International Journal of Computer Applications and Technology
Volume 1- Issue 2, 2012, 44-48

International Journal of Computer Applications and Technology
journal homepage: www.ijcat.com

Agent based Task Scheduling in Grid

Ashish Chandak
Department of Computer Science and Engineering,
National Institute of Technology, Rourkela
Rourkela, Odisha, India

Bibhudatta Sahoo
Department of Computer Science and Engineering
National Institute of Technology, Rourkela
Rourkela, Odisha, India

Ashok Kumar Turuk
Department of Computer Science and Engineering
National Institute of Technology, Rourkela
Rourkela, Odisha, India

Abstract: Grid computing is considered to be wide area distributed computing which provides sharing, selection and aggregation of distributed resources. Agent paradigm has been widely used in large number of research area and now a days it is widely used in grid computing. In this paper, we proposed agent based strategy that uses knowledge base reasoning framework for task scheduling in grid which minimizes makespan.

Keywords: Grid Computing; Task Scheduling; Agent; Knowledge Base.

1. INTRODUCTION

Autonomous agents are intelligent entities that can operate on behalf of the human users autonomously to solve the problems, negotiate with other agents (peers), learn from the past and predict upcoming events. Agents are used in various application domains such as industrial applications viz. process control [1], commercial applications viz. electronic commerce [2], business process management [3], medical applications viz. patient monitoring [4], health care [5]. They get the problem from the users or other agents, discover needed resources, consult with other agents (negotiation) and offer a proper solution. They also learn from the past, update their knowledge and predict the future events [6]. Agent technology is new in distributed system and can be used in computing network. The main difference between agent and scheduler is coordination, cooperation and learning [6]. Agents work together, use the resources located on each other optimally and work as a team to solve a problem. Agents are flexible entities and are capable to adapt themselves to new environments. This means agents are well suited in dynamic environment. Even though agents are independent, they always communicate with others to discover needed resources. Following are some properties of agent:-

- Autonomous: - Agents are proactive, goal directed and is capable of acting without direct external intervention.
- Interactive: - Communicate with the environment and other agents.
- Adaptive: - Agents dynamically adapt to and learn about their environment. They are adaptive to uncertainty and change.
- Cooperative: - Able to coordinate with other agents to achieve common purpose.
- Social: - They work together.
- Coordinative: - Able to perform some activity in a shared environment with other agents, via plans, workflows, or some other process mechanism.

The agent paradigm has been successfully used in a large number of research areas. An agent-based methodology is developed for building large-scale distributed systems with highly dynamic behaviors [7,8]. Authors in [9] use ordinal sharing learning (OSL) method based on multi-agent reinforcement learning to solve the task scheduling problem in grid. A combination of intelligent agents and multi-agent approaches is applied to both local grid resource scheduling and global grid load balancing is used in [11]. Authors in [10] applied use of economic agent in grid computing. What distinguishes our work from other is that we applied knowledge base reasoning in agent for task scheduling. This chapter describes the conceptual agent framework which provides the management of various resources in grid environment. The design goals of our model focus on combining distributed resources and knowledge base reasoning agent technology to manage various resources on the grid. The evaluation of our proposal shows the advantage of using agents in grid computing.

2. AGENT DESCRIPTION

Agent is an autonomous entity that can interact with its environment. In other words, it is anything that can be viewed as perceiving its environment through sensors, and acting upon on environment with effectors. Agent processing overview is shown in Fig. 1.
An agent system is represented as:
Agent System=< Agents, Coupling >
An agent is defined in a formal way as: <Agent_{Id}, Role>
Agent_{Id}: Agent identification, which is used in order to identify every agent of the system.

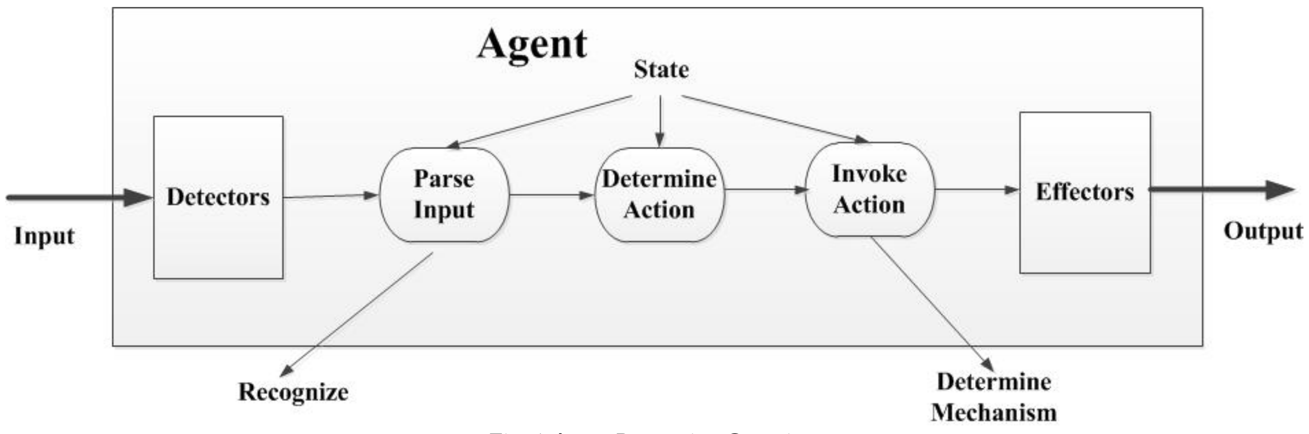

Fig. 1 Agent Processing Overview

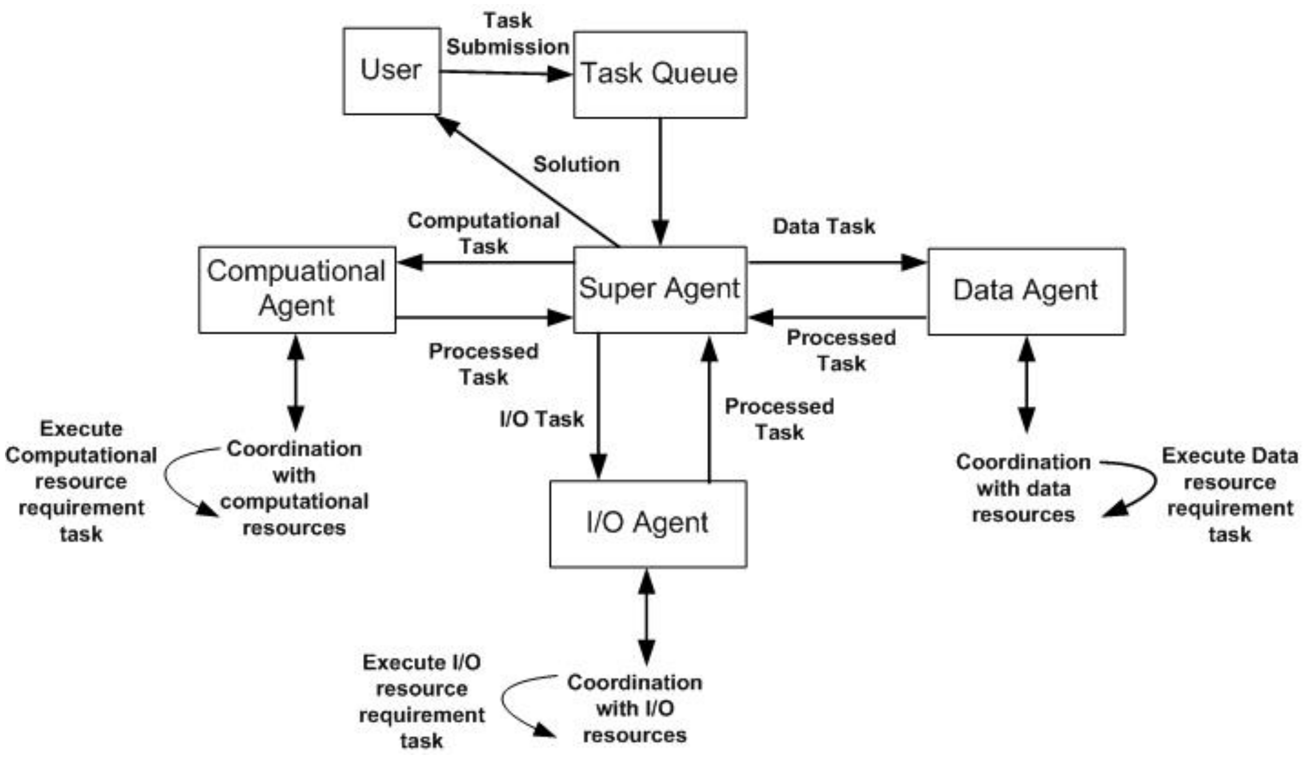
User
Task Submission
Task Queue
Solution
Computational Task
Data Task
Compuational Agent
Super Agent
Data Agent
Processed Task
Processed Task
I/O Task
Processed Task
Execute Computational resource requirement task
Coordination with computational resources
Coordination with data resources
Execute Data resource requirement task
I/O Agent
Execute I/O resource requirement task
Coordination with I/O resources

Fig. 2 Task Execution in Agent System

Role: This field represents the kind of agent i.e. computational, I/O or data.
Coupling is a mapping of an input and output from/ to with other agent.

3. TASK EXECUTION MODEL

Fig. 2 shows how a solution can be found for a specific problem using cooperation of the super-agent and other agent. Super-agent, after receiving a task is processed internally and each task is sent to the responsible agent for further processing. Results are sent back to the super-agent and a comprehensive solution is offered by agent. In our model, agent has

- the ability to link resources with/into the grid via some kind of interconnection mechanism.
- the ability to use grid resources to perform some task.
- the ability to compose grid resources to form new combined resources that can be used in the same way as the individual resources.

Internal architecture of agent is shown in Fig. 3. Agent can add new information obtained from the result of inquiries to expand their current knowledge. This causes agent to learn about what they received from other agent and to extend their current information about problems and solutions. It also prevents redundant inquires and searching. Agent can offer solution, a set of results created by inquires, for the users's problems that consist of group of low-level tasks. A problem can solve by many agents with different knowledge. The agent consists of several internal modules. Modules are responsible for performing the internal activities of the agent.

