韓國컴퓨터情報學會 論文誌
第16卷 第7號, 2011. 7.

2011-16-7-1-2

그리드 컴퓨팅을 위한 NSGA-II 기반 다목적 작업 스케줄링 모델

김 솔 지* 김 태 호* 이 홍 철**

Multi-Objective Job Scheduling Model Based on NSGA-II for Grid Computing

SolJi Kim* TaeHo Kim* HongChul Lee **

요 약

그리드 컴퓨팅은 지리적으로 분산된 이기종의 컴퓨팅 자원들을 상호 연결하고 공유하여 가상의 고성능 컴퓨팅 시스템을 구성함으로서 대용량의 컴퓨팅 연산 등을 수행하는 차세대 컴퓨팅 기술이다. 이러한 그리드 컴퓨팅의 성능을 극대화하기 위해서는 효율적으로 작업을 자원에 할당하는 작업 스케줄링 기법이 필요하다. 따라서 작업 총 완료시간 등을 고려한 작업 스케줄링 기법에 대한 많은 연구가 진행되었다. 그러나 작업 스케줄링에 있어서 자원의 사용에 따른 자원 비용을 고려하는 것 역시 매우 중요하며, 자원 비용의 최소화를 통해 그리드 컴퓨팅의 전체적인 성능 및 경제적 효율성을 높일 수 있다. 따라서 본 논문에서는 시간과 비용을 모두 고려한 다목적 작업 스케줄링 모델을 제안한다. 제안하는 모델은 다목적 유전 알고리즘 기법의 하나인 NSGA-Ⅱ를 적용하여 최적 해를 도출하였고, 모델의 효율성을 증명하기 위해 시뮬레이션 환경을 구성하여 기존의 스케줄링 모델인 Min-Min, Max-Min 알고리즘과의 비교 실험을 수행하였다. 이를 통해 제안한 스케줄링 모델이 기존 스케줄링 모델에 비해 작업 총 완료시간과 자원 비용을 더욱 효율적으로 최소화함을 증명하였다.

▸ Keyword : 그리드 컴퓨팅, 그리드 스케줄링, 다목적 유전 알고리즘, NSGA-II

Abstract

Grid computing is a new generation computing technology which organizes virtual high-performance computing system by connecting and sharing geographically distributed heterogeneous resources, and performing large-scaled computing operations. In order to maximize the performance of grid

• 제1저자 : 김솔지 • 교신저자 : 김태호 • 책임저자 : 이홍철
• 투고일 : 2011.03.22, 심사일 : 2011.04.03, 게재확정일 : 2011.04.21.
* 고려대학교 정보경영공학부(Dept. of Information Management Engineering, Korea University)
** 고려대학교 산업경영공학부 교수(Dept. of Industrial Management Engineering, Korea University)

computing, job scheduling is essential which allocates jobs to resources effectively. Many studies have been performed which minimize total completion times, etc. However, resource costs are also important, and through the minimization of resource costs, the overall performance of grid computing and economic efficiency will be improved. So in this paper, we propose a multi-objective job scheduling model considering both time and cost. This model derives from the optimal scheduling solution using NSGA-II, which is a multi objective genetic algorithm, and guarantees the effectiveness of the proposed model by executing experiments with those of existing scheduling models such as Min-Min and Max-Min models. Through experiments, we prove that the proposed scheduling model minimizes time and cost more efficiently than existing scheduling models.

▸ Keyword : Grid Computing, Grid Scheduling, Multi Objective Genetic Algorithm, NSGA-II

I. 서 론

오늘날 IT 기술은 지속적인 발전을 통해 다양한 분야에서 폭넓게 활용되고 있으며, 이와 더불어 컴퓨팅 자원의 성능 또한 지속적으로 향상되고 있다. 최근에는 고속 네트워크의 개발과 인터넷의 대중화로 인해 많은 양의 데이터를 신속하게 처리하는 것이 중요하게 되었으며, 이에 따라 고성능의 컴퓨팅 자원이 요구되고 있다. 이러한 고성능 컴퓨팅 자원에 대한 요구를 충족시키기 위하여 새롭게 등장한 기술이 바로 그리드 컴퓨팅이다. 그리드 컴퓨팅은 1998년 미국 시카고 대학의 이안 포스터(Ian Foster) 교수가 자신의 저서[1]에서 처음 제시한 개념으로, 지리적으로 분산된 이기종의 컴퓨팅 자원들을 상호 연결하고 공유하여 가상의 고성능 컴퓨팅 시스템을 구성함으로서 대용량의 컴퓨팅 연산 등을 수행하는 차세대 컴퓨팅 기술이다[1]. 그리드 컴퓨팅은 지리적으로 분산되어 있는 유휴 컴퓨팅 자원들을 동적으로 활용하므로 기존의 분산 컴퓨팅이나 클러스터 컴퓨팅에 비해 확장성이 높으며, 가상의 고성능 시스템을 구축함으로서 자원, 비용적인 효율성을 극대화할 수 있다.

그리드 컴퓨팅에서 각 작업들은 다양한 컴퓨팅 자원들에 할당된다. 이 때 작업과 자원의 특성은 매우 다양하므로 작업들을 어떤 자원에 할당하느냐하는 문제는 그리드 컴퓨팅 시스템의 전체적인 성능에 매우 중요한 영향을 미친다[2]. 특히 작업들은 다양한 데이터 크기를 가지며, 컴퓨팅 자원들은 이기종이고, 지리적으로 분산되어 있으며, 각기 다른 컴퓨팅 성능을 가지고 있다. 따라서 그리드 컴퓨팅에서의 작업 스케줄링 문제는 NP-Complele로서 수리적으로 해결하기 어려운 문제이며[3], 이를 해결하기 위해 다양한 작업 스케줄링 기법에 대한 연구가 진행되고 있다.

