LiXiang
宁可抱香枝上老，不随黄叶舞秋风

比较OpenBLAS，Intel MKL和Eigen的矩阵相乘性能

发表于2014-03-14

对于机器学习的很多问题来说，计算的瓶颈往往在于大规模以及频繁的矩阵运算，主要在于以下两方面：

- (Dense/Sparse) Matrix – Vector product
- (Dense/Sparse) Matrix – Dense Matrix product

如何使机器学习算法运行更高效摆在我们面前，很多人都会在代码中直接采用一个比较成熟的矩阵运算数学库，面对繁多的数学库，选择一个合适的库往往会令人头疼，这既跟你的运算环境有关，也跟你的运算需求有关，不是每个库都能完胜的。

这篇文章的主要目的就是比较几个常见的BLAS库的矩阵运算性能，分别是

1. **EIGEN**: 是一个线性算术的C++模板库。功能强大、快速、优雅以及支持多平台，可以使用该库来方便处理一些矩阵的操作，达到类似matlab那样的快捷。 需要定义 EIGEN_NO_DEBUG 阻止运行时assertion。编译单线程版本需要开启 -DEIGEN_DONT_PARALLELIZE. 在试验中，我们采用 EIGEN 原生 BLAS 实现。
2. **Intel MKL**: 英特尔数学核心函数库是一套经过高度优化和广泛线程化的数学例程，专为需要极致性能的科学、工程及金融等领域的应用而设计。它可以为当前及下一代英特尔处理器提供性能优化，包括更出色地与 Microsoft Visual Studio、Eclipse和XCode相集成。英特尔 MKL 支持完全集成英特尔兼容性 OpenMP 运行时库，以实现更出色的 Windows/Linux 跨平台兼容性。在试验中的多线程版本需要链接到 mkl_gnu_thread，而不是 mkl_intel_thread，单线程版本需要链接到 mkl_sequential_thread。
3. **OpenBLAS**: 是一个高性能多核 BLAS 库，是 GotoBLAS2 1.13 BSD 版本的衍生版。OpenBLAS 的编译依赖系统环境，并且没有原生单线程版本，在实验这哦那个，通过设置 OMP_NUM_THREADS=1 来模拟单线程版本，可能会带来一点点的性能下降。

每个测试程序的编译都采用 “-O4 -msse2 -msse3 -msse4” 优化， 通过设置 OMP_NUM_THREADS 来控制使用的线程数量. 除了 OpenBLAS，其他两个库的测试程序都分别有单线程和多线程的编译版本。

如果MKL编译出现问题，建议参考Intel Math Kernel Library Link Line Advisor

- 单线程版本

我在实验中进行了一系列的非稀疏矩阵相乘运算，矩阵规模也逐渐增大，单线程的运行时间如下表所示，其中采用的测试轮数为5轮，其中红色表示性能最好的一组实验结果。

Matrix-Dimension	Eigen	MKL	OpenBLAS
500	0.04159	0.03122	0.03058
1000	0.31789	0.24339	0.23730
1500	1.04589	0.81445	0.79869
2000	2.37567	1.92036	1.87102
2500	4.68266	3.78569	3.64548
3000	8.28073	6.42630	6.29797
3500	13.07470	10.25096	9.98417
4000	19.34550	15.21931	14.87500
4500	27.52767	21.45024	21.18227
5000	37.67552	29.31631	29.07229

从图中可以看出，OpenBLAS的性能最好，MKL的表现也很不错，而EIGEN的表现却很糟糕。

- 多线程版本

在多线程的测试中，我们采用多个CPU核心来做矩阵乘法运算，所有的结果也同样采用5轮训练，我们采用的CPU核数分别是8，16，32，48。

- Cores = 8

Matrix-Dimension	Eigen	MKL	OpenBLAS
1000	0.05658	0.03955	0.06468
2000	0.34981	0.26200	0.23879
3000	1.20781	0.85449	0.80737
4000	2.65490	1.90273	1.88366
5000	5.03304	3.73005	3.67966