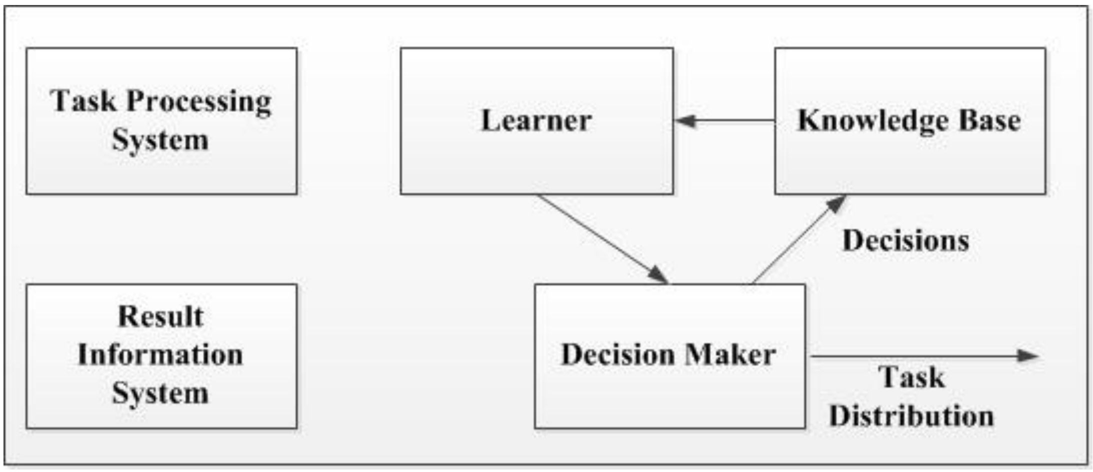

Fig. 3 Internal Architecture of Agent

Knowledge Base No. of attempt for ‘X’ resource	Node A	Node B	Node C
Search No. 1	Failure	Success	Success
Search No. 2	Success	Success	Failure
Search No. 3	Failure	Success	Success
Success Ratio	1/3	2/3	3/3
Ranking	3	1	2

Table 1: Inquiry Table maintained by Knowledge Base

Cooperation of the modules enables the agent to perform the required tasks. Following are components of agent.

- Result Management System (RMS):- This module collect and return results to super-agent
- Knowledge Base: - This module is responsible for storing and retrieving in formation from the result management system and learner module, which carry resource related data and task execution results respectively. Knowledge base includes computing capability and status of every node of each site. The table 1 shows inquiry table maintain by knowledge base. Ranking helps to know which node is best match for task 'X'. Ranking shows sequences of the destinations for the next search. Thus before sending inquires agent will decide which site should be targeted first. The inquiry table is updated every time when an inquiry is issued. We can see that node B should be first destination for sending next inquiry.
- Learner: - Learner takes the input from knowledge base regarding resources. Learner always looking for newer solutions and updates it to knowledge base according to latest results obtained from inquires and overwrites the old results. For example, if computational agent after refereeing its internal knowledge realizes that if task came with x demand then it is forwarded to x resource without any further inquiry. This technique prevents iterative inquires for resources.
- Decision maker: - Decision maker is central part of an agent and center of operations. Decision maker takes the resource selection decision and forward task to appropriate nodes. Decisions taken by this module are updated into knowledge base.
- Task processing system: - It process tasks.

4. SIMULATION EXPERIMENT

We developed a simulation application in MATLAB to carry out the experiments. Each simulation experiment ends when 1000 tasks executions gets completed. The arrival of tasks is modeled as Poisson random process. To evaluate performance we have considered following three types of tasks: a) I/O intensive tasks b) Data intensive tasks c) Computational intensive tasks. We consider nine nodes for computation. In without agent based scheduling, task are randomly assigned to any of the available nodes regardless of task type but in agent based scheduling, first of all incoming task type is determined and allocated to appropriate node i.e. I/O task is assigned to I/O specific node, data task is assigned to data specific node, and computational resource requirement is assigned to computation specific node since agent has knowledge of all resource in the system. We dedicate three nodes to each of task type. In agent based scheduling, if same resource requirement task came then it is directly forwarded to node where same type of task is executed previously since knowledge base keeps all the information about previous tasks execution. We evaluate performance of without agent and with agent based system. Tasks are schedule using Max- Min, Min-Min and FCFS heuristics.

4.1 Makespan Results

Makespan results are shown in Fig. 4, 5 and 6. These results clearly indicate that with agent based scheduling gives less makespan as compared to without agent based scheduling of tasks. The performance of non-agent based scheduling is turned out to be poor as compared to agent based task scheduling strategy.

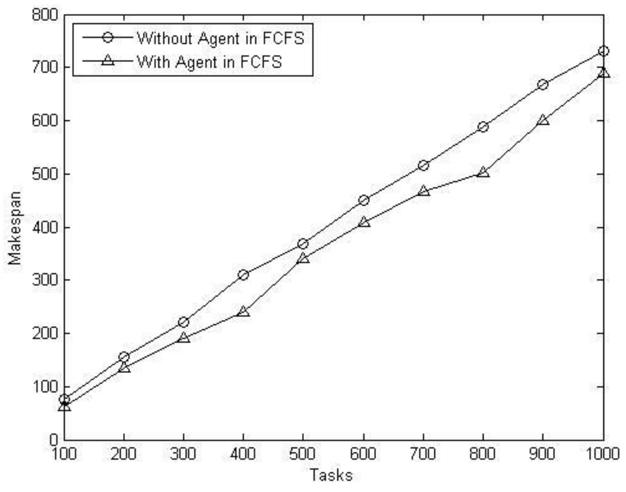

Fig. 4 Makespan Comparison using FCFS

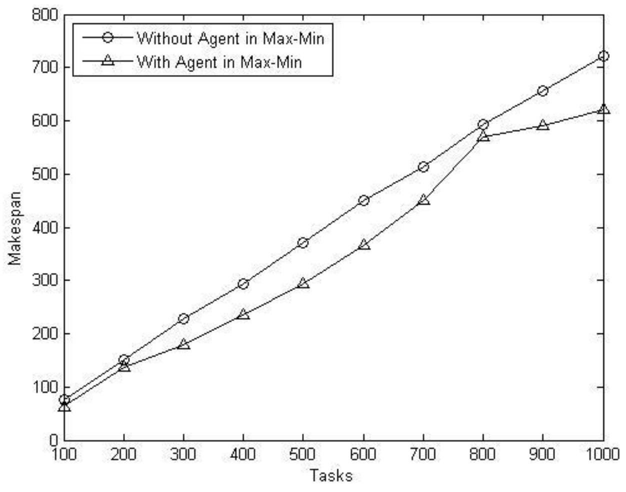

Fig. 5 Makespan Comparison using Max-Min

5. CONCLUSION

In this chapter, an agent based strategy for task scheduling in grid is presented. The proposed model consists of two different types of agent: super-agent and task specific agent. Super-agent determines task type and forwards it to task specific agent while task specific agent coordinate with specific resources and return results to super-agent. A performance evaluation of with agent based scheduling and non-agent based task scheduling is conducted. The experiment shows that our proposed framework shows optimal makespan.

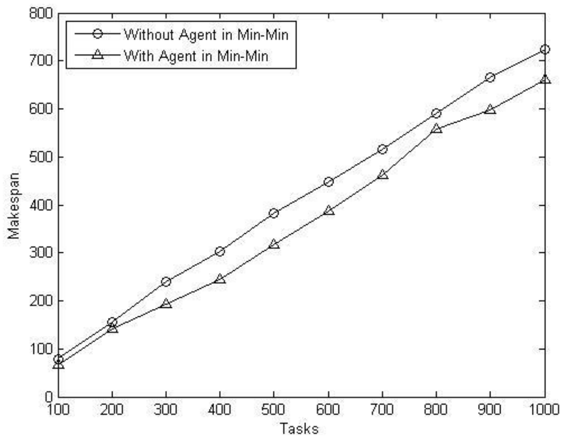

Fig. 6 Makespan Comparison using Min-Min

6. REFERENCES

[1] Corera J. M. Laresgoiti Jennings, N. R. Developing Industrial Multi-Agent Systems. In In: Proceedings of the First International Conference on Multiagent Systems, (ICMAS-95), pp. 423-430, 1995.

[2] Maes P. Chavez, A. Kasbah: An Agent Marketplace for Buying and Selling Goods. In Proceedings of First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Systems, London, UK, 1996.

[3] Faratin P. Johnson M. J. Norman T. J. O'Brien P. Wiegand M. E. Jennings, N. R. Agent-based business process management. International Journal of Cooperative Information Systems, 5:pp. 105 - 130, 1996.

[4] Hewett M. Waashington R. Hewett R. Seiver A Hayes-Roth, B. Distributing intelligence within an individual, volume 2. Morgan Kaufmann, second edition, 1995.

[5] Jennings N. R. Fox J. Huang, J. An agent-based approach to health care management. International Journal of Applied Artificial Intelligence, 9, pp. 401-420, 1995.

[6] Wang F. Areibi S. Homayounfar, H. Advanced P2P Architecture Using Autonomous Agents. International Journal of Computers, Systems and Signals, 4:pp. 115 - 118, 2002.

[7] G.R. Nudd J. Cao, D.J. Kerbyson. Dynamic Application Integration Using Agent-based Operational Administration. In: Proceedings of the PAAM00, pp. 393 - 396, 2000.

[8] G.R. Nudd J. Cao, D.J. Kerbyson. High Performance Service Discovery in Large-Scale Multi-Agent and Mobile-Agent Systems. International Journal of Software Engineering and Knowledge Engineering, 11(5):pp.621 - 641, 2001.

[9] Jun Wu, Xin Xu, Pengcheng Zhang, and Chunming Liu. A Novel Multi-Agent Reinforcement Learning Approach for Job Scheduling in Grid Computing. Future Generation Computer Systems, 27(5):pp. 430 - 439, 2011.

[10] Junwei Cao, Daniel P. Spooner, Stephen A. Jarvis, and Graham R. Nudd. Grid Load Balancing using Intelligent Agents. Future Generation Computer Systems, 21(1):pp. 135 - 149, 2005.

[11] Li Chunlin and Li Layuan. The Use of Economic Agents Under Price Driven Mechanism in Grid Resource Management. Journal of Systems Architecture, 50(9):521 - 535, 2004.

International Journal of Computer Applications and Technology
Volume 1- Issue 2, 2012, 49-52

International Journal of Computer Applications and Technology
journal homepage: www.ijcat.com

Task Scheduling Heuristic in Grid Computing

Ashish Chandak
Department of Computer Science and Engineering,
National Institute of Technology, Rourkela
Rourkela, Odisha, India

Bibhudatta Sahoo
Department of Computer Science and Engineering,
National Institute of Technology, Rourkela
Rourkela, Odisha, India

Ashok Kumar Turuk
Department of Computer Science and Engineering,
National Institute of Technology, Rourkela
Rourkela, Odisha, India

Abstract: Task scheduling is heart of any grid application which guides resource allocation in grid. Heuristic task scheduling strategies have been used for optimal task scheduling. Heuristic techniques have been widely used by the researchers to solve resource allocation problem in grid computing. In this paper, we classify heuristic task scheduling strategies in grid on the basis of their characteristics. We identified different types of heuristics such as population based heuristic, economic heuristic, meta heuristic, simple heuristic, and hybrid heuristic.

Keywords: Task Scheduling; Simple Heuristic; Economic Heuristic; Iterative Heuristic.

1. INTRODUCTION

Optimal or near optimal resource allocation for task scheduling is complex undertaking in grid computing. Task scheduling strategies have been widely used by researches to solve resource allocation problem in grid computing. Heuristic task scheduling strategies help to search optimal task scheduling in grid computing. Heuristics do not try to optimize i.e. find the best solution, but rather satisfied i.e. find good-enough solution. [1] proposed that all heuristics rely on effort reduction by one or more of the following: a) examining fewer cues, b) reducing the effort of retrieving cue values, c) simplifying the weighting of cues, d) integrating less information, and e) examining fewer alternatives.

