LLM 서빙시 Head-of-Line 블로킹 완화를 위한 SLM 기반 LLM 출력 길이 예측 및 스케줄링 기법

장성[O1], 김진우[1], 김기현[1],김홍연[2], 김영재[†1]
[1]서강대학교 컴퓨터공학과, [2]한국전자통신연구원
{jesjsjes, jinwookim, kion777, youkim}@sogang.ac.kr, {kimhy}@etri.re.kr

SLM-Assisted Output Length Prediction and Scheduling to Alleviate Head-of-Line Blocking in LLM Serving

Seong Jang[O1], Jinwoo Kim[1], Kihyun Kim[1], Hong-Yeon Kim[2], Youngjae Kim[†1]
[1]Department of Computer Science and Engineering, Sogang University, [2]ETRI, Daejeon

요 약

대규모 언어 모델(LLM) 서빙 시스템의 전체 처리량(throughput)은 스케줄링 효율성에 크게 좌우된다. 하지만 기존 QLM(queue management system for LLM serving)시스템은 과거 쿼리 통계에 기반하여 출력 길이를 예측하므로, 학습 데이터셋에 존재하지 않는 쿼리(Out-Of-Distribution, OOD queries)의 출력 길이를 정확히 예측하지 못하는 한계가 있다. 이는 긴 출력 길이의 쿼리가 우선순위를 잘못 받아 짧은 쿼리의 실행을 가로막는 Head-of-Line (HOL) 블로킹 현상을 유발하며, 결과적으로 전체 시스템 성능을 저하시키는 주요 원인이 된다. 본 논문은 이러한 한계를 극복하기 위해, 경량 언어 모델(Small Language Model, SLM)을 활용하여 LLM의 실제 출력 길이를 사전에 예측하여 이용하는 새로운 스케쥴링 기법인 S-QLM(SLM-enhanced QLM)을 제안한다. 이 기법은 예측된 출력 길이가 짧은 쿼리에게 동적으로 높은 우선순위를 할당하여 병목 현상을 완화한다. ShareGPT 데이터셋을 사용하여 실험한 결과, S-QLM은 기존 시스템보다 TPS(tokens/sec)가 최대 11.5% 증가하였고 OOD 쿼리의 비율이 높아질수록 성능 향상이 높아졌다.

1. 서론

OpenAI GPT-4와 Google Gemini를 비롯한 대규모 언어 모델(Large Language Model, LLM)은 챗봇, 코드 자동생성, 콘텐츠 제작, 데이터 분석 등 다양한 응용 분야에서 이용되고 있다. 이러한 확산은 동시 다발적인 추론 요청의 증가를 초래하였으며, 이를 효율적으로 처리하고 추론 서비스를 제공하는 것은 시스템의 자원 활용 효율과 운영 비용을 결정하는 핵심 요인으로 부상하였다.

LLM 추론 서비스 환경에서는 다수의 사용자가 제출한 요청이 공용 대기열(queue)에 적재된 후 순차적으로 처리된다. 각 요청은 모델이 생성해야 하는 응답의 길이, 즉 출력 토큰 수가 상이하며, 이로 인해 요청별 총 추론 시간에 큰 편차가 발생한다. 이러한 길이 편차는 스케줄링 과정에서 Head-of-Line(HOL) 블로킹을 유발할 수 있다. HOL 블로킹은 출력 길이가 긴 요청이 대기열의 선두에 위치함으로써 뒤따르는 상대적으로 짧은 요청들의 처리를 지연시키는 현상으로, 결과적으로 전체 응답 지연(latency)을 증가시키고 평균 응답 시간을 악화시키며 시스템의 처리량(throughput)을 저하시킨다.

이러한 HOL 블로킹 문제는 기존의 단순 선입선출 스케줄링으로는 효과적으로 완화하기 어렵다. 이를위해 QLM(Queue Management System for LLM Serving) 스케줄링 시스템이 제안되었다 [1]. QLM은 과거 쿼리들의 입/출력 길이 통계를 학습한 통계적 모델을 사용하여, SLO가 비슷한 요청들 사이에서 LLM 추론 시간이 짧은 쿼리에 높은 우선순위를 부여한다. QLM의 통계 기반 예측 방식은 학습된 분포 내요청에서는 안정적으로 동작하지만, 분포 외(Out-of-Distribution, OOD) 입력에 대해서는 출력 길이 예측의 일반화 성능이 낮다. 이러한 예측 부정확성은 우선순위 결정의 신뢰도를 저하시켜 HOL 블로킹을 발생시켜 결과적으로 전체 서빙 효율을 감소시킨다.

