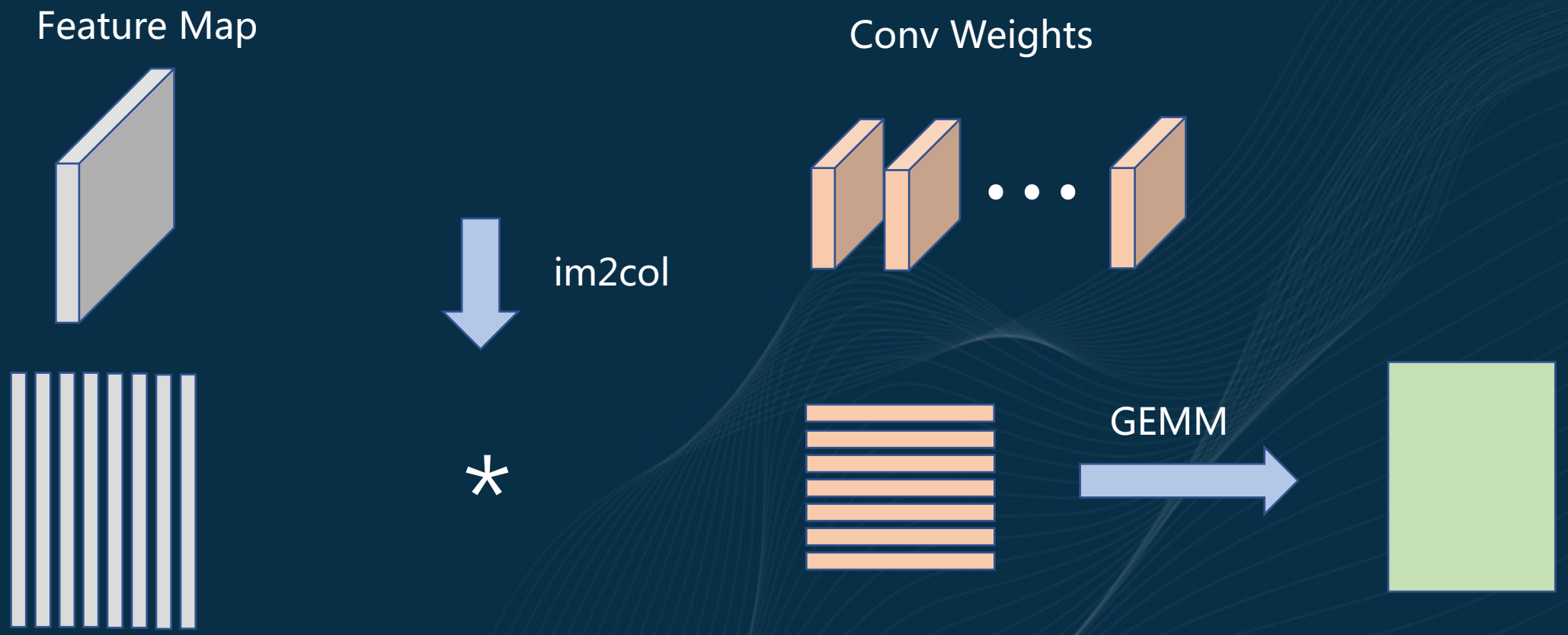
StarFive 赛昉科技

RISC-V平台下的高性能GEMM

丘 庆　吴志刚
赛昉科技

GEMM及其应用

- GEMM中文全称是通用矩阵乘法
  - 求解$\alpha AB + \beta C$运算，OpenBLAS等线性代数运算库有标准接口定义及具体实现
  - 广泛应用于科学计算领域（航空航天、无人驾驶、人工智能等）



