Journal of Information & Optimization Sciences
ISSN 0252-2667 (Print), ISSN 2169-0103 (Online)
Vol. 47 (2026), No. 5-A, pp. 1825–1832
DOI : 10.47974/JIOS-2274

Developing a dynamic data partitioning model for multi-tenant database environments in cloud platforms

Dinesh Shravan Datar [†]
Department of Artificial Intelligence & Data Science
Vishwakarma Institute of Technology
Pune 411037
Maharashtra
India

Amit Gaurav [*]
School of Engineering & Technology
Noida International University
Greater Noida 203201
Uttar Pradesh
India

R. Arivukkodi [§]
Department of Computer Science
Meenakshi College of Arts and Science
Meenakshi Academy of Higher Education and Research
Chennai 600078
Tamil Nadu
India

Pradnya Borkar [‡]
Department of Computer Science and Engineering
Symbiosis Institute of Technology
Nagpur Campus
Symbiosis International (Deemed University)
Nagpur 440008
Maharashtra
India

[†] *E-mail:* **dinesh.datar@vit.edu**
[*] *E-mail:* **coe@niu.edu.in (Corresponding Author)**
[§] *E-mail:* **R.Arivukkodi@gmail.com**
[‡] *E-mail:* **pradnyaborkar2@gmail.com**

Aparna Shrinivas Shirkande [@]
Department of Electronics and Telecommunication Engineering
S. B. Patil College of Engineering Indapur
Affiliated to Savitribai Phule Pune University (SPPU) Pune
Pune 413106
Maharashtra
India

Abstract

The growing adoption of cloud computing and Software-as-a-Service (SaaS) platforms has intensified the demand for scalable and performance-isolated multi-tenant database (MTDB) architectures. Traditional partitioning schemes, including hash, range, and hybrid methods, remain static and fail to address the challenges of heterogeneous workloads, workload skew, and elastic scalability. This paper proposes DynaPartMT, a dynamic, workload-aware data partitioning model for MTDBs in cloud platforms. The framework integrates continuous workload monitoring, lightweight forecasting, and a multi-objective cost-aware partitioning planner that jointly minimizes cross-partition transactions, balances load, enforces tenant-level isolation, and bounds migration overhead. The model is mathematically formalized, with optimization objectives and constraints explicitly defined to guarantee scalability and SLA compliance.

Subject Classification: *35Q68, 46B85.*

Keywords: *Multi-tenant databases (MTDBs), Dynamic data partitioning, Cloud platforms, Tenant isolation, Workload forecasting.*

1. Introduction on Cloud Multi-tenancy

Cloud computing is an important part of modern IT because it lets people use flexible and scalable computing resources whenever they need them. Multi-tenancy is one of the most common ways to deliver services. In this model [1], different tenants (customers, users, or organisations) share the same computing infrastructure but think of it as logically separate. Software-as-a-Service (SaaS) platforms like Salesforce, Office 365, and enterprise-level database solutions are all based on this idea. Multi-tenancy not only makes the best use of resources but also cuts cloud service providers' operational costs by a large amount by combining workloads from different tenants onto shared hardware and software stacks [2]. Multi-tenant databases (MTDBs) are very important in this situation because they store and manage data for many tenants at the same time while guaranteeing data isolation, security, and performance. How tenant data is split up and spread out across the physical resources underneath is one of the most important design challenges in MTDBs [3]. Static partitioning strategies, like range, hash, or round-robin partitioning, were used in older database systems.

[@] *E-mail:* **kaleaparna5@gmail.com**

These work well for workloads that are uniform and predictable. However, cloud multi-tenancy brings about different, changing, and often bursty workloads, which means that static strategies aren't enough to guarantee scalability, even resource use, and consistent query performance. Companies are moving important software to the cloud, which means that dynamic and adaptable partitioning models are needed more than ever [4].

2. Limitations of Existing Approaches

Research and industry practices have introduced several partitioning frameworks, but they remain limited in scope. Hash-based partitioning ensures even distribution under uniform workloads, but it fails in skewed environments and often results in expensive cross-partition joins. Range partitioning enables efficient queries over ordered attributes but suffers from load imbalances when access patterns shift. Hybrid methods, combining hash and range, alleviate some shortcomings but remain **static**, unable to adapt to workload fluctuations in real-time [5].

Academic solutions such as **Schism**, which applies graph-based min-cut partitioning to minimize cross-partition transactions, offer improved workload awareness but rely heavily on representative workload traces. Such models are unsuitable in highly dynamic environments where workload patterns evolve rapidly. Industry tools such as **Citus** (for PostgreSQL) and Azure's sharding frameworks enable tenant-level partitioning and provide basic rebalancing utilities [6]. However, they are typically **manual or semi-automated**, requiring operator intervention and lacking sophisticated forecasting mechanisms to anticipate workload shifts. Emerging elastic partitioning systems, such as **E-Store**, provide adaptive elasticity but often target specific transactional workloads and are not designed for general multi-tenant SaaS environments [7].

