Integrating Reservations and Queuing in Remote Laboratory Scheduling

David Lowe, *Senior Member*, *IEEE*

Abstract—Remote laboratories (RLs) have become increasingly seen as a useful tool in supporting flexible shared access to scarce laboratory resources. An important element in supporting shared access is coordinating the scheduling of the laboratory usage. Optimized scheduling can significantly decrease access waiting times and improve the utilization level of RL resources, with associated reductions in per-use costs. Current RL systems have typically supported scheduling based on either reservations or queuing, though rarely both. In this paper, we investigate issues that arise when a single RL resource (or pool of resources) supports both modes for gaining access, and how these two approaches can be effectively integrated. This research analyzes the scheduling algorithm utilized by the Sahara RL system to investigate any limitations that affect the system utilization. We identify a number of current issues and propose specific modifications to address them. The proposed modifications will lead to increased utilization and improved student experiences.

Index Terms—Computer-assisted instruction, education, physical sciences and engineering, scheduling

1 Introduction

Laboratories are important tools for supporting both student learning and scientific research. They do, however, represent a significant financial and logistical investment that can be difficult to develop and maintain [1]. Traditionally, the requirement for physical access to the laboratory apparatus has limited the ability to support both flexible usage and sharing of facilities. However, rapid evolution of both computer technologies and near-ubiquitous networking supported the emergence of remote laboratories (RLs) from the mid-1990s. RLs have subsequently become increasingly sophisticated with a growing level of deployment. Various benefits have been shown to arise from their use, including flexibility of access [2], the ability to share resources and labs [3], [4], [5], security of users, data, and devices [6], among many other benefits.

The current generation of remote laboratories can be classified into two main forms: batch laboratories and interactive laboratories. Batch laboratories are based around users submitting a specification of the experimental operations, and this experiment is then placed in a queue and executed asynchronously as a "batch" task once the apparatus is available [3]. The results are recorded and retrieved by the user at a later time. In this scenario, the user does not interact at all with the experimental operation while it is executing and the overall utilization level of the apparatus is constrained only by the algorithms that process the queue.

In contrast to batch laboratories, the more prevalent interactive laboratories allow users to synchronously monitor and adjust the experiment as it executes. This has the obvious implication that the user must be allocated to a laboratory resource at a time when it is available. This allocation process represents a relatively conventional resource scheduling task, though the nature of the laboratory infrastructure and the way in which it is used provides some interesting opportunities and challenges. The specific design of the laboratory scheduling will depend on a number of factors, including: the number of users; the number of available laboratory rigs; the typical duration (and variability) of usage; and the access guarantees provided to users. The resultant design of the scheduling algorithms can have a significant impact on both the user experience (e.g., how long they must wait for access and how this access is then managed) and the maximum level of utilization that can be achieved (and hence the amortized cost per user).

Current approaches to scheduling are typically based on either reservations or queuing [7], each of which has advantages and shortcomings. A scheme that merges both these approaches successfully can potentially benefit from the advantages of both and limit the disadvantages. In this paper, we consider the scheduling scheme incorporated within the current release of the Sahara RL system [8]—an approach that merges reservations and queuing. We show that unintended side-effects can emerge from the interplay between the two schemes, resulting in significant performance problems. We then propose several modifications and show that these are not only able to address the problems but can result in significant overall performance enhancements.

The remainder of the paper is organized as follows: In Section 2, we review existing research on resource scheduling in the context of RLs. In Section 3, we introduce the Sahara Remote Laboratory System and consider the scheduling system that it implements. Section 4 provides a case study of Sahara scheduling using live usage data that illustrate the interplay between scheduling and queuing and how this can inadvertently lead to performance degradation. In Section 5, we propose alternatives and then

• *The author is with the School of Information Technologies, The University of Sydney, Building J12, Camperdown, NSW 2006, Australia. E-mail: david.lowe@sydney.edu.au.*

Manuscript received 14 May 2012; revised 18 Oct. 2012; accepted 9 Dec. 2012; published online 15 Jan. 2013.
For information on obtaining reprints of this article, please send e-mail to: lt@computer.org, and reference IEEECS Log Number TLT-2012-05-0070.
Digital Object Identifier no. 10.1109/TLT.2013.5.

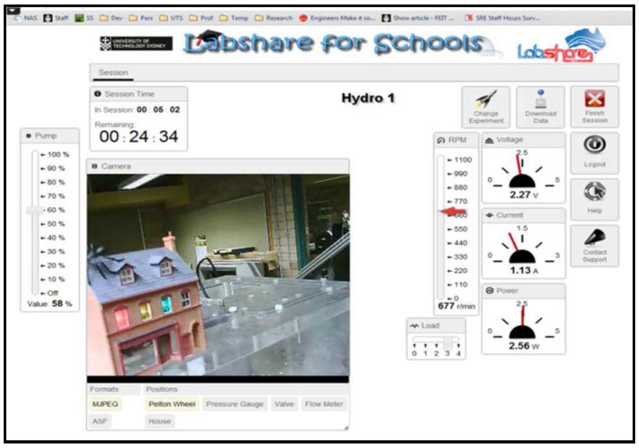


Fig. 1. A typical RL system: The UTS hydroelectric apparatus allows students to explore hydroelectric power generation using a system containing water tanks, a pelton wheel and turbine, and a variable load.

evaluate the performance improvements that can be obtained through adaptation of the scheduling algorithms. Finally, in Section 6, we provide conclusions and discuss future work.

2 Background

Access to appropriate and diverse laboratory apparatus has long been recognized as an important enabler in both education and research across a diverse range of science and engineering disciplines [1], [9], [10]. The recent emergence of RLs provides an additional approach to supporting this access. RLs provide remote access, typically across the internet, to physical laboratory apparatus that has been appropriately instrumented so that the equipment can be controlled and monitored by the user. A typical example is given in Fig. 1.

RL research has considered both the supporting technologies [2], [11], [12] and the learning pedagogies [13], [14] that underpin their use. Of particular significance is research that has compared learning outcomes between different laboratory modalities. Studies containing detailed experimental analysis on student learning outcomes have shown that while overall learning is still achieved students' performances on different criteria can vary significantly depending upon the form of access used [15], [16]. Indeed some outcomes appear to be enhanced by non-hands-on access modes (e.g., data analysis), while others seem to be degraded (e.g., identification of assumptions). The overall conclusion has been that RLs do indeed have a distinct role to play in supporting science and engineering education provided that their use takes into account the intended educational outcomes of the laboratory experience [17].

Having accepted the educational benefits that can arise from the use of RLs, consideration can shift to the logistical benefits. Indeed, it is often the logistical benefits that are used to motivate work on RLs (see, e.g., [15], [18], [19]). Two of the key benefits that have been regularly identified are the opportunities for flexible access and for laboratory sharing. To illustrate the ubiquity of this argument, a random selection of 50 RL papers from 2003-2012 (27 journal papers, 23 conference papers) was analyzed. Of the 23 papers published during 2003-2007, 61 percent (14 papers) mentioned the sharing of resources as a significant factor. Of the 27 papers published during 2008-2012, this had increased to 81 percent (22 papers) mentioning sharing as a factor.

There have also been numerous projects focused explicitly on facilitating sharing of RLs. Examples include, but are not limited to: the NSF-funded World Wide Student Laboratory project (originating in the mid-1990s, though dormant since 2007) [20]; LearNet [21], ProLearn [2], and PEMCWebLab [22] (all run during the early to mid-2000s and involving consortium of universities focused on sharing laboratories); the iLabs project [3] which focused, in part, on sharing laboratory resources with disadvantaged groups; the library of labs (LiLa) project [5] that created a RL access portal; and the Labshare project [4], [23], an Australian-based initiative that explicitly aimed to promote cross-institutional sharing of RLs.

Despite this substantial emphasis on the potential for shared access as a key motivating factor, there has been little research on the factors that influence this shared access, and more explicitly on understanding the capacity of a given pool of laboratory resources. This capacity will be highly dependent upon how the RL system manages requests from users for access to the apparatus and this is in turn dependent upon the number of rigs in the apparatus pool,[1] the number of users, and the usage patterns.

