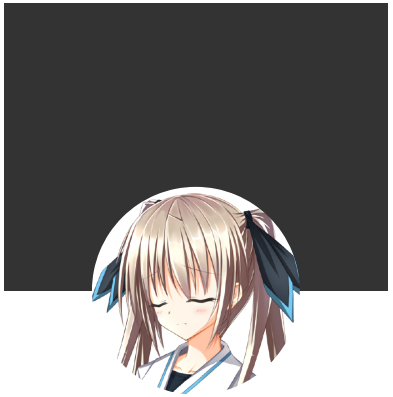


Rainmaker's Notebook

『求雨巫师的神奇之处在于他总是躲着不见你，却总说刚下完的雨是拜他所赐。』——《天真的人类学家》

Home

Archives

About

SiteXC

使用 AVX 系列指令集进行向量化

最近上课讲到了向量化的内容，我顺便整理了一下自己手头以前相关的资料。下面内容的主要关注点是在 x86 平台上让编译器进行自动向量化，但是第二部分所提到的一些原则在各个平台上都适用。

指令集基本情况

指令集	向量部件长度（bits）	每周期浮点数操作（SP/DP）
AVX	256	16 / 8
AVX-2	256	32 / 16 (with FMA)
AVX-512	512	64 / 32 (with FMA)

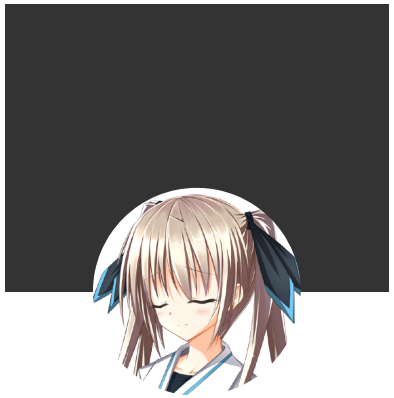
Xeon Phi 与下一代 Xeon 共有指令集：AVX512-{F, CD}
Xeon Phi 专有指令集：AVX512-{ER, PF}
下一代 Xeon 专有指令集：AVX512-{BW, DQ, VL, IFMA52, VBMI}

Arch	Instruction Set
Nehalem, Westmere	SSE 4.1
Sandy Bridge, Ivy Bridge	SSE 4.1, AVX
Haswell, Broadwell	SSE 4.1, AVX, AVX2
KNL Xeon Phi	SSE 4.1, AVX, AVX2, AVX-512F, CDI, ERI, PFI
Skylake, Skymont	SSE 4.1, AVX, AVX2, AVX-512F, CDI, AVX-512{VL,BW,DQ}

编译器与编译参数

ICC 15 及以上、GCC 4.9.2 及以上支持 AVX-512。使用 `-O2` 或更高优化级别将自动进行向量化优化。

GCC 与 ICC 都接受 `-S` 参数，输出中间汇编文件，以检查是否使用了向量化指令。以 `v` 开头的指令，如 `vaddpd` `vmovupd` 则是向量化指令。`xmm` `ymm` `zmm` 分别表示 128 bit、256 bit 和 512 bit 寄存器。



Rainmaker's Notebook

『求雨巫师的神奇之处在于他总是躲着不见你，却总说刚下完的雨是拜他所赐。』——《天真的人类学家》

Home

Archives

About

SiteXC

在 ICC 中，使用 `-qopt-report=5` （5 可以替换为 1~4）来输出优化信息，可以看到做了向量化的地方。
在 GCC 中，使用 `-fopt-info-vec-all` 来输出所有向量化信息（更多选项参见 Ref-5）。

查看当前编译器针对本机定义了哪些 SSE / AVX 宏：

- ICC： `icc -dM -E -xHost - < /dev/null | egrep "SSE|AVX" | sort`
- GCC： `gcc -dM -E -march=native - < /dev/null | egrep "SSE|AVX" | sort`

指示编译器使用 AVX / AVX2 指令集：

- ICC： `-xAVX`， `-xAVX2`， `-xHost` （针对本机架构）
- GCC： `-mavx`， `-mavx2`， `-march=native` （针对本机架构）

指示编译器使用 AVX-512 指令集：

Arch	ICC	GCC
KNL and Xeon	-xCOMMON-AVX512	-mavx512f -mavx512cd
KNL only	-xMIC-AVX512	-mavx512f -mavx512cd -mavx512er -mavx512pf
Xeon only	-xCORE-AVX512	-mavx512f -mavx512cd -mavx512bw -mavx512dq -mavx512vl -mavx512ifma -mavx512vbmi

