Xiamu-ssr/Mmul: 简单的矩阵加乘 (github.com)

矩阵结构体

```
#define D1 1023
#define D2 1023
#define D3 1023
#define D4 1023

struct matrix
{
    char *ch = (char*)malloc(sizeof(char)*D1*D1);
    int *inter1 = (int*)malloc(sizeof(int)*D2*D2);
    float *flo = (float*)malloc(sizeof(float)*D3*D3);
    int *inter2 = (int*)malloc(sizeof(int)*D4*D4);
};
```

编译器	优化选项	时间 s	加速比
Intel(R) Xeon(R) CPU E5-2689 0 @ 2.60GHz 8 核 16 线程			
gcc/g++	默认无	20	1
	-O2	2.6	7.7
icc/icpc	-O0	17	1.2
	-O1	3	6.7
	-O1 -ipo	2.1	9.5
	-O2	1.1	18.2
	-O2 -ipo -vec -xAVX	0.72	27.8
	-O2 -ipo -vec -xAVX -parallel -par-num-threads=14 -par-schedule-guided=1	0.105	190.5
2 * Intel(R) Xeon(R) Silver 4216 CPU @ 2.10GHz 16 核 32 线程 Di *= 2			
gcc/g++	默认无	111	1
icc/icpc	-O0	100	1.1
	-O2 -ipo	6.7	16.5
	-O2 -ipo -vec -xCORE-AVX512	4.5	24.6
	-O2 -ipo -vec -xCORE-AVX512 -parallel -par-num-threads=62 -par-schedule-guided=1	0.16	693.7
	-O2 -ipo -vec -xCORE-AVX512 -parallel -par-num-threads=62 -par-schedule-guided=1 -par-affinity=granularity=fine,scatter	0.141	787.2

- -vec -xAVX
  启用向量化并告知优化 AVX 指令集
  其他指令集参考
- -ipo
  启用过程间优化。PGO 第一阶段无法使用。
- -parallel -par-num-threads=14 -par-schedule-guided=1
  启用自动 omp 并行，设置线程数14，策略为(guided,1)
  -par-affinity=granularity=fine,scatter
  指定线程关联。在启用了 Intel®超线程技术(Intel®HT Technology)的 Intel 处理器上，将 OpenMP 线程绑定到特定的包和内核通常会提高性能。fine 指最好的粒度级别。使每个 OpenMP 线程绑定到单个线程上下文。指定 scatter 将线程尽可能均匀地分布在整个系统中。
  具体参考
- 使用 PGO
  1. 使用-prof-gen -prof-dir，生成可执行文件
  2. 运行可执行文件，将自动采集信息，输出在 prof-dir 下
  3. 使用-prof-use -ipo -prof-dir 再次生成可执行文件
  4. 运行可执行文件，本次运行将结合 prof-dir 指导文件，可能会更快

  具体参考
  本项目 readme.md 记录了 GPO 使用。
  测试后发现对于串行程序有更好的微弱优化，以及当-par-affinity=granularity 存在时，反向加速。