Optimizing Large Language Model (LLM) Inference for Arm CPUs

Dibakar Gope
03/27/2024

© 2024 Arm

LLMs on Arm CPUs

- LLMs have transformed the way we think about language understanding and generation
- Facilitating their efficient execution on Arm CPUs will expand their reach to billions of Arm devices
- LLMs are often BW bound and have a large weight memory footprint – Arm CPUs can achieve competitive performance against other IP
- Arm CPUs are pervasive, providing portability and flexibility – SW compression schemes, etc.
- Question: What is the potential performance of LLMs on Arm CPUs deployed in smartphones and edge devices?

© 2024 Arm
arm

Key results – LLMs on Arm CPUs

- Focusing on Phi2 2.7B 4b quantized (Q4) model as a benchmark

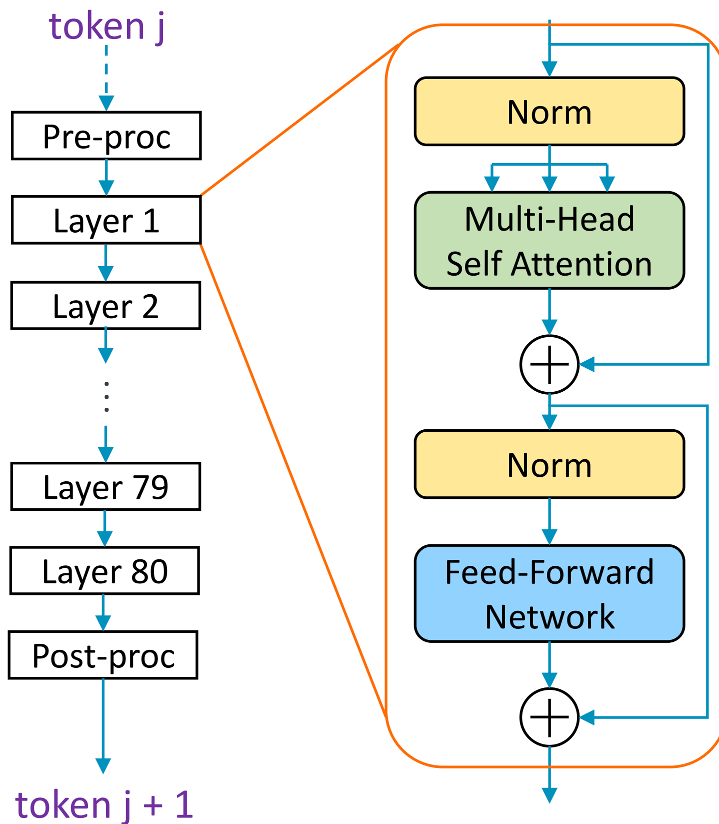
- State-of-the-art C/C++ runtime (e.g., Llama.cpp (GGML)) demonstrates performance on existing Arm platforms but fails to demonstrate the true potential of Arm CPUs

- Developed highly optimized GEMV and GEMM kernels for 4b quantized LLMs

- End-to-end Phi2 2.7B 4bit Speedup on Arm Cortex series CPUs (Cortex-A/Cortex-X):
  Time-to-first-token: 2.3x speedup for 4 threads over GGML
  Text generations afterward: 1.45x speedup for 4 threads over GGML

© 2024 Arm
arm

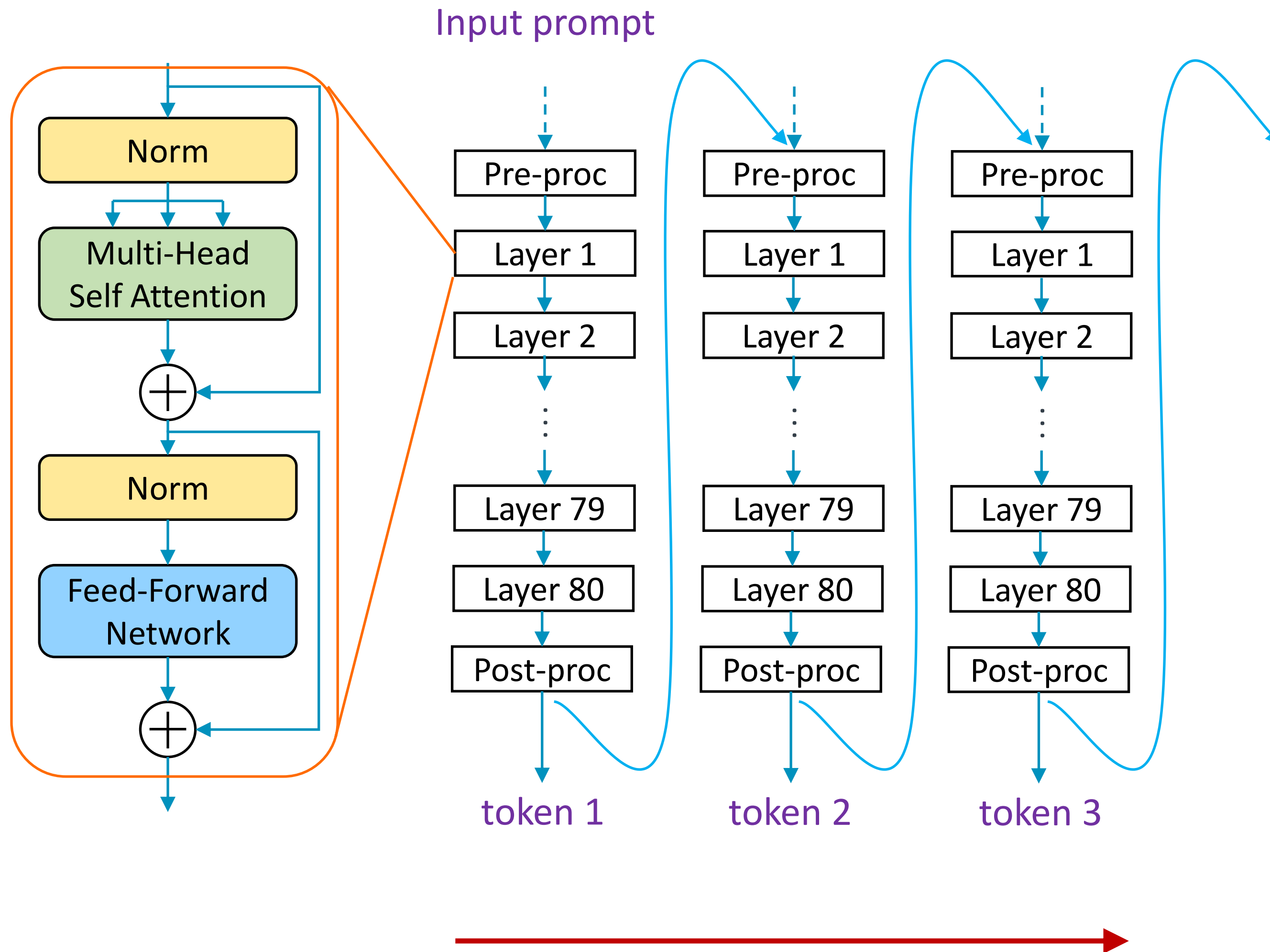
Microsoft’s 2.7B parameter Phi 2 LLM



- Phi2 is a well-known open-source LLM released by Microsoft
- A stack of self-attention transformer layers
- 32 layers for Phi2 2.7B parameters model

© 2024 Arm

How does a language model like Phi2 work?



