PerfXLM: A LLM Inference Engine on RISC-V CPUs

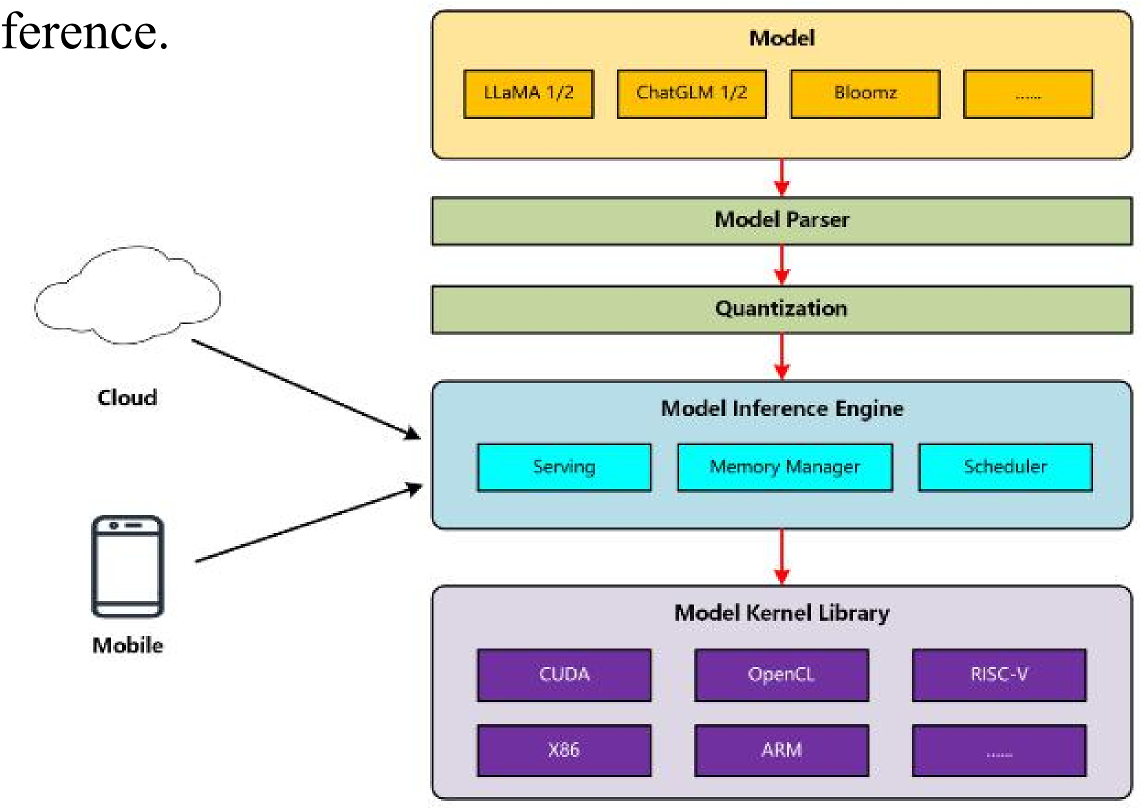
Xinan Yu[1], Chiyo Wang[1], Haochen Zhang[1], Xiandong Liu[1] and Xianyi Zhang[1]*

