WebSocket 기반 실시간 채팅 시스템

실시간 동기화 구조 설계 검증 실험 보고서

https://github.com/bumjinDev/ChatService

목차

동시성 시나리오 테스트 목적

1.1 프로젝트 개요 및 실험 목적

본 보고서는 실무 시스템에서 자원 점유 통제를 설계함에 있어 동기화 구조의 통제 단위가 정합성과 병목 회피를 동시에 충족할 수 있는가라는 구조적 물음을 출발점으로 한다. 이 실험은 성능 비교가 아닌 통제 구조의 설계 타당성에 대한 이론적 검증을 핵심 목표로 한다.
조건 분기 기반 구조는 상태 판단과 자원 갱신이 분리되어 있어 동시 접근 시 판단 이후 상태가 변경되는 경우를 제어할 수 없다. 이는 구조적으로 race condition을 방지할 수 없음을 의미하며, 병목은 발생하지 않지만 결과는 항상 비일관성이다.
뮤텍스 락 기반 구조는 조건 판단과 상태 갱신을 임계 구역 내에서 수행하므로 자원 보호는 가능하지만, 모든 진입자가 단일 락을 획득해야 하므로, 동시 요청이 집중될 경우 지연이 누적되며 병목이 발생한다. 또한 락 내부에서 상태 객체가 외부에 의존하면 일관성 유지가 구조 외부에 의존하게 된다.
세마포어 기반 구조는 점유 허용량을 permit 단위로 명시적으로 표현하며, tryAcquire 시점에서 판단과 자원 점유가 동시에 atomic하게 이루어진다. 이는 병목을 피하고 race를 차단하며, 통제 단위가 구조 내부에 응집된다는 점에서 설계상 가장 명료한 통제 경계를 가진다.
이러한 의문은 단순 성능 비교나 구현 편의성의 문제가 아닌 설계 구조 자체의 정당성을 판단하기 위한 것이다. 따라서 본 보고서는 관측된 문제에 대한 분석이 아니라, 설계적 의심을 실험적으로 입증하기 위한 목적에서 출발한다.

1.2 프로젝트 중점 사항

본 보고서는 실시간 서비스 시스템의 동시 입장 제어 구조가 단순한 기능적 구현의 효율성이나 처리량 비교에 그치는 것이 아니라, 각 구조가 실제 운영 환경에서 race condition의 구조적 차단, 병목의 실질적 억제, 상태 일관성의 정량적 입증, 그리고 반복 실험을 통한 결과의 일관성과 재현성까지 통합적으로 검증할 수 있는지에 중점을 두고 설계되었다. 모든 계측 기준과 실험 절차는 개별 사례의 성공/실패가 아닌, 동일 조건에서 반복 수행 시 일관된 결과가 재현되는지, 그리고 각 구조가 내포한 설계적 한계가 실제 데이터상 어떻게 표출되는지를 명확히 드러내는 데 집중된다.

따라서, 실험의 모든 평가 관점은 단일 스냅샷 기반의 일시적 수치가 아닌, 상태 변화의 전구간 추적, race condition 발생 구간의 실시간 검출, DB·메모리·세션 등 상태 객체의 일관성 붕괴/복구 흐름의 정량적 계측, 그리고 대량 동시 요청 환경에서의 병목 및 경합 발생 시점의 자동 검출로 귀결된다.

이러한 실험적 중점 사항은 단순히 구조별 성능 차이를 논증하는 데 그치지 않고, 실제 서비스 수준에서 요구되는 구조적 안정성과 실무적 신뢰성을 객관적 데이터로 입증하는 데 있으며, 궁극적으로는 설계 선택의 명확한 기준, 실무 적용성 평가, 실험 결과의 재현성과 검증 가능성을 최우선 원칙으로 한다.

1.3 프로젝트 구조

본 보고서는 실험의 계측·분석 원칙을 먼저 고정한 뒤, 각 구조별 실험 시나리오의 전개와 결과 해석, 실무적 타당성 평가까지의 전 과정을 계층적으로 기술하였다.

2장에서는 계측·분석 기준과 실험 자동화 환경을 명확히 정의하여, 실험의 객관성과 반복 가능성, 데이터 신뢰성을 확보하는 방법론을 기술한다.
3장에서는 각 동시성 제어 구조별 실험 시나리오 및 계측 포인트를 구체적으로 서술하여, 구조별 차별화된 측정 항목과 분석 프레임을 제시한다.
4장에서는 실험 결과의 정량적 데이터 및 주요 현상을 비교 분석하여, 구조적 병목, race condition 발생 여부, 상태 일관성 붕괴 및 복구 가능성 등 실질적 관찰 데이터를 통계적으로 제시한다.
5장에서는 각 구조별 실험 결과의 해석과 실무적 효과, 한계 및 확장 방향을 심층적으로 분석하며, 단순 수치 비교를 넘어 구조별 설계 선택의 실무 적용성과 장기적 확장성까지 입증한다.
마지막 6장에서는 실험 결과에 기반한 종합 결론 및 구조 선택 가이드를 제시하여, 실무 환경에서의 최적 선택 기준, 적용 시 주의점, 향후 연구 및 개선 방향 등을 체계적으로 정리한다.