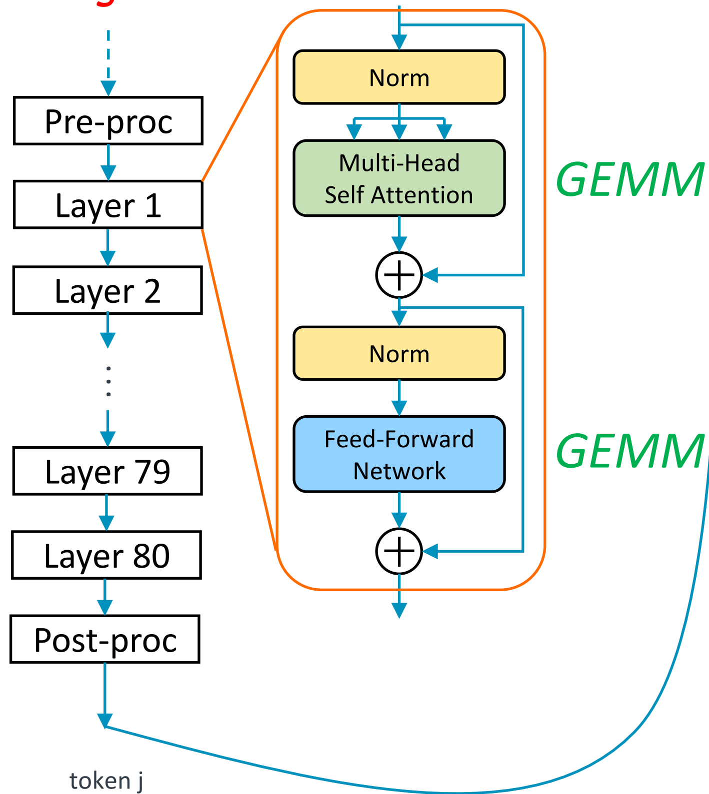
- Each round through the network generates a new token.
- The new token is fed into the network's next round.
- The "state" gradually builds up and is carried from left to right in the figure (through LLM's Key Value cache).
- A typical LLM inference involves going through the network multiple times and generating many tokens.

© 2024 Arm

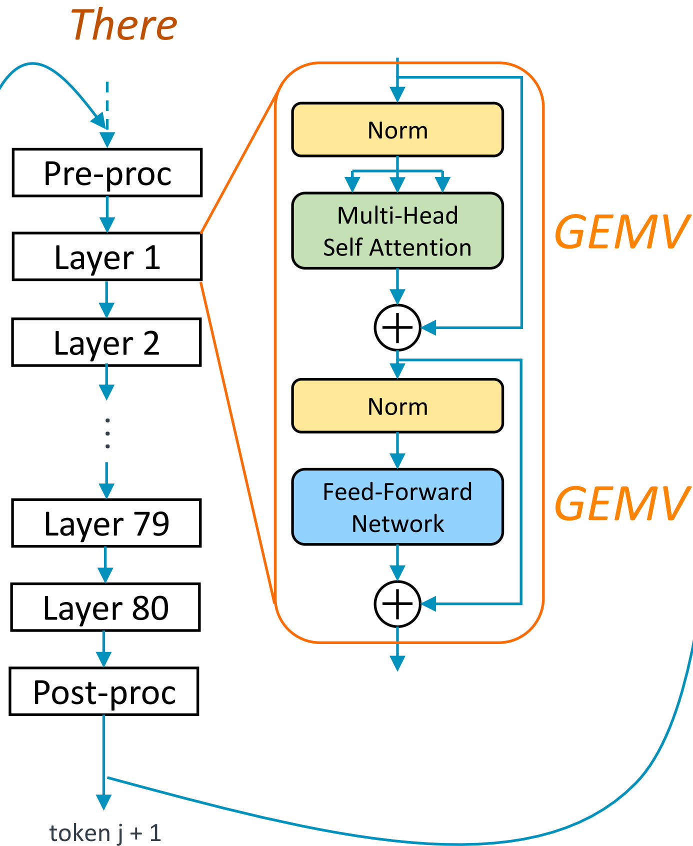
arm

What are three popular chess openings?

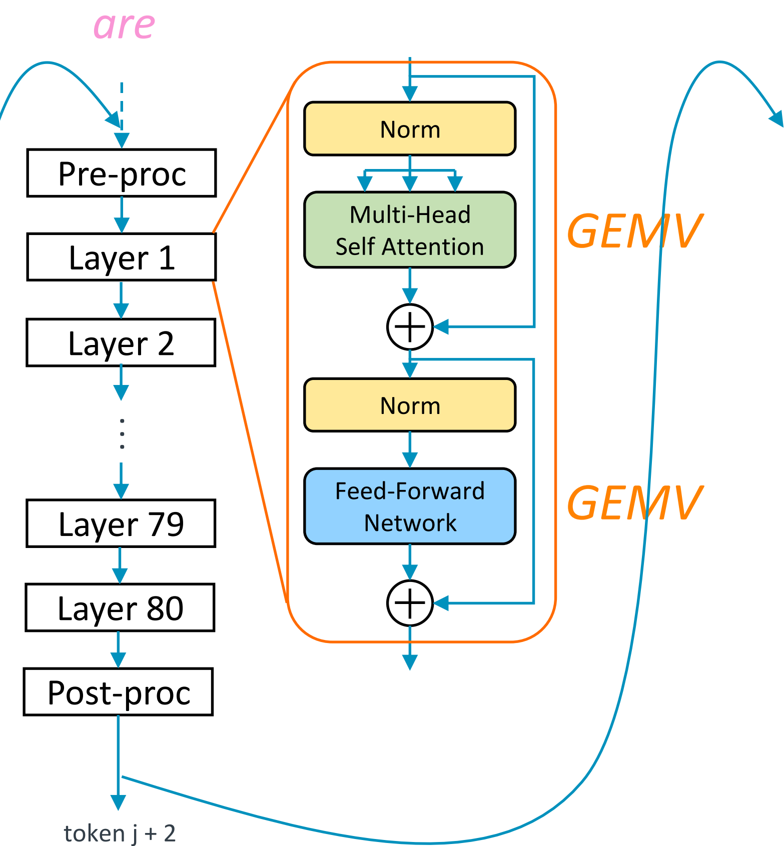
Example



There



are



LLM's key value cache:

After 1st round: What are three popular chess openings?

