

TỐI ƯU HÓA HIỆU NĂNG HỆ THỐNG TRA CỨU KẾT QUẢ TUYỂN SINH BẰNG KIẾN TRÚC SNAPSHOT IN-MEMORY BẤT BIẾN: MÔ HÌNH HÀNG ĐỢI M/M/C, AMAT VÀ BẢNG CHỨNG THỰC NGHIỆM

OPTIMIZING ADMISSIONS-RESULT LOOKUP PERFORMANCE USING IMMUTABLE IN-MEMORY SNAPSHOTS: AN M/M/C QUEUEING, AMAT-BASED, AND EXPERIMENT-DRIVEN EVALUATION

Nguyễn Thanh Huy*, Đặng Huy Minh*

*Trường Đại học Khoa học Xã hội và Nhân văn, Đại học Quốc gia TP. Hồ Chí Minh

Received: 05/02/2026; Accepted: 12/02/2026; Published: 26/02/2026

Tóm tắt: Hệ thống tra cứu kết quả tuyển sinh tại các trường đại học thường chịu tải tăng đột biến vào thời điểm công bố, trong khi dữ liệu lại mang tính bất biến và chỉ phục vụ truy vấn đọc. Nghiên cứu này xây dựng một khung phân tích định lượng để đánh giá hai kiến trúc lưu trữ cho dịch vụ này, DBMS dùng B-tree và snapshot in-memory bất biến, thông qua mô hình hàng đợi M/M/c, mô hình truy cập bộ nhớ AMAT và thí nghiệm với tập dữ liệu đến một triệu bản ghi. Kết quả cho thấy kiến trúc in-memory giúp giảm độ trễ trung bình 15–30 lần, giảm P95 20–25 lần, tăng thông lượng cực đại lên hơn một bậc độ lớn và giảm năng lượng tiêu thụ. Kết quả hàm ý rằng snapshot in-memory bất biến là nền tảng phù hợp và tiết kiệm chi phí cho các dịch vụ học vụ đòi hỏi tính ổn định và phản hồi cao, đặc biệt trong các đợt công bố tại các trường đại học chuyển đổi số.

Từ khóa: tra cứu kết quả tuyển sinh; tải đột biến học vụ; kiến trúc độ trễ thấp; kiểm soát đuôi độ trễ; ngưỡng khả thi bộ nhớ

Abstract: University admissions-result lookup systems face extreme burst loads during announcement periods, while operating on semester-frozen, strictly read-only datasets. This study develops a quantitative framework to evaluate two architectural approaches for such services—traditional B-tree-based DBMS and immutable in-memory snapshots—using an integrated methodology combining M/M/c queueing analysis, AMAT-based memory modeling, and controlled experiments on datasets of up to one million admission records. Results show that the in-memory architecture reduces average latency by 15–30×, lowers P95 by 20–25×, and increases peak throughput by more than an order of magnitude, while consuming substantially less energy. These findings indicate that immutable in-memory snapshots provide a robust, scalable, and cost-efficient foundation for real-world admissions services that demand high responsiveness and stability during high-pressure release windows in digitally transformed universities.

Keywords: admissions-result lookup; burst-load academic services; low-latency architecture; tail-latency control; memory-feasibility threshold

1. Giới thiệu

Hệ thống tra cứu kết quả là một dịch vụ học vụ trọng yếu trong các cơ sở giáo dục đại học, đặc biệt tại những thời điểm công bố khi nhu cầu truy cập tăng đột biến, gây ảnh hưởng trực tiếp đến chất lượng trải nghiệm. Các nghiên cứu về hiệu năng dịch vụ trực tuyến cho thấy độ trễ và mức ổn định của hệ thống chịu tác động mạnh từ tải người dùng và thiết kế hạ tầng. Trong bối cảnh chuyển đổi số giáo dục, các trường đại học được yêu cầu nâng cao năng lực cung cấp dịch vụ học vụ trực tuyến và tối ưu hiệu quả vận hành.

