An - Najah Univ. J. Res. (N. Sc.) Vol. 32(1), 2018

Toward Inference Based Distributed Processing, Analytical Modelling and Performance Evaluation

نحو المعالجة الموزعة المستندة إلى الاستدلال: نمذجة التحليلية وتقييم لالأداء

Muhannad Al-Jabi & Aladdin Masri*

مهند الجابي، وعلاء الدين المصري

Computer Engineering Department, Faculty of Engineering & Information Technology, An-Najah National University, Nablus, Palestine

***Corresponding author: masri@najah.edu**

Received: (9/6/2017), Accepted: (24/1/2018)

Abstract

Multi-processor distributed systems are very useful for the computation of intensive load. However, the optimal load fractions allocated to each processor are the main issues that affect the performance of these systems. Therefore, these load fractions depend on different parameters, such as processing capability for each processor and communication time with each processor. So, these values must be known before we start to distribute the load. In this paper, we present a model-based approach to study the performance of multi-processor distributed systems and the different communication times that use an inference message to collect such information. We notably propose a new model for load distribution over different regions connected through wide area network (WAN). We mainly focus on the effect of total communication time over the final result. Performance analyses are evaluated by a simulator, based on C++ programming language that can be an excellent solution to calculate the total finish time, in addition to the limits over the maximum number of regions.

Keywords: Distributed systems; Intensive Load; Communication Time; WAN; Model-based Approach.

ملخص

تعتبر أنظمة متعددة المعالج الموزعة مفيدة جدا لحساب الحمل المكثف. ومع ذلك، فإن تجزئة الحمل الأمثل المخصصة لكل معالج هي القضية الرئيسية التي تؤثر على أداء هذه الأنظمة. لذلك، تعتمد هذه الاجزاء على عوامل مختلفة، مثل القدرة التجهيز لكل معالج ووقت الاتصال مع كل معالج. لذلك، يجب أن تكون هذه القيم معروفة قبل أن نبدأ في توزيع الحمل. في هذه الورقة، نقدم نموذجا لدراسة أداء الأنظمة الموزعة متعددة المعالجات وأوقات الاتصال المختلفة التي تستخدم رسالة الاستدلال لجمع هذه المعلومات. نحن نقترح بشكل خاص نموذجا جديدا لتوزيع الحمل على مناطق مختلفة المتصلة من خلال شبكة منطقة واسعة (وان). ونحن نركز أساسا على تأثير إجمالي وقت الاتصال على النتيجة النهائية. يتم تقييم تحليلات الأداء من قبل جهاز محاكاة، استنادا إلى لغة البرمجة C ++ التي يمكن أن تكون حلا ممتازا لحساب إجمالي وقت الانتهاء، بالإضافة إلى القيود على الحد الأقصى لعدد المناطق.

الكلمات المفتاحية: الانظمة الموزعة، الاحمال الكثيفة، وقت الاتصال، وان، منهج قائم على النمذجة.

Introduction

During the last decade, the huge development in the speed of the processors and communication links has resulted significant changes on high-performance computing. This development has let high-performance computing environments to consist of powerful workstations interconnected via high-speed communication links instead of expensive special purpose super computers.

Distributed memory systems with multiple address space such as clusters of workstations or network-based multi-computers have become the most prominent. A good example is the Tera-scale computing system project which achieved 6Tflops peak capability (Bataineh, S. M.). This system consists of (2728) alpha processors, 2.7Tbytes memory and 50T bytes disk space. In the literature (Pfister, G. F.), there are more than 1,000,000 computer clusters in use worldwide. Example of such systems, include IBM SP2, DEC TruCluster, HP, Intel /Sandia ASCI Option Red, etc. The importance of distributed systems increased not only because of

their scalability but also because they can handle large computational loads. Therefore, the performance of such systems is important and of interest to many researchers (Pfister, G. F.; Thomas, R.).

Consequently, computational job has been increasing according in diverse needs like the demands to resources such as high energy and nuclear physics experiments (Yu, D. & Robertazzi, T.) or large size image processing (Veeravalli, B. & Ranganath, S.). However, by coupling numerous heterogeneous computational and storage resources the term "Grid computing" has been added to the world of distributed computing. Grids are able to accommodate very large resource-demanding jobs (Thysebaert, P. & De Leenheer, M.; Mamar, A. & Lu, Y.) shows many of the currently working grids and the huge number of processors of each grid. Moreover, each Grid System consists of many clusters and each cluster consists of a large number of processors (Broberg, J. & Venugopal, S.; Singh, S. & Bawa, R.; Sarkar, A. D. & Roy, S.; Seinstra, F. J. & Maassen, J.).

