Federal Synergy Computing Model Based on Network Interconnection

Sorush Niknamian

Military Medicine Department, Liberty University,USA

A B S T R A C T

To solve the shortage problem of the computing power provided by the single machine or the small cluster system in scientific research, we offer a collaborative computing system for users. This system has massive operation ability. It introduced a scalable mixed collaborative computing model. Through the internet and the heterogeneous computing equipment, the system uses the task decomposition model. This system can solve the research and development problem because of the shortage of capacity. To test the model, a subtask decomposition example is used. The results of the example analysis show that the computing work can obtain the shortest computation time when the number of calculation nodes is more than the number of subtasks; Maximum calculation efficiency can be achieved when the number of the calculating nodes closes to the number of subtasks. Through joint collaborative computing, the extensible mixed collaborative computing mode can effectively solve the mass computing problem for the system with heterogeneous hardware and software. This paper provides the reference for the system, which provides large scale computing power through the Internet and the research problem of due to the lack of computing ability.

Keywords: Federal Computing; Task Decomposition; Task Scheduling; Data Communication

1. Introduction

Under the impetus of science and engineering applications, many breakthroughs have been made in the computational model with the support of the algorithm and the architecture by integrating in modern computing network technology [1-4].High performance computing capacity through the model provides a powerful foundation platform for science computing of related areas. The simulations of various natural phenomena have reached unprecedented accuracy by these high performance computing platforms [5], and the platform provides a supercomputing ability to design new drugs against newer viruses and other diseases [6]. Large systems (such as cosmology) and small systems (such as cell research) are hungry for computing power, this is our relentless power to research computing model, improve system computing power and build large-scale computing platform.Computing hardware stack and parallel system development can provide general-purpose computing large scale parallel computing capabilities; however, expensive computing hardware input and complex system design are difficult in small system scientific computing. This is our Research objectives that designing a cheap platform for many small system scientific computing and providing a super computing power to solve the problem of insufficient computing power, and the platform is easy to expand and develop parallel computing system.

This research is devoted to the development of a scalable hybrid federal computing model for building cheap, scalable high energy computing systems, to provide a computing method for the field such as the classified system or the special scene that cannot be applied by other ways such as cloud computing. In the whole research, the construction of system model is the core content and key technology of system design.

2. Design the System Architecture

2.1 Topology of the System Network

Although the network computing provides unprecedented computing power for scientific computing, It is difficult to complete large-scale computing because the lack of scientific research computing equipment and special requirements in certain fields that cannot provide by the network computing such

as cloud computing. When necessary, we need to build a heterogeneous platform through simple redeploying and connecting existed computing resources with a certain computing scale, or spend large amount of funds to build a private computing platform with large computing power, we expect the platform to have remote control characteristics. Motivated by this demand, we design a heterogeneous scalable hybrid federal computing model based on network, the overall network topology of the system is shown as Figure 1.

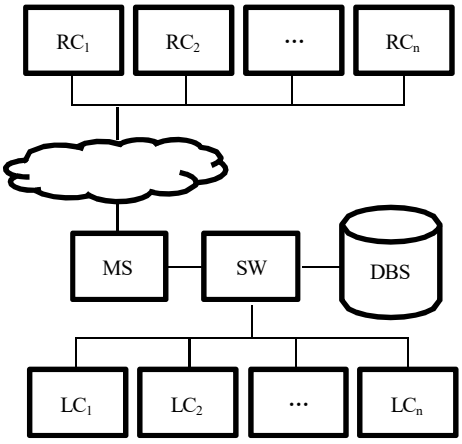

Figure 1: The System network topology

Fig1.network topology diagram of the system LC1~n are the local computing nodes, the number of which depends on the size of t he network IP address allocation pool. Therefore, different types of network provide different number of node accesses, and as a result constrain the computing capacity of the computing system constructed by them

DBS provides a data storage system for the system; the whole system can share this data source, with the help of this node the system can complete the data sharing and publishing. MS system is designated as the uniquely management server, all the computing nodes and the access node must be registered again for this server. The server is responsible for the establishment of the task, the task assignment, task scheduling, as the center core server that are respectively connected intranet and extranet, building a connection channel for RC and LC. Through RC1~n, a remote random access client, the MS can be established and asked for computing tasks, and one can also download the COM component from the MS server to join computing, and to accept the MS scheduling. MS, DBS and LC1~n are connected together through a switch SW, which is responsible for assigning network addresses to them in turn. In this way, LC and RC do not need the same physical structure or software system, which can shield the difference of the system structure and provide the capability of heterogeneous collaborative computing through the upper layer of software design.

2.2 Task Processing Flow

The remote device RC request to the MS for initiates task， and the task will be submitted in the form of task description. The design of task description Reference [9], and introduce the concept of multitasking job processing [10], we design the MS task flow as shown in the Figure 2. Remote users access the MS via the Internet and submit an application to MS. When the RC application login successfully, MS will be set the RC identity to the server. The RC will become a remote computing node of MS when it accesses to MS after the application is approved by the server, then MS will issue a general computing task for it, If the RC submits to the MS application for computing tasks, MS will review the calculation of the RC application, if the application is accepted, the establishment of RC computing tasks will be successful, if not accepted, the establishment of the task will be failed. If the RC does not have a task description, then load the job description after writing the job description, then MS task will be scheduled according to the task description. If the access node of the MS is 0 or the idle node is 0, the system cannot complete the task, and the task is waiting to be executed, otherwise the task will be assigned to perform. After the end of the computing，the RCs Will report to the system that the Task completed, and release system resources. According to the needs of the application, the system constructs by hybrid system architecture. The Map Reduce computing model is introduced in the process of assigning tasks to each computing node [11]. The system task specification describes a subset of subtasks that split a large problem into a number of small problems, and then perform the tasks on each node in the cluster computing node, it is a Map process. At the end of the Map process, each node in the cluster will compile, execute and solve the tasks according to the task specification. After the completion of the task, there will be a reduce process, this process will bring all the computing output results of the decomposition of the subtasks together, and send it to MS and DBS. Whether it is a Reduce process that brings the results together after the system is completed, or the Map process that is executed when the system is initialized, Subtask execution nodes need to the distribute server for the necessary tasks description, the task specification describes data sources, remote storage of intermediate key/value results and submit the results of the implementation, the task distribute server shall provide an entrance to this service or service. Therefore, it is necessary to provide a large data query and analysis model for the data nodes, and provide remote data access API to capture the data of the system design; In order to avoid the accumulation and loss of data, it is necessary to introduce a new method to store the new data of the system to server when the computing nodes need to save the new generated data in a certain time window, This will ensure different computing nodes computing performance that bounds to different servers on the data movement and the operation not movement; In order to solve the problem of large data

query and analysis, we need to calculate the cluster configuration of a small memory computing cluster, The introduction of memory computing model to improve the computing performance of a variety of computing models to deal with large data, A variety of computing models are mixed with the memory computing model, which can achieve high real-time data query and analysis.

