Malaysian Journal of Computer Science, Vol. 12 No. 2, December 1999, pp. 27-36

PRACTICAL EXPERIENCES WITH TASK SCHEDULING STRATEGIES FOR IMAGE PROCESSING APPLICATION ON HETEROGENEOUS DISTRIBUTED COMPUTING SYSTEM

Kalim Qureshi and Masahiko Hatanaka
Dept. of Computer Science and Systems Engg., Muroran Institute of Technology
Mizumoto cho 27-1, Muroran , 050-8585 Hokkaido, Japan
Tel: 81-143-46-5427
Fax: 81-143-46-5499
email: qur@wave.csse.muroran-it.ac.jp

ABSTRACT

Heterogeneous Distributed Computing (HDC) system consists of Workstations (WSs) and Personal Computers (PCs). In HDC system, each WS/PC may have different processor and performance. In order to take advantage of this diversity of processing power of a system, an effective task partitioning, scheduling, and load balancing are needed to get the optimum performance.

This paper examines the effectiveness of task partitioning and scheduling strategies for image (raytracing) processing application on HDC system. The static and dynamic/Run-time Task Scheduling (RTS) strategies are shown inadequate for balancing the load of HDC system. Two adaptive tasks scheduling strategies are proposed for HDC image computing system. The adaptive strategies are: i) Master Initiated Sub-task size (MIS) strategy (based on centralized resources management approach), and ii) Worker Initiated Sub-task size (WIS) strategy (based on semi-decentralized resources management approach). Performances of all investigated strategies are evaluated on manager/master and workers model of HDC system.

Keywords: **Distributed image/raytracing computing, Task partitioning and scheduling, Load balancing, Heterogeneous distributed computing, Performance evaluation**

1.0 INTRODUCTION

Graphics rendering is notoriously computation intensive application, particularly for visualization of realistic images and fast updates demand. The applications, such as scientific visualization, CAD, vehicle simulation, and virtual reality can require hundreds of Mflops computing power, that is in many cases, far beyond the capabilities of a single WS/PC. It has been realized that massively parallel computers can be an effective way to accelerate these computations [1, 2].
Two types of parallelism are quite evident while generating raytraced images. The first is the object base image partitioning and the second is the pixel/horizontal scan-line based image partitioning. The first data parallelizing approach requires a possibly large amount of communication during raytracing, because information about other subset of objects are needed to perform the computation. In the second task parallelizing approach, each processor/worker must have the copy of complete scene description, so that it can access all the information about the rays involved in and determining the color of a pixel. Nevertheless, the second approach is an ideal candidate for achieving high speedups, but it is better than the first one [3]. Therefore, it is used in this investigation.

The advent of commodity based distributed computing environment has made it possible to process such kind of images in reasonable time on modest budgets. For better utilization of the computational powers of all processors/ remote-machines in HDC system, the efficient application partitioning and scheduling are necessary. In general, the purpose of partitioning operation is to split the task (image) evenly among all processors/workers in the HDC system. Whereas the approach of efficient task scheduling and load balancing are to ensure that no machine stay idle until whole application is completed.

In HDC system, the static and dynamic/RTS strategies are used to balance the loads among the WSs/PCs in HDC system. Since WSs/PCs have performance variation characteristic [4, 5], therefore, the static task distribution is not effective for HDC system [6, 7]. RTS strategy can achieve nearly perfect load balancing [8], because of the machines' performance variations and non-homogeneous nature of the application (image) adjusted at runtime [9]. The RTS strategy performance depends upon the size of the sub-task. If sub-task size is too small then it generates a serious inter-process communication overhead. In other case, if the sub-task size is too large then it may create a longer master/client machine waiting time due to the inappropriate sub-tasks size of slow performance machine [9]. Many researchers suggest enhancement in adaptive task scheduling strategies for homogeneous distributed system [10-12]. However, these strategies do not work well for HDC system without further modifications. The crucial point is that these are based on fixed parameters that are tuned for the specified hardware. In heterogeneous systems, this tuning is often not possible because both the computational power and the network bandwidth are not known in advance, which may change unpredictably during runtime.

The strategies are based on task distribution and then task migration from heavily loaded to light-loaded workers are expressed [13, 4]. The task migration has two serious drawbacks [14]:

i) All workers should continuously monitor the status of other workers
ii) During the computations, a worker has to search its load and float the information on the network, hence, this produces a large amount of communication overhead.

In contrast to all above, two adaptive strategies are developed for HDC system with carefully keeping in mind that, the strategies should not be complex, their overhead involved may not negate their benefit, and should support for large HDC system. The adaptive strategies are:

- Master (client machine) Initiated Sub-task size (MIS) strategy
- Worker (server machine) Initiated Sub-task size (WIS) strategy.

The several enhancements are proposed in both strategies, i.e. the strategies have adaptive nature, they reduce inter-process communication time overhead, and effectively balance the load among the workers.

