韓國IT서비스學會誌
第5卷 第3號
2006年 12月

벤치마크 테스트를 통한 공개소프트웨어 검증 절차에 관한 연구*

김두연* · 류성열**

A study for Verification Procedures on Open-source Software Via Benchmark Testing*

Doo Yeon Kim* · Sung-Yul Rhew**

■ Abstract ■

Public institutions are considering adoption of open-source software in the process of information projects. However, there doesn't exist reliable information about an adoption process for open-source software. Performance and stability problems of this software also persist, as a result, current open-source software is not widely used. As a software market and industry grows, Benchmark test has been performed more often than before in order to help customers understand and select the most appropriate product among myriad similar ones. It is certain that more objective and trustful data evidence should be obtained by way of utilizing the procedures and methods of Benchmark Test in decision making process for selecting an open-source software.

For this research, Benchmark test was applied as a way of demonstrating performance verification of an open-source software in the public institutions. It is certain that more objective and trustful data evidence should be obtained by way of utilizing the procedures and methods of Benchmark Test in decision making process for selecting an open-source software. It also introduces a case study of a information system, which selected and implemented open-source software, in order to confirm the validity of this research. This research will serve as a guideline to adopt open-source software in governments as well as public institutions.

Keyword : Open Source Software, Linux, Verification, Benchmark test, E-goverment

* 본 연구는 숭실대학교 교내연구비 지원으로 수행되었습니다.
** 교육부 교육행정정보화팀
*** 숭실대학교

1. 서 론

최근 정부·공공기관의 각종 정보시스템 구축에 공개소프트웨어의 사용이 권장되고 있다. 그러나 현재 많은 공공기관의 중요한 시스템 구축 사업에서는 공개소프트웨어 도입의 기준, 절차, 성능 및 안정성 등에 대한 신뢰할 만한 데이터가 부족하여 공개소프트웨어 도입을 꺼리고 있다[1, 2]. 정책적인 고려만으로 공개소프트웨어를 도입하였다가 시스템 구축 후 기술적인 문제가 발생할 경우에 여러 가지 어려움에 직면할 수 있다. 따라서 시스템 구축 후 발생할 수 있는 제반 문제를 예방하기 위해서는 사전에 무엇을 어떻게 검증하는 것이 타당한가 하는 것이 담당자의 고민이다.

한편, 국내 S/W 산업이 점차 성숙해감에 따라 다양한 동종 제품 출시되고 있다. 구매자는 이 중에 최적의 제품을 어떻게 선택할 것인지가 새로운 과제로 등장했다. 점차 일반화되고 있는 S/W 벤치마크 테스트는 구매자가 객관적으로 각 제품의 장단점을 파악할 수 있게 도와줌으로써 구매 목적에 더욱 근접한 제품을 선택하는데 유용한 지침이 된다.

공개소프트웨어의 도입을 위한 의사 결정의 과정에 이러한 벤치마크 테스트의 절차와 기법이 적용된다면 보다 객관적이고 신뢰할 수 있는 데이터를 확보할 수 있다. 이에 본 논문에서는 일반 상용소프트웨어에 비해 공개소프트웨어가 가지는 특성과 위험요소를 고려하여 공공기관의 중요한 시스템에 공개소프트웨어를 도입하기 위한 벤치마크 테스트 절차를 제시하였다. 특히, 공개소프트웨어를 적용한 세계 최대 규모의 시스템인 교육행정정보시스템(NEIS : National Education Information System)의 사례를 중심으로 본 연구의 유효성을 검증하였다.

본 연구는 정부를 비롯한 공공기관에서 중요 시스템에 공개소프트웨어를 도입할 경우, 기술적 검토 절차 및 성능검증 방법에 대한 지침으로 활용할 수 있다.

2. 관련 연구

2.1 소프트웨어 벤치마크 테스트

국내 S/W 산업이 점차 성숙해감에 따라 유사한 제품이 많이 개발되고 이들은 점차 제품군을 형성해 가고 있다. 이러한 상황에서 S/W 벤치마크테스트는 구매자가 객관적인 자료를 통해 구매 목적에 더욱 근접한 제품을 선택하는 유용한 지침이 된다. 반면 개발사 입장에서는 동종 제품들 간의 객관적인 비교평가를 통해 각 제품의 강점을 파악하고, 취약점을 알아내는 등 개선방안을 제시할 수 있으므로 S/W 품질 향상에 큰 도움을 주게 된다[7].

