그리드 컴퓨팅 시스템에서 최적 자원 선택 브로커 설계

진성호*O 정광식** 이화민* 이대원* 유헌창* 정순영*
* 고려대학교 대학원 컴퓨터교육과 ** 고려대학교 대학원 컴퓨터학과
. {wingtop^O, zelkova, ldw1996, yuhc, jsy}@comedu.korea.ac.kr, k.chung@ucl.ac.uk

A Design of Optimal Resource Selection Broker in Grid Computing Systems

Sung Ho Chin Kwang Sik Chung Hwa Min Lee Dae Won Lee Heon Chang Yu Soon Young Jung
* Dept. of Computer Science Education, Korea University ** Dept. of Computer Science, Korea University

요 약

그리드 컴퓨팅은 광범위 분산 컴퓨팅 시스템(wide area distributed computing system)으로, 고성능의 유휴 컴퓨팅 자원을 서로 공유하여 효율적으로 작업을 수행하는 것을 목적으로 한다. 그리드 컴퓨팅에서 사용자가 요구하는 자원의 검색, 선택, 할당하는 문제는 시스템 성능에 큰 영향을 미친다. 그리드 컴퓨팅을 지원하는 대표적인 미들웨어인 글로버스(Globus Toolkit)에서는 위와 같은 과정들이 사용자에 의해 수동적으로 이루어지며, 검색된 후보 자원의 최적 선택 방법은 제공하지 않고 있다.

본 논문에서는 글로버스에서 사용자의 요구에 의해 검색된 후보 자원들 중 최적화된 자원 선택과 할당 요청을 담당하는 최적 자원 선택 브로커를 설계하였다. 이 브로커는 유전자 알고리즘을 이용하여 최적 자원을 선택하므로 사용자의 임의적 자원 선택으로 인한 시스템의 성능 저하를 막아준다. 자원 검색, 선택, 할당 요청이 하나의 브로커에서 이루어짐으로써 작업 수행 시 발생하는 사용자의 불필요한 관여를 막아 작업 수행에 대한 편의성을 제공한다.

1. 서 론[1)]

기존의 분산 컴퓨팅 시스템에 대한 연구가 활발히 이루어지면서 1990년 후반 미국의 슈퍼 컴퓨팅 센터를 중심으로 고성능의 분산 컴퓨팅 인프라를 구축하는 과정에서 차세대 인터넷으로 연결하는 그리드(grid)라는 개념이 등장하였다. 그리드는 한번에 한 자원에만 연결할 수 있는 웹과는 달리 여러 자원과 연결하여 작동하는 네트워크 구조를 말한다. 그리드 컴퓨팅은 지리적으로 분산되어 있는 고성능의 컴퓨팅 자원을 네트워크로 연결하여 사용함으로써 단일 슈퍼 컴퓨터와 같은 성능을 발휘할 수 있는 시스템이다. 그리드 컴퓨팅 개념에서 자원공유를 위한 단위로써 가상 조직(VO : Virtual Organization)을 정의한다. 가상 조직내의 구성원들은 자신의 컴퓨팅 자원과 상대방의 컴퓨팅 자원을 상호 공유할 수 있다.

그리드의 한 종류인 계산 그리드(computational grid)는 지리적으로 분산되어 있는 많은 자원들을 연결하여 큰 로드의 작업을 수행한다. 지리적으로 분산되어 있는 다양한 컴퓨팅 자원을 이용하여 작업을 수행할 때, 그리드 상의 수많은 자원들 중에서 사용자가 작업 수행을 위해 요구하는 자원을 적절히 선택하는 것은 중요한 문제가 된다. 자원 선택의 과정에서 그리드 환경이 원거리 통신망을 기반으로 하고 있다는 점은 자원선택 문제에 큰 영향을 주는 요소가 된다. 근거리 통신망을 기반으로 하는 기존의 분산 시스템과는 달리 그리드 컴퓨팅은 네트워크의 지연이나 대역폭의 변화에 따라 컴퓨팅 성능이 변화하게 된다. 또한, 작업 수행을 위해 선택하고자 하는 프로세서의 성능이나 유휴상태 등도 컴퓨팅 성능에 영향을 주는 중요한 요인이다. 따라서, 컴퓨팅 성능에 영향을 미칠 수 있는 요소들을 고려하여 사용자가 요구하는 자원을 검색, 선택, 할당하는 문제는 그리드 컴퓨팅에서 중요한 문제가 된다.

