PREDICTION METHODS OF MAPREDUCE JOB COMPLETION TIME IN HETEROGENEOUS CLUSTER ENVIRONMENTS

Jenia Afrin Jeba

Student ID: 11259150
jaj212@usask.ca

Abstract **-** MapReduce, a programming model, used for high performance big data computing and processing, where assuming and managing a Map Reduce job's runtime performance ahead of time is quite a challenging task. There are several types of complications regarding the parameter configuration. Various types of adjustments have also to be considered regarding the actual performance gains achieved from optimization techniques and the real overhead bought out during the runtime of the jobs. One simple approach could be to repeatedly run these jobs applying various combinations of configuration settings and optimization techniques and then picking the one which would be providing the shortest runtime. Nevertheless, this is quite impractical an approach because there maybe numerous possible combinations and the real execution of all those techniques would take up so much time. Hence, it would be much appreciated if there is any kind of method for estimating the runtime of any MapReduce job efficiently without the real implementation, just via utilizing the parameter configuration settings and input data of the cluster machines. There are some existing approaches which have looked into this research area. This survey work tries to review those research works by giving a brief description of the methodologies and experimental designs along with a general idea of predicting MapReduce task time duration.

I. INTRODUCTION

Apache Hadoop MapReduce [15] is being used as a widespread model because of the extensive convenience of its employment as an open source framework for processing big data. Mainly, this framework is composed as a model for optimum implementation of applications using a cluster of similar types of nodes (homogenous) having almost same hardware configurations. Nevertheless, most organizations have heterogeneous machines with variable number of CPU cores, RAM sizes, chip organizations, memory sizes and disk speed. So, an issue might be raised for an organization regarding the efficiency of setting up a heterogeneous Hadoop cluster using existing set of diverse types of nodes having diverse hardware configuration for accomplishing workloads which would aid organizations in cutting down the e-waste and cost. Any kind of potential approach can be adopted for predicting job execution time of a MapReduce based application that has to be run on variable size of cluster with variable data sizes with various sorts of divergency in the cluster for performance concerns.

As it is known, a MapReduce (MR) job is processed in two steps: map and reduce phases, across several machines in a distributed cluster environment [15][24]. Each machine processes many map tasks in the map phase and after that a list of key-value pairs are generated as intermediate results. In the map phase, each node processes a number of map tasks, and the chunk of input data on each node is generated into a list of key-value pairs as intermediate results. All those values which have the same key, are then grouped together in the reduce phase [15].

For achieving efficiency, it is substantial and challenging to tune the performance of a MapReduce task; there are primarily two reasons behind it. Firstly, the configuration constraints, for instance – input data's chunk size and output data's buffer size during the map phase, the quantity of reduce tasks, etc. influence the performance of a MapReduce task in various ways. If these parameters aren't tuned carefully, the runtime performance of any MapReduce job is expected to be affected seriously. For instance, only some of the nodes would be occupied in the reduce phase if the reduce tasks are very less in number. Consequently, resulting into a reduction of the degree of parallelism in a great extent. Also, many intermediate results generated from the map phase would be assigned to each reduce task. In such a scenario, when any task fails, the additional time required for processing the abortive job again would be very lengthy. Then again, the overhead of starting each reduce task have a tendency to be excessive if there are several reduce tasks because it will rule over the total processing time of the reduce phase.

Researchers have developed various types of optimization algorithms for further improving the performance of MapReduce tasks, for example scheduling of jobs, optimizing algorithms, handling skewing etc. [1-25]. However, the performance wouldn't be as good as desired if various tradeoffs aren't considered during the implementation of these approaches. Such as – researchers have proposed load balancing algorithms for handling skewed input data, some of which come up with substantial overheads because of larger quantity of input chunks among the cluster machines. Therefore, the tradeoffs between the actual efficiency gains and the extra overhead created by these approaches, should be evaluated wisely.

In this survey, I will try to review such research works which have proposed the methodologies of estimating the runtime and/or performance of a MapReduce task having different types of heterogeneity among the cluster nodes. The focus of this survey will be describing the existing methods, briefly presenting their architectural design, representing their experimental results and understanding probable reasoning behind their respective performance. Followed by these, a brief analysis of the strengths and weaknesses of these approaches from my point of view would be presented. I would also try to improvise a possible idea on the basis of the comparative analysis of these methods.

This work is described as follows: Section II describes the existing approaches, Section III category wise explains the prediction methods of MapReduce jobs completion time as used in those existing approaches followed by a self-driven proposed method in Section IV and Section V summarizes the work.

II. EXISTING APPROACHES

Maximum of the existing works for predicting job accomplishment time in homogeneous clusters [1], [2], [4] consider computing the average job lengths and the quantity of map/reduce job series. But the machines would have diverse job completion time due to variances in amount of RAM, in CPU speed, disk speeds etc. in heterogeneous cluster environments letting any kind of theoretical or mathematical models to be tough to suit in such environment. Since, different nodes are assigned different amounts of tasks dynamically, so simulator-based approaches are more appropriate rather than building any kind of mathematical equation / model for the accomplishment time of a MapReduce job. To date, different types of simulators for MapReduce tasks' runtime approximation, have been projected which predict the runtime of the tasks with the help of the information of the cluster setting and tasks. Some instances are MRSim [1], SimMapReduce [2], SimMR [3], MRSG [4], HSim [6], MRPerf [9] etc. But most of these methods presume that input data are distributed across the machines evenly and all the reduce tasks will be assigned the similar number of intermediate key-value pairs generated from the map tasks and only contain the quantity of records and the input size. Thus, no modification on runtime approximations are made based on whether the input data are skewed or not.

Neither any sub-phase within a map or reduce task is modeled, nor a process on producing the function is provided by MRSG [4]. But MRSim [1] has put an effort in modeling additional facts of a MapReduce task. MRSim lets users decide to stipulate few facts of a MapReduce task, e.g. the use of a combiner function during producing intermediate output results. MRSim is based on Grid for job and task scheduling program and is written on top of SimJava [17].

SimMapReduce [2], SimMR [3] and MRSG [4] are some of the several approaches which have put focus on job and task scheduling. Grid [18], a distributed computing paradigm, was the base on which these approaches were designed. Grid is used for services such as data concentrated scientific applications or compute intensive high-performance tasks. SimMapReduce [2], alike MRPerf [9], place a uniform speed of processing for each node for computing the execution time of a map / reduce task.

According to the scheduling algorithms, SimMR [3] repeated implementation traces of workloads gathered in Hadoop clusters with diverse organizations. SimMR does not offer any model for the approximation of the runtime of a job though the traces accumulated from log files denote the runtime of jobs. Users can describe the function of any computing task's runtime.

There are quite a number of existing works proposed in past four-five years which have put their focus on estimating the runtime of MapReduce tasks. In particular, MRPerf [9] is one of the primary approaches among them which has put focus on MapReduce tasks' behavior in complex network environments e.g. - with the employment of the NS-2 [14] network emulator, with single / double rack and tiered network layers etc. MRPerf tries to simplify the computational processes by considering the runtime of any map / reduce task proportionate to the input data size and has also conducted some experiments for analyzing the effect of failure of any MapReduce task which could rarely be determined from the output results. MRPerf is mainly a simulator-based approach for simulating MR jobs for fine-tuning Hadoop systems, it doesn't explicitly predict the completion time of any MR task.

