

SunTMGrid Engine architecture overview

Andrea Sottoriva, Julien Bonjean
MSc Grid Computing
Universiteit van Amsterdam, The Netherlands
18th October 2006

Abstract

A grid can abstractly be viewed simply as a collection of computational resources, able to compute tasks. Ones having this set of entities, the challenge is to manage them as a single huge machine. In this way the end users can view the grid as an opaque resource to which submit jobs, without being concerned about where they run. The SunTMGrid Engine provides a management solution for a specific type of grid that consists of many computational resources working together for users of a single project or department: the Cluster grids.

1 Introduction

The main tasks that Sun Grid Engine performs can be categorized as follows:

- Distributed job scheduling
- Resource balance and run-time management
- Authentication and authorization
- Resource and jobs monitoring
- Fault tolerance

Besides its functionality it's fundamental to keep in mind that the structure of SGE is strongly influenced by its native environment: UN*X.

The services and the other entities that live and communicate in the grid are implemented as UN*X daemons. Moreover SGE offers a huge set of command-line tools for job scheduling, monitoring and general managing without neglect a counterpart user-friendly interface. This dual approach raises the power of the software permitting, for example, to work in a scriptable environment such as a terminal.

In addition to the discussed features, it has to be considered that implementing a grid infrastructure in a UN*X environment permits to use all the already available standard and stable communication features such as *rsh* and *ssh*, *X-window forwarding*, *NFS*, *NIS+*, *RPC* etc..

2 SGE hosts and daemons

The first step to design a SGE cluster is to define the topology of the resource collection by selecting the right machines for the right tasks. SGE uses basically 4 hosts types: *Master host*, *Execution host*, *Administration host* and *Submit host*.

Each host can be at the same time a member of more than one category and according to that, it will run the appropriate SGE daemons. The only restriction to this subdivision of hosts is that it can exist only one *Master host* inside a single high-coupled SGE grid (called a *Cell*).

The master host is basically responsible to maintain all the information about the cluster and take decisions on jobs scheduling, acting as the *director* of the grid.

The execution hosts are nodes that have the permissions to run SGE jobs using a set of SGE queues, they represent the computational core of the cluster, sharing the concrete resource and providing CPU power, storage and memory capabilities.

The administration host perform any kind of administration functionality like for example monitoring, usually this tasks are performed from the same machine that is the *Master host*.

The submit hosts are those nodes allow to schedule *batch jobs* to the cluster, these kind of jobs are all the tasks that don't need interaction from the user, therefore the scheduling consists only on submit the job and save the output to a file rather than manage an interaction between the user and the job in the case of for example programs with a graphical user interface.

The main components of SGE technology are the daemons, those entities are implemented as a classical UN*X daemon, and are divided as follows:

- Master daemon (runs on the Master host)
 - maintains information about hosts, scheduling queues, jobs, systems load and permissions
 - receives scheduling decision from the scheduler daemon
 - requests actions to the execution daemon according to the scheduler daemon instructions
- Scheduler daemon (runs on the Master host)
 - decides which jobs are submitted to which queues in the system
 - forwards the decisions to the master daemon
- Execution daemon (runs on every Execution host)
 - it's responsible for all the jobs scheduled in its queues
 - periodically it sends monitoring information to the master daemon
- Communication daemon (runs on every host)
 - it manages the communication between all the hosts of the grid
 - all the communication is handle using a well-known TCP port

3 Jobs scheduling and management

From the point of view of the computation, the most important entity of the SGE infrastructure is the *queue*. An SGE queue is a container for jobs with specific needs (for example specific kinds of resources). There can be several queues for each host also of different types that fit different job profiles.

When a job is submitted the scheduler daemon will choose the queue that best fits the job requests according to a *job profile* provided by the user and considering also the system load distribution.

In the case the job profile is not specified, any queue enough free can be chosen without preferences beside the resource load distribution. Once that a queue is selected the order on which the jobs are processed inside it follows a FIFO policy with two additional criteria adopted for balancing issues: *Job priority* and *Equal share*.

The first permits to specify a priority value for a single job, and in order to maintain a load balance between users a further *Equal share* policy is applied by schedule those jobs of users with already other jobs on the end of the queue.

The idea behind these rules is to try to maintain both a resources load balancing provided with balancing to the users as well. More in the detail the SGE job system provides some other interesting features:

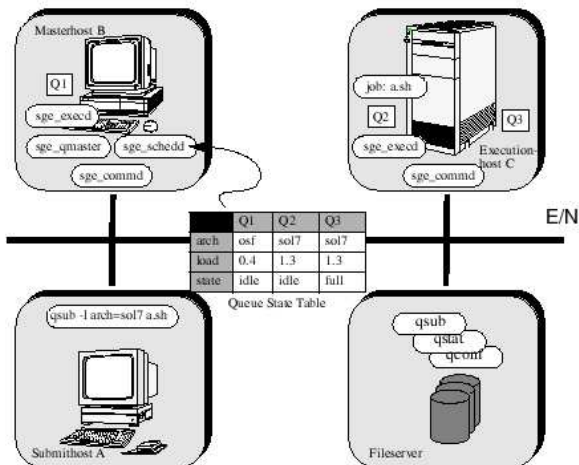
- spooling of jobs without a specific requirement in case of lack of free queues
- support for check-pointing of jobs (both user-level and, if available kernel-level)
- support for dependency between groups of jobs
- support for jobs-array (groups of jobs sharing the same code but with different parameters)

3.1 Queue configuration

The most challenging task that an SGE administrator has to perform in order to have a tidy and well working grid environment is the configuration of the queues on the system.

