HIGH-AVAILABILITY

The Future of High-Availability

Introduction

High-Availability solutions have evolved significantly over the past couple of years; many of today's solutions combine mirroring or replication technologies with intelligent management software to enable geographic separation of cluster nodes and storage to provide disaster recovery within the High-Availability framework. But what will tomorrow's High-Availability solutions look like?

Solution Architectures

There are probably three realistic architectures for high-availability which will exist once consolidation in the market place is complete;

Distributed Computing

Data is distributed among many machines. Changes are normally made on one machine and *roll out* over time to all machines. This *model* is in use today, in a simplistic form, by software vendors who use 'mirrors' for downloads like tucows or in distributed computing projects like the 'search for extraterrestrial life' (SETI).

This model allows fairly low powered machines to provide a high computing capability and more importantly a system resilient to large scale failure. However, conflicts can occur if the same 'object' is changed in two different places at the same time. This is normally overcome by running a master, for writes, and multiple slaves, for read operations. Problems can occur if the master is unavailable for any length of time.

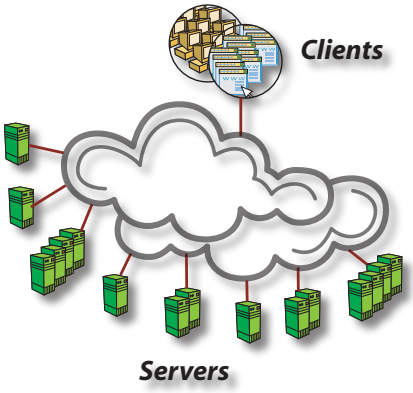

Figure 1.
Distributed Servers

Parallel Computing

Transactions are completed simultaneously on different servers at the same time. However, in contrast to the distributed computing model, the servers perform a co-ordinated locking arrangement, so that no conflict can occur. Today only a handful of vendors, like Oracle's with their RAC solution, have begun to provide some of the components needed for true parallel operations.

The theoretical benefits of parallel compute include improved scalability and performance. Depending on the specific solution, the servers don't need to be co-located or share the same disks but if performance is also important then the links between the machines must be very low latency, normally only available with co-location and specialised hardware.

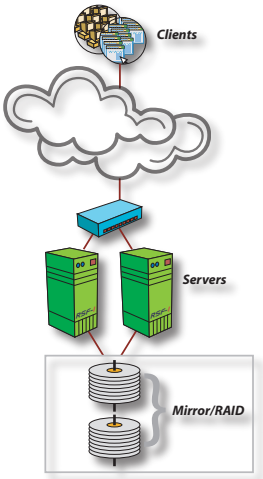

Figure 2.
Parallel Servers

There are a number of ways to achieve parallel computing operations; at the application level, at the operating system level and through middle-ware.

On-Demand Computing

Only one instance of an application processes all transactions but other instances are available to take over if a fault should occur.

This can be achieved using co-located hot-standby servers or disparate mirrored data. The mirrored solution is only practicable where performance is not important, as to provide a reliable system a form of dual phase commit is required.

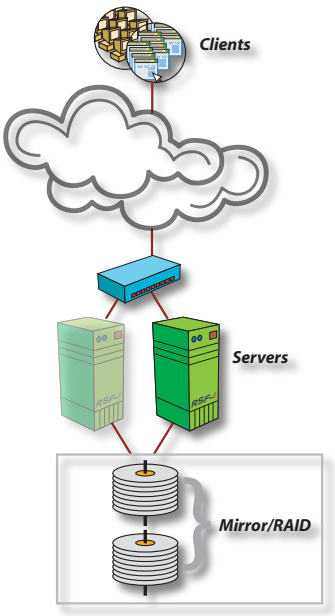

Figure 3.
On Demand Servers

So What is Grid Computing?

The constant marketing push for new buzz words and news items has perpetuated a view that true grid computing for the enterprise is available, or will shortly be available. Distributed applications, like SETI, are already running in a form of grid like architecture but the enterprise needs databases and other applications where synchronised data must be provided to be useful. Even the director of SETI rejects the grid label, preferring the term “public resource computing”.