A typical scenario might involve a RL system that manages a pool of interchangeable rigs. A student will log into the system and be authorized to access this pool—often with a set of access conditions that are dependent upon which class of users they belong to (such as the default and maximum access times, their priority of access, and the time periods during which they are allowed access). The student might then request access to a rig, upon which the RL system will determine an appropriate scheduling of that access and which rig within the pool can be allocated. In essence, this is a conventional resource scheduling problem—albeit one which has a specific set of constraints.

The problem of resource scheduling is very common, extending across areas as diverse as communication bandwidth allocation, parallel processing, transportation timetabling, project management, operations management, production scheduling, and so on. Solutions to these problems are, however, very diverse. It has generally been acknowledged that "*the scheduling problem has so many dimensions that it has no accepted taxonomy*" [24].

Where resource demand and resource availability is predictable, static scheduling can often be used to identify an optimal allocation of resources [25]. Unfortunately, with RLs, both the resource demand (i.e., request for access to a rig) and the resource availability (i.e., rigs that are currently available for use) are typically less predictable: Lab demand varies due to the inherent variability in both when students may desire access and for how long that access may be needed; and lab availability varies due to both fluctuations in how long a rig is used as well as maintenance and related issues which might remove a rig from service.

To date, the scheduling approaches that have been used in operational RL systems have been either nonexistent or relatively simplistic. Many RLs are stand-alone rigs with direct access by users (essentially on a "first-come, first-served" basis) with no scheduling capability at all. Those

1. Each discrete item of experimental apparatus is often referred to, within the RL literature, as a *rig* and a collection of functionally identical rigs is referred to as a *pool*.

extant systems that do support some form of scheduling have typically either adopted a simple reservation system or a simple queuing system. An overview of many of the key schemes used by existing laboratories is given in [7].

Reservation (or booking) systems typically allow a user to reserve an access time slot in advance. This approach has been the dominant paradigm and was used in the earliest RL scheduling systems (see, e.g., [26]). Over the last decade, the underlying functionality has remained very similar but the architecture has become more sophisticated and support additional ancillary operations. For example, the reservation system implemented within the iLab architecture [3] includes a pair of services that together support user reservation of rigs (the lab-side scheduling service that coordinates access to the laboratory, and the user-side scheduling service that maps a reservation to a particular user, among other associated functionality). The scheduling service also provides ancillary functionality such as notification of canceled reservations due, for example, to unscheduled rig downtime. Other variations of reservation systems are also possible. The BTH Security Lab [27] uses shared resource reservations, where the user specifies a set of resources required and the system identifies times when all components of this set might be available.

A reservation system has the advantage of providing the user with guaranteed access at a known time. It has the disadvantage that typically a time slot must be reserved that is at least as long as the maximum time allowed—and hence there is likely to be significant proportions of idle time when users exit the system early.

With queuing systems, when a user makes a request, they are added to a queue. Each time a rig becomes available, it is allocated to the user at the front of the queue. This approach was adopted by early versions of Sahara [28], which also provided support for priority users to be able to "jump" the queue, the ability for a user to queue for either a specified rig or the first available rig from a pool, and the option for a user to extend a laboratory session if no one was waiting in the queue. Queuing systems have the advantage that as soon as a rig becomes available it can be reallocated to a waiting user (rather than having to wait until the next reservation), thereby maximizing utilization. It has the disadvantage that users do not have a guaranteed time at which they have access.

Reservation systems are typically more commonly used when there is only a single rig available (and hence queuing may lead to extensive delays). Conversely, a queuing system is potentially more effective when there is a pool of interchangeable rigs that can be allocated.

While few other approaches to scheduling have been demonstrated in existing operational RLs, a number of alternatives have been explored within the research literature, often drawing from techniques in related domains. As an example, Wieder et al. [29] treat RLs as another grid resource, and so argue that existing tools developed for this domain, including the use of scheduling services and service level agreements, can be used. They do not however discuss how this might be implemented nor does there appear to have been any implementation or evaluation.

A number of researchers and developers have also started to explore hybrid approaches, often blending reservation and queuing schemes. One of the first to consider this approach was Li et al. [30], who discussed a system that was based on reservations, but allowed queued users to gain access during periods when there were no reservations. They did not, however, describe an implementation or evaluation. In more recent work, the Sahara system incorporated a similar blended approach. This will be the subject of detailed analysis in the following sections.

Maiti [31], [32] has also described a hybrid (or "mixed-mode") approach, referred to as "slotted queuing." This approach involves allowing a limited number of users to make a reservation for the same time window, and then those users queue for use within that window. A prototype implementation is reported but no evaluation of the effect on performance is yet available.

Finally, it is also useful to note that the degree of coupling between the laboratory and the scheduling system varies enormously. Many RLs—particularly those which include a single rig—have the scheduling mechanism integrated directly with the software systems that manage the hardware. This is the case, for example, with VISIR—a widely used electronics testbed RL [33]. Other systems, such as Sahara [28], iLabs [3], and WebLab-Deusto [34], extract the scheduling system into a separate "RL management system (RLMS)" which is capable of coordinating multiple pools of rigs. This approach avoids the need to reimplement the scheduling system for each new rig. It also potentially allows for more complex coordination of multiple resources (such as offering a user an alternative rig if the requested rig is not available).

Approaches such as the LiLa Booking system [35] use a "metascheduling" system that supports access to diverse laboratories managed through the central portal. In the case of LiLa, this includes a ticket-based reservation service that is used both by users to make access reservations and by RL providers to check these reservations. While the LiLa Booking system provides a novel architecture, the underlying algorithms to allocate laboratories remains a simple calendar-based reservation system.

Another approach to handling of scheduling at a metalevel is to make use of preexisting functionality within learning management systems. A general discussion on this is provided by Bochicchio and Longo [36], and a specific example (using a Moodle extension) is given in [37].

Despite the emerging research on RL scheduling techniques, and the growing recognition of the importance of effective scheduling to the management of the scarce RL resources, there has not to date been a detailed analysis of how different approaches can affect the availability of the resources or the ease of access for users. Understanding this issue in more detail is an important step in improving RL flexibility and cost.

3 Scheduling in Sahara

In this section, we introduce Sahara—an RLMS developed as part of the Labshare project. We then analyze the current performance of Sahara with regard to its scheduling of access to RL resources. We have chosen Sahara as it is the only extant RL system that is in active use, supports a significant number of RLs, and is capable of supporting both queuing and reservation approaches, as well as combining them into a hybrid approach.

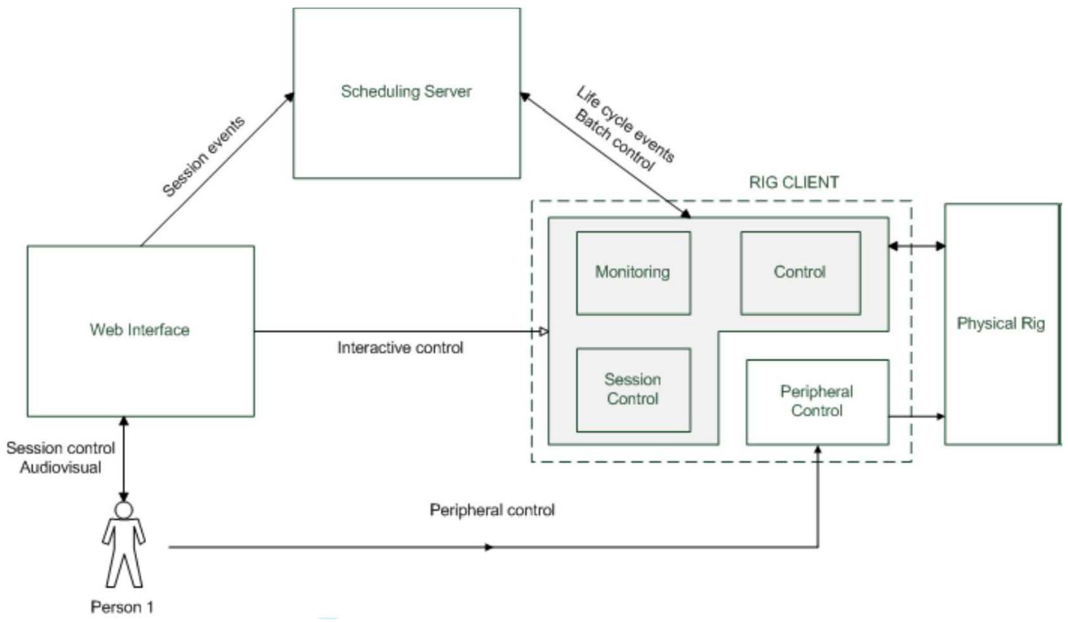


Fig. 2. The architecture of the Sahara RLMS (from the *Sahara Development Handbook*, available at http://sourceforge.net/projects/labshare-sahara/files/Documentation, by permission of The University of Technology, Sydney).

