Journal of Korea Spatial Information Society Vol.20, No.6 : 139-144, December 2012
http://dx.doi.org/10.12672/ksis.2012.20.6.139
ISSN 2287-9242(Print)
ISSN 2287-9250(Online)

GPU를 활용한 R*-tree에서의 부분 노드 병렬 처리 방법

A Parallel Processing Method for Partial Nodes in R*-tree Using GPU

김 성* 오 병 우**
Seong Kim Byoung-Woo Oh

요 약 공간 데이터 처리는 GIS, 텔레매틱스 등 광범위한 분야에서 널리 사용되고 있다. 그러나 현재 사용되고 있는 공간 데이터 질의 처리 기법은 CPU를 사용하여 순차적으로 질의 처리를 수행하므로 질의 처리 시간이 상대적으로 오래 걸린다는 단점이 존재한다. 그러나 공간 데이터 질의 처리를 병렬로 수행했을 때 처리 시간을 줄이는 것이 가능하다. 따라서 본 연구에서는 GPU를 활용하여 공간 데이터 질의 처리를 병렬로 수행하는 연구를 진행한다. 또한, CPU를 이용하여 질의 처리를 수행한 결과와 비교하여 속도 향상 정도에 대한 결과를 제시한다.

키워드 : R*-tree, GPU, 인덱스, 공간 데이터, 병렬 처리

Abstract The R*-tree manages hierarchical nodes for efficient access of spatial data. We propose a method that maintains partial nodes of R*-tree in the GPU memory to improve efficiency using parallel processing. The proposed method attempts to load as many nodes as possible to the GPU memory. The new nodes are inserted to manage the rest of R*-tree nodes in the main memory. The experimental result shows that the proposed method is more efficient than the main memory based R*-tree.

Keywords : R*-tree, GPU, Index, Spatial Data, Parallel Processing

1. 서 론

최근 GPU의 성능 및 관련 산업이 발전하면서 이를 범용 목적에 사용하려는 시도가 곳곳에서 이루어지고 있다. GPU는 Figure 1.(b)와 같이 많은 수의 ALU를 보유하고 있어 병렬 처리에 매우 효과적인 구조이므로 이를 이용한 영상처리, 대용량 데이터 처리 등의 분야에서 사용하고 있는 추세이다. 이에 따라 공간 데이터 처리 분야에서도 GPU를 활용하는 기법의 연구가 다수 진행되고 있다[8].

본 논문에서는 이러한 GPU의 병렬 처리 능력을 바탕으로 트리 기반의 공간 데이터에 대한 범위 질의를 병렬로 처리하는 기법을 제시한다. 또한 대용량의 공간 데이터의 입력에 대해서도 병렬 처리가

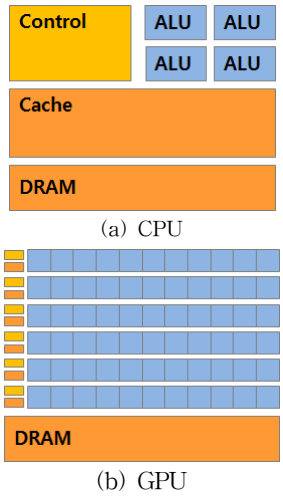

Figure 1. Internal Structure of CPU and GPU

† This paper was supported by Sabbatical Year Research Fund, Kumoh National Institute of Technology.

* Seong Kim, Master's Student, Dept. of Computer Engneering, Kumoh National Institute of Technology, sseong2ya@kumoh.ac.kr
** Byoung-Woo Oh, Professor, Dept. of Computer Engneering, Kumoh National Institute of Technology, bwoh@kumoh.ac.kr (Corresponding Author)

가능하도록 확장성을 고려한 알고리즘을 제시한다.

본 논문의 구성은 다음과 같다. 2장에서 GPGPU와 공간 데이터 처리 기술에 대하여 간략하게 설명하고 3장에서는 본 논문에서 제시한 기법을 설명한다. 4장에서는 본 논문에서 제시한 기법과 기존의 기법을 비교하기 위한 실험결과를 제시하고 마지막 5장에서 결론과 향후 과제를 제시한다.

