한국형 공개SW 표준 플랫폼 BOOYO 기반의 WAS 구축 및 SPECjAppServer2004 벤치마킹 연구

김리라[O], 김두현, 정성인*
건국대학교 인터넷미디어공학부, 한국전자통신연구원*
{otut, doohyun}@konkuk.ac.kr, sijung@etri.re.kr

A Study of SPECjAppServer2004 Benchmarking on WAS based on Korea Open Source Platform: BOOYO

Li-la Kim[O], Doo-hyun Kim, Sung-In Jung*
Dept. of Internet and Multimedia Eng., Konkuk Univ, and ETRI*

요 약

본 논문은 공개소스 기반의 WAS 구성의 핵심인 Tomcat, JBoss, PostgreSQL, Apache 등을 한국형 공개SW 표준 플랫폼 BOOYO 상에 효과적으로 탑재하고 SPECjAppServer2004 벤치마크를 실시하여 외산 제품과 성능을 비교 한다. 벤치마크 결과 부하가 증가할수록 성능 측정단위인 JOPS가 기존의 공개SW 플랫폼에 비하여 현저히 덜 하락하는 결과를 보이며 비교적 우수한 결과를 나타내었다. 본 논문에서는 이러한 벤치마크를 실시하기 위하여 구축한 시험용 WAS의 구성 내용과 아울러 그 결과를 보다 상세히 기술한다.

1. 서론

최근 들어 세계 S/W 산업은 소수의 업체들이 세계 시장의 70%를 지배하는 독점하고 있는 상황이며, 이러한 불균형을 해소하고자 각국에서 공개 S/W에 대한 관심이 고조되고 있다[1].

BOOYO란 데스크탑과 서버 분야에서 리눅스 2.6을 기반으로 하는 한국형 공개 S/W 표준 플랫폼이다[2]. BOOYO의 개발 목적은 소수의 업체들이 세계 S/W시장을 독점하는 현상 문제점을 극복하고, 국산 리눅스 OS 배포판 사용을 확대하여 진정한 소프트웨어 강국으로 나아가기 위한 것으로 이 중에 미들웨어 분야는 현재 세계적으로 가장 많은 사용자를 확보하고 있는 Tomcat[3], JBoss[4], PostgreSQL[5], Apache[6]를 BOOYO에 탑재한다. 이는 보다 보편적인 시스템을 구성함으로써 초기부터 다수의 사용자와 응용 분야를 확보하고자 하고 있다.

본 논문에서는 이러한 BOOYO의 공개소스 기반 WAS 미들웨어에 대하여 벤치마킹 툴인 SPECjAppServer2004[7]를 이용하여 벤치마킹을 실시하고, 그 결과를 외산 제품인 RedHat, RHEL, Suse Linux와 비교하고 그 결과를 분석한다[8].

2. BOOYO RC2의 특징

2.1 Ext3 Block Reservation 성능 개선

리눅스 커널 2.6.10 이후 버전에서는 ext3 파일시스템의 default block allocator에 block reservation 기능이 포함되어 있다. Block reservation 기능이란, inode별로 지정된 개수의 연속된 디스크 블록을 미리 예약해두어 필요 시 이를 이용하여 파일의 fragmentation 및 interleaving 현상을 방지함으로써 성능을 향상시키는 기법으로 파티션 mount시에 옵션으로 enable / disable할 수 있다. 하지만 AIM7[9] 벤치마크 시험을 통해 경우에 따라 오히려 block reservation 기능이 성능 저하를 야기시킬 수도 있음을 발견하였고, 이와 관련된 커널 코드를 수정하여 이로 인한 성능 저하 루틴을 피해서 작동할 수 있도록 수정 및 개선하였다.

2.2 Access Key Retention 관련 성능 개선

리눅스 커널 내의 인증 토큰 및 access key 정보를 저장 및 관리하는 기능으로, NFS 및 암호화 기능 등을 위해 키 값과 프로세스와의 연동 방식에 대한 provision을 포함한다. 커널 2.6.10에서 본 기능은 default로 enable되어 있으나 AIM7을 이용한 벤치마크 시험 도중 커널 프로파일링 작업을 통해 본 기능의 사용으로 인한 오버헤드가 상당함을 발견하였고, 이를 disable시킴으로써 성능 향상을 도모하였다.

2.3 Force Unmount Lock 문제 개선

파일시스템에 대한 force unmount를 실행하면 해당 파일시스템과 파일시스템에 속한 파일에 대한 모든 시스템 콜이 차단된다. 단, close 함수 만이 유일하게 실행 가능하다. 하지만 force unmount가 커널의 파일 객체에 대해 참조 중일 때, 다른 프로세스가 그 파일에 대한 close를 호출하여 파일을 닫으면 커널의 파일 객체가 갑자기 사라지므로 force unmount의 실행에 심각한 오류가 발생할 수 있다. 따라서 이러한 문제를 해결하기 위하여 close_sem이라는 세마포어를 사용하여 critical section을 설정한다.

