High-availability cluster

High-availability clusters (also known as **HA clusters** or **fail-over clusters**) are groups of computers that support server applications that can be reliably utilized with a minimum amount of down-time. They operate by using high availability software to harness redundant computers in groups or clusters that provide continued service when system components fail. Without clustering, if a server running a particular application crashes, the application will be unavailable until the crashed server is fixed. HA clustering remedies this situation by detecting hardware/software faults, and immediately restarting the application on another system without requiring administrative intervention, a process known as failover. As part of this process, clustering software may configure the node before starting the application on it. For example, appropriate file systems may need to be imported and mounted, network hardware may have to be configured, and some supporting applications may need to be running as well.[1]

HA clusters are often used for critical databases, file sharing on a network, business applications, and customer services such as electronic commerce websites.

HA cluster implementations attempt to build redundancy into a cluster to eliminate single points of failure, including multiple network connections and data storage which is redundantly connected via storage area networks.

HA clusters usually use a heartbeat private network connection which is used to monitor the health and status of each node in the cluster. One subtle but serious condition all clustering software must be able to handle is split-brain, which occurs when all of the private links go down simultaneously, but the cluster nodes are still running. If that happens, each node in the cluster may mistakenly decide that every other node has gone down and attempt to start services that other nodes are still running. Having duplicate instances of services may cause data corruption on the shared storage.

HA clusters often also use quorum witness storage (local or cloud) to avoid this scenario. A witness device cannot be shared between two halves of a split cluster, so in the event that all cluster members cannot communicate with each other (e.g., failed heartbeat), if a member cannot access the witness, it cannot become active.

Contents

Application design requirements
Node configurations
Node reliability
Failover strategies
Implementations
See also
References
Further reading

Application design requirements

Not every application can run in a high-availability cluster environment, and the necessary design decisions need to be made early in the software design phase. In order to run in a high-availability cluster environment, an application must satisfy at least the following technical requirements, the last two of which are critical to its reliable function in a cluster, and are the most difficult to satisfy fully:

- There must be a relatively easy way to start, stop, force-stop, and check the status of the application. In practical terms, this means the application must have a command line interface or scripts to control the application, including support for multiple instances of the application.
- The application must be able to use shared storage (NAS/SAN).
- Most importantly, the application must store as much of its state on non-volatile shared storage as possible. Equally important is the ability to restart on another node at the last state before failure using the saved state from the shared storage.