본 논문에서는 이러한 한계를 해결하기 위해 S-QLM(SLM-enhanced QLM)을 제안한다. S-QLM은 QLM의 통계 기반 예측기와 경량 언어 모델(Small Language Model, SLM)을 결합하여, 학습 데이터에 없는 쿼리에서도 높은 출력 길이 예측 정확도를 유지하는 스케줄링 프레임워크이다. 시스템은 다음 절차로 동작한다. (1) 기존 QLM이 학습 데이터에 없는 쿼리를 제외하고 출력 길이를 예측한 후 출력 길이가 짧은 순서대로 우선순위를 부여한다. (2) SLM이 OOD 쿼리의 출력 길이를 예측한 후, 예측값에 따라 우선순위를 산정하고 스케줄링에 반영한다.

S-QLM은 vLLM에 구현이 되었으며 S-QLM의 효과는 ShareGPT 기반 워크로드를 이용한 평가를 통해 검증하였다. OOD 쿼리 비율을 변화시키며 비교한 결과, S-QLM은 기존 QLM 대비 전체 토큰 처리량(Tokens/sec)을 최대 11.5% 향상시켰다.

2. 배경지식 및 연구동기

2.1 LLM 추론

LLM의 추론 과정은 이전에 생성된 토큰을 입력으로 사용해 다음 토큰을 생성하는 자가회귀적(autoregressive) 방식으로 동작한다. 이 과정은 일반적으로 Prefill 단계와 Decode 단계로 구분된다. Prefill 단계에서는 입력 쿼리 전체를 한 번에 처리하여 첫

*본 연구는 2025년 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원(No. 2022-0-00498, 거대 학습 모델 초고속 처리를 위한 고효율 AI 컴퓨팅 SW 핵심 기술개발) 및 SW중심대학사업(2024-0-00043)의 지원을 받아 수행되었음.
† Corresponding author

번째 출력 토큰을 생성한다. 이후 Decode 단계에서는 이전에 생성된 토큰들을 반복적으로 입력으로 사용하며, 다음 토큰을 순차적으로 하나씩 생성한다.

2.2 Continuous Batching과 FCFS 스케줄링의 한계

전통적인 LLM 서빙 시스템은 Static Batching 방식을 사용하여 여러 요청을 고정된 크기의 배치로 묶어 처리한다 [2]. 이 방식은 생성 길이가 다른 여러 쿼리들이 동일한 배치로 묶여 prefill과 decode 과정을 함께 처리하게 된다. 이때 배치 전체의 종료는 가장 긴 요청 기준으로 결정되므로, 상대적으로 짧은 요청들은 이미 완료되었음에도 배치가 끝날 때까지 대기해야 한다. 이로 인해 이미 완료된 요청이 점유하던 GPU 연산 자원은 유휴 상태로 남게 되어 자원 활용률이 저하되는 문제가 발생한다.

이러한 문제를 해결하기 위해 Continuous Batching 기법이 등장했다. Continuous Batching은 Static Batching의 한계를 다음과 같이 극복한다.배치 레벨이 아닌 토큰 레벨(iteration-level)에서 스케줄링을 수행한다. 매 토큰 생성 단계마다 완료된 요청이 있는지 확인한 후, 완료된 요청을 배치에서 즉시 제거하고 큐에서 대기 중인 새로운 요청을 바로 배치에 넣는다. 이때 큐에 도착한 순서대로(First-Come, First-Served; FCFS) 배치에 요청을 추가한다. 결과적으로 GPU는 유휴 시간 없이 지속적으로 작업을 수행하며, Static Batching 대비 높은 처리량 향상으로 이어진다 [2]. 하지만 FCFS 정책은 단순히 큐에 도착한 순서대로 요청을 처리하기 때문에 HOL 블로킹 문제가 발생할 수 있다. HOL 블로킹은 시스템의 평균 응답 시간을 증가시키고 처리량을 저하시키는 주요 원인이 된다. 따라서 이를 해결하고 처리량을 높이기 위해서는 단순한 FCFS 정책을 넘어선 스케줄링 전략이 필요하다.

