iLab Scheduling - Overview

Authors: Jud Harward, Philip Bailey

Last Modified By: Philip Bailey

Last Modified Date: 26~~25~~ January, 2010

Introduction

The interactive architecture permits students to observe the progress of the experiment and to interact with the experiment in ways that can change the experiment's course. Such labs typically require more time to execute than batched experiments because they proceed in human not machine time. A typical interactive experiment requires 20 minutes to several hours to execute. Because users control the lab equipment, they usually require exclusive access to it. Asking users to queue for their turn to use the lab wastes their time.

Hence most interactive experiments require a scheduling application that allows the users to sign up in advance for time on a particular piece of lab equipment. The scheduling application must also notify users if their reservation must be cancelled or changed. Finally certain labs have operating requirements that require actions either before or after the execution of an experiment. For instance, a heat exchanger experiment may require the apparatus to achieve thermal equilibrium before the start of an experiment. A chemical diffusion experiment may require that the diffusion apparatus be flushed at the end of the experiment. The scheduling application must allocate time for these actions in scheduling experiment sessions.

Scheduling can be looked at from two perspectives. From the lab provider's perspective, the scheduling application coordinates reservations to use a lab from multiple campuses. The scheduling server is also the process that holds the information required to "wake up" a lab server to perform required actions before a scheduled experiment. The lab provider may want to allocate blocks or percentages of time to different groups of users according to certain policies. For instance, Lab A may want to permit use from universities B and C between 6 pm and midnight, Monday through Friday, with university B allocated 60% of the time and university C the remainder. On the other hand, the lab provider generally does not want to be aware of the details of a user's reservation. If the lab server must be taken down for maintenance, the lab provider would simply like to notify the scheduling application of the down time and have the scheduling application take care of informing the affected users and rescheduling their work.

From a teacher's and a student's perspective, the scheduling application must act as their agent in scheduling time on lab servers. The application must accept authorizations to schedule from the users' SB and must record reservations in a way that can be associated with individual users. If a reservation must be cancelled, the scheduling application must take the responsibility for informing the user. Teachers may want to stipulate policies that govern how their students may make reservations. For instance, a teacher may decide that students can only sign up for two hours of lab access per week with no single reservation lasting more than one hour. Different teachers using the same lab may want to set different policies for their students.

Given the different requirements from the lab-side and the student-side perspectives, where should the scheduling application be located? The need to coordinate reservations from multiple campuses for a single lab server argues that there should be a single scheduling application located close in network terms to the lab server. But the requirement to accommodate the different policies of individual teachers suggests the need for multiple scheduling applications, at least one on each student campus like batched service brokers. Simplicity suggests that there should be only one Scheduling Server, but in fact we believe the design will be easier to implement, evolve and maintain if there are complementary lab-side and student-side scheduling servers. We have decided that the two perspectives require two related scheduling applications, a Lab-Side Scheduling Server (LSS) and a User-Side Scheduling Server (USS).

Segregation of Information and Policy

The division of scheduling functionality between two servers will lead to unnecessary complexity if they are tightly coupled. The twin design therefore requires a clear separation of the LSS and USS data models. Both servers will support policy or rule sets to determine whether a proposed reservation can be scheduled. But those rule sets must not depend on each other and each rule set must be decidable based on the information available on its system’s data model.

The LSS looks at scheduling from the lab provider’s point of view. It coordinates reservations from multiple universities, and it allows a lab provider to allocate time between these universities. This allocation process is governed by LSS policy. It may involve as simple a mechanism as partitioning the available time in non-overlapping blocks, each of which is allocate to a separate university. More complicated mechanisms might allow one campus to reserve time one week ahead while another might only be able to reserve time a single day ahead. But LSS policy should not involve any decisions based on user identity.

