DOI: http://dx.doi.org/10.26483/ijarcs.v9i1.5265

ISSN No. 0976-5697

Volume 9, No. 1, January-February 2018

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

STATIC AND DYNAMIC RESOURCE ALLOCATION STRATEGIES IN HIGH PERFORMANCE HETEROGENEOUS COMPUTING APPLICATION

Ms. Ragini Karwayun
Research Scholar: Mewar University
Rajasthan, India

Dr. K. P. Yadav
Director: IIMT
Greater Noida, India

Dr. H. S. Sharma
Pro Vice Chancellor: Mewar University
Rajasthan, India

Abstract: More and more computing services are running in clouds as the number and scope of internet services are increasing exponentially. Major objective of cloud computing is to give users virtually unlimited pay per use computing resources without any concern for managing the underlying infrastructure. But this results in huge increase in size of computing environment which makes it very difficult to measure the performance of allocation strategies that use the description of underlying infrastructure and resource dependency graphs for making decisions. Both dynamic and static allocation strategies have their share of advantages and drawbacks. In this paper we will try to define a hybrid scheme for resource allocation that will use the positive features of both schemes to give better performance.

Keywords: Scheduling, Resource Allocation, Dynamic Scheduling strategies;

I. INTRODUCTION

Predicting the performance of allocation and scheduling algorithms is a difficult task. Static resource allocation and scheduling schemes demand the resource requirements and platform description beforehand thus resulting in unpredictable time difference between the start and finish of sequence of jobs. On the other hand, dynamic schemes make decisions based on the available information (resource requirement and platform specifications) during run time thus making them too myopic.

Different parts of a very large computing application have different computation requirements. It is nor practically feasible to allocate the entire application to a single machine. With the advent of high speed communication infrastructure, distributed high performance machines can be connected together to provide heterogeneous computing environment.

Static algorithms use the resource dependency and description of the platform to make scheduling and allocation decisions. The detail description includes the execution time required by all types of jobs on all types of resources, the time required for communication between any two resources and the congestion affecting the communication. The scheduling decision is made prior to the actual execution of the application.

Dynamic algorithms make dynamic decisions for scheduling and resource allocation at run time. These decisions are based on the information about the platform such as the list of available resources, set of available jobs and the location of the data. Dynamic schemes can be classified into two categories: Task driven or resource driven. Task driven schemes make allocation decision as soon as a task becomes ready whereas resource driven schemes initiate the allocation decision as soon as a resource becomes free.

Both schemes require complex computations to map the communication and computing time to decide the priorities of the requesting jobs so as to allocate a resource to a job/task.

II. STATIC ALGORITHMS

Static strategies are classified into three categories:
List based scheduling schemes consists of two phases. In the first phase all the active tasks are defined as nodes in a DAG and each node is assigned a priority. This phase is referred as Task prioritization phase and is followed by resource assignment phase where the resources are assigned to each task according to the priorities assigned in the first phase in order to minimize the cost defined according to some function. These algorithms are not complex and give good results.HEFT(Heterogeneous Earliest Finish Time) and CPOP(Critical Path One Processor) are examples of list based Static algorithms.[1,2,3]

Task duplication based schemes involves duplicating the tasks and running them on multiple processors to reduce the waiting time of the dependent tasks. This helps in reducing the communication cost of intermediate results and also reduces the possibility of processors waiting for the successive results in a long computation. Examples of task duplication based static schemes are STDS (Scalable Task Duplication based Scheduling) and HCNF (Heterogeneous Critical Node First).[6,7]

A much generalized workflow of a static scheduling algorithm that strikes a balance between different degrees of price and speed of execution is as follows:

i. User specifies the job characteristics such as maximum task duration and data size.
ii. A DAG depicting the execution plan is defined by parsing the above specifications. This DAG is the input to the static job scheduler.
iii. The schedules for the job execution on the cloud are computed by the scheduler.

iv. These price and finishing time schedules are provided the user.
v. The user selects the desired schedule that defines the deadline and pricing details according to their preferences.
vi. The selected plan is used by the execution platform to dispatch the user's task to the selected virtual machines where they are executed.
vii. The scheduler is informed of the completion of the tasks by the execution platform and the users are refunded the amount charged if their job uses fewer resources than allocated, according to the pricing policy.

