핵 물리에서의 QCD 병렬화

사재원†, 노병준†, 김희곤†‡, 최동휘†, 이성주†, 정용화†, 박대희†, 조충호†
†고려대학교 컴퓨터정보학과
‡유럽원자핵공동연구소, 스위스, 제네바
e-mail : sjwon92@korea.ac.kr

Parallel QCD in Nuclear Physics

Jaewon Sa†, Byeongjoon Noh†, Heegon Kim†‡, Dongwhee Choi†, Sungju Lee†, Yongwha Chung†, Daihee Park†, Choong-ho Cho†
†Dept. of Computer & Information Science, Korea University
‡European Organization for Nuclear Research(CERN), Geneva, Switzerland

요 약

격자 양자 색역학(Lattice Quantum ChromoDynamics; Lattice QCD)은 자연계에 존재하는 중력, 전자기력, 약한 핵력, 그리고 강한 핵력 등의 기본적인 상호작용 중 강한 핵력의 상호작용을 이해하기 위한 핵물리 분야의 이론이다. 이 물리 역학은 몬테 카를로(Monte Carlo) 기법을 이용하여 대규모 수치 연산을 필요로 하고, 수행시간 단축을 위하여 병렬처리가 필요하다. 본 논문에서는 격자 양자 색역학에서 요구되는 대규모 수치 연산에 대하여 마이크로프로세서와 성능가속기에 최적의 작업부하 분배를 통한 이기종 병렬처리 방법을 제안하고 성능가속기만을 사용한 방법과 제안 방법의 성능을 비교한다.

1. 서론

물질을 구성하는 가장 기본적인 입자를 연구하는 입자 물리학 분야는 물질을 구성하는 입자가 무엇인지 알아내고, 그 입자들 사이의 상호작용을 알아내는 연구이다[1]. 이러한 입자 물리학 연구의 한 방법으로 양자 색역학(Quantum ChromoDynamics, QCD)을 이용한 방법이 있다. 이는 중력, 전자기력, 핵간의 인력 등과 같은 자연계에 존재하는 힘들 중에서 핵간의 강한 상호 작용을 이해하고자 하는 입자 물리 및 핵 물리 분야의 이론 및 시뮬레이션 도구이다[2]. QCD 이론은 입자 물리학의 이론 중 하나인 게이지 이론(gauge theories)의 환치 계산이 가능하고, 리군(Lie group)을 가지는 특성을 이용하여 입자 모델의 표본인 SU(Nc)와 전자석과 자계의 약한 상호작용에 관한 SU(2), SU(3)의 강한 상호작용인 양자 색역학을 나타낸다[3-4].

최근 GPU, DSP, FPGA 등 가속기의 성능이 지속적으로 개선됨에 따라 고성능 컴퓨팅을 통한 입자 물리학 분야의 시뮬레이터에 대한 병렬 처리 연구가 진행되고 있다[5]. 대표적인 성능 가속기로는 GPU 가 있고, GPU 를 범용으로 활용할 수 있는 기존의 병렬 처리 프레임워크로는 CUDA 가 있다. 그러나 CUDA 의 경우 NVIDIA 계열의 제품에서만 동작하는 단점이 있고, CPU 와 같은 마이크로프로세서 또한 지원을 하지 않는다. 이러한 단점을 해결한 것이 OpenCL[6]이다. OpenCL 은 이기종 컴퓨팅 환경에서 병렬처리를 할 수 있는 프레임워크이기 때문에, GPU 와 CPU 를 선택하여 병렬처리 할 수 있다. 본 논문에서는 이러한 OpenCL 의 특징을 이용하여, GPU 와 CPU 를 동시에 사용한 경우 QCD 의 수행시간을 측정하고, 처리기 하나만을 사용한 방법과 비교한다.

본 논문의 구성은 다음과 같다. 2 장에서는 이기종 컴퓨팅 환경에서의 프레임워크인 OpenCL 을 소개하고, QCD 병렬화를 위한 QCD 플랫폼에 대하여 설명한다. 3 장에서는 이기종 컴퓨팅 환경에서 OpenCL 의 특징을 이용하여 고성능 컴퓨팅을 필요로 하는 QCD 를 CPU 와 GPU 를 동시에 사용한 병렬처리 방법을 제안하고, 4 장에서는 실험을 통하여 제안된 방법의 성능을 확인한다. 마지막 5 장에서는 결론으로 논문을 마친다.