HSim [6] has put its focus on the simulation of MapReduce jobs which models the elementary constituents within the map / reduce tasks, by means of system clock for estimating the total time spent. HSim has analyzed the parameters in more detail than other works with node properties and job specifications, as well as few configuration parameters e.g. buffer size. HSim is a simulator particularly designed for MR jobs in cloud computing and again doesn't explicitly forecast the time duration of any MR task, so this paper would be out of the scope of this survey.

Starfish [7], a self-tuning system, develops an abstract model [17] for estimating the runtime of MapReduce tasks using a Profiler for collecting comprehensive statistical information of jobs. It also uses a What-if-Engine to estimate the runtime, required by a cost-based optimizer and it doesn't take into account the skewed key distribution of data though it is cognizant of skewed data distribution across the cluster machines. It is assumed by the Profiler that the same amount of intermediate results is selected by each reduce task from map tasks. In such a scenario, on condition that input data is uniformly spread across the cluster nodes, each reduce task having uneven runtime because of uneven key allocation of input data will be disregarded.

Again, authors in ARIA [8] suggest that a MR job can be profiled from the logs and based on that profile the information can be used to design MapReduce performance model for estimating the required amount of resources for meeting the duration of any task's deadline. Their proposed SLO-scheduler enforces these resource requirements.

So far, these are the notable methods being used for estimating the runtime of any MR job / simulating the MR Hadoop environment. Based on my findings, I have decided to broadly categorize them into four types:

A. Simulation based approaches: MRSim [1], MRPerf [9], HSim [6] etc.
B. Job and task scheduling based approaches: SimMapReduce [2], SimMR [3] **etc.**
C. Task tuning based approach: MRTune [10], rTuner [11] etc.
D. Abstract model using a Job Profiler based approach: Starfish [7], ARIA [8] etc.

I would provide a brief description on the design, working mechanism and experimental tests / results of these methods category-wise in the following section.

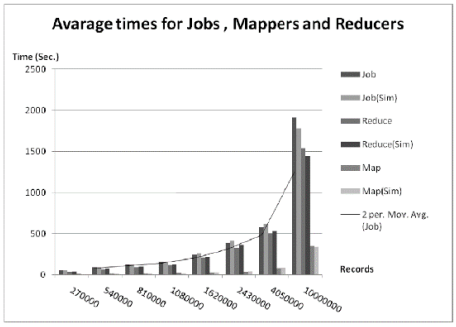
III. PREDICTION METHODS OF MAPREDUCE JOBS COMPLETION TIME

A. Simulation based approaches

(a) MRSim: A Discrete Event based MapReduce Simulator [1]

Authors in this work has tried to develop a simulator for specifically simulating the Hadoop MapReduce (MR) environment with different configurations. Based on discrete event simulation they have designed and implemented a MR simulator which accurately simulate the Hadoop environment allowing to measure the scalability of MR based applications easily and quickly. In terms of Hadoop job completion times and hardware utilization, it also portrayed the impacts of various configurations of Hadoop environment setup on MR based applications behavior.

Based on GridSim [18], MRSim builds its model for simulating the network topology and the traffic. It uses SimJava [17] discrete event engine for modelling the rest of the entities of the system. Based on object-oriented models the whole system is designed which includes HDD, CPU and Network Interface models as the basic design blocks. There is a Job Tracker assigned for each MapReduce cluster; Job Tracker and Task tracker is hosted by each of the machine which is part of the Network Topology. Several Map and Reduce tasks up to the maximum limit can be launched by each Task Tracker. Very basic simulation for Hadoop distributed file system HDFS is used by MRSim. Data splits locations and splits replication is simulated by it in local rack only.



(a)

Records	Job	Reduce	Map
270,000.00	101.75%	110.00%	81.00%
540,000.00	100.00%	113.48%	77.40%
810,000.00	100.80%	106.68%	88.14%
1,080,000.00	98.08%	105.79%	65.87%
1,620,000.00	104.86%	106.28%	101.03%
2,430,000.00	108.57%	109.41%	112.67%
4,050,000.00	105.82%	105.73%	108.60%
10,000,000.00	93.16%	93.83%	97.73%

(b)

Fig 1: (a) Execution times for simulation and experimental results in MRSim (b) Ratio of execution times (MRSim value) / (experiments value) [1]

I found that MRSim has put a good attempt in simulating additional details of a MR job. Users are allowed to specify some details of a MR job in MRSim, like if they want to use a combiner function, they can specify that while producing intermediate results.

They ran their experiments for testing job execution time, the average task times for any job, the number of HDD read and written records for each job and intermediate records in Map and Reduce tasks on different sizes of datasets on the word count benchmark where Each job consists of 60 Maps and one reducer.

It has been observed from their experimental results (shown in fig-1(a)) that when input data size increases, for Map / Reduce tasks average times for the jobs follow the same pattern of increase. Very good accuracy was found from MRSim simulator since the data flow between the Map and Reduce Tasks could be accurately simulated by it. Figure-1(b) shows a comparison between the real time taken and the times of the simulated jobs.

(b) A Hadoop MapReduce Performance Prediction Method [19]

The job analyzer proposed in this work extracts some important information from the job submitted by the user which should be attained in a reasonable time within a low latency, the information are – the input data size and amount of records, the complexity of Map / Reduce functions, the data conversion rate between the input-output data etc. Instead of processing the complete data set, the analyzer first gets a sample of the input data; then only for Map and Reduce functions it extracts and modifies the method code, all the costs for transferring data is eliminated, the cluster is initiated and lastly, the reflection mechanism of Java is used for instantiating the processing of Map and Reduce tasks submitted by the users. After that using a local weighted linear regression method the prediction module estimates the performance.

For a compute-intensive job, the main features would be the complexity of map and reduce function, for a data-intensive job the significant feature would be the amount of input and the number of records etc. which lets the prediction module to choose the most suitable weighted regression method. The weighted distance between the new instance and the ones in the history trace is measured for choosing the similar type of jobs as a new training set which would be used as the training set for their model for making prediction.

Sampling is an important part of the proposed Job Analyzer. The authors tested their sampling module using different data sizes and for each data set, compared the amount of records approximated by the job analyzer with the actual records recorded by the history trace. With a great success the average error rate scored as low as 0.02 which indicates that their proposed sampling algorithm is very efficient since it almost matches the real estimation (fig 2). However, some jobs with the highest possible input sizes produced higher error rates and their model failed to make a good prediction on those jobs their trained model had no similar jobs during training. For measuring the overheads created by their model, they defined the overhead to be an amount in between the actual job completion time derived from the history and the completion time of their framework for prediction of time; their model tended to create smaller overhead with the increase of data.

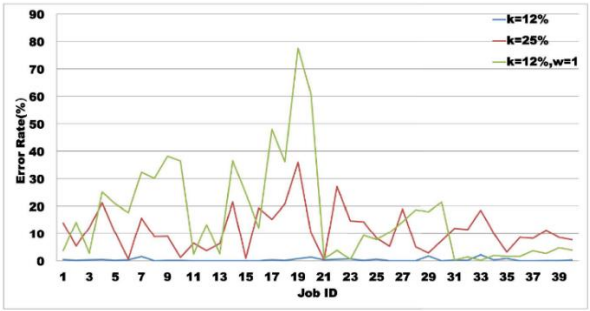


Fig 2: Map task prediction [19]

According to my opinion, the advantage of the job analyzer proposed in this work is that it can both support actual Hadoop jobs and also profile valuable information about MR jobs and pass them to the predictor. While most of the others works do not emphasize on analysis of jobs and cannot offer sufficient information (e.g. the

complication of the Map and/or Reduce functions, the rate of conversion of the data and so on), their job analyzer can. Moreover, their job analyzer is very light-weight, since it consumes only a slight extra cost to deliver abundant information. It performs not only a simulator, it is devised as a tool for debugging a Map and Reduce function, it can also gather information and evaluate job performance.

