그리드 환경에서의 작업 시나리오에 기반한 제약적 권한 위임[1)]

[0]김승현* 김 종* 홍성제* 김상완**

*포항공과대학교 컴퓨터공학과
**한국과학기술정보연구원 슈퍼컴퓨팅센터

Task Execution Scenario based Restricted Delegation in Grid

[0]Seung-Hyun Kim*, Jong Kim*, Sung-Je Hong*, Sangwan Kim**

*Dept. of Computer Science & Engineering,
Pohang Univ. of Science and Technology
**Supercomputing Center, KISTI

요 약

제약적인 권한 위임은 분산 환경에서 작업을 처리하기 위해서 필요한 권한을 부여하는 중요한 과정이다. 하지만 기존 그리드 환경의 아키텍처에서는 작업에 따른 적절한 권한을 부여하지 못하고, 부여한 권한의 오용을 막을 수 없는 등의 문제들이 있다. 본 연구에서는 그리드 환경에서 작업 시나리오에 기반한 제약적 권한 위임을 제안한다. 제안한 기법은 인증서에 작업 시나리오를 유지하여 관리하고, 유일한 식별자로 작업을 구분하는 것으로 기존의 문제점을 해결한다.

I. 서론

분산 환경에서의 작업은 여러 노드들로 분배되어 자체적으로 동작하며, 결과는 이들의 처리 결과를 조합함으로써 이루어진다. 자신의 작업을 다른 노드에게 맡기는 것을 위임(Delegation)이라고 부르며, 이 작업을 처리하는데 필요한 권한을 같이 위임해 주는 과정을 제약적인 권한 위임(Restricted Delegation)[1]이라고 한다.

현재 활성화되어 수행중인 분산 처리 환경 중, 대표적인 것으로 그리드(Grid)[2]를 들 수 있으며 여기서 동작하는 미들웨어는 글로버스, 리전 등이 있다.

제안하는 방식에서는 기존 방식이 작업 수행에 맞는 권한을 제공하지 못하기 때문에 제약적인 권한 위임을 제대로 반영하지 못한다고 본다. 따라서 시나리오에 기반한 인증서를 사용하여 작업의 진행 상황에 맞는 권한 사용을 가능하게 하였으며 인증서 자체에 기반한 권한 제어와 연계하여 동작하도록 하였다.

본 논문은 다음과 같이 구성된다. 본문의 1장에서는 기존 방식에서의 제약적인 권한 위임에 대해 알아보고 연구 동기를 제시한다. 2장에서는 제안하는 방식과 이를 사용하여 동작하는 아키텍처에 대하여 설명하고, 3장에서는 기존의 방식과의 비교를 통해서 제안하는 방식의 이점을 살펴본다. 마지막으로 결론에서 결론 및 연구 방향을 제시한다.

1) 본 연구는 정보통신부 '국가 그리드 기반 구축' 사업의 연구 결과로 KISTI 슈퍼컴퓨팅센터의 지원을 받음.

II. 본문

1. 기존 연구와 연구 동기

1) 그리드 툴킷의 권한 위임

대표적인 그리드 툴킷인 글로버스2.0, 글로버스 CAS, 그리고 리전을 예로 들어 이들이 제약적인 권한 위임을 어느 정도로 제공하는지를 알아본다.

우선 글로버스2.0[3]은 사용자와의 상호 인증이 완료된 다음 로컬의 한 사용자의 권한으로 작업을 수행한다. 작업 수행 중에 특별한 권한 제어는 지원되지 않는다.

글로버스 CAS[4]는 사용자의 역할에 기반하여 권한을 부여받을 수 있다. 하지만 이는 사용자가 사용할 수 있는 모든 권한을 받게 되는 것으로, 작업에 따른 최소한의 권한 부여는 제공하지 못한다.

리전[5]은 사용자 본인이 작업에 기반한 권한을 부여할 수 있다. 설정을 통하여 어느 호스트가 어떤 권한을 사용할 수 있는지를 표시할 수도 있다. 하지만 리전은 확장성이 부족해 그리드 환경의

작업 환경에는 적합하지 않다는 단점이 있다.

