Hadoop Performance Modeling for Job Estimation and Resource Provisioning

Mukhtaj Khan, Yong Jin, Maozhen Li, Yang Xiang and Changjun Jiang

Abstract- MapReduce has become a major computing model for data intensive applications. Hadoop, an open source implementation of MapReduce, has been adopted by an increasingly growing user community. Cloud computing service providers such as Amazon EC2 Cloud offer the opportunities for Hadoop users to lease a certain amount of resources and pay for their use. However, a key challenge is that cloud service providers do not have a resource provisioning mechanism to satisfy user jobs with deadline requirements. Currently, it is solely the user's responsibility to estimate the required amount of resources for running a job in the cloud. This paper presents a Hadoop job performance model that accurately estimates job completion time and further provisions the required amount of resources for a job to be completed within a deadline. The proposed model builds on historical job execution records and employs Locally Weighted Linear Regression (LWLR) technique to estimate the execution time of a job. Furthermore, it employs Lagrange Multipliers technique for resource provisioning to satisfy jobs with deadline requirements. The proposed model is initially evaluated on an in-house Hadoop cluster and subsequently evaluated in the Amazon EC2 Cloud. Experimental results show that the accuracy of the proposed model in job execution estimation is in the range of 94.97% and 95.51%, and jobs are completed within the required deadlines following on the resource provisioning scheme of the proposed model.

***Index Terms*— Cloud computing, Hadoop MapReduce, performance modeling, job estimation, resource provisioning**

I. Introduction

Many organizations are continuously collecting massive amounts of datasets from various sources such as the World Wide Web, sensor networks and social networks. The ability to perform scalable and timely analytics on these unstructured datasets is a high priority task for many enterprises. It has become difficult for traditional network storage and database systems to process these continuously growing datasets. MapReduce [1], originally developed by Google, has become a major computing model in support of data intensive applications. It is a highly scalable, fault-tolerant and data parallel model that automatically distributes the data and parallelizes the computation across a cluster of computers [2]. Among its implementations such as Mars[3], Phoenix[4], Dryad[5] and Hadoop [6], Hadoop has received a wide uptake by the community due to its open source nature [7][8][9][10].

One feature of Hadoop MapReduce is its support of public cloud computing that enables the organizations to utilize cloud services in a pay-as-you-go manner. This facility is beneficial to small and medium size organizations where the setup of a large scale and complex private cloud is not feasible due to financial constraints. Hence, executing Hadoop MapReduce applications in a cloud environment for big data analytics has become a realistic option for both the industrial practitioners and academic researchers. For example, Amazon has designed Elastic MapReduce (EMR) that enables users to run Hadoop applications across its Elastic Cloud Computing (EC2) nodes.

The EC2 Cloud makes it easier for users to set up and run Hadoop applications on a large-scale virtual cluster. To use the EC2 Cloud, users have to configure the required amount of resources (virtual nodes) for their applications. However, the EC2 Cloud in its current form does not support Hadoop jobs with deadline requirements. It is purely the user's responsibility to estimate the amount of resources to complete their jobs which is a highly challenging task. Hence, Hadoop performance modeling has become a necessity in estimating the right amount of resources for user jobs with deadline requirements. It should be pointed out that modeling Hadoop performance is challenging because Hadoop jobs normally involve multiple processing phases including three core phases (i.e. map phase, shuffle phase and reduce phase). Moreover, the first wave of the shuffle phase is normally processed in parallel with the map phase (i.e. overlapping stage) and the other waves of the shuffle phase are processed after the map phase is completed (i.e. non-overlapping stage).

To effectively manage cloud resources, several Hadoop performance models have been proposed [11][12][13][14]. However, these models do not consider the overlapping and non-overlapping stages of the shuffle phase which leads to an inaccurate estimation of job execution.

Mukhtaj Khan is with the Department of Electronic and Computer Engineering, Brunel University, Uxbridge, UB8 3PH, UK. Email: Mukhtaj.Khan@brunel.ac.uk.

Yong Jin is with the National Key Lab for Electronic Measurement Technology, North University of China, Taiyuan 030051, China. He is a Visiting Professor in the Department of Electronic and Computer Engineering, Brunel University, Uxbridge, UB8 3PH, UK. Email: Yong.Jin@brunel.ac.uk.

Maozhen Li is with the Department of Electronic and Computer Engineering, Brunel University, Uxbridge, UB8 3PH, UK. He is also with the Key Laboratory of Embedded Systems and Service Computing, Ministry of Education, Tongji University, Shanghai, 200092, China. Email: Maozhen.Li@brunel.ac.uk.

Changjun Jiang and Yang Xiang are with the Department of Computer Science & Technology, Tongji University, 1239 Siping Road, Shanghai 200092, China. Email: {cjjiang, shxiangyang}@tongji.edu.cn.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2015.2405552, IEEE Transactions on Parallel and Distributed Systems

Recently, a number of sophisticated Hadoop performance models are proposed [15][16][17][18]. Starfish [15] collects a running Hadoop job profile at a fine granularity with detailed information for job estimation and optimization. On the top of Starfish, Elasticiser [16] is proposed for resource provisioning in terms of virtual machines. However, collecting the detailed execution profile of a Hadoop job incurs a high overhead which leads to an overestimated job execution time. The HP model [17] considers both the overlapping and non-overlapping stages and uses simple linear regression for job estimation. This model also estimates the amount of resources for jobs with deadline requirements. CRESP [18] estimates job execution and supports resource provisioning in terms of map and reduce slots. However, both the HP model and CRESP ignore the impact of the number of reduce tasks on job performance. The HP model is restricted to a constant number of reduce tasks, whereas CRESP only considers a single wave of the reduce phase. In CRESP, the number of reduce tasks has to be equal to number of reduce slots. It is unrealistic to configure either the same number of reduce tasks or the single wave of the reduce phase for all the jobs. It can be argued that in practice, the number of reduce tasks varies depending on the size of the input dataset, the type of a Hadoop application (e.g. CPU intensive, or disk I/O intensive) and user requirements. Furthermore, for the reduce phase, using multiple waves generates better performance than using a single wave especially when Hadoop processes a large dataset on a small amount of resources. While a single wave reduces the task setup overhead, multiple waves improve the utilization of the disk I/O.

Building on the HP model, this paper presents an improved HP model for Hadoop job execution estimation and resource provisioning. The major contributions of this paper are as follows:

- The improved HP work mathematically models all the three core phases of a Hadoop job. In contrast, the HP work does not mathematically model the non-overlapping shuffle phase in the first wave.
- The improved HP model employs Locally Weighted Linear Regression (LWLR) technique to estimate the execution time of a Hadoop job with a varied number of reduce tasks. In contrast, the HP model employs a simple linear regress technique for job execution estimation which restricts to a constant number of reduce tasks.
- Based on job execution estimation, the improved HP model employs Langrage Multiplier technique to provision the amount of resources for a Hadoop job to complete within a given deadline.

The performance of the improved HP model is initially evaluated on an in-house Hadoop cluster and subsequently on Amazon EC2 Cloud. The evaluation results show that the improved HP model outperforms both the HP model and Starfish in job execution estimation with an accuracy of level in the range of 94.97% and 95.51%. For resource provisioning, 4 job scenarios are considered with a varied number of map slots and reduce slots. The experimental results show that the improved HP model is more economical in resource provisioning than the HP model.

The remainder of paper is organized as follows. Section II models job phases in Hadoop. Section III presents the improved HP model in job execution estimation and Section IV further enhances the improved HP model for resource provisioning. Section V first evaluates the performance of the improved HP model on an in-house Hadoop cluster and subsequently on Amazon EC2 Cloud. Section VI discusses a number of related works. Finally, Section VII concludes the paper and points out some future work.

II. Modeling Job Phases in Hadoop

Normally a Hadoop job execution is divided into a map phase and a reduce phase. The reduce phase involves data shuffling, data sorting and user-defined reduce functions. Data shuffling and sorting are performed simultaneously. Therefore, the reduce phase can be further divided into a shuffle (or sort) phase and a reduce phase performing user-defined functions. As a result, an overall Hadoop job execution work flow consists of a map phase, a shuffle phase and a reduce phase as shown in Fig.1. Map tasks are executed in map slots at a map phase and reduce tasks run in reduce slots at a reduce phase. Every task runs in one slot at a time. A slot is allocated with a certain amount of resources in terms of CPU and RAM. A Hadoop job phase can be completed in a single wave or multiple waves. Tasks in a wave run in parallel on the assigned slots.

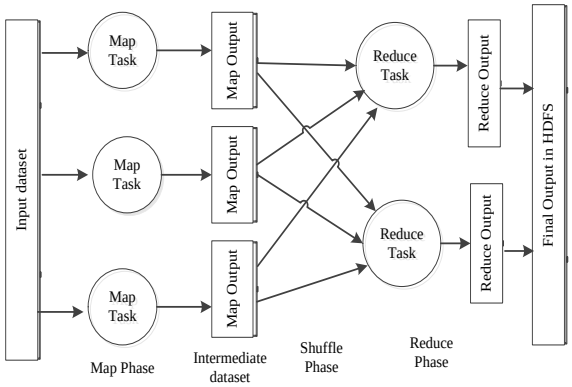

Fig.1. Hadoop job execution flow.