이러한 전체 구조는 실험 설계의 철학적 배경과 계측의 객관성, 결과의 재현성과 실무 적용성까지 일관된 논리로 연결되는 것을 목표로 하며,
각 장의 배치와 내용은 모든 실험적 관찰이 단순 현상 나열이 아니라 구조적 타당성과 실증적 입증으로 이어지도록 설계되었다.

동시성 테스트 분석 기준 및 환경 설정

1. 공통 계측·분석 기준

실험의 모든 구조(조건 분기/뮤텍스 락/세마포어)는 동일한 계측 항목, 집계 방식, 통계 환경 하에서 테스트된다.

1.1 계측 항목 및 집계 지표

- 처리량(ops/sec) : 단위 시간(1초) 내 처리 완료 건수
- 평균/최대/최소 응답 시간(ms) : 요청-응답까지 전체 소요 시간(밀리초)
- 락 대기/임계구역 시간: 각 요청별 락 진입 전 대기/임계구역(permit) 반환까지의 실제 소요 시간
- race condition 발생률 : 허용 인원(permit/registry/DB 기준) 초과 입장, 중복 permit/세션/DB 상태 불일치 등 구조적으로 불가한 상태 발생 비율
- 상태 일관성 : permit/세션/DB 등 주요 상태의 불일치 발생 건수/비율

1.2 계측 방식 및 자동화 집계

- 자동화 툴: Playwright, Selenium 5 기반 멀티 세션 시뮬레이터, 멀티 세션 자동화 도구로 1000명 동시 입장/종료 시나리오 반복 수행
- 상태 계측: DB/메모리/permit/세션 상태를 REST API, DB CLI 등으로 자동 스냅샷/로그 저장, 모든 데이터는 csv/json/엑셀로 자동 집계
- 시각화 및 분석: 서버/DB/응답시간/대기시간/분산 등 모든 계측값은 boxplot, 히스토그램, 타임라인 등 그래프로 통합 분석

1.3 계측·분석 데이터 구조

- 집계 단위 : 구조별 10회 반복, 각 반복마다 처리량/응답시간/race 발생률/상태 불일치 등 집계
- 로그/스냅샷 : 모든 주요 이벤트(입장/permit 획득/중복/초과/상태 불일치 등)는 자동 로그 및 상태 스냅샷으로 기록
- 상태 일관성: 모든 주요 이벤트(입장 permit 획득/등록/초과/불일치/누락 등)는 자동 로그 및 상태 스냅샷으로 기록
- 분석 기준 : 평균/최대/최소/표준편차/99th percentile 등 수치화, 대표 케이스는 boxplot/그래프 등으로 시각화

1.4 계측·분석 항목 표

계측 항목	정의	계측 방식
처리량(ops/sec)	단위 시간 내 처리 완료 건수	자동 스크립트 통계 집계
응답 시간	요청~응답까지 총 소요 시간	클라이언트/서버 자동 로그
락 진입 대기 시간	동기화 지점 진입까지 대기한 시간	서버 로그 자동 기록
임계구역 처리 시간	동기화 지점 통과 후 실제 처리 완료까지 소요 시간	서버 로그 자동 기록
race condition	허용 인원 초과/중복 등 구조적으로 불가한 상태 발생 건수/비율	상태 snapshot, 자동 집계
상태 일관성	permit/세션/DB 등 주요 상태 불일치 발생 건수/비율	상태 snapshot, 자동 집계

2 계측 환경

구분	상세 내용
서버/클라이언트 환경	Spring Boot 3.x 기반 WebSocket 서버, JDK 17, 리눅스, 단일 인스턴스, 8 vCPU/32GB RAM VM
테스트 자동화	Playwright/Selenium headless, 1000명 동시 입장 자동화 스크립트, 각 세션·쿠키·permit 독립 상태
계측 자동화	서버/DB/permit/세션 상태 자동 스냅샷 및 csv/json 저장, 에러 로그/중복/초과/불일치 등 자동 이벤트 파싱
시각화/분석	matplotlib, pandas 등으로 그래프/통계 집계
반복/통계	각 구조별 10회 반복, 평균/최대/최소/분산 등 통계 산출

동시성 시나리오 테스트 정의 및 계측 포인트

본 장에서는 2장에서 정의한 공통 계측/분석 기준을 전제로, 각 동기화 구조(조건 분기/뮤텍스 락/세마포어/리엔트런트락)의 계측 시나리오와 구조적 차별화 포인트, 그리고 계측 방식(ops/sec, 병목, 상태 일관성 등) 및 결과 해석의 실질적 기준을 명확히 제시한다.
모든 시나리오는 처리량(ops/sec), 응답 지연, 상태 일관성, race condition, 반복 실험, 데이터 분석 등 2장 기준을 따른다.