[그림 1] S/W 벤치마크 테스트 프로세스

효율적인 벤치마크테스트 수행을 위해서는 벤치마크테스트 프로세스 정의가 선행되어야 한다. [그림 1]은 일반적인 벤치마크테스트 수행을 위한 프로세스를 도식화한 것이다[6].

요구분석 단계는 시험 대상 항목과 벤치마크테스트 항목 리스트를 도출하기 위해 대상 S/W에 대해 철저한 분석을 한다. 분석 이후 항목 도출 시에는 벤치마크테스트에 참여하는 제품의 업체 개발자 및 담당자들과의 미팅을 통해 도출 항목에 대해 합의하고, 실제 벤치마크테스트 가능 여부를 검증하기 위해 프로토타입 단계를 포함하여 설계단계에 앞서 준비해야 할 조건을 파악한다. 계획단계에서는 요

구분석 단계에서 도출한 결과물들을 토대로, 일정 및 자원 투입 계획을 구체적으로 산정한다. 별도의 교육이 필요한 경우 교육계획을 수립하며 전문가 그룹을 구성한다. 시험환경 구축을 위한 환경설정과 시험도구 및 S/W 선정 등이 계획단계에서 이루어진다. 설계 단계에서는 계획 수립 이후 도메인별 전문가 그룹을 구성하여 시험 대상 주요 항목 및 부가항목을 선정한다. 벤치마크테스트 항목 확정 이후 항목별 측정기준 및 공식을 도출하여 메트릭을 작성하고, 벤치마크 테스트에 참여하는 업체들과 항목에 대한 합의를 진행한 후 벤치마크테스트 평가모델을 완성한다. 실행 단계에서는 개발된 평가모델의 항목별 벤치마크테스트를 수행한다. 비교대상 제품별로 동일한 조건에서 시험을 하고 시험결과가 도출되면 결과에 대한 문제점을 정밀 검토 후, 필요한 경우 보완 시험을 수행하고 결과를 정리한다. 마감 단계에서는 벤치마크테스트의 절차, 방법 및 비교분석 결과를 정리 검토하고 결과에 대한 고객 의견을 조사한다. 최종적으로 벤치마크테스트 과정 및 결과를 저장한다[6].

2.2 공개소프트웨어

공개소프트웨어는(Open Source Software) 1984년 자유소프트웨어(Free Software)운동으로부터 분화하여 현재에 이르렀으며, 소스코드에 접근할 수 있는 권리, 프로그램을 복제하여 배포할 수 있는 권리, 프로그램을 개선할 수 있는 권리를 보장한다. 이러한 공개소프트웨어는 소스코드에 대한 접근성이 보장되므로 시스템 간의 호환성을 확보할 수 있을 뿐 아니라 사용자의 요구에 맞는 일관성과 함께 일치성을 보장받을 수 있다[8, 9, 13].

2002년부터 정부에서 공개소프트웨어 정책추진을 위한 본격적인 사업을 착수하여 현재 그 성과를 보이고 있으나[3~5], 이러한 적극적인 정책지원과 비용 절감에 대한 기대[10, 11]에도 불구하고 현재 중요 업무 시스템(Mission Critical System)에 활발히 도입되지는 못하고 있다. 공개소프트웨어 도입을 주저하는 대부분의 공공부문 기술 관계자들의 의견은 공개소프트웨어 도입에 대한 검증된 데이터가 부족하여 신뢰할 수 없다는 것이다. 또한, 공개소프트웨어에 대한 체계적인 기술지원, 안정적인 유지보수, 각종 응용 소프트웨어가 부족 등도 공개소프트웨어를 도입을 꺼리는 이유 들이다. 그러나 이것은 공개소프트웨어에 대한 오해와 편견에서 비롯된 것이며 이를 객관적으로 평가하면 다음과 같다.

첫째, 공개소프트웨어의 기술지원 문제이다. 공개소프트웨어에 대한 기술지원을 받고 싶으나 지원업체를 찾기가 어렵다고 호소한다. 그러나 대표적인 공개소프트웨어인 리눅스는 배포판 업체뿐만이 아닌 다국적 하드웨어 업체는 물론 국내 소프트웨어 개발 업체들로부터 기술지원을 직접 받을 수 있다.

