PerfXLM : a High Performance LLM Inference Engine on RISC-V CPUs

Xianyi Zhang

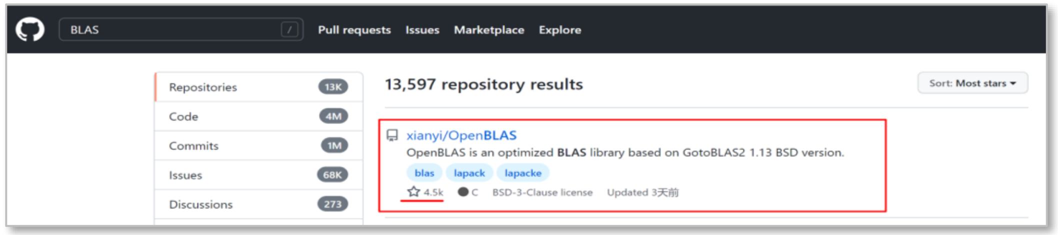
PerfXLab Technologies
xianyi@perfxlab.com

2024.06

PerfXLab

About US

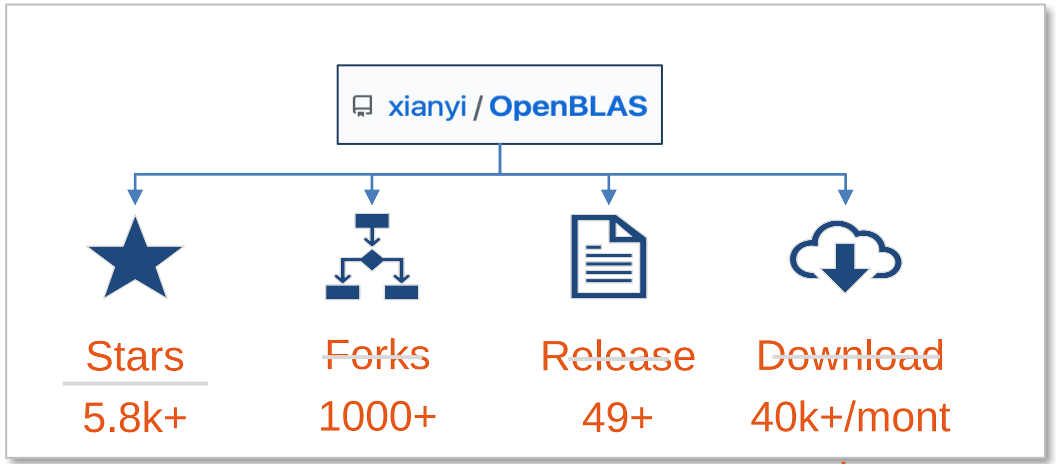
- 2016: PerfXLab Technologies founded in Beijing.
- Xianyi Zhang： CAS Ph.D., OpenBLAS creator, postdoc research at UT Austin & MIT
- Heterogeneous Computing Software Stack and Solutions, including
  - High Performance Computing Libraries
  - Heterogeneous Software Framework
  - Domain Accelerated Computing Solutions (e.g. AI Infra, EDA, Signal Processing, and etc.)


BLAS Pull requests Issues Marketplace Explore

Repositories	13K
Code	4M
Commits	1M
Issues	68K
Discussions	273

13,597 repository results Sort: Most stars

xianyi/OpenBLAS
OpenBLAS is an optimized BLAS library based on GotoBLAS2 1.13 BSD version.
blas lapack lapacke
4.5k C BSD-3-Clause license Updated 3天前



Stars
5.8k+

~~Forks~~
1000+

~~Release~~
49+

~~Download~~
40k+/mont
h

Created and Maintain OpenBLAS Since 2011

RISC-V Global Forum September 3, 2020

Optimize OpenBLAS by RISC-V "V" Vector Extension

Xianyi Zhang

2020 RISC-V Global Forum

2019 CRVA RISC-V Ecosystem

2016 CCF Prize for OpenBLAS

Khronos Group
Advisory Panel Participation Agreement

This agreement enables membership on
Advisory Panels for one or more Khronos Working Groups

PLEASE TYPE OR PRINT CLEARLY: THIS IS A LEGAL DOCUMENT
ILLEGIBLE AGREEMENTS CANNOT BE PROCESSED

Contact for processing this agreement:

NAME:	Xianyi Zhang
COMPANY:	PerfXLab Technology Co., Ltd.
EMAIL:	xianyi@perfxlab.com
PHONE:	+86-13466545921

Guidelines for completing this agreement:

1. This Advisory Panel agreement is normally executed by invited individuals, however if you wish to execute on behalf of a company/institute which is a legal entity that you are able to represent, and has been invited to participate in the Advisory Panel, you may execute this agreement on behalf of that entity and designate colleagues within that entity that will also be bound by this agreement and so able to access Advisory Panel resources.

2. Please enter your name, email address on this cover page and select the working groups to which you have been invited to advise on the signature page.

3. Select the tick box on the signature to confirm whether you are executing this agreement as an individual or on behalf of a legal entity. If you are executing this on behalf of an entity, then enter the name of the legal entity on this cover page and as the Signatory on the signature page (instead of your own name) and enter a list of Designated Colleagues in Attachment B. If the legal entity requires more than one authorized signatory, duplicate the signature page for each required signature.

4. Mail a copy **OR** email a PDF of the executed agreement to the contact address below. If mailed, send two signed copies and one completed copy will be returned for your records

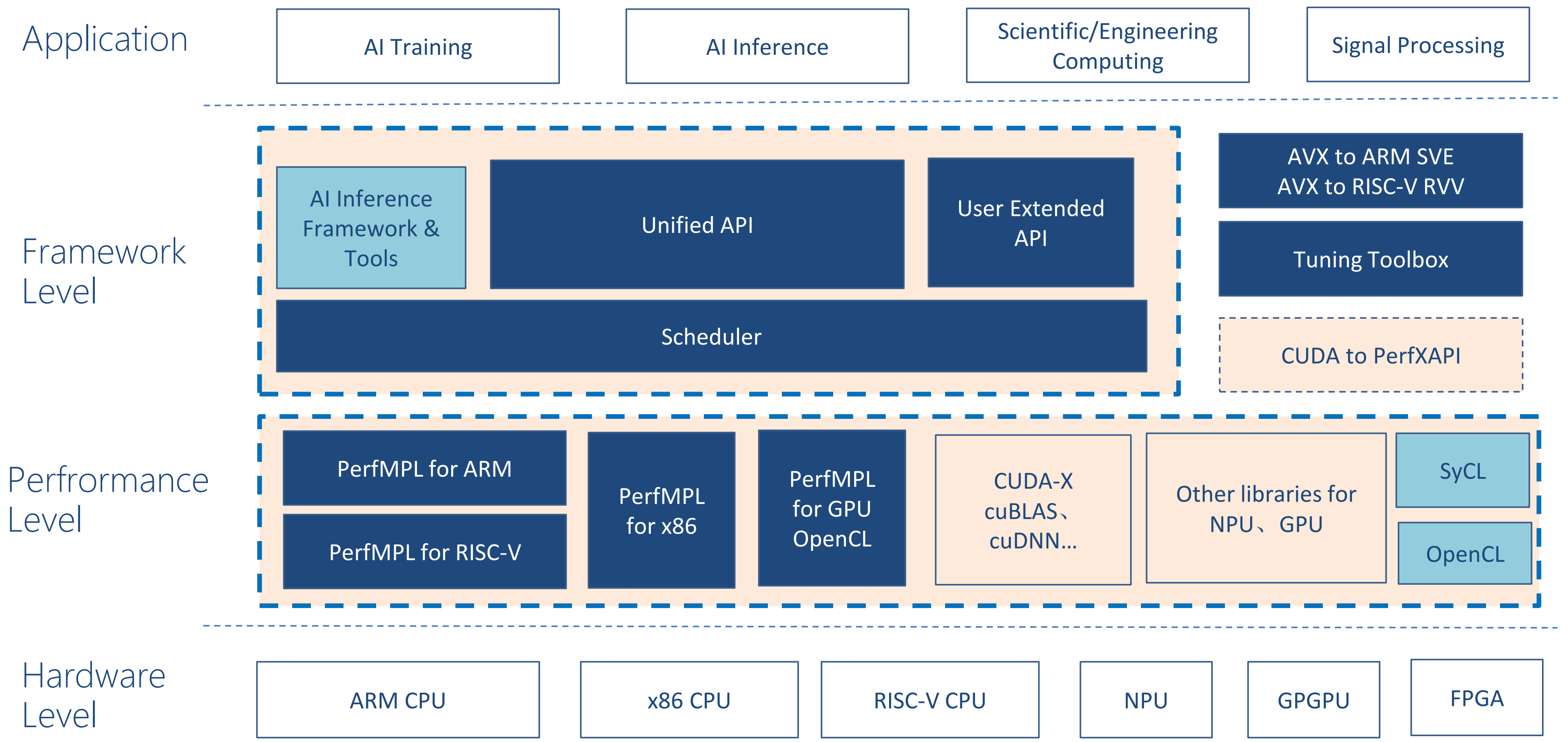
Contact Details:
Khronos Group Member Services
Khronos Group Inc, 9450 SW Gemini Drive #45043, Beaverton, OR 970088-6018, USA
memberservices@khronos.org
Voice mail: +1 (415) 869-8627

