제 18 회 한국정보처리학회 추계학술발표대회 논문집 제 9 권 제 2 호 (2002.11)

효율적인 그리드 자원 정보 서비스를 위한 적응형 캐싱 기법

임민열, 박형우, 이상산
한국과학기술정보연구원 슈퍼컴퓨팅센터
e-mail : {mylim, hwpark, sslee} @hpcnet.ne.kr

An adaptive caching mechanism for efficient Grid Information Service

Minyeol Lim, Hyoungwoo Park, Sangsan Lee
Supercomputing center, Korea Institute of Science and Technology Information(KISTI)

요 약

초고속 네트워크에 연동된 고성능 컴퓨터 자원들을 효과적으로 활용하려는 그리드 컴퓨팅 연구는 네트워크를 통한 기존의 단순한 정보 공유보다 정보 및 자원의 공유를 더욱 더 밀접하게 결합시키는 것을 가능하게 한다. 이상적인 그리드 컴퓨팅 환경을 구축하기 위해서는 무엇보다도 그리드 환경에 속한 자원들의 정보를 통합적으로 관리 및 서비스하는 것이 무엇보다도 그리드 정보 서비스가 중요하다. 그리드 정보서비스의 역할은 그리드 사용자가 필요한 자원 및 그에 대한 정보를 쉽게 찾을 수 있도록 다양한 검색 기능을 제공하는 것 이외에도, 전세계에 분산된 자원들의 최신 정보를 빠르게 서비스하도록 하여야 한다. 이를 위해서는 그리드 환경의 특성을 고려하여 효율적인 그리드 정보 서비스 아키텍처를 구성하는 것이 필요하다. 본 논문에서는 그리드 미들웨어인 글로버스 툴킷(Globus Toolkit™)내에 포함된 그리드 정보 서비스를 기반으로 자원 정보의 적응형 캐싱 기법을 이용하여 그리드 정보 서비스 아키텍처를 재구성한다. 이를 통해, 자원 정보의 일관성을 향상시키면서 기존의 그리드 정보 서비스의 빠른 응답 속도를 제공할 수 있다.

1. 서론

최근의 인터넷 기반의 컴퓨팅 환경은 정보의 공유(Information sharing)을 가능하게 한다. 이를 이용하여 전자상거래와 같은 다양한 서비스가 생성되었다. 하지만, 단순히 정보의 공유를 통해서는 실제로 우리가 할 수 있는 다양한 일들을 실현할 수 없다. 그리드 컴퓨팅은 이 한계를 뛰어 넘는 것을 가능하게 한다. 즉, 앞에서 말했던 정보의 공유뿐만 아니라, 다양한 종류의 자원의 공유(Resource sharing)를 가능하게 하기 때문이다. 예를 들어, 어떤 사용자가 엄청난 컴퓨팅 자원을 필요로 한다면, 그리드 컴퓨팅환경은 전세계에 존재하는 컴퓨팅 자원을 그리드 컴퓨팅 환경을 통해 마치 자신이 슈퍼컴퓨터를 보유한 것처럼 이용할 수 있다.

위와 같은 그리드 컴퓨팅 환경을 구축하기 위해서는 기본적으로 분산 컴퓨팅 환경에 존재하는 자원들에 대한 정보를 효율적으로 통합하여 접근할 수 있도록 지원하는 것이 필수적이다. 이것을 일반적으로 그리드 정보서비스라고 말한다. 그리드 정보 서비스는 이기종의 다양하고 분산되어진 자원들에 대한 정보(Metadata)를 효율적으로 통합 관리하여, 그리드 사용자가 원하는 자원에 대한 정보를 효율적으로 서비스해야 한다. 그리드 사용자는 이를 통해 이질적이고 분산된 자원들을 자신이 직접 보유한 로컬 자원이고 또 하나의 자원인 것처럼 볼 수 있게 된다.

