Platform PreEmption Management Solution

Optimization Strategies for Critical Resources Using Platform LSF, Platform LSF License Scheduler, and Librato Smart Suspend

A TECHNOLOGY WHITEPAPER

AUTHORS:
Renita W. K. Leung
Platform Computing Inc.
(rleung@platform.com)

Keith D. Ball
Librato, Inc.
(kball@librato.com)

Date: April 21st, 2009
Table of Contents

Platform PreEmption Management Solution ................................................................. 1

3.1 Smart Suspend Architecture .................................................................................. 7

4.1 Job Pre-emption Using Smart Suspend with Platform LSF Queues ......................... 10

4.2 Job Pre-emption Using Smart Suspend with Platform LSF License Scheduler ......... 10

5.1 Deployment Overview ............................................................................................ 12

5.2 Integration with Platform LSF ............................................................................... 12

5.2.1 Configuring Platform LSF to Use Smart Suspend ............................................ 12

5.2.2 Controlling Platform LSF Jobs Using Smart Suspend ..................................... 14

5.3 Integration with Platform LSF License Scheduler ............................................... 15

5.3.1 Configuring Platform LSF and Platform LSF License Scheduler to Use Smart Suspend .... 15

5.3.2 Controlling Platform LSF License Scheduler Jobs Using Smart Suspend ........... 17

6.1 About Librato Inc. ................................................................................................ 18

6.2 About Platform Computing Inc. ........................................................................... 18
1 Common Resource Management Challenges

Managing finite compute resources and making sure their utilization is optimized can prove to be a significant and challenging endeavour. Presented in this document are a description of these challenges and a few techniques for mitigating and addressing them.

Resource service level agreement (SLA) guarantees
Users may blame missed project deadlines on poor license SLAs, while it is difficult for the computing resource manager to understand and control resource utilization, it is important for him/her to deliver expected SLAs for end-users and mitigate places of usage inefficiencies.

Effective resource utilization and sharing
In compute environments where resources are shared among different groups, instances of 100% license utilization are common, leading to user complaints about lack of licenses resources to complete critical work. Often, these peaks in license utilization are cyclical, occurring during times of high demand in the electronic design cycle, and are therefore both predictable, and expensive to mitigate. Purchasing additional licenses that will only be used during peak demand times and then lie idle for the rest of the week, month, or year does not make economic sense, especially in today’s stressed business climate. To alleviate delays caused by competition for limited licenses, some users will learn to hoard licenses through various techniques to ensure access in time of need. This practice of “gaming the system” results in idle licenses that are held but almost always left unused. This further exacerbates the shortage caused by legitimate demand for licenses by taking even more licenses out of circulation just when they are needed most. To maintain a competitive edge, companies must implement solutions that ensure effective engineering resource utilization in support of timely, efficient and cost effective design production.

Limited premium resources
The integrated circuit design process relies, in certain stages, on large-memory machines and expensive (and scarce) application licenses to complete the design flow. The high cost of these resources and the relatively small fraction of the overall compute cycle the design process consumes present a dilemma to infrastructure managers: How best to optimize the use of these resources to maintain the most cost-effective production cycle? Traditionally, these computing resources have been limited to mission-critical tasks to ensure their availability when required, but isolating them deprives the enterprise of powerful hardware resources that would compress multiple design cycles and simultaneously increase hardware utilization. Unfortunately, by making these resources available to non-critical design flow tasks, corporations run the risk they will not be immediately available to perform critical tasks when required.
Clean suspension of workload
While it is possible to stop (pre-empt) low priority tasks to allow other higher priority jobs to run on a compute resource, resources such as memory and licenses may still be held by the lower priority job, ultimately preventing the higher priority work from running. This often leads to the termination of lower priority jobs when high priority work arrives, wasting time and resources already consumed by the lower priority work. As a result, many organizations do not implement priority based job suspension because such waste is unacceptable. This reduces the deliverable service level and impacts business effectiveness.

Globalization
As organizations expand globally, remote collaboration becomes more important. Part of the globalization process requires decoupling engineering personnel from project data and computation. It is therefore increasingly imperative to have continuous, automated, and centralized resource scheduling capabilities to keep resources utilized. Moreover, with global design centers working on a multitude of ongoing projects, strategic capacity planning becomes much more demanding to ensure efficient service delivery and maximized resources.

