STM JOURNALS

Journal of
Advances in Shell Programming

ISSN: 2395-6690
Volume 11, Issue 3, 2024
September–December
DOI (Journal): 10.37591/JoASP

https://journals.stmjournals.com/joasp

Review JoASP

Optimizing System Management: Innovative Approaches to Shell Scripting for Automation in Large-Scale Systems

Mritunjay Kr. Ranjan[1,*], Shilpi Saxena[2], Rohit Gupta[1], Ashish Singh[3], Avishkar Sanjay Anarthe[3], Aniket Anand[4]

Abstract

In today's dynamically changing technology environment, effective systems management is of utmost importance for enterprises that run their applications or services on large-scale infrastructure. In this article, I detail some of these practices as applied to shell scripting, a fundamental building block for process automation in such environments. Using these advanced scripting methods, system administrators can perform routine tasks much more efficiently than they would manually and greatly reduce the chances of human error that come with doing things at a manual level as such. This paper discusses different shell scripting techniques with regard to modular design, error handling, and performance enhancements. Moreover, it discusses the inclusion of version control systems and test frameworks to secure script reliability and maintainability. Case studies and proof best illustrate how these approaches bring about tangible benefits through a drastic reduction of time plus an increase in system reliability. The research emphasizes the significance of shell scripting in automating the execution of system management activities and thereby brings more efficient operations to large-scale systems. With how much organizations rely on automation to manage complex infrastructures this research emphasizes the importance of shell scripting practices keeping pace and employing constant adaptation/improvement for ever-changing needs. This study examines how sophisticated shell scripting approaches can increase automation for large-scale IT infrastructure management, boost productivity, and lower human error rates. System administrators may carry out common tasks far more efficiently with shell scripts than with manual processes thanks to their modular design, error management, and speed optimization. Case studies show how these methods significantly improve system reliability and cut down on task completion times. Script dependability and maintainability are further guaranteed by the use of test frameworks and version control systems. The study highlights the crucial role shell scripting plays in contemporary IT environments and the necessity of changing shell scripting techniques to meet the changing demands of large-scale systems.

***Author for Correspondence**
Mritunjay Kr. Ranjan
E-mail: mritunjaykranjan@gmail.com

[1]Assistant Professor, School of Computer Sciences and Engineering, Sandip University, Nashik, Maharashtra, India
[2]Assistant Professor, Department of Computer Application and IT, Lords University, Alwar, Rajasthan, India
[3]Student, School of Computer Sciences and Engineering, Sandip University, Nashik, Maharashtra, India
[4]Scholar, Department of Computer Science and Information Technology, Magadh University, Bodh Gaya, Bihar, India

Received Date: October 22, 2024
Accepted Date: October 29, 2024
Published Date: November 04, 2024

Citation: Mritunjay Kr. Ranjan, Shilpi Saxena, Rohit Gupta, Ashish Singh, Avishkar Sanjay Anarthe, Aniket Anand. Optimizing System Management: Innovative Approaches to Shell Scripting for Automation in Large-Scale Systems. Journal of Advances in Shell Programming. 2024; 11(3): 32–40p.

Keywords: Shell scripting, automation, system management, large-scale systems, operational efficiency, error handling

INTRODUCTION

Digital transformation demands that companies control larger, increasingly complex IT infrastructures. The larger and more complex a system, the greater the need for its effective management [1]. However, the manual processes of the old are simply insufficient and result in higher operational costs, coupled with a growing number of human errors and increased