Importance of heuristics task scheduling strategies:-

- Find perfect resource match for task.
- Incorporate intelligence to give optimal solution.
- Reduce uncertainty.
- Provide smart choices.
- Utilize fewer resources.
- Save formulation time.
- Reduce computational time.
- Produce acceptable solution.
- Find solution in reasonable time.
- Gives better judgments.

2. HEURISTIC CLASSIFICATION

It is reported in literature that grid task scheduling is NP complete problem [2]. Various heuristic have been proposed in literature for grid scheduling which we broadly classified into five types. They are i) Economic Heuristic [3, 4, 5, 6], ii) Population Based Heuristic [7, 8, 9] iii) Meta-Heuristic [10, 2, 11, 12, 13, 15-19], iv) Simple Heuristic [20-23] and, v) Hybrid Heuristic [24-28]. Fig. 1 shows the proposed classification of heuristics for task scheduling in grid. Based on approach used to arrive at the solution we categorized heuristic into three types:-

1) Iterative Heuristic: - It designates a computational method that optimizes a problem by iteratively trying to improve a candidate solution. There are two types of iterative heuristics.

- Population Based Heuristic:- It is computational method that optimizes problem by taking population of individuals. This heuristic calculates the fitness of each individual and based on their fitness individual are takes out .The new population is used for next iteration of algorithm. When satisfactory fitness level is reached then algorithm stops [29]. We have identified two of population based heuristic: Genetic algorithm (GA) [7, 9], Memetic algorithm (MA) [8].
- Meta Heuristic: - Meta heuristics make few or no assumptions about the problem being optimized and can search very large spaces of candidate solutions. Meta heuristics are used for combinatorial optimization in which an optimal solution is sought over a discrete search-space [30]. We distinguished four meta-heuristics viz. Simulated Annealing (SA) [13], Tabu Search (TS)

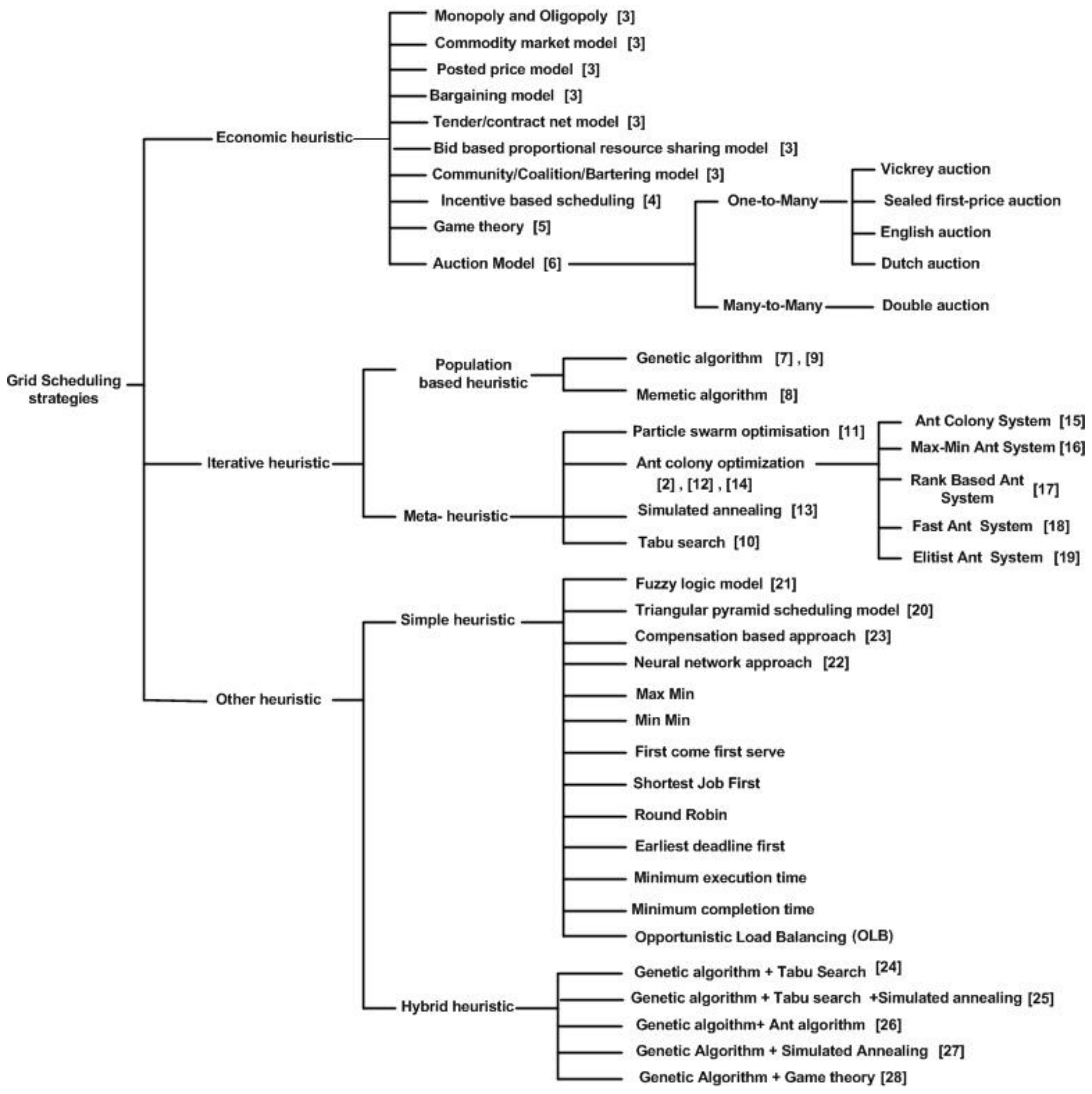
Grid Scheduling strategies
Economic heuristic
Monopoly and Oligopoly [3]
Commodity market model [3]
Posted price model [3]
Bargaining model [3]
Tender/contract net model [3]
Bid based proportional resource sharing model [3]
Community/Coalition/Bartering model [3]
Incentive based scheduling [4]
Game theory [5]
Auction Model [6]
One-to-Many
Vickrey auction
Sealed first-price auction
English auction
Dutch auction
Many-to-Many
Double auction
Iterative heuristic
Population based heuristic
Genetic algorithm [7] , [9]
Memetic algorithm [8]
Meta- heuristic
Particle swarm optimisation [11]
Ant colony optimization [2] , [12] , [14]
Simulated annealing [13]
Tabu search [10]
Ant Colony System [15]
Max-Min Ant System [16]
Rank Based Ant System [17]
Fast Ant System [18]
Elitist Ant System [19]
Other heuristic
Simple heuristic
Fuzzy logic model [21]
Triangular pyramid scheduling model [20]
Compensation based approach [23]
Neural network approach [22]
Max Min
Min Min
First come first serve
Shortest Job First
Round Robin
Earliest deadline first
Minimum execution time
Minimum completion time
Opportunistic Load Balancing (OLB)
Hybrid heuristic
Genetic algorithm + Tabu Search [24]
Genetic algorithm + Tabu search +Simulated annealing [25]
Genetic algoithm+ Ant algorithm [26]
Genetic Algorithm + Simulated Annealing [27]
Genetic Algorithm + Game theory [28]

Fig. 1 Classification of Heuristic for Task Scheduling in Grid

[10], Particle Swarm optimization (PSO) [11] and Ant Colony Optimization (ACO) [2, 12].

2) Economic Based Heuristic: - In competitive market, there is always scarcity of resources. Economic heuristic deals with matching tasks to available resources provider and consumer get sufficient incentive to stay and play in competitive market. [3] introduced various economic approaches such as commodity market model, posted price model, bargaining model, tendering/contract-net model, auction model, bid-based, proportional resource sharing model, and community/coalition/bartering model for grid resource allocations while Incentive based Scheduling [4], Game Theory [5] and Auction Model [6] are other models used for task scheduling.

3) Other Heuristic: -

Hybrid heuristic:- Hybrid heuristic optimize the problem by combining two or more heuristic for scheduling tasks in grid. We identified various heuristic such as GA+TS [24], GA+TS+SA [25], GA+Ant Algorithm [26], GA+SA [27], and GA+ Game Theory [28] for task scheduling.

- Simple Heuristic:- Simple heuristic differs from cognitive science and economics. Traditional heuristics have limited knowledge and limited reasons to make scheduling choice. But sometimes task scheduling requires more plausible notion to solve scheduling problem. It provides fast and frugal way for decision making. It provides smart choices Min, Min-min, First Come First Serve (FCFS), Shortest Job First (SJF), Round Robin

(RR), Earliest Deadline First (EDF), Minimum Execution Time (MET), Minimum Completion Time (MCT), and Opportunistic Load Balancing (OLB).

3. COMPARISON OF SIMPLE HEURISTIC

In this section, we compare simple heuristics viz. Max-Min, Min-Min and FCFS.

3.1 Simulation Model

A scheduling algorithm can be classified into clairvoyant or non-clairvoyant, with regard to knowledge about characteristics of tasks. A clairvoyant scheduling algorithm may use information of tasks characteristics such as service demand, whereas a non-clairvoyant algorithm assumes nothing about the characteristics of the tasks [42]. We assume that tasks service demands are known to the scheduler. We developed a simulation application using Matlab 7.11. Each simulation experiment ends when 1000 tasks execution gets completed. To evaluate performance, we have considered following three types of tasks: a) I/O intensive tasks b) Data intensive tasks c) Computational intensive tasks. We consider site which consist nine nodes for computation. We dedicate three nodes to each type of task. First of all, incoming task type is determined and is allocated to appropriate node i.e. I/O task is assigned to I/O specific node, data task is assigned to data specific node and computational resource requirement task is assigned to computation specific node.

3.2 Makespan Result

Makespan result is shown in Fig. 2. From comparison, it is observed that scheduling tasks using Max-Min heuristic gives less makespan as compared Min-Min and FCFS heuristics.

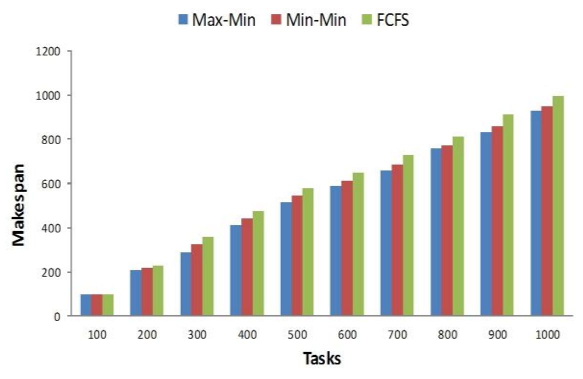

Fig. 2 Makespan Comparison

4. CONCLUSION

In this paper, we presented classification of heuristics for task scheduling. Since simple heuristic has no bounds, we compare various simple heuristics viz. Max- Min, Min-Min and FCFS. The experimental results clearly revealed that Max-Min gives better results for minimizing makepan. The result indicates that Max-Min heuristic for task scheduling is a suitable selection.

