ijecs open access www.ijecs.in
*International Journal Of Engineering And Computer Science ISSN:2319-7242*
*Volume 3 Issue 10 October,2014 Page No.8752-8764*

# Task Scheduling of Special Types of Distributed Software in the Presence of Communication and Computation Faults

***Kamal Sheel Mishra[1], Anil Kumar Tripathi[2]***

Department of Computer Science and Engineering, Indian Institute of Technology (Banaras Hindu University), Varanasi, India

ksmishra@smsvaranasi.com [1] , aktripathi.cse@iitbhu.ac.in [2]

## ABSTRACT:

**This work is an extension of our previous work on task scheduling of a Distributed Computing Software in the presence of faults [2] in which an attempt was made to identify the task scheduling algorithms used for distributed environments that perform well in the presence of faults due to network failure or processor failure in the distributed system. In this paper we give some extensive results for identifying the task scheduling algorithms that perform well in the presence of communication and computation faults present in some special task graphs like systolic array task graphs, Gaussian elimination task graphs, Fast Fourier transform task graphs, and divide-and-conquer task graphs which can represent the distributed software. For our study we have selected six task scheduling algorithms. We have compared these algorithms using three comparison parameters like normalized schedule length, number of processors used and average running time, and evaluated them on the above mentioned task graphs in the presence of communication and computation faults. It is further evaluated under random and constant fault delays**.

**Keywords**: Clustering, distributed computing, homogeneous systems, scheduling, task allocation, task scheduling

## 1. INTRODUCTION

Task scheduling is one of the important foundations for distributed computing. Distributed computing software can be represented as a task graph .Special task graphs have their own characteristics and nature. The tasks are allocated on distributed processors to exploit parallelism and to reduce the execution time of the distributed software. The communication and computation delays due to faults play a vital role in the execution of the task. Further if communication and computation faults are also considered which is more practical ,then only effective task execution in distributed environment may be achieved. In this paper four special types of task graphs are used: systolic array task graphs, Gaussian elimination task graphs, Fast Fourier transform task graphs, and divide-and-conquer task graphs. In all the task graphs, communication and computation fault delays are injected through emulators. The fault delays injected are further categorized as constant and random fault delays. An emulator gives the result very much similar to an actual system. The emulator is of a fully connected distributed system in which any two processors can directly communicate. Here homogeneous nodes have been considered.

The main objective of this experiment is to find out the task scheduling algorithm that best performs in the presence of communication fault delays and computation fault delays as well as to identify the algorithm that performs worst in the presence of above faults. For experimental purpose we have taken only six task scheduling algorithms out of many. Further, communication fault delays may be constant or random. Similarly computation fault delays may also be constant or random. The above faults are evaluated under the following three parameters: (i) normalized schedule length, (ii) average number of processors used and (iii) average running time. Using above parameters we identify the algorithms that perform best as well as those that perform worst in the presence of faults in the distributed system. This paper is organized as follows. Section 2 outlines the computation and communication fault delays which may be present in the distributed environments. Section 3 discusses the deferent task scheduling algorithms used for performance evaluation. In this paper we are considering only six task scheduling algorithms for experimental purpose. In section 4 deferent performance evaluation parameters used are discussed. Section 5 describes the special task graphs used. Section 6 shows the related work done in this area. Section 7 explains the experimental setup used in this work. Sections 8, 9, 10, and 11 show the performance results for systolic array task graphs, Gaussian elimination task graphs, fast Fourier transform task graphs and divide-and-conquer task graphs respectively. Section 12 summarizes the performance results and the future work to be done. Lastly section 13 gives the list of different references used in completing this paper.

## 2. Delays Due to Computation and Communication Faults in Distributed Systems

In a distributed system, some computing nodes may fail. To recover those computing node may take some time. So this will introduce a computation fault delay. We may also have a computation fault delay due to performance degradation due to some temporary hardware faults. Similarly, the links connecting the computing nodes may become faulty. So this will introduce a communication fault delay. In some of the cases we may have both computation fault delay and communication fault delay. In these situations we say that the distributed system is having computation and communication fault delay.

## 3. Task Scheduling Algorithms Used for Performance Evaluation

We have considered six task scheduling algorithms for performance evaluation [1]:

1. CPPS algorithm: The Cluster Pair Priority Scheduling [13] algorithm uses a cluster dependent priority function of tasks.

2. DCCL algorithm: The Dynamic Computation Communication Load Scheduling algorithm [14] is based on a computation and communication load of the module and current allocation.

3. DSC algorithm: The Dominant Sequence Clustering algorithm [15] is based on the critical path of the graph.

4. EZ algorithm: The Edge Zeroing algorithm [16] is used to minimize the communication delay. Based on the edge weight it selects tasks for merging.

5. LC algorithm: The Linear Clustering algorithm [17] is used to create clusters in a parallel system. It merges nodes iteratively to form a single cluster based on critical path.

6. RDCC algorithm: The Randomized Computation Communication Load Scheduling [1] algorithm is the dynamic priority version of the RCCL Randomized Computation Communication Load [18] scheduling algorithm.

## 4. Performance Evaluation Parameters Used

1. NSL : Normalized Schedule Length [1] is the schedule length over the sum
of computation cost on the critical path of the task graph.

$$NSL = SL / \sum_{v \in CP} w(v)$$

where SL is the schedule length and w(v) is the computation cost.

2. Average Number of Processor Used: It is the average of the number of processors used in computation of the task graph.

3. Average Running Time: It is the average of running time used in computing the task in the presence of computation fault, communication fault or both (computation and communication fault) delay.

## 5. A Description of Task Graphs Used

5.1. Systolic array task graphs

A systolic array effectively exploits parallelism. Systolic arrays have balanced, uniform, grid-like architecture in which each line represents a communication path and each intersection represents a systolic element. Technology advances, concurrent processing and demanding scientific applications have contributed a lot towards leading approach for handling computationally intensive applications. Figure 1 shows a sample systolic

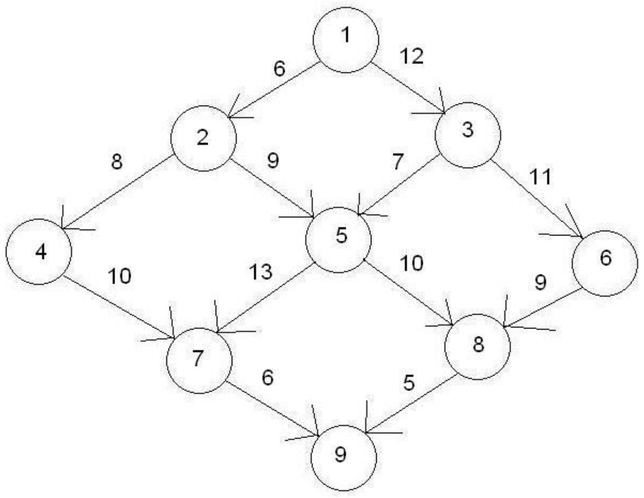

Figure 1: A sample systolic array task graph for n = 3.

array task graph [19]. A systolic array task graph has $n^2$ nodes and 2n(n - 1) edges where n is the number of nodes on a path from the start node to the center node. We have selected a total of 180 systolic array task graphs.

5.2. Gaussian elimination task graphs:

Gaussian elimination method is a well known graph theoretical model used in distributed architecture. Figure 2 shows a sample Gaussian elimination task graph [20]. A Gaussian elimination task graph has $(n^2 + n + 4)/2$ nodes and $n^2+1$ edges where n is the number of nodes on the task graph. We have selected a total of 180 random Gaussian elimination task graphs.

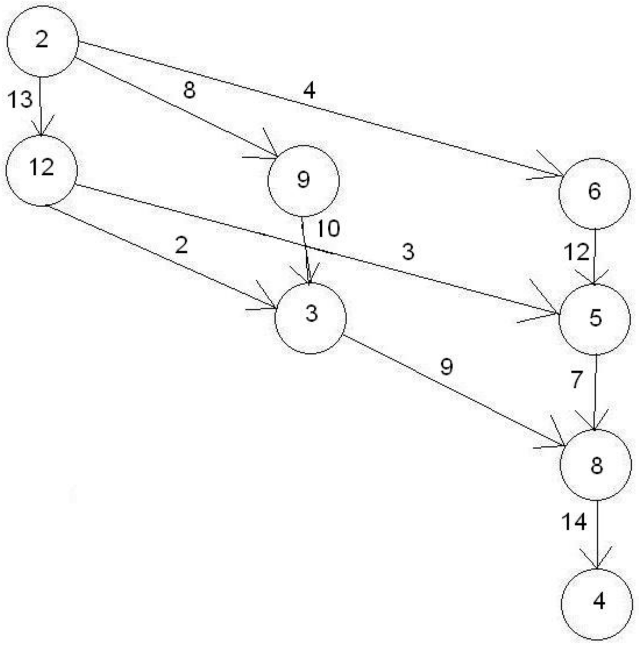

Figure 2: A sample Gaussian elimination task graph for n = 3.