둘째, 공개소프트웨어의 신뢰도의 문제이다. 독점소프트웨어 업계와 마찬가지로 공개소프트웨어도 매우 다양한 소프트웨어를 사용할 수 있다. 2004년 10월 포레스터 리서치가 미국 140개 대기업을 대상으로 조사한 결과에 따르면 조사 대상 업체의 53%가 자사의 주요 핵심 업무용 소프트웨어 운영환경으로 리눅스를 활용하고 있으며, 새로운 업무용 소프트웨어 도입에 리눅스를 선택할 의사 또한 52%에 달하는 것으로 나타났다[12].

셋째, 성능과 안정성의 문제이다. 리눅스는 유닉스보다 성능이 떨어진다는 인식이 일반적이나, TPC-C[14]나 SPECfp[15] 등에서 제공하는 각종 벤치마크 결과를 보면 리눅스는 유닉스와 윈도우에 비해 높은 성능을 보이고 있다. 특히 국내 사례에서 운영체제 수준의 운용에서 안정성이 뛰어난 것으로 평가받고 있다[5].

3. 공개소프트웨어의 벤치마크 테스트

3.1 공개소프트웨어 도입 위험 요인

공개소프트웨어를 도입하는 방법에는 크게 3가

지 방법이 있다. 첫째, 기관이 직접 선택하여 도입하는 직접선택, 둘째, 민간기업 등 외부의 추천에 의한 간접공급, 셋째, 기관 내부에서 공개소프트웨어를 개발 및 수정하는 내부개발 등이다. 공공기관에서 이러한 3가지 방법의 하나를 선택하여 공개소프트웨어를 도입할 수 있지만 국내 공개소프트웨어 업계와 공공에서의 현실을 고려할 때에 가장 실용적이고, 일반적으로 사용할 수 있는 방법은 간접공급 방식이다[5].

따라서 간접공급에 의한 위험 요인을 분석해 본다. 먼저 하자보증에 대한 위험요인이다. 1차적으로 공급업체는 하자보증에 대해 지원을 해야 한다. 하지만, 공개소프트웨어의 GPL(general public license, 일반 공중 허가서)라이센스가 적용된다면 근원적인 문제점에 대해서는 공급업체에서 보증에 대한 책임을 회피할 수 있다. 따라서 공개소프트웨어를 도입하는 기관 담당자는 보증과 관련한 위험요소에 대한 대비책을 세워두어야 한다.

다음으로, 일반적인 공급계약방식에서 발생할 수 있는 위험요소가 고려되어야 한다. 공급업체의 적격성에 대한 위험성 평가, 시스템의 확장성 가능 여부에 대한 위험요소 평가, 소스코드의 성숙도 및 신뢰도에 대한 위험요소 평가 등이 모두 고려되어야 한다.

또한, 공급업체가 지원하는 공개소프트웨어가 최종 시스템 환경에서 당초 요구한 성능과 안정적 운영을 보장하는지에 대한 위험요소이다. 이는 유사 사례분석과 벤치마킹 테스트 등 충분한 성능검증 절차를 거쳐 위험요인을 최소화해야 한다.

마지막으로 간접방식에서는 외부 공급업체의 기술지원 능력에 대한 위험요소가 존재한다. 즉, 공급업체의 기술지원 능력에 대한 객관적인 검증없이 공개소프트웨어를 도입하였다면 도입 이후의 기술지원 문제가 발생할 수 있다는 점을 명심해야 한다. 아울러 비공개소프트웨어의 경우와 마찬가지로 공급업체의 파산 등으로 기술지원을 받지 못하는 사태가 발생할 수 있으므로 동일 솔루션의 기술지원업체에 대한 정보를 확보해 두는 것도 공개소프트웨어의 장점을 극대화하여 위험 요소를 줄일 수 있는 좋은 방법이다.

3.2 공개소프트웨어 벤치마크 테스트 절차

3.2.1 벤치마크 테스트 환경

시스템 테스트의 목적은 설치 후 시스템이 성공적으로 가동하는 데 있다. 시스템 도입 전 하는 벤치마크 테스트는 최종 시스템 성능에 직접 영향을 미치게 된다. 따라서 테스트 대상업무, 방법 및 항목 등 테스트 환경은 최종 시스템 운영에 매우 큰 영향을 미치게 된다. 본 연구에서는 먼저 테스트 방법을 2개 영역으로 나누고 영역별 테스트 항목을 제시하고, 다음으로 테스트 대상 업무 선정을 위한 기준을 제시하였다.