2 Solution Introduction
The PreEmption Management Solution enables pre-emptive scheduling and resource sharing by delivering clean-release resource suspension technology that effectively frees up resources for use by priority workload. The solution dynamically liberates memory, application licenses, CPUs, network and storage I/O on demand. By enabling suspend-resume for streamlined pre-emptive scheduling this solution helps to:

- Reduce capital investment and associated operating and administrative expenses
- Reduce idle time for expensive application licenses
- Maintain SLAs for high priority jobs and resource allocation
- Reliably optimize license usage with prioritization
- Maximize usage of large memory machines
- Increase server utilization and job throughput by 10-50% depending on priority mix
- Guarantee project completion times by allocating critical resources according to business priority
- Maximize overall job slot utilization
- Reduce production implementation cost with an out-of-the-box solution
The PreEmption Management Solution consists of an integration between Librato Smart Suspend, Platform LSF, and Platform LSF License Scheduler. Librato Smart Suspend is a lightweight resource optimization solution for computing environments. It enables the safe suspension of a currently running job, relinquishing all the system resources consumed by that job (CPU, memory, and even application licenses) in order to run a higher priority job.

Platform LSF is a highly scalable, sophisticated workload management system that intelligently schedules batch, interactive, flow, parallel, parametric and data-centric workloads according to policies to maximize productivity of systems, software and users by reducing application run-times and optimizing resources in a controlled heterogeneous environment.

Platform LSF License Scheduler provides intelligent, policy-driven application license consumption optimization for both Platform LSF clusters and non-Platform LSF workloads by allocating a virtualized pool of FLEXnet-based licenses to users based on an organization’s established distribution policy. By optimizing the efficiency of license usage, Platform LSF License Scheduler compresses work schedules, reduces expenses and ensures license usage compliance.

Platform LSF and Platform LSF License Scheduler together with Smart Suspend provide for the proper suspension and resumption of these jobs by intelligently managing the necessary system resources, allowing a clean release of application licenses and subsequent resumption of jobs when requisite resources are available again. This solution addresses a variety of requirements described in the following paragraphs.

Meet job scheduling SLAs
Smart Suspend enhances Platform LSF and Platform LSF License Scheduler’s ability to provide a priority guarantee for a given job without wasting valuable time and compute cycles. A high priority job now can always be guaranteed full access to hardware resources on the execution host, since suspension of lower priority jobs will free up all of the system resources they hold, including system memory and application licenses. This is different from standard suspension methods because the memory is not left to the OS kernel to be swapped out. Used memory is literally written out to disk and released rather than swapped. This allows for the higher priority (suspending) job to have full access to the hardware resources rather than wait for used memory pages to be swapped. Removing the dependency on configured swap space also allows the suspension of jobs running on diskless nodes or hosts with insufficient swap space, as Smart Suspend will write memory to either local or network storage. Smart Suspend provides virtually limitless memory storage compared to the fixed and limited storage provided by swap space.
Eliminate resource waste
With Smart Suspend there is no longer a need to kill running jobs in order to accommodate higher priority jobs. Platform LSF and Platform LSF License Scheduler can more optimally suspend lower priority running jobs and resume them later from where they left off, thereby saving valuable time and compute cycles. Unlike conventional job suspension solutions, this solution frees all resources consumed by a job, including system memory.

Make optimal use of application licenses
Technical computing application licenses can be very costly and to remain competitive enterprises must make the best use of the licenses they have purchased. Conventional job suspension methods do not free up unused application licenses when a job is suspended, leaving these costly resources allocated and in use but idle, even though the job using them has been suspended. Without the use of Smart Suspend, Platform LSF License Scheduler commonly employs the signal SIGTSTP or FLEXnet command lmremove as a mechanism for license release, the results of which are application dependent. Together with Smart Suspend, this application dependency has been removed. As a result, lower priority jobs frequently must be killed in order to free up the license or extra application licenses need to be purchased. With Platform LSF, Platform LSF License Scheduler and Smart Suspend in place, neither of these undesirable choices is required.

