그리드 어카운팅을 고려한 자원 접근 제어 메커니즘

황 호 전[†] · 안 동 언[††] · 정 성 종[†††]

요 약

현재 그리드 환경에서 자원 접근 제어에 관련된 다양한 방법들이 연구되고 있다. 대부분 그리드 사용자의 자원 접근 권한은 사용자의 특성 및 역할에 따라 부여하도록 설계되었다. 그러나 그리드에 안정적인 자원을 지속적으로 제공하기 위해서는 유틸리티 컴퓨팅에 의한 자원 접근 제어가 이루어져야 한다. 따라서 본 논문에서는 자원 접근 제어에 그리드 어카운팅 개념을 접목시킨 메커니즘을 제안한다. 이 메커니즘은 자원 사용에 대한 어카운팅 정보를 기초로 처리 비용을 산출하고, 사용자의 가용 자금에 따라 자원 접근 여부를 결정하게 된다. 만약 사용자의 가용 자금이 자원 사용에 대한 처리 비용보다 부족할 경우, 사이트의 자원 접근 제어 정책에 따라 그리드 작업을 제어하게 된다. 최종적으로 그리드 작업이 완료되면, 자원 소비자가 자원 제공자측의 유휴 자원을 사용함으로써 발생되는 처리 비용을 지불한다. 그럼으로 본 논문은 그리드 어카운팅에 의한 사용자의 자원 접근을 제어할 수 있는 메카니즘을 제공함으로써, 경제 원리에 준하는 유틸리티 컴퓨팅 환경을 실현할 수 있는 연구로 평가된다.

키워드 : 자원 접근 제어, 그리드 어카운팅, 유틸리티 모델, 자원 사용량

A Resource Access Control Mechanism Considering Grid Accounting

Ho-Joen Hwang[†] · Dong-Un An[††] · Seung-Jong Chung[†††]

ABSTRACT

Currently, many people have been researching diverse mechansmims related to a resource access control in Grid environment. Mostly Grid user's resource access control was designed to authorize according to their attributes and roles. But, to provide Grid with resources continuously, a resource access based on utility computing must be controlled. So, in this paper we propose and implement mechanism that intergrates Grid accounting concept with resource access control. This mechanism calcuates costs of Grid service on the basis of accounting, and determines based on user's fund availibility whether they continue to make use of site resources or not. Grid jobs will be controlled according to a site resource access control policy only if the amount of available fund is less than its costs. If Grid job completed, resource consumer pays for the costs generated by using provider's idle resources. Therefore, this paper provides mechansim to be able to control user's resource access by Grid accounting, so that it is evaluated as the research to realize utility computing environment corresponding to economic principle.

Key Words : Resource Access Control, Grid Accounting, Utility Model, Resource Usage

1. 서 론

그리드[1-3]는 지리적으로 분산된 컴퓨팅 자원들을 통합하여 고성능 메타 컴퓨팅 환경을 제공하는 분산 및 병렬 처리의 새로운 개념으로, 서로 다른 관리 도메인상에 존재하는 정보 및 자원들을 공유한 하나의 유기체적인 VO(Virtual Organization)를 형성하여 방대하고 복잡한 문제를 해결할 수 있는 기술이다.

이와 같이 그리드는 수퍼 컴퓨터, 단일 시스템 이미지, 대용량 저장장치, 어플리케이션 그리고 특수 목적의 장비 등 다양한 형태의 자원들을 제공하는 자원 제공자와 이를 활용하는 자원 소비자로 구성된다[2]. 자원 제공자는 시스템 자원들을 효율적으로 활용하기 위해 유휴 자원들을 그리드에 제공하고, 자원 소비자는 그리드에서 제공하는 자원들을 이용함으로써 단일 시스템상에서 처리할 수 없는 문제들을 해결한다.

따라서 그리드 환경에 안정적인 자원을 공급하고, 지속적인 수요를 창출하기 위해서는 유틸리티 컴퓨팅 모델[4, 5]이 필요하다. 이를 위해서는 먼저 그리드 어카운팅[6, 7] 시스템 개발이 필수적으로 요구된다. 그리드에 참여하는 사이트의

† 준 회 원 : 전북대학교 컴퓨터공학과 박사과정
†† 종신회원 : 전북대학교 전자정보공학부 부교수
††† 정 회 원 : 전북대학교 전자정보공학부 교수
논문접수 : 2006년 3월 24일, 심사완료 : 2006년 6월 15일

어카운팅 시스템은 그들 고유 정책에 따라 사용자의 접근 제어를 설정하고, 소비한 자원 사용량을 측정하여 사용자에게 과금을 부과한다[8]. 그리드 어카운팅 시스템이 자원 접근 제어를 수행하기 위해서는 사용자의 요구사항에 따라 자원을 활용할 수 있도록 작업 환경을 제공해야 한다. 왜냐하면 그리드 어카운팅 시스템이 사용자의 작업을 모니터링하여 자원 사용에 대한 어카운팅 정보(예를 들어, CPU 사용시간, 평균 메모리 사용량 등)를 수집하고, 수집된 어카운팅 정보는 가격 정책에 따라 과금을 산출한다. 그리고 사용자의 가용 자금에 따라 작업을 계속해서 진행할지, 아니면 중지, 삭제할 것인지를 결정해야 하기 때문이다.

그러나 그리드 서비스를 제공하는 큐잉 시스템들이 로컬 작업과 그리드 작업을 동일 큐에서 처리할 경우, 이들간의 자원 접근 제어가 불가능할 뿐만 아니라 보안성을 확보하기가 어렵게 된다. 따라서 본 논문에서는 사이트의 유휴 자원들을 활용해 그리드 사용자의 작업을 처리할 수 있는 환경을 구성하고, 가용 자금에 따라 그리드 작업이 로컬 자원을 사용하도록 제어하는 메커니즘을 제안한다. 이를 위해 동적 큐 관리 방법을 도입하여 로컬 작업과 독립된 그리드 작업 환경을 제공함으로써 로컬 작업과는 별다른 자원 접근 제어가 가능하도록 한다. 그리고 사용자의 가용 자금에 따라 자원 접근을 제어하기 위해 자체 개발한 그리드 어카운팅 시스템과 연동하도록 한다.

