Java/Spring 환경의 동시성 문제 탐지 및 제어 기법별 성능 분석

Race Condition을 증명하고, 4가지 제어 기법의 성능 트레이드오프를 나노초 단위로 시각화

https://github.com/bumjinDev/ChatService

동시성 시나리오 테스트 및 결과 분석

동시성 제어 구조별 종합 성능·상태 비교 분석

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.”

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, race condition 발생 구간의 실시간 검출, 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

공통 계측·분석 기준

구분	계측 항목	정의	집계·측정 방식	분석 및 활용 관점
데이터 집계	처리량 (ops/sec)	단위 시간 내 처리 완료 건수	자동 스크립트·JMeter 통계 집계	구조별 성능 비교, 시스템 최대 처리량 검증
	응답 시간	요청~응답 완료까지 전체 소요 시간	JMeter/클라이언트 자동 로그	사용자 체감 성능, End-to-End 성능 변화
	대기 시간	임계구역 진입 대기 구간 소요 시간	서버 로그/스크립트 계측	경합/락 경쟁 병목 검증, 구조별 차이
	임계구역 처리 시간	임계구역 점유~처리 완료까지 소요 시간	서버 로그/스크립트 계측	락 점유/병목 구간, 구조적 효율성 평가
이벤트·상태 집계	race condition	허용 인원 초과/상태 불일치 등 구조적으로 문제 발생 건수/비율	상태 snapshot, 자동 로그	구조적 결함, 동시성 오류, race 예방 효과
반복성/재현성	10회 반복 통계 집계	각 구조별·조건별 10회 이상 반복·통계적 신뢰성 확보	자동 반복, 독립 로그 파일 저장	통계적 유의성, 실험 신뢰도
시각화	분포·변동 분석	표, 히스토그램, boxplot 등 다양한 시각화	matplotlib, pandas, Excel 등	구조별 처리량/병목/이상값 등 시계열·분포 분석
결과 검증/증적	증적 파일/로그	실험 결과/스냅샷/자동 로그 등 결과물 증적화	자동 파일·로그 저장, snapshot	결과 재현, 보고서·포트폴리오 근거 제공

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: 200명 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 로그를 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: 200명 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)

3. ReentrantLock 기반 구조

[실험 시나리오 및 계측 플로우]

- Step 1: 200명 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 5: 이벤트 로그 자동 저장, 10회 반복 통계 집계

4. 세마포어 기반 구조 (Semaphore)

[실험 시나리오 및 계측 플로우]

- Step 1: 200명 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)	이벤트 로그, race condition 유무	반복적 불일치/장애(중복 입장, 인원 초과, 데이터 불일치) 실증
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

동시성 시나리오 테스트 및 결과 분석

테스트 환경 정의

1. 테스트 환경

구분	도구/프레임워크	용도 및 목적
부하 시뮬레이션	JMeter	동시 입장/요청/퇴장 등 대량 시나리오 자동 반복 및 응답 계측
서버 로그 계측	Log4j2 (Spring Boot)	race 발생 등 주요 이벤트 자동 로깅
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

2. 계측 지표 정의 및 집계 기준

구분	계측 지표	정의/설명	집계·측정 방식	분석/활용 관점
서버 측	race condition 발생률	실험 중 상태 불일치, 중복 갱신 등 race 이벤트 비율	서버 로그/자동 스크립트	동시성 구조 결함 검증 및 경합 예방 효과 검증
	정원 초과 진입 비율 (over-capacity)	최대 인원 초과 시도 건수 및 전체 대비 비율	서버 로그/자동 스크립트	구조별 경합/병목 현상 계량, 정책 효과 검증
	대기 시간 (waitingMillis)	임계구역 진입 대기 구간 소요 시간	서버 로그/스크립트 통계	구조별 경합/락 병목 분석, 동시성 개선 효과
	임계구역 처리 시간 (criticalElapsed)	임계구역 점유~처리 완료까지 소요 시간	서버 로그/스크립트 통계	락 점유/병목 구간 분석, 구조적 효율성 평가
클라이언트 측	응답 시간 (Response Time)	클라이언트 요청~응답 완료까지 전체 소요 시간 (평균/중앙값/p99 등)	JMeter 등 부하 시뮬레이터 자동 집계	End-to-End 성능, 사용자 체감 품질 변화
	처리량 (Throughput, ops/sec)	단위 시간(초)당 성공적으로 처리된 요청 건수	JMeter 등 부하 시뮬레이터 자동 집계	시스템 성능 한계 및 구조별 최대 처리량 비교

동시성 시나리오 테스트 및 결과 분석

전체 요약 보고서 - 서버 지표 : RaceCondtion 지표

본 절의 목적은 다중 스레드 요청 환경에서 서버의 각기 다른 동기화 구조(조건 분기, 뮤텍스 락, 세마포어)가 데이터 정합성과 시스템 안정성을 어떻게 보장하는지 실증적으로 분석하는 것이다. 10회 반복 계측을 통해 각 구조의 Race Condition 발생률, 성능 지표로써 이론이 아닌 실측 데이터에 기반하여, 어떤 아키텍처가 서버의 내부 상태를 가장 안정적으로 유지하고 시스템 부하를 효율적으로 처리하는지 증명한다.

