

Optimization of Job Schedule Model Based on Grid Environment

Homer Wu

Department of Information Management, Chinese Culture University, Taipei, TAIWAN
homerwu@faculty.pccu.edu.tw

Chong-Yen Lee, Wu-Yee Chen, and Tsang-Yean Lee

Department of Information Management, Chinese Culture University, Taipei, TAIWAN
{cylee, wuuyee, tylee}@faculty.pccu.edu.tw

Abstract—CPU utilization, throughput, turnaround time, waiting time, and response time are the factor to influence system performance. Every system uses different scheduling algorithms to achieve his goal. In the grid computing, resources usages are important to get good performance. When jobs are not uniformly distributed in all grid nodes, job scheduling is more important. In order to achieve performance optimization, we propose a Maximum Processor Utilization and Minimum Turnaround Time (MPUMTT) scheduling algorithm which maximizes CPU utilization and minimizes turnaround time as well. The algorithm is used in all execute grid nodes to schedule their jobs and get good performance.

Index Terms—job schedule, system performance, optimization

I. INTRODUCTION

The operating system can help users to run their jobs friendly [1-3, 7-13]. The scheduling criteria includes CPU utilization, throughput, turnaround time, waiting time, response time, deadlines, predictability, fairness, enforcing priority and balancing resources[10-11,13]. Different manufactures apply different scheduling criteria to design their operating systems. When jobs are submitted to the computer system, the operating system uses its scheduling algorithm to schedule these jobs without any extra information regarding these jobs. In grid environment, users may access the computational resources at many sites [4]. Resources (e.g., CPUs, storages, etc.) sharing, collaborative processing, reliable and secure connection are important on grid computing architectures. However, each resource of coordination node in the grid environment changes dynamically. How to optimize these resources usage is an important issue. Lee et al. [5, 6] proposed a dynamic analyzing resources model which can receive the information about CPU usages, number of running jobs of each grid node resource to achieve load-balancing and make the plans and allocations of the resources of collaborated nodes optimize. Wu et. al. [14] proposed “A Job Schedule

Model Based on Grid Environment”. Based on this paper, we propose a Maximum Processor Utilization and Minimum Turnaround Time (MPUMTT) scheduling algorithm which obtains maximum CPU utilization and keeps minimum turnaround time to achieve performance optimization. We install this algorithm to all execute grid nodes and supervisor grid node. Before applying MPUMTT scheduling algorithm, job information for all jobs is stored in Job Information History Database (JIHDB) manually or automatically and sends to supervisor to be collected. When a job is submitted, the job information related to the job is retrieved from JIHDB to create an entry in Running Job Information Table (RJIT). MPUMTT scheduling algorithm uses RJIT to schedule all jobs in the execute grid node.

II. JOB DESCRIPTION

In a computer system, jobs processing compilation, testing, or running may be defined as one of the following characteristics:

- (1) One-time running job
- (2) Periodically running job
- (3) Frequently running job

For one-time running jobs, no special handle is needed. For periodically running and frequently running jobs, they should be handled carefully to get best performance. Since they are running more than once, the computer system must know their job information and control. Performance can be improved if better control of these jobs is handled. From different evolution of performance improvement, we conclude the following:

- (1) In order to maximize the average of CPU utilization, CPU must be kept busy.
- (2) Smaller jobs are scheduled to run first and finish quickly can increase the average of throughput.
- (3) Smaller jobs are scheduled to run first and finish quickly can shorten the average of turnaround time.
- (4) Smaller jobs are scheduled to run first and finish quickly can decrease the average of waiting time.

- (5) In order to respond quickly and shorten the average of response time, interactive jobs should be run first.
- (6) CPU-bound jobs and I-O bound jobs must run together to get best system performance.

From the above view point, better performance can be achieved if smaller and shorter jobs are processed first. For the long run, processes of periodically running jobs and frequently running jobs need exclusive consideration. In our research, we consider all first-time running jobs have no history job information (default CM usage, default Turnaround time and ICRatio see later) to schedule. When any job has finished, we have information such as the starting time, the ending time, CPU usage, IO activity, and memory used of the job. This information is used to create entries of Job Information Table. TABLE 1 is an example of the Job Information Table to store job information.

