

Moset: An anonymous remote mobile cluster computing paradigm

M.A. Maluk Mohamed*, A. Vijay Srinivas, D. Janakiram

Distributed & Object Systems Lab, Department of CS & E, Indian Institute of Technology Madras, Chennai, India

Received 1 April 2005; accepted 1 April 2005

Available online 8 August 2005

Abstract

The advance of technology in terms of cellular communications and the increasing computing power of the mobile systems have made it convenient for people to use more of mobile systems rather than static systems. This has seen more of mobile devices in personal and distributed computing, thus making the computing power ubiquitous. The combination of wireless communication and cluster computing in many applications has led to the integration of these two technologies to emerge as Mobile Cluster Computing (MCC) paradigm. This has made parallel computing feasible on mobile clusters, by making use of the idle processing power of the static and mobile nodes that form the cluster. To realize such a system for parallel computing, various issues such as connectivity, architecture and operating system heterogeneities, timeliness issues, load fluctuations on machines, machine availability variations and failures in workstations and network connectivities need to be handled. Moset, an Anonymous Remote Mobile Cluster Computing (ARMCC) paradigm is being proposed to handle these issues. Moset provides transparency to mobility of nodes, distribution of computing resources and heterogeneity of wired and wireless networks. The model has been verified and validated by implementing a distributed image-rendering algorithm over a simulated mobile cluster model.

© 2005 Elsevier Inc. All rights reserved.

Keywords: Mobile cluster computing; Cluster computing; Dynamic-load balancing; Parallel computing; Cluster-subgroups

1. Introduction

Clusters are formed by exploiting the existing computing resources on the network to work together as a single system. Thus, they eliminate the need for supercomputers by providing better price-performance ratio and fault tolerance compared to the traditional mainframes or supercomputers. The availability of high-speed networks and high-performance workstations has made networks of workstations ideal for parallel computing. A few models like the Condor [12], NOW [2] and Batrun [16] have been proposed earlier for improving the utility of workstation clusters, which are connected using a wired network. However, these clusters are formed using only homogeneous devices.

Parallel programming on workstation clusters mostly follow the collection of processes model in which processes communicate via message passing or shared memory abstractions. This model is not suitable for loosely coupled open-ended workstations because of the heterogeneity in architecture and operating systems, dynamic load fluctuations on machines, variations in machine availability and failure susceptibility of networks and workstations. To address these issues, two models namely ARC [8] and DP [7] were recently proposed.

The rapid advances made in technology recently have made mobile devices more powerful (in terms of computing) and ubiquitous (in terms of connectivity). However, it is becoming essential to support seamless computing and communication for mobile users, which would offer flexibility and increased information availability. In addition, it helps in exploiting the capacities of the distributed information over the global network, which could be accessed by the user at any time without regard to the location or mobility. There are many trends that points to the increasingly

* Corresponding author.

E-mail addresses: maluk@cs.iitm.ernet.in (M.A.M. Mohamed), avs@cs.iitm.ernet.in (A.V. Srinivas), djram@cs.iitm.ernet.in (D. Janakiram)

URL: <http://dos.iitm.ac.in> (M.A.M. Mohamed).

widespread use of mobile devices in contrast to the static nodes. For example, it is estimated that the number of wireless internet users is likely to grow from 35.0% to 55.2% by 2007, which indicates that the number of Mobile Hosts (MH) are going to be more than the static nodes that are connected to the network [14]. These factors imply that in addition to static nodes, mobile nodes also constitute a major part of the cluster. This results in the emergence of the essential future paradigm, namely Mobile Cluster Computing (MCC).

In addition, the idea of integrating mobile devices with the backend parallel computing cluster will be essential for many classes of parallel applications such as weather forecasting, earth quake predictions, etc., where the mobile devices can act as data collectors. In such cases, the collected data, might require some kind of processing before being stored at the backend databases. With adequate software support, these classes of applications can handle such geographically independent high-performance computing requests from mobile clients. There are many similar applications like image rendering on a battlefield, surveillance and other military applications that could benefit from the proposed mobile cluster model.

The technological growth is transforming the regular wired structure in most of the university campuses to a wireless structure. This results in a situation where there could be more number of powerful mobile devices in contrast to the static ones. In such environments, it is very clear that significant amount of idle computing power is available on these devices. Thus when these idle computing powers are harnessed along with the idle computing powers of the static nodes, it could provide an immense platform for executing scientific applications for research purpose which is very common in universities. This provides greater need for making mobile devices to be a part of the cluster computing model.

To the best of our knowledge, only three proposals have come out on MCC, namely [20,4,3]. However, [20,3] do not discuss how parallel programming on the mobile cluster could be done. The cluster of [20], just defines and analyzes the potential application environment of MCC, and gives a generic architecture of a MCC. On the other hand, [3] gives a basic architecture of MCC which uses IPv6 as a primary protocol, which relies on network level mechanisms. [4] discusses a prototype of a mobile cluster model namely hybrid cluster computing based on their framework with JAVA mobile objects and the mobile IP. [4] addresses the mobile cluster by making the mobile host as the coordinator and other participating static nodes as donors of computing power. However, it neither addresses mobility of the mobile hosts nor harnessing the idle computing power of the MHs. Further, [4] also does not consider the scenario where a MH acting as a cluster coordinator moves to different cell.

In this paper, we present the Maset, an Anonymous Remote Mobile Cluster Computing (ARMCC) paradigm for parallel programming on a mobile cluster that consists of

both static and mobile nodes. The model addresses the research issue of harnessing the idle computing power of both the mobile and static nodes for parallel computing. It also helps in providing the resource poor device with the required computing power to execute any highly compute intensive application at any time from anywhere. Maset is an extension of the Anonymous Remote Computing (ARC) model [8] over distributed mobile systems. The proposed novel paradigm provides transparency to mobility of nodes, distribution of computing power and heterogeneity of network architecture, thus providing better anonymity to applications. We also present a detailed case study for the proposed model using a computation intense application of image rendering. Our model was successfully implemented on top of a reliable multicast protocol for distributed mobile systems [13].