2) 작업 기반의 권한 제어

기존의 권한 부여 방식에서는 권한이 너무 일찍 또는 너무 늦게 부여되거나, 작업이 끝난 뒤에도 남아있는 경우가 대부분이다. 이로 인해 생기는 취약점을 줄이기 위해서는 관리자가 모든 작업에 대한 감시를 해야 한다. 하지만 이는 실수할 경우가 많고, 사람이 하기에는 불가능하다.

따라서 권한 부여와 연관된 접근 제어 과정이 자동화되고, 필요한 시점에 필요한 권한을 제공해 주어야 할 필요성이 있으며, 이를 지원하는 작업 기반의 권한 제어(Task-based Authorization Controls : TBAC)[6]가 제안되었다.

그림 1은 작업 기반의 권한 제어 모델을 보인 것으로, 작업의 흐름을 워크 플로우로 모델링하고 A1, A2, A3, A4와 같은 권한 수행 과정을 만나면 그에 해당하는 권한들을 제공해주며 완료된 경우 권한을 회수하게 된다.

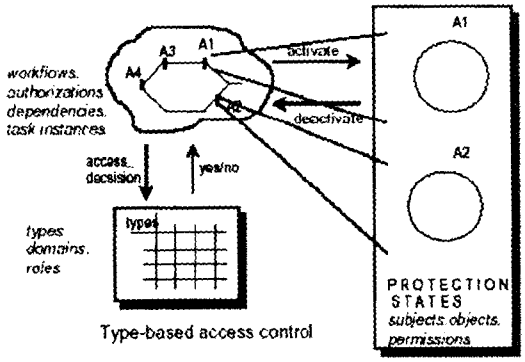

그림 1: 작업 기반의 권한 제어 모델

3) 연구 동기

기존의 그리드 툴킷에서는 현재 작업을 수행하기에 충분하고, 권한의 노출 시에 피해를 최소화하기 위해서 해당 작업에 한정되도록 권한을 부여하지 못한다.

따라서 본 논문에서는 작업에 한정되는 권한을 부여하기 위해, 인증서에 기록된 작업 시나리오에 따라서 권한을 부여하고 해당 프로그램만이 인증서를 사용할 수 있도록 한다. 이를 이용하여 기존의 그리드 툴킷에서 제공하지 못하는 제약적인 권한 위임을 효과적으로 지원할 수 있다.

2. 제안하는 아키텍처

1) 권한 위임 인증서

권한 위임 인증서에는 다음과 같은 파라미터들이 중요하게 고려된다.

□ 발행자(Issuer)
- 해당 인증서를 발행한 주체

□ 주체(Subject)
- 인증서를 사용하는 프로그램의 식별자

□ 시나리오(Scenario)
- 요청하는 권한들의 순서

발행자는 커뮤니티 내의 모든 개체들이 신뢰할 수 있는 제 3의 호스트에 해당하며, 개체들에 대한 모든 접근 권한들을 보유하고 있어서 사용자의 요청이 정당한 지를 판단할 수 있다. 또한 모든 개체들은 이 호스트가 인증한 권한은 신뢰할 수 있다고 미리 합의한다.

주체는 해당 인증서를 사용하여 동작하는 프로그램의 식별자를 명시한다. 프로그램의 식별자는 유일하며 해당 프로그램에서만 사용할 수 있어야 한다. 인증서의 권한을 제공할지는 해당 프로그램의 식별자로 구분한다.

시나리오는 해당 프로그램이 동작하기 위해서 필요한 권한들을 순서에 따라 명기한 것이다. 이 때의 순서는 그림 1과 같이 작업의 흐름을 워크 플로우로 모델링 한 것으로 나타난다.

2) 프로토콜

제안하는 아키텍처는 그림 2와 같다. CA(Comm unity Authority)는 커뮤니티의 모든 개체들이 신뢰할 수 있는 제 3의 호스트이며, 사용자의 요청에 따른 인증서를 인가하는 역할을 한다. 사용자(User)는 작업에 따른 시나리오를 작성하고 CA로부터 인증서를 얻어서 이를 그리드 환경에 넘겨준다. 노드(Node)는 그리드 환경에서 실제로 작업을 처리하는 개체이며 인증서에 기록된 시나리오에 기반하여 프로그램을 제어하게 된다. 표 1에서 이 절에 사용되는 기호들을 표시하였다.