[각 계측 항목 별 10회 테스트 수행 결과 표 - RaceCondtion 규칙 1 : 값 불일치]

계측 항목	조건 분기 기반	뮤텍스 락 기반	ReentrantLock	세마포어 기반
전체 요청 수	2000	2000	2000	2000
값 불일치 발생 횟수 / 비율(%)	127 / 6.35 %			
값 불일치 누적 총합 값	168			
값 불일치 누적 최대 값	5			
값 불일치 누적 최소 값	1			
값 불일치 누적 평균 값	1.32			

[각 계측 항목 별 10회 테스트 수행 결과 표 - RaceCondtion 규칙 2 : 경합 발생]

계측 항목	조건 분기 기반	뮤텍스 락 기반	ReentrantLock	세마포어 기반
전체 요청 수	2000	2000	2000	2000
경합 발생 스레드 수 / 발생률(%)	1839 / 91.95			
경합 발생 총합 값	9618			
경합 발생 최대 값	18			
경합 발생 최소 값	2			
경합 발생 평균 값	5.23			

[각 계측 항목 별 10회 테스트 수행 결과 표 - RaceCondtion 규칙 3 : 정원 초과 오류]

계측 항목	조건 분기 기반	뮤텍스 락 기반	ReentrantLock	세마포어 기반
전체 요청 수	2000	2000	2000	2000
정원 초과 입장 건수 / 비율 (%)	1 / 0.05			
정원 초과 최대 값	1			
정원 초과 최소 값	1			
정원 초과 평균 값	1			

[각 계측 항목 별 10회 테스트 수행 결과 표 - RaceCondtion 규칙 4 : 상태 전이 오류]

계측 항목	조건 분기 기반	뮤텍스 락 기반	ReentrantLock	세마포어 기반
전체 요청 수	2000	2000	2000	2000
상태 전이 발생 횟수 / 비율(%)	183 / 9.15			
상태 전이 누적 총 값	242			
상태 전이 누적 최대 값	4			
상태 전이 누적 최소 값	-3			
상태 전이 누적 평균 값	1.32			

동시성 시나리오 테스트 및 결과 분석

전체 요약 보고서 - 서버 지표 : 임계구역 대기 시간과 임계구역 수행 시간

본 절의 목적은 다중 스레드 요청 환경에서 서버의 각기 다른 동기화 구조(조건 분기, 뮤텍스 락, 세마포어)가 데이터 정합성과 시스템 안정성을 어떻게 보장하는지 실증적으로 분석하는 것이다. 10회 반복 계측을 통해 각 구조 별 서버의 핵심 신뢰성 지표를 정량적으로 평가한다. 이를 통해 이론이 아닌 실측 데이터에 기반하여, 어떤 아키텍처가 서버의 내부 상태를 가장 안정적으로 유지하고 시스템 부하를 효율적으로 처리하는지 증명하여, 견고한 백엔드 설계를 위한 기술적 타당성을 확보한다.

[각 계측 항목 별 10회 테스트 수행 결과 표]

계측 항목	조건 분기 기반	뮤텍스 락 기반	ReentrantLock	세마포어 기반
전체 요청 수	2000			
방 입장 성공 수 / 비율 (%)	991 / 49.55			
방 입장 실패 수 / 비율 (%)	1009 / 50.45			
임계구역 진입 실패 수 / 비율 (%)	0 / 0.00			
방 입장 성공 대기시간 평균 (μs)	48.4			
방 입장 성공 대기시간 최대 (μs)	1,609.1			
방 입장 성공 대기시간 중간 (μs)	22.0			
방 입장 실패 대기시간 평균 (μs)	83.2			
방 입장 실패 대기시간 최대 (μs)	7,061.0			
방 입장 실패 대기시간 중간 (μs)	20.8			
방 입장 성공 임계구간 실행 시간 평균 (μs)	140.6			
방 입장 성공 임계구간 실행 시간 최대 (μs)	4,073.0			
방 입장 성공 임계구간 실행 시간 중간 (μs)	54.1			
방 입장 실패 임계구간 실행 시간 평균 (μs)	145.6			
방 입장 실패 임계구간 실행 시간 최대 (μs)	8,148.7			
방 입장 실패 임계구간 실행 시간 중간 (μs)	26.8			

동시성 시나리오 테스트 및 결과 분석

전체 요약 보고서 - 클라이언트 지표

본 절의 목적은 서버의 다양한 동시성 제어 구조(조건 분기, 뮤텍스 락, 세마포어)가 클라이언트의 사용자 경험에 어떤 영향을 미치는지 실증적으로 측정하는 것이다. 10회 반복 테스트를 통해 클라이언트가 체감하는 응답 시간(Latency), 처리량(Throughput), 요청 성공률 등 핵심 성능 지표를 집계·분석한다. 이를 통해 단순 서버의 내부 로직을 넘어, 어떤 백엔드 구조가 최종 사용자에게 가장 빠르고 안정적인 서비스를 제공하는지 데이터에 기반하여 증명하고, 최적의 사용자 경험을 위한 아키텍처 선택의 근거를 마련한다.

[각 계측 항목 별 10회 테스트 수행 결과 표 - 클라이언트 지표 : Jmeter 테스트]

계측 항목	조건 분기 기반	뮤텍스 락 기반	ReentrantLock	세마포어 기반
표본 수	2000	2000	2000	2000
평균 값 (ms)	200			
중간 값 (ms)	265			
99% 선 (ms)	305			
최소 값 (ms)	75			
최대 값 (ms)	306			
오류 %	0.00			

[JMeter 핵심 지표별 의미]

- 표본 수 : 특정 요청을 서버에 보낸 총 횟수.
- 평균 : 모든 요청의 응답 시간을 산술 평균한 값으로, 전체적인 성능 수준.
- 중앙값 : 응답 시간을 순서대로 나열했을 때 중앙에 위치하는 값으로 극단적인 이상치(outlier)의 영향을 덜 받아 일반적인 사용자의 경험을 더 잘 반영함.
- 99% : 각각 전체 요청의 99% 이 시간 안에 처리되었음을 의미함.
- 최소/최대 : 기록된 가장 빠른 응답 시간과 가장 느린 응답 시간.
- 오류 % : 전체 요청 중 실패한 요청의 비율로, 서비스의 안정성을 나타내는 직접적인 지표.

