科技部補助專題研究計畫成果報告
期末報告

分解合成系列快速計算法之平行化改進

計 畫 類 別 ：個別型計畫
計 畫 編 號 ：NSC 102-2115-M-004-003-
執 行 期 間 ：102 年 08 月 01 日至 103 年 07 月 31 日
執 行 單 位 ：國立政治大學應用數學學系

計 畫 主 持 人 ：曾正男

計畫參與人員：碩士班研究生-兼任助理人員：林昱

報 告 附 件 ：出席國際會議研究心得報告及發表論文

處 理 方 式 ：
1.公開資訊：本計畫涉及專利或其他智慧財產權，1 年後可公開查詢
2.「本研究」是否已有嚴重損及公共利益之發現：否
3.「本報告」是否建議提供政府單位施政參考：否

中 華 民 國 103 年 10 月 14 日

中 文 摘 要 ： 本計畫是將 split and combine 系列的演算法推廣到平行計算，我們利用 Python 程式的架構成功的將此系列的加速法推廣到多核心計算以及 GPU 的計算。對於 split and combine 系列的計算法如何分群是一個重要的關鍵，我們經過這次的計畫補助理解了如何有效率的分群，讓分解合成計算更有效率。特別在資料維度高以及資料量大時，我們平行化的版本程式表現會更為優異。

中文關鍵詞： 分解合成，多元尺度，平行化

英 文 摘 要 ： We have implement the SCMDS method from the serial version to the multicore version. We use the python programming to do this work. Our project gives a very friendly introduction for parallel programming in python. In our experiments, we can see that the performance of multicore version of SCMDS makes the linear SCMDS better. When the data dimension is large and the size of data is huge, the performance of parallel version is pretty well. This parallel SCMDS is proved to be good for large data analysis.

英文關鍵詞： Split-and-combine, MDS, parallel

Project title:

The parallel computing implement for split-and-combine series techniques

Contents:

1. Preface

This project is started in 2013/8/1. One master student is hired as the assistant and he is Mr. Yu Lin. He helped me to collect some experimental data and write some simple coding. Another master student, Mr. Pei-Chen Li, produces the main results of this project. He gives a very easy user guide of parallel computing by python programming and complete this project in the multicore version code.

This project, "The parallel computing implement for split-and-combine series techniques", is an implement of the SCMDS method. The SCMDS method is a fast algorithm for the CMDS method, which is a famous dimensional reduction method that represents data from high dimensional space into the low dimensional space.

Represent data in the low dimensional space and preserve the data structure is very important in recent ages. Due to the curse of dimensionality, the classifier performance decreases with the dimensionality increasing. We have to present data in the lower dimensional space. The dimensionality is as lower as better. MDS method is one of the techniques in the dimensional reduction techniques. Most of the dimensional reduction techniques are undetermined. That is the result changes when we start with different initial guess. CMDS is the determined method, that's why we try to develop the fast algorithm of CMDS.

The SCMDS method splits data into the overlapping subgroup, applies CMDS method on each group to obtain the data coordinate and then uses the information of the overlapping regions to combine each CMDS results into

one. SCMDS makes the $O(n^3)$ traditional method to the $O(n)$. When the data is small, the $O(n^3)$ is not a big deal for computing. However, when the data is huge, the $O(n^3)$ algorithm becomes infeasible. Thus SCMDS is better to use in the large data analysis.

From the main idea of the SCMDS, we have also implement the traditional PCA method and SVD method to the fast algorithm with the constraint that the numerical dimensional is low rank. They are called the SCPCA and SCSVD respectively. Hence, we called the SCMDS, SCPCA and SCSVD as the SC-serials methods. The flow of the SCMDS is as the following.

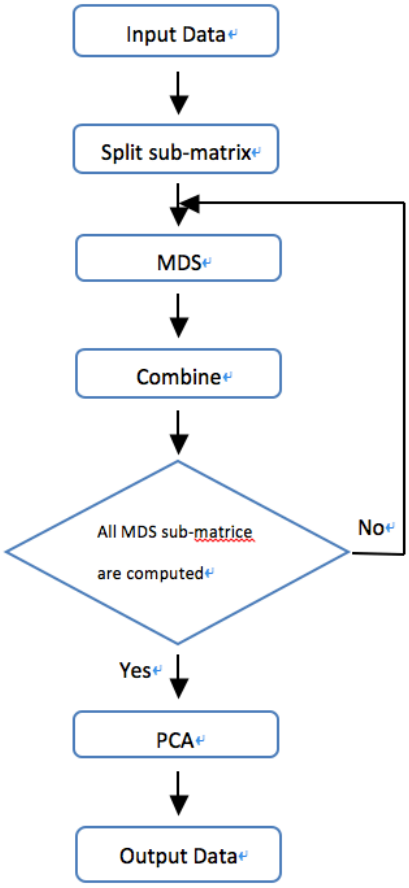

Figure 1.

In figure 1 the MDS step after the split sub-matrix can be assigned to difference computing units, it is natural to do the parallel version implementation of SCMDS method. If we can have a parallel version of

SCMDS, then we can easily to construct the parallel version of SCPCA and SCSVD.

2. Research purposes

The research purpose of this project is to develop the parallel version of SCMDS, SCPCA and SVSVD methods by python programming. Python is more and more popular in recent year. It has many good features for scientific computing, for example, it is cross platform, it is easily to embed other language like C++ and Matlab, and it has many modules for each application field. Since python has many advantages in the scientific computing, we want to create the parallel python code for these methods that researchers or students can use these codes for their research. We have to study how to control the parallel flow and know the cost for each step. Moreover, we can know the best strategy to divide the whole data into subgroups.