시스템 테스트 방법은 성능 테스트와 과부하 테스트 등 2가지 영역으로 나누어 실시한다. 성능 테스트는 운영시스템 환경에서 시스템 구성요소별로 부하를 적절히 처리하는지 테스트하는 것이다. 확인하는 세부항목은 처리량, 자원 사용량(CPU, Memory), Disk I/O, 핵심업무 응답시간 등이다. 과부하 테스트는 요구사항의 한계 또는 그 이상에서 시스템 구성요소별로 스트레스를 적절히 분배하는지 테스트한다. 그 세부 점검 항목은 과부하상태에서 응답시간, 시스템 자원 사용량 등이다.

테스트 대상 업무는 시스템 테스트의 유효성을 높일 수 있도록 최대한 다양한 경우의 케이스를 선정한다. 본 연구에서는 다음과 같은 기준을 제시한다.

- 사용빈도가 높은 업무를 선정한다. 평상시 자주 사용되는 화면이 그 예 중 하나이다.
- 특정시간(Peak-time)에 많이 사용되는 업무를 선정한다. 학교는 학기 초에 집중적으로 사용되는 화면이 그 예이다.
- 복잡한 업무를 선정한다. 프로세스가 복잡한 업무, 여러 사이트가 공통으로 사용되는 업무

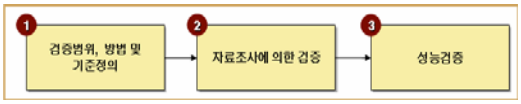
가 된다.

- 많은 양의 데이터를 발생하는 업무를 선정한다. 데이터 처리량이 많은 업무가 그 사례이다.
- 네트워크에 많은 부하를 발생시키는 업무를 선정에 고려한다.
- 온라인 성능에 영향을 줄 수 있는 일괄처리 업무를 선정에 고려한다. 온라인 운영 중 실행되는 배치작업이 그 대상이다.
- 아키텍쳐 구성상 부하 경로를 검증할 수 있는 업무를 선정한다. Web만 부하가 발생하는 화면, Was까지 부하가 발생하는 화면, DB까지 부하가 발생하는 화면 등 각각 선정한다.

위에서 제시한 테스트 환경은 정부나 공공기관의 일반적인 업무프로세스를 고려했다는 점에서 기업 등 특수한 업무프로세스에는 일정한 한계가 있을 수 있다.

3.2.2 공개소프트웨어 검증 절차

3.1에서 분석한 내용과 같은 위험 요인들은 공개 소프트웨어 검증 절차와 기준에 반영된다. 그 기준과 검증 밥법은 각각 [그림 2] 및 <표 1>과 같다. [그림 2]는 공개소프트웨어의 도입에 따르는 위험 요인을 고려한 벤치마크 테스트 절차이다.



[그림 2] 공개 소프트웨어 벤치마크 테스트 절차

공개소프트웨어 벤치마크 테스트 절차는 검증 범위와 방법 및 기준의 정의, 자료조사에 의한 검증, 벤치마크 테스트를 통한 성능검증 단계로 구성된다. 이는 목표 시스템(TO-BE 시스템)의 아키텍처(Architecture) 수립에 방향을 제시하는 핵심 절차이다. 먼저, 검증 범위와 방법 및 기준을 정의 단계는 공개소프트웨어 전문가 참여하에 검증의 범위와 방법 및 기준을 상세히 정의한다.

자료조사에 의한 검증 단계는 공개소프트웨어 도입에 따르는 위험을 최소화하고, 직접 검증에 따르는 비용을 최소화하기 위한 사전 조사 작업으로 시장추세, 기능 및 성능, 지원체계, 적용사례(Reference Site) 등과 같은 관점에서 분석한다. 다음의 <표 1> 은 자료조사를 통한 검증의 기준이다.