Program → Parser → Execution Plan → Job Scheduler
→ Schedule → User choice → User chosen schedule
→ Job execution platform

Finding an optimal schedule using static schemes is very difficult and problem is NP-complete. Approximation algorithms do not give acceptable results. Therefore list based static algorithms are more common and HEFT algorithm defines priorities for each task which are calculated from the difference in time before the scheduling time of the next task and the execution time of the previous task, that depends on the average values of communication and processing times.
Other than optimization problem, static scheduling algorithms suffer from one another problem. As the computing resources scale upwards the static schemes suffer unpredictable failures. A single resource failure may result in arbitrarily long execution times and replication becomes mandatory to overcome this obstacle.

III. DYNAMIC ALGORITHMS

In contrast to static schemes discussed above, task based runtime systems uses dynamic strategies more commonly. These schemes adapt their decisions very quickly to the current state of the resources and the actual computing environment. These schemes are very myopic giving very short term view. Most of the task based runtime systems rely on dynamic schemes for scheduling. Dynamic schemes can be classified into two categories - task centric and resource centric.

In Task centric strategies, decision to schedule is taken as soon as a task becomes ready for execution. Minimum Completion Time [MCT] scheme assigns a resource to a ready task in such a way that it minimizes the finishing time of the task. Computation and data both are modeled for selection .

In Resource centric strategies decision to schedule is taken as soon as the queue corresponding to a computing resource gets empty. A task is either selected from the set of ready tasks or stolen from other resources. The algorithm that handles the distribution of work at runtime uses two approaches. In the first approach, referred as work – dealing, a master node takes the responsibility of distributing and balancing the work among all available resources . In the second approach, known as work – stealing, when any resource becomes free can steal tasks from other resources termed as victim. The criteria of stealing relies heavily on the principle of locality, so as to, have minimum data movements. The selection of a victim node is very critical in the successful implementation of work –stealing schemes. If the selection is not done properly, it can lead to significant underutilization of available resources and performance degradation. For example, if the resources steal from only a subset of related resources, will result in highly unbalanced system.

Replication is an important tool used in scheduling strategies to reduce waiting time for dependent tasks waiting for intermediate results. At the end of computation, when a resource becomes idle but there are no available tasks, it duplicates the execution of already running tasks on another resource. Several studies have analyzed the cost and benefits of replication. This technique is very commonly used in grid computing.

Cloud computing gives unlimited usage of computing resources on pay per use model.Most commercial clouds rents instances on per hour basis. Ideally the payment scheme should include the time required by each task and the amount of data transferred required executing the task. So a perfect pricing model should declare the cost of computation taking into account the number of rented instances and the duration of use. If all communication and processing time are accurately known in advance, the dynamic schemes perform in a greedy static behavior.

IV. HYBRID ALGORITHMS

Hybrid Scheduling techniques use static allocation policies along with a dynamic strategy to adjust and comply with changes in timing predictions which arise due to several real time factors such as error in predictions, resource failures or concurrent applications.

Hybrid schemes use initial static mapping and a dynamic policy to cope with runtime issues of communication and processing. For example, dynamic schedulers can use priorities defined by static algorithm HEFT, to decide which task should be scheduled first in situations when more than one tasks becomes available.

[8][9] have considered the static and dynamic strategies for the outer product computation. [5] focuses on the design and analysis of static, dynamic and hybrid schemes for matrix multiplication. Several divide and conquer schemes are defined for matrix multiplication such as strassen's where successive steps can be considered as sequence of phases of independent tasks which share data.

This context is very apt to compare static and dynamic schemes and to define a hybrid scheme that take advantages of both the worlds. Analyzing the performance of an allocation strategy is difficult due to large number of parameters, designing a dynamic strategy that considers data reuse is a tedious task.

Proposed Hybrid strategy will be a two phase model . In the first phase it will use an efficient static algorithm, taking into account variable processor speeds and resource performances to define an allocation schedule. In the second phase modify these scheduling and allocation decisions depending on the state of the system and the applications running on the system.