그리드 컴퓨팅의 전체 성능을 향상 시키기 위해서는 그리드 정보 서비스의 성능 향상이 필요하다. 왜냐하면, 그리드 사용자가 실제 필요한 자원을 사용하기 전에 그리드 정보 서비스를 통해 가용한 자원의 다양한 정보(예를 들어, 자원의 설정 및 성능)를 얻으려고 시도하기 때문이다. 따라서, 사용자에게 빠르고 항상

최상의 자원 정보를 제공하는 것이 그리드 정보 서비스의 주된 목표가 된다.

본 논문에서는 그리드 미들웨어의 시장 표준인 글로버스 툴킷(Globus Toolkit™)내에 포함된 그리드 정보 서비스 컴포넌트인 MDS(Monitoring and Discovery Services)를 기반으로 그리드 정보 서비스의 성능을 향상시키고자 한다. 이를 위해, 우선 현재 MDS 의 성능 향상 기법 중 하나인 캐쉥 기법을 개선시켜 그리드 자원 정보의 특성을 반영한 캐쉥 기법을 적용하도록 한다. 그 외에도 자원 인기도(Popularity)에 따라 자원 정보의 일관성(Consistency) 유지 수준을 변화시켜 그리드 정보 시스템의 성능을 개선시킨다.

본 논문의 구성은 우선 2 장에서 글로버스 툴킷의 MDS 에 의한 그리드 정보 서비스 구조를 살펴보고, 3 장에서는 이를 개선한 구조를 살펴본다. 4 장에서는 시뮬레이션을 통해 이들의 성능을 비교하고 분석한다. 그리고 마지막 장에서 정리 및 향후 발전 방향을 기술한다.

2. 글로버스 툴킷내 그리드 정보 서비스

현재 글로버스 툴킷 2.0 버전에 포함된 MDS2.1 버전은 그림 1 과 같은 서비스 구조를 가지고 있다. MDS 는 기본적으로 두 가지 서비스 즉, GRIS(Grid Resource Information Services)와 GIIS(Grid Index Information Services)로 나누어 진다.

- GRIS 는 특정 로컬 자원에 대한 메타데이타를 유지한다. 이는 다양한 정보 제공자(Information Provider)로부터 특정한 주기마다 메타데이타를 받아 유지한다.
- GIIS 는 GRIS 를 통해 제공되는 메타데이타를 계층적 구조로 모아 가상의 통합된 정보 서비스를 제공한다. 이를 통해 그리드 컴퓨팅의 가상의 조직(VO : Virtual Organization) 단위로 통합된 그리드 정보 서비스를 제공할 수 있다.

MDS 는 위의 두 가지 서비스를 통해 그리드 사용자가 원하는 자원의 메타데이타를 제공한다. 즉, 사용자가 GRIS 를 통해 자신이 알고 있는 특정 자원에 대한 메타데이타를 제공받을 수 있고, GIIS 를 통해 사용자가 필요한 자원에 대한 명세를 가지고 가용한 자원의 메타데이타를 얻을 수 있다.

GRIS 는 자신이 속한 자원에 대한 정보를 여러 개의 정보 제공자(Information Provider)들로 세분화하여 수집하게 된다. 예를 들어, CPU, Memory, Disk, Network, Job manager 와 같은 정보 제공자가 존재한다. 이 정보 제공자들는 미리 정해진 서로 다른 주기를 가지고 GRIS 에 의해 호출되며 담당하는 메타데이타를 리턴하게 된다. GRIS 는 이 메타데이타를 LDAP(Lightweight Directory Access Protocol) 서버에 저장한다. 또한, 미리 정해진 설정에 따라 자신이 속한 GIIS 서버에 주기적으로 등록 메시지(Registration Message)를 보낸다.