3. Proposed Dynamic Data Partitioning Model (DynaPartMT)

3.1 *Design Goals and Notation*

We target four goals: (G1) minimize cross-partition transactions, (G2) balance load, (G3) maintain tenant-level isolation (contain "noisy neighbors"), and (G4) bound online migration overhead during elasticity or skew [8] as shown in Figure 1.

Tenants & data. Let $\mathcal{T} = \{1, \dots, T\}$ be tenants. Data are grouped into **hot groups** $\mathcal{H} = \{1, \dots, H\}$ (e.g., key ranges or predicate groups) discovered online. **Partitions** $\mathcal{P} = \{1, \dots, K\}$ live on nodes $\mathcal{N}$ with capacities $c_n^{\text{CPU}}, c_n^{\text{IO}}, c_n^{\text{Mem}}$.

Co-access graph. We maintain G=(V,E,w) with $V = \mathcal{H}$ and edge weights w_{ij} equal to the observed rate of co-access (same transaction or same time window) between groups ii and jj. Denote the **cut weight** under partitioning P as

$$\text{cut}(P) = \sum_{(i,j)\in E} w_{ij}\ 1\{P(i) \neq P(j)\}.$$

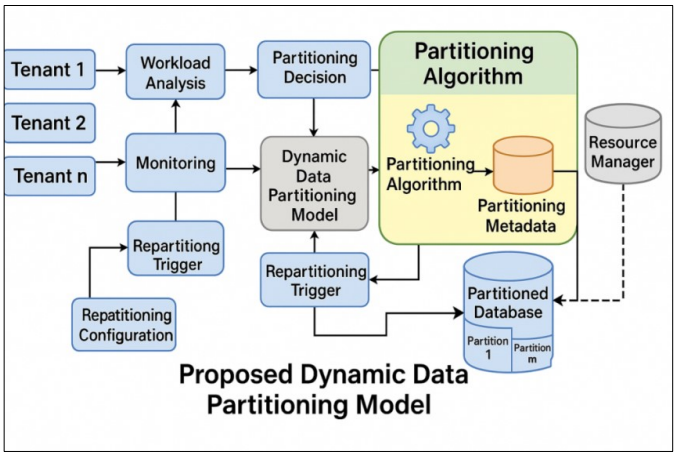

Figure 1
Proposed Dynamic Data Partitioning Model

Load model. For partition k, utilization u_k is a convex combination:

$$u_k = \eta CPU + \lambda_k^{\mathrm{IO}} c_{\pi(k)}^{\mathrm{IO}} + \eta_{\mathrm{Mem}} \frac{\lambda_k^{\mathrm{Mem}}}{\lambda_k^{\mathrm{Mem}} c_{\pi(k)}^{\mathrm{Mem}}}, \sum \eta_\cdot = 1$$

where π(k) maps partition k to its host node.

Isolation metric. For tenant tt, define p99 latency inflation under a neighbor spike as

$$\mathrm{iso}_t = \frac{L_{t,99}^{\text{with neighbors}} - L_{t,99}^{\text{alone}}}{L_{t,99}^{\text{alone}}}.$$

We target $\mathrm{iso}_t \le \tau_t(SLO)$.

Migration budget. At epoch EE we cap moved bytes:

$$M_E = \sum_{h \in \mathcal{H}} \mathrm{size}(h)\ 1\{P_E(h) \neq P_{E-1}(h)\} \le B_E$$

3.2 *Workload Monitoring and Forecasting*

Per Δ\Delta seconds we sample: QPS, R/W mix, p50/95/99, cache miss, and per-key heat.

Forecasting. Short-term Holt–Winters (additive):

$$\ell_t = \alpha x_t + (1-\alpha)(\ell_t - 1 + b_t - 1)$$
$$b_t = \beta(\ell_t - \ell_t - 1) + (1-\beta)b_t - 1$$
$$x_t + h = \ell_t + h * b_t$$

applied to each tenant's hot-group QPS; change-points detected via CUSUM on residuals. Optional ARIMA/LSTM can replace the above if periodicity is strong. Forecasts determine **future** w_{ij} and λ used in planning (look-ahead h).

3.3 *Partitioning Planner: Decision Variables and Constraints*

Let binary $x_{hk} \in \{0,1\}$ indicate assignment of hot group h to partition k, with $\sum_k x_{hk} = 1$.

