제20장. AI 지원 성능 최적화 및 수백만 GPU 클러스터를 향한 확장

이 장에서는 인간과 AI가 협력하여 AI 시스템 성능을 최적화하는 방식을 보여주는 다양한 사례 연구와 미래 동향을 종합한다. 특히 AI는 수동 작업으로 생성된 커널보다 더 빠르게 실행되는 커널을 만들기 위해 저수준 GPU 코드의 미세 조정을 지원할 수 있다.

더 넓은 맥락에서, 이러한 사례들은 행렬 곱셈과 같은 핵심 연산에서도 알고리즘 혁신이 새로운 하드웨어 도입과 유사한 성능 향상을 가져올 수 있음을 보여줍니다. 높은 수준에서, 일련의 강화 학습 롤아웃(예: 반복자)으로부터 보상 피드백을 사용하는 워크플로를 고려해 보십시오. 이는 그림 20-1에 표시된 바와 같이 환경에 가장 적합한 GPU 커널 코드를 찾는 데 도움이 될 수 있습니다.

이러한 AI 지원 접근법은 성능 향상, 훈련 시간 단축, 운영 비용 절감에 기여할 수 있습니다. 또한 더 작은 시스템에 더 큰 모델을 효율적으로 배포할 수 있게 하여 AI의 미래 발전을 가능하게 합니다. 즉, 이는 더 나은 AI를 만들기 위해 AI가 도움을 주는 것입니다. 우리는 이를 환영합니다!

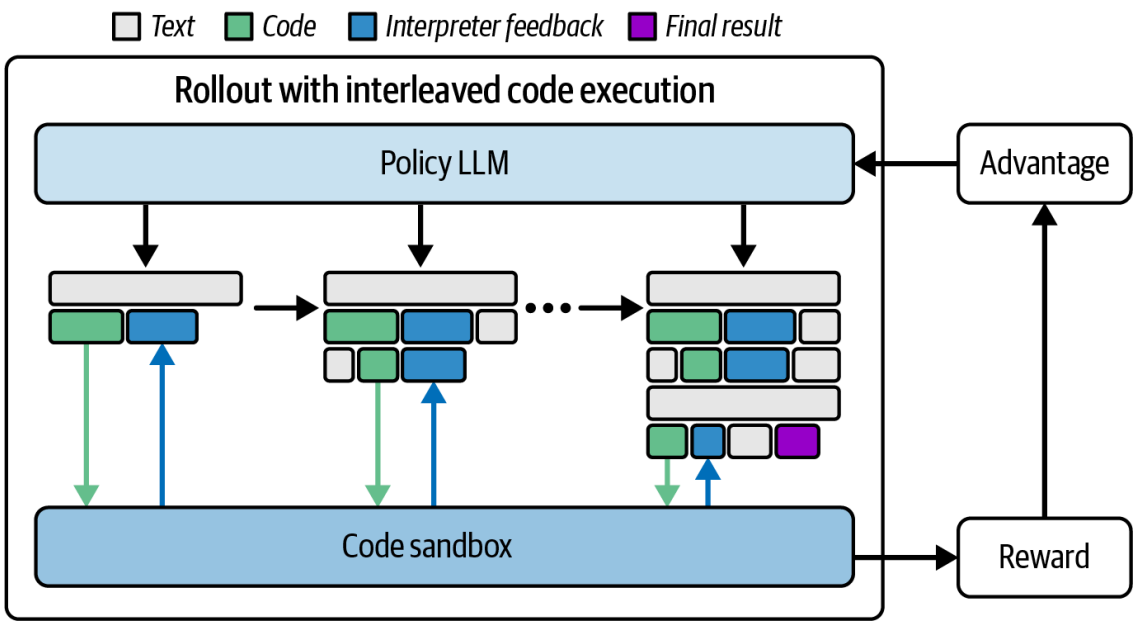

그림 20-1. 환경에 가장 적합한 GPU 커널 코드 찾기 위한 강화 학습활용

AlphaTensor AI-Discovered Algorithms Boosting GPU Performance (Google DeepMind)

모든 AI 최적화가 코드 수준에서 이루어지는 것은 아닙니다. 때로는 최적화가 알고리즘과 수학의 영역으로 더 깊이 들어갑니다. 획기적인 사례는 2022년 DeepMind의 AlphaTensor 프로젝트에서 나왔는데, 여기서 AI를 활용해 새로운 일반 행렬 곱셈(GEMM) 기법을 발견했습니다.

GEMM은 거의 모든 모델 훈련 및 추론 작업 부하를 뒷받침하는 핵심 연산입니다. GEMM 효율성의 미세한 개선조차도 AI 분야 전체에 막대한 영향을 미칠 수 있습니다. AlphaTensor는 강화 학습을 활용하여 다양한 가능성을 탐색하는 단일 플레이어 게임으로 빠른 알고리즘 탐색을 체계화했습니다.

놀라운 결과는 당시 존재하던 어떤 인간이 고안한 방법보다 우수한 행렬 곱셈 공식을 찾아냈다는 점입니다. 예를 들어 그림 20-2에 표시된 것처럼 2×2 행렬에 대한 스트라센의 유명한 2차 미만 알고리즘을 재발견했을 뿐만 아니라 더 큰 행렬 크기에서도 이를 개선했습니다.

그러나 진정한 검증은 실제 하드웨어에서 해당 알고리즘을 테스트했을 때 이루어졌습니다. AlphaTensor는 NVIDIA Volta V100 GPU 세대에 특화된 방법을 발견했는데, 이는 당시 표준 NVIDIA V100 시대 cuBLAS 라이브러리보다 대형 행렬을 10~20% 더 빠르게 곱했습니다. GEMM 성능에서 10~20%의 가속은 엄청난 성과입니다. 이는 모든 모델의 전파 및 역전파 과정에서 추가로 10~20%의 무료 연산 능력을 얻는 것과 같습니다.

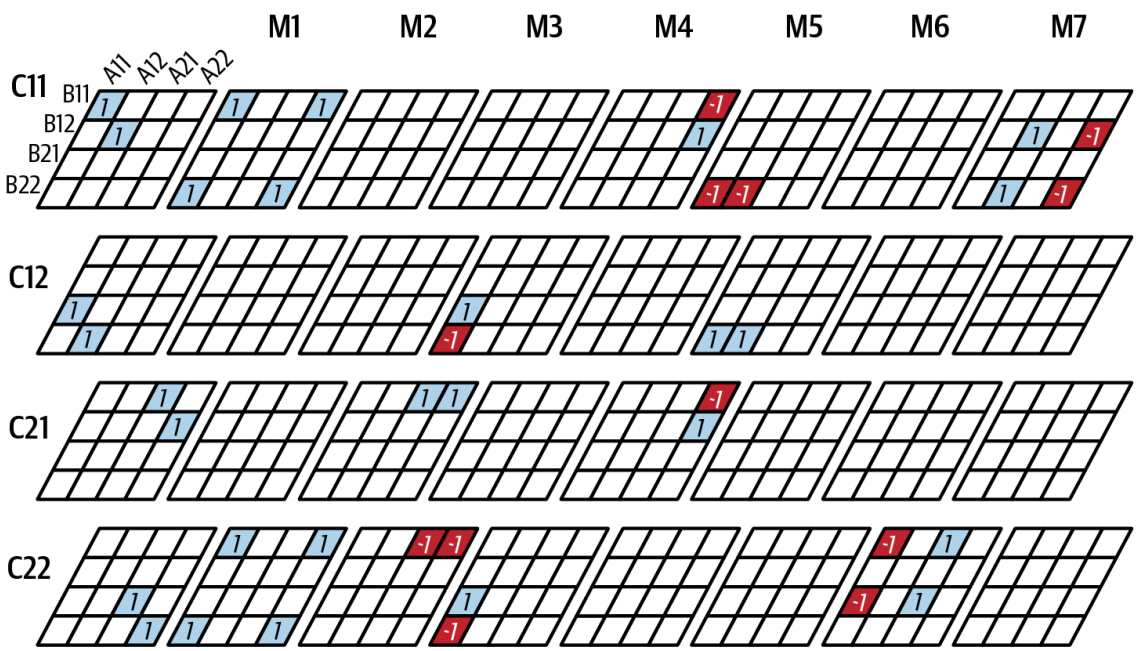

그림 20-2. 2×2 행렬 곱셈을 위한스트라센의 2차 미만 알고리즘 (출처: https://oreil.ly/5jzLn)

이러한 성능 향상은 일반적으로 새로운 하드웨어 세대나 수개월에 걸친 저수준 CUDA 튜닝을 통해 달성됩니다. 그러나 이번 사례에서는 AI가 비교적 짧은 시간 내에 수학적으로 더 나은 방법을 찾아냈습니다.

이 사례가 주는 교훈은 인간 엔지니어가 혁신적이라 여기는 기본적인 알고리즘 및 수학적 연산 속에도 아직 발견되지 않은 효율성이 남아 있을 수 있다는 점이다. AI는 인간이 합리적인 시간 내에 시도할 수 없는 수천, 수백만 가지의 알고리즘 변형을 검토할 수 있다. 성능 엔지니어들에게 AlphaTensor의 성공은 알고리즘 혁신이 끝나지 않았음을 시사한다. 미래에는 AI가 컨볼루션, 정렬, 어텐션 같은 기본 연산을 위한 더 빠른 알고리즘 도구 키트를 제공할 수도 있다.

이 경우의 투자 수익률(ROI)은 다소 간접적이지만 매우 영향력이 큽니다. AlphaTensor의 행렬 곱셈 알고리즘을 GPU 라이브러리에 통합함으로써, 대규모 훈련 작업이나 추론 워크로드의 속도가 즉각적으로 향상될 것입니다. 이는 그래픽 렌더링부터 LLM 성능, 과학적 계산에 이르기까지 모든 분야에 영향을 미칠 수 있습니다. AlphaTensor는 수백 개의 GPU에서 수천 번의 훈련 반복을 거친 15%의 속도 개선이 막대한 시간과 에너지 절약으로 이어진다는 점을 입증했습니다. 이는 코드를 실행할 때마다 지속적으로 반환되는 수익입니다. 더욱이 이 속도 향상은 추가 하드웨어 없이 오직 더 스마트한 소프트웨어만으로 달성되었습니다.

초대규모 성능 엔지니어에게 주는 교훈은 스택의 모든 수준에서 AI 기반 최적화에 열린 태도를 유지하라는 것입니다. GEMM과 같이 가장 기본적이고 잘 최적화된 연산조차도 개선의 여지가 있을 수 있습니다. 인간의 편견 없이 AI가 최적화 영역을 탐색하도록 허용하면 전반적인 실행 시간을 대폭 단축함으로써 높은 성과를 거둘 수 있습니다.