2. 관련연구

2.1 GPGPU

GPGPU(General-Purpose computing on Graphic Processing Unit) 기술은 GPU가 보유한 자원을 사용하여 CPU가 전통적으로 취급했던 응용 프로그램들의 계산을 보다 빠르게 처리하기 위하여 제안된 기술이다[7]. GPGPU 기술에 대한 지원을 위하여 GPU 업계를 이끄는 두 업체인 Nvidia사와 AMD사에서 Toolkit을 발표하였는데 그것이 각각 CUDA와 ATI Stream이다[1,12].

그 중에서 CUDA는 많은 수의 블록과 쓰레드를 사용하여 응용 프로그램의 계산을 병렬로 처리한다. 이 때 병렬로 처리하기 위한 커널 함수를 호출하게 되는데 이 커널 함수는 블록과 쓰레드의 개수만큼 호출되어 계산을 수행한다.

이 논문에서는 공간 데이터 질의를 처리하기 위해 CUDA Toolkit을 사용하여 병렬로 질의 처리를 수행하였다.

2.2 공간 데이터 처리

공간 데이터를 효율적으로 처리하기 위하여 R-tree나 R*-tree 등의 트리구조를 사용한다. 그 중 R-tree는 공간 데이터를 표현하는 대표적인 트리구조로서 객체에 대한 최소 경계 사각형(MBR)을 저장한다[2]. R*-tree는 R-tree의 탐색 부분의 성능 저하라는 단점을 보완한 트리구조로 재삽입을 통하여 검색성능을 향상시킨다[11].

현재 공간 데이터에 대한 질의를 GPU를 사용하여 병렬로 처리하는 방법에 관한 연구가 활발히 진행되고 있다.

[6]은 R-tree의 질의를 병렬로 처리하기 위하여 Leaf 노드들을 GPU 메모리에 복사하여 병렬 질의 처리를 수행하는 방법을 제안하였다. 이 논문에서 제시하는 방법은 GPU 메모리의 크기보다 큰 크기의 데이터 입력에 대하여 반응하지 못하는 단점이 존재한다.

[9]와 [10]은 R-tree에서 질의를 병렬로 처리하기 위하여 노드구조의 변경과 질의 결과 리스트를 이용한 질의 결과 확인 방법을 사용하였다. 그러나 이 방법은 공간 데이터에 관하여 scalability를 보장하지 못하는 단점이 존재한다.

[3]과 [4]는 GPU를 버퍼로 사용하는 알고리즘을 제안하였다. GPU에 일정량의 데이터를 복사한 후 복사된 데이터들에 관한 R-tree를 생성한다. 이를 Q-tree라고 지칭하는데 질의 처리를 수행하며 Q-tree에 검색하려는 노드가 존재하면 GPU 메모리에 대하여 병렬로 검색을 수행하고 존재하지 않으면 CPU를 사용하여 기존의 R-tree에 대한 검색을 수행한다.

3. 제안 기법

본 논문에서는 CPU를 활용한 공간 데이터 질의 처리 속도를 개선하기 위해 트리 기반의 공간 데이터에 대하여 GPU를 활용하여 병렬 질의 처리를 수행한다. 또한, GPU 메모리 크기를 초과하는 공간 데이터에 대한 처리를 위해 CPU와 GPU를 함께 사용하여 질의 처리를 수행한다. 병렬 질의 처리를 수행하기 위해서는 CPU와 GPU의 두 메모리상에 데이터의 할당 및 로딩, 그리고 병렬 질의 처리 과정을 거친다.

3.1 메모리 구조

Figure 2는 본 논문에서 제안한 기법의 메모리 구조를 설명한 그림이다. GPU 메모리에는 MBR 테이블과 결과 배열, CPU 메모리에는 공간 데이터와 결과 배열을 포함한다.

CPU 메모리의 공간 데이터는 2차원 공간상의 객체에 대한 좌표 등의 정보를 가지고 있다. 이 데이터는 디스크에 저장된 데이터를 메모리에 로딩하여 만들어지는 데이터로써 화면에 표시할 때에도 사용된다. GPU 메모리의 MBR 테이블은 객체의 MBR들을 저장한다. MBR은 공간 객체의 최소 경계를 이루는 사각형을 말하며 왼쪽 상단과 오른쪽 하단의 좌표로 이루어져 있기 때문에 16Bytes로 구성된다. 마지막으로 결과 배열은 각 MBR의 질의 처리 수행 결과를 저장하는 배열로써 수행 결과에 따라

1 혹은 0의 값을 가지는 배열이다.

