5장
지표 모니터링 및 경보 시스템

한진호

지표 모니터링 및 경보 시스템

- 지표 모니터링 및 경보 시스템은 인프라의 상태를 선명하게 볼 수 있도록 하여 높은 가용성과 안정성을 달성하는 데 중추적인 역할을 한다.

- 지표 모니터링 및 경보 시스템의 의미는 회사마다 중요시 하는 시스템 우선순위에 따라 지표의 초점이 다를 수 있으므로 요구사항을 확실하게 확인하는 것이 중요하다.

문제 이해 및 설계범위 확정

개략적 요구사항 및 가정 - 대상 서버 환경

- 일간 능동 사용자 수 1억명 수준
- 서버 풀 1,000 / 풀 당 서버 수 100 / 서버당 100개의 운영 지표 수집
  - 모니터링 해야 하는 지표의 수는 천만 개 수준.
- 데이터 보관 기간은 1년
- 수집된 데이터는 7일간은 그대로 유지.
  - 7일 후에는 1분 단위로 변환하여 30일동안 유지
  - 30일 후에는 1시간 단위로 데이터를 변환해서 보관

문제 이해 및 설계범위 확정

개략적 요구사항 및 가정 - 수집 지표 요구사항

- CPU 사용률
- 요청 수
- 메모리 사용량
- 메시지 큐 내 메시지 수

문제 이해 및 설계범위 확정

비기능 요구사항

- 규모 확장성
  - 늘어나는 지표 수와 경보의 양에 맞게 확장될 수 있어야 한다.
- 낮은 응답지연
  - 대시보드와 경보를 신속하게 처리할 수 있도록, 질의에 대한 낮은 응답지연 보장
- 안정성
  - 중요 경보를 놓치지 않도록 높은 안정성 제공
- 유연성
  - 미래의 새로운 기술을 도입하여도 쉽게 통합될 수 있도록 유연한 파이프 라인 구성