行乘列法则实现GEMM（baseline）

➢缺点：

- 频繁Cache miss
- 数据重复利用率低
- 不利于向量化

➢解决办法：

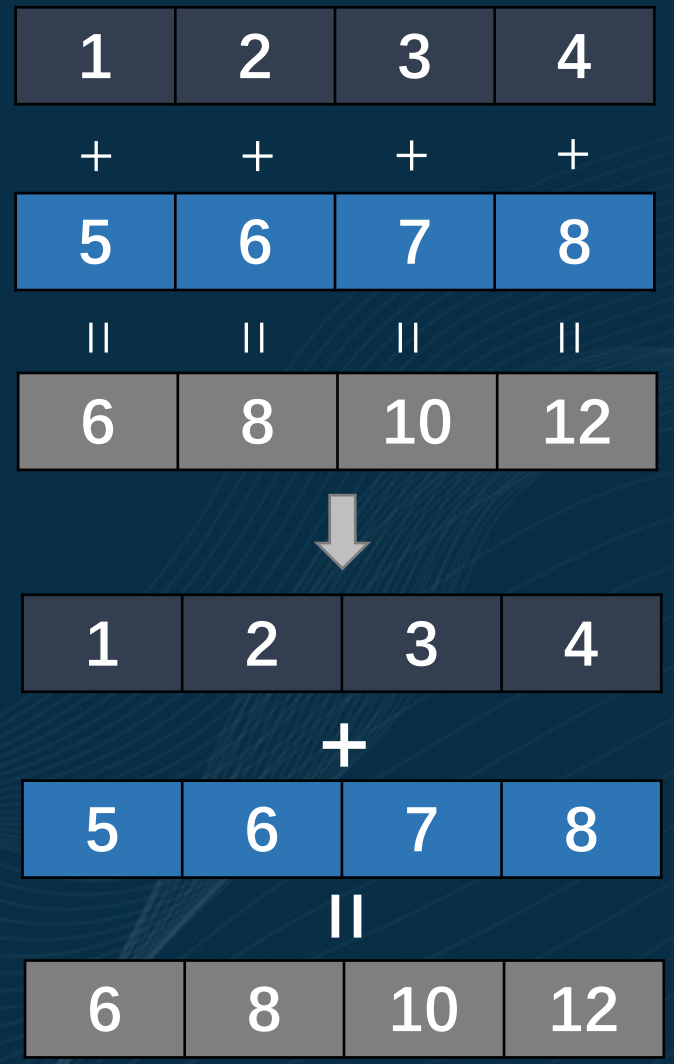
- 矩阵分块、Packing
- 向量化

```
for (i = 0; i < M; i++)
{
    for (j = 0; j < N; j++)
    {
        tmp = 0.0;
        for (k = 0; k < K; k++)
        {
            tmp += A[i][k] * B[k][j];
        }
        C[i][j] = alpha * tmp + beta * C[i][j];
}
```

➢ 赛昉科技自主研发的高性能RISC-V处理器昉·天枢全面支持RVV 1.0，这里基于该处理器实现RISC-V平台下的高性能GEMM

RISC-V Vector扩展指令介绍

- 单条指令可并行处理一批数据
- 支持动态调整向量长度
  - 相比传统SIMD，使用起来更高效、节能
  - 有利于减少代码体积
  - 循环展开更简洁
- 统一的标准向量扩展ISA
  - 大大简化跨向量处理器软件开发流程
- 相同的代码可以在任意向量长度处理器上运行
  - 代码迁移以及维护更加方便，无需重新开发或调整



