미들웨어 분야
Stack 통합 Test 결과보고서
[ActiveMQ]

2013. 10.

nipa 정보통신산업진흥원 National IT Industry Promotion Agency
nipa 공개SW역량프라자 Open Source Software COMPENTENCE PLAZA

목 차

I. 테스트 대상 소개

1. ActiveMQ

ActiveMQ는 아파치 소프트웨어 재단의 오픈 소스 프로젝트로, 기업 연동 작업을 쉽고 확장성있게 구현할수 있도록 지원하는, 고기능 JAVA 메시지기반 미들웨어(MOM) 이며 통합패턴(Intergration Patterns) 서버 임

- 주요 기능

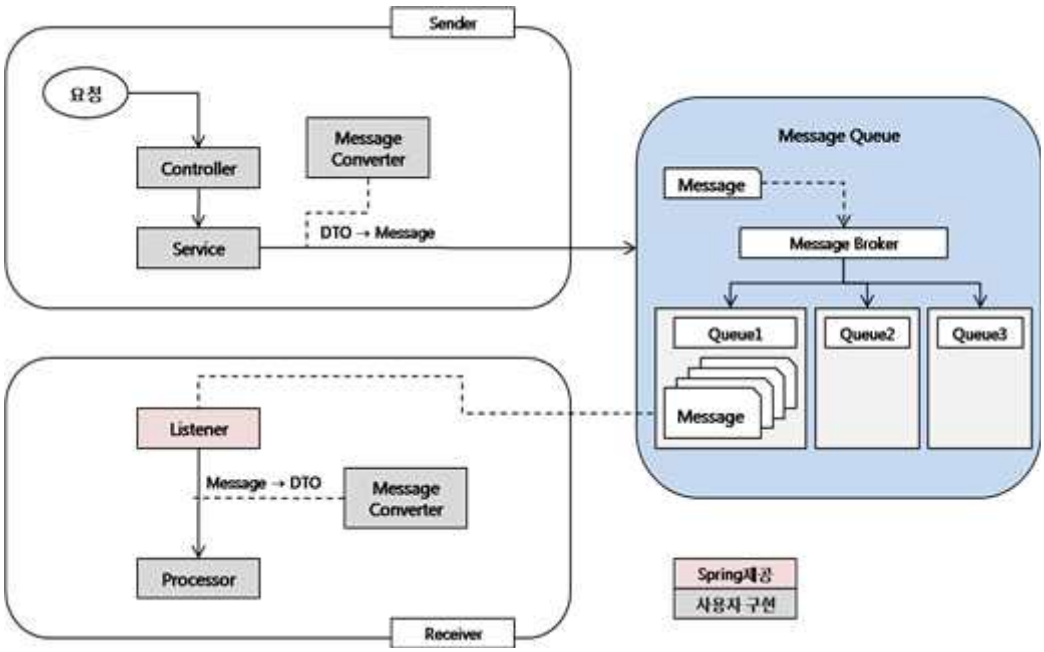
1.1 JMS Client와 메시지브로커 사이에서 기업통합패턴(EIP) 기능을 지원

1.2 다양한 프로토콜 지원

1.3 다양한 개발언어로 API 지원

1.4 강력한 기능의 메시지 브로커 기능

1.5 손쉬운 유지보수 관리 기능



[그림 I-1. Spring MVC 연동 ActiveMQ 구조도]

2 . ActiveMQ 기능

[표 I-1. ActiveMQ 기능 설명]

구분	기능
JMS준수	JMS 1.1 스팩을 준수하며, 동기와 비동기 메시지 전달과 1:1 메시지 전달, 구독자들에게 메시지 전달 보장을 제공.
연결성	ActiveMQ는 HTTP/S, IP multicast, SSL, STOMP, TCP, UDP, XMPP 등의 프로토콜의 지원을 포함하여 넓은 영역의 연결옵션을 제공. 넓은 영역의 프로토콜 등의 지원은 시스템에 유연성을 제공.
플러그 방식의 저장과 보안	ActiveMQ는 다양한 저장매체를 사용 가능. ActiveMQ의 보안성은 원하는 최상의 인증과 승인 정의 가능. (예: ActiveMQ는 KahaDB를 통해 초고속 메시지 저장방식을 제공하고 표준 JDBC접근방식 데이타베이스도 지원. 또한 속성파일을 사용하여 자체적인 간단한 인증과 승인을 지원 할 뿐만 아니라 표준 JAAS 로그인 모듈을 지원.)
다양한 어플리케이션간의 통합	WAS(Web Application Server)와 ActiveMQ를 통하여 다양한 어플리케이션간의 통합 제공. (예: Apache Tomcat, Jetty, Apache Geronimo, 그리고 JBoss같은 응용프로그램서버들의 통합을 제공)
다양한 언어의 클라이언트를 위한 API제공	Java 뿐만 아니라 C/C++, NET, Perl, PHP, Python, Ruby 등의 많은 언어를 위한 API를 제공. ActiveMQ브로커는 여전히 Java VM 환경에서 실행되지만 클라이언트는 지원되는 어떤 언어라도 사용하여 연동 가능.
브로커 클러스터링	다수의 ActiveMQ브로커는 확장성을 위해 브로커의 연합네트워크로서 함께 동작 함. 이것은 브로커 네트워크(Network of Brokers) 라고 알려져 있고 많은 다른 토폴로지를 지원할 수 있음.
많은 전문적인 브로커기능과 클라이언트 옵션	ActiveMQ는 브로커와, 브로커에 연결하는 클라이언트 모두에게 다양한 기능을 옵션으로 제공. ActiveMQ는 브로커의 XML 설정파일을 통하여 Apache Camel의 사용을 지원.
단순한 관리	JMX(Java Management eXtensions)를 통하여 JConsole이나 ActiveMQ웹 콘솔, ActiveMQ조회(advisory) 메시지처리, 명령줄 스크립트 등 다양한 측면의 모니터링 방법을 제공.