Dữ liệu sau khi được phê duyệt mang tính bất biến. Tập dữ liệu có thể biểu diễn hình thức dưới dạng $D = \{(k_i, v_i)\}_{i=1}^N$, trong đó k_i là khóa định danh và v_i là kết quả cùng thông tin liên quan. Toàn bộ truy vấn đều thuộc loại đọc, không có thay đổi trực tuyến. Tuy vậy, nhiều hệ thống tra cứu vẫn vận hành trên kiến trúc cơ sở dữ liệu giao dịch tổng quát, vốn được thiết kế cho tải đọc–ghi hỗn hợp, dẫn đến chi phí vận hành và độ trễ không cần thiết trong các đợt tải cực đại.

Hiện chưa có nhiều nghiên cứu định lượng về việc lựa chọn kiến trúc tối ưu cho lớp dịch vụ đọc–bất biến đặc thù của dữ liệu. Do đó, việc xây dựng một khung phân tích kỹ thuật để đánh giá hiệu năng và điều kiện triển khai của các mô hình lưu trữ khác nhau là cần thiết nhằm hỗ trợ các cơ sở giáo dục ra quyết định phù hợp về mặt công nghệ, chi phí và chất lượng phục vụ.

2. Khung mô hình toán cho tối ưu hóa tra cứu điểm

Hiệu năng của dịch vụ tra cứu khi tải tăng đột biến được mô hình hóa bằng hệ thống hàng đợi nhiều nút M/M/c. Với tốc độ đến λ , tốc độ phục vụ μ , số nút c và hệ số tải $\rho = \frac{\lambda}{c\mu}$, hành vi xử lý được mô tả bằng công thức Erlang–C [5]:

$$P(W > 0) = \frac{\frac{(c\rho)^c}{c!(1-\rho)}}{\sum_{k=0}^{c-1} \frac{(c\rho)^k}{k!} + \frac{(c\rho)^c}{c!(1-\rho)}}.$$

Thời gian chờ trung bình $W_q = \frac{P(W > 0)}{c\mu - \lambda}$ và độ trễ trung bình $W = W_q + \frac{1}{\mu}$. Phân vị P95 khi $\rho \rightarrow 1$: $P95 \approx W + \frac{\ln 20}{c\mu - \lambda}$. Những biểu thức này phản ánh trực tiếp sự tăng mạnh của đuôi phân phối thời gian phản hồi trong giờ công bố.

Giá trị μ phụ thuộc vào chi phí truy vấn. Với DBMS sử dụng chỉ mục B-tree: $T_{Btree} \approx O(\log_b N) \cdot t_{page}$, trong đó t_{page} thuộc miền mili-giây khi phải tải trang từ bộ nhớ thứ cấp. Trong kiến trúc in-memory, thời gian truy vấn đối với mảng sắp xếp $T_{array} \approx O(\log N) \cdot t_{mem}$, và đối với bảng băm $T_{hash} \approx O(1) \cdot t_{mem}$, trong đó t_{mem} nằm trong miền nano-micro-giây, nhỏ hơn nhiều bậc độ lớn so với truy cập trang.

Sự khác biệt này được lý giải bằng mô hình truy cập bộ nhớ nhiều tầng (AMAT):

$$AMAT = H_1 T_1 + (1 - H_1) (H_2 T_2 + (1 - H_2) (H_3 T_3 + (1 - H_3) T_{mem})).$$

Nếu dữ liệu có hệ số phân mảnh F , độ trùng cache hiệu dụng $H_i^{eff} = H_i (1 - F)$. Thời gian phục vụ trung bình cho một truy vấn $T = k \cdot AMAT$, với k là số lần đọc ngẫu nhiên của binary search hoặc hash probing.