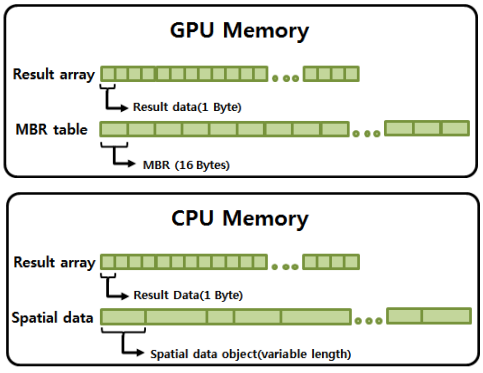

Figure 2. Memory Structure

3.2 공간 데이터 로드

Figure 3은 공간 데이터를 로드하여 GPU 메모리에 분배하는 방법을 설명한 그림이다. 시스템은 디스크로부터 공간 데이터를 읽어와 메인 메모리에 트리 형태로 저장된다. 트리 형태로 저장된 공간 데이터는 GPU에서 병렬 질의 처리를 수행하기 위하여 단말노드들을 GPU 메모리에 복사한다. 이 때 GPU 메모리 공간을 검사하여 여유 공간 크기만큼의 단말 노드의 데이터를 복사한다. 이는 CUDA의 메모리 할당 함수인 CudaMalloc()과 복사 함수인 CudaMemcpy()를 통하여 수행한다.

GPU에 저장된 데이터 외에는 CPU에서 기존 검색 방법에 최소한의 수정을 가해 사용할 수 있도록 트리 레벨 별로 GPU에 저장된 노드들의 인덱스를 저장한다. 그리고 질의 처리를 수행할 때 각 레벨별로 인덱스 값 이상의 노드들에 대해서만 검색을 수행하여 결과를 도출할 수 있도록 한다.

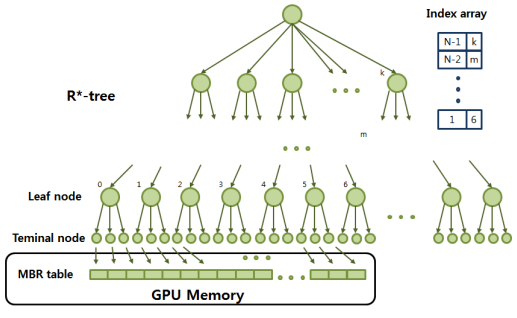

Figure 3. Spatial Data Loading Method

3.3 공간 데이터 질의 처리

공간 데이터를 로드한 후 사용자의 질의 요청이 생기면 질의 처리를 시작한다. 공간 데이터 질의 처리는 CPU와 GPU 두 프로세서에서 각각 진행되는데 GPU는 공간 데이터 질의 처리를 병렬로 수행하고 CPU는 저장된 인덱스 값 이상의 노드들에 대하여 기존의 질의 처리를 수행한다.

Figure 4는 커널 함수의 의사코드를 나타낸 그림이다. GPU에서 공간 데이터 질의 처리를 수행하기 위해서는 커널 함수를 실행한다. 커널 함수는 CUDA의 블록과 쓰레드에서 동시에 질의 처리를 수행하여 결과를 결과 배열에 저장한다.

각 쓰레드들의 커널 함수가 모두 종료 되면 결과를 CPU 메모리로 복사하기 위해 다시 한 번 CudaMemcpy() 함수를 수행한다. CPU에서는 이 결과를 바탕으로 사용자가 질의 결과를 확인할 수 있도록 화면에 질의 결과를 출력한다.

```
__global__ void KernelSearchFunction(MBR of Query Region)
{
        i = blockIdx.x * Count of Leaf Nodes + threadIdx.x
        if(i < Total Count of Objects & in MBR Query Region) {
                result[i] = 1;
        else
                result[i] = 0;
}
```

Figure 4. Kernel Function

4. 실험 결과

실험 환경은 다음과 같다. 운영체제는 Windows 7이고, CPU는 인텔 코어2 6600 2.4GHz, RAM은 2GB이다. GPU는 8개의 코어와 256MB의 메모리를 가진 NVidia사의 Geforce 8600GTS이고, CUDA 4.2 버전을 사용하였다.