Your **Advisory Privileges** will commence when Khronos has acknowledged receipt of the executed agreement.

Khronos Advisory Panel Participation Agreement V13 Dec21 1

2023 OpenCL Advisory Panel

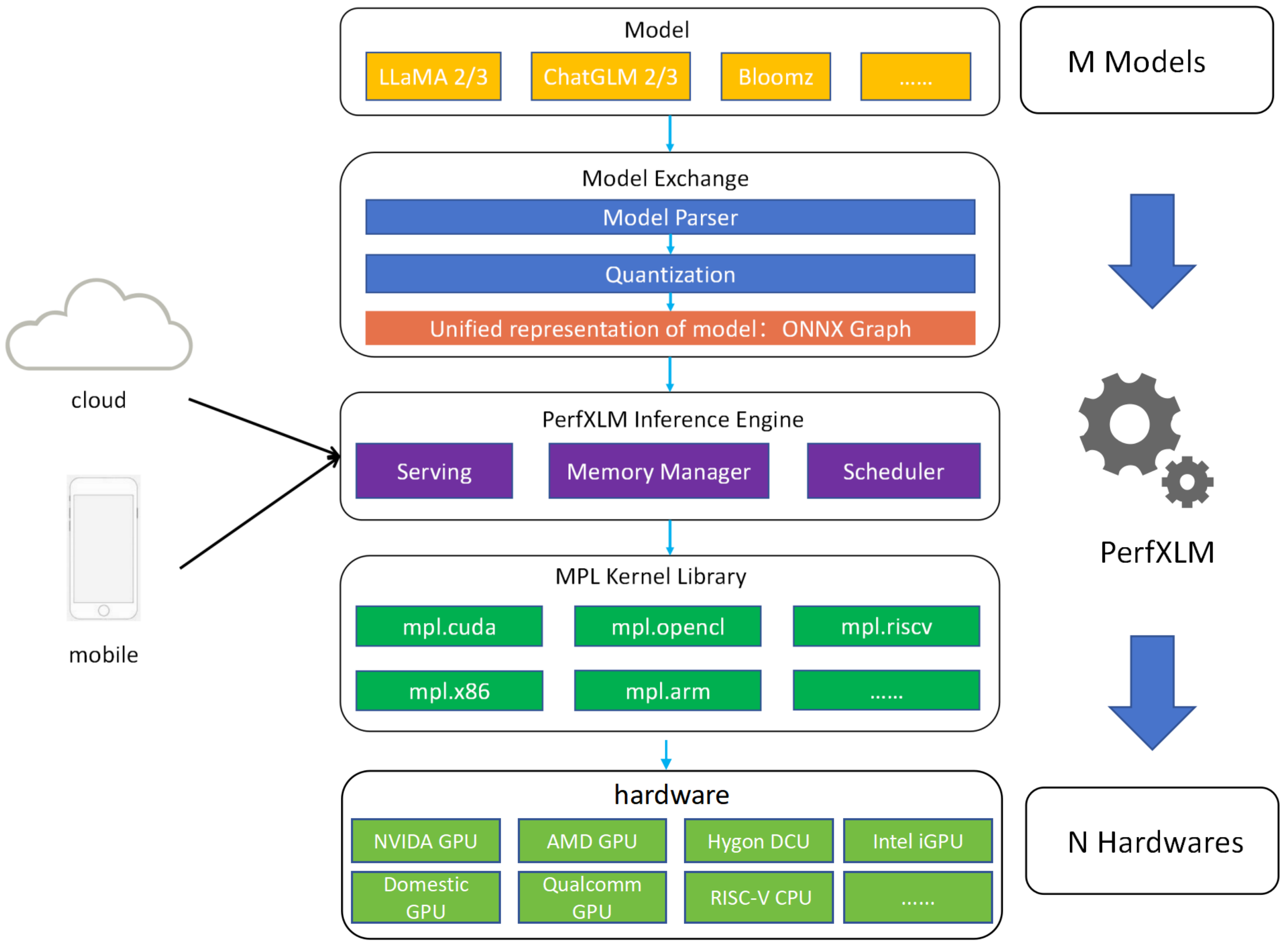
PerfXAPI Heterogeneous Computing Framework



