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 (8): 1176-1181, 2008
ISSN 1812-5638

$QPS_{Max\text{-}Min<>Min\text{-}Min}$: A QoS Based Predictive Max-Min, Min-Min Switcher Algorithm for Job Scheduling in a Grid

[1]M. Singh and [2]P.K. Suri
[1]Department of Computer Engineering, M.M. Engineering College, Haryana, India
[2]Department of Computer Sciences and Applications, Kurukshetra University, Kurukshetra, India

Abstract: This study present a QoS based predictive Max-Min, Min-Min Switcher ($QPS_{Max\text{-}Min<>Min\text{-}Min}$) algorithm for scheduling jobs in a grid. The algorithm makes an appropriate selection among the QoS based Max-Min or QoS based Min-Min algorithm on the basis of heuristic applied, before scheduling the next job. The effect on the execution time of grid jobs due to non-dedicated property of resources has also been considered. The algorithm uses the history information about the execution of jobs to predict the performance of non-dedicated resources. Simulation demonstrates that ($QPS_{Max\text{-}Min<>Min\text{-}Min}$) outweighs the traditional QoS guided algorithms a lot in makespan.

Key words: Job scheduling, QoS, makespan, grid computing

INTRODUCTION

A computational grid is a hardware and software infrastructure that provides dependable, consistent, pervasive and inexpensive access to high-end computational capabilities. Grid computing enables the sharing and aggregation of geographically distributed resources and support wide-area distributed computing (Foster and Kesselman, 2001; Singh and Sari, 2007). As a heterogeneous computing system, the job scheduling strategy directly influences the performance of grid applications, so one of the major research objectives in grid job scheduling is to devise new and efficient methods for improving computational efficiency. Reduction in makespan is one of the fundamental objectives of optimizing job scheduling problems in distributed heterogeneous computing systems. The problem of optimally scheduling large, diverse groups of jobs onto machines of a distributed heterogeneous computing environment has been shown to be NP-complete.

In grid environment, it is desirable to compete for the best QoS provided by resources to fulfill job constraints. The scheduler in grid needs to consider jobs and QoS constraint to get a better match between jobs and resources. The algorithm, based on job requirement of QoS (Shan *et al*., 2003; Dong *et al*., 2006), classifies jobs into two categories: high QoS-requiring jobs and low-QoS requiring jobs and schedules the two separately. It enhances the adaptability of scheduling algorithm to take QoS to some extent. Although the grid is aiming at coordinated resource sharing, this sharing is often conditional: resource owner make resources available, subject to constrain on when, where and what can be done (Martinovic and Budin, 2004). The grid scheduler places the jobs submitted by grid user in a job queue and later on maps these jobs over suitable resources to make them run. A grid resource usually contains many jobs on it, including the grid jobs and the resource owner's local jobs. The local scheduler may schedule all these jobs in a parallel manner, which makes an impact on the execution time of grid jobs. So, the grid job scheduling can hardly promise the realization of its target. This study present a QoS based predictive max-min, min-min switcher ($QPS_{Max\text{-}Min<>Min\text{-}Min}$) algorithm for scheduling jobs in a grid. The proposed algorithm makes an appropriate selection among the QoS based min-min or QoS based max-min algorithm on the basis of heuristic applied, before scheduling the next job. The effect on the execution time of grid jobs due to non-dedicated resources has been reduced by using the history information about the execution of jobs to predict the performance of non-dedicated resources. Makespan is used as a bi-criterion for job scheduling with QoS consideration.

QoS BASED GRID SCHEDULING MODEL

QoS is an extensive concept. It is used differently based on its context while applying it to a resource (Daniel and Casalicchio, 2004). For instance, QoS for a network means the desirable bandwidth for the application, QoS for a computing-intensive jobs means the operation speed. We have used QoS parameter as

Corresponding Author: Manpreet Singh, Department of Computer Engineering, M.M. Engineering College, Haryana, India

network bandwidth (Plestys *et al*., 2007). A job with high QoS request can only be executed on a resource providing high quality of service (Ligang and Stephan, 2004). The traditional algorithms (Munir *et al*., 2007; (Jinnquan *et al*., 2005) may lead to a scenario where jobs with no special QoS requirement may be allotted to resources with high QoS guarantee. To overcome this drawback, modification has been made into the traditional Min-Min and Max-Min algorithms.

