Qlusters

Automating Data Center Management with openQRM

A Technical Overview

Qlusters, Inc.
1841 Page Mill Road
Palo Alto, CA 94034
www.qlusters.com

March 2006

Qlusters

Introduction **3**

Architecture and Framework **5**

Deployment Architecture 5

Conceptual Framework: Virtual Environments.. 7

Automation **10**

The Components of an Automation System 10

Automation in Action 11

Agility **15**

Centralized Image Management 15

Availability **20**

Selecting a High Availability Solution Based on Recovery Objectives 20

Option 1: Hardware High Availability 22

Option 2: Application High Availability 23

Summary **25**

Qlusters

Introduction

Today's data centers focus on executing business processes at the lowest possible cost, while at the same time fulfilling service level objectives for availability, performance and agility. A greater emphasis on costs in recent years has driven data centers to migrate from proprietary UNIX systems to industry-standard server hardware and Linux operating systems.

This migration to Linux on standard hardware lowers server acquisition costs dramatically, but introduces other costs and challenges to the data center. For example, the number of servers expands rapidly — Gartner predicts 20% annual growth of Linux systems through 2008[1]. Like large multiprocessor servers before them, commodity servers and operating systems typically possess a "static nature," with the identity of an application or service tied to a single physical system. The complexity of managing these static relationships grows exponentially as the number of Linux servers grows. Data centers face the dilemma of either adding staff to manage more servers and greater complexity, or compromising on service level objectives.

Today's data centers need a software-based solution that can leverage the cost advantage of commodity, standards-based computing without adding administrative costs or compromising on the high availability, performance and agility capabilities offered by proprietary legacy systems. openQRM provides these capabilities — enabling the next step in data center operations, while offering dramatic benefits:

- **Automated IT Operations**: data center where the IT group establishes service levels, rules and policies, and the management platform monitors and discovers computing resources and automatically provisions or decommissions resources from applications or services to enforce policies.
- **Centralized Management**: a single view of all systems that comprise a flexible and scalable pool of resources, with a central repository of boot images for the online services.
- **Dynamic Server Provisioning**: simplified and rapid provisioning of systems to deliver new or improved services to customers, or address new business opportunities.
- **Superior Availability**: an array of solutions for providing high availability, ranging from an extremely efficient and easy to implement mechanism for making applications highly available, to a solution for mission-critical applications that approaches an "always on" objective.

[1] Gartner Research 2003.

- **Reduced Costs**: through greater utilization of server hardware, reduced staffing cost per server and superb service levels.

This paper summarizes the technologies that enable open to deliver agility, high availability and low cost and describes:

- how the technologies work
- their ability to support operational agility and high availability
- the importance of the technologies
- how they fit into a data center environment

Qlusters

Architecture and Framework

This section covers the architecture of an openQRM deployment and its conceptual framework. The deployment perspective demonstrates how openQRM fits into the overall data center environment and relates to the other elements. The conceptual framework provides an overview of how openQRM presents the environment to its users: IT administrators and managers.

Deployment Architecture

The openQRM deployment architecture is depicted in Figure 1 with the central elements being:

- the openQRM server
- the pool of computing resources
- storage
- network

openQRM server

The openQRM server is the hub of provisioning, monitoring, and automation of the Linux data center. However, it is important to understand that the openQRM server is not part of the operational or data path for any of the computing resources. It does not add any overhead to the operation of the computing resources, does not sit between the resources and their storage, and has no impact on the operation should it itself experience a hardware or software failure.[2]

[2] A system provisioned by Qlusters Resource Manager is not dependent on the QRM server for its operation. That is, the QRM server is not a single point of failure. Furthermore, when the QRM server is deployed with high availability enabled, a system failure only briefly affects monitoring and other activities.

Qlusters

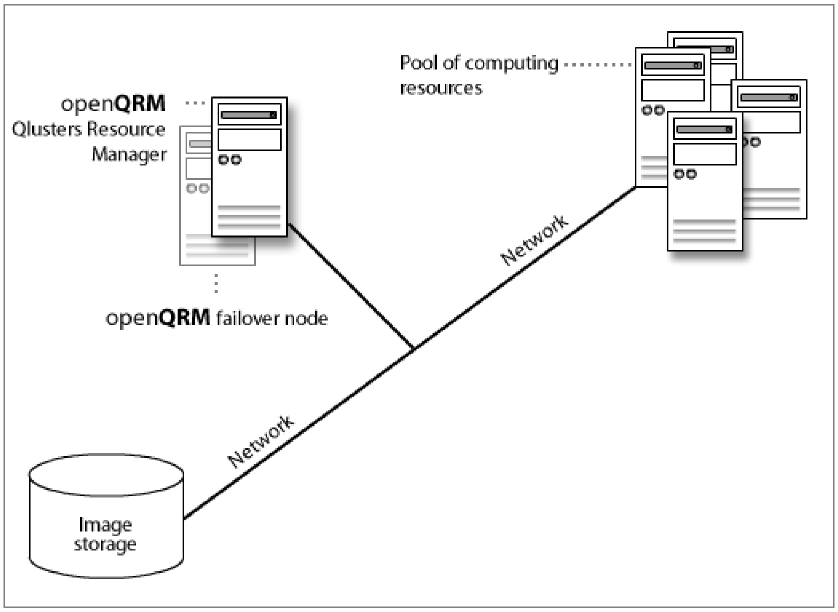

Figure 1: openQRM deployment architecture.

