Cooperative multi-tasking with **Node.js**

Node.js로 협력적 멀티태스킹 처리하기

Speaker

Irho Park (박일호)

aka. **iMaZiNe** @ Kakao corp.

Daum자동차 서비스 개발

고백

거창하게 제목을 정해 놓았지만…

하려고 하는 이야기는

적다고 할 수 없는 숫자의 데이터 마이그레이션 경험담

사건의 시작

2016년 카카오 입사

11개월 만에 꿈에 그리던 제주생활

출근 20분, 퇴근 15분

월정리 20분, 함덕서우봉해변 15분

그렇게 2016년 11월
Daum자동차 서비스를 담당하게 됩니다.

여기까지는 Happy함

첫번째 임무

Legacy article migration

대부분 먼저 하는 생각

1min = 60s

1hour = 3,600sec

1day = 86,400sec

초당 100개씩만 처리하면…

여기까지도 Happy함

요구사항 분석

구문서 파싱 및 가공

어뷰징 제거

연결된 정보들 처리 (ex. 댓글)

이 모든것을 안전하게 처리

어떻게

DB를 읽고, 쓰고

파일을 읽고, 쓰고

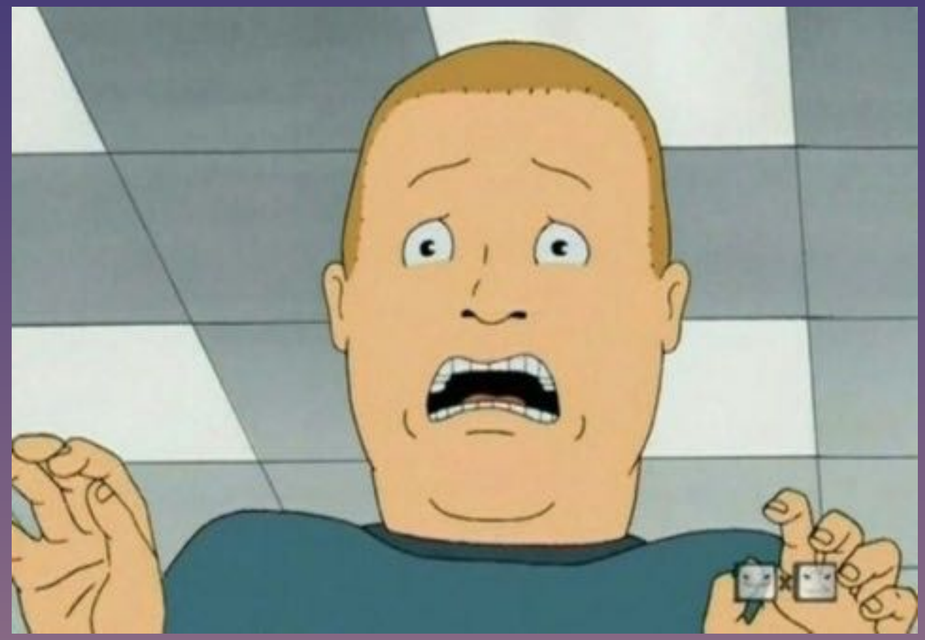
API를 호출하고

결국

비동기 요청들(**Asynchronous** Requests)을

순차적(**Sequentially**)으로

안전하게(**Transactionally**) 처리



정해진 시간 안에

Why use **Node.JS**

Because, I love it!

Simulation

1개의 article을 처리하기 위해 필요한 비동기요청 수는

최소 8개 + 댓글 수 * 2

댓글이 없다고 가정하고 비동기 요청 1개당 20ms 로 잡아도

최소 160ms 필요 (6.25/sec)

86400 * 6.25 = 540,000

실제 효율은 절반 이하...

느려도 너무 느리다!

Solution

병렬처리로 속도를 높이거나

프로세스의 효율을 높이거나

마음을 비우고 순차적으로…

결국 동시성(Concurrency) 문제를 해결해야 하는 상황!

Message Queue 를 사용해 스케쥴링 하고

병렬처리 하기로…

아직까지도 Happy함

구현해 봅시다

비동기를 처리하는 대표적인 방법

Callback!

