유비쿼터스 시스템을 위한 실시간 모니터링 에이전트

권성현[O1], 이병훈[1], 김재훈[1], 조위덕[2]
[1]아주대학교 정보통신공학과
{ liebeym[O], componer, jaikim }@ajou.ac.kr
[2]아주대학교 유비쿼터스시스템 연구센터
chowd@ajou.ac.kr

Real-Time Monitoring Agent for Ubiquitous System

Sung-Hyun Kwon[O1], Byoung-Hoon Lee[1], Jai-Hoon Kim[1], We-Duke Cho[2]
[1]Graduate School of Information and Communication, Ajou University
[2]Center of excellence for Ubiquitous System, Ajou University

1. 서론

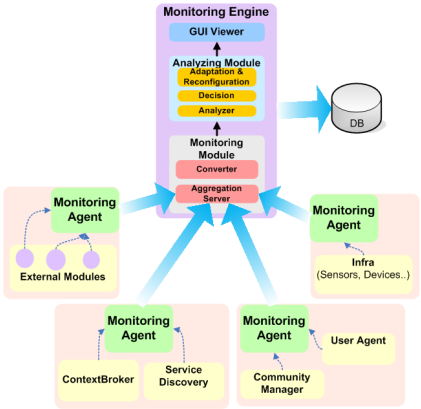
유비쿼터스 시스템은 전 세계적으로 차세대 동력 산업 또는 미래 산업으로 부각되고 있는 추세이다. 그리고 풍부한 정보통신 인프라가 갖추어져 있는 우리나라는 유비쿼터스 환경에 최적의 조건을 갖추고 있다. 또한 우리나라는 유비쿼터스 기술이 사람들에게 편리하고 쾌적한 생활을 할 수 있게 도와주는 미래 환경을 기축하는 것으로 인식하고 국가적인 차원에서 유비쿼터스 환경 개발진행을 하고 있다. 유비쿼터스 시스템은 주거환경을 개선하는 것뿐만 아니라 물류, 재난/환경, 정부 및 비즈니스 시스템등 많은 곳에 응용될 전망이다. 그런데 지금까지 편리한 유비쿼터스 시스템을 구성하기 위한 기술적인 구현 위주로만 연구를 했다. 그래서 유비쿼터스 시스템의 확장된 응용환경과 사용자에게 편리하고 안정적인 서비스를 제공하기 위해서 유비쿼터스적인 모니터링 시스템에 대한 연구가 필요하다. 기존의 모니터링기법으로는 협업 어플리케이션의 응답성과 신뢰성 보장, 실시간 모니터링, 유연한 모니터링 시스템을 지원하기에는 문제가 있다.

본 논문에서 제안하는 모니터링 에이전트는 위의 3가지를 지원하기 위해서 어플리케이션의 동작 상태를 메서드 단위로 성능 분석하고 변수 할당 상태, 자원상태를 실시간으로 모니터링해서 응답성과 신뢰성을 보장한다. 또한 모니터링 시스템이 어플리케이션을 모니터링하기위해 필요한 부분은 간단한 모니터링 API를 제공해 결합요소를 최소화했고 모니터링 시스템의 비동기적인 동작으로 유연한 모니터링을 지원한다.

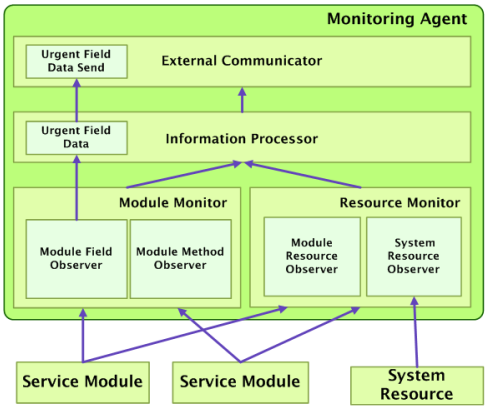
2. 본론

2.1 배경

<그림1>과같이 모니터링 엔진은 각 컴퓨터에서 수행되고 모니터링 에이전트로부터 서비스 모듈의 상황 정보를 받아서 원격에 있는 모니터링 서버에게 모니터링된 데이터를 보내준다. 모니터링 모듈이 각 전달 받은 데이터를 필터링하고 분석 모듈은 각 수집된 데이터를 가지고 서비스 모듈의 상황을 분석하고 판단한다. 분석한 결과를 사용자가 보기 쉽게 GUI로 화면에 나타낸다.