5.3. Divide and conquer task graphs

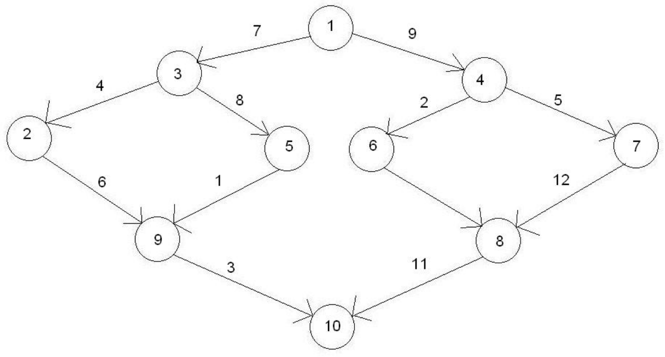

Figure 3: A sample divides and conquer task graph for n = 3.

Divide and conquer task graph is an important task graph based on multibranched recursion. A divide and conquer algorithm works by recursively breaking a problem into subproblems of the same type until it becomes simple to solve directly. The sub problem solutions are combined to solve the original problem. The correctness of this algorithm is proved by mathematical induction and the computational cost is determined by solving recurrence relations. Figure 3 shows a sample divide and conquer task graph [21]. A divide and conquer task graph has $3(2^{n-1}) - 2$ nodes and $2^{n+1} - 4$ edges where n is the number of nodes on a path from the start node to the middle level of the task graph. We have selected a total of 180 random divide and conquer task graphs.

5.4. Fast Fourier transform task graphs

Figure 4 shows a sample fast Fourier transform task graph [22]. A fast Fourier transform task graph has $2 + (n + 1)2^n$ nodes and $(n + 1)2^{n+1}$ edges where $2^n$ is the number of nodes on the second level of the task graph. We have selected a total of 180 fast Fourier transform task graphs.

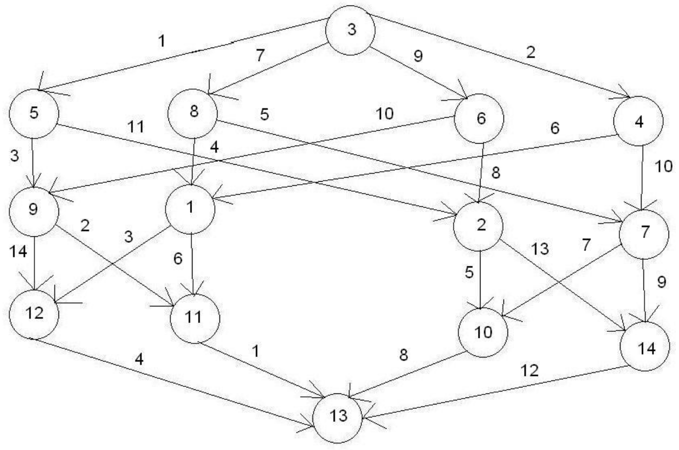

Figure 4: A sample Fast Fourier transform task graph for n = 2.

## 6. RELATED WORK

Goldsmith [3] worked on distributed computing and communication in peer to peer networks. Amoura [4] focused on scheduling algorithms for parallel Gaussian elimination with communication costs. Sinnen [5] elaborated the task scheduling for parallel systems. Mishra et al. focused on a clustering heuristic for multiprocessor environments using computation and communication loads of modules. Tobita [7] worked on a standard task graph set for fair evaluation of multiprocessor scheduling algorithms. Bertsekas [8] worked on parallel and distributed computation.

## 7. EXPERIMENTAL SETUP

The simulator Evaluate-Time is used to calculate the time taken by a given clustering [6]. Event queue model is used in which there is a queue of events. There can be two types of possible events: computation completion event, and communication completion event. Fault delays are added in the task graph before simulation. Two types of faults are simulated: constant delay, and random delay. The random delay is added using a random number generator.

## 8. Performance results of systolic array task graphs for Constant Communication and Computation Fault delay

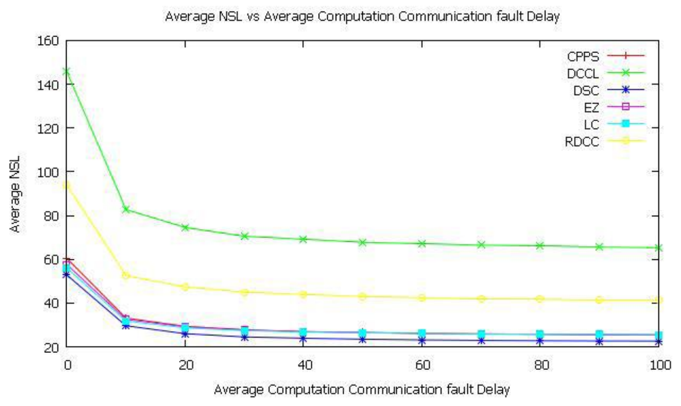

Figure 5: Average NSL vs average communication computation fault delay for constant communication computation fault delay. The average percentage variation order of NSL is: CPPS < DSC < RDCC < EZ < DCCL < LC.

Figure 5 shows the average NSL vs average communication computation fault delay for constant communication computation fault delay. Average percentage variation of NSL for CPPS ranges from -58.030445 to 0.000000 with an average of -49.974980. Average percentage variation of NSL for DCCL ranges from -55.207558 to 0.000000 with an average of -47.537540. Average percentage variation of NSL for DSC ranges from -57.284315 to 0.000000 with an average of -49.367017. Average percentage variation of NSL for EZ ranges from -55.684356 to 0.000000 with an average of -48.058303. Average percentage variation of NSL for LC ranges from -54.426600 to 0.000000 with an average of -47.141153. Average percentage variation of NSL for RDCC ranges from -56.075015 to 0.000000 with an average of -48.351840. The average percentage variation order of NSL is: CPPS < DSC < RDCC < EZ < DCCL < LC.

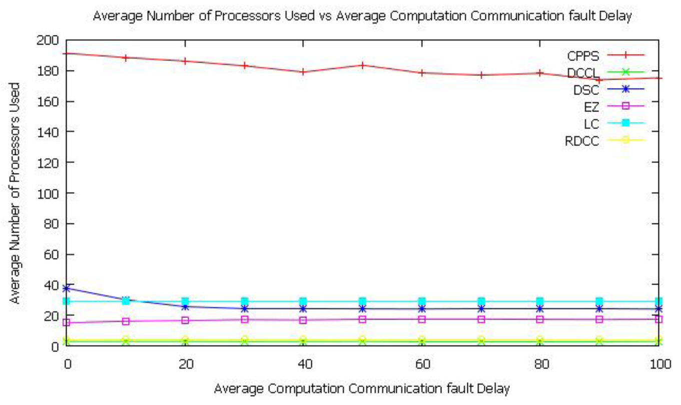

Figure 6: Average number of processors used vs average communication computation fault delay for constant communication computation fault delay. The average percentage variation order of average number of processors used is: DSC < CPPS < DCCL < LC < RDCC < EZ.

Figure 6 shows the average number of processors used vs average communication computation fault delay for constant communication computation fault delay. Average percentage variation of number of processors used by CPPS ranges from -9.072686 to 0.000000 with an average of -5.217329. Average percentage variation of number of processors used by DCCL ranges from -2.259887 to 1.694915 with an average of -0.821777. Average percentage variation of number of processors used by DSC ranges from -36.210713 to 0.000000 with an average of -30.830214. Average percentage variation of number of processors used by EZ ranges from 0.000000 to 15.929204 with an average of 12.007241. Average percentage variation of number of processors used by LC ranges from 0.000000 to 0.000000 with an average of 0.000000. Average percentage variation of number of processors used by RDCC ranges from 0.000000 to 4.329004 with an average of 2.085793. The average percentage variation order of average number of processors used is: DSC < CPPS < DCCL < LC < RDCC < EZ.