TABLE 1
JOB INFORMATION TABLE

Job Name	Date	Starting Time	Ending Time	CPU Usage	IO Activity	CM Usage
----------	------	---------------	-------------	-----------	-------------	----------

The information in TABLE 1 can then used to create entries in JIHDB. TABLE 2 shows the result from TABLE 1. Turnaround Time = (Ending Time - Starting Time) and ICRatio = (Turnaround Time / CPU Usage). The value of ICRatio is greater or equal to 1. The smaller the ICRatio value means more intensive used of CPU (CPU bound job) and the larger the ICRatio value means less CPU used and I/O access is critical (I/O bound job).

TABLE 2
JOB INFORMATION HISTORY DATABASE

Job Name	Turnaround Time	CPU Usage	CM Usage	ICRatio
----------	-----------------	-----------	----------	---------

The job information is collected through one of the following method:

- (1) Run the job alone and record the needed information.
- (2) When a job is finished, the new information of the job is used to update JIHDB.

If the optimal performance of computer system required, CPU bound and I/O bound jobs should be run together. In the computer system, when a job is entered, its record is found in JIHDB. If the record exists, a related entry of the job is created in Run Job Information Table (RJIT). If the record does not exist, we use default parameters to create a RJIT entry. TABLE 3 is an example of RJIT. RJIT is used along with MPUMTT scheduling algorithm to schedule all jobs in the computer system. With MPUMTT scheduling algorithms, the values of Run and Part fields will be inserted in RJIT.

TABLE 3
RUN JOB INFORMATION TABLE

Job Name	Turn-around Time	CPU Usage (sec.)	CM Usage (KB)	ICRatio	Run	Part	CPU Used
----------	------------------	------------------	---------------	---------	-----	------	----------

III. FRAMEWORK OF THE PROPOSED MODULE

In this section, we present the framework of the proposed scheduling model based on grid environment. Grid nodes are divided into supervisor grid node (S_0), supervisor backup grid node (B_1), and execute grid nodes (X_i). There are two modules, Supervisor Job Schedule Module (SJSM) and Execute Job Schedule Module (EJSM). In the supervisor grid node runs SJSM and both the backup supervisor and all execute grid nodes run EJSM as shown in Figure 1.

A. Supervisor Grid Node

Functions of the supervisor grid node are described as follows:

- (1) Processing new jobs.
- (2) Performing job scheduling.
- (3) Processing transferred jobs.
- (4) Receiving completion of transferring jobs.
- (5) Collecting job information.
- (6) Collecting grid information.
- (7) Updating grid node information and total account information to the log file.

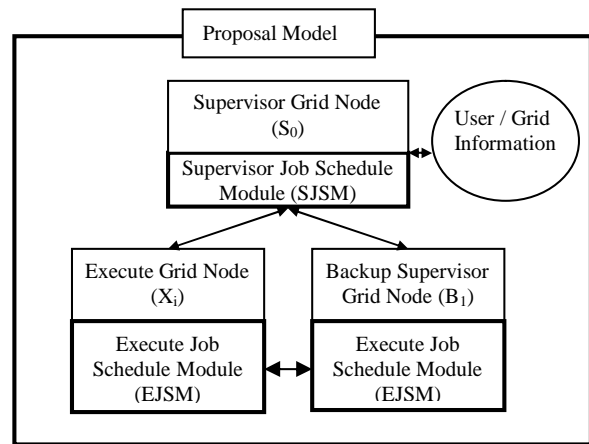


Figure 1. Framework of the proposed model

SJSM in the supervisor grid node contains supervisor message receiving module (SMRM), supervisor job scheduling module (SJSJ), supervisor job transferring module (SJTJ), supervisor job completing module (SJCJ), supervisor collecting information module (SCIM) and supervisor message sending module (SMSM) as shown in Figure 2.

Functions of these modules are described as follows:

- (1) SMRM receives information from local or other execute grid node. It calls SJSJ to receive job entering. It calls SJTJ to receive transferring job.

It calls SJCM to process job completion. It calls SCIM to collect grid information.

- (2) SJSM processes new jobs entering. It calls MPUMTTNJE (Maximum Processor Utilization and Minimum Turnaround Time algorithm for new job entering) and updates SRJIDB and SJIHDB.
- (3) SJTM processes job transferring received. It has following components.
 - i. Supervisor job transfer receiving component (SJTRC) receives jobs transferred from all execute grid nodes. It calls SNJPC (Supervisor New Job process component), if the loading of supervisor is low, otherwise it calls SJTC (Supervisor Job Transfer Component).
 - ii. SNJPC processes this job. It calls MPUMTTNJE (Maximum Processor Utilization and Minimum Turnaround Time algorithm for new job entering) and updates RJIDB and SJIHDB.
 - iii. SJTC checks SGIDB to find the best grid node to perform transferring and updates SJTDB.

