International Journal of Innovative Research in Advanced Engineering (IJIRAE) ISSN: 2349-2763
Issue 01, Volume 3 (January 2016) www.ijirae.com

Improving the Performance of Mapping based on Availability-Alert Algorithm Using Poisson Arrival for Heterogeneous Systems in Multicore Systems

*Sheshappa S.N, ** Dr. G. Appa Rao, ***Dr. K V Ramakrishnan
*Associate Professor, Dept. of Information Science & Engineering, Sir MVIT, Bangalore,
**Professor, Dept. of Computer Science & Engineering, GITAM, Vishakapatnam,
***Professor, Dept. of Electronics & Communications Engineering, CMRIT, Bangalore,

Abstract- Performance of Mapping can be improved and it is needed arise in several fields of science and engineering. They'll be parallelized in master-worker fashion and relevant programming ways have been projected to cut back applications. In existing system, the performance of application is considered only for homogenous systems due to simplicity. In this we use Availability-Alert algorithm using Poisson arrival to extend our approach for Heterogeneous systems in Multi core Architecture systems. Our proposed algorithm also considers the requirement needed for the application for their execution in Heterogeneous systems in Multi core Architecture systems while maintaining good performance. Performance prediction errors are minimized by using this approach at the end of the execution. We present simulation results to quantify the benefits of our approach.

Index Terms – Availability- Alert algorithm, Poisson arrival, performance Prediction errors, Performance Mapping.

1. INTRODUCTION

Over the last decade, Heterogeneous systems in Multi core Architecture systems are widely used for scientific and business applications. Applications that comprise several freelance procedure tasks arise in several domains and square measure similar temperament to master-worker execution on cluster platforms. In this paper we tend to address the matter of planning these applications with the goal of reducing execution time, or make span. This downside has been studied for 2 totally different scenarios: fixed-sized tasks and Separable work. In the former situation, the application's work consists of tasks whose size (i.e. quantity of needed computation) square measure pre-determined and variety of economical programming ways have been planned. In this work we tend to focus on the latter state of affairs, within which the computer hardware will partition the work in discretional, continuous "chunks" (in sensible situations, this usually implies that the appliance consists of many similar machine tasks). Examples of such applications area unit: feature extraction, in which a giant image is metameric, and every section is transferred to a employee and processed locally; Signal process, which tries to recover a symptom buried in a very giant file recording measurements; and sequence matching, [2], in which a single sequence is compared to a giant lexicon file, and the running time is proportional to the letters in this lexicon. The objective of mapping algorithms is to map tasks onto nodes and order their execution in an exceedingly thanks to optimize overall performance. In mapping theory, the fundamental assumption is that each one machines are always available for process. This assumption may well be affordable in some cases; however it's not valid in eventualities wherever there exist sure maintenance requirements, breakdowns, or different constraints, which make the machines unavailable for process. Examples of such constraints are often found in several application areas. As an example, machine nodes in Heterogeneous systems in Multi core Architecture systems got to be sporadically maintained to stop malfunctions. We aim to develop a mapping strategy which is used to enhance the availability of resources in Heterogeneous systems in Multi core Architecture systems while maintaining good performance. In our previous work, we tend to studied security-aware mapping for embedded systems, clusters and Grids. However, these planning algorithms area unit designed for homogenized systems. Further, our previous mapping algorithms will not seems to be appropriate for multiclass tasks with availableness necessities. The main question in mapping Separable load is how to select an optimal division of the load into chunks. One potential approach is to divide the load in as several chunks as processors and to dispatch work in a single round of allocation. This has several drawbacks, particularly poor overlap of communication and computation and poor hardiness to performance prediction errors. Consequently, variety of researchers has investigated multi-round algorithms. Main observations include dividing the workload include larger chunks reduces overhead. The use of smaller chunks reduces performance prediction errors. THMR (Tough Homogenous Multi-Round) borrows from HMR to achieve high performance and hardiness to prediction errors by increasing and then decreasing chunk sizes throughout execution.

2. RELATED WORK

A number of multi-round mapping algorithms for separable loads are projected with the belief that performance predictions are absolutely correct.