Scheduling model for QoS based Max-Min and Min-Min: Let $J = \{j_1, j_2,..., j_k\}$ is a set of jobs to be scheduled on $M = \{m_1, m_2,..., m_k\}$ list of machines. The job set J is further partitioned in two sets: J^h (jobs with high QoS requests) and J^l (jobs with low QoS requests). The expected execution time E_{ij} is defined as the amount of time taken by machine m_j to execute job j_i, when there is no load on m_j. Let C_{ij} represents the expected completion time of job j_i on m_j and R_j refers to the ready time of machine m_j. The makespan of the schedule is $\max_{j_i \in J}(C_{ij})$.

QoS guided Min-Min begins with the set J^h of all unassigned jobs. It has two phases: In the first phase, the set of minimum expected completion time for each job in J^h is found. In the second phase, the job with overall minimum expected completion time from J^h is chosen and assigned to the corresponding machine. Then this job is removed from J^h and process is repeated until all jobs in J^h are mapped. After finishing the mapping of all jobs with high QoS request, the algorithm repeats the same procedure for mapping the jobs with low QoS request of set J^l. The pseudo code of the algorithm follows:

Begin

/*Mapping of jobs with high QoS request on machines

While ($J^h \neq \phi$)

For $j_i \in j^h$ each **Do**

For $m_j \in M$ each **Do**

$C_{ij} = E_{ij}+R_j$

End For

Find the minimum completion time for j_i and the corresponding machine that obtains it.

End For

Find the job j_u with the minimum completion time among all jobs.

Assign job j_u to the machine m_v that gives the minimum completion time.

$J^h = j^h - \{j_u\}$ /*Delete job j_u from job set

$R_v = C_{uv}$ /* Update the ready time of machine m_v

For each $j_i \in J^h$ **Do**

$C_{iv} = E_{iv}+R_v$

End For

End While

/*Mapping of jobs with low QoS request on machines

While ($J^l \neq \phi$)

For each $j_i \in J^l$ **Do**

For each $m_j \in M$ **Do**

$C_{ij = Eij}+Rj$

End For

Find the minimum completion time for j_i and the corresponding machine that obtains it.

End For

Find the job j_u with the minimum completion time among all jobs.

Assign job j_u to the machine m_v that gives the minimum completion time.

$J^l = j^l - \{j_u\}$ /*Delete job j_u from job set

$R_v = C_{uv}$ /* Update the ready time of machine m_v

For each $j_i \in J^l$ **Do**

$C_{iv} = E_{iv}+R_v$

End For

End While

End

QoS based max-min is similar to QoS based min-min, except in phase 2. Max-min assigns job with maximum expected completion time to the corresponding machine in phase 2. An example to illustrate the execution of QoS based min-min and max-min heuristic is given below. The expected execution time of 10 jobs on 5 machines are shown in Table 1. In Table 1 X denotes that the machine does not have the capability to perform that particular job due to its low QoS provision. Figure 1 shows the results for QoS min-min with makespan equal to 27.4 while Fig. 2 shows the result for QoS max-Min with makespan equal to 28.4.

Model for QoS based max-min, min-min switcher algorithm: The idea behind min-min (Wu *et al*., 2000) is to assign the jobs to the resources that will execute them fastest and in case of max-min (Maheswaran *et al*., 1999) the idea is to overlap the long-running jobs with the

Table 1: ETC (Expected Time to Compute) matrix

Parameters	m_0	m_1	m_2	m_3	m_4
j_0	16.3	X	X	X	X
j_1	X	X	X	13.2	X
j_2	X	4.2	X	8.3	X
j_3	X	37.3	6.5	X	X
j_4	X	X	8.8	9.3	12.5
j_5	X	X	6.3	7.2	4.8
j_6	18.2	16.8	17.4	X	24.2
j_7	31.5	13.2	X	4.9	14.3
j_8	7.6	9.2	13.1	38.4	11.5
j_9	9.5	19.4	20.1	37.2	25.3