GIIS 는 분산된 자원들을 계층적 구조로 통합시킨다. 그림 1 에 보이는 것과 같이, GRIS 로부터 독립된 자원들의 메타데이타를 받아서 모으게 되며, 이들을 다시 상위의 GIIS 에 보내 통합하게 된다. 이를 이용하여 그리드 컴퓨팅을 위해 생성된 VO 단위로 자원들을 통합할 수 있다. GIIS 는 GRIS 와 마찬가지로 LDAP 서버를 이용하여 메타데이타를 유지하고 질의 서비스를 제공한다.

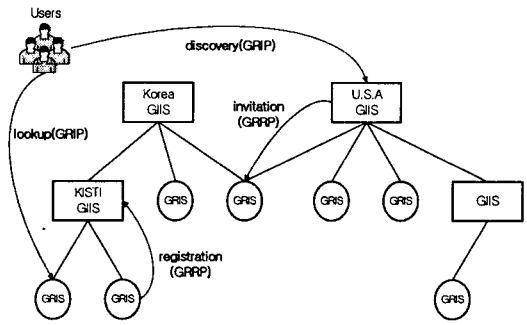

그림 1. 그리드 정보 서비스 구조

그림 2 는 GRIS 와 GIIS 가 자원 정보를 유지하기 위한 방법을 보여준다. MDS 가 시작되면 GRIS 와 GIIS 가 메타데이타를 유지하고 사용자의 질의를 처리하는 LDAP 서버가 동작한다. GRIS 는 별도의 프로세스로 동작하면서 설정에 따라 다양한 정보 제공자들을 각각의 주기에 맞게 호출한다. 이런 호출의 결과로 정보 제공자들은 각각 맡은 부분에 대한 메타데이타를 LDIF(LDAP Data Interchange Format) 형식으로 전달한다. 정보 제공자들이 제공하는 정보는 정적인 특성을 가지는 시스템 설정 정보(예를 들어, 하드웨어 및 소프트웨어 사양)와 동적인 특성을 가지는 모니터링 정보(예를 들어, 자원의 이용률)를 포함한다. GRIS 는 메타데이타를 정보 제공자로부터 수집하는 것 이외에, 자신이 속한 VO 에게 이 정보를 전달하기 위해 GIIS 에게 등록 메시지를 보낸다.

GIIS 는 자신이 포함하는 자원들에 대한 정보를 통합하기 위해서 GRIS 로부터 메타데이타를 가져온다. 이것은 이전에 GRIS 가 보내는 등록 메시지를 토대로 직접 GRIS 로부터 LDAP 검색 기능(Search Operation)을 이용하여 모든 정보를 가져오게 된다. GIIS 는 TTL(Time To Live)을 두어 GRIS 로부터 가져온 메타데이타의 일관성을 유지하게 되며, GIIS 의 사용자 질의에 대한 응답속도를 빠르게 하기 위해 GRIS 로부터 가져온 메타데이타를 캐쉥(Caching)한다.

지금 현재 MDS 의 캐쉥 기법은 매우 단순하다. GRIS 에서의 캐쉥 객체는 정보제공자에 의해 유지되는 정보 단위로 이루어 지며, GIIS 에서는 GRIS 에 의해 유지되는 전체 메타데이타가 하나의 캐쉥 객체 단위로 이루어진다. GIIS 는 GRIS 로부터 등록메세지를 받으면 GRIS 로부터 메타데이타를 가져와 캐쉥 객체로 복제하여 저장한다.

