http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 7 (4): 679-683, 2008
ISSN 1812-5638

MaxStd: A Task Scheduling Heuristic for Heterogeneous Computing Environment

Ehsan Ullah Munir, Jianzhong Li, Shengfei Shi, Zhaonian Zou and Donghua Yang
School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China

Abstract: In this study, a new task scheduling heuristic, maximum standard deviation (MaxStd) is proposed which considers that task for scheduling first, which has maximum standard deviation of the expected execution time on different machines. Experimentation results verify the effectiveness of proposed heuristic in different scenarios and comparison with the existing heuristics implies that proposed heuristic outperforms existing heuristics in terms of makespan.

Key words: Heterogeneous computing, task scheduling, MaxStd heuristic

INTRODUCTION

Heterogeneous Computing (HC) environment consists of different resources connected with high-speed links to provide a variety of computational capabilities for computing-intensive applications having multifarious computational requirements (Braun *et al*., 2001). In HC environment system application is decomposed into various tasks and each task should be assigned to one of the machines, which is best suited for its execution to minimize the total execution time. Therefore, an efficient assignment scheme responsible for allocating the application tasks to the machines is needed. Developing these strategies is a focus of lot of researchers (Sakellariou and Zhao, 2004; Shivle *et al*., 2006; Luo *et al*., 2007) nowadays, which makes it an important area of research.

The problem of an optimal assignment of tasks to machines is proven to be NP-complete requiring use of heuristics to find the near optimal solution (Baca, 1989). Plethora of heuristics has been proposed for assignment of tasks to machines in HC environment. Min-min (Freund *et al*., 1998) gives the highest priority to the task for scheduling, which can be completed earliest. The idea behind Min-min heuristic is to finish each task as early as possible and hence, it schedules the tasks with the selection criterion of minimum earliest completion time. Max-min (Freund *et al*., 1998) heuristic is very similar to the Min-min, which gives the highest priority to the task with the maximum earliest completion time for scheduling. In Max-min heuristic long running tasks are overlapped with short-running ones. The Heaviest Task First (HTF) heuristic (Yarmolenko *et al*., 2000) computes each task's minimum execution time on all machines and the task with the maximum execution time is selected. The selected task is the heaviest task among all tasks (note that the Max-Min algorithm selects the task with the latest minimum completion time, which may not be the heaviest one). Then this heaviest task is assigned to the machine on which this task has minimum completion time. The Sufferage heuristic (Maheswaran *et al*., 1999) is based on the idea that better mappings can be generated by assigning a task to machine that would suffer most in terms of expected completion time if that particular machine is not assigned to it. Sufferage heuristic assigns each task its priority according to its sufferage value. For each task, its sufferage value is equal to the difference between its best completion time and its second best completion time. In Segmented Min-min heuristic (Wu *et al*., 2000) the tasks are divided into four groups based on their minimum, maximum or average expected execution time and then Min-min is applied on each group for scheduling. A new criterion to minimize completion time of non-makespan machines is introduced (Briceno *et al*., 2007). It is noted that although completion time of non-makespan machine can be reduced but it can increase the overall system makespan as well. The comparison of eleven heuristics is given in Braun *et al*. (2001) and the Min-min heuristic is declared the best among all the other heuristics considered based on makespan criterion. Balanced Minimum Completion Time (BMCT), heuristic for scheduling independent tasks is given and compared (Sakellariou and Zhao, 2004), which works in two steps: In first step it assigns each task to machine which minimize the execution time and then in second phase try to balance the load by swapping tasks in order to minimize completion time. Minimum standard deviation first (minSD) heuristics is proposed in Luo *et al*. (2007) where the task having the minimum standard deviation is scheduled first.

Corresponding Author: Ehsan Ullah Munir, School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China

In this study, a new task scheduling heuristic, maximum standard deviation (MaxStd) is proposed which considers that task for scheduling first which has maximum standard deviation of the expected execution time on different machines. Intuition is, tasks having low standard deviation exhibit less difference in their expected execution time on different machines, therefore assigning them earlier or later do not impact system performance much. Conversely, tasks with high standard deviation have more divergence in execution time on different machines, so these tasks must be assigned earlier for scheduling, otherwise it can deteriorate the system performance.