The rest of the paper is organized as follows. Section 2 gives the key issues involved in parallel computing on mobile clusters. Section 3 discusses in brief the design principles and gives an overview of the proposed Maset model. Section 4 discusses the computational model of Maset. Section 5 discusses the implementation of the Maset. Section 6 describes the image-rendering application briefly and gives the performance analysis of the model and Section 7 concludes the paper with summary and future directions.

2. Issues in mobile clusters and parallel computing on mobile clusters

There has been an increasing interest in the use of clusters of workstations connected together by high-speed networks for solving large computation intensive problems. The trend is mainly driven by the cost-effectiveness of such systems as compared to large multiprocessor systems with tightly coupled processors and memories. However, recent proliferation of mobile devices and advancement in wireless connectivity has made parallel computing on mobile clusters a feasible proposal. Mobile devices can be part of the cluster playing several unique roles. They can be used as a front-end to the cluster functionality, such as submitting a job, managing processes, or viewing statistics. In case of a MH that has very poor computing power, the device must be able to utilize the cluster seamlessly to access the computational power. However, the MH can also be a contributor of computing power to the cluster, as in case of devices such as laptops that have a substantial amount of computing power equal to their static counter parts. Distributing computing power in a cluster consisting of a network of heterogeneous computing devices represents a very complex task. However, it becomes even complicated when mobile devices are also a part of it. There are several key issues that distinguish parallel computing on mobile clusters from that of the traditional workstation clusters, namely

- Asymmetry in connectivity,
- Mobility of nodes ,
- Disconnectivity of mobile nodes ,
- Timeliness issue,

- Changing loads on the participating nodes,
- Changing node availability on the network,
- Difference in computing capability and memory availability,
- Heterogeneity in architecture and operating systems.

2.1. Asymmetry in connectivity

The traditional cluster computing models do not face the problem of heterogeneity in the network connection as the entire set of workstations that are participating in the clusters are connected only by the wired network. Wireless networks deliver much lower bandwidth than wired networks and have higher error rates. Mobile devices are characterized by high variation in the network bandwidth that can shift from one to four orders of magnitude, depending on whether it is a static host or a mobile host and on the type of connection used at its current cell. Thus, the programming model must be able to distinguish among the types of connectivity and provide flexibility for easy variation of the grain size of the task to account for the variations in bandwidth. However, these systems are suitable only for coarse grain level parallelism due to the communication overhead.

2.2. Mobility of nodes

Due to mobility of nodes, the notion of locality becomes important as users move from one cell to the other cell. The locality becomes important as the change in the mobile node's location means a change in the route to that node and consequent communication overhead. The ability to change locations while connected to the network increases the volatility of some of the information. Static data could become mobile in the context of mobile computing. As a node moves, nearby information servers get farther away and should be replaced by closer ones offering the same or more relevant contextual information. Traditional computers do not move, as a result information that is reliant on location can be configured statically, such as the local DNS server or gateway, the available printers, and the time zone. A challenge for mobile computing is to define this information intelligently and supply means to locate configuration data appropriate to the present location. Mobile computing devices need to access more location-related information than stationary computers if they are to serve as ubiquitous guides to a user's environment. As the mobile device moves and as the speed of motion changes, the quality of the network link and other available resources might change significantly. Thus, the system should be able to adjust according to the changing conditions. For example, when a MH that has taken the task moves from one cell to another, then the system still requires tracking these MHs.

2.3. Disconnectivity of mobile nodes

The period of disconnectivity of nodes in static networks are usually treated as faults. However, in the context of mo-

bile nodes, the disconnectivity may be due to roaming (and MH in an out of coverage area) or voluntary disconnection (doze mode) to save battery power.

2.4. Timeliness issue

Timeliness refers to the delay that is taken for the mobile device to regain its full state when it moves from one cell to the other or after reentering a coverage area after disconnection. Timeliness issue is an important issue especially in real-time systems. Whenever a mobile host moves from one cell to the other, it is associated with a handoff, to ensure that data structures related to the mobile host are also moved to the new connecting point (MSS). This involves exchange of several registration messages. This may cause some delay and it should be fast enough to avoid loss of message delivery. In addition to this, there is a possibility that the mobile host could move out of coverage after accepting the task for execution. These issues need to be addressed with respect to the mobile cluster model.

2.5. Changing loads on the participating nodes

When using workstations for executing parallel applications the concept of ownership is frequently present. Workstation owners do not want their machine to be overloaded by the execution of parallel applications, or they may want exclusive access to their machine when they are working. Reconfiguration mechanisms are thus required to balance the load among the nodes, and to allow parallel computations to coexist with other applications. To overcome these problems, some dynamic load balancing mechanisms are needed. There are differences in loads among the nodes due to multi-user environment, and when an application is run on heterogeneous cluster. In these cases, it is important to balance loads among the nodes to achieve sufficient performance. As static load balancing techniques would be insufficient, dynamic load balancing techniques based on runtime load information would be essential. This would be difficult for a programmer to perform load balancing explicitly for each environment/application, and automatic adaptation by the underlying runtime is indispensable. This gets aggravated when mobile devices are part of the cluster.

2.6. Changing node availability on the network

In traditional distributed systems, nodes keep leaving and joining the system dynamically. The join and leave of nodes may be due to either node failure or link failure. However, the system must be smart enough to continue with the computation. The availability of node becomes fuzzier in a distributed mobile computing scenario as the movement of the nodes also affects the availability. It is possible that the node may enter an area which is not under the coverage area of any MSS. It is also possible that node availability is transient

with respect to the execution of the program. While a mobile node is computing a subtask, it can go out of coverage and enter back into the coverage area before the completion of the execution of the program.