동시성 테스트 시나리오

1. 조건 분기 기반 구조 (If-Else)

[실험 시나리오 및 계측 플로우]

- Step 1: 1000명 동시 입장 요청(자동화 도구 이용), 각 요청은 currentCount < MAX 조건 분기 후 입장 시도
- Step 2: 조건 통과 시 상태 객체(Map/Counter) 직접 갱신, 입장 성공 처리
- Step 3: 모든 요청의 상태 snapshot(DB, 세션, permit 등) 자동 저장
- Step 4: 입장 성공·실패·중복 등 이벤트 발생 즉시 자동 로그 및 이벤트 카운트
- Step 5: 10회 반복, 평균/최대/최소/표준편차 집계

2. 뮤텍스 락 기반 구조 (Mutex/Synchronized)

[실험 시나리오 및 계측 플로우]

- Step 1: 1000명 동시 입장 요청, 각 요청은 synchronized/lock 임계구역 진입 시도
- Step 2: 임계구역 내에서 currentCount < MAX 판단 및 상태 객체(Map/Counter) 갱신
- Step 3: 임계구역 진입 대기, 점유, 해제 시각 자동 계측(락 경합, 대기 시간 포함)
- Step 4: 상태 snapshot(DB, 세션, permit 등) 및 이벤트 로그 자동 저장

3. ReentrantLock 기반 구조

[실험 시나리오 및 계측 플로우]

- Step 1: 1000명 동시 입장 요청, 각 요청은 ReentrantLock.lock() 또는 tryLock()으로 임계구역 진입 시도
- Step 2: 임계구역 내 currentCount < MAX 판별 및 상태(Map/Counter) 갱신
- Step 3: 락 진입 대기/점유/해제/반납 시각, 대기 큐 길이, 락 획득 대기 시간 등 자동 계측
- Step 4: Fair/NonFair 옵션별 실험, 각각의 락 경합/대기/Starvation/공정성 계측
- Step 5: 상태 snapshot 및 이벤트 로그 자동 저장, 10회 반복 통계 집계

4. 세마포어 기반 구조 (Semaphore)

[실험 시나리오 및 계측 플로우]

- Step 1: 1000명 동시 입장 요청, 각 요청은 Semaphore.tryAcquire()로 permit 원자적 점유 시도
- Step 2: permit 점유 성공 시 상태 객체(Map/Counter/permit) 동기화, 실패 시 거부 처리
- Step 3: permit/세션/DB 상태 snapshot 및 모든 이벤트(중복, 초과, 불일치 등) 자동 저장
- Step 4: tryAcquire/release 타이밍 및 경합 상황 자동 로그
- Step 5: 10회 반복, 통계 집계

동시성 테스트 별 관찰 기준

동기화 구조 유형	계측 항목 (세부 관찰/실험 포인트)	측정/관찰 방식 (정량/정성)	해석/분석 관점 (실험 목적)
조건 분기 기반	동시 입장 중복 진입률 / race condition 빈도	이벤트 카운트, WARN/Error 로그, 자동 계측	조건-실행 분리의 구조적 한계, race condition 불가피 발생, 실무 적용 한계 증거화
	상태 일관성(불일치율)	permit/세션/DB 상태 스냅샷, 자동 비교	반복적 불일치/장애(중복 입장, 인원 초과, 데이터 불일치) 실증
	복구율	반복 실험, 수동/자동 복구 테스트	0%에 수렴 시 구조 내 복구 경로 부재, 장애 자동 복구 불가
뮤텍스 락 기반	race condition 차단율	이벤트 로그, race 유무 자동 계측	100% 근접 시 동시성 보호 효과, 구조적 race 차단 효과 입증
	병목/응답 지연	평균/최대 응답 시간, 처리량(ops/sec)	락 경합에 따른 처리량 저하/확장성 한계 실증
	상태 일관성/결함	상태 스냅샷, 이벤트 로그	락 내부/외부 상태 경계/제한 명확화, 동기화 범위 한계
세마포어 기반	race condition 발생률	이벤트 로그, race condition 유무	0% 일치 시 CAS 기반 atomic 구조의 오류 차단 실증
	처리량/응답 시간	처리량(ops/sec), 평균/최대 응답 시간	락 기반 대비 우수, 병목 원천 차단, 구조적 효과 입증
	상태 일관성	permit/세션/DB 상태 스냅샷, 자동 비교	반복적 유지 시 permit 기반 활동 통제 구조 신뢰성 수치로 증명

주요 관찰 대상 및 집계 방법

본 장에서는 실시간 동기화 실험에서 상태 추적·이벤트 계측이 실제로 가능한 객체 및 모듈을 대상으로 역할, 계측 포인트, 자동화 로그 등의 기준을 명확히 정의한다.

[주요 관찰 대상]

구분	핵심 역할	주요 계측 이벤트 / 관찰 포인트	계측 방식
SemaphoreRegistry	Permit(동시 입장 허용 수) 관리 및 세마포어 동기화의 중심	permit 잔여 수 변화	서버 로그, 자동화 스냅샷, 누락/중복 모니터링
		점유/해제 성공/실패	
		TTL 만료 permit 자동 회수	
		동시 입장 race 차단 실증	
ChatSessionRegistry	실시간 WebSocket 세션 추적/관리	세션 생성/입장	세션 수, permit 연동 상태, 자동화 로그/스냅샷
		중복/미반납/session-leak	
		세션-permit 연동 일치성	
RoomJoinRepository	DB상 방 인원/입장/퇴장 따른 세션 목록 관리	입장/퇴장 트랜잭션 성공/실패	DB/메모리/세션 실시간 스냅샷, 불일치 건수 기록
		인원 초과/불일치 실증	
		트랜잭션 롤백/상태 불일치 자동 감지	
ChatServiceScheduler	방/permit/세션 비정상 상태 자동 정리, dead room 관리	dead room/permit 미반납/세션 누수 감지	스케줄러 실행 로그, 정리/복구 성공·실패 집계
		TTL 기반 자동 permit 반환	
		강제 정리/복구 이벤트 기록	

[관찰 및 집계 방법]