All user-based policy, for instance user reservation quotas, should be implemented by USS policy and supported by the USS data model. If lab server maintenance requires the cancellation of previously approved reservations, the revocation of reservations will be initiated on the LSS, but notification of the users must be carried out on the USS. In general, faculty will not need accounts on an LSS (unless they are the lab owner) and lab providers will not need accounts on USSs. The web service API between the LSS and USS should not have to communicate policy, only reservation information.

Cardinality

A LSS may serve more than one lab server, but each lab server will schedule through a single LSS. One could imagine a university like MIT supporting one lab-side Scheduling Server for all online labs housed on campus and a corresponding student- or user- side server to schedule lab sessions for all MIT students regardless of where the labs were located.

A USS server may allow users to make reservations on more than one lab server. In fact, it may communicate with multiple LSSs. It may also serve students from more than one campus or Service Broker domain. A Service Broker may also employ more than one USS. So to summarize,

- the relation between Service Brokers and USSs is many to many;
- the relation between USSs and LSSs is many to many;
- the relation between LSS and lab servers is 1 to many.

Closely related to cardinality is the set of keys that defines the instances of the main scheduling object types.

- An experiment is defined as the combination of a lab client and a lab server.
- A credential set is defined as the combination of a Service Broker and an effective group accessed through a USS.

Scheduling Concepts and Principles

Time

A student may make reservations to use a lab that is located in another time zone. The USS and/or the LSS through which the reservation is secured may themselves be in different time zones. As a general principle, any iLab application should display dates and time in the user's current culture and time frame. The most straightforward way to implement this is for all internal representations of time to be in UTC, but whenever displayed to the user the date and time should be formatted into the user's culture and current time zone. During the login process we determine the user's current time zone offset from UTC and culture. Methods defined in the DateUtil class are used to format and parse date-time strings.

An exception to this model is the decision to display time on the Lab Scheduling Server in the LSS's local machine time, independent of which time zone is current for the user or the lab server. LSS pages are only visible to users with administer or manager privileges.

TimeBlock – Scheduling consists of managing available TimeBlocks, properties of a TimeBlock are Start, End and Duration, Duration is in seconds.

TimePeriod – Extends TimeBlock with the additional property quantum, Quantum is the recommended boundary for splitting a time period into TimeBlocks represented in minutes. The quantum is used both in the display of available time and to minimize the fragmentation of available time. If there is a request for a reservation starting now (2:33 PM) and the quantum is five minutes the shortest available reservation will be for two minutes ending at 2:35 PM and all reservation end times will be multiples of five minutes up to one more than the maximum reservation time.

Reservation – Extends TimeBlock with the addition of a user name property, this is a WebService data type and used primarily to send notifications from the USS to the ServiceBroker.

Each of the scheduling servers use different internal classes to represent reservations.

Managment

CredentialSet - A CredentialSet is used to define a group on the scheduling servers. Minimally it is a group name and a ServiceBroker GUID, on the LSS it also contains the GUID of the USS that manages the group.

ExperimentInfo – An experiment is defined by the combination of client and lab server GUIDs, what additional information about the experiment is included depends on the type of scheduling service.

Rules – Rules are policy for scheduling, currently you may set minimum and maximum times for a reservation on the USS. The LSS defaults to a first-come-first-served rule if the group has permission to run the experiment at the requested time.

LSS Only

LabServerResource – Each lab server must have at least one lsResource. A resource models a single thread of hardware access. It is possible to assign multiple clients to a single resource or if the lab server has multiple hardware instances to create multiple lsResources for the lab server.

Recurrence – A recurrence specifies a collection of TimeBlocks that span a number of dates and provide access to a single LabServerResource. Recurrences that share a LabServerResource may not overlap. Each recurrence may be assigned multiple permitted experiments and credentialSets. Recurrences have a type, start date, end date, start time, end time, day mask, and quantum. Depending on the recurrence’s type the collection of TimeBlocks generated range from a single TimeBlock to ones that are specific to days of the week.