2.7. Difference in computing capability and memory availability

As each host may have different capabilities (such as memory) and different processing power, it is essential to allocate tasks to the nodes based on their capabilities and processing power. MHs may especially have lower computing power and memory in contrast to its static counter part.

2.8. Heterogeneity in architecture and operating systems

Although it is reasonable to assume that a new and stand-alone cluster system may be configured with a set of homogeneous nodes, there is a strong likelihood for upgraded clusters or networked clusters to have nodes with heterogeneous operating systems and architectures. As discussed in [8], the operating system heterogeneity could be handled through distributed operating systems. However, it will be non-trivial to handle architectural heterogeneity, since the executable files are not compatible among architectures.

The issues discussed in this section make parallel programming on mobile clusters difficult. With the issue of mobility and other constraints associated with mobile devices, the management of distribution at the programming level further hardens the task. The existing cluster computing models solve only a subset of these issues. None of the earlier work in mobile clusters discusses these except for the timeliness issue that was discussed in [20].

3. Moset overview

The basic principle with which the Moset model was designed was to abstract out the heterogeneity of the constituting devices from the user. The user is made transparent to the hardware, bandwidth, operating system and other heterogeneity existing below the kernel. The user is given freedom to avail any computing power required for his application, without being concerned about whether he is working with a constrained device or not. Moset is designed with clear separation between the administration functionality and the user functionality. It is the function of the administration to install and maintain the system. Once the system is deployed, the user needs only to use the APIs to interact with the system for performing parallel computing. Fig. 1 illustrates the basic architectural overview of the proposed Moset model.

The Mobile Support Station (MSS), which is a static node, covers a geographical region, namely the cell. Mobile nodes, which are within that cell, will be under the control of that MSS, and all communications from or to the mobile nodes in the cell can be made only through the associated MSS.

In our model, the MSS aggregates the computing resources that are within its cell and present to the distributed system as a set of its own resources. The nodes that are participating in the cluster computing are grouped based on the memory capability of the nodes. The nodes which are participating in the Moset kernel register their computing entity to the coordinator of the system, based on their capability. The entire data that needs to be processed is multicast to all the participating nodes in the particular group based on the size of the data. The run-time system of the kernel decides on the anonymous node on which the task is to be executed based on its capability. The details of the architecture are described in Section 5.

Unlike the nodes in a traditional distributed system, the mobile nodes cannot maintain high levels of availability or reliability due to wireless connectivity. Hence, in order to achieve reliable delivery of the data, considering the constraints of the mobile devices, Moset is built over a exactly-once reliable multicast protocol.

4. Moset computation model

The Moset computational model is designed in such a way that it could handle the heterogeneity, fault-tolerance, dynamic load balancing and computing power availability. The dynamic load on the participating systems and the nodes and link failures make the traditional cluster computing model unsuitable for parallel programming on MCC. These issues were effectively handled in [8], however the model does not address the mobile device participation in computation and the issues related to it. The Moset model is aimed to integrate the mobile devices with the static nodes to form a mobile cluster, and to harness the idle computing power, of static and mobile nodes to utilize them for parallel computing.

4.1. Cluster-Subgroups

In our model, we use a notion of Cluster-Subgroups (or Subgroups), based on the memory capability of the nodes. A Cluster-Subgroup refers to the characteristics of the task submitted to that subgroup. Each host (static and mobile) based on its capabilities, joins the respective subgroups. For example, Subgroup LOW may refer to those tasks having memory requirements < 10 MB. A MODERATE Subgroup may have tasks having memory requirements < 50 MB. A HIGH Cluster-Subgroup may have tasks having memory requirements < 100 MB. Tasks with memory requirements > 100 MB may be in Subgroup VERY HIGH.

4.2. Horse power factor and dynamic load balancing

As each host may have different capabilities (such as memory) and different processing power, it is essential to allocate tasks to the static hosts and MHs based on their capabilities and processing power. To incorporate this, each

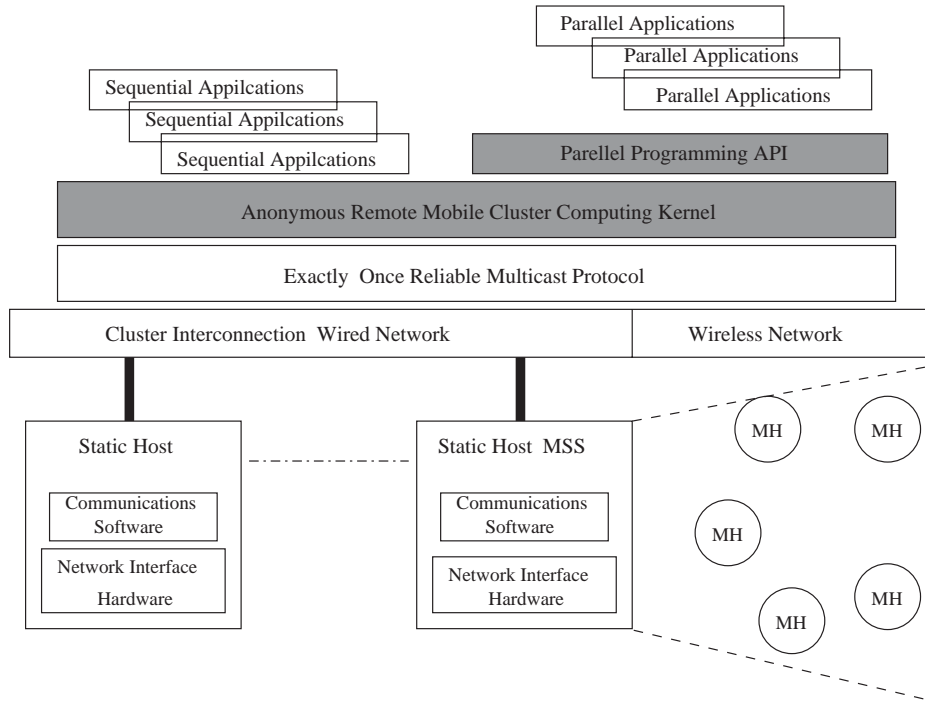


Fig. 1. Moset architecture.

