International Journal for Innovative Engineering and Management Research

PEER REVIEWED OPEN ACCESS INTERNATIONAL JOURNAL

www.ijiemr.org

IJIEMR Transactions, online available on 07th Sept 2023. Link :http://www.ijiemr.org/downloads.php?vol=Volume-12&issue=Issue 09

10.48047/IJIEMR/V12/ISSUE 09/07

Title **Investigation of Distributed Embedded System Frameworks**

Volume 12, ISSUE 09, Pages: 62-68

Paper Authors **Dr. S M Shamsheer Daula, Dr. G Ramesh, Dr. G Amjad Khan, Mr. M Madhusudhan Reddy**

International Journal for Innovative Engineering and Management Research
PEER REVIEWED OPEN ACCESS INTERNATIONAL JOURNAL
www.ijiemr.org

Investigation of Distributed Embedded System Frameworks

[1]Dr. S M Shamsheer Daula, [2]Dr. G Ramesh, [3]Dr. G Amjad Khan, [4]Mr. M Madhusudhan Reddy
[1] Associate Professor, [2] Associate Professor, [3] Associate Professor, [4] Assistant Professor
[1][2][3][4] Department of ECE, G Pulla Reddy Enginrreing College (A), Kurnool, 518007,
Andhra Pradesh, India.
[1]shamsheer.ece@gprec.ac.in, [2] ramesh.ece@gprec.ac.in, [3]amjadkhan.ece@gprec.ac.in,
[4]msreddym11.ece@gprec.ac.in

Abstract— **The manner that traditional systems are built is as a centralized system. In addition to this conventional strategy, decentralized systems and distributed systems are also frequently used. The trend in system construction is moving away from centralized systems and toward distributed systems. These three strategies are described and contrasted in the section that follows. Traditionally, embedded systems have been centralized. One controller manages the various system components in such a system. Sensors and actuators are among the component parts that are frequently near to one another and connected directly to the controller. This kind of system can be simply implemented in small-scale systems. The direct and quick management of sensors and actuators without the use of a hierarchical control system makes it ideal for small system that demands exceptional performance. A good choice for real-time system implementation, such a system has low communication costs between sensors and controller. However, centralized systems do not scale well. The complexity of handling the control of the units would rise as the number of component units increased, leading to an increase in connections as well. The controller would need to have a large bandwidth and high performance to handle all the operations due to the communication demand. Additionally, the physical routing and arrangement of the sensors and actuators would be restricted by the central control unit.**

Key Words: — ***client, peer, server, embedded system, distributed system.***

I. INTRODUCTION

A distributed real-time embedded system works like other constant implanted frameworks, yet with a serious level of heterogeneity. The heterogeneity permits the framework to perform equal assignments that have dierent necessities on the equipment, which in wording benefit the usefulness of the framework. A DRE framework is heterogeneous in numerous viewpoints, from equipment designs and programming parts to the planning arrangements, correspondence conventions and memory administrations. [1]. he handling execution and adaptability of the framework can profit from the intricacy of the heterogeneity. Such intricacy, be that as it may, presents new difficulties in keeping up with and planning the framework. It likewise expands the decultures for engineers to program equal applications that can completely use the handling potential of the system. [2-8].

The correspondence across hubs and the planning and planning of errands in a DRE framework can be confounded. A middleware is in many cases utilized in such framework to keep up with the coordination and to permit the different free equipment parts of the organization function overall framework. Aside from the correspondence and dispatching of errands, the framework likewise needs to meet the ongoing requirement also as different limitations for an inserted framework. This raises the test for planning a middleware for a DRE framework.

II. RELATED WORK

There have been several studies and research papers focusing on the performance analysis of selective memory balancing techniques in architecture analysis. Here are some related works in this area:

Title " Reconfiguration Strategies for Critical Adaptive Distributed Embedded Systems”" Authors: Adam Ballest., Johnson, A., Brown, M. Conference/Journal: IEEE Distributed System, 2020

This paper proposes a machine learning-based memory balancing technique for Distributed Embedded devices. It employs a predictive model to analyze the data patterns and dynamically allocate memory resources. The authors evaluate the technique's performance using various distributed Embedded workloads and demonstrate its effectiveness in improving memory utilization and reducing data loss.