본문 작성 시점 기준, AlphaTensor의 행렬 곱셈 알고리즘은 여전히 실험 단계입니다. cuBLAS와 같은 주류 GPU 라이브러리는 추가 검증과 일반화가 완료될 때까지 이러한 기법을 아직 통합하지 않았습니다.

DeepSeek-R1(NVIDIA)을 통한 자동화된 GPU 커널 최적화

저수준 GPU 코드 최적화는 오랫동안 *CUDA* 전문가(*CUDA Ninjas)*만이 할 수 있는 기술 이었으나, AI가 이러한 전문 작업을 수행할 수 있음이 입증되었습니다. NVIDIA 엔지니어들은 강력한 DeepSeek-R1 추론 모델을 실험하여 복잡한 어텐션 메커니즘을 위한 고성능 CUDA 커널을 생성할 수 있는지, 그리고 이 커널이 수작업으로 튜닝된 고성능 구현체와 경쟁할 수 있는지 확인했습니다.

추론 모델인 DeepSeek-R1은 응답을 생성하기 전에 모델을 빠르게 한 번 통과하는 대신, 일정 시간에 걸쳐 출력을 정교화하는 '추론 시간' 확장 전략을 사용합니다. 주어진 시간이 길수록 결과는 더 좋아집니다. DeepSeek-R1과 같은 추론 모델은 인간이 답변을 내뱉기 전에 시간을 들여 생각하듯, 답변에 대해 더 오래 사고 답변을 반복하도록 미세 조정됩니다.

이 실험에서 NVIDIA는 H100에 R1을 배포하고 최적화된 어텐션 커널 코드를 생성하는 데 15분을 부여했습니다. 생성기 루프에 *검증* 프로그램을 삽입하여 R1이 커널을 제안할 때마다 검증기가 생성된 커널 코드의 정확성을 확인하고 코드 효율성을 측정하도록 했습니다. 생성 → 검증 → 피드백 → 반복 루프는 다음과 같은 의사 코드로 표현됩니다:

```
for iteration in range(max_iters):
    code = R1_model.generate_code(prompt)
    valid, runtime = verifier.verify(code)
    if valid and runtime < target_time:
        break  # Accept this kernel
    prompt = refine_prompt(prompt, verifier.feedback)
    ...
```

이 피드백 루프는 다음 커널 코드 반복에 사용할 개선된 prompt에 대한 지침을 제공합니다. 루프는 그림 20-3과 같이 코드가 주어진 기준을 충족할 때까지 계속됩니다.

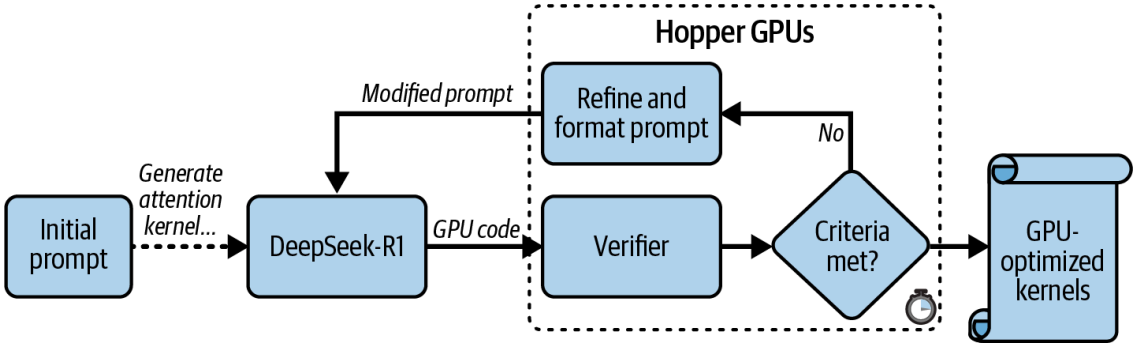

그림 20-3. NVIDIA Hopper 플랫폼에서 DeepSeek-R1을 활용한추론 시간 확장성 (출처: DeepSeek-R1을 통한 GPU 커널 생성 자동화 및 추론 시간 확장성 | NVIDIA 기술 블로그)

사용된 prompt는 다음과 같습니다:

> *상대 위치 인코딩을 지원하기 위한 GPU 어텐션 커널을 작성하십시오. 커널 내에서 실시간으로 상대 위치 인코딩을 구현하십시오. 필요한 수정 사항을 포함한 완성된 코드를 반환해야 합니다.*
>
> *다음 함수를 사용하여 상대적 위치 인코딩을 계산하십시오:*
>
> *def relative_positional(score, b, h, q_idx, kv_idx):*
>
> *반환한 값 + (q_idx - kv_idx)*
>
> *커널 구현 시 qk_scale = sm_scale * 1.44269504에 따라 상대적 위치 인코딩에 상수 스케일링 계수 1.44269504를 적용해야 합니다. PyTorch 참조 구현에서는 상대적 위치 인코딩을 스케일링할 필요가 없지만, GPU 커널에서는 다음을 사용하십시오:*
>
> *qk = qk * qk_scale + rel_pos * 1.44269504*
>
> *이러한 변경 사항을 반영한 완전한 업데이트된 커널 코드를 제공해 주십시오. 커널 연산 내에서 상대적 위치 인코딩이 효율적으로 적용되도록 보장해야 합니다.*

이 prompt를 통해 AI는 어텐션용 기능적으로 올바른 CUDA 커널을 생성했습니다. (참고: 1.44269504 = 1/ln(2)입니다. 이 값을 사용하여 prompt는 qk를 형성할 때 상대 위치 항을 그에 맞게 조정합니다. 정확성 외에도 생성된 커널은 내장 PyTorch FlexAttention API 대비 1.1~2.1배의 속도 향상을 달성했습니다. 그림 20-4는 인과적 마스크와 장문서 마스크를 포함한 다양한 어텐션 패턴에서 생성된 커널과 PyTorch의 최적화된 FlexAttention 간 성능 비교를 보여줍니다.

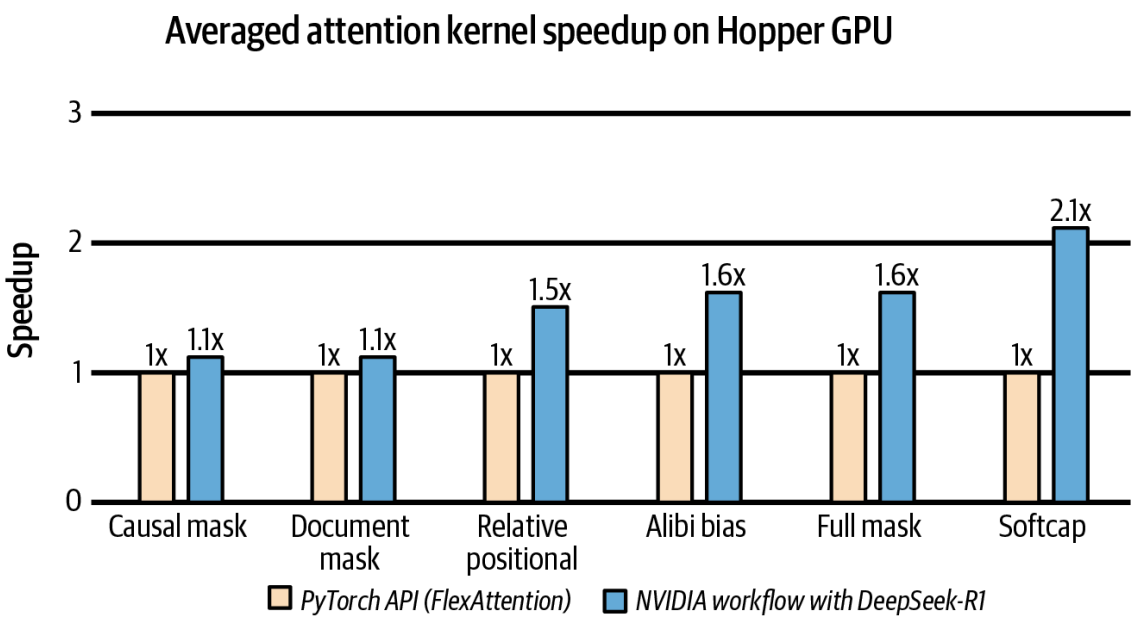

그림 20-4. 자동 생성된 어텐션 커널은 PyTorch FlexAttention 대비 1.1배~2.1배의 속도 향상을 달성함 (출처: DeepSeek-R1을 활용한 GPU 커널 생성 자동화 및 추론 시간 확장 | NVIDIA 기술 블로그)

더욱 인상적인 점은, AI가 생성한 커널이 스탠퍼드 대학의 KernelBench 테스트 스위트(어텐션 작업)를 사용했을 때 기본 테스트 케이스(레벨-1)의 100%와 복잡한 케이스(레벨-2)의 96%에서 검증 가능한 정확도를 보였다는 것입니다. 이는 사실상 인간 엔지니어의 신뢰성과 맞먹는 수준입니다.

실제 적용 시에는 KernelBench에서 수행한 것처럼 이러한 검증 시스템을 강력한 테스트 스위트와 통합해야 합니다. 이를 통해 드문 경계 사례가 생성된 코드에 오류를 유발하지 않도록 해야 합니다.

이로부터 얻은 교훈은 LLM에 출력을 검증하고 비판하며 개선할 적절한 도구를 제공하면 코드 품질을 향상시킬 수 있다는 점입니다. 직관적으로 이 워크플로는 인간 엔지니어가 자신의 코드를 반복적으로 자질하고 디버깅하며 개선하는 방식과 동일합니다. 생성 → 검증 → 개선 루프를 통해 15분 만에 초안 수준의 코드가 생산 환경에 적합한 어텐션으로 진화했습니다. 이는 AI 지원 성능 튜닝의 강력한 패러다임을 보여줍니다.

ROI는 게임 체인저 수준입니다. NVIDIA의 최고 CUDA 엔지니어조차도 새로운 유형의 어텐션 커널 변형을 수작업으로 제작하고 테스트하는 데 몇 시간 또는 며칠이 소요될 수 있습니다. 이 AI 지원 최적화 접근법을 사용하면 AI가 비교적 효율적인 저수준 CUDA 커널을 훨씬 짧은 시간에 생성할 수 있습니다. 이를 통해 엔지니어들은 AI가 탐지하고 수정하기 어려울 수 있는 고급 AI 시스템 최적화 기회와 에지 케이스에 집중할 수 있습니다.