그리드 컴퓨팅에서의 작업 스케줄링에 대한 연구는 주로 작업 총 완료시간을 최소화하기 위한 알고리즘의 연구가 진행되어져 왔다. 작업 총 완료시간은 모든 작업들이 자원에 할당되어 완료될 때까지의 시간을 말하며, 온라인 방식, 배치 방식 등의 연구가 진행되고 있다[4]. 온라인 방식은 스케줄링이 필요한 작업들이 발생할 때마다 실시간으로 자원에 작업을 할당하는 방식으로서 Minimum Completion Time(MCT), Minimum Execution Time(MET), Switching Algorithm(SA), K-Percent Best(KPB), Opportunistic Load Balancing(OLB) 알고리즘이 이에 해당한다[4]. 배치 방식은 작업을 실시간으로 스케줄링하지 않고 배치 단위로 묶어서 수행하는 방식으로서 Min-Min, Max-Min, Sufferage 알고리즘 등이 이에 해당한다[4].

그러나 이러한 기존의 연구는 그리드 컴퓨팅 시스템의 전체적인 관점에서 볼 때 자원의 사용에 따른 비용적 측면을 고려하지 않는 문제점이 있다. 그리드 컴퓨팅 시스템 제공자의 입장에서 볼 때, 그리드 컴퓨팅 시스템은 컴퓨팅 자원을 통해 사용자에게 서비스를 제공하는 일종의 경제적 수단이며, 서비스의 제공에 따른 경제적인 이익 창출이 중요한 목적이 될 수 있다. 이러한 시스템 제공자의 입장에서 볼 때, 자원의 사용에 따른 비용을 고려하는 것은 시스템의 경제적 효율성을 높이는 매우 중요한 요소라고 할 수 있다. 따라서 사용자의 QoS(Quality of Service)를 충족시키는 동시에 시스템 제공자의 경제적인 이익을 극대화하기 위해 작업 총 완료시간과 자원 사용에 따른 비용을 동시에 고려하는 작업 스케줄링 기법이 필요하다.

따라서 본 논문에서는 시간과 비용을 모두 고려한 다목적 작업 스케줄링 모델을 제안한다. 제안하는 모델은 다목적 유전 알고리즘 기법의 하나인 NSGA-II를 적용하여 작업 총 완료시간과 자원 비용의 다목적 최적화를 수행한다.

본 논문의 구성은 다음과 같다. 2장에서는 그리드 컴퓨팅의 기존 작업 스케줄링 모델과 NSGA-II 기법에 대해 설명하

고, 3장에서는 본 논문에서 제안하는 NSGA-Ⅱ 기반의 그리드 컴퓨팅 작업 스케줄링 모델에 대하여 설명한다. 4장에서는 시뮬레이션 환경에서 기존 스케줄링 모델과의 비교 실험을 통해 제안하는 모델의 효율성을 증명하고, 마지막 5장에서는 결론을 맺는다.

Ⅱ. 관련 연구

1. 그리드 스케줄링

그리드 컴퓨팅에서의 작업 스케줄링에 대한 연구는 크게 온라인 방식과 배치 방식으로 나뉘어 연구가 진행되고 있다[4].

온라인 방식의 대표적인 알고리즘으로는 MCT 방식을 들 수 있다. MCT는 각각의 작업에 대하여 가장 최소의 시간으로 작업을 완료할 수 있는 자원에 작업을 순서대로 할당한다. 작업이 각 자원에 할당되었을 때의 준비시간과 실행시간을 더하여 완료시간을 구하고, 이 중 가장 최소의 완료시간을 갖는 자원을 선택하여 작업을 할당하는 방식이다[5]. MCT는 온라인 방식의 가장 기본이 되는 알고리즘으로서, 많은 온라인 방식 알고리즘이 MCT에 기초하고 있다[4].

다음으로 배치 방식의 대표적인 알고리즘으로는 Min-Min, Max-Min 방식을 들 수 있다[6]. Min-Min 알고리즘은 크게 두 단계로 나뉘어 스케줄링이 이루어진다. 첫 번째 단계에서는 아직 수행되지 않은 작업들의 집합을 생성한 후, 각 작업의 완료시간을 최소로 하는 작업과 자원의 집합을 구성한다. 두 번째 단계에서는 작업, 자원으로 이루어진 집합 중 가장 최소의 작업 완료시간을 갖는 작업을 선택하여 해당 자원에 할당한다. 할당이 이루어진 작업은 작업 집합에서 제외되며, 모든 작업들이 자원에 할당될 때까지 위의 두 단계가 반복된다. Max-Min 알고리즘은 Min-Min 알고리즘과 방식이 거의 비슷하며, 두 번째 단계에서 가장 최대의 작업 완료시간을 갖는 작업을 해당 자원에 할당한다는 차이점이 있다.

Min-Min, Max-Min 알고리즘은 그리드 스케줄링에 다양한 방식으로 적용되었다. M. Y. Wu(2000)등은 기존의 Min-Min 알고리즘을 이용하여 Segmented Min-Min Algorithm을 제안하였다[7]. 이 알고리즘은 작업들을 기대 완료 시간(expected completion time) 순서대로 정렬한 후, 정렬된 작업들을 조각화하여 각 조각 별로 Min-Min 알고리즘을 적용하였다. 이 알고리즘은 작업들의 길이가 매우 다양할 때 길이가 긴 작업들이 먼저 실행되게 함으로서 기존의 Min-Min 알고리즘에 비해 향상된 성능을 나타내었다. He XiaoShan(2003)등은 네트워크의 bandwidth에 따른 각 작업의 Quality of Service(QoS)를 제약조건으로 하는 QoS Guided Min-Min heuristic을 제안하였다[8]. 이 알고리즘은 각 작업별로 요구하는 네트워크의 bandwidth가 다른 것에 착안하여, 높은 QoS를 요구하는 작업을 Min-Min 알고리즘에 의해 우선적으로 자원에 할당하여 QoS를 만족시키는 동시에 작업 총 완료시간을 최소화하였다. K. Etminani(2007)등은 Min-Min, Max-Min 알고리즘을 동시에 사용하는 알고리즘을 제안하였다[6]. 이 알고리즘은 각 작업들이 자원에 할당될 때의 기대 완료 시간을 구하여, 이 값들의 표준 편차를 기준으로 두 알고리즘 중 하나를 선택하는 알고리즘을 제안하였다.

이 밖에도 Z. Xu(2003)등은 Ant Algorithm을 이용한 그리드 스케줄링 기법을 연구하였고[9], S. Fidanova(2006)은 Simulated Annealing(SA)를 이용하여 각 자원의 Load Balancing을 보장하는 스케줄링 기법을 연구하였다[10].