테스트는 Tiger/Line 데이터 셋 중에서 Figure 5에서 보이는 캘리포니아 지역의 전체 데이터를 사용하였다. 성능 평가를 위한 질의 범위는 데이터 셋의 한 변에 대한 질의 영역의 한 변의 길이의 비를 사용하였다. 길이의 비율은 5%부터 50%까지 10단계로 나누었고 신뢰성을 얻기 위하여 단계 당 1000번을 수행한 결과의 평균을 제시하였다.

Figure 5. Spatial Data Set

Table 1은 제안한 기법의 질의 범위 크기에 따른 질의 수행 시간을 측정한 결과이다. Search 항목은 실제 질의를 CPU와 GPU에서 각각 검색하는 시간을 의미하고, Transfer 항목은 GPU에서 검색한 결과를 CPU의 메모리에 적재하는 시간을 의미한다. 측정한 결과를 살펴보면 CPU에서 질의를 처리하는 시간은 질의 범위가 커질수록 급격하게 증가하는 반면에, GPU에서 검색을 수행하는 시간은 CPU에서 일부 데이터에 대하여 질의를 처리하는 수행 시간 때문에 조금씩 증가하나 기존 기법에 비해 증가량이 적은 것을 확인할 수 있다.

Table 1. Execution Time

Query Region	Search		Transfer	Sum
	CPU	GPU		
5%	1.786	24.270	5.076	31.131
10%	5.504	24.901	5.093	35.499
15%	10.099	25.602	5.162	40.863
20%	18.739	26.414	5.257	50.410
25%	30.093	27.258	5.343	62.694
30%	41.739	28.139	5.340	75.218
35%	56.393	28.985	5.371	90.749
40%	71.867	29.761	5.406	107.033
45%	91.018	30.529	5.445	126.992
50%	110.583	31.585	5.455	147.623

Table 2와 Figure 6은 제안한 기법과 기존 CPU에서 질의를 수행하는 시간을 비교한 결과이다. 제안한 기법에서 10%까지는 기존 기법보다 수행시간이 느리나 질의 범위 한 변 길이의 비가 15% 이상의 구간부터는 기존의 기법보다 질의 처리 속도가 CPU보다 빠른 것을 확인할 수 있다.

Table 2. Speedup

Query Region	CPU	GPU	Speedup
5%	8.101	31.131	0.26
10%	24.622	35.499	0.69
15%	46.244	40.863	1.13
20%	83.811	50.410	1.66
25%	132.189	62.694	2.11
30%	188.02	75.218	2.50
35%	257.702	90.749	2.84
40%	322.062	107.033	3.01
45%	403.594	126.992	3.18
50%	497.399	147.623	3.37

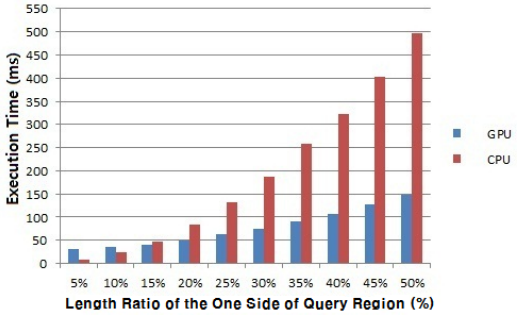

Figure 6. Execution Time Graph

5. 결 론

현대의 공간 데이터를 처리하는 시스템에서 대용량화 된 공간 데이터를 처리하기 위하여 GPU를 이용하는 기법들이 다수 연구되고 있다. 그러나 GPU의 프로세서나 메모리의 사양에 따라서 시스템 성능의 차이가 발생한다.

본 논문에서는 GPU 사양과 관계없이 대용량의 공간 데이터에 대하여 질의를 처리하기 위하여 GPU와 CPU를 함께 사용하는 방법을 제안하였다. 본 논문에서 제안하는 기법은 GPU 프로세서와 메모리의 처리범위를 뛰어 넘는 데이터의 입력에 대한 문제에 대하여 GPU 메모리가 허락하는 최대한의 공간 데이터를 GPU에서 처리하고, 나머지 데이터를 CPU에서 처리하는 방법을 적용하여 문제를 해결하였다.