A large number of experiments were conducted on synthetic datasets to show the superiority of the proposed heuristic against the existing ones. We used the Coefficient of Variation (COV) based method for generating synthetic datasets, which provides greater control over spread of heterogeneity (Ali *et al*., 2000). The experimental results clearly show that the proposed heuristic outperforms the existing heuristics in terms of makespan.

PROBLEM STATEMENT

Let $T = \{t_1, t_2, ..., t_m\}$ be set of tasks, $M = \{m_1, m_2, ..., m_n\}$ be set of machines and expected time to compute (ETC) is a m×n matrix where entry e_{ij} represents the expected execution time of task t_i on machine m_j. Machine availability time mat (m_j) is the earliest time machine m_j can complete the execution of all the tasks that have previously been assigned to it (based on the ETC entries for those tasks). The completion time ct (t_i, m_j) of a task is equal to the expected execution time of task t_i plus the machine availability time of machine m_j and is given as ct (t_i, m_j) = e_{ij} + mat (m_j). Makespan is equal to maximum completion time of all tasks i.e., max mat (m_j) for ($1 \le j \le n$). Provided with T, M and ETC our objective is to find the assignment which minimizes makespan.

MaxStd HEURISTIC

Here we explain the idea and working of MaxStd heuristic. In MaxStd heuristic task having the highest standard deviation of their expected execution time is scheduled first. Tasks having low standard deviation of task execution time have less variation in execution time on different machines and hence, their delayed assignment for scheduling will not affect overall makespan much. Moreover, the tasks with higher standard deviation of task execution time exhibit more variation in their execution time on different machines. A delayed assignment of such tasks might hinder their chances of occupying faster machines as some other tasks might occupy these machines earlier. Such a scenario would result in an increase in the system makespan. Hence, the tasks having high standard deviation must be scheduled first.

1. **while** $T \neq \phi$
2. **for** each task $t_i \in T$ **do**
3. **for** $j = 1,...,n$ **do**
4. $c_{ij} = e_{ij} + mat(m_j)$ **do**
5. **endfor**
6. **endfor**
7. compute the standard deviation of each task
8. find a task t_k having the highest standard deviation
9. assign task t_k to machine m_l which gives the earliest completion time
10. delete task t_k from T
11. update mat (m_l) which is completion time for machine
12. **endwhile**

Fig. 1: MaxStd heuristic

Given a set of tasks $T = \{t_1, t_2, ..., t_m\}$, a set of machines $M = \{m_1, m_2, ..., m_n\}$ and an ETC matrix, the solution using MaxStd heuristic can be found in following steps: 1) Compute the standard deviation of expected execution time of all tasks on machines 2) Find the task having highest standard deviation 3) Find the machine which has the earliest completion time for the task 4) Assign the task to machine and remove it from the list of tasks 5) update the completion time for machine 6) repeat step 1-5 until there are no tasks to schedule.

These steps are formulated in the form of Algorithm and given in Fig. 1.

In Fig. 1 lines 3-5 calculate the completion time for task t_i on n machines so time taken from lines 3-5 is O(n). There are total m tasks so time taken by lines 2-6 is O(mn). In lines 7-11 one task is assigned to machine so time taken is O(mn), hence the time complexity of proposed algorithm is $O(m^2n)$.

A scenario of ETC is given in Table 1 where MaxStd heuristic performs better than Min-min, Max-min and Sufferage heuristic. All the machines are assumed to be idle for this case. Makespan of MaxStd heuristic is 29 while the makespan produced by Min-min, Max-min and Sufferage are 44, 42 and 42, respectively, which clearly show the proposed heuristic outperforms existing heuristics in terms of makespan. Table 2 shows how the results are derived using MaxStd heuristic. Figure 2 shows the visual representation of task assignment in MaxStd heuristic.