하지만 위의 모델들은 공통적으로 자원의 비용을 고려하지 않는다는 문제점이 있다. 자원의 사용에 따른 비용을 고려함으로서 그리드 컴퓨팅 시스템 전체의 경제적 효율성을 높일 수 있으므로, 본 논문에서는 작업 총 완료시간과 자원 사용에 따른 비용을 동시에 최적화하기 위해 NSGA-Ⅱ를 적용하여 다목적 작업 스케줄링을 수행한다.

2. 다목적 최적화 문제

다목적 최적화 문제는 다수의 상충 관계를 갖는 목적함수들을 만족시키기 위한 변수들을 최적화하는 문제를 말하며, 다음과 같이 정의된다[11].

$$\min F(x) = [f_1(x), f_2(x), \cdots, f_k(x)]$$
$$s.t.\ x \in S,\ \ x = [x_1, x_2, \cdots, x_n] \quad \cdots\cdots\cdots (1)$$

위 식에서, $[f_1(x), f_2(x), \cdots, f_k(x)]$는 k개의 목적함수를 나타내고, 집합 x는 변수 공간 S의 부분집합이다. 예를 들어 그림 1과 같이 두 변수로 구성된 최적화 변수들이 존재할 때, 다목적 함수 F(x)에 의해 변수 공간인 S에서 f_1과 f_2로 표현되는 목적함수 공간인 Y로 변환될 수 있으며, 변환된 해들에 의해 형성된 경계면을 ∂Y로 표시한다.

이 때 모든 목적함수를 동시에 최소화하는 해를 구하는 것은 사실상 불가능하며, 이를 고려할 때 여러 후보해 중 다른 해에 지배되지 않는 해를 비지배해(Non-Dominated Solution)라 한다.

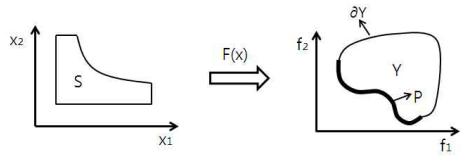


그림 1. 변수 공간과 목적함수 공간의 상관관계
Fig. 1. Relation of variable space and objective function space

일반적으로 다음의 식을 만족하면 해 x가 해 y를 지배한다고 하며, x는 비지배해가 되고 x≺y로 표시한다.

$$\forall i \in \{1,2,...,k\}: f_i(x) \leq f_i(y)$$
$$\text{and } \exists j \in \{1,2,...,k\}: f_j(x) < f_j(y) \cdots\cdots\cdots (2)$$

식 (2)를 통해 비지배해로 분류된 해들을 파레토 최적해라 하며, 그 집합을 파레토 최적 집합이라고 한다. 파레토 최적해의 목적함수들에 의해 형성된 공간을 파레토 최적 프론트라 하고, 그림 1에서 P로 표시된 부분을 나타낸다. 파레토 최적해들 중에서 최종적으로 해가 선택되었다는 것은 모든 목적함수에 있어 더 나은 해가 존재하지 않는다는 것을 의미한다[11].

3. NSGA-II 알고리즘

Nondominated Sorting Genetic Algorithm Ⅱ (NSGA-Ⅱ) 알고리즘은 다목적 최적화 문제를 풀기 위한 다목적 유전 알고리즘의 대표적 기법이다. 다목적 유전 알고리즘은 다양한 파레토 최적해들을 찾는데 그 목적이 있다. 유전 알고리즘을 다목적 문제에 적용할 때, 모집단에서 생존할 개체를 어떻게 선별하느냐는 매우 중요한 문제이다[12]. 또한 단일 목적 최적화 문제와 달리 다목적 문제는 여러 목적함수가 존재하므로 각 개체의 적응도를 어떤 기준에 의해 측정하느냐가 문제이다. 따라서 다목적 유전 알고리즘의 연구 초점은 주로 선별방법에 있으며, 이러한 선별방법에 쓰이는 알고리즘이 바로 NSGA-Ⅱ 알고리즘이다. NSGA-Ⅱ 알고리즘은 기존의 NSGA 알고리즘의 단점을 보완한 기법으로, 비지배해 정렬방법의 복잡도가 감소하였으며, 군집거리(Crowding Distance)를 도입하여 각각의 자원을 보다 효율적으로 배분하도록 하였고, 현재 세대의 최적해를 다음 세대로 넘겨주는 엘리티즘이 적용되었다[13].

NSGA-Ⅱ의 실행 단계는 그림 2와 같다[14]. 첫 번째 단계로 초기 모집단을 선정하고 종료 조건을 만족하는지 판단한다. 종료 조건을 만족하지 않는 경우 토너먼트 선별(Tournament selection), 재생산(Reproduction) 등을 통해 자손 집단을 생성하고, 모집단과 자손집단을 결합하여 새로운 집단을 생성한다. 그리고 Fast non-dominating sorting algorithm을 이용하여 비지배해를 정렬하고, 정렬된 비지배 프론트와 군집거리 계산을 통해 선별 과정이 이루어진다. 선별된 집단에 대하여 교차와 돌연변이를 통해 다음 세대로 진화할 모집단이 결정되고, 적합도 평가가 이루어진다. 위의 과정을 종료조건이 만족될 때까지 반복적으로 수행하게 된다.

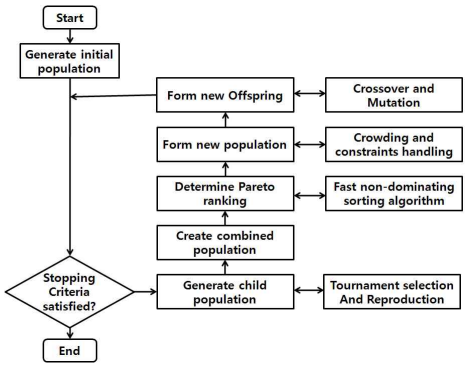


그림 2. NSGA-II의 실행단계
Fig. 2. Process of NSGA-II

III. NSGA-II 기반의 작업 스케줄링 모델

본 절에서는 NSGA-Ⅱ 알고리즘을 기반으로 하여 시간과 비용을 동시에 최소화하는 그리드 컴퓨팅 작업 스케줄링 모델을 제안한다. NSGA-Ⅱ 알고리즘은 단일 목적 유전 알고리즘의 수행 과정과 거의 비슷하며, 선별 방법에 있어서 차이를 보인다. 본 논문에서는 배치 방식의 스케줄링 모델을 제안하며, 각 작업을 배치 단위로 묶어서 스케줄링을 수행한다.