본 논문에서 제안한 방법을 적용하여 질의 처리를 수행한 결과 질의 영역 한 변 길이의 비 15%이상의 질의 처리 수행에서 GPU와 CPU를 함께 사용

하는 기법이 CPU에서 질의를 처리하는 기존의 기법보다 최대 3.37배의 성능향상 정도를 확인할 수 있었다.

향후 과제로 본 논문에서 제안하는 기법은 GPU 메모리에 질의 범위에 포함된 영역이 얼마나 많이 포함되어 있느냐에 따라 성능의 차이가 발생한다. 따라서 알고리즘 성능의 개선을 위하여 GPU 메모리에 공간 데이터를 로딩하는 부분에 대한 연구가 필요하다. 또한, 테스트를 위해 주어진 현재의 데이터 셋보다 더 큰 용량의 데이터 셋을 적용하여 실험을 하는 추가 연구가 필요하다.

References

[1] AMD, 2011, "AMD Accelerated Parallel Processing OpenCL Programming Guide (Version 1.3f)".

[2] Antonin Guttman, 1984, "R-trees: a dynamic index structure for spatial searching", SIGMOD '84: Proceedings of the 1984 ACM SIGMOD international conference on Management of data, 14(2):47-57.

[3] BoSeon Yu, 2010, "Parallel Range Query processing on R-tree with Graphics Processing Units", pp. 1-32, Inha University.

[4] BoSeon Yu, Hyunduk Kim, Wonik Choi, Dongseop Kwon, 2011, "Parallel Range Query processing on R-tree with Graphics Processing Units ", Journal of Korea Multimedia Society, 14(5):669-680.

[5] Byoung-Woo Oh, 2008, "Efficient Spatial Index for Mobile Software", The Journal of GIS Association of Korea, 16(1):113-127.

[6] Byoung-Woo Oh, 2012, "A Parallel Access Method for Spatial Data Using GPU", International Journal on Computer Science and Engineering(IJCSE), 4(3):492-500.

[7] GPGPU, 2012, General-Purpose Computation on Graphics Hardware, Accessed June 7. http://www.gpgpu.org

[8] Jae-Il Lee, Byoung-Woo Oh, 2009, "An Efficient Technique for Processing of Spatial Data Using GPU", The Journal of GIS Association of Korea, 17(3):371-379.

[9] Mincheol Kim, Wonik Choi, 2011, "Acceleration of Range Query in R-tree Using GPU Parallel Processing", Journal of KIISE : Korea Computer Congress 2011, 38(1):37-40.

[10] Mincheol Kim, Wonik Choi, 2011, "GPU Sensitive R-tree for Efficient Parallel Processing of Range Queries", Journal of KIISE : Autumn Conference 2011, 38(2):61-64.

[11] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, Bernhard Seeger, 1990, "The R*-tree: an efficient and robust access method for points and rectangles", SIGMOD '90: Proceedings of the 1990 ACM SIGMOD international conference on Management of data, 19(2):322-331.

[12] NVIDIA, 2012, "NVIDIA CUDA™ C Programming Guide (Version 4.2)".

논문접수 : 2012.12.03
수 정 일 : 2012.12.20
심사완료 : 2012.12.21

Seong Kim

2011 Dept. of Computer Engineering, Kumoh National Institute of Technology Graduation (Bachelor of Engineering)

2011~present Dept. of Computer Engineering, Kumoh National Institute of Technology Graduate School Attending (Master's Course)

Research Expertise

- Spatial Database
- Mobile GIS
- Telematics

Byoung-Woo Oh

1993 Dept. of Computer Engineering, Konkuk University Graduation (Bachelor of Engineering)
1995 Dept. of Computer Engineering, Konkuk University Graduation (Master of Engineering)
1999 Dept. of Computer Engineering, Konkuk University Graduation (Ph.D. Computer Engineering)
1999~2004 Senior Researcher, Electronics and Telecommunications Research Institute
2004~present Professor, Dept. of Computer Engineering, Kumoh National Institute of Technology
Research Expertise
• Spatial Database
• Location Based Service
• Telematics
• Mobile S/W