스프링(Spring) 지원	스프링을 지원하기 때문에 ActiveMQ는 쉽게 스프링 응용프로그램 안으로 포함될 수 있으며 스프링의 XML 설정 메커니즘을 사용하여 설정도 가능.
Ajax 지원	순수한 DHTML을 사용하는 웹 브라우저를 지원하기 위한 웹 스트리밍을 지원하는 Ajax, 웹 브라우저가 메시징구조의 일부가 될 수 있도록 지원.
CXF 와 Axis	CXF와 Axis 를 지원하기 때문에 ActiveMQ는 신뢰성 있는 메시징을 제공하기위한 웹서비스 스택 중 하나를 쉽게 적용될 수 있습니다.
고급기능지원	메시지그룹과 가상목적지, 와일드카드, 그리고 합성주소 (Composite Destination) 같은 고급기능을 지원
엔터프라이즈 통합패턴 지원	JMS클라이언트와 메시지 브로커, 양 쪽을 엔터프라이즈 통합패턴으로 연동 지원

3. ActiveMQ 특징

[표 I-2. ActiveMQ 특징 설명]

구분	내용
이기종 응용프로그램 통합	ActiveMQ브로커는 자바언어를 사용하여 작성되었지만 ActiveMQ는 C/C++, .NET, Perl, PHP, Python, Ruby, 그리고 기타 다른 언어들을 위한 클라이언트에도 API를 제공 - 다양한 클라이언트 API는 개발언어에 상관없이 ActiveMQ를 통하여 메시지를 보내고 받는 것이 가능 - ActiveMQ에서 교차언어능력뿐만 아니라 메시징이 진정으로 응용프로그램을 분리시키는 것을 돕기 때문에 RPC를 사용하지 않고 응용프로그램을 통합
RPC대용으로 사용	RPC방식의 동기화 호출을 사용하는 응용프로그램은 보편화되었음에도 불구하고 비동기메시징의 사용으로의 전환은 응답의 보장하면서 장시간 대기 해야 하는 메시지의 응답에 대한 결과 보장 - 비동기메시징 사용은 , 메시지가 동시에 소비되기 때문에 추가적인 메시지 수신자를 더 따르게 추가할 수 있고 더 빠르게 처리 함.
응용프로그램간	느슨한 결합 아키텍처는 뜻하지 않은 변경 사항처리에 대해 우수성을

느슨한 결합	가지고 있어 유지보수에 효율적임. 시스템의 한 컴포넌트에 대한 변경사항은 시스템전체에 걸쳐 파급되지 않을 뿐만 아니라, 컴포넌트의 상호작용에 대한 변경사항 또한 간소화 컴포넌트 상호작용(하나의 서로를 호출하는 메소드와 피호출자로부터 응답을 기다리는 호출자)을 위한 동기화 스키마 대신에, 컴포넌트는 비동기화 통신(간단히 메시지를 보내고 응답을 기다리지 않는 fire-and-forget 이라고 알려진)을 사용 가능.
이벤트기반 아키텍처의 백본	이벤트기반 아키텍처를 사용하여 커넥션오리엔트 보다 자원을 효율적으로 사용할 수 있고, 통합 속도를 개선할 수 있음. (예 : 아마존 같은 매우 높은 트래픽의 전자상거래 사이트 같이, 유저가 아마존에서 구매를 발생 할 때 재빨리 몇 개의 별도의 단계를 통해 그 주문은 순서대로 배치, 송장작성, 주문이행, 선적 등을 거치게 할 경우, 유저가 실제로 주문을 하면 유저는 즉시 "주문 주셔서 감사합니다"라는 페이지로 이동을 하고 하며, 유저는 주문이 접수되었다는 이메일을 받게 됨. 아마존에 의해 채용된 주문 배치 프로세스는 보다 더 비동기 프로세스의 좋은 예임. 주문의 매 단계는 별도의 서비스에 의해 분리되어 처리됨. 유저가 주문에 위치하면 주문을 승인하는 동기 호출이지만, 전체 주문 프로세스는 웹브라우저를 통해 동기호출처리를 사용하지 않고, 비동기 적으로 즉시 승인되고 발송이 됨. 나머지 단계의 프로세스는 비동기로 처리됨. 만약에 프로세스의 진행을 막는 문제가 발생되면 유저는 이메일을 통해 알림을 받게 됨. 이러한 비동기 프로세스는 확장을 여유롭게 하고 높은 가용성을 제공함.)
응용프로그램의 확장성 향상	많은 응용프로그램이 몇 가지 예를 들자면 전자상거래, 정부기관, 제조업, 온라인게임 등의 도메인에서 유용한 확장성을 제공하기 위해 이벤트기반 아키텍처를 사용. 비동기메시징을 사용하는 기업도메인의 라인을 따라 응용프로그램을 분리하여 다른 더 많은 가능성을 활용할 수 있음. 비동기 메시징을 통한 SOA의 백그라운드 연동 엔진으로 활용. 기업 사내 시스템의 정보를 통합하여 새로운 정보 생성하여, 서비스 통합 제공에 활용.