© STM Journals 2024. All Rights Reserved

downtime. In response, organizations are increasingly looking at automation with shell scripting, proving a very robust tool in that regard. Shell scripting is an ideal way to automate repetitive tasks, streamline workflows, and improve the system performance. The simplest way an administrator can interact with computers is through shell scripting, which provides a command line interface (CLI) to communicate directly, and a task written in a text file that allows multiple tasks to be executed within a single command [2]. This feature is especially attractive in massive systems because the administration of thousands of servers can soon become burdensome. Automation of routine operations, including backups, updates, and monitoring, allows organizations to reduce the operational burden on their resources so that IT personnel can engage in more strategic initiatives instead of tedious physical labor [3]. Moreover, the versatility of shell scripting makes it possible to complicate workflows by introducing other programming paradigms and tools. More recent shell scripts feature conditional statements, loops, and functions to form modular bits, which are often deployable throughout multiple projects. Scripts can also be made to fail gracefully and reduce the likelihood of failures causing service outages by using error-handling practices. Teams can take advantage of this process by using version control systems to work together on scripts, maintain the history of changes, and go back anytime they need to have their best practices throughout script development. The need for shell scripting is still far from going away because it plays a fundamental role in improving the operability of large-scale systems [4]. Companies using automated shell scripts can remove large amounts of lead time for tasks, such as answering system alerts or increasing service delivery and reducing the time to market. Moreover, automation prevents human errors that, in reality, such as finance, could be catastrophic. This has led to organizations adopting shell scripting as part of their standard system management practices. In this study, we examine new paradigms of shell scripting to allow more automation in large systems. In this analysis, we investigate case studies and best practices that help us identify these techniques, which are the core contributors to optimizing the system management processes. The idea is to give some examples of what these insights should look like and not only demonstrate the shell scripting capabilities in automation but also push for more advanced approaches that match the operational needs of organizations. Above all, these findings underscore the need for advanced changes in system management practices to cater to modern IT environments and show that shell scripting is an indispensable tool toward this end [5].

Objective

- Repetitive tasks are performed through shell scripting to minimize manual errors and operational costs.
- Facilitate the reuse of shell scripts with exception handling that helps streamline system tasks while minimizing downtime.
- Suggesting a version control system for shell scripts can be made use of to encourage teamwork, keep a record of changes, and learn to deal with the scope and diversity of advanced scripting techniques required in today's IT.

RELATED WORK

The authors stated that maintaining a reliable, high-performing inventory management system lowers maintenance costs and minimizes lost opportunities for production output. Nonetheless, the creation and maintenance of such a system is difficult, with intricate and arduous processes. These applications allow companies to go through a tool they will use over the next couple of years to streamline their maintenance spare parts reordering processes. This paper illustrates how this software enables maintenance managers to accurately perform their ABC-XYZ calculations and make quick count entries, thus lowering equipment downtime costs. Thus, the mechanism allows companies to work efficiently and deliver maximal output. Nevertheless, staff training and setting ground rules are vital for the proper execution of an inventory management system based on the CMMS [6].

In this study, an electric power optimization method was developed for a PV-ESS grid-connected microgrid system. The home energy management system (HEMS) can be used to monitor electrical

© STM Journals 2024. All Rights Reserved

usage data in the home, including production systems and electricity consumption. The produced is grid-tied, and health is optimized with load. This results in a decrease in the cost per kWh for electricity, and hence, the overall energy storage system [7].

One of the most important functions of energy management is in a smart farm, which plays a major role in intelligent battery management systems. It describes the general power demand of the entire farm, the amount of energy generated by the PV system, and the solar battery storage capacity. However, for the energy monitoring system to be effective, it must operate well in perceiving how this harvested and available energy may be efficiently distributed. Because the loads may originate from aquaponics, vision systems, irrigation, etc., an effective system is required. Therefore, an intelligent monitoring system records all the potential energy demands at each loading system [8].

This paper proposes an energy management system for microgrids with mainly renewable sources and thermal user-side buildings. The system looks to partner with energy storage to facilitate the deeper penetration of renewables, improve power supply reliability, and lower operational costs. In this study, a modeling approach for building thermal radiation processes is presented. This study also proposed a calculation method for determining the adjustable power of electric heating loads. Finally, a paper presents an energy management system that maximizes microgrid functionalities. The EMS is planned based on forecasted data to form an optimal microgrid operation management plan. It allows dynamic power scheduling through the collection of real-time data and uses a rolling optimization strategy to compensate for prediction errors [18]. User load power control will also have adjustable loads depending on renewable generation so as not to degrade microgrid stability during peak demand times. The simulation results show that the microgrid can use renewable energy more actively and bring great advantages in optimal operation, reliability enhancement with load curve smoothing, and cost reduction [9].

