應用於平行計算的排序演算法
A novel Sorting Algorithm applied in the Parallel Computing

藍國桐
德明財經科技大學
資訊科技學系 副教授
ktlan@takming.edu.tw

王李吉
東南科技大學
資訊科技與通訊學系 講 師
richwang@mail.tnu.edu.tw

摘 要

本論文提出一個可適用於平行計算環境的排序演算法。近年來，利用圖形顯示卡多顆圖形加速處器(GPU)來改善傳統單一中央處理器(CPU)的計算效能的研究日益興盛，其基本的精神就是採用多顆處理器的平行計算機制(parallel computing)。平行計算可以大幅提昇單一計算核心的計算能力，藉由圖形顯示卡的平行計算功能，傳統上的冗長模擬計算過程都在瞬間被計算出來，也造就了目前精緻的遊戲/電影動畫工業。平行計算雖然可應用於加速計算的目標，但是由於它是一個分散式的計算行為，對於有些需要中央統籌計算與整體資料分析的功能卻不易達成。本研究就是針對平行計算不易完成的單一排序問題的缺點，提出一個新的排序演算法，並在 nVidia GeForce 8600 的圖形顯像卡為 GPU 計算環境之下，實現平行計算的中央排序能力。

關鍵詞：排序演算法、GPU、CPU、平行計算、nVidia GeForce 8600

一、前 言

近年來，利用圖形顯示卡多顆圖形加速處器(GPU)來改善傳統單一中央處理器(CPU)的計算效能的研究日益興盛，其基本的精神就是採用多顆處理器的平行計算機制(parallel computing)。平行計算實際上已發展多年，而近年來因為平行電腦叢集(PC Cluster) 成本低，效能漸漸超過傳統的超級電腦，於是慢慢受到重視。而圖形顯示卡多顆圖形加速處器(GPU)架構的公開給大眾使用，更是加速平行計算的普及性。其中以 nVidia 公司的 CUDA 平行計架構[1, 2] 最受到重視，已從單純的圖形加速繪圖演進到一般的平行計算功能了。平行計算可以大幅提昇單一計算核心的計算能力，藉由圖形顯示卡的平行計算功能，傳統上的冗長模擬計算過程都在瞬間被計算出來，也造就了目前精緻的遊戲/電影動畫工業。平行計算雖然可應用於加速計算的目標，但是由於它是一個分散式的計算行為，對於有些需要中央統籌計算與整體資料分析的功能卻不易達成。本研究基於演化計算研究[3]上的天擇演化時常常需要用到全體排序的需求，但是在將演化計算實現於平行計算環境時無法作全體排序的動作，因而經常必須修改傳統的天擇功能以適用於平行計算[4, 5]。基於上述原因本論文提出一個以計數排序法為基礎的併行計數排序法，企盼能解決排序演算法無法應用於平行計算環境上的缺點。

二、併行計數排序演算法

針對 GPU 是一個以 block/thread 架構的平行計算系統，同一個 block 中的 threads 可以同時進行 simple instruction multiple data (SIMD)的計算，當資料量小於一個 block 中的 threads 數目時，尚可以進行集體的排序動作。當資料量大於這個 threads 數目時，對於排序中的資料和尚未排入 threads 的資料之間會有資料更新不同步的現象，要解決這個問題必須用到 preemptive 的技巧，不幸的是這張 nVidia GeForce 8600 的圖形顯像卡並無法支援此類的高階平行計算功能。

另外一個問題是，平行計算中每個處理器只針對自己的資料作動作，它並無法作一個集中式的資料分析處理，對於排序的動作不易完成。

我們的演算法是採用 CPU-GPU 同步併行的方式來進行的，它的構想就是先利用 GPU 的多個處理器將待排序資料作分區段的排序。這部分為了能適用於 GPU 的平行計算環境，我們採用 counting sort 的方式，採用 counting sort 的優點是無需去更動原始資料，以避免資料更新不同步的現象。CPU 的部分則是採用 merge sort，將 GPU 已排好的某段資料傳回主機端，利用 merge sort 將它排入原有已排好的資料段。這兩部分的排序動作可以併行處理，也達到同時計算的目的。這個併行計數排序演算法的過程如下：

Step 1.

