Grid 서비스를 이용한 Jini 룩업서비스의 이동성

정승환° 이태동 정혜선 유승훈 최기영 정창성
고려대학교 전자컴퓨터공학과
{seung0120°, lyadlove, sepia5706, friendyu, twoxx195}@snoopy.korea.ac.kr, csjeong@chalie.korea.ac.kr

Nomadic Jini LUS Using Grid Service

Seung-Hwan Jung, Tae-Dong Lee, Hye-Sun Jeong, Seung-Hun Yoo, Ki-Young Choi, Chang-Sung Jeong

요 약

Jini는 네트워크를 통해 사람이나, 기기, 프로그램이 특정 자원을 찾거나 사용하고자 할 때, 관리자의 개입 없이 유연하게 동작하게 하는 분산 미들웨어의 하나이다. 기존의 Jini 서비스는 Lookup Service 데몬(Httpd, Reggie, Rmid)이 실행되어 Discovery, Join, Lookup, Service Invocation의 과정을 통해 이루어지는데, 데몬 fault 발생 시 어떠한 해결점이 없다는 것이 문제점이다. 이에 본 논문에서는 Jini 서비스 데몬 손실로 인한 Jini 서비스의 Fault 문제를 해결하기 위해 Grid환경 하에서 그리드 서비스를 사용하여 Jini 서비스에 Fault Tolerance를 제공하는 시스템의 구조와 방법을 제시한다.

1. 서 론

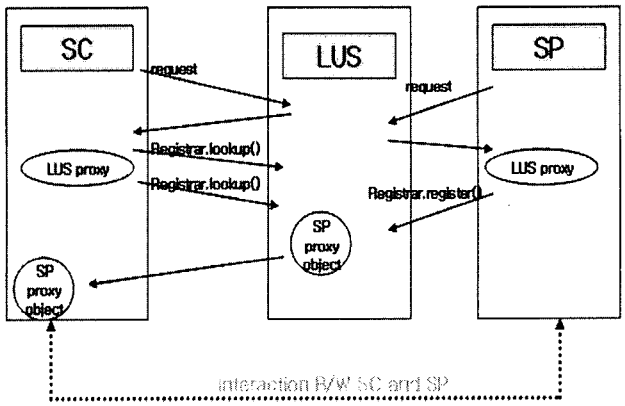
Jini는 네트워크상의 모든 종류의 장치와 소프트웨어 자원을 이용, 공유하기 위한 JAVA기반의 분산 미들웨어이다. 네트워크상에서의 'Plug and Work' 방식으로서 새로운 서비스가 다른 네트워크에 접속하면 바로 사용할 수 있게 되고 클라이언트는 이러한 서비스들을 찾아서 사용할 수 있게 된다. 이러한 Jini 서비스를 실행하기 위해서는 Lookup Service를 실행하여야 한다. 기존의 Jini 서비스는 Lookup Service를 사용하여 네트워크상의 서비스를 자동적으로 검색함으로써, 사용하고자하는 Service에 대한 Fault Tolerance 문제를 발견할 수 없다. 하지만, Lookup Service에 대해서는 Fault Tolerance를 보장하지 못한다. 본 논문은 Globus Toolkit에서 사용되는 GRAM, MDS, GridFTP 등의 서비스 환경 하에 Checkpoint Mechanism을 사용하여 Jini Lookup Service에 대한 이동성을 보장함으로써, Jini Lookup Service의 Failure에 대한 Fault Tolerance 해결책을 제시하였다.

2. Related Works

2.1. Jini

Jini는 가용한 서비스(Service Provider:이하 SP)를 동적으로 발견하고 특정장치(Service Client:이하 SC)가 원하는 서비스(SP)를 정확하게 매핑하는 메카니즘을 제시한다. [그림 1]과 같이 Jini 서비스를 동작하는 데는 크게 서비스를 제공하는 SP와 서비스를 이용하는 SC, 그리고 이러한 SP, SC의 서비스를 등록하는 Service Locator인 Lookup Service(이하 LUS)로 구성되어진다. 서비스 등록과정은 Jini의 SP의 프록시를 LUS에 등록함으로서, SC는 LUS로부터 SP의 프록시를 전송받아 SP와 RMI통신을 통해 서비스를 사용할 수 있다.

