International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 07 Issue: 12 | December - 2023 SJIF Rating: 8.176 ISSN: 2582-3930

Remote Service Manager using Zookeeper

N. Duraimurugan[1] , Dhivya S[2] , Keerthana S[3]
Department of Computer Science and Engineering, Rajalakshmi Engineering College, Chennai
Email:duraimurugan.n@rajalakshmi.edu.in [1], 200701067@rajalakshmi.edu.in [2],
keerthanajayanthi391@gmail.com [3]

Abstract: Navigating the intricate landscape of service management and monitoring within on-premises applications, burdened by extensive legacy codebases, presents challenges. The complexities associated with deploying applications to the cloud for enhanced management are worsened by the complexity arising from vast on-premises applications. The mandatory process of containerization becomes an intricate and daunting task. The lack of centralized management worsens service control fragmentation, leading to inefficiencies. Manual interventions hinder progress due to potential errors. Scalability challenges increase with more machines and services, compounded by a lack of real-time insights into performance. Dependencies on tools limit adaptability, adding complexity to administering services across diverse machines, requiring extensive training. In response, the proposed "Service Manager" emerges as a centralized tool, designed to overcome these limitations. The Service Manager orchestrates diverse operations on services across machines, offering a user-friendly web interface for seamless triggering and monitoring. Accompanied by the Service Manager Agent on each machine, it interacts with services, leveraging Apache Zookeeper for streamlined coordination, ensuring efficient execution of operations. This solution empowers administrators, marking a pivotal advancement in managing and monitoring services within complex and dynamic environments.
Keywords: Remote Monitoring, coordination, apache zookeeper

1. INTRODUCTION

The way that computing resources are utilized has fundamentally changed with the advent of distributed computing. It enables smooth collaboration across several computers and systems over a network, combining their resources and computing capacity to take on challenging tasks or accomplish shared goals. Under this model, a single task is divided into multiple smaller tasks, each of which is competently handled by a separate system. As an alternative, distributed computing can potentially take the form of several software components and services operating simultaneously on various machines. With the help of these distributed systems, modern computing may now be more efficient, scalable, and fault-tolerant.

Managing a large number of services, frequently each operating on a separate system, is a daily struggle in many organizations. In order to handle this complexity, a consolidation tool that is capable of thoroughly monitoring every active service becomes essential. With its centralized management and control, this consolidation tool is a shining example of a distributed computing application. It gives administrators the ability to carry out a number of crucial tasks, including installing, upgrading, and terminating services all at once. This simplifies processes and boosts output. A system designed to make managing a variety of services in a distributed computing environment easier, the Service Manager is best represented by this particular service.

However, the difficulties in a distributed context go beyond just managing services. Without a centralized coordination service, managing centralized configuration data and metadata, monitoring every network node, and enabling group services are difficult tasks. In this case, Apache Zookeeper stands out as the key component, serving as a centralized coordination service. It offers the fundamental foundation

needed to handle the complexities of distributed systems with ease. Keeping order and coherence in the face of complexity, Apache Zookeeper allows organizations to fully utilize their networked resources by ensuring that the distributed computing ecosystem runs smoothly and efficiently through tight integration with the Service Manager tool.

2. LITERATURE SURVEY

This paper**[1]** explores the automated generation of distributed implementations for logically centralized service orchestrations, focusing on ensuring correctness and managing synchronization and consensus challenges. It introduces a choreography-based model that offers scalability, fault tolerance, adaptability, and synchronization. Challenges include potential scalability issues, increased network traffic, synchronization delays, higher memory consumption, and complexities in validating choreography-based applications.

This paper**[2]** introduces Cognitive Application Area Networks (AAN), a novel approach that separates application management from infrastructure orchestration. The main disadvantages that are present in this system is that it encompasses vendor lock-in, complex management, cost escalation when relocating virtual machines, and operational complexities in heterogeneous infrastructures.

A study **[3]** emphasizes the importance of observability in cloud environments, highlighting three key pillars: logs, metrics, and tracing. It delves into both whitebox and blackbox monitoring approaches, underscoring the significance of collecting data from both applications and infrastructure. But the implementation of this system includes increased data volume impact, increase in complexity, intricate application-infrastructure relationships, and the lack of detailed resource allocation information.

