jUniGrid: A Simplistic Framework for Integration of Mobile Devices in Heterogeneous Grid Computing

Ketan B. Parmar[1],* Nalinbhai N. Jani[1], Pranav S. Shrivastav[2], Mitesh H. Patel[2]

[1]Department of Computer Sicence, Kadi Sarva Vishwavidyalaya, Gandhinagar – 382015, Gujarat, India

[2]Chemistry Department, School of Sciences, Gujarat University, Navrangpura – 380009, Gujarat, India

Abstract– **While inclusion of mobile devices into grid computing provides numerous advantages, the current implementations are either too complex or too specific, hence, cannot often be used efficiently by small to medium scale organizations. Further, diversity of platforms & development environments and virtual non-existence of high-end mobile devices until last few years were some of the hindrances in the path of incorporating mobile devices in real world grids. To address these lacunae, herein we introduce a new simplistic framework "jUniGrid" for integrating mobile devices as workstations in heterogeneous grid computing (desktop volunteer grid computing systems, DGVCS) to be used for performing high complexity computational problems. jUniGrid is a flexible grid architecture designed to deploy java-enabled devices (mobile or otherwise) as grid-nodes.**

Keywords— **Desktop Grid, Volunteer Computing, Grid Computing, Mobile Grid, Android Mobile, Heterogeneous Grid, DGVCS and M/DGVCS**

I. INTRODUCTION

The term "Grid Computing" [9, 12, 13, 39] represents a distributed computing infrastructure enveloping a number of separate/independent computing devices, for accomplishing highly resource intensive computing tasks. The definition of Grids has been redefined along the years. Initially Grids were defined as an infrastructure to provide easy and inexpensive access to high-end computing. Then, it was refined as an infrastructure to share resources for collaborative problem solving. More recently, the Grid definition evolved into an infrastructure to pool and virtualize resources and enable their use in a transparent fashion [19].

To keep up with the exponential demand for computing power & to control costs, grid computing has emerged as an important alternative of supercomputing with several competitive advantages [2, 20, 22, 23]. In fact, volunteer computing (VC) platforms are one of the largest and most powerful distributed computing systems on the planet, running applications from diverse scientific domains (including computational biology, climate prediction, and high-energy physics) [23].

A mobile grid (as opposed to conventional grid) makes use of computing power available in mobile computing devices (specifically smart phones and tablets). A mobile grid infrastructure [4, 30] proposes the integration of mobile devices with grid environments. This proposal has two possibilities to interact with the grid (1) A mobile device may be used as an access interface to a grid (e.g., grid portal) or (2) Mobile devices may be deployed as resources for a grid, providing processing and/or other resources facilities. Although, mobile nodes may not be the best choice to participate in high-performance distributed computation facility for large-scale/grand-challenge applications as of now, they offer very impressive computational performance (say in 3D-rendering or multimedia – think iPhone or android mobile), and the scenario is improving with maturing technology [16]. As these devices are continuously powered and available for potential assignments, they are fair, if not the best, candidates for grid computing applications. Further, as would be discussed later in this article, a "mobile grid" not only utilizes available mobile devices' idle potential – providing additional computing power in various deployment scenarios, but also provides an efficient means for creating on-demand ad-hoc grid computing solutions or yet further, extending existing grid implementations. From a business process viewpoint, this implies that mobile nodes can be involved directly in workflow activities (such as task-submission). Thus, mobile grids have to overcome content delivery and portal-based approaches (common practice in current implementations) for extending grids to mobile devices [34].

II. RELATED WORKS

The field of the grid computing is rich with regard to diverse concepts, architectures and solutions. Several frameworks/APIs/Toolkits/Middleware are available for developing grid software, perhaps most notable of them being the Globus Toolkit [11, 38], Legion [15], Condor [28] and GridGain (www.gridgain.com/). These frameworks are meant to furnish mainly two types of systems, referred as "Service Grids" (e.g., EGEE) and "Desktop Grids" (volunteer computing, e.g. BOINC and XtremWeb). Technological cross-over projects to facilitate interoperability between "service" and "desktop" grids are also being pursued [2, 22]. Although current state of the art has not addressed the problems of mobile grid computing in their entirety, some issues related to mobile grid computing have been addressed (and corresponding architectures proposed) in last decade [8, 27, 29]. These are referred below with brevity.