작업 스케줄링의 단계는 크게 개체 표현, 초기 모집단의 선정, 적합도 평가, NSGA-Ⅱ를 이용한 선별, 교차, 돌연변이로 이루어지며, 각 단계의 구성은 다음과 같다[12].

1. 개체 표현

작업 스케줄링 문제를 유전적으로 표현하기 위하여, 본 논문에서는 그리드 컴퓨팅의 각 자원을 원소로 하는 염색체 개체를 구성한다. n 개의 작업으로 구성된 J_i, $i = \{1,...,n\}$과 m 개의 자원으로 구성된 R_j, $j = \{1,...,m\}$에 대하여, 개체의 i 번째 원소 값 j는 작업 J_i가 자원 R_j에 할당된다는 것을 의미한다. 예를 들어 그림 3과 같이 그리드 개체가 구성

되었다고 할 때, 개체 $X=\{4,1,3,4,\cdots\}$ 는 J_1은 R_4에 할당되고, J_2는 R_1, J_3는 R_3, 그리고 J_4는 R_4에 할당된다는 것을 나타낸다. 이 때 J_1과 J_4는 모두 R_4에 할당되었으며, J_1-J_4 순서대로 작업이 수행된다는 것을 의미한다.

그림 3. 개체의 구성
Fig. 3. Configuration of the Gene

2. 초기 모집단

초기 모집단을 구성하는 방법으로는 문제의 특성을 이용한 발견적 방법과 임의생성방법이 있다[12]. 발견적 기법에 의해 생성된 초기 해들은 진화를 거듭하는 과정에서 해 공간의 다양한 탐색을 방해하는 경향이 있으므로, 본 논문에서는 임의생성방법을 사용한다. n 개의 작업과 m 개의 그리드 자원이 존재하는 경우, 한 개체의 원소의 개수는 n 개가 되며, 1에서 m 사이의 값 n 개를 임의로 발생하여 한 개체를 구성한다. 모집단의 크기만큼 위의 과정을 반복하여 그리드 자원의 초기 모집단을 구성한다.

3. 적합도 평가

적합도 평가는 목적함수 값을 통해 이루어지며, 아래 식과 같이 작업 총 완료시간과 자원 사용 비용을 동시에 최소화하는 다목적함수로 구성한다.

$$\min F=(f_1,f_2)$$
$$\begin{cases} f_1 = \max(Completion\ Time[i][Res(i)]) \\ f_2 = \sum_{i=1}^{n}(Cost[i][Res(i)]) \end{cases}$$
$$\text{for } i=1,2,\cdots,n \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots (3)$$

위의 식에서 $Completion\ Time[i][Res(i)]$은 작업 J_i가 할당된 자원 $Res(i)$에서 완료되는 시간을 나타내며, f_1은 이 값들 중 최대값인 가장 마지막으로 작업이 완료되는 시간을 나타낸다. $Cost[i][Res(i)]$는 작업 J_i가 할당된 자원 $Res(i)$에서 수행될 때의 자원 사용 비용을 나타내며, 모든 작업들의 수행에 따른 비용의 총합으로 f_2값이 결정된다.

목적함수 계산의 실제적인 과정은 그림 4와 같다. 각 자원의 작업 총 완료시간 resource[i]는 기존의 누적된 실행시간에 할당된 작업들의 실행시간 obj[j][i]을 계속적으로 더하여 구한다. 그리고 자원 사용 비용의 총합 costSum은 기존의 비용에 할당된 작업들의 실행시간에 따른 비용을 더하여 구한다. 그리고 각 자원 별 작업 총 완료시간 값들을 작은 순서대로 나열한 후 가장 큰 값을 작업 총 완료시간으로 결정한다. 특정 작업을 어떤 자원에 할당하느냐에 따라 작업 실행시간과 자원 비용이 달라지므로, 모집단의 각 그리드 개체의 작업-자원 할당 구성에 따라 작업 총 완료시간 (f_1), 자원 사용 비용 (f_2)의 값이 달라지며, 이를 통해 모집단 내의 모든 그리드 개체에 대한 적합도 평가가 이루어진다.

```
for(int i=0;i<5;i++)
{
    for(int j=0;j<70;j++)
    {
        resource[i] = resource[i]+ obj[j][i];
    }
}

for(int i=0;i<5;i++)
{
    for(int j=0;j<70;j++)
    {
        costSum = costSum+ obj[j][i]*cost[i];
    }
}

Arrays.sort(resource);
f[0] = resource[resource.length-1];
f[1] = costSum;
```

그림 4. 목적함수의 계산 과정
Fig. 4. Computation of objective functions

4. NSGA-Ⅱ를 이용한 선별 (Selection)

모집단 선정 및 적합도 평가가 끝난 후 NSGA-Ⅱ를 이용하여 그리드 개체의 선별 단계를 거친다. 선별(Selection)은 현 세대의 모집단으로부터 다음 세대에 생존할 개체를 선택하는 과정이다[12]. NSGA-Ⅱ 알고리즘의 전체적인 과정은 다음과 같다. t 세대의 모집단 P_t에 대하여 기존의 선별 방법인 토너먼트 선별, 재생산 등을 통해 자손 집단 Q_t를 생성한다. 그 후 모집단과 자손 집단을 결합하여 새로운 집단 $R_t=P_t\cup Q_t$ 을 생성하고, R_t의 각 그리드 개체에 대하

여 작업-자원 할당 구성에 따라 작업 총 완료시간, 자원 사용 비용을 구하여 이를 비지배 관계에 따라 정렬한다. 정렬된 비지배 프론트들에 대하여 다음 세대 P_{t+1}로 진화할 N 개의 그리드 개체에 대한 선별이 이루어진다. 이 때 가장 먼저 일차 비지배 프론트의 개체들이 다음 세대로 우선 선택되고, N 개의 개체가 모두 선택될 때까지 이차, 삼차 비지배 프론트의 개체들이 선택된다. 만약 마지막에 같은 비지배 프론트 내의 그리드 개체들 중 일부를 선택해야 하는 경우, 군집거리를 계산하여 이 값에 따라 그리드 개체의 일부를 다음 세대의 개체로 선택한다. 이 과정을 통해 다음 세대로 진화하기 위한 그리드 개체의 선별 작업이 이루어진다.

