A Grid Computing Tool for Engineering Simulations

Bhakti S. Lad[*] bhakti@zeusnumeix.com,
Mridula U. Pillai, Ashish Choudhari
And Yatish Gupta
Zeus Numerix Pvt Ltd, M-03, SINE, CSRE Building, IIT Bombay Campus, Powai
Mumbai - 400 076, INDIA

ABSTRACT

There has been steady and notable rise in the use of numerical simulation packages for predicting behavior of engineering systems. The numerical simulations have become an important design tool. However, experimental tests have not been replaced by numerical simulations tools as of yet completely [1]. Experimental methods seem to be better for predicting off-design design conditions, whereas numerical simulations seem to be a winner as a design tool at design point and for optimization studies.

Computational Fluid Dynamics (CFD) is one such tool which is on the verge of reducing, if not completely replacing expensive wind tunnel and flight tests. This is possible due to due integration of CAD, meshing and post processing tools with CFD solvers. However, the most important contribution to this will happen when software is provided to an aircraft designer for executing CFD faster, cheaper and with confidence on his desk top. This possibility can become a reality with availability of Grid Computing environments such as Garuda Grid. The present work describes, how CFD software "**CFDExpert™**" which as CAD, Meshing and Post-producing modules has been encapsulated to produce a formidable for aircraft designer "**CFDManger™**" at Zeus Numerix.

INTRODUCTION

CFDManager™ is a system architecture which is developed to automate the grid computing and it lets people share computing power, databases, and high computational machines securely across corporate, institutional, and geographic boundaries without sacrificing local autonomy. **CFDManager™** is designed and developed by **Zeus Numerix Pvt. Ltd**. This architecture is tested for Computational Fluid Dynamics (CFD) simulations.

The challenge is to improve operational efficiency by automating the file transfer job, managing the execution status, allowing bulk transfer of multiple files, thus improving the overall availability of a distributed system by monitoring critical system components, notifying system administrators when services fail and automatic rescheduling of jobs in case of failures. This mechanism must not require users to learn the details about service configurations, networks, or system architecture.

The current system does not address the concerns and requirements just listed. CFD users need to access large datasets generated by various complex computational models in order to carry out simulations. This operation is performed at remote machines, held by institutions capable of providing high computational power and other valuable computational resources as required by the operation. The file transfer operation is carried out manually wherein the user is responsible for selecting the appropriate node for remote submission and execution of the jobs.

[*] Author for communication

Once the job is submitted all is not over. Since the computation may run over several hours, it is necessary to repeatedly check the job status.

For a naïve user it is necessary to keep note of the jobs submitted, as submission of a job having the same name as an existing job cause result in the overwriting of the existing job. Also users have to invest time in performing functions like ftp, scp or ssh to transfer jobs to the remote machine by manually logging into the specific remote node, and all the intermediate servers laid in the path of the remote machine. The current technology does not support automatic retrieval of finished jobs. It is extremely tedious for the user to log in every time he needs to see the status of the submitted job. The user also needs to be aware of the commands required for execution which is separate for different nodes in question. Any failed execution attempt requires that the users submit the jobs again and reschedule it. The Remote Server does not have any provision for rescheduling of failed jobs.

This results in a lot of wastage of precious time, as the user has to bother about every stage involved in job submission and retrieval procedure. By automating the process, human resources can be redeployed to higher-value activities.

SYSTEM DESCRIPTION

Transfer of job files between machines for any kind of computation purpose is one of the most fundamental operations in any Computational Grid is in use, but the major and critical point in such operations is the user understanding of these operations.