A new method Heterogeneous Task Allocation Strategy (HTAS)**[4]** is introduced and this is tailored for cloud workloads characterized by varying business volumes. Nevertheless, it has its limitations, such as only supporting heterogeneous VM sizes and not types. Container migrations through CRIU may introduce downtime and network latency issues, restricting its applicability. Future goals include addressing instance acquisition lag and enhancing workload prediction to further refine auto scaling capabilities.

This paper**[5]** presents a decentralized framework designed to autonomously deploy and scale containerized applications on edge devices with limited resources. However, this pioneering approach also faces difficulties related to managing the complexity of decentralized container deployment, addressing resource constraints on edge devices, and ensuring robust security measures.

A study **[6]** investigates container technology, its adoption, and container orchestration, analyzing aspects such as scheduling, load balancing, fault tolerance, and autoscaling. Even though the idea is novel it becomes increasingly difficult to include fine-grained application quality of service awareness, robust container monitoring, and ongoing security concerns within diverse adoption requirements.

The **[7]**Cloud Monitor system is specifically tailored for cloud computing environments, featuring multi-level monitoring servers and web-based interfaces for centralized oversight and management. However, the study's limitations include a narrow scope,

oversimplified metrics, a small sample size, reliance on outdated technology, and a notable absence of cost analysis, collectively diminishing the experiment's relevance and applicability in real-world cloud computing scenarios.

A comprehensive overview of Chef**[8]** was studied and its insights were found, focusing on its core concepts and practical applications in cloud-native and heterogeneous computing environments. However, the paper notes that mastering Chef's domain-specific language (DSL) may present a learning curve for some users, particularly those who are new to infrastructure as code.

This paper**[9]** underscores the fundamental concept of Chef automation and its three nodes, offering a comprehensive, step-by-step guide to its implementation, covering various tasks such as virtual machine provisioning, MySQL database configuration, virtual private cloud management, and subnet setup. However, the challenges emerge when migrating applications and the necessity to automate both virtual and physical infrastructure. While Chef Automation can significantly streamline and automate processes, it may require a certain level of expertise in writing cookbooks using the Ruby programming language.

The real-time server performance**[10]** was monitored and it uses Prometheus and Grafana, with Prometheus collecting metrics from diverse sources and Grafana offering a visualization platform. While the paper mentions limitations of the previous monitoring system (CloudWatch), it also lacks a comprehensive comparison or evaluation of Prometheus and Grafana. Additionally, a more in-depth exploration of technical implementation details would be beneficial.

This paper**[11]** outlines a methodology for constructing a monitoring framework for a distributed cloud application, utilizing Prometheus and Chef. The method, however, necessitates tailoring the monitoring framework to specific use cases, and its complexity requires a strong grasp of various tools and commands, highlighting the importance of comprehensive documentation and expertise in the process.

[12]PROMETHEUS presents a structured framework comprising seven stages for the creation of domain-specific usability heuristics, emphasizing the development of artifacts and quality indicators through iterative refinement. While it enhances the precision and standardization of heuristics for system monitoring, it may be considered complex and reliant on surveys, potentially limiting practicality.

This paper**[13]** outlines an approach that leverages Puppet 5 and a range of utilities to achieve centralized configuration management, automated testing, and deployment across diverse systems, thereby improving efficiency and compliance. The difficulties include scaling Puppet compilation, extending module code testing, safeguarding critical files, and improving intrusion detection.

A multi-level heartbeat protocol**[14]** was studied and it features multipoint and bidirectional components, with self-adapting load balancing and fault tolerance mechanisms aimed at enhancing system reliability and performance. However, this approach comes with increased complexity, resource consumption, scalability challenges, configuration intricacies, network overhead, susceptibility to false alarms, and the essential requirement for robust security measures to mitigate potential vulnerabilities.

