쿠버네티스 클러스터에서의 머신러닝 워크로드의 메모리 부족 오류 자기 치유를 위한 오퍼레이터 설계

황태욱, 김영한

숭실대학교

taeuk.h@dcn.ssu.ac.kr, younghak@ssu.ac.kr

Design of an Operator for Out-of-Memory Self-Healing of Machine Learning Workloads in a Kubernetes Cluster

Hwang Tae Uk, Kim Young Han

Soongsil Univ.

요 약

GPU를 사용하는 머신러닝 워크로드가 Kubernetes 환경에서 실행될 때, GPU 메모리 부족 오류가 발생하면 단순 재시작만으로는 문제를 해결할 수 없다. 이는 고정된 GPU 리소스 요청으로 인해 동일한 조건의 재시작이 반복되기 때문이다. 본 논문에서는 Fluentd 기반 로그 수집과 Custom Controller를 활용하여, GPU 메모리 부족 오류 발생 시 GPU 자원을 동적으로 조정하고 Pod를 자가 치유(Self-Healing)할 수 있는 MLAutoHealer 시스템을 제안한다. 제안 시스템은 Kubernetes-native 방식으로 설계되었으며, 선언형 정책 리소스를 기반으로 GPU 재할당 및 리소스 최적화를 자동화한다.

Ⅰ. 서론

Kubernetes는 다양한 워크로드의 자동 배포, 확장, 자가 치유 등을 지원하는 클라우드 네이티브 플랫폼이다. 최근 머신러닝(ML, Machine Learning) 모델의 대형화와 GPU 수요 증가에 따라, Kubernetes 환경에서 GPU 기반 ML 워크로드의 안정적인 실행이 중요해지고 있다[1]. 그러나 실행 중 GPU 메모리 부족(OOM, Out-of-Memory) 오류는 ML 작업의 중단을 초래하며, Kubernetes의 기본 재시작 정책만으로는 이러한 문제를 근본적으로 해결할 수 없다[2].

이에 본 논문에서는, OOM 오류 발생 시 GPU 리소스 요구량을 동적으로 조정하여 자가 치유(Self-Healing) 할 수 있도록 하는 MLAutoHealer 시스템을 제안한다.

Ⅱ. 관련 연구

Kubernetes 기반 리소스 자동화 연구는 오퍼레이터 기반 구성 자동화, HPA/VPA 기반 확장 정책, KEDA 등 이벤트 기반 스케일러 중심으로 이루어져 왔다[3]. 이에 대한 발전으로 최근에는 강화학습 기반 오토스케일러(AWARE[4])를 활용하거나, 이벤트 기반 워크플로우 확장(HyperFlow WMS[5])처럼, 동적 상황에 맞추어 리소스를 실시간 조정하는 방향으로 발전하고 있다.

또한 NVIDIA GPU Operator, DRA(Dynamic Resource Allocation[6]) 등은 리소스 관리에 기여하고 있지만, ML 워크로드에 특화된 OOM 대응 구조는 부재하다. 최근 연구에서는 GPU 메모리 요구량을 사전에 예측하여 스케줄링에 반영하는 VeritasEst[7]와, Kubernetes 환경에서 GPU 메모리 초과 할당을 허용하는 nvshare[8]가 제안되었으나, 여전히 OOM 발생 시 동적으로 GPU 요청량을 조정하여 자가 치유하는 시스템은 드물다.

Ⅲ. MLAutoHealer 시스템 설계

Ⅲ-Ⅰ. 시스템 구성

MLAutoHealer는 다음과 같은 주요 구성 요소로 이루어진다:

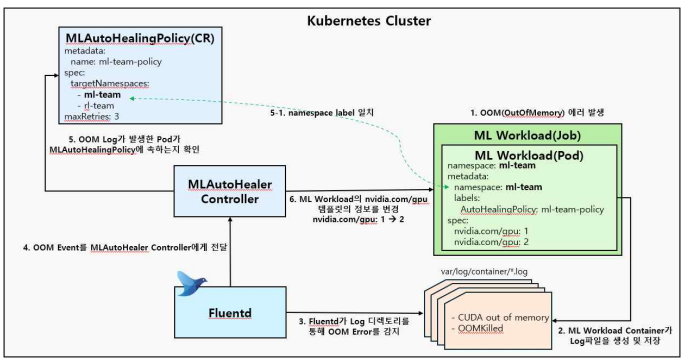

그림 1. MLAutoHealer 시스템 구성도

1. Fluentd: Pod 로그를 실시간 수집하여 OOM 메시지를 탐지

2. MLAutoHealer Controller: OOM 이벤트 수신 후 리소스 조정 로직 수행

3. MLAutoHealingPolicy CRD: 정책 선언형 리소스로 대상 Pod 범위와 대응 방식을 정의

Ⅲ-Ⅱ. 정책 기반 자가 치유 흐름

그림 1과 같이 MLAutoHealer 시스템은 GPU 기반의 머신러닝 워크로드 실행 중 발생하는 OOM 오류에 대해 자동으로 대응할 수 있도록 설계되어 있다. 워크로드가 실행되는 동안 GPU 메모리가 부족하여 OOM(CUDA out of memory) 오류가 발생하면, 해당 오류 메시지는 컨테이너의 표준 출력에 로그로 기록된다. Fluentd는 클러스터 내 모든 노드에 DaemonSet 형태로 배포되어, 각 컨테이너의 로그 파일을 지속적으로 수집하고 분석한다. 이 과정에서 Fluentd는 OOM 패턴을 감지하고, 사전에 구성된 HTTP 출력 플러그인을 통해 MLAutoHealer Controller에 이벤트를 전송한다.

이벤트를 수신한 MLAutoHealer Controller는 먼저 해당 Pod의 이름과

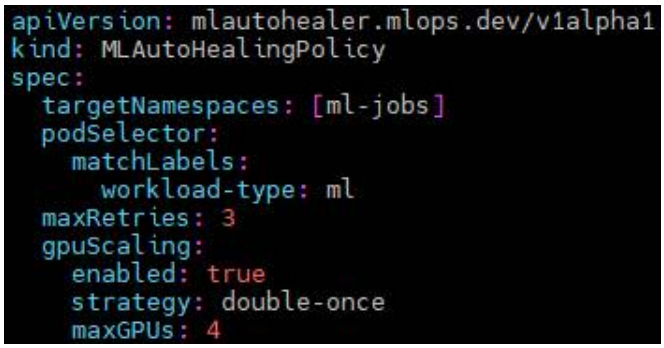

그림 2. MLAutoHealingPolicy CR

네임스페이스를 바탕으로 Kubernetes API를 통해 Pod 객체를 조회하고, 이로부터 ownerReference 정보를 추출한다. 이를 통해 해당 Pod를 생성한 상위 리소스(예: Deployment, Job 등)를 식별하고, 정책 적용 여부를 판단하기 위해 클러스터 내 정의된 MLAutoHealingPolicy 리소스를 탐색한다. 정책에는 적용 대상 네임스페이스, 라벨 셀렉터, 최대 재시도 횟수, GPU 확장 전략 등이 정의되어 있으며, 조건이 만족되면 지정된 GPU 스케일링 전략에 따라 리소스 조정 작업이 수행된다.

MLAutoHealer는 상위 리소스의 spec.template을 patch하여 GPU 요청량(nvidia.com/gpu)을 증가시키고, 필요 시 Deployment 혹은 Job을 재시작하여 새로운 리소스 조건이 반영된 Pod가 생성되도록 유도한다. 이 과정을 통해 동일 조건에서 반복적으로 발생하던 OOM 문제가 해결되며, 사용자의 개입 없이 자가 치유가 이루어지는 구조를 완성하게 된다.

Ⅲ-Ⅲ. 정책 리소스: MLAutoHealingPolicy

MLAutoHealer 시스템은 GPU OOM 상황에 대한 자가 치유 기능을 정책적으로 제어할 수 있도록, 커스텀 리소스인 MLAutoHealingPolicy를 정의한다. 이 리소스는 클러스터 내 특정 워크로드에 대해 어떤 조건에서 어떤 방식으로 자가 치유 조치를 수행할지를 선언형으로 기술할 수 있도록 설계되었다.

그림 2와 같이 정책 리소스의 spec 필드는 여러 설정 항목으로 구성된다. 먼저 targetNamespaces 필드를 통해, 자가 치유 정책이 적용될 네임스페이스를 명시할 수 있다. 예를 들어, "ml-jobs"라는 네임스페이스를 지정하면, 해당 네임스페이스 내에서 실행 중인 모든 Pod가 이 정책의 대상이 된다.