PerfXLM Architecture

Features of PerfXLM Architecture：

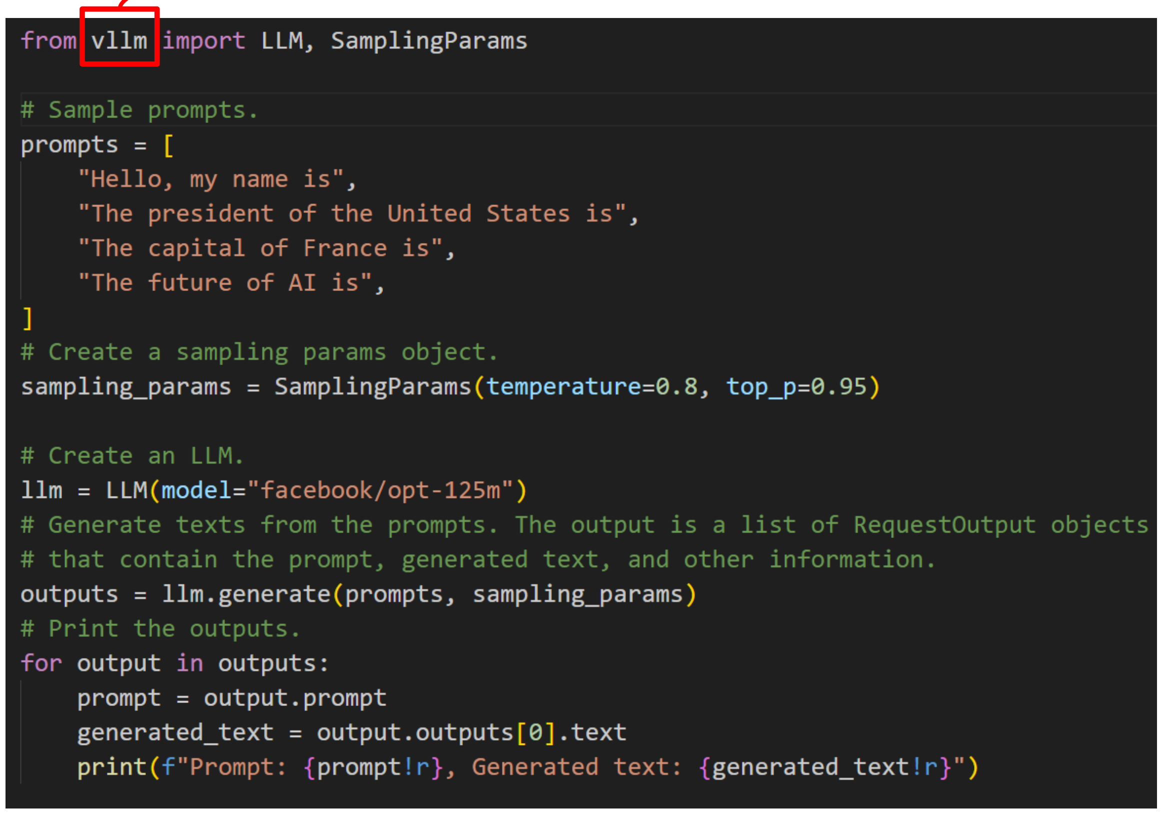
- **Cloud-edge integration：** supporting both cloud and edge inference for large models, making them suitable for various application scenarios.
- **Support for various heterogeneous platforms：** supporting NVIDIA GPU, Hygon DCU, Qualcomm Adreno GPU, Intel iGPU, and other hardware devices.
- **Customized high-performance kernels:** it tailored for hardware characteristics and data properties of large model inference.



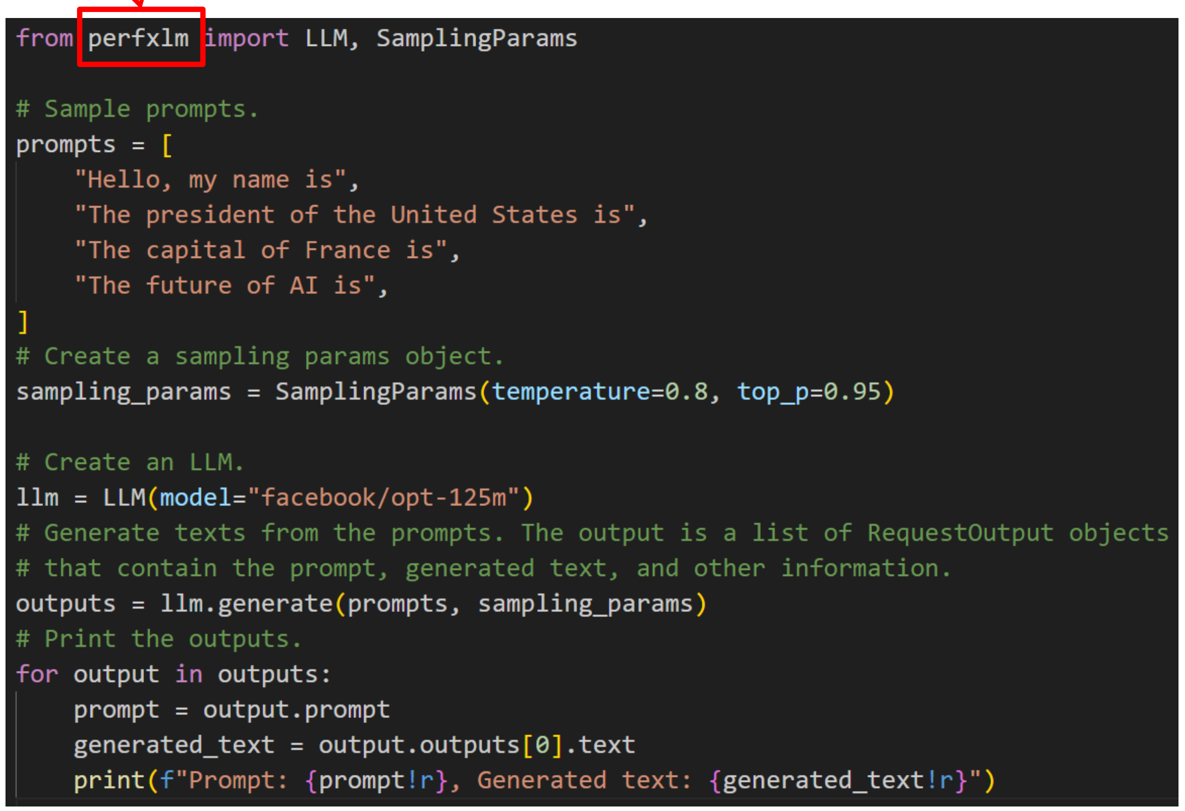
PerfXLM Architecture (User interface)

PerfXLab

PerfXLM User interface, vllm compatible



```python
from vllm import LLM, SamplingParams

# Sample prompts.
prompts = [
    "Hello, my name is",
    "The president of the United States is",
    "The capital of France is",
    "The future of AI is",
]
# Create a sampling params object.
sampling_params = SamplingParams(temperature=0.8, top_p=0.95)

# Create an LLM.
llm = LLM(model="facebook/opt-125m")
# Generate texts from the prompts. The output is a list of RequestOutput objects
# that contain the prompt, generated text, and other information.
outputs = llm.generate(prompts, sampling_params)
# Print the outputs.
for output in outputs:
    prompt = output.prompt
    generated_text = output.outputs[0].text
    print(f"Prompt: {prompt!r}, Generated text: {generated_text!r}")
```

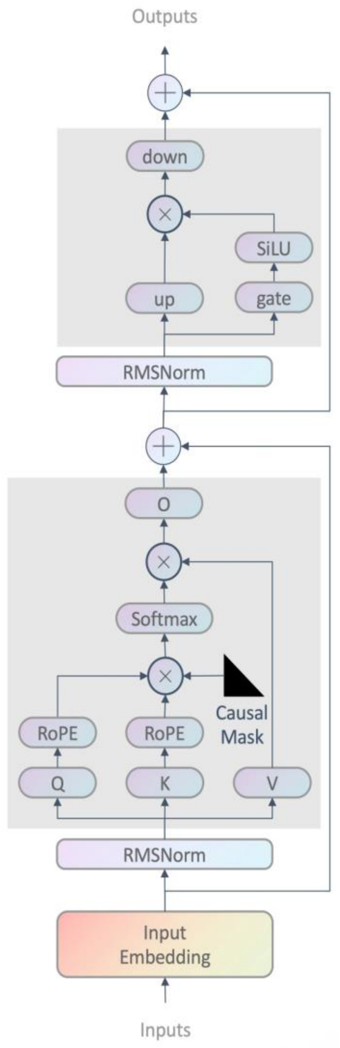


```python
from perfxlm import LLM, SamplingParams

# Sample prompts.
prompts = [
    "Hello, my name is",
    "The president of the United States is",
    "The capital of France is",
    "The future of AI is",
]
# Create a sampling params object.
sampling_params = SamplingParams(temperature=0.8, top_p=0.95)

# Create an LLM.
llm = LLM(model="facebook/opt-125m")
# Generate texts from the prompts. The output is a list of RequestOutput objects
# that contain the prompt, generated text, and other information.
outputs = llm.generate(prompts, sampling_params)
# Print the outputs.
for output in outputs:
    prompt = output.prompt
    generated_text = output.outputs[0].text
    print(f"Prompt: {prompt!r}, Generated text: {generated_text!r}")
```