Therefore, evaluation of multicomputer systems is an interest of computer designers and a challenge for computer scientists and researchers. When evaluating the performance of multicomputer systems, one has to take different parameters in considerations, such as, system parameters (processors speed, link speed), application parameters (degree of divisibility of the parallel job and the level of interactions among tasks of the same job), fault tolerance at both the application and system levels, number of processors available, scheduling algorithm, and the processors allocation algorithm. The conventional modeling techniques failed in to integrate all those elements in a generic model because they normally evaluated the performance of a specific machine. Therefore, there is a need to develop an efficient performance evaluation model, which has the ability of scheduling a given load on a number of available processors in order to minimize the finish time. The model has to be generic while

considering the system parameters, application parameters and algorithms adopted for process allocation and scheduling. That is, the system has to be independent of machine, link, applications, etc. An efficient performance model is the model that can explain all normal behaviors, predict any abnormality in the system and allow the designer to adjust some of the parameters and ignoring unimportant details.

Conventional computer modeling techniques involve detailed simulation of individual hardware components and introduce too many details to be of wide practical use. Some of these techniques measure the system as a whole, while others measure specific aspects of the computer system such as node utilization, I/O speed, operating system performance, etc. For example, (Thysebaert, P. & De Leenheer, M.) focused on the communication speed only.

Most applications, such as signal and image processing, Kalman filtering, cryptography, and genetic algorithms, all involve parallel and distributed computing in order to improve their execution performance. These applications lend themselves to divisible load theory (DLT). DLT has been considered as a powerful tool for modeling data intensive computational problems incorporating communication and communications issues (Drozdowski, M.; Wong, H. M. & Yu, D.; Moges, M. & Yu, D.).

In this paper, we present a new design of a load distribution model for a grid of processors by taking into account the time needed to collect information about processors (inference time) in addition to communication time, processing time and collecting the results time. The main contribution of this paper is the closed form solution for obtaining the minimum finish time and optimal load allocation over each processing region. We validate our model through mathematical proof and comprehensive simulations. The model is based on theoretical load distribution. It can be very efficient for huge distributed system load.

The originality in our work is that when a controller processor (initiator processor) has a load, it has the ability to infer information about other regions by inference messages to/from the controller of each region. After that, each controller has to reply to the inference message by a message contains information such as the number of busy/idle helper processors in its region, the communication time, etc.

Moreover, we developed a simulator by using the C++ language for our model that tries to find the finish time when the load is distributed over different numbers of regions. Then, based on the minimum finish time, the optimum number of regions that can share the specified load is easily specified.

This paper is organized as follows. In Section 2, related work is presented. Section 3 shows the system model. Section 4 and 5 contain results and conclusion, respectively.

Literature Review

The attempts to find an analytical model to study the performance of multicomputer distributed systems are very important. What is mostly common among all the work of multicomputer distributed systems is that the load is divisible. That is, the parallel job consists of independent tasks. Several previous studies have considered this type of load under a variety of assumptions. In literature, several previous models (Balasubramaniam, M. Banicescu, I. & Ciorba, F. M.; Rosas, C. Sikora, A. Jorba, J. Moreno, A. & César, E.; Abdullah, M. & Othman, M.; Abdullah, M. & Othman, M.) tried to study the optimal distribution of highly divisible load using DLT, but they considered specific parameters. (Balasubramaniam, M. Banicescu, I. & Ciorba, F. M.) focused on the perturbation parameters in terms of processor availability and network latency and bandwidth. However, our model is more general that is it considers these parameters (processor availability and network latency) in terms of processing power and

communication time. In addition, the main contribution in our model is the inference message to collect information about the state of the system parameters.

In (Rosas, C. Sikora, A. Jorba, J. Moreno, A. & César, E.) a methodology is proposed to dynamically improve the performance of certain data-intensive applications based on adapting the size and the number of data partitions and the number of processing nodes. In addition, the tuning parameters considered in that methodology are the partition factor for the data set, the distribution of the data chunks and the number of the processing nodes to be used. In (Abdullah, M. & Othman, M.; Abdullah, M. & Othman, M.), a closed form solution is proposed to calculate the fractions of the data file to be transferred to each cluster. In this model, the input data sources send the data concurrently to a certain cluster and computation starts only after the assigned data set is totally transferred to that cluster. However, it did not mention the criteria by which the data sources decide to transfer data to a specific cluster; also, it did not mention the criteria by which the cluster decides the computation power of other clusters. (Viswanathan, S. & Veeravalli, B.) present scheduling strategies for scheduling large-scale computational task on cluster/grid computing environment. Each cluster has its coordinator processors that collect information about other processors in the cluster and distribute the load among them. However, it did not consider the coordination time. Consequently, in (Bataineh, S. M.) the model was introduced as a network of processors (workstations) connected through a high performance local area network (LAN). Processors, in that model, are classified into central processors and helper processors. The central processors are distributed evenly. Each helper processor can be idle or processing a local task (from its central processor) or processing an external task (from nearby central processor). Therefore, the central processor views each one of its helper processors as available or busy. When a central processor receives a

