서비스 데이터 수집기를 확장한 OGSA 기반 서비스 컴포넌트 구현

Implementation of an Extended Service Data Aggregator Service Component-based on OGSA

강윤희, 조광문, 강경우
천안대학교 정보통신학부

Yun-Hee Kang(yhkang@cheonan.ac.kr), Kwang-Moon Cho(ckmoon@cheonan.ac.kr),
Kyung-Woo Kang(kwkang@cheonan.ac.kr)

요약

이 논문에서는 OGSA 기반 그리드 서비스의 특징을 기술하고 서비스 데이터 정보 수집을 위한 서비스 컴포넌트를 기술한다. SOA를 위한 그리드 서비스의 구성을 위해서는 주요한 시스템 컴포넌트와 이들 간의 상호작용을 기술하는 시스템의 고수준의 소프트웨어 아키텍처로부터 SOA를 위한 그리드 서비스를 구성하기 위한 체계적인 접근이 고려되어야 한다. 이 논문의 목적은 서비스 데이터 수집기 서비스의 설계 및 구현이며 서비스 데이터 수집기 서비스는 인터넷과 같은 광대역 환경에서의 신뢰성 있는 수행을 위해 통지 메커니즘에 의해 비동기적으로 작동한다.

■ 중심어 : | 그리드 컴퓨팅 | 서비스 데이터 수집기 | OGSA | 서비스 컴포넌트 |

Abstract

This paper describes main characteristics of Grid Services based on OGSA and a service component for aggregating service data element. In order to build a Grid Service for SOA, it needs to consider a systematic approach from the high-level software architecture of a system that describes the main system components and their interactions. The purpose of this paper is to design and implement an extended service data aggregator service. To provide reliable aggregating service for service data elements, which is running under wide area environment like Internet, the aggregator service is operated asynchronously by notification mechanism.

■ keyword : | Grid Computing | Service Data Aggregator | OGSA | Service Component |

Ⅰ. 서 론

그리드 컴퓨팅(Grid Computing)은 방대하고 광대역의 자원 공유와 고성능 처리를 위한 분산 시스템의 설계, 구현 및 관리를 위한 기술이다[1-3]. 최근 OGSA(Open Grid Services Architecture기반의 GT3(Globus Toolkit Version 3)는 XML 스키마, SOAP(Simple Object Access Protocol), WSDL

접수번호 : #040918-001
접수일자 : 2004년 9월 18일
심사완료일 : 2004년 10월 14일
교신저자 : 강윤희, e-mail : yhkang@cheonan.ac.kr

(Web Services Description Language) 등의 XML 기반 기술을 사용하고 있으며, 인터넷을 통해 응용 및 컴퓨팅 자원을 공유하는 연구가 진행 중이다[4-6,11,12,15]. 그러나 GT3의 정보 서비스 시스템은 서비스 정보 유지 측면에서 단일 저장 스킴 및 비영속적(non-persistence) 저장 구조의 제약점을 갖는다. 기존의 GT3 정보 서비스 시스템의 두 가지 문제점을 해결하기 위해 서비스 및 자원 종류에 따른 분류 스킴과 영속 저장을 위해 Xindice XML 데이터베이스를 도입하였다. 이를 위해 OGSA 기반의 서비스 데이터 수집기(Service data aggregator)를 확장하였다.

분산 컴퓨팅 환경에서 시스템을 구성하는 요소들은 각각 다른 위치에 배치되어 비즈니스 로직과 데이터 처리를 수행한다[7-9]. 분산 컴퓨팅을 위한 객체 기술로는 자바 RMI, CORBA, DCOM 등이 있으며 이들은 공통적으로 원격지에 위치한 비즈니스 로직에 호출을 통해 접근하여 작업을 처리한다[10]. 웹 서비스(Web Services)는 기존의 CORBA, RMI, DCOM 등의 다른 분산 컴퓨팅 기술을 통합할 수 있는 새로운 클라이언트/서버 응용 개발 환경을 제공한다[5-6]. 그리드 컴퓨팅을 위한 미들웨어인 GT3의 개발 환경은 웹 서비스를 기반으로 하고 있다[4].

