Porting RWKV to RK3588 NPU

Martin Chang

Disclaimer

- Opinions are my own and not the views of my employer
- My opinions are my own
- My work, non is done for my employer
- etc.... you know the deal

- I want a low-power and fast local LLM

Degenda

- RK3588 NPU
- RWKV
- Attempt 1 - Convert from ONNX
- Attempt 2 - offloading MatMul from GGML
- Conclusion

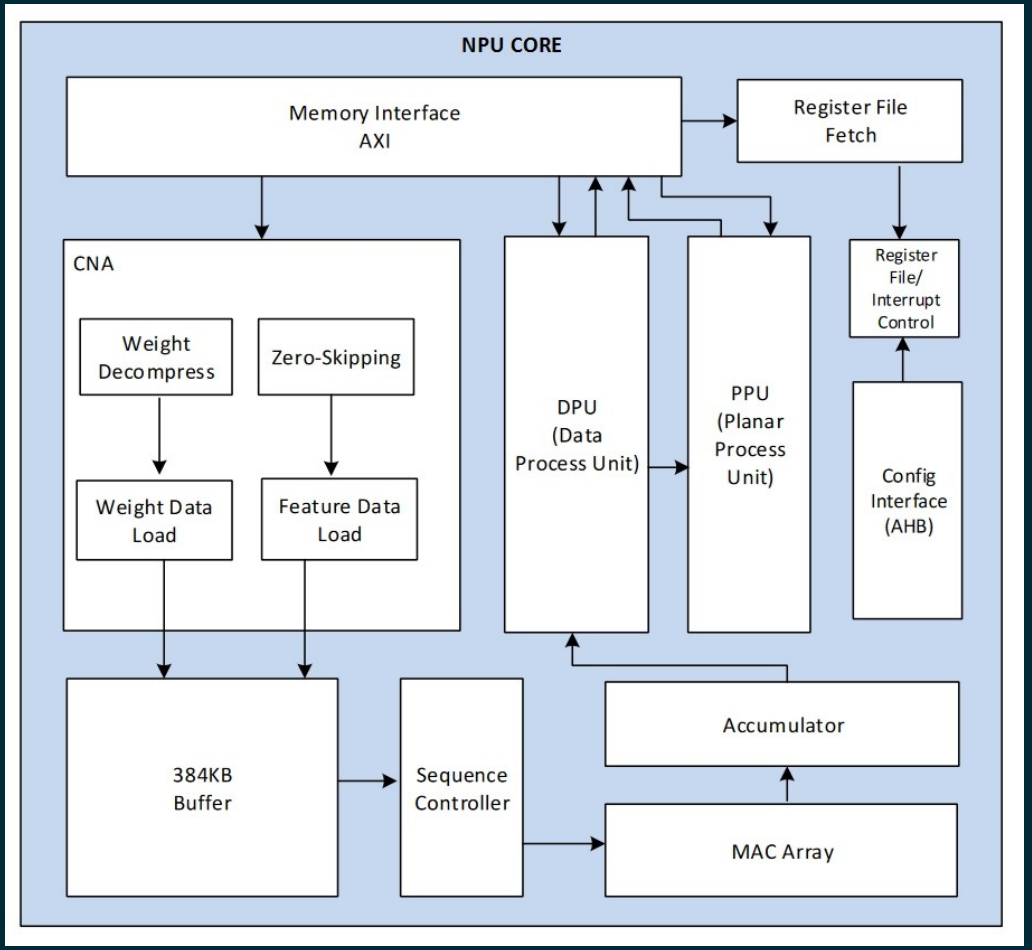
RK3588 NPU

RK3588 NPU

- Fixed pipeline convolution processor
- 3 cores
- 6TOPS @ INT8
- 3TOPS @ FP16
- HW sepc claims INT4/INT16/FP32 capablity. No SDK support
- Really is designed for vision models

RK3588 NPU

- Fixed pipeline
- Non programmible



rknn-toolkit2

- Rockchip proveides RKNN-toolkit2 to compile ONNX
- Then load RKNN using Python or C
- Run the model

rknn-toolkit2

- Compiles ONNX into RKNN
- Compile can crash if graph too complex
- Can only quantize image files as input
- Does not support dynamic input size or graph

MauMul performance

- As of RKNPU 1.5.2, single NPU core only
  - Much better then naive or basic tiling
    - As fast as OpenBLAS multi-core
      - Max K = 2048 @ FP16