In the scenario, where we have independent tasks operating on independent data, replication strategies can be used to achieve a good life cycle in such a way that there are limited number of tasks having multiple executions. In order to avoid unnecessary communication expenditure, the decisions should be taken carefully on the basis of location of data and the assumed processing speeds of the resources.

In the context of matrix multiplication, many studies have focused on comparing different schedulers on dense computations on heterogeneous systems [11]. [10] has proposed and analyzed some hybrid techniques for the scheduling problem with precedence constraints. [12] has analyzed the cost of communication incurred for matrix multiplication where we have limited memory.

V. ALGORITHM DETAIL

Several hybrid scheduling schemes have been proposed for analyzing some very complex problems of scheduling with predefined parameters and constraints. In the proposed scheme we will use independent tasks, which use shared data and both the lifecycle and communication cost will be analyzed. The amount of communication that is required to perform matrix multiplications has been analyzed in [18] and a lower bound for the communication cost is calculated. Static algorithms for the matrix multiplication satisfying analyze hybrid schemes that matches the lower bounds on communication costs while having good operational behavior.

The objective of the static algorithm is to optimally distribute the computation tasks between different processors in such a way so as to obtain an optimal lifespan, while keeping a close tab on communication costs. While multiplying algorithms using divide and conquer approach like Strassen's , the partial product of sub matrices are divided between the processors depending on their processing speeds. If we assume that the area allocated to each processor is a rectangular cell, then half of the perimeter of each cell represent volume of communication and so processing cost is directly proportional to the area of the cell. Thus the total amount of communication can be obtained from the product of area of each cell and the cost of transferring the by-product of the matrix multiplication after each intermediate phase. Major problem is to achieve a perfect load balance and can be solved by partitioning a square into rectangles of fixed area. This problem has been already studied in [15], [16], [17]. The Column based algorithm proposed in [15] is a 7/4 approximation scheme, but in reality it gives approximation value as low as 1.1. Divide and conquer algorithm proposed by Nagamochi et al. achieves approximation ratio upto 5/4 theoretically as well as practically. In [17] a different version of Divide and Conquer algorithm gives an approximation ratio of $2\sqrt{3}$.

On relatively small size of input data, static column based and divide and conquer algorithm give similar optimal results. But as the block sizes of order of 1000 are considered, divide and conquer algorithm gives relatively poor performance with respect to lifespan minimization. However the performance regarding communication cost is relatively better. On heterogeneous platforms, lifespan ratio is as high as 1.38 and can get higher if new processors are added.

Two new variants static column based new and static divide and conquer new provide non rectangular area assignment, while retaining the partitions provided by Column based or Divide and Conquer. Following procedure is used to implement the new variants- the output of Column based is directed to a processor in such a way that each processor is assigned equal number of tasks. The assignment is first done column wise and is followed by doing it row wise so that each cell contains exactly same number of tasks in the range of $S_k N^2 /[\sum k' S_{k}']$ where S_k is the size of each cell. In [05] it is observed that Static Column based new is more efficient in terms of lifespan than Divide and conquer and achieves similar performance with respect to communication costs.

Dynamic algorithms are easily able to adapt their decisions according to the real time state of the resources and make a good choice for task based runtime systems. However, these schemes provide a very short term and limited view .

As discussed earlier, greedy dynamic algorithms to allocate and schedule task on resources can be classified into two main categories. Task-centric view and the resource-centric approach Here , we will discuss one algorithm from each class of dynamic algorithms: Minimum Completion Time [MCT], a task-centric strategy , and MINCOST, a resource-centric strategy which selects a task among the available tasks randomly considering only those tasks which incur a minimal amount of communication, assuming that the data is already received by their source.

Hybrid algorithms use resource-centric static schemes to allocate tasks to resources at the start of the computation and then greedy strategy is used to allocate tasks to resources. This phenomenon is referred as "task-stealing"

There can be several versions of this hybrid approach, each version using a different static allocation approach in the first phase. Hybrid Column based used column based scheme, and the new version Hybrid Column based new. By using Divide and Conquer we get Hybrid divide and conquer and the new version Hybrid Divide and Conquer new..