host is allocated an integer called Horse Power Factor (HPF) [8], which is a measure of the computing power of a machine, load on the machine and the network bandwidth of the communication channel. Machines in the network are normalized by a benchmark program to obtain a relative index of the machine, which is a static factor. The dynamic HPF of a machine is obtained using this static relative index, the load on the machine and the communication bandwidth with which the machine is connected to the network. This dynamic factor is normalized as a factor that represents the number of entities that it could compute. When a host has HPF ‘ h ’, then ‘ h ’ computing entities are allocated to the host. For example, if hosts A and B have h_1 and h_2 as their respective HPFs, then the time taken by A to compute h_1 amount of a task is approximately equal to the time taken by B to compute h_2 amount of the same task. Abstracting the heterogeneity in this way makes it viable for parallel processing on unevenly loaded heterogeneous machines. The dynamic communication bandwidth is not taken into consideration in HPF, as measuring dynamic bandwidth may lead to overhead. It is also that to the best of our knowledge no technique has come out with solution that could exactly measure the dynamic bandwidth without introducing significant overheads. This is clear from the fact that the schemes like PathMon [9] (which is a relatively better scheme compared to the other techniques like pathchirp, pathload, etc.) requires about 0.25 s to report the available bandwidth in a wired network, which could be even worse in a wireless channel. This also does not guarantee the exact measurement and is likely to have a relative error of 12%. During handoff when the MH is in the process of receiving data, the HPF

variations matter. But as with current technology relating to channel allocation such as the dynamic channel-allocation techniques [19], this has become a matter of negligence.

Dynamic load balancing [15] is done by maintaining two thresholds viz., Upper Threshold (UT) and Lower Threshold (LT) at the MH. These thresholds are shared by all the computing entities within a MH. This can be achieved by creating the computing entities as threads of the MH. When the load on the MH is greater than UT, a computing entity on that MH leaves the group and increments UT and LT on that MH. Both UT and LT are incremented so that all entities do not leave the groups at the same time. Similarly, when the load on the MH becomes lesser than LT, a computing entity on that MH joins the group and decrements both UT and LT. The need for two thresholds UT and LT is to avoid oscillations of frequent join and leave.

Further, the load balancing mechanism used is non-preemptive [15] and hence migration of already executing task is not done. This is due to the additional communication overhead involved in process migration.

4.3. Parallelism in the model

The dataset that is very large is multicast to the subgroup, based on the size of the file. Multicast provides an efficient mechanism for transferring the data to the computing entities for processing. Each computing entity independently splits the task based on its ID and N . For example, in the case of distributed image-rendering application, frames having frame number ‘ f ’ such that $\text{mod}(f, N) = ID$ are rendered

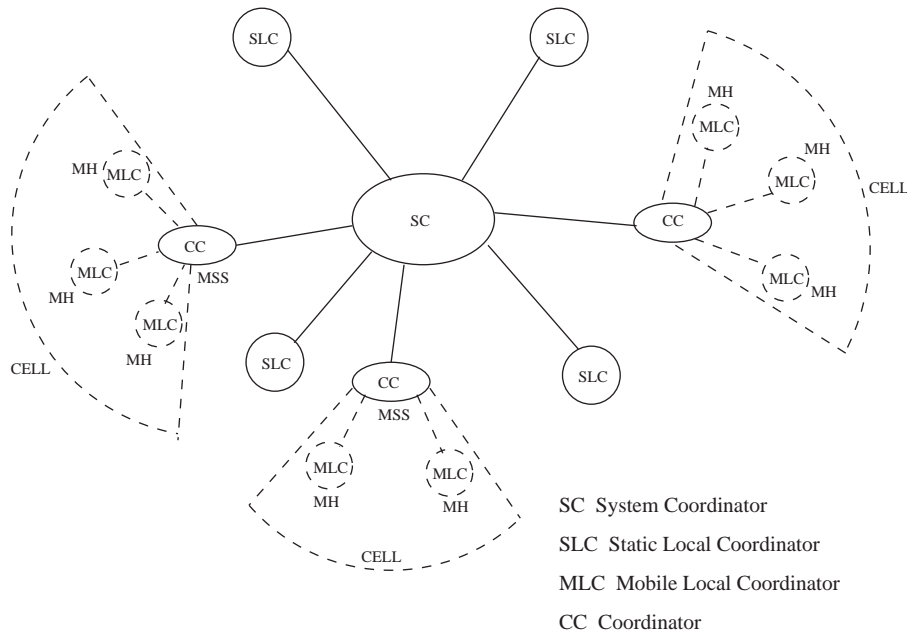


Fig. 2. Maset system structure.

by the computing entity with identifier ID . When an entity completes its share, it sends the result back to the destination host.

5. Implementation

The system structure of Maset is as shown in Fig. 2. A Distributed Maset kernel is spread over the nodes that participate in Maset. Maset kernel consists of multiple Local Coordinators (LC) to coordinate local activities, multiple Co-Coordinators (CC) to coordinate the global activities within their cell, and one System Coordinator (SC) to coordinate the overall global activity of the entire cluster. Each node, either static or mobile that intends to participate in the Maset kernel runs a local coordinator. MH has a client process and a daemon. The client process and daemon run over a reliable multicast protocol. The client processes are used to submit tasks to the MHs (via multicast protocol) for distributed processing. The daemons are the computing entities at the MHs that execute a part of the submitted task concurrently with other daemons.