This study aims to design an information system management system for the Child-friendly Integrated Public Space (RPTRA) with the Laravel framework. The RPTRA operational system is an automatic system for managing integrated public space use data and assists in the preparation of RPTRA activity reports. This research implemented an observation, interview, and literature study data collection method as a system development researcher using the Rapid Application Development (RAD) method and the UML diagram tool to design this requirement to be ridden [10].

METHODOLOGY

The methodology for optimizing system management through innovative approaches to shell scripting in large-scale systems is structured around three core components: a theoretical framework, mathematical modeling, and implementation techniques. Each component plays a crucial role in understanding how shell scripts can be utilized effectively for automation [11].

S1: Theoretical Framework

The theoretical framework is based on systems theory, which emphasizes the interdependence of various components within a large-scale system. Automation of tasks can be viewed as a system of inputs (tasks), processes (scripts), and outputs (results). The aim was to minimize the response time and maximize the efficiency of the system [12].

S2: Mathematical Modelling

To mathematically model the efficiency of shell scripting in automating system management, we define several key parameters [13].

- T = Total time taken to execute a set of tasks (in seconds)
- N = Number of tasks to be automated
- t_i = Time taken to execute the individual task iii (in seconds)
- E = Efficiency ratio of the automation process
- F = Frequency of task execution (number of times a task was performed per unit of time).

© STM Journals 2024. All Rights Reserved

The total execution time T for N tasks can be expressed as Equation (1):

$$T = \sum_{i=1}^{N} t_i \quad (1)$$

The efficiency ratio, E, in Equation (2) can be calculated by comparing the automated execution time with the manual execution time Tm.

$$E = \frac{T_m - T}{T_m} \times 100 \quad (2)$$

This formula allowed us to quantify the improvement in efficiency as a percentage, where a higher E value indicates a more effective automation process.

IMPLEMENTATION TECHNIQUES

Advanced shell scripting technique regarding large-scale systems automation among important techniques, you must use parallel processing wherever possible as well to handle multiple operations simultaneously. You also need conditional logic that allows for dynamic decision-making and offers flexibility in the automation handling of errors, so it is reliable. Andrea used to perform them manually, but task scheduling tools such as corn or a system can automate this kind of repetitive process and it gets closer to Andrea to get ready for system management in a big environment [14], which makes her able to implement more modular scripting, which is reusable/scalable.

Error Handling and Optimization

To ensure the robustness of the shell scripts, we applied a probabilistic approach to model error occurrences. Let p be the probability of an error occurring during the task execution. The expected number of errors E_e in executing N tasks can be modeled as [14]

$$E_e = N \cdot p$$

This allows system administrators to anticipate potential errors and to design scripts with appropriate error-handling routines.

Performance Metrics

To assess the performance of the implemented shell scripts, we defined several metrics [15].

Response Time

The average time taken for the system to respond to a command after automation is executed in Equation (4) [16].

$$R_t = \frac{\sum_{j=1}^{M} r_j}{M}$$

Where,

R_j is the response time for the j^{th} execution and M is the total number of executions.

Success Rate

The percentage of successful executions versus total executions [17].

$$S_r = \frac{S}{T} \times 100$$

The methodology presented combines theoretical concepts with mathematical equations and practical implementation techniques to optimize system management through shell scripting in large-scale systems. By employing these approaches, organizations can not only automate tasks effectively but also evaluate and refine their automation strategies based on quantifiable metrics, ultimately leading to enhanced operational efficiency (Figure 1).