Herodotou presented a detailed set of mathematical models on Hadoop performance at a fine granularity [19]. For the purpose of simplicity, we only consider the three core phases (i.e. map phase, shuffle phase and reduce phase) in modeling the performance of Hadoop jobs. Table 1 defines the variables used in Hadoop job performance modeling.

A. Modeling Map Phase

In this phase, a Hadoop job reads an input dataset from Hadoop Distributed File System (HDFS), splits the input dataset into data chunks based on a specified size and then passes the data chunks to a user-define map function. The map function processes the data chunks and produces a map output. The map output is called intermediate data. The average map

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2015.2405552, IEEE Transactions on Parallel and Distributed Systems

output and the total map phase execution time can be computed using Eq.(1) and Eq.(2) respectively.

Table 1. Defined variables in modeling job phases.

Variables	Expressions
D_{m-avg}^{output}	The average output data size of a map task.
T_m^{total}	The total execution time of a map phase.
D_{m-avg}^{input}	The average input data size of a map task.
$M_{selectivity}$	The map selectivity which is the ratio of a map output to a map input.
N_m	The total number of map tasks.
T_m^{avg}	The average execution time of a map task.
N_m^{slot}	The total number of configured map slots.
D_{sh-avg}	The average size of a shuffled data.
T_{sh}^{total}	The total execution time of a shuffle phase.
N_r	The total number of reduce tasks.
T_{sh}^{avg}	The average execution duration of a shuffle task.
N_r^{slot}	The total number of configured reduce slots.
N_{sh}^{w1}	The total number of shuffle tasks that complete in the first wave.
N_{sh}^{w2}	The total number of shuffle tasks that complete in other waves.
T_{w1}^{avg}	The average execution time of a shuffle task that completes in the first wave.
T_{w2}^{avg}	The average execution time of a shuffle task that completes in other waves.
D_{r-avg}^{output}	The average output data size of a reduce task.
T_r^{total}	The total execution time of a reduce phase.
D_{r-avg}^{input}	The average input size of a reduce task.
$R_{selectivity}$	The reduce selectivity which is the ratio of a reduce output to a reduce input.
T_r^{avg}	The average execution time of a reduce task.

$$D_{m-avg}^{output} = D_{m-avg}^{input} \times M_{selectivity} \quad (1)$$

$$T_m^{total} = \frac{T_m^{avg} \times N_m}{N_m^{slot}} \quad (2)$$

B. Modeling Shuffle Phase

In this phase, a Hadoop job fetches the intermediate data, sorts it and copies it to one or more reducers. The shuffle tasks and sort tasks are performed simultaneously, therefore, we generally consider them as a shuffle phase. The average size of shuffled data can be computed using Eq.(3).

$$D_{Sh-avg} = \frac{D_{m-avg}^{output} \times N_m}{N_r} \quad (3)$$

If $N_r \le N_r^{slot}$, then the shuffle phase will be completed in a single wave. The total execution time of a shuffle phase can be computed using Eq.(4).

$$T_{sh}^{total} = \frac{T_{sh}^{avg} \times N_r}{N_r^{slot}} \quad (4)$$

Otherwise, the shuffle phase will be completed in multiple waves and its execution time can be computed using Eq.(5).

$$T_{sh}^{total} = \frac{(T_{w1}^{avg} \times N_{sh}^{w1}) + (T_{w2}^{avg} \times N_{sh}^{w2})}{N_r^{slot}} \quad (5)$$

C. Modeling Reduce Phase

In this phase, a job reads the sorted intermediate data as input and passes to a user-defined reduce function. The reduce function processes the intermediate data and produces a final output. In general, the reduce output is written back into the HDFS. The average output of the reduce tasks and the total execution time of the reduce phase can be computed using Eq.(6) and Eq.(7) respectively.

$$D_{r-avg}^{output} = D_{r-avg}^{input} \times R_{selectivity} \quad (6)$$

$$T_r^{total} = \frac{T_r^{avg} \times N_r}{N_r^{slot}} \quad (7)$$

III. An Improved HP Performance Model

As also mentioned before, Hadoop jobs have three core execution phases – map phase, shuffle phase and reduce phase. The map phase and the shuffle phase can have overlapping and non-overlapping stages. In this section, we present an improved HP model which takes into account both overlapping stage and non-overlapping stage of the shuffle phase during the execution of a Hadoop job. We consider single Hadoop jobs without logical dependencies.

A. Design Rationale

A Hadoop job normally runs with multiple phases in a single wave or in multiple waves. If a job runs in a single wave then all the phases will be completed without overlapping stages as shown in Fig.2.

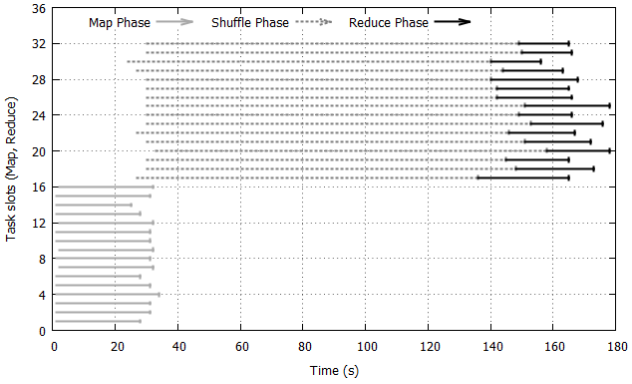

Fig.2. A Hadoop job running in a single wave (16 map tasks and 16 reduce tasks).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2015.2405552, IEEE Transactions on Parallel and Distributed Systems

However, if a job runs in multiple waves, then the job will be progressed through both overlapping (parallel) and non-overlapping (sequential) stages among the phases as show in Fig.3.

In the case of multiple waves, the first wave of the shuffle phase starts immediately after the first map task completes. Furthermore, the first wave of the shuffle phase continues until all the map tasks complete and all the intermediate data is shuffled and sorted. Thus, the first wave of the shuffle phase is progressed in parallel with the other waves of the map phase as shown in Fig.3. After completion of the first wave of the shuffle phase, the reduce tasks start running and produce output. Afterwards, these reduce slots will become available to the shuffle tasks running in other waves. It can be observed from Fig.3 that the shuffle phase takes longer to complete in the first wave than in other waves. In order to estimate the execution time of a job in multiple waves, we need to estimate two sets of parameters for the shuffle phase - the average and the maximum durations of the first wave, together with the average and the maximum durations of the other waves. Moreover, there is no significant difference between the durations of the map tasks running in non-overlapping and overlapping stages due to the equal size of data chunks. Therefore, we only estimate one set of parameters for the map phase which are the average and the maximum durations of the map tasks. The reduce tasks run in a non-overlapping stage, therefore we only estimate one set of parameters for the reduce phase which are the average and the maximum durations of the reduce tasks. Finally, we aggregate the durations of all the three phases to estimate the overall job execution time.

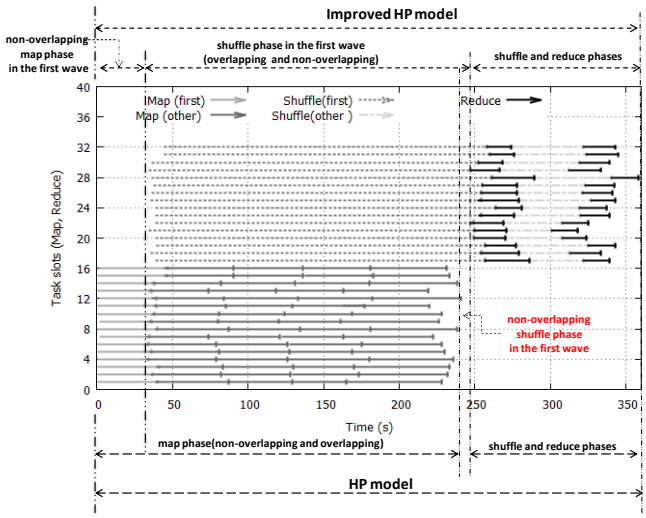

Fig.3. A Hadoop job running in multiple waves (80 map tasks, 32 reduce tasks).

It should be pointed out that Fig.3 also shows the differences between the HP model and the improved model in Hadoop job modeling. The HP work mathematically models the whole map phase which includes the non-overlapping stage of the map phase and the stage overlapping with the shuffle phase, but it does not provide any mathematical equations to model the non-overlapping stage of the shuffle phase in the first wave.

Whereas the improved HP work mathematically models the non-overlapping map phase in the first wave, and the shuffle phase in the first wave which includes both the stage overlapping with the map phase and the non-overlapping stage. This can be reflected in the mathematical equations of the improved HP model which are different from the HP model.

B. *Mathematical Expressions*

In this section, we present the mathematical expressions of the improved HP work in modeling a Hadoop job which completes in multiple waves. Table 2 defines the variables used in the improved model.

Table 2. Defined variables in the improved HP model.