Previous work: For the last ten years, task partitioning, scheduling, and load balancing problems for homogeneous parallel distributed systems have been thoroughly studied. However, these topics are relatively new for HDC platforms and have been investigated less frequently. The eight different task partition strategies for images in homogeneous distributed computing system are presented in [15] and the improved work is presented in [16], where: *The authors suggested to partition the whole image into two parts as upper and lower half of the image. The upper half was evenly distributed among all worker processes. The lower half was dynamically distributed to the workers, as worker became idle.*

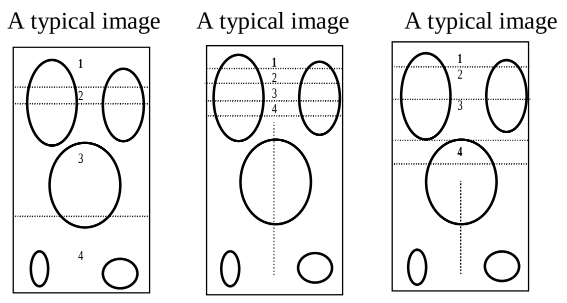
The authors' image partitioning and scheduling was unadaptive and they did not clarify about how much part of the image is fixed for upper and lower partitions. If upper partition is more then the serious load imbalance may occur at the end of the application due to bigger sub-task size for low performance machine. If lower partition is larger, then the number of requests to be performed by the master increases, that may cause the overhead of inter-process communication time. They also have not expressed the load balancing mechanism in their proposed strategies, which are proposed in this paper.

This paper is an extension of our work [17] in which two adaptive task partitioning and scheduling strategies were studied where conclusion was same as that of authors in [15, 16].

The main features of MIS and WIS strategies are as follows:

i) Adaptive task sizing according to the machine's performance
ii) Both of the strategies avoid the master machine waiting time at the end of the application
iii) Reduction in number of requests made by the master machine for tasks distribution.

In MIS strategy, all workers' performance and sub-task size for the workers are estimated by master machine. Where as in WIS strategy, each worker has authority to estimate its next sub-task size without needing other workers state information and to suggest its sub-task size to the master machine while returning results. Since the HDC system environment is commonly available and it is rapidly growing to large number of WSs/PCs, for that system the performance verses scalability is more important. Therefore, in this paper, the effects of centralized and semi-decentralized resources management and decision about the task scheduling on HDC system are practically examined. The static and RTS strategies are included in this investigation to practically identify and improve their deficiencies in proposed adaptive strategies.



a) Image partitioning in static strategy
b) Image partition-ing in RTS strategy
c) Image partitioning in adaptive MIS/ WIS strategy

Fig. 1: Tasks mapping among four workers: a) Static, b) RTS, and c) adaptive (MIS/WIS) strategies

2.0 TASK PARTITIONING AND SCHEDULING

The distributed raytracing program that consists of several program modules or tasks that are free to reside on any WS/PC in a HDC system. The goal of the task/program partitioning and scheduling is to minimize the overhead caused by inter-task communication time, while preserving the highest possible degree of parallelism. The descriptions of the investigated strategies are here and mathematical symbols used in strategies are defined in Table 1.

Table 1: Symbols used in strategies and their definitions

Symbol	Definition
T	Total number of tasks (640 horizontal scan-lines of the image).
i	Worker number.
t_i	Worker's assigned tasks or computed sub-task size.
$t_{initial(i)}$	Worker's initial sub-task size.
T_b	Total number of unprocessed tasks $(T_b = T - t_i)$.
W_{pi}	Worker's estimated current performance.
W_{pui}	Worker's performance at unloaded condition.
W_{puslow}	Slowest worker's performance at unloaded condition.
W_{qui}	Worker's job queue length at unloaded condition.
W_{qi}	Worker's current job queue length.
W_t	Total number of workers in HDC system configuration.
W_{pdp}	Sum of all workers' estimated performance in HDC system.
W_{npi}	Worker's normalized performance.
F_i	Worker's sub-task size multiplication factor.
p_i	Worker's processing time per scan-line.
W_{pk}	Total number of workers in HDC system configuration who's estimated performances are known or the worker who completes the first assigned task.
W_{npslow}	Slowest worker's normalized performance in HDC system configuration.
R_i	Worker's job request.

2.1 Static Task Partitioning and Scheduling Strategy

The static tasks distribution strategies' chief advantage is their simplicity, there is no need to collect current system state information. Their disadvantage is that they cannot respond to changes in system states, so the performance improvement they provide is limited [14].

Since the heterogeneous HDC system is composed of WSs/PCs of different performance machines, therefore, the equal task partitioning is not feasible for such type of system. The Optimum Task Distribution (OTD) rule is used to divide the task optimally among the workers at compile time using the pre-measured workers' performance. The machines' performances are measured by well-known UNIX BYTE benchmark. The OTD rule is defined as below:

$$W_{pi} = \frac{1}{P_i} \tag{1}$$

Where P_i is the processing time taken by the i th worker to process a specified program.

Sum of all workers estimated performance:

$$W_{pdp} = \sum_{i=1}^{W_t} W_{pi} \tag{2}$$

The worker's normalized performance is computed by:

$$W_{npi} = \frac{W_{pi}}{W_{pdp}} \tag{3}$$

Optimum task distribution rule is used to divide the task among all workers in a HDC system:

$$t_i = T * W_{npi} \tag{4}$$

2.2 Runtime Task Scheduling (RTS) Strategy

A unit of task (one horizontal scan-line of the image) is distributed at runtime. As the worker completes the previous assigned sub-task, new sub-task is assigned for processing. Viewing Fig. 1 can easily differentiate the task mapping mechanism of static and RTS strategies.