When process the parallel computing, we have to separate data into the different computing units. The transmission cost of data is one of the important issues to discover. What is the cost for use to send data into the share memory; what is the cost to send data into each difference core; what is the cost for CMDS in each core; what is the cost for QR in each core; and what is the optimal setting for the overlapping region and number of each subgroups. These problems would be transformed to a mathematical optimization problem. We will try to look for the solution.

3. Literature reviews

After the python version of 2.6, there exist the parallel solutions for multicore CPUs or shared memory environment, or the cluster environment. For cluster we use Python pp module to write the python code on SMP (systems with multiple processors or cores) and clusters. If we only want to write the multicore program on single computer, we can use the multiprocessing module to do the parallel computing.

We will give a brief introduction for how to use python multiprocessing module as the following.

After import the multiprocessing module, we can use the *cpu_count* function to know how many cores in your computer. For example:

```
In [1]: import multiprocessing as mp

In [2]: mp.cpu_count()
Out[2]: 2
```

In the line [1] we import *multiprocessing* module and rename it as *mp*, the line [2] show that my computer has two cores CPUs.

The *Pool* function in the *multiprocessing* module will return a process pool object. In the Pool function, the processes parameter is a controller to setup the number of CPUs. That is we can use this function to control how many computational unit will be ready to the following processing. For example:

```
In [6]: po = mp.Pool(processes=4)
```

We set the Pool have two real cores and two virtual cores (since my hardware has only two cores) for further computing.

The *map_async* function can assign a function job to process and return the function value to a vector variable. Then we can collect the parallel results from the vector. We give a simple example to demonstrate the above concepts.

Assume we want to compute the value of $\sum_{x=1}^{8}\sum_{k=1}^{100} x \cdot k^2$, we have the following code named simple.py .

```
 1 from multiprocessing import Pool
 2 import time
 3 K = 100
 4
 5 def Fx((x,)):
 6   r = 0
 7   for k in xrange(1,K+1):
 8     time.sleep(0.01)
 9     r+=x*(k**2)
10   return r
11
12 def sum_Fx(N=8,ncpu=4):
13   if __name__ == "simple":
14     tic = time.time()
15     po = Pool(processes = ncpu)
16     res = po.map_async(Fx,((i,) for i in xrange(1,N+1)))
17     w = sum(res.get())
18     toc = time.time()
19     print toc-tic
20     return w
```

In this simple python parallel code, we define two functions, *Fx( )* and *sum_Fx( )*. The first one is to compute the value of $\sum_{k=1}^{100} x \cdot k^2$ for given x. This function is the serial code, in which will be used for each core. The second code is the parallel part. In line [13], if we check the function comes from the *simple.py* file, then we start the parallel computing. In line [16], we will get a vector type return. In each element of the res vector, it is the result for $\sum_{k=1}^{100} x \cdot k^2$ for given x. In line [17], the *res.get( )* statement will get the value from the memory, then we use *sum( )* function to get the summation of this vector.

```
In [1]: import simple

In [2]: simple.sum_Fx(20,4)
6.50191712379
Out[2]: 71053500

In [3]: simple.sum_Fx(20,2)
11.9099731445
Out[3]: 71053500

In [4]: simple.sum_Fx(20,1)
21.6615281105
Out[4]: 71053500
```

When we have the simple.py file, we can run the above commands in the same folder to obtain the parallel result. We can see that they have the same summation output 71053500. When we use 4 cores, the total time is 6.5019; when we use 2 cores, the total time is 11.9099; and when we use single core to

compute it, the time is 21.6615. The time decay performance is linear as we wish.

Another important thing is to control the share memory due to the bester performance of parallel computing. When the data is huge, we should not copy the same data again and again for each core computation. The best way for us is to locate a certain memory that each core can read the same data from one source. That is the main idea of the shared memory to be used. The following code is how we define a shared memory object in python.

```
11 class shared_array(object):
12   def __init__(self,matrix_shape=(1,1)):
13     (m,n) = matrix_shape
14     self.shared_base = mp.Array(ctypes.c_double,m*n)
15     self.array = np.ctypeslib.as_array(self.shared_base.get_obj())
16     self.array = self.array.reshape(m,n)
17     self.shape = self.array.shape
```

In the *multiprocessing* module, we can use *Array( )* function to create the shared memory. We have to use *ctypes* module to allocate the memory type and size.

In line [12], we set the default size of the matrix is 1-by-1. If the input argument is (m,n), we will set the matrix size to m-by-n. In the lines [14-15], we create a zero matrix with size m-by-n. In line [16], we reshape the matrix to the rectangular form. If some variable *A* is defined as the shared memory, *A.array* will show the items of the matrix in the shared memory. In line[17], this will make *A.shape* show the matrix size of the shared memory.

```
In [2]: A = shared_array()

In [3]: A.array
Out[3]: array([[ 0.]])

In [4]: A.shape
Out[4]: (1, 1)

In [5]: A = shared_array((2,4))

In [6]: A.array
Out[6]:
array([[ 0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.]])
```

The previous code shows how to create a shared memory. In line [2], we create a variable *A* to be a 1-by-1 zero matrix in the shared memory. In line [5], we create the 2-by-4 zero matrix in the shared memory. We use *A.array* in the line [6] to show contains of *A*.

After we can create the shared memory, we must know how to transmit data from local memory to shared memory. The following function will copy the array data from the local to the shared memory.

```
32 def array2shared(A,shared_A):
33   m1,n1 = A.shape
34   m2,n2 = shared_A.shape
35   if m1<>m2 or n1<>n2:
36     print 'The size of matrices must be the same'
37     return
38   else:
39     for i in range(m1):
40       shared_A.array[i] = A[i]
```

From line [33] to line [37], we will check weather the local matrix size and the desired shared memory size are the same. If not, we will return an error message. It is very easy to copy the data from the local memory to the shared memory. We can copy the contents by each row instead of copying them by each element.