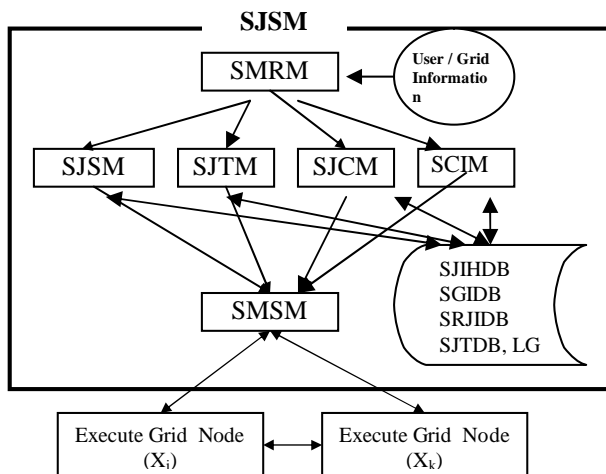


Figure 2. Architecture of the SSM

- (4) SJCM processes job completion. It calls SJCC (Supervisor Job Completion Component) to process job finished from supervisor or calls SJCRC (Supervisor Job Completion Return Component) to process job completion returned from execute grid node. It has the following components..
 - i. SJCC calls MPUMTTJF (Maximum Processor Utilization and Minimum Turnaround Time algorithm for job finished), which is an algorithm to handle the job is finished and updates ERJIDB and writes the information to log file.
 - ii. SJCRC updates SJTDB and sends the job to the requested grid nodes and writes the information to log file..

- (5) SCIM processes grid information received. It calls SRGIC (Supervisor Receive Grid Information Component) to receive grid information or calls SRJIC (Supervisor Receive Job Information Component) to receive job information from supervisor and execute grid nodes.
 - i. SRGIC receives grid information from supervisor and execute grid nodes and updates SGIDB and writes the information to log file.
 - ii. SRJIC receives job information from supervisor and execute grid nodes to update SJIHDB and writes the information to log file.

- (6) SMSM sends messages to all execute grid nodes.

B. Execute Grid Node

Functions of the execute grid node are described as follows:

- (1) Processing new jobs.
- (2) Sending grid node information to supervisor.
- (3) Sending job information to supervisor n.
- (4) Sending transferring jobs to supervisor.
- (5) Receiving completed jobs from supervisor.
- (6) Receiving transferring jobs from supervisor.
- (7) Returning completed jobs back to supervisor.

EJSM resided in all execute grid nodes contains execute message receiving module (EMRM), execute job schedule module (EJSM), execute job transfer module (EJTM), execute job completion module (EJCM), execute send information module (ESIM) and execute message send module (EMSM) as shown in Figure 3.

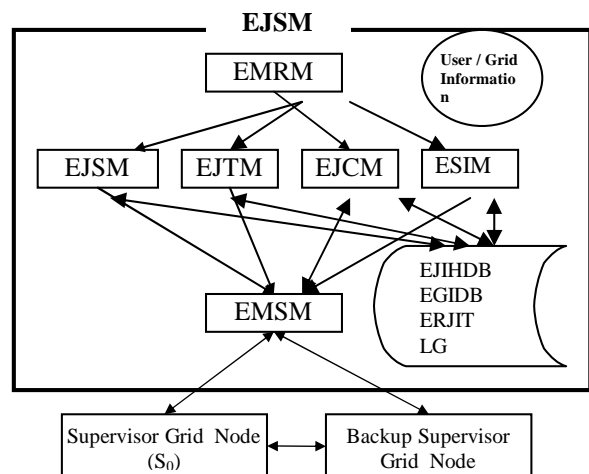


Figure 3. Architecture of EJSM

Functions of these components are described as follows:

- (1) EMRM receives jobs and information from local or supervisor grid node. It calls EJSM to receive job entering from local or supervisor. It calls EJTM to receive transferring job. It calls EJCM

to process job completion. It calls ESIM to send grid information..