5.1. Local coordinator

The local coordinator runs on mobile and static hosts that participates in the Maset system, either to improve utilization or to share its work with an anonymous remote node. Any Maset communication to or from the local processes is achieved through the local coordinator. In case of image-rendering application, the huge dataset which is to be rendered is multicast by system coordinator to the subgroup

local coordinators, based on the size of the file. Each local coordinator independently splits the task based on its ID and N . For example, frames having frame number ' f ' such that $\text{mod}(f, N) = ID$ are rendered by the computing entity with identifier ID . When an entity completes its share, the local coordinator sends the result back directly to the system coordinator if it is executed on a static host or through the co-coordinator if it is executed on a mobile host.

5.2. System coordinator

The Maset kernel has one system coordinator in the logically grouped mobile cluster. The functions of the system coordinator are to manage all the nodes that take part in the cluster process, to manage the spawned computing entities, to coordinate all the functions related to task distribution and execution, as well as to maintain the migration history of the tasks.

Whenever a static machine wants to participate in distributed processing, thus enhancing the machine utilization, it spawns computing entities based on its capabilities through its local coordinator. The static machine also includes the MSS which spawns a set of computing entities representing the MHs which are within its cell and interested in participating in the computation. As discussed in the previous section, the system coordinator groups these computing entities into cluster-subgroups based on the memory capacity of the machine which has spawned the entities. The system coordinator keeps track of the total number of computing entities (called N) under each cluster-subgroup. Further, the system coordinator assigns each computing entity a unique membership identifier (called ID , ranging from 0 to $N - 1$) asso-

ciated with each group subscribed by it. The exact number of computing entities spawned by each daemon will depend upon the HPF of the node.

The dataset which is very large is multicast to the subgroup's local coordinators by the system coordinator directly to the static hosts local coordinator, and through the coordinators to the mobile hosts local coordinator, based on the size of the file. Multicast provides an efficient mechanism for transferring the data to the computing entities for processing. A history of recent migrations is maintained.

The single point failure of the system coordinator is handled by replicating the state of the system coordinator in another nearby static node so that in case of failure the states are not lost and the system could still survive. When a local coordinator identifies that the system coordinator has failed it initiates the process of identifying the next system coordinator using an election algorithm [6].

5.3. Co-coordinator

The Moset has multiple co-coordinators running on MSS, which has at least, one MH participating in the Moset kernel. The co-coordinator acts a system coordinator with respect to the MHs which are within its cell. Any MH within the coverage area of the MSS, which wants to participate in the sharing of resource, will register a set of computing entities to the co-coordinator running on that MSS. The co-coordinator collects the set of computing entities spawned and registers with the system coordinator. The co-coordinator takes care of multicasting the dataset to be rendered to the participating MHs and also maintains the history of the execution that takes place in the MH within its cell.

The co-coordinator also takes care of the mobility of the MHs. When a MH takes the frames for rendering and moves out of the cell, and enters another cell, then the new MSS through handoff will be able to inform the co-coordinator of the old MSS. If the new MSS already has the co-coordinator daemon running then it continues with the process by exchanging the information among the co-coordinators. In case if the new MSS does not run the co-coordinator daemon, then it gets registered with the system coordinator and runs the co-coordinator.

5.4. Timeouts, mobility and fault tolerance

Timeliness issue is an important issue especially in real-time systems. However, in cluster computing systems, the system is mainly meant for computation intensive problems, like environmental modeling where the factor of time can be relaxed. But still the workstation cannot take an infinite amount of time for executing the subtask which it has accepted to execute. To handle this, timeout mechanisms are used. The timeouts are maintained by the system coordinator and the co-coordinators. A timer is a data counter that ticks at regular intervals. If the workstation do not return

Table 1
System parameters

Parameter	Value
Number of groups	4
Number of MSS	32
Wired bandwidth	100 Mbps
Wireless bandwidth	10 Mbps
Wired propagation delay	0.5 ms
Wireless propagation delay	0.0 ms
Message loss probability	0.001
NACK loss probability	0.00
Out-of-range probability	0.00
NACK transmission period	1 s
MCASTREQ timeout	25 ms
SYNACK timeout	500 ms

within the stipulated time set in the timer, then the subtask is re-submitted to some other idle workstation for getting executed or in the worst case gets executed in the coordinator. In case of subtasks executing in MHs the co-coordinator takes care of the timer. When a subtask assigned to a MH does not return before the timeout then the co-coordinator tries to reassign the subtask to some other MH within the subgroup within the cell or as a worst case executes the task itself.

The system coordinator detects failure of remote node when the local coordinator fails. However, this will work only with static nodes. Mobile nodes may move out of the cell after taking the task for execution and return before the timer timeouts. In this scenario, as the MH was out of coverage for a while, the co-coordinator will be able to detect this and cannot decide that the MH has failed. Thus, the co-coordinator needs to wait until the timer timeouts. This ensures fault tolerance of the system.

6. Performance

The Moset approach provides parallel programming on a mobile cluster thus improving the utilization of the computing resources of the participating nodes. Moset provides for heterogeneity, fault tolerance and dynamic load balancing to parallel computing. The performance study of the model was done by implementing the distributed image-rendering application over the FTEORMP [13]. The FTEORMP is an exactly once reliable multicast protocol. The simulation of FTEORMP was carried out on an object-oriented discrete event simulator in C++ similar to that used in [1]. The parameters used for simulation are given in Tables 1 and 2. The cell permanency time in Table 2 refers to the average time for which a MH will be inside a cell.