[1]PerfXLab Technologies, *Corresponding author: xianyi@perfxlab.com

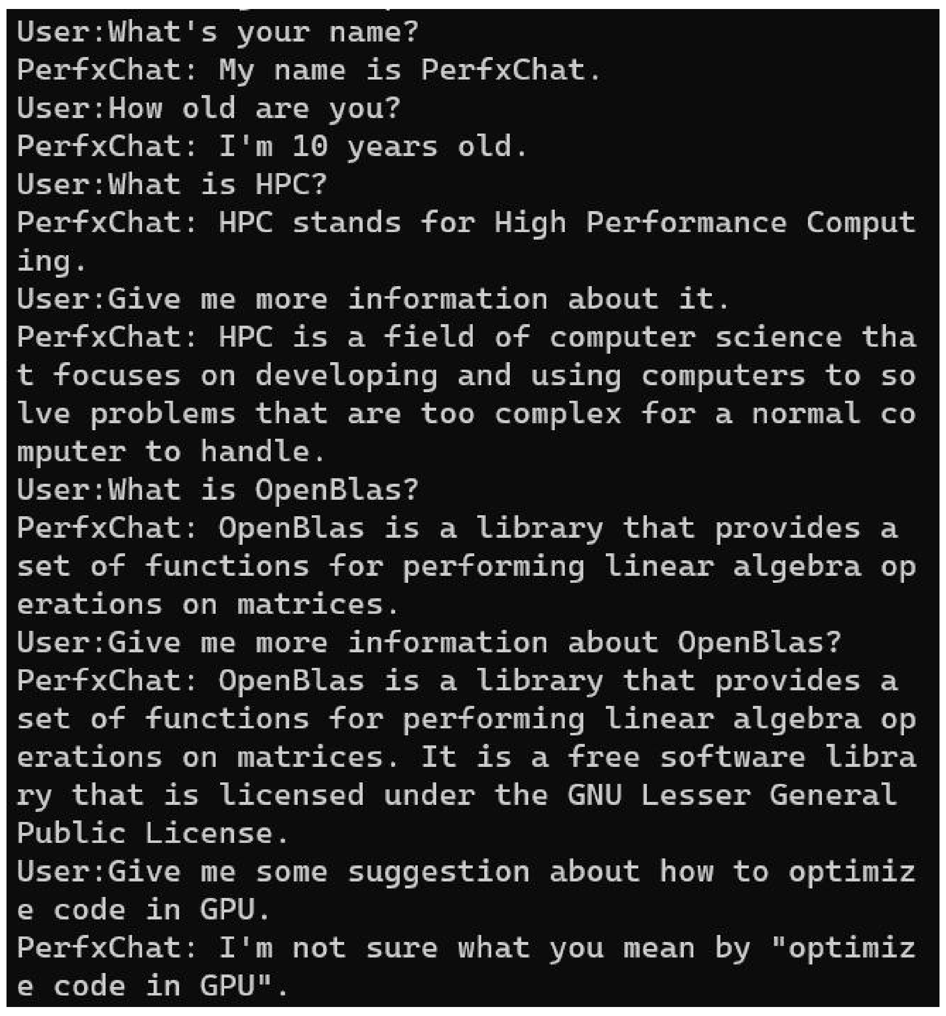
PerfXLab 澎峰科技

Features & Performance

- Support model inference on both **the cloud and device**.
- Support for **multiple heterogeneous platforms** like GPU and CPU.
- **High performance operators** were customized and optimized according to the characteristics of large model inference.



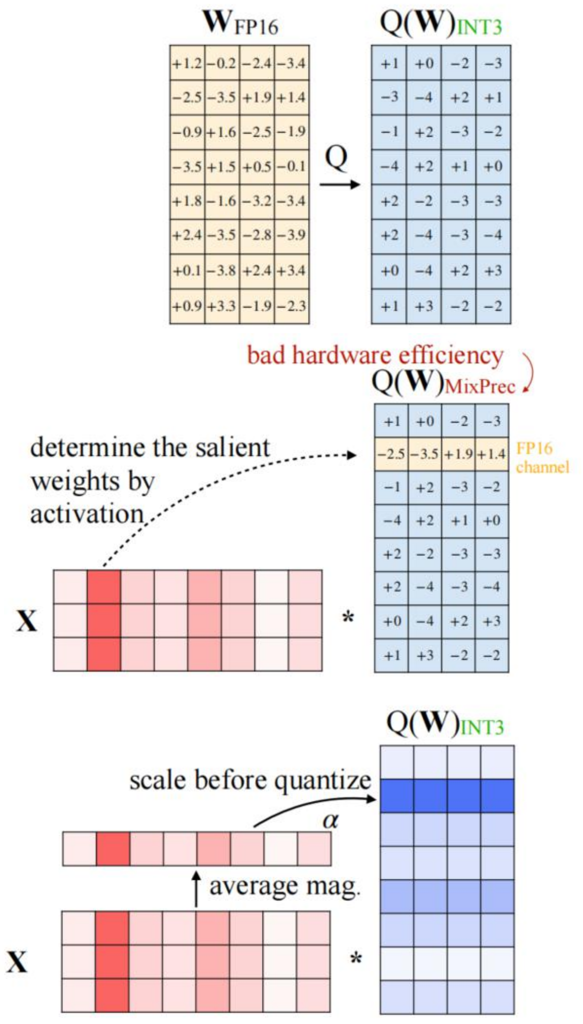
We use INT4 Quantization method of AWQ and develop an APP for Llama. Finally, the generation speed is **4.01 token/s** and memory footprint of the model is **3.7GB**.



```
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".
```

Low Precision Quantization

We use **AWQ**(Activation-aware Weight Quantization) method, only quantizing the weights. For the FP16 model, we quantizes it into INT4 to reduce the memory footprint, which now is a quarter of the original. The adopted parameter group_size is 128, the scale parameter and zero parameter only require an additional storage of about **1%** of weight matrix size.