Most of this work assumes that the quantity of knowledge to be sent for a chunk is proportional to the chunk size. The work in [16] presents a "multi-installment" algorithm that uses increasing chunk sizes throughout application execution to attenuate make span. Though this approach provides associate degree optimum schedule for a given range of rounds, it's the subsequent limitations: latencies related to resource utilization are not modeled; and there's no thanks to verify the optimum range of rounds. Our recent add [19, 20] addresses each these limitations. By imposing the restriction that equal sized chunks are sent to employees inside a round, the HMR formula makes it attainable to reason associate degree optimal range of rounds whereas modeling resource latencies. In this work we have a tendency to extend HMR to account for performance prediction errors. Many works aim at maximizing the steady-state performance of terribly long-running applications [4, 11]. The goal isn't to attenuate application make span however to get asymptotically optimum schedules. Note that in these works it's attainable to adapt to unsteady performance characteristics of the underlying resources as the optimum schedule is periodic and might so be modified from one amount to consecutive. Multi-round programming for separable loads has conjointly been studied presumptuous non-zero performance prediction errors. The algorithms in [13, 14] begin application execution with massive chunks and reduce chunk sizes throughout. Assuming uncertainties on task execution times, this ensures that the last chunks won't expertise massive entirely insecurity. These works assume a set network overhead to dispatch chunks of any sizes. Against this, we have a tendency to assume that the amount of knowledge to be sent for a piece is proportional to the chunk size, that is additional realistic for many applications. With this assumption, beginning by causation an oversized chunk to the first employee would cause all the remaining employees to be idle throughout that doubtless long information transfer. However, in this paper we have a tendency to use the basic ideas in [13] to increase our previous work on the HMR formula. The notion of programming applications by combining a performance-oriented and a robustness-oriented approach is not new and has been explored as an example in [9], which uses each static programming and self-mapping. Our approach is additional performance economical as a result of it achieves better overlap of computation and communication and leverage the add [13] for improved strength to uncertainty.

3. BACKGROUND

3.1 PLATFORM AND APPLICATION ORIENTATION

We take into account applications that include a unceasingly mapping work, W_{total} , and that we assume that the number of application knowledge required for process a piece is proportional to the number of computation for that chunk. As done in most previous work, we tend to solely take into account transfer of application input data. The works in [1, 5] take into consideration output knowledge transfers however use one. Overhead incurred by the master to initiate information spherical of labor allocation. Similarly, the add [4] models output however considers only steady-state performance. We assume a master-worker model with N workers processes running on N processors. We tend to assume that the master does not send chunks to staff at the same time, although some pipelining of communication will occur [1]. Although this is a standard assumption in most previous work, it may well be helpful to permit for coincident transfers for better output in some cases (e.g. WANs). We've got provided an initial investigation of this issue in [20] and leave a lot of complete study for future work. The effective platform topology will then be viewed as heterogeneous processors connected to a master by heterogeneous network links. Finally, we tend to assume that staff will receive data from the network and perform computation at the same time (as for the "with front-end" model in [17]).

Computation may be overlapped with communication. We tend to model the time spent for the master to send chunk units of employment to worker *i*, as: transfer to worker *i* (i.e. initiate a TCP connection). When the master finishes pushing information on the network to worker and also the time once employee receives the last computer memory unit of data. We tend to assume that this model was mentioned well in [19, 20]. The key point is that it's versatile and may be instantiated to model platforms, portion of the transfer isn't overlap able with different information transfer. Supported our expertise with actual package [7], we tend to found that the machine latency, is prime for realistic modeling. We know of only 1 work that models this latency within the context of cleavable load planning [3]. Note that for cases that the required information files area unit replicated or pre-staged on workers, we will model these cases by using an appropriately large or infinitely giant. Relevant previous works on cleavable load planning [1, 10, 16].

3.2 THE HOMOGENOUS MULTI-ROUND

In this section we offer a short outline of the work and ends up in [19, 20] to line the stage for the THMR formula, which we tend to conferred in Section four. Figure 3 shows however HMR dispatches chunks of loads in multiple rounds. Whereas this is often similar in spirit to the "multi-installment" algorithm [16], HMR keeps chunk sizes fixed within each round. The chunk size is exaggerated between rounds so as to scale back the overhead of beginning communication and computation. While our work in [19, 20] addresses heterogeneous platforms, but we only discuss the same case here for simplicity.

The unknowns that HMR should verify square measure one, the quantity of rounds, and the chunk size used at each round. Our development of HMR was as follows.

