# Grid Enabled Optimization

Hee-Khiang Ng, Yew-Soon Ong, Terence Hung[2], and Bu-Sung Lee[1]

[1] School of Computer Engineering, Nanyang Technological University,
Nanyang Avenue, Singapore 639798
{mhkng, asysong, ebslee}@ntu.edu.sg
[2] Institute of High Performance Computing, Singapore Science Park II,
Singapore 117528
terence@ihpc.a-star.edu.sg

**Abstract.** In this paper, we present a scalable parallel framework, which employs grid computing technologies, for solving computationally expensive and intractable design problems. Using an aerodynamic airfoil design optimization problem as an example the application of the grid computing strategies is discussed.

## 1 Introduction

Grid [1] Computing has gained widespread popularity in the research community for distributed and parallel computing because it has tremendous potential in enabling complex applications, especially those requiring huge amount of computation power and data storage requirements. A rising trend in the science and engineering world is in the utilization of increasingly high-fidelity and accurate analysis codes in the design analysis and optimization process. In many application areas such as photonics, electro-magnetics, aerospace, biomedical, micro-electro-mechanical systems and coupled-field multidisciplinary system design processes, the design process generally requires a Computational Structural Mechanics (CSM), a Computational Fluid Dynamics (CFD) or a Computational Electronics & Electro-magnetics (CEE) simulation procedure. The time taken for these processes generally varies from many minutes to hours or days of supercomputing time. This would often lead to high computing costs in the design optimization process, hence a much longer design cycle time to locate the optimum design solution. The benefits of Grid computing in the context of evolutionary computation, especially on computational expensive design optimization problem are numerous. In particular, the ability to tap on vast compute power. For instance, specialized high-fidelity analysis codes and computing nodes owned by different design teams that spans across geographically distributed locations may be shared and better utilized.

Solving large scale optimization problems requires a huge amount of computational power. The size of optimization problems that can be solved on a few CPUs has been limited due to a lack of computational power. The recent development in Grid has received much attention as a powerful and inexpensive way of solving large scale optimization problems that an existing single-unit CPU cannot process. The aim of

P.M.A. Sloot et al. (Eds.): EGC 2005, LNCS 3470, pp. 296 – 304, 2005.

this paper is to show that grid computing provides tremendous power to solve such large scale optimization problems.

The rest of this paper is organized as follows: In section 2, we present a brief overview of the fundamental concept of Evolutionary Algorithms, and in Section 3 we discuss the aerodynamic design and in section 4, we describe the implementation aspects of wrapping airfoil analysis code as services and realizing genetic algorithms as Grid enabled services. Finally in section 5, we conclude this paper.

## 2 Evolutionary Algorithms

In this section, we offer a brief overview on evolutionary algorithms, in particular, the general forms of parallel evolutionary algorithms that exist. Evolutionary Algorithms (EAs) are modern stochastic search techniques inspired by the 'survival of the fitness' principle of the Darwinian theory of natural evolution. By simulating natural evolution, EAs has been employed for solving many complex problems. A well-known strength of EAs is the ease of extensions to incorporate parallelism []. For instance, Parallel Genetic Algorithms (PGAs) are extensions of the standard Genetic Algorithm. Since the algorithm works with sets of populations, instead of a single individual, the basic concept of PGA is a simple division of the tasks in a classical GA across different processors. The other advantage of PGA is that it facilitates speciation, a process by which different populations evolve in different directions simultaneously. They have been shown to speed up the search process as well as to obtain higher quality solutions when dealing with complex design problems. In the paper, the PGA is applied to the optimization of an aerodynamic airfoil design problem, whereby the optimization is carried out through a number of generations. In each generation, the algorithm produces populations for the design analysis process. These results would then act as inputs in the next iterative stage of the numerical procedure.