Resource Pool

The openQRM server manages a pool of computing resources: standards servers running Linux. openQRM maintains an inventory of information about each server (e.g. processor type, number of CPUs, memory size, etc.). This information allows openQRM to automatically select an appropriate server when a pre-defined policy requires that a new server be deployed.

The servers can be added to openQRM's resource pool simply by connecting power and network. openQRM automatically performs all tasks needed to take a new server from a non-operational to an operational state.

Provisioning, Storage and Network

Provisioning, preparing and providing a server's software environment, is a fundamental piece of openQRM's functionality. Using network boot technology and centralized storage of operating system and application images, openQRM provisions a server in the time it takes to perform a single reboot. All system images are kept in central NAS storage on the network. The NAS storage may operate as a SAN head, thus allowing images to be stored in a SAN. A complete high availability solution will require that the storage system itself be designed to be highly available through the use of redundant disks and networking.

The individual server nodes connect to the image storage and openQRM server using traditional networking technology. High-speed interconnects provide an advantage for some environments, but are not a

Qlusters

requirement. openQRM can support a variety of network configurations and does not require any modifications to the network architecture.

Servers often connect to the primary network via two separate network interface cards (NICs) that connect through different network switches. This prevents a single NIC failure from making a server unavailable, as well as keeping a single switch failure from disabling an entire pool of servers. openQRM does not require configurations of this kind; however, they are an essential part of a highly-available environment.

Conceptual Framework: Virtual Environments

openQRM employs a concept known as a Virtual Environment to define a unit of functionality for the data center. This unit of functionality can be an application or a service, such as:

- A Web farm
- A database instance
- A trading application
- A DNS server

Through openQRM, an administrator defines Virtual Environments corresponding to the infrastructure needs of the business units served by the data center. Rather than choosing and deploying individual systems, installing software on each, configuring high availability, and doing the many other tasks required to provision and effectively operate an application, the IT administrator simply creates the Virtual Environment that describes the application. openQRM uses the Virtual Environment definition to do all the work required to deploy and operate the application or service.

The administrator specifies the Virtual Environment by describing what is necessary to run the applications it is responsible for, including:

- Software required
- Type and quantity of server resources
- High availability requirements
- Other policy requirements

For example, the IT administrator would define a Virtual Environment corresponding to a Web farm by specifying each of these components:

- **Software**: The administrator performs a one-time install of the full complement of software on a server. openQRM provides a mechanism for creating an image of the installation and placing it into the central image storage mentioned above. Two images are created: one encapsulating the operating system, the other a root file system that typically contains the applications and related data. (See figure 2.) This reduces the number and complexity of images required for applications that share a common operating system version.

- **Server Characteristics**: openQRM uses the Virtual Environment definition to choose appropriate physical servers on which to deploy the Virtual Environment. For example, if the Virtual Environment requires a minimum of a 2GHz server with 1GB of RAM, openQRM will consult its inventory of systems to find compatible machines.

- **Dynamic Policies**: The administrator describes the requisite policies associated with the Virtual Environment. This includes the necessary collection of servers to implement this application, and the mandated policies or rules. At a Web farm, the administrator might decide that there should always be a minimum of three servers but openQRM may add up to a total of 10 servers based on a predetermined maximum (and possibly minimum) load per system. Other external events can be used to change the configuration of the environment via the openQRM API. (See the automation section below).

- **High Availability Requirements**: The administrator can select the most appropriate high availability policy for a given application. For example, the administrator of a Web farm may choose simply to replace any failed server with a new one.

- **Tags**: Some information about the servers cannot be discovered automatically, but should still be considered when matching physical machines to Virtual Environments. To do this, the administrator may associate textual tags with physical servers. For example, a Web farm administrator might create a tag corresponding to a position in a network topology, such as inside the DMZ. A Virtual Environment definition can, in turn, require that a certain tag be present in its servers. When openQRM looks for physical servers to use for a Virtual Environment, it will consider the tag requirements of the Virtual Environment as well as other specified server characteristics.

openQRM relies on the administrator to define the Virtual Environment. It then automatically maps this definition onto the physical resources in the deployment environment so that the appropriate computing resources can be assigned as needed.