〈표 1〉 공개소프트웨어 사전 검증 기준

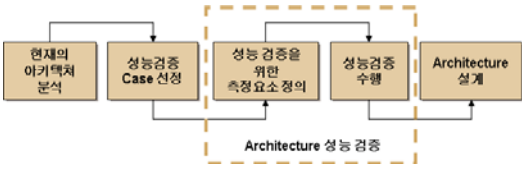
구분	내용
시장추세	• 공개 소프트웨어의 시장동향 및 향후기술 추세에 따른 지속적인 제품기능 강화 여부 • 공개 소프트웨어 주 활용도 분석을 통한 새로운 시스템(TO-BE)에 적용 가능성 여부
기능 및 성능	• 공개 소프트웨어 보유기능 및 안정성 • 성능과 안정성을 지원하는 주요기능 구비 여부(컨넥션 풀링, 클러스터링, 보안성, DBMS 트랜잭션 등) • 오픈 아키텍처 설계를 위한 표준화 지원 여부 • 상용 소프트웨어와의 기능 비교 분석 • 신뢰성 있는 기관의 공개 소프트웨어 성능 검증 결과를 통한 성능 분석
지원체계	• 공개 소프트웨어 공급 업체 현황과 체계적 기술지원 여부 • 공개 소프트웨어의 법적 분쟁에 대비한 고객프로그램 보유 여부 • 사후관리 분석
적용사례	주요기관의 공개 소프트웨어 적용 사례
전문기관 의견	• 정보화정책 및 정보시스템 전문기관 의견 • 공개 소프트웨어 전문기관(한국소프트웨어진흥원 등)의 평가의견 • 정보보호 전문기관 및 기타 관련 기관 의견

다음으로, 성능검증 단계는 기술적 검증의 단계로 벤치마크 테스트를 통한 성능을 검증한다. 이를 위해서 사전에 도입 검토 후보 시스템을 대상으로 핵심 업무 모델링을 통한 기초자료 확보와 테스트 환경 정의, 테스트 데이터와 애플리케이션 프로토타입을 개발하는 등 <표 2>와 같이 성능검증을 위한 기본모델을 수립한다.

〈표 2〉 성능 검증 기본모델

단계	단계별 주요 수행업무	비 고
1	업무 특성 및 처리량 분석	업무 특성에 따른 트랜잭션을 모델링하여 데이터 용량(I/O) 산출
2	서버 용량 산정	서버의 용량산정 (CPU 점유율, 응답시간, 개인식별정보 암호화)
3	서버 성능검증 및 결과 분석	리눅스, 유닉스 등 운영체제에 대한성능 검증
4	종합평가	안정성, 처리 성능, 경제성, 관리 효율성, 보안성 등을 종합평가

기본모델 수립은 작업의 객관성 확보와 체계적 추진을 위하여 <표 2>와 같은 4단계 절차에 따라 작업이 진행되며, 실질적인 성능검증은 다음의 [그림 3]과 같은 절차에 따라 수행한다.



[그림 3] 성능 검증 절차

현재의 아키텍처 분석은 현황분석 및 요구사항 정의 과정에서 도출된 사항을 바탕으로 현재의 IT 환경에서 구현 가능한 케이스(Case)를 개발하는 과정이다. 이는 시스템구현 이미지(Image)를 구성하고, 시스템 구성 명세(Draft)를 정의하는 단계이기도 하다. 다음으로, 성능검증 대상(Case)을 선정하고 측정요소를 정의한다. 성능측정 요소에 대한 정의는 성능검증 시나리오에 반영하며, 이 시나리오에 따라 아키텍처(Architecture) 성능검증을 수행하며, 그 결과에 대한 분석을 토대로 아키텍처를 설계한다. 여기서 제시한 공개소프트웨어 벤치마크 테스트 절차는 공개소프트웨어의 위험요소와 특성을 고려한 결과 자료조사에 의한 검증 단계가 추가되었다. 즉, 이 본문에서 제안하는 벤치마크 테스트 절차는 공개소프트웨어 도입에 가장 적합한 품질평가 기준 중 하나이다. 반면 관련 연구가 많지 않고 적용사례가 부족하여 신뢰성을 입증하기 어려운 한계도 있다.

4. 적용 사례

4.1 적용 대상과 검증 환경

우선 본 연구의 적용사례인 NEIS의 교무업무 시스템은 16개 시·도 교육청 단위로 단독 또는 그룹서버로 운영한다. 특수학교와 고등학교는 1개 학교당 1대의 단독 서버로, 초·중학교는 15개 학교당 1대의 그룹서버로 구성되었다. 목표 시스템의 서버 구성은 단독DB 서버 2,338대, 그룹 DB 서버 606대, WEB/WAS 서버 393대 등 3,300여 대에 달한다. 여기에 학교의 교무/학사, 입(진)학, 보건 업무가 운영되며 인증서를 발급받은 이용자는 교사 등 약 432,000여 명, 학생 약 7백만여 명의 자료가 DB화되었다. 테스트를 위한 단독서버 사용자는 최대 120명, 15개교를 기준의 그룹서버는 최대 이용자 800명을 기준으로 하였다.

