Noname manuscript No.
(will be inserted by the editor)

Guidelines for Selecting Hadoop Schedulers based on System Heterogeneity

Aysan Rasooli · **Douglas G. Down**

Received: date / Accepted: date

Abstract Hadoop has been developed as a solution for performing large-scale data-parallel applications in Cloud computing. A Hadoop system can be described based on three factors: cluster, workload, and user. Each factor is either heterogeneous or homogeneous, which reflects the heterogeneity level of the Hadoop system. This paper studies the effect of heterogeneity in each of these factors on the performance of Hadoop schedulers. Three schedulers which consider different levels of Hadoop heterogeneity are used for the analysis: FIFO, Fair sharing, and COSHH (Classification and Optimization based Scheduler for Heterogeneous Hadoop). Performance issues are introduced for Hadoop schedulers, and experiments are provided to evaluate these issues. The reported results suggest guidelines for selecting an appropriate scheduler for Hadoop systems. Finally, the proposed guidelines are evaluated in different Hadoop systems.

Keywords Hadoop System · Scheduling System · Heterogeneous Hadoop

1 Introduction

Hadoop [20] is a data-intensive cluster computing system, in which incoming jobs are defined based on the MapReduce [1] programming model. MapReduce is a popular paradigm for performing computations on BigData in Cloud computing systems [26]. A Hadoop system consists of a cluster, which is a group of linked resources. Organizations could use existing resources to build Hadoop clusters - small companies may use their available (heterogeneous) resources to build a Hadoop cluster, or a large company may specify a number of (homogeneous) resources for setting up its Hadoop cluster. There can be a variety of users in a Hadoop system who are differentiated based on features such as priority, usage, guaranteed shares, etc. Similarly,

A. Rasooli, D. G. Down
Department of Computing and Software
McMaster University, L8S 4K1, Hamilton, Canada
E-mail: {rasooa, downd}@mcmaster.ca

workload in the Hadoop system may have differing numbers of users' jobs and corresponding requirements. Therefore, a Hadoop system can be specified using three main factors: cluster, workload, and user, where each can be either heterogeneous or homogeneous.

There is a growing demand to use Hadoop for various applications [2], which leads to sharing a Hadoop cluster between multiple users. To increase the utilization of a Hadoop cluster, different types of applications may be assigned to one cluster, which leads to increasing the heterogeneity level of workload. However, there are situations where a company assigns a Hadoop cluster to specific jobs as the jobs are critical, confidential, or highly data or computation intensive. Accordingly, the types of applications assigned by different users to a Hadoop cluster define the heterogeneity level of workload and users in the corresponding Hadoop system. Similarly, the types of resources define the heterogeneity of Hadoop clusters.

Schedulers play a critical role in achieving desired performance levels in a Hadoop system. However, the heterogeneity level of each factor potentially has a significant effect on performance. It is critical to select a scheduling algorithm by considering the Hadoop factors, and the desired performance level. A scheduling algorithm which performs well in one Hadoop system, may not work well for a system that differs in these factors. In [3], the authors introduced a hybrid method for selecting scheduling algorithms based on the scalability level of the Hadoop system. This study provides guidelines for selecting scheduling algorithms based on Hadoop factors, and their heterogeneity level.

In this paper, we first provide a specification for the Hadoop factors and their possible settings (Section 2). Then, performance issues for Hadoop schedulers are introduced (Section 3). The performed analysis and proposed guidelines are based on three Hadoop schedulers: FIFO, Fair Sharing [4], and COSHH [27]. The FIFO and Fair Sharing algorithms are the two best known and most widely used Hadoop schedulers. FIFO does not take into account heterogeneity in any Hadoop factor, while Fair Sharing considers user heterogeneity. The COSHH algorithm (introduced by the authors of this article), is a Hadoop scheduler which considers the heterogeneity in all three Hadoop factors.

To reduce the dimensionality of the space of heterogeneity factors, this paper performs a categorization of Hadoop systems, and experimentally analyzes the schedulers' performance in each category (Section 4). Sections 5 and 6 provide the experimental results and analysis for homogeneous and heterogeneous Hadoop environments, respectively. Finally, using the experiments and discussions, Hadoop scheduler selection guidelines are proposed in Section 7. The guidelines are evaluated using different Hadoop systems. Related work is discussed in Section 8, and Section 9 provides a conclusion.

2 Hadoop System

Heterogeneity in Hadoop is defined based on the level of heterogeneity in the following Hadoop factors:

- *Cluster:* is a group of linked resources, where each resource (R_j) has a computation unit and a data storage unit. The computation unit consists of a set of slots, where each slot has a given execution rate. In most Hadoop systems, each CPU core is considered as one slot. Similarly, the data storage unit has a given capacity and data retrieval rate. Data in the Hadoop system is organized into files, which are usually large. Each file is split into small pieces, which are called slices. Usually, all slices in a system have the same size.

- *User:* submits jobs to the system. Hadoop assigns a priority and a minimum share to each user based on a particular policy (e.g. the pricing policy in [6]). The user's minimum share is the minimum number of slots guaranteed for the user at each point in time.

- *Workload:* consists of a set of jobs, where each job (J_i) has a number of *map tasks* and *reduce tasks*. A *map task* performs a process on the slice where the required data for this task is located. A *reduce task* processes the results of a subset of a job's *map tasks*. The value $m(J_i, R_j)$ defines the mean execution time of job J_i on resource R_j. Investigations on real Hadoop workloads show that it is possible to classify these workloads into classes of "common jobs" [7]. We define the class of jobs to be the set of jobs whose mean execution times (on each resource) are in the same range.

There are various Hadoop schedulers, where each scheduler may consider different levels of heterogeneity in making scheduling decisions. Moreover, schedulers are differentiated based on different performance metrics (e.g., fairness, minimum share satisfaction, locality, and average completion time) that they address. However, to the best of our knowledge there is no scheduling algorithm which simultaneously considers all of these performance metrics. In some cases, optimizing one metric can result in significant degradation in another metric. For instance, a scheduler which optimizes fairness may need to repeatedly switch the processor between different jobs. This can add significant overhead, which can result in larger average completion times.

To analyze the behaviour of schedulers at different levels of heterogeneity, this paper uses three Hadoop scheduling algorithms: FIFO, Fair Sharing, and COSHH. The FIFO and Fair Sharing algorithms are used as the basis of a majority of Hadoop schedulers [4, 6, 8, 9]. The COSHH algorithm was first introduced in [5], and considers system parameters and state information in making scheduling decisions. These algorithms are selected as representatives of schedulers which consider heterogeneity at different levels. The FIFO scheduler does not consider heterogeneity in its scheduling decisions. However, the Fair Sharing and COSHH algorithms are representatives of schedulers with partial, and full consideration of heterogeneity, respectively.

- *FIFO:* is the default Hadoop scheduling algorithm [1]. It orders the jobs in a queue based on their arrival times, ignoring any heterogeneity in the system. The experience from deploying Hadoop in large systems shows simple algorithms like FIFO can cause severe performance degradation; particularly in systems that share data among multiple users [4].

– *Fair Sharing:* is a Hadoop scheduler introduced to address the shortcomings of FIFO, when dealing with small jobs and user heterogeneity [4]. This scheduler defines a pool for each user, where each pool has a number of map and reduce slots on a resource. Each user can use its pool to execute her jobs. If a pool of a user becomes idle, the slots of the pool are divided among other users. This scheduler aims to assign a fair share to users, which means resources are assigned to jobs such that all users receive, on average, an equal share of resources over time. Therefore, the Fair Sharing algorithm only takes user heterogeneity into account.

