ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 2, February 2015

Process Failure Recovery Mechanism for Computational Grid Environment

S.Manimala[1], P. Gayathri[*2]

Assistant Professor, Dept. of Information Technology, Jerusalem College of Engineering, Chennai, Tamil Nadu, India[1]

Assistant Professor, Dept. of Information Technology, Bharath University, Chennai, Tamil Nadu, India[2]

[*] Corresponding Author

ABSTRACT: Most of the existing grid scheduling algorithms allocates jobs to the resources and when they fails it simply reschedules the job to another optimal resource. So in this method, the failed resource is not attempted to recover instead it is left as it was. This leads to the affects the maximum utilization of resources, also it causes excess bandwidth to transfer job from one resource to the other. As the failed resource is not recovered or it is not taken care, in due course the failure rate of the resource is increased and job becomes unfit for job allocation.

In the proposed system, resource allocation is done using a conventional algorithm, but when the job fails due to process (which is currently executing the job) failure it is attempted to recover. By doing so the resource failure due to process failure is recovered. While the resource fails it is detected for process failure. When it detected for process failure it attempted to recover. When we detect the process failure, the process is attempted to recover with another process in the same resource. When the process is unrecoverable we complete the job with another process in the same resource. Thus in our method, when process failure happens the rescheduling is avoided.

KEYWORDS: Ontology, personalization, knowledge base, user profiles, web information gathering.

I. INTRODUCTION

Grid computing combines computers from multiple administrative domains to reach a common goal, to solve a single task, and may then disappear just as quickly.One of the main strategies of grid computing is to use middleware to divide and apportion pieces of a program among several computers, sometimes up to many thousands. Grid computing involves computation in a distributed fashion, which may also involve the aggregation of large-scale cluster computing-based systems.

The size of a grid may vary from small—confined to a network of computer workstations within a corporation, for example—to large, public collaborations across many companies and networks. "The notion of a confined grid may also be known as an intra-nodes cooperation whilst the notion of a larger, wider grid may thus refer to an inter-nodes cooperation".

Grids are a form of distributed computing whereby a “super virtual computer” is composed of many networked loosely coupled computers acting together to perform very large tasks. This technology has been applied to computationally intensive scientific, mathematical, and academic problems through volunteer computing, and it is used in commercial enterprises for such diverse applications as drug discovery, economic forecasting, seismic analysis, and back office data processing in support for e-commerce and Web services.

Coordinating applications on Grids can be a complex task, especially when coordinating the flow of information across distributed computing resources. Grid workflow systems have been developed as a specialized form of a workflow management system designed specifically to compose and execute a series of computational or data manipulation steps, or a workflow, in the Grid context.

At a very high level, Grid Architecture can be best represented in component layers. This layered architecture is by no means a rigid dictation of the components, but it is extensible and follows the open architectural framework.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 2, February 2015

As shown in the Figure 1.1, the Layered Grid Architecture, Grid Architecture follows the hourglass model, where the narrow neck of hourglass defines a small set of core abstractions and protocols such as TCP and HTTP.
While the base of the model conveys the different underlying technologies, the top of model shows high-level behaviors that translate into services and user applications.

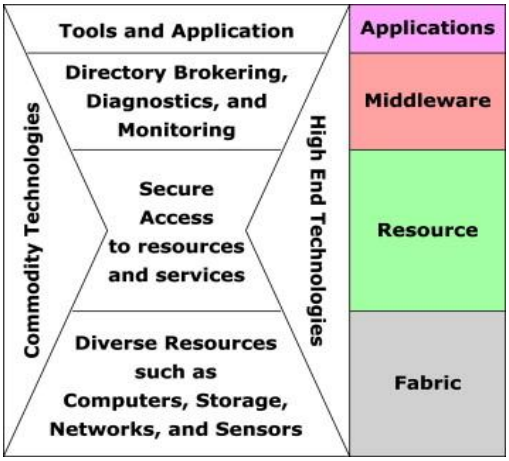


Figure 1.1: Layered Grid Architecture

The fabric layer provides the resources to which the shared access is controlled by the grid protocols. The resources normally include physical and logical entities. Physical entities are resources like storage systems, catalogs, servers, and network resources. The resource may be a logical entity like distributed file system, computer cluster or distributed computer pool, and database systems to store structured data. The Grid mechanism normally permits the capability for the resource management, which involves discovery and control.

