유전자 알고리즘을 이용한 Backfilling 스케줄러의 작업 패킹 기법

이효영°, 이동우, R.S.Ramakrishna
광주과학기술원 정보통신공학과
{hyylee°, leepro, rsr}@gist.ac.kr

Job Packing Technique in Backfilling Scheduler using Genetic Algorithm

HyoYoung Lee°, DongWoo Lee, R.S.Ramakrishna
Department of Information and Communications,
Gwangju Institute of Science and Technology

요 약

본 논문에서는 병렬 컴퓨팅의 스케줄링 시스템인 EASY Backfilling 알고리즘에 기반한 작업 패킹 기법의 최적화에 대해 논의한다. 이를 위해 최적의 작업 집합을 구성하기 위한 탐색 기법으로 유전자 알고리즘을 활용하여 작업 패킹을 효율적으로 수행함과 동시에, 적은 노드를 요청한 작업에 가중치를 부여함으로써 다수 작업의 동시 실행을 우선 고려하게 하였다. 스케줄링 정책은 컴퓨터 성능에 직접적인 영향을 미치는 요소이기 때문에 시스템 부하별로 각 워크로드의 평균 대기 시간을 측정한 실험을 통해 제안 기법이 전반적인 병렬 컴퓨팅의 성능을 개선함을 확인하였다.

1. 서 론

기상 예측, 인간 유전자 지도와 같이 계산 집중적 컴퓨팅을 요구하는 복잡한 수학적 문제의 해결을 위해, 멀티 프로세서 시스템으로의 확장은 그 영역을 꾸준히 넓혀 왔다. 이는 강력한 병렬 컴퓨팅 패러다임으로의 변화를 자연스럽게 유도해 왔으며, 최근에는 계산 그리드 컴퓨팅의 한 부분으로 자리잡아 가고 있다.

병렬 컴퓨터의 시스템 사용율을 높이기 위한 다양한 스케줄링 알고리즘 기법들이 소개되었지만, 단편화(Fragmentation)와 기아 현상(Starvation)은 해결해야 할 과제로 남아 있다. 이에 병렬 프로세스를 위한 작업 스케줄링 기법으로서 Backfilling 알고리즘이 제안되었다. Backfilling 기법은 크게 Conservative Backfilling과 EASY Backfilling의 두 가지 형태로 나누어지며, 각 알고리즘은 이전에 요청된 작업의 예약을 지연시키지 않는 조건하에서 나중에 요청된 적은 노드의 작업이 이전에 요청된 큰 노드의 작업보다 먼저 실행될 수 있도록 하는 것이다[1]. 이 기법의 특징은 예약 기법을 통한 기아 현상의 방지와 함께, 적은 노드를 요청한 작업이 유휴 자원을 활용함으로써 기존 알고리즘에서 발생되는 단편화를 줄일 수 있도록 보장하는 것이다. 하지만, 기존 Backfilling 알고리즘의 한계는 장치의 유휴 용량이 최소화됨을 보장하지 않는다는 점이다.

이에 본 연구에서는 각 스케줄링 단계에서 큐에 대기중인 작업들을 대상으로 시스템 사용율을 최대화할 수 있는 작업들의 집합을 구성한다. 이렇게 구성된 작업 집합을 우선 실행함으로써 병렬 시스템상에서 작업의 대기 시간을 단축시키는 것을 목적으로 한다. 이 때, 작업 집합을 구성하는 작업 패킹(Job Packing) 기법으로 bin-packing 문제를 효과적으로 해결하는데 널리 쓰이는 유전자 알고리즘[5]을 활용하고, 적합도 함수를 이용하여 실행 가능한 작업 집합 가운데 최적의 작업 집합을 선택하도록 한다.

본 논문에서는 기존의 변형된 Backfilling 알고리즘을 소개하고 작업 패킹과 적용한 유전자 알고리즘의 설계를 기술한다. 또한, 제안한 방법의 실험 결과에 대해 분석하고 결론과 향후 계획으로 글을 맺는다.