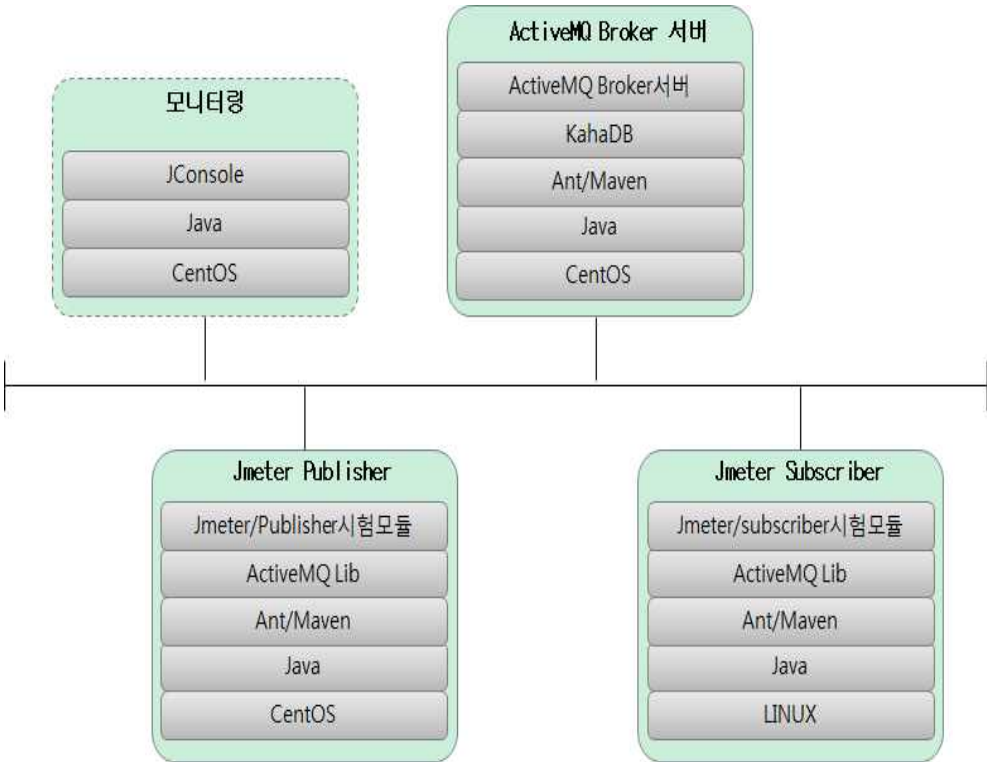
II. 테스트 환경

1. Stack 및 SW환경

[표 II-1. Stack 환경]

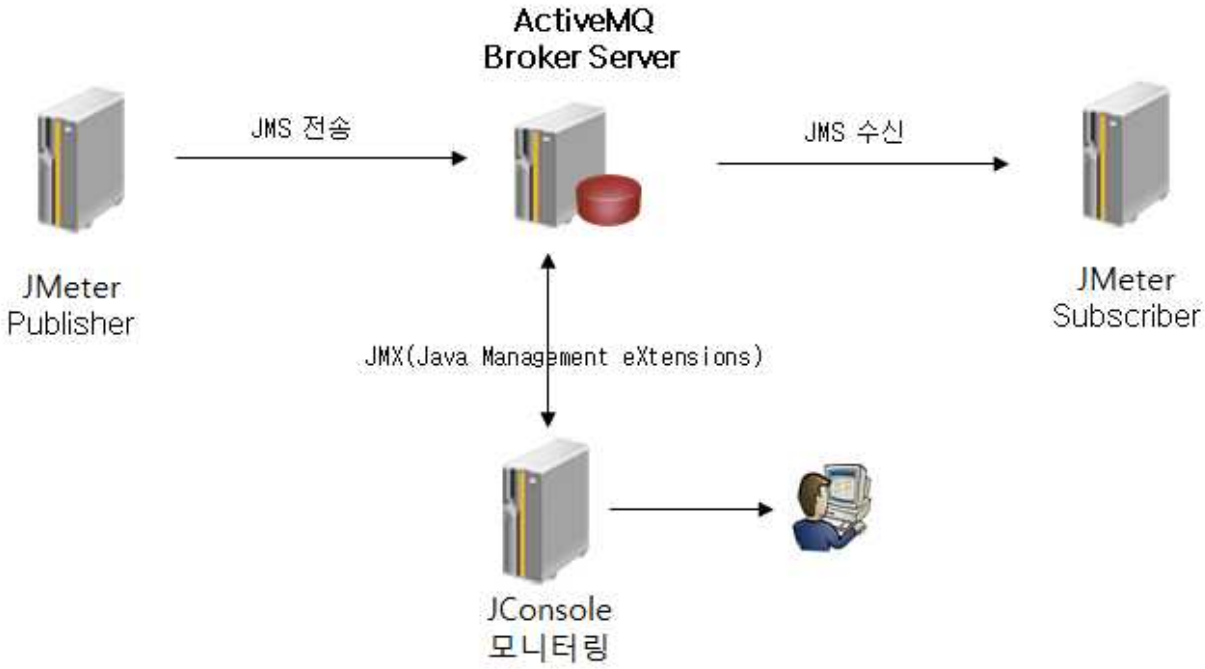
구분	솔루션명	버전
OS	CentOS	6.2 final
VM	Java	1.6.0_39
프로그램 설치	ant	1.9.2
프로그램 설치	maven	2.2.1
모니터링	JConsole	1.6.0
부하 생성	JMeter	2.9
JMS 서버	ActiveMQ	5.7

2. SW 구성도



[그림 II-1. SW 구성도]

3. HW 구성도



[그림 II-2. HW 구성도]

[표 II-2. 하드웨어 사양]

제조사	모델명	CPU	MEM	Disk	NIC	수량
IBM	X2550M2	Intel Xeon(R)CPU 2.40GHz	8GB	320GB	1Gigabit 1Port	x4

Ⅲ. 성능 테스트 수행 결과

1. 테스트 수행

□ 성능 테스트 방법

[표 Ⅲ-1. 성능 테스트 방법]

구분	내용
기능 테스트	ActiveMQ의 동작/종료, 모니터링에 대한 테스트
단위 테스트	단위 업무(기능)의 최대 성능 산출을 위한 테스트
통합 테스트	업무별 가중치를 부여해 실제 시스템 사용과 동일한 시나리오를 이용하여 목표 성능 테스트

□ 수행 조건

o 시스템 구성 : ActiveMQ서버 1대, JMS 전송 서버 1대, JMS 수신 서버 1대, 모니터링 1대.

o 연동 및 부하 시험에 측정결과를 정확하게 측정하기 위해 각 기능마다 서버를 분리하여 측정.

o 기능 시험은 1대의 서버가 100만 개의 메시지를 ActiveMQ 브로커에 보내고, 메시지를 수신한 ActiveMQ 브로커서버는 JMS 수신 서버에 브로커에서 보낸 메시지 전송.

o 부하 시험에 네트워크의 상태를 일정하게 하기 위하여, 1G 스위칭 허브를 사용하여 네트워크 구성 후 진행.(일반 인터넷을 사용 시 타 사용자와 사용이 동시 사용할 경우 네트워크 대역이 부족한 경우가 발생하여, 네트워크 구성을 일반 인터넷과 분리하여 시험)

o 부하 테스트는 가상전송서버 200대와 가상수신서버 200대를 동시에 접속하여 1억 개의 메시지 전송한 결과를 테스트함.

o 모니터링은 원격의 Jconsole에서 ActiveMQ Broker서버에 JMS연동 방법으로 추진하여, CPU, heap Memory 사용량, 쓰래드 수, classe 사용량, 초당 처리 건수, 전송 에러 건수를 측정.

o JMeter을 사용하여, 부하 시험을 진행.