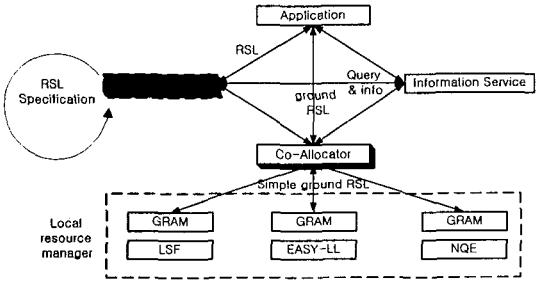
하지만, 그리드 컴퓨팅을 지원하는 대표적인 미들웨어인 글로버스[1]에서는 검색되어진 후보 자원에 대한 선택과 할당에 관한 문제를 사용자가 수동적으로 해결해야 한다. 사용자에 의한 수동적인 자원 선택과 할당 방식은 처리하려는 작업의 크기가 커지거나 사용하려는 후보 자원의 양이 증가하면 작업 처리 과정시 사용자에게 큰 불편을 주게된다. 그러므로, 본 논문에서는 유전자 알고리즘[2]을 이용하여 사용자의 작업을 처리하기 위한 최적의 자원을 선택하고 자동적으로 할당요청을 하는 최적 자원 선택 브로커를 설계하였다. 최적 자원 선택 브로커는 작업의 수행에서 사용자의 관여를 제거하여 편의성을 제공한다.

본 논문의 구성은 다음과 같다. 2장에서 기존의 자원선택 기법과 현재 그리드에서 사용되고 있는 자원관리 구조를 소개한다. 3장에서는 기존 연구들의 문제점을 해결할 수 있는 브로커를 설계하고 그 구성요소들의 역할에 대해서 소개한다. 4장에서는, 유전자 알고리즘을 이용한 최적 자원 선택 기법을 소개한다. 5장에서는 모의실험을 통해 유전자 알고리즘을 이용한 최적 자원 선택 기법의 성능을 검증한다. 끝으로, 6장에서는 본 연구의 결론과 향후 연구과제에 대해 논의한다.

1) * 본 과제(결과물)는 정보통신부의 정보통신학술기초연구지원사업(정보통신연구진흥원)으로 수행한 연구결과입니다.

2. 관련연구

그리드의 자원 관리 구조는 <그림 1>과 같이 자원의 정보 검색을 지원하는 정보 서비스(information service), 여러 가지 자원을 동시에 할당하여 주는 동시 할당자(Co-Allocator), 사용자의 작업 수행 요청을 대신하는 브로커(Broker)와 지역자원관리를 담당하는 GRAM으로 구성된다[3]. <그림 1>에서 사용자가 작업을 수행하기 위해서는 두 가지 방법을 이용할 수 있다. 첫 번째 방법은 사용자가 직접 작업에 필요한 자원을 정보 서비스를 이용하여 검색하고 검색된 정보를 바탕으로 적절한 지역 자원 관리자에게 자원 할당을 요청하는 방법이다. 두 번째 방법은 브로커를 이용하는 방법으로 사용자가 작업을 수행하기 위해 필요한 자원을 기술하여 브로커에게 전달하면 브로커는 정보서비스를 이용해 요청에 적합한 자원을 선택하고 할당 요청을 한다. 이 방법은 첫 번째 방법에 비해 작업 수행과정이 간편한 장점이 있다. 하지만, 현재 글로버스에서는 자원의 검색, 선택, 할당요청을 담당하는 브로커를 지원하지 않기 때문에 글로버스에서 작업 수행 방법은 첫 번째 방법과 같다.