The term “grid computing” was originally coined by Carl Kesselman and Dr Ian Foster of America’s Argonne National Laboratory in 1998, he drew an analogy between the supply of computing power and the supply of electricity, which is delivered when and where you need without needing to worry about where it came from.

In 2000 Dr Foster refined his definition with Steve Tuecke to describe the essence of grid computing. They came up with three essential components;

1) *coordinates resources that are not subject to centralized control …*
(A Grid integrates and coordinates resources and users that live within different control domains—for example, the user’s desktop vs. central computing; different administrative units of the same company; or different companies; and addresses the issues of security, policy, payment, membership, and so forth that arise in these settings. Otherwise, we are dealing with a local management system.)

2) *… using standard, open, general-purpose protocols and interfaces …*
(A Grid is built from multi-purpose protocols and interfaces that address such fundamental issues as authentication, authorization, resource discovery, and resource access. It is important that these protocols and interfaces be standard and open. Otherwise, we are dealing with an application specific system.)

3) *… to deliver non-trivial qualities of service.*
(A Grid allows its constituent resources to be used in a coordinated fashion to deliver various qualities of service, relating for example to response time, throughput, availability, and security, and/or co-allocation of multiple resource types to meet complex user demands, so that the utility of the combined system is significantly greater than that of the sum of its parts.)

Drawing from Dr Foster’s work, when or if grid computing is delivered it will be a collection of self healing, ultimately scalable and self learning systems that can accept new servers or applications without downtime or manual re-configuration. Servers will re-configure on the fly to provide extra resources to applications when and where they are needed. However, it is unrealistic to expect for this to deliver enterprise computing facilities for all applications or users in the coming decade. Security and predictable performance as well as cultural changes are all challenges that need to met before the enterprise will embrace true grid computing.

The Needs of Corporate Enterprise

Distributed computing provides excellent availability and potential performance but, for reasons of security, predictability and the need to have coherent synchronised data, distributed computing is not a model that is normally be accepted by enterprises outside the field of *number crunching* based research processing.

Enterprises need a balance between cost and performance, while maintaining control, availability and security.

There is very little to choose between the availability provided by configurations based on the parallel model described above and the reliability provided by configurations base on the on-demand model. However, there are other differences that present challenges to successful deployment that we'll cover in more detail here;

Building Parallel Server Configurations

Probably the best known companies selling '*would be'* grid solutions are Sun and Oracle.

Sun certainly provide excellent hardware, which includes entry level Opteron servers up to their mainframe class high-end SPARC servers. Solaris is probably the most capable, widely adopted enterprise class operating system available. However, Sun produce components that can in future be used to build a grid, not the whole solution.

Oracle's 10*g* and former incantations, like Real Application Clusters (RAC), again are excellent components. However, even when combined with Sun's components there are still substantial pieces required to be built before a high-availability solution can be offered, let alone a grid computing model.

To build a highly available system using parallel technology we start with the hardware. An enterprise server from Sun running Solaris, with hardware based RAID and mirroring between locations provides a stable platform on which to host applications.