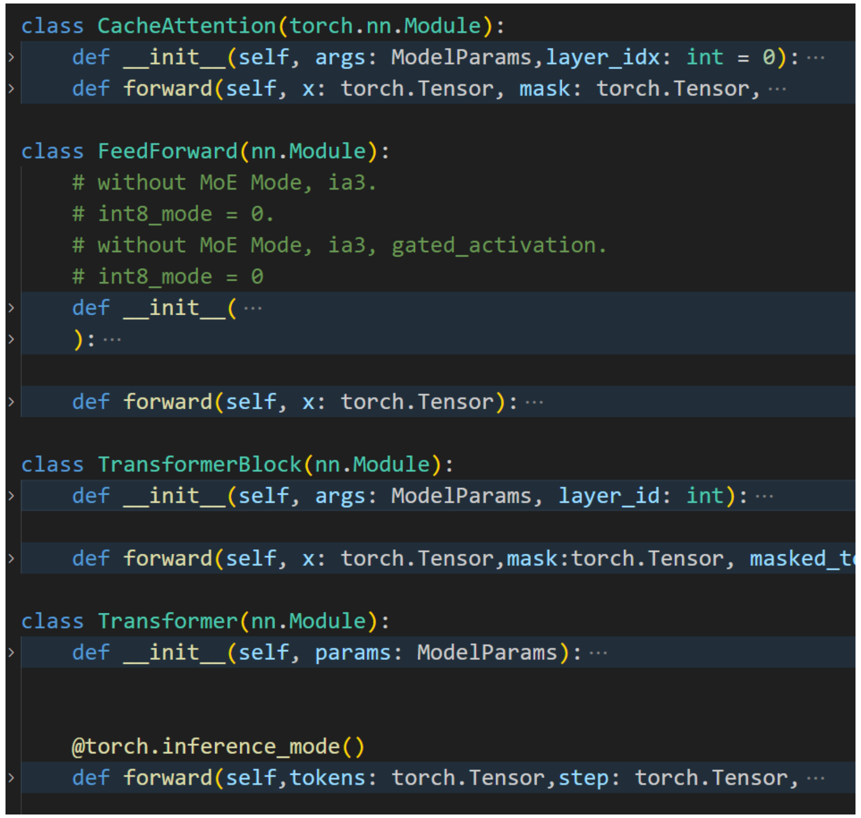
All the user needs to do is change import vllm to import perfxlm.

PerfXLM Architecture (Create Model)

PerfXLM Usage: Step 1 Create Model in Python



LLaMA Model

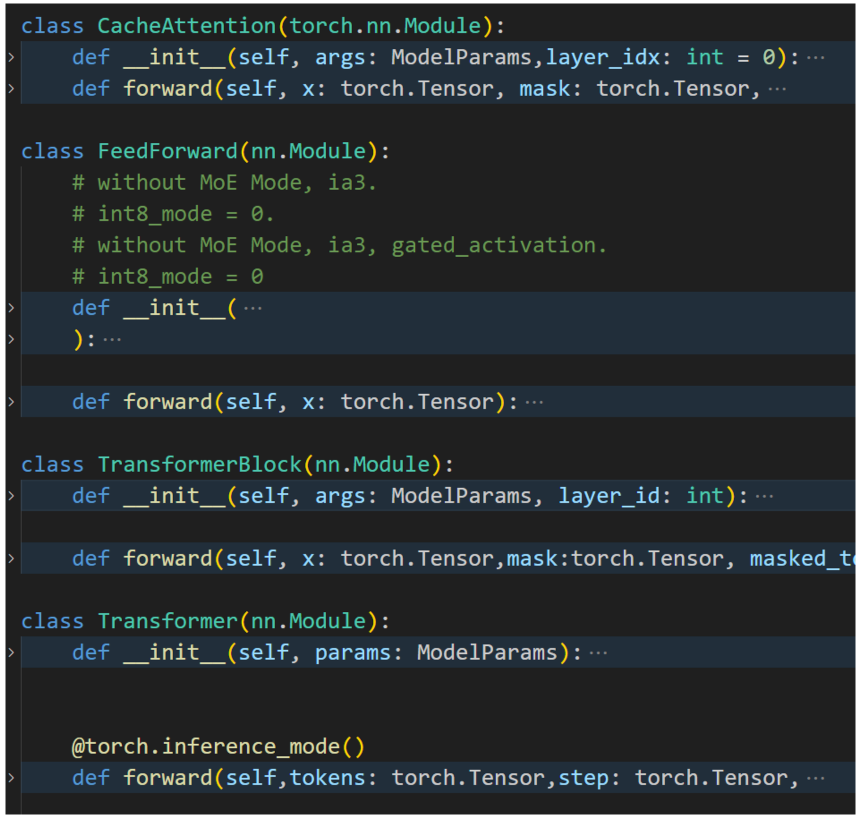


```
class CacheAttention(torch.nn.Module):
    def __init__(self, args: ModelParams,layer_idx: int = 0):···
    def forward(self, x: torch.Tensor, mask: torch.Tensor,···

class FeedForward(nn.Module):
    # without MoE Mode, ia3.
    # int8_mode = 0.
    # without MoE Mode, ia3, gated_activation.
    # int8_mode = 0
    def __init__(···
    ):···

    def forward(self, x: torch.Tensor):···

class TransformerBlock(nn.Module):
    def __init__(self, args: ModelParams, layer_id: int):···

    def forward(self, x: torch.Tensor,mask:torch.Tensor, masked_t

class Transformer(nn.Module):
    def __init__(self, params: ModelParams):···


    @torch.inference_mode()
    def forward(self,tokens: torch.Tensor,step: torch.Tensor,···
```