2.3 Master Initiated Sub-Task (MIS) Size Scheduling Strategy

The MIS strategy is composed of following five components.

1) Worker's performance estimation: The workers performance is estimated from their runtime-processing rate per scan-line. As a result, the estimated performance of i th worker (W_{pi}), to be the inverse of time taken by the worker to process the unit of scan-line, in P_i second.

$$W_{pi} = \frac{1}{p_i} \tag{5}$$

Initially, the HDC application task scheduling starts with a unit of scan-line and processing time of unit of task is used to estimate the worker's performance, which is further, used for worker's normalized performance and sub-task size computations. It is obvious that some of the workers process the first assigned task very rapidly and others may take long time. For computing the worker's sub-task size, the worker's normalized performance is needed to compute. For this purpose, we have to compute the sum of all workers' estimated performance in the current HDC system. The total number of workers who complete the first assigned task and their estimated performance are expressed by (W_{pk}) and the estimated performance of the workers who still could not complete a single task replaced by the least known worker performance in the current HDC configuration. Let the sum of all workers estimated performances in HDC system be defined as:

$$W_{pdp} = \sum_{i=1}^{W_{pk}} W_{pi} + \sum_{i=1}^{W_t - W_{pk}} W_{pslow} \qquad (6)$$

The worker's normalized estimated performance is expressed as:

$$W_{npi} = \frac{W_{pi}}{W_{pdp}(1/W_t)} \qquad (7)$$

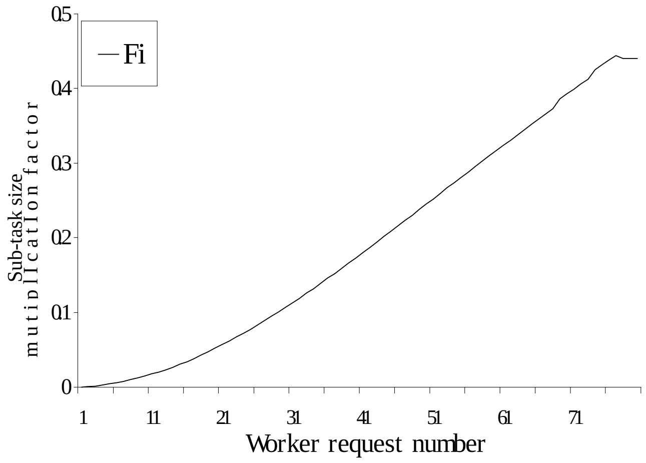


Fig. 2: Sub-task size multiplication factor versus worker's requests number

2) Worker's sub-task size multiplication factor: From previous experiences [9, 17], the optimum performance can be obtained from the HDC system. If the tasks are scheduled in such a way that at the starting of a parallel application the sub-task size of each worker increases gradually according to the worker's performance up to a certain limit (70% of the total tasks scheduled) and then each worker's task size decreases gradually. To implement this phenomena automatically, a sub-task size multiplication expression using the worker's job request number (R_i) information and total number of workers whose performance are known (W_{pk}) in the HDC system. Basically it is a tuning factor for controling the gradual increase and decrease in computed sub task size of each worker. This factor is tuned for this applications and system. The sub-task size multiplication factor is defined as:

$$F_i = \frac{R_i^{\,2}}{R_i^{\,2} + W_{pk}^{\,3}} \qquad (8)$$

If a single value of W_{pk} is used in this formula, in that case the F_i increases rapidly in few requests of the worker and becomes saturated. Effectively this increases the sub-task size of worker instantly at the starting of the HDC application, which creates a load imbalance at the end of the application. To avoid this load imbalance, a cubic value of W_{pk} is used, therefore, each worker task increases and decreases moderately. Let us consider the case, when HDC system is composed of 20 workers. The computed F_i values of highest performance worker corresponding to its job request number is shown in Fig. 2.

3) Worker's sub-task size estimation: The sub-task size estimation formula is derived from the practical experiences on parallel distributed raytracing application [17]. The assumptions for the derivation are given below:

i) The sub-task size of a worker (t_i) is found to be proportional to the total number of balance tasks (T_b - t_i) and inversely proportional to the total number of worker (W_t) in HDC system.
ii) The sub-task size of a worker (t_i) is found to be proportional to the estimated normalized performance of the worker (W_{npi}).

Therefore the worker's sub-task size (t_i) is defined as:

$$t_i = \frac{T_b - t_i}{W_t} * W_{npi} * F_i \qquad (9)$$

4) Slow worker's omission: The HDC system is composed of co-operative computing resources. Its performance also depends on the other user's application load. It is observed on several occasions that during the execution of parallel application, some users may give extensive computing task to the machine or without notices owner shutdown the machine. In these circumstances, the machine's performance degrades/not available, which creates a long waiting time for the master/client machine at the end of the parallel application. To avoid such type of waiting, the following checkpoint is included in the proposed strategy:
After 70% of the total task completion, the manager checks those workers, which still have not completed the first assigned task. The manager marks that worker "as not available" in the current HDC configuration and adds its assigned task (t_i) to the unprocessed task balance (T_b). It eliminates serious manager waiting time that may occur at the end of application.

