JMS 메시지 송수신 시간의 최소화를 위한 대용량 메시지 송수신 플랫폼 구현

[An Implementation of Large Scale JMS(Java Message System) for Transmission Time Minimization]

조 풍 연 † 박 제 원 †† 최 재 현 † 이 남 용 †††
(Poongyoun Cho) (Jaewon Park) (Jaehyun Choi) (Namyong Lee)

요 약 동기/비동기 방식을 통한 메시지 송수신은 근래에 들어 많은 분야의 기업 환경에서 실시간 메시지 전송을 위해 널리 사용되고 있는 실정이다. 특히 JMS(Java Message Service) 메시지 플랫폼은 가장 많이 활용되고 있는 메시지 송수신 미들웨어로서 내부 및 외부의 정보시스템 통합을 위한 기반기술로 각광받고 있다. JMS는 실시간/배치 작업 지원, 안정성 보장 및 분산 환경 지원 등 매우 효율적인 기능을 지원하는 반면에, JMS가 갖는 신뢰성 보장 기능과 메시지 단위의 송수신 특징은 인터넷과 같은 분산된 환경에서 논리적으로 매우 거리가 먼 두 시스템간의 대용량 메시지 송수신을 위해 활용할 경우에는 일반 단위 트랜잭션 처리와는 다른 고려요소가 필요하다. 특히 송수신할 단위 메시지 크기는 총 메시지 송수신 시간을 좌우하는 매우 중요한 요소가 된다. 따라서 본 논문에서는 JMS 메시지 플랫폼의 환경에 따른 최적화된 단위 메시지 크기를 산출하는 기법과 이를 실현할 수 있는 시스템을 제안하고자 한다. 이는 JMS 응용 시스템으로서 대용량 메시지 전송을 위한 JMS 기반의 메시지 송수신 시스템 개발 시 최적화된 단위 메시지 크기를 산출하여 적용함으로써 총 메시지 송수신시간을 최소화 할 수 있도록 한다. 마지막으로 이를 실제 환경에서 테스트하고 기존의 JMS 처리 방식과 비교 평가함으로써 본 논문에서 제안하는 기법과 구현 시스템에 대한 검증을 수행하였다.

키워드 : 자바 메시지 서비스, 대용량 메시지 송수신

Abstract Recently, message based data transmission plays an important role in modern computing systems. Especially JMS(Java Message Service) is one of the most popular messaging platform. However, because of its characteristics for maintaining reliability, if we want to use it for transmission of large scale messages in a distributed Internet environment by using a WAN connection which may not be robust enough, we need to employ a different method to minimize total transmission time of messages. We found the fact that the total time of message transmission heavily depends on size of a message. In order to achieve the ideal size of a message, we develope a novel mechanism and a system which finds the ideal size of a message and automatically control JMS applications for minimizing transmission time. Finally, we test the proposed mechanism and system using real-data in order to prove advantages and compared with the naive mechanism. As a conclusion, we showed that our proposed mechanism and system provide an effective way to reduce transmission time of large scale messages in distributed environment.

Key words : JMS(Java Message Service), Transmission of Large Scale Messages

· 본 연구는 숭실대학교 교내연구비 지원으로 이루어졌음

† 정 회 원 : 숭실대학교 컴퓨터학과
cpy@metabuild.co.kr
uniker80@lycos.co.kr
†† 학생회원 : 숭실대학교 컴퓨터학과
kkkjw22@hotmail.com
††† 정 회 원 : 숭실대학교 컴퓨터학과 교수
nylee@ssu.ac.kr

논문접수 : 2008년 1월 22일
심사완료 : 2008년 11월 11일

정보과학회논문지: 컴퓨팅의 실제 및 레터 제15권 제1호(2009.1)

1. 서 론

근래에 들어 동기/비동기 방식을 통한 메시지 송수신은 많은 분야의 기업 환경에서 실시간 메시지 전송을 위해 널리 사용되고 있다. 특히 JMS(Java Message Service) 메시지 플랫폼은 가장 많이 활용되고 있는 메시지 송수신 미들웨어로서 내부 및 외부의 정보시스템 통합을 위한 기반기술로 각광받고 있다. JMS는 실시간/배치 작업 뿐 아니라 안정성이 보장되는 분산 환경을 제공하여 오늘날 컴퓨팅 환경에서 핵심적인 기능을 제공하고 있다. 예를 들어, 보내고자 하는 데이타를 메시지 단위로 포장하고 송수신에 필요한 연결관리, 트랜잭션관리 등을 신뢰성 있게 처리하고 표준화된 API를 제공하여 시스템의 개발 및 유지보수의 시간과 비용을 획기적으로 줄여준다.

JMS 메시지 플랫폼을 활용하는 방식은 매우 다양한데 크게 전송 데이타의 크기와 메시지가 갖는 의미에 따라 구분하면 단일 트랜잭션 메시지 처리와 대용량 메시지 송수신으로 구분될 수 있다. 전자는 단위 업무처리를 위한 메시지를 처리하는 목적으로 한 건의 메시지가 스스로의 의미를 가지며 크기가 작고 비교적 빈번하게 발생하는 특징이 있다[1,2]. 후자의 경우는 JMS를 활용하여 데이타웨어하우스 구축과 같은 목적으로 대용량 데이타 및 멀티미디어 데이타의 송수신 등에 활용하는 방식으로 단일 메시지는 스스로의 의미가 없고 여러 메시지가 순서에 맞게 위치해야 의미가 존재하며, 메시지의 크기가 크고 비교적 특정 시간에 일시적으로 발생하는 특징을 가진다. 근래에 들어 조직 내부 및 외부 시스템이 유기적으로 통합되는 요구에 따라 조직 외부의 시스템 간의 일괄 처리를 수행하는 필요성이 증대되고 있는 실정이다[3,4].