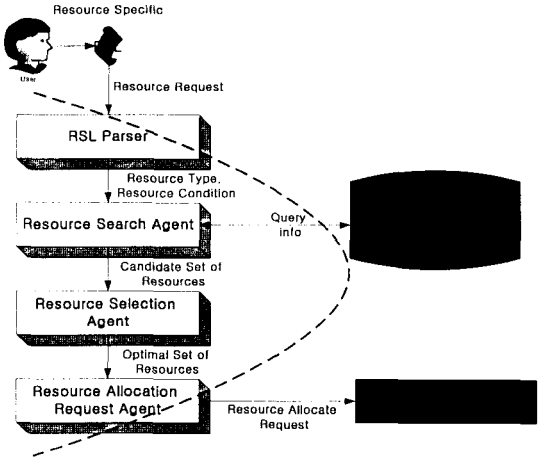


<그림 1> 그리드의 자원 관리 구조

또한, 기존 연구[3, 4, 5]에서는 사용자의 요구를 만족하는 자원의 수가 사용자가 필요로 하는 자원의 수보다 클 때 발생하는 최적 자원 선택문제에 대한 해결책이 제시되지 않았다. 따라서 본 논문에서는 글로버스에서 사용할 수 있는 최적 자원 선택 브로커를 설계한다. 최적 자원 선택 브로커는 유전자 알고리즘을 이용하여 후보 자원에 대한 최적 자원 선택 문제를 해결하고 글로버스를 이용한 작업 수행을 더욱 간편하게 해 준다.

3. 최적 자원 선택 브로커

본 논문에서 설계한 최적 자원 선택 브로커는 그리드 환경에서 수행되는 작업을 위한 자원의 검색, 선택, 할당 요청을 담당한다. 그리드 환경에서 작업수행을 원하는 사용자는 자원 할당 과정에 대해 관여하지 않고 최적 자원 선택 브로커에게 작업에 대한 자원 할당을 요청한다. 최적 자원 선택 브로커가 제공하는 작업 수행의 편의성으로 인해 처리 대상의 작업양이 많은 그리드 환경에서 최적 자원 선택 브로커를 이용하면 더욱 효과적으로 작업을 수행할 수 있다. 특히, 최적 자원 선택 브로커는 자원에 대한 사용자 임의 선택으로 인해 발생될 수 있는 작업수행 성능 저하를 막아 준다. 자원에 대한 사용자 임의선택은 작업 수행환경을 고려하지 않아 최적 자원을 선택하지 못하므로 작업 수행 능력을 저하시키는 요인이 된다. 최적 자원 선택 브로커는 자원 선택 과정에서 자원에 대한 사용자 임의 선택 대신 유전자 알고리즘을 이용하여 최적 자원 집합을 시스템이나 사용자에게 제공한다. 최적 자원 집합은 사용자의 요구를 만족하는 후보 자원 중에서 작업의 효율적 수행 환경을 고려하여 선택된 자원 집합이다.



<그림 2> 최적 자원 선택 브로커의 구조

이 논문에서 설계한 최적 자원 선택 브로커의 전체적인 구조는 <그림 2>와 같다. <그림 2>에서 최적 자원 선택 브로커는 사용자의 요구를 분석하는 RSL Parser, 자원의 정보를 검색하는 자원 검색 에이전트(Resource Search Agent), 검색된 자원들 중 최적 자원을 선택하는 자원 선택 에이전트(Resource Selection Agent), 그리고 선택된 자원 정보를 바탕으로 자원 할당을 요청하는 자원 할당 요청 에이전트(Resource Allocate Request Agent)로 구성되어 있다. 최적 자원 선택 브로커는 글로버스에서 사용하는 구성요소들과 결합하여 사용할 수 있도록 설계하였다. 자원 검색을 담당하는 자원 검색 에이전트는 글로버스에서 자원 정보 검색을 지원하는 MDS(Metacomputing Directory Service)[1]와 수행할 수 있도록 설계하였고 작업에 대한 자원 할당을 요청하는 자원 할당 요청 에이전트는 글로버스에서 사용하고 있는 동시 자원 할당자인 DUROC(Dynamically Updated Request On-line Co-allocator)[6]과 수행 될 수 있도록 설계하였다.

이 논문에서 설계한 최적 자원 선택 브로커의 구성 요소들의 자세한 역할은 다음과 같다.

1) RSL Parser

RSL(Resource Specific Language)은 글로버스에서 구성요소들 간의 통신을 위해 사용되는 언어이다. 사용자는 RSL을 이용하여 자원을 검색하거나 자원 할당 요청을 할 수 있다. 사용자는 최적 자원 선택 브로커를 이용하여 작업을 수행하기 위해 작업에 필요한 자원의 종류, 요구하는 자원의 상태, 자원의 수 등을 RSL을 이용하여 기술하고 최적 자원 선택 브로커에게 넘겨준다. 최적 자원 선택 브로커에게 사용자의 요구가 기술된 RSL이 전달되면, RSL Parser는 사용자가 요구하는 자원에 관한 정보를 추출하여 자원 검색 에이전트에게 전달한다.