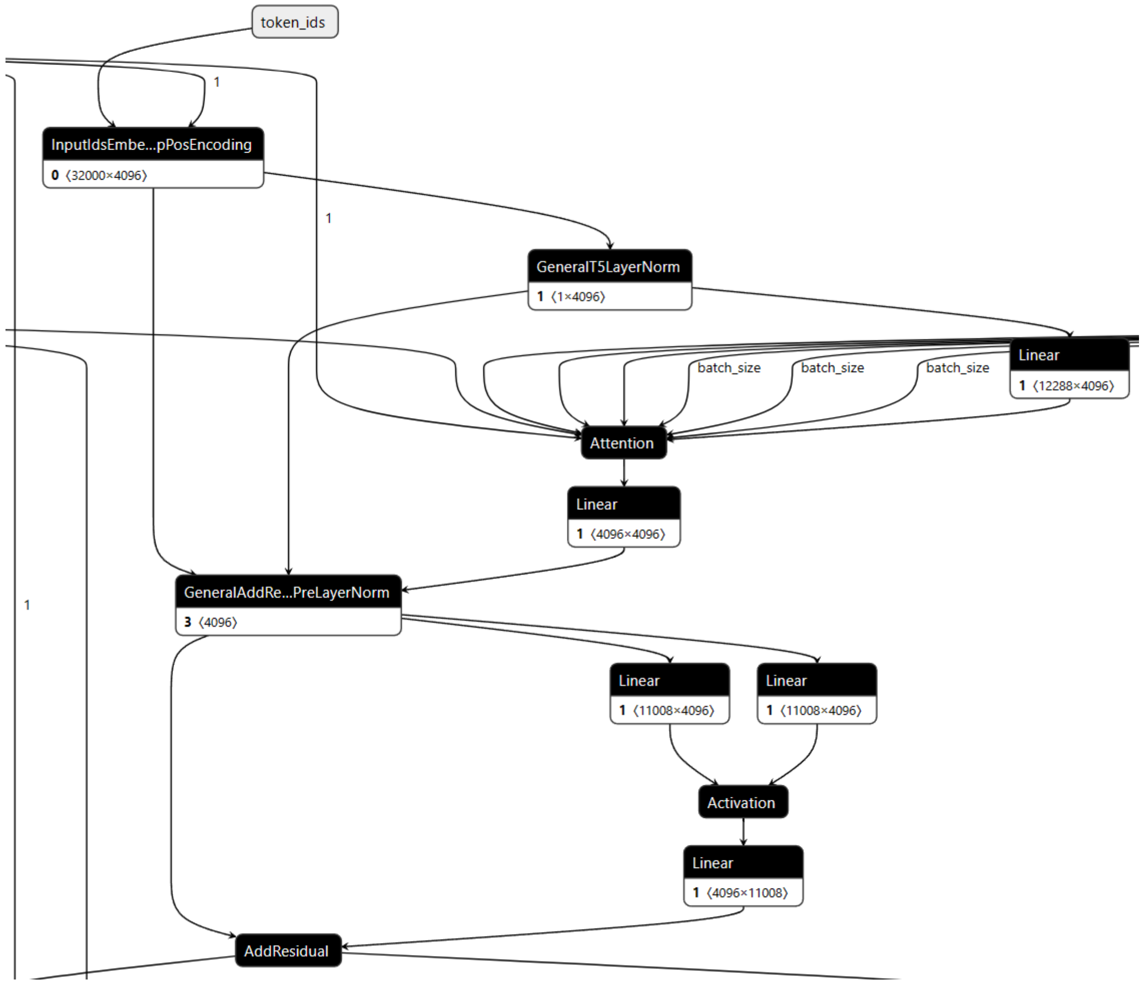
Pure Python Code and torch style.

PerfXLM Architecture (Model Parser)

Parse Python code, export ONNX graph, op fusion



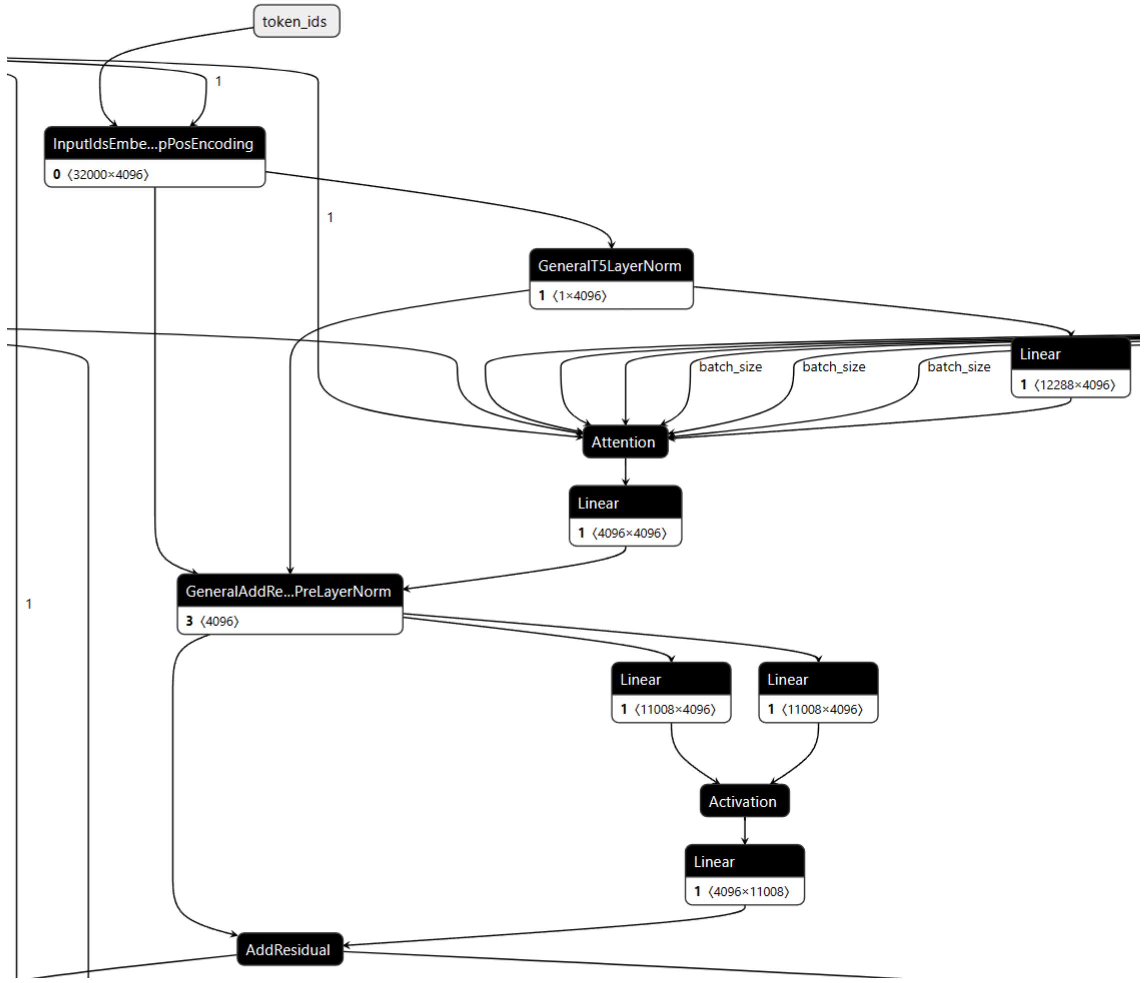
```
class CacheAttention(torch.nn.Module):
    def __init__(self, args: ModelParams,layer_idx: int = 0):···
    def forward(self, x: torch.Tensor, mask: torch.Tensor,···

class FeedForward(nn.Module):
    # without MoE Mode, ia3.
    # int8_mode = 0.
    # without MoE Mode, ia3, gated_activation.
    # int8_mode = 0
    def __init__(···
    ):···

    def forward(self, x: torch.Tensor):···

class TransformerBlock(nn.Module):
    def __init__(self, args: ModelParams, layer_id: int):···

    def forward(self, x: torch.Tensor,mask:torch.Tensor, masked_t

class Transformer(nn.Module):
    def __init__(self, params: ModelParams):···


    @torch.inference_mode()
    def forward(self,tokens: torch.Tensor,step: torch.Tensor,···
```

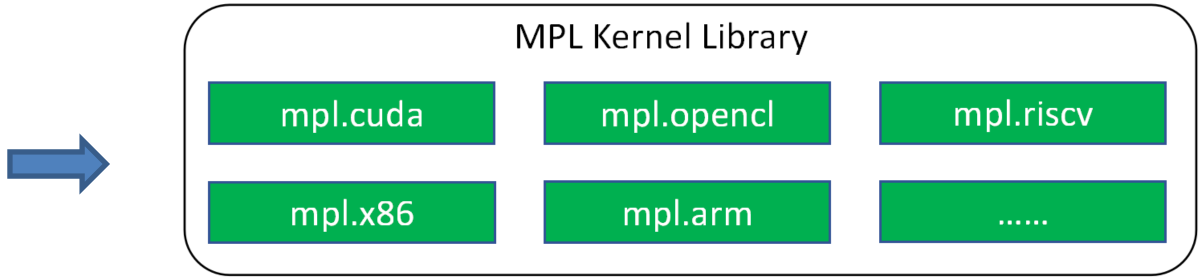


For the llama model, only 7 fused OPs are left in the final fusion, which is high performance and easy to maintain.