The connectivity layer defines core communications and authentication protocols required for Grid specific network transactions. These protocols enable the exchange of data between fabric layer resources. The resource layer, based on the connectivity and authentication protocols, controls the access resources.

The collective services layer deals with the directory brokering services, scheduling services, data replications services, and diagnostics/monitoring services. These services are not associated with any one specific resource but focus on interactions across resources. The programming models and tools define and invoke the collective layer functions. This layer is a key component in the whole grid architecture and its functioning. This is the layer that glues all the resources together in expedient exchange.
The top layer, User Applications, comprises the user applications that operate within a virtual organization (VO) environment.

The components in each layer share common characteristics but can build on new capabilities and behaviors provided by the lower layer. This model demonstrates the flexibility with which Grid Architecture can be extended and evolved. This is the precise reason why grid architecture is taking shape and form based on the guidelines developed by the visionaries and grid standard forums.

With Grid Technology being a new and unexplored arena for many, the tendency is to associate the technology with variations of what has been seen and what is comfortable. One such confusion is the perception of a large cluster to be synonymous with the grid framework. Clusters have been around a good number of years and their architecture and functionality are understood very well. Clusters focus on a single objective and are a collection of servers with homogenous nature.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 2, February 2015

1.1.2TYPES OF GRID

- **1.1.2.1 Computational Grid**

Compute Grids allow you to take a computation, optionally split it into multiple parts, and execute them on different grid nodes in parallel. The obvious benefit here is that your computation will perform faster as it now can use resources from all grid nodes in parallel. However, Compute Grids are useful even if you don't need to split your computation - they help you improve overall scalability and fault-tolerance of your system by offloading your computations onto most available nodes. Some of the "must have" Compute Grid features are:

Automatic Deployment - allows for automatic deployment of classes and resources onto grid without any extra steps from user. This feature alone provides one of the largest productivity boosts in distributed systems. Users usually are able to simply execute a task from one grid node and as task execution penetrates the grid, all classes and resources are also automatically deployed.
Topology Resolution - allows to provision nodes based on any node characteristic or user-specific configuration. For example, we can decide to only include Linux nodes for execution, or to only include a certain group of nodes within certain time window. We should also be able to choose all nodes with CPU loaded, say, under 50% that have more than 2GB of available Heap memory.
Collision Resolution - allows users to control which jobs get executed, which jobs get rejected, how many jobs can be executed in parallel, order of overall execution, etc.
Load Balancing - allows to balance properly balance your system load within grid. Usually range of load balancing policies varies within products. Some of the most common ones are Round Robin, Random, or Adaptive. More advanced vendors also provide Affinity Load Balancing where grid jobs always end up on the same node based on job's affinity key. This policy works well with Data Grids described below.
Fail-over - grid jobs should automatically fail-over onto other nodes in case of node crash or some other job failure.
Checkpoints - long running jobs should be able to periodically store their intermediate state. This is useful for fail-overs, when a failed job should be able to pick up its execution from the latest checkpoint, rather than start from scratch.
Grid Events - a querying mechanism for all grid events is essential. Any grid node should be able to query all events that happened on remote grid nodes during grid task execution.
Node Metrics - a good compute grid solution should be able to provide dynamic grid metrics for all grid nodes. Metrics should include vital node statistics, from CPU load to average job execution time. This is especially useful for load balancing, when the system or user need to pick the least loaded node for execution.
Pluggability - in order to blend into any environment a good compute grid should have well thought out pluggability points. For example, if running on top of JBoss, a compute grid should totally reuse JBoss communication and discovery protocols.
Data Grid Integration - it is important that Compute Grid are able to natively integrate with Data Grids as quite often businesses will need both, computational and data features working within same application.

- **1.1.3.2 Data Grid**

Data Grids allow you to distribute your data across the grid. Most of us are used to the term Distributed Cache rather than Data Grid (data grid does sound more savvy though). The main goal of Data Grid is to provide as much data as possible from memory on every grid node and to ensure data coherency. Some of the important Data Grid features include:Data Replication, Data Invalidation, Distributed Transactions, Data Backups, and Data Affinity/Partitioning.

