Java OutOfMemory AtoZ

AWS ECS 환경의 메모리 이슈 해결 과정

배만진

목차

결론

원인

- Scale In & Out 정책 오설정

대응

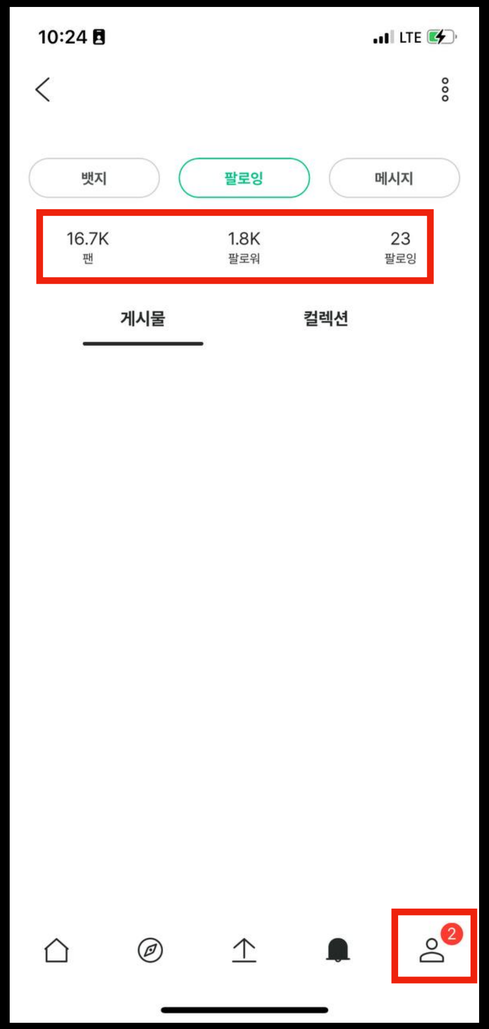
- 기존 인스턴스 대비 2배 증가 운영
- 서버 인스턴스 고정 운영

결과

- 모니터링 지표 상 안정적인 변화 확인
- OutOfMemoryError Log 미발생 확인

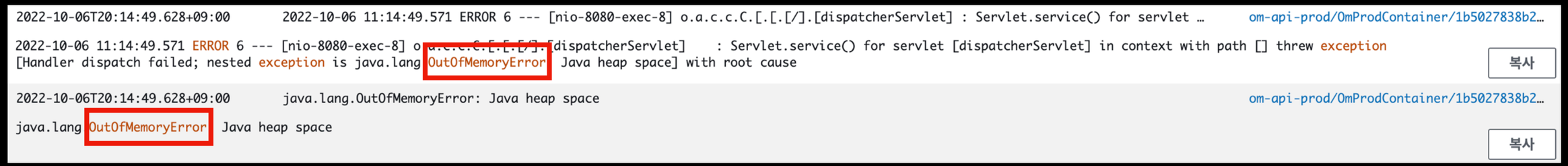
현상 조사

‘콘텐츠 목록 조회 API’ 간헐적 동작 이상



현상 조사

OutOfMemoryError Log 확인

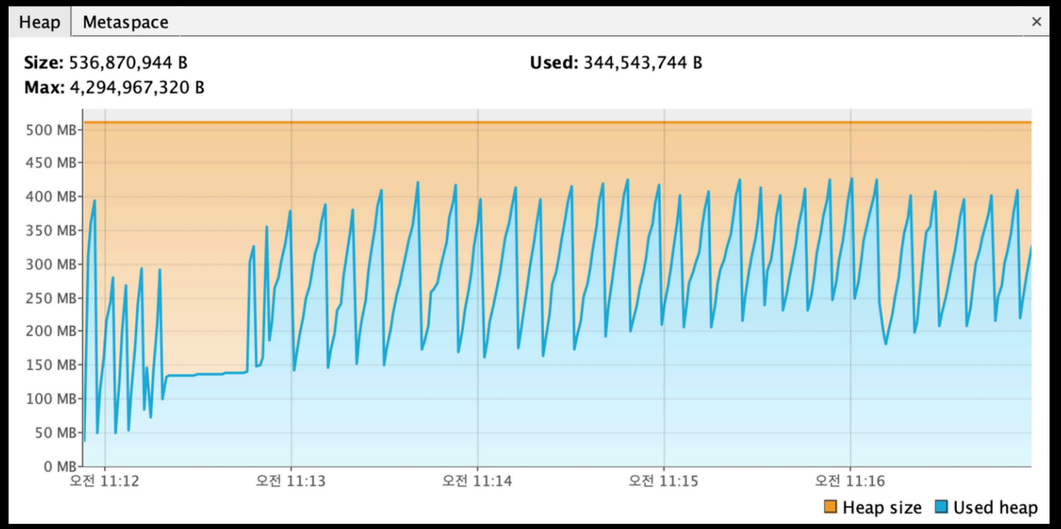


현상 조사

운영환경 의심

로컬환경 부하 테스트 결과