Variables	Expressions
T_{m-w1}^{low}	The lower bound duration of the map phase in the first wave (non-overlapping).
T_{m-w1}^{up}	The upper bound duration of the map phase in the first wave (non-overlapping).
N_m^{w1}	The number of map tasks that complete in the first wave of the map phase.
N_m^{w2}	The number of map tasks that complete in other waves of the map phase.
T_m^{max}	The maximum execution time of a map task.
T_{sh-w1}^{low}	The lower bound duration of the shuffle phase in the first wave (overlapping with the map phase).
T_{sh-w1}^{up}	The upper bound duration of the shuffle phase in the first wave (overlapping with the map phase).
$T_{sh\ -\ w1}^{avg}$	The average execution time of a shuffle task that completes in the first wave of the shuffle phase.
$T_{sh\ -\ w1}^{\max}$	The maximum execution time of a shuffle task that completes in the first wave of the shuffle phase.
$T_{sh\ -\ w2}^{low}$	The lower bound duration of the shuffle phase in other waves (non-overlapping)
$T_{sh\ -\ w2}^{up}$	The upper bound duration of the shuffle phase in other waves (non-overlapping).
$T_{sh\ -\ w2}^{avg}$	The average execution time of a shuffle task that completes in other waves of the shuffle phase.
$T_{sh\ -\ w2}^{max}$	The maximum execution time of a shuffle task that completes in other waves of the shuffle phase.
T_r^{low}	The lower bound duration of the reduce phase.
T_r^{up}	The upper bound duration of the reduce phase.
T_r^{max}	The maximum execution time of a reduce task.
T_{job}^{low}	The lower bound execution time of a Hadoop job.
T_{job}^{up}	The upper bound execution time of a Hadoop job.
T_{job}^{avg}	The average execution time of a Hadoop job.

In practice, job tasks in different waves may not complete exactly at the same time due to varied overhead in disk I/O operations and network communication. Therefore, the improved HP model estimates the lower bound and the upper bound of the execution time for each phase to cover the best-case and the worse-case scenarios respectively.

We consider a job that runs in both non-overlapping and overlapping stages. The lower bound and the upper bound of the map phase in the first wave which is a non-overlapping stage can be computed using Eq.(8) and Eq.(9) respectively.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2015.2405552, IEEE Transactions on Parallel and Distributed Systems

$$T_{m-w1}^{low} = \frac{T_m^{avg} \times N_m^{w1}}{N_m^{slot}} \quad (8)$$

$$T_{m-w1}^{up} = \frac{T_m^{\max} \times N_m^{w1}}{N_m^{slot}} \quad (9)$$

In the overlapping stage of a running job, the map phase overlaps with the shuffle phase. Specifically, the tasks running in other waves of the map phase run in parallel with the tasks running in the first wave of the shuffle phase. As the shuffle phase always completes after the map phase which means that the shuffle phase takes longer than the map phase, therefore we use the duration of the shuffle phase in the first wave to compute the lower bound and the upper bound of the overlapping stage of the job using Eq.(10) and Eq.(11) respectively.

$$T_{sh-w1}^{low} = \frac{T_{sh-w1}^{avg} \times N_{sh}^{w1}}{N_r^{slot}} \quad (10)$$

$$T_{sh-w1}^{up} = \frac{T_{sh-w1}^{\max} \times N_{sh}^{w1}}{N_r^{slot}} \quad (11)$$

In other waves of the shuffle phase, the tasks run in a non-overlapping stage. Hence, the lower bound and the upper bound of the non-overlapping stage of the shuffle phase can be computed using Eq.(12) and Eq.(13) respectively.

$$T_{sh-w2}^{low} = \frac{T_{sh-w2}^{avg} \times N_{sh}^{w2}}{N_r^{slot}} \quad (12)$$

$$T_{sh-w2}^{up} = \frac{T_{sh-w2}^{\max} \times N_{sh}^{w2}}{N_r^{slot}} \quad (13)$$

The reduce tasks start after completion of the shuffle tasks. Therefore, the reduce tasks complete in a non-overlapping stage. The lower bound and the upper bound of the reduce phase can be computed using Eq.(14) and Eq.(15) respectively.

$$T_r^{low} = \frac{T_r^{avg} \times N_r}{N_r^{slot}} \quad (14)$$

$$T_r^{up} = \frac{T_r^{\max} \times N_r}{N_r^{slot}} \quad (15)$$

As a result, the lower bound and upper bound of the execution time of a Hadoop job can be computed by combining the execution durations of all the three phases using Eq.(16) and Eq.(17) respectively.

$$T_{job}^{low} = T_{m-w1}^{low} + T_{sh-w1}^{low} + T_{sh-w2}^{low} + T_r^{low} \quad (16)$$

$$T_{job}^{up} = T_{m-w1}^{up} + T_{sh-w1}^{up} + T_{sh-w2}^{up} + T_r^{up} \quad (17)$$

By substituting the values in Eq.(16) and Eq.(17), we have

$$T_{job}^{low} = \frac{T_m^{avg} \times N_m^{w1}}{N_m^{slot}} + \frac{T_{sh-w1}^{avg} \times N_{sh}^{w1}}{N_r^{slot}} + \frac{T_{sh-w2}^{avg} \times N_{sh}^{w2}}{N_r^{slot}} + \frac{T_r^{avg} \times N_r}{N_r^{slot}} \quad (18)$$

$$T_{job}^{up} = \frac{T_m^{\max} \times N_m^{w1}}{N_m^{slot}} + \frac{T_{sh-w1}^{\max} \times N_{sh}^{w1}}{N_r^{slot}} + \frac{T_{sh-w2}^{\max} \times N_{sh}^{w2}}{N_r^{slot}} + \frac{T_r^{\max} \times N_r}{N_r^{slot}} \quad (19)$$

Finally, we take an average of Eq.(18) and Eq.(19) to estimate the execution time of a Hadoop job using Eq.(20).

$$T_{job}^{avg} = \frac{T_{job}^{low} + T_{job}^{up}}{2} \quad (20)$$

C. Job Execution Estimation

In the previous section, we have presented the mathematical expressions of the improved HP model. The lower bound and the upper bound of a map phase can be computed using Eq.(8) and Eq.(9) respectively. However, the durations of the shuffle phase and the reduce phase have to be estimated based on the running records of a Hadoop job.

When a job processes an increasing size of an input dataset, the number of map tasks is proportionally increased while the number of reduce tasks is specified by a user in the configuration file. The number of reduce tasks can vary depending on user's configurations. When the number of reduce tasks is kept constant, the execution durations of both the shuffle tasks and the reduce tasks are linearly increased with the increasing size of the input dataset as considered in the HP model. This is because the volume of an intermediate data block equals to the total volume of the generated intermediate data divided by the number of reduce tasks. As a result, the volume of an intermediate data block is also linearly increased with the increasing size of the input dataset. However, when the number of reduce tasks varies, the execution durations of both the shuffle tasks and the reduce tasks are not linear to the increasing size of an input dataset.

In either the shuffle phase or the reduce phase, we consider the tasks running in both overlapping and non-overlapping stages. Unlike the HP model, the improved model considers a varied number of reduce tasks. As a result, the durations of both the shuffle tasks and the reduce tasks are nonlinear to the size of an input dataset. Therefore, instead of using a simple linear regression as adopted by the HP model, we apply Locally Weighted Linear Regression (LWLR) [20][21] in the improved

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2015.2405552, IEEE Transactions on Parallel and Distributed Systems

model to estimate the execution durations of both the shuffle tasks and the reduce tasks.

LWLR is an instance-based nonparametric function, which assigns a weight to each instance x according to its Euclidean distance from the query instance x_q. LWLR assigns a high weight to an instance x which is close to the query instance x_q and a low weight to the instances that are far away from the query instance x_q. The weight of an instance can be computed using a Gaussian function as illustrated in Eq.(21).

$$w_k = \exp\left(-\frac{(distance(x_k, x_q)^2)}{2h^2}\right), (k = 1,2,3,....,m) \quad (21)$$

where,

- w_k is the weight of the training instance at location k.
- x_k is the training instance at location k.
- m is the total number of the training instances.
- h is a smoothing parameter which determines the width of the local neighborhood of the query instance.

The value of h is crucial to LWLR. Users have the option of using a new value of h for each estimation or a single global value of h. However, finding an optimal value for h is a challenging issue itself [22]. In the improved HP model, a single global value of h is used to minimize the estimated mean square errors.

In the improved HP model, LWLR is used to estimate the durations of both the shuffle tasks and the reduce tasks. First, we estimate T_{sh-w1}^{avg}, which is the average duration of the shuffle tasks running in the first wave of the shuffle phase. To estimate T_{sh-w1}^{avg}, we define a matrix $X \in \Re^{m \times n}$ whose rows contain the training dataset $x_1, x_2, x_3, ..., x_m$ and n is the number of feature variables which is set to 2 (i.e. the size of an intermediate dataset and the number of reduce tasks). We define a vector $Y = \lfloor y_1, y_2 ..., y_m \rfloor$ of dependent variables that are used for the average durations of the shuffle tasks. For example, y_i represents the average execution time of the shuffle task that corresponds to the training instance of x_i. We define another matrix X_q whose rows are query instances. Each query instance x_q contains both the size of the intermediate dataset d_{new} and the number of reduce tasks r_{new} of a new job. We calculate d_{new} based on the average input data size of a map task, the total number of map tasks and the map selectivity metric which is $d_{new} = D_{m-input}^{avg} \times N_m \times M_{selectivity}$.

For the estimation of T_{sh-w1}^{avg}, we calculate the weight for each training instance using Eq. (21) and then compute the parameter β using Eq. (22) which is the coefficient of LWLR.

$$\beta = (X^T \times W \times X)^{-1} (X^T \times W \times Y) \quad (22)$$

Here $W = diag(w_k)$ is the diagonal matrix where all the non-diagonal cells are 0 values. The value of a diagonal cell is increased when the distance between a training instance and the query instance is decreased.

