基于RISC-V向量化指令集优化OpenBLAS和OpenCV初步进展

张先轶

PerfXLab 澎峰（北京）科技有限公司

xianyi@perfxlab.com

2020.7

澎峰科技——智能边缘计算解决方案提供商

- 2016年成立，来自中科院的高性能计算技术团队
- 领导国际知名开源矩阵计算库OpenBLAS（全球前3）
  - 应用范围广泛：高性能计算、深度学习等等

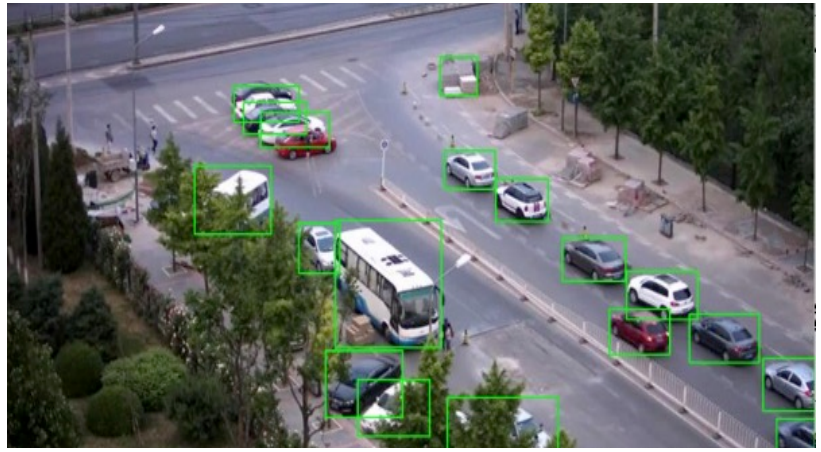
澎峰智能边缘计算全栈解决方案

- 全栈式（边缘端算法+软件框架+硬件加速）
- 覆盖平台广（高性能/高性价比/低功耗）
- 赋能各行业边缘端智能化

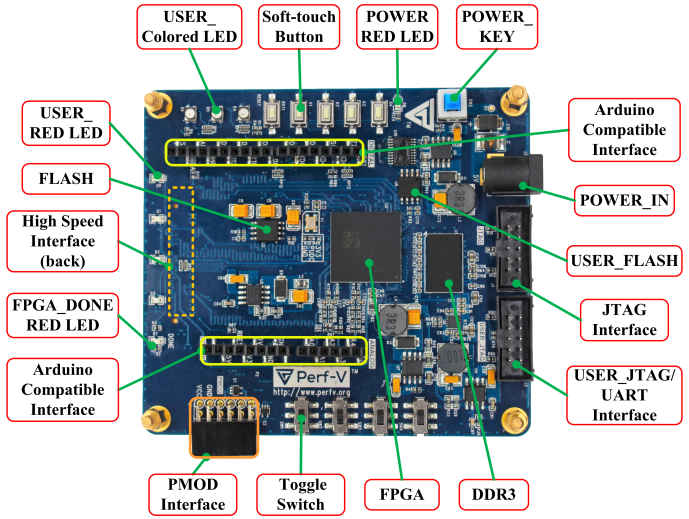
软件生态产品与服务

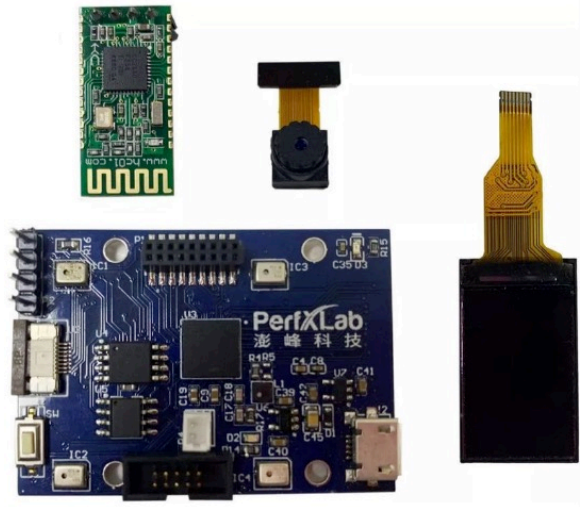
- 支持国产芯片软件生态建设
- RISC-V相关生态产品（开发板、软件等）

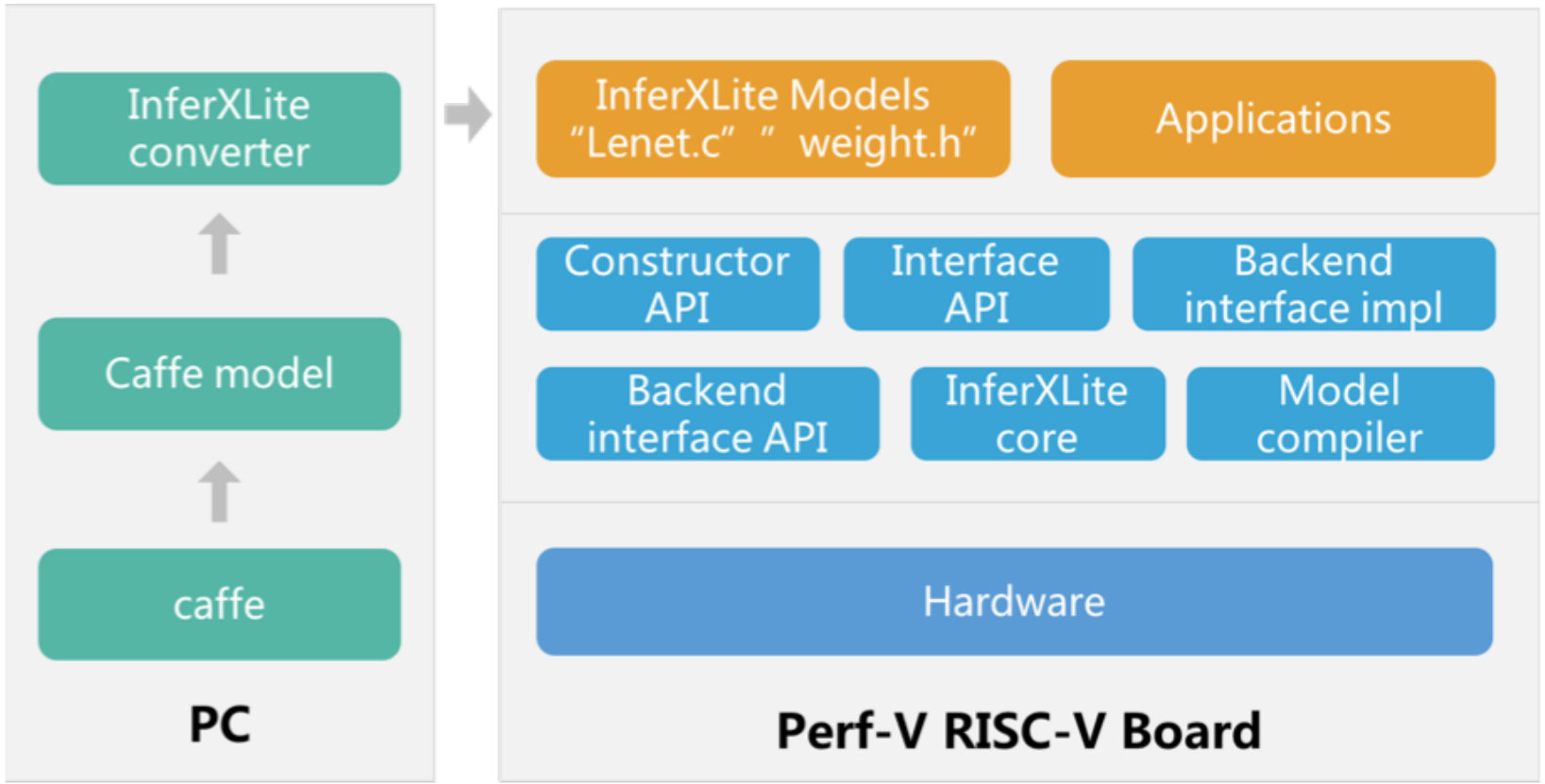
应用案例
无人机平台目标检测与追踪、图像处理等



Perf-V系列RISC-V开发板和轻量级深度学习推理框架

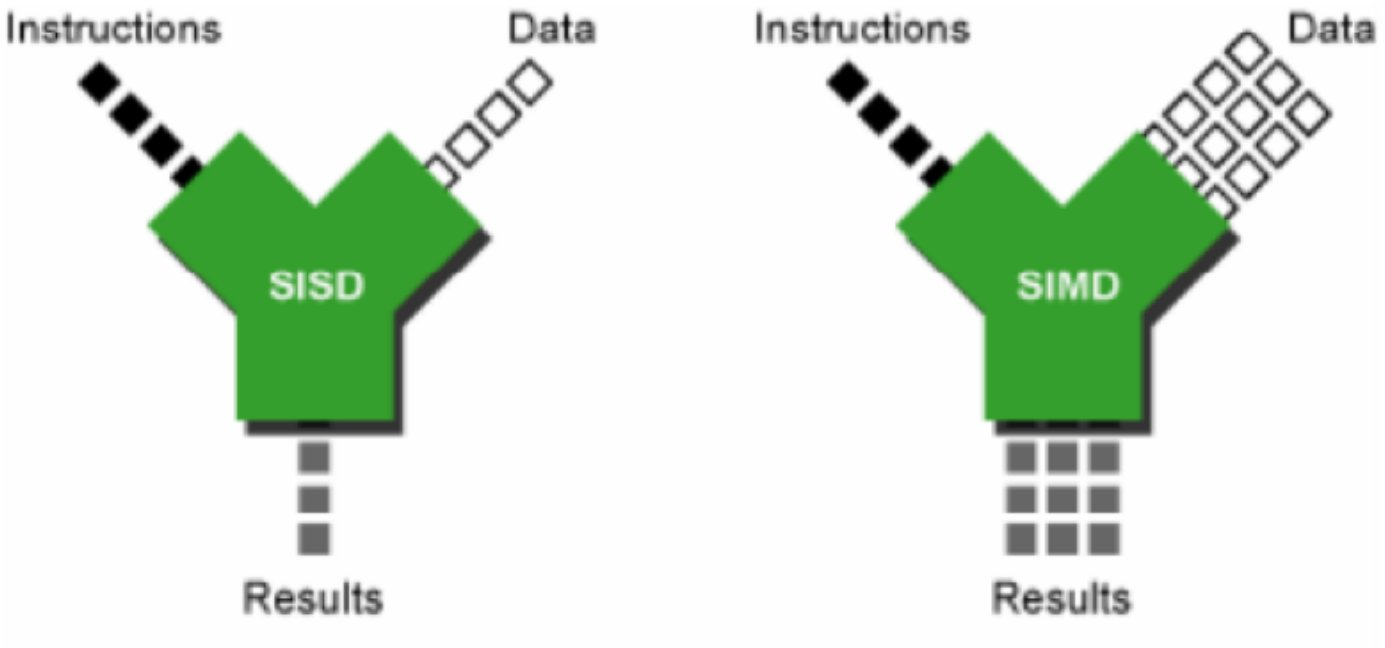