One important function in parallel computing is to synchronize the processing. In the module of multiprocessing in python, we use *Lock( )* function to make sure all the synchronized processes have been done. Another important module is *affinity*, we can use this module to check current number of cores in process. The detail of these functions and modules can be found on the web.

4. Methodology

As the previous introduce, we can make the CMDS step and the QR step of the SCMDS become a function type processes. Therefore, we can easily implement the SCMDS to the parallel form. Compare with the serial code of SCMDS in figure 1, the following is the flaw of the parallel SCMDS.

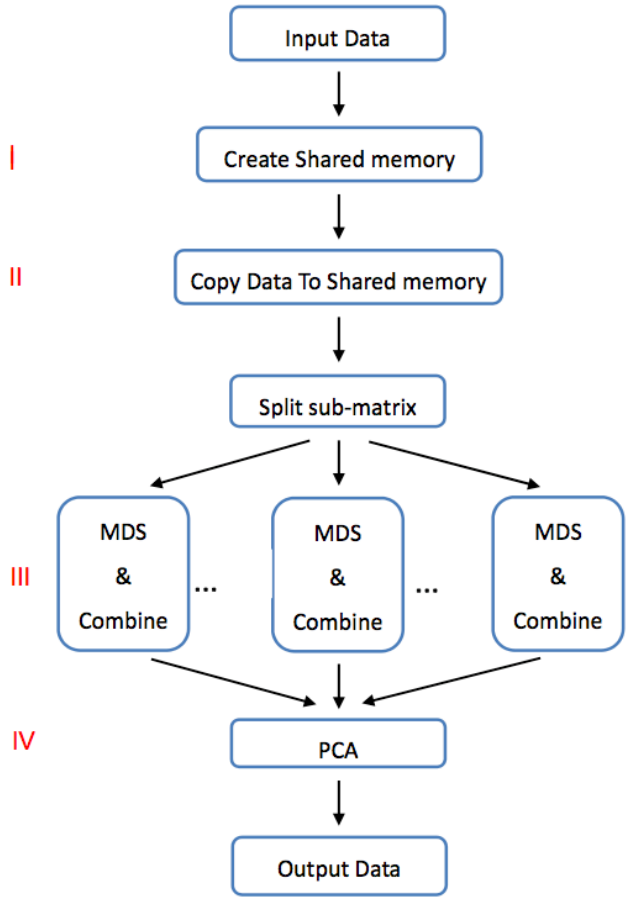

Figure 2. The flaw of parallel SCMDS

The part (I) is creating the shared memory; the part (II) is copy the data into the shared memory. Since the input data is a huge matrix, we will split the matrix into the overlapping sub-matrix first and then compute each MDS result by parallel in part (III).

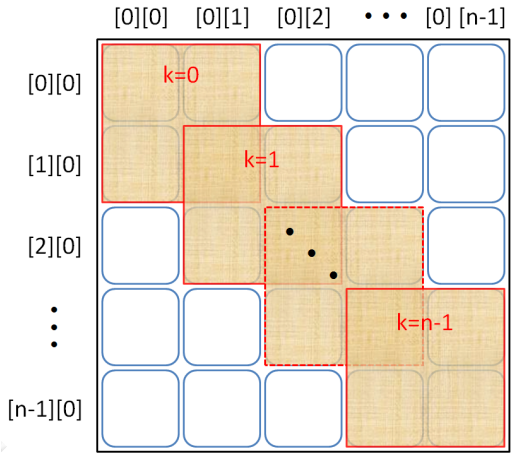

Figure 3. The overlapping sub-matrix

Figure 3 is the concept of overlapping sub-matrix. Each sub-matrix must overlap to other sub-matrix. After we have each MDS result of the sub-matrix, we will combine them into one MDS result.

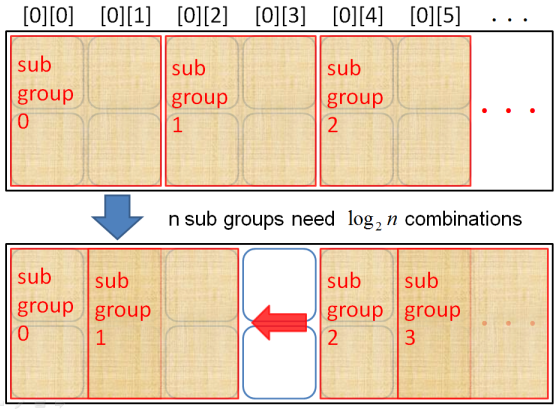

Figure 4. The combination process of SCMDS

Figure 4 is the concept of combination processing. We use the information in the overlapping region to combine two MDS configurations into one. In this grouping method, we will combine the MDS result sequentially. After combine the MDS results into one, we will process PCA again to get the whole configures of true MDS data.

Since the CMDS result has the property that the first coordinate refers to the first component of data, we process PCA in the final step. This PCA is dominated by the reduced size. Hence, its cost is not heavy in general. In parallel version, the total computing speed is saved by the MDS step in part (III) of figure 2. Another heavy cost appears also in the combination step in part (III). However, if we want to make the combination step can be parallelized, we have to change the grouping method as the following

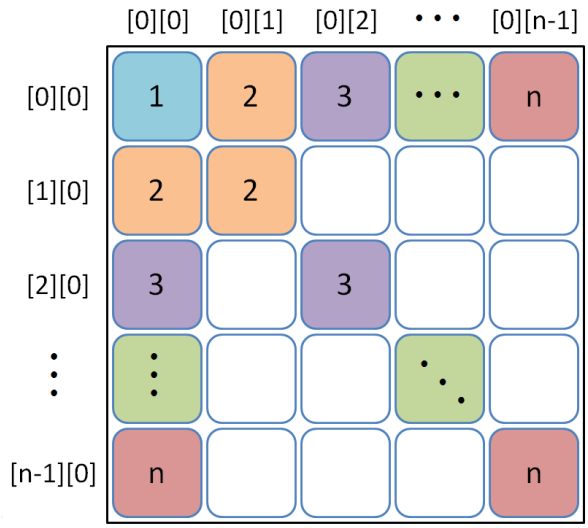

Figure 5. The new grouping method.

