# 2023A3-算子实现和性能优化

队伍：arcsysu1

队长：郭天宇

# 队伍介绍

arcsysu1 成员：郭天宇——中山大学计算机学院arcSYSU团队成员，导师张献伟，团队研究的主要方向为GPU、编译、存储系统、高性能计算和仿真/建模/分析。

## Education

**Sun Yat-sen University**, Guangzhou, China — 2022 – 2025 (expected)
*Master student* in Computer Science (CS)

**Xidian University**, Shaanxi, China — 2018 – 2022
*B.S.* in Computer Science (CS)

## Experience and Projects

**Fine-grained cache Management in GPU** — 2023
To achieve accurate/fast objection recognition and detection, various machine learning models arise. After characterizing those models, we achieve fine-grained cache management to speed up the inference procedure through modifying cache control bits in the binary level before the launch of each kernel.

**Shared Memory Expansion to Improve GPU TLP** — 2023
GPU thread level parallelism tends to be bounded by shared memory or registers. To expose the potential performance restricted by hardware resources, We purpose to harness the abundant last level cache(LLC) to be shared memory extension. Experiments in Accel-Sim shows that it can greatly improve TLP and reduce the execution time.

**Optimize GEMM step by step** — 2023
"GEMM MMA" first implementates a naive kernel of GEMM by CUDA mma.sync and then optimize it step by step(using vectorization, asynchronous copy, conflict-free shared memory access, consolidated memory access and so on), which achieves above 70% of peak performance relative to CUTLASS in the final version.

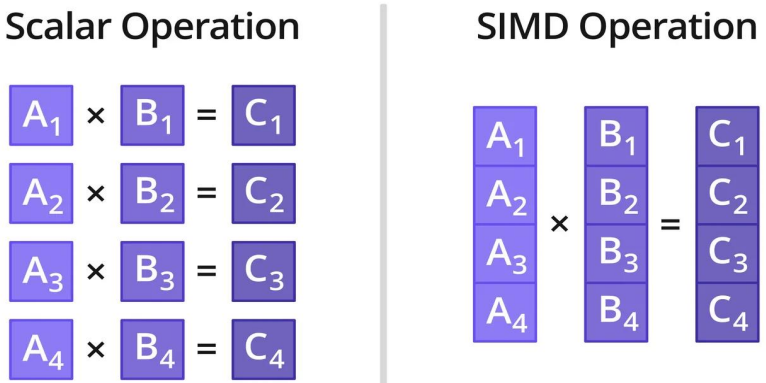
**Teaching Assistant of "SYSU-DCS3013 : Computer Architecture"** — 2022
Release "SYSU-ARCH LAB" which focuses on simulators(gem5, GPGPU-Sim and Accel-Sim).

**Design PTX-EMU** — 2022
"PTX-EMU" is an simulator for NVIDIA's virtual instruction set PTX.
You can use it to generate image by simulating rendering program.

**Design CNN framework on CPU and GPU** — 2022
"CovNN" is a CNN framework support on CPU and GPU(built on CUDNN).
To validate its availability, CNNs are built to solve MNIST or CIFAR-10 training on GPU and achieve 98% or 70% accuracy respectively.

# 背景介绍

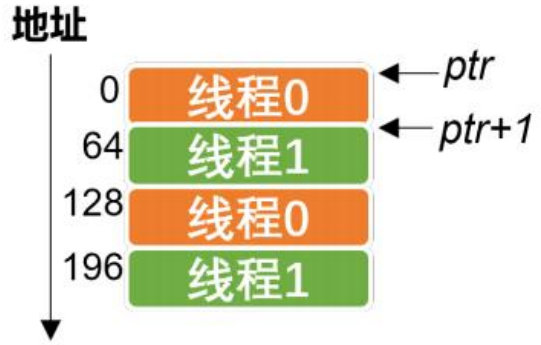
SIMD (Single Instruction, Multiple Data) 通常也被称为“单指令多数据”，是一种较为常见的并行计算技术。它能够同时对多个数据元素执行相同的操作，从而提高程序的执行效率。




SIMD向量处理器模型

GCU中的SIMD编程

- 数据类型v16f32，代表16个单精度浮点数
- 多线程SIMD访存，竞赛平台采用两线程硬件
- SIMD中的分支分歧与GCU的掩码操作
- SIMD内置函数接口




