CHAPTER 3
INTERLINKED MESH GRID ARCHITECTURE
WITH OUTER CONNECTION

3.1 RECENT DEVELOPMENTS IN ARCHITECTURAL
DESIGN OF GRID COMPUTING

Grid computing has evolved into an efficient global computing infrastructure by introducing optical network technology to support the advanced data-intensive distributed applications. Scheduling such data-intensive applications includes assigning tasks on computational resources, routing light paths, and assigning wavelength channels for data communication. The scheduling problem is NP-hard in a traditional grid system, and is very complicated in optical grids due to the character of optical networks.

The evolution of grid system and the architecture of grid system are shown in Figure 3.1. They are based on requirements of the grid which grows rapidly. At present, the grid computing has become a part of major engineering organizations.

Load balancing is an important and complex problem in computational grids. A computational grid differs from traditional high-performance computing systems in the heterogeneity of the computing nodes, as well as the communication links that connects the different nodes together. This complexity needs to be captured by algorithms, which can be easily implemented and used to solve a wide range of load balancing scenarios.
Figure 3.1  Evolution of grid system

Figure 3.2  Layered internet protocol architecture of grid system
An efficient resource discovery mechanism is one of the fundamental requirements for grid computing systems, as it aids in resource management and scheduling of applications. Resource discovery activity involves searching for the appropriate resource types that match the user's application requirements. Various kinds of solutions to grid resource discovery have been suggested, including centralized and hierarchical information server approaches. However, both these approaches have serious limitations with respect to scalability, fault tolerance and network congestion.

To overcome these limitations, indexing resource information using a decentralized, for example, the P2P network model, has been actively proposed in the past few years.

Figure 3.2 describes the architecture of grid computing which is termed Internet Protocol Architecture for grids. This proposal has many layers, which are broadly described as application layer, transport layer and link layer. The application layer is further classified into collective and resource layers based on the co-ordinating resources. This section offers a brief introduction to various international consortium and various global grid systems developed in recent years.

In scheduling a large number of user jobs for parallel execution on an open-resource grid system, the jobs are subjected to system failures or the delays caused by infected hardware, software vulnerability, and distrusted security policy. The relative performance is measured by total job makespan, grid resource utilization, job failure rate, slowdown ratio, replication overhead, etc.
The grid consortium consists of the following and focuses on grid computing and its engineering applications:

- Asia Pacific Grid
- Australian Grid Forum
- Content Alliance: About Content Peering
- Distributed.Net
- Egrid: European Grid Computing Initiative
- Eurotools SIG On Meta Computing
- Global Grid Forum
- Global Grid Forum (GGF)
- Grid Computing Info Centre
- Grid Forum Korea
- IEEE Task Force on Cluster Computing
- New Productivity Initiative (NPI)
- Peer-to-Peer (P2P) Working Group
- SETI @home
- Distributed Coalition

And the following are the few grid applications in the recent worldwide implementation:

- Access Grid
- APEC Cooperation For Earthquake Simulation
- Australian Computational Earth Systems Simulator
- Australian Virtual Observatory
- Cellular Micro-Physiology
- Data GRID—WP9: Earth Observation Science Application
- Distributed Proofreaders
- DREAM Project: Evolutionary Computing And Agents Applications
- Earth System Grid
- Fusion Collaboratory
- Geodise: Aerospace Design Optimisation
- Globus Applications
- Grid Search & Categorization Engine (GRACE)
- HEP Grid: High Energy Physics And The Grid Network
- Italian Grid (GRID.IT) Applications
- Japanese Bio Grid
- NEES Grid: Earthquake Engineering Virtual Collaboratory
- Knowledge Grid
- Molecular Modeling For Drug Design
- NC Bio Grid
- Neuro Science—Brain Activity Analysis
- NLANR Distributed Applications
- Open Mol Grid
- Particle Physics Data Grid
- International Grid (Igrid)
3.2 INTRODUCTION TO GRID ARCHITECTURE

Figure 3.3 shows the architectural designs for connecting the local grid computing systems. Figure 3.4 shows the architectural design for connecting the intra grid computing systems.

The intragrid and intergrid architectures are proposed to connect two various grid systems that are located at two or more different places. Figure 3.5 shows the architectural design for connecting the intergrid computing systems.