– *COSHH*[1]*:* is a Hadoop scheduler which considers cluster, workload, and user heterogeneity in making scheduling decisions [5]. Using the parameters and state information, COSHH classifies the jobs and finds a matching of the resulting job classes to the resources based on the requirements of the job classes and features of the resources. This algorithm solves a Linear Programming problem (LP) to find an appropriate matching. At the time of a scheduling decision, the COSHH algorithm uses the set of suggested job classes for each resource, and considers the priority, required minimum share, and fair share of users to make a scheduling decision. Therefore, this algorithm takes into account heterogeneity in all three Hadoop factors that we have introduced.

3 Performance Issues

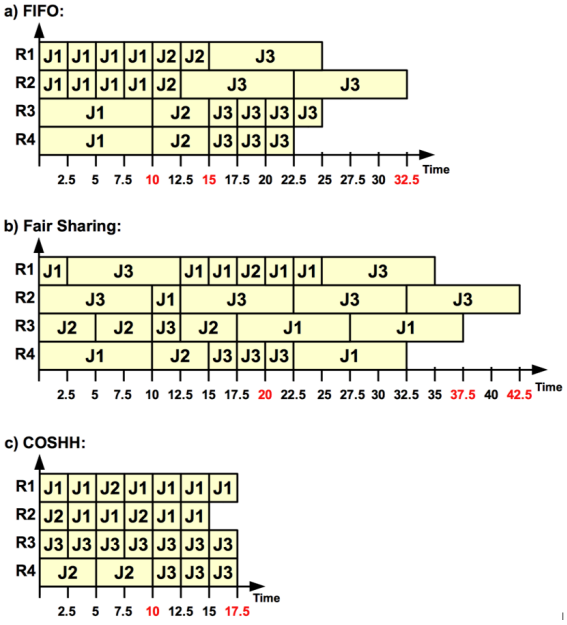
This section analyzes the main drawbacks of each scheduler for various heterogeneity levels in the Hadoop factors.

1. Small Jobs Starvation: Let us assume a heterogeneous Hadoop system, which will be referred as *System A* includes four different resources and three users with the following characteristics. The choice of system sizes in this system is only for ease of presentation, the same issues arise in larger systems.
   – Task1, Task2, and Task3 represent three heterogeneous task types with the following mean execution times. The execution time of each job on each resource depends on various factors such as the computation and storage features of the resource and the job's data and computation requirements. Here, $m_t(T_i,R_j)$ is the execution time of task T_i on resource R_j.
   $$m_t = \begin{bmatrix} 2.5 & 2.5 & 10 & 10 \\ 2.5 & 2.5 & 5 & 5 \\ 10 & 10 & 2.5 & 2.5 \end{bmatrix}$$
   – Three users submit three jobs to the system, where each job consists of a number of similar tasks. Jobs arrive to the system in the order: Job1, Job2, and Job3.
   – Users are homogeneous with zero minimum share and priority equal to one. Each user submits one job to the system as follows:

[1] While the name COSHH was not introduced in [5], we have since adopted it.

User1: Job1 (consists of 10 Task1)
User2: Job3 (consists of 10 Task3)
User3: Job2 (consists of 5 Task2)

Figure 1 shows the job assignments for the FIFO, Fair Sharing, and COSHH schedulers. The completion time of the last task in each job is highlighted to show the overall job completion times.



Scheduler	Job	Completion Time	Average Completion Time
FIFO	J1 J2 J3	10 15 32.5	19.17
Fair Sharing	J1 J2 J3	37.5 20 42.5	33.33
COSHH	J1 J2 J3	17.5 10 17.5	15

Fig. 1: Job assignment by a) FIFO, b) Fair Sharing, and c) COSHH schedulers, and their average completion times in *System A*.

The FIFO algorithm assigns incoming jobs to the resources based on their arrival times (Figure 1a). Consequently in the FIFO scheduler, execution of the smaller job (Job2) will be delayed significantly. In a heterogeneous Hadoop workload, jobs have different execution times. For such workloads, as the FIFO algorithm

does not take into account job sizes, it has the problem that small jobs potentially get stuck behind large ones.
The Fair Sharing and the COSHH algorithms do not have this problem. Fair Sharing puts the jobs in different pools based on their sizes, and assigns a fair share to each pool. As a result, the Fair Sharing algorithm executes different size jobs in parallel. The COSHH algorithm assigns the jobs to resources based on the job sizes and the execution rates of resources. As a result, it can avoid this problem.

2. Sticky Slots: Figure 1b shows the job-resource assignment for the Fair Sharing algorithm in *System A*. As the users are homogeneous, the Fair Sharing scheduler goes through all of the users' pools, and assigns a slot to one user at each heartbeat. Upon completion of a task, the free slot is assigned to a new task of the same user to preserve fairness among users.
Resource2 is an inefficient choice for Job3 with respect to completion time, but the Fair Sharing scheduler assigns this job to this resource multiple times. There is a similar problem for Job1 assigned to Resource3 and Resource4. Consequently, the average completion time will be increased.
This problem arises when the scheduler assigns a job to the same resource at each heartbeat. This issue is first mentioned in [4] for the Fair Sharing algorithm, where the authors considered the effect of this problem on locality. However, our example shows Sticky Slots can also significantly increase average completion times, when an inefficient resource is selected for a job.
The FIFO algorithm does not have this problem because it only considers arrival times in making scheduling decisions. The COSHH algorithm has two levels of classification, which avoids the Sticky Slot problem. Even when the same resource is assigned for a job in different rounds, the optimizations in the COSHH algorithm guarantee an appropriate selection of resource for the corresponding job.

3. Resource and Job Mismatch: In a heterogeneous Hadoop system, resources can have different features with respect to their computation or storage units. Moreover, jobs in a heterogeneous workload have different requirements. To reduce the average completion time, it is critical to assign the jobs to resources by considering resource features and job requirements.
The FIFO and the Fair Sharing algorithms both have the problem of resource and job mismatch, as they do not consider heterogeneity in the scheduling. On the other hand, the COSHH algorithm has the advantage of appropriate matching of jobs and resources, applying the following process (Figure 1c). The COSHH algorithm classifies the jobs into three classes: Class1, Class2, and Class3, which contain Job1, Job2, and Job3, respectively. This scheduler solves an LP to find the best set of suggested job classes for each resource, as follows.

Resource1:	{Class1, Class2}
Resource2:	{Class1, Class2}
Resource3:	{Class2, Class3}
Resource4:	{Class2, Class3}

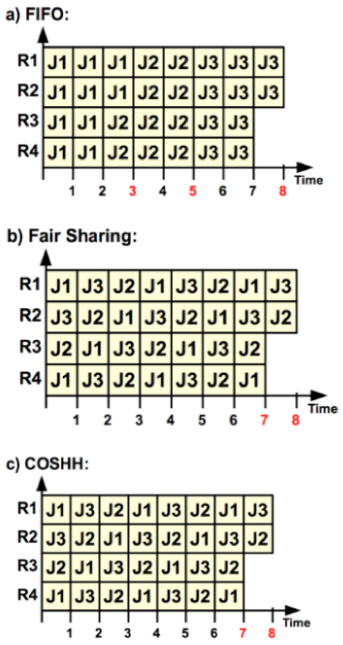
After computing the suggested sets, the COSHH scheduler considers fairness and minimum share satisfaction to assign a job to each resource. Although the COSHH algorithm assigns Job1 exclusively to Resource1 (Sticky Slot Problem), it does not increase the completion time of Job1. This is one of the main advantages of the COSHH algorithm over FIFO and Fair Sharing in a heterogeneous system.

4. Scheduling Complexity: Let us assume a homogeneous Hadoop system, which will be referred as *System B* includes four homogeneous resources and three users with the following characteristics.
   - There is one task type, and one job class, where each job consists of 10 tasks. Tasks are homogeneous, and they have mean execution time of 1 second on all resources.
   - There are three homogeneous users similar to *System A*.
   - Users submit three jobs to the system, where each job consists of a number of similar tasks. Jobs arrive to the system in this order: Job1, Job2, and Job3.