5. REFERENCES

[1] Oppenheimer DM. Shah AK. Heuristics made easy: an effort-reduction framework. Psychol. Bull.

[2] Kousalya.K and Balasubramanie.P. Ant Algorithm for Grid Scheduling Powered by Local Search. International Journal of Open Problems in Computer

[3] R. Buyya, D. Abramson, J. Giddy, and H. Stockinger. Economic Models for Resource Management and Scheduling in Grid Computing. The Journal of Concurrency and Computation, 14:1507-1542, 2002.

[4] Ni L.M. Zhiwei Xu Lijuan Xiao, Yanmin Zhu. Incentive based Scheduling for Market Like Computational Grids. IEEE Transactions on Parallel and Distributed Systems, 19:903-913, 2008.

[5] Preetam Ghosh, Nirmalya Roy, Sajal K. Das, and Kalyan Basu. A Pricing Strategy for Job Allocation in Mobile Grids using a Non-cooperative Bargaining Theory Framework. Journal of Parallel and Distributed Computing, 65(11):1366 - 1383, 2005.

[6] B. Pourebrahimi, K. Bertels, G.M. Kandru, and S. Vassiliadis. Market Based Resource Allocation in Grids. e-Science and Grid Computing, International Conference on, 2006.

[7] Yang Gao, Hongqiang Rong, and Joshua Zhexue Huang. Adaptive Grid Job Scheduling with Genetic Algorithms. Future Generation Computer Systems, 21(1):151 - 161, 2005.

[8] Fatos Xhafa, Enrique Alba, Bernab Dorronsoro, and Bernat Duran. Efficient Batch Job Scheduling in Grids Using Cellular Memetic Algorithms. Journal of Mathematical Modelling and Algorithms, 7:217-236, 2008.

[9] Yuan-Shun Dai and Xiao-Long Wang. Optimal Resource Allocation on Grid Systems for Maximizing Service Reliability using a Genetic Algorithm. Reliability Engineering and System Safety, 91(9):1071 - 1082, 2006

[10] C. Fayad, J.M. Garibaldi, and D. Ouelhadj. Fuzzy Grid Scheduling Using Tabu Search. In Fuzzy Systems Conference, 2007. FUZZ-IEEE 2007. IEEE International, pages 1 -6, july.

[11] Lei Zhang, Yuehui Chen, and Bo Yang. Task Scheduling Based on PSO Algorithm in Computational Grid. International Conference on Intelligent Systems Design and Applications, 2:696-704, 2006.

[12] Li Liu, Yi Yang, Lian Li, and Wanbin Shi. Using Ant Colony Optimization for Superscheduling in Computational Grid. In Services Computing, 2006. APSCC '06. IEEE Asia Pacific Conference on, pages 539 - 545, dec. 2006.

[13] S. Fidanova. Simulated Annealing for Grid Scheduling Problem. In IEEE International Symposium on Modern Computing, 2006. John Vincent Atanaso, 2006 (JVA '06)

[14] Ruay-Shiung Chang, Jih-Sheng Chang, and Po-Sheng Lin. An Ant Algorithm for Balanced Job Scheduling in Grids. Future Generation Computer Systems, 25(1):20 - 27, 2009.

[15] L.M. Gambardella M. Dorigo. Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. IEEE Transactions on Evolutionary Computation, 1:53 - 66, 1997.

[16] T. Stutzle. MAX-MIN Ant System for Quadratic Assignment Problems. Technical Report,Intellectics Group, Department of Compute Science, Darmstadt, University of Technology, Germany, 1997.

[17] C. Strauss B. Bullnheimer, R.F. Hartl. A New Rank-based Version of the Ant System. Central European Journal for Operations Research and Economics, 1:25 - 38, 1999.

[18] L.M. Gambardella E.D. Taillard. Adaptive Memories for the Quadratic Assignment Problem. Technical Report IDSIA-87-97, IDSIA, Lugano, Switzerland, 1997.

[19] A. Colorni M. Dorigo, V. Maniezzo. The Ant System: Optimization by a Colony of Cooperating Agents. IEEE Transactions on Systems, Man, and Cybernetics, 26:29 - 41, 1996.

[20] Zhihui Du, Man Wang, Yinong Chen, Yin Ye, and Xudong Chai. The Triangular Pyramid Scheduling Model and Algorithm for PDES in Grid. Simulation Modelling Practice and Theory, 17(10):1678 - 1689, 2009.

[21] Wei wei JIANG, Hong yan CUI, and Jian ya CHEN. A Fuzzy Modeling based Dynamic Resource Allocation Strategy in Service Grid. The Journal of China Universities of Posts and Telecommunications, 16:108 - 113, 2009.

[22] Jingbo Yuan, Shunli Ding, and Cuirong Wang. Tasks Scheduling Based on Neural Networks in Grid. In Third International Conference on Natural Computation, 2007. ICNC 2007., volume 3, pages 372 -376, aug. 2007.

[23] X. Wang Y.M. Teo and J.P. Gozali. A Compensation-based Scheduling Scheme for Grid Computing. In Proceedings of the Seventh International Conference on High Performance Computing and Grid in Asia Pacific Region, 2004.

[24] Fatos Xhafa, Juan Gonzalez, Keshav Dahal, and Ajith Abraham. A GA(TS) Hybrid Algorithm for Scheduling in Computational Grids. In Hybrid Artificial Intelligence Systems, volume 5572 of Lecture Notes in Computer Science, pages 285-292. Springer Berlin / Heidelberg, 2009.

[25] Ajith Abraham, Rajkumar Buyya, and Baikunth Nath. Nature's Heuristics for Scheduling Jobs on Computational Grids. In in Proc. of 8th IEEE International Conference on Advanced Computing and Communications (ADCOM 2000), pages 45-52, 2000.

[26] Hao Tian. A New Resource Management and Scheduling Model in Grid Computing Based on a Hybrid Genetic Algorithm. In Computing, Communication, Control, and Management, 2008. CCCM '08. ISECS International Colloquium on, volume 3, pages 113-117, 2008.

[27] Wanneng Shu, Shijue Zheng, Li Gao, and Xiong Wang. An Hybrid Evaluative Algorithm Applied to Task Scheduling. In International Conference on Communications, Circuits and Systems Proceedings, 2006, volume 3, pages 2070 - 2073, june 2006.

[28] J. Kolodziej and F. Xhafa. A Game-Theoretic and Hybrid Genetic Meta-Heuristics Model for Security-Assured Scheduling of Independent Jobs in Computational Grids. In Complex, Intelligent and Software Intensive Systems (CISIS), 2010 International Conference on", month="February", pages="93 -100", year="2010".

[29] Yuan-Shun Dai and Xiao-Long Wang. Optimal Resource Allocation on Grid Systems for Maximizing Service Reliability using a Genetic Algorithm. Reliability Engineering and System Safety, 91(9):1071 - 1082, 2006.

[30] Helen D. Karatza Sofia K. Dimitriadou. Multi-Site Allocation Policies on a Grid and Local Level. Electronic Notes in Theoretical Computer Science, 261:163 - 179, 2010.

International Journal of Computer Applications and Technology
Volume 1- Issue 2, 2012, 53-56

International Journal of Computer Applications and Technology
journal homepage: www.ijcat.com

Audio-Video Based Segmentation and Classification using AANN

K. Subashini
Dept of Comp Sci. and Engg.
Annamalai University
Chidambaram, India

S. Palanivel
Dept of Comp Sci. and Engg.
Annamalai University
Chidambaram, India

V. Ramaligam
Dept of Comp Sci. and Engg.
Annamalai University
Chidambaram, India

Abstract: This paper presents a method to classify audio-video data into one of seven classes: advertisement, cartoon, news, movie, and songs. Automatic audio-video classification is very useful to audio-video indexing, content based audio-video retrieval. Mel frequency cepstral coefficients are used to characterize the audio data. The color histogram features extracted from the images in the video clips are used as visual features. Auto associative neural network is used for audio and video segmentation and classification. The experiments on different genres illustrate the results of segmentation and classifications are significant and effective. Experimental results of audio classification and video segmentation and classification results are combined using weighted sum rule for audio-video based classification. The method classifies the audio-video clips with effective and efficient results obtained.

Keywords: Mel frequency cepstral coefficients, color histogram, Auto associative neural network, audio segmentation, video segmentation, audio classification, video classification, audio-video Classification and weighted sum rule.

1. INTRODUCTION

To retrieve the user required information in huge multimedia data stream an automatic classification of the audio-video content plays major role. Audio-video clips can be classified and stored in a well organized database system, which can produce good results for fast and accurate recovery of audio-video clips. The above approach has two major issues: feature selection and classification based on selected features. Recent years have seen an increasing interest in the use of AANN for audio and video classification.

2. Outline of the work

This work presents a method for audio-video segmentation and classification. The paper is organized as follows. The acoustic and visual feature extractions are presented in section 4. Modeling techniques for audio and video segmentation and classification are described in section 5. Experimental results of audio- video segmentation and classification are reported in section 6. Conclusion is given in section 7.

3. Related Work

Recent study shows that the approach to automatic audio classification uses several features. To classify speech/music element in audio data stream plays an important role in automatic audio classification. The method described in [1] uses SVM and Mel frequency cepstral coefficients, to accomplish multi group audio classification and categorization. The method gives in [11] uses audio classification algorithm that is based on conventional and widely accepted approach namely signal parameters by MFCC followed by GMM classification. In [6] a generic audio classification and segmentation approach for multimedia indexing and retrieval is described. Musical classification of audio signal in cultural style like timber, rhythm, and wavelet confident based musicology feature is explained in [5]. An approach given in [8] uses support vector machine (SVM) for audio scene classification, which classifies audio clips into one of five classes: pure speech, non pure speech, music, environment sound, and silence.

Automatic video retrieval requires video classification. In [7], surveys of automatic video classification features like text, visual and large variety of combinations of features have been explored. Video database communication widely uses low-level features, such as color histogram, motion and texture. In many existing video data base management systems content-based queries uses low-level features. At the highest level of hierarchy, video database can be categorized into different genres such as cartoon, sports, commercials, news and music and are discussed in [13], [14], and [15]. Video data stream can be classified into various sub categories cartoon, sports, commercial, news and serial are analysis in [2], [3], [7] and [16]. The problems of video genre classification for five classes with a set of visual feature and SVM is used for classification is discussed in [16].

4. Feature Extraction

4.1 Acoustic Feature

The computation of MFCC can be divided into five steps.

1. Audio signal is divided into frames.
2. Coefficients are obtained from the Fourier transform.
3. Logarithm is applied to the Fourier coefficients.
4. Fourier coefficients are converted into a perceptually based spectrum.
5. Discrete cosine transform is performed.

In our experiments Fourier transformation uses a hamming window and the signal should have first order pre-emphasis using a coefficient of 0.97. The frame period is 10 ms, and the window size is 20 ms. to represent the dynamic information of the features, the first and second derivatives, are appended to the original feature vector to form a 39 – dimensional feature.

