The Journal of The Institute of Internet, Broadcasting and Communication (IIBC)
Vol. 24, No. 3, pp.63-67, Jun. 30, 2024. pISSN 2289-0238, eISSN 2289-0246

https://doi.org/10.7236/JIIBC.2024.24.3.63
JIIBC 2024-3-10

유전 알고리즘을 이용한 클라우드 환경의 인공지능 워크로드 스케줄링

Scheduling of Artificial Intelligence Workloads in Could Environments Using Genetic Algorithms

권석민[*], 반효경[**]

Seokmin Kwon[*], Hyokyung Bahn[**]

요 약 최근 스마트 물류, 핀테크, 엔터테인먼트 등 다양한 산업 분야의 인공지능 워크로드들이 클라우드 상에서 실행되고 있다. 본 논문은 이기종 GPU 클러스터로 구성된 다중 테넌트 클라우드 시스템에서 다양한 인공지능 워크로드가 실행될 때 발생하는 스케줄링 문제를 다룬다. 전통적인 스케줄링은 이러한 환경에서 GPU 이용률을 크게 저하시켜 시스템의 성능을 떨어뜨린다. 이러한 문제를 해결하기 위해, 본 논문에서는 유전 알고리즘 기반의 최적화 기법을 사용하는 새로운 스케줄링 접근 방식을 제안하고, 이를 프로세스 기반 이벤트 시뮬레이션 프레임워크에 구현하였다. 알리바바의 MLaaS 클러스터에서 수집한 광범위한 인공지능 작업들의 트레이스를 재현하는 실험을 통해 제안하는 스케줄링이 기존 스케줄링에 비해 GPU 이용률을 크게 개선함을 확인하였다.

Abstract Recently, artificial intelligence (AI) workloads encompassing various industries such as smart logistics, FinTech, and entertainment are being executed on the cloud. In this paper, we address the scheduling issues of various AI workloads on a multi-tenant cloud system composed of heterogeneous GPU clusters. Traditional scheduling decreases GPU utilization in such environments, degrading system performance significantly. To resolve these issues, we present a new scheduling approach utilizing genetic algorithm-based optimization techniques, implemented within a process-based event simulation framework. Trace driven simulations with diverse AI workload traces collected from Alibaba's MLaaS cluster demonstrate that the proposed scheduling improves GPU utilization compared to conventional scheduling significantly.

Key Words : task scheduling, artificial intelligence, machine learning, cloud, genetic algorithm

Ⅰ. 서 론

AI 산업의 발전과 대용량 데이터의 급격한 증가로 AI 응용을 클라우드 상의 GPU 클러스터에서 실행하는 MLaaS(Machine Learning as a Service) 서비스가 증가하고 있다[1]. 특히, 개인용 모바일 장치에서 산업용 서비스에 이르는 다양한 분야에서 AI 워크로드가 급증함에 따라[2, 3, 4], 이를 실현하기 위한 학습 및 추론의 상당 부

[*]정회원, 이화여자대학교 컴퓨터공학과
[**]정회원, 이화여자대학교 컴퓨터공학과
접수일자 2024년 5월 7일, 수정완료 2024년 5월 29일
게재확정일자 2024년 6월 7일

Received: 7 May, 2024 / Revised: 29 May, 2024 /
Accepted: 7 June, 2024
[*]Corresponding Author: bahn@ewha.ac.kr
Dept. of Computer Engineering, Ewha University, Korea

분을 클라우드 상의 GPU 클러스터에 의존하고 있다[5].

이와 같은 다양한 AI 워크로드 및 GPU 구성으로 인해 자원 관리와 태스크 스케줄링에 상당한 어려움이 초래되고 있다. 태스크 스케줄링은 모바일 장치에서 대규모 산업용 시스템에 이르는 광범위한 시스템 환경에서 널리 연구돼 왔다[6, 7]. 그러나, AI 워크로드는 전통적인 워크로드와는 상이한 자원 사용 패턴을 나타내는 것으로 알려져 있어 시스템 자원을 효율적으로 관리하는 데에 어려움이 따른다[8, 9]. 특히, 대규모 GPU 클러스터 환경에서 AI 워크로드를 실행할 때 GPU 이용률이 크게 저하되는 문제가 발생하는 것으로 알려져 있다[10]. 이러한 낮은 활용도는 소규모의 AI 추론 작업이 끊임없이 생성되면서 더욱 심각해지는 추세이며, 이와 같은 상황은 향후 지속될 것으로 예상된다.