   The scheduling and completion times for the schedulers are presented in Figure 2. Figures 2a, 2b, and 2c show the job assignments in the FIFO, Fair Sharing, and COSHH schedulers, respectively.
   In a fully homogeneous Hadoop system, a simple algorithm like FIFO, which quickly multiplexes jobs to resources, leads to the best average completion time compared to the other, more complex algorithms (Figure 2a). As the users are homogeneous, at each heartbeat the Fair Sharing algorithm assigns a task of a user to the current free resource (Figure 2b). In the case of the Fair Sharing scheduler, at each heartbeat there is the overhead of comparing previous usage of resources to make a fair assignment. Finally, for the COSHH scheduler, as the system is homogeneous, and there is only one job class (i.e., jobs have similar features), solving the LP suggests all job classes for all resources. In this system, the scheduling decisions of the COSHH algorithm are identical to those of the Fair Sharing algorithm (Figure 2c). However, its scheduling complexity is greater than the Fair Sharing algorithm. The complexity of scheduling algorithms can result from different features, such as gathering more system parameters and state information, and considering various factors in making scheduling decisions. In a homogeneous system such as *System B*, a simple and fast algorithm like FIFO can achieve better performance.

4 Evaluation: Settings

Reported analysis on several Hadoop systems found their workloads extremely heterogeneous with very different execution times [10]. However, due to privacy issues, most companies are unable or unwilling to release information about their Hadoop systems. Therefore, reported information about heterogeneity in Hadoop may be only partial. This paper considers various possible settings of heterogeneity for the Hadoop factors to provide a complete performance analysis of Hadoop schedulers in terms of



Scheduler	Job	Completion Time	Average Completion Time
FIFO	J1 J2 J3	3 5 8	5.33
Fair Sharing	J1 J2 J3	7 8 8	7.67
COSHH	J1 J2 J3	7 8 8	7.67

Fig. 2: Job assignment by a) FIFO, b) Fair Sharing, and c) COSHH schedulers, and their average completion times in *System B*

system heterogeneity. This section defines the experimental environment and performance metrics.

4.1 High-level Description

In a Hadoop system, the incoming jobs can be heterogeneous with respect to various features such as number of tasks, data and computation requirements, arrival rates, and execution times. Also, Hadoop resources may differ in capabilities such as data storage and processing units. The assigned priorities and minimum share requirements may differ between users. Moreover, the type and number of jobs assigned by each user can be different. As the mean execution time for a job, $m(J_i, R_j)$, reflects the heterogeneity of both workload and cluster factors, this paper considers four possible cases of heterogeneity of users and mean execution times, as follows.

- Homogeneous System: both the workload and cluster are homogeneous, and the users can be either homogeneous or heterogeneous. In these systems, the job sizes can significantly affect the performance of the Hadoop schedulers. Therefore, two case studies are defined for this category:
  - *Homogeneous-Small* (all jobs are small).
  - *Homogeneous-Large* (all jobs are large).
- Heterogeneous System: both the workload and cluster are heterogeneous, and users are either homogeneous or heterogeneous. In this system, the challenging issue for the schedulers is the arrival rates of different size jobs. This issue motivated us to define three case studies in this category:
  - *Heterogeneous-Small* (higher arrival rate for small jobs).
  - *Heterogeneous-Equal* (equal arrival rates for all job sizes).
  - *Heterogeneous-Large* (higher arrival rate for large jobs).

  Overall, five case studies are defined which are evaluated in Sections 5 and 6.

4.2 Experimental Environment

The general settings of the Hadoop factors in our experiments are defined as follows:

1. *Workload*: jobs are selected from Yahoo! production Hadoop MapReduce traces, presented in [7]. The trace is from a cluster at Yahoo!, covering three weeks in late February/early March 2009. It contains a list of job submission and completion times, data sizes of the input, shuffle and output stages, and the running time for the map and reduce functions. The research discussed in [7] performed an analysis of the trace, which provides classes of "common jobs" using k-means clustering. The details of the workloads used for evaluating schedulers are provided in Table 1. It should be clarified that the sizes of jobs are defined based on their execution

Job Categories	Duration (sec)	Job	Input	Shuffle	Output	Map Time	Reduce Time
Small jobs	60	114	174 MB	73*MB*	6*MB*	412	740
Fast aggregate	2100	23	568 GB	76*GB*	3.9*GB*	270376	589385
Expand and aggregate	2400	10	206 GB	1.5*TB*	133*MB*	983998	1425941
Transform expand	9300	5	806 GB	235*GB*	10*TB*	257567	979181
Data summary	13500	7	4.9 TB	78*GB*	775*MB*	4481926	1663358
Large data summary	30900	4	31 TB	937*GB*	475*MB*	33606055	31884004
Data transform	3600	36	36 GB	15*GB*	4.0*GB*	15021	13614
Large data transform	16800	1	5.5 TB	10*TB*	2.5*TB*	7729409	8305880

Table 1: Job categories in the Yahoo! trace. Map time and Reduce time are in Task-seconds, e.g., 2 tasks of 10 seconds each is 20 Task-seconds [7].

   times reported in [7]. Wherever it is not defined explicitly, by "small jobs" and "large jobs" we mean the "Small jobs" and "Large data summary" classes in the Yahoo! workload. The default number of jobs is 100, which is sufficient to contain a variety of the behaviours in our Hadoop workload.

2. *Clusters:* have different configurations for the heterogeneous and homogeneous case studies. In experiments with heterogeneous resources, a cluster of six resources is used (Table 2). The bandwidth between the resources is 100Mbps. Experiments with a homogeneous cluster use a cluster consisting of six R_3 resources.

Resources	Slot		Mem	
	slot#	*execRate*	*Capacity*	*RetrieveRate*
R_1	2	$5MHz$	$4TB$	$9Gbps$
R_2	16	$500MHz$	$400KB$	$40Kbps$
R_3	16	$500MHz$	$4TB$	$9Gbps$
R_4	2	$5MHz$	$4TB$	$9Gbps$
R_5	16	$500MHz$	$400KB$	$40Kbps$
R_6	2	$5MHz$	$400KB$	$40Kbps$

Table 2: Resources in the heterogeneous cluster

3. *Users:* have two different settings in these experiments. In both settings, each user submits jobs from one of the job classes in Table 1. The heterogeneous users are defined with different minimum shares and priorities (Table 3). The minimum share of each user is defined to be proportional to its submitted job size. Therefore, the minimum share of U_2 (who is submitting the smallest jobs) is defined to be zero, and the minimum share of U_8 (who is submitting the largest jobs) is set to the maximum amount.

User	*MinimumShare*	*Priority*
U_1	5	1
U_2	0	2
U_3	10	2
U_4	15	1
U_5	10	1
U_6	15	2
U_7	10	1
U_8	15	1

Table 3: Heterogeneous Users

The priorities and the minimum shares of users are usually defined by the Hadoop provider. Finally, in the case of homogeneous users, there are eight users, each with zero minimum share, and priority equal to one.

A MapReduce simulator, MRSIM [11], is used to simulate a Hadoop cluster and evaluate the schedulers. The Hadoop block size is set to 128MB, which is the default size in Hadoop 0.21. Also, the data replication number is set to the default value of three in all algorithms. In the experiments, we use the version of the Fair Sharing algorithm presented in [13]. The pools and weights are set based on the users and the priority of the users. The preemption is set to be off, and there can be multiple tasks assigned to slots of one resource in each heartbeat.

4.3 Performance Metrics

There is a range of performance metrics that are of interest to both users and Hadoop providers. Five Hadoop performance metrics are used for evaluating the schedulers in this paper, including:

1. *Average Completion Time*: is the average completion time of all completed jobs.
2. *Dissatisfaction*: measures how much the scheduling algorithm is successful in satisfying the minimum share requirements of the users.
3. *Fairness*: measures how fair a scheduling algorithm is in dividing the resources among users.
4. *Locality*: is defined as the proportion of tasks which are running on the same resource as where their stored data are located.
5. *Scheduling Time*: is the total time spent for scheduling all of the incoming jobs. This measures the overhead of each Hadoop scheduler.