기존의 force unmount 구현에 있어서 close_sem에 의해서 locking 된 critical section이 불필요하게 넓은 범위로 설정된 부분이 있었다. 파일시스템에 대한 성능 테스트를 통하여 sys_close()와 sys_dup2() 함수 내에 이러한 부분이 포함되어 있고, 그로 인하여 다중 프로세스가 파일시스템에 접근할 때 파일시스템 성능이 저하됨을 알 수 있었다. 따라서, critical section을 최소한의 범위로 축소시킴으로써 성능을 개선하였다.

3. 벤치마킹 시스템 구성

3.1 시스템 환경

SPECjAppServer2004 벤치마크는 실제 비즈니스 환경에서의 J2EE 애플리케이션 서버의 성능을 측정하는 새로운 산업표준 벤치마크로 자동차 제조업자와 그와 관련된 딜러의 교환을 수행하는 어플리케이션이다.

SPECjAppServer2004는 벤치마크 실행 시작과 부하를 발생시키는 딜러 드라이버와 제조업 드라이버로 구성된 Load Driver와 부품의 발주를 시작하는 Supplier 도메인을 실행시키는 Supplier Emulator, 그리고 SPECjAppServer 본체로 구성된다. 본체가 동작하는 환경은 SUT(System Under Test)라고 불리며 AP 서버/컨테이너, DB, 네트워크를 총체적으로 포함한 테스트 환경을 말한다. Load Driver 및 Supplier Emulator는 SUT 외부에서 실행해야 한다. System Under Test(SUT)는 <그림 1>과 같다.

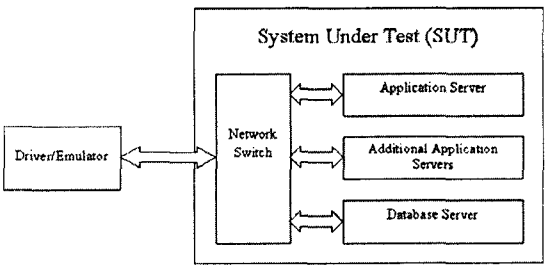

그림 1. SPECjAppServer2004 구성도

SPECjAppserver2004 어플리케이션은 도메인이라고 불리는 dealer, customer, manufacturing, supplier, corporate 5개의 비즈니스 영역으로 정의되어있다. 도메인간의 관계 및 시나리오는 <그림 2>에 나타냈다.

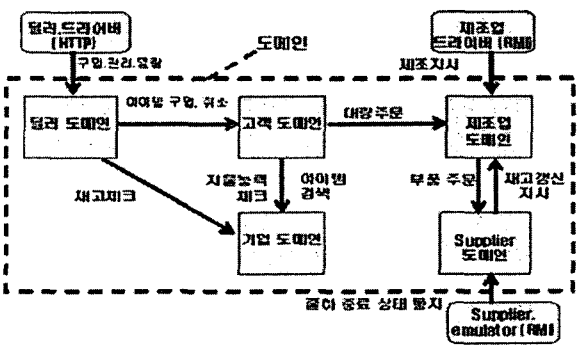

그림 2. SPECjAppServer2004 어플리케이션 도메인

SPECjAppserver는 WEB 트랜잭션과 EC 트랜잭션, 2개의 트랜잭션에 대해 규정하고 있다. Web 트랜잭션은 AP의 Web 부분의 request로 딜러 도메인에 입력되는 것이며, EC 트랜잭션은 EJB의 리모트 메소드 호출로 딜러 도메인을 제외한 4개의 도메인에 입력되는 것이다. 그리고 이 두 개의 트랜잭션을 조합하여 비즈니스 트랜잭션이라고 불리는 작업 단위로 정의한다. 이는 JOPS(jAppServer Operation Per Second)라고 불리며 일반적인 TPS(Transaction Per Second)에 상당하는 것이다.

JOPS = Dealer Transactions/sec + WorkOrders/sec

여기서 JOPS는 단위시간(1초간) 당에 딜러 도메인으로 실행되는 비즈니스 트랜잭션과 제조업 도메인으로 실행되는 작업 주문(워크·오더)의 총수로서 산출되며, 각 비즈니스·트랜잭션(transaction)의 타입에 따라 결과 리포트를 딜러 드라이브에 출력된다.