구분	세부 항목/활동	설명 및 분석 관점
데이터 집계	구조별/반복별 결과 분리 집계	테스트 구조, 반복, 시나리오별로 결과(처리량, 상태, 에러 등) 자동 집계, csv/json으로 저장
로그/스냅샷	실시간 상태 snapshot, 자동 로그	서버/DB/permit/세션 등 주요 상태를 실시간 스냅샷/로그로 남기고, 오류/불일치/race 이벤트를 자동 기록
시각적 분석	표/그래프화	집계된 데이터를 표, 히스토그램, 박스플롯 등으로 시각화하여 구조별 병목, race, 처리량 등 변동을 한눈에 분석
정량/정성 분석	수치 통계 + 예외 로그	평균/최대/99th percentile, 분산 등 통계적 수치와, 대표 로그/이벤트 등 정성 분석을 병행
Cross-check	상태 일관성 자동 교차검증	DB, 세션, permit 등 모든 상태의 불일치 여부를 자동 비교, 구조별/반복별 일관성 지표 산출
반복성/재현성	10회 반복, 통계 집계	각 구조별 10회 이상 반복 실험, 반복별 결과(처리량, 상태, 에러 등)의 평균/최대/분산 등 통계 분석
결과 검증/보고	자동화 증거 파일, 로그 증적	각 실험별 결과, 로그, snapshot을 증적(자동화 저장)하여 신뢰성·재현성 보장

동시성 시나리오 테스트 및 결과 분석

테스트 환경 정의

1. 테스트 요구 사항 정리

구분	도구/프레임워크	용도 및 목적
부하 시뮬레이션	JMeter, Playwright	동시 입장/요청/퇴장 등 대량 시나리오 자동 반복 및 응답 계측
서버 로그 계측	Log4j2 (Spring Boot)	race 발생, 상태 불일치, permit 누수/복구 등 주요 이벤트 자동 로깅
DB 계측	Oracle SQL	permit 상태, DB row 실시간 스냅샷 및 상태 불일치 탐지
상태/이벤트 캡처	Shell, Python Script	실험 반복별 상태 dump, 복구 이벤트 발생 시점 캡처 자동화
데이터 저장/분석	Excel, CSV, Pandas	반복 계측 데이터 저장, 통계 분석, 정량 결과 산출
시각화	matplotlib, Excel Chart	처리량/응답시간/에러/복구 시계열 및 분포 그래프 생성
복구 자동화	Scheduler, Bash Script	permit/세션 누락, DB 불일치 자동 복구 실행 및 성공률 측정

2. 실험 실행 조건

- 실험 대상 구조: 조건 분기 기반, 뮤텍스 락 기반, 세마포어 기반(필요시 추가 구조)
- DB/메모리/캐시 초기화: 실험 전 매 시나리오별 상태 초기화 스크립트 사용
- 로그 및 계측 방식: 각 반복마다 서버 로그, DB 스냅샷, permit/queue 상태 dump, 클라이언트 응답 코드 자동 저장
- 테스트 반복: 각 구조별 10회 반복, 각 반복마다 완전 독립 상태(초기화) 보장
- 에러/복구/불일치 이벤트: 자동 탐지 및 로그 캡처, 상태 변화 전후 스냅샷 수집

3. 계측·검증 항목

- 주요 계측 지표: 처리량(ops/sec), 평균 응답시간(ms), 최대/최소/표준편차, race event 발생률, 상태 불일치 비율, permit 누수/이중 할당/복구 성공률
- 추가 검증: DB-permit-session 상태 일치 검증, 에러 발생 시 자동 복구 시나리오 진입/성공률 측정
- 시계열/분포 데이터: 요청-응답 타임라인, 병목 발생 구간, race event 타임라인 등

4. 실험 자동화 및 계측 도구

- 자동화 스크립트: Python, Bash, Playwright 등 조합
- 계측 로깅: Log4j2(서버), shell/redis-cli/DB CLI(dump), 엑셀/CSV/JSON(데이터 저장)
- 결과 시각화: matplotlib, pandas, Excel 그래프, 직접 스크린샷 등

동시성 시나리오 테스트 및 결과 분석

전체 요약 보고서

본 절의 목적은 실시간 동기화 구조(조건 분기 기반, 뮤텍스 락 기반, 세마포어 기반) 각각에 대해 10회 반복 계측을 통해 처리량, 응답시간, race condition 발생률, permit 누수율, 복구 성공률 등 주요 성능/신뢰성 지표를 집계·분석함으로써, 각 구조가 실제 동시성 제어·자원 정합성·에러 복구 측면에서 어떤 한계와 설계적 강점을 가지는지 실증적으로 비교·증명하는 것이다.
이를 통해 단순 이론적 우열이 아닌, 실제 반복 계측에서 드러나는 구조적 차이와 예외/복구/불일치 현상에 대한 실측 근거를 도출하고, 실무 설계 선택의 타당성을 확보한다.