Aerodynamic design optimization is one of the most frequently tackled problems in the aeronautics industry. It generally poses serious economic questions as the analysis is often very computationally expensive and often requires very intensive design solutions. In such situations, due to the large aerodynamic design search space often required, stochastic optimization algorithms such as EA are often employed, in the search for the near optimum design. Typically, such algorithms requires thousands of function evaluations, which often requires excessive CPU times, in locating the near optimal solution, whilst each function evaluation, using the high-fidelity analysis codes, may take up to many minutes to hours of computing time. Hence, it is not uncommon to hear that the design of an aircraft wing, using such algorithm, usually takes up to several months of CPU time even while running on supercomputers. This is a major obstacle in the practical application of EA optimization to complex engineering design problems, as the design process may prove to be computationally intractable if the optimal design is desired.

On the other extreme, however, the main advantage of EA is that it is a population-based technique, which allows the parallelization of the algorithm, in such a way that the designs in each generation can be evaluated simultaneously across the range of available machines. This is undertaken here as the computing complications encountered in the aerodynamic airfoil design problem are tackled with a parallelized

EA, deployed on a High Performance Computing (HPC) platform such as NetSolve, across multiple HPC clusters. This allowed a large number of these evaluation tasks to be farmed out to various nodes/CPUs within the cluster using a cluster scheduler arrangement. As a result, improved optimization solutions in the design cycle are achieved. The details of the numerical implementation of these algorithms are described in subsequent sections.

### 2.1 Genetic Algorithms

Genetic Algorithms is a member of the EA family employed in the numerical procedure. The term Genetic Algorithms or GA was first introduced by Holland in his seminal book referring to the probabilistic search techniques inspired by the 'survival of the fittest' principle of the Darwin theory of natural selection. Artificial creatures are created, which will then compete in a struggle for life. The fittest will then survive and are then allowed to reproduce. New creatures will then be created again using some operators such as crossover and mutation. The schematic workflow of this repetitive process is shown in Figure 1.

```
Procedure GA
Begin
  for i:=1 to n do begin
      initialise(x_i)          // with random values
      f(x_i):=evaluate(x_i)    // determine fitness
  end
  while termination condition not satisfied do begin
      p_1...n:= select from x_1...n in random order
      for i:=1 to n step 2 do begin
              x'_i,x'_i+1:=crossover(pi,pi+1)
              mutate(x'_i)
              mutate(x'_i+1)
      end
      x_1...n:=x'_1...n        // copy new population
      for i:=1 to n do begin
              f(x_i):=evaluate(x_i)
      end
    end {while}
end
```

**Fig. 1.** The Schematic Workflow of Canonical GA

## 3 Aerodynamic Airfoil Design

One of the most important features of an aircraft is during its takeoff, *i.e.* to get the body off the ground and subsequently to maintain it there. This usually comes from the efficient process of lifting the body up. By far the most common means of lifting, widely used in today's industry, is the wing. It is a concept ergonomically designed and embodied in current modern machinery. In actual fact, wings of all shapes and

sizes can be used to lift these bodies up, but special care must be taken to ensure that these wings are of the proper geometry. This is to ensure stability, efficiency and performance excellence.

Simplifications were made to the original design of the aerodynamic airfoil design problem. As the main emphasis of the study is to show the possibility of using EA, with the proposed alternative procedure, to solve such problems, the assumptions and changes made does not affect the gist of the problem. As such, the details of the problem are not discussed here, but can be found in literatures published elsewhere [8].

The simplified problem, considered in this study, is to achieve the best 24 design variables with an optimized drag profile, *i.e.* a minimized drag and a maximized lift, at a Mach number value of $M\propto \in [0.45,0.5]$, with the best midpoint value of $0.5$ at an angle of attack (AOA) of $0.2$. These requirements are schematically illustrated in Figure 2.

