서비스설계 스터디 1주차

- 사용자의 요청이 처리되는 과정
- 데이터베이스 사용
- 로드벨런서
- 캐시
- 인증
- 메시지큐
- 토의

사용자의 요청이 처리되는 과정

(1) 사용자의 요청이 처리되는 과정

1. 사용자는 접속할 사이트의 도메인을 브라우저에 입력한다.
2. 도메인 조회 결과로 IP 주소가 반환된다.
3. 해당 IP 주소로 HTTP 요청이 전달된다.
4. 서버는 응답으로 컨텐츠를 반환한다.
5. 브라우저는 화면을 출력한다.

(1) 사용자의 요청이 처리되는 과정

1. 사용자는 접속할 사이트의 도메인을 브라우저에 입력한다.
    1. 크로미니움 기반 브라우저 (크롬, 엣지, 웨일)
    2. Web Engine (Renderer)
    3. Frame Engine (Compositor)
    4. JS Engine (V8)
    5. Network Stack (OS Kernel)

2. 도메인 조회 결과로 IP 주소가 반환된다.
3. 해당 IP 주소로 HTTP 요청이 전달된다.
4. 서버는 응답으로 컨텐츠를 반환한다.
5. 브라우저는 화면을 출력한다.

(1) 사용자의 요청이 처리되는 과정

1. 사용자는 접속할 사이트의 도메인을 브라우저에 입력한다.
2. 도메인 조회 결과로 IP 주소가 반환된다.
    1. DNS 동작 원리
    2. TCP/IP 통신 원리

3. 해당 IP 주소로 HTTP 요청이 전달된다.
4. 서버는 응답으로 컨텐츠를 반환한다.
5. 브라우저는 화면을 출력한다.

(1) 사용자의 요청이 처리되는 과정

1. 사용자는 접속할 사이트의 도메인을 브라우저에 입력한다.
2. 도메인 조회 결과로 IP 주소가 반환된다.
3. 해당 IP 주소로 HTTP 요청이 전달된다.
    1. Http 프로토콜 (Stateless)
    2. Http Request 객체
    3. Http Response 객체
    4. Stateless 상태에서 사용자를 구분하는 방법

4. 서버는 응답으로 컨텐츠를 반환한다.
5. 브라우저는 화면을 출력한다.

(1) 사용자의 요청이 처리되는 과정

1. 사용자는 접속할 사이트의 도메인을 브라우저에 입력한다.
2. 도메인 조회 결과로 IP 주소가 반환된다.
3. 해당 IP 주소로 HTTP 요청이 전달된다.
4. 서버는 응답으로 컨텐츠를 반환한다.
    1. Servlet Life-Cycle
    2. Tomcat 서블릿 엔진
    3. Tomcat 스레드 모델
    4. Server Side Rendering vs Client Side Rendering

5. 브라우저는 화면을 출력한다.

(1) 사용자의 요청이 처리되는 과정

1. 사용자는 접속할 사이트의 도메인을 브라우저에 입력한다.
2. 도메인 조회 결과로 IP 주소가 반환된다.
3. 해당 IP 주소로 HTTP 요청이 전달된다.
4. 서버는 응답으로 컨텐츠를 반환한다.
5. 브라우저는 화면을 출력한다.
    1. Dom Tree
    2. Render Tree
    3. Network Latency

데이터베이스 사용

(2) 데이터베이스 사용

1. RDB vs NOSQL
2. RDB
3. NOSQL
4. 수직적 규모 확장 (Scale UP)
5. 수평적 규모 확장 (Scale OUT)
6. 데이터베이스 이중화
7. 읽기 분산

(2) 데이터베이스 사용

1. RDB vs NOSQL
    1. ACID
    2. NOSQL과 트랜잭션
    3. Multi Master가 힘든 이유
    4. RDB 기반 Multi Master (AWS 오로라DB)
    5. NoSQL 기반 Multi Master (GCP Spanner)

2. RDB
3. NOSQL
4. 수직적 규모 확장 (Scale UP)
5. 수평적 규모 확장 (Scale OUT)
6. 데이터베이스 이중화
7. 읽기 분산

(2) 데이터베이스 사용

1. RDB vs NOSQL
2. RDB
    1. Isolation Level
    2. PK 발행 (Lock, Max Value)
    3. Auto Increment 키를 재활용하려면?
    4. Char vs Varchar
    5. 타임존, 다국어