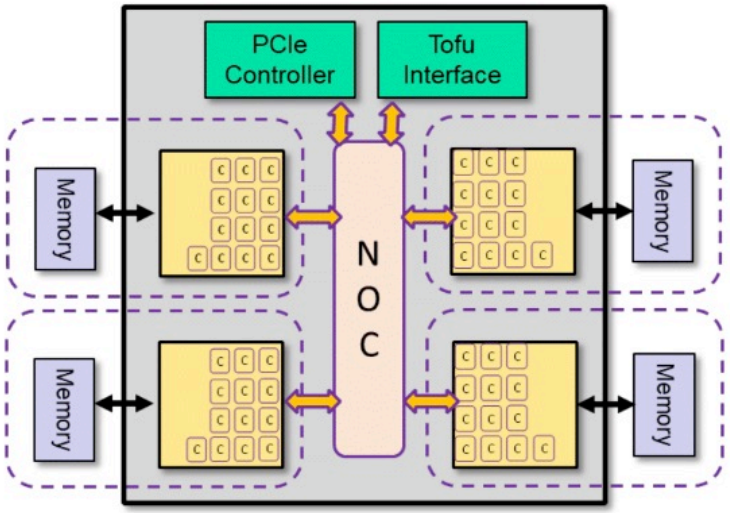
向量指令

- 标量指令，每次处理一个数据
  - 传统指令/通用指令
- 向量指令，每次处理一批数据
  - 也叫SIMD指令，单指令，多数据
  - 很多应用需要这种细粒度的并行性（比如，典型的多媒体应用）



基于ARM的超算第一次取得Top500第一

- 超级计算机Top500排行榜（ https://www.top500.org/ ）
  - 基于LINPACK测试（求解稠密线性方程组）
  - 2020.6月份榜单，第一名，Fugaku富岳，日本，富士通公司
- A64FX 处理器
  - ARM v8.2-A
  - 48/52核
  - **SVE指令集，512-bit向量 x2**



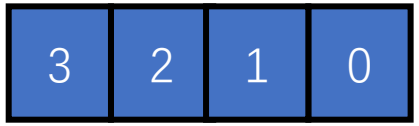
RISC-V V指令扩展

PerfXLab

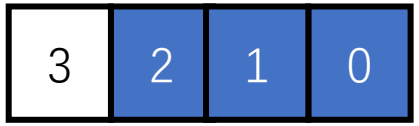
- RISC-V “V” Vector Extension
  - 当前版本0.9，0.8，0.7.1
  - https://github.com/riscv/riscv-v-spec
  - 变长，引入vector length, vl

VLEN=128b, SEW=32bit, LMUL=1

VI=4，操作4个数



VI=3，操作3个数



RISC-V V指令扩展与ARM Neon编程对比

PerfXLab

```
void vvaddint32(size_t n, const int *x, const int*y, int *z){
        for(size_t i=0; i<n; i++){
                z[i]=x[i]+y[i];
        }
}
```

ARM Neon，需要边角处理

```
void vvaddint32_arm_neon(size_t n, const int *x, const int*y, int *z){

        float32x4_t vz, vx, vy;

        for(size_t i=0; i<n/4*4; i+=4){
                vx=vld1q_f32(&x[i]);
                vy=vld1q_f32(&y[i]);
                vz=vaddq_f32(vx,vy);
                vst1q_f32(&z[i], vz);
        }

        for(; i<n; i++){
                z[i]=x[i]+y[i];
        }
}
```