SGE offers a lot of different parameter used to specify the characteristics of a queue based on its host resources like load thresholds and resource usage limits. It's also possible to change at run-time the behavior of a queue, or schedule a *queue calendar* in order to define in which period that queue has those characteristic and so is suitable for certain specific kinds of jobs.

The whole system of queue capability specification, available resources definition per host and resource limits definition represents the *Consumable resource* SGE facility and have an hard influence on jobs scheduling and load balancing in the SGE system.



A typical SGE grid configuration

Is always due to the administration of the single execution host to configure its machine as he/she prefers, according to the amount and type of resource he/she wants to share with the grid.

4 Authentication and authorization

Given the UN*X environment on which SGE is developed, the entire authentication and authorization is based on the UN*X users and permissions model.

In order to submit jobs a user needs an account both on the submit host and on the execution host. All the resource permissions are managed using the standard DAC infrastructure provided by any UN*X system, therefore any user that has an account on at least one submit hosts and one execution host can use the SGE system to schedule a task.

Moreover, if a user wants to manage any component of SGE needs also the access to the SGE directories and its permissions are different according to one of the follows account types:

- Managers have the complete access to all the SGE functionality, he can manipulate every SGE component, such as queues, hosts and users
- Operators can perform almost every task of a Manager, except that he cannot add, modify or delete queues
- Owners are responsible only of their specific queue, they can modify its behavior

An additional security level based on CA certificates can be overlapped to the classical DAC infrastructure: the CSP (Certificate Security Protocol) system.

Setting up a Master host with the CSP system implies that all the communication between the grid components have to be authorized requesting to each entity to authenticate itself using a certificate. After that every message will be encrypted using a key exchanged through the public key of one of the component.

Besides this security layer at communication level inside the grid, the CSP system impose that the SGE users need not only to have an account on the grid hosts but also to provide a valid certificate and use the public keys to perform cryptography. Only the SGE administrator has the permissions to create certificates for users and to decide which kind of policy apply to its grid. Choosing to use the CSP system implies to impose this security layer to all the entities of the grid.

5 Managing parallel environments

Besides all the features already described, SGE can directly deal with parallel virtual environments such as PVM

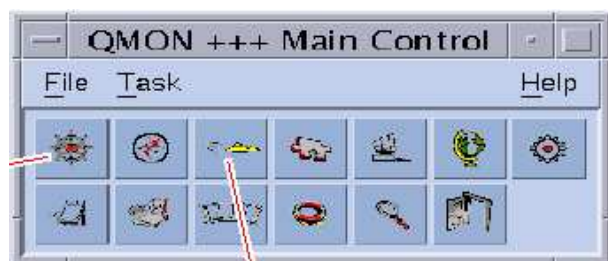
or MPI. In order to schedule jobs which this kind of characteristics SGE provides a simple interface to define *Parallel Environments*. A PE is a group of parallel jobs that have to be scheduled together giving them the change to communicate each other like they would be alone in a parallel environment.

Any kind of relationship between jobs of a single PE can be selected at scheduling time using the appropriate PE scheduling tools provided by SGE. In this way it's possible to manage parallel jobs in the same way of normal jobs, but with more restriction rules. Moreover this system offers an alternative way to spawn parallel tasks: most of the parallel technologies infact don't provide reliable resource managing and are often not enough flexible compared to the grid infrastructure, for that reasons SGE has the capability to perform a tight integration with parallel environments. A tight integration means that the responsibility to spawn parallel tasks is moved from the parallel environment level to the SGE grid level, providing the discussed resource management features.

6 Monitoring and managing

SGE provides a wide set of monitoring command-lines tools in order to manage user jobs and get resource consumption all around the grid. The chief interface program to the whole SGE system is *QMON*. QMON provides all the possible tasks available inside an SGE grid system, from job scheduling and management to queue specifying.

This software gather all the command line tools inside a single user interface that permits to manage all the processes according to the user permissions. From the point of view of the grid administrator, to have such a common interface to deal with the entire structure could be very useful and in much cases mandatory when a frequent queue and resource configuration changing is needed.



The QMON interface

7 Conclusions

Studying the Sun™Grid Engine infrastructure, the first thing that is easy to recognize is the hard relationship that binds SGE to a familiar UN*X environment. The structure of the system is quite simple and it often uses already existed and trained technologies without trying to provide complex features and with the constant idea to keep the things simple.

This solution can appear sometimes rude and limited, but it probably fits a great amount of needs, in particular in the research fields, where a relative small amount of people, such as the students of a department or people involved in an experiment, have to share and manage a flexible grid infrastructure for both parallel and non-parallel tasks.

Obviously, as reported at the beginning, this kind of grid management system is suitable only for relative small grids used for similar or at least well-known tasks and it cannot be the right idea for very huge grid structures that needs the maximum flexibility and where the profile of scheduled jobs is totally unknown a priori.

8 References

[1] Sun™Microsystem inc., *Sun™ONE Grid Engine Administration and User's Guide*. October 2002

[2] Sun™Microsystem inc., *Sun™ONE Grid Engine 5.3 Release Notes*. April 2002

[3] Tutorials and howtos, <http://gridengine.sunsource.net/documentation.html>.