4.2 Visual Feature

Color histogram is used to compare images in many applications. In this work, RGB (888) color space is quantized into 64 dimensional feature vector, only the dominant top 16 values are used as features. The image/video histogram is a simply bar graph of pixel intensities. The pixels are plotted along the x – axis and the number of occurrences for each intensity represent the y-axis.

$p(r_k) = n_k / n, \ 0 \le k \le L-1$ (1)

where

r_k – kth gray level

n_k – Number of pixels in the image with that gray level

L – Number of levels (16)

n – Total number of pixels in the image

$p(r_k)$ – gives the probability of occurrence of gray level r_k.

5. Autoassociate Neural Network (AANN)

Autoassociative neural network models are feedforward neural networks performing an identity mapping. The modality would be the ablity to solve the scaling problem. The AANN is used to capture the distribution of the input data and learning rule . Let us consider the five layer AANN model shown in Fig1, which has three hidden layers. The processing units in the first and third hidden layers are non-linear, and the units in the second compression/hidden layer can be linear or non-linear. As the error between the actual and the desired output vectors is minimized, the cluster of points in the input space determines the shape of the hypersurface obtained by the projection onto the lower dimensional space. A five layer autoassociative neural network model is used to capture the distribution of the feature vectors. The second and fourth layers of the network have more units than the input layer. The third layer has fewer units than the first or fifth. The activation functions at the second, third and fourth layers are non-linear. The non-linear output function for each unit is tanh{s}, Where s is the activation value of the unit. The standard backpropagation learning algorithm is used to adjust the weights of the network to minimize the mean square error for each feature vector. The AANN captures the distribution of the input data depending on the constraints imposed by the structure of the network, just as the number of mixtures and Gaussian functions do in the case of Gaussian mixture model.

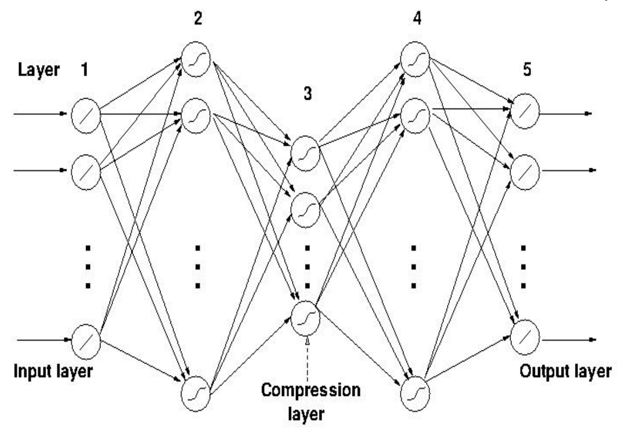

Fig.1. A Five Layer AANN model

6. Experimental Results

Performance of the proposed audio-video segementation and classification system is evaluated using the Television broadcast audio database collected from various channels and various genres. Audio samples are of different length, ranging from 2seconds to 6seconds, with a sampling rate of 8 kHz, 16-bits per sample, monophonic and 128 kbps audio bit rate. The waveform audio format is converted into raw values (conversion from binary into ASCII). Silence segments are removed from the audio sequence for further processing 39 MFCC coefficients are extracted for each audio clip as described in Section 4.1. A non-linear support vector classifier is used to discriminate the various categories. The training data is segmented into fixed-length and overlapping frames (in our experiments we used 20 ms frames with 10 ms frame shift.)

The distribution of 39 dimensional MFCC feature vectors in the feature space and 64 dimensional feature vectors is capture the dimension of feature vectors of each class. The acoustic feature vectors are given as input to the AANN model and the network is trained of 100 epochs. One epoch of training is a single presentation of all training vector. The training takes about 2 mints on a pc with dual core 2.2 GHz CPU.

6.1 Audio and video data for segmentation

The experiments are conducted using the television broadcast audio-video data collected from various channels(both Tamil and English) evaluation database. A total dataset of 50 recorded is used in our studies. This includes 10 datasets for each dual combination of dataset such as news flowed by advertisement, advertisement followed by sports ect. The audio is sampled at 8 kHz and encoded by 16-bit. Video is recorded with resolution 320*240 at 25 fbs. The category change points are manually marked. The manual segmentation results are used as the reference for evaluation of the proposed audio-video segmentation method. A total of 1,800 audio segments and 3,600 are marked in the 50 datasets. Excluding the silence periods for audio signal, the segment duration is mostly between 2 to 6 seconds.

6.2 Feature Representation

The extraction of MFCC features is based on first pre-emphasising the input speech data using a first order digital filter and then segmenting it into 20 ms frames with an overlap of 50% between adjacent frames using Hamming window. For each frame the first 13 cepstral coefficients other than the zeroth value are used. The color histogram is obtained from the video signal using 16 order analysis. The extraction of color histogram is based on first pre-emphasising the input video data using a first-order digital filter and then segmenting it into 20 ms frames with an overlap of 50% between adjacent frames using Hamming window. The color histogram feature is computed for the entire video signal using the method described in above Section. For each frame 16 samples around the highest Hilbert envelope is extracted.

6.3 Audio and Video Segmentation

The tests in this experimental investigation are conducted using the procedure mentioned in above Section. The procedure is applied for MFCC features and color histogram separately in order to locate the category change frames. The 200 frames analysing window size is used in our experiments.

The proposed audio (video) segmentation uses a sliding window of about 2 sec assuming the category change point occurs in the middle of the window. The sliding window is intially placed at the left end of the audio (video) signal. The AANN model is trained to capture the distribution of the feature vectors in the left half of the window, and the feature vectores in the right half of the window are used for testing as shown in Fig.2.

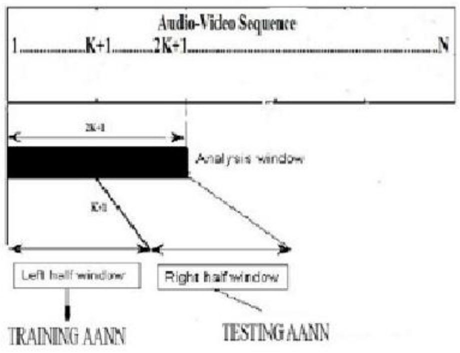

Fig.2. Proposed Segmentation Algorithm Using AANN

The out put of the model is compared with the input to compute the narmalized squared error (e_i) for the ith feature vector ($\mathbf{y}_i$) is given by,

$$e_k=|x_i-o_i|^2 \ / \ 2 \qquad (9)$$

where o_i is the output vector given by the model.

The error e_k is transformed into a confidence score s using

$s = \exp(-e_k)$

Average confidence score is obtained for the right half of the window. A low confidence score indicates that the characteristics of the audio(video) signal in the right half of the window are different from the signal in the left half of the window, and hence, the middle of the window is a category change point. The above process is repeated by moving the window with a above progress is repeated by moving the window with a shift of the about 80msec until it reaches the right end of the audio(video) signal.

6.4 Combining Audio-Video Segmentation

The evidence from audio and video segmentation from AANN are combined using weighted sum rule. The weighted sum rule states that "`If the category change point is detected at t_1 from the audio and at t_2 is within a threshold t then the category change point is fixed at t_1+t_2/2".

6.5 Audio and Video Classification

For evaluating the performance of the system, the feature vector is given as input to each of the model. The output of the input to compute the normalized squared error. The normalized squared error E for the feature vector y is given by $E = \frac{||y-0||^2}{||y||^2}$, where 0 is the output vector given by the model. The error E is transformed into a confidence score c using c=exp (-E). Similarly the experiments are conducted using histogram as features in video classification. Compared to audio classification, video classification is more complicated. The memory used for video classification is twice that used for audio classification. From the results, we observe that the overall classification accuracy is beterr using color histogram as feature and Mel-frequency cepstral coefficients

6.6 Combining Audio and Video Classification

In this work, combining the modalities has been done at the score level. The methods to combine the two levels of information present in the audio signal and video signal have been proposed. The audio based scores and video based scores are combined for obtaining audio-video based scores as given equation (10). It is shown experimentally that the combined system outperforms the individual system, indicating complementary nature. The weight for each modality is decided empirically. The weight for each modality is decided empirically.

$$S = \frac{w}{n}\sum_{i=1}^{n} a_i + \frac{(1-w)}{p}\sum_{i=1}^{p} v_i$$

$$1 \le j \le c \qquad (10)$$

Where

$$a_j = \sum_{i=1}^{n} x_i^j \qquad 1 \le j \le c$$

$$v_j = \sum_{i=1}^{p} y_i^j \qquad 1 \le j \le c$$

$$x_i^j = \begin{cases} 1 & \text{if } \quad q_i = j \\ 0 & \text{otherwi} \end{cases}$$

$$1 \le i \le n \ , \quad 1 \le j \le c$$

$$y_i^j = \begin{cases} 1 & \text{if } \quad c_i^v = j \\ 0 & \text{otherwi} \end{cases}$$

$$1 \leq i \leq p \quad , \quad 1 \leq j \leq c$$

s -is the combinined audio and video confidence score.

s_i^a -is the Confidence score rate of the ith audio frame.

s_i^v -is the Confidence score rate of the ith video frame.

v_j - Video based score for jth frame.

a_j -Audio based score for jth frame.

m_j - Audio-video based score for jth frame.

c -number of classes.

n -number of audio frames.

p -number of video frames.

w -weight.

The category is decided based on the highest confidence score various from 0 to 1. Audio and video frames are combined based on 4:1 ration of frame shifts. The weight for each of modality is decided by the parameter w is chosen such that the system gives optimal performance for audio-video based classification.

7. CONCLUSION

This paper proposed an automatic audio-video based segmentation and classification using AANN. Mel frequency cepstral coefficients are used as features to characterize audio content. Color Histogram coefficients are used as features to characterize the video content. A non linear support vector machine learning algorithm is applied to obtain the optimal class boundary between the various classes namely advertisement, cartoon, sports, songs by learning from training data. An experimental result shows that proposed audio-video segmentation and classification gives an effecttive and efficient results obtained.

8. REFERENCES

[1] Dhanalakshmi. P.; Palanivel. S.; and Ramaligam. V.; (2008), "Classification of audio signals using SVM and RBFNN", In Elsevier, Expert systems with application, Vol. 36, pp. 6069–6075.

[2] Kalaiselvi Geetha. M.; Palanivel. S.; and Ramaligam. V.; (2008), "A novel block intensity comparison code for video classification and retrivel", In Elsevier, Expert systems with application, Vol. 36, pp 6415-6420.

[3] Kalaiselvi Geetha, M.; Palanivel, S.; and Ramaligam, V.; (2007), "HMM based video classification using static and dynamic features", *In proceedings of the IEEE international conference on computational intelligence and multimedia applications.*

[4] Palanivel. S.; (2004)., "Person authentication using speech, face and visual speech", *Ph.D thesis, I IT, Madras.*

[5] Jing Liu.; and Lingyun Xie.; "SVM-based Automatic classification of musical instruments", *IEEE Int'l Conf., Intelligent Computation Technology and Automation (2010.),* vol. 3, pp 669–673.