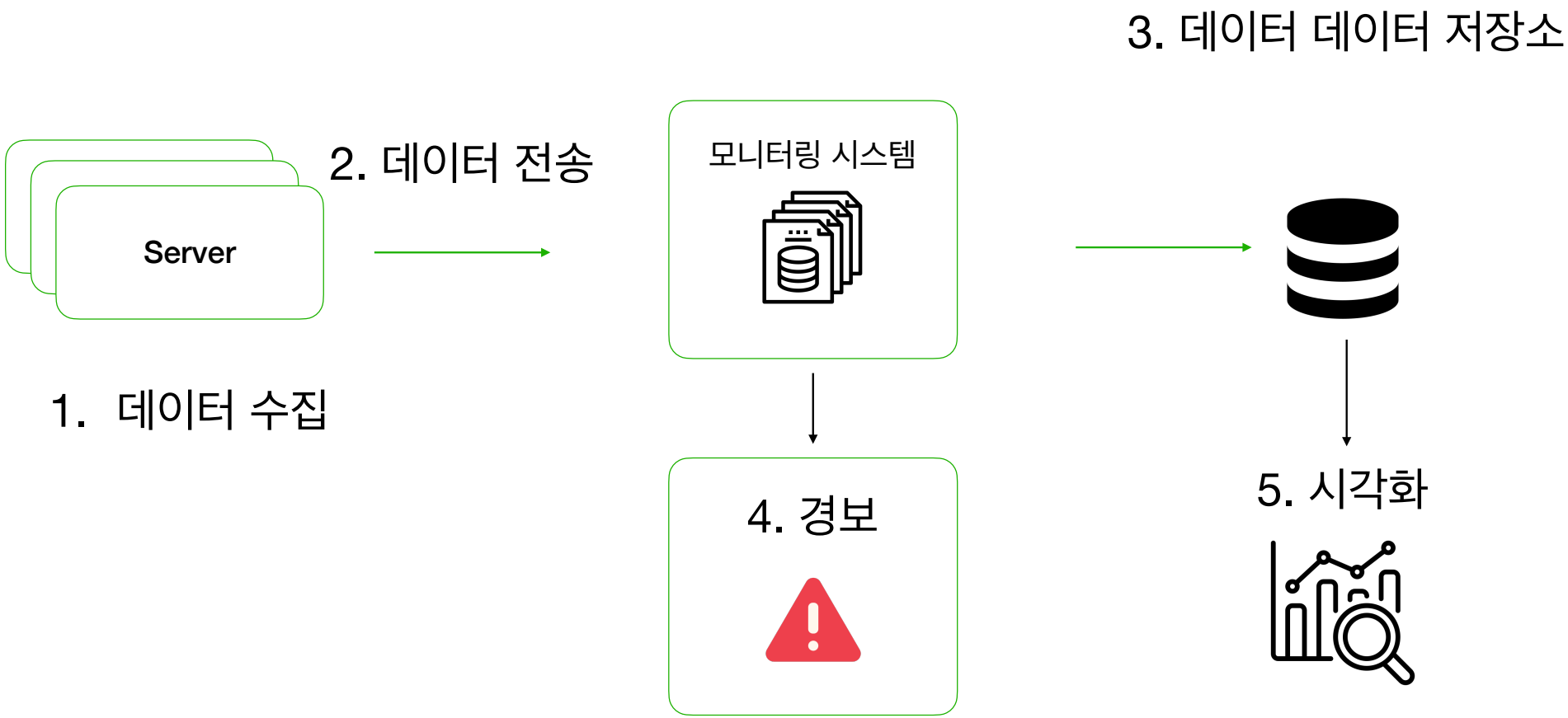
문제 이해 및 설계범위 확정

요구사항에 따른 기술 선택지

시스템	추적 가능한 지표	기본 유지기간	권장 인프라 규모	분산 추적	주요 목적/특징
Prometheus	인프라 메트릭, 애플리케이션 커스텀 메트릭	기본 15일(설정 가능), 장기 보관 불가 → Thanos/Mimir 필요	소~중 규모 (K8s와 최적)	X (OTel+Jaeger 필요)	시계열 메트릭에 최적화, 오픈소스 표준
Grafana	메트릭·로그·트레이스 **시각화 전용**	데이터 저장 안 함	모든 규모	O (연결된 백엔드 의존)	대시보드 및 알림 GUI
Datadog	메트릭, 로그, 트레이스, RUM, 프로파일링	지표 15개월, 로그 7~30일	중~대규모	O	SaaS 통합 Observability
New Relic	메트릭, 로그, 트레이스, Infra, Browser	지표 13개월, 로그 30일	중~대규모	O	Datadog과 유사한 엔터프라이즈 APM
InfluxDB	고해상도 시계열 메트릭	무한 설정 가능(스토리지 기반)	중규모 이상	X	타임시리즈 DB 전문
Graphite	시계열 메트릭	무한, 직접 스토리지 구축	소~중 규모	X	구형 시계열 시스템, Prometheus에 대체됨
Nagios	서버 상태, 프로세스, 서비스 헬스, 네트워크 장비	무한(로그 기반)	소~중 규모(대규모 부적합)	X	전통적 서버·장비 모니터링
Munin	서버 리소스 메트릭 (CPU/MEM/IO)	무한	소규모	X	레거시 환경용 간편 모니터링

개략적 설계안 제시 및 동의 구하기

기본적 사항



개략적 설계안 제시 및 동의 구하기

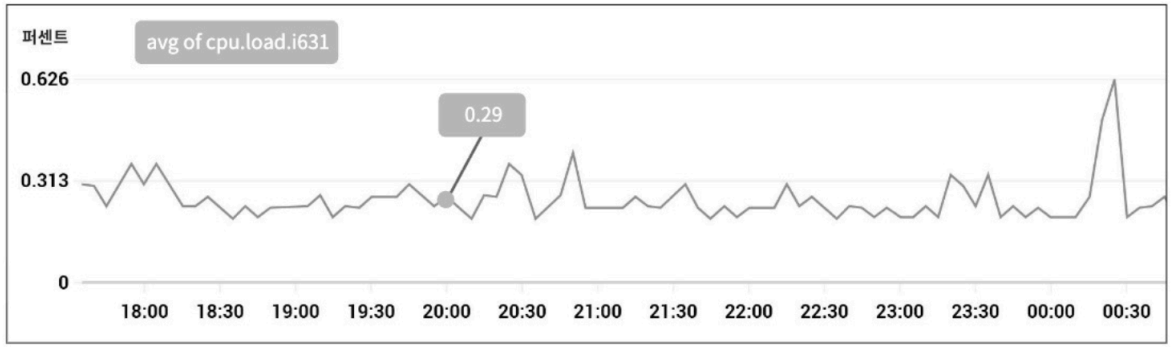
데이터 모델 - Time Series Model

- Timestamp (시간)
  - 데이터가 발생된 시간
- Metric Name (측정 지표 이름)
  - CPU Load, Memory Usage, API Latency 등 측정 대상
- Labels / Tags (차원 정보)
  - 수치가 어떤 대상/속성에 대한 것인지 나타내는 key-value 쌍 (서버, 지역, HTTP-Method 등을 기록)
- Value (수치 값)
  - 지표에 대한 값

