CHAPTER 4

AN OPTIMIZED PARALLEL COMPUTING PARADIGM
FOR MOBILE GRIDS BASED ON DSOM

4.1 INTRODUCTION

Parallel computing comprises several processors or computers that collaborate with each other to accomplish a common goal in which numerous estimations are carried out simultaneously. Parallel computing is an important aspect of mobile distributed system to minimize fault tolerance, computational time and overhead. According to the work load, it allocates the computations in a flexible way. When parallel computing methodologies are established from the reduced mobile clusters to extensive mobile grids, they are prone to issues like high latency or jitter, processing speed, communication overhead, and low data transfer rate (Keckler, et al. 2011).

Parallel and distributed mobile computing entirely makes use of the computing power of mobile systems. These systems must preferably offer flexible communication, unlimited computing capacity and higher availability of the mobile distributed information (Delporte-Gallet, et al. 2011). The computing resources of a distributed mobile system are integrated to work in a frequent system. A cluster of mobile clusters generates a mobile grid that can be described further as a mobile cluster of mobile clusters.

When Mobile Heads (MHs) create a portion of the mobile grid, it can be characterized as both ‘consumer’ and ‘provider’ of services and other data resources. The tasks are computationally complex in mobile grid and
dispersed across multiple hosts. This necessitates a collective process between distributed tasks. Parallel computing, on mobile distributed system minimizes the computation time, but also endures some issues like load variation, limited node availability, and heterogeneity in processor speed, network speed, operating system (OS), and architecture.

A mobile distributed system comprising interconnected mobile clusters manages a massive load fluctuation on individual mobile nodes. The unexploited network computing capacity must be employed in every way for maximum efficiency. The program that executes on the mobile grid is load adaptive and the communication processes are executed as a collection of processes (COP). By using the COP model, a programmer will not be able to identify the fluctuating network load patterns. During the execution of the program, the initiator node gets heavily loaded and minimizes the system performance.

In general, mobile distributed systems involve varying numbers of active and inactive mobile nodes over a period of time. This should not affect the computation time, but there would be a significant change with larger differences in the number of mobile nodes. When two different program instances use different number of mobile nodes, the occurrence of node failures is less relative to a single program execution (Kumar and Garg 2010). When a mobile node or a connection crashes during a subtask computation, the mobile node may be activated before the completion of the program execution.

The variations in the network speeds and processor speeds result in the higher communication overhead of the mobile distributed systems. This makes them suitable only for common parallelism. When the communication overhead is increased, the speedup of parallel processes is decreased. The
processors in a heterogeneous cluster have varying speeds and the effective capacity may vary due to severe loading. Thus, appropriate grain sizes must be selected during the runtime as an effect of load variation. So, the programming paradigm must be designed for flexible task grain size.

4.2 SURROGATE OBJECT MODEL (SOM)

The mobile host that participates in the mobile grid system as a unique object is considered as the Surrogate Object (SO). SOM is used to manage the asymmetry problem in the distributed mobile systems. The surrogate object is the typical model for a specific mobile host in the wired network. It upholds application specific data structures and approaches. SOM preserves the cached data that stored in the mobile host and minimizes wireless data transfers (Mohamed, et al. 2005). The important features of the SOM are mentioned as:

- SOM offers an understandable and easy solution for managing the asymmetry problems in distributed mobile systems.
- SOM enforced the constraints of host specific and application specific.
- It performances as a data source for controlling data dissemination.
- SOM provides data access for the mobile in both client-pull and server-push models.
- SOM can also cache mobile host specific data and minimize the comeback times for several queries of clients.
- This model offers best exploitation of wireless bandwidth, by means of knowing the present connectivity of the network.

For distributed mobile system, the SOM states an architecture that permits mobile devices to contribute flawlessly in communication and
computing. This associates the mobile devices that connected together with its support environment. It can be accomplished by generating the surrogate in the static network. This model can remain active and would be transparent to the instability of wireless communication. The benefits behind this are as follows:

- It upholds the information of the location about the mobile devices.
- It’s performs as a placeholder, which recognizes local caching for the rapid access of information.
- It manages the mobile host distribution.
- Depends upon the present location of the mobile host and its connectivity constraints, SOM acts as a data sink.
- This can gather data from different sources and further it distributes the suitable data to the mobile host.

