Stochastic Approach to Scheduling Multiple Divisible Tasks on a Heterogeneous Distributed Computing System

Ankur Kamthe
Department of Mathematics and Statistics
Auburn University
Auburn, AL 36830
Email: kamthan@auburn.edu

Soo-Young Lee
Department of Electrical and Computer Engineering
Auburn University
Auburn, AL 36830
Email: leesooy@auburn.edu

***Abstract*— Heterogeneity has been considered in scheduling, but without considering the temporal variation of completion times of the sub-tasks for a divisible, independent task. In this paper, the problem of scheduling multiple, divisible independent tasks on a heterogeneous distributed computing system is addressed. In our paper, a new "stochastic" approach to scheduling groups of multiple divisible as well as whole independent tasks is proposed. It explicitly considers the standard deviations (temporal heterogeneity) in addition to the mean execution time in deriving a schedule. By utilizing the second order moments, the heuristics employed in the new approach attempt to "follow" more closely "on average" what would actually happen on a temporally heterogeneous system (instead of approximating the random weights by their means only as in other approaches). Through an extensive computer simulation, it has been shown that the proposed approach can improve schedules significantly over those by a scheme which uses the average weights only.**

I. Introduction

Divisible tasks are generally divided into smaller load fractions so that they can be assigned to multiple processors for faster execution. Scheduling algorithms exist for directed acyclic task graphs (DAG), where there are precedence relationships between the group of tasks, and independent tasks, where there are no precedence relationships between tasks. Generally, scheduling schemes adopt some kinds of heuristics for mapping the tasks to processors to optimize (reduce) the overall parallel execution time. In these heuristics, the computation and communication times are considered to be *deterministic*, i.e., do not vary with time.

These days, resources in a distributed system such as a cluster and a computational grid are shared by multiple users. Therefore, the power of a resource available for a task may vary with time, i.e., *temporal heterogeneity* [1]. Also, resources may not be all identical or a different resource may exhibit a different behavior of availability, i.e., *spatial heterogeneity*. In addition, an application itself may be inherently *random*, e.g., simulated annealing, genetic algorithm, etc. Hence, the computation and communication times are no longer deterministic on such a heterogeneous distributed computing system even when all processors in the system are identical.

For tackling the problem posed by heterogeneity in task scheduling, many approaches have been proposed by a number of researchers. A brief review of their work is as follows: In DAG scheduling, spatial heterogeneity has been considered in scheduling [2] [3] [4] [5] [6] [7]. In [8], task computation time is treated as a random variable and is statistically estimated from past observations. In [9], the standard deviations of computation and communication times were used to model the behavior of sub-tasks in a DAG instead of approximating using means only as in other approaches. It was showed that a significant improvement can be achieved by considering the standard deviation when heterogeneity is present.

In [10], a stochastic approach was taken to scheduling a set of *independent* tasks using a Genetic Algorithm wherein computation times are not deterministic. In [11], an optimal steady-state scheduling strategy was proposed for a suite of identical, independent problems where each problem consists of a set of tasks. It attempted to exploit the mixed data and task parallelism on heterogeneous clusters and grids. In [12], a method for scheduling independent tasks consisting of non-linear DAGs was proposed. However, heterogeneity was not considered. In [13], the problem of allocating and scheduling a collection of independent, equal-sized tasks on heterogeneous computing platforms was addressed, taking memoy constraints into account. In [14], the problem of scheduling multiple divisible load applications on a grid platform was addressed. The authors do not explicitly consider the heterogeneity in completion times of the divisible task executing on a cluster when scheduling successive tasks.

Many massively parallel applications consist of multiple divisible load applications, here certain applications (tasks) are further divisible into sub-tasks (requiring use of multiple processors), whereas the remaining may be treated as whole tasks (requiring single processor for execution). When a distributed system such as a cluster or a computational grid is used to execute such an application, heterogeneity would lead to delays in the execution of the divisible subtasks, affecting (increasing) the overall parallel execution times. As discussed above, scheduling schemes are targeted towards DAGs or

groups of independent tasks but not a hybrid mix of divisible and indivisible tasks.

In our paper, a new approach to scheduling a group of multiple divisible as well as whole independent tasks is proposed. It explicitly considers the standard deviations (temporal heterogeneity) in addition to the mean execution time in deriving a schedule. By utilizing the second order moments of weights also, the heuristics employed in the new approach attempt to "follow" more closely "on average" what would actually happen on a temporally heterogeneous system (instead of approximating the random weights by their means only as in other approaches). Through an extensive computer simulation, it has been shown that the proposed approach can improve schedules significantly over those by a scheme which uses the average weights only.

The rest of this paper is organized as follows. In Section II, the terms and notations used in the paper are described. In Section III, the different scheduling heuristics used in the paper are described. In Section IV, the divisible load scheduling model is described. In Section V, the new approach to scheduling groups of divisible tasks is described. In Section VI, the simulation procedure is discussed. In Section VII, the simulation results are discussed in detail and at the end, the summary in Section VIII.

II. Terms and Notations

- P_j: processor j.
- n_i: task i.
- t_i: computation time of n_i, which has the mean m_{ij} and standard deviation σ_{ij} on P_j.
- t_{cij}: completion time of n_i, which has the mean m_{cij} and standard deviation σ_{cij} on P_j.
- Availability Time of Processor, ATP_j: the time at which P_j becomes available.
- Earliest Start Time (EST): the earliest time at which a task can be scheduled for execution on an available processor.
- Schedule length, T_{SL}: the make-span of a schedule, or the parallel execution time predicted by a scheduling algorithm.
- Average parallel execution time, t_p: the actual average time required to execute a group of independent tasks according to a schedule. The corresponding standard deviation is denoted by σ_{T_p}.
- E[ ]: the expectation operator used to compute the mean of a random variable.

III. Scheduling Heuristics

Scheduling a group of independent tasks on multiple processors is an NP-complete problem. Therefore, scheduling algorithms use heuristics for obtaining near optimal schedules. If the tasks to be scheduled are independent and compute intensive, then algorithms such as maxmin and minmin algorithms can be used. They have time complexities of $O(V^2P)$, where P is the number of processors and V is the number of tasks to schedule.

Min-Min Algorithm

The algorithm is described below:

1) A task list is generated that includes all unmapped tasks.
2) Find the completion time (CT) of each unmapped task on each machine (ignoring other unmapped tasks).
3) Find the machine that gives the minimum CT for each task.
4) Among all the task/machine pairs found in 3, find the pair that gives the minimum CT.
5) Remove the above task from the task list and map it to the chosen machine.
6) Update the available time of the machine on which the task is mapped.
7) Repeat steps 2-6 until all the tasks have been mapped.

Max-Min Algorithm

The algorithm is described below:

1) A task list is generated that includes all unmapped tasks.
2) Find the CT of each unmapped task on each machine (ignoring other unmapped tasks).
3) Find the machine that gives the minimum CT for each task.
4) Among all the task/machine pairs found in 3, find the pair that gives the maximum CT.
5) Remove the above task from the task list and map it to the chosen machine.
6) Update the available time of the machine on which the task is mapped.
7) Repeat steps 2-6 until all the tasks have been mapped.

IV. Divisible Task Scheduling

One essential step in both max-min and min-min schemes is to compute the CT of each task. When a group of independent tasks is executed on a temporally (and spatially) heterogeneous distributed computing system, the CT of each task may vary with time in general. That is, CT t_{cij} of n_i can be considered to be a random variable. Suppose some of the tasks are divisible i.e. they require multiple processors for execution, there is an uncertainty in the CT of the task due to the heterogeneity in resources. Consider the following cases:

- Case 1: When an independent task requiring multiple processors is executed on a temporally heterogeneous but spatially homogeneous system, t_{cij} would change with respect to time in a random manner.
- Case 2: When an independent task requiring multiple processors is executed on a spatially heterogeneous but temporally homogeneous system, the processor-dependent CT of a task can be modelled by the random variables t_{cij}.
- Case 3: When an independent task requiring multiple processors is executed on a temporally and spatially heterogeneous system, the random variables t_{cij} reflects both temporal and spatial variation of completation time of n_i.