그리드 시스템의 사용자는 자신이 속한 VO(Virtual Organization) 내에서 서비스 및 계산 자원들의 정보를 쉽게 검색하고 자신이 필요한 서비스 및 계산 자원을 동적으로 사용하기 원한다. 이를 위해서는 VO 내의 가용한 서비스 및 계산 자원들에 관한 정보를 수집하고 정리해서 사용자 또는 다른 응용에 제공하여야 한다[5,6].

본 논문에서는 OGSA/OGSI 기반 그리드 서비스의 특징을 기술하고 기존의 분산객체 컴포넌트와의 비교를 통해 장단점을 분석한다. 또한 그리드 서비스인 서비스 데이터 수집기의 설계 기법 및 구현을 중심으로 기술한다. 구현된 서비스 데이터 수집기는 인터넷과 같은 광대역 환경에서의 신뢰성 있는 수행을 위해 통지 메커니즘에 의해 비동기적으로 작동한다.

본 논문의 제 2절에서는 관련연구를 기술하고 제 3절에서는 서비스 컴포넌트의 설계를 기술하고 제 4절에서는 서비스 컴포넌트인 데이터 수집 서비스의 구현, 마지막으로 제 5절에서는 결론 및 향후연구를 기술한다.

II. 관련 연구

본 절에서는 서비스 지향 아키텍처(Service Oriented Architecture, SOA) 측면에서의 웹 서비스의 특징 및 제약점을 설명하고 OGSA/OGSI 기반 환경의 특징을 기술한다.

1. 웹 서비스 개요

서비스는 인터넷을 통해 응용간의 동적인 연결을 위한 분산 컴포넌트(distributed component) 기술로서 복잡한 통합 시스템에 대한 의존도를 줄이는 대신 메시지 기반의 상호 연동을 통한 약 결합을 갖는 통합 시스템을 구성한다. 웹 서비스는 CORBA와 DCOM과 같은 분산 컴퓨팅 기술을 웹 기술에 차용하여 필요한 서비스를 웹에서 동적으로 찾은 후 서비스를 요청하고 그 결과를 사용하는 URI 기반의 접근 가능 소프트웨어 에이전트(software agent)를 총칭한다[9-12].

웹 서비스는 SOAP을 사용한 메시지 기반 표준 및 WSDL을 사용한 서비스 정의 기술 명세를 적용하므로 높은 상호운용성(interoperability)을 제공한다[11,12]. 웹 서비스에서 SOAP은 XML 포맷을 사용하여 웹 서비스의 메소드를 호출할 수 있는 기능을 제공한다. SOAP은 분산 환경에서 정보 교환에 사용하는 경량(lightweight)의 프로토콜로서 텍스트 기반의 XML을 프로토콜로 사용함으로써 하드웨어 플랫폼, 운영체제, 프로그래밍 언어, 네트워크 및 하드웨어 플랫폼에 종속적이지 않은 장점을 갖는다[12].

WSDL은 네트워크 상의 웹 서비스를 정의하기 위해 설계된 것으로 W3C에 의해 표준화가 주도되고 있다. WSDL은 웹 서비스가 제공하는 서비스에 대한 명세를 기술하기 위한 XML 문서로서 웹 서비스가 제공하는 기능, 접근을 위한 주소 및 호출 방법을 기술하기 위해 사용한다. WSDL은 메소드의 기술(description), 인자 유형(argument type) 및 리턴 값(return value)을 기

술하는 면에서 CORBA의 IDL과 비교할 수 있다[9,11]. 표 1은 CORBA와 웹 서비스의 대응되는 개념을 비교한 것이다.

