에이전트-온-디맨드를 이용한 분산 시스템 관리

(Distributed System Management using Agent-On-Demand)

설 승 진 † 이 금 석 ††
(Seungjin Sul) (Keumsuk Lee)

요 약 분산 시스템이 널리 적용됨에 따라 효율적인 분산 시스템 관리 방안에 관한 연구가 다양하게 진행되고 있다. SNMP나 CMIP에 바탕을 둔 클라이언트/서버 방식의 분산 시스템 관리 환경은 확장성, 상호운영성, 유연성 등과 관련하여 많은 제한점을 드러내고 있다. 근래에는 이러한 단점들을 극복하기 위해 시스템 관리에 이동 에이전트(mobile agent)의 적용 관한 연구가 집중적으로 이루어지고 있으며, 자바 언어의 출현으로 인해 가속화되고 있다. 하지만 이동 에이전트 기법을 시스템 관리에 적용하여 얻을 수 있는 성능 향상에 대해서는 분석이 필수적이다.

본 논문에서는 이동 에이전트를 이용한 시스템 관리의 효율성을 분석하고 이를 개선할 수 있는 에이전트-온-디맨드 방식을 제안한다. 에이전트-온-디맨드 방식은 관리자 응용(manager application)이 관리 작업을 수행하는 이동 에이전트를 관리 대상 노드에 파견(dispatch)하는 일반적인 방식이 아니라 관리 대상 노드가 관리자 응용에게 특정 에이전트의 파견을 요청하는 방식으로 이를 위해 계층적 상태 임계값(Hierarchical State Threshold)을 사용한다. 성능 분석을 위해 자바 RMI와 이동 에이전트를 위한 분석 모델을 제시하고 AOD를 적용한 분산 시스템 관리 기법과 기존의 방법을 네트워크 부하와 실행 시간 관점에서 비교한다.

Abstract As distributed systems become used in wider area of applications, many works has been done to invent more efficient way to manage the distributed systems. The client-server based distributed system management by using SNMP or CMIP has many problems such as scalability, interoperability, flexibility, and so on. Recently, it is evident that managing distributed systems using mobile agents have popularity, and the Java language helps the trend. However, the improvement of performance has to be analyzed when the mechanism of mobile agent is applied

In this paper, we discuss a more efficient way to make use of the mobile agent mechanism for managing distributed systems, and propose an Agent-On-Demand (AOD) method. The core of the method is not to use the previous method that mobile agents doing management job is dispatched to the managed nodes by a manager application, but to make agents requested by the managed nodes and then the manager application send the requested agents to that nodes. This process is done through Hierarchical State Threshold (HST). Also we present a performance model for Java RMI and mobile agent, and compare the AOD method with the previous work in terms of network overhead and execution time.

1. 개 요

분산 시스템은 효율적인 자원 공유, 확장성 및 작업 부하의 분산 등과 같은 많은 장점을 제공하기 때문에 다양한 분야에 걸쳐 적용되고 있다. 하지만 분산 시스템의 규모가 방대해지고 더욱 다양한 서비스들이 제공됨에 따라 뜻하지 않은 성능 저하나 자원 할당의 비효율성, 그리고 물리적으로 원거리에 위치한 장치들의 고장 및 보안 문제 등 심각한 문제점들이 노출되고 있다. 이러한 문제들을 해결하기 위해 효율적인 분산 시스템 관

† 학생회원 : 동국대학교 컴퓨터공학과
ssj@cakra.dongguk.ac.kr
†† 정 회 원 : 동국대학교 컴퓨터공학과 교수
kslee@cakra.dongguk.ac.kr
논문접수 : 1998년 12월 16일
심사완료 : 1999년 11월 1일

리 방안에 관한 연구가 활발히 진행 중에 있으며, 대부분 SNMP(Simple Network Management Protocol)나 CMIP(Common Management Information Protocol)를 적용하는 방향으로 연구가 진행되었다.

그러나 SNMP나 CMIP를 기반으로 하는 중앙집중형 분산 시스템 관리 방법은 단순한 폴링(polling)을 기반으로 하며, 중앙 관리자 응용이 관리 대상에 대한 관리 정보의 수집으로부터 분석 및 제어 연산의 실행까지 모든 작업을 처리하는 것이 일반적이다. 이러한 관리 방식은 중앙 관리자 응용으로 처리 부하가 집중될 뿐만 아니라 네트워크 통신량을 급격히 증가시키는 결과를 초래한다. 더욱이 관리자 응용에 내장된 관리 작업이나 관리 정책 등을 쉽게 변경할 수 없기 때문에 확장성, 상호운영성, 신뢰성 및 유연성을 저하시키는 문제점을 갖는다[2].

