The Instant Glidein

A generic approach for the late-binding of jobs to various resource types

High-Throughput Computing

A batch system enables computational tasks (jobs) to be executed on computing resources that are physically separated by a network, without the job submitter considering either the structure or load of the system. The algorithms that the system implements optimises the utilisation of those resources with respect to some selected metrics through the assignment of jobs to resources. As the demand on those resources from multiple users is a stochastic property for both the arrival and execution times , this leads to the establishment of queues of waiting jobs which can undergo dynamic and continuous reconsideration of priorities.

In an environment comprising of geographically-separated computing resources, each is typically under the control of a Local Resource Management System (LRMS), a batch system. Metaschedulers have been developed to optimise the utilization of those computing resources based on state information provided by the LRMS.

As the number of independent jobs submitted by user increases, the focus shifts from the performance of an individual job to the throughput over time of many jobs. This has been defined as High-Throughput Computing and suites applications that can be broken down into independent jobs, which can be individually scheduled across the different computing resources.

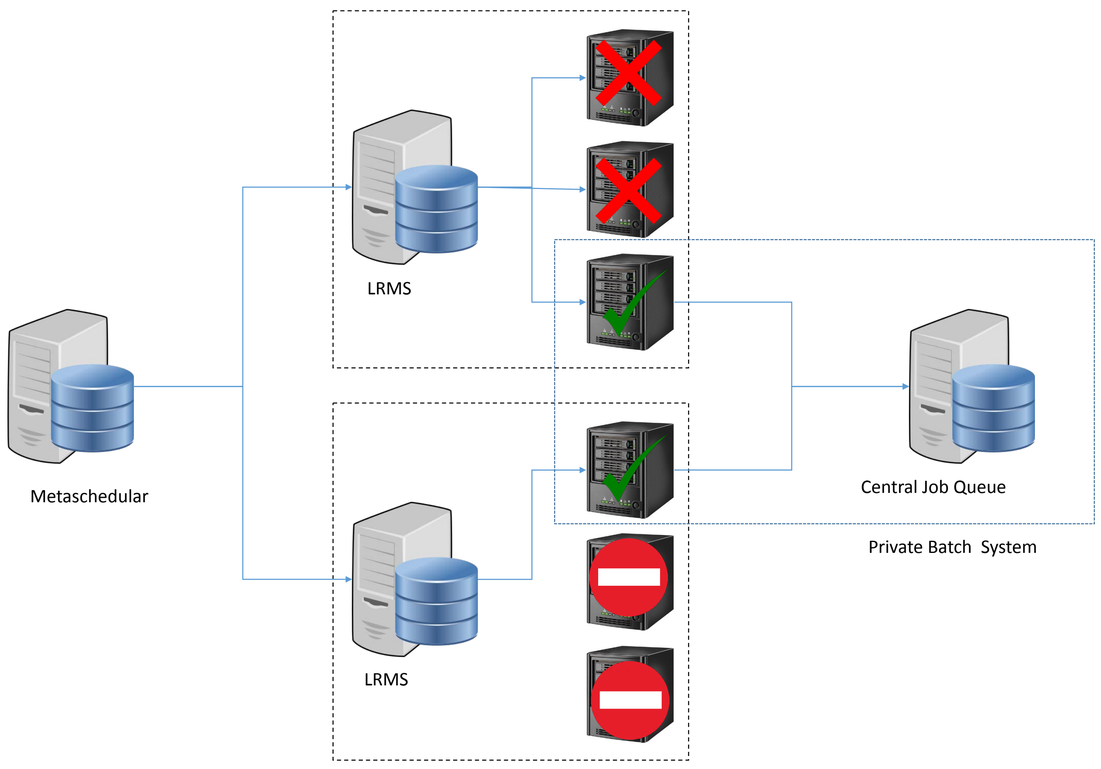

Figure 1: Using the late binding with metascheulars to create a private batch system

Late Binding

One of the issues with the metascheduling approach is that the decision of where the job is to be executed is made quite in advance. Fundamentally this means job assignment does not take into consideration the latest state such as the arrival of higher-priority jobs. In addition the final execution environment is not known so even if the job starts, it may not complete successfully and hence adversely affects the turn around time as that job has to be rescheduled.

A solution to this is the concept of late-binding in which a place-holder job is used as resource reservation agent that when executing on the computing resources requests the most appropriate real job from a central job queue. This approach allows scheduling decisions to be delayed until the last possible moment and hence can take into consideration all information that is available at the time of the request. These agents can also perform basic sanity checks of the environment to avoid jobs being requested if the resource is not suitable. The result is that a private or application specific batch system is overlaid on the original infrastructure and is shown in Figure 1. A lesson learned from the experience of metascheduling is that resource allocation should be decoupled from job execution and the order respected.

An implementation of the late-binding approach is the HTCondor Glidein. They are placeholder jobs that are submitted using the standard HTCondor tooling which start a HTCondor daemon process. The result is a private HTCondor pool created out of dispersed computing resources by *gliding-in* HTCondor daemons.