We can see that every sub-matrix has the same overlapping part that is in the upper left corner. When every sub-matrix use the part 1 region of Figure 5 to combine the MDS result for i-th sub-matrix for $i \geq 2$, we don't wait the previous sub-matrix for synchronization. That is every sub-matrix can do the combination step with the first one sub-matrix. All that we do is make the MDS result of the first sub-matrix complete before other sub-matrix process the combination step. And then we can make the combination step also parallel. We will show that this improvement makes the parallel version much better then before.

5. Experimental results

We use the AMD Opteron™ Processor 6128 2GHz CPU for our experimentation. It has 8 cores. The memory of computation server is 64 GB with the Ubuntu 12.04 64 bit OS and Python of 2.7.1. We will compare the serial version and the parallel version of SCMDS. And in the parallel version of SCMDS, we will also compare the performance between different numbers of cores.

For the advantage of SCMDS, we use three kinds of dimensional data as our testing data. The true dimension of data are 50, 100 and 150 respectively. The sample size of testing data is 3000, 6000, …, 15000. And we will compare the serial version, 2 cores, 4 cores and 8 cores. In the case of true dimension p is

50, we assume the number of overlapping $N_i = p + 1$ and the number of group $N_g = 2.2 * N_i$. For every combination of parameter, we will repeat 16 times to obtain the average as our result.

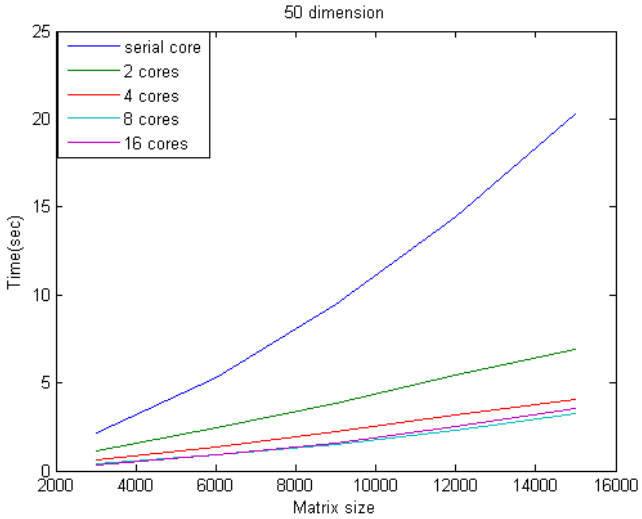

Figure 6.1 Comparison time cost for different cores (p=50)

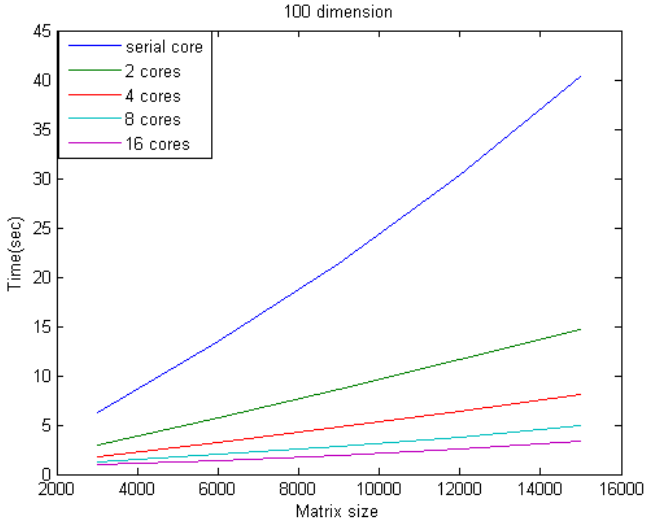

Figure 6.2 Comparison time cost for different cores (p=100)

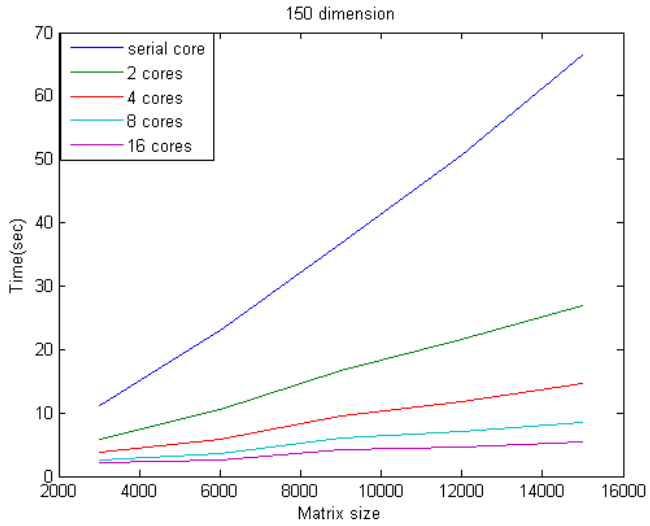

Figure 6.3 Comparison time cost for different cores (p=150)