parallel job, it has to first: stop running the local job and determine the optimum number of helper processors, second: it distributes to the helper processors their assigned task while computing its own job, finally it collects the results. It seems clear that the models introduced in (Bataineh, S. M.; Abdullah, M. & Othman, M.; Abdullah, M. & Othman, M.) have not considered the congestion effects due to the concurrent transmission, and have not considered the time needed to collect information about the available processors.

In the next section, we focus on the problem of scheduling large-volume loads (divisible loads) among multiple regions (clusters of computers) and the model we propose.

System Modeling

In this work, the grid computing system infrastructure considered consists of a network of super computers and/or clusters of computers (regions) connected by a wide area network (WAN), having different computational and communication capabilities. Communication is assumed to be predominant between different regions and to be negligible within a single region. Each region has a controller processor that has to store and view the information of other busy/idle helper processors within its region. In addition, the controller processor has to infer whether to distribute the load fractions among idle processors in its region or send the load fractions to other regions based on the available idle helper processors. We assume each region to be homogenous according to the processing power of all processors within that region.

When a controller processor (initiator processor) has a load, it has the ability to infer other regions by inference messages to the controller of each region. Each controller has to reply by a message with information such as the number of busy/idle helper processors in its region, the communication time, etc. Thereafter, the initiator processor collects the

information inside the reply messages and the initiator processor can now take its decision to distribute the load among other regions or to process the load locally within its region based on the optimum finish time.

Based on the useful concept of the Divisible Load Theory, one can replace a network of processors by an equivalent processor that has exactly the same computational power as the original network (Yu, D. & Robertazzi, T.). In our work, we use this concept in order to view the system introduced as a multidimensional distributed communication "tree" network. Each "leaf of the tree" represents a region. Transmitting message between two regions requires that the transmitted message must be received by the controller of that region.

From Fig.1, we can see the communication steps needed between the controller of the originating region and the controllers of each other regions and the controller of the destination region and a processor in that region. Where the time needed to:

- Send an inference message to region*i*=inference time * region number, also the same time is needed to reply to the inference message.
- Send the load fraction to region *i* = communication time *region number.
- Process the load fraction by region *i* = inverse of the processing power* load fraction.
- Send the results back = inference time * region number.

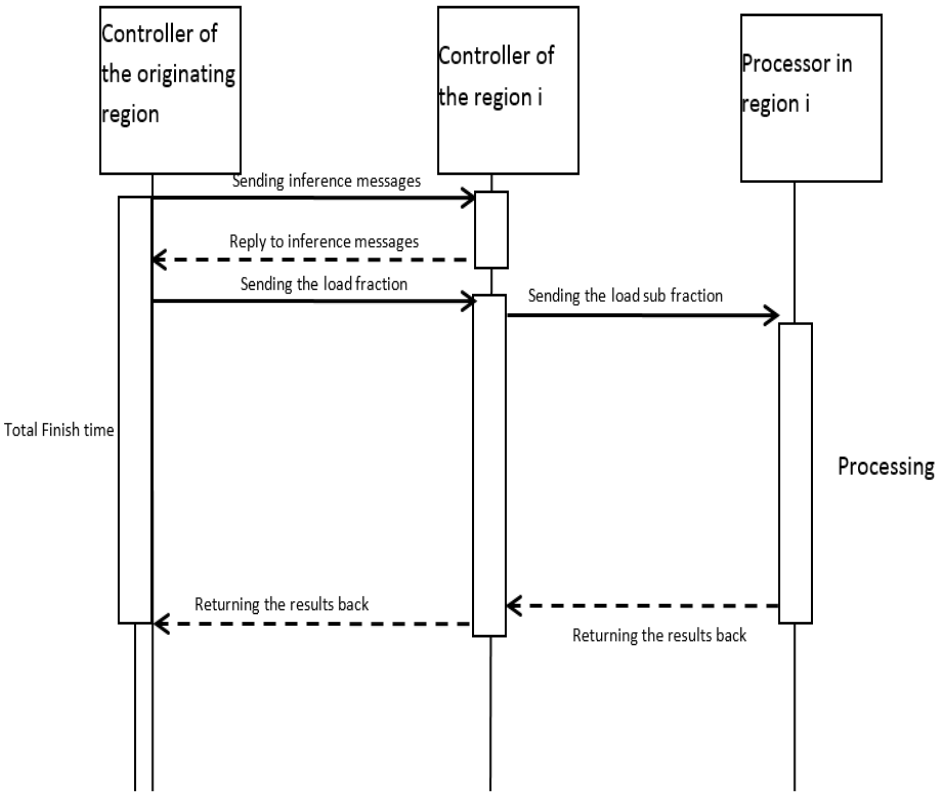

Figure (1): Time Sequence Diagram.