The paper**[15]** introduces a dynamic keep-alive messaging strategy for timely loss-of-contact detection in mobile pervasive systems, offering the flexibility to adjust message frequency according to application requirements. However, potential complexities arise in implementing and managing dynamic keep-alive messaging, necessitating careful tuning and potentially introducing challenges in system configuration and maintenance.

This study**[16]** tackles the issue of recovery latency in fault-tolerant coordination services, with a specific focus on ZooKeeper. However, it's important to note that the study's primary focus is on addressing ZooKeeper's limitations and may not fully account for the broader spectrum of fault-tolerant coordination services, potentially limiting the generalizability of its findings to other similar systems.

The paper**[17]** outlines a methodology focused on assessing various versions of the ZooKeeper atomic broadcast (Zab) protocol, considering different fault assumptions and workloads. The approach involves conducting experiments with concurrent clients to gauge latency and throughput for write requests. The proposed enhancements to the Zab protocol demonstrate improved performance and fault tolerance, rendering them valuable for distributed systems. These modifications are adaptable and easily integrable into existing implementations, thereby augmenting the efficiency and reliability of distributed applications, particularly in scenarios characterized by high workloads and rigorous fault-tolerance demands. The study does not include evaluations of ZooKeeper's primary use cases which have limitations in practical real life applications.

This paper[18] introduces Google Borg, a large-scale cluster management system that harnesses containerization and distributed scheduling. Borg employs a declarative configuration language and autonomous management modules to optimize resource utilization, ensure high reliability, and simplify application management. However, it's important to note that Google Borg's complexity and resource requirements may present challenges for smaller organizations seeking to implement it. Its non-open-source nature restricts its accessibility to external users.

The paper[19] introduces the PalCom middleware's device discovery mechanism, which improves device detection across a variety of networks. This approach provides efficient discovery and undiscovery while minimizing traffic by using discovery event notifications instead of constantly bombarding networks with keep-alive messages. Furthermore, it builds an overlay communication substrate to provide a connection between network islands, improving cross-network heartbeat for service discovery. Though a thorough assessment of the mechanism's applicability and performance impact is not stated directly, the paper concentrates on introducing the device discovery.

In the paper[20] coordination mechanisms for managing large distributed relational process structures are introduced by the object-aware process management method, which emphasizes decentralization and distribution to boost performance and maintainability. The process consists of phases for decentralized coordination, a switch from central to decentralized coordination, and a validation of the proof-of-concept. It is advantageous to use decentralized process coordination for complex structure coordination because it provides benefits including improved performance, scalability, flexibility, fault tolerance, and lower communication overhead. It is possible that this technique will add complexity to the design and

management of decentralized coordinating processes, necessitating meticulous planning and collaboration amongst several coordinators.

The overview of the literature describes the problems facing modern computing, including distributed systems, cloud computing, and containerized settings. The survey highlights the need for solutions by focusing on issues related to scalability, security, and networks. Complex management problems are introduced when deploying containers on edge devices. Although there are benefits to containers, scalability and security issues must be addressed. Chef, Prometheus, Puppet, and ZooKeeper technology discussions highlight their advantages and disadvantages. Chef's domain-specific language presents a challenge, but its comprehensive configuration management capabilities stand out. PROMETHEUS performs well in usability heuristics but struggles with subjective evaluations and complexity. Puppet has a complex methodology, scalability concerns, and security requirements, but it maintains nodes efficiently. ZooKeeper enhances performance, but real-world implementation may pose challenges, requiring careful consideration.In conclusion, the survey examines issues with modern computing, stressing the significance of resolving network, security, and scalability issues.

3. Problem Statement

The management and coordination of services across distributed environments faces complexities due to the absence of a centralized solution, resulting in fragmented processes, frequent manual interventions, and added complexity to maintenance efforts. Furthermore, the lack of a standardized framework for service coordination compounds this issue. To address these issues, there is a pressing need for a solution that can streamline management through the provision of a centralized interface for operations. This solution should guarantee the consistent execution of tasks and optimize resource utilization. The ultimate goal is to elevate operational efficiency, ensure uniform service operations, and simplify the often burdensome maintenance tasks associated with distributed environments.