성능 검증을 위한 시험시스템은 실제 학교에서 운영하는 시스템과 같게 구성하였다. DB서버는 단독서버 1대, 그룹서버 1대 그리고 WEB/WAS 서버로 구성하였다. 단독서버는 CUP 2개와 메모리 4GB이며 그룹서버는 CPU 4개와 메모리 8GB이며, 시스템 소프트웨어는 운영체제로 리눅스와 유닉스, DBMS, WAS 등으로 구성하였다. 응용소프트웨어는 NEIS 교무/학사 3영역이다.

성능검증은 하드웨어나 소프트웨어의 기본성능을 평가하는 목적이 아니라 교무업무시스템에 필요한 응용소프트웨어를 탑재하여 어느 정도의 성능을 발휘하는가를 검증하는 것이다. 따라서 모든 종류의 유닉스와 리눅스 플랫폼에서 검증을 하지 않고, 요구되는 트랜잭션 처리 성능 만족 여부에 대해 검증을 하는 것으로 그 범위를 제한하였다. 또한, 검증을 위한 애플리케이션과 데이터는 현재의 NEIS 애플리케이션과 샘플 데이터를 활용하였고, 측정 도구는 Mercury사의 Loadrunner

[16]를 사용하였다.

4.2 공개소프트웨어 검증 수행

4.2.1 성능검증 범위

공개 소프트웨어의 검증은 본 연구에서 제시한 절차를 적용하여 기술검증을 통한 적용 가능성을 검토하고, 성능 검증 및 최적화 작업을 수행하였다. 우선, 검증 범위의 대상 공개소프트웨어는 운영체제로는 리눅스를, 시스템소프트웨어로는 Apache, MySQL 등을 선정하였다.

〈표 3〉 검토 대상 공개소프트웨어

구 분	검증대상 공개소프트웨어
운영체제	리눅스
시스템 소프트웨어	• 웹서버용(Apache) • 웹어플리케이션용(Tomcat) • DBMS용 (MySQL)

1차적으로 자료 조사를 통해 검토한 결과 리눅스 운영체제는 시장동향, 기능 및 성능분석, 지원체계 등에서 활용 가능하다고 판단되어 성능 검증 대상에 포함하였으나, 시스템 소프트웨어(Apache, Tomcat, MySQL)는 요구기능 미흡 및 시스템 컴포넌트 간의 연계성이 미흡하여 교무업무시스템 구축을 위한 공개소프트웨어 활용 대상에서 제외하였다. 다음으로, NEIS 교무업무시스템 구축과 관련된 아키텍처 설계를 위하여 리눅스 및 유닉스 시스템을 대상으로 핵심 업무 중심의 서버 성능검증(서버용량 산정을 위한 기초자료 확보, 리눅스 타당성 검증)을 실시하였다.

4.2.2 성능검증 항목 및 시나리오

기존 NEIS의 교무/학사 업무 중 가장 시스템 부하가 많은 학교생활기록부 업무(조회, 저장, 마감)와 월 출결 업무(조회, 저장, 마감)를 선택하였다. 성능검증은 사용자 수 증가에 따른 시스템 응답시간과 CPU 사용률을 측정하고, 개인 식별 자료 중심으로 최소한의 항목(주민번호, 이름, 석차, 번호)을 암호화하여 암호화에 따른 성능 차이를 검증하였다. 필요 이상의 모든 항목을 암호화할 경우 시스템에 많은 부담이 되기 때문에 누구의 자료인지 식별할 수 없도록 학생 개인의 식별정보를 DB 저장 시에 암호화함으로써 학생 자료에 대한 신뢰성과 무결성, 보안성 등을 확보할 수 있다.

〈표 4〉 성능검증 시나리오

단독서버 (Web/Was/DB)	그룹서버		암호화 적용
	물리적 2 Tier (Web/Was-DB)	물리적 3 Tier (Web-Was-DB)	
LX UX	UX - UX LX - LX LX - UX	UX - UX - UX LX - LX - LX LX - UX - UX	UX - UX LX - LX LX - UX

UX : UNIX, LX : LINUX

<표 4>의 성능검증 시나리오는 단독서버, 그룹서버 그리고 암호화 적용 여부로 크게 구분하고, 그룹서버는 2 Tier와 3 Tier 구조로 나누어 구성한다. 이 4가지 경우별 시나리오는 UNIX와 LINUX의 경우의 수 조합이다.