□ 테스트 도구

[표 III-2. 테스트 도구]

도구	내용
JMeter	아파치 재단에서 제공하는 자바 플랫폼 기반 공개SW 성능 테스트 도구

□ 모니터링 도구

[표 III-2. 모니터링 도구]

도구	내용
Jconsole	Java기반 JMS 모니터링 도구
ActiveMQ Web Console	ActiveMQ에서 제공하는 기본 Web Console

□ 측정항목

[표 III-4. 측정항목]

항목	내용
응답시간	서버 통신 요청을 처리하기 위해 소요된 총시간
초당평균 처리건수	초당 연동 처리건수
에러율	연동 시 발생한 에러율
Heap Memory Usage	Activemq vm 메모리 사용량

Threads	Activemq 총 사용 Thread수
Classes	Activemq 총 사용 Classe수
CPU Usage	Activemq 사용 CPU 사용량

□ 용어설명

[표 III-5. 용어설명]

항목	내용
동시접속 서버 (Concurrent Server)	서버 시스템에 접속하여 사용하고 있는 모든 접속 서버를 의미하며, 동시 동작을 하기 위한 시험 시간중 접속을 유지하는 서버의 수를 의미
전송 가상 서버 (Active Server)	테스트 툴에서 생성한 쓰레드 수(사용자)로, 실제 서버에 요청을 하거나, 요청에 대한 응답을 올 때까지 대기하고 있는 서버의 행동을 시뮬레이션 하는 서버
평균 전송건수 (Response Per Seconds)	초당 전송 또는 수신 하는 메시지 건수
총 전송 건수 (Total Message)	단위 또는 부하시험에 사용한 전송 건수
총 소요시간 (Total Time)	총 전송건수에 대한 소요 시간

2. 테스트 결과

□ 기능 테스트 결과

o 정상적인 스타트업/상태 및 종료 수행.

o 서버상태 점검을 위한 Web Console 정상 동작 확인.

o JConsole을 이용한 ActiveMQ 상태 모니터링 결과 정상동작 확인.

□ 단위 테스트 결과

o 1대의 JMS 메시지 전송서버에서 ActiveMQ Broker Server로 100만 건의 메시지 전송.

o ActiveMQ Broker Server에서는 수신된 100만 건의 메시지를 원격의 JMS 수신서버에 중계.

[표 III-7. 단위 테스트 결과]

총 전송 건수	총 소요 시간	평균 전송시간(초)	서버
100만건	전송 : 1분 38초(98초)	약 10,204.8	전송 1대 수신 1대
	수신 : 1분 40초(100초)	10,000	

o 100만 건 전송에 1분 36초 수행(평균 초당 10,000 전송)하여 매우 고속의 메시지 중계를 수행함.

o ActiveMQ Broker의 CPU 사용은 5% 정도로 적은 리소스 사용을 보임

o 전송서버와 수신서버의 총소요시간이 2초 차이가 발생, 이는 activemq에서 수신서버로 전송 시 발생한 지연시간으로, 100만 건당 2초의 지연시간을 1건으로 환산할 경우 1건당 0.002ms의 지연으로, 사용에 무리가 없는 것으로 판단됨.

□ 부하 테스트 결과

o 전송 JMeter 1대에서 200명의 가상서버를 생성 후 1억 개의 JMS 메시지를 ActiveMQ Broker Server로 전송.

o ActiveMQ Broker Server는 1억 개의 메시지를 수신 후 원격의 가상서버를 동작하는 JMeter서버에 메시지 중계.

[표 III-8. 부하 테스트 결과]

총 전송 건수	총 소요 시간	평균 전송건수(초)	가상 서버
100,000,000	전송 : 4시간 20분 36초 (15,636초)	6,259.9	전송 : 200 수신 : 200
	수신 : 4시간 28분 58초 (16,138초)	6,236.0	

o 1억 건 전송에 4시간 36분 수행(평균 초당 6,259.9 전송)하여 고속의 메시지 중계를 수행함.

o ActiveMQ Broker의 CPU 사용은 평균 10.65%의 적은 리소스 사용.

o 1대의 가상서버에서 보낸 것보다 200대의 가상서버에서 보낸 것이 평균 전송시간이 많이 느린 것으로 확인 되었고, 이는 ActiveMQ 설정을 통해 튜닝이 필요하다는 것을 확인 할 수 있음.

o 평균 전송시간은 전송이 수신보다 초당 23.9 개 더 전송 함.

o 가상서버의 전송은 1억 개 중 1억 개가 모두 성공.

o 가상서버의 수신은 1억 개 중 약 2,509건(0.0025%)의 미 수신 건수가 Web Console에서 확인.

o ActiveMQ에서 미 전송 건수에 대한 이력을 가지고 있어 재 전송 요청 시 전송이 완료 되지만, JMeter상에서는 재전송 요청 로직이 없어 수신을 못한 것으로 판단이 됨.