2. 관련 연구

기존의 Backfilling 연구는 크게 두 가지로 진행되어 왔다. 하나는 사용자의 작업을 처리하는데 소요되는 시간에 대한 예측 기술의 연구이고 다른 하나는 요청 작업의 재배치 알고리즘에 관한 것이다. Backfilling 기법의 특징은 작업 요청 시에 사용자로부터 예상되는 작업 소요 시간을 입력 받아 스케줄링을 위한 중요 매개 변수로 사용한다는 것이다. 연구 결과에 따르면 사용자의 작업 예측 시간에 대한 과대평가가 실제 실행시간으로 예측한 경우보다 더 나은 성능을 제공한다[2]. 효율적인 시스템 사용을 위해 사용자가 요청한 작업 소요 시간에 따라 요청 작업을 범주화 시켜, 다중 큐에 각 작업을 재배치하고 우선 순위를 부여하는 연구도 진행되어 왔다[3]. 이 기법에서는 다중 큐를 이용한

Backfilling이 모든 작업 범주에서 단일 큐를 이용한 Backfilling보다 우수하다는 것을 입증하였다. 이 밖에도 본래 Backfilling 기법을 변형한 다양한 알고리즘의 연구가 활발하게 진행되어 왔다.

3. 작업 패킹 최적화를 위한 유전자 알고리즘

유휴 자원의 사용율 최적화를 위한 작업 패킹 기법으로 탐색 기반 기법을 Backfilling에 적용한 연구가 있었다[6]. 이 알고리즘의 복잡도는 대기 작업의 크기에 영향을 많이 받기 때문에, 부하가 집중될 경우에 스케줄러의 성능 저하가 발생하는 단점이 있다. 이에 계산에 상당한 시간을 소요하는 분기 한정 탐색법(branch & bound search methods)의 성능을 개선하기 위하여, 효율적인 유전자 알고리즘을 적용한 작업 패킹 기법을 제안한다. 또한, 작업 선택을 위한 패킹 원칙으로 유휴 자원 공간을 최소화하기 위해 각 구획에 적합한 작업의 우선 고려와 다수 작업의 동시 수행을 목표로 적합도 함수를 정의한다.

3.1. 작업 패킹(Job Packing)

Backfilling 기법은 그림1 (a)와 같이 시간과 노드의 2차원 작업 공간으로 표현될 수 있다. 유휴 자원 공간(그림1 (b))의 작업 배치를 고려하기 위해 하나의 구획은 그림1 (c)처럼 몇 개의 구획으로 분할되도록 구성할 수 있다.

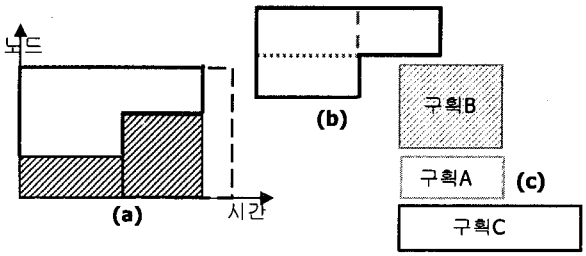

그림1. **Gantt** 차트의 유휴 노드 분할 방법 예시

Backfilling 기법의 특성상 요청된 작업이 필요로 하는 노드의 크기는 작업이 실행될 때 고정되지만, 실제 작업을 수행하는데 소용되는 시간은 예측이 어렵기 때문에 요청한 작업의 노드별로 가중치를 부여한다. 또한, 큰 구획에 적합한 작업보다는 작은 구획(구획A)에 적합한 작업을 먼저 고려함으로써 각 스케줄링 단계에서 노드의 낭비를 최소화할 수 있게 한다.

3.2. 유전자 알고리즘 설계

작업 패킹을 위한 유전자 알고리즘의 세부 설계를 다음과 같이 형식화하여 정의한다.