표 1. CORBA와 웹 서비스 비교

개념	CORBA	웹 서비스
서비스 정의	IDL	WSDL
원격 객체 호출 프로토콜	IIOP	SOAP
명명 및 발견 메커니즘	ORB	UDDI/WSIL
메시지 형식	바이너리 메시지	텍스트 기반 메시지
상호 운용 수준	플랫폼 수준	메시지 수준

SOA 측면에서 웹 서비스는 시스템 간의 통합을 위해 HTTP, SMTP 등의 인터넷 기반 전송 프로토콜을 사용함으로써 편재 요구사항(pervasive requirement)을 만족시킬 수 있으며, XML 포맷을 기반으로 한 SOAP 프로토콜 및 확장 가능 메시지 포맷을 제공하므로 상호 운용성 및 개방형 기능 제공을 위한 확장성 요구사항(extensibility requirement)을 만족시키며, UDDI (Universal Description, Discovery and Integration), WSIL(Web Service Inspection Language) 등을 기반으로 한 레지스트리 메커니즘(registry mechanism)을 통해 배포 요구(deployment requirement)를 만족시킨다[7].

편재 요구사항 측면에서 웹 서비스는 인터넷 표준 프로토콜을 통한 메시지 전달 메커니즘을 채용하고 있으나 HTTP를 사용하는 경우 클라이언트 요청에 대한 상태정보가 유지되지 못하는 문제점으로 인해 트랜잭션 처리(transaction processing), 신뢰성 있는 메시지 전달(reliable message delivery)에 문제점을 가질 수 있다. 확장성 요구사항 측면에서 웹 서비스는 SOAP을 사용한 정보 전달을 제공하고 있으나 다양한 SOAP 형식에 따른 호환성의 문제, XML 기반 SOAP 메시지 파싱에 따른 지연, 비동기 메시지 기능 부재 등의 문제점을 갖는다. 마지막으로 배포 요구사항 측면에서 웹 서비스의 종류와 수가 증가됨에 따라 서비스 배치(service deployment)를 위한 동적 서비스 발견 메커니즘이 요구된다.

표 2. SOA 측면에서의 웹 서비스 장단점 비교

요구사항	장점	단점
편재	인터넷 기반 표준 프로토콜 사용에 따른 접근 용이성	트랜잭션 처리 메커니즘 보안 메커니즘
확장성	다양한 데이터 타입을 갖는 메시지	SOAP 메시지 처리 비동기 메시지 기능 상호협력 프로세스 처리
배포	UDDI 기반 중앙집중형 서비스 등록 메커니즘	동적 서비스 발견 메커니즘

표 2는 SOA 측면에서의 요구사항 별 웹 서비스의 장단점을 비교한 것으로 웹 서비스에 대한 관리를 위한 미들웨어에 대한 필요성 및 관리 기능 수행을 위한 도구가 요구되며, 특히 신뢰할 수 있는 응용의 개발을 위해서는 관리 기능에 대한 구현을 필수적으로 서비스 플랫폼, 서비스 배치 및 응용에 대한 관리가 필요함을 보인다.

2. OGSA/OGSI 구성

OGSA는 소프트웨어 아키텍처(software architecture) 모델로서 OGSI 표준의 참조 구현(reference implementation)으로 SOA의 일종인 웹 서비스를 기반으로 한다. OGSA는 동적으로 구성되는 VO 내에서의 협력적인 자원 공유와 지속적인 정보 서비스를 제공한다[13,14].

OGSA를 위한 프로그래밍 하부 구조의 표준인 OGSI는 위치 투명성(location transparency), 프로토콜 독립성(protocol independence) 및 SOA의 특징을 갖는다. OGSI는 그리드 컴퓨팅을 위한 프로그램 모델로서 임시 서비스 인스턴스(service instance) 생성, 상태 공개(status exposure), 생명주기(life cycle) 관리, 통지(notification), 등록(registration) 및 펙토리 패턴(factory pattern)을 제공한다. OGSA 기반 미들웨어인 GT3는 웹 서비스가 갖는 비상태유지 연결(stateless connection) 및 영속 서비스(persistent service)의 문제점을 개선한 그리드 운영환경을 제공한다[13,14]. 그리드 서비스 설계를 위한 WSDL은 연산

(operation)과 서비스 데이터 요소(Service Data Element, SDE)를 기술하기 위해 사용한다.

SDE는 그리드 서비스의 상태 정보를 유지하고 필요시 다른 그리드 서비스에 제공하기 위해 사용한다. 또한 SDE는 특징에 따라 분류되고 색인될 수 있다. 즉, 서비스 데이터는 그리드 서비스를 색인하기 위해 사용하며 서비스 데이터는 서비스의 특성 및 제공 정보에 따라 질의(query) 또는 통지(notification) 기법으로 사용한다.

