(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
9 August 2012 (09.08.2012)

WIPO | PCT

(10) International Publication Number
WO 2012/105969 A1

(51) International Patent Classification:
*G06F 9/44* (2006.01) *G06F 15/16* (2006.01) *G06F 9/30* (2006.01)

(21) International Application Number:
PCT/US2011/023438

(22) International Filing Date:
2 February 2011 (02.02.2011)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant *(for all designated States except US)*: **HEWLETT-PACKARD DEVELOPMENT COMPANY, L.P.** [US/US]; 11445 Compaq Center Drive W., Houston, Texas 77070 (US).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: **VERMA, Abhishek** [IN/US]; 1501 Page Mill Road, Palo Alto, California 94304-1100 (US). **CHERKASOVA, Ludmila** [US/US]; 1501 Page Mill Road, Palo Alto, California 94304-1101 (US).

(74) Agents: **DAKIN, Lloyd E.** et al.; Hewlett-Packard Development Company, L.P., Intellectual Property Administration, Mail Stop 35 3404 E. Harmony Road, Fort Collins, Colorado 80528 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PE, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

**Declarations under Rule 4.17**:
— *as to the identity of the inventor (Rule 4.17(i))*
— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

**Published**:
— *with international search report (Art. 21(3))*

(54) **Title**: ESTIMATING A PERFORMANCE CHARACTERISTIC OF A JOB USING A PERFORMANCE MODEL

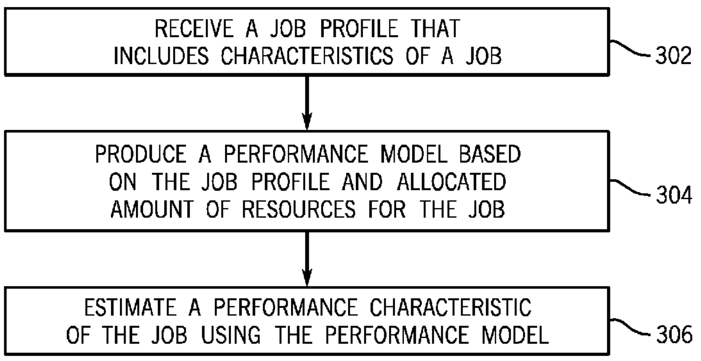
RECEIVE A JOB PROFILE THAT INCLUDES CHARACTERISTICS OF A JOB
302
PRODUCE A PERFORMANCE MODEL BASED ON THE JOB PROFILE AND ALLOCATED AMOUNT OF RESOURCES FOR THE JOB
304
ESTIMATE A PERFORMANCE CHARACTERISTIC OF THE JOB USING THE PERFORMANCE MODEL
306

FIG. 3

(57) **Abstract**: A job profile is received (302) that describes a job to be executed. A performance model is produced (304) based on the job profile and allocated amount of resources for the job, and a performance characteristic of the job is estimated (306) using the performance model.

WO 2012/105969 A1

# Estimating A Performance Characteristic Of A Job Using A Performance Model

## Background

[0001] Many enterprises (such as companies, educational organizations, and government agencies) employ relatively large volumes of data that are often subject to analysis. A substantial amount of the data of an enterprise can be unstructured data, which is data that is not in the format used in typical commercial databases. Existing infrastructure may not be able to efficiently handle the processing of relatively large volumes of unstructured data.

## Brief Description Of The Drawings

[0002] Some embodiments are described with respect to the following figures:

Fig. 1 is a block diagram of an example arrangement that incorporates some implementations;

Figs. 2A-2B are graphs illustrating map tasks and reduce tasks of a job in a MapReduce environment, according to some examples; and

Fig. 3 is a flow diagram of a process of estimating a performance characteristic of a job, according to some implementations.

## Detailed Description

[0003] For processing relatively large volumes of unstructured data, a MapReduce framework provides a distributed computing platform can be employed. Unstructured data refers to data not formatted according to a format of a relational database management system. An open-source implementation of the MapReduce framework is Hadoop. The MapReduce framework is increasingly being used across an enterprise for distributed, advanced data analytics and to provide new applications associated with data retention, regulatory compliance, e-discovery, litigation, or other issues. Diverse applications can be run over the same data sets to efficiently utilize the resources of large distributed systems.

[0004] Generally, the MapReduce framework includes a master node and multiple slave nodes. A MapReduce job submitted to the master node is divided into multiple map tasks and multiple reduce tasks, which are executed in parallel by the slave nodes. The map tasks are defined by a map function, while the reduce tasks are defined by a reduce function. Each of the map and reduce functions are user-defined functions that are programmable to perform target functionalities.

[0005] The map function processes corresponding segments of input data to produce intermediate results, where each of the multiple map tasks (that are based on the map function) process corresponding segments of the input data. For example, the map tasks process input key-value pairs to generate a set of intermediate key-value pairs. The reduce tasks (based on the reduce function) produce an output from the intermediate results. For example, the reduce tasks merge the intermediate values associated with the same intermediate key.

[0006] More specifically, the map function takes input key-value pairs ($k_1$, $v_1$) and produces a list of intermediate key-value pairs ($k_2$, $v_2$). The intermediate values associated with the same key $k_2$ are grouped together and then passed to the reduce function. The reduce function takes an intermediate key $k_2$ with a list of values and processes them to form a new list of values ($v_3$), as expressed below.

$$\begin{aligned} map(k_1,v_1) &\rightarrow list(k_2,v_2) \\ reduce(k_2,list(v_2)) &\rightarrow list(v_3) \end{aligned}.$$

[0007] Although reference is made to the MapReduce framework in some examples, it is noted that techniques or mechanisms according to some implementations can be applied in other distributed processing frameworks. More generally, map tasks are used to process input data to output intermediate results, based on a predefined function that defines the processing to be performed by the map tasks. Reduce tasks take as input partitions of the intermediate results to produce outputs, based on a predefined function that defines the processing to be performed by the reduce tasks. The map tasks are considered to be part of a map stage, whereas the reduce tasks are considered to be part of a reduce stage. In addition, although reference is made to unstructured data in some examples, techniques or mechanisms according to some implementations can also be applied to structured data formatted for relational database management systems.