앞서 설명한 바와 같이, Jini는 서비스 서버를 찾는 *Discovery*과정, Jini 네트워크에 서비스를 등록하는 *Join*과정, 서비스 사용을 위한 SC의 *Lookup*과정, 사용자(SC)요구에 의한 서비스 수행과정인 *Service Invocation*으로 나눌 수 있다.



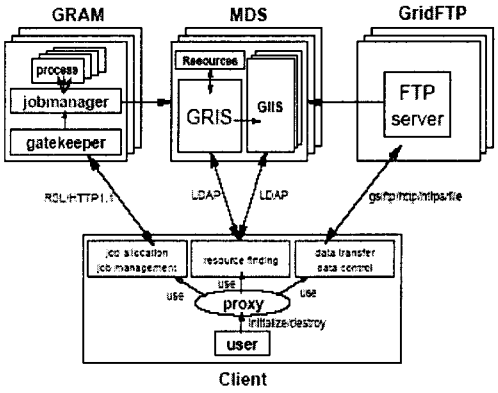
[그림 1] Jini 서비스 동작과정

다음은 이러한 Jini 서비스과정을 수행하기 위한 Jini의 LUS 데몬들이 어떠한 역할을 하는지 설명하였다.

-HTTP데몬: SP의 서비스 코드의 전송 데몬
-RMID 데몬: SP와 SC간의 RMI통신을 위한 데몬
-Reggie데몬: Jini의 실질적인 LUS부분, 서비스(SP)에 대한 Proxy를 저장

2.2. Globus

Grid Computing은 이 기종 간에 자원을 공유하여, 분산 처리를 효율적으로 하기 위한 환경을 제공하거나, 지리적으로 분산되어 있는 자료를 하나의 Data처럼 사용할 수 있는 환경을 제공한다. 이러한 Grid의 환경을 제공하는 미들웨어로서 Globus Toolkit이 있다. Globus는 Grid에서 필요로 하는 다양한 서비스들을 위한 Tool과 API로 제공하고 있다. 다음은 Globus에서 제공하는 기능들이다.



[그림 2] Globus 구성요소

-MDS(Metacomputing Directory Service) : LDAP (Lightweight Directory Access Protocol)을 사용해서 자원의 위치와 자원에 대한 메타데이터를 제공한다.

-GRAM(Grid Resource Allocation Management) : Globus에서 자원의 할당과 관리를 담당한다. GRAM을 통해서 사용자는 원격지의 자원들을 사용할 수가 있다.

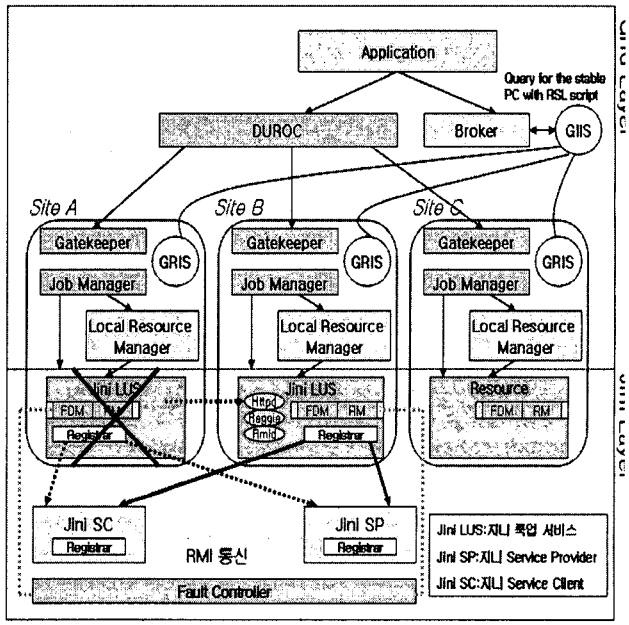
-GridFTP(Grid File Transfer Protocol) : 그리드 내의 데이터가 대규모 대용량이란 점을 고려하여 고속으로 파일 전송과 파일의 이어받기를 가능케 하는 요소이다.