3. NOSQL
4. 수직적 규모 확장 (Scale UP)
5. 수평적 규모 확장 (Scale OUT)
6. 데이터베이스 이중화
7. 읽기 분산

(2) 데이터베이스 사용

1. RDB vs NOSQL
2. RDB
3. NOSQL
    1. 키 생성 규칙
    2. 샤드 구성과 리벨런싱
    3. Aggregation 동작원리

4. 수직적 규모 확장 (Scale UP)
5. 수평적 규모 확장 (Scale OUT)
6. 데이터베이스 이중화
7. 읽기 분산

(2) 데이터베이스 사용

1. RDB vs NOSQL
2. RDB
3. NOSQL
4. 수직적 규모 확장 (Scale UP)
    1. 서버의 하드웨어 증설 (비용은?)
    2. 테이블 분할 (Table Partitioning)

5. 수평적 규모 확장 (Scale OUT)
6. 데이터베이스 이중화
7. 읽기 분산

(2) 데이터베이스 사용

1. RDB vs NOSQL
2. RDB
3. NOSQL
4. 수직적 규모 확장 (Scale UP)
5. 수평적 규모 확장 (Scale OUT)
    1. 서버의 숫자를 증설
    2. 테이블 샤딩
    3. 샤딩키 발급 알고리즘이 중요한 이유
    4. 샤딩 환경에서 JOIN
6. 데이터베이스 이중화
7. 읽기 분산

(2) 데이터베이스 사용

1. RDB vs NOSQL
2. RDB
3. NOSQL
4. 수직적 규모 확장 (Scale UP)
5. 수평적 규모 확장 (Scale OUT)
6. 데이터베이스 이중화
    1. Master DB
    2. Slave DB
    3. Replica DB
    4. Mysql binlog (트랜잭션로그)
7. 읽기 분산

(2) 데이터베이스 사용

1. RDB vs NOSQL
2. RDB
3. NOSQL
4. 수직적 규모 확장 (Scale UP)
5. 수평적 규모 확장 (Scale OUT)
6. 데이터베이스 이중화
7. 읽기 분산
    1. Replica DB가 많아지면 발생하는 문제들
    2. IDC 이중화
    3. Replica DB 사이의 Latency
    4. DB와 웹서버 사이의 Latency

로드벨런서

(3) 로드벨런서

1. Public IP와 Private IP
2. L4
3. L7
4. 로드벨런서 Failover

(3) 로드벨런서

1. Public IP와 Private IP
    1. 내부망, 외부망 분리하는 이유
    2. X-forworked-for 헤더
    3. 프록시 헬

2. L4
3. L7
4. 로드벨런서 Failover

(3) 로드벨런서

1. Public IP와 Private IP
2. L4
    1. 장단점

3. L7
4. 로드벨런서 Failover

(3) 로드벨런서

1. Public IP와 Private IP
2. L4
3. L7
    1. 장단점

4. 로드벨런서 Failover

(3) 로드벨런서

1. Public IP와 Private IP
2. L4
3. L7
4. 로드벨런서 Failover
    1. VIP
    2. L7 Check
    3. 장비가 추가되거나 빠질때 문제점

캐시

(4) 캐시

1. 정적인 컨텐츠 캐시
2. 동적인 컨텐츠 캐시
3. 캐싱 전략
4. 데이터 일관성
5. SPOF (Single Point of Failure)
6. 레디스 클러스터

(4) 캐시

1. 정적인 컨텐츠 캐시
    1. Static 서버 분리가 필요한 이유
    2. CDN
    3. 네트워크 환경에 따른 CDN 전략

2. 동적인 컨텐츠 캐시
3. 캐싱 전략
4. 데이터 일관성
5. SPOF (Single Point of Failure)
6. 레디스 클러스터

(4) 캐시

1. 정적인 컨텐츠 캐시
2. 동적인 컨텐츠 캐시
    1. 로컬 캐시
    2. 글로벌 캐시
    3. 하이브리드 캐시

3. 캐싱 전략
4. 데이터 일관성
5. SPOF (Single Point of Failure)
6. 레디스 클러스터

(4) 캐시

1. 정적인 컨텐츠 캐시
2. 동적인 컨텐츠 캐시
3. 캐싱 전략
    1. 읽기가 중요한 서비스 (8:2)
    2. 쓰기가 중요한 서비스 (2:8)
    3. Eviction vs Expiration
    4. Cache Expiration 정책
    5. Cache Eviction 정책

4. 데이터 일관성
5. SPOF (Single Point of Failure)
6. 레디스 클러스터

(4) 캐시

1. 정적인 컨텐츠 캐시
2. 동적인 컨텐츠 캐시
3. 캐싱 전략
4. 데이터 일관성
    1. Thead-Safe
    2. ConcurrentHashMap

5. SPOF (Single Point of Failure)
6. 레디스 클러스터

(4) 캐시

1. 정적인 컨텐츠 캐시
2. 동적인 컨텐츠 캐시
3. 캐싱 전략
4. 데이터 일관성
5. SPOF (Single Point of Failure)
    1. 레디스는 싱글 스레드로 동작한다. (keys?)
    2. Redis Master, Redis Slave
    3. Redis Sentinel
    4. Redis Cluster

