A SURVEY OF DESKTOP GRID SYSTEM SCHEDULING

G Mahesh Kumar
Department of Computer Science,
Bhavan's Vivekananda College,
Sainikpuri, Secunderabad, Andhra Pradesh, INDIA
gmaheshkumar.cs@bhavansvc.org

Abstract—Grid Computing forms virtual, collaborative organizations that share applications and data in an open heterogeneous server environment in order to work on common problems. Desktop Grid is a named collection of machines in a shared network where resource providers have heterogeneous properties such as CPU, network, memory complicated by various capabilities, failures, lack of trust based on desktop computers on the internet. In Desktop Grid Computing environment resource providers can participate in computations for free and can leave them in middle of the execution where some malicious resource providers may corrupt the computational results. Therefore, it is necessary to develop scheduling mechanisms that adapts this dynamic computational environment. This paper presents a survey of scheduling for desktop grid systems.

Keywords- Grid Computing, Desktop Grid, Volunteers, Scheduling, Grid Scheduler

I. INTRODUCTION

Grid Computing enables virtual organizations to share geographically distributed resources as they pursue common goals, assuming the absence of central location, central control, omniscience, and an existing trust relationship [1]. A huge computational problem cannot be executed in one system due to lack of required resources, so the problem is divided into some sub tasks and make it execute on remote machines by borrowing the resources from other resource providers(volunteers). In this the volunteers and users both need to be reliable to each other in the communication, so that each entity can trust each other for sharing resources. In this the resource allocation and resource management need to be done only for the authenticated users. Desktop systems are underutilized in most of the organizations where 95% of the time is idle and only less than 5% of the time is busy for computations [2].

Desktop Grid computing allows the resource providers to join the network for free and has the flexibility to move out of network any time that might be before the completion of the task also. This raises the problem where many tasks may be incomplete or the same problem may be redundantly given to many volunteers to execute and whichever completes the task that will be accepted and others will be discarded. In order to maintain all the volunteers properly a grid scheduler is required for scheduling all the tasks to volunteers properly.

II. DESKTOP GRIDS

The common architecture of desktop grids consists of one or more central servers and a large number of clients. The central server provides the applications and their input data. Clients join the desktop grid voluntarily, offering to download and run an application with a set of input data. When the application has finished, the client uploads the results to the server. Based on the environment where the desktop grid is deployed we must distinguish between two different concepts.

Global Desktop Grids also known as Public Desktop Grids consist of a publicly accessible server providing projects and the attached clients. There are several unique aspects of this computing model compared to traditional grid systems. First, clients may come and go at any time, and there is no guarantee that a client that started a computation will indeed finish it. Furthermore, the clients cannot be trusted to be free of either hardware or software defects, meaning the server can never be sure that an uploaded result is in fact correct. Therefore, redundancy is often used by giving the same piece of work to multiple clients and comparing the results to filter out corrupt ones.

Local Desktop Grids are intended for institutional or industrial use. This environment gives more flexibility by allowing the clients to access local resources securely and since the resources are not voluntarily offered the performance is more applicable. The SZTAKI local desktop grid [3] is based on BOINC technology and extends to the needs of institutional and business users. BOINC [4] is originating from the SETI@Home [4] project to provide an open infrastructure for utilizing the computers of people interested in the outcome of a project. A grid scheduler takes the responsibility of assigning the tasks to the volunteers based on their availability.

III. SURVEY OF DESKTOP GRID SYSTEMS

BOINC (Berkeley Open Infrastructure for Network Computing) [4]-[7], is a well-known middleware system for volunteer computing. There are a lot of BOINC based projects: SETI@Home, Predictor@Home, Folding@Home, LHC@Home Climatepredication.net, and so on. BOINC consists of a server and volunteer (client). In this system a server has a task server that dispatches tasks and processes the results of tasks, a data server that handles file transfer, a database that stores description of applications, volunteers, scheduling, etc. A volunteer participates in several projects and specify the preferences for the projects execution. This is mainly based on voluntary participants connected through internet. BOINC server is responsible for scheduling where a volunteer request is accepted by the task server which allocates a list of new tasks to the volunteer. This middleware system also supports locality scheduling where a volunteer performs a local scheduling on its computer to decide which task to run among multiple projects, when to ask a server for more works, which project to ask and how much work to ask for. Volunteers can join and leave freely, so scheduling should be dynamic in this system. BOINC is used for applications in mathematics, climate, medicine, chemistry, physics, astronomy, molecular biology, etc. These applications are mainly compute-intensive and independent. BOINC scheduler assigns tasks to those volunteers who can complete the task within time period. If the task fails then it create a new instance of task and redistributes it. This system supports redundant computing to identify and reject erroneous results.

