CERTIFICATE

It is certified that the work contained in this thesis entitled "SOME OBSERVATIONS ON TASK SCHEDULING ALGORITHMS FOR DISTRIBUTED COMPUTING ON MULTIPROCESSORS" by "ASHISH KUMAR MAURYA" has been carried out under my supervision and that it has not been submitted elsewhere for a degree.

It is further certified that the student has fulfilled all the requirements of Comprehensive, Candidacy and SOTA.

Prof. Anil Kumar Tripathi

Department of Computer Science and Engineering
Indian Institute of Technology
(Banaras Hindu University)
Varanasi-221005, India

October 2018

DECLARATION BY THE CANDIDATE

I, ASHISH KUMAR MAURYA certify that the work embodied in this thesis is my own bonafide work and carried out by me under the supervision of Prof. Anil Kumar Tripathi from July-2014 to July-2018, at the Department of Computer Science and Engineering, Indian Institute of Technology (BHU), Varanasi, India. The matter embodied in this thesis has not been submitted for the award of any other degree/diploma. I declare that I have faithfully acknowledged and given credits to the research workers wherever their works have been cited in my work in this thesis. I further declare that I have not willfully lifted up any other's work, paragraphs, text, data, results, etc., reported in journals, books, magazines, reports dissertations, theses, etc., or available at websites and included them in this thesis and cited as my own work.

Date : **Signature of the Student**

Place : **(ASHISH KUMAR MAURYA)**

CERTIFICATE BY THE SUPERVISOR

It is certified that the above statement made by the student is correct to the best of my/our knowledge.

Signature of Supervisor

(Prof. Anil Kumar Tripathi)

Signature of Head of Department

COPYRIGHT TRANSFER CERTIFICATE

Title of the Thesis: SOME OBSERVATIONS ON TASK SCHEDULING ALGORITHMS FOR DISTRIBUTED COMPUTING ON MULTIPROCESSORS

Name of the Student: ASHISH KUMAR MAURYA

Copyright Transfer

The undersigned hereby assigns to the Indian Institute of Technology (Banaras Hindu University) Varanasi, India all rights under copyright that may exist in and for the above thesis submitted for the award of the DOCTOR OF PHILOSOPHY.

Date :

(ASHISH KUMAR MAURYA)

Place :

Note: However, the author may reproduce or authorize others to reproduce material extracted verbatim from the thesis or derivative of the thesis for author's personal use provided that the source and the Institute's copyright notice are indicated.

ACKNOWLEDGEMENTS

Foremost, I would like to express my sincere gratitude to my supervisor, Professor Anil Kumar Tripathi, for giving me the opportunity to undertake this Ph.D. I am deeply grateful for his invaluable guidance, advice, and motivation throughout my candidature. I especially thank him for providing me with all the tools that enabled me to successfully complete this thesis.

I would like to express my gratitude to the Research Program Evaluation Committee members, Professor Sanjay Kumar Singh, Dr. Ravi Shankar Singh and Dr. Ashok Ji Gupta for their encouragement and constructive suggestions during my candidature. Special thanks to Dr. Bhaskar Biswas for their support during course work. I would like to thank all the faculties of the department, especially Professor Ravi Bhushan Mishra, Professor Kaushal Kumar Shukla, and Professor Rajeev Srivastava for their valuable suggestions.

I would like to thank all the past and current members of the laboratory, in which I have done my research work, for their friendship, support and having useful discussions. In particular, I thank Dr. Ratneshwar, Dr. Lalit Kumar Singh, Dr. Karm Veer Singh, Dr. Kamal Sheel Mishra, Dr. Vinay Kumar, Shubham Jalan, Dipty Tripathi, and Amit Biswas. I would also like to thank my other co-researchers, especially Dr. Durgesh Singh, Dr. Mridula Verma, Kuldeep Singh, Sumit Jaiswal, Dr. Anupam Biswas and Vibhav Prakash Singh, and non-teaching staff of the department, who supported me a lot and made my Ph.D. journey more enjoyable. I would like to convey my heartfelt gratitude to my friend Dr. Dinesh Singh for his constant encouragement and support.

I acknowledge the Indian Institute of Technology (BHU), Varanasi, India and Ministry of Human Resource and Development, Government of India for providing me with scholarships to pursue my doctoral studies.