동시성 시나리오 테스트 및 결과 분석

'Check-Then-Act' 구조의 동시성 결함 실증 - RaceCondtion 규칙 1 : 값 불일치

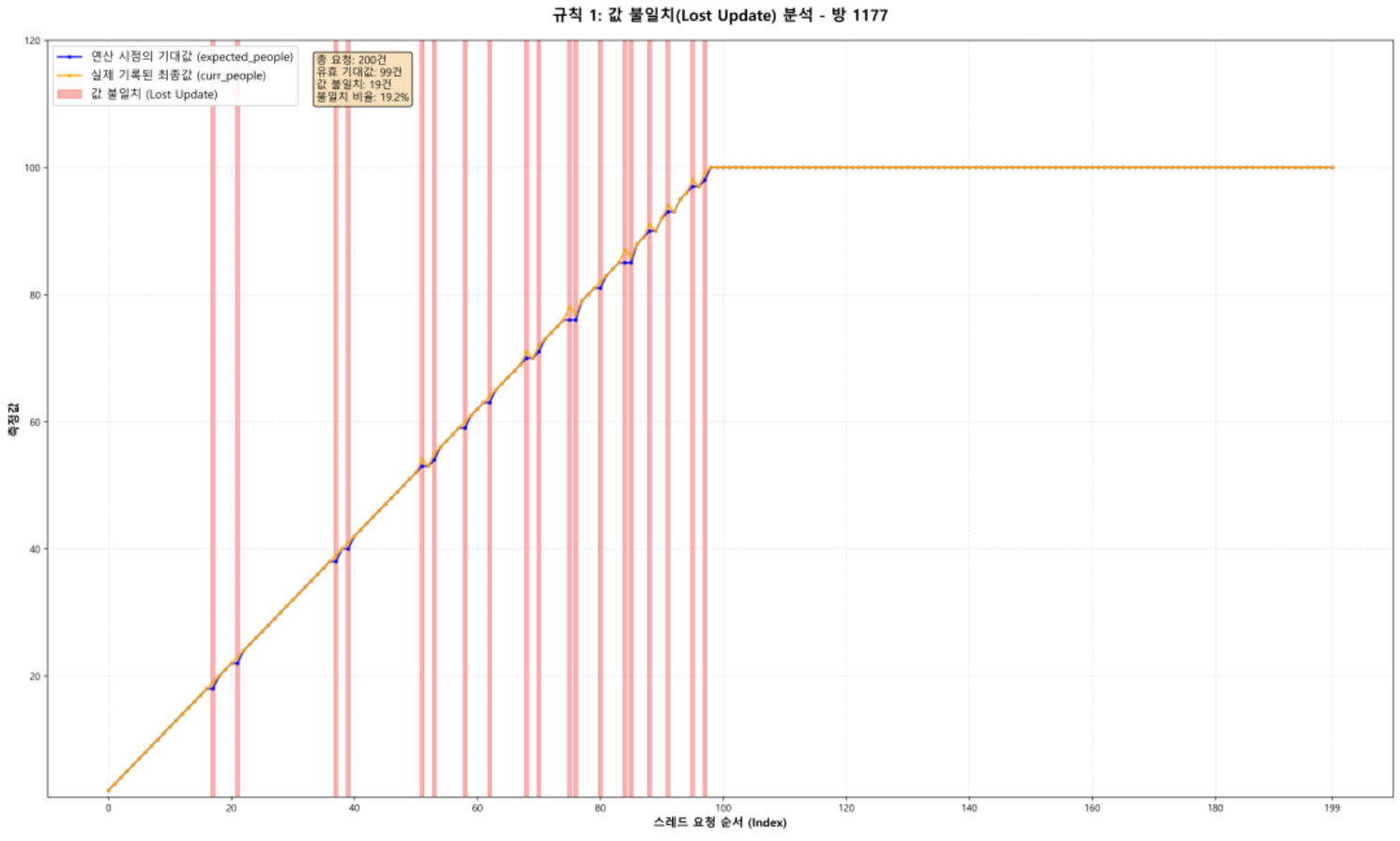
조건 분기(If-Else) 구조는 동시성 환경에서 데이터 정합성을 보장할 수 없는 대표적인 안티패턴이다. 본 절의 목적은 이 'Check-Then-Act' 로직이 실제 부하 상황에서 어떻게, 그리고 왜 실패하는지를 정량적 데이터로 증명하는 데 있으며. 이를 위해 해당 로직에 JMeter로 동시 요청 부하를 가하는 실험을 총 10회 반복 수행했으며, 이 과정에서 발생한 모든 상태 변화를 나노초 단위의 로그로 기록하였다.
아래에 제시된 데이터는 각 실험 회차에서 관측된 데이터를 통해 '값 불일치(Lost Update) 오류가 어떤 패턴과 빈도로 발생하는지, 그리고 왜 실무에서 명시적인 락(Lock) 매커니즘이 반드시 필요한지를 명확히 확인할 수 있다.