위의 과정에서 핵심적인 단계는 비지배해 정렬과 군집거리 계산이다. 먼저 비지배해 정렬은 개체의 비지배 정도에 따라 순위를 부여하는 단계로서, NSGA-Ⅱ는 Fast Nondominated Sorting Approach를 통해 비지배해를 정렬한다. 비지배해 정렬 방법은 그림 5와 같다[13]. 비지배해 정렬을 위해 우선 모든 그리드 개체의 작업-자원 할당에 따른 작업 총 완료시간과 자원 사용 비용 값을 구한다. 이 값들을 통해 각 그리드 개체들 간의 지배 관계를 판단하게 된다. 이 때 S_p는 그리드 개체 p가 지배하는 그리드 개체들의 집합이고, n_p는 p를 지배하는 그리드 개체의 개수를 나타낸다. 모집단 P에 속하는 임의의 개체 p와 q에 대하여 지배성을 판단하여, 만약 p가 q를 지배한다면 q는 S_p에 포함되고, 반대로 q가 p를 지배한다면 p를 지배하는 그리드 개체의 개수를 나타내는 n_p를 1 증가시킨다. 이러한 과정은 모집단 P에 속하는 모든 그리드 개체들 간의 비교과정이 완료될 때까지 반복되며, 완료 후 p를 지배하는 개체 수 n_p가 0인 개체들이 모두 1차 비지배 프론트로 분류된다. 그 다음 1차 비지배 프론트로 분류된 모든 p 개체에 대하여, S_p에 속하는 모든 개체들 q의 n_q를 1씩 감소시킴으로서, n_q가 0이 되는 모든 개체들을 2차 비지배 프론트로 분류한다. 모든 그리드 개체에 비지배 프론트가 할당될 때까지 위의 과정을 반복함으로서 비지배해를 정렬하게 된다.

```
for each p∈P
  Compute f1, f2
  Sp = ∅
  np = 0
  for each q∈P
    if (p < q) then
      Sp = Sp ∪ {q}
    else if (q < p) then
      np = np + 1
  end
 if np = 0 then
  F1 = F1 ∪ {p}
 end
end

i = 1
while Fi ≠ ∅
 Q = ∅
 for each p∈Fi
  for each q∈Sp
   nq = nq − 1
   if nq = 0 then Q = Q ∪ {q}
i = i + 1
```

그림 5. 비지배 프론트 정렬 과정
Fig. 5. Process of the nondominated front sorting

그리드 스케줄링에서의 실제적인 비지배해 정렬 과정은 그림 6과 같다. 그림 6에서 iDominate[p]는 p가 지배하는 그리드 개체들의 집합을 나타내며, dominateMe[p]는 p를 지배하는 그리드 개체들의 개수를 나타낸다. 지배 관계를 판단하여 p를 지배하는 그리드 개체 수가 0인 경우 비지배 프론트에 추가함으로서 정렬이 이루어진다.

두 번째로 군집거리(Crowding Distance) 계산은 개체들의 다양성을 보존하기 위한 방법으로서, 같은 비지배 프론트에 속하는 그리드 개체들에 대하여 각 목적함수 값에 따라 인접한 두 개체의 거리의 합으로 밀도를 측정한다. 각 개체의 밀도는 같은 비지배 프론트의 개체들에 대한 부분 순서로서 사용된다. 즉, 같은 비지배 프론트 차수를 가지고 있는 개체일 지라도 높은 밀도를 가지는 개체는 낮은 밀도를 가지고 있는 개체에 비해 다음 세대에서 탈락될 확률이 높아짐으로서 작업-자원 할당 구성의 다양성을 보존한다.

```
//Fast non dominated sorting Approach
for (int p = 0; p < solutionSet_.size(); p++) {
  iDominate[p] = new LinkedList<Integer>();
  dominateMe[p] = 0;
  // 지배관계에 따라 정렬
  for (int q = 0; q < solutionSet_.size(); q++) {
    flagDominate =constraint_.compare(solutionSet.get(p),solutionSet.get(q));
    if (flagDominate == 0) {
      flagDominate =dominance_.compare(solutionSet.get(p),solutionSet.get(q));
    }

    if (flagDominate == -1) {
      iDominate[p].add(new Integer(q));
    } else if (flagDominate == 1) {
                dominateMe[p]++;
    }
  }

  // p 를 지배하는 개체 수가 0인 경우, 비지배 프론트에 추가
  if (dominateMe[p] == 0) {
    front[0].add(new Integer(p));
    solutionSet.get(p).setRank(0);
  }
}
```

그림 6. 비지배 프론트 정렬의 실제 코드
Fig. 6. Codes of the nondominated front sorting

군집거리의 계산 과정은 그림 7과 같다[13].

d각각의 비지배 프론트 F_i에 속하는 그리드 개체의 개수를 $n = n(F_i)$로 놓고 군집거리를 0으로 초기화한다. 그 다음 목적함수인 작업 총 실행시간, 자원 사용 비용의 값에 따라 그리드 개체들을 정렬하고, 이를 $I = sort(F_i, m)$ 로 나타낸다. 정렬된 개체들에 대하여 첫 번째와 마지막 개체의 군집거리를 무한대로 설정하고, 나머지 그리드 개체들에 대하여 군집거리 $I[k]_{distance}$를 구한다. 군집거리 계산식에서 $I[k]_m$은 목적함수 m의 값에 따라 정렬된 개체들 중 k번째 개체의 목적함수 값을 나타내며, $f_m^{max} - f_m^{min}$ 은 개체들의 목적함수 m 의 값 중 최대값과 최소값의 차이를 나타낸다. 군집거리 값이 큰 개체일수록 밀도가 작으므로 같은 비지배 프론트 내의 그리드 개체들을 군집거리 값이 큰 순서대로 정렬하고, 이 순서대로 그리드 개체를 선택함으로서 개체의 다양성을 보존한다. 군집거리 계산을 그리드 스케줄링에 적용하는 과정은 그림 8과 같으며, crowdingDistanceAssignment() 메서드를 통해 계산이 이루어진다.

```
// 군집거리 계산 과정
while ((remain > 0) && (remain >= front.size())) {
  //각 그리드 개체의 군집거리 계산
    distance.crowdingDistanceAssignment(front, problem_.getNumberOfObjectives());
  //그리드 개체의 군집 거리에 따라 비지배 프론트에 추가
  for (int k = 0; k < front.size(); k++) {
    population.add(front.get(k));
  }

  remain = remain - front.size();

  //다음 차수의 비지배 프론트 생성
  index++;
  if (remain > 0) {
    front = ranking.getSubfront(index);
  }
}
```