3.1 Background

The earliest RLMS of the University of Technology Sydney (UTS) was developed over the period 2000-2005. This system, which retrospectively has come to be referred to as Sahara release 1, was adopted as part of the much broader Labshare project. This project was a joint initiative of the universities belonging to the Australian Technology Network (UTS, Curtin, UniSA, RMIT, and QUT). Labshare aimed to "*establish a national approach to the sharing of remote laboratories that will provide higher quality remote laboratories with greater student flexibility, improved educational outcomes, improved financial sustainability, and enhanced scalability in terms of coping with high loads of students*."[2] To satisfy each of these goals, Sahara has undergone a major redesign from the ground up, producing new revisions of Sahara (release 2, 3, and onward). A final outcome of the Labshare project was the recent creation of The Labshare Institute as a not-for-profit organization that will be an independent service broker promoting, maintaining and even hosting RLs [4].

3.2 Architecture

The basic architecture of Sahara is shown in Fig. 2. This contains the following components:

- The web interface: through which users are authenticated and interacted with the broader system functionality, including selecting the rigs to which they wish to gain access.
- The scheduling server: is the middleware that manages the scheduling processes of the RL rigs, including tracking the state of rigs and assigning them to users. It is responsible for managing the running sessions according to the allocated times as well as logging all events and activities.
- The rig client: This component provides a software abstraction of each rig and converts abstract requests from the scheduling server into rig specific actions.

2. See http://www.labshare.edu.au/.

3.3 Scheduling

From release 3 onward, Sahara supports both queue-based and reservation-based scheduling.

3.3.1 Reservation (Calendar Booking)

This form of access request allows users to make a reservation that provides guaranteed access to a rig (or one of a pool of rigs) at a specified time. The resource allocations occur as follows:

1. A user attempts to log into the Sahara server, is authenticated, and is then provided with a list of the rigs (and pools of rigs) for which they have authorization.
2. The user selects a particular rig (or pool of rigs) and then is presented with the current status of the rig and the option to either make a reservation or to queue for access (see Fig. 3).
3. If the user selects "reserve," then the reservation page appears with the available time slots (see Fig. 4).

   If the user selected an individual rig, then the periods shown as unavailable will be those where the specified rig is explicitly reserved, is marked as being offline during that time, or because the pool to which it belongs has the same number of reservations as there are available rigs. In some cases, this rig may be shown as available, but may be tentatively allocated to a user who made a reservation for any rig within the rig pool. In this case, the system is able to attempt to move that previous reservation to a different rig from the pool before confirming the reservation. If this fails, the system will attempt to the next best reservation time the user can create.

   If the user selected a rig pool then the periods shown as unavailable will only be those where the number of reservations is the same as the number of available rigs.

   Once the user selects a period for a reservation, if the period is longer than the maximum allowed for

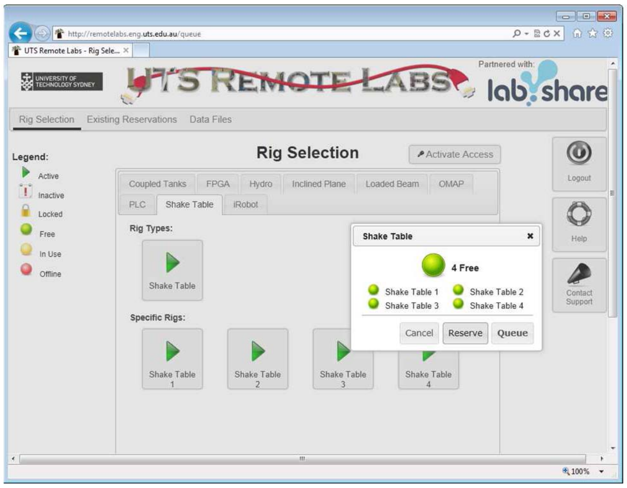


Fig. 3. Within Sahara, a user selects a rig pool to gain access (http://remotelabs.eng.uts.edu.au/queue).

this class of user then the time is reduced to the maximum.

4. Once a time slot is selected, the reservation is created, a confirmation message appears, and an e-mail message is sent to the user with the reservation details. The rig reservation page will show the reserved time slots (colored in gray) for all future accesses. The system can be configured to limit the number of active reservations that can be made by a user, and to block the user from making more than one concurrent reservation across multiple rig types.
5. When a user is logged in, and they have a pending reservation (within a configurable amount of time) their primary screen changes to provide them with a countdown to the start of their session. As soon as their reservation time arrives, the user is allocated to the rig, the rig page appears, and the experiment session commences.

3.3.2 *Priority Queuing*

Queuing supports "on demand" requests from users, providing "soonest available" access to a selected rig (or collection of rigs). The basic logic for the queuing process is the same for the first two steps of a reservation-based access, and then proceeds as follows:

1. The user selects the "queue" option (see Fig. 3). The request is placed in the queue, with associated information indicating which rig (or collection of rigs) they have requested access to, the default usage time specified for their user class, and any other relevant parameters. The user is provided with information as to their current position within the queue (see Fig. 6).
2. When any rig becomes available, Sahara scans through the queue looking for the first request from among those that are the highest priority, and which satisfies the following criteria:

    a. The request is either for the available rig, or for a pool which contains the available rig; and
    b. The request can be completed prior to the next reservation for this resource (i.e., the next reserved session for this resource is no sooner than the current time plus the default usage time for this user on this rig);

    Note that this allocation strategy ensures that a rig will only be allocated to a user in the queue when it can be guaranteed that they will have the full amount of time to which they are entitled, if they choose to use it.
3. Once a user is allocated to a rig, the rig page will appear and the experiment session will be started, including a timer counting the session time defined by the system.

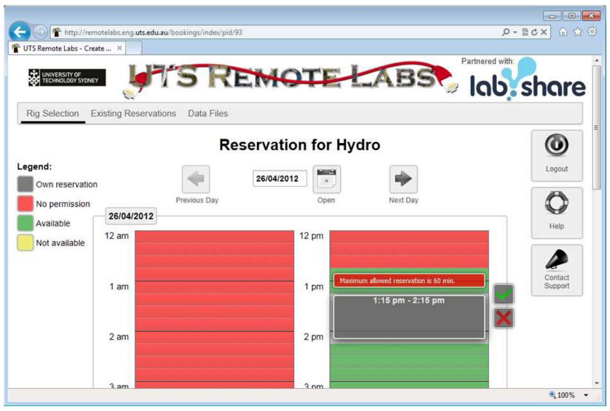


Fig. 4. Within Sahara, a user is making a reservation to use a selected rig (http://remotelabs.eng.uts.edu.au/bookings/).

In both types of scheduling, once a rig has been allocated, the rig session will be immediately commenced. If the user is currently logged in, then they will be redirected to the relevant rig access page. If they are not logged in, and they remain not logged in for a configurable amount of time (typically 10 minutes), then the session times out and the rig is returned to the available pool. The session is also considered to have ended when the user explicitly terminates their session or is inactive for a configurable amount of time, or the allowed session time is completed and no further extensions are allowed (a session can usually be extended a finite number of times if there is no queued user who wishes to use the rig or no pending booking for that rig).

The above descriptions imply a number of dependencies between the queuing and booking algorithms. For example, consider the case of a user in a queue waiting for access to a specific rig. The user cannot be allocated to the rig even if it is currently available unless their session (at maximum length allowed) is guaranteed to be complete before any pending booking comes due. This interdependence can therefore, potentially, lead to time periods where a rig is idle even though it is currently the subject of usage demands. An analysis of how serious an issue this is will be presented in Section 4.