-GSI(Grid Security Infrastructure) : GSI는 Globus의 보안을 담당한다. PKI와 SSL에 기반을 두고 Single sign-on 기능을 지원한다.

-GASS(Globus Access to Secondary Storage) : 원격지에 있는 파일의 처리나 데이터의 분산 저장을 담당한다.

3. Jini on Grid Architecture

앞에서 설명하였듯이, Jini 서비스를 실행하기 위해서는 LUS를 실행하여야 한다. 기존의 Jini 서비스는 LUS를 사용하여 네트워크상의 SP를 자동적으로 검색함으로써, SP의 failure에 대한 Fault 문제를 발견할 수는 없다. 본 논문에서는 Jini의 LUS의 등록과정을 Grid 서비스를 사용함으로써, 네트워크상의 최적의 Computer Resource에 LUS를 실행할 수 있다. 그리고 LUS의 Failure에 대한 LUS의 이동성을 보장함으로서, Fault Tolerance를 제공하였다. 기존의 Jini 서비스에서는 LUS에서 Failure가 발생하면 새로운 LUS를 다시 실행하여야만 한다. 하지만, 본 논문에서 제시하는 모델은 Grid 서비스를 사용하여 이러한 문제를 해결하였다. 다음은 Grid 환경에서 Jini 모델이 어떻게 구현 되었는지 설명하겠다.



[그림 3] Jini on Grid Architecture

3.1 Grid Layer

[그림 3]은 Jini에서 실행된 LUS가 Grid환경에서 다른 PC로 이동하는 과정이다. 예를 들어 *Site A*에 있는 PC1에서 엄청난 Overflow로 인해 PC가 멈추어 버렸다면, 이 PC에서 사용되었던 Jini의 룩업 서버는 멈출 것이다. 이때 본 시스템의 Application은 Broker에 RSL script를 넘겨주고, MDS의 정보를 query하여 *Site B*에 있는 최적의 PC를 찾을 수 있다. 그리하여 얻어진 MDS정보를 이용하여 LUS에 필요한 데몬(HTTPD, REGGIE, RMID)을 GridFTP를 사용하여 *Site B*로 옮겨질 수 있다. 옮겨진 LUS 데몬은 자원들을 할당하고 실행하는 DUROC을 통해 실행 요청되어 진다. 이 때, Broker는 주기적으로 MDS를 Query하고 최적의 리소스를 찾기 위해 RSL script를 사용하였다. 다음은 간략한 RSL script에 대한 예제이다.

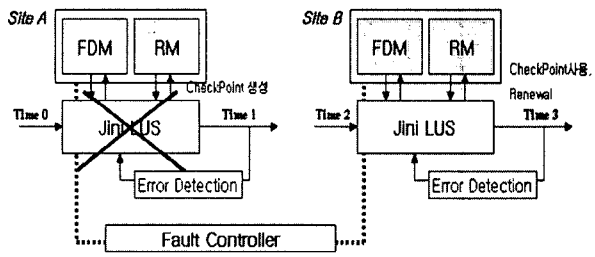
```
&(Mds-Os-name=Linux)(Mds-Cpumodel=Pentium*)(Mds-Cpu-speedMHz>=500)
```

3.2 Jini Layer

[그림 3]에서의 이동한 LUS는 새로운 환경의 네트워크에서 SC가 요구하는 SP에 대한 서비스들의 객체 프록시(registrar)을 획득하게 된다. 룩업서비스에 등록된 Jini SP의 서비스는 SC로 인해 재사용되어 질 수 있다. 마지막으로, SP에 대한 프록시 정보를 가지고 있는 SC는 SP와 RMI로 직접 통신함으로서, 사용자가 원하는 서비스를 사용할 수 있다.