2) 자원 검색 에이전트(Resource Search Agent)

자원 검색 에이전트는 RSL Parser를 통해 얻어진 사용자의 요구 자원에 대한 정보를 이용하여 글로버스에서 자원 정보 검색을 지원하는 MDS를 통해 적절한 자원을 검색한다. 그리고 사용자의 요구를 만족하는 자원들을 수집하여 후보 자원 집합(*set of candidate resources*)을 생성한다. 후보 자원 집합은 사용자의 요구를 만족하는 그리드상의 모든 유휴 자원으로써 사용자가 요구한 자원의 양보다 크거나 작은 경우를 허용한다. 자원 검색 에이전트는 수집한 후보자원 집합을 자원 선택 에이전트에게 전달한다.

3) 자원 선택 에이전트(Resource Search Agent)

자원 선택 에이전트는 후보자원 선택 문제 해결을 담당한다. 자원 검색 에이전트에서 넘어온 후보자원 집합을 바탕으로 작업이 가장 효율적으로 수행될 수 있는 최적 자원 집합(set of optimal resources)을 찾기 위해 유전자 알고리즘이 수행된다. 유전자 알고리즘을 이용하여 최적 자원 집합이 생성되고 자원 할당 요청 에이전트에게 최적 자원 집합이 전달된다. 최적 자원 선택을 위한 유전자 알고리즘은 4절에서 자세하게 제시한다.

4) 자원 할당 요청 에이전트(Resource Allocation Agent)

자원 할당 요청 에이전트는 자원 선택 에이전트에서 넘어온 최적 자원 집합의 원소 자원들에 대한 할당을 위해 사용자의 RSL을 좀더 명확한 자원이 기술된 ground RSL로 변환한 후, 자원 동시 할당자에게 자원 할당을 요청한다.

4. 유전자 알고리즘을 이용한 최적 자원 선택 기법

그리드 환경에서 수행되는 대부분의 작업들은 많은 양의 컴퓨팅 자원을 요구하는 대규모 작업이다. 그리드 컴퓨팅은 자원 공유를 통해 대규모 작업에 대한 병렬처리를 지원한다. 그리드를 이용한 병렬처리에서 작업 분할과 분할된 작업의 수행을 위한 자원 선택 문제는 작업수행 능력에 큰 영향을 준다. 자원의 컴퓨팅 성능을 고려한 적절한 자원 선택 뿐만 아니라 작업의 전달이나 결과 값의 전달과정에서 발생하는 네트워크의 지연이나 대역폭의 변화 등도 작업수행 능력에 영향을 미친다. 원거리 통신망을 기반으로 수행되는 그리드 컴퓨팅은 근거리 통신망을 기반으로 수행되는 기존의 분산 컴퓨팅보다 네트워크의 상태 변화에 따라 작업 수행 성능이 크게 변화된다.

본 논문에서 설계한 최적 자원 선택 브로커의 자원 선택 에이전트는 자원의 컴퓨팅 성능이나 네트워크 상태를 고려해 병렬 작업이 가장 효율적으로 수행될 수 있는 자원을 선택한다. 최적 자원 선택 브로커는 최적 자원 선택을 위해서 유전자 알고리즘을 이용한다. 유전자 알고리즘은 기계학습에 사용되는 알고리즘으로써 많은 수의 탐색공간을 가지는 문제해결에 적당하며 설계의 용의성과 최적화 문제 해결에 높은 성능을 발휘한다. 그러나, 유전자 알고리즘을 이용하여 정확한 해를 찾기 위해서는 적합도의 올바른 정의가 필요하다. 이를 위해 본 논문에서는 작업의 양과 대상 자원의 컴퓨팅 성능을 고려한 작업의 수행시간 예측을 이용하여 적합도를 정의하고 최적 자원 선택을 위한 유전자 알고리즘을 구현하였다.

다음은 최적 자원 선택 브로커가 이용하는 유전자 알고리즘에서 사용되는 요소들을 정의한다.