개략적 설계안 제시 및 동의 구하기

데이터 모델 - e.g.

e.g. 책 예시



metric_name	cpu.load
labels	host:i631,env:prod
timestamp	1613707265
value	0.29

e.g. Prometheus 형식

http_request_duration_seconds_bucket

{method="GET", endpoint="/login", le="0.1"}

1699863605000 425

개략적 설계안 제시 및 동의 구하기

데이터 접근 패턴

- 앞서 언급한 “개략적 요구사항” 과 같이 매일 천만 개의 운영 지표를 기록

- Write 가 압도적으로 많음

- 매일 천만 개의 Write 부하를 감당할 수 있는 시스템을 구축해야함

개략적 설계안 제시 및 동의 구하기

데이터 저장소 시스템 선택 - RDB

- 이론적으로는 시계열 데이터 처리가 가능
- 해당 서버의 부하를 감당하려면 전문가 수준의 튜닝이 필요
- 시계열 데이터에 대한 연산에 최적화 되어있지 않음
- 데이터 베이스 튜닝에 많은 비용이 사용됨

개략적 설계안 제시 및 동의 구하기

데이터 저장소 시스템 선택 - NoSQL

- 카산드라, 빅테이블 등 범용적으로 쓰이는 NoSQL에서도 시계열 데이터 처리가 가능
  - 하지만 효율적으로 사용하기 위해선 확장이 용이한 스카마 설계 필요
  - 이는 NoSQL 내부구조에 대한 해박한 지식 필요
- 범용 NoSQL이 아닌 시계열 데이터 처리를 위한 NoSQL을 사용할 것