이에 대처하기 위해, NVIDIA의 Ampere 아키텍처에서는 MIG(Multi Instance-GPU) 기능을 통해 태스크 간에 GPU를 공유하는 방식을 도입하고 있다[11]. 이러한 방식은 GPU 이용률을 어느 정도 개선할 수 있는 효과가 있으나, 여전히 대부분의 GPU에서 부분 할당으로 인해 조각화 문제(fragmentation)가 발생하며, 이로 인해 사용 가능한 전체 GPU 용량 대비 실제 이용률은 여전히 낮은 상황이 지속되고 있다.

조각화 문제를 해결하기 위해 전통적으로 Bin-Packing 방식이 제안된 바 있다[12]. 개념적으로 GPU 환경에서의 태스크 스케줄링은 Bin-Packing 문제와 유사하게 모델링될 수 있으나, 전통적인 Bin-Packing 문제는 GPU 공유 환경에서의 스케줄링 문제와 근본적인 요구 사항의 차이가 있어 다중 GPU 클러스터 시스템에서의 AI 워크로드 스케줄링에 직접 사용하기에는 한계가 있다.

이에 본 논문에서는 GPU 스케줄링을 두 가지 유형의 Bin-Packing 문제로 모델링하는 시도를 하였다. 첫 번째 모델은 여러 GPU들을 하나의 논리적인 머신으로 모델링하여 GPU 전체의 용량을 통합하고, 하나의 논리적인 GPU 상에서 AI 워크로드를 스케줄링하는 방식이다. 이러한 방식은 이론적으로 단순한 장점이 있지만, 개별 GPU에 발생하는 조각화 문제를 고려하지 못해 실제 스케줄링에 사용하기에는 한계가 있다. 두 번째 모델은 각 GPU를 별도의 자원으로 간주하고 CPU, 메모리 등과 다르게 GPU 머신들 간에 발생하는 의존성을 고려하는 방식이다. 이러한 모델에서는 하나의 GPU 머신에서 실행되는 작업이 다른 머신에서의 실행에 의존적이라는 점에서 전통적인 Bin-Packing 전략을 단순히 확장하여 실세계의 GPU 스케줄링에 활용하는 데에 한계가 있다.

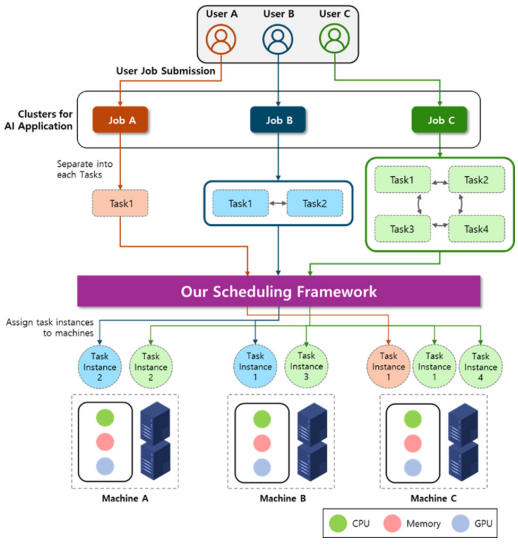

그림 1. 스케줄링 시스템의 기본적인 아키텍처
Fig. 1. The basic architecture of the scheduling system.

이와 같은 복잡성을 해결하기 위해 본 논문에서는 유전 알고리즘에 기반한 최적화 기술을 활용하는 새로운 태스크 스케줄링 기법을 제안한다. 제안하는 방식은 조각화로 인한 GPU 이용률 저하 문제를 해결하고 전체적인 GPU 효율성을 향상시키는 데에 초점을 맞추어 스케줄링을 최적화한다. 알리바바의 AI 플랫폼 클러스터에서 추출한 대규모 이벤트 트레이스를 재현하는 시뮬레이션 실험을 통해 제안하는 스케줄링 방식이 라운드 로빈 스케줄링에 비해 GPU 이용률을 12.8% 향상시키며, 또한 기존의 Bin-Packing 기반의 대표적인 스케줄링 방법인 Tetris[13]보다 우수한 성능을 나타냄을 확인하였다. 이러한 성능 개선은 클라우드 기반 AI 워크로드 실행 환경에서 자원 할당을 최적화하는 데에 크게 기여할 수 있을 것으로 기대된다.

