International Journal of Advanced Scientific Research and Engineering Trends

ISSN: 2456-0774

Volume 10 Issue 1s (2026) | Pages: 117-121

ACS: Cost Effective System Design for Critical Application Availability

Nisha Auti[1], Sakshi Patil[2], Snehal Patil[3], Apurva Jha[4], Harsh Jhunjhunwala[5]

[1,2,3,4,5]Bharati Vidyapeeth (DU) College of Engineering, Pune 411043, Maharashtra, India,

Peer Review Information

Type: *Article*
Received: *02 April 2026*
Revised: *28 April 2026*
Accepted: *05 June 2026*
Published: *23 June 2026*

Abstract

Parallel and distributed computing technology is a need of the future. The commercially available systems are very expensive and complex and hence require regular maintenance. The aim of the proposed system is to focus on critical applications in the system so high availability can be provided instead of eliminating all potential SPOF (Single Point of Failure) in the system, as the cost is associated with the elimination of each potential SPOF in the system and this makes system unnecessarily complex and costly. Nowadays many universities are offering cluster computing courses at UG and PG level. They require a viable cluster system as an educational tool for teaching the course. Commercially available clustering systems are very expensive with much more functionality. Universities in developing countries cannot invest millions of dollars to buy such systems and cannot spend on its maintenance with its limited budget. This paper presents the cost-effective system design for improving the availability of applications by introducing an ACS (Application Cluster System) as an educational tool.

Keywords: Application Cluster System; High Availability; Heartbeat Communication; Replicated State Machine; Single Point of Failure

How to Cite This Article

Auti, N., Patil, S., Patil, S., Jha, A., & Jhunjhunwala, H. (2026). ACS: cost effective system design for critical application availability. *International Journal of Advanced Scientific Research and Engineering Trends*, 10(1s), 117–121.

Introduction

It is the age of cutting technology and evolution is an everyday norm. The desire to increase the effective computing power and reliability of a system has given rise to many concepts like clusters. As the economy of a country, nowadays, is dependent on software, high availability is a necessity, downfall of which may cost them huge amounts of money. Thus, Application Cluster System gives the design to these problems.[1][7]

The basic approach to clusters is to connect various available computer nodes and incorporate them into a single system to provide high availability of services for clients at the lowest possible cost. To ensure high availability, the key strategy is to incorporate redundant components, with system clustering facilitating the transfer of applications from malfunctioning components to their backup counterparts. [2]

The primary focus of the proposed system design is to create an ACS that conserves significant capital by utilizing a straightforward high availability cluster application, eliminating the need for costly hardware like dual ported RAID (Redundant Array of Independent Disks) and high-speed optical fiber networks. While this clustering solution may not be as sophisticated as some commercial clusters, it will adequately fulfill our requirements. Generally, this ACS system will not going to be in real competition with commercial HA (High Availability) solution very soon but will give cost effective solution for archiving high availability.[3][9]

ACS Basic Design

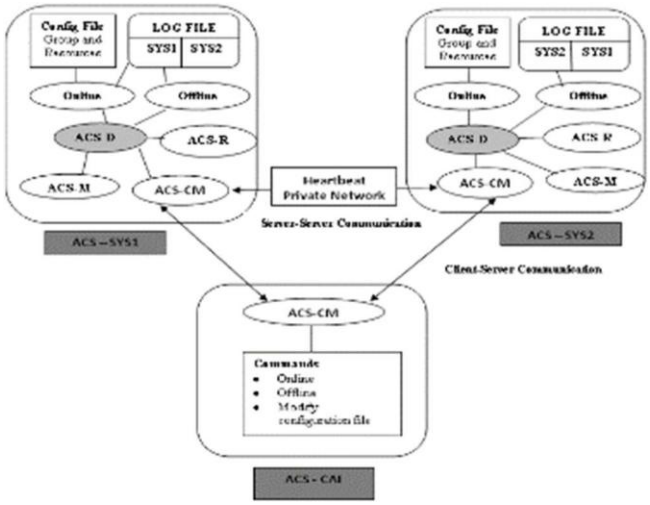

Fig. 1. *ACS configuration framework*

ACS Configuration Framework

The configuration framework of ACS system is shown in Fig. 1. The proposed ACS system is composed of a set of two computer systems connected with a dedicated communication infrastructure. ACS hardware consists of two PCs say primary (main) node and secondary (standby) node. Each cluster has a unique ID say ACS-sys1 and ACS-sys2.