[표 – 계측 데이터]

방 번호	bin	순번	사용자 ID	이전 인원	기대인원	현재인원	오차
1177	1	18	yhjj852	17	18	19	1
1177	2	22	yhjj977	21	22	23	1
1177	4	76	yhjj847	75	76	78	2
1177	4	77	yhjj889	75	76	77	1
1177	5	81	yhjj998	80	81	82	1
1177	5	85	yhjj853	84	85	87	2
1177	5	86	yhjj999	84	85	86	1
1177	5	89	yhjj1009	89	90	91	1
1177	5	92	yhjj925	92	93	94	1
1177	5	96	yhjj870	96	97	98	1

[데이터 분석]

[차트]



[심층 분석 - 방 번호 : 1177]

동시성 제어가 없는 '읽기-수정-쓰기' 과정에서 데이터가 어떻게 오염되는지를 증명하기 위해, 1177번 방에서 발생한 '값 불일치(Lost Update)' 오류 로그를 집중 분석하였다. 분석 결과, 해당 방에서만 총 19건의 값 불일치 오류를 확인하였다. 오류로 인해 발생한 데이터의 오차 값은 대부분 '1' 수준이었으나, 후반부로 갈수록 경합이 심화되며 오차가 '2'까지 벌어지는 심각한 오류도 2건 발견되었다. 또한, 오류 발생 빈도는 시간의 흐름에 따른 테스트 구간별로 점진적인 증가 추이를 보였다. 초기 구간에서는 1건에 불과했으나, 마지막 구간에서는 8건으로 급증하여 시스템 부하와 오류 발생 간의 상관관계를 시사한다.

이러한 값 불일치 현상은 '읽기-수정-쓰기'의 전체 과정이 원자적(Atomic)으로 보장되지 않았기 때문에 발생한 직접적인 결과이다. 결론적으로, 이 분석은 동시성 제어의 부재가 데이터의 정합성을 어떻게 파괴하는지 실증적으로 보여준다. 따라서 안정적인 백엔드 시스템은 임계 구역 내의 모든 연산이 원자적으로 처리되도록 보장하는 동기화 메K커니즘이 필수적이라는 결론을 가진다.

동시성 시나리오 테스트 및 결과 분석

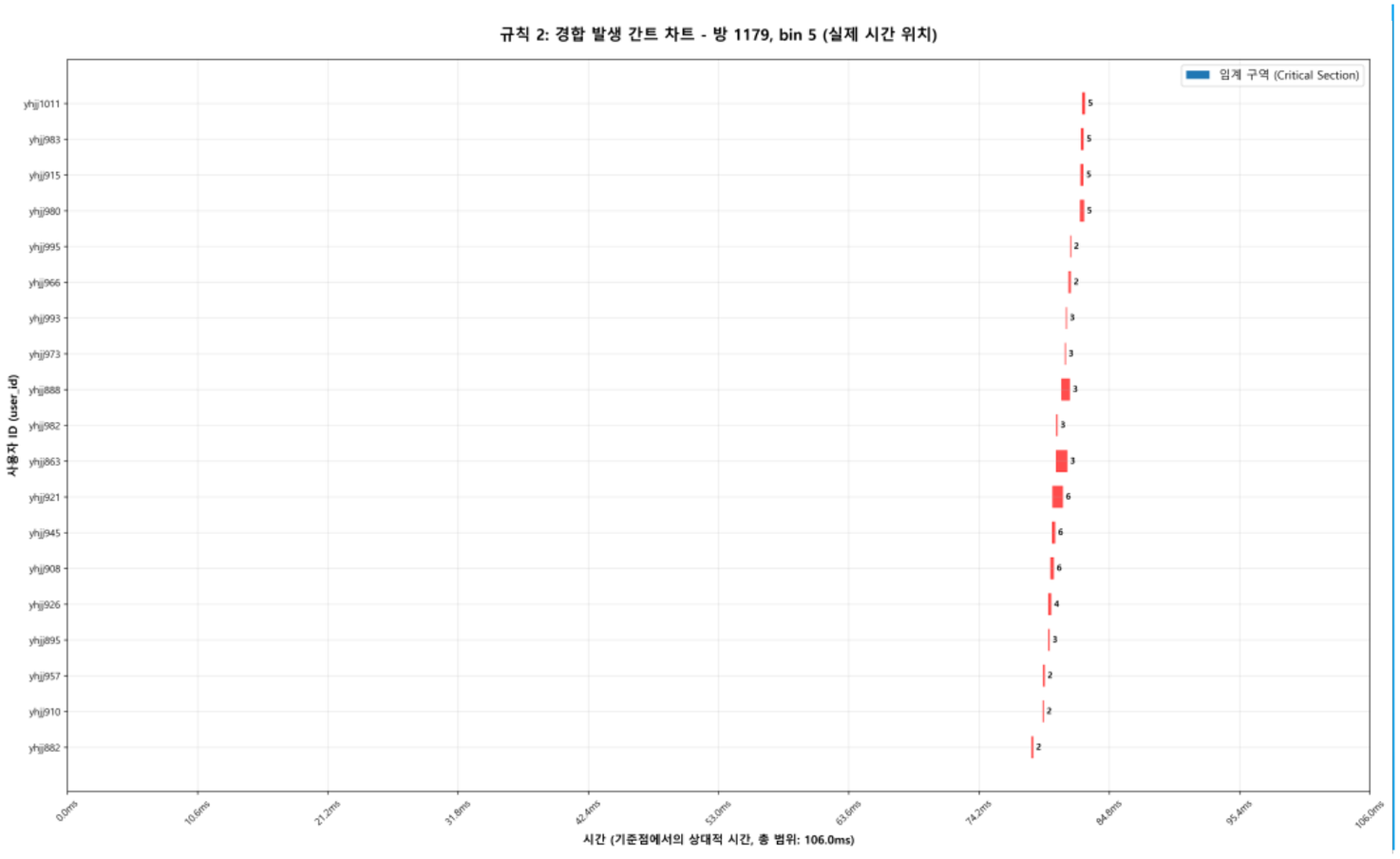
'Check-Then-Act' 구조의 동시성 결함 실증 - 규칙 2: 경합 발생(Contention) 분석

본 절은 'Check-Then-Act' 구조에서 임계 구역(Critical Section) 보호가 실패하여 여러 스레드가 동시에 같은 자원에 접근하는 '경합 발생(Contention)' 현상을 간트 차트(Gantt Chart)로 시각화하여 증명한다
차트의 각 수평 막대는 개별 스레드가 임계 구역을 점유한 시간을 나타내며, 이 막대들이 시간 축 위에서 겹치는 모든 구간이 바로 동시성 제어가 실패한 Race Condition의 발생 지점이며 이를 통해 단순히 오류 발생 여부를 넘어, 어떤 스레드들이, 언제, 얼마나 오랫동안 충돌했는지를 직관적으로 분석하고 동시성 문제의 심각도를 정량적으로 파악할 수 있습니다.