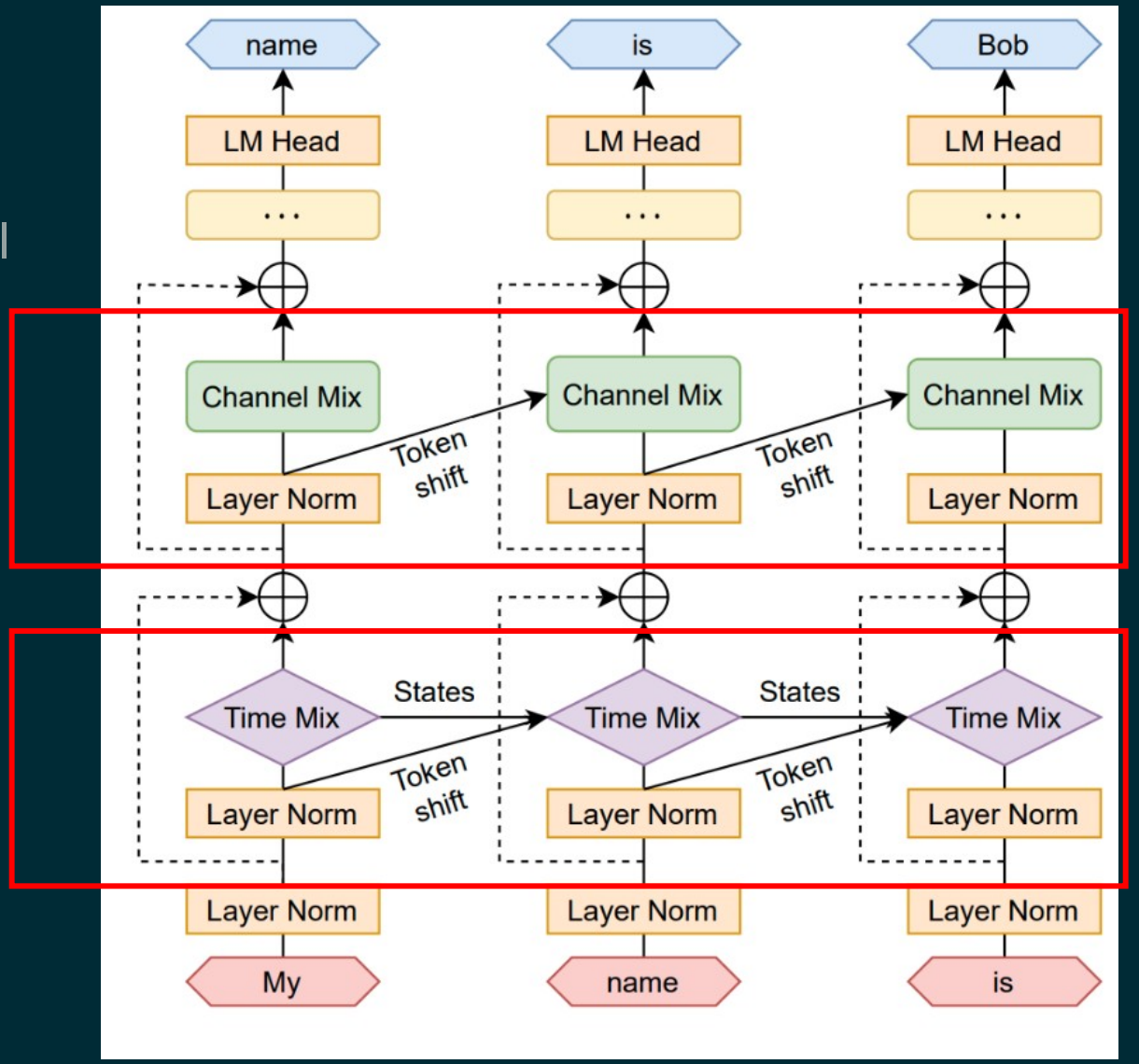
RWKV

RWKV - BlinkDL/RWKV-LM

- RNN with Transformer performance
- Easy to implement - MatMul only
- arXiv: 2305.13048

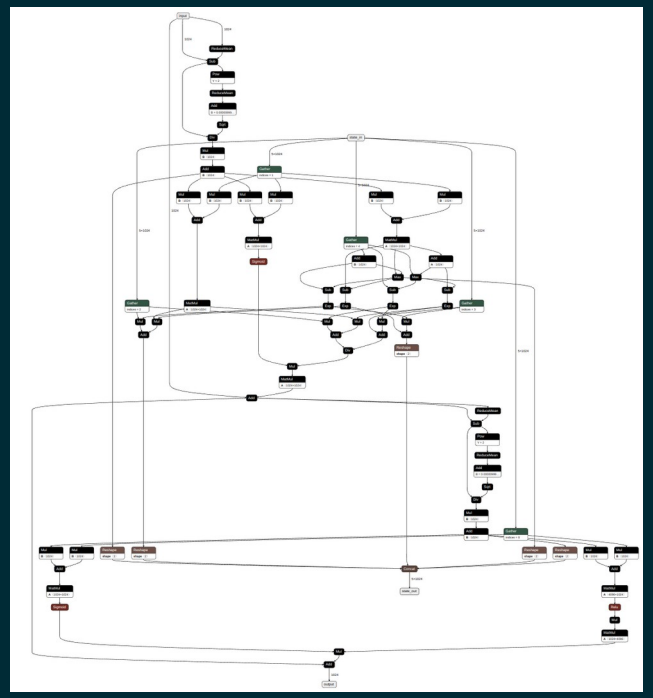
RWKV

- Channel Mix and Time MIx are just 4 matmul
- Output of layer N goes to layer N+1
- ⊕ means element-wise multiplication
- To generate the next token
  - Grab the current token. Feed into network
  - Also use the intermid state
- No multihead attention
- O(1) context O(N) state size