5) Sub-task duplication: When all the tasks of the application are distributed to the workers, then no task is left to assign further to idle worker. The faster worker finishes the last assigned task earlier as compared to the slow workers, the slow worker very often creates a manager waiting time. To avoid such type of waiting, the following slow worker's task duplication mechanism is used in this strategy:
The manager searches the least performance worker in the current HDC configuration and duplicates its task to the next idle worker. It creates a competition between two workers, the result of the worker who completes the task first is considered and that of the other is ignored. Each least worker's task is duplicated only once. So all computing resources are utilized at the end of application. This also eliminates the serious manager waiting time that may occur at the end of application.

2.4 Worker Initiated Sub-Task Size (WIS) Scheduling Strategy

For centralized workers state information gathering and making decision about the scheduling of task has seen to be saturated when HDC system consists of hundreds of WSs/PCs. Because for each worker, the master has to collect the worker's state information and decide the size of the task for next assignment. Therefore, a novel approach of semi-decentralized resources collections is proposed. In that, each worker has autonomy to decide next sub-task size according to its current performance. The WIS strategy is designed by considering the following points in mind:

i) The adaptive nature of task sizing scheduling strategy is essential to absorb the heterogeneity and performance variation characteristics of the machine/worker.

ii) The worker should compute its sub-task size according to its current performance without knowledge of other workers' performance. This approach cuts the worker waiting time, which occurs due to collection of other workers' state information [18]. Effectively this reduces the network usage that may occur, as every worker wants to float its performance information on the network in a specified interval.

For making decision about the task partitioning and scheduling the researchers use benchmark programs to monitor the machines' performances in a HDC system. Since due to the non-homogeneous nature of application (image), the machine's performance estimated from worker's runtime responses may not give the accurate worker's performance [19] (for example, see Fig. 5, the processing diversity of one scan-line of image A). The authors [20, 21] clearly indicate that, the number of CPU job queues indices length approach for load balancing gives better result. In WIS strategy, the CPU jobs queue length information is used for estimating the worker's performance. The estimated worker's performance is used for worker is sub-task size computation.

In this strategy, the manager has prior knowledge of each worker's performance and its job queue lengths' information at unloaded condition (no other user was logged in the network). Manager utilizes these information to compute the initial sub-task size for each worker in HDC system using equation 10. The slowest performance worker's sub-task size is supposed to be unit of task (one horizontal scan-line of the image).

$$t_{initial(i)} = \frac{W_{pui}}{W_{puslow}} \qquad (10)$$

Each worker of the HDC system records its initial sub-task size ($t_{initial(i)}$). The worker also has knowledge of its performance and job queue lengths information at unloaded condition. Whenever the worker finishes the task, it estimates its current performance with reference to its noted performance and queues length at unloaded condition using equation 11:

$$W_{pi} = \frac{W_{pui} * W_{qi}}{W_{qui}} \qquad (11)$$

Further, worker estimates its next sub-task size using equation 12:

$$t_i = \frac{W_{pui} * W_{pi}}{t_{initial(i)}} \qquad (12)$$

Note that, in MIS and WIS strategies, each worker's sub task size is modified only when worker's performance ratio or ($\frac{Current \quad W_{pi}}{Previous \quad W_{pi}}$) or vise versa becomes 0.25, because the rapidly fluctuating task sizing never gives better results for HDC system [22]. The 70% of the total task is scheduled according to the above described procedure and for remaining the manager reduces the sub-task size (t_i) of each worker by 30% on each worker's request. This reduces the work load-imbalance and manager waiting time that occurs usually at the end of the application. To achieve a better load balancing and avoid master machine waiting time at the end of the application, the components 4 and 5 of MIS strategy are included in this strategy.

2.5 Workers Sub-Task Sizing

Before implementing the MIS and WIS strategies on actual HDC system, the sub-task sizing behavior of workers are simulated. For this purpose, separate C codes are implemented for these strategies to feed the pre-measured machine's performance at compile time. Let us, consider the case when the HDC system is composed of number of machines (NM) = 20. The two higher performance machines' sub-task sizing (HPM 20-1 and HPM 20-2) behavior are listed in Fig. 3a and Fig. 3b.