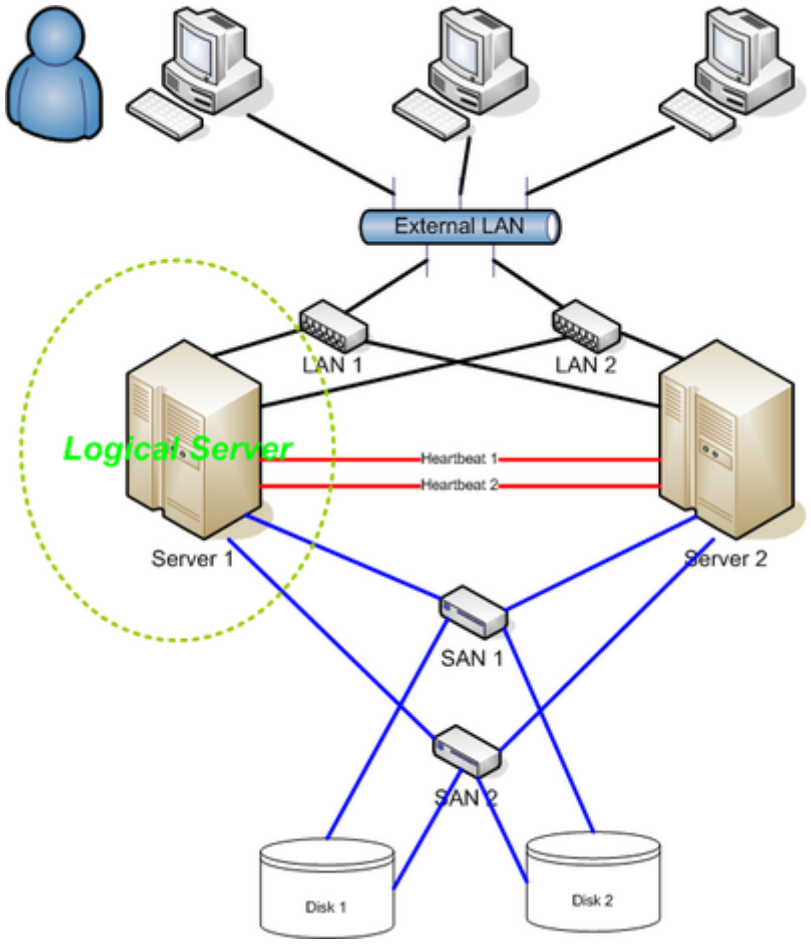
- The application must not corrupt data if it crashes, or restarts from the saved state.
- A number of these constraints can be minimized through the use of virtual server environments, wherein the hypervisor itself is cluster-aware and provides seamless migration of virtual machines (including running memory state) between physical hosts -- see Microsoft Server 2012 and 2016 Failover Clusters (https://docs.microsoft.com/en-us/windows-server/failover-clustering/failover-clustering-overview).
    - A key difference between this approach and running cluster-aware applications is that the latter can deal with server application crashes and support live "rolling" software upgrades while maintaining client access to the service (e.g. database), by having one instance provide service while another is being upgraded or repaired. This requires the cluster instances to communicate, flush caches and coordinate file access during hand-off.

Node configurations

The most common size for an HA cluster is a two-node cluster, since that is the minimum required to provide redundancy, but many clusters consist of many more, sometimes dozens of nodes.

The attached diagram is a good overview of a classic HA cluster, with the caveat that it does not make any mention of quorum/witness functionality (see above).

Such configurations can sometimes be categorized into one of the following models:



2 node High Availability Cluster network diagram

- Active/active — Traffic intended for the failed node is either passed onto an existing node or load balanced across the remaining nodes. This is usually only possible when the nodes use a homogeneous software configuration.
- Active/passive — Provides a fully redundant instance of each node, which is only brought online when its associated primary node fails.[2] This configuration typically requires the most extra hardware.
- N+1 — Provides a single extra node that is brought online to take over the role of the node that has failed. In the case of heterogeneous software configuration on each primary node, the extra node must be universally capable of assuming any of the roles of the primary nodes it is responsible for. This normally refers to clusters that have multiple services running simultaneously; in the single service case, this degenerates to active/passive.
- N+M — In cases where a single cluster is managing many services, having only one dedicated failover node might not offer sufficient redundancy. In such cases, more than one (M) standby servers are included and available. The number of standby servers is a tradeoff between cost and reliability requirements.
- N-to-1 — Allows the failover standby node to become the active one temporarily, until the original node can be restored or brought back online, at which point the services or instances must be failed-back to it in order to restore high availability.
- N-to-N — A combination of active/active and N+M clusters, N to N clusters redistribute the services, instances or connections from the failed node among the remaining active nodes, thus eliminating (as with active/active) the need for a 'standby' node, but introducing a need for extra capacity on all active nodes.

The terms *logical host* or *cluster logical host* is used to describe the network address that is used to access services provided by the cluster. This logical host identity is not tied to a single cluster node. It is actually a network address/hostname that is linked with the service(s) provided by the cluster. If a cluster node with a running database goes down, the database will be restarted on another cluster node.

Node reliability

HA clusters usually use all available techniques to make the individual systems and shared infrastructure as reliable as possible. These include:

- Disk mirroring (or Redundant Arrays of Independent Disks --RAID) so that failure of internal disks does not result in system crashes. The Distributed Replicated Block Device is one example.