- (2) EJSM processes new jobs entered. New job may be local or transfer from supervisor. Functions of the module are:
 - i. Execute new job process component (ENJPC) calls MPUTNJE, which is an algorithm when a new job is entered and updates ERJIDB and EJIHDB.
 - ii. Execute job transfer received component (EJTRC) receives jobs transferred from supervisor. When loading is low, the execute grid node can accept the job of other execute grid nodes. Process ENJPC.
- (3) EJTM processes transferring jobs to supervisor grid node. Functions of the module are:
 - i. Execute job transfer component (EJTC) finds some jobs to send to supervisor to process, when the load of the execute grid node is too heavy. It updates job send field of ERJIDB.
- (4) EJCM processes jobs completion from local or supervisor grid node. Functions of the module are:
 - i Execute job completion process component (EJCPC) calls MPUTJF and updates ERJIDB.
 - ii Execute return job transfer component (ERJTC) receives completed jobs from supervisor when transferred jobs are completed. It updates ERJIDB.
 - iii Execute job transfer completion return component (EJTCRC) calls MPUTJF and updates ERJIDB. It returns completed transferring jobs to supervisor.
- (5) ESIM processes information sending. Functions of the module are:
 - i. Execute job information sending component (EJISC) receives job information and update EJIHDB and sends to supervisor.
 - ii. Execute run job information sending component (ERJISC) sends execute run job information table (ERJIT) and execute grid information database (EGIDB) to supervisor.
- (6) EMSM sends messages to supervisor grid node.

C Backup Supervisor Grid Node

Functions of backup supervisor grid node are described as follows:

- (1) Performing functions as execute grid node normally.
- (2) Performing functions of supervisor grid node when supervisor grid node is inactive.

Backup supervisor grid node works as the execute grid node normally. When it does not receive any message from supervisor for a finite period, it sends an inactive message to supervisor grid node to make sure supervisor is active. If it does not receive any response from the

supervisor in a certain period, this means that the supervisor is inactive. It sends supervisor change message to inform all execute grid nodes that backup supervisor is the current supervisor until the original supervisor grid node is available again. From the priority of grid node capacity information, it can select new backup supervisor grid node.

IV. DATABASE AND TABLES

The database and tables in both supervisor and execute grid nodes are described in the following two sections.

A. Data in supervisor grid node

The relative information in supervisor grid node is as follows:

- (1) Supervisor job information history database (SJIHDB): Job information from all grid nodes (include supervisor) is used to create JIHDB as TABLE 4.

TABLE 4
SUPERVISOR JOB INFORMATION HISTORY DATABASE (SJIHDB)

Job Name	Grid Node Name	Turnaround Time	CPU Usage	CM Usage	ICRatio
JOB A	GRID1	2385	190	80	12.55
JOB B	GRID2	600	140	100	4.28
JOB C	GRID3	2070	310	70	6.68
JOB D	GRID1	1200	620	120	1.93
JOB E	GRID2	2565	70	200	36.64
JOB F	GRID3	240	160	180	1.50

- (2) Supervisor run job information table: An entry in the run job information table is created when a job is entered. Fields of the table are as TABLE 5.

TABLE 5
SUPERVISOR RUN JOB INFORMATION TABLE (SRJIT)

Job Name	Turnaround Time	CPU Usage (sec.)	CM Usage (KB)	ICRatio	Run	Part	CPU Used	Job Send	Job From
JOB F	240	160	180	1.50	R	5	46		GRID3
JOB B	600	140	100	4.28	R	4	32		GRID2
JOB 1			60	5.00		3			
JOB 2			60	5.00	R	3	3		
JOB C	2070	310	70	6.68	R	2	18		GRID3
JOB A	2385	190	80	12.55		1			GRID1
JOB E	2565	70	200	36.64	R	1	26		GRID2

- (3) Supervisor grid information database: Execute grid nodes periodically send grid information to supervisor. Supervisor creates supervisor grid information database as TABLE 6.

TABLE 6
SUPERVISOR GRID INFORMATION DATABASE (SGIDB)

Node Name	Job Count	CPU Usage	Running Count	Completed Count
-----------	-----------	-----------	---------------	-----------------

Where Node Name is the name of this node, Job Count is total number of jobs in the node, CPU Usage is total CPU time used in this period, Running Count is total number of jobs running in this period, and Completed Count is the total number of jobs completed in this period.

- (4) Supervisor job transfer database: The content of this database is listed as TABLE 7.