: Heap Memory 정상 관리에 따라 동일 현상 재현 불가
: 운영환경 이슈 추정

참고) 부하 테스트 환경
- 호출 대상 API: 메모리 소요 높은 목록 조회 API 다수
- number of threads: 2000
- JVM memory limit: 4GB



원인 분석

추정 원인

1. QueryPlanCache 오설정

2. Linux User Limit 오설정

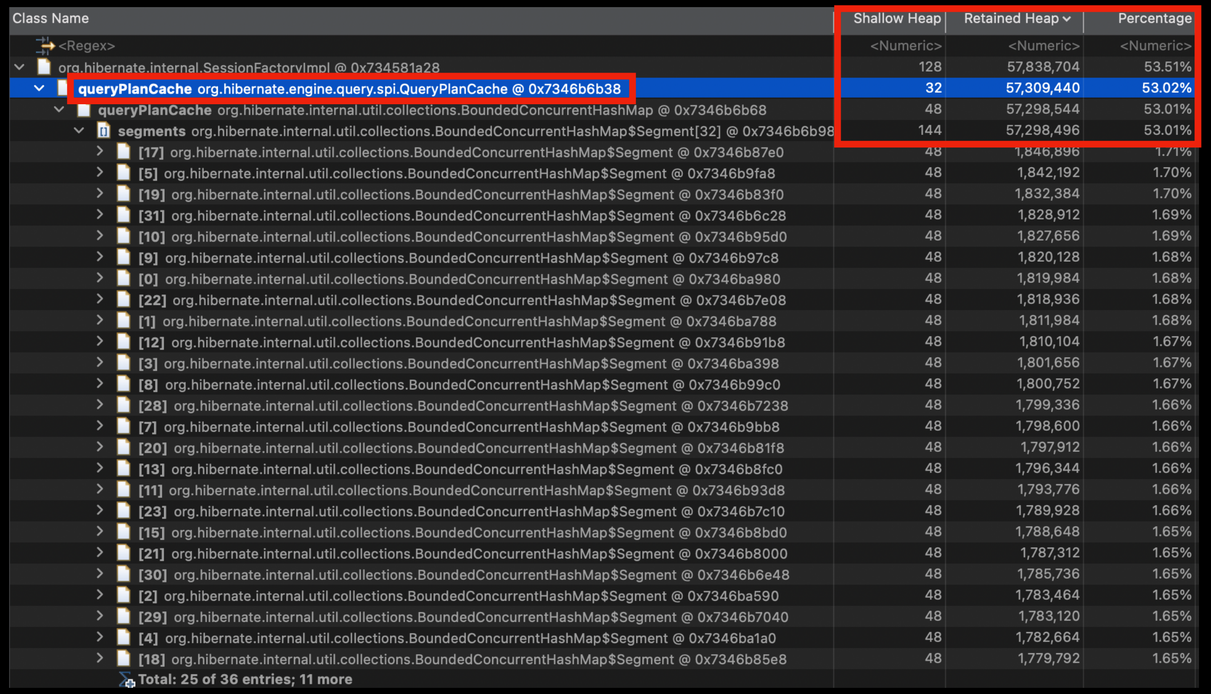
3. Scale In & Out 정책 오설정

원인 분석 QueryPlanCache 오설정

MAT 기반 메모리 분석

QueryPlanCache Class
: Shallow Heap 대비 Retained Heap이 높음
하위 Collection에도 50% 이상의 메모리가 할당됨