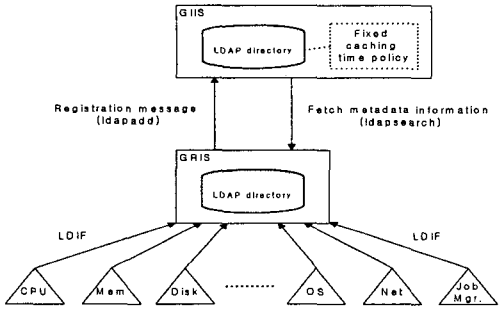

그림 2. MDS 내 자원정보의 유지 구조

이러한 캐쉉 객체들은 GRIS 에 의해 각각 생성되거나 GIIS 로 복제될 때 캐쉉 TTL 이 정해진다. 이 값은 MDS 의 설정을 통해 정적으로 고정되어지며, 캐쉉 객체의 유효기간을 결정하는데 사용되어 진다. 즉, 캐쉉 객체가 가지는 validfrom, validto, keepto 값에 따라 캐쉉 객체의 상태가 유효한지를 체크하게 된다. 이러한 캐쉉 객체는 응답속도를 향상시키기 위해 GRIS 와 GIIS 의 메모리에 유지된다. 사용자가 질의를 던지면 먼저 만족하는 객체를 모두 찾아낸 다음 그 결과를 가지고 유효성을 검사한다.

현재 MDS 의 캐쉉 기법은 앞에서 설명한 것과 같이 단순한 구조로 되어 있다. 이는 그리드 정보 서비스의 주된 성능 평가 요소인 서비스의 응답 속도와 데이터의 일관성측면에서 문제점을 드러낸다. 우선, 메타데이타의 갱신 주기가 미리 정해진 값에 따라 동일하게 동작하기 때문에 메타데이타의 일관성을 유지하는 데는 한계가 존재한다. 또한, 캐쉉 객체의 특성을 고려하지 않는다. 자원의 메타데이타는 정적인 속성을 가지는 데이터뿐만 아니라 동적인 속성을 가지는 데이터를 함께 포함한다. 이것을 동일한 캐쉉 객체로 취급하는 것은 정적인 데이터에 대해서도 주기적으로 갱신을 하기 때문에 전체적인 캐쉉의 효율성을 떨어지게 한다. 마지막으로, 캐쉉 객체의 단위가 비교적 크기 때문에 GRIS 와 GIIS, GIIS 간에 데이터 전송으로 인한 성능의 저하가 있다.

3. 적응형 캐쉉 기법을 적용한 그리드 정보 서비스 구조

기존의 MDS 의 서비스 구조를 개선하기 위해서는 먼저 메타데이타의 특성을 반영해야 한다. 일반적으로 하나의 GRIS 에 의해 유지되는 메타데이타는 정적인 데이터와 동적인 데이터를 함께 가지고 있지만 대부분 정적인 데이터들로 구성되어 있다. 하지만, 사용자에 의해 자주 접근되는 데이터는 동적 데이터이다. 따라서, 이와 같은 데이터는 높은 일관성을 유지하는 것이 요구되어 진다. 뿐만 아니라, 사용자의 접근 패턴(Access Pattern)에 따라 선호되어지는 자원이 존재하므로 전체적으로 볼 때 객체의 인기도에 따라 일관성의 수준을 변화시키는 것이 자원의 활용도(Utilization) 높이기 위해서 필요할 것이다.

본 논문이 이를 위해 제안하는 개선된 MDS 구조는 아래 3 가지로 요약할 수 있다.

- 데이터 생성 및 분산 채널의 이원화
  통합되어 관리되고 있는 자원의 메타데이타를 정적인 데이터와 동적인 데이터로 나누어 생성 및 관리 채널을 다르게 유지하는 것이다. 이를 위해서는 기존의 정보제공자들 중에서 동적인 데이터만을 따로 분리하여 생성하는 정보 제공자를 두는 것이 필요하다.
- 적응형 갱신 주기를 가지는 메타데이타의 캐쉉 기법
  GIIS 가 하위 GIIS 또는 GRIS 로부터 동일한 주기로 등록 메시지를 받지 않고 사용자의 워크로드(Workload)에 따라 등록 주기를 동적으로 변화시켜 인기있는 객체(*Hot Object*)에 대해서는 주기를 높이고 그렇지 않는 객체에 대해서는 주기를 낮춘다.
- 등록 메시지에 갱신된 데이터의 피기백(Piggyback)
  MDS 에서 등록 메시지는 일종의 LDAP 데이터로 LDAP 서버에 유지된다. 따라서, 이를 이용하여 갱신된 데이터에 대한 내용을 등록 메시지와 함께 실어 보내어 GIIS 가 GRIS 를 통해 접속하는 빈도를 줄일 수 있다.