[0008] Fig. 1 illustrates an example arrangement that provides a distributed processing framework that includes mechanisms according to some implementations for estimating performance characteristics of jobs to be executed in the distributed processing framework. As depicted in Fig. 1, a storage subsystem 100 includes multiple storage modules 102, where the multiple storage modules 102 can provide a distributed file system 104. The distributed file system 104 stores multiple segments 106 of input data across the multiple storage modules 102. The distributed file system 104 can also store outputs of map and reduce tasks.

[0009] The storage modules 102 can be implemented with storage devices such as disk-based storage devices or integrated circuit storage devices. In some examples, the storage modules 102 correspond to respective different physical storage devices. In other examples, plural ones of the storage modules 102 can be implemented on one physical storage device, where the plural storage modules correspond to different partitions of the storage device.

[0010] The system of Fig. 1 further includes a master node 110 that is connected to slave nodes 112 over a network 114. The network 114 can be a private network (*e.g.,* a local area network or wide area network) or a public network (*e.g.,* the Internet), or some combination thereof. The master node 110 includes one or more central processing units (CPUs) 124. Each slave node 112 also includes one or more CPUs (not shown). Although the master node 110 is depicted as being separate from the slave nodes 112, it is noted that in alternative examples, the master node 112 can be one of the slave nodes 112.

[0011] A "node" refers generally to processing infrastructure to perform computing operations. A node can refer to a computer, or a system having multiple computers. Alternatively, a node can refer to a CPU within a computer. As yet another example, a node can refer to a processing core within a CPU that has multiple processing cores. More generally, the system can be considered to have multiple processors, where each processor can be a computer, a system having multiple computers, a CPU, a core of a CPU, or some other physical processing partition.

[0012] In accordance with some implementations, the master node 110 is configured to perform scheduling of jobs on the slave nodes 112. The slave nodes 112 are considered the working nodes within the cluster that makes up the distributed processing environment.

[0013] Each slave node 112 has a fixed number of map slots and reduce slots, where map tasks are run in respective map slots, and reduce tasks are run in respective reduce slots. The number of map slots and reduce slots within each slave node 112 can be preconfigured, such as by an administrator or by some other mechanism. The available map slots and reduce slots can be allocated to the jobs. The map slots and reduce slots are considered the resources used for performing map and reduce tasks. A “slot” can refer to a time slot or alternatively, to some other share of a processing resource that can be used for performing the respective map or reduce task. Depending upon the load of the overall system, the number of map slots and number of reduce slots that can be allocated to any given job can vary.

[0014] The slave nodes 112 can periodically (or repeatedly) send messages to the master node 110 to report the number of free slots and the progress of the tasks that are currently running in the corresponding slave nodes. Based on the availability of free slots (map slots and reduce slots) and the rules of a scheduling policy, the master node 110 assigns map and reduce tasks to respective slots in the slave nodes 112.

[0015] Each map task processes a logical segment of the input data that generally resides on a distributed file system, such as the distributed file system 104 shown in Fig. 1. The map task applies the map function on each data segment and buffers the resulting intermediate data. This intermediate data is partitioned for input to the multiple reduce tasks.

[0016] The reduce stage (that includes the reduce tasks) has three phases: shuffle phase, sort phase, and reduce phase. In the shuffle phase, the reduce tasks fetch the intermediate data from the map tasks. In the sort phase, the intermediate data from the map tasks are sorted. An external merge sort is used in case the intermediate data does not fit in memory. Finally, in the reduce phase, the sorted intermediate data (in the form of a key and all its corresponding values, for example) is passed on the reduce function. The output from the reduce function is usually written back to the distributed file system 104.

[0017] The master node 110 of Fig. 1 includes a job profiler 120 that is able to create a job profile for a given job, in accordance with some implementations. The job profile describes characteristics of the given job to be performed by the system of Fig. 1. A job profile created by the job profiler 120 can be stored in a job profile database 122. The job profile database 122 can store multiple job profiles, including job profiles of jobs that have executed in the past.

[0018] In other implementations, the job profiler 120 and/or profile database 122 can be located at another node.

[0019] The master node 110 also includes a performance characteristic estimator 116 according to some implementations. The estimator 116 is able to produce an estimated performance characteristic, such as an estimated completion time, of a job, based on the corresponding job profile and resources (*e.g.,* numbers of map slots and reduce slots) allocated to the job. The estimated completion time refers to either a total time duration for the job, or an estimated time at which the job will complete. In other examples, other performance characteristics of a job can be estimated, such as cost of the job, error rate of the job, and so forth.

[0020] Figs. 2A and 2B illustrate differences in completion times of performing map and reduce tasks of a given job due to different allocations of map slots and reduce slots. Fig. 2A illustrates an example in which there are 64 map slots and 64 reduce slots allocated to the given job. The example also assumes that the total input data to be processed for the given job can be separated into 64 partitions. Since each partition is processed by a corresponding different map task, the given job includes 64 map tasks. Similarly, 64 partitions of intermediate results output by the map tasks can be processed by corresponding 64 reduce tasks. Since there are

64 map slots allocated to the map tasks, the execution of the given job can be completed in a single map wave.

[0021] As depicted in Fig. 2A, the 64 map tasks are performed in corresponding 64 map slots 202, in a single wave (represented generally as 204). Similarly, the 64 reduce tasks are performed in corresponding 64 reduce slots 206, also in a single reduce wave 208, which includes shuffle, sort, and reduce phases represented by different line patterns in Fig. 2A.

