International Journal of Computer Applications Technology and Research
Volume 4– Issue 4, 312 - 317, 2015, ISSN:- 2319–8656

# Virtual Organization of Grid – Pipeline Virtual Organization (PVO) Approach

J.Prema
Department of CSE
SRM University
Ramapuram
Chennai, India

Dojohn Loyd B.
Department of CSE
SRM University
Ramapuram
Chennai, India

**Abstract:** Grid computing is a modularized way of structuring network resources so as to share the information and resources, to perform heavily intense problems. Grid computing is a part of distributed processing which allows the distribution of the problem over multiple computational resources. The formation of grid decides the computation cost, the performance metrics of the operation to be carried out. The grid computing favours the formation of an adhoc structure formed at the time of request (it does not hold an infrastructure) that is termed as virtual organization. This paper presents dynamic source routing for modeling virtual organization.

**Keywords**: Grid computing, grid layers, virtual organization, MSVOF Mechanism, minimum path algorithm, pipelining VO formation

## 1. INTRODUCTION

In the modern world, scientific problem has grown beyond proportion; hence the computation could take time anywhere between hours and years. This is due to the presence of huge data set or the complexity of the computation. The best way to deal with this is the distribution of problem's data set over multiple computational units which led to the concept of GRID computing. Grid is a type of parallel and distributed system that enables sharing, selection and aggregation of geographically distributed autonomous resources dynamically based on their availability, capability and cost.

## 2. ARCHITECTURE OF GRID

Grid computing involves a five layered architecture providing protocols and services at different levels of layers.

## 2.1 Layers of Grid

### *2.1.1 Fabric Layer:*

Fabric layer takes the bottom of the layered architecture that provides shareable resources such as network bandwidth, CPU, time, memory, etc. Operating System, queuing system and processing kernel also forms the part of this layer.

### *2.1.2 Connectivity Layer*

The protocols related to the communication and authentication becomes the part of this layer.

### *2.1.3 Resource Layer*

This layer specifies the protocols for operating with the shared resources. These protocols are concerned about the allocation and monitoring of resources.

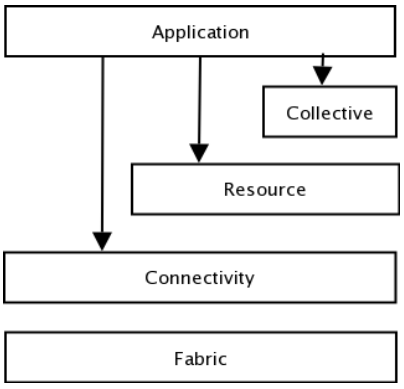


**Figure : Architecture Of Grid**

### *2.1.4 Collective Layer*

This layer holds general purpose utilities like directory services, diagnostic services, data replication services, etc.

### *2.1.5 Application Layer*

This layer is the placeholder where the user's applications are deployed.

## 2.2 Types of Grid

Grids are categorized into five types based on their utility.

### *2.2.1 Computational Grids:*

These are the grids capable of providing secure access to computational resources for complex computation which would require high computing machines.

### *2.2.2 Utility Grids:*

These are the grids that promote the sharing of CPU cycles, software and peripherals like sensors, etc.

### *2.2.3 Data Grids:*

These are the grids that offers database storage related functionalities.

### *2.2.4 Collaborative Grids:*

These are the grids that lead to formation of social web forums.

## 3. VIRTUAL ORGANIZATION IN GRID

A virtual organization is a collection of geographically distributed functionally and diverse entities that are linked to each other by means of electronic communication. VO are always formed suddenly in an adhoc fashion, they perform task assigned and then dismantle them immediately. The mechanism that decides the formation of virtual organization pattern contributes to the revenue and the cost of the computation.

Dynamic grid Virtual organization is collections of workspaces with the computing systems which gets tasks in a sequential manner and gets a separate method for each computing system and get that job completed and gives the output.

## 3.1 Necessity for VO Formation Algorithm:

Virtual Organization is formed with electronic communication. But when it is grouped together formation is more important to organize it in a sequential way. So that the task can be performed in a good manner and then the cost also will be decreased so that we are using formation algorithms for VO. We have used some previous algorithms like Merge and split VO formation (MSVOF) and Random Select Virtual organization Formation (RSVOF), Same Size Virtual organization formation (SSVOF), Grand Coalition VO formation (GVOF). In this paper we have introduced Pipelining Virtual Organization Formation (PVOF).