Detailed definitions of these performance metrics are provided in [27]. To calculate the execution time of the jobs, we used the Task Scheduling Process component defined in [27], which uses the algorithm introduced in [29].

5 Evaluation: Homogeneous Hadoop System

This section includes the performance analysis of three schedulers on a homogeneous Hadoop system. An example of the these systems is the use of storage and computing power from Amazon Web Services to convert 11 million public domain articles in the New York Times archives from scanned images into PDF format [12].

5.1 Case Study 1: Homogeneous-Small

This case study analyzes the performance of Hadoop schedulers for a homogeneous cluster and workload, where all the jobs are of small size. The workload consists of 100 “Small jobs” as defined in Table 1. We ran two experiments. In the first experiment the users are heterogeneous, while in the second they are homogeneous (as defined in Section 4.2).

In this homogeneous environment, the average completion times of all schedulers are almost equal. As the cluster and workload are homogeneous, the COSHH algorithm suggests all resources as the best choice for all job classes. Therefore, its performance is similar to the Fair Sharing algorithm. Moreover, due to the homogeneity in users, the Fair Sharing algorithm defines similar job pools for all users, where each job pool uses the FIFO algorithm to select a job. Therefore, despite the heterogeneity of users, the average completion time of all the algorithms are almost the same (around 98.8 seconds).

The scheduling overheads in the homogeneous-small Hadoop system are presented in Figure 3. The Scheduling Complexity problem in the COSHH algorithm leads to higher scheduling time and overhead. This is caused by the classification

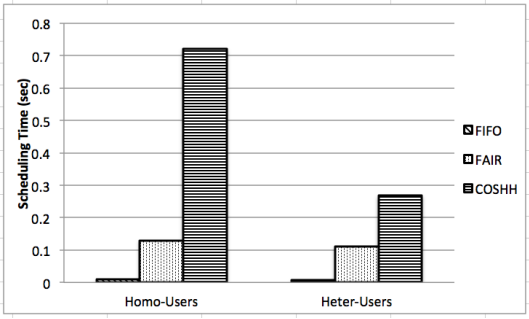


Fig. 3: Scheduling time - Homogeneous-Small

and LP solving processes. As the job sizes are small, the scheduling overhead of the COSHH algorithm does not lead to a significant increase in its average completion time. The total scheduling overhead here is less than a second, which is negligible compared to the processing times.

Tables 4 and 5 present the fairness, dissatisfaction, and locality when the users are heterogeneous and homogeneous, respectively. As the main goal of the Fair Sharing algorithm is to improve the fairness and minimum share satisfaction, it leads to better fairness and dissatisfaction.

Metrics	FIFO	Fair	COSHH
Dissatisfaction	8.615	0.801	0.824
Fairness	4.004	2.043	2.198
Locality (%)	97	97	97

Table 4: Dissatisfaction, fairness, and locality - heterogeneous users

Metrics	FIFO	Fair	COSHH
Fairness	0.447	0.447	0.447
Locality (%)	97	97	97

Table 5: Fairness and locality - homogeneous users

5.2 Case Study 2: Homogeneous-Large

Here we ran an experiment in a homogenous cluster and workload in which all the jobs are of large size. A workload consisting of 100 “Large data summary” jobs was used (Table 1) . The evaluation was performed for both homogeneous and heterogeneous users.

Figure 4 presents the average completion times for this system. Because the jobs are large, the scheduling complexity problem in the COSHH and the Fair Sharing

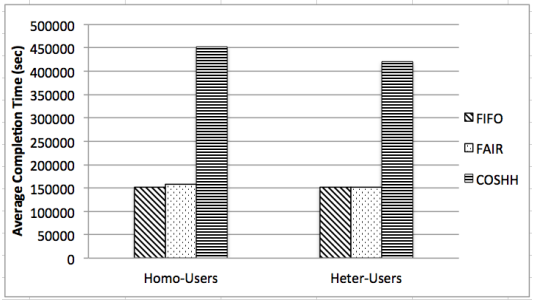


Fig. 4: Average completion time - Homogeneous-Large

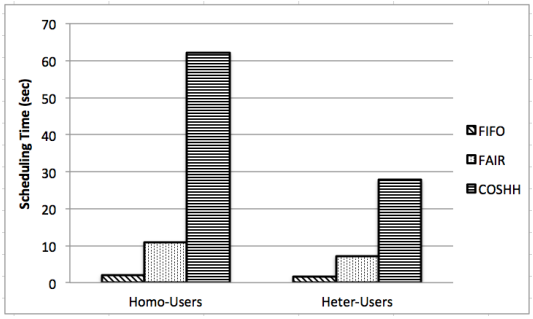


Fig. 5: Scheduling time - Homogeneous-Large

algorithms leads to increases in their average completion times. The scheduling overheads are presented in Figure 5. As the COSHH algorithm suggests all job classes as the best choice for each resource, and all jobs are large, the sort and search spaces are large. Therefore, the scheduling time of the COSHH algorithm is higher compared to COSHH for the small homogeneous workload. This leads to an increase in average completion time for the COSHH algorithm.

Tables 6 and 7 present the fairness, dissatisfaction, and locality of the large homogeneous Hadoop cluster for heterogeneous and homogeneous users, respectively. The results show the competitive performance of the Fair Sharing and the COSHH algorithms for these metrics.

Metrics	FIFO	Fair	COSHH
Dissatisfaction	8.345	5.470	6.954
Fairness	4.183	1.753	1.870
Locality (%)	97	97	96

Table 6: Dissatisfaction, fairness, and locality - heterogeneous users

Metrics	FIFO	Fair	COSHH
Fairness	0.278	0.247	0.322
Locality (%)	97	97	97

Table 7: Fairness and locality - homogeneous users

6 Evaluation: Heterogeneous Hadoop System

In this section, experimental results are provided to analyze the performance of schedulers in more heterogeneous environments. In these experiments we use a cluster of six heterogeneous resources as presented in Table 2. Each experiment is performed for both heterogeneous and homogeneous users.

6.1 Case Study 3: Heterogeneous-Small

This case study evaluates the schedulers when the system is heterogeneous, and the proportion of small jobs is high. This workload is similar to the Yahoo! workload, where the arrival rates of small jobs are higher. Figure 6 presents the average completion times for this system when the users are either homogeneous or heterogeneous.

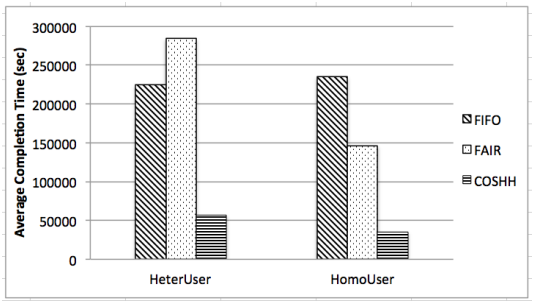


Fig. 6: Average completion time - Heterogeneous-Small

As the Fair Sharing algorithm does not have the problem of Small Jobs Starvation, we expect better average completion time for this scheduler than for the FIFO algorithm. However, in the case of heterogeneous users, because the minimum shares are defined based on the job sizes, and the Fair Sharing algorithm first satisfies the minimum shares, it executes most of the small jobs after satisfying the minimum shares of the larger jobs. Therefore, the completion times of the small jobs (the majority of the jobs in this workload) are increased. As the COSHH algorithm solves the Resource and Job Mismatch problem, it leads to 74.49% and 79.73% improvement in average completion time over the FIFO algorithm, and the Fair Sharing algorithm, respectively.