참고) 용어 정리
- QueryPlanCache: JPQL을 Native Query로 변환하기 전 Caching에 사용
- Shallow Heap: 객체 자체에 할당된 Heap Memory
- Retained Heap: Shallow Heap을 포함하여 객체에 할당된 모든 Heap Memory



Class Name	Shallow Heap	Retained Heap	Percentage
<Regex>	<Numeric>	<Numeric>	<Numeric>
org.hibernate.internal.SessionFactoryImpl @ 0x734581a28	128	57,838,704	53.51%
queryPlanCache org.hibernate.engine.query.spi.QueryPlanCache @ 0x7346b6b38	32	57,309,440	53.02%
queryPlanCache org.hibernate.internal.util.collections.BoundedConcurrentHashMap @ 0x7346b6b68	48	57,298,544	53.01%
segments org.hibernate.internal.util.collections.BoundedConcurrentHashMap$Segment[32] @ 0x7346b6b98	144	57,298,496	53.01%
[17] org.hibernate.internal.util.collections.BoundedConcurrentHashMap$Segment @ 0x7346b87e0	48	1,846,896	1.71%
[5] org.hibernate.internal.util.collections.BoundedConcurrentHashMap$Segment @ 0x7346b9fa8	48	1,842,192	1.70%
[19] org.hibernate.internal.util.collections.BoundedConcurrentHashMap$Segment @ 0x7346b83f0	48	1,832,384	1.70%
[31] org.hibernate.internal.util.collections.BoundedConcurrentHashMap$Segment @ 0x7346b6c28	48	1,828,912	1.69%
[10] org.hibernate.internal.util.collections.BoundedConcurrentHashMap$Segment @ 0x7346b95d0	48	1,827,656	1.69%
[9] org.hibernate.internal.util.collections.BoundedConcurrentHashMap$Segment @ 0x7346b97c8	48	1,820,128	1.68%
[0] org.hibernate.internal.util.collections.BoundedConcurrentHashMap$Segment @ 0x7346ba980	48	1,819,984	1.68%
[22] org.hibernate.internal.util.collections.BoundedConcurrentHashMap$Segment @ 0x7346b7e08	48	1,818,936	1.68%
[1] org.hibernate.internal.util.collections.BoundedConcurrentHashMap$Segment @ 0x7346ba788	48	1,811,984	1.68%
[12] org.hibernate.internal.util.collections.BoundedConcurrentHashMap$Segment @ 0x7346b91b8	48	1,810,104	1.67%
[3] org.hibernate.internal.util.collections.BoundedConcurrentHashMap$Segment @ 0x7346ba398	48	1,801,656	1.67%
[8] org.hibernate.internal.util.collections.BoundedConcurrentHashMap$Segment @ 0x7346b99c0	48	1,800,752	1.67%
[28] org.hibernate.internal.util.collections.BoundedConcurrentHashMap$Segment @ 0x7346b7238	48	1,799,336	1.66%
[7] org.hibernate.internal.util.collections.BoundedConcurrentHashMap$Segment @ 0x7346b9bb8	48	1,798,600	1.66%
[20] org.hibernate.internal.util.collections.BoundedConcurrentHashMap$Segment @ 0x7346b81f8	48	1,797,912	1.66%
[13] org.hibernate.internal.util.collections.BoundedConcurrentHashMap$Segment @ 0x7346b8fc0	48	1,796,344	1.66%
[11] org.hibernate.internal.util.collections.BoundedConcurrentHashMap$Segment @ 0x7346b93d8	48	1,793,776	1.66%
[23] org.hibernate.internal.util.collections.BoundedConcurrentHashMap$Segment @ 0x7346b7c10	48	1,789,928	1.66%
[15] org.hibernate.internal.util.collections.BoundedConcurrentHashMap$Segment @ 0x7346b8bd0	48	1,788,648	1.65%
[21] org.hibernate.internal.util.collections.BoundedConcurrentHashMap$Segment @ 0x7346b8000	48	1,787,312	1.65%
[30] org.hibernate.internal.util.collections.BoundedConcurrentHashMap$Segment @ 0x7346b6e48	48	1,785,736	1.65%
[2] org.hibernate.internal.util.collections.BoundedConcurrentHashMap$Segment @ 0x7346ba590	48	1,783,464	1.65%
[29] org.hibernate.internal.util.collections.BoundedConcurrentHashMap$Segment @ 0x7346b7040	48	1,783,120	1.65%
[4] org.hibernate.internal.util.collections.BoundedConcurrentHashMap$Segment @ 0x7346ba1a0	48	1,782,664	1.65%
[18] org.hibernate.internal.util.collections.BoundedConcurrentHashMap$Segment @ 0x7346b85e8	48	1,779,792	1.65%
Total: 25 of 36 entries; 11 more			