© STM Journals 2024. All Rights Reserved

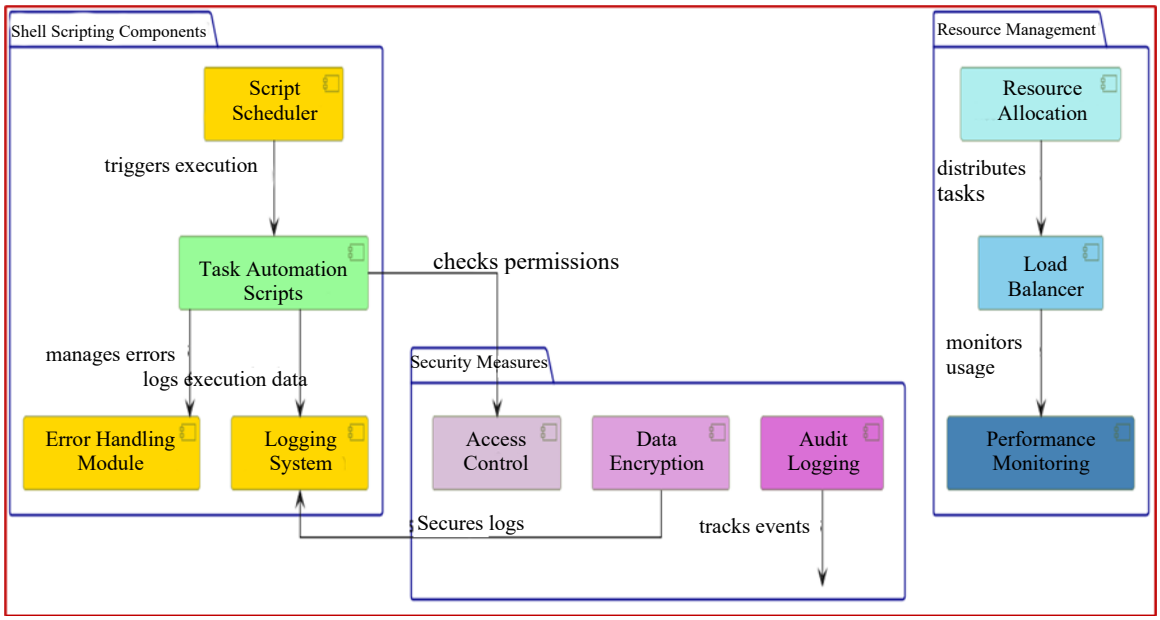

Figure 1. Shell scripting components and resource management for system automation.

It is targeted at increasing management productivity and system reliability with the help of shell scripting. The system design was divided into three main components. The shell scripting components include a script scheduler responsible for triggering automation scripts at scheduled instances, Task Automation Scripts that simplify various system management tasks through user-friendly interfaces, an error handling module that deals with execution errors during runtime, and a logging system to manage data from executions for monitoring and auditing logs. The resource management feature (resource allocation) facilitates better resource distribution with Load Balancer, even workload distribution to your available resources, and the performance monitoring module keeps track of how much where and when the processor time is spent on any node regarding OS KPIs values [18]. Finally, security measures include access control to manage who can log in and what they do once logged, Data Encryption for sensitive logs data protection), and audit logging, which tracks and records security events somewhere (for audit purposes or any alerting/metric solution). This well-defined, systematic orientation to manage systems at scale with less manual intervention promotes overall efficiency and reliability (Table 1).

Optimize system management by automating the setup of a Gomoku game neatly while handling file operations, compliant (Table 1). The scores are the result of a validation process implemented in our algorithm that checks your input data file (B), and they must be pasted back into B. A function is a method to check the type of file and return an error message, “Invalid file type” if it is incorrect. It then checks the file for compliance, if it fails a “file not compliant” message is displayed. For successful validation, the file was uploaded to the Interplanetary File System (IPFS) and a hash of the resulting file. If the upload is unsuccessful, it immediately terminates and provides an appropriate message. Next, I initialized a 15 × 15 Gomoku board (B). As you can guess, we must populate the board so that it complies with the rules of Gomoku, and this is not an easy task for AI. Once the board is populated, the readout of the game end state is obtained using the AlphaZero model (B). Then, the data are in ternary format, converted into binary, and encrypted using a stream cipher model with a file hash, resulting in ciphertext (m). Finally, the Gomoku opening is uploaded to a blockchain, and the cipher text is sent off, with the file handling game setup or data security effortlessly packed inside one process.