3.2 시스템 구축

<그림 3>은 이상에서 설명한 벤치마킹 구성도에 따라 이를 실현하기 위하여 구성한 네트워크의 구성도이다.

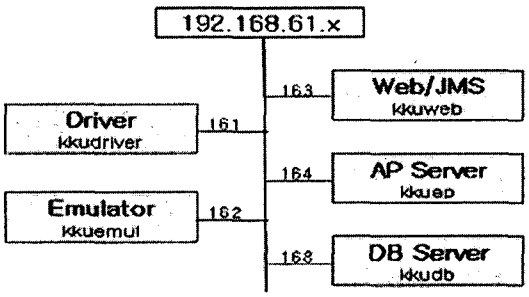

그림 3. 네트워크 구성

Driver, Emulator, Web/JMS, AP Server, DB Server는 하나의 공유기에 연결되어 있으며, AP Server는 한 대로 구성된다. 다음은 <그림 3>과 같이 서로 통신이 가능하도록 네트워크를 구성한 후에 실시하는 과정이다.

- 1단계: DB Server에는 PostgreSQL를 설치하고, AP에는 JBOSS를 설치한다. 그리고 Web에는 Apachehttpd의 OSS로 구축한다. - 소프트웨어 인스톨.설정
- 2단계: 데이터베이스의 각종 파라미터를 결정하여 SPECjAppServer2004가 사용하는 데이터를 이용 가능한 데이터베이스로 작성한다. - 데이터베이스 초기화
- 3단계: SPECjAppServer 2004 어플리케이션을 빌드하여 AP 서버 등의 배치를 실시한다. - 어플리케이션 빌드.배치
- 4단계: DB 서버, JMS 서버, AP 서버, WEB 서버, Emulator 순으로 각 서버 프로세스를 실행시킨다. - 서버 실행
- 5단계: 작성한 데이터베이스에 데이터를 넣는다.
- 6단계: 측정 조건 파라미터인 txRate(IR), rampUp, stdyState, rampDown, triggerTime를 설정한 후, driver 쉘을 실행시켜 벤치마크 측정 프로그램을 시작한다. txRate(IR)는 AP서버에게 주는 부하를 나타내는 값이며 디폴트 값으로 1로 설정되어 있다. - 벤치마크 측정 실행
- 7단계: 측정 후 정상적인 종료 유무와 처리 성능, 응답 시간 등의 결과를 확인한다. - 벤치마크 측정 결과 출력
- 8단계: 측정 결과가 출력이 되면 실행 중인 서버 프로세스 모두 정지시킨다. - 서버 프로세스 정지

벤치마크 측정 프로그램 종료 후, Driver 디렉토리 내에 측정 결과 파일들이 생성된다. 이 측정 결과 파일들은 모두 텍스트 파일 형식이다. 단, 벤치마크 측정을 도중에 정지하거나 이상 종료 시에는 파일이 생성되지 않는다. SPECjAppserver.summary 파일은 벤치마크 측정의 전체적인 결과 요약이며 위에서 언급한 종합적 지표인 JOPS의 수치를 포함하고 있다.

4. 실험 결과

본 논문에서 수행한 실험은 SPECjAppserver 벤치마킹을 통해 여러 OS간의 성능 차이를 밝히려는 것이다. 테스트의 하드웨어 측면의 제반 사항은 동일하며, OS의 차이만을 두었다. 테스트는 각 OS당 txRate를 10단계로 세분화하여 실시하였으며, 다음과 같이 도표를 이용하여 변화 경향을 나타내 보였다.

<그림 4>은 txRate와 transaction/sec으로 나타낸 것이다. 즉, 초당 transaction의 발생 빈도를 의미하며 증가하는 txRate에 따라 일정한 상승 곡선을 그리는 것이 이상적인 결과라고 볼 수 있다. txRate 15까지는 각각의 OS의 txRate의 증가분에 대해 일정한 transaction 증가를 보이나, txRate 15를 기점으로 상승폭이 줄어들고, txRate 30이상에서는 초당 transaction의 수가 감소하면서 큰 폭의 성능저하를 보여주었다. 그리고 txRate 30까지 OS간에 미세한 수치의 차이를 보이며 상승하지만, 이후의 성능 저하에는 큰 차이를 보였다. BOOYO RC2와 Suse9의 경우에 비교적 낮은 폭의 하강 곡선을 보이지만, RedHat9의 경우에는 최고치를 보인 txRate 30에 비해 txRate 40에서는 이전 수치에서 50%이상 하락한 결과를 보여주며 가장 큰 폭의 성능 저하를 보여주었다.