SDE의 변화된 정보를 얻기 위한 방법은 pull 기반과 push 기반 수집으로 구분할 수 있다. Pull 기반 수집은 주기적으로 SDE의 변경을 요청하고, 요청을 받은 그리드 서비스는 요청된 정보를 제공하는 방법이다. 그러나 pull 기반 SDE 수집 방법은 호출간의 간격이 작은 경우 네트워크 트래픽과 CPU 사용을 증가시키고, 다수의 클라이언트 요청이 있는 경우 시스템의 부하 증가 및 성능 저하 문제를 발생시킨다[8]. Push 기반 방법은 SDE가 변경된 경우 비동기적으로 통지(notification)를 받는 방법으로 pull 기반의 성능 저하를 해결할 수 있다.

III. OGSA 기반 서비스 컴포넌트 설계

1. 서비스 컴포넌트 설계 개요

서비스 컴포넌트(service component)는 비즈니스 로직을 갖는 서비스 인터페이스로서 단일의 개념적인 모듈인, 모듈화되어진 서비스 기반 응용을 표현한다. 서비스 컴포넌트는 확장될 수 있고 특화되고 상속될 수 있으며 응용의 생성에 이용할 수 있다[10].

SOA를 위한 그리드 서비스의 구성을 위해서는 주요한 시스템 컴포넌트와 이들 간의 상호작용을 기술하는 시스템의 고수준의 소프트웨어 아키텍처로부터 SOA를 위한 그리드 서비스를 구성하기 위한 체계적인 접근이 고려되어야 한다. 웹 서비스 기반의 서비스 컴포넌트 개발은 상향식(bottom-up) 및 하향식(top-down) 방법으로 구분할 수 있다. 표 3은 서비스 컴포넌트의 개발 방법을 비교한 것으로 Java 인터페이스로부터 서비스 컴포넌트를 개발하는 상향식 설계 방법보다 하향식 설계가 높은 재사용 설계와 언어 중립성을 제공할 수 있음을 알 수 있다. 또한 하향식 설계에서 WSDL 설계 시에 사용되는 XML 스키마는 데이터 타입을 갖는 프로그래밍 언어 객체, DOM의 요소 집합 및 문자열 버퍼로 비순차화(de-serialize) 될 수 있다[13,14].

표 3. 서비스 설계 방법

	상향식 설계	하향식 설계
설계 개요	Java로 작성된 인터페이스를 작성 후 WSDL 인터페이스를 생성	WSDL portType 정의를 사용하여 인터페이스 제공 처리 기능
설계 기법	기존의 코드에 대한 re-factoring	WSDL 내의 바인딩과 서비스 부분은 도구를 통한 스텁 코드 및 타입 클래스로 자동 생성
설계 특징	특정 복합 자바 자료 형을 WSDL로 사상할 수 없음에 따른 낮은 상호운영성	PortType Interface 정의를 통한 서비스 내에서 사용되는 자료형, 메시지, 포트형을 포함하는 추상적인 정의를 제공

서비스 데이터 수집 그리드 서비스 설계는 WSDL 설계 후 해당 서비스를 구현하는 하향식 접근 방법을 통해 이루어진다. 이를 위해 WSDL의 확장 버전을 사용한다. WSDL 1.1은 portType의 상속 및 내재된 확장을 제공하지 못한 반면 WSDL 1.2는 OGSI 명세를 반영하여 설계됨으로써 상속 및 내재된 확장을 제공한다. OGSI에서는 그리드 서비스의 소프트상태 관리, 조사, 통지, 발견 및 전역 상태 명명 지원을 제공한다. 추가적으로 보안 하부구조 및 로깅 및 관리의 시스템 수준 서비스를 제공한다[13]. 또한 연산 제공자(operation provider)를 기반으로 한 위임 모델(delegation model)은 WSDL 연산들의 다양한 구현을 쉽게 플러그인 할 수 있는 설계를 가능하게 한다.