4 Performance Evaluation

4.1 Process

To illustrate the challenges that can arise when both booking and queuing are supported for the same set of rigs, an analysis was undertaken of a set of live rig allocation data. For this analysis, an existing pool of coupled tank rigs, hosted by the UTS, was selected. These rigs are used by students to perform open-loop modeling of the dynamics of a well-known complex nonlinear system, and the subsequent linearization and design of the parameters for a proportional-integral-differential controller for the system.

During Spring semester 2011 (Spring semester runs from August to November and is one to the two main teaching semesters at most Australian Universities), two of the UTS coupled tank rigs were made available to a cohort of 114 students enrolled at another Australian University. These students made use of the coupled tanks over the period 02-Sept-2011 to 10-Oct-2011, with a total of 1,415 allocation requests and 700 successfully allocated rigs that resulted in a usage session. Note that a reservation request will always result in the subsequent allocation of a rig (unless the reservation is explicitly removed) though in some cases the user may not log in within the time-out period, and so the session would then be canceled. Conversely, a queue-based request will only result in the allocation of a rig if the student remains in the queue until the allocation occurs. The full data are as follows:

Total Requests

- Total number of requests = 1,415
  - Reservation-based requests = 353
  - Queue-based requests = 1,062
- Total number of successful allocations = 700
  - Reservation-based rig allocations = 353

    277 were used, 76 timed-out
  - Queue-based rig allocations = 347

    176 requests allocated without a wait

    171 requests allocated after waiting

    715 requests terminated before being allocated

Total rig usage time = 333 hrs 03 mins

Average allocations per active student = 7.46

Minimum student use = 1 session / 10 min 01 sec

Maximum student use = 49 sessions / 16 hrs 35 mins

25 students used the rigs for more than 5 hrs each.

While the system normally allows users to request either a specific rig instance (*"I would like to use Coupled Tanks #3"*) or any one of a given pool (*"I would like to use any available Coupled Tank rig"*) in this particular case only the former mechanism was enabled, so the students had to reserve or queue for a specific rig. This had implications where in busy periods there was a (relatively small) tendency for students to queue for a specific rig, discover they are not close to the front of the queue, and so leave the queue and make a request for a different rig. This explains, in part, the large number of queue-based requests that were terminated prior to being allocated.

After the end of the usage period extracts from the UTS Sahara system, log files were obtained and analyzed in Matlab. The data obtained included:

1. A listing of all reservations that were made for the relevant rigs and, where a reservation was not canceled, the rig session that corresponded to the redemption of this reservation;
2. A list of all allocated rig sessions, including: which rig was accessed; when the session was requested; when it commenced; when it ended; and why the session was terminated (by user request, due to an idle time out, or for some other reason); and
3. A set of information on the rig types, user classes, permissions, rig capabilities, and so on.

Using Matlab, relevant information was extracted from the database logs and analyzed to determine which rigs were used and how they were used. The results were then plotted to allow visualization of the patterns of use so that a deeper understanding of the interplay between reservations and queuing could be obtained.

4.2 Analysis of Actual Use

Fig. 7 shows an extract from the results of the coupled tanks analysis. This figure contains the data for both rigs that were in use during the period under analysis. The top graph provides a broad view of the overall usage of the coupled tanks rig 3. From this, it can be readily seen that for most of the time there was no queue, and even during heavy usage the queue rarely exceeded two users.

The bottom graph provides a more detailed view of specific period of approximately 18 hours (extending from hour 982 to hour 1,000, corresponding to approximately 11:00 am on 1-Oct-2011 to 5 am on 2-Oct-2011). A study of this figure shows some interesting behaviors emerging from the interplay between the booking and scheduling algorithms. Tracing through the time period, we can see the following events:

982.5: The rig is in use by a student (#1), who had queued for access. The queue is initially empty, but around 983 another student arrives (#2) and queues for access.

983.6: Student #1 completes. Student #2 is still in the queue, but is not allocated the rig because there is a pending booking in 24 minutes (at 984). The student therefore remains in the queue.

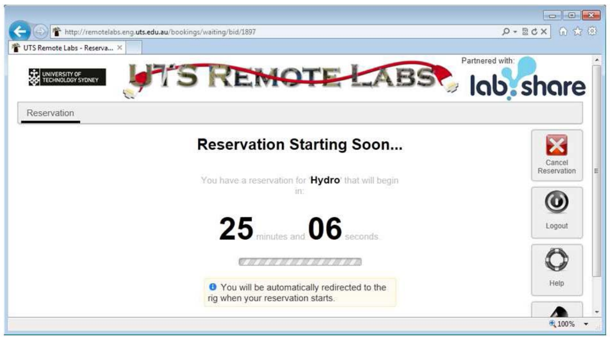
Slightly later

986.2: By this time, student #2 is still in the queue due to a sequence of booked sessions. Several more students arrive (#3 and #4) around 986.5 and the queue grows in length to 3.

987: The current student completes their session, but the queued students are not allocated the rig as there is not sufficient time before the next booking. Over the next 3 hours (until 990), a series of booked sessions are allocated and then quickly ended due to time outs, i.e., the students who made the bookings did not redeem those bookings. The unredeemed bookings however did mean that the queued students could not access the rig, and it spent considerable time sitting idle.

990: Finally, at 990, the rig is again allocated to a booked session, which again times out (after 10 minutes). At this point, the next booking is not until 991.5, and so finally a student in the queue can be allocated (after waiting over 7 hours, despite 4.5 hours of unused rig time during that period). Not unsurprisingly, the student is not available and their session quickly times out, as does the session for the next student in the queue.

This sequence of events highlights a key problem with the current algorithms used for managing the interplay between the booking and queuing algorithms. In particular, there may potentially be significant proportions of time (especially during heavy usage periods) where there are students queued for use, but the rig is idle and cannot be allocated to users in the queue because it is waiting for a pending reservation. To assess how significant a problem


http://remotelabs.eng.uts.edu.au/bookings/waiting/bid/1897

UTS Remote Labs - Reserva...

Reservation

Reservation Starting Soon...

You have a reservation for '**Hydro**' that will begin in:

25 minutes and **06** seconds

You will be automatically redirected to the rig when your reservation starts.

Fig. 5. Within Sahara, a user is waiting for the time of their reservation (http://remotelabs.eng.uts.edu.au/bookings/).

this is, we analyzed the peak usage periods during the timeline shown in Fig. 5. This showed the following results for the four day period from hour 968 to hour 1,064:

Total time: 92.00 hours

In use or pending allocation:	74.90 hours
In active use:	46.64 hours (62 percent of usage time)
Idle use:	9.18 hours (12 percent of usage time)
Waiting:	19.08 hours (26 percent of usage time)

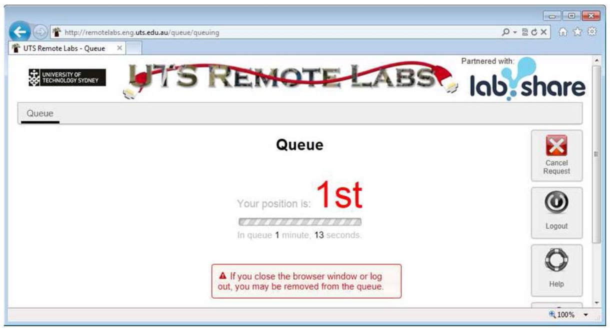
In other words, there was 74.90 hours during which the rig was either allocated to a user, or there were 1 or more queued users waiting to be allocated. Of this time, approximately 9 hours (12 percent) was spent allocated to a user who was not actually using the rig. This is typically a user who had made a reservation, but had not yet arrived to redeem that reservation—in this case, the system will after 10 minutes time out the user and make the rig allocatable to the next user. More surprisingly, there was 19 hours (26 percent of the time) that the rig was not allocated to anyone, but there were queued users waiting who could not be allocated because of a pending booking.

More problematic than the lost capacity that this represents is the additional waiting time required for users in the queue. An excellent example of this situation is shown in Fig. 7 for coupled tank 1. As discussed above, a user arrives at time 983 and enters the queue. They are not subsequently allocated to the rig until just after 990 (7 hours later!) despite significant periods of availability: 30 minutes at 983.5, 50 minutes at 985, 30 minutes at 987, and so on. Not unsurprisingly, by the time the student is finally allocated to the rig (just after time 990), they fail to make use of the allocated session and the session subsequently times out. Presumably, they have remained logged in, but are no longer monitoring the queue and so are unaware that they have finally been allocated the rig.