Bảng 1. Độ trễ truy cập điển hình của các tầng bộ nhớ

Tầng nhớ / thao tác	Ký hiệu	Độ trễ trung bình
Cache cấp 1	t_{L1}	1–2 ns
Cache cấp 2	t_{L2}	4–5 ns
Cache cấp 3	t_{L3}	10–15 ns
RAM	t_{mem}	80–120 ns
I/O đĩa (SSD)	t_{page}	0.1–0.2 ms
I/O đĩa (HDD)	t_{page}	5–10 ms

Do t_{mem} thấp và H_i^{eff} cao, tỉ lệ cải thiện hiệu năng giữa hai kiến trúc:

$$\frac{W_{DBMS}}{W_{inmem}} \approx \frac{T_{Btree}}{T_{array}} \text{ hoặc } \frac{T_{Btree}}{T_{hash}},$$

với giá trị thực nghiệm trong khoảng 15–30 lần đối với W và 20–25 lần đối với P95.

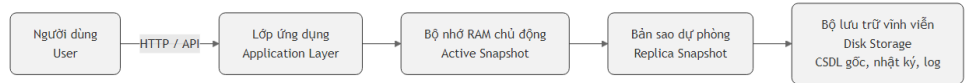
Điều kiện khả thi bộ nhớ khi lưu snapshot $M = N \cdot S \cdot r$, với N số bản ghi, S kích thước mỗi bản ghi, r hệ số nhân bản. Ràng buộc triển khai $M \leq M_{RAM}$.

Điều kiện SLA về độ trễ theo phân vị P95 $c\mu - \lambda \geq \frac{\ln 20}{P_{req}}$, hay $\lambda \leq c \left(\frac{1}{T} \right) - \frac{\ln 20}{P_{req}}$.

Các mô hình này là cơ sở định lượng cho phân tích kiến trúc triển khai ở Mục 3.

3. Kiến trúc triển khai in-memory bất biến cho dịch vụ tra cứu

Với dữ liệu bất biến, kiến trúc in-memory sử dụng snapshot $D^{(i)}$ nạp toàn bộ vào RAM và phục vụ ở chế độ chỉ đọc $D^{(i)} = \{(k_i, v_i)\}_{i=1}^N$. Snapshot mới được tạo ngoại tuyến, kiểm tra toàn vẹn (checksum, số lượng bản ghi, khóa trùng), sau đó thay thế bằng thao tác hoán đổi nguyên tử $D^{(i+1)} \xleftarrow{\text{atomic_swap}} D^{tmp}$, với độ phức tạp thời gian $T_{swap} = O(1)$, do chỉ thay con trỏ tham chiếu vùng nhớ.



Hình 1. Sơ đồ kiến trúc triển khai in-memory bất biến cho dịch vụ tra cứu

Để đảm bảo độ sẵn sàng, snapshot được nhân bản thành r bản sao. Với xác suất lỗi độc lập của nút là P . $A = 1 - P^r$, cho $r = 2, P = 0.01 \rightarrow A \approx 0.9999$, tương đương ~52 phút downtime/năm.

Thời gian nạp snapshot: $T_{load} = \frac{|snapshot|}{B_{mem}}$, với $|snapshot|$ 0.2–1 GB và $B_{mem} \approx 10\text{--}20$ GB/s, cho thời gian nạp từ vài ms đến vài trăm ms. Việc nạp diễn ra song song và không ảnh hưởng truy vấn do snapshot chỉ có hiệu lực sau khi atomic swap hoàn tất.

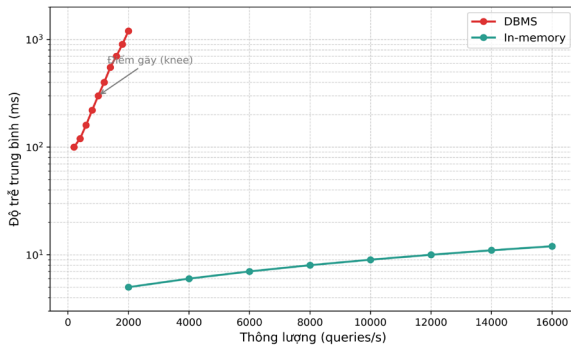
Hai cấu trúc dữ liệu được sử dụng: $\mu_{array} \approx \frac{1}{O(\log N) \cdot t_{mem}}$, $\mu_{hash} \approx \frac{1}{O(1) \cdot t_{mem}}$, với t_{mem} nano-micro giây. Giá trị μ vượt xa μ của DBMS dùng B-tree, vốn bị chi phối bởi truy cập trang.