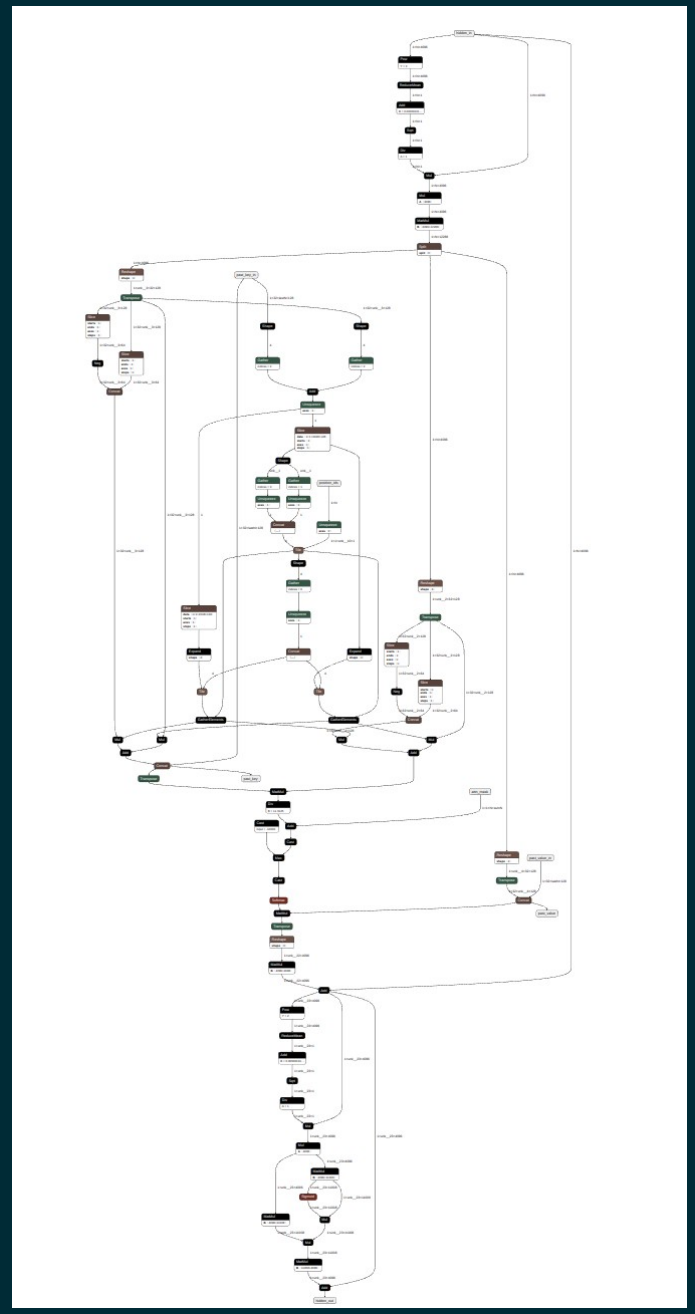


Much Simpler - images to scale

- Layer in RWKV

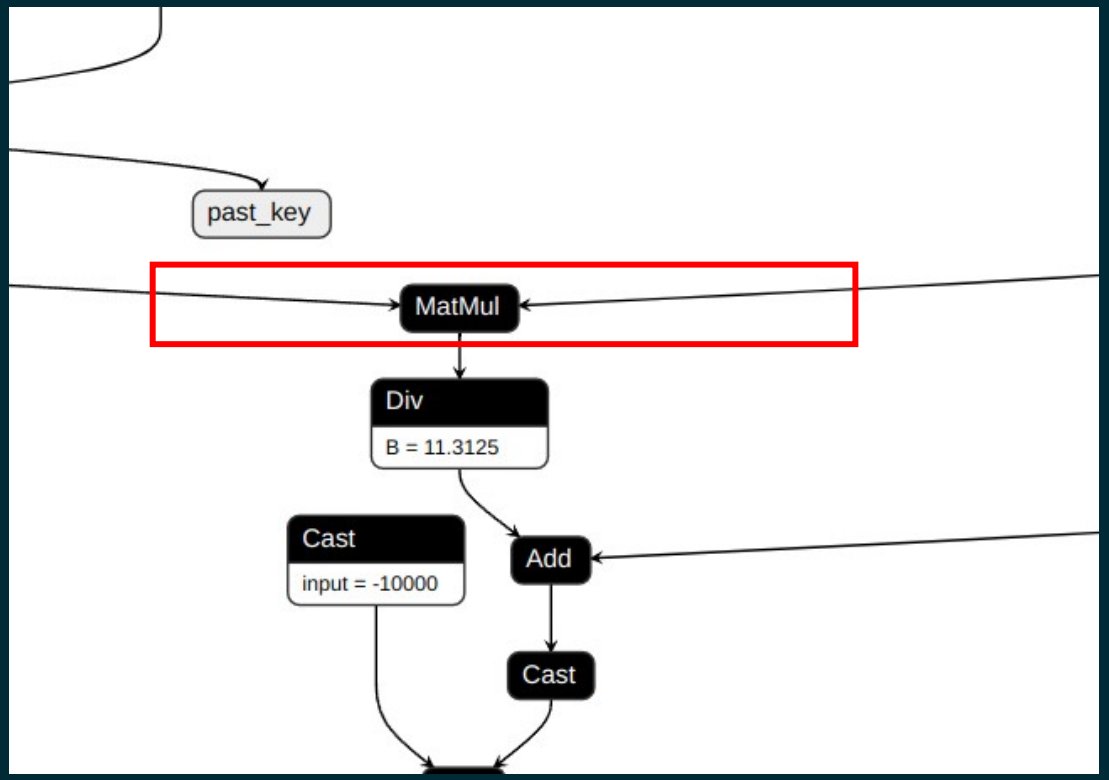


- Layer in LLaMA2



Transformer Self Attention

- RKNN cannot do self attention
- Need to multiply 2 computed matrix
- But RKNN needs mat B be ordered
- LLaMA is impossible

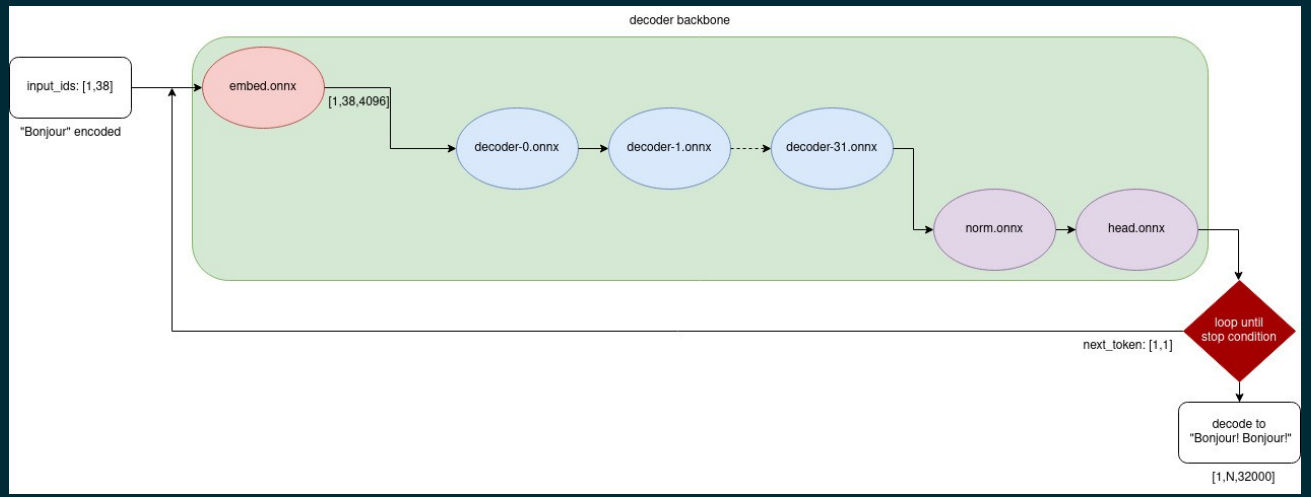


Attempt 1 - Convert from ONNX

Fighting with the toolchain

tpoisonooo/llama.onnx

- Like I said, RKNN can’t do attention
- Also contains RWKV model?????
- Layers are split into different ONNX files



Converting to RKNN

- embed and head layer works well!
- Crash on mixing layers
- DieHard did not save it :(
- Other compiler bugs..