[표 – 계측 데이터]

방 번호	구간	사용자 ID	경합 발생 스레드 개수	경합 발생 사용자 ID 들	임계구역 시작 지점 시각	임계구역 종료 시점 시각	임계구역 유지 시간	선별 사유
1179	1	yhjj828	2	yhjj839, yhjj828	0.000ms	0.152ms	152.0ns	구간 내 임계구역 최대 지속 시간
1179	1	yhjj839	2	yhjj839, yhjj828	0.019ms	0.061ms	42.0ns	구간 내 임계구역 최소 지속 시간
1179	2	yhjj925	2	yhjj856, yhjj925	41.535ms	41.640ms	104.4ns	구간 내 임계구역 최대 지속 시간
1179	2	yhjj856	2	yhjj856, yhjj925	41.549ms	41.622ms	73.4ns	구간 내 임계구역 최소 지속 시간
1179	3	yhjj970	2	yhjj907, yhjj970	64.815ms	65.072ms	256.5ns	구간 내 임계구역 최대 지속 시간
1179	3	yhjj907	2	yhjj907, yhjj970	64.940ms	64.985ms	44.5ns	구간 내 임계구역 최소 지속 시간
1179	4	yhjj1017	3	yhjj1012, yhjj1017, yhjj845	76.837ms	76.986ms	149.1ns	구간 내 임계구역 최대 지속 시간
1179	4	yhjj845	3	yhjj1012, yhjj1017, yhjj845	76.850ms	76.917ms	67.1ns	구간 내 임계구역 최소 지속 시간
1179	5	yhjj945	6	yhjj921, yhjj908, yhjj945, yhjj863, yhjj982, yhjj888	80.145ms	80.381ms	236.2ns	구간 내 임계구역 최대 지속 시간
1179	5	yhjj921	6	yhjj888, yhjj921, yhjj863, yhjj973, yhjj993, yhjj966	80.189ms	81.005ms	816.1ns	구간 내 임계구역 최소 지속 시간

[데이터 분석]



[심층 분석 : 구간 5]

동시성 제어가 없는 'Check-Then-Act' 구조에서 자원 경합이 필연적으로 발생하는지를 증명하기 위해 방 번호 1179의 로그 데이터를 심층 분석하였다. 분석 결과, 총 116건의 임계 구역충돌이 발견되었으며, 평균 2.94개의 스레드가 동일 자원을 놓고 경쟁하였다. 특히 가장 심각한 경우에는 최대 7개의 스레드가 동시에 임계 구역에 진입하는 등, 시스템이 잠재적 데이터 오염 위험에 직접적으로 노출되었음을 정량적으로 확인하였다.

이는 '자원 확인(Check)'과 '상태 변경(Act)' 로직이 원자적으로 실행되지 않아 발생한 직접적인 결과이다. 결론적으로, 경합의 발생은 그 자체로 'Lost Update'나 '정원 초과'와 같은 심각한 데이터 정합성 문제의 전조 증상임을 의미하며 이 분석을 통해 안정적인 백엔드 시스템을 설계하기 위해서는 모든 임계 구역을 식별하고, Lock이나 Semaphore 등의 동기화 메커니즘을 통해 자원 접근을 반드시 제어해야 한다는 결론을 가진다.