I would like to thank my parents, parents-in-laws, my brothers and my sister for their unconditional love and support. I thank my father, Mr. Anant Ram Maurya, for always believing in me, for encouraging me to achieve every step of this difficult journey, and for his words of praise when they were most needed. I thank my mother, Mrs. Meena Maurya, for her blessing and caring love. Finally, I would like to thank my wife, Namita, and our son Yashveer for their selflessness, unconditional love, endless support, and inspiration. Achieving this amazing goal would not have been possible without them and I will forever be grateful for that.

- Ashish Kumar Maurya

Contents

Certificate iii

Declaration by the Candidate v

Copyright Transfer Certificate vii

Acknowledgements ix

Contents xi

List of Figures xv

List of Tables xix

Abbreviations xxiii

Symbols xxvii

Preface xxix

1 Introduction 1
- 1.1 Background . . . 3
  - 1.1.1 Distributed Systems . . . 3
  - 1.1.2 Distributed Computing versus Parallel Computing . . . 5
  - 1.1.3 Task Scheduling in Distributed Computing . . . 5
- 1.2 Motivation . . . 7
- 1.3 Objective . . . 7
- 1.4 Thesis Contributions . . . 8
- 1.5 Thesis Organization . . . 8

2 Preliminaries and Literature Review 13
2.1 Preliminaries 13
2.1.1 Assumptions 13
2.1.2 Definitions 14
2.1.3 Task Graph 17
2.1.4 Task Scheduling Problem 18
2.1.5 Real-world Application Graphs 19
2.1.5.1 Gaussian Elimination 19
2.1.5.2 Fast Fourier Transform 19
2.1.5.3 Systolic Array 20
2.1.5.4 Montage Workflow 21
2.1.5.5 Epigenomics Workflow 22
2.2 Literature Review 23
2.2.1 List Scheduling Algorithms 23
2.2.2 Duplication-based Scheduling Algorithms 27
2.2.3 Clustering-based Scheduling Algorithms 31
2.2.4 Outcome of Literature Review 34
2.3 Summary 35

3 Benchmarking Task Scheduling Algorithms and a Possible Framework 37
3.1 Benchmarking 38
3.2 Metrics 39
3.2.1 Schedule Length Ratio 39
3.2.2 Speedup 40
3.2.3 Efficiency 40
3.2.4 Frequency of best schedules 40
3.2.5 Slack 40
3.3 Algorithms considered for Benchmarking 41
3.3.1 HEFT Algorithm 41
3.3.2 PETS Algorithm 42
3.3.3 LDCP Algorithm 43
3.3.4 Lookahead Algorithm 44
3.3.5 CEFT Algorithm 45
3.3.6 PEFT Algorithm 46
3.4 Experimental Results and Discussion 47
3.4.1 Randomly Generated Application Graphs 47
3.4.2 Real-world Application Graphs 53
3.4.2.1 Gaussian Elimination 54
3.4.2.2 Fast Fourier Transform 56
3.4.2.3 Montage Workflow 58
3.4.2.4 Epigenomics Workflow 61

3.5 A Possible Framework for Benchmarking 65
  3.5.1 Advantages and Limitations of the Proposed Framework . . . 67
3.6 Summary 68

4 An Edge Priority-based Scheduling Algorithm 71
4.1 The EPS Algorithm 72
  4.1.1 Edge Prioritization 74
  4.1.2 Clustering 74
  4.1.3 The Algorithm 75
  4.1.4 Algorithm Complexity Analysis 76
  4.1.5 An Illustrative Example 77
4.2 Performance Metrics 80
  4.2.1 Normalized Schedule Length 80
  4.2.2 Speedup 80
  4.2.3 CCR 80
  4.2.4 Percentage Improvement in NSL 81
  4.2.5 Percentage Improvement in Speedup 81
4.3 Experimental Results and Discussion 81
  4.3.1 Randomly Generated Task Graphs 82
  4.3.2 Real-World Application Graphs 92
    4.3.2.1 Gaussian Elimination 92
    4.3.2.2 Fast Fourier Transform 103
4.4 Summary 112

