SCHEDULER CONFIGURATION FOR MULTI SITE RESOURCE BROKERING

L. Hajdu, J. Lauret
STAR, Brookhaven National Laboratory, Upton, NY 11973-5000

Abstract:

In the distributed computing world of heterogeneity, sites may have from the bare minimum Globus package available to a plethora of advanced services. Moreover, sites may have restrictions and limitations which need to be understood by resource brokers and planner in order to take the best advantage of resource and computing cycles. Unfortunately, such information is usually not provided by information services or high level Grid services making the full exploitation of optimal resources far from reach. In this paper, we will discuss the evolution of the layout of the configuration of the STAR Unified Meta Scheduler (SUMS)[6] and will show how the approach allows to take full advantage of any available site as well as local resources in a transparent and "*universal*" manner.

INTRODUCTION

In the distributed computing environment, one of the most challenging tasks is the represent the rich set of resources and capabilities of each and all sites into a consistent picture so a high level service could take into account these information for an optimal decision making outcome. For a resource broker or Meta-Scheduler (i.e. a wrapper around multiple local or distributed resource management systems), the challenge is to not only get this information in-situ, but also be able to provide ways to new sites to declare their resources and become part of the Virtual Organization reachable resources. Some of this information already resides in information services and information providers and schemas have been developed to represent the fine granularity of the computing element and storage element resources: The GLUE schema [5] for example has such detail information, the MonaLisa [1] retained historical information on resources and load or the GridCat [3] catalogue aims at providing a high level view of resource usage. None provide a consistent and simple set of information allowing however for a Meta-Scheduler to understand the relationship between computing resources, storage, network and services allowed to communicate in or out of the worker nodes, queue or pools and their mapping to those resources or even the multitude of choices when it comes to select one based on the user's need or privilege to use those resources (sometimes only accessible by specific VOs). While the configuration of a site is of the choice and responsibility of a the site's system administrator, local queue policies and site restrictions should be known as quickly and efficiently as possible to the scheduler.

To this effect, the earliest as possible strategy could be employed: in such approach, the configuration of the software itself has to represent to a usable granularity, available computing resources, methods to access these resources as well as methods to transmit in and out data sets used for processing or as product of the jobs. This should ideally happen without any user manual intervention regardless of accessible sites. While the target implementation presented here is carried within the SUMS framework (a meta-scheduler or wrapper providing access to both grid and local scheduler), such idea has been independently carried within other projects such as the SPHINX scheduler [2] or the MAUI/MOAB scheduler to some extent [9]. Our intent is to describe and explain the scheme allowing a site to declare information which combines into a consistent set of parameters usable by a high level resource broker. Such information is not specific to our Meta-Scheduler and could be later expanded to a resource information schema for wider use.

THE SUMS META-SCHEDULER

The STAR experiment, as many High Energy and Nuclear Physics experiment, tend to have their workload which could be qualified as embarrassingly parallel: no particular effort is needed to segment the problem into a large number of parallel independent tasks allowing harnessing the power of grids by "*dismantling*" large computing requests into subsections that can be apportioned to smaller computing elements within the grid. However, apportioning of computational load is a non trivial task as frivolous distributions of load can yield longer turnaround in contrast to using local resources. Strategies to overcome such

scenarios traditional require copious amount of user intervention on the part of grid production coordinators. Because of the heterogeneity that exist among the resources being shared, the "human" element takes care of mentally selecting a destination, accessing the resources, formatting the computing load appropriately for the destination, as well as selecting methods for retrieving the data whenever the unit of work is accomplished. For this reason, the STAR experiment engaged and developed a strategy by which all decision making process would be done into a high level layer named the STAR Unified Meta-Scheduler system (or SUMS). SUMS is aimed to replicate the human logic leading to such decision and encompass all component performing these functions. Its flexible design allows for multiple scheduling algorithm and decision making strategies as well as a universal grid or local scheduler "plug and play" interface.