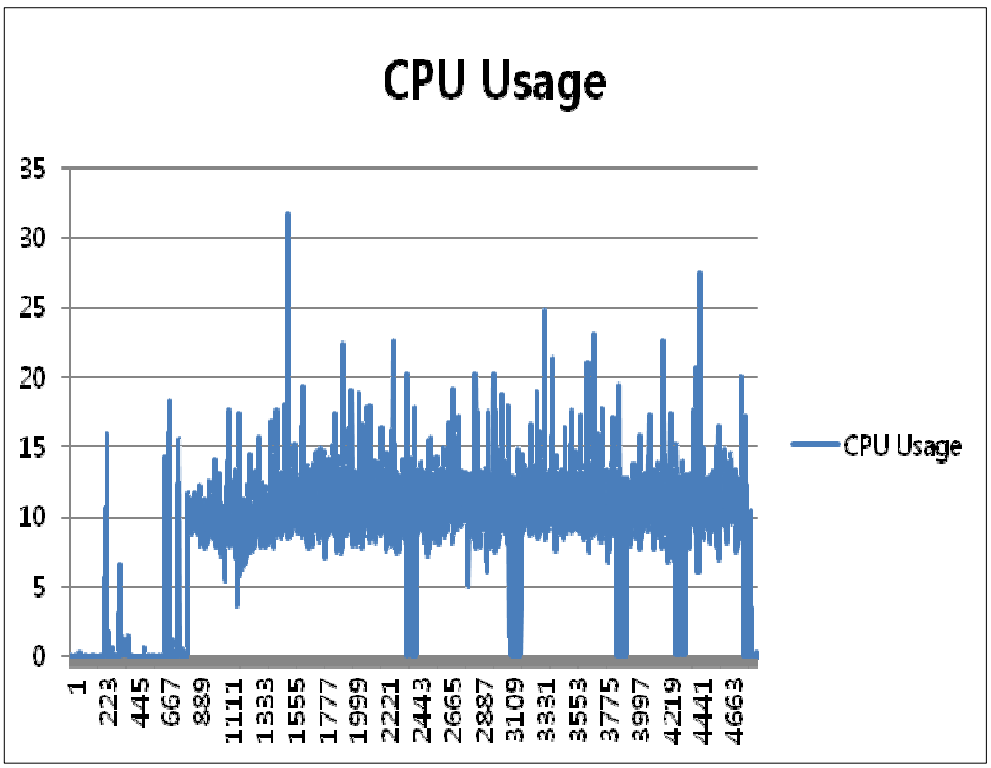
o 시험 결과 ActiveMQ에서 다중서버의 연계 처리 시, 성능 최적화에 대한 튜닝이 필요하며, 향후 추가 시험이 필요함.

4. 테스트 상세 내용

□ 부하 테스트 모니터링 결과

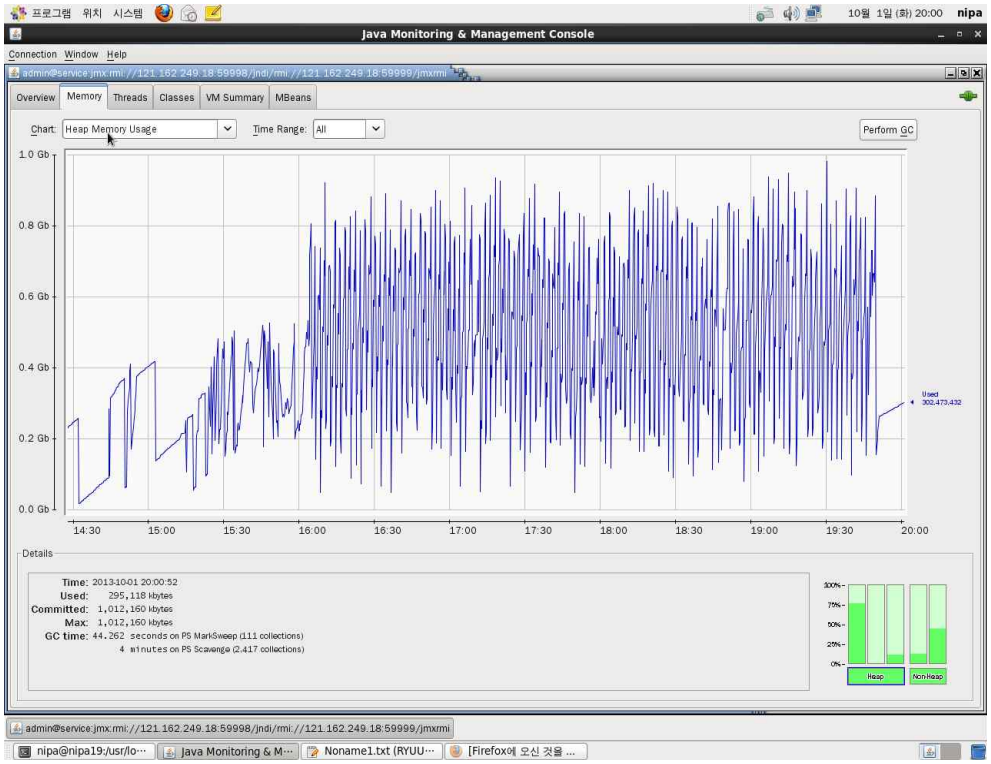
수행 조건	- 가상서버 : 400명(송·수신 포함) - 총 메시지 수 : 전송 1억 건, 수신 1억 건 - 동작시간 : 4시간 20분 36초

o 모니터링 결과



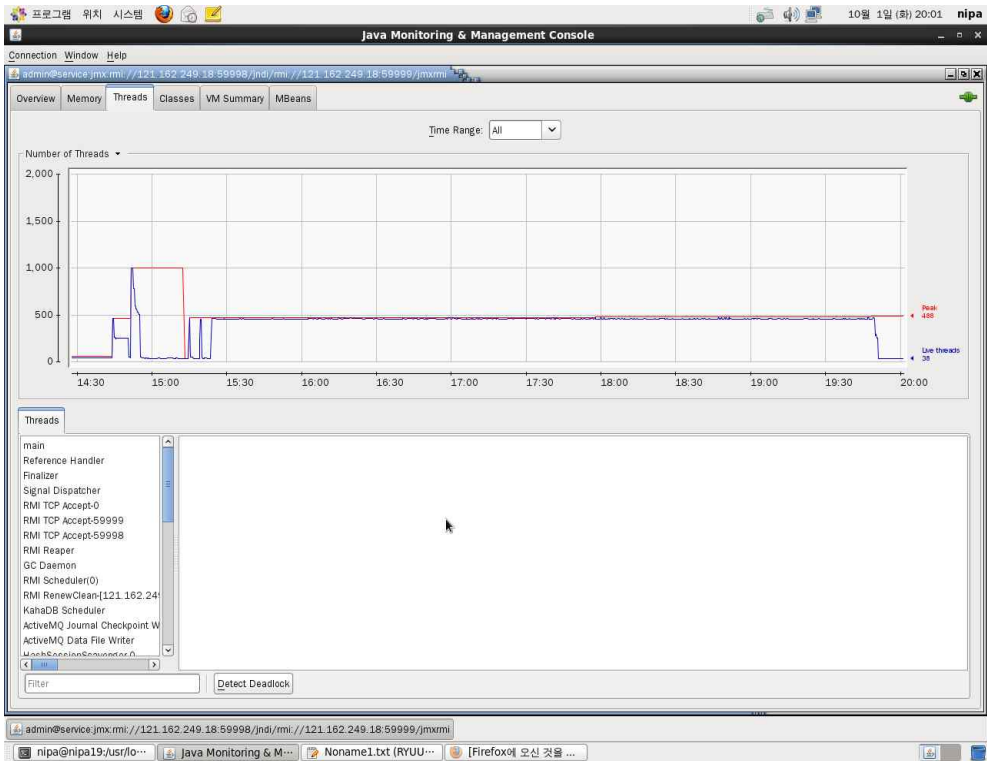
[그림 III-1. CPU 사용률]