[각 계측 항목 별 10회 테스트 수행 결과 표]

계측 항목	조건 분기 기반	뮤텍스 락 기반	ReentrantLock	세마포어 기반
평균 처리량(ops)				
평균 응답시간(ms)				
최대 처리량(ops)				
최소 처리량(ops)				
표준편차(ops)				
race 발생률(%)				
상태 불일치율(%)				
permit 누수율(%)				
복구 성공률(%)				
주요 에러 발생(건)				

[그래프/시각화]

[박스플롯/히스토그램/라인차트 등]
(각 구조별 처리량, 응답시간, 에러 분포 등 대표 그래프 삽입)

[대표값 해설/경향성 요약]

[박스플롯/히스토그램/라인차트 등]
(각 구조별 처리량, 응답시간, 에러 분포 등 대표 그래프 삽입)

동시성 시나리오 테스트 및 결과 분석

조건 분기 (IF-ELSE)

본 절은 조건 분기(If-Else) 기반 임계구역 진입 구조에 대해 10회 반복 실험에서 관측된 원본 데이터를 반복별로 제공한다.
해당 구조는 동시성 통제를 자바 레벨의 단순 조건문 분기로만 수행하므로, race condition, 상태 불일치, permit 이중 할당 등 구조적 결함이 실험 반복별로 어떤 양상과 빈도로 발생하는지를 실증적으로 드러내는 데 목적이 있다.

표 아래에는 대표 반복에서 발견된 이중 점유, DB-세션 불일치, permit 누락 등 실제 이벤트 로그와 상태 변화를 함께 제시하여, "조건문 분기 기반 설계가 실무 환경에서 왜 신뢰성을 보장하지 못하는지"를 구조적으로 입증한다.

[표 – 반복 별 계측 데이터]

반복	처리량(ops)	평균 응답(ms)	race 발생(건)	permit 누수(건)	상태 불일치(건)
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					

[대표 반복 상세 로그/상태 변화(캡처/설명)]

1. 반복 N회 시점 [] 문제 발생

[Dump]

[설명/해설]

동시성 시나리오 테스트 및 결과 분석

모니터 락(synchronized)

본 절은 자바 기본 모니터 락(synchronized) 기반 임계구역 보호 구조에서 10회 반복 실험을 통해 확보한 원본 데이터를 반복별로 제시한다.
synchronized 구조는 단일 모니터락에 모든 경쟁 스레드를 집중시키므로, race condition 자체는 원천적으로 차단되지만, 임계구역 경합에 의한 처리량 저하, 대기시간 증가, 특정 반복에서의 병목 현상이 실제로 어떻게 발생하는지 실측 데이터를 통해 검증한다.

표 하단에는 대표 반복의 대기열/경합 로그, 평균 응답시간 급증, 스레드 starvation/lock 경합 등 락 기반 구조의 병목/처리량 한계를 보여주는 실제 로그/상태 변화를 함께 수록한다.

[표 – 반복별 계측 데이터]

반복	처리량(ops)	평균 응답(ms)	race 발생(건)	permit 누수(건)	상태 불일치(건)
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					

[대표 반복 상세 로그/상태 변화(캡처/설명)]

1. 반복 N회 시점 [] 문제 발생

[Dump]

[설명/해설]

동시성 시나리오 테스트 및 결과 분석

ReentrantLock

본 절은 자바의 ReentrantLock 기반 동기화 구조에 대해 10회 반복 실험에서 관측된 반복별 원본 데이터를 표로 제공한다.
ReentrantLock은 모니터 락 대비 공정성 설정, 락 획득 시도 타임아웃 등 세밀한 제어가 가능하지만, 비공정 모드/공정 모드, 락 대기 큐 관리에 따라 처리량·응답시간의 변동성과 경합 분산 효과가 어떻게 달라지는지 반복별 데이터를 통해 실증한다.
대표 반복의 상세 이벤트 로그, 락 대기열 상태, 공정/비공정 설정에 따른 극단값/이상값(Outlier) 등 ReentrantLock 구조의 동작 특성과 실효성을 구체적으로 입증한다.

[표 – 반복별 계측 데이터]

반복	처리량(ops)	평균 응답(ms)	race 발생(건)	permit 누수(건)	상태 불일치(건)
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					

[대표 반복 상세 로그/상태 변화(캡처/설명)]

1. 반복 N회 시점 [] 문제 발생

[Dump]

[설명/해설]

동시성 시나리오 테스트 및 결과 분석

세마포어(Semaphore)

본 절은 세마포어(Semaphore) 기반 동시성 제어 구조에서 10회 반복 실험으로 계측된 원본 데이터를 반복별로 표로 정리한다.
세마포어 구조는 카운트 기반의 permit 획득/반납으로 임계구역 동시 접근을 제어하므로, permit 미반납/이중 점유/복구 성공률/상태 불일치 등 실무적 위험 요소와 자동화 복구 효과가 각 반복별로 어떻게 발생·해소되는지 실제 데이터로 검증한다.
표 아래에는 permit 누수 발생/복구 이벤트, 동시 진입·퇴장 시 permit 일관성 변화, 복구 실패 반복에서의 상태 스냅샷 등 카운트 기반 구조 특유의 상태 전이와 복원력을 입증하는 로그·스냅샷이 포함된다.