## 4. MSVOF MECHANISM

This mechanism is to find the minimum cost between the nodes and splitting it into some partitions and execute their program. This algorithm comes after some analysis on Random Select VOF and Grand Coalition VOF.

*VO ={{Vo1},... {Vom}}*
*Map program P on each Si € VO*
***repeat***
*stop ←T rue*
***for all*** *Si, Sj € VO, i≠ j* ***do***
*visited[Si][Sj] ← False*
***end for***
*{Merge process starts:}*
***repeat***
*flag ←T rue*
*Randomly select Si,Sj € VO for which*
*Visited [Si][Sj]=False; i ≠ j*
visited[Si][Sj]←T rue
*B&B-MIN-COST-ASSIGN(Si U Sj)*
*{Map program T on Si [ Sj}*
***if*** *(Si U Sj) €{Si,Sj}* ***then***
*Si ←Si U Sj {merge Si and Sj}*
*Sj←ϕ {Sj is removed from VO}*
***for all*** *Sk €VO, k ≠ i* ***do***
*visited[Si][Sk] ←False*
***end for***
***end if***
***for all*** *Si,Sj€VO, i ≠ j* ***do***
***if*** *not visited [Si][Sj]* ***then***
*flag ←False*
***end if***
***end for***
***until*** *(|VO|=1) or (flag ←T rue)*
*{Split process starts:}*
***for all*** *Si € VO where |Si| > 1* ***do***
***for all*** *partitions {Sj,Sk} of Si,*
*where Si=SjU Sk, Sj ∩ Sk=ϕ* ***do***
*B&B-MIN-COST-ASSIGN(Sj)*
*{Map program P on Sj}*
*B&B-MIN-COST-ASSIGN(Sk)*

*{Map program P on Sk}*

***if*** *{Sj,Sk}€Si* ***then***

*Si ←Sj {that is VO=VO\Si}*

*VO=VO U Sk*

*stop ←False*

*Break (one split occurs; no need to check other splits)*

***end if***

***end for***

***end for***

***until*** *stop=T rue*

*Find k=arg max Si € VO {v(Si)/\|(Si)|}*

*Map and execute program P on VO Sk*

**Algorithm-1 : MSVOF Algorithm**

| Symbols | Description |
|---|---|
| **VO** | Virtual Organization |
| **P** | Program |
| **Si, Sj, Sk** | Nodes |

We can see this algorithm has 2 major parts. In the first part of this algorithm it finds the min cost of consecutive nodes and merging into VO through merge algorithm. In the second part the merged nodes are separated through split algorithm based on their cost. It is based on the nodes cost and not about the job. Also, the execution time is not measured accurately. So we are moving for another formation mechanism called PVOF (Pipelining Virtual Organization Formation).

Here we are proposing pipelining VO formation. This is used to split the job first, then process the job, then concatenate it for the desired output. To get the shortest distance and choosing the correct node we are using the min path algorithm. It used to connect all the nodes and the shortest distance between all the nodes.