- Redundant network connections so that single cable, switch, or network interface failures do not result in network outages.
- Redundant storage area network (SAN) connections so that single cable, switch, or interface failures do not lead to loss of connectivity to the storage (this would violate shared nothing architecture).
- Redundant electrical power inputs on different circuits, usually both or all protected by uninterruptible power supply units, and redundant power supply units, so that single power feed, cable, UPS, or power supply failures do not lead to loss of power to the system.

These features help minimize the chances that the clustering failover between systems will be required. In such a failover, the service provided is unavailable for at least a little while, so measures to avoid failover are preferred.

Failover strategies

Systems that handle failures in distributed computing have different strategies to cure a failure. For instance, the Apache Cassandra API Hector defines three ways to configure a failover:

- **Fail Fast**, scripted as "FAIL_FAST", means that the attempt to cure the failure fails if the first node cannot be reached.
- **On Fail, Try One - Next Available**, scripted as "ON_FAIL_TRY_ONE_NEXT_AVAILABLE", means that the system tries one host, the most accessible or available, before giving up.
- **On Fail, Try All**, scripted as "ON_FAIL_TRY_ALL_AVAILABLE", means that the system tries all existing, available nodes before giving up.

Implementations

There are several free and commercial solutions available, such as:

- IBM PowerHA SystemMirror
- Linux-HA
- HP Serviceguard, available since 1990[3]
- Oracle Solaris Cluster
- Red Hat Cluster
- Veritas Cluster Server
- Evidian SafeKit (https://www.evidian.com/products/high-availability-software-for-application-clustering/)
- Microsoft Failover Clusters (https://docs.microsoft.com/en-us/windows-server/failover-clustering/failover-clustering-overview) see also Microsoft Scale-Out File Services, which may be combined in Hyper-Converged computing.
- StarWind Virtual SAN (https://www.starwindsoftware.com/starwind-virtual-san) which virtualizes the SAN itself, eliminating the need for external SAN hardware.
- HP StoreVirtual VSA (https://www.hpe.com/h20195/v2/GetPDF.aspx%2Fc04111621.pdf) virtual SAN software (formerly LeftHand)
- SANless clustering capabilities for application HA both on-premise, and in the cloud - SIOS Technology (https://us.sios.com/solutions/high-availability-clusters/)

See also

- Service Availability Forum
- SAFplus
- OpenSAF
- Urgent computing
- Pacemaker (software)
- IBM Parallel Sysplex
- Multi-language blog on high-availability and disaster recovery (http://www.sios-apac.com/)

References

1. van Vugt, Sander (2014), *Pro Linux High Availability Clustering*, p.3, Apress, ISBN 978-1484200803
2. Bornschlegl, Susanne (2012). *Railway Computer 3.0: An Innovative Board Design Could Revolutionize The Market* (https://www.menmicro.com/downloads/search/dl/sk/%22White%20Paper%3A%20Railway%20Computer%203.0%3A%20An%20Innovative%20Board%20Design%20Could%20Revolutionize%20The%20Market%22/dx/1/) (pdf). MEN Mikro Elektronik. Retrieved 2015-09-21.
3. HP Serviceguard#cite note-sghistory-1

Further reading

- Greg Pfister: *In Search of Clusters*, Prentice Hall, ISBN 0-13-899709-8
- Evan Marcus, Hal Stern: *Blueprints for High Availability: Designing Resilient Distributed Systems*, John Wiley & Sons, ISBN 0-471-35601-8
- Chee-Wei Ang, Chen-Khong Tham: *Analysis and optimization of service availability in a HA cluster with load-dependent machine availability*, IEEE Transactions on Parallel and Distributed Systems, Volume 18, Issue 9 (September 2007), Pages 1307-1319, ISSN 1045-9219 (https://www.worldcat.org/search?fq=x0:jrnl&q=n2:1045-9219) [1] (http://portal.acm.org/citation.cfm?id=1313074)