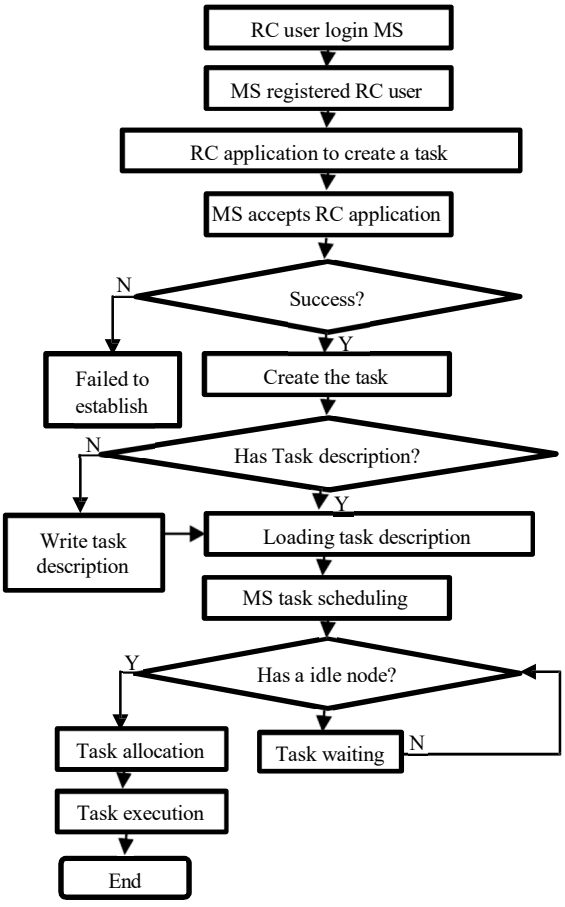

Figure 2: Task processing flow chart

Remote users access the MS via the Internet and submit an application to MS. When the RC application login successfully, MS will be set the RC identity to the server. The RC will become a remote computing node of MS when it accesses to MS after the application is approved by the server, then MS will issue a general computing task for it, If the RC submits to the MS application for computing tasks, MS will review the calculation of the RC application, if the application is accepted, the establishment of RC computing tasks will be successful, if not accepted, the establishment of the task will be failed. If the RC does not have a task description, then load the job description after writing the job description, then MS task will be scheduled according to the task description. If the access node of the MS is 0 or the idle node is 0, the system cannot complete the task, and the task i s waiting to be executed, otherwise the task will be assigned to perform. After the end of the computing，the RCs Will report to the system that the Task completed, and release system resources.

According to the needs of the application, the system constructs by hybrid system architecture. The MapReduce computing model is introduced in the process of assigning tasks to each computing node [11]. The system task specification describes a subset of subtasks that split a large problem into a number of small problems, and then perform the tasks on each node in the cluster computing node, it is a Map process. At the end of the Map process, each node in the cluster will compile, execute and solve the tasks according to the task specification. After the completion of the task, there will be a reduce process, this process will bring all the computing output results of the decomposition of the subtasks together, and send it to MS and DBS. Whether it is a Reduce process that brings the results together after the system is completed, or the Map process that is executed when the system is initialized, Subtask execution nodes need to the distribute server for the necessary tasks description, the task specification describes data sources, remote storage of intermediate key/value results and submit the results of the implementation, the task distribute server shall provide an entrance to this service or service. Therefore, it is necessary to provide a large data query and analysis model for the data nodes, and provide remote data access API to capture the data of the system design; In order to avoid the accumulation and loss of data, it is necessary to introduce a new method to store the new data of the system to server when the computing nodes need to save the new generated data in a certain time window, This will ensure different computing nodes computing performance that bounds to different servers on the data movement and the operation not movement; In order to solve the problem of large data query and analysis, we need to calculate the cluster configuration of a small memory computing cluster, The introduction of memory computing model to improve

the computing performance of a variety of computing models to deal with large data, A variety of computing models are mixed with the memory computing model, which can achieve high real-time data query and analysis.

3. Subtask Decomposition Model and Task Description

Reference the literature [12], the system model of subtask decomposition method is designed.Given the computing task T, when the complexity of the task O (T) is greater than the given threshold value, Continue to resolve the decomposition Subtask T_i of task T, T_i can be described by the task tree view description language (TTVDL) based on XML. Create a list of tasks on the basis of task representation, computing task requests from computing node N, and establishing the Thread of computing node. Open the leaf node (i) after the root traversal calculation based on the tree depth first algorithm.
Task decomposition scheduling algorithm divides the simulation task into 2 layer m fork tree, assigned to each computing unit. If the Subtask is larger, it can continue to decompose. The task can be decomposed statically or dynamically. It is necessary to determine the granularity of decomposition, the coefficient of convergence and the convergent boundary of decomposition.

3.1 Task Decomposition Algorithm

A computing task can be described by a task system (T, M, S, L, P). The task decomposition model is shown in Figure 3. The system uses two layers of nested DAG, the sub_DAG is a collection of subtasks DAG_i decomposed by DAG, E is a collection of communication edge e_i, T is a collection of communication costs T_i.

.