<그림 5>는 Manufacturing Domain에서의 초당 WorkOrder를 의미하며, 가로축은 txRate(부하)를 세로축은 초당 WorkOder의 수를 나타낸다. 수치는 Manufacturing Domain에서의 성능을 측정하는 지표가 된다. txRate 15까지의 증가분에 대해 WorkOrder가 일정하게 상승하지만, txRate 15 이후에는 BOOYO RC2의 경우, txRate 20을 기점으로 미세하게 증가하지만 그 외 3개의 OS는 급격한 성능 저하를 보였다.

<그림 6>은 SPECjAppServer2004 Metric으로 Dealer Domain의 Transaction Rate와 Manufacturing Domain에서의 Manufacturing Rate 수치의 합으로 Dealer Domain의 초당 트랜잭션 발생 수와 Manufacturing Domain의 주문에 따른 초당 처리 대수를 합한 것을 나타낸다. txRate 20까지는 증가분에 대해 비례하는 결과값을 보여주지만 txRate 15를 기점으로 성능 저하를 볼 수 있으며, txRate 30 이후에는 상승폭이 크게 떨어지는 것을 알 수 있었다. txRate 20 이상에서는 BOOYO RC2가 다른 OS에 비해 높은 수치를 나타내었다. txRate 30 이상에서는 모든 OS의 JOPS가 떨어지는 결과를 보이며, OS간의 차이를 뚜렷하게 파악 할 수 있었다.

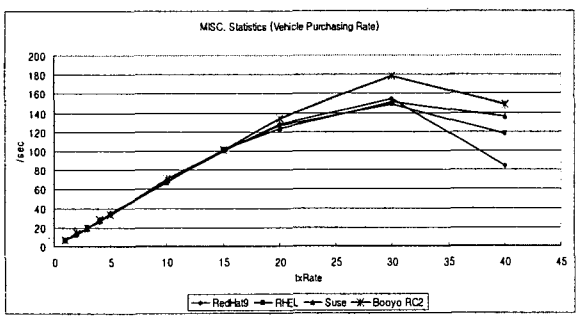

그림 4. Dealer Transaction Rate

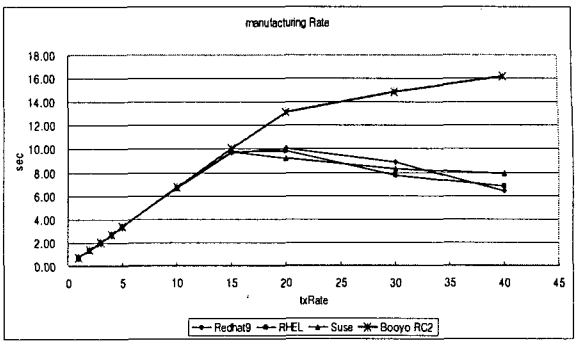

그림 5. Manufacturing Rate

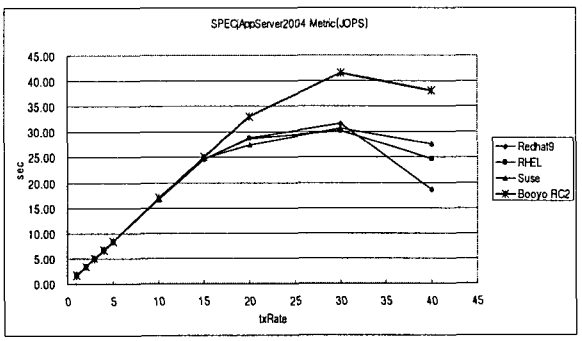

그림 6. SPECjAppServer2004 Metric(JOPS)

5. 결론

SPECjAppserver2004 벤치마크를 이용하여 BOOYO RC2와 외산 제품의 성능 평가 시험을 수행한 결과, BOOYO RC2는 현재 상용화된 리눅스 플랫폼과 비등하거나 경우에 따라서는 비교적 우수한 결과를 보여주었다. 이와 같은 결과는 공개소스 기반의 기업 및 공공서비스 SI 사업에 BOOYO의 적극적인 활용을 위한 촉진제 역할을 할 수 있을 것으로 기대된다.

<참고문헌>

1. 한국소프트웨어진흥원, " 오픈 소스 소프트웨어 연구보고서," 2002. 12.
2. http://www.etri.re.kr/www_05/org/main.htm?pagecode=0201&url=../org/dhrd/team/team_01.html
3. http://jakarta.apache.org/tomcat/index.html
4. http://www.jboss.com/developers/index
5. http://www.postgresql.org/
6. http://httpd.apache.org/
7. http://www.spec.org/jAppServer2004
8. http://www.ipa.go.jp/software/open/forum/north_asia/wg1-e.html
9. http://www.caldera.com