TABLE 7
SUPERVISOR JOB TRANSFER DATABASE (SJTDB)

Job Name	Time Begin	CPU Usage	CM Usage	ICRatio	File Size	Transfer to Node	Transfer From Node
----------	------------	-----------	----------	---------	-----------	------------------	--------------------

Where Job Name is the name of the node, Time Begin is the time of job entered, CPU Usage is total CPU time used in this period, CM Usage is the required memory used, ICRatio is IO/CPU ratio, File Size is total size of used files, Transfer to Node is node to be transferred, Transfer From is job accepted to process from this node.

B. Data in execute grid nodes

Data and relative information in execute grid nodes are as follows:

- (1) Execute job information history database: the job information in execute grid nodes is used to create EJHDB as TABLE 8 and send it to supervisor grid node.

TABLE 8
EXECUTE JOB INFORMATION HISTORY DATABASE (EJHDB)

Job Name	Grid Node Name	Turnaround Time	CPU Usage	CM Usage	IC Ratio
JOB_A	GRID1	2385	190	80	12.55
JOB_D	GRID1	1200	620	120	1.93

- (2) Execute run job information table (ERJIT): when a job is entered, an entry in the ERJIT is created. Fields of the table are as TABLE 9.

TABLE 9
EXECUTE RUN JOB INFORMATION TABLE (ERJIT)

Job Name	Grid Node Name	Turnaround Time	CPU Usage (sec.)	CM Usage (KB)	ICRatio	Run	Part	CPU Used	Job Sent	Job From
JOB_I	GRID3	1200	620	120	1.93	R	2	18		
JOB_J			60	500	F	3	3			
JOB_A	GRID1	2385	190	80	12.55		1			

- (3) Execute grid information database (EGIDB): execute grid nodes collect the information as TABLE 10 from ERJIT and sends grid information to the supervisor.

TABLE 10
EXECUTE GRID INFORMATION DATABASE (EGIDB)

Node Name	Job Count	CPU Usage	Running Count	Completed Count
-----------	-----------	-----------	---------------	-----------------

Where Node Name is the name of this node, Job Count is total number of jobs in the node, CPU Usage is total CPU time used in this period, Running Count is total number of jobs run in this period, and Completed Count is the total number of jobs completed in this period.

V. MPUMTT SCHEDULING ALGORITHM

Maximum Processor Utilization and Minimum Turnaround Time (MPUMTT) scheduling algorithm works under the following conditions:

- (1) The computer system is a single CPU system.
- (2) The system will provide the value of available CM.
- (3) Jobs are loaded in continuous memory.

In order to achieve the system performance optimization, CPU bound and I/O bound jobs need to run balanced and the larger the number of running jobs the better. The computer system must keep the history of job information for analysis purpose. Job information used to create JIHDB can be found in system log file or can be generated manually. When a new job is entered, its information is found in JIHDB and used to schedule. From JIHDB, turnaround time and CPU usage are used to compute ICRatio (= turnaround time/CPU usage). Jobs are divided into MAXPART parts. When Part=1 means ICRatio is the smallest and these jobs use more CPU. When Part=MAXPART means ICRatio is the highest and these jobs use more I/O and less CPU. MPUMTT scheduling algorithm is described as follows:

A. Algorithm of first initial process (MPUMTTI)

When the computer system is initialized, the following two steps are in action.

Step 1: First initial process

- Set Muse to 0.
- Sort RJIT by using ICRatio as the first sorting key (in ascendant order) and Turnaround Time as the secondary sorting key (in ascendant order) and CM Usage as the third sorting key (in ascendant order)
- Set TotJob to the number of the total jobs in RJIT
- Set NoJobPerPart to TotJob divide by MAXPART and set NoRem to the remainder
- Store 1 to the field Part of first NoJobPerPart (+1, if NoRem >0) of RJIT
- Store 2, 3, ..., MAXPART to relative Part fields in RJIT.
- Sort RJIT by using Part as the first sorting key (in descending order) and Turnaround Time as the secondary sorting key (in ascendant order) and CM Usage as the third sorting key (in ascendant order)

```

Store relative numbers of jobs to
  NoJobPart[1:MAXPART]
Set NoRun[1:MAXPART] to zeros
Clear Run field in RJIT
Set DefPart to INT(MAXPART+1)/2)
Set DefRatio to 5
Set DefCMusage to 60 (60k memory)
Set PartSet[1:MAXPART] to zero
Set PartFlag[1:MAXPART] to zero
Step 2: Set to run in memory
For i from 1 to NoJobPart[1]
  If all PartFlag[1:MAXPART] are set Then
    exit
  Else
    Set L to i
    For j from 1 to MAXPART
      If PartFlag[j] is not set Then
        If PartFlag[L] is not set Then
          If i <= NoJobPart[L] Then
            If CM Usage[L] <= available
              CM Then
                Set NoRun[j] to NoRun[j] + 1
                Set Run [L] to 'R'
            EndIf
          EndIf
        EndIf
      EndIf
    Set L to L+NoJobPart[j]
  Exit j
Exit i