In reality, heterogeneity cannot be accurately characterized due to the random nature of the problem. It is comprised

TABLE I

THE MEAN AND STANDARD DEVIATIONS OF TASK EXECUTION TIMES USED FOR COMPARISON OF SCHEDULE AND AVERAGE PARALLEL EXECUTION TIME FROM THE STATIC AND STOCHASTIC APPROACH. (NOTE: FOR n_0, TIMES GIVEN ARE FOR EACH SPLIT PORTION.)

Task	m_{ci1}, σ_{ci1}	m_{ci2}, σ_{ci2}	m_{ci3}, σ_{ci3}
n_0	$(10, 3)$	$(10, 1)$	$(10, 1)$
n_1	$(10, 2)$	$(10, 4)$	$(10, 4)$

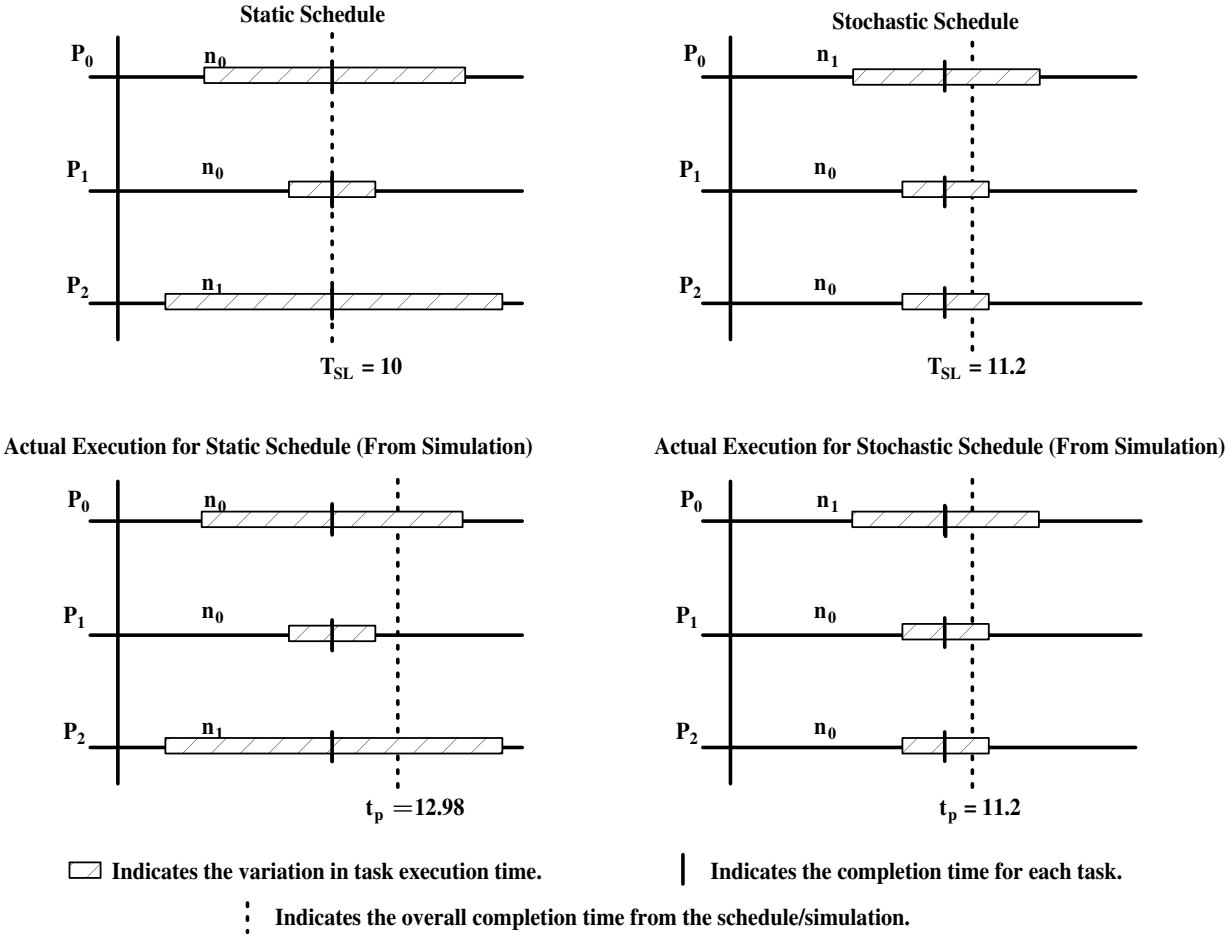

Fig. 1. Comparison of schedule and average parallel execution time from the static and stochastic schedules.

TABLE II

THE MEAN AND STANDARD DEVIATIONS OF TASK EXECUTION TIMES USED FOR COMPARISON OF SCHEDULE AND AVERAGE PARALLEL EXECUTION TIME FROM THE STATIC AND STOCHASTIC APPROACH. (NOTE: FOR n_3, TIMES GIVEN ARE FOR EACH SPLIT PORTION.)

Task	m_{ci1}, σ_{ci1}	m_{ci2}, σ_{ci2}	m_{ci3}, σ_{ci3}
n_0	$(15, 7)$	$(15, 4)$	$(15, 6)$
n_1	$(15, 4)$	$(15, 3)$	$(15, 4)$
n_2	$(15, 3)$	$(15, 2)$	$(15, 1)$
n_3	$(10, 3)$	$(10, 1)$	$(10, 3)$