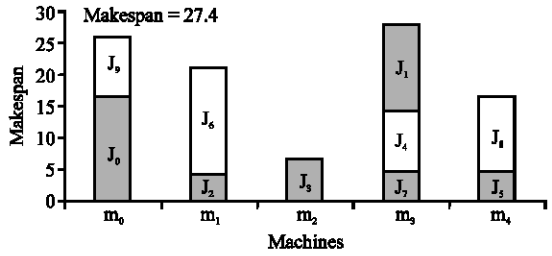

Fig. 1: Result of QoS min-min

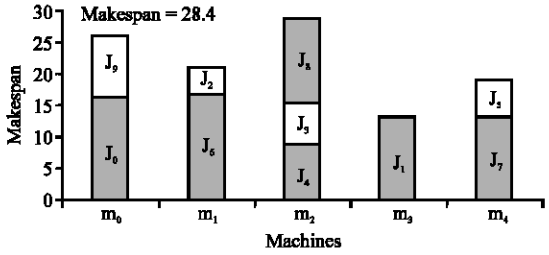

Fig. 2: Result of QoS max-min

short-running ones. For example, if there is a long job, max-min will execute many short jobs in parallel with the long job, however, min-min will execute short jobs in parallel and long job will follow the short job. The proposed scheduling algorithm selects the best algorithm between QoS max-min and QoS min-min according to the length of jobs while making each scheduling decision. Jobs with high QoS request are mapped first and after finishing the mapping of such jobs, it maps the jobs with low QoS requests. The pseudo code of $QS_{Max\text{-}Min<>Min\text{-}Min}$ (QoS based max-min, min-min switcher) algorithm follows:

Begin

While ($J^{\Psi} \neq \phi$)

/*Initially $J^{\Psi} = J^{h}$ and in the next call to this algorithm $J^{\Psi} = J^{l}$

For each $m_j \in J^{\Psi}$ **Do**

For each $m_j \in M$ **Do**

$C_{ij = Eij} + R_j$

End For

Find the minimum completion time for j_i and the corresponding machine m_j that obtains it.

End For

Calculate the Standard Deviation (SD).

/*Standard deviation of C_{ij} of all unassigned jobs of J^{Ψ} can be obtained

/* as: $\text{Avg} = \frac{\sum_{i=1}^{|J^{\Psi}|} C_{ij}}{|J^{\Psi}|}$ and $SD = \sqrt{\frac{\sum_{i=1}^{|J^{\Psi}|} (C_{ij-Avg})^2}{|J^{\Psi}|}}$

Sort all unassigned jobs of J^{Ψ} in the increasing order of their completion time and let the resulting job set is SJ^{Ψ}

Find a position pos in SJ^{Ψ} where, the difference of two consecutive C_{ij} is more than SD.

If $pos \leq \frac{|J_{\Psi}|}{2}$ **or** SD < threshold **then**

Assign first job j_f of to machine m_r that provides minimum completion time C_{fr} for j_f among all machines.

$R_r = C_{fr}$ /*Update the ready time of machine m_r

For each $j_i \in J^{\Psi}$ **Do**

$C_{ir} = E_{ir} + Rr$

End For

$j^{\Psi} = j^{\Psi} - \{j_f\}$ /*Delete job j_f from job set

Else

Assign $j_{|J^{\Psi}|}$ to machine m_r that provides minimum completion time $C_{|J^{\Psi}|r}$ for $j_{|J^{\Psi}|}$ among all machines.

$R_r = C_{|J^{\Psi}|r}$ /*Update the ready time of machine m_r

For each $j_i \in J^{\Psi}$ **Do**

$C_{ir} = E_{ir} + Rr$

End For

$j^{\Psi} = j^{\Psi} - \{j_{|J^{\Psi}|}\}$ /*delete job from job set

End If

End While

End

The algorithm selects between the two conventional algorithms, max-min and min-min, whenever one acts better than the other on the basis of SD of minimum completion time of unassigned jobs. A search is made to find a position in sorted list (on the basis of completion time) where, the difference between the completion times of two successive jobs is more than SD. It is a place where a big increase in length of jobs had occurred. If this position is in first half of list, it shows that the number of long jobs is more than the number of short jobs i.e., a situation where min-min outperforms max-min, so min-min is selected by mapping the first job of sorted list on corresponding machine. If this position is in second half, it means the number of short jobs is more than the number of long jobs i.e., a situation where max-min outperforms min-min, so max-min is selected by assigning the last job of sorted list. If this position does not exist then SD is compared with a threshold value. If SD is less than threshold, it means length of jobs are in small range, so min-min is selected. Otherwise, Max-Min is selected for assigning the next job.

