## 1 Topic

**Influence of hyper-threading technology and a SSD hard disk on the processing time of some machine learning algorithms available in Sipina.**

After more than 6 years of good and faithful service, I decided to change my computer. It must be said that the former (Intel Core 2 Quad Q9400 2.66 Ghz - 4 cores - running Windows 7 - 64 bit) began to make disturbing sounds. I am obliged to put music to cover the rumbling of the beast and be able to work quietly.

The choice of the new computer was another matter. I spent the age of the race to the power which is necessarily fruitless anyway, given the rapid evolution of PCs. Nevertheless, I was sensitive to two aspects that I could not evaluate previously: The hyper-threading[1] technology is effective in programming multithreaded algorithms of data mining? The use of temporary files to relieve the memory occupation takes advantage of SSD[2] disk technology?

The new PC runs under Windows 8.1 (I wrote the French version of this tutorial one year ago). The processor is a Core I7 4770S (3.1 Ghz). It has 4 physical cores but 8 logical cores with the hyper-threading technology. The system disk is a SSD. These characteristics allows evaluate to their influences on (1) the implementation of multithreaded version of the linear discriminant analysis described in a previous paper ("Load balanced multithreading for LDA", September 2013), where the number of threads used can be specified by the user; (2) the use of temporary files for the induction of decision trees algorithm, which enables us to handle very large dataset ("Dealing with very large dataset in Sipina", January 2010; up to 9,634,198 instances and 41 variables).

In this tutorial, we reproduce the two studies using the SIPINA software. Our goal is to evaluate the behavior of these solutions (multi-threaded implementation, copy of data into temporary files to alleviate the memory occupation) on our new machine which, due to its characteristics, should expressly take advantage of them.

## 2 Multithreaded implementation of LDA

In this section, we revisit a tutorial published in September 2013 ("Load balanced multithreading for LDA"). The idea was to propose a multi-threaded implementation of the

[1] https://en.wikipedia.org/wiki/Hyper-threading

[2] https://en.wikipedia.org/wiki/Solid-state_drive

linear discriminant analysis and use the capabilities of multi-core processors which are incorporated into the most of the computers today. One of the issues was to balance the loads better in order to avoid that some cores remain idle when others are working. We show that for the solution implemented in Sipina, the reduction of the calculation time was almost proportional to the number of threads requested, as long as that it did not exceed the number of available cores.

With my new computer, I wondered if when the number of threads exceeds the number of physical cores, the hyper-threading technology still enables to improve the computing time, knowing that it should not exceed the number of logical cores. I therefore reiterate the experimentation on the MIT FACE IMAGES file corresponding to a binary classification problem with 513,345 instances and 361 descriptors. On my old computer (Quad Core 9400 - Windows 7 - 64 bit - No hyper threading technology), the processing time with one thread was 142.73 seconds; with 4 threads, we obtain 37.75 seconds. When the number of threads exceeds the number of available cores, the gain is zero.

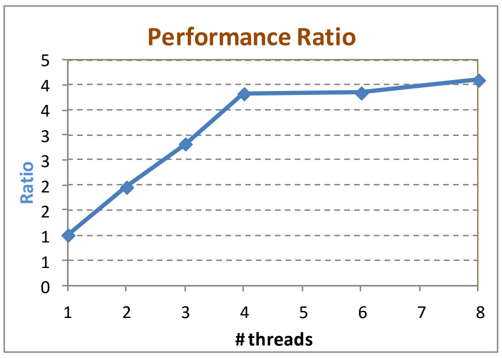
We perform the same analysis on the new PC, by increasing the number of threads (1, 2, 3, 4, 6, 8). We analyze the performance improvement with the following indicator:

$$Ratio = \frac{Time\ single\ thread}{Time\ multiple\ thread}$$

We obtain the following results:

| Threads | Comp. Time (sec.) | Ratio |
|---|---|---|
| 1 | 106.44 | 1.00 |
| 2 | 54.38 | 1.96 |
| 3 | 37.70 | 2.82 |
| 4 | 27.81 | 3.83 |
| 6 | 27.70 | 3.84 |
| 8 | 25.97 | 4.10 |

We can visualize the growth of ratio in a chart.