2. 서비스 인터페이스 정의

GT3에서는 GWSDL을 사용하여 서비스 정의 구성 요소를 기술한다. 그림 1은 설계된 서비스 데이터 수집기의 스키마 파일인 service_data_aggregator_port_type.gwsdl 내에 포함되는 portType 태그를 보인 것

이다. portType 태그는 서비스 접근점에서 지원하는 연산(operation)의 집합의 추상화된 정의로서 서비스에 의해 공개된 메소드의 집합이다. portType 태그의 name 속성은 객체인 ServicecDataAggregatorPortType 을 표현한다. 해당 객체의 연산은 서비스 구현 시에 Java 클래스 내의 메소드로 구현되게 된다. 연산 addSubscription의 입력은 AddSubscriptionInputMessage() 를 사용하고, 출력은 AddSubscriptionOutputMessage() 를 통해 얻어진다. 서비스 데이터 수집기의 각 메시지는 해당되는 데이터 타입을 가질 수 있다. 메시지의 name 속성 값 AddSubscritionInputMessage는 연산 AddSubscription 의 입력 메시지tns:AddSubscriptionInputMessage를 정의한 것으로 각 message 태그는 서비스 호출 파라미터를 나타낸다.

```
〈gwsdl:portType
  name="ServiceDataAggregatorPortType"
 extends="ogsi:NotificationSink"〉
 〈operation name="addSubscription"〉
〈input
    message="tns:AddSubscriptionInputMessage"/〉
 〈output
    message="tns:AddSubscriptionOutputMessage"/〉
 〈/operation〉
 〈operation name="removeSubscription"〉
〈input
    message="tns:RemoveSubscriptionInputMessage"/〉
 〈/operation〉
〈/gwsdl:portType〉
```

그림 1. 서비스 데이터 수집기 스키마 파일

IV. 서비스 데이터 수집기 서비스 구현

본 절에서는 OGSA 기반 서비스 구현을 예로 보이기 위해 서비스 데이터 수집기 서비스 구현을 기술한다.

1. 그리드 서비스 구현 환경 및 도구

본 서비스 데이터 수집기 서비스는 GT3 기본 서비스인 Service Data Aggregator를 확장하여 통합 정보 서비스 시스템을 구축하였다. 또한 수집된 서비스 데이터는 XML 전용 데이터베이스 관리 시스템인 Xindice를 이용하여 저장한 후 정보 접근이 가능하도록 데이터베이스 설계하고 구축하였다. 본 시스템의 구현환경으로는 Java 기반의 호스팅 환경을 제공하는 글로버스 미들웨어인 GT3.0.2를 사용하여 Linux 및 Windows XP에서 구동하도록 하였다.

OGSA/OGSI 기반의 그리드 서비스 생성을 위해서는 GSDL2Java, GenerateBinding 및 GWSDL2WSDL를 사용한다. 표 4는 그리드 서비스 생성 도구의 기능을 보인 것이다.

표 4. 그리드 서비스 생성 도구

도구 명	기능
GSDL2Java	스텁(stub), 바인딩(binding), 서비스 인터페이스 및 서버 구현 스켈레톤(skeleton)을 위해 GWSDL 파일 및 Java 프로그램을 자동 생성
GenerateBinding	WSDL 내의 추상 portType 표현을 위한 바인딩 정보를 자동 생성
GWSDL2WSDL	자동 생성되어진 GWSDL 파일을 표준 WSDL파일로 변환

GWSDL2WSDL 도구는 서비스의 호출을 위해 사용되는 전송 프로토콜과 요청 방식을 기술하는 binding 정보, 실제 구현과 관련된 연결 포트(port)를 요소로 갖는 서비스 및 외부에서의 인터페이스를 위한 portType에 관련된 파일을 생성한다. WSDL 1.2의 확장 기능인 extends는 GWSDL2WSDL 도구에 의해 WSDL 1.1 형식으로 변환된다. 다음은 GWSDL2WSDL 도구를 사용하여 3.2 절의 GWSDL 파일을 자동 생성한 후의 생성된 WSDL 파일이다.