Để kiểm chứng mô hình toán, mô phỏng được triển khai trên máy chủ 8 lõi CPU Xeon, RAM 64 GB. Hai kiến trúc: DBMS (PostgreSQL 15, B-tree) và in-memory (mảng sắp xếp, bảng băm). Tập dữ liệu gồm $10^5\text{--}10^6$ bản ghi (khóa 12 bytes, giá trị 64 bytes). Tải đến sinh theo Poisson, tăng dần cho đến khi P95 vượt ngưỡng. Các chỉ tiêu đo: độ trễ trung bình W , phân vị P95, thông lượng cực đại, hệ số biến thiên.

Bảng 2. Kết quả đo hiệu năng

Kiến trúc	W (ms)	P95 (ms)	Throughput (qps)	CV
DBMS (B-tree)	150–160	330–350	~1,200	0.4+
In-memory (sorted)	8	15	12,000+	0.05
In-memory (hash)	6	12	15,000+	0.03
In-memory (hybrid 80%)	18	25	9,200+	0.07

Đường cong thông lượng–độ trễ (Hình 2) cho thấy DBMS xuất hiện điểm “knee” sớm, trong khi in-memory duy trì quan hệ tuyến tính đến vùng tải rất cao. Điều này phù hợp với mô hình tail latency: ổn định của μ và phương sai nhỏ dẫn tới phân phối thời gian phản hồi phẳng hơn.



Hình 2. Quan hệ giữa thông lượng và độ trễ trung bình (knee curve)

Đo năng lượng cho thấy DBMS tiêu thụ ~78 W, in-memory ~52 W. Vì vậy hiệu suất-trên-chi-phí $\eta_{inmem} \approx 5-10 \times \eta_{DBMS}$.

Kết quả xác nhận rằng kiến trúc in-memory bất biến đáp ứng tốt yêu cầu phục vụ nhanh, ổn định và quy mô lớn theo chu kỳ công bố.

4. Thảo luận và hàm ý cho quản lý giáo dục

Kết quả mô hình và thực nghiệm cho thấy kiến trúc in-memory bất biến phù hợp tự nhiên với đặc tính dữ liệu. Khi tải λ tăng đột biến trong giai đoạn công bố kết quả, hệ thống DBMS truyền thống nhanh chóng tiến vào trạng thái $\rho \rightarrow 1$, làm P95 tăng mạnh theo hàm mũ. Ngược lại, nhờ μ lớn và phương sai thời gian phục vụ nhỏ, kiến trúc in-memory duy trì sự ổn định của toàn bộ phân phối thời gian phản hồi, cải thiện đáng kể khả năng đáp ứng tức thời.

Snapshot bất biến giúp chuẩn hóa vòng đời dữ liệu: phòng đào tạo tạo snapshot sau khi được phê duyệt và triển khai trực tiếp lên hệ thống tra cứu, không phụ thuộc các cơ chế ghi hoặc khóa trong DBMS giao dịch. Điều này làm giảm sai lệch dữ liệu chạy, loại bỏ xung đột đọc-ghi, và đơn giản hóa quy trình kiểm tra – đối chiếu giữa các đơn vị quản lý học vụ.

Việc tách lớp tra cứu ra khỏi hệ thống quản lý đào tạo cũng giúp giảm tải đáng kể cho DBMS gốc, vì hầu hết các truy vấn thực hiện trong thời gian cao điểm đều thuộc loại đọc. Lớp tra cứu có thể hoạt động như một API dữ liệu bất biến có độ trễ thấp, tăng khả năng mở rộng theo số lượng người dùng.