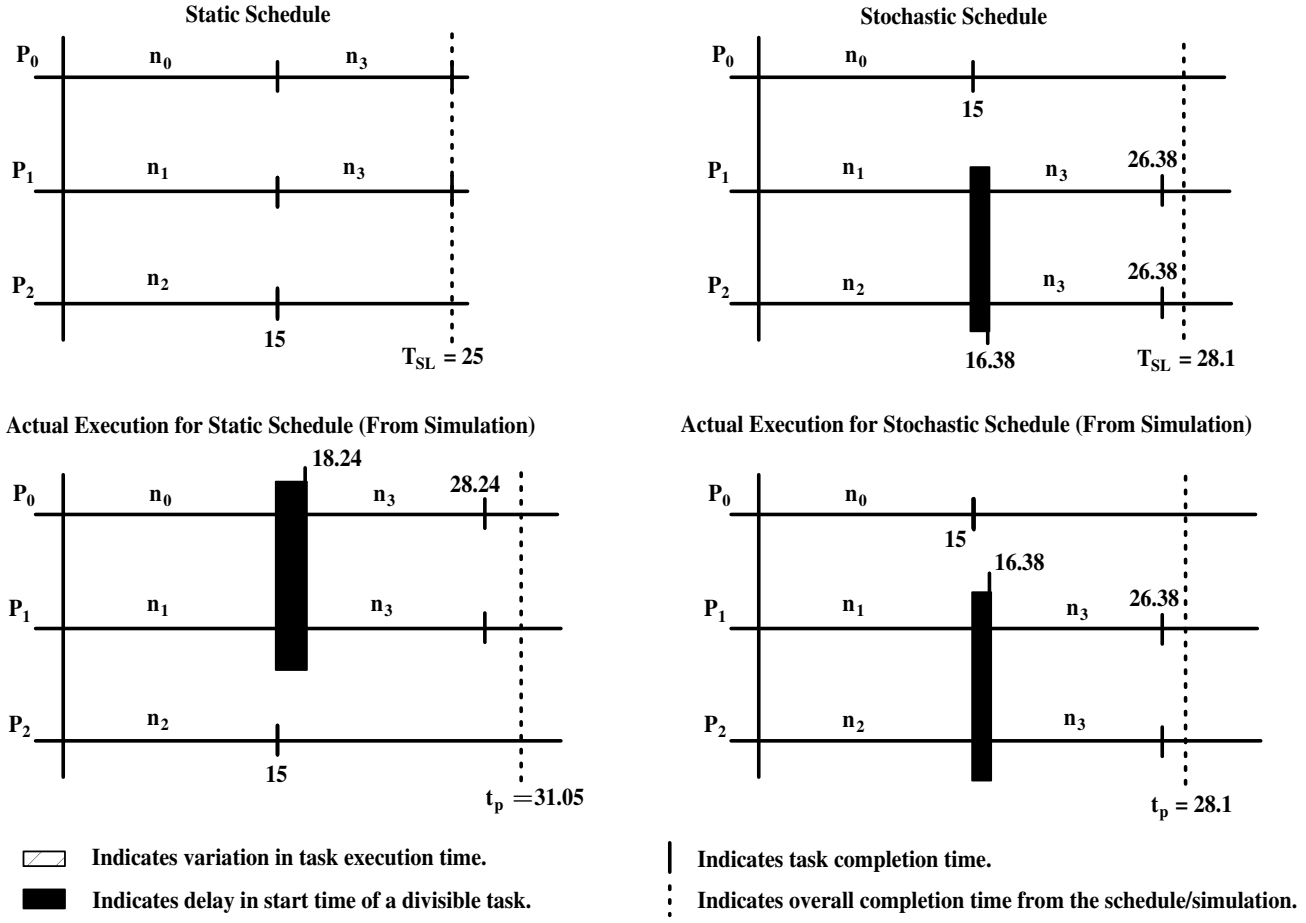

Fig. 2. Comparison of schedule and average parallel execution time from the static and stochastic schedules.

of a combination of temporal and spatial heterogeneity, and can be characterized as either Cases 1-3 mentioned above or a mix of these three cases where task heterogeneity is combined with machine or environmental heterogeneity. In the remainder of this paper, the first case is assumed for the convenience of discussion though all three cases can be handled by the stochastic model. In our model for divisible task scheduling, the random variable t_{cij} is replaced by a pair of its first two moments, i.e., the mean m_{cij} and standard deviation σ_{cij}. Similarly, the random variable t_{cik} is replaced by its mean m_{cik} and standard deviation σ_{cik}. Consider the following cases:

- **Case 1: Difference in computing completion time for a divisible task.**
  Suppose two independent tasks are to be scheduled, one of the tasks (task 0) is divisible and requires two processors for execution (refer Table I for task execution times and Figure 1). Both of these tasks exhibit temporal heterogeneity due to machine and/or environmental heterogeneity, or the random nature of the task itself (task heterogeneity e.g. simulated annealing, markov chains). In the static case (refer Figure 1), the CT for all tasks is computed to be $t_{cij} = m_{cij}$ (leading to n_0 having same CT (= 10) for all processor pairs). Therefore, task 1 is scheduled on P_0 and P_1, and task 2 is scheduled on P_2, leading to schedule length $T_{SL} = 10$. Execution of this schedule leads to $t_p = 12.98$. In the stochastic case, the CT of task 1 is computed as $E[max(t_{c1j}, t_{c1k})]$ and hence is different for each $(j \neq k)$ (least CT (= 10.9) if scheduled on P_1 and P_2). CT of task 2 (t_{c2j}) remains the same for all j. Therefore, task 1 is scheduled on P_1 and P_2, and task 2 is scheduled on P_0. The schedule length is computed as $E[max(t_{c10}, t_{c11}, t_{c12})]$ in the stochastic case. This leads to $T_{SL} = 11.2$. Execution of this schedule leads to $t_p = 11.2$, which is the same as T_{SL} predicted earlier. This is the one of main difference between ignoring (static case) or considering (stochastic case) the heterogeneity in execution times of divisible, independent tasks which motivates this study.
- **Case 2: Difference in computing completion time for a divisible task scheduled after independent tasks.**
  Suppose four independent tasks are to be scheduled, one of the tasks (task 3) is divisible and requires two processors for execution (refer Table II for task execution times and Figure 2). All of these tasks exhibit temporal heterogeneity due to either machine and/or environmental

heterogeneity or the random nature of the task itself (task heterogeneity). In the static case, tasks n_0, n_1 and n_2 are scheduled on P_0, P_1 and P_2 respectively. Task 3 which is a divisible task is scheduled on P_0 and P_1 (for the static case, CT for a divisible task is same for all processor pairs (= 25)) This leads to $T_{SL} = 25$. During execution of this schedule, the start of n_3 is delayed due to the heterogeneity in execution times of n_0 and n_1. This leads to $t_p = 31.05$. In the stochastic case, just as in the static case, tasks n_0, n_1 and n_2 are scheduled on P_0, P_1 and P_2 respectively. The CT of n_3 is computed as $E[max(t_{c00}, t_{c11}] + E[max(t_{c30}, t_{c31}]$ (or $E[max(t_{c11}, t_{c22}] + E[max(t_{c31}, t_{c32}]$ or $E[max(t_{c00}, t_{c22}] + E[max(t_{c30}, t_{c32}]$). It is least ($\approx 28$) when n_3 is scheduled on P_1 and P_2. Thus, the stochastic case considers the heterogeneity in the start time for a divisible task and uses it while computing the CT for that task. This leads to $T_{SL} = 28.1$ (computed as $E[max(t_{c00}, t_{c31}, t_{c32})]$). Execution of this schedule leads to $t_p = 28.1$, which is the same as T_{SL} predicted earlier. This is the other main difference between ignoring (static case) or considering (stochastic case) the heterogeneity in execution times of divisible, independent tasks which motivates this study.

Parallel execution time of a group of independent tasks is defined to be the total time required to execute all the tasks. Since the parallel execution time (T_p) is random, one may adopt the *average parallel execution time* $t_p = E[T_p]$ to quantify the quality of a schedule. As mentioned in Section I, this measure is useful especially when a task needs to be repeatedly executed.

V. Stochastic Approach to Divisible Task Scheduling

In this section, a new approach to scheduling divisible tasks, which follows the same scheduling procedures as those of the min-min and max-min algorithms reviewed in Section IV, is described. Its implementations for min-min and max-min are referred to as *stochastic* min-min and *stochastic* max-min, respectively. It is assumed that $\{m_{ij}, \sigma_{ij}\}$ is known at the time of scheduling, i.e., static scheduling.

A. Completion Time for a divisible independent task

In the stochastic min-min (or max-min), the *average* CT (just CT hereafter) of each node is used in computing the priority of the node for scheduling. One of the differences between the stochastic and static versions of min-min and max-min is how the CT of a divisible independent task is computed when it is executed concurrently on multiple processors. Referring to Figure 1, the CT of n_1 is $E[max((t_{c11}), (t_{c12}))]$. When $(m_{c11}) \gg (m_{c12})$ (or $(m_{c11}) \ll (m_{c12})$), then $E[max((t_{c11}), (t_{c12}))] \approx (m_{c11})$ (or $E[max((t_{c11}), (t_{c12}))] \approx (m_{c12})$). However, when m_{c11} is comparable to m_{c12} (this is mostly the case due to load balancing), and σ_{c11} or $\sigma_{c12} \neq 0$, $E[max((t_{c11}), (t_{c12}))]$ is substantially greater than $max((m_{c11}), (m_{c12}))$. In general, the CT for a divisible task i requiring p processors for execution can be expressed as $E[max(t_{ci1}, ..., t_{cip})]$. Referring to Figure 2, the CT of n_3 is $E[max(t_{c11}, t_{c22}] + E[max(t_{c31}, t_{c32}]$. However, when m_{c11} is comparable to m_{c22}, m_{c31} is comparable to m_{c32} and $\sigma_{c11}, \sigma_{c22}, \sigma_{c31}$ or $\sigma_{c32} \neq 0$, $E[max(t_{c11}, t_{c22}] + E[max(t_{c31}, t_{c32}]$ is substantially greater than $max(t_{c11}, t_{c22}) + max(t_{c31}, t_{c32})$.