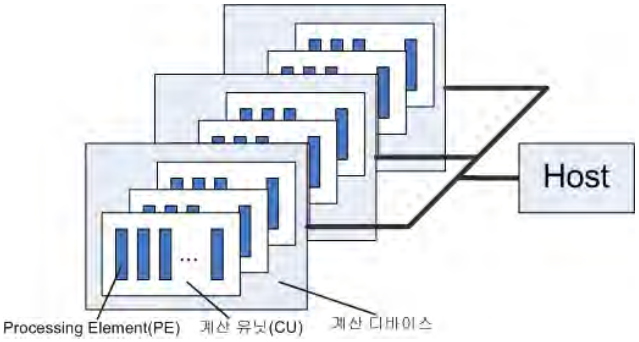
2. 관련 연구

2.1. OpenCL(Open Computing Language)

OpenCL 은 개방형 범용 병렬 컴퓨팅 프레임워크이다. 많은 제조사와 소프트웨어 전문가의 견해를 반영하여 Khronos 그룹에서 표준화시킨 OpenCL 은 여러 제조사들과의 협력을 통해 등장하였다. OpenCL 구현은 CPU, GPU 외에도 DSP 와 FPGA 에 기반한 플랫폼에서도 수행되기 때문에 하나의 프로그램을 작성하여 여러 종류의 하드웨어를 사용 할 수 있는 장점이 있다. 이것은 응용프로그램 수행 중에 OpenCL 커널 프로그램이 만들어지고 디바이스에 맞는 이진코드를 생성하기 때문이다. 또한, C99 를 기반으로 한 언어인 OpenCL C

로 커널 프로그램을 작성하기 때문에 프로그램을 작성하는데 있어서 C 언어와 흡사한 특징도 갖고 있다 [7-8]. OpenCL 플랫폼은 한 개의 호스트와 한 개 이상의 디바이스들로 구성되고, OpenCL 이 동작하기 위해서는 그림 1 과 같이 호스트에서 디바이스로 커널에서 연산된 데이터 메모리가 복사되어야 한다. 커널에서 연산이 끝난 후의 데이터 메모리는 역으로 디바이스에서 호스트로 복사되어야 한다. 또한, 여러 개의 GPU 를 사용하는 프로그램을 작성할 때, GPU 들에게 데이터 메모리를 복사하는 시간, 디바이스의 메모리 제약 등을 고려해야 한다.

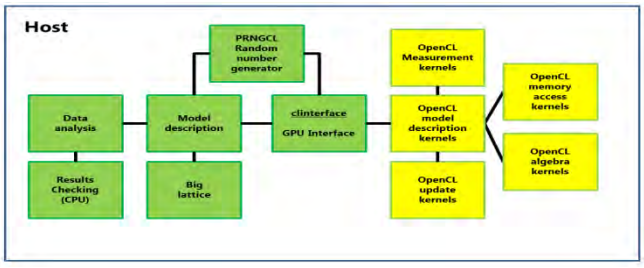
본 논문에서는 CPU 와 GPU 의 이기종 컴퓨팅을 통하여 데이터 메모리의 복사 시간을 고려함과 동시에 디바이스의 메모리 제약을 파악하여 수행시간을 줄이도록 작성하였다.