이러한 중앙집중형 시스템 관리의 문제점을 해결하기 위해 관리자 응용의 관리 기능을 관리 대상 노드로 위임(*delegation*)하기 위한 연구와 함께 이동 에이전트 개념을 시스템 관리에 적용하기 위한 노력이 이루어지고 있으며, 주로 네트워크 관리 분야에서 많은 성과를 이루고 있다. 이동 에이전트에 기반을 둔 시스템 관리는 관리상의 복잡성을 완화시킬 수 있으며, 이동성 및 지능성을 활용하여 관리 시스템의 확장성과 호환성을 증대시킬 수 있다[2][3][4].

그러나 분산 시스템 관리를 위한 이동 에이전트는 각종 정책과 기능들을 포함하기 때문에 그 크기가 상당히 커지며, 기능 면에서의 복잡성도 증가하여 실행 시간을 지연시키게 된다. 이러한 대규모 이동 에이전트를 분산 시스템을 구성하는 모든 노드로 파견하는 것은 네트워크 자원과 관리 대상 노드의 시스템 자원을 지나치게 사용하게 된다.

따라서 본 논문에서는 계층적 상태 임계값(Hierarchical State Threshold; 이하 HST)을 바탕으로 한 에이전트-온-디맨드(Agent-On-Demand; 이하 AOD) 방법을 제안한다. AOD 방법에서는 전체 관리 작업을 수행하는 대규모 이동 에이전트를 모든 관리 대상 노드로 파견하는 일반적인 방법에서 탈피하여 관리 대상 노드가 필요한 에이전트의 파견을 요청하도록 하고, 관리자 응용은 요청된 에이전트, 즉 해당 관리 대상 노드를 관리하기 위해 필요한 기능만을 갖는 에이전트를 여행 계획(travel plan)을 사용하여 파견한다. 제안한 AOD 방법에서의 HST는 관리 대상 노드가 파견된 에이전트의 업그레이드를 요청하는 기준을 제공한다. 즉, 관리작업 수행하는 도중 더 많은 기능을 갖는 에이전트를 필요로 할 경우, 관리 대상 노드는 에이전트 업그레이드를 요청하게 되는데 이러한 요청은 각 관리 대상 노드에서 유지하는 HST를 기반으로 이루어진다.

성능 분석을 위해 자바 RMI와 이동 에이전트를 위한 성능 모델을 제시하고, 제안한 AOD 방법의 성능을 평가하였다.

본 논문의 구성은 2장에서 이동 에이전트 모델과 분산 시스템 관리, 3장에서 제안한 AOD에 대해 설명한다. 4장에서는 성능 모델을 제시하며, 5장에서는 제시된 모델을 기반으로 성능을 평가한다. 마지막으로 6장에서 결론 및 향후 연구에 대해 기술한다.

2. 관련 연구

2.1 이동 에이전트 모델

2.1.1 이동 에이전트의 정의

이동 에이전트는 적용 분야에 따라 차이는 있지만 다른 개체(entity)의 역할을 대행하는 소프트웨어 프로그램이라는 공통된 개념을 갖는다. 또한 이동 에이전트는 어느 정도의 자치성(autonomy)을 가지며, 명세된 행위의 수행(proactivity)뿐만 아니라 발생하는 사건에 따라 적절한 반응 조치(reactivity)를 행할 수 있어야 한다.

2.1.2 이동 에이전트의 분류

이동 에이전트는 에이전트의 이동을 요청하는 주체에 따라 원격 실행(remote execution)형과 코드-온-디맨드(*Code-On-Demand*)형으로 구분되며, 이동 에이전트가 유지하는 상태 정보에 따라 강한 이주(strong migration)와 약한 이주(weak migration)의 형태로 구분할 수 있다[5].

분산 시스템 관리 측면에서 살펴보면, 원격 실행형은 관리자 응용이 특정 이동 에이전트를 관리 대상 노드에 파견하는 방식이며 코드-온-디맨드형은 이동 에이전트를 받아들이는 관리 대상 노드에서 에이전트의 파견을 요청하는 형태라고 할 수 있다. 그리고 강한 이주는 이동 에이전트의 이동시 에이전트의 코드, 데이타 및 상태 정보 모두를 목적지 노드로 이전하는 형태인 반면, 약한 이주는 단지 데이타 및 관련된 상태 정보만을 이전한다.

2.2 이동 에이전트에 기반한 분산 시스템 관리 모델

이동 에이전트를 적용한 분산 시스템 관리는 중앙 관리자 응용이 수행해야 하는 관리 작업을 이동 에이전트에게 위임하고, 이동 에이전트가 관리 대상 노드로 파견되어 관리 작업을 수행한 후, 그 결과를 관리자 응용에게 전송하는 형태를 갖는다. 이러한 이동 에이전트는 관리 대상 노드의 상태에 따라 특정 행위를 수행을 독립적이고 자치적으로 수행하게 된다. 이동 에이전트는 관

리자 응용의 단순한 기능만을 대행하는 수동적인 형태와 관리 대상 노드의 상태 변화에 따라 지능적으로 의사결정을 내리고 필요한 행위를 수행할 수 있는 능동적인 형태로 구분할 수 있다.