4. Proposed system

In response to the issues explained in the problem statement and the insights gained from the literature survey, the proposed system is designed to provide an all-encompassing solution. This solution must not only comprehensively address the above mentioned challenges but also ensure reliability and efficiency. The aim is to offer a system that effectively mitigates the identified problems.

4.1 Service Manager Components

The Service Manager tool consists of three main components:

- Service Manager web UI
- Service Manager Agent
- Apache Zookeeper

4.1.1 Service Manager web UI

The Service Manager web UI is a web application which can be used to view the cluster of members installed in a system, perform an operation on individual members, perform rolling operations, i.e., operations over multiple members in a specified order, view and modify configurations and even upload a custom action. The web UI monitors the cluster and updates the UI about the status and/or changes in the cluster when an action is performed. Also, the status information is sent to Zookeeper as and when the action happens.

4.1.2 Service Manager Agent

The Service Manager Agent is basically another service which is supposed to run in the same machine as the service being monitored. It performs two tasks: it fetches the status of the service and sends it to Zookeeper, and it performs the required operation (triggered in the web UI) on the service.

4.1.3 Apache Zookeeper

Apache Zookeeper is responsible for coordinating all the members/nodes in the cluster. It interacts with both the SM web UI and SM Agent, and it receives status updates from them. It also sends the appropriate actions to the corresponding nodes. It also stores required persistent data (such as configurations) in its ensemble.

4.2 Service Manager Workflow

The high-level working of Service Manager Manager is as follows:

1. The user triggers an action through the SM web UI. The SM web UI sends the action information to Zookeeper.
2. Zookeeper sends this action information to the corresponding SM Agent(s).
3. SM Agent performs the required operation on the service, which is present in the same system as the SM Agent.
4. SM Agent will send the update in the service status to Zookeeper, which is in turn sent to the SM web UI. The SM web UI displays these changes accordingly to the user.
5. The applications being monitored are represented as Nodes. Their status information is therefore obtained from their MBean projections.
6. In the case of rolling actions, the action is performed on multiple nodes in a specific order. This order is configurable through the SM web UI.

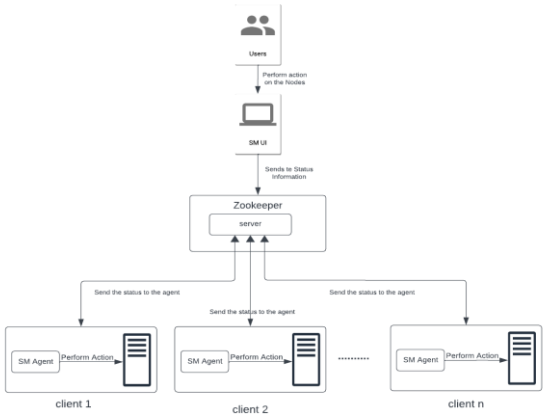

Fig. 4.1 Architecture of Service Manager

The above image represents the flow of work in Service Manager, from the user to the component servers and back.

5. Implementation

The implementation focused on agent registration in the client node with the Zookeeper server. The selected technology stack incorporated cutting-edge tools, featuring Spring Boot 3 and Java 17 for robust application development, while Zookeeper assumed a pivotal role in the centralized management of all systems within the cluster. The project's practical implementation involved the establishment of a user-friendly interface. This interface serves as a control center, empowering users with the ability to effortlessly monitor the health and performance of nodes within the network. Through this interface, users can access vital metrics and status updates, fostering a proactive approach to system oversight.

The tool's capabilities extend beyond static monitoring. It enables real-time surveillance of all nodes in the cluster, fostering a dynamic environment where changes are promptly detected and addressed. This real-time

functionality not only enhances the user experience but also contributes to the efficiency of system monitoring.