In this scenario, the controller of the region *j* sends an inference message to the controller of region *i*. Then the controller of region *i* replies to the inference message with the required information. The load fractions are calculated by the controller of region *j* and transmitted to region *i*. Consequently, region *i* starts execution of its load fraction then the results are returned back. Table 1 lists the notations used in our equations for the mentioned concepts.

Table (1): Terminology, definitions and notations.

Notation	Significance
α_i	Fraction of the total load that is assigned to region i.
T_{inf}	Inference time.
T_c	Communication time between regions.
T_{fpi}	The time at which the region i will finish processing its load.
T_{fi}	Total finish time including returning back the results.
Q_i	The inverse of the processing power for a region i.
T_{cp}	The computation time for the entire load if the entire load is assigned to a region having $Q = 1$.

Suppose a controller of a region decided to distribute the load to other regions, it has to infer to which regions the load will be distributed and then send the load fractions to the desired regions, so the total finish time for each region will be:

T_{fi}= inference time + processing time + communication time + return the results time

In order to actualize consistency and bandwidth preservation, we consider that different regions have different inference time and communication time with multiplicity of T_{inf} and $T_c$respectively. So, a communication time T_c to send the load fraction or return the results from region1, and $2T_c$to send the load fraction or return the results from region2 are needed. This means:

- The communication time with region number $i = i*T_c$
- The inference time, to infer region i, $= i*T_{inf}$
- The time to return back the results from region $i = i*T_c$

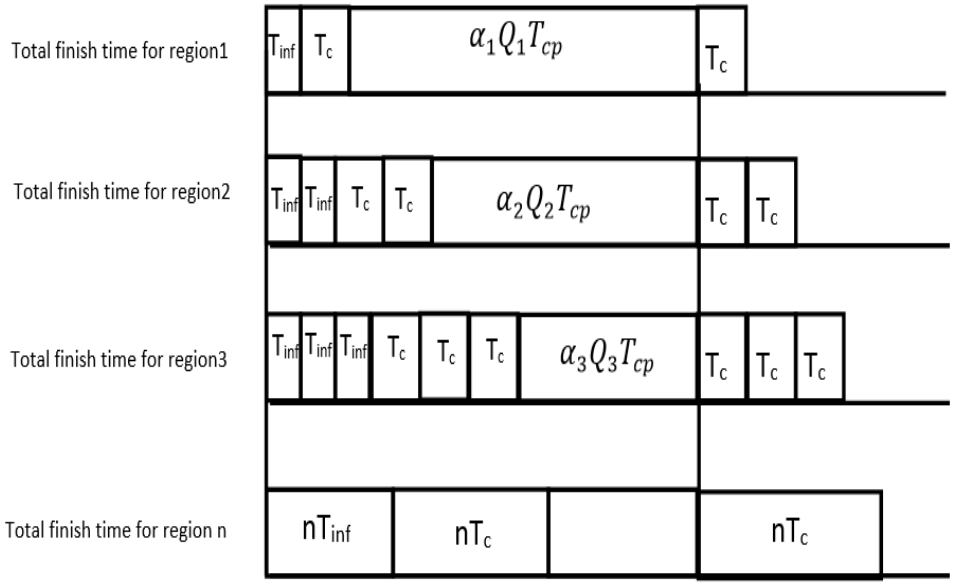

Figure (2): Timeline diagram.