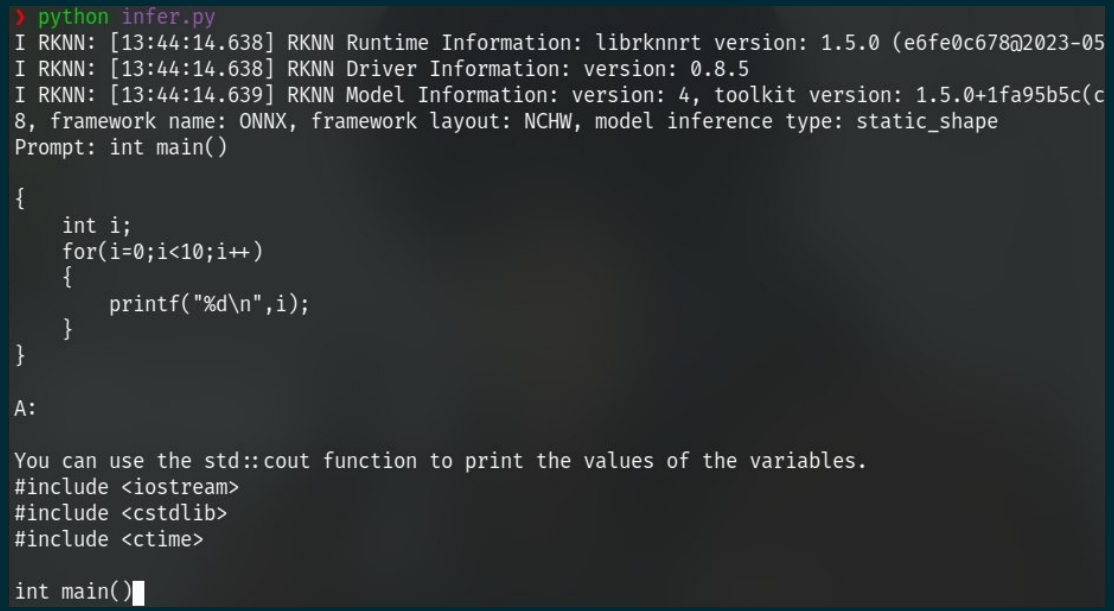
```
❯ cd rknn-rwkv
❯ python convert.py /media/2TB/marty/ChatRWKV/models/mixing_0.onnx /media/2TB/marty/ChatRWKV/models/mixing_0.rknn
W __init__: rknn-toolkit2 version: 1.5.2+b642f30c
W load_onnx: Onnx opset15 is not fully supported, it may cause convert fail, it is recommended to use opset12!
Loading : 100%|                                                  | 32/32 [00:00<00:00, 1247.93it/s]
W load_onnx: The config.mean_values is None, zeros will be set for input 0!
W load_onnx: The config.std_values is None, ones will be set for input 0!
W load_onnx: The config.mean_values is None, zeros will be set for input 1!
W load_onnx: The config.std_values is None, ones will be set for input 1!
W load_onnx: The config.mean_values is None, zeros will be set for input 2!
W load_onnx: The config.std_values is None, ones will be set for input 2!
W load_onnx: The config.mean_values is None, zeros will be set for input 3!
W load_onnx: The config.std_values is None, ones will be set for input 3!
W load_onnx: The config.mean_values is None, zeros will be set for input 4!
W load_onnx: The config.std_values is None, ones will be set for input 4!
W load_onnx: The config.mean_values is None, zeros will be set for input 5!
W load_onnx: The config.std_values is None, ones will be set for input 5!
ONNX loaded. Building RKNN
[1]    113493 segmentation fault (core dumped)  python convert.py /media/2TB/marty/ChatRWKV/models/mixing_0.onnx
```

ONNX Graph hacking!

- Disable FP16 conversion
  - else face compiler bug. RKNN converts anyway
- Walk ONNX graph, find MatMuls
- Split graph
- Send a small, compute heavy graph to RKNN
- Keep the rest in ONNX

It works, but SLOW

- Very slow. ~500ms/token, 430M param
  - Expected, too much overhead



Attempt 2 - offloading MatMul from GGML

This actually works

saharNooby/rwkv.cpp

- Used GGML, C library for inference
- GGML runs LLaMA and Whisper
- Support quantizatrion down to 2 bits
- Rockchip fixed their MatMul C API in v1.5.2
- Compiler still crash

