Task Partitioning and Load Balancing Strategy for Matrix Applications on Distributed System

Adeela Bashir†, Sajjad A. Madani†, Jawad Haider Kazmi†, Kalim Qureshi§
†Department of Computer Science, COMSATS Institute of Information Technology, Abbottabad, Pakistan
§Department of Information Science, Kuwait University, Kuwait
Email: {adeelams@gmail.com}

***Abstract*— In this paper, we present a load-balancing strategy (Adaptive Load Balancing strategy) for data parallel applications to balance the work load effectively on a distributed system. We study its impact on computation-hungry matrix multiplication application. The ALB strategy enhances the performance with features such as intelligent node selection, pre-task assignment, adaptive task sizing and buffer allocation, and load balancing. The ALB strategy exhibits reduced nodes idle time and inter process communication time, and improved speed up as compared to Run Time task Scheduling strategy.**

***Index Terms*— task partitioning, load balancing, heterogeneous distributed systems, matrix multiplication, and performance evaluation**

I. INTRODUCTION

Distributed computing system (DS) is a better choice for multifarious computations of high processing demand scientific applications. Nodes in the DS show diverse behaviour in terms of performance under variations in workload [1]. Such variations mandate the effective distribution of tasks which is a major issue in distributed computing environment [2].

Parallel matrix multiplication is one of the most studied fundamental problems in distributed and high performance computing. Our focus is to minimize the total execution time of an application that can be achieve by *reducing inter process communication and nodes idle time*, and *incorporating effective load balancing components* [1]–[3]. Matrix multiplication is used in many scientific and engineering applications such as robots, remote sensing, and medical imaging etc. The matrix multiplication is an extensive data parallel application. We have been evaluating the performance of matrix multiplication on network of workstations [4].

Recently, the matrix multiplication computational complexity has been reduced using Strassens algorithm [5]. In [5], author replace the number of matrix multiplication by matrix additions that reduced the time complexity of matrix multiplication from $O(n^3)$ to $O(n^2.8)$. The proposed model of [4] is shown in Figure 1.

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \times \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix}$$

$$\begin{aligned} Q_1 &= (a_{11} + a_{22})(b_{11} + b_{22}) \\ Q_2 &= (a_{21} + a_{22})b_{11} \\ Q_3 &= a_{11}(b_{12} - b_{22}) \\ Q_4 &= a_{22}(b_{21} - b_{11}) \\ Q_5 &= a_{11} + a_{12})b_{22}) \\ Q_6 &= (a_{21} - a_{11})(b_{11} - b_{12}) \\ Q_7 &= (a_{12} - a_{22})(b_{21} + b_{22}) \end{aligned}$$

$$\begin{aligned} C_{11} &= Q_1 + Q_4 - Q_5 + Q_7 \\ C_{12} &= Q_3 + Q_5 \\ C_{21} &= Q_2 + Q_4 \\ C_{22} &= Q_1 + Q_3 - Q_2 + Q_6 \end{aligned}$$

Figure 1. Matrix multiplication by Strassen's algorithm [4]

The authors in [5] reduced the mathematical complexity of the application at the cost of supporting features of parallel data applications, for example, independency of sub-tasks and pattern repetition. The more is the independency and pattern repetitions, the less is the inter process communication time (this is not the case in [5] as shown in Figure 1). In [6] the authors have provided a new parallel matrix multiplication algorithm based on the Strassen's algorithm, which is named CAPS (Communication-Optimal Parallel Algorithm for Strassen's Matrix Multiplication), in which they have reduced the communication overhead of Strassen's algorithm. The authors also provided a comparison of different matrix multiplication algorithms which is shown in Table I. Here $\omega_0 = \log_2 7 \approx 2.81$ is the exponent of Strassen; ℓ is the number of Strassen steps taken.

doi:10.4304/jcp.8.3.576-584