스케줄링 알고리즘에 따른 협업 시스템의 성능 분석

진동규*, 조성우**, 조용연**, 김상욱§**, 오현옥***
*한양대학교 컴퓨터공학과
**한양대학교 컴퓨터 소프트웨어 학과
*** 한양대학교 정보시스템학과
e-mail : {proxiajd, nera, jyy0430, wook, hoh}@hanyang.ac.kr

Performance Analysis of Collaborative Processing by Scheduling Algorithm

Dong-Kyu Jin*, Sung-Woo Cho**, Yong-Yeon Jo**, Sang-Wook Kim**
*Dept. of Computer Science & Engineering, Han-Yang University
**Dept. of Computer and Software, Han-Yang University

요 약

대량의 정보를 효과적으로 처리하기 위한 기술로 CPU 뿐만 아니라 iSSD 와 GPGPU 를 개별적으로 이용하는 연구가 진행되고 있다. 본 논문에서는 더 나아가 CPU, iSSD 와 GPU 를 협업시켜 프로그램 수행 성능을 향상시키는 방법을 연구한다. 이러한 이질 시스템의 협업을 위해 이질 스케줄링 알고리즘을 적용하고, 스케줄링 알고리즘에 따른 협업 시스템의 성능을 분석한다.

1. 서론

SNS 의 보편화로 대량의 데이터가 실시간으로 생성되는 빅 데이터 시대가 오면서, 많은 양의 데이터를 효과적으로 처리하기 위한 연구들이 많이 진행되어 왔다 [1]. 이러한 빅데이터를 처리하기 위해 CPU 뿐만 아니라 SSD 에 in-storage processing 을 부여한 intelligent SSD(iSSD)까지 활용하여 데이터를 처리하고자 하는 시도들이 있었다 [2]. 이와 더불어 GPU 를 이용하여 데이터를 처리하는 GPGPU 연구가 있다 [3]. 따라서 본 논문은 대용량 데이터의 효과적인 처리를 위해 iSSD, GPU 와 CPU 로 구성된 협업 시스템을 구성하여 프로그램 수행 성능을 향상시키는 방법에 대해 연구한다.

CPU 는 소수의 고성능 core 로 이루어져 있기 때문에 복잡한 연산과 순차적인 연산에는 효율적이다. 반면 iSSD 는 저장되어 있는 데이터를 내부에서 처리하여 전송하기 때문에 CPU 로 전송하는 데이터를 줄일 수 있다 [2]. 또한 GPU 는 다수의 core 를 가지고 있어 CPU 로는 효과적으로 처리할 수 없는 병렬 처리에 적합하다 [3]. 따라서 CPU 와 iSSD, GPU 를 협업시켜 각 장치의 특성에 맞는 작업을 처리하게 하여 프로그램 수행 성능을 향상시킬 수 있다.

본 연구는 (1) 한양대-삼성전자 반도체 산학협력 연구과제와 (2) 미래창조과학부 및 정보통신산업 진흥원의 대학IT연구센터 지원사업 (NIPA-2013-H0301-13-4009)의 연구 결과로 수행되었음.

§교신저자

본 논문의 협업 시스템을 효율적으로 사용하기 위해서는 스케줄링이 필요하다. 스케줄링은 각 프로세서가 알고리즘 내의 함수들을 어떠한 시점에 처리할지 결정하는 과정이다. iSSD와 GPU, CPU는 각 프로세서들의 성능이 다르며, IPC가 발생하는 이질 시스템이다.

본 논문에서는 이러한 이질 스케줄링 알고리즘인 general dynamic level (GDL)과 best imaginary level (BIL) 을 적용하여 적합한 스케줄링을 찾고자 한다 [4,5]. 각 스케줄링 알고리즘에 따라 작업이 처리되는 프로세서의 종류와 처리순서가 다르게 결정되기 때문에 전체 성능에 차이가 발생한다. 따라서 본 논문은 스케줄링 알고리즘에 따른 협업 시스템의 수행 성능을 비교하였다. 실험결과, 리소스를 더욱 많이 사용할수록 좋은 성능을 보였으며, BIL 스케줄링을 적용한 결과가 CPU만 사용하는 in-host processing (IHP)에 비해 약 5배이상 좋은 성능을 보였다.

2. 배경지식.