2.3 통계 기반 출력 길이 예측 스케줄링 및 한계

최근 연구들은 HOL 블로킹을 완화하기 위해 여러 접근법을 제안했다 [1, 3]. 대표적으로 QLM [1]은 기존 Continuous Batching의 FCFS 방식에서 더 나아가 대기 큐에서 LLM 추론 요청들의 출력 길이(L_{out})를 예측하여, HOL 블로킹을 예방하기 위해서 예상 추론 시간이 짧은 요청을 먼저 처리한다.

이러한 접근법의 핵심은 요청의 출력 토큰 길이를 정확하게 예측하는 것이다. 그러나 L_{out}은 요청이 실제로 수행되기 전까지는 알 수 없으므로, QLM은 과거 요청들의 입 · 출력 길이 분포를 학습한 통계적 모델(statistical model)을 이용해 L_{out}를 예측한다. 하지만 이러한 통계 모델은 근본적인 한계가 존재한다.

통계 모델은 학습 분포에서 벗어난 OOD 쿼리에 대해 예측 오차와 불확실성이 크게 증가한다 [4]. 그 결과, 잘못 예측된 L_{out}으로 인해 실제로는 긴 출력을 생성하는 요청이 높은 우선순위를 부여받을 수 있고, 이는 HOL 블로킹을 야기한다.

3. S-QLM 설계 및 구현

3.1 SLM기반 실시간 LLM 출력 길이 예측

본 논문에서는 통계 모델 기반 출력 길이 예측 오류로 인한 HOL 블로킹 문제를 완화하기 위한 시스템인 S-QLM(SLM-enhanced QLM)을 제안한다. S-QLM은 기존 QLM의 통계 기반 예측 방식을 확장하여 분포 외 OOD 쿼리에 대해서는 SLM을 활용 함으로써 예측 정확도를 향상시킨다.

SLM은 LLM보다 상대적으로 적은 파라미터 수를 가지며, 축소된 모델 구조로 빠른 추론 속도를 제공하는 경량화된 언어 모델이다. 이러한 특성 덕분에, 각 요청이 큐에 도착하는 시점에 SLM이 요청의 답변을 빠르게 출력하여 예상 출력 길이를 신속하게 추론할 수 있다. 만약 SLM이 사전 데이터 없이도 LLM의 실제 출력 길이를 합리적인 수준으로 예측할 수 있다면, 이는 학습 데이터에 없는 쿼리의 출력 길이 예측값을 실시간으로 제공할 수 있다는 이점을 갖는다.

3.2 동작 과정

S-QLM의 동작 과정은 다음과 같이 4가지 단계로 동작한다.

- **(1) 예측 단계:** 사용자 쿼리가 입력되면, QLM의 통계 기반 모델을 이용하여 각 요청의 예상 출력 길이를 추정한다. 이때 사용자 쿼리가 학습 분포 내에 있는지 여부에 따라서 OOD 쿼리 여부를 판단한다.
- **(2) 우선순위 상정단계:** 이후 OOD 쿼리를 제외한 쿼리들의 예상 출력 길이가 짧은 순서대로 스케줄링 우선순위를 부여한다.
- **(3) OOD 쿼리 보정 단계:** OOD 쿼리의 우선순위는 SLM이 해당 쿼리의 답변을 직접 생성하여 출력 길이를 예측한다.
- **(4) 우선순위 갱신 단계:** 예측값을 기반으로 OOD 쿼리의 우선순위를 산정하고, 이를 스케줄링에 반영한다.