RISC-V V，通过vl设置

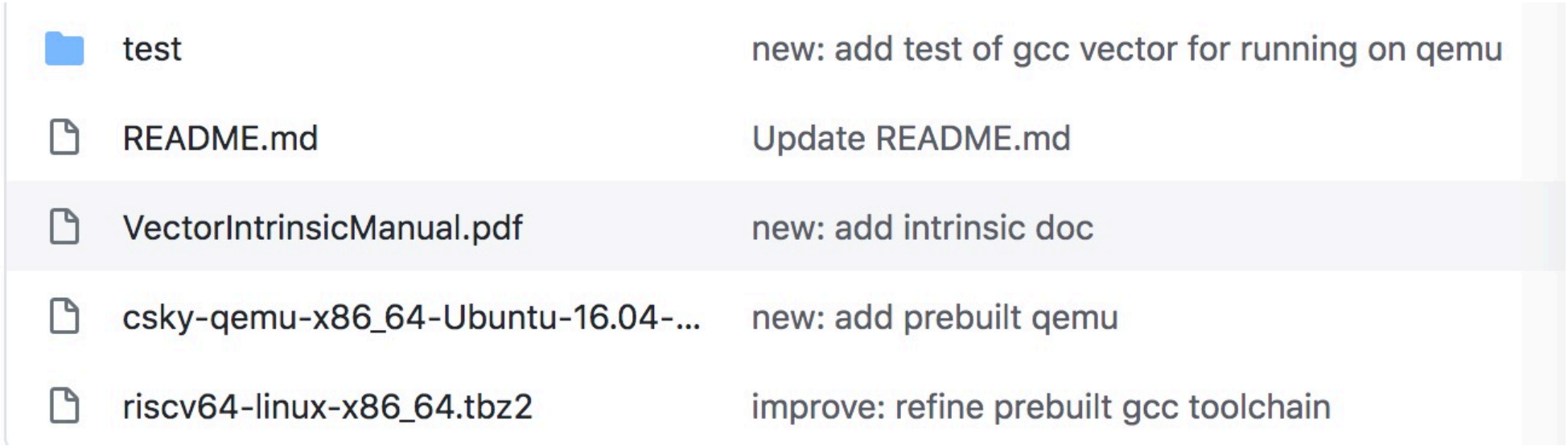
```
    # a0 = n, a1 = x, a2 = y, a3 = z
    # Non-vector instructions are indented
vvaddint32:
    vsetvli t0, a0, e32, ta,ma  # Set vector length based on 32-bit vectors
    vle32.v v0, (a1)         # Get first vector
      sub a0, a0, t0         # Decrement number done
      slli t0, t0, 2         # Multiply number done by 4 bytes
      add a1, a1, t0         # Bump pointer
    vle32.v v1, (a2)         # Get second vector
      add a2, a2, t0         # Bump pointer
    vadd.vv v2, v0, v1       # Sum vectors
    vse32.v v2, (a3)         # Store result
      add a3, a3, t0         # Bump pointer
      bnez a0, vvaddint32    # Loop back
      ret                    # Finished
```

RISC-V V指令的Intrinsic支持

PerfXLab

- 阿里平头哥的版本

  https://github.com/c-sky/xuantie-vector-demos



目前Intrinsic支持 0.7.1和0.8版本的V指令

具体看VectorIntrinsicManual.pdf文档

案例介绍—优化OpenBLAS

PerfXLab

- **开源矩阵计算库，10年历史**
  - https://github.com/xianyi/openblas
- **矩阵和向量基本计算**
  - 当前版本：0.3.10
- **支持主流处理器**
  - X86/ARM/Power/龙芯
- **支持多种操作系统**
  - Linux/Windows/Mac OSX/Android
- **目前简单支持RISC-V架构**
  - 但是不支持V指令集优化

OpenBLAS用户

julia

GNU Octave

dmlc mxnet

Caffe

IBM

国家超级计算济南中心 NATIONAL SUPERCOMPUTER CENTER IN JINAN

VERSATUShpc HIGH PERFORMANCE CONSULTING

PathScale High Performance Compilers

HUAQIAO UNIVERSITY 華僑大學

XIDIAN UNIVERSITY

360 www.360.com

網易 NETEASE www·163·com

案例介绍—优化OpenBLAS

PerfXLab

- OpenBLAS调用结构分为3层

```
interface/gemm.c
        |
driver/level3/level3.c
        |
gemm assembly kernels at kernel/
```

- 其中kernel层，是汇编或者Intrinsic实现，体系架构相关的

Here is an example for `kernel/x86_64/KERNEL.HASWELL`

```
...
DTRMMKERNEL    =  dtrmm_kernel_4x8_haswell.c
DGEMMKERNEL    =  dgemm_kernel_4x8_haswell.S
...
```

案例介绍—优化OpenBLAS

- 以RISC-V V指令集的Intrinsic优化OpenBLAS
- BLAS1级的 axpy函数

The ?axpy routines perform a vector-vector operation defined as

```
y := a*x + y
```

where:

a is a scalar

x and *y* are vectors each with a number of elements that equals *n*.

- C语言接口如下，向量x和y不一定是连续存储的，可以以inc间隔存储

```
void cblas_daxpy (const MKL_INT n, const double a, const double *x, const MKL_INT incx, double *y, const MKL_INT incy);
```

案例介绍—优化OpenBLAS

PerfXLab

OpenBLAS中axpy内核的naïve实现

```
BLASLONG i=0;
BLASLONG ix,iy;

if ( n < 0     )  return(0);
if ( da == 0.0 ) return(0);

ix = 0;
iy = 0;

while(i < n)
{

        y[iy] += da * x[ix] ;
        ix += inc_x ;
        iy += inc_y ;
        i++ ;

}
return(0);
```

OpenBLAS中axpy内核的ARM向量化实现

（200行汇编）

```
    PROLOGUE

    cmp     N, xzr
    ble     .Laxpy_kernel_L999

    fcmp    DA, #0.0
    beq     .Laxpy_kernel_L999

    cmp     INC_X, #1
    bne     .Laxpy_kernel_S_BEGIN
    cmp     INC_Y, #1
    bne     .Laxpy_kernel_S_BEGIN

.Laxpy_kernel_F_BEGIN:
```