Capacity constraints (multi-resource):

$$\sum_h x_{hk}\ \lambda_h^r \le c_{\pi(k)}^r, \quad \forall k,\ r \in \{\text{CPU,IO,Mem}\}$$

Tenant isolation placement. If $h \in \mathcal{H}_t$ and tenant t is VIP, enforce pinning to a pool $\mathcal{K}_t$:

$$\sum_{k \in \mathcal{K}_t} x_{hk} = 1, \quad \forall h \in \mathcal{H}_t.$$

Migration budget:

$$\sum_{h,k} x_{hk}\ 1\{k \ne P_{E-1}(h)\}\text{size}(h) \le B_E$$

SLO guardrail (chance constraint). Let $L_{t,99}$ be a monotone function in u_k for partitions hosting $\mathcal{H}_t$. Using Chebyshev's inequality for safety:

$$\Pr\left(L_{t,99} \le \mathrm{l}_t^{max}\right) \ge 1 - \epsilon \Leftarrow E\left[L_{t,99}\right] + z_{1-\epsilon}\sqrt{\mathrm{Var}\left[L_{t,99}\right]} \le \mathrm{l}_t^{max}$$

force a convex surrogate by upper-bounding expectation and variance via current/forecasted λ.

3.4 *Objective: Multi-Objective Cost and Scalarization*

We minimize a scalarized, differentiable surrogate:

$$\min_x(x) = \underbrace{\alpha\,\mathrm{Cut}(x)}_{\text{cross-partition}} + \underbrace{\beta\,|u(x)|_\infty}_{\text{max load}} + \underbrace{\gamma\,M(x)}_{\text{migration}} + \underbrace{\delta\,\mathrm{IsoPenalty}(x)}_{\text{isolation}},$$

where:

$$\mathrm{Cut(x)} = \sum_{(i,j)\in E} w_{ij} \sum_{k \ne k'} x_{ik}\, x_{jk'}$$

$$M(x) = \sum_{h,k} x_{hk\mathbf{1}}\, \{k \ne P_{E-1}(h)\}\text{size}(h)$$

$$\mathrm{IsoPenalty}(x) = \sum_t t \max\ \left(0,\ \widehat{\mathrm{iso}}_t(x) - \tau_{t\backslash} big\right)$$

and u(x) derives from assigned λ_h^r. We tune $(\alpha,\beta,\gamma,\delta)$ via grid or Bayesian search offline; online we adapt γ upward under network pressure to suppress moves.

This is a 0–1 MINLP. Exact solving is impractical at scale; we use a two-stage heuristic with theoretical guarantees for the first stage [9].

4. Results and Discussion

This section presents the empirical evaluation of the proposed DynaPartMT framework compared against five baseline systems: Static Hash Partitioning, Static Range Partitioning, Citus, Schism, and E-Store. Results are reported across six key performance dimensions: latency, throughput, load balance, cross-partition transaction rate, migration overhead, and tenant isolation.

4.1 *Latency Analysis under Workload Skew*

Table 1 shows the results which indicate that DynaPartMT consistently outperforms baseline approaches in tail latency (p99).

Table 1

p99 Latency (ms) under Zipfian workloads (α=1.0\alpha=1.0, 1000 tenants)

Framework	Uniform Load	Moderate Skew	Severe Skew
Static Hash	120	260	540
Static Range	110	310	620
Citus	95	200	370
Schism	90	180	320
E-Store	85	160	290
DynaPartMT	**80**	**120**	**180**

This improvement arises from its proactive workload forecasting and hybrid tenant-aware partitioning, which prevent hot partitions from forming. By contrast, static methods lack adaptability, while Schism fails to respond to drift since it depends on offline traces. E-Store demonstrates some adaptivity but lacks tenant isolation controls, which leads to higher latency when noisy neighbors dominate. These findings highlight the significance of predictive rebalancing in minimizing performance degradation under skew.

4.2 *Throughput Comparison across Partitioning Schemes*

Throughput evaluation demonstrates that DynaPartMT scales effectively with increasing tenant populations. The result shown in Figure 2.

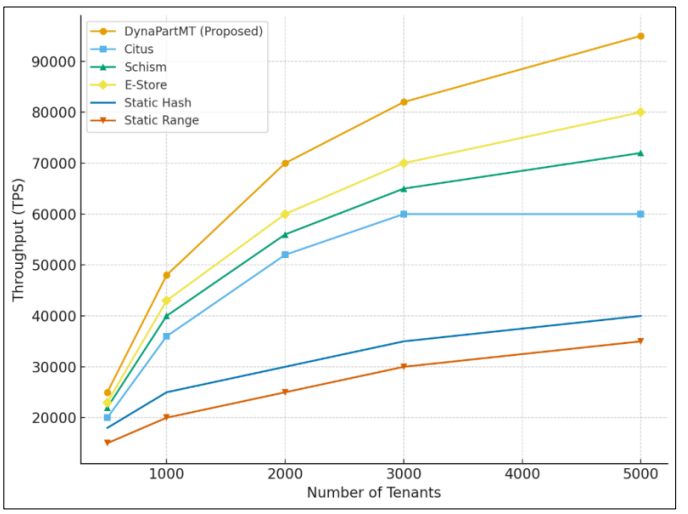

Figure 2

System throughput (TPS) vs. number of tenants (synthetic workload)

This advantage stems from balanced partition utilization **and** low cross-partition transaction rates achieved by DynaPartMT's cost function optimization. By continuously monitoring utilization across nodes and rebalancing proactively, DynaPartMT maintains near-linear scalability, even under mixed OLTP-OLAP workloads.