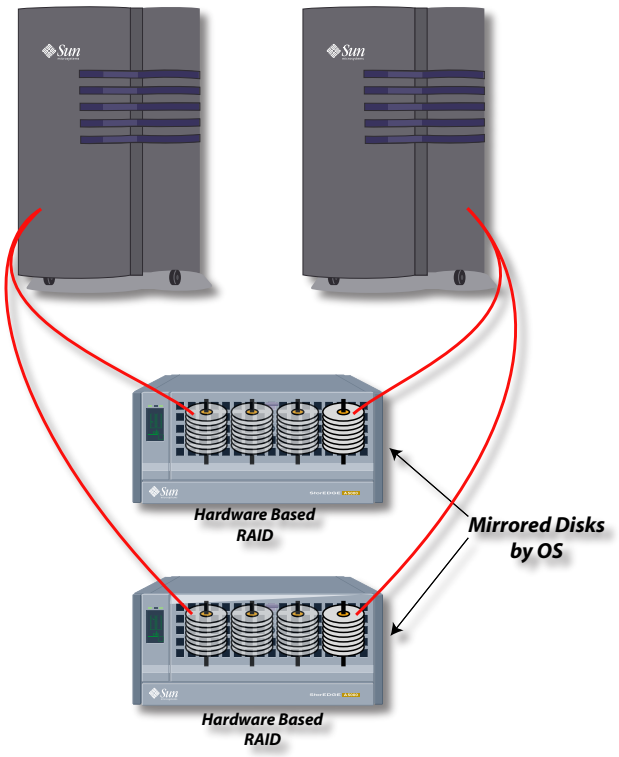

Figure 4.
Stable Hardware Platform

The database can now be deployed on the operating system and then the application can be installed. An example is shown in Figure 5., which shows a typical application; AR System from Remedy. Oracle RAC requires a cluster file system and specialised hardware to facilitate low latency links, which is used for inter-node state and lock information. In addition clustering must be added to the solution to be enable the use of Oracle RAC.

Most applications are written to be database independent, rather than locked in to Oracle, so operate outside of Oracle's application server components. This means that they need monitoring and clustering as well.

The Oracle database, when deployed as RAC, runs on multiple nodes. Each node will have *locks* over certain parts of the database and will release the locks to another node when required, and free. Clustering is employed to monitor the status of each node and inform Oracle RAC of a node failure. Connections to a node that fails will have to be restarted unless the application is specifically written to use Oracle's Transparent Application Failover (TAF).

For an application to be run using the power of RAC, the application MUST be written specifically to take advantage of the features and functions provided by RAC, you can not just deploy an application in RAC without modification. The extent of modification required is specific to each application but is often extensive and as a result the number of applications supported in a true RAC environment is relatively small.

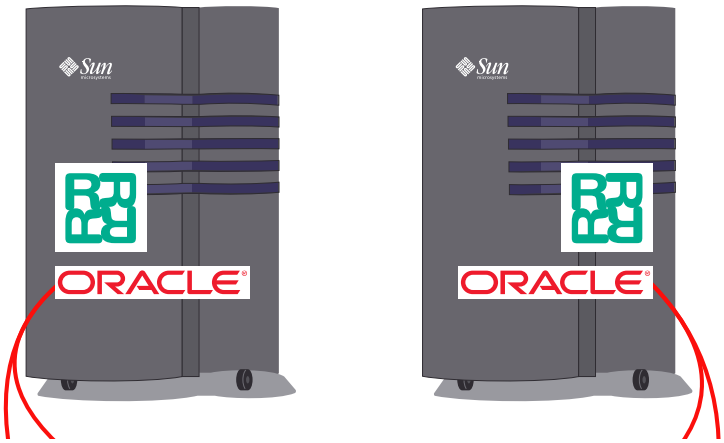

Figure 5.
Applications and Database installed

Applications Vendors Catching Up?

Application vendors, like Remedy, have been slow to adopt technologies like RAC and this unlikely to change, as the level of investment required would normally result in the adoption of Oracle as the only database supported. The slow uptake is not just because RAC is a form of vendor lock-in but also because Remedy's customers already have substantial install bases of other databases. Particularly in specific industry sectors, for instance Sybase is very popular in financial institutions, Progress is used extensively in manufacturing and MySQL is very popular with web based technologies.

Applications vendors who want to provide users with parallel access to their applications have a number of technical challenges that are often *glossed* over.

Vendor Built Highly Available Application Access

A remote user of a corporate network does not want to know which of several available servers should be used. The user just wants to run his application client, which could be web based but normally uses proprietary protocols, without even having to know about the server. There are a number of architectural options available to vendors who want to provide automated, transparent to the user, client-server connections;

Client Side Intelligence - where the client software is aware of all servers, polls all the servers for view on responsiveness, connects in to the selected server and maintains a view of availability. Switching to another server in the instance of a failure.

Mediated Intelligence - Load Balancers - often using industry standard components, like Alteon, Foundry, F5 or Cisco traffic load balancers. Because the applications often use proprietary protocols the load balancers must often use a basic *'ping'* test and the distribute traffic on a round-robin basis. Further, the client must maintain state and authentication information if a fail-over is to be transparent.