후자의 경우 대부분 원격지에 존재하는 데이타 소스에서 데이타를 취합 및 통합하기 위해 JMS를 사용하는데, 물리적으로 또는 논리적으로 떨어진 두 시스템 간에 대용량 데이타를 안정적으로 전송해야 하는 과제를 가지고 있다. 이 경우에 원격지에서 대용량 메시지 송수신 시에 발생되는 네트워크 에러로 인해 메시지 송수신 시간이 길어지게 되어 전체 시스템의 성능을 저하시킬 수 있다. 안정적인 네트워크가 제공되는 내부 시스템간의 메시지 송수신시에는 전송 오류가 무시할만한 수준이지만, 분산 환경에서는 네트워크의 상태나 조건에 따라 전송 에러가 발생할 수 있는 가능성이 매우 높다. 이러한 에러 발생은 단일 트랜잭션 메시지 처리와는 달리 대용량 메시지 송수신 방식에서는 단일 메시지의 크기가 매우 커질 우려가 있기 때문에 대용량 메시지를 전송할 때 단일 메시지의 크기 결정에 중요한 영향을 미치게 된다[5-7].

만일 단일 메시지의 크기가 매우 커질 경우 신뢰성 있는 메시지 송수신을 위해 메시지 재전송을 고려한다면 메시지 재전송이 없는 이상적인 경우에 비해 매우 많은 시간이 소요될 수 있다. 왜냐하면 JMS 메시지 플랫폼에서는 메시지 전송 중 에러가 발생할 경우 전송중인 메시지 전체를 롤백하고 해당 메시지를 처음부터 새롭게 보내는 방식을 사용하기 때문이다[8]. 따라서 대용량 메시지 송수신 시에 메시지 전송 오류가 발생하게 된다면 오류 손실시간(오류 발생 전까지의 메시지 전송 시간과 오류가 발생함으로써 필요한 처리시간의 합)이 추가로 소요된다.

또한 반대로 단위 메시지 크기를 너무 작게 설정하는 경우에는 매번 메시지를 송수신하는데 기본적으로 필요한 JMS 처리시간(JMS 연결 시간과 큐 처리시간 및 피드백 전송시간의 합)이 증가하여 오히려 총 메시지 송수신 시간을 증가시킨다. 따라서 적당한 단위 메시지 크기를 계산하고 시스템에 반영하는 것이 총 데이타 송수신시간에 중요한 변수로 영향을 미치게 되는데, 대부분의 JMS 기반의 시스템 구축 시에는 이러한 문제점을 간과한 채 대략적으로 적당한 단위 메시지 송수신 크기를 결정하여 적용하고 있는 실정이다. 이로 인해 불필요한 JMS 처리시간이 매 메시지마다 반복적으로 소요되거나 잦은 롤백으로 오류 손실시간이 급격히 증가하는 문제가 있다.

이러한 문제를 해결하기 위해 본 논문에서는 JMS 시스템의 운영환경을 분석하고 필요한 파라미터를 도출한 후 해당 환경에 최적화된 단위 메시지 크기를 계산한 후 이를 실제 시스템에 적용하여 성과를 비교 평가하고자 한다. 이를 통해 기존의 비효율적인 부분을 제거함으로써 분산 환경에서의 대용량 메시지 송수신 시 전송시간을 상당히 줄일 수 있을 것으로 기대된다.

본 논문의 구성은 2절에서는 본 연구의 배경이 되는 JMS에서의 메시지 송수신 방법을 살펴봄으로써 JMS가 갖는 특징이 대용량 메시지 송수신 환경에 어떠한 영향을 주는지 살펴보고 기존의 유사한 관련연구와 한계점에 대해 알아본다. 3절에서는 본 연구에서 제안하고자 하는 최적화된 단위 메시지 크기를 유도하기 위한 메시지 크기와 전송시간의 관계를 분석하고 이에 영향을 주는 파라미터에 대해 분석한다. 다음으로 4절에서는 구현 시스템을 이용하여 실제 데이타를 기반으로 평가한 실험 결과를 보이고 분석하고, 5절에서는 본 연구의 중요성 및 향후 과제와 함께 결론을 내린다.

2. 연구 배경

2.1 JMS에서의 메시지 송수신 방법

JMS는 오늘날 가장 많이 사용되는 메시지 송수신 플

랫폼으로서 실시간/비실시간 및 동기/비동기 통신을 모두 지원한다. JMS 시스템의 구성요소는 JMS 프로바이더(JMS Provider), JMS 클라이언트(JMS Client), 메시지(JMS Message), 그리고 관리객체(JMS Administrative Tool) 등으로 구성된다.

그림 1은 JMS에서의 메시지 송수신 모델을 나타낸다. CF(Connection Factory)와 D(Destination)를 JMS 객체(JMS Objects)라 부르며, 관리객체는 JMS 클라이언트가 사용하기 위하여 사전에 정의된 JMS 객체를 의미한다[9]. JMS 프로바이더와 JMS 클라이언트는 JMS에서 제공하는 API를 통해 다양한 형태의 메시지를 표준화된 인터페이스를 이용하여 송수신할 수 있다. 또한 분산 환경에서 원격의 메시지를 송수신할 수 있을 뿐만 아니라 메시지 전송 보장기능이 지원되어 매우 신뢰할 수 있는 수준의 메시지 처리가 가능하다.

송수신 시스템은 전송할 데이타를 메시지 형태의 객체에 담아 JMS 프로바이더를 통해 정해진 메시지 큐(queue) 또는 메시지 토픽(topic)에 데이타를 담게 된다. 메시지 큐는 JMS 프로바이더가 제공하는 메시지의 중간 저장소 역할을 하며 송신 시스템에서 보낸 메시지를 일정시간동안 임시 저장하며 수신 시스템에서의 요청이 있을 경우 조건에 따라 메시지 큐에서 해당 메시지를 수신 시스템으로 전송하고 메시지 큐에서는 삭제한다. 메시지 토픽은 메시지 큐와 유사한 역할을 하지만 동일한 메시지를 다수의 구독자가 수신할 경우 활용한다.