본 논문의 구성은 다음과 같다. 2장에서는 그리드 환경에서의 자원 접근 제어에 관련된 방법들을 살펴보고, 3장에서는 그리드 어카운팅을 고려한 자원 접근 제어 메커니즘을 설명한다. 그리고 4장에서는 실험 환경 및 실험 결과를 기술하고, 마지막 5장에서 결론 및 향후 연구에 대해 기술한다.

2. 관련 연구

그리드 환경에서 사용자의 자원 접근 제어에 관련된 다양한 방법들이 존재한다. 그 대표적인 접근 제어 방법들이 CAS(Community Authorization Service)[9], VOMS(Virtual Organization Membership Service)[10], RBAC(Role Based Access Control) Models[11] 등이다.

CAS는 Globus 프로젝트에서 고안한 서비스로, 커뮤니티를 구성하고 있는 자원 제공자가 자원을 요청하는 사용자들에 대해 접근 제어를 할 수 있도록 하였다. 자원 제공자는 신뢰할 수 있는 CAS 서버에게 사용자들의 자원 접근 권한을 부여할 수 있도록 위임하였다. CAS 서버는 Community Policy Database를 운영하여 사용자들의 권리와 자원 제공자의 접근 제어 정책들을 관리한다. 사용자는 커뮤니티에 속해 있는 자원에 접근하고자 한다면, 먼저 CAS 서버로부터 사용자의 자원 접근 권한 정보가 담긴 Proxy Certificate를 발행받아야 한다. 자원 제공자는 사용자의 Proxy Certificate를 기초로 사용자가 자원에 접근할 수 있는지를 결정하였다.

VOMS는 DataGrid[12]와 DataTag 프로젝트의 프레임워크에 맞춰, VO 영역에서 사용자의 접근 권한을 제어하도록 설계되었다. VO는 자체적으로 사용자들의 Policy Attributes (Group memberships, Roles, Capabilities)를 관리하는 서버를 두어 사용자들에게 자원을 접근할 수 있는 퍼미션을 설정한다. 또한 자원 제공자는 여러 VO에 구성될 수 있기 때문에, 그들 고유의 로컬 권한 정책을 실시하여 사용자들에 대한 자원 접근 권한을 제어한다. 사용자는 자신의 Policy Attributes 정보가 담긴 Pseudo-Certificate를 VOMS 서버로부터 발급받아 Proxy Certificate를 만들어 자원에 접근한다. VO와의 사전 협약에 따라, 자원 제공자는 사용자의 퍼미션을 승인하고, 만약 로컬 권한 정책에 위배되는 경우에는 VO에 의해 설정된 퍼미션을 변경함으로써 접근 제어가 가능하도록 하였다.

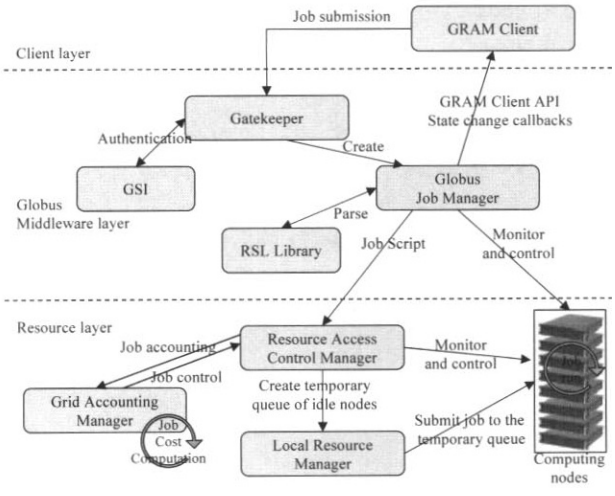
RBAC는 조직내의 구성원들이 가지고 있는 역할에 따라 자원 접근 여부를 결정하는 모델이다. 구성원들의 역할과 자질에 따라 여러개의 Role들을 만들고, 그 Role들마다의 적절한 퍼미션을 부여한다. 만약 사용자가 자원에 접근하게 되면, RBAC는 사용자의 Role에 따른 퍼미션을 부여하고, 그 퍼미션에 따라 액세스할 수 있는 프로시저와 자원의 종류를 결정하게 된다.

위와 같이 그리드 환경에서 사용자의 역할 기반에 따라 자원 접근 제어 방법들이 활발히 연구되고 있다. 그러나 그리드 서비스에 대한 자원 사용량을 측정하고, 다양한 가격 정책에 따라 사용자의 처리 비용을 산출할 수 있는 어카운팅 및 사용자의 가용 자금에 따라 자원에 대한 접근을 제어할 수 있는 연구가 부족한 실정이다. 그리드가 활성화되기 위해서는 그리드 비즈니스 모델 측면에서 사용자의 자원 접근 제어가 반드시 이루져야 한다. 즉 그리드에 참여하는 자원 제공자는 자원 소비자들의 요구사항에 따른 자원 접근 권한을 사용자의 가용 자금에 따라 제어할 수 있어야 한다. 그래서 자원 제공자는 유휴 자원을 활용하여 경제적 이윤을 얻고, 그리드에 자원을 안정적으로 공급할 수 있게 된다. 그러므로 본 논문에서는 그리드 서비스를 위한 작업 환경을 제공하고, 어카운팅 정보를 토대로 처리 비용을 산출하여 사용자의 가용 자금에 의해 자원 접근 제어가 가능한 메커니즘을 제안한다.

3. 그리드 어카운팅을 고려한 자원 접근 제어 메커니즘

Globus Toolkit[1] 기반 그리드 환경에서 전형적인 사용자 작업 처리 과정[13]에서, (그림 1)은 본 논문에서 제안하는 자원 접근 제어 메커니즘이 로컬 사이트에서 Globus Toolkit 주요 구성 요소들과 어떻게 동작하는지를 나타낸 그림이다.