그림 8. 군집거리 계산의 실제 코드
Fig. 8. Codes of computation of the crowding distance

Ⅳ. 실험 및 결과 분석

본 논문에서는 자바 기반의 메타휴리스틱 알고리즘 구현 도구인 JMetal[15]과 그리드 컴퓨팅 시뮬레이션 도구인 GridSim Toolkit[16]을 이용하여 제안하는 작업 스케줄링 모델의 성능을 평가하고, 기존 알고리즘과의 비교 실험을 통해 제안하는 모델의 효용성을 검증한다.

1. 시뮬레이션 환경

제안하는 작업 스케줄링 모델의 성능을 평가하기 위해 본 논문에서는 JMetal과 GridSim Toolkit을 사용한다. JMetal은 다양한 메타휴리스틱 알고리즘을 개발하고 실험할 수 있는 프레임워크로, NSGA-Ⅱ와 같은 알고리즘의 실제적인 구현이 가능하다[15]. GridSim은 자바 기반의 그리드 컴퓨팅 시뮬레이션 도구로서 사용자, 작업, 자원, 스케줄러 등의 설계와 스케줄링 알고리즘의 성능 평가에 유용하게 쓰인다[17].

실험에 사용되는 그리드 자원은 총 5개의 이질적인 성능을 가진 자원으로 구성하였으며, 각 자원의 성능 및 자원 사용 비용은 WWG(World-Wide Grid) testbed를 기준으로 하였다[18]. 자원의 구성은 표 1과 같다. 표 1에서 A PE SPEC/ MIPS Rating은 자원의 성능을 나타내는 단위로, 1 MIPS (Million Instructions Per Second)는 1초당 백만 회의 명령을 실행하는 자원의 성능을 나타낸다. 또한 Price 항목은 자원 비용을 나타내며, G$는 Grid Dollar를 나타낸다.

작업의 개수는 30개, 50개, 70개로 나누어 실험을 수행하였고, 각 작업의 크기는 15 MB로 하였다. 작업의 길이(Length)는 10000 MI(Million Instructions)를 기준으로 0~10% 사이의 난수 값을 생성하여 다양한 길이의 작업을 구성하였다.

NSGA-Ⅱ 알고리즘의 모집단 크기는 100 개로 하였고, 교차연산자로는 SBX mf Crossover 연산자를 사용하였으며 교차율은 0.9로 설정하였다. 또한 돌연변이 연산자로는 Polynomial Mutation을 사용하였고 돌연변이율은 (1/개체당 원소 개수)로 설정하였다. 알고리즘의 종료조건으로는 100 세대를 기준으로 하였다.

표 1. WWG 테스트베드

Resource Name	Vendor, Resource Type, Node OS, No of PEs	Host Name, Location	A PE SPEC/ MIPS Rating	Price(G$/PE time unit)
R0	Compaq, AlphaServer, CPU, OSF1, 4	VPAC, Melb, Australia	515	8
R1	Sun, Ultra, Solaris, 4	AIST, Tokyo, Japan	377	4
R2	SGI, Origin 3200, IRIX, 4	Manchester, UK	410	6
R3	Sun, Ultra, Solaris, 8	ANL, Chicago, USA	377	3
R4	SGI, Origin 3200, IRIX, 6	ZIB, Berlin, Germany	410	5

Table 1. WWG Testbed

실험의 전체적인 순서는 그림 9와 같다. 먼저 작업, 자원의 성능 및 특성을 입력 값으로 하여 JMetal 프로그램의 NSGA-Ⅱ 실험 환경을 구성한다. 구성된 프로그램을 통해 100세대의 진화가 이루어지며, 이 결과로 최적의 작업-자원 할당 개체가 도출된다. 도출된 결과를 GridSim Toolkit의 입력 값으로 하여 실제 스케줄링 시뮬레이션이 이루어지며, 스케줄링에 따른 총 실행시간과 자원 사용 비용의 총합이 결과 값으로 나오게 된다. 또한 기존의 Min-Min, Max-Min 알고리즘에 대해서도 스케줄링 시뮬레이션을 수행하여 결과 값을 도출하고, 이를 제안하는 모델과 비교한다. 5개의 자원 성능 및 자원 사용 비용이 다양하게 설정되어 있으므로, 각 모델 별로 작업-자원을 어떻게 할당하느냐에 따라 작업 총 실행시간과 자원 비용이 다르게 나올 것이며, 이를 통해 제안한 모델과 기존 모델과의 성능을 비교한다.

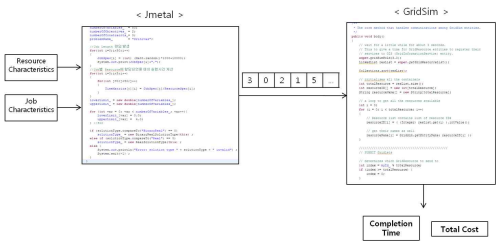


그림 9. 스케줄링 순서
Fig. 9. Sequece of the scheduling

2. 시뮬레이션 결과 분석

본 논문에서는 작업 수에 따른 작업 총 완료시간과 총 자원 비용을 성능 평가의 기준으로 사용한다. 작업 총 완료시간은 모든 작업들이 그리드 자원에 할당되어 작업이 완료되었을 때의 시간을 말하며, 자원 비용은 각각의 컴퓨팅 자원의 사용에 따른 비용의 총합을 나타낸다. 위의 두 가지 성능 평가 척도를 기준으로 하여 제안한 알고리즘과 기존의 배치 방식 스케줄링 알고리즘인 Min-Min, Max-Min 알고리즘의 성능을 비교하였다.