JMS에서 제공하는 메시지 전송 보장기능은 일련의 JMS 송수신 트랜잭션 중 발생한 오류를 감지하여 오류가 발생한 메시지를 롤백 시킴으로써 전송을 취소하고

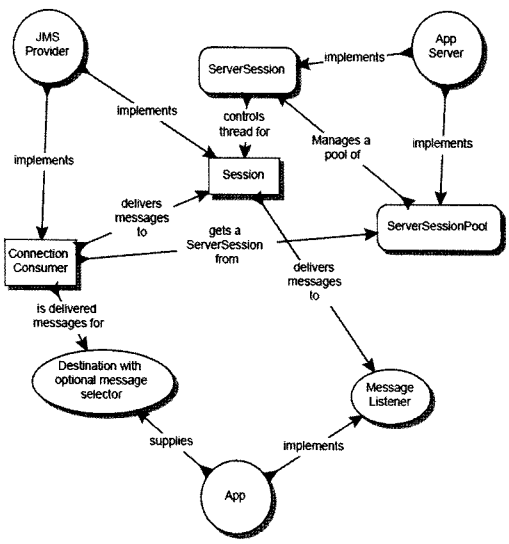

그림 1 JMS에서의 메시지 송수신 모델

해당 메시지를 자동으로 재전송하도록 하고 있다. 따라서 메시지 전송 시에 오류가 발생한 경우에 송수신 시스템은 이에 영향을 받지 않게 되며 메시지 생성 및 처리를 위한 비즈니스 로직에만 집중할 수 있게 된다.

이러한 JMS의 특징은 일반적으로 분산 환경에서의 메시지 송수신 시에 많은 장점을 제공하지만, 인터넷과 같이 완전히 신뢰할 수 없는 네트워크 환경에서 대용량 데이타를 송수신할 경우에는 다음과 같은 문제점이 발생할 수 있다.

우선 단위 전송 용량이 커지면 롤백 시 재전송 부담이 크다는 점이 있다. JMS는 메시지 송신/수신 시 네트워크 오류 및 데이타 처리 오류 및 비즈니스 로직 수행 시 발생할 수 있는 오류 등으로 인해 메시지 롤백이 발생하게 되면 해당 메시지를 처음부터 다시 전송해야 하는 구조이다. 따라서 송수신해야 할 메시지의 크기가 커지면 커질수록 오류 발생 시 재전송 시간이 길어지게 되고 대용량 데이타의 재처리 작업이 빈번해 질 확률이 높으므로 시스템의 성능 저하를 유발하게 된다. 그러므로 대용량 메시지 통신 시 빈번한 데이타웨어하우스, 멀티미디어시스템 또는 정보시스템에서 자주 사용되는 일괄처리(batch) 작업에서 문제가 발생할 소지가 크다. 그리고 단위 전송 용량이 작아질 경우 개별 메시지의 전송/재전송 속도는 빠르지만 크기가 작은 메시지가 다량 전송 될 경우 전체 전송 시간이 매우 느려지는 문제점이 있다.

이는 JMS API와의 통신을 위한 비용이 트랜잭션이 발생할 때마다 일어나기 때문인데 이때 전체 전송 시간은 하나의 메시지를 전송할 때 소요되는 시간과 메시지 큐와의 연결 시간, 전송/수신 시 피드백에 소요되는 시간의 합에 전체 메시지의 개수를 곱한 만큼의 시간이 소요되므로 금융권이나 전자 상거래 등 소용량 메시지의 다량 전송이 빠르게 이루어져야 하는 시스템 에서는 사용되기가 힘들다. 그러므로 위와 같은 JMS의 문제점을 해결하기 위하여 대용량 메시지 송수신의 최적화 기법이 필요하다.

2.2 관련연구

본 논문에서 다루고자 하는 연구주제에 대해 구체적으로 언급하기 전에 메시지 전송 성능 최적화를 목표로 한 기존의 관련 연구들을 분석해 보겠다.

인터넷 환경에서 신뢰성 있는 대용량 메시지 송수신을 위해 SUN은 메시지 타입과 응답 방법 조절을 통한 전송 속도 최적화 방안[10,11]을 제안하였고, M. Menth 외 3인은 대용량 메시지 송수신 환경에서 메시지 필터링을 통해 송수신의 효율성을 높이는 방안에 대해 연구하였으며[12], 마지막으로 Gorton, I외 4인은 컨텐트 기반의 메시징 방식을 통해 분산 환경에서 대용량 메시지

의 라우팅, 포맷팅 및 필터링 연구[13]를 수행하였다.

우선 SUN에서 권장하는 메시지 전송 최적화를 위한 방안은 바이트 메시지를 통한 메시지 전달방법과 메시징 시 응답(acknowledge)방법 변경을 통해 속도를 향상시키는 두 가지 방안이다.

첫째로, 바이트 메시지를 사용하여 다른 메시지 형태와 다르게 직렬화에 요구되는 시간을 단축시켜 송수신 시간을 단축하는 방법이 있다. 다음으로, 응답방법 변경을 통한 전송 시간 단축은 수신자로부터 메시지 수신이 확인 될 때 까지 메시지의 세션을 유지시키지 않는 방법이다. 이 경우 수신자와 프로파이더 간의 상호작용에 소비되는 시간이 줄어들기 때문에 전송 시간 자체는 줄어들 수 있으나, 프로바이더의 메시지 관리 정보가 불분명하게 되므로 메시지의 중복 수신가능성이 발생한다.

이상의 두 가지 방안은 총 메시지 송수신 시간을 단축시킬 수는 있지만 각각 한계점이 있다. 우선 첫 번째 방안의 경우, 바이트 메시지를 사용하게 되면 동일한 크기의 메시지를 전달할 때에는 텍스트나 객체 메시지에 비해 상대적으로 빠른 전송이 가능하지만, 메시지 자체의 크기가 커지게 될 경우 최적화된 전송 속도를 보장하는 것은 여전히 해결할 수 없다. 또한 두 번째 방안의 경우 역시 전송되는 메시지의 크기가 커질 경우에는 최적화된 전송 속도를 보장하기 어려우며 특히 메시지 전달 순서와 전송 횟수가 중요한 문제가 되는 시스템에서는 사용할 수 없다는 치명적인 문제를 가지고 있다.