Each MH has a client and a daemon. The clients are used to submit tasks to the MHs for distributed processing. The daemons are the computing entities at the MHs, that execute a part of the submitted task concurrently with other daemons. Both client and daemon run over FTEORMP and use the socket abstractions of FTEORMP simulator. The daemons run on the SPARC machines and communicate with the

Table 2
Tunable parameters

Parameter	Value
Number of mobile hosts	512
Number of senders	4
Message rate	250 messages/s
Message size	256 bytes
Cell permanency time	1 s

respective MHs of the simulator by sockets. The image-rendering application developed renders an image obtained by CT scan. The characteristics of a CT scan image [18] are that they contain information from a transverse plane only. CT scanners produce 3-D stacks of parallel plane images that each consists of an array of X-ray absorption co-efficients. Due to the availability of stacks of parallel plane images, the volume data sets can be viewed as a 3-D field rather than individual planes.

The stack of planes is converted to an image by volume rendering [18,10] using ray-casting. A ray-casting algorithm casts parallel rays from the viewer into the volume. At each point along the ray, the progressive attenuation due to particle fields is computed. At the same time, the light scattered in the eye direction from the light source is also computed at each point. This rendering procedure is used in the volpack library [17]. The volpack library is an implementation of [10] and is used in this application.

The task was submitted to render 360 frames of the CT scan data (a frame for each of image rotation). An online image animator written in C, using X11 graphics, [5] was used to animate the frames as and when they arrive from the entities. Due to an online animation, the real-time effects (such as fast rendering when more hosts are involved) were noticed.

The CT scan data of a human head with skull partially removed to reveal the brain is taken as the input data. The CT scan data consists of 84 slices of 128×128 samples each. The final output frame is 256×256 pixel image. The daemons were run on SPARC 333 MHz and SPARC 500 MHz machines having 256MB RAM. The daemons use FTEORMP simulator [13] for multicasting to other MHs. FTEORMP simulator provides socket abstraction to the applications on the MHs. Hence, the daemons run on the SPARC machines and communicate with the respective MHs of the simulator by sockets.

Table 3 shows the results when daemons were executed on SPARC 333 MHz and SPARC 500 MHz under no load condition. On a SPARC 333 MHz, super-linear speedup is noticed when number of hosts is 2. On a SPARC 500 MHz, super-linear speedup is noticed when the number of hosts are 2, 3 and 4. The reason for super-linear speedup is because, during the entire computation of 360 frames, the full volume data (which is reasonably large—1.37 MB) needs to be in memory. During this time, page faults occur and the computation time increases. The context switch time is very negligible as the experiment was conducted in no load con-

Table 3
No load performance analysis for ultra SPARC 333 MHz and SPARC 500 MHz

Number of hosts	SPARC 333 MHz		SPARC 500 MHz	
	Time taken (s)	Speedup	Time taken (s)	Speedup
1	434		303	
2	211	2.056	149	2.033
3	149	2.912	102	2.971
4	113	3.841	77	3.935
8	61	7.232	42	7.214
16	47	9.234	33	9.181

ditions (when the loads on the machines were only due to the daemons). When the task is split to two hosts, computation time is only for half the task. Hence, the data needs to be in the memory only for a much lesser time. As the memory residence time reduces, the page fault overhead also decreases. Reduction in these overheads (such as page faults and context switches, if any) and by overlapping computation and communication across hosts, the communication delay seems to be nullified. Thus, super linear speedup is achieved. But, as the number of hosts increases to 8 and 16, the communication delay seems to dominate. This is partly because, multicast simulator runs on a single host and is not distributed. So, a multicast is simulated by multiple unicasts. If multicast is done physically (or a distributed simulator is used), then the communication overheads can be reduced further. Scalability limitation is also due to the problem size. If problem size is increased, with more nodes better speedup could be attained.

Fig. 3 shows the effect of load on total execution time and speedup, respectively, for rendering 360 frames. The figure shows the execution time and speedup in the presence of multiple half-load processes in the processor run-queue. The half load on the processor is achieved using a program called cpuhog [11]. As the cpuhog loads the processor only half the time, it creates a half-loaded environment. The first graph in Fig. 3 shows that the execution time increases as the number of half-loaded processes increase. This is due to the increased context switch time, due to an increase in the number of processes. The second graph in Fig. 3 shows that speedup is almost constant irrespective of the load. If the number of hosts involved in computation increases, then speedup seems to increase very marginally with the load on the host. The effects of load on the execution time and the speedup where taken by using cpuhog with memory of 1 MB.

Apart from loading the processor, cpuhog also hogs the memory resource. The image-rendering application needs 10 MB of memory space and if cpuhog occupies more memory, then a lesser amount of memory is left for the image rendering application. By varying the memory used by the cpuhog, the effects on execution time and speedup of the image rendering application can be seen in Fig. 4. These figures show an interesting behavior explained below.