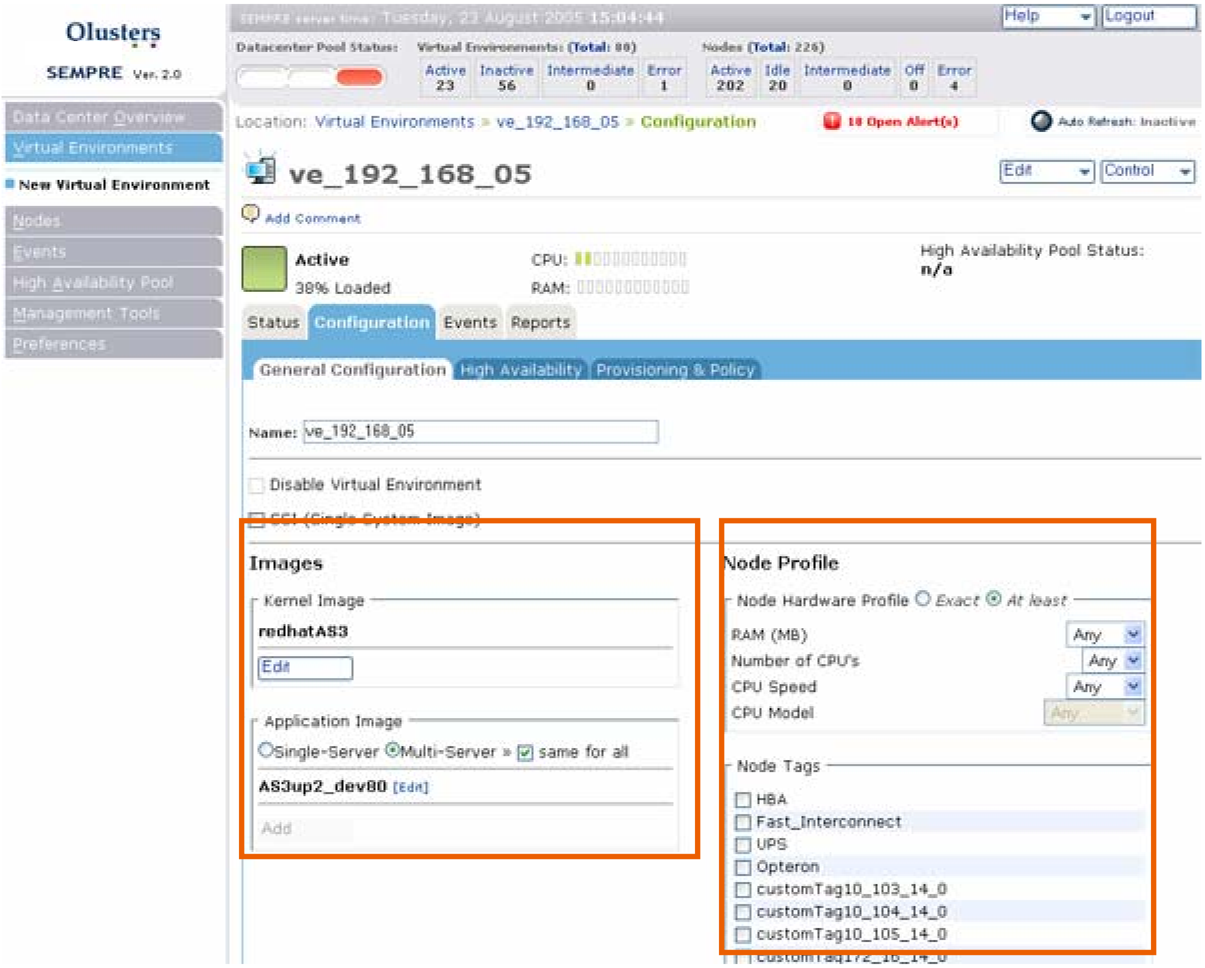

Figure2: Virtual Environment Definition with applications images and physical server requirements highlighted.

Qlusters

Automation

As applications become larger and more successful, the number of physical servers required to support each application can be expected to grow. In order to support ever larger data centers without a proportional increase in staff, each employee must be able to manage more equipment. To handle more work without increasing the likelihood of error requires an automated infrastructure. Operators administer classes of servers instead of individual machines.

The Components of an Automation System

Three major components deliver effectiveness within an automated infrastructure:

- A means of specifying the actions to be automated and dictating when those actions should be taken (i.e. policies). The subsystem that implements these policies is often referred to as the controller.
- A mechanism that can implement the required actions, such as provisioning a new server. These mechanisms are often referred to as effectors.
- A monitoring mechanism that provides the information necessary to make decisions to a policy-based automation component. These monitoring components are often referred to as sensors.