4. Fault Tolerance Solution

[그림 3]은 Jini LUS에서 Failure가 발생시 LUS에 대한 데몬(Httpd,Reggie,Rmid)이동을 보여주었다. 이때 단순한 LUS 데몬 뿐만 아니라 Fault 문제 해결을 위한 몇 가지 Manager들을 이용하여 *site B*에 다시 실행 될 수 있다. [그림 4]는 LUS의 failure에 대한 Fault detection 을 위한 FDM(Fault Detection Manager)과 LUS의 새로운 환경 *site B*에서의 Recovery를 위한 RM(Recovery Manager)에 대한 Interaction과정이다. Fault Controller는 *site A*에서의 Checkpoint값을 *site B*로 넘겨준다.



[그림 4] Fault Tolerance Managers(FCM, BM,Fault Controller)

4.1 FDM(Fault Detection Manager)

FDM은 실행중인 Jini LUS의 Fault를 detect하는 Manager이다. FDM은 LUS 프로세스가 현재 로컬시스템에서 active한지를 체크하여 프로세스가 active하면 Failure는 발생하지 않는다. 그와 반대로 LUS 프로세스가 active하지 않다면 LUS는 Failure가 발생한다. 이렇게 LUS의 프로세스를 체크하여 Fault가 발생하였는지 체크하였다. FDM은 주기적으로 LUS에 대한 프로세스를 체크한다.

4.2 RM(Recovery Manager)

RM(Recovery Manager)는 Jini LUS의 현재 state대한 정보를 저장하기 위해 Checkpoint Mechanism을 사용하였다. Jini의 LUS에서 failure가 발생하면, FDM에 의해 Detect되고, BM에 의해 Jini LUS의 state대한 Checkpoint를 생성한다.

[그림 4]의 *Site A*에서 LUS의 Failure 발생 시 새로운 환경 *Site B*에서는 기존의 LUS state정보를 가진 Checkpoint로 인해 Jini 서비스를 계속하여 사용할 수 있도록 하였다. 이때 Checkpoint는 *Site A*에서 사용하였던 Jini SP의 서비스의 정보 등을 가지고 있다. 즉, 기존의 사용하였던 서비스의 Service ID, Service object proxy, Attribute들에 대한 정보를 포함하고 있어야 한다.

4.3 Fault Controller

Fault Controller는 [그림 4]와 같이 Fault가 발생한 Jini LUS를 체크하고, 생성된 Checkpoint Value를 MDS에 의해 Query된 최적의 리소스 *site B*로 넘겨준다. 그 후 site B의 RM은 Checkpoint에 의해 기존의 Jini 서비스를 Renew할 수 있다.

5. Conclusion

본 논문에서 제시하는 시스템은 Jini 서비스에 Grid를 사용하여 Jini가 제공하지 못하는 Fault Tolerance의 문제를 해결하는 방법을 제시하였다. 이를 위하여 MDS를 통한 자원의 Query와 Broker 알고리즘을 사용한 최적의 리소스에 Jini LUS실행을 요청, 등록함으로서, Jini Community안에 있는 모든 장치들의 서비스를 등록하고 사용할 수 있게 하였으며, 또한 Jini상의 Fault 문제를 Checkpoint Mechanism을 통해 해결하였다. 결과적으로, Jini LUS에 동적인 이동성을 확보함으로서, 기존의 Jini서비스의 LUS Failure 대한 Fault Tolerance보장함으로서 지니 서비스를 사용하는 Execution time등을 줄일 수 있는 동시에 자동적인 LUS의 등록, 실행 하는 시스템의 구조와 그 해결 방법을 제시하였다.

6. References

[1] Mark Baker, Garry Smith, "Jini meets the Grid.", Parallel Processing Workshops 2001

[2] Jim Waldo, "The JINI ARCHITECTURE for Network-Centric Computing"

[3] K. Czajkowski, S. Fitzgerald, I. Foster, "Grid Information Services for Distributed Resource Sharing"

[4] Johannes Luthi, Steffen Grobmann, "FT-RSS: A Flexible Framework for Fault Tolerance HLA Federations", ICCS 2004, LNCS 3038, pp865-872

[5] Jini Community, "http://www.jini.org"

[6] Globus Project, "http://www.globus.org"

[7] The Jini Network Technology Specifications, " http://sun.com/jini/specs/"

[8] Globus Project: Metacomputing and Discovering Service, "http://www-unix.globus.org/toolkit/mds/"