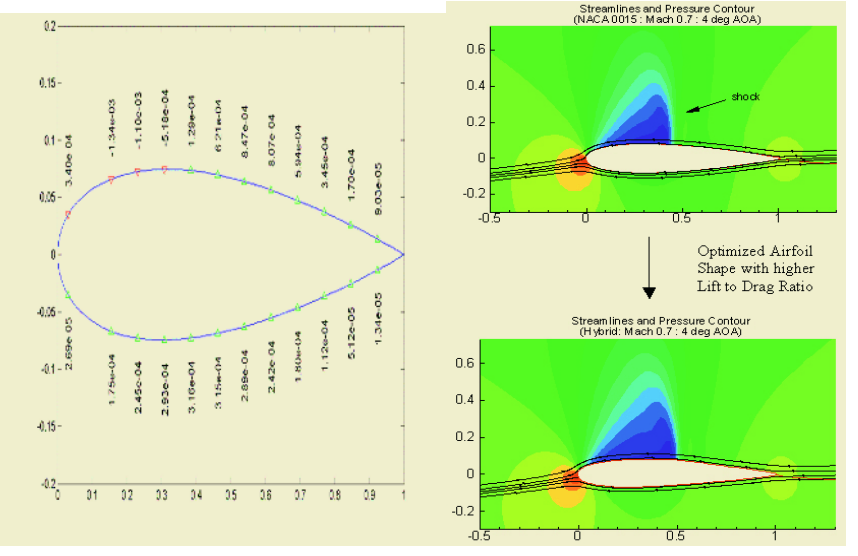


**Fig. 2.** Simplified aerodynamic airfoil design problem

## 4 Grid Enabled Optimization (GEO)

Here, the architecture of the grid enabled optimization services for each grid cluster, on the aerodynamic airfoil design optimization problem, is described. For this particular computing setup, the airfoil analysis problem enabled as an 'airfoil analysis' service is implemented to facilitate parallelism in the algorithm. This service were deployed on the cluster and registered to the resource Agent. This implies that the latter would search for the available resources and allocate them to the 'airfoil analysis' Services.

In the implementation of the proposed grid computing technology in this paper, two essential features are highlighted. The first is that a extended GridRPC API is used to 'gridify' the applications. This is done or 'gridified' using Globus [13] API, CoG built into our extension of the GridRPC API. This choice is down to its relative simplicity in implementation and its ability to offer high-level abstraction, *i.e.* the concealing of complex grid environment from the users.

The second feature of the proposed procedure is the implementation of a meta-scheduler, to be discussed later. From reviews conducted, it was discovered that the current state-of-the-art GridRPC implementations lack the feature for automatic resource discovery and selection on the grid environment. As a result, users have to perform look-up and select resources that suit their needs manually before they can assign computing tasks to the respective resources. This leads to the inefficient use of resources since large amount of resources exist on Grid and these are often dynamic. Furthermore, most optimization problems generally have a large number of evaluation tasks that requires many identical computations, with different analysis parameter sets. Hence it is extremely inefficient if the interactions between the Clients and resources are repeated many times for the same remote procedure call. This is overcome here with the use of a meta-scheduler which schedules GridRPC requests and balances the workload across multiple clusters on grid.

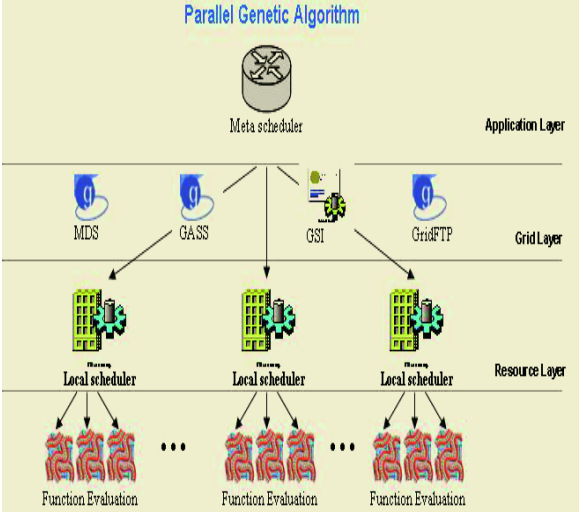


**Fig. 3.** Components in the GEO

Another characteristic of the proposed grid computing system is the employment of a Lightweight Directory Access Protocol (LDAP), to query the MDS database for available resources and workload information. To the GridRPC Client, this acts as a centralized directory service to scout for the available resources and identify their location within the grid environment. In its initialization process, the GridRPC function requires a simple entry from the MDS server and a virtual organization name to

retrieve the available resources and check whether or not these resources are capable of servicing further GridRPC requests. These checks were performed for software, *i.e.* remote procedure, specialized libraries, *etc*, hardware, *i.e.* platform, operating system, *etc* and the workload. In this regard, the Ganglia monitor toolkit is employed, to monitor and provide information relating to the available clusters in the computing systems.