## 4.1 Min Path Algorithm:

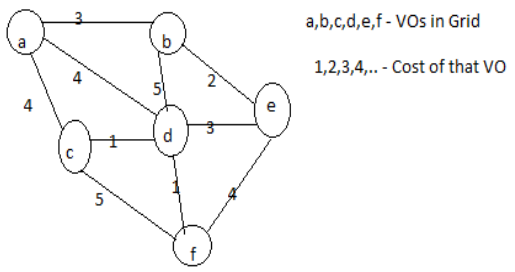


Fig. 1.1 GIven VO Path to find MIn Path

To find the minimum cost we are using the min path algorithm. From this all the modes of the VO will be connected and the resources will be shared among the Grids. The cost will be the distance between one node to the other. This min path algorithm to define the minimum cost of the traversing nodes as well as the time consumed is given as follows:

```
IntV = VO->V;
  structnode result[V];
  intx = 0;  // E An index variable, used for result[]
  intj = 0;  // A variable, used for sorted nodes
// array of nodes
  nsort(VO->node, VO->X, sizeof(VO->node[0]), class);
  // Allocate memory for creating V ssubsets
  structsubset *subsets =
    (structsubset*) malloc( V * sizeof(structsubset) );
  // Create V subsets with single elements
  for(intv = 0; v < V; ++v)
  {
    subsets[v].parent = v;
    subsets[v].rank = 0;
  }
  // Number of nodes to be taken is equal to V-1
  while(x< V - 1)
  {
    // Step 2: Pick the smallest node. And increment the
index
     // for next iteration
     structnodenext_node = VO->node[i++];
      intk = find(subsets, next_node.src);
     intl = find(subsets, next_node.dest);
      // If including this nodedoes't cause cycle, include it
     // in result and increment the index of result for next
node
     if(k != l)
     {
       result[x++] = next_node;
       Union(subsets, k, l);
     }
     // Else discard the next_node
  }
      for(i = 0; i< x; ++j)
```

```
    print("%d -- %d == %d\n", result[j].src, result[j].dest,
                                result[j].weight);
  return;
}
```

**Algorithm 2 : Min Path Cost Algorithm**

| Symbols | Description |
|---|---|
| **V** | Virtual Organization |
| **n** | No. of Nodes |
| **k** | Index |
| **node** | VO node |

By using this algorithm we are finding the minimum cost path for the given VO formation. With respect to the given cost, the minimum path has been detected. The VO following this will be the final solution for the given problem.

After detecting the minimum path we have to divide the original job into the sub tasks. Then the subtasks will be divided into some other sub tasks. In this Fig 2.1 the minimum path calculated VO has been given.

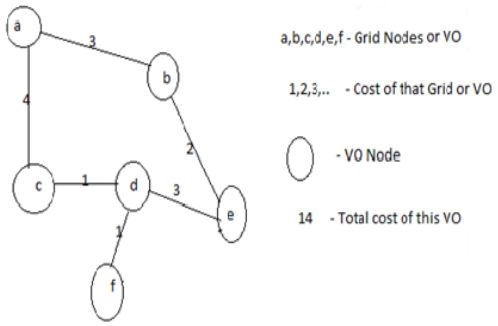

Fig 2.1 MIn path calculated VO

## 5. PIPELINING VIRTUAL ORGANIZATION FORMATION:

We are going for the pipeline algorithm which can be used to divide the task into sub tasks and then process it. It will process the subtask1; at the same time it will process the subtask2 in another VO. So it works with parallel time at distributed VO grids. It is given by the algorithm as follows:

*G={{Vo1},{Vo2},{Vo3},…}*
*for all VOx,VOy €G where x≠y do*
*for all partitions {Vox,Voy} of G*
*where Vox=Vox U Voy ∩ Voz*
*Map program P for min cost on Voy*
*Map program P for min cost on Voz*
*If {Voy,Voz}>Vox then*
*Vox←Voyi.e G={G/Voy}*
*G=GUVoz*
*Stop ←false*
*Break (one split occurs;no need to check other splits)*
*end if*
*end for*
*end for*
*until stop = True*
*Find z=arg max Vo€G{Vox/Voz}*
*Map and execute program P on VOz*

**Algorithm 3 : Parallel VO Formation**

| Symbols | Description |
|---|---|
| **G** | Virtual Organization |
| **P** | Program |
| **Vox, Voy** | VO node |

This algorithm is used to break the task into sub tasks and then doing the jobs in parallel time. In this algorithm G is having the virtual organization group which is divided into some other VO. Then the task will be provided into that VO. VO1 is the group having some grid formation. In that, the tasks will be shared among them and processed.

The minimum cost of VO will be calculated before executing this algorithm. Based on that cost, the task will be shared and executed. The advantage of this approach is to achieve the output by using parallel systems and memory usage efficiently.

