Librato Load Manager™ Technical Overview

1 Load Manager Overview

Librato Load Manager (LM) is a workload management solution that allows authorized users to monitor and control the behavior of their data center applications with respect to CPU, memory, network I/O and storage I/O. As a user space technology, LM achieves this without any modification to the underlying operating system (OS) or the applications under management. Load Manager software is also very light weight with typically less than 1% overhead.

Customers use Load Manager for a variety of reasons such as to:

Load Manager Key Features:

- Fine-grained, real-time monitoring and control of CPU, memory, network I/O, and storage I/O utilization
- No OS or application modification required
- Flexible views into monitoring data
- Chargeback billing
- Policy-driven, fair share scheduler of system resources
- Dynamic system resource rebalancing

- **Consolidate**

  Consolidate different applications on to one physical machine, increasing overall resource utilization and reducing total cost of ownership (TCO).

- **Guarantee SLAs**

  Guarantee service level agreements (SLAs) for consolidated applications (including latency sensitive ones) by ensuring that specified minimum system resource (i.e. CPU, memory, network I/O or storage I/O) allocations are guaranteed to each application.

- **Increase utilization**

  Dramatically improve machine utilization by dynamically re-allocating unused system resources to applications that can benefit from additional capacity. For example, if an application is allocated or guaranteed 80% of the CPU but is currently only using 20%, the remaining capacity can be provided to other applications until it is needed.

- **Monitoring**

  Provide fine-grained (down to an individual process/thread) statistics for machine utilization which can in turn be used for capacity planning, application profiling or chargeback billing purposes.

Load Monitor

Librato offers two different load management products, Load Monitor and Load Manager. Although both products are licensed separately, Load Monitor is a subset of Load Manager with only the monitoring, as opposed to monitoring and control, aspects of Load Manager. Customers can easily upgrade from Load Monitor to Load Manager at any time by simply purchasing an upgrade license key. Throughout this paper we will only refer to Load Manager, but unless noted otherwise the information all applies to Load Monitor as well.

2 Load Manager Architecture

A complete Load Manager deployment consists of three components; LM Console, LM Director and LM Service. These components are interconnected as shown in **Figure 1** and together provide a comprehensive work load management solution.

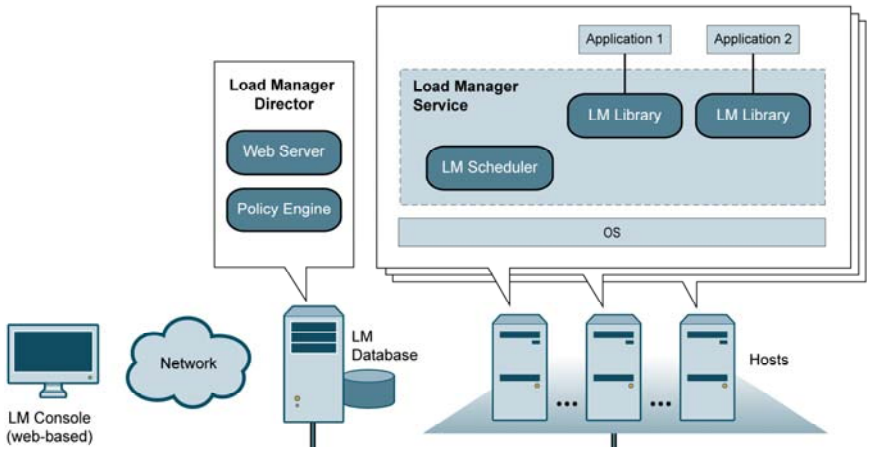


Figure 1: Load Manager Architecture

2.1 Load Manager Service

Load Manager (LM) Service refers to the Librato software that is running on each host under workload management. A key characteristic of LM Service is that it runs 100% in user space and doesn't require any modification to the underlying operating system or the applications under management. LM Service in total requires less than 10MB of system RAM and typically introduces less than 1% performance overhead.

LM Service is subdivided in to two main components, LM Library and LM Scheduler, which both also operate in user space.

LM Library is a Linux shared library named *liblm.so*. As shown in **Figure 2**, LM Library sits between an application (and any associated middleware such as a JVM) and the C standard library (libc). LM Library intercepts all OS system calls that are related to resource consumption such as read/write and brk/mmap.