또한, podSelector 항목은 라벨 기반 필터링 조건을 정의한다. 여기서 workload-type: ml이라는 라벨을 가진 Pod만 정책의 적용 대상이 되도록 제한함으로써, 동일한 네임스페이스 내에서도 특정 유형의 워크로드만 선별적으로 제어할 수 있다. 이와 같은 필터링 기능은 정책의 적용 대상을 세밀하게 조정할 수 있게 하며, 다양한 팀과 서비스가 혼재하는 멀티테넌시 환경에서도 유용하게 활용될 수 있다.

maxRetries 필드는 OOM 오류 발생 후 자가 치유 시도 횟수의 상한선을 정의한다. 이를 통해 무한 반복되는 조정 시도로 인한 자원 낭비와 불안정성을 방지할 수 있다. 예를 들어 3으로 설정된 경우, MLAutoHealer는 동일한 Pod 또는 상위 리소스에 대해 최대 세 번까지 GPU 리소스 조정 시도를 수행한다.

GPU 조정 전략은 gpuScaling 필드를 통해 제어된다. enabled: true로 설정되면 GPU 조정이 활성화되며, strategy 항목에는 조정 방식이 정의된다. 예를 들어 "double-once"는 현재 할당된 GPU 수를 한 번에 두 배로 늘리는 전략을 의미한다. 이외에도 "increment" 방식처럼 점진적으로 조정하는 전략도 향후 확장이 가능하다. maxGPUs는 조정 가능한 GPU 수의 최대치를 설정하여, 무분별한 리소스 할당을 방지한다.

이러한 정책 리소스는 MLAutoHealer Controller가 동작하는 데 있어 기준점이 되는 선언형 매개체로 기능하며, 관리자가 복잡한 로직을 코드로 작성하지 않아도 고수준의 복구 정책을 Kubernetes-native 방식으로 정의할 수 있도록 한다. 나아가 이 구조는 GPU 외에도 Memory, CPU, 혹은 DRA 기반 리소스에까지 확장 가능하여, 일반화된 ML 워크로드 자가 치유 프레임워크로 발전할 수 있는 기반이 된다.

Ⅳ. 결론

본 논문에서는 Kubernetes 환경에서 GPU 기반 ML 워크로드의 OOM 오류에 대응하여 자가 치유를 가능하게 하는 MLAutoHealer 시스템을 제안하였다. 제안 시스템은 로그 기반 이벤트 탐지와 정책 기반 리소스 조정을 결합하여 반복적인 OOM 문제를 자동으로 해결할 수 있다. 앞으로는 다양한 ML 워크로드 유형에 대한 적용 및 실험을 통해 성능과 안정성을 더욱 검증할 예정이다.

ACKNOWLEDGMENT

이 논문은 정부(과학기술정보통신부)의 재원으로
정보통신기획평가원-대학ICT연구센터(ITRC)의 지원을 받아 수행된
연구임(IITP-2025-RS-2023-00258649)

참 고 문 헌

[1] Madabushini, P., "Leveraging Kubernetes for AI/ML Workloads: Case Studies in Autonomous Driving and Large Language Model Infrastructure," World Journal of Advanced Engineering Technology and Sciences, vol. 15, no. 1, pp. 1044–1052, Apr. 2025.

[2] Ray, J., "Taming the Memory Beast: Strategies for Reliable ML Training on Kubernetes," arXiv preprint arXiv:2412.14701, Dec. 2024, (http://arxiv.org/abs/2412.14701).

[3] Molleti, R., "Kubernetes Advanced Auto Scaling Techniques," Journal of Mathematical & Computer Applications, vol. 1, pp. 1–4, 2022.

[4] Qiu, H., Mao, W., Wang, C., Franke, H., Youssef, A., Kalbarczyk, Z. T., Basar, T., and Iyer, R. K., "AWARE: Automate Workload Autoscaling with Reinforcement Learning in Production Cloud Systems," Proc. USENIX Annual Technical Conference, pp. 385–398, 2023.

[5] Orzechowski, M., Balis, B., and Janecki, K., "Towards Cloud-Native Scientific Workflow Management," arXiv preprint arXiv:2408.15445, Aug. 2024, (http://arxiv.org/abs/2408.15445).

[6] Malvankar, A., and Tardieu, O., "Unleashing the Power of DRA (Dynamic Resource Allocation) for Just-in-Time GPU Slicing," Presented at KubeCon EU 2024.

[7] Shi, J., and Elkhatib, Y., "Accurate GPU Memory Prediction for Deep Learning Jobs through Dynamic Analysis," arXiv preprint arXiv:2504.03887, Apr. 2025, (http://arxiv.org/abs/2504.03887).

[8] Alexopoulos, G., and Mitropoulos, D., "nvshare: Practical GPU Sharing Without Memory Size Constraints," Proc. IEEE/ACM 46th Int. Conf. Software Engineering: Companion Proceedings (ICSE-Companion), pp. 1–4, Apr. 2024.