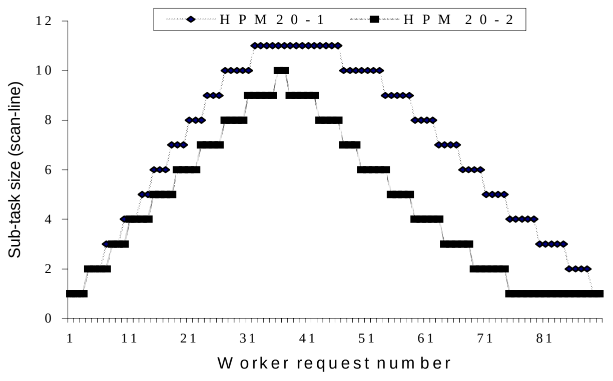


Fig. 3a: Sub-task sizing of two workers in MIS strategy

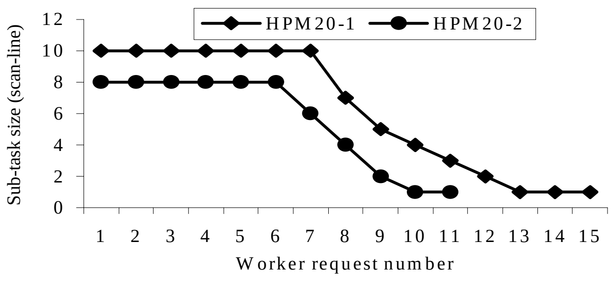


Fig. 3b: Sub-task sizing of two workers in WIS strategy

It is observed from Fig. 3a and Fig. 3b, for all workers' requests the sub-task size of first higher performance worker (HPM 20-1) is always larger than the sub-task size of second higher performance worker (HPM 20-2). In MIS strategy, the worker's task size increases and decreases gradually. However in case of WIS strategy, each worker's sub-task size starts from its peak value and saturates after certain requests, the worker's sub-task size decreases on each worker's request at the end of the application.

3.0 IMPLEMENTATION

The performance of static, RTS, MIS and, WIS strategies are mainly evaluated in terms of speedup and workers idle time cost [23]. Here these terms are defined briefly.

Speedup: The speedup is used to quantify the performance gain from a parallel computation (T_{pdp}) of an application over its independent computation (T_{pf}) on a single fastest machine in the network of WSs/PCs. It is defined as follows:

$$\text{Speedup of HDC system (SP)} = \frac{T_{pf}}{T_{pdp}} \qquad (13)$$

Workers idle time cost: The activities of worker's process in a HDC image processing system is mainly composed of three factors that are:

i) Worker setup time to load pattern data file and initialize all programming parameters
ii) Worker raytracing computation time
iii) Worker time taken to report result (data) and getting new task from master.

The value of the third factor is directly affected when the number of requests made by the worker increases. This overhead in time is computed as follows:

$$O_i = \text{(Worker's previous task completion time)} - \text{(Worker's new task starting time)} \qquad (14)$$

where O_i is the overhead time, which includes all communications data access delays to start the processing of new task on idle worker.

The workers idle time cost can be expressed as:

$$\text{Workers idle time cost} = \sum_{i=1}^{W_t} O_i \qquad (15)$$

The HDC system used in our investigation is composed of seven Sun workstations (WSs) loaded with SunOS/Solaris, and thirteen PCs (Intel based machines) loaded with Linux/FreeBSD operating system; all of these machines are connected via Ethernet. The network communication is handled by *Open Consortium Remote Procedure Call* (ONC-RPC) *library and XDR filters,* UNIX heavy weight processing technique is used to control many workers at the manager's end. The investigation is carried out on raytracing images because it can be parallelized without complex inter-processors communication.

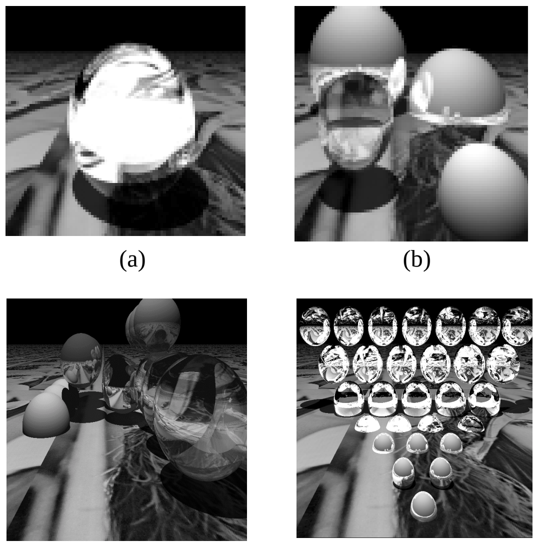


Fig. 4: Raytracing images: a) Scene A, b) Scene B, c) Scene C, and d) Scene D

Table 2: The single fastest machine in the network image scenes processing time

Image Scene A	Image Scene B	Image Scene A	Image Scene D
63 sec	122 sec	152 sec	244 sec

To analyze the performances of the proposed MIS and WIS strategies, we used four distributed raytracing images that were known as scene A, B, C, and D and each image is composed of 840 x 640 pixels. Fig. 4 gives the impressions of the images. The five HDC configurations are set, i.e., the HDC system is composed of number of machine (NM) = 20, 16, 12, 8 and 4. These HDC configurations were configured in such a way that (starting from the maximum workers, i.e., NM=20) each next HDC configuration is arranged by omitting the four slowest workers from the current configuration.

4.0 RESULTS AND DISCUSSION

The performances of MIS and WIS strategies are compared in terms of speedup, number of request made by the master and workers idle time cost and finally their performances are compared with static and RTS strategies. The investigated HDC system configurations have high heterogeneity in machine's performance; some machines have a speed of 400Mhz and other have 40Mhz. The processing time taken to process the image scenes A, B, C and D by the single fastest machine in the network are shown in Table 2. From Table 2, it is clear that image scene A is lightly densed and takes less processing time, while scene D is the heavily densed among all scenes.