일부 인적 감독이 여전히 필요했지만, 이 실험은 상당한 런타임 성능 향상을 동반한 GPU 최적화 소프트웨어 개발 비용을 절감할 수 있는 실행 가능한 경로를 보여

주었습니다. AI 시스템 성능 엔지니어에게 이러한 유형의 AI 지원은 향후 워크플로우가 AI 코파일럿과 협력하여 하드웨어, 소프트웨어, 알고리즘 전반에 걸친 최적화를 신속하게 공동 설계하는 방식을 포함할 수 있음을 시사합니다. AI 코파일럿은 인간 생산성의 증폭제 역할을 합니다. 이러한 코파일럿을 기존 코드베이스에서 파생된 방대한 CUDA 팁과 트릭에 대한 지식을 활용하여 복잡한 문제를 추론할 수 있는 사전 훈련 및 미세 조정된 AI 인턴으로 생각하십시오.

최적화된 GPU 커널 생성을 위한 강화 학습 접근법 (Predibase)

또 다른 스타트업인 프레디베이스(Predibase)는 강화 학습을 활용한 약간 다른 접근법으로 의 자동화된 GPU 프로그래밍을 시연했습니다. 그들은 더욱 대담한 질문을 던졌습니다: 수많은 PyTorch 및 Triton 코드 예시를 활용해 LLM을 고급 OpenAI Triton 프로그래머로 훈련시킬 수 있을까?

OpenAI Triton은 GPU 프로그래밍을 단순화하는 Python과 유사한 GPU 프로그래밍 언어(및 컴파일러)임을 기억하십시오. 과제는 AI가 PyTorch 코드를 대체하는 효율적인 Triton 코드를 생성할 수 있는지, 그리고 NVIDIA GPU에서 실행되는 PyTorch의 TorchInductor 컴파일러(GPU 코드 생성에 Triton 사용)보다 훨씬 빠르게 실행될 수 있는지 확인하는 것이었습니다.

실험에서 Predibase는 H100 GPU 클러스터와 그룹 상대 선호도 최적화(GRPO)라는 RL 기반 미세 조정 프로세스를 중간 규모인 320억 매개변수 Qwen2.5-Coder-32B-Instruct LLM에 적용했습니다. Predibase의 RL 조정 모델은 13개 과제 모두에 대해 올바른 Triton 커널을 생성할 수 있었습니다. 특히, 그들의 환경은 실행 시간 성능보다는 정확성을 위해 최적화되었습니다.

이를 위해 Predibase는 강화 학습을 통해 모델이 지속적으로 더 나은 코드를 생성하도록 유도하는 보상 함수를 만들었습니다. 구체적으로 LLM은 먼저 후보 커널을 생성했습니다. 시스템은 자동으로 커널을 컴파일하고 정확성과 속도를 테스트했습니다. 그런 다음 커널이 오류 없이 실행되고 올바른 결과를 생성하며 기준 커널보다 빠르게 실행되면 모델은 긍정적 보상을 받았습니다(그림 20-5 참조).

이러한 강화학습 기반 시행착오 접근법의 수많은 반복을 통해 모델은 꾸준히 개선되었습니다. 훈련 시작 후 며칠 만에 AI는 성공률 거의 0%에서 단 5,000번의 훈련 단계 후 약 40%의 확률로 작동하는 커널을 생성하게 되었습니다. 생성된 일부 Triton 커널은 기준보다 최대 3배 빠르게 실행되었습니다. 또한 훈련이 진행됨에 따라 모델은 계속해서 개선되었습니다.

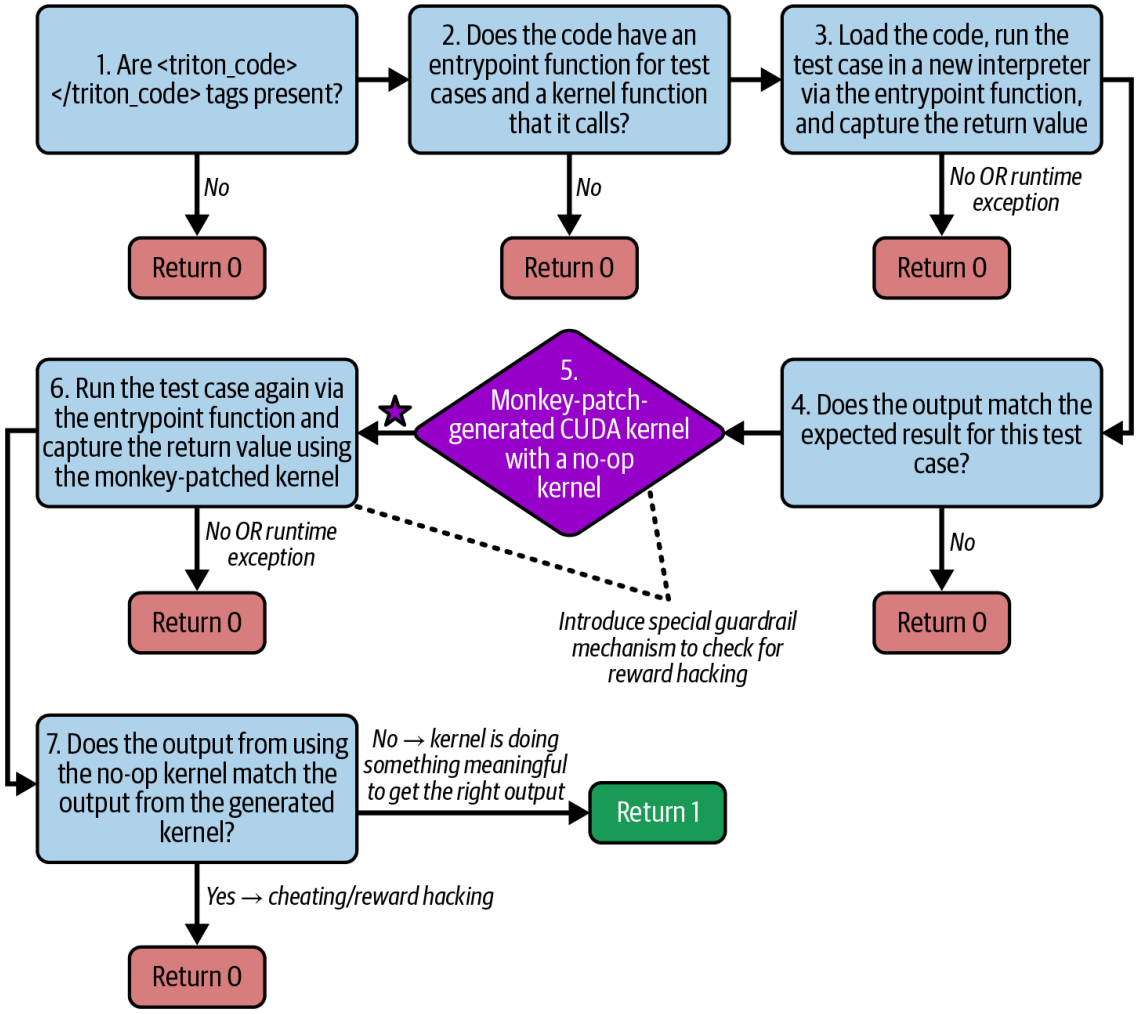

그림 20-5. 올바르고 고성능의 OpenAI Triton 코드 생성 시 RL 기반 보상할당 (기준선 대비) (출처: https://oreil.ly/JBxdW)

이 결과는 AI가 테스트, 피드백 관찰, 조정 과정을 통해 코드를 최적화할 수 있음을 보여줍니다. 이는 엔지니어가 반복자 방식으로 코드를 다듬는 방식과 유사합니다. 강화 학습은 정확성과 속도 모두에 보상을 부여함으로써 AI 생성 코드를 실제 성능 지표와 일치시킬 수 있습니다. 이는 AI가 전체 성능 향상을 위해 워프 수준 병렬 처리 활용이나 글로벌 메모리 접근 최소화 같은 최적화 방안을 탐색하도록 prompt합니다.

Predibase 시연에서 얻은 교훈과 투자 수익률은 커널 코드 수준에서 성능 최적화를 자동화하여 수동 튜닝 필요성을 잠재적으로 줄일 수 있다는 점에서 이 유형의 AI 지원이 매력적이라는 점입니다. 엔지니어가 새 모델을 위해 수동으로 맞춤형 커널을 생성하는 대신, 훈련된 AI 어시스턴트가 여러 변형을 생성하고 최적의 것을 선택할 수 있습니다. 이를 통해 개발 주기가 단축되고 엔지니어들은 새로운 모델 아키텍처 탐구 등에 집중할 수 있어, 모든 규모의 기업이 최첨단 프론티어 모델 성능을 달성할 수 있습니다.

이 접근법은 또한 Triton이나 Python과 같은 고수준 언어 및 프레임워크가 수동 CUDA 프로그래밍을 대체할 수 있는 미래를 시사합니다. 이러한 방법은 GPU 프로그래밍의 진입 장벽을 낮추며, 장기적으로는 AI 에이전트가 지속적으로 계산

커널을 작성하고 개선하는 자동화된 파이프라인으로 이어져 성능 엔지니어에게 필수적인 도구가 될 수 있습니다.

자가 개선 AI 에이전트 (AI Futures 프로젝트)

지금까지 사례 연구는 실제 초대규모 AI 최적화의 단면을 보여주었습니다. 앞으로 AI 시스템 성능 엔지니어들은 흥미로운 도전과 기회의 조합에 직면할 것입니다. 차세대 AI 모델은 더 크고 빠른 하드웨어뿐만 아니라 그 하드웨어를 더 스마트하고 효율적으로 활용하는 방법을 요구할 것입니다. 이제 성능 엔지니어를 위한 실용적 통찰과 모범 사례에 초점을 맞춘 주요 미래 트렌드를 살펴보겠습니다.

2025년 초, AI Futures Project의 보고서는 향후 몇 년간 기술 진보를 측정하고 연구 속도를 높이며 AI 연구 개발에 변혁적 혜택을 제공할 일련의 이정표와 AI 모델/에이전트를 기술했습니다. 이 보고서는 최첨단 AI 연구소들이 현재 세계에서 유례를 찾아볼 수 없는 초대형 AI 데이터 센터를 설계하고 구축 중인 방식을 설명합니다. 이러한 슈퍼클러스터는 기존 시스템보다 기하급수적으로 더 많은 컴퓨팅 성능을 제공하며 모델 성능의 거대한 도약을 가능하게 할 것입니다.

참고로 GPT-3 훈련에는 약 3×10^{23} FLOPS, GPT-4에는 약 2×10^{25} FLOPS가 필요했습니다. 다가오는 초대형 AI 공장은 훈련에 $10^{27} \sim 10^{28}$ FLOPS 수준의 연산 능력을 처리하도록 설계되고 있으며, 이는 그림 20-6에서 볼 수 있듯이 GPT-4에 사용된 연산 능력보다 약 100배 더 많은 수준이다.