Finally, the duration of a new shuffle task running in the first wave of the shuffle phase can be estimated using Eq. (23).

$$T_{sh-w1}^{avg} = X_q \times \beta \quad (23)$$

Similarly, the durations of T_{sh-w1}^{max}, T_{sh-w2}^{avg}, T_{sh-w2}^{max}, T_r^{avg} and T_r^{max} can be estimated.

The estimated values of both the shuffle phase and the reduce phase are used in the improved HP model to estimate the overall execution time of a Hadoop job when processing a new input dataset. Fig.4 shows the overall architecture of the improved HP model, which summarizes the work of the improved HP model in job execution estimation. The boxes in gray represent the same work presented in the HP model. It is worth noting that the improved HP model works in an offline mode and estimates the execution time of a job based on the job profile.

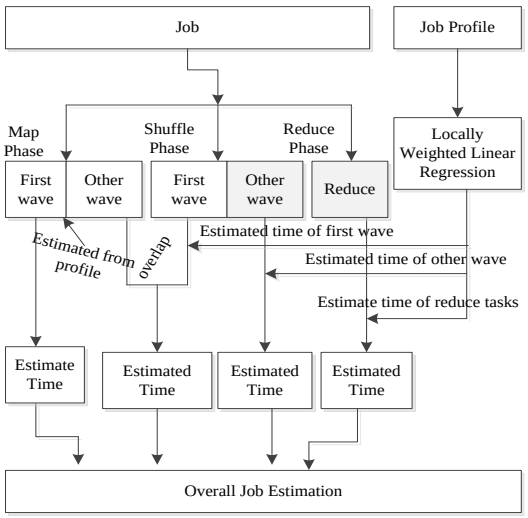

Fig.4. The architecture of the improved HP model.

IV. Resource Provisioning

The improved HP model presented in Section III can estimate the execution time of a Hadoop job based on the job execution profile, allocated resources (i.e. map slots and reduce slots), and the size of an input dataset. The improved HP model is further enhanced to estimate the amount of resources for Hadoop jobs with deadline requirements.

Consider a deadline t for a job that is targeted at the lower bound of the execution time. To estimate the number of map slots and reduce slots, we consider the non-lapping map phase in the first wave, the map phase in other waves together with the overlapped shuffle phase in the first wave, the shuffle phase in other waves and the reduce phase. Therefore we simplify

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2015.2405552, IEEE Transactions on Parallel and Distributed Systems

Eq.(18) into Eq.(24) with a modification of Eq.(10) for resource estimation.

$$\frac{a}{m}+\frac{b}{m+r}+\frac{c}{r}+\frac{d}{r}=t \quad (24)$$

where

- $t = T_{job}^{low}$
- $a = T_{m}^{avg} \times N_{m}^{w1}$
- $b = (T_{m}^{avg} \times N_{m}^{w2}) + (T_{sh-w1}^{avg} \times N_{sh}^{w1})$
- $c = T_{sh-w2}^{avg} \times N_{sh}^{w2}$
- $d = T_{r}^{avg} \times N_{r}$
- $m = N_{m}^{slot}$
- $r = N_{r}^{slot}$

The method of Lagrange Multipliers [23] is used to estimate the amounts of resources (i.e. map slots and the reduce slots) for a job to complete within a deadline. Lagrange Multipliers is an optimization technique in multivariable calculus that minimizes or maximizes the objective function subject to a constraint function. The objective function is $f(m,r)=m+r$ and the constraint function is $g(m,r)=0$, where $g(m,r)=\frac{a}{m}+\frac{b}{m+r}+\frac{c}{r}+\frac{d}{r}-t$ is derived from Eq.(24). To minimize the objective function, the Lagrangian function is expressed as Eq.(25).

$$L(m,r,\lambda)=f(m,r)+\lambda g(m,r) \quad (25)$$

where λ is the Lagrange Multiplier. We take partial differentiation of Eq.(25) with respect to m, r, λ, we have

$$\frac{\partial L}{\partial m}=1-\frac{\lambda a}{m^2}-\frac{\lambda b}{(m+r)^2}=0 \quad (26)$$

$$\frac{\partial L}{\partial r}=1-\frac{\lambda b}{(m+r)^2}-\frac{\lambda (c+d)}{r^2}=0 \quad (27)$$

$$\frac{\partial L}{\partial \lambda}=\frac{a}{m}+\frac{b}{m+r}+\frac{c}{r}+\frac{d}{r}-t=0 \quad (28)$$

Solving Eq.(26), Eq.(27), and Eq.(28) simultaneously for m and r, we have

$$m=\frac{\lambda}{x+1}\left[a(x+1)^2+\frac{ab}{c+d}\right]^{\frac{1}{2}}$$

$$r=\frac{\lambda}{x(x+1)}\left[a(x+1)^2+\frac{ab}{c+d}\right]^{\frac{1}{2}}$$

where

$$\lambda=\frac{1}{t\left[(x+1)a+\frac{ab}{c+d}\right]^{\frac{1}{2}}}\left[a(x+1)+bx+(c+d)(x(x+1))\right]$$

and $x=\sqrt{\frac{a}{c+d}}$

Here, the values of m and r are the numbers of map slots and reduce slots respectively. As we have targeted at the lower bound of the execution time of a job, the estimated amount of resources might not be sufficient for the job to complete within the deadline. This is because the lower bound corresponds to the best-case scenario which is hardly achievable in a real Hadoop environment. Therefore, we also target at the upper bound of the execution time of a job. For this purpose we use Eq.(19) as a constraint function in Lagrange Multipliers, and apply the same method as applied to Eq.(18) to compute the values of both m and r. In this case, the amounts of resources might be overestimated for a job to complete within the deadline. This is because the upper bound corresponds to the worst-case execution of a job. As a result, an average amount of resources between the lower and the upper bounds might be more sensible for resource provisioning for a job to complete within a deadline.

V. Performance Evaluation

The performance of the improved HP model was initially evaluated on an in-house Hadoop cluster and subsequently on Amazon EC2 cloud. In this section, we present the evaluation results. First, we give a brief description on the experimental environments that were used in the evaluation process.

A. Experimental Setup

We set up an in-house Hadoop cluster using an Intel Xeon server machine. The specifications and configurations of the server are shown in Table 3. We installed Oracle Virtual Box and configured 8 Virtual Machines (VMs) on the server. Each VM was assigned with 4 CPU cores, 8GB RAM and 150GB hard disk storage. We used Hadoop-1.2.1 and configured one VM as the Name Node and the remaining 7 VMs as Data Nodes. The Name Node was also used as a Data Node. The data block size of the HDFS was set to 64MB and the replication level of data block was set to 2. Two map slots and two reduce slots were configured on each VM. We employed two typical MapReduce applications, i.e. the WordCount application and the Sort application which are CPU intensive and IO intensive applications respectively. The teraGen application was used to generate input datasets of different sizes.

The second experimental Hadoop cluster was setup on Amazon EC2 Cloud using 20 m1.large instances. The specifications of the m1.large are shown in Table 3. In this cluster, we used Hadoop-1.2.1 and configured one instance as Name Node and other 19 instances as Data Nodes. The Name Node was also used as a Data Node. The data block size of the HDFS was set to 64MB and the replication level of data block

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2015.2405552, IEEE Transactions on Parallel and Distributed Systems

was set to 3. Each instance was configured with one map slot and one reduce slot.

Table 3: Experimental Hadoop cluster.

Intel Xeon Server 1	CPU	40 cores
	Processor	2.27GHz
	Hard disk	2TB
	Connectivity	100Mbps Ethernet LAN
	Memory	128GB
Amazon m1.large instance	vCPU	2
	Hard disk	420GB
	Memory	7.5GB
Software	Operating System	Ubuntu 12.04 TLS
	JDK	1.6
	Hadoop	1.2.1
	Oracle Virtual Box	4.2.8
	Starfish	0.3.0

B. *Job Profile Information*

We run both the WordCount and the Sort applications on the two Hadoop clusters respectively and employed Starfish to collect the job profiles. For each application running on each cluster, we conducted 10 tests. For each test, we run 5 times and took the average durations of the phases. Table 4 and Table 5 present the job profiles of the two applications that run on the EC2 Cloud.

Table 4: The job profile of the WordCount application in EC2 environment.

Data size (GB)	Map tasks	Map task duration (s)		Shuffle duration(s) in the first wave (overlapping)		Shuffle duration(s) in other waves (non-overlapping)		Reduce duration (s)	
		Avg.	Max	Avg.	Max	Avg.	Max	Avg.	Max
5	80	12	23	69	73	20	22	18	25
10	160	12	24	139	143	26	29	20	32
15	240	13	23	212	215	38	44	23	35
20	320	13	23	274	278	34	39	17	26
25	400	11	25	346	350	41	47	20	27
30	480	11	24	408	411	47	57	22	41
35	560	12	27	486	489	59	71	27	42
40	640	12	24	545	549	45	52	19	30
45	720	11	23	625	629	50	58	20	32
50	800	14	24	693	696	55	65	23	37

Table 5: The profile of the Sort application in EC2 environment.