- 전체 1억 건의 메시지 전송 중 CPU가 가장 많이 사용되었을 때도 35% 미만으로 안정적으로 사용되는 것을 알 수 있다.

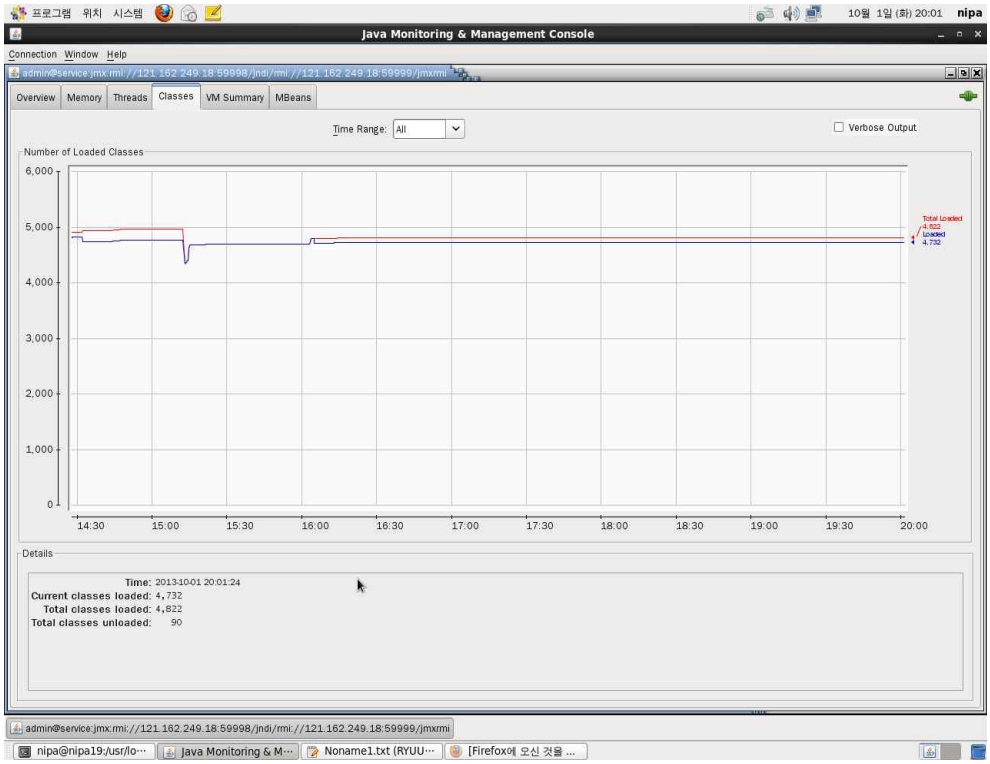


[그림 III-2. 메모리 사용률]

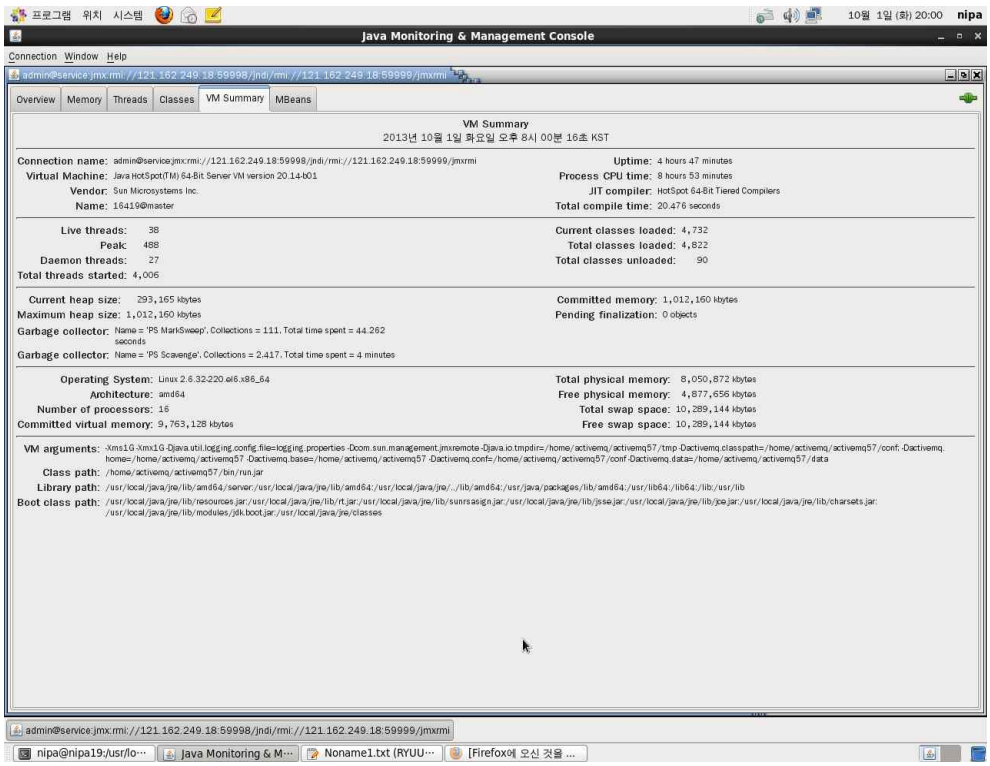
- 자바 Heap 메모리 사용량은 1GB 이내의 사용률을 보임



[그림 III-3. Threads 사용률]