Entropia [8], [9], is a middleware system for commercial desktop grid. This system has two solutions: Internet Desktop Grid and Enterprise Desktop Grid. This system consists of server and client (volunteer). The server in Internet Desktop Grid consists of a task server, file server and application server. A task server is responsible for registration, scheduling and resource management which maintains a database and forms an application pool containing a list of clients, the number of assigned jobs, and the number of completed jobs. The clients within a pool have similar capabilities such as disk space. The application server decomposes a task into subtasks and assigns them to clients. The task server performs scheduling in a pull mode. Enterprise Desktop Grid has three layers: job management layer, subjob management layer, and resource management layer. The resource management layer is responsible for registration and resource management, and maintains resource pools. The subjob management layer is responsible for scheduling. The job management layer is responsible for job decomposition and management. Clients periodically report their resource status to node manager in the resource management layer and the subjob scheduler. The scheduler assigns subjobs to available clients according to client's attributes such as memory capacity. If a client fails, the scheduler redistributes the subjob to another client.

CPM (Compute Power Market) [10], [11], is a market-based middleware system for grid computing on low-end personal computing devices connected to internet. It aims to develop a computational marketplace for the regulation of resource demand and supply. CPM consists of a market, a resource consumer (client) and a resource provider (volunteer). A market maintains information about consumers and providers. A resource consumer downloads a market resource broker from a market to buy computing power. The market resource broker finds appropriate providers depending on the information provided by the market. It selects resources based on deadline or budget, negotiates the cost of resources, and distributes the task to them. A resource provider has a market resource agent helps in selling computing power through the market. The market resource agent updates information about its resource provider, deploys and executes the tasks. Resource Consumer is responsible for scheduling the tasks by negotiating the resources cost.

Charlotte [12] is a Java-based middleware system for meta-computing on the web. This system consists of manager (server) and a worker (volunteer). A manger provides a scheduling service and a memory service for accessing shared data whereas a worker provides a computing service implemented as an applet. This system follows eager scheduling, i.e. first, participating workers pickup the task from manager and execute each task and if a worker finishes its task then the worker contacts the eager scheduler for another task. The eager scheduler assigns the unassigned tasks if it is available or it assigns the uncompleted tasks to various workers which leads to redundancy of task execution but we gain benefit by eager or fast workers to complete tasks overtake slow or failed workers.

Javelin [13]-[15], is a Java-based infrastructure for parallel internet computing. In this system applications run as java applets. Applications are mainly compute-intensive and independent. This system consists of three entities: broker, client and host (volunteer). A client registers its tasks with a broker. A host offers computing resources and a broker coordinates the supply and demand for computing resources. When a host communicates with a broker, the broker adds the host to a logical tree structure. If a host requests tasks to the broker, the broker informs the host with the client ID and application information. Then the host executes tasks. In this system work stealing and eager scheduling is performed. With work stealing, when a host runs out of work, it request tasks from other hosts in deterministic or probabilistic approach. In a deterministic approach, a host asks tasks from their children or parents on the basis of tree structure. In a probabilistic approach, the host selects the targets randomly from the list of hosts it currently holds. Eager scheduling allows the client to select the

unassigned task and reissues it to another host. In this system work stealing is performed at a host, and eager scheduling is done at a client. A broker is a simple mediator between clients and hosts.

XtremWeb [16]-[18], is a Java-based middleware system for global computing. This system consists of client, server (coordinator), and worker (volunteer). A client is responsible for tasks submission and results retrieval. A coordinator is composed of a repository that advertises or publishes applications, a scheduler, a result server that collects results and database. All the communications are initiated by a worker. A worker contacts its server to get tasks. In this system the scheduler uses pull mode to allocate tasks. FIFO (Fist in First Out) mechanism is applied by the scheduler to distribute the tasks to the workers. The applications are mainly compute-intensive and independent. A worker periodically sends alive signal to the server and if the server has not received the message during a predefined time, it reschedules the task to another worker.