5 Effective Critical Path Scheduling Algorithm 115
5.1 The ECP Algorithm 116
  5.1.1 Critical Path Computation and Edge Selection 117
  5.1.2 Clustering 118
  5.1.3 The ECP Algorithm 119
  5.1.4 Algorithm Complexity Analysis 120
    5.1.4.1 Analysis of Algorithm 2 120
    5.1.4.2 Analysis of Algorithm 3 121
    5.1.4.3 Analysis of Algorithm 4 121
  5.1.5 An Illustrative Example 121
5.2 Experimental Results and Discussion 124
  5.2.1 Randomly Generated Task Graphs 124
  5.2.2 Real-World Application Graphs 126
    5.2.2.1 Gaussian Elimination 126
    5.2.2.2 Fast Fourier Transform 131
    5.2.2.3 Systolic Array 136
5.3 Summary 144

6 An Energy-aware Task Scheduling Algorithm 147
- 6.1 Related Work . . . 148
- 6.2 System Model and Problem Formulation . . . 149
  - 6.2.1 Application Model . . . 150
  - 6.2.2 Processor Model . . . 150
  - 6.2.3 Energy Model . . . 151
  - 6.2.4 Problem Formulation . . . 152
- 6.3 The EAEPS Algorithm . . . 152
  - 6.3.1 Priority Function for the Edges . . . 153
  - 6.3.2 Voltage Scaling for Non-critical Tasks . . . 153
  - 6.3.3 The Algorithm . . . 154
- 6.4 Experimental Results . . . 155
- 6.5 Summary . . . 156

7 Conclusions and Future Directions 157

Bibliography 161

List of Publications 173

List of Figures

1.1 Thesis organization. 9

2.1 A sample task graph containing 15 tasks for a multiprocessor system having homogeneous processors. 18
2.2 A sample task graph containing 10 tasks for a multiprocessor system having heterogeneous processors and the corresponding matrix showing execution times of the tasks on each processor for a 3- processor system. 19
2.3 A Gaussian Elimination DAG for matrix size 5. 20
2.4 A Fast Fourier Transform DAG for four input points. 21
2.5 A systolic array DAG for n = 3. 21
2.6 A Montage workflow graph having 25 nodes. 22
2.7 An Epigenomics workflow graph having 24 nodes. 23

3.1 Average SLR results for random graphs with respect to the DAG size. 49
3.2 Average slack results for random graphs with respect to the DAG size. 50
3.3 Average SLR results for random graphs with respect to the CCR. . . 51
3.4 Average SLR results for random graphs with respect to the heterogeneity. 51
3.5 Efficiency of algorithms for random graphs with respect to the number of processors. 52
3.6 Average SLR for Gaussian Elimination graphs with respect to the matrix size. 54
3.7 Average SLR for Gaussian Elimination graphs with respect to the CCR. 55
3.8 Efficiency of algorithms for Gaussian Elimination graphs with respect to the number of processors. 55
3.9 Average SLR for FFT graphs with respect to the input points. 56
3.10 Average SLR for FFT graphs with respect to the CCR. 57
3.11 Efficiency of algorithms for FFT graphs with respect to the number of processors. 58
3.12 Average SLR for Montage workflow graphs with respect to the CCR. 59
3.13 Average SLR for Montage workflow graphs with respect to the number of processors. 60

3.14 Average SLR for Montage workflow graphs with respect to the heterogeneity. 60
3.15 Efficiency of algorithms for Montage workflow graphs with respect to the number of processors. 61
3.16 Average SLR for Epigenomics workflow graphs with respect to the CCR. 62
3.17 Average SLR for Epigenomics workflow graphs with respect to the number of processors. 63
3.18 Average SLR for Epigenomics workflow graphs with respect to the heterogeneity. 63
3.19 Efficiency of algorithms for Epigenomics workflow graphs with respect to the number of processors. 64
3.20 Proposed framework for benchmarking of task scheduling algorithms. 65

