이질적인 계산자원환경에서 독립적인 작업들을 위한 온라인 휴리스틱 스케줄링 알고리즘

김학두[0] 김진석
{hdkim94, jskim}@venus.uos.ac.kr

On-line Heuristic Scheduling Algorithm for Independent Tasks in Heterogeneous Computing Environment

Hak Du Kim[0] and Jin Suk Kim
Dept. of Computer Science and Statistics, University of Seoul

요 약

이질적인 계산자원들로 구성된 환경에서 독립적인 작업들을 스케줄링하기 위한 최적의 방법을 찾는 것은 NP-Complete 문제로 알려져 있다 [4]. 현재까지 이 문제를 풀기 위한 다양한 휴리스틱 스케줄링 방법이 연구되어 왔다 [1, 8, 9, 10]. 작업의 선후 관계를 예측할 수 없는 상황에서는 동적 스케줄링 방법을 사용하며 동적 스케줄링 방법은 스케줄링 시기에 따라 온라인방식과 배치방식으로 나누어진다 [1, 12]. 본 논문에서는 새로운 스케줄링 알고리즘을 제안하였으며 제안된 스케줄링 알고리즘의 성능이 기존의 스케줄링 알고리즘의 성능보다 뛰어남을 실험을 통하여 보였다.

1. 서 론

동일한 자원으로 한곳에 시스템을 구성하는데는 많은 비용이 소요되며 많은 물리적인 공간이 필요하게 된다. 이런 이유로 최근 지역적으로 분산되어 있는 이질적인 자원들을 하나로 묶어 거대한 시스템을 구성하고자 하는 연구가 진행되고 있으며 예의 하나가 GRID [2]이다. 이러한 시스템을 구성하는 자원들은 특성이 다르기 때문에 같은 작업을 실행할 때 실행 능력이 다르게 된다. 예를 들면, 다른 자원보다 입출력 속도가 빠른 자원은 입출력의 속도를 요구하는 작업을 실행할 때 그렇지 않은 자원보다 많은 이점을 가지게 된다. 그러므로 작업들을 어떤 자원에 배치할 것인가가 문제가 되고 현재까지 이 문제에 대한 연구가 진행되고 있다 [1, 8, 9, 10]. 서로 독립적인 작업들을 이질적인 자원들로 구성된 환경에서 어떻게 배치하여 실행할 것인가 하는 것은 NP-Complete 문제로 알려져 있다 [4]. 본 논문에서 제안한 알고리즘은 독립적인 작업을 스케줄링하기 위한 동적 스케줄링의 온라인방식 알고리즘이다. 본 논문의 관련 연구에서는 알고리즘을 기술하는데 필요한 기본 용어 설명과 기존의 동적 스케줄링의 온라인 휴리스틱 알고리즘을 소개하고 간략하게 그 방법을 설명하였다. 본론에서는 본 논문에서 제안한 알고리즘을 구체적으로 설명하고 실험 및 결과에서는 실험을 하기 위한 기본 가정들과 구성 방법들을 기술하였으며 실험에 따른 결과를 그래프를 통하여 보이고 분석하였다.

2. 관련용어

알고리즘의 성능을 측정하는 데는 makespan [5]이 사용된다. makespan은 전체 작업 중 마지막 작업이 스케줄링 되었을 때 자원들의 이용가능시간 (ready time) 중 가장 큰 값이다. 이용가능시간은 자원이 자신에게 주어진 모든 작업의 실행을 마친 시간을 말한다. n개의 작업으로 구성된 작업집합을 $T = \{T_1, T_2,...,T_n\}$, m개의 자원으로 구성된 자원집합을 $R=\{R_1, R_2,...,R_m\}$이라고 할 때, 작업 T_i를 자원 R_i에서 실행했을 때 작업이 완료되는 시간인 완료시간 (completion time)을 ct_{ij}로 나타내며, 식은 다음과 같다 [1].

$$ct_{ij} = b_j + e_{ij}$$

b_i는 자원 R_i 가 자신에게 할당된 모든 작업의 실행을 마친 후의 이용가능시간이며, e_{ij}는 작업 T_i을 자원 R_j에서 실행했을 때의 실행시간 (execution time)이다.