For reducing the cost of cluster, freely available open-source Linux operating system is to be installed on the system for managing node resources. These two server systems are connected by cross-over cable for inter-node communication. This system allows cluster members to share status updates with one another. The private network connecting ACS nodes utilizes fast Ethernet. One of the key benefits of fast Ethernet is its affordability, ease of installation, and low maintenance requirements.[4]

To ensure high availability, critical applications must be managed under cluster control. ACS accomplishes this by utilizing three configuration files in its default setup: 1. The main file outlines the entire cluster. 2. The definition file includes the specifications of critical applications. 3. The configuration file specifies the service group for the application. The proposed system is designed to provide high availability to the defined critical applications in the system. The system is divided into five modules namely 1. ACS-CM (ACS-Communication) 2. ACS-D (ACS-Daemon) 3. ACS-M (ACS-Monitor) 4. ACS-R (ACS-Recovery), 5. ACS-Log File.

ACS-CM

The communication module is responsible for the communication between clients and servers

- Servers-to-Server Communication: This module is responsible for the communication between two server systems. The inter-node communication network between two server systems is called as heartbeat private network. Two servers communicate with each other through Heartbeat Packets, to share the information about currently running and crashed applications o each

server. Heartbeat packets are sending after a configured interval of time. A heartbeat contains all the client commands that client has given to the server and the application crashed reports.[5]

- Client-Server Communications: This module is responsible for the communication between server and client for giving administrative commands to server and it forms cluster administration interface (CAI). The client can send requests to display the configuration and definition of all the applications placed under cluster control. A client can fire command to any server to online or offline the critical application in configuration file. A client can also modify the configuration file at run time and can save changes. Multiple clients can connect to the single server.

ACS-CM is a simple peer to peer network. The network uses simple socket interface and TCP/IP network protocol suited for inter-process communication between two server systems and between client and server for message passing between them.

ACS – D

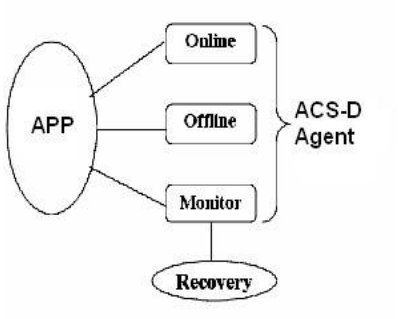

Fig. 2. *ACS-D Agents*

The Application Cluster Daemon, abbreviated as ACS-D, serves as the primary ACS Daemon on each server within a cluster, often referred to as the ACS-D Engine. Its duties include constructing the active cluster configuration from configuration files, executing client commands, identifying critical applications within the system, managing the start and stop of applications (SSA), overseeing resources, and implementing corrective measures in case of failures. ACS-D functions as a replicated state machine (RSM) [10], ensuring that each node maintains a fully synchronized perspective of the resource status across all nodes.

ACS-D is utilized to ensure the availability of cluster components that are managed under cluster control. The ACS-D engine employs agents to oversee and administer resources. Resource state information is gathered from ACS agents on the primary system and shared with its counterpart (secondary) cluster members, and vice versa. Each system's local engine also receives updates from other cluster members to maintain an accurate view of the cluster. Every instance of ACS-D adheres to the same code path for corrective measures when necessary. The system comprises two primary modules and several user controls to simplify the user interface.[6]

ACS – SSA

This module is responsible for starting and stopping critical application (SSA) on any one of the systems by sending commands through ACS-CAI to particular system. ACS system has ACS-online agent per application type to make application or application service group online. Shell scripts are used to start the applications.[11] Once the application has been started by sending command then application running on that system is saved in log file and the information about online application is sent to other system through heartbeat packet. The applications running on one system are also being saved in the log file of another system. ACS system has ACS-offline agent per application type to make that type of application or group of application Offline. Shell scripts are used to stop the running application. Once the application running on the system is offline, then same thing will be recorded on both the system.[7]

ACS – M

This module is responsible for monitoring the application online by the client. After every interval of time the application running on any node is monitored to detect any application failure.[11] If any application or application service group is crashed then server is responsible for sending the crashed reports to the other server. All the applications which are to be monitored are stored in a database. This module is tasked with overseeing the well-being of the other system. It achieves this through the heartbeat protocol, which operates on both systems. The heartbeat protocol is in charge of dispatching heartbeat messages to every other system within the cluster at consistent time intervals. This process indicates whether the system is operational or not.

ACS – R

This module is responsible for recovery of the crashed applications. This module is responsible for restarting the entire group or the individual application on the redundant system which was previously running on the other system.[12] For this it maintains the recovery database which contains the list of all the currently running applications on other systems. Whenever the applications running on the other system fail, the second system uses this list to recover then in a particular state.