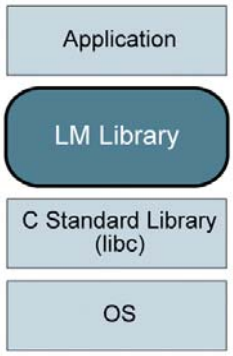


Figure 2: LM Library

LM Scheduler is a policy-based, fair share scheduler that can provide system resource quotas for large numbers of applications running on an individual host. There is only one LM Scheduler running on each managed host but it interacts with many LM Libraries (for each application that is being managed).

In order to provide fine grained application control, LM libraries periodically exchange data with LM Scheduler to determine the optimal resource quota to give each application. **Figure 3** shows a step-by- step description of the interaction between one LM Library and LM Scheduler.

Application Processes

For ease of explanation we refer to the control of "applications", but to be more specific Load Manager controls and monitors at the process (and thread) level. If an application is comprised of multiple processes (e.g. Oracle database), Load Manager can control all of them as a group (we call this a Container) as if they are one entity.

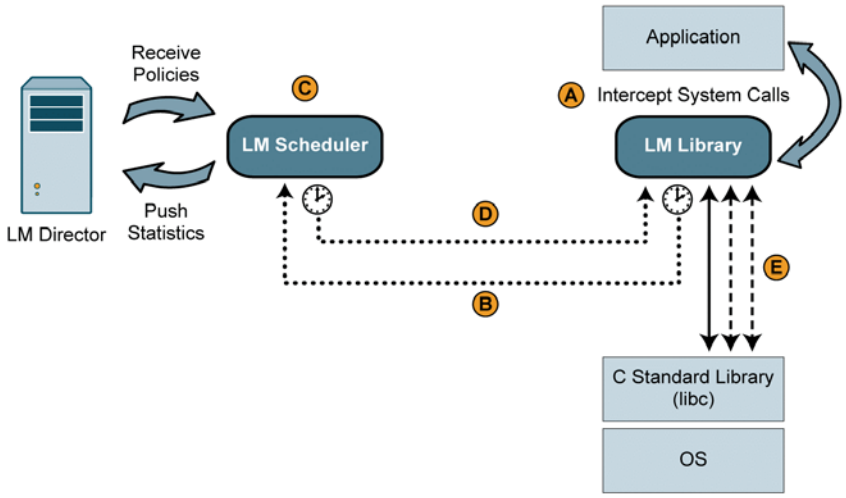


Figure 3: Load Manager Service Interaction

- (A) Intercept resource consumption related system calls such as read(), write(), mmap() or brk()
- (B) LM Library periodically communicates application resource usage to LM Scheduler
- (C) LM Scheduler's scheduling algorithm takes the usage statistics it receives from all applications and combines them with business policies to generate an optimal resource quota for each individual application.
- (D) LM Scheduler periodically communicates application resource quotas to LM Library
- (E) LM Library controls application behavior by, for example, breaking up a single system call in to multiple system calls

To better understand how LM Service controls system resource usage, let's take an example and assume that a policy has been set which limits *App1* to a maximum of 100Mbytes per second of file reading and *App1* makes a request to read a 200Mbyte file via the read() system call. Without Load Manager a single read system call requesting 200Mbytes of data would be issued to the OS and for a period of time *App1* would overuse storage I/O resources resulting in degraded performance of other applications on the same

server. With Load Manager instead of a single read() system call, LM Library would break up the 200Mbyte request in to a series of properly spaced, smaller read() system calls. For example, a read call of 1Mbyte every 10ms resulting in the allocated 100MBytes per second. This same concept can be extrapolated to CPU, memory and network I/O to better understand how Load Manager can guarantee pre-defined usage policies or SLAs across all applications.

Communication between LM Library and LM Scheduler occurs as often as hundreds of times per second and as a result, resource allocation can be dynamically rebalanced in real time based upon actual application demand. For example, if a high priority application called *App1* has been assigned a guarantee or allocation of 80% of the CPU but is currently not using that much, LM Scheduler will allow another application, *App2*, to utilize the spare capacity. However, if *App1* suddenly demands 80% of the CPU it will receive that much instantaneously.

In the event that LM Scheduler fails load management will continue, however there will no longer be any dynamic rebalancing. Using the previous example, *App1* would be statically assigned 80% of the CPU whether it used it or not and *App2* would be statically assigned 20% and would not be able to use more even if spare capacity was available. This situation would only occur for a brief period of time because LM Scheduler will be automatically restarted by a watchdog process within LM Service that is itself started by the Linux *init* process.