Table 1: ETC matrix where MaxStd outperforms all other heuristics

	Machines		
Task	m_1	m_2	m_3
t_1	23	12	25
t_2	03	11	11
t_3	24	17	44
t_4	03	02	04
t_5	22	23	26
t_6	14	18	11
t_7	26	13	17

Table 2: Execution process of MaxStd heuristic

Assignment order	Std	m_1.CT	m_2.CT	m_3.CT
$t_3 \rightarrow m_2$	14.01	24	17	44
$t_6 \rightarrow m_3$	13.07	14	35	11
$t_2 \rightarrow m_1$	13.05	3	28	22
$t_5 \rightarrow m_1$	07.93	25	40	37
$t_7 \rightarrow m_3$	12.74	51	30	28
$t_1 \rightarrow m_2$	12.66	48	29	53
$t_4 \rightarrow m_1$	02.08	28	31	32

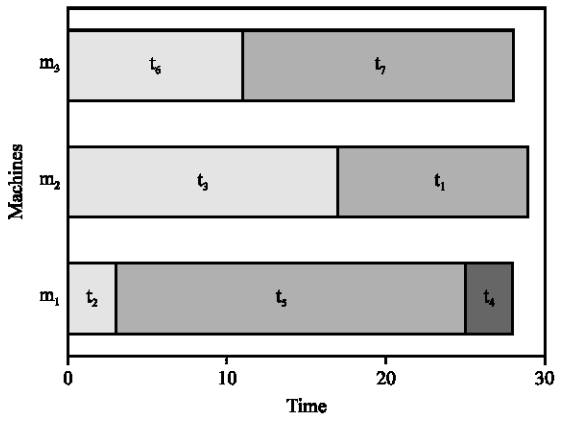

Fig. 2: Visual representation of task assignment in MaxStd heuristic

RESULTS AND DISCUSSION

Dataset: In this experiments, COV-based ETC generation method is used to simulate different HC environments by changing the parameters μ_{task}, V_{task} and $V_{machine}$, which represent the mean task execution time, the task heterogeneity and the machine heterogeneity, respectively. The COV based method provides greater control over the spread of the execution time values than the common range-based method used previously (Braun *et al*., 2001; Shivle *et al*., 2005).

The COV-based ETC generation method works as follows (Ali *et al*., 2000): First, a task vector, q, of expected execution times with the desired task heterogeneity is generated following gamma distribution with mean μ_{task} and standard deviation $\mu_{task}*V_{task}$. The input parameter μ_{task} is used to set the average of the values in q. The input parameter V_{task} is the desired coefficient of variation of the values in q. The value of V_{task} quantifies task heterogeneity and is larger for high task heterogeneity. Each element of the task vector q is then used to produce one row of the ETC matrix following gamma distribution with mean q[i] and standard deviation $q[i]*V_{machine}$ such that the desired coefficient of variation of values in each row is $V_{machine}$, another input parameter. The value of $V_{machine}$ quantifies machine heterogeneity and is larger for high machine heterogeneity.

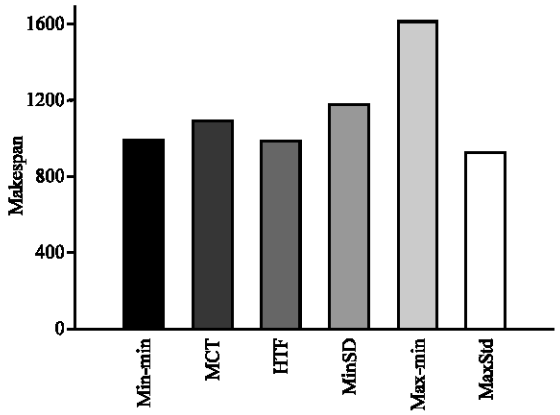

Fig. 3: Comparison in makespan for high task heterogeneity, high machine heterogeneity ETC

Performance evaluation: A simulation-based framework is used to evaluate the performance of proposed heuristic. In all the experiments, the size of ETCs is 512×16, the mean of task execution time μ_{task} is 100 and the task COV V_{task} is in [0.1, 0.6] while the machine COV $V_{machine}$ is between [0.1, 0.6]. Small values of V_{task} and $V_{machine}$ represent low heterogeneity, while large values of V_{task} and $V_{machine}$ represent high heterogeneity. Performance metric used for comparison is makespan, which is defined as the maximum time required completing set of tasks. The heuristic, which gives the shortest makespan, is declared the best.