4.3 DISTRIBUTED SHARED OBJECT (DSO)

The shared objects consist of two key principles: (1) all operations of an instance object are atomic and serializable, i.e., updating the order in non-deterministic manner, and (2) the shared objects implement only on single object operations. The distributed objects are the group of local objects that interconnects the object user with the shared state (Homburg, et al. 1995). The enhancement over the remote object model determines that it is not limited to a small set of predicted communication patterns.

The Distributed Shared Objects (DSO) are used to design the distributed system and also to express the communication at a high level of abstraction (Homburg, et al. 1996). This utilizes the idea of a shared object to encapsulate communication by giving operations of user-defined based on shared state. It affords the model of single and high level communication and can be used in various applications. It makes simpler interaction among the
applications (Van Steen, et al. 1995). Simultaneously, it acknowledges for effective communication that is established by the implementation of application-oriented system.

The major benefits of DSO are as follows:

- A distributed object offers a precise interface to its applications.
- The application is unique for the implementation process of communication, replication, and consistency.
- This authorizes the application of objects to be loaded during runtime.
- In a distributed object, endurance and communication are exclusively encapsulated.
- An efficient implementation is feasible beyond the operating system (OS).
- An address space is implicitly segregated.

A DSO encompasses one or more interfaces with the set of methods. The objects used in the DSO are inactive and at the same time numerous methods may access the similar object. Variations made by one process to the state of an object are detectable to the other process. A major difference with other models is declared as objects distributed physically, i.e., the active copies of the state of an object exist in several machines simultaneously. The implementation aspects such as communication protocols, replication approaches, distribution and migration state are the portion of the object. These are concealed subsequently in its interfaces and are distinct from the remote objects.

In this research work, an efficient and optimized parallel computing paradigm for mobile grids is developed. Distributed Shared Object Model (DSOM) is integrated with Mobile Distributed Pipes (MDP) to formulate the
environment with the inter-task communication across the distributed shared objects. DSOM is based on the Surrogate Object Model (SOM) and Distributed Shared Object (DSO). SOM is preferred to improve the sharing of resources in mobile grid computing, while DSO is selected to minimize the computational complexity. By incorporating SOM and DSO, the following advantages can be accomplished:

- DSO is employed to maximize the information processing capacity, service sharing by providing context and location sensitive information.
- The problems due to the asymmetry of the mobile distributed systems in network connectivity, mobility, and computing power are answered by SOM.
- The heterogeneity in operating system, system architectures, and load variations are answered in a fair manner by the DSO approaches.
- But, the unused computing determinant is used by SOM to save the processing time.
- The transparency of the DSO model in terms of distribution and heterogeneity decreases the computational complexity.
- Whereas, SOM is chosen to enhance the resource sharing of mobile grid computing.
- DSO also develops the load adaptability and fault tolerance for parallel programs in the mobile grid.

### 4.4 DSOM BASED MOBILE GRID

The proposed mobile grid is defined as a collection of mobile clusters (Mohamed 2011). Each cluster is a combination of Mobile Hosts (MHs) and Static Hosts (SHs) grouped logically/virtually. The MHs are served by a conventional Mobile Support Station (MSS). Each cluster is managed by
a SH designated as Cluster Head (CH). The role of a CH is to manage the services and resources within its cluster. MSS and CH can be assigned to the same host when the MSS load is less. The communication between the CHs of the mobile grid is performed by a P2P overlay (Liu, et al. 2010).

4.4.1 Visualization of Mobile Grid

The example mobile grid structure in terms of distributed shared objects is shown in Figure 4.1. The mobile grid consists of Mobile Clusters (MCs), Cells, Mobile Support Stations (MSSs), Proxy Objects (PO), Cluster Head (CH) which is also a Static Host (SH), P2P overlay for the communication between the CHs, and Mobile Hosts (MHs).

Each MH is visualized by a PO in the wired network. This ensures the conservation of transparency to the imbalance in wireless communication. An active PO monitors the information pertaining to the current state. When a MH comes into a specific cell, it transmits a control signal to the MSS of the cell, which contains the address of the MH’s former MSS. When a MH enters the cluster for the first time, the address of the former MSS is specified as NULL, and a PO is created for complete encapsulation of the mobile device.