We have a tendency to initial obtained a simple induction relation on the chunk sizes. Then, we have a tendency to frame the mapping drawback as a constrained improvement problem: the goal being to reduce the application execution time subject to the constraint that all the chunks total up to the entire employment. Using the Lagrange multiplier factor technique [8] we have a tendency to obtain a system of 2 equations as unknown. This method can be resolved numerically by division (requiring regarding 0.07 seconds on a 400MHz PIII). Complete details square measure provided in [17]. Our main contribution is that we have a tendency to be ready to reckon associate approximately best range of rounds whereas employing a realistic platform model incorporating resource latencies. To evaluate the effectiveness of our approach we have a tendency to used simulation and compared HMR with the multi-round rule in [16] and also the one-round rule in [1] for an intensive space of platform configurations. We initiate that:

1. HMR results in higher schedules than its competitors in an awesome majority of the cases in our experiments (95%);
2. Once HMR is outperformed, it's terribly near the competitors (on average among 2.04% with a regular deviation of zero.035);
3. Neither competition ever outperforms HMR “across the panel” (that means ranges of computation/communication ratios).

HMR is in a position to realize such improvement over previous work in spite of the uniform round restriction, because this restriction makes it doable to compute a best number of rounds.

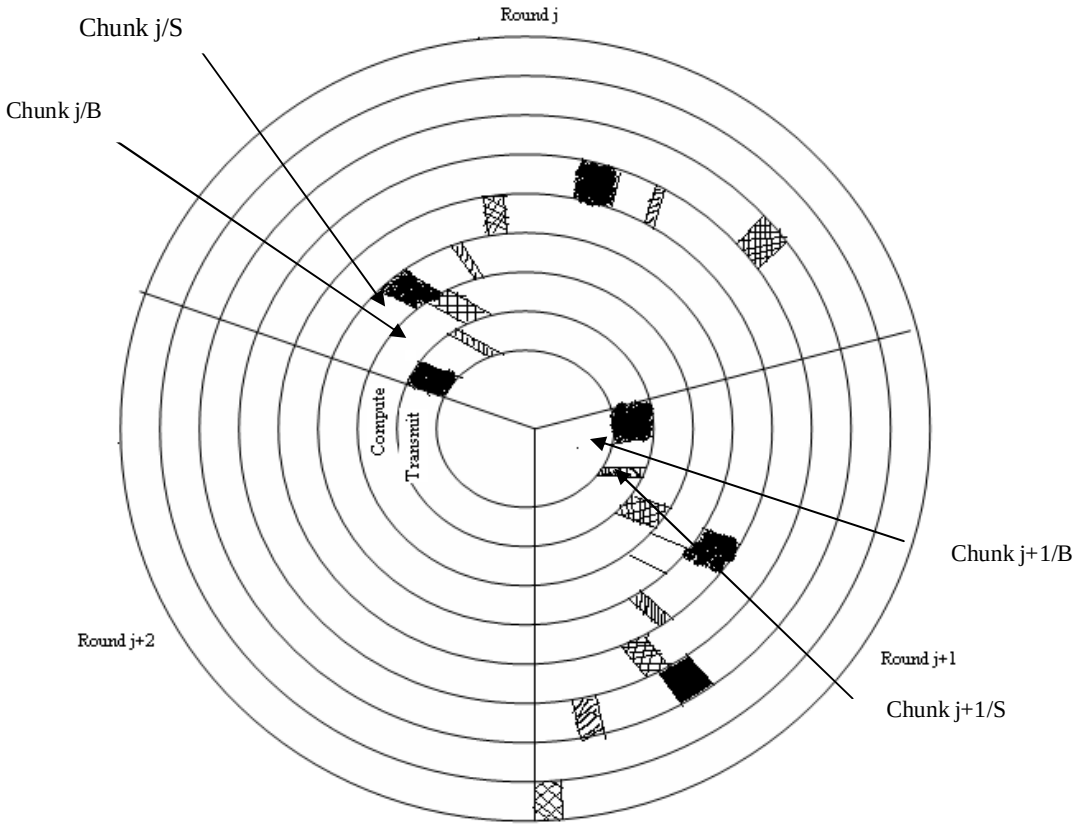
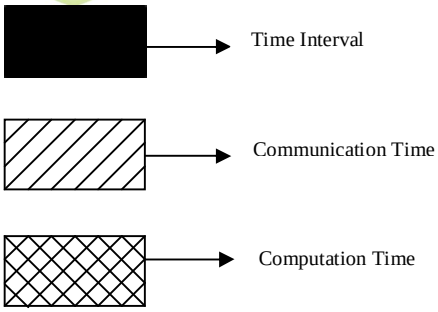


Figure 1: EHMM dispatches the mapping into chunks in each round



This is often one in every of the most results of our previous work. We tend to additionally showed that HMR tolerates high platform heterogeneity as a result of a good resource choice.