1.1.3GRID SCHEDULING

Grid scheduling involves three main phases:

- Resource discovery, which generates a list of potential resources
- Information gathering about those resources and selection of a best set
- Job execution, which includes file staging and clean-up

These phases, and the steps that make them up, are shown in Figure 1.2.
Phase 1: Resource Discovery
The first stage in any scheduling interaction involves determining which resources are available to a given user. The resource discovery phase involves selecting a set of resources to be investigated in more detail in Phase 2, information gathering. i At the beginning of Phase 1, the potential set of resources is the empty set; at the end of this phase, the

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 2, February 2015

potential of resources is some set that has passed a minimal feasibility requirements. The resource discovery phase is done in three steps: authorization filtering, job requirement definition, and filtering to meet the minimal job requirements.

Step 1: Authorization Filtering
The first step of resource discovery for Grid scheduling is to determine the set of resources that the user submitting the job has access to. In this regard, computing over the Grid is no different from remotely submitting a job to a single site: without authorization to run on a resource, the job will not run. At the end of this step the user will have a list of machines or resources to which he or she has access. The main difference that Grid computing lends to this problem is

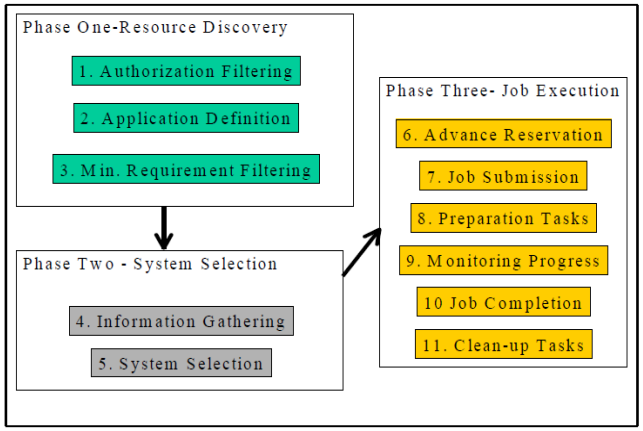


Figure 1.2 Steps in Grid Scheduling

sheer numbers. It is now easier to get access to more resources, although equally difficult to keep track of them. Also, with current GIS implementations, a user can often find out the status of many more machines than where he or she has accounts on. As the number of resources grows, it simply does not make sense to examine those resources that are not authorized for use.

Step 2: Application Requirement Definition
To proceed in resource discovery, the user must be able to specify some minimal set of job requirements in order to further filter the set of feasible resources
Step 3: Minimal Requirement Filtering
Given a set of resources to which a user has access and at least a small set of job requirements, the third step in the resource discovery phase is to filter out the resources that do not meet the minimal job requirements.
Phase 2: System Selection
Given a group of possible resources (or a group of possible resource sets), all of which meet the minimum requirements for the job, a single resource (or single resource set) must be selected on which to schedule the job. This selection is generally done in two steps: gathering detailed information and making a decision. We discuss these two steps separately, but they are inherently intertwined, as the decision process depends on the available information.
Step 4: Dynamic Information Gathering
In order to make the best possible job/resource match, detailed dynamic information about the resources is needed. Since this information may vary with respect to the application being scheduled and the resources being examined, no single solution will work in all, or even most, settings. The dynamic information gathering step has two components: what information is available and how the user can get access to it.
Step 5: System Selection
With the detailed information gathered in Step 4, the next step is to decide which resource (or set of resources) to use.
Phase 3: Job Execution
The third phase of Grid scheduling is running a job. This involves a number of steps, few of which have been defined in a uniform way between resources.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 2, February 2015

Step 6: Advance Reservation (Optional)
In order to make the best use of a given system, part or all of the resources may have to be reserved in advance. Depending on the resource, an advance reservation can be easy or hard to do and may be done with mechanical means or human means. Moreover, the reservations may or may not expire with or without cost. One issue in having advance reservations become more common is the need for the lower-level resource to support the fundamental services on the native resources. Currently, such support is not implemented for many resources, although as service level agreements become more common this is likely to change.

Step 7: Job Submission
Once resources are chosen, the application can be submitted to the resources. Job submission may be as easy as running a single command or as complicated as running a series of scripts and may or may not include setup or staging (see Step 8).In a Grid system, the simple act of submitting a job can be made very complicated by the lack of any standards for job submission

Step 8: Preparation Tasks
The preparation stage may involve setup, staging, claiming a reservation, or other actions needed to prepare the resource to run the application.

