고도 컴퓨팅의 저전력 설계 연구*

김동승, 박기홍
고려대학교 전기전자전파공학부
e-mail: dkimku@korea.ac.kr

Design of energy-efficient high-performance computing algorithms

Dongseung Kim and Ki-Hong Park
School of Electrical Engineering, Korea University

고도 컴퓨팅용 응용 프로그램 작성시 “에너지”를 감안하여 가급적 최단시간 내에 최소의 에너지로 해당 연산을 완료하는 설계정책을 고안하고자 한다. 이 논문은 연산과 I/O 비율이 상이한 두 알고리즘에 대해 에너지 소모특성분석, 전력효율 평가를 통해서 고효율 연산 설계법을 제시한다. 정렬 및 매트릭스 곱셈 알고리즘을 대상으로 실험하였고, 연구결과는 대규모 데이터 처리, 가공 등의 영역에서 그린 컴퓨팅을 실현하는데 기여할 것이다.

1. 서론

컴퓨팅 분야에서도 에너지 사용량을 줄여 녹색 환경을 지키려는 노력의 일원으로 그린 컴퓨팅 (green computing)이 강조되면서 종전의 최단시간에 주어진 계산을 완료하는 성능위주의 컴퓨팅 방식에서 저전력 컴퓨팅이 중요 설계 요구사항으로 등장하게 되었다 [1,4] 동일한 데이터에 대해 병렬연산을 실행할 경우 단독 시스템으로 계산하는 것에 비해 실행시간 단축효과는 쉽게 예상할 수 있으나 전력소모가 증가하는지, 감소하는지, 또한 어떤 조건하에 최적의 연산이 되는지가 불확실하다. 본 연구는 그린 컴퓨팅을 위한 기초연구로 대표적 알고리즘으로 정렬, 매트릭스 연산을 병렬 실행하여 실행시간 향상, 에너지/파워 소모량, 그 효율을 분석하고 전력절감효과를 얻기 위한 방향을 제시하고자 한다.

2. 대표 알고리즘 선정

문헌[2]에 따르면, CPU 작업의 정도에 따라 에너지 소모범위가 크고, I/O 장치도 작업정도에 따른 에너지 소모가 가변되나 그 변화정도는 상대적으로 작은 것으로 알려졌다. 따라서 CPU 및 I/O 영역에서의 작업조절/스케줄링이 에너지 효율을 높이는데 효력을 발휘할 수 있다. 정해진 작업에 대해 CPU의 가동상태에 따라 전력소모율이 달라지는 것을 측정하고 모델링할 수 있도록, CPU 활용에서의 전력 비중을 구하고 I/O 연산정도에 따른 전력사용량을 추정한다. 데이터 교환, 확보에 소요되는 가동시간에 대해 순연산에 소요되는 비중이 높을수록 해당 코드의 전력사용이 높아지고 데이터 이동/접근의 빈도를 낮추는 것은 프로세서에서 연산횟수를 줄이는 것보다 훨씬 에너지 절감효과가 큰 것으로 알려져 있다. 따라서 알고리즘 설계시에 시간복잡도 위주의 성능평가지수를 수정해서 에너지 소모에 대한 비용요소에 메모리 접근비용을 강조해 수식화하는 접근이 필요하다.

고도컴퓨팅의 대표성을 갖는 알고리즘을 선택해서 병렬화한 후 그 동작특성 및 에너지 사용형태(profile)를 파악 분석하고, 전력효율을 높여 동작시키기 위한 설계요소를 파악하려 한다. 주로 순수연산 위주인 매트릭스 곱셈 알고리즘과 비교적 I/O (파일) 연산비중이 큰 외부정렬(external sort) 알고리즘[3]을 선택하였다. 각각의 실행시간과 입출력(파일 I/O) 소요시간, 속도 개선(speedup)을 해석하고 그 효과를 실험적으로 측정한다. 에너지소모량은 공평한 비교를 위해서 단위 연산당 평균에너지 소모량을 환산하도록 하였다. 하드웨어의 다양성을 고려해서 표1에 수록한 바와 같이 4 종류의 저전력 노트북 PC 및 탁상형 컴퓨터, 클러스터 시스템을 실험대상으로 한다. 파일 접근의 성능 및 에너지 효율을 비교하기 위해 HDD 는 물론 SSD(solid state disk)를 장착하고 동일 연산을 수행하도록 하였다.

