백엔드 & 인프라 스터디 2 회차 문제

2025-10-03

작성자 : 김호중

백엔드 – 논리적사고문제: 병렬프로그래밍

상황

당신은 전문 포토그래퍼와 이커머스 기업을 위한 B2B 클라우드 이미지 처리 플랫폼의 백엔드 엔지니어로 일하고 있습니다. 고객은 수백, 수천 장에 달하는 고해상도 RAW 이미지 파일(개당 50~100MB)을 업로드하고, 우리는 이 이미지들에 대해 다음과 같은 순차적인 처리 파이프라인을 실행하여 최종 결과물을 생성합니다.

디코딩: RAW 이미지 파일을 읽어 메모리 상의 비트맵데이터로 변환합니다.
리사이징: 다양한 디바이스에 맞게 여러 해상도(4K, 1080p, 썸네일 등)의 버전을 생성합니다.
필터 적용: 가우시안 블러, 샤프닝 등 CPU 연산량이 매우 높은 커스텀 필터를 적용합니다.
워터마킹: 고객사의 로고 이미지를 반투명하게 오버레이합니다.
인코딩 및 저장: 처리된 이미지를 JPEG, PNG 등 최종 포맷으로 변환하여 클라우드 스토리지에 저장합니다.

초기 버전의 처리기는 단일 스레드 Python 스크립트로 작성되었습니다. 이 스크립트는 32 코어의 최신 CPU 를 탑재한 서버에서 실행됨에도 불구하고, 단 하나의 CPU 코어만 사용하며 나머지 31 개 코어는 유휴 상태로 남아있습니다. 이로 인해 수천 장의 이미지를 처리하는 배치작업이 몇 시간씩 소요되어 고객의 불만이 증가하고 있습니다. 당신은 이 처리 시간을 획기적으로 단축시키기 위해, 모든 CPU 코어를 활용하는 병렬 처리 아키텍처를 설계해야 하는 과제를 받았습니다.

문제

첫째, 이 문제의 워크로드가 가진 특성, 즉 CPU 집약적(CPU-bound) 연산이 대부분이라는 점을 분석하고, 병렬 처리를 위해 멀티스레딩이 아닌 멀티프로세싱을 선택해야 하는 이유를 Python 의 전역 인터프리터 락과 연관 지어 근본적인 수준에서 설명하시오. 두 접근법의 메모리 사용량과 컨텍스트 스위칭 비용 측면의 트레이드오프를 함께 논해야 합니다.

둘째, 파이프라인의 4 단계인 워터마킹 로직에 "지금까지 처리된 이미지 총 개수"를 세어 과금 시스템에 전달하는 공유 카운터 기능이 추가되었다고 가정해 봅시다. 여러 워커 프로세스가 단순히 전역 변수 total_processed_images += 1 과 같은 코드를 실행할 때, 왜 경쟁 상태가 발생하여 최종 집계 수가 실제 처리량보다 적게 나올 수 있는지 그 과정을 '읽기-수정-쓰기'문제의 관점에서 상세히 설명해야 합니다. 이 문제를 해결하기 위해 multiprocessing 모듈이 제공하는락과 같은 동기화 기본요소를 사용하여 어떻게 데이터의 원자성을 보장할 수 있는지 구체적인 방안을 제시하시오.

셋째, 멀티프로세싱 모델에서 주 프로세스와 워커 프로세스 간의 데이터 통신은 직렬화(Serialization), 즉 파이썬의 피클링(pickling)을 통해 이루어집니다. 100MB 에 달하는 거대한 비트맵 이미지 데이터를 각 워커 프로세스에게 전달하는 현재의 방식이 왜 심각한 성능 병목을 유발할 수 있는지, 직렬화와 데이터 복사에 드는 오버헤드 관점에서 분석하시오. 이를 해결하기 위해, 각 워커 프로세스가 메모리를 통해 거대한 데이터를 직접 주고받는 대신, 처리할 원본 파일의 경로만 전달받아 디코딩부터 저장까지 모든 단계를 독립적으로 수행하고 결과만 보고하는 방식으로 프로세스 간 통신(IPC)을 최소화하는 아키텍처의 장점을 논하시오.

넷째, 32 개의 모든 코어를 완벽하게 활용하더라도 처리 속도가 32 배가 될 것이라고 기대할 수는 없습니다. 병렬 컴퓨팅의 성능 향상 한계를 설명하는 암달의 법칙의 관점에서, 현재 이미지 처리 파이프라인의 전체 과정 중 필연적으로 순차적으로 실행될 수밖에 없는 부분(작업 분배, 최종 결과 취합)과 병렬화 가능한부분(개별 이미지 처리)을 구분하여 설명해야 합니다. 이 순차적인 부분의 비율이 전체 시스템의 최대 성능 향상률을 어떻게 제한하는지 그 관계를 분석하시오.