[0022] A "map wave" refers to an iteration of the map stage. If the number of allocated map slots is greater than or equal to the number of map tasks, then the map stage can be completed in a single iteration (single wave). However, if the number of map slots allocated to the map stage is less than the number of map tasks, then the map stage would have to be completed in multiple iterations (multiple waves). Similarly, the number of iterations (waves) of the reduce stage is based on the number of allocated reduce slots as compared to the number of reduce tasks.

[0023] Fig. 2B illustrates a different allocation of map slots and reduce slots. Assuming the same given job (input data that is divided into 64 partitions), if the number of resources allocated is reduced to 16 map slots and 22 reduce slots, for example, then the completion time for the given job will change (increase). Fig. 2B illustrates execution of map tasks in the 16 map slots 210. In Fig. 2B, instead of performing the map tasks in a single wave as in Fig. 2A, the example of Fig. 2B illustrates four waves 212A, 212B, 212C, and 212D of map tasks. The reduce tasks are performed in the 22 reduce slots 214, in three waves 216A, 216B, and 216C. The completion time of the given job in the Fig. 2B example is greater than the completion time in the Fig. 2A example, since a smaller amount of resources was allocated to the given job in the Fig. 2B example than in the Fig. 2A example.

[0024] Thus, it can be observed from the examples of Figs. 2A and 2B that it can be difficult to predict the execution time of any given job when different amounts of resources are allocated to the job.

[0025] In accordance with some implementations, mechanisms are provided to estimate a job completion time of a job as a function of allocated resources. By being able to estimate a job completion time as a function of allocated resources, the master node 110 (Fig. 1) is able to determine whether the given job is able to achieve a performance goal associated with the given job. In some examples, the performance goal is expressed as a specific deadline, or some other indication of a time duration within which the job should be executed. Other performance goals can be used in other examples. For example, a performance goal can be expressed as a service level objective (SLO), which specifies a level of service to be provided (expected performance, expected time, expected cost, etc.).

[0026] Fig. 3 is a flow diagram of a process according to some implementations. The process includes receiving (at 302) a job profile that includes characteristics of a particular job. Receiving the job profile can refer to a given node (such as the master node 110) receiving the job profile that was created at another node. Alternatively, receiving the job profile can involve the given node creating the job profile, such as by the job profiler 120 in Fig. 1.

[0027] Next, a performance model is produced (at 304) based on the job profile and allocated amount of resources for the job (*e.g.*, allocated number of map slots and allocated number of reduce slots). Using the performance model, a performance characteristic of the job is estimated (at 306). For example, this estimation can be performed by the performance characteristic estimator 116 in Fig. 1. In some implementations, the estimated performance characteristic is an estimated completion time of the job (an amount of time for the job to complete execution) given the allocated resources (*e.g.,* number of map slots and number of reduce slots). Alternatively, in other implementations, other performance characteristics of the job on a given set of resources can be estimated.

[0028] In some implementations, the particular job is executed in a given environment (including a system having a specific arrangement of physical machines and respective map and reduce slots in the physical machines), and the job profile and performance model are applied with respect to the particular job in this given environment.

[0029] A job profile reflects performance invariants that are independent of the amount of resources assigned to the job over time, for each of the phases of the job: map, shuffle, sort, and reduce phases.

[0030] The map stage includes a number of map tasks. To characterize the distribution of the map task durations and other invariant properties, the following metrics can be specified in some examples:

$$\left(M_{\min}, M_{avg}, M_{\max}, AvgSize_M^{input}, Selectivity_M\right), \text{where}$$

- $M_{\min}$ is the minimum map task duration. Since the shuffle phase starts when the first map task completes, $M_{\min}$ is used as an estimate for the shuffle phase beginning.
- $M_{avg}$ is the average duration of map tasks to indicate the average duration of a map wave.
- $M_{\max}$ is the maximum duration of a map task. Since the sort phase of the reduce stage can start only when the entire map stage is complete, *i.e.,* all the map tasks complete, $M_{\max}$ is used as an estimate for a worst map wave completion time.
- $AvgSize_M^{input}$ is the average amount of input data for a map stage. This parameter is used to estimate the number of map tasks to be spawned for a new data set processing.
- $Selectivity_M$ is the ratio of the map data output size to the map data input size. It is used to estimate the amount of intermediate data produced by the map stage as the input to the reduce stage (note that the size of the input data to the map stage is known).

[0031] The duration of the map tasks is affected by whether the input data is local to the machine running the task (local node), or on another machine on the same rack (local rack), or on a different machine of a different rack (remote rack). These different types of map tasks are tracked separately. The foregoing metrics can be used to improve the prediction accuracy of the performance model and decision making when the types of available map slots are known.

[0032] As described earlier, the reduce stage includes the shuffle, sort and reduce phases. The shuffle phase begins only after the first map task has completed. The shuffle phase (of any reduce wave) completes when the entire map stage is complete and all the intermediate data generated by the map tasks have been shuffled to the reduce tasks.

[0033] The completion of the shuffle phase is a prerequisite for the beginning of the sort phase. Similarly, the reduce phase begins only after the sort phase is complete. Thus the profiles of the shuffle, sort, and reduce phases are represented by their average and maximum time durations. In addition, for the reduce phase, the reduce selectivity, denoted as $Selectivity_R$, is computed, which is defined as the ratio of the reduce data output size to its data input size.

[0034] The shuffle phase of the first reduce wave may be different from the shuffle phase that belongs to the subsequent reduce waves (after the first reduce wave). This can happen because the shuffle phase of the first reduce wave overlaps with the map stage and depends on the number of map waves andtheir durations. Therefore, two sets of measurements are collected: $(Sh_{avg}^1, Sh_{\max}^1)$for a shuffle phase of the first reduce wave (referred to as the “first shuffle phase”), and $(Sh_{avg}^{typ}, Sh_{\max}^{typ})$for the shuffle phase of the subsequent reduce waves (referred to as “typical shuffle phase”). Since techniques according to some implementations are looking for the performance invariants that are independent of the amount of allocated resources to the job, a shuffle phase of the first reduce wave is characterized in a special way and the parameters $(Sh_{avg}^1 \text{ and } Sh_{\max}^1)$ reflect only durations of the non-overlapping portions (non-overlapping with the map stage) of the first shuffle. In other words, the durations represented by $Sh^1_{avg}$ and $Sh^1_{\max}$ represent portions of the duration of the shuffle phase of the first reduce wave that do not overlap with the map stage.