(그림 1) OpenCL 의 플랫폼 모델

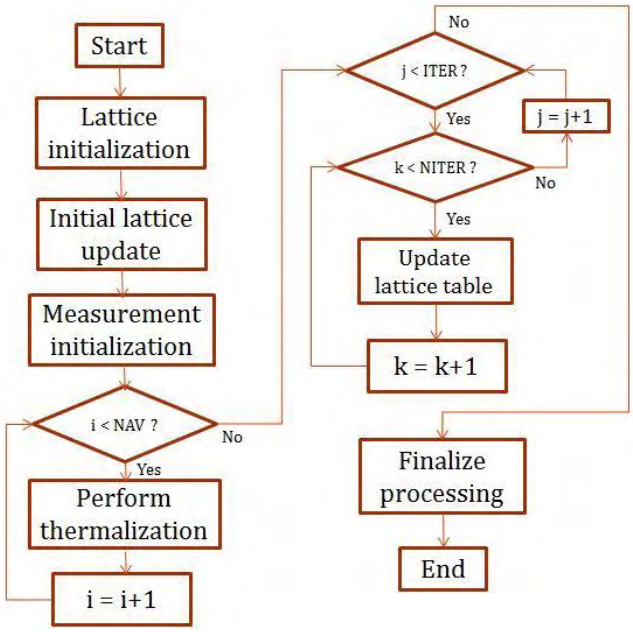
2.2. QCDGPU

QCDGPU 는 격자 QCD 의 대규모 물리학적 연산을 위해 GPU 를 이용하여 병렬처리를 수행하는 오픈 소스이다[5]. 먼저, 격자를 4 차원으로 만들어 각각 X, Y, Z, T 축으로 나누고 각 축의 크기를 정하여 격자의 크기를 계산한다. 초기의 격자에 존재하는 각각의 쿼크와 글루온의 물리학적 성질을 할당하기 위해 격자를 초기화 한다. 그 후, 쿼크와 글루온의 물리학적 연산을 위해 각각의 격자를 'sweep(update)'한다. 'sweep'하는 부분에 있어서, 이중 루프로 인한 대규모의 'sweep'을 하기 때문에 시간이 가장 오래 걸리는 부분이고 이 부분을 병렬 처리하였다. 그림 2 는 이 오픈 소스의 아키텍처를 도식화한 것이다.



(그림 2) QCDGPU 오픈 소스 아키텍처

이 오픈 소스의 데이터 플로우는 그림 3 과 같다. 격자 QCD 를 연산하기 위해, 격자와 격자 안에서의 쿼크와 글루온을 측정하기 위한 속성과 변수들을 초기화한다. 그 후, 열역학에 관한 연산을 처리하기 위해 'NAV'만큼의 단일 루프를 실행한다. 여기에서, 'NAV'는 처음의 측정을 수행하기 전에 격자의 'sweep'을 얼마나 많이 거쳐야 하는지 나타내주는 인덱스이다. 열역학에 관한 연산을 처리하고 나서 대규모의 연산을 위한 이중 구조의 루프를 수행한다. 'ITER'와 'NITER'는 대규모 연산을 수행하기 위한 인덱스이며 특히, 'NITER'는 다음의 측정을 하기 전에 얼마나 많이 'sweep'을 거쳐야 하는지를 의미하는 인덱스이다. 현재 QCDGPU 에 대한 오픈 소스가 배포되어 있으며, 본 논문에서는 이 오픈 소스를 이용하여 이기종 컴퓨팅 환경에서의 병렬처리가 가능하도록 수정하였다.



(그림 3) QCDGPU 오픈 소스 데이터 플로우

3. 이기종 컴퓨팅 환경에서의 격자 QCD

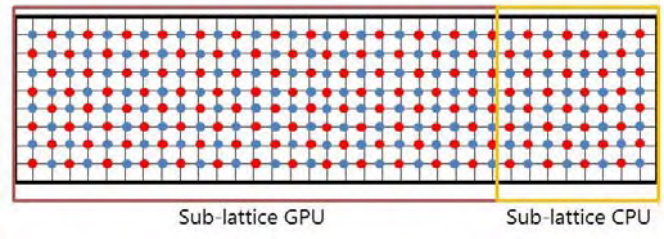
격자 QCD 는 대규모 연산 특성상 단순 부동소수점 연산이 많다. 따라서, 부동소수점에 유리한 GPU 로 병렬처리를 하는 것이 유리하다. 하지만, GPU 의 메모리 제약에 따라 격자의 크기가 GPU 의 메모리 가용범위를 넘을 경우 동작하지 않는 치명적인 단점이 있다.