PerfXLM Architecture (Engine)

Parses the op-fused onnx graph and calls the corresponding high-performance operator.





PerfXLM Optimization Methods

Low-precision quantization

- Quantization of weights
  - AWQ-int4, A16
- Quantization of KV cache
  - int8 kv cache
- For embedded
  - Int4
- For server
  - Fp8? Int8?
- ……

Operator fusion

- Fused add bias and Layernorm
- Fused QKV GEMM
- Fused Decoder Attention
- Flash attention
- ……

Core Kernel optimization

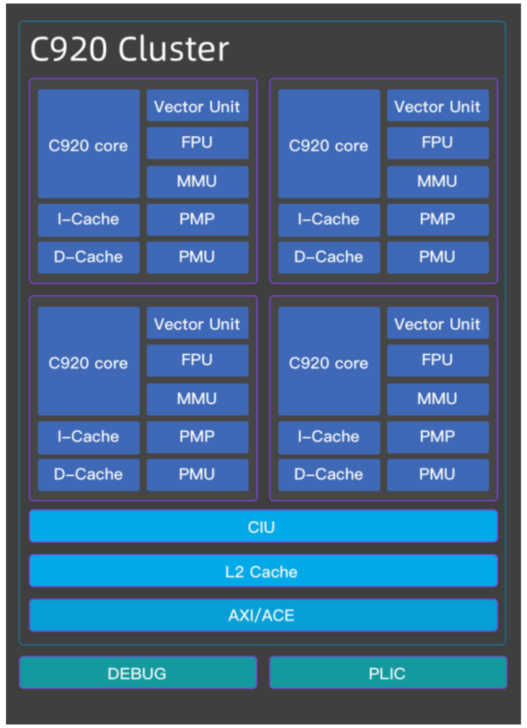
- GEMM optimization
  - Tiling to reuse data
  - Vectorized memory access
  - unroll loops
- Tall-skin matrix
- Attention kernel optimization
- ……

PerfXLM on RISC-V

RISC-V CPUs Testbed

- SG2042 CPU
  - Xuantie C920 2.0GHz
    - RVV 0.7.1, 128-bit
    - L1-D 64KB
  - 4 cores/cluster
    - Shared L2 1MB
  - 64 cores (16 cluster)
    - Shared L3 64MB

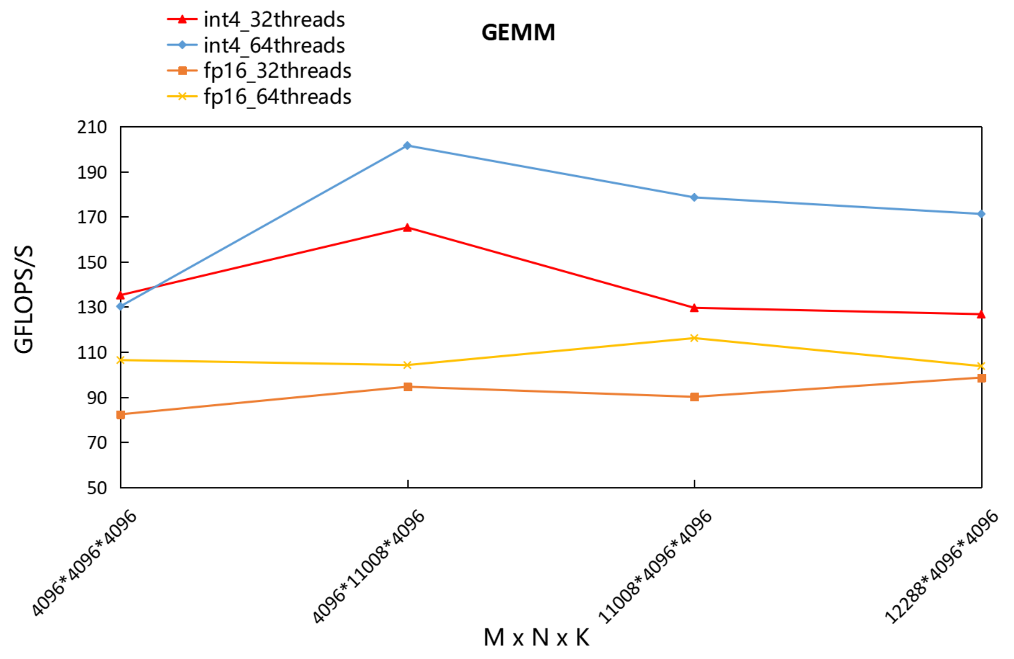
- **4x** DDR4 memory controllers



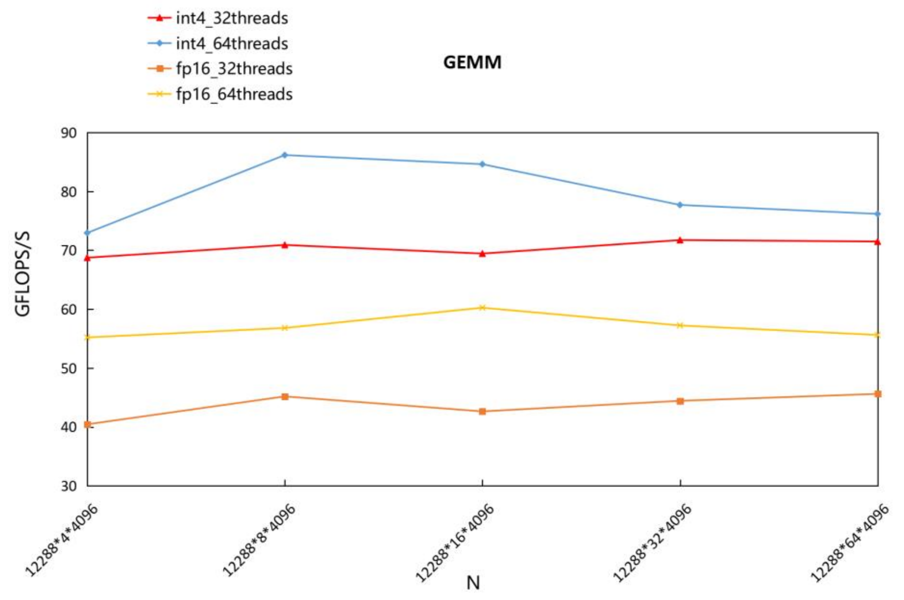
PerfXLM on RISC-V

GEMM optimization for prefill stage:

- Single batch: tall-and-skinny matrix
- Vector instruction optimization
- GEMM blocking
- Multi-threading



GEMM Performance (Big and square matrices)