동시성 시나리오 테스트 및 결과 분석

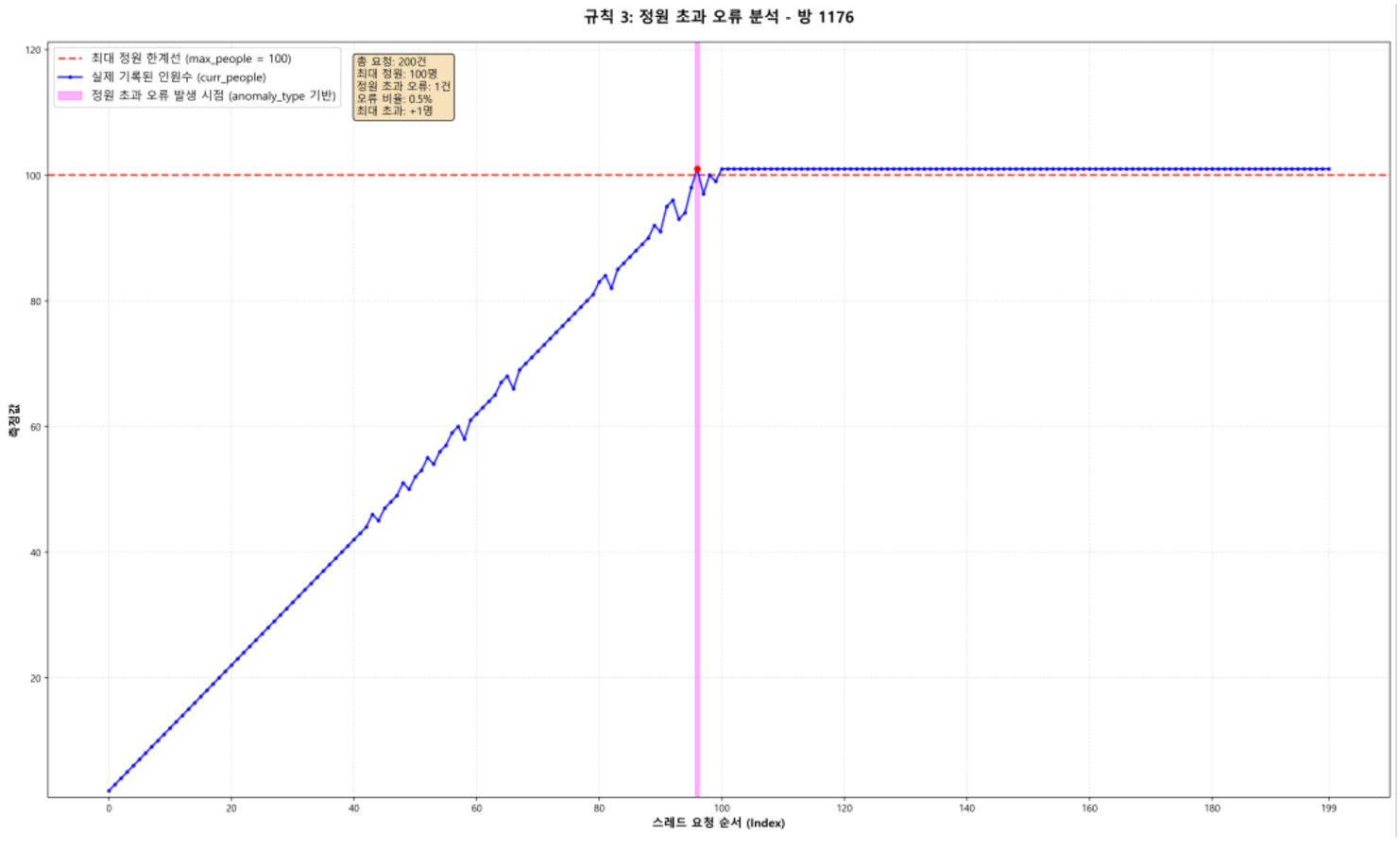
'Check-Then-Act' 구조의 동시성 결함 실증 - 규칙 3 : 정원 초과 오류

본 절은 'Check-Then-Act' 구조의 동시성 결함이 어떻게 시스템의 비즈니스 규칙(Business Rule)을 파괴하는지를 '정원 초과' 현상을 통해 실증적으로 분석한다. 시각화 차트는 시간의 흐름에 따른 '실제 기록 인원수'의 변화와, 시스템이 허용하는 '최대 정원 한계선'을 함께 표시하여 두 데이터의 관계를 비교한다. 차트의 '최대 정원 한계선'을 '실제 기록 인원수'가 초과하는 모든 지점이 바로 동시성 제어 실패로 인해 비즈니스 규칙이 위반된 명백한 증거이다. 이를 통해 단순히 오류 발생 여부를 넘어, 어느 시점에서, 얼마나 많은 인원이 정원을 초과했는지 직관적으로 추적하고 시스템의 안정성 훼손 수준을 정량적으로 파악할 수 있다.

[표 – 계측 데이터]

구간	전체 요청 수	발생 건수	발생 률(%)	총 초과 인원	평균 초과 인원	최소 초과 인원	최대 초과 인원	중간값 초과 인원
1	20	0	0	0	0	0	0	0
2	20	0	0	0	0	0	0	0
3	20	0	0	0	0	0	0	0
4	20	0	0	0	0	0	0	0
5	20	1	5	1	1	1	1	1
6	20	0	0	0	0	0	0	0
7	20	0	0	0	0	0	0	0
8	20	0	0	0	0	0	0	0
9	20	0	0	0	0	0	0	0
10	20	0	0	0	0	0	0	0

[데이터 분석]



[심층 분석 : 구간 5]

동시성 제어가 없는 환경에서 '조회 후 처리(Check-Then-Act)' 로직이 어떻게 데이터 정합성을 파괴하는지 실증적으로 분석하였다.

가설은 다음과 같다.

"여러 스레드가 동시에 '방에 여유가 있는지 확인(Check)'하고 '입장(Act)'한다면, 정원(Max People)을 초과하는 심각한 비즈니스 규칙 위반이 발생할 것이다."

[분석 과정 및 결과]

각기 다른 채팅방(ID: 1~10)에 20개의 동시 입장 요청을 보내는 테스트를 설계하고, 그 결과를 집계했다.

분석 결과, 대부분의 방에서는 오류가 발생하지 않았으나 5번 방에서 5%의 확률(20개 요청 중 1건)로 정원 초과 오류가 발생함을 확인했습니다. 이 단 한 번의 오류가 왜, 어떻게 발생했는지 심층적으로 분석함.

