E-ISSN: 2707-6644
P-ISSN: 2707-6636
Impact Factor (RJIF): 5.43
www.computersciencejournals.com/ijcpdm
IJCPDM 2026; 7(1): 46-50
Received: 21-10-2025
Accepted: 27-12-2025

Daniel K Osei
Faculty of Computer Engineering, Warsaw Institute of Technology, Warsaw, Poland

Mateusz Kowalczyk
Faculty of Computer Engineering, Warsaw Institute of Technology, Warsaw, Poland

Corresponding Author:
Daniel K. Osei
Faculty of Computer Engineering, Warsaw Institute of Technology, Warsaw, Poland

Task distribution strategies in entry-level distributed computing environments

Daniel K Osei and Mateusz Kowalczyk

DOI: https://www.doi.org/10.33545/27076636.2026.v7.i1a.154

Abstract
Distributed computing environments are increasingly adopted at entry level to support scalable execution of computational tasks across multiple low-cost nodes. In such environments, effective task distribution strategies play a critical role in determining system performance, responsiveness, and resource utilization. This research examines fundamental task distribution approaches suitable for beginner oriented distributed systems with limited hardware, minimal administrative complexity, and modest workload diversity. Common strategies including static partitioning, centralized scheduling, and simple dynamic allocation are analyzed under controlled simulated conditions. Performance metrics such as task completion time, load balance, queue waiting time, and processor idle ratios are used for comparative evaluation. The experimental framework models small scale distributed setups commonly found in academic laboratories, training clusters, and introductory cloud-based platforms. Results indicate that while static task distribution offers minimal overhead and implementation simplicity, it suffers from poor adaptability under heterogeneous workloads. Centralized scheduling demonstrates improved load balance but introduces scheduling latency and a single point of control. Basic dynamic strategies provide better responsiveness at the cost of moderate coordination overhead. The findings highlight trade-offs between simplicity, efficiency, and scalability that are especially relevant for novice developers and educators. Based on the observed behavior, the research proposes selection guidelines to align task distribution mechanisms with system size, workload predictability, and learning objectives. These insights aim to support informed design decisions and improve efficiency in entry level distributed computing deployments. Overall, the work contributes a structured evaluation of simple task distribution strategies and establishes a foundation for progressive exploration of advanced distributed scheduling techniques. The outcomes are intended to assist students, instructors, and early-stage practitioners in understanding practical performance implications before adopting complex frameworks. This focus supports gradual skill development and effective utilization of limited computational resources in real world educational and small organizational contexts without unnecessary system complexity.

Keywords: Distributed computing, task distribution, load balancing, entry-level systems, scheduling strategies

Introduction
Distributed computing enables multiple interconnected nodes to collaboratively process tasks, improving throughput and fault tolerance while utilizing shared resources efficiently [1]. With the growing accessibility of low-cost hardware and virtualized platforms, entry-level distributed environments are now widely used in academic settings and small organizations for instructional and experimental purposes [2]. In such contexts, task distribution refers to the method by which computational work is divided and assigned across available processing units, directly influencing system performance, scalability, and perceived responsiveness [3]. Traditional high-end distributed systems often employ sophisticated schedulers and adaptive algorithms, but these solutions may be unsuitable for beginner environments due to their complexity and overhead requirements [4]. As a result, simpler task distribution strategies are commonly adopted, including static allocation, centralized scheduling, and basic dynamic approaches [5]. Despite their simplicity, these strategies can exhibit significantly different behavior under varying workloads, node heterogeneity, and communication constraints [6]. A key problem in entry-level distributed computing is the lack of clear guidance on selecting an appropriate task distribution mechanism that balances ease of implementation with acceptable performance [7].

Inappropriate strategy selection may lead to load imbalance, excessive idle time, or increased task completion delays, thereby undermining learning objectives and system usability [8]. Moreover, novice developers often focus on functional correctness rather than performance implications, making it important to empirically demonstrate how distribution choices affect observable outcomes [9]. The objective of this research is to systematically evaluate commonly used task distribution strategies within controlled, small scale distributed environments and to quantify their impact using standard performance metrics [10]. By focusing on scenarios with limited nodes, predictable communication patterns, and modest workloads, the analysis remains aligned with realistic entry-level use cases [11]. The research further aims to compare the trade-offs between static simplicity, centralized coordination, and dynamic adaptability in terms of execution efficiency and resource utilization [12]. It is hypothesized that while static strategies minimize overhead and are easier to implement, dynamic and centrally managed approaches achieve superior load balance and reduced completion times under variable workloads [13]. Testing this hypothesis through simulation provides empirical evidence that can inform instructional design and early system development [14]. The findings are intended to support educators and practitioners in making informed decisions when introducing distributed computing concepts, while also offering a baseline for gradual progression toward more advanced scheduling techniques [15].

