컴퓨팅 성능 대비 탄소 배출량 임계점 분석

백송현[1], 서인준[2], 엄영익[3]
[1]성균관대학교 소프트웨어학과
[2]성균관대학교 대학원 전자전기컴퓨터공학과
[3]성균관대학교 전자전기컴퓨터공학과/소프트웨어융합대학

xmxm00@skku.edu, sij9323@skku.edu, yieom@skku.edu

Analysis of Carbon Emissions Threshold Relative to Computing Performance

Songhyun Baek[1], Injune Seo[2], Young Ik Eom[3]
[1]Dept. of Computer Science and Engineering, Sungkyunkwan University
[2]Dept. of Electrical and Computer Engineering, Sungkyunkwan University
[3]Dept. of Electrical and Computer Engineering/College of Computing and Informatics, Sungkyunkwan University

요 약

고성능 컴퓨팅의 수요가 증가하며 컴퓨터 분야에서 탄소 배출량을 추적하고 최적화하는 것이 중요한 과제로 여겨진다. 본 연구에서는 리눅스의 탄소 사용 정보를 제공하는 Carbond를 활용해 CPU와 DRAM에 대한 탄소 배출량을 추적하고, CPU affinity를 조절하며 워크로드의 수행 시간과 탄소 배출량을 관찰한다. 실험 결과 C-ray 벤치마크에서 4개의 코어를 할당하였을 때 적은 성능 감소로 탄소 배출량을 줄일 수 있음을 확인하였다.

1. 서론

최근 컴퓨팅 기술의 발전으로 하드웨어를 비롯해 클라우드, 인공지능과 같은 분야가 빠르게 발전하고 있다. 특히 빅데이터, 인공지능 처리를 위한 고성능 컴퓨팅 자원에 대한 수요가 크게 증가하면서 높은 수준의 전력 사용량을 보이게 되었고, 이에 따라 탄소 배출량 역시 크게 증가하고 있다. 따라서 클라우드 업체에서는 재생 에너지를 운영에 이용하는 방식으로 탄소 중립 달성에 노력하고 있으나, 빠르게 발전하고 이용자가 증가하는 클라우드 시스템의 특성상 이제는 소프트웨어 수준에서도 탄소 배출량을 추적하고 분석하여 최적화하는 것이 중요한 과제로 여겨진다.

컴퓨팅 분야에서는 시스템이 사용하는 전력량에 따른 탄소 배출량을 측정하고 최적화하는 연구들이 진행되고 있다. 최근 연구[1]에서는 시스템을 구성하는 하드웨어를 생산하고 운반하는 과정에서 발생하는 탄소 배출량이 전체 탄소 배출량에서 큰 부분을 차지한다는 결과가 발표되었다. 이에 2023년 소프트웨어 영역에서 발생하는 탄소 배출량의 표준 명세(Software Carbon Intensity Specification)[2]에서는 전체 탄소 배출량을 시스템 운영 중에 전력 사용에 의해 발생하는 탄소 배출량(operational carbon emission)과 하드웨어의 생산과 운반 과정에서 발생하는 탄소 배출량(embodied carbon emission)으로 구분하는 형태로 탄소 배출량 계산의 템플릿을 표준화하여 제공하고 있다.

본 연구에서는 탄소 배출량 계산의 표준 명세를 기준으로 탄소 사용 정보를 제공하는 Carbond[3]라는 리눅스 시스템 데몬을 활용한다. Carbond에서 워크로드를 수행하는 동안 발생하는 탄소 배출량을 추적하고, 컴퓨팅 리소스를 제한하며 성능과 탄소 배출량의 추이를 관찰한다. 또한 이 과정에서 성능 대비 탄소 배출량이 많아지는 지점을 찾아 탄소 배출량을 최적화할 수 있는 방법을 찾고자 하는 것이 본 연구의 목적이다.

2. 연구방법

본 연구에서는 Carbond를 활용하여 operational carbon emission과 embodied carbon emission을 추적한다. 각각의 계산은 소프트웨어 탄소 배출량 표준에 따라 계산한다.

2.1. Operational carbon emission

표준 명세에 따르면 operational carbon emission은 소프트웨어에서 사용한 에너지와 지역에서 에너지 생산

에 따라 발생하는 탄소 배출량(carbon intensity)을 곱하여 계산한다. Carbond에서는 워크로드를 수행하는 동안 RAPL 인터페이스[4]를 활용해 프로세서와 메모리에 대한 전력량을 추적하고, WattTime API를 이용해 carbon intensity를 가져와 operational carbon emission을 계산한다. 본 연구에서는 carbon intensity는 430 gCO_2 eq/kWh으로 고정하여 operational carbon emission이 전력량에만 의존적이도록 설정하였다.

2.2. Embodied carbon emission

Embodied carbon emission은 하드웨어의 전체 embodied emission(TE, Total Embodied Emission)과 수명 대비 소프트웨어가 리소스를 사용한 시간(TS, Time Share), 전체 리소스 대비 소프트웨어가 사용한 비율(RS, Resource Share)을 곱하여 계산한다. Carbond workload를 수행하는 동안 1초에 한 번 프로세서와 메모리의 사용률을 확인하여 embodied carbon emission을 계산하고 누적 합산한다.

