCT_p = runTimestampBasedEWMA (data_set$_p$)
14. JCT_{diff} = (JCT_v-JCT_p)/JCT_p
15. **end if**
16. **If** JCT_{diff} > threshold$_{diff}$ **then**
17. Return P_CLUSTER.
18. **else then**
19. Return V_CLUSTER.
20. **end if**

To be more accurate in estimation, we will estimate the JCT (Job Completion Times) using EWMA (exponentially weighted moving average) [23].

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

EWMA can be expressed as the above formula (1) where Z(t) is the estimated JCT at time t while Y(t) is the observed JCT at time t (actually in our case Y(t) will be the average time of the data set which belongs to the same type – either virtual or physical). α here indicates how much weight we should put on observed JCT at time t and in BatchMR, we will take 0.3 for it which is determined by the result of our experiments. Now the Rule Engine can be built up by lots of batch jobs in different cluster sizes and data sizes. After training, when a job with certain cluster size and data size comes, the Rule Engine will return a classification result. The details will be as follows: first, Rule Engine will determine the cluster_size_range and

data_size_range according to $threshold_{cluster\_size}$ and $threshold_{data\_size}$ and then retrieve the data set from DB which will meet the condition (2).

$$cluster\_size \in cluster\_size\_range \;\&\&\; data\_size \in data\_size\_range \quad (2)$$

But if this retrieving operation failed and return an empty set, Rule Engine will just return UNKNOWN to indicate the status. If not, it will split up the data set into virtual and physical sub-sets based on the cluster type in the data set and again if one of the sub-sets is empty, Rule Engine will just return UNKNOWN; otherwise based on the sub-sets, Rule Engine will then calculate the estimated JCT on virtual and physical cluster respectively using the EWMA formula discussed above. At last, we will calculate the performance gap between these two estimated JCT. If it overpasses the $threshold_{diff}$, Rule Engine will return P_CLUSTER otherwise V_CLUSTER to guide the placement decision in BatchMR Placer.

But when the Rule Engine returns an UNKNOWN, the BatchMR Placer will have to throw the batch job to the Sampling Area where the representative part of the job will be started separately on a small cluster, physical and virtual respectively (displayed in figure 2). By comparing the estimated JCTs of the two instances of the job, we can easily estimate the performance gap between physical and virtual cluster. Similarly if the overhead is greater than $threshold_{diff}$, then the job is selected for the physical cluster, otherwise virtual cluster.

To actually determine these three thresholds mentioned above, we run lots of experiments on a mix of workloads in different cluster sizes and data sizes, when we are building up the Rule Engine. The Rule Engine will be trained by the offline batch jobs and further optimized by the jobs processed, a typical off-line training and on-line adjusting model.

Take $threshold_{cluster\_size}$ for an example, we will try 20 different values to train the Rule Engine including 0%, 5%, 10%, 15%, 20%, …, 100% to better evaluate its influence while the other two thresholds, $threshold_{diff\_ratio}$ and $threshold_{data\_size}$, will be set to 30% and 10% respectively according to our experience. During each value, we will try different pairs of cluster and data sizes including (3, 5GB), (4, 5GB) (5, 10GB), (6, 10GB), (7, 16GB), (8, 16GB), (9, 32GB) and (10, 32GB) as the training MapReduce batch jobs. To be more representative and accurate, we will run the current standard benchmarks including Grep, Sort, Kmeans and WordCount, which are all easy to control in data and cluster size. We will process the benchmarks mentioned above in different pairs of cluster and data size in physical and virtual cluster separately. In the end the record composed of JCT, cluster size, data size, timestamp and cluster type will be stored in the DB for Rule Engine. After we finish the training process collecting the essential data, we will run a random mix of workloads (different cluster size and data size but similar batch jobs used in training), all the jobs will be classified and scheduled by the BatchMR Placer with a selected $threshold_{cluster\_size}$ (0% as the first) and then we will get the completion times of the workloads. To ensure accuracy, we will run this workloads 3 times, classified and scheduled by the BatchMR Placer with a selected $threshold_{cluster\_size}$. At last we will get the average completion times and normalize it, using the time when the same workloads run in a pure physical cluster without extra classifying and scheduling. Then we will try another $threshold_{cluster\_size}$ (5% as the second) till the end 100% for the BatchMR Placer and the same workloads will be again processed three times (we will take the average JCT), classified and scheduled by the BatchMR Placer. But we

now do not need to re-run the same workloads on the pure physical cluster again, since we already have in our first experiment when the $threshold_{cluster\_size}$ is set to 0%. We will retrieve a normalized running time for each $threshold_{cluster\_size}$ from 0% to 100%.