(c) Using Realistic Simulation for Performance Analysis of MapReduce Setups [9]

A fine-grained simulation model of MapReduce setups throughout different phases in a MR job has been proposed in design of MRPerf [9]. It can serve as both a design tool and a planning tool for MapReduce infrastructure, for making MapReduce arrangement far simpler via lessening the number of parameters which presently have to be tuned manually. To precisely model the network conduct during inter and intra rack task communications over network, MRPerf is built on the broadly-used ns-2 network simulator [14]. Diverse cluster topologies and their effect on the MR job performance has been the main interest of the authors are interested while modeling MRPerf. A number of simulated nodes is created by MRPerf for modeling map/reduce task where each node may have several processors. A job has simple map and reduce tasks with compute time requirements that are proportional to the data size but not the content (or type of processing) of the data.

They explored the effect of run-time parameters and cluster design on application performance and hence developed MRPerf, for facilitating performance analysis with goal of expansively capturing several design parameters of Hadoop MapReduce. The performance of MapReduce applications depends on the interaction of many factors since those applications run on large clusters. The insights gained through MRPerf lets users to understand how their MR applications might behave on a specific configuration, how the runtime performance can be improved via optimizing platforms, how the node resources, network topology and failure rates etc. should be dealt etc.

Fine-grained simulation is provided by MRPerf at sub-phase level, inter- and intra-rack network communications are also modeled by it; in addition to that activities inside a single node are also addressed by it, for instance – how much processor time a job consumes, the time required by disk I/O for reading inputs and writing results etc. Several files are taken as input by the simulator, which include cluster topology, node specification, job description and data layout. There is a ns-2 driver module in the proposed architecture which exploits the ns-2 network simulator for practically simulating the network traffic. Firstly, the simulator reads all the configuration parameters and instantiates the necessary figure of nodes organized according to the stipulated topology. Then individual node runs the job which has been scheduled for it along with sending and receiving related messages through ns-2 network simulator. The output contains a comprehensive trace regarding the timeline of individual stage of the task, the total execution time of MR job and the volume of data transferred.

Authors have performed a number of experiments utilizing a number of nodes set in a single and also double Hadoop rack which used a chunk size of 64 MB and the input size of 4 GB per node. They fixed the number of cores to 128 and studied the chunk size of 64 MB and also 128 MB both under single and double rack configuration (Figure 3(a)(b)). We also study input data size. Results for different input data size of 4GB and 8GB per node under both single and double rack configuration is showed in figure 3. Results indicate that MRPerf appropriately forecast performance for variable input data size and chunk sizes and its competence in capturing Hadoop cluster behavior.

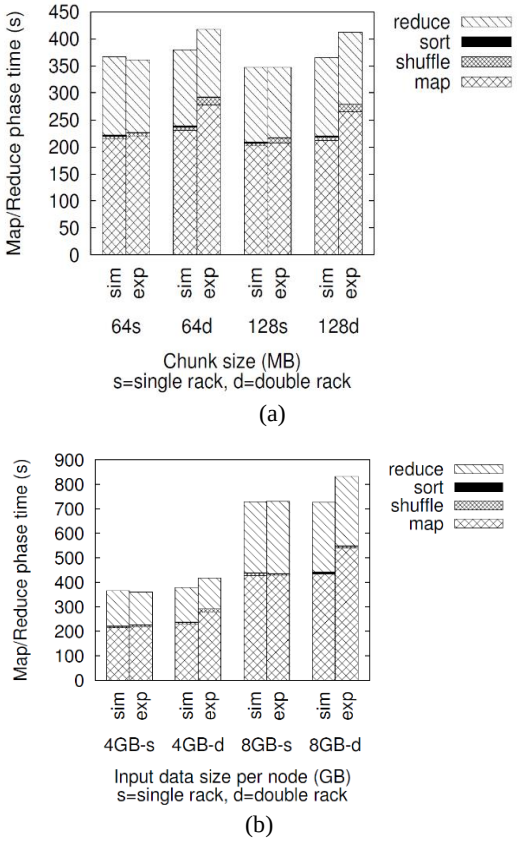


Fig 3: (a) Execution times with varying chunk size; (b) Execution times with variable input size, using real measurements and MRPerf [9]

Due to adoption of ns-2 for network simulation MRPerf shows high accuracy in simulating the effects of network topologies. But I found that the behaviors of the Hadoop framework are simulated in a limited way here. A large number of parameters affect the behaviors of Hadoop, for instance, the current states of node processors, buffers, hard disk and networks etc. highly influence the performance of Map instances in the Map phase. Again, the performances of reduce tasks are extremely reliant on the present IO states in the reduce phase. Due to the heavy dependencies on the approximations of the values of parameters, MRPerf does not simulate these real time communications correctly. The main limitations of MRPerf are:

• In MRPerf the required resources for processing a job allocated for each user is fixed. But, usually a number of users share resources in a Hadoop environment which again change dynamically.

• The exact behaviors of Map and Reduce phases cannot be simulated by MRPerf. In a Map instance, while Map task is running the spilled data will be kept writing into buffer. Again, the in-memory data too keeps spilling into hard disk concurrently when the engaged size of

the buffer is smaller than a particular threshold value. This mechanism significantly affects the number of spilled files which will further affect the IO behaviors because real time states of the system is highly uncertain. MRPerf uses a pre-defined data value and simply overlooks these dealings.

• The CPU processing will be congested until the entire content in buffer is flushed if the engaged size of the buffer is bigger than a particular threshold. MRPerf does not consider this but this event can also influence system behaviors.

• Due to lack of correct simulations in Map phase, MRPerf does a simple simulation for starting reduce tasks concurrently in the reduce phase.

• Another weakness of MRPerf is that it only homogeneous environment is supported by it, but then again both homogeneous and heterogeneous environments can be applied to Hadoop. Again, authors validated MRPerf using Search, TeraSort and Index which do not involve intricate behaviors of Hadoop while the tests were performed in a homogeneous environment.

Hence such simplifications and approximations cannot accurately imitate real world Hadoop environment letting MRPerf to become somehow weaker when simulating complex behaviors of Hadoop.

B. Job and task scheduling based approaches

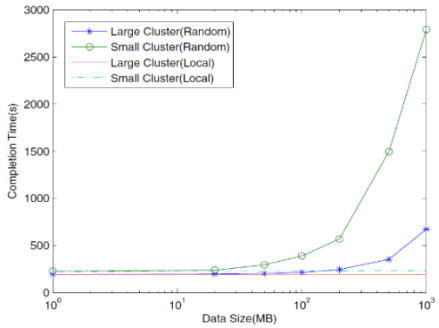
(a) SimMapReduce: a simulator for modeling MapReduce framework [2]

Authors proposed a MapReduce simulator SimMapReduce, which is based on GridSim [18], for facilitating resource management and evaluation of performance. Usability of configuration and reproducibility of deployment are two key advantages of Simulation based methods. The main contribution of this simulator is that it is able to inherit the provided java classes without concerns of employment details for applying scheduling algorithms and resource allocation policies. For managing files transfers, a FileManager is integrated which records file transmission time as a part of job completion time.