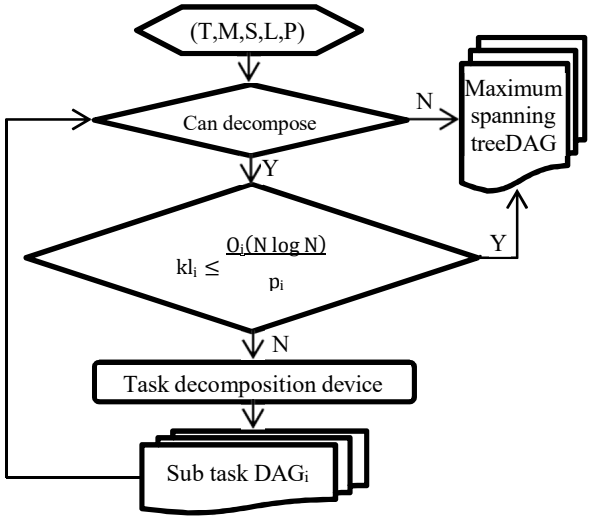

Figure 3: Task decomposition model

Thus we obtained:

$$\begin{cases} DAG = \{sub_{DAG}, E, T, V\} \\ sub\_DAG = \{DAG_1, DAG_2 DAG_3 ..., DAG_n\} \\ E = \{e_1, e_2, e_3 ... e_n\} \\ T = \{t_1, t_2, t_3 ... t_n\} \\ DAG_i = \{subV_i subE_i subT_i subC_i\} \end{cases} \quad (1)$$

DAG_i is the No. i task in the collection of t subtask, E is a collection of communication edges between DAGi, Cij ∈C, Cij is a collection of communication costs between DAG_i. Because the subtask uses the decomposition method of 2 layer m fork tree, the communication cost will not change because of the task decomposition; the cost is related to the relationship between subtasks and task size; subtask DAG_i is a decision directed acyclic graph, and DAG_i is the 2 layer of M fork tree. If DAG_i cannot be decomposed, then m=0, otherwise the value of M is related to the decomposition strategy. $subV_i$, $subE_i$, subTi , $subC_i$ are the collection of DAG_i neutron tasks, the collection of communication edges between subtasks, the collection of sub task completion times, and the collection of communication between the leaf node and the root node. If the number of tasks is m+1, the number of communication side is M.
In the whole DAG collection, there is a data dependency between DAG_i. Data dependency between DAGi constitutes a dependency collection, Dependency collection is defined as a implementation results collection of Subtask DAG_i requires subtasks DAG_m,..., DAG_n or the transfer variables during the execution of this task. Then the {DAG_m,... , DAG_n} is called data dependency collection of DAG_i. The data dependency collection can be

extracted from the task control process according to the subtask function, the subtask DAG_i must be executed after its data dependency collection is executed. The relation between subtasks Data dependencies are called data dependencies. In addition, there are control dependencies, mutual exclusion relationship, concurrency relationship and interest relationship among subtasks in the whole task decomposition process. When the output of task DAG_m is the input of subtasks DAG_n, there is control dependent relationship between DAG_m and DAG_n; When the task DAG_m is runing, the DAG_n subtasks cannot be performed, and when the task DAG_n is runing, the DAG_m subtasks cannot be performed, It shows that there is a mutually exclusive relationship between subtask DAG_m and subtask DAG_n; When the DAG_m and DAG_n subtask can be executed at the same time, and the execution of a subtask does not affect the execution of another subtask, then the subtask DAG_m and DAG_n constitute a concurrent relationship; When the subtask of the implementation of DAG_m can improve the efficiency and quality of the subtask DAG_n, then the subtask DAG_m and subtask DAG_n constitute an interest relationship.

In order to satisfy the data dependency between DAG_i, the first root traversal must be performed. To solve this problem, we must solve the collection of previous node and the collection of next node. Let $\Pr eviousNode(i)$ is the collection of previous node，NextNode(i) is the collection of next node，Thus we obtained:

$$\begin{cases} \Pr eviousNode(i) = \{ j \mid e_{ji} \in E \} \\ NextNode(i) = \{ j \mid e_{ij} \in E \} \end{cases} \quad (2)$$

As a 2 layer m fork tree, task DAGi has the explicitly previous and subsequent relationship between each task, therefore, it do not need to seek the relationship between the subtasks.

3.2 Define Subtask Convergence Boundary

In order to reduce the transmission of the original data, reduce the traffic and improve the network throughput, a copy of the original data is saved in the access unit m_i, the MI can be either a computer or a computing independent network unit composed of several computers. The first layer of the task can be extracted from the original data copy of the local computing unit; it does not require data transmissions. The original data and the final results are stored in the S_i of data center DBS.

Set M as a collection of computing unit m_i in the system, S is a collection of data center s_i, L is a collection of computing unit capacity L_i, P is a collection of computing power p_i.

Thus we obtained:

$$\begin{cases} M = \{ m_1, m_2, m_3, ..., m_m \} \\ S = \{ s_1, s_2, s_3, ..., s_s \} \\ L = \{ l_1, l_2, l_3, ..., l_m \} \\ P = \{ p_1, p_2, p_3, ..., p_m \} \end{cases} \quad (3)$$

m_i is the No. i unit in the collection of computing units, s_i is the No. i unit in the collection of storage units, l_i corresponds to the load capacity of the computing unit m_i, p_i corresponds to the computing power of the computing unit m_i

When the system task is decomposed into an m fork tree with hierarchical structure, the tree has a total of N subtasks, the complexity O is introduced, which reduces the complexity from $O(N^2)$ to $O(N \log N)$ [13]. Thus we obtained:

$$kl_i \le \frac{O_i (N \log N)}{p_i}$$

In the formula, K is the coefficient of convergence. Given the k value, when the ratio of the decomposition subtask complexit y and the matched with computing power less than or equal to the given boundary convergence condition kl_i, then stop decomposition, and the decomposition tree is sent to the computing unit m_{i_n}

3.3 Task Decomposition description

Direct at the computational tasks proposed by RC, The system uses task descriptions to describe the task decomposition, task allocation, task recovery and so on, each task corresponds to a task description. The nodes involved in the computation need to get the task description fr om the server and compile it locally. When the computing node LC is ready, the ready signal is sent to the management server, waiting for system scheduling. Manage server MS to maintain a task description for each computing task, the task computing dictionary is generated in the MS, the MS implementation processor scheduling by polling the task description calculates dictionary and queries the status of each computing node. The Reference [9] used XML as a task description method; we also use the XML task tree view to describe the task when designing the task description of the system. The task specification base node is as follows:

<?xml version="1.0" encoding="utf-8" ?>
<TaskDescription>
<TaskDividedTree></TaskDividedTree>
<SubTaskMapping>

<ComputingNode treeID=“”>
<ImportData></ImportData>
<ExportData></ExportData>
<NodeDependence></NodeDependence>
<ComputingCode></ComputingCode>
</ComputingNode>
</SubTaskMapping>
</TaskDescription>