昉·天枢——高性能RISC-V处理器

高性能RISC-V处理器

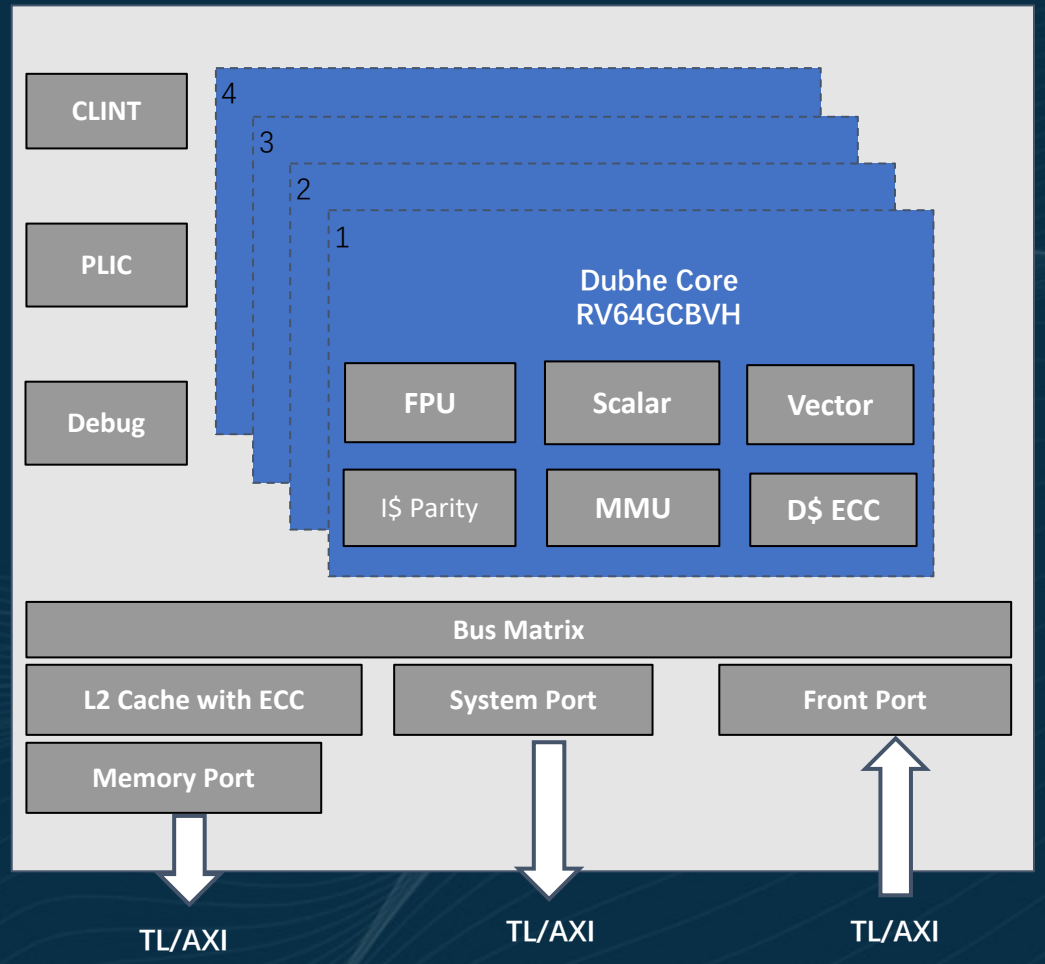
- 带向量扩展的64位深度乱序超标量处理器
- RISC-V ISA 扩展：RV64GCBVH
- 支持单精度、双精度 FPU
- 支持多核与内存
- 完全支持Hypervisor 扩展
- 完全支持位操作（“B”）扩展（Zba, Zbb, Zbc, and Zbs）