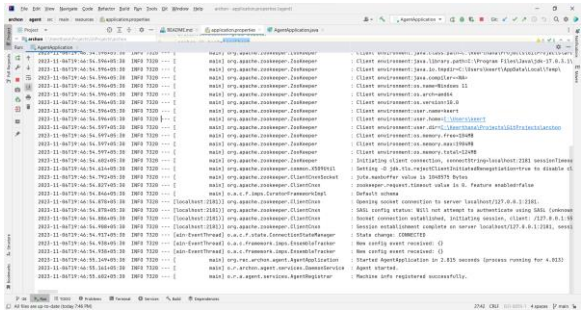

Fig 5.1 agent started in the client

Fig 5.1 depicts the initiation of the agent program in the client. The agent program is started, and it autonomously registers itself with the server to announce the node's participation in the cluster.

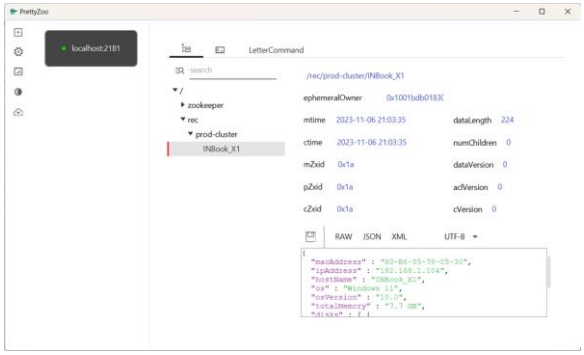

Fig 5.2 Node connected to the Server

Fig 5.2 illustrates the agent initiated in the previous screenshot as it establishes a connection to the server. The connected agent is then represented under the persistent zNode 'prod-cluster.' This depiction highlights the crucial step of the agent's connection to the server and its association with the designated persistent zNode within the cluster architecture.

6. Conclusion

Thus, a centralized tool which provides one place to control multiple services in different remote machines is built successfully. Service Manager is an effective distributed computing application with an intuitive and easy-to-use user interface. It leverages the features of Apache Zookeeper in order to coordinate and monitor the services running in all the nodes in the cluster it is supposed to manage. It is easily customizable to suit the user's needs. It provides the user with a slew of default actions that can affect a single node or the entire cluster. It also permits the user to define their own custom node/cluster-wide action as well. The primary objective is to dynamically view the status of all nodes in the cluster without human intervention, ensuring the system promptly recognizes the addition of a new system to the cluster. The Service Manager thus emerges as a robust solution for remote service management, streamlining operations and promoting effective monitoring in distributed computing environments.

Reference

[1] L. Mostarda, S. Marinovic and N. Dulay, "Distributed Orchestration of Pervasive Services," 2010 24th IEEE International Conference on Advanced Information Networking and Applications, Perth, WA, Australia, 2010, pp. 166-173, doi: 10.1109/AINA.2010.100.

[2] Mikkilineni, R., Morana, G., Zito, D. and Keshan, S., 2017. Cognitive application area networks. International Journal of Grid and Utility Computing, 8(2), pp.74-81.

[3] R. Picoreti, A. Pereira do Carmo, F. Mendonça de Queiroz, A. Salles Garcia, R. Frizera Vassallo and D. Simeonidou, "Multilevel Observability in Cloud Orchestration," 2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing

and Cyber Science and Technology Congress(DASC/PiCom/DataCom/CyberSciTech), Athens, Greece, 2018, pp. 776-784, doi: 10.1109/DASC/PiCom/DataCom/CyberSciTec.2018.00134.

[4] Zhiheng Zhong and Rajkumar Buyya. 2020. A Cost-Efficient Container Orchestration Strategy in Kubernetes-Based Cloud Computing Infrastructures with Heterogeneous Resources. ACM Trans. Internet Technol. 20, 2, Article 15 (May 2020), 24 pages. https://doi.org/10.1145/3378447

[5] U. C. Özyar and A. Yurdakul, "A Decentralized Framework with Dynamic and Event-Driven Container Orchestration at the Edge," 2022 IEEE International Conferences on Internet of Things (iThings) and IEEE Green Computing & Communications (GreenCom) and IEEE Cyber, Physical & Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics), Espoo, Finland, 2022, pp. 33-40, doi: 10.1109/iThings-GreenCom-CPSCom-SmartData-Cybermatics55523.2022.00017.