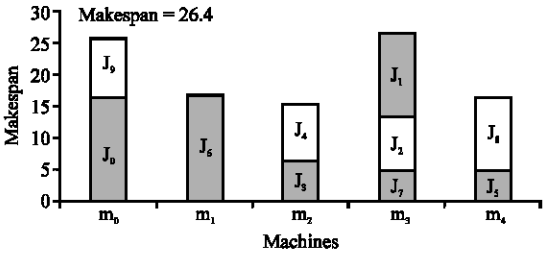

Fig. 3: Result of $QS_{Max\text{-}Min<>Min\text{-}Min}$

The result of $QS_{Max\text{-}Min<>Min\text{-}Min}$ algorithm corresponding to Table 1 is shown in Fig. 3 and the makespan is 26.4.

APPLYING PREDICTIVE SCHEDULING MECHANISM TO $QS_{Max\text{-}Min<>Min\text{-}Min}$

A grid resource usually contains many jobs on it, including the grid jobs and the resource owner's local jobs. The local scheduler may schedule all these jobs in a parallel manner which makes an impact on the execution time of grid jobs. It is unpractical to expect the machines in a grid to be dedicated. Due to the simultaneous execution of local jobs, the actual execution time of grid jobs usually increases. If the scheduler does not know this change, it will persist on the former schedule, which leads to an increase in makespan. In continuation with the earlier example, let Table 2 shows the actual execution time of 10 jobs on the same 5 machines. Figure 4 shows the scenario of $QS_{Max\text{-}Min<>Min\text{-}Min}$ scheduling, where scheduler persists on former schedule and the makespan is 33.5.

Although, such change in the execution time of jobs is unavoidable due to site autonomy (Akl and Dong, 2006), if the scheduling strategy can predict in advance, a better schedule will be made. So, $QS_{Max\text{-}Min<>Min\text{-}Min}$ algorithm is modified by adding a predictive scheduling mechanism to it, in which every expected execution time is treated as a random variable rather than a predetermined constant. By estimating the value of each random variable, the scheduler can make a better schedule, which takes into account the actual status of resources (Gong *et al*., 2002). The random variable PE_{ij} is predicted from past observations. The relation between PE_{ij} and E_{ij} can be expressed as:

$$PE_{ij} = E_{ij} \times \theta_{ij}$$

where, θ_{ij} is the additional amount of time needed by machine m_j to finish job j_i caused by the execution of local jobs and other grid jobs. Let $\theta_{ij} = \frac{PE_{ij}}{E_{ij}}, (\theta > 1)$. Suppose before job j_i is assigned to machine m_j, it has already executed n jobs. θ_{ij} can be predicted as:

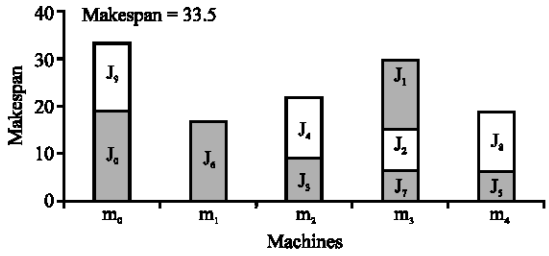

Fig. 4: Result of $QS_{Max\text{-}Min<>Min\text{-}Min}$ on non-dedicated machine with same schedule

Table 2: Actual execution time matrix

Parameters	m_0	m_1	m_2	m_3	m_4
j_0	19	X	X	X	X
j_1	X	X	X	14.6	X
j_2	X	6.0	X	8.3	X
j_3	X	38.9	9.0	X	X
j_4	X	X	12.8	9.3	12.5
j_5	X	X	14.2	7.2	6.3
j_6	18.2	16.8	19.0	X	24.2
j_7	38.5	14.4	X	6.8	16.5
j_8	12.2	9.2	16.8	39.6	12.5
j_9	14.5	22.4	26.4	38.2	25.3

$$\theta_{ij} = \sum_{u=1}^{n} X_u \theta_{uj}, \quad \sum_{u=1}^{n} X_u = 1$$

where, X_u is the weight of θ_{uj}.