Data Size (GB)	Map tasks	Map task duration (s)		Shuffle duration(s) in the first wave (overlapping)		Shuffle duration(s) in other waves (non-overlapping)		Reduce duration (s)	
		Avg.	Max	Avg.	Max	Avg.	Max	Avg.	Max
5	80	11	15	48	50	15	18	13	24
10	160	12	24	108	111	23	32	30	42
15	240	12	20	161	165	31	41	50	68
20	320	12	22	218	221	29	35	44	63
25	400	13	22	277	281	37	63	57	73
30	480	13	33	325	330	42	56	75	112
35	560	12	27	375	378	55	82	87	132
40	640	13	26	424	428	52	74	71	104
45	720	13	26	484	488	63	94	97	128
50	800	13	29	537	541	71	102	104	144

C. *Evaluating the Impact of the Number of Reduce Tasks on Job Performance*

In this section we evaluate the impact of the number of reduce tasks on job performance. We run both the WordCount and the Sort applications on the in-house Hadoop cluster with a varied number of reduce tasks. The experimental results are shown in Fig.5 and Fig.6 respectively. For both applications, it can be observed that when the size of the input dataset is small (e.g. 10GB), using a small number of reduce tasks (e.g. 16) generates less execution time than the case of using a large number of reduce tasks (e.g. 64). However, when the size of the input dataset is large (e.g. 25GB), using a large number of reduce tasks (e.g. 64) generates less execution time than the case of using a small number of reduce tasks (e.g. 16). It can also be observed that when the size of the input dataset is small (e.g. 10GB or 15GB), using a single wave of reduce tasks (i.e. the number of reduce tasks is equal to the number of reduce slots which is 16) performs better than the case of using multiple waves of reduce tasks (i.e. the number of reduce tasks is larger than the number of reduce slots). However, when the size of the input dataset is large (e.g. 25GB), both the WordCount and the Sort applications perform better in the case of using multiple waves of reduce tasks than the case of using a single wave of reduce tasks. While a single wave reduces the task setup overhead on a small dataset, multiple waves improve the utilization of the disk I/O on a large dataset. As a result, the number of reduce tasks affects the performance of a Hadoop application.

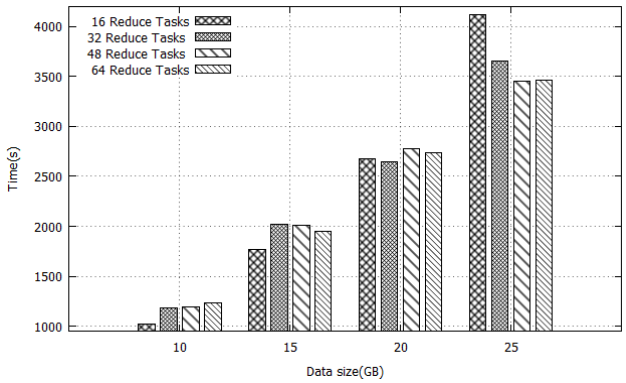

Fig.5. The performance of the WordCount application with a varied number of reduce tasks.

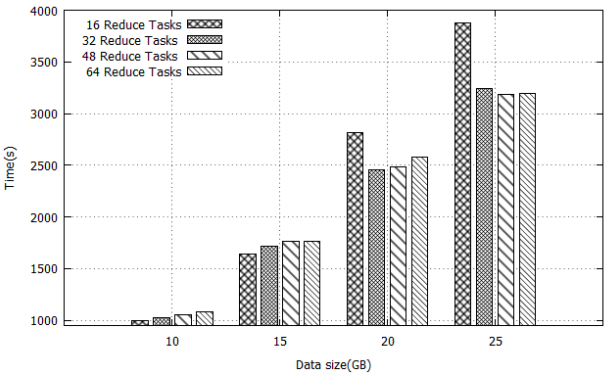

Fig.6.The performance of the Sort application with a varied number of reduce tasks.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2015.2405552, IEEE Transactions on Parallel and Distributed Systems

D. Estimating the Execution Times of Shuffle Tasks and Reduce Tasks

Both the WordCount and the Sort applications processed a dataset on the in-house Hadoop cluster with a varied number of reduce tasks from 32 to 64. The size of the dataset was varied from 2GB to 20GB. Both applications also processed another dataset from 5GB to 50GB on the EC2 Cloud with the number of reduce tasks varying from 40 to 80. The LWLR regression model presented in Section III.C was employed to estimate the execution times of both the shuffle tasks and the reduce tasks of a new job. The estimated values were used in Eq.(18) and Eq.(19) to estimate the overall job execution time.

Fig.7 and Fig.8 show respectively the estimated execution times of both the shuffle tasks and the reduce tasks for both applications running on the Hadoop cluster in EC2. Similar evaluation results were obtained from both applications running on the in-house Hadoop cluster. We can observe that the execution times of both the shuffle tasks (non-overlapping stage) and reduce tasks are not linear to the size of an input dataset. It should be noted that the execution times of the shuffle tasks that run in an overlapping stage are linear to the size of an input dataset because the durations of these tasks depend on the number of map waves, as shown in Table 4 and Table 5.

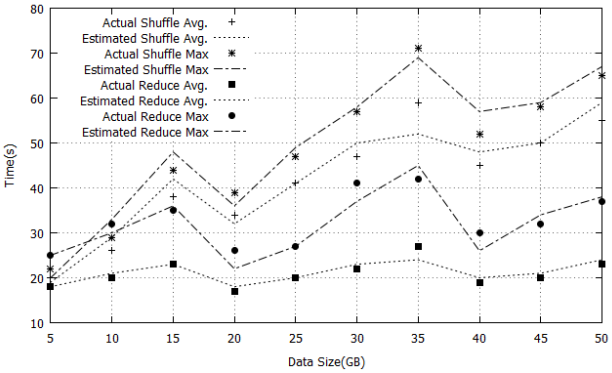

Fig.7.The estimated durations of both the shuffle phase (non-overlapping stage) and the reduce phase in the WordCount application. The points represent the actual execution time and dashed lines represent the estimated durations.

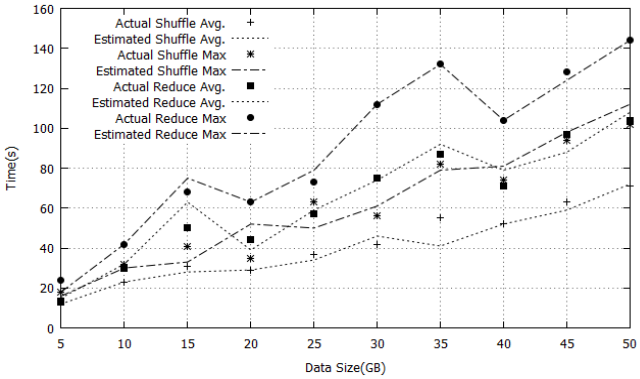

Fig.8. The estimated durations of both the shuffle phase (non-overlapping stage) and the reduce phase in the Sort application. The points represent the actual execution time and dashed lines represent the estimated duration.

E. Job Execution Estimation

A number of experiments were carried out on both the in-house Hadoop cluster and the EC2 Cloud to evaluate the performance of the improved HP model. First, we evaluated the performance of the improved HP model on the in-house cluster and subsequently evaluated the performance of the model on the EC2 Cloud.

For the in-house cluster, the experimental results obtained from both the WordCount and the Sort applications are shown in Fig.9 and Fig.10 respectively. From these two figures we can observe that the improved HP model outperforms the HP model in both applications. The overall accuracy of the improved HP model in job estimation is within 95% compared with the actual job execution times, whereas the overall accuracy of the HP model is less than 89% which uses a simple linear regression. It is worth noting that the HP model does not generate a straight line in performance as shown in [17]. This is because a varied number of reduce tasks was used in the tests whereas the work presented in [17] used a constant number of reduce tasks.

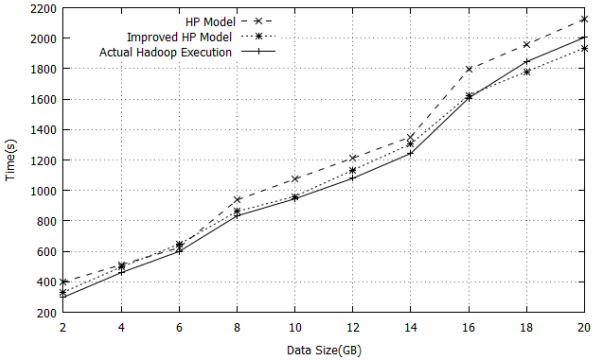

Fig.9. The performance of the improved HP model in job estimation of running the WordCount application on the in-house cluster.

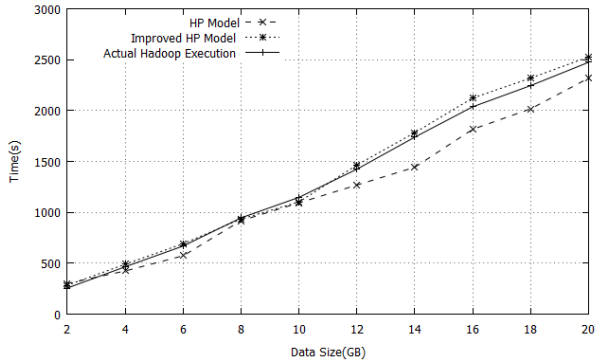

Fig.10. The performance of the improved HP model in job estimation of running the Sort application on the in-house cluster.