[0035] Thus, the job profile in the shuffle phase is characterized by two pairs of measurements:

$$\left(Sh^1_{avg}, Sh^1_{\max}\right),\ \left(Sh^{typ}_{avg}, Sh^{typ}_{\max}\right).$$

[0036] If the job execution has only a single reduce wave, the typical shuffle phase duration is estimated using the sort benchmark (since the shuffle phase duration is defined entirely by the size of the intermediate results output by the map stage).

[0037] Once the job profile is provided, then a performance model that is based on the job profile can be produced (304 in Fig. 3). In some implementations, the performance model is based on the job profile and lower and upper bounds of time durations of different phases of the job. The performance model is also produced based on an allocated amount of resources for the job (*e.g.*, allocated number of map slots and allocated number of reduce slots). Such a performance model can be used for predicting the job completion time as a function of the job input data set and the allocated resources, where the job input data set refers to the input data to the job that is to be performed.

[0038] In some implementations, the performance model is characterized by lower and upper bounds for a makespan (a completion time of the job) of a given set of $n$ ($n > 1$) tasks that are processed by $k$ ($k > 1$) servers (or by $k$ slots in a MapReduce environment). Let $T_1, T_2, ..., T_n$ be the durations of $n$ tasks of a given job. Let $k$ be the number of slots that can each execute one task at a time. The assignment of tasks to slots is done using a simple, online, greedy algorithm, *e.g.,* assign each task to the slot with the earliest finishing time.

[0039] Let $\mu = \left(\sum_{i=1}^{n} T_i\right)/n$ and $\lambda = \max_i \{T_i\}$ be the mean and maximum durations of the $n$ tasks, respectively. The makespan of the greedy task assignment is at least $n \cdot \mu/k$ and at most $(n-1) \cdot \mu/k + \lambda$. The lower bound is trivial, as the best case is when all $n$ tasks are equally distributed among the $k$ slots (or the overall amount of work $n \cdot \mu$ is processed as fast as it can by $k$ slots). Thus, the overall makespan (completion time of the job) is at least $n \cdot \mu/k$ (lower bound of the completion time).

[0040] For the upper bound of the completion time for the job, the worst case scenario is considered, *i.e.,* the longest task $(T) \in (T_1, T_2, \ldots, T_n)$ with duration $\lambda$ is the last task processed. In this case, the time elapsed before the last task is scheduled is $\left(\sum_{i=1}^{n-1} T_i\right)/k \leq (\mathrm{n}-1) \cdot \mu/k$. Thus, the makespan of the overall assignment is at most $(n-1) \cdot \mu/k + \lambda$. These bounds are particularly useful when $\lambda << n \cdot \mu/k$, in other words, when the duration of the longest task is small as compared to the total makespan.

[0041] The difference between lower and upper bounds (of the completion time) represents the range of possible job completion times due to non-determinism and scheduling. As discussed below, these lower and upper bounds, which are part of the properties of the performance model, are used to estimate a completion time for a corresponding job *J*.

[0042] The given job *J* has a given profile created by the job profiler 120 (Fig. 1) or extracted from the profile database 122. Let *J* be executed with a new input dataset that can be partitioned into $N_M$ map tasks and $N_R$ reduce tasks. Let $S_M$ *and* $S_R$ be the number of map slots and the number of reduce slots, respectively, allocated to job *J*.

[0043] Let $M_{avg}$ and $M_{\max}$ be the average and maximum time durations of map tasks (defined by the job *J* profile). Then, based on the Makespan theorem, the lower and upper bounds on the duration of the entire map stage (denoted as $T_M^{up}$ and $T_M^{up}$, respectively) are estimated as follows:

$$T_M^{low} = N_M/S_M \cdot M_{avg},$$

$$T_M^{up} = (N_M - 1)/S_M \cdot M_{avg} + M_{\max}.$$

[0044] Stated differently, the lower bound of the duration of the entire map stage is based on a product of the average duration ($M_{avg}$) of map tasks multiplied by the ratio of the number map tasks ($N_M$) to the number of allocated map slots ($S_M$). The upper bound of the duration of the entire map stage is based on a sum of the maximum duration of map tasks ($M_{\max}$) and the product of $M_{avg}$ with $(N_M - 1)/S_M$. Thus, it can be seen that the lower and upper bounds of durations of the map stage are based on properties of the job *J* profile relating to the map stage, and based on the allocated number of map slots.

[0045] The reduce stage includes shuffle, sort and reduce phases. Similar to the computation of the lower and upper bounds of the map stage, the lower and upper bounds of time durations for each of the shuffle phase ($T_{Sh}^{low}, T_{Sh}^{up}$), sort phase ($T_{Sort}^{low}, T_{Sort}^{up}$), and reduce phase ($T_R^{low}, T_R^{up}$) are computed. The computation of the Makespan theorem is based on the average and maximum durations of the tasks in these phases (respective values of the average and maximum time durations of the shuffle phase, the average and maximum time durations of the sort phase, and the average and maximum time duration of the reduce phase) and the numbers of reduce tasks $N_R$ and allocated reduce slots $S_R$, respectively. The formulae for calculating ($T_{Sh}^{low}, T_{Sh}^{up}$), ($T_{Sort}^{low}, T_{Sort}^{up}$), and ($T_R^{low}, T_R^{up}$) are similar to the formulate for calculating $T_M^{up}$ and $T_M^{up}$ set forth above, except variables associated with the reduce tasks and reduce slots and the respective phases of the reduce stage are used instead.