<그림1> 모니터링 시스템 구조도



<그림2> 모니터링 에이전트 구조도

2.2 구조도

어플리케이션을 개별 또는 다수를 모니터링하는 모니터링 에이전트는 어플리케이션의 추가 및 재거에 모니터링 엔진과 독립됨으로써 지속적인 모니터링이 가능하다. 그리고 어플리케이션과의 연관성을 최소화하기 위해서 어플리케이션과 코드 결합이 필요한 변수 모니터링, 어플리케이션 자원 모니터링은 간단한 API를 제공함으로써 코드 변경을 최소화 했다. 그리고 호스트 자원 분석 모니터링, 어플리케이션 메서드 호출 모니터링은 어플리케이션 외부에서 모니터링 할 수 있도록 해서 어플리케이션과 결합을 하지 않도록 했다. 모니터링 에이전트는 JAVA로 구현이 되었다.

* 본 연구는 21세기 프론티어 연구개발사업의 일환으로 추진되고 있는 지식경제부의 유비쿼터스컴퓨팅 및 네트워크원천기반기술 개발사업의 08B3-S2-10M 과제로 지원된 것임

그리고 JMX(Java Management eXtensions)[1]를 사용해 IPC(Inter Process Communication)기능과 데이터저장을 하고 서버와 연결한다. 그리고 JDI(Java Debug Interface)[2]를 사용해서 메서드 분석을 하고 JNI(Java Native Interface)[3]를 사용해 호스트 자원 정보를 획득한다. 모니터링 에이전트는 어플리케이션의 상태를 모니터링 하는 Module Monitor와 어플리케이션 자원, 호스트 자원을 감시하는 Resource Monitor로 구성 된다.

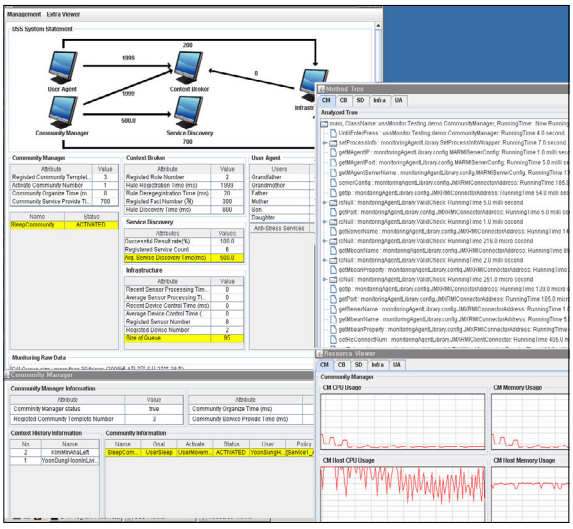
2.3 데 모 및 성능분석

<그림3>은 각 서비스 어플리케이션들이 구동이 되고 모니터링 에이전트에 의해 모니터링된 데이터를 모니터링 엔진에 전송했을 때 전체 어플리케이션들의 모니터링 상태를 간략히 보여준다. 그리고 모니터 그림의 On/Off로 모니터링 접속 유무를 판별해서 각 어플리케이션들의 실행 상태를 검사한다. 그리고 각 변수의 상태를 색깔별로 표시해서 각 서비스 어플리케이션의 현재 동작 상태를 판별 한다.

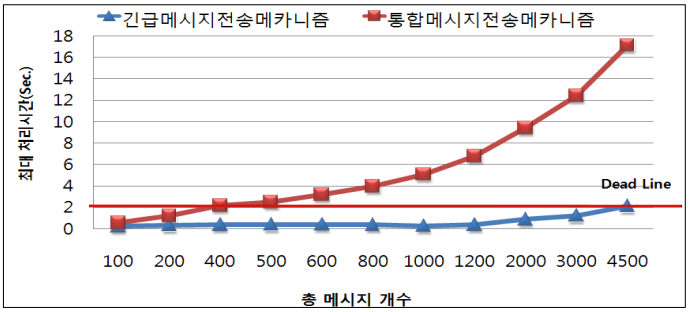
모니터그림을 클릭 했을 때 세부 변수들의 동작 상태를 모니터링하는 화면이 나타나고 각 어플리케이션의 실시간 데이터 변경상태를 파악 할 수 있다. 그리고 데이터 값이 성능 요구조건에 따라 색상별로 구분됨으로써 각 어플리케이션의 데이터 성능을 평가를 한다. 또한 각 어플리케이션의 자원 정보와 호스트의 자원정보 그리고 각 메서드 분기를 분석한 결과를 보여준다. 이정보로 어플리케이션의 실행 상태를 분석한다.