After estimating θ_{ij}, we can obtain Pe_{ij} and PR_j as:

$$PE_{ij} = E_{ij} \times \theta_{ij}$$
$$PR_j = \min_{1 \le i \le t} PC_{ij}$$

where, t stands for the No. of grid jobs that machine m_j allow running simultaneously. $QS_{Max\text{-}Min<>Min\text{-}Min}$ algorithm is modified after substituting the predicted values of C_{ij}, E_{ij} and R_j as PC_{ij}, PE_{ij} and PR_j to produce QoS based predictive max-min, min-min switcher ($QPS_{Max\text{-}Min<>Min\text{-}Min}$). QoS based max-min and QoS based min-min algorithms are modified in the same way to produce $QP_{Max\text{-}Min}$ and $QP_{Min\text{-}Min}$ algorithms.

SIMULATION AND RESULTS

A simulation program is implemented in java to evaluate the performance of $QPS_{Max\text{-}Min<>Min\text{-}Min}$, running on a Pentium 4 1.73 GHz laptop In order to compare the performance of $QPS_{Max\text{-}Min<>Min\text{-}Min}$ with $QS_{Max\text{-}Min<>Min\text{-}Min}$ algorithm, the predicted value of 10 jobs on the same 5 machines is generated and recorded in Table 3 for the previous problem. Figure 5 shows the scenario of

Table 3: Predicted execution time matrix

Parameters	m_0	m_1	m_2	m_3	m_4
j_0	18.9	X	X	X	X
j_1	X	X	X	14.20	X
j_2	X	4.60	X	8.46	X
j_3	X	40.66	8.45	X	X
j_4	X	X	11.44	10.04	12.75
j_5	X	X	8.19	7.78	4.89
j_6	16.82	18.31	22.62	X	24.68
j_7	36.54	14.40	X	5.29	14.59
j_8	8.82	10.02	17.03	39.16	11.73
j_9	11.02	21.15	26.13	39.40	25.81

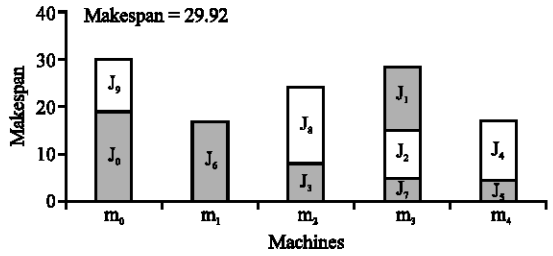

Fig. 5: Result of $QPS_{Max\text{-}Min<>Min\text{-}Min}$

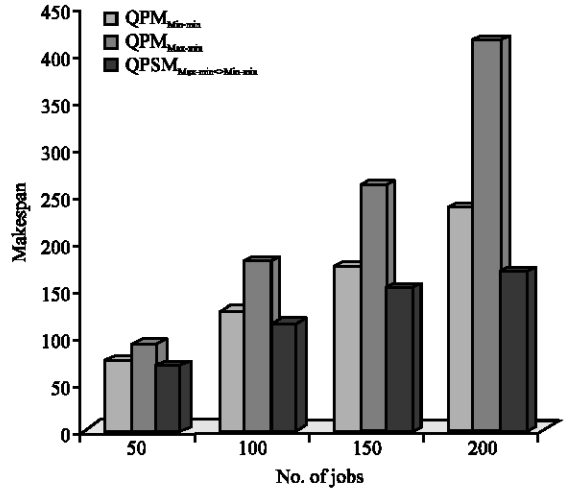

Fig. 6: Comparison of three heuristics in makespan. The $QPS_{Max\text{-}Min<>Min\text{-}Min}$ gives a shortest makespan

$QPS_{Max\text{-}Min<>Min\text{-}Min}$ scheduling, where scheduler has already predicted the change in completion time of jobs leading to a new schedule and the makespan is 29.2.