The <TaskDividedTree /> Node is the static description of the whole task decomposition tree. Each Node contain ed in the node has a strict description of the communication edge ei, the communication cost ti and others of the subtask DAGi. The node“s $Node(i)$ hierarchical relationship reflects the relationship between the previous and next nodes, this node is the basis of task scheduling. Node <Su bTaskMapping /> is the input, output, static description and calculation method of dependence of each sub node, How many sub nodes are described in <TaskDividedTree />, the <SubTaskMapping /> will contain a description of the number of tasks that do not exceed <TaskDividedTree />, TreeID is the computing node <ComputingNode /> association Key between <TaskDividedTree /> and <SubTaskMapping /> .Node <ImportData /> contains the input requirements of the computational tasks, and Node <ExportData /> contains the final results, The results of the calculation of the node will upload to storage server after the computing completion. the node will be recovered before rescheduling when the task computing completion, and the results of the last task before the recovery will still be maintained in the node, can provide P2P node data access, this can reduce server data transfer pressure. <NodeDependence /> is a collection of dependencies of nodes. By accessing the nodes, the nodes can be set to wait, sleep, and wake up and so on. <ComputingCode /> is the algorithm description of the computing nodes. According to this algorithm, the computing nodes are dynamically compiled and calculated locally. The algorithm is compiled only when the first load, the second running is no longer compiled, which is different from the interpretation of the implementation, so the performance loss can be ignored.

4. Task Scheduling Algorithm and Model Evaluation

4.1 Task Scheduling Algorithm

The system algorithm has improved which based on the original task scheduling algorithm in Reference [10]. The improved model uses a hybrid strategy, and its algorithm is described as follows.

In order to improve the efficiency and throughput of the cluster, the task allocation is reasonable when scheduling a group of tasks, so that the computing resources of each computing node can be fully utilized. In order to prevent some computing tasks from being permanently executed, we must consider the equalization of the computing resources as much as possible in each task group when the system was first designed.

A task will go through seven states from the task submission to the execution end, such as wait, Map, ready, execute, reduce and complete. When a computational task is successfully created, it needs to be submitted to the system, in the first the system checks completeness of task description, and following the task instructions are itemized audit verification. Each LC is queried according to the task description of the sub task description tree when the instructions through the inspection.it is necessary to wait for the non-idle computing node to complete other tasks when the idle LC is not able to satisfy the computing task, then the submitted task at this time enters the wait state; According to the task description tree, each sub task will be mapped to each local computing node LC when the system has the idle LC to meet the computing task, then the submitted task at this time enters the ready state; Next the management server assign each node in turn to start the calculation according to the instructions of the dependencies in the mission, then the system into the implementation state; When each task node performs all the tasks in turn, the Complete signal is reported to the management server, and the results are transmitted to the storage center, then the system enters the reduce state; The system enters the finished state when all the tasks have been completed and return all the results. The management server sends the GC command to each node that joins the computation, carries on the garbage collection, releases the resource, and wait until the next scheduling.

Task scheduling algorithm adopts priority algorithm and first come first serve (FCFS) hybrid scheduling algorithm, and add rotation method basic idea. Maintain a Dictionary<int, Queue <Task>> dictionary in the MS server. Where Key is the priority of the task queue, Queue <Task> is task queue, Task is a single computing task. The algorithm principle is shown in Figure 4.

When the computational task is established, the system is statically assigned a priority value K, the K-value is between 1-n. The task enters the corresponding priority queue according to the K value. The task is queued according to the first come first serve (FCFS) scheduling algorithm when it enters the queue because they have the same priority. Viewed from a straight line, the algorithm is fair in general sense, that is, each task depends on how long they wait in the queue to determine whether or not they prioritize services. But for those tasks that have a shorter execution time, they will wait a long time if they arrive after a long time of execution. To this end, this system uses the round robin, and set a time slice for each task. When the task is out of time slice, the execution of the task is aborted, and the K-1 value of the task is determined; If the value of the K-1 is in the Dictionary Keys, that is, the value of Dictionary.ContainsKey (K-1) is equal to true, then the task is removed from head and added to the end of the Dictionary, it is contained in Dictionary[K-1] team; Otherwise it is added to the end of the Dictionary[K] team. The choice of the time slice length will directly affect the system overhead and response time. The number that the programs deprive the system of computation will increase if the length of the time slice is too short, and this will increase the cost of the system. If the time slice length is too long, in extreme cases, a time slice can guarantee the required execution time of the longest task that can be executed in the queue, the system will lose the round robin, and just use FCFS algorithm. The selection of the time slice length can be determined according to the requirement of the response time of the system R and the maximum allowable tasks number N_{max} in the queue, and it can

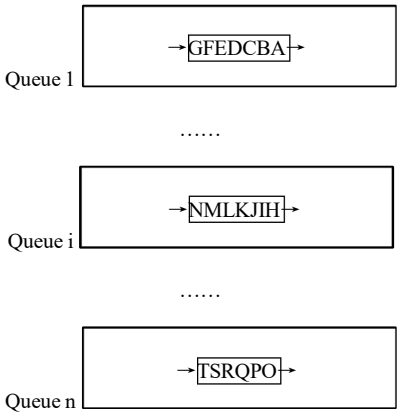

Figure 4: Algorithm principle

be expressed as: q=R/Nmax. In the const value of Q, the response time of R seems to be greatly reduced if the number of tasks in the queue is far less than Nmax. But for system overhead, the timing of task switching will not change due to the fixed value of Q. For simplicity, the system uses a fixed time slice.

The performance of task scheduling can be measured by the parameters, such as task turnaround time, response time, throughput, and the utilization ratio of computing nodes. Here we focus on the task turnaround time. The turnaround time for the task i is defined as Ti, thus: Ti=Tie-Tis. Where the Tis is the start time of the task and the Tie is the end time of the task completion. For n (n>=1) tasks, the average turnaround time is:

$$T = \frac{1}{n}\sum_{i=1}^{n} T_i$$

When the task is submitted to the system, it will be executed immediately until the task is Mapped, Therefore, the task is likely to enter the wait state. Set T_{iw} as the waiting time that the task from the submission to Map, then correct turnaround time as Ti, and $T_i=T_{ir}+T_{iw}$, there T_{ir} is the execution time. Furthermore, we can use the weight of the turnaround time to measure the scheduling performance. Define the weighted turnaround time as the ratio of task turnaround time to task execution time: $W_i=T_i/T_{ir}$. For the n tasks contained in the task flow, the average weighted turnaround time is:

$$W = \frac{1}{n}\sum_{i=1}^{n} W_i$$

4.2 Model Evaluation

Through the revision and improvement of the scheduling algorithm in literature [12], the evaluation model of the system is as follows Assuming that the size of the particle had linearly related to the size of the task, the execution time Ti is:

$$T_i = b_i + a_i x_i \tag{4}$$

b_i is the time of initializing the system, a_i is the task granularity linear growth factor, x_i is the size of tasks.