先將待排序的資料切成 N 段(如圖一)，每段的資料量符合 CUDA 的 threads 數目(threads/block 設定為 256)。

圖一，配合 GPU 環境，將資料切成 N 段

Step 2.

將 block #1 送入 GPU 作 counting sort。

Step 3.

For i=2 to N-1 {

GPU:

將 block #i 送入 GPU 作 counting sort。

CPU:

將 block #1 ~ block #i-1 作 merge sort。

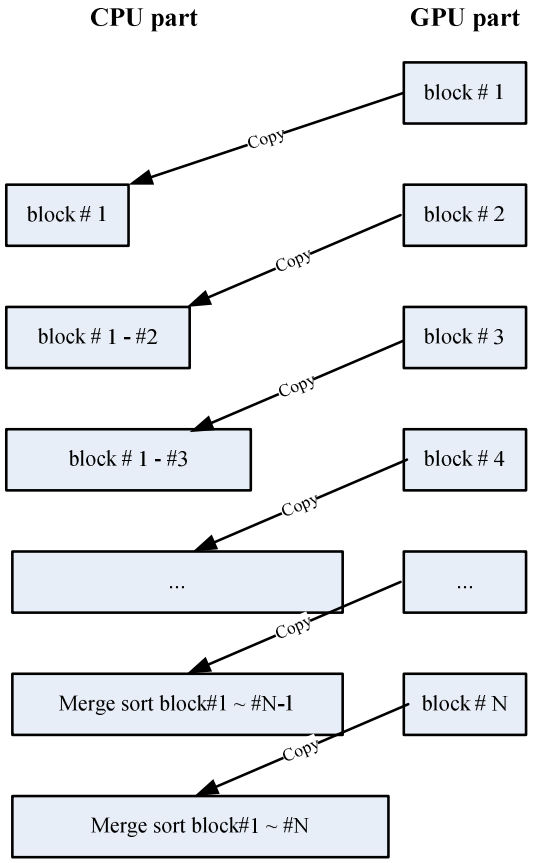
(注意：GPU/CPU 同步進行計算。)

}

Step 4.

將 block #1 ~ block #i 作 merge sort。

將 CPU-GPU 同步併行的併行計數排序演算法示成圖二。



圖二，CPU-GPU 同步併行的併行計數排序演算法

三、實驗結果與分析

本研究的目的在於提出一個適用於平行計算環境(以 CUDA 的 GPU 架構為例)的排序演算法。除了實現第二節所描述的併行計數排序演算法外，為了比較在單一中央處理器環境(CPU)與平行計算環境(GPU)下，兩種演算法的效率，我們比較了 CPU Heap Sort 和 GPU Counting Sort 的排序時間。

以下是為本研究的模擬實驗所採用的一些參數：

1. 所使用的排序演算法

CPU 環境：採用傳統的 Heap Sort 作為排序演算法。

GPU 環境：採用我們所提出的傳統的併行計數排序演算法。

2. 實驗的硬體環境

CPU 環境：採用 Intel Core 2 Duo, 2.26GHz。

GPU 環境：採用 nVidia GeForce 8600 的圖形顯像卡。

3. 排序資料筆數： 7,200 – 13,200,000 筆。

4. 資料來源：隨機亂數- Gaussian Distribution, mean: 0, deviation: 20。

5. 開發環境：MS-VS2008, C++/Cuda C。

表一、兩種排序架構的排序效能(未採併行計算架構)