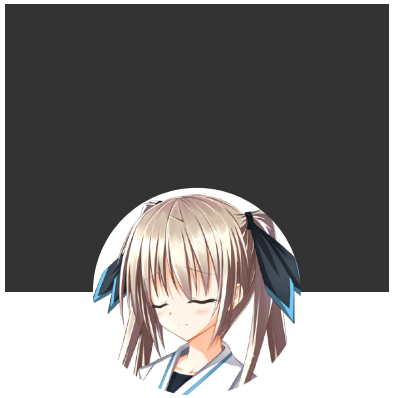
编程使用

可以手动使用 Intrinsic （内联）函数进行向量化操作，函数文档参见 Ref-8。

符合一定条件的循环，编译器可以自动向量化：

- 必须是最内层循环
- 循环的长度是确定的（如果是固定的会更好）
- 循环体内没有变量依赖关系（比如 `a[i] = a[i - 1] + a[i - 2]` ），但是归约类操作不在此列（比如对一个数组进行求和）。注意，有时候编译器会认为循环有潜在的数据依赖而无法向量化，比如以下的代码：

```
1   for (int i = 0; i < size; i++)
2       c[i] += a[i] * b[i];
```



Rainmaker's Notebook

『求雨巫师的神奇之处在于他总是躲着不见你，却总说刚下完的雨是拜他所赐。』——《天真的人类学家》

编译器可能会怀疑 `a` 或 `b` 的部分位置与 `c` 的部分位置重叠，比如存在如下关系 `&a[i] == &c[i - 4]`。这个时候可以通过编译制导语句（见下）来提示编译器进行向量化。

- 循环访问的数据是连续的，或者最起码是相同的步长间隔（间接寻址如 `y[pos[i]] += data[i] * x[i]` 的效率很低）
- 循环体内没有复杂的分支判断；一个 `if` 是可以接受的，因为可以被转换成 mask operation，比如以下函数：

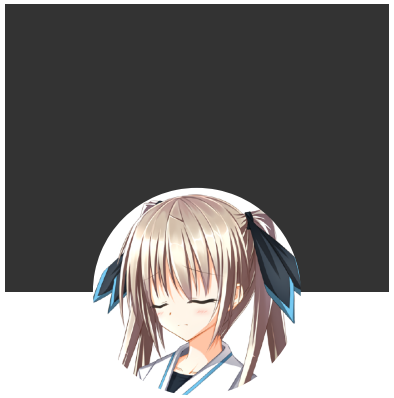
```
1  void quad(const int &len, const double *a, const double *b, const double *c, double *x1
2  {
3      for (int i = 0; i < len; i++)
4      {
5          double s = b[i] * b[i] - 4.0 * a[i] * c[i];
6          if (s >= 0)
7          {
8              s = sqrt(s);
9              x2[i] = (-b[i] + s) / (2.0 * a[i]);
10             x1[i] = (-b[i] - s) / (2.0 * a[i]);
11         } else {
12             x2[i] = 0.0;
13             x1[i] = 0.0;
14         }
15     }
16 }
```

- 循环内没有调用其它复杂的函数；简单的可以被 inline 的函数与以下数学函数例外：`acos`, `acosh`, `asin`, `asinh`, `atan`, `atan2`, `atanh`, `cbrt`, `ceil`, `cos`, `cosh`, `erf`, `erfc`, `erfinv`, `exp`, `exp2`, `fabs`, `floor`, `fmax`, `fmin`, `log`, `log10`, `log2`, `pow`, `round`, `sin`, `sinh`, `sqrt`, `tan`, `tanh`, `trunc`

以下要点有助于编译器自动进行向量化或提高向量化效率：

- 使用对齐了的数据，即向量化操作的数据段前端应该对齐到若干字节。常用的内存申请和释放函数是：

```
1 #include <x86intrin.h>
2
3 _mm_malloc(size_t size, size_t align) // 内存块大小为 size，对齐到 align bit 的起始位置
4 _mm_free(void *)
```



Rainmaker's Notebook

『求雨巫师的神奇之处在于他总是躲着不见你，却总说刚下完的雨是拜他所赐。』——《天真的人类学家》

Home

Archives

About

SiteXC

分配的数组被用作二维数组时，可以在每一行的尾部留一些多余的空间，以使得每一行的左端可以对齐。

- 使用 SoA（Structure of Array，数组结构） 而不是 AoS（Array of Structures，结构体数组），这样内存的访问连续。比如如下代码：

```
struct sCoord
{
    double x, y;
};
/* other codes */
sCoord a[N], b[N];
/* other codes */
for (int i = 0; i < N; i++)
{
    a[i].x += b[i].x;
    a[i].y += b[i].y;
}
```

