Imperial College London

MEng Individual Project

Department of Computing

Imperial College of Science, Technology and Medicine

A New Scalable Runtime for LLM Inference

Author:
Hamish McCreanor

Supervisor:
Peter Pietzuch

January 6, 2025

Submitted in partial fulfillment of the requirements for the MEng Computing of Imperial College London

Contents

Chapter 1

Introduction

As large language models (LLMs) are found useful for ever-wider classes of applications, a trend has arisen focusing on the low-cost, local deployment of these systems. While the training of LLMs like LLaMA, BERT and OpenAI's GPTs typically requires months of training and is prohibitive for all but the most well-funded of organisations, performing inference on these models locally is comparatively more feasible. This enables developers to create services with tighter LLM integrations - instead of calling a black-box API provided by an LLM provider, they can instead run a local version of the LLM, tuning the inference runtime to more appropriately match the context in which it is called.

As a result, there is currently a vast body of research aiming to improve existing inference systems. The aim of this is to improve LLM inference performance along various axes. These include running on lower-powered hardware; running with improved throughput and running at greater energy efficiencies. These optimisations focus on specific elements of the inference pipeline, particularly improving KV cache usage and kernel fusion. To build systems containing these optimisations, developers frequently turn to high level languages like Python in order to quickly develop the infrastructure surrounding their new technique. Developing this way limits the ability of the system to exploit memory-access patterns and application parallelism (especially in a language like Python, with its global interpreter lock) and incurs unnecessary overhead.

This project aims to build on the existing llama.cpp inference server (see 2.2.5) to deliver a system that improves the dispatch of compute kernels by better parallelising the inference pipeline.

Chapter 2

Background

2.1 Preliminaries

2.1.1 LLM Architecture

Transformer Architecture

2.1.2 LLM Inference

KV Cache

Parallelism

Request Batching

2.2 Related Work

2.2.1 vLLM

PagedAttention

2.2.2 Triton

2.2.3 SGLang

2.2.4 Triton

2.2.5 llama.cpp

Chapter 3

Project Plan

Chapter 4

Evaluation Plan

4.1 Functional Requirements

4.2 Performance Metrics

Chapter 5

Ethical Issues