Dynamic Computing Resources

The emergence of Virtual Machines delivered by Infrastructure as a Service (IaaS) has brought a new dimension to HTC. Computing resources can now be dynamically provisioned on demand using *resource leases* which are typically defined in terms of wall time when using a pay-as-you-go model. A potential benefit of this approach is that if short-term leases are used and the resources are recycled, then system can become self-healing. As the potential capacity available from large-scale commercial providers can be significantly greater than the needs of the applications, this gives the appearance of infinite capacity. This would raise the question of whether queues and job scheduling are still required as all jobs could immediately be assigned a vacant resource. However, in reality the available capacity is finite due to other factors such as budget and price. The result is that the utilization of those available resources needs to be optimized with respect to some selected metrics and hence the algorithms that are implemented by batch systems are still required. The late-binding approach is appropriate in this scenario to ensure that job execution occurs after the resource has been allocated. The solution is therefore to overlay a batch system upon those dynamically provisioned computing resources. The remaining question is how to manage the lifecycle of those dynamic computing resources. The VAC model is where the lifecyce management occurs transparently for the user or application. The alternative is where the lifecycle management is done via the user or application themselves. We will define this model as Throughput Amplification by the Stimulated Emission of Resources (TASER). Figure 2 shows how the late-binding approach can be used with either the VAC or TASER model for dynamic resources.

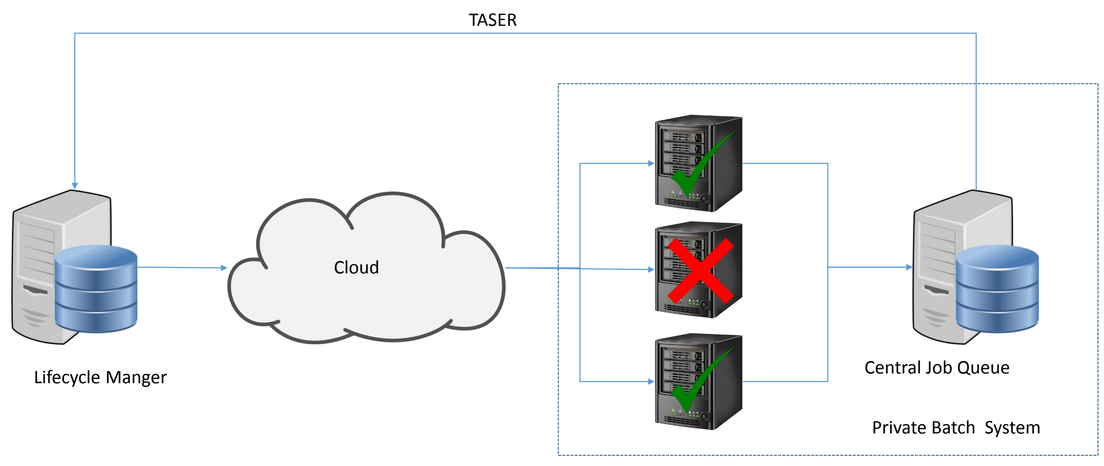

Figure 2: The use of late binding with IaaS, showing the feedback loop of the TASER model

The Instant Glidein

While provisioning the HTCondor Glidein for dynamic computing resources has proved to be successful, it can still be optimized. The original Glidein was designed to be pushed as a job via a batch system to unknown heterogeneous resources. Hence a significant portion of the code handles the installation of HTCondor in user-space for many different platforms. It also has to handle differences in the configuration for HTCondor versions, how these interact with the platforms and what is required for the application. Dynamically provisioning resources can reduce some of this diversity and results in a more homogeneous environment. By using a common VM image, the platform and also version of HTCondor can be known in advance. If a custom image is used, HTCondor can also be installed as part of the operating systems and hence removes the need to install in user-space, further reducing what is required.

The result is that only customizing the configuration and starting HTCondor to run as a short term lease is still required. We will define this shorter recipe as the *Instant Glidein*.

The CernVM image has been selected as the base platform for this work. It is a small (17MB) image that delivers it's contents just in time by means of the CernVM File System. The advantage of using CernVM for the Instant Glidein is that HTCondor is already available. HTCondor can behave as a short-term lease by providing as START expression so that only new jobs will be fetched if the lease is not ending. The STARTD_NOCLAIM_SHUTDOWN period will then cause the startd to terminate and the MASTER.DAEMON_SHUTDOWN expression will ensure that all HTCondor processes terminate. The main process (*condor_master*) is forked and the process that executed the fork waits for the HTCondor process to terminate. On termination the machine is shutdown and the lifecycle manager can then watch for virtual machines in the shutdown state and clean them up before new virtual machines are regenerated.

The Instant Glidein is being used to provide a common platform for LHC@home where BOINC is use as the lifecycle manager.

Laurence Field - CERN
Iain Steers - CERN