Title: "Performance Analysis of Data Compression Techniques in Distributed Embedded Systems using Machine Learning" Authors: Lee, S., Kim, H., Park, J.Conference/Journal: ACM Transactions on Internet of Things, 2019

This study focuses on analyzing the performance of data compression techniques in Distributed Embedded systems using machine learning. The authors compare different compression algorithms and evaluate their impact on memory utilization, processing time, and energy consumption. They leverage machine learning models to predict the optimal compression technique for a given Distributed Embedded workload.

Title: "An Experimental Study on Data Offloading Techniques for Memory-Constrained Distributed Embedded Devices" Authors: Chen, L., Zhang, Q., Li, L.Conference/Journal: International Conference on Mobile Computing and Networking, 2018

This research investigates the performance of data offloading techniques in memory-constrained Distributed Embedded devices. The authors conduct experiments to analyze the impact of offloading strategies on memory utilization, network latency, and energy consumption. They employ machine learning algorithms to predict the most suitable data offloading technique based on the device's available memory and network conditions.

Title: "Performance Evaluation of Data Aggregation Techniques in Distributed Embedded Networks using Machine Learning" Authors: Wang, X., Li, C., Zhang, Y. Conference/Journal: IEEE Transactions on Network Science and Engineering, 2021

This paper presents a performance evaluation of data aggregation techniques in Distributed Embedded networks using machine learning. The authors compare different aggregation algorithms and assess their efficiency in reducing data transmission overhead and conserving memory resources. They utilize machine learning models to predict the optimal aggregation technique based on the Distributed Embedded network's characteristics.

These related works provide insights into the performance analysis of selective memory balancing techniques in Distributed Embedded using machine learning. They contribute to the understanding of the benefits and limitations of different techniques and help in identifying optimized memory management strategies for Distributed Embedded devices.

III. ARCHITECTURAL REPRESENTATIONS

Centralized systems are frameworks that utilization client/server design where at least one client hubs are straightforwardly associated with a focal server. This is the most ordinarily involved kind of framework in numerous associations where a client sends a solicitation to an organization server and gets the reaction.

Machine learning techniques have emerged as a promising approach to enhance page memory management by leveraging the ability of models to learn patterns and make intelligent decisions. We discuss various machine learning-based approaches, including prediction models, reinforcement learning, and neural networks, and their impact on improving memory allocation, reducing page faults, and enhancing overall system performance.

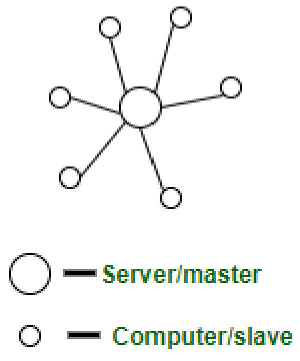

Figure 1. Centralized Model.

As mentioned in figure 1, Consider a huge server to which we send our solicitations and the server answers with the article that we mentioned. Assume we enter the hunt term 'low quality food' in the Wikipedia search bar. This search term is sent as a solicitation to the Wikipedia servers (generally situated in Virginia, U.S.A) which then, at that point, answers back with the articles in view of pertinence. In this present circumstance, we are the client hub, Wikipedia servers are the focal server.

Qualities of Unified Framework -

Presence of a worldwide clock: As the whole framework comprises of a focal node(a server/an expert) and numerous client nodes(a PC/a slave), all client hubs sync up with the worldwide clock(the clock of the focal hub).

One single focal unit: One single focal unit which serves/arranges the wide range of various hubs in the framework.

Subordinate disappointment of parts: Focal hub disappointment makes the whole framework come up short. This seems OK since when the server is down, no other element is there to send/get reactions/demands.

Scaling -

Just upward scaling on the focal server is conceivable. Flat scaling will go against the single focal unit normal for this arrangement of a solitary focal element.