Finally, LM Scheduler aggregates collected statistics from all applications and periodically (default is every 60 seconds but is user configurable) pushes them to the LM Director database.

2.2 Load Manager Director

LM Director is a dedicated machine that serves as a central management point for all load managed hosts. One LM Director can scale to accommodate over 2,000 load managed hosts and can be federated to support even larger deployments. LM Director consists of a policy engine, management software, database and web server.

Load management policies are defined on LM Director and pushed out to each LM Service for execution. LM Director in turn receives raw resource usage statistics from each LM Service every 60 seconds (interval is user configurable) and stores it in the database. LM Director summarizes or averages the raw statistics every hour and stores these in the database. Based upon user configuration, raw statistics that are over 1 hour old can be archived before they are deleted. For archiving an administrator can either use the tool provided with LM Director or use his/her own tool. If archiving is enabled and LM Director's tool is used, before statistics are deleted they are compressed and stored to a specified location as a series of SQL insert statements. The archive can then be easily uncompressed and directly inserted in to any SQL-92 compatible database. Database-specific table creation (schema) scripts are provided for Oracle, MySQL, Sybase, DB2 and PostgreSQL. By default LM Director uses a local PostgreSQL database, it can however be easily configured to use other common databases.

The data in LM Director is used by LM Console to provide a dashboard style view of system resource usage at the host or container level. This data can also be used as input to a graphical or analytical tool (e.g. Crystal Reports) to generate detailed trend analysis information which can be used for capacity planning, application profiling or chargeback billing purposes.

All communication between LM Director and each LM Service is secured using Kerberos authentication. This is important to ensure that only a valid LM Director can communicate with an LM Service and to ensure that a rogue server cannot provide statistics updates to LM Director. LM Director comes with a Key Distribution Center (KDC) as part of the RPM install to make setting up the Kerberos authentication as seamless as possible. LM Director and LM Service can also integrate seamlessly with existing KDC infrastructure.

2.3 Load Manager Console

LM console is a web-based graphical user interface accessible using a standard web browser. LM Console communicates with LM Director via HTTP (HTTPS is available as a configuration option). LM Console provides a unified view of system resource utilization across all managed hosts via a dashboard and provides a graphical interface to set or edit load management policies.

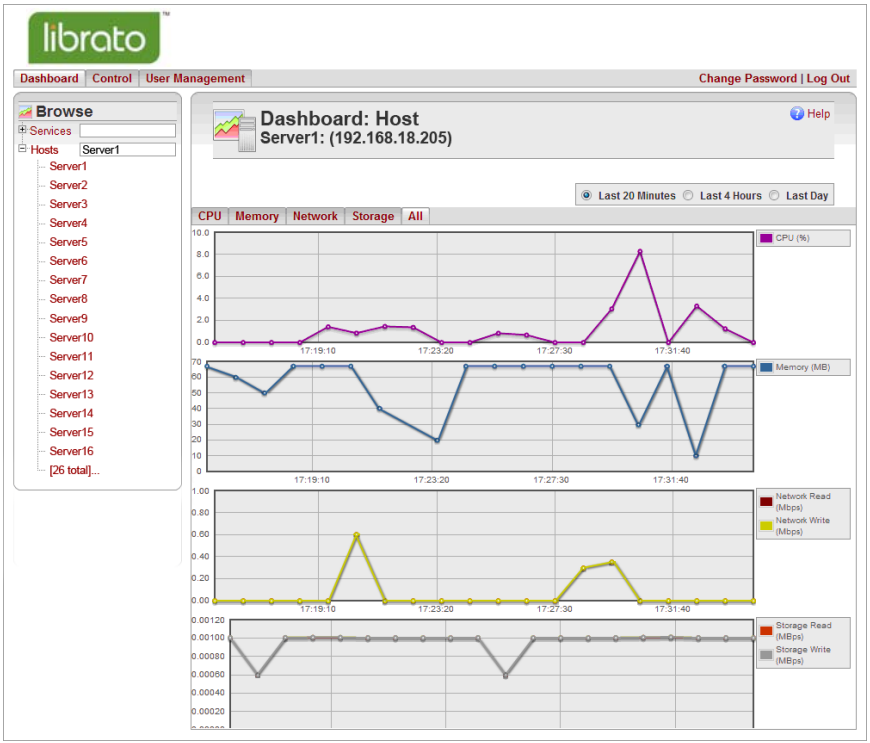


Figure 4: Load Manager Console

3 Key Concepts

3.1 Axes of Control

In Load Manager terminology, an axis represents a physical resource that can be monitored and controlled by Load Manager independently of all other axes. Each axis is monitored and controlled on a per-process (or thread) basis. The supported axes for monitoring and control are:

- **CPU** — The consumed CPU cycles, measured as a percentage of a single CPU core. An n-core machine is capable of providing (n*100)%, for example a dual-socket, dual-core machine has a total of 4 cores and hence 400% of CPU available.
- **Virtual Memory** —The quantity of virtual memory, measured in megabytes (MB).
- **Network Read Bandwidth**—The rate of reading data from TCP/IP-based BSD sockets, measured in megabits per second (Mbps).
- **Network Write Bandwidth**—The rate of writing data to TCP/IP-based BSD sockets, measured in megabits per second (Mbps).
- **Storage Read Bandwidth**—The rate of reading data from storage, measured in megabytes per second (MBps).

- **Storage Write Bandwidth**—The rate of writing data to storage, measured in megabytes per second (MBps).

3.2 Bounds, Containers and Allocation

The concepts of *bounds*, *containers* and *allocation* are best described by an example. We will use the following example to illustrate all three concepts.

Assume we have a host named *Server1* that is a dual-socket, dual-core machine which means it contains four CPU cores. Furthermore assume there are two applications running on Server1 called *App1* and *App2*. *App1* is a high priority foreground application that consists of two processes, *ProcA* and *ProcB*. *App2* is a low-priority, background application that consists of one process, *ProcC*.

A *bound* is the overall usable capacity of a host that is available for Load Manager and is specified across all four axes. One can think of bounds as a means to characterize the physical limits of a host with respect to CPU, memory, network I/O and storage I/O. In our example, *Server1* has four cores and hence the maximum CPU bound for this host is 400% (100% x 4 cores). An administrator could choose to set the CPU bound to 300% in which case Load Manager would only allocate three-fourths of the available CPU cycles. Similarly host bounds can be set independently for memory, Network I/O and Storage I/O. Load Manager attempts to automatically discover the available bounds on each host, but an administrator should check the discovered bounds to make sure they are accurate and meet requirements. **Figure 5** shows the bounds that have been defined for *Server1*. Network I/O is set to 1000Mbps indicating a Gigabit Ethernet connection and Storage I/O is being limited to 100MBps for both read and write. Finally, memory shows a bound of "unlimited" which means that memory is not available as an axis of control. In other words, memory usage for an application can only be monitored and not controlled in any way.

Unlimited Bounds

If all bounds for a host are set to "unlimited" this means that only monitoring of resource usage is allowed and this would effectively equate to the Load Monitor product.

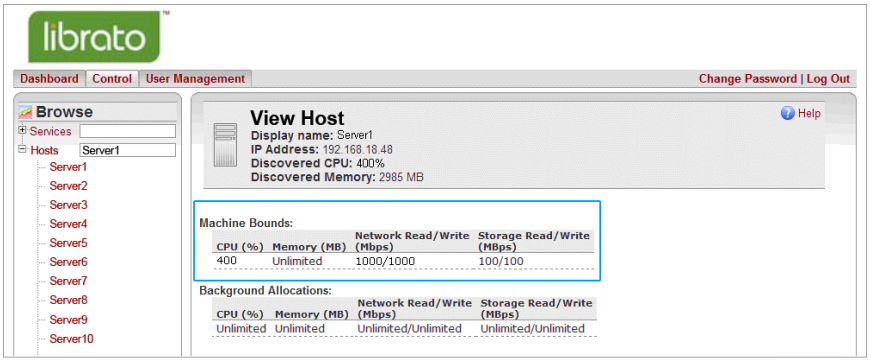


Figure 5: Bounds

Once bounds for a host have been established, the next step is to divide the host into *containers* which are a user-defined, logical unit of control and are the fundamental building block for Load Manager. Each load managed host is carved up into one or more containers based upon the desired level and granularity of control. Each container is assigned one or more processes which are to be controlled equally under group policy. Although administratively individual processes are assigned to a container, the most common practice is to assign an entire application (e.g. Oracle database, HR portal) to its own container thereby

enabling fine-grained control over the system resources that are allocated to that application. Since most modern applications are comprised of multiple processes, an administrator will most likely wind up assigning multiple processes to a given container. If a process is assigned to a container and it spawns a process, the child process will be automatically included in the container.