Reduce total cost of ownership
Without automation, management of computing resources such as CPU, memory and licenses can be a costly and laborious undertaking. With Platform LSF and Platform LSF License Scheduler, various workloads are intelligently scheduled against existing resources without manual intervention. Smart Suspend further complements this automation by faithfully managing resources from workloads that have been suspended by Platform LSF and Platform LSF License Scheduler to ensure the completion of more mission-critical workloads. The solution is well integrated and works out of box. This allows organizations to focus on process and policies as opposed to the mechanics and development time of building and troubleshooting multiple discrete technologies. Total cost of ownership is significantly reduced compared to conventional, passive static methods of resource management.

Key Solution Highlights:
- Suspends batch job and frees memory, CPUs, and application licenses
- Completely transparent to application and operating system
- Integrated seamlessly with Platform LSF and Platform LSF License Scheduler
- Smart Suspend technology incurs less than 0.1% performance overhead
- Supports both serial and parallel batch applications using Ethernet MPICH and HP-MPI
3 Solution Components

3.1 Smart Suspend Architecture

Smart Suspend is based on Librato’s operating system abstraction technology. It employs a user space library that is loaded at run-time for any dynamically linked application by setting the LD_PRELOAD environment variable. This additional library becomes a filter through which all system related library calls are performed. As such, Librato technology has granular control to allow, disallow, or modify the way any dynamically linked application behaves on a system by intercepting these standard library calls. Hence this technology is often referred to as an OS abstraction layer.

Through the OS abstraction layer, Smart Suspend can pre-empt running jobs by suspending the application’s CPU usage, writing memory to disk (local or remote), and keeping only a minimal footprint. Unlike traditional job suspension (or pre-emption) methods, a job suspended using Smart Suspend relinquishes both memory and application licenses to enable a higher priority job to take advantage of these system resources.

An application using Smart Suspend technology incurs a negligible performance overhead of less than 0.1%. Suspended applications have their memory written to a filesystem, and thus the speed of that operation depends on the speed of the filesystem. The benefit of course is that the high priority job, once it is started, does not have to wait for kernel I/O to swap used memory pages while it is running.

Figure 1: Smart Suspend Architecture
Note:  **Mlocked Memory**

Sometimes an application will explicitly request that the operating system not allow certain portions of resident memory to be swapped out. The application does this by using the mlock() system call. Smart Suspend is able to free mlocked memory and upon resumption make sure that it is correctly locked as before.

### 3.2 Platform LSF Architecture

In a Platform LSF cluster, heterogeneous hosts are configured centrally and managed from a master Platform LSF host. Each slave machine in the cluster collects its own computational resource information periodically and reports back to the master. Users then submit their jobs to Platform LSF and the master makes a decision on where to run the job based on the collected resource information together with scheduling and queue policies. Scheduling factors include the active time windows of the queues or hosts, resource requirements of the job, availability of eligible hosts, various job slot limits, job dependency conditions, fairshare constraints, and load conditions.

![Figure 2: Platform LSF Architecture](image)
3.3 Platform LSF License Scheduler Architecture

Platform LSF License Scheduler bridges multiple Platform LSF workloads and FLEXnet-based license resources. Platform LSF License Scheduler employs a daemon-based architecture in which license information is collected and used to schedule against configurable project-based policies. Scheduling decisions are communicated by the master daemon (bld) to the master batch daemon (mbatchd) running on the master host of Platform LSF clusters.

Figure 3: Platform LSF License Scheduler Architecture
4 Pre-emption Strategies

4.1 Job Pre-emption Using Smart Suspend with Platform LSF Queues

Figure 4 below shows how Smart Suspend is used for Platform LSF queue pre-emption. Jobs 1-4 are the currently running jobs. Assume that Job 1 and Job 2 have higher priorities than Job 3 and Job 4. While these four jobs are running, Job 5 is submitted to the queue with a higher priority than Jobs 3 and 4 but insufficient resources (i.e. job slot) are available to start it. Platform LSF determines that it can free up the job slot required by Job 5 by pre-empting Job 4. Once Job 5 is completed, Job 4 is resumed automatically.

Note that while Job 4 is suspended, all system resources are relinquished, including CPU, memory and application licenses (if configured). Upon resumption, all required resources including the application licenses are reacquired.