3. 관련연구

이질적인 자원으로 구성된 환경에서 독립적인 작업의 스케줄링에 관한 연구는 크게 정적 스케줄링 방식과 동적 스케줄링 방식으로 나눌 수 있다 [11]. 동적 스케줄링 방식은 작업들간의 선후관계가 불분명할 때 사용되며 스케줄링 시기에 따라 작업이 도착하는 즉시 스케줄링하는 온라인방식과 간격을 두고 그 간격에 도착한 작업들을 모두 스케줄링하는 배치방식으로 나눌 수 있다. 온라인방식의 알고리즘에는 MET (Minimum Execution Time), MCT (Minimum Completion Time), SA (Switching Algorithm), KPB (K-Percent Best) 등이 있다 [1]. 본 논문에서는 MET, MCT, KPB 알고리즘과 본 논문에서 제안한 알고리즘을 비교하였다. MET는 작업의 실행시간의 기대값이 가장 작은 자원을 찾아 그 자원에 작업을 할당한다. 즉, T_i를 스케줄링할 때, R에서 e_{ij}가 가장 작은 자원을 찾아 T_i에 할당한다. MET는 무조건 실행시간이 가장 작은 자원을 작업에 할당하기 때문에 자원사용의 불균형을 초래할 수 있다. MET 스케줄링의 시간 복잡도는 $O(m)$이다. MCT는 해당 작업에 대하여 완료시간이 가장 작은 자원을 작업에 할당한다. 즉, T_i을 스케줄링할 때 그 작업에 대하여 R에서 ct_{ij}가 가장 작은 자원을 T_i에게 할당한다. MCT는 간단하면서도 좋은 성능을 보여주며 이 분야에 관한 연구에서 새로운 알고리즘을 제안할 때 비교하는 대상으로 많이 이용되고 있다. MCT의 시간 복잡도는 $O(m)$이다. KPB는 자원 중 실행시간이 가장 작은 $km/100$개의 자원을 선택하여 그 중에서 완료시간이 가장 작은 자원을 할당하게 된다. 여기서 k는 스케줄링 전에 미리 주어지는 상수

값으로 만약, k=100이면 KPB알고리즘의 실행 결과는 MCT와 같게 되고 k=100/m이면 MET와 같게 된다. 이 논문에서는 k=20으로 하여 실험하였다 [1]. KPB의 시간 복잡도는 $O(m\log m)$이다. 이 외에도 한계 값을 적용하여 MET와 MCT를 교대로 사용하는 SA(Switching Algorithm)과 가장 빠른 이용가능시간을 갖는 자원을 작업에 할당하는 OLB (Opportunistic Load Balancing) 방법 등이 있다.

4. 알고리즘

본 논문에서 제시한 MECT (Minimum Execution and Completion Time)알고리즘은 동적 스케줄링의 온라인 방식에 속하며 독립적인 작업들을 그 대상으로 한다. <그림 1>은 알고리즘의 pseudo code이다. 알고리즘을 살펴보면, 먼저 스케줄링 하고자 하는 작업인 T_i와 T_i에 대한 각 자원의 실행시간을 나타내는 벡터가 입력된다. I 에서는 현재 자원들의 이용가능시간 (b_j) 중 가장 큰 값을 찾는다. II에서는 I 에서 구한 값보다 작은 완료시간 (ct_{ij})를 갖는 자원들을 찾는다. III에서는 만약 II의 조건을 만족하는 자원이 존재할 경우 그 자원 중 T_i에 대한 실행시간이 가장 작은 자원을 찾고 IV에서는 그 자원의 인덱스를 출력한다. 만약 II에서 조건을 만족하는 자원이 없을 경우 MCT와 같은 원리로 시스템의 전체 자원 중 가장 작은 완료시간을 찾아 그 값을 갖는 자원의 인덱스를 출력한다.