We can make several comments here:

- First of all, SIPINA detects the 8 logical cores when we ask the number of available processors.
- The increase in performance (or the reduction of processing time) is almost proportional to the number of threads used, as long as **it remains within the limits of the number of physical cores**.
- When it exceeds this threshold (4 cores for our machine), the gain is negligible, while we remain within the limits of the number of logical cores.

It is obvious that hyper-threading is not relevant in our implementation of the linear discriminant analysis. It is not easy to clearly identify the reasons for this result. The phenomenon of hyper-threading performance degradation is discussed in several articles on the web[3]. This result is confirmed by another experiment that I conducted on a laptop equipped with a Core i5 processor with 2 physical cores, but with 4 logical cores based on the hyper-threading technology.

Similarly, I reiterate my experimentation for the multithreaded decision trees algorithm. In this context also[4], the passage of 4 to 8 threads does not improve the computing time.

## 3 Temporary files on SSD drive

In this section, we revisit a tutorial published in January 2010 ("Dealing with very large dataset in Sipina"). The idea was to use temporary files to relieve the memory occupation. The dataset is organized in columns i.e. one column corresponds to one variable, and it is stored in a distinct file. Accesses are therefore performed directly on disk during the execution of the machine learning algorithms. A caching system is used in order to improve reading time. We used the version 3.2 of Sipina. We are in the single thread context here.

SSDs have no moving mechanical components. They have a lower access time. In our case, we are interested by the speed of reading the data stored in temporary file in the context of the decision tree induction. I was very curious to measure the processing time improvement in comparison to the standard disk storage.

Since version 3.8, Sipina uses the FASTMM memory manager which allows addressing more memory. Compared to the previous tutorial, it is now possible to load all data in main

[3] See for instance "When hyper-threading hurts", https://bitsum.com/pl_when_hyperthreading_hurts.php.

[4] « Multithreading for decision tree induction », November, 2010.

memory (9,634,198 instances, 41 descriptors + class attribute). We can compare the processing time when the data are all loaded in memory, and when we store them in temporary files on hard drive. The experimentation carried out on my old PC with a standard hard disk is performed on my new computer with a SSD system. We will see if the transition to the SSD is really beneficial when treatments involve repeated disk accesses.

We calculate a new ratio:

$$Ratio = \frac{Comp.Time\ disk}{Comp.Time\ main\ memory}$$

We summarize the results into the following table:

| Comp. Time (sec.) | Q9400 + Std HD | Core i7 + SSD HD |
|---|---|---|
| Processing in memory | 210.5 | 89.9 |
| Processing on disk (Temp. Files) | 271.4 | 114.6 |
| **Ratio** | **1.3** | **1.3** |

We can make several comments here:

- The new computer is approximately twice more powerful than the old one. We observe this result when all the calculations are performed exclusively in main memory.
- The calculations from the data stored on disk affects the performance compared with the treatment in memory. This is obvious. But we see that the degradation remains very reasonable given the volume to handle.
- The ratio is the same (≈ 1.3) when we use a SSD or a standard hard disk. There, I confess that is not what I expected. The strategy that I have implemented to accelerate treatments during the construction of the trees does not take advantage of the performance of SSD.

To understand this behavior which seems surprising, I inspect the Sipina source code. I realized that the class attribute is always in main memory during the construction of the tree. Then, for the evaluation of the candidates splitting variables, I temporarily load the entire column in memory. We understand why it is so important to organize data in columns in this context. The additional cost (the ratio of 1.3) is the consequence of the successive loading and unloading of data blocks. The memory occupation remains contained because, at worst, we have two columns of values. And the increased performance of the SSD drive is not crucial because we read blocks of columns.

## 4 Conclusion

I have mixed feelings about these experiments. Obviously, I have a new more powerful machine in comparison with the old one. We note this when we launch single threaded processing in memory. But the specific characteristics of the new PC are disappointing. On the one hand, the hyper-threading technology does not enable us to take advantage of the increased power by specifying more threads than physical cores for the multithreaded algorithms that I programmed in Sipina, even if we remain within the limits of the number of logical cores. On the other hand, the system disk SSD used for the storage of temporary files is not decisive for my implementation of decision tree algorithm.

The results are certainly reserved. But the key issues remain. The computer technology evolves very quickly and offers us new opportunities constantly. It is our role, researchers, to take advantage of these new characteristics for our implementations of machine learning algorithms, in order to make our programs more efficient.