II. 문제 정의

본 논문의 시스템 아키텍처는 그림 1에서 보는 것처럼 TensorFlow, PyTorch, Graph-Learn과 같은 다양한 프레임워크에서 개발된 AI 태스크들을 수용한다. 이러한 태스크들은 그 기능에 따라 여러 태스크 인스턴스로 구성되어 복수의 클러스터 GPU에서 실행될 수 있다. 본 논문의 스케줄링 시스템은 사용자가 제출한 작업에서 CPU, 메모리, GPU 요구 사항을 모니터링하고 해당 요

구를 충족할 수 있는 GPU와 태스크 인스턴스의 쌍을 탐색한다. 스케줄링 방법은 클러스터 내에서 GPU 이용률을 최적화하는 방향으로 결정한다.

스케줄링의 주요 목표는 각 태스크 인스턴스의 자원 요구 사항과 각 GPU의 가용 자원을 고려하여 가장 적합한 GPU에 태스크 인스턴스를 배치함으로써 GPU 이용률을 극대화하는 것이다. 이를 위해 스케줄링 알고리즘은 먼저 클러스터 내의 사용 가능한 GPU와 태스크 인스턴스 목록을 조합하여 모든 가능한 GPU, 태스크 인스턴스 쌍을 후보 스케줄링 군에 포함시킨다. 이때, 자원의 가용 여부에 따라 GPU 머신 m_j가 태스크 인스턴스 t_i를 수용할 수 있는 경우에만 해당 쌍을 후보 스케줄링에 포함시킨다. 머신 m_j가 태스크 인스턴스 t_i를 수용하려면, m_j의 가용 CPU, 메모리, GPU 자원들이 작업 인스턴스 t_i의 CPU, 메모리, GPU 요구량을 모두 충족해야 한다. 이때, GPU 이용률이 가장 높은 태스크 인스턴스와 머신의 쌍 $[m^*, t^*]$을 찾는 것이 최적화의 목표이다. 태스크 인스턴스 t^*를 머신 m^*에 할당함으로써 하나의 태스크 인스턴스에 대한 스케줄링이 완료되며, 이 과정은 클러스터 내에 제출된 모든 태스크 인스턴스에 대해 반복된다.

III. 스케줄링 최적화

유전 알고리즘(GA)은 집단 유전학의 개체 진화 과정을 모방하는 탐색 최적화 휴리스틱이다. 본 논문에서의 탐색 목표는 클러스터 내 각 머신에서 사용 가능한 GPU 자원 용량을 토대로 전체 클러스터의 GPU 이용률을 최적화하는 스케줄링 해를 찾는 것이다. 이러한 목표를 달성하기 위해, 유전 알고리즘은 특정 수의 해집합을 관리하며, 이를 지속적으로 진화시켜 최상의 해를 찾게 된다.

유전 알고리즘은 각 해들의 우수성을 평가하기 위해 적합도 함수를 사용한다. 적합도 함수는 머신 m과 태스크 인스턴스 t로 구성된 해의 품질을 평가한다. 태스크 인스턴스 t가 머신 m에 스케줄링 되기 위해서는 m의 가용 자원이 t의 필요 자원을 충족해야 한다. 이러한 조건이 만족되면, 적합도 함수는 태스크 인스턴스 t가 머신 m에 스케줄될 때 증가된 GPU 이용률을 적합도 값으로 반환한다. 반대로, 태스크 인스턴스 t가 머신 m에 수용이 불가능할 경우 -1을 적합도 값으로 반환한다.

$[m_j, t_i]$의 쌍에서 m_j는 클러스터 내 N대의 머신 중 하나의 식별자이고 t_i는 스케줄링을 대기하는 n개의 태스크 인스턴스 중 하나의 식별자를 나타낸다. 하나의 해는 태스크 인스턴스 t_i를 머신 m_j에 할당할지 여부를 결정한다. 클러스터 내 N대의 머신과 스케줄링을 기다리는 n개의 태스크 인스턴스를 조합하면 가능한 해의 수를 생성할 수 있다. 즉, 전체 해집합은 n개의 태스크 인스턴스와 N대의 머신에서 조합 가능한 모든 경우의 수, 즉 N^n가지로 구성 가능하다. 본 논문에서는 초기 해집합으로 100개의 해를 무작위로 생성한다.