The performance of $QPS_{Max\text{-}Min<>Min\text{-}Min}$ is also compared with QoS guided Min-Min and QoS based Max-Min (generating schedule on non-dedicated machines with prediction). The PTC (Predicted Time to Compute) matrices containing the predicted execution time of 50 ($|j^h| = 30, |J^l| = 20$), 100 ($|J^h| = 75, |J^l| = 25$), 150 ($|J^h| = 110, |J^l| = 40$), 200 ($J^h|| = 120, |J^l| = 80$) jobs on 10 machines are generated to make a comparison among these heuristic algorithms. The expected execution time of jobs varies from 1 to 50. The heuristic with shortest makespan is declared the best heuristic to perform job scheduling in grid. As shown in Fig. 6, the $QPS_{Max\text{-}Min<>Min\text{-}Min}$ out performs the conventional scheduling algorithms. In conclusion, the proposed algorithm by exploiting the merits of QoS based min-min and max-min along with prediction scheduling mechanism leads to significant performance gain for a variety of jobs having different QoS requirements.

CONCLUSION

Job scheduling is one of the important problem to be solved in grid computing. In this study, a novel scheduling algorithm is presented which merges the efficiency of max-min along with min-min and also considers both QoS and non-dedicated property of grid resources. A simulation system was developed to test the proposed algorithm and the simulation results show that $QPS_{Max\text{-}Min<>Min\text{-}Min}$ algorithm outperforms the traditional heuristics. The future work will focus on job scheduling with multi-dimensional QoS and extend the proposed scheduling algorithm in a real grid.

REFERENCES

Akl, G.S. and F. Dong, 2006. Scheduling algorithms for grid computing: State of the art and open problems. http://www.cs.queensu.ca/home/akl/techreports/GridCompuitng.pdf.

Daniel, A.M. and E. Casalicchio, 2004. QoS in grid computing. IEEE Internet Comput., 8: 85-87.

Dong, F., J. Luo, L. Gao and L. Ge, 2006. A grid task scheduling algorithm based on qos priority grouping. Proceeding of the 5th International Conference on Grid and Cooperative Computing (GCC), October (GCC 2006), Hunan, China, pp: 58-61.

Foster, I. and C. Kesselman, 2001. The anatomy of the grid: Enabling scalable virtual organizations. Int. J. High Performance Comput. Appl., 15: 200-222.

Gong, L., S. Xian-He and E.F. Watson, 2002. Performance modeling and prediction of nondedicated network computing. IEEE Trans. Comput., 51: 1041-1055.

Jinnquan, Z., N. Lina and J. Changjun, 2005. A heuristic scheduling strategy for independent tasks on grid. Proceeding of the 8th International Conference on High-Performance Computing in Asia-Pacific Region (HPCASIA), Nov. (HPCASIA), Beijing, China, pp: 5-6.

Ligang, H. and A. Stephen, 2004. Dynamic scheduling of parallel jobs with QoS demands in multiclusters and grids. Proceeding of 5th IEEE/ACM International Workshop on Grid Computing, Nov. 2004, Pittsburgh, USA., pp: 402-409.

Maheswaran, M., S. Ali, H.J. Seigel, D. Hensgen and R. Freund, 1999. Dynamic mapping of a class of independent tasks onto heterogeneous computing systems. J. Parallel Dist. Comput., 59: 107-131.

Martinovic, G. and L. Budin, 2004. Human in computational grid establishment. Proceeding of IEEE International Conference on Systems, Man and Cybernetics (SMC), Oct. (SMC), Hague, Netherlands, pp: 88-93.

Munir, E.U., L. Jian-Zhong, S. Sheng-Fei and Q. Rasool, 2007. Performance analysis of task scheduling heuristics in grid. Proceeding of the 6th International Conference on Machine Learning and Cybernetics, Aug. 2007, Hong-Kong, pp: 3093-3098.

Plestys, R., G. Vilutis and D. Sandonavicius, 2007. The measurement of grid qos parameters. Proceeding of 29th International Conference on Information Technology Interface, June 2007, Cavtat, Croatia, pp: 703-707.

Shan, X.H., S. Xian-He and G.V. Laszewski, 2003. QoS guided min-min heuristic for grid task scheduling. JCST., 18: 442-451.

Singh, M. and P.K. Suri, 2007. Analysis of service, challenges and performance of a grid. Int. J. Comput. Sci. Nnetwork Security, 7: 84-88.

Wu, M., W. Shu and H. Zhang, 2000. Segmented min-min: A static mapping algorithm for meta-tasks on heterogeneous computing system. Proceeding of the 9th Heterogeneous Workshop (HCW), May (HCW), Ancun, Mexico, pp: 375-385.