Anatomy Study of Execution Time Predictions in Heterogeneous Systems

J. Shanthini
Assistant professor,
Department of Computer Science and Engineering,
INFO Institute of Engineering,
Coimbatore,

K. R. Shankarkumar
Professor,
Department of Electronics and Communication Engineering,
Sri Ramakrishna Engineering College,
Coimbatore,

ABSTRACT
To make effective job placement policies for a volatile large scale heterogeneous system or in grid systems, scheduler must consider the job execution time. In most grid schedulers, execution time of job is to be known in the prior. The execution time given by user may not be more precise, execution time predictors are used in order to facilitate the dynamic scheduling. The prediction algorithms use analytical bench marking/ code profiling, historical data, and code analysis. The prediction algorithm should be nonclairvoyant in nature. This study reviews execution time prediction algorithms in a different perspective. This algorithm considers memory accessing, network performance, and fluctuation of competing CPU load and so on, as interference factors for prediction. Based on the understanding comprehensive analysis is made among them.

General Terms
KNN Smoothing, Linear Regression, Instance Based Learning

Keywords
Task scheduling, historical data, code analysis, profiling, benchmarking.

1. INTRODUCTION
We are now in the era of production grids, where large no of users with different needs, use different application areas, live geographically distributed areas share the same resources. Guaranteed quality of service is one of the most challenging aspects of grid computing environment. Due to increased parallelism, tools to match code with candidate architecture and to evaluate the performance of such match are essential. So we need new ways for predicting the run time estimation of the job submitted by user. The fig. 1 shows the methods for execution time prediction, that can be classified in two categories like, code profiling/ Analytical benchmarking [1,2,3,4,5], and Statistical prediction [8,9,10,11].

2. CODE PROFILING
Code profiling was initially introduced by Freund et at. The code profiling is a code specific function, signified to identify the embedded parallelism inside the code. Actually the source code is broken into code segments, where the processing requirements of the each segment may be same or heterogeneous. These segments were given an optimum processor that produces better efficiency on the given code segments. Once the code is divided into parts, the code-type profiler is called, which will identify the nature of the code. The types of code may be vectorizable decomposable, vectorizable non-decomposable, fine/coarse grain parallel, SIMD/MIMD parallel, scalar, and special perfectly purpose (like Fast Fourier Transforms, or specialized sort algorithms).

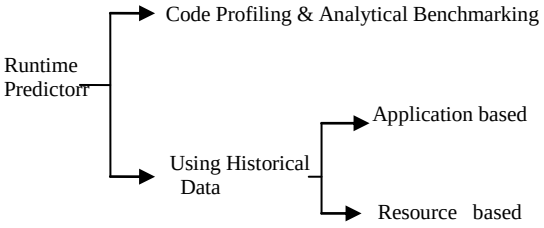

Fig.1. Taxonomy of Run time predictions

3. ANALYTICAL BENCHMARKING
In parallel programming, user may give their input data statically or dynamically. The static timing is generally is unaffected by the run time environment .But dynamic time values are highly sensitive to run time environment, these are identified by analytical benchmarking [1,2,3,4,5]. Analytic benchmarking/ code profiling was first presented by Freund [5], and has been extended by Pease et al. [3], Yang et al. [6],and Khokhar et al. [1,2]. Ashfaq A. Khokhar, Viktor K. Prasanna [1] stated that Analytical benchmarking is a test to make sure how well a machine performs on given code type. Once the code profiling is done, the type of code is identified. Now analytical bench marking determines the performance of the machine on identifying code type. Levit [7] has proposed a novel analytical model for grid communication, it takes geometry of physical and virtual processor, dimensions of communication and size of input data's in to consideration.

4. STATISTICAL PREDICTIONS
Here the run time of a task is predicted using the past observations. Each machine consists of set of past observation; this history is used to make a new prediction. The prediction method takes data's from input parameters, so no additional knowledge is required. This method has two iterations, in first iteration similar jobs are found and in the second iteration the prediction is performed. The intrinsic complexity of this method falls in selecting the similar jobs. The similar jobs an identified using or comparing different job parameters like user of the job, OS, No. Of. Nodes used and machine architecture etc. Once similar jobs are found they are grouped together. In the second iteration, the prediction is performed either by considering mean, or by using probability

functions of previous execution times or even by regression analysis model. Isolation of job histories in individual machine, prediction error on lower data sets, are seems to be a major drawback of this method.