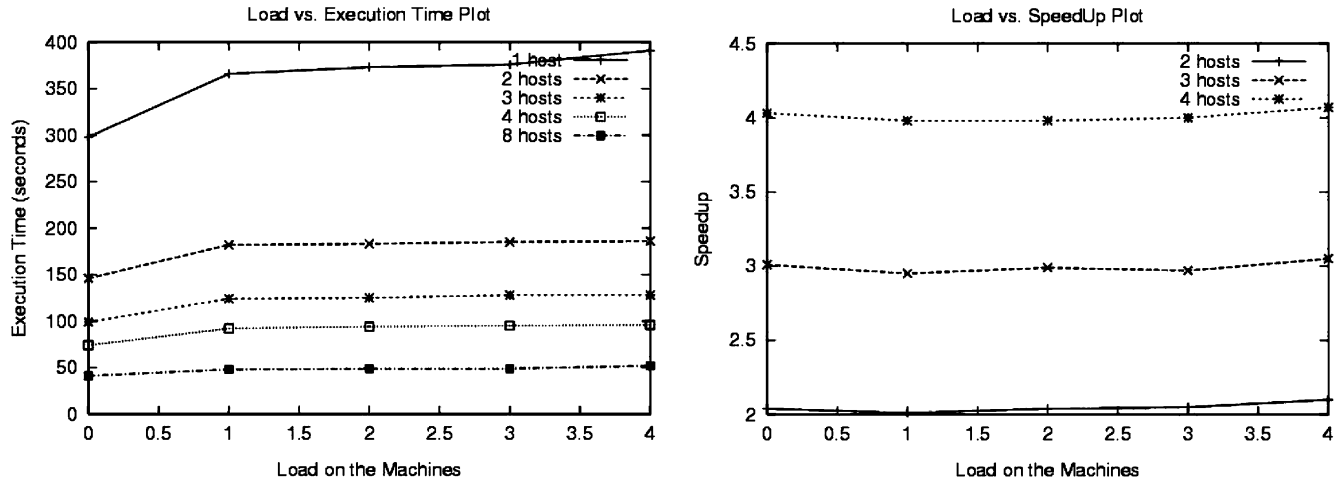


Fig. 3. Effect of load on execution time and speedup.

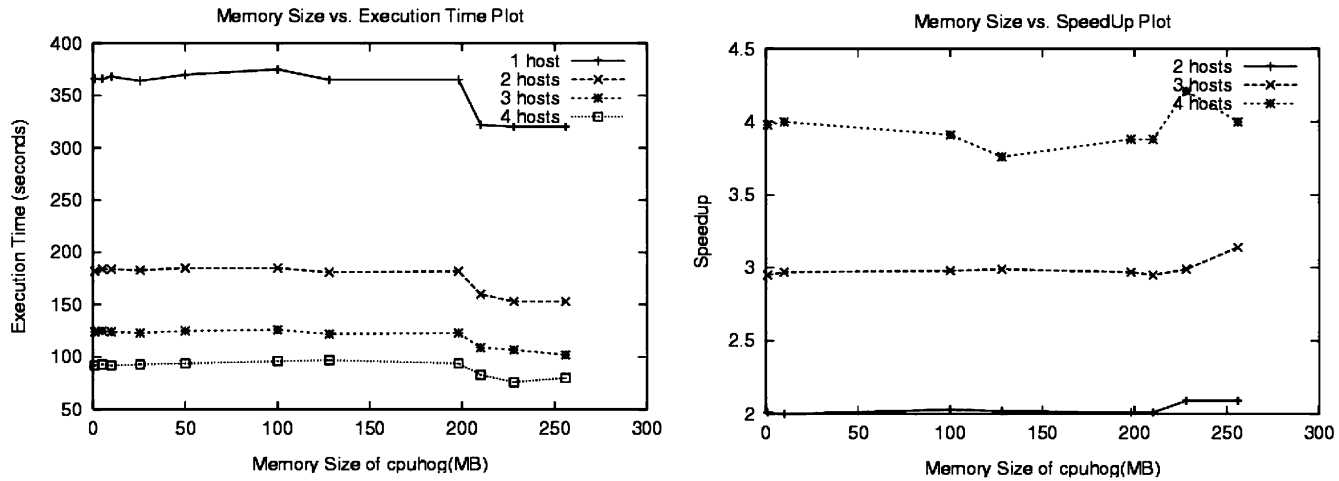


Fig. 4. Effect of process memory size on execution time and speedup.

The first graph in Fig. 4 shows that as the memory of cpuhog increases from 1 to 200 MB, the execution time for the image-rendering application is almost constant with only a very marginal increase. This is because there are not much page faults until the memory occupied by cpuhog increases to threshold. The reason for this behavior is that, due to the availability of large memory space (256 MB), most of the pages are free. However, when the memory occupied by cpuhog increases above a threshold (200 MB), cpuhog encounters frequent page faults. The main reason is that as cpuhog occupies a large number of pages, the oldest page in memory is very likely to be the page belonging to cpuhog. Hence, the oldest page of cpuhog in memory is swapped out and a new page for cpuhog is paged in. As a result, cpuhog sleeps for more time waiting for the required pages to be brought into memory. During the time when cpuhog sleeps, the image-rendering application runs. Hence, the total execution time of the application drops sharply as the application gets more time to run.

The second graph in Fig. 4 shows the effect of memory used by cpuhog, on the speedup of the image rendering application. The speedup is almost constant as the memory of cpuhog increases until 200 MB. Upon further increase in the memory usage, the speedup shows a marginal increase. Moreover, when the application is executed on 4 hosts, super-linear speedup of 4.21 is noticed. Thus, the effect of load on the processor and the memory usage by cpuhog has only a marginal effect on the execution time and the speedup of the image-rendering application.

7. Conclusions and future work

This paper has presented Moset, an anonymous remote mobile cluster computing model based on integrating mobile and static nodes as clusters interconnected by wireless and wired networks. The model handles the heterogeneity in architecture, operating system, network connectivities and

other constraints related to the mobile device. Maset provides one of the first comprehensive efforts at integrating mobile devices into the cluster and effectively harness the computing power of mobile devices. Further, mobile devices can also utilize large computing power available in the cluster seamlessly. A performance study of an image-rendering application also demonstrates the feasibility of the idea.

As a future work, the model could be extended to provide inter-subtask communications. Problems that exhibit a high communication to computation ratio may not be suitable for this model, whereas if the computation time can be overlapped with communication delays, significant speedups can be achieved, even in mobile clusters. However, this overlapping is a non-trivial task. Considerable work needs to be done to evolve a generic model for cluster computing and communication for mobile systems. Another interesting direction we are currently pursuing is to extend Maset to a mobile grid, a global collection of resources.

Acknowledgments

The authors would like to thank the anonymous referees for their useful comments. We also acknowledge the support of the Department of Science and Technology (DST), Govt. of India, for sponsoring part of the research project.