In the case of homogeneous users, where there is no minimum share defined, the Fair Sharing algorithm achieves better average completion time than the FIFO

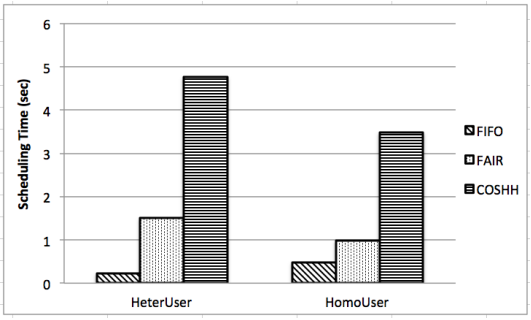


Fig. 7: Scheduling time - Heterogeneous-Small

algorithm. The Fair Sharing algorithm achieves a 37.72% smaller average completion time than the FIFO algorithm, and the COSHH algorithm reduces the average completion time of the Fair Sharing algorithm by 75.92%.

The overhead of the scheduling algorithms is presented in Figure 7. Because most of the jobs in this workload are small, and they have fewer tasks, the scheduling overheads are low. Fairness, dissatisfaction, and the locality of the algorithms are presented in Tables 8 and 9 for heterogeneous and homogeneous users, respectively. The results show that the Fair Sharing algorithm has the best performance in these metrics, and the COSHH algorithm has the second best performance.

Metrics	FIFO	Fair	COSHH
Dissatisfaction	8.618	$7.16E-04$	1.209
Fairness	4.974	0.965	2.779
Locality (%)	95.6	97.4	96.5

Table 8: Dissatisfaction, fairness, and locality - heterogeneous users

Metrics	FIFO	Fair	COSHH
Fairness	1.032	0.504	0.856
Locality (%)	94.9	97.9	98.1

Table 9: Fairness and locality - homogeneous users

6.2 Case Study 4: Heterogeneous-Large

In this case study, we evaluate the schedulers in a heterogeneous cluster with a greater proportion of large jobs. In this workload, the number of jobs from larger size Yahoo! classes (classes 4 and higher in Table 1) is greater than the number from the smaller size job classes. Figure 8 presents the average completion times for heterogeneous and homogeneous users. When the users are heterogeneous, the Fair Sharing algorithm achieves the best average completion time. The reason is that this algorithm satisfies the minimum shares first, where the minimum shares are defined based on

the job sizes. As a result, minimum share satisfaction helps to reduce the average completion time. The COSHH algorithm has the second best average completion time as a result of addressing the Resource and Job Mismatch problem. In this system, the Fair Sharing algorithm reduces the average completion time of the FIFO and the COSHH algorithms by 47% and 22%, respectively.

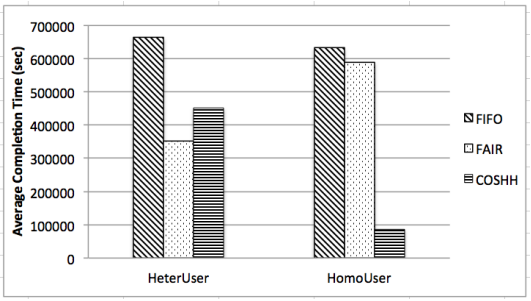


Fig. 8: Average completion time - Heterogeneous-Large

However, when the users are homogeneous, and no minimum share is defined, the average completion time of the Fair Sharing algorithm becomes higher than the COSHH algorithm. As the Fair Sharing algorithm does not have the problem of Small Jobs Starvation, it achieves better average completion time than the FIFO algorithm. Based on the simulation results, the COSHH algorithm reduces the average completion time of the FIFO and Fair Sharing algorithms by 86.18% and 85.17%, respectively.

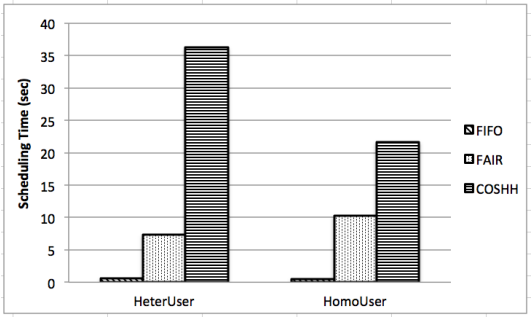


Fig. 9: Scheduling time - Heterogeneous-Large

As the job sizes of this workload are larger than for the workload in Section 6.1, the scheduling times in Figure 9 are correspondingly higher. When the users are homogeneous the scheduling overhead of the COSHH algorithm is lower, as the classification and LP solving processes are shorter. When there is no minimum share

defined, the Fair Sharing algorithm has to spend more time on computing the fair shares, and sorting the jobs accordingly. This leads to higher scheduling times.

Metrics	FIFO	Fair	COSHH
Dissatisfaction	8.223	$3.12E-04$	0.141
Fairness	6.523	1.651	0.697
Locality (%)	93.5	97.4	97.6

Table 10: Dissatisfaction, fairness, and locality - heterogeneous users

Metrics	FIFO	Fair	COSHH
Fairness	2.152	1.264	0.605
Locality (%)	95.3	98.0	98.1

Table 11: Fairness and locality - homogeneous users

Tables 10 and 11 present the dissatisfaction, fairness, and locality when the users are heterogeneous and homogeneous, respectively. The COSHH algorithm has competitive performance with the Fair Sharing algorithm with respect to all three metrics.

6.3 Case Study 5: Heterogeneous-Equal

In this case study an equal number of jobs are submitted from each of the Yahoo! classes. Figure 10 shows the average completion times when the users are homogeneous and heterogeneous. As the jobs and resources are all heterogeneous, it is important to consider the Resource and Job Mismatch problem in making scheduling decisions.

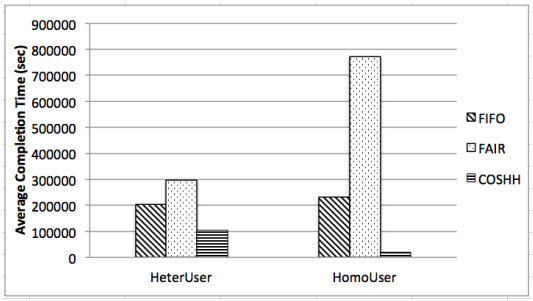


Fig. 10: Average completion time - Heterogeneous-Equal

The COSHH algorithm achieves the best average completion time. Because the arrival rates of all job classes are similar, the Sticky Slot problem in the Fair Sharing algorithm happens with higher frequency. Therefore, the Fair Sharing algorithm has larger average completion time than the FIFO algorithm. The COSHH algorithm reduces the average completion time of the FIFO and Fair Sharing algorithms by

49.28% and 65.24%, respectively. In the case of homogeneous users, where no minimum shares are assigned, the Fair Sharing algorithm has the highest average completion time, which is caused by the Sticky Slot problem. When the users are homogeneous, the COSHH algorithm reduces the average completion time of the FIFO and Fair Sharing algorithms by 90.28% and 97.29%, respectively. When the users do not have minimum shares, the COSHH algorithm has just one class. Therefore, its overhead is reduced, which leads to a greater reduction in the average completion times.

The overheads in Figure 11 show that the improvement of average completion time in the COSHH scheduler is achieved at the cost of increasing the overhead of scheduling. The additional 10 second overhead for the COSHH algorithm, compared to the improvement for average completion time (which is around 200K seconds) is negligible. Further studies in [3] show that even if the number of resources in the Hadoop cluster scales up, the COSHH algorithm still can provide a similar level of improvement in the average completion time. Because both the Fair Sharing and the COSHH algorithms search over the users and jobs to satisfy the minimum shares and fairness, they both have higher scheduling time than the FIFO algorithm.