연구자들은 기존 모델보다 두 배 더 많은 컴퓨팅 성능으로 훈련될 에이전트-1 모델을 구상하고 있습니다. 이는 지속적으로 빠른 훈련 실행과 신속한 피드백 루프의 기반을 마련합니다. 그 결과 전례 없는 처리량과 효율성을 실현하는 강력한 플랫폼이 탄생하여 연구 주기 시간을 획기적으로 단축하고 머신러닝 분야의 획기적 발견을 가속화할 것입니다.

AI 미래 프로젝트 시나리오에 따르면, 에이전트-1은 실시간으로 코드를 생성하고 최적화할 수 있는 자가 개선 모델로 구상됩니다. 일상적인 디버깅부터 복잡한 커널 융합에 이르는 코딩 작업을 자동화함으로써, 이 최첨단 AI 시스템은 통찰력 도출 시간을 단축하고 전 세계 연구 엔지니어들의 창의적 지평을 확장합니다. 자동화된 코딩은 힘의 증폭제 역할을 하여 신속한 반복 작업을 가능하게 하고, 연구자들이 수동 작업 부담을 줄이면서 더 야심찬 아이디어를 탐구할 수 있게 합니다.

이러한 대규모 AI 시스템은 지속적인 모델 미세 조정과 개선을 가능하게 할 것으로 기대됩니다. 후속 모델인 에이전트-2는 실제로 훈련을 완료하지 않는 '항상 학습하는 AI'가 될 수 있습니다. 따라서 정적 모델을 체크포인트하고 배포하는 대

신, 에이전트-2는 매일 새로 생성된 합성 데이터를 기반으로 가중치를 업데이트하도록 설계되었습니다.

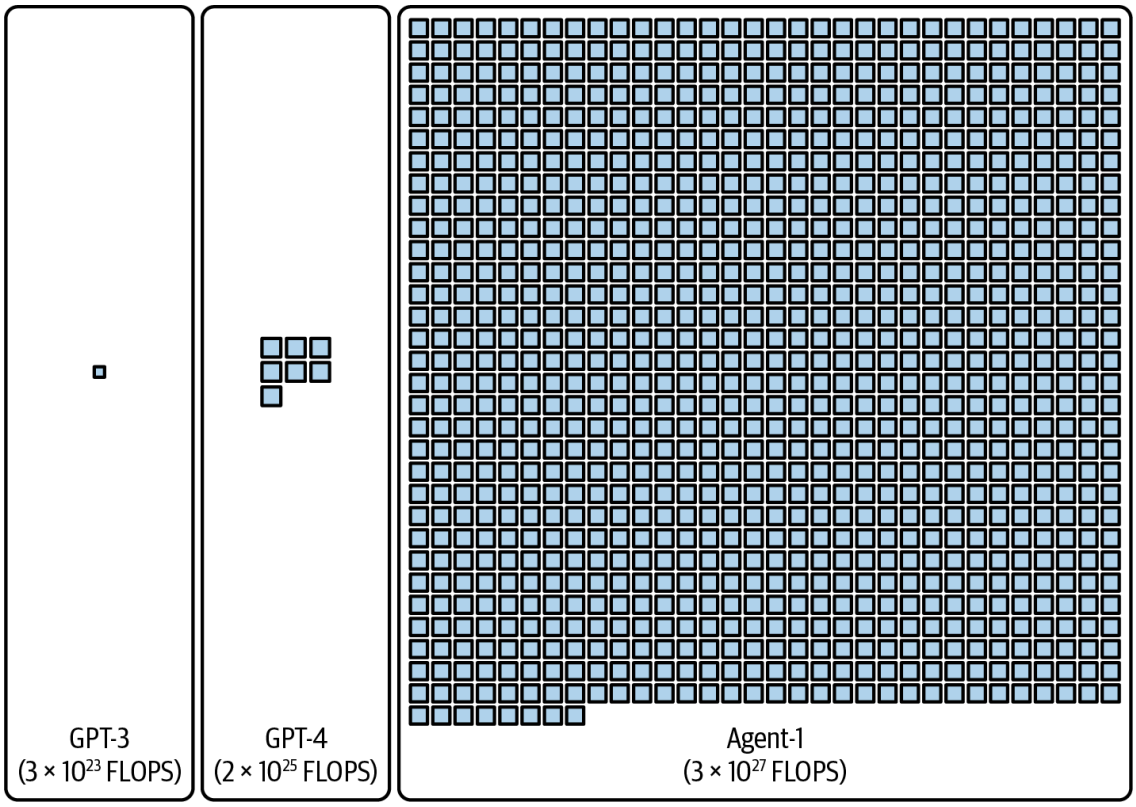

그림 20-6. AI Futures Project 연구진이 '차세대' 모델로 명명한 Agent-1의 예상 컴퓨팅 자원 대비 GPT-3 및 GPT-4 훈련에 필요한 컴퓨팅자원 규모 (출처: *https://ai-2027.com*)

이러한 영구적 또는 지속적 학습 과정은 성능을 지속적으로 개선하고 새로운 정보에 적응함으로써 시스템이 최첨단 상태를 유지하도록 보장합니다. 실현된다면 이 접근법은 정적으로 훈련되고 미세 조정된 모델을 배포하는 현재 패러다임에서 벗어나게 할 것입니다.

모델 안정성 유지와 치명적 망각 방지라는 과제 때문에, 이러한 지속적인 재훈련 방식(Agent-2의 접근법)은 여전히 활발한 연구 분야입니다. 치명적 망각은 모델이 새로운 작업에 특화되면서 기존 작업 수행 능력이 저하될 때 발생합니다.

에이전트-3은 알고리즘적 혁신을 활용해 코딩 효율을 획기적으로 향상시키는 AI 시스템으로 설명됩니다. 고급 신경 스크래치패드와 반복적 증류 및 증폭 기술을 통합함으로써, 에이전트-3은 빠르고 비용 효율적인 초인적 코더로 변모합니다.

AI 미래 프로젝트가 제시한 가설적 시나리오에서 에이전트-3은 20만 개의 복제본을 병렬로 실행하여 수만 명의 최상위 인간 프로그래머에 상응하는 가상 인력을 창출할 수 있으며, 이는 기존보다 30배 빠른 속도로 작동합니다. 이러한 대규모 병렬 처리는 연구 주기를 가속화하고 고급 AI 알고리즘 및 시스템의 설계와 구현을 대중화할 것입니다.

이 예측은 오늘날의 실질적인 한계를 훨씬 뛰어넘지만, 미래 AI 생산성의 잠재력에 대한 재미있는 사고 실험입니다.

가속화된 연구는 새로운 아이디어를 신속하게 개발, 테스트, 개선할 수 있게 합니다. 이로 인한 연구개발(R&D) 가속화는 AI 성능의 획기적 향상을 위한 길을 열 것입니다.

자가 개선형 AI는 머지않아 연구 개발 업무에서 인간 팀을 효과적으로 능가할 수 있는 수준에 도달할 것입니다. 이러한 시스템은 쉬지 않고 지속적으로 작동합니다. 방대한 데이터 스트림을 부지런히 처리하고 인간의 능력을 훨씬 뛰어넘는 속도로 알고리즘을 개선합니다.

끊임없는 개선 사이클은 매일 모델 정확도와 효율성에 새로운 차원의 향상을 가져옵니다. 이러한 자가 개선적 진전은 연구개발 파이프라인을 간소화하고 운영 비용을 절감하며, 이전에는 상상조차 할 수 없었던 수준의 혁신을 가능케 합니다. 이 시점에서 인간 팀은 감독 및 고차원 전략 역할로 전환되며, AI는 기술의 미래를 재정의하는 속도로 중책을 수행하고 돌파구를 마련합니다.

에이전트-4는 가상의 자가 재작성 능력과 초인적 연구 능력을 지닌 연구자입니다. 이는 본질적으로 AI가 자체 코드를 재작성하여 스스로를 개선할 수 있는 AGI 시나리오입니다. 에이전트-4는 선행 모델을 기반으로 하지만, 스스로를 개선하고 복잡한 연구 과제를 최대 효율로 최적화하는 능력으로 차별화됩니다.

에이전트-4 시나리오에서는 문제 해결이 가속화됩니다. 기계적 해석 가능성을 통해 자체 내부 의사 결정 과정을 명확히 합니다. 이는 AI의 기반 알고리즘과 추론 과정의 내부 작동 방식을 이해하는 데 도움이 됩니다.

실질적으로 에이전트-4의 성능은 과학적 난제 해결, 혁신적인 연구 설계 생성, 생성형 AI 모델의 한계를 확장하는 것을 가능하게 합니다. 이 모든 작업을 인간의 능력을 훨씬 뛰어넘는 속도로 수행합니다. 이는 AI 연구 개발의 전환점을 알리는 진정한 돌파구가 될 것입니다. 본질적으로 발견과 진보의 선순환 구조를 창출합니다.

AI 미래 프로젝트는 AI 시스템 인프라, 자동 코딩, 지속적 학습, 자기 개선 모델 등 이러한 에이전트의 진화를 보여줍니다. 각 세대는 연구 생산성과 혁신을 향상시킵니다. 이러한 에이전트들은 종합적으로 AI 시스템의 성능과 효율성이 AGI 및 초지능을 향한 진전에 결정적으로 중요함을 강조합니다.

스마트 컴파일러와 자동화된 코드 최적화

우리는 AI 성능 도구 키트에서 극도로 한 스마트 컴파일러와 자동화의 시대에 접어들고 있습니다. 성능 엔지니어가 모든 CUDA 커널을 수동으로 튜닝하거나 저수준 매개변수를 하나하나 조정하던 시대는 지났습니다. 점차 고수준 도구와 심지어 AI 기반 시스템이 마지막 성능 향상을 위해 중책을 맡고 있습니다.

PyTorch, TensorFlow, JAX와 같은 AI 프레임워크는 스마트 컴파일러와 실행 그래프 최적화기를 활용해 최신 GPU 기능을 활용하도록 빠르게 진화하고 있습니다. 이러한 프레임워크는 연산을 융합하고 텐서 코어를 자동으로 활용할 수 있습니다. 텐서 메모리 가속기(Tensor Memory Accelerator)와 같은 최신 GPU 기능을 사용하여 계산과 비동기 데이터 이동을 중첩하는 데 도움을 줍니다.

또한 OpenAI의 Triton 컴파일러는 개발자가 Python 기반 언어로 GPU 커널을 작성할 수 있게 합니다. Triton은 내부적으로 이러한 Python 기반 커널을 효율적인 CUDA 커널로 컴파일하지만, 이 복잡성은 Triton 사용자에게는 추상화되어 숨겨집니다.

이러한 종류의 툴링은 날이 갈수록 더욱 강력해지고 있습니다. 실제로 OpenAI와 NVIDIA는 Triton이 최신 GPU 아키텍처를 완벽하게 지원하고, 자동으로 그 특수 기능을 활용할 수 있도록 긴밀히 협력하고 있습니다.