4. EFFECTIVE HOMOGENOUS MULTI-MAPPING(EHMM)

The length of a computation or of a knowledge transfer usually cannot be foretold dead accurately in observe. Prediction errors arise owing to uncertainties of each the platform and the application. On a non-dedicated platform it's virtually application the time taken to trace through one pixel depends greatly on the quality of the scene. As a result, associate degree approach within which the complete schedule is pre calculated at the onset of the applying [16, 19] is probably going to be inefficient. All the same, this can be a regular approach in the planning literature, and specially for developing Separable employment algorithms that use increasing chunk sizes [16], together with our own work on Effective Homogenous Multi-Mapping (EHMM). On the opposite extreme, algorithms notably targeted at tolerating prediction errors don't create use of performance predictions in the least [13, 14]. They exponentially decrease chunk sizes and schedule chunks in an exceedingly greedy fashion. One issue there's the overhead for programming little chunks that is self-addressed in [14]. Additional significantly, these algorithms don't bring home the good overlap of communication and computation, which is crucial for top performance.

Our basic approach is to mix each approaches: EHMM schedules the employment in 2 consecutive phases: Phase #1 uses a Good Book of EHMM to pre-calculate the initial portion of the schedule, first use little chunk sizes and step by step increasing chunk sizes; phase #2 uses the factoring approach in [13] to decrease chunk sizes. Phase #1 aims for top performance via economical communication computation overlap and overhead reduction, whereas Phase #2 limits the negative impact of performance prediction errors at the top of execution. In what follows we have a tendency to describe a model for these errors and our key style decisions for EHMM.

4.1 ENHANCEMENT OF PERFORMANCE FORECAST FAULT MODEL

We assume an easy prediction error model each for information transfers and computations: the magnitude relation of expected execution time to effective execution time is often distributed with mean 1 and variance *error* (the distribution is truncated to avoid negative values). This model is sort of general and was employed in the relevant previous literature [13, 14]. Its simplicity makes it easy to interpret simulation results. A number of our intuitions for developing ETHMR are supported the belief of commonly distributed errors (as it absolutely was tired [13, 14]). We tend to conjointly assume that the likelihood distribution of prediction errors is stationary throughout the application run. If it's not stationary however doesn't change too quickly, our approach ought to still be effective as phase #2 doesn't use prediction errors the least bit. We also ran all the experiments beneath a uniformly distributed error model, however our results were basically similar.

A key question is whether or not *error* is a known amount, i.e. whether or not THMR will use its worth to make your mind up on however to organize the schedule at the onset of the applying. Estimations of *error* can be obtained by past expertise with the applying and therefore the platform, by querying resource monitoring or forecasting services [12], by watching prediction errors because the application runs, or by any combination of these. In what follows we tend to discuss alternate ways whether *error* is known or unknown.

5. AVAILABILITY ALERT ALGORITHM USING POISSON ARRIVAL

We currently present an Availability Alert algorithm that is used in a judicious way to improve the provision of Heterogeneous systems in Multi core Architecture systems whereas maintaining sensible performance in response time. This algorithm creates a set N_a of nodes, finds the expected finishing time of the entire node in the set and also calculates the availability level of each node. Then allocate the job to the node that has the least finishing time. This algorithm is implemented by using Poisson arrival.

1. *t=0, r=0 rate λ up to time T;*
2. *Generate work W_j;*
3. *t=t+[-(1/λ) ln (work W_j)]. If t>T, then stop;*
4. *set r=r_1 and set r=t;*
5. *Place the work W_j in the queue in ascending order*
6. *Create a set of node N_a;*
7. *Label the node N_a*
8. *Assign the availability cost and response time to node N_a, CA← ∞, RT← ∞*
9. *if (N_a ≠empty) then*
10. *for each node b belongs to N_a, do*
11. *calculate the expected finish time of the work W_j*
12. *if the response time of the node j is less than the assigned response time, i.e RT_j <RT, then*
13. *RT=RT_j ; x←j;*
14. *end for*
15. *else*
16. *for each node b in the system do*
17. *calculate the availability cost of work W_j on node b, CA_j*
18. *if the availability cost of the work on node b is less than assigned availability cost, i.e CA_j <CA then*
19. *CA=CA_j ; RT=RT_j ; x←j ;*
20. *end for*
21. *end if*
22. *WL_{min}=N_1 ; LI_{min} =∞; /* Assume that node 1 is lightly loaded and its load capacity is ∞*/*
23. *for each node b belongs to N_a do*
24. *calculate its work load LI_b ;*
25. *if the load of the node b is less than minimum load index, i.e LI_b <LI_{min} then*
26. *set the load index of b as the minimum load index LI_b and node b is the lightly loaded node*
27. *Allocate work W_j to node b*
28. *else*
29. *Allocate work W_j to node WL_{min}*
30. *end if*
31. *end for*