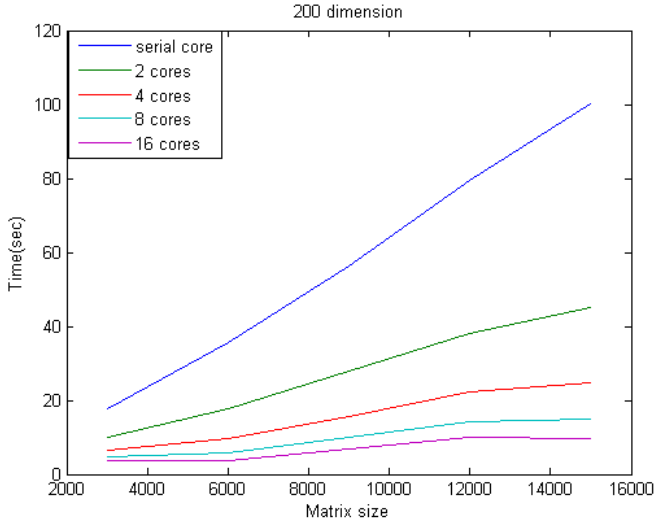

Figure 6.2 Comparison time cost for different cores (p=200)

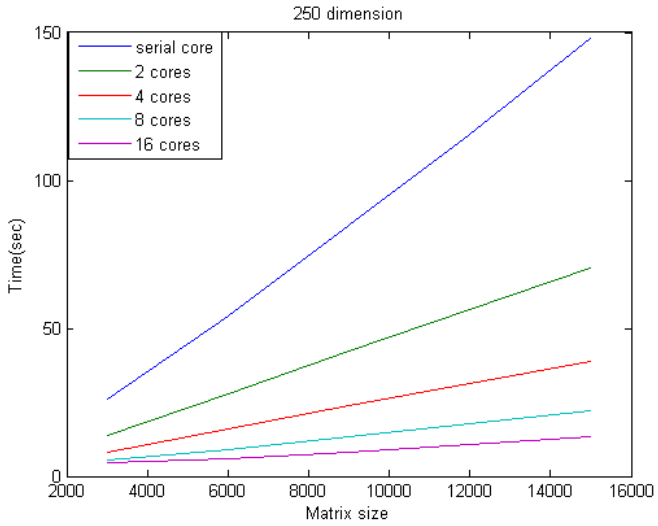

Figure 6.2 Comparison time cost for different cores (p=250)

We can see that it is reasonable that the serial core version is almost linear as the theoretical proof. When we increase the number of cores, the time cost decreases. Although, the double of number of cores not save the half of time, it is almost near the ratio as our expectation.

One more interesting is that when the dimension become larger, the performance will become better. That is the parallel version of SCMDS is better to be used in the large dimensional data.

6. Conclusions

We have implement the SCMDS method from the serial version to the multicore version. We use the python programming to do this work. Our project gives a very friendly introduction for parallel programming in python. In our experiments, we can see that the performance of multicore version of SCMDS makes the linear SCMDS better. When the data dimension is large and the size of data is huge, the performance of parallel version is pretty well. This parallel SCMDS is proved to be good for large data analysis.

7. 精簡報告之篇幅以 4 至 10 頁為原則，完整報告之篇幅則不限制頁數。
   附表及附圖可列在文中或參考文獻之後，各表、圖請說明內容。
   若該計畫已有論文發表者，可作為成果報告內容或附錄，並請註明發表刊物名稱、卷期及出版日期。

Reference:

[1] Ingwer Borg and Patrick J. F. Groenen. Modern multidimensional scaling. Springer Series in Statistics. Springer, New York, second edition, 2005. Theory and applications.
[2] TOIBE Software BV. Tiobe programming community index, 2013. [online] http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html.
[3] Matthew Chalmers. A linear iteration time layout algorithm for visualising high-dimensional data. In Proceedings of the 7th conference on Visualization '96, VIS '96, pages 127–ff., Los Alamitos, CA, USA, 1996. IEEE Computer Society Press.
[4] Pei-Chi Chen. Optimal grouping and missing data handling for split-and-combine multidimensional scaling. 2008.
[5] Jengnan Tzeng, Henry HS Lu and Wen-Hsiung Li : Multidimensional scaling for large genomic data sets , *BMC Bioinformatics* , Vol.9, No.179.(SCI),2008.04.
[6] Pearu Peterson Eric Jones, Travis Oliphant et al. Open source scientific tools for python, 2001. [online] http://www.scipy.org/.
[7] Warren S. Torgerson. Multidimensional scaling. I. Theory and method. Psychometrika, 17:401–419, 1952.
[8] Guido van Rossum. Python tutorial, 2008. [online] http://docs.python.org/2.5/tut/tut.html.

若有與執行本計畫相關之著作、專利、技術報告、或學生畢業論文等，請在參考文獻內註明之，俾可供進一步查考。

This report is one of the part of the thesis of my student Miss Jing-Ru Pan.

Self-assessment of program results:

請就研究內容與原計畫相符程度、達成預期目標情況、研究成果之學術或應用價值、是否適合在學術期刊發表或申請專利、主要發現或其他有關價值等，作一綜合評 估。

We have successfully implemented the serial version of SCMDS to the parallel version. This report is demonstrated by the multicore version. Since the GPU version will be published in the journal paper, we did not show it in this report. Two of the assistants learn how to code the parallel code. One of them got a very good job in IT field and the other will graduate this year. Thanks for the supporting of NSC in this research.

