분산 시스템 관리를 위한 에이전트-온-디맨드 방법의 성능 평가

설승진*, 김태성, 이금석
동국대학교 컴퓨터공학과
e-mail: {ssj, ruthere, kslee}@dgu.ac.kr

A Performance Evaluation of the Agent-On-Demand for Distributed System Management

Seung-Jin Sul*, Tae-Sung Kim, Keum-Suk Lee
Dept. of Computer Engineering, Dongguk University

요 약

SNMP나 CMIP에 바탕을 둔 클라이언트/서버 방식의 분산 시스템 관리 환경은 확장성, 상호운영성, 유연성 등과 관련하여 많은 제한점을 드러내고 있다. 따라서 이러한 단점들을 극복하기 위해 이동 에이전트(mobile agent)를 적용하려는 노력이 진행되고 있지만 이동 에이전트 기법을 시스템 관리에 적용하여 얻을 수 있는 성능 향상에 대해서는 면밀한 성능 분석이 필요하다. 제안한 에이전트-온-디맨드 방법에서는 에이전트를 기능별 업그레이드가 가능하도록 구성하고 관리 대상 노드가 필요한 에이전트를 요청할 수 있도록 하였다. 또한, 이동 에이전트를 위한 적절한 성능 모델을 수립하여 분석적 방법으로 성능을 평가하였다[10].

본 논문에서는 분석적 방법을 통한 성능 모델을 기반으로 제안한 에이전트-온-디맨드 방식에 대한 성능 평가를 모의 실험을 통해 분석하였다. AOD를 적용한 분산 시스템 관리 기법과 기존의 방법의 성능 평가를 위해 IBM Aglets 소프트웨어 개발 키트 (ASDK) 1.1b2 버전과 JDK 1.1.7b 버전을 이용하여 프로토타입 관리 환경을 구현하였다.

1. 서론

분산 시스템은 효율적인 자원 공유, 확장성 및 작업 부하의 분산 등과 같은 많은 장점을 제공하기 때문에 다양한 분야에 적용되고 있다. 하지만 분산 시스템의 규모가 방대해지고 다양한 서비스가 제공됨에 따라 예상하지 못한 성능 저하나 자원 할당의 비효율성, 보안 문제 등 심각한 문제점들이 노출되고 있다. 이러한 문제들을 해결하기 위해 효율적인 분산 시스템 관리 방안에 관한 연구가 활발히 진행되고 있다[1].

그러나 SNMP(Simple Network Management Protocol)나 CMIP(Common Management Information Protocol)를 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].

중앙집중형 시스템 관리의 문제점을 해결하기 위해 관리자 응용의 관리 기능을 관리 대상 노드에게 위임(delegation)하려는 연구와 함께 이동 에이전트 개념을 시스템 관리에 적용하기 위한 노력이 이루어지고 있으며, 주로 네트워크 관리 분야에서 많은 성

과를 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].

하지만 분산 시스템 관리를 위한 이동 에이전트는 각종 정책과 기능들을 포함하기 때문에 그 크기가 상당히 커지며, 기능 면에서 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 [10]에서는 계층적 상태 임계값(Hierarchical State Threshold; 이하 HST)이라는 자료 구조를 이용하는 에이전트-온-디맨드(Agent-On-Demand; 이하 AOD) 방법을 제안하였다. AOD에서는 관리 작업을 수행하는 대규모 이동 에이전트를 모든 관리 대상 노드로 파견하는 방법이 아니라 관리 대상 노드가 필요한 에이전트의 파견을 요청하도록 하고, 관리자 응용은 요청된 에이전트, 즉 해당 관리 대상 노드를 관리하기 위해 최소한으로 필요한 기능만을 갖는 에이전트를 여행 계획(travel plan)을 사용하여 파견한다. HST는 관리 대상 노드가 파견된 에이전트의 업그레이드를 요청하는 기준을 제공한다.