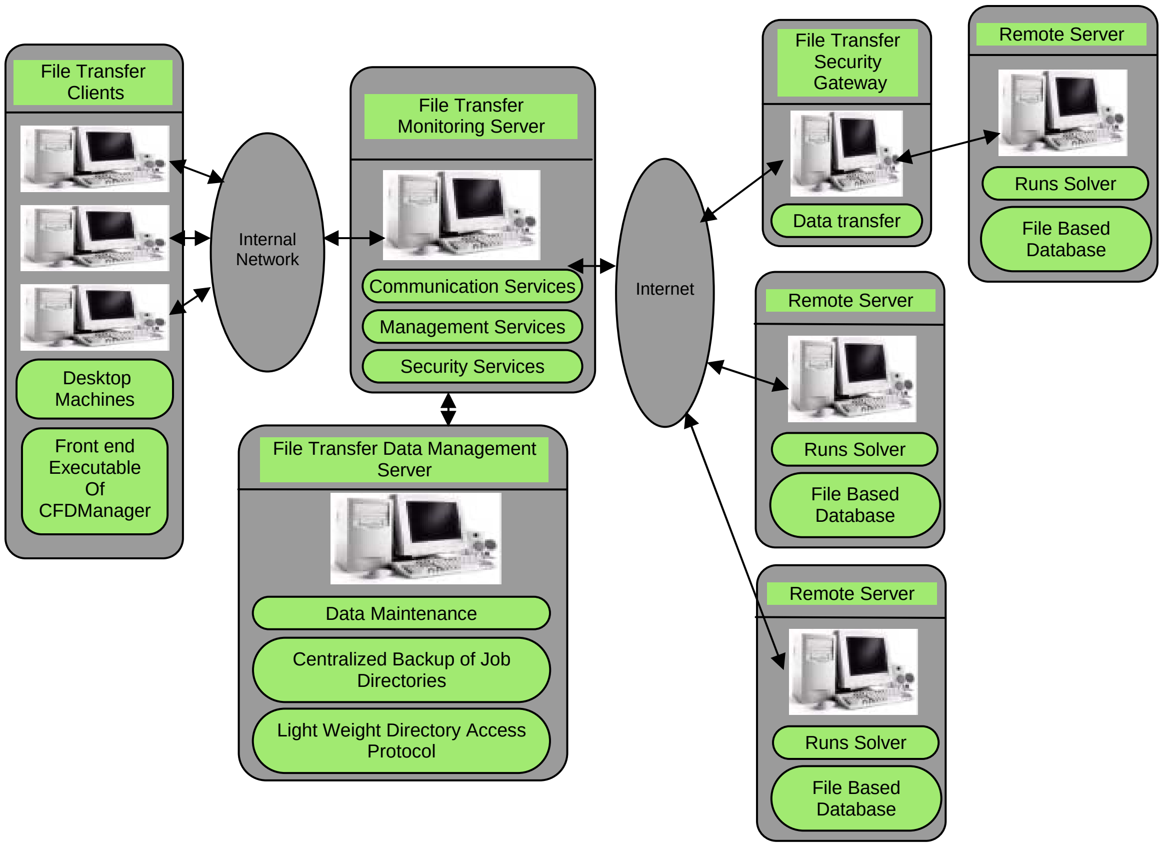

Figure 1

So the architectures and technologies used to perform all the operations involved, taking into consideration the users, have a great impact on usability, scalability, and security of the business applications that rely on the underlying file transfers. **CFDManager™** gives flexibility of transparently accessing the appropriate external computational resource without human intervention. Instead of going all the way to the remote machine if the remote machine could provide the user with the data at their desktop machine it would be a boon for CFD users. And this is just what **CFDManager™** does; following the analogy “Everything is provided in the plate. All you need to do is Eat.”

To achieve this **CFDManager™** is divided into major 4 parts as shown in the Figure 1 above:

- Clients
- Data Management Servers and Monitoring Servers
- Security Gateway (Optional)
- Remote Servers

Clients

These are the desktop machines on which user’s data reside. This is the “Plate” and all the “Food” is provided here. The user can create any number of jobs for execution; can select the type of node (serial or parallel) on which they want to fire the job for computation, and then just click a button to SUBMIT. The user is no longer responsible for monitoring the submitted job. He need not be aware of the details of the remote machine to know the status of job, or if he wants to retrieve the end computation results (or intermediate result).This can be achieved by sending STATUS and RETRIEVE request. He need not use any command to transfer files, view status, and retrieve the solution.

This is where the simplification of all operations carried out by the user comes. The Client does this in conjunction with Data Management Servers and Monitoring Servers. For all the jobs to be submitted, the Client is assigned a unique jobid by the Monitoring and Data Management Servers on request. This allows the users the freedom of submitting multiple jobs with the same name without the fear of having one copy overwritten by the other at the remote machine. The Client maintains a database locally which keeps the information about all the jobs submitted by Client. This information takes the form of data header which is the primary communication between all the four parts of the **CFDManager™**. The data header is identified by the unique job id. The Client database is periodically updated by Monitoring Servers on request. The Client also provides the users with the capability of issuing KILL request to submitted jobs. Each command or functionality is associated with an operation code. The operation code is part of the data header and allows the **CFDManager™** to take necessary action on the submitted job identified by the jobid specified by the data header. So all the actions make necessary changes to the data header and the updated header is communicated with all parts of **CFDManager™**.