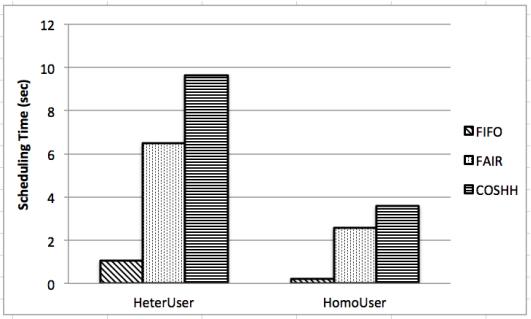


Fig. 11: Scheduling time - Heterogeneous-Equal

Metrics	FIFO	Fair	COSHH
Dissatisfaction	8.287	0.0158	0.0247
Fairness	5.961	2.939	2.309
Locality (%)	93.7	95.2	87.6

Table 12: Dissatisfaction, fairness, and locality - heterogeneous users

Metrics	FIFO	Fair	COSHH
Fairness	1.195	1.127	1.085
Locality (%)	92.5	97.7	98.6

Table 13: Fairness and locality - homogeneous users

Tables 12 and 13 present the dissatisfaction, fairness, and locality when the users are heterogeneous and homogeneous, respectively. The COSHH algorithm has competitive performance with the Fair Sharing algorithm in improving the fairness and dissatisfaction. Further detailed analysis of these Hadoop schedulers is provided in the first author's PhD thesis [28].

7 Guidelines for Scheduler Selection

The provided experimental results and analysis show that a scheduler should be selected based on the heterogeneity levels of the Hadoop factors.

Figure 12 presents guidelines suggested by our observations. The main performance metric used for determining these guidelines is the average completion time. However, the selected algorithms are also either competitive or better than the other schedulers with respect to the other performance metrics. To determine small size jobs, a threshold is defined based on the mean job execution times on resources and the time between heartbeats from the system. In these experiments, if the execution time of a job is less than the interval between two heartbeats, the job is considered to be small. However, the threshold can be customized for different Hadoop systems. How to do this (and which features are important to take into consideration) would be a useful topic for further work.

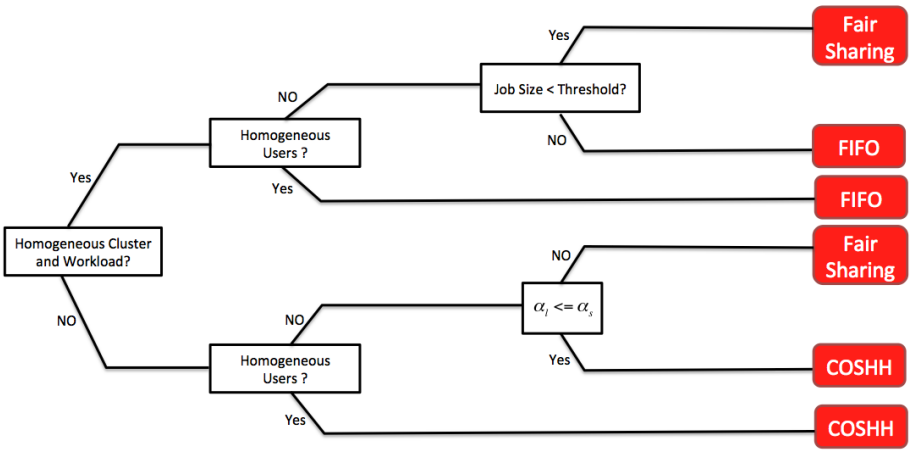


Fig. 12: Suggested schedulers. Here α_l and α_s are the arrival rates for large and small jobs, respectively.

To evaluate the proposed guidelines, we used another real Hadoop workload, presented in Table 14. The workload contains 100 jobs of a trace from a cluster at Facebook (FB), spanning six months from May to October 2009. In the evaluation, there are 10 homogeneous users with zero minimum shares, and equal priorities. Moreover, there are 10 heterogeneous users as presented in Table 15. Each user submits

jobs from one category in Table 14. The experimental environment is defined similar to that in Section 4.

Job Categories	Duration (sec)	Job	Input	Shuffle	Output	Map Time	Reduce Time
Small jobs	32	126	21*KB*	0	871*KB*	20	0
Fast data load	1260	25	381*KB*	0	1.9*GB*	6079	0
Slow data load	6600	3	10 KB	0	4.2*GB*	26321	0
Large data load	4200	10	405 KB	0	447*GB*	66657	0
Huge data load	18300	3	446 KB	0	1.1*TB*	125662	0
Fast aggregate	900	10	230 GB	8.8*GB*	491*MB*	104338	66760
Aggregate and expand	1800	6	1.9 TB	502*MB*	2.6*GB*	348942	76736
Expand and aggregate	5100	2	418 GB	2.5*TB*	45*GB*	1076089	974395
Data transform	2100	14	255 GB	788*GB*	1.6*GB*	384562	338050
Data summary	3300	1	7.6 TB	51*GB*	104*KB*	4843452	853911

Table 14: Job categories in Facebook trace. Map time and Reduce time are in Task-seconds [7].

User	*MinimumShare*	*Priority*
U_1	5	1
U_2	0	2
U_3	0	2
U_4	5	1
U_5	10	2
U_6	15	1
U_7	4	2
U_8	10	1
U_9	10	1
U_{10}	15	1

Table 15: Heterogeneous Users in FB workload

7.1 Homogeneous Hadoop

Figures 13-15 show the average completion time and the scheduling overhead in two case studies of homogeneous Hadoop systems. As the average completion time of all the algorithms in the Homogeneous-Small case are almost the same (around 98.8 seconds), its corresponding chart is not included in the figures. The results confirm our observations for the Yahoo! workloads. The guideline selects the FIFO algorithm when the system is homogeneous in all three factors. When the job size is small and the users are heterogeneous, the guideline suggests the Fair Sharing algorithm to improve the fairness.

7.2 Heterogeneous Hadoop

Figures 16-21 show the average completion times and the scheduling times in the three case studies involving heterogeneous systems. In these experiments, the COSHH algorithm is the recommended scheduler in the majority of cases.

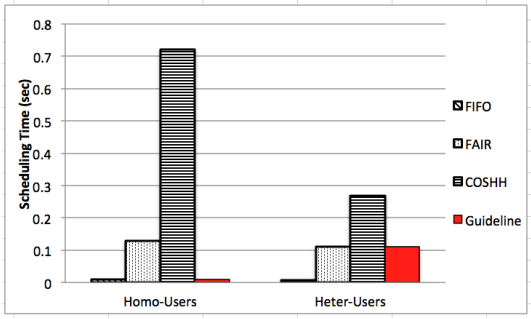


Fig. 13: Scheduling time - Homogeneous-Small

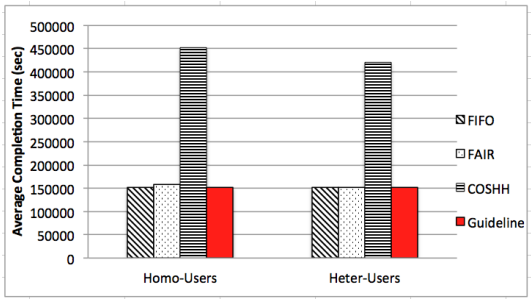


Fig. 14: Average completion time - Homogeneous-Large

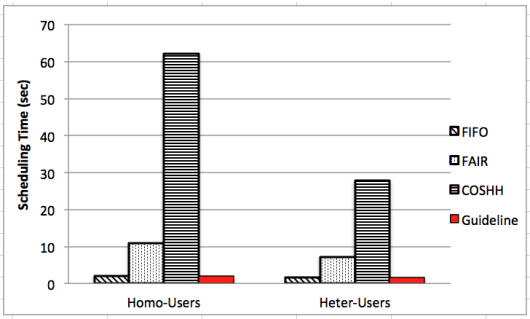


Fig. 15: Scheduling time - Homogeneous-Large

8 Related Work

Hadoop has gained from rapidly growing applications which rely on Big Data analytics. However, practical Hadoop applications require considering a variety of performance issues and architectural requirements for different components of the deployed Hadoop ecosystem [17] [18].