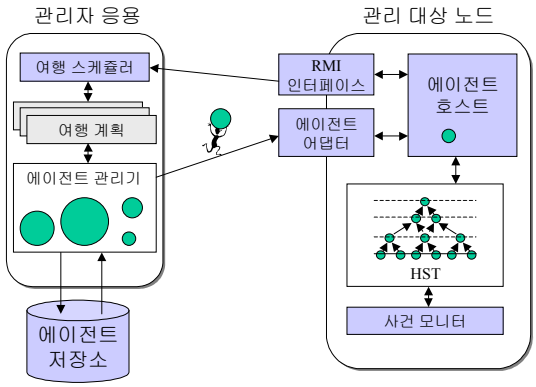
본 논문의 구성은 2장에서 에이전트-온-디맨드 방법에 대해 설명하며, 3장에서 성능 모델에 대해 소개한다. 4장에서는 구현 환경, 5장에서 실험 결과를 제시하며, 마지막으로 6장에서 결론 및 향후 연구에 대해 기술한다.

2. 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 가정에 기반을 두고 있다. 관리 대상 노드는 관리자 응용에게 더 많은, 혹은 더 적은 기능을 가진 에이전트의 파견을 요청하기 위해 HST를 생성 및 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)라 한다.

2.1 AOD 시스템의 구성

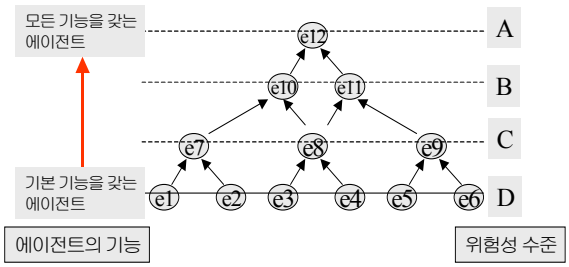
(그림 1)은 AOD 시스템의 구성을 나타낸 것으로 관리 시스템 가운데 HST 유지 및 에이전트 파견과 관련된 부분만을 나타낸다. 에이전트 관리기는 시스템 관리자에 의해 작성된 관리 에이전트의 저장 및 파견을 담당한다. 여행 스케줄러는 관리 대상 노드의 요청들을 전달받으며, 이를 바탕으로 각 에이전트에 대한 여행 계획을 조정한다. 에이전트 호스트는 에이전트 관리기를 통해 파견된 에이전트를 전송받은 후, 실행을 위해 초기화하고, 사건 모니터가 수집한 관리 정보와 HST를 바탕으로 에이전트 요청을 결정한다. 에이전트 호스트는 자바 RMI 인터페이스를 통해 관리자 응용의 여행 스케줄러와 통신함으로써 에이전트를 요청한다.



(그림 1) 에이전트-온-디맨드 시스템의 구성

2.2 계층적 상태 임계값

관리 대상 노드가 유지하는 HST는 시스템 관리자에 의해 작성된 관리 정책(management policy)을 바탕으로 구성된다. 관리 정책은 SGML을 이용하여 기술될 수 있으며[6], 관리 대상 노드의 상태 정보는 위험성 수준에 따라 여러 등급으로 구성될 수 있다. 에이전트 호스트는 관리 정책을 바탕으로 계층 구조로 구성되는 HST[9]와 관리 정보를 참조하며, 정의된 사건의 발생 여부에 의해 상위 위험성 등급이나 하위 위험성 등급으로 조정될 수 있다. (그림 2)는 HST와 이동 에이전트의 기능과 위험성 등급간의 관계에 대한 예를 나타낸다.



(그림 2) HST와 위험성 수준

2.3 여행 계획

AOD에서 여행 계획은 에이전트 종류별로 생성되며, 시작 노드와 방문 노드로 이루어진 튜플의 집합이다. 여행 계획의 초기값은 가장 최소의 기능을 갖는 기본 이동 에이전트가 모든 노드를 방문하도록 설정된다. 또한, 모든 이동 에이전트는 반드시 관리자 응용으로 복귀한다고 가정한다. 예를 들어, 분산 시스템이 관리자 응용 m과 관리 대상 노드 a1, a2, a3, a4, a5로 구성되었다면 일대일로 방문하는 경우의 여행 계획은 {(m, a1), (m, a2), (m, a3), (m, a4), (m, a5)}로 표현되며, 모든 관리 대상 노드를 링 형태로 방문할 경우의 여행 계획은 {(m, a1, a2, a3, a4, a5)}로 표현된다. 제안한 AOD에서 이동 에이전트의 모든 이동 형태는 여행 계획을 통해 표현할 수 있으며, 기본적인 이동 형식은 링 방식의 순환으로 가정한다.