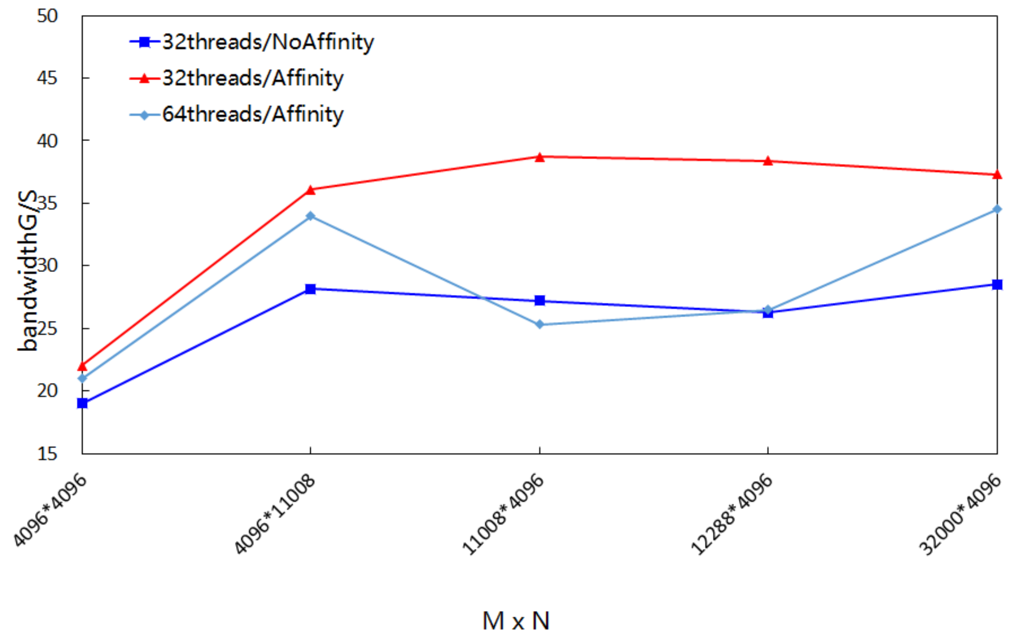
GEMM Performance (tall-and-skinny matrices)

PerfXLM on RISC-V

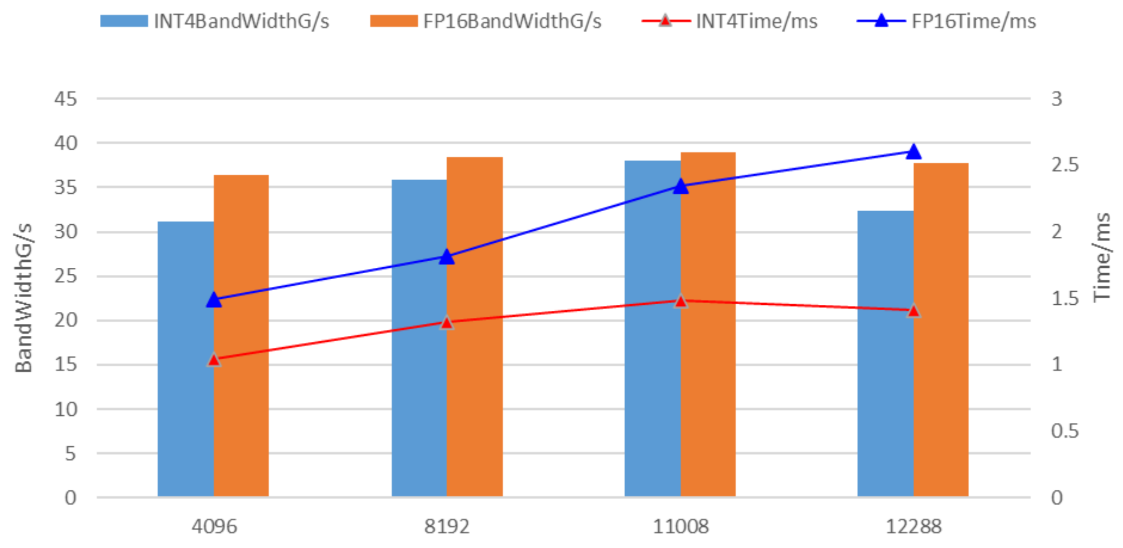
PerfXLab

GEMV optimization for decoding stage:

- Vector instruction optimization
- Multi-threading and affinity



GEMV performance under different number of threads and affinity.



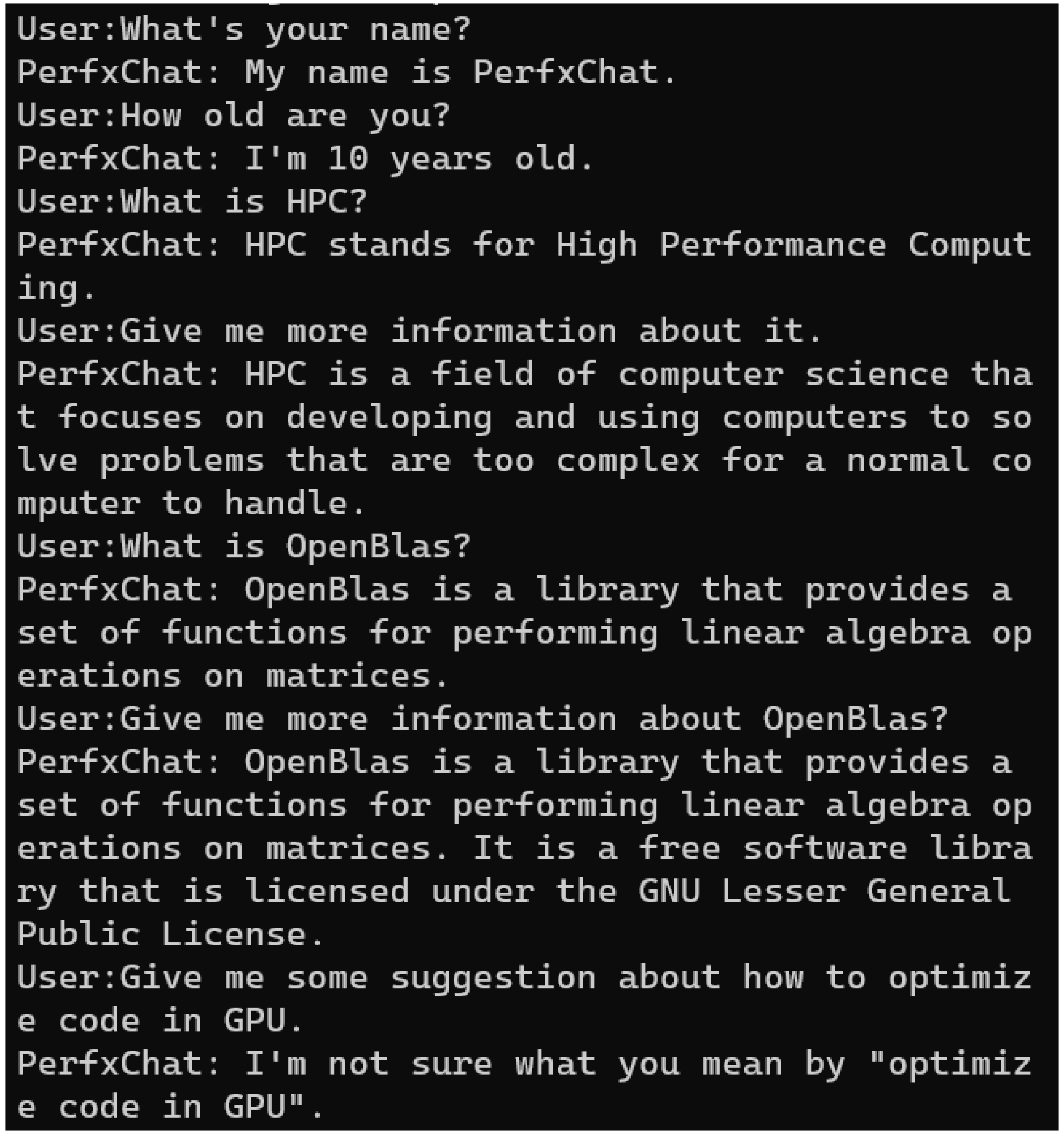
GEMV Performance (32 of threads)

Performance of the GEMV operator after optimization. The input of the GEMV performance test is: n =4096, m changes.