MapReduce is the programming paradigm that allows for massive scalability across large amounts of data for Hadoop. Zhang et al. [16] propose iMapReduce

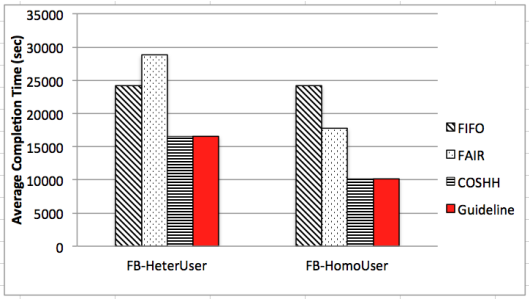


Fig. 16: Average completion time - Heterogeneous-Small

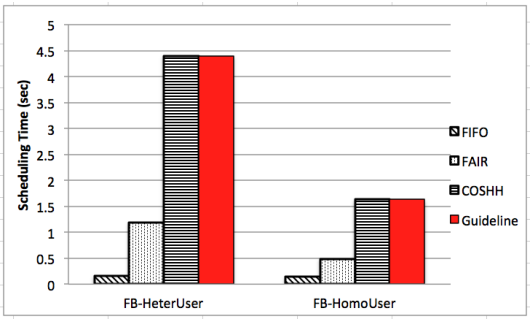


Fig. 17: Scheduling time - Heterogeneous-Small

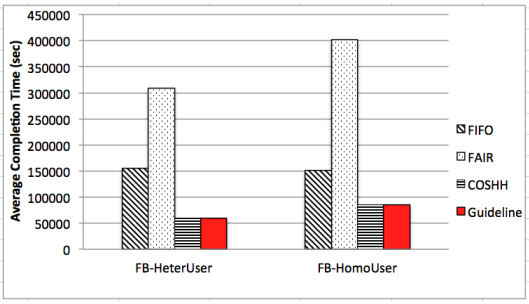


Fig. 18: Average completion time - Heterogeneous-Equal

to speed up the processing time for Hadoop by reducing the overhead of creating new MapReduce jobs repeatedly as well as allowing asynchronous execution of map tasks. Similarly, [23] and [25] address optimizing MapReduce tasks on heterogeneous clusters. Yang et al. [15] propose a statistical analytics approach to predict the performance of various workloads under different Hadoop configurations. In addition to the MapReduce and workload models, the choice of Hadoop impacts the overall performance of a Hadoop ecosystem.

The scheduler is the centrepiece of a Hadoop system. Thanks to recent developments, Hadoop implements the ability for pluggable schedulers [19]. Desired per-

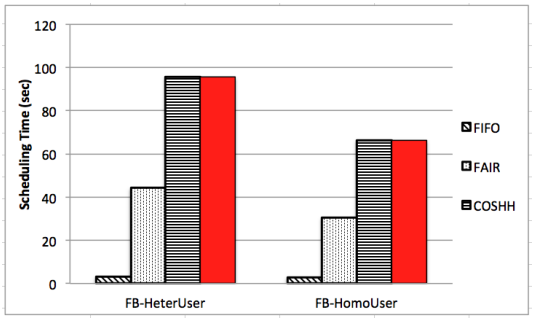


Fig. 19: Scheduling time - Heterogeneous-Equal

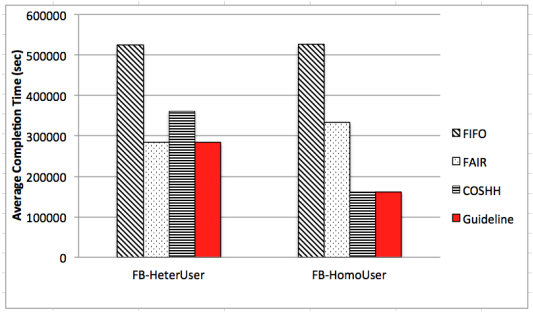


Fig. 20: Average completion time - Heterogeneous-Large

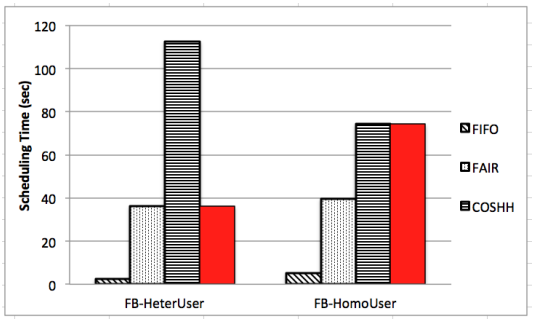


Fig. 21: Scheduling time - Heterogeneous-Large

formance levels can be achieved by an appropriate submission of jobs to resources based on the system heterogeneity. The primary Hadoop schedulers, like FIFO, are simple algorithms which use small amounts of system parameters and state information to make quick scheduling decisions. The FIFO algorithm works well for the first generation of small Hadoop clusters. However, experience from deploying Hadoop in large systems shows that simple scheduling algorithms like FIFO can cause severe performance degradation; particularly in systems that share data among multiple users [13]. To address some of the shortcomings of the FIFO algorithm, additional

schedulers are introduced in [13], where they are collectively known as Fair Sharing. The Fair Sharing algorithm does not achieve good performance with respect to data locality [4]. Delay Scheduler ([4] and [21]) is a complementary algorithm for Fair Sharing which improves data locality. However, even Delay Scheduler does not consider heterogeneity in the system.

The main concern in the most popular Hadoop schedulers is to quickly multiplex the incoming jobs on the available resources. Therefore, they use less system parameters and state information, which makes these algorithms an appropriate choice for homogeneous Hadoop systems. However, a scheduling decision based on a small number of parameters and state information may cause some challenges such as less locality, and neglecting the system heterogeneity. Later proposed algorithms, such as [14], improve scheduling decisions by providing the system parameters and state information as an input to the scheduler. However, the poor adaptability, and the large overhead of this algorithm (it is based on virtual machine scheduling), make it an impractical choice for our research.

There are a number of Hadoop schedulers developed for restricted heterogeneous systems such as Dynamic Priority (DP) [6] and Dominant Resource Fairness (DRF) [9]. The former is a parallel task scheduler which enables users to interactively control their allocated capacities by dynamically adjusting their budgets. The latter addresses the problem of fair allocation of multiple types of resources to users with heterogeneous demands. Finally COSHH [5] is specifically proposed for heterogeneous environments.

This paper evaluates Hadoop schedulers to propose heterogeneity-based guidelines. Although COSHH has shown promising results for systems with various types of jobs and resources, its scheduling overhead can be a barrier for small and homogeneous systems. DP was developed for user-interactive environments, differing from our target systems. The DRF scheduler addresses the problem of how to fairly share multiple resources when users have heterogenous demands on them. It suggests to implement a max-min fairshare algorithm over the so-called dominant user's share. Dominant share is the maximum share that a user has been allocated of any resource. Such a resource is then called a dominant resource. Whenever there are available resources and tasks to run, the DRF scheduler assigns a resource to the user with smallest dominant share. This scheduler considers heterogeneity in terms of user requests for the resources. However, heterogeneity in terms of various types of resources in the system, and providing suggestions for matching the jobs with resources based on their heterogeneity are fully addressed in the DRF scheduler.

9 Conclusion

This paper studies three key Hadoop factors, and the effect of heterogeneity in these factors on the performance of Hadoop schedulers. Performance issues for Hadoop schedulers are analyzed and evaluated in different heterogeneous and homogeneous settings. Five case studies are defined based on different levels of heterogeneity in the three Hadoop factors. Based on these observations, guidelines are suggested for

choosing a Hadoop scheduler according to the level of heterogeneity in each of the factors considered.

We plan to extend this work in three directions: (i) the proposed guideline will be evaluated in much larger systems. We plan to scale up the number of jobs, resources, and users; (ii) the required threshold specifying small versus large jobs will be further investigated. The outcome will be a selection function that considers system parameters including type, number, and complexity of jobs as well as specification of available resources; (iii) the guidelines will be extended to consider other performance metrics. The end result should be that based on a desired performance level in different metrics, and the system information, the guideline suggests an appropriate algorithm.