3. 성능 모델

3.1 이동 에이전트의 성능 모델

이동 에이전트는 코드, 데이터, 상태 정보로 구성되므로 $B_{AGENT} = (B_{code}, B_{data}, B_{state})$로 표현할 수 있다[8]. 에이전트 B_{AGENT}가 노드 L_1에서 L_2로 이동하는 경우, 네트워크 부하는 다음과 같이 나타낼 수 있다.

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

그러므로 이동 에이전트의 실행 시간은 다음과 같다.

$$T_{MOBILE}(L_1,L_2,B_{AGENT})=\delta(L_1,L_2)+\frac{B_{MOBILE}(L_1,L_2,B_{AGENT})}{\tau(L_1,L_2)}+\begin{cases} 0, \text{ if } L_1=L_2 \\ 2\mu(B_{code}+B_{data}+B_{state}), \text{ else} \end{cases}$$

3.2 AOD를 적용한 경우의 성능 모델

분석을 간소화하기 위해 이동 에이전트의 이동 방식을 링 형태의 순환 방식으로 고정하며, 모두 n개의 노드로 이루어진 분산 시스템을 m번 순환한다고 가정한다. 또한 AOD에서 요청할 수 있는 이동 에이전트의 종류는 B_{A1}'와 B_{A2}' 등 2 가지로만 제한하며 에이전트의 업그레이드 경우만 고려하였다.

AOD를 적용할 경우 네트워크 부하와 실행 시간 모델은 다음과 같다.

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

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

여기서 S는 관리자 응용, D는 여행 계획에 기술된 이동 대상 노드를 의미한다. 한편 P는 에이전트 업그레이드 확률을 의미한다. 즉, 매 순환마다 n개의 노드 가운데 P*n개의 노드가 에이전트 업그레이드를 요청한다는 것을 의미한다.

전체 관리 대상 노드를 m번 순환할 경우, 네트워크 부하와 실행 시간 모델은 다음과 같다.

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

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

여기서 Q_{j-1}은 j-1번째 순환에서 업그레이드를 요청한 노드의 개수, P_{j-1}*n을 의미한다.

상기한 모델을 이용하여 AOD 방법의 성능을 평가한 결과, HST의 유지 오버헤드와 에이전트 요청을 처리하기 위한 오버헤드를 충분히 반영하여도 매 순환마다 전체 노드 가운데 업그레이드를 요청하는 노드의 수가 30% 미만일 경우에는 기존 방법에 비해 성능이 개선됨을 알 수 있었다[9][10].

4. 실험 환경

에이전트를 이용한 관리 환경을 모의실험하기 위한 이동 에이전트 개발환경으로는 IBM사의 Aglets 소프트웨어 개발 키트(Aglets Software Development

Kit; ASDK)를 사용하였다. ASDK는 자바 언어를 기반으로 편리하게 이동 에이전트 소프트웨어를 개발할 수 있도록 지원해준다[11]. 자바 컴파일러와 JVM은 JDK 1.1.8 버전을 사용하였는데, 이는 ASDK가 아직 JDK 1.2 버전을 지원하지 않기 때문이다.

모두 10대의 PC를 10Mbps 이더넷을 기반으로 랜으로 구성하고 운영체제로는 윈도우NT를 사용하였다.