또한 M. Menth나 Gorton, I의 연구는 주제기반 메시징 시스템을 이용하여 메시지의 라우팅, 포맷팅 및 필터링을 통해 분산 환경에서 대용량 메시지 전송 시에 송수신 시스템간의 메시지 전송 효율을 극대화 할 뿐 아니라 다양한 메시지의 동적인 운영을 가능하게 하였다. 특히 Gorton, I의 연구는 분산 환경에서 대용량의 메시지를 많은 수의 publisher와 subscriber가 동시에 접근할 경우 발생하는 부하 문제를 메시지 필터링과 라우팅 기법을 활용하여 효과적으로 해결하였다. 그러나 연구의 주제가 다수의 소용량 메시지 송수신에 국한되어 있어 단일 메시지의 크기가 일정 수준을 넘어설 때 필요한 메시지 트랜잭션 처리 및 분할 송수신 환경에 적용하기 어렵다는 단점이 있다. 그러므로 메시지의 종류나 크기에 구애 받지 않고, 신뢰성을 보장 할 수 있으면서 최적화된 전송 속도를 얻을 수 있는 새로운 방안이 필요하다.

3. 송수신 시간 최소화를 위한 방법연구

JMS 메시지 플랫폼에서 단위 메시지 크기와 총 전송 시간의 관계를 분석하기에 앞서 분산 환경에서의 JMS 활용 시 고려되어야 할 두 가지 사항에 대해서 언급하도록 하겠다. 첫째는 JMS 메시지 플랫폼을 위해 한건의 메시지를 보낼 때마다 반복적으로 필요로 하는 JMS 처리시간이며, 둘째는 분산 환경에서 네트워크 상태, I/O 오류, 메시지 프로세싱 오류 등으로 발생할 수 있는 메시지 전송오류이다.

JMS 처리시간은 JMS 메시지 플랫폼에서는 메시지를 송수신하기 위해서 기본적으로 JMS 연결과 JMS 세션을 생성하는 단계, 즉 JMS 서버에 연결을 설정하기 위한 단계에 소비되는 시간을 의미한다. 이는 데이타베이스에서 질의를 수행하기 위해서는 데이타베이스 연결을 맺어야 하는 것과 동일하다고 할 수 있다. 따라서 매번 메시지 송수신 시에 반복적으로 JMS 연결 시간이 필요하게 되기 때문에, 일반적으로 이를 줄이기 위해 단일 연결로 최대한 많은 메시지를 보내는 것이 속도 향상에 매우 유리하게 작용하게 된다.

메시지 송수신 오류는 물리적인 오류와 논리적인 오류로 구분할 수 있는데, 전자는 네트워크의 오류 및 I/O 처리 오류 등의 시스템 이상동작으로 발생하는 오류를 의미하고, 후자는 메시지 송수신 전후에 발생할 수 있는 시스템 처리상의 의미적인 오류를 의미한다. 논리적인 오류는 송수신 메시지의 코드 값의 불일치 등의 무결성 문제가 대표적이라 할 수 있다.

위와 같은 사항들을 고려하여 단위 메시지 크기와 총 전송시간의 관계를 분석하기 위해서 가장 중요한 변수인 메시지 전송오류에 대한 요소를 구분하여 파악하기 위해서 본 절에서는 우선 매우 신뢰할 수 있는(에러가 발생하지 않는) 네트워크 환경과 응용시스템을 가정하고 단위 메시지 크기와 전송 시간의 관계를 파악한 후, 에러가 발생할 수 있는 경우를 고려하여 이를 발전시킴으로써 원하는 최종 관계를 이끌어내도록 한다.

보내고자 하는 전체 정보를 M이라 할 때 전체 정보는 일정한 메시지 집합으로 구성되며 각 메시지는 m으로 표현한다. 예를 들어, m_i는 i번째에 해당하는 메시지이다. 각 메시지는 개별적으로 특정 의미를 가질 수 도 있으며 여러 메시지가 병합되어야만 의미를 가질 수 있다. 하지만 대부분의 대용량 메시지 처리 시에는 전체 메시지 M이 모두 병행되어야만 의미를 갖는 경우가 많다. 따라서 s를 단위 메시지 크기라 하고 n개 메시지를 보낼 경우 총 $s \times n$ 만큼의 데이타가 전송되어야 한다.

또한 r을 운영되는 네트워크의 평균 송수신 속도라고 하고, 평균 JMS 연결 시간과 평균 JMS 큐 처리시간 및 평균 JMS 피드백 전송 송수신 시간의합을 *JPT* (*JMS Processing Time*)로 정의하기로 한다. 이는 JMS 시스템에 종속 변수로, 시스템 상에서는 상수로 처리될 수 있다.

본 연구에서 최소화하고자 하는 총 메시지 전송시간을 T로 표현하고 단위 메시지의 전송시간을 t로 표현

한다면 단위 메시지의 크기(s)와 전송시간의 관계를 $t_i = \frac{s}{r} + JPT$ 로 정의할 수 있다. 이는 단위 메시지 m_i의 전송을 위해 필요한 시간으로서 $\frac{s}{r}$는 실제 메시지 내용을 전송할 때 필요한 시간이며, 개별 메시지 전송 시 마다 JPT 만큼의 시간이 반복적으로 필요함을 알 수 있다.

$$T = t_1 + t_2 \ldots + t_n = nJPT + \frac{ns}{r} \quad (1)$$

식 (1)은 송신하고자 하는 전체 메시지 집합(M)을 전송하는데 필요한 총 메시지 전송 시간(T)을 의미한다. 이때 여기에서 JPT 및 r은 상수이고, n는 M이 가지는 개별 메시지의 개수를 의미하므로, 개별 메시지의 개수가 늘어날수록 총 전송시간은 증가한다. 또한 개별 메시지의 크기(s)가 줄어들수록 전체 메시지 전송시간은 감소하지만, 반대로 메시지의 개수가 증가하여야만 하기 때문에 이 두 변수간의 적절한 관계를 찾는 것이 중요하다고 할 수 있다.

따라서 에러가 발생하지 않는 매우 신뢰할 수 있는 네트워크상에서의 대용량 메시지 전송 시에는 단위 메시지 크기(s)가 늘어날수록 전체 메시지 전송시간이 감소하다가 단위 메시지 크기(s)가 ns과 동일해질 때 최소화된 메시지 전송시간이 가능해 진다. 이 경우에는 전체 메시지 전송을 위해 JMS 처리시간(JPT)이 단 1회만 필요로 하기 때문이다. 그 외의 $\frac{ns}{r}$시간만큼의 실제 메시지 내용을 전달하기 위한 전송시간만이 추가로 필요하게 된다.