All administrative policies are defined at the container level and therefore any application that is to be controlled must be assigned to a container. Often there is only one application per container, but in the event there are multiple applications they will all be treated as one entity from a resource allocation perspective. In our continuing example, we have two applications and in order to control them independently we define two containers: *Container1* and *Container2*. Since *App1* is a high priority application we will assign it to *Container1* and assign *App2* to *Container2*.

Background Container

By default each host is allocated a container named Background which contains all managed processes that have not been explicitly assigned to another container. In our example, if either App1 or App2 were not explicitly assigned to a container, they would have been automatically assigned to the Background container.

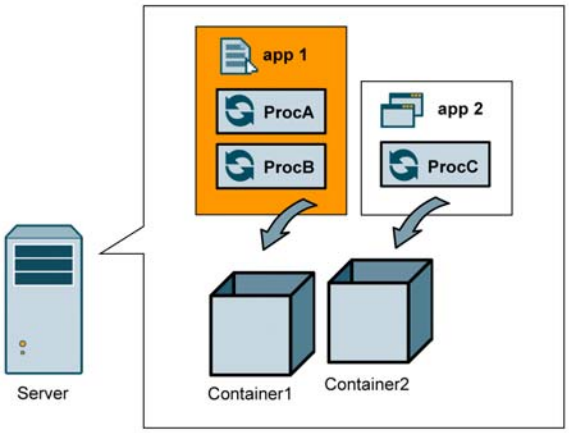


Figure 6: Containers

Container policy settings are typically assigned by an administrator using the graphical interface but can also be set via the command line interface (CLI).

After containers have been defined the final step is to allocate resources to the containers. We use the term *allocation* to refer to the specified quantity of resources that are guaranteed to a container. For our example, let's assume that we guarantee *Container1* 360% (recall the maximum is 400%) of the CPU and *Container2* only 40%. This means that no matter what happens *Container1* (and by association *App1*) is guaranteed to receive at least 90% (360/400) of the available CPU cycles across all four cores. Conversely, *Container2* (*App2*) is only guaranteed 10% (40/400) of the CPU cycles. This does not mean however that *Container2* will never get more than 10% of the CPU cycles. Because of Load Manager's ability to dynamically rebalance resources, if *Container1* is not using 90% of the CPU cycles the excess capacity can be consumed by *Container2*. However, if *App1* suddenly demands 90% of the CPU cycles it will receive that much instantaneously even if that means throttling down *App2*.

Allocations can be set independently across any of the four axes and the sum of the allocations cannot exceed the machine bounds.

In addition to an explicitly specified value, allocation can also be assigned two special values:

Unlimited - Disables any resource guarantees. A common use for this special unlimited *allocation* is for deploying Load Manager as a monitoring-only solution by setting all *allocations* system-wide to unlimited.

Fair Share - Guarantees a shared slice of whatever resources are not explicitly guaranteed to other application processes. Resources are shared equally among all processes that are assigned *Fair allocation.* For example, assume *App1* is assigned 70% of the CPU and there are two other applications, *App2* and *App3 that* are designated as fair share. These applications would evenly share the unassigned 30% of the CPU and hence get 15% each. If another application, *App4*, was also assigned fair share each of the three **applications would now get 10% CPU each.**

3.3 Load

Load is the amount of resources consumed by a host or container. The LM Console dashboard shows graphs of host or container load over time. Fine-grained load statistics are captured in the LM Director database and are very useful for monitoring resource usage, capacity planning and chargeback billing.