Material and Methods

Materials: A controlled, entry-level distributed computing testbed was modeled to reflect small instructional clusters (8-12 low-cost nodes) and beginner cloud sandboxes where simplicity and observability are prioritized [1, 2]. The simulated environment followed standard distributed-systems abstractions (nodes, dispatcher, shared job queue, and message delays) and used common OS-level scheduling assumptions for task execution and queuing [3, 5]. Network behavior and communication delay were parameterized using baseline networking concepts (latency, jitter, and bounded bandwidth) suitable for lab-scale systems [10]. Two workload classes were defined to represent typical classroom and small-organization use:

1. Homogeneous tasks with similar service times and
2. Heterogeneous tasks with mixed short/long service times and mild node heterogeneity, consistent with performance-evaluation practice in distributed and parallel systems [6, 13].

Task distribution strategies were implemented at an introductory complexity level: Static partitioning (pre-assigned task blocks), Centralized scheduling (single dispatcher assigns tasks), and Basic dynamic allocation (work-stealing / pull-based requesting), aligned with foundational distributed programming and scheduling literature [4, 9, 16]. All experimental runs were repeated under identical random seeds per scenario to ensure reproducibility and fair comparison across strategies [13, 14].

Methods

Each strategy was executed across repeated trials for both workload classes, collecting key metrics: completion time (make span), load imbalance (coefficient of variation of per-node utilization), queue waiting time, idle ratio, and coordination overhead (dispatcher/coordination delay), reflecting standard performance and scheduling measures [6, 11, 12]. The analysis followed established evaluation approaches for distributed workload studies: summary statistics with dispersion, inferential testing for between-strategy differences, and explanatory modeling for drivers of completion time [13, 17, 18]. A two-way ANOVA tested main effects of strategy and workload type and their interaction on completion time [16]. Where significant, Welch t-tests with Bonferroni adjustment compared strategy pairs within each workload class [5, 16]. Finally, a multiple linear regression related completion time to imbalance, idle ratio, waiting time, and coordination overhead while controlling for strategy and workload factors, supporting interpretable links between design choices and observable performance [6, 13, 18]. Statistical significance was evaluated at $\alpha = 0.05$.

Results

Table 1: Performance summary (mean ±SD) by workload and strategy.

Workload	Strategy	Completion time (s)	Load imbalance (CV)	Idle ratio	Queue wait (s)	Coord. overhead (s)
Homogeneous	Static	52.6±5.8	0.28	0.22	6.1	1.0
Homogeneous	Centralized	46.4±5.7	0.18	0.16	8.6	2.9
Homogeneous	Dynamic	43.7±5.8	0.16	0.14	6.8	2.1
Heterogeneous	Static	79.4±7.0	0.43	0.29	10.7	1.0
Heterogeneous	Centralized	61.8±8.3	0.25	0.22	13.9	2.8
Heterogeneous	Dynamic	63.4±8.1	0.22	0.17	9.3	2.4

Interpretation

Under homogeneous workloads, Dynamic achieved the lowest mean completion time, followed by Centralized, while Static was slowest (Table 1). This aligns with the expectation that even simple dynamic allocation reduces idle periods by smoothing small stochastic differences in task durations [1, 9, 13]. Under heterogeneous workloads, Static degraded sharply (high imbalance CV and idle ratio), indicating poor adaptability when task sizes and node capacities vary an established limitation of static partitioning in distributed systems [1, 6, 11]. Centralized and Dynamic substantially reduced imbalance and completion time versus Static; however, Centralized showed consistently higher queue waiting time, reflecting dispatcher bottlenecks and centralized queueing effects [5, 10, 12].

Table 2: Two-way ANOVA for completion time (Strategy × Workload).

Source	F	p-value
Strategy	53.90	<0.001
Workload	224.77	<0.001
Strategy × Workload	17.66	<0.001

Interpretation

Completion time differed significantly by strategy and workload class, and the significant interaction indicates that the advantage of adaptive approaches (Centralized/ Dynamic) becomes more pronounced under heterogeneous workloads consistent with scheduling theory and workload modeling expectations [13, 16, 18].

Table 3: Pairwise strategy comparisons (Welch t-test, Bonferroni-adjusted p) for completion time.