References

- [1] G. Anastasi, A. Bartoli, F. Spadoni, A reliable multicast protocol for distributed mobile systems: design and evaluation, *IEEE Trans. Parallel Distrib. Systems* 12 (10) (October 2001) 1009–1022.
- [2] T.E. Anderson, D.E. Culler, D.A. Patterson, The NOW Team. A case for networks of workstations (NOW), *IEEE Micro.*, 15(1) (February 1995) 54–64.
- [3] A. Basit, C.-C. Chang, Mobile Cluster Computing using IPv6, in: *Linux 2002 Symposium*, Ottawa, Canada, June 2002.
- [4] L. Cheng, A. Wanchoo, I. Marsic. Hybrid Cluster Computing with Mobile Objects, in: *Proceedings of the Fourth International Conference on High-Performance Computing in the Asia-Pacific Region (HPC-Asia 2000)*, Beijing, China, May 2000, pp. 909–914.
- [5] V.R. Devanathan, EOMP: An exactly-once multicast protocol for distributed mobile systems. Master's Thesis, Department of C.S.E., IIT Madras, Chennai, India, 2002.
- [6] H. Garcia-Molina, Elections in a distributed computing system, *IEEE Trans. Comput.* 31 (1) (1982) 48–59.
- [7] B.K. Johnson, R. Karthikeyan, D. Janaki Ram, DP: a paradigm for anonymous remote computation and communication for cluster computing, *IEEE Trans. Parallel Distrib. Systems* 12 (10) (October 2001) 1052–1065.
- [8] R.K. Joshi, D. Janaki Ram, Anonymous remote computing: a paradigm for parallel programming on interconnected workstations, *IEEE Trans. Software Eng.* 25 (1) (January 1999) 75–90.
- [9] D. Kiwior, J. Kingston, A. Spratt, PATHMON, A Methodology for Determining available Bandwidth over an Unknown Network, The MITRE Corporation, available at <http://www.mitre.org/>, March 2004.
- [10] P. Lacroite, M. Levoy, Fast volume rendering using a shear-warp factorization of the viewing transformation, *Comput. Graphics* 28 (Annual Conference Series) (1994) 451–458.
- [11] D. Libenzi, CPUHOG—A kernel scheduler latency tester, Free Software Foundation, Inc., Boston, MA, USA, 2001.
- [12] M. Litzkow, M. Livny, M.W. Mutka, Condor—a hunter of idle workstations, in: *Proceedings of the Eighth International Conference on Distributed Computer Systems*, June 1988, pp. 104–111.
- [13] M.A.M. Mohamed, D.J. Ram, Fault-tolerant exactly-once reliable multicast protocol for distributed mobile systems, in: *Proceedings of the IASTED International Conference on Communication Systems and Networks (CSN 2003)*. Malaga, Spain, September 2003.
- [14] A. Puder, K. Romer, Internet users forecast by country, *eTForecasts*. www.etforecasts.com, 2003.
- [15] M. Singhal, N.G. Shivaratri, *Advanced Concepts in Operating Systems*, McGraw-Hill Inc., New York, 1994.
- [16] F. Tandary, S.C. Kothari, A. Dixit, W. Anderson, Batrun: Utilizing idle workstations for large-scale computing, *IEEE Parallel Distrib. Technol.* 4 (2) (Summer 1996) 41–49.
- [17] Volpack, Volpack—A Volume Rendering Library. www.graphics.stanford.edu/software/volpack/, 1995.
- [18] A. Watt, M. Watt, *Advanced Animation and Rendering Techniques: Theory and Practise*, Addison-Wesley Publishing Company, Reading, MA, 1992.
- [19] J. Yang, D. Manivannan, M. Singhal. A fault-tolerant dynamic channel allocation scheme for enhancing QoS in cellular networks. in: *Proceedings of the 36th Annual Hawaii International Conference on System Sciences (HICSS'03)—Track 9*, Big Island, Hawaii, 2003, pp. 306–315.
- [20] H. Zheng, R. Buyya, S. Bhattacharya, Mobile cluster computing and timeliness issues, *Informatica: Internat. J. Comput. Inform.* 23 (1) (1999).



M.A. Maluk Mohamed is pursuing his Ph.D. in the Distributed & Object Systems Lab, Department of CS & E, Indian Institute of Technology, Madras. He has obtained Masters in Engineering from National Institute of Technology, Tiruchy in 1995 and a Bachelors in Engineering from the Bharathidasan University in 1993. His research interests span distributed mobile systems, grid computing and software engineering. He is a student member of the ACM, IEEE, ISA, IARCS and life member of the Computer Society of India.



A. Vijay Srinivas obtained the M.S. (by Research) degree from the Distributed & Object Systems Lab, Indian Institute of Technology, Madras in 2001 and a Bachelors in Engineering from the University of Madras in 1998. He is currently pursuing his Ph.D. from the same lab under Prof. D. Janakiram. His research interests include parallel and distributed computing, distributed mobile systems, grid computing, object technology and software engineering. He is a student member of the ACM, IEEE and IARCS.



D. Janakiram obtained the Ph.D. degree from the Indian Institute of Technology, Delhi in 1989. He is currently a professor in the Department of Computer Science and Engineering, Indian Institute of Technology (IIT), Madras, India. He coordinates research activities for the Distributed and Object Systems Group at IIT Madras. He has taught courses on distributed systems, software engineering, object-oriented software development, operating systems, and programming languages at graduate and undergraduate levels. He is a consulting

engineer in the area of software development for various organizations. He served as program chair for the Seventh International Conference on Management of Data. He is one of the founders of the Forum for Promotion of Software Engineering, a special interest group in the area of software engineering and object technology. His research interests

include distributed computing, grid computing, distributed mobile systems, wireless sensor networks, object technology, software engineering and distributed & object databases. He is a member of the IEEE, the IEEE Computer Society, the ACM, and a life member of the Computer Society of India.