In recent grid computing environments[15], it is suggested that, this history can be stored in cluster heads. These cluster heads will make the decision on the scheduling hierarchical workflow model.

5. RECENT WORKS

R. F. Freund [5] proposed a new idea the , where his proposal consists of 2 parts. In the first part he tends to split the code, these code groups may homogeneous or heterogeneous requirements. After finding the code types, he tries to assign to a machine which optimally suits to execute the code type. Jaehyung Yang, Ishfaq Ahmad, Arif Ghafoor [4] has proposed Augmented code profiling and augmented analytical benchmarking, to characterize applications and architectures in a Distributed Heterogeneous Supercomputing System. This method is based on generating Representative Set of Templates (RST). This RST's allows user to generate code profiles, which can represent the execution behaviour of the task at varying levels of details.

In [9] the run times were predicted using the past observations. Initially high density usage of resources was identified and clustered, using these cluster state transitions were built in order to characterize the resource usage in previous runs. They have used k-means clustering with three dimensional populations which include CPU time, file I/O, memory used. In [10,11] Ian Foster and their colleagues presented methods for predicting queue wait times and run time of a job. In earlier cases, the authors concentrated on eliciting the similarity between the jobs. They proposed two templates (q,u)and (u,n),which are used to categorize the application A. The category C_A was eliminated from the universal set of C. For each C_A run-time estimates were calculated and the one with smallest confidence interval was chosen as run time of the job A, Either mean or linear regression was used to compute the estimates.

M. A. Iverson et al [12] and their fellow mates have proposed a method that uses a non parametric regression technique based execution time prediction that used past observations. Since the non parametric regression technique does not any machine architecture, this algorithm also acts like architectural neutral. This algorithm uses K-Nearest Neighbour (K-NN) smoothing. Here estimate m (x) for the parameter A is calculated from k observations with x values closest to the parameter A. From their result it is determined that 1 D K-NN give lower computational cost than 4-D and 11-D K-NN smoothing, It is also found that is method works well on increasing jobs. Maciej Drozdowski et al. [12] Proposed some metrics for a good prediction system they are 1) Actual order of the program operation 2) respect communication delay 3) eliminates the influence of other user and application sharing computing environment and 4) having simple implementation. G (V, A) represents DAG with V forking events and A activity arcs .The duration of the computational arc between two events is measured as process time of jobs. The GetProcessTime and time methods give process time and eliminates eliminating influences by other user. It was shown that with growing contributions of processing time in overall execution time, the difference between estimated and astronomical times diminishes. It is also shown that estimation of communication time is an important source of error in this method. In [13] Christian Glasner, Jens Volkert has proposed an adaptive architecture for running time predictions. The method they proposed not only considers job parameters but also considers behaviour of users as single or as group. Here are set of observations cost is taken at time t and historic information is described as a tuple (ti, x_i), using which a filtered set is created. These sets are given as input to the predictors, and quantifiers are used to select the most opted predictor for the situation. This selection is based on the earlier behaviour of the user, and final forecast is made.

A prediction model for grid environment has been proposed by Xilong Che, Liang Hu, Dong Guo, Kuo Tang and Debin Hu sed in [14]. This work is utilizes the Globus Toolkit [15], where GT has four major components such as Grid Information System (GIS), Monitoring and Discovery System (MDS), Grid Resource Allocation Management (GRAM),and Grid File Transfer Protocol (GridFTP). This work makes use of MDS and GIS component. The prototype system proposed here registers and publishes the prediction information in GIS and realized through MDS.

In this work, the prototype runs in each grid node, monitors the usage of resources, which is used in the run time prediction. This model has two methods for collecting information namely local registration and global registration. The local registration service collects the resource information and stores it in native information service; later global registration method aggregates it with superior. Run time prediction subsystem uses this buffered information to predict the future run time in accordance with confidence degree.Hui Li [16] have used Instance based learning algorithm. His approach is based on statistical learning on historical dataset with two performance metrics namely Application run time and queue wait time. The non parametric learning technique called Instance based learning (IBL) to predict run time from historical data. The run time attributes like group name, user name, job name is used for prediction analysis.

