CAP 이론

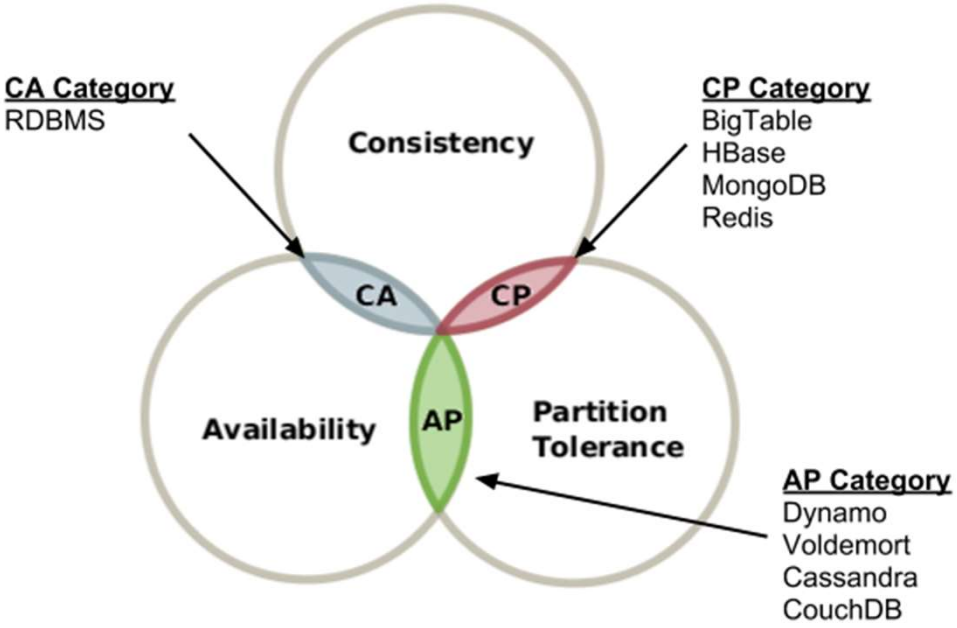
목차

- CAP이란?
- CAP이론에 따른 DB 분류 및 한계
- PACELC

CAP이란?

- 다음과 같은 세 가지 조건을 모두 만족하는 분산 컴퓨터 시스템이 존재하지 않음을 증명한 정리
  - **일관성**(**C**onsistency): 모든 노드가 같은 순간에 같은 데이터를 볼 수 있다.
  - **가용성**(**A**vailability): 모든 요청이 성공 또는 실패 결과를 반환할 수 있다.
  - **분할내성**(**P**artition tolerance): 메시지 전달이 실패하거나 시스템 일부가 망가져도 시스템이 계속 동작할 수 있다.
- 3가지 중 최대 2가지만 만족할 수 있다. (다음 장에서 설명)

CAP란? (CAP 삼각형)



- **CP 시스템**

완벽한 일관성을 갖는 분산 시스템에서 데이터 변경은 존재하는 모든 노드에 복제되어야 완료되는데, 이는 가용성과 성능에 악영향을 끼친다. 만약 하나의 노드라도 문제가 있으면 트랜잭션은 실패하고, 노드가 늘어날 수록 지연시간은 길어진다.

- **AP 시스템**

완벽한 가용성을 갖는 분산 시스템에서는 모든 노드가 어떤 상황에서도 응답할 수 있어야 한다. 네트워크 문제가 발생해서 어떤 노드에 복제가 제대로 이루어지지 않아도 가용성을 위해 해당 노드에 접근한 사용자에게 데이터를 반환한다고 생각해보면 일관성이 깨지는 것은 당연하고, 사용자는 문제가 발생한 것을 인지할 수 없다.