표 1: 논문에서 사용되는 기호

기호	의미
Sp	프로그램의 시나리오
Cu	사용자의 인증서
Ip	프로그램의 식별자
Cp	프로그램의 인증서
P	프로그램

제안하는 아키텍처에서 사용하는 프로토콜은 다음과 같다.

□ Step 1 : 사용자 , [Sp]

□ Step 2 : 사용자 → CA , [Cu, Ip, Sp]

□ Step 3 : CA → 사용자 , [Cp]

□ Step 4 : 사용자 → 노드 , [Cu, Cp, P]

□ Step 5 : 노드 , [Sp]

사용자는 자신의 프로그램에서 필요로 하는 권한들을 명시한 시나리오를 작성한다. 작성한 시나리오와 함께 자신의 정보와 프로그램의 식별자를 CA에게 전달하면, 해당 사용자에게 시나리오에 명시된 권한이 있는지를 확인하게 된다. 시나리오의 권한이 허용 가능하다면 CA는 해당 프로그램이 커뮤니티 내의 객체에서 수행할 수 있다는 인증서를 발행한다. 사용자가 해당 인증서와 프로그램을 노드에게 할당하면, 해당 인증서의 정당성 여부와 프로그램의 식별자를 확인하게 된다. 확인 과정을 통과하면 프로그램은 실행되고, 실행 도중의 권한 요구가 발생할 경우 인증서의 시나리오와 비교를 하게 된다. 해당 시점에서 해당 권한을 요청한 경우에만 권한을 허용한다.

3) 구현 시 고려 사항

제안한 아키텍처를 실제로 구현할 시에 다음과 같은 기술상의 문제를 고려해야 한다.

□ 시나리오 작성

□ 노드의 인증서 확인

□ 수행 도중의 시나리오에 기반한 작업 제어

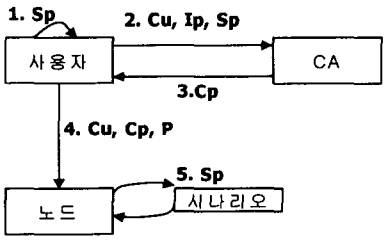

그림 2: 제안하는 아키텍처의 동작과정

본 논문에서는 사용자가 자신의 프로그램에서 필요로 하는 권한들의 흐름을 알 수 있고 이를 순차적으로 명시한 시나리오를 작성할 수 있다고 가정한다. 따라서 사용자가 직접 수작업으로 시나리오를 작성해야 하기 때문에 부담이 가중되는 문제가 생긴다. 컴파일러 기술을 적용하여 자동적으로 시나리오를 생성해주는 방법이 필요하지만, 해당 프로그램이 권한을 요청하는 과정이 결정적이지 않다면, 가령 임의로 반복되는 권한 요청인 경우에는 시나리오를 사용할 수가 없다. 그러므로 프로그램이 권한을 요청하는 과정이 결정적이도록 사용자의 고려가 필요하다.

사용자가 그리드 환경의 노드에게 작업을 넘겨줄 때 사용자의 인증서와 프로그램의 인증서, 그리고 해당 프로그램을 넘겨주게 된다. 이를 바탕으로 작업을 요청하는 사용자의 인증과, 프로그램의 실행에 대한 인증을 처리한다. 프로그램의 식별자를 이용하여 해당 인증서의 유효성을 확인하게 되는데 이 식별자가 유일하며 해당 프로그램에서만 얻을 수 있다고 보장할 수 있어야 한다. 구현 과정에서는 MD5같은 프로그램의 해쉬 값을 염두에 두고 있다.

인증이 끝난 다음에 노드에서 프로그램을 수행하며, 권한이 필요한 작업을 요청할 경우에 시나리오에 명시된 권한인지를 체크해서 해당 권한을 제공해 주어야 한다. 프로그램의 수행 중에 권한이 필요한 작업을 요청할 때 어떤 방식으로 노드에서 처리해야 하는지가 문제가 된다. 이러한 기능을 사용하기 위해서 추가적인 라이브러리를 제공하거나 기존의 프로그래밍 언어에서 제공하는 제약적인 실행 환경[7]을 사용하는 등의 방법을 고려하고 있다.

3. 기존 방식과의 비교

표 2는 제약적인 권한 위임의 요구사항으로 기존의 방식과 제안한 아키텍처를 비교한 것이다.