Shonali Krishnaswamy, Arkady Zaslavsky and Seng Wai Loke [17] have used rough set theory to predict the run time of application. They have used rough set to find the similarity analysis and mean value for prediction. In rough set algorithm, reduct and Dispensability are used to reduce the iterations in finding similarities. Their conditional attributes include the application name, size, computational resources used etc. This work outperforms the previous work [10], where [10] suffers from a mean error in runtime. The work presented in [18] made an assumption that similar jobs have similar run times. They have used around

Seven conditional attributes for making decision on runtime. Templates were prepared to fine similarity between job properties, function sim(j1,j2) finds similarity between j1 and j2. Assume k is the neatest jobs found, and then the estimation of job j is the mean of run times of k nearest jobs. A standard deviation was applied to the estimation in order to overcome the problem of under estimation. It was shown that this proposal out performs the one presented in [19] and [10]. The result shows low range to mean absolute error 0% on 50% of load and 7.5% on the rest of the 50%.

Maleeha Kiran , Aisha-Hassan A. Hashim, Lim Mei Kuan and, Yap Yee Jiun[21] have projected a method exclusively for R! Scripts. Their idea has a clue from compiler operations, source code of an R job (or script) will be parsed and

tokenized similar to the way a compiler does. The execution time each token were obtained from the database and help to improve the accuracy of the prediction. Only the benchmarked combined as a whole using mathematical calculation. Here each machine begins with benchmarking application, which data's are stored in the repository hence it servers as architectural neutral method. Using R! Scripts limit this method in using different jobs. The method shows around 91% of accuracy with only 9% of prediction error

Table 1: Performance analysis of prediction algorithms

Method Proposed	**Type of prediction used**	**Application/ Resource Based**	**Run time value estimated by**	**Performance**	**Limitations /Advantages**
Augmented Analytical Benchmarking	Representative Set Analytical Benchmarking	Application Based	RST	Methodology alone	
Predictability of process resource usage [9]	K-means	Resource based	By estimating Resource usage	**Mean error 7.3%**	
Statistical Prediction of Task Execution Times [8]	Hybrid method Analytical Benchmarking and K-NN smoothing	Application based	Linear Regression	Low prediction error with large no of jobs 1-D K-NN smoothing give lower computational cost.	
Run time of a job using historical data[10,11]	Template Based Historic Information	Application	linear regression	Wait time prediction error – 34%	Suffers from mean error in runtime
Estimating execution time of distributed applications, [12]	Graph Based	Application	Weight of vectors	Low error with diminishing job matrix	Low contribution of communication delay is source of error
Information Service Prototype System for Run-time Prediction [14]	Statistical Resource Based	GIS,MDS Application	Regression	Demanded run time decreases if relative error on prediction decreases	Adv: Architectural Neutral
An Architecture for an Adaptive Run-time Prediction System[13]	Historic information	Application	quantifiers	14% deviation in prediction of 95% of submitted	
Evaluation of grid computing a model and prediction perspective [16]	Historic Information	Application	Instance based learning (IBL)	Methodology alone	
Predicting runtime of application using rough sets[17]	Historic Information Rough set	Application	Regression	Mean error less lower by 50% than worst case analysis [10]	
Using Historical Data to Predict Application Runtimes [18]	Historic information	Application	Mean with std.deviation	Mean absolute error 0% on 50% load and . 7.5% on rest , out performs [9]	

Emmanuel Jeannot, Keith Seymour and their fellow mates [22] have proposed a template based run time prediction, which done on Network Enabled Servers implemented on GridSolve. The GridSolve is a client-agent server system which provides remote access to hardware and software through a variety of client interfaces. Here the template used to predict the run time has two parts, the first part complaints polynomial of parameters of the problem and the second part consists of a set of categories far the parameters be passed for operation. The parametric regression system used to set or update the coefficient at run time . Each time when model asks for the run time prediction, system switches to the corresponding model or category and gives the execution time value. There is inadequacy in this approach on initial stage when coefficient is one this model may not work well, but later stage when the information is sufficiently high, this model gives precise result.