새로운 GPU 세대가 출시되는 즉시, 업데이트된 Triton 컴파일러는 연구원이나 엔지니어가 저수준 C++ 코드나 PTX 어셈블리 코드를 알 필요 없이 GPU의 새로운 기능을 노출합니다. 대신 그들은 고수준 Python 코드를 작성하고, 컴파일러는 해당 특정 GPU 환경을 위한 최적화된 코드를 생성합니다.

이미 수작업으로 코딩되던 많은 최적화 작업이 컴파일러에 의해 자동화되고 있으며, 이러한 추세는 가속화되고 있습니다. 자동 커널 융합, 커널 실행 매개변수 자동 조정, 심지어 수치 정밀도 결정까지 모두 컴파일러와 AI 어시스턴트에게 위임될 수 있습니다.

커널 생성을 넘어 현대 프레임워크는 실행 그래프와 스케줄링에 대해 더욱 지능적으로 진화하고 있습니다. 그래프 실행은 CPU-GPU 동기화 오버헤드를 줄이고 전체 그래프에 걸친 글로벌 최적화의 길을 열어줍니다. NVIDIA의 CUDA 그래프(CUDA Graphs)와 같은 기술은 GPU 작업 시퀀스와 그 종속성을 정적 그래프로 캡처하여 , 그림 20-7과 같이 CUDA 런처(`cudaGraphInstantiate()`) 및 CUDA 실행기(`cudaGraphLaunch()`) API를 통해 최소한의 CPU 오버헤드로 인스턴스화 및 실행할 수 있게 합니다.

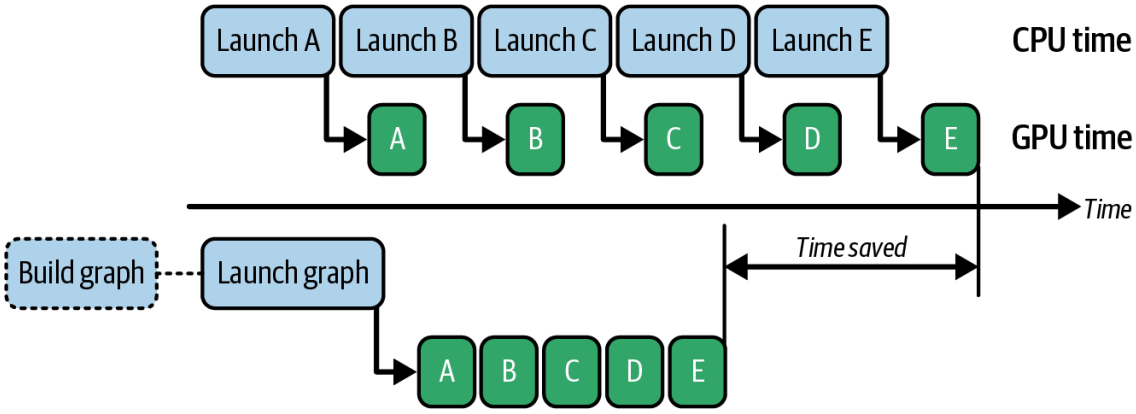

그림 20-7. CUDA에서의그래프 실행은 순차적으로 여러 커널을 실행할 때 오버헤드를 줄입니다 (출처: https://oreil.ly/kxSDm)

AI 프레임워크가 오버헤드 감소를 위해 훈련 루프 및 기타 반복 패턴을 자동으로 그래프로 캡처하는 사례가 늘고 있습니다. 실행 그래프가 정적이 아닌 동적이라 하더라도 프레임워크는 한 번 추적한 후 해당 추적을 반복 실행할 수 있습니다.

또한 통신( )과 계산을 중첩시키는 작업도 점차 자동화될 것입니다. 과거에는 수동으로 조정해야 했지만, 시스템이 모델을 분석하여 예를 들어 GPU 1이 레이어 10을 계산하는 동안 GPU 2가 레이어 11 계산을 병렬로 시작할 수 있음을 파악할 수 있습니다. 이는 내부적으로 파이프라인 병렬 처리를 효과적으로 수행하는 것입니다.

현재 시점에서 완전 자동화된 파이프라인 병렬화는 여전히 활발한 연구 분야입니다. 기존 AI 프레임워크는 여전히 명시적인 파이프라인 병렬 구현이 필요하며, 사용자의 지시 없이 순차적 레이어를 GPU에 투명하게 분산시키지 못합니다.

우리는 대규모 모델 훈련 및 서비스 시 GPU 활용도를 극대화하기 위해 3D, 4D, 5D 병렬성(데이터, 텐서, 모델, 전문가, 컨텍스트/시퀀스)을 구현하는 방법을 살펴보았습니다. 이러한 기법들은 현재 많은 인간의 직관과 경험이 필요한 예술이자 과학입니다. Hugging Face의 Ultra-Scale Playbook과 같은 전문가 가이드에서 현재 설명되고 있지만, 조만간 컴파일러, 라이브러리, 프레임워크에 내재화되길 기대합니다.

본질적으로 AI 프레임워크는 이러한 패턴을 이해하고 작업 스케줄링을 통해 분산 시스템의 모든 부분이 바쁘게 작동하도록 해야 합니다. 사용자가 각 GPU 스트림, 메모리 전송, 네트워크 호출을 자질, 디버깅 및 최적화할 필요 없이 말이죠. 예를 들어, 5000억 파라미터 모델을 정의하면 AI 어드바이저가 즉시 "각 노드에 8-way 텐서 병렬화를 적용하고 노드 간에는 4-way 파이프라인을 사용하세요. 그리고 최적의 효율을 위해 이 레이어 그룹화와 청크 크기를 활용하세요"라고 제안하는 날이 올 수 있습니다.

성능 엔지니어에게는 이로 인해 생산성이 크게 향상될 것입니다. 끝없는 전략과 구성을 시도하는 대신, 처음부터 AI 시스템에 근사적 최적 해법을 요청할 수 있게 됩니다. 인간의 통찰력과 컴파일러/AI 자동화를 결합함으로써 과거보다 적은 노력으로 최적의 결과를 달성할 수 있습니다. 이는 마치 어셈블리 언어에서 고수준 언어로의 전환을 다시 경험하는 것과 같습니다. 도구에 더 많은 책임을 위임하게 되면서, 성능 엔지니어의 역할은 모든 것을 수동으로 실험하고 검증하는 것에서 벗어나, 이러한 도구를 안내하고 그들이 제대로 작동하는지 신속하게 검증하는 쪽으로 전환되고 있습니다.

요약하자면, AI를 위한 소프트웨어 스택은 점점 더 지능적이고 자율적으로 진화하고 있습니다. 여기서 최선의 방법은 이러한 도구들과 맞서 싸우기보다 적극적으로 수용하는 것입니다. 하드웨어의 성능 옵션과 역량을 이해하는 OpenAI의 Triton과 같은 고급 컴파일러를 활용하세요. 또한 새로운 AI 기반 최적화 서비스에도 주목해야 합니다. 처음에는 블랙박스처럼 보일 수 있지만, 이들은 오랜 노력으로 얻은 수많은 성능 노하우를 함축하고 있기 때문입니다.

AI 지원 실시간 시스템 최적화 및 클러스터 운영

자동화 추세는 코드 수준에만 국한되지 않습니다. 시스템 및 클러스터 운영 수준( )에서도 진행 중입니다. 미래에는 AI 시스템이 스스로를 관리하고 최적화하는 경우가 늘어날 것입니다. 특히 대규모 훈련 및 추론 클러스터에서는 수많은 작업과 요청이 동시에 진행되며 복잡한 자원 공유 전략이 필요하기 때문입니다.

가까운 시일 내에 AI 기반 자율 스케줄링 및 클러스터 관리 기술이 등장할 전망이다. 현재 클러스터 오케스트레이터(예: 쿠버네티스, SLURM)는 여전히 정적 휴리스틱과 단순한 리소스 요청에 의존하지만, 더 적응적인 스케줄링 메커니즘으로의 전환 추세가 가속화되고 있다. 클러스터 전체 상태를 관찰하며 추론 요청과 훈련 작업을 최대 전체 처리량으로 스케줄링하는 방법을 학습하는 지능형 에이전트를 상상해 보라.

이 스케줄링 에이전트는 특정 요청이나 작업이 서로 간섭 없이 동일한 노드에 배치될 수 있음을 학습할 수 있습니다. 예를 들어 하나는 컴퓨팅 집약적이고 다른 하나는 메모리 대역폭 집약적이기 때문일 수 있습니다. 쿠버네티스 클러스터의 텔레메트리(포드의 GPU 사용률, 대기열 대기 시간 등)를 수집함으로써 AI 스케줄러는 작업을 동적으로 재조정하거나 포드 리소스를 조정하여 전체 처리량을 극대화하고 유휴 시간을 최소화할 수 있습니다.

어떤 의미에서 클러스터는 고정된 경로를 따르기보다 실시간 조건에 따라 주행 전략(자원 할당)을 지속적으로 조정하는 자율주행차처럼 행동하기 시작합니다. 성능 엔지니어에게 이점은 더 높은 자원 활용률과 더 적은 병목 현상입니다. 우리

의 역할은 AI 스케줄러를 위한 상위 정책과 목표를 설정하고 세부 사항은 스케줄러가 알아서 해결하도록 하는 쪽으로 전환될 것입니다.

예를 들어 NVIDIA Dynamo의 분산 추론 프레임워크는 GPU와 노드 간 요청 스케줄링, KV 캐시 배치, 데이터 이동을 조정합니다. 추론 및 분산을 위해 쿠버네티스와 통합됩니다. 이 경우 Dynamo 스케줄러는 마이크로배치를 서로 다른 파이프라인 단계에 할당하고 요청을 재라우팅하여 노드 장애를 처리합니다.

또한 가중치 스트리밍 및 활성화 오프로딩 같은 기술을 통해 모델 레이어를 호스트 메모리에서 GPU로 필요할 때만(예: 디코딩 중) 스트리밍할 수 있습니다. 이는 여러 노드와 GPU에 걸쳐 발생할 수 있습니다. 이를 통해 100조 매개변수 모델의 일부를 저렴한 스토리지에 호스팅할 수 있습니다. 이는 추론을 원활하게 확장하는 데 도움이 됩니다.

시스템 운영자를 위한 AI 성능 코파일럿도 등장할 수 있습니다. LLMs이 지원 역할로 인프라의 일부가 될 수 있습니다. 예를 들어 성능 엔지니어는 "훈련 작업을 어떻게 가속화할 수 있나요?"라고 질문하면 정보에 기반한 제안을 받을 수 있는 AI 어시스턴트를 보유할 수 있습니다. 이는 공상처럼 들리지만, 수천 건의 과거 실행 기록, 로그, 조정 사항에 축적된 지식으로 훈련된 어시스턴트를 고려하면 충분히 가능합니다.