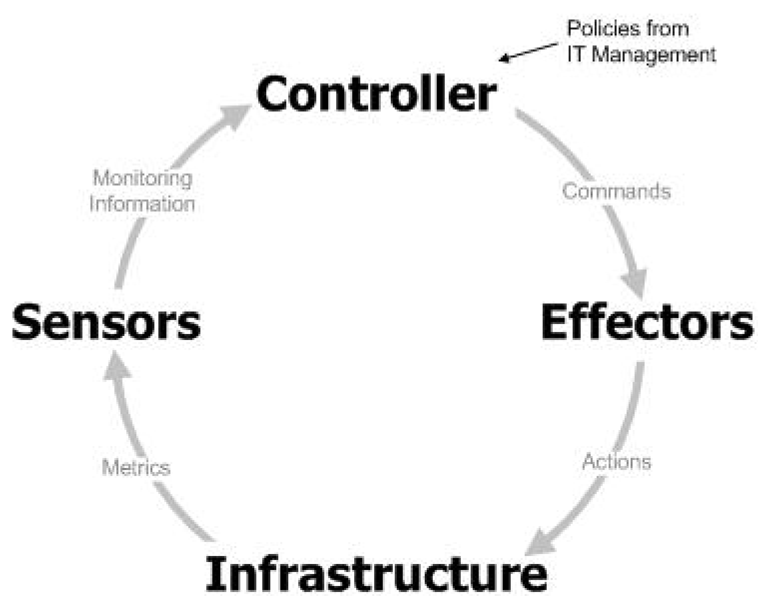

Figure 3: Automation components

Policies that are specified by IT management are mapped to rules that determine what actions are taken based on the dynamic state of the system. The controller evaluates the rules and requests that the effectors take appropriate actions. The sensors monitor the dynamic state of the system and feed back the appropriate information to the controller.

For example, suppose an IT manager specifies a policy that indicates that the processor load on a certain application should never exceed 80 percent. The sensors monitor the load on the server that runs the application and feed that data to the controller. Once the load exceeds 80 percent, the controller identifies a policy violation. The controller determines the specified action, which may be to provision an additional server and make it available to the application. The controller then instructs the appropriate effector to perform the provisioning step and, as a result, the infrastructure is modified to suit the current dynamic workload requirements.

openQRM provides all of the components needed for an effective automation solution. Its automation capabilities include:

- A controller
- An interface for creating policies
- A set of effectors,including the server provisioning engine
- Monitoring agents

Monitoring agents are deployed on each of the provisioned servers, acting as a kind of sensor array. Metrics from the sensors are fed back to the openQRM server, where pre-defined policies are put into action based on the information received. The combination of these components and their tight coordination creates a powerful and flexible system.

Automation in Action

The following scenario describes openQRM automation in action.

1. Installing a new machine

When an IT administrator attaches a new system to power and the network, the system performs a network boot using the Pre-Boot Execution Environment, better known as PXE.[3] PXE actually involves two logically, if not physically, distinct software services: a DHCP service that provides the new server with its network address and a tftp server that provides its boot image. The openQRM server handles both roles.

Responding to the PXE boot request, the openQRM server provides the new machine with a standard Linux kernel and an inventory agent. This agent collects information about the new machine, such as the number

[3] PXE is a de facto industry standard for network booting.

Qlusters

of CPUs, CPU type, and memory configuration, and reports that information back to the openQRM server. openQRM keeps this new machine's specifications in its inventory of servers.

This discovery process is repeated each time a new server is added to the network. openQRM maintains an up-to-date inventory and profile of all server resources.

openQRM technologies:

- Network boot support
- System discovery
- System profiling
- System inventory

2. Defining a Virtual Environment

The IT administrator uses openQRM to define a Virtual Environment. As previously described, the Virtual Environment defines not only the application and operating system, but also the availability requirements of the application. The availability requirements are only one type of policy.

In this case, the user selects a Hardware High Availability policy. This policy states that if a server fails, it should be replaced by another server and the application should be restarted.

openQRM technologies:

- Virtual Environment definition, including policies

  (In this case, the policy is related to high availability.)

3. Instantiating a Virtual Environment

Once the Virtual Environment is defined, the administrator can instantiate it through the openQRM user interface. At this point, the controller determines what needs to be done to make the Virtual Environment active. For this example, we will assume that only one physical server is needed. The controller matches the requirements of the Virtual Environment against its inventory of available servers and chooses the best fit. It instructs the openQRM agent on the selected server to perform an orderly shutdown followed by a network boot.

Qlusters

The selected server performs these tasks, and upon reboot, the openQRM server responds to its PXE request. At this time, it provides the operating system specified in the Virtual Environment definition. The openQRM server instructs the system to mount (via NFS) the root file system image that corresponds to the application specified in the Virtual Environment. Additional specialization steps can occur at this point as well.

Along with the application itself, an openQRM monitoring agent is initiated on the newly provisioned server. This agent acts as a sensor, reporting back both health and performance information.

openQRM technologies:

- Server provisioning (including remote system reboot, network boot, and NFS mounted root file system image)
- Monitoring agent
- Automatic server selection logic to choose the best machine for a Virtual Environment

4. Handling a Failure

Suppose that at some point the server running the application fails. The controller stops receiving health information from the server. In essence, the heartbeat "flatlines." This triggers the high availability policy specified in Step 2. The policy states that in the event of a server failure, the server should be replaced and the application should be restarted.