그리드 자원 관리를 담당하는 GRAM(Grid Resource Allocation and Management)[13]은 원격지 자원을 사용할 수 있도록 자원 할당, 프로세스 생성, 모니터링 및 관리 서비스를 제공한다. GRAM 클라이언트는 원격 호스트에 작업을 요청할 때, 자원 사용에 대한 요구사항을 RSL(Resource



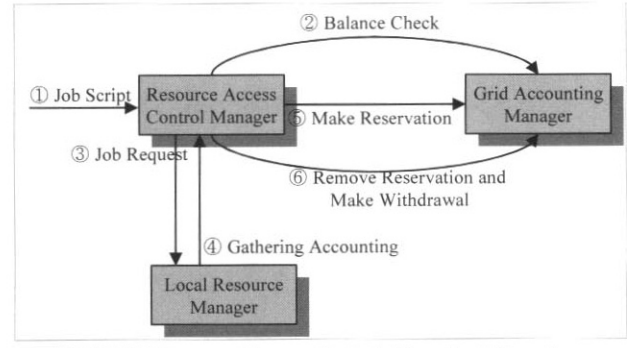
(그림 1) Resource Access Control Manager Considering Grid Accounting in Resource Layer

Specification Language)[14]이라는 언어를 사용해 작업 명세서를 작성하여 전달한다. 원격 호스트의 Gatekeeper는 GSI(Globus Security Infrastructure) 프로시저를 호출하여 사용자와 자원간의 상호 인증을 수행하며, 그리드 작업을 처리할 로컬 계정을 결정하게 된다. 그리고 Globus Job Manager는 RSL Library를 호출하여 RSL 스크립트를 파싱하고, 로컬 스케줄러가 처리할 수 있는 형태의 스크립트로 변환한 후, 로컬 스케줄러에게 사용자 작업을 요청하게 된다.

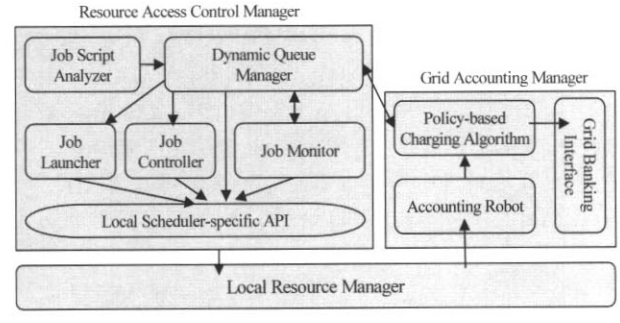
본 논문에서 제안하는 메커니즘을 구현하기 위해 (그림 1)에서와 같이 그리드에 참여하는 사이트에 RACM(Resource Access Control Manager)를 배치한다. (그림 1)에서 작업 스크립트를 해석한 후에 RACM은 그리드 작업을 제출할 새로운 큐를 생성하고, 큐에 제출된 그리드 작업이 로컬 스케줄러로 하여금 유휴 노드에 할당될 수 있도록 작업 노드들의 속성들을 변경한다. 그리고 RACM은 실행중인 그리드 작업에 대해 어카운팅 정보를 수집하고, 수집된 어카운팅 정보는 Grid Accounting Manager에게 전달하여 처리 비용을 산출한 후에 사용자의 가용 자금에 따라 자원 접근 여부를 결정하게 된다. Grid Accounting Manager는 자체 개발한 사이트 가격 정책 기반의 그리드 어카운팅 관리자로서, 어카운팅 정보를 토대로 과금을 책정하고, 여러 가지 지불 방법들을 제공하고 있다.

3.1 Resource Access Control Manager 구조 및 동작 과정

RACM은 Local Resource Manager와 Grid Accounting Manager와의 상호 작용을 통해 사용자 작업을 제어한다. RACM과 Grid Accounting Manager, Local Resource Manager와의 상호 관계를 도식화하면 (그림 2)와 같다. RACM은 그리드 작업이 실행되도록 Local Resource Manager에게 요청한다. 그리고 RACM은 사용자의 가용 자금에 따라 자원 접근 여부를 결정하기 위해 자원 사용에 대한 어카운팅 정보를 수집하여 Grid Accounting Manager에게 전달한다. 작업 완료 후, RACM은 사용자에게 자원 사용에 따른 최종 처리



(그림 2) Resource Access Control Manager Interaction



(그림 3) Resource Access Control Manager Structure

비용을 부과하도록 Grid Accounting Manager에게 요청하고, 사용자를 위한 작업 환경을 해제하게 된다.

이와 같이, 그리드 어카운팅을 고려한 자원 접근 제어 메커니즘을 수행하는 RACM의 구조는 (그림 3)과 같다. RACM는 작업 스크립트를 해석하여 사용자의 자원 요구사항을 분석하는 Job Script Analyzer, 로컬 스케줄러로 하여금 유휴 작업 노드에 그리드 작업이 할당될 수 있도록 하고, 사용자의 가용 자금에 따른 자원 접근 제어를 수행하는 Dynamic Queue Manager, 로컬 스케줄러에게 그리드 작업을 제출하는 Job Launcher, 사용자의 작업 제어를 담당하는 Job Controller, 그리고 일정한 시간마다 작업 상태 및 어카운팅 정보를 모니터링하는 Job Monitor로 구성된다.

먼저, Job Script Analyzer는 작업 스크립트를 분석하여 어플리케이션을 실행하는 데, 필요한 자원 정보를 추출한다. 이를 위해 Job Script Analyzer는 GRAM RSL의 패러미터들의 값들을 얻어오게 된다. 그 대표적은 패러미터들은 count, maxWallTime, maxCpuTime, minMemory 그리고 maxMemory 등이다. Job Script Analyzer가 추출한 자원 정보는 Dynamic Queue Manager에게 전달한다.

Dynamic Queue Manager는 RACM의 핵심 모듈로 사용자의 요구사항에 따라 유휴 작업 노드에 그리드 작업이 스케줄링될 수 있도록 작업 환경을 구축하고, Grid Accounting Manager와의 상호 연동을 통해 자원 접근 제어 메커니즘을 수행한다. Dynamic Queue Manager의 주요 기능과 같다.

● 유휴 노드 검색

로컬 시스템의 전체 작업 노드들을 검색하여, 유휴 상태

인 노드들을 찾아낸다.

●큐 생성 및 작업 노드 속성 변경

그리드 작업을 제출할 큐를 생성하고, 로컬 스케줄러가 유휴 상태의 노드에 작업이 할당될 수 있도록 작업 환경을 설정한다. 생성되는 큐의 이름은 이름 충돌을 방지하기 위해 그리드 작업을 수행하는 로컬 계정으로 한다. 그리고 큐에 제출된 작업이 로컬 스케줄러에 의해 유휴 상태의 작업 노드에 할당될 수 있도록 노드들의 속성들을 변경한다.