Can't increase upward after a specific cutoff - After a breaking point, regardless of whether you increment the equipment and programming capacities of the server hub, the exhibition won't increment obviously prompting an expense/benefit proportion < 1.

Bottlenecks can seem when the traffic spikes - as the server can have a limited number of open ports to which can pay attention to associations from client hubs. In this way, when high traffic happens like a shopping deal, the server can basically experience a Forswearing of-Administration assault or Conveyed Disavowal of-Administration assault.

Simple to get truly. It is not difficult to get and support the server and client hubs by righteousness of their area

Smooth and rich individual experience - A client has a devoted framework which he uses(for model, a PC) and the organization has a comparable framework which can be changed to suit custom necessities. Committed assets (memory, computer chip centers, and so forth) More expense productive for little frameworks up to a specific breaking point - As the focal frameworks take less assets to set up, they have an edge when little frameworks must be fabricated. Speedy updates are conceivable - Just a single machine to refresh. Simple separation of a hub from the framework. Simply eliminate the association of the client hub from the server and presto! Hub withdrew.

Concentrated control: In a unified framework, the focal authority has unlimited authority over the framework, which can prompt better coordination and direction.

Simpler to make due: As there is just a single focal hub to make due, it is more straightforward to keep up with and deal with the framework. Lower idleness: Concentrated frameworks can give lower dormancy contrasted with appropriated frameworks as there is no defer in correspondence between various hubs .Better execution: Unified frameworks can accomplish better execution as the assets can be upgraded for explicit undertakings.

Less complex execution: Concentrated frameworks are more straightforward to carry out as they require less perplexing calculations and conventions.

Weaknesses of Incorporated Framework -

Exceptionally reliant upon the organization availability - The framework can come up short on the off chance that the hubs lose network as there is just a single focal hub.

No effortless debasement of the framework - unexpected disappointment of the whole framework

Less chance of information reinforcement. In the event that the server hub falls flat and there is no reinforcement, you lose the information straight away

Troublesome server upkeep - There is just a single server hub and because of accessibility reasons, it is wasteful and amateurish to bring the server down for support. In this way, refreshes must be finished on-the-fly(hot refreshes) which is troublesome and the framework could break.

Weak link: Incorporated frameworks have a weak link, which can make the whole framework come up short in the event that the focal hub goes down.

Absence of straightforwardness: Unified frameworks need straightforwardness as the focal authority has unlimited oversight over the framework, which can prompt issues like control and inclination.

Security chances: Concentrated frameworks are more powerless against security gambles as the focal authority has total admittance to every one of the information.

Restricted versatility: Brought together frameworks have restricted versatility as the focal hub can deal with a predetermined number of clients all at once. Restricted development: Concentrated frameworks can smother advancement as the focal authority has unlimited oversight over the framework, which can restrict the extension for trial and error and innovativeness.

Uses of Brought together Framework -

Application improvement - Exceptionally simple to set up a focal server and send client demands. Current innovation these days really do accompany default test servers which can be sent off with several orders. For instance, Express server, Django server. Information examination - Simple to do information investigation when every one of the information is in one spot and accessible for examination

Individualized computing concentrated data sets - every one of the information in one server for use. Single-player games like Requirement For Speed, GTA Bad habit City - a whole game in one system(commonly, a PC) Application improvement by conveying test servers prompting simple troubleshooting, simple arrangement.

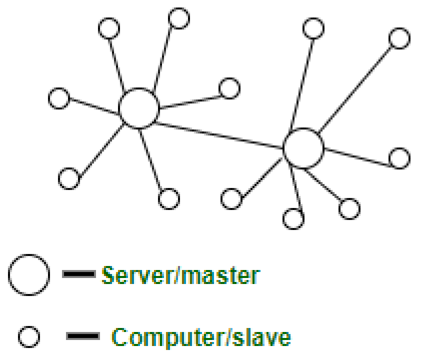

Figure 2. DeCentralized Model

However, as shown in figure 2, It is feasible to scale vertically. Each hub has the ability to add resources (hardware, programming) to create the exhibition, which prompts an expansion in the presentation of the entire framework.