5 Scheduling Refinement

5.1 Modifications to the Scheduling

The above case study highlights an unanticipated consequence of the interplay between booking and queuing schemes. More importantly, it demonstrates that considerable care must be taken in the design of scheduling algorithms if we are to optimize performance measures of RLMSs, including aspects such as overall level of utilization and queue waiting times.


http://remotelabs.eng.uts.edu.au/queue/queuing

UTS Remote Labs - Queue

Queue

Queue

Your position is: 1st

In queue 1 minute, 13 seconds.

If you close the browser window or log out, you may be removed from the queue.

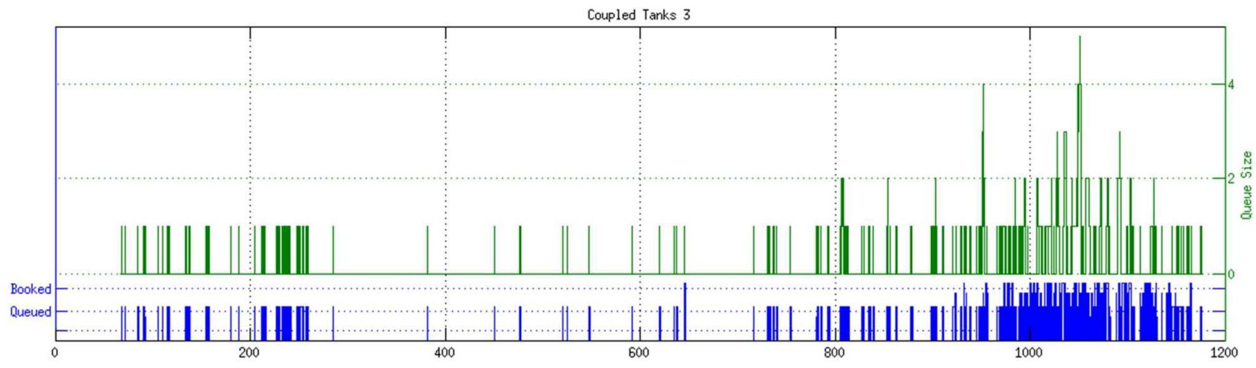
Fig. 6. A user is at the front of the queue, waiting for access to a selected rig (http://remotelabs.eng.uts.edu.au/queue/).

In this particular case, the degradation of performance arises due to queue-based allocations and reservation-based allocations having the same default duration. This means that when a reservation is not redeemed by the time it has timed out, the remaining time until the next reservation will always be less than the default duration for queued users, and hence they will not able to be allocated to the available rig.

There are a number of possible ways in which this issue could be addressed. To illustrate the possibilities, consider the following scenario: There are two 1-hour reservations from 3:00 pm to 4:00 pm and from 4:00 pm to 5:00 pm, respectively, and one user waiting in the queue. At 3:00 pm, the user with the reservation fails to log in. In the current system, the session will not time out until 3:10 pm, leaving only 50 minutes until the 4:00 pm booking and so the queued user (who is entitled to 60 minutes) will not be allocated.

Consider the following alternative scheduling algorithms that address the above issues:

1. *No allowance: Removal of the time allowance at the commencement of a session*. The current Sahara implementation provides a configurable period (typically 10 minutes) during which a session will not be canceled if the user is either idle or has not yet logged in to the system. This means, for example, that a user who has a reservation at 3:00 pm will have until 3:10 pm to log in and commence their session before the session is canceled (though if they log in after 3:00 pm, the session would still be limited to completion by 4:00 pm). We could remove this period, so that if a user is not logged in at the commencement of a reserved session, then it will be immediately canceled, leaving the full period for a different user. While this would be beneficial, and would have minimal effect on the user experience, it only addresses part of the problems identified above. There are still many cases where a student may be logged in but idle (e.g., they may have left their computer) or complete an experiment very quickly, resulting in very short sessions, and the associated problem arising from a subsequent gap until the next reservation that is large, but not quite a full time slot.
2. *FReduce: Allocation strategy modified to include forced reduction in maximum session duration for queued users*. In this approach, the system can choose to allocate shorter periods than the normal session length (up to



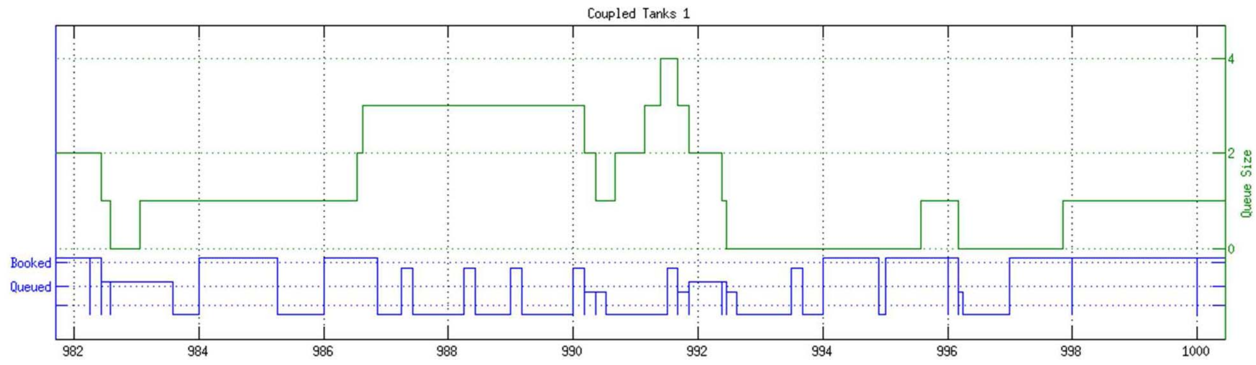


Fig. 7. Rig usage analysis: In each of the two graphs (one for each rig instance), the top curve shows the queue length over time (with the time axis measured in hours) and the bottom curve shows the status of the rig—specifically, whether it is idle, in use by a user who queued for access, or in use by a user who made a reservation. Where the rig status line is above the dotted line, this indicates that the rig was actively used (e.g., at time 984 coupled tanks 1 was allocated to a user with a reservation who made active use of the rig). Where the status line is below the dotted line, it indicates the session was allocated to the user but the rig was not used during that session (e.g., just after time 990 the rig was allocated to a user with a reservation, and then twice allocated to queued users, but none of these three made use of the rig and so all were timed out, after 10 minutes each). The upper graph shows the full time period of 1,200 hours (50 days). The lower graph has been zoomed into a period of 18 hours so that specific detail can be seen.

some maximum reduction in session length) in circumstances where this facilitates system performance. In the above scenario, at 3:10 pm the reservation is canceled, and then the queued user could be allocated a shorter 50 minute session rather than the normal default 60 minutes session. This is possible the simplest of the solutions and ensures a greater utilization of available rig time and minimization of queue lengths and waiting times, but at the expense of compromising the amount of time available to individual users.

3. *OReduce: Allocation strategy modified to include optional reduction in maximum session duration for queued users*. This is similar to the above approach but the user is given a choice as to whether they will accept the shorter session. When a shorter time window becomes available (the 50 minute window from 3:10 to 4:00 pm in the above example) users in the queue, who are nominally entitled to a longer period, are sequentially asked whether they would like to accept this shorter period until one is found who will accept the session. In many cases, they may know that they are able to complete the experiment quickly and would be willing to accept a shorter time slot if one is available in preference to waiting for a slot that guarantees their maximum time allocation. While requiring a more complex set of user interactions, it has the benefit that it does not force a user to accept a shorter session, and hence allows sessions of almost any length to be offered to users.
4. *Delay: Allocation strategy modified to include possible delay of commencement of reserved sessions by up to a configurable amount*. This involves modifying the interpretation of reservations so that they represent an approximate time at which a rig will be made available, rather than a guaranteed time (much like a reservation to see a doctor, where you arrive at the appointment time, and may need to then wait a "short" amount of time before the doctor is able to see you). If we have the flexibility to delay the start of the 4:00 pm booking until 4:10 pm, then the queued use can be allocated at 3:10 pm and still have the full 60 minute session (if they need it). In many cases, it would be likely that the queued user would complete the experiment in less time than the maximum allowed, and the 4:00 pm booking may not end up being affected at all.
5. *Both: Allocation strategy modified to include both the option of a reduced length session and the possible delay of commencement of reserved sessions (as described above).*