다섯째, 시스템에 새로운 요구사항으로 필터 적용 시 제한된 수량의 '필터 라이선스'를 먼저 획득해야 하는 기능이 추가되었다고 가정해 봅시다. 만약 특정 워커 프로세스가 작업을 위해 '필터 라이선스'와 '워터마크 라이선스'라는 두 개의 공유 자원을 모두 획득해야 하는데, 일부 프로세스는 필터→워터마크 순서로, 다른 프로세스는 워터마크→필터 순서로 자원을 요청한다면 어떤 치명적인 문제가 발생할 수 있는지 설명하시오. 이러한 교착 상태가 발생하는 네 가지 필요조건(상호 배제, 점유 대기, 비선점, 환형 대기)을 언급하고, 이 문제를 예방하기 위해 모든 프로세스가 항상 동일한 순서로 자원을 획득하도록 강제하는 정책이 왜 효과적인 해결책인지 논하시오.

인프라 심층 탐구: 단일 서버의 한계와 분산 시스템

상황

당신은 책 추천 및 독서 클럽 소셜 플랫폼의 첫 인프라 개발자로 합류했습니다. 이것은 뛰어난 아이디어 덕분에 빠르게 성장 가능성을 인정받았지만, 아직 시드 투자를 받기 전이라 기술 팀과 예산이 매우 제한적입니다.

현재 BookNook 의 전체 인프라는 클라우드 제공업체의 단일 가상 서버(예: AWS EC2 t3.medium) 한 대로 운영되고 있습니다. 이 서버 안에는 다음과 같은 모든 구성 요소가 함께 실행되고 있습니다.

Django 백엔드 애플리케이션: Gunicorn 을 통해 실행됩니다.
PostgreSQL 데이터베이스: 애플리케이션과 동일한 서버에 설치되어 있습니다.
Nginx 웹 서버: 정적 파일 서빙 및 Gunicorn 으로의 리버스 프록시 역할을 합니다.
DNS: booknook.com 도메인이 이 단일 서버의 고정 IP 주소로 직접 연결되어 있습니다.

얼마 전, 유명 북튜버가 클럽 소셜 플랫폼을 극찬하는 영상을 올린 후, 트래픽이 평소의 20 배 이상으로 폭증했습니다. 단일 서버는 몰려드는 요청을 감당하지 못했고, CPU 사용률이 100%에 달하며 데이터베이스 응답이 느려졌습니다. 결국 사이트는 수시로 멈추거나 접속 불가 상태에 빠졌고, 사용자들의 불만이 폭주했습니다.
투자 유치를 앞둔 CEO 는 당신에게 "다시는 이런 일이 발생하지 않도록, 안정적이고 확장 가능한 인프라 V2 를 설계해달라"는 특명을 내렸습니다.

문제

첫째, 현재 V1 아키텍처가 가진 근본적인 취약점을 단일 실패 지점의 관점에서 분석하시오. 사용자가 웹사이트에 접속하는 과정 전체를 짚어보며, 이 아키텍처에서 단 하나라도 장애가 발생하면 전체 서비스가 중단되는 지점이 몇 군데나 있는지, 그리고 각 지점이 어떤 역할을 하는지 구체적으로 설명해야 합니다.

둘째, V1 의 문제를 해결하기 위한 V2 아키텍처를 새롭게 설계하시오. 로드 밸런서, 오토 스케일링 그룹, 관리형 데이터베이스와 같은 기본적인 인프라 구성 요소를 활용하여, 트래픽 증가에 유연하게 대응하고 단일 컴포넌트의 장애가 전체 서비스 중단으로 이어지지 않는 그림을 제시해야 합니다. 사용자의 요청이 DNS 를 거쳐 최종적으로 데이터베이스에 도달하기까지의 전체 트래픽 흐름을 단계별로 설명하시오.

셋째, 당신이 설계한 V2 아키텍처는 두 대 이상의 애플리케이션 서버가 동시에 운영되는 것을 전제로 합니다. 이 구조가 V1 에는 없었던 새로운 두 가지의 심각한 상태 문제를 어떻게 발생시키는지 설명하시오. 첫 번째는 사용자의 세션 정보 처리 문제이며, 두 번째는 사용자가 업로드한 '프로필 이미지 파일' 처리 문제입니다. 이 두 가지 상태 저장 문제를 해결하기 위해 각각 어떤 외부의 중앙 집중식 서비스(Redis, S3)를 도입해야 하는지, 그리고 그 이유를 논리적으로 설명해야 합니다.