4.2.3 성능검증 기준, 내용 및 결과

교무업무시스템 시범사업을 위해 시험 시스템을 구축한 후 진행한 시험운영 환경, 성능 기준, 시험내용 및 시험결과를 정리하면 <표 5>와 같다.

리눅스를 기반으로 한 시범학교의 NEIS 교무업무시스템에 보안 시스템을 탑재하고, 학생 개인 식별정보를 암호화 후 적용한 시험운영에서 업무처리 속도 및 자원 사용율 기준을 모두 통과함에 따라 교무업무시스템의 전국단위 물적기반 구축 사업에서 단독DB 서버 2,300여 대에 최종적으로 리눅스가 채택되었다. 교무업무시스템의 단독DB 서버에 리눅스 도입 이 후에도 성능검증 및 최적화 작업(2회)을 통하여 시스템 개통 이전에 최상의 성능을 발휘 할 수 있도록 최적화 작업을 수행하였다.

〈표 5〉 공개 S/W 성능검증 요약

구분	기준
검증 환경	- 실제 운영환경과 동일하게 구성하여 성능시험 실시 - 대상 : 단독 및 그룹서버, WEB/WAS 서버 - H/W : 단독서버(CPU 2개, 메모리 4GB), 그룹서버(CPU 4개, 메모리 8GB) - 주요 시스템S/W : O/S(단독DB서버 : 리눅스, 그룹 DB서버 : 유닉스), DBMS (UniSql), WAS (JEUS) 등 - 응용 S/W : 실제 새롭게 개발 완료된 응용S/W 중 최대 부하를 유발하는 학교생활기록부 조회, 반영 부분을 시험 대상항목으로 선정
성능 기준	- 1개 고등학교의 최대 사용자 수 : 120명(동시 사용자 수 14명) - 15개 초·중학교의 최대 사용자 수 : 800명(동시 사용자 수 96명) - 응답시간 : 평균 3초 이내(배치성 업무 제외) - 자원(CPU, 메모리) 사용율 : 75% 이하
시험 내용	- 학교생활기록부 관련 개발 응용S/W 적용 시험 - 동시 사용자 수(단독서버 : 14명, 그룹서버 : 96명)를 50%, 100%, 200%로 조정하면서 학생목록 및 해당 학생의 학교생활기록부 조회, 반영 등 시험
시험 결과	- 시험결과 응답시간, 자원 사용률 모두 성능기준 통과 - 업무처리 속도 : 응답시간은 평균 3초 이내로 기준 통과 - 자원 사용율 : CPU 및 메모리 사용율 모두 70% 이내로 기준 통과 - 시스템S/W : DBMS, Web/Was 등 모두 성능 및 안정성 통과

5. 평가 및 결론

본 연구에서는 공공기관의 중요 시스템에 공개소프트웨어를 도입하기 위한 검증 절차를 검증범위와 방법 및 기준의 정의, 자료조사에 의한 검증, 성능검증 등 3단계로 제안하였다. 또한, 각 단계별 활동에 대하여 구체적인 내역을 제시함으로써 공개소프트웨어 검증 절차에 직접 적용이 가능하게 되었다. 특히, 공개소프트웨어 검증 절차의 핵심 활동인 성능검증은 5개 세부 활동으로 나누어 제안하였다. 공개소프트웨어 제반 위험요소를 일반 벤치마크 테스트절차에 반영하여 자료조사에 의한 검증 단계가 추가되었다. 본 논문의 벤치마크 테스트에 의한 성능검증은 구매자가 제품 선정에 더욱 객관적이고 신뢰할 수 있는 데이터를 확보할 수 있다 데 장점이 있다.

또한, 벤치마크 테스트에 의한 성능검증은 교육행정정보시스템(NEIS)의 교무업무시스템 구축에 공개소프트웨어 도입 사례를 중심으로 본 연구의 유효성을 검증하였다. NEIS 교무업무시스템에 공개소프트웨어 적용을 위한 성능검증 절차는 성능검증 환경 구축, 성능검증 기준 및 시나리오 작성, 성능 시험 및 결과 측정 등의 순으로 진행하였다. 이렇게 공개소프트웨어 도입 이전에 사전 성능검증 등 충분한 검토 절차를 거쳐 교무업무시스템의 시범사업 중 단독서버에 공개소프트웨어인 리눅스가 도입되었다. 시범사업은 2개 시도교육청 관내 28개 시범학교 및 108개 학교를 대상으로 진행되었다. 시범사업 결과도 벤치마킹 테스트에 의한 성능 기준이 충족됨에 따라 본 사업의 단독서버 2,300여 대에 리눅스가 최종 적용되었다.