4. 실험 평가

4.1 실험 환경

S-QLM은 vLLM 프레임워크를 기반으로 구현되었다. S-QLM의 성능을 QLM 대비 평가하기 위해 ShareGPT 데이터셋과 함께 SLM으로 Llama-3.2-1B-Instruct 모델, LLM으로 Mistral-7B-Instruct-v0.3 모델을 vLLM을 통해 서빙했고, Temperature는 모두 1.0, 최대 입/출력 토큰 길이의 합은 2,024, Batch Size(모델에서 동시에 처리할 수 있는 쿼리의 개수)는 256으로 설정했다. [표 2] 입력 쿼리 1,000개 중 OOD 쿼리 비율(OOD Ratio)과 를 변화시켰으며, 처리량인 TPS(Tokens Per Second, 입/출력 토큰의 길이의 합을 LLM 추론 시간으로 나눈 값)를 측정했다. 실험에서는 HOL 블로킹 효과를 명확히 관찰하기 위해, 긴 출력 길이를 갖는 쿼리가 배치의 선두에 오도록 Baseline(QLM) 큐를 구성하였다.

표 1: Server Experiment Environment

Component	Specification
CPU	Intel Xeon Gold 6548Y+ (2 units, 32cores)
GPU	NVIDIA L40S (2 units, 48 GB VRAM each)
Memory	1 TB RAM
vLLM Version	v0.10.1
CUDA Version	12.8

표 2: Experimental Environment

Component	Specification
SLM	Llama-3.2-1B-Instruct
LLM	Mistral-7B-Instruct-v0.3
Model Temperature	1.0
Dataset	ShareGPT_V3 (unfiltered cleaned split, 1000 samples)

4.2 실험 결과 분석

그림 1은 쿼리 비율 변화에 따른 기존 QLM과 S-QLM의 처리율(TPS, Tokens/sec)을 비교한 결과를 나타낸다. 실험 결과, 모든 쿼리가 학습 데이터 분포 내에 있는 OOD Ratio 0.0 환경에서 S-QLM이 모든 쿼리에 대해 길이 예측을 수행하도록 설정했음에도 불구하고, 기존 QLM 대비 약 1.2% 낮은 TPS를 보였다. 이는 SLM의 예측 오차로 인한 것으로, 학습 데이터 분포 내에서는 통계 기반 QLM이 상대적으로 정확한 예측을 했기 때문이다. 그러나 이 차이는 미미하며, 실시간 예측 기능이 제공하는 안정성 대비 성능 손실은 거의 없다고 볼 수 있다.

OOD Ratio가 0.5로 증가하자, S-QLM은 기존 QLM 대비 9.7%의 TPS 향상을 보였다. 이는 절반의 쿼리가 학습 데이터 분포 밖에 존재할 때, QLM의 통계 기반 예측이 실제 출력 길이와 큰 차이를 보이며 우선순위 스케줄링의 효율이 급격히 저하되기 때문이다. 그 결과 긴 쿼리가 큐의 앞단에서 정체되는 HOL 블로킹이 발생하지만, S-QLM은 SLM을 활용해 각 쿼리의 출력 길이를 실시간으로 예측함으로써 HOL 블로킹을 효과적으로 완화하고 처리율 저하를 방지했다.

OOD Ratio가 0.8로 높아진 환경에서는 S-QLM의 성능 향상이 더욱 두드러져, QLM 대비 11.5% 높은 TPS를 기록했다. 이때 대부분의 쿼리가 학습 데이터 분포에서 벗어나므로, 통계 기반 QLM의 예측은 사실상 무력화되어 스케줄링 비효율이 극대화된다. 반면, S-QLM은 입력 문맥을 직접 해석하여 출력 길이를 동적으로 예측할 수 있어, 다양한 쿼리 분포에서도 안정적인 스케줄링과 처리율을 유지할 수 있었다.

종합적으로, S-QLM은 OOD 환경에서 기존 QLM보다 훨씬 안정적인 성능을 보이며, 통계 기반 접근 방식의 한계를 극복했다. 이는 S-QLM이 입력 쿼리의 의미적 특성을 반영한 실시간 예측을 수행함으로써, HOL 블로킹을 완화하고 LLM 서빙 시스템의 전반적인 효율성을 향상시켰기 때문이다.