Step 9: Monitoring Progress
Depending on the application and its running time, users may monitor the progress of their application and possibly change their mind about where or how it is executing. Historically, such monitoring is typically done by repetitively querying the resource for status information, but this is changing over time to allow easier access to the data. If a job is not making sufficient progress, it may be rescheduled (i.e., returning to Step 4). Such rescheduling is significantly harder on a Grid system than on a single parallel machine because of the lack of control involved - other jobs may be scheduled and the one of concern pre-empted, possibly without warning or notification. In general, a Grid scheduler may not be able to address this situation. It may be possible to develop additional primitives for interactions between local systems and Grid schedulers to make this behavior more straight-forward.[1]

Step 10: Job Completion
When the job is finished, the user needs to be notified. Often, submission scripts for parallel machines will include an e-mail notification parameter. For fault-tolerant reasons, however, such notification can prove surprisingly difficult. Moreover, with so many interacting systems one can easily envision situations in which a completion state cannot be reached.[2]

Step 11: Cleanup Tasks
After a job is run, the user may need to retrieve files from that resource in order to do data analysis on the results, remove temporary settings, and so forth. Any of the current systems that do staging (Step 8) also handle cleanup. Users generally do this by hand after a job is run, or by including clean-up information in their job submission scripts.[3]

II. SYSTEM DESIGN

The implementation of the proposed strategy is done in Alea3 simulation tool kit. Depending on the implementation of this tool kit, the proposed system is designed.

A. ARCHITECTURAL DIAGRAM

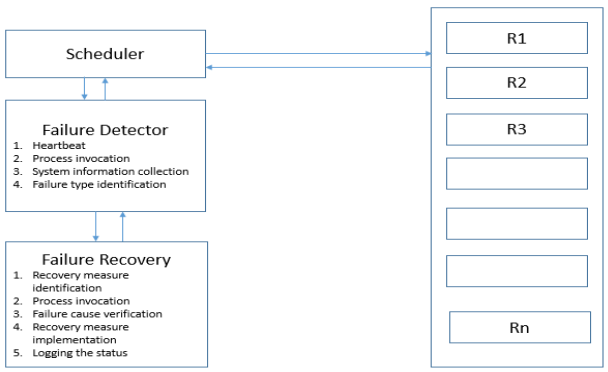


Figure 1.2Architecture Diagram

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 2, February 2015

ARCHITECTURE DESCRIPTION

The job requirements are received from the user and the resource information is received from the GIS. As per the details available, assign the job to the resource that fits the job requirements by FCFS scheduling. When the completed job is received from the scheduler then return the completed job to the user. If the job fails, sent it to the Failure Recovery Engine which has encloses Failure Detection and Failure Recovery.[4]

The Initialization of the failure detection is done by getting failed resource information from the scheduler. The immediate next step is to ensure connectivity by sending ping message to the resource. If the ping message succeed then it proves there is no connectivity problem with the resource. The next step is to send message to the process (say p) and wait for the acknowledgement from that process. If acknowledgement is received from the job, then it implies that the job has failed unexpectedly and the job is resumed again in the same process. If acknowledgement is not received initiate another process (say q) and get the all connection information of the resource.[5]

The connection information is received by the scheduler, as it ensures the scheduler that the process q can communicate with scheduler without any barriers. If process q is successfully connected, then an attempt is made to restart the process p. If the process p is successfully restarted and connected with scheduler, then the job is resumed at process p itself. If process p is not connected with scheduler even after the process restart then process q attempts to finish the job done by process p. Thus the rescheduling of resource is avoided by process recovery.[6]

MODULES

- Scheduling
- Failure Detection
- Failure Recovery

SCHEDULING

When the job is submitted, the scheduler will look for a better fit of the resource by checking the GIS. After the better fit of resource is selected the job is assigned to the resource. After the job is completed the result is collected from the resource and is then given back to the user. When the resource fails to complete the job then the scheduler points to the failure detector.[7]

FAILURE DETECTION

When the job fails the failure detector is invoked. The detector will look for the heartbeat message from the failed resource. If the ping message succeed then it proves there is no connectivity problem with the resource.[8] The next step is to send message to the process (say p) and wait for the acknowledgement from that process. If acknowledgement is received from the job, then it implies that the job has failed unexpectedly and the job is resumed again by the same process. If acknowledgement is not received another process (say q) is initiated and all the connection information of the resource are collected.[9]

