서비스 기반 그리드 환경에서의 적응적 스케쥴링 기법

이종혁[O†] 진성호[†] 이화민[†] 이대원[†] 유헌창[†]
† 고려대학교 대학원 컴퓨터교육학과
{spurt[O], wingtop, zelkova, daelee, yuhc}@comedu.korea.ac.kr

Adaptive Scheduling in a Service-Based Grid Environment

JongHyuk Lee[O†] SungHo Chin[†] HwaMin Lee[†] DaeWon Lee[†] HeonChang Yu[†]
† Dept. of Computer Science Education, Korea University

요 약

서비스를 기반으로 하지 않은 기존의 그리드 환경에서 병렬 작업 처리를 위한 그리드 어플리케이션은 여래 개의 노드에서 병렬적으로 동일한 작업을 수행하였지만, 웹서비스가 통합된 서비스 기반의 그리드 환경에서는 선형 워크플로우가 여러 개의 서비스 노드에 걸쳐 수행된다. 그러므로 그리드 어플리케이션의 수행 속도 향상을 위해서는 서비스와 서비스 간의 관계를 고려하여야 한다. 그러나 그리드 자원이 각 관리영역마다 이질적이고 그리드 자원의 상태가 동적이어서 그리드 어플리케이션의 성능을 예측하기는 어렵다. 또한 서비스 기반의 그리드 컴퓨팅 환경에서는 사용자의 QoS(Quality of Service)를 보장하여 사용자의 요구 사항을 만족시킬 수 있어야 하기 때문에 사용자 요구에 미달하는 성능 저하와 같은 결함이 발생하였을 경우 이에 대처할 수 있는 방법이 필요하다. 따라서 본 논문에서는 서비스 기반 그리드 컴퓨팅 환경의 특징을 반영하는 워크플로우 관리 시스템의 구조를 설계하고 서비스 수행 성능을 예측하기 위한 서비스 특성에 관한 모델링과 적응적 스케쥴링 기법을 제안한다. 제안하는 적응적 스케쥴링 기법에서는 서비스 간의 관계를 고려하기 위해 최대 흐름 알고리즘(Maximum-Flow Algorithm)을 이용하는 방법과 사용자의 QoS에 대한 수행 성능을 보장하기 위해 서비스의 성능 저하 시에 발생하는 결함을 포용할 수 있는 방법을 제안한다.

1. 서 론

그리드 컴퓨팅은 세계 IT 시장의 핵심 기술로 인식되고 있는 웹서비스와의 접목으로 차세대 인터넷이라 칭해지며 세계 각국에서 그리드 컴퓨팅에 관한 연구가 활발히 이루어지고 있다. 그리드와 웹서비스의 통합을 시도하기 위해 웹서비스를 확장한 OGSI(Open Grid Service Infrastructure)와 웹서비스의 표준을 따르는 WSRF(Web Service Resource Framework)가 차례대로 발표되면서 이를 기반으로 하는 OGSA(Open Grid Service Architecture)는 그리드 컴퓨팅 기술과 웹서비스 기술을 빠르게 결합하고 있다. 현재 그리드 미들웨어 표준으로 자리 잡은 GT4(Globus Toolkit version 4)가 WSRF를 기반으로 발표되면서 그리드 컴퓨팅은 서비스 기반 컴퓨팅 환경이 되었다.

서비스 기반의 그리드 컴퓨팅 환경에서 그리드 어플리케이션은 그리드 서비스들 사이의 협업을 통해 이루어진다. 이를 위해 과학 및 비즈니스 처리를 위한 작업은 그리드 서비스 단위로 표현되고 여러 개의 그리드 서비스를 조합하여 그리드 어플리케이션 또는 그리드 서비스를 생성한다. 그리드 어플리케이션은 각 그리드 서비스의 의존 관계를 표현한 워크플로우로 나타낼 수 있다. 대부분의 과학 어플리케이션과 트랜잭션 처리를 위한 비즈니스 어플리케이션은 간단한 선형 워크플로우 구조를 갖는다. 즉, 과학 어플리케이션은 같은 종류의 상이한 데이터에 대해 같은 코드를 이용하여 다중 인스턴스를 생성·수행하여 결과를 수집하는 파라미터 스윕 어플리케이션(Parameter Sweep Application) 형태이며 비즈니스 어플리케이션은 데이터베이스에서 자료를 조회하고 처리하여 다시 데이터베이스에 저장하는 형태이다. 과학 어플리케이션은 넓은 파라미터 범위를 가지고 대용량의 데이터를 병렬적으로 처리할 수 있기 때문에 그리드 환경에서 수행 성능의 이익을 가져올 수 있다. 기존의 그리드 환경에서의 과학 어플리케이션은 여러 개의 노드에서 병렬적으로 동일한 작업을 수행하였지만, 서비스 기반의 그리드 환경에서는 선형 워크플로우가 여러 개의 서비스 노드에 걸쳐 수행되기 때문에 과학 어플리케이션의 수행 속도 향상을 위해서는 서비스와 서비스 간의 관계를 고려하여야 한다.