Figure 7 shows the average running time (in seconds) vs average communication computation fault delay for constant communication computation fault Delay .Average percentage variation of execution time for CPPS ranges from -2.608134 seconds to 58.335806 seconds with an average of 22.274306 seconds. Average percentage variation of execution time for DCCL ranges from -2.889328 seconds to 0.000000 seconds with an average of -2.198489 seconds. Average percentage variation of execution time for DSC ranges from -48.688208 seconds to 0.000000 seconds with an average of -31.002585 seconds. Average percentage variation of execution time for EZ ranges from -2.850187 seconds to 0.194850 seconds with an average of -1.524941 seconds. Average percentage variation of execution time for LC ranges from -31.023250 seconds to 52.328014 seconds with an average of 4.714710 seconds. Average percentage variation of execution time

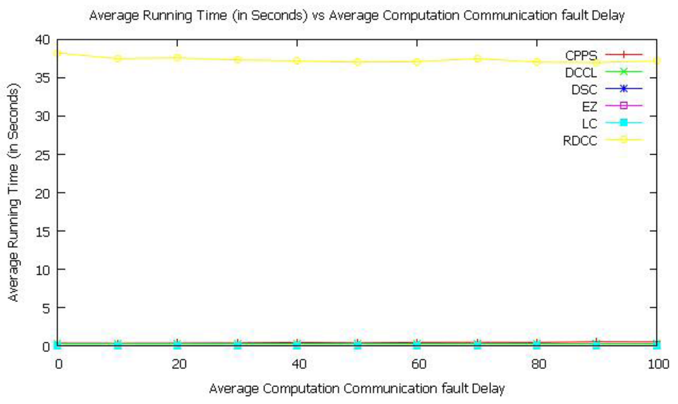

Figure 7: Average running time (in seconds) vs average communication computation fault delay for constant communication computation fault delay. The average percentage variation order of average running time (in seconds) is: DSC < RDCC < DCCL < EZ < LC <CPPS.

for RDCC ranges from -3.277439 seconds to 0.000000 seconds with an average of -2.332842 seconds. The average percentage variation order of average running time (in seconds) is: DSC < RDCC < DCCL < EZ < LC < CPPS.

### 8.1. Performance results of systolic array task graphs for Random Communication and Computation Fault delay

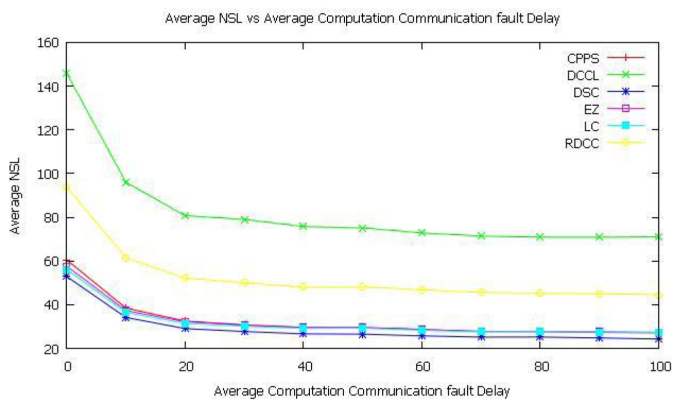

Figure 8: Average NSL vs average communication computation fault delay for random communication computation fault delay. The average percentage variation order of average NSL is: CPPS < DSC < EZ < RDCC < DCCL < LC.

Figure 8 shows the average NSL vs average communication computation fault delay for random communication computation fault delay. Average percentage variation of NSL for CPPS ranges from -55.211285 to 0.000000 with an average of -45.917632. Average percentage variation of NSL for DCCL ranges from -51.413559 to 0.000000 with an average of -43.293403. Average percentage variation of NSL for DSC ranges from -54.121460 to 0.000000 with an average of -44.642742. Average percentage variation of NSL for EZ ranges from -52.520573 to 0.000000 with an average of -43.860976. Average percentage variation of NSL for LC ranges from -51.417506 to 0.000000 with an average of -43.151901. Average percentage variation of NSL for RDCC ranges from -52.539658 to 0.000000 with an average of -43.726319. The average percentage variation order of average NSL is: CPPS < DSC < EZ < RDCC < DCCL < LC.

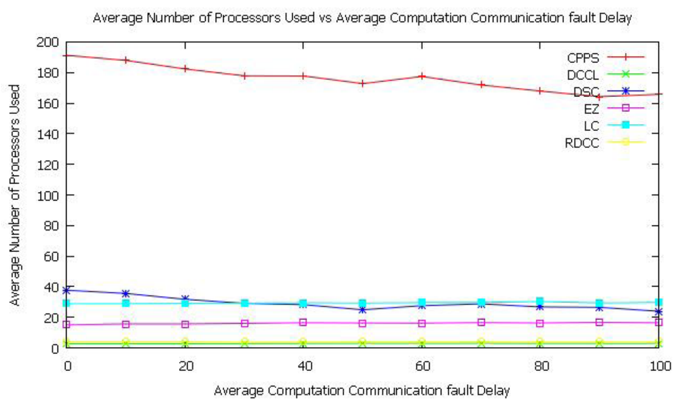

Figure 9: Average number of processors used vs average communication computation fault delay for random communication computation fault delay. The average percentage variation order of average number of processors used is: DSC < CPPS < RDCC < DCCL < LC <EZ.

Figure 9 shows the average number of processors used vs average communication computation fault delay for random communication computation fault delay. Average percentage variation of number of processors used by CPPS ranges from -14.145024 to 0.000000 with an average of -7.937313. Average percentage variation of number of processors used by DCCL ranges from -3.389831 to 3.389831 with an average of 0.564972. Average percentage variation of number of processors used by DSC ranges from -36.520584 to 0.000000 with an average of -22.580386. Average percentage variation of number of processors used by EZ ranges from 0.000000 to 10.840708 with an average of 6.818182.
Average percentage variation of number of processors used by LC ranges from 0.000000 to 4.852686 with an average of 1.874902. Average percentage variation of number of processors used by RDCC ranges from -2.953586 to 0.843882 with an average of -0.652091. The average percentage variation order of average number of processors used is: DSC < CPPS < RDCC < DCCL < LC < EZ.

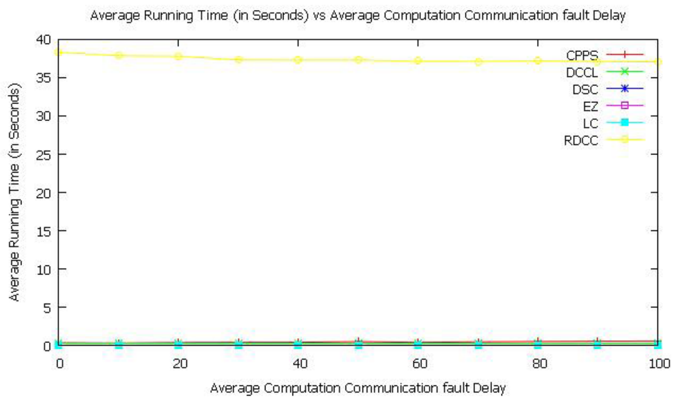

Figure 10: Average running time (in seconds) vs average communication computation fault delay for random communication computation fault delay. The average percentage variation order of running time (in seconds) is: RDCC < DCCL < EZ < CPPS < LC < DSC.

Figure 10 shows the average running time (in seconds) vs average communication computation fault delay for random communication computation fault delay. Average percentage variation of execution time for CPPS ranges from -11.247626 seconds to 67.992507 seconds with an average of 26.901226 seconds. Average percentage variation of execution time for DCCL ranges from -2.961047 seconds to 0.000000 seconds with an average of -2.012974 seconds. Average percentage variation of execution time for DSC ranges from 0.000000 seconds to 70.153580 seconds with an average of 30.707585 seconds. Average percentage variation of execution time for EZ ranges from -1.596928 seconds to 1.202561 seconds with an average of -0.475513 seconds. Average percentage variation of execution time for LC ranges from -10.500741 seconds to 104.569735 seconds with an average of 30.223975 seconds. Average percentage variation of execution time for RDCC ranges from -3.286028 seconds to 0.000000 seconds with an average of -2.365309 seconds. The average percentage variation order of running time (in seconds) is: RDCC < DCCL < EZ < CPPS < LC < DSC.