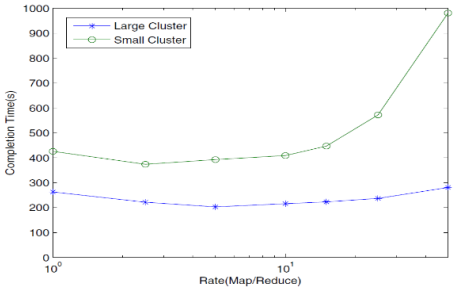
SimMapReduce applies multi-layer architecture leveraging several existing packages as separate components. Separated slots for Map and Reduce are reserved on Every node on cluster. Broker allocates nodes to users, Map/Reduce tasks are mapped by the job dispatcher to a specific node and their execution is also monitored by it. A file manager monitors the file transmission time which is actually the completion time of jobs; it also monitors initiation of input files, management of intermediate file and transmission of file.

The Modeling process goes as follows. In the beginning, every user sends request for jobs resulting into generating copies of MapTask and ReduceTask. Upon checking the availability of nodes and based on this request information, MRBroker allots free nodes to the master node. MRMaster chooses idle nodes to schedule MapTasks from the list of available nodes and MapTask is executed. When all MapTasks finishes, based on the grouping of key/value pairs MRMaster sends data having the same key to one particular ReduceTask and eventually results are written to a final output file. Upon the completion of all MapTasks and ReduceTasks, job completion is reported by MRMaster to user.

One of their experiment sketches how the completion time of whole job sequence is influenced by the data size (fig 4). If the data size is small for random scheduler, then the amount of accomplishment time primarily is contingent upon the computing capability of cluster. As there are more nodes provided large clusters, so computation can be finished more quickly than small clusters. Especially for the small cluster, with respect to data size the delay caused by transferring files among nodes worsens. For the reason that before the input data arrives the submitted task has to wait and the completion time of job sequence is greatly extended with the limitation of bandwidth during the queuing of data transmission. It is shown in the figure 4 that with the change of input size completion time slightly fluctuates.



(a)



(b)

Fig 4: (a) Influence of Data Locality (b) Influence of Task Granularity [2]

For computing the runtime of a map / reduce task, SimMapReduce set a constant processing speed to each machine. Execution traces of real workloads are replayed by SimMR which were collected in Hadoop clusters according to the scheduling algorithms. One lacking of SimMR that I felt is that, it doesn't provide any model for runtime approximation even though the collected traces from log files capture the runtime of each task in a real job.

(b) Play It Again, SimMR [3]

Since a MR cluster is generally shared thus in such shared environments a challenging problem is the capability to competently control resource provisioning among the MR jobs for accomplishing performance goals. For evaluating and comparing diverse resource provisioning and job scheduling methods authors have considered and employed a simulation environment called *SimMR* which encompasses three main components as: *i)* a Trace Generator for

generating a MR workload; *ii)* a Simulator Engine for emulating the Hadoop job master functionality; *iii)* a scheduling strategy for determining the quantity of resources allocated to each job and the scheduler choices on job ordering. They executed a set of genuine MR applications in a Hadoop cluster with 66-node and validated the accuracy of SimMR by repeating the collected job implementation traces in their simulator. Original job processing was accurately reproduced by their simulator: the accomplishment time of the simulated tasks were within 5% of the real tasks' completion time. Over one million events per second can be processed by SimMR; which means, instead of multi-hour executions in the real testbed, users are allowed to simulate complex workloads in a few seconds using SimMR. So, certainly SimMR is leveraging a simulation-based approach for resource provisioning and job completion time estimation environment without / before real-world execution of the tasks letting them to understand the task execution time duration in heterogeneous cluster scenario.

During different phases of MapReduce processing in the cluster they have used job profiles (with durations of tasks) for representing the latencies while designing SimMR. They generated job traces by means of two procedures: MRProfiler and Synthetic TraceGen. The job performance metrics are extracted at the end of each job by MRProfiler by administering the counters and logs kept at the JobTracker. Detailed information about the map and reduce tasks' processing, important information about the shuffle/sort phases etc. are recorded by the job tracker logs. On the other hand, based on the statistical properties of the workloads, Synthetic TraceGen model the distributions of the durations with the consideration of what-if scenarios. So, a job's crucial performance characteristics are summarized by these two methods during its execution in the cluster.

They say, it is required to estimate the average and maximum task durations during various execution stages of the job, e.g. map, shuffle/sort and reduce phases for approximating the overall completion time of a MR task, which is done by the MRProfiler. Based on this data, average and maximum task durations in different phases is computed and then lower and upper bounds for each execution phase of the job is also computed. By using the bounds model, they expressed the estimations for job accomplishment time (lower bound and upper bound) as a function of map/reduce tasks) and the assigned map/reduce slots.

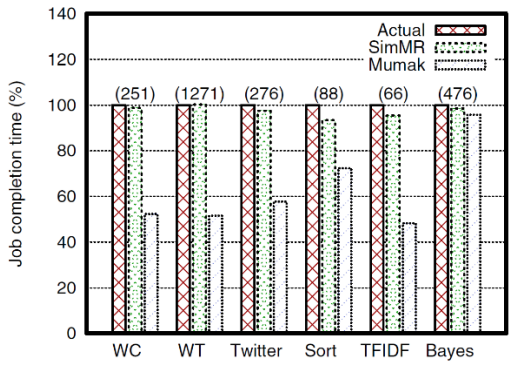


Fig 5: Simulator accuracy across FIFO scheduling policy. The numbers above the bars characterize the real job accomplishment time (in seconds) [3]

They used a real workload trace involving three accomplishments of six applications and replayed it using SimMR and Mumak. a comparison of the duration of the simulated job vs. the real job duration is showed in the fig 5. It is observed from their graphs (fig 5) that Mumak has 37% average (maximum 51.7%) error for the job completion time and SimMR repeats same the traces with less than 2.7% average (maximum 6.6%) error across all the applications.

The strength of this work according to me is that they emphasized on simulating the decisions of the job master and also the allocations of the task/slot across several jobs. Though they did not simulate detailed facts of the TaskTrackers as performed by MRPerf, their method precisely reflected the processing of job because during different phases of MapReduce processing their profiling technique represented job latencies. Also, compared to MRPerf the approach of SimMR is fast since it doesn't deal with network level packets.

C. Task tuning based approach

(a) MRTune: A Simulator for Performance Tuning of MapReduce Jobs with Skewed Data [10]

MRTune is a MapReduce simulator for estimating the runtime of MapReduce jobs; it considers the key distribution of input data and can predict the runtime of any job with high accuracy and efficiency despite skewed key distribution of data and even if any unpredictable task failure is present. Based on Discrete Event Simulation (DES), MRTune defines objects like Client, Job Tracker, Task Tracker, Mapper, Reducer and Shuffle. The function of job tracker is to schedule map and reduce tasks and after that disperse them to task trackers for running. The function of a task tracker is to process tasks and then send the reports to the Job Tracker. The runtime taken for computation is not the same between the map and the reduce phase. The shuffling phase transfers map tasks to reduce tasks. Data skewing put effect on the reduce phase and multiple threads are involved in the shuffling of a single reduce task. Figure 1 demonstrates the workflow of a MapReduce job.

There are few components in DES model e.g. - event list, clock, running state and event handler. The clock captures the time (in millisecond unit) spent till the last event being simulated by the system.