Therefore, the time at which the region *i* will finish processing its load will be:

$$T_{fpi}=i*T_{inf}+ \alpha_i * Q_i * T_{cp}+ i*T_c \quad (1)$$

Moreover, the time at which region i will finish processing its load and will return the results will be, as in Fig. 2:

$$T_{fi}=i*T_{inf}+ \alpha_i * Q_i * T_{cp}+ i* T_c + i*T_c=T_{fpi}+i*T_c$$

This yields to:

$$T_{f1}=T_{inf}+ \alpha_1 * Q_1 * T_{cp}+ T_c + T_c =T_{fp1}+T_c$$

$$T_{f2} =2*T_{inf}+ \alpha_2 * Q_2 * T_{cp}+ 2*T_c+2*T_c=T_{fp2}+2*T_c$$

and:

$$T_{fn}=n*T_{inf}+ \alpha_n * Q_n * T_{cp}+ n*T_c+n*T_c=T_{fpi}+n*T_c$$

Since all regions must stop processing on the same time:

$$T_{fp1} = T_{fp2} = T_{fp3} = \ldots\ldots = T_{fpn}$$

We get:

$$T_{inf} + \alpha_1 * Q_1 * T_{cp} + T_c = 2*T_{inf} + \alpha_2 * Q_2 * T_{cp} + 2*T_c$$

$$\alpha_2 = \frac{\alpha_1 Q_1 T_{CP}}{Q_2 T_{CP}} - \frac{T_C + T_{inf}}{Q_2 T_{CP}}$$

Alternatively:

$$\alpha_n = \frac{\alpha_1 Q_1 T_{CP}}{Q_n T_{CP}} - \frac{(n-1)(T_C + T_{inf})}{Q_n T_{CP}} \quad \square \qquad \square\square\square$$

In order to satisfy that α_n is positive, the following condition must hold:

$$\alpha_1 Q_1 T_{CP} \geq (n-1)(T_C + T_{inf})$$

Knowing that, the summation of all load fractions equals to 1:

$$\sum_{i=1}^{n} \alpha_i = 1 \qquad \square\square\square$$

From (2) and (3), we get:

$$\alpha_1 = \frac{1 + \sum_{i=2}^{n} \frac{(i-1)*(T_C + T_{inf})}{Q_i * T_{CP}}}{1 + \sum_{i=2}^{n} \frac{Q_1}{Q_i}} \qquad (4)$$

Simulation and Results

We can now start investigating the correctness of our model by assuming homogenous regions according to the processing power of all regions, such that, $Q_1=Q_2=Q_i$. Therefore, the dominant system parameter in satisfying the condition from (2) will be (n) for certain values of T_c and T_{inf}. In other words, the value of T_{cp} will limit the maximum number of regions that can take part in processing the load since all regions must stop at the same time. Therefore, when T_{cp} is small, the maximum value of n has to be smaller than when T_{cp} is large due to the communication and inference overhead.

Moreover, in our simulation we have found that there is a maximum value of n and the optimum value of n. The optimum value of n is obtained based on the minimum total finish time. Based on the equations in the system model we have built our simulator by using C++ language and in our simulation, we assume that the value of T_c=0.001s and T_{inf}=0.001s and Q =0.01 for different values of T_{cp} and for different values of n.

We found that for fixed values of T_{inf}, T_c, T_{cp} the value of n cannot be increased more than a certain value, because $\alpha_n$starts to be negative. This violates the condition from (2). Therefore, we called the largest value of n at which α_n still positive "The maximum number of regions".

Fig.3 shows the variations in α_n for different values of T_{cp}. Such that the lowest curve shows that when T_{cp}=5, the load fraction that is assigned to the last region (α_n) starts to be negative sooner than when T_{cp}=20 and T_{cp}=150. Note that the middle curve is when T_{cp}=20 and the upper curve when T_{cp}=150.

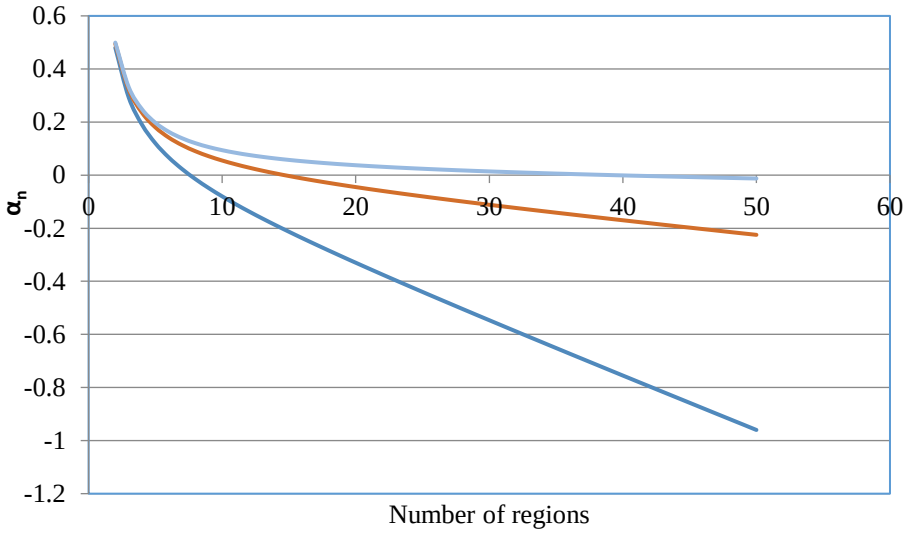

Figure (3): α_n vs number of regions.