## 9. PERFORMANCE RESULTS OF GAUSSIAN ELIMINATION TASK GRAPHS FOR CONSTANT COMMUNICATION AND COMPUTATION FAULT DELAY

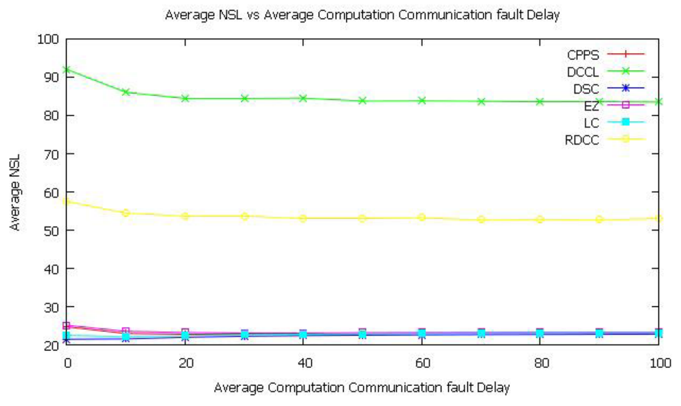

Figure 11: Average NSL vs average computation communication fault delay for constant computation communication fault delay. The average percentage variation order of NSL is: DCCL < RDCC < EZ < CPPS < LC < DSC.

Figure 11 shows the average NSL vs average computation communication fault delay for constant computation communication fault delay. Average percentage variation of NSL for CPPS ranges from -7.551641 to 0.000000 with an average of -6.467052. Average percentage variation of NSL for DCCL ranges from -9.288952 to 0.000000 with an average of -7.817920. Average percentage variation of NSL for DSC ranges from 0.000000 to 6.217408 with an average of 4.101296. Average percentage variation of NSL for EZ ranges from -7.750759 to 0.000000 with an average of -6.664127. Average percentage variation of NSL for LC ranges from -1.310508 to 2.916676 with an average of 1.468501. Average percentage variation of NSL for RDCC ranges from -8.418240 to 0.000000 with an average of -6.766760. The average percentage variation order of NSL is: DCCL <RDCC < EZ < CPPS < LC < DSC.

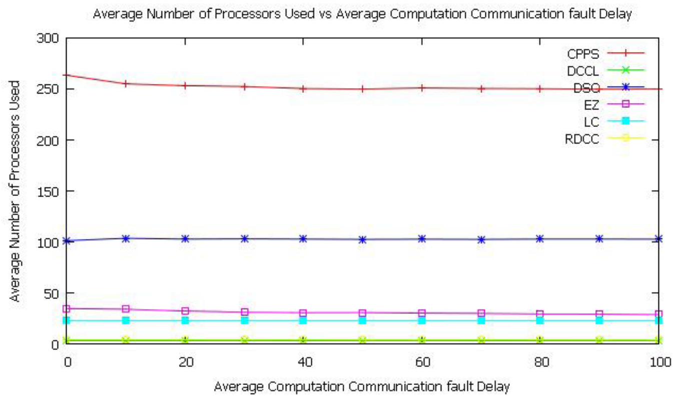

Figure 12: Average number of processors used vs average computation communication fault delay for constant computation communication fault delay. The average percentage variation order of number of processors used is: EZ < CPPS < RDCC < LC < DCCL < DSC.

Figure 12 shows the average number of processors used vs average computation communication fault delay for constant computation communication fault delay. Average percentage variation of number of processors used by CPPS ranges from -5.170886 to 0.000000 with an average of -4.199079. Average percentage variation of number of processors used by DCCL ranges from -0.520833 to 2.604167 with an average of 1.373106. Average percentage variation of number of processors used by DSC ranges from 0.000000 to 2.255887 with an average of 1.386165. Average percentage variation of number of processors used by EZ ranges from -17.028461 to 0.000000 with an average of -10.577556. Average percentage variation of number of processors used by LC ranges from 0.000000 to 0.000000 with an average of 0.000000. Average percentage variation of number of processors used by RDCC ranges from -2.419355 to 2.016129 with an average of -0.733138. The average percentage variation order of number of processors used is: EZ < CPPS < RDCC < LC < DCCL < DSC.

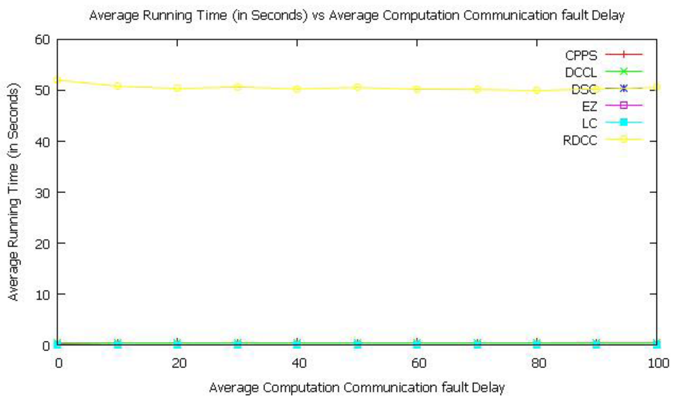

Figure 13: Average running time (in seconds) vs average computation communication fault delay for constant computation communication fault delay. The average percentage variation order of running time (in seconds) is: EZ < RDCC < DSC < DCCL < LC < CPPS.

Figure 13 shows the average running time (in seconds) vs average computation communication fault delay for constant computation communication fault delay.
Average percentage variation of execution time for CPPS ranges from 0.000000 seconds to 48.137222 seconds with an average of 37.903072 seconds. Average percentage variation of execution time for DCCL ranges from -1.638895 seconds to 0.000000 seconds with an average of -1.173872 seconds. Average percentage variation of execution time for DSC ranges from -24.181258 seconds to 20.886958 seconds with an average of -2.435070 seconds. Average percentage variation of execution time for EZ ranges from -12.167287 seconds to 0.000000 seconds with an average of -9.328306 seconds. Average percentage variation of execution time for LC ranges from -28.101012 seconds to 65.212792 seconds with an average of 9.314070 seconds. Average percentage variation of execution time for RDCC ranges from -3.894122 seconds to 0.000000 seconds with an average of -2.825527 seconds. The average percentage variation order of running time (in seconds) is: EZ < RDCC < DSC < DCCL < LC < CPPS.

### 9.1. Performance results of Gaussian elimination task graphs for Random Communication and Computation Fault delay

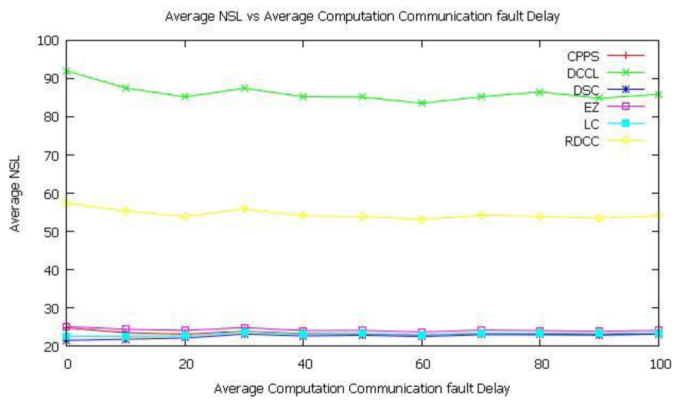

Figure 14: Average NSL vs average communication computation fault delay for random communication computation fault delay. The average percentage variation order of NSL is: DCCL < CPPS < RDCC < EZ < LC < DSC.

Figure 14 shows the average NSL vs average communication computation fault delay for random communication computation fault delay. Average percentage variation of NSL for CPPS ranges from -7.555230 to 0.000000 with an average of -5.549895. Average percentage variation of NSL for DCCL ranges from -9.230700 to 0.000000 with an average of -6.295551. Average percentage variation of NSL for DSC ranges from 0.000000 to 7.668250 with an average of 5.114242. Average percentage variation of NSL for EZ ranges from -5.819980 to 0.000000 with an average of -3.682232. Average percentage variation of NSL for LC ranges from 0.000000 to 5.402467 with an average of 2.803627. Average percentage variation of NSL for RDCC ranges from -7.401646 to 0.000000 with an average of -5.108383. The average percentage variation order of NSL is: DCCL < CPPS < RDCC < EZ < LC < DSC.