Webcom [19], [20], is a web-based distributed computation platform using Java. This system consists of a master (server) and a client (volunteer). A master maintains a set of clients and is responsible for scheduling. A client receives a task as the form of a Java applet and executes it within its browser. A master generates atomic task or a condensed graph (set of tasks) that represents an acyclic graph of interacting sequential programs. Execution begins at the master, and the tasks are distributed to clients when clients are available. A client can act as a potential master and it can be promoted to be a master if it receives a condensed graph. The promoted master is assigned a number of clients according to communication latency and if it needs more clients, it requests them to the primary master. In this way, the primary master, promoted master, and clients form a tree structure. Scheduling is performed hierarchically at the primary master and the promoted masters. Scheduling is done according to the network latency between client and master where it allocates tasks to clients on the basis of an expected execution and CPU performance. If a task fails, it is rescheduled to another client.

Bayanihan [21], [22], is a web-based volunteer computing system using Java. Bayanihan system consists of client (volunteer) and server. A client can either be a Java applet started from a web browser or Java application. A client has a worker engine for performing computation or a watcher engine for viewing results and statistics. A server consists of HTTP server, work manager, watch manager, and data pool. The HTTP server serves out Java class files. The work manager distributes tasks and collects tasks. The watch manger distributes results to watcher engines in clients. The work manager in a server is responsible for scheduling. A client makes a remote call to the server to get a task. This system uses eager scheduling, in which volunteer asks its server for a new task as it completes its current task. This system also provides credibility-based fault tolerance mechanism to tolerate erroneous results from malicious volunteers. The credibility-enhanced eager scheduling estimates the probability of result and worker being correct by using voting, spot-checking. In a majority voting approach, the same task is performed at different volunteers as much as the number of redundancy. Redundancy is used to identify the correct result against erroneous one if there are sufficiently more good volunteers than bad one. In a spot-checking approach, the special task whose result is already known is performed at randomly selected volunteers. The volunteers that produce erroneous results will be blacklisted. If a worker executes a task slowly or if it fails, the scheduler reassigns the task to different workers. These applications are mainly compute-intensive and independent.

POPCORN [23] is a Java-based infrastructure for globally distributed computation over the Internet. This system provides a market-based mechanism for trade of computational resources. This system consists of market, seller (volunteer), and buyer (client). A market matches buyers and sellers according to economic model. A seller provides its resource to a buyer by using Java-enabled browser. These applications are mainly compute-intensive and independent. A market uses the popcoin and maintains a database about it. A seller can earn popcorn or get any other type of reward such as on-line game. A buyer can buy resources with popcoins. The market is responsible for matching buyers and sellers, for transferring tasks and results between them, and for handling all payments and accounts. This system deals with failure and verification. If a task fails, simply resend the tasks to a different host.

Organic Grid [24], [25], provides a self-organizing and distributed approach to the organization of the computation. Hosts (volunteer) keep a list of other hosts called friends list, and build tree-based overlay network. A user starts the computation on its host. If a host receives a request from other hosts, it distributes tasks to them as the form of a mobile agent and the requesting host becomes the child of the original host. An agent requests its parent for more work when it completes its task. If the parent does not have tasks, it sends a request to its parent. If a host obtains results from children or finishes its tasks, it sends them to its parent. The tree overlay network is restructured during the computation according to the performance, that is, the rate at which a host sends a result. The hosts with high performance move towards the root of the tree. The reconstruction minimizes communication delay between the root and the best host, and makes it possible to firstly allocate tasks to the best hosts. If the parent of a host fails, the node contacts its parent's ancestor, where now the ancestor becomes the parents of the host and the computation resumes.

Messor [26], [27], aims to support the concurrent execution of highly parallel and compute-intensive computations. Messor is composed of interconnected nests. A nest is a peer entity sharing its computational and

storage resources. It can generate autonomous agents that travel across the nest network. Every nest can submit tasks to the nest network and the submitted tasks are scheduled to other nests in a distributed way. Messor constructs workload-based random graph on the fly and achieve load balancing. Paradropper [28], [29], is a global computing system that supports self-organizing overlay network by using peer-to-peer technologies. Peers are organized into an overlay network to form a small world graph. A new volunteer send a message to the entry point selected randomly among the list of peers. The entry point in turn introduces the new peer to its neighbors which forms a small world graph. Every peer maintains a workload, whenever a peer accepts a task its workload increases by 1. Whenever a peer finishes a task and returns a result its workload decreases by 1. A new peer has the workload 0. When the workload gets changed, a peer sends Load Change Report messages to its neighbors. In this system tasks are distributed according to workload in a distributed way where each peer selects the target that has the smallest workload in its neighbors. When a peer gets overloaded, it can redistribute its tasks to the network. This system accepts dynamic scheduling.