Assuming that the data transmission time had linearly related to the size of the task, then,

$$Data\_T_{ij} = Data\_b_{ij} + Data\_a_{ij} x_i \tag{5}$$

In the formula, $Data\_T_{ij}$ is the required time that transferred data from the task i to the task j. Where $Data\_b_{ij}$ is the time required to transmit the initialization data, $Data\_a_{ij}$ is a linear factor.

Formula (6) (7) can be adopted to solve the TCP traffic model, the model can be referenced literature [14] [15]. In the high speed local area network with 100 M/1000 M adaptation, the ratio of the data transfer time and the computation time are small in the whole simulation process, that is because the transmission rate between computers is very high, while $Data\_b_{ij}$ and $Data\_a_{ij}$ are relatively small and a copy of the original data has been saved in the computing unit prior to the start of the calculation.

$$T\left(t_{RTT}, s, p\right) = \frac{c \times s}{t_{RTT} \times \sqrt{p}} \tag{6}$$

$$T\left(t_{RTT}, s, p\right) = \min\left(\frac{w_m \times s}{t_{RTT}}, \frac{s}{t_{RTT} \times \sqrt{\frac{2bp}{3}} + t_{RTo} \min\left(1, 3, \frac{3bp}{8}\right) p\left(1 + 32p^2\right)}\right) \tag{7}$$

For the 2 layer m fork tree DAG_i, the size of the sub task is total M copies, but the granularity of the M subtasks are different. Then, the relationship of the sub task scale is $x_i = x_{1i} + x_{2i} + \cdots + x_{mi}$, we can assume that the time of task execution and the task scale are linear relationship, and the execution time of each subtask is $sub\_T_{ki} = b_i + a_i x_{ki}$. The data transfer time between the leaf node and the root node is $subData\_T_{ki} = Data\_b_{ij} + Data\_a_{ij} x_{ij}$, here k=1,2,3,... M. For task i, if it is not decomposed, the task completion time is calculated by the formula (1); if it is decomposed, then the formula (5) is used

$$T_i = Max\left(sub_{T_{1i}},...sub_{T_{mi}}\right) + sub\_root_i \quad (8)$$

According to the characteristics of sub task diversity, the primary role of the root task is: transmits data from the root node to the leaf node, the compute and collect results from the leaf nodes to the root node, and transform the root task computing result to the DAG map of lower sub task. Therefore, the computing time of sub_root_i is mainly the data transmission time. If the scheduling algorithm supports a data parallel transmission, sub_root_i can be approximated by the formula (9):

$$\begin{aligned} sub\_root_i &= Max\left(subData_{T_{1k}},...,subData_{Tmk}\right) \\ &+Max\left(Data_{T_{i1}},...,Data\_T_{is}\right) \end{aligned} \quad (9)$$

5. Compute Node Assignment

MS loads the subtasks into a task list List<T> by reading the task description. Then, the task priority of each sub task is determined according to the dependency set List List<R> of each task in their description. In the calculation of node allocation, at first each sub tasks in the List<T> will be distributed into a different Dictionary<int, Queue<T>> according to the level of each sub tasks. Where the int is the task queue level, the Queue<T> is the same level task queue. The tasks in the queue are scheduled according to the FIFO strategy, and the high level sub task queue is given priority to compute the node assignment. The FIFO strategy is used to compute node allocation between tasks and tasks. When the time slice of the task T in the queue is used up, it will release the computing node, and then return to the end of the queue, waiting for rescheduling. When the task interdependence leads to competition for resources, the task will be sent to the low level queue by reducing the level of sub tasks, and this can solve the problem of deadlock caused by task preemption. The computing node is released and the system task is completed when the task is completed. The node will request to reassign the task and modify the state of the task in the MS. The MS will notify the subtasks that are waiting for the dependency to continue execution by event method.

6. Computing data communication model

According to the calculation model of the above design, master-slave mode and P2P mode are adapted to the communication and data exchange between the nodes, the chart of Compute node LC_n startup flow as shown in Figure5:

The node will run the joint computing program, which maps on when it is started. After the program starts, it initializes the parameter information of the node in the first. The service address of the managed server MS is stored in each LC compute node. It is done when the LC is remotely deployed by the system configuration; using this parameter, LC can sense the presence of the server and try to connect to the management server; If the connection is not successful, then the hardware link fails, the node cannot access the collaborative computing system, It will become a calculation of ac-node; If the node can connect to the server, it will be registered on the MS itself, and the registration information contains the basic information of the node, computing power, etc.; The MS server will run programs in computing task nodes after successful registration. If the MS server does not have a task at this time, that is to say, the federal computing system is idle, then the node will set itself to idle, waiting for scheduling; MS will scan the status of the local compute node client LC after it completes the initialization of the task when the MS server has a RC application task; If the number of idle computing nodes LC which the MS scanned is more than 0, then the resource allocation and task scheduling, if the idle nodes which the MS scanned is 0, the task will be set the task into scheduling queue and wait for being scheduled because the lack of resources; The idle LC will download the task specification and load it when it receives the MS scheduler. LC compiles the subtask execution code in task specification through a dynamic compilation system, and applies for the issuance of subtasks from MS after the task specification was compiled. Under normal circumstances, the subtask execution code in task specification can be compiled through. This can only show that the calculation of the computing power of the node cannot meet the requirements of the task description if it cannot be compiled by the instructions. When the LC receives the sub task of MS, it carries out the task loading, and analyzes whether there are other sub task dependencies; If there is a dependency, the output parameters of the subtasks a ssociated with subtasks are first obtained; If LC can get data, it is illustrated that the sub task has been terminates and its output can be used as input parameters for the this task, otherwise, the output data cannot meet the input of the task and then the task is required to re calculate the output in accordance with the requirements of this task. When the output of all dependent subtasks can satisfy the input of the task, the task is executed; the results of the calculation will be uploaded to the data sharing area for other subtasks. The LC that completed the task computing can be reinitiated and request to the MS for another subtask. If there are no subtasks available, LC is set to be idle and waiting for the MS scheduler.