The architecture of SUMS can be represented in three component architecture or layers. Each layer represents a virtual interface where one implementation and instance can be substituted with another that performs the same task by means of different algorithms. In other words, the information exchange between the layers is well organized into stable virtual objects. The most top layer, known as the "*initializer"*, parses, validates, and converts the users request to a generic form so that it can be passed to the second layer, known as the "*policy*". The policy layer is really the heart of SUMS as it resolves request for data sets (which could be available only from one and one only computing element) and map this request to the available resources which could be either local or grid resources, reserved or opportunistic. The policy also has the ability to fold decision making based on accounting information or historical queue behaviour like in Ref [10].

The request or task is consequently split into a subset of jobs which characteristics are optimal for the chosen policy. The final component is the dispatcher layer: its role is to receive and examine the information about a job and format it appropriately for each target scheduler and dispatch it. To be specific on its role, while one need to generate a script to be dispatched by a local resource manager such as LSF [11], one would need instead to generate a Condor job description [8] instead whenever using the Condor job dispatcher and so on. Usually, we use GRAM/DAG for grid dispatching but could equally use a submission Web-service by transforming the SUMS virtualized job into the appropriate physical representation.

POLICY'S INPUT: STRUCTURE OF SUMS' CONFIGURATION

Our initial implementation of the primary input for decision making was very pragmatic and only allowed submitting all jobs to one type of resource per submission. Predefined combinations of the policy and dispatcher blocks existed in the configuration. The submitter could select from the list only one of these structures to be used for submission of their whole request. The inability to submit to more than one location existed because one could only select one dispatcher block to which all jobs have to be sent for dispatching. Each policy block above the dispatcher would contain a list of homogeneous computing elements such as different queues in the same batch system at the same site. If the user wanted to split there request between two sites they would have to split it manually by submitting two requests.

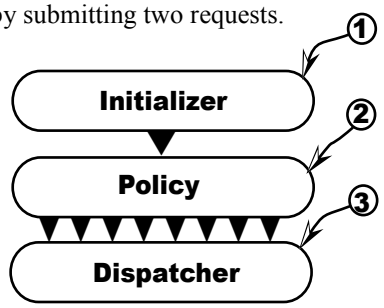

Fig 1

FIRST PHASE OF EVOLUTION

The first phase of evolution involved inserting a thin layer (Fig. 2) between the policy and the dispatcher layers. This additional layer allows heterogeneous computing resources to be inserted into the policy. The layer in-between de-multiplexes the stream the jobs coming through it and redirects each one to the appropriate dispatcher (Fig. 2) for the given target computing element of the job. This allows the formation of policies with heterogeneous site specific computing elements allowing one user request to be split between multiple site and batch systems. Policies could also be configured that split the computing load up between local resources and grids.

If users at siteA wish to submit to siteA a local dispatcher would be used (LSF, SGE, PBS, Condor). If users on siteA wanted to submit to siteB a grid dispatcher would have to be used (Condor-G, Web-services). However a user on siteB would not want to use a grid dispatcher for submitting to siteB thus there existed a need for different configuration files. While this initial

approach allowed to go far along the development and use of SUMS in STAR, this solution would not be scalable as the number of sites increases. If any site made a change it could not easily be applied because experts on each site had to adjusts there custom configurations.