다음으로는 식 (1)을 발전시켜 네트워크나 응용 시스템의 오류 때문에 발생하는 전송오류를 반영하도록 한다. JMS 메시지 플랫폼의 신뢰성 보장 기능은 만일 전송 시 어떠한 이유에서의 전송오류가 발생할 경우 해당 메시지를 롤백처리하고 다시 전송하도록 한다. 따라서 전송오류가 발생한 경우에는 앞서 전송했던 부분적인 메시지 내용이 유실되고 다시 처음부터 해당 메시지를 전송해야 하기 하며, 이를 위한 메시지 롤백 처리시간이 추가적으로 필요하다.

메시지 롤백이 발생할 경우 JMS 메시지 플랫폼 상에서 추가적으로 필요로 하는 시간을 b, 메시지 송수신 에러계수를 e 로 정의하도록 한다. $e \times s$는 단위 메시지 송수신 시 발생하는 평균 에러건수로서, 예를 들어 e =0.0003, s=3일 경우 단위 메시지를 송수신 할 때 평균 0.0009회의 에러가 단위 메시지에서 발생함을 의미한다.

JMS 메시지 플랫폼은 각 전송 메시지가 개별적으로 동작하고 전송 메시지 간에 일정한 관계가 없기 때문에 메시지 송수신 에러계수(e)는 단위 메시지 전송 시에 개별적으로 적용하기로 한다. 따라서 한 단위 메시지가 전송오류가 발생하여 메시자가 롤백된 경우 같은 단위 메시지가 재전송될 경우에도 마찬가지로 동일한 메시지 송수신 에러계수의 영향을 받게 된다.

다음으로는 앞서 정의한 평균 메시지 롤백 처리시간(b), 메시지 송수신 에러계수(e)를 고려한 i번째 단위 메시지 처리시간의 기댓값 $E(t^{B_i})$을 알아보면 $es(b + \frac{1}{2}\frac{s}{r}) + t_i$와 같다.

하지만 각 단위 메시지에서 일정 확률로 에러가 발생할 경우 전송 중이던 부분이 무시되고 새로운 전송이 이루어지게 되므로 위의 결과에 추가적인 시간이 더 필요하게 된다. 또한 전체 메시지 M의 총 전송시간은 아래 식 (2)와 같다.

$$E(T^B) = E(t_1) + E(t_2) + \ldots + E(t_n) = nes(b + \frac{s}{2r}) + T = ebns + \frac{ns^2 e}{2r} + T \quad (2)$$

전송오류 발생을 고려한 총 전송시간은 메시지 롤백 처리를 위한 시간 및 전송 중이던 메시지 유실로 인해 소모되는 시간이 더해진다. 메시지 롤백이 존재하지 않는 무결한 시스템에서는 단위 메시지 크기가 클수록 총 메시지 전송시간이 줄어들지만, 메시지 롤백이 존재한다면 반대로 단위 메시지 크기가 작을수록 총 메시지 전송시간이 줄어들게 된다.

메시지 롤백이 존재하는 환경에서 단위 메시지 크기를 매우 작은 단위로 줄이게 되면 식 (2)에서의 총 메시지 전송시간(T)값이 커지기 때문에 일정 이상부터는 오히려 $E(T^B)$값이 증가하게 된다. 이때 $n \times s$는 전송하고자 하는 총 메시지 크기와 동일하며 상수로 취급될 수 있다. 따라서 최적 단위 메시지 크기(s^*)는 다음 식 (3)과 같이 구해질 수 있다.

$$s^* = \sqrt{\frac{2rJPT}{e}} \quad (3)$$

그러므로 식 (3)과 같이 단위 메시지 전송크기는 메시지 전송률이 높을수록 그리고 JMS 처리시간(JPT)이 클수록 높아져야 하고, 메시지 송수신 에러계수(e)가 높을수록 작아져야 한다.

이를 이용하여 주어진 메시지 전송률과 메시지 송수신 에러계수 등의 상수에 대한 최적 단위 메시지 전송크기(s^*)를 계산해 낼 수 있다.

최적 메시지 송수신 크기(s^*)를 계산하기 위해 식 (3)를 이용하기 전 우선 송신측 및 수신측 시스템에 대해

여러 변수에 대한 값을 추출하여야 한다.

추출해야 하는 변수는 네트워크 송수신 속도(r), 메시지 송수신 에러계수(e), 네트워크 송수신 속도(r) 및 JMS 처리시간이다. 이들 변수는 JMS 시스템의 상태나 운영 환경에 따라 매우 달라질 수 있기 때문에 해당 JMS 시스템의 운영시점 바로 전의 설정시점에서 추출되어야 가장 이상적인 최적화된 단위 메시지 크기(s^*)를 계산할 수 있다.

이를 위해 다양한 방법이 존재할 수 있겠지만, 본 논문에서는 테스트 메시지를 이용하여 단위 메시지 크기를 조사할 수 있는 방법을 취하고자 한다. 우선 송신측 및 수신측 시스템을 제어하여 일정한 간격을 가지고 정해진 시간 내에서 지속적인 테스트 메시지를 전송하도록 한다. 테스트 메시지의 크기는 시스템 상황에 따라 매우 상이하므로 사전에 정의하는 것이 어렵지만, 일반적인 경우에서는 약 10 Mbytes 정도 크기가 되는 임의의 단위 메시지를 가지고 테스트하는 것이 가장 적합한 것으로 판단된다. 테스트 메시지가 지속적으로 송수신되는 동안 JMS API와 응용 시스템으로부터 필요한 변수를 추출하게 된다. 추출된 변수를 바탕으로 최적 메시지 송수신 크기(s^*)를 구한 후, 이는 각각의 송신측 및 수신측 시스템에 설정되어 향후 대용량 메시지 처리 시 단위 메시지 구성의 기준 값으로 활용되어 진다.