案例介绍—优化OpenBLAS

PerfXLab

OpenBLAS中axpy内核RISC-V V优化

- 假如x和y连续，调用向量连续load/store，访存效率更高

- Vl大部分是不变的，为了减少vsetvli的使用
  - 分为主干和末尾
  - 主干循环只调用一次
- Vlev和vsev都是连续的向量访存
- Vfmacc.vf是标量和向量相乘，累加到向量

```
if (inc_x == 1 && inc_y == 1) {

        gvl = vsetvli(n, RVV_EFLOAT, RVV_M);

        if (gvl <= n/2) {
                for (i = 0, j=0; i < n/(2*gvl); i++, j+=2*gvl) {
                        vx0 = VLEV_FLOAT(&x[j], gvl);
                        vy0 = VLEV_FLOAT(&y[j], gvl);
                        vy0 = VFMACCVF_FLOAT(vy0, da, vx0, gvl);
                        VSEV_FLOAT(&y[j], vy0, gvl);

                        vx1 = VLEV_FLOAT(&x[j+gvl], gvl);
                        vy1 = VLEV_FLOAT(&y[j+gvl], gvl);
                        vy1 = VFMACCVF_FLOAT(vy1, da, vx1, gvl);
                        VSEV_FLOAT(&y[j+gvl], vy1, gvl);
                }
        }
```

案例介绍—优化OpenBLAS

OpenBLAS中axpy内核RISC-V V优化

- 尾部处理
- 一样使用向量指令，而不是标量指令

```
//tail
if (j<n) {
        gvl = vsetvli(n - j, RVV_EFLOAT, RVV_M);
        vx0 = VLSEV_FLOAT(&x[j*inc_x], stride_x, gvl);
        vy0 = VLEV_FLOAT(&y[j], gvl);
        vy0 = VFMACCVF_FLOAT(vy0, da, vx0, gvl);
        VSEV_FLOAT(&y[j], vy0, gvl);
}
```