6000	8.78654	6.52766	6.31980
7000	13.55611	10.13758	10.07120
8000	19.81634	15.03530	14.89440
9000	29.11329	21.54359	21.26992
10000	39.01563	29.93075	29.22034

- Cores = 16

Matrix-Dimension	Eigen	MKL	OpenBLAS
1000	0.05708	0.02185	0.03897
2000	0.26694	0.13807	0.30461
3000	0.70686	0.43692	0.93511
4000	1.45129	0.97720	2.06761
5000	2.59477	1.90665	2.49280
6000	5.43438	3.30945	7.01299
7000	8.01124	5.17896	6.84496
8000	11.22280	7.81439	12.99240
9000	15.15625	11.08906	21.82488
10000	19.91151	15.22039	30.86908

- Cores = 32

Matrix-Dimension	Eigen	MKL	OpenBLAS
1000	0.04003	0.02792	0.02244
2000	0.51213	0.14363	0.16990
3000	1.13647	0.51105	0.54635
4000	1.58793	1.10219	1.26401
5000	2.88341	2.07923	2.48735
6000	5.92779	3.42785	4.26794
7000	7.91650	5.32176	6.69391

8000	11.96467	7.65395	9.98951
9000	17.45420	10.28328	14.14108
10000	23.31314	15.10077	19.34171

- Cores = 40

Matrix-Dimension	Eigen	MKL	OpenBLAS
1000	0.03691	0.02877	0.01779
2000	0.37739	0.14037	0.13655
3000	0.61183	0.41057	0.44113
4000	2.43670	1.02625	1.01414
5000	3.18099	1.91092	1.97898
6000	8.24002	2.96157	3.40685
7000	11.59889	4.68312	5.38634
8000	9.50613	6.98434	7.95971
9000	14.83066	9.60891	11.37585
10000	23.67187	15.52151	15.52680

- Cores = 48

Matrix-Dimension	Eigen	MKL	OpenBLAS
1000	0.03635	0.02398	0.01548
2000	0.36417	0.13408	0.11496
3000	2.32388	0.39291	0.36669
4000	2.32030	1.13244	0.85790
5000	2.08269	1.75812	1.66785
6000	8.70766	2.98694	2.85609
7000	8.23543	4.62340	4.53257
8000	21.18603	6.68886	6.72820

9000	19.86504	9.59635	9.50597
10000	16.10920	13.13038	13.04432

可以看出，MKL和OpenBLAS都提供了比较好的性能，MKL性能还更好一点，在各别多线程条件下了，可能某些原因或者我机器设置的问题，出现了各别性能异常，比如小矩阵运算时间反倒比大矩阵运算长，或者更多的线程却不能提供更好的性能。这些情况后面可能还需要查一查。

- 伸缩性

  另外，我也测试了使用不同的cpu核数对性能的影响，下面两个图描述了把cpu从1增加到20的条件下，5000×5000的矩阵相乘的时间开销和加速比。

- 结论

  就我的测试环境而言，Intel MKL 和 OpenBLAS 似乎是矩阵相乘运算方面性能最佳的 BLAS 库，在多核以及不同规模的矩阵方面都具有较好的伸展性和稳定性，而对于单线程情况，OpenBLAS相比 MKL 在性能上有一定提升。

  本文参考gcdart的文章，代码可以下载。

此条目由**lixiang**发表在**Machine Learning**分类目录，并贴了**BLAS**、**Eigen**、**Intel MKL**、**OpenBLAS**标签。将固定链接 **[http://www.leexiang.com/the-performance-of-matrix-multiplication-among-openblas-intel-mkl-and-eigen]** 加入收藏夹。

《比较OPENBLAS，INTEL MKL和EIGEN的矩阵相乘性能》上有2条评论

celery
在**2014-07-3118:38**说道：

请问你的机器是什么配置，我的矩阵乘法单核2048大小的双精度矩阵乘法大概4s，机器为XP系统，I3,2.1Ghz，三级缓存3M。

andy
在**2015-09-2111:24**说道：

你好，请问你有这个代码吗？我用的Eigen库时间运行效率很低