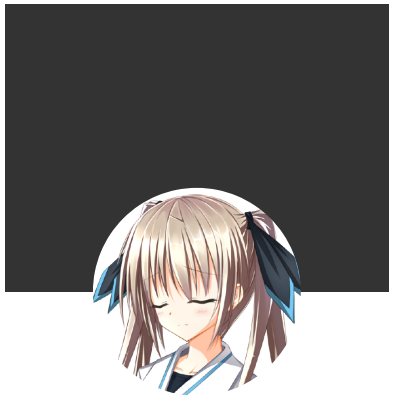
可以被改写为以下代码以获得更好的性能：

```
double Coord_a_x[N], Coord_a_y[N], Coord_b_x[N], Coord_b_y[N];
/* other codes */
for (int i = 0; i < N; i++)
{
    Coord_a_x[i] += Coord_b_x[i];
    Coord_a_y[i] += Coord_b_y[i];
}
```

- 在满足精度的要求下，使用最小的数据类型：可以用 `short` 的就不要用 `int`，可以用 `float` 的就不要用 `double`
- 不要在想要向量化的循环中混合使用不同的数据类型
- 避免使用向量化硬件部件不支持的操作，比如使用 80 bit 浮点数或者使用去余操作（`%`）

常用编译制导语句

- `#pragma ivdep`：告诉编译器，下面的循环中没有变量依赖关系
- `#pragma vector aligned`：提示编译器进行向量化，并且编译器将使用对齐的数据存取指令
- `#pragma vector temporal`：提示编译器进行向量化，并且**数据从内存中先取到 Cache 再读写**



Rainmaker's Notebook
『求雨巫师的神奇之处在于他总是躲着不见你，却总说刚下完的雨是拜他所赐。』——《天真的人类学家》

- `#pragma vector nontemporal` ：提示编译器进行向量化，并且**数据不经过 Cache，直接从内存进行读写**

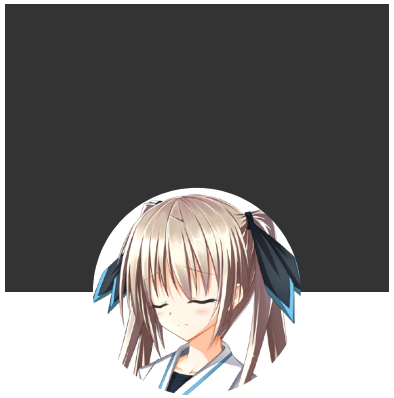
  如果不指定是 `temporal` 还是 `nontemporal` ，编译器会自动决定。
- `#pragma simd` ：**要求**编译器进行向量化
- `#pragma prefetch var:hint:distance` ：KNC only，指示编译器进行预取。 `var` 指定要预取的数组； `hint` 选 0 表示取到 L1 Cache，1 表示取到 L2 Cache； `distance` 表示提前多少个向量单元进行预取，比如其取值为 8 时，则提前 8 *16 个 float 或者 8* 8 个 double 进行预取。

  除了编译制导语句，也可以使用 Intrinsic 进行手动预取。函数是 `void _mm_prefetch(char const*address, int hint)` ， `address` 是要预取回来的数据的头地址， `hint` 同制导语句含义。在手动预取时，需要设置 `-opt-prefetch=0` 或 `#pragma noprefetch` 以关闭编译器预取，避免发生不可知的冲突。
- `#pragma unroll(<UNROLL_NUM>)` ：提示编译器进行 `<UNROLL_NUM>` 路循环展开，因为太短的循环体不利于进行指令调度。下面是一个等效 Demo：

```
for (int i = 0; i < N; i += 4) // 假设 N 是 4 的倍数
{
    y[i    ] += alpha * x[i];
    y[i + 1] += alpha * x[i + 1];
    y[i + 2] += alpha * x[i + 2];
    y[i + 3] += alpha * x[i + 3];
}
// 上面的代码等效于下面的代码
#pragma unroll(4)
for (int i = 0; i < N; i++)
    y[i] += alpha * x[i];
```

References

1. A Guide to Vectorization with Intel C++ Compilers
2. Preparing for a smooth landing: Intel’ s Knights Landing and Modern Applications
3. Guide to Automatic Vectorization with Intel AVX-512 Instructions in Knights Landing Processors
4. Intel Advanced Vector Extensions 2015/2016 Support in GNU Compiler Collection
5. GCC Developer Options



Rainmaker's Notebook

『求雨巫师的神奇之处在于他总是躲着不见你，却总说刚下完的雨是拜他所赐。』——《天真的人类学家》

Home

Archives

About

SiteXC

6. GCC 5.4.0 x86 Options
7. ICC 15 Reference - Pragmas - vector
8. Intel Intrinsics Guide（Intel 内联函数文档）
9. Compiler Prefetching for KNC
10. Intel Knights Corner 的结点级内存访问优化

发布于 2017-09-29

tags: { C } { SIMD } { ICC }

Related Issues not found

Please contact @EnigmaHuang to initialize the comment

Login with GitHub

© 2023 - Enigma Huang
Powered by Hexo , Theme - Icalm