문제 사례: 대기 작업 – $wj_1, wj_2, \cdots, wj_n$

요청 노드 수 – $p_1, p_2, \cdots, p_n$

적합한 해 : $\sum_{i=1}^{n} px_i \le$ 유휴 노드 수를 만족하는 벡터

$$\vec{x} = (x_1, x_2, \ldots, x_n),\ x_i \in \{0,1\}$$

적합도 함수 : $F(x) = \sum_{i=1}^{n} wx_i$, $w_i = \begin{cases} 3\alpha & p_i \in \text{구획A} \\ 2\alpha & p_i \in \text{구획B} \\ 1\alpha & p_i \in \text{구획C} \end{cases}$

유전자 알고리즘의 개체군 생성 단계에서 0과 1의 랜덤한 조합으로 구성된 벡터를 이용하여 적합도 함수를 계산하고, 적합도 함수를 최대화하면서 적합한 해의 조건을 만족하는 벡터를 최적의 작업 벡터로 정의한다. 또한, 사용 가능한 노드 수를 초과하는 벡터의 경우에는 요청한 작업 노드와 작업 소요 시간으로 구성된 2차원 공간 면적을 비교하여 가장 큰 작업을 제거한다. 이는 상대적으로 많은 자원을 요청한 작업에 감점을 부과함으로써 다수 작업을 동시에 수행하는 효과를 얻기 위함이다.

그림2는 새로운 작업의 입력이나 기존 실행 작업의 종료와 같은 이벤트 발생시, 대기 작업의 Backfill 여부를 확인하기 위한 알고리즘의 일부를 순서도로 나타낸 것이다. 대기 작업의 개수를 고려하여 작업 패킹의 필요성이 존재하지 않을 경우에는 기존의 Backfilling 알고리즘을 이용하고 그 이외에는 유전자 알고리즘 기법으로 최적화된 작업 집합을 구한다.

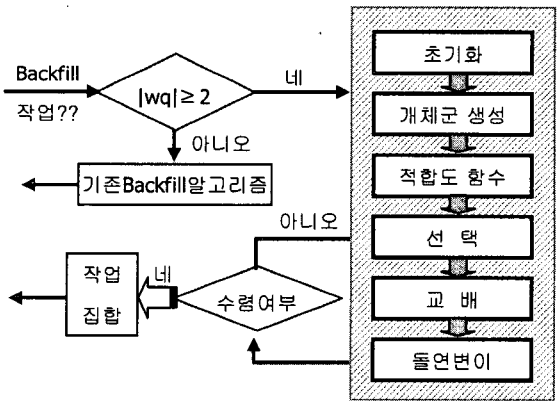

그림2. **Backfilling**에의 유전자 알고리즘 도입

4. 실험 결과 및 분석

본 논문에서는 EASY Backfilling 알고리즘을 이용한 온라인 시뮬레이션 기법으로, 3개의 워크로드[4]을 선별하여 제안 기법의 성능을 비교 및 검증한다. 유전자 알고리즘의 초기 값은 개체군의 크기를 10, 염색체의 길이는 각 스케줄 단계에서 backfill 가능한 대기 작업의 크기로 설정했다. 또한, 토너먼트 선택 기법으로 1-점 교배를 사용하고 돌연변이가 발생할 확률은 0.05로 고정시켰다. 실험에서는 적합도 함수를 $\alpha = 1$로 설정하여 적은 노드를 필요로 하는 작업에 대한 가중치를 조절하고 다수 작업의 동시 실행을 최대한 고려하였다. 최종 결과값은 각 워크로드에 대해서 부하별로 각각 10번의 실험을 수행하여 얻은 평균값을 기록하였다.

그림3은 워크로드의 부하 상태별로 평균 대기 시간을 비교한 확률이 낮은 워크로드에서는 성능이 악화되는 경향을 보였다.

것이다. 기준 부하 상태(0.6)에서는 제안 기법이 기존 Backfilling 기법과 비슷한 양상을 보이지만, 부하 집중시(0.9)에 제안 기법이 보다 우수한 성능 향상을 나타낸다. 그러나, KTH 로그에서는 부하 증가 시에 성능 악화를 보이는데, 이는 작업 크기별 분포율을 비교해 볼 때 적은 수의 노드를 필요로 하는 작업의 요청 비율이 KTH 로그에서 상대적으로 낮기 때문인 것으로 확인되었다.