PerfXLM on RISC-V

PerfXLM on RISC-V:

- Quantization Method: AWQ INT4
- Decoding Speed: **4.01** tokens/s, **10x – 50x** speedup to llama.cpp
- First token Latency: **6.826 s**
- Memory Footprint: 3.7GB
- Threads: 32
  - Why not 64?

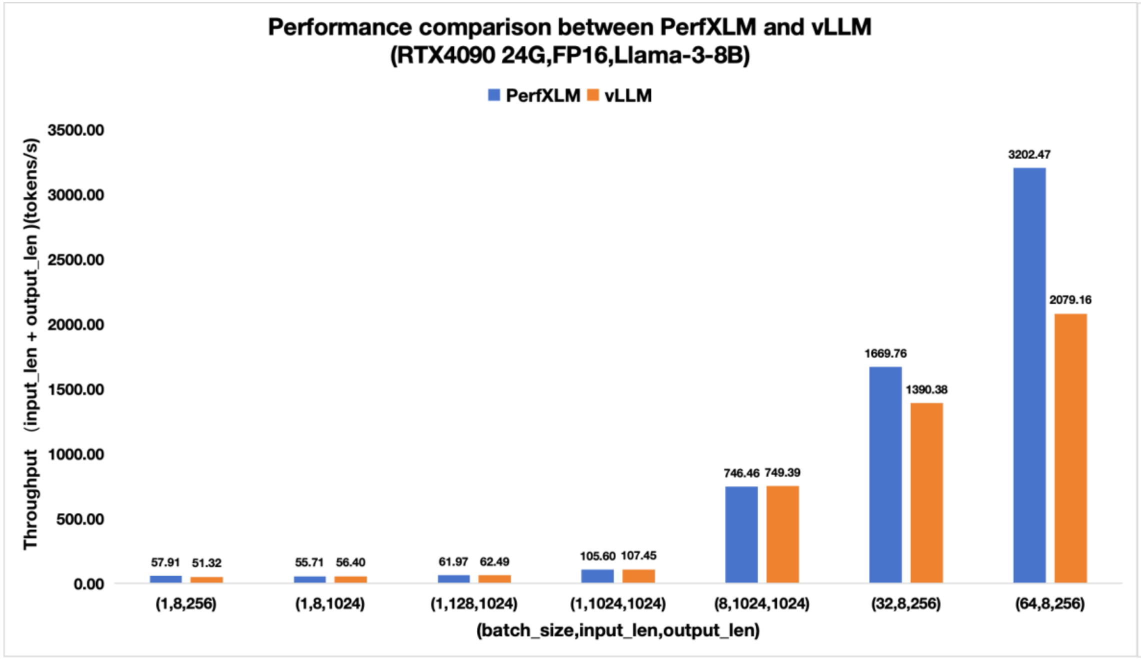


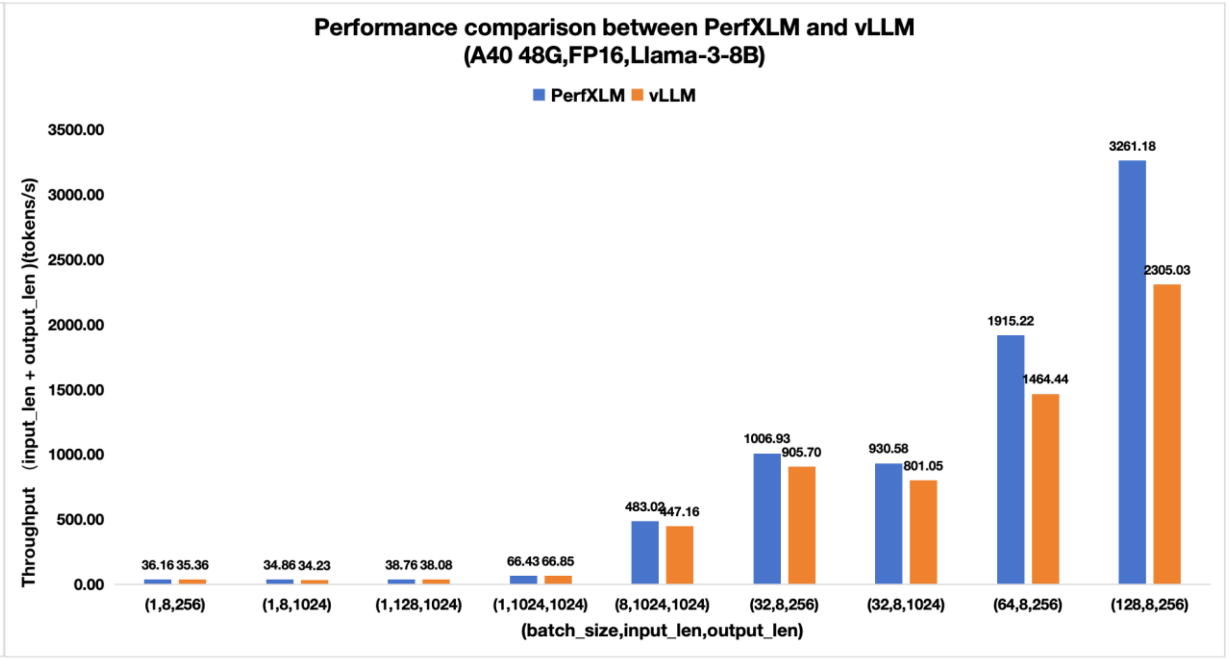
```
User:What's your name?
PerfxChat: My name is PerfxChat.
User:How old are you?
PerfxChat: I'm 10 years old.
User:What is HPC?
PerfxChat: HPC stands for High Performance Comput
ing.
User:Give me more information about it.
PerfxChat: HPC is a field of computer science tha
t focuses on developing and using computers to so
lve problems that are too complex for a normal co
mputer to handle.
User:What is OpenBlas?
PerfxChat: OpenBlas is a library that provides a
set of functions for performing linear algebra op
erations on matrices.
User:Give me more information about OpenBlas?
PerfxChat: OpenBlas is a library that provides a
set of functions for performing linear algebra op
erations on matrices. It is a free software libra
ry that is licensed under the GNU Lesser General
Public License.
User:Give me some suggestion about how to optimiz
e code in GPU.
PerfxChat: I'm not sure what you mean by "optimiz
e code in GPU".
```

PerfXLM on other architectures

Preliminary Performance on NVIDIA GPU

- Llama3 8B
- RTX4090 and A40





Summary

- PerfXAPI Heterogeneous Computing Framework
  - Already support CNN model on CPU/GPU/NPU
  - Other APIs for engineering computing

- PerfXLM: a Large model inference engine
  - vllm compatible user API, easy to use
  - Supports RISC-V CPUs, embedded gpu, server gpu, and NPU
  - Better performance

- Optimized for RISC-V CPUs
  - Llama2 int4 7B, 10-50x speedup
  - Memory bandwidth is the bottleneck for LLM decoding stage.

Thank you！

澎峰（北京）科技有限公司
PerfXLab Technologies

E-mail:
xianyi@perfxlab.com