These case are referred to as *competing situations*, i.e., tasks compete to determine the completion time of n_1 (or n_3). The competing situation implies that the divisible portions of n_1 (or n_3) are scheduled on multiple different processors. Note that a node may not be scheduled on the processors involved in executing the divisible portions of a task, until after all of the sub-tasks of a divisible task complete execution or until the independent tasks executing on processors where the divisible task would be scheduled have completed execution.

B. Derivation of CT

As discussed in Section V-A, calculation of the (average) CT of a divisible task n_i requires evaluation of E[max$\{S_i\}$] where S_i is the set of random variables which determine the CT of a divisible task n_i. For example, in Figure 1, $S_1 = \{(t_{c11}), (t_{c12})\}$ for n_1 if it is scheduled on P_1 and P_2 (Note: Here t_{cij} indicates the time for each divisible portion of n_i executed on P_j). One may derive an analytic formula of E[max$\{S_i\}$] so that it can be used during scheduling. However, it is not possible to derive the closed-form analytic formula of E[max$\{S_i\}$] for any distribution of each random variable in S_i. In such a case, one may approximate E[max$\{S_i\}$] in terms of the mean and standard deviation of each random variable in S_i. Also, one can take a simulation approach to evaluating E[max$\{S_i\}$] given $\{S_i\}$. This would incur a substantial (computational) overhead during scheduling. However, it is not a fundamental limitation for a static scheduling scheme. In addition, in practice, the system may be monitored to measure the statistical parameters on processor speed, from which CT may be estimated.

In this study, the closed-form analytic formulas of E[max$\{S_i\}$] have been derived for the uniform distribution of each random variable in S_i. Let $max\{S_i\}$ be denoted by w. Then, the (average) CT of n_i is E[w] and the standard deviation of the CT, σ_w, (which is needed to derive the CT of any successor of n_i) is computed as $\sigma_w = E[w^2] - E[w]^2$. In the Appendix, E[w] and σ_w are provided for $|S_i| = 2$.

C. Complexity

The stochastic min-min and max-min have the same computational complexity. The complexity of the stochastic max-min is the same as that of the static max-min, which is $\boldsymbol{O(V^2P)}$, assuming that the average number of tasks is proportional to V throughout scheduling, where V is the number of total tasks and P is the number of processors. However, the step of deriving the CT of each node requires slightly more computation in the stochastic min-min and max-min using the formulas shown

in Section V-B, compared to the static min-min and max-min. In Figure 3, the average computation time required for deriving a schedule is compared, as a function of the number of processors, among the static min-min and max-min schemes, and stochastic min-min and max-min schemes. It is seen that the stochastic approaches at times took approximately two times longer than the static approaches. However, it is still less than 3 seconds in all the cases considered. Also, note that the scheduling overhead is incurred just once.

VI. Simulation

The simulation is divided into two sets of results. In the first set (refer to Tables III, IV, V and VI), task execution times exhibit varying temporal heterogeneity i.e., they have same mean execution time but different standard deviations in execution times for different processors. In the second set (refer to Tables VII and VIII), task execution times exhibit varying spatial and temporal heterogeneity i.e., they have different mean execution time and standard deviations for different processors. (Means of task execution times vary up to a maximum of 10% for different processors.)

- For tasks exhibiting temporal heterogeneity:
  - Divisible Tasks: $m_{ij} = (20, 40)$ and $\sigma_{ij} = (0, 15)$
  - Non-Divisible Tasks: $m_{ij} = (0, 20)$ and $\sigma_{ij} = (0, 10)$
- For tasks exhibiting temporal and spatial heterogeneity:
  - Divisible Tasks: $m_{ij} = (25, 45)$ and $\sigma_{ij} = (0, 15)$
  - Non-Divisible Tasks: $m_{ij} = (0, 20)$ and $\sigma_{ij} = (0, 10)$
- For max-min scheme, 15% of the tasks and for min-min scheme, 80% of the tasks are divisible whereas the rest are non-divisible.

A. Simulation Procedure

In order to analyze effectiveness of the new scheduling schemes (stochastic min-min and max-min), an extensive simulation has been carried out where they are compared with the static min-min and max-min in terms of the average parallel execution time t_p. The simulation procedures are described below.

1) Designate the number of tasks to be scheduled in the simulation.
2) Assign each task with a positive mean and standard deviation of execution time, which define a uniformly-distributed random variable.
3) Find a schedule using each of the static min-min, stochastic min-min, static max-min and stochastic max-min.
4) Generate random weights for all tasks according to their means and standard deviations.
5) Compute parallel execution time (T_p) for each of the schedules obtained in Step 3 using the static min-min, stochastic min-min, static max-min and stochastic max-min algorithms.
6) Repeat Steps 4 and 5 multiple times to obtain the average parallel execution time (t_p).

To obtain results for a different number of tasks, repeat Steps 1-7. In Step 2, equivalently, the range of task execution time, t_i, of n_i may be specified by its minimum t_{imin} and maximum t_{imax} such that the mean computation time, m_i, of n_i is $\frac{t_{imin}+t_{imax}}{2}$ and the standard deviation, σ_i, is $\frac{t_{imax}-t_{imin}}{2\sqrt{3}}$.

Step 6 is repeated 500 times in order to obtain simulation results (t_p).

B. Performance Measures

In addition to t_p, the following measures are used to analyze performance of the stochastic min-min and max-min algorithms.

- Δ: the difference between the schedule length, T_{SL}, and the average parallel execution time, t_p, which quantifies the accuracy of a scheduling scheme, i.e.,

$$\Delta = |T_{SL} - t_p|.$$

- η: the percent improvement in t_p by the stochastic algorithms (min-min and max-min) over their respective static versions, i.e.,

$$\eta = \frac{t_p^{static} - t_p^{stochastic}}{t_p^{static}} \times 100$$

where t_p^{static} and $t_p^{stochastic}$ are the average parallel execution times achieved by the static and stochastic min-min (or max-min), respectively.

VII. Results and Discussion

A. Effects of Temporal Heterogeneity

In Tables III, IV, VII and VIII, the make-span of a schedule (schedule length), T_{SL}, and the average parallel execution time, t_p, achieved by the schedule are compared. It can be seen that T_{SL} and t_p for the stochastic min-min and max-min match very closely, i.e., small Δ. This small difference (Δ) is due to the accurate estimation of CT in a temporally heterogeneous environment by considering the standard deviation (σ_i) in addition to the mean (m_i) of computation time for each task (n_i) (divisible and otherwise). However, there is a significant difference between T_{SL} and t_p in the static min-min and max-min which use only m_i ignoring σ_i, i.e., temporal heterogeneity. This is because effect of temporal heterogeneity on the CT of a task in the competing situation (due to presence of divisible tasks) becomes significant, but the static min-min and max-min ignore it. They tend to "under-estimate" the CT of tasks (n_k) involved in the competing situation since $E[max((t_{cij}), (t_{cik}))] > max((m_{cij}), (m_{cik}))$. Therefore, it often ends up with a premature scheduling of further tasks on processors which may be executing separate portions of a divisible task, leading to a worst schedule resulting in a longer t_p. Also, the under-estimation is a source for the large discrepancy between T_{SL} and t_p. Note that the difference is very small for the stochastic min-min and max-min as compared to their static counterparts (refer Tables III, IV, VII and VIII).