2.3. C-ray 벤치마크

본 연구에서는 C-ray 벤치마크를 프로세서의 코어 수를 변경해가며 코어에서 발생하는 탄소 배출량과 벤치마크 수행에 걸리는 시간을 토대로 임계점을 분석한다. C-ray 벤치마크는 부동 소수점 계산에 대한 프로세서의 성능을 ray-tracing 계산을 통해 테스트한다. 이는 RAM이나 I/O의 영향을 최소한으로 받도록 설계되었고, cache miss가 적게 발생하는 특성을 보인다.

3. 실험환경 및 실험결과

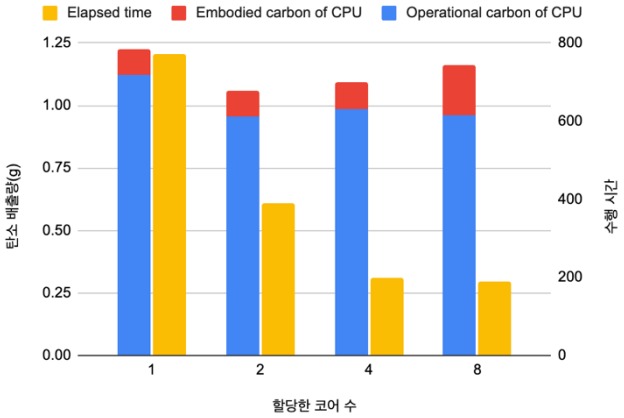
3.1. 실험환경

본 연구에서는 Intel(R) Xeon(R) CPU E3-1275 v5 프로세서를 탑재하고 32GB의 DDR4 메모리를 탑재한 시스템에서 실험을 진행한다. 또한 리눅스 커널 6.5가 설치된 Ubuntu 22.04 운영체제에서 실험을 진행한다.

3.2. 실험결과

실험은 리눅스의 taskset 유틸리티를 사용해 C-ray벤치마크를 수행할 때의 CPU affinity를 설정하고, Carbond를 활용하여 전력량, 탄소 배출량, 수행 시간을 추적한다.

실험 결과는 <그림 1>과 같이 나타난다. 할당한 코어 수가 증가할수록 수행 시간은 점차 줄어들지만, 탄소 배출량은 2개의 코어를 할당했을 때 가장 적게 배출되고 다시 점차 증가하는 모습을 보인다. 특히 4개의 코어를 할당했을 때와 8개의 코어를 할당했을 때 수행 시간은 약 10초정도의 차이를 보이는 반면, 프로세서의 사용률이 두배로 증가하기 때문에 embodied carbon emission이 크게 증가하여 전체적인 탄소 배출량이 더 높게 측정된다. 따라서 해당 환경에서는 코어 수를 2개 혹은 4개를 할당한 지점이 탄소 배출량이 최적인 지점이며, 수행 시간까지 고려한다면 4개 정도의 코어를 할당하여 적은 성능 감소로 큰 폭의 탄소 배출량 감소를 이루는 최적화가 가능함을 알 수 있다.



<그림 1> 코어 수에 따른 탄소 배출량과 수행 시간

4. 결론 및 향후 연구

본 연구를 통해 프로세서와 같은 컴퓨터 시스템의 리소스를 제한함으로써 적은 성능 감소로 적절하게 탄소 배출량을 감소시킬 수 있음을 확인하였다.

향후 연구에서는 이를 확장하여 데이터 센터와 같은 고성능 컴퓨터 시스템에서도 사용자의 불편을 최소화하며 탄소 배출량을 최적화할 수 있는지 연구를 진행하고자 한다. 또한 프로세서 이외에도 저장장치, 메모리, GPU와 같은 시스템 리소스에 대한 탄소 배출량 추적에 대한 구현과 함께, 각각의 리소스에 대한 탄소 배출량 최적화를 위한 임계점을 분석하는 연구를 진행할 예정이다.

참고문헌

[1] U. Gupta et al., “ACT: Designing Sustainable Computer Systems with an Architectural Carbon Modeling Tool,” Proceedings of the 49th Annual International Symposium on Computer Architecture, Jun. 2022.

[2] Software Carbon Intensity Specification, https://github.com/Green-Software-Foundation/sci

[3] A. Schmidt, G. Stock, R. Ohs, L. Gerhorst, B. Herzog, and Timo Hönig, “carbond: An Operating-System Daemon for Carbon Awareness,” Proceedings of the 2nd Workshop on Sustainable Computer Systems, Jul. 2023.

[4] K. N. Khan, M. Hirki, T. Niemi, J. K. Nurminen, and Z. Ou, “RAPL in Action,” ACM Transactions on Modeling and Performance Evaluation of Computing Systems, Vol. 3, No. 2, pp. 1–26, Jun. 2018.