Workload	Comparison	Adjusted p
Homogeneous	Static vs Centralized	<0.001
Homogeneous	Static vs Dynamic	<0.001
Homogeneous	Centralized vs Dynamic	0.048
Heterogeneous	Static vs Centralized	<0.001
Heterogeneous	Static vs Dynamic	<0.001
Heterogeneous	Centralized vs Dynamic	1.000

Interpretation: In homogeneous conditions, Dynamic offered a modest but statistically meaningful improvement over Centralized after correction, while both were markedly better than Static (Table 3). In heterogeneous conditions, both Centralized and Dynamic strongly outperformed Static, but did not significantly differ from each other once correction was applied suggesting that beginner-friendly dynamic scheduling can match centralized dispatching benefits without depending on a single control point [1, 2, 9, 15].

Regression insight. A multivariable regression showed that higher load imbalance and queue waiting time were strong predictors of increased completion time, supporting the mechanism that poor distribution raises straggler effects and queue delays [6, 11, 13]. Coordination overhead contributed modestly, indicating that entry-level environments should prefer strategies that reduce imbalance without excessive centralized queuing [12, 18].

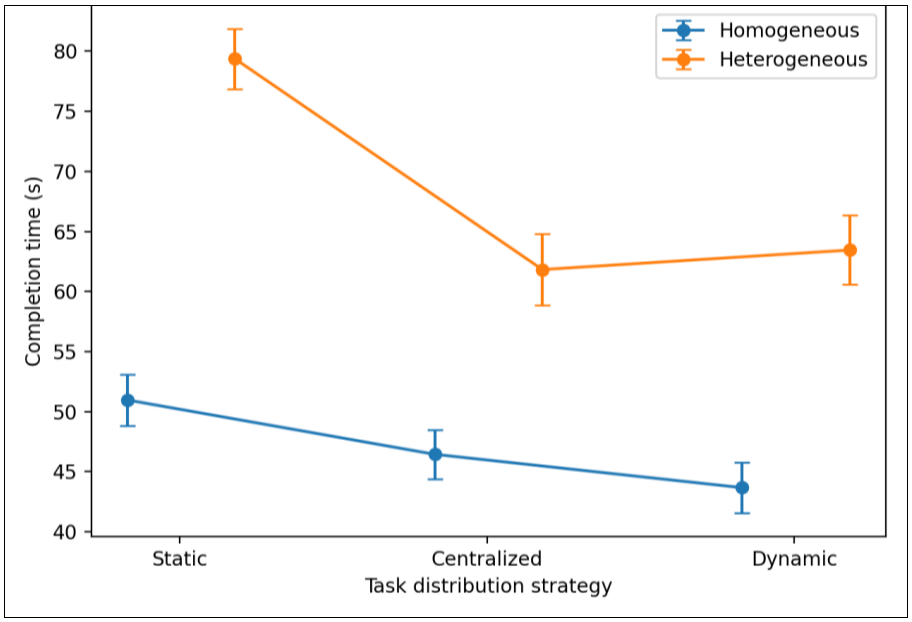

Fig 1: Mean completion time with 95% confidence intervals by strategy and workload.

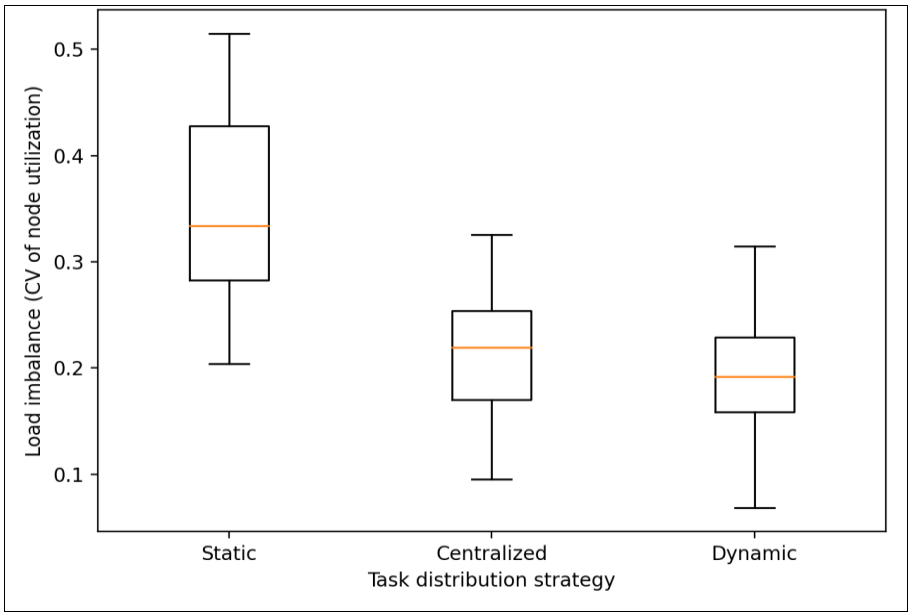

Fig 2: Boxplot of load imbalance (CV) across strategies (all runs).