After 2nd round: What are three popular chess openings? There

…

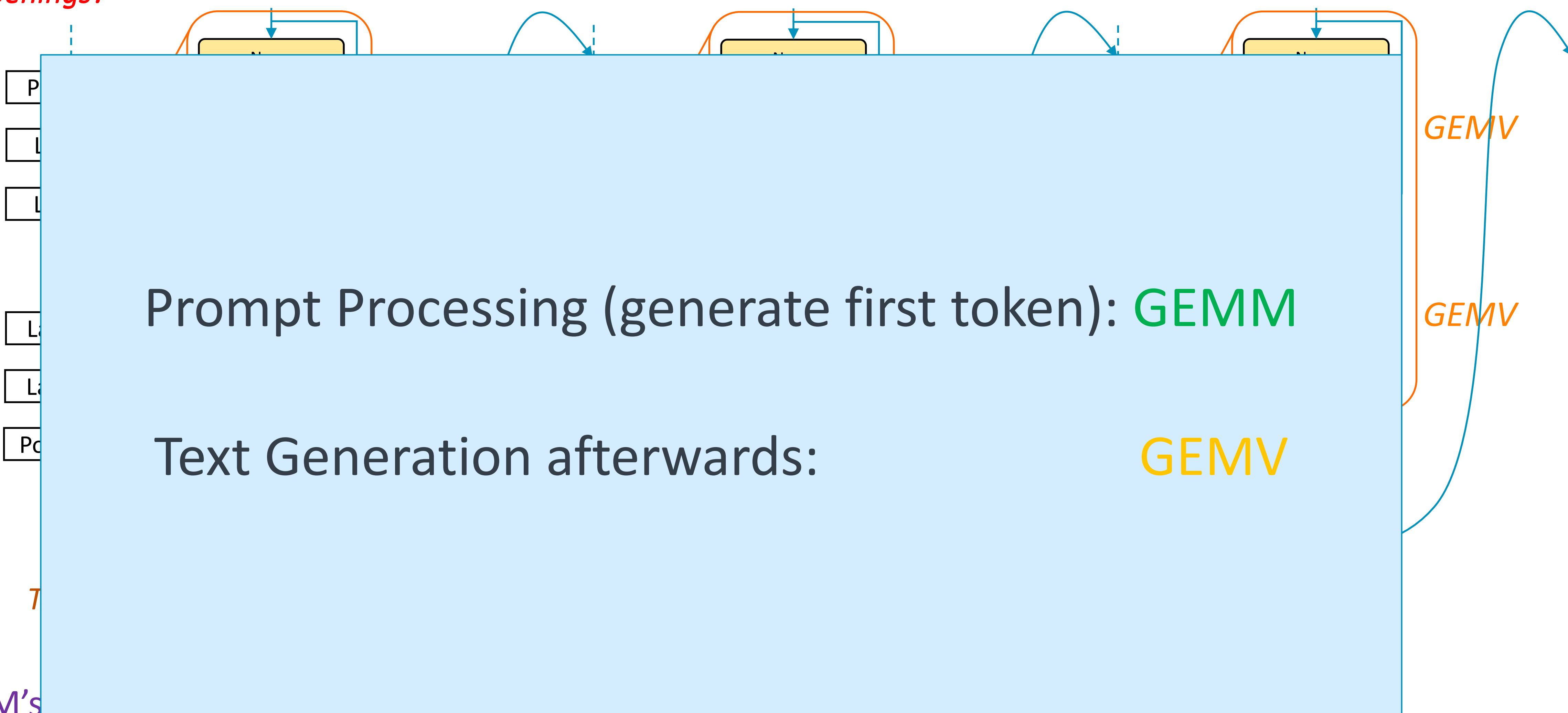
After Nth round: What are three popular chess openings? There are …

© 2024 Arm

arm

What are three popular chess openings?

Example



LLM's
cache:

After 2nd round: What are three popular chess openings? There

…

After Nth round: What are three popular chess openings? There are …

© 2024 Arm

arm

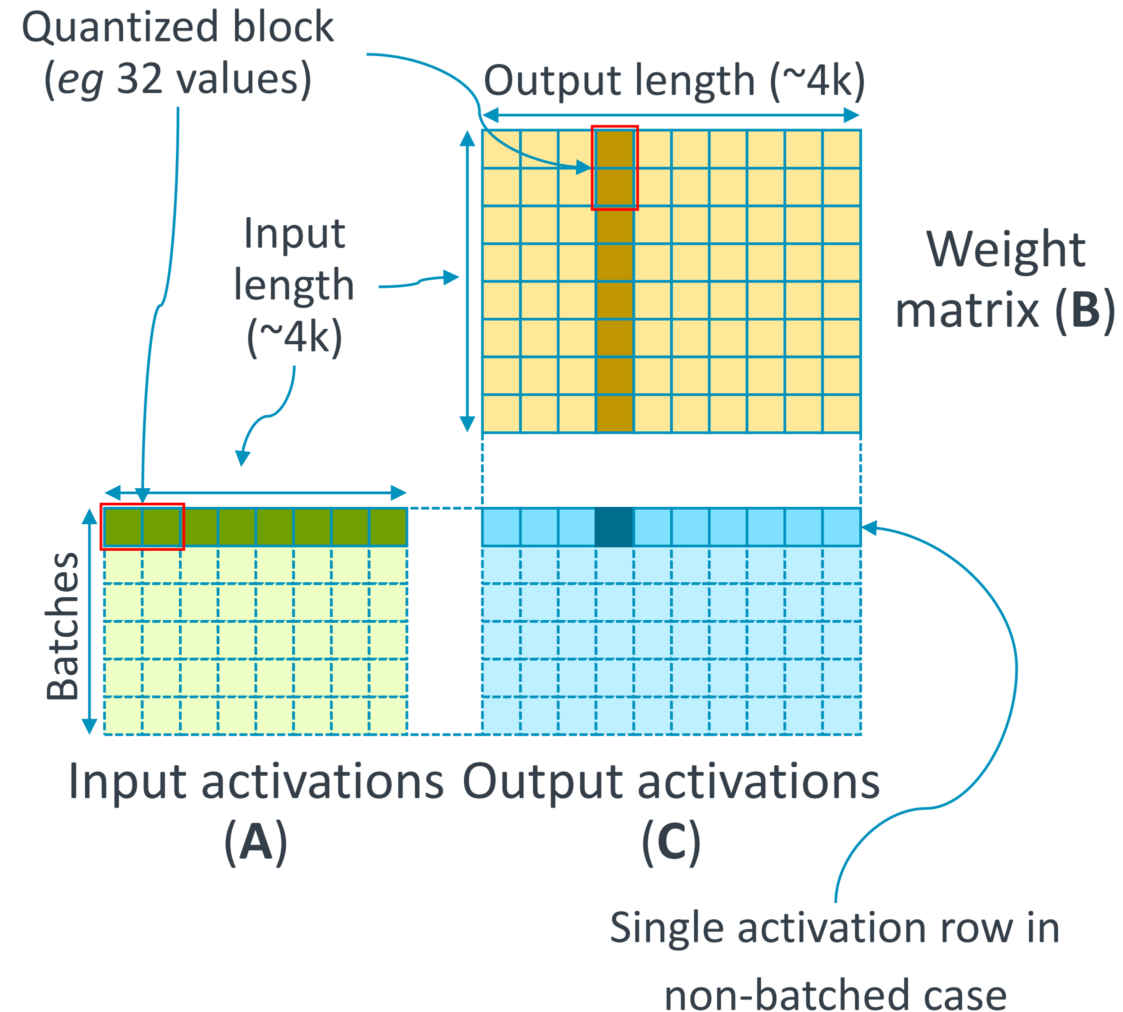
Arm CPUs for benchmarking

- Using Arm Cortex series CPUs
- Readily available, reliable benchmarking platform.
- Using 4-bit “q4_0” quantization scheme - simple, high performance.
- There are other quantization schemes; the optimizations that we perform for Q4_0 should also extend to other schemes.

© 2024 Arm
arm

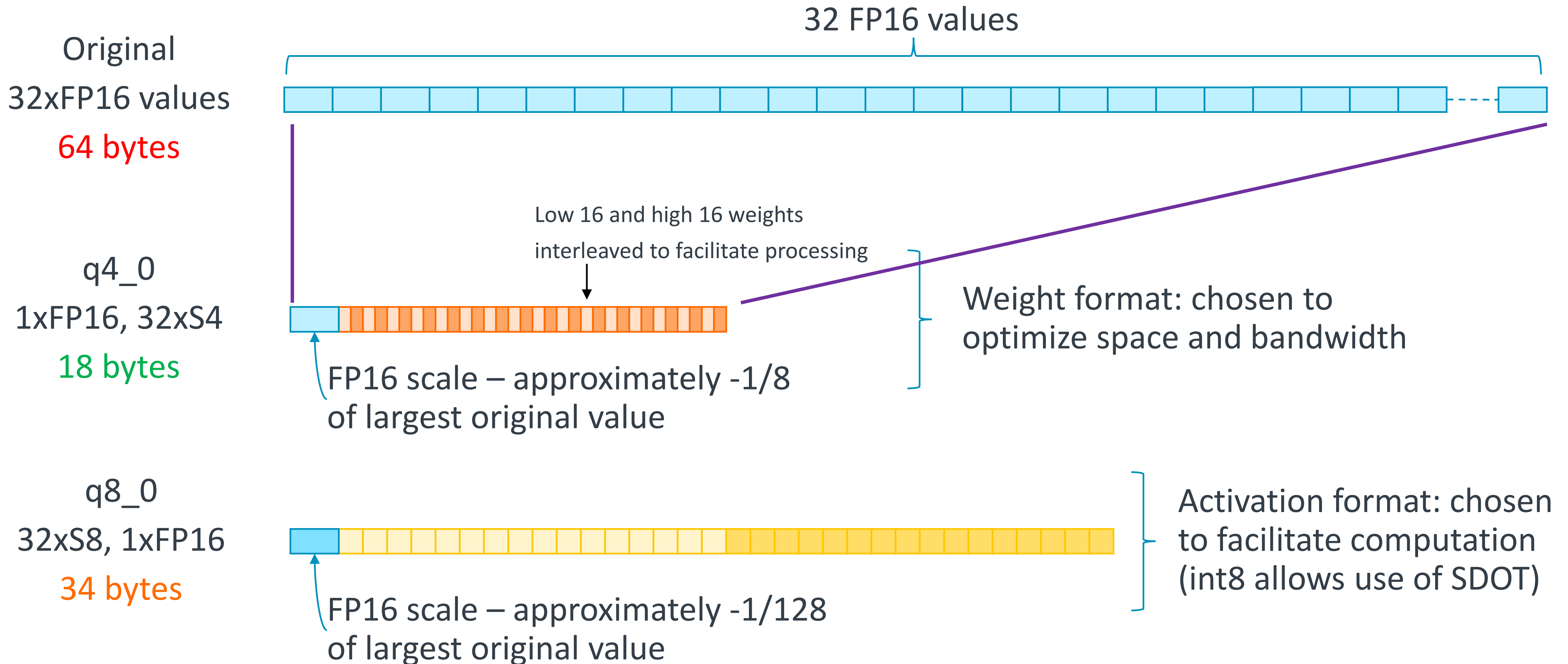
GEMM/GEMV background

- For typical operators in LLMs, weight matrix (B) is much larger than the input (A) and output (C).
- Compression of weight matrix is key to reducing memory and bandwidth consumption.
- In GGML, a dot-product kernel computes a single result – it's called at each point to populate the whole of C.
- LLMs typically use block quantized scheme to store chunks of weight columns and activations