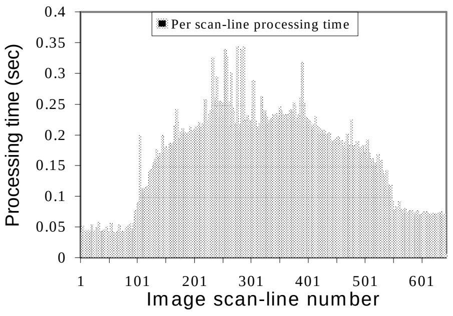


Fig. 5: Image scene A scan-line processing time histogram

Static strategy: The obtained speedups listed in Table 4, are the worst as compared to RTS, MIS, and WIS strategies. This is because of two main reasons:

i) Due to the non-homogeneous nature of the applications (images). Therefore, some workers complete the assigned tasks quickly, where other may take longer time due to the heavy processing requirements of the assigned task. For example, the non-homogeneity of an image A in term of per scan-line processing time can be realized from Fig. 5. If dense processing required task assigned to the slow performance worker, it may create a long master waiting time at the end of the application. The mechanism of master waiting time can easily be understood from Fig. 6

ii) The machine's performance has a dynamic variation characteristic that can not be accurately estimated at compile time [6].

Due to the above stated reasons, in most of the cases it is observed that the slowest machine causes a long master waiting time at the end of the application. The measured master waiting time for images A and D are shown in Fig. 7. From Fig. 7, it can be realized that the dense application has more master's waiting time as compared to less dense application.

The 27%, 44%, and 49% at average approximate degradations are found in measured speedups in all five HDC configurations with respect to RTS, MIS, and WIS strategies, respectively.

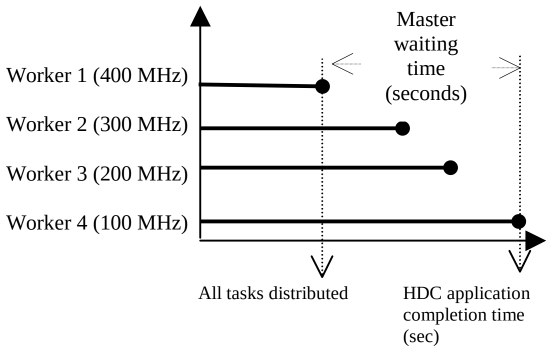


Fig. 6: Master machine waiting time mechanism

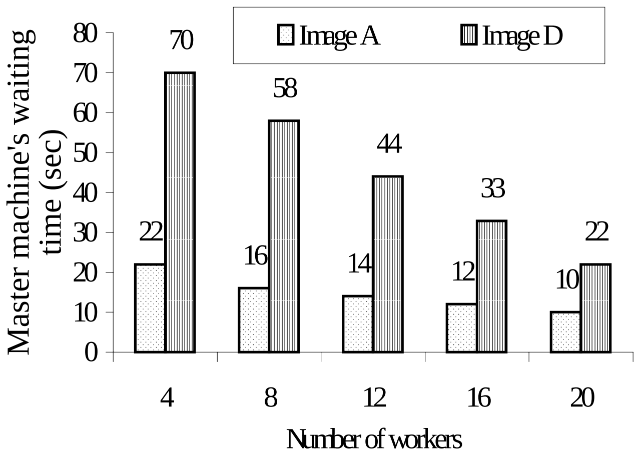


Fig. 7: In static task scheduling strategy, the obtained master machine waiting time in images A and D

Runtime Task Scheduling (RTS) strategy: The RTS strategy has a potential to absorb the machine's performance variation characteristics and non-homogeneous nature of the application at runtime tasks assignment. The number of tasks processed by the worker is proportional to its runtime performance. However, RTS strategy has two serious drawbacks that are explained in section 1. The performance of RTS strategy is evaluated using the fixed unit of task (one horizontal scan-line of the image). Due to the small task size, all workers are utilized at the end of the application and very less master waiting time (less then one second) is generated in all HDC configurations. The measured speedups of this strategy are listed in Table 4. The speedup results show that the RTS has speedup improvements of approximate 27% at average with respect to static strategy. However, it has at average 29% and 37% degradation with respect to MIS and WIS strategies, respectively. Due to the large number of master's requests occurred in RTS strategy (see Table 3), more workers idle time cost is generated (see Fig. 8). From Fig. 8, the average measured workers idle time cost increases as the number of workers increases in HDC configurations, which is due to the following reasons:

i) As the number of workers increases the waiting child processes increases at the master machine, which increases the auxiliary workload at master machine, therefore, it may decrease its task paralellization capability, which effectively increases the worker's waiting time

ii) As the number of workers increases in master and workers HDC system model, the low bandwidth network usage will increase, because every worker has a responsibility to report the results to the master. Therefore, it creates auxiliary load on the network, which may be the cause of long workers idle time cost.

Table 3: Comparison of static, RTS, MIS, and WIS strategies in term of number of requests made by master machine to distribute the whole task

HDC System Configuration (NM)	Strategies			
	Static	RTS	MIS	WIS
4	4	640	96	80
8	8	640	129	88
12	12	640	143	130
16	16	640	206	170
20	20	640	253	225

Master initiated sub-task size (MIS) strategy: In this strategy, the master estimates the workers' performances from their runtime responses and make decisions about the size of sub-task size for the workers.