[6] Kiranyaz. S.; Qureshi. A. F.; and Gabbouj. M. ; (2006), "A Generic Audio Classification and Segmentation approach for Multimedia Indexing and Retrieval"., *IEEE Trans. Audi., Speech and Lang Processing,* Vol.14, No.3, pp. 1062–1081.

[7] Darin Brezeale and Diane J. cook, Fellow. IEEE (2008), "Automatic video classification: A Survey of the literature", *IEEE Transactions on systems, man, and cybernetics-part c: application and reviews,* vol. 38, no. 3, pp. 416-430.

[8] Hongchen Jiang. ; Junmei Bai. ; Shuwu .Zhang. ; and BoXu. ; (2005)," SVM - based audio scene classification", *Proceeding of NLP-KE,* pp. 131–136.

[9] V. Vapnik.;"Statistical Learning Theory", *John Wiley and Sons,* New York, 1995.

[10] J.C. Burges Christophe.; "A tutorial on support vector machines for pattern recognition," Data mining and knowledge discovery, No. 2, pp. 121–167, 1998.

[11] Rajapakse. M .; and Wyse. L.; (2005), "Generic audio classification using a hybrid model based on GMMs and HMMs ",*In Proceedings of the IEEE ,pp-1550-1555.*

[12] Jarina. R.; Paralici. M.; Kuba. M.; Olajec. J.; Lukan. A.; and Dzurek. M.; "Development of reference platform for generic audio classification development of reference plat from for generic audio classification", *IEEE Computer society, Work shop on Image Analysis for Multimedia Interactive (2008),* pp-239–242.

[13] Kaabneh,K. ; Abdullah. A.; and Al-Halalemah,A. (2006). , "Video classification using normalized information distance", In *proceedings of the geometric modeling and imaging – new trends* (GMAP06) (pp. 34-40).

[14] Suresh. V.; Krishna Mohan. C.; Kumaraswamy. R.; and Yegnanarayana. B.; (2004).,"Combining multiple evidence for video classification", *In IEEE international conference Intelligent sensing and information processing (ICISIP-05),* India (pp.187–192).

[15] Gillespie. W. J.; and Nguyen, D.T (2005).; "Hierarchical decision making scheme for sports video categorization with temporal post processing", *In Proceedings of the IEEE computer society conference on computer vision and pattern recognition* (CVPR04) (pp. 908 -913).

[16] Suresh. V.; Krishna Mohan. C.; Kumaraswamy. R.; and Yegnanarayana. B.; (2004).,"Content-based video classification using SVM", *In International conference on neural information processing,* Kolkata (pp. 726–731).

[17] Subashini, K.; Palanivel, S.; and Ramaligam, V.; (2007), "Combining audio-video based segmentation and classification using SVM", *In International journal of Signal system control and engineering applications,* Vol.14,Issue.4, pp. 69–73.

International Journal of Computer Applications and Technology
Volume 1- Issue 2, 2012, 57-63

International Journal of Computer Applications and Technology
journal homepage: www.ijcat.com

An Improved Adaptive Space-Sharing Scheduling Policy for Non-dedicated Heterogeneous Cluster Systems

Amit Chhabra
Department of Computer Science & Engineering
Guru Nanak Dev University,
Amritsar, INDIA

Gurvinder Singh
Department of Computer Science & Engineering
Guru Nanak Dev University,
Amritsar, INDIA

Abstract: Adaptive space-sharing scheduling algorithms tend to improve the performance of clusters by allocating processors to jobs based on the current system load. The focus of existing adaptive algorithms is on dedicated homogeneous and heterogeneous clusters. However commodity clusters are naturally non-dedicated and tend to be heterogeneous over the time as cluster hardware is usually upgraded and new fast machines are also added to improve cluster performance. The existing adaptive policies for dedicated cluster systems are not suitable for such conditions. Moreover existing adaptive policies use First-come-first-serve (FCFS) which is known to be sensitive of variance in service demand, as a job-selection policy for processor allocation. FCFS allocation of processors to jobs results in a situation where small jobs could be blocked by an earlier arrived large job. This paper fills these gaps by designing an efficient adaptive space-sharing scheduling algorithm for non-dedicated heterogeneous cluster systems. Evaluation results show that the proposed algorithm provide substantial improvement over existing algorithms at moderate to high system utilizations.

Keywords: First-come-first-serve, Adaptive space-sharing scheduling, Cluster computing systems, Non-dedicated heterogeneous clusters, and Mean response time

1. INTRODUCTION

In recent times, paradigm of parallel processing has been shifted from traditional expensive multiprocessors to commodity-based high-performance cluster computing systems due to their high-performance and cost-effectiveness. Depending upon the ownership, cluster systems can be classified into two categories; dedicated cluster systems and non-dedicated cluster systems. Dedicated cluster uses a network of dedicated PCs collectively to form an effective high-performance parallel solution. On the other hand, non-dedicated cluster aims to utilize the abundant computing cycles "available" on the network of PCs to provide high-computing power. Computers in the non-dedicated clusters are privately owned and likely to be heterogeneous. The heterogeneity and "availability" of processing power in non-dedicated environment distinguishes itself from dedicated cluster systems. Community-owned cluster computing (CCC) systems [1][2] are an example of non-dedicated heterogeneous clusters.

One of the most important challenges that must be addressed in order to realize the fullest advantages of *Cluster computing systems* is that of designing efficient job scheduling algorithms. Space-sharing policies are commonly used to schedule parallel jobs in distributed-memory cluster systems. In space-sharing policy, parallel system of multiple processors is divided into disjoint set of processors (known as *partitions*) so that each partition can be assigned to a single job. In this way, number of jobs can be executed side-by-side by simultaneously providing processor partitions. The number of processors in each partition to be assigned to a job is known as *partition size*. The primary reason for preferring space-sharing over time-sharing for cluster systems is to avoid the cost of context switching due to frequent preemptions in time-sharing systems.

Space-sharing policies can be broadly divided into fixed, variable, adaptive and dynamic policies [3][4] based on the decision that whether the partition size once assigned to the jobs can be changed during execution time or not. In fixed policies, partition sizes are fixed by the administrator before the system actually starts operating and any modification to these partition sizes require a system reboot. Variable policies require partition sizes to be specified by the user at the time of job arrival. In adaptive policies, partition sizes are determined by the scheduler at the time of job scheduling on the basis of current system load and any available job characteristics. However partition size once assigned to a job can not be changed during job execution. In dynamic policies, partition size of a job can be changed during its execution.

High performance applications for cluster computing systems are mostly presented as *parallel jobs.* A parallel job is said to consist of a set of tasks/processes running concurrently to achieve a certain common objective. Each task runs to completion on its assigned processor. The number of tasks (and hence processors required) a certain job has is referred to as the job *size.*

Characteristics of on-line job streams that act as input workload to the job schedulers influence the performance of the schedulers. Parallel jobs can be classified into four types [3][4]; (i) Rigid, (ii) Moldable (iii) Evolving, and (iv) Malleable, depending upon the number of processor to be allocated at submission time or during execution. A rigid job demands a fixed number of processors at the time of submission and executes on these processors exclusively until completion. Moldable jobs can be made to execute on different number of processors based on the current system load. For example if system load is high, then few processors can be assigned to the moldable job and if system load then large number of processors can be allotted to the job. However this flexibility is only available at job start time and partition size cannot be reconfigured during execution. The processor requirements of both evolving and malleable jobs can be changed during execution. For evolving jobs, requirement changes are initiated by the application itself during the various phases of its execution. If the system cannot satisfy the job's demand, the job has to wait for exact processor allocation. For malleable jobs, the decision to change the number of processors is made by an external job scheduler.

Adaptive policies perform better than fixed-partitioning and variable-partitioning scheduling policies due to their ability adapt to the current load on the system while calculating partition-size for jobs. Adaptive space-sharing scheduling policies to schedule moldable jobs are widely studied in homogeneous parallel systems (i.e. multiprocessors and clusters) [5-12] and to less extent in heterogeneous cluster computing systems [2][13]. A common assumption in the existing adaptive policies in both the systems has been that all processors in the system are dedicated to only parallel workload. It means that processors in the system are not shared simultaneously with the local jobs executing at individual processor. In this paper we focus on proposing a scheduling algorithm to allocate processors to jobs in a non-dedicated heterogeneous cluster computing environment.

The rest of the paper is organized as follows: Section 2 states the problem statement. Section 3 gives an overview of previous literature work related to the problem. Section 4 describes the details of the proposed solution. Section 5 describes simulation model which discusses the workload and system model used. Section 6 evaluates the performance of new policies and compares them with existing solutions and Section 7 concludes the paper.

2. PROBLEM STATEMENT

The research problem chosen in this paper seems significant as the partition sizes obtained in non-dedicated heterogeneous cluster systems will be different from those obtained in dedicated homogeneous systems. When we partition a dedicated homogeneous cluster, partition size is obtained by dividing total number of physical processors by the total number of jobs in the system. But in case of non-dedicated heterogeneous systems, partition size is calculated by dividing the total available computing power of all processors by the number of jobs currently available in the system. However total available computing power will be different at different times due to variations in the computing power of individual processors in the presence of varying background workload. Hence corresponding calculated partition size changes continuously. Moreover existing adaptive policies focus on using FCFS as a job selection algorithm to allocate processors to parallel jobs. FCFS is known to be sensitive to the variance in the service times which means that large number of smaller jobs can be blocked by few larger jobs that have arrived earlier.

Therefore an efficient adaptive scheduling policy is required which can take care of heterogeneity of processor speeds as well as run-time load variations due to background workloads executing at individual processors and which is relatively less sensitive to the variance in the service times.

3. RELATED WORK

The focus of the current job scheduling research in distributed-memory multiprocessors and cluster systems is towards adaptive algorithms to schedule moldable jobs [5-12] as they have shown to achieve better mean response time than the scheduling algorithms for rigid jobs. This is due to the fact that adaptive algorithms decide the partition sizes by adapting to current system load at job scheduling time whereas rigid jobs only require a fixed number of processors resulting into increased processor fragmentation and mean response times. Dynamic policies are shown to more suitable for shared-memory parallel systems in which the associated overheads of dynamic-partitioning are outweighed by the benefits.

Adaptive scheduling algorithms for assigning partition sizes to moldable jobs have been extensively studied in homogeneous parallel systems and to less extent in heterogeneous parallel systems [2][13]. Existing adaptive algorithms in both homogeneous and heterogeneous cluster systems share one common assumption that processors are dedicated to execute only cluster applications (no other applications can be executed locally). Available adaptive policies also differ from each other by the amount of job characteristics used in making processor allocation decisions.

In [5-6] , Rosti et al. introduced several adaptive partitioning policies (known as Fixed Processors per Job (FPPJ)), Equal Partitioning with a Maximum (EPM), Insurance Policy and Adaptive Policies (known as AP1, AP2, AP3, AP4 and AP5)) for distributed-memory multiprocessors over a wide range of workload types and with different possible arrival rates. These policies try to allocate equal-sized partitions to the waiting applications since no a priori job characteristics were assumed to be available. However these policies differ from each other in how the target partition-size is computed.

