저장장치 및 캐쉬를 고려한 저전력 알고리즘 설계

박기홍*
*고려대학교 전기전자전파공학부
e-mail:khpark85@korea.ac.kr

Design of energy-efficient considering cache and storage algorithms

Ki-Hong Park*
*School of Electric Engineering, Korea University

요 약

병렬 및 분산 컴퓨팅 영역에서 기존 실험의 대부분은 속도 개선만을 고려하여 알고리즘을 설계 하였다. 이 연구는 일정 수준의 속도 개선을 보이면서도 에너지 절감효과를 기대하고자 저장장치 및 캐쉬의 활용을 생각하여 알고리즘을 설계하는 연구를 진행 한다. 이를 보이기 위해 입출력 비중이 높은 경우를 대표하는 외부 정렬 실험과 순수 연산의 비중이 높은 경우를 대표하는 매트릭스 실험을 하였다. 연구 결과를 통해 저장장치 및 캐쉬를 고려한 알고리즘이 그린 컴퓨팅에 이바지 할 수 있다는 것을 말하고자 한다.

1. 서론

환경을 파괴하지 않고 지속될 수 있는 IT를 유지하며 IT를 활용함으로써 IT 스스로 친환경 보존에 공헌 할 수 있는 그린 IT(Green IT)가 대두됨에 따라 컴퓨팅 분야에서도 그린 컴퓨팅(Green Computing)을 고려하지 않을 수 없게 되었다. 그린 컴퓨팅이란 컴퓨터와 관련 시스템의 설계, 생산, 사용, 폐기 과정 등에서 환경에 최소한의 영향을 미치도록 연구 실천하는 행위이다[1]. 이 연구는 그린 컴퓨팅 범주 안에서도 저장장치(storage)와 캐쉬(cache)를 고려한 알고리즘 설계를 통한 전력 절감 효과에 그 주안점을 두고자 한다.

2. 관련연구 및 실험환경

멀티 코어 프로세서 환경에서는 일반 클러스터(cluster) 시스템에서와는 달리 서로 다른 프로세서가 데이터를 교환하는 경우 네트워크가 아닌 저장장치(디스크)를 직접 통한다. 서로 다른 코어에서 디스크에 동시 접근하는 경우 코어들의 접근 충돌로 인한 오버헤드가 발생하여 병렬 및 분산 컴퓨팅에 대한 성능효과를 저하 시킨다. 본 연구에서는 입출력 비중이 높은 외부 정렬(external sort) 실험을 하드웨어적인 측면과 소프트웨어적인 측면으로 접근하여 성능 개선을 시도 한다.

프로세서는 프로그램을 실행하기 위하여 디스크로부터 프로그램 코드와 데이터를 읽어오고, 필요한 경우에는 연산 처리의 결과를 디스크에 저장하기도 한다. 이 과정에서 프로세서와 디스크의 속도가 일치하지 못하기 때문에 디스크의 실행이 완료될 때 까지 프로세서가 기다려야 하는데 현재 컴퓨터 시스템에서는 프로세서와 주기억 장치 사이에 캐쉬를 삽입함으로써 이 둘의 속도 차이를 보완하고 있다[2]. 즉, 캐쉬 사이즈를 고려한 알고리즘 설계는 캐쉬의 적중률(hit ratio)을 향상 시킬 수 있으며, 이는 벤치마크(benchmark)를 통한 성능 예측과 그 맥락을 같이 할 수 있다.

실험환경은 하드웨어의 다양성을 고려해서 표 1에 수록한 바와 같이 싱글코어, 듀얼코어, 쿼드코어 프로세서가 탑재된 컴퓨터를 실험대상으로 하며 병렬연산 실험은 64-bit Fedora 17 Linux 환경에서 MPICH-2 라이브러리를 활용하여 수행한다. 소모 전력은 컴퓨터 전체에 공급되는 AC 전력량을 측정하여 이를 세분화하지 않고 컴퓨터에 장착된 모든 장치의 값을 측정한다.