Algorithm MECT
Input: T_i, $\{e_{i1}, e_{i2}, \ldots, e_{im}\}$, $m=|R|$
Output: k (resource index)
I find $b_{max} = \max_{R_j \in R} b_j$
II find a set of resources, $\hat{R}$ that have completion time smaler than b_{max}
if ($|\hat{R}|>0$)
III find k, such that $e_{ik} = \min_{R_j \in \hat{R}} e_{ij}$
else
III′ find k, such that $ct_{ik} = \min_{R_j \in R} ct_{ij}$
endif
IV return k

그림 1. MECT 알고리즘

<그림 2>는 MECT 알고리즘과 기존의 알고리즘이 5개의 이질적 자원으로 구성된 시스템에서 어떻게 작업을 스케줄링 하는지 보이는 예이다. <그림 2>의 (a)는 스케줄링하고자 하는 작업에 대한 각 자원에서의 실행시간(e)을 나타낸 표이다. 스케줄링의 대상이 되는 작업을 T_i라고 했을 때, 실행시간은 각각 e_{i1}=10, e_{i2}=30, e_{i3}=20, e_{i4}=15, e_{i5}=12이다. <그림 2>의 (b)는 스케줄링 전의 자원 상황과 각 스케줄링 알고리즘이 작업을 스케줄링한 후의 결과를 나타내고 있다. MET는 실행시간 중에서 가장 작은 값을 찾아 그 자원을 작업에 할당하게 되는데 표에서 R_1의 실행시간, e_{i1}=10가 가장 작으므로 R_1를 T_i에 할당한다. MCT는 각 자원의 완료시간 중에서 가장 작은 값을 갖는 자원을 찾아 할당하게 되는데 <그림 2>의 예에서는 R_2의 완료시간, ct_{i2}=5+30=35이 가장 작으므로 R_2를 선택하여 T_i에 할당한다. KPB는 k=40이므로 전체 자원 중에서 가장 작은 실행시간을 갖는 자원 2개를 선택하게 된다. <그림 2>에서는 R_1과 R_5가 선택된다. 최종적으로 두 자원 중 완료시간이 작은 R_5가 선택된다. MECT는 먼저 각 자원의 이용가능시간 중에서 최대값을 찾는다. <그림 2>에서 그 값은 R_4의 이용가능시간, b_4=45이다. 다음으로 자원 중에서 완료시간이 45보다 작은 값을 갖는 것들을 찾는다. ct_{i2}=5+30=35, ct_{i3}=20+20=40이므로 R_2와 R_3이 이에 속한다. 이 자원 중에서 가장 작은 실행시간을 갖는 자원인 R_3 (e_{i3}=20)을 선택하게 되고 R_3을 T_i에 할당하게 된다.

