What's the difference, really?

There are actually quite a few differences, even if you ignore that clustering is generally used to refer to the capability of a software product to provide load-balancing services and load-balancing is often used to refer to a hardware-based (or at least third-party software) solution.

Clustering is most often used in conjunction with application servers such as BEA WebLogic, IBM WebSphere, and Oracle AS (10g). So are load-balancing features found within Application Delivery Controllers (ADC) like BIG-IP.

In the world of hardware load balancers the term "pool" or "farm" is used to describe a grouping of servers across which application requests will be distributed. In the world of software load balancing the term used is "cluster".

I will try to forget the use of the term factotum for this concept as it still gives me nightmares.

**Scalability**

Clustering typically makes one instance of an application server into a master controller through which all requests are processed and distributed to a number of instances using industry standard algorithms like round robin, weighted round robin, and least connections. Clustering, like load balancing, enables horizontal scalability, that is the ability to add more instances of an application server nearly transparently to increase the capacity or response time performance of an application. Clustering features usually include the ability to ensure an instance is available through the use of ICMP ping checks and, in some cases, TCP or HTTP connection checks.

ADCs typically support these same industry standard algorithms, but add more complex calculations and parameters that can include per-server CPU and memory resource utilization and fastest response times. ADCs also support health monitoring capabilities, but they generally go beyond the rudimentary capabilities of those found in application server clustering solutions. This includes the ability to verify content or perform passive monitoring which removes the relatively low impact of health checking on application server instances.

**Server Affinity**

Clustering uses server affinity to ensure that applications requiring the user interact with the same server during a session get to the right server. This is most often used in applications executing a process, for example order entry, in which the session is used between requests (pages) to store information that will be used to conclude a transaction, for example a shopping cart.

ADCs use persistence to provide the same functionality. While clustering solutions are generally limited in the variables that can be used, ADCs can use traditional application variables as well as custom information from within the application data or network-based information.

**High Availability (Failover)**

Clustering solutions claim to provide HA/Failover capabilities, when this failover is related to application process level failover, not high availability of the clustering controller itself. This is an important distinction as in the event the clustering controller instance fails, the entire system falls apart. While cluster-based load-balancing provides high availability for members of the cluster, the controller instance becomes a single point of failure in the data path.

ADCs are built for redundancy and include sophisticated features that not only ensure applications are still available if one ADC fails, but also replicates session state between two ADCs such that if the primary fails the application sessions are not lost. This replication capability is also available in most clustering application server solutions.

**Transparency**
Many clustering solutions require a node-agent be deployed on each instance of an application server being clustered by the controller. This agent is often already deployed, so it’s often not a burden in terms of deployment and management, but it is another process running on each server that is consuming resources such as memory and CPU and which adds another point of failure into the data path.

ADCs require no server-side components, they are completely transparent.

**Making A Choice**

So which should you chose? That depends highly on the reasons you are considering implementing either clustering or deploying an ADC and whether or not you will have to make an additional purchase to enable clustering capabilities for your particular application server. There’s also a broader question of whether you will need to provide this support for more than one application server brand. Clustering is proprietary to the application server while ADCs can provide these services for any application or web server.

**Clustering**

**The pros:**
- Generally available as part of an enterprise package for an application server
- Solution doesn’t require a lot of networking skills
- Generally less expensive than a redundant ADC deployment

**The cons:**
- High availability is not assured using clustering solutions
- Best practices dictate the cluster controller be deployed on separate hardware
- Requires node agents on managed application server instances
- Clustering is "proprietary" in that you can only cluster homogeneous servers.

**ADCs**

**The pros:**
- Can provide high availability and load balancing across heterogeneous environments
- Offers additional value such as **optimization, security, and acceleration** for applications
- Transparent - doesn’t require changes to applications or the servers on which they are deployed

**The cons:**
- Adds another piece of infrastructure to the architecture
- Generally more expensive than clustering solutions
- May require a new set of skills to deploy and manage
Want more insight into performance, configuration, and use cases? Check out this testing-based article on ADCs, and this testing-based review of application server clustering.

Imbibing: Water

Technorati tags: F5, application delivery, load balancing, clustering, MacVittie