[표1] 실험 대상의 컴퓨터/CPU 상세 사양

CPU	N270	L2500	i5-750	E7200
CPU clock	1.6GHz	1.83GHz	2.67GHz	2.53GHz
No. of cores	1	2	4	2
RAM /cache capacity	1GB/ 0.5M	1GB/ 2M	4GB/ 8M	4GB/ 3M
Storage type	HDD	SSD	HDD	HDD
Idle to peak power, Watts	12-17	13-21	49-135	49-62

* 이 연구는 한국연구재단 일반연구(2011-0003942)와 2011 고려대 특별연구비의 지원에 의해 수행되었음.

3. 소요시간 해석 함수 도출

(1) 병렬 정렬 모델링

파일 접근시간을 포함한 정렬 알고리즘의 소요시간을 모델링에 사용된 변수, 기호는 아래와 같다

N : 입력 크기 (정렬에서는 정수 key의 개수)

m : 주메모리 크기, 주메모리에 로드되는 key 개수

P : 프로세서 또는 코어 개수

T_C , T_F ,T_X: 각각 key 한개당 비교시간/ 파일 읽기,쓰기 시간/ 네트워크로 전송 시간

$$K \equiv \frac{N}{m} \ , \ k = \frac{N}{m} \cdot \frac{1}{P}, \quad (K = kP) \cdots (Eq.0)$$

순차정렬 시간은 그림1과 같이 초기에 size-m으로 입력을 분할하여 sort-run을 만들어 파일로 저장 한 후, 각 sort-run을 2진 트리 알고리즘으로 병합하여 크기 N인 sort-run (파일)을 생성하여 완료된다. 따라서

$$\begin{aligned} & t_{seq}(N) \\ &= K(m\log m)T_C + NT_F + (\log K)N(T_C + T_F) \\ &= (Km\log m + N\log K)T_C + (N + N\log K)T_F \\ &= (N\log N)T_C + N(1 + \log\frac{N}{m})T_F \cdots (Eq.1) \end{aligned}$$

한편, 병렬 정렬은 우선 root 프로세서가 N key가 저장된 입력 데이터 파일을 읽어 P 프로세서에 동일한 크기로 분산(Scatter)하되, 크기-m 씩 블록으로 만들어 전송하는 과정을 반복한다. 각 프로세서는 도달한 블록을 별개의 파일로 저장한다. 이후 각 프로세서는 내부적으로 위 순차정렬 방식을 써서 모든 블록을 읽어 sort-run을 생성하고 트리 알고리즘을 써서 크기 $\frac{N}{P}$의 run(파일)로 병합하여 저장한다. 이후 프로세서간 트리 알고리즘으로 P개의 $\frac{N}{P}$-runs 을 $\log P$ 단계를 거쳐 병합하여 최종 크기 N의 정렬된 파일을 생성한다. 총소요시간은

$t_{PAR}(N) = t_{Loc}(N) + t_{Pmerge}(N)$ 인데

$$\begin{cases} t_{Loc}(N) = km(T_F + T_X) + km\log m\, T_C \\ \qquad\quad + \ km\, T_F + (\log k)km(T_C + T_F) \\ t_{Pmerge} = (\log P)N(T_C + T_F + T_X) \end{cases}$$

$$t_{Loc}(N) = \frac{N}{P}\log\frac{N}{P}T_C + \frac{N}{P}(2 + \log k)T_F + \frac{N}{P}T_X$$

위 식은 (Eq.0)을 대입한 결과이다. 전체 관계식은

$$t_{PAR}(N) = \frac{N}{P}(P\log P + \log\frac{N}{P})T_C$$

$$+ \frac{N}{P}(P\log P + 2 + \log k)T_F + \frac{N}{P}(1 + P\log P)T_X \cdots (Eq.2)$$

병렬수행의 speedup은 $S = \frac{t_{Seq}(N)}{t_{PAR}(N)}$ 은 (Eq.1)과 (Eq.2)의 관계에 의해 구해지며, scalability 속성이 낮음을 알 수 있다.