분산 시스템 관리에 있어 수동적 이동 에이전트는 단순한 관리 정보의 수집을 예로 들 수 있다. 즉, 관리자 응용은 관리 정보의 수집 작업만을 수행하는 간단한 이동 에이전트를 관리 대상 노드에 파견하여 필요한 관리 정보를 되돌려 받고, 이러한 관리 정보를 종합하여 각각의 관리 대상 노드에 적절한 조치를 취하게 된다. 반면, 능동적인 이동 에이전트는 관리 정보의 수집과 더불어 관리 대상 노드의 상태 변화에 따라 적절한 행위를 수행할 수 있는 능력까지 포함되므로 훨씬 더 유연한 관리 작업이 가능하다.

이동 에이전트를 분산 시스템 관리에 적용하기 위해서는 (그림 1)과 같은 구성 요소를 필요로 한다. 우선 관리 인터페이스는 관리자 혹은 관리자 응용이 이동 에이전트에게 위임할 관리 작업을 입력받고, 이를 바탕으로 이동 에이전트를 생성한다. 관리자 응용의 에이전트 파견기는 관리 대상 노드로 생성된 에이전트를 파견하는 역할을 수행하며, 각 관리 대상 노드에는 이동 에이전트를 전송 받아 관리 작업을 수행할 수 있도록 하는 에이전트 호스트가 존재한다.

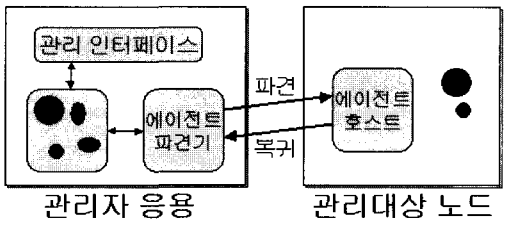

그림 1 에이전트-온-디맨드 시스템의 구성

이동 에이전트를 분산 시스템 관리에 적용하기 위해 고려해야할 다른 문제는 이동 에이전트의 이동 방식에 관한 것이다. 에이전트의 이동 방식은 관리 행위의 결과를 관리자 노드에게 되돌려 주는 문제와도 밀접한 관계가 있다. [2]에서는 네트워크 관리 모델을 비교하여, 이동 에이전트를 링 형태로 순환시켜 관리 작업을 처리하는 것이 폴링 방식보다 나은 성능을 보인다고 결론지었다. 본 논문에서 대상으로 하는 (그림 1)과 같은 분산 시스템 관리 모델은 이동 에이전트의 크기가 전체 관리 시스템에 미치는 영향을 분석하며, 이동 방식은 링 형태의 순환 방식을 선택한다.

3. 에이전트-온-디맨드

제안한 AOD는 관리 대상 노드로 관련된 모든 관리 작업을 처리하기 위한 대규모의 에이전트가 이동하면서 네트워크 및 시스템 자원을 낭비할 필요가 없으며, 또한 이동 에이전트의 관리 기능이 변경되어야 하는 조건의 발생 빈도가 비교적 낮다는 사실에 바탕을 두고 있다. 관리 대상 노드는 관리자 응용에게 더 많은, 혹은 더 적은 기능을 가진 에이전트의 파견을 요청하기 위한 기준으로써 HST를 유지하며, 관리자 응용은 관리 대상 노드의 요청을 기반으로 파견할 에이전트의 종류 및 여행 계획을 생성한 후, 에이전트를 파견한다. 여기서 더 많은 기능을 제공하는 에이전트의 요청을 에이전트 업그레이드(agent upgrade), 그리고 더 적은 기능을 제공하는 에이전트의 요청을 에이전트 다운그레이드(agent downgrade)라고도 한다.

3.1 AOD 시스템의 구성

(그림 2)는 AOD 시스템의 구성을 나타낸 것으로 관리 시스템 가운데 HST 유지 및 에이전트 파견과 관련된 부분만을 나타내었다. 관리자 응용의 에이전트 관리기는 시스템 관리자에 의해 작성된 관리 에이전트의 저장 및 파견을 담당한다. 여행 스케줄러는 관리 대상 노드의 요청들을 전달받으며, 이를 바탕으로 각 에이전트에 대한 여행 계획을 조정한다. 에이전트 호스트는 에이전트 관리기를 통해 파견된 에이전트를 전송 받은 후, 실행가능한 상태로 초기화하고, 사건 모니터에 의해 구성되는 HST를 접근하여 필요하다면 다른 에이전트에 대한 요청 작업을 수행한다. 에이전트 호스트는 자바

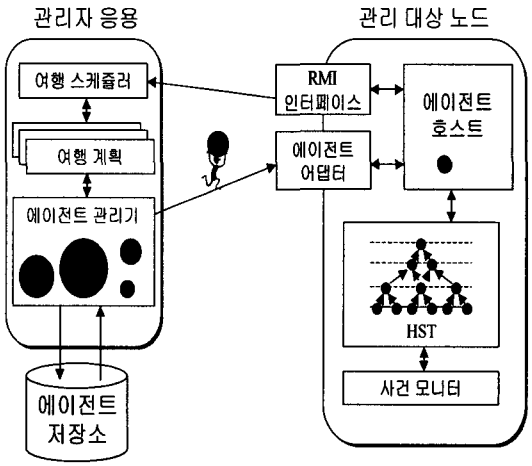

그림 2 에이전트-온-디맨드 시스템의 구성

RMI 인터페이스를 통해 관리자 응용의 여행 스케줄러와 통신한다.