예를 들어 (그림 4)에서와 같이, 그리드에 자원을 제공하는 큐잉 시스템이 단일 큐와 12개의 작업 노드를 가지고 있다고 가정하자. 단일 로컬 큐에 제출된 작업은 스케줄링 정책에 따라 로컬 스케줄러에 의해 작업노드 GC01에서 GC12까지 임의의 노드에 할당된다. (그림 4)의 단일 로컬 큐에 작업 A, B가 제출되면, 이 두개의 작업은 (그림 4)와 같이 로컬 스케줄러에 의해 6개의 노드에 할당되어 동시에 처리되고, 그리고 나머지 일부 6개 노드는 유휴 상태에 있다. 이때 GRAM 클라이언트로부터 4개의 노드가 필요한 그리드 작업이 들어오면, Dynamic Queue Manager는 그리드 작업을 제출할 새로운 큐를 생성한다. 그리고 전체 노드를 검색하여 4개의 유휴 상태의 작업 노드, GC07, GC08, GC09, GC10를 찾는다. 그리고 새로 생성된 큐에 제출된 작업이 로컬 스케줄러에 의해 GC07, GC08, GC09, GC10노드에 할당될 수 있도록 노드들의 속성을 변경한다.

또한 멀티 큐를 갖는 시스템일 경우도 마찬가지로, 그리드 작업을 동시에 처리할 수 있는 유휴 상태의 노드들을 검색하고, 그리드 작업을 제출할 큐를 생성한다. 그리고 로컬 스케줄러에 의해 여러 큐에 할당된 유휴 노드들에 작업이 할당될 수 있도록 노드들의 속성을 변경한다. 그리드 작업을 큐에 제출하기 전에 사용자의 요구 사항에 따라 사용할 수 자원의 양(CPU Time, Wall Time 등)을 설정한다. 그런 다음, Job Launcher를 호출하여 새로 생성된 큐에 그리드 작업이 제출한다.

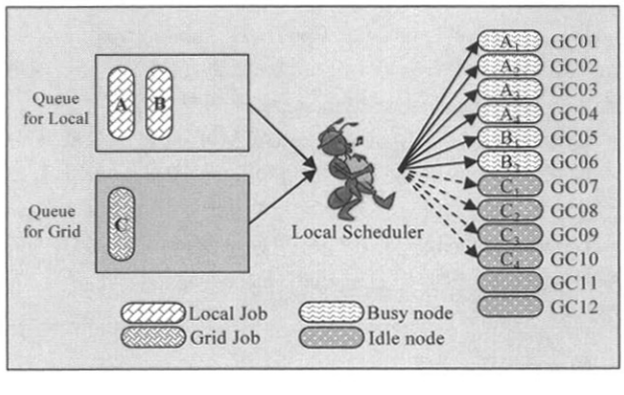
●가용 자금에 따른 자원 접근 제어

그리드 작업을 수행하는 동안, 자원 사용량에 대한 과금을 산출하고, 사용자의 Bank Balance를 초과할 경우 적절한 자원 접근 제어를 할 수 있도록 한다. 자세한 내용은 3.2절에서 언급한다.

●큐 삭제 및 노드 속성 제거

그리드 작업이 완료되면, 원래의 상태로 복구하기 위해 큐를 삭제하고 각각의 노드에 추가된 속성들을 제거한다.

Job Launcher는 Dynamic Queue Manager의 지시에 따라 생성된 큐에 그리드 작업을 제출하고, Job Controller는 처리중인 작업을 중지하거나 삭제한다. 마지막으로 Job Monitor는 Local Scheduler-specific API로부터 자원에 대한 어카운팅 정보를 수집하고, 그리드 작업이 완료되었는지를 감시한다.



(그림 4) Resource Allocation for Grid Job

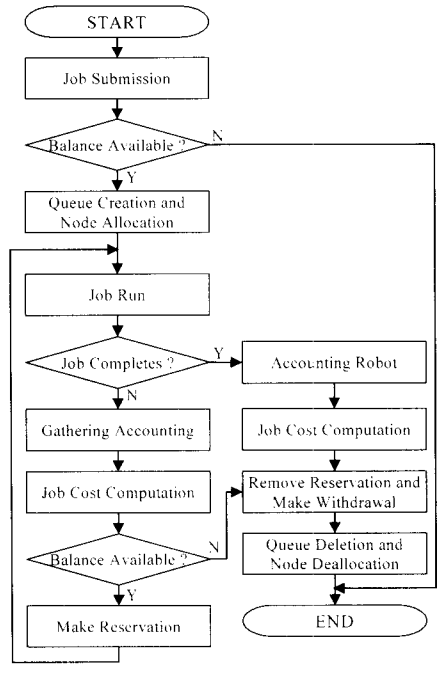
3.2 사용자의 가용 자금에 따른 자원 접근 제어 메커니즘

RACM에서 사용자의 가용 자금에 따른 자원 접근 제어 메커니즘은 (그림 5)와 같다. RACM은 사용자의 작업 처리 요청이 들어오면, 먼저 Grid Accounting Manager에게 사용자의 DN(Distinguish Name) 정보를 넘겨주고, 가용 자금이 있는지를 검사하도록 요청한다. 그래서 일정 이상의 가용 자금이 있으면, RACM은 큐 생성 및 노드 속성을 추가하고, 작업을 제출한다. 작업 수행중에, RACM은 주기적으로 Job Monitor를 호출하여 그리드 작업에 대한 자원 사용량 정보를 수집한다. 수집된 정보는 Grid Accounting Manager에게 전달한다.

Grid Accounting Manager는 RACM으로부터 넘겨 받은 어카운팅 정보를 토대로 과금을 산출하고, Grid Banking System에게 인출 예약을 신청한다. 만약 사용자의 가용 자금이 부족한 경우, Grid Accounting Manager는 RACM에게 작업 제어 정보를 넘겨주게 된다. RACM는 Grid Accounting Manager에게 넘겨 받은 작업 제어 정보로 그리드 작업을 계속할 것인지, 중지할 것인지, 아니면 삭제할 것인지를 결정하게 된다. 최종적으로 그리드 작업이 완료되면, RACM은 Grid Accounting Manager에게 이를 통보한다. 작업 완료 통보를 받은 Grid Accounting Manager는 로컬 스케줄러가 생성한 로그 파일을 분석하여 사용자 과금을 정산한 후에, 인출 예약을 해지하고 사용자의 계좌로부터 처리 비용을 인출하도록 한다.