Figure 3.3 Local grid architecture
Figure 3.4 Intragrid architecture

Figure 3.5 Intergrid architecture

Figure 3.6 shows a formal grid based application environment using ethernet access in which the grids are connected with Multi Protocol Label Switch and Internet Protocol. Mining in such massive data set is a complex problem and needs high computation and parallel computing architecture for effective computing, as explained by Faro et al. (2011). In general, the company grid systems may have independent access via Virtual Private Network.
Over the past decade, the grid has emerged as an effective platform to tackle various large-scale problems, especially in science and engineering. One primary issue associated with the efficient and effective utilization of heterogeneous resources in a grid is scheduling. Grid scheduling involves a number of challenging issues, mainly due to the dynamic nature of the grid. There are only a handful of scheduling schemes for grid environments that realistically deal with this dynamic nature that have been proposed in the literature.

Distributed application (e.g., grid-enabled application) performance can be improved by complementing the computational resource information advertised to the clients with network information (e.g., topology and link capacity). In this way, clients may choose the jointly optimal resources.

A sustainable market-like computational grid has two characteristics. It must allow resource providers and resource consumers to make autonomous scheduling decisions, and both providers and consumers must have sufficient incentives to stay and play in the market.
Recently, utility grids have emerged as a new model of service provisioning in heterogeneous distributed systems. In this model, users negotiate with service providers on their required QoS and on the corresponding price to reach a Service Level Agreement. One of the most challenging problems in utility grids is workflow scheduling, i.e., the problem of satisfying the QoS of the users as well as minimizing the cost of workflow execution.

QoS-based workflow scheduling algorithm is based on a novel concept called partial critical paths (Dabhi and Prajapati, 2008), to minimize the cost of workflow execution while meeting a user-defined deadline. The partial critical paths algorithm has two phases. In the deadline distribution phase, it recursively assigns sub-deadlines to the tasks on the partial critical paths ending at previously assigned tasks. And in the planning phase, it assigns the cheapest service to each task while meeting its sub-deadline.

3.3 OVERVIEW OF THE PROPOSED HYBRID MODEL

Smart grid applications impose challenging requirements of security and reliability on the N-way communication infrastructure being designed to support multiple grid applications. These challenges stem from the increasing incorporation of distributed renewable energy sources onto the grid, the rising deployment of electric vehicles, and active consumer participation in power grid operations, all of which communicate with the utility control centre with varying degrees of priority and reliability.

The key functional blocks of the proposed hybrid scheduling model are: (1) an interlinked mesh grid architecture with outer connection, (2) a grid control system which efficiently organizes grid resources using splitter, (3) a geographic hash forwarding algorithm that operates over the job scheduling called Hybrid Hash Table (HHT) and (4) a Parameter Optimized ACO (POACO) for task scheduling.
The proposed interlinked mesh grid architecture with outer connection is explained in section 3.4. An overview of other functional blocks is explained in the following sub-sections.

3.3.1 Overview of Grid Control System (GCS)

Grids are interconnected for optimized data transmission through a centralized grid control system. The proposed GCS is locally distributive and globally centralized. In the grid environment, GCS is elected through optimal algorithm, which provides best results in both square and rectangular grid models. Each grid has a local GCS which provides resource management for a particular grid.

The jobs received by local GCS are directed by the Global GCS. If the grid system is heavily loaded, the scheduling system has to study the load balancing, and increase the system throughput and resource utilization under restricted conditions. Under these conditions the splitter in the GCS decides whether to perform task scheduling or job scheduling based on the available jobs in the queue.

The proposed grid model is virtualized on user perspective. Therefore, each user can receive and operate the entire resources available in the local grids connected with local GCS and the resources available outside local GCS.

3.3.2 Overview of HHT

The hash table is a fair and feasible way of resource optimization in the last few years. Traditional approaches like linear hashing are the simplest open addressing schemes which use a hash function and test positions. The main problem with linear hashing is that it degenerates into a sequential
search and wastes space due to empty positions in the hash table. Other approaches such as double hashing, quadratic hashing, cuckoo and hopscotch hashing have been proposed for resource optimization. But most of them have a problem of storage allocation and common problems of waste of memory space, and resource discovery.

Resource discovery is an important step in grid resource management. A hybrid hash table which contains all resource groups on a monitor node. A grid scheduler determines the set of resources to match the application execution requirements. The HHT maintains the neighbour list, which is a checking procedure to be performed periodically. For each period, every host node tests the network speed with a number of randomly selected neighbours. The number of selected neighbours is usually relatively small compared with the total number of neighbours.

The proposed method considers lookup time for successful and unsuccessful searches in two scenarios: (i) the Minimal Perfect Hashing Function (MPHF) description fits in the CPU cache and (ii) the MPHF description does not fit entirely in the CPU cache.