5.2 Evaluation Process

Options (3), (4), and (5) above represent the modifications that are most likely to result in the greatest enhancements in the scheduling system performance, while having minimal consequences for the user experience. To assess the impact of these possible changes, a simulation of the scheduling system has been developed, and this has been applied to the same reservation and queue requests made by the students from the case study using the following assumptions:

- For reservation-based accesses
  - Where a user failed to redeem a reservation in the original case study, they will also fail to redeem the reservation in the simulations;
  - Where a user redeemed a reservation, in the simulations their session time will be the same as for the case study.
- For queue-based accesses
  - When a user queued for access, but failed to redeem a session once allocated, the same situation will apply in the simulations. In the case where this user is offered a shorter session, they will not respond to the offer.
  - When a user queued for access and subsequently made use of an allocated session, their usage time in the simulation will remain the same. If they are offered a session that is shorter than the maximum, then their probability of accepting the offer will be modeled as follows:

$$p(accept) = \left((T_{offered} - T_{taken})/(T_{default} - T_{taken})\right)^k, \quad (1)$$

where $T_{offered}$ is the length of the session offered to the student, T_{taken} is the actual time that the student took in the real case study, and $T_{default}$ is the normal session time that is allocated (usually 1 hour in the case study). The parameter k adjusts the likelihood that a student will accept the offer (and for our simulations, we set $k = 1$). This formula results in users always accepting any offer when $T_{offer} \geq T_{default}$, and never accepting any session offer when $T_{offer} \leq T_{taken}$.

To assess the effect of these modified scheduling algorithms, we report on the following performance characteristics:

- Average queue length;
- Average time spent waiting in queue;
- Proportion of time a rig was not allocated while there were users waiting in a queue.
- For the *OReduce* and *Both* algorithms, the number of times the option of a reduced length session was accepted, and the average reduction in maximum allowed time.
- For the *Delay* and *Both* algorithms, the number of times the commencement of a reservation was delayed, and the average delay.

5.3 Results

Fig. 8 provides an extract showing the rig allocations for one of the coupled tank rigs, comparing the existing baseline allocation strategy (Fig. 8a) with the performance achieved using the *OReduce* and *Delay* alternatives that have been studied in depth. The impacts can be illustrated subjectively as follows:

OReduce: Allocation strategy modified to include optional reduction in maximum session duration for queued users (Fig. 8b);

- At time 1,036, we have three queued users. With the original allocation strategy there was only 50 minutes between the end of the reservation at 1,036:10 and the commencement of the next reservation at 1,037:00, and so none of the three queued users were allocated. In the modified strategy, each of the three was offered a shorter session (50 mins, 40 and 30 mins, respectively) and all three accepted, thereby reducing the queue length and wait time. A similar situation occurred at time 1,049.
- At time 1,051, again a 50 minute session was offered to a queued user, who accepted the offer and subsequently used the rig for 27 minutes. This contrasts to the queued user at time 1,044, who was offered a 50 minute session at time 1,044:10 but declined the offer, subsequently using the rig for a full 60 minutes at 1,048:10.

Delay: Allocation strategy modified to include possible delay of commencement of reserved sessions by up to a configurable amount

- At time 1,043:17, a queued user arrives, and there is only 43 minutes until the reservation at 1,044. With the original strategy, this user cannot be allocated, but in the modified algorithm, the reservation can be delayed by up to 20 minutes, and so the queued user is immediately allocated. This user uses the rig for a full 60 minutes, and so the user with the reservation at 1,044 is delayed until 1,044:17.
- At time 1,036:10, the next reservation was in 50 minutes, at 1,037:00, but given the option of a delay, this created the possibility of a 70 minute session—10 minutes longer than the default 60 minutes. Consequently, a queued user was allocated to the rig. This session subsequently only lasted 10 minutes and so there was no resultant delay to the reservation at time 1,037. A similar situation occurs at time 1,051:10, though in this case the resultant user uses the rig for 25 minutes—again with no resultant delay to the reservation. This appears to be the predominant situation—i.e., a potential delay that does not eventuate.

Subjectively, the results shown in Fig. 8 illustrate the impact of the changes to the scheduling algorithm, and highlight specific instances of changes in the timing of user allocation to rigs. It does not, however, provide a quantitative indication of the overall impact. To achieve this, we measured the performance parameters identified in Section 5.2. The results are shown in Table 1, both for the whole period of 1,200 hours, and for the peak period of use between hour 968 and hour 1,064. These results show what would be likely to have occurred during the use of the coupled tank rigs, based on the recorded data of the actual usage.

We begin by considering the *OReduce* rig allocation algorithm that incorporated offering shorter sessions, when available, to queued users. The results show that we would be likely to achieve significant improvements—particularly during heavy utilization periods. During the peak period of use, the uptake of the offers was 40 percent, with a resultant reduction in the average time that users spent waiting in the

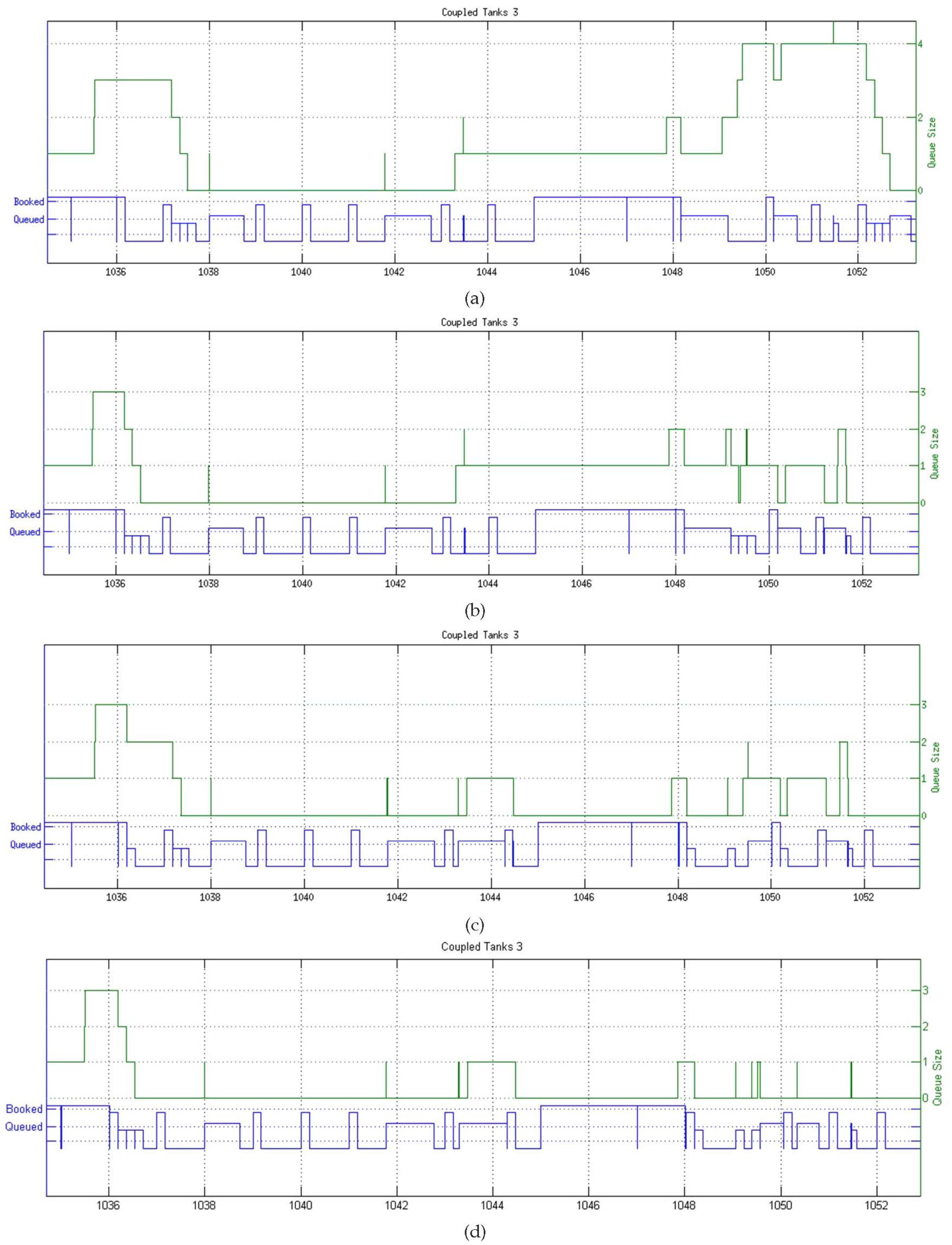


Fig. 8. Comparison of performance of resource allocation strategies: (a) an allocation strategy as implemented in current Sahara releases; (b) a *OReduce* allocation strategy modified to include offer of available shorter sessions to queued users; (c) a *Delay* allocation strategy modified to include delay of commencement of reserved sessions (by up to 20 minutes); and (d) a *Both* strategy including both offer of a reduced duration session and delayed commencement of session. See Fig. 7 for an interpretation of the graphs.