최적 메시지 송수신 크기(s^*)가 송신측 및 수신측 시스템에 세팅된 후 JMS 응용 시스템은 이에 따라 전송할 메시지를 분할 및 병합하여야 한다. 따라서 본 논문에서 제안하는 분산 환경에서의 대용량 메시지 송수신 최적화 시스템을 구현하였다. 구현 시스템의 시스템 구성 및 역할은 다음과 같다.

- 시스템 라이브러리 - 송신 및 수신 프로그램과 JMS API 사이에 위치하는 모듈로서, 컨트롤러의 설정에 의거하여 최적화된 송신 메시지를 구성하는 역할을 한다.
- 송수신 컨트롤러 - 데이타의 크기와 양을 분석하여 전송 시 최적화된 형태로 메시지의 형태를 구성할 수 있도록 시스템 라이브러리를 조절하는 역할을 한다.
- 컨트롤러 설정 - 컨트롤러가 구동 시에 참조할 운영 규칙을 명시한다.

우선 시스템의 운영 전 설정 조절 단계에서는 송/수신 시스템의 통신 구간에 시스템 라이브러리를 배치한 후 시스템에서 연계될 가상의 데이타를 생성하여 충분한 메시지 송수신 테스트를 거친 후 최적 메시지 송수신 크기 도출 로직에 의하여 최적화된 메시지를 구성할 수 있도록 시스템 라이브러리의 운영정보를 설정한다. 컨트롤러가 단위 메시지 구성의 크기를 결정하기 위한 중요 로직은 다음과 같다.

```
//최근 송수신건수
int compareBasis;
//현재의 최적 메시지 크기
public static int standardMsgSize;
//모니터링 정보를 저장하는 배열
public static ArrayList monitorInfoStore;
//최적 메시지 크기를 저장하는 배열
public static ArrayList msgSizeStore
= new ArrayList();

public static void scheduleWake() throws
LVMSException  {

//이번 모니터링시 도출될 최적 메시지 크기
int sizeOfMessage=0;
//최적 메시지 크기의 변화 추세
int msgSizeTendency=0;

for(int i=0;i<indexArr.size();i++) {
LVMSLibIdx libIdx
   = (LVMSLibIdx)indexArr.get(i);

//최적 메시지 크기를 도출해내기 위한 상수 값을
//해당 LVMS library로부터 도출
meta = extractCurrentMetaData(libIdx.getIndex());
                        addMonitorInfoStore(meta);

//최적 메시지 크기 계산
sizeOfMessage
  = sqrt((2*meta.R*(meta.C+meta.Q+meta.F))/meta.e);
addMsgSizeStore(sizeOfMessage);

//메시지 크기 변화 추세를 계산한다.
msgSizeTendency =
  getMsgSizeTendency(compareBasis);

//만약 현재 최적 메시지 크기보다 변화추세가 크다면
//최적 메시지 크기를 배포하고, 설정 변경
if(msgSizeTendency>standardMsgSize) {
deployNewMsgSize(libIdx.getIndex(),sizeOfMessage);
standardMsgSize=sizeOfMessage;
}
}
}
```

시스템 운영 전 최적화 테스트를 거쳐 라이브러리를 설정하였다 하더라도 운영 중에는 언제든지 변동 상황이 발생할 수 있다. 그러므로 컨트롤러는 주기적으로 가상의 데이타를 이용한 시스템 운영환경을 파악하여 라

이브러리의 설정이 최적화된 상태가 아니라면 라이브러리의 운영정보를 재설정하여 유동적으로 시스템의 환경 변화에 대처할 수 있도록 한다.

실제의 대용량 배치처리 환경에서는 일반적으로 일, 주, 월 단위 또는 분기 단위의 비교적 긴 처리 주기를 바탕으로 시스템이 운영된다. 따라서 처리 주기 사이에 라이브러리의 운영정보를 재설정하기 위한 테스트 메시지를 발생시킴으로써 시스템 운영에 영향을 미치지 않는 범위에서 운영정보를 갱신할 수 있으며, 이를 바탕으로 시스템이 가장 효율적인 방법으로 메시지를 송수신할 수 있도록 한다.

라이브러리는 JMS 클라이언트에서 제공되는 JMS API를 응용하며 표준화된 인터페이스를 응용 프로그램에 제공한다. 따라서 응용프로그램 작성자는 JMS API에 관여 받지 않을 뿐 아니라 대용량 데이타 송수신 시 최적화된 메시지 송수신이 신뢰성 있게 이루어짐을 보장받을 수 있다.

4. 실 험

다음으로는 본 연구에서 제안한 최적 메시지 도출기법과 이를 구현한 시스템의 효과를 측정해보기 위한 실험을 수행하고 결과를 분석한다. 인터넷을 통해 연결된 메시지 송신 시스템과 수신 시스템이 대용량 데이타를 JMS 플랫폼을 통해 송수신 하는 환경에서 시스템을 다양한 조건으로 운영하여 기존의 일반적인 고정적 분할 방식에 대비한 효과를 살펴보았다.

실험을 실시한 환경은 인터넷을 기반으로 하루에 한 번씩 대용량의 데이타를 전송 처리하는 일괄처리 데이타 송수신 시스템과 잦은 데이타 송/수신 처리가 발생하는 시스템을 선택하였다. 실험 조건은 아래와 같다.

- 하드웨어: IBM RS6000 서버, HP ProLiant DL360
- 운영체제: AIX, HPUX
- 네트워크: 평균속도 230kbytes/sec의 인터넷 환경
- 자바 런타임 환경: JRE 1.4.2
- JMS 프로바이더: Bizstore Indigo 3.0 MCManager

우선 기존 방식 대비 효과를 알아보기 위해 데이타 구현 시스템을 사용하여 10Mb부터 500Mb 까지 메시지 사이즈를 달리하면서 각 100 건씩 데이타를 전송하였다.

데이타 분할 단위를 10Mb단위로 하여 10 / 50 / 100 / 300 / 500Mb 크기의 데이타를 100건 전송하는 실험을 본 시스템과 일반적인 JMS API를 사용한 송수신 시스템에서 10회 실시하였으며, 실험 결과는 표 1, 표 2와 같다.