Decentralized Framework Design -

Shared design – all hubs are close pals. No hub is superior to other hubs incomparably. One hub can become an expert by voting and assisting in the organization of a component of the framework, but this does not imply that the hub has superior qualities to the other hub that it is planning.

IV. PROPOSEFD WORK

Problem Statement:

What structure and components are required in a distributed real-time embedded system?

What different protocols and techniques can be used to develop a DRE system?

How does the middleware handle the coordination of the system?

How should the middleware be constructed in order to maintain high scalability and transparency?

There are a ton of viewpoints to be thought about while building a middleware and the focal point of this proposition is to act as need might have arisen in a dispersed constant inserted framework. Taking advantage of how developer is going can deal with the intricacy by embracing conventions and methods that are appropriate for such a framework in the middleware.

The compose support's hit percentage as well as the typical size and quantity of bunches expelled to the Flash memory are taken into consideration. The composing cushion for the preceding techniques is a 4-MB RAM, and the subsequent data is captured in the test section. It does all calculations in a test system that is driven by results. With the number of refresh squares set to 8, in accordance with the recent works, the completely familiar area interpretation layer (FAST) [13] is used as the interpretation layer of server client link. According to the analysis, there are significant gaps between the ideal methodology and the current approaches, indicating that there is a lot of room for improvement. The gaps between the ideal procedure and other approaches are depicted in this picture.

V. PERFORMANCE COMPARISON

Delivery Factor: The number of messages that are left unattended, the number of message loss is comparatively less in the proposed SMB. This is because, the storage overflow is prevented through optimal storage allocation and the messages are categorized based on priority. Message drop occurs when $t_w > t_{cd}$ or $t_w > t_{ed}$. In order to prevent overflow, the priority messages are emptied first followed by the non real-time messages. The number of messages dropped is less as the operation of the appliances is paused when $pa \rightarrow 0$ and also the EMU notifies about the power availability to the Distributed Embedded device. User messages are also restricted by sending passive acknowledgement therefore unnecessary storage overflow and unattended messages are controlled.

Ratio Devices --- Feasibility	5 * N	455 * N	1067 * N	15400 * N
GEL	29	39	45	52
DESD	29	39	46	53
MESD	29	40	47	53
MISD	31	45	53	66

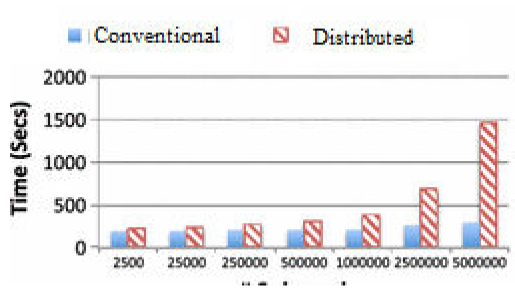

Figure 3: Delivery Factor of various techniques

As shown in figure 4, The number of serviced messages is almost equal to the number of service requests, maximizing the delivery factor in the proposed work

Lifetime: The lifetime of the Distributed Embedded gadget regarding the quantity of solicitations took care of in a day is noticed for SMB and contrasted and the current techniques in figure 5. The activities of the gadget incorporate update of sense and delay records, administration sending and recognizing. These tasks are not intermittent rather it depends on the machine and power accessibility. The activity of the gadget is additionally constrained by EMU through ideal updates in regards to dad. Subsequently, the gadget is kept

Table 1: Comparison of Distributed frameworks

from performing superfluous or intermittent tasks. Besides, the machines associate with the following accessible gadgets in the event of a disappointment, guaranteeing consistent help. This works on the quantity of dynamic gadgets with held measure of energy that draws out the tasks of the gadget somewhat higher than the current methodologies.