개략적 설계안 제시 및 동의 구하기

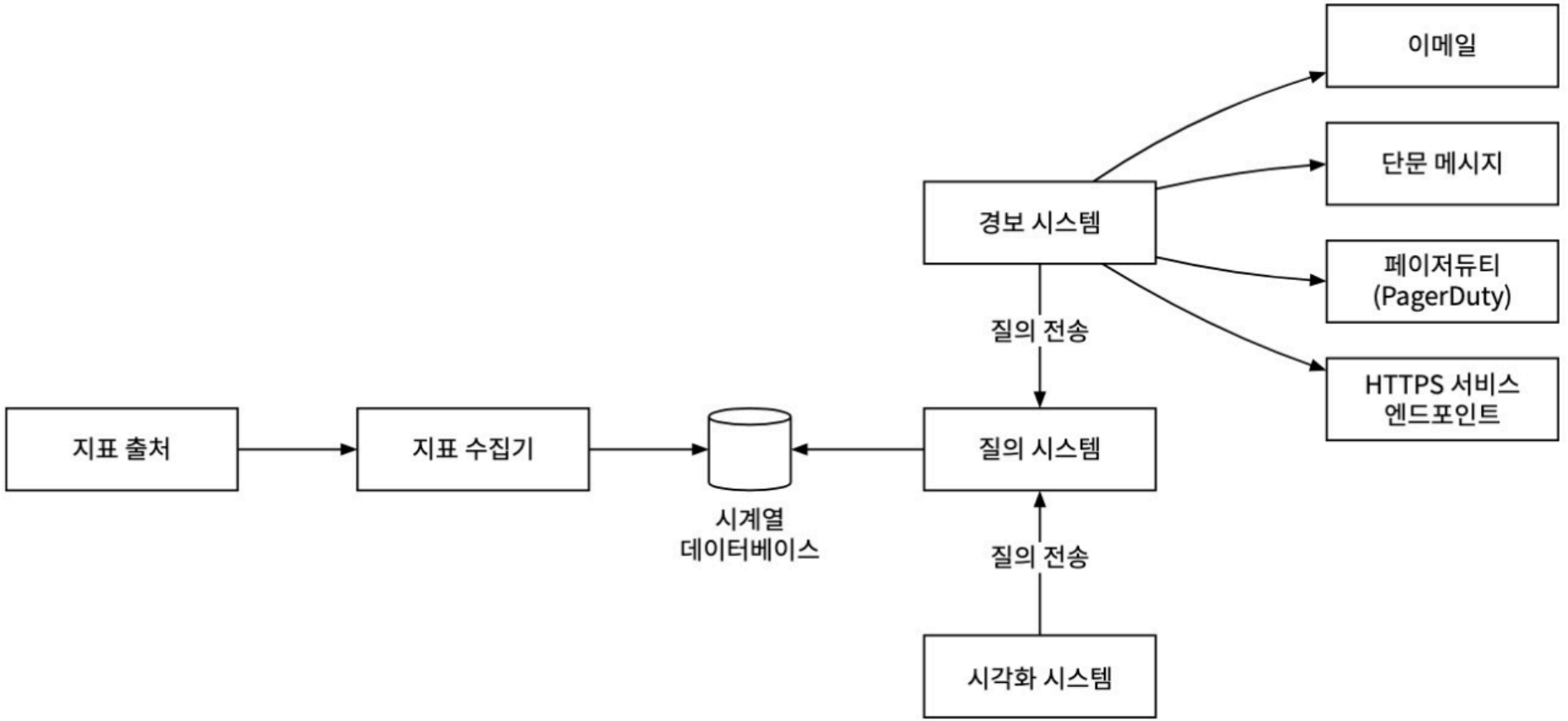
데이터 저장소 시스템 선택 - 시계열 데이터베이스

- 다량의 시계열 데이터를 저장하고 빠른 실시간 분석을 지원
- 영속성 요건과 높은 요구사항도 만족 가능
- 비교적 적은 리소스를 사용해서 효율적인 연산 처리 가능
- Prometheus, InfluxDB 등…

vCPU 또는 CPU	RAM	IOPS	초당 쓰기 연산 횟수	초당 질의 횟수	고유한 시계열 개수
2-4 코어	2-4 GB	500	< 5,000	< 5	< 100,000
4-6 코어	8-32 GB	500-1000	< 250,000	< 25	< 1,000,000
8+ 코어	32+ GB	1000+	> 250,000	> 25	> 1,000,000

개략적 설계안 제시 및 동의 구하기

개략적 설계안



상세설계

핵심 컴포넌트

- 지표 수집
- 지표 전송 파이프라인의 규모 확장
- 질의 서비스
- 저장소 계층
- 경보 시스템
- 시각화 시스템

상세설계

핵심 컴포넌트 - 지표 수집

- Pull Model VS Push Model

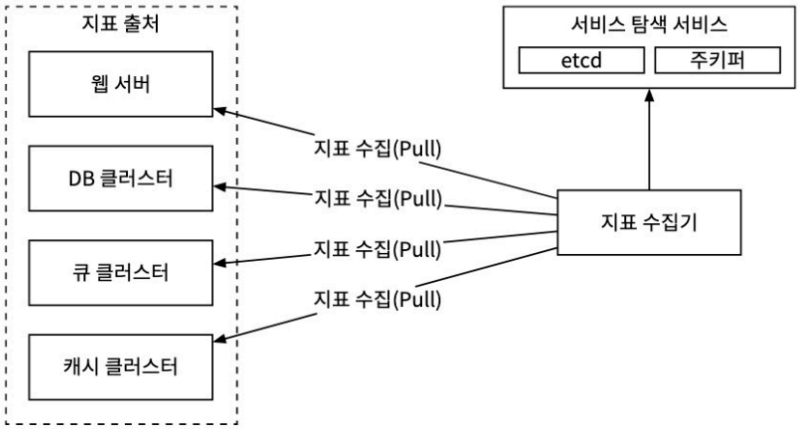


그림 5.8 풀 모델

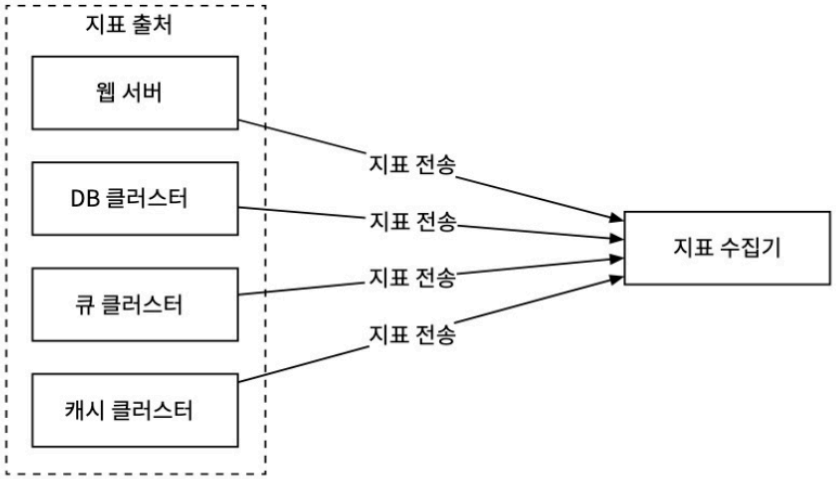


그림 5.12 푸시 모델

상세설계

핵심 컴포넌트 - 지표 수집 (Pull Model)