CCOF (Cluster Computing on the Fly) [30]-[33], is a cycle sharing peer-to-peer system. It harvests idle cycles from users at the edges of the internet. This system supports a distributed model, in which there is no server and any peer can be either a producer or consumer or both. Hosts (volunteers) join a community based overlay networks to donate their idle cycles. Clients discover these resources from those overlay networks and schedule tasks according to timezone. If a scheduler fails to find enough resources at day timezone, it reschedules the tasks at night timezone. If a host is not able to complete its tasks, the task migrates to a new host at night timezone for faster execution. If a host fails to find a new host, the original client reschedules the job. This system proposes a trust-based scheduling which uses the reputation based system to select trusted hosts.

Condor [34]-[36], is a batching system for high-throughput computing on large collection of distributed resources. This system provides a job management, scheduling, and resource monitoring and resource management. Condor is comprised a central manager and other resources. A central manager is responsible for scheduling and information management about jobs and resources. This system can be used to manage dedicated cluster, or to harness wasted CPU power from otherwise idle desktop workstations within the boundaries of an organization. This system can be configured to run jobs only when the keyboard and CPU are idle. This system provides ClassAd in order to describe characteristics and requirements of both jobs and resources. It also provides a matchmaker for matching a job with an available resource. Condor performs scheduling as follows: Agent and resources advertise their ClassAds to its matchmaker. The matchmaker investigates the ClassAds, and selects job and resource pairs that satisfy each other's constraints and preferences and finally they establish a contract, and then cooperate to execute. ClassAd describes the special attributes called Requirements and Rank. Requirements attribute indicates the desirability of a match. The matchmaker first selects both job and resource and satisfy Requirements value, then it chooses the one with the highest Rank value among compatible matches.

Kondo et al. [37]-[41], proposes centralized scheduling mechanism in Desktop Grid. They proposed resource selection approaches for short-lived application: resource prioritization, resource exclusion, and task replication [39]. Resource prioritization is to sort hosts according to some criteria such as clock rate. Resource exclusion is to exclude some hosts according to clock. Task replication is to replicate tasks to multiple hosts in order to reduce the probability of tasks failure and completion delay or to schedule tasks to faster hosts. They proposed a timeout mechanism, that is, if a server does not receive result from a host within timeout, it redistributes the task to another host. They proposed a scheduling mechanism by using a buffer in order to complete application within deadline [41].

IV. CONCLUSION

This paper presents various desktop grid systems focusing the mechanism of dynamic scheduling and trust-based scheduling in which the volunteers can join and leave the network for free due to its private execution, then the schedulers are assigning the same task to multiple users to complete the task and also the volunteers and clients establishes trust for further communication.

REFERENCES

[1] Ahmar Abbas, "Grid Computing: A Practical Guide to Technology and Applications", Firewall media, 2004.
[2] P.Suresh Kumar, P.Sateesh Kumar, S.Ramachandram, "*Recent Trust Models in Grid*," Journal of Theoretical and Applied Information Technology, vol.26 No.1, pp.64-68, April 15, 2011.
[3] SETI@home: Search for Extraterrestrial Intelligence at Home. [Online]. Available: http://setiathome.berkeley.edu
[4] BOINC (Berkeley Open Infrastructure for Network Computing), [Online]. Available: http://boinc.berkeley.edu/
[5] D. P. Anderson, "BOINC: A System for Public-Resource Computing and Storage," *The Fifth IEEE/ACM International Workshop on Grid Computing (GRID'04)*, IEEE CS Press, pp. 4-10, Nov. 2004.
[6] M. Taufer, D. Anderson, P. Cicotti, and C. L. Brooks III, "Homogeneous Redundancy: a Technique to Ensure Integrity of Molecular Simulation Results Using Public Computing," *The 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05), Heterogeneous Computing Workshop (HCW'05)*, pp. 119a, Apr. 2005.
[7] D. P. Anderson, E. Korpela, and R. Walton, "High-Performance Task Distribution for Volunteer Computing," *The First IEEE International Conference on e-Science and Grid Technologies (e-Science2005)*, pp. 196-203, Dec. 2005.