两线程交替处理数据

# 背景介绍

向量化方式：

- 手动向量化 —— 通过内嵌汇编或编译器提供的intrinsic函数来添加SIMD指令
- 自动向量化 —— 利用编译器分析串行程序中控制流和数据流的特征，识别程序中可以向量执行的部分，将标量语句自动转换为相应的SIMD向量语句。默认情况下，GCC、ICC或AOCC/LLVM编译器中都启用了部分自动向量化功能，可以对一些简单循环体(无较多条件语句、函数调用，无嵌套循环)实现自动向量化操作。

Intel高级向量扩展：

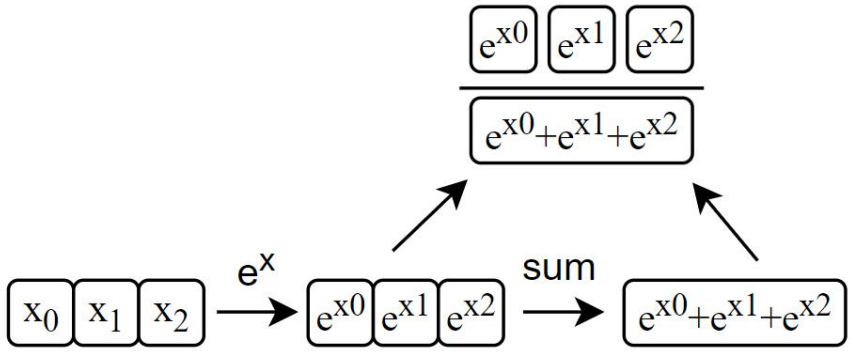
| 位宽(bit) | 64 | 128 | 256 | 512 |
|---|---|---|---|---|
| 指令集 | MMX | SSE | AVX/AVX2 | AVX512 |

# Softmax算子(90.95分)

Softmax算子计算公式如下：

$$\mathrm{Softmax}(x_i) = \frac{e^{x_i}}{\sum_{c=1}^{C} e^{x_c}}$$

其中$x_i$代表第$i$个元素，$C$代表这组数值的元素个数




Softmax计算流程：

1. 对所有输入数值取的指数次方
2. 完成对步骤1得到的中间结果累加求和
3. 将步骤1得到中间结果除以步骤2得到的中间结果

```
for (int i = 0; i < height * width; i++) {
  input[i] = expf(input[i]);
}
-----------------------------------------------------------------
int k;
for (k = 0; k + 32 <= height * width; k += 32) {
  reinterpret_cast<v16f32 *>(&input[k])[0] =
    __s20v_exp_f32__(reinterpret_cast<v16f32 *>(&input[k])[0]);
}
for (; k < height * width; k++) {
  input[k] = expf(input[k]);
}
```