Next, we evaluated the performance of the improved HP model on the EC2 Cloud. The experimental results in running both applications are shown in Fig.11 and Fig.12 respectively. It can be observed that the improved HP model also performs better than the HP model. The overall accuracy of the improved HP model in job estimation is over 94% compared with the actual job execution times, whereas the overall accuracy of the HP model is less than 88%. The HP model performs better on small datasets but its accuracy level is decreased to 76.15% when the dataset is large (e.g. 40GB). The reason is that the HP model employs a simple linear regression which cannot accurately estimate the execution times of the shuffle tasks and the reduce tasks which are not linear to the size of an input dataset.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2015.2405552, IEEE Transactions on Parallel and Distributed Systems

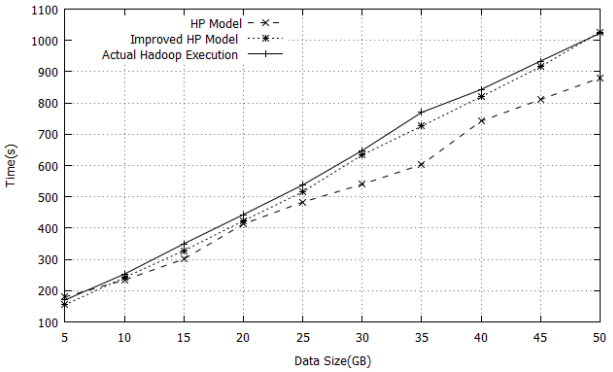

Fig.11. The performance of the improved HP model in job estimation of running the WordCount application on the EC2 Cloud.

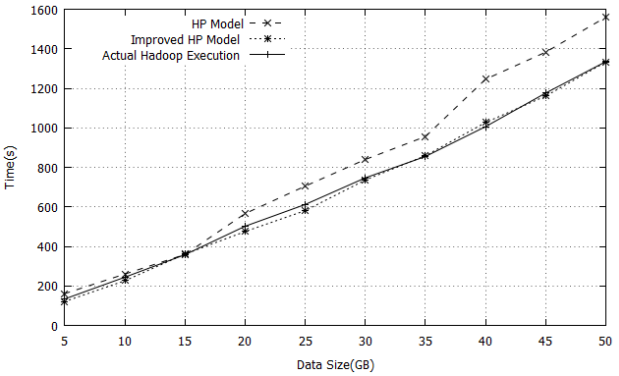

Fig.12. The performance of the improved HP model in job estimation of running the Sort application on the EC2 Cloud.

Finally, we compared the performance of the improved HP model in job estimation with that of both Starfish and the HP model collectively. Fig.13 and Fig.14 show the comparison results of the three models running the two applications on the EC2 Cloud respectively.

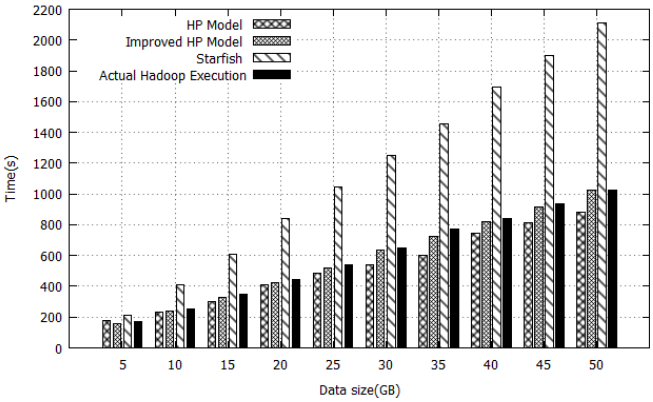

Fig.13. A performance comparison among the improved HP model, the HP model and Starfish in running the WordCount application on the EC2 Cloud.

It can be observed that the improved HP model produces the best results in job estimation for both applications. Starfish performs better than the HP model on the Sort application in some cases as shown in Fig.14. However, Starfish overestimates the job execution times of the WordCount application as shown in Fig.13. This is mainly due to the high overhead of Starfish in collecting a large set of profile information of a running job. The Starfish profiler generates a high overhead for CPU intensive applications like WordCount because the Starfish uses Btrace to collect job profiles which requires additional CPU cycles [16]. Starfish performs better on the Sort application because Sort is less CPU-intensive than the WordCount application.

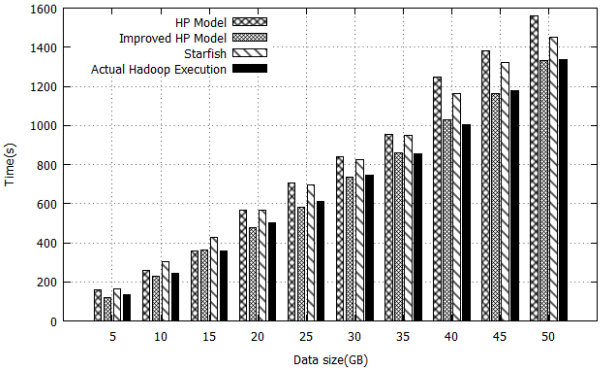

Fig.14. A performance comparison among the improved HP model, the HP model and Starfish in running the Sort application on the EC2 Cloud.

We have validated the LWLR regression model in job execution estimation using 10-fold cross validation technique. We considered the execution of an entire job with three phases (i.e. map phase, shuffle phase and reduce phase). The mean absolute percentage errors of the WordCount application and the Sort application are 2.37% and 1.89% respectively which show high generalizability of the LWLR in job execution estimation. Furthermore, the R-squared values of the two applications are 0.9986 and 0.9979 respectively which reflects the goodness of fit of LWLR.

F. Resource Provisioning

In this section, we present the evaluation results of the improved HP model in resource provisioning using the in-house Hadoop cluster. We considered 4 scenarios as shown in Table 6. The intention of varying the number of both map slots and reduce slots from 1 to 4 was twofold. One was to evaluate the impact of the resources available on the performance of the improved HP model in resource estimation. The other was to evaluate the performance of the Hadoop cluster in resource utilization with a varied number of map and reduce slots.

Table 6: Scenario configurations.

Scenarios	Number of map slots on each VM	Number of reduce slots on each VM
1	1	1
2	2	2
3	3	3
4	4	4

To compare the performance of the improved HP model with the HP model in resource estimation in the 4 scenarios, we employed the WordCount application as a Hadoop job processing 9.41GB input dataset. In each scenario, we set 7 completion deadlines for the job which are 920, 750, 590, 500, 450, 390 and 350 in seconds. We first built a job profile in each scenario. We set a deadline for the job, and employed both the HP model and the improved HP model to estimate the amount

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2015.2405552, IEEE Transactions on Parallel and Distributed Systems

of resources (i.e. the number of map slots and the number of reduce slots). We then assigned the estimated resources to the job using the in-house Hadoop cluster and measured the actual upper bound and the lower bound execution durations. We took an average of an upper bound and a lower bound and compared it with the given deadline. It should be noted that for resource provisioning experiments we configured 16VMs to satisfy the requirement of a job. Therefore, we employed another Xeon server machine with the same specification of the first server as shown in Table 3. We installed the Oracle Virtual Box and configured 8 VMs on the second server. Fig.15 to Fig.18 show the results in resource provisioning of the 4 scenarios respectively.

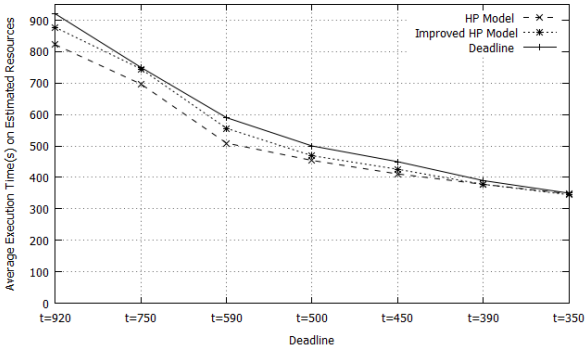

Fig.15. Resource provisioning in Scenario 1.

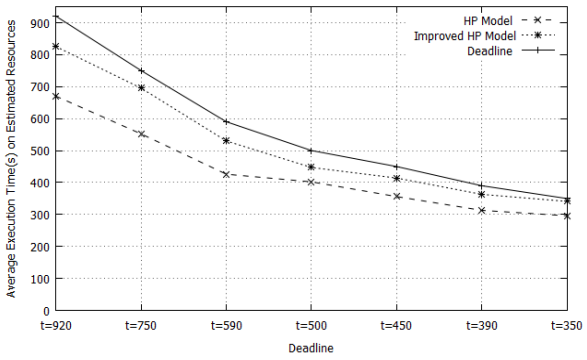

Fig.16. Resource provisioning in Scenario 2.

From the 4 scenarios we can see that overall the improved HP model slightly performs better than the HP model in resource provisioning due to its high accuracy in job execution estimation. Both models perform well in the first two scenarios especially in Scenario 1 where the two models generate a near optimal performance. However, the two models over-provision resources in both Scenario 3 and Scenario 4 especially in the cases where the job deadlines are large. The reason is that when we built the training dataset for resource estimation, we run all the VMs in the tests. One rationale was that we consider the worst cases in resource provisioning to make sure all the user job deadlines would be met. However, the overhead incurred in running all the VMs was high and included in resource provisioning for all the jobs. As a result, for jobs with large deadlines, both models over estimate the overhead of the VMs involved. Therefore, both models over-provision the amounts of resources for jobs with large deadlines which can be completed using a small number of VMs instead of all the VMs.