<표1> 시뮬레이션 조건

구분	내용
매시지 개수	어플리케이션 당 최대 100개/Sec. 생성
어플리케이션 개수	5개
긴급메시지 발생 빈도	총 메시지에서 10% (정규분포 생성)
데드라인	2초



<그림3> 모니터링 에이전트가 모니터링한 상황



<그림4> 메시지 발생량별 처리시간 비교

<그림4>은 긴급메시지전송메커니즘과 일반 메시지와 긴급메시지를 같은 레벨로 전송하는 통합메시지전송메커니즘의 최대 처리시간(Sec.)을 비교한다. 시뮬레이션 조건은 <표1>과 같다. 모니터링 메시지가가 도착했을 때 통합메시지전송메커니즘의 경우 는 순차적으로 메시지를 읽으면서 처리한다. 그래서 모니터링할 어플리케이션의 개수가 증가 되어 메시지가 약 450개 이상이 한 번에 모니터링 엔진에 들어 올 때 긴급 메시지에 대해 실시간 모니터링을 할 수 없다. 그러나 긴급메시지전송메커니즘을 사용할 경우 긴급메시지가 모니터링 되었을 때 긴급전송 데이터로 분류되어 전용 전송 클라이언트에 메시지를 전송하게 한다. 그래서 긴급메시지 발생 비율이 450개/Sec. 이상 전송되지 않는 한도 내에 긴급 메시지 모니터링 데이터는 데드라인을 만족한다.

3. 결론

유비쿼터스 커뮤니티 컴퓨팅 서비스[4], 상황인지서비스[5]등 최근 유비쿼터스 시스템을 구현하기 위해 많은 연구들이 진행되고 있다. 그러나 이들 서비스의 신뢰성, 응답성을 보장하는 모니터링 관련 연구 분야는 아직 부족하다. 그리고 기존 모니터링 방법론으로는 유비쿼터스적인 모니터링을 지원하기에는 역부족이다. 이러한 문제점을 해결하기 위해서 본 논문에서는 분산 환경에서 동작하는 유비쿼터스 모니터링 시스템을 지원하는 모니터링 에이전트를 제안했다. 분산 환경에서 실시간 동작하는 서비스 어플리케이션들이 안정적이고 응답성 있게 동작하도록 모니터링 에이전트는 최소한의 모니터링 코드로 어플리케이션을 효과적으로 모니터링 할 수 있게 했다. 그리고 모니터링 엔진과 독립된 동작으로 모니터링 할 서비스 어플리케이션의 재시작 또는 추가, 제거에 상관없이 지속적인 모니터링을 할 수 있도록 했다. 향후 네트워크 대역폭 측정, Module Method Observer의 분석 성능 향상 및 C, C++,VB등으로 개발된 어플리케이션을 지원하는 모니터링 에이전트를 연구 할 것이다.

4. 참고 문헌

[1]JMX 1.4 specification, http://java.sun.com/javase/6/docs/technotes/guides/jmx/index.html.
[2]JDI Specification, http://java.sun.com/javase/6/docs/jdk/api/jpda/jdi/index.html.
[3]JNI 6.0 Specification, http://java.sun.com/javase/6/docs/technotes/guides/jni/index.html.
[4]Y.N. Lee, J.T. Lee and M.K. Kim, "Multi-agent based Community Computing System Development with the Model Driven Architecture", *Autonomous Agents and Multi-Agent Systems,* May 2006.
[5]D.Zheng, et al., "Deployment of Context-Aware Component-Based Application Based on Middleware", Ubiquitous Intelligence and Computing 2007, pp.908-918, 2007.