[0046] The subtlety lies in estimating the duration of the shuffle phase. As noted above, the first shuffle phase is distinguished from the task durations in the typical shuffle phase (which is a shuffle phase subsequent to the first shuffle phase). As noted above, the first shuffle phase includes measurements of a portion of the first shuffle phase that does not overlap the map stage. The portion of the typical shuffle phase in the subsequent reduce waves (after the first reduce wave) is computed as follows:

$$T_{Sh}^{low} = \left( \frac{N_R}{S_R} - 1 \right) \cdot Sh_{avg}^{typ} ,$$

$$T_{Sh}^{up} = \left( \frac{N_R - 1}{S_R} - 1 \right) \cdot Sh_{avg}^{typ} + Sh_{\max}^{typ} .$$

where $Sh_{avg}^{typ}$ is the average duration of a typical shuffle phase, and $Sh_{\max}^{typ}$ is the average duration of the typical shuffle phase. The formulae for the lower and upper bounds of the overall completion time of job *J* are as follows:

$$T_J^{low} = T_M^{low} + Sh_{avg}^1 + T_{Sh}^{low} + T_{Sort}^{low} + T_R^{low} ,$$

$$T_J^{up} = T_M^{up} + Sh_{\max}^1 + T_{Sh}^{up} + T_{Sort}^{up} + T_R^{up} ,$$

where $Sh_{avg}^1$ is the average duration of the first shuffle phase, and $Sh_{\max}^1$ is the maximum duration of the first shuffle phase. $T_J^{low}$ and $T_J^{up}$ represent optimistic and pessimistic predictions (lower and upper bounds) of the job *J* completion time. Thus, it can be seen that the lower and upper bounds of durations of the job *J* are based on properties of the job *J* profile and based on the allocated numbers of map and reduce slots. The properties of the performance model, which include $T_J^{low}$ and $T_J^{up}$ in some implementations, are thus based on both the job profile as well as allocated numbers of map and reduce slots.

[0047] In some implementations, estimates based on the average value between the lower and upper bounds tend to be closer to the measured duration. Therefore, $T_J^{avg}$ is defined as follows:

$$T_J^{avg} = \left(T_M^{up} + \right) T_J^{low} / 2.$$

[0048] In some implementations, the value $T_J^{avg}$ is considered the estimated completion time for job *J* (estimated at 306 in Fig. 3). In other implementations, other estimated time duration based on $T_J^{low}$ and $T_J^{up}$ can be derived, such as a weighted average or the application of some other predefined function based on the lower and upper bounds ($T_J^{low}$ and $T_J^{up}$).

[0049] The estimation of a performance characteristic of a job, such as its completion time, can be computed relatively quickly, since the calculations as discussed above are relatively simple. As a result, the master node 110 (Fig. 1) or other decision maker in a distributed processing framework (such as a MapReduce framework) can quickly obtain such performance characteristic information of a job to make decisions, such as scheduling decisions, resource allocation decisions, and so forth.

[0050] Machine-readable instructions of modules described above (including 116, 120, 122 in Fig. 1) are loaded for execution on one or more CPUs (such as 124 in Fig. 1). A CPU can include a microprocessor, microcontroller, processor module or subsystem, programmable integrated circuit, programmable gate array, or another control or computing device.

[0051] Data and instructions are stored in respective storage devices, which are implemented as one or more computer-readable or machine-readable storage media. The storage media include different forms of memory including semiconductor memory devices such as dynamic or static random access memories (DRAMs or SRAMs), erasable and programmable read-only memories (EPROMs), electrically erasable and programmable read-only memories (EEPROMs) and flash memories; magnetic disks such as fixed, floppy and removable disks; other magnetic media including tape; optical media such as compact disks (CDs) or digital video disks (DVDs); or other types of storage devices. Note that the instructions discussed above can be provided on one computer-readable or machine-readable storage medium, or alternatively, can be provided on multiple computer-readable or machine-readable storage media distributed in a large system having possibly plural nodes. Such computer-readable or machine-readable storage medium or media is (are) considered to be part of an article (or article of manufacture). An article or article of manufacture can refer to any manufactured single component or multiple components. The storage medium or media can be located either in the machine running the machine-readable instructions, or located at a remote site from which machine-readable instructions can be downloaded over a network for execution.

[0052] In the foregoing description, numerous details are set forth to provide an understanding of the subject disclosed herein. However, implementations may be practiced without some or all of these details. Other implementations may include modifications and variations from the details discussed above. It is intended that the appended claims cover such modifications and variations.

What is claimed is:

1. A method comprising:

receiving (302), in a system having a plurality of processors, a job profile that includes characteristics of a job to be executed, wherein the characteristics of the job profile relate to map tasks and reduce tasks of the job, wherein the map tasks produce intermediate results based on segments of input data, and the reduce tasks produce an output based on the intermediate results;

producing (304), by the system, a performance model based on the job profile and an allocated amount of resources for the job; and

estimating (306), by the system, a performance characteristic of the job using the performance model.

2. The method of claim 1, further comprising:

determining, by the system based on the estimated performance characteristic, whether a performance goal of the job will be satisfied.

3. The method of claim 2, further comprising receiving an indication of the allocated amount of resources for the job, wherein the allocated amount of resources comprises an allocated number of map slots and number of reduce slots, wherein the map tasks are performed in the map slots, and the reduce tasks are performed in the reduce slots.

4. The method of claim 1, wherein estimating the performance characteristic comprises estimating a completion time of the job.

5. The method of claim 1, wherein producing the performance model comprises producing the performance model having a lower bound and an upper bound of the performance characteristic.