All the objects / entities have some variables in the execution state for capturing the MapReduce jobs' processes. When the running state of any entity changes, some events will be created, e.g. – a Mapper start event is initiated to change the task tracker's state from idle to busy at the moment when any task tracker is allocated to any map task.

There are some parameters for general event in MRTune such as - a list of event type event time and event ID. Some events have their own specific parameters such as - task ID and slot ID is the specialized parameter for Mapper Start and Mapper End events respectively, job ID is the specialized parameter for Job Start and Job End event. For creating the subsequent event as per the workflow and for modifying the respective variables in the execution state, there is also an event handler for each type of event.

It is the responsibility of the MRTune driver to read the input stipulations and to create both the list of future events and the running state. During a simulation, the very first event termed as Job Submission is created by the MRTune driver; then it calls the event handler for handling the event and in such way till the last event is

handled, the simulation will be kept executing. The clock keeps track of the entire procedure's time duration and at the end provides the total runtime of the job as an output; it also records the runtime duration of map and reduce phases.

For generating the key distribution, they provide two methods. In the first method, the real input data is divided into map partitions. Instead of storing the content of records, the way is as similar as the way Hadoop does for storing the statistics of key distribution. When the key distribution of the input can barely be estimated by means of a probabilistic model, this method is advantageous then. But, this method reduces the efficiency and will create overhead, particularly once the amount of records is vast. The second method attempts to produce the distribution of key by probabilistic computation on basis of the input specifications (e.g. the probabilistic model of key distribution, input data distribution type etc.). This method can efficiently store the key distribution information during approximation of the key distribution of the input.

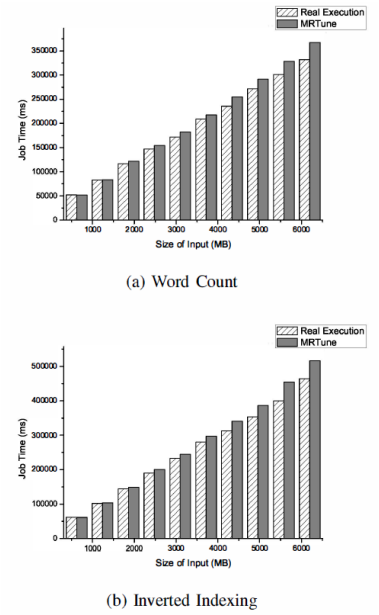


Fig 6: Evaluation result of large inputs having uniform distribution [10]

For the experimental setup they set both a Hadoop cluster and a MRTune cluster each having 5 nodes for real execution of MapReduce jobs. For evaluating the accuracy of MRTune, the MapReduce applications they implemented were: Inverted Indexing and Word Count. For both Inverted Indexing and Word Count, when the input size is lower than 5GB, the result demonstrated that the projected time of MRTune is near to the real accomplishment time (fig 6). The arithmetic error is still within 15% When the size of input goes beyond 5GB. From their evaluation, it was clearly concludable that MapReduce jobs with skewed input can be simulated by MRTune with high proficiency. After some testing they have shown that the number of reduce tasks need to be set to 25 to mitigate the failure influence for MRTune which is as similar as the real execution setting follows. This indicates that an effective performance tuning can be provided by MRTune.

The method used in MRTune is advantageous for the case when the key distribution of the input data can barely be estimated with the help of a probabilistic model. Nevertheless, particularly when the quantity of records is vast it will create overhead that brings down the proficiency of the system.

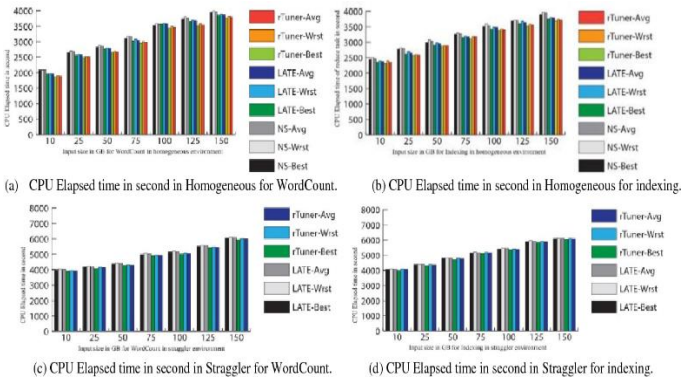
(b) rTuner: A Performance Enhancement of MapReduce Job [11]

Authors in this work has proposed a task scheduling algorithm for heterogeneous MapReduce environment called rTuner which enhances the task execution time of reduce phase. Since the reduce task is an expensive procedure (since this phase involves three stages, i.e. copy phase, shuffle phase and reduce phase), so not analyzing the underlying situation of the scheduling algorithms can negatively affect the performance of the rescheduling of a straggler reduce task. The rTuner doesn't explicitly estimate the job completion time of the whole MapReduce task rather it tries to analyze the straggling reason of the reduce tasks' and accordingly tries to tune the reduce task, which in turn improves the job completion time in a heterogeneous environment. Depending on the situation, if a reduce task becomes straggler then rTuner tries to reschedule it in another appropriate machine in the cluster. Such kind of straggler task is speculated in a fast node by the Longest Approximate Time to End (LATE) algorithm [21] too but authors say that they have found this algorithm and almost existing all task schedulers dealing with map and reduce task considering as similar task type; their designs are either based on only map task or map and reduce tasks in together, but not only reduce task. For scheduling the reduce tasks, LARTS [20] has to some extent considered the data locality but authors here say that they would like to discourage that data locality approach because that is unsuitable for heterogeneous environment.

The first step in the rTuner algorithm is to check whether a task is straggler / laggard or not; if the task is a laggard one then LATE algorithm [21] is used to calculate the probable required completion time of a task. The calculation goes like: $ProgressRate = ProgressScore/T$ where the T indicates time the task spends on. The next step of calculation is $TimeToComplete = (1 - ProgressScore)/ProgressRate$ where ProgressScore refers to the portion of input data read, ProgressRate indicates the rate of progress of a given task and $TimeToComplete$ refers to an estimation of completion time of a MapReduce task. So from this calculation if $TimeToComplete$ is known then administrators are able to tell whether the task is a laggard one or not and it can be easily decided whether it is required to reschedule a task or not (nevertheless, the $TimeToComplete$ is based on approximations and therefore, the anticipation of time completion can vary). After that there is a method in the algorithm called as **REASONFORSTRAGGLER**(RT) which checks the reason behind the task for being a straggler one and for identifying the reason, it sets some parameters of RT (Reduce Task) and makes the final decision of whether it would be beneficial to speculate the RT or not. Therefore, the straggler reduce task is rescheduled on a suitable node depending on the status and progress of the task.

Their benchmark result demonstrated that improvement of reduce tasks enhances the CPU elapsed time notably (fig 7). Furthermore, they showed the effectiveness of their proposed rTuner by wide-ranging experimentation on low-cost commodity hardware. The proposed rTuner algorithm and methodology calculates the metric TimeToComplete to estimate the execution time of any MapReduce task and improve the total job execution time appreciably, both in a

heterogeneous and/or homogeneous environment. Over the LATE algorithm in homogeneous and heterogeneous environment respectively, on an average the rTuner reduces the execution time by 86.86 seconds and 100.67 seconds and at the best situation by 142.44 seconds and 132.52.



(a) CPU Elapsed time in second in Homogeneous for WordCount. (b) CPU Elapsed time in second in Homogeneous for indexing.