① 유전자의 길이(length of genetic) :
RR : RSL parer에서 구한 사용자가 요구하는 자원 집합, $|RR|=n$

② 유전자의 표현(representation of genetic) :
RC : 자원 검색 에이전트에서 전달된 후보자원 집합
$G_n \Rightarrow {}^{\forall}S$, $S \subset RC$ and $|S|=n$, $|RC|=rc$, $G_n = \{r_0, r_1, \cdots, r_{n-1}\}$

③ 가설공간(hypotheses) : $2^A=\{X|X\subset A\}$, $H_{{}_{\pi}C_n} = 2^{\bigcup^{\pi C_n} G_n}$

④ 개체군(Population) : $P \Rightarrow {}^{\forall}SP$, $SP \subset H$ and $|SP|=p$, p:인구수

⑤ 적합도 함수(fitness function) : $g_n \in P$

$r_n \xrightarrow{executiontime(x)} excutiontime(r_n)$,

$Fitness(g_n) = Max\{(executiontime(r_0), executiontime(r_1),$
$\cdots, executiontime(r_{n-1}))$ $\quad excutiontime(r)$: r에서의 수행시간

⑥ 선택(selection) :
$PS = P \cup Mutation(P) \cup Crossover(P)$, $Select(PS) = P'$, $|PS| = ps$

⑦ 교배(crossover) : $g_i, g_j \in P$, c : 교배점
$Crossover(g_i, g_j) = g_k$, $g_k = \{r_0 \cdots r_c \in g_i, r_{c+1} \cdots r_{n-1} \in g_j\}$

⑧ 돌연변이(mutation) : r_x, r_y : 대립 유전자, $g_i \in P$,
$Mutation(g_i) = g_i'$,
$g_i' = \{r_0, r_1, r_x, r_{x+1}, \cdots, r_y, r_{y+1}, \cdots, r_{n-1} \mid r_x, r_y \in RC, r_x, r_y \notin g_i\}$

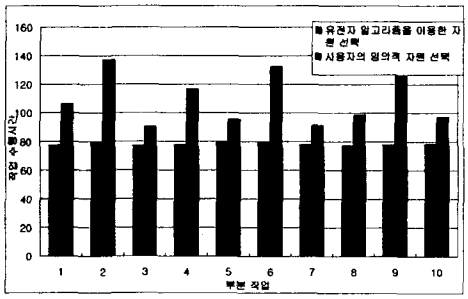
<그림 3>은 새롭게 정의한 요소들을 이용하여 유전자 알고리즘을 이용한 최적 자원 선택 기법이다.

```
GA For Select Set Of Optimal Resources
( RC, TT , p , c , m , n ) {
   P=φ ;
   Initialization ( p, RC, P);
   Evaluation ( p, P) {
   while ( MAX(Fitness( g_n in P ) ) > TT ) {
         PS=φ ;
         Select ( p, P , PS );
         Crossover ( c, P, PS );
         Mutation ( m, P, PS );
         update PS to P;
         Evaluation ( p, P);
   }
for ( 1 to p )
   if ( Rank( g_n in P) == 1 )
        then g_n to Resource Allocation Request Agent ;
}
Initialization ( p, RC , P) {
   for ( 1 to p ) {
        generate new g_n∈H from RC ;
        if ( g_n∉P ) then add g_n to P ;
   }
}
Evaluation ( p, P ) {
    for ( 1 to p ) compute Fitness( g_n) ;
}
Select ( p, P , PS ) {
   for ( 1 to p )
        compute Rank( g_n ) using Fitness( g_n) ;
   for ( 1 to p )
        if ( 1≤ Rank( g_n) ≤p )
            then add g_n to PS ;
}
Crossover ( c, P, PS ) {
   for ( 1 to (c · p)/100 ) {
        select (g_i∈P,g_j∈P) from P at random ;
        find crossover pointer c_p( 1≤c_p≤n) at random ;
        for ( 1 to c_p )
             add r_n from g_i to g_k
        for ( n-c_p to n )
             add r_n from g_j to g_k
       if ( g_k∉PS ∧ ∀(r_n∈g_k,r_m∈g_k)→r_n≠r_m )
             then add g_k to PS;
   }
}
Mutation ( m, P, PS ) {
   for ( 1 to (m · p)/100 ) {
        select g_m from P at random ;
        for ( 1 to n )
             if ( Random( 1, 0 ) == 1 ) then
                  replace r_n by r_m( r_m∈RC,r_m≠r_n ) ;
        if ( g_m∉PS ∧ ∀(r_n∈g_m,r_m∈g_m)→r_n≠r_m ) ;
             then add g_m to PS ;
   }
}
```