6. The method of claim 5, wherein the performance characteristic is a completion time of a job, the method further comprising:

computing the lower bound based on a number of the map tasks, a number of reduce tasks, a number of allocated map slots, a number of allocated reduce slots, an average time duration of a map task, an average time duration of a shuffle phase in a reduce stage, an average time duration of a sort phase in the reduce stage, and an average time duration of a reduce phase in the reduce stage, wherein the reduce stage includes the reduce tasks; and

computing the upper bound based on the number of the map tasks, the number of reduce tasks, the number of allocated map slots, the number of allocated reduce slots, the average time duration of a map task, a maximum time duration of a map task, the average time duration of the shuffle phase, a maximum time duration of the shuffle phase, the average time duration of the sort phase, a maximum time duration of the sort phase, the average time duration of the reduce phase, and a maximum time duration of the reduce phase.

7. The method of claim 1, wherein receiving the job profile including the characteristics of the job includes receiving the job profile including plural ones of: a minimum time duration of a map task, an average time duration of a map task, a maximum time duration of a map task, an average size of input data for a map task, an average time duration of a reduce task, and a maximum time duration of a reduce task.

8. The method of claim 7, wherein the job profile further includes: a parameter indicating a ratio between an output data size of a map stage that includes the map tasks and an input data size to the map stage, and a parameter indicating a ratio between an output data size and an input data size associated with a reduce stage that includes the reduce tasks.

9. An article comprising at least one machine-readable storage medium storing instructions that upon execution cause a system having a processor to perform a method according to any of claims 1-8.

10. A system comprising:

storage media (122) to store a job profile, wherein the job profile describes a job including a map stage to produce an intermediate result based on input data, and a reduce stage to produce an output based on the intermediate result; and

at least one processor (124) to:

produce parameters of a performance model based on the job profile and an allocated amount of resources for the job; and

generate an estimated performance characteristic of the job using the performance model.

11. The system of claim 10, wherein the parameters include an upper bound of the performance characteristic and a lower bound of the performance characteristic.

12. The system of claim 10, wherein the performance characteristic is an estimated completion time of the job.

13. The system of claim 12, wherein the at least one processor is to further:

compute the lower bound based on a number of map tasks in the map stage, a number of reduce tasks in the reduce stage, a number of allocated map slots, a number of allocated reduce slots, an average time duration of a map task, an average time duration of a shuffle phase in the reduce stage, an average time duration of a sort phase in the reduce stage, and an average time duration of a reduce phase in the reduce stage; and

compute the upper bound based on the number of the map tasks, the number of the reduce tasks, the number of allocated map slots, the number of allocated reduce slots, the average time duration of a map task, a maximum time duration of a map task, the average time duration of the shuffle phase, a maximum time duration of the shuffle phase, the average time duration of the sort phase, a maximum time duration of the sort phase, the average time duration of the reduce phase, and a maximum time duration of the reduce phase.

14. The system of claim 10, wherein the allocated amount of resources includes a number of map slots and a number of reduce slots on physical machines, wherein map tasks of the map stage are performed in the map slots, and reduce tasks of the reduce stage are performed in the reduce slots.

15. The system of claim 10, wherein the job profile includes parameters selected from among a minimum time duration of a map task in the map stage, an average time duration of a map task in the map stage, a maximum time duration of a map task in the map stage, an average size of input data for a map task in the map stage, an average duration of a phase of the reduce stage, and a maximum time duration of a phase in the reduce stage.

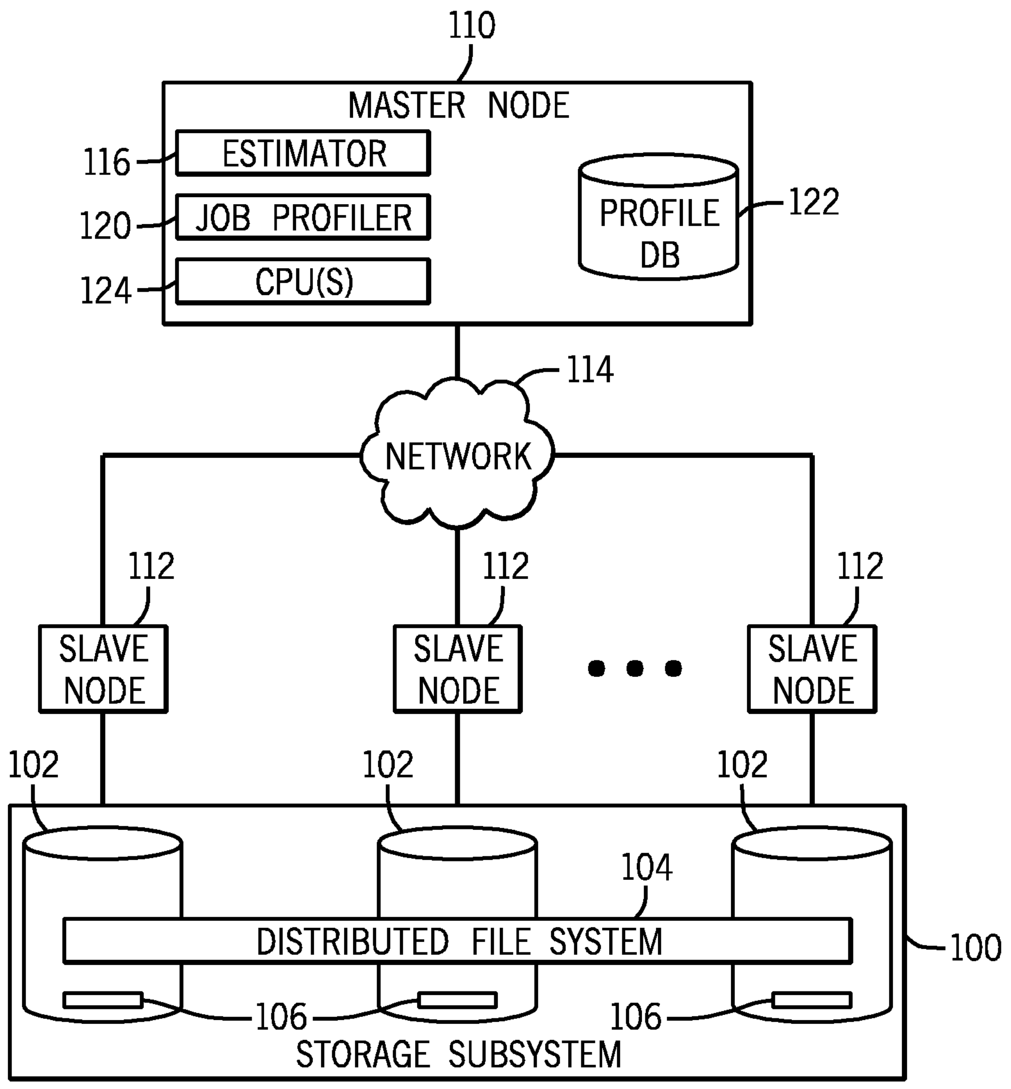
110
MASTER NODE
116
ESTIMATOR
120
JOB PROFILER
124
CPU(S)
PROFILE DB
122
114
NETWORK
112
SLAVE NODE
112
SLAVE NODE
112
SLAVE NODE
102
102
102
104
DISTRIBUTED FILE SYSTEM
100
106
106
STORAGE SUBSYSTEM