이제 전력소모에 큰 영향을 주는 요소인 입출력(I/O) 연산에 드는 에너지량을 감안하도록 병렬 실행에서 파일 입출력 연산량/시간 $t_{FILE}(N)$ 을 구한다.

$$t_{File}(N) = t_{Scatter}(N) + t_{Local\_sort}(\frac{N}{P}) + t_{Global\_merge}(N)$$

구성요소 각각을 구해보면

$$t_{Scatter}(N) = (\frac{N}{mP}\cdot m)P = N$$

$$t_{Local\_sort}(\frac{N}{P}) = \frac{N}{P}(1 + \log k)$$

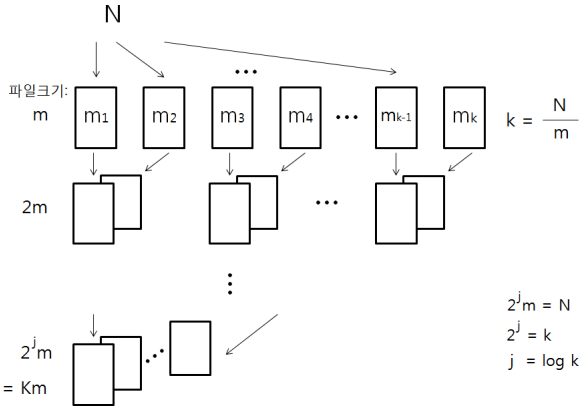

그림1 주메모리 버퍼크기 m을 이용한 머지 순차정렬의 병합과정

$$t_{Grobal\_merge}(N) = \frac{k}{2}(m + m) + \frac{k}{4}(2m + 2m)$$

$$+ \cdots + \frac{k}{k}(\frac{km}{2} + \frac{km}{2})$$

$$= km \cdot \log k = \frac{N}{P}\log k$$

따라서 $t_{File}(N) = N + \frac{N}{P}(1 + \log k) + \frac{N}{P}\log k$

$$= \Theta(\frac{N}{P}\log k)$$

$$= \Theta(\frac{N}{P}\log(\frac{N}{P}\cdot\frac{1}{m}))$$

(2) 매트릭스 곱셈 모델링

C = AXB 병렬 매트릭스 곱셈 알고리즘은 A 매트릭스를 N/P 열씩 분할하여 각 프로세서에 전송하고, B는 broadcast에 의해 공통으로 저장한 후 개별 프로세서가 동시에 할당된 C의 부분 매트릭스(N/P 열)를 계산하고 루트 프로세서로 반송하는 알고리즘을 썼다. 소요시간은 단계별 실행과정에 의해 정해진다.

$t_{ParMat}(N) = t_{Bcast}(N) + t_{Mult}(N) + t_{Gather}(N)$

$$t_{Bcast}(N) = (\log P)\frac{N}{P}\cdot T_X$$

$$t_{Mult}(N) = \frac{N^3}{P}T_C$$

$$t_{Gather}(N) = (\log P)\frac{N}{P}\cdot T_X$$

따라서

$$t_{ParMat}(N) = \frac{N^3}{P} T_C + 2\frac{N}{P}(\log P) \cdot T_X$$

단, 데이터 이동시간은 순수 연산시간에 비해 무시될 수 있게 된다. 즉 $N^3 T_C \gg 2N(\log P) T_X$

$$t_{Matmul}(N) = \frac{N^3}{P} \cdot T_C = \Theta(\frac{N^3}{P}) \quad \cdots (Eq.3)$$

순수 곱셈 연산 동안 프로세서가 최대전력을 소모하는 관측 결과로 볼 때 전력소모량도 N^3 에 비례하여 나타난다. 순차형의 경우는 통신시간이 없이 순수연산으로 간략히 표현된다. 즉

$$t_{SeqMat}(N) = N^3 T_C \quad \cdots (Eq.4)$$

따라서, 병렬 매트릭스 연산은 scalability 속성이 잘 유지되는 알고리즘이다.