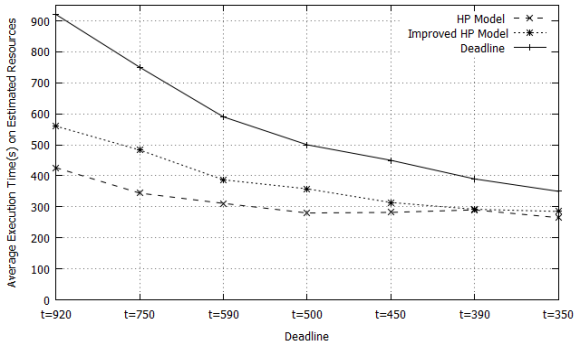

Fig.17. Resource provisioning in Scenario 3.

It is worth noting that all the job deadlines are met in the 4 scenarios except the last job deadline in Scenario 4 where t=350. This could be caused by the communication overhead incurred among the VMs running across the two server machines. Although both the improved HP model and the HP model include communication overhead in resource provisioning when the training dataset was built, they only consider static communication overhead. It can be expected that the communication overhead varies from time to time due to the dynamic nature of a communication network.

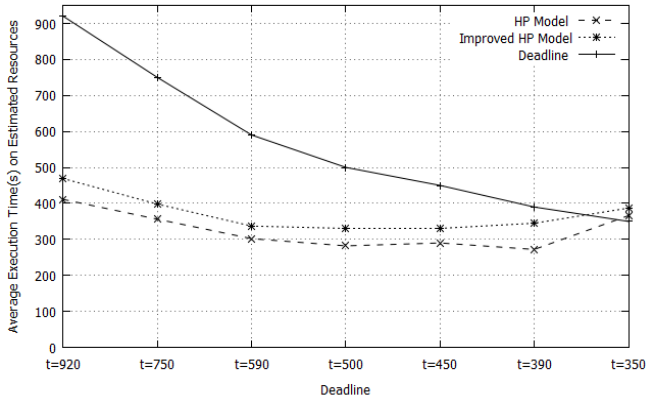

Fig.18. Resource provisioning in Scenario 4.

Table 7 summarizes the resources estimated by both the HP model and the improved HP model in the 4 scenarios. It can be observed that the HP model recommends more resources in terms of map slots, especially in Scenario 3. This is because the HP model largely considers the map slots in resource provisioning. As a result, the jobs following the HP model are completed quicker than the jobs following the improved HP model but with larger gaps from the given deadlines. Therefore, the improved HP model is more economical than the HP model in resource provisioning due to its recommendations of less map slots.

VI. RELATED WORK

Hadoop performance modeling is an emerging topic that deals with job optimization, scheduling, estimation and resource provisioning. Recently this topic has received a great attention from the research community and a number of models have been proposed.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2015.2405552, IEEE Transactions on Parallel and Distributed Systems

Table 7: The amounts of resources estimated by the HP model and the improved HP model.

	Scenario 1		Scenario 2		Scenario 3		Scenario 4	
Deadlines	HP model (m, r)	Improved HP model (m, r)	HP model (m, r)	Improved HP model (m, r)	HP model (m, r)	Improved HP model (m, r)	HP model (m, r)	Improved HP model (m, r)
920	(5,1)	(4,4)	(8,2)	(6,5)	(18,4)	(11,5)	(20,5)	(19,5)
750	(5,2)	(5,5)	(9,3)	(7,6)	(22,5)	(12,6)	(24,6)	(23,6)
590	(7,2)	(6,6)	(12,4)	(9,8)	(28,5)	(16,8)	(30,6)	(29,8)
500	(8,2)	(7,7)	(14,4)	(10,9)	(33,6)	(19,9)	(36,7)	(34,10)
450	(9,3)	(8,8)	(15,5)	(11,10)	(37,7)	(21,10)	(40,8)	(39,10)
390	(10,3)	(9,9)	(18,5)	(13,11)	(42,8)	(24,12)	(46,9)	(44,11)
350	(11,3)	(10,10)	(20,6)	(14,13)	(47,9)	(27,13)	(51,10)	(49,13)

Legends: m= map slots, r= reduce slots

Morton et al. proposed the parallax model [24] and later the ParaTimer model [25] that estimate the performance of the Pig parallel queries, which can be translated into series of MapReduce jobs. They use debug runs of the same query on input data samples to predict the relative progress of the map and reduce phases. This work is based on simplified suppositions that the durations of the map tasks and the reduce tasks are the same for a MapReduce application. However, in reality, the durations of the map tasks and the reduce tasks cannot be the same because the durations of these tasks are depended on a number of factors. More importantly, the durations of the reduce tasks in overlapping and non-overlapping stages are very different. Ganapathi et al. [26] employed a multivariate Kernel Canonical Correlation Analysis (KCCA) regression technique to predict the performance of Hive query. However, their intention was to show the applicability of KCCA technique in the context of MapReduce.

Kadirvel et al. [27] proposed Machine Learning (ML) techniques to predict the performance of Hadoop jobs. However, this work does not have a comprehensive mathematical model for job estimation. Lin et al. [11] proposed a cost vector which contains the cost of disk I/O, network traffic, computational complexity, CPU and internal sort. The cost vector is used to estimate the execution durations of the map and reduce tasks. It is challenging to accurately estimate the cost of these factors in a situation where multiple tasks compete for resources. Furthermore, this work is only evaluated to estimate the execution times of the map tasks and no estimations on reduce tasks are presented. The later work [12] considers resource contention and tasks failure situations. A simulator is employed to evaluate the effectiveness of the model. However, simulator base approaches are potentially error-prone because it is challenging to design an accurate simulator that can comprehensively simulate the internal dynamics of complex MapReduce applications.

Jalaparti et al. [13] proposed a system called Bazaar that predicts Hadoop job performance and provisions resources in term of VMs to satisfy user requirements. The work presented in [14] uses the Principle Component Analysis technique to optimize Hadoop jobs based on various configuration parameters. However, these models leave out both the overlapping and non-overlapping stages of the shuffle phase.

There is body of work that focuses on optimal resource provisioning for Hadoop jobs. Tian et al. [28] proposed a cost model that estimates the performance of a job and provisions the resources for the job using a simple regression technique. Chen et al. [18] further improved the cost model and proposed CRESP which employs the brute-force search technique for provisioning the optimal cluster resources in term of map slots and reduce slots for Hadoop jobs. The proposed cost model is able to predict the performance of a job and provisions the resources needed. However, in the two models , the number of reduce tasks have to be equal to the number of reduce slots which means that these two models only consider a single wave of the reduce phase. It is arguable that a Hadoop job performs better when multiple waves of the reduce phase are used in comparison with the use of a single, especially in situations where a small amount of resources is available but processing a large dataset. Lama et al. [29] proposed AROMA, a system that automatically provisions the optimal resources and optimizes the configuration parameters of Hadoop for a job to achieve the service level objectives. AROMA uses clustering techniques to group the jobs with similar behaviors. AROMA uses Support Vector Machine to predict the performance of a Hadoop job and uses a pattern search technique to find the optimal set of resources for a job to achieve the required deadline with a minimum cost. However, AROMA cannot predict the performance of a Hadoop job whose resource utilization pattern is different from any previous ones. More importantly, AROMA does not provide a comprehensive mathematical model to estimate a job execution time as well as optimal configuration parameter values of Hadoop.

There are a few other sophisticated models such as [15][16][17][30] that are similar to the improve HP model in the sense that they use the previous executed job profiles for performance prediction. Herodotou et al. proposed Starfish [15] which collects the past executed jobs profile information at a

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2015.2405552, IEEE Transactions on Parallel and Distributed Systems

fine granularity for job estimation and automatic optimization. On the top of the Starfish, Herodotou et al. proposed Elasticiser [16] which provisions a Hadoop cluster resources in term of VMs. However, collecting detailed job profile information with a large set of metrics generates an extra overhead, especially for CPU-intensive applications. As a result, Starfish overestimate the execution time of a Hadoop job. Verma et al. [30] presented the ARIA model for job execution estimations and resource provisioning. The HP model [17] extends the ARIA mode by adding scaling factors to estimate the job execution time on larger datasets using a simple linear regression. The work presented in [31] divides the map phase and reduce phase into six generic sub-phases (i.e. read, collect, spill, merge, shuffle and write), and uses a regression technique to estimate the durations of these sub-phases. The estimated values are then used in the analytical model presented in [30] to estimate the overall job execution time. In [32] , Zhang et al. employed the bound-based approach [30] in heterogeneous Hadoop cluster environments.

It should be pointed out that the aforementioned models are limited to the case that they only consider a constant number of the reduce tasks. As a result, the impact of the number of reduce tasks on the performance of a Hadoop job is ignored. The improved HP model considers a varied number of reduce tasks and employs a sophisticated LWLR technique to estimate the overall execution time of a Hadoop job.

VII. Conclusion

Running a MapReduce Hadoop job on a public cloud such as Amazon EC2 necessitates a performance model to estimate the job execution time and further to provision a certain amount of resources for the job to complete within a given deadline. This paper has presented an improved HP model to achieve this goal taking into account multiple waves of the shuffle phase of a Hadoop job. The improved HP model was initially evaluated on an in-house Hadoop cluster and subsequently evaluated on the EC2 Cloud. The experimental results showed that the improved HP model outperforms both Starfish and the HP model in job execution estimation. Similar to the HP model, the improved HP model provisions resources for Hadoop jobs with deadline requirements. However, the improved HP model is more economical in resource provisioning than the HP model.