The similar experiments will also be carried out on $threshold_{data\_size}$, but this time we will try different values for the $threshold_{data\_size}$ while fixing the $threshold_{diff\_ratio}$ and $threshold_{cluster\_size}$ to be 30% and 20% separately (based on the previous experiments, we believe these values are more efficient) to train the Rule Engine. After these experiments on $threshold_{cluster\_size}$ and $threshold_{data\_size}$, we try to search for the possible best $threshold_{diff\_ratio}$. We set the $threshold_{cluster\_size}$ and $threshold_{data\_size}$ to be 20% and 15% separately and we will try different $threshold_{diff\_ratio}$. The results of the experiments are displayed in figure 3. We can see that 30% for $threshold_{diff\_ratio}$ that we previously chose is not too bad proved now by the experiments, though not the best.

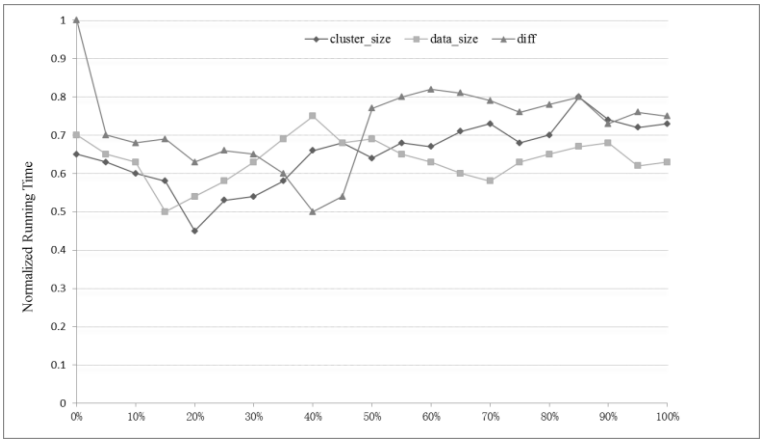


Fig. 3. Thresholds in different values affects JCT

In figure 3, clearly we can see that when the $threshold_{cluster\_size}$ is 0% the performance is barely acceptable and at 20% the performance reaches the best, which is actually due to the fact that when the $threshold_{cluster\_size}$ is 0%~10%, the Rule Engine either will have a very small set of records to estimate or it will just miss it in which case BatchMR Placer will just throw it to the Sampling Area for a second estimation resulting extra calculation overheads. As for the 20% where the BatchMR performs the best according to the test cases we have run, compared to other higher thresholds, Because this will constrain the range of the batch jobs that have been processed, which should be more representative to predict the coming batch job. As we can see, that after 40% the prediction becomes unstable and the result is quite bad even worse than that of 0% sometimes due to their comparably unlimited range of samples.

The experiments we run on $threshold_{data\_size}$ show us a similar result. When the $threshold_{data\_size}$ is set to be 0%, the performance is also hardly acceptable, caused by similar reasons we discussed above; about 15%, the performance hits its top in our experiments; and about 40%, the performance reaches its bottom, worst situation and then fluctuates around it as the threshold becomes higher, which should be caused by

the size of the records used in prediction. This phenomenon could be the result of the training data sets, which can be further studied. However in this paper, we only care about the most efficient hot spot.

When it comes to $threshold_{diff\_ratio}$, we can clearly pick out that when it is set as 0%, the performance is the worst, same as that in pure physical machines. As we anticipated, all the batch jobs will be directly guided to the physical clusters due to this threshold. Actually, there are few errors at times when the JCT prediction in the virtual cluster is higher than that in the physical cluster. When this happens, the batch job will be guided to the virtual cluster ignoring the $threshold_{diff\_ratio}$. However, as far as we know, this kind of error is quite rare (only when the data set is very small and the JCTs are very close). Furthermore, the comparably much higher JCT of large-scale workloads makes this side effect even more trivial. Consequently, the performance is quite the same as that in the physical cluster. When the $threshold_{diff\_ratio}$ is set to be 100%, the performance is not the worst but belongs to the worst area where almost all the batch jobs are placed to the virtual clusters, which will incur worst performance for some batch jobs due to their special preference for physical clusters.