워크플로우 관리 시스템에서 그리드 어플리케이션의 워크플로우를 생성하기 위해서는 워크플로우 언어를 이용하여 사용자가 요구하는 그리드 서비스를 구성하는 단계와 그리드 서비스를 실행할 자원을 선택하여 매핑하는 스케쥴링 단계를 거치게 된다. 그러나 스케쥴링 단계에서 요구하는 그리드 자원이 각 관리영역마다 이질적이어서 관리영역마다 자원 관리자를 고려하기 어렵고 그리드 자원의 상태가 동적이어서 그리드 어플리케이션의 성능을 예측하기 어렵다. 또한 서비스 기반의 그리드 컴퓨팅 환경에서는 사용자의 요구 사항을 만족시키는 사용자의 QoS(Quality of Service)를 보장하여야 하기 때문에 사용자 요구에 미달하는 성능 저하와 같은 결함이 발생하였을 경우 이에 대처할 수 있는 방법이 요구된다.

따라서 본 논문에서는 서비스 기반 그리드 컴퓨팅 환경의 특징을 반영하는 워크플로우 관리 시스템의 구조를 설계하고 서비스 수행 성능을 예측하기 위한 서비스 특성에 관한 모델링과 최대 흐름 알고리즘을 이용한 적응적 스케쥴링 기법을 제안한다.

2. 관련 연구

그리드 컴퓨팅 환경에서는 여러 개의 관리 영역을 고려하여 스케쥴링하기 때문에 슈퍼스케쥴링 또는 메타스케쥴링이라고 한다. 일반적으로 스케쥴링은 스케쥴링 시점에 따라 정적 스케쥴링과 동적 스케쥴링으로 나뉜다. 정적 스케쥴링은 사용자 작업이 수행되기 전에 모든 작업 순서가 미리 정해진 스케쥴링이다. 반면에 동적 스케쥴링은 사용자 작업이 수행되기 전 뿐만 아니라 수행 중에도 작업 순서를 정할 수 있는 스케쥴링이다.

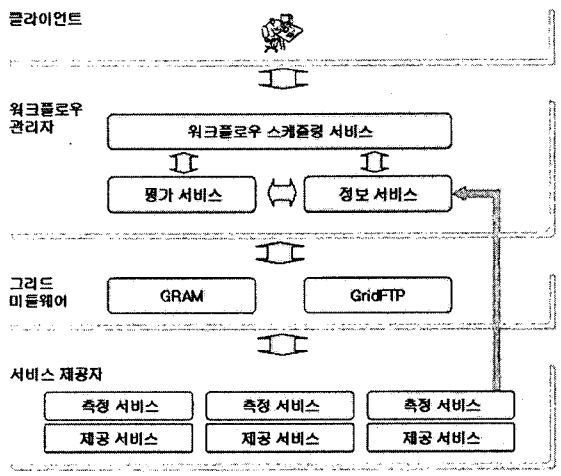
관련연구[1]은 이질적인 분산 컴퓨팅 시스템을 사용하여 독립적인 작업을 동적으로 매핑하는 휴리스틱 기법을 연구하였다. 이 기법에서 매핑 휴리스틱은 작업이 매퍼(mapper)에 도착하자마자 자원에 할당하는 온라인 방식과 작업을 매퍼에 모았다가 매핑 이벤트가 발생하면 모든 작업을 매핑하는 배치 방식의 두 가지로 나누어 설명하였다. 첫 번째 배치 방식은 min-min 방식으로 각각의 작업에 대하여 MinECT(Minimum Earliest Completion Time)를 가지는 작업을 ECT(Earliest Completion Time)를 갖는 호스트에 할당하는 방식이다. 두 번째 배치 방식은 max-min 방식으로 min-min과는 다르게 MaxECT(Maximum Earliest Completion Time)를 가지는 작업을 ECT를 갖는 호스트에 할당하는 방식이다. 즉, max-min은 긴 작업 보다 짧은 작업이 더 많을 경우에 유리하고 min-min은 그 반대로 짧은 작업보다 긴 작업이 더 많을 경우에 유리하다. 마지막 배치 방식은 sufferage 방식으로 각각의 작업에 대하여 어떤 호스트 m_x에서 EST(Earliest Completion Time)와 어떤 호스트 m_y에서 SEST(Second Earliest Completion Time)의 차가 가장 큰 작업을 호스트 m_x에 할당하는 방식이다. 관련연구[2]는 관련연구[1]에서 제안한 세 가지 배치 방식에 대하여 MinECT를 계산할 때 입력, 출력 데이터 전송 시간과 이미 파일이 원격 저장 장소에 저장될 수 있다는 가정을 추가하여 수정 보완하였다. 특히 sufferage 값을 클러스터 레벨로 측정하여