<표 1> 실험 대상의 컴퓨터 / CPU 상세사양

CPU	sempron 140	E7200	I7-3770
CPU clock	2.70GHz	2.54GHz	3.40GHz
NO. of cores	1	2	4
RAM / cache capacity	2GB / 64KB	4GB / 32KB	8GB / 32KB
Storage type	HDD	HDD	HDD
Idle to peak power, Watts	53 - 70W	49 -135W	36 - 79W

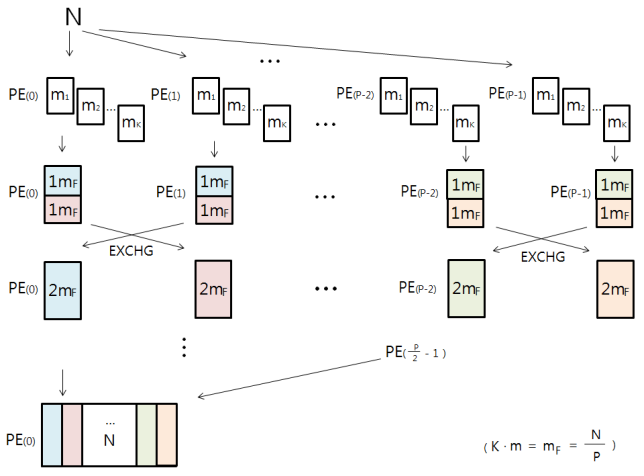
3. 저장장치 설계

3.1 하드웨어적인 접근의 성능 개선

이전 연구[3]에서 이진트리 알고리즘으로 병합하는 외부 정렬 (external sort) 실험을 하였다. 위의 실험은 클러스터 시스템에서는 네트워크를 통하여 데이터 이동이 이루어지므로 프로세서 각자의 디스크를 통한 파일 입출력이 가능하지만 멀티 코어 프로세서 환경에서는 별다른 설정이 없다면 디스크를 공유하는 체계이므로 클러스터와 동일한 방식의 운용은 디스크 접근 충돌에 의한 오버헤드가 발생하게 되어 병렬 실행에 대한 에너지 효율이 떨어지게 된다. 따라서 디스크를 추가한 후 코어별로 절대 주소를 부여하는 방식으로 파일 입출력 시 발생하는 오버헤드를 감소 시켰다.

3.2 소프트웨어적인 접근의 성능 개선

이전 연구[3]에서 사용한 이진트리 알고리즘으로 병합하는 과정은 단계가 진행될수록 점차적으로 idle 상태의 프로세서 수가 늘어나 불필요한 전력이 소모된다. 이는 병합하는 단계에서 한 프로세서가 다른 프로세서에 데이터 파일을 전송하고 나면 자신이 맡은 작업을 끝마치게 되는데 병렬연산의 특성상 모든 프로세서의 작업이 끝날 때까지 idle 상태에서 기다리기 때문이다. 위의 알고리즘을 기준으로 PSRS 알고리즘에서 pivot을 선정하고 pivot에 따라 데이터를 분할 및 전송하는 과정을 적용하면 그림 1에서와 같이 프로세서가 동시에 작업을 진행 하는 경우가 증가하여 전력 소모량이 감소한다. 데이터 교환 시 데이터가 한 프로세서에 집중될 수 있으므로 pivot은 각 프로세서 데이터의 중앙값의 평균값으로 정한다. 데이터 접근 시 파일 포인터를 사용하여 파일을 입출력하여 모든 프로세서에서 같은 파일의 접근을 가능하도록 설계 한다.



(그림 1) pivot이 데이터의 중앙값일 때
버퍼크기 m을 이용한 머지 순차정렬의 병합과정

4. 캐쉬를 고려한 설계

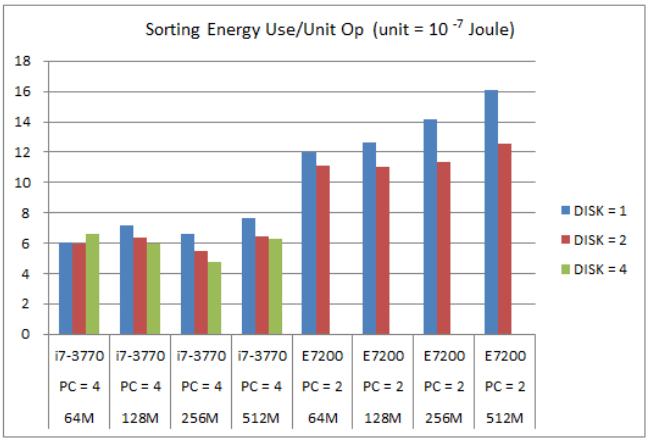
이전 연구[4]에서 행렬 곱셈의 성능 개선을 위한 시도로서 행렬 곱을 분할정복(divide-and-conquer) 형식으로 실행 하여 에너지 절감 효과가 있었다. 이전 연구에서 분할정복 알고리즘을 1단계 적용하였는데 본 연구에서는 캐쉬를 고려한 설계를 통해 이를 좀 더 구체화하여 최적의 효율이 나타나는 조건을 구한다.