3.2 Speculative execution with BatchMR Scheduler

Apart from the classification detailedly described in the previous part, there exists severe performance degradation due to the inappropriate speculative execution mechanism [7] [8] [9] in heterogeneous environments. To resolve this performance degradation problem, we have customized the Hadoop speculative execution algorithm for heterogeneous environments. The primary thought behind our algorithm is as follows: unlike the previous work concentrating the whole performance on the speculative execution and considering every possible aspect, we are trying to handle the most severe parts that directly cause the performance degradation most.

First we will use a new method to predict the process speed which will be better than the progress rate [7] and the process bandwidth [8]. Progress rate used in LATE [7] can be easily broken down. First, if different phases of the map or reduce varies, the fixed ratios it uses to rate will fail. Second, if there is severe data skewness, which is quite common for large-scale batch jobs processed in heterogeneous environments, it will become meaningless to compare different tasks by rate. As for process bandwidth, it will also not accurate when the reduce task just starts up, whose speed can be much higher considering its pre-copy operations. To more accurately predict the processing speed to locate the real stragglers, a more advanced and robust method has to be adopted and as a result, we are trying to make use of EWMA whose features (accuracy and effectiveness) are also proved in [9].

Second we will determine which task to back up based on the most beneficial principle. The remaining time for the current task can be easily retrieved according to the following formula (3).

$$time_{remain} = \frac{data_{left}}{speed_{cur}} + \frac{data_{cur}}{data_{avg}} \sum_{p=cur+1}^{end} \frac{data_{avg}}{speed_{p.avg}} \tag{3}$$

$data_{left}$ is the data left to be processed in the current phase; $speed_{cur}$ is the data processing speed in the current phase of the current task (predicted based on the method discussed above); $data_{cur}$ is the data size of the current task; $data_{avg}$ is the average data size of all the processed and processing tasks till now (since some tasks might not be started); $speed_{p.avg}$ is the average speed for all tasks in the phase *p*. We only need to accumulate the time starting from the next phase till the end for the current task, since the time for the current phase can be calculated by the first part. One might wonder if there are some phases that have not happened yet, then the result will not be right. However, what we really need is to compare the remaining time of each task, these missing phases won't be a problem since all of them will be ignored for all tasks. Based on the process speed discussed before, we can accurately identify the stragglers, but the most beneficial backup should be the one from which we can gain the most (the longest time we possibly can save from this backup) so we have to accurately estimate the time the backup will take. There is a major factor that affects the benefits we can truly gain from backup is data-locality which has been an issue for years [3] [9] [11] [16]. To reduce this influence and maintain the simplicity, we try to classify the data-locality under several different levels which will contribute different time weight for the tasks completion time (e.g. 1t, 2t, 3t).

$$time_{backup} = \frac{locality\_type_{backup}}{locality\_type_{cur}} * \frac{data_{cur}}{data_{avg}} \sum_{p=start}^{end} \frac{data_{avg}}{speed_{p.avg}} \quad (4)$$

And in our evaluations, we find this simple formula (4) just works fine and generate pretty good results. Then the time we can gain from the backup can be simply retrieved as formula (5).

$$time_{gain} = time_{remain} - time_{backup} \quad (5)$$

Third, there are some details we need to care about to maintain accuracy and efficiency when executing the algorithm. First, we will make sure a task has been processed for a certain amount of time (only when it's stable we can accurately predict). Second, after sorting the stragglers by the $time_{gain}$, we will also need to control the amount of tasks to back up to ensure fairness for other jobs in the same heterogeneous environments. So there will be a *speculativeCap* [7] to limit the maximal amount of tasks that can be backed up concurrently. More details can be provided, but to make this paper terse enough, most of them have been removed.

4 Evaluation

4.1 Evaluation on Job Placement

Our experiment environments are quite limited as we have mentioned before there will be 4VMs in all 10 PMs each. 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. We retain the training results in Rule Engine to save time. Take set1 for an example, first we run the set1 without special placing mechanism in the physical cluster, just placing them randomly (first-come-first-served discipline) using the original Hadoop. Later on, we enable the BatchMR Placer and also use the original Hadoop, and run set1 in the hybrid cluster. Each of them, random placing and BatchMR Placer placing, will be executed 3 times in each set including set1, set2 and set3. We will take the average to ensure better accuracy. The normalized running time is clearly presented in figure 4.