2.1 iSSD

iSSD는 기존 SSD의 구조에서 각 채널에 범용 프로세서와 DRAM을 추가하여 내부 데이터 처리 능력을 부여한 SSD이다[1]. iSSD는 데이터를 처리할 때 채널 별로 나누어 병렬 처리한다[1]. iSSD의 코어 프로세서는 채널 프로세서에 비하여 상대적으로 메모리 크기가 크고 작업처리속도가 빠르기 때문에 채널 프로세서에서 처리된 결과를 병합하는 역할을 한다.

2.2 GPGPU

GPU는 하드웨어 구조의 대부분이 연산을 위한 streaming processor(SP)로 이루어져 있다. GPU의 프로세서 구조는 스트리밍 멀티프로세서(streaming multiprocessor: SM)의 집합으로 이루어져 있으며, 각 SM은 스트리밍 프로세서(streaming processor: SP)의 집합으로 이루어져 있다 [3]. GPU에서 응용프로그램을 수행 시, 하나의 응용프로그램은 여러 개의 스레드 블록으로 구성되고, 스레드 블록은 여러 개의 스레드로 구성된다. 하나의 스레드 블록은 하나의 SM에 할당되어 수행되고, 스레드 블록내의 각 스레드는 SP에 의해 수행된다 [3].

3. 실험

본 실험의 비교대상으로는 IHP과 ISP, 그리고 iSSD, GPU, CPU를 협업시킨 collaborative processing (CP)을 사용하였다. CP와 ISP 방식은 이질 프로세서들로 이루어져 있기 때문에 이질 스케줄링 알고리즘인 GDL과 BIL을 적용하였다. 따라서 CP는 GDL과 BIL로 분류된다.

실험에 사용한 매개변수는 FMC core 개수, SSD core 개수, FMC/SSD core rate, CPU core 개수이고, 기본값은 <표 1>과 같다. 실험에서는 하나의 매개변수를 변화시키는 동안 다른 매개 변수들은 기본값으로 고정시키고 성능을 측정하였다.

실험에 사용한 프로그램은 PageRank 알고리즘[6]을 수행하는데, PageRank 알고리즘은 행렬과 벡터의 곱셈을 반복적으로 수행하여 결과로 생성된 벡터가 수렴할 때까지 반복적으로 수행된다. 이 때, 행렬의 한 행과 벡터의 곱셈을 수행하여 하나의 원소를 생성하는 작업과 생성된 원소를 취합하여 새로운 벡터를 생성하는 작업이 수행된다.

<표 1> 실험 매개 변수의 기본값

매개 변수	값
# of FMC cores	32
# of SSD cores	2
FMC, SSD core rate (MHz)	200,400
# of CPU cores	2

3.1 기본 값에 대한 성능 평가

본 실험은 각 시스템의 성능을 비교하기 위한 실험이다. <표 2>는 각 시스템의 수행시간을 나타낸다. BIL을 사용한 CP가 가장 좋은 성능을 보였다. IHP 방식과 비교했을 때, ISP 는 1.35배, GDL는 2.78배, BIL은 5.26배 좋은 성능을 보였다. CP만을 비교하였을 때, BIL 방식이 GDL 방식에 비해 1.88배 좋은 성능을 보였다. 이는 프로세서의 종류가 4개이고 처리할 작업의 단계가 많기 때문에, GDL에 비하여 BIL의 스케줄링 성능이 좋기 때문이다[4].

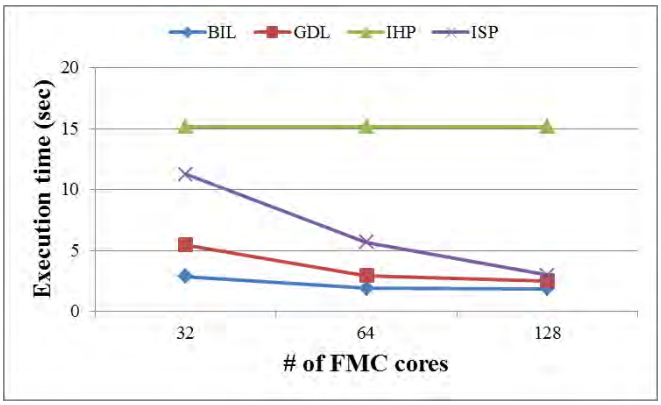
<표 2> 기본 매개 변수에서의 수행 시간

	IHP	ISP	GDL	BIL
수행 시간 (초)	15.19	11.25	5.45	2.89

3.2 FMC core 개수에 따른 각 방법의 성능 변화

본 실험은 FMC core의 개수에 따른 각 방법의 성능 변화를 보인 실험이다. 이때, FMC core의 개수를 32개에서 128개까지 증가시켰다. (그림 3)은 FMC core 개수에 따른 실험 결과를 나타낸 그래프로, X축은 매개변수인 FMC core 개수를 나타내고, Y축은 그에 따른 수행 시간을 나타낸다.