따라서 본 논문에서 제안하는 그리드 어카운팅을 고려한 자원 접근 제어 메카니즘은 자원 소비자가 자원 제공자의 유휴 자원을 사용할 때 발생하는 처리 비용을 일정한 시간 간격으로 산출하여, 사용자의 Bank Balance보다 부족할 경우 자원 접근 제어 정책에 따라 그리드 작업을 삭제하거나 중지하게 된다.

RACM은 사용자의 가용 자금이 처리 비용보다 부족한 경우, 자원 접근 제어 정책에 따라 그리드 작업을 제어하게 된다. 자원 접근 제어 정책은 "DISCARD"와 "HOLD"로 나눠진다. 자원 접근 제어 정책이 "DISCARD"인 경우에는 가용 자금이 처리 비용보다 부족하면 그리드 작업은 바로 삭제된다. 하지만 "HOLD"인 경우에는 가용 자금이 처리 비용



(그림 5) Resource Access Control Mechanism

보다 부족하면 그리드 작업을 중지하였다가, 후에 가용 자금이 처리 비용보다 많을 경우 다시 로컬 자원을 사용할 수 있도록 한다.

3.3 어카운팅 정보 수집

본 논문에서 제안하는 그리드 어카운팅을 고려한 자원 접근 제어 메커니즘을 구현하기 위해 먼저 그리드 작업에 대한 자원 사용량을 수집해야 한다. 그리드 작업에 대한 어카운팅 정보는 작업이 진행중일 때와 완료될 때 두가지로 나눠서 각각 다른 방법으로 수집한다. 그리드 작업이 진행중일 때는 로컬 스케줄러에 특화된 API를 호출함으로써 자원 사용에 대한 어카운팅 정보를 수집하고, 완료 후에는 로컬 스케줄러가 생성하는 어카운팅 로그 파일에서 그리드 작업에 대한 어카운팅 정보를 수집한다. 수집된 어카운팅 정보는 <표 1>과 같이 Grid Job Accounting Structure로 변환하여 Grid Accounting Manager에게 보내져, 처리 비용을 산출하도록 한다.

〈표 1〉 Grid Job Accounting Structure

```
char userDN[128];      // User's DN
char userID[16];       // Local user ID
char jobName[32];      // Job or application name
char localJobID[32];   // Local job identifier
long jobState;         // Completion status of the job
long startTime;        // The time at which the job started
long endTime;          // The time at which the job completed
long usedNodes;        // Number of nodes used
long usedCPUTime;      // CPU time used
long usedWallTime;     // Wall clock time elapsed
long usedPMem;         // The amount of pysical memory used
long usedVMem;         // The amount of virtual memory used
char unused[16];       // padding bytes
```

3.4 그리드 사용자를 위한 로컬 사용자 계정 관리

GRAM 클라이언트로부터 작업 수행 요청을 받게 되면, 해당 사이트에서는 그리드 작업을 처리할 로컬 계정이 반드시 필요하다[15]. 따라서 본 논문에서는 Identity-based Gridmap 파일 방식으로, 미리 정의된 Account Pool내의 로컬 계정과 그리드 사용자의 DN을 다이나믹하게 일대일 매핑되도록 처리한다(그림 3). 그럼으로써 로컬 사이트에서 수행하는 그리드 사용자의 작업들에 대해 구별 가능한 어카운팅 정보를 수집할 수 있게 된다. 작업 완료 후에는 사용자의 DN과 바인딩된 로컬 계정을 해제한다.

# User's DN	Local Account
/O=Grid/O=Globus/OU=chonbuk.ac.kr/CN=hjhwang	gridusr02
/O=Grid/O=Globus/OU=chonbuk.ac.kr/CN=jeongjin2	gridusr01
/O=Grid/O=Globus/OU=chonbuk.ac.kr/CN=kyongsu	gridusr04
/O=Grid/O=Globus/OU=chonbuk.ac.kr/CN=test	gridusr05

(그림 6) Identity-based Gridmap File

4. 실험 환경 및 결과

4.1 실험 환경

본 연구에서는 PBS(Protable Batch System) 로컬 스케줄러를 사용한 리눅스 클러스터링 시스템에서 MPI 그리드 작업에 대해 자원 접근 제어를 수행하였다. 그리고 본 논문에서 제안한 그리드 어카운팅을 고려한 자원 접근 제어 메커니즘은 자체 개발한 Grid Accounting Manager와 연동하였다. 개발 환경은 다음 <표 2>와 같다.

〈표 2〉 실험 환경

항 목	내 용
리눅스 클러스터링 시스템 (CPU, Memory)	- 그리드 프런트 노드 1대(1GHz, 256M) 작업 노드 8대(866MHz, 256M) - NFS 서버 1대(800Mhz, 256M)
운영체제, 로컬 스케줄러 및 그리드 미들웨어	- Red Hat Linux 7.1(kernel version 2.4.2-2) - Open PBS 2.3.16 - Globus Toolkit 2.4
컴파일러 및 라이브러리	- GCC 2.96 PBS interface library

4.2 실험 결과 및 평가

본 논문에서 제안한 그리드 어카운팅을 고려한 자원 접근 제어 메커니즘을 실험하기 위해 (그림 7)과 같이 Globus Middleware의 RSL Library로부터 파싱된 Job RSL을 입력으로 하였다. 본 논문에서 구현한 RACM의 Job Script Analyzer는 (그림 7)에서 그리드 작업을 처리할 로컬 계정, 작업의 종류, 노드 개수 정보를 추출하였다.