© 2024 Arm
arm

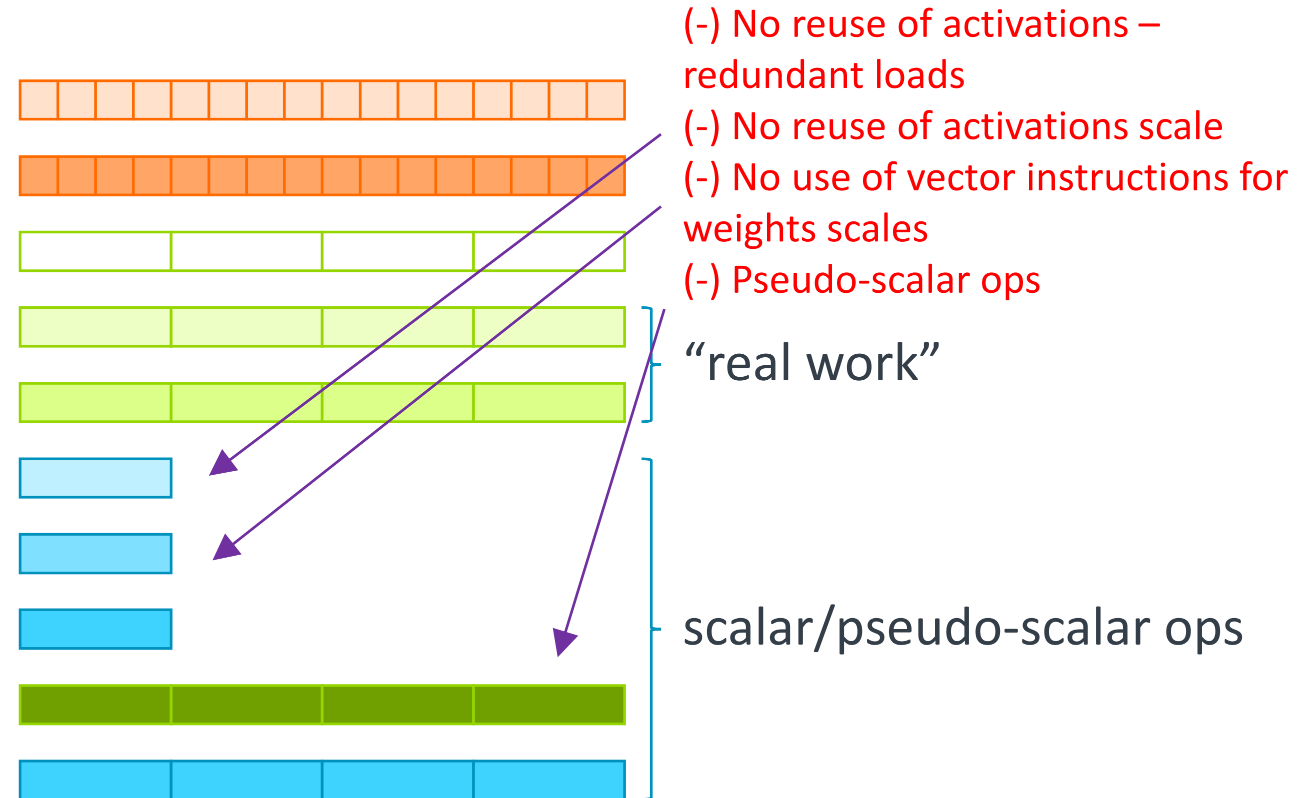
Block Quantized Formats



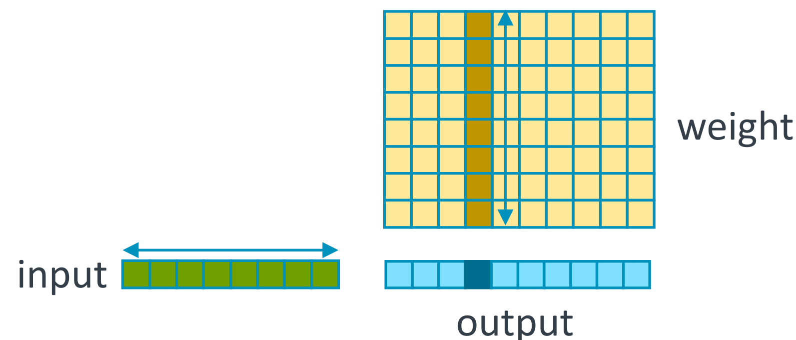
© 2024 Arm

Block processing steps – (original llama.cpp/GGML)

- Expand low weights to 8b (AND, SUB)
- Expand high weights to 8b (SHR, SUB)
- Initialize integer accumulator (MOV)
- Multiply low part (DOT)
- Multiply high part (DOT)
- Convert LHS scale to FP32 (FCVT)
- Convert RHS scale to FP32 (FCVT)
- Combine scales (FMUL)
- Convert integer sum to FP32 (SCVTF)
- Scale + Accumulate (FMLA)



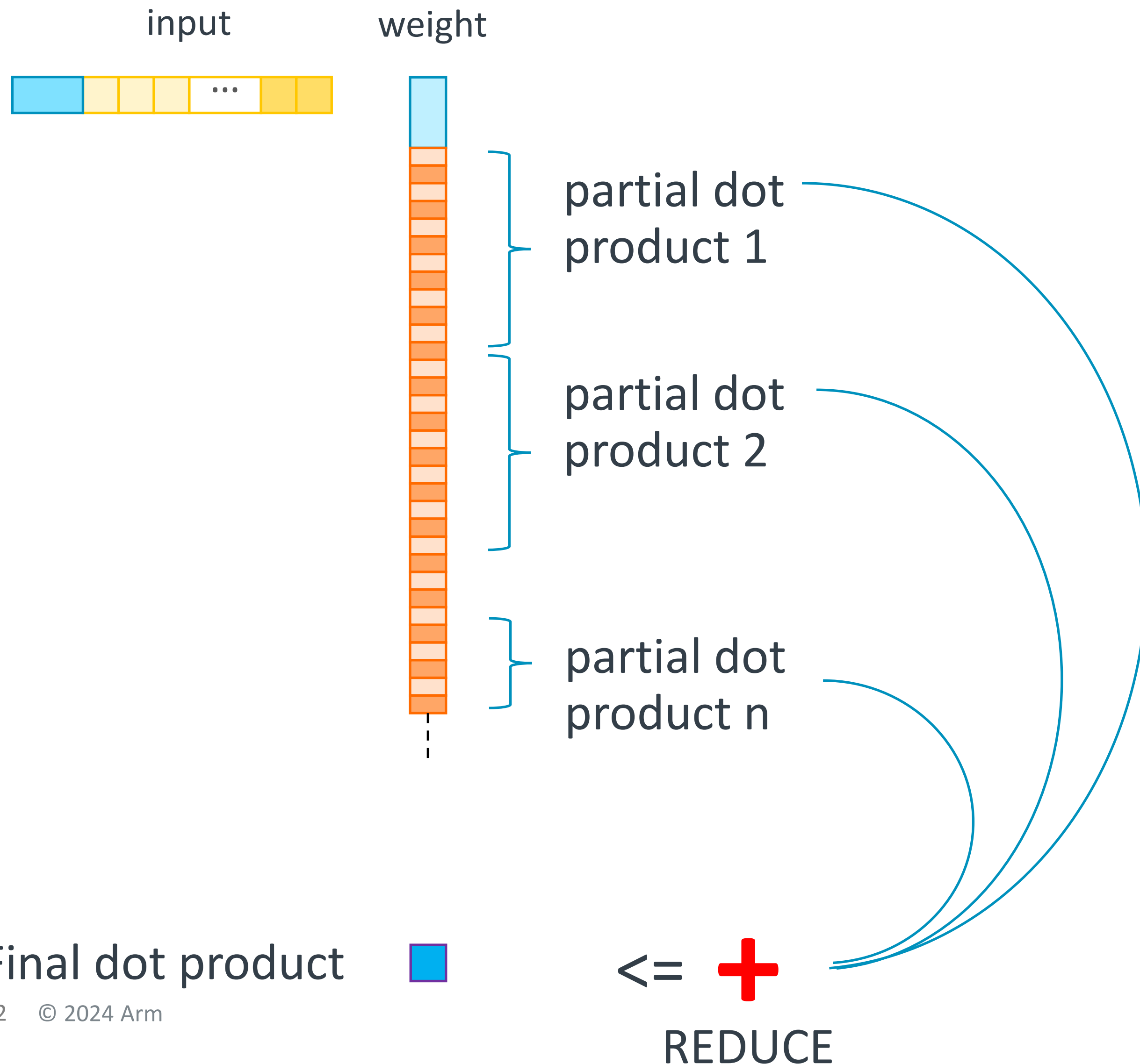
Dot-product kernel in the baseline c/c++ runtime:



 © 2024 Arm

arm

Scalar operations and overall poor MAC efficiency

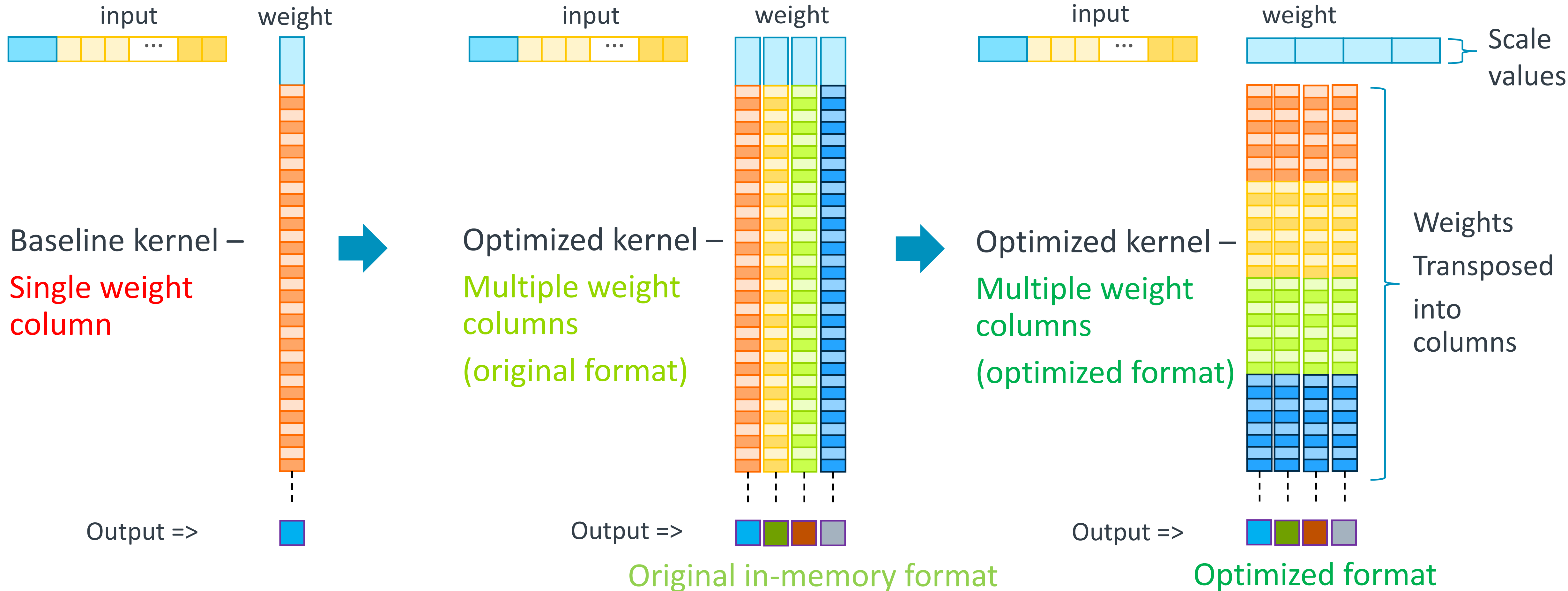


- 12 operations, of which 2 are doing the "real" MAC work (17%)
  - Plus 5 load ops (not shown)
- 50% (6/12) are scalar/pseudo-scalar (work on a vector that is later reduced)
- A lot of compute instructions are not doing useful MAC work
- MAC utilization efficiency: 17%

 © 2024 Arm

arm

Transformed block layout

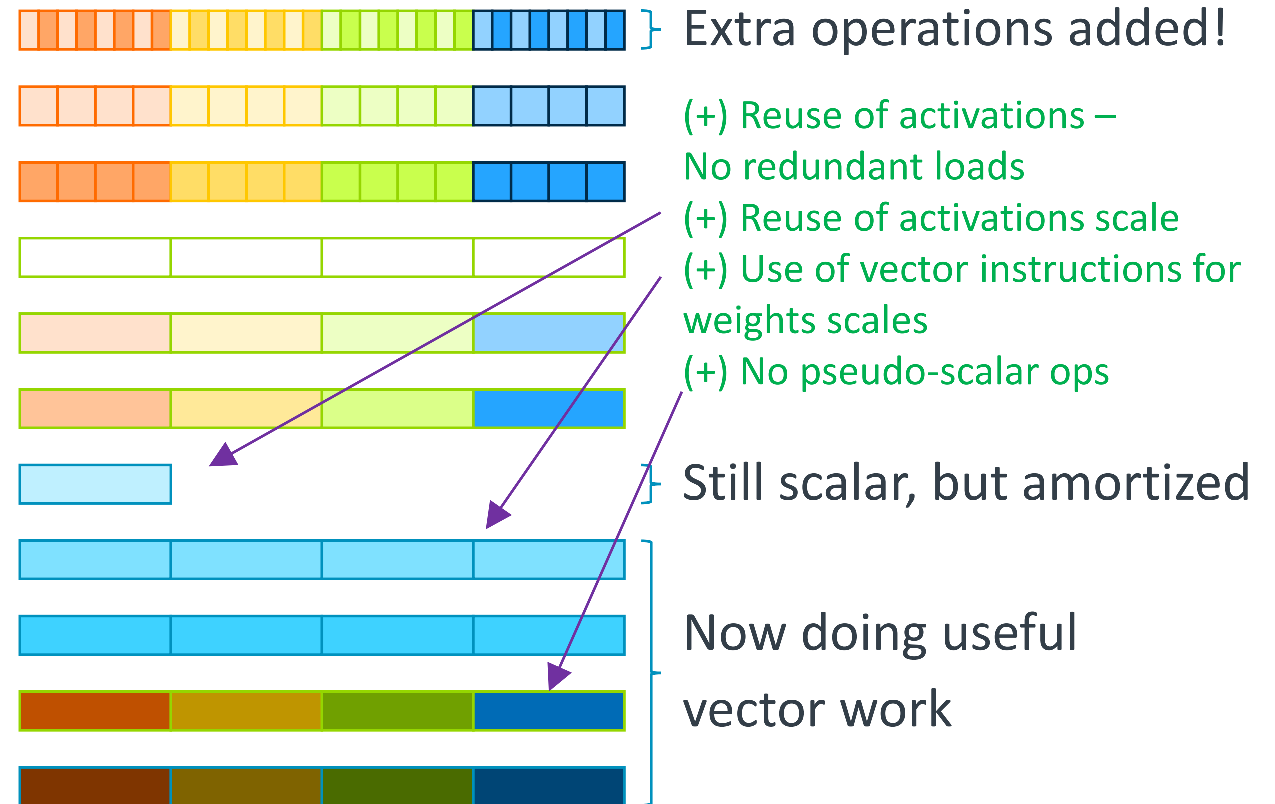


- To avoid pseudo-scalar ops, need to arrange than each lane is working on unique result.
- This means moving data into the relevant lane (transposing).