完全支持Vector 1.0 ISA扩展

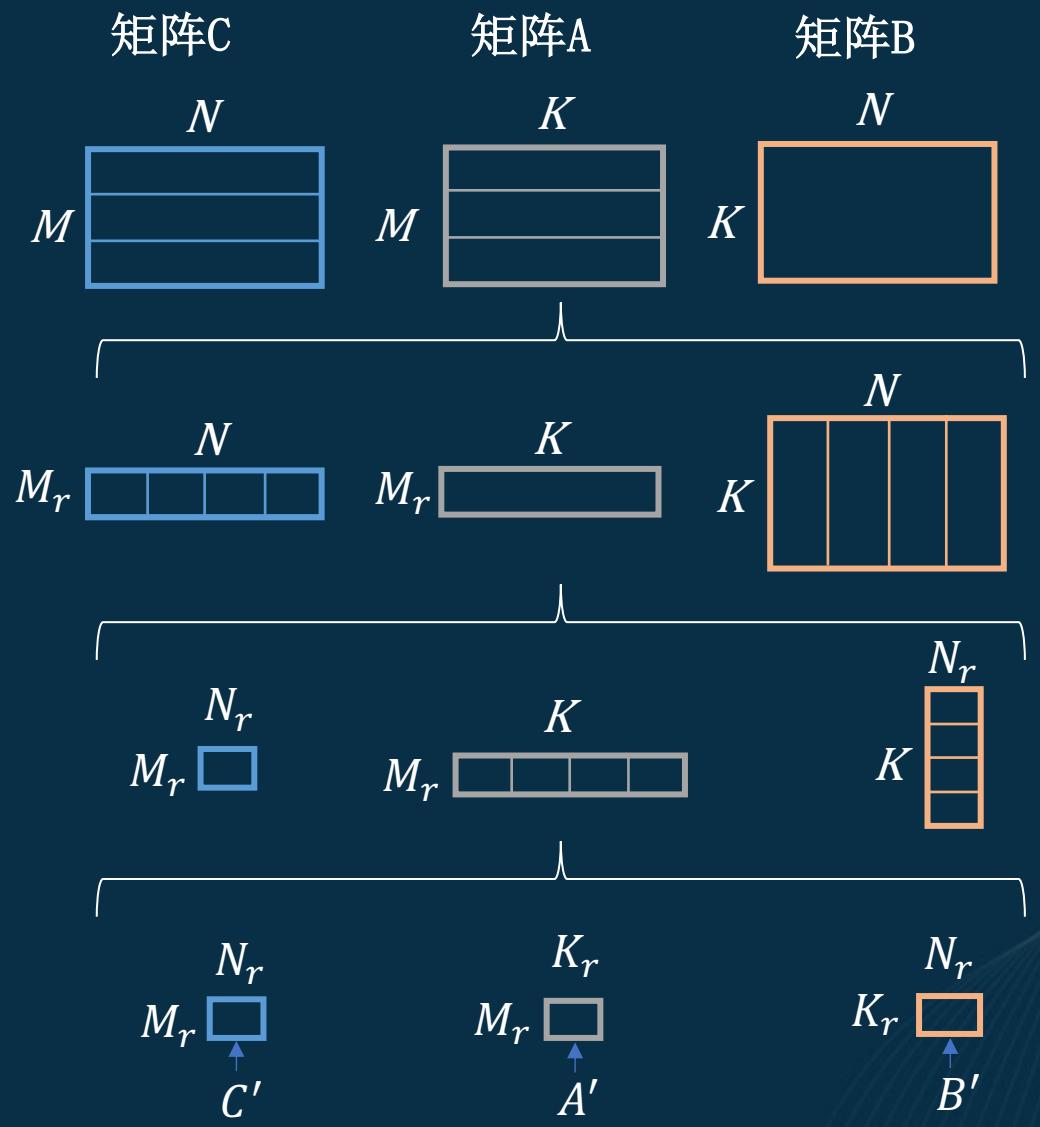
- 完全支持向量寄存器分组（LMUL）
- 支持的数据类型：整型， 浮点数以及定点数
- 数据类型为8b至64b时向量运算可达到128b/cycle（数据通路宽度为128b的情况下）
- 向量ALU以及数据通路宽度为128bits（默认）

内存子系统

- MMU = SV48 （选项：SV39，SV48）
- L1 Cache: 64KB I-Cache with parity; 64KB D-Cache with ECC
- L2 Cache: 2 MB with ECC