[8] A. Chien, B. Calder, S. Elbert, and K. Bhatia, "Entropia: architecture and performance of an enterprise desktop grid system," *Journal of Parallel and Distributed Computing*, vol. 63, issue 5, pp. 597-610, May 2003.

[9] A. A. Chien, S. Marlin, and S. T. Elbert, "Resource management in the Entropia System," *Chapter 26 in Grid Resource Management: Sate of the Art and Future Trends*, Kluwer Academic, 2003.

[10] R. Buyya and S. Vazhkudai, "Compute Power Market: towards a market-oriented grid," *The First IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid'01)*, pp. 574-581, May 2001.

[11] T. T. Ping, G. C. Sodhy, C. H. Yong, F. Haron, and R. Buyya, "A Market-based Scheduler for JXTA-based Peer-to-Peer Computing System," *International Conference on Computational Science and its Applications (ICCSA 2004)*, LNCS 3046, pp.147-157, May 2004.

[12] A. Baratloo, M. Karaul, Z. M. Kedem, and P. Wijcko®, "Charlotte:Metacomputing on the Web," *Future Generation Computer Systems, Special Issue on Metacomputing*, vol. 15, issues 5-6, pp. 559-570, Oct.1999.

[13] M. O. Neary, B. O. Christiansen, P. Cappello, and K. Schauser, "Javelin: Parallel computing on the internet," *Future Generation Computer Systems, Special Issue on Metacomputing*, vol. 15, issue 5-6, pp. 659-674, Oct. 1999.

[14] M. O. Neary, S. P. Brydon, P. Kmiec, S. Rollins, and P. Cappello,"Javelin++: Scalability Issues in Global Computing," *Concurrency: Parctice and Experience*, vol. 12, issue 8, pp. 727-753, Aug. 2000.

[15] M. O. Neary and P. Cappello, "Advanced eager scheduling for Java-based adaptive parallel computing," *Concurrency and Computation: Practice and Experience*, vol. 17, issue 7-8, pp. 797-819, Jun. 2005.

[16] XtremWeb, [Online]. Available: http://www.lri.fr/ fedak/XtremWeb/

[17] G. Fedak, C. Germain, V. Neri, and F. Cappello, "XtremWeb: A Generic Global Computing System," The 1st IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID2001): Workshop on Global Computing on Personal Devices, pp. 582-587, May 2001.

[18] F. Cappello, S. Djilali, G. Fedak, T. Herault, F. Magniette, V.Neri, and O. Lodygensky, "Computing on large-scale distributed systems: XtremWeb architecture, programming models, security, tests and convergence with grid," *Future Generation Computer Systems*, vol. 21, issue 3, pp. 417-437, Mar. 2005.

[19] J. P. Morrison, J. J. Kennedy, and D. A. Power, "WebCom: A WebBased Volunteer Computer," *Journal of Supercomputing*, vol. 18, no.1, pp. 47-61, Jan. 2001.

[20] J. P. Morrison, J. J. Kennedy, and D. A. Power, "Load balancing and fault tolerance in a condensed graphs based metacomputer," *The Journal of Internet Technologies, Special Issue on Web based Programming*, vol. 3, no. 4, pp. 221-234, Dec. 2002.

[21] L. F. G. Sarmenta and S. Hirano. "Bayanihan: building and studying web-based volunteer computing systems using Java," *Future Gen*eration Computer Systems, Special Issue on Metacomputing, vol. 15,issue 5-6., Oct. 1999.

[22] L. F. G. Sarmenta, "Sabotage-Tolerance Mechanisms for Volunteer Computing Systems," *Future Generation Computer Systems*, vol. 18, issue 4, pp. 561-572, Mar. 2002.

[23] N. Nisan, S. London, O. Regev, and N. Camiel, "Globally distributed computation over the Internet-the POPCORN project," *The 18th International Conference on Distributed Computing Systems*, pp.592-601, May 1998.

[24] A.J. Chakravarti, G. Baumgartner, and M. Lauria, "The Organic Grid: Self-Organizing Computation on a Peer-to-Peer Network,"*IEEE Transactions on Systems, Man, and Cybernetics*, vol. 35, no.3, pp. 1-12, May 2005.

[25] A.J. Chakravarti, G. Baumgartner, and M. Lauria, "The Organic Grid: Self-Organizing Computational Biology on Desktop Grids," *Chapter 27 in Parallel Computing for Bioinformatics and Computational Biology: Models, Enabling Technologies, and Case Studies*, Wiley, 2006.