4.1 Clustering steps of the example task graph in Fig. 1 with the EPS algorithm (a) Initial clustering (initial makespan = 39), (b) clustering after zeroing the edge $e_{13,14}$ (partial makespan = 36), (c) clustering after zeroing the edge $e_{0,1}$ (partial makespan = 33), (d) clustering after zeroing the edge $e_{1,3}$ (partial makespan = 31) 78
4.1 Clustering steps of the example task graph in Fig. 1 with the EPS algorithm (e) clustering after zeroing the edge $e_{6,10}$ (partial makespan = 29), (f) clustering after zeroing the edge $e_{7,12}$ (partial makespan = 26), (g) clustering after zeroing the edge $e_{12,13}$ (partial makespan = 25), (h) clustering after zeroing the edge $e_{3,7}$ (final makespan = 23). . 79
4.2 Average NSL results obtained for random task graphs as a function of number of nodes. 83
4.3 Average Speedup results obtained for random task graphs as a function of number of nodes. 83
4.4 Average NSL results obtained for Gaussian Elimination task graphs as a function of matrix size. 93
4.5 Average NSL results obtained for Gaussian Elimination task graphs as a function of CCR. 94
4.6 Average Speedup results obtained for Gaussian Elimination task graphs as a function of matrix size. 95
4.7 Average Speedup results obtained for Gaussian Elimination task graphs as a function of CCR. 95
4.8 Average NSL results obtained for FFT task graphs as a function of input points. 104
4.9 Average NSL results obtained for FFT task graphs as a function of CCR. 104
4.10 Average speedup results obtained for FFT task graphs as a function of input points. 105

4.11 Average speedup results obtained for FFT task graphs as a function of CCR. 106

5.1 Clustering steps of the example task graph in Fig. 1 with the ECP algorithm (a) Initial clustering (initial schedule length = 39), (b) clustering after merging T_1 and T_3 (partial schedule length = 35), (c) clustering after merging T_7 and T_{12} (partial schedule length = 34), (d) clustering after merging T_6 and T_{10} (partial schedule length = 31). 122
5.1 Clustering steps of the example task graph in Fig. 1 with the ECP algorithm (e) clustering after merging T_{13} and T_{14} (partial schedule length = 28), (f) clustering after merging T_0 and $\{T_1, T_3\}$ (partial schedule length = 26), (g) clustering after merging $\{T_7, T_{12}\}$ and $\{T_{13}, T_{14}\}$ (partial schedule length = 25), (h) clustering after merging $\{T_0, T_1, T_3\}$ and $\{T_{12}, T_{13}, T_{14}\}$ (final schedule length = 23). 123
5.2 Average NSL results obtained for random task graphs as a function of number of nodes. 125
5.3 Average Speedup results obtained for random task graphs as a function of number of nodes. 125
5.4 Average NSL results obtained for Gaussian Elimination task graphs as a function of matrix size. 129
5.5 Average NSL results obtained for Gaussian Elimination task graphs as a function of CCR. 129
5.6 Average Speedup results obtained for Gaussian Elimination task graphs as a function of matrix size. 130
5.7 Average Speedup results obtained for Gaussian Elimination task graphs as a function of CCR. 130
5.8 Average NSL results obtained for FFT task graphs as a function of input points. 132
5.9 Average NSL results obtained for FFT task graphs as a function of CCR. 133
5.10 Average speedup results obtained for FFT task graphs as a function of input points. 133
5.11 Average speedup results obtained for FFT task graphs as a function of CCR. 134
5.12 Average NSL results obtained for systolic array task graphs as a function of number of nodes. 137
5.13 Average NSL results obtained for systolic array task graphs as a function of CCR. 137
5.14 Average speedup results obtained for systolic array task graphs as a function of number of nodes. 138
5.15 Average speedup results obtained for systolic array task graphs as a function of CCR. 138

List of Tables

2.1 A comparison of some known list scheduling algorithms. 27
2.2 A comparison of some known duplication-based task scheduling algorithms. 31
2.3 A comparison of some known clustering-based task scheduling algorithms. 34

3.1 Parameters and their corresponding values used in the generation of random graphs 48
3.2 Pair-wise Schedule Length Comparison of the Scheduling Algorithms. 53
3.3 Configuration parameters for Gaussian Elimination 54
3.4 Configuration parameters for Fast Fourier Transform 56
3.5 Configuration parameters for Montage workflow 58
3.6 Configuration parameters for Epigenomics workflow 62
3.7 Summary of the results of benchmarking for comparison of task scheduling algorithms. 69