실험 결과, ISP 방식의 경우에는 행렬과 벡터의 곱셈이 주로 이루어지는 FMC core 개수에 가장 큰 영향을 받기 때문에, FMC core 개수가 32개에서 128개로 4배 증가할 때 3.75배의 성능 향상을 보였다. BIL과 GDL의 경우에는 FMC core뿐 아니라 GPU와 CPU도 사용하기 때문에 각각 1.54배, 2.18배 성능이 향상되었다. IHP의 경우 iSSD를 사용하지 않기 때문에 성능변화가 없었다.

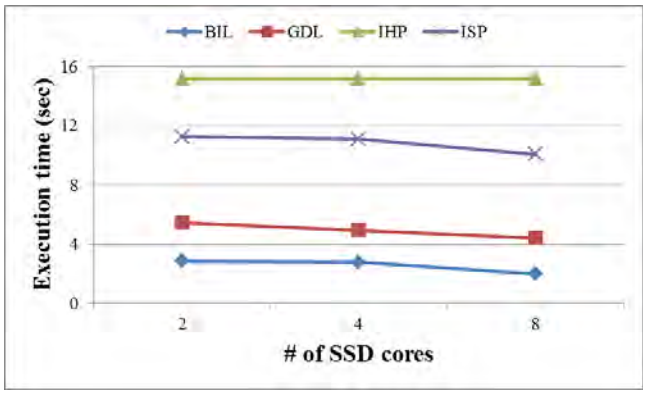


(그림 3) FMC core 개수에 따른 성능 평가

3.3 SSD core 개수에 따른 각 방법의 성능 변화

본 실험은 SSD core의 개수에 따른 각 방법의 성능 변화를 보인 실험이다. 이때, SSD core의 개수를 2개에서 8개까지 증가시켰다. (그림 4)는 SSD core 개수에 따른 실험 결과를 나타낸 그래프로, X축은 매개변수인 SSD core 개수를 나타내고, Y축은 그에 따른 수행 시간을 나타낸다.

실험 결과, SSD core 개수가 2개에서 8개로 증가할 때 ISP는 약 1.11배, GDL은 1.23배, BIL은 1.43배의 성능 향상을 보였다. iSSD에서 스케줄링 결과에 따라 SSD core는 주로 FMC에서 계산된 결과들을 취합하여 새로운 벡터를 생성하는 역할을 하기 때문에 성능 향상이 많지 않지만 SSD core 개수가 많아질수록 행렬과 벡터의 곱셈도 일부 수행하게 되어 성능 향상이 커지게 된다. IHP의 경우 iSSD를 사용하지 않기 때문에 성능변화가 없었다.

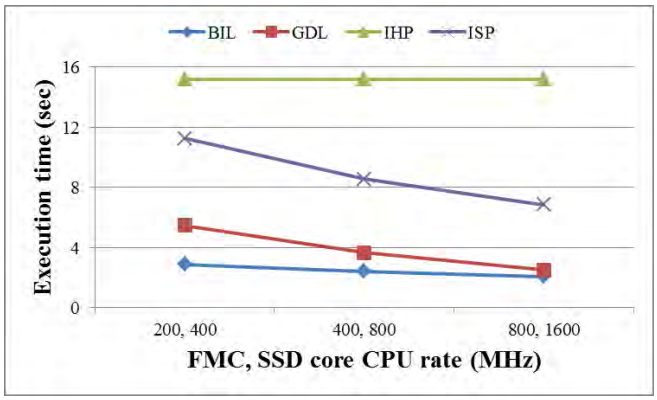


(그림 4) SSD core 개수에 따른 성능 평가

3.4 iSSD 성능에 따른 각 방법의 성능 변화

본 실험은 iSSD CPU 성능에 따른 각 방법의 성능 변화를 보인 실험이다. 이때, FMC core의 성능은 200MHz부터 800MHz까지, SSD core의 성능은 400MHz부터 1600MHz까지 증가시켰다. (그림 5)는 iSSD의 FMC, SSD core의 성능에 따른 실험 결과를 나타낸 그래프로, X축은 FMC core와 SSD core의 CPU rate를 나타내고, Y축은 그에 따른 수행 시간을 나타낸다.