원인 분석 QueryPlanCache 오설정

QueryPlanCache 오설정이 원인?

NO

서버 메모리 : QueryPlanCache Max Size

= 4GB : 2GB

= QueryPlanCache 메모리 점유율 53%는 정상 범위

QueryPlanCache 비효율 사용이 원인?

NO

Padding 설정 후 QueryPlanCache 메모리 사용 양상 유사

원인 분석 Linux User Limit 오설정

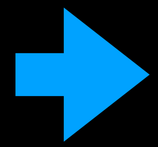
메모리 스펙 비례 ulimit 설정

서버 메모리 : ulimit

= 1GB : 3902

= 4GB : 10240

```
"ulimits": [{
    "name": "nofile",
    "softLimit": 100000,
    "hardLimit": 100000
}]
```



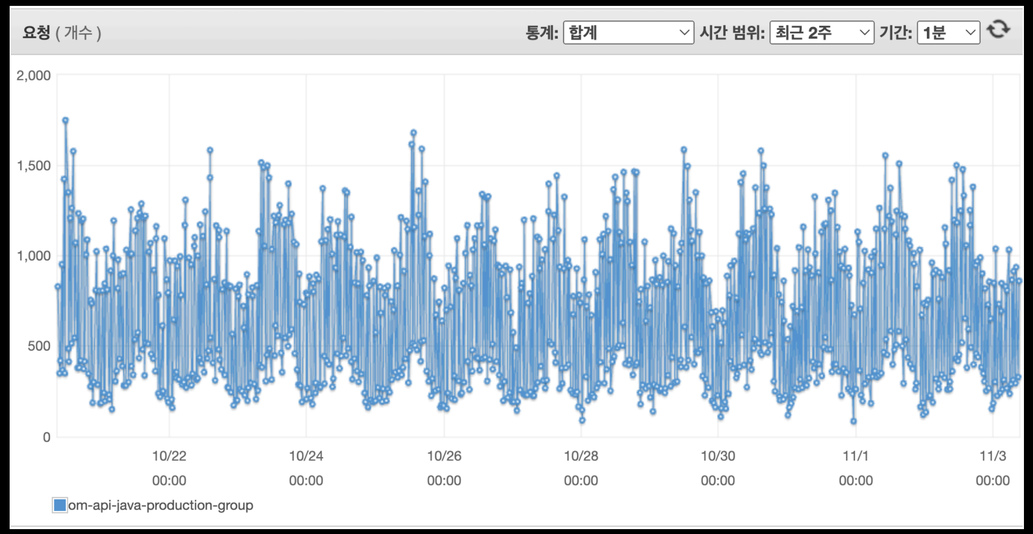
```
"ulimits": [{
    "name": "nofile",
    "softLimit": 10240,
    "hardLimit": 10240
}]
```

원인 분석 Linux User Limit 오설정

Linux User Limit 오설정이 원인?

