Project RP014: Optimizing Llama.cpp and GGML for RVV

Implementation Plan

- **Phase 1: VLEN-agnostic RVV Kernels**
  - **Floating-Point Kernels**
    - SIMD Mappings
    - High-priority Kernels
    - Activation Functions and other Utilities
    - Llamafile SGEMM
    - **Repacking GEMM and GEMV**
  - Quantization Kernels
    - Quantization/Dequantization
    - Vec Dot
    - Repack GEMM and GEMV
    - Llamafile SGEMM Quantization
- Phase 2: Dynamic Dispatching
  - Integration
- Phase 3: Benchmarking and Testing Software
  - Development
  - Integration with Mainline Llama.cpp

Floating-Point Improvements

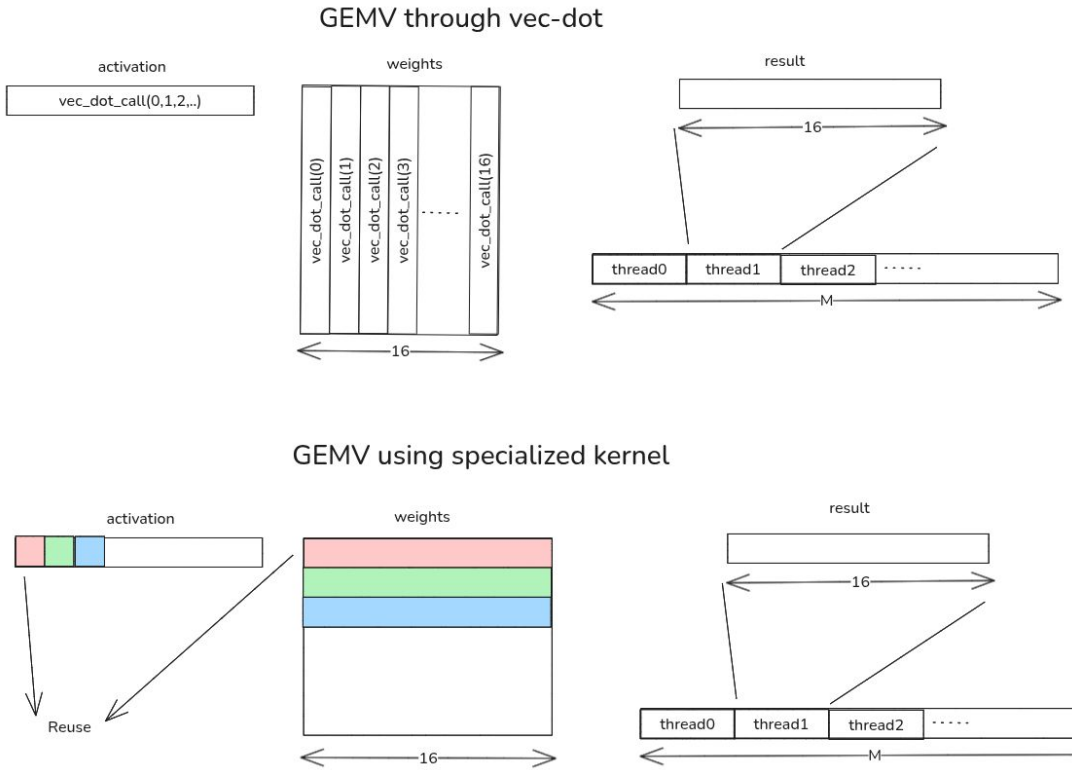
Improvement in GEMV of decode phase in FP16 and FP32:

Data Reuse:

- Currently GEMV is performed through multiple vecdot kernel calls
- We lose the inherent data-reuse benefits of GEMV, resulting in poor cache utilization and lower efficiency.

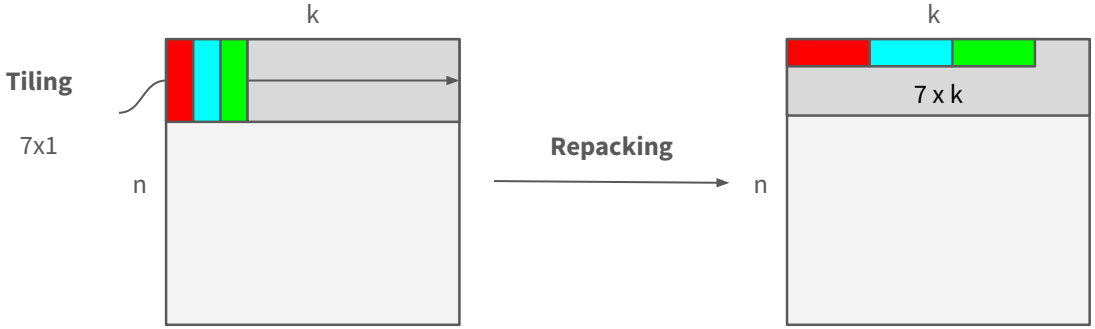
Repacking:

- We will also experiment with repacking of weights to reduce misses at different levels of memory hierarchy

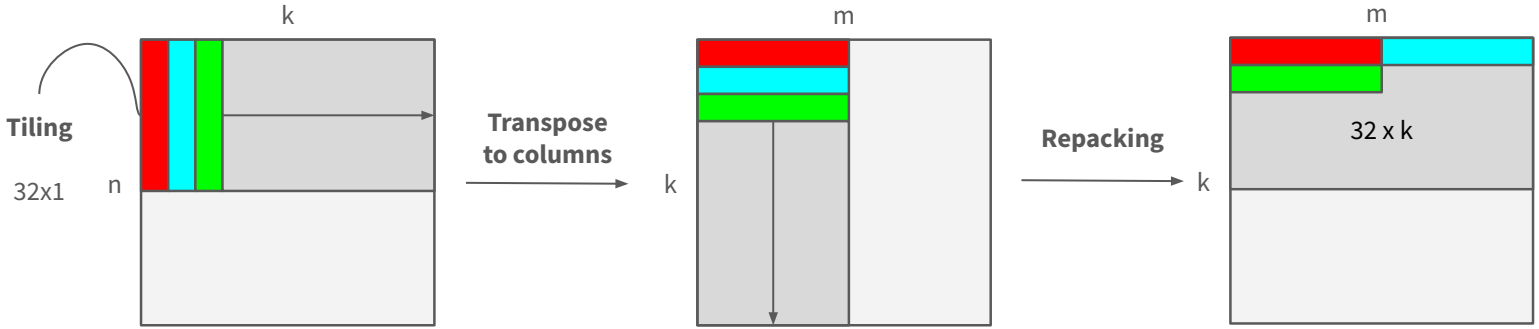


Repacking Layout

Activations



Weights

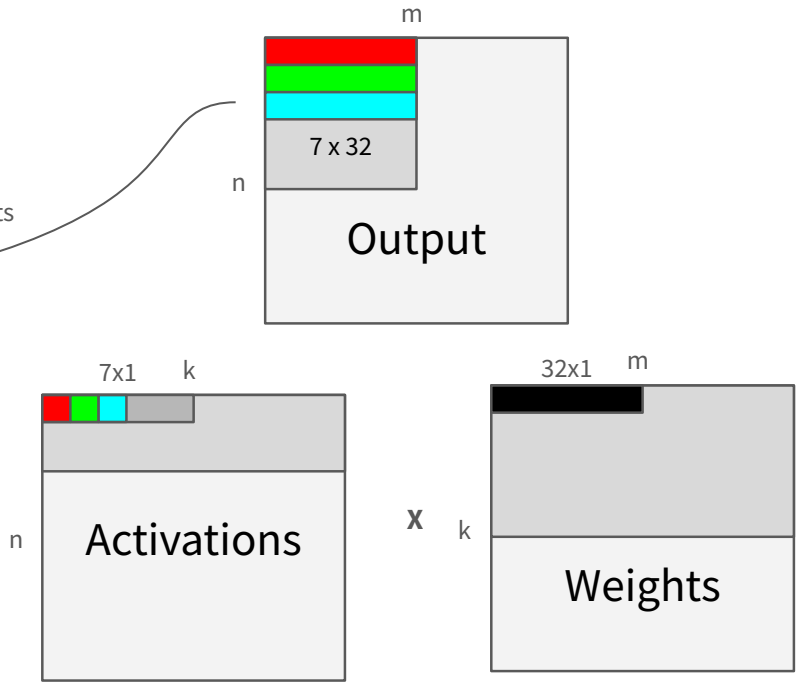


Repacked GEMM (Implementation)

```
void ggml_gemm_f16_7x32_f16(int n, float *s, size_t bs, const void * GGML_RESTRICT vx,
const void * G) {
    const block_f16_7x1 * a_ptr = ...  // Activations
    const block_f16_32x1 * b_ptr = ... // Weights
    // Accumulators
    vfloat32m4_t sumf_0 = __riscv_vfmv_v_f_f32m4(0.0f, 32);
    vfloat32m4_t sumf_1 = __riscv_vfmv_v_f_f32m4(0.0f, 32);
    vfloat32m4_t sumf_2 = __riscv_vfmv_v_f_f32m4(0.0f, 32);
    vfloat32m4_t sumf_3 = __riscv_vfmv_v_f_f32m4(0.0f, 32);
    vfloat32m4_t sumf_4 = __riscv_vfmv_v_f_f32m4(0.0f, 32);
    vfloat32m4_t sumf_5 = __riscv_vfmv_v_f_f32m4(0.0f, 32);
    vfloat32m4_t sumf_6 = __riscv_vfmv_v_f_f32m4(0.0f, 32);
    for (int l = 0; l < nb; l++) {
        vfloat16m2_t b_0 = __riscv_vle16_v_f16m2(&b_ptr[l].d[0], 32);
        sumf_0 = __riscv_vfwmacc_vf_f32m4(sumf_0, a_ptr[l].d[0], b_0, 32);
        sumf_1 = __riscv_vfwmacc_vf_f32m4(sumf_1, a_ptr[l].d[1], b_0, 32);
        sumf_2 = __riscv_vfwmacc_vf_f32m4(sumf_2, a_ptr[l].d[2], b_0, 32);