Figure 15 shows the average number of processors used vs average communication computation fault delay for random communication computation fault delay. Average percentage variation of number of processors used by CPPS ranges from -12.044304 to 0.000000

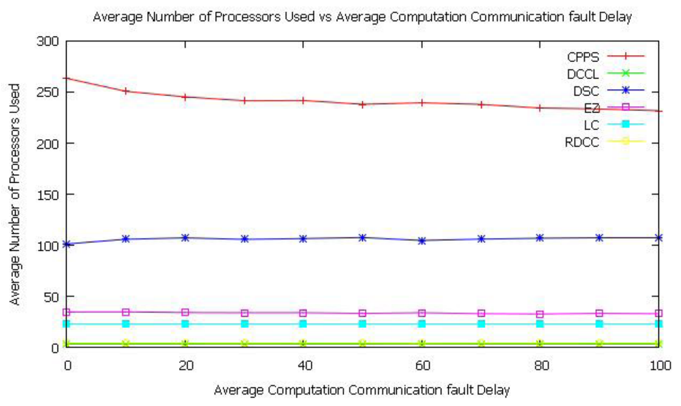

Figure 15: Average number of processors used vs average communication computation fault delay for random communication computation fault delay. The average percentage variation order of number of processors used is: CPPS < EZ < LC < DCCL < RDCC < DSC.

with an average of -8.280783. Average percentage variation of number of processors used by DCCL ranges from -1.041667 to 1.562500 with an average of 0.094697. Average percentage variation of number of processors used by DSC ranges from 0.000000 to 6.421867 with an average of 4.772241. Average percentage variation of number of processors used by EZ ranges from -5.595755 to 0.289436 with an average of -2.784721. Average per centage variation of number of processors used by LC ranges from 0.000000 to 0.000000 with an average of 0.000000. Average percentage variation of number of processors used by RDCC ranges from -0.823045 to 2.880658 with an average of 0.785634. The average percentage variation order of number of processors used is: CPPS < EZ < LC < DCCL < RDCC < DSC.

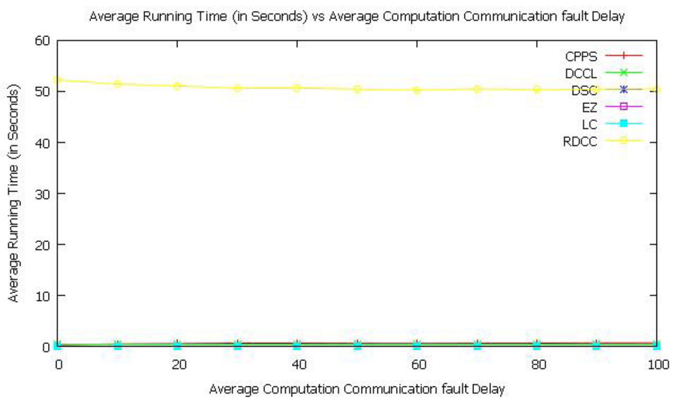

Figure 16: Average running time (in seconds) vs average communication computation fault delay for random communication computation fault delay. The average percentage variation order of running time (in seconds) is: EZ < DSC < RDCC < DCCL < LC < CPPS.

Figure 16 shows the average running time (in seconds) vs average communication computation fault delay for random communication computation fault delay. Average percentage variation of execution time for CPPS ranges from 0.000000 seconds to 122.850023 seconds with an average of 83.225975 seconds. Average percentage variation of execution time for DCCL ranges from -1.085166 seconds to 0.148587 seconds with an average of -0.305673 seconds. Average percentage variation of execution time for DSC ranges from -21.684314 seconds to 27.556533 seconds with an average of -5.885352 seconds. Average percentage variation of execution time for EZ ranges from -8.545753 seconds to 0.000000 seconds with an average of -6.639976 seconds. Average percentage variation of execution time for LC ranges from -3.488695 seconds to 115.876492 seconds with an average of 29.790331 seconds. Average percentage variation of execution time for RDCC ranges from -3.754037 seconds to 0.000000 seconds with an average of -2.756291 seconds. The average percentage variation order of running time (in seconds) is: EZ < DSC < RDCC < DCCL < LC < CPPS.

## 10. PERFORMANCE RESULTS OF FAST FOURIER TRANSFORM TASK GRAPHS FOR CONSTANT COMMUNICATION AND COMPUTATION FAULT DELAY

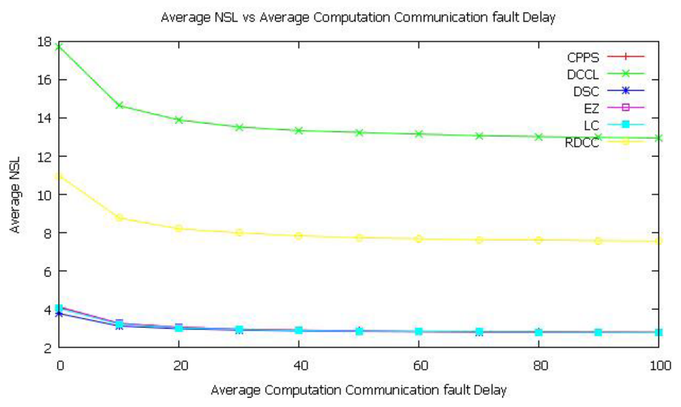

Figure 17: Average NSL vs average computation communication fault delay for constant computation communication fault delay. The average percentage variation order of NSL is: CPPS < EZ < RDCC < LC < DCCL < DSC.

Figure 17 shows the average NSL vs average computation communication fault delay for constant computation communication fault delay. Average percentage variation of NSL for CPPS ranges from -32.033450 to 0.000000 with an average of -26.582122. Average percentage variation of NSL for DCCL ranges from -26.893349 to 0.000000 with an average of -22.238283. Average percentage variation of NSL for DSC ranges from -26.227048 to 0.000000 with an average of -21.716524. Average percentage variation of NSL for EZ ranges from -31.734216 to 0.000000 with an average of -26.208394. Average percentage variation of NSL for LC ranges from -30.773233 to 0.000000 with an average of -25.583130. Average percentage variation of NSL for RDCC ranges from -31.017114 to 0.000000 with an average of -25.777003.The average percentage variation order of NSL is: CPPS < EZ < RDCC < LC < DCCL < DSC.

Figure 18 shows the average numbeer of processors used vs average computation communication fault delay for constant computation communication fault delay. Average percentage variation of number of processors used by CPPS ranges from -0.039888 to 0.026592 with an average of -0.030218. Average percentage variation of number of processors used by DCCL ranges from -1.315789 to 0.000000 with an average of -0.538278.

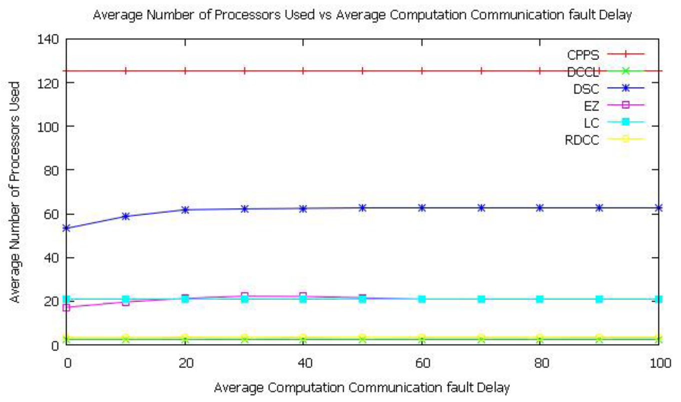

Figure 18: Average number of processors used vs average computation communication fault delay for constant computation fault delay. The average percentage variation order of number of processors used is: DCCL < CPPS < LC < RDCC < DSC < EZ.

Average percentage variation of number of processors used by DSC ranges from 0.000000 to 17.281250 with an average of 14.872159. Average percentage variation of number of processors used by EZ ranges from 0.000000 to 29.525653 with an average of 20.575552. Average percentage variation of number of processors used by LC ranges from 0.000000 to 0.000000 with an average of 0.000000. Average percentage variation of number of processors used by RDCC ranges from -1.507538 to 9.045226 with an average of 4.613979.The average percentage variation order of number of processors used is: DCCL < CPPS < LC < RDCC < DSC < EZ.

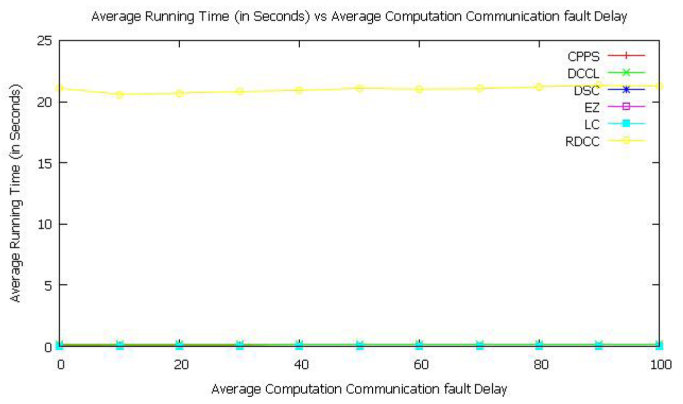

Figure 19: Average running time (in seconds) vs average computation communication fault delay for constant computation communication fault delay. The average percentage variation order of running time (in seconds) is: EZ < CPPS < LC < DCCL < RDCC < DSC.