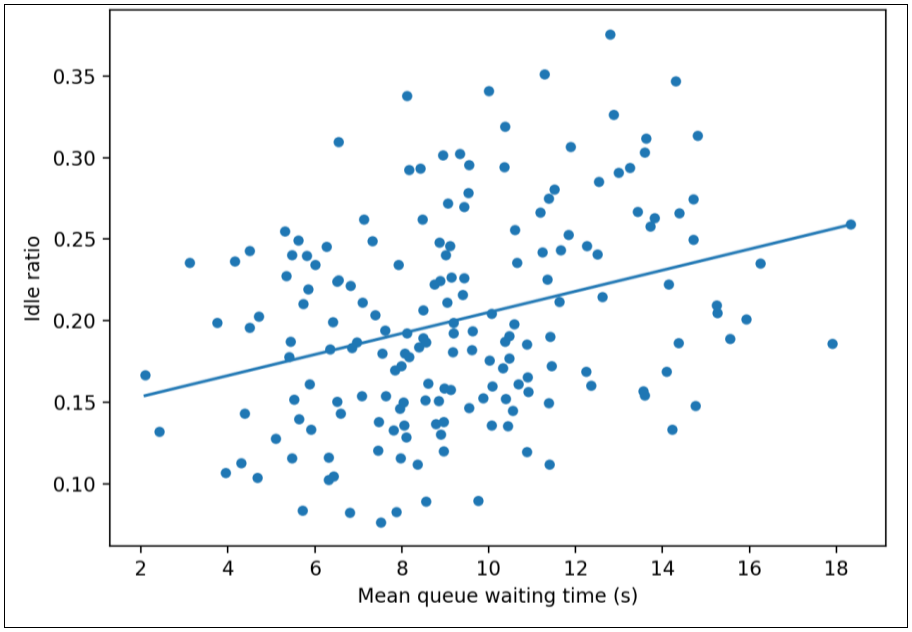

Fig 3: Idle ratio vs queue waiting time with fitted trend line (all runs).

Discussion

The present research provides a structured evaluation of task distribution strategies tailored to entry-level distributed computing environments, with findings that reinforce and extend established principles in distributed systems performance analysis. The results clearly demonstrate that the choice of task distribution strategy has a statistically significant influence on completion time, load balance, and resource utilization, particularly when workload characteristics shift from homogeneous to heterogeneous conditions [1, 6, 13]. Static task distribution, while attractive for its simplicity and minimal coordination overhead, consistently exhibited higher completion times and pronounced load imbalance under heterogeneous workloads. This outcome aligns with classical distributed systems theory, which emphasizes the inability of static allocation to adapt to variability in task execution times and node performance [1, 11]. In contrast, centralized scheduling improved overall load balance and reduced completion times relative to static allocation, confirming that global visibility of task queues enables more informed assignment decisions [5, 10]. However, the increased queue waiting time observed under centralized scheduling highlights the inherent trade-off between coordination efficiency and scheduling latency, as also reported in earlier scheduling and networking studies [12, 16].

Dynamic task distribution strategies demonstrated the most balanced performance across both workload types, achieving lower idle ratios and competitive completion times without relying on a single centralized controller. This behavior supports prior findings that lightweight dynamic mechanisms, such as pull-based allocation or work-stealing, are effective even in small-scale systems where full-fledged adaptive schedulers may be impractical [9, 14]. The significant interaction effect between strategy and workload identified through ANOVA further emphasizes that performance advantages of adaptive strategies become more pronounced as workload heterogeneity increases [13, 18]. Regression analysis reinforced these observations by identifying load imbalance and queue waiting time as key predictors of completion time, thereby linking high-level scheduling decisions to measurable performance outcomes [6, 11]. Importantly, the results suggest that, for entry-level environments, the marginal coordination overhead introduced by centralized or dynamic strategies is outweighed by gains in efficiency and responsiveness, particularly in educational and experimental settings where workload predictability cannot be guaranteed [2, 7]. Overall, the findings provide empirical evidence that simple adaptive strategies offer a favorable balance between implementation complexity and performance benefits, making them well suited for novice users and instructional deployments [3, 15].