- **CA 시스템**

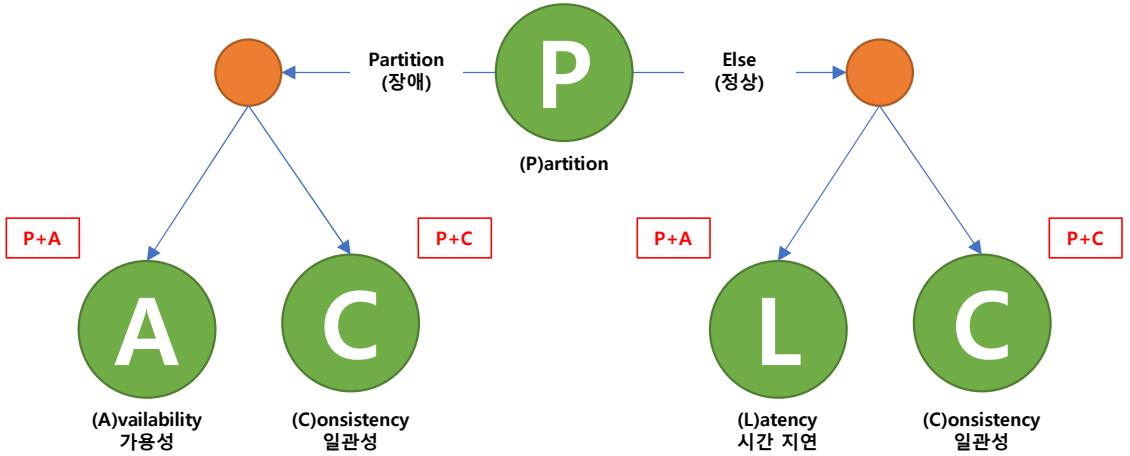
일관성과 가용성을 동시에 완벽히 만족하려면, 네트워크 장애를 허용하지 않아야 한다. 네트워크 장애가 절대 일어나지 않는 네트워크 구성은 불가하므로, CAP 이론은 네트워크 파티션 허용을 전제 한다. 무조건 P를 선택하고, C와 A 중 하나를 고르게 되면 CA는 배제하고, CP, AP로만 얘기하게 된다.

CAP정리에 따른 DB 분류 및 한계

- **MySQL**
  - MySQL은 기본적으로 **CA**에 속한다.
  - master 노드가 있고 그 노드를 복제해 사용하는 slave 패러다임을 사용하기 때문 (보통 읽기와 쓰기 부하를 분산) 이때 MySQL은 **가용성(A)과 일관성(C)을 만족**하게 된다.
  - MySQL은 cluster 설정을 지원한다. **cluster 설정을 하면 MySQL은 CP를 만족하는 시스템**이 된다. data를 유지할 만한 cluster 노드가 존재하지 않으면 cluster는 종료될 것이기 때문.
- **DynamoDB**
  - **DynamoDB는 aws에서 지원하는 Key/Value의 NoSQL로 AP에 속한다**.
    DynamoDB는 key값을 hashing하고 또 이 값을 mod하여 맞는 서버에 저장해둔다.
  - DynamoDB에는 strongly consistent 설정을 할 수 있다. 이 설정으로 가장 최신의 데이터를 반드시 리턴하므로 **DynamoDB는 CP로 변하게 된다.** 따라서 모든 노드를 찾기 때문에 가용성이 조금 떨어지게 된다.
- **위와 같이 DB를 CAP의 이론으로 분류 (AP, CP)하고, 사용할 DB를 채택하는 것은** 시간이 지날수록 (DB에서 여러가지 기능을 지원하게 됨으로써) **실효성을 잃고 있다.** 이를 보완하기 위해 나타난 것이 *PACELC 이론*이다.

PACELC

- CAP 이론의 한계인 C와 A는 상충하는 개념이고, Partition은 항상 발생하지 않는 점을 고려한 이론



PACELC

- Partition 발생에 따라 **PA/EL, PA/EC, PC/EL, PC/EC** 기준이 생성되고, 아래와 같이 NoSQL을 분류할 수 있다.

장애	정상	설명
PA	EL	장애상황: 가용 노드만 반영, 정상상황: Latency 우선 고려
PA	EC	장애상황: 가용 노드만 반영, 정상상황: 모든 메시지 보장
PC	EL	장애상황: Timeline 일관성 보장, 정상상황: Latency 우선 고려
PC	EC	장애상황: 일관성 우선 보장, 정상상황: 모든 메시지 보장

PA/EL	다이나모 , 카산드라
PA/EC	몽고DB
PC/EL	PNUTS
PC/EC	Megastore, VoltDB