Figure 19 shows the average running time (in seconds) vs average computation communication fault delay for constant computation communication fault delay. Average percentage variation of execution time for CPPS ranges from -10.051132 seconds to 0.000000 seconds with an average of -8.636224 seconds. Average percentage variation of execution time for DCCL ranges from -1.249324 seconds to 0.008873 seconds with an average of -0.612947 seconds. Average percentage variation of execution time for DSC ranges from -7.625609 seconds to 65.342904 seconds with an average of 17.515968 seconds. Average percentage variation of execution time for EZ ranges from -18.512653 seconds to 0.000000 seconds with an average of -13.092788 seconds. Average percentage variation of execution time for LC ranges from -18.979249 seconds to 15.350542 seconds with an average of -4.204960 seconds. Average percentage variation of execution time for RDCC ranges from -2.341522 seconds to 1.138953 seconds with an average of -0.386761 seconds. The average percentage variation order of running time (in seconds) is: EZ < CPPS < LC < DCCL < RDCC < DSC.

## 10.1. Performance results of fast fourier transform task graphs for Random Communication and Computation Fault delay

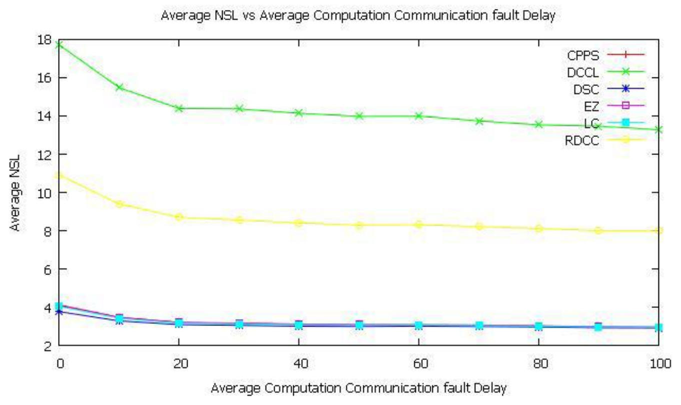

Figure 20: Average NSL vs average computation communication fault delay for random computation communication fault delay. The average percentage variation order of NSL is: CPPS < EZ < LC < RDCC < DCCL < DSC.

Figure 20 shows the average NSL vs average computation communication fault delay for random computation communication fault delay. Average percentage variation of NSL for CPPS ranges from -28.286067 to 0.000000 with an average of -22.274371. Average percentage variation of NSL for DCCL ranges from -25.033410 to 0.000000 with an average of -18.874905. Average percentage variation of NSL for DSC ranges from -22.652459 to 0.000000 with an average of -18.055520. Average percentage variation of NSL for EZ ranges from -27.312959 to 0.000000 with an average of -21.601798. Average percentage variation of NSL for LC ranges from -26.683568 to 0.000000 with an average of -21.338724. Average percentage variation of NSL for RDCC ranges from -26.709171 to 0.000000 with an average of -20.899349.The average percentage variation order of NSL is: CPPS < EZ < LC < RDCC < DCCL < DSC.

Figure 21 shows the average number of processors used vs average computation communication fault delay for random computation communication fault delay. Average percentage variation of number of processors used by CPPS ranges from -2.845366 to 0.000000 with an average of -1.037096. Average percentage variation of number of processors used by DCCL ranges from 0.000000 to 5.263158 with an average of 2.392344. Average percentage variation of number of processors used by DSC ranges from 0.000000 to 15.781250 with an average of 12.403409.Average percentage variation of number of processors used by EZ ranges from 0.000000 to 32.333011 with an average of 21.675614. Average percentage variation of number of processors used by LC ranges from 0.000000 to 0.000000 with an average of 0.000000. Average percentage variation of number of processors used by RDCC ranges from 0.000000 to 13.297872 with an average of 8.945841.The average percentage variation order of number of processors used is: CPPS < LC < DCCL < RDCC < DSC < EZ.

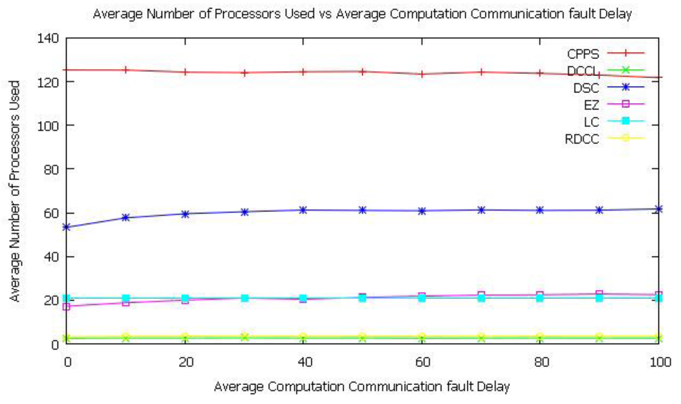

Figure 21: Average number of processors used vs average computation communication fault delay for random computation communication fault delay. The average percentage variation order of number of processors used is: CPPS < LC < DCCL < RDCC < DSC < EZ.

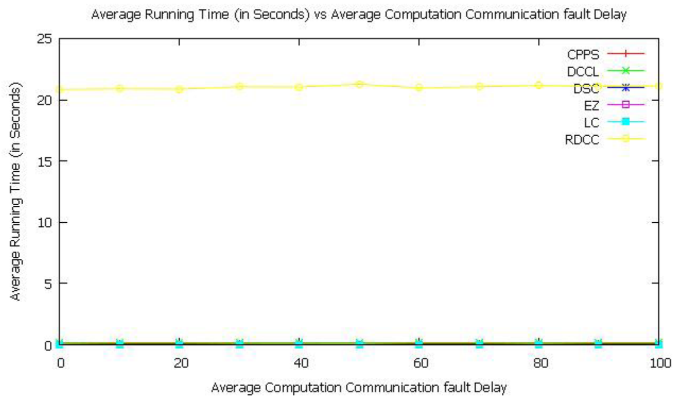

Figure 22: Average running time (in seconds) vs average computation communication fault delay for random computation communication fault delay. The average percentage variation order of running time (in seconds) is: LC < EZ < DCCL < RDCC < DSC < CPPS.

Figure 22 shows the average running time (in seconds) vs average computation communication fault delay for random computation communication fault delay .Average percentage variation of execution time for CPPS ranges from -8.925985 seconds to 25.984653 seconds with an average of 9.314486 seconds. Average percentage variation of execution time for DCCL ranges from -1.149733 seconds to 0.566031 seconds with an average of -0.390620 seconds. Average percentage variation of execution time for DSC ranges from -15.451331 seconds to 39.671315 seconds with an average of 3.396834 seconds. Average percentage variation of execution time for EZ ranges from -11.858071 seconds to 0.000000 seconds with an average of -7.245864 seconds. Average percentage variation of execution time for LC ranges from -38.596667 seconds to 6.926753 seconds with an average of -23.268047 seconds. Average percentage variation of execution time for RDCC ranges from 0.000000 seconds to 1.996918 seconds with an average of 0.891951 seconds. The average percentage variation order of running time (in seconds) is: LC < EZ < DCCL < RDCC < DSC < CPPS.

## 11. Performance results of Divide-and-conquer task graphs for Constant Communication and Computation Fault delay

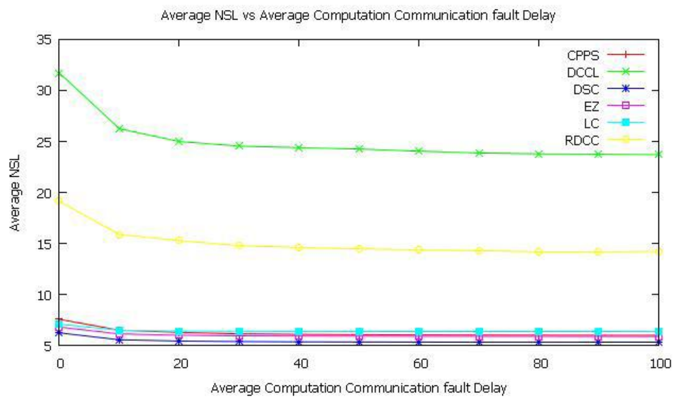

Figure 23: Average NSL vs average computation communication fault delay for constant computation communication fault delay. The average percentage variation order of NSL is: RDCC < DCCL < CPPS < DSC < EZ < LC.