Callback DEMO

Callback Image?

조금 더 나은 방법

Promise

Promise DEMO

How to solve memory issue

협력적 멀티태스킹

Cooperative Multi-tasking

일종의 시분할(Time-sharing)방식

운영체제의 개입 없이 task가 독점적으로 CPU를 사용

미사용시 자발적 CPU 자원 반환

Critical section 보호를 위한 Lock이나 Semaphore 불필요

서광열의 코딩스쿨(https://gamecodingschool.org) 참조

구현방법의 선택

Coroutine

ES7 스펙 async, await

Node.js 7.6 부터 공식적으로 지원

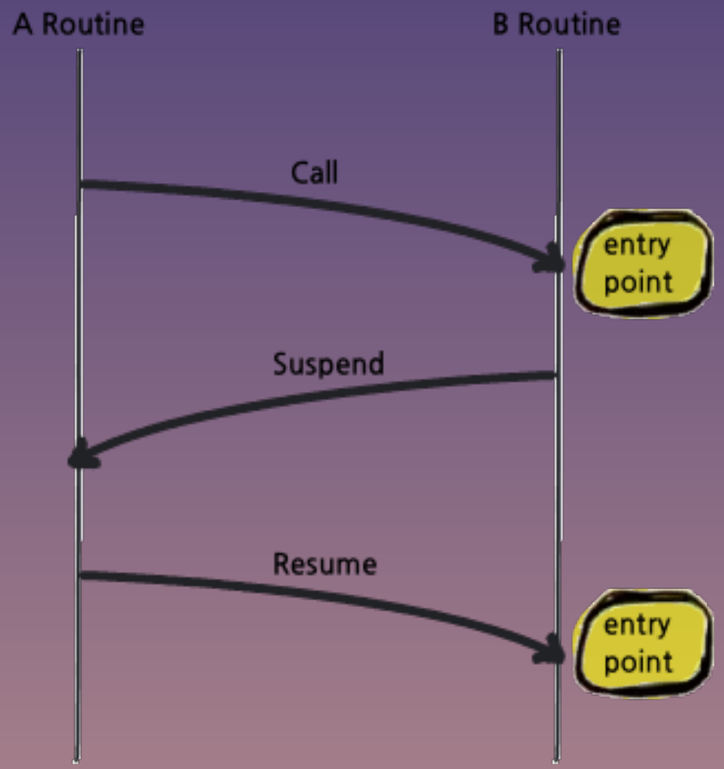
Node.js 6.x —harmony 옵션과 함께 사용

Coroutine

코루틴은 우리가 잘 알고 있는 서브루틴(Subroutine)과 달리 진입점(Entry Point)이 여러 개일 수 있습니다.
쉽게 이야기하면 실행을 멈췄다가(Suspend) 재개(Resume)할 수 있다는 점인데요.
이 특성을 살리면 우리가 익히 아는 스레드(Thread)처럼 쓸 수 있게 됩니다.
다만 스레드와 달리 코루틴은 비선점적(Non-Preemptive)이기때문에
코드의 흐름을 전적으로 사용자가 제어할 수 있습니다.

spoqa 기술블로그 - Concurrency and eventlet(문성원님) 참조

Coroutine


A Routine
B Routine
Call
entry point
Suspend
Resume
entry point

Coroutine DEMO

코드 가독성이 좋아짐

대규모 처리시 안전함

비동기 다중 중첩시 유의해야 함

결과적으로 Happy함

TIPs

Coroutine = Generator + promise + dispatcher(trampoline)

UV_THREADPOOL_SIZE

Coroutine 예외 처리 방법

References

TOAST ES6의 제너레이터를 사용한 비동기 프로그래밍(김동우님)

spoqa 기술블로그 - Concurrency and eventlet(문성원님)

서광열의 코딩스쿨 - 코루틴(Coroutine) 이해하기(서광열님)

Wikiedia Coroutine 한글 번역(dogfeet님)

One more thing…

We want **you**!

https://**careers.kakao.com**/jobs/P-10465

Thank you

개발자라면 지금 방문하세요! developer.ibm.com/kr

https://github.com/imazine/playnode2017_sample