Using equation (13), the computed speedups of all four images and five HDC configurations are listed in Table 4. For image scenes A, B, and C the MIS strategy has an average speedup of 16% to 19% as compared to RTS, but for image scene D, an average 22% performance speedup is found as compared to RTS. It is observed that, the densed application (like, image scene D) has high speedup as compared to less dense applications (i.e., scenes A, B, and C). The reason is that the workers idle time cost effect is reduced due to the large processing time.

From Table 3, the number of requests made by the master for tasks assigning and collecting data from the workers for all five HDC configurations also reduces in MIS strategy as compared to RTS. In MIS strategy, the number of requests made by the workers in HDC configurations NM = 4, 8, 12, 16 and 20 increases as shown in Table 2. The reason is that, i) the sub-task size of the worker is proportional to its normalized performance (see equation 5). In addition, the worker's normalized performance decreases as the number of workers increases in the investigated PDP configurations. Therefore, the decrease in each worker's sub-task size effectively increases the number of requests made by the master machine to distribute the whole task.

The measured average workers idle time cost for RTS, MIS, and WIS strategies are shown in Fig. 8. From Fig. 8, it is noted that as the number of workers in HDC system increases, the worker idle time cost increases. The reasons for these increments are already explained in section 4 (RTS strategy).

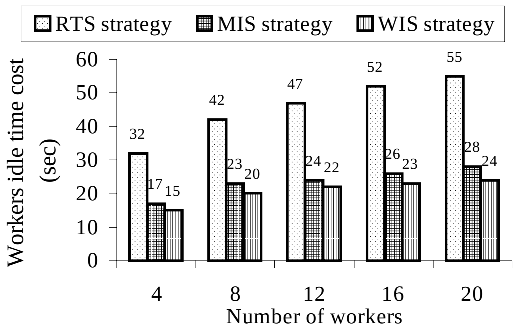


Fig. 8: Measured average workers idle time cost (sec) in five HDC system configurations obtained in RTS, MIS, and WIS strategies

Worker initiated sub-task size (WIS) strategy: The strategy is based on the semi-decentralized approach of resources management. For 70% of the total task, each worker in HDC system itself suggests its next sub-task size to the master machine while returning the results. However, for remaining 30%, the master reduces the sub-task size of each worker on each worker's request. The strategy is carefully designed so that each worker should not wait to get other workers state information. According to the sub-task assignment mechanism of this strategy (see Fig. 3b), at the starting of the application each worker's sub-task size is greater as compared to the sub-task size assignment in MIS strategy. Due to this reason, the number of request made by the master is slightly less as compared to MIS strategy (see Table .3).

The performance of WIS strategy highly depends on the performance heterogeneity of the workers in HDC system. If performance heterogeneity of HDC system is lesser, in that case each worker's sub-task size is shorter that causes the increment in number of requests made by the client machine to distribute the whole tasks that may degrade its performance. This strategy is tested on maximum of 20 WSs/PCs. It has average slight improvement in speedup of 3% to 6% over MIS strategy. From Table 4, it is clear that as the number of workers increases in HDC system the strategy gives better scalability as compared to MIS strategy.

Table 4: Measured speedup for five HDC configurations in static, RTS, MIS, and WIS strategies

HDC system configuration (NM)	Static strategy Speedup for image scenes:				RTS strategy Speedup for image scenes:				MIS strategy Speedup for image scenes:				WIS strategy Speedup for image scenes:			
	"A"	"B"	"C"	"D"	"A"	"B"	"C"	"D"	"A"	"B"	"C"	"D"	"A"	"B"	"C"	"D"
4	1.2	1.7	1.8	1.8	1.8	2.2	2.5	2.5	2.6	2.9	2.9	3.2	2.8	3.2	3.3	3.8
8	1.4	2.1	2.2	2.6	2.6	3.0	3.2	3.4	3.5	3.9	4.2	5.2	3.9	4.0	4.4	5.5
12	1.5	2.3	2.5	3.2	3.2	3.5	4.0	4.0	4.2	4.7	5.0	5.9	4.8	5.0	5.2	6.4
16	1.6	2.4	2.9	3.6	3.6	3.7	4.3	4.5	4.5	5.5	5.2	6.2	5.2	5.5	5.6	6.7
20	1.8	2.8	3.4	4.0	4.0	4.2	4.8	5.0	5.3	5.8	6.1	6.7	5.7	6.1	6.3	7.3

5.0 CONCLUSIONS

In this paper, the adaptive centralized (MIS strategy) and semi-decentralized (WIS strategy) resources management and worker's sub-task size computation decision making for master workers model of HDC system are presented. The results show that the static and RTS are inefficient for heterogeneous HDC image computing system. The adaptive MIS and WIS strategies remedy the defects of static and RTS strategies. The MIS strategy has 44% and 19% approximate speedup improvement over static and RTS strategies, respectively. The workers idle time cost also reduces to 43% approximately with respect to RTS strategy.

The WIS strategy performance is evaluated on 20 (maximum) WSs/PCs. It shows slightly better performance and scalability over MIS strategy.

REFERENCES

[1] I. Grimstead and S. Hurley, "Accelerated Raytracing Using an NCUBE2 Multicomputer", *Concurrency Practice and Experience*, 2(6): 1995, pp. 571-586.

[2] S. Molnar, M. Cox, D. Ellsworth and H. Fuchs, "A Sorting Classification of Parallel Rendering", *IEEE Computer Graphics and Applications*, 1994, 14(4), pp. 23-32.