NEIS 사례와 같이 시스템에 공개소프트웨어가 도입되고 운영하기까지의 과정은 검증범위 및 기준 정의, 자료조사에 의한 간접 검증, 벤치마크 테스트에 의한 성능검증 등 다단계에 걸쳐 진행되었다. 그 결과 시스템이 정상 가동되고 있다. 이는 공공기관의 중요 업무시스템 구축 시 공개소프트웨어를 도입하기 위한 가장 적절한 업무 적용 모델이라 평가된다. 향후 전자정부 사업을 비롯한 중요사업에서도 본 논문에서 제시한 벤치마크 테스트에 의한 성능검증 등 검증 프레임워크를 활용한다면, 성공적인 공개소프트웨어 적용 시스템 구축과 함께 비용절감, 신뢰성 확보 등의 성과도 얻을 수 있다.

그러나 본 연구에서 제시한 검증 프레임워크는 대규모 사업에 적용한 사례이므로 향후 다양한 시스템 규모에 적용해 보고 이에 따른 보완과 적용업무, 시스템 규모, 사업 규모 등에 따른 세분화된 모델 연구가 필요하다.

참 고 문 헌

[1] 송위진, “오픈소스 소프트웨어의 기술혁신 특성 : 리뷰”, 한국기술혁신학회지, 제5권, 제

2호(2002), pp.212-227.
[2] 한국소프트웨어진흥원, '공개소프트웨어 도입 가이드라인 연구', 2003. 12.
[3] 한국소프트웨어진흥원, 오픈소스소프트웨어 연구보고서, 2002.
[4] 한국소프트웨어진흥원, 공개소프트웨어 도입 성공사례집, 2005.
[5] 정보통신부, 공개소프트웨어 가이드, 2006. 1.
[6] 한국소프트웨어진흥원, S/W BMT 기술동향 및 활성화 방안 연구보고서, 2003. 12.
[7] 한국정보통신기술협회, 소프트웨어 품질 벤치마킹을 위한 평가기술에 관한 연구, TTA, 2001.
[8] Research, FLOSS(Free/Libre and Open Source Software : Survey and Study), FINAL REPORT, 2002.
[9] MICHAEL J. KARELS, "Commercializing Open Source Software", ACM Queue, Vol.1, no.5(2003), pp.46-55
[10] Al Gillen, Dan Kusnetzky, and Scott McLarnon, The Role of Linux in Reducing the Cost of Enterprise Computing, IDC White Paper, 2002.
[11] Robert Frances Group, Total Cost of Ownership for Linux in the Enterprise, 2002.
[12] Brad Day, Laura Koretzle, Linux Crosses Into Mission-Critical Apps, Gartner, 2004.
[13] OSI, The Open Source Definition, http://www.opensource.org/definition.php, 2006. 4.
[14] Transaction Processing Performance Council, TPC, http://www.tpc.org/tpcc/, 2006. 2.
[15] SPEC, Standard Performance Evaluation Corporation, http://www.spec.org/results.html, 2006. 4.
[16] Mercury, "Mercury Performance Center", http://www.mercury.com/us/products/performance-center/, 2006. 4.

◈ 저 자 소 개 ◈

김 두 연 (kimdoo@moe.go.kr)

현재 교육인적자원부 교육행정정보화팀장으로 재직 중이며, 서울산업대학교 컴퓨터학과를 졸업하고, 숭실대학교 산업대학원에서 석사를 취득, 현재 박사과정에 재학중이다. 주요 관심분야는 소프트웨어 재사용, 오픈 소스, 정보기술아키텍쳐, 유비쿼터스 컴퓨팅, 정보보호 등이다.

류 성 열(syrhew@comp.ssu.ac.kr)

현재 숭실대학교 컴퓨터학부 교수로 재직중이며, 아주대학교 컴퓨터학부에서 박사를 취득하였다. George Mason University에서 교환교수를 지낸바 있으며, 주요 관심연구분야는 소프트웨어 유지보수/재사용, 소프트웨어 재공학/역공학, 정보보호 등이다.