NO

분당 최대 요청수 2000 미만
: 쓰레드 과생성에 따른 메모리 이슈 발생 가능성 낮음

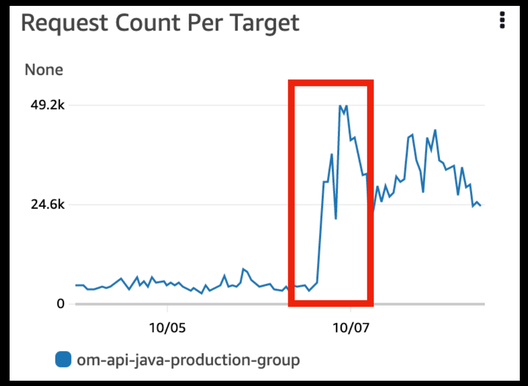


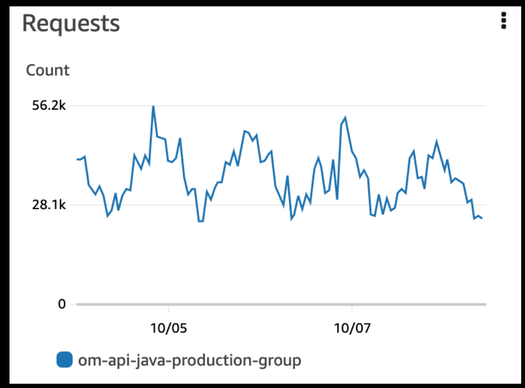
원인 분석 Scale In & Out 정책 오설정

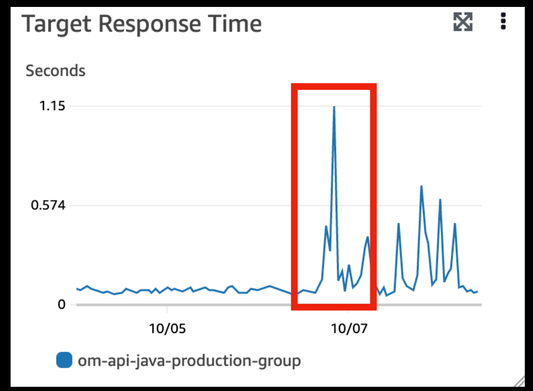
LB Monitoring 지표 분석

Requests VS Request Count Per Target
: 서버 인스턴스별 트래픽 급증에 따른 부하 급증

참고) 용어 정리
- Requests: LB ‘대상 그룹’에 대한 전체 요청 수
- Request Count Per Target: LB ’대상 그룹’ 내 서버 인스턴스별 요청 수
- Target Response Time: LB ‘대상 그룹’ 내 서버 인스턴스별 요청에 대한 응답시간