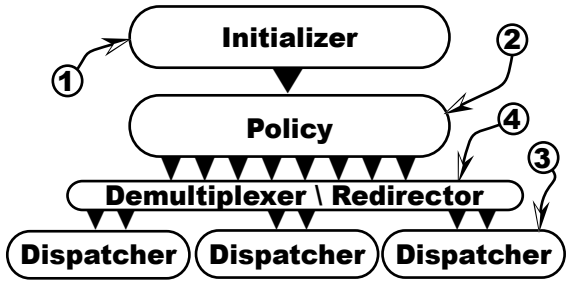

Fig. 2

SECOND PHASE OF EVOLUTION

In order to make one configuration file for all sites without redundancy the configuration was reshaped. A list of site containers (Fig. 3) is the root of the configuration. The site container encapsulates a representation of the resources of its site. Each site container contains a list of zero through N batch system objects that are accessible it the site. Each batch system object holds a list of queue objects. The queue object like all objects in the configuration is virtual this means queues also represent batch system pools in the case of the condor batch system [8]. The queue objects hold information on the parameters of the queue. Such as allowed run time, memory, and swap space, the assumption being that all nodes represented by a queue are roughly identical. After a job has been assigned to a queue the de-multiplexer determines if this queue is locally accessible or if it resides on a different site. If the physical queue is on the same site the job is redirected to the local dispatcher. If the job is on a different site the job is passed to the grid dispatcher.

The user now selects only the policy and not the final dispatcher to be used unlike the previous configuration where the three layers where selected at initialization. , The policy is a subset of (grid and local) computing elements encapsulated by an algorithm that governs the distribution of jobs to the elements. If two users call the same policy at different sites this would have failed in the previous version of the configuration. In the current version submitting is always possible because the policy is no longer configured as grid or local, dispatchers are available for both types of local submissions. Modifications to the configuration are replicated at all sites providing more current versions of the configuration by automated or manual configuration file replacement.

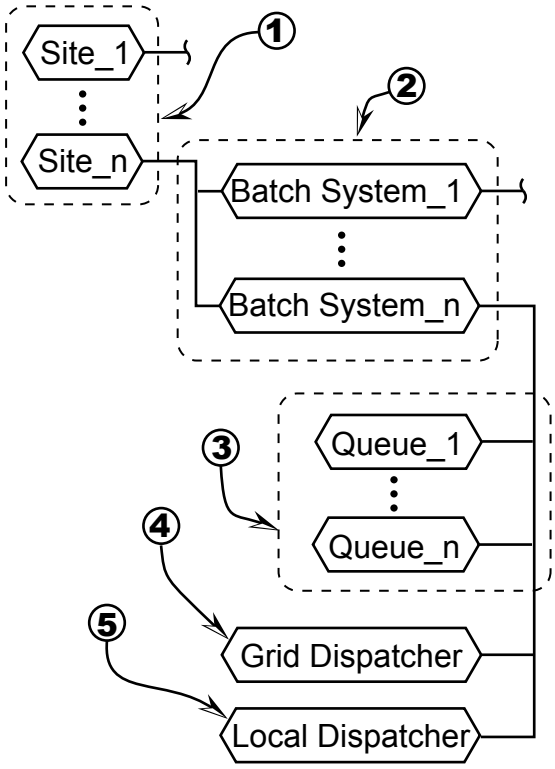

Fig. 3

Each site object was also given parameters to allow the resolution of site specific nuances. For example some sites require the loading of modules or setting of paths that the user has grown accustom to having loaded by default at logon at the users home site. These nuances are typically not available in information services. However they are necessary to insure that the user's job can run on all sites.

FUTURE PLANES OF EVOLUTION

Our current configuration approach meets the needs for all STAR sites. Future work enhancing our approach includes access to dynamic configuration by recovering the information from a distributed service but even before that, a deployment and support for more than one VO access to resources. In fact, our next version will take the concept of two access methods (grid and local) and expand it into one or more access methods for each VO within a single configuration schema.