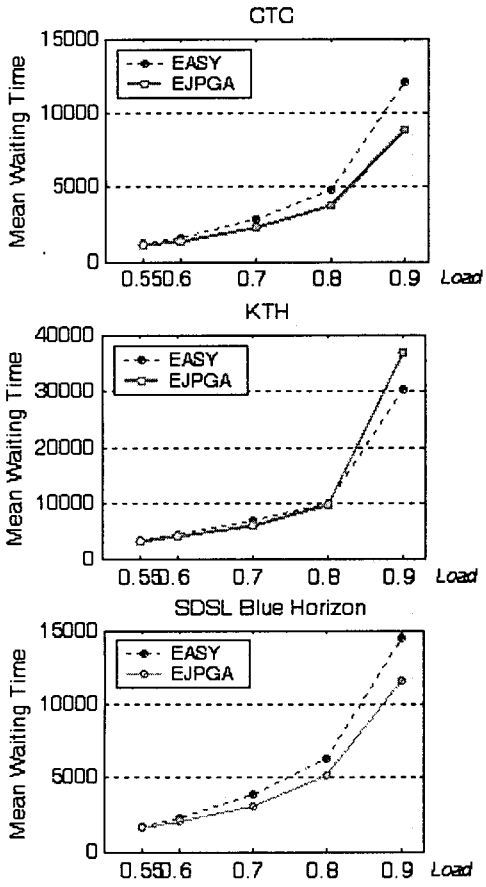

그림3. 워크로드 부하별 평균 대기 시간 비교

한편, 표1은 워크로드별 부하 상태에 따른 큐에 대기중인 요청 작업의 길이를 나타낸 것이다. 평균 큐의 길이(MQL)는 식(1)에 의해 계산한다.

$$MQL = \frac{1}{T}\int_0^T Queue\_length(t)dt \quad \textbf{(1)}$$

평균 큐 길이에 있어서도 기준 부하 상태보다는 부하가 집중될 경우에 20% 이상의 감소량을 보여 의도적으로 다수 작업의 동시 수행을 우선 고려하여 스케줄링 되었음을 확인할 수 있었다. 또한, 작은 크기의 작업에 가중치를 부여한 본 제안 기법이, KTH 로그에서와 같이 상대적으로 적은 노드를 요청한 작업의

표1. 워크로드 부하별 평균 큐 길이 비교

워크로드	부하	EASY	EJPGA
CTC	0.6 (low)	4.68	3.95
	0.9 (high)	53.22	39.19
KTH	0.6	3.71	3.46
	0.9	38.96	47.10
SDSC Blue Horizon	0.6	5.31	4.64
	0.9	50.46	40.24

5. 결 론

본 논문에서는 병렬 컴퓨팅에서 스케줄링 시스템의 성능 향상을 위한 요소로서 작업 패킹 기법을 고려하고 이를 위해 탐색 기법에 우수한 유전자 알고리즘의 활용을 제안하였다. 실험을 통해서 제안 기법이 기존의 EASY Backfilling 기법과 비교하여, 요청한 작업의 대기 시간을 감소시킴으로써 시스템의 성능을 향상시킴을 확인하였다. 향후에는 수학적인 분석을 위한 워크로드 모델을 개발하여 유전자 알고리즘의 적합도 함수를 보완하는 연구를 진행할 계획이다.

참고문헌

[1] A. W. Mu'alem and D. G. Feitelson, "Utilization, Predictability, Workloads, and User Runtime Estimates in Scheduling the IBM SP2 with Backfilling," IEEE Trans. on Parallel and Distributed Computing, Vol. 12, pp. 529-543, 2001.

[2] D. Zotkin, P. Keleher, "Job-Length Estimation and Performance in Backfilling Schedulers," In Proc. of the 8th High Performance Distributed Computing Conf. 1999.

[3] B. G. Lawson and E. Smirni, "Multiple-Queue Backfilling Scheduling with Priorities and Reservations for Parallel Systems," Job Scheduling Strategies for Parallel Processing, D. G. Feitelson and L. Rudolph(eds.), LNCS. Vol. 2537, pp. 72-87, 2002.

[4] Parallel Workloads Archive, http://www.cs.huji.ac.il/labs/parallel/workload.

[5] M. Mitchell, "An Introduction to Genetic Algorithms," 3rd ed., MIT Press, 1998.

[6] E. Shmueli and D.G. Feitelson, "Backfilling with Lookahead to Optimize the Performance of Parallel Job Scheduling," Job Scheduling Strategies for Parallel Processing, D.G. Feitelson and L. Rudolph(eds.), pp. 228-251, 2003.