- 지표 수집기가 흐름의 중심이 되어 대상 서버의 메트릭을 가져온다.
- 지표 수집기가 모든 서버의 엔드포인트의 정보를 알아야한다.
- 수시로 추가/삭제되는 대규모 운영환경에는 적용하기 어려우나,
  etcd, Apache Zookeeper 등 서비스 탐색 기술을 활용해 해결 가능
- 수천 대의 서버가 만들어내는 지표 데이터를 요청하려면 지표 수집기 서버 한대로 부족
  - 지표 수집기를 클러스터링 하여 대규모 데이터를 감당 가능
  - 여러 서버가 같은 출처의 데이터를 중복해서 가져올 가능성이 있으므로, 중재 매커니즘 필요

상세설계

핵심 컴포넌트 - 지표 수집 (Push Model)

- 모니터링 대상 서버에 해당 서버의 지표를 수집/전송하는 Agent가 필요
- 각 Agent가 데이터 집계를 연산 후 전달할 수 있음
  - 집계 등의 데이터만 필요한 경우 불필요한 네트워크 부하를 줄일 수 있음
  - 지표 수집기는 오로지 수집이라는 역할 분리 가능
- 풀 모델 대비 장애의 원인을 파악하기 힘듬

상세설계

핵심 컴포넌트 - 지표 전송 파이프라인의 규모 확장

- 지표 수집기는 대규모의 데이터를 처리해야 한다.
- 안정성과 고가용성을 유지하기 위해 지표 수집기 서버는 확장 가능한 클러스터로 구축
  - 지표 수집기에 장애가 발생해도 다른 인스턴스가 수집 수행 가능
- 시계열 DB에 장애 발생 시를 대비해 카프카 도입
  - 지표를 파티션으로 나눠 병렬처리 가능
  - 상용 규모의 시스템에 모니터링을 위한 카프카를 신규 도입하는건 오버엔지니어링 가능성
- 큐 없이 대규모 처리가 가능한 모니터링 시스템도 대안 중 하나.

상세설계

핵심 컴포넌트 - 데이터 집계

- 수집 에이전트
  - 클라이언트 별 간단한 집계는 간단
- 데이터 수집 파이프라인
  - 저장소에 기록하기 전 집계하는 방식
  - 해당 방식을 사용하면 계산된 결과만 기록하므로 데이터 양 축소
  - 원본 데이터가 없으므로 유연성, 정밀도 하락
- 질의에 의한 집계
  - 원본 데이터를 저장 후 필요 시 질의를 통해 집계 연산
  - 모든 데이터를 저장해야 하므로 시간, 공간 복잡도 증가

상세설계

핵심 컴포넌트 - 질의 서비스

- 시스템에서 접수된 요청을 시계열 DB를 통해 처리하는 역할
- 질의 처리 전담 서비스를 분리하여 시계열 DB와 클라이언트의 결합도를 낮출 수 있다.
- 질의 결과를 저장할 캐시 서버를 도입하여 DB에 대한 질의 부하를 낮추고 질의 서비스의 성능을 높일 수 있다.
- 시각화 및 경보 시스템은 시계열 DB와 연동을 처리하는 기능을 이미 갖추고 있을 수 있음
  - 별도의 캐시를 도입할 필요가 없는 시계열 DB도 존재하므로 도입 시 고민 필요