[표 – 반복별 계측 데이터]

반복	처리량(ops)	평균 응답(ms)	race 발생(건)	permit 누수(건)	상태 불일치(건)
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					

[대표 반복 상세 로그/상태 변화(캡처/설명)]

1. 반복 N회 시점 [] 문제 발생

[Dump]

[설명/해설]

동시성 시나리오 테스트 및 결과 분석

동시성 제어 구조별 종합 성능·상태 비교 분석

조건 분기(if-else), 모니터 락(synchronized), ReentrantLock, 세마포어 등 4가지 동기화 구조의 10회 반복 실험 데이터를 한눈에 비교·분석하는 데 있다.
각 구조별 처리량(ops/sec), 평균 응답시간(ms), race 발생률, permit 누수율, 복구 성공률 등 주요 성능 및 신뢰성 지표를 박스플롯, 히스토그램, 라인차트 등 비교 그래프로 시각화함으로써, 단순 표 기반 비교를 넘어 구조적 변동성, 병목 구간, 에러 집중 구간 등 실무적으로 설계 선택의 결정적 근거가 되는 데이터를 입증한다.

[비교 그래프]

- 박스플롯
  구조별 처리량/응답시간 분포 비교)
- 히스토그램
  (구조별 처리량/응답시간 분포 비교)
- 다중 라인 차트
  (구조별 처리량/응답시간 분포 비교)

[그래프 해설/경향성]

- “세마포어 구조는 평균 처리량과 표준편차에서 모든 구조 중 가장 안정적이고 일관된 분포를 보인다.
- 조건 분기 구조는 특정 반복에서 race event 및 permit 누수 집중 발생으로, 분포의 하단 극단값이 도드라진다.
- ReentrantLock과 모니터 락은 평균 응답시간의 변동성, 병목 구간의 집중 발생이 명확히 시각화된다.”

[비교 결론]

- “구조별 실측 데이터를 기반으로,
- 단순 수치 요약을 넘어 ‘분포/변동성/이벤트 집중 구간/복구 성공률’ 등
- 실무 설계에 즉각 적용 가능한 결정적 근거를 확보할 수 있다.
- 실제 시스템 선택 시, race/event 집중 위험을 회피하고,
- permit 일관성·복구 안정성·응답시간 분포를 통제하려면
- 세마포어/공정 락 등 카운트 기반 동시성 제어 구조가 실질적 우위를 가진다.”

동시성 시나리오 테스트 및 결과 분석

조건 분기(If-Else) 기반 동기화

본 절은 단순 조건 분기(If-Else) 기반 임계구역 진입 구조에 대해 10회 반복 실험에서 계측된 처리량, 응답시간, race event 발생, permit 누수, 상태 불일치 등 주요 지표를 단독 그래프(박스플롯, 히스토그램, 라인차트, 이벤트 타임라인 등)로 시각화한다.
조건문 분기만으로 동시성 제어를 시도할 경우, race condition, permit 이중 점유, 상태 불일치 등 구조적 결함이 반복별로 어떤 양상으로 표출되는지를 실증적으로 해부하는 데 목적이 있다.

[비교 그래프]

- 박스플롯
  (조건 분기 기반 처리량/응답시간 분포, outlier 집중 구간 강조)
- 히스토그램
  (race event/permit 누수 분포, 에러 집중 회차 강조)
- 다중 라인 차트
  (반복별 permit 누수/복구 실패/상태 변화 추이)

[그래프 해설/경향성]

"조건 분기 기반 구조에서는 반복 2, 4, 7회차 등 특정 구간에서 race event, permit 이중 할당, 상태 불일치가 집중적으로 발생했다.
박스플롯 기준, 처리량·응답시간 분포의 하단 극단값(outlier)이 두드러지며, permit 누수율이 타 구조 대비 월등히 높아 복구 불가/상태 오염이 반복적으로 관측된다."

[구조적 결론]

- "조건문 분기만으로 동시성 제어를 시도할 경우,
- 임계구역 진입의 무결성, 상태 일관성, permit 일치성이 실무 환경에서 전혀 보장되지 않으며,
- race condition, permit 누락/이중 점유, 상태 불일치 등 본질적 구조 결함이 반복 계측에서 일관되게 노출된다."

동시성 시나리오 테스트 및 결과 분석

모니터 락(synchronized) 기반 동기화

본 절은 자바의 모니터 락(synchronized) 기반 동기화 구조에 대해 10회 반복 실험에서 계측된 처리량, 응답시간, race event, permit 누수, 상태 불일치 등 주요 지표를 단독 그래프(박스플롯, 히스토그램, 라인차트, 타임라인 등)로 시각화한다. synchronized 구조는 단일 객체 락에 모든 스레드의 임계구역 진입을 집중시키기 때문에, race condition은 발생하지 않지만, 경합·대기·병목 현상, 응답시간 변동성, 스레드 starvation 등 실무적 성능 저하가 반복별로 어떤 양상으로 발생하는지 실증적으로 해부하는 데 목적이 있다.

[비교 그래프]

- 박스플롯
  (synchronized 구조의 처리량/응답시간 분포, 병목·outlier 구간 강조)
- 히스토그램
  (응답시간/대기시간 분포, 경합 집중 회차 강조)
- 다중 라인 차트
  (반복별 락 대기시간/응답 변화, 병목 집중 구간 시각화)

[그래프 해설/경향성]

- “synchronized 기반 구조에서는 race event가 전혀 발생하지 않으나, 반복 5, 8, 10회차 등 특정 구간에서 락 경합에 따른 대기시간/응답시간 급증, 처리량 감소, 병목(outlier) 발생이 뚜렷하다.
- 박스플롯, 히스토그램 기준으로, 구조적 경합 집중 구간이 일관되게 관측되며, permit 일관성은 보장되나, 스레드 starvation·응답 변동성 등 성능 저하가 반복적으로 드러난다.”