- service_data_aggregator_bindings.wsdl
- service_data_aggregator_service.wsdl
- service_data_aggregator_port_type.wsdl

그림 2는 GWSDL2WSDL 도구를 사용하여 생성된 WSDL 파일인 service_data_aggregator_port_type.wsdl 일부를 보인 것으로 OGSI에 의해 통지를 받기 위한 NotificationSink 인터페이스 내의 연산인 deliverNotification이 정의된다. 정의된 연산 deliverNotification은 portType인 ServiceDataAggregatorPortType 내에 포함된다.

```
〈portType
   name="ServiceDataAggregatorPortType"〉
 〈operation name="deliverNotification"〉
〈input
   message="ns3:DeliverNotificationInputMessage"
xmlns:ns3=" OGSI 해당 URI"/〉
 〈/operation〉
 〈operation name="addSubscription"〉
〈input
   message= "ns0:AddSubscriptionInputMessage"
   xmlns:ns0=" service_data_aggregator 해당 URI"/〉
〈output
   message="ns1:AddSubscriptionOutputMessage"
     xmlns:ns1=" service_data_aggregator 해당 URI"〉
 〈/operation〉
   ...
〈/portType〉
```

그림 2. 생성된 WSDL 파일

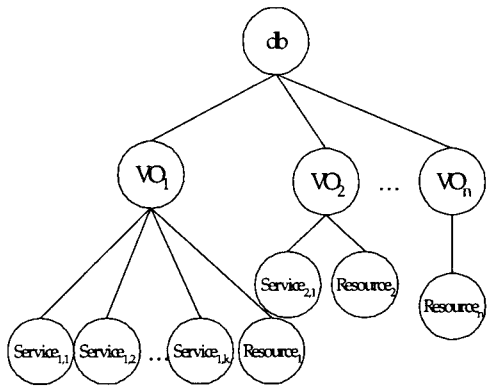

그림 3. 다중 컬렉션을 갖는 Xindice DB 스키마

2. ServiceDataAggregator 서비스 구현

기존의 GT3.X에서는 Index 서비스에서 XindiceServiceDataSet 클래스를 초기화하여 하나의 컨테이너(container) 인스턴스를 만들고 이것을 이용하여 데이터를 저장하는 방식을 사용하였으나 본 논문에서는 서비스 데이터의 종류별로 서로 다른 컬렉션을 생성하고 구분하여 저장함으로써 데이터베이스의 활용 효율을 높이고자 한다. 따라서 하나의 컨테이너 외에 다른 컨테이너를 생성할 수가 없어서 근본적으로 다중 컬렉션을 지원할 수 없는 구조를 갖는다. 구현한 다중 컬렉션 구조는 VO에 기반으로 구성된다. GT3.X에서 하나의 컬렉션만을 생성할 수 있는 구조를 변경하기 위하여 본 논문에서는 Xindice 데이터베이스에 정보를 저장하기 위한 메소드 호출이 발생하는 시점에 컬렉션을 위한 XindiceServiceDataSet 컨테이너를 VO 이름에 기반을 두어 생성한다. VO 이름과 더불어 각 서비스 이름도 컨테이너 생성에 사용된다. 이미 존재하고 있는 VO 이름이나 서비스 이름인 경우에 해당 컨테이너가 생성되지 않고 그대로 사용된다. 그림 3은 다중 컬렉션의 계층 구조를 보여준다.

전체적인 서비스 데이터 수집 서비스는 IndexSvc, ServiceDataAggregator, XindiceSvc와 RIPS의 4가지 모듈로 구성된다. RIPS는 자원에 대한 상태 정보의 모니터링을 수행하며 observerable로서의 기능을 수행한다. IndexSvc는 서비스 데이터 전달을 위한 접근점으로서 RIPS의 자원 상태 변경을 통보 받는 observer의 기능을 수행하고 XindiceSvc는 데이터의 영속 저장을 수행한다. IndexSvc는 사용자 및 다른 서비스의 전반부(front-end)로서 작동한다. 본 절에서는 ServiceData 수집 기능을 수행하는 ServiceDataAggregator를 중심으로 수집 기능의 구현을 기술한다.

