HPC 환경에서의 스케줄링을 위한 강화학습 및 휴리스틱 알고리즘의 비교 분석

안성배, 이재현, 박보현, 오상윤*
아주대학교

aqua1765@ajou.ac.kr, dlwogus8888@ajou.ac.kr, pbh7080@ajou.ac.kr, *syoh@ajou.ac.kr

Comparative Analysis of Reinforcement Learning Algorithms and Heuristic Algorithms for Scheduling in HPC Environments

Seongbae An, Jaehyun Lee, Bohyun Park, Sangyoon Oh *
Ajou University

요 약

본 논문은 고성능 컴퓨팅(HPC) 환경에서 효율적인 스케줄링을 위한 방법으로 강화학습의 적용 가능성을 탐구한다. 이를 위해 일반적인 휴리스틱 방식인 First-Come First-Served (FCFS), Short Job First (SJF)를 적용한 스케줄러와 Proximal Policy Optimization (PPO) 강화학습 알고리즘으로 학습한 스케줄러의 성능을 실험적으로 비교 분석한다. 실험 결과에서 강화학습 스케줄러는 4 개의 작업 데이터에 대해 기존 휴리스틱 방식과 유사한 자원 활용도를 보이면서 작업 처리 속도 면에서 더 안정적임을 확인하였다. 이러한 결과를 통해 강화학습이 동적이고 복잡한 HPC 환경에서의 스케줄링 문제에 효과적으로 적용될 수 있음을 보였다.

Ⅰ. 서 론

최근 인공지능(AI)과 같은 복잡한 과학, 공학 문제를 해결하기 위해서 고성능 컴퓨팅(HPC) 환경에서의 애플리케이션 사용이 증가하고 있다. 따라서, HPC 클러스터 환경에서의 작업을 수행할 시 실행 시간을 단축하고 시스템 자원의 활용도를 높이기 위해 효율적인 스케줄링 기술의 필요성이 점차 커지고 있다.

일반적인 HPC 환경에서는 보편성 및 예측 가능성 등의 이유로 First-Come First-Served (FCFS), Short Job First (SJF)과 같은 휴리스틱 방식에 기반을 둔 스케줄러를 많이 사용한다. 하지만, 대규모 시스템이나 실시간 시스템 등 동적이고 복잡한 환경에서는 기존의 스케줄러의 성능이 낮아지는 문제가 있어서, 더 높은 성능(즉, high throughput, short waiting time)의 스케줄링을 위해 기계학습 기반의 스케줄링 방안이 제안되고 있다[1]. 특히, 강화학습은 에이전트가 경험을 바탕으로 직접 의사 결정 정책을 직접 학습하고 동적 환경에 적응할 수 있으므로 자원 관리에 적합하다는 연구 결과[2]가 있다.

본 논문에서는 강화학습을 스케줄링에 적용한 선행 연구[3]를 바탕으로, Proximal Policy Optimization (PPO) 알고리즘을 활용한 강화학습 스케줄러와 기존의 FCFS 그리고 SJF 기반의 휴리스틱 스케줄러를 Average Bounded Slowdown (ABS)과 Resource Utilization 를 지표로 하는 결과 분석을 통해 강화학습 기반 스케줄링의 유용성을 확인하고자 한다.

Ⅱ. 이론적 배경

본 논문에서는 강화학습을 적용하여 스케줄링 문제를 해결하는 방법을 탐구한다. 강화학습은 에이전트가 주어진 상태에서 어떤 행동을 수행하면 환경으로부터 다음 상태와 보상을 받고, 이를 바탕으로 최대 보상을 얻기 위한 행동을 학습하는 과정이다. 싱글 에이전트 강화학습 기술은 주로 Q-Learning 과 정책 경사(Policy Gradient) 방법으로 나뉜다[4].

Q-Learning 은 상태와 행동의 쌍에 대해 Q 값을 추정하여 해당 상태에서 어떤 행동을 취할지를 결정하는 방식이다. ε-탐욕 정책을 통해 특정 상태에서 ε의 확률로 무작위 행동을 선택하고, $(1-\varepsilon)$ 확률로 현재 추정된 최적의 행동을 선택하여 학습을 진행한다.