From these results, it is clear that (i) temporal heterogeneity has a significant effect on t_p, (ii) the proposed model of divisible task scheduling and the stochastic scheduling algorithms

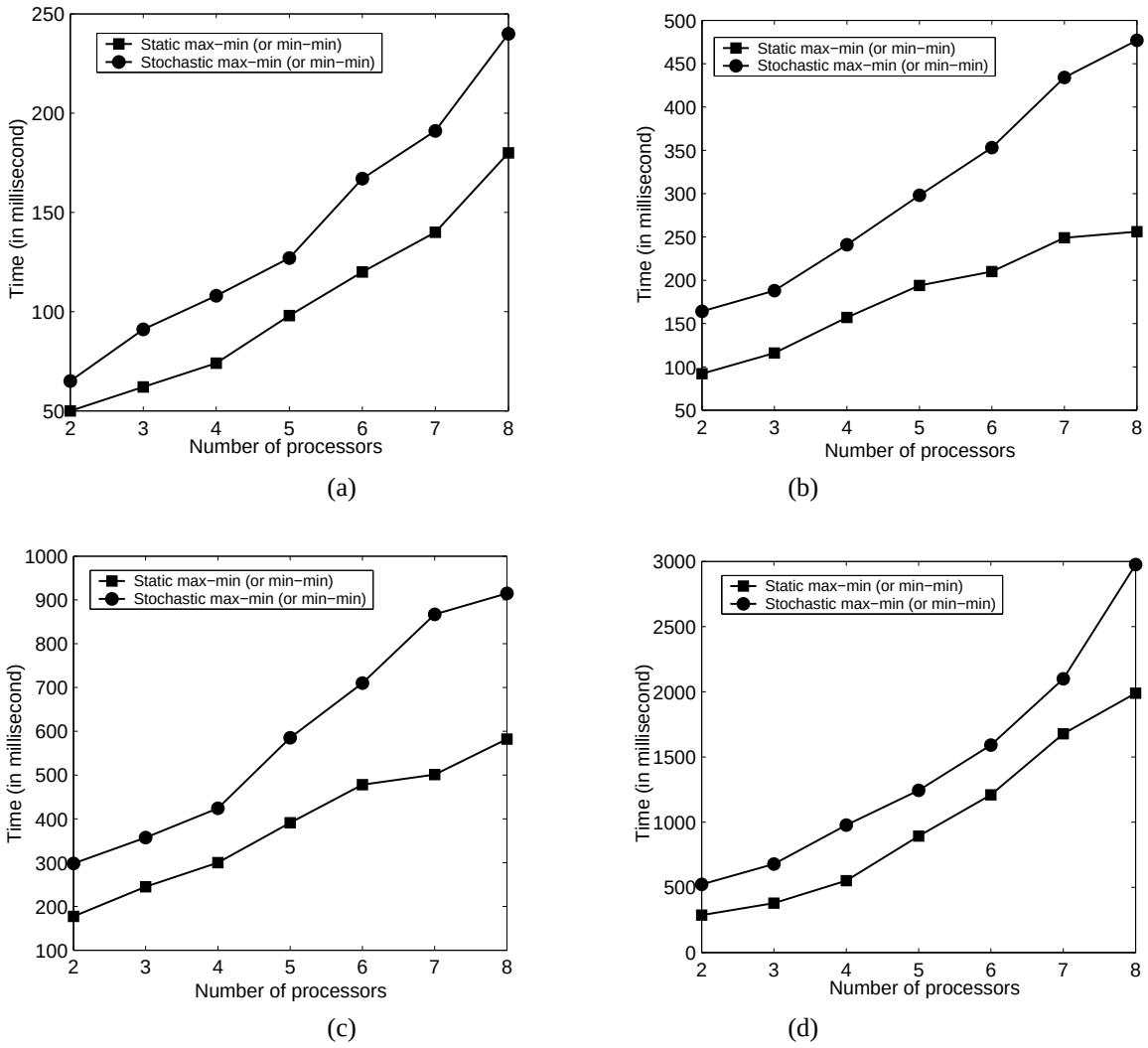

Fig. 3. Average time (in milliseconds) required to derive the schedule on Sun Enterprise 420R server (4 450-MHz UltraSPARC[tm]-II modules with 4GB memory) for (a) 40 nodes, (b) 100 nodes, (c) 200 nodes and (d) 500 nodes.

allow a highly accurate estimation of the average parallel execution time and (iii) scheduling divisible, independent tasks using only the average computation times of nodes would not lead to the best possible schedule.

B. Improvements by Stochastic min-min and max-min

In Tables V, VI, VII and VIII, the stochastic max-min and min-min are compared to the static max-min and min-min in terms of *percent improvement*, η.

When the competing situation is present, a large improvement has been achieved by the stochastic max-min and min-min (refer to Tables V, VI, VII and VIII). This significant improvement is due to the accurately estimated CT for divisible, independent tasks and its proper use in scheduling. Also, it can be observed that there is only a very small difference in variation of T_p between the static max-min (or min-min) and stochastic max-min (or min-min). It is noticed that in each of the simulations the improvement varies significantly with the number of processors used. The "degree" of competition in a competing situation depends not only on the number of divisble tasks and execution times but also on the number of divisible tasks. Suppose that a task is divided into two sub-tasks where their average computation times are the same or similar and at least one of them shows significant temporal heterogeneity. When these sub-tasks are scheduled concurrently, it will lead to a competing situation and any further tasks will be executed when the current divisible task has completed execution. A different number of processors implies a different "degree" of the competing situation. Hence, the improvement depends on the number of processors employed in scheduling. Higher the number of sub-tasks for a divisible tasks, greater is the "degree" of competing situation.

The competing situation is a natural occurance in parallel and distributed computing. When an application is decomposed into a set of tasks which in turn can be divided into sub-tasks for concurrent execution on multiple processors, the decomposition needs to be done such that the load is well balanced especially among the sub-tasks which are to

TABLE III

COMPARISON OF SCHEDULE LENGTH (T_{SL}) AND t_p FOR THE STATIC AND STOCHASTIC VERSIONS OF MAX-MIN IN THE PRESENCE OF TEMPORAL HETEROGENEITY IN TASK EXECUTION TIMES.

number of tasks	number of processors	Static max-min				Stochastic max-min			
		T_{SL}	t_p	σ_{T_p}	Δ	T_{SL}	t_p	σ_{T_p}	Δ
40	2	249.4	279.2	14.1	29.8	280.7	274.2	16.9	6.5
	3	167.4	199.5	10.8	32.1	197.1	195.9	9.3	1.2
	4	124.9	152.0	8.8	27.1	146.7	147.3	7.8	0.6
	5	100.7	121.4	7.3	20.7	117.5	117.0	6.3	0.5
	6	85.5	104.5	6.8	19.0	97.9	101.1	5.6	3.2
	7	73.0	91.4	5.9	18.4	83.3	85.8	4.4	2.5
	8	64.1	80.3	5.3	16.2	75.8	77.8	4.9	2.0
100	2	638.6	708.1	20.7	69.5	683.7	676.3	19.4	7.3
	3	426.9	497.0	17.7	70.0	488.1	485.9	16.0	2.2
	4	319.9	375.9	14.3	56.0	364.7	359.8	14.0	5.0
	5	256.1	303.1	11.3	47.0	289.3	295.6	11.2	6.3
	6	214.7	256.7	10.9	42.0	239.5	249.4	10.7	9.9
	7	183.3	223.8	9.7	40.5	207.1	217.8	10.2	10.7
	8	160.6	197.3	9.3	36.7	182.8	190.8	8.6	8.1
200	2	1223.8	1367.7	31.4	143.9	1324.6	1313.9	31.5	10.8
	3	816.5	945.8	25.0	129.2	931.6	923.8	27.1	7.8
	4	612.4	719.8	19.6	107.4	670.6	673.2	19.7	2.6
	5	490.4	575.3	16.4	84.9	557.0	555.6	16.7	1.5
	6	409.0	487.8	14.4	78.8	466.5	461.7	14.0	4.7
	7	350.3	416.0	12.1	65.8	399.2	406.5	15.0	7.3
	8	306.8	373.6	12.8	66.8	350.9	356.7	12.2	5.9
500	2	3156.3	3504.2	52.1	347.9	3384.8	3341.4	53.8	43.3
	3	2104.7	2403.5	41.3	298.8	2393.9	2376.9	39.7	17.1
	4	1578.4	1813.5	32.3	235.1	1742.0	1727.1	33.3	15.0
	5	1262.9	1469.5	26.3	206.6	1450.3	1432.4	27.7	17.9
	6	1052.7	1239.6	26.0	186.9	1201.8	1184.6	24.4	17.2
	7	902.5	1066.2	22.3	163.6	1039.7	1038.5	22.7	1.3
	8	789.7	937.6	18.6	147.9	898.6	896.6	18.8	1.9