동시성 시나리오 테스트 및 결과 분석

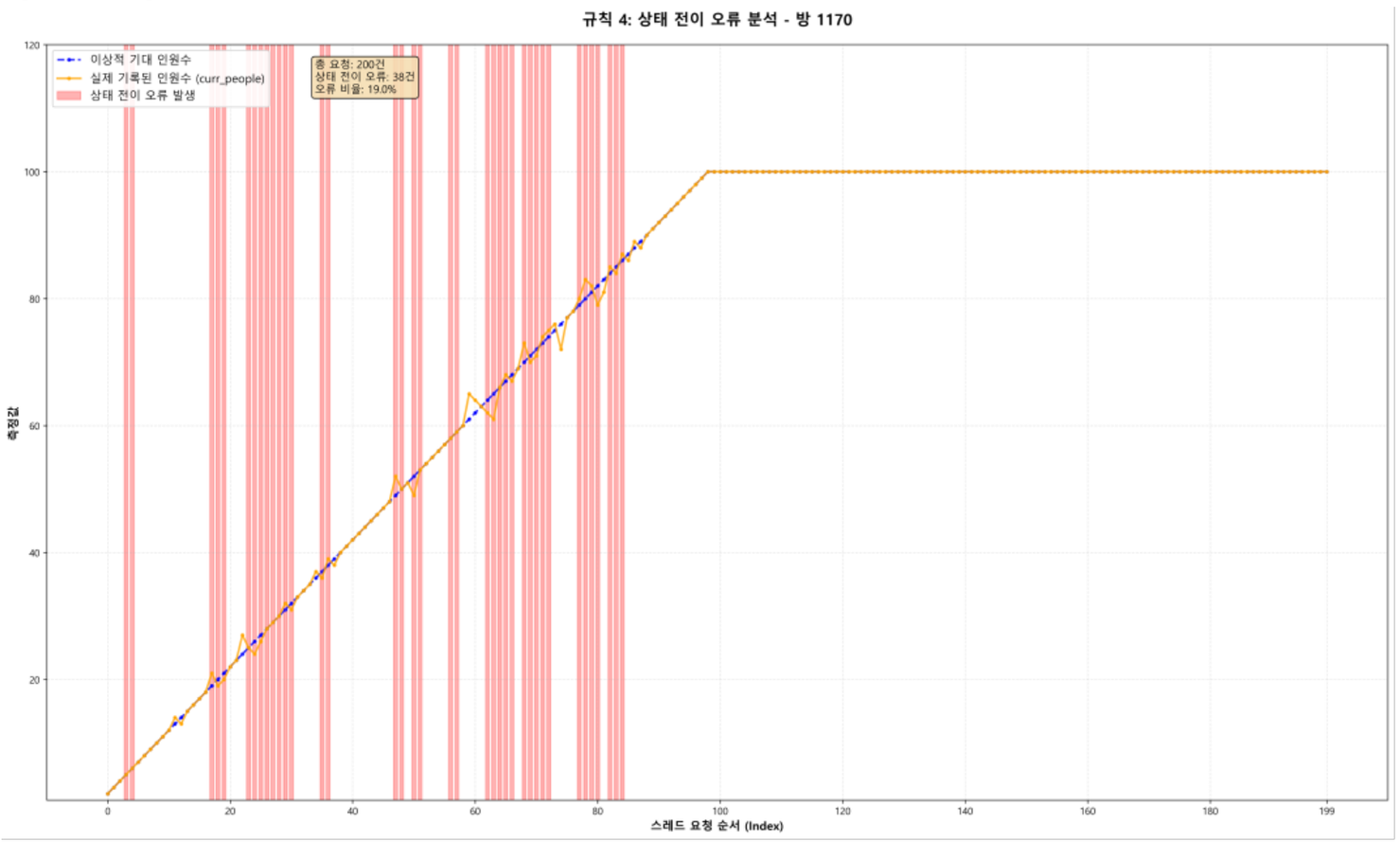
'Check-Then-Act' 구조의 동시성 결함 실증 - 규칙 4: 상대 전이 오류

본 절은 경쟁 상태가 없는 이상적인 환경에서의 데이터 증가 패턴과, 실제 부하 상황에서 기록된 데이터의 변화를 비교하여 '상태 전이 오류'를 분석한다. 시각화 차트는 시간 순서에 따라 순차적으로 데이터가 증가하는 '이상적 기대치'의 흐름과, 실제 동시성 환경에서 최종 기록된 '현실의 상태값' 변화를 하나의 그래프에 중첩하여 표현한다. 두 흐름이 서로 일치하지 않고 벌어지는 모든 지점은 '상태 전이 오류'의 증거이다. 이는 특정 스레드가 자신의 처리 순서에 맞는 최신 상태를 읽지 못하고 연산을 수행한 경우뿐만 아니라, 정상적인 최신 값을 읽었더라도 그 연산 결과를 최종 기록하기 전에 다른 스레드가 먼저 상태를 변경하여 데이터의 순차적 일관성을 훼손시킨 경우에도 발생한다. 즉, 이 분석은 '읽기-수정-쓰기'의 전체 과정이 원자적으로 보장되지 않을 때, 시스템의 상태 변화 과정 자체에 결함이 발생함을 정량적으로 증명한다.

[표 – 계측 데이터]

방 번호	구간	스레드 요청 순서	기대 인원 수	실제 인원 수	오차 수
1170	1	4	5	6	1
1170	1	18	19	21	2
1170	1	19	20	19	-1
1170	1	20	21	20	-1
1170	2	25	26	25	-1
1170	2	26	27	30	3
1170	2	27	28	32	4
1170	2	28	29	27	-2
1170	2	29	30	28	-2
1170	5	85	86	84	-2

[데이터 분석]



[심층 분석 : 방 1170 번]

본 절은 경쟁 상태가 없는 이상적 환경에서의 데이터 증가 패턴과, 실제 부하 상황에서 기록된 데이터의 변화를 비교하여 '상태 전이 오류'를 분석한다. 시각화 차트는 시간 순서에 따라 순차적으로 데이터가 증가하는 '이상적 기대치'의 흐름과, 실제 동시성 환경에서 최종 기록된 '현실의 상태값' 변화를 하나의 그래프에 중첩하여 표현한다.

두 흐름이 서로 일치하지 않고 벌어지는 모든 지점은 '상태 전이 오류'의 증거이다. 이는 개별 스레드의 '읽기-수정-쓰기' 연산이 전체적으로 원자성을 보장받지 못하기 때문에 발생하는 직접적인 결과이다. 즉, 특정 스레드가 연산을 시작하는 시점에 읽은 데이터가 이미 오래된 값이거나(Stale Read), 혹은 데이터를 읽고 쓰는 사이에 다른 스레드가 개입하여 상태를 먼저 변경함으로써(Race Condition), 최종적으로 기록되는 값이 시간적 순서에 맞지 않게 되는 것이다.