넷째, V2 아키텍처의 핵심인 로드 밸런서는 어떻게 특정 애플리케이션 서버가 '건강하지 않다'고 판단하고 해당 서버로 트래픽을 보내지 않을 수 있는지 그 원리를 설명하시오. 로드 밸런서의 헬스 체크 기능이 무엇이며, 이 기능이 어떻게 시스템의 고가용성과 '자동 장애 복구를 보장하는지 그 과정을 상세히 기술해야 합니다.

다섯째, 당신이 제안한 V2 아키텍처는 V1 에 비해 명백히 더 많은 비용을 요구합니다. 예산에 민감한 비기술직 CEO 에게 이 아키텍처 변경의 필요성과 투자 가치를 어떻게 설득하시겠습니까? 다운타임의 기회비용과 확장성 확보를 통한 미래 성장 기회'라는 비즈니스 관점에서, 기술적인 인프라 투자가 어떻게 회사의 신뢰도와 직접적인 매출로 이어질 수 있는지 그 논리를 설득력 있게 제시하시오.

인프라 심층 탐구: 인프라 좀더 어려운거

상황

당신은 빠르게 성장하는 B2B SaaS 스타트업에 첫 인프라 개발자로 입사했습니다. 스타트업의 핵심 서비스는, 고객사 마케팅팀이 업로드한 대용량 원본 데이터(Raw Data)를 기반으로 복잡한 연산을 수행하여, 마케팅 성과 분석 리포트를 생성해주는 웹 애플리케이션입니다.

현재 시스템의 가장 큰 문제점은 리포트 생성 기능입니다. 리포트 하나를 생성하는 데는 평균 5 분에서 10 분이 소요되며, 상당한 CPU 와 메모리 자원을 소모합니다. 현재 아키텍처는 이 모든 작업을 사용자의 HTTP 요청을 받은 웹 서버가 직접 동기적으로 처리하고 있습니다. 이로 인해 다음과 같은 심각한 문제들이 발생하고 있습니다.

타임아웃과 응답성 저하: 사용자가 리포트 생성버튼을 누르면, 웹 브라우저는 응답을 하염없이 기다리다 결국 타임아웃 오류를 냅니다. 리포트가 생성되는 동안 해당 웹 서버는 다른 가벼운 요청조차 제대로 처리하지 못해 서비스 전체의 응답성이 저하됩니다.

작업 유실: 웹 서버는 트래픽에 따라 자동으로 확장 및 축소되도록 설정되어 있습니다. 트래픽이 줄어드는 시점에 오토 스케일링 그룹이 서버 인스턴스를 축소하면, 10 분짜리 리포트를 생성하던 중이던 서버가 그대로 종료되어 사용자의 작업이 아무런 알림 없이 유실됩니다.

보안 및 데이터 관리의 어려움: 생성된 리포트 파일(1~2GB)은 웹 서버의 로컬 디스크(/tmp)에 임시 저장된 후 사용자에게 다운로드됩니다. 이는 서버가 종료되면 파일이 함께 사라지는 문제를 낳을 뿐만 아니라, 여러 서버 간에 데이터를 공유할 방법도 없습니다. 또한, 리포트 생성을 위해 S3 버킷에 접근해야 할 때, EC2 인스턴스에 IAM 사용자의 액세스 키를 직접 하드코딩해두어 심각한 보안 위험에 노출되어 있습니다.

당신에게 주어진 임무는, 이러한 문제들을 해결하고 안정적이며, 확장 가능하고, 안전한 V2 아키텍처를 AWS 클라우드 위에 설계하는 것입니다.

문제

첫째, 현재 시스템의 가장 큰 아키텍처적 결함은 웹 티어와 작업 티어 분리되지 않았다는 점입니다. 사용자의 긴급하지 않은 요청과 즉각적인 응답이 필요한 요청을 분리하기 위한 비동기처리 아키텍처를 설계하시오. 사용자가 API 를 통해 리포트 생성 요청을 보냈을 때, 서버가 어떻게 즉시 202 Accepted 와 같은 응답을 반환하고, 실제 무거운 작업은 백그라운드에서 처리하도록 만들 수 있는지 AWS SQS 와 별도의 워커 인스턴스 그룹을 활용하여 그 전체적인 흐름을 설명해야 합니다.