(그림3)과 (그림4)는 실험을 위해 구현된 관리자 응용과 이동 에이전트 코드의 일부를 나타낸다.

```
1: package aod;

2: import com.ibm.aglet.*;
3: import com.ibm.aglet.event.*;
4: import com.ibm.aglet.util.*;
5: import com.ibm.agletx.patterns.*;
6: import com.ibm.agletx.util.*;

7: public class Manager extends Aglet {

    /* overrided methods*/
8:   public void onCreation(Object o){ }
9:   public boolean handleMessage(Message msg){ }
10:  private void go() { }
11:  public void dispose() { }

    /* AOD methods */
12:  private void createGUI(){ }
13:  public void createSlave(){ }

     /* 여행 계획 관련 메쏘드 */
14:  private void addTP(URL url){ }
15:  private void removeTP(int I){ }
16:  private Vector getTPList(){ }

  }
```

(그림 3) AOD를 위한 Manager 소스코드

(그림3)에서 2~6행은 IBM ASDK에서 지원하는 패키지를 사용함을 의미하며, 8~11행은 각각 관리자 응용의 초기화, 메시지 관리, 이동 에이전트의 파견과 제거를 위한 메소드이다. 14~16행은 여행계획을 관리하는 메소드이다.

```
1: package aod;

2: import com.ibm.aglet.*;
3: import com.ibm.agletx.util.*;
4: import com.ibm.aglet.event.*;

5: public class Agent extends Aglet {

6:    public void onCreation(java.lang.Object init) { }
7:    public void run() {
        ...
        /* 에이전트의 작업 설정 */
8:      itin.setTask(new TaskB(mp));
    }
  }
```

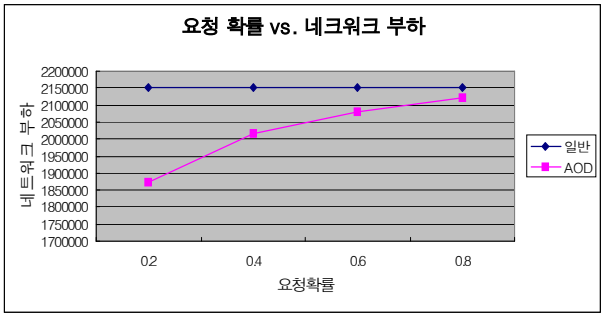
(그림4) AOD를 위한 이동에이전트 소스코드

(그림4)는 AOD의 이동 에이전트를 위한 코드의 부분이며, 8행은 에이전트가 특정 노드에 도착하여 수행하는 작업을 설정하는 부분이다.

5. 실험 결과

AOD 방법과 일반적인 방법의 비교는 네트워크를 통해 전송되는 데이터양을 의미하는 네트워크 부하를 기준으로 하였다. 실험은 모두 6개의 관리 대상 노드를 대상으로 이동 에이전트를 100번 순환하는 과정을 반복하여 평균치를 측정하였다. 현재 적절한 관리 응용을 적용하지 못한 상태이기 때문에 관리 대상 노드에 도착한 에이전트는 관리 작업에 상응하는 연산을 수행하도록 하였으며, AOD 경우에는 HST 자료구조를 유지하고 에이전트를 요청하는 오버헤드를 추가하였다.

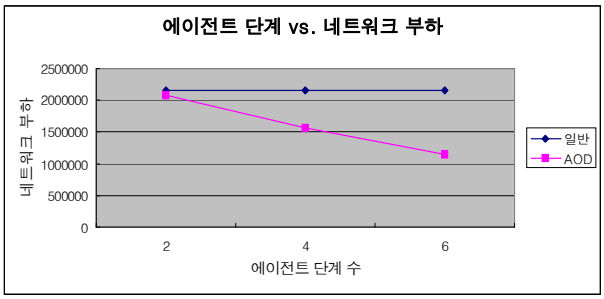
(그림5)는 전체 노드 가운데 에이전트를 요청하는 확률, P와 네트워크 부하 사이의 관계를 나타낸다.



(그림 5) 요청 확률과 네트워크 부하의 관계

(그림5)의 결과는 분석적 방법에 의한 결과와 아주 흡사한 것으로써 실제 환경에서의 AOD 방법 역시 전체 노드 가운데 에이전트를 요청하는 노드의 비율이 줄어들수록 AOD를 사용하지 않는 경우보다 훨씬 성능이 개선됨을 알 수 있다.