Acknowledgements This work was supported by the Natural Sciences and Engineering Research Council of Canada. A major part of this work was done while both authors were visiting UC Berkeley. In particular, the first author would like to thank Randy Katz, Ion Stoica, Yanpei Chen and Sameer Agarwal for their comments on our research. Also, the authors gratefully acknowledge Facebook and Yahoo! for permission to use their workload traces in this research.

References

1. J. Dean, S. Ghemawat, MapReduce: Simplified data processing on large clusters, Communications of the ACM 51 (2008) 107–113. doi:http://doi.acm.org/10.1145/1327452.1327492.
2. K. Sankar, S. A. Bouchard, Enterprise Web 2.0, Cisco Press, 2009.
3. A. Rasooli, D. G. Down, A hybrid scheduling approach for scalable heterogeneous Hadoop systems, in: Proceedings of the 5th IEEE Workshop on Many-Task Computing on Grids and Supercomputers (MTAGS12), Salt Lake City, Utah, USA, 2012.
4. M. Zaharia, D. Borthakur, J. S. Sarma, K. Elmeleegy, S. Shenker, I. Stoica, Delay scheduling: A simple technique for achieving locality and fairness in cluster scheduling, in: Proceedings of the 5th European Conference on Computer Systems, Paris, France, 2010, pp. 265–278. doi:http://doi.acm.org/10.1145/1755913.1755940.
5. A. Rasooli, D. G. Down, An adaptive scheduling algorithm for dynamic heterogeneous Hadoop systems, in: Proceedings of the 2011 Conference of the Center for Advanced Studies on Collaborative Research, CASCON '11, IBM Corp., Toronto, Ontario, Canada, 2011, pp. 30–44.
URL `http://dl.acm.org/citation.cfm?id=2093889.2093893`
6. T. Sandholm, K. Lai, Dynamic proportional share scheduling in Hadoop, in: Proceedings of the 15th Workshop on Job Scheduling Strategies for Parallel Processing, Heidelberg, 2010, pp. 110–131.
7. Y. Chen, A. Ganapathi, R. Griffith, R. H. Katz, The case for evaluating MapReduce performance using workload suites, in: Proceedings of the 19th Annual IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, Washington, DC, USA, 2011, pp. 390–399. doi:http://doi.ieeecomputersociety.org/10.1109/MASCOTS.2011 .12.
8. Apache, Hadoop On Demand documentation, [Online; accessed 30-November-2010] (2007).
URL `http://hadoop.apache.org/common/docs/r0.17.2/hod.html`
9. A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, S. Shenker, I. Stoica, Dominant Resource Fairness: Fair allocation of multiple resource types, in: Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation, USENIX Association, 2011, pp. 24–24.
URL `http://dl.acm.org/citation.cfm?id=1972457.1972490`
10. Y. Chen, S. Alspaugh, R. Katz, Interactive analytical processing in Big Data systems: A cross-industry study of MapReduce workloads, Proceedings of the International Conference on Very Large Data Bases (VLDB) Endowment 5 (12) (2012) 1802–1813.
URL `http://dl.acm.org/citation.cfm?id=2367502.2367519`
11. S. Hammoud, M. Li, Y. Liu, N. K. Alham, Z. Liu, MRSim: A discrete event based MapReduce simulator, in: Proceedings of the 7th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2010), IEEE, 2010, pp. 2993–2997.

12. D. Gottfrid, Self-service, Prorated super computing fun, http://tinyurl.com/2pjh5n (March 2009).
13. M. Zaharia, D. Borthakur, J. S. Sarma, K. Elmeleegy, S. Shenker, I. Stoica, Job scheduling for multi-user MapReduce clusters, Tech. Rep. UCB/EECS-2009-55, EECS Department, University of California, Berkeley (April 2009).
URL http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS -2009-55.html
14. A. Aboulnaga, Z. Wang, Z. Y. Zhang, Packing the most onto your Cloud, in: Proceedings of the First International Workshop on Cloud Data Management, 2009, pp. 25–28. doi:10.1145/1651263.1651268
15. H. Yang, Z. Luan, W. Li, D. Qian,MapReduce Workload Modeling with Statistical Approach, in: Journal of Grid Computing, 2012, vol 10, no 2, pp. 279-310, doi:10.1007/s10723-011-9201-4.
16. Y. Zhang, Q. Gao, L. Gao, C. Wang, iMapReduce: A Distributed Computing Framework for Iterative Computation, in: Journal of Grid Computing, 2012, vol 10, no 1, pp. 47-68, doi:10.1007/s10723-012-9204-9.
17. B. Rimal, A. Jukan, D. Katsaros, Y. Goeleven, Architectural Requirements for Cloud Computing Systems: An Enterprise Cloud Approach, in: Journal of Grid Computing, 2011, vol 9, no 1, pp. 3-26, doi:10.1007/s10723-010-9171-y.
18. J. Shamsi, M. Khojaye, M. Qasmi, Data-Intensive Cloud Computing: Requirements, Expectations, Challenges, and Solutions, in: Journal of Grid Computing, 2011, vol 9, no 1, pp. 3-26, doi:10.1007/s10723-010-9171-y.
19. M. Jones, Self-service, Scheduling in Hadoop: An introduction to the pluggable scheduler framework, http://www.ibm.com/developerworks/library/os-hadoop-scheduling/ (December 2011).
20. T. White,Hadoop: The Definitive Guide, Book, Third Edition, O'Reilly Media, ISBN-10:1449311520.
21. K. He-yang, Y. Qun, W. Li-song, D. Xi, Improved delay-scheduler algorithm in homogeneous Hadoop cluster, in: Application Research of Computers, 2013, issue 5, pp. 1397-1401.
22. Z. Zhang, L. Cherkasova, B. Loo, Performance Modeling of MapReduce Jobs in Heterogeneous Cloud Environments, in: Proceedings of the 2013 IEEE Sixth International Conference on Cloud Computing, 2013, pp. 839-846,doi:10.1109/CLOUD.2013.107.
23. F. Ahmad, S. Chakradhar, A. Raghunathan, T. Vijaykumar, Tarazu: Optimizing MapReduce on Heterogeneous Clusters, in: ACM SIGARCH Comput. Architure News, March 2012, vol 40, no 1, pp. 61-74,doi:10.1145/2189750.2150984.
24. K. Morton, A. Friesen, M. Balazinska, D. Grossman, Estimating the progress of MapReduce pipelines, in: Data Engineering (ICDE), 2010 IEEE 26th International Conference on, 2010, pp. 681-684,doi:10.1109/ICDE.2010.5447919.
25. M. Zaharia, A. Konwinski, A. Joseph, R. Katz, I. Stoica, Improving MapReduce Performance in Heterogeneous Environments, in: Proceedings of the 8th USENIX Conference on Operating Systems Design and Implementation, 2008, pp. 29-42.
26. M. Zaharia, A. Konwinski, A. Joseph, R. Katz, I. Stoica, Big Data Processing with Hadoop MapReduce in Cloud Systems, in: International Journal of Cloud Computing and Services Science (IJ-CLOSER), 2013, vol 2, no 1, pp. 16-27.
27. A. Rasooli, D. G. Down, COSHH: A classification and optimization based scheduler for heterogeneous Hadoop systems, to appear in: Future Generation Computer Systems. http://dx.doi.org/10.1016/j.future.2014.01.002.
28. A. Rasooli, Improving scheduling in heterogeneous Grid and Hadoop systems, Ph.D. thesis, McMaster University, Hamilton, Canada (July 2013).
29. S. Agarwal, I. Stoica, Chronos: A predictive task scheduler for MapReduce, Tech. rep., EECS Department, University of California, Berkeley, Author1 URL: http://www.cs.berkeley.edu/∼sameerag/, Author1 email: sameerag@cs.berkeley.edu (December 2010).