Dynamic Queue Manager는 PBS 전체 작업 노드들을 검사하여 유휴 상태인 노드들을 찾아내고(그림 8), 그리드 작업을 처리할 로컬 계정의 큐를 일시적으로 생성하였다(그림 9). 마지막으로 로컬 스케줄러로 하여금 일시적으로 생성된

```
&("environment" = ("HOME" "/home/gridusr02" ) ("LOGNAME"
    "gridusr02" ) ("GLOBUS_DUROC_SUBJOB_SERIALNO" "1" )
    ("GLOBUS
_DUROC_CHECKIN_CONTACT"
    "LSP3000000000000000010100040000000000000240000000000005EC6
    861656A756B2E63686F6E62756
B2E61632E6B72000001000600000040300" )
    ("GLOBUS_DUROC_JOB_SERIALNO" "1" )
    ("GLOBUS_DUROC_DUCT_CONTACT" "LSP3000
0000010000001010004000000010000002400000000000005ec6861656a756b2
    e63686f6e62756b2e61632e6b72000001000600000040
300" ) ("GLOBUS_DUROC_DUCT_ID" "1" ) )("rslsubstitution" =
    ("GLOBUSRUN_GASS_URL" "https://iat.chonbuk.ac.kr:
1028" ) )("stderr" = $("GLOBUSRUN_GASS_URL") # "/dev/stderr"
    )("stdout" = $("GLOBUSRUN_GASS_URL") # "/dev/stdou
t" )("resourcemanagercontact" = "grid.chonbuk.ac.kr/jobmanager-pbs"
    )("jobtype" = "mpi" )("count" = "3" )("executable" =
    "gsiftp://iat.chonbuk.ac.kr/home/hjhwang/jobmanager-pbs/mpi/Test
    MPI" )
```

(그림 7) A Sample of Job RSL

```
int getPBSFreeNode( char *freeNodeName[MAXNODES] ) {
    int i = 0;
    struct attrl *attributeList;
    struct batch_status *pbsNode, *nodePtr;

    pbsNode = pbs_statnode( pbsConnector, NULL, NULL, NULL );

    for( nodePtr = pbsNode; nodePtr != NULL; nodePtr = nodePtr->next ) {
        for( attributeList = nodePtr->attribs; attributeList != NULL; attributeList = attributeList->next ) {
            if( strcmp( attributeList->name, "state" ) == 0 && strcmp( attributeList->value, "free" ) == 0 )
                freeNodeName[i++] = strdup( nodePtr->name );
        }
    }
    pbs_statfree( pbsNode );
    return i;
}
```

(그림 8) Search Idle Nodes

```
void createQueue( struct QueueAttribute * queue ) {
    struct attropl *attribute = NULL;

    pbs_manager( pbsConnector, MGR_CMD_CREATE, MGR_OBJ_QUEUE, queue->name, NULL, NULL );
    attribute = addAttributes( attribute, "queue_type", NULL, EQ, queue->type );
    attribute = addAttributes( attribute, "resources_max", "walltime", EQ, queue->walltime );
    pbs_manager( pbsConnector, MGR_CMD_SET, MGR_OBJ_QUEUE, queue->name, attribute, NULL );
    // The rest is omitted.
}
```

(그림 9) Temporary Queue Creation

```
void setNodeAttribute( char *freeNodeName[], struct QueueAttribute *queue ) {
    int i;
    struct attropl *nodeAttribute = NULL;

    nodeAttribute = addAttributes( nodeAttribute, "properties", NULL, INCR, queue->name );
    for( i = 0; i < queue->count; i++ ) {
        pbs_manager( pbsConnector, MGR_CMD_SET, MGR_OBJ_NODE, freeNodeName[i], nodeAttribute, NULL );
    }
    // The rest is omitted
}
```

(그림 10) Set Node Attribute Associated with Queue

큐에 들어온 작업이 유휴 상태의 노드에 할당될 수 있도록 (그림 10)과 같이 작업 노드의 속성들을 변경하였다.

이와 같은 일련의 과정을 거쳐 그리드 작업이 로컬 시스템에서 수행되어진다. RACM은 사용자의 가용 자금에 따라 자원 접근 메커니즘을 수행하기 위해, 주기적으로 그리드 작업의 어카운팅 정보를 수집(그림 11)한 후, 작업 노드 개수 및 CPU 사용시간을 추출하여 Grid Accounting Manger에게 전달한다. Grid Accounting Manager는 어카운팅 정보를 토대로 처리 비용을 산출하고, 사용자의 가용 자금에 따라 앞으로의 작업 진행 여부를 RACM에게 알려준다. RACM은 자원 접근 정책에 따라 그리드 작업을 삭제할것인지, 보류할 것인지를 결정하게 된다. 본 논문에서는 PBS 로컬 스케줄러가 실행중인 작업을 중지하였다가 다시 실행시

```
339.grid.chonbuk.ac.kr
    Job_Name = STDIN
    Job_Owner = gridusr02@grid.chonbuk.ac.kr
    resources_used.cput = 00:00:00
    resources_used.mem = 3728kb
    resources_used.vmem = 8092kb
    resources_used.walltime = 00:13:17
    job_state = R
    queue = gridusr02
    server = grid.chonbuk.ac.kr
    Checkpoint = u
    ctime = 1138685249
    exec_host = gc07/0+gc06/0+gc05/0
    Hold_Types = n
    Join_Path = n
    Keep_Files = n
    Main_Points = n
    mtime = 1138685249
    Priority = 0
    qtime = 1138685249
    Rerunable = True
    Resource_List.cput = 00:32:00
    Resource_List.ncpus = 3
    Resource_List.nodect = 3
    Resource_List.nodes = 3
    Resource_List.walltime = 00:32:00
    session_id = 3658
```

(그림 11) Monitoring Grid Job Accounting by calling Local Scheduler-specific API

```
01/31/2006 14:47:45;E;339.grid.chonbuk.ac.kr;user=gridusr02
    group=users jobname=STDIN queue=gridusr02 ctime=1138685249
    qtime=1138685249
 etime=1138685249 start=1138685249 exec_host=gc07/0+gc06/0+gc05/0
    Resource_List.cput=00:32:00 Resource_List.ncpus=3 Resource_List
.neednodes=3 Resource_List.nodect=3 Resource_List.nodes=3
    Resource_List.walltime=00:32:00 session=3658 end=1138686465
    Exit_status
=0 resources_used.cput=00:20:15 resources_used.mem=3728kb
    resources_used.vmem=8092kb resources_used.walltime=00:20:16
```

(그림 12) Grid Job Accounting in PBS Accounting Log File

킬 수 있는 기능이 없어, "DISCARD" 자원 접근 제어 정책만을 설정하였다. 따라서 RACM은 사용자의 가용자금이 처리 비용보다 부족한 경우, 바로 그리드 작업을 삭제하도록 처리하였다.