정책 경사 기반 강화학습은 상태에 따라 행동을 결정하는 정책을 직접 구현한다. 이 정책은 관측 값을 입력 받아서 행동 값을 출력하는 함수로, 매개변수 θ(신경망의 가중치와 편향)에 의해 정의된다. 목적 함수, 즉 정책에 따라 행동 시 얻게 되는 누적 보상의 기대치에 대한 경사를 계산하고, 이를 상승하는 방향으로 θ를 업데이트한다. 이 과정은 경사가 수렴하거나 최대 타임 스텝에 도달할 때까지 반복된다.

본 논문에서 사용한 Proximal Policy Optimization (PPO) 알고리즘은 정책 경사 기반 강화학습의 한 형태이다. PPO 는 목적 함수의 Clipping 기법을 사용하여 정책 업데이트의 크기를 제한하여 너무 급격하게 변동이 이루어지지 않도록 함으로써 학습의 안정성을 향상시킨다. 이 기법은 학습 과정을 안정화하고 계산을 단순화하여 알고리즘의 구현을 용이하게 한다.

Ⅲ. 실험 조건

실험에 사용된 스케줄러는 OpenAI 의 Spinning Up[5]에서 제공하는 PPO 알고리즘을 Tensorflow 로 구현한 스케줄러[3]를 사용했다. 이 스케줄러는 epoch 단위로 훈련되며, 각 epoch 에서는 환경에서 200 개의 경로(trajectories)를 랜덤하게 샘플링한다. 각 경로는 256 개의 연속된 작업 스케줄링 결정을 포함하며, 이는 에이전트와 환경 간의 상호작용을 나타내고, 얻은 보상을 에이전트 업데이트에 활용한다.

실험에 사용된 작업 데이터(Job trace)는 두 그룹으로 나뉘어 있다. 첫 번째 그룹은 [6]에서 제안된 워크로드 모델을 기반으로 생성된 합성 데이터이고, 두 번째 그룹은 Standard Workload Format[7]에서 가져온 실제 데이터이다.

OS	Ubuntu 20.04.3 LTS
CPU	INTEL Core 13th i7-13700KF
RAM	64GB
Storage	1TB
GPU	RTX 4090 D6X 24GB

<표 1. 실험 환경>

합성 데이터	Lublin-256, Lublin-256-new
실제 데이터	SDSC-BLUE, HPC2N

<표 2. 실험 데이터>

Ⅳ. 실험 결과 및 분석

Data	FCFS	SJF	PPO
Lublin-256	7273.77	**277.35**	316.12
Lublin-256-new	7842.47	787.89	**722.24**
SDSC-BLUE	1727.54	2680.55	**1644.32**
HPC2N	297.18	157.71	**108.85**

<표 3. Average bounded slowdown>

Data	FCFS	SJF	PPO
Lublin-256	**0.868**	0.778	0.809
Lublin-256-new	0.587	**0.593**	0.514
SDSC-BLUE	**0.682**	0.661	0.671
HPC2N	0.640	**0.641**	0.638

<표 4. Resource utilization>

학습 과정에서는 256 개의 연속된 작업 스케줄링을 가진 경로를 사용했다. 평가를 위해서 학습 중 발생할 수 있는 과적합의 영향을 피하기 위해서 1024 개의 긴 연속된 작업으로 이루어진 무작위 작업 순서를 10 회 반복하여 스케줄링하고 이를 평가했다. 위에서 제시된 <표 3>와 <표 4>의 값은 모두 10 회 동안 스케줄링을 진행한 결과의 평균 값이다.

Average bounded slowdown 은 작업의 완료 시간이 예상보다 지연되는 정도를 확인하는 지표이다. 즉, 작은 값을 가지면 작업이 더 효율적으로 진행이 된 것이다. Lublin-256 데이터에서는 PPO 가 SJF 에 비해 더 긴 시간이 걸렸지만 차이는 크지 않고, 다른 데이터에서는 모두 PPO 가 가장 작은 slowdown 값을 가진다.