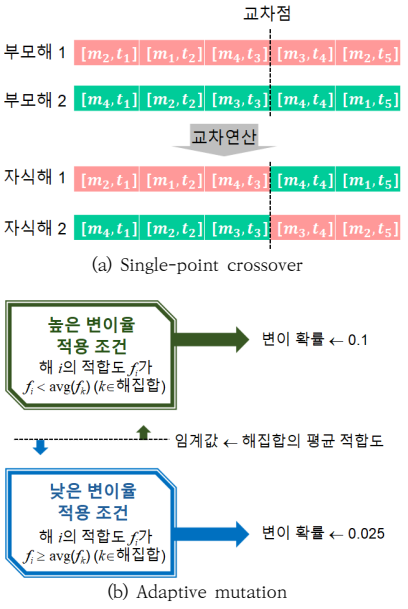

그림 2. 본 유전 알고리즘의 교차 연산과 변이 연산
Fig. 2. Crossover and mutation operations in our GA

다음 세대의 해를 생성하기 위해 해집합 내에서 부모해를 선택하는 과정을 선택 연산이라고 하며, 일반적으로 적합도 함수에 의해 선택 대상을 결정한다. 적합도 함수는 우수한 적합도 값을 가지는 해에 우선순위를 부여하며 부모해로 선택될 가능성을 높인다. 그러나, 선택 연산 시 지나치게 적합도 값에 의존할 경우 지역 최적해에 조기 수렴하는 문제가 발생할 수 있다. 이 문제를 완화하기 위해, 본 논문에서는 토너먼트 기반 선택 연산을 사용하여 해집합 내에서 20%의 해를 무작위로 샘플링한 후 이 샘플링된 해들 중 가장 높은 적합도 값을 가진 해를 부모해로 선택한다.

선택된 부모해에 대해서는 교차 연산을 수행하여 다음 세대의 해집합을 위한 자식해를 생성한다. 본 논문에서는 유전 알고리즘에서 널리 사용하는 1점 교차 연산을 사용하여 자식해를 생성한다. 1점 교차 연산에서는 부모해의 위치 중 임의의 지점을 교차점으로 설정한 후 자식해 생성시 교차점을 기반으로 두 부모해의 서로 다른 부

분을 계승하도록 한다. 문제 공간의 탐색 범위를 넓히기 위해 교차 연산으로 생성된 해의 특정 부분을 변경하는 변이 연산을 수행한다. 변이 연산은 해집합 내의 다양성을 유지하고 지역 최적해로의 미성숙 조기 수렴을 방지하는 역할을 한다. 본 논문에서는 해집합 전체의 평균 적합도 값을 초과하는 해의 경우 변이 확률 0.25를 적용하고, 그렇지 않은 해의 경우 변이 확률 0.1을 적용하는 적응형 변이 연산을 사용하였다.

교차 및 변이 연산 수행 후 생성된 자식해를 기존 해집합에서 대체하여 새로운 해집합을 구성한다. 본 논문에서는 전체 해집합의 40%가 새롭게 생성된 자식해들로 대체된다. 일반적으로 적합도 값이 낮은 해들은 세대를 반복하면서 새로 생성된 해들로 대체되면서 진화가 이루어지고 결국 전체 해집합이 최적화된 결과로 수렴하게 된다.

IV. 성능 평가

본 장에서는 제안된 스케줄링의 효과를 검증하기 위해 시뮬레이션 실험을 수행한다. 태스크 스케줄링의 성능을 평가하기 위해 프로세스 기반 이벤트 시뮬레이션 프레임워크인 Simpy에 기반하여 시뮬레이터를 구현했으며, 유전 알고리즘을 위한 오픈소스 파이썬 라이브러리인 PyGAD를 사용하였다. 본 논문에서는 알리바바 PAI 클러스터에서 수집된 두 달간의 AI 워크로드 트레이스를 스케줄링 입력 데이터로 사용하였다.

제안한 알고리즘과의 비교 대상으로는 최초 적합(First-Fit), 라운드 로빈(Round-Robin), 그리고 테트리스(Tetris) 알고리즘을 사용하였다. 그림 3은 시간이 흐름에 따라 하루 동안의 GPU 클러스터 이용률을 각 알고리즘 별로 보여주고 있다. 그림에서 x축은 해당 시점까지 처리된 작업 수를 나타내며, y축은 해당 작업 수를 스케줄링한 후 클러스터 내 GPU의 이용률을 보여준다.

그림에서 보는 것처럼 태스크 스케줄링에 널리 사용되는 라운드 로빈 알고리즘이 다양한 AI 워크로드가 공존하고 이질적인 GPU 머신이 존재하는 클러스터 상황에서 낮은 자원 이용률을 나타내는 것을 확인할 수 있다. 라운드 로빈 스케줄링은 태스크의 작업량과 GPU의 처리 용량을 고려하지 않고 큐에 있는 작업을 차례로 각 GPU에 할당하므로 일부 GPU는 과부하가 걸리는 반면 또다른 GPU는 낮은 이용률을 초래하게 된다.