그리드 작업이 완료되면, 최종적으로 PBS 로컬 스케줄러가 생성한 어카운팅 로그 파일(그림 12)에서 <표 1>의 그리드 작업 어카운팅 구조체를 생성하였다. 그리고 그리드 작업 어카운팅 정보를 Grid Accounting Manager에게 전달하여 사용자의 계좌에서 처리 비용이 인출되도록 하였다. 마지막으로 그리드 작업을 위해 생성한 큐를 삭제하고, 작업 노드들의 속성들을 원래대로 변경하였다.

그리고, Grid Accounting Manager에서는 그리드에 참여하는 각 사이트들간의 자원 사용량 정보를 교환하기 위해 공통된 포맷의 Usage Record[16][17]를 정의하였다. (그림 13)의 Usage Record를 보면, "/O=Grid/O=Globus/OU=chonbuk.ac.kr/CN=hjhwang"라는 DN을 가진 그리드 사용자는 iat.chonbuk.ac.kr 클라이언트에서 grid.chonbuk.ac.kr 사이트에 그리드

```
<?xml version="1.0" encoding="UTF-8" ?>
- <JobUsageRecord xmlns="http://www.gridforum.org/2003/ur-wg" xmlns:urwg="http://www.gridforum.org/2003/ur-wg"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.gridforum.org/2003/ur-wg
   file:/Users/bekah/Documents/GGF/URWG/urwg-schema.09.xsd">
   <RecordIdentity urwg:recordId="339.grid.chonbuk.ac.kr" urwg:createTime="2006-01-31 15:00:00" />
  - <JobIdentity>
     <LocalJobId>339.grid.chonbuk.ac.kr</LocalJobId>
   </JobIdentity>
  - <UserIdentity>
     <GlobalUsername>/O=Grid/O=Globus/OU=chonbuk.ac.kr/CN=hjhwang</GlobalUsername>
   </UserIdentity>
   <JobName>STDIN</JobName>
   <Charge>20</Charge>
   <Status>completed</Status>
   <WallDuration>00:20:16</WallDuration>
   <CpuDuration>00:00:00</CpuDuration>
   <EndTime>2006-1-31 14:47:45</EndTime>
   <StartTime>2006-1-31 14:27:29</StartTime>
   <MachineName>IATLab Linux Cluster</MachineName>
   <Host>grid.chonbuk.ac.kr</Host>
   <SubmitHost>iat.chonbuk.ac.kr</SubmitHost>
   <Queue>aTemporaryQueue</Queue>
   <ProjectName>Site Resource Access Control Mechanism</ProjectName>
</JobUsageRecord>
```

(그림 13) GGF-UR Formatted XML

작업을 제출하였다. 그리고 해당 사이트에서 사용자의 작업을 처리하는데 걸리는 시간은 20분 16초이며 작업 상태는 completed이다. 또한 사이트의 가격 정책에 따라 Grid Accounting Manager으로부터 산출된 처리 비용은 20G(Virtual Currency Unit)이다.

지금까지 본 논문에서 제안하는 그리드 어카운팅을 고려한 자원 접근 제어 메커니즘을 살펴보았다. 현재 VOMS는 메타 컴퓨팅 환경하에서 사용자의 정보 및 접근 권한등이 VUS(Virtual Users Server)에 의해 제어되는 중앙 집중화된 구조를 지니고 있다[10]. 한 서버에 모든 권한을 위임하고 각 원격 호스트에 대한 사용자의 접근 권한을 부여하는 것은 그리드 본질과는 상반된 개념이다. CAS는 사용자의 인증서에 직접 자원 접근 권한을 기술하고, 각 자원에서는 인증서를 토대로 자원 접근 제어가 이루어진다[9]. 그러나 자원 제공자쪽에서는 CAS 서버로부터 이미 결정된 접근 권한을 자신들의 정책에 따라 변경할 수 있는 권리를 침해 받는다. 그러므로 CAS 기반 자원 접근 제어 방법은 그리드에 참여하는 자원 제공자의 자율성을 침해하는 문제점을 지니고 있다.

DGAS(DataGrid Accounting System)[12], SWAS(SweGrid Accounting System)[18]와 같은 그리드 어카운팅 시스템들은 사용자의 작업 어카운팅 수집 및 처리 비용을 산출하는 쪽에 초점만을 맞추고 있다. DGAS는 자원 제공자들간의 자원 교환 측면만을 고려하였고, 또한 사이트들간의 어카운팅 정보를 교환할 수 있는 공통된 포맷이 존재하지 않기 때문에 그리드 서비스를 상업화하는데 있어 취약하다는 단점들을 드러내고 있다. 그리고 SWAS는 사용자의 가용 자금에 따라 그리드 서비스를 제어할 수 있는 메커니즘을 지닌 정교한 시스템이라 할 수 있다. 그러나 그리드에 참여하기 위해 각 사이트의 로컬 어카운팅 시스템을 변경하는 일은 사이트의 자율성을 침해한다.

그러나 본 논문에서는 제안하는 메커니즘은 그리드 어카운팅 개념을 접목하여, 사용자의 가용 자금에 따라 자원에 접근할 수 있도록 하였다. 각 사이트에 설치된 로컬 어카운팅 시스템으로부터 측정된 어카운팅 정보를 사용하고, 중앙 집중화된 구조의 자원 접근 제어가 아닌, 사용자의 가용 자금에 따라 자원에 접근 제어가 이루어짐으로써 사이트의 자율성을 보장하였다. 그리고 로컬 작업과 그리드 작업들간의 독립된 작업 공간을 제공함으로써 보안성을 확보할 수 있게 된다. 따라서 본 연구를 통해 그리드에 참여하는 각 사이트에서는 유휴 자원들을 효율적으로 활용할 수 있게 되고, 사이트의 경제적인 이윤을 통해 그리드 환경에 안정적인 자원을 공급할 수 있다고 사료된다.

5. 결론 및 향후 연구

그리드 자원의 안정적인 수요와 공급이 뒷받침되기 위해서는 유틸리티 모델을 지원하는 그리드 컴퓨팅 환경을 구축해야 한다. 이를 위해서, 그리드에 참여하는 사이트에서는 사용자의 가용 자금에 따라 그리드 작업의 자원 접근 제어가 필수적으로 요구된다. 그리드 어카운팅을 고려한 자원 접근 제어를 수행하기 위해 그리드 작업을 처리하고 어카운팅 정보를 수집하는 로컬 스케줄러와, 처리 비용을 산출하고 사용자의 가용 자금 정보를 얻어낼수 있는 그리드 어카운팅 시스템과 연동하도록 설계하였다.