J. Hwang et al. [17] propose a virtual cluster approach and a middleware to provide peer-to-peer operations. Siegel et al.

[29] and Datta [8] have advanced conceptual models of ad-hoc mobile grid architectures that deploy intelligent agents and relay on the virtual backbone (constructed from powerful mobile nodes and maintained with a proactive protocol) to organize the network scale resources for a specific mission. Similar models have been the theme in many recent papers and research projects lately [4, 5, 10, 14] but none of these provide any implementation methodology or an architecture to support this integration. T. Phan et al. [34] have visited the challenges of integrating mobile devices with computational grid in detail within the scope of their project. They propose, the integration be provided through the use of an Interlocutor (software agent), which acts as proxy for cluster of Minions.

Kurkovsky et al. [24, 25, 26] have proposed an agent based wireless grid architecture to solve computationally expensive tasks. Their architecture enables mobile devices within a wireless cell to form computational grid. P. Mudali et al. [31] have addressed the mobility concerns to certain extent by extending the architecture (by Kurkovsky et al in [25]) to a multi-cell wireless computational grid based on location area concept in GSM cellular networks. The proposed wireless computational gird is capable of greater device mobility tolerance than previous work [25].

Ian F. Akyildiz[1] et al. have proposed a simplified random walk model for hexagonal cell configuration where, probability states gives performance of the model. Further Guoliang Xue et al. [42] and A.B.M. Siddique Hossain [18] have proposed improved models by reducing number of probability states to improve the performance. Chiu-Ching Tuan et al. have proposed two seperate models for PCS networks, namely 'novel normal walk model' [41] (for mesh cell configuration) and 'compact normal walk model' [40] (for hexagonal cell configuration) to describe the daily mobility behavior of an individual mobile device that moves from a cell to another, Reades et al. [35] also looked at how call volume at a cell tower correlates with urban activities in the geographic vicinity of the tower in PCS networks. Jingyuan Zhang [43] has described various location management schemes and mobility modeling method used for cellular networks and also provided the comparison of these methods. M. N. Birje [3] et al. have proposed a prediction based handover model for multiclass traffic in wireless mobile networks by using software agents, considering two cases: local handoff (between BSC's connected to same mobile switching center (MSC)), and global handoff (between BSC's connected to different MSC). Reades et al. [3] monitors the locations of mobile users in an urban environment and studies the dynamics of mobile usage and crowd movement over time.

Mobile OGSI.NET [7] is an implementation of an OGSI based grid container on the .NET hosting environment on mobile devices based on Microsoft's PocketPC. Mobile OGSI.NEt al.lows for Grid service state saving and restoring and distribution of workload among devices with the same types of services, but with the cost of having to change existing services to adhere to the specific Mobile OGSI.NET programming model. AKOGRIMO [37] is a European funded project that is geared to deal with mobility issues in the Grid. The purpose of the project is to evaluate the mobile grid introducing the notion of mobile dynamic virtual organizations through applications that highlight the challenges present in such mobile environments, like e-health, e-learning and crisis management.

'Ibis for mobility' project by Palmer et al. [33] applies grid computing techniques to distributed computing on mobile devices, which includes integrating mobile phones into the grid. Similarly, WIPdroid by Chou and Li [6] is another distributed computing platform for Android. It is based on the Web Services Session Initiation Protocol (WIP), which allows "real-time service-oriented communication over IP". Further, GridGain Systems has succeeded in running the GridGain cloud computing platform on Android phones Kharif[44], but this is still in early stages of development. The GridGain architecture is probably the closest to Hadoop's of all of the grid systems that are being targeted at mobiles. GridGain directly supports deployment on a cloud, and MapReduce is an important feature of the system.

