BLAS Implementation of Matrix Multiplication

B05902073 資工二 張庭與

Abstract

Due to the difference of memory alignment and algorithm implement between different programming languages, it take us different amount of time to multiply two large matrices. By using BLAS, or Basic Linear Algebra Subprograms, we can definitely save our time.

Methods

※ All of the running results based on a laptop with Intel i7-6700HQ CPU at about 3.00 GHz (quad-core, with Intel HT and Turbo boost on), and dual-channel DDR4-2133 memory modules. The matrices are 8000-by-8000-sized if not specified.

1. Naïve methods (Single-Threaded)

In most cases, naïve method is the most time-consuming way of doing things, including multiplication of matrices.

We use three implements of naïve methods (in C programming language) to test the efficiency.

Matrix alignment	Multiply orientation	Real time User time System time
Linear	$C_{ij} = A_{ik} \times B_{kj}$	At least 4 Hours[1] Unknown Unknown
2-Dimensional	$C_{ij} = A_{ik} \times B_{kj}$	44’ 49” 44’ 44” 0” 52
Linear	$C_{ik} = A_{ij} \times B_{jk}$	32’ 5” 32’ 2” 0” 67
2-Dimensional	$C_{ik} = A_{ij} \times B_{jk}$	6’ 43” 6’ 42” 0” 16

1. Run for whole evening but not finished.

By comparing between 2-D and Linear matrix alignment, we can find that 2-D methods are faster than their linear counterpart. Moreover, the multiplication orientation also affects the efficiency of algorithm. The upper two is slower due to more leaping between unsuccessive memory addresses.

2. Intel MKL and CBLAS(Single-Threaded)

Using Intel MKL library reduces running time significantly from (best result of naïve methods) 6′ 42″ to 0′ 22″ 88. This shows that we have better knowing good libraries than reinventing the square wheel.

However, CBLAS library reduces running time much less effectively, only from 6′ 42″ to 5′ 8″ .

3. OpenBLAS (Single-Threaded or Multi-Threaded)

The singled-threaded OpenBLAS implement is as almost fast as that of Intel KML. Therefore, we can focus on the influence of multi-threading, and relation between efficiency, multi-threading and job size.

Job Size	Single-Threaded	2 Threads	4 Threads (All Cores)	8 Threads (All Cores, Intel HT)	Fastest Result Real Time / Threads	
500	0”014	0”013	0”012	0”013	0”012	
by	0”004	0”019	0”006	0”031	0”006	4
500	0”010	0”004	0”008	0”018	0”008	
1000	0”061	0”038	0”027	0”031	0”027	
by	0”043	0”052	0”067	0”105	0”067	4
1000	0”016	0”016	0”063	0”055	0”063	
2000	0”401	0”218	0”154	0”149	0”149	
by	0”383	0”395	0”508	0”953	0”953	8
2000	0”016	0”024	0”078	0”172	0”172	
4000	3”156	1”590	1”050	1”310	1”023	
by	3”039	3”102	3”844	12”055	6”571	7
4000	0”094	0”071	0”321	0”665	0”586	
8000	23”86	12”40	8”09	11”77	8”09	
by	23”59	24”61	30”89	1’30”56	30”89	4
8000	0”20	0”17	1”38	2”72	1”38	

※ For precision, we take average of 1, 2, 4, 8, 8 samples for 8000 by 8000, 4000 by 4000, 2000 by 2000, 1000 by 1000 and 500 by 500, respectively. All tests run under the condition that only OS running, too.

In the physical test, it ran from all 1-threaded test to 8-threaded tests, but results between 4-threaded tests and 7-threaded tests had not many differences, and 8-threaded tests is almost as slow as 3-threaded tests . The 7-threaded tests actually reached 100% CPU usage as well as 8-threaded tests, but in most cases, 8-threaded tests is slower.

Indeed, there is a huge improvement from single-threading to multi-threading in large test cases, 4-thread tests is roughly three times faster than single-thread. We can also see that user time increases drastically when all eight threads are busy, but the performance is not as good as that of 4-threads. Furthermore, it takes more calculating resources to finish the same job when multithreading, especially when Intel HT is enabled.

Conclusion and Impression

There are so many (optimized) libraries on the Internet for us to use, why bother reinventing the (square) wheel? I personally love this idea since I have been finding some libraries useful. More than this, libraries not only bring up convenience accesses to some functions, but also enhances program efficiency.

On the other hand, although the use of multi-core CPUs and multi-threading seems increases the speed of programs, we should be aware of if it is an illusion of consuming more calculating resources.

(I have done some Google and Wikipedia searching on Intel HT.) When the processes requires large quantities of the same calculating resources, Intel HT actually slows down the efficiency of those processes. This is the case when we use eight threads in OpenBLAS tests.