TABLE IV

COMPARISON OF SCHEDULE LENGTH (T_{SL}) AND t_p FOR THE STATIC AND STOCHASTIC VERSIONS OF MIN-MIN IN THE PRESENCE OF TEMPORAL HETEROGENEITY IN TASK EXECUTION TIMES.

number of tasks	number of processors	Static min-min				Stochastic min-min			
		T_{SL}	t_p	σ_{T_p}	Δ	T_{SL}	t_p	σ_{T_p}	Δ
40	2	559.4	636.8	20.8	77.4	629.9	636.8	20.8	6.9
	3	520.6	595.9	19.4	75.3	580.1	588.6	17.5	8.6
	4	284.9	326.8	11.3	41.9	333.4	320.3	9.6	13.1
	5	262.9	316.6	12.5	53.7	291.4	297.3	6.8	5.9
	6	194.5	228.3	9.3	33.8	220.0	215.4	7.0	4.6
	7	182.8	215.0	8.5	32.3	204.8	209.9	6.8	5.0
	8	147.5	169.7	7.2	22.2	174.1	172.6	4.5	1.6
100	2	1339.0	1526.6	29.5	187.6	1532.1	1514.7	28.9	17.4
	3	1281.0	1474.0	29.1	192.9	1449.9	1433.9	23.2	16.1
	4	676.8	791.1	18.0	114.3	756.8	767.3	13.8	10.5
	5	648.0	774.2	18.3	126.2	715.1	725.7	12.7	10.6
	6	454.3	534.9	13.8	80.6	532.4	522.5	10.6	9.9
	7	434.8	529.6	13.7	94.7	490.0	493.5	10.1	3.5
	8	345.0	412.0	10.6	67.0	396.0	396.2	8.4	0.2
200	2	2613.8	2986.4	41.0	372.6	3025.1	2980.0	41.5	45.2
	3	2463.2	2852.9	42.9	389.6	2842.1	2816.6	35.1	25.4
	4	1321.1	1535.1	26.6	214.0	1470.1	1491.1	22.3	21.0
	5	1233.7	1469.6	23.7	235.9	1388.2	1412.0	17.2	23.8
	6	881.0	1032.0	22.0	151.0	1028.0	1016.5	15.8	11.4
	7	825.4	1004.8	18.7	179.4	939.7	949.5	12.5	9.8
	8	664.9	791.7	17.1	126.8	762.1	749.3	11.6	12.8
500	2	6429.1	7323.9	64.2	894.8	7386.8	7318.8	64.5	68.0
	3	6058.8	6980.5	62.9	921.7	6813.0	6907.2	48.7	94.3
	4	3219.6	3712.3	37.4	492.8	3697.3	3627.8	31.1	69.5
	5	3028.0	3596.5	37.0	568.5	3416.1	3449.2	24.2	33.1
	6	2150.7	2508.0	30.6	357.3	2395.7	2420.7	20.8	25.0
	7	2029.5	2453.3	27.5	423.8	2227.5	2283.0	16.7	55.5
	8	1616.3	1894.2	25.3	277.9	1805.3	1820.9	14.4	15.6

TABLE IX

VARIATION IN η WITH NUMBER OF TASKS WHEN TASKS EXHIBIT TEMPORAL HETEROGENEITY.

Scheduling Scheme	number of of tasks	η			
		Average	Maximum	Minimum	p-value
max-min	40	1.99	6.12	-3.13	< 0.001
	100	2.71	5.95	-1.52	< 0.001
	200	3.21	7.48	-1.56	< 0.001
	500	3.56	6.71	-0.26	< 0.001
min-min	40	2.45	7.5	-4.05	< 0.001
	100	2.86	8.51	-2.23	< 0.001
	200	2.91	8.15	-1.18	< 0.001
	500	3.21	8.39	-0.14	< 0.001

TABLE X

VARIATION IN η WITH NUMBER OF TASKS WHEN TASKS EXHIBIT TEMPORAL AND SPATIAL HETEROGENEITY.

Scheduling Scheme	number of of tasks	η			
		Average	Maximum	Minimum	p-value
max-min	40	1.89	6.3	-5.4	< 0.001
	100	2.1	4.7	-3.1	< 0.001
	200	2.41	4.48	-2	< 0.001
	500	2.86	3.8	0.2	< 0.001
min-min	40	1.45	7.9	-4.49	< 0.001
	100	1.86	4.97	-1.3	< 0.001
	200	2.11	6.7	-3.4	< 0.001
	500	2.54	5.29	-3.21	< 0.001

be executed simultaneously. That is, their sizes would be similar (if not equal), which is likely to lead to a competing situation. Therefore, one can expect a significant performance improvement by the proposed approach (the stochastic max-min and min-min) in most cases.

C. Dependency on number of tasks

The values of η for the different number of tasks (over all simulations wherein number of processors ranges from 2-8) used in the simulation are given in Tables IX and X. Here, it is seen that as the number of tasks involved in the simulation increases, η increases for both max-min and min-min scheduling schemes. The last column in the table indicates the one-sided p-value from the Wilcoxon Signed Rank test. For all cases, the p-value is < 0.001 indicating that our results are statistically significant i.e., a reduction in t_p is achieved by using the stochastic scheduling approach over the static approach. Also, it can be concluded that for the stochastic max-min and min-min to be more effective number of tasks should be higher.

VIII. SUMMARY

Heterogeneity is common in cluster and Grid computing environments. In particular, temporal heterogeneity makes computation time of a task on a computer or communication time between tasks vary with time. That is, the task execution times are now random variables. A conventional scheduling scheme which considers only the means of task execution times is not able to find the best possible schedule.

In this paper, a new approach to scheduling groups of independent tasks is proposed and performance of its first implementations has been analyzed through simulation. The approach considers the possibility of an independent task requiring multiple processors for execution. It takes into account the standard deviations of completion times of sub-tasks belonging to an independent task being executed concurrently on multile processors. It uses the standard deviation in addition to the mean execution times in order to accurately estimate the CT of each divisible, independent task, and utilizes the accurate CT judiciously in scheduling. This approach allows one to closely "follow" during scheduling what would happen on a temporally heterogeneous distributed system if a given application consisting of independent tasks (divisible and whole) is executed on the system, and find a schedule leading to a shorter average parallel execution time.

The first implementations of the approach based on the max-min and min-min algorithms, called the stochastic max-min and min-min, have well demonstrated that the schedules derived by the proposed approach are significantly better in terms of the average parallel execution time than those by the static max-min and min-min which consider only the average execution times of tasks, ignoring the standard deviations. Also, the stochastic max-min and min-min are able to accurately predict the actual performance one can expect on a temporally heterogeneous distributed computing system, i.e., the schedule length obtained by the stochastic max-min and min-min is very close to the average parallel execution time. While the max-min and min-min were considered in this study, it should be clear that the proposed approach is applicable to any other scheduling schemes which use time parameters in computing the priority levels of tasks during scheduling.

REFERENCES

[1] Soo-Young Lee and Jun Huang, "A Theoretical Approach to Load Balancing of a Target Task in a Temporally and Spatially Heterogeneous Grid Computing Environment," *GRID 2002*, pp. 70-81.