FAILURE RECOVERY

The connection information is received by the scheduler, as it ensures the scheduler that the process q can communicate with scheduler without any barriers. If process q is successfully connected, then an attempt is made to restart the process p. If the process p is successfully restarted and connected with scheduler, then the job is resumed at process p itself. If process p is not connected with scheduler even after the process restart then process q attempts to finish the job done by process p.[10]

B. ALGORITHM

/* The algorithm run by a process *p* ***/**
init(*I***) :**
$N = \{p\}$
call add node(q)

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 2, February 2015

procedure add node(*q*) :
/* *called when the process learns a new process q* */
send*monitor req(p)* to *q*
procedure detect failure(*f, q*) :
/* *called when the process detects or learns a process f's death* */
if*f =2 F* :
call handle failure(*f*)
call flood(*failure*(*p; f*), *q*)
procedure handle failure(*f*) :
pick*q* randomly from (*N, H, A*)
send*monitor req(p)* to *q*
procedure flood(*m, q*) :
Recv*monitor req(q)* :
send*node info(p, N)* to *q*
send*monitor ack(p)* to *q*
call add node(*q*)
call flood(*node info*(*p, fqg*), *q*)
Recv*monitor ack(q)* :
pick*q0* randomly from *H*
send*monitor cancel(p)* to *q0*
Recv*monitor cancel(q)* :
Recv*nodes info(q, N0)* :
call flood(*nodes info*(*p;N*), *q*)
call add node(*q*)
send*heartbeat(p)* to *q*
Recv*failure(q; f)* :
call detect failure(*f; q*)
*Tto***has passed since** *p* **sent *monitor req(p)* to** *q* **:**
call detect failure(*q*)
*Tto***has passed since** *p* **received the last *heartbeat(q)* (***q 2 M***)**
call detect failure(*q*)
procedure recover(*p, q*) :
q attempts to repair p
if fails, conncet(q)
procedure connect(q)
p is replaced by q

Before you begin to format your paper, first write and save the content as a separate text file. Keep your text and graphic files separate until after the text has been formatted and styled. [11] Do not use hard tabs, and limit use of hard returns to only one return at the end of a paragraph. Do not add any kind of pagination anywhere in the paper. Do not number text heads-the template will do that for you.[12]
Finally, complete content and organizational editing before formatting. Please take note of the following items when proofreading spelling and grammar:

C. DISCUSSION AND ANALYSIS

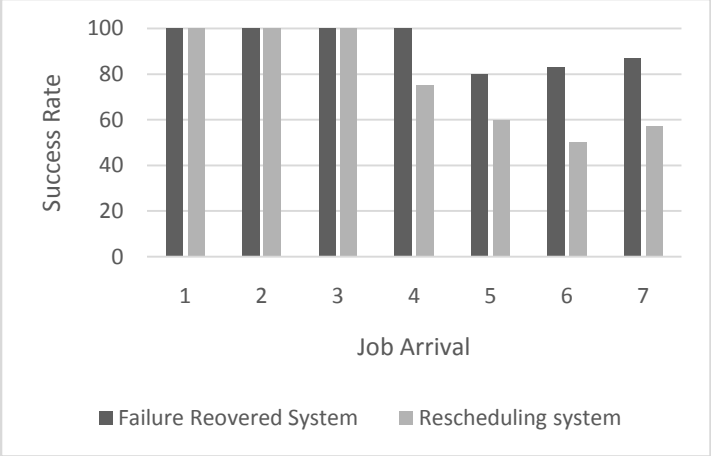
In our proposed system, rescheduling of the failed resource is done when the resource is failed due to memory crash, resource lock andlow bandwidth. When the job is failed due to memory crash then the fresh memory is allocated with memory wiped and the job is resumed. [13]When the job fails due to resource lock, then the process (p or q) is given elevated privileges and all the process using the required resource is terminated and the job is resumed. When the bandwidth is very low then the result is collected in a linear fashion and in timely manner.[14]

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 2, February 2015



Success rate of average resource under Conventional System (Rescheduling system) and proposed system (Failure Recovered System)

In the above chart is very clear that, at the arrival of the job 4, 5 and 6 the failure happens, the proposed system recovers the jobs 4 and 6. Hence the success rate of the resource goes up.[15] Though job 5 is unrecoverable our proposed system has maximum success rate than the conventional system

V. CONCLUSION AND FUTURE WORK

The Scheduling of jobs to the grid resource, failure detection and the recovery of the failed jobs has been implemented. To conclude that the proposed system is the best one with optimal resource utilisation it has to be compared with the commonly used existing systems. As shown in the figure 7.1 our proposed system stands efficient when compared to conventional systems.However the proposed system provides solutions for only major causes of failure and for the remaining undiscovered problems rescheduling is done.