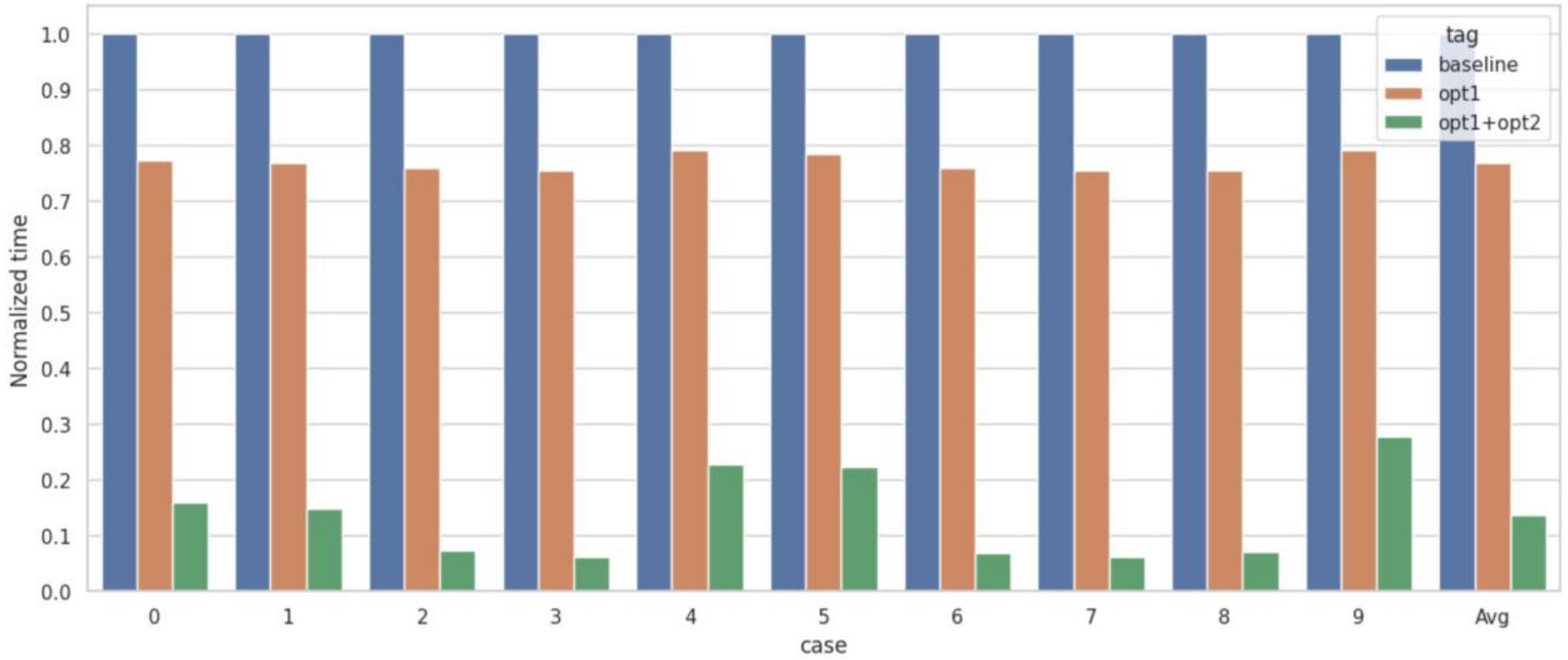
Softmax 优化1（上：优化前 下：优化后）

```
for (int h = 0; h < height; h++) {
  float acc = 0;
  for (int w = 0; w < width; w++) {
    acc += input[h * width + w];
  }
  for (int w = 0; w < width; w++) {
    output[h * width + w] =
      input[h * width + w] / acc;
  }
}
```

```
for (int h = 0; h < height; h++) {
  float acc = 0;
  int w;
  for (w = 0; w + 32 <= width; w += 32) {
    for (int j = 0; j < 32; j++) {
      acc += input[h * width + w + j];
    }
  }
  for (; w < width; w++) {
    acc += input[h * width + w];
  }
  float t = 1 / acc;
  for (w = 0; w + 32 <= width; w += 32) {
    for (int j = 0; j < 32; j++) {
      output[h * width + w + j] =
        input[h * width + w + j] * t;
    }
  }
  for (; w < width; w++) {
    output[h * width + w] =
      input[h * width + w] * t;
  }
}
```

