성능 테스트 및 튜닝

한태웅

코딩하는 오징어
나는 컴창이다

Mash up 4th ~ 7th

2018.07 ~ 2019.05 : 정산 플랫폼 개발

2019.05 ~ 현재 : 코어 플랫폼 개발

- Batch Application의 핵심 쿼리 성능 개선
- 전사 API Gateway JVM GC 튜닝
- IFTTT Trigger 시스템 성능 테스트 및 개선

목차

CPU Utilizing

➔ CPU Intensive한 작업들은 어떤 것들이 있을까??

- String 처리
- Json Serialize & Deserialize
- 정렬 작업
- 객체 생성 (new …)
- 쿼리 statement 생성 등등

Context Switching

- ➔ 다른 프로세스 OR 쓰레드에게 CPU점유를 이전해주기 위해 작업중인 것들(돌아오는 주소 포함)을 레지스터나 메모리에 저장한 후 CPU를 할당 시켜줘야 하는 프로세스나 쓰레드에게 리소스를 넘기는 작업

- ➔ 커널이 해줌

Context Switching

➔ 적절한 쓰레드 수를 찾는게 중요함. 너무 많은 쓰레드를 사용하게 된다면 비즈니스 로직을 처리해야할 CPU자원을 Context Switching에 소모하게 됨으로써 자원낭비를 유발함.

Cache & Memory

➔ 캐싱은 보통 로컬 메모리나 Redis, Memcached같은 메모리 스토리지에 데이터를 보관하여 Disk I/O에 병목을 줄인다. 하지만 캐싱 작업은 너무너무 어렵다!

➔ 캐싱되어있는 데이터는 웬만하면 건드리지말자! expire time을 지정해두고 자연스럽게 삭제되게하자

Disk I/O & Network I/O

➔ 메모리가 부족해지면서 swap이 발생하지 않는지

➔ Socket을 재사용하지 않는지

➔ File Descirptor 튜닝은 잘되어있는지

➔ 근데 Socket은 뭘까? IPC !! PIPE 파일, RPC, Socket통신

➔ https://tech.kakao.com/2016/04/21/closewait-timewait

Disk I/O & Network I/O

➔ 인덱스는 잘 걸려있을까?
➔ 불필요한 네트워크 호출은 없을까?
➔ 비동기 처리 할 수 있진 않을까?
➔ Non-blocking VS Asynchronous

Linux 모니터링 명령어

➔ vmstat

➔ iostat

➔ top -p {pid}

➔ netstat

➔ tip: netstat -nat | awk ‘{print $6}’ | sort | uniq -c | sort -n: 소켓의 상태를 기준으로 그룹핑 되어 각 상태의 소켓수를 화면에 표시

QA

➔ CPU Utilizing & Context Switching

➔ Cache & Memory

➔ Disk I/O & Network I/O

➔ Linux 모니터링 명령어

The End