sufferage를 확장한 XSufferage를 제안하였다. 즉, 클러스터 내의 호스트 중 가장 작은 MinECT를 클러스터의 MinECT으로 하고 이를 클러스터 간 sufferage 값으로 사용한다. 관련연구[3]은 그리드 환경에서 파라미터 스윕 어플리케이션을 효율적으로 수행하고 스케쥴링하기 위하여 GridWay 프레임워크를 제안하였다. 이 프레임워크는 동적인 그리드 자원 환경을 반영한 적응적 스케쥴링과 실행 중인 작업을 좀 더 좋은 자원으로 이주시켜 결함을 포용하는 적응적 수행, 파일 전송의 오버헤드를 줄이기 위해 작업 간의 공통적 파일을 재사용하는 기법을 스케쥴링에 적용하였다. 관련연구[4]는 그리드의 이질적인 자원 환경 하에서 서비스 제공자와 사용자의 협약(agreement)을 통해 자원을 사용하도록 하는 SLA(Service Level Agreement)를 제안하였다.

관련연구[1][2]의 배치 방식은 스케쥴링 시 작업 실행 대기 시간을 중간에 계속 갱신하여 ECT를 변경한다는 점에서 동적인 요소가 있지만 실행 중에는 변경되지 않으므로 정적 스케쥴링이다. 관련연구[1][2][3][4] 뿐만 아니라 최근의 파라미터 스윕 어플리케이션에서의 스케쥴링 방식은 서비스 기반 그리드 컴퓨팅 환경에서 선형 워크플로우를 고려하고 있지 않으며 동적인 그리드 환경에서 가용성의 변화에 효과적으로 대처하고 있지 못해 성능 보장을 지원하지 못한다.

3. 적응적 스케쥴링을 위한 워크플로우 관리 시스템 구조

서비스 기반의 그리드 컴퓨팅에서 클라이언트의 그리드 어플리케이션을 수행하기 위해서는 그리드 어플리케이션을 구성하는 워크플로우의 수행 방법과 사용자가 요구하는 서비스의 질을 보장하는 시스템이 요구된다. <그림 1>은 본 논문에서 제안하는 스케쥴링 기법을 위한 그리드 워크플로우 관리 시스템의 구조이다.



<그림 1> 그리드 워크플로우 관리 시스템 구조

<그림 1>의 서비스 제공자 레벨에서 각각의 제공 서비스는 측정 서비스를 가지게 되며, 측정 서비스는 다음 장에서 설명하는 서비스 모델링 항목들을 측정하여 워크플로우 관리자 레벨에 있는 정보 서비스를 통해 측정치가 수집된다.

워크플로우 스케쥴링 서비스는 클라이언트로부터 요청 받은 워크플로우에 가장 적합한 서비스를 선택하기 위하여 SLA와 정보 서비스를 이용하여 분석하고 최적의 서비스 자원을 선택한다. 그리고 그리드 미들웨어의 GRAM을 이용하여 자원을 할당받아 서비스를 실행하며 GridFTP를 이용하여 파일을 전송한다.

평가 서비스는 SLA에 기술된 성능 기준에 부적합 여부를 판단하기 위하여 평가 메트릭을 생성하고 정보 서비스를 이용하여 QoS 결과를 통지 받는다. 통지 결과에 따라 결함 발생 여부를 결정하고 결함 발생 시 워크플로우 스케쥴링 서비스에 통지하여 재스케쥴링 되도록 한다.