GGML hacking

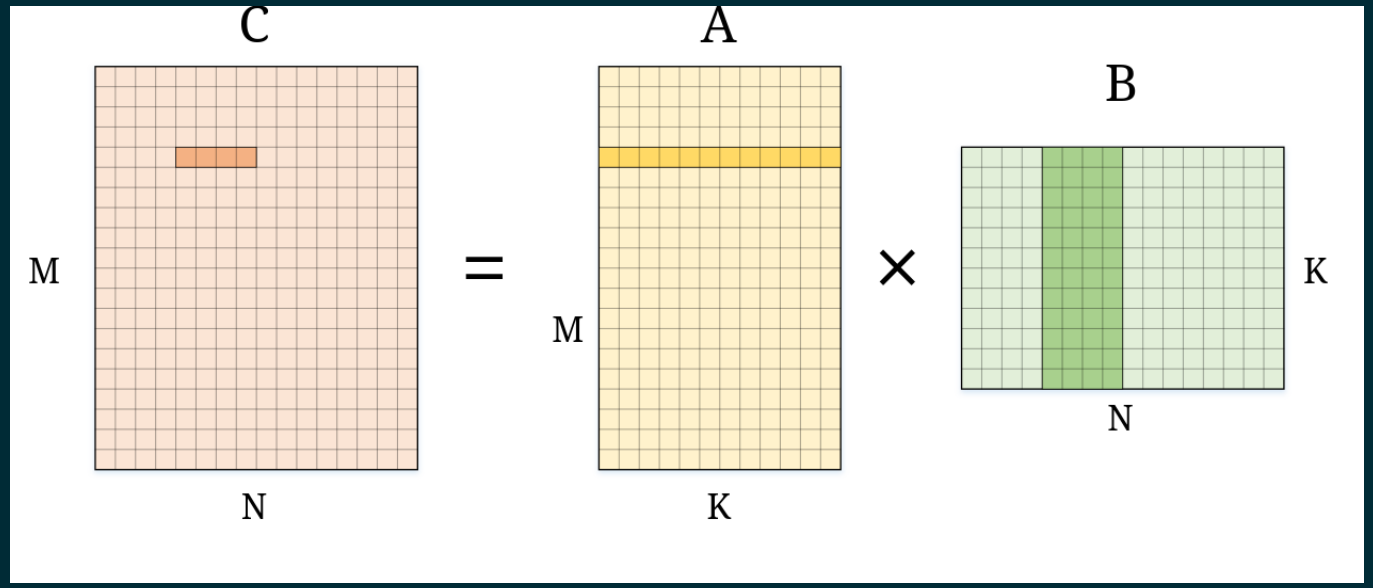
- Like Linux, moving target
- A day to learn how GGML works
- `ggml_compute_forward_mul_mat_f16_f32`
- Weight fp16, input fp32
- Allocate handles, memory and reorder during init

Works, but...

- Slower then CPU
  - CPU: 61ms
- NPU: 83ms/token (load 53/300%)
  - CPU load: 50%
  - 25% load in kernel

Using multi NPU

- RK3588 has 3 NPU cores
- Matmul can only use 1
- Manually split along the N axis into 2
- Then stich back together



Source: Hulalazz/A-_Guide_-to_Data_Sciecne_from_mathematics/tree/master

Faster..