테트리스와 최초 적합 알고리즘은 시간이 지남에 따라

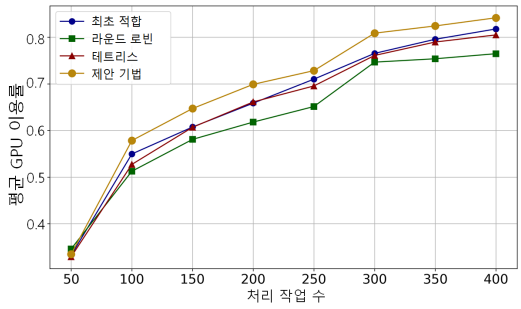

그림 3. 각 알고리즘 별 GPU 이용률의 비교
Fig. 3. Comparison of GPU Utilization for each algorithm.

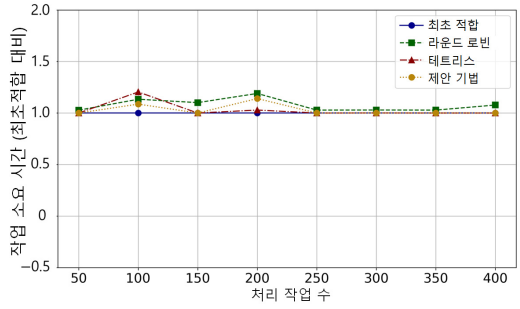

그림 4. 각 알고리즘별 작업 완료시간의 비교
Fig. 4. Comparison of completion time for each algorithm.

조금씩 차이를 나타내지만 전반적으로 유사한 GPU 이용률을 나타내는 것을 확인할 수 있었다. 제안한 알고리즘은 클러스터 전반에 걸쳐 GPU 이용률 측면에서 일관되게 최상의 결과를 나타내었다. 이는 GPU 이용률 측면에서 태스크 인스턴스와 머신의 최적화된 쌍을 선택하여 클러스터 내 GPU 자원의 조각화를 최소화하여 얻게 된 결과로 볼 수 있다. 제안한 알고리즘의 GPU 이용률은 라운드 로빈, 최초 적합, 테트리스 알고리즘 대비 12.8%, 6.5%, 6.6% 우수한 결과를 나타내었다.

한편, GPU 이용률이 개선되더라도 그 부작용으로 사용자 관점의 성능 저하가 발생할 수 있다. 이를 확인하기 위해, 각 알고리즘을 사용한 경우의 작업 완료 시간을 측정하고 그 결과를 비교하였다. 그림 4는 제안한 알고리즘과 비교 알고리즘들 간에 스케줄링된 작업들의 완료 시간을 나타낸다. 각 작업의 완료 시간은 작업의 시작 시간부터 끝나는 시간까지의 시간 차로 계산하였다. 그림에서 보는 것처럼 제안한 알고리즘이 다른 세 알고리즘들과 비슷한 작업 완료 시간을 나타내었으며, 이는 제안한 알고리즘이 성능 상의 오버헤드 없이 GPU 이용률을 높이는 스케줄링 결과를 얻을 수 있음을 시사한다.

V. 결 론

현대의 AI 워크로드용 시스템은 MLaaS 클러스터 내의 다양한 GPU 자원을 큐에 쌓인 태스크들에 효율적으로 할당하기 위에 GPU 공유 기술을 활용한다. 그러나, 기존의 태스크 스케줄링 기법들은 자원의 조각화를 유발하여, GPU 이용률을 크게 저하시킨다. 본 논문에서는 이기종 GPU를 갖춘 대규모 클러스터 내에서 GPU 이용률을 최적화하기 위한 스케줄링 기법을 제안하였다. 제안한 기법은 유전 알고리즘에 기반한 최적화 기술을 토대로 GPU 이용률을 극대화시켰으며, 광범위한 워크로드 트레이스를 이용한 실험을 통해 전통적인 스케줄링 기법에 비해 GPU 이용률을 개선할 수 있음을 확인하였다.

References

[1] Q. Weng, W. Xiao, Y. Yu, W. Wang, C. Wang, J. He, Y. Li, L. Zhang, W. Lin, and Y. Ding, "MLaaS in the Wild: Workload Analysis and Scheduling in Large-Scale Heterogeneous GPU Clusters," Proc. 19th USENIX NSDI Conf., pp. 945-960, 2022.