행렬 곱을 위한 다단계의 소규모행렬로 블록화 할 때 소규모행렬들의 주소를 전체 행렬의 주소로 사용 할 경우 행(row)의 마지막 주소와 그 다음 열(column)의 첫 번째 행의 주소가 전체 행렬의 열의 크기만큼 주소 점프(stride distance)가 발생하므로 주소패턴을 연속화하고자 블록화된 행렬들에 독립적인 주소를 부여한다. 소규모행렬들의 크기는 캐쉬 사이즈 보다 작을 경우 캐쉬를 여러 번 읽어와야 하고 캐쉬 사이즈 보다 클 경우 캐쉬 미스가 발생하므로 캐쉬 사이즈와 같게 한다.

5. 실험 결과

전력 소모량을 비교할 시 전력소모의 절대치는 CPU와 부가적으로 장착된 장치의 종류의 특성 등에 따라 차이를 보이므로 같은 플랫폼에서 대상의 알고리즘을 실행하고 동일한 디스크를 추가하는 방식으로 전력을 측정 한 후 단위 연산당 에너지 소모량으로 변환한다.

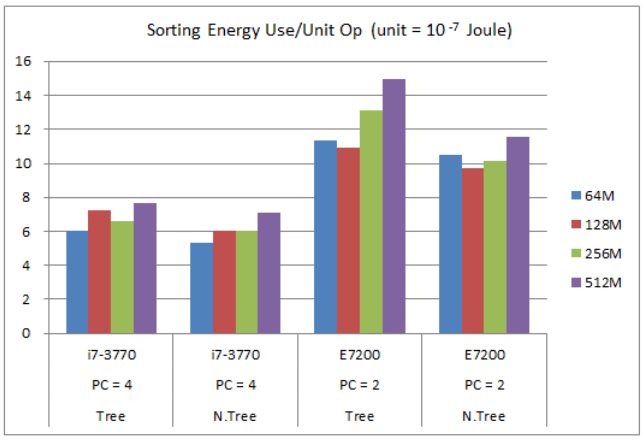
그림 2는 디스크 추가 개수에 따른 에너지 소모량을 표시하는데 데이터의 크기가 작을 경우 정렬하는 시간이 몇 초 단위에 불과하므로 전력 소모량을 비교하는데 큰 의미가 없다. 에너지 효율은 디스크를 추가할수록 좋아지나 점점 그 차이가 줄어 디스크 자원이 제한적일 경우 여러 개의 디스크를 하나의 컴퓨터에 장착하는 것 보다 최소한의 디스크를 여러 개의 컴퓨터에 분할하여 장착하는 상황에서 최고의 효율이 나타난다. 또한 데이터의 크기가 클수록 디스크 입출력이 증가 하므로 디스크 추가에 따른 효율은 높아진다. 위와 같은 결과는 디스크 추가나 작업정도에 따른 에너지 소모가 발생하나 그 변화정도가 전체 에너지 소모량에 비해 상대적으로 미비함으로 결과적으로 전체 전력소모량이 감소하게 된다.



(그림 2) 디스크 개수에 따른 전력 소모량 비교

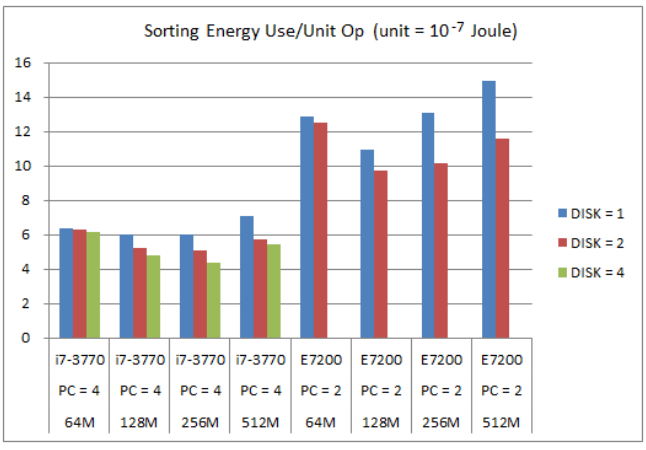
설계부분에서 거론한 새로운 알고리즘(N.Tree)과 기존의 이진트리 알고리즘의 실험 데이터를 비교한 그림 3을