- Now 76ms/token
- 2 cores, total load 106/300%
- Still slower then CPU

3 NPU cores + 1 CPU core

- 65ms/token
- Close. But not CPU speed yet.
- Unstable. Have bugs
- Sad

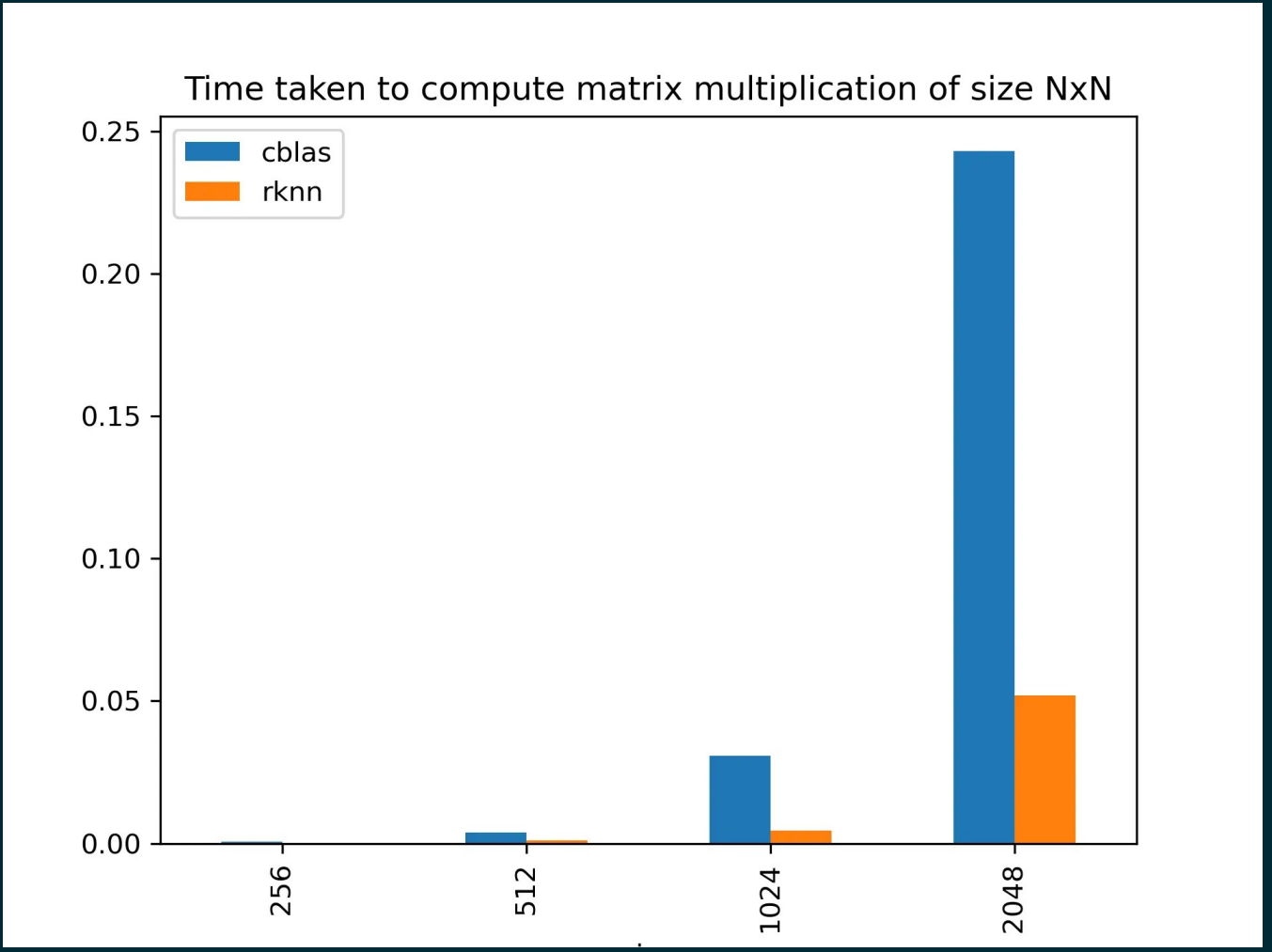
Tried every trick in the book

- Vectorize fp32 to fp16 conversion
- Fixing overhead from GGML hack
- Reduce system call
- Use 3 NPU cores instead of 2
- etc...

- Does not help

Benchmarking

- Should have benchmarked first
- What???????
- RKNN is extremely fast
- M = K = N (y axis)



GGML strike back

M = 1, K = 1024, N = 1024, mat B pre-transposed

- GGML: 0.1ms
- RKNN: 0.2ms
- Issue is M = 1. For large M, RKNN is faster

- NPU needs to be better at GEMV
- But GGML is optmized for this

Conclusion

RK3588 NPU

- As of SDK v1.5.2
- Good MatMul, bad at GEMV
- Driver too high latency/heavy
- Need larger K support to be usible in LLM
- SDK design flaws

If you just want any LLM to run

- Add your accelerator to GGML
- Only need to support MatMul
- Maybe LayerNorm
- Enough to run RWKV
- Or just have a good ONNX compiler