(c) CPU Elapsed time in second in Straggler for WordCount. (d) CPU Elapsed time in second in Straggler for indexing.

Fig 7: Comparison of Reduce task for WordCount on LATE, No Speculation (NS) and rTuner in Best, Average and Worst case in Straggler and Homogeneous environment [11]

D. Abstract model using a Job Profiler based approach

(a) ARIA: Automatic Resource Inference and Allocation for MapReduce Environments [8]

In this paper, authors developed a job scheduler 'ARIA' for MapReduce environments which create a job profile that efficiently sums up the necessary performance characteristics of the original application during the map and reduce stages and based on that predicted time duration approximates the quantity of required resources for completing the job within the deadline.

They have discussed a few existing executional approaches of the same MapReduce job which they consider to be a function of the assigned map and reduce slots. From these, they mentioned their prime motive is to obtain a solitary job profile for capturing performance characteristics of the job accomplishment in various phases uniquely.

Their objective is creating a solid job profile which encompasses performance invariants not dependent on the quantity of resources given to the job over time. All stages of a specific job such as mapping, shuffling, sorting and reduce phases should be reflected while taking the decision. While the job is executing or parsed from the logs, this information can be gained from the counters at the job master. for estimating the job accomplishment time as a function of the size of the input dataset and the assigned resources, this model uses the performance boundaries for an execution period of a specified set of n tasks to be handled by k slots in MapReduce cluster settings. A simple greedy algorithm takes care of assigning the tasks to slots (e.g. assigning a piece of task to the slot with the least finishing time). They showed some mathematical calculations to define the lower and upper bound of the makespan where the difference between these two bounds are nothing but representing the scope of probable job accomplishment times due to scheduling and non-determinism. When the period of the lengthiest task is trivial as compared to the total makespan these bounds are mostly useful then.

They have considered a job j with a given profile either obtained from past job implementations or created from implementing this job in a staging environment. The job j was considered to be executed using a new dataset partitioned into some map tasks and reduce tasks with some particular allocated time slots to job j correspondingly. They assumed the average and maximum periods of map tasks as described by the profile of job J and predicted the lower and upper bounds on the duration of the entire map stage into mathematical equations using the Makespan Theorem. Similarly, they computed the lower and upper bounds of completion times for reduce and shuffle phases using this theorem. Consequently, the formulae for the lower and upper bounds of the overall completion time of job J was computed by them.

For validating their model, they used the data from the Trending Topics (TT) (http://trendingtopics.org): Wikipedia article on traffic log records accumulated every single hour in the month of September and October 2010. Using different numbers of map and reduce slots they predicted the job accomplishment times when the job is implemented on a dataset using a specific configuration setting. They say that they found less than 10% relative error between the measured job completion time and the predicted average time in almost all cases, and henceforth, the estimates are appropriate for confirming the job SLOs (fig 9).

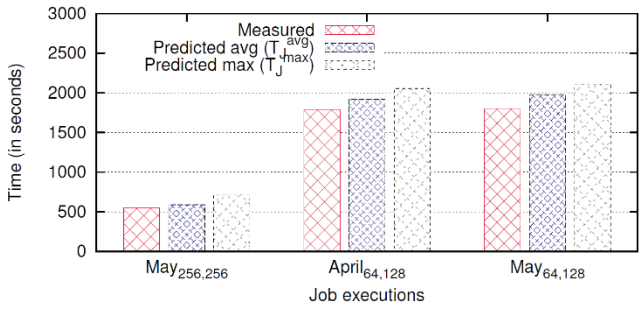


Fig 9: Comparison of measured job completion times and predicted time on different datasets [8]

After that they proposed a novel SLO-scheduler called ARIA which consists of five components. It supports a new API: where a job can be offered with a required job accomplishment deadline. The appropriate number of map and reduce slots to the job will then be estimated and allocated by the scheduler for meeting the necessary deadline.

The advantage of ARIA is that it offers a faster and effective model for a MR job with requirements of timing. Their later work [22] extended this scheme obtaining the scaling factors for predicting the job accomplishment time by running the application on a smaller data chunk while the initial application is employed for handling a larger dataset.

One weak point that I have felt is, ARIA has dealt with heterogeneity in a limited sense such as only having various number of cores across various machines within the cluster. Again, they have assumed that at each node there are uniform map waves; however, in a conventional heterogeneous cluster which have variable CPU, disk speed and memory, it might not be exact always.

(b) Starfish: A Self tuning System for Big Data Analytics [7]

Authors have proposed a self-tuning system Starfish, for big data analytics built on Hadoop. The advantage of Starfish is the adapting capability system workloads and to user needs for automatically providing good performance. It doesn't require the users to understand and manipulate the many tuning knobs in Hadoop cluster environment at all because Starfish's system architecture is led by labor on self-tuning database structures. in Hadoop more than 190 configuration parameters control the settings of the behavior of a MapReduce job.

Default values distributed with the system or defined by the system administrator are applied if the user does not stipulate parameter settings (Job, data, and cluster characteristics - these parameters lead to good settings) while submitting job.

Properties of the data in the files as well as the schema could have been remained unidentified; so, given the limited information available, the system should swiftly select the join implementation method from among 10+ conducts to perform joins in Starfish and the conforming settings of job parameters for configuration. unique optimization problems like those above is addressed by Starfish's Just-in-Time Optimizer for selecting efficient execution techniques for MapReduce jobs automatically. The online nature of decisions enforced on the optimizer by Hadoop's MADDER features are captured by "Just-in-time" optimizer which derives the help of the Profiler and the Sampler. A method called dynamic instrumentation is used by the Profiler for understanding performance representations, termed as job profiles, for unchanged MapReduce programs developed in Java and Python. Statistics about the input, intermediate, and output key-value pairs of a MapReduce task is collected efficiently by the Sampler. Sampling the execution of a MapReduce job for enabling the Profiler to gather estimated job profiles at a portion of the full job implementation cost is an exclusive feature of the Sampler.

Starfish also takes care of workflow level and workload level tuning into consideration. There are some concerns which further complicate the efficient scheduling of a Hadoop workflow e.g. (b) guaranteeing computing which is power proportional, (a) cascading re-execution avoidance capability under failure of node or corruption of data [11], and (c) load or cost imbalance adaptation in case of energy and time duration at the datacenter level [16]. Such concerns are addressed by Starfish's Workflow-aware Scheduler and the Data Manager and the What-if Engine. On the other hand, a comparable, but improved, group of workflows generated by Starfish's Workload Optimizer are passed off to the Workflow-aware Scheduler for implementation. Data-flow sharing, Materialization and Reorganization, are three significant types of optimization prospects existing at the workload level.

For showing the impact of various job configuration parameter settings (as discussed above) they have used WordCount (fig 10) and TeraSort MapReduce benchmarks to show the running time of jobs. It was observed that poor performance occurred due to the rule-of-thumb settings. The challenges a user faces while tuning the parameters by himself / herself is highlighted when the complexity of the surfaces arises and the failure of rules of thumb occurs. This process can be efficiently automated by Starfish's job-level tuning components— What-if Engine, Sampler, Profiler and Just-in-Time Optimizer.