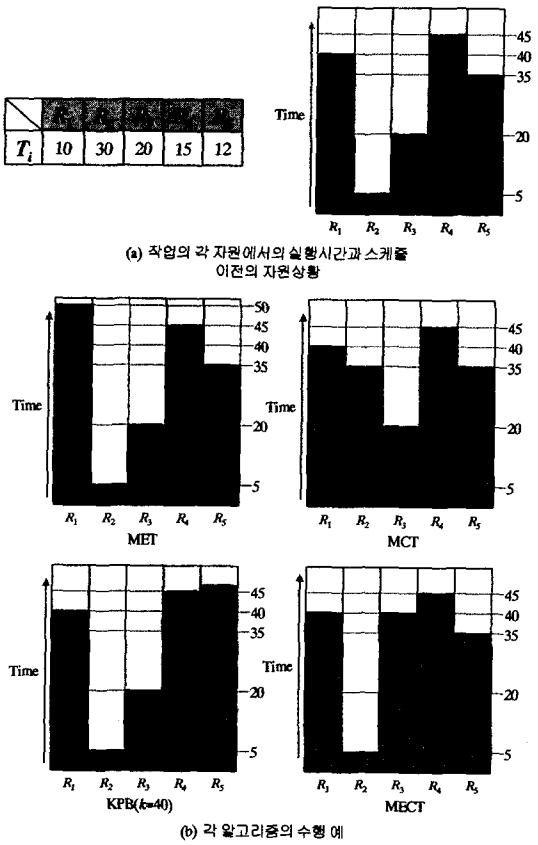

그림 2. 각 알고리즘의 예

5. 실험 및 결과

본 연구에서 사용된 시뮬레이터는 이산형 모델의 실험에 많이 이용되고 있는 SimJava [6]를 사용하여 작성되었다. 실험을 하기 위해 필요한 값들인 작업의 각 자원에서의 실행시간은 미리 계산되어 진다는 가정에 의해서 작성된 작업·자원행렬을 이용하였다. 각 행은 작업을 나타내고 각 열은 자원을 나타내며 행렬의 각 원소는 특정 작업에 대한 특정 자원에서의 실행시간을 나타낸다. 실제 자원의 실행시간은 작업의 프로파일 등 여러 가지 방법을 통하여 구할 수 있다 [1, 7]. 실험에 사용된 작업·자원행렬은 inconsistent 모델과 semi-consistent 모델로 나뉘어진다 [1, 3]. inconsistent 모델은 각 행에 대하여 열의 값들이 무작위로 이루어져 있으며, semi-consistent 모델은 각 행에 대하여 특정 열들이 그 값의 오름차순으로 정렬되어 있다 [1, 3]. 본 연구는 이질적 자원을 고려하여 설계되었으므로 consistent 모델은 제외하였다. 작업·자원행렬의 각 행의 값은 우선 작업을 1과 작업의 이질성사이의 값을 갖는 난수를 이용하여 산출하고 각 값에 다시 1과 자원의 이질성사이의 값을 갖는 난수를 곱하는 식으로 구할 수 있다. 작업의 이질성은 작업

간의 특성의 차이이다. 작업의 이질성이 크다는 것은 작업들의 실행시간이 큰 차이를 보인다는 것을 의미한다. 즉 작업의 이질성을 1000으로 하였을 때보다 3000으로 하였을 때의 작업간 실행시간의 차이가 크게 된다. 본 실험에서는 자원의 개수를 20, 작업의 개수를 1000, 작업의 도착 시간 간격을 100, 그리고 작업의 이질성을 3000으로 고정하여 실험하였다. 결과는 자원 이질성의 값에 따른 변화와 inconsistent 모델과 semi-consistent 모델의 작업·자원행렬에 대하여 각 알고리즘을 적용하여 10번 반복한 결과를 평균을 내어 작성하였다. 자원의 이질성에 따른 결과는 <그림 3>과 <그림 4>에 나타내었다. <그림 3>은 자원의 이질성을 10으로 하였을 때의 결과이며 <그림 4>는 50으로 했을 때의 값이다. 이질성이 10일 때는 KPB가 약간의 차이로 가장 좋은 결과를 보이고 있다. 하지만, 이질성이 50으로 증가하면서 다른 알고리즘과 MET와의 격차가 줄어들고 있다. MET와 다른 알고리즘의 makespan의 차이가 줄어드는 것은 자원의 이질성의 값이 클수록 자원간의 실행시간의 격차가 커지기 때문이다. 이것은 자원의 이질성의 차이가 클수록 실행시간이 알고리즘의 성능에 미치는 영향이 커진다는 것을 의미한다. MECT 알고리즘과 MCT, KPB와의 makespan의 차이는 이질성이 증가함에 따라 벌어지고 있는 것을 알 수 있다.

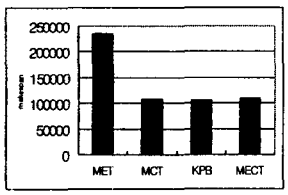

그림 3. 자원이질성이 10일 때의 makespan

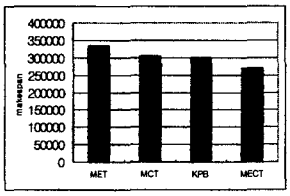

그림 4. 자원 이질성이 50일 때의 makespan

<그림 5>은 semi-consistent 모델 작업·자원행렬을 사용하여 실험한 결과이다. MECT 알고리즘이 가장 좋은 결과를 보이고 있다. MET의 makespan의 값이 가장 크게 나오고 MECT 알고리즘이 MCT보다 좋은 결과를 보이고 있다. 본 논문의 semi-consistent 모델 작업·자원행렬은 각 행의 홀수 번째 열에 대하여 오름차순으로 정렬하여 구하였다 [3]. 이러한 작업·자원행렬에서 실제 자원의 할당되는 모습을 보면 MET와 KPB는 특정 자원에 편중되게 되어 MCT나 MECT 알고리즘보다 좋지 않은 결과를 보인다.