Out of these adaptive policies, AP2 (known as work-conserving policy) seems to be an interesting policy that reserves one additional partition for the future job arrivals. The partition size in the AP2 policy is calculated as shown in (1).

$$Partition\ Size\ (PS) = max\left(1, ceil\left(\frac{total\_procesors}{Waiting\_jobs+1} + 0.5\right)\right) \quad (1)$$

In [7], Dandamudi and Yu show that AP2 considers only queued jobs to calculate partition size. This will lead to a situation that contravenes the principal of allocating equal-sized partitions to all jobs. Dandamudi and Yu, suggested a modified version of AP2 known as Modified adaptive policy (MAP) which considers waiting as well as running jobs to calculate partition size as shown in (2).

$$Partition\ Size\ (PS) = max\left(1, ceil\left(\frac{total\_processors}{Waiting\_jobs+(f*Running\_jobs)+1} + 0.5\right)\right) \quad (2)$$

Target partition size to be finally allocated to the waiting job is calculated using equation (3). It is the minimum of the partition size calculated using equation (2) and maximum parallelism of the job.

$$Traget\ partition\ size = min(PS, maximum\ parallelism\ of\ the\ job) \quad (3)$$

The parameter f (whose value lies between 0 and 1) is used to control the contribution of the “running” jobs to the partition size. It has been shown that the MAP policy provides significant improvement in performance over policies like AP2, ASP and ASP-max etc. that do not consider the contribution of running jobs while calculating partition size. The amount of improvement obtained is a function of parameter f, system load, and workload.

The adaptive policy proposed in [8][10] is more restrictive, in that users must specify a range of the number of processors for each job. Availability of service demand knowledge of an individual job is assumed in the paper. Schedulers will select a number which gives the best performance. Schedulers in [8][10] use a submit-time greedy strategy to schedule moldable jobs.

In [11], Srinivasan et al. have some improvement to [1][3]: (i) using schedule time-scheduler which defers the choice of partition size until the actual job schedule time instead of job submission time and, (ii) using aggressive backfilling instead of conservative backfilling.

In [12], Srinivasan et al. argue that an equal-sized partition strategy tends to benefit jobs with small computation size (light jobs). On the other hand, allocating processors to jobs proportional to the job computation size tends to benefit heavy jobs significantly. A compromise policy is that each job will have a partition size proportional to the square root of its computation size (Weight) as in (4). This equation is used to calculate partition size in an enhanced backfilling scheme proposed in [12].

$$WeightFraction_i = \frac{\sqrt{Weight_i}}{\sum_{i \in \{ParallelJobInSystem\}} \sqrt{Weight_i}} \qquad (4)$$

In [2], a variation of MAP, known as Heterogeneous Adaptive Policy (HAP) was suggested by Dandamudi and Zhou to work with heterogeneous parallel systems. The work introduced the concept of Basic Processor Unit (BPU) to differentiate the heterogeneous processors from each other. Partition sizes are allocated to the jobs on the basis of their computation power in terms of number of BPUs rather than using a physical processor level as in homogeneous systems. The research paper showed the supremacy of HAP over MAP and AP2 policies. Partition size in HAP is calculated as in equation (5) and target partition size is calculated using equation (3).

$$Partition\ Size\ (PS) = max\left(1, ceil\left(\frac{total\_BPUs}{Waiting\_jobs + (f * Running\_jobs) + 1} + 0.5\right)\right) \quad (5)$$

In [13], Shim suggested various adaptive policies for shared heterogeneous network of workstations (NOW) considering the priority of sequential local jobs as well as the parallel jobs. No in-depth details about the working of the algorithms are provided in the paper and no comparisons are made with the existing policies. The shortcoming of this paper is that it considers only the contribution of waiting jobs to calculate the partition size which usually lead to worse results.

In [14], Doan et al. suggested priority-based adaptive policy for homogeneous PC-based cluster systems for both rigid and moldable jobs. The user can assign priority to both types of jobs. The jobs with higher priority are given preference in execution. Since rigid jobs require the fixed number of processors (e.g. partition size), so partition-function for only moldable parallel jobs is derived from equation (2) given in [7].

In [15], Abawajy proposed another adaptive policy known as SOUL for heterogeneous multi-cluster systems which calculates partition size on the basis of mean service rate of heterogeneous processors, local load at processors and maximum parallelism information of waiting jobs. It has been shown that SOUL policy tends to produce shorter mean job response times as compared to both AEP and MAP at various workloads. But no comparison between HAP and SOUL policy is available in literature.

4. PROPOSED ADAPTIVE POLICY

From the literature survey, following lessons have been learnt which will help us to design a robust adaptive policy for non-dedicated heterogeneous parallel systems.

1) Adaptive policies which consider both current waiting and running jobs in the parallel system perform better than those policies which consider only current waiting jobs.
2) In heterogeneous systems, BPU mechanism is used frequently to differentiate the computing power of different physical processors.
3) When no job knowledge or only maximum parallelism information is available, equal-sized (or equivalent) partitioning mechanism is preferred over weighted square-root fair-share strategy which requires the service demand knowledge of jobs.

4.1 An Improved Heterogeneous Adaptive Policy (IHAP)

Using these observations and lessons, we have suggested few modifications to HAP policy which have shown good results over various policies in dedicated heterogeneous systems. The new policy is named as an Improved Heterogeneous Adaptive Policy (IHAP) to schedule jobs in non-dedicated heterogeneous cluster environment and requires only maximum parallelism (Pmax) information of jobs to calculate final target partition size for the current waiting jobs.

Partitioning-function of IHAP:

Since cluster processors can be shared between local and parallel jobs, therefore at any point of time, current available computing power for execution of parallel workload at each processor in the presence of local workload is given as in equation (6).

$$Computing\ power\ (CP_k) = BPU_k * (1 - Local\_load_k) \qquad (6)$$

In a cluster system with P processors, BPU_k represents the computing power of kth processor and $Local\_load_k$ denotes the load at individual processor due to the execution of local jobs.
Ideal partition size in IHAP is then calculated on the basis of current available computing as shown in (7).

$$Partition\ Size\ (PS) = max\left(1, ceil\left(\frac{\sum_{k=1}^{P} BPU_k * (1 - Local\_load_k)}{Waiting\_jobs + (f * Running\_jobs) + 1} + 0.5\right)\right) \quad (7)$$

It should be noted that job scheduler is invoked only at arrival and departure time of jobs. Information about local load and computing power of each processor is also collected by the job scheduler at these times. The number of BPUs finally allocated is calculated as follows in (8).

$$Traget\ partition\ size = min(PS, P_{max}) \qquad (8)$$

Job-selection rule of IHAP:

It should be noted that jobs are selected for processor-allocation from the waiting queue using Fit-Processors-First-Served (FPFS) as opposed to FCFS used in many adaptive policies [2][7][13]. Partition-size for the waiting jobs is calculated using equation (7) and (8). If the idle BPUs are less than the target partition-size for the current job, then next job from the waiting queue is searched who's target partition-size fits into the idle BPUs.

5. SIMULATION MODEL

We have implemented a discrete event simulator in VB.Net language to evaluate the performance of proposed adaptive scheduling algorithms under various workload conditions. Simulation modeling is preferred over the actual experimentation as it gave us the greater flexibility of covering a wide range of application characteristics and controlled parameters like arrival rates, system utilization etc. and allowed us to abstract away trivial details of the environment under study, which otherwise would complicate the performance evaluation procedure.

The developed simulator takes the on-line job stream as input parallel workload, executes parallel workload with the specified adaptive policy and generates the output in the form of mean response time. Response time of a job is defined as the sum of its execution time and waiting time. Waiting time of job is the difference between job arrival time and job scheduling time. Execution time is the actual time spent to execute the job.

5.1 System Model

We have used an open system model of community-owned cluster of 64 independent commodity single-processor personal computers and each computer is used in a shared mode i.e. it is able to service local sequential tasks as well as the tasks of parallel job submitted by the central job scheduler. The computers differ from each other in terms of heterogeneity in processor speeds i.e. computing power they possess. Computer and processor terms are used interchangeably in context of this paper. We assume that computers in the cluster are connected using 100Mbps Ethernet switch. Relative computing power of different physical processors is represented in terms of Basis Processing Unit (BPU) [2] which can either be derived with the help of SPECfp2000 ratings based on the processor speeds or by executing independent benchmarking programs on heterogeneous processors. We have used two types of processors in the computers of cluster system; First 32 computers contain Type I processors; Next 32 computers contain Type II processors that are twice faster than Type I processors. Hence each processor in Type I has 1 BPU and Type II processor has 2 BPUs.

5.2 Parallel Workload Model

Parallel workload model containing online stream of parallel jobs for scheduling contains three components; 1) job arrival process 2) Maximum parallelism and 3) job service demand. The job arrival process is characterized by job arrival rate (λ) and coefficient of variation of inter-arrival times (CVa). High arrival rate represents that inter-arrival time between successive jobs is small. We have modeled the job arrival process using exponential distribution with CVa equal to one.

Maximum parallelism of jobs (Pmax) indicates the maximum number of processors that can be effectively utilized by the parallel jobs. Pmax is varied from 1 to 32 using uniform distribution. Mean service demand (D) parameter is the uncorrelated cumulative mean service demand which represents the total time required to execute the job in a dedicated environment, independent of how many processors are used. Service demand of jobs is generated using 2-stage hyper-exponential distribution with coefficient of variation of service demand (CVs) greater than one. Since moldable jobs can be made to run on the varying number of processors, therefore time (tj) taken by the parallel job varies based on the number of processors (pj) assigned to it when the job starts executing. It should be noted that dj= (tj)*(pj) as we have ignored the communication and synchronization overheads, when overall mean service demand of a parallel job (dj) is distributed equally among tasks (which are always equal to "pj" processors assigned to the job) of the job.

5.3 Background Workload Model

We assume abstract model for representing load due to background jobs at each processor by hiding the internal details of arrival and execution times of sequential local jobs. Each cluster processor is assumed to service a stream of background jobs that arrive at individual computers independently. Local load at each processor indicates the load due to the execution of sequential local jobs. As the local load increases, computing power available to service parallel workload decreases. We model the local load using uniform distribution ranging from 0% to 30% and this information is assumed to be available to job scheduler at job arrival and departure times.

6. Performance Evaluation and Results

In this section we will evaluate the performance of proposed algorithms in terms of mean response time and also compare the simulation results with the existing approaches. In all the simulation experiments performed in this paper, 31 batches of 7000 jobs per each batch were used and results of first batch were discarded to ignore start-up effects. The number of batches is such that the mean response times obtained have relative errors not exceeding 5% under the 90% confidence interval. The default parameters and values used in simulation experiments are for various simulation parameters shown in table 1.

Table 1: Default parameters and values used in experiments

Parameters of Parallel Jobs	Values
Mean service demand (D)	16
Coefficient of variation (CV_a) of Job arrival	1
Coefficient of variation (CV_s) of Service demand	4
Number of processors in the cluster	64
Pmax	32

Average load or utilization of the cluster system due to parallel jobs is derived using equation (9) as follows:

$$Average\ utilization = \frac{Job\ arrival\ rate * Mean\ service\ demand}{Number\ of\ processors} \qquad (9)$$

6.1 Relative performance of the scheduling policies

In this section we compare the performance of the proposed adaptive scheduling policy i.e. IHAP with the HAP and MAP policy. The default value of 'f' in the partitioning-function for IHAP, HAP and MAP policies is set to 0.5 which is suggested as a reasonable value in existing similar research works [2][7].