S-QLM 오버헤드: SLM을 통한 출력 길이 예측 과정에서 추가적인 추론 오버헤드가 발생하는 한계가 존재하였으며, 1,000개 요청 처리 시 OOD 비율이 0.5와 0.8인 경우 각각 약 27.8초와 41.4초의 추가 시간이 소요되어 요청당 평균 28–41ms의 추가 지연이 발생하였다. 향후 연구에서는 서버 부하가 높아 큐 대기 시간이 길어지는 상황[5]을 활용해 대기 시간 동안 SLM 추론을 비동기적으로 수행하거나, 더 작은 파라미터의 경량 모델을 적용하여 추론 오버헤드를 최소화하는 방향으로 확장할 예정이다.

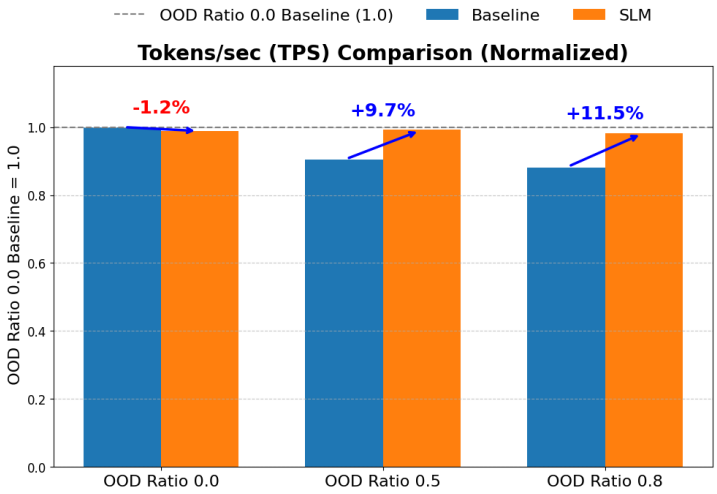

그림 1: Batch Size 256 일 때 QLM과 S-QLM의 처리율 비교. 레전드에서 Baseline은 QLM이며 SLM은 S-QLM을 의미한다.

5. 결론 및 향후 연구

기존 QLM은 통계 기반의 출력 길이 예측 모델을 사용하기 때문에 학습 데이터 분포에서 벗어난 OOD(Out-of-Distribution) 쿼리가 유입될 경우 예측 오차가 커지고, 이로 인해 Head-of-Line(HOL) 블로킹이 발생하여 전체 처리량(TPS)을 저하시키는 문제가 있었다. 이를 해결하기 위해 본 연구에서는 OOD 쿼리에 한해 경량 언어 모델(SLM)을 활용해 출력 길이를 실시간으로 예측하고 이를 스케줄링에 반영함으로써 HOL 블로킹을 완화하는 S-QLM을 제안하였다. 실험 결과, S-QLM은 기존 QLM 대비 TPS가 최대 11.5% 향상되었으며, 이는 SLM 기반의 실시간 예측이 기존 통계적 접근의 한계를 보완해 보다 효율적인 스케줄링을 가능하게 했기 때문이다.

참고 문헌

[1] A. Patke, D. Reddy, S. Jha, H. Qiu, C. Pinto, C. Narayanaswami, Z. Kalbarczyk, and R. Iyer, "Queue management for slo-oriented large language model serving," in *ACM Symposium on Cloud Computing*, SoCC '24, p. 18–35, 2024.

[2] G.-I. Yu, J. S. Jeong, G.-W. Kim, S. Kim, and B.-G. Chun, "Orca: A distributed serving system for transformer-based generative models," in *USENIX Symposium on Operating Systems Design and Implementation (OSDI 22)*, pp. 521–538, 2022.

[3] Y. Fu, S. Zhu, R. Su, A. Qiao, I. Stoica, and H. Zhang, "Efficient llm scheduling by learning to rank," in *Advances in Neural Information Processing Systems*, vol. 37, pp. 59006–59029, 2024.

[4] V. Flovik, "Quantifying distribution shifts and uncertainties for enhanced model robustness in machine learning applications," *arXiv preprint arXiv:2405.01978*, 2024.

[5] H. Lee, K. Kim, J. Kim, J. So, M.-H. Cha, H.-Y. Kim, J. J. Kim, and Y. Kim, "Disk-based shared kv cache management for fast inference in multi-instance llm rag systems," in *IEEE International Conference on Cloud Computing (CLOUD)*, pp. 199–209, 2025.