Four categories were used for the ETC matrix (a) high task heterogeneity and high machine heterogeneity (HiHi), (b) high task heterogeneity and low machine heterogeneity (HiLo), (c) low task heterogeneity and high machine heterogeneity (LoHi) and (d) low task heterogeneity and low machine heterogeneity (LoLo).

Figure 3 shows the results for high task heterogeneity and high machine heterogeneity (HiHi) ETC, from the results we can clearly see that makespan of MaxStd is less than all other heuristics considered here for comparison. Moreover we can see that makespan produced by Max-min is the worst among all. Min-min performed second best.

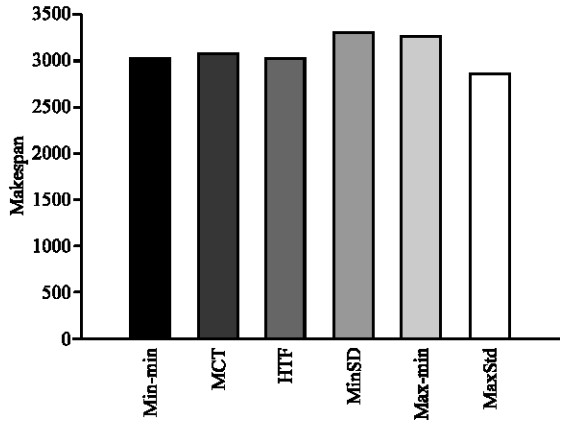

Fig. 4: Comparison in makespan for high task heterogeneity, low machine heterogeneity ETC

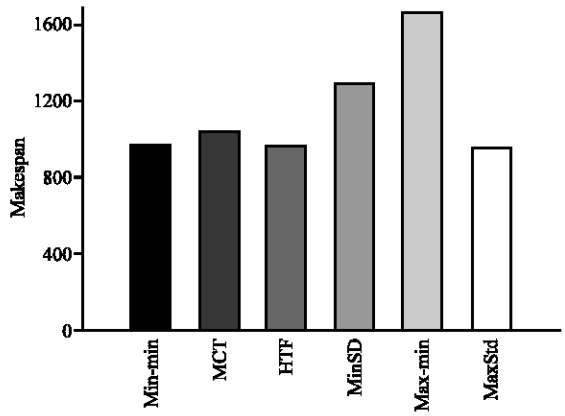

Fig. 5: Comparison in makespan for low task heterogeneity and high machine heterogeneity ETC

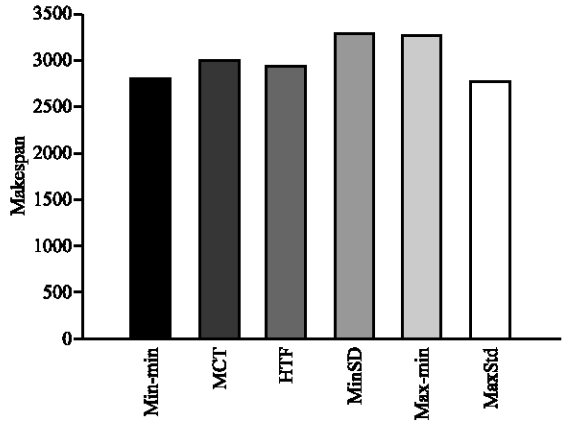

Fig. 6: Comparison in makespan for low task heterogeneity and low machine heterogeneity ETC

The results for high task heterogeneity and low machine heterogeneity (HiLo) are depicted in Fig. 4 where we can clearly see the MaxStd heuristic outperforms all other heuristics in terms of makespan and minSD performed the worst in this case.

Figure 5 and 6 show the results for low task heterogeneity and high machine heterogeneity (LoHi) and low task heterogeneity and low machine heterogeneity (LoLo) respectively. From the results it is evident that MaxStd outperforms all other heuristics considered here for comparison.

CONCLUSIONS

In this study, we have proposed a novel scheduling heuristic MaxStd for task scheduling in heterogeneous computing environment which considers first, the task having highest standard deviation of expected execution time for scheduling. We have compared proposed heuristic to other heuristics within a simulated heterogeneous computing environment. Simulation results show that proposed scheduling heuristic has a significant performance gain in terms of reduced makespan and outperforms all other heuristics considered here for comparison.