표 2. 기존 방식과의 비교
G : 글로버스 2.0 , C : 글로버스 CAS
L : 리전 , P : 제안하는 아키텍처

문제	G	C	L	P
작업에 맞는 권한 부여	×	△	○	○
인증서 오용 제한	×	×	△	○
수행 도중의 권한 제어	×	△	△	○

제안하는 아키텍처에서는 프로그램의 시나리오에 기반하여 권한을 부여하기 때문에 작업에 꼭 필요한 권한만이 제공된다. 기존의 방식에서는 이렇게 세밀한 권한 부여를 지원하지 못한다. 리전은 이를 지원하지만 그리드 환경에 적합하지 못한 확장성의 문제가 있다.

악의적인 사용자가 인증서를 가로채어 임의의 프로그램을 실행하려고 하는 경우에, 제안하는 아키텍처에서는 프로그램의 식별자로 인증서의 사용이 잘못 되었음을 판단할 수 있다. 식별자를 위조해서 사용하거나 식별자를 판단하는 단계가 끝난 다음에 임의의 프로그램으로 바꾸는 경우도 고려할 수 있는데, 이 경우는 실행 도중에 요청하는 권한이 시나리오에 명시된 권한과 틀릴 경우에 오용으로 간주하고 해당 권한을 거부할 수 있다.

마지막으로 기존의 방식에서는 작업이 실행되는 시점부터 종료될 때까지 주어진 권한을 모두 사용할 수 있다. 이는 인증서가 악의적인 사용자에 의해 사용될 경우, 인증서에 부여된 모든 권한을 사용할 수 있다는 것을 의미한다. 하지만 제안한 아키텍처에서는 사용자의 프로그램만이 해당 권한을 사용할 수 있으며, 필요한 시점에 필요한 권한만을 제공하기 때문에 인증서가 노출되어도

권한의 무조건적인 오용을 피할 수 있다.

III. 결론

본 연구는 그리드 환경에서 작업을 수행할 때 필요한 시점에 필요한 권한을 제공함으로써 제약적인 권한 위임 문제에 대한 효과적인 방법을 제시하였다. 이전의 방식은 작업에 특정 권한을 부여하지 못하거나, 권한이 할당되어도 작업에 제대로 부합하지 못하였다. 또한 인증서가 악의적인 사용자에 의해 사용되는 경우 이를 제대로 막을 방법이 없었다. 본 연구에서는 사용자가 작성한 시나리오를 사용하여 프로그램의 권한 사용을 제어하였으며 인증서에 프로그램의 식별자를 명시하여 해당 프로그램만이 사용할 수 있으므로 기존의 방식에서 제시된 많은 문제들을 해결할 수 있었다.

향후 연구 방향은 다음과 같다. 우선, 제안한 아키텍처를 좀 더 세밀하고 사용이 용이하도록 정의하고 구현한다. 그리고 실제 그리드 환경에서의 미들웨어와 연계하여 기능을 제공할 수 있도록 구현을 확장한다.

참고문헌

[1] M. Gasser and E. McDermott, "An architecture for practical delegation in a distributed system," *IEEE Symposium on Research in Security and Privacy*, pp. 20-30, 1990.

[2] I. Foster, C. Kesselman, and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," *International Journal of High Performance Computing Applications*, pp. 15(3):200-222, 2001.

[3] I. Foster, C. Kesslcman, G. Tsudik, and S. Tuecke, "A Security Architecture for Computational Grids," *Proceedings of the 5th ACM Conference on Computer and Communications Security Conference*, pp. 83-92, 1998.

[4] L. Pearlman, I. Foster, V. Welch, C. Kesselman, and S. Tuecke, "A Community Authorization Service for Group Collaboration," *Proceedings of the IEEE 3rd International Workshop on Policies for Distributed Systems and Networks*, pp. 50-59, 2002.

[5] A. Ferrari, F. Knabe, M. Humphrey, S. Chapin, and A. Grimshaw, "A Flexible Security System for Metacomputing Environments," *Technical Report CS-98-36, University of Virginia*, 1998

[6] Roshan K. Thomas and Ravi S. Sandhu, "Task-Based Authorization Controls ({TBAC}): A Family of Models for Active and Enterprise-Oriented Autorization Management," *(IFIP) Workshop on Database Security*, pp. 166-181, 1997.

[7] http://py-howto.sourceforge.net/rexec/rexec.html