科技部專題研究計畫成果報告撰寫格式

一、說明

科技部基於學術公開之立場，鼓勵一般專題研究計畫主持人發表其研究成果，但主持人對於研究成果之內容應負完全責任。計畫內容及研究成果如涉及專利或其他智慧財產權、違異現行醫藥衛生規範、影響公序良俗或政治社會安定等顧慮者，應事先通知科技部不宜將所繳交之成果報告蒐錄於學門成果報告彙編或公開查詢，以免造成無謂之困擾。另外，各學門在製作成果報告彙編時，將直接使用主持人提供的成果報告，因此主持人在繳交報告之前，應對內容詳細校對，以確定其正確性。

成果報告繳交之期限及種類（期中進度報告及期末報告），應依本部補助專題研究計畫作業要點及專題研究計畫經費核定清單之規定辦理。至報告內容之篇幅，期中進度報告以 4 至 10 頁為原則，並應忠實呈現截至繳交時之研究成果，期末報告不得少於 10 頁。

二、報告格式：依序為封面、目錄、中英文摘要及關鍵詞、報告內容、參考文獻、計畫成果自評、可供推廣之研發成果資料表、附錄。

(一)報告封面：請至本部網站（http：//web1.most.gov.tw）線上製作（格式如附件一）。

(二)中、英文摘要及關鍵詞 (keywords)。

(三)報告內容：包括前言、研究目的、文獻探討、研究方法、結果與討論（含結論與建議）等。

(四)計畫成果自評部分：請就研究內容與原計畫相符程度、達成預期目標情況、研究成果之學術或應用價值(簡要敘述成果所代表之意義、價值、影響或進一步發展之可能性)、是否適合在學術期刊發表或申請專利、主要發現（簡要敘述成果是否有嚴重損及公共利益之發現）或其他有關價值等，作一綜合評估，並請至本部網站線上製作（格式如附件二）。

(五)頁碼編寫：請對摘要及目錄部分用羅馬字 I 、II、 III……標在每頁下方中央；報告內容至附錄部分請以阿拉伯數字 1.2.3.……順序標在每頁下方中央。

(六)附表及附圖可列在文中或參考文獻之後，各表、圖請說明內容。

(七)可供推廣之研發成果資料表：

1.研究計畫所產生之研發成果，應至科技部科技研發成果資訊系統（STRIKE 系統，http://ap0569.most.gov.tw/strike/homepageIndex.do）填列研發成果資料表（如附件三），循執行機構行政程序，由研發成果推廣單位（如技轉中心）線上繳交送出。

2.每項研發成果填寫一份。

(八)若該計畫已有論文發表者(須於論文致謝部分註明補助計畫編號)，得作為成果報告內容或附錄，並請註明發表刊物名稱、卷期及出版日期。若有與執行本計畫相關之著作、專利、技術報告、或學生畢業論文等，請在參考文獻內註明之。

(九)該計畫若列屬國際合作研究，應將雙方互訪及合作研究情況、共同研究成果及是否持續雙方合作等，於報告中重點式敘明。

三、計畫中獲補助國外差旅費，出國進行國際合作與移地研究、出席國際學術會議者，每次均須依規定分別撰寫出國心得報告（其中，出席國際學術會議者須另附發表之論文全文或摘要，但受邀專題演講或擔任會議主持人者不在此限），並至本部網站線上繳交電子檔，出國心得報告格式如附件四、五。

四、計畫中獲補助國外學者來臺費用，每次均須分別撰寫國外學者來臺訪問成果報告，並至本部網站線上繳交電子檔，報告格式如附件六。

五、報告編排注意事項

(一)版面設定：A4 紙，即長 29.7 公分，寬 21 公分。

(二)格式：中文打字規格為每行繕打（行間不另留間距），英文打字規格為 Single Space。

(三)字體：以中英文撰寫均可。英文使用 Times New Roman Font，中文使用標楷體，字體大小以 12 號為主。

附件一

科技部補助專題研究計畫成果報告

（□期中進度報告/■期末報告）

分解合成系列快速計算法之平行化改進

計畫類別：■個別型計畫　□整合型計畫
計畫編號：MOST 102－2115－M－004－003－
執行期間：　102 年 8 月 1 日至 103 年 7 月 31 日

執行機構及系所：應用數學學系

計畫主持人：曾正男
共同主持人：
計畫參與人員：林昱

本計畫除繳交成果報告外，另含下列出國報告，共 1 份：
□執行國際合作與移地研究心得報告
■出席國際學術會議心得報告

期末報告處理方式：

1. 公開方式：
   □非列管計畫亦不具下列情形，立即公開查詢
   □涉及專利或其他智慧財產權，□一年□二年後可公開查詢
2. 「本研究」是否已有嚴重損及公共利益之發現：□否 □是
3. 「本報告」是否建議提供政府單位施政參考 □否 □是，____（請列舉提供之單位；本部不經審議，依勾選逕予轉送）

中　華　民　國　年　月　日

附件二

科技部補助專題研究計畫成果報告自評表

請就研究內容與原計畫相符程度、達成預期目標情況、研究成果之學術或應用價值（簡要敘述成果所代表之意義、價值、影響或進一步發展之可能性）、是否適合在學術期刊發表或申請專利、主要發現（簡要敘述成果是否有嚴重損及公共利益之發現）或其他有關價值等，作一綜合評估。

1. 請就研究內容與原計畫相符程度、達成預期目標情況作一綜合評估

■ 達成目標

□ 未達成目標（請說明，以 100 字為限）

□ 實驗失敗

□ 因故實驗中斷

□ 其他原因

說明：本計畫是將 split and combine 系列的演算法推廣到平行計算，我們利用 Python 程式的架構成功的將此系列的加速法推廣到多核心計算以及 GPU 的計算。對於 split and combine 系列的計算法如何分群是一個重要的關鍵，我們經過這次的計畫補助理解了如何有效率的分群，讓分解合成計算更有效率。