보면 프로세스(PC: Processor Count)가 4개인 CPU에서 사용했을 경우 보다 프로세스가 2개인 CPU에서 사용 했을 경우 뚜렷한 개선 효과를 보이는데 이는 파일 입출력 시 발생하는 충돌이 프로세스가 2개에서 4개로 증가하는 경우 충돌 횟수가 2배로 증가하는 것이 아니라 전체적인 코어에 모두 영향을 끼치므로 상호간에 2배 이상의 더 많은 충돌이 발생하여 전력 효율은 2배 이하로 감소한다. 또한 전반적으로 데이터의 크기가 클수록 뚜렷한 개선효과를 보이는데 이는 위의 알고리즘에서 파일 입출력이 차지하는 비중이 크기 때문이다.



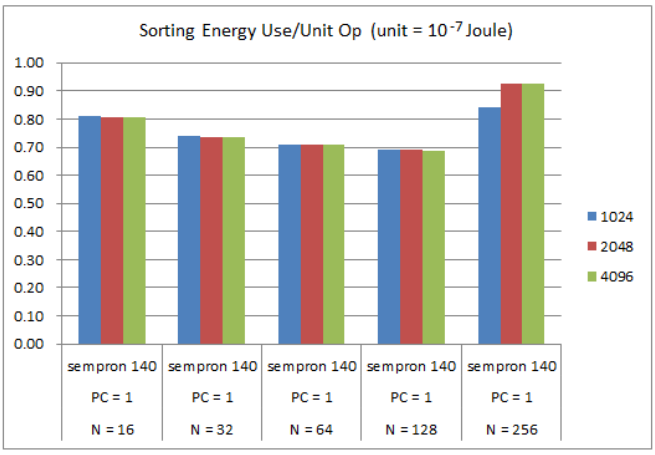
(그림 3) 외부정렬의 단위 연산당 에너지 소모량

파일 입출력 시 충돌에 따른 에너지 효율의 차가 크므로 이 충돌을 줄이기 위해 새로운 알고리즘을 적용한 후 동시에 데이터에 접근하는 횟수를 줄이기 위한 디스크를 추가하면 그림 4에서와 같이 디스크의 개수와 데이터 크기가 클수록 효율이 높아지며 최대 20%가량 전력 소모량이 감소한다.



(그림 4) 외부정렬의 단위 연산당 에너지 소모량

그림 5, 그림 6, 그림 7은 캐쉬와 블록 크기(N: 분할되는 행렬의 크기)의 관계를 분석하기 위해 CPU의 종류를 달리하고 행렬의 크기가 1024, 2048, 4096일 경우 분할되는 행렬의 크기를 16, 32, 64, 128, 256으로 증가시키면서 단위 연산당 에너지 소모량을 비교해 보인다. 그림 5에서 블록 사이즈가 128일 경우 최적의 효율이 나타난다. 블록 사이즈가 128이하일 경우 캐쉬를 부분적으로 사용하게 되므로 사이즈가 128보다 작을수록 캐쉬 블록에 반복적으로 호출되는 수는 증가 하고 에너지 소모량은 커진다. 블록 사이즈가 128이상일 경우 부분 행렬의 한 행을 캐쉬 블록에 담을 수 없으므로 블록 사이즈가 128보다 클수록 에너지 소모량이 커진다. 결과적으로 블록 사이즈가 128인 데이터에서 극소점 형태를 보이고 최대 30%가량 전력 소모량이 감소한다.