Replication is another important feature for resource-centric algorithms. As soon as a resource becomes available at the end of a computation it looks for available tasks for allocation. But if no new tasks are available, this feature allows duplicating the execution of an already started task on some other resource to this resource. The objective is to decrease execution time of a big computation.

VI. ALGORITHM EVALUATION

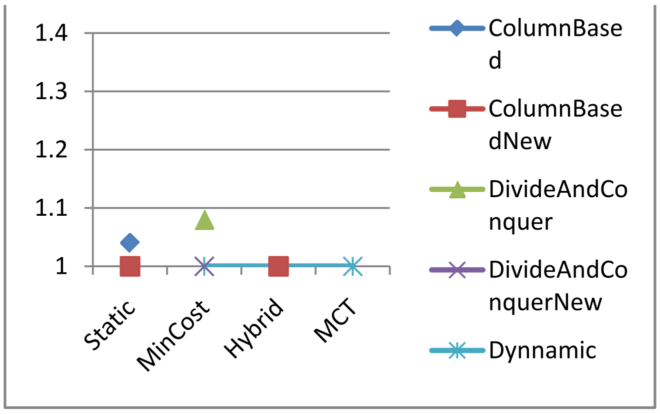
In this section we present the results of our evaluations of the different algorithms presented above.

A. Static Scenario

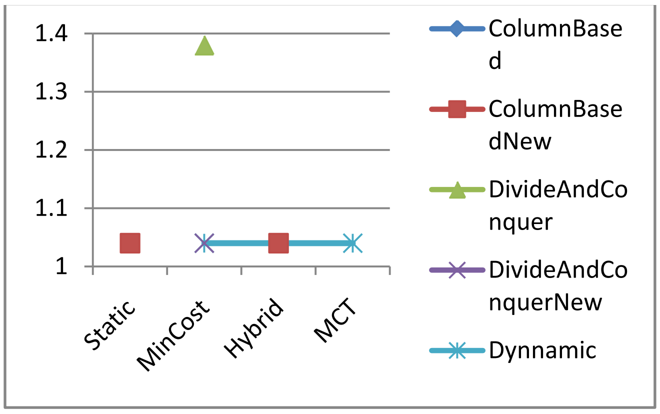
Static algorithms rely on completely accurate parameters where we have stable resource performances with time. In [05] it is observed that all the algorithms give near optimal performance except Static column based and Divide and Conquer. There is non-significant variation in case of heterogeneous platforms.

As far as communication costs are concerned, static algorithms give better performance as there is no transfer of information once allocation schedule is calculated. But the hybrid new versions do not incur any significant extra cost. Their performance is equally good and approximation ratio is always under 1.5. The reason behind this performance is that in practice very few task stealing operations take place. The only exception is hybrid column scheme for heterogeneous platforms which has high communication overhead. The reason is that it unnecessarily replicates task on multiple machines which does not contribute in minimizing the execution time.

The communication costs for purely dynamic strategies are considerable high. MINCOST algorithm gives approximation ratio of 2 for heterogeneous platform and 2.5 for homogeneous ones. It is larger than 9 for homogeneous platform in MCT algorithm.



Lifespan as a function of the chosen algorithm for homogeneous systems for static settings



Lifespan as a function of the chosen algorithm for heterogeneous systems for static settings