        sumf_3 = __riscv_vfwmacc_vf_f32m4(sumf_3, a_ptr[l].d[3], b_0, 32);
        sumf_4 = __riscv_vfwmacc_vf_f32m4(sumf_4, a_ptr[l].d[4], b_0, 32);
        sumf_5 = __riscv_vfwmacc_vf_f32m4(sumf_5, a_ptr[l].d[5], b_0, 32);
        sumf_6 = __riscv_vfwmacc_vf_f32m4(sumf_6, a_ptr[l].d[6], b_0, 32);
    }
    __riscv_vse32_v_f32m4(&s[(y * 7 + 0) * bs + x * ncols_interleaved], sumf_0, 32);
    __riscv_vse32_v_f32m4(&s[(y * 7 + 1) * bs + x * ncols_interleaved], sumf_1, 32);
    __riscv_vse32_v_f32m4(&s[(y * 7 + 2) * bs + x * ncols_interleaved], sumf_2, 32);
    __riscv_vse32_v_f32m4(&s[(y * 7 + 3) * bs + x * ncols_interleaved], sumf_3, 32);
    __riscv_vse32_v_f32m4(&s[(y * 7 + 4) * bs + x * ncols_interleaved], sumf_4, 32);
    __riscv_vse32_v_f32m4(&s[(y * 7 + 5) * bs + x * ncols_interleaved], sumf_5, 32);
    __riscv_vse32_v_f32m4(&s[(y * 7 + 6) * bs + x * ncols_interleaved], sumf_6, 32);
```

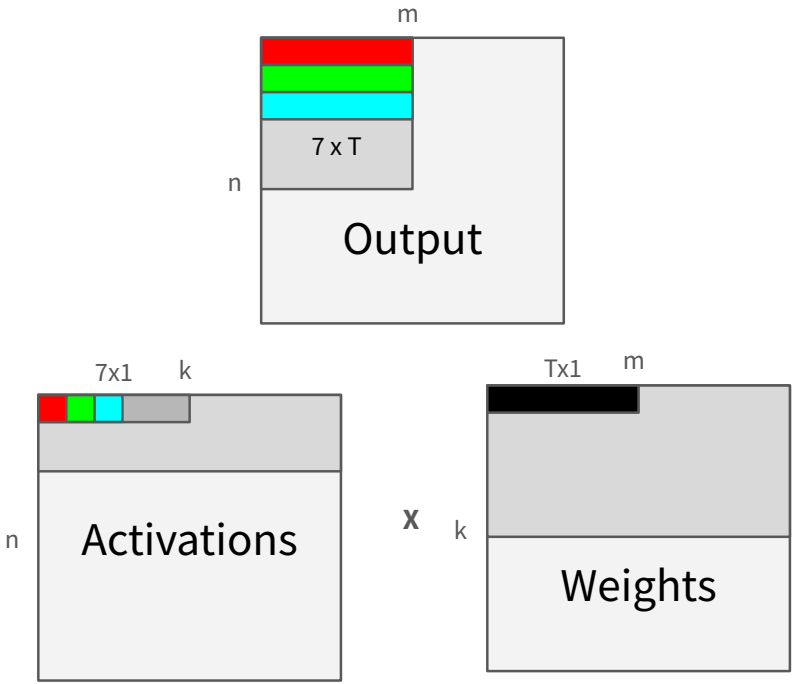
7 accumulators

Each accumulates 32 results



Repacked Outer Product - Tiling

VLEN	Tiling	Tile Sizes		
		LHS	RHS	OUT
128	7, 16, 1	7x1	16x1	7x16
256	7, 32, 1	7x1	32x1	7x32
512	7, 64, 1	7x1	64x1	7x64
1024	7, 128, 1	7x1	128x1	7x128