![Figure 4: Platform LSF Job Pre-emption Example](image)

4.2 Job Pre-emption Using Smart Suspend with Platform LSF License Scheduler

Figure 5 below shows how Smart Suspend is used for Platform LSF License Scheduler project pre-emption. Jobs 1-4 are the currently running jobs. Assume again that Job 1 and Job 2 belong to a project that owns all of the available licenses and Job 3 and Job 4 belong to a project that does not own any licenses. While these four jobs are running, Job 5 belonging to the project with ownership is submitted but insufficient resources (i.e. licenses) are available to start it. Platform LSF License Scheduler determines that
it can free up the license required by Job 5 by pre-empting Job 4. Once Job 5 is completed, Job 4 is resumed automatically.

Note that while Job 4 is suspended, all system resources are relinquished, including CPU, memory and application licenses. Upon resumption, all required resources including the application licenses are reacquired.

Figure 5: Platform LSF License Scheduler Job Pre-emption Example
5 PreEmption Management Solution Deployment

5.1 Deployment Overview

Installation of Smart Suspend takes just a few minutes and once the software is installed on each node in a cluster, there are a few simple configuration steps:

- Set up a directory to which Smart Suspend can store job status. For parallel jobs or jobs involving multiple nodes, this directory should reside on a shared file system.
- Define a location to which to save the suspended job’s memory on every host. This can be either a local or shared filesystem.
- Configure integration with Platform LSF and Platform LSF License Scheduler.

Besides Platform LSF and Platform LSF License Scheduler automated pre-emption mechanisms, once the above is set up, Platform LSF commands (bstop, bresume) can be used to suspend and resume jobs; these commands invoke the underlying Smart Suspend command (ssrcmd).

The `ssrcmd` command is the only command provided by Smart Suspend. It uses different options to perform suspend and resume operations. A user can call this command directly although it is typically not necessary because Smart Suspend has been seamlessly integrated with Platform LSF and Platform LSF License Scheduler.

5.2 Integration with Platform LSF

Platform LSF has built-in hooks to customize suspension and resumption of jobs. Smart Suspend has been seamlessly integrated and tested with Platform LSF to allow users to take advantage of the job pre-emption capability without reengineering their existing job management infrastructure.

The default Platform LSF suspend/resume behaviour is to send `SIGSTOP/SIGCONT` signals to running jobs. The Smart Suspend integration substitutes these signals with the Smart Suspend `ssrcmd` command.

5.2.1 Configuring Platform LSF to Use Smart Suspend

Platform LSF provides a pre-emptive scheduling feature that allows pending high-priority jobs to take resources away from running jobs of a lower priority. To keep the integration as simple and transparent as possible, Smart Suspend takes advantage of Platform LSF’s queue-based pre-emptive scheduling model, but uses the Smart Suspend technology for handling the underlying job suspension and resumption operations.
At least two Platform LSF queues are required for queue pre-emption:

**Pre-emptive queue** – Jobs in this queue can pre-empt jobs in any pre-emptable queue of lower priority.

**Pre-emptable queue** – Jobs in this queue can be pre-empted by jobs from any high priority pre-emptive queue.

For Platform LSF queues that will participate in the queue pre-emption with Smart Suspend, the following four parameters are required to be set in the queue pre-emption file (lsb.queues):

I. **PRIORITY**
II. **JOB_CONTROLS**
III. **PREEMPTION**

The following is an example of a queue definition for a high priority (pre-emptive) queue and a low priority (pre-emptable) queue:

```plaintext
Begin Queue
QUEUE_NAME = hipri
DESCRIPTION = For HIGH Priority Jobs
PRIORITY = 100
NICE = 20
JOB_CONTROLS = SUSPEND[/ssrlsf.sh -s] \ RESUME[/ssrlsf.sh -r]
PREEMPTION = PREEMPTIVE[lowpri]
End Queue

Begin Queue
QUEUE_NAME = lowpri
DESCRIPTION = For LOW Priority Jobs
PRIORITY = 50
NICE = 20
JOB_CONTROLS = SUSPEND[/ssrlsf.sh -s] \ RESUME[/ssrlsf.sh -r]
PREEMPTION = PREEMPTABLE[hipri]
End Queue
```
Where:

- `<path_to_lsf_integration>` specifies the path where the Platform LSF integration wrapper script (`ssrlsf.sh`) is installed. By default, it is installed in `/usr/share/smartsuspend`, but it can also be installed on a shared file system.