The object pointer to a new PO is transferred to its MH for its further communication with the PO. A unified object model can be accomplished by visualizing the other MHs and SHs as independent objects. The multiple interface of an object represents the multiple nodal services. The mobile nodes involved in the computations are optimized into distributed mobile objects and the mobile grid is converted from a nodal collection into a distributed mobile object. These mobile objects form the basic blocks of the mobile grid.
Figure 4.1 An overall proposed architecture of mobile grid based on Distributed Shared Object Model (DSOM)

The abstraction of MH into PO addresses most of the critical issues affiliated with the MH (Mohamed, et al. 2005). Some of the relevant aspects of proxy objects are as follows:
• It gives a solution to the mobile asymmetry problem due to difference in wired and wireless bandwidths.
• It avoids the location management issue by defining a storage point for MH location details.
• It reduces the query response time and avoids data loss by caching the host specific data and buffering the user requests temporarily, when MH is disconnected.
• It ensures the optimal utilization of wireless bandwidth.

The distributed and decentralized nature of the proxy objects enhances the resource sharing in the mobile grid. The proxy objects represent the active hosts and the wired network represents them. The mobility of the MHs does not degrade the provision of the services because of the full-state maintenance of hosts. The proxy objects are fully dynamic, secure, and autonomous i.e., the operational capabilities of POs even after the MHs are not reachable and disconnected. The PO may be related to MHs of three cases:

1. MH outside the local but inside the same MC.
2. MH outside any MC.
3. Foreign MH.

The dynamicity of the mobile grid architecture is enabled by permitting the alteration of new services and extending the wired network. This aids the adaptability of the services based on the necessities of the mobile grid users. PO possesses the properties and characteristics of a host. The characteristics of a PO are represented as attributes, sub-objects, and methods. The attributes of a PO comprise the computational capability, bandwidth, and memory consumptions (Mohamed 2011). The sub-objects and the methods contain the services and other data resources offered by the host. The integration of the agreement and security policy by PO for all services specify the operation mode of services.
SOM operates in wireless connectivity mode, when the surrogate objects are transferred to the wired network from the MHs. The state information is used to maintain the services of PO, even when the MH is disconnected. The data and services results can be delivered once the MH reconnects. This programming principle (or) paradigm enhances security by integrating authority schemes within the POs to operate the objects and services. MHs are also information service providers (Mohamed, et al. 2005). The energy and bandwidth consumption can be reduced by contacting the PO for information rather than contacting the MHs. Congestion can also be avoided by replicating the POs, thereby enhancing the system scalability.

4.4.2 Management of Services

The mobile grid comprises services as the basic blocks for large-scale computations (Mohamed 2011). The main issues in the mobile grid are service detection and cooperation, i.e. to synchronize a group of services executing on heterogeneous resources under various controls in order to solve a unified problem. The computational resources and their functionalities are defined as composable services. Mobile grid applications can be constructed by creating these services at a higher abstraction level.

Service composition is an important function which permits various autonomous services to be formed into a new service with a unique functionality and permits development of independent and modular services. Service composition is dependent on the location of the users and focuses on management of the integrated services for task completion (Mohamed 2011). Mobile grid formed by MHs possesses new obstacles for service composition. The service results are not displayed, when a user requests a service from a host, but moves to another host with higher functionality.
The result becomes invalid, when the user moves across various access points for location dependent services. The infrastructure for service composition of the mobile grid is given in Figure 4.2. The Distributed Shared Object Model (DSOM) structures the principles through which services are defined and used for resource sharing in the mobile grid. The services are virtualized as software functional components which involve abstraction of pre-defined functionalities.

The services are announced and discovered using inspection and directories. After service detection, an appealing element can connect to the selected mobile device and begin the communication with its explicit functions via platform autonomous protocols. A vast conglomeration of composable software and hardware resources can be obtained by integrating the services with virtualized hardware resources.

Figure 4.2 Layered reference architecture for services management in mobile grid
Service detection is an important step, as the system must find a service before it could utilize it. The mobile grid should be capable of supporting variable service announcement and discovery. The service detection is based on the characteristics, location, and functionality of the services (Mohamed, et al. 2005). Runtime connection enhances reliability and load balancing of the system. It also supports a wide range of application compatibility in terms of network configurations and platforms. When the task is completed, the services are returned back to the conglomeration for assignment to other users.

4.4.3 Service Characterization

The service providers need to issue the constraints and features of the services. These descriptions are sent to CH for registration. A discovery protocol is required to map the services to the application queries. This service model has the following advantages:

- No particular schema is needed for the system to work in a heterogeneous environment.
- Freedom to express conditions on the services, the model is ready to serve.