본 논문에서는 그리드 어카운팅 개념을 접목하여 사용자의 가용 자금에 따라 자원에 접근할 수 있도록 구현하였다. 이를 위해 동적 큐 관리 방법을 도입하여 그리드 작업이 로컬 작업과는 독립된 자원 접근 제어가 가능하도록 하였다. CAS나 VOMS에서의 중앙 집중식 자원 접근 제어 문제점을 해결하기 위해, 각 사이트의 로컬 권한 정책에 따라 사용자의 접근 여부를 승인하고 사용자 어카운팅 정보에 따라 접근 제어가 이루어지도록 하였다. 마지막으로 그리드에 참여하는 사이트의 자율성을 침해하지 않도록 하기 위해, 기존의 로컬 어카운팅 시스템을 변경하지 않고 그리드 어카운팅 정보를 추출하였다. 그리고 그리드 작업을 제어하기 위해 로컬 스케줄러와의 상호 작용을 통해 로컬 어카운팅 정책에 따라 그리드 작업을 제어할 수 있도록 처리하였다.

향후 그리드 사용자들이 경제 원리에 준하는 그리드 컴퓨팅 자원들을 활용할 수 있도록 다양한 가격 정책 기반의 그리드 어카운팅 모델을 고려한 자원 접근 제어 메커니즘이 연구되어야 할 것이다. 이를 위해서는 여러 종류의 경제 모델들을 제공하는 그리드 환경이 조성되어야 할 것이다. 그리고 무엇보다도 그리드 환경을 인지할 수 있는 로컬 스케줄러 및 그리드 어카운팅 시스템 연구가 필요할 것이다.

참 고 문 헌

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

[2] Ian Foster, Carl Kesselman, "The Grid: Blueprint for a New Computing Infrastructure," Mogan Kaufmann Publishers, 1998.

[3] I. Foster, C. Kesselman, S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," International J. Supercomputer Applications, Vol.15, No.3, 2001.

[4] Michael Treaster, Nadir Kiyanclar, Gregory A. Koenig, and William Yurcik, "A Distributed Economics-based Infrastructure for Utility Computing," ACM Computing Research Repository Technical Report cs.DC/0412121, December, 2004.

[5] G. A. Koenig and W. Yurcik, "Design of an economics-based software infrastructure for secure utility computing on supercomputing clusters," in 12th Intl. Conference on Telecommunication Systems - Modeling and Analysis, 2004.

[6] Rajkumar Buyya, Heinz Stockinger, Jonathan Giddy, David Abramson, "Economic Models for Management of Resources in Grid Computing," European Council for Nuclear Research(CERN), 2001.

[7] R. Buyya, D. Abramson, and J. Giddy, "A Case for Economy Grid Architecture for Service Oriented Grid Computing," http://www.buyya.com/papers/ecogrid.pdf.

[8] Anthony Beardsmore, Keith Hartley, et al, "GSAX(Grid Service Accounting Extensions)," GGF OGSA Resource Usage Service Working Group, 2002.

[9] L. Pearlman, V. Welch, I. Foster, C. Kesselman, S. Tuecke, "A Community Authorization Service for Group Collaboration," 3rd International Workshop on Pplicies for Distributed Systems and networks, IEEE Computer Society, pp.50-59, 2002.

[10] R. Alfieri, R, Cecchini, V. Ciaschini, L. Dell'Agnello, A Frohner, A. Gianoli, K. Lorentey and F. Spataro, "VOMS, an authorization system for virtual organizations," 1st European Access Grids Conference, Lecture Notes in Computer Science, Spinger, Vol.2790, pp.33-40, 2004.

[11] Hai Jin, Weizhong Quiang, Xuanhua Shi, Deqing Zou, "RB-GACA:A RBAC based Grid Access Control Architecture," International Journal of Grid and Utility Computing (IJGUC), Vol.1, pp.61-70.

[12] Guarise, A., Piro, R., and Werbrouck, A. DataGrid Accounting System - Architecture - v1.0. DataGrid-01-TED-0126-1_0. EU DataGrid 2003. http://server11.infn.it/workload-grid/docs/DataGrid-01-TED-0126-1_0.pdf.

[13] Karl Czajkowski, Ian Foster, et al, "A Resource Management Architecture for Metacomputing Systems," Procs. of the 4th Workshop on Job Scheduling Strategies for Parallel Processing, 1998.

[14] The Global Grid Forum, http://www.ggf.org.

[15] Laura F. McGinnis and William Thigpen and Thomas J. Hacker. Accounting and Accountability for Distributed and Grid Systems. 2nd IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2002), 22-24 May 2002, Berlin, Germany 2002.

[16] Usage Record, GGF Working Group, http://forge.gridforum.org/projects/ur-wg.

[17] OGSA Resource Usage Service, GGF Working Group, http://forge.gridforum.org/projects/rus-wg.

[18] Erik Elmroth, Peter Gardfjäll, Olle Mulmo, Åke Sandgren, Thomas Sandholm, "A Coordinated Accounting Solution for SweGrid", Draft 0.1.3, October, 2003.

황 호 전

e-mail : hjhwang@mail.chonbuk.ac.kr
1997년 전북대학교 컴퓨터공학과(공학사)
1999년 전북대학교 컴퓨터공학과(공학석사)
1999년~현재 전북대학교 컴퓨터공학과 박사과정
관심분야 : 분산 및 병렬처리, 그리드

안 동 언

e-mail : duan@moak.chonbuk.ac.kr
1981년 한양대학교 전자공학과(공학사)
1987년 KAIST 전산학과(공학석사)
1995년 KAIST 전산학과(공학박사)
1995년~현재 전북대학교 전자정보공학부 부교수
2001년~2002년 전북대학교 정보검색시스템연구센터 센터장
관심분야 : 정보검색, 한국어정보처리, 문서분류, 문서요약

정 성 종

e-mail : sjchung@moak.chonbuk.ac.kr
1975년 한양대학교 전기공학과(공학사)
1981년 휴스턴대학교 전자공학과(공학석사)
1988년 충남대학교 전산공학과(공학박사)
1985년~현재 전북대학교 전자정보공학부 교수
1996년~1998년 전북대학교 전자계산소 소장
2001년~현재 전북대학교 BK21 전자정보사업단 단장
관심분야 : 정보검색, Grid