高性能GEMM实现(1):矩阵分块



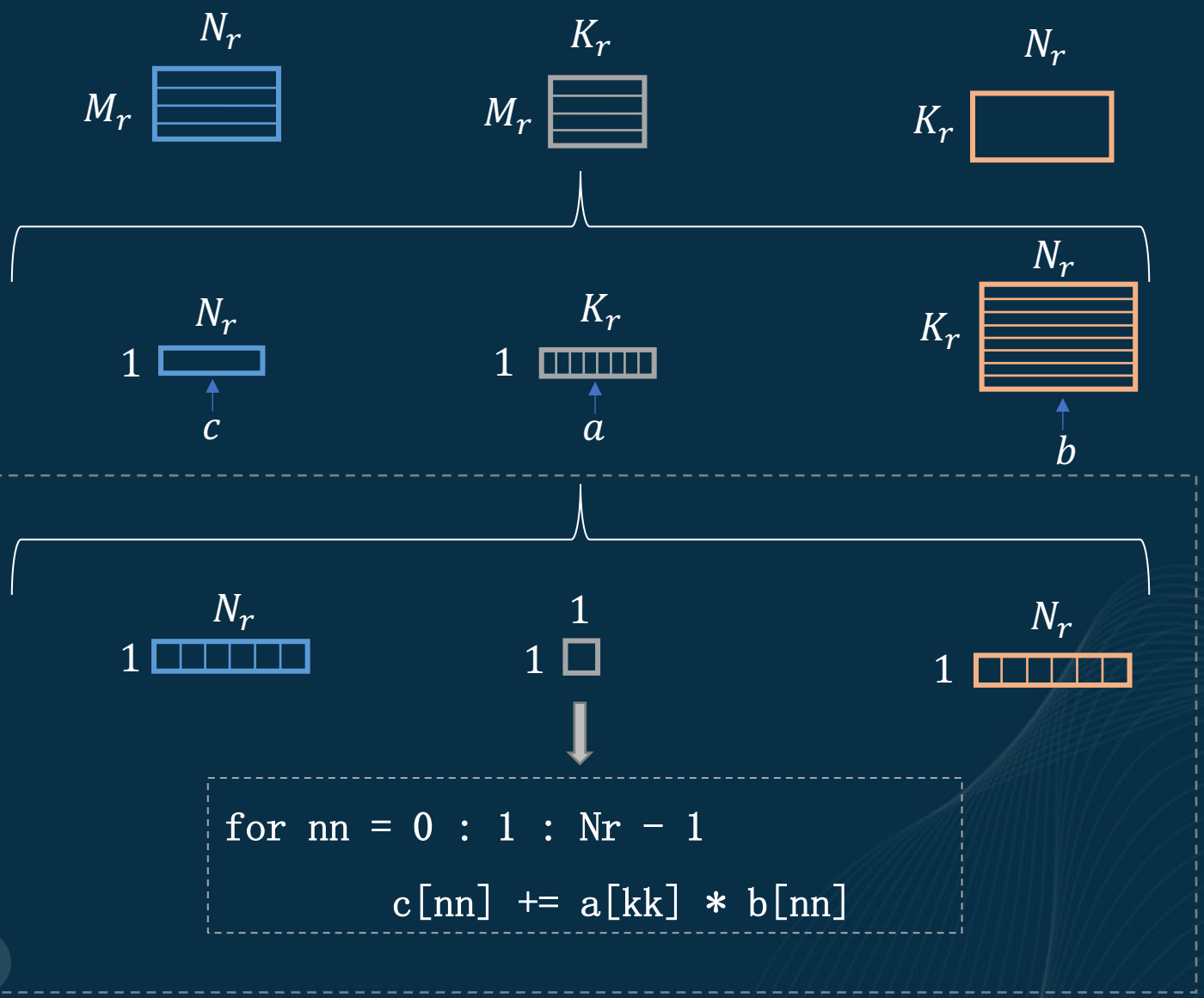
$for\ ii = 0 : M_r : M - 1$ 以M_r为步长对M划分

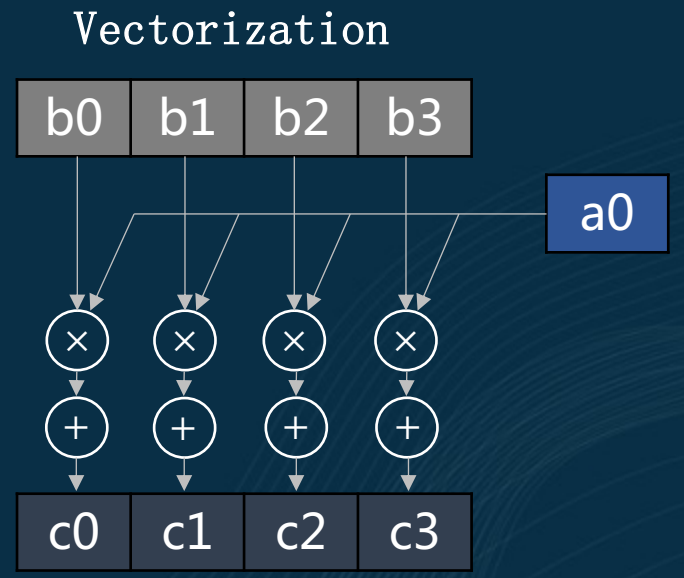
$for\ jj = 0 : N_r : N - 1$ 以N_r为步长对N划分

$for\ kk = 0 : K_r : K - 1$ 以K_r为步长对K划分

$C' \ += \ A' \ * \ B'$ 迭代计算子矩阵C'

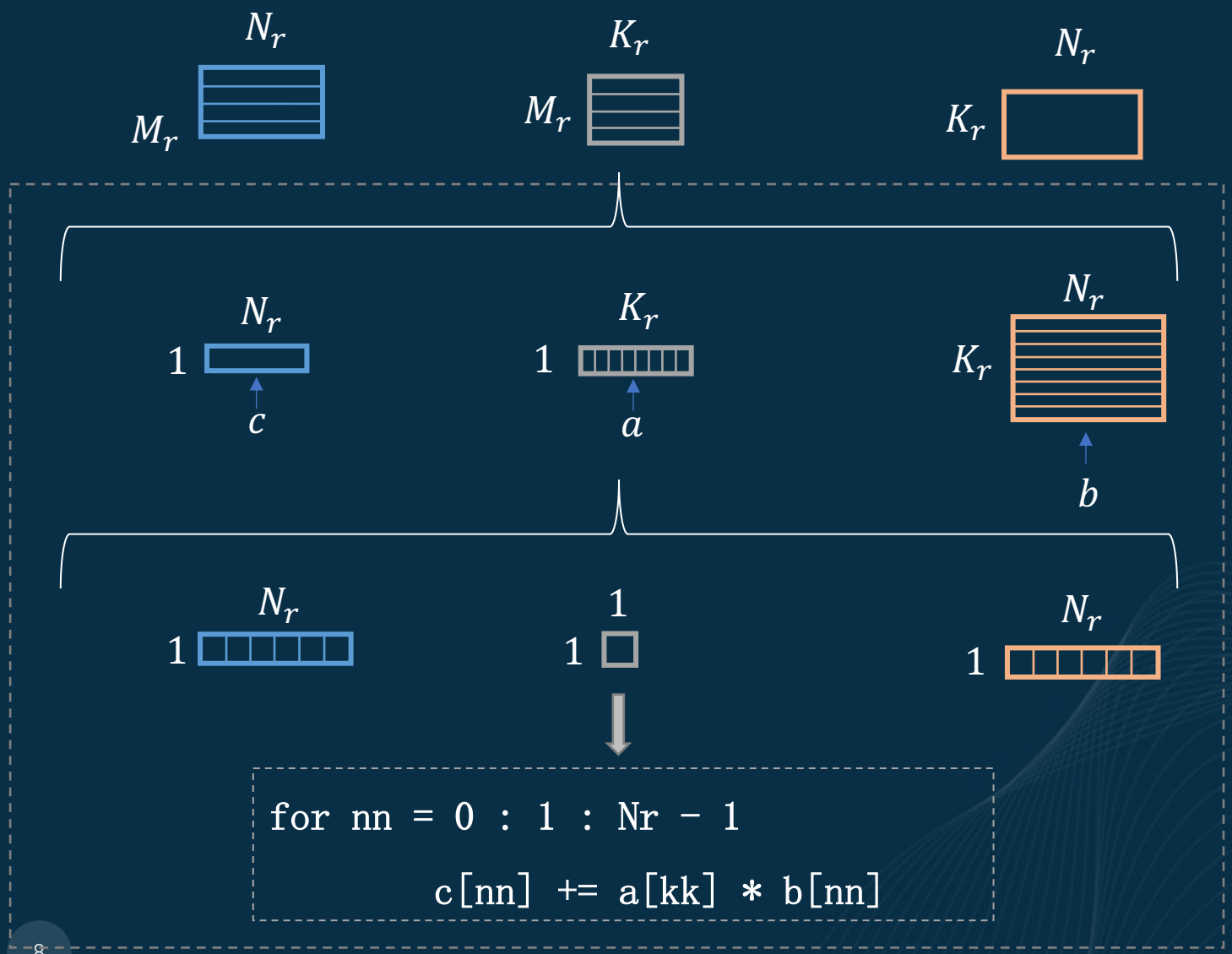
高性能GEMM实现(2):向量化






```
//v16 for c, ft0 for a, v0 for b
//pa->a, pb->b, pc->c
for kk = 0 : 1 : Kr - 1
        vle32.v v0, (%[pb])
        flw     ft0, (%[pa])
        addi    %[pa], %[pa], 4
        addi    %[pb], %[pb], %[ldb]
        vfmacc.vf v16, ft0, v0
end for
vse32.v v16, (%[pc])
```