As can be observed, the problem with many of above mentioned works are related to the idea of developing a specialized framework for Mobile Grid computing, however, a better approach to utilize mobile devices would be to engage them as nodes along with desktop systems, as supplement/additional resources. To explain this approach, we report herewith a new extremely light framework/API for developing grid applications that can leverage computing potential of mobile devices either to boost DGVCSs or as standalone desktop/mobile/heterogeneous grids.

III. CHALLENGES AND OBJECTIVE

The potential benefit of the integration of mobile devices into the Grid comes at a cost of resolving many technical difficulties, which have been the theme in many other papers and research projects lately [10], yet none of these provide any implementation methodology to support the said integration. The use of mobile devices in grid computing presents following challenges…

Availability: As the primary purpose of some of these devices is not computing, and the very fact that they are "mobile" implies that they will not be available "locally" for the computing purpose for extended hours. Further, as these devices are "battery powered", battery life is also a major concern.

Processing capability: The processing capability of mobile devices nowadays ranges from a few MHz to ca. 2.x GHz (dual or quad core), or even more, which is comparable with typical workstations of specialized clusters, however, these are still far inferior on RAM and local storage, thus their use is currently expected to be limited for computational grids, rather than data grids, as computational Grids are based on large-scale resource sharing assuming a virtual pool of resources rather than computational nodes [32]

Connectivity: Conventional (wired) networking is neither advisable nor feasible for mobile devices, which defeats very purpose of their being mobile. The other available options, like GPRS are quite slow where as Wi-Fi networking is quite limited in the range. 3G or 4G/LTE networks can be good options, albeit in future.

Platform: Mobile devices meant for different purpose and developed by different venders run different Operating Software (OS-often closed source), and software development tools for some of them are not available altogether. This poses a very difficult task of designing a general architecture that can leverage advantages of a vast majority of software/hardware platforms.

The goal of present investigation is to provide a basic generalized grid mechanism for co-operative processing to allow multi-platform operability, i.e. support for software & hardware heterogeneity. Here, multi-platform is supposed to mean "supporting any device (hereafter referred as workstation or node, unless otherwise specified) comprising computational ability and capable of being connected to the grid via appropriate communication mechanism (eg. Wi-Fi, GPRS, Wired LAN etc.)". In practice however, some prerequisites do exist, such as support for JAVA in present case to contest platform diversity issue discussed above. Further, support for dynamic scalability (on-the-fly addition or removal of nodes) is also a prime criterion, which can address major concerns regarding "Availability", "Processing Capacity" and "Connectivity", wherein the grid-size can be altered dynamically, subject to complexity of the task and availability of resources.

Here, the "User Application" and "Node Application" are collectively termed "Grid Application" (sometimes also referred as middleware). As the "Grid Application" is tailor-made for each individual project, herein we introduce jUniGrid – A generic framework (API) for developing Grid Applications for mobile & fixed computing devices.

IV. THE "JUNIGRID" FRAMEWORK

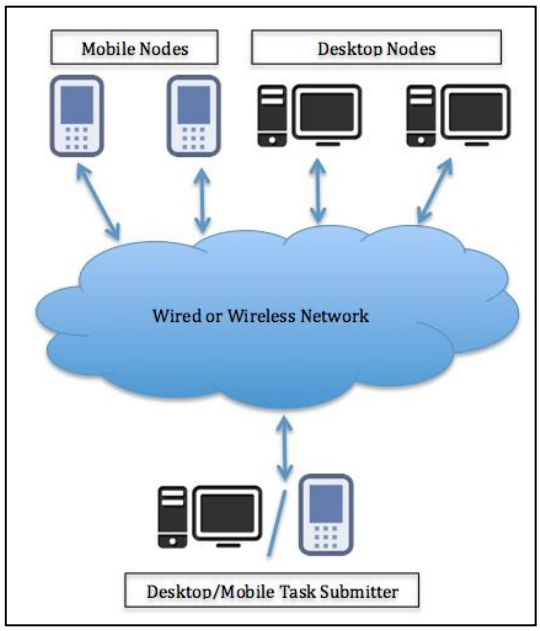

Figure 1: Network diagram of a grid junigrid

Figure 1 shows network diagram of a jUniGrid. The grid operates using two different applications, namely Task-Submitter TS (one instance of TS each per grid in normal deployment) and Node-Application NA (one or more instance on every Node/workstation). Both these applications NA and TS are of the type user-application, developed by the user for particular grid implementation using jUniGrid framework/API. Working together, they collectively serve as Grid Application. Both applications can be deployed on any combination of Mobile device or Desktop System. NA and TS can be connected through any combination of wired or wireless network technology.