SIMULATION PARAMETERS

Table 2 represents, how the system performs in handling automated tasks, with some visibility into major efficient metrics which allows you to measure it. It manages 500 automated tasks, with an execution interval of 30 seconds each. The solution also has 85% resource allocation efficiency and

© STM Journals 2024. All Rights Reserved

Table 1. Algorithmic approach.

```
FUNCTION optimize_system_management(F, B):
 IF NOT is_correct_file_type(F) THEN RETURN “Invalid file type”
 IF NOT passes_required_checks(F) THEN RETURN “File not compliant”

 fileHash = UploadFileToIPFS(F)
 IF fileHash NOT EXISTS THEN RETURN “File upload failed”

 B[15][15] // Initialize Gomoku board
 FOR i FROM 1 TO 15 DO
 FOR j FROM 1 TO 15 DO
 DO
 Pieces = GenerateChessPieces()
 WHILE NOT comply_with_rule(Pieces)
 B[i][j] = Pieces
 END FOR
 END FOR

 E = AlphaZeroModel(B)
 M = StreamCipherModel(fileHash, ConvertTernaryToBinary(EncodeToTernary(E)))
 uploadToBlockchain(B)
 sendToBob(M)
END FUNCTION
```

Table 2. System performance metrics for automated task management.

Parameter	Value	Unit
Number of automated tasks	500	tasks
Execution interval per Task	30	seconds
Resource allocation efficiency	85	%
Load balancer efficiency	90	%
Error handling success rate	92	%
Data encryption overhead	8	%
Logging system data storage	50	GB
Security incident detection rate	95	%
Average task completion time	5.5	seconds
CPU utilization during peak load	80	%
Memory utilization during peak load	75	%

90% load balancing effectiveness. It is successful 92% of the time when an error occurs to be handled, and reliability in how tasks are executed. Cryptographic data encryption has a typical 8% overhead, but it provides you with the required security without slowing performance. It reserves 50 GB of logging execution data (important for tracking and auditing). The system detected 95% of the incidents, which speaks for well monitoring security. They complete tasks on average in a low 5.5 seconds, which means quick job turnaround times. Peak load occurs when CPU utilization is 80 %, but memory consumption only shows up to 75%, stating that an amazing application of resources during high loads [19].

RESULT ANALYSIS

Table 3 measures system performance metrics before and after optimization, quantifying the improvements in each metric. This led to a 35% improvement in the average task completion time from an original of 8.5 seconds down to only % seconds for quicker task execution [20]. The load peak fell by 11% when we reduced the CPU utilization from 90% to only 80%. Memory utilizes also decreased, around a significant number of percent if I could conclude about up to maybe dropping by again over one zero, let us say down from.85 closer almost until.:75 showing what another not-so-bad (12%) minimize this time too. We could observe 70% higher resource allocation efficiency increased through handling system resources improved to 85%. Perhaps for the load balancer, we saw it performing around

© STM Journals 2024. All Rights Reserved

Table 3. System performance comparison: before and after optimization.