To implement this policy, openQRM re-instantiates the Virtual Environment on a new server by following the process outlined in the previous step. It also marks the original system as being in a failed state and performs various administrator notifications, which can include logging, sending an e-mail, and other steps.

openQRM technologies:

- Policy execution
- Health monitoring
- Logging and notification

This example demonstrates how a high-level goal, such as high availability, can be implemented extremely effectively using policy-based automation. openQRM provides the fundamental technology, but it is the integration and automation of these technologies that yields a truly powerful system.

Qlusters

Controllers	Effectors	Sensors
System Inventory	Network boot support	Server discovery
Policy Definition	Remote system control	Server Profiling
Automatic server selection logic	Server provisioning (which includes multiple effectors)	Health Monitoring
Policy Execution		Performance monitoring
Logging and notification		

Table 1: Technologies deployed

This is by no means an exhaustive list of openQRM technology or functionality. It reflects only the components of automation that we have discussed in this example. High availability is only one type of policy that can be implemented through automation. Other examples include performance and schedule-based policies. For example, a policy might allocate an additional server to a Virtual Environment if the load gets too high. Of course, removing servers when load decreases or they are needed elsewhere is possible as well.

Qlusters

Agility

openQRM allows data centers to realize greater agility through centralized control and dynamic provisioning of server hardware. Centralized Image Management provides the ability to update, modify, roll-back and audit online applications and operating system images. Dynamic provisioning enables centralized management for the data center as it performs these tasks rapidly and well within the timeframes set by IT customers.

Centralized Image Management

openQRM manages both application images and operating system images. These images are maintained by openQRM in a centralized image store. This approach helps to lower operating costs and decreases operational risks in a number of ways:

- Providing tighter control over the software that is deployed in the data center — including applications, operating system versions, and patches.
- Allowing for easy testing of upgrades and safe deployments of proven combinations, thus decreasing the number of individual points of management.

The idea behind centralized image management is to create a single, perfectly installed, configured, patched and tested version of a piece of software and then take a snapshot of it. This snapshot, or image, is then stored in a central repository. Once it is in the repository, it can be referred to in the definition of a Virtual Environment. Every time this Virtual Environment is realized on physical servers, those servers will use the appropriate software image from the central repository.

openQRM divides software images into two major categories:

- Operating system
- Application

The application image actually references the entire root file system — not just the application code. In this way it captures any other files and configuration information needed to make the application function.

A Virtual Environment can mix and match the operating system and applications to be used. One Virtual Environment might specify an operating system image which corresponds to version X of Red Hat Enterprise Linux and version Y of the Oracle database. There may be many other virtual environments that also specify version X of Red Hat Enterprise Linux. All of them will share the same pristine, carefully configured image.

Others might specify version Y of the Oracle database, but a newer version of Red Hat Enterprise Linux. This group will share an application image with the former group, but will run a different operating system image.

This approach has several benefits:

- IT can guarantee that each system is running a known and validated collection of software (operating system + application). Misconfigured or “hand-tweaked” software installations account for a huge portion of support and diagnostic headaches. Misconfigured software is also a leading cause of security vulnerabilities. Creating, validating, and automatically ensuring the use of centralized software images eliminates these problems and provides a means for IT to meet compliance requirements.

- Centralized images facilitate simpler software upgrades. An administrator can quickly and easily create an image of a new version of a software component and store the image (e.g., version X+1 of Red Hat Enterprise Linux). The administrator can then easily create a new Virtual Environment definition that refers to version X+1 of Red Hat Enterprise Linux and version Y of the Oracle database. The administrator can deploy a test version of this Virtual Environment with the push of a button. Once the administrator demonstrates that Oracle version Y and Red Hat version X+1 coexist nicely in his environment, he or she can roll out the upgraded combination into production. Rolling back simply requires going back to the old Virtual Environment definition. All of these operations are easily performed within openQRM.

- Patches represent a special case of software upgrades which are typically much smaller and released more frequently than full upgrades. Centralized images benefit the patching process. Patches can be applied to existing images and then recaptured as new images. This allows rollback, as well.

Centralized image management reduces the number of individual points of management. Instead of thousands of deployed operating systems, one for each running server, the administrator now needs to manage at most three or four.

Data centers often strive to limit the number of operating systems they use — along with the supported versions of these operating systems — in order to reduce administration costs. With openQRM, an IT department can decide that it will always have three versions of the operating system available — the current version, the previous version, and a pre-release version for testing. Each of these versions is carefully installed, configured, and validated on a server and then captured into the openQRM image store. At this point, the IT department is aware that these three versions are the only ones that need to be deployed and managed.

Qlusters

All images are stored in Network Attached Storage and accessed via NFS.[4] openQRM enables the data center to use whatever NAS storage it prefers, but it must be robust, compatible and performant:

- **robust** because it is a central component to the deployment architecture
- **compatible** so that applications can rely on those file system semantics they are accustomed to
- **performant** because it must handle the operation of multiple servers within a large scale data center

Fortunately, many good solutions are available for NAS storage which meet all of the criteria listed above. The NAS storage may, in fact, be a front-end to a SAN, but completely transparent to openQRM.