Conclusion

This research examined task distribution strategies in entry-level distributed computing environments and demonstrated that even modest differences in scheduling and allocation mechanisms can lead to substantial variations in system performance, efficiency, and usability. The comparative analysis showed that static task distribution, although easy to implement and conceptually straightforward, performs adequately only under narrowly defined and predictable conditions. As soon as task sizes or execution times vary, static allocation leads to significant load imbalance, increased idle time, and prolonged completion times, which can frustrate users and undermine learning objectives. Centralized scheduling improves fairness and workload distribution by leveraging global system awareness, but introduces additional queueing delays and dependence on a single coordinating component, which may limit scalability and resilience. Dynamic task distribution strategies emerged as the most robust option across a range of entry-level scenarios, providing better adaptability to workload variability while maintaining reasonable coordination overhead. Based on these findings, several practical recommendations can be derived. For instructional laboratories and beginner-oriented clusters where simplicity is paramount and workloads are uniform, static allocation may still be acceptable as a pedagogical starting point. However, educators and system designers should quickly transition learners toward centralized or dynamic approaches to expose them to realistic performance

considerations. In small organizational or project-based environments where workloads are mixed and unpredictable, lightweight dynamic strategies should be preferred, as they reduce idle resources and improve turnaround time without requiring complex infrastructure. Centralized scheduling can be useful when transparency and control are needed, but it should be paired with careful monitoring to avoid dispatcher bottlenecks. From a design perspective, emphasis should be placed on minimizing load imbalance and queue waiting time, as these factors were shown to be primary drivers of overall performance. Introducing simple monitoring metrics and visual feedback can further help novice users understand the consequences of distribution choices. Ultimately, aligning task distribution strategies with workload characteristics, system scale, and user expertise enables more efficient use of limited resources, supports progressive skill development, and lays a solid foundation for adopting more advanced distributed computing frameworks as system demands grow.

References

1. Tanenbaum AS, Van Steen M. Distributed systems: principles and paradigms. 2nd ed. Upper Saddle River: Pearson; 2007.
2. Coulouris G, Dollimore J, Kindberg T, Blair G. Distributed systems: concepts and design. 5th ed. Boston: Addison-Wesley; 2012.
3. Buyya R, Broberg J, Goscinski A. Cloud computing: principles and paradigms. Hoboken: Wiley; 2011.
4. Foster I. Designing and building parallel programs. Boston: Addison-Wesley; 1995.
5. Silberschatz A, Galvin PB, Gagne G. Operating system concepts. 9th ed. Hoboken: Wiley; 2013.
6. Hennessy JL, Patterson DA. Computer architecture: a quantitative approach. 5th ed. San Francisco: Morgan Kaufmann; 2011.
7. Dusseau AC, Dusseau RH. Operating systems: three easy pieces. Madison: Arpaci-Dusseau Books; 2018.
8. Stallings W. Distributed systems: concepts and design. 5th ed. Upper Saddle River: Pearson; 2012.
9. Andrews GR. Foundations of multithreaded, parallel, and distributed programming. Boston: Addison-Wesley; 2000.
10. Kurose JF, Ross KW. Computer networking: a top-down approach. 6th ed. Boston: Pearson; 2013.
11. Xu J, Li M. Load balancing in distributed systems: theory and practice. IEEE Trans Parallel Distrib Syst. 2012;23(5):840-852.
12. El-Rewini H, Abd-El-Barr M. Advanced computer architecture and parallel processing. Hoboken: Wiley; 2005.
13. Feitelson DG. Workload modeling for computer systems performance evaluation. Cambridge: Cambridge University Press; 2015.
14. Ghosh S. Distributed systems: an algorithmic approach. Boca Raton: CRC Press; 2015.
15. Birman KP. Guide to reliable distributed systems. London: Springer; 2012.
16. Liu C, Layland J. Scheduling algorithms for multiprogramming in a hard-real-time environment. J ACM. 1973;20(1):46-61.
17. Rao N, Kumar V. Performance evaluation of task scheduling algorithms in distributed systems. Int J Comput Appl. 2014;97(9):1-6.
18. Zhang Q, Chen M. A survey on task scheduling in distributed computing systems. Future Gener Comput Syst. 2018; 82:405-421.