From Fig.4 and Fig.5, we can see that as the processing time (T_{cp}) increased, the maximum number of regions that can participate in processing that load will be increased which agrees with the condition from 2. In addition, we found that for fixed values of T_{inf}, T_c, T_{cp} there is an optimum value of n at which the total finish is minimum. Therefore, we called this value "the optimum number of regions".

From Fig.5, we can see that the optimum number of regions increases proportional to the increase in the processing time (*Tcp*). This seems evident because for smaller processing time adding more regions means adding more significant communication and inference overhead.

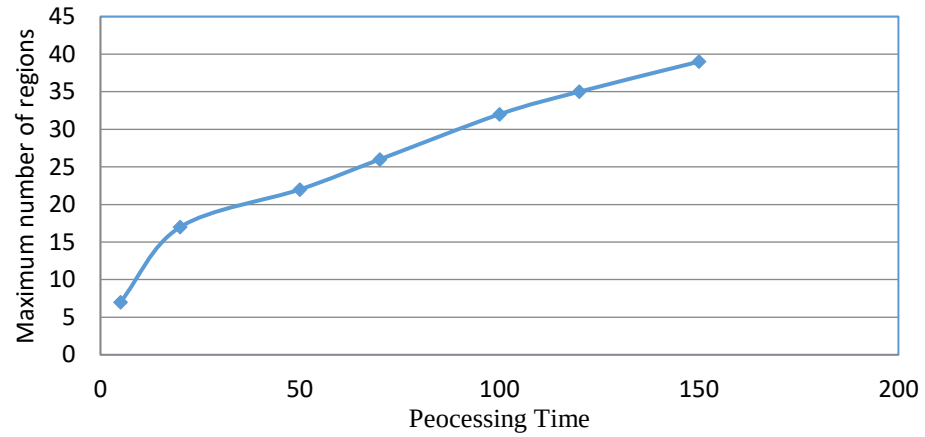

Figure (4): Maximum number of regions vs the processing(T_{cp}).

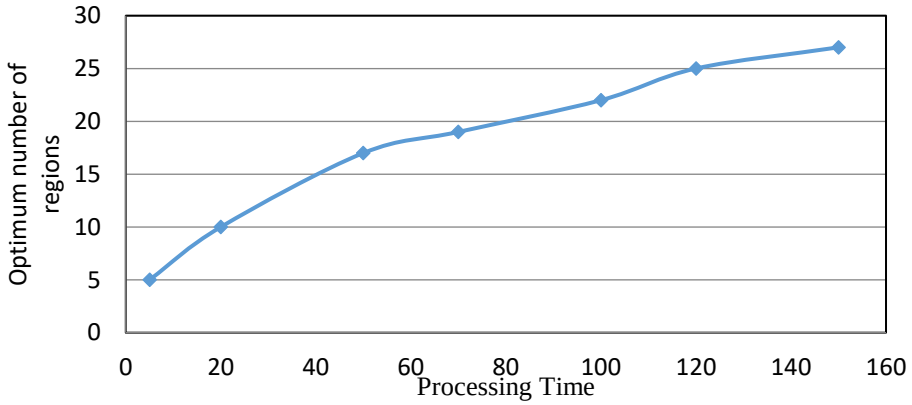

Figure (5): Optimum number of regions vs Processing Time.

However, for larger load size the extra communication and inference overhead can be negligible in comparison to the decrease in the total finish time due to distribute the load over more regions. In the upper curve, T_{cp} is 10 and in the lower curve, T_{cp} is 5. Therefore, the lower curve starts to increase before the upper curve.