Server Side Intelligence - where the servers co-operate and have a traffic handling component that the client connects to. The server then feeds-back information about which server to use to the client. The client then connects into the appropriate server.

All of these approaches require the *application* client or server to be specifically architected to include the high-availability capability within the core of the application. Application vendors are often reluctant to implement these strategies partly because the expertise required to implement these architectural models successfully is rather specialised. Probably the most compelling reason for not developing their own solution is that the costs of developing and deploying such technologies is greater than that of deploying the *On-Demand* model from an existing middle-ware clustering vendor.

Vertical or Horizontal Expansion

Vendors often like to portray their software applications as extending vertically within a frame work and while some notable exceptions exist, there are very few successful applications that truly extend vertically. The most notable exception would be the application environment provided by Oracle, which has had some take up but, as discussed above, is limited by the lack of universal customer and application vendor acceptance. It should also be noted that a single vendor solution approach conflicts with Dr Foster's second rule of the Grid, *using standard, open, general-purpose protocols and interfaces*.

To build highly available systems it is essential to ensure that sufficient horizontal expansion exists to remove all single points of failure. Vertical expansion can in some cases **reduce** the resilience of a solution unless properly implemented. As for example with Oracle, applications must be coded to use TAF and to support the RAC structure if they are to be setup in a highly available way, with transparent failover.

Moving To A Common Protocol?

Referring back to Dr Foster's refined definition, *using standard, open, general-purpose protocols and interfaces,* one can argue that the web (http/html and XML) does provide a common protocol and it does, using XML, allow for extensive customisation to, in theory at least, allow for data exchange and management of processes. So http and XML could be used as the basis of a common protocol. Most application vendors now offer some sort of web based access to their application and many offer integration with their applications based on http & XML.

While providing users with access to applications across the web is not grid computing it does simplify some of the issues related to providing users with highly available systems. A web based application can be designed to be run from more than one server and indeed to be able to maintain state in the event of a server failure. This is normally achieved by using cookies (stored on the client machine) and maintaining a record of user (identified with the cookie) actions in a database. An example of this type of application is on-line shopping, where the items selected are written into a database and associated with a specific user, or session if the user has yet to 'register'. Should the web server that was in use fail then the user selected items will still be visible in the users basket if he accesses using a different server.