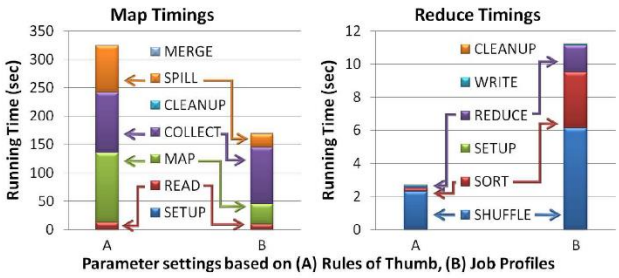


Fig 10: Map and reduce time breakdown for two WordCount jobs run with different settings of job configuration parameters [7]

One weak point that I found about Starfish is that, it has deliberated an analytical model which is based on a what-if engine for predicting execution time of a MR job for variable data / cluster size in the context of homogeneous clusters, so it might not work appropriately for heterogeneous clusters. Again, the lightweight Profiler for collecting comprehensive statistical information about MR jobs doesn't consider skewed key distribution of data.

(c) Improving MapReduce Performance in Heterogeneous Environments with Adaptive Task Tuning [23]

Since the homogeneous configuration of MapReduce tasks on heterogeneous machines might arise a significant cause of poor performance due to load imbalance, authors in this research proposed a task tuning method Ant, which finds the optimal settings for individual tasks and adapts to heterogeneous cluster configuration automatically. For large jobs with many rounds of a MapReduce task execution Ant works best by configuring tasks initially with randomly selected configurations (uses genetic functions during configuration) and progressively improvising configuration settings; it is done by repeatedly producing new settings, choosing the best performing configuration setting and eliminating the poor ones. The proposed task tuner Ant gives a little insight about predicting job completion time in heterogeneous cluster settings. Compared to stock Hadoop Ant increases the average job completion time by 23%, 11%, and 16% as shown in experimental results.

For multi-wave MapReduce applications, the proposed self-tuning approach Ant executes jobs which involve quite a lot of rounds of map and reduce tasks. It differs from the traditional MapReduce implementations in two ways from the architectural point of view: (1) here, matching with the abilities of the host machines, tasks though belonging to the same job, run with different configurations; (2) the configurations of any specific tasks are changed into quest for the optimum settings dynamically; which in return reduces the job completion time of any MapReduce task. Tasks with random configurations are spawned by Ant and then are executed in parallel. Task runtimes are collected by Ant upon task completion and according to the best-performing tasks, task settings are adjusted adaptively. Task configurations on different nodes congregate to the best settings after several rounds of tuning. Ant does not require any prior knowledge of MapReduce jobs task tuning since it starts with random settings and gradually improves with job completion; this method in turn comes up a random nature of predicting job completion time and execute individual task to accomplish within the minimal time as possible. Ant functions as: upon submission of a job to the JobTracker, a set of parameters are generated by the configuration optimizer randomly for the task-level configuration to

be initialized. After that randomly initialized tasks are sent to their respective TaskTrackers by the JobTracker.

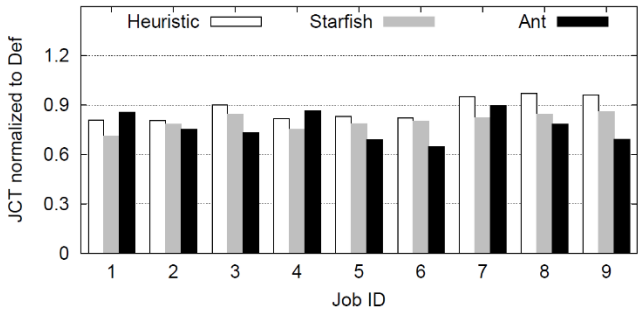


Fig 11: Job completion times on physical cluster [23]

The stages of tuning the tasks then match to the manifold waves of tasks implementation. Upon completion of a wave, for the next wave of execution good configurations are recommended by the task analyzer to the configuration optimizer. Until the job completes, this process is repeated.

They have compared Ant's dynamic configuration tuning capability with Heuristic and Starfish and showed the effect of this on the ability of reduction in various job completion times as shown in fig. The results as seen in the graphs showed that all these configuration settings enhanced the job execution times when compared with the performance achieved by Stock, Heuristic and Starfish by 31%, 20% and 14%, respectively (fig 11). Since Stock, Heuristic and Starfish all rely on a static task-level parameter setting Hadoop could beat all of them via dynamic task tuning and thus reducing job completion time. In the heterogeneous cluster such unified configuration is seemingly ineffective as proved. It is also notable from graph that Starfish performed better than Heuristic since Starfish has the capability of job profiling and the learning process of Starfish is also more accurate in case of capturing the characteristic of discrete jobs. Again, it is visible from the graphs that Ant is better in predicting and completing larger jobs rather than smaller jobs as compared to stock Hadoop. For e.g., J1, J4 and J7 jobs, Ant just slightly reduced the completion time than stock Hadoop whereas for jobs, e.g., J3, J6 and J9, Ant performed more effectively. Since, small jobs typically have quick execution times so the optimal configuration solutions can not immediately be found by the self-tuning process. Thus, such small jobs are less ideal to be predicted by Ant.

Moreover, it is revealed in fig that the job completion time of I/O intensive workloads for e.g. Grep and Terasort, is also reduced by Ant. Since Ant focuses on the task-level I/O operation parameter Tuning, thus I/O intensive workloads are more affected by it. Overall, better performance is achieved consistently by Ant than Starfish and Heuristic because of its capability of adaptively tuning task-level parameters in heterogeneous platforms with various workload preferences.

IV. PROPOSED APPROACH

After going through all these approaches, I felt that a simulation-based approach for estimating MR job duration can be quite efficient. For approximating completion period of a MapReduce job in a heterogeneous cluster environment, I would like to suggest a very general idea, I feel some kind of discrete event simulator-based approach along with linear regression with different amount of data size and different hardware configurations. At first, measurements on smaller data chunks for executing applications on heterogeneous cluster environment need to be collected including minimum one machine of individual category of hardware configuration. Then the measurements need to be fed into MR simulator for estimating the completion time of a map / reduce task for variable amount of data size encompassing variable combination of given categories of nodes with the help of linear regression. Various existing MapReduce benchmarks e.g. WordCount, TeraSort etc. can be used for testing the simulator performance.

To talk about a general idea about the architectural design of such a simulator, there could be a MR Job profile which would contain the quantity of map and reduce jobs and their period of execution for individual category of machine; a Cluster configuration manager for specifying the quantity of map/reduce slots executing on the nodes and the quantity of nodes and some kind of counter for measuring the Data size which would indicate the amount of data being processed. A priority queue could be implemented for seven event types to be maintained by the simulator: job arrivals, job departures, map tasks arrivals, map tasks departures, reduce tasks arrivals, reduce task departures and an event signing the end of the map phase. The simulator engine would dispatch events, keep track of the quantity of finished map and reduce MR jobs and the quantity of free slots and calculate the amount of time taken to finish each task. Finally, in a heterogeneous cluster the MR simulator will generate the predicted job execution time as the desired output.

In this way, the simulator would be trained using smaller data chunks and later larger data chunks need to be tested on it using linear regression method. The iterations would be performed as long as the results are fitting the straight line in the regression method as perfectly as possible.