案例介绍—优化OpenBLAS

PerfXLab

OpenBLAS中axpy内核RISC-V V优化

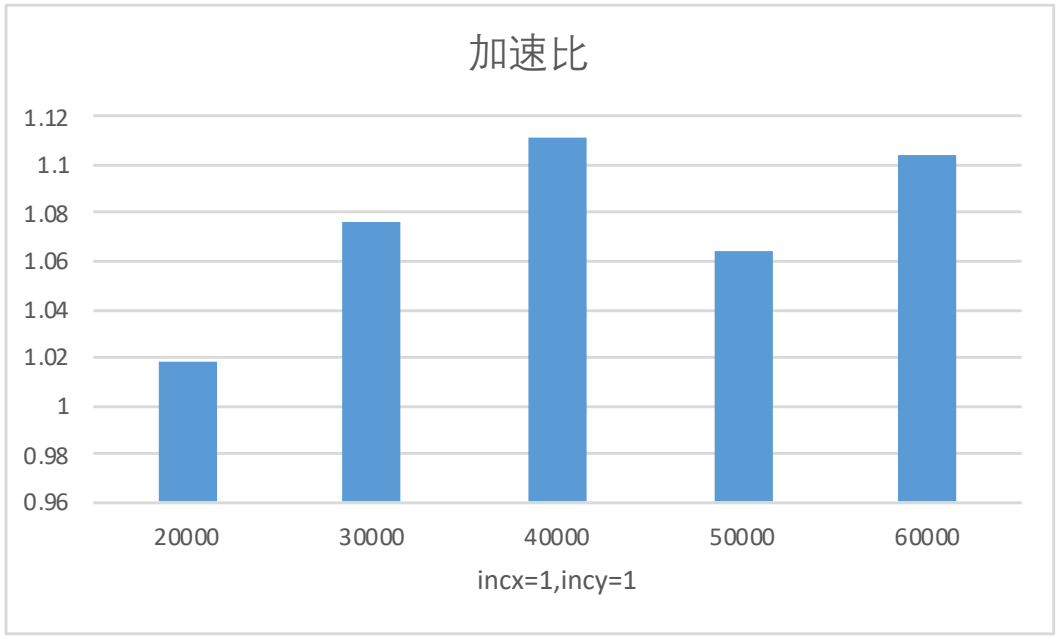
- 对于inc不等于1的情况，使用vlsev这种带stride的访存指令

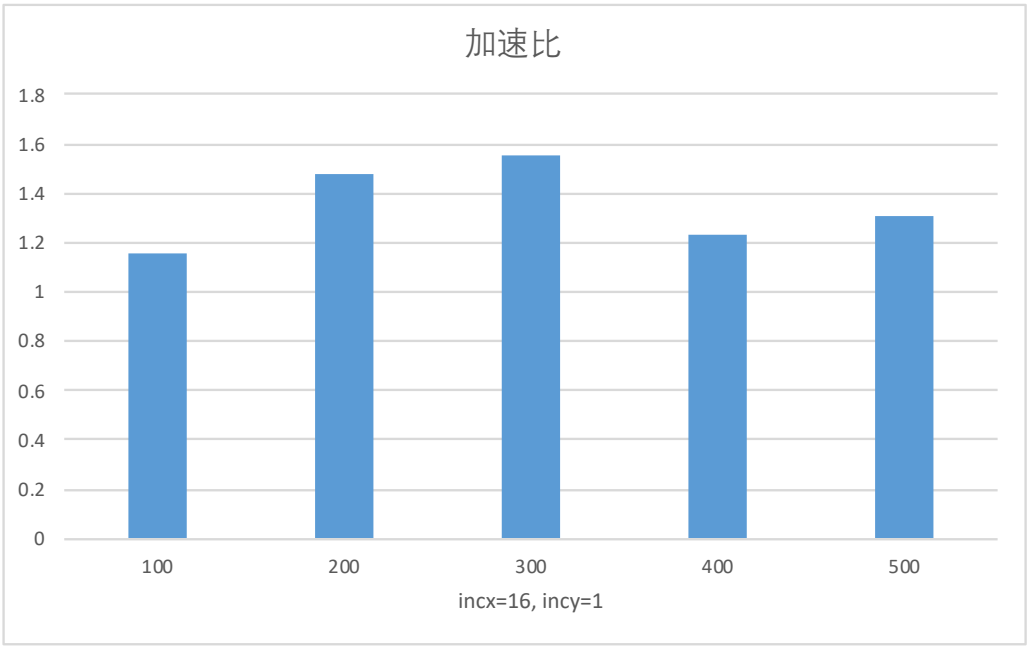
  Stride是以字节为单位

```
stride_x = inc_x * sizeof(FLOAT);
stride_y = inc_y * sizeof(FLOAT);
gvl = vsetvli(n, RVV_EFLOAT, RVV_M);
if(gvl <= n/2){
        BLASLONG inc_xv = inc_x * gvl;
        BLASLONG inc_yv = inc_y * gvl;
        for(i=0,j=0; i<n/(2*gvl); i++){
                vx0 = VLSEV_FLOAT(&x[jx], stride_x, gvl);
                vy0 = VLSEV_FLOAT(&y[jy], stride_y, gvl);
                vy0 = VFMACCVF_FLOAT(vy0, da, vx0, gvl);
                VSSEV_FLOAT(&y[jy], stride_y, vy0, gvl);

                vx1 = VLSEV_FLOAT(&x[jx+inc_xv], stride_x, gvl);
                vy1 = VLSEV_FLOAT(&y[jy+inc_yv], stride_y, gvl);
                vy1 = VFMACCVF_FLOAT(vy1, da, vx1, gvl);
                VSSEV_FLOAT(&y[jy+inc_yv], stride_y, vy1, gvl);

                j += gvl * 2;
                jx += inc_xv * 2;
                jy += inc_yv * 2;
        }
}
```

案例介绍—优化OpenBLAS

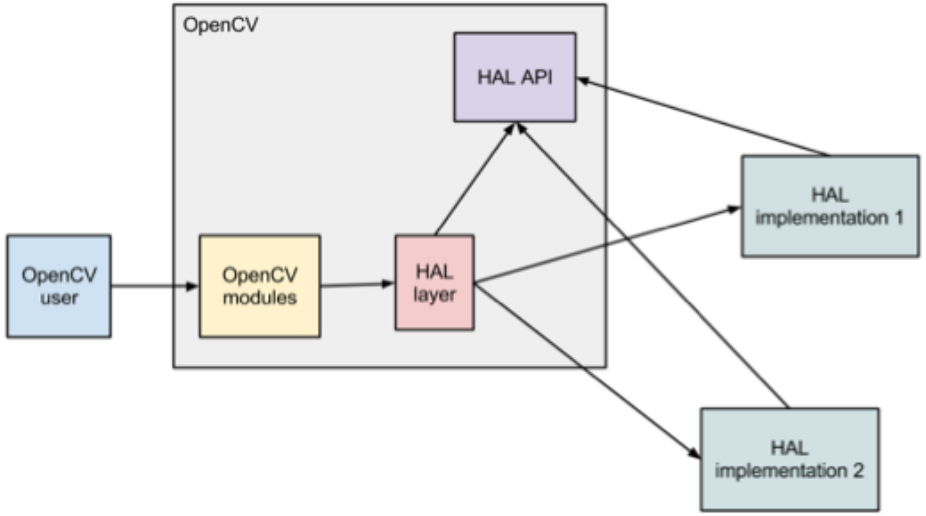
OpenBLAS中daxpy RISC-V V优化性能数据





案例介绍—优化OpenCV

- 计算机视觉库OpenCV
- 通过HAL（Hardware Acceleration Layer）



HAL API：

- 一组函数原型，普通C接口（没有类、STL、异常）
- 只能扩展，不能改变原型
- 不必支持所有操作、输入参数
- 不依赖OpenCV API，独立类型、宏、常量定义
- HAL layer是HAL API的C++瘦包装器（thin wrapper），core模块的一部分

案例介绍—优化OpenCV

PerfXLab

- HAL RISC-V V实现

数据结构：

- 指定向量长度(目前只做128-bit向量)
- HAL数据结构映射，比如
  v_int8x16 -> int8xm1_t intrinsic数据结构
  v_float32x4 -> float32xm1_ t intrinsic数据结构

函数接口：

- HAL函数，有直接指令支持
  - 向量加减乘除，取最大值等
  - ....
- HAL函数，无RISC-V V指令支持
  - v_invsqrt -> vfrdiv(vsqrt, 1)
  - v_absdiff -> vsub(vmax, vmin)
  - .....