© 2024 Arm

Block processing steps – 4 simultaneous blocks

- Transpose weights into columns (8x ZIP)
- Expand low weights to 8b (4x AND, SUB)
- Expand high weights to 8b (4x SHR, SUB)
- Initialize integer accumulator (MOV)
- Multiply low parts (4x DOT)
- Multiply high parts (4x DOT)
- Convert LHS scale to FP32 (FCVT)
- Convert RHS scales to FP32 (FCVT)
- Combine scales (FMUL)
- Convert integer sum to FP32 (SCVTF)
- Scale + Accumulate (FMLA)

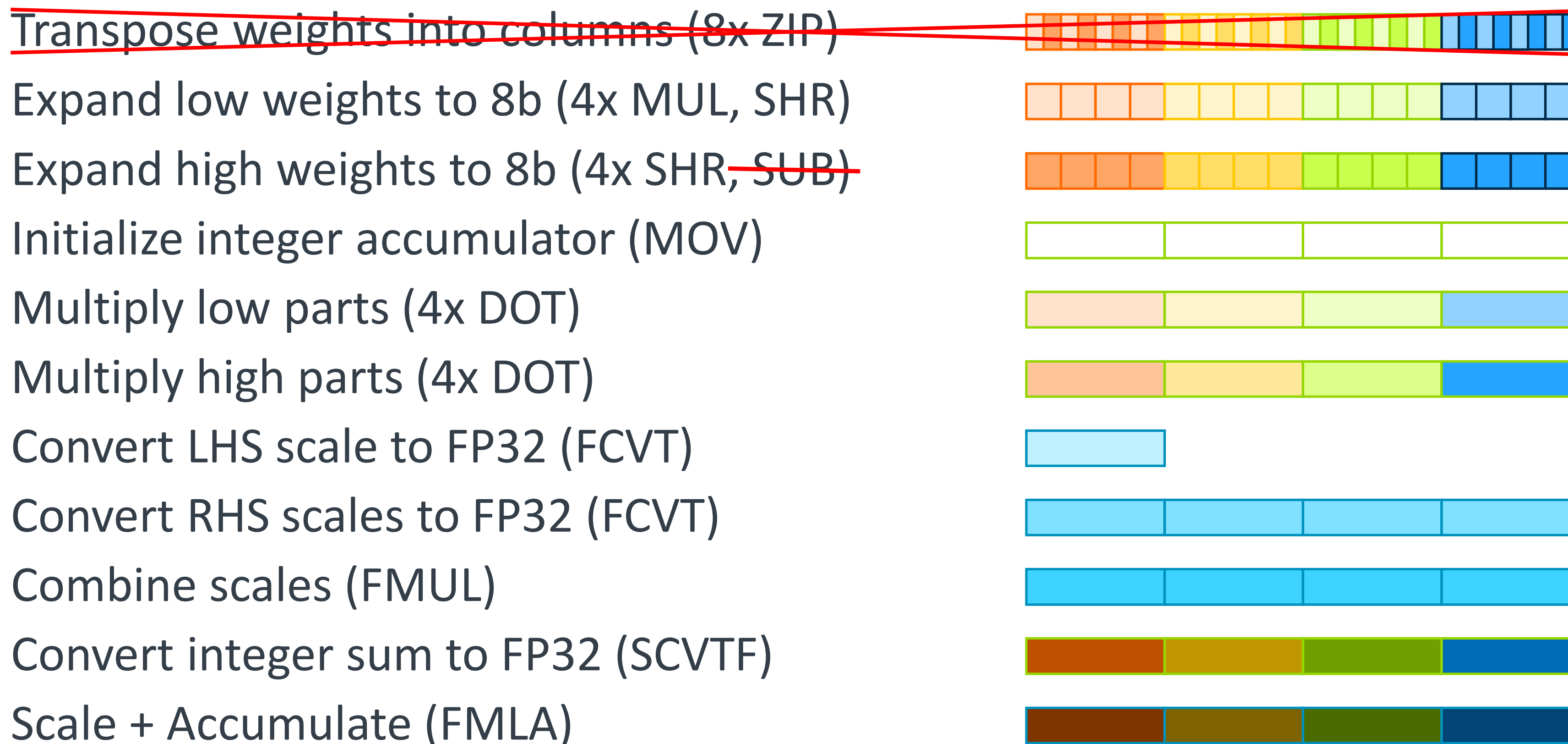


- 38 operations, computing 4 blocks => 9.5 operations per block
- MAC utilization efficiency: 21% => 26% speedup over original code

© 2024 Arm

arm

Block processing steps – optimized memory format



- 26 operations, computing 4 blocks => 6.5 operations per block
- MAC utilization efficiency: 31% => 85% speedup over original code

© 2024 Arm

arm

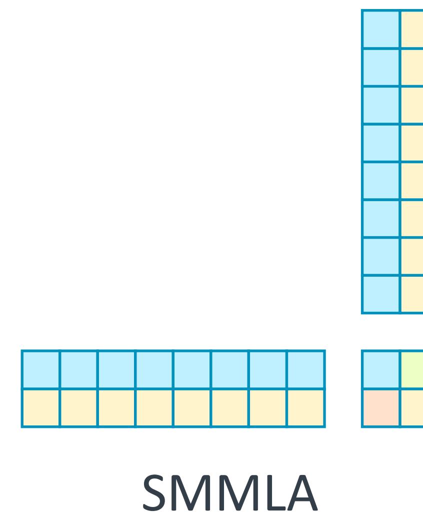
Optimized GEMM for prompt phase (time-to-first-token)

- Uses the same optimizations as GEMV:
  - Weights in blocks prearranged ready for processing.
  - Apply the same to activations (as we now process multiple rows of activations).

- Uses SMMLA instruction (doubles the MAC count of SDOT)
  - Considers multiple inputs rows at a time, and generates multiple output rows

- Reduced bandwidth consumption as each input is processed by several SMMLA instructions.