4. 실험 결과, 검토

병렬연산 실험은 Linux(Fedora 15) 환경에서 MPICH-2 라이브러리를 활용하여 수행되었다. 그림2는 순수연산 실행, 데이터 이동연산 등에 의해 순간 전력소모량과 총 에너지 사용량이 변화하는 모습을 보인다. 이 과정에서 데이터 분산과 내부정렬(local sort)과정은 최대전력으로 가동되고, 이후 프로세서간 데이터 교환과 병합하는 기간 동안은 전력량이 감소함을 보인다. 트리 알고리즘 특성상 뒷 단계로 갈수록 idle 상태의 프로세서 수가 늘어나 소비하는 전력이 감소되기 때문이다. 반면 매트릭스 연산에서는 입출력 시간은 매우 짧아 전체 기간 동안 최대전력으로 가동됨을 볼 수 있는데, 이는 순수계산 실행에는 최대전력이 소모되는 것을 의미한다.

그림3은 NXN 크기 (N=1,024~4,096) 매트릭스 곱셈연산의 speedup을 표시하는데 프로세서의 증가에 따라 비례적인 시간단축효과를 볼 수 있다. 반면, 정렬은 앞의 해석 결과와 같이 성능증가가 완만하다.

컴퓨터 시스템마다 무부하시 전력량이 상이하고, 저장장치나 RAM 크기 등이 다름을 고려해서 시스템간 전력효율치를 공평하게 비교하기가 어렵다. 그래서 같은 플랫폼에서 소모한 총 에너지량을 (정렬 알고리즘상 필요한 총 연산량인) $N\log N$ 으로 나누어 "단위연산당 평균 에너지소모량"으로 상호 비교하도록 하였다(그림 4-5, N=128, 256, 512M). 같은 플랫폼에서 볼 때 대체로 입력 데이터 크기가 증가하면 실행시간은 물론 연산당 소요에너지도 증가하는데, 가용메모리 내에서 최대크기로 설정된 버퍼로 파일을 읽고 쓰는 과정에서의 전력량이 입력 규모가 커질수록 증가함에 기인하는 것으로 추정된다. 병렬성(PC: Processor Count 즉 참여한 코어 개수)이 증가할수록 에너지값이 증가하는 것은 $\log P$ 에 비례하여 네트웍 데이터 이동시간이 증가하고 더욱이 코어 중 일부가 작업을 완료해서 더 이상 처리할 자료가 없어도 종료될 때까지 idle 상태로 기다리게 되면서 에너지 낭비가 초래한 결과이다 (E7200, PC=1,2,4,8의 경우). i5-750 4-core CPU의 경우는 동일한 파일 시스템과 주메모리를 공유하는데서 자원할당/사용에서의 충돌, 지연이 심화되어 늘어나는 것으로 판단된다. 단, N270 단일코어를 제외한 나머지 CPU의 경우 PC=1일 때와 PC=2일 때를 비교하면 후자에서 에너지 효율이 개선되는 현상이 나타나는데, 칩에 공급되는 기본전력(무부하 전력)이 동일한 상태에서 코어 1개만 연산에 쓰는 것은 2개 모두 가동하는 것에 비해 에너지 효율이 떨어지는 것 때문으로 생각된다. 프로세서끼리 비교해 볼 때 노트북 타입의 프로세서가 에너지 소모량이 작지 않게 나타난 것은 화면(display)에 공급되는 전력량이 측정치에 포함되어 수치화되었기 때문이며, 이를 보정하면 다른 탁상형 PC에 비해 낮은 전력소모를 보일 것이다.

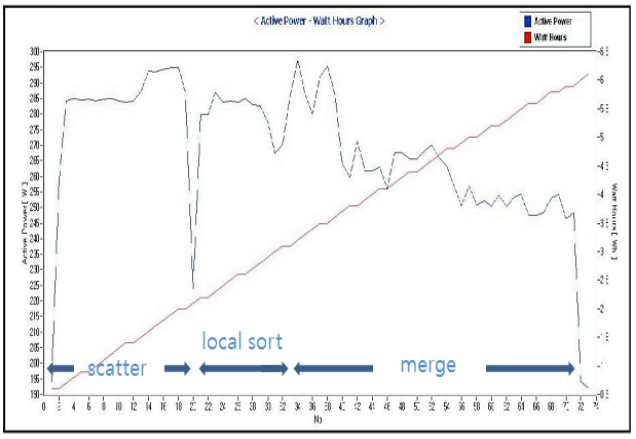

그림2 병렬 정렬의 전력측정 (E7200 클러스터, 코어사용 개수 PC=8)