2. 研究成果在學術期刊發表或申請專利等情形：

論文：□已發表 □未發表之文稿 ■撰寫中 □無

專利：□已獲得 □申請中 □無

技轉：□已技轉 □洽談中 □無

其他：（以 100 字為限）

3. 請依學術成就、技術創新、社會影響等方面，評估研究成果之學術或應用價值（簡要敘述成果所代表之意義、價值、影響或進一步發展之可能性），如已有嚴重損及公共利益之發現，請簡述可能損及之相關程度（以 500 字為限）

近年來熱門的大資料計算為了因應龐大的資料我們需要更有效率的計算方式對資料進行分析，多數的網頁語意資料會成為矩陣化的資料，並且在預處理的資料整理過程中資料的量需要變大才能成為矩陣的形式，本研究所提供的快速計算法搭配平行化的計算更能符合現代大資料計算的需求。

以網頁資料為例，流量龐大的網頁本來就具有分散儲存與分散登入的特性，在結構上很自然就有 Split 分群的架構。龐大的資料在資料庫中若需要在集中寫入到同一個地方儲存再進行運算，不只有效率不彰的問題，在巨量資料的環境下也不可行。因此，在各自的資料庫裡計算，再將計算的結果丟出集中整合，是一個可行的結果，這樣的計算架構剛好也是 Split-and-combine 的計算架構。我們的分散式計算可以隨著計算核心數目的增加來增加計算的速度，計算所需的速度隨著計算單元硬體的增加而遞減。

過去我們在資料分群的過程是採隨機分群的方式，在正確資料維度估計的前提下透過隨機分群，群與群之間重疊的維度有很高的機率讓重疊的維度高於估計的維度，這部份經過我們的改善新的演算方式可以保證讓資料分群時重疊的維度都高於估計的維度。這部份的改進確保了計算的精準度，不需要隨機分群的方式讓計算更有效率。

附件三

科技部補助計畫衍生研發成果推廣資料表

日期：___年__月__日

科技部補助計畫	計畫名稱： 計畫主持人： 計畫編號：　　領域：		
研發成果名稱	（中文）		
	（英文）		
成果歸屬機構		**發明人 (創作人)**	

技術說明	（中文） （200-500 字）
	（英文）
產業別	
技術/產品應用範圍	
技術移轉可行性及預期效益	

註：本項研發成果若尚未申請專利，請勿揭露可申請專利之主要內容。

附件四

科技部補助專題研究計畫執行國際合作與移地研究心得報告

日期：___年___月___日

計畫編號	MOST　　－　　－　　－　　－　　－		
計畫名稱			
出國人員 姓名		服務機構 及職稱	
出國時間	年　月　日至 年　月　日	出國地點	
出國研究 目的	□實驗 □田野調查 □採集樣本 □國際合作研究 □使用國外研究設施		

一、執行國際合作與移地研究過程

二、研究成果

三、建議

四、本次出國若屬國際合作研究，雙方合作性質係屬：(可複選)

☐分工收集研究資料
☐交換分析實驗或調查結果
☐共同執行理論建立模式並驗証
☐共同執行歸納與比較分析
☐元件或產品分工研發
☐其他 (請填寫) _____

五、其他

附件五

科技部補助專題研究計畫出席國際學術會議心得報告

日期：___年___月___日

計畫編號	MOST　　102－2115－M－004－003－		
計畫名稱	分解合成系列快速計算之平行化改進		
出國人員姓名	曾正男	服務機構及職稱	政治大學應用數學系
會議時間	2014 年 6 月 23 至 2014 年 6 月 25 日	會議地點	泰國
會議名稱	(中文)2014 年東亞地區工業與應用數學國際會議 (英文)EASIAM 2014		
發表題目	(中文) 解自然三階樣條函數的平行化方法 (英文) Parallel method for solving the natural cubic spline		

一、 參加會議經過

這次的東亞區工業與應用數學年會舉辦的時間正值泰國政變時節，因此與會者皆觀望大會是否會繼續舉辦會議，經過延宕後來將會議的地點更改在離曼谷較遠的 Pattaya，並且在國賓飯店（Ambassador City Jomtien）舉行，會議期間為 2014 年 6 月 23 日至 6 月 25 日。

大會在 23 日早上 9:00 開始，由賴明治老師幽默開場，並請到 Roger Grimshaw 演講，與會人數約一百二十人。三天的時間除了主題演講之外，還有六場平行演講。參加本次會議，除了 24 日下午的報告之外，也臨時被告之要擔任該場次的 Chair。

第二天晚上有大會有舉辦自由參加的文化之夜參訪活動，並且在 25 日中午圓滿結束會議。

二、 與會心得

這次的會議很短暫，特別是在政變期間舉辦，由於大會有寫信提醒告知我們泰國最新的政治進展，也告知我們這樣的政變結果反而會讓治安變得更穩定，所以才在最後的期限內決定繼續參加此次的會議。雖然如此，對於到一個正在政變中的國家進行學術交流，在正式到達泰國之前心理還是很忐忑。落地後在泰國機場雖然多了許多荷槍實彈的軍人，但是從他們與民眾的輕鬆互動看來，情勢並沒有像台灣新聞媒體報導的這麼緊張，機場的人潮還

是非常的熱絡。

由於到 Pattaya 需要再搭了個小時左右的公車，我們需要到機場下方的區域搭乘公車，從機場到 Pattaya 的車票是在台灣透過網路付款的，一路上拿著印出的夠票資訊，沿途的服務人員就會自動的引導我們行進的方向，這一點深刻感受到泰國對於國際化的友善，即使是一般的商家，也會自動地協助引導旅客。特別的是，在我們走往車站的路上還有電信業者免費發放手機的 Sim 卡提供我們在泰國期間使用，我們用大約台幣 200 元的價格購買了一週的 1G 的無線網路服務，相信這一點對於來到泰國的觀光客會感到泰國在通訊網路上的友善。