둘째, 비동기 아키텍처에서 데이터의 흐름과 상태 관리는 매우 중요합니다. 워커 인스턴스가 10 분에 걸쳐 생성한 2GB 크기의 리포트 파일을 어떻게 사용자에게 안전하고 효율적으로 전달할 수 있을지 그 과정을 상세히 설명하시오. AWS S3 를 어떻게 활용할 것이며, 작업의 진행 상태(예: '대기중', '처리중', '완료', '실패')는 어떻게 관리하고 사용자에게 보여줄 것인지, 그리고 최종적으로 사용자에게 파일을 다운로드시킬 때 S3 버킷을 외부에 공개하지 않으면서도 안전한 접근을 제공하는 S3 사전 서명된 URL(Pre-Signed URL)의 역할과 그 동작 원리를 포함하여 논하시오.

셋째, V2 아키텍처의 모든 EC2 인스턴스는 SQS, S3, RDS 등 다른 AWS 서비스와 상호작용해야 합니다. 인스턴스에 IAM 사용자의 액세스 키를 하드코딩하는 치명적인 보안 실수를 피하고, AWS 가 권장하는 모범 사례를 따라 어떻게 인스턴스에 안전하게 권한을 부여할 수 있는지 설명하시오. IAM 역할의 개념과 동작 원리를 설명하고, 왜 이 방식이 액세스 키를 직접 관리하는 것보다 근본적으로 더 안전한지 자동 자격 증명 교체(Automatic Credential Rotation)와 최소 권한 원칙(Principle of Least Privilege)의 관점에서 분석해야 합니다.

넷째, 리포트 생성 작업은 월말에 집중되고 평소에는 거의 발생하지 않는, 매우 변동성이 큰(Spiky) 워크로드를 가집니다. 이러한 특성을 고려할 때, 항상 고정된 수의 강력한 워커 인스턴스를 띄워두는 것은 심각한 비용 낭비입니다. 워커 티어의 인프라를 비용 효율적이고 탄력적으로 운영하기 위한 두 가지 서로 다른 전략을 제시하고 비교하시오. 첫 번째 전략은 SQS 대기열 길이(Queue Depth)에 기반한 오토 스케일링 그룹을 구성하는 방안입니다. 두 번째 전략은 EC2 인스턴스 자체를 관리할 필요가 없는 서버리스 컴퓨팅 서비스, 예를 들어 AWS Fargate 를 활용하는 방안입니다. 각 전략의 장단점과 어떤 상황에서 더 적합할지 당신의 통찰을 제시하시오.

다섯째, 당신이 설계한 V2 아키텍처의 견고함을 증명하기 위해, 시스템의 실패 시나리오를 스스로 가정하고 그에 대한 자동화된 복구 방안을 설명하시오. 예를 들어, 워커 인스턴스가 리포트를 생성하던 도중 알 수 없는 오류로 비정상 종료되었다면, 이 실패를 어떻게 감지하고 해당 작업을 다른 워커가 자동으로 재시도하도록 만들 수 있는지 SQS 의 가시성 시간 초과(Visibility Timeout)와 데드 레터 큐(Dead Letter Queue)의 개념을 활용하여 설명해야 합니다. 이는 시스템이 단순히 '잘 동작하는' 것을 넘어, '실패를 견딜 수 있도록' 설계되었음을 증명하는 핵심적인 부분입니다.

인프라 심층 탐구: 네트워크

상황

당신은 민감한 개인 금융 정보를 다루는 차세대 FinTech 서비스 'SecureWallet'의 인프라 설계자로 면접을 보고 있습니다. 이 서비스는 최고의 보안과 안정성을 요구하며, 당신의 임무는 AWS 클라우드 위에 이 서비스의 초기 네트워크 아키텍처를 설계하는 것입니다.

아키텍처 요구사항:

3-Tier 아키텍처: 사용자의 요청을 받는 웹 티어(Nginx), 비즈니스 로직을 처리하는 애플리케이션 티어(Django/Spring), 그리고 데이터를 저장하는 데이터베이스 티어(PostgreSQL)로 명확히 분리되어야 합니다.

보안: 애플리케이션 서버와 데이터베이스 서버는 어떠한 경우에도 인터넷에서 직접 접근할 수 없어야 합니다. 오직 허가된 경로를 통해서만 통신이 이루어져야 합니다.

외부 통신: 애플리케이션 서버는 신용 등급 조회 등을 위해 외부의 써드파티 금융 API 를 인터넷을 통해 호출해야 합니다.