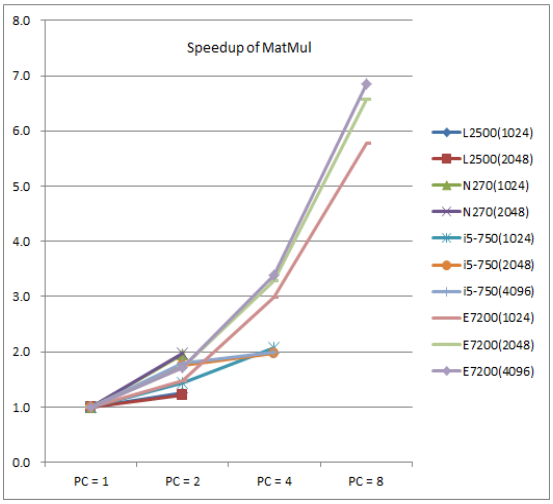

그림3 매트릭스 곱셈의 Speedup

그림 6은 동일한 CPU에서 정해진 크기의 파일에 대해 버퍼(주메모리) 크기를 가변하면서 단순 읽어오기 및 쓰기 (file read & write)를 실행하여 소요된 평균 에너지량을 비교한 결과를 보인다. 버퍼크기가 증가할수록 에너지 사용량이 줄어드는 것은, 버퍼용량을 늘이면 파일에서 주메모리 버퍼로 한번에 로드/저장할 블록크기가 커지고 따라서 블록 이동 I/O 횟수가 줄어들어 평균 접근 오버헤드가 감소하게 되고 시간

단축 및 에너지 효율 증대 효과가 나타나기 때문으로 보인다. 따라서 응용프로그램에서 버퍼크기를 크게 잡을 수 있도록 메모리 시스템 관리가 필요하다.

표2는 하드 디스크 (HDD)와 SSD(solid-state disk drive)의 에너지 효율성/단위 word의 접근 비용을 보인다. 장치별로 동일한 크기(256MB)의 데이터 블록을 파일에서 읽어와 되쓰는 동안 걸린 시간과 소요된 에너지를 측정하고, 단위 워드 당 접근시간과 에너지소모량을 계산하여 수록하였다. 예상대로 SSD의 응답속도 및 에너지 효율성이 HDD에 비해 좋게 나타났으나, 소규모 랜덤 Read, Write 가 아닌 블록형 데이터 읽기 쓰기인 관계로 SSD의 응답속도가 HDD에 비해 월등히 차이나는 현상은 보이지 않는다. 한편, PC=1의 순차실행에 비해 PC=2의 병렬수행에서 실행시간/에너지값이 증가하는 것은 동일한 파일 시스템/저장장치를 두 개의 코어가 동시에 접근하면서 자원공유/충돌에 기인한 것으로 보인다.

최근의 멀티코어 CPU에는 부하정도에 따라 일부 코어의 전원공급을 차단하여 절전가동이 가능한 것으로 알려져 있으나, 한 코어라도 대기상태로 되면 전체 CPU가 정지하는 현상이 있어 이 실험에서는 활용하지 못하였다.

5. 결론

전력효율을 높이려면 병렬 알고리즘 개발과 실행과정에서 다각도의 에너지 절약설계를 요한다. 즉,

- 병렬/멀티코어 연산은 순차연산에 비해 에너지 절약요소가 다양하다. 특히 계산완료 때까지 모든 코어가 활용되도록 하고, 사용이 끝난 순간부터 그 코어를 비활성화(sleep) 상태로 만들면 에너지 절약이 커진다.
- 컴퓨터간 통신으로 대기하는 동안 전력소모가 지속되므로 그 시간이 최소화되도록 설계한다. 이것은 최단시간 계산을 지향할 때에 이미 적용하던 사항이다
- 시스템 운영으로 유발되는 작업이나 자원 사용규모를 줄이도록 한다. (불필요한 임시파일 등을 삭제하고, 초기에 설정된 버퍼가 더 이상 필요하지 않는 경우 즉시 시스템에 반납하여 메모리 운영의 효율을 높인다).
- 파일 I/O 시간을 줄여 에너지를 절약한다. 즉, 메모리 버퍼용량을 최대한으로 확보해 단위 자료당 접근비용을 줄이게 되면 에너지 소모도 낮아진다.