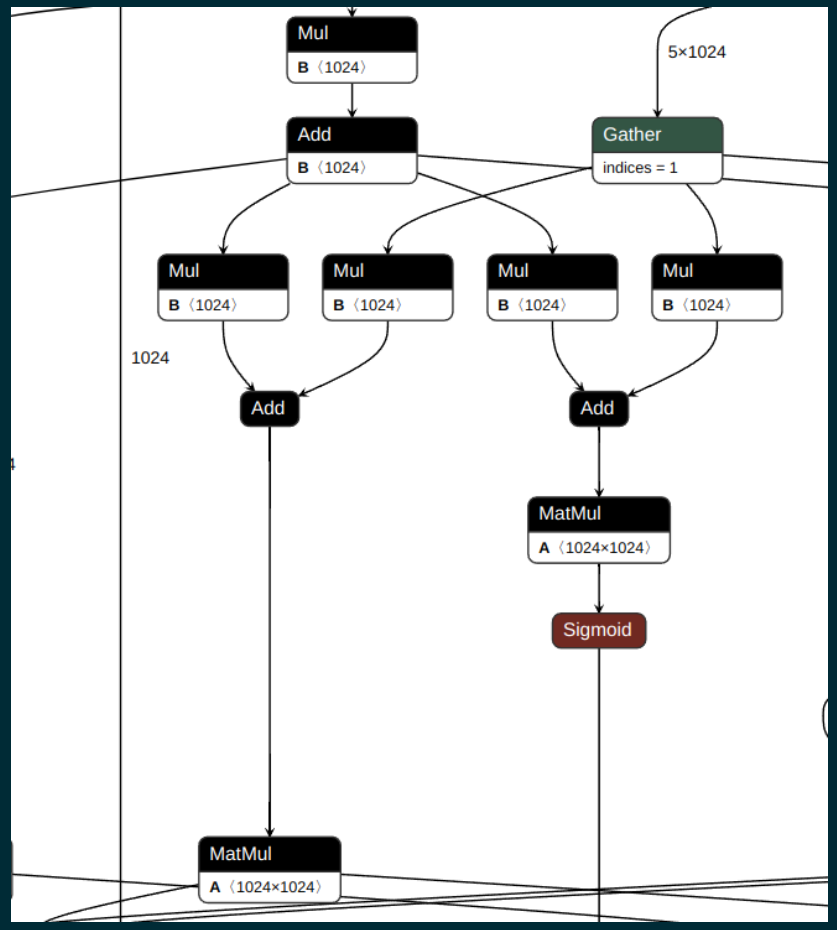
If you just want LLaMA to run

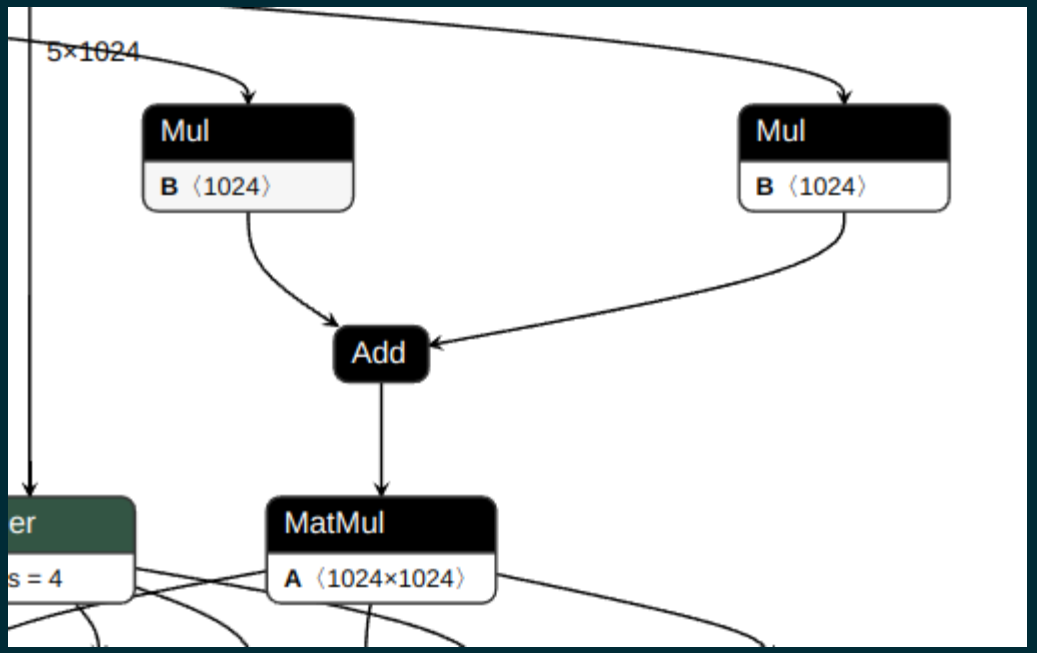
- Add accelerator to GGML
- Must support MatMul without reordering mat B
- Avoid transpose in attention
- Pratially MatMul must support K >= 4096

If you want RWKV to run fast

- Support parallel graph walk
- Unlikely full use HW on matmul K=2048
- Optimize GEMV
- Ideally K >= 8192 for large RWKV
- Hardware support for WKV operation
- Support unconventional and deep op fusion
- Mul → Add → Mul → Add → MatMul
- Store intermid on chip

These should be fusible





Features helpful to GGML

- Accelerator can access Vritual Memory
- Run multiple ops at the same time
- Supports fp32/fp16 multipling with fp16/int8
- Low latency driver/runtime
- Async runtime, able to wait for completion

Thank you

Backup slides

Single NPU core vs 2

GGML + RKNN 1 core

```
Loading 20B tokenizer
System info: AVX=0 AVX2=0 AVX512=0 FMA=0 NEON=1 ARM_FMA=1 F16C=0 FP16_VA=1 WASM_SIMD=0 BLAS=1 SSE3=0 VSX=0
Loading RWKV model
Processing 185 prompt tokens, may take a while
Processed in 17 s, 96 ms per token

Chat initialized! Your name is User. Write something and press Enter. Use \n to add line breaks to your message.
> User:
```

- GGML + RKNN 2 cores

```
Loading 20B tokenizer
System info: AVX=0 AVX2=0 AVX512=0 FMA=0 NEON=1 ARM_FMA=1 F16C=0 FP16_VA=1 WASM_SIMD=0 BLAS=1 SSE3=0 VSX=0
Loading RWKV model
Processing 185 prompt tokens, may take a while
Processed in 14 s, 76 ms per token

Chat initialized! Your name is User. Write something and press Enter. Use \n to add line breaks to your message.
> User:
```

3 NPU cores (split matrix into 4 pieces)

- GGML + 3 NPU cores
- 70ms

```
Loading 20B tokenizer
System info: AVX=0 AVX2=0 AVX512=0 FMA=0 NEON=1 ARM_FMA=1 F16C=0 FP16_VA=1 WASM_SIMD=0 BLAS=1 SSE3=0 VSX=0
Loading RWKV model
Processing 185 prompt tokens, may take a while
Processed in 13 s, 70 ms per token

Chat initialized! Your name is User. Write something and press Enter. Use \n to add line breaks to your message.
> User:
```