```

B. Algorithm of new job entered (MPUMTTNJE)

When a new job is entered, the following steps are processed.

```

Step 1: Check if this job has an entry in JIHDB
Step 2: If it is not exist then go to step 3
  Get values of Turnaround Time, CPU Usage,
  and CM Usage
  Set ICRatio to Turnaround Time / CPU Usage
  Find the corresponding Part from ICRatio
  Go to Step 4
Step 3: Set Part to DefPart and CM Usage to
  DefCMusage
  Set ICRatio to DefRatio
Step 4: From size of CM usage and insert this job
  information to RJIDB
  Increase NoJob[Part] by 1

```

C. Algorithm of job finished (MPUMTTJF)

When a job is finished, its Part is recorded. New value of Muse is set to (M – available CM) and NoRun[Part] is decreased by 1. Part is scanned to find new job to be loaded for execution. If no job is found, we find the Part with the smallest NoRun[Part] to start. The steps are as follows:

```

Step 1: Write job statistics to the log file
  Decrease NoRun [Part] by 1
  Decrease NoJobPart[Part] by 1
  Delete the job entry in RJIDB.
  Find smallest NoRun[Part]

```

```

Set PartSet[1:MAXPART] to zero
Step 2: Find entry i such that the CM Usage[i] of this
  Part < available CM
  If it exists then
    Set NoRun[Part] as NoRun [Part] +1
    Set Run [I + begin of this Part] to 'R'
  Else
    Set PartSet[Part] to 1
  EndIf
Step 3: If all PartSet are set Then go to step 4
  Find NoRun[Part] is the smallest which
  PartSet[Part] has not been set
  Go to step 2
Step 4: Clear PartSet to zero

```

D. Algorithm of adjustment(MPUMTTAD)

The algorithm is processed periodically (ex. for every 30 minutes). It contains several steps:

```

Step1: Sort RJIT by using IC Ratio as the first
  sorting key (in descending order) and
  Turnaround Time as the secondary sorting
  key (in ascendant order) and CM Usage as the
  third sorting key (in ascendant order)
Step 2: Set TotJob to the number of the total jobs in
  RJIT
Step 3: Set NoJobPerPart to TotJob divide by
  MAXPART and set NoRem to the remainder
Step 4: Store 1 to the field Part of first NoJobPerPart
  (+1, if NoRem >0) of RJIT
Step 5: Store 2, 3...MAXPART to relative Part field
  in RJIT
Step 6: Sort RJIT by using Part as the first sorting key
  (in descending order) and Turnaround Time
  as the secondary sorting key (in ascendant
  order) and CM Usage as the third sorting key
  (in ascendant order)
Step 7: Store relative number of jobs to
  NoJobPart[1:MAXPART]
Step 8: For each Part, Count NoRun [Part] if Run
  field is 'R'

```

VI. DISCUSSION

When this MPUMTT scheduling algorithm is installed in all grid nodes, we conclude the following:

- (1) Job information history database in both supervisor and execute grid nodes must be unique name to access.
- (2) An execute grid node periodically sends the current grid information to the supervisor to inform the supervisor its current state. When any of its running jobs has changed, it also sends the information to the supervisor to update.
- (3) When the load of the execute grid node is too heavy, it sends some jobs to supervisor and supervisor finds the best execute grid node from grid information database to accept these jobs to process.
- (4) When the load is low, the execute grid node can accept transferred job from supervisor.

- (5) MPUMTT adjust algorithm can run period to redefine job parts, because jobs change of completion and new job entering.
- (6) Supervisor and backup supervisor grid nodes can run as execute grid node also.
- (7) Execute grid node can decide to transfer jobs to assigned grid node, because used files are stored there.
- (8) The performance of the system can be measured from the log file.
- (9) Some jobs do not have chance to be processed, the algorithm can then be modified by decreasing ICRatio of these jobs periodically.