Both models over-provision resources for user jobs with large deadlines in the cases where VMs are configured with a large number of both map slots and reduce slots. One future work would be to consider dynamic overhead of the VMs involved in running the user jobs to minimize resource over-provisioning. Currently the improved HP model only considers individual Hadoop jobs without logical dependencies. Another future work will be to model multiple Hadoop jobs with execution conditions.

Acknowledgement

This research is partially supported by the 973 project on Network Big Data Analytics funded by the Ministry of Science and Technology, China. No. 2014CB340404.

References

[1] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[2] R. Lämmel, "Google's MapReduce programming model — Revisited," *Sci. Comput. Program.*, vol. 70, no. 1, pp. 1–30, 2008.

[3] B. He, W. Fang, Q. Luo, N. K. Govindaraju, and T. Wang, "Mars: a MapReduce framework on graphics processors," in *Proceedings of the 17th international conference on Parallel architectures and compilation techniques - PACT '08*, 2008, p. 260.

[4] K. Taura, T. Endo, K. Kaneda, and A. Yonezawa, "Phoenix: a parallel programming model for accommodating dynamically joining/leaving resources," in *SIGPLAN Not.*, 2003, vol. 38, no. 10, pp. 216–229.

[5] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly, "Dryad: distributed data-parallel programs from sequential building blocks," *ACM SIGOPS Oper. Syst. Rev.*, vol. 41, no. 3, pp. 59–72, Mar. 2007.

[6] "Apache Hadoop." [Online]. Available: http://hadoop.apache.org/. [Accessed: 21-Oct-2013].

[7] D. Jiang, B. C. Ooi, L. Shi, and S. Wu, "The Performance of MapReduce: An In-depth Study," *Proc. VLDB Endow.*, vol. 3, no. 1–2, pp. 472–483, Sep. 2010.

[8] U. Kang, C. E. Tsourakakis, and C. Faloutsos, "PEGASUS: Mining Peta-scale Graphs," *Knowl. Inf. Syst.*, vol. 27, no. 2, pp. 303–325, May 2011.

[9] B. Panda, J. S. Herbach, S. Basu, and R. J. Bayardo, "PLANET: Massively Parallel Learning of Tree Ensembles with MapReduce," *Proc. VLDB Endow.*, vol. 2, no. 2, pp. 1426–1437, Aug. 2009.

[10] A. Pavlo, E. Paulson, and A. Rasin, "A comparison of approaches to large-scale data analysis," in *SIGMOD '09 Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, 2009, pp. 165–178.

[11] X. Lin, Z. Meng, C. Xu, and M. Wang, "A Practical Performance Model for Hadoop MapReduce," in *Cluster Computing Workshops (CLUSTER WORKSHOPS), 2012 IEEE International Conference on*, 2012, pp. 231–239.

[12] X. Cui, X. Lin, C. Hu, R. Zhang, and C. Wang, "Modeling the Performance of MapReduce under Resource Contentions and Task Failures," in *Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on*, 2013, vol. 1, pp. 158–163.

[13] J. Virajith, B. Hitesh, C. Paolo, K. Thomas, and R. Antony, "Bazaar: Enabling Predictable Performance in Datacenters," Microsoft Reasearch, MSR-TR-2012-38, [Online].Available: http://research.microsoft.com/apps/pubs/default.aspx?id=162192.

[14] H. Yang, Z. Luan, W. Li, D. Qian, and G. Guan, "Statistics-based Workload Modeling for MapReduce," in *Parallel and Distributed Processing Symposium Workshops PhD Forum (IPDPSW), 2012 IEEE 26th International*, 2012, pp. 2043–2051.

[15] 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 *In CIDR*, 2011, pp. 261–272.

[16] H. Herodotou, F. Dong, and S. Babu, "No One (Cluster) Size Fits All: Automatic Cluster Sizing for Data-intensive Analytics," in *Proceedings of the 2nd ACM Symposium on Cloud Computing (SOCC '11)*, 2011, pp. 1–14.

[17] 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.

[18] K. Chen, J. Powers, S. Guo, and F. Tian, "CRESP: Towards Optimal Resource Provisioning for MapReduce Computing in Public Clouds," *IEEE Transcation Parallel Distrib. Syst.*, vol. 25, no. 6, pp. 1403 – 1412, 2014.

[19] H. Herodotou, "Hadoop Performance Models," 2011. [Online]. Available: http://www.cs.duke.edu/starfish/files/hadoop-models.pdf. [Accessed: 22-Oct-2013].

[20] W. S. Cleveland and S. J. Delvin, "Locally Weighted Regression: An Approach to Regression Analysis by Local Fitting.," *J. Am. Stat. Assoc.*, vol. 83, no. 403, pp. 596–610, 1988.

[21] M. Rallis and M. Vazirgiannis, "Rank Prediction in graphs with Locally Weighted Polynomial Regression and EM of Polynomial Mixture Models," in *Advances in Social Networks Analysis and Mining (ASONAM), 2011 International Conference on*, 2011, pp. 515–519.

[22] J. Fan and I. Gijbels, *Local Polynomial Modelling and Its Applications: Monographs on Statistics and Applied Probability 66*. CRC Press, 1996.

[23] A. George, W. Hans, and H. Frank, *Mathematical Methods for Physicists*, 6th ed. Orlando, FL: Academic Press, 2005, p. 1060.

[24] K. Morton, A. Friesen, M. Balazinska, and D. Grossman, "Estimating the progress of MapReduce pipelines," in *Data Engineering (ICDE), 2010 IEEE 26th International Conference on*, 2010, pp. 681–684.

[25] K. Morton, M. Balazinska, and D. Grossman, "ParaTimer: A Progress Indicator for MapReduce DAGs," in *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, 2010, pp. 507–518.

[26] A. Ganapathi, Y. Chen, A. Fox, R. Katz, and D. Patterson, "Statistics-driven workload modeling for the Cloud," in *Data Engineering Workshops (ICDEW), 2010 IEEE 26th International Conference on*, 2010, pp. 87–92.

[27] S. Kadirvel and J. A. B. Fortes, "Grey-Box Approach for Performance Prediction in Map-Reduce Based Platforms," in *Computer Communications and Networks (ICCCN), 2012 21st International Conference on*, 2012, pp. 1–9.

[28] F. Tian and K. Chen, "Towards Optimal Resource Provisioning for Running MapReduce Programs in Public Clouds," in *2011 IEEE 4th International Conference on Cloud Computing*, 2011, pp. 155–162.

[29] P. Lama and X. Zhou, "AROMA: Automated Resource Allocation and Configuration of Mapreduce Environment in the Cloud," in *Proceedings of the 9th International Conference on Autonomic Computing*, 2012, pp. 63–72.

[30] A. Verma, L. Cherkasova, and R. H. Campbell, "ARIA: automatic resource inference and allocation for MapReduce environments.," in *8th ACM International conference on autonomic computing*, 2011, pp. 235–244.

[31] Z. Zhang, L. Cherkasova, and B. T. Loo, "Benchmarking Approach for Designing a Mapreduce Performance Model," in *Proceedings of the 4th ACM/SPEC International Conference on Performance Engineering*, 2013, pp. 253–258.

[32] Z. Zhang, L. Cherkasova, and B. T. 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.

Mukhtaj Khan received his MSc in Mobile Computer System from Staffordshire University, UK in 2006. He is currently a PhD student in the School of Engineering and Design at Brunel University, UK. The PhD study is sponsored by Abdul Wali Khan University Mardan, Pakistan. His research interests are focused on high performance computing for big data analysis.

Yong Jin received the PhD from North University of China in 2013. He is an Associate Professor in the School of Information and Communication Engineering at North University of China. He is also a Visiting Professor in the School of Engineering and Design at Brunel University, UK. His research interests are in the areas of image processing, online inspections and big data analytics.

Maozhen Li is currently a Professor in the Department of Electronic and Computer Engineering at Brunel University London, UK. He received the PhD from Institute of Software, Chinese Academy of Sciences in 1997. He was a Post-Doctoral Research Fellow in the School of Computer Science and Informatics, Cardiff University, UK in 1999-2002. His research interests are in the areas of high performance computing (grid and cloud computing), big data analytics and intelligent systems. He is on the Editorial Boards of Computing and Informatics journal and journal of Cloud Computing: Advances, Systems and Applications. He has over 100 research publications in these areas. He is a Fellow of the British Computer Society.

Yang Xiang received the PhD degree from Harbin Institute of Technology, China in 1999. He completed his Post-Doctoral research at Dalian University of Technology, China in 2003. He is now a Professor in the Department of Computer Science and Technology, Tongji University, Shanghai, China. His research interests are in the areas of machine learning, semantic web, and big data analytics.

Changjun Jiang received the PhD degree from the Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 1995 and conducted postdoctoral research at the Institute of Computing Technology, Chinese Academy of Sciences, in 1997. Currently, he is a professor with the Department of Computer Science and Engineering, Tongji University, Shanghai. He is also a council member of China Automation Federation and Artificial Intelligence Federation, the director of Professional Committee of Petri Net of China Computer Federation, and the vice director of Professional Committee of Management Systems of China Automation Federation. He was a visiting professor of Institute of Computing Technology, Chinese Academy of Science; a research fellow of the City University of Hong Kong, Kowloon, Hong Kong; and an information area specialist of Shanghai Municipal Government. His current areas of research are concurrent theory, Petri net, and formal verification of software, concurrency processing and intelligent transportation systems. He is a senior member of the IEEE.