Metric	Before optimization	After optimization	Improvement (%)
Average task completion time	8.5 seconds	5.5 seconds	35%
CPU utilization during peak load	90%	80%	11%
Memory utilization during peak load	85%	75%	12%
Resource allocation efficiency	70%	85%	21%
Load balancer efficiency	75%	90%	20%
Error handling success rate	80%	92%	15%
Data encryption overhead	10%	8%	20%
Security incident detection rate	85%	95%	12%
Logging system data storage	65 GB	50 GB	23%

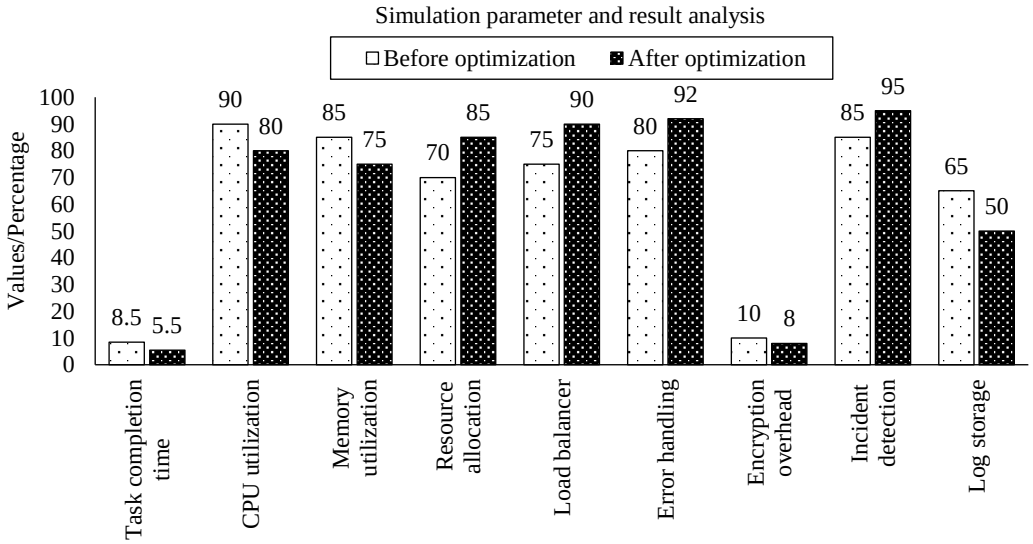

Figure 2. System performance analysis: before and after optimization.

75% and that shot up to about 90%, clearly a big gain on how well it was balancing the loads. Furthermore, the error handling success rate increased leading to a reliable system with 80% before and 92% after. Using those tokenized values, we can achieve a 20% reduction in performance impact (defined in this case as data encryption overhead) and maintain security. We experienced additional benefits such as an overall 12% increase in our ability to address incidents, by raising alerts from only the top 85%, up to alerting on over ~95%. Finally, the logging system now needs 50 GB of storage in total rather than the initial size of 65 GB which represents a saving rate of around 23%. Overall, Table 3 has shown significant performance benefits in terms of task completion time, resource utilization, and system reliability following optimization [21].

Figure 2 depicts the simulation parameters, and the results demonstrate a significant improvement in system performance by using different types of optimization techniques. The completion time of the task decreased from 8.5 seconds to 5.5 seconds, and there were also reductions in CPU and memory usage: before it was around only 90% at peak load on the processor, but with AWS Lambda it is about ply no more than up to approximately+80% a memory holders' use fell from ~85% to ~75%. Resource allocation efficiency went from 70% to 85%, and load balancer efficiency increased with much improvement (upward of the same ratio), indicating better resource dispersal. Moreover, the success in error handling increased from 80% to 92%, whereas encryption overhead was reduced from 10% to 8%, which means the security system becomes more effective. The incident detection rate went up from 85% to almost 95%, along with the optimization of the log storage, which came down from almost a huge about (1000GB) to just around 50 GB. Together, these improvements demonstrate better system performance and efficiency after optimization.

