수강신청 시스템 부하 테스트 및 캐시 최적화 성능 비교 보고서

1. 테스트 개요

1.1 목적

수강신청 시스템에 스파이크 트래픽을 가하여 시스템의 처리 한계를 측정하고, Caffeine Write-Through 캐시 적용 전후의 성능 차이를 정량적으로 비교한다.

1.2 테스트 환경

항목	사양
Application	Spring Boot 4.0.2 + Java 17
Database	H2 In-Memory
동시성 제어	JPA Pessimistic Lock
캐시	Caffeine (maximumSize=100, expireAfterWrite=60s)
부하 도구	k6 v1.6.1
실행 환경	macOS (로컬)

1.3 트래픽 배분

요청 유형	비율	엔드포인트
수강신청	50%	`POST /enrollments`
강좌 조회	30%	`GET /courses` (50% 확률로 학과 필터)
시간표 조회	20%	`GET /students/{id}/timetable`

1.4 테스트 시나리오

200 VU 스파이크 테스트 - 10s 워밍업(→20 VU) → 5s 급증(→200 VU) → 30s 피크 유지 → 감소

10,000 VU 스파이크 테스트 - 10s 워밍업(→500 VU) → 10s 급증(→10,000 VU) → 30s 피크 유지 → 감소

---

2. 테스트 결과

2.1 200 VU 스파이크 테스트

전체 요약

지표	Cache OFF	Cache ON	변화
총 요청 수	77,426	78,482	+1.4%
TPS	1,290 req/s	1,306 req/s	+1.2%
전체 평균 응답 시간	3.23ms	2.00ms	**-38.1%**
전체 p95 응답 시간	10.30ms	5.93ms	**-42.4%**
5xx 에러율	0.00%	0.00%	-

엔드포인트별 레이턴시 비교

엔드포인트	지표	Cache OFF	Cache ON	개선율
강좌 조회	avg	5.08ms	2.78ms	**-45.3%**
	p50	4.68ms	1.43ms	**-69.4%**
	p95	13.21ms	7.48ms	**-43.4%**
수강신청	avg	2.55ms	1.76ms	**-31.0%**
	p50	1.18ms	1.09ms	-7.6%
	p95	8.42ms	4.39ms	**-47.9%**
시간표	avg	2.15ms	1.42ms	**-34.0%**
	p50	0.89ms	0.84ms	-5.6%
	p95	7.78ms	3.95ms	**-49.2%**

2.2 10,000 VU 스파이크 테스트

전체 요약

지표	Cache OFF	Cache ON	변화
총 요청 수	183,279	359,892	+96.4%
TPS	1,932 req/s	3,787 req/s	+96.1%
전체 평균 응답 시간	1,860ms	888ms	-52.3%
전체 p95 응답 시간	3,380ms	1,490ms	-55.9%
5xx 에러율	0.00%	0.00%	-

엔드포인트별 레이턴시 비교

엔드포인트	지표	Cache OFF	Cache ON	개선율
강좌 조회	avg	1,880ms	867ms	-53.9%
	p50	2,480ms	955ms	-61.5%
	p95	3,380ms	1,440ms	-57.4%
수강신청	avg	1,860ms	898ms	-51.7%
	p50	2,470ms	972ms	-60.6%
	p95	3,400ms	1,510ms	-55.6%
시간표	avg	1,850ms	896ms	-51.6%
	p50	2,460ms	967ms	-60.7%
	p95	3,360ms	1,510ms	-55.1%

---

3. 분석

3.1 200 VU vs 10,000 VU 비교

저부하(200 VU) 환경에서는 캐시 효과가 주로 **강좌 조회**에 집중되었다 (p50 69% 개선). 수강신청과 시간표 조회는 DB 직접 접근이므로 평균 레이턴시 개선은 제한적이었다.

그러나 **10,000 VU 고부하** 환경에서는 양상이 크게 달라진다:

관점	설명
처리량 2배	캐시 OFF 1,932 TPS → 캐시 ON 3,787 TPS. 동일한 부하에서 약 2배의 요청을 처리.
전 엔드포인트 개선	강좌 조회뿐 아니라 수강신청, 시간표까지 50% 이상 개선. 이는 캐시가 DB 커넥션 풀 경합을 줄여 간접적으로 모든 엔드포인트에 효과를 준 것.
p50 60% 감소	중위 응답 시간이 2.4초대에서 950ms대로 감소. 대다수 사용자의 체감 성능이 크게 개선됨.

3.2 캐시 전략 분석

적용된 Write-Through 캐시 전략:

- **캐시 대상**: `GET /courses` (전체 목록), `GET /courses?departmentId=` (학과별)
- **캐시 무효화**: 수강신청/취소 성공 시 해당 캐시 즉시 evict
- **안전망**: `expireAfterWrite=60s` TTL로 최대 60초 내 자동 갱신

이 전략은 **읽기 비율이 높은(30%) 강좌 조회**를 캐시하여 DB 부하를 줄이고, 결과적으로 **DB 커넥션 풀의 가용 자원**이 수강신청(비관적 락) 처리에 집중될 수 있게 했다.

3.3 안정성

두 시나리오 모두 **5xx 에러 0건**을 기록했다. 비관적 락 기반의 동시성 제어가 10,000 VU에서도 정원 초과 없이 안전하게 동작함을 확인했다.

---

4. 결론

규모	핵심 수치	요약
200 VU	p95 42% 감소	저부하에서는 강좌 조회 중심으로 유의미한 개선
10,000 VU	**TPS 96% 증가, p95 56% 감소**	고부하에서 처리량 2배, 응답 시간 절반으로 감소

Caffeine Write-Through 캐시는 **고부하 환경에서 효과가 극대화**되며, 캐시 대상이 아닌 엔드포인트까지 **DB 경합 해소를 통한 간접적 성능 개선** 효과를 보였다.