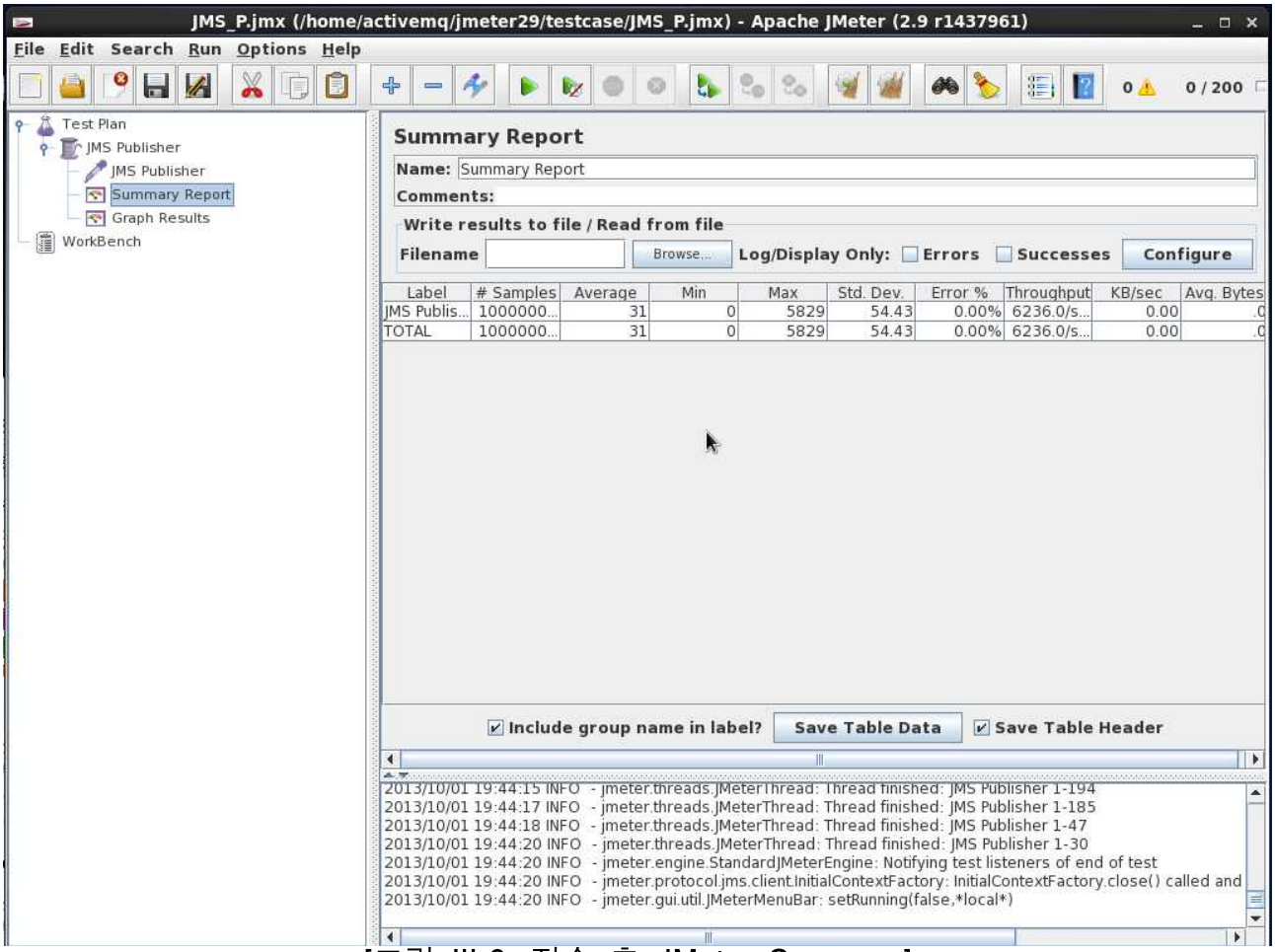


[그림 III-4. 초당 Classes]



[그림 III-5. 모니터링 Summary]