Note:

- Please ensure that the variable `SSRLSF_LOGDIR_ROOT_DEFAULT` in `ssrlsf.sh` specifies the default path for `SSRLSF_LOGDIR_ROOT`, which is the base directory where `ssrlsf.sh` creates the subdirectory `<SSRLSF_LOGDIR_ROOT>/<LSB_JOBID>`. This subdirectory contains the Smart Suspend status directory (required for storing job state communication data and process IDs (PIDs)), in addition to various log files. Ideally, `SSRLSF_LOGDIR_ROOT_DEFAULT` is a location on a shared file system, because the directories for jobs started on all machines are then kept in one location. A location on a local file system may also be used if desired.

Jobs will then be submitted using Platform LSF’s job submission commands (`bsub`) to submit jobs to queues using the `-q bsub` flag, based on the job’s priority or SLA.

### 5.2.2 Controlling Platform LSF Jobs Using Smart Suspend

Once the job queues are set up using Smart Suspend, they behave the same as normal Platform LSF queues working with suspend/resume. A user can also suspend and resume jobs manually using commands such as `bstop` and `bresume`, which will invoke calls to the Smart Suspend command with corresponding options instead of the default actions used by those that come with Platform LSF.

To submit a Smart Suspend-enabled Platform LSF job, the job command needs to be prefixed by the Librato-Platform LSF integration wrapper script `ssrlsf.sh`.

A user can then submit a Smart Suspend enabled job to the pre-emptable queue using:

```
# bsub -q lowpri <path_to_lsf_integration>/ssrlsf.sh -- <application_name> <application_arguments>
```

Jobs will be suspended and resumed or will run to completion automatically based on their priorities.

To suspend a job using Smart Suspend:

```
# bstop <job_id>
```

To resume a suspended job:

```
# bresume <job_id>
```

To kill a job:

```
# bkill <job_id>
```
5.3 Integration with Platform LSF License Scheduler

Platform LSF License Scheduler shares with Platform LSF the same built-in hooks to customize suspension and resumption of a job. Smart Suspend has also been seamlessly integrated and tested with Platform LSF and Platform LSF License Scheduler to allow users to take advantage of the license optimization capabilities without reengineering their existing Platform LSF and Platform LSF License Scheduler job management infrastructure.

The default Platform LSF License Scheduler suspend/resume behaviour is to send SIGTSTP and SIGCONT signals to running jobs. The Smart Suspend integration substitutes these signals with the Smart Suspend `ssrcmd` command.

5.3.1 Configuring Platform LSF and Platform LSF License Scheduler to Use Smart Suspend

Platform LSF License Scheduler provides a pre-emptive scheduling feature that allows projects with ownership of license resources guaranteed access to those licenses by way of pre-emption if another project not having ownership is currently using them.

As with the Platform LSF case, Smart Suspend leverages Platform LSF License Scheduler’s project-based pre-emptive scheduling model, but uses the Smart Suspend technology to handle the underlying job suspension and resumption operations.

There are three main configurations components:

1. A distribution policy must be configured with license ownerships in the feature section of the Platform LSF License Scheduler configuration file (lsf.licensescheduler). The following example shows that license project “p1” owns 10 hsim licenses and “p2” does not own any hsim licenses.

   ```
   Begin Feature
   NAME = hsim
   DISTRIBUTION = SynopsysServer(p1 1/10 p2 1)
   End Feature
   ```

2. Similar to Smart Suspend’s integration with Platform LSF, the following parameter is required to be configured for queues used by Platform LSF License Scheduler jobs in the Platform LSF queues configuration file (lsb.queues):

   ```
   I. JOB_CONTROLS
   ```
Note:
Unlike the Platform LSF queue pre-emption case, the parameter PREEMPTION is not required.

The following is an example of a queue definition queue:

Begin Queue
QUEUE_NAME = normal
DESCRIPTION = For License Scheduler Jobs
PRIORITY = 100
NICE = 20
JOB_CONTROLS = SUSPEND[<path_to_lsf_integration>/ssrlsf.sh -s]\ RESUME[<path_to_lsf_integration>/ssrlsf.sh -r]
End Queue

Where:
- `<path_to_lsf_integration>` specifies the path where the integration wrapper script (ssrlsf.sh) is installed. By default, it is installed in /usr/share/smartsuspend, but it can also be installed on a shared file system.