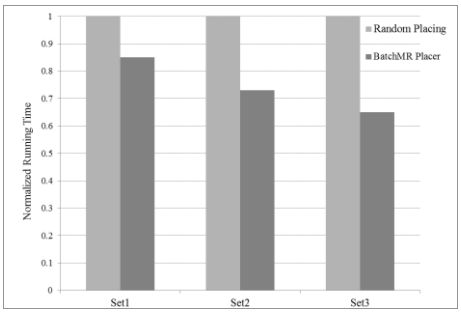


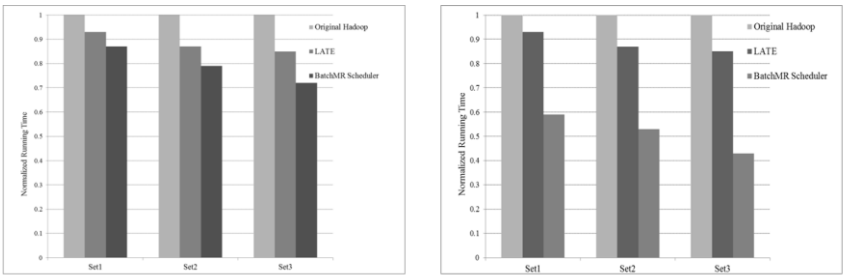
Fig. 4. Performance comparison with BatchMR enabled and disabled

It is clearly presented that as the size of the batch jobs goes up, the better performance the BatchMR presents (shown in figure 4). 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 placement is quite essential to improve performance and resources utilization.

4.2 Evaluation on customized speculative execution algorithm

The whole experiment environment and test cases are not changed since previous evaluation in BatchMR Placer. We first disable the BatchMR Placer and run our experiments in each set three times in the hybrid cluster and take the average for

accuracy. Also we will only try the original Hadoop and Hadoop-LATE in these three data sets in the hybrid cluster for comparison.



(a) With BatchMR Placer disabled (b) With BatchMR Placer enabled

Fig. 5. Comparisons among BatchMR and previous works

It's clearly shown in figure 5 (a) that our customized speculative execution is the best due to its accuracy to search out the stragglers, its underlying mechanism to back up the proper tasks, its optimization in data locality and the most beneficial principle and so on. The same sets of experiments have been done to check the performance of BatchMR with Placer enabled. We can see it clearly in figure 5 (b) that even not all similar batch jobs have been trained, its overall throughput is still much higher than the original Hadoop and the Hadoop-LATE. As the size of the batch jobs goes up, the improvements are becoming much more evident. The reason for this dramatic improvement can be quite straightforward, since the other two just consider Hadoop scheduling algorithm without any consideration to the placement, while BatchMR places the batch jobs according to the features of the jobs themselves. It's clearly demonstrated that BatchMR can achieve up to almost twice improvement in the throughput of MapReduce jobs than that of original Hadoop on a pure physical cluster.

5 Related Work

Cardosa [13] proposes techniques for MapReduce provisioning in virtual cloud clusters, with emphasis on energy reduction. Preliminary evaluations of Hadoop MapReduce performance on virtualized clusters is done in [17]. 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 heterogeneous environment.

Actually speculative execution mechanism in MapReduce shares some ideas with "speculative execution" in distributed file systems [20], configuration management [21], and information gathering [22], which indeed inspires us much for further optimization in heterogeneous environments. Zaharia [7] provides LATE, based on speed rate which actually is not accurate when it comes to the nowadays variable workloads in heterogeneous environments. 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 far complicated and really hard to carry out considering her unstable benefit-cost model which is also based on lots of assumptions. 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. Second, the whole structure and calculation process is simple, direct and easy to be carried out compared to the previous works [3] [9] [16]. Third, our 2-phase scheduler, BatchMR, significantly improves the throughput of large-scale MapReduce batch jobs in a heterogeneous environment while enhancing the resources utilization.

6 Conclusion

This paper presents a 2-phase hierarchical scheduler, called BatchMR, to improve the resources utilization and Hadoop performance in heterogeneous environments, especially for the large-scale batch jobs. In the first phase, BatchMR Placer classifies the incoming MapReduce batch jobs based on the estimated virtualization overheads. In the second phase, BatchMR Scheduler adopts a more customized Hadoop speculative execution algorithm for the heterogeneous environments. 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 original Hadoop on a pure physical cluster.

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] Barish G, Knoblock CA (2008) Speculative plan execution for information gathering. Artif Intell 172:413–453. doi: 10.1016/j.artint.2007.08.002
[23] https://en.wikipedia.org/wiki/EWMA_chart