For the multi-clusters execution of the PGA task, the adopted procedure is developed from the Globus toolkits. It basically involves using the Global Access Secondary Storage (GASS) to marshal the results back to the Client program, where the multi-clusters farming process initiates. GridFTP is then used to perform the transfer of the design input variables files to the selected clusters, which are available on grid. This is then followed by the use of Globus job submission protocol to facilitate the execution of the PGA process at the grid resources. This outline is shown in Figure 3, clearly presenting the different components in the procedure.

### 4.1 GEO Workflow

For the aerodynamic airfoil design optimization problem considered, the process begins with the meta-scheduler searching for the grid resources, to obtain the list of compute resources. The meta-scheduler also contains the information relating to the aerodynamic problem, which is deployed on the resources through the lookup on Globus MDS. Once the required computational resources are identified, the input design variables files are then transferred to the selected resources using GridFTP. Upon the successful transfer of these files to the resources, the application will then farm out the different chromosomes to the different clusters in parallel using the Globus execution command. Globus will then initiate and execute this task on the remote grid resources. When execution completes, the results are then marshalled back to the Globus Client application based on the GASS mechanism. The Client will then collect this data and continue with the GA program execution, *i.e.* iterating the program until the termination criterion is met, which in this report, is number of generations to run the analyses.

As discussed, the meta-scheduler is used to achieve immense parallelism in the search process as well as the seamless access to the grid resources. In the context of evolutionary algorithm, the term parallelism here meant the division of a population into isolated subpopulations. This gives rise to the concept of distributed EA, where each compute node may be allocated to evolve a subpopulation of individuals as well as the periodic migration of chromosomes among each another. Its implementation here is based on GridRPC [10], a remote procedure call standard interface for grid applications, proposed by the Grid Forum Advanced Programming Models research group of the Global Grid Forum [11].

Basically, the workflow of the Grid-enabled GA procedure in Figure 4 can be assembled into 9 main stages. These stages are as follows: -

1) The procedure begins with the Client contacting the meta-scheduler for services and resources necessary for the performance of the analyses.
2) The metascheduler, which in essence is the heart of the solution process, then obtains a list of the available resources together with their availability status. This is obtained from services such as the Globus Monitoring and Discovery

Service (Globus MDS) [12] and the Ganglia Monitoring Service (Ganglia) [13]. It then farms out the parallel analysis requests to the available grid resources based on the workload and the resource information retrieved from the monitoring services. The important point to take note here is that every time a new resource joins the grid, it is registered to the Globus MDS. This implies that as more computer resources and power are available, it can be added to the system.