(그림6)은 업그레이드되는 에이전트의 단계의 수와 네트워크 부하 사이의 관계를 나타낸다.

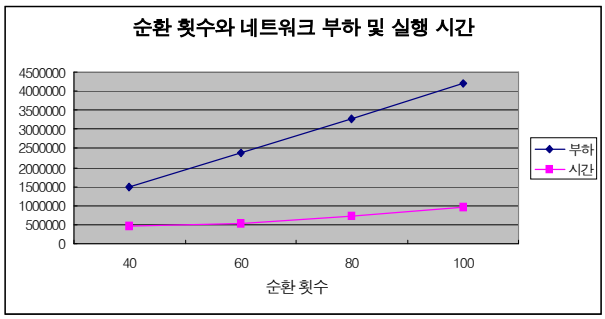


(그림 6) 에이전트 단계와 네트워크 부하

(그림6)은 AOD를 구성하는 에이전트 단계 종류의

개수가 1에 가까워질수록, 즉 하나의 에이전트를 모든 노드에 파견할 경우보다 에이전트 단계 종류의 개수를 증가시킬수록 네트워크 부하가 감소함을 나타내고 있다. 이는 AOD 방법의 HST 자료구조 유지 비용과 에이전트 요청 오버헤드를 충분히 감안하더라도 크기와 기능에 따라 에이전트를 단계적으로 구성하는 AOD 방법이 개선된 성능을 나타냄을 나타낸다.

(그림7)은 AOD 방법의 성능이 순환 횟수의 증가에 따라 선형적으로 증가함을 나타낸다. (그림7)에서 실행시간은 순환 횟수만큼 이동 에이전트를 이동시켰을 때 소요된 총 시간(단위:밀리초)을 의미하며, 노드수의 증가해도 급격한 증가를 보이지 않고 있다.



(그림 7) 순환 횟수와 네트워크 부하 및 실행 시간과의 관계

구현에 의한 다양한 실험 결과는 대부분 분석적 방법에 의한 실험 결과와 거의 일치했으며, 이는 분석적 방법에서의 가정 사항들이 크게 틀리지 않았음을 입증하는 것이라 말할 수 있다.

6. 결론 및 향후 연구

지리적으로 원거리에 위치한 자원을 바탕으로 구성되는 분산 시스템은 중앙집중형 시스템에 비해 다양한 장점들을 제공하는 반면, 효율적인 관리에 상당한 어려움을 갖고 있다. 이러한 분산 시스템을 관리하기 위한 기존의 방법들은 SNMP나 CMIP와 같은 프로토콜을 사용하여 관리 대상 노드들로부터 관리 정보를 수집하고 이를 종합적으로 분석해 관리 작업을 수행하기 때문에 네트워크 트래픽의 집중 및 관리 작업의 부하가 집중되는 치명적인 단점을 갖는다.

에이전트 기반 시스템 관리 기법은 이와 같은 단점을 해결할 수 있을 뿐만 아니라, 에이전트 기법의 유연성과 지능성을 이용하여 더욱 효율적인 관리가 가능하다고 알려져 있다.

본 논문에서는 에이전트 기반의 시스템 관리에 대한 진보된 기법인 에이전트-온-디맨드 방법의 분석적 성능 평가 방법을 소개하고 IBM Aglets 소프트웨어 개발 키트를 이용한 구현을 통해 AOD 성능 평가의 당위성을 입증하였다.

향후, 지금까지 구현된 AOD 관리 환경을 더욱 확장하여 실제 관리 응용에 적용하는 연구와 AOD 방법의 개선 방안에 대해 연구가 진행되어야 할 것이다.

[참고 문헌]

[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 *98봄 학술발표 논문집, 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.

[10] 설승진, 이금석, "에이전트-온-디맨드를 이용한 분산 시스템 관리," 한국정보과학회, 정보과학회논문지: 시스템 및 이론, 제27권, 제1호, 2000년 1월, pp.81-88.

[11] Aglets Software Development Kit Home, [http://www.trl.ibm.co.jp/aglets/]