참고문헌

[1] K.W. Cameron, The Road to greener IT pastures, *IEEE Computer*, May 2009, pp. 87-89

[2] W. Dally, Power Efficient Supercomputing, *LBLWorkshop on Accelerator-based Computing and Manycore,* Nov. 2009.

[3] Peter Sanders, Algorithm Engineering for Scalable Parallel External Sorting, 2011 *IEEE International Parallel & Distributed Processing Symposium*, Anchorage, Alaska USA, May 16-May 2011.

[4] 김동승, 최성운, 윤성로, 저전력 정렬알고리즘 설계, *제35회 한국정보처리학회 춘계학술발표대회 논문집* 제18권 제1호 (May 2011)

[표2] 복수 코어 사용시 저장장치 접근시간 및 소요 에너지 비교 (L2500)

Core Count	Avg.access time (nano_second)		Engy/Op (10^{-6} Joules/Op)	
	SSD	HDD	SSD	HDD
1	86.20	195.15	1.35	2.81
2	113.86	452.29	1.73	6.25

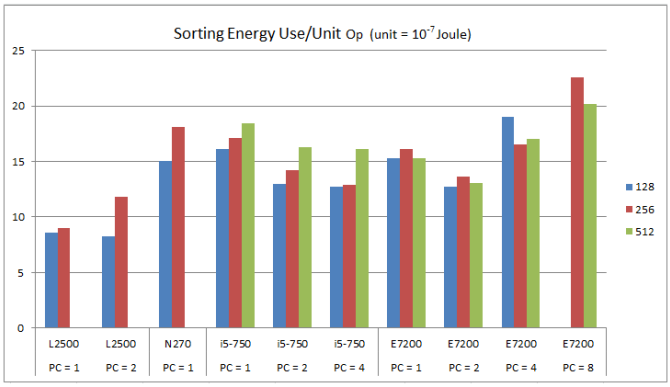

그림4 외부정렬의 단위연산당 에너지 소모량

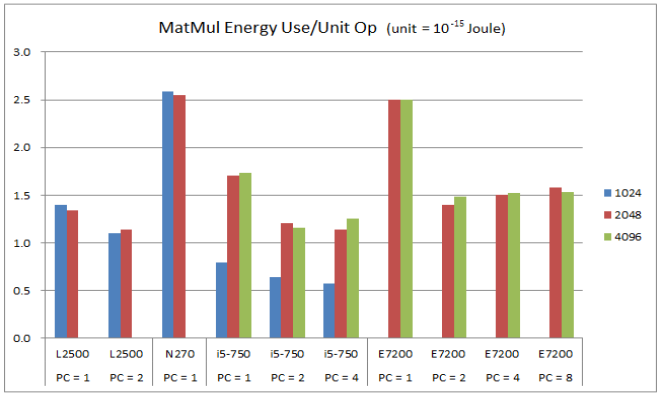

그림5 매트릭스 곱셈의 단위연산당 에너지 소모량

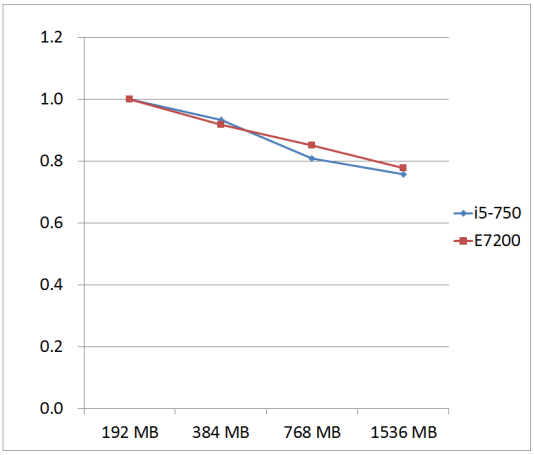

그림6 버퍼크기에 따른 주메모리/디스크 load-store의 전력소모량 비교