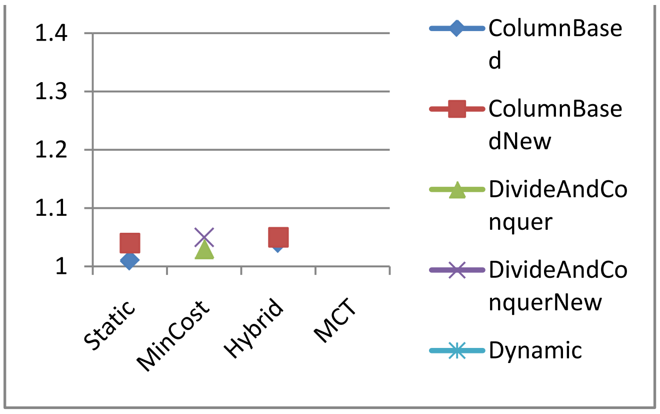
B. Dynamic scenario

In a dynamic environment, static algorithms perform poorly. This is because the processor speeds vary and the performance of a single processor can have a huge impact on overall performance.. The situation worsens further if the processors involved have significant variations in their speeds. For dynamic algorithms, replication is compulsory to have a good lifespan, particularly in cases where variation in parameters is very high. Replication once is enough. However, if instead of faster processors, slower processors are used for replication, it will make a very large impact on computation time.

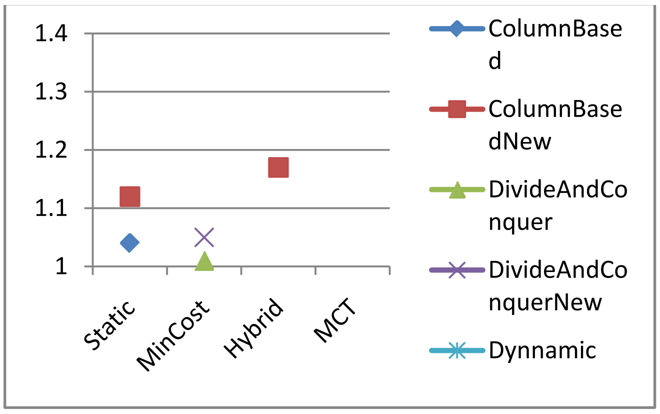
In case of communication cost, the static algorithms give similar performance as they give for static scenario. MCT performs poorly , whereas the MINCOST and the hybrid algorithms perform considerably better keeping approximation ratio below 3, even in cases where only a single replication is allowed. In addition MCT displays better robustness against the varying parameters of the underlying platform.

Varying parameters affect Hybrid strategies more since predicting unreliable processing speeds have negative effect on their static assignments It is observed that Hybrid Divide and Conquer new , achieve a better balance, and is more effective when there is less variance in parameters since there is less job stealing.

From these observations we can infer some facts. First, resource based strategies gives better performance if we wish to have minimal communication cost, and in addition these strategies give optimal life-span. Second, purely static schemes are not reliable enough to be used in practice, but when we use hybrid schemes by adding a dynamic twist to the static algorithm, it becomes both a cost-efficient and time-optimal strategy.



Communication Cost as a function of the chosen algorithm for homogeneous systems for static settings



Communication Cost as a function of the chosen algorithm for heterogeneous systems for static settings

VI. CONCLUSION

We have considered the problem of allocating and scheduling a mathematical problem onto a set of heterogeneous resources whose performance cannot be predicted and may change with time. On the one hand, since input data is shared among different tasks, it is important to use intelligent allocation schemes that generally cannot be found with the short term view of a purely dynamic runtime strategy. It is often believed that only dynamic runtime strategies can perform in dynamic real time environments. We have thus studied and analysed the behaviour of static, dynamic and hybrid strategies. We have observed that both static strategies and dynamic strategies fail to perform well and obtain a reasonable lifespan in absence of replication, but when done it is seen that it is enough if replication is done once. On the other hand, dynamic algorithms achieves a consistently low lifespan but at the cost of a very high communication overhead. Finally, hybrid strategies are able to get the advantages of both schemes, even in situations with very large variance. This advocates strongly the addition of more static knowledge in task-based runtime systems. In addition to this, it also motivates the design and analysis of good hybrid algorithms. This also helps in developing more static algorithms whose results can be used as input of hybrid strategies.

VII. REFERENCES

[1] R. Eswari and S. Nickolas, "Path-based Heuristic Task Scheduling Algorithm for Heterogeneous Distributed Computing Systems", International Conference on Advances in Recent Technologies in Communication and Computing, 2010.

[2] H. Arabnejad, J. Barbosa, " List Scheduling Algorithm for Heterogeneous Systems by an Optimistic Cost Table", IEEE Transactions on Parallel & Distributed Systems, Vol. 25, PP. 682-694, March 2013.