[6] Bairagi, Hritwik, Uday Chourasiya, Sanjay Silakari, Priyanka Dixit and Smita Sharma. "A Survey On Efficient Container Orchestration Tools And Techniques In Cloud Environment." International Journal of Scientific & Technology Research 9 (2020): 1425-1430.

[7] Wang Liwei, Zhang Zhi, Wang Teng, and Liu Rukun. 2020. Research and Application of Cloud Computing Platform Monitoring System Based on Virtual Cluster. In Proceedings of the 2020 International Conference on Computers, Information Processing and Advanced Education (CIPAE 2020). Association for Computing Machinery, New York, NY, USA, 58–63. https://doi.org/10.1145/3419635.3419657

[8] Rajashekhar. "A LITERATURE STUDY ON CHEF - AN OPEN SOURCE SOFTWARE AGENT." (2017).

[9] N. Kumaran, Ramya H, V. Apoorva. "Chef Automation on Google Cloud" International Journal of Scientific & Technology, Volume 10 (2022): 1491-1498

[10] Arun Kumar K, Vinutha B S, Vinayaditya B V. "Real time monitoring of servers with Prometheus and Grafana for high availability". International Research Journal of Engineering and Technology, Volume 6, Issue 4 (2019): 5093-5096

[11] Adamyaa D N, Dinesh Babu S, Priya D, Soumya A. "Building a monitoring framework for a distributed cloud application using promethus and chef". International Journal of Scientific & Technology Volume 10 (2022): 3164-3168

[12] Granizo, Cristhy Jiménez, Héctor Allende-Cid and Ismael Figueroa. "PROMETHEUS: Procedural Methodology For Developing Heuristics Of Usability." IEEE Latin America Transactions 15 (2017): 541-549.

[13] Raychel M. Benson, Edward Munsell, Nicholas Bertrand, Michael Baynton, Evan F. Bollig, and Jeffrey McDonald. 2019. A Multi-Environment HPC-Scale Puppet Infrastructure for Compliance and Systems Automation. In Proceedings of the Practice and Experience in Advanced Research Computing on Rise of the Machines (learning) (PEARC '19). Association for Computing Machinery, New York, NY, USA, Article 29, 1–8. https://doi.org/10.1145/3332186.3332240

[14] F. -f. Li, X. -z. Yu and G. Wu, "Design and Implementation of High Availability Distributed

System Based on Multi-level Heartbeat Protocol," 2009 IITA International Conference on Control, Automation and Systems Engineering (case 2009), Zhangjiajie, China, 2009, pp. 83-87, doi: 10.1109/CASE.2009.115.

[15] Johnsson, BA, Nordahl, M & Magnusson, B 2017, 'Evaluating a Dynamic Keep-Alive Messaging Strategy for Mobile Pervasive Systems', Procedia Computer Science, vol. 109, pp. 319-326

[16] Li, Haoran, Chenyang Lu and Christopher D. Gill. "RT-ZooKeeper: Taming the Recovery Latency of a Coordination Service." ACM Transactions on Embedded Computing Systems (TECS) 20 (2021): 1 - 22.

[17] EL-Sanosi, Ibrahim and Paul D. Ezhilchelvan. "Improving Zookeeper Atomic Broadcast Performance When a Server Quorum Never Crashes." EAI Endorsed Trans. Energy Web 5 (2018): e11.

[18] Verma, A., Pedrosa, L., Korupolu, M., Oppenheimer, D., Tune, E. and Wilkes, J., 2015, April. Large-scale cluster management at Google with Borg. In Proceedings of the tenth European conference on computer systems (pp. 1-17).

[19] Amr Ergawy, Boris Magnusson,Device Discovery for the PalCom Pervasive Middleware with Eliminated Cross-networks Periodic Heart-beat Messages,Procedia Computer Science,Volume 37,2014,Pages 64-71,ISSN 1877-0509, https://doi.org/10.1016/j.procs.2014.08.013.

[20] Steinau, S., Andrews, K. & Reichert, M. Coordinating large distributed relational process structures. Softw Syst Model 20, 1403–1435 (2021). https://doi.org/10.1007/s10270-020-00835-0