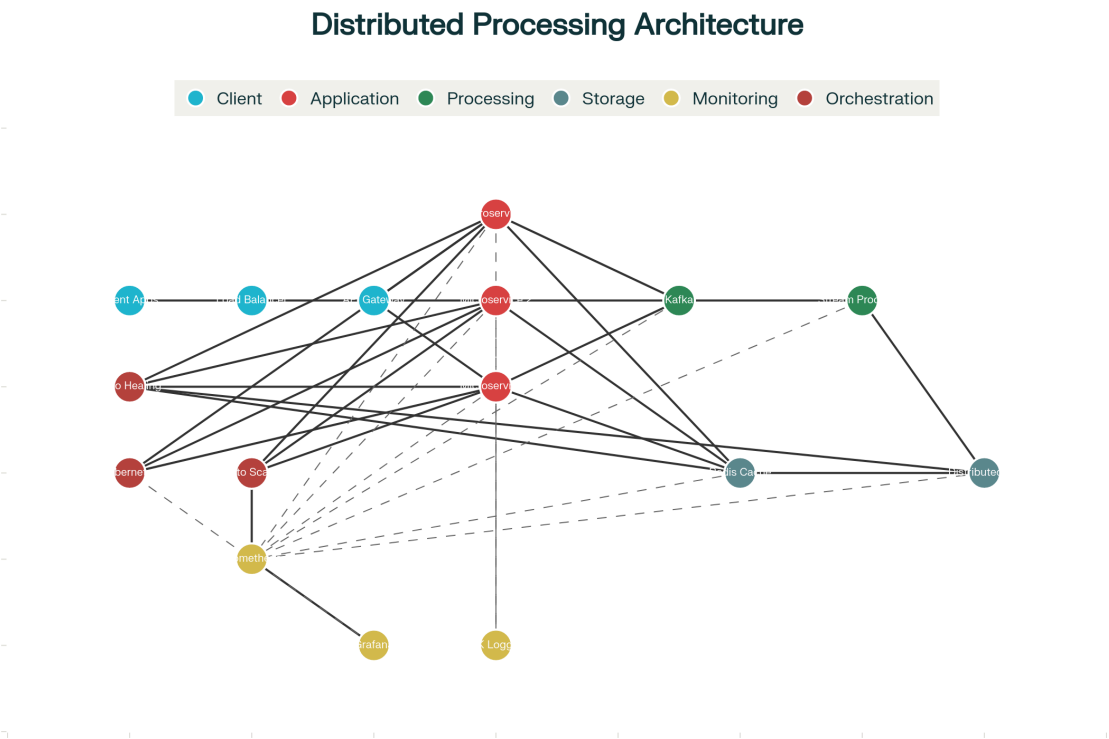
탄력성, 고가용성, 고성능, 데이터 일관성을 고려한 Distributed Processing Solution을 개발하기 위한 구조 설계를 어떻게 해야 하는지 알려줘

분산 처리 솔루션을 설계할 때 탄력성, 고가용성, 고성능, 데이터 일관성을 모두 충족시키려면 각 계층의 역할과 상호작용을 명확히 정의해야 한다. 아래 다이어그램은 이러한 목표를 달성하기 위한 대표적인 아키텍처 구조를 보여준다.



탄력성, 고가용성, 고성능, 데이터 일관성을 고려한 분산 처리 솔루션 아키텍처

핵심 설계 원칙

1. **탄력성 (Elasticity)**
   Kubernetes 기반으로 마이크로서비스 컨테이너를 오케스트레이션하여 트래픽 변화에 따라 자동으로 확장하거나 축소한다. Horizontal Pod Autoscaler와 Cluster Autoscaler를 통해 자원을 동적으로 조정한다.

2. **고가용성 (High Availability)**
   각 서비스와 데이터 저장소는 다중 리전 및 존에 분산 배치되어 장애 발생 시에도 지속적인 서비스가 가능하다. 로드 밸런서와 헬스체크 기반 트래픽 라우팅으로 무중단 운영을 보장한다.

3. **고성능 (High Performance)**
    - Redis Cluster로 자주 사용되는 데이터 캐싱
    - Kafka 기반 비동기 메시지 큐로 대규모 이벤트 처리
    - Spark 또는 Flink를 통한 실시간 스트리밍 분석 및 데이터 파이프라인 구축
    - gRPC 또는 REST API를 통한 효율적인 마이크로서비스 간 통신
4. **데이터 일관성 (Data Consistency)**
   분산 DB(Cassandra, CockroachDB 등)는 강한 일관성 또는 조정 가능한 일관성 모델을 사용해 트랜잭션 무결성을 유지한다. 트랜잭션 로그 기반 CDC(Change Data Capture) 시스템으로 장애 복구 시 데이터 손실을 최소화한다.

계층별 구성

- **클라이언트 계층**: API Gateway + Load Balancer
- **애플리케이션 계층**: Kubernetes 기반 마이크로서비스
- **데이터 처리 계층**: Kafka, Spark/Flink 처리 엔진
- **저장소 계층**: Cassandra, Redis
- **모니터링 및 운영 계층**: Prometheus, Grafana, ELK Stack

이러한 구조를 통해 시스템은 대규모 트래픽을 처리하면서도 유연한 확장, 장애 허용성, 데이터 일관성을 모두 보장할 수 있다.