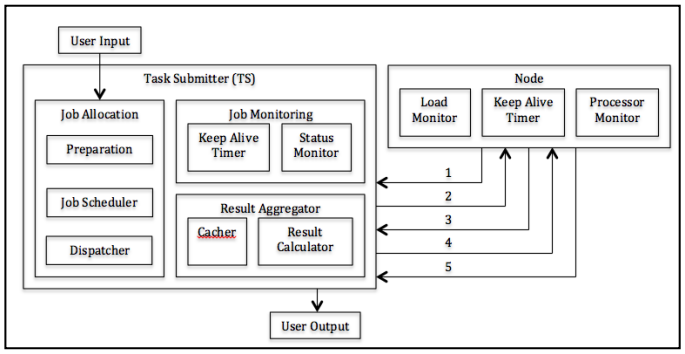

Figure 2: is depiction the architecture and workflow of junigrid

Figure 2 Features representative units of the grid, namely, NA (Node Application or Node) and TS (Task Submitter). The NA executes the job and submits the results to TS. The TS registers and keeps track of Node(s), distributes the jobs to the NA(s), monitors the status of NA(s) and collects the results. If both applications are hosted on different workstations, the type of workstation is identified based on the application it is running (i.e. Task-Submitter and Node). In principle, jUniGrid architecture allows multiple instances of TS (each managing different applications) to run on single or multiple workstations of the grid by appropriate configuration. This may give rise to some interesting possibilities, all of which may not be discussed here for the sake of brevity; however, one particular arrangement deserves a brief mention here, wherein, multiple instances of TS are hosted in a closed network to set up a small task-submitter grid, which is not only a more reliable setup but also a cost-effective alternative of high-end master servers, especially in volunteer grid computing scenarios.

jUniGrid works based on Map/Reduce (split/merge) algorithm. It provides the user with flexibility to split the job and merge the results according to requirement of the job. This split and merge is accomplished by TS, as defined by user. TS comprises of three modules, the responsibilities of which are explained below.

Job Allocation: When user submits job(s) to TS, it generates unique id for every job (preparation) and make a list in FIFO manner for execution (scheduler). The initial status of every job is pending. Job status can be Pending, Assigned, Completed and Fail. jUniGrid maintain job start time, end time, node IP (After assign job to the Node), job Name, job Status for every job. This information is useful for analysis and monitoring. Upon receipt of a job request from NA (Node), the dispatcher dispatches the Jobs to the requesters, subject to availability.

Job Monitoring: Job monitoring for a simplistic framework such as jUniGrid is only required for situations where the device becomes non-responsive while connected with the grid, and as such, may prove to be redundant in a sense that it will scavenge computing power which may otherwise be used for application computations. Nonetheless, redundant or not, a monitoring system is required for any grid expected to scale up.

Result Aggregation: jUniGrid has event based notification mechanism, which is used to reduce (merge) calculation to generate final result. It cache all result in memory or storage (as per configuration) until batch completed.

The applications TS and NA are synonymous with the workstations in the scope of present discussion, thus the following explanation will address only two entities, the applications NA and TS.

Next, we discuss the execution flow and communication between Task Submitter (TS) and Node Application (NA).

1. **[Task Submitter]** Upon inception, TaskInfoManager creates an jobList. JobList contains an object of GridJobInfo. GridJobInfo Object is comprised of a JobHeader and JobBody. NodeHandler listen on configured socket and waits for connection requests from node(s).

2. **[Node]** Upon inception, node continuously requests a connection with Task Submitter until successful. Once a connection is established, it sends a syntectic signal "Requrest_For_Job" and waits until a job is received in the form of a GridJobInfo Object.