Figure 23 shows the average NSL vs average computation communication fault delay for constant computation communication fault delay. Average percentage variation of NSL for CPPS ranges from -20.754456 to 0.000000 with an average of -17.467447. Average percentage variation of NSL for DCCL ranges from -25.275453 to 0.000000 with an average of -21.102228. Average percentage variation of NSL for DSC ranges from -14.943757 to 0.000000 with an average of -12.740658. Average percentage variation of NSL for EZ ranges from -13.478138 to 0.000000 with an average of -11.511678. Average percentage variation of NSL for LC ranges from -10.216266 to 0.000000 with an average of -8.942914. Average percentage variation of NSL for RDCC ranges from -25.906343 to 0.000000 with an average of -21.398239.The average percentage variation order of NSL is: RDCC < DCCL < CPPS < DSC < EZ < LC.

Figure 24 shows the average number of processors used vs average computation communication fault delay for constant computation communication fault delay. Average percentage variation of number of processors used by CPPS ranges from -3.375887 to 0.000000 with an average of -2.737589. Average percentage variation of number of processors used by DCCL ranges from -1.273885 to 1.910828 with an average of 0.173712. Average percentage variation of number of processors used by DSC ranges from 0.000000 to 0.323027 with an average of 0.230734. Average percentage variation of number of processors used by EZ ranges from -3.052632 to 0.000000 with an average of -2.258373. Average percentage variation of number of processors used by LC ranges from 0.000000 to 0.000000 with an average

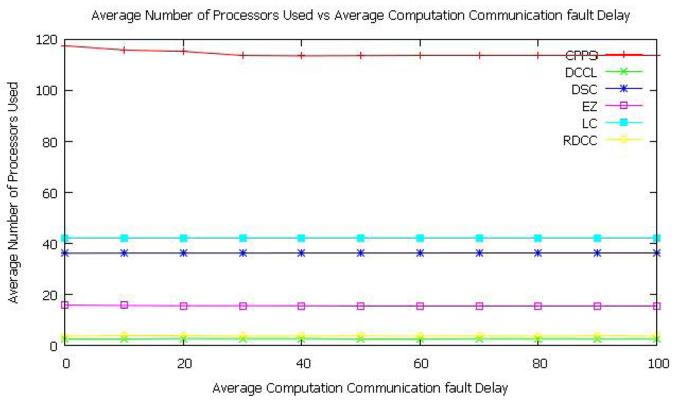

Figure 24: Average number of processors used vs average computation communication fault delay for constant computation communication fault delay. The average percentage variation order of number of processors used is: CPPS < EZ < LC < DCCL < DSC < RDCC.

of 0.000000. Average percentage variation of number of processors used by RDCC ranges from 0.000000 to 4.739336 with an average of 2.455838. The average percentage variation order of number of processors used is: CPPS < EZ < LC < DCCL < DSC < RDCC.

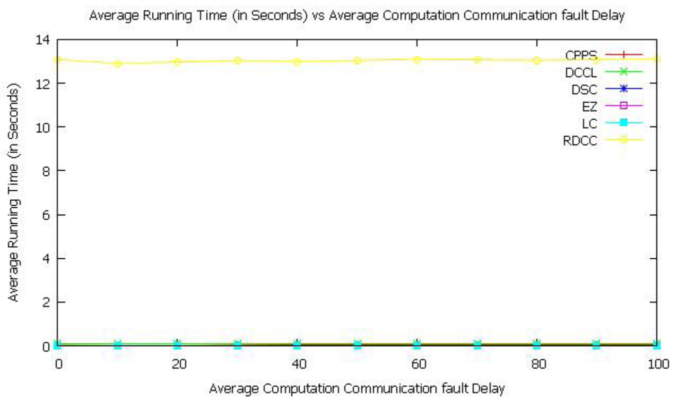

Figure 25: Average running time (in seconds) vs average computation communication fault delay for constant computation communication fault delay. The average percentage variation order of running time (in seconds) is: LC < DCCL < RDCC < EZ < DSC < CPPS.

Figure 25 shows the average running time (in seconds) vs average computation communication fault delay for constant computation communication fault delay. Average percentage variation of execution time for CPPS ranges from 0.000000 seconds to 65.871142 seconds with an average of 49.417149 seconds. Average percentage variation of execution time for DCCL ranges from -1.414007 seconds to 0.000000 seconds with an average of -0.756574 seconds. Average percentage variation of execution time for DSC ranges from -11.784990 seconds to 64.080460 seconds with an average of 4.906571 seconds. Average percentage variation of execution time for EZ ranges from -4.588138 seconds to 4.629644 seconds with an average of 0.450249 seconds. Average percentage variation of execution time for LC ranges from -21.386323 seconds to 8.187810 seconds with an average of -11.221015 seconds. Average percentage variation of execution time for RDCC ranges from -1.383551 seconds to 0.297899 seconds with an average of -0.331793 seconds. The average percentage variation order of running time (in seconds) is: LC < DCCL < RDCC < EZ < DSC < CPPS.

## 11.1. Performance results of divide-and-conquer task graphs for Random Communication and Computation Fault delay

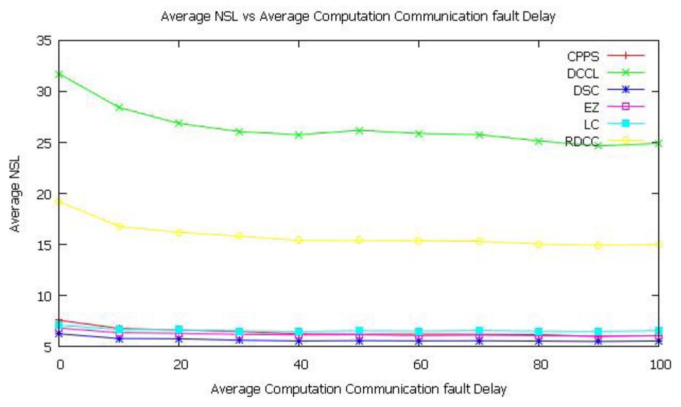

Figure 26: Average NSL vs average computation communication fault delay for random computation communication fault delay. The average percentage variation order of NSL is: RDCC < DCCL < CPPS < DSC < LC < EZ.

Figure 26 shows the average NSL vs average computation communication fault delay for random computation communication fault delay. Average percentage variation of NSL for CPPS ranges from -20.834341 to 0.000000 with an average of -15.424460. Average percentage variation of NSL for DCCL ranges from -22.242009 to 0.000000 with an average of -16.487258. Average percentage variation of NSL for DSC ranges from -12.059344 to 0.000000 with an average of -9.425610. Average percentage variation of NSL for EZ ranges from -11.636598 to 0.000000 with an average of -8.764203. Average percentage variation of NSL for LC ranges from -8.724656 to 0.000000 with an average of -6.965674. Average percentage variation of NSL for RDCC ranges from -22.043911 to 0.000000 with an average of -17.342865.The average percentage variation order of NSL is: RDCC < DCCL < CPPS < DSC < LC < EZ.

Figure 27 shows the average number of processors used vs average computation communication fault delay for random computation communication fault delay. Average percentage variation of number of processors used by CPPS ranges from -10.879433 to 0.000000 with an average of -6.308188. Average percentage variation of number of processors used by DCCL ranges from -5.732484 to 0.000000 with an average of -2.084540. Average percentage variation of number of processors used by DSC ranges from -0.184587 to 1.199815 with an average of 0.469858. Average percentage variation of number of processors used by EZ ranges from -0.842105 to 3.789474 with an average of 1.157895. Average percentage variation of number of processors used by LC ranges from 0.000000 to 0.000000 with an average of 0.000000. Average percentage variation of number of processors used by RDCC ranges from -2.392344 to 7.655502 with an average of 3.827751.

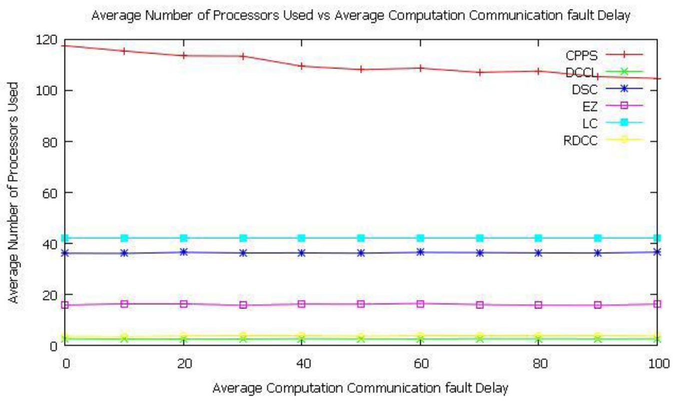

Figure 27: Average number of processors used vs average computation communication fault delay for random computation communication fault delay. The average percentage variation order of number of processors used is: CPPS < DCCL < LC < DSC < EZ < RDCC.

The average percentage variation order of number of processors used is: CPPS < DCCL < LC < DSC < EZ < RDCC.