3.2 계층적 상태 임계값

각각의 관리 대상 노드가 유지하는 HST는 시스템 관리자에 의해 작성된 관리 정책(management policy)을 바탕으로 구성된다. 관리 대상 노드의 상태 정보는 위험성 수준에 따라 여러 등급으로 구성될 수 있다. 예를 들면, 위험성 수준은 안전-임계 시스템(safety-critical system)의 위험 등급(risk level)을 따라 4 가지 단계로 설정될 수 있다[7]. 관리자의 관리 정책을 바탕으로 계층 구조로 구성된 HST[9]는 이동 에이전트에 의해 참조되며, 정의된 사건의 발생 여부에 의해 상위 위험성 등급이나 하위 위험성 등급으로 조정될 수 있으며, 이것은 곧 에이전트 업그레이드나 에이전트 다운그레이드를 야기하게 된다. (그림 3)은 HST와 이동 에이전트의 기능과 위험성 등급간의 관계에 대한 예를 나타내었다.

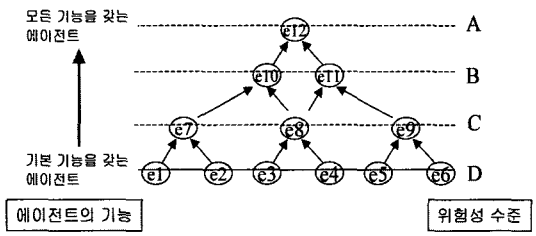

그림 3 HST와 위험성 수준

3.3 여행 계획

AOD에서 여행 계획은 각각의 에이전트 종류별로 생성되며, 시작 노드와 방문 노드로 이루어진 튜플의 집합이다. 여행 계획의 초기값은 가장 크기가 작은 기본 이동 에이전트가 모든 노드를 방문하는 것으로 설정되며, 모든 이동 에이전트는 반드시 관리자 응용으로 복귀한다고 가정한다. 예를 들어, 분산 시스템이 관리자 응용 m과 5개의 관리 대상 노드 a_1, a_2, a_3, a_4, a_5로 구성되었다고 가정하자. 만약 이동 에이전트 MA_1이 모든 관리 대상 노드를 일대일 방식으로 방문한다면 여행 계획 MA_{1tp}는 다음과 같이 표현할 수 있다.

$$TP(MA_1)=\{(m,a_1),(m,a_2),(m,a_3),(m,a_4),(m,a_5)\}$$

다음의 여행 계획은 이동 에이전트 MA_2가 모든 관리 대상 노드를 링 방식으로 순환하는 경우이다.

$$TP(MA_2)=\{(m,a_1,a_2,a_3,a_4,a_5)\}$$

다시 말해서 제안한 AOD에서 이동 에이전트의 모든 이동 형태는 여행 계획을 통해 표현가능하며, 기본적인 이동 형식은 링 방식의 순환으로 가정한다.

4. 성능 모델

4.1 자바 RMI를 위한 성능 모델

자바 RMI의 성능 모델을 언급하는 이유는 현재 구현된 대부분의 이동 에이전트 환경이 자바 언어를 바탕으로 하며, 더욱이 AOD 경우에는 각 관리 대상 노드가 HST의 상태 변화를 기반으로 에이전트에 대한 요청을 관리자 응용으로 전송할 때 자바 RMI를 사용하기 때문이다. 자바 RMI를 위한 성능 모델은 RPC의 경우[8]에서와는 달리 자바 RMI 서버로부터 클라이언트 스텁을 전송받기 위한 오버헤드와 이와 관련된 요청과 응답들로 인한 네트워크 지연 등을 고려해야 한다.

자바 RMI의 연산 과정은 클라이언트 스텁의 요청, 스텁의 전송, 실제적인 연산 요청, 그리고 실제적인 응답의 4 단계를 통하여 이루어지게 된다. 따라서 임의의 노드 L_1에서 L_2로의 자바 RMI 호출에 대한 네트워크 부하(단위; *kbyte*), B_{RMI}는 다음 식(1)과 같이 나타낼 수 있다.

$$B_{RMI}(L_1,L_2,B_{req},B_{rep},B_{stub})=\begin{cases}0, & \text{if } L_1=L_2\\ 2B_{req}+B_{rep}+B_{stub}, & else\end{cases} \quad (1)$$