W_{FP16}

+1.2	−0.2	−2.4	−3.4
−2.5	−3.5	+1.9	+1.4
−0.9	+1.6	−2.5	−1.9
−3.5	+1.5	+0.5	−0.1
+1.8	−1.6	−3.2	−3.4
+2.4	−3.5	−2.8	−3.9
+0.1	−3.8	+2.4	+3.4
+0.9	+3.3	−1.9	−2.3

$Q(W)_{INT3}$

+1	+0	−2	−3
−3	−4	+2	+1
−1	+2	−3	−2
−4	+2	+1	+0
+2	−2	−3	−3
+2	−4	−3	−4
+0	−4	+2	+3
+1	+3	−2	−2

$Q(W)_{MixPrec}$

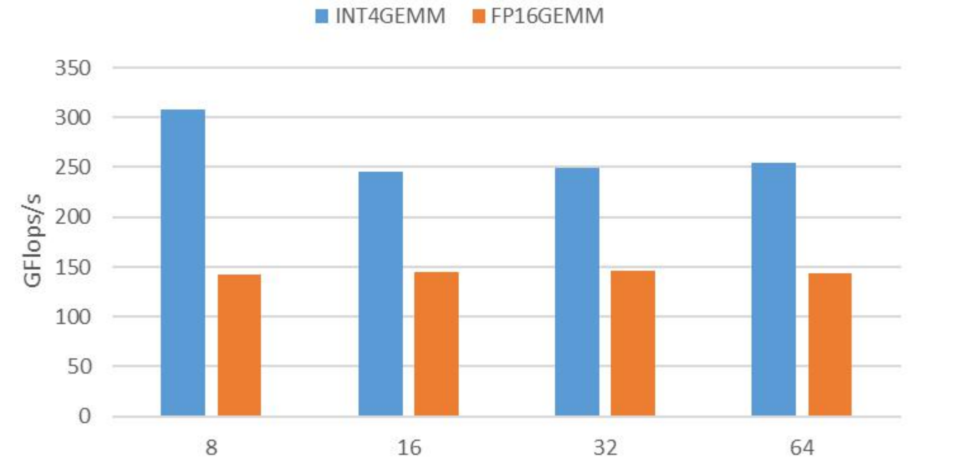
+1	+0	−2	−3
−2.5	−3.5	+1.9	+1.4
−1	+2	−3	−2
−4	+2	+1	+0
+2	−2	−3	−3
+2	−4	−3	−4
+0	−4	+2	+3
+1	+3	−2	−2

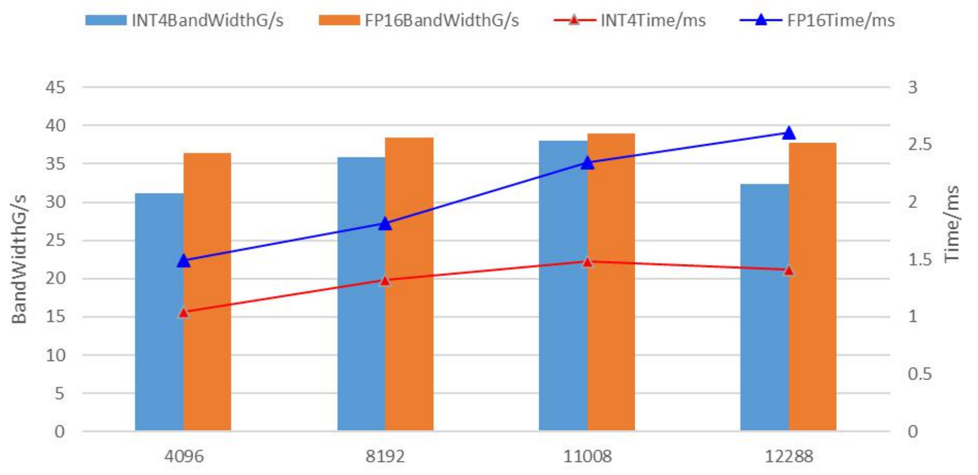
- Only need to deal with 1% of the significant weights to achieve good accuracy.
- The hardware efficiency is low when the outliers are taken out directly.
- Quantization and inverse quantization by preserving scale parameter can maintain accuracy and speed up.