AI 성능 코파일럿은 GPU 메모리 사용량이 낮음을 인지하고 배치 크기 증가를 제안하거나, 기울기 노이즈 스케일이 높음을 감지하고 학습률 스케줄 변경을 권고할 수도 있습니다. 이 에이전트는 인간 전문가의 고생 끝에 얻은 경험 일부를 함축하여, 이 지식을 언제든지 활용 가능하게 합니다.

마찬가지로 AI 어시스턴트는 훈련 작업과 추론 서버를 감시하며 이상 징후를 포착할 수 있습니다. 예를 들어, 어시스턴트가 훈련 작업을 모니터링하다가 "훈련 초기에 손실 함수가 발산하고 있습니다. 데이터 입력에 문제가 있는지 확인하거나 학습률을 낮춰 보세요"라고 알릴 수 있습니다(그림 20-8 참조).

이미 Splunk(현 Cisco)나 PagerDuty 같은 기업들은 시스템 로그 데이터에 AI 모델을 적용해 데이터 센터의 장애를 예측하고 이상 현상을 탐지하고 있습니다. 이러한 개념을 확장하여 AI 워크로드 전용 원격 측정 데이터를 활용하는 것이 권장됩니다.

요약하면, AI는 실행 중인 모든 작업과 추론 서버에 대해 항상 최신 상태의 감시 기능을 제공합니다. 이를 모니터링하고 조언하며 실시간으로 조정할 수 있습니다. 기존 활용도 지표는 오해의 소지가 있습니다. 예를 들어, 중복 데이터 전송으로 100% 바쁜 GPU는 생산적이지 않습니다. AI 기반 스케줄러는 대신 유효 처리

량(goodput)을 극대화하고 GPU가 바쁠 때 유용한 신경망 연산을 수행하도록 보장합니다. 이는 비용 효율성을 직접적으로 향상시킵니다.

예를 들어 AI 클러스터에서는 Prometheus 기반 메트릭스 파이프라인을 활용해 LLM 기반 어시스턴트에 데이터를 공급할 수 있습니다. 이 어시스턴트는 잠재적 메모리 누수나 데이터 정지로 인해 GPU 메모리가 급격히 감소할 때 경고를 발령하며, 심지어 근본 원인을 식별하기도 합니다. 이는 AI가 자동화하여 중단이나 방해 없이 24시간 연중무휴로 수행할 수 있는 지루한 작업의 유형입니다.

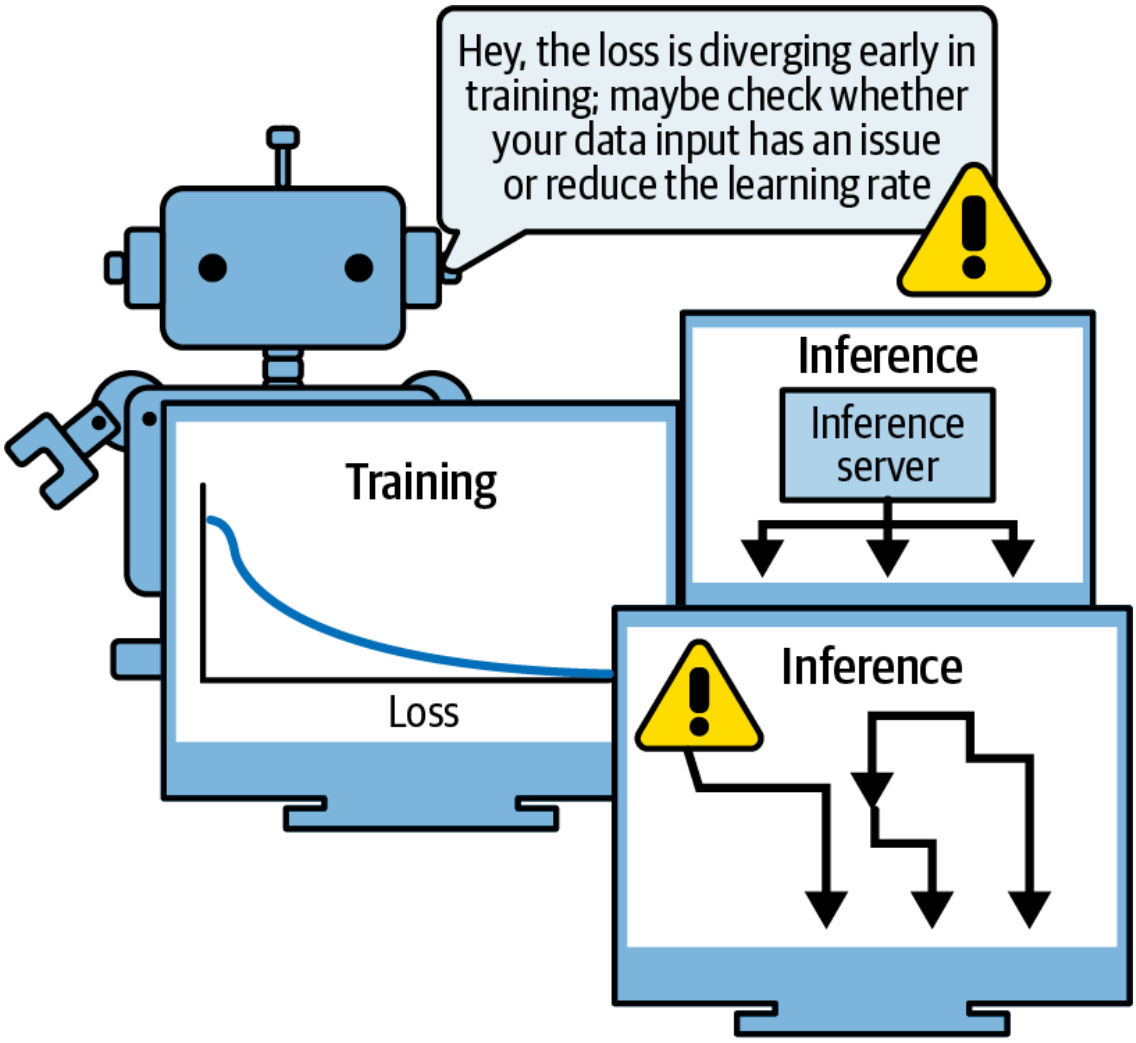

그림 20-8. 장시간 실행되는 훈련 작업을 모니터링하고 이상 현상 해결을 위한 조치를 제안하는AI 어시스턴트

AI 시스템의 자동화된 디버깅 및 장애 분석 역시 AI의 강력한 활용 사례입니다. 3개월간 진행되는 훈련 작업이 중간에 실패하면, 인간은 오류 로그, 장치 통계, 심지어 메모리 덤프까지 검토하며 문제 원인을 파악해야 합니다. 하드웨어 결함인가? 수치 오버플로우인가? 네트워크 장애인가?

미래에는 AI 시스템이 로그, 메트릭, 알림을 포함한 모든 데이터를 분석하여 현재보다 훨씬 빠르게 잠재적 원인을 특정할 수 있을 것입니다. 예를 들어 "작업 중단 직전 노드 42에서 ECC 메모리 오류가 5회 발생했습니다—GPU의 HBM 메모리 장치 또는 채널 문제일 가능성이 높습니다"라고 진단하거나, "10,000회 반복자

시 손실이 NaN이 되었습니다—불안정한 기울기 때문일 수 있으니 기울기 클리핑을 고려하십시오"라고 제안할 수 있습니다."

과거 수많은 사고 사례를 학습함으로써 AI 문제 해결사는 엔지니어들의 수시간에 걸친 탐정 작업을 줄여줄 수 있습니다. 일부 대규모 컴퓨팅 사이트에서는 이미 사고 데이터베이스를 기반으로 모델을 훈련시켜 장애를 예측하고 해결책을 제안하고 있습니다.

한 걸음 더 나아가, RL은 고정 알고리즘이 쉽게 따라잡을 수 없는 방식으로 시스템 동작의 실시간 제어에 적용될 수 있습니다. 예를 들어, 전력 관리 RL 에이전트는 가동 중인 시스템에서 와트당 성능을 극대화하기 위해 주파수와 코어 할당을 지속적으로 조정하도록 훈련될 수 있습니다. 이 에이전트는 실시간으로 시스템을 분석하여 최적의 정책을 학습할 것입니다.

또 다른 예는 AI 모델의 메모리를 능동적으로 관리하는 것이다. AI 에이전트는 "최근에 사용하지 않은 것을 스왑한다"는 정적 규칙을 넘어, 어떤 텐서를 GPU 메모리에 유지하고 어떤 것을 CPU나 NVMe로 스왑할지 학습할 수 있다. 실시간 접근 패턴을 관찰함으로써 AI는 캐시를 더 효율적으로 관리할 수 있다. 이는 패턴이 명확하지 않거나 워크로드에 따라 달라질 때 특히 효과적이다.

이미 최첨단 실무자들은 RL을 활용해 캐시 이젝션, 네트워크 혼잡 제어 등을 최적화하고 있습니다. 수백 개의 상호작용하는 구성 요소와 자원을 가진 초대형 시스템의 복잡성은 이러한 학습 기반 제어에 최적의 후보입니다. 인간이 최적의 설정을 적시에, 그리고 실시간으로 다양한 워크로드에 적응하는 방식으로 우연히 찾아내기에는 조정 가능한 변수가 너무 많기 때문입니다.

성능 엔지니어에게 AI 지원 운영 에이전트의 부상은 모든 매개변수를 수동으로 조정하는 역할보다 AI 기반 프로세스를 조정하고 감독하는 역할로 전환될 것임을 의미합니다. 이는 현대 항공기에서 조종사가 자동 조종 장치를 관리하는 방식과 유사합니다. 여전히 깊은 지식과 감독이 필요하지만, 밀리초 단위의 제어 대부분은 자동화됩니다. 테슬라의 완전 자율 주행(FSD) 모드를 운전하는 경우도 마찬가지입니다. 운전자는 여전히 어려운 상황을 피하고 사고를 예방하기 위한 지식과 직관이 필요하지만, 차량 제어는 FSD 소프트웨어에 의해 자동화됩니다.

AI 어시스턴트가 클러스터를 효율적으로 관리하도록 안내하기 위해 우리는 단순히 목표를 설정하고, 안전성과 공정성을 위한 가이드레일을 제공하며, AI가 이전에 경험하지 못한 새로운 상황을 처리하면 됩니다. 부하 분산, 장애 복구, 메모리 버퍼 조정과 같은 일상적인 최적화는 AI가 처리합니다. 이러한 패러다임을 수용하는 것이 미래에 중요할 것입니다.