[2] K. Lee, H. Jung, and S. Lee, "Anomaly detection method of user trajectories based on deep learning technologies," JKIIT, vol. 20, no. 11, pp. 101-116, 2022.
DOI: https://doi.org/10.14801/jkiit.2022.20.11.101

[3] S. Kim, W. Hur, and J. Ahn, "A progressive web application for mobile crop disease diagnostics based on transfer learning," Journal of the Korea Academia-Industrial cooperation Society (JKAIS), vol. 23, no. 2 pp. 22-29, 2022.
DOI: https://doi.org/10.5762/KAIS.2022.23.2.22

[4] D. Kim, S. Lee, and H. Bahn, "An Adaptive Location Detection Scheme for Energy-Efficiency of Smartphones," Pervasive and Mobile Computing, vol. 31, pp. 67-78, 2016.
DOI: https://doi.org/10.1016/j.pmcj.2016.04.012

[5] J. Mohan, A. Phanishayee, J. Kulkarni, and V. Chidambaram, "Looking Beyond GPUs for DNN Scheduling on Multi-Tenant Clusters," Proc. 16th USENIX OSDI Conf., pp. 579-596, 2022.

[6] S. Yoo, Y. Jo, and H. Bahn, "Integrated Scheduling of Real-time and Interactive Tasks for Configurable Industrial Systems," IEEE Trans. Industrial Informatics, vol. 18, pp. 631-641, 2022.
DOI: https://doi.org/10.1109/TII.2021.3067714

[7] S. Ki, G. Byun, K. Cho, and H. Bahn, "Co-Optimizing CPU Voltage, Memory Placement, and Task Offloading for Energy-Efficient Mobile Systems," IEEE Internet of Things Journal, vol. 10, pp. 9177-9192, 2023.
DOI: https://doi.org/10.1109/JIOT.2022.3233830

[8] S. Kwon and H. Bahn, "Memory Reference Analysis and Implications for Executing AI Workloads in Mobile Systems," Proc. IEEE IEIT Conf., pp. 281-285, 2023.
DOI: https://doi.org/10.1109/IEIT59852.2023.10335577

[9] S. Park, and H. Bahn, "Trace-based Performance Analysis for Deep Learning in Edge Container Environments," Proc. 8th IEEE FMEC Conf., pp. 87-92, 2023.
DOI: https://doi.org/10.1109/FMEC59375.2023.10306027

[10] J. Li, H. Xu, Y. Zhu, Z. Liu, C. Guo, and C. Wang "Lyra: Elastic Scheduling for Deep Learning Clusters," Proc. 18th EuroSys Conf., pp. 835-850, 2023.
DOI: https://doi.org/10.1145/3552326.3587445

[11] NVIDIA Multi-Instance GPU. Available online: https://www.nvidia.com/en-us/technologies/multi-instance-gpu/.

[12] W. Xiao, R. Bhardwaj, R. Ramjee, M. Sivathanu, N. Kwatra, Z. Han, P. Patel, X. Peng, H. Zhao, Q. Zhang, F. Yang, and L. Zhou, "Gandiva: Introspective Cluster Scheduling for Deep Learning," Proc. 13th USENIX OSDI Conf. pp. 595-610, 2018.

[13] R. Grandl, G. Ananthanarayanan, S. Kandula, S. Rao, and A. Akella, "Multi-resource Packing for Cluster Schedulers," ACM SIGCOMM Computer Communication Review, vol. 44, pp. 455-466, 2014.
DOI: https://doi.org/10.1145/2619239.2626334

저 자 소 개

권 석 민(정회원)

- 2018년 2월 : 이화여자대학교 컴퓨터공학과 학사
- 2021년 2월 : 서울대학교 컴퓨터공학부 석사
- 2022년 3월 ~ : 이화여자대학교 컴퓨터공학과 박사과정
- 주관심분야 : 운영체제, 인공지능시스템, 스토리지 시스템

반 효 경(정회원)

- 1997년 2월 : 서울대학교 계산통계학과 학사
- 1999년 2월 : 서울대학교 전산과학과 석사
- 2002년 2월 : 서울대학교 컴퓨터공학부 박사.
- 2002년 9월 ~ : 이화여자대학교 컴퓨터공학과 교수.
- 주관심분야 : 운영체제, 스토리지시스템, 임베디드시스템