[2] Topcuoglu, H., Hariri, S. and Min-You Wu, "Task scheduling algorithms for heterogeneous processors," *Proceedings of the Eighth Heterogeneous Computing Workshop, 1999 (HCW '99)*, pp. 3-14, April 1999.

[3] Zhenying Liu, Binxing Fang, Yi Zhang and Jianqi Tang, "Scheduling algorithms for a fork DAG in a NOWs," *Proceedings of the Fourth International Conference/Exhibition on High Performance Computing in the Asia-Pacific Region, Vol. 2* , pp. 959-960, May 2000.

[4] V. A. F. Almeida , I. M. M. Vasconcelos , J. N. C. rabe and D. A. Menasc, "Using random task graphs to investigate the potential benefits of heterogeneity in parallel systems," *Proceedings of the 1992 ACM/IEEE conference on Supercomputing*, pp. 683-691, Nov. 1992.

[5] Topcuoglu, H., Hariri, S. and Min-You Wu, " Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE Transactions on Parallel and Distributed Systems, Vol. 13 Issue: 3*, pp. 260-274, March 2002.

[6] Dingchao Li and Ishii, N., "Scheduling task graphs onto heterogeneous multiprocessors," *Proceedings of IEEE Region 10's Ninth Annual International Conference. Theme: 'Frontiers of Computer Technology' (TENCON '94)*, pp. 556-563 vol.2, Aug. 1994.

[7] Rashmi Bajaj and Dharma P. Agarwal, "Improving Scheduling of Tasks in a Heterogeneous Environment," *IEEE Transactions on Parallel and Distributed Systems, Vol. 15 No. 2*,pp. 107-118 February 2004.

[8] Michael A. Iverson, Fusun Ozguner and Lee C. Potter, "Statistical Prediction of Task Execution Times Through Analytic Benchmarking for Scheduling in a Heterogeneous Environment," *Proceedings of the 8th Heterogeneous Computing Workshop (HCW '99)*, p. 99, April 1999.

TABLE V

IMPROVEMENTS IN t_p BY THE STOCHASTIC MAX-MIN OVER THE STATIC MAX-MIN WHEN THE COMPETING SITUATION IS SIGNIFICANT IN THE PRESENCE OF TEMPORAL HETEROGENEITY IN TASK EXECUTION TIMES.

number of tasks	number of processors	Static max-min			Stochastic max-min			η
		T_{SL}	t_p	σ_{T_p}	T_{SL}	t_p	σ_{T_p}	
40	2	271.2	306.4	16.0	296.3	293.3	15.8	4.3
	3	181.3	213.3	12.4	198.5	207.9	11.3	2.5
	4	136.2	164.5	9.8	150.5	157.0	11.0	4.6
	5	109.5	130.3	8.2	120.5	129.2	8.4	0.8
	6	92.3	109.3	6.3	102.2	107.9	7.7	1.3
	7	79.7	98.7	6.4	86.8	94.2	6.6	4.5
	8	69.8	85.5	5.3	77.7	81.7	5.1	4.5
100	2	686.4	758.3	21.7	738.4	721.6	25.1	4.8
	3	455.8	525.7	20.6	518.7	515.6	19.3	1.9
	4	343.1	397.8	17.1	390.0	378.5	14.9	4.9
	5	276.9	326.1	15.0	307.9	309.9	12.7	5.0
	6	227.5	271.7	12.1	259.7	260.5	11.4	4.1
	7	193.3	234.8	9.7	222.8	230.1	10.6	2.0
	8	169.5	207.1	8.6	195.0	199.7	9.1	3.6
200	2	1281.8	1437.3	32.5	1357.4	1373.9	32.2	4.4
	3	855.4	979.5	24.1	985.3	975.8	24.5	0.4
	4	641.8	750.5	20.2	721.5	707.1	20.2	5.8
	5	514.0	611.1	18.3	592.0	588.5	17.7	3.7
	6	427.8	509.2	14.4	489.8	490.9	15.5	3.6
	7	367.3	438.0	13.5	414.5	425.1	13.5	3.0
	8	321.4	387.5	12.2	370.2	374.9	12.1	3.2
500	2	3316.9	3653.1	52.8	3479.6	3457.8	51.6	5.3
	3	2216.3	2490.6	37.2	2450.0	2429.9	41.6	2.4
	4	1654.8	1886.6	30.4	1794.9	1773.1	33.3	6.0
	5	1320.7	1537.4	28.7	1508.7	1492.3	30.0	2.9
	6	1099.2	1278.7	24.3	1233.6	1205.5	25.2	5.7
	7	940.8	1100.6	21.1	1088.4	1081.8	24.4	1.7
	8	826.4	973.7	19.0	948.1	927.4	21.1	4.8

TABLE VI

IMPROVEMENTS IN t_p BY THE STOCHASTIC MIN-MIN OVER THE STATIC MIN-MIN WHEN THE COMPETING SITUATION IS SIGNIFICANT IN THE PRESENCE OF TEMPORAL HETEROGENEITY IN TASK EXECUTION TIMES.

number of tasks	number of processors	Static min-min			Stochastic min-min			η
		T_{SL}	t_p	σ_{T_p}	T_{SL}	t_p	σ_{T_p}	
40	2	506.4	577.9	19.8	581.6	571.5	19.3	1.1
	3	478.8	555.4	20.1	524.5	539.4	16.0	2.9
	4	258.5	298.1	11.1	301.4	289.8	9.8	2.8
	5	244.7	289.3	11.2	262.0	271.6	7.2	6.1
	6	175.1	205.7	9.4	202.8	196.4	6.8	4.5
	7	167.5	201.4	8.8	183.3	188.7	7.2	6.3
	8	134.7	160.6	8.0	160.4	150.0	5.5	6.6
100	2	1339.0	1526.6	29.5	1552.1	1514.7	28.9	0.8
	3	1281.0	1474.0	29.1	1419.9	1433.9	23.2	2.7
	4	676.8	791.1	18.0	746.8	767.3	13.8	3.0
	5	648.0	774.2	18.3	705.1	725.7	12.7	6.3
	6	454.3	534.9	13.8	542.4	522.5	10.6	2.3
	7	434.8	529.6	13.7	480.0	493.5	10.1	6.8
	8	345.0	412.0	10.6	406.0	396.2	8.4	3.8
200	2	2615.0	3006.2	40.2	3062.4	2995.6	41.1	0.4
	3	2503.6	2892.2	41.2	2875.9	2841.5	34.5	1.8
	4	1310.8	1518.6	24.1	1484.5	1476.9	20.3	2.7
	5	1257.0	1497.5	25.3	1388.4	1401.0	15.6	6.4
	6	878.9	1026.0	18.6	1020.4	995.9	12.7	2.9
	7	842.6	1026.0	18.3	929.4	945.4	10.9	7.9
	8	662.3	789.2	17.4	773.9	753.0	9.1	4.6
500	2	6380.8	7286.4	68.0	7315.6	7270.5	67.5	0.2
	3	6050.9	6949.4	61.6	6829.6	6887.4	52.5	0.9
	4	3197.1	3720.2	40.2	3515.8	3592.0	30.1	3.4
	5	3031.9	3599.6	36.7	3479.2	3419.6	24.1	5.0
	6	2136.2	2496.4	30.8	2364.5	2397.0	19.8	4.0
	7	2015.0	2441.9	26.4	2211.5	2264.9	16.8	7.2
	8	1605.6	1892.0	22.8	1850.5	1805.4	13.9	4.6

TABLE VII

COMPARISON OF SCHEDULE LENGTH (T_{SL}) AND t_p AND IMPROVEMENTS IN t_p BY THE STOCHASTIC MIN-MIN OVER THE STATIC MIN-MIN WHEN THE COMPETING SITUATION IS SIGNIFICANT IN PRESENCE OF TEMPORAL AND SPATIAL HETEROGENEITY IN TASK EXECUTION TIMES.