이미 언급한 바와 같이, 자바 RMI는 실제적인 처리를 위한 요청/응답 연산과 함께 스텁에 대한 요청/응답 연산을 고려해야 한다. 따라서 총 네트워크 지연시간은 4δ로 나타낼 수 있다. 따라서 $L1$에서 $L2$로의 자바 RMI의 실행 시간, T_{RMI}는 다음 식(2)와 같다.

$$T_{RMI}(L_1,L_2,B_{req},B_{rep},B_{stub})=4\delta(L_1,L_2)+\left(\frac{1}{\tau(L_1,L_2)}+2\mu\right)B_{RMI}(L_1,L_2,B_{req},B_{rep},B_{stub}) \quad (2)$$

여기서 δ는 노드간 네트워크 지연 시간, τ는 노드간 네트워크 처리율, μ는 요청, 응답 및 클라이언트 스텁의 전송준비 처리시간(단위; *ms/kbyte*)을 나타낸다. 앞으로 B_{RMI}와 T_{RMI}는 각각 상수 C_b와 C_t로 표기한다.

4.2 이동 에이전트의 성능 모델

이동 에이전트는 코드, 데이타, 상태 정보로 구성되므로 B_{AGENT} = $(B_{code}, B_{data}, B_{state})$로 나타낼 수 있다[8]. 4.1절에서 사용한 것과 동일한 가정 하에서 에이전트 B_{AGENT}가 노드 L_1에서 L_2로 이동하는 경우, 네트워크 부하는 다음 식과 같이 나타낼 수 있다.

$$B_{MOBILE}(L_1,L_2,B_{AGENT})=\begin{cases}0, & \text{if } L_1=L_2\\ B_{req}+B_{rep}+B_{code}+B_{data}+B_{state}, & else\end{cases} \quad (3)$$

자바 RMI의 연산 과정과는 달리, 이동 에이전트는 에이전트 호스트가 이동 에이전트를 수용할 준비가 되었는지를 질의하는 요청과 그에 대한 응답, 그리고 실제적인 에이전트의 이동 과정 순서를 따르게 된다. 그러므로 총 네트워크 지연 시간은 3δ으로 표현할 수 있으며, 이동 에이전트의 실행 시간은 다음 식과 같다.

$$T_{MOBILE}(L_1, L_2, B_{AGENT}) = 3\delta(L_1, L_2) + \frac{B_{MOBILE}(L_1, L_2, B_{AGENT})}{\tau(L_1, L_2)} + \begin{cases} 0, & \text{if } L_1 = L_2 \\ 2\mu B_{MOBILE}, & else \end{cases} \quad (4)$$

4.3 AOD를 적용한 경우의 성능 모델

우선 비교 대상인 AOD를 적용하지 않은 경우와 AOD를 적용한 경우 각각에 대해 초기 상태를 살펴본다. 이동 에이전트의 이동 방식을 링 형태의 순환 방식으로 고정하며, 모두 n개의 노드로 이루어진 분산 시스템을 m번 순환한다고 가정한다. 또한 모델을 간소화하기 위해 AOD에서 요청할 수 있는 이동 에이전트의 종류는 B_{A1}'와 B_{A2}'의 2 가지로 제한한다.

AOD를 적용하지 않은 경우에는 4.2절에서 정의한 식(3)과 식(4)를 만족한다.

다음 식(5)와 식(6)은 AOD를 적용할 경우 첫 번째 순환, 즉 m=1인 경우 네트워크 부하와 실행 시간을 나타낸다.

$$B_{AODfirst}(S, D, B_{A_1}') = \sum_{i=1}^{n}(B_{MOBILE}(D_{i-1}, D_i, B_{A_1}')) + Pn \cdot C_b \quad (5)$$

$$T_{AODfirst}(S, D, B_{A_1}') = \sum_{i=1}^{n}(T_{MOBILE}(D_{i-1}, D_i, B_{A_1}')) + Pn \cdot C_t \quad (6)$$

여기서 S는 관리자 응용, D는 여행 계획에 기술된 이동 대상 노드를 의미한다. 한편 P는 에이전트 업그레이드 확률을 의미한다. 즉, 매 순환마다 n개의 노드 중 $P \cdot n$개의 노드가 에이전트 업그레이드를 요청한다는 것을 의미한다. 기존 에이전트 모델과 가장 큰 차이점은 초기 순환시에 파견하는 이동 에이전트가 B_{A1}'인 점이다. 즉, 초기에는 가장 기본적인 기능만을 갖는 에이전트만을 파견하는 것이 AOD 방법의 특징이라고 말할 수 있다.