飯店和會場中的環境都有無線網路設施，讓人覺得泰國在網路建設上的進步。會議進行中也有非大會的人員到場對我們進行問卷調查，只要是提供他們在觀光上的評價與建議。感覺上泰國也與新加坡一樣，越來越注重來訪國外旅客的意見。

大會因為是在海邊的飯店舉行，因此也吸引了一些西方的應用數學學者來參加。這次泰國當地大學的博士生所發表的內容，讓人覺得泰國的大學在應用數學上的進步也與他們的硬體建設一樣是非常跳躍的。不只是演講的內容深度，連表達的方式都很有國際水準。我們需要更加強學生的研究與國際化的能力，這是我此行最大的心得。

三、發表論文全文或摘要

In computing the natural cubic spline, we have to solve a tri-diagonal matrix linear system. If we have the slop of each interpolation points, we can compute the Hermite spline instead of the natural cubic spline. Computing the Hermite spline is much easier than computing the natural cubic spline. We can solve the piece-wise cubic polynomial in each interval independently. Therefore, the Hermite spline can be easily implemented to the parallel computing. We will demonstrate an iterative method for computing the natural cubic spline by the implementation of the approach to compute the Hermite spline. The performance of this proposed method is better than the Gauss- Seidel method. It is faster and can be easily implemented to the parallel computing.

三、 建議

我們需要多舉辦國際化的學術活動，讓學生更習慣與世界接軌的潮流。

四、 攜回資料名稱及內容

論文集。

六、其他

附件六

科技部補助專題研究計畫國外學者來臺訪問成果報告

日期：___年___月___日

計畫編號	MOST　　－　　－　　－　　－　　－		
計畫名稱			
邀訪學者 姓名		服務機構 及職稱	
國籍		來臺時間	年　月　日至　年　月　日
來訪目的 (可複選)	☐技術指導 ☐實驗設備設立 ☐計畫諮詢/顧問 ☐學術演講 ☐國際會議主講員 ☐其他		

一、訪問過程

二、對本項專題計畫產生之影響、貢獻或主要成果

三、建議

四、其他

科技部補助計畫衍生研發成果推廣資料表

日期:2014/10/14

科技部補助計畫	計畫名稱: 分解合成系列快速計算法之平行化改進
	計畫主持人: 曾正男
	計畫編號: 102-2115-M-004-003- 學門領域: 矩陣計算及應用

無研發成果推廣資料

102 年度專題研究計畫研究成果彙整表

計畫主持人：曾正男	計畫編號：102-2115-M-004-003-
計畫名稱：分解合成系列快速計算法之平行化改進	

成果項目			量化			單位	備註（質化說明：如數個計畫共同成果、成果列為該期刊之封面故事...等）
			實際已達成數（被接受或已發表）	預期總達成數(含實際已達成數)	本計畫實際貢獻百分比		
國內	論文著作	期刊論文	0	0	100%	篇	
		研究報告/技術報告	1	1	100%		
		研討會論文	0	0	100%		
		專書	0	0	100%		
	專利	申請中件數	0	0	100%	件	
		已獲得件數	0	0	100%		
	技術移轉	件數	0	0	100%	件	
		權利金	0	0	100%	千元	
	參與計畫人力（本國籍）	碩士生	1	0	100%	人次	
		博士生	0	0	100%		
		博士後研究員	0	0	100%		
		專任助理	0	0	100%		
國外	論文著作	期刊論文	0	1	100%	篇	
		研究報告/技術報告	0	0	100%		
		研討會論文	1	1	100%		
		專書	0	0	100%	章/本	
	專利	申請中件數	0	0	100%	件	
		已獲得件數	0	0	100%		
	技術移轉	件數	0	0	100%	件	
		權利金	0	0	100%	千元	
	參與計畫人力（外國籍）	碩士生	0	0	100%	人次	
		博士生	0	0	100%		
		博士後研究員	0	0	100%		
		專任助理	0	0	100%		

其他成果 (無法以量化表達之成果如辦理學術活動、獲得獎項、重要國際合作、研究成果國際影響力及其他協助產業技術發展之具體效益事項等，請以文字敘述填列。)	邀請何炳生教授來台短期研究與講學。

	成果項目	量化	名稱或內容性質簡述
科教處計畫加填項目	測驗工具(含質性與量性)	0	
	課程/模組	0	
	電腦及網路系統或工具	0	
	教材	0	
	舉辦之活動/競賽	0	
	研討會/工作坊	0	
	電子報、網站	0	
	計畫成果推廣之參與（閱聽）人數	0	

科技部補助專題研究計畫成果報告自評表

請就研究內容與原計畫相符程度、達成預期目標情況、研究成果之學術或應用價值（簡要敘述成果所代表之意義、價值、影響或進一步發展之可能性）、是否適合在學術期刊發表或申請專利、主要發現或其他有關價值等，作一綜合評估。

1. 請就研究內容與原計畫相符程度、達成預期目標情況作一綜合評估

■達成目標

□未達成目標（請說明，以 100 字為限）

□實驗失敗

□因故實驗中斷

□其他原因

說明：

2. 研究成果在學術期刊發表或申請專利等情形：

論文：□已發表 □未發表之文稿 ■撰寫中 □無

專利：□已獲得 □申請中 ■無

技轉：□已技轉 □洽談中 ■無

其他：（以 100 字為限）

3. 請依學術成就、技術創新、社會影響等方面，評估研究成果之學術或應用價值（簡要敘述成果所代表之意義、價值、影響或進一步發展之可能性）（以 500 字為限）

此平行化版本的 SCMDS 很適合用在大資料分析時將資料從高維度下降到低維度使用，我們使用 Python 作為軟體基礎，更方便主流分析大資料數據時所使用的程式語言。