4.1 t-value used for the estimation of confidence intervals when confidence level is 95 % 82
4.2 Statistical analysis of NSL obtained for random task graphs. 84
4.3 Statistical analysis of speedup obtained for random task graphs. . . . 88
4.4 Statistical analysis of NSL obtained for Gaussian Elimination task graphs with CCR value 1. 96
4.5 Statistical analysis of NSL obtained for Gaussian Elimination task graphs with CCR value 10. 97
4.6 Statistical analysis of speedup results obtained for Gaussian Elimination task graphs with CCR value 0.1. 99
4.7 Statistical analysis of speedup results obtained for Gaussian Elimination task graphs with CCR value 1. 100
4.8 Statistical analysis of speedup results obtained for Gaussian Elimination task graphs with CCR value 10. 102
4.9 Statistical analysis of NSL obtained for FFT task graphs with CCR value 1. 106
4.10 Statistical analysis of NSL obtained for FFT task graphs with CCR value 10. 107

4.11 Statistical analysis of speedup results obtained for FFT task graphs with CCR value 0.1. 109
4.12 Statistical analysis of speedup results obtained for FFT task graphs with CCR value 1. 110
4.13 Statistical analysis of speedup results obtained for FFT task graphs with CCR value 10. 111

5.1 Statistical analysis of NSL results of ECP for random task graphs. . . 127
5.2 Statistical analysis of speedup results of ECP for random task graphs. 128
5.3 Statistical analysis of NSL results of ECP for Gaussian Elimination task graphs with CCR value 1. 131
5.4 Statistical analysis of NSL results of ECP for Gaussian Elimination task graphs with CCR value 10. 131
5.5 Statistical analysis of speedup results of ECP for Gaussian Elimination task graphs with CCR value 0.1. 131
5.6 Statistical analysis of speedup results of ECP for Gaussian Elimination task graphs with CCR value 1. 132
5.7 Statistical analysis of speedup results of ECP for Gaussian Elimination task graphs with CCR value 10. 132
5.8 Statistical analysis of NSL results of ECP for FFT task graphs with CCR value 1. 135
5.9 Statistical analysis of NSL results of ECP for FFT task graphs with CCR value 10. 135
5.10 Statistical analysis of speedup results of ECP for FFT task graphs with CCR value 0.1. 136
5.11 Statistical analysis of speedup results of ECP for FFT task graphs with CCR value 1. 136
5.12 Statistical analysis of speedup results of ECP for FFT task graphs with CCR value 10. 136
5.13 Statistical analysis of NSL results obtained for systolic array task graphs with CCR value 0.1. 139
5.14 Statistical analysis of NSL results obtained for systolic array task graphs with CCR value 1. 140
5.15 Statistical analysis of NSL results obtained for systolic array task graphs with CCR value 10. 141
5.16 Statistical analysis of speedup results obtained for systolic array task graphs with CCR value 0.1. 142
5.17 Statistical analysis of speedup results obtained for systolic array task graphs with CCR value 1. 143
5.18 Statistical analysis of speedup results obtained for systolic array task graphs with CCR value 10. 144

6.1 Frequency and supply voltages used in this work 155

6.2 Energy saving comparison of EAEPS with other existing energy-aware algorithms 156

Abbreviations

ACC	**A**verage **C**omputation **C**ost
AFT	**A**ctual **F**inish **T**ime
ALAP	**A**s-**L**ate-**A**s-**P**ossible
ALST	**A**verage **L**atest **S**tart **T**ime
BDSC	**B**ounded **D**ominant **S**equence **C**lustering
BFS	**B**readth **F**irst **S**earch
BL	**B**ottom **L**evel
CASC	**C**lustering **A**lgorithm for **S**ynchronous **C**ommunication
CCLoad	**C**omputation-**C**ommunication-**Load**
CCP	**C**onstrained **C**ritical **P**ath
CCR	**C**ommunication to **C**omputation **R**atio
CEFT	**C**onstrained **E**arliest **F**inish **T**ime
CI	**C**onfidence **I**nterval
CNPT	**C**ritical **N**odes **P**arent **T**rees
CP	**C**ritical **P**ath
CPFD	**C**ritical **P**ath **F**ast **D**uplication
CPN	**C**ritical **P**ath **N**ode
CPOP	**C**ritical **P**ath **O**n a **P**rocessor
CPPS	**C**luster **P**air **P**riority **S**cheduling
CT	**C**ommunication **T**ime
DAG	**D**irected **A**cyclic **G**raph
DAGP	**D**irected **A**cyclic **G**raph that corresponds to a **P**rocessor