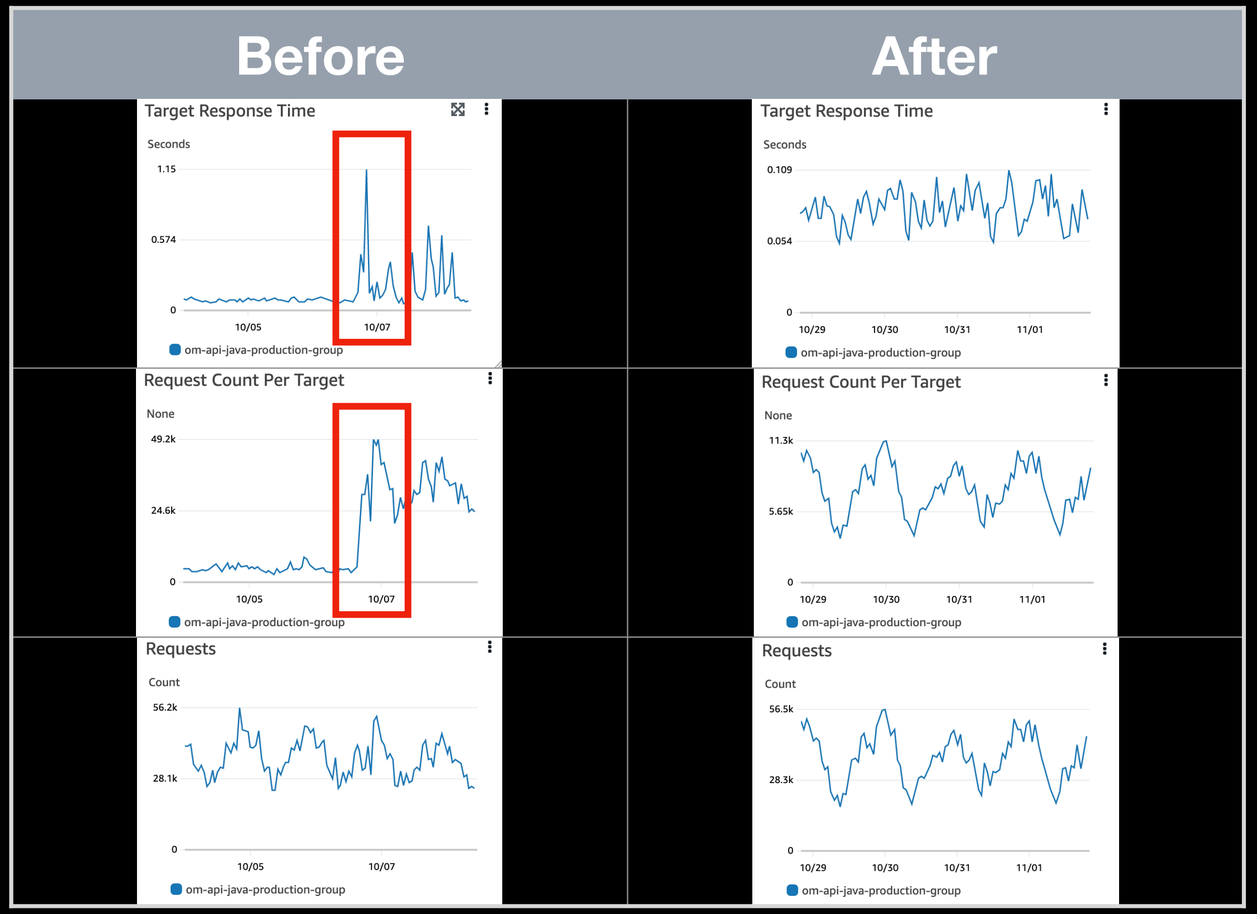
원인 분석 Scale In & Out 정책 오설정

Scale In & Out 오설정이 원인?

YES

Before
: CPU 사용률 기준 인스턴스 조절

After
: 서버 인스턴스 고정 운영 후 지표 정상화



향후 계획

1. Scale In & Out 기준 재정립

2. 모니터링 지표의 다양화

질의 응답

참고 자료

- Analyzing Memory Leak with Mat
  - https://www.cleantutorials.com/jconsole/heap-dump-analysis-using-eclipse-memory-analyzer-tool-mat
  - https://medium.com/@chrishantha/basic-concepts-of-java-heap-dump-analysis-with-mat-e3615fd79eb
  - https://thorben-janssen.com/hibernate-query-plan-cache/
- Debugging Memory Leak
  - https://techblog.woowahan.com/2628/
  - https://inside.getyourguide.com/blog/find-and-fix-oom-and-memory-leaks-in-java-services
  - https://meetup.toast.com/posts/211
- Setting User Limits
  - https://techblog.woowahan.com/2569/
- Setting Fargate SSH Access
  - https://medium.com/ci-t/9-steps-to-ssh-into-an-aws-fargate-managed-container-46c1d5f834e2