이를 해결하기 위해 여러 개의 GPU 를 설치하고 격자 크기를 GPU 의 메모리 가용범위만큼 적절히 나누고 그것을 각각의 GPU 에 분배하는 방법이 있다. 이러한 방법은 위의 문제를 해결함과 동시에 좀 더 성능을 향상시킬 수 있지만, GPU 를 여러 개 구매해야 하기 때문에 비용이 올라가고, GPU 를 여러 개 설치할 수 없는 환경이라면 불가능한 방법이다. 또한, 많은 양의 데이터를 GPU 로 병렬처리를 할 경우 GPU 의 연산처리 시간 동안 CPU 는 유휴상태가 되어 자원의 낭비를 초래하게 된다. 본 논문에서는 CPU 의 유휴 자원의 낭비를 절감하고 병렬처리의 성능을 향상시키기 위해 GPU 뿐만 아니라 CPU 에도 격자의 크기를 분배하여 최적의 이기종 컴퓨팅 방법을 제안한다.

격자 QCD 이기종 컴퓨팅 환경에서의 병렬처리를 실

험하기 앞서, 격자 크기에 대한 GPU 의 메모리 제약을 알아내기 위해 Fedora release 17, Intel® Core™ i5-2500(4 cores, 4 threads), NVIDIA GeForce GTX 550(336 cores)의 환경에서 간단한 실험을 진행하였다. 먼저, GPU 에서 어느 정도의 격자 크기까지 실행할 수 있는지 실험을 하였는데, 각 X, Y, Z, T 축을 32 의 크기보다 크게 하였을 경우 GPU 의 메모리 제약 때문에 수행이 되지 않는 것을 확인하였다. 그에 반해, CPU 는 각 X, Y, Z, T 축을 36 의 크기보다 크게 하였을 경우 메모리 제약으로 수행되지 않았다. 따라서, 격자의 크기를 각 축에 대해 36^4으로 설정하여 실험하였다.

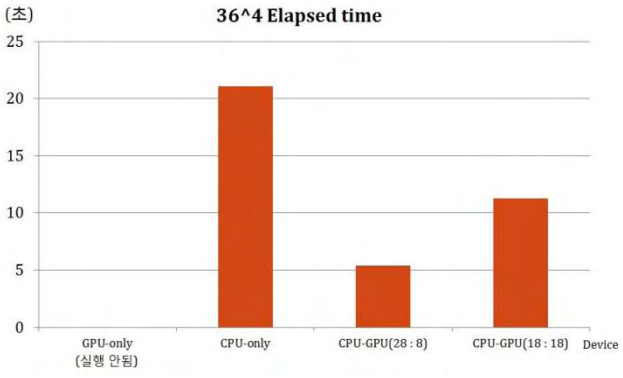
본 실험에 앞서, 첫 번째로 해야 하는 것은 최적의 격자 크기 분배이다. 본 실험에 사용한 오픈 소스는 멀티 디바이스를 사용하는 환경에서 'Big lattice'라는 모드로 전환이 된다. 이것은 멀티 디바이스에 각각 두 개의 부분 격자들로 나눠지면서 이 격자들의 boundary 계산을 위한 모드이다. 이 모드에서, 두 개의 부분 격자의 boundary 계산을 위해 부분 격자들로 나눠주는 기준 축에 각각 2 를 더해준다. 이러한 특성 때문에 GPU 와 CPU 의 메모리 제약을 고려하여 격자 크기를 분배하였다. 이 오픈 소스에서는 X 축을 기준으로 격자를 잘라 GPU 와 CPU 에 분배하였으며, 각 Y, Z, T 축이 36 의 크기일 때 X 축에 대한 GPU 의 한계는 30 의 크기를 넘지 못하기 때문에 'Big lattice' 모드의 특성을 고려하여 GPU 와 CPU 에 각각 28, 8 의 크기로 분배하였다. 그림 4 는 2 차원 기준으로 격자 테이블을 부분 격자들로 나타낸 것이다.