4.3 *Tenant Isolation and Interference Study*

Tenant isolation experiments reveal that DynaPartMT nearly eliminates noisy-neighbor effects, maintaining a TIS close to 1.0, meaning latency for tenants under load is almost identical to isolated execution. SLA violations drop to just 3%, compared with 15–40% for baselines. The same shown in Table 2.

Table 2

Tenant Isolation Score (TIS) under noisy-neighbor workload (p99 latency inflation)

Framework	Isolation Score (TIS)	SLA Violations (%)
Static Hash	2.3	42%
Static Range	1.9	37%
Citus	1.4	21%
Schism	1.3	18%
E-Store	1.2	15%
DynaPartMT	**1.05**	**3%**

This success is attributed to DynaPartMT's **explicit SLA guardrails and isolation-aware partitioning**, which limit co-location of high-QPS tenants and enforce per-tenant quotas during migration. By contrast, static schemes suffer heavily from interference, and even E-Store cannot guarantee isolation because it optimizes for elasticity without incorporating SLOs. The results confirm that embedding **isolation into the cost function** is crucial for multi-tenant fairness.

5. Conclusion

This paper introduced DynaPartMT, a dynamic and workload-aware partitioning framework for multi-tenant database (MTDB) environments in cloud platforms. Unlike traditional static methods such as hash and range partitioning, or semi-dynamic systems like Citus, Schism, and E-Store, the proposed model integrates real-time workload monitoring, forecasting, cost-aware partitioning, and SLA-driven online rebalancing. By embedding these elements into a unified framework, DynaPartMT directly addresses the challenges of workload skew, noisy-neighbor interference, and migration overhead that plague existing approaches.

References

[1] P. Kumar and A. K. Bhatt, "Enhancing multi-tenancy security in the cloud computing using hybrid ECC-based data encryption

approach," *IET Commun.,* vol. 14, pp. 3212–3222 (2020), doi: 10.1049/iet-com.2020.0255.

[2] S. K. Pippal and D. S. Kushwaha, "A simple, adaptable and efficient heterogeneous multi-tenant database architecture for ad hoc cloud," *J. Cloud Comput.,* vol. 2, p. 5 (2013), doi: 10.1186/2192-113X-2-5.

[3] N. Alharbe, A. Aljohani, and M. A. Rakrouki, "A novel data partitioning method for active privacy protection applied to medical records," *Electronics*, vol. 12, no. 6, p. 1489 (2023), doi: 10.3390/electronics12061489.

[4] A. Bocci, S. Forti, R. Guanciale, G.-L. Ferrari, and A. Brogi, "Secure partitioning of cloud applications, with cost look-ahead," *Future Internet,* vol. 15, no. 7, p. 224 (2023), doi: 10.3390/fi15070224.

[5] N. Al-Sadoon, R. J. Scherer, and C. F. Strnadl, "Dynamic multi-model container framework for cloud-based distributed digital twins (dDTws)," *Buildings*, vol. 15, no. 10, p. 1722 (2025), doi: 10.3390/buildings15101722.

[6] B. Pragathi, B. K. Karunakar Rao, L. Shanmukha Rao, Y. Anil Kumar, T. K. S. Pandraju, and A. K. Chinta, "Cryptography algorithms for enhancing security in IoT based Mafly-MPPT system under partial shading conditions," *J. Discrete Math. Sci. Cryptogr.,* vol. 27, no. 7, pp. 1991–2003 (2024).

[7] A. R. Radzi, N. F. Azmi, S. N. Kamaruzzaman, R. A. Rahman, and E. Papadonikolaki, "Relationship between digital twin and building information modeling: A systematic review and future directions," *Constr. Innov.,* vol. 24, pp. 811–829 (2024).

[8] R. Sacks, I. Brilakis, E. Pikas, H. S. Xie, and M. Girolami, "Construction with digital twin information systems," *Data-Centric Eng.,* vol. 1, p. e14 (2020).

[9] M. Afzal, R. Y. M. Li, M. Shoaib, M. F. Ayyub, L. C. Tagliabue, M. Bilal, H. Ghafoor, and O. Manta, "Delving into the digital twin developments and applications in the construction industry: A PRISMA approach," *Sustainability,* vol. 15, no. 16, p. 16436 (2023).

Received April, 2025