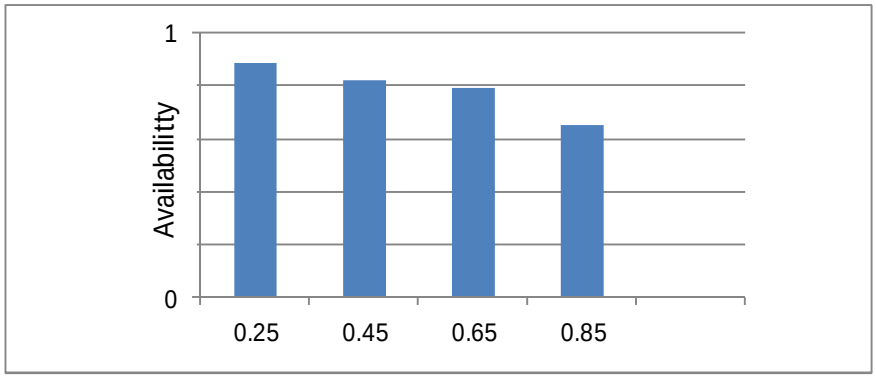
6. SIMULATION RESULTS

The simulation results have been obtained by using the GridSim. Simulation appears to be one of the feasible ways to examine the algorithms on large scale distributed systems containing heterogeneous assets. Compared to using the real systems in nature, replication works good without making the analysis mechanism difficult. Simulation is additionally effective in operating with terribly massive hypothetic issues that may otherwise need involvement of an outsized variety of active users and resources, that is extremely onerous to coordinate and build at large-scale analysis setting for investigation purpose. The modeling and simulation of entities by using GridSim toolkit can took part in parallel and distributed computing. To design and evaluate the mapping algorithms resources, applications are used. It has a wide facility for creating classes categorized under heterogeneous resources that can be aggregated by using resource brokers. To solve data rigorous applications resources are used which can be a single processor or multi-processor, with or without shared or distributed memory. Appropriate scheduler can be used for managing which are based on instant or space.

In this section we tend to provide experimental results obtained in simulation with the goals of quantifying the impact and effectiveness of our design selections. This is for many reasons: the results are additional straightforward to know and compare; a number of the competing algorithms don't seem to be amenable to heterogeneous platforms; and also the purpose of our analysis is primarily to know the impact of performance prediction errors. Our analysis is also used to eliminate the negative impact of performance prediction errors at the end of execution.

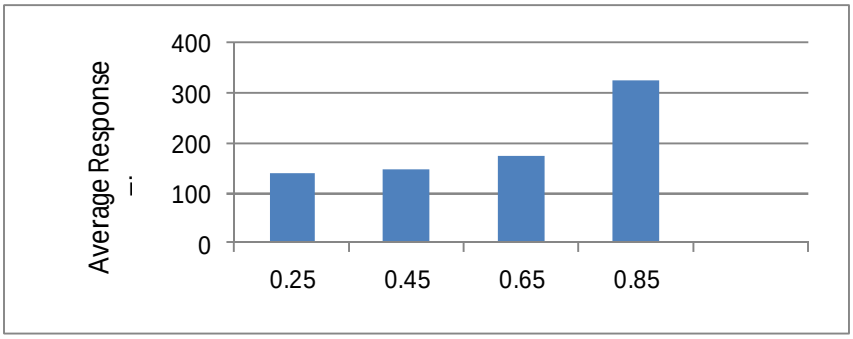
7. CONCLUSION AND FUTURE WORK

In this paper we've bestowed THMR (Tough Homogenous Multi-Round), a mapping algorithm for minimizing the makespan of dividable work applications underneath uncertainties of resource performance; our final goal is to develop a mapping strategy for dividable loads that may be employed in observe for real-world applications on real-world platforms. In our previous work [19, 20] we have a tendency to create a primary contribution by developing EHMM, a rule that outperforms antecedently planned algorithms while tolerating additional realistic latency models.



Arrival Rate - (Fig. 2)

Fig. 2 shows the arrival rate of the job and the resource availability for mapping the job.



Arrival Rate - (Fig. 3)

Fig. 3 shows the arrival mapping rate of the job and the average mapping finishing time for the job.