Bảng 3. Miền đề xuất cho triển khai in-memory trong dịch vụ tra cứu

Thông số	Miền khả thi
Quy mô dữ liệu N	$10^5 - 10^6$ bản ghi
Kích thước bản ghi S	60–100 bytes
RAM khả dụng	32–128 GB
Tải đỉnh λ	5,000–20,000 qps
Độ trễ yêu cầu P95	$\leq 20-40$ ms
Số nút c	4–8

Khía cạnh tin cậy cũng được đảm bảo nhờ mô hình nhân bản snapshot và kỹ thuật state-machine

replication. Với xác suất lỗi nút thấp và nhân bản kép, độ sẵn sàng hệ thống đạt gần mức phần cứng. Việc dữ liệu bất biến và không tồn tại ghi trực tiếp làm giảm các tình huống mất đồng bộ thường gặp trong DBMS giao dịch.

Tuy nhiên, mô hình cũng có giới hạn. Khi trường có nhu cầu cập nhật nhiều lần trong ngày, việc tạo snapshot liên tục có thể tốn kém. Các truy vấn phức hợp hoặc dữ liệu quy mô hàng chục triệu bản ghi vẫn yêu cầu DBMS truyền thống. Do vậy, kiến trúc in-memory phù hợp nhất với tập truy vấn đọc lớn, dữ liệu được chuẩn hóa và cập nhật theo chu kỳ rõ ràng.

5. Kết luận

Nghiên cứu đã thiết lập một khung phân tích toán học dựa trên mô hình hàng đợi M/M/c, mô hình AMAT và chi phí truy vấn để đánh giá hiệu năng của kiến trúc in-memory bất biến trong dịch vụ tra cứu. Cùng với mô phỏng thực nghiệm, kết quả cho thấy kiến trúc này giảm độ trễ trung bình 15–30 lần, giảm P95 20–25 lần và tăng thông lượng lên hàng chục nghìn truy vấn/giây so với DBMS sử dụng B-tree, đồng thời giảm chi phí năng lượng và vận hành.

Các ràng buộc triển khai được xác định bởi ba bất đẳng thức: khả thi bộ nhớ $M \leq M_{RAM}$, biên tải SLA $\lambda \leq c(1/T) - \ln(20)/P_{req}$, hiệu suất-trên-chi-phí $\eta_{inmem} > \eta_{DBMS}$.

Khi thỏa mãn các điều kiện này, in-memory bất biến trở thành giải pháp tối ưu cho dịch vụ tra cứu nhanh, ổn định và chi phí thấp, phù hợp với yêu cầu chuyển đổi số trong giáo dục đại học.

Hướng phát triển gồm mở rộng snapshot sang cụm phân tán, tích hợp chữ ký số cho mỗi snapshot và nghiên cứu mô hình lai ghép khi quy mô dữ liệu vượt giới hạn RAM.

Tài liệu tham khảo

- [1] Akamai Technologies. (2017). *The State of Online Performance Report*.
- [2] Barroso, L. A., Clidaras, J., & Hölzle, U. (2013). *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines* (2nd ed.). Morgan & Claypool Publishers.
- [3] Thủ tướng Chính phủ. (2022). *Quyết định số 131/QĐ-TTg ngày 25/01/2022 phê duyệt Đề án “Tăng cường ứng dụng công nghệ thông tin và chuyển đổi số trong giáo dục và đào tạo giai đoạn 2022, 2025, định hướng đến năm 2030”*.
- [4] Kleinrock, L. (1975). *Queueing Systems, Volume 1: Theory*. Wiley-Interscience.
- [5] Garcia-Molina, H., Ullman, J., & Widom, J. (2008). *Database Systems: The Complete Book*. Pearson Education.
- [6] Kabakus, A. T., & Kara, R. (2017). A performance evaluation of in-memory databases. *Journal of King Saud University – Computer and Information Sciences*, 29(4), 520–525.