Three methods are used to realize the communication and data exchange between nodes. These data which has calculated by the computing node and merged to the server can be applied by the other nodes that apply to data management server. When the application for the identification of the identity of the consumer data is audited, the application node can consume data provided by the production data node; if the node is unable to meet the requ est of the data node to the management server, the reason for the failure of the data is checked; if the other computing node is calculating the application data, the calculation node enters the wait state, and registers the waiting resource application to the MS server. When all the calculations are completed and all

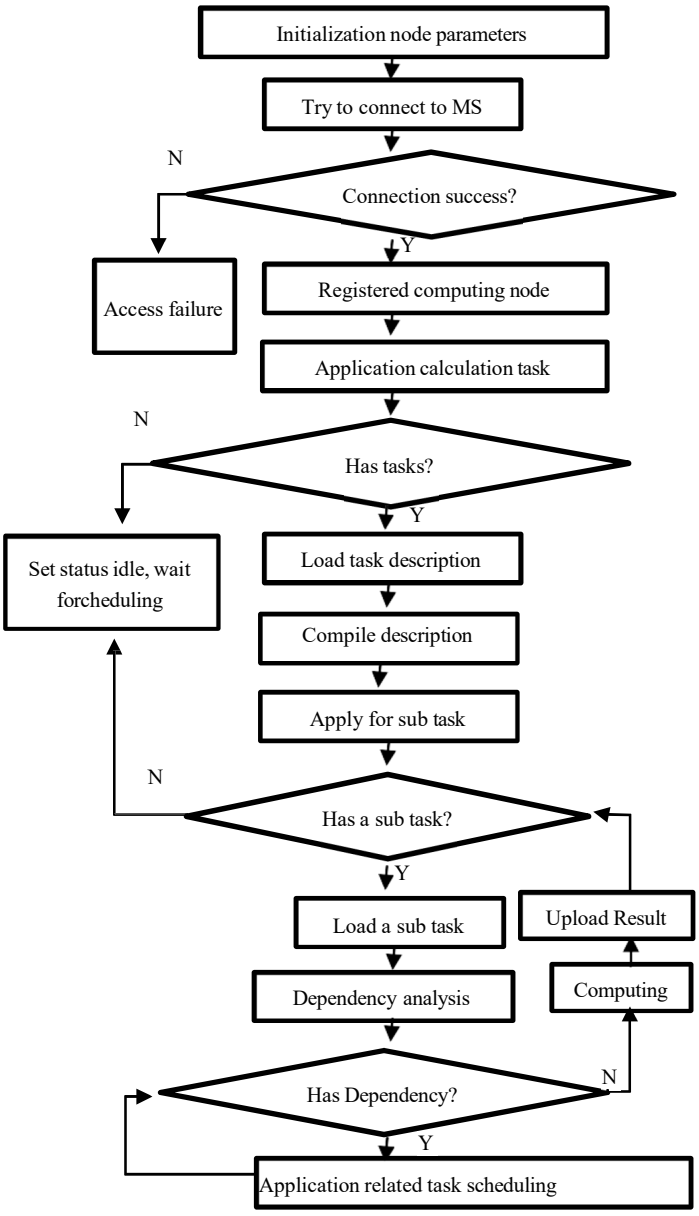

Figure 5: Compute node LC_n startup flow

the MS server will find the waiting nodes from resource application, and inform those application data nodes which is listed in the application form to loading data; If the data is not retrieved on the management server, and the current computing network does not have a computing node to compute the data, then set the current computing node into the stack, and compute dependent data set.

In order to ensure the communication and event notification between the computing node and the MS server, shield the difference of computing nodes hardware and heterogeneous structure of the operating system, the computing nodes use the Net.Tcp communication protocol to provide remote services.by the open Web service. In the system design, the Windows Communication Foundation (WCF) is adopted to provide data sources. The WCF provides a high performance network communication protocol which based on Net.Tcp protocol and system components with Net.Tcp Port Sharing Service, so that the port can be shared between multiple user processes. The data exchange uses the XML language which based on the object transfer protocol, and this provides the possibility for the exchange of structured and solidified information between heterogeneous computing nodes. In addition, in order to ensure the data access security between nodes, the system uses a security algorithm based on the elliptic curve algorithm and federal verification [16].

Due to the different environment of LC and RC, the complexity of the RC host itself and the limitation of the access rights, the RC and LC system are designed by different strategies. LC and RC also use different protocols in the communication method. LC uses the Net.Tcp protocol, but RC uses the pollingDuplexHttpBinding protocol. Because HTTP has a strong ability to penetrate the firewall, it can prevents the scheduling failure because of the RC node blocked by host firewall

In order to make the management server can simultaneously serve as two kinds of protocols, it needs to configure netTcpBinding and pollingDuplexHttpBinding in the bindings section of the serviceModel section of the protocol. The pollingDuplexHttpBinding configuration is as follows:

<pollingDuplexHttpBinding>

……

</pollingDuplexHttpBinding>

And add this section into the <bindings> section. When configuring the netTcpBinding protocol, you need to add the following section to the <bindings> section:

<netTcpBinding>

……

</netTcpBinding>

The system uses the Silverlight rich client as the development model of RC in the RC endpoint. Because Silverlight does not support the WCF Security model, if you want to call this service in SL, you must set the Security to None. By default, Security Mode is Transport, so this section must not be omitted and must be explicitly configured.

When configuring the information about the service, two endpoint points need be added because of the adoption of the two protocols. There are two kinds of endpoints in the <services> node, one is called by the client, and the other is the publication of metadata for the generation of service information. Using <endpoint contract= "IMetadataExchange" binding= "mexTcpBinding" address= "mex" /> node to publish metadata. Using <endpoint address= "ForWinform" contract="NetTcpDuplexCommunication.Server.IService1" binding= "netTcpBinding" bindingConfiguration= "tcpConfig" /> node to Configure client Net.TCP calls. Using <endpoint address= "ForSilverLight" binding= "pollingDuplexHttpBinding" binding Configuration= "pollingDuplexHttpBinding1" contract= "EndoscopeIMS.Server.IServiceForEndoscopeCDS"/>node to Configure client pollingDuplexHttpBinding calls.

It should be attention that when adding baseAddresss to the host section, because Silverlight can only use ports between 4502 and 4534 (with very complete hints in the Silverlight exception), we use a port of 4502.

It also should be attention that when adding references to write completely:

net.tcp://localhost:4502/NetTcpDuplexCommunication.Server/Service1.svc/ForWinform

In addition, you need to set the aspNetCompatibilityEnableds to true, such as: <serviceHostingEnvironment multipleSiteBindingsEnabled="true" aspNetCompatibilityEnabled="true"/>

In order to complete the communication between the server and the client, the interface must be specified in the WCF interface. When a callback [ServiceContract (CallbackContract= typeof IClientCallback))] attribute for the WCF interface is added, the [OperationContract(IsOneWay = true)] attribute must be added to all callback methods for the interface IClientCallback.