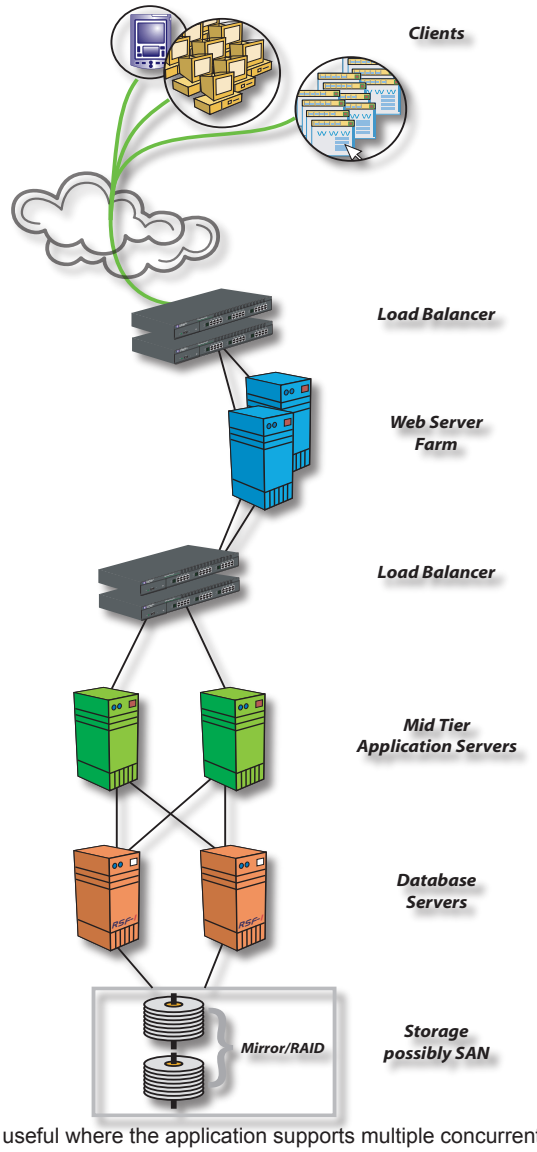

Figure 6.
Typical End-to-End Solution

Figure 6. shows a typical example of current implementations that do provide high-availability. Users access the applications through the web servers. The use of the load balancing devices between the web server farm and the 'mid-tier' application servers is only useful where the application supports multiple concurrent instances. The load balancers at this point are unlikely to be able to correctly track user connections as they are 'once removed' from user traffic. However, sometimes cookies or other user connection specific data can be tracked by the load balancers to make individual connections 'sticky' at the web server to mid-tier level.

What Can Be Achieved Today

The best way to get true high-availablity today is to use an architecture like that shown in Figure 6. Where possible the applications should be extended horizontally - meaning that multiple live instances will be run but fail over is also a viable option. The choice of live or fail over depends on the application's capabilities/dependancies.

The best practice methodology of identifying and eliminating single points of failure should be applied. Which will normally include; diverse and resilient networking and power supply paths, redundant load balancers and servers.

What About Sun Grid?

Sun Grid is an interesting service offering but it is not high-availability. While the service offers a very reliable computation capability it is useful for low investment compute-on-demand rather than corporate service (application) provision. The offering is based around a cost per CPU hour and data stored per month. Quoting Sun; *"a $1 per CPU per hour pay per use offering for batch workloads such as Monte Carlo simulations, protein modelling, geologic exploration, and mechanical CAD simulations, among others."*

While interesting and novel, Sun Grid does not solve the fundamental issue: the efficient allocation of networked computing resources. Companies do not think of their computing needs in terms of, say, 12 processor-hours; instead, they have specific tasks of varying importance and urgency, and want to get those tasks done economically, using whatever resources are available.
Dealing with practical (commercial) considerations, Sun will be unable to provide an infinite supply of computing

power and while the service remains in it's infancy the duration of a job will be small as the load can be widely spread on very powerful computers. However, some jobs are less important to be provided quickly than others and unless some way of prioritising jobs is applied then the value of such a novel service will be reduced.

Dr Bernardo Huberman, a researcher at Hewlett-Packard (HP) and his team, have been working on software project called Tycoon which will turn computing grids into "stock markets or clearing houses", where jobs can be completed faster by bidding for CPU bandwidth.

The HP team has tested Tycoon on a distributed cluster of 22 Linux servers located in HP's headquarters in Palo Alto, California, and its offices in Bristol, England. Tycoon performed well in these tests, and some animated films were rendered using its system. HP has now given Tycoon to CERN, the world's largest particle physics laboratory and a hot bed of grid-computing research, for more testing.

Conclusions

Remember OSI seven layer model? The OSI seven layer model was designed as the universal model for networking and applications, a true standard. TCP/IP, which doesn't fit the OSI model, has continued to occupy the networking space and shows no sign of being replaced despite substantial activity and backing from giants like IBM in the 80's and 90's. The application levels are much further removed from universal standards than the networking world.

Grid computing as defined by Dr Foster is still a long way off. The pieces are coming together now and what is today sold as grid computing does have some limited abilities to address some of the real needs Dr Foster has identified.

High-availability solutions as opposed to massive compute solutions, like Sun Grid, must still be built in the traditional way using cluster solutions. This situation is unlikely to change fast as the vested interests of applications vendors and hardware vendors encourages them to maintain their proprietary approach to solution building.

Application vendors are usually keen to avoid lock in to other vendor applications and hardware. For this reason the independent solutions to enable high availability are the future of application integrations. The advantages of working closely with a clustering vendor are well known to the application vendors.

Clustering will continue to be the dominant method of providing highly available services for the enterprise. The use of distributed model computing will grow but the panacea of 'Grid' computing is not going to be available for a number of years to come - if we can get over the marketing spin.

HIGH-AVAILABILITY