실험 결과, ISP 방식은 1.64배, BIL 스케줄 방식은 1.40배, GDL 스케줄 방식은 2.18배의 성능 향상을 보인다. FMC core와 SSD core의 성능 향상은 FMC core수의 증가와 마찬가지로 행렬과 벡터의 곱셈 연산의 수행 속도가 향상되는 효과가 있다. 하지만 데이터 I/O에서 병목현상이 발생해 FMC core수의 증가에 비해 성능 향상이 적다. IHP의 경우 iSSD를 사용하지 않기 때문에 성능변화가 없었다.

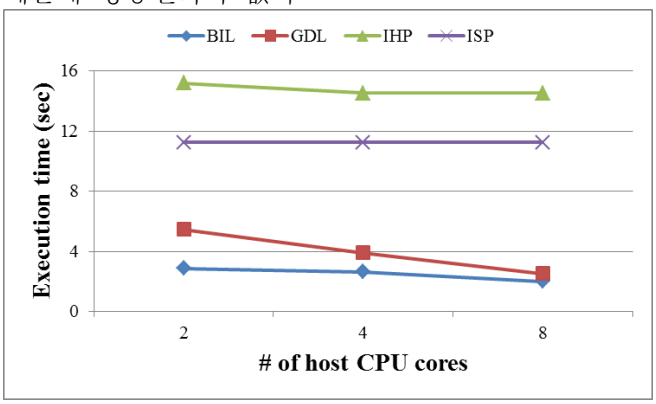


(그림 5) iSSD 성능에 따른 성능 평가

3.5 CPU core 개수에 따른 각 방법의 성능 변화

본 실험은 CPU core의 개수에 따른 각 방법의 성능 변화를 보인 실험이다. 이때, CPU core의 개수를 2개에서 8개까지 증가시켰다. (그림 6)은 CPU core 개수에 따른 실험 결과를 나타낸 그래프로, X축은 매개변수인 CPU core 개수를 나타내고, Y축은 그에 따른 수행 시간을 나타낸다.

실험 결과, IHP 방식의 경우 데이터 I/O에서 병목현상이 발생해 CPU core 개수가 4배로 증가할 때 성능 향상이 4.6%로 거의 향상되지 않았다. BIL과 GDL 스케줄 방식의 경우 CPU에서 처리하는 데이터가 많아지면서 성능이 향상되지만, iSSD와 CPU 사이에는 IPC가 존재하기 때문에 성능이 각각 1.43배, 2.16배 향상되었다. ISP 방식의 경우 CPU를 사용하지 않기 때문에 성능변화가 없다



(그림 6) CPU core 개수에 따른 성능 평가

4. 결론

본 논문에서는 프로그램의 수행 성능을 향상시키기 위해 CPU와 iSSD, GPU의 협업을 위해 이질 스케줄링 알고리즘을 적용하는 방법을 연구하였다. 실험을 통해 협업 시스템을 구성하는 장치들의 성능변화와 스케줄링 알고리즘에 따른 전체 협업 시스템의 성능을 분석하였다. 실험 결과를 통해 협업 시스템을 구성할 때 FMC core 개수가 성능에 가장 큰 영향을 주고, BIL을 이용한 방식이 좋은 성능을 나타냄을 확인하였다.

참고문헌

[1] Duck-Ho Bae, Jin-Hyung Kim, Sang-Wook Kim, Hyunok Oh, Chanik Park, “Intelligent SSD: A Turbo for Big Data Mining”, In Proc. of ACM Int'l Conf. on Information and Knowledge Management, ACM CIKM, 2013.

[2] 조성우, 조용연, 배덕호, 김상욱, 오현옥, “Intelligent SSD 를 위한 이질 스케줄링 알고리즘 활용 방안,”한국정보과학회, 2013.

[3] 정영훈, “CUDA 병렬 프로그래밍”, 프리렉, 2011.

[4] Gilbert C. Sih, Edward A. Lee, “A Compile-Time Scheduling Heuristic for Interconnection-Constrained Heterogeneous Processor Architectures”, In IEEE Transactions on Parallel and Distributed Systems, Vol. 4, No. 2, pp. 175-187, 1993.

[5] Hyunok Oh, Soonhoi Ha, “A Static Scheduling Heuristic for Heterogeneous Processors”, In Euro-Par’96 Parallel Processing, pp. 573-577, 1996.

[6] Larry Page, Sergey Brin, Rajeev Motwani, Terry Winograd, “The PageRank Citation Ranking: Bringing Order to the Web”, 1999.