FIG. 1

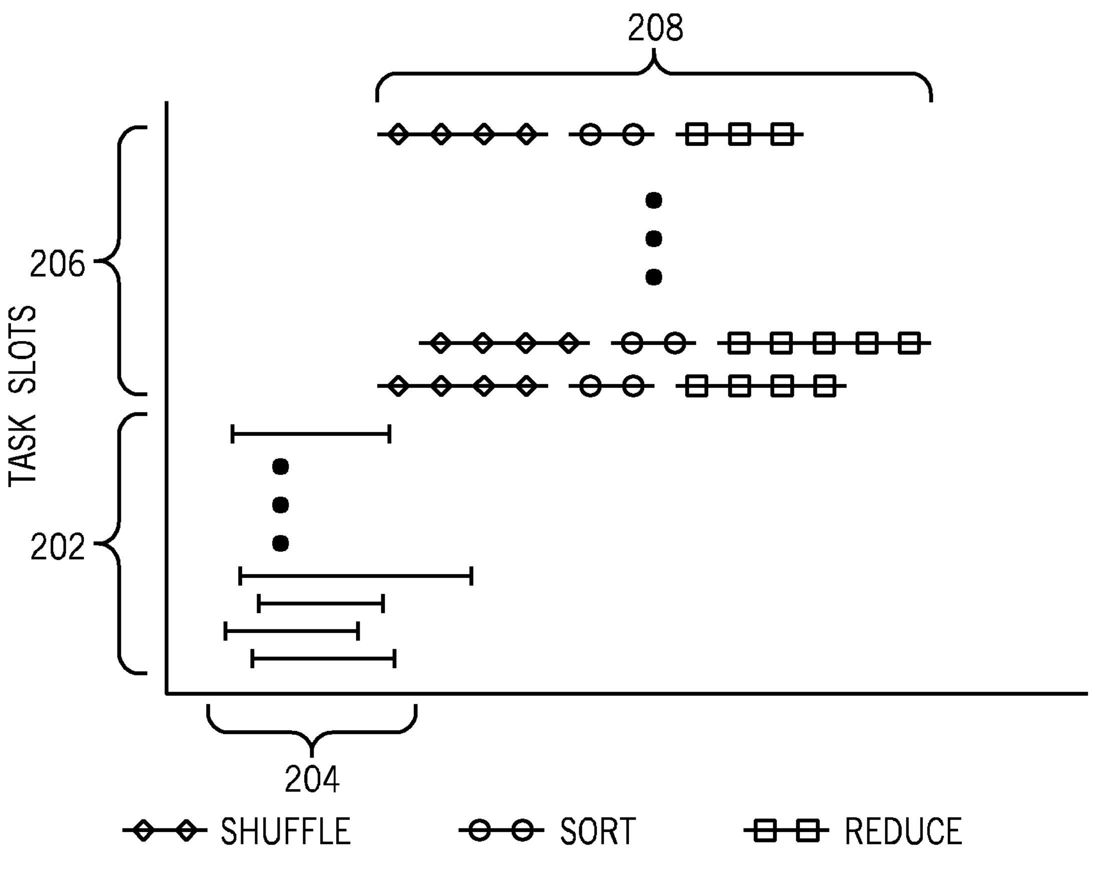
208
206
TASK SLOTS
202
204
SHUFFLE
SORT
REDUCE

FIG. 2A

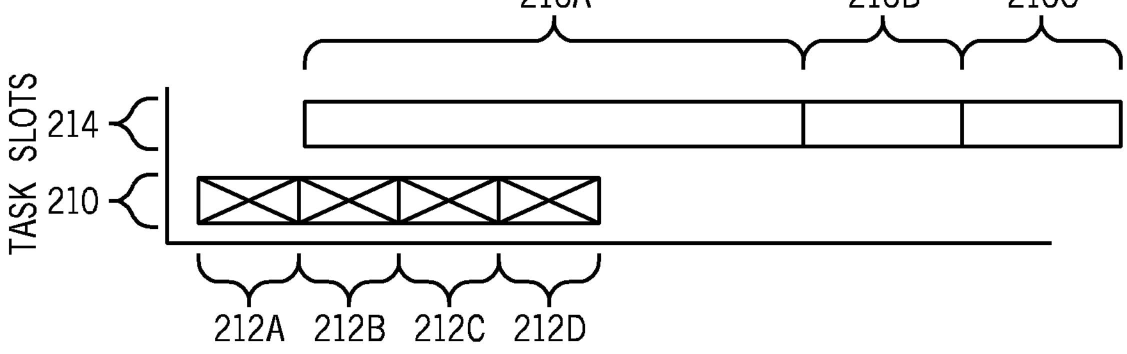
216A
216B
216C
TASK SLOTS
214
210
212A
212B
212C
212D