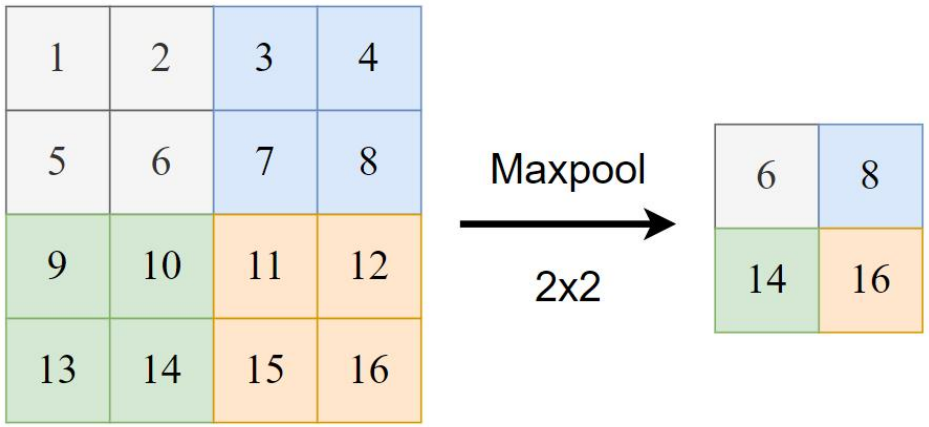
Softmax 优化2（左：优化前 右：优化后）

# Softmax算子(90.95分)

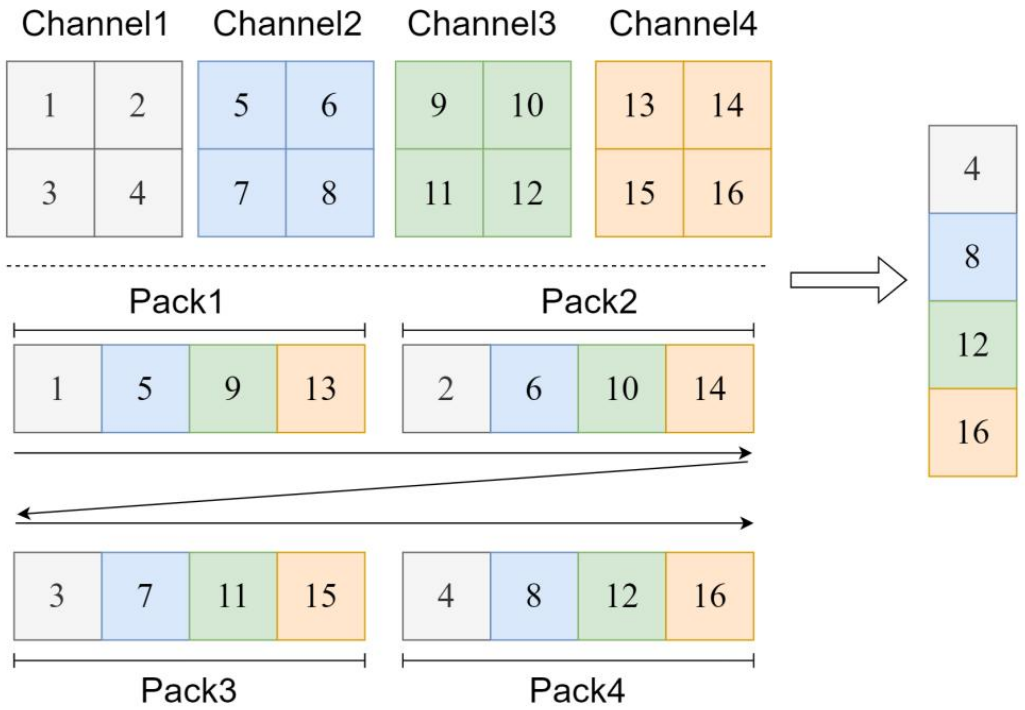



上图展示了Softmax算子分别在优化1（opt1）以及优化1加优化2（opt1+opt2）版本相对于baseline版本的归一化运行时间的变化。opt1平均实现1.30倍加速比，opt1+opt2平均实现了7.30倍加速比，由此可见自动向量化有着很大的优化潜力。