6. 레디스 클러스터

(4) 캐시

1. 정적인 컨텐츠 캐시
2. 동적인 컨텐츠 캐시
3. 캐싱 전략
4. 데이터 일관성
5. SPOF (Single Point of Failure)
6. 레디스 클러스터
    1. 자동 샤딩
    2. 자동 리벨런싱
    3. Hash Slot (16,384)
    4. 자료구조 (Key-value, Hash, List)

인증

(5) 인증

1. 상태정보(Session) 의존적인 아키텍처
2. 로드벨런서 (Sticky Session)
3. 무상태 아키텍처
4. SSO (Single Sign On)
5. 서비스 이중화 (IDC 이중화)

(5) 인증

1. 상태정보(Session) 의존적인 아키텍처
    1. Session 정보가 필요한 이유
    2. Session 클러스터링

2. 로드벨런서 (Sticky Session)
3. 무상태 아키텍처
4. SSO (Single Sign On)
5. 서비스 이중화 (IDC 이중화)

(5) 인증

1. 상태정보(Session) 의존적인 아키텍처
2. 로드벨런서 (Sticky Session)
    1. Session과 Sticky

3. 무상태 아키텍처
4. SSO (Single Sign On)
5. 서비스 이중화 (IDC 이중화)

(5) 인증

1. 상태정보(Session) 의존적인 아키텍처
2. 로드벨런서 (Sticky Session)
3. 무상태 아키텍처
    1. Session 정보는 어디에?
    2. Session 저장소

4. SSO (Single Sign On)
5. 서비스 이중화 (IDC 이중화)

(5) 인증

1. 상태정보(Session) 의존적인 아키텍처
2. 로드벨런서 (Sticky Session)
3. 무상태 아키텍처
4. SSO (Single Sign On)
    1. SSO와 Oauth
    2. 인증 쿠키 발급과정
    3. 인증 Validation
    4. 접근제한, 차단, 연장
    5. 모바일앱 인증

5. 서비스 이중화 (IDC 이중화)

(5) 인증

1. 상태정보(Session) 의존적인 아키텍처
2. 로드벨런서 (Sticky Session)
3. 무상태 아키텍처
4. SSO (Single Sign On)
5. 서비스 이중화 (IDC 이중화)
    1. GSLB를 이용한 IDC 이중화 (Active-Active, Active-Standby)
    2. DB 미러링
    3. NAS 미러링
    4. 소스 배포 전략

메시지큐

(6) 메시지큐

1. Sync vs Async
2. 생산자 소비자 패턴 (Producer – Consumer)
3. AMQP
4. RabbitMQ

(6) 메시지큐

1. Sync vs Async
    1. Sync 장단점
    2. Async 장단점
    3. Near Realtime?

2. 생산자 소비자 패턴 (Producer – Consumer)
3. AMQP
4. RabbitMQ

(6) 메시지큐

1. Sync vs Async
2. 생산자 소비자 패턴 (Producer – Consumer)
    1. 서비스 처리량을 10배 늘리려면?
    2. 처리결과를 리턴하는 방법?

3. AMQP
4. RabbitMQ

(6) 메시지큐

1. Sync vs Async
2. 생산자 소비자 패턴 (Producer – Consumer)
3. AMQP
    1. Work Queues (Direct Exchange)
    2. Publish/Subscribe (Fanout Exchange)
    3. Msg Routing (Topic Exchange)

4. RabbitMQ

(6) 메시지큐

1. Sync vs Async
2. 생산자 소비자 패턴 (Producer – Consumer)
3. AMQP
4. RabbitMQ
    1. Queue != FIFO
    2. Dead-Letter Queue
    3. RabbitMQ Failover

토의

1장 사용자 수에 따른 규모 확장성 - TBD #1

Open · 6 tasks · **rygh4775** opened this issue 4 days ago · 0 comments

rygh4775 commented 4 days ago • edited

- [ ] 로드밸런서 SPOF / 스위치, VIP, 기타
- [ ] 데이터베이스 복구 스크립트 / 백업 미존재시, 경험 사례, 기타
- [ ] 데이터베이스 다중 마스터(multi-masters) / 실 운영 사례, 기타
- [ ] 데이터베이스 원형 다중화(circular replication) / 실 운영 사례, 기타
- [ ] 캐시 전략에 대한 비교
  - https://luran.me/357
    - Cache-Aside (일반적으로 많이 사용)
    - Read-Through
    - Write-Through
    - Write-Behind
- [ ] 지리적 라우팅(geo-routing) by 로드밸런서 or GeoDNS

https://github.com/JAVACAFE-STUDY/system-design-interview/issues/1