4. 적응적 스케쥴링 기법

본 장에서는 서비스 기반 그리드 컴퓨팅 환경에서 선형 워크플로우를 수행할 때 사용자의 QoS를 보장하기 위해서 서비스 성능을 예측하는 방법과 최대 흐름 알고리즘을 이용한 최적의 자원 사용률을 보장하는 기법을 제시한다.

4.1 서비스 수행 성능 예측

그리드 컴퓨팅 환경에서 특정 어플리케이션의 수행 시간은 계산 시간과 통신 시간으로 계산된다. 어떤 그리드 어플리케이션이 서비스 A와 B로 구성되고 서비스 A는 B와의 통신 전에 완료된다고 가정할 때, 이 어플리케이션의 수행 시간은 다음과 같이 정의될 수 있다.

$$수행\ 시간 = 계산시간(A) + 통신시간(A, B) + 계산\ 시간(B)$$

그리드 서비스의 수행 시간을 예측하기 위해서는 그리드 서비스들의 특성들을 기술하는 그리드 서비스 성능 모델링에 따라 SLA에서 정의된 필요한 서비스를 탐색하고 최적의 서비스를 선택하여 워크플로우를 구성하여야 한다. 또한 서비스 기반의 그리드 컴퓨팅에서는 그리드 자원 자체의 물리적인 자원의 스케쥴링 뿐만 아니라 서비스 스케쥴링에서 서비스의 질을 보장하는 QoS를 지원해야 한다. 따라서 본 논문에서는 다음과 같은 정보를 고려하여 그리드 서비스들의 수행 성능을 모델링 한다.

- 서비스 정적 자원 모델링 : 기본적으로 정보가 고정적인 CPU, 메모리, 하드 디스크, 네트워크 대역폭을 그리드 서비스가 사용하는 공유 자원으로 본다.
- 서비스 동적 자원 모델링 : 서비스의 수행 성능은 서비스를 수행하는 자원들의 가용성(capability)에 직접적으로 영향을 받기 때문에 서비스 모델링을 위한 자원의 특성들은 CPU 속도와 사용량, 가용 메모리 크기, 남은 디스크 공간, 노드간 네트워크 대역폭 등의 물리적 자원의 가용성을 기술하여 이용한다. 또한 그리드 스케쥴링은 현재의 자원 상태 뿐만 아니라 미래의 자원 상태와 서비스 이용 패턴을 고려해야 하므로 각 그리드 서비스의 특징인 서비스 예약 상태, 이용 빈도, 서비스 처리율을 기술한다.

그리드는 서비스 노드의 참여와 이탈이 자유롭기 때문에 새로 추가되는 서비스의 성능을 예측하여 동적으로 그리드 서비스 스케쥴링에 사용할 수 있어야 한다. 이를 위해 본 논문에서는 그리드 서비스의 성능을 통계적 기법을 이용하여 예측하도록 한다. 통계 기법 중 회귀분석은 한 변수가 다른 변수들에 의해 설명되고 그로부터 어떻게 예측될 수 있는지를 알아보기 위한 통계적 방법이다. 결과가 되는 변수는 종속변수(y)가 되며 원인이 되는 변수는 독립변수(x)가 되어 x→y 의 관계로 표시할 수 있다. 즉 다음과 같은 식으로 표현할 수 있다.

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i,$$
$$(i = 1, \ldots, n,\ \beta_0 : 상수,\ \beta_1 : 회귀계수,\ \varepsilon : 에러율)$$

회귀분석 결과 회귀계수(β_1)로 두 변수 간 연관성의 정도를 측정할 수가 있다. 즉, VO(Virtual Organization)에 추가된 새 노드의 서비스에 대한 성능이 종속변수이고 자원이 독립변수라고 한다면 기존에 측정된 회귀계수에 의해 성능을 예측할 수 있다. 위의 예는 독립변수가 하나인 단순회귀분석이고 실제로 서비스의 성능을 예측하기 위해서는 독립변수가 여러 개인 다중회귀분석을 사용한다. 회귀분석을 사용하기 위해서는 그리드 서비스 모델링에서 정의한 각 항목을 독립변수로 정의한다.