A method is included in the IService1 interface to register subscribe server. The server call back RC or LC client by the registration of computing nodes in Register, the client callback, and the client to unsubscribe by UnRegister. Custom callback method PumpMessageModel is the implementation of callback push message pump. When the server callback the client failed, the server will think that the client is offline and must be remove it from the list server computing nodes.

On the server side, the [AspNetCompatibilityRequirements(RequirementsMode=AspNetCompatibilityRequirementsMode.Required)] and [ServiceBehavior(ConcurrencyMode=ConcurrencyMode.Multiple,InstanceContextMode=InstanceContextMode.Single)] attributes must be added to the Service1 implementation class that inherits the IService1 interface. There is a static Dictionary<string, IClientCallback> _clients dictionary to save the registered remote computing node in the Class, and it is a static list of clients. The implementation of the Register is to add the current channel directly to the _clients dictionary for the invocation.

When the specified channel is not callback, the channel is removed from the _clients dictionary, and declares that the computing node is dead. The node will no longer be assigned sub tasks and scheduled. The server will reclaim the task that has been assigned to the node, and then re - perform the Map. on the other idle nodes.

When the system is deployed, since the WCF host uses a high version of IIS that supports net.tcp binding, it is necessary to enable the HTTP and the Net.Tcp two protocols, and modify the net.tcp configuration to bind 4502:*. Since WCF Activation is an optional component of Windows, it is not installed by default, so WCF Activation need to be installed for the IIS support WCF calls to non HTTP pipes.

In order to ensure that the RC of the Silverlight allows crossing domain access applications in the absence of policy files, cross domain configuration clientaccesspolicy.xml files are added to the WCf host publishing directory. In addition, the built-in program must have trust level elevated trusted in OOB mode.

7. Use Case Test of Computing Model

Given a computing cluster system C which consists of Management server M, Storage service cluster Si, and Computing node N_i, Then C can be described as: C={M， S_i， N_i}.

Given i=1 of use case C test calculation cluster S_i. N_i is a collection of {N_1， N_2， N_3， N_4， N_5， N_{10}， N_{20}， N_{40}， N_{80}， N_{120}}.

Given the job J, the J can be broken down into subtasks set T_i and subtask dependency set R_i. Then J can be described as: J={T_i， R_i}.

Given the subtask set T_i and the subtask dependency R_i of the test case J, it can be described as:

T_i={T_A,T_B,T_C,T_D,T_E,T_F,T_G,T_H,T_I,T_J,T_K,T_L,T_M,T_N,T_O,T_P,T_Q}；

R_i={$R_A \rightarrow R_{BCDEFHN}$,$R_B \rightarrow R_{HL}$, $R_C \rightarrow R_{HINO}$, $R_D \rightarrow R_I$, $R_E \rightarrow R_{GJ,RG} \rightarrow R_K$, $R_H \rightarrow R_{MN}$, $R_I \rightarrow R_P$, $R_J \rightarrow R_{PQ}$ }

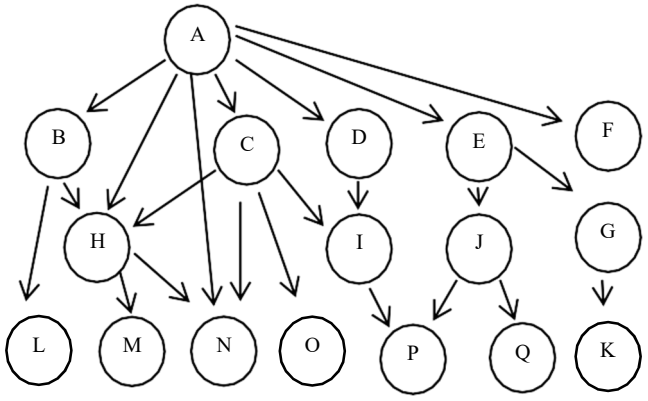

Figure 6: The dependency set R_i relation

The dependency set Ri relation is shown in Figure 6.By the dependency set R_i, the priority queue Q_i in M can be defined: $Q_i=\{Q_1, Q_2, Q_3, Q_4\}=\{\{A\}, \{BCDEF\}, \{HIJG\}, \{LMNOPQK\}\}$, here the queue priority is as follows: $Q_1>Q_2>Q_3>Q_4$. For the decomposed subtask T_i, its execution time can be described by a four tuple(T_{in}, T_{out}, $T_{instructions}$, $T_{commcapacity}$), Where T_{in} is the time required to execute the task execution, which is dependent on the functional dependencies of the dependency set R_i and the input data size; T_{out} is the result of the output of the task to the data center, which is mainly affected by the output data scale and network communication ability; $T_{instructions}$ is the time required to compute the node N_i execution of the subtask T_i, whose length is determined by the computing power of the node N_i (the total number of instructions executed per second) and the total number of subtasks. The $T_{commcapacity}$ is a main expression of measuring the communication capacity of the node, the communication throughput of the node N_i is greater, and the time of each communication is shorter. The task simulation test case data are shown in Table 1.

Table 1 Simulation test case data

Task Name	TotalTask Instruction	Output DataSize	Network Communication Capability
A	6128701	23552	921
B	5243421	22528	614
C	8336538	44032	204
D	4481950	46080	716
E	7788828	29696	716
F	9793721	47104	716
G	6797833	49152	716
H	6534583	16384	921
I	9688247	11264	102
J	2105543	46080	102
K	7359003	46080	716
L	1510364	38912	512
M	1215425	49152	819
N	9784983	26624	307
O	2083561	25600	512
P	1855908	32768	102
Q	5329746	27648	614

Test results are shown in figure 7 for use case:

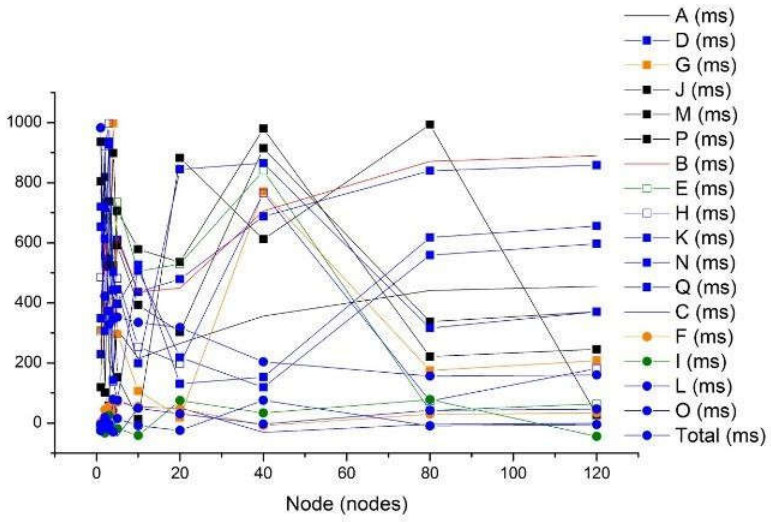

Figure 7: Use case test results

As can be seen from the graph, with the increase of computing nodes, the time required for the task is gradually reduced; However, when the number of nodes reaches a certain threshold, the time required by the number of nodes is gradually weakened; The main factor that affects the time required to complete the task is the computing capacity of a single node and the network communication speed, the computing capacity stronger, the communication speed faster, and the less time required.