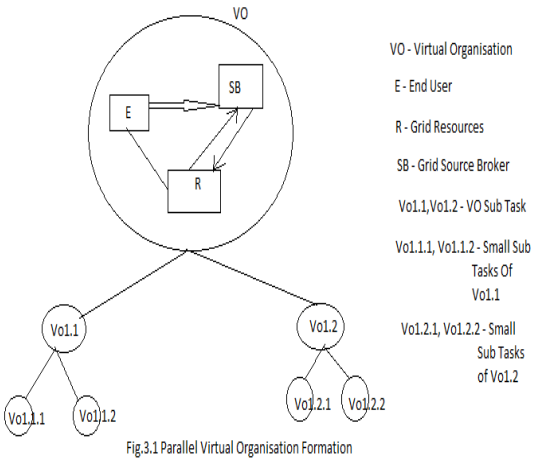

Fig.3.1 Parallel Virtual Organisation Formation

## 6. EXPERIMENTAL ANALYSIS:

In network stimulator (NS2), we have implemented PVOF and MSVOF with different number of nodes and cost. It is given from the output that PVOF is better than MSVOF based on their performance, since PVOF is performing many tasks at the same time. So it is better in performance. NS2 stimulates the MSVOF as well as PVOF with the exact monitoring values.

Based on the cost and number of nodes the PVOF will give output. If the number of nodes is increased, then the minimum cost will also be calculated and the cost also increased. Based on the experimental and performance analysis it shows that the PVOF is better than MSVOF. It has been calculated through our experiments.

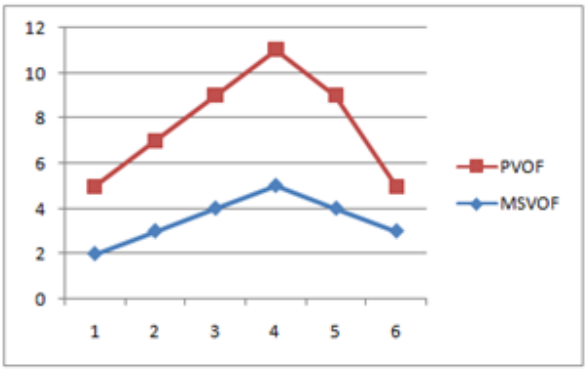


Fig 5.1 Experimental Analysis

The PVOF is based on time and not based on the number of nodes, because it splits the task with respect to time. MSVOF does not split the task. PVOF gives the output based on time. The proposed PVOF gives better performance than MSVOF based on the experimental results.

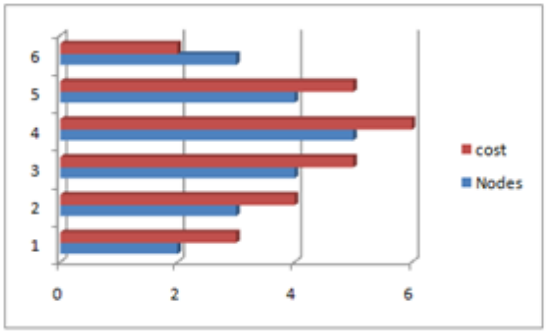


Fig 4.1 Perfomance Analysis Graph

## 7. RELATED WORKS:

Grid Information service architecture defines low level enquiry protocol and registration protocol to incorporate individual entities and discovery strategies.

These combined Grid protocols are constructing high level services that are having the capabilities of brokering, monitoring, fault detection and trouble shooting.

Through Monitoring and Discovery System (MDS-2) architecture, information services have been implemented in Globus tool kit which is widely used. This architecture has to develop flexible configuration tools for VO formation and should extend our security model. [1]

The Xtreem OS (Linux-based Operating System) is used to support VO Management. It is based on Linux operating system. It is providing core grid services to handle different collaboration models. It ensures the scalability and compatibility across all the applications. This is implemented in three flavors, i.e. Clusters, PCs and Mobiles. This will pose new challenges in cloud computing for trustworthy mechanism and services. This will take an effort to explore the architecture for Cloud Domains for managing credentials and ensuring security among users and resources. [2]

Security will play an essential part in exchanging data. SSL (Secure Socket Layer) is used to exchange data instead of message level security. When the message level security is not required we can use SSL. In message level security Web Services (WS) Secure Connection is used. But it is not suited for huge number of clients. [3]

When Globus toolkit is facing few security issues, it has been proposed with some improvements from the previous versions and evolution of open grid services architecture. It needs some more to extend experimental works in WS routing for compatibility, implementing the standard services and identity mapping. [4]

The Virtual Grid is similar to physical grid with the workspaces of their applications. It should have coarse grained control over workspace images via attestation. Thus virtual organization works flexibly and independently in a specific way and provides the workspace to be deployed from different communities and platforms. It reduces the administrative cost of maintaining grid resources. Much research is going for virtual playground which consists of group of virtual organization and also ensuring the security in the virtual playground. [5]