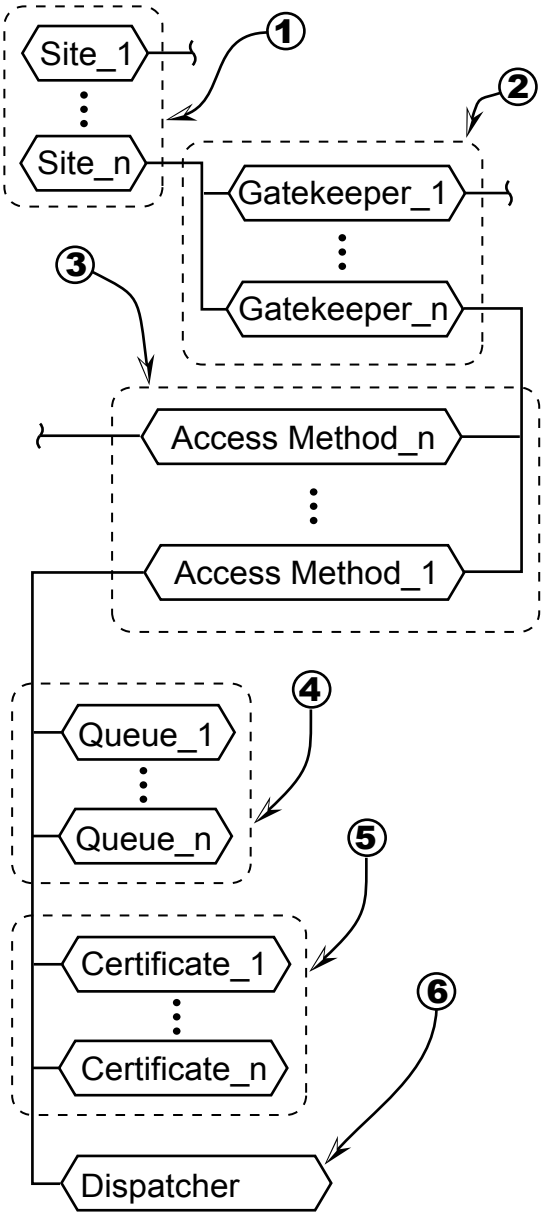

Fig. 4

In this evolutionary step we take into account that one gatekeeper can allow access to zero or more queues and this access can change based on the users grid certificate or credentials. It is taken into account that one gatekeeper may be accessed via many access methods. For example one gatekeeper may allow job to be submitted via web services and globus RSL.

The structure of such a configuration is shown in Fig. 4. The top root element is a list containing sites. Each site container holds a list of gatekeepers, available at the site. Each gatekeeper container holds a list of access methods that may be used to submit jobs through that gatekeeper. Each access method contains a list of queues accessible via that method. The access method container also contains a list of certificates or roles, one of which must be held by the submitter if they wish to submit to that gatekeeper via that access method. Finally each access method must hold one dispatcher object which is used by the scheduler itself to take care of the mechanics of: submitting the job, file staging and recovering produced data.

There will still be a policy which holds a list of queues but when the algorithm selects one of these queues, it will traverse the portion of the tree holding the queues objects of the configuration and locate the occurrences of the queue within the tree. Then it will drop all of the access methods for witch the user does not have a certificate. If no access methods remain the next queue will be selected and the process will be repeated until an access method to a gatekeeper on a site can be found where the user can submit the job. The access method can be quickly tested to verify that the connection is up. Taking data about the state of the connection from information services is also useful for this task however the information is not as up-to-dateas one would like. The access method is also drop if the connection via that method is not functional.

REFERENCES

[1] The MonaLisa monitoring system http://monalisa.caltech.edu/

[2]...http://sphinx.phys.ufl.edu/

[3]...http://www.ivdgl.org/gridcat/

[4] http://www.cnaf.infn.it/~andreozzi/datatag/glue/

[5] http://tyne.dl.ac.uk/StarterKit/gt2/mds/gluesch ema.html

[6] http://www.star.bnl.gov/STAR/comp/Grid/sch eduler/

[7] http://java.sun.com/products/jfc/tsc/articles/pe rsistence4/

[8] http://www.cs.wisc.edu/condor/

[9] http://www.clusterresources.com/pages/produ cts/maui-cluster-scheduler.php

[10].Development and use of MonALISA high level monitoring services for the star unified Meta-Scheduler, CHEP04 proceedings, http://indico.cern.ch/contributionDisplay.py?c ontribId=393&sessionId=7&confId=0

[11] http://www.platform.com

[12] I.Foster “The Anatomy of the Grid: Enabling Scalable Virtual Organizations” Supercomputer Applications, 15(3), 2001