[3] H.Topcuoglu, S. Hariri, and M.Y.Wu, "Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing", IEEE Trans. Parallel and Distributed Systems,March 2002, Vol. 13, No.3, pp. 260-274.

[4] Aida A. Nasr,Nirmeen A. El-Bahnasawy and Ayman El-Sayed "Task Scheduling Algorithm for High Performance Heterogeneous Distributed Computing Systems"International Journal of Computer Applications (0975 – 8887) Volume 110 – No. 16, January 2015.

[5] Olivier Beaumont, Lionel Eyraud-Dubois, AbdouGuermouche, Thomas Lambert. Comparisonof Static and Dynamic Resource Allocation Strategies for Matrix Multiplication.26th IEEEInternational Symposium on Computer Architecture and High Performance Computing (SBAC-PAD), 2015, Oct 2015, Florianopolis, Brazil.

[6] M. Jing and L. Kenli, "Energy-Aware Scheduling Algorithm with Duplication on Heterogeneous Computing Systems," Publish in: Grid Computing (GRID), ACM/IEEE 13th International Conference, Page: 122 -129, Sept. 2012.

[7] T. Gautier, X. Besseron, and L. Pigeon, "Kaapi: A thread scheduling runtime system for data flow computations on cluster of multi-processors," in PASCO '07. New York, NY, USA: ACM, 2007.

[8] C. Boeres, A. Lima, and V. Rebello, "Hybrid task scheduling: integratingstatic and dynamic heuristics," in 15th Symposium on ComputerArchitecture and High Performance Computing, 2003. Proceedings, Nov.2003, pp. 199–206.

[9] J. V. F. Lima, T. Gautier, V. Danjean, B. Raffin, and N. Maillard,"Design and analysis of scheduling strategies for multi-cpu and multi-gpuarchitectures," Parallel Computing, vol. 44, pp. 37–52, 2015.

[10] G. Ballard, J. Demmel, O. Holtz, and O. Schwartz, "Minimizingcommunication in linear algebra," SIAM Journal on Matrix Analysisand Applications, vol. 32, no. 3, pp. 866–901, Jul. 2011, arXiv:0905.2485. [Online]. Available: http://arxiv.org/abs/0905.2485

[11] O. Beaumont, V. Boudet, F. Rastello, and Y. Robert, "Partitioning asquare into rectangles: Np-completeness and approximation algorithms,"Algorithmica, vol. 34, no. 3, pp. 217–239, 2002.

[12] ThomasA. Henzinger, Anmol V. Singh, Vasu Singh and ThomasWies," Static scheduling in clouds"

[13] O. Beaumont and L. Marchal, "Analysis of dynamic scheduling strategiesfor matrix multiplication on heterogeneous platforms," in HPDC'14.ACM, 2014.

[14] H.-J. Lee, J. P. Robertson, and J. A. Fortes, "Generalized cannon'salgorithm for parallel matrix multiplication," in Proceedings of the 11thinternational conference on Supercomputing. ACM, 1997, pp. 44–51.

[15] O. Beaumont, V. Boudet, F. Rastello, and Y. Robert, "Partitioning asquare into rectangles: Np-completeness and approximation algorithms,"Algorithmica, vol. 34, no. 3, pp. 217–239, 2002.

[16] H. Nagamochi and Y. Abe, "An approximation algorithm for dissectinga rectangle into rectangles with specified areas," Discrete AppliedMathematics, vol. 155, no. 4, pp. 523 – 537, 2007.

[17] A. Fügenschuh, K. Junosza-Szaniawski, and Z. Lonc, "Exact andapproximation algorithms for a soft rectangle packing problem," Optimization,vol. 63, no. 11, pp. 1637–1663, 2014.

[18] G. Ballard, J. Demmel, O. Holtz, and O. Schwartz, "Minimizingcommunication in linear algebra," SIAM Journal on Matrix Analysisand Applications, vol. 32, no. 3, pp. 866–901, Jul. 2011, arXiv:0905.2485. [Online]. Available: http://arxiv.org/abs/0905.2485