ACKNOWLEDGMENTS

This study is supported by the Key Program of the National Natural Science Foundation of China under Grant No. 60533110; the National Natural Science Foundation of China under Grant No. 60473075; the National Grand Fundamental Research 973 Program of China under Grant No. 2006CB303000; the Program for New Century Excellent Talents in University of China under Grant No. NCET-05-0333; the Heilongjiang Province Scientific and Technological Special Fund for Young Scholars under Grant No. QC06C033. COMSATS Institute of Information Technology (CIIT), Pakistan provides Ph.D scholarship for Mr. Ehsan Ullah Munir.

REFERENCES

Ali, S., H.J. Siegel, M. Maheswaran, S. Ali and D. Hensgen, 2000. Task execution time modeling for heterogeneous computing systems. Proceedings of the 9th Heterogeneous Computing Workshop, pp: 185-199.

Baca, D.F., 1989. Allocating modules to processors in a distributed system. IEEE Trans. Software Eng., 15 (11): 1427-1436.

Braun, T.D., H.J. Siegel and N. Beck, 2001. A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. J. Parallel Distributed Comput., 61 (6): 810-837.

Briceno, L.D., M. Oltikar, H.J. Siegel and A.A. Maciejewski, 2007. Study of an iterative technique to minimize completion times of non-makespan machines. Proceedings of the 17th Parallel and Distributed Processing Symposium, IPDPS 26-30 March 2007, pp: 1-14.

Freund, R.F., M. Gherrity, S. Ambrosius, M. Campbell, M. Halderman, D. Hensgen, E. Keith, T. Kidd, M. Kussow, J.D. Lima, M. Mirabile, L. Moore, B. Rust and H.J. Siegel, 1998. Scheduling resources in multi-user heterogeneous computing environments with smartnet. Proceedings of the 7th Heterogeneous Computing Workshop, pp: 184-199.

Luo, P., K. Lu and Z.Z. Shi, 2007. A revisit of fast greedy heuristics for mapping a class of independent tasks onto heterogeneous computing systems. J. Parallel Distributed Comput., 67 (6): 695-714.

Maheswaran, M., S. Ali, H.J. Siegel, D. Hensgen and R.F. Freund, 1999. Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems. Proceedings of the 8th IEEE Heterogeneous Computing Workshop, pp: 30-44.

Sakellariou, R. and H. Zhao, 2004. A hybrid heuristic for DAG scheduling on heterogeneous systems. Proceedings of the 18th International Parallel and Distributed Processing Symposium, 26-30 April 2004, pp: 111.

Shivle, S., P. Sugavanam, H.J. Siegel, A.A. Maciejewski, T. Banka, K. Chindam, S. Dussinger, A. Kutruff, P. Penumarthy, P. Pichumani, P. Satyasekaran, D. Sendek, J. Smith, J. Sousa, J. Sridharan and J. Velazco, 2005. Mapping subtasks with multiple versions on an adhoc grid. Parallel Computing. Special Issue on Heterogeneous Computing, 31 (7): 671-690.

Shivle, S., H.J. Siegel, A.A. Maciejewski, P. Sugavanam, T. Banka, R. Castain, K. Chindam, S. Dussinger, P. Pichumani, P. Satyasekaran, W. Saylor, D. Sendek, J. Sousa, J. Sridharan and J. Velazco, 2006. Static allocation of resources to communicating subtasks in a heterogeneous ad hoc grid environment. J. Parallel Distributed Comput., 66 (4): 600-611.

Wu, M.Y., W. Shu and H. Zhnag, 2000. Segmented min-min: A static mapping algorithm for meta-tasks on heterogeneous computing systems. Proceedings of the 9th Heterogeneous Computing Workshop, pp: 375-385.

Yarmolenko, V., J. Duato, D.K. Panda and P. Sadayappan, 2000. Characterization and enhancement of static mapping heuristics for heterogeneous systems. International Conference on Parallel Processing, pp: 437-444.