Dynamic provisioning of Virtual Environments

With openQRM, IT administrators define their operations in terms of these Virtual Environments rather than specific hardware resources. openQRM's policy-based automation capabilities map the Virtual Environment dynamically to available server resources. Allocation of resources can vary automatically over time depending on workload demands, available resources, failures and other changes to the overall data center environment.

Most fundamentally, openQRM can provision and de-provision one or more servers according to the needs of a Virtual Environment. This provisioning happens dynamically, so the hardware serving a given Virtual Environment can change over time. Dynamic provisioning provides value in several ways:

Scaling out

A scale-out application is one whose performance can be increased by adding more servers. A common example of a scale-out application is a Web server. The task of serving Web pages can be effectively distributed across multiple servers. From the outside, there appears to be a single large Web server, even if the environment is actually a collection of servers acting together.[5]

openQRM makes scale-out applications efficient by allowing servers to be dynamically and automatically provisioned and de-provisioned based on pre-defined policies, including performance policies. The number of physical servers being applied to the task of Web serving can vary significantly over time depending on the load and other factors.

---

[4] The kernel image is actually supplied via tftp at boot time as part of the PXE process.

[5] This is in contrast to a scale-up application, which is monolithic and cannot be distributed across servers. With scale-up applications, the only way to provide more performance is to run the application on a bigger server; that is, to scale up.

Qlusters

This automatic provisioning and de-provisioning enables IT to avoid static over-provisioning and makes computing resources available to other applications when they would otherwise be idle. Many applications support scale-out architectures today, and more are being written or rewritten to take advantage of it.

Right-sizing Applications and Systems

Scale-out applications are inevitably over-provisioned to deal with the maximum anticipated demand. This is a huge waste of computing resources, since typical load is often a tiny fraction of that. It is not unusual for scale-out applications to run at 20% or less of capacity.

Why so much waste? The problem is that adding capacity is complex, time consuming and error prone. In statically provisioned environments, it is not practical to change configurations in response to demand. The choice it to provide too little capacity and risk not meeting the demands of the application's users, or to provide too much and tie up an organization's budget in hardware that isn't being used. Neither option is an appealing one.

openQRM allows the data center to provision Virtual Environments as needed to comply with policies. This reduces the need to over-provision for every application and thereby increases utilization of servers. Data centers realize more pain from under-provisioning through customer complaints. Static provisioning usually imposes a "pain threshold" on IT in the form of customer complaints. As static provisioned servers require more time and effort to change, IT will act only once the pain threshold has been reached. openQRM dynamic provisioning reduces the over-provisioning and the pain threshold of static deployments.

Enables Simplified and Efficient Centralized Management

openQRM not only makes scale-out efficient, it also makes it easy to administer. An administrator simply identifies the need for dynamic scale-out when defining the Virtual Environment. The administrator specifies the minimum and maximum number of servers to be deployed to the Virtual Environment and a policy for when to add and remove servers. Once the requirement is defined in the Virtual Environment, openQRM uses its policy-based automation capabilities to automate the task.

Dynamic Provisioning Provides More Efficient High Availability

The openQRM Virtual Environment model supports mechanisms for resource-efficient high availability. In traditional high availability systems, an active system and one or more standby systems are allocated and provisioned. When the active system fails, a standby takes over. The combination of the active and standby machines forms a highly-available virtual server, which appears to be a single machine.

Unfortunately, this approach is resource-inefficient; standby machines can double or triple resource requirements. openQRM's ability to perform rapid provisioning means that standby servers do not need to be pre-provisioned. They can be allocated from the server pool as needed.