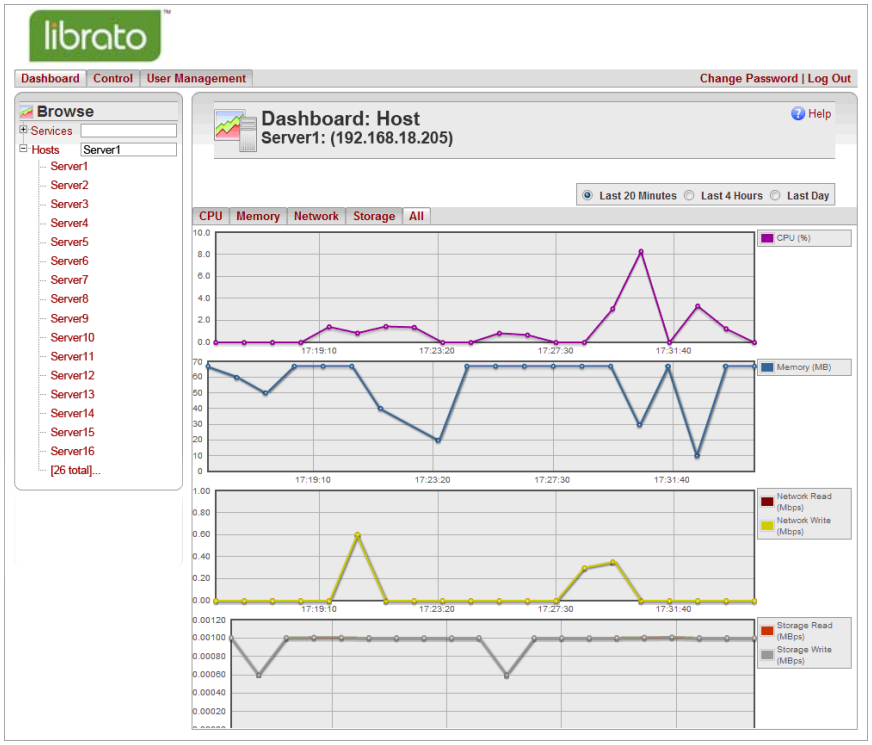


Figure 7: Load

3.4 Dynamic Resource Rebalancing

A container allocation is a guaranteed amount of resource that will always be available whenever it is needed. For example, in **Figure 8** *Container1* is allocated or guaranteed 80% of the available CPU capacity. However at most times *Container1* is not using exactly 80% of the CPU and hence this “spare” capacity is available for other containers to use. Load Manager optimally rebalances unused system resources (i.e. CPU, memory, network I/O, storage I/O) between containers while always guaranteeing that a container will receive its allocation whenever demanded. In **Figure 8,** for 3 minutes from 4:13am to 4:16am *Container1* is using less than 80% CPU and hence during that period unused CPU capacity is dynamically reassigned to other containers. At 4:16am the situation reverses with *Container1* demanding its allocation and more (if

capacity is available). During that period spare capacity was available from other containers so *Container1* was able to exceed its 80% allocation. However, at 4:17am though *Container1* demanded more than 80% CPU capacity it was capped because no spare capacity was available. In other words, at 4:17am all containers on the host were using their full CPU allocation.

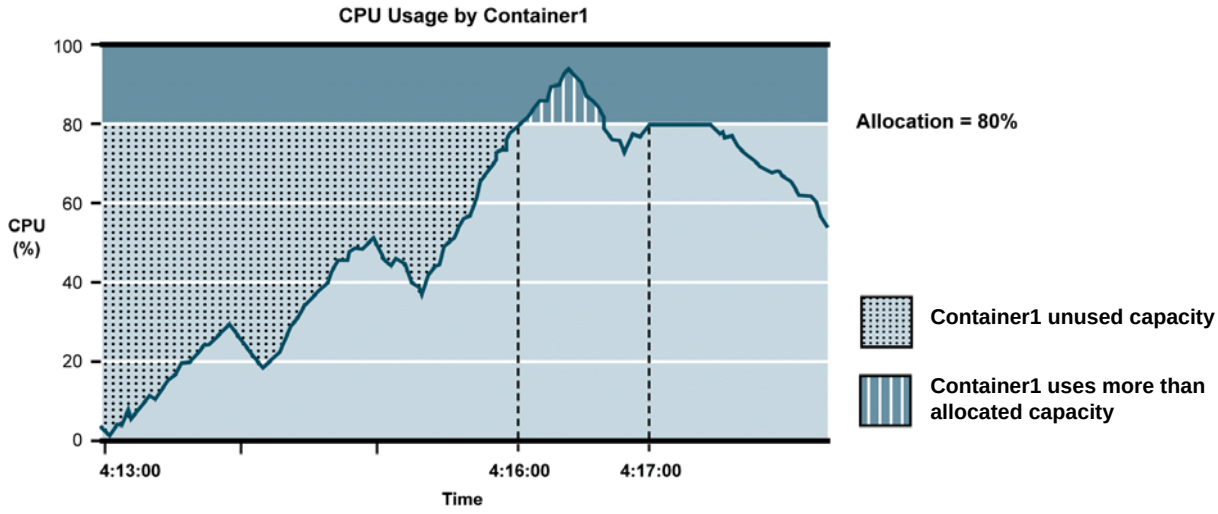


Figure 8: Dynamic Resource Rebalancing

Dynamic resource rebalancing is a key Load Manager capability because it provides optimal resource utilization while always strictly enforcing defined business guarantees or SLAs.