The scheduling is processed in two levels: at the job level the HHT is proposed and at the task level the parameter optimized ACO is proposed. Consider a HHT consisting of ‘N’ nodes. Grid IDs are assigned from an identifier space with a distance metric. Each node ‘s’ maintains a task table T_s of entries (u, GID_u), where ‘u’ is a neighbor and GID_u is its grid ID. Hence, ‘s’ request tasks to ‘u’, a task to a destination grid ‘d’ goes sequentially to grids whose IDs are progressively closer to ‘d’ according to the distance metric.
If v ∈ T_s then ‘s’ can request resources to ‘v’ forming the one-hop path s → v. Therefore, assigning a grid from ‘s’ to ‘d’ takes several hops forming a multi-hop path s→d. HHT scheduling is divided into global and local parts. In global scheduling, a request is delivered close to the destination. In local scheduling, the destination is at a nearby node. HHT scheduling is either iterative or recursive. In iterative scheduling, each grid returns the next-hop grid ‘v’ to the querying node. The recursive scheduling contacts ‘v’ to get iteratively closer to the destination.

Basically, there is no grid scheduling process on a physical node. When a new job arrives, one or more grids are assigned to run it. Unlike parallel computing and cluster computing, grid computing does not schedule all processing units for the available job. Generally, the executing system of the grid will handle the job scheduling after a job is submitted to the node. If the grid system is heavily loaded, the scheduling system has to study the load balancing, and increase the system throughput and resource utilization under restricted conditions. Under these circumstances, the splitter acts and proceeds, which is explained in the fourth chapter in detail.

3.3.3 Overview of Parameter Optimized Ant Colony Optimization (POACO)

The objective of the scheduling algorithm is to minimize the makespan of task. The use of ACO algorithms to obtain good solutions for combinatorial optimization problems has become very popular in recent years. ACO algorithms have the ability to integrate knowledge about the problem into the construction of a new solution. The proposed POACO is an improved version of ACO. The POACO applies the state transition rule for calculating the probability of a given task, in which two numeric constants are used as traffic impact and as shortest path impact. Optimization of these two parameters improves the performance of ACO.
The proposed POACO was implemented on a variety of test cases and performance was recorded. The performance of POACO was verified on three parameters namely, execution time, maximum execution time and idle time. The performance of ACO in terms of execution time, maximum execution time and idle time of each grid was optimal in test case 2. The performance of test case 2 was comparatively better than other three test cases. Therefore, the proposed work concludes that the optimal values for two parameters (k and h values) were 1 and 0, respectively. This value was applied to the proposed hybrid scheduling system and observed the performance improvement.

3.4 PROPOSED INTERLINKED OUTER CONNECTED MESH GRID ARCHITECTURE

Interlinked mesh grid architecture is designed in the proposed work, which is shown in Figure 3.8, and it reduces hop count than existing systems, which reduces the number of communications to 50%.

Later, this interlinked mesh grid architecture is improved using outer connection, which is shown in Figure 3.9, which is a improved version of Figure 3.8, which has outer connection, therefore, the number of IMP / hop count are further reduced at terminal nodes.

In almost all grid models, the proposed interlinked mesh grid architecture with outer connection offers only one communication for queries transmission at the terminal nodes.
The hop count / IMP for almost all grid architecture is 1. In a 4 x 4 grid, the existing grid model transmits query through 3 IMP in the worst case and at least 2 IMP in the best case. The proposal shown in Figure 3.8, transmits through 2 IMP on any case using interlinked grid mesh architecture. Whereas this is further reduced as 1 IMP in the terminal nodes as shown in Figure 3.9 using the outer connection. This is described in details in the section 3.6.
3.5 PSEUDO CODE FOR COMPUTING THE NUMBER OF CONNECTION IN MESH GRID ARCHITECTURE

Let ‘b’ be the number of base in the mesh grid architecture where base means the dimensions of square matrix.

Therefore number of nodes in the grid is given in the equation is 

\[ N = b^2 \]

The computation of interconnections is as follows

Case 1:

If \( b = 2 \)

\[ C_2 = b^2 \times (b^2 - 1) \div 2 \]

Where \( C_2 \) is Interconnection when base (b) is 2

Case 2:

If \( b > 2 \)

Step1: Calculate the number of 2*2 square matrix (S2)
Step2: Compute the total interconnection (T) = S2*C2
Step3: Find the number of shared interconnections (SI)

\[ SI = 2 \times \text{number of inter layer connections} \]

Step 4: \( C_b = \text{Total interconnection (T)} - \text{Shared interconnection (SI)} \)

Therefore the total number of connections in the interlinked mesh grid architecture with outer connection is,

\[ \text{No. of interconnections in interlinked mesh grid architecture} + \text{Number of Outer connections (i.e., 4).} \]
3.6 PERFORMANCE ANALYSIS

In this section, the performance of proposed interlinked mesh grid architecture with outer connection is compared with existing grid architecture and interlinked mesh grid architecture. Figure 3.9 shown a sample of existing grid architecture. The performance of the same is given as hop count in the Table 3.1.