[3] C. Giertsen, J. Pettersen, "Parallel Volume Rendering on a Network of Workstations", *IEEE Computer Graphics and Applications*, Nov. 1993, pp. 17-23.

[4] M. Hamid, C. Lee, "Dynamic Load-Balancing of Image Processing Applications on Cluster of Workstations", *Parallel Computing*, 22 (1997), pp. 1477-1492.

[5] K. Qureshi, H. Terasaki and M. Hatanaka, "A Network Parallel Distributed Processing System Using PCs and WSs", *Journal of Geotechnology*, Research Association of Japan, Muroran Institute of Tech., 38 (1996), pp. 1-8.

[6] Y. Zhang, H. Kameda, S. L. Hung, "Comparison of Dynamic and Static Load-Balancing Strategies in Heterogeneous Distributed Systems", *IEE Proceeding Comput. Digit. Tech*., 144 (2), 1997, pp. 100-107.

[7] K. Qureshi, M. Hatanaka, "PC-UNIX Parallel Distributed Processing on Heterogeneous Hardware", Research *Conference at Muroran Institute of Technology*, Hokkaido Japan (1996), pp. 1-2.

[8] J. Bloomer, *Power Programming with RPC*, O'Reilly and Associates (1991), pp. 409-451.

[9] K. Qureshi, M. Hatanaka, "Practical Aspect and Experiences of Runtime Task Scheduling for Parallel Raytracing on a Heterogeneous Distributed Computing System", *Third InterMedia Symposium*, Sapporo, Japan (1998), pp. 64-70.

[10] C. Lee, M. Hamid, "Parallel Image Processing Applications on a Network of Workstations", *Parallel Computing*, 21(1995), pp. 137-160.

[11] E. Reinhard, W. Jansen, "Rendering Large Scenes Using Parallel Raytracing", *Parallel Computing*, 23(1997), pp. 873-885.

[12] S. Zhou, "A Trace-Driven Simulation Study of Dynamic Load Balancing", *IEEE Tran. on Software Engg*. 14(9), 1988, pp. 1327-1341.

[13] A. Sarje, G. Sagar, "Heuristic Model for Task Allocation in Distributed Computing Systems", *IEE Proceedings –E*, 5 (1991), pp. 313-318.

[14] S. P. Dandamudi, "Sensitivity Evaluation of Dynamic Load Sharing in Distributed Systems", *IEEE Concurrency*, July-Sept. (1998), pp. 63-72.

[15] B. Freisleben, D. Hartmann, T. Kielmann, "Parallel Raytracing: A Case Study on Partitioning and Scheduling on Workstation Clusters", *Proceeding of the Thirtieth Annual Hawaii International Conference on System Sciences*, 1997, pp. 596-605.

[16] B. Freisleben, T. Kielmann, "Parallel Incremental Raytracing of Animation on a Network of Workstations", PDPTA'98 International Conference, Las Vegas, U.S.A, 1998, pp. 1305-1312.

[17] K. Qureshi and M. Hatanaka, "An Investigation on Runtime Task Scheduling for Parallel Raytracing on a Heterogeneous Distributed Computing System", *PDPTA'98 International Conference*, Las Vegas, U.S.A, 1998, pp. 1066-1074.

[18] J. G. Vaughan, "Static Performance of a Divide and Conquer Information Distributed Protocol Supporting a Load-Balancing Scheme", *IEE Proceedings-E*, Vol. 139, No. 5, September 1992, pp. 431-437.

[19] T. Lee, "Exploitation of Image Parallelism for Raytracing 3D Scenes on 2D Mesh Multi-Computers", *Parallel Computing*, 23(1997), pp. 1993-2015.

[20] B. A. Shirazi, A. Hurson, "Scheduling and Load Balancing in Parallel and Distributed Systems", *IEEE Computer Society press*, ISBN 0-8186-6587-4, 1995, pp. 487-496.

[21] T. Kubz, "The Influence of Different Workload Descriptions on a Heuristic Load Balancing Scheme", *IEEE Trans. on Software Eng.*, Vol. 17, No. 7, July 1991, pp. 725-730.

[22] J. Watts, Stephen Taylor, "A Practical Approach to Dynamic Load Balancing", *IEEE Trans. on Parallel and Distributed Systems*, 9 (3) 1998, 235-248.

[23] X. Du, X. Zhang, "Coordinate Parallel Processes on Networks of Workstations", *Journal of Parallel Distributed Computing*, Vol. 46, No. 2, 1997, pp. 125-135.

BIOGRAPHY

Kalim Qureshi is a Ph.D candidate in the Dept. of Computer Science and Systems Engg., Muroran Institute of Technology, Hokkaido, Japan. His research interests include network Parallel Distributed Computing, Thread Programming, Concurrent Algorithms Designing, Task Scheduling, and Performance Measurement. He is a student member of IEEE Computer Society since 1993.

Masahiko Hatanaka (Dr. Eng.) is an Associate Professor in the Dept. of Computer Science and Systems Engg., Muroran Institute of Technology, Hokkaido, Japan. His research interests include Medical Imaging, Remote Sensing, Image processing, Parallel Distributed Computing, and Computer Networking. He is a member of IEEE Computer Society.