# Maxpooling2d(99.76分)

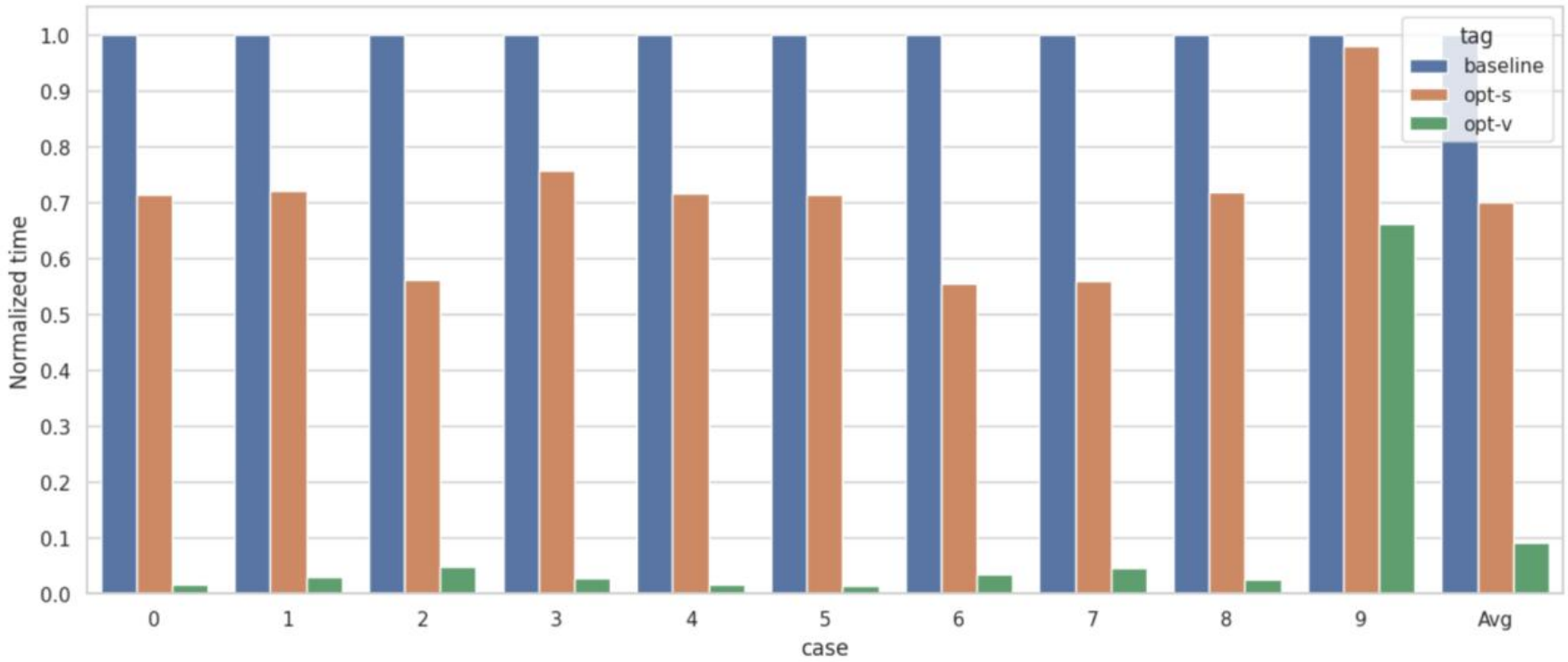



Maxpooling2d的操作示意图，其中输入为4x4，步长为2x2，输出为2x2




通过打包操作实现Maxpooling2d的示意图，其中输入为（2,2,4），虚线上展示了输入的逻辑视角，虚线下则展示了输入的物理视角

# Maxpooling2d(99.76分)



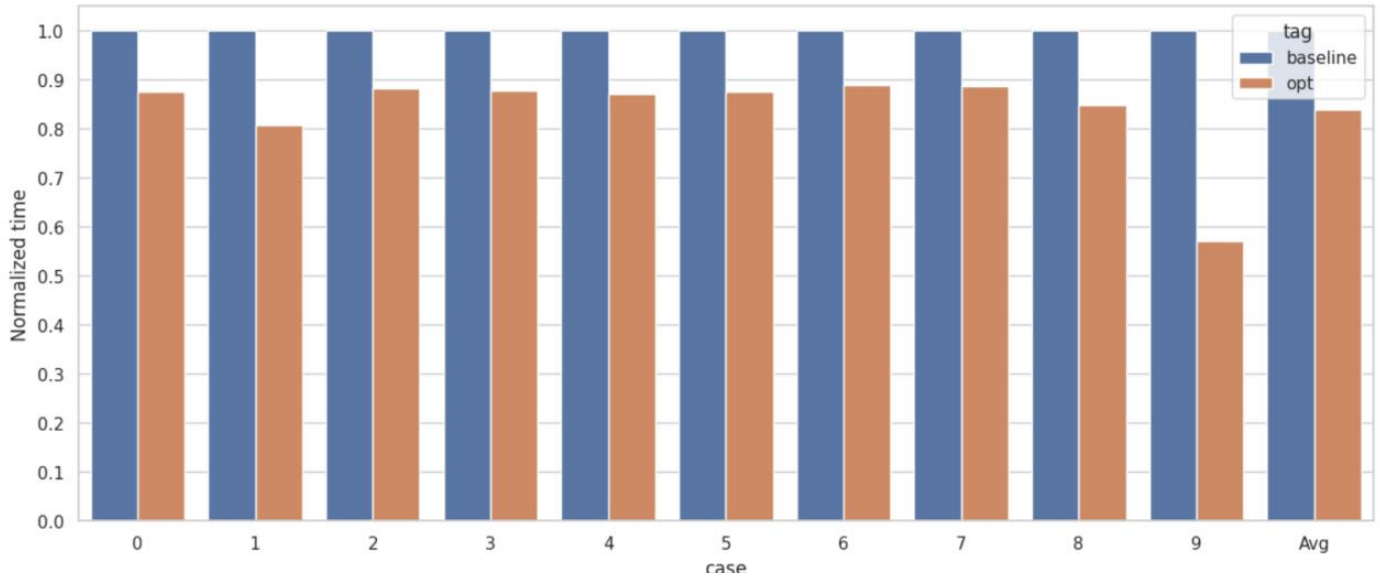

Maxpooling2d算子分别在标量(opt-s)和向量(opt-v)优化版本相对于baseline版本的归一化运行时间，opt-s平均实现了1.43倍的加速比，而opt-v平均实现了10.87倍加速比

# Sort算子(51.53分)

快速排序相比其他同级别平均时间复杂度的算法（归并排序，堆排序）在实际应用中有着出色的表现。

关于快速排序的优化方法：

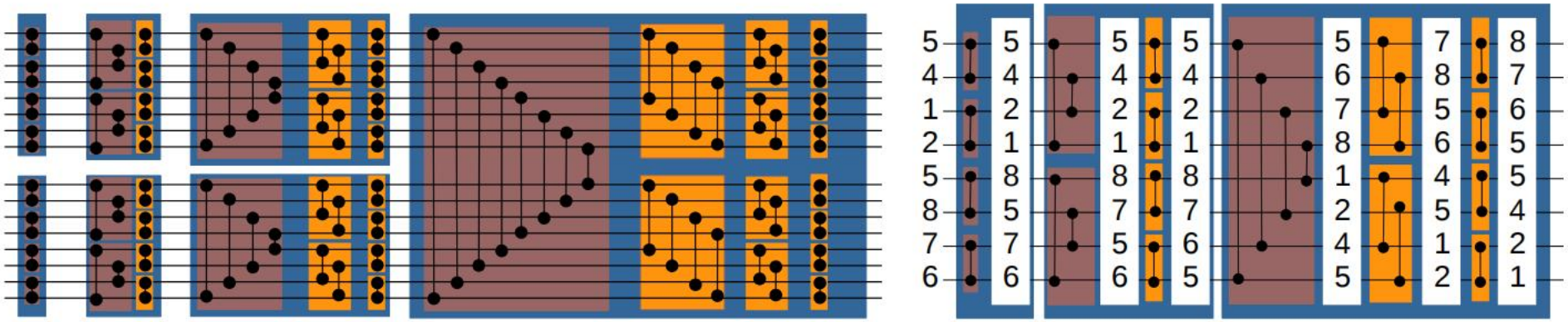
- 当递归遇到的数据长度小于某一阈值后，直接返回，最后采用一遍插入排序完成
- 锚点（pivot）的选择对于快速排序至关重要，所以可以采取三数取中法来获得离数组中位数较接近的数字




优化后的快速排序（opt）相比baseline的归一化运行时间，opt版本平均实现了1.19倍加速比

# Sort算子(51.53分)

关于向量化排序，之前有工作提出可以基于Intel的向量扩AVX-512[1]或AVX-2[2]实现基本数组的双调排序和元素划分。




但是上述方法会使用到基于掩码的操作来实现数据shuffle/permute，而本次实验平台暂不支持这个操作，替代操作开销会较大，故将排序算子的向量化工作留到以后。


[1] Bramas, B. (2017). A Novel Hybrid Quicksort Algorithm Vectorized using AVX-512 on Intel Skylake. International Journal of Advanced Computer Science and Applications, 8(10). https://doi.org/10.14569/ijacsa.2017.081044