6. 결론

이질적 자원으로 구성된 환경은 어떤 작업을 실행할 때 그 작업이 요구하는 특성에 따라 다양한 실행시간의 차이를 보인다는 점에서 균질의 자원으로 구성된 환경과 다르다. 각 자원에서의 실행시간이 이질적 자원으로 구성된 환경에서의 독립적인 작업들을 스케줄링하는 알고리즘을 설계하는데 중요한 부분을 차지하게 됨을 실험을 통하여 보였다. 본 논문에서 제안한 알고리즘은 MCT를 기본으로 따르는 반면, 자원의 실행시간을 고려하여 설계되었다. semi-consistent 모델 작업과 inconsistent 모델 작업 모두에서 좋은 결과를 보이고 있으며 또한 자원의 이질성이 증가함에 따른 알고리즘보다 좋은 결과를 보인다.

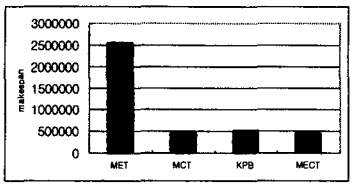

그림 5. semi-consistent 모델 행렬일 때의 makespan

참고문헌

[1] M. Maheswaran, S. Ali, H. J. Siegel, D. Hensgen, and R. F. Freund, "Dynamic Matching and Scheduling of a Class of Indenpendent Tasks onto Heterogeneous Computing Systems," *Proc. of the 8th Heterogeneous Computing Workshop*, pp. 30-44, April, 1999.

[2] I. Foster, C. Kesselman, and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," *Journal of High-Performance Computing Applications*, vol. 15, no. 3, pp. 200-222, 2001.

[3] T. D. Braun, H. J. Siegel, and Noah Beck, "A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems," *Journal of Parallel and Distributed Computing*, vol. 61, pp. 810-837, 2001.

[4] 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, April, 1977.

[5] M. Pinedo, *Scheduling: Theory, Algorithms, and Systems*, Prentice Hall, NJ, 1995.

[6] F. Howell and R. McNab, "SimJava: A Discrete Event Simulation Package For Java With Applications In Computer Systems Modelling," *Proc. of the 1st International Conference on Web-based Modelling and Simulation*, January, 1998.

[7] A. A. Khokhar, V. K. Prasanna, M. E. Shaaban, and C. L. Wang, "Heterogeneous Computing: Challenges and Opportunities," *Journal of the IEEE Computer*, vol. 26, pp. 18-27, June, 1993.

[8] R. Buyya, J. Giddy, and D. Abramson, "An Evaluation of Economy-based Resource Trading and Scheduling on Computational Power Grids for Parameter Sweep Applications", *Proc. of the 2nd International Workshop on Active Middleware Services*, August, 2000.

[9] H. Barada, S. M. Sait, and N. Baig, "Task Matching and Scheduling in Heterogeneous Systems using Simulated Evolution," *Proc. of the 15th Parallel and Distributed Processing Symposium*, pp. 875-882, 2001.

[10] B. Hamidzadeh, Lau Ying Kit, and D.J. Lilja, "Dynamic Task Scheduling using Online Optimization, " *Journal of Parallel and Distributed Systems*, vol. 11, pp. 1151-1163, 2000.

[11] M. Maheswaran, T. D. Braun, and H. J. Siegel, "Heterogeneous Distributed Computing," *Encyclopedia of Electrical and Electronics Engineering*, J. G. Wdbster, editor, John Wiley & Sons, vol. 8, pp. 679-690, 1999.