4 Common Use Cases

Load Manager is used in a variety of different ways, below are three examples of common use cases for the product.

4.1 Sponge

Many companies run their business critical applications, such as a financial trading or online transaction processing app, on a dedicated server so they can guarantee that the application will **always** get the system resources (e.g. CPU, memory etc.) it requires. The by-product of this decision is a large number of servers that are woefully underutilized, resulting in higher than necessary capital and operational expense.

Load Manager provides an alternative approach which **guarantees** a business critical application all the system resources it requires when it requires them, while allowing background applications (e.g. grid engines) to run on the same server but only take advantage of unused system resources. We refer to all the background tasks as “sponge” applications because they safely “absorb” all unclaimed system resources and allow companies to “squeeze” out the maximum possible value from their server infrastructure. Load Manager is able to guarantee system resources to the foreground or business critical app because it samples resource consumption at a very high rate (up to 100 times per second) and dynamically reallocates resources from a low priority “sponge” application to the high priority, foreground application the instant they are needed.

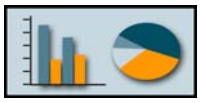
4.2 Database Consolidation

Most database administrators (DBAs) are reluctant to run more than one database instance on the same server because spikes in workload on one database can prolong the response time of queries on the other

to unacceptable levels. With Load Manager, databases can be safely consolidated by forcing resources to be scheduled on a “fair share” basis. The DBAs can also assign priorities among the database instances so mission critical and other important applications can always meet SLAs. For example, a DBA could run an online transaction processing (OLTP) database and a decision support system (DSS) database on the same server. The OLTP database supports a response time sensitive application so the DBA could assign 80% of the resources to the OLTP database and 20% to the DSS one.

The nature of the OLTP workload is that it is only busy for short periods of time, so Load Manager could dynamically rebalance system resources to the DSS database during periods when the OLTP database doesn’t need them. Of course as soon as a query is issued to the OLTP database, LM would guarantee it had all needed system resources so business level SLAs are met.

4.3 Monitoring



Precise, fine-grained information about application resource usage is an invaluable tool for IT managers to make planning and billing decisions. With Load Monitor (monitoring only version of Load Manager) IT managers can get detailed reports about their application behavior down to the process or even thread level. Information across all four axis such as current load, maximum over time, or average is captured at the host, container or service level. The collected data can then be used for a variety of purposes including the following three:

- **Capacity Planning**

  The precise system resource (e.g. CPU, memory etc.) usage by each application (or individual process/thread) can be used to determine which applications can and cannot be run effectively on the monitored host. IT managers use this information to validate assumptions they have made or determine which applications need to be relocated to another physical server or even a virtual server.

  In addition to resource usage data, Load Monitor also provides demand monitoring data or in other words how much of each system resource a particular application requires or demands. This quickly alerts IT managers to applications that are underperforming and could benefit from additional allocation of resources.

- **Application Profiling**

  Comprehensive data about each individual process (or thread) in an application can be used to profile the precise behavior of each component of the application. This information can be used by IT managers for a variety of purposes such as to identify processes that are using an unexpectedly high or low amount of a system resource (e.g. memory).

- **Chargeback Billing**

  Monitoring data can be used to provide detailed information about which user, group or organization used how much of a given resource (e.g. CPU hours) over what time period. With this information IT managers can provide detailed “cell phone like” billing data for internal or external billing purposes.