DBUS	**D**uplication-based **B**ottom-**U**p **S**cheduling
DCCL	**D**ynamic **C**omputation **C**ommunication **L**oad
DCP	**D**ynamic **C**ritical **P**ath
DFRN	**D**uplication **F**irst and **R**eduction **N**ext
DPM	**D**ynamic **P**ower **M**anagement
DSC	**D**ominant **S**equence **C**lustering
DTC	**D**ata **T**ransfer **C**ost
DVFS	**D**ynamic **V**oltage and **F**requency **S**caling
DVS	**D**ynamic **V**oltage **S**caling
EAD	**E**nergy-**A**ware **D**uplication
EAEPS	**E**nergy **A**ware **E**dge **P**riority **S**cheduling
EASLA	**E**nergy **A**ware **S**ervice **L**evel **A**greement
ECP	**E**ffective **C**ritical **P**ath
ECS	**E**nergy-**C**onscious **S**cheduling
EFT	**E**arliest **F**inish **T**ime
EPS	**E**dge **P**riority **S**cheduling
EST	**E**arliest **St**art **T**ime
ET	**E**xecution **T**ime
EZ	**E**dge **Z**eroing
FFT	**F**ast **F**ourier **T**ransform
GE	**G**aussian **E**limination
HCPFD	**H**eterogeneous **C**ritical **P**arents with **F**ast **D**uplicator
HCPT	**H**eterogenous **C**ritical **P**arent **T**rees
HEFD	**H**eterogeneous **E**arliest **F**inish with **D**uplicator
HEFT	**H**eterogeneous **E**arliest **F**inish **T**ime
HLD	**H**eterogeneous **L**imited **D**uplication
HNPD	**H**eterogeneous **N**-**P**redecessor **D**uplication
HPRV	**H**eterogeneous **P**riority **R**ank **V**alue
HPS	**H**igh **P**erformance **T**ask **S**cheduling

HSV	**H**eterogeneous **S**election **V**alue
IBN	**I**n-**B**ranch **N**ode
ILS	**I**terative **L**ist **S**cheduling
LC	**L**inear **C**lustering
LDBS	**L**evelized **D**uplication-**B**ased **S**cheduling
LFT	**L**atest **F**inish **T**ime
LDCP	**L**ongest **D**ynamic **C**ritical **P**ath
LMT	**L**evelized **M**in **T**ime
MCP	**M**odified **C**ritical **P**ath
MD	**M**obility **D**irected
NSL	**N**ormalized **S**chedule **L**ength
OBN	**O**ut-**B**ranch **N**ode
OCT	**O**ptimistic **C**ost **T**able
PALS	**P**ower **A**ware **L**ist **S**cheduling
PATC	**P**ower **A**ware **T**ask **C**lustering
PATS	**P**redict and **A**rrange **T**ask **S**cheduling
PEBD	**P**erformance-**E**nergy **B**alanced **D**uplication
PEFT	**P**redict **E**arliest **F**inish **T**ime
PETS	**P**erformance **E**ffective **T**ask **S**cheduling
RADS	**R**esource-**A**ware **S**cheduling Algorithm with **D**uplications
RDCC	**R**andomized **D**ynamic **C**omputation **C**ommunication
RPT	**R**ank of **P**redecessor **T**ask
SD	**S**elective **D**uplication
SFD	**S**cheduling with **F**ull **D**uplication
SLA	**S**ervice **L**evel **A**greement
SLR	**S**chedule **L**ength **R**atio
SPD	**S**cheduling with **P**artial **D**uplication
TANH	**T**ask duplication based scheduling **A**lgorithm for **N**etwork of **H**eterogeneous systems