[구조적 결론]

- “모니터 락 기반 동기화 구조는 임계구역 무결성과 상태 일관성은 보장하지만,
- 대량 동시 요청·긴급 자원 점유 환경에서는 경합에 따른 병목, 대기 구간, 성능 저하가 실험 반복별로 불가피하게 발생하며,
- 이는 실무 시스템에서 처리량·응답시간의 예측 불가성과 스케일링 한계로 귀결된다.”

동시성 시나리오 테스트 및 결과 분석

ReentrantLock 기반 동기화

본 절은 자바의 ReentrantLock 기반 동기화 구조에 대해 10회 반복 실험에서 계측된 처리량, 응답시간, 락 경합, race event, permit 누수, 상태 불일치 등 주요 지표를 단독 그래프(박스플롯, 히스토그램, 라인차트, 타임라인 등)로 시각화한다. ReentrantLock은 모니터 락 대비 공정성(Fair/NonFair) 설정, 락 획득 대기 시도, 락 대기열 모니터링 등 세밀한 제어가 가능하지만, 공정/비공정 설정, 경쟁 큐 구조, 반복별 경합·병목·변동성 등이 실험 데이터에서 어떻게 드러나는지 구조적으로 해부하는 데 목적이 있다.

[비교 그래프]

- 박스플롯
  (ReentrantLock 구조의 처리량/응답시간 분포, 공정/비공정 outlier 차이 강조)
- 히스토그램
  (락 대기/경합, 응답시간 분포, 반복별 변동성 강조)
- 다중 라인 차트
  (반복별 락 대기/해제 이벤트, 병목/성공/실패 흐름)

[그래프 해설/경향성]

- “ReentrantLock 기반 구조에서는 공정성 모드에 따라 락 획득/경합 패턴, 대기시간, 처리량 변동성이 크게 다르게 나타난다.
- 반복 1, 6, 9회차 등에서 락 경합에 의한 응답시간 급등, 처리량 저하, 락 starvation 또는 병목(outlier) 구간이 분명히 관측되었으며, 락 대기열/락 해제 타임라인에서 경쟁 스레드 우선순위 전환 효과도 확인된다.”

[구조적 결론]

- “ReentrantLock 구조는 세밀한 동시성 제어가 가능하지만,
- 락 대기열, 공정성 설정, 스레드 스케줄링 등에 따라 실무적 처리량/응답시간의 예측 불가성과 특정 반복에서의 병목/락 starvation 위험이 상존한다.
- 따라서 구조 설계 시, 락 경합 패턴과 스케줄링 정책을 반드시 실험·검증해야 한다.”

동시성 시나리오 테스트 및 결과 분석

세마포어(Semaphore) 기반 동기화

본 절은 세마포어(Semaphore) 기반 동기화 구조에 대해 10회 반복 실험에서 계측된 처리량, 응답시간, permit 획득/반납, race event, permit 누수/이중 할당, 복구 성공/실패 등 주요 지표를 단독 그래프(박스플롯, 히스토그램, 라인차트, 이벤트 타임라인 등)로 시각화한다.
세마포어는 카운트 기반 동시성 통제 구조로, 임계구역 동시 진입 허용 및 자원 상태 추적이 가능하지만, permit 미반납, 이중 점유, 복구 자동화 등 실무적 설계 위험 및 복원력의 차이가 실험 반복별로 어떻게 발생·해소되는지 실증적으로 해부하는 데 목적이 있다.

[비교 그래프]

- 박스플롯
  (세마포어 구조의 처리량/응답시간 분포, permit 일관성 강조)
- 히스토그램
  (permit 누수/복구율 분포, permit 불일치/자동 복구 이벤트 강조)
- 다중 라인 차트
  (반복별 permit 복구 성공률/상태 변화, 이벤트 집중 구간)

[그래프 해설/경향성]

- “세마포어 기반 구조는 전체적으로 처리량/응답시간의 안정성과 permit 일관성이 높게 유지되나, 반복 4, 7, 9회차 등 특정 구간에서 permit 미반납·이중 점유·복구 실패 이벤트가 일시적으로 발생했다.
- 박스플롯, 히스토그램 기준, 복구 자동화 로직이 정상 작동할 경우 permit 일치성이 즉시 회복됨을 확인할 수 있으며, permit 누수/불일치가 미복구 시 전체 상태 오염으로 확산될 수 있음을 데이터로 입증했다.”

[구조적 결론]

- “세마포어 기반 동기화 구조는 race condition과 permit 일관성 측면에서 가장 안정적인 통제력을 제공하지만,
- permit 미반납/이중 점유, 자동 복구 실패 등 극히 일부 반복에서만 한계가 노출된다.
- 실무적으로는 복구 자동화, 상태 모니터링, permit 일치성 검증이 반드시 결합되어야만 구조적 안정성이 장기적으로 보장된다.”

동시성 실험 주요 예외 이벤트 및 상태 이슈 보고

본 절은 실험 반복 중 실제로 발생한 race event, permit 누수, 상태 불일치, 복구 성공/실패, 병목 등 핵심 이벤트 및 상태 변동의 실제 원본 로그, DB/메모리 스냅샷, 상태 변화 이미지를 대표적으로 발췌하여 증거로 제시한다.
단순 통계·그래프가 아닌, 실제 동작 시점에 시스템 내부에서 어떤 이벤트/상태가 발생했는지를 구체적으로 캡처해 "계측 결과의 신뢰성"을 실증하는 것이 목적이다.