queue from 2.28 to 0.90 hours. It is important to note that there are factors that will be likely to affect the performance gains that are beyond our ability to simulate—such as the extent to which users are aware of the amount of time they are likely to need. Nevertheless, we expect that these factors will be more likely to lead to greater improvements than lesser, than was shown by the simulations. For example, in our simulation, if a user who took 40 minutes in reality was offered 30 minutes, then they would decline the offer. In reality, we may find that such a user, rather than declining the offer and continuing to wait in the queue, would accept the offer and attempt to complete within the reduced time, or else use the allocated session and then reenter the queue again to complete their experiment.

TABLE 1
Summary of Scheduling Simulation Results

(a) Performance over period 0 to 1200 for Coupled-Tanks-1

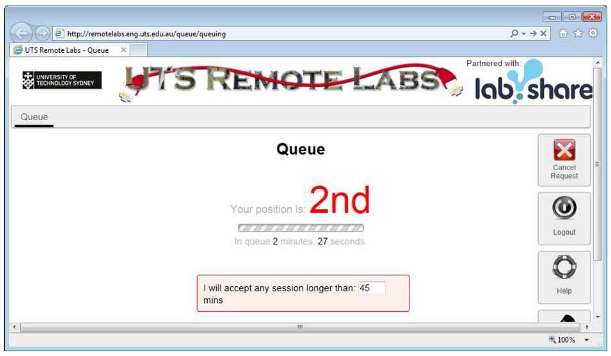
Performance metric	*Normal*	*ORedu.*	*Delay*	*Both*
Total time (hours)	1200	1200	1200	1200
# of reserv. sessions	174	174	174	174
# of queued sessions	432	432	432	432
Time in use or w. queue	283.6	262.9	266.4	257.6
Time in active use	225.2	225.2	225.2	225.2
Time in idle use	28.3	28.3	28.3	28.3
Time waiting for reserv.	32.2	11.5	13.97	5.19
Average queue length	0.148	0.078	0.096	0.063
Average time in queue	0.402	0.212	0.260	0.171
% of time rig unallocated whilst queued users	2.7%	0.9%	1.1%	0.4%
Queued users who	-	30	-	21
accepted short session		6.9%		4.9%
users with reservation who	-	-	18	25
were delayed			10.3%	14.4%
mean delay (mins)			2.19	3.18
maximum delay (mins)			15.07	15.07

(b) Performance over period 968 to 1064 for Coupled-Tanks-1

Performance metric	*Normal*	*ORedu.*	*Delay*	*Both*
Total time (hours)	96	96	96	96
# of reservation sessions	69	69	69	69
# of queued sessions	40	40	40	40
Time in use or w. queue	75.3	64.5	64.8	59.4
Time in active use	46.7	46.7	46.7	46.7
Time in idle use	9.4	9.4	9.4	9.4
Time waiting for reserv.	19.1	8.3	8.7	3.2
Average queue length	0.950	0.376	0.492	0.258
Average time in queue	2.28	0.90	1.18	0.62
% of time rig unallocated whilst queued users	19.9%	8.6%	9.1%	3.3%
Queued users who	-	16	-	13
accepted short session		40.0%		32.5%
users with reservation	-	-	12	16
who were delayed			17.4%	23.2%
mean delay (mins)			3.00	3.28
maximum delay (mins)			15.07	15.07

Next consider the *Delay* strategy, where we delay the commencement of reservations (by up to 20 minutes, in this simulation) when this allows a queued user to gain access earlier. In this case, we have also achieved significant improvements, though not quite as substantial as for the option to have reduced session allocations: The average waiting time in the queue, during the peak period, reduced from 2.28 to 1.18 hours. Of the 69 reservations that were assigned during this period, 12 (17.4 percent) were delayed, though the average delay was only 3.0 minutes (with only two delays greater than 4 mins: 10.9 and 15.1 minutes, respectively).

If we use both the option for reduced sessions and the possibility of delayed reservations (the *Both* strategy), then we find even greater improvements. The average waiting time in the queue, during the peak period reduces down to 0.62 of an hour (from the original level of 2.28 hours).


http://remotelabs.eng.uts.edu.au/queue/queuing

Queue

Your position is: 2nd

In queue 2 minutes, 27 seconds.

I will accept any session longer than: 45 mins

Fig. 9. Example user-interface mock-up for accommodating changes to allocation strategies: Users waiting in the queue are provided with the option to indicate the shortest session they would be prepared to accept.

5.4 Final Observations

The results show that, in general, significant improvements in performance can be achieved, and that both strategies that have been identified (*OReduce* and *Delay*) can provide significant performance improvements, but that the best outcomes are achieved when the two strategies are combined (*Both*).

One aspect that has been ignored in the previous discussions is the mechanisms by which users are informed about, and respond to, the proposed changes. A key aspect of managing the scheduling will be user management, and hence offering shorter sessions if they are available and delaying reservations when necessary will both require careful provision of information to users. Fig. 9 shows a mock up of a typical interface for supporting the usage of shorter sessions. In this scenario, users who are waiting in the queue can indicate that they would be prepared to accept any session that is longer than a specified minimum value (with the entry box initialized with the default session length for this user). This mechanism removes the need to repeatedly ask the user as each new session becomes available, but also allows the user to change the setting as desired.

A similar mechanism would be used to inform users with reservations when the start of their booked session has been delayed.

6 Conclusion

This paper has considered current approaches to the allocation of RL resources to users. The majority of existing RLMSs utilizes an allocation scheme based on either prior reservations or first-available queuing. Both approaches have merits and are typically suited to different usage characteristics. More recently, consideration has begun to be given to an approach that merges the two schemes to leverage the benefits of both.

The analysis of a case study based on live usage data has shown that a simple merge of queuing and booking can lead to a complex interplay that results in suboptimal performance. A series of modifications have been proposed that address these problems. The modifications have been analyzed through a series of simulations and shown to lead to enhanced performance, particularly during periods of heavy resource utilization.

The techniques that have been explored have the potential to be of significance for the management of access to any resources that utilize a combination of queuing and booking techniques.

Acknowledgments

The author thanks Michel de la Villefromoy, Michael Diponio, and Tania Machet for assistance with access to, and interpretation of, the case study data used in this paper. He also thanks Nagham Orou for discussions related to the potential variations to the scheduling algorithms.

References

[1] A. Hofstein and V. Lunetta, "The Laboratory in Science Education: Foundations for the Twenty-First Century," *Science Education,* vol. 88, no. 1, pp. 28-54, Jan. 2004.

[2] L. Gomes and S. Bogosyan, "Current Trends in Remote Laboratories," *IEEE Trans. Industrial Electronics,* vol. 56, no. 12, pp. 4744-4756, Dec. 2009.

[3] V.J. Harward et al., "The iLAB Shared Architecture: A Web Services Infrastructure to Build Communities of Internet Accessible Laboratories," *Proc. IEEE,* vol. 96, no. 6, pp. 931-950, June 2008.

[4] D. Lowe et al., "LabShare: Towards Cross-Institutional Laboratory Sharing," *Internet Accessible Remote Laboratories: Scalable E-Learning Tools for Engineering and Science Disciplines,* first ed., A. Azad, M. Auer, and J. Harward, eds., pp. 453-467, IGI Global, 2012.