Marco A.S. Netto ,Christian Vecchiola, Michael Kirley , Carlos A. Varela, Rajkumar Buyya [23], has suggested a new prediction method to be used processor collocation, which employees iterative analysis. If the job has n iterations in an execution and K be iteration where execution becomes steady. The iteration starts with zero and incremented for a sweep, until the prediction become steady. The second method expects user to write outputs on each iteration, time interval between two iterations were computed. J. M. Ramirez Alcaraz et al [24] have projected a technique for run time prediction, which uses user run time estimate, system run time estimate and run time prediction. Once the job is submitted to the system a template consists of <P_j, P_j ', P_j"> is prepared, P_j execution of job j , P_j '
denotes user run time estimate and P_j" denotes
system generate predictions. Dan Tsafrir, Yoav Etsion, and Dror G. Feitelson [25] has shown that the prediction of a new job J is derived from the average runtime of the two most recent jobs that were submitted by the same user prior to J and that have already terminated will improve the prediction performance. If no such jobs exist, they fall back on the associated user estimate. David Talby Dan Tsafrir Zviki Goldberg Dror G. Feitelson [26] has come with session based, estimation–less and information-less run time prediction for parallel and grid jobs. In session based prediction they used historical data about users and the user runtime estimates of submitted jobs. Estimation is based on user session, it takes work period of user and population of job in the jobs in the same session. In Estimation less module, the user need not submit their estimation, predictor uses history to generate the estimation. In Information less prediction methods, it neither uses user estimate nor system generated prediction. This predictor is the constant predictor predict the same, constant runtime for all jobs

6. CONCLUSIONS

The run time estimates are an important research area in the current Meta computing environment because of twofold: in a grid like environment user don't know in which machine their job is going to scheduled so their estimation to their job is of no use or incorrect, the user may overestimate their job in order to get their job done as earlier. There are many methods presented for predicting the run time, each of them has their own advantages and limitations. Table 1. Presents various aspects of prediction algorithms. This paper analyses about the statistical prediction. They fall into broad categories of analytical benchmarking and history based. Great influence of program execution frequency is a clue for success history based methods. There are two ways to store the historical information from job execution. It can be stored in a central job history database, or it can be decentralized with each execution site maintaining its own job execution history. Prediction of run time is a key to improve the scheduling performance. The more research is anticipated to use data mining techniques in this field. The prediction of run time is large area this study is a small light towards that, out of which we erudite a lot about, run time prediction and branches in that

7. ACKNOWLEDGEMENTS

I would like to thank Dr. ShankarKumar for guiding me to come out with this paper. I record my gratitude to , Dr.V.Palanisamy, Pricipal and Dr.ChitraManohar, secretary INFO Intitute of Engineering. I also thank Dr. Umamaheswari for her valuable suggestions and guidance.

8. REFERENCES

[1] A. A. Khokhar, V. K. Prasanna, M. E. Shaaban, and C.-L. Wang.Heterogeneous computing: Challenges and opportunities. *IEEE Computer*, 26(6):18–27, June 1993

[2] A. Khokhar, V. Prasanna, M. Shaaban, and C.-L.Wang. Hetro- geneous supercomputing: Problems and issues. In Proc. of the 1992 Workshop on Heterogeneous Processing, pages 3–12 . IEEE Computer Society Press, Mar. 1992

[3] D. Pease, A. Ghafoor, I. Ahmad, D. L. Andrews,K. Foudil-Bey, T. E. Karpinski, M. A. Mikki, and M. Zerrouki. PAWS: A performance evaluation tool for parallel computing systems. *IEEE Computer*, 24(1):18–29, Jan. 1991

[4] J. Yang, I. Ahmad, and A. Ghafoor. Estimation of Execution times on heterogeneous supercomputer architectures.In the 1993 Inter. Conf. on Parallel Processing,volume 1, pages 219–226. CRC Press, Aug. 1993.

[5] R. Freund. Optimal selection theory for superconcurrency. In Proceedings of the 1989 Supercomputing Conference,pages 13–17. IEEE Computer Society Press,1989.

[6] J. Yang, I. Ahmad, and A. Ghafoor. Estimation of execution times on heterogeneous supercomputer architectures. In *the 1993 Inter. Conf. on Parallel Processing*, volume 1, pages 219–226. CRC Press, Aug. 1993.

[7] C. Levit, "Grid Communication on the Connection Machine: Analysis, Performance and Improvements", Tech. Report 88.19, Research Inst. for Advanced Computer Science, NASA Ames Research Center,1988

[8] Michael A. Iverson, FuEsun OE zguEner and LeePotter,Statistical Prediction of Task Execution Timesthrough Analytic Benchmarking for Scheduling in Heterogeneous Environment, IEEE Transactions On Computers, VOL. 48, NO. 12, DECEMBER 1999.