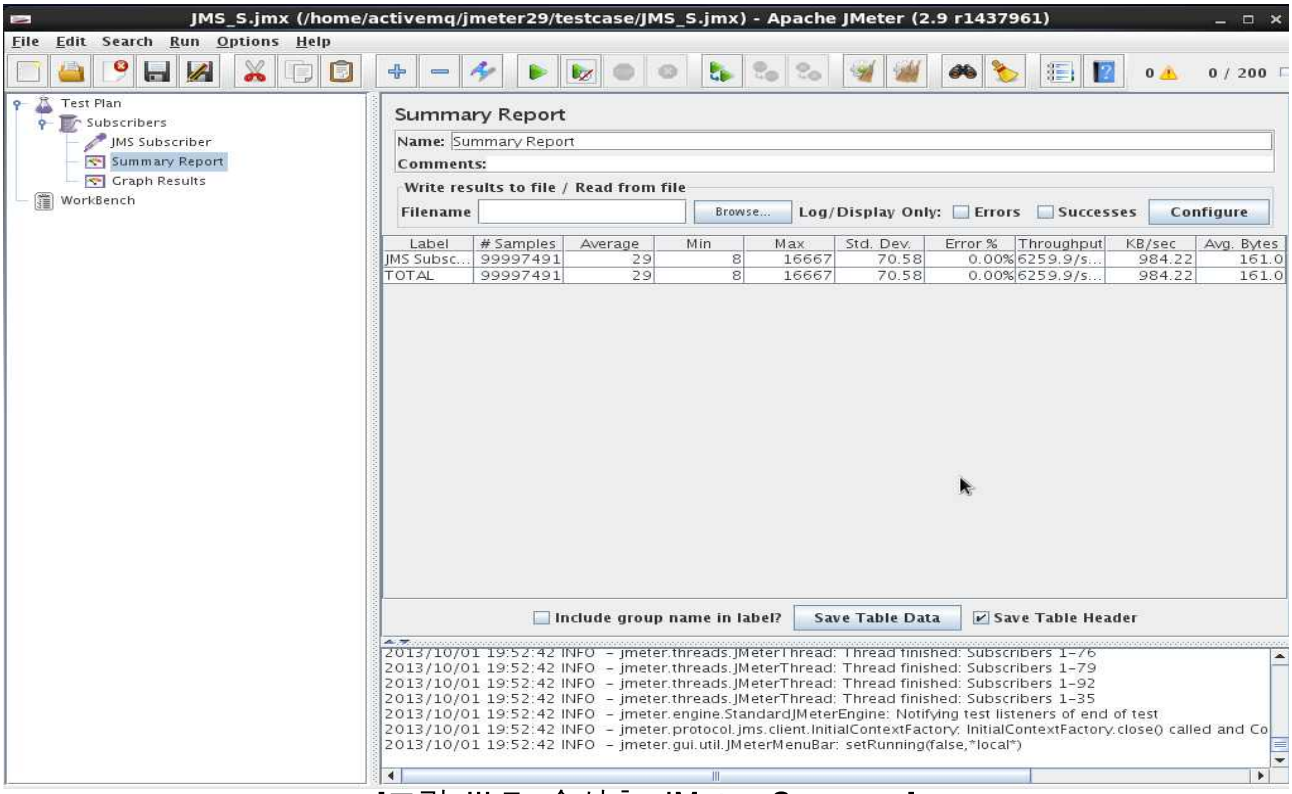
o 전송측 Summary Report



[그림 III-6. 전송 측 JMeter Summary]

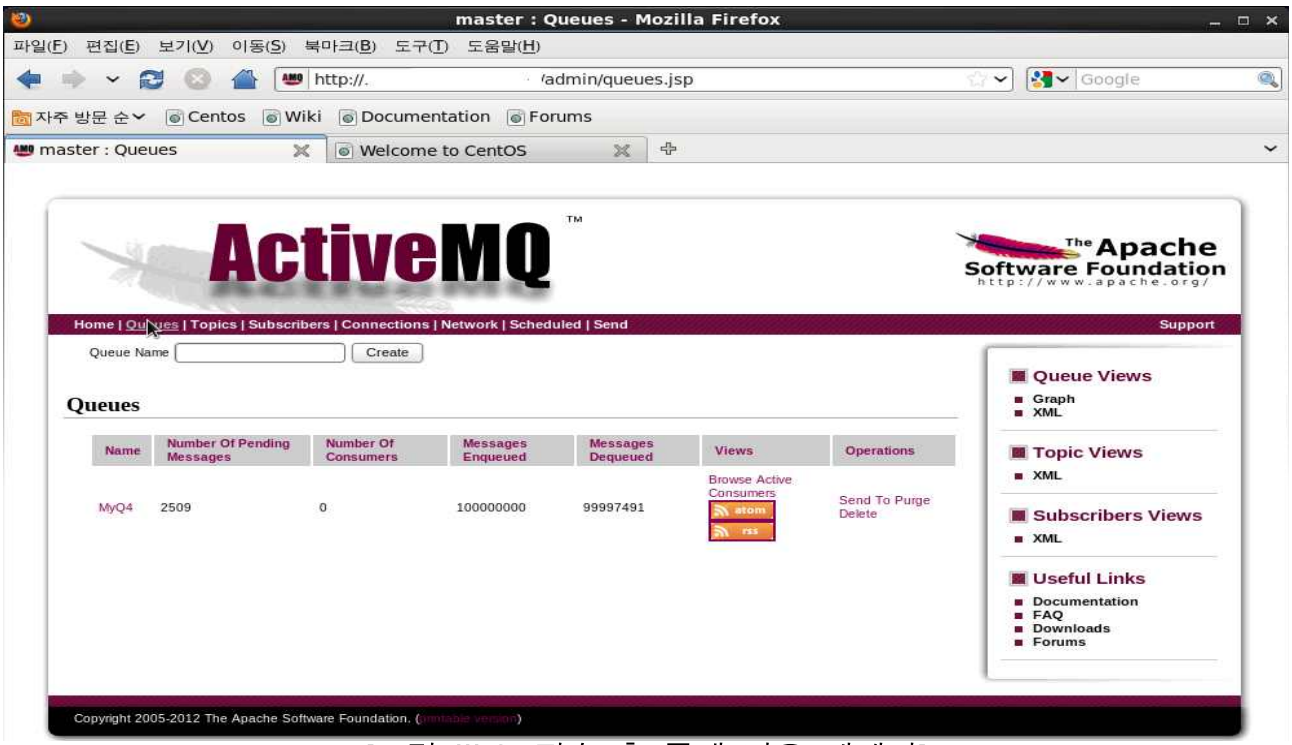
- 전송 측에서 1억 건의 메시지를 100% 완료 하였다.

o 수신측 Summary Report



[그림 III-7. 수신측 JMeter Summary]

- 최종 샘플은 99,997,491 건으로 1억 건 에 못 미치는 결함이 발생되었다.



[그림 III-8. 전송 후 큐에 남은 메세지]

Ⅳ. 종합

□ JMS 부하 시험은 JMeter를 사용하여 400(전송서버 200대, 수신서버200대)대의 가상 서버가 연동되는 것으로 부하 시험을 수행 및 측정이 용이함.

□ 서버의 사양에 따라 성능차가 있지만, 시험서버 기준으로 One user가 100만 건을 보낸 경우 초당 1만 건의 처리 가능.

□ 부하 시험은 총 1억 건을 200대의 서버가 동시에 접속하는 형태로 시험하여, 평균 초당 6,259건을 처리, 이때 평균 CPU 사용량은 10.65% 사용.

□ ActiveMQ의 동시접속 200대의 경우 연동처리에 초당 6,236건의 처리 능력과, 낮은 CPU사용을 하고 있어 안정적인 성능을 보여 주고 있음.

□ 향후 ActiveMQ의 테스트에는 CPU 더 사용량을 줄이고, 동시접속 처리 건수를 늘릴 수 있도록 튜닝이 필요함.

□ ActiveMQ의 차기버전(5.9)은 하둡을 사용한 이중화를 하고 있어, 버전업 이후에 기능 및 성능 테스트가 필요.