Cost and Resource Analysis

ACS minimizes the total cost of ownership by utilizing standard hardware, open-source Linux, and Ethernet-based communication. In contrast to Kubernetes-based systems, which necessitate container runtimes, orchestration layers, and expert management, ACS eliminates virtualization overhead and complex control planes [13][14]. The lack of shared storage and licensing fees further cuts down on deployment and maintenance costs, making ACS an ideal choice for budget-conscious academic institutions and small businesses.

Educational Deployment Model

ACS is ideally designed for instructing students in cluster computing and fault-tolerant systems. Its clear architecture allows learners to grasp concepts like heartbeat protocols, replicated state machines, and fail over mechanisms without the complexity of abstraction [15]. ACS can be implemented on current lab equipment, facilitating practical exercises such as fault injection, configuration adjustments, and recovery analysis, which enhances the practical comprehension of designing high-availability systems [16].

Failure Handling Scenarios

ACS utilizes heartbeat-based monitoring to identify failures in applications and nodes. When an application crashes, it automatically restarts or switches over to a standby node, leveraging synchronized recovery logs [17]. If a node completely fails, the secondary node is elevated to primary status, ensuring minimal disruption to services. Configurable timeout thresholds help prevent false failure detections due to temporary network delays [18].

Conclusion

The above proposed model can be used as a learning tool in educational institutions. It's costing is very low and can be used to teach parallel processing and cluster computing. Clustering is an upcoming concept and thus this simplified version should be used to educate the students. It can also be used for small enterprises and scientific computing. Using the above proposed model makes the system flexible, mobile and reliable.

Full synchronization between the primary and secondary servers and the ability to promote secondary servers into primary servers are a few important characteristics of the above model. This system design provides the configuration framework to place critical application under cluster control for high availability at low cost, ease of installation and low maintenance.

References

1. Apon, A, Buyya, R, Hai Jin, Mache, J. Dept. of Comput. Sci. & Eng., Arkansas Univ., Fayetteville, AR "Cluster computing in the classroom: topics, guidelines, and experiences" Proceedings. First IEEE/ACM International Symposium on Cluster Computing and the Grid, 2001
2. http://sunsite.unc.edu/Linux/linux-ha/High-Availability-HOWTO.html
[3]http://www.stanford.edu/dept/itss/docs/oracle/server.101/b10726/hadesign.h
3. tm
4. Ahrens, G. Chandra, A. Kanthanathan, M. Cox, D.P. IBM Corp., Austin, TX "Evaluating HACMP/6000: a clustering solution for high availability distributed systems" Proceedings of IEEE Workshop on Fault-Tolerant Parallel and Distributed System Systems, 1994.
5. http://www.drbd.org/users-guide/ch-heartbeat.html
6. Vijay Chaurasiya1, Prabhash Dhyani2, Siddharth Munot3 "Linux Highly Available (HA) Fault-Tolerant Servers" 10th International Conference on TT 2007 IEEE
7. P. Somasekaram, R. Calinescu, and R. Buyya, "High-Availability Clusters: A Taxonomy, Survey, and Future Directions", 2021.
8. S. A. B. Muntaka *et al*., "A Hybrid Integrated High Availability Cluster (iHAC) for Enhancing System Resilience and Addressing Persistent Cyber Threats," SSRN, May 2025.
9. R. Gupta, "Development of a High-Availability Cluster for Fault-Tolerant Computing," IJEET, vol. 11, no. 1, pp. 132-141, Feb. 2020.
10. Kopetz, H. "Real-Time Systems: Design Principles for Distributed Embedded clus." Springer, 2nd Edition. (2011).
11. Pacemaker Project Contributors "Pacemaker Explained – Architecture and Concepts." Open-source project documentation. (2023).
12. Zhang, Q., Chen, M., & Li, X. "Fault-Tolerant Cluster Architecture for Cloud-Based Services."Journal of Cloud Computing: Advances, Systems and Applications, Springer. (2020).
13. B. Burns, B. Grant, D. Oppenheimer, E. Brewer, and J. Wilkes, "Borg, Omega, and Kubernetes," ACM Queue, vol. 14, no. 1, 2016.
14. P. Pahl, "Containerization and the PaaS cloud," IEEE Cloud Computing, vol. 2, no. 3, pp. 24–31, 2015.

15. A. Tanenbaum and M. van Steen, Distributed Systems: Principles and Paradigms, 2nd ed., Pearson, 2017.
16. R. Buyya, High Performance Cluster Computing, Prentice Hall, 1999.
17. L. Lamport, “Time, clocks, and the ordering of events in a distributed system,” Communications of the ACM, vol. 21, no. 7, pp. 558–565, 1978.
18. E. A. Brewer, “Towards robust distributed systems,” Proceedings of PODC, 2000.