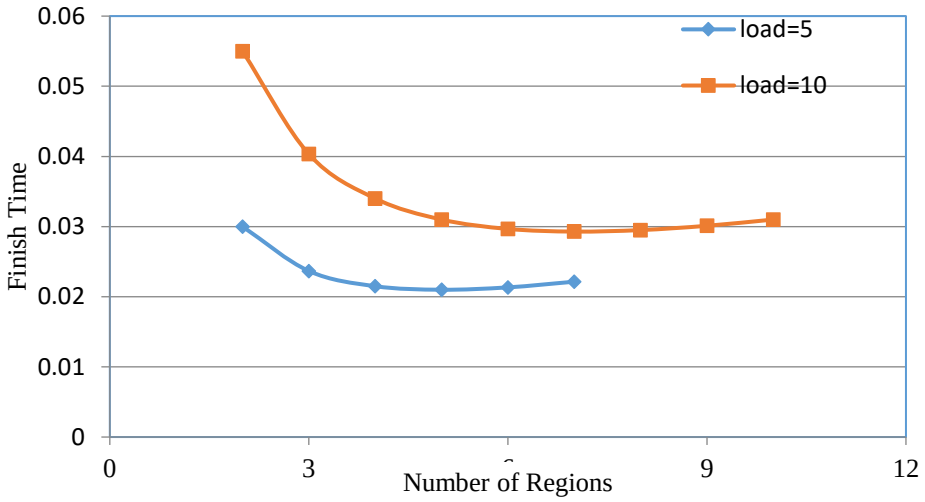

Figure (6): Total finish time vs number of regions.

In Fig.6 and Fig.7, we can see that the upper curve in Fig. 6 cannot have a value for number of regions greater than 10 and the lower curve cannot have values greater than 7. Where in Fig. 7, the upper curve cannot have values greater than 14 and the lower curve cannot have number of regions greater than 12 because of the maximum number of regions concept which we discussed before. In addition, if the number of regions exceeds the maximum number of regions, the results starts to be not realistic. That is because the total finish time starts to contain a negative term ($\alpha_n * Q_i * T_{cp}$), which not so realistic to have a negative term in time calculations.

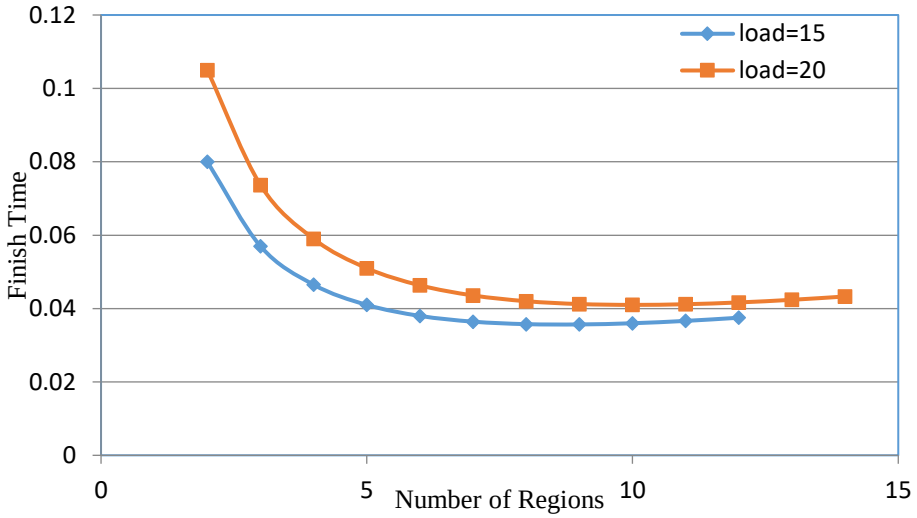

Figure (7): The effect of increasing the number of regions on the total finish time for different processing times.

In Fig.8, we investigate the effect of variation in $T_c$on the optimum number of regions by assuming different values of T_{cp}=50, 75 and 100, Q=0.01, T_{inf}=0.001 and different values of T_c and different values of n. We found that as the value of T_c increases, the optimum number of regions deceases. Also, we found that as Tcp increases, the optimum number of regions increases. That seems reasonable because when the value of T_c increase, the communication overhead increases.