IHAP policy tends to produce shorter MRT values at system loads of interest (i.e. at medium to high loads) as shown in figure 1. This is due to two reasons; 1) IHAP policy produce smaller partition sizes as compared to both HAP and MAP as it considers the background workload into account. 2) FPFS job-selection policy reduces processor fragmentation which exists in HAP and MAP policies due to use of FCFS as a job-selection policy.

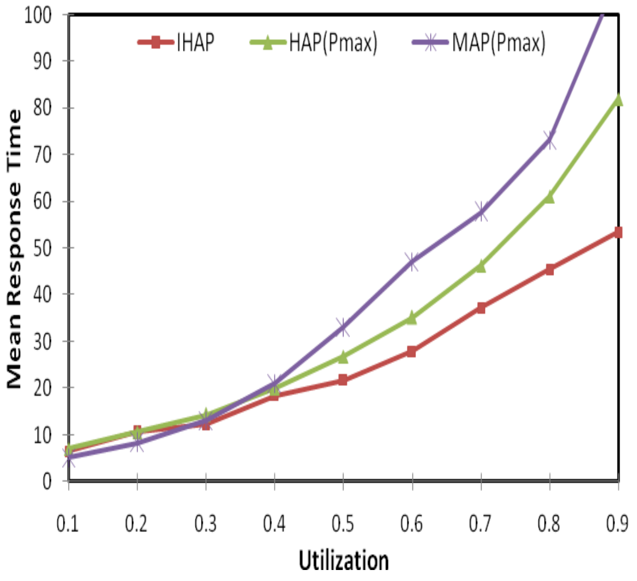

Figure 1: Performance of the scheduling policies

On the other hand, both HAP and MAP try to allocate larger partition sizes since they are not aware of any background workload. But in reality the total available computing power of all processors is much less than that of assumed by MAP and HAP. Therefore jobs have to wait for a long time to receive calculated partition sizes. HAP and MAP policies also tend to produce bigger partition sizes at low to medium system utilization since they impose no upper limit on the number of processors to be allocated to jobs. This will apparently result into allocation of large partition sizes to even smaller jobs.

6.2 Sensitivity Analysis

In this section, we study the sensitivity of the three policies to variances in inter-¬arrival and service times. When the arrival CV is varied, the service CV is held at 4. Similarly arrival CV is fixed at 1 when the performance sensitivity to service time CV is studied. The system utilization for parallel load is fixed at 80%.

6.2.1 Sensitivity to Arrival Time Variations

The performance sensitivity of the three policies to inter-arrival CV is shown in figure 2. The mean response time increases with increasing inter-arrival CV for the three policies. The IHAP policy maintains its performance superiority over HAP and MAP policy at 80% system utilization.

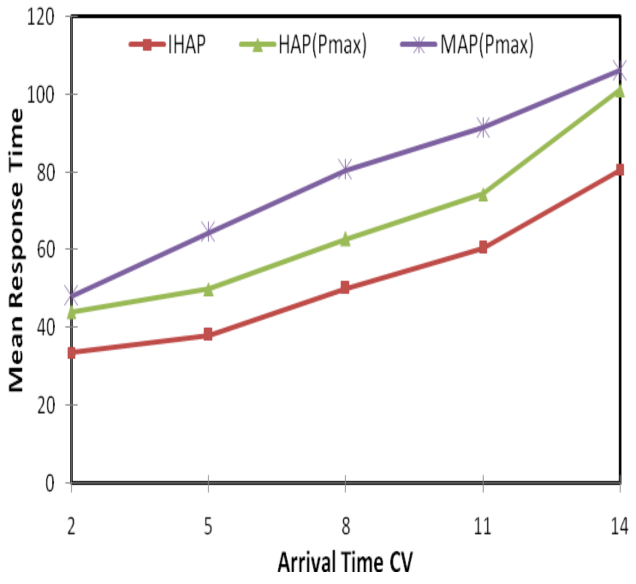

Figure 2: Sensitivity of the policies to arrival time variance

The increase in arrival time variance means the clustered arrival of jobs into the system. This also led to longer gaps in the job arrivals. The impact of variance in arrival time is more on HAP and MAP policies as shown in figure. These two policies suffer from processor fragmentation induced by the background workload and the way the partition-size is computed for the jobs. Since the partition sizes are computed on the basis of total number of BPUs (in case of HAP) and total number of processors (in case of MAP), the actual number of available BPUs (in case of HAP) and available processors (in case of MAP) can be lower than the partition-size computed. This is due to the fact that there is possibility of background tasks running on some of processors at the time and both HAP and MAP doe not consider background workload when computing partition size. But IHAP policy tend to produce smaller partition sizes due to consideration of background workload, therefore the impact of arrival time variance is reduced as compared to other two policies.

6.2.2 Sensitivity to Service Demand Variations

The figure shows that MRT of the three policies increases with the increase in the variance in the service demand. With the increase in service demand variance, there will few large service demand jobs and large number of small service demand jobs. As the service time CV increases, the service demand of the larger jobs will increase even though their number goes down as a fraction of the total jobs.

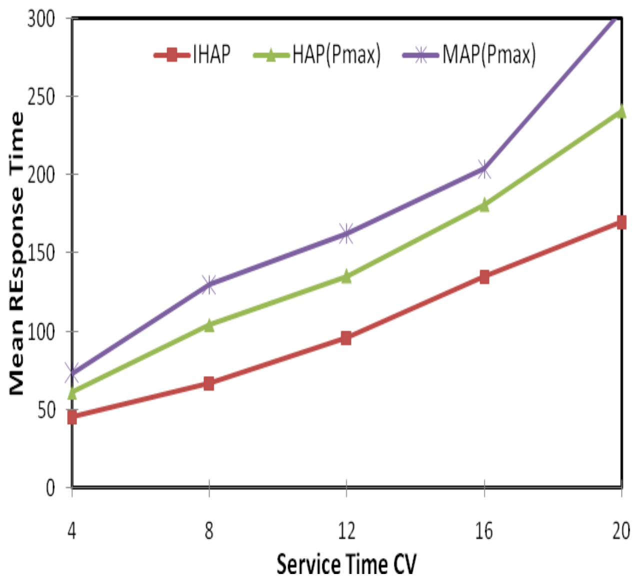

Figure 3: Sensitivity of the policies to service time variance

The impact of service time variance on HAP and MAP policies is more than the impact on IHAP policy. This is due to the fact that both HAP and MAP use FCFS as a job selection policy which is known to be sensitive of variance in service demand, to allocate processors to jobs. FCFS allocation of processors to jobs results in a situation where small jobs could be blocked by an earlier arrived large job. This problem gets more serious as the variance in service demand increases.

7. Conclusion

Space-sharing algorithms are preferred in distributed-memory cluster systems to avoid the overhead due to frequent preemptions involved in time-sharing systems. Adaptive space-sharing algorithms are used in cluster computing systems and dynamic space-sharing algorithms are more suited to shared-memory multiprocessors. Most of popular adaptive algorithms are only designed for dedicated homogeneous as well as dedicated heterogeneous cluster systems. Moreover existing adaptive policies use FCFS as a job-selection policy which is known to be sensitive to service demand variance. Hence these algorithms produce increased mean response times for workloads having high service demand variance. This paper proposes an improved adaptive policy for non-dedicated heterogeneous cluster systems. Comparative results have shown the dominance of the proposed policy over the existing similar policies at medium to high system loads of interest. Also the policy has shown to be relatively less sensitive to service demand variance as compared to existing policies.

8. REFERENCES

[1] J.H. Abawajy. Parallel Job Scheduling Policies on Cluster Computing Systems. Ph.D. Thesis. Ottawa-Carleton Institute for Computer Science, Carleton University, Ottawa, Canada, November, 2003.

[2] S.P. Dandamudi and Z. Zhou, "Performance of Adaptive Space-Sharing Policies in Dedicated Heterogeneous Cluster Systems", Future Generation Computer Systems, 20(5), 895-906 (2004).

[3] D.G. Feitelson, L. Rudolph, U. Schwiegelshohn, K.C. Sevcik, P. Wong, Theory and practice in parallel job scheduling, in: Job Scheduling Strategies for Parallel Processing, Lecture Notes in Computer Science, vol. 1291, Springer-Verlag, Berlin, 1997, pp. 1–34.

[4] D. G. Feitelson and L. Rudolph. Parallel Job Scheduling - A Status Report. Lecture Notes in Computer Science, Springer, Vol. 3277 (2005).

[5] E. Rosti, E. Smirni, L. W. Dowdy, G. Serazzi, and B. M. Carlson. Robust Partitioning Policies for Multiprocessor Systems. Performance Evaluation, Vol.19, 141-265 (1994).

[6] E. Rosti, E. Smirni, L.W. Dowdy, G. Serrazi, K.C. Sevcik, Processor saving scheduling policies for multiprocessor systems, IEEE Transactions on Computers 47 (2) (1998).

[7] S.P. Dandamudi and H. Yu, "Performance of Adaptive Space Sharing Processor Allocation Policies for Distributed-Memory Multicomputers", Journal of Parallel and Distributed Computing, vol. 58, pp. 109-125 (1999).

[8] W. Cirne and F. Berman. Adaptive Selection of Partition Size for Supercomputer Requests. Lecture Notes in Computer Science, Springer, Vol. 1911, 187-208 (2000).

[9] W. Cirne and F. Berman. Using Moldability to Improve the Performance of Supercomputer Jobs. Journal of Parallel and Distributed Computing, Vol. 62, 1571-1601 (2002).

[10] W. Cirne and F. Berman. A Comprehensive Model of the Supercomputer Workload. Proc. of IEEE 4th Annual Workshop on Job Scheduling Strategies for Parallel Processing (2005).

[11] S. Srinivasan, V. Subramani, R. Kettimuthu, P. Holenarsipur, and P. Sadayappan. Effective Selection of Partition Sizes for Moldable Scheduling of Parallel Jobs. Lecture Notes In Computer Science, Springer, Vol. 2552, 174- 183 (2002).

[12] S. Srinivasan, S. Krishnamoorthy, and P. Sadayappan. A Robust Scheduling Strategy for Moldable Scheduling of Parallel Jobs. Proc. of 2003 IEEE International Conference On Cluster Computing (2003).

[13] Young-Chul Shim, "Performance evaluation of scheduling schemes for NOW with heterogeneous computing power", Future Generation Computer Systems. 20(2): 229-236 (2004).

[14] V.H. Doan. An Adaptive Space-Sharing Scheduling Algorithm for PC-Based Clusters, Modeling, Simulation and Optimization of Complex Processes, pp 225-234, 2008.

[15] J.H. Abawajy, "An Efficient Adaptive Policy for High-Performance Computing", Future Generation Computer systems, Vol. 25, 364-370, (2009).