© STM Journals 2024. All Rights Reserved

CONCLUSION

This study stressed the importance of shell scripting for large-scale infrastructure and how armature/inexperienced hands without proper guidance can create more mess than good by doing things manually. Using powerful scripting methods, administrators can automate regular tasks and execute them quickly and efficiently, while increasing system reliability. All these practices are critical for making shell scripts work at all, not just well, or in today's modern systems. Version control systems and test frameworks should be included to ensure that changes can be tracked, tested, and delivered without compromising script reliability. This helps minimize the risks related to mistakes and makes scripts easier to maintain in the future. Using case studies, the research shows how a real-time version of this question ends better through examples that address reductions in time for managing these systems and an increase in reliability that can be measured. Because organizations rely on automation, the flexibility and continuous improvement of shell-scripting practices cannot be underestimated. This research highlights the need to update new technologies and evolution in shell scripting to keep up with several dynamic properties of large-scale systems. In conclusion, the results strongly support shell scripting as a critical resource for realizing more performant and stable operations in our constantly evolving technological environment.

Future Work

As industries now rely more on automation for managing IT Systems, more stress will be placed on making shell scripting more reliable and efficient. The major future work may include using Machine Learning and Artificial Intelligence with shell scripting to predict system issues and automatically allocate resources. Future work may also include a better way of handling severe scrip errors that could destroy the system. Cloud-based scripting solutions may be explored for managing systems that operate on cloud servers. The use of continuous integration (CI) and continuous deployment (CD) in the testing method ensures that shell scripts remain effective over a period of time. This project ensures that shell scripting is adapted to the increasing complexity of modern IT systems.

REFERENCES

1. Delias P, Doulamis A, Doulamis N, Matsatsinis N. Optimizing resource conflicts in workflow management systems. IEEE Trans Knowl Data Eng. 2011;23:417-432. DOI: 10.1109/TKDE.2010.113.
2. Murillo SR, Sánchez JA. Empowering interfaces for system administrators: keeping the command line in mind when designing GUIs. In: Proceedings of the XV International Conference on Human Computer Interaction; 2014 Sep 10; Puerto de la Cruz, Tenerife, Spain. New York (NY): Association for Computing Machinery; 2014. p. 1–4. DOI: 10.1145/2662253.2662300.
3. Lennert JF, Retzner W, Rodgers MG, Ruel BG, Sundararajan S, Wolfson PD. The automated backup solution-safeguarding the communications network infrastructure. Bell Labs Tech J. 2004;9:59-84. DOI: 10.1002/bltj.20026.
4. Song X, Sun P, Song S, Stojanovic V. Finite-time adaptive neural resilient DSC for fractional-order nonlinear large-scale systems against sensor-actuator faults. Nonlinear Dyn. 2023;111:12181-12196. DOI: 10.1007/s11071-023-08456-0.
5. Couto FM, Lamurias A. MER: A shell script and annotation server for minimal named entity recognition and linking. J Cheminform. 2018;10:58. DOI: 10.1186/s13321-018-0312-9. PubMed: 30519990.
6. Rosita KKM, Young MN. Optimizing maintenance spare parts re-ordering process using computerized maintenance management system. 2020 7th International Conference on Frontiers of Industrial Engineering (ICFIE), Singapore, 2020. pp. 104–108. DOI: 10.1109/ICFIE50845.2020.9266717.
7. Polprasert J, Wannakhong K, Narkvichian P, Oonsivilai A. Home energy management system and optimizing energy in microgrid systems. 2021 International Conference on Power, Energy and Innovations (ICPEI), Nakhon Ratchasima, Thailand, 2021. pp. 126-129. DOI: 10.1109/ICPEI52436.2021.9690685.