상세설계

핵심 컴포넌트 - 저장소 계층, 경보 시스템

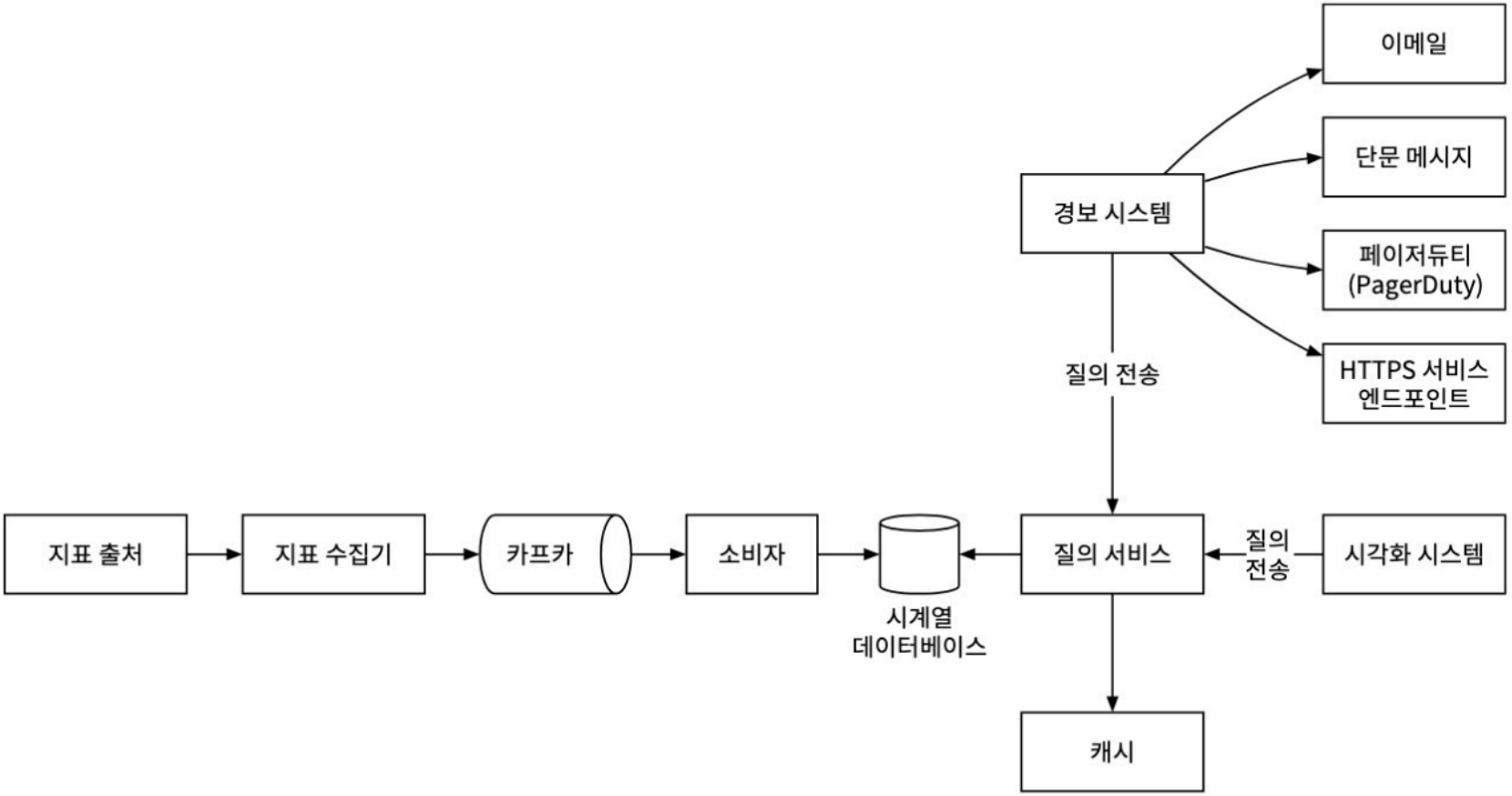
- 저장소는 질의의 빈도와 목적에 맞게 선택하면 비용을 획기적으로 줄일 수 있다.

- 다운 샘플링을 통한 저장 용량 최적화를 실행하고,
  잘 사용되지 않는 데이터는 Cold Storage에 보관하여 효율적으로 관리

- 경보 시스템은서 중요한 알림이 누락되지 않도록 보장되어야 한다

- 시각화와 경보 시스템은 직접 구축하는 것 보다, 만들어진 상용품을
  잘 선택하여 사용하는 것이 효율적

최종 설계안


이메일
단문 메시지
경보 시스템
페이저듀티
(PagerDuty)
HTTPS 서비스
엔드포인트
질의 전송
지표 출처
지표 수집기
카프카
소비자
시계열
데이터베이스
질의 서비스
질의
전송
시각화 시스템
캐시

최종 설계안

개인적인 의견

최근 n건, 특정 주기(최근 1시간, 1주, 1달) 에 대한 집계 및 자주 질의하는 집계 데이터 캐싱

-> 집계 질의 결과를 캐싱하는 건 캐시히트 확률이 적을 것이라 생각