3. **[Task Submitter]** When NodeHandler receives any request from Node, It notifies NodeInfoManager to update NodeList accordingly. If the IP of the requesting node is not already registered in NodeList, NodeInfoManager will considers requester as new node. NodeHandler refers to the JobList and assigns the requester with a Job in FIFO manner.

Status of all the Jobs is monitored/managed by NodeHandler. Each new job entry in JobList bears InQueue Status. Once assigned, The status is changed to assigned. If the connection with the node is severed, the job status is reflected as failed. If the node completes and submits the job back, the status is completed. In FIFO assignment of Jobs. If there is no job with InQueue status, all Jobs with failed status are converted to InQueue status.

NodeHandler waits for response from Node(s) concerning assigned Job(s).

4. **[Node]** Upon receipt of a job, in form of GridJobInfo object, the node invokes Execute method of the received GridJobInfo object (which is actual processing – specified by user), upon completion of which the results are filled in the JobBody of GridJobInfo object and JobHeader is updated accordingly and the Node is notified of the completion of processing. So notified, the Node returns the GridJobInfo Object to the Task Submitter followed by requests for a New job. The process is repeated in infinite manner till the node is par of the grid.

5. **[Task Submitter]** When NodeHandler Receives a completed Job from Node in the form of GridJobInfo Object, It updates the JobList and notifies the user about completion of individual job or proceeds to further process the results as specified by the user.

Above exchange continues in a repetitive manner until all the jobs completed.

To implement the jUniGrid framework, one needs to develop two applications TS and NA using the jUniGrid API. The actual computation part is coded in the job class, depicted below:

V. BENCHMARKING, RESULTS AND DISCUSSION

The test application was designed to compare the performance of desktop (Dell Dimension 8200 with Intel Pentium 4, 2.4 GHz processor with 1 GB RAM, running Windows XP SP2, JVM 1.6) and mobile (HTC Nexus One, 1 GHz processor, 512 MB RAM) nodes in grid configurations for computation-intensive tasks. The task was DNA sequence matching (sequence a: 500 Nucleosides, sequence b: 250 Nucleosides – same sequences matched 60 times, using Crochemore-Landau-ZivUkelson Local Alignment method with NeoBio – a library of bioinformatics algorithms implemented in Java. http://neobio.sourceforge.net/).

To measure effect of integrated mobile nodes in grid computing system we performed three tests 1. Grid system has all mobile nodes 2. Grid system has all desktop nodes 3. Grid system has both mobile and desktop nodes in various configurations.

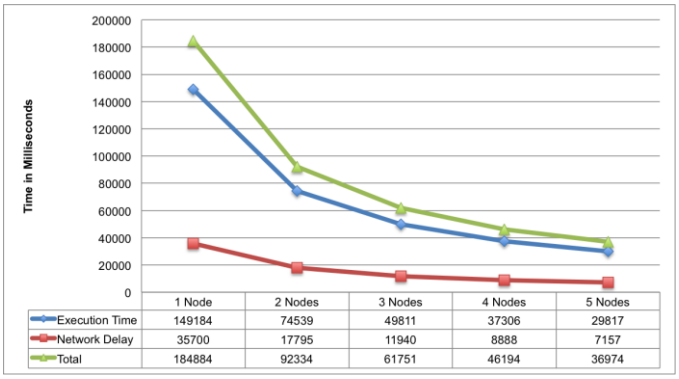

Figure 3: Performance variation with # mobile nodes

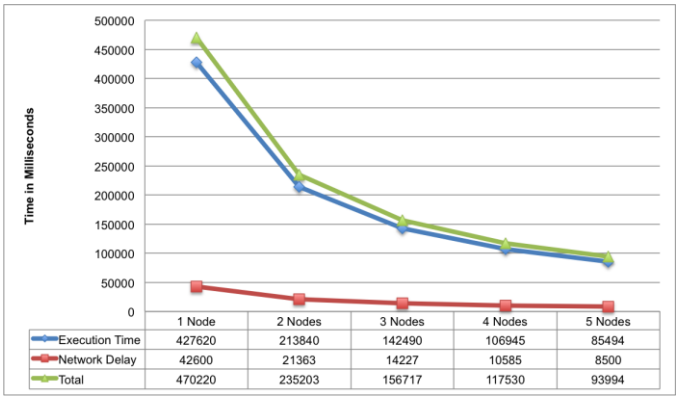

Figure 4: Performance variation with # desktop nodes.