[5] T. Richter, D. Böhringer, and S. Jeschke, "LiLa: A European Project on Networked Experiments," *Proc. Sixth Int'l Conf. Remote Eng. and Virtual Instrumentation (REV '09),* 2009.

[6] C. Gravier, J. Fayolle, B. Bayard, M. Ates, and J. Lardon, "State of the Art about Remote Laboratories Paradigms-Foundations of Ongoing Mutations," *Int'l J. Online Eng.,* vol. 4, no. 1, p. 19, 2008.

[7] P. Orduña, "Scheduling Schemes among Internet Laboratories Ecosystems," *Proc. Eighth Int'l Conf. Remote Eng. and Virtual Instrumentation (REV '11),* pp. 1-6, 2011.

[8] Labshare, "Sahara Labs," http://sourceforge.net/projects/labshare-sahara/, 2010.

[9] E.H. Hegarty, "Levels of Scientific Enquiry in University Science Laboratory Classes: Implications for Curriculum Deliberations," *Research in Science Education,* vol. 8, no. 1, pp. 45-57, 1978.

[10] A. Hofstein and V. Lunetta, "The Role of the Laboratory in Science Teaching: Neglected Aspects of Research," *Rev. Educational Research,* vol. 52, no. 2, pp. 201-217, 1982.

[11] J. Trevelyan, "Lessons Learned from 10 Years Experience with Remote Laboratories," *Proc. Int'l Conf. Eng. Education and Research Progress through Partnership (iCEER '04),* no. 2001, pp. 687-697, 2004.

[12] S. Murray, D. Lowe, E. Lindsay, V. Lasky, and D. Liu, "Experiences with a Hybrid Architecture for Remote Laboratories," *Proc. 38th Ann. Frontiers in Education Conf. (FiE '08),* 2008.

[13] C. Bright, E. Lindsay, D. Lowe, S. Murray, and D. Liu, "Factors that Impact Learning Outcomes in Remote Laboratories," *Proc. World Conf. Educational Multimedia, Hypermedia and Telecomm. (Ed-Media '08),* p. 15, 2008.

[14] J. Ma and J.V. Nickerson, "Hands-On, Simulated, and Remote Laboratories," *ACM Computing Surveys,* vol. 38, no. 3, article 7, Sept. 2006.

[15] E. Lindsay and M. Good, "Effects of Laboratory Access Modes upon Learning Outcomes," *IEEE Trans. Education,* vol. 48, no. 4, pp. 619-631, Nov. 2005.

[16] S.K. Taradi, M. Taradi, K. Radic, and N. Pokrajac, "Blending Problem-Based Learning with Web Technology Positively Impacts Student Learning Outcomes in Acid-Base Physiology.," *Advances in Physiology Education,* vol. 29, no. 1, pp. 35-39, Mar. 2005.

[17] E. Lindsay, D. Liu, S. Murray, and D. Lowe, "Remote Laboratories in Engineering Education: Trends in Students' Perceptions," *Proc. 18th Ann. Conf. Australasian Assoc. for Eng. Education (AaeE '07),* 2007.

[18] Z. Nedic, J. Machotka, and A. Nafalski, "Remote Laboratories versus Virtual and Real Laboratories," *Proc. 33rd ASEE/IEEE Frontiers in Education Conf.,* 2003.

[19] J.E. Corter, J.V. Nickerson, S.K. Esche, C. Chassapis, S. Im, and J. Ma, "Constructing Reality: A Study of Remote, Hands-On and Simulated Laboratories," *ACM Trans. Computer-Human Interaction,* vol. 14, no. 2, article 7, 2007.

[20] A. Arodzero, "World Wide Student Laboratory," Dept. of Physics, Univ. of Oregon, 1998.

[21] C. Rohrig and A. Bischoff, "Web-Based Environment for Collaborative Remote Experimentation," *Proc. 42nd IEEE Conf. Decision and Control,* Dec. 2003.

[22] P. Bauer, V. Fedák, and O. Rompelman, "PEMCWebLab—Distance and Virtual Laboratories in Electrical Engineering—Development and Trends," *Proc. Power Electronics and Motion Control Conf.,* pp. 2354-2359, 2008.

[23] D. Lowe et al., "Towards a National Approach to Laboratory Sharing," *Proc. Australasian Assoc. for Eng. Education Conf. (AAEE '09),* pp. 458-463, 2009.

[24] J.A. Stankovic, M. Spuri, M. Di Natale, and G.C. Buttazzo, "Implications of Classical Scheduling Results for Real-Time Systems," *Computer,* vol. 21, no. 1, pp. 287-25, June 1995.

[25] T.L. Casavant and J.G. Kuhl, "A Taxonomy of Scheduling in General-Purpose Distributed Computing Systems," *IEEE Trans. Software Eng.,* vol. 14, no. 2, pp. 141-154, 1988.

[26] C. Rohrig and A. Jochheim, "Java-Based Framework for Remote Access to Laboratory Experiments," *Proc. IFAC/IEEE Symp. Advances in Control Education,* pp. 1-6, 2000.

[27] J. Zackrisson and C. Svahnberg, "OpenLabs Security Laboratory - The Online Security Experiment Platform," *Int'l J. Online Eng.,* vol. 4, no. 2008, pp. 63-68, 2008.

[28] H. Yeung, D. Lowe, and S. Murray, "An Investigation into Supporting Interoperability of Remote Laboratories," *Proc. Seventh Int'l Conf. Remote Eng. and Virtual Instrumentation (REV '10),* pp. 71-79, 2010.

[29] P. Wieder, O. Waldrich, and W. Ziegler, "Advanced Techniques for Scheduling, Reservation, and Access Management for Remote Laboratories," *Proc. Second IEEE Int'l Conf. e-Science and Grid Computing (e-Science '06),* pp. 128-134, 2006.

[30] Y. Li, S.K. Esche, and C. Chassapis, "A Scheduling System for Shared Online Laboratory Resources," *Proc. 38th Ann. Frontiers in Education Conf.,* pp. T2B-1-T2B-6, 2008.

[31] A. Maiti, "Time Scheduling Schemes in Online Laboratory Management Systems," *Int'l J. Online Eng.,* vol. 6, no. 4, pp. 44-48, Nov. 2010.

[32] A. Maiti, "A Hybrid Algorithm for Time Scheduling in Remotely Triggered Online Laboratories," *Proc. IEEE Global Eng. Education Conf.,* pp. 921-926, 2011.

[33] I. Gustavsson, J. Zackrisson, L. Håkansson, I. Claesson, and T. Lagö, "The VISIR Project—An Open Source Software Initiative for Distributed Online Laboratories," *Proc. Int'l Conf. Remote Eng. and Virtual Instrumentation (REV '07),* pp. 1-6, 2007.

[34] J. Garcia-Zubia et al., "Towards an Extensible WebLab Architecture," *Proc. Third IEEE Int'l Conf. E-Learning in Industrial Electronics (ICELIE '09),* pp. 115-120, 2009.

[35] V. Mateos, A. Gallardo, T. Richter, L. Bellido, P. Debicki, and V. Villagra, "LiLa Booking System: Architecture and Conceptual Model of a Rig Booking System for On-Line Laboratories," *Int'l J. Online Eng.,* vol. 7, no. 4, pp. 26-35, 2011.

[36] M.A. Bochicchio and A. Longo, "Extending LMS with Collaborative Remote Lab Features," *Proc. 10th IEEE Int'l Conf. Advanced Learning Technologies,* pp. 310-314, 2010.

[37] J.M. Martins Ferreira and A.M. Cardoso, "A Moodle Extension to Book Online Labs," *Int'l J. Online Eng.,* vol. 1, no. 2, 2005.

David Lowe received the BE (hons.), PhD, and GradCert (higher education) degrees. He is the associate dean (education) and a professor of software engineering at the Faculty of Engineering and IT, The University of Sydney. He is also the CEO of the not-for-profit organization The LabShare Institute, and president of the Global Online Laboratory Consortium. He has active research interests in the areas of real-time control of embedded systems in the web environment, and remote access to, and control of, physical laboratory systems. He has published widely, including more than 150 papers and three books. He is a senior member of the IEEE.