8. Conclusion

This federal synergy computing model which the system provided with heterogeneous and dynamic characteristics can be applied to large-scale network and support the dynamic check in and check out. Using computer networks connect heterogeneous computer devices to provide high performance computing capabilities is currently common method of supper-large scale computing. With the help of previous research results, this paper proposes a compact scalable hybrid federal computing model based onliterature [17-22]. Tocompared with the current mainstream network computing model, the implementation of the proposed method shields computing nodes differences in the software and hardware by design of the application network protocollayer. Any computing device can access the system at any time to participate in the operation. It greatly reduces the cost of computing equipment and the formation of a network of inexpensive computing and provides an alternative solution for the rapid implementation of a large scale computing network. The system has high expansibility and feasibility to compare with the method provided by literature [18]. The task decomposition algorithm in this paper is a further extension of the method mentioned in the literature [12], and further improves the application environment of the method. However, Task decomposition algorithm in this system cannot be completely decomposed by MS. this paper will focus on enhance the automation and intelligence of the program and improve the task diversity algorithm. The calculation model proposed in this paper, to a certain extent, has the advanced nature and reference to solve this kind of method, and has certain practical significance for engineering guidance.

Acknowledgements

This research was supported by the Science and technology plan project of Henan province (162102210082); High level talent introduction research start project of North China University of Water Resources and Electric Power (40427)

REFERENCES

[1] Szymanski T. High performance computing with optical interconnects. Proceedings of SPIE - The International Society for Optical Engineering, 2000, 4109: 217-225.

[2] Aoki K, Yamagiwa S, Ferreira K. Maestro2: High speed network technology for high performance computing. Proc of IEEE International Conference on Communications, NJ: IEEE, 2004, 2: 1033-1037.

[3] Walker E. Creating private network overlays for high performance scientific computing. Lecture Notes in Computer Science, 2007, 4834: 204-222.

[4] Nishi H, Tasho K, Yamamoto J, et al. A local area system network RHiNET-1: A network for high performance parallel computing. Proceedings of the IEEE International Symposium on High Performance Distributed Computing, 2000, 1: 296-297.

[5] Pronk S, Pouya I, Lundborg M. Molecular Simulation Workflows as Parallel Algorithms: The Execution Engine of Copernicus, a Distributed High-Performance Computing Platform. Journal of Chemical Theory and Computation, 2015, 11(6): 2600-2608.

[6] Calabrese B,Cannataro M. Cloud computing in healthcare and biomedicine. Scalable Computing, 2015, 16(1): 1-18.

[7] Bezbradica M,Crane M,Ruskin H J. Applications of high performance algorithms to large scale cellular automata frameworks used in pharmaceutical modelling. Journal of Cellular Automata, 2016, 11(1): 21-45.

[8] Somasundaram T S, Govindarajan K. CLOUDRB: A framework for scheduling and managing High-Performance Computing (HPC) applications in science cloud. Future Generation Computer Systems, 2014, 34(5): 47-65.

[9] Liu B. Agent task decomposition and scheduling in distributed network management. Nanjing: Southeast University, 2006:56-62.

[10] .Operating System Concepts. Zheng Kougen. Translation, 7, Beijing: Higher Education Press. 2010:11-13, 138-147.

[11] Xiong Kaiqi, He Yuxiong. Power-efficent resource allocation in MapReduce clusters Proceedings of the 2013 IFIP/IEEE International Symposium on Integrated Network Management, 2013: 603-608.

[12] Chen Ming. Distributed computing application models. Beijing: Science Press, 2009, 38-46.

[13] J Barnes, P Hut. A hierarchical O (NlogN) force-calculation algorithm. Nature, 1986, 324 (6096):446-449.

[14] Floyd S, Fall K. Promoting the use of end-to-end congestion control in the internet. IEEE/ACM Transcation on Networking, 1999, 7(4): 458-472.

[15] Padhye J, Firoiu V, Towsley D. et al. Modeling TCP throughput: a simple model and its empirical validation. In: Oran D, ed. P roceeding of the SIGCOMM. Vancouver: ACM Press, 1998: 303-314.

[16] Chen Hairui, Wang Hechuang, Li Yinghao, Et al. Security Token and Federated Authentication Based on Elliptic Curve Groups Algorithm. Microelectronics & Computer, 2014, 31(11): 88-91.

[17] Javadi B, Abawajy J H, Akbari M K. Performance modeling and analysis of heterogeneous meta-computing systems interconnection networks. Computers and Electrical Engineering, 2008, 34(6): 488-502.

[18] Yuan Linfeng. A heterogeneous wireless network interconnection strategy based on IP switching. Proc of the 2011 IEEE International Conference on Computer Science and Automation Engineering, CSAE 2011, 2011, 4: 474-477.

[19] Javadi B, Akbari M K, Abawajy J H, Et al. Multi-cluster computing interconnection network performance modeling and analysis. Future Generation Computer Systems, 2009,25(7): 737-746

[20] Tang, C. and Li, M. Location of Wireless Sensor Networks Based On Port Management System. Journal of Discrete Mathematical Sciences and Cryptography, 2018, 21(2):595-599.

[21] Adel, E., El-Sappagh, S., Barakat, S., and Elmogy, M. Distributed Electronic Health Record Based On Semantic Interoperability Using Fuzzy Ontology: A Survey. International Journal of Computers and Applications, 2018, 40(4):223-241.

[22] Ezenwoke, A., Daramola, O. and Adigun, M. Qos-Based Ranking and Selection of Saas Applications Using Heterogeneous Similarity Metrics. Journal of Cloud Computing, 2018, 7(1).