案例介绍—优化OpenCV

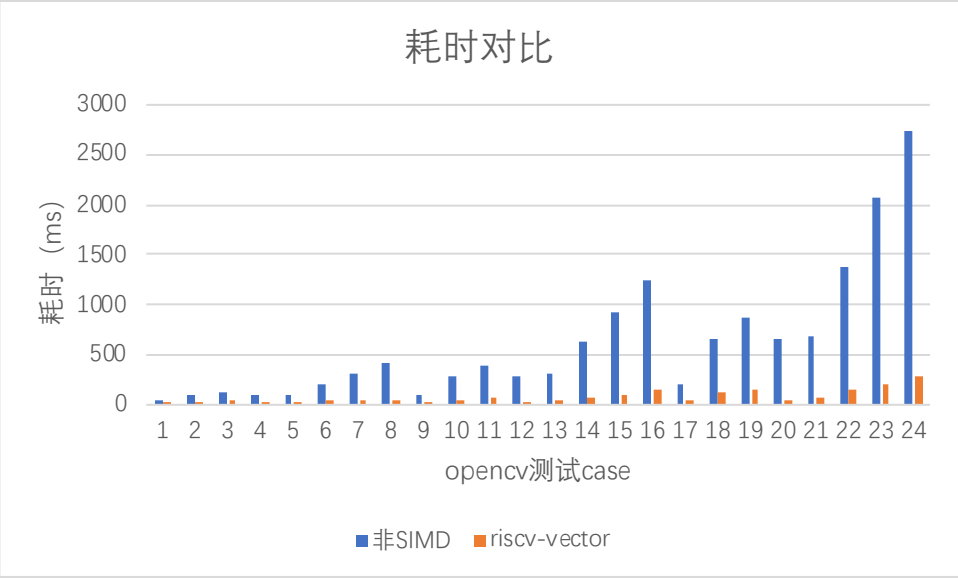
- 函数 ADD

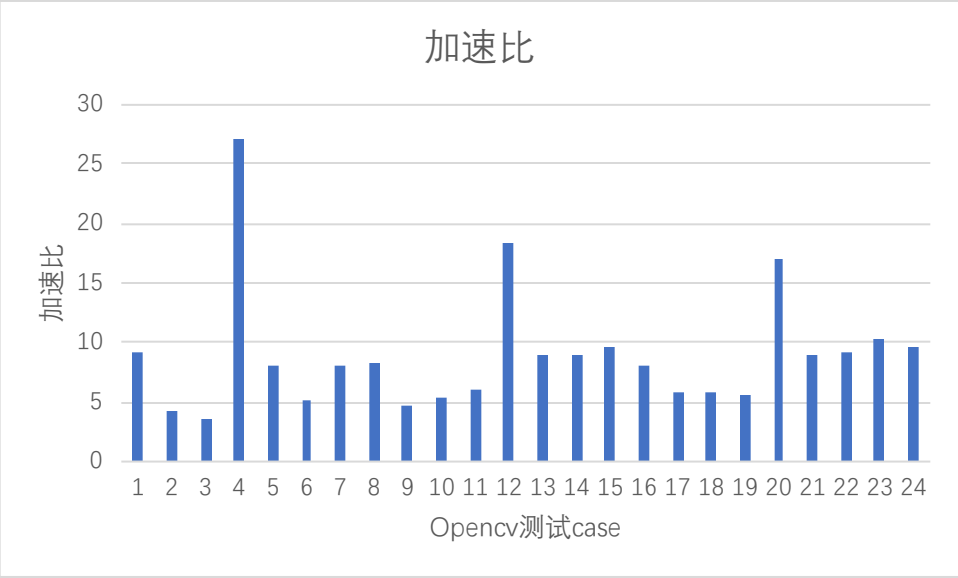
向量化优化：

- 向量加法指令

测试规模：640x480~1920x1080

测试数据类型：8U、8S、16S、32S、32F
测试通道数：C1、C3、C4





案例介绍—优化OpenCV

- 函数 medianBlur

中值滤波算法说明：

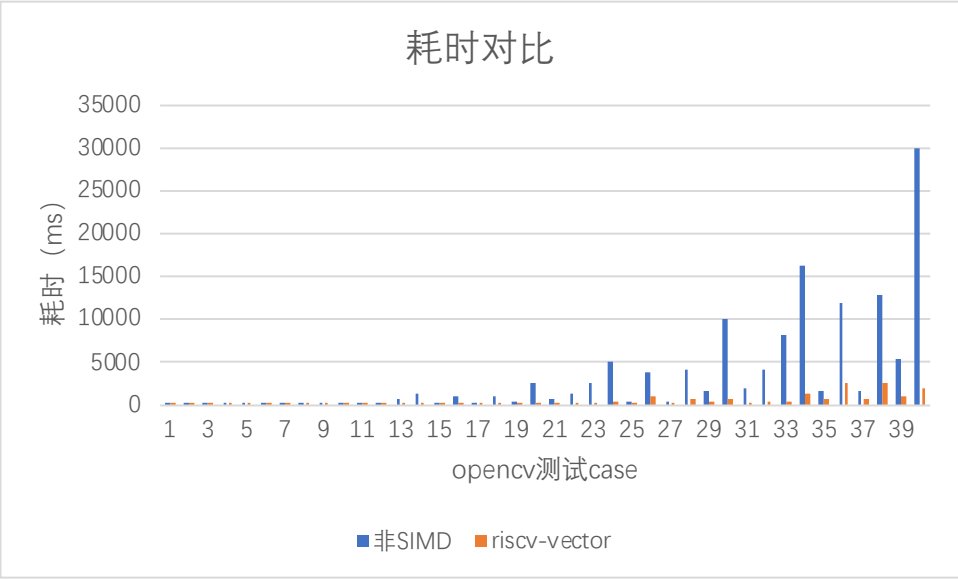
- 对于一个3x3的kernel，通过k次的max、min操作求中值。

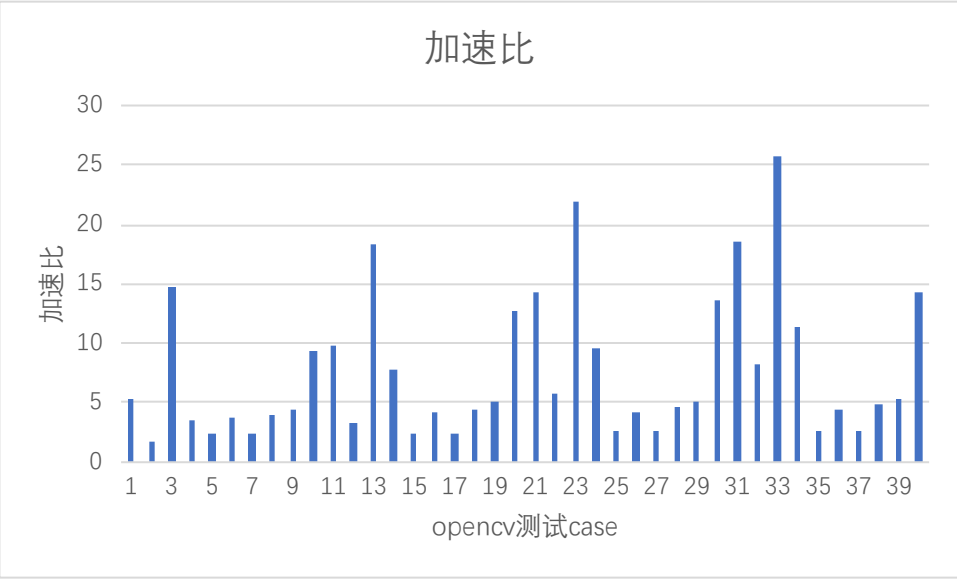
向量化优化：

- 读取3x3个向量，通过k次的v_max、v_min操作求3x3个向量的中值。

测试规模：127x61~1280x720
测试kernel：3、5

测试数据类型：8U、16U、16U、32F
测试通道数：C1、C4



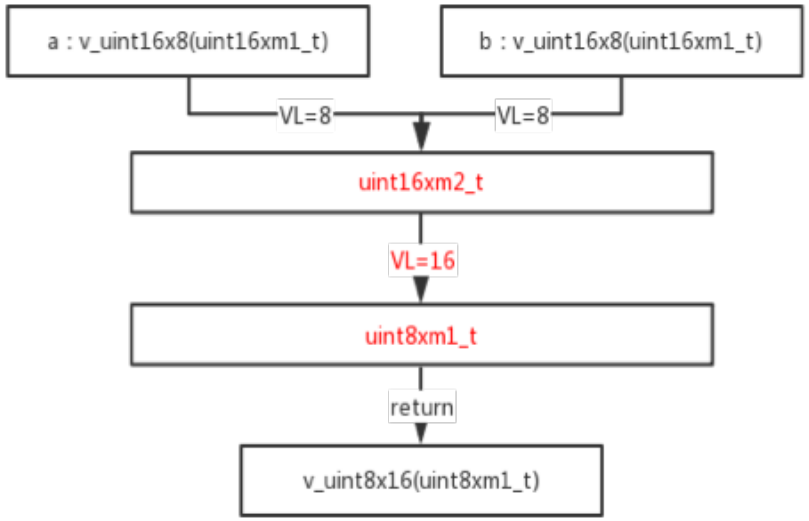


案例介绍—优化OpenCV 遇到的问题

- **函数 Size_MatType_CmpType_compare.compare/18~19**

算法说明：