동시성 시나리오 테스트 및 결과 분석

'Check-Then-Act' 동시성 제어 기법에 따른 성능 트레이드오프 분석

백엔드 시스템의 안정성을 저해하는 경쟁 상태 문제를 데이터에 기반하여 증명하고, 이를 해결하기 위한 각 동시성 제어 기법들의 성능 특성과 트레이드오프 관계를 정량적으로 분석하였다. 동기화 제어가 없는 IF-ELSE 구조의 성능을 10개의 테스트 방에 걸쳐 측정한 결과, 단일한 실패 패턴이 아닌, 각기 다른 유형의 성능 병목 현상이 예측 불가능하게 발생하는 것을 확인하였다., 이후 진행될 모든 동기화 기법의 성능 오버헤드와 효율성을 비교하기 위한 객관적인 기준선을 수립한다.

[동시성 결함 탐지 결과 요약 - 방 별 집계]

방 번호	전체 요청 수	요청 성공률	용량 초과 실패율	요청 성공 평균 대기 시간 (μs)	요청 성공 평균 작업 시간 (μs)	용량 초과 실패 평균 대기시간 (μs)	용량 초과 평균 작업 시간 (μs)
1170	200	49.50%	50.50%	128,592	456,527	39,807	87,476
1171	200	49.50%	50.50%	44,575	108,349	41,272	106,307
1172	200	49.50%	50.50%	48,146	90,779	82,908	144,667
1173	200	49.50%	50.50%	42,421	101,077	184,333	331,229
1174	200	49.50%	50.50%	29,530	96,037	67,830	199,023
1175	200	49.50%	50.50%	34,685	99,532	84,334	103,727
1176	200	50.00%	50.00%	58,554	149,122	72,020	86,233
1177	200	49.50%	50.50%	30,469	91,804	117,658	94,314
1178	200	49.50%	50.50%	38,344	109,353	37,271	105,331
1179	200	49.50%	50.50%	28,267	103,313	103,974	197,131

위 표는 10개의 테스트 방에서 측정한 핵심 성능 지표를 요약한 것이다. 주목할 점은, '가장 성능이 나쁜 방'을 하나의 지표로 특정할 수 없다는 점이다. 특정 방은 임계 구역 진입 전 평균 대기 시간이 가장 길었던 반면, 다른 방은 임계 구역 내 평균 체류 시간이 가장 길게 나타나는 등, 각 방이 서로 다른 유형의 성능 병목을 무작위적으로 겪고 있음을 보여준다. 이는 동기화 제어가 없는 시스템의 성능이 얼마나 예측 불가능한지를 증명하는 결과이다.

[문제 구간 특정 - 방 1170]

방 번호	구간	전체 요청 수	요청 성공률	용량 초과 실패율	요청 성공 평균 대기 시간 (μs)	요청 성공 평균 작업 시간 (μs)	용량 초과 실패 평균 대기시간 (μs)	용량 초과 평균 작업 시간 (μs)
1170	1	20	100 %	0 %	150.76	505.34	0	0
1170	5	20	95 %	5 %	74.8368	265.66	17	19.2
1170	6	20	0	100%	0	0	30.255	35.28
1170	10	20	0	100%	0	0	46.255	128.28

Per_Room_Stats에서 확인된 일관된 50%의 실패율이 시간의 흐름에 따라 어떻게 형성되는지 추적하기 위해, 대표적인 방들의 구간 별 데이터를 심층 분석하였다.
분석 결과, 모든 방에서 매우 유사한 패턴이 발견되었다. 테스트 초기 구간(1~4)에서는 요청 성공률이 100%에 육박하며 안정적인 모습을 보였으나, 정확히 bin 5 시점에서 95%로 소폭 하락하며 첫 번째 실패가 발생하였다. 그리고 그 직후인 bin 6부터는 성공률이 0%로 급락하며 시스템이 완전한 '비생산적 점유' 상태로 고착되는 현상을 보였다.
이는 IF-ELSE 구조가 특정 부하 임계점을 넘어서는 순간, 급격한 성능 붕괴를 겪게 됨을 명확히 보여준다.으로 겪고 있음을 보여준다. 이는 동기화 제어가 없는 시스템의 성능이 얼마나 예측 불가능한지를 증명하는 결과이다.

[분석 결과]

제어 없는 IF-ELSE 구조의 성능 측정 결과, 시스템의 근본적인 비효율과 예측 불가능성을 증명하는 두 가지 주요 현상을 확인하였다.

첫째, 전체 요청의 약 50%가 '용량 초과'로 실패 처리되는 높은 자원 낭비율을 보였다. 이는 임계 구역에 진입했음에도 아무 작업을 수행하지 못하고 나온 '비생산적 점유'가 빈번함을 의미하며, 시스템 리소스의 절반이 잠재적으로 낭비되고 있음을 의미한다.
둘째, 성공한 요청과 실패한 요청 모두에서 성능 편차가 매우 크게 나타났다. 각 방의 구간 별 데이터를 심층 분석한 결과, '대기 시간'과 '임계 구역 체류 시간' 모두 명확한 규칙성 없이 불규칙하게 측정되었다. 이는 명시적인 락 경합이 없음에도 OS 스케줄링 등 외부 요인에 의해 시스템 성능이 무작위적으로 좌우되는, 제어 불가능한 상태임을 증명한다.

결론적으로 IF-ELSE 구조는 높은 자원 낭비율과 극심한 성능 변동성을 동시에 보여주는, 가장 비효율적이고 불안정한 방식이다. 이 측정 결과는 이후 분석할 동기화 기법들이 이러한 '낭비'와 '불확실성'을 얼마나 효과적으로 제어하는지 평가하는 핵심 기준선이 된다.

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!