Figure 5: Performance variation with # of mobile and desktop nodes

Figure 5 and Figure 6 show performance of the jUniGrid framework with only mobile or desktop nodes respectively in various configurations. Figure 7 shows the performance with mixed (x mobile + y desktop) nodes. The most obvious observation from the first two graphs is the reduction in resultant network delay due to parallel task submissions, a characteristic of every grid implementation. That is to say, the total network delay remains the same however, as there is no waiting for results involved between submissions or lake of synchronicity (which would be the case with only one node), the tasks can be submitted to different nodes simultaneously. The data in figure 5 and 6 also demonstrate that load balancing is automatically accomplished in such simple test cases. Figure 7 explains the comparative task execution efficiency of the mobile and desktop nodes. Notable inferences are 1. Even when number of mobiles is greater than desktops, the grid performs as expected, albeit slow. 2. When the grid consists equal number of mobile and desktop nodes, the mobile nodes accomplish almost 30% of the total task. 3. When a single mobile node is added to a 5 desktop grid, it still accomplishes 5 tasks (out of total 60). 4. When 2 mobile nodes are added to a grid comprising 4 desktops, they perform almost like a fifth desktop node.

The android mobile phone is capable of executing the job approximately 1/3th as fast as a single computer system, which is substantial, considering limited resources of a mobile phone. Also notice that replacing one computer system of a three nexus one devices (or similar capacity device) grid with a mobile node results.

It should be stressed here that the applications for benchmarking were not optimized to give best results or high computational capacity, rather, the benchmarking is undertaken merely to show the feasibility of the concept and a rough estimate of what is to be gained out of it. In actual implementation however, the node applications as well as jobs can be optimized with respect to network latency, computational capacity range of various devices, RAM usage and application stability, which is sure to show even better results.

Let's see different scenarios in which advantage of mobile grid computing can be leveraged. Analyze the following use cases.

The first and foremost use of jUniGrid is envisaged for ad-hoc grid computing solutions for small to medium organizations to manage computationally intensive tasks with minimal resources, say a few java enabled mobile phones. One of the available mobile phones may be deployed as Master & Task submitter and the rest may work as nodes. Thus a working grid may be up-and-running in a few minutes, with any number of nodes. Also, the number of nodes may be changed dynamically, i.e. the nodes may be added/removed even while the grid is processing. Up-scale extension of such ad-hoc grid will result in what can be termed as "Mobile/Desktop Grid & Volunteer Computing System (M/DGVCS – in analogy with traditional DGVCS)" which utilizes the free resources available in Intranet or Internet environments for supporting large-scale computations (and hopefully storage).

The second scenario is On-Demand expansion of an existing grid system. It is generally not easy to expand the conventional grid systems without considerable expenses (additional hardware). With mobile phones added to the grid, the performance of an existing grid may be boosted up, especially in critical situations where physical expansion of grids is not possible due to technical, local or financial problems. As has been observed in the benchmarking tests, the android mobile connected to an existing grid can provide ca. 30% computing power compared to a conventional node (PC/Laptop). Thus, in critical situations, the performance can be boosted considerably on-demand. This will result in dramatic performance improvement in smaller grids, where this increased processing power will be more visible.

VI. CONCLUSION

In this paper, we have surveyed the challenges for mobile grid computing and introduced a feasible solution for the same, in the form of a lightweight architecture/API for developing grid applications. Although the framework is adaptable universally, its use is primarily intended to integrate mobile devices for conventional or ad-hoc heterogeneous grid implementations to crop the resource potential created by the networked mobile devices. The performance issues are given their due importance and have been examined by appropriate preliminary benchmarking tests. Also discussed are different deployment scenarios and use-cases for implementing mobile grids as real world, rather than conceptual, solutions.