AOD를 적용한 경우, 두 번째 순환, 즉 m=2일 때 이동 에이전트 순환에 대한 네트워크 부하와 실행 시간 성능 모델은 다음 식과 같다.

$$B_{AODsecond}(S, D, B_{A_1}', B_{A_2}') = \sum_{i=1}^{(1-Pn)}(B_{MOBILE}(D_{i-1}, D_i, B_{A_1}')) + \sum_{j=1}^{Pn}(B_{MOBILE}(D_{j-1}, D_j, B_{A_2}')) + Pn \cdot C_b$$

$$T_{AODsecond}(S, D, B_{A_1}', B_{A_2}') = \sum_{i=1}^{(1-Pn)}(T_{MOBILE}(D_{i-1}, D_i, B_{A_1}')) + \sum_{j=1}^{Pn}(T_{MOBILE}(D_{j-1}, D_j, B_{A_2}')) + Pn \cdot C_t$$

이미 첫 번째 순환에서 $P \cdot n$개의 노드가 이동 에이전트의 업그레이드를 요청하였으므로 $P \cdot n$개의 노드로 파견하는 이동 에이전트는 B_{A2}'가 된다. 하지만 업그레이드를 요청하지 않은 나머지 $(1-P)n$개의 노드에는 여전히 에이전트 B_{A1}'를 파견한다.

AOD를 적용할 경우, 전체 관리 대상 노드를 m번 순환한다고 가정하였을 때 네트워크 부하와 실행 시간 모델은 식(7)과 식(8)에 나타내었다.

$$B_{AODtotal}(S, D, B_{A_1}', B_{A_2}') = B_{AODfirst}(S, D, B_{A_1}') + \sum_{i=2}^{m}(\sum_{j=1}^{(n-Pn-Q_{i-1})}(B_{MOBILE}(D_{j-1}, D_j, B_{A_1}')) + \sum_{k=1}^{(Pn+Q_{i-1})}(B_{MOBILE}(D_{k-1}, D_k, B_{A_2}')) + Pn \cdot C_b) \quad (7)$$

$$T_{AODtotal}(S, D, B_{A_1}', B_{A_2}') = T_{AODfirst}(S, D, B_{A_1}') + \sum_{i=2}^{m}(\sum_{j=1}^{(n-Pn-Q_{i-1})}(T_{MOBILE}(D_{j-1}, D_j, B_{A_1}')) + \sum_{k=1}^{(Pn+Q_{i-1})} T_{MOBILE}(D_{k-1}, D_k, B_{A_2}')) + Pn \cdot C_t) \quad (8)$$

여기서 Q_{j-1}은 $j-1$번째 순환에서 업그레이드를 요청한 노드의 개수, $P_{j-1} \cdot n$을 의미한다.

5. 성능 평가

성능 평가를 위해 임의의 두 노드 사이의 네트워크 지연시간 δ는 10ms, 네트워크 처리율 τ는 400kbytes/s로 고정하고 이동 에이전트의 크기 B_A는 93kbytes로 하였다. AOD를 적용할 경우 기본 이동 에이전트의 크기인 B_{A1}'는 $B_A/2$인 46.5kbytes, 업그레이드된 에이전트의 크기 B_{A2}'는 B_A와 동일한 93kbytes로 하였다. 또한 자바 RMI의 요청과 응답 크기는 1kbyte, 그리고 자바 RMI 스텁의 크기는 10kbytes로 가정하였다. 성능 평가를 위한 파라미터 값의 선택은 [8]을 참고하였으며, 이동 에이전트의 크기는 실제 자바로 구현된 이동 에이전트의 크기를 충분히 반영하여 결정하였다.

우선 전체 노드 수에 대한 에이전트 업그레이드를 요청하는 노드 수의 비율을 나타내는 P와 네트워크 부하 및 실행 시간과의 관계에 대한 비교는 (그림4)과 (그림5)에 나타내었다.

(그림 4)와 (그림 5)에서와 같이 네트워크 부하의 경우 P=0.45, 그리고 실행 시간의 경우 P=0.3을 기준으로 제안한 AOD 방법과 기존 에이전트 이동 방법의 성능이 급격한 차이를 보이기 시작하는 것을 알 수 있으며, P가 작을수록 효율성이 높아짐을 알 수 있다. 최악의 경우(P=1), 모든 노드가 에이전트 업그레이드를 요청하게 되지만 AOD 방법에서는 첫 번째 에이전트 순환에

서 작은 규모(B_{AI}')의 에이전트를 이동시키므로 일반적인 에이전트 이동 방식과 약간의 성능 차이를 보이고 있음을 알 수 있다.