Resource utilization 측면에서는 PPO 가 더 낮은 활용도를 보인다. 이는 FCFS 와 SJF 가 자원을 더 효율적으로 사용하는 것을 보인다. 하지만 그 차이가 최소 0.003, 최대 0.079 로 크게 차이가 나지 않는다.

즉, FCFS 와 SJF 를 활용한 휴리스틱 스케줄링과 PPO 로 학습시킨 강화학습 스케줄링을 비교한 결과, 자원 활용도 면에서 강화학습 스케줄링이 휴리스틱 스케줄링과 비슷한 자원 활용도를 유지하면서, 작업의 처리 속도 면에서는 더 효율적임을 보인다.

Ⅴ. 결론 및 토의

제한된 조건이지만, 실험 결과를 통해 FCFS 와 SJF 는 데이터의 종류에 따라서 처리 성능이 달라질 수 있음을, 즉 데이터 특성에 따라 기대성능이 달라지는 것을 보였다. 반면, PPO 를 활용한 스케줄링은 데이터의 종류에 상관없이 안정적으로 처리 성능을 보였다. 이는 강화학습의 에이전트가 경험을 바탕으로 의사 결정 정책을 직접 학습하고 동적 환경에 적응할 수 있으며, 자원 관리에 대해 가지는 장점을 실험을 통해 확인할 수 있는 결과이다.

실험을 통해 강화학습으로 학습한 스케줄링의 작업 처리 속도면에서의 장점을 확인할 수 있었지만, 실제 환경에서의 적용 가능성과 그 효과를 검증하기 위해서는 추가적으로 실험 및 분석이 필요하다. 후속 연구를 통해서 다양한 강화학습 알고리즘을 스케줄링에 활용한 실험과 더 많은 작업 데이터로 스케줄링을 수행한 결과의 분석을 진행하고자 한다. 이를 통해 강화학습 기술이 HPC 환경의 복잡한 스케줄링 문제에 실질적으로 기여할 수 있는지를 확인할 수 있을 것이다.

ACKNOWLEDGMENT

본 연구는 2023 년도 한국연구재단 기본연구사업 (NRF-2021R1F1A1062779) 및 초고성능컴퓨팅 SW 생태계조성사업(RS-2023-00283799)의 연구비 지원으로 수행하였습니다.

참 고 문 헌

[1] Haldun Aytug, Siddhartha Bhattacharyya, Gary J. Koehler, and Jane L. Snowdon, “A Review of Machine Learning in Scheduling” in IEEE TRANSACTIONS ON ENGINEERING MANAGEMENT, VOL. 41, NO. 2, MAY 1994, pp. 165 - 171.

[2] Fengcun Li, Bo Hu, “DeepJS: Job Scheduling Based on Deep Reinforcement Learning in Cloud Data Center” in ICBDC ’19: Proceedings of the 4th International Conference on Big Data and Computing, May 2019, pp. 48-53.

[3] Zhang, Di and Dai, Dong and He, Youbiao and Bao, Forrest Sheng and Xie, Bing." RLScheduler: an automated HPC batch job scheduler using reinforcement learning," IEEE International Conference for High Performance Computing, Networking, Storage and Analysis, 2020, pp. 1-15.

[4] Jang, S.Y. ; Yoon, H.J. ; Park, N.S. ; Yun, J.K. ; Son, Y.S. “Research Trends on Deep Reinforcement Learning” Electronics and Telecommunications Trends, 1225-6455(pISSN), Volume 34 Issue 4, Pages.1-14, 2019

[5] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms”, arXiv:1707.06347, 2017.

[6] Lublin and D. G. Feitelson, “The workload on parallel supercomputers: modeling the characteristics of rigid jobs,” Journal of Parallel and Distributed Computing, vol. 63, no. 11, pp. 1105- 1122, 2003.

[7] D. G. Feitelson, D. Tsafrir, and D. Krakov, “Experience with using the Parallel Workloads Archive,” Journal of Parallel and Distributed Computing, vol. 74, no. 10, pp.2967- 2982, 2014.