4.2 최대 흐름 알고리즘을 이용한 작업 부하 조정

서비스 수행 시간을 최소화하기 위해서는 서비스를 수행할 계산 노드와 각 서브 작업마다 필요한 데이터베이스 서버나 파일 서버를 결정할 때 데이터 전송 시간과 작업 연산 시간을 최소화할 수 있는 자원의 선택이 중요하다. 그러나 서비스 수행 시간을 최소화하기 위해서는 스케쥴링 알고리즘은 선택된 자원에 대해 최적으로 사용할 수 있도록 조정해 주어야 한다. 본 논문에서는 네트워크 내의 시작 지점에서 목표 지점까지 흐를 수 있는 최대량을 구하는 최대 흐름 알고리즘을 이용하여 다음과 같은 동적 스케쥴링을 제안한다.

[단계 1] 사용자가 요구하는 선형 워크플로우를 입력 받고 SLA에 따라 서비스를 선택한다.

[단계 2] 선택된 서비스에 대해 워크플로우를 만들고 그리드 성능 모델링을 이용하여 간선의 용량을 구한다. 만약 이전의 성능 모델링의 서비스와 서비스 간의 용량이 이미 서비스 사용 이력으로 남아있다면 그것을 사용한다.

[단계 3] 최대 흐름 알고리즘에 따라 최대 흐름을 구한다.

[단계 4] SLA 요구에 맞지 않는 성능 저하 시 서비스의 현재의 모니터링된 성능을 용량으로 지정한 후 [단계 3]을 반복한다. 급격한 성능 변화로 발생하는 문제를 방지하기 위해 다음 식을 이용하여 성능을 평가한다.

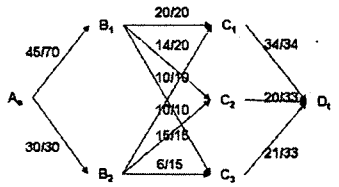
$$T_{t+1} = (1-\alpha) * T_{t-1} + \alpha * T_t$$

(T : 성능, t : 시간, α : 상수)

재스케쥴링하기 이전의 흐름과 이후의 흐름을 비교하여 흐름이 증가되었으면 감소된 흐름의 작업을 맡을 수 있도록 감소된 서비스로부터 작업을 이주시킨다. 그리고 총 흐름 양이 감소될 경우 새로운 서비스 노드를 추가하여 [단계 2][단계 3]을 반복한다.

[단계 5] 워크플로우 수행이 완료되었을 때 서비스와 서비스간의 용량을 서비스 사용 이력으로 남긴다.

위의 알고리즘에 따른 서비스 A, B, C, D로 이루어진 그리드 어플리케이션의 선형 워크플로우를 [단계 1]에 따라 가용할 수 있는 서비스를 선택하면 <그림 2>와 같은 워크플로우를 생성할 수 있다. 워크플로우의 간선의 크기는 그리드 서비스 성능 모델링을 통해 추상화시켜 구한다. 계속해서 [단계 2]와 [단계 3]을 수행하면 <그림 2>의 간선처럼 흐름/용량이 표시된다. <그림 2>의 표는 서비스 순서와 흐름, 즉 처리할 수 있는 양을 나타낸다. 즉 서비스 순서대로 서비스 구성을 한 후 서비스를 수행하였을 경우 흐름만큼의 성능을 보장하는 것이다. 이를 통해 예상 처리량, 병목 지점 등에 대한 QoS에 대해 예측이 가능하다.

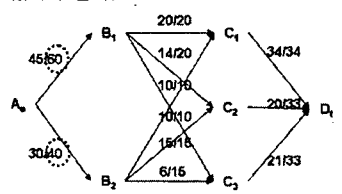


서비스 순서	플로우
$A_0B_1C_2D_1$	10
$A_0B_1C_1D_1$	20
$A_0B_1C_3D_1$	15
$A_0B_2C_1D_1$	14
$A_0B_2C_2D_1$	10
$A_0B_2C_3D_1$	6

<그림 2> 최대 흐름 알고리즘을 이용한 스케쥴링

4.3 적응적 스케쥴링

그리드에서 동적 스케쥴링을 제공하기 위해서는 처음 스케쥴링이 이루어져 작업이 시작된 후에도 작업들의 수행 상태를 모니터링하여 수행 성능이 떨어질 경우 수행 성능 보장을 위해 재스케쥴링을 제공할 수 있어야 한다.