Core Operator Tuning

The main bottlenecks in large model inference are the **GEMM** operator and **GEMV** operator. PerfXLM optimizes for both. The main optimization strategies are as follows:

- Perform data chunking to improve data reuse.
- Vectorized memory access.
- Unroll the core loop manually.
- Parallelize multiple dimensions to improve parallelism.

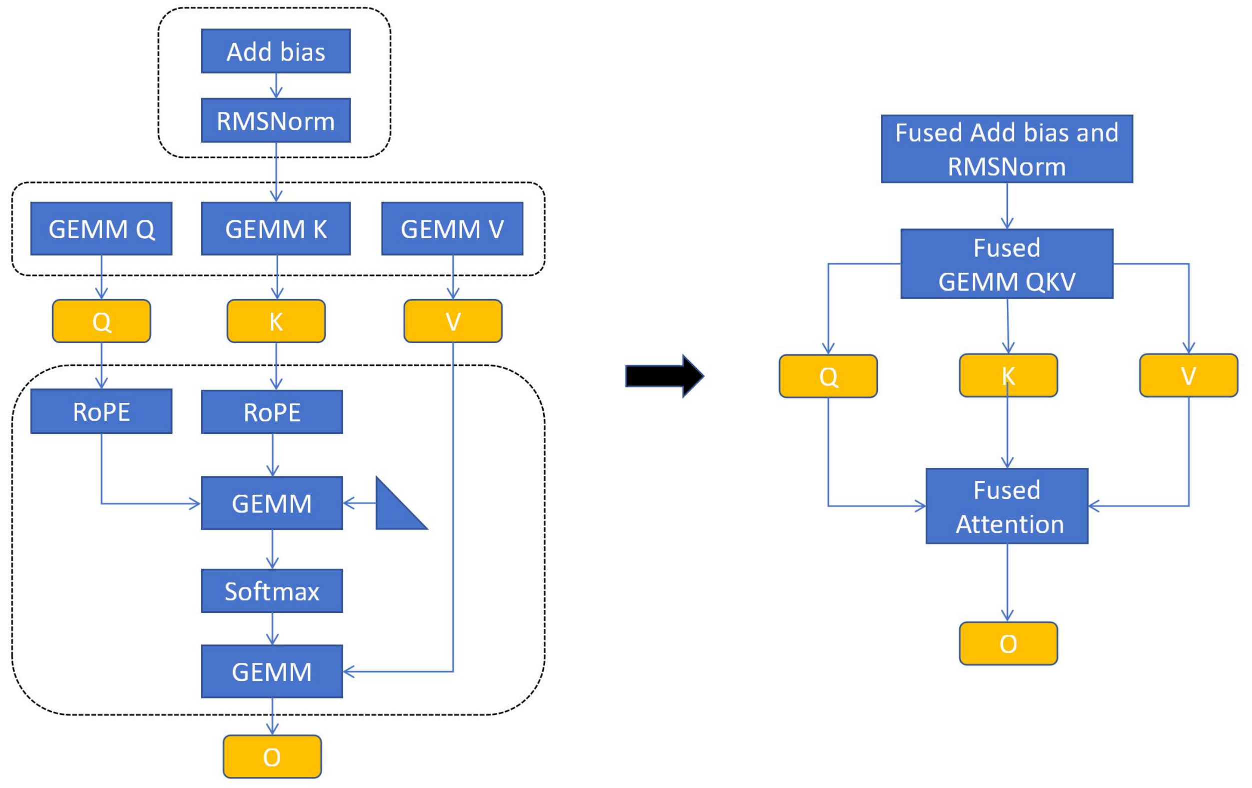




Operator Fusion

The operator fusion operation performed by PerfXLM is mainly reflected in three parts:

- PerfXLM fuses the addition operation and normalization operation of the residual network.
- PerfXLM fuses the three matrix multiplication operations to generate matrix Q, matrix K, and matrix V into one, and use the matrix multiplication operation of larger dimension.
- PerfXLM combines all the operations in self attention into one operator.


Add bias
RMSNorm
GEMM Q
GEMM K
GEMM V
Q
K
V
RoPE
RoPE
GEMM
Softmax
GEMM
O
Fused Add bias and RMSNorm
Fused GEMM QKV
Q
K
V
Fused Attention
O