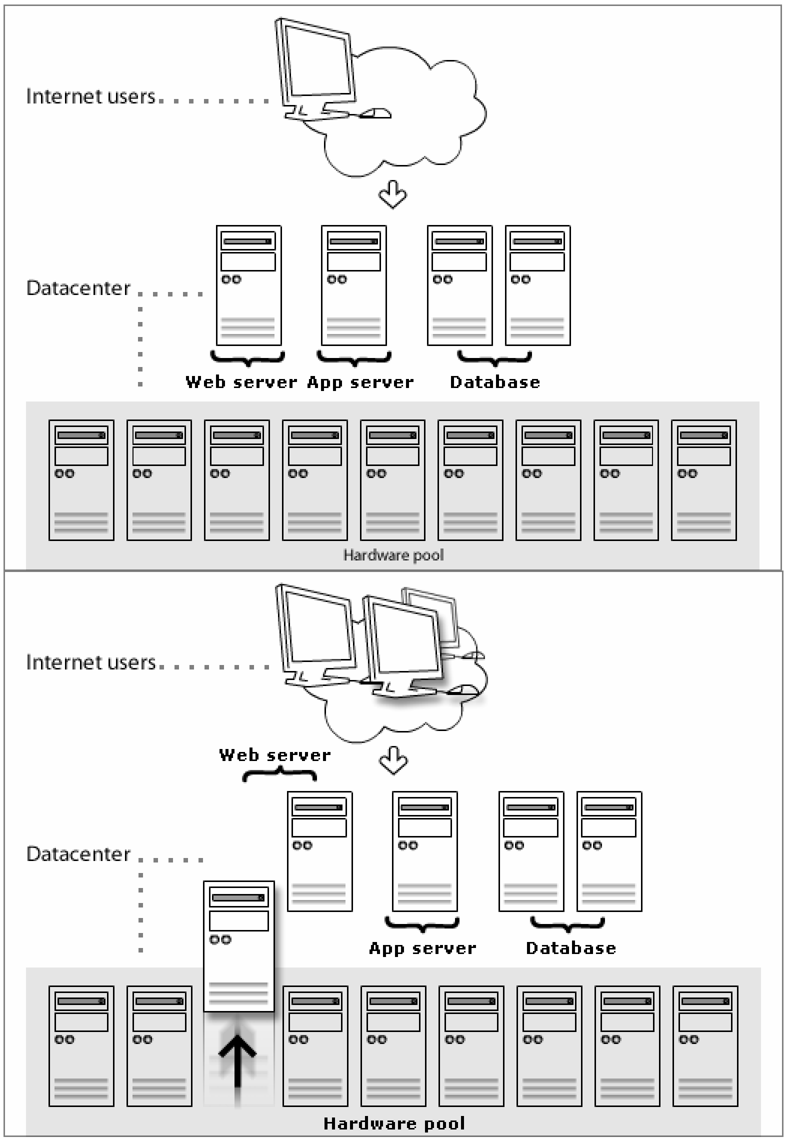

Figure 4 A: Scale-out Example. Three-tier deployment of web, application and database servers.

Figure4 B: Scale-out Example. Demand increases as more users access the internet application, triggering a policy to add an additional web server.

Qlusters

Availability

openQRM provides a range of high availability options. Each Virtual Environment can specify an appropriate type of high availability for the particular application. This section covers the various types of high availability, how they work, and how to choose among them.

Selecting a High Availability Solution Based on Recovery Objectives

openQRM offers three levels of high availability:

- Hardware High Availability
- Application High Availability

High availability is often measured with two metrics: Recovery Time Objective (RTO) and Recovery Point Objective (RPO).

- RTO is a measure of how long it takes to recover from a failure.
- RPO is a measure of how far back in time the application has to go to find a consistent set of data from which it can restart.

Clearly, the ideal situation is for an RTO and RPO to be as close to zero time as possible. If both were zero, continuous operation would exist.

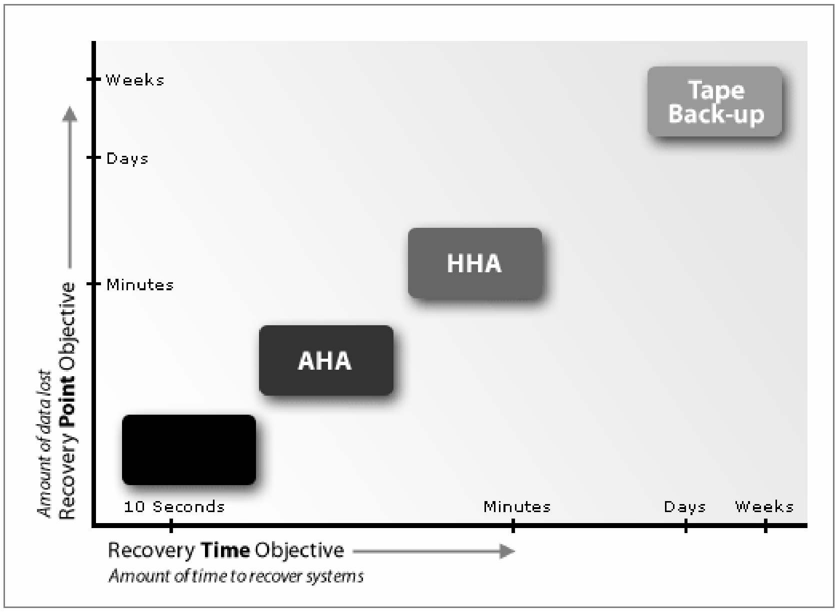

Figure 5: RPO and RTO Visualized

Recovery time consists of four major components:

- The time to determine that a failure has occurred.
- The time to prepare a system to take over on behalf of the failed system.
- The time to restart the application on the new system, including any application-specific recovery (e.g., a database processing log).
- The time it takes for other applications in the environment to recognize that the recovery has occurred.

The last two items are often the dominant factors in overall recovery time.

The recovery point is application-specific as it requires determining where an application can resume after a failure. For largely stateless applications (like Web servers serving static pages), this is simple; the recovery point can be zero time. For applications that maintain state, a rollback to a clean version of the state is required. This often corresponds to the last committed database transaction.