高性能GEMM实现(3):循环展开



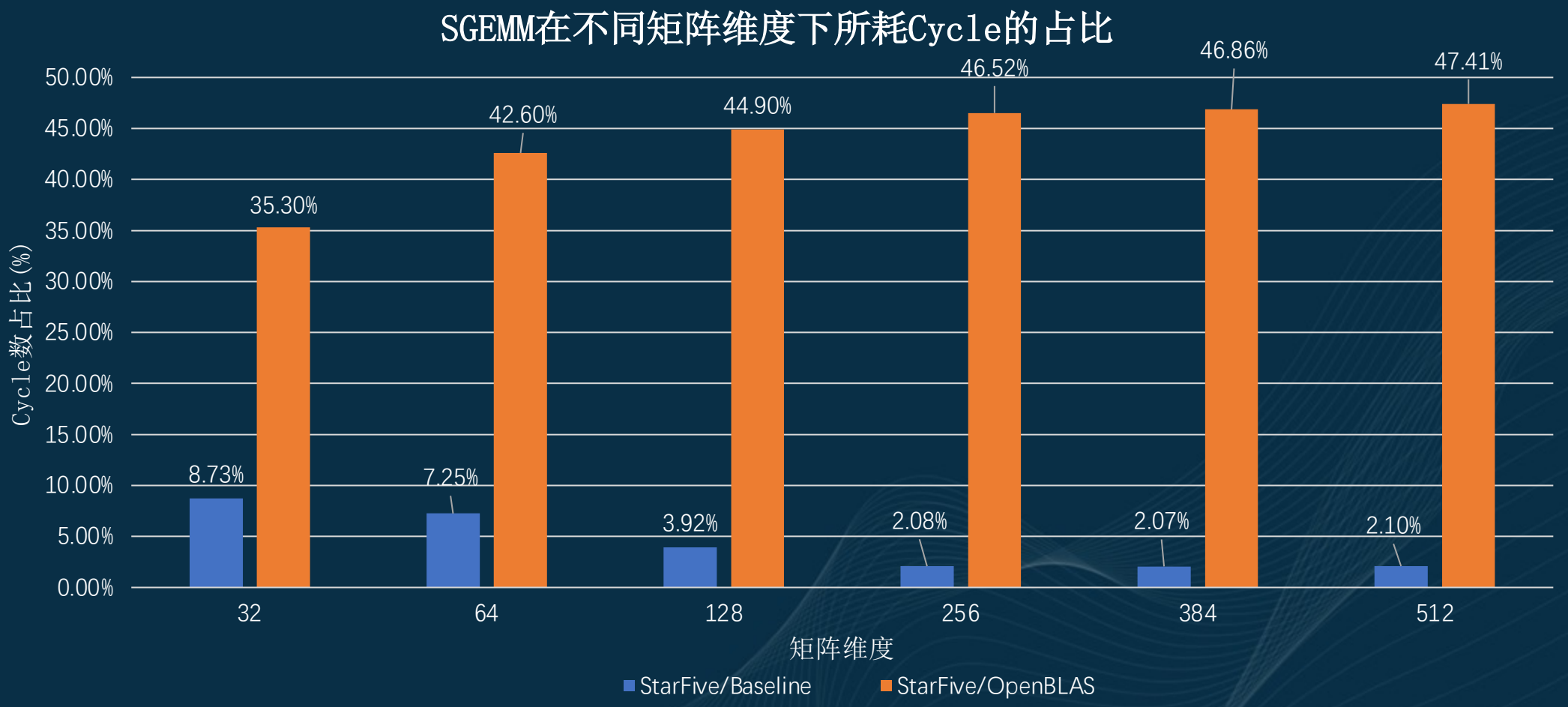
Loop Unrolling

```
//v16-19 for c,ft0-3 for a,v0-3for b
//pa->a, pb1-3->b, pc1-3->c
for kk = 0 : 1 : Kr - 1
        vle32.v v0, (%[pb1])
        flw     ft0, (%[pa])
        addi    %[pa], %[pa], 4
        vle32.v  v1, (%[pb2])
        flw      ft1, (%[pa])
        addi    %[pa], %[pa], 4
        vle32.v  v2, (%[pb3])
        flw      ft2, (%[pa])
        addi    %[pa], %[pa], 4
        vle32.v  v3, (%[pb4])
        flw      ft3, (%[pa])
        addi    %[pa], %[pa], 4
        vfmacc.vf v16, ft0, v0
        vfmacc.vf v17, ft1, v1
        vfmacc.vf v18, ft2, v2
        vfmacc.vf v19, ft3, v3
end for