FIG. 2B

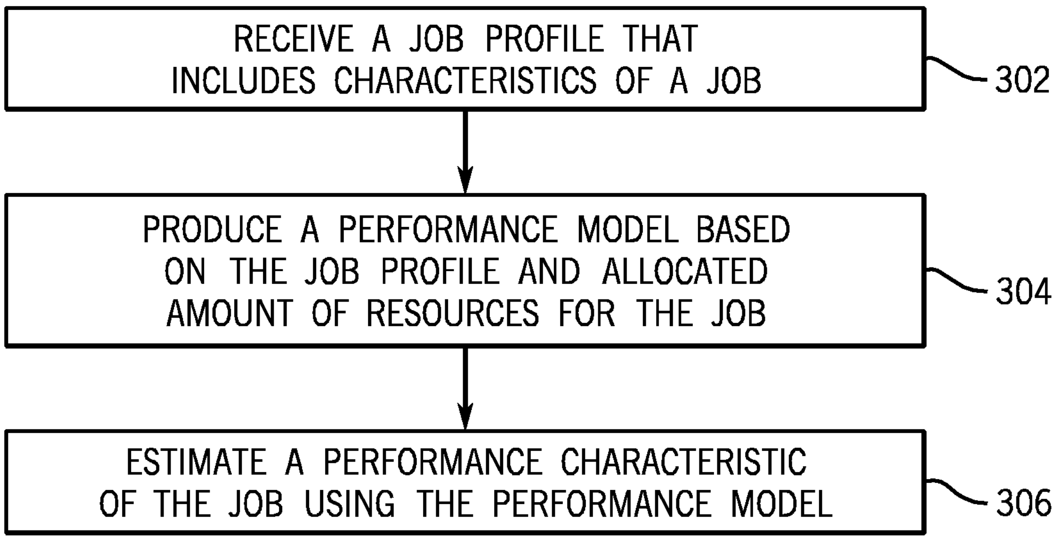
RECEIVE A JOB PROFILE THAT INCLUDES CHARACTERISTICS OF A JOB
302
PRODUCE A PERFORMANCE MODEL BASED ON THE JOB PROFILE AND ALLOCATED AMOUNT OF RESOURCES FOR THE JOB
304
ESTIMATE A PERFORMANCE CHARACTERISTIC OF THE JOB USING THE PERFORMANCE MODEL
306

FIG. 3

**INTERNATIONAL SEARCH REPORT**

International application No.
**PCT/US2011/023438**

**A. CLASSIFICATION OF SUBJECT MATTER**

***G06F 9/44(2006.01)i, G06F 9/30(2006.01)i, G06F 15/16(2006.01)i***

According to International Patent Classification (IPC) or to both national classification and IPC

**B. FIELDS SEARCHED**

Minimum documentation searched (classification system followed by classification symbols)
G06F 9/44

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
Korean utility models and applications for utility models
Japanese utility models and applications for utility models

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
eKOMPASS(KIPO internal) & Keywords: map-reduce, performance model, optimization

**C. DOCUMENTS CONSIDERED TO BE RELEVANT**

| Category* | Citation of document, with indication, where appropriate, of the relevant passages | Relevant to claim No. |
|---|---|---|
| X | ABOULNAGA, A., et al Packing the most onto your cloud, IN: the first international workshop on Cloud data management, ACM New York, NY, USA, 2009 (ABOULNAGA, A., et al). | 1,2,4,9,10,12 |
| A | See abstract and chapter 1, 4. | 3,5-8,11,13-15 |
| A | RANGER, C., et al Evaluating MapReduce for multi-core and multiprocessor systems, Evaluating mapreduce for multi-core and multiprocessor systems. IN: Proceedings of 13th International Symposium on High-Performance Computer Arctetecture (HPCA), 2007(RANGER, C., et al) See abstract and chapter 5.3. | 1-15 |
| T | HERODOTOU, H., et al Profiling, What-if Analysis, and Cost-based Optimization of MapReduce Programs, IN: the VLDB endowment, vol.4, no.11. 11 August - 3 September 2011 (HERODOTOU, H., et al) See abstract and chapter 1.1, 2.1, 3. | 1-15 |

☐ Further documents are listed in the continuation of Box C.  ☒ See patent family annex.

| | |
|---|---|
| * Special categories of cited documents: | "T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention |
| "A" document defining the general state of the art which is not considered to be of particular relevance | "X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone |
| "E" earlier application or patent but published on or after the international filing date | "Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents,such combination being obvious to a person skilled in the art |
| "L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of citation or other special reason (as specified) | "&" document member of the same patent family |
| "O" document referring to an oral disclosure, use, exhibition or other means | |
| "P" document published prior to the international filing date but later than the priority date claimed | |

| | |
|---|---|
| Date of the actual completion of the international search<br>30 SEPTEMBER 2011 (30.09.2011) | Date of mailing of the international search report<br>**30 SEPTEMBER 2011 (30.09.2011)** |
| Name and mailing address of the ISA/KR<br>Korean Intellectual Property Office<br>Government Complex-Daejeon, 189 Cheongsa-ro, Seo-gu, Daejeon 302-701, Republic of Korea<br>Facsimile No. 82-42-472-7140 | Authorized officer<br>Hwang, Seung Hee<br>Telephone No. 82-42-481-5749 |

**INTERNATIONAL SEARCH REPORT**
Information on patent family members

International application No.
**PCT/US2011/023438**

| Patent document cited in search report | Publication date | Patent family member(s) | Publication date |
|---|---|---|---|
| None | | | |