The characterization of services involves attributes and conditions (Mohamed 2011). The attributes of the service model comprise the resource characteristics such as CPU usage, location, and free memory. Dynamic characteristics of the services can be obtained by DSOMs running on the data resource. Conditions include the limitation expression given by the service provider for service allocation.

The services are maintained by an exchange service repository. The logical mobile clusters contain a service directory in the CH, which is the
center for services registry. An MH, that desires to give its services, avails the information in the service directory.

4.4.4 Service Detection

The services must be identified in order to coordinate with other mobile users and utilize them. In the present methods, for service detection, the interoperability constrain exists and which leads to expressive power loss during component characterization. The DSOM gives a clear differentiation between global service management and local service management. Global service management groups the data from the cells into large mobile clusters, thus availing the data available to the users (Mohamed, et al. 2005).

It also comprises global tracking services and search methods for clients. Local service management controls the services within a logical mobile cluster or a domain like cell. The directory service regards the location of two processes in a similar cell also as same. Local service management occurs in CH of each logical mobile cluster. A service is affiliated with the Proxy Object Identifier (POID) depending on the client’s needs. The basic requirements for this management are execution monitor, information database, and scheduler.

- The information related to the data resources and services are stored in the information database.
- The scheduler computes an association of objects with services based on the information database.
- This connection is used to contact the objects and affirm the schedule.
- All these processes are managed by the execution monitor.
4.4.5 Service Composition

The combination of high and low performance devices interconnected to each other makes the integration and execution of heterogeneous tasks, a tough task (Mohamed 2011). Service composition is used to construct complex services from basic services, thus resulting flexibility of new service creations. It can be shown as a variable integration of multiple services in the mobile grid in response to a client request.

Service composition is an important function that allows various autonomous services to be formed into a new service with a unique functionality. It allows the development of independent and modular services. The critical issue in service composition involves the management of disconnections during the execution of services and formulating context dependent service execution. The execution of service composition is performed based on optimal computing resources under conditions like data resource reliability and execution cost. A user initializing a service composite request can also indulge in another service composition. The P2P overlay within the CHs ensures the fault-tolerance of the system.

Here, MDP is used to know the status about the nodes, previous and current transmissions among the nodes. Hence, the disconnections among the nodes are highly reduced through the MDP pipe status. The pipe status is periodically updated based on the MDP packet mode. It includes the pipe id, object id, communication object id and pipe status. MDP uses the six set of control messages:

1. EstablishMDP(): It provides the new connection request
2. InitiateMDP(): It confirms the new connection request to initiate the process
3. GetMDP(): It retrieves the connection among the nodes
4. PutMDP(): It transfers the data among the nodes
5. EndMDP(): It notifies that the current data transformation is completed and the node is set to sleep state
6. RemoveMDP(): After the successful completion the corresponding link is deleted.

4.5 DISTRIBUTION AND SHARING OF OBJECTS

A DSO makes use of multiple interfaces, each comprising a group of methods. The proxy objects act as local objects in a DSOM. The mobile objects in this model are passive, while the user threads use these mobile objects by executing the code for their methods. A single mobile object can be accessed by multiple processes simultaneously. The modifications to a mobile object’s state by a process are visible by the other processes. The distributive nature of the mobile shared objects enables the active copies of a mobile object’s state to be stored simultaneously on multiple machines. But, the communication protocols, distribution/migration of states, and replication methodologies are embedded in the interface.

A significant difference between DSOMs and remote objects is that there is no priori differentiation between users and servers. The processes that communicate via method invocation combine in its object implementation.

4.5.1 Merits of DSOMs

Some of the advantages of DSOMs are:

- A distributed proxy object enables well-structured interfaces to its applications.
- The user is separated from the communication and replication processes.
- Complete encapsulation of communication and persistence are
provided in a distributed proxy object.

- This implies that the implementation of a distributed system is not bounded to a small group of consistency algorithms or communication protocols.
- Ability to load object implementations at runtime.
- A process consists of a local implementation of the distributed proxy object’s interface in its own address space.
- A DSOM is visualized only as a local object in the perspective of a process.