VI. FUTURE *WORK*

The future work of the project includes finding all the possible reasons that could make the job fail and make suitable solution for its recovery. By making so the system will be very much reliable with maximum efficiency.

REFERENCES

[1] G. Eason, B. Noble, and I. N. Sneddon, “On certain integrals of Lipschitz-Hankel type involving products of Bessel functions,” Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955.
[2] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.
[3] Krishnamoorthy P., Jayalakshmi T., "Preparation, characterization and synthesis of silver nanoparticles by using phyllanthusniruri for the antimicrobial activity and cytotoxic effects", Journal of Chemical and Pharmaceutical Research, ISSN : 0975 – 7384, 4(11) (2012) pp.4783-4794.
[4] I. S. Jacobs and C. P. Bean, “Fine particles, thin films and exchange anisotropy,” in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.
[5] K. Elissa, “Title of paper if known,” unpublished.
[6] Madhubala V., Subhashree A.R., Shanthi B., "Serum carbohydrate deficient transferrin as a sensitive marker in diagnosing alcohol abuse: A case - Control study", Journal of Clinical and Diagnostic Research, ISSN : 0973 - 709X, 7(2) (2013) pp.197-200.
[7] R. Nicole, “Title of paper with only first word capitalized,” J. Name Stand. Abbrev., in press.
[8] Khanaa V., Thooyamani K.P., Saravanan T., "Simulation of an all optical full adder using optical switch", Indian Journal of Science and Technology, ISSN : 0974-6846, 6(S6)(2013) pp.4733-4736.
[9] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, “Electron spectroscopy studies on magneto-optical media and plastic substrate interface,” IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].
[10] Nagarajan C., Madheswaran M., "Stability analysis of series parallel resonant converter with fuzzy logic controller using state space techniques", Electric Power Components and Systems, ISSN : 1532-5008, 39(8) (2011) pp.780-793.
[11] M. Young, The Technical Writer’s Handbook. Mill Valley, CA: University Science, 1989.
[12] Electronic Publication: Digital Object Identifiers (DOIs):
[13] D. Kornack and P. Rakic, “Cell Proliferation without Neurogenesis in Adult Primate Neocortex,” Science, vol. 294, Dec. 2001, pp. 2127-2130, doi:10.1126/science.1065467. **(Article in a journal)**

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 2, February 2015

[14] Bhat V., "A close-up on obturators using magnets: Part I - Magnets in dentistry", Journal of Indian Prosthodontist Society, ISSN : 0972-4052 , 5(3) (2005) pp.114-118.
[15] H. Goto, Y. Hasegawa, and M. Tanaka, "Efficient Scheduling Focusing on the Duality of MPL Representatives," Proc. IEEE Symp. Computational Intelligence in Scheduling (SCIS 07), IEEE Press, Dec. 2007, pp. 57-64, doi:10.1109/SCIS.2007.357670. **(Article in a conference proceedings)**
[16] M.Sundararajan & R.Pugazhanthi," Human finger print recognition based biometric security using wavelet analysis", Publication of International Journal of Artificial Intelligent and Computational Research, Vol.2. No.2. pp.97-100(July-Dec 2010).
[17] .M.Sundararajan & E.Kanniga," Modeling and Characterization of DCO using Pass Transistor", proceeding of Springer – Lecturer Notes in Electrical Engineering-2011 Vol. 86, pp. 451-457(2011). ISSN 1876-1100.(Ref. Jor- Anne-II)
[18] M.Sundararajan & C.Lakshmi, "Wavelet based finger print identification for effective biometric security", Publication of Elixir Advanced Engineering Informatics-35(2011)-pp.2830-2832.
[19] M.Sundararajan, "Optical Instrument for correlative analysis of human ECG and Breathing Signal" Publications of International Journal of Biomedical Engineering and Technology- Vol. 6, No.4, pp. 350-362 (2011). ISSN 1752-6418.(Ref. Jor-Anne-I
[20] .M.Sundararajan, C.Lakshmi & D.Malathi, "Performance Analysis Restoration filter for satellite Images" Publications of Research Journal of Computer Systems and Engineering-Vol.2, Issue-04- July-December-2011-pp 277-287