[26] O. Babaoglu, H. Meling, and A. Montresor, "Anthill: a framework for the development of agent-based peer-to-peer systems," *The 22nd International Conference on Distributed Computing Systems*, pp. 15-22, Jul. 2002.

[27] A. Montresor, H. Meling, and O. Babaoglu, "Messor: Load-Balancing through a Swarm of Autonomous Agents," *International Workshop on Agents and Peer-to-Peer Computing (AP2PC 2002)*, LNCS 2530, pp. 125-137, Jul. 2003.

[28] L. Zhong, D. Wen, Z. W. Ming, and Z. Peng, "Paradropper: a general-purpose global computing environment built on peer-to-peer overlay network," *The 23rd International Conference on Distributed Computing Systems (ICDCS 2003), Workshop on New Advances of Web Server and Proxy Technologies (NAWSPT)*, pp. 954-957, May 2003.

[29] W. Dou, Y. Jia, H. M. Wang, W. Q. Song, and P. Zou, "A P2P ap-proach for global computing," *International Parallel and Distributed Processing Symposium 2003 (IPDPS 2003)*, pp. 6-11, Apr. 2003.

[30] D. Zhou and V. Lo, "Cluster Computing on the Fly: resource discovery in a cycle sharing peer-to-peer system," *IEEE International Symposium on Cluster Computing and the Grid (CCGrid'04)*, pp. 66-73, May 2004.

[31] V. Lo, D. Zhou, D. Zappala, Y. Liu, and S. Zhao, "Cluster Computing on the Fly: P2P Scheduling of Idle Cycles in the Internet," *The3rd International Workshop on Peer-to-Peer Systems (IPTPS'04)*, LNCS 3279, pp.227-236, Feb. 2004.

[32] D. Zhou and V. Lo, "Wave Scheduler: Scheduling for Faster Turnaround Time in Peer-to-peer Desktop Grid Systems," *The11th Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP'05)*, LNCS 3834, pp. 194-218, Jun. 2005.

[33] S. Zhao and V. Lo, "Result Verification and Trust-based Scheduling in Open Peer-to-Peer Cycle Sharing Systems," *The Fifth IEEE International Conference on Peer-to-Peer Computing (P2P 2005)*, pp.31-38, Sept. 2005.

[34] D. Thain, T. Tannenbaum, and M. Livny, "Condor and the Grid," *Chapter 11 in Grid Computing: Making the Global Infrastructure a Reality*, Wiley, 2003.

[35] D. Thain, T. Tannenbaum, and M. Livny, "Distributed Computing in Practice: The Condor Experience," *Concurrency and Computation: Practice and Experience*, vol. 17, issue 2-4, pp. 323-356, Feb.2005.

[36] T. Tannenbaum, D. Wright, K. Miller, and M. Livny, "Condor-A Distributed Job Scheduler," *Chapter 15 in Beowulf Cluster Computing with Linux*, The MIT Press, 2002.

[37] D. Kondo, H. Casanova, E. Wing, and F. Berman, "Models and scheduling mechanisms for global computing applications," *The 16th International Parallel and Distributed Processing Symposium (IPDPS2002)*, pp.79-86, Apr. 2002.

[38] D. Kondo, M. Taufer, J. Karanicolas, C. L. Brooks, H. Casanova, and A. Chien, "Characterizing and Evaluating Desktop Grids: An Empirical Study," *The 18th International Parallel and Distributed Processing Symposium (IPDPS 2004)*, pp. 26-35, Apr. 2004.

[39] D. Kondo, A. A. Chien, and H. Casanova, "Resource Management for Rapid Application Turnaround on Enterprise Desktop Grids," *The ACM/IEEE Conference on Supercomputing (SC2004)*, pp. 17-29, Nov. 2004.

[40] D. Kondo, "Scheduling Task Parallel Applications for Rapid Turnaround on Desktop Grids," *Ph.D. Thesis*, Department of computer Science and Engineering, University of California, San Diego, 2005.

[41] D. Kondo, B. Kindarji, G. Fedak, and F. Cappello, "Towards Soft Real-Time Applications on Enterprise Desktop Grids," *The SixthIEEE International Symposium on Cluster Computing and the Grid (CCGRID 2006)*, pp. 65-72, May 2006.