(그림 4) 부분 격자로 나타낸 격자 테이블

4. 실험 결과

본 실험은 한 격자의 GPU-only, CPU-only, 그리고 최적의 격자 크기로 분배한 CPU-GPU, 격자 크기를 절반의 크기로 분배한 CPU-GPU 환경에서 수행하였다. 그림 5 는 위의 환경에서 수행한 실험의 시간 측정 막대 그래프를 나타낸 것이다.



(그림 5) 다른 환경에서의 시간 측정

GPU-only 는 메모리 제약 때문에 격자의 크기가 36^4 일 때 수행되지 않아 시간측정을 할 수가 없었다. CPU-only 는 36^4인 격자의 크기에서 한 번의 'sweep' 당 약 21 초가 걸렸다. 격자 크기를 최적의 크기로 GPU 와 CPU 에 분배한 환경에서는 한 번의 'sweep'당 약 5 초가 걸렸고, 격자 크기를 절반의 크기로 GPU 와 CPU 에 분배한 환경에서는 약 11 초의 시간이 걸렸다. GPU-only 에서는 수행되지 않으므로 CPU-only 를 기준으로 성능향상을 비교하면, 최적의 분배량에 대한 실험은 약 3.9 배의 성능향상을 보였고, 절반의 분배량에 대한 실험은 약 1.9 배의 성능향상을 보였다.

5. 결론

격자 QCD 는 강한 상호 작용을 이해하고자 하는 입자 물리 및 핵 물리 분야의 이론이며, 이 이론은 쿼크와 글루온의 상호 작용에 따른 대규모 연산이 수반된다. 본 논문에서는 CPU 의 자원낭비를 절감하는 동시에 GPU 메모리 제약을 넘어 크기가 큰 격자 QCD 의 대규모 이기종 환경에서의 병렬처리 수행 및 성능향상을 제안하였다. 실험 결과, 모든 자원을 사용하는 환경에서 격자의 크기를 최적의 분배량으로 각각의 디바이스에 분배했을 경우 약 3.9 배, 절반의 분배량으로 분배했을 경우 약 1.9 배의 향상된 성능을 확인하였다.

감사의 글

이 논문은 BK 21 Plus와 정부(미래창조과학부)의 재원으로 한국연구재단 기초연구 실험데이터 글로벌 허브 구축사업(N-14-NM-IR06)의 지원을 받아 수행된 연구임.

참고문헌

[1] 신성식, 김아미, 김승완, 송주환, 권오봉, “Geant4 시뮬레이션 자동화를 위한 통합 프레임 워크 환경 개발 : Geant4Editor,” 전자공학회 논문지, 45 권 CI 편 제 4 호, pp. 12-18, 2008.

[2] 김세용, “Beowulf 클러스터를 이용한 격자 양자색소 역학 계산,” 정보과학회지 제 18 권 제 2 호, pp. 40-50, 2000.

[3] N. Cardoso and P. Bicudo, “Generating SU(Nc) pure gauge lattice QCD configurations on GPUs with CUDA,” Computer Physics Communications, Volume 184, Issue 3, pp. 509–518, 2013.

[4] M. Bach, V. Lindenstruth, O. Philipsen and C. Pinke, “Lattice QCD based on OpenCL,” Computer Physics Communications, Volume 184, Issue 9, pp. 2042–2052, 2013.

[5] V. Demchik and N. Kolomoyets, “QCDGPU: open-source package for Monte Carlo lattice simulations on OpenCL-compatible multi-GPU systems,” arXiv preprint arXiv:1310.7087, 2013.

[6] J. Stone, D. Gohara, G. Shi, “OpenCL: a parallel programming standard for heterogeneous computing systems,” Comput. Sci. Eng., pp. 66–73, 2010.

[7] 김희곤, 이성주, 최동휘, 정용화, 박대희, “이기종 컴퓨팅 환경에서 대용량 이미지 처리,” 한국멀티미디어학회 춘계학술발표대회 논문집 제 17 권 1 호, pp. 326-328, 2014.

[8] 김희곤, 이성주, 정용화, 박대희, “이기종 컴퓨팅 환경에서 OpenCL 을 이용한 효율적인 병렬처리,” 제 40 회 한국정보처리학회 추계학술발표대회 논문집, 제 20 권 2 호, pp. 111-114, 2013.