고가용성: 단일 데이터 센터의 장애가 전체 서비스의 중단으로 이어져서는 안 됩니다.

당신은 이 요구사항들을 충족하는 AWS VPC(Virtual Private Cloud) 환경을 처음부터 설계해야 합니다.

문제

첫째, 당신은 가장 먼저 VPC 와 서브넷구조를 설계해야 합니다. '고가용성' 요구사항을 충족시키기 위해, 단일 가용 영역에 의존하는 설계를 넘어, 어떻게 VPC 와 서브넷을 구성하시겠습니까? CIDR 블록 설정의 기본 원칙을 설명하고, 퍼블릭 서브넷과 프라이빗 서브넷을 각각 최소 두 개 이상의 가용 영역에 걸쳐 생성하는 이유가 무엇인지, 그리고 웹, 애플리케이션, 데이터베이스 각 티어가 어떤 서브넷에 위치해야 하는지 그 근거를 논리적으로 설명해야 합니다.

둘째, 사용자가 브라우저에서 securewallet.com 을 입력했을 때, 그 요청 패킷이 당신이 설계한 VPC 의 프라이빗 서브넷에 있는 애플리케이션 서버까지 도달하는 전체 여정을 단계별로 상세히 추적하여 설명하시오. 이 여정에는 Route 53, Internet Gateway, 라우팅 테이블, Application Load Balancer, 그리고 보안 그룹과 같은 AWS 의 핵심 네트워크 컴포넌트들이 모두 포함되어야 하며, 각 컴포넌트가 패킷의 경로를 결정하고 보안을 검사하는 데 어떤 역할을 하는지 그 원리를 설명해야 합니다.

셋째, 프라이빗 서브넷에 위치한 애플리케이션 서버는 외부 인터넷으로 응답을 보낼 수 없습니다. 하지만 외부 통신 요구사항에 따라 외부 금융 API 를 호출해야 하는 모순적인 상황입니다. 이 문제를 어떻게 해결하시겠습니까? NAT Gateway 의 역할과 동작 원리를 설명하고, 이 NAT Gateway 를 어떤 서브넷에 위치시켜야 하며, 프라이빗 서브넷의 라우팅 테이블을 어떻게 설정해야만 내부 인스턴스가 안전하게 외부 인터넷으로 나가는(Egress) 통신을 할 수 있는지 그 아키텍처를 상세히 기술해야 합니다.

넷째, FinTech 서비스의 보안은 아무리 강조해도 지나치지 않습니다. 당신은 보안 그룹과 네트워크 ACL 이라는 두 가지 방화벽을 모두 사용하여 심층 방어(Defense in Depth)전략을 구현해야 합니다. 이 두 가지 기술의 근본적인 차이점, 즉 상태 저장(Stateful)과 상태 비저장(Stateless)의 차이, 그리고 적용되는 범위(인스턴스 레벨 vs. 서브넷 레벨)의 차이를 설명하시오. 이어서, 웹 티어와 애플리케이션 티어 사이의 통신을 제어하기 위해, 보안 그룹으로는 "오직 ALB 로부터 오는 트래픽만 허용"과 같이 정교한 규칙을 적용하고, 네트워크 ACL 로는 "이 서브넷으로는 SSH 트래픽(포트 22)은 절대 들어올 수 없음"과 같이 더 광범위한 규칙을 적용하는 구체적인 보안 정책 설계안을 제시하시오.

다섯째, 서비스가 성공적으로 성장하여 애플리케이션 서버가 매일 수 테라바이트의 로그를 S3 에 저장하고, 서버 관리를 위해 AWS Systems Manager 를 활발히 사용하기 시작했다고 가정해 봅시다. 이 트래픽이 모두 NAT Gateway 를 통해 인터넷을 거쳐감에 따라, 예기치 않은 막대한 데이터 전송 비용이 발생하기 시작했습니다. 이 문제를 어떻게 해결하시겠습니까? VPC 엔드포인트의 개념을 설명하고, 게이트웨이 엔드포인트(Gateway Endpoint)와 인터페이스 엔드포인트(Interface Endpoint)의 차이점을 분석해야 합니다. S3 로 향하는 트래픽과 SSM 으로 향하는 트래픽을 각각 어떤 유형의 엔드포인트를 통해 NAT Gateway 를 우회시키고, 이를 통해 어떻게 비용을 절감하고 보안을 강화할 수 있는지 그 아키텍처적 개선 방안을 논하시오.