(그림 5) 행렬연산의 단위 연산당 에너지 소모량

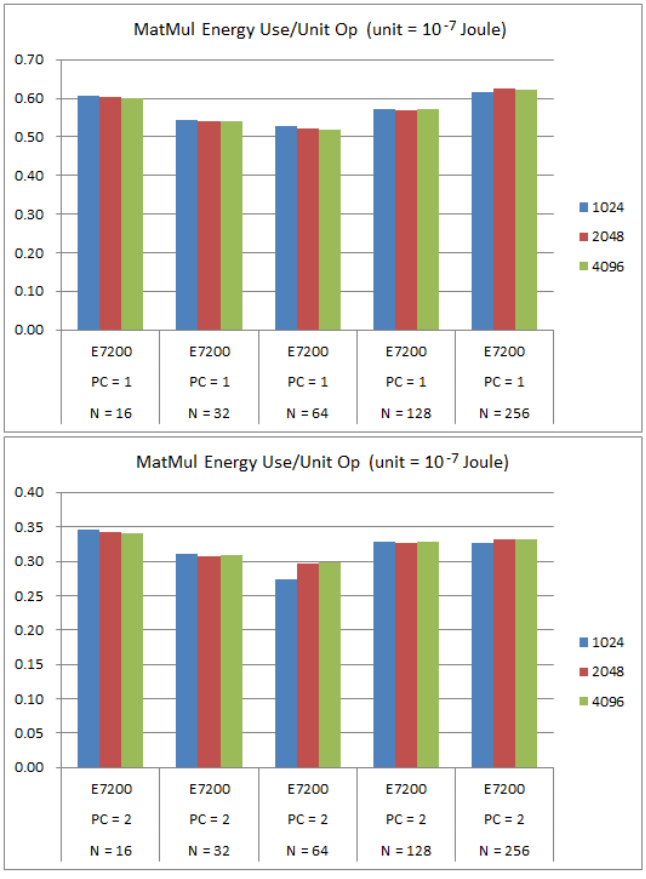
클러스터 시스템에서는 프로세스마다 독립적인 컴퓨터(machine)를 사용하기에 멀티 코어 환경이 아닌 이상 프로세스 수를 증가 시킬 때마다 전력 소모가 크게 증가하게 되는데 위의 실험은 멀티코어 시스템만으로 실행 하였기에 듀얼 코어로 실험한 그림 6과 쿼드 코어로 실험한 그림 7에서 프로세스의 개수가 증가하면 에너지 소모량은 감소한다. 이는 멀티 코어 시스템에서 코어 하나당 소모되는 전력이 전체 에너지 소모량에 일정부분 영향을 미치긴 하지만 일반적으로 입출력 장치나 기타 부가 장치들을 공유하기 때문에 전체 에너지 소모량은 감소한다. 그림 6과 그림 7에서 사용한 CPU의 캐쉬의 크기가 같고, 그림 5에서 사용한 캐쉬의 크기의 절반이므로 두 그림 모두 블록의 크기가 128의 절반인 64일 경우 최적의 효율이 나타나며 각각 최대 20%, 30%가량 전력 소모량이 감소한다.

6. 결론

병렬 및 분산 컴퓨팅 영역에서 기존 실험의 대부분은 속도 개선만을 고려하여 알고리즘을 설계 하였지만 이 연구는 일정 수준의 속도 개선을 보이면서도 에너지 절감효과를 기대하고자 저장장치 및 캐쉬의 활용을 생각하여 알고리즘을 설계하는 연구를 진행 하였다. 이를 보이기 위해 입출력 비중이 높은 경우를 대표하는 외부 정렬 실험과 순수 연산의 비중이 높은 경우를 대표하는 매트릭스 실험을 하드웨어적인 방법과 소프트웨어적인 방법을 통해 접

근하였다. 하드웨어적인 방법으로 디스크를 추가하여 프로세스의 디스크 충돌을 억제하였고, 소프트웨어적인 방법으로 기존의 이진트리 알고리즘으로 병합하는 부분에서 프로세스의 병렬화를 극대화 하여 불필요한 전력 소모를 감소 시켰다.