Figure 3.9 Sample of existing grid architecture
The performance in terms of maximum hop count in the existing mesh grid architecture is shown in Table 3.1. For example, no. of communications path between a node in the Grid1 (G1) to a node in the Grid9 (G9) is computed as (hop count) four. Latency, processing time, and response time are increased when hop count is increases. Therefore, the objective of the design of optimal grid architecture is to reduce hop count. Hence, interlinked mesh grid architecture is proposed. A sample architecture of interlinked mesh grid is shown in Figure 3.10.

<table>
<thead>
<tr>
<th>Nodes</th>
<th>G1</th>
<th>G2</th>
<th>G3</th>
<th>G4</th>
<th>G5</th>
<th>G6</th>
<th>G7</th>
<th>G8</th>
<th>G9</th>
</tr>
</thead>
<tbody>
<tr>
<td>G1</td>
<td>----</td>
<td>01</td>
<td>02</td>
<td>01</td>
<td>02</td>
<td>03</td>
<td>02</td>
<td>03</td>
<td>04</td>
</tr>
<tr>
<td>G2</td>
<td>01</td>
<td>----</td>
<td>01</td>
<td>02</td>
<td>01</td>
<td>02</td>
<td>02</td>
<td>02</td>
<td>02</td>
</tr>
<tr>
<td>G3</td>
<td>02</td>
<td>01</td>
<td>----</td>
<td>03</td>
<td>02</td>
<td>01</td>
<td>04</td>
<td>03</td>
<td>02</td>
</tr>
<tr>
<td>G4</td>
<td>01</td>
<td>02</td>
<td>03</td>
<td>----</td>
<td>01</td>
<td>02</td>
<td>01</td>
<td>02</td>
<td>03</td>
</tr>
<tr>
<td>G5</td>
<td>02</td>
<td>01</td>
<td>02</td>
<td>01</td>
<td>----</td>
<td>01</td>
<td>02</td>
<td>01</td>
<td>02</td>
</tr>
<tr>
<td>G6</td>
<td>03</td>
<td>02</td>
<td>01</td>
<td>02</td>
<td>01</td>
<td>----</td>
<td>03</td>
<td>02</td>
<td>01</td>
</tr>
<tr>
<td>G7</td>
<td>02</td>
<td>03</td>
<td>04</td>
<td>01</td>
<td>02</td>
<td>03</td>
<td>----</td>
<td>01</td>
<td>02</td>
</tr>
<tr>
<td>G8</td>
<td>03</td>
<td>02</td>
<td>03</td>
<td>02</td>
<td>01</td>
<td>02</td>
<td>01</td>
<td>----</td>
<td>01</td>
</tr>
<tr>
<td>G9</td>
<td>04</td>
<td>03</td>
<td>02</td>
<td>03</td>
<td>02</td>
<td>01</td>
<td>02</td>
<td>01</td>
<td>---</td>
</tr>
</tbody>
</table>
Figure 3.10 Sample of interlinked mesh grid architecture