V. CONCLUSION

Assuming and managing a MapReduce job's runtime performance ahead of time is a challenging task since there are several types of complications regarding the parameter configuration settings while running a job in a heterogeneous cluster environment. Repeatedly running an MR job applying various combinations of configuration settings and optimization techniques and then picking the one providing the shortest runtime is quite impractical because there maybe numerous possible combinations and the real execution of all them would be very much time consuming. Hence, any kind of method for estimating the runtime of any MapReduce job efficiently without the real implementation, just via utilizing the parameter configuration settings and input data of the cluster machines would be helpful. So, in this survey, I tried to review some of the research works where authors have tried to estimate the execution time of a MapReduce job in homogeneous and/or heterogeneous environment before the real accomplishment of that task. Several approaches have been adopted for accomplishing this research, which I have broadly categorized into four types namely - Simulation based approach, Job and task scheduling based approach, Task tuning based approach and Abstract model using a Job Profiler based approach. I have discussed the related works which fall into each of these categories from the perspective of their main focus of the work, a brief overview on their working mechanism and successful experimental evaluations; also discussed the strengths and weaknesses of these works from my point of views. Each of the approaches have their own efficient way

of estimating the task time duration and/or runtime performance. After going through these works, I felt simulation-based approach is quite established already, being simple but efficient, so I suggested a general idea that a simulator along with linear regression method can be developed where the simulator would be trained on smaller data chunks and later be tested on using larger data chunks, the iterations would be performed as long as the results are fitting the straight line in the regression method as perfectly as possible. Additionally, I found that tuning based approaches are quite new in this research area which as a future work, can be explored more by the researchers and hopefully they would come up with better and accurate results.

REFERENCES

[1] S. Hammoud, M. Li, Y. Liu, N. K. Alham, and Z. Liu, " MRSim: A discrete event-based MapReduce simulator," in Proceedings of Fuzzy Systems and Knowledge Discovery, ser. FSKD ' 10. IEEE, 2010, pp. 2993-2997.

[2] F. Teng, L. Yu, and F. Magoules, " SimMapReduce: A simulator for modeling MapReduce framework," in Proceedings of the 2011 Fifth FTRA International Conference on Multimedia and Ubiquitous Engineering, ser. MUE '11. IEEE, 2011, pp. 277-282.

[3] A. Verma, L. Cherkasova, and R. H. Campbell, "Play it again, SimMR!" in Proceedings of the 2011 IEEE International Conference on Cluster Computing, ser. CLUSTER '11. IEEE, 2011, pp. 253-26l.

[4] W. Kolberg, P. D. B. Marcos, J. C. S. Anjos, A. K. S. Miyazaki, C. R. Geyer, and L. B. Arantes, " MRSG - a MapReduce simulator over simgrid," Parallel Comput., vol. 39, no. 4-5, pp. 233-244.

[5] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," Commun. ACM, vol. 51, no. 1, pp. 107-113.

[6] Y. Liu, M. Li, N. K. Alham, and S. Hammoud, "HSim: A MapReduce simulator in enabling cloud computing," Future Gener. Comput. Syst., vol. 29, no. 1, pp. 300-308.

[7] H. Herodotou, H. Lim, G. Luo, N. Borisov, L. Dong, F. B. Cetin, and S. Babu, “Starfish: A Self-tuning System for Big Data Analytics,” in CIDR, 2011, pp. 261–272.

[8] A. Verma, L. Cherkasova, and R. H. Campbell, “ARIA: Automatic Resource Inference and Allocation for MapReduce Environments,” in Proceedings of the 8th ACM International Conference on Autonomic Computing, ser. ICAC ’11. New York, NY, USA: ACM, 2011, pp. 235–244.

[9] G. Wang, A. R. Butt, P. Pandey, and K. Gupta, "Using realistic simulation for performance analysis of MapReduce setups," in Proceedings of the 1st ACM workshop on Large-Scale system and application peiformance, ser. LSAP '09. ACM, 2009, pp. 19-26.

[10] Xibo Zhou, Wuman Luo, Haoyu Tan, “MRTune: A Simulator for Performance Tuning of MapReduce Jobs with Skewed Data”, in 2014 20th IEEE International Conference on Parallel and Distributed Systems (ICPADS), DOI: 10.1109/PADSW.2014.7097828, 30 April 2015, Hsinchu, Taiwan.

[11] Ripon Patgiri, Rajdeep Das, “rTuner: A Performance Enhancement of MapReduce Job”, in ICCMS 2018 Proceedings of the 10th International Conference on Computer Modeling and Simulation, Pages 176-183, DOI: 10.1145/3177457.3191710, Sydney, Australia, January 08 - 10, 2018.

[12] Rekha Singhal, Abhishek Verma, “Predicting Job Completion Time In Heterogeneous MapReduce Environments”, 2016 IEEE International Parallel and Distributed Processing Symposium Workshops, DOI: 10.1109/IPDPSW.2016.10, 04 August 2016, Chicago, IL, USA.

[13] "Apache Hadoop," http://hadoop.apache.org.

[14] "The Network Simulator - ns-2 "https://www.isi.edu/nsnam/ns/.

[15] A. Sulistio, c. S. Yeo, and R. Buyya, "Visual modeler for grid modeling and simulation (gridsim) toolkit," in Proceedings of the 2003 international conference on Computational science: Partlll, ser. ICCS'03. Springer, 2003, pp. 1123-1132.

[16] H. Casanova, A. Legrand, and M. Quinson, "Simgrid: A generic framework for large-scale distributed experiments," in Proceedings of the Tenth International Conference on Computer Modeling and Simulation, ser. UKSIM '08. IEEE, 2008, pp. 126-13l.

[17] "Simjava," http://www.dcs.ed.ac.uklhome/hase/simjava/.

[18] R. Buyya and M. Murshed, “Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing,” Concurrency and computation: practice and experience, vol. 14, no. 13-15, pp. 1175–1220, 2002.

[19] Ge Song, Zide Meng, Fabrice Huet, Fr´ed´eric Magoul`es, Lei Yu and Xuelian Lin, “A Hadoop MapReduce Performance Prediction Method”, 2013 IEEE 10th International Conference on High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing, DOI: 10.1109/HPCC.and.EUC.2013.118, 13-15 Nov. 2013, Zhangjiajie, China.

[20] Mohammad Hammoud and Majd F. Sakr. Locality-aware reduce task scheduling for MapReduce. In Proceedings of the 2011 IEEE Third International Conference on Cloud Computing Technology and Science, pages 570–576, Washington, DC, USA, 2011. IEEE Computer Society.

[21] Matei Zaharia, Andy Konwinski, Anthony D. Joseph, Randy Katz, and Ion Sto- ica. Improving MapReduce performance in heterogeneous environments. In Proceedings of the 8th USENIX conference on Operating systems design and implementation (OSDI’08), pages 29–42. USENIX Association, Berkeley, CA, USA, 2008.

[22] A. Verma, L. Cherkasova, and R. H. Campbell, “Resource provisioning framework for MapReduce jobs with performance goals,” in Proceedings of the 12th ACM/IFIP/USENIX International Conference on Middleware, 2011, pp. 165–186.

[23] Dazhao Cheng, Jia Rao, Yanfei Guo, Xiaobo Zhou, “Improving MapReduce Performance in Heterogeneous Environments with Adaptive Task Tuning”, in Proceedings of Middleware '14 Proceedings of the 15th International Middleware Conference, doi: 10.1145/2663165.2666089, Pages 97-108, Bordeaux, France — December 08 - 12, 2014.

[24] H. Herodotou and S. Babu, "Profiling, what-if analysis, and cost-based optimization of MapReduce programs," Proc. of the VLDB Endowment, vol. 4, no. 11, pp. 1111-1122, 201l.

[25] "Hadoop job tracker web ui," http://jobtracker-host:50030/.