4.5.2 Architecture of DSOM

The distributed proxy objects are a group of local proxy objects that communicate and furnish the user of the object with the delusion of the shared state (Ørbekk 2012). This characteristic is advanced over a remote object model, since it is not bound to a small set of predefined communication patterns. The example architecture of a DSOM is shown in Figure 4.3.

The distributed proxy object is used in four address spaces, where each address space comprises a proxy object. The group of proxy objects forms the distributed proxy object. The proxy objects utilize the communication services of a network to operate on the distributed proxy object and maintain the distributed proxy object.
The proxy object implementation is assorted from an application via an explicit interface table comprising method pointers (Ørbekk 2012). The interface table is triggered, when the process connects to the object. The contents of the interface table are variable over time. This enables the dynamic adaption of the distributed object’s local implementation. The adaption process does not affect the interface of the application that triggers its methods.

The implementation of a distributed proxy object can utilize random communication patterns, while communicating with local objects. The communication process can also integrate data placement and replication. This
scheme is applicable for efficient implementations of various communication paradigms. There are no limitations on the predefined operations, since the interfaces are completely user-defined.

4.5.3 Transparent Communication

The developer of the distributed proxy object requires being isolated from the data placement and replication (Ørbekk 2012). So, a standard hierarchy is developed for the implementation of a distributed proxy object. The hierarchy of a local object is given in Figure 4.4.

The distributed proxy object’s developer is segregated from communication, consistency management, and replication by using a communication object and a replication object. The object developer implements the semantics object which enables the actual functionality of the distributed proxy object, while the communication and replication objects are chosen from a library. A control object is used to manage the interactions between the replication objects and the semantic objects as a consequence of method invocation by an application. The control object can be generated automatically, which has been produced based on the semantic object.
Figure 4.4 Hierarchy of a proxy object

The proxy object is capable of exporting methods that can execute on internal state. The access to the control object is synchronized with that of the distributed proxy object by serializing permissions to the semantics object. This prohibits the race conditions by triggering the replication object to stabilize the state of the distributed proxy object. The interface exported by the control object is similar to that of the semantic object.

The implementation of a method invocation by the control object is performed via three consecutive steps:

1. The `start` method controls the execution of global state functions.
   a. During remote execution, the control object passes the mobilized arguments of the method invocation to the replication object.
b. The execution of the replication object occurs according to a specific replication protocol and returns the mobilized results to the control object according to the remote method invocation.

2. During local execution, the control object triggers the related method on the semantics object.
   a. During active replication with a local copy, the control object offers the replication object with the mobilized arguments of the method invocation.
   b. Then, the replication object executes the protocol to transfer the arguments to all replicas. This enables the synchronization with the other replicas.

3. Finally, the control object triggers the corresponding method on the semantics object.

The control object triggers the finish method on the replication object, which yields the replication object an opportunity to update the remote replicas. Two extensions are needed in this model to enhance its practical performance.

1. The control object and replication object need to distinguish various operation types.
   a. It is also required to differentiate operations that alter only a part of the global state.
   b. It may occur in the case of nested or segmented objects.

2. Few extensions are required on specific criteria to handle synchronization, since the semantic operations are serialized and not permitted to block for a long time.

The operations can be secured by providing blocks on a conditional basis. The status information is returned to the control object after the possible
alterations are made. The control object will delay the execution of the operation until the next state alteration.

This model of shared state comprising operations results in passive objects, where the activity is given by the threads executing in the processes. For seamless integration of communication in this model, *pop-up threads* are instantiated with the incoming messages (Ørbekk 2012). Hence, the communication object will open a new thread to handle an incoming message. The communication object inside the new thread triggers a method on the replication object’s callback interface. The replication object requests the control object’s callback interface with the mobilized arguments of the request.

### 4.6 SUMMARY

The performance of the DSOM scheme depends on the mobility of the nodes, tolerance to network traffic, and node density. The number of mobile nodes entering a group of cells should be managed properly. Parallel computing on mobile grid depends upon several key issues such as mobility of nodes, fault-tolerance, connectivity of mobile nodes, uneven nodal distribution, transmission time, heterogeneity of nodal performances and uneven load in the network. The proxy objects do not support flexible wireless bandwidth. This can be overcome in the DSOM model by differentiating the type of connectivity in a cell of the mobile cluster. The handoff issue is not properly addressed by the proxy objects, but it is efficiently handled by the topology scheme introduced in the DSOM model. The proposed work shares the file based on the shared object model. It helps to reduce the storage space required for data transmission.