TDS	**T**ask **D**uplication-based **S**cheduling
TL	Top Level

Symbols

$\|V\|$	Number of nodes in a task graph
$\|E\|$	Number of edges in a task graph
T_i	i^{th} task in a task graph
$e_{i,j}$	A directed edge with precedence constraint from task T_i to T_j
p_k	k^{th} processor
T_{entry}	Task without any predecessor
T_{exit}	Task without any successor
$pred(T_i)$	Set of immediate predecessors of task T_i
$succ(T_i)$	Set of immediate successors of task T_i
$AFT(T_i)$	Actual Finish Time of task T_i
$ET(T_i)$	Execution Time of task T_i
$CT(e_{i,j})$	Communication Time from task T_i to T_j
$\overline{ET(T_i)}$	Average Execution Time of task T_i
$\overline{CT(e_{i,j})}$	Average Communication Time between tasks T_i and T_j
$ET(T_i, p_j)$	Execution Time of task T_i on processor p_j
$BL(T_i)$	Bottom Level of task T_i
$TL(T_i)$	Top Level of task T_i
$EST(T_i)$	Earliest Start Time of task T_i
$EFT(T_i)$	Earliest Finish Time of task T_i
$LFT(T_i)$	Latest Finish Time of task T_i
$p(e_{i,j})$	Priority of an edge $e_{i,j}$
$slack(T_i)$	slack of a task T_i

$ET_{slack}(T_i)$	Execution Time of task T_i after considering its slack
ξ	Total energy consumption
$\xi_{dynamic}$	Dynamic energy consumption
ξ_{static}	Static energy consumption
$P_{dynamic}$	Dynamic power consumption
f	Operating frequency of the processor
V_{dd}	Supply voltage of the processor
f_{min}	Minimum operating frequency of the processor
f_{max}	Maximum operating frequency of the processor
$f_k(T_i)$	Frequency when a task T_i executed at frequency f_k
$V_k(T_i)$	Voltage when a task T_i executed at frequency f_k

PREFACE

In distributed computing, a big computational application is solved by dividing it into many tasks and executing them onto different processing units. The distributed computing environment may be homogeneous in which all processors have same processing capabilities, or it may be heterogeneous in which all processors are comprised of different processing capabilities. It involves potentially a great deal of communication overhead which restricts the performance of applications if tasks are not scheduled efficiently. The scheduling of tasks, with precedence constraints, on different processors is one of the core concerns for distributed computing in multiprocessor environments and significantly relies on the techniques employed to schedule the tasks with the aim of optimizing makespan and energy consumption. The task scheduling problem is known to be NP-complete. Therefore, many task scheduling algorithms are proposed in literature to solve this problem and new methods keep coming in. It is always useful to look for a fresh approach, towards understanding and interpretation of the existing algorithms and such an effort may lead to some possible newer ways of solving the problem.

The thesis benchmarks some well-known task scheduling algorithms for distributed computing on multiprocessors and proposes a possible framework for this purpose. The proposed approach provides for generation of graphs through a Directed Acyclic Graph generator, then produces schedules through a scheduler which makes use of scheduling algorithms and finally analyses the results obtained by using various performance metrics. The proposed framework is general in nature.

The work also attempts to propose some new algorithms for working out possible scheduling, of tasks that optimize makespan. We propose two clustering-based algorithms for scheduling of precedence constrained tasks in multiprocessor environments. The first algorithm proposes and uses the idea of edge prioritization to obtain

meaningful clustering of the tasks. The second algorithm makes use of edge zeroing concept on the critical path to reduce the communication cost among the tasks of an application. We have performed an average analysis of the results obtained for various real-world application graphs and random graphs. Along with average analysis, we also performed a statistical analysis of the results using confidence intervals.

Further, we propose an energy-aware scheduling algorithm for multiprocessor environments which aims to reduce power consumption by exploiting dynamic voltage and frequency scaling technique. This algorithm is an energy aware version of our first proposed algorithm and uses the idea edge prioritization to save energy consumption. It also studies the slack time for non-critical tasks, extends their execution time and reduces the energy consumption without increasing the makespan of the application. The simulation experiments conducted with four well-known energy aware scheduling algorithms for some selected benchmark random graphs demonstrate that the proposed energy-aware scheduling algorithm achieves more energy saving than compared algorithms.

DEDICATED

To
My Beloved Parents, Wife and Son