The Grid Trust Security framework is used for trust and security in the Next Generation grid architecture. The Grid Trust approach has 3 layers. Grid Application layer, Grid Service middleware layer, grid foundation layer. This architecture works with the last 2 layers. It composes of tools and services authorizing credentials and ensuring that the service task has been completed. It uses a secure resource broker and a reputation service in the middleware and in the foundation layer. Also, it has VO level policies and computational level policies. [6]

## 8. CONCLUSION:

We have already started using Grid Programming, so it is important to know what kind of results we are going to give to the real world. Here we have used PVOF for organization of the Virtual Grids. In future, exploring the dynamic representation of grid and examining the process flow will be an added advantage. It will reduce the cost management and the execution time.

## 9. ACKNOWLEDGMENTS:

I hereby thank Mr.DOJOHN LOYD B. for his valuable guidance for preparation of this paper. I am also grateful to our HOD, Mr.J.Jagadeesan and other staff members of Computer Science Department, SRM University, Ramapuram Campus, for their support and guidance.

## 10. REFERENCES:

[1] Karl Czajkowski, Steven Fitzgerald, Ian Foster, Carl Kesselman, "*Grid Information Services for Distributed Resource Sharing"*

[2] Massimo Coppola, YvonJégou, Brian Matthew, Christine Morin, Luis Pablo Prieto, Óscar David Sánchez, Erica Y. Yang, Haiyan Yu, "*Virtual Organization Support within a Grid-Wide Operating System"*

[3] Von Welch, Frank Siebenlist, Ian Foster, John Bresnahan, Karl Czajkowski, JarekGawor, Carl Kesselman, Sam Meder, Laura Pearlman, Steven Tuecke, "*Security for grid Sevices"*

[4] Satoshi Shirasuna, AleksanderSlominski, Liang Fang, Dennis Gannon, "*Performance Comparison of Security Mechanisms for Grid Services".*

[5] K. Keahey et al., "*Virtual Workspaces: Achieving Quality of Service and Quality of Life in the Grid*," Scienti!c Programming J., special issue on dynamic grids and worldwide computing, vol. 13, no. 4, 2005, pp. 265–275

[6] Philippe Massonet, "*Trust and security for next generation Grids"*

[7] Lena Mashayekhy, Student Member, IEEE, and Daniel Grosu, Senior Member, IEEE. "*A Merge-and-Split Mechanism for Dynamic Virtual Organization Formation in Grids",* VOL. 25, NO. 3, MARCH 2014

[8] Krsul, I., A. Ganguly, J. Zhang, J. Fortes, and R. Figueiredo. "*VMPlants: Providing and Managing Virtual Machine Execution Environments for Grid Computing. in SC04".* 2004. Pittsburgh, PA

[9] Keahey, K., I. Foster, T. Freeman, X. Zhang, and D. Galron, "*Virtual Workspaces in the Grid."* ANL/MCS-P1231-0205, 2005

[10] Foster, I., C. Kesselman, and S. Tuecke, "*The Anatomy of the Grid: Enabling Scalable Virtual Organizations."*International Journal of Supercomputer Applications, 2001. 15(3): p. 200-222

[11] Chase, J., L. Grit, D. Irwin, J. Moore, and S. Sprenkle, "*Dynamic Virtual Clusters in a Grid Site Manager."* accepted to the 12th International Symposium on High Performance Distributed Computing (HPDC-12), 2003.

[12] Foster, I. and Kesselman, C. "*Globus: A Toolkit-Based Grid Architecture".* Foster, I. and Kesselman, C. eds. The Grid: Blueprint for a New Computing Infrastructure, Morgan Kaufmann, 1999, 259-278.

[13] Foster, I., Kesselman, C., Tsudik, G. and Tuecke, S. "*A Security Architecture for Computational Grids".* ACM Conference on Computers and Security, 1998, 83-91

[14] Nakada, H., Sato, M. and Sekiguchi, S. "*Design and Implementations of Ninf: towards a Global Computing Infrastructure. Future Generation Computing Systems,"* 1999.

[15] Wolski, R. Forecasting "*Network Performance to Support Dynamic Scheduling Using the Network Weather Service."* In Proc. 6th IEEE Symp. on High Performance Distributed Computing, Portland, Oregon, 1997.