Repacked GEMM (Row Sizes)

Added option to select row size

```
// gemm
template <typename BLOC_TYPE, int64_t NB_ROWS, int64_t INTER_SIZE, int64_t NB_COLS, ggml_type PARAM_TYPE>
void gemm(int, float *, size_t, const void *, const void *, int, int);

template <> void gemm<block_q4_0, 4, 4, 4, GGML_TYPE_Q8_0>(int n, float * s, size_t bs, const void * vx, const
void * vy, int nr, int nc) {
   ggml_gemm_q4_0_4x4_q8_0 (n, s, bs, vx, vy, nr, nc);
}

template <> void gemm<ggml_half, 7, 1, 32, GGML_TYPE_F16>(...) {
   ggml_gemm_f16_7x1x32_f16 (n, s, bs, vx, vy, nr, nc);
}
```

Repacked Outer Product - Prompt Processing

BananaPI BPI-F3 (Spacemit X60)			
Model	Prompt Size	Tokens / second (avg)	
		Repack GEMM 7x32	Vec Dot
Tinyllama F16 1.1B	32	16.72	8.42
Tinyllama F16 1.1B	64	21.55	7.57
Tinyllama F16 1.1B	128	21.2	8.78
Tinyllama F16 1.1B	256	21.82	8.57
Tinyllama F16 1.1B	512	21.81	8.68

Repacked Outer Product - Prompt Processing

BananaPI BPI-F3 (Spacemit X60)			
Model	Prompt Size	Tokens / second (avg)	
		Repack GEMM 7x32	Vec Dot
Tinyllama F32 1.1B	32	7.13	3.75
Tinyllama F32 1.1B	64	10.76	3.74
Tinyllama F32 1.1B	128	10.86	3.73
Tinyllama F32 1.1B	256	10.94	3.68
Tinyllama F32 1.1B	512	11.12	3.79

Repacked Outer Product - Prompt Processing

BananaPI BPI-F3 (Spacemit X60)			
Model	Prompt Size	Tokens / second (avg)	
		Repack GEMM 7x32	Vec Dot
BERT Large Uncased F16	32	47.33	27.33
BERT Large Uncased F16	64	51.01	30.01
BERT Large Uncased F16	128	57.64	30.80
BERT Large Uncased F16	256	50.01	28.01
BERT Large Uncased F16	512	47.16	24.26

Repacked Outer Product - Prompt Processing

BananaPI BPI-F3 (Spacemit X60)			
Model	Prompt Size	Tokens / second (avg)	
		Repack GEMM 7x32	Vec Dot
BERT Large Uncased F32	32	26.54	11.08
BERT Large Uncased F32	64	36.51	11.35
BERT Large Uncased F32	128	36.26	11.16
BERT Large Uncased F32	256	33.62	11.95
BERT Large Uncased F32	512	30.60	11.18

Repacked Outer Product - Decode

Model	Decode Size (Prompt=32)	Token / seconds	
		Repack GEMV	Vec Dot
		1x32	
Tinyllama F16 1.1B	10	3.37	3.11
Tinyllama F16 1.1B	16	3.29	3.45
Tinyllama F16 1.1B	32	3.12	3.25
Tinyllama F16 1.1B	64	3.23	3.27
Tinyllama F16 1.1B	100	3.04	3.15
Tinyllama F16 1.1B	128	3.09	3.2
Tinyllama F16 1.1B	256	3.15	3.19

Repacked Outer Product - Decode

Model	Decode Size (Prompt=32)	Token / seconds	
		Repack GEMV	Vec Dot
		1x32	
Tinyllama F32 1.1B	10	1.66	1.74
Tinyllama F32 1.1B	16	1.73	1.63
Tinyllama F32 1.1B	32	1.81	1.68
Tinyllama F32 1.1B	64	1.61	1.69
Tinyllama F32 1.1B	100	1.72	1.75
Tinyllama F32 1.1B	128	1.76	1.72
Tinyllama F32 1.1B	256	1.75	1.69