Data Management and Monitoring Servers

As the name suggests these Servers manage the data related to submitted jobs and periodically monitor them to take necessary actions. These Servers keep track of all possible servers in the grid available for computation and the list of all the jobs submitted, thus acting as a mediator between Client and Remote Server. Essentially what this layer of Servers does is to accept the job folders and their data headers from Client and assign it, i.e., transfer them, to an appropriate available node or Gateway Server (if involved). All the logging-in involved (between the internal and external network) is transparent to the user. These servers support only data transfer by authenticated clients.

The major tasks performed by Monitoring Servers are:

- *Security Service*: Adds a level of security in accessing and using high computing resources. Only authenticated users can fire their jobs.
- *Communication Service*: Checking the availability of the Remote Server of choice. If unavailable, job is queued or assigned to some other Server.
- *Management Service*: Assigning a unique identifier to each job for ease of data maintenance as well as client job identification; checking for the updated status of the job from the Remote server, and then updating it into the database.

The major tasks performed by Data Management Servers are:

- *Data Maintenance*: Keeps database of all client and Remote Servers in the network.
- *Centralized Backup of Job Directories*: This is the server where all directories of various users have been stored and user can look on this server as their backup server.
- *LDAP*: CFDManager uses Lightweight Data Access Protocol to maintain two databases. One for the remote resources or servers and another for all the submitted jobs. The submitted jobs are represented as data header in the database.

The tasks mentioned above are performed by two daemons loaded on these Servers. They form a part of the back end program for **CFDManager™**.

The two daemons are as follows

1) Daemon communicating with Client: This daemon constantly listens to Client, accepts the request and provides the Client with a response if required. The various request –response(s) are:

SUMBIT-JOBID
DISPLAY STATUS-UPDATED DATAHEADER
RETRIEVE-UPDATED DATAHEADER+JOB FOLDER

2) Daemon communicating with Remote Servers or Gateway: This daemon constantly listens to Remote Server for update in status of submitted jobs and also monitors the database for new entries of jobs to be submitted. It determines which resources are best for the task; chooses a resource if available, and attempts to use the resource by transferring the job directory and data header to Remote Servers. Due to periodic updates the ldap database always reflects the most recent status of the submitted job.

The functionality of the two Servers can be clubbed together into one if required.

Security Gateway

On external networks where security is of prime importance only authenticated users are allowed to use the resources. These are the special purpose Servers which will transfer the data header and job directory between Monitoring Servers and Remote Server incase there exists an authentication gateway between the internal and external networks. On this machine one daemon of **CFDManager™** is loaded.

Remote Servers

These are the servers on which actual computation is done. Here various solver executables are placed, which are used for solving given problem. The daemon stored on this machine will constantly listen to Monitoring Server or Gateway. It maintains a database of all the incoming data headers and on the arrival of a new data header and job directory it updates the database and puts the job into execution. It periodically updates the database with the current status of the submitted jobs and forwards these updates to the Data Management and Monitoring Servers. Based on the operation codes of a data header the Server takes the necessary actions. For instance, if the operation code of the incoming data header specifies a kill request, it aborts the job. On completion or in case the job is aborted, then the server transfers back the updated data header and job directories containing the solution files to the Monitoring Server or Gateway.

FUTURE WORK

While CFD solutions are being desired for increasingly large set of input data for design purposes, its application to practical cases involving complex domains & large mesh points needs attention.

Solution of linear equations consumes the major portion of computing time in a CFD based flow solver. Development of an effective matrix solver, thus, becomes critical for solution of future CFD applications. We identify a combination of efficient algorithm & associated computational hardware for achieving the desirable performance. The following characteristics are desirable from such a matrix solver:

1. Obtaining CFD results on domain with more than 10 million mesh points in reasonable time frame.
2. Inclusion of parallelized multigrid method as efficient mean to obtain solution to such large set of equations.
3. Utilization of nationwide grid computing facility, GARUDA.
4. Dynamic domain decomposition based on problem size & availability,
5. Scalability with increasing number of processors.

Initial results have been generated on the above framework & have demonstrated excellent performance on many cases.

CONCLUSION

As grids grow, automation becomes essential. Here (In this paper), we have outlined the importance of automation to grid computing and how automation is a key element of the grid computing solution architecture.

Clearly, scheduling automation and system monitoring are the logical choice to improve system performance. The innovative development of **CFDManager™** will yield more benefits to the users as it makes the pre-computation requirement on a grid environment an easy and smooth task. Removing the human element from processes that can be automated usually results in greater consistency and decreases the chance for errors.

REFERENCES

1. Forrester T. Johnson, , Edward N. Tinoco and N. Jong Yu : Thirty years of development and application of CFD at Boeing Commercial Airplanes, Seattle, Computers & Fluids, Volume 34, Issue 10, December 2005, Page 1115-1151