[9] M. V. Devarakonda and R. K. Iyer. Predictabilityof process resource usage: A measurement-based study on UNIX. *IEEE Trans. Software Engineering*, 15(12):1579–1586, Dec. 1989

[10] WarrenSmith, ValeirTaylor, and IanFoster, Using Run-Time predictions to estimate queue wait times and improve scheduler performance, Lecture Notes in Computer Science (LNCS), 1659, Springer-Verlag,pp.202-229.

[11] Warren Smith, Ian Foster, and Valerie Taylor, Predicting application run times using historical information, J.parallel and Dist.computing. 64(2004)1007-1016

[12] Maciej Drozdowski, Estimating execution time of distributed applications,PRAM 2001,LNCS 2328,pp 137-144,2002

[13] Christian Glasner, Jens Volkert, An Architecture for an Adaptive Run-time Prediction System. 2008 International Symposium on Parallel and Distributed Computing.

[14] Xilong Che, Liang Hu, Dong Guo, Kuo Tang and Debin Hu, Information Service Prototype System for Run-time Predictionof Grid Applications, 2nd International conference on Pervasive Computing and Applications, ICPCA 2007.

[15] www.globus.org

[16] Hui Li, Performance Evaluation of grid computing a model and prediction perspective, Seventh IEEE International Symposium on Cluster Computing and the Grid(CCGrid'07)

[17] Shonali Krishnaswamy, Arkady Zaslavsky and Seng Wai Loke,Predicting runtime of application using rough sets.IEEE distributed systems online,April 2004.Vol.5.

[18] Tran Ngoc Minh, Lex Wolters, Using Historical Data to Predict Application Runtimes on Backfilling Parallel Systems, 2010 18th Euromicro Conference on Parallel, Distributed and Network-based Processing

[19] D. Tsafrir, Y. Etsion, D. G. Feitelson, "Backfilling Using System Generated Predictions Rather than User Runtime Estimates", IEEE Transactions on Parallel and Distributed Systems, Volume 18, Pages 789-803, 2007

[20] H. Li, D. Groep, L. Wolters, "Mining Performance Data for Metascheduling Decision Support in the Grid", Future Generation Computer Systems 23, Pages 92-99, Elsevier, 2007.

[21] Maleeha Kiran , Aisha-Hassan A. Hashim1, Lim Mei Kuan, Yap Yee Jiun, Execution Time Prediction of Imperative Paradigm Tasks for Grid Scheduling Optimization, IJCSNS International Journal of Computer Science and Network Security, VOL.9 No.2,February 2009

[22] Emmanuel Jeannot, Keith Seymour, Jack J. Dongarra, and Asym Yarkhan, Improved runtime and transfer time prediction mechanisms in a network enabled servers middleware, Parallel Processing Letters, World Scientific Publishing Company, January 2006

[23] Marco A.S. Netto , Christian Vecchiola , Michael Kirley , Carlos A. Varelab, Rajkumar Buyya, Use of run time predictions for automatic co-allocation of multi-clusterresources for iterative parallel applications, J. Parallel Distrib. Comput. 71 (2011) 1388–1399

[24] Juaan manuel ramirez, Alcaraz ,Andrei Tchernykh,raminyahyapour,Uwe Schwiegelshohn , Ariel Quezada Pina,Jose Luis Gonzalez,garacia,Adan Hiralgies ,Carbajal, Job Allocation Strategies with with User run time Estimation for online scheduling for hierarchical grids, Journal of Grid Computing, Volume 9 Issue 1, March 2011 9:95–116

[25] Dan Tsafrir, Yoav Etsion, and Dror G. Feitelson, Backfilling Using System-Generated Predictions Rather than User Runtime Estimates, IEEE Trans. on Parallel And Distributed Systems, Vol. 18, No. 6, June 2007

[26] David Talby Dan Tsafrir Zviki Goldberg Dror G. Feitelson, Session-Based, Estimation-less, and Information-less Runtime Prediction Algorithms for Parallel and Grid Job Scheduling, Technical Report, school of computer science and Engineering, Hebrew University of Jerusalem 2006