number of tasks	number of processors	Static min-min				Stochastic min-min				
		T_{SL}	t_p	σ_{T_p}	Δ	T_{SL}	t_p	σ_{T_p}	Δ	η
40	2	703.9	778.1	21	74.1	795.5	778.1	21	17.4	0
	3	649.4	754.2	23.1	104.8	768.2	745.5	18	22.7	1.2
	4	353	400.7	12.8	47.7	403.1	394.3	11	8.9	1.6
	5	333.9	389.4	12.7	55.4	368.2	380.2	10.1	11.9	2.4
	6	232	273.3	9.5	41.4	290.6	273.1	8.5	17.5	0.1
	7	228.8	269.7	9.9	40.9	252.9	261	8.9	8.1	3.2
	8	184.3	217.8	9.9	33.6	216.8	204.5	5.5	12.3	6.1
100	2	1643.5	1791.2	32	147.7	1748.2	1784.4	32	36.2	0.4
	3	1516.2	1719	31	202.7	1679.2	1693.3	25	14.1	1.5
	4	809	902.1	19	93	879.5	892.8	16.9	13.3	1
	5	760.8	881	18.5	120.2	833.2	846.7	12.4	13.5	3.9
	6	542	608.5	15.5	66.4	616.6	599.2	10.3	7.3	1.5
	7	509.2	602.5	14.6	93.3	561.3	576.7	10.8	15.4	4.3
	8	406.7	464.9	12.7	58.1	465	451.6	8.2	13.4	2.9
200	2	2900.5	3221.5	44.3	320.9	3239.5	3210	42.1	29.5	0.4
	3	2587.5	3022.7	43.8	435.2	2925	3018.4	35.8	93.3	0.1
	4	1431.1	1628.8	28.1	197.8	1648.7	1604.3	21.2	44.4	1.5
	5	1280.3	1544.1	25.7	263.8	1540.5	1505.2	19.6	35.3	2.5
	6	942.3	1104.5	20.4	162.2	1054	1068.5	15.5	14.5	3.3
	7	856.4	1052.3	20.3	195.9	979.5	1012.3	13.2	32.9	3.8
	8	704.5	845.7	16.9	141.3	820.6	804.4	12.2	16.2	4.9
500	2	7496	8344	67.4	848	8396.3	8338	67.7	58.3	0.1
	3	6759.2	7860.5	64	1101.3	7854.7	7885.7	54.1	31	-0.3
	4	3657.4	4179.7	43.4	522.3	4007.7	4083.5	31.9	75.8	2.3
	5	3353.1	3987.5	40.5	634.4	3823.8	3896.6	26.3	72.8	2.3
	6	2416.1	2811.7	32.4	395.6	2831.1	2716.2	20.7	114.9	3.4
	7	2233.6	2689.9	28.9	456.3	2528	2586.6	18	58.5	3.8
	8	1798.2	2125	24.7	326.8	2053.9	2043.2	16.7	10.8	3.9

TABLE VIII

COMPARISON OF SCHEDULE LENGTH (T_{SL}) AND t_p AND IMPROVEMENTS IN t_p BY THE STOCHASTIC MAX-MIN OVER THE STATIC MAX-MIN WHEN THE COMPETING SITUATION IS SIGNIFICANT IN PRESENCE OF TEMPORAL AND SPATIAL HETEROGENEITY IN TASK EXECUTION TIMES.

number of tasks	number of processors	Static max-min				Stochastic max-min				
		T_{SL}	t_p	σ_{T_p}	Δ	T_{SL}	t_p	σ_{T_p}	Δ	η
40	2	413.5	464.6	21.4	51.1	460.3	453	21.8	7.3	2.5
	3	282.9	328.2	15.5	45.3	323	329.6	15.4	6.7	-0.4
	4	207.5	243.8	11.7	36.3	237.4	238.6	10.5	1.2	2.1
	5	166.4	201.5	9.8	35.1	196.8	194.9	10.2	1.9	3.2
	6	135.5	167	8.8	31.5	163.6	161.1	9.3	2.5	3.5
	7	120	147.1	8.4	27.1	145.2	142.3	7.2	2.9	3.3
	8	102.3	128.5	7.6	26.2	120.8	127.5	7.2	6.7	0.8
100	2	928	1027.2	31	99.1	981	994	31.1	13	3.2
	3	631.9	719.4	22	87.6	711.1	718.8	26.9	7.7	0.1
	4	474	544.5	17.9	70.4	519.8	526.4	16.7	6.6	3.3
	5	379.4	443.6	14.5	64.2	432.6	435	14.4	2.4	1.9
	6	312.4	373	14	60.6	368.1	372.5	14.8	4.3	0.1
	7	273.6	323.1	12.1	49.4	309.8	318.9	11.9	9	1.3
	8	236	281.4	11	45.5	270	272.7	10.4	2.8	3.1
200	2	1441.1	1562.2	38.6	121.1	1539.1	1509.1	38.8	29.9	3.4
	3	951.9	1077.2	30.5	125.2	1050.8	1064.5	30.7	13.7	1.2
	4	711.6	809.1	20.8	97.5	787.6	800.6	24.8	12.9	1
	5	573	673.4	22.4	100.4	639.7	643.4	19.4	3.7	4.5
	6	477.6	563.7	16.9	86.1	548	540.2	19.2	7.9	4.2
	7	404.5	476.7	15.9	72.2	452.4	466.3	16	13.9	2.2
	8	352.7	418.8	14.2	66.1	387.9	409.5	14.9	21.6	2.2
500	2	6377.1	7063.4	88.5	686.3	6895.9	6845	88.1	50.9	3.1
	3	4306	4952.3	68.1	646.3	4919.7	4902.9	64.1	16.8	1
	4	3240.5	3782.8	54.9	542.2	3660.4	3676.7	51.9	16.3	2.8
	5	2585.8	3072.3	47.2	486.4	3019.2	3011.6	43.1	7.6	2
	6	2141.6	2551.1	41.7	409.5	2474.7	2493.9	39.7	19.2	2.2
	7	1823.5	2214.2	39.3	390.8	2140.8	2155.5	36.9	14.7	2.7
	8	1583.8	1933.7	33.5	349.9	1860.9	1885.6	32.6	24.7	2.5

[9] Kamthe, A.; Soo-Young Lee, "A Stochastic Approach to Estimating Earliest Start Times of Nodes for Scheduling DAGs on Heterogeneous Distributed Computing Systems," *Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International* , vol., no.pp. 121b- 121b, 04-08 April 2005.

[10] Dogan, A. and Ozguner, F., "Stochastic scheduling of a meta-task in heterogeneous distributed computing," *International Conference on Parallel Processing Workshops*,pp. 369-374, 2001.

[11] Beaumont, O.; Legrand, A.; Robert, Y., "Scheduling strategies for mixed data and task parallelism on heterogeneous clusters and Grids," *Parallel, Distributed and Network-Based Processing, 2003. Proceedings. Eleventh Euromicro Conference on* , Feb. 2003.

[12] Qiang-Sheng Hua; Zhi-Gang Chen; Lau, F.C.M., "A new method for independent task scheduling in nonlinearly DAG clustering," *Proceedings of the 7th International Symposium on Parallel Architectures, Algorithms and Networks*, 2004.

[13] Beaumont, O.; Legrand, A.; Marchal, L.; Robert, Y., "Independent and divisible tasks scheduling on heterogeneous star-shaped platforms with limited memory," *Parallel, Distributed and Network-Based Processing, 2005. PDP 2005. 13th Euromicro Conference on*, Feb. 2005.

[14] Marchal, L.; Yang, Y.; Casanova. H., Robert, Y., "Steady State Scheduling of Multiple Divisible Load Applications on Wide-Area Distributed Computing Platform," *to appear in the International Journal of High Performance Computing Applications.*