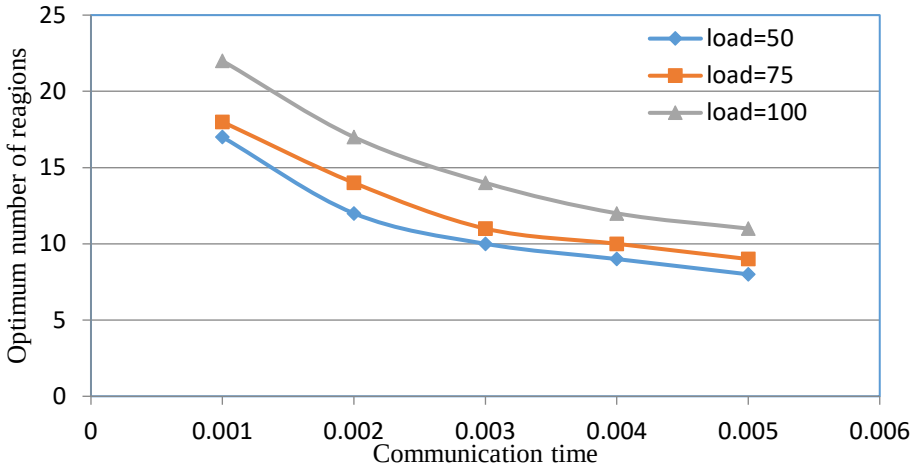

Figure (8): Optimum number of regions vs T_c

Conclusion

In this research, we have developed an effective model for optimal workload allocation. The model is proposed for load allocation for multiple regions and for scheduling divisible data grid applications. The results showed that the proposed model is capable of almost optimal solution for single source scheduling. Hence, the proposed model can balance the processing loads efficiently. We are planning to adapt the proposed model to be implemented in multiple sources. With such improvements, the proposed model can be integrated in the existing data grid schedulers in order to improve their performance.

References

- Abdullah, M. & Othman, M. (2009). Closed form Solution for Scheduling Arbitrarily Divisible Load Model in Data Grid Applications: Multiple Sources, *American Journal of Applied Sciences,* pp. 626-630.
- Abdullah, M. & Othman, M. (2009). Load Allocation Model for

Scheduling Divisible Data Grid Applications, *Journal of Computer Science,* 5: 760-763.

- Balasubramaniam, M. Banicescu, I. & Ciorba, F. M. (2013). Analyzing the Robustness of Scheduling Algorithms Using Divisible Load Theory on Heterogeneous Systems, in *IEEE 12th International Symposium on Parallel and Distributed Computing (ISPDC).*
- Bataineh, S. M. (2005). Toward an analytical solution to task allocation, processor assignment, and performance evaluation of network processors, *Journal of Parallel and Distributed Computing,* 29-47.
- Broberg, J. & Venugopal, S. (2008). Market-oriented Grids and Utility Computing: The state-of-the-art and future directions, *Journal of Grid computing,* 6:255-276.
- Drozdowski, M. (2010). Energy Considerations for Divisible Load Processing, *Parallel Processing and Applied Mathematics Lecture Notes in Computer Science,* 6068/2010: 92-101.
- Mamar, A. & Lu, Y. (2009). Department of computer Science and engineering, [Online].
- Moges, M. & Yu, D. (2009). Grid scheduling divisible loads from two sources," *Computers and Mathematics with Applications,* 58: 1081-1092.
- Pfister, G. F. (1996). Clusters of computers: Characteristics of an invisible architecture, in *IEEE Int'1. Parallel processing Symp.*, Hunolulu.
- Rosas, C. Sikora, A. Jorba, J. Moreno, A. & César, E. (2014). Improving Performance on Data-Intensive Applications Using a Load Balancing Methodology Based on Divisible Load Theory, *International Journal of Parallel Programming,* 94-118.
- Sarkar, A. D. & Roy, S. (2010). An Integrated Framework for

Performance Analysis and Tuning in Grid Environment, [Online]. Available: http://arxiv.org/abs/1005.2037.

- Seinstra, F. J. & Maassen, J. (2011). Jungle Computing: Distributed Supercomputing Beyond Clusters, Grids, and Clouds, *Grids, Clouds and Virtualization Computer Communications and Networks,* 167-197.
- Singh, S. & Bawa, R. (2011). An efficient job scheduling algorithm for grid computing, in *ACAI '11 Proceedings of the International Conference on Advances in Computing and Artificial Intelligence.*
- Thomas, R. (2003).Ten Reasons to use Divisible Load Theory, *IEEE Computer Society,* 63-68.
- Thysebaert, P. & De Leenheer, M. (2005). Using divisible load theory to diminsion optical transport networks for grid excess load handling, in *Autonomic and Autonomous Systems and International Conference on Networking and Services.*
- Veeravalli, B. & Ranganath, S. (2002). Theoritical and Experemental study on large size image processing applications using divisible load paradigm on distributed bus networks, *Image and Vision Computing,* 20:917: 935.
- Viswanathan, S. & Veeravalli, B. (2007). Resource-Aware Distributed Scheduling Strategies for Large-Scale Computational Cluster/Grid Systems, *IEEE Transactions on Parallel and Distributed Systems 18.*
- Wong, H. M. & Yu, D. (2003). Data Intensive Grid Scheduling: Multiple Sources with Capacity Constraints," [Online]. Available: http://www.ee.sunysb.edu/~tom/MATBE/multisource.pdf.
- Yu, D. & Robertazzi, T. (2003). Divisible Load Scheduling for Grid Computing, in *Proc. 15th Int'l Conf. Parallel and Distributed Computing and Systems*.
  Available:www.cse.unl.edu/~ylu/papers/rtas10.pdf.