병렬 및 분산 컴퓨팅 영역에서 알고리즘을 설계할 때 클러스터 시스템과 멀티코어 시스템을 구분지어 설계할 필요가 있다. 디스크의 경우 클러스터 환경에선 각자 독립적인 디스크를 사용하기 때문에 디스크의 영향이 미비하지만 멀티코어 시스템의 경우 이와는 상반된다. 캐쉬 사이즈를 고려하여 알고리즘을 설계할 경우 다방면을 고려해야 한다. 데이터의 분할 정복에 있어 최적의 조건이 캐쉬의 값과 동일하지 않은 이유는 행렬연산 시 소규모 된 블록 값만이 캐쉬에 들어가는 것 이 아니라 여타 다른 연산과 같은 조건도 포함되기 때문이다. 앞으로 Bitonic sort, Odd-even Merge Sort 등 정렬 단계를 최소화 할 수 있는 알고리즘을 저전력형으로 실행하는 연구를 진행할 계획이다.



(그림 6) 행렬연산의 단위 연산당 에너지 소모량

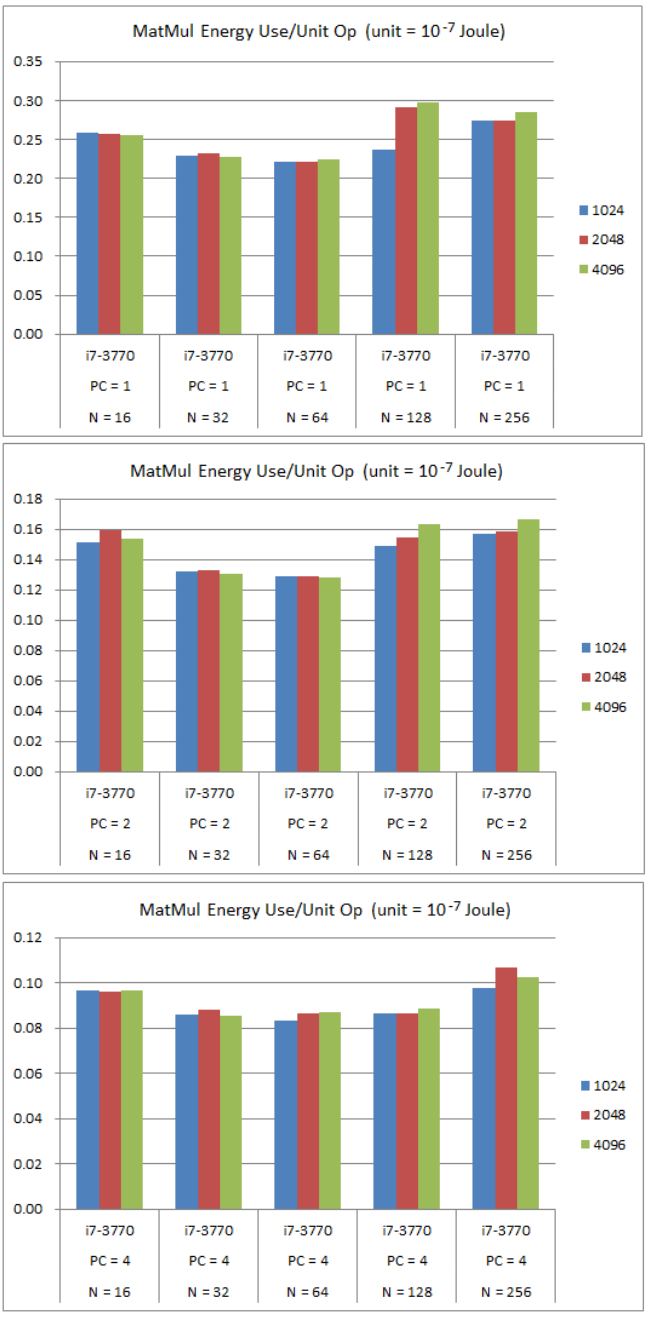
참고문헌

[1] Murugesan, San "Harnessing Green IT:Principles and Practices" IT Professional, Jan. 2008

[2] David E. Culler, Parallel Computer Architecture:a hardware/software approach, 1999

[3] 김동승, 박기홍, 외부정렬 및 행렬곱셈의 저전력 알고리즘 설계, 제 37회 한국정보처리학회 춘계학술발표대회 논문집 제19권 제1호 (2012, 4)

[4] 김동승, 박기홍, 외부정렬 및 행렬곱셈의 저전력 알고리즘 설계, 제 38회 한국정보처리학회 추계학술발표대회 논문집 제19권 제2호 (2012, 11)



(그림 7) 행렬연산의 단위 연산당 에너지 소모량