The dispersed working and common correspondence between the gadgets control the tasks in the Distributed Embedded climate to hold a higher lifetime of the device

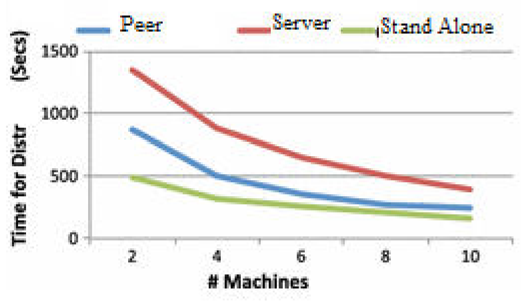

Figure 5: Efficiency of various frameworks

VI. Conclusion

The framework is simultaneousness straightforward to the clients if the instrument of simultaneousness dealing with is stowing away from the clients. This infers that a client doesn't have to deal with the simultaneousness clashes in the event that there is more than one application getting tosimilar shared assets in the framework. The middleware ought to determine such clashes. In the event that a framework is simultaneousness straightforward, the application doesn't have to know the number of uses that are having similar assets. By this straightforwardness, the framework likewise safeguards itself from unlawful access of assets made by application, as the application ought to have no control on direct getting to and locking of the assets. The simultaneousness clashes ought to be dealt with by the middleware yet not the application..

VIII. REFERENCES

[1] D. Harnik, E. Khaitzin, D. Sotnikov, and S. Taharlev. A Fast Implementation Of Deflate. In DCC. IEEE Computer Society, 2014.

[2] L. Shi, C. J. Xue, J. Hu, W.-C. Tseng, X. Zhou, and E. H.-M. Sha, “Write activity reduction on flash main memory via smart victim cache,” in *Proc. 20th Symp. Great Lakes Very Large Scale Integr. Syst., 2014,* pp. 91–94.

[3] Radu Stoica and Anastasia Ailamaki. Improving flash write performance by update frequency. Proc. VLDB Endow., 6(9):733–744, July 2013.

[4] Tola John Odule and Idowun Ademola Osinuga, “Dynamically Self- Adjustin Cache Replacement Algorithm”, International Journal of Future Generation Communication and Networking, Vol. 6, No. 1, Feb. 2013.

[5] Benny Van Houdt. A mean field model for a class of garbage collection in flash-based solid state drives. In Proceedings of SIGMETRICS /InternationalConference on Measurement and Modeling of Computer Systems, 2013.

[6] Saurabh Gao, Hongliang Gao and Huiyang Zhou, "Adaptive Cache Bypassing for Inclusive Last Level Caches", IEEE 27th International Symposium on & Distributed Processing (IPDPS), pp. 1243-1253, 2013.

[7] Liang Shi, Jianhua Li, Chun Jason Xue, and Xuehai Zhou, "Co operating Virtual Memory and Write Buffer Management for flash storage systems", IEEE transactions On Very Large Scale Integration (VLSI) Systems, Vol. 21, No. 4, April 2013.

[8] Daniel A Jimenez, "Insertion and promotion for tree-based Pseudo LRU last- caches", 46th Annual IEEE/ACM International Symposium on Micro architecture, pp. 84-296, 2013.

[9] Fazal Hameed, Lars Bauer and Jorg Henkel, "Adaptive cache management for a combined SRAM and DRAM cache hierarchy for MAQD", Design, & Test in Europe Conference & Exhibition (DATE), pp. 77-82, 2020.

[10] Young-Sik Lee, Sang-Hoon Kim, Jin-Soo Kim, Jaesoo Lee, Chanik Park, and Seungryoul Maeng 2013 IEEE 29th Symposium on, pages 1–13,May 2013.

[11] Cristian Ungureanu, Biplob Debnath, Stephen Rago and Akshat Aranya, "TBF: memory-efficient replacement policy for flash-based caches", IEEE 29th International Conference on In Data Engineering (ICDE), pp. 1117-1128, 2013

[12] Yingying Tian, Samira M. Khan and Daniel A. Jimenez, "Temporal-based Multilevel correlating inclusive cache replacement", ACM Transactions on Architecture and Code Optimization (TACO), Vol. 10, No. 4, Article. 33, 2013

[13] Tripti Warrier S, B. Anupama and Madhu Mutyam, "An application-aware replacement policy for last-level caches", Architecture of Computing Systems–ARCS, Springer Berlin Heidelberg, pp. 207-219, 2013.