[대표 이벤트 로그]

■ Race event 발생 로그

※ 동시 입장 요청 충돌로 인한 permit 이중 점유 및 상태 오염

■ Permit 누수/이중 할당 로그

※ permit 반납 누락으로 인한 세션 상태 불일치 및 자원 병목

■ 상태 불일치/DB-메모리 차이 스냅샷

※ DB와 메모리(캐시) 간 permit 상태 불일치

동시성 제어 구조 실무 적용 설계 기준

본 실험 및 구조 분석 결과를 바탕으로,실무 시스템에 동시 입장 제어 구조를 설계·반영할 때다음과 같은 구조적 선택 기준을 필수적으로 적용해야 한다.

1. Race condition이 구조적으로 차단되는가?→ 단순 코드 상의 조건 분기·임계구역 보호가 아닌,구조 내부에서 자원 점유와 상태 추적이 atomic하게 통합된 구조(예: 세마포어 permit 기반)가 적용되어야 한다.

2. 병목/성능 저하 없이 확장성이 보장되는가?→ 처리량, 응답 시간, 락 경합/임계구역 병목 등의 실측 데이터와 구조적 확장성 관점에서,구조적 병목이 없는 설계만이 실무에 적합하다.

3. 상태 정합성 및 자동 복구가 구조 내부에 내재되어 있는가?→ permit/세션/DB/메모리 상태가 항상 일치하며,미반납 permit, 세션 누락 등 비정상 상태를 스케줄러 등 내부 자동화로 복구할 수 있어야 한다.

4. 구조 자체가 테스트·모니터링·운영 자동화에 적합한가?→ 실시간 상태 스냅샷, 이벤트 기반 로그, 운영 중 재현 가능한 구조 설계가 가능해야 하며,확장/운영 자동화/알림/장애 복구 등 실무적 운용이 용이한 구조여야 한다.

5. 핵심 통제 단위가 구조 내부에 응집되어 있는가?→ 코드 외부에서 임시 상태 관리, 조건 분기, 임계구역 분할 등으로일관성을 유지하려는 구조는 실무 적용에서 배제되어야 한다.

미 실증/심층 계측 필요 과제 리스트

이 항목은, 본 보고서 작성·실험 과정에서"표면적 이론, 공식 문서, 강의, 간접 자료 등으로만 존재/구조를 인지하고 있으며 직접 코드 테스트, 상태 계측, 실험 기반 검증은 이루어지지 않았다"는 항목들이며,향후에 JVM/OS/동기화 내부 구조, AQS 큐 구조, 디버깅, 상태 스냅샷 계측 등을 체계적으로 학습·관찰할 계획이다.

[직접 실험/계측/증명이 이루어지지 않은 영역]

- JVM ObjectMonitor의 내부 필드(예: _owner, succ, EntryList, WaitSet 등) 및 각 필드의 상태 전이 구조
- monitorenter/monitorexit opcode의 진입/퇴장 시점 및 JVM 내부 락 점유·해제 흐름
- WaitSet/EntryList 큐의 실제 동작 방식(스케줄링/경쟁 정책, succ 후보 선정 기준 포함)
- AQS(AbstractQueuedSynchronizer) 내부 대기 큐 구조 및 ConditionObject의 큐 관리, 신호 전이 로직
- 공정성/비공정성 설정에 따른 AQS, Semaphore, ReentrantLock의 실제 동작 변화 및 race 발생 패턴의 상세 비교
- OS 수준 park/unpark, JVM과 OS 스케줄러의 상호작용 및 스레드 상태 변화

실험 증거 자료 및 재현 리소스

전체 계측/원본 로그/스냅샷/실험 자동화 코드

본 절은 모든 실험·계측·시각화에서 사용한 원본 데이터, 반복별 로그, DB/메모리 상태 dump, 주요 스크린샷, 실험 자동화/테스트 스크립트 등 "증명 근거 전체"를 부록으로 첨부한다.
문서 본문에 인용된 표·그래프·로그·스냅샷의 신뢰성·재현성·검증 가능성을 보장하기 위한 실질적 근거로 활용된다.
각 파일/자료별로 내용/포맷/활용 방법을 간단히 안내하며, 필요시 실험 재현·추가 검증·타 환경 이식 등에 즉시 사용할 수 있도록 한다.

[첨부 자료]

■ 전체 계측 원본 데이터

- /data/experiment_result_raw.csv

10회 반복 전체 처리량/응답/에러/복구 등 실험 원본 데이터

■ 반복별 로그 파일

- /logs/repeat_1.log ~ /logs/repeat_10.log

10회 반복 전체 처리량/응답/에러/복구 등 실험 원본 데이터

■ 상태 dump/스냅샷

- /dump/db_state_3rd_repeat.txt
- /dump/redis_permit_dump.json

DB/Redis/메모리 permit, 상태 변화 원본

■ 시각화 원본 이미지

- /img/boxplot_ops.png
- /img/timeline_race_event.png

문서 본문에 인용된 모든 그래프/스냅샷의 실제 원본 이미지

■ 실험 자동화 코드/스크립트

- /src/experiment_runner.py
- /src/collect_state_dump.sh
- /src/test_jmeter.jmx

10회 반복 전체 처리량/응답/에러/복구 등 실험 원본 데이터

감사합니다!