- 采用将非8bit（非8U/8S）的数据类型转换成8bit进行操作和存储，即存在v_pack_b操作
- Compare函数处理16U最终输出为8U类型，存在narrow操作，通过v_pack_b接口实现，v_pack_b操作如下：



- Vector Length变化导致的性能损失
- OpenCV的HAL层实现更多采用定长的数据结构
  - 对于expand，返回扩展后的多个定长的数据类型，v_int8x16 变成 2个 v_int16x8，造成VL减小一半，narrow类似
  - 是否需要修改HAL的设计？

总结

PerfXLab

- 向量化指令对于处理器的性能提升至关重要
  - Intel AVX，AVX-512
  - ARM Neon, SVE
- RISC-V V扩展指令集还在发展中
  - 大部分指令稳定了，也有编译工具链和模拟器支持
  - Intrinsic编程也有阿里平头哥做了支持
- 开源软件生态
  - 很多的开源软件需要RISC-V的移植和优化
  - OpenBLAS优化中，近期完成
  - OpenCV HAL层的设计是否与RISC-V V匹配？

PerfXLab

感谢阿里平头哥的支持

期待与各位交流，为RISC-V生态发展贡献力量

欢迎实习生☺

张先轶@PerfXLab

北京 海淀

扫一扫上面的二维码图案，加我微信