<그림 3> 유전자 알고리즘을 이용한 최적 자원 선택 기법

5. 모의 실험

이 논문에서 설계한 최적 자원 선택 브로커에서 유전자 알고리즘을 이용한 최적 자원 선택 기법의 성능을 검증하기 위해 모의 실험 하였다. 모의 실험 환경은 1000개의 가상 그리드 노드를 생성하여 이용하였다. 각 노드들은 프로세서의 성능, 프로세서의 유휴상태, 각 노드들간의 통신 대역폭에 관한 정보와 같은 작업수행 능력에 영향을 미치는 요소를 가지고 있다. 모의 실험에서 가정하고 있는 작업은 임의적으로 여러 개의 부분 작업으로 나누어질 수 있는 병렬 작업이다. 나누어진 부분작업들은 다른 부분작업과 독립적으로 수행하는 것으로 가정하였다. 이 가정에 따라서 한 병렬작업이 수행되는 시간은 여러 개의 부분 작업의 수행시간 중 가장 긴 것으로 결정된다.



<그림 4> 부분 작업의 수행 시간

<그림 4>는 모의실험에서 유전자 알고리즘을 이용한 최적 자원 선택 기법과 자원에 대한 사용자 임의선택에 대해 부분 작업의 수행 시간을 비교한 것이다. <그림 4>에서 자원에 대한 사용자 임의 선택에 의해 수행되는 부분 작업들은 부분 작업들의 수행 시간이 심한 차이를 보인다. 부분 작업의 수행 환경을 고려하지 않은 자원 선택으로 인해 발생하는 부분 작업 사이의 수행시간 차이는 전체 작업의 수행에 대해서 성능 저하를 가져온다. 그러나 유전자 알고리즘을 이용한 최적 자원 선택 기법을 통해 작업을 수행하였을 때는 <그림 4>에서 부분 작업들의 수행 시간이 비슷해지므로 전체 작업의 수행에 대한 성능 저하를 막아줄 수 있다.

6. 결론 및 향후 연구과제

본 논문에서는 기존의 글로버스에서 지원하지 않았던 최적 자원 선택 브로커를 설계하였다. 최적 자원 선택 브로커는 자원 선택 과정에서 발생하는 후보 자원들에 대한 최적 자원 선택 문제를 유전자 알고리즘을 이용하여 해결하였다. 유전자 알고리즘을 이용한 최적 자원 선택 기법은 작업의 양이나 자원의 컴퓨팅 성능을 고려하므로 자원에 대한 사용자 임의 선택으로 인해 발생 될 수 있는 작업 수행 성능의 저하를 막아준다. 향후 연구과제로는 글로버스를 이용한 실제적인 구현과 최적 자원 선택 브로커의 성능을 측정하기 위한 다양한 실험이 요구된다. 그리고 유전자 알고리즘을 이용하였을 때 발생되는 부하를 줄이는 방법에 관한 연구가 필요하다.

참고문헌

[1] I. Foster and Carl Kesselman, Globus : A metacomputing infrastructure Toolkit, Intl J, Supercomputer Application 11(2):115-128, 1997

[2] Tom N. Mitchell, Machine Learning (pp 249-270), McGRAW HILL 1997

[3] Karl Czajkowski, Ian Foster, Nicholas Karonis, Carl Kessleman, Stuart Martin , A Resource Management Architecture for Metaccomputing Systems, Proc. IPPS/SPDP '98 Workshop on Job Scheduling Strategies for Parallel Processing, pg. 62-82, 1998

[4] Chuang Liu, Lingyou Yang, Ian Foster, Dave Angulo , Design and Evaluation of a Resource Selection Framework for Grid Application , Proceedings of IEEE International Symposium on High Per-Distributed Computing (HPDC-11), Edinburgh, Scotland, July 2002.

[5] Chapin. s.j , Katranntos, D, Karpovish, j and Grinshaw , Resource Management in Legion , (JSSPP 99

[6] K. Czajkowski, I. Foster, and C. Kesselman, Resource Co-Allocation in Computational Grids, Proceedings of the Eighth IEEE International Symposium on High Performance Distributed Computing (HPDC-8), pp. 219-228, 1999.