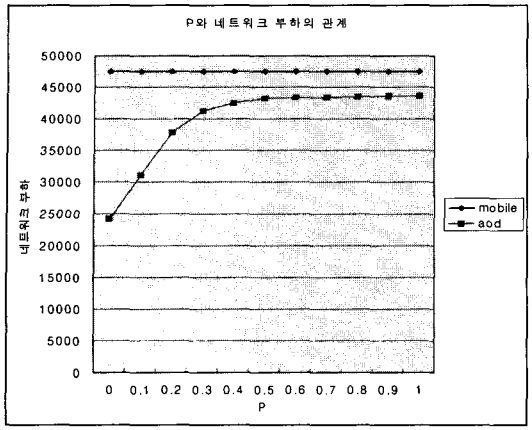

그림 4 요청 비율 P와 네트워크 부하의 관계

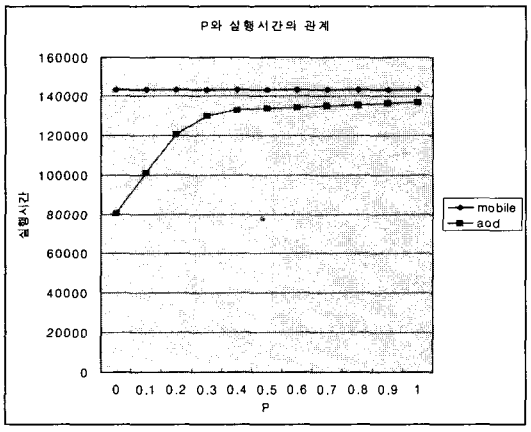

그림 5 요청 비율 P와 실행시간과의 관계

노드 수에 대한 비교는 (그림 6)과 (그림 7)에 나타내었다. 이 경우 업그레이드 요청 확률, P는 0.2로 고정하였다. AOD를 적용한 경우, 네트워크 부하나 실행 시간은 노드 수가 증가할수록 일반적인 에이전트 이동 방식과 많은 성능 차이를 보임을 알 수 있다. (그림 4)에서 살펴본 바와 같이 P를 증가시키거나 감소시키면 (그림 6)과 (그림 7)에서 두 방법의 성능 차이가 감소할 뿐이며, AOD의 성능 개선 양상에는 변화가 없었다.

한편 AOD에서의 기본 에이전트와 기존의 에이전트의 크기 비율에 대한 평가에서도 P를 0.2로 고정하고

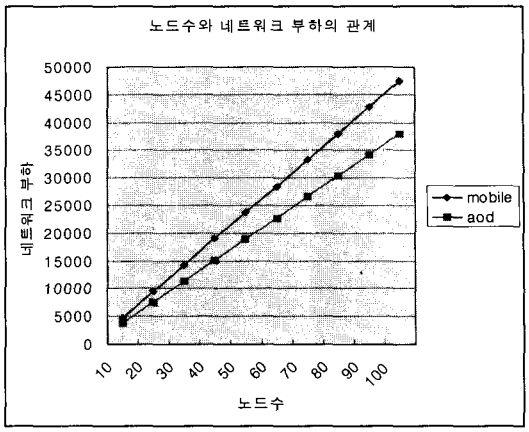

그림 6 노드 수와 네트워크 부하의 관계

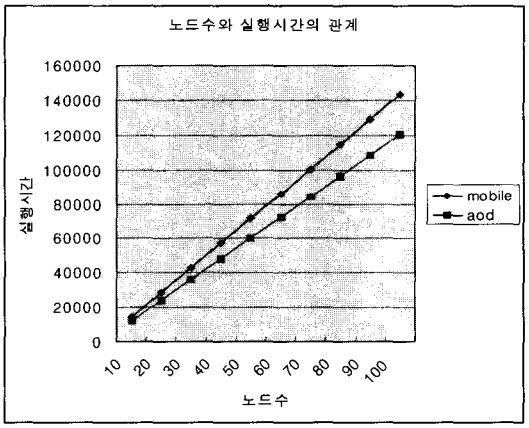

그림 7 노드 수와 실행시간과의 관계

살펴본 결과는 (그림 8)과 (그림 9)와 같다. 이 실험은 AOD를 적용할 경우 에이전트의 크기와 일반적인 에이전트 크기 사이의 관계를 보여주기 위한 것으로, AOD에서 초기 전송 에이전트의 크기는 일반 에이전트보다 약간만 작아져도 성능이 개선됨을 알 수 있다.

일반적인 관리 환경에서 이동 에이전트는 전체 시스템은 계속 반복 순환하면서 관리자 응용이 필요한 정보를 수집하게 되므로 순환 횟수에 대한 평가도 수행하였으며, 역시 AOD를 적용한 경우의 성능이 월등히 나아짐을 알 수 있었다. 또한 AOD에서 HST를 위한 오버헤드라 말할 수 있는 RMI 연산의 부하에 대한 평가 결과는 RMI 연산 부하가 어느 정도 증가하여도 AOD의 성능 개선이 가능함을 알 수 있었다.