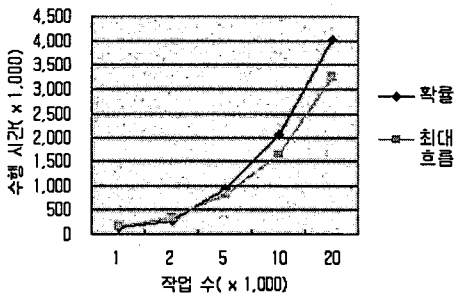
서비스 순서	이전 플로우	이후 플로우
$A_0B_1C_2D_1$	10	10
$A_0B_1C_1D_1$	20	20
$A_0B_1C_3D_1$	15	15
$A_0B_2C_1D_1$	14	14
$A_0B_2C_2D_1$	10	10
$A_0B_2C_3D_1$	6	15

<그림 3> 최대 흐름 알고리즘을 이용한 스케쥴링

<그림 3>은 [단계 4]를 수행한 결과이다. 즉, <그림 3>의 점선 원안의 용량이 변하는 것과 같이 SLA의 요구사항에 맞지 않는 성능 저하가 발생하면 서비스 구성을 다시 할 수 있도록 재스케쥴링하는 것이다. <그림 3>의 표에서 볼 수 있듯이 결과적으로 총 흐름 양이 증가되는 경우를 볼 수도 있으나 만약 총 흐름 양이 감소할 경우 새로운 서비스 노드를 추가하는 방법을 쓸 수 있다.

5. 실험

본 논문에서는 그리드 컴퓨팅 환경에서 스케쥴링 알고리즘을 개발하고 평가하기 위해 널리 사용되고 있는 SimGrid 툴킷을 이용하여 모의 실험하였다. 실험용 그리드 어플리케이션은 데이터를 조회하고 데이터를 다운받아 데이터 처리 후 저장하는 총 3 개의 서비스를 이용하도록 하였고 각 서비스마다 가용할 수 있는 호스트 개수는 각각 5, 15, 1개로 구성하였다.



<그림 4> 모의 실험 결과

확률에 의한 방법은 성능이 좋은 호스트에 작업이 많이 가도록 하는 스케쥴링 방법이고 최대 흐름 알고리즘에 의한 방법은 본 논문에서 제시한 방법을 사용하였다. <그림 4>의 실험 결과에서도 알 수 있듯이 작업 수가 적을 시에는 두 방법의 수행 시간이 큰 차이가 나지 않지만 작업 수가 많아질수록 본 논문에서 제시한 방법이 더 효율적임을 알 수 있다.

6. 결론 및 향후 과제

본 논문에서는 서비스 기반의 그리드 컴퓨팅 환경에서 동적인 자원 환경을 고려하여 선형 워크플로우를 병렬 수행할 수 있는 적응적 스케쥴링 방식을 제안하였다. 이를 위해 워크플로우 관리자 시스템 구조와 그리드 서비스 성능 모델링, 이를 이용한 성능 예측 방법, 그리고 최대 흐름 알고리즘을 이용한 적응적 스케쥴링 방법을 설명하였다.

향후 연구과제로는 분산 스케쥴링 기법을 연구하여 본 논문에서 제시한 선형 워크플로우를 복잡한 워크플로우에 적용하는 기법 연구가 요구된다.

참고문헌

[1] Muthucumaru Maheswaran, Shoukat Ali, Howard Jay Siegel, Debra Hensgen, and Richard F. Freund, *"Dynamic Matching and Scheduling of a Class of Independent Tasks onto Heterogeneous Computing Systems"*, Proceedings of the 8th Workshop on Heterogeneous Computing Systems (HCW '99), San Juan, Puerto Rico, Apr. 1999.

[2] Casanova, H., Legrand, A., Zagorodnov, D., and Berman, F., *"Heuristics for Scheduling Parameter Sweep Applications in Grid Environments"*, Proceedings of the 9th Heterogeneous Computing Workshop (HCW'00), pp. 349-363, 2000.

[3] Eduardo Heudo, Ruben S. Montero, Ignacio M. Lorente, *"Experiences on Adaptive Grid Scheduling of Parameter Sweep Applications"*, Proceedings of the 12th Euromicro Conference on Parallel Distributed and Network-Based Processing(EUROMICRO-PDP'04), 2004.

[4] Karl Czajkowski, Ian Foster, Carl Kesselman, *"Agreement-Based Resource Management"*, Proceedings of The IEEE, Vol. 93. No 3, March, 2005.