첫 번째 실험인 대용량 데이타 전송의 경우 데이타의 크기가 비교적 작을 때는 일반적인 송수신 시스템과 구현 시스템과의 처리시간의 차이가 크지 않았지만, 데이타의 크기가 커지면 커질수록 구현 시스템 사용 시와의 차이가 현격히 커진다는 것을 알 수 있다. 특히 전송량이 가장 많은 300Mbytes의 100회 전송의 경우 일반적인 JMS 송수신 시스템에서의 평균 전송시간이 141330.5초 이고 구현 시스템 사용 시 평균 전송시간이 132400.4초로써 약 149분의 차이가 발생하게 된다. 실험의 결과는 그림 2와 같다.

표 1 일반적인 송수신 시스템에서 대용량 데이타 전송 실험 결과(단위: 초)

메시지 크기	10Mb	50Mb	100Mb	200Mb	300Mb
평균 시간	4,298.2	24,033.4	450,925	90,506.7	141,330

표 2 구현 시스템을 사용한 대용량 데이타 전송 실험 결과(단위: 초)

메시지 크기	10Mb	50Mb	100Mb	200Mb	300Mb
평균 시간	4,400.8	22,095.6	44,041.7	87,224	132,400

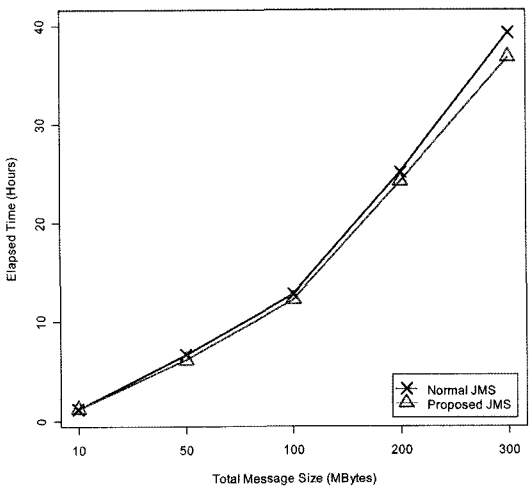

그림 2 일반적인 JMS 송수신과의 비교

두 번째 실험은 단위 전송 용량은 적으나 다량의 데이타를 송/수신할 경우를 가정하여, 1Mbytes 단위의 데이타를 10000건 전송 시 1회 전송량 조건에 따른 총 전송시간 비교 실험을 10회씩 실시하였다. 실험에 사용된 변수들은 아래와 같다.

- r = 0.23 (단위: Mbytes/sec)
- JMS 처리시간 = 0.56 (단위: 초)
- e = 0.00045
- b = 0.05 (단위: 초)

주어진 조건에서 최적 단위 메시지 크기(s^*)는 23.93Mbytes로 계산된다. 이 결과를 바탕으로 1회 전송량 조건을 우선 일반적인 JMS 송수신 시스템의 경우인 각 데이타의 개별 전송(1회 1Mbytes), 그리고 1회 전송량을 아주 크게 가정하였을 경우(1 회 500 Mbytes와 1000Mbytes), 그리고 도출된 최적 단위 메시지 크기가 실제로 유효한지를 비교하기 위하여 위에서 구한 23.93Mbytes와 근접한 21/23/25Mbytes의 총 6가지 조건으로 실험을 하였다. 결과는 아래 표 3과 같다.

위 실험 결과의 평균을 그래프로 나타내어 보면 그림 3과 같다. 두 번째 실험의 결과를 보면 10000건의 데이타를 트랜잭션 당 1 Mbytes씩 전송할 경우와 한 번에 매우 큰 데이타를 전송할 경우 전체 전송시간이 가운데의 21 에서 25 Mbytes에 비하여 눈에 띄게 높은 것을 알 수 있다. 이는 1회 전송량이 적을 경우에는 메시지 자체의 전송시간 외에 메시지 서버 및 큐와의 통신시간 및 피드백 시간이 전송 시간에 추가가 되고, 전송 데이타 자체의 크기가 적절한 수치를 넘어서게 되면 오류 발생의 가능성이 높아지기 때문에 재전송 시간이 오히려 더 늘어나므로 전송시간이 늘어나게 되기 때문이다.

실험 결과의 평균 전송시간을 보면 실험 전 도출한 최적 단위 메시지 크기보다 약간 큰 25Mbytes에서 전송시간이 가장 낮게 나오는 것을 알 수 있는데, 이는 네트워크 속도가 정확하게 유지되지 않고 에러계수의 편차 때문에 나타나는 차이로 볼 수 있다.

표 3 구현 시스템을 이용한 소 용량 데이타 묶음 전송 실험 결과(단위: 초)

메시지 크기	1Mb	21Mb	23Mb	25Mb	500Mb	1000Mb
평균 시간	49,098	44,028	43,979	43,970	48,462	52,365

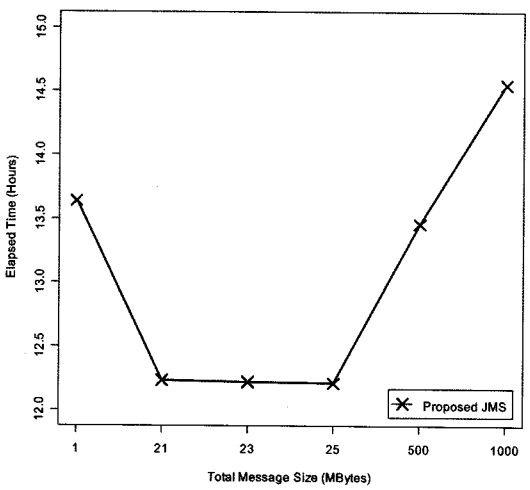

그림 3 단위 메시지에 따른 전송시간 비교

실험 결과는 네트워크 상태를 측정하고 시스템을 설정하기 위해 필요한 테스트 메시지 송수신 오버헤드를 포함하지 않는다. 본 연구의 배경인 대용량 일괄처리 환경에서는 추가적인 오버헤드가 전체 시스템 성능에 영향을 미치는 실시간 처리와는 달리 테스트 메시지 송수신 오버헤드를 실제 시스템 운영 전에 시스템 설정 및 튜닝 단계에서 이루어지는 작업으로 볼 수 있다.