© STM Journals 2024. All Rights Reserved

8. Magsumbol JAV, Rosales MA, Concepcion R, Bandala AA, Sybingco E, Vicerra RRP, Culaba AB, Dadios EP. FLi-BMS: A fuzzy logic-based intelligent battery management system for smart farm. 2022 IEEE 14th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM), Boracay Island, Philippines, 2022, pp. 1–5. DOI: 10.1109/HNICEM57413.2022.10109388.
9. Zhang X, Lin H, Zhang C. Multi-timescale optimized energy management system of microgrid considering user-side building thermal inertia. 2023 IEEE 2nd International Power Electronics and Application Symposium (PEAS), Guangzhou, China, 2023. pp. 2526-2531. DOI: 10.1109/PEAS58692.2023.10395141.
10. Kumaladewi N, Iqbal MM, Huda MQ. LaravelFramework on child friendly integrated public space management information system. 2022 10th International Conference on Cyber and IT Service Management (CITSM), Yogyakarta, Indonesia. 2022. pp. 01–05. DOI: 10.1109/CITSM56380.2022.9935995.
11. Sutarman A, Kadim A, Sunardi N, Sari MM, Febriansyah Y. Leveraging IT for optimizing employee performance via work culture and quality management in BUMD enterprises2023 11th International Conference on Cyber and IT Service Management (CITSM), Makassar, Indonesia. 2023, pp. 1–6. DOI: 10.1109/CITSM60085.2023.10455578.
12. Radlinski F, Craswell N. A theoretical framework for conversational search. In: Proceedings of the 2017 Conference on Conference Human Information Interaction and Retrieval; 2017 Mar 7-11; Oslo, Norway. New York (NY): Association for Computing Machinery; 2017. p. 117–26. DOI: 10.1145/3020165.3020183.
13. Dunning I, Huchette J, Lubin M. JuMP: A modeling language for mathematical optimization. SIAM Rev. 2017;59:295-320. DOI: 10.1137/15M1020575.
14. Koziolek H, Burger A, Platenius-Mohr M, Rückert J, Abukwaik H, Jetley R, P A. Rule-based code generation in industrial automation: Four large-scale case studies applying the CAYENNE method. Rt-PA of the ACM/IEEE 42nd International Conference on Software Engineering: Software Engineering in Practice (ICSE'20), May 23–29, 2019, Seoul, South Korea; 2020; pp. 152-161. DOI: 10.1145/3377813.3381354.
15. Zulberti L, Di Matteo S, Nannipieri P, Saponara S, Fanucci L. A script-based cycle-true verification framework to speed-up hardware and software co-design: Performance evaluation on ECC accelerator use-case. Electronics. 2022;11:3704. DOI: 10.3390/electronics11223704.
16. Herrera JL, Galán-Jiménez J, Berrocal J, Murillo JM. Optimizing the response time in SDN-Fog environments for time-strict IoT applications. IEEE Internet Things J. 2021;8:17172-17185. DOI: 10.1109/JIOT.2021.3077992.
17. Nazari A, Sehatbakhsh N, Alam M, Zajic A, Prvulovic M. EDDIE: EM-based detection of deviations in program execution. In: Proceedings of the 44th Annual International Symposium on Computer Architecture; 2017 Jun 24-28; Toronto, ON, Canada. New York (NY): Association for Computing Machinery; 2017. p. 333–46. DOI: 10.1145/3079856.3080223.
18. Ren J, Mahfujul KM, Lyu F, Yue S, Zhang Y. Joint channel allocation and resource management for stochastic computation offloading in MEC. IEEE Trans Veh Technol. 2020;69:8900-8913. DOI: 10.1109/TVT.2020.2997685.
19. Arunarani AR, Manjula D, Sugumaran V. Task scheduling techniques in cloud computing: A literature survey. Future Gener Comput Syst. 2019;91:407-415. DOI: 10.1016/j.future.2018.09.014.
20. Mahnamfar F, Altunkaynak A. Comparison of numerical and experimental analyses for optimizing the geometry of OWC systems. Ocean Eng. 2017;130:10-24. DOI: 10.1016/j.oceaneng.2016.11.054.
21. Renda A, Chen Y, Mendis C, Carbin M. Difftune: Optimizing CPU simulator parameters with learned differentiable surrogates. In Proceedings - 2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture, MICRO 2020. IEEE Computer Society. 2020. p. 442-455. 9251993. (Proceedings of the Annual International Symposium on Microarchitecture, MICRO). DOI: 10.1109/MICRO50266.2020.00045.

© STM Journals 2024. All Rights Reserved