Traditional high availability systems typically dedicate standby computers to each application with a high availability requirement. By doing this, they eliminate the time required to prepare a new system to take over for the failed one. Unfortunately, this doubles server hardware requirements, as each high available server is shadowed by a second node whose only purpose is to wait for the primary to fail.

The ability of openQRM to rapidly provision a node from a pool of available server resources renders the traditional standby server unnecessary. openQRM allows the IT manager to indicate the size of the overall standby pool. If the pool should drop below that required size threshold, an alert is issued.

A threshold can be violated in the event of multiple failures, or if unusual high-priority demands are made on the pool of server resources. In this case, the IT manager may take a corrective action, such as dialing back the number of servers allocated to other Virtual Environments or by deploying new or repaired hardware sooner than planned.

In many high availability systems, failure is detected by the loss of a "heartbeat" between the active system and its dedicated standby (or standbys). This heartbeat is a lightweight network communication that tells the standby system that the active system is still alive. Lack of a heartbeat for a certain period of time indicates failure. This interval is known as the failure detection time.

Since openQRM does not require that standby servers be allocated in advance, an active server cannot send a heartbeat to them. Instead, it sends the heartbeat to the openQRM server, which determines when a node has failed by the absence of its heartbeat. Failure detection time is equivalent to that of traditional high availability systems with dedicated standby nodes, but resources are used much more efficiently.

Option 1: Hardware High Availability

openQRM's Hardware High Availability functionality focuses on detecting and responding to hardware failures. When openQRM provisions a system, it installs a lightweight agent along with the software defined by the Virtual Environment. This agent collects information and delivers it to the openQRM server in the form of heartbeat transmissions.

If the system fails, it ceases to send heartbeats. After a detection interval, the openQRM server declares that the system has failed. At this point, it can initiate the fail-over process, which includes:

- Identifying a new server from the pool of available servers which matches the requirements specified by the Virtual Environment
- Starting the operating system and application on the newly provisioned server

The bulk of the fail-over time for significant applications usually involves application restart and recovery. The provisioning step basically consists of a server reboot.

Hardware High Availability only detects system failure — it has no notion of application state. As long as the hardware and the operating system are healthy, no failure is detected. Hardware High Availability does not notice failures at the application level.

Option 2: Application High Availability

A high availability system must also monitor the health of the application, and determine whether the application is actually functioning. For example, a database consists of many processes; even if all of its processes are still alive, it is not guaranteed that the database will respond properly to clients.

An application-specific agent is designed to monitor an application and determines its well-being. In the database example, the agent may act as a database client and issue queries against system tables. If the application does not respond as expected, or at all, the agent notifies the high availability system to take corrective action.

openQRM provides a toolkit for creating application-specific agents. When the agent's application-specific logic detects an application failure, it uses the openQRM library to send an event to the openQRM server to report it.

openQRM responds to application failures in much the same fashion as it would in the event of a hardware or operating system failure. It finds a new server, provisions it to run the required operating system and application, and starts running it. openQRM takes the additional step of taking down the original server node and returning it to the pool.

Application High Availability provides a superset of the detection capabilities provided by Hardware High Availability: application level monitoring as well as system heat-beat. A loss of heartbeat or an explicit event from an application agent can identify a failure and trigger a fail-over.

Hardware High Availability and Application High Availability are similar in their operational model:

- Both heartbeat to the QRM server
- Both rely on a pool of standby nodes
- Both require that the application restart and recover from scratch
- Neither preserves application state

Qlusters

- Neither affects the performance of the running application

The approach taken by Application High Availability and Hardware High Availability is compatible with a broad spectrum of applications that require high availability.

This approach provides a good tradeoff between an acceptable RPO/RTO and good resource utilization. However, for some very mission-critical applications, which cannot experience downtime, a different balance is required. RPO and RTO must be driven close to zero, and other factors must be considered to achieve this goal.

Summary

openQRM uses a combination of technologies that provide automation, agility and high availability capabilities. This combination of technologies sustains the viability of data centers built around commodity servers and Linux operating systems. A data center without openQRM faces higher costs due to the low resource utilization and high management overhead associated with a large deployment of commodity servers. openQRM allows data centers to deliver acceptable, if not superior, availability and reduces the complexity of management.

The “lynchpin” of openQRM is the Virtual Environment. This allows IT administrators to define an application environment independent of the physical hardware used to process the applications. Rapid server provisioning makes this practical.

openQRM’s policy-based engine is the core of its automation capabilities. The control logic automatically takes actions based on business-based policies defined by IT.

openQRM also includes a range of technologies that provide varying levels of high availability. The business demands of the application dictate the appropriate level of high availability to apply: this is specified as a policy in the Virtual Environment definition. The openQRM approach to hardware and application high availability offers simplicity and efficient server resource utilization in a robust fail-over solution.

This paper provides an overview of the technologies embodied in the openQRM management platform. Qlusters works with large corporate data centers to advance these products and technologies so that IT can execute business processes at the lowest possible cost with the best possible availability and operational agility.