Table 3.2 Hop count of interlinked mesh grid architecture

<table>
<thead>
<tr>
<th>Nodes</th>
<th>G1</th>
<th>G2</th>
<th>G3</th>
<th>G4</th>
<th>G5</th>
<th>G6</th>
<th>G7</th>
<th>G8</th>
<th>G9</th>
</tr>
</thead>
<tbody>
<tr>
<td>G1</td>
<td>----</td>
<td>01</td>
<td>02</td>
<td>01</td>
<td>01</td>
<td>02</td>
<td>02</td>
<td>02</td>
<td>02</td>
</tr>
<tr>
<td>G2</td>
<td>01</td>
<td>----</td>
<td>01</td>
<td>01</td>
<td>01</td>
<td>01</td>
<td>02</td>
<td>02</td>
<td>02</td>
</tr>
<tr>
<td>G3</td>
<td>02</td>
<td>01</td>
<td>----</td>
<td>02</td>
<td>01</td>
<td>01</td>
<td>02</td>
<td>02</td>
<td>02</td>
</tr>
<tr>
<td>G4</td>
<td>01</td>
<td>01</td>
<td>02</td>
<td>----</td>
<td>01</td>
<td>02</td>
<td>01</td>
<td>01</td>
<td>02</td>
</tr>
<tr>
<td>G5</td>
<td>01</td>
<td>01</td>
<td>01</td>
<td>01</td>
<td>----</td>
<td>01</td>
<td>01</td>
<td>01</td>
<td>01</td>
</tr>
<tr>
<td>G6</td>
<td>02</td>
<td>01</td>
<td>01</td>
<td>02</td>
<td>01</td>
<td>----</td>
<td>02</td>
<td>01</td>
<td>01</td>
</tr>
<tr>
<td>G7</td>
<td>02</td>
<td>02</td>
<td>02</td>
<td>01</td>
<td>01</td>
<td>02</td>
<td>----</td>
<td>01</td>
<td>02</td>
</tr>
<tr>
<td>G8</td>
<td>02</td>
<td>02</td>
<td>02</td>
<td>01</td>
<td>01</td>
<td>01</td>
<td>01</td>
<td>----</td>
<td>01</td>
</tr>
<tr>
<td>G9</td>
<td>02</td>
<td>02</td>
<td>02</td>
<td>02</td>
<td>01</td>
<td>01</td>
<td>02</td>
<td>01</td>
<td>----</td>
</tr>
</tbody>
</table>

Table 3.2 shows hop count in interlinked mesh grid architecture. No. of communications path between a node in the Grid1 (G1) to a node in the Grid9 (G9) is reduced to two which is 50% less than existing mesh grid architecture. Hence, it is concluded that interlinked mesh grid architecture is optimal than existing mesh grid architecture. In addition to this achievement, the research on optimal design is extended for reducing hop count. Therefore
an improved version of interlinked mesh grid architecture, which is termed interlinked mesh grid architecture with outer connection. A sample architecture of interlinked mesh grid with outer connection is shown in Figure 3.11, and subsequent hop count of the same is shown in Table 3.3.

Table 3.3 shows the performance of interlinked mesh grid architecture with outer connection. In which, the hop count between terminal node is achieved as one. This is effective than existing mesh grid architecture and interlinked mesh grid architecture.
Table 3.3 Hop count of interlinked mesh grid architecture with outer connection

<table>
<thead>
<tr>
<th>Nodes</th>
<th>G1</th>
<th>G2</th>
<th>G3</th>
<th>G4</th>
<th>G5</th>
<th>G6</th>
<th>G7</th>
<th>G8</th>
<th>G9</th>
</tr>
</thead>
<tbody>
<tr>
<td>G1</td>
<td>----</td>
<td>01</td>
<td>01</td>
<td>01</td>
<td>02</td>
<td>01</td>
<td>02</td>
<td>02</td>
<td></td>
</tr>
<tr>
<td>G2</td>
<td>01</td>
<td>----</td>
<td>01</td>
<td>02</td>
<td>01</td>
<td>02</td>
<td>02</td>
<td></td>
<td></td>
</tr>
<tr>
<td>G3</td>
<td>01</td>
<td>01</td>
<td>----</td>
<td>02</td>
<td>01</td>
<td>02</td>
<td>02</td>
<td>01</td>
<td></td>
</tr>
<tr>
<td>G4</td>
<td>01</td>
<td>01</td>
<td>02</td>
<td>----</td>
<td>01</td>
<td>02</td>
<td>01</td>
<td>01</td>
<td>02</td>
</tr>
<tr>
<td>G5</td>
<td>01</td>
<td>01</td>
<td>01</td>
<td>----</td>
<td>----</td>
<td>01</td>
<td>01</td>
<td>01</td>
<td>01</td>
</tr>
<tr>
<td>G6</td>
<td>02</td>
<td>01</td>
<td>01</td>
<td>02</td>
<td>01</td>
<td>----</td>
<td>02</td>
<td>01</td>
<td>01</td>
</tr>
<tr>
<td>G7</td>
<td>01</td>
<td>02</td>
<td>02</td>
<td>01</td>
<td>01</td>
<td>02</td>
<td>----</td>
<td>01</td>
<td>01</td>
</tr>
<tr>
<td>G8</td>
<td>02</td>
<td>02</td>
<td>02</td>
<td>01</td>
<td>01</td>
<td>01</td>
<td>----</td>
<td>01</td>
<td></td>
</tr>
<tr>
<td>G9</td>
<td>02</td>
<td>02</td>
<td>01</td>
<td>02</td>
<td>01</td>
<td>01</td>
<td>01</td>
<td>----</td>
<td></td>
</tr>
</tbody>
</table>