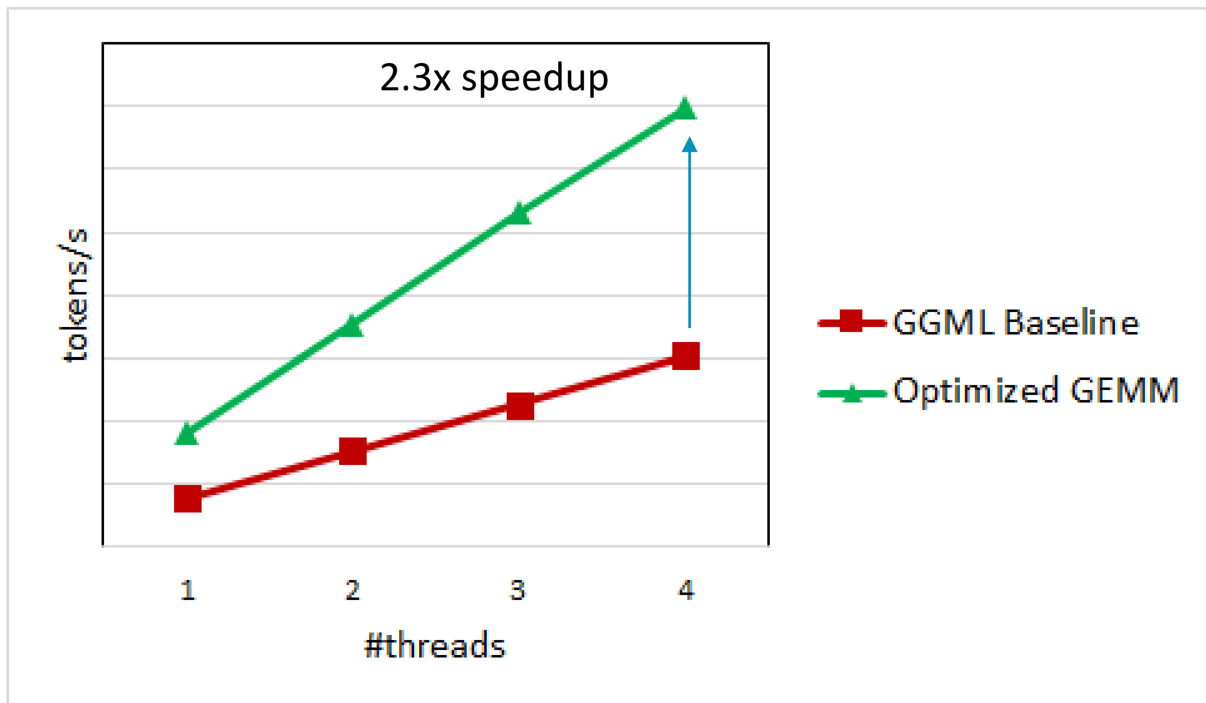
- MAC utilization efficiency: 40%



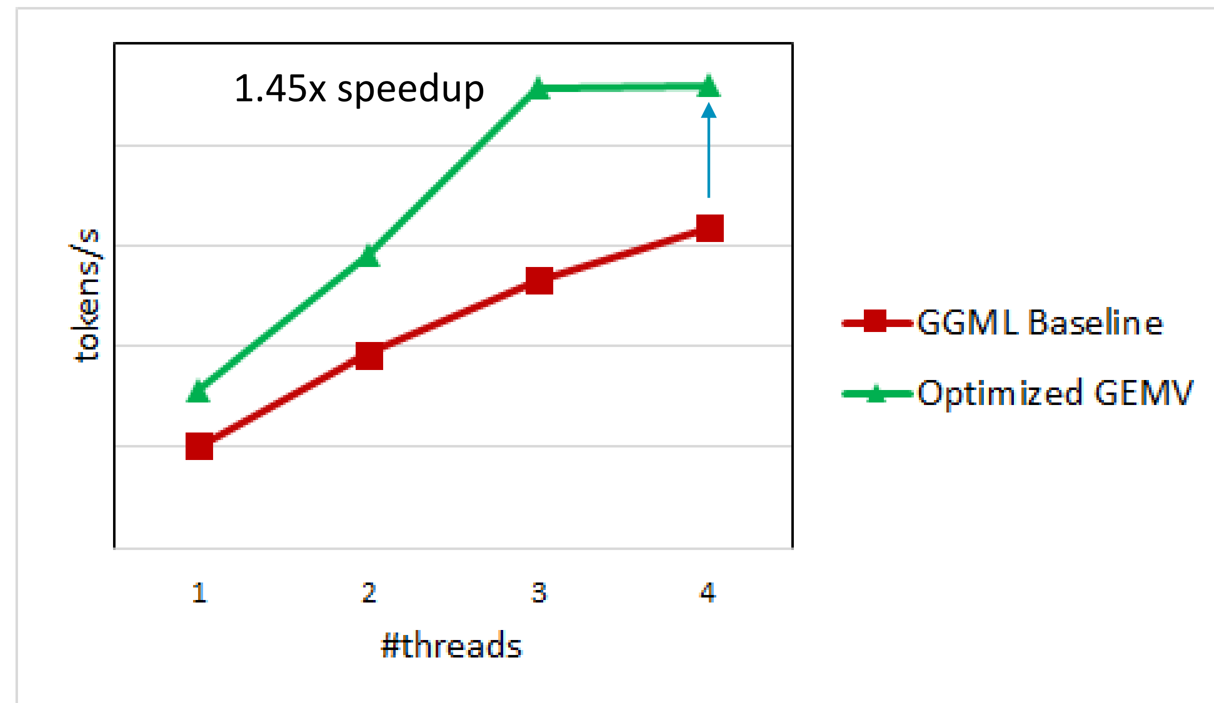
© 2024 Arm
arm

Phi2 2.7B Q4_0 for single inference case on Arm Cortex series CPUs

Prompt processing tokens/s (using GEMM)



Text generation tokens/s (using GEMV)



© 2024 Arm
arm

Conclusions

- Arm CPUs are a suitable platform for LLM inference
- We have other model and runtime optimizations that improve the runtime even further (outside the scope of this talk)
- Arm blog: https://community.arm.com/arm-community-blogs/b/ai-and-ml-blog/posts/generative-ai-on-mobile-on-arm-cpu

© 2024 Arm

arm

arm

Thank You
Danke
Gracias
Grazie
谢谢
ありがとう
Asante
Merci
감사합니다
धन्यवाद
Kiitos
شكرًا
ধন্যবাদ
תודה
ధన్యవాదములు

© 2024 Arm

The Arm trademarks featured in this presentation are registered trademarks or trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All rights reserved. All other marks featured may be trademarks of their respective owners.

www.arm.com/company/policies/trademarks

© 2024 Arm