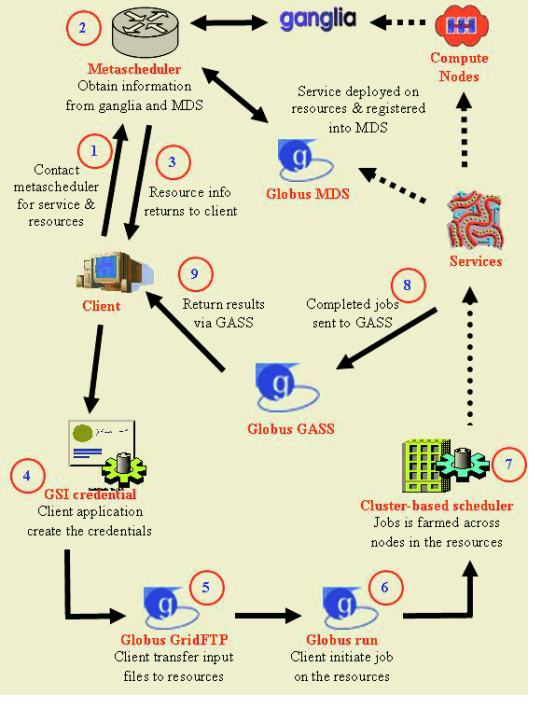


**Fig. 4.** GEO Workflow

3) These resources information and services are then fed back to the Client.
4) Upon obtaining the information in relation to the resources and services, the GSI [14] credentials are then generated. This can be viewed upon as an authentication or the authority to use the computing resources available in the system.
5) GridFTP [15] is then performed, to transfer the necessary input files from the Client to the resources and services.
6) The analysis job is then initiated at the resource cluster through the Globus submission protocol.
7) Whenever the Grid Resource Allocation Manager (GRAM) [16] gatekeeper of a cluster receives a service request, an instance of the service will be ini-

tialized on the master node of the cluster. Subsequently, the set of service requests will then be farmed across multiple computing nodes in the cluster by the cluster level scheduler, such as NetSolve. The responsibility of the local agent in each cluster is to perform local scheduling and resource discovery across computing nodes in the cluster for basic services.

8) The results of the completed jobs are then staged on the GASS [17].
9) Upon which, the results are channeled back from the GASS to the Client for assessment and evaluation.

## 5 Conclusions

The aim of this paper is to present our grid computing platform, which employs grid computing technologies, in solving computationally expensive and intractable design optimization problems. We successfully deployed the airfoil design optimization problem, onto the Nanyang campus grid [18]. From which, the optimization problem is executed across the grid computing platform, unleashing and utilizing the full power of grid computing technology.

## References


1. SIGHT, http://www.engineous.com, Engineous software Inc, "*iSIGHT, Version 7.1*", 2004
2. I.Foster, C.Kesselman, and S.Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations" , *International J.Supercomputer Applications*, vol. 15, no. 3, 2001
3. Agrawal S., Dongarra J., Seymour K., Vadhiyar S., "NetSolve: past, present, and future; a look at a grid enabled server", 2002.
4. Arnold D. C., Casanova H., Dongarra J., "Innovations of the NetSolve grid computing system", 2002.
5. The Globus Alliance, http://www.globus.org/, 2004
6. Goldberg D. E., "*Genetic algortihms in search, optimisation and machine learning*", 1989.
7. Gregor von Laszewski, Ian Foster, Jarek Gawor, Peter Lane, Nell Rehn, Mike Russell. Designing Grid-based Problem Solving Envionments and Portals Argonne National Laboratory, Argonne, IL 60439.
8. Giannakoglou K. C., "Design of optimal aerodynamic shapes using optimization methods and computational intelligence", Process in Aerospace Sciences, Vol 38, pp 43-76, 2002.
9. Ho Q. T., Ong Y. S., Cai W. T., "Gridifying aerodynamic design problem using GridRPC", 2nd International Workshop on Grid and Cooperative Computing, Shanghai, China, 2003.
10. Nakada H., Matsuoka S., Seymour K., Dongarra J., Lee C., Casanova H., "GridRPC: A remote procedure call API for grid computing", Grid Computing – Grid 2002, LNCS 2536, pp 274-278, 2002.
11. http://www.ggf.org/, GGF, 2004.
12. Fitzgerald S., Foster I., Kesselman G., Laszewski V., Smith W., Tuecke S., "A directory service for configuring high-performance distributed computations", Proceedings 6th IEEE Symposium on High-Performance Distributed Computing, pp 365-375, 1997.
13. http://sourge.ganglia.net, 2004
14. http://www-unix.globus.org/security/, The Globus GSI, 2004.
15. http://www.globus.org/datagrid/gridftp.html, GridFTP, 2004.

16. The Globus Resource Allocation Manager (GRAM), http://www-unix.globus.org/developer/resource-management.html, 2004,
17. http://www.globus.org/gass/, Global access to secondary storage (GASS), 2004.
18. Nanyang Campus Grid, http://www.ntu-cg.ntu.edu.sg, 2004