ACKNOWLEDGMENT

Ketan Parmar wishes to acknowledge Mr. Devang Rughani of DevIndia InfoWay for the infrastructural support.

REFERENCES

[1] Akyildiz IF, Lin YB, Lai WR, Chen RJ, A new random walk model for PCS networks. IEEE J. Selected Areas Communication 18, 7 (2000),1254-1260.

[2] Balaton Z, Farkas Z, et al, EdgES: the Common Boundary between Service and Desktop Grids. Parallel Process. Lett, 18 (2008), 433-445.
[3] Birje MN, Manvi SS, Kakkasageri MS, Saboji SV, Prediction Based Handover for Multiclass Traffic in Wireless Mobile Networks: An Agent Based Approach. ICICS, (2007)1-5.
[4] Bruneo D, Scarpa M, Zaia A, Puliafito A, Communication Paradigms for Mobile Grid Users. IEEE/ACM International Symposium on Cluster Computing and the Grid, (2003) 1-8.
[5] Chlamtac,Redi J, Mobile Computing: Challenges and Potential, Encyclopedia of Computer Science, 4th edition, (1998).
[6] Chou W, Li Li, WIPdroid - a two-way web services and real-time communication enabled mobile computing platform for distributed services computing. In SCC '08: Proceedings of the 2008 IEEE International Conference on Services Computing, (2008) 205–212.
[7] Chu D, Humphrey M, Mobile OGSI.NET: Grid Computing on Mobile Devices, (2004). http://www.cs.virginia.edu/~humphrey/papers/MobileOGSI.pdf. Accessed 26 July 2012.
[8] Datta A, Mobigrid: Peer-to-peer overlay and mobile ad-hoc network rendezvous -a data management perspective. the 15th Conference On Advanced Information Systems Engineering, Austria, (2003).
[9] European Grid Forum, http://www.egrid.org Accessed 26 July 2012
[10] Forman GH, Zahorjan J, The Challenges of Mobile Computing. IEEE Computing Milieux, (1994) 38-47.
[11] Foster I, Keeselman C, Globus: A metacomputing Infrastructure Toolkit. Int. J. High Perform. Comput. Appl, 11 (1997), 115–128.
[12] Foster I, Kesselman C (eds.), The Grid2: Blueprint for a New Computing Infrastructure. Morgan Kaufman, (2004).
[13] Foster I, Kesselman C,Tuecke S, The anatomy of the grid: enabling scalable virtual organizations. Int. J. High Perform. Comput, 15 (2001), 200-222.
[14] Franz M, A Fresh Look at Low Power Mobile Computing, (2003). http://research.ac.upc.es/pact01/colp/paper15.pdf Accesssed 26 July 2012
[15] Grimshaw A, et al., Legion: The next logical step towards a nationwide virtual supercomputer. Technical Report CS-94-21, University of Virginia, Computer Sciences Department, (1994).
[16] Heng H, Buyya R, Bhattacharya S, Mobile Cluster Computing and Timeliness Issues. Informatica (Ljubl.), 23 (1999), 5-17.
[17] Hwang J, Aravamudham P, Proxy-based Middleware Services for Peer-to-Peer Computing in Virtually Clustered Wireless Grid Networks. Proc. of International Conference on Computer, Communication and Control Technologies, (2003).
[18] Imdadul Islam MD,Siddique Hossain ABM, A proposed random walk model for mobile cellular network. ICECE, (2004), 386-389.
[19] Jiménez PR, Patiño MM, Kemme B, Enterprise Grids: Challenges Ahead. J. Grid Computing, 5 (2007), 283–294.
[20] Kacsuk P, Podhorszki N, Kiss T, Scalable Desktop Grid System. International Meeting of High Performance Computing for Computational Science (VECPAR'06), Rio de Janeiro, Brazil, (2008).
[21] Kharif O (2008) A warm welcome for android. BusinessWeek
[22] Kiss T, et al., Solving Grid interoperability between 2nd and 3rd generation Grids by the integrated P-GRADE/GEMLCA. Proceedings of the UK e-Science All Hands Meeting, Nottingham, UK, (2006).