RIPS 모니터는 주기적으로 시스템의 자원 정보를 모니터링하다. 자원 정보의 예인 시스템 부하는 OS 스케쥴러 내의 준비 큐의 길이로서 정의될 수 있다. RIPS 서비스는 1분, 5분 및 15 분 동안의 시스템 로드를 얻기 후 Cluster SDE에 정보를 유지하는 portType을 갖는다. 상태 수집기는 Cluster SDE를 구독하여 비동기적으로 통지를 수신한다.

서비스 데이터 수집 서비스의 GUI는 수집하고자 하는 SDE에 대한 구독을 위해 사용한다. 이를 위해 버튼과 사용자 입력을 처리하기 위한 리스너를 사용한다. 사용자가 구독을 위한 SDE와 소스를 입력한 후 구독을 위한 리스너를 생성한다. 이를 위해 SDESubscribe가 사용된다. 구독이 성공한 후에는 반환 값인 sink의 URL을 검사하여 구독이 성공하였음을 알 수 있다.

그림 4는 가시화되어진 시스템의 부하 정보를 모니터링하는 ServiceDataAggregator 서비스의 처리 내용을 보인 것이다. 구독의 sink 에서는 NotificationSinkCallback 인터페이스와 ListenerBasePortType 인터페이스를 구현하는 형태를 갖으며, NotificationSinkManager의 객체를 사용하여 source로부터의 자료 통지를 수신하기 위한 자신의 Listener 쓰레드를 구동시킨다. deliverNotification 메소드는 구독에 대한 메시지를 전달 받은 후에 구동되는

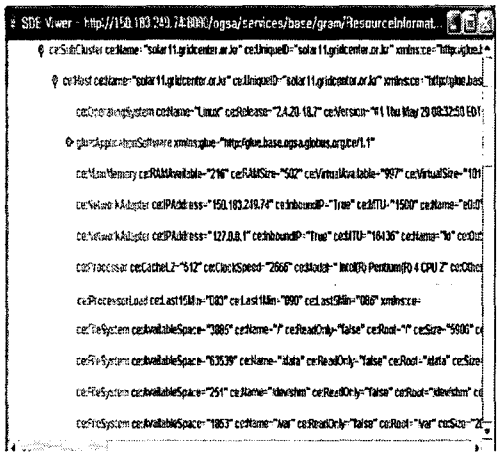

그림 4. 통지 후에 수집된 부하 정보

콜백(callback) 메소드로서 내부에서는 전달 받은 메시지에 대한 해석을 수행한다.

V. 결론 및 향후 연구

본 논문에서는 그리드 환경을 제공하기 위해 GT3 기반의 서비스 데이터 수집기 서비스를 기술하였다. 이를 위해 소프트웨어 아키텍처 측면에서 웹 서비스의 확장 기능을 제공하는 그리드 서비스의 기능을 기술하였다. 또한 확장된 그리드 서비스 설계 과정에서는 그리드 서비스 인터페이스 명세인 GWSDL 명세의 작성을 통해 주요한 시스템 컴포넌트와 이들 간의 상호작용을 기술하는 하향식 개발 방법을 사용하였다. 구현된 서비스 데이터 수집 서비스는 인터넷과 같은 광대역 환경에서의 신뢰성 있는 수행을 위해 통지 메커니즘에 의해 비동기적으로 작동하도록 구현하였다.

향후 연구로는 본 시스템을 확장하여 지속적인 서비스 수행을 위해 결함 포용을 갖는 OGSA 기반 그리드 서비스를 설계할 예정이다.

참 고 문 헌

[1] I. Foster and C. Kesselman, "The Grid: Blueprint for a New Computing Infrastructure," Morgan Kaufmann, San Fransisco, CA, 1999.

[2] I. Foster, C. Kesselman, "Globus: A Metacomputing Infrastructure Toolkit," Intl. J. Supercomputer Applications, Vol.11, No.2, pp. 115-128, 1997.

[3] S. Fitzgerald, I. Foster, C. Kesselman, G. Von Laszewski, W. Smith, and S. Tuecke, "A directory service for configuring high-performance distributedcomputings," In Proc. 6th IEEE Symp. On High Performance DistributedComputing, pp. 365-375, 1997.