Lab-Side Scheduling Server (LSS)

Each lab that requires scheduling must have one and only one Lab-Side scheduling server, but an LSS may manage multiple LabServers. The role of the LSS is to maintain all reservations for each LabServerResource

normally the two services are within the same domain.

RecurringTimeBlock

User-Side Scheduling Server (USS

The Scheduling Process)

Scheduling Servers

The scheduling process will involve multiple constituencies:

- Lab server providers will want to register and administer the blocks of time when their lab service is available. They may need to cancel previously scheduled lab sessions for unexpected maintenance.
- Teaching faculty may want to monitor the signups of their students and set policies under which students can reserve lab sessions.
- Students will want to make reservations for future lab sessions, check on their previously made reservations, and on occasion change or cancel these reservations.

The first constituency is lab-side, the other two student-side. Scheduling functionality can be located on either side or both.

If a lab provider wishes to allow two separate Service Broker domains to make reservations in the same block of time, then there should be a lab-side scheduling service to coordinate reservations between the two campuses. If all the scheduling functionality is located on the lab server side, then the lab-side Scheduling Server must be aware of the identity of individual students and must assume responsibility for implementing the sign-up policies of the faculty at different universities. We, therefore, envision two Scheduling Servers, one lab-side and one student- or user-side, each with its own responsibilities and functionality.

Lab-Side Scheduling Functionality

The lab-side Scheduling Server (LSS) must provide a web application that will allow lab administrators to control the scheduling of the usage of their lab servers. The basic actions that must be supported are as follows:

- A lab administrator must be able to register a user-side scheduling server (USS) on an LSS so that user reservations forwarded from that USS will be honored if they correspond to an available period and the reservation does not violate a lab-side policy rule.
- A lab administrator must be able to offer a block of time for reservations to one or more USSs. The administrator may restrict reservations to credential sets (effective groups and originating Service Broker) from each USS. The usage of each combination of experiment and credential set~~, USS, and effective group~~ may be governed by a separate lab-side scheduling policy.
- A lab administrator must be able to specify rule sets (policies) to determine whether a reservation from a particular USS for a particular time should be accepted or not.
- An administrator should be able to revoke previously confirmed reservations to accommodate maintenance or other unexpected requirements of the lab server team. The USS should handle notifying the students and faculty affected by the cancellation.

The LSS web service will be invoked only by USSs and must provide the following functionality:

- It must be able to provide a listing of available reservation times for a given experiment and credential set within a particular USS specified interval.
- It must be able to confirm or deny a particular reservation request from a USS; if the request is denied, the LSS should provide a brief explanation string.

Finally the LSS must be able to invoke an alert method built into the interactive lab server web service. The purpose of this method is to wake up a lab server that needs to prepare for an upcoming scheduled reservation. For example, if a heat exchanger experiment requires certain initial conditions, then the apparatus may actually have to be started a significant period before the user logs on to execute the experiment.

User-Side Scheduling Functionality

The student – or user-side Scheduling Server (USS) must provide a web application that will allow users to make reservations to execute experiments on lab servers and will allow faculty to manage and observe those reservations. The basic actions that must be supported are as follows:

- A user must be able to make reservations to use permitted experiments as well as to view, cancel or modify previously scheduled reservations.
- A staff member should be able to review reservations made with a credential set (for a class) with which the staff member is associated. If the staff member has the required permission, he or she should be able to make, modify, or cancel reservations for members of the group over which the staff member has authority.
- A faculty or staff member should be able to specify a rule set (a policy) that governs whether a reservation request to execute an experiment at a certain time will be accepted from a student with a particular credential set. If the student's reservation request is rejected, the student should be given some indication of why the reservation was rejected, e.g., the student does not have access, the student has exceeded his or her time quota, or the reservation is too short to perform the specified experiment.
- The USS web service must allow an LSS to revoke a block of reservation (e.g., due to maintenance) and assume the responsibility for informing the affected staff and users.