[23] Kondo D, et al., Cost-Benefit Analysis of Cloud Computing versus Desktop Grids. 18th International Heterogeneity in Computing Workshop, Rome, (2009).
[24] Kurkovsky S, Bhagyavati, Modeling a Computational Grid of Mobile Devices as a Multi-Agent System. Proc. of the 2003 International Conference on Artificial Intelligence,Las Vegas, NV, (2003).
[25] Kurkovsky S, Bhagyavati, Ray A, A Collaborative Problem-Solving Framework for Mobile Devices. ACMSE,Huntseville, Alabama, USA, (2004).
[26] Kurkovsky S, Bhagyavati,Ray A, Modeling a Grid-Based Problem Solving Environment for Mobile Devices. Journal of Digital Information Management, 2 (2004),109-114.
[27] Litke A, Skoutas D, Varvarigou T, Mobile grid computing: Changes and challenges of resource management in a mobile grid environment. Proceedings of Practical Aspects of Knowledge Management (PAKM 2004), Austria, (2004).
[28] Litzkow M, Livny M, Mutka M, Condor – a hunter of idle workstations. Proceedings of the 8th International Conference on Distributed Computing SysTems. (1988),104–111.
[29] Marinescu DC, Marinescuand GM, Ji Y, Boloni L, Siegel HJ, Ad hoc grids: Communication and computing in a power constrained environment. Workshop on Energy-Efficient Wireless Communications and Networks (EWCN), Phoenix, USA, (2003).
[30] McKnight LW, Howison J, Bradner S, Guest editors' introduction: Wireless grids– distributed resource sharing by mobile, nomadic, and fixed devices. IEEE Internet Computing, 8 (2004), 24–31.
[31] Mudali P, Adigun MO, Emuoyibofarhe JO, Minimizing the Negative Effects of Device Mobility in Cell-based Ad-hoc Wireless Computational Grids. SATNAC: South African Telecommunication Network and Application Conference, 1 (2006), 10-11.
[32] Nemeth Z, Sunderam V, Characterizing Grids: Attributes, Definitions, and Formalisms. Journal of Grid Computing, 1 (2003), 9–23.
[33] Palmer N, Kemp R, Kielmann T, Bal H, Ibis for mobility: solving challenges of mobile computing using Grid techniques. In HotMobile '09: Proceedings of the 10th workshop on Mobile Computing Systems and applications. ACM, (2009) 1–6.
[34] Phan T, Huang L, Dulan C, Challenge: Integrating Mobile Wireless Devices Into the Computational Grid. 8th Annual International Conference on Mobile Computing and Networking (MobiCom 2002), Atlanta, Georgia, USA, (2002) 271-278.
[35] Reades J, Calabrese F, Sevtsuk A, Ratti C, Cellular census: Explorations in urban data collection. IEEE Pervasive Computing. 6 (2007), 30–38.
[36] Redi J, Mobile Computing: Challenges and Potential. Encyclopedia of Computer Science, 4th edn. (1998).
[37] The AKOGRIMO project: http://www.akogrimo.org Accessed 26 July 2012
[38] The Globus homepage. www.globus.org Accessed 26 July 2012
[39] The Grid Forum, http://www.gridforum.org Accessed 26 July 2012
[40] Tuan CC, Yang CC, A Compact Normal Walk Model for PCS, Networks. Mobile Computing and Communications Review, 7 (2004) 20-31.
[41] Tuan CC, Yang CC, A novel normal walk model for PCS networks with mesh cell configuration, Journal of the Chinese Institute of Engineers. 28 (2005), 677-690.
[42] Xue G, An improved Random Walk Model for PCS Networks. IEEE Trans. Communication 50, 8 (2001), 1224-1226.
[43] Zhang J, Handbook of wireless networking and mobile computing, John Wiley & Sons, Inc, (2004).