排序方式 / 資料筆數	CPU HeapSorting	GPU Counting Sort
7,200	0 ms	0 ms
72,000	47 ms	31 ms
720,000	563 ms	328 ms
7,200,000	7,016 ms	3,781 ms
8,200,000	8,062 ms	4,328 ms
9,200,000	9,125 ms	4,907 ms
10,200,000	10,203 ms	5,469 ms
11,200,000	11,297 ms	6,000 ms
12,200,000	12,390 ms	(跑不動)
13,200,000	13,468 ms	(跑不動)

實驗一是先作一個 CPU 的 Heap Sort 排序，資料量由 7,200 ~ 13,200,000，而 GPU 的部分則完全採用 Counting Sort 的方式。

由表一的實驗數據顯示：併行計數排序演算法明顯可以提昇排序的速度。當資料筆數在 720,000 筆以下時，使用單一中央處理器- CPU 或使用平行處理器- GPU 的方式，其速度上的差別不大，因為現在的 CPU 效能都相當高，因此其計算能力不輸給平行環境。但是當待處理的資料筆數太多時，如表一，7,200,000 – 11,200,000 時，CPU 的處理時間明顯要超過 GPU 使用的時間。這個現象很明確，因為平行計算的效率要比單一處理器要大很多。

表一中最後兩個實驗結果，GPU 無法執行更多資料的原因是因為本實驗室所使用的 nVidia GeForce 8600 的圖形顯像卡是一個相當低階的繪圖卡，其記憶體的容量有限，因此無法繼續執行本

程式。

表二、兩種排序架構的排序效能(採併行計算架構)

排序方式 / 資料筆數	CPU HeapSorting	GPU Counting Sort + CPU Merge Sort
11,200,000	11,297 ms	**6,016 ms**
12,200,000	12,390 ms	**6,563 ms**
13,200,000	13,468 ms	**7,078 ms**
14,200,000	14,532 ms	**7,532 ms**

表二是我們採用併行架構下的實驗結果。實驗數據顯示：當資料筆數太大時，使用分段排序及併行計算方式可以克服 nVidia GeForce 8600 的圖形顯像卡的記憶體限制的缺點，這項技術也可以應用在其他高階 GPU 環境。大量資料的排序雖然需要作頻繁的資料搬移動作，但是在併行處理的機制之下，面對百萬筆資料的資料排序(11,200,000 ~ 14,200,000)，CPU-GPU 的併行計算所需的時間是單純使用 CPU 作排序的一半以上。

四、結論

本論文提出一個可適用於平行計算環境的排序演算法- 併行計數排序演算法。也就是搭配 nVidia 圖形顯像卡的 GPU 平行計算環境，由 GPU 作分段的 counting sort，同時由 CPU 將 GPU 排好的部分資料作 merge sort。由實驗資料可以看得出來，它的效率高於單純使用 CPU 的排序演算法。這也是平行計算受到重視的因素。

本研究只是針對平行計算環境不易作到整體排序的缺點，提出一個可行的排序演算法以便應用於分散式演化計算。在這初步的結論中，我們沒有和傳統各式各樣的排序演算法作計算速度的比較，尤其是知名的 Quick Sort，這也將是未來的研究方向之一。

參考文獻

[1] CUDA Zone- http://www.nvidia.com/object/cuda_home_new.html

[2] J. Sanders, E. Kandrot, *CUDA By Example- An Introduction to General-Purpose GPU Programming*, Addison-Wesley, 2011.

[3] T. Back, U. Hammel, and H. P. Schewfel, "Evolutionary computation: comments on the history and current state," IEEE Trans. Evol. Comput., vol.1, no.1, pp.3-17, 1997.

[4] R. Arora, R. Tulshyan, and K, Deb, "Parallelization of Binary and Real-Coded Genetic Algorithms on CUDA," 2010 IEEE CEC, Barcelona, Spain.

[5] S. Harding and W. Banzhaf, "Distributed genetic programming on GPUs using CUDA," Proceedings of the Second International Workshop on Parallel Architectures and Bioinspired Algorithms (WPABA 2009), pages 1--10, Raleigh, NC, USA.