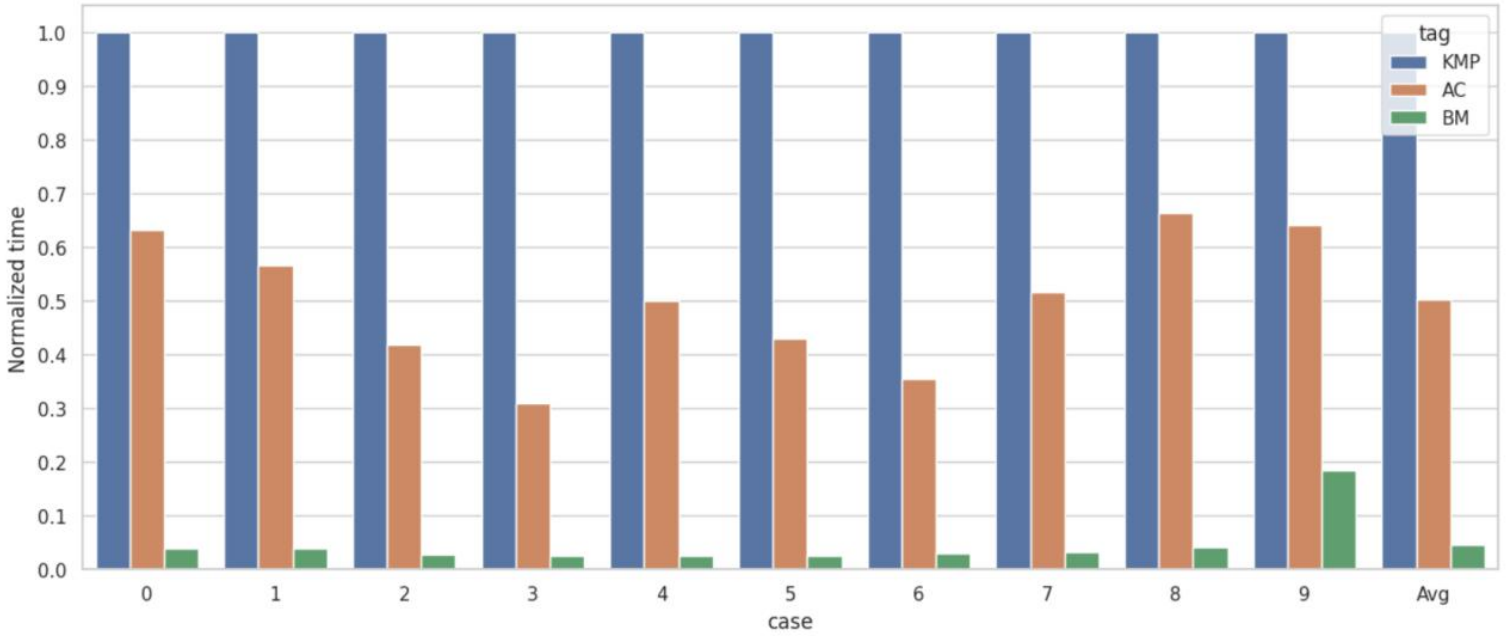
[2] Blacher, M., Giesen, J., & Kühne, L. (2021). Fast and Robust Vectorized In-Place Sorting of Primitive Types. In D. Coudert & E. Natale (Eds.), 19th International Symposium on Experimental Algorithms (SEA 2021) (Vol. 190, p. 3:1-3:16). Schloss Dagstuhl – Leibniz-Zentrum für Informatik. https://doi.org/10.4230/LIPIcs.SEA.2021.3

# StringMatch算子(96.3分)

StringMatch算子要求实现多匹配串和多模式串的字符串匹配查询。

尝试了以下算法：

- KMP——利用模式串的最长的相同的前后缀进行移动，而主串不需要回溯，从而达到快速匹配的目的
- AC自动机——以字典树结构为基础，结合KMP思想建立的自动机，用于解决多模式匹配等任务
- BM——将模式串从右往左进行比较，同时用到了两种规则——坏字符规则和好后缀规则，来计算移动的偏移量




AC和BM算法相对于KMP算法的归一化运行时间，其中AC算法实现了1.99倍加速比，BM算法21.42倍加速比

# 线上打榜成绩

| 排名 | 用户名 | 分数 |
| --- | --- | --- |
|  | 张力中 | 95.98 分 |
|  | 申邦渝 | 94.24 分 |
|  | 郭天宇 | 86.97 分 |
| 4 | 王鑫 | 83.95 分 |
| 5 | 赵文薪 | 81.37 分 |
| 6 | 胡鹏 | 79.66 分 |
| 7 | 赵全军 | 65.07 分 |
| 8 | 林玮臻 | 59.03 分 |

共 8 条 < 1 > 10 条/页

# THANKS & QA