먼저 첫 번째 실험은 작업 수에 따른 작업 총 완료시간을 비교하였다. 그 결과는 그림 10과 같다.

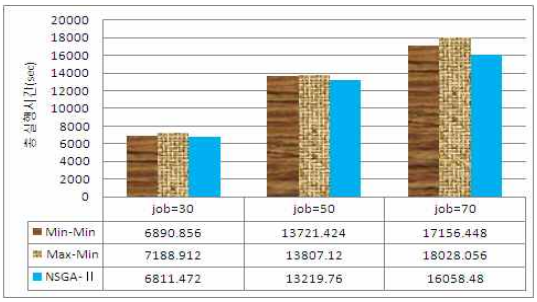


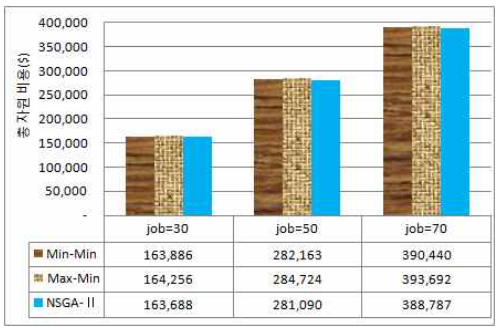
그림 10. 작업 수에 따른 작업 총 실행시간
Fig. 10. Total execution time

그림 10을 보면 우선 작업의 개수에 상관없이 제안하는 모델, Min-Min, Max-Min 모델 순으로 작업 총 완료시간이 작음을 알 수 있다. 이러한 결과는 제안하는 모델이 기존 두 모델에 비해 더 효과적으로 작업 총 실행시간을 줄이는 스케줄링을 수행 한다는 것을 나타낸다. 특히 제안하는 모델과 두 번째로 좋은 결과를 나타낸 Min-Min 모델의 실행시간 차이를 보면, 작업의 개수가 30 개일 때는 79.384 초, 50 개일 때는 501.664 초, 그리고 70 개일 때는 1097.968 초로 작업의 개수가 증가함에 따라 제안하는 모델이 더욱 효율적으로 총 실행시간을 줄이고 있음을 알 수 있다. 제안하는 모델인 NSGA-Ⅱ 기반 스케줄링 알고리즘은 총 실행시간과 자원 비용을 동시에 최소화하는 데 목적이 있음에도 불구하고, 기존의 Min-Min, Max-Min 알고리즘과 같은 총 실행시간을 줄이는 것을 목적으로 한 알고리즘보다 더 좋은 성능을 내고 있음을 알 수 있다.

두 번째 실험은 작업 수에 따른 총 자원 비용을 비교하였다. 실험 결과는 그림 11과 같다.

그림 11을 보면 우선 작업 수가 30 개일 때는 제안하는 모델이 가장 비용이 낮으나, 대체적으로 세 모델 모두 비슷한

결과를 나타낸다. 그러나 작업 수가 50 개, 70 개로 증가함에 따라 제안하는 모델이 훨씬 좋은 결과를 나타낸다. 작업의 개수가 50 개일 때 제안하는 모델의 비용은 281,090이고 Min-Min 모델은 282,163로 1,073의 비용 차이가 난다. 특히 작업의 개수가 70 개일 때는 1,653 정도의 많은 비용 차이가 난다. 이러한 결과는 기존의 Min-Min, Max-Min 모델은 자원 비용을 고려하지 않는 것에 비해, 제안한 모델에 사용된 NSGA-Ⅱ 알고리즘은 총 실행시간 뿐만 아니라 자원 사용 비용을 동시에 최소화하는 것을 목적함수로 두고 작업-자원 할당 스케줄링을 수행했기 때문이라고 할 수 있다.



	job=30	job=50	job=70
Min-Min	163,886	282,163	390,440
Max-Min	164,256	284,724	393,692
NSGA-II	163,688	281,090	388,787

그림 11. 작업 수에 따른 총 자원 비용
Fig. 11. Total Resource Cost

이 두 가지 실험의 결과는 제안한 그리드 컴퓨팅 스케줄링 모델이 기존의 모델에 비해 총 실행시간과 자원 비용 측면에서 모두 효율적으로 스케줄링을 수행함을 나타낸다. 특히 NSGA-Ⅱ를 이용한 작업 스케줄링 알고리즘은 시간, 비용을 동시에 최적화할 뿐만 아니라, 다양한 해 공간을 탐색하여 적절한 스케줄링 방안을 제시하므로 훨씬 효율적으로 스케줄링을 수행할 수 있다. 이를 통해 그리드 컴퓨팅 사용자의 QoS(Quality of Service)를 충족시키는 동시에 제공자의 입장에서는 경제적인 비용을 최소화하는 그리드 작업 스케줄링이 이루어진다.

V. 결론

그리드 컴퓨팅은 이기종의 컴퓨팅 자원들을 상호 연결하여 가상의 고성능 컴퓨팅 환경을 구현하는 기술이다. 이러한 그리드 컴퓨팅의 성능을 극대화하기 위해서는 각기 다른 특성을 갖고 있는 작업들을 이기종의 자원에 어떻게 할당하느냐가 매우 중요하다. 따라서 그리드 컴퓨팅 작업 스케줄링에 대한 많은 연구가 진행되었고, 특히 작업 총 실행시간을 줄이기 위한 연구가 주로 진행되었다. 그러나 자원 사용에 따른 비용을 줄이는 것 역시 그리드 컴퓨팅 시스템의 전체적인 성능에 중요한 영향을 미치며, 경제적인 효율성을 높일 수 있다. 따라서 본 논문에서는 시간과 비용을 모두 고려한 다목적 작업 스케줄링 모델을 제안하였다.