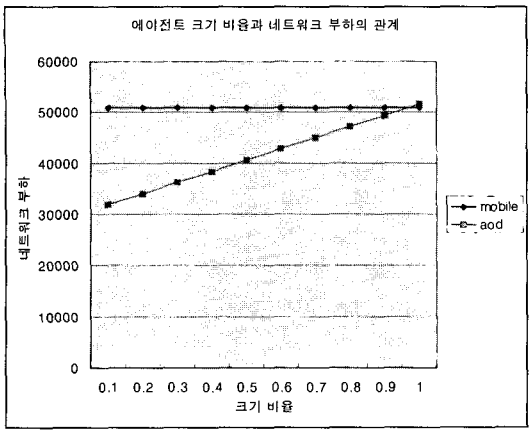

그림 8 이동 에이전트 크기 비율과 네트워크 부하의 관계

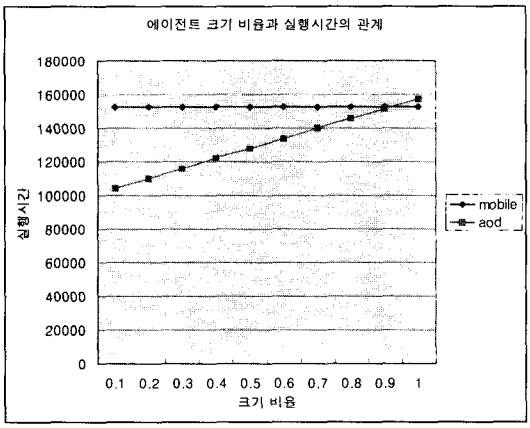

그림 9 이동 에이전트 크기 비율과 실행시간의 관계

6. 결론 및 향후 연구

중앙집중형 분산 시스템 관리는 확장성, 상호운영성, 신뢰성 및 유연성 등의 관점에서 문제점을 드러내고 있으며, 이에 따라 이동 에이전트 개념을 분산 시스템 관리에 접목시키려는 노력이 이루어지고 있다. 본 논문에서는 계층적 상태 임계값(HST)을 바탕으로 한 에이전트-온-디맨드(AOD) 방법을 제안하였다. 제안한 AOD는 성능 평가 결과 네트워크 부하와 실행 시간 관점에서 성능이 개선됨을 알 수 있었다.

본 논문에서의 이동 에이전트에 기반한 분산 시스템 관리 모델에서는 AOD 방법을 적용하였을 때 성능 향상을 기대할 수 있었다. 하지만 실제적인 환경에서 AOD 방법에 대한 타당성을 분석해야 한다. 또한 이동 에이전트의 이동 방식뿐만 아니라 수집된 관리 정보를 관리자 응용에게 전달하는 방법에 관해서도 분석이 필요하다. 제안한 AOD 방법에서는 이동 에이전트가 계속적으로 관리 대상 노드들로 이동하면서 관리 작업을 수행해야 하므로 네트워크 부하 관점에서 개선이 필요하다. 이러한 개선을 위해 이동 에이전트를 이동에 관련된 부분과 특정 행위를 수행하는 부분으로 분리하고 에이전트 업그레이드 요청시에는 필요한 행위 부분만 업그레이드하는 방안에 관해서도 연구가 진행 중이다. 또한 제안한 분석 모델에 대한 증명을 위한 구현 과정도 진행 중에 있다.

참 고 문 헌

[1] Alexander Keller, "System Management with Distributed Objects: Porting SNMP Agents to a CORBA Environment," Proc. of the 4th Workshop of the OpenView University Association OVUA'97, 1997.

[2] Hosoon Ku, et al., "An Intelligent Mobile Agent Framework for Distributed Network Management," Globecom'97, 1997.

[3] Gatot Susilo, et al., "Infrastructure for Advanced Network Management based on Mobile Code," Carleton Univ., 1997.

[4] Gottfried Luderer, et al., "Network Management Agents Supported by a Java Environment," ISINM'97, 1997.

[5] J.Baumann, et al., "Mole-Concepts of a Mobile Agent System," Technical Report TR-1997-15, Stuttgart Univ., 1997.

[6] 유응구, 설승진, 이금석, "SGML과 RDBMS를 이용한 분산 시스템 관리 정책 프레임워크", 한국정보과학회 *98 봄 학술발표 논문집, 제25권, 제1호, 1998, pp.158-160.

[7] Neil Storey, "Safety-Critical Computer Systems," Addison Wesley, 1996.

[8] Markus StraBer and Markus Schwehm, "A Performance Model for Mobile Systems," Proc. of the Int. Conf. on Parallel and Distributed Processing Techniques and Applications PDPTA'97, 1997.

[9] 설승진, 이금석, "분산 시스템 관리를 위한 에이전트-온-디맨드에서의 에이전트 요청 기법", 한국정보과학회 *99 봄 학술발표논문집, 제26권, 제1호, 1999, pp.137-139.

설 승 진

1994년 동국대학교 컴퓨터공학과 졸업. 1996년 동국대학교 컴퓨터공학과 공학석사. 1996년 3월 ~ 현재 동국대학교 컴퓨터공학과 박사과정. 관심분야는 분산컴퓨팅, 분산시스템관리, 이동에이전트

이 금 석

1971년 서울대학교 공과대학 응용수학과 졸업(학사). 1973년 한국과학기술연구소 전산개발센터 전산기술과 근무. 1978년 한국과학기술원 전산학과 졸업(이학석사). 1981년 ~ 현재 동국대학교 컴퓨터공학과 교수. 관심분야는 운영체계론, 컴퓨터 성능평가, 소프트웨어 공학 등임.