위의 두 가지 실험으로 얻을 수 있는 결과를 통해 JMS 송수신 시스템 구축 시 송수신 환경에 최적화된 메시지의 크기를 도출할 수 있고 이를 모니터링 하여 시스템 환경의 변화나 전송되는 데이타 량의 변화에도 최적화된 상태를 유지할 수 있다면 효율적이고 안정된 시스템을 운영할 수 있다고 판단된다.

5. 결 론

오늘날의 응용 시스템 통합의 많은 부분이 인터넷과 같은 분산된 형태의 시스템 간에서 이루어지고 있으며, 이를 위해 JMS 메시지 플랫폼이 매우 많이 활용되고 있는 실정이다. 특히 대용량 메시지 송수신 시에는 일반적으로 일정한 단위 메시지로 분할하거나 합쳐서 전송하는 경우가 많은데, 단위 트랜잭션 처리를 위한 방식과는 다른 고려사항이 존재한다.

본 논문에서는 JMS 메시지 플랫폼을 이용하여 분산 환경에서의 대용량 메시지 송수신 시 메시지 롤백이 발생함으로써 단위 메시지 크기에 따라 총 메시지 송수신 시간이 의존적임을 밝히고, 주어진 환경에서 최적화된 단위 메시지 크기를 계산하는 기법을 제시하였다. 또한 제시한 기법을 시스템에 구현하여 비교 실험하여 본 기법이 실제 환경에서 송수신 시간을 상당히 줄일 수 있음을 확인하였다.

제시한 최적화 단위 메시지를 계산하고 적용하는 기법은 기존에 연구되었던 메시지 송수신 효율을 높이는 방식에 비해 송수신되는 단위 메시지 크기를 통해 전송시간을 최소화하려는 방법에서 차별화된다고 할 수 있다. 또한 본 연구의 결과는 JMS 응용 시스템에서 일반적으로 활용되고 있는 임의로 선정한 단위 메시지 크기를 이용하여 대용량 메시지를 송수신 하는 방법에 대한 비효율성을 제거할 수 있는 실제적이고 구체적인 의미를 갖는다고 할 수 있다.

하지만 JMS 메시지 플랫폼을 이용하여 메시지를 송수신 할 때 본 논문에서 상수로 설정한 JMS 처리시간(JMS 연결 시간, 큐 처리 시간 및 피드백정보 송수신시간)은 실제적으로는 매우 동적으로 변화하기 마련이다. 따라서 향후 이러한 값들에 대한 패턴과 송수신 메시지

크기 과의 비교를 분석하여 반영하는 연구가 이루어진다면 본 기법의 효과와 정확성이 더욱 높아 질 것으로 판단된다.

참 고 문 헌

[1] Michael Kovacs, Paul Giotta, Sott Grant, "Professional JMS," Wrox, 2001.

[2] JEFFREY C. LUTZ, "EAI Architectural Patterns," EAI Journal, 2000.

[3] 문남두, 이근웅, "분할 가능한 분산환경에서 견고한 자바 객체 그룹을 지원하는 그룹통신 모델의 설계", 한국 정보 과학회, 제28권, 제2호(III), 649-651, 2001.

[4] 노성주, 정광식, 이화민, 유헌창, 황종선, "분산 이동 시스템에서 인과적 메시지 전달을 위한 효율적인 프로토콜", 한국 정보 과학회 논문지: 정보통신, 제30권, 제2호, 2003.

[5] IBM JMS(Java Message Service), "http://www-903.ibm.com/developerworks/kr/java/library/j-jmsvendor.html".

[6] Francis X. Maginnis, William A. Ruh, "Enterprise Application Integration," John Wiley & Sons, 2000.

[7] R. Koo, S. Toueg, "Checkpointing and rollback recovery for distributed systems," IEEE Transactions on Software Engineering, 13(1), 23-31, 1987.

[8] D Kuo, D Palmer, "Automated Analysis of Java Message Service Providers," Lecture Notes in Computer Science, 2218, 1-14, 2001.

[9] JMS Message Types Reference Document, "http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/index.jsp?topic=/com.ibm.IBMDI.doc/referenceguide91.htm."

[10] Kim Haase, "Java Message Service API Tutorial," Sun microsystems, Inc., 2002.

[11] Java Message Service(TM) API Specification, "http://java.sun.com/products/jms/javadoc-102a/index.html."

[12] M Menth, R Henjes, C Zepfel, S Gehrsitz, "Throughput Performance of popular JMS servers," Proceedings of the joint international conference on Measurement and modeling of computer systems, 367-368, 2006.

[13] I.Gorton, Justin Almquist, Nick Cramer, Jereme Haack, Mark Hoza, "An Efficient, Scalable Content-Based Messaging System," In Proceedings of The 7th IEEE International Enterprise Distributed Object Computing Conference, 278-285, 2003.

조 풍 연

1996년 동국대학교 정보산업대학원 전자계산과(공학석사). 2005년 숭실대학교 일반대학원 컴퓨터학과. 관심분야는 EAI/BPM, 웹서비스, SOA/ESB, RFID, XMLDB

박 재 원

2006년 숭실대학교 컴퓨터학과(석사). 2006년~현재 숭실대학교 컴퓨터학과 박사과정. 관심분야는 소프트웨어테스팅, 소프트웨어 프로세스, 웹서비스, SOA/ESB, 소프트웨어 아키텍처

최 재 현

2006년 숭실대학교 일반대학원 컴퓨터학과 졸업(공학석사). 2006년~현재 숭실대학교 일반대학원 컴퓨터학과 박사과정 관심분야는 소프트웨어 개발방법론. 소프트웨어프로젝트 관리, 소프트웨어 아키텍처, SOA/ESB, 웹서비스

이 남 용

1983년 고려대학교 경영정보학과(석사) 1993년 미시시피주립대학 경영정보학과(경영학박사). 1979년~1983년 국군정보사령부 정보처 정보시스템분석 장교. 1983년~1999년 한국국방연구원 군수체계 및 정보체계연구부장. 2000년 한국전자거래학회 논문편집위원장. 2004년 한국정보통신기술사협회 회장. 1999년~현재 숭실대학교 컴퓨터학과 교수. 관심분야는 소프트웨어테스팅, 시스템엔지니어링 등