제안하는 모델은 배치 방식의 스케줄링 기법으로서, 다목적 유전 알고리즘인 NSGA-Ⅱ를 적용하여 스케줄링을 수행하였다. 먼저 각 자원 집합을 한 개체로 하여 모집단을 구성하고, 총 실행시간과 자원 비용을 다목적함수로 하여 적합도를 평가하였다. 그 다음 NSGA-Ⅱ를 적용하여 비지배해를 정렬하고, 군집거리 계산을 통해 다양한 그리드 개체의 해 공간을 탐색하였다. 이를 통해 모집단에 대한 선별 과정이 이루어졌고, 교차와 돌연변이를 통해 최종적으로 다음 세대의 모집단을 구성하였다. 위의 과정을 계속적으로 반복함으로서 시간과 비용을 동시에 최소화하는 최적의 그리드 작업-자원 할당 결과를 도출하였다.

제안한 모델의 성능을 평가하기 위해, JMetal과 GridSim Toolkit을 이용하여 실제 스케줄링 알고리즘을 구현하였다. 그리고 작업 수에 따른 총 실행시간, 자원 비용을 성능 평가 기준으로 하여 기존의 배치 방식 알고리즘인 Min-Min, Max-Min 알고리즘과의 비교 실험을 수행하였다. 실험 결과 작업 수에 상관없이 제안한 모델이 더 효율적으로 총 실행시간을 줄임을 알 수 있었다. 또한 자원 비용 측면에서도 작업 수가 증가함에 따라 제안한 모델이 기존 모델에 비해 비용을 최소화하는 스케줄링을 수행함을 볼 수 있었다. 이를 통해 그리드 컴퓨팅 환경에서, 본 논문의 스케줄링 기법이 기존의 시간을 고려한 스케줄링 기법에 비해 시간, 비용 면에서 모두 효율적으로 스케줄링을 수행함을 증명하였다.

Ⅵ. 감사의 글

이 연구에 참여한 연구자(의 일부)는 '2단계BK21사업'의 지원비를 받았음.

참고문헌

[1] I. Foster, and C. Kesselman, "The Grid: Blueprint for a New Computing Infrastructure," Morgan Kaufmann Publishers, 1998.

[2] Junzhou Luo, Peng Ji, Xiaozhi Wang, Ye Zhu, Feng

Li, Teng Ma, and Xiaopeng Wang, "Resource management and task scheduling in grid computing," Proceedings of The 8th International Conference on Computer Supported Cooperative Work in Design, Vol. 2, pp. 431-436, May. 2004.

[3] O. H. Ibarra and C. E. Kim, "Heuristic Algorithm for Scheduling Independent Tasks on Nonidentical Processors," Journal of the ACM, Vol. 24, No. 2, pp. 280-289, Apr. 1977.

[4] M. Maheswaran, S. Ali, H. J. Siegal, D. Hensgen, and R. F. Freund, "Dynamic Matching and Scheduling of a Class of Independent Tasks onto Heterogeneous Computing Systems," Proceedings of 8th Heterogeneous Computing Workshop, pp. 30-44, 1999.

[5] L. D. Briceno, M. Oltikar, H. J. Siegel, and A. A. Maciejewski, "Study of an Iterative Technique to Minimize Completion Times of Non-Makespan Machines," Proceedings of 2007 IEEE International Parallel and Distributed Processing Symposium, pp. 1-14, Mar. 2007.

[6] K. Etminani, and M. Naghibzadeh, "A Min-Min Max-Min selective algorithm for grid task scheduling," Proceedings of the 3rd IEEE/IFIP International Conference in Central Asia on Internet, pp. 1-7, Sep. 2007.

[7] M. Y. Wu, W. Shu, and H. Zhang, "Segmented Min-Min: A Static Mapping Algorithm for Meta-Tasks on Heterogeneous Computing Systems," 9th IEEE Heterogeneous Computing Workshop, pp. 375-385, May. 2000.

[8] He XiaoShan, Sun XianHe, and Gregor von Las zewski, "QoS Guided Min-Min Heuristic for Grid Task Scheduling," Journal of Computer Science and Technology, Vol. 18, No. 4, pp. 442-451, July. 2003.

[9] Z. Xu, X. Hou, and J. Sun, "Ant algorithm-based task scheduling in grid computing," Proceedings of 2003 IEEE Canadian Conference on Electrical and Computer Engineering, pp. 1107-1110, 2003.

[10] S. Fidanova, "Simulated Annealing for Grid Sched uling Problem," Proceedings of the IEEE John Vincent Atanasoff 2006 International Symposium on Modern Computing (JVA'06), pp. 41-45, 2006.

[11] J. Andersson, "A survey of multiobjective optimiz ation in engineering design," Reports of the Department of Mechanical Engineering, LiTH-IKP-R-1097, Linkoping University, Linkoping, 2000.

[12] Kim Yeo Geun, Yun Bok Sik, Lee Sang Bock, "Meta Heuristics," Young-Ji Books, pp. 3-150, 1997.

[13] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," IEEE Transactions on Evolutionary Computation, Vol. 6, No. 2, pp. 182-197, Apr. 2002.

[14] S. T. Khu, and H. Madsen, "Multi-objective calibr ation with Pareto preference ordering: An application to rainfall-runoff model calibration," Water Resources Research, Vol. 41, No. 3, Mar. 2005.

[15] JMetal, http://jmetal.sourceforge.net

[16] GridSim, http://www.gridbus.org/gridsim

[17] R. Buyya, and M. Murshed, "GridSim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing," The Journal of Concurrency and Computation: Practice and Experience, Vol. 14, No. 13-15, pp. 1175-1220, 2002.

[18] WWG testbed, http://www.buyya.com/ecogrid/wwg

저 자 소 개

김 솔 지

2009: 고려대학교
산업시스템정보공학과 공학사.
현 재: 고려대학교
정보경영공학과 석사과정
관심분야: 그리드컴퓨팅, SI
Email : anngung@hanmail.net

김 태 호

2006: 고려대학교
산업시스템정보공학과 공학사.
2008: 고려대학교
산업시스템정보공학과 공학석사.
현 재: 고려대학교
정보경영공학과 박사과정 수료
관심분야: SI, SOA
Email : airth@korea.ac.kr

이 홍 철

1983: 고려대학교 산업공학과 학사.
1988: Univ. of Texas
산업공학 석사.
1993: Texas A&M Univ.
산업공학 박사
현 재: 고려대학교
산업경영공학부 교수
관심분야: SCM, 생산 및 물류정보
시스템
Email : hclee@korea.ac.kr