REFERENCES

- [1] C. Crowley, *Operating Systems: A Design-Oriented Approach*, Irwin Publishing, Chicago, IL, 1977.
 - [2] W. S. Davis and T. M. Rajkumar, *Operating Systems: A Systematic View*, 6th Edition, Addison-Wesley, Reading, MA, 2004.
 - [3] I. Englander, *The Architecture of Computer Hardware and Systems Software: An Information Technology Approach*, John Wiley & Sons, Inc., New York, NY, 1996.
 - [4] I. Foster, C. Kesselman, and A. Globus, "Metacomputing Infrastructure Toolkit", *International Journal of Supercomputer Application*, Vol.11, No.2, pp.115-128, 1997.
 - [5] H.-M. Lee, T.-Y. Lee, C.-H. Yang, and M.-H. Hsu, "An Optimal Analyzing Resources Model Based on Grid Environment," *WSEAS Transactions on Information Science and Applications*, Vol.3, No.5, pp.960-964, 2006.
 - [6] H.-M. Lee, T.-Y. Lee, and M.-H. Hsu, "A Process Schedule Analyzing Model Based on Grid Environment," *KES 2006*, Part III. LNAI 4253, Springer-Verlag, Berlin Heidelberg, pp.938-947, 2006.
 - [7] G. Nutt, *Operating Systems: Modern Perspective*, 3rd Edition, Addison-Wesley, Reading, MA, 2004.
 - [8] J. R. Pinkert and L. L. Wear, *Operating Systems, Concepts, Policies, and Mechanisms*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1989.
 - [9] W. Shay, *An Introduction to Operating Systems*, Harper Collins College Publishers, New York, NY, 1993.
 - [10] A. Silberschatz, P. B. Galvin, and G. Gagne, *Operating System Principles*, 7th Edition, John Wiley & Sons (Asia) Pte Ltd, 2004.
 - [11] W. Stallings, *Operating System: Internals and Design Principles*, 3rd Edition, Prentice-Hall, Inc., Upper Saddle River, NJ, 1998.
 - [12] H. S. Stone, *High-Performance Computer Architecture*, Addison-Wesley, Reading, MA, 1993.
 - [13] A. S. Tanenbaum, *Modern Operating Systems*, 2nd Edition, Prentice-Hall, Inc., Upper Saddle River, NJ, 2001.
 - [14] Homer Wu, Chong-Yen Lee, Wu-Yee Chen, Tsang-Yean Lee "A Job Schedule Model Based on Grid Environment," First International Conference on Complex, Intelligent and Software Intensive Systems. Vienna, Austria 2007/4/10 - 2007/4/13, pp. 43-52.
- Homer Wu**, born in Taiwan on September 1, 1951, earned his Master of Science degree in management information systems from United States International University (Alliant International University), San Diego, California, USA in 1986.
- He was Director of Computer Center at Chinese Culture University of Taiwan 1980-85 and 1988-94, Consultant of Barits Securities Corporation of Taiwan 1997-2001, and Consultant of Golden Bridge Electech of Taiwan 2005-07. He is currently an Instructor in the Department of Information Management at Chinese Culture University of Taiwan.
- Chong-Yen Lee**, born in Taiwan on July 8, 1960, obtained his Master of Science degree in computer science from North Dakota State University, Fargo, ND, USA in 1987 and Ph.D. degree in computer science from Illinois Institute of Technology, Chicago, IL, USA in 1992.
- He was Chairman of the Department of Information Science at Chinese Culture University of Taiwan 1995-98 and Information Security Committee member of Taiwan Cooperative Bank of Taiwan 2003-07. He is currently an Associate Professor in the Department of Information Management at Chinese Culture University of Taiwan. His research interests include computer and information security, systems design, database management, knowledge management, and data mining.
- Wuu-Yee Chen** received his Ph.D. degree in management information systems from Drexel University at Philadelphia, USA in 1996.
- He is currently an associate professor in the Department of Information Management at Chinese Culture University of Taiwan. His research interests are Internet marketing, data mining, and knowledge management.
- Tsang-Yean Lee** received his Master degree in electrical engineering from National Taiwan University at Taipei, Taiwan in 1959.
- He is currently an associate professor at the Department of Information Management at Chinese Culture University of Taiwan. His research interests are operating system, information security, and grid computing.