그림 3 은 제안된 기능을 포함하도록 개선한 MDS 의 서비스 구조이다. GRIS 는 정적과 동적으로 분류된 메타데이타를 다른 정보 제공자(State Information Provider)로부터 받는다. 이 정보 제공자는 주로 동적인 특성을 가지는 자원의 모니터링 정보들을 생성하게 된다. 여기서, 동적인 데이터의 정보 제공자는 메타데이타의 속성에 따라 갱신 주기를 변화시켜 데이터의 정확성을 높인다. 또한, GRIS 는 GIIS 에 보내는 등록 메시지의 주기를 변화시켜 GIIS 에 캐쉉된 객체의 갱신 주기를 변화시키게 된다. 그리고, 등록 메시지를 보낼 때, 갱신된 데이터를 실어 보내게 된다.

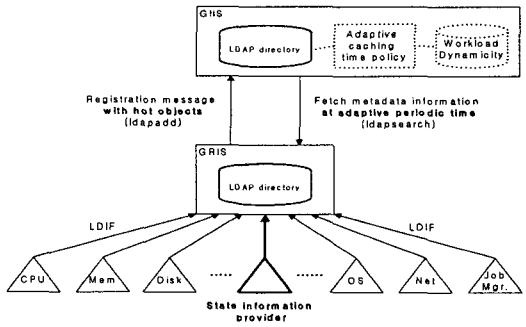

그림 3. 개선된 MDS 서비스 구조

위의 기능을 포함하게 될 경우 얻을 수 있는 효과는 우선 캐쉉 객체의 단위를 동적인 데이터와 정적인 데이터로 분리함으로써 자주 변화하거나 자주 사용되

는 데이터에 대해서는 정확성을 높이고, 이로 인한 오버헤드(예를 들어, 네트웍 비용)를 최소로 감소시킬 수 있다.

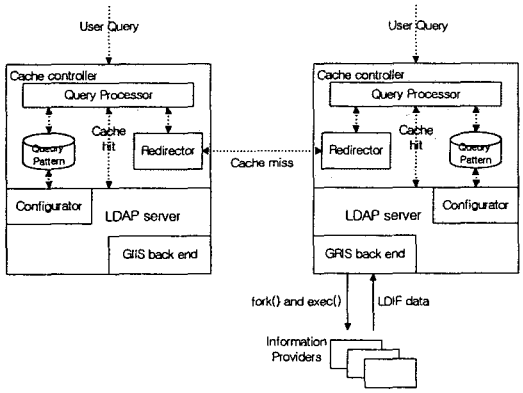

그림 4. 개선된 MDS 구현 방법

그림 4 는 위의 기능을 포함하도록 MDS 를 구현하기 위한 구조이다. 진하게 표시된 부분은 기존에 비해 추가된 캐쉬 컨트롤러(Cache Controller)를 의미한다. 이것은 사용자의 질의를 받아 만족하는 캐쉉 객체를 찾는다. 그 캐쉉 객체가 유효하다면 단순히 캐쉉 객체를 리턴한다(Cache hit). 만약, 유효하지 않다면 하위의 GIIS 나 GRIS 로부터 메타데이타를 갱신하게 된다(Cache miss). 여기서 캐쉬 컨트롤러의 중요한 역할은 하나는 사용자의 질의 패턴을 저장하고 이에 따라 GRIS 와 GIIS 의 갱신 주기를 동적으로 변화시키는 것이다. 즉, 아래와 같이 각 캐쉉 객체에 대해 Cache miss 의 빈도를 표준화하여 계산한 뒤 이를 정렬하여 갱신주기가 긴 캐쉉 객체부터 우선적으로 갱신 주기를 높이게 된다.