5 Load Manager Deployment

5.1 LM Service

The LM Service software must be installed on each host targeted for workload management. The installation process is very streamlined and on average takes less than 10 minutes for an experienced administrator. All LM Service software is contained in a single RPM package which enables automated installation across a large number of servers. The Command Line Interface (CLI) is contained in a separate

RPM package that is optional and need only be installed if you wish to use CLI commands on the managed host. After installing the RPM package(s) the remaining procedure is as follows:

1. Copy the Kerberos configuration file from the Load Manager Director's KDC or from your KDC if you are using your own. The configuration file typically contains Kerberos information, including the locations of KDCs and administrative servers for the Kerberos realms of interest, defaults for the current realm and for Kerberos applications, and mappings of hostnames to Kerberos realms.

   Communication between the LM Director and each LM Service is kerberized for security purposes.

2. Edit a configuration file (`/etc/load_manager/lmd.conf`) to enable secure communication between this LM Service and the designated LM Director. At a minimum the following variables must be defined.

Variable	Description
`LM_SERVER_HOST`	Specifies either the numerical IP address or resolvable host name of the Load Manager Director
`LM_SERVER_PRINCIPAL`	Specifies the name that the Load Manager Director will use to authenticate itself using Kerberos to the Load Manager Service.
`LMD_KEYTAB`	Identifies the location of the keytab file that is used when authenticating the Load Manager Service using Kerberos to the Load Manager Director
`LMD_PRINCIPAL`	Identifies the principal name that this LM Service uses when authenticating to the Director.

3. Edit /etc/inittab so that LM Scheduler is automatically started each time the system is booted.

4. Add LM Library (liblm.so) to /etc/ld.so.preload to run all applications on the host under Load Manager. This action instructs the linker/loader (ld) to insert LM library whenever it links and executes an application.

Note

Although it is recommended that all applications on a host be placed under Load Manager control, there is a mechanism to selectively enable load management on an application by application basis.

The $LD_PRELOAD environment variable can be used (LD_PRELOAD = liblm.so) to instruct the dynamic linker/loader (ld) to insert the LM Library for an individual application. This environment variable would typically be set in the application startup script. For example, to start an Oracle instance under Load Manager control:

export LD_PRELOAD=liblm.so

sqlplus <user>/<password> as sysdba

sqlplus> startup

5. Reboot the machine

6. Edit the startup script for each application under workload management to include the LM_CONTAINER_NAME environment variable. This variable is set with the container you wish to place the application in and the service name you want to assign to that application. For example, assume you had an application named “Payroll” and you wanted to put it in a container named “Critical”. The environment variable would be defined as follows:

   LM_CONTAINER_NAME = Payroll::Critical

5.2 LM Director

LM Director is a dedicated Linux server used to manage multiple LM Services and should not be used for any other purpose. LM Director should not be hosting any other applications or databases. The hardware and software requirements for LM Director are as follows:

Hardware

Platform: x86 or x86_64
Memory: 2GB minimum (4GB recommended)
Disk space: 200GB minimum (recommend 250GB)

Operating System

LM Director has been tested on a variety of Red Hat and SUSE Linux distributions, but for optimal performance we recommend one of the following.

- Red Hat Enterprise Linux 4.6 (64-bit)
- SUSE Linux Enterprise Server 10 with SP1 (64-bit)

Also, the following RPMs must be pre-installed on the system for Kerberos and to use the default PostgreSQL database.

For Red Hat Enterprise Linux:

- krb5-server
- krb5-libs
- postgresql-server
- postgresql

For SUSE Linux Enterprise Server:

- krb5-server
- krb5
- postgresql-server
- postgresql

LM Director comes as a self-extracting installer that contains two RPM packages with all required components. Simply install both RPMs and modify a few settings and LM Director is ready to manage your server infrastructure.

6 About Librato

Librato is a leader in application workload management software. Its flagship Load Manager product helps deliver predictable application performance and optimal resource utilization for x86 platforms running Linux and Windows on physical or virtual servers. It enables our customers to reduce expenses by continuing to meet business goals with fewer data center resources.

In addition, Load Manager provides fine-grained, application level monitoring of resource usage, which can be used for functions such as capacity planning, chargeback billing and application profiling.

The company's unique and patented technology is completely application transparent and works with standard, unmodified operating system distributions. This allows for easy deployment in existing data center infrastructures, and a rapid return on investment by reducing the required hardware and software resources and the associated operational expenses for operating and managing these resources.

Librato is a privately held company with headquarters in Santa Clara, CA and offices in Blacksburg, VA and Pune, India.

Contact:

US: 1 866.921.4743
International: +1 408.588.1716
sales@librato.com
www.librato.com