In this Research work, we have taken subsequent step and self-addressed the problem of performance prediction errors that arise thanks to uncertainties about platforms and applications. EHMM leverages EHMM attain each high performance and robustness to prediction errors: it uses two consecutive phases for application execution, with increasing and decreasing work chunk sizes. We have evaluated our approach with in depth simulation experiments. We've got that EHMM outperforms previously proposed algorithms each in terms of performance and robustness. We have implemented Availability-Alert Algorithm using Poisson arrival for providing the resources at the right time for carrying out processing in Heterogeneous systems in Multi core Architecture systems while maintaining good performance in response time. Performance prediction errors can be minimized by using this approach. Future work can be carried out by providing priority for the application based on the mapping available in each application.

8. REFERENCES

[1] I.Foster, C. Kesselman, and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," Int'l Journal Super computer Applications, vol. 15, no. 3, pp. 200-222, Aug. 2001.

[2] R. Ranjan, A. Harwood, and R. Buyya, "A Study on Peer-to-Peer Based Discovery of Grid Resource Information," IEEE Comm. Surveys and Tutorials, vol. 10, no. 2, pp. 1-42, Apr.-June 2008.

[3] D. Abramson, J. Giddy, and L. Kotler, "High Performance Parametric Modeling with Nimrod/G: Killer Application for the Global Grid?," Proc. 14th Int'l Parallel and Distributed Processing (IPDPS '00), pp. 520-528, May 2000.

[4] D.P. da Silva, W. Cirne, and F.V. Brasileiro, "Trading Cycles for Information: Using Replication to Schedule Bag-of-tasks Applications on Computational Grids," Proc. Int'l Conf. Parallel and Distributed Computing, Euro-Par '03, pp. 169-180, Aug. 2003.

[5] I. Chlamtac and A. Farago, "Making Transmission Schedules Immune to Topology Changes in Multi-Hop Packet Radio Networks," IEEE/ACM Trans. Networking, vol. 2, no. 1, pp. 23-29, Feb. 1994. 1630 IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 22, NO. 10, OCTOBER 2011

[6] Altilar and Y. Paker. Optimal Mapping algorithms for Communication Constrained Parallel Processing. In *Proceedings of Europar'02*, pages 197 206, 2002.

[7] Altilar and Y. Paker. An Optimal Mapping Algorithm for Parallel Video Processing. In *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, 1998.

[8] H. Casanova and F. Berman. *Parameter Sweeps on the Grid with APST*, chapter 26. Wiley Publisher, Inc., 2002. F. Berman, G. Fox, and T. Hey, editors.

[9] L. Rosenberg. Sharing Partitionable Loads in Heterogeneous NOWs: Greedier Is Not Better. In *Proceedings of the 3rd IEEE International Conference on Cluster Computing (Cluster 2001)*, pages 124–131, 2001.

[10] O. Beaumont, A. Legrand, and Y. Robert. The Master- Slave Paradigm with Heterogeneous Processors. In *Proceedings of Cluster'2001*, pages 419–426. IEEE Press, 2001.

[11] O. Beaumont, A. Legrand, and Y. Robert. Optimal algorithms for mapping Separable loads on Heterogeneous systems in Multi core Architecture systems. Technical Report RR-2002-36, Laboratoire de l'Informatique du Parall´elism, ´ Ecole Normale Sup´erieure de Lyon, France, October 2002.

[12] T. Hagerup. Allocating Independent Tasks to Parallel Processors: An Experimental Study. *Journal of Parallel and Distributed Computing*, 47:185–197, 1997.

[13] T. Xie and X. Qin, "A Security-Oriented Task Scheduler for Heterogeneous Distributed Systems," Proc. 13th IEEE Int'l Conf. High Performance Computing (HiPC '06), Dec. 2006.

[14] V. Bharadwaj, D. Ghose, V. Mani, and T. G. Robertazzi. *Mapping Divisible Loads in Parallel and Distributed Systems*, chapter 10. IEEE Computer Society Press, 1996.

[15] V. Bharadwaj, D. Ghose, V. Mani, and T. G. Robertazzi. *Mapping Divisible Loads in Parallel and Distributed Systems*. IEEE Computer Society Press, 1996.

[16] Xiao Qin, Tao Xie, " An Availability- Aware Task Mapping Strategy for Heterogeneous systems in Multi core Architecture systems", *IEEE Transactions on computers*, vol. 57,no. 2,pp. 188-198, Feb. 2008.