$$\text{Cache miss의 횟수} \times \frac{\text{캐쉉 객체의 요청 횟수}}{\text{총 질의 횟수}}$$

사용자 질의 처리 과정을 살펴보면 다음과 같다. 사용자의 질의를 MDS 가 받게 되면 요청된 객체가 이미 캐쉉되어 있는지 체크를 하게 된다. 만약, 유효한 객체가 존재하면 그 객체를 리턴하면서 그 객체에 대한 질의 패턴 저장소를 갱신한다. 이 데이터는 결국 MDS 의 성능을 최적화하기 위한 데이터로 사용된다. 그리고, 요청된 객체가 존재하지 않으면 리다이렉터를 통해 그 질의가 요청하는 객체를 하위의 GIIS 나 GRIS 에 요청하게 된다.

4. 결론

일반적으로 데이터의 일관성과 그 시스템의 성능은 반비례한다. 즉, 분산된 두 데이터의 일관성을 엄격하게 유지하려면 시스템의 동작에 많은 부담을 주게 되어 전체 시스템의 성능은 떨어진다. 따라서 단순히 응답속도를 빠르게 하는 것은 올바른 접근 방법이 아니다. 사용자는 그리드 컴퓨팅 환경에 존재하는 자원들의 효율적인 공유를 원한다. 이를 위해서는 정확한 자원 정보를 통합된 형태로 빠르게 서비스하는 것이 무엇보다 중요하다. 본 논문에서는 이를 해결하기 위해 기존의 그리드 정보 서비스 구조를 분석하고 이를 개선시키는 구조를 제안한다. 위에서 제공된 서비스 구조는 분산된 메타데이타의 일관성을 높이면서 전체 시스템의 성능을 향상 시킬 수 있도록 메타데이타를 그 특성에 따라 정적과 동적인 속성으로 분류하고, 이를 이용하여 가능한 전체 성능에 최소의 부담을 주면서 데이터의 정확성을 개선시킨다. 또한, 사용자의 접근 패턴에 따라 데이터의 정확성을 한층 증가시켜 자원의 이용률를 높일 수 있도록 한다.

향후 연구 과제로는 개선된 MDS 의 디자인을 실제 구현하여 메타데이타의 일관성과 서비스 성능을 평가해야 할 것이다. 이를 위해서는 사용자의 다양한 접근 패턴에 대한 워크로드(Workload)를 함께 분석하는 것이 필요하다.

참고문헌

[1] K. Czajkowski, S. Fitzgerald, I. Foster, C. Kesselman, "Grid Information Services for Distributed Resource Sharing.", Proceedings of the Tenth IEEE International Symposium on High-Performance Distributed Computing (HPDC-10), IEEE Press, August 2001.

[2] I. Foster, C. Kesselman, S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations", International J. Supercomputer Applications, 15(3), 2001.

[3] I. Foster, C. Kesselman, "Globus: A Metacomputing Infrastructure Toolkit", Intl J. Supercomputer Applications, 11(2):115-128, 1997.

[4] I. Foster, G. von Laszewski, "Usage of LDAP in Globus", http://www.globus.org, 1998

[5] S. Fitzgerald, I. Foster, C. Kesselman, G. von Laszewski, W. Smith, S. Tuecke, "A Directory Service for Configuring High-Performance Distributed Computations", Proc. 6th IEEE Symp. on High-Performance Distributed Computing, pp. 365-375, 1997.

[6] Globus site, "MDS2.1 New Features", http://www.globus.org/mds, 2002

[6] Globus site, "MDS2.1 : Creating a Hierarchical GIIS", http://www.globus.org/mds, 2002

[7] Globus site, "MDS2.1 Core GRIS Providers", http://www.globus.org/mds, 2002

[8] RFC2251, "Lightweight Directory Access Protocol(v3)"

[9] RFC2252, "Lightweight Directory Access Protocol(v3) : Attribute Syntax Definitions

[10] OpenLDAP site, "OpenLDAP 2.1 Administrator's Guide", http://www.openldap.org, 2002