Note:
- Please ensure that the variable SSRLSF_LOGDIR_ROOT_DEFAULT in ssrlsf.sh specifies the default path for SSRLSF_LOGDIR_ROOT, which is the base directory where ssrlsf.sh creates the subdirectory `<SSRLSF_LOGDIR_ROOT>/<LSB_JOBID>`. This subdirectory contains the Smart Suspend status directory (required for storing job state communication data and process IDs (PIDs)), in addition to various log files. Ideally, SSRLSF_LOGDIR_ROOT_DEFAULT is a location on a shared file system, because the directories for jobs started on all machines are then kept in one location. A location on a local file system may also be used if desired.

3. The following parameters need to be set up in the main Platform LSF configuration file (lsf.conf) to direct Platform LSF License Scheduler to use JOB_CONTROLS in lsb.queues for suspension and resumption mechanisms and release the job slot on suspension:

```
LSF_LIC_SCHED_PREEMPT_STOP=Y
LSF_LIC_SCHED_PREEMPT_SLOT_RELEASE=Y
```

Jobs can then be submitted to their respective license projects (specified by –Lp on the bsub command line).
5.3.2 Controlling Platform LSF License Scheduler Jobs Using Smart Suspend

Once the appropriate configurations are made in Platform LSF and Platform LSF License Scheduler (detailed in 4.3.1 above), job behaviours are the same as Platform LSF License Scheduler with suspend/resume. A user can also suspend and resume jobs manually using commands such as **bstop** and **bresume**, which will invoke calls to the Smart Suspend command with corresponding options instead of the default actions used by Platform LSF License Scheduler.

Platform LSF License Scheduler jobs need to augment the job submission line to prefix the actual job command with the Librato-Platform LSF integration wrapper script `ssrlsf.sh` (the same script as specified in 4.2.2) before the application name. Either the `-f` or `-l ssrlsf.sh` option needs to be set in order to set up Smart Suspend license support. This modification can be done on the command line or within a user’s submission script. License support allows any subsequent job suspend operation to properly relinquish the license and then re-obtain the license when the job is resumed:

```
ssrlsf.sh -f <license_file> -- <application_name> <application_arguments>
```

**Where:**

- `-f` specifies the FLEXnet `$LM_LICENSE_FILE` environment variable as the license file.
- Alternatively, `-l` flag can be used instead of `-f` to specify FLEXnet license server host or multiple hosts in a colon-separated list (`<host>:<host>`).

**Note:** `-f` and `-l` flag cannot be used together

For example:
```
# bsub -Lp p1 -R "rusage[hsim=1]" <path_to_lsf_integration>/ssrlsf.sh -f "1718@sjlapl1" -- hsim_job_script
```

Jobs will be suspended or resumed or will run to completion automatically based on their license ownerships. Much the same as the Platform LSF integration case, administrators can manipulate Smart Suspend enabled jobs manually using normal Platform LSF commands.

To suspend a job using Smart Suspend:
```
# bstop <job_id>
```

To resume a suspended job:
```
# bresume <job_id>
```

To kill a job:
```
# bkill <job_id>
```
6 About Solutions Developers

6.1 About Librato Inc.

Librato is an early pioneer in checkpoint/restart software for the high performance computing (HPC) market.

The company’s core technology is based on innovative techniques to monitor and control system resource usage by applications without requiring any application or operating system modifications, allowing Librato products to be seamlessly deployed in existing data center infrastructures and generate a rapid return on investment. This core technology is at the heart of all Librato products including the two HPC/Grid related products: Availability Services (AvS) and Smart Suspend. Visit www.librato.com for more information.

6.2 About Platform Computing Inc.

Platform Computing is the leader in grid and cloud computing software that dynamically connects IT resources to workload demand according to business policies. Over 2,000 of the world’s largest organizations rely on our solutions to improve IT productivity and reduce data center costs. Platform has strategic relationships with Dell, HP, IBM, Intel, Microsoft, Red Hat, and SAS. Building on 16 years of market leadership, Platform continues to help data centers be more efficient, responsive and dynamic. Visit www.platform.com.