[4] M. P. Papazoglou and D. Georgakopoulos, "Service-Oriented Computing," CACM, Vol.46, No.10, Oct. 2003.

[5] I. Foster, C. Kesselman, J. Nick, S. Tuecke, "Grid Services for Distributed System Integration," IEEE Computer, Vol.35, No.6, 2002.

[6] Gustavo Alonso, Fabio Casati, Harumi Kuno, Vijay Machiraju, "Web Services-Concepts, Architectures and Applications," Springer Verlag, 2004.

[7] P. Baglietto, M. Maresca, A. Parodi and N. Zingirian, "Deployment of Service Oriented Architecture for a Business Community," In Proc. of the 6th International Enterprise Distributed Object Computing (EDOC'02), 2002.

[8] A. Carzaniga, D. S. Rosenblum, and A.L. Wolf, "Design and Evaluation of a Wide-Area Event Notification Service," ACM Transactions on Computer Systems, Vol.19, No.3, pp. 332-383, Aug. 2001.

[9] http://www.w3.org/TR/2003/WD-ws-arch-20030808

[10] F. Curbera, et. al., "Unraveling the Web Services Web: An Introduction to SOAP, WSDL, and UDDI," IEEE Internet Computing, Vol.6, No.2, pp. 86-93, March/

April 2002.

[11] http://www.w3c.org/TR/wsdl

[12] http://www.w3c.org/TR/SOAP.

[13] T. Sandholm, S. Tuecke, J. Gawor, R. Seed, T. Maguire, J. Rofrano, S. Sylvester, and M. Williams, "Java OGSI hosting environment design a portable grid service container framework," Tech. Rep., Globus, 2002.

[14] T. Sandholm, R. Seed, and J. Gawor, "Globus toolkit 3 core-a grid service container framework," Tech. Rep., Globus, Jan. 2003.

[15] I. Foster, C. Kesselman, J. Nick, and S. Tuecke, "The physiology of the grid: An open grid services architecture for distributed systems integration," Technical report, Open Grid Service Infrastructure WG, Global Grid Forum, June 2002.

저 자 소 개

강 윤 희(Yun-Hee Kang) 정회원

- 1989년 2월 : 동국대학교 컴퓨팅공학과(공학사)
- 1991년 8월 : 동국대학교 컴퓨터공학과(공학석사)
- 2002년 8월 : 고려대학교 컴퓨터과학과(이학박사)
- 1991년 7월~1994년 4월 : 한국전자통신연구원(연구원)
- 1994년 4월~1997년 2월 : 한국문화예술진흥원, 전산개발부(선임연구원)
- 1997년 3월~2000년 2월 : (주)오롬정보 개발부(과장)
- 2000년 3월~현재 : 천안대학교 정보통신학부 조교수

<관심분야> : 그리드 컴퓨팅, 분산처리, 에이전트 시스템

조 광 문(Kwang-Moon Cho) 정회원

- 1988년 2월 : 고려대학교 컴퓨터학과 졸업(이학사)
- 1991년 8월 : 고려대학교 컴퓨터학과 졸업(이학석사)
- 1995년 8월 : 고려대학교 컴퓨터학과 졸업(이학박사)
- 1995년 9월~2000년 2월 : 삼성전자 통신연구소 선임연구원
- 2000년 3월~현재 : 천안대학교 정보통신학부 조교수

<관심분야> : 모바일 콘텐츠, 콘텐츠 유통, 전자상거래, 데이터베이스, 그리드컴퓨팅

강 경 우(Kyung-Woo Kang) 정회원

- 1990년 2월 : 경성대학교 전산통계학과 졸업(이학사)
- 1992년 2월 : 한국과학기술원 전산학과 졸업(공학석사)
- 1998년 2월 : 한국과학기술원 전산학과 졸업(공학박사)
- 1998년 3월~2000년 2월 : 한국과학기술정보연구원 슈퍼컴퓨팅센터 선임연구원
- 2000년 3월~현재 : 천안대학교 정보통신학부 조교수

<관심분야> : 그리드 컴퓨팅, 유비쿼터스 컴퓨팅, 컴파일러