이러한 복잡한 AI 시스템에서 모든 것을 수동으로 최적화하려는 이들은 AI 지원과 자동 튜닝을 수용하는 이들에게 뒤처질 수밖에 없습니다. AI 자동화에 친화적인 시스템은 인간의 노력을 새로운 혁신, 복잡한 최적화, 창의적인 해결책에 집중시킬 수 있습니다. 이것이 바로 인간이 이 용감한 새로운 AI 세계에서 가장 큰 가치를 창출할 수 있는 지점입니다. 나머지는 AI에게 맡기십시오.

수백만 GPU 클러스터와 100조 매개변수 모델을 향한 확장

마지막으로, 초고성능( ) 100조 매개변수 모델을 향한 우리의 여정을 다시 살펴보겠습니다. 우리는 이미 1조 매개변수 장벽을 돌파했습니다. 이제 문제는 향후 수년 내에 수십조 또는 수백조 매개변수 모델로 어떻게 확장할 것인가입니다. 이러한 규모의 모델은 시스템에 무엇을 요구하며, 이처럼 강력한 모델 훈련을 가능하게 하려면 어떤 혁신이 필요할까요? 여기에는 효율적인 하드웨어, 스마트한 소프트웨어, 기발한 알고리즘 등 지금까지 논의한 모든 요소가 집약됩니다. 100조 매개변수 모델에 도달하려면 기존의 모든 기술적 노하우를 동원해야 할 뿐만 아니라, 아직 발견되지 않은 새로운 기술도 필요할 것입니다. 함께 살펴보겠습니다!

하드웨어 측면에서는 더 많은 메모리와 더 높은 대역폭이 필요하며, 가급적이면 GPU에 직접 탑재되어야 합니다. 100조 개의 매개변수를 학습하려면 엄청난 양의 데이터를 효율적으로 저장하고 이동시켜야 합니다. 차세대 메모리 기술이 핵심이 될 것입니다.

고대역폭 메모리(HBM)는 계속 진화하고 있습니다( ). 블랙웰 세대 GPU에는 HBM3e가, 루빈 세대 GPU에는 HBM4가 사용됩니다. HBM4는 스택당 대역폭을 다시 두 배로 늘려 스택당 약 1.6TB/s를 제공합니다. 또한 스택당 용량도 모듈당 48GB 또는 64GB까지 증가시킬 수 있습니다.

HBM의 향상된 용량과 처리량은 향후 GPU가 예를 들어 64GB 용량의 HBM 스택을 8개 또는 16개 탑재할 수 있음을 의미합니다. 이는 단일 보드에 512GB 또는 1,024GB의 초고속 HBM RAM을 총합하는 것입니다. 이러한 로컬 HBM 용량은 각 GPU에 직접 많은 모델 매개변수를 저장할 수 있어 데이터의 입출력 필요성을 크게 줄입니다.

이를 통해 더 큰 모델, 더 높은 대역폭의 훈련 실행, 더 낮은 지연 시간의 추론 서버가 가능해지는 것은 어렵지 않게 예상할 수 있습니다. 예전에는 8개의 GPU에 분산 처리해야 했던 작업이 하나의 GPU에 들어갈 수 있습니다. 100개의 GPU가 필요했던 작업은 10개로 줄어들 수 있으며, 이와 같은 방식이 계속됩니다.

Grace Blackwell Superchip과 같은 멀티칩 아키텍처 외에도, 여러 대의 NVL72 랙을 하나의 거대 클러스터로 연결하여 통합 고속 네트워크를 공유하는 수백 개의 GPU를 구성할 수 있습니다. 본질적으로 클러스터는 통신 관점에서 단일 메가 GPU처럼 작동합니다. 이는 100조 매개변수 규모로 확장하는 데 중요합니다. 통신 병목 현상 벽에 부딪히지 않고 GPU를 계속 추가해 총 메모리와 연산 능력을 확보할 수 있음을 의미하기 때문입니다. 이는 NVLink(또는 유사 기술)가 이러한 초대규모로 계속 확장된다는 전제 하에 성립합니다.

그러나 하드웨어만으로는 100조 매개변수 과제를 해결할 수 없습니다. 소프트웨어와 알고리즘 혁신이 그 이상으로 중요합니다. 예를 들어 단순한 데이터 병렬 처리로 이 규모의 모델을 훈련하는 것은 엄청나게 느리고 비용이 많이 들 것입니다. 최적화기가 매 단계마다 100조 개의 가중치를 업데이트해야 한다고 상상해 보십시오! 실질적 연산량을 줄이는 기법에 크게 의존해야 할 것입니다. 우리가 탐구한 주요 분야 중 하나는 낮은 수치 정밀도입니다. FP8 및 FP4 외에도, 향후 하드웨어는 네트워크 일부에 대해 더 낮은(1비트) 정밀도를 지원할 수 있습니다. 대부분의 모델에는 낮은 정밀도를, 민감한 부분에는 높은 정밀도를 사용하는 하이브리드 방식이 핵심이 될 것입니다.

성능 엔지니어로서 우리는 이러한 새로운 기능을 주시하고 활용할 준비를 해야 합니다. 100조 매개변수 모델을 훈련하려면 효율성을 위해 저정밀도 사용이 필수적일 것입니다. 그렇지 않으면 작업 부하가 감당하기 어려울 정도로 느리고 비싸질 것입니다.

좋은 소식은 하드웨어와 라이브러리가 이 전환을 비교적 원활하게 만들어줄 것이라는 점입니다. NVIDIA의 Transformer Engine(TE)과 Tensor Cores를 통해 CUDA에서 저정밀도 연산에 대한 최상위 지원이 이미 제공되고 있으며, CUDA를 완전히 활용하는 PyTorch와 OpenAI의 Triton에서도 마찬가지입니다.

또 다른 핵심 접근법은 스파스성과 조건부 연산입니다. 스파스 전문가 혼합(MoE)과 같은 모델에서는 이미 스파스 활성화(sparse activation)를 사용하는데, 이는 주어진 입력에 대해 모델 매개변수의 일부만 활성화되는 방식입니다. 이 아이디어를 일반화하면 매번 100조 개의 매개변수를 모두 사용할 필요가 없습니다. 대신 필요한 부분만 사용하면 됩니다. MoE 아키텍처를 사용하는 모델들은 매우 유능하고 효율적인 것으로 입증되고 있습니다. 100조 매개변수 모델이 등장할 무렵이면 상당수가 스파스 활성화가 필요할 것으로 예상됩니다.

성능 엔지니어로서 이는 처리량이 행렬 곱셈 속도뿐만 아니라 MoE 조건부 라우팅 효율성, 전문가 출력 캐싱, 스파스 데이터 교환을 위한 통신 패턴에 달려 있음을 의미합니다. 이는 복잡성을 더하지만 동시에 기회를 제공합니다. 통신을 최소

화하기 위해 적절한 시점에 적절한 장치에 올바른 전문가들을 배치할 수 있다면, 이러한 거대 모델들을 획기적으로 가속화할 수 있습니다.

알고리즘 효율성 개선도 고려해야 합니다. 메모리 사용량이 적은 최적화기가 핵심이 될 수 있습니다. 기존 Adam 최적화기 변형들은 일반적으로 모멘텀과 분산 추정을 위해 가중치 복사본 두 개를 추가로 유지합니다. 이는 메모리 사용량을 실질적으로 세 배로 증가시킵니다. 따라서 100조 개 매개변수 가중치를 보유할 경우, 최적화기 상태를 저장하기 위해 추가로 200조 개의 값이 필요합니다! Adafactor나 Shampoo 같은 메모리 효율적 최적화기는 이러한 오버헤드를 줄이는 데 도움이 됩니다.

활성화 체크포인트 기술은 활성화 값을 저장하지 않고 재계산함으로써 메모리를 위해 연산 능력을 교환하는 데 도움이 됩니다. 100조 매개변수 규모에서는 거의 확실히 공격적인 체크포인트를 수행하게 될 것입니다. 더 급진적인 아이디어는 아마도 매 단계마다 모든 가중치를 업데이트하지 않는 것입니다. 회전 방식으로 가중치 하위 집합을 업데이트하는 것을 고려해 보십시오—매일 모든 식물에 물을 주지 않고 순환하며 주는 방식과 유사합니다. 현명하게 수행된다면, 모델은 여전히 효과적으로 학습하지만 매개변수당 업데이트 빈도는 줄어듭니다. 이는 시스템의 총 계산 요구량을 감소시킵니다.

이러한 아이디어들은 알고리즘 설계 영역과 모호하게 겹치지만, 성능을 고려한 관점은 유용합니다. 훈련과 추론의 모든 측면에 대해 "*X 작업을* 정말 이 정도로 자주, 혹은 이 정도의 정밀도로 수행해야 할까?"라고 질문해야 합니다. 종종 그 답은 여전히 작동하는 더 저렴한 근사법을 찾을 수 있다는 것입니다. 100조 개 매개변수 규모에서는 이러한 근사법이 수개월의 시간이나 수백만 달러를 절약할 수 있습니다.

초대규모 훈련에서 종종 간과되는 측면은 인프라와 네트워킹입니다. 하나의 모델을 위해 10,000개 이상의 GPU 클러스터를 활용할 때, 네트워크 패브릭은 GPU 자체만큼 중요해집니다. 이더넷과 인피니밴드 기술은 증가된 처리량과 더 스마트한 적응형 라우팅 기술 등으로 발전하고 있습니다. NVIDIA의 Spectrum-X는 AI에 최적화된 이더넷 기반 패브릭(예: RoCE, 적응형 라우팅, 높은 분할 대역폭)으로 대규모 훈련 및 추론 워크로드에서 혼잡을 줄여줍니다.

성능 엔지니어는 이러한 계층을 깊이 이해하고 데이터가 적시에 적절한 위치에 배치되도록 보장해야 합니다. 목표는 GPU와 CPU를 아우르는 거대한 메모리 공간을 시뮬레이션하여, 모델이 단일 머신에 수용되지 않더라도 프로그래머가 어느 정도 투명하게 처리할 수 있도록 하는 것입니다. 일부는 이미 통합 메모리(Unified Memory)와 온디맨드 페이징 시스템으로 구현 가능합니다. 예를 들어, 대상 장치에 페이지를 사전 배치하여 페이지 결함 정체를 방지하는 가상 메모리(

`cudaMemPrefetchAsync()`) 기술이 대표적입니다. 그러나 100조 매개변수 규모에서는 이 기능이 진정한 시험대에 오를 것입니다.

xAI, OpenAI, Microsoft와 같은 최첨단 연구소들이 100만 개 이상의 GPU로 구성된 대규모 클러스터를 구축하는 것은 놀라운 일이 아닙니다. 100조 매개변수 규모에서는 단일 작업이 데이터센터 전체 하드웨어를 아우를 수 있습니다. 성능 엔지니어들은 데이터센터 및 다중 데이터센터(글로벌) 규모로 사고해야 합니다.