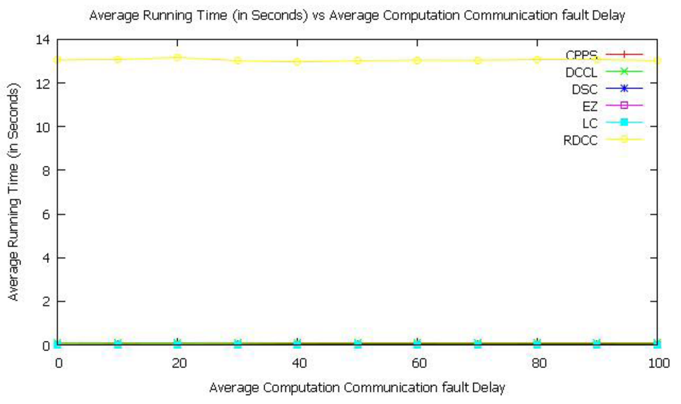

Figure 28: Average running time (in seconds) vs average computation communication fault delay for random computation communication fault delay. The average percentage variation order of running time (in seconds) is: LC < DSC < EZ < DCCL < RDCC < CPPS.

Figure 28 shows the average running time (in seconds) vs average computation communication fault delay for random computation communication fault delay. Average percentage variation of execution time for CPPS ranges from 0.000000 seconds to 158.861740 seconds with an average of 93.582661 seconds. Average percentage variation of execution time for DCCL ranges from -2.092929 seconds to 0.233569 seconds with an average of -1.146830 seconds. Average percentage variation of execution time for DSC ranges from -34.176645 seconds to 11.433718 seconds with an average of -15.800717 seconds. Average percentage variation of execution time for EZ ranges from -6.252946 seconds to 0.352861 seconds with an average of -2.470733 seconds. Average percentage variation of execution time for LC ranges from -48.319187 seconds to 0.000000 seconds with an average of -33.140748 seconds. Average percentage variation of execution time for RDCC ranges from -0.665362 seconds to 0.900635 seconds with an average of 0.001368 seconds. The average percentage variation order of running time (in seconds) is: LC < DSC < EZ < DCCL < RDCC < CPPS.

## 12. CONCLUSION

In this paper we performed the experiments on task scheduling of special task graphs to identify its behavior in the presence of communication and computation fault delays for distributed environment. We evaluated six algorithms namely CPPS, DCCL, DSC, EZ, LC and RDCC for four types of task graphs (systolic array, Gaussian elimination, fast Fourier transform and divide-and-conquer) using three types of comparisons (average NSL vs average computation and communication fault delay, average number of processor used vs average computation and communication fault delay and average running time (in second) vs average computation and communication fault delay). Each task graph is further evaluated under two category (i) task graphs with random fault delay and (ii) task graphs with constant fault delay.
From the above graphs and results it can be concluded that in terms of average running time RDCC algorithm gives the best result in systolic array whereas EZ algorithm gives best result in Gaussian elimination and fast Fourier transform task graphs and LC algorithm gives best result in divide-and-conquer task graph. In terms of number of processor used DSC algorithm gives better performance in systolic array and Gaussian elimination task graph and CPPS algorithm gives better performance in fast Fourier transform and divide-and-conquer task graphs. For future work we can consider random faults with deferent types of probability distributions like normal distribution, Poisson distribution, Bernoulli distribution, etc.

## 13. REFERENCES

[1] Anil Kumar Tripathi, P.K. Mishra, Abhishek Mishra,Kamal sheel Mishra, Benchmarking the clustering algorithms for multiprocessor environments
using dynamic priority of modules, Elsevier Applied Mathematical Modelling 36 (2012) 6243-6263.

[2] Kamal Sheel Mishra, Anil Kumar Tripathi, Task Sheduling of a Distributed Computing Software in the Presence of Faults, International journal of Computer Applications Vol 72 (2013)No.13 0975 8887.

[3] Bradley Charles Goldsmith, distributed Computing and Communication in Peer-to-peer Network,University of Tasmania (2010)Bertsekas89.

[4] Amoura A.K., Scheduling algorithms for parallel Gaussian elimination with communication costs, IEEE Transactions ,Vol-9 issue:7 (1998).

[5] O sinnen, Task scheduling for parallel systems, Wiley-Interscience (2007).

[6] P.K. Mishra, K.S. Mishra, A. Mishra, A clustering heuristic for multiprocessor environments using computation and communication loads of modules, International Journal of Computer Science & Information Technology, 2(5):170{182, 2010.

[7] T.Tobita,A standard task graph set for fail evaluation of multiprocessor scheduling algorithm,J.Sched. 5 (2002) 379-394.

[8] D.P. Bertsekas, Parallel and Distributed Computation-Numerical methods,Athena Sc., (1989).

[9] Y. K. Kwok, I. Ahmad, Benchmarking and comparison of the task graph scheduling algorithms, Journal of Parallel and Distributed Computing 59 (1999) 381{422.

[10] john A. Stankovic, K. Ramamritham, S.Cheng, Evaluation of a Flexible task scheduling algorithm for distributed hard real time systems, IEEE Transactions on Computers Vol c-34 , no. 12 (1985) 1130{1143.

[11] V.S. Tondre, V.M.Thakare, S.S.Sherekar, R.V. Dharaskar, Technical computation and communication delay in distributed system, NCICT (2011) IJCA.

[12] R.C.Nunes,I.J. Porto, Modeling communication delays in distributed systems using time series, IEEE transactions (2002) 1060-9857/02, Brazil.

[13] A. Mishra, A.K. Tripathi, An extension of edge zeroing heuristic for scheduling precedence constrained task graphs on parallel systems using cluster dependent priority scheme, J. Inform. Comput. Sci. 6 (2011) 83{96.
An extended abstract of this paper appears in the Proceedings of IEEE International Conference on Computer and Communication Technology, 2010, pp. 647{651.

[14] P.K. Mishra, K.S. Mishra, A. Mishra, A clustering algorithm for multiprocessor environments using dynamic priority of modules, Ann. Math. Inform. 38 (2011) 99{110.

[15] T. Yang, A. Gerasoulis, A fast static scheduling algorithm for DAGs on an unbounded number of processors, in: Proceedings of the 1991 ACM/IEEE Conference on Supercomputing, 1991, pp. 633{642.

[16] V. Sarkar, Partitioning and scheduling parallel programs for multiprocessors, Research Monographs in Parallel and Distributed Computing, MIT Press, 1989.
[17] S.J. Kim, J.C. Browne, A general approach to mapping of parallel computation upon multiprocessor architectures, in: Proceedings of 1988 International Conference on Parallel Processing, 3, 1988, pp. 18.

[18] Mishra, P.K., Mishra, K.S., Mishra, A. and Tripathi, A.K., A Randomized Scheduling Algorithm for Multiprocessor Environments, Parallel Processing Letters, Vol. Vol 22 No 4, pp 125005, 2012 , World Scienti_c.

[19] O.H. Ibarra, S.M. Sohn, On mapping systolic algorithms onto the hypercube, IEEE Trans. Parallel Distrib. Syst. 1 (1990) 4863.

[20] D.P. Bertsekas, J.N. Tsitsiklis, Parallel and Distributed Computation Numerical Methods, Athena Sci. (1989).

[21] S. Madal, J.B. Sinclair, Performance of synchronous parallel algorithms with regular structures, IEEE Trans. Parallel Distrib. Syst. 2 (1991) 105116.

[22] J.P. Kitajima, Modles Quantitatifs dAlgorithmes Parallles, Doctorate the sis, Institut National Ploytechnique de Grenoble, 1992.

## AUTHOR'S PROFILE

**Anil Kumar Tripathi** is Professor of Computer Engineering at Indian Institute of Technology (Banaras Hindu University), Varanasi, India. He received his Ph.D. degree in Computer Engineering from Banaras Hindu University; and M.Sc. Engg. (Computer) degree from Odessa National Polytechnic University, Ukraine. His research interests include parallel and distributing computing, and software engineering. He has to his credit more than 60 research papers in International journals. He has co-authored two research monographs: one from Springer USA and other from John Wiley USA .More than Fifteen students have completed their Ph.D under his supervision.

**Kamal Sheel Mishra** is M.Tech (Computer Engg.) and working as Associate Professor and Dean ,Computer Science department in the School of Management Sciences , Varanasi, India. He is having more than 18 years of teaching experience. His research interests include Software engineering, Parallel and Distributed Computing. Currently he is Pursuing Ph.D. from Department of Computer Science and Engineering, Indian Institute of Technology (Banaras Hindu University), Varanasi, India.