vse32.v v16, (%[pc1])
vse32.v v17, (%[pc2])
vse32.v v18, (%[pc3])
vse32.v v19, (%[pc4])
```

高性能GEMM实现(4):其他优化策略

- lmul设为更大的值(如2,4,8)，让相邻两个或多个向量寄存器并为一组,增大向量长度，提高数据级并行度
- 软件层面的prefetch，使用两组寄存器，在一组参与计算的同时，另一组预取下一阶段数据
- 合理调整指令顺序，进一步降低指令间依赖性
- 压缩循环中非必要的指令数目

结果展示


SGEMM在不同矩阵维度下所耗Cycle的占比
Cycle数占比(%)
50.00%
45.00%
40.00%
35.00%
30.00%
25.00%
20.00%
15.00%
10.00%
5.00%
0.00%
8.73%
35.30%
7.25%
42.60%
3.92%
44.90%
2.08%
46.52%
2.07%
46.86%
2.10%
47.41%
32
64
128
256
384
512
矩阵维度
StarFive/Baseline
StarFive/OpenBLAS

总结与展望

总结

- 优化后的SGEMM比OpenBLAS中的实现快一倍以上
- 优化后的SGEMM速度接近Baseline速度的50倍
- 软件或算法的优化加速很有必要
- Vector指令在优化加速方面至关重要

展望

- 利用天枢的硬件prefetch指令进一步提升GEMM性能

StarFive
赛昉科技

THANK YOU