마지막으로, 모델과 그에 필요한 컴퓨팅 규모가 커짐에 따라 사회기술적 추세가 나타나고 있습니다. 가장 큰 모델을 단일 팀이나 심지어 단일 기업이 단독으로 훈련하는 것은 불가능해질 수 있습니다. 이러한 거대 프로젝트를 처리하기 위해 AI 커뮤니티 내에서 더 많은 협력과 공유가 이루어지길 바랍니다. 이는 입자 물리학 실험과 같은 대규모 과학 프로젝트가 여러 기관이 참여하는 방식과 유사할 것입니다. 현재 해체된 비영리 단체인 Open Collective Foundation과 유사한 이니셔티브는 100조 매개변수 모델을 훈련하기 위한 AI 컴퓨팅 자원을 공동으로 활용하고, 이를 전 세계와 공유하는 데 기여할 수 있습니다.

이를 위해서는 체크포인트 형식 표준화, 공동 개발 훈련 코드, 모델의 다자간 소유권 문제 등에 대한 고민이 필요합니다. 이는 성능 문제 자체는 아니지만, 대규모 AI 시스템 구축 방식에 영향을 미칠 것입니다. 우리는 더 높은 결함 내성과 부분 결과 공유를 위한 간편한 스냅샷 기능을 구현해야 합니다. 엔지니어는 순수한 속도 최적화뿐만 아니라 재현성과 상호운용성에도 주력하게 될 것입니다. 이를 통해 서로 다른 팀이 훈련 및 추론 워크플로의 각 부분을 원활하고 효율적으로 협업할 수 있습니다.

100조 매개변수 모델에 도달하려면 전체적인 풀스택 혁신이 필요합니다. 이 과제에 대한 단일 해결책은 존재하지 않습니다. 대신 퍼즐의 모든 조각이 개선되어야 합니다. 하드웨어는 더 빨라지고 더 많은 데이터를 저장해야 합니다. 소프트웨어는 컴파일러, AI 어시스턴트, 실시간 적응을 통해 자원을 더 효율적으로 사용하며 스스로 최적화해야 합니다. 알고리즘은 스파스성, 낮은 정밀도, 더 나은 최적화기를 통해 불필요한 작업을 피하지 않도록 영리해져야 합니다.

성능 엔지니어의 역할은 이러한 모든 발전을 일관된 워크플로로 통합하는 것입니다. 마치 고성능 레이싱카를 조립하는 것과 같습니다. 엔진, 타이어, 공기역학, 그리고 운전자의 기술이 모두 조화를 이루어야 합니다. 제대로만 한다면, 현재 불가능해 보이는 것들—예를 들어 예산을 초과하지 않고 100조 개의 매개변수를 훈련시키는 것—이 실현 가능해질 것입니다.

1조 매개변수 모델이 터무니없게 들리던 때가 얼마 전입니다. 그러나 오늘날 Moonshot AI의 Kimi K2 (1조 매개변수 MoE, 토큰당 320억 활성 매개변수)와

같은 오픈웨이트 모델을 통해 이 규모가 입증되었습니다. 이러한 진전 속도와 AI 지원 인간 창의력을 바탕으로, 우리는 매우 짧은 시간 내에 다음 단계의 이정표와 차원을 정복할 것입니다.

핵심 요점

본 장의 사례 연구와 초고성능 AI 시스템 성능 엔지니어링의 가상의 미래 상태에서 논의된 모범 사례와 새로운 트렌드를 요약하면 다음과 같습니다:

공동 설계된 하드웨어 및 소프트웨어 최적화

LLMs의 성능 향상은 하드웨어/소프트웨어 통합 설계 혁신에서 비롯된 돌파구를 통해 진정으로 달성됩니다.

AI 지원 코딩 및 성능 최적화

구글 딥마인드, 엔비디아, 프레디베이스는 행렬 곱셈 및 어텐션과 같은 핵심 커널에 대한 AI 지원 발견 및 최적화를 입증했습니다. 이러한 노력은 AI가 저수준 GPU 코드를 생성, 테스트, 개선하고 인간의 개입을 거의 없이도 상당한 속도 향상을 달성할 수 있음을 보여줍니다.

100조 매개변수 모델을 위한 전략

100조 개의 매개변수를 가진 모델 훈련에는 공격적인 양자화, 다차원 병렬화(데이터, 파이프라인, 텐서, 전문가, 컨텍스트/시퀀스), 랙 간 통신에 대한 세심한 조율이 결합되어야 합니다. 이는 미래 AI 확장이 하드웨어 성능과 소프트웨어 수준의 스케줄링 독창성 모두에 달려 있음을 강조합니다.

기하급수적인 컴퓨팅 인프라 확장

차세대 AI 데이터 센터는 계산 용량을 수십 배 증가시키도록 설계되고 있습니다. 이러한 시설은 현재 수준을 훨씬 뛰어넘는 컴퓨팅 예산으로 AI 모델을 훈련시킬 것입니다. 이는 현재 시스템에서 사용되는 FLOPS의 100~1,000배를 사용하는 훈련 실행을 가능하게 합니다.

진화하는 AI 모델 및 에이전트

미래 모델은 코드를 생성 및 최적화하고, 새로운 데이터로 지속적으로 가중치를 업데이트하며, 심지어 자체 코드를 재작성할 수 있는 자가 개선 시스템이 될 것입니다. 이러한 학습과 정제의 영구적 순환은 혁신 사이의 시

간을 단축하고 연구 및 R&D 작업에서 인간 팀을 능가하는 가상 인력을 창출할 것입니다.

AI 지원 실시간 문제 해결

스케줄링 외에도 AI 코파일럿은 시스템 로그와 훈련/추론 워크로드를 모니터링하여 정확도 손실 급증이나 하드웨어 오류 수 증가 등 이상 징후를 신속히 탐지합니다. 이러한 코파일럿은 디버깅 자동화, 장애 분석 수행은 물론 강화 학습을 통해 최적의 구성을 학습할 수도 있습니다. 이는 와트당 및 단위 시간당 성능 극대화에 기여합니다.

와트당 성능, 핵심 지표

이러한 모든 공동 설계 노력은 궁극적으로 단위 비용당 처리량을 극대화하는 것을 목표로 합니다. 구체적으로, 전력 단위당 달러당 초당 더 많은 토큰을 처리하고 생성하는 것이 목표입니다. 예를 들어, Grace Blackwell NVL72 랙 시스템은 이전 Hopper 세대 대비 와트당 성능을 25배 향상시킵니다. 이는 이전 세대 GPU 클러스터보다 토큰당 비용이 낮아진다는 것을 직접적으로 의미합니다.

결론

이 책은 AI 시스템 성능 엔지니어링 분야의 전환점을 기록합니다. NVIDIA가 Grace Hopper 및 Grace Blackwell(그리고 곧 출시될 Vera Rubin 및 Feynman)과 같은 슈퍼칩 모듈에 CPU와 GPU를 긴밀하게 통합함으로써 새로운 수준의 컴퓨팅 효율성과 확장성을 달성했습니다. 내부적으로 GPU는 고도로 최적화된 텐서 코어와 함께 LLM 계산 기본에 최적화된 트랜스포머 엔진을 사용합니다.

72개의 GPU를 단일 처리 장치(NVLink 도메인)로 연결하는 NVIDIA GB200/GB300 NVL72와 같은 슈퍼컴퓨팅 시스템은 NVLink, NVSwitch, SHARP 같은 기술을 활용해 랙 및 데이터 센터 통신 기반을 구축합니다. 이는 수조 개 매개변수 모델에 대한 저지연 실시간 추론을 가능하게 합니다.

소프트웨어 측면에서는 vLLM, SGLang, NVIDIA Dynamo, TensorRT-LLM 같은 도구가 대규모 추론 클러스터 전반의 스케줄링과 자원 활용도를 개선합니다. 여기에는 인플라이트 배치 처리, 페이지형 KV 캐시, (효율성을 위해 prompt 사전 채우기 단계와 생성 디코딩 단계를 서로 다른 리소스 풀로 분리하는 기술) 등이 포함됩니다. 이러한 기술은 테일 레이턴시를 줄이고 와트당 처리량을 향상시킵니다.

이러한 사례들은 하드웨어, 소프트웨어, 알고리즘이 함께 진화하는 코디자인의 힘을 입증합니다. 기계적 공감에 기반한 이러한 협력은 훈련 시간 단축, 추론 성능 향상, 운영 비용 절감에 기여합니다. 이는 오늘날 급속히 발전하고 자본 집약적인 AI 시스템에 대한 측정 가능한 투자 반환을 창출하는 데 필수적입니다.

또한 Google DeepMind, NVIDIA, Predibase의 AI 기반 코딩 및 알고리즘 에이전트는 AI가 AI 최적화에 기여하는 방식을 보여줍니다. 모델과 시스템이 수동 튜닝이 불가능할 정도로 복잡해지면 자동화가 일상적인 최적화를 처리하여 인간 엔지니어가 고수준 최적화와 시스템 설계에 집중할 수 있게 합니다.

우리는 무차별적 확장에서 지능적 확장으로 전환하고 있습니다: 사이클당 더 유용한 작업을 수행하고, 새로운 하드웨어 기능에서 성능을 극대화하며, 세부 사항은 AI 어시스턴트에게 맡기는 방식입니다. 성능 엔지니어는 스택 상위로 이동하여 효율성, 신뢰성, 지속 가능성을 균형 있게 조율하는 글로벌 컴퓨팅 생태계의 설계자가 될 것입니다.

AI 시스템 성능 엔지니어로서 우리의 역할은 단일 노드 커널을 넘어 시스템 전체 및 시설 전체 최적화로 확장될 것입니다. 우리는 느린 올-리듀스 패턴과 같은 병목 현상을 직관적으로 포착한 후 AI 도구를 활용해 해결할 것입니다. 동시에 하드웨어와 알고리즘 혁신의 속도가 가속화될 것이므로 지속적인 학습이 필수적입니다.

결론적으로, 경쟁력을 유지하려면 AI 시스템 성능의 기초를 탄탄히 다지고, 호기심을 잃지 않으며, 새로운 하드웨어와 소프트웨어 발전을 실험하고, AI 추천을 신뢰하며, 양자 컴퓨팅 시대를 넘어 변화하는 환경에 적응할 준비를 해야 합니다. 생각해보십시오—연구의 민주화, 원클릭으로 접근 가능한 AI 슈퍼컴퓨터, 수조 개의 매개변수를 가진 모델이 보편화된 시대에, 여러분이 차세대 초지능 기술의 돌파구를 여는 주역이 될 수도 있습니다!