KOKKOS 환경에서의 병렬 숄레스키 분해 구현

강준범 [1], 이명호 [2], 박능수 [1]
[1] 건국대학교 컴퓨터공학과
[2] 명지대학교 컴퓨터공학과

aopko@konkuk.ac.kr, myungho@mju.ac.kr, neungsoo@konkuk.ac.kr

Implementation of Parallel Cholesky Decomposition in the Kokkos environment

Junbeom Kang[1], Myungho Lee[2], Neungsoo Park[3]
[1]Dept. of Computer Science and Engineering, Konkuk University
[2]Dept. of Computer Science and Engineering, Myongji University

요 약

최근 병렬컴퓨팅 연구는 슈퍼컴퓨터의 성능 향상에 직접 큰 영향을 끼치는 고성능 GPU를 활용한 대형 데이터셋 고속 병렬화를 중점적으로 진행되고 있다. 이를 해결하기 위해 Sandia 연구소에서 개발한 Kokkos 프로그래밍 모델이 등장했다. 이 논문에서는 CUDA 기반의 병렬 숄레스키 분해 구현을 해당 환경에 이식했을 때 어떤 성능을 보이는지 실험을 통해 확인했다. 1000x1000 크기의 양의 정부호 에르미트 행렬에 대해서 직렬 숄레스키 분해 프로그램 대비 498.16 배의 성능 향상을 보였으며, 이를 통해 자동으로 메모리를 관리하는 Kokkos 프로그래밍 모델이 추후 대형 데이터셋을 대상으로 하는 병렬화 프로그램 구현 시, 더욱 편리하고 좋은 성능 향상을 보일 것임을 기대한다.

1. 서론

최근 병렬컴퓨팅 연구는 슈퍼컴퓨터의 수요가 증가하게 되고, 해당 기술의 중요성이 부각되기 시작하며 상당히 매력적인 주제로 선택되어지고 있다. 좋은 슈퍼컴퓨팅 능력을 보이기 위해선 다양한 부분에서 환경 최적화를 진행해야 하며, 대표적인 최적화 대상은 효율적인 메모리 구조 설계와 통신 구조 설계, 연산을 수행하는 GPU 내 파라미터 조정이다 [1].

이 때, 다양한 모델에서도 일관적인 최적화를 위해 Sandia 연구소에서는 Kokkos 프로그래밍 모델[2]을 개발했다. 따라서 본 논문에서는 Kokkos-CUDA 를 활용한 숄레스키 분해 코드를 구현하여 직렬 숄레스키 분해와 실행 시간을 비교해 병렬화를 통한 실행 시간 가속화 정도를 계산하여 모델의 실용성을 평가한다.

2. 연구배경

2.1 Kokkos 프로그래밍 모델

Kokkos 프로그래밍 모델(Kokkos)는 Sandia 연구소에서 개발한 멀티/매니코어 기반의 프로그래밍 모델로, 멀티코어 프로세서와 매니코어 프로세서 그리고 General-purpose graphics processor unit(GPGPU) 프로세서를 대상으로 C++ 기반의 API 를 제공한다. OpenMP, OpenACC, OpenCL, CUDA 와 같은 다양한 프로그래밍 모델에 대한 컴퓨팅 및 데이터 할당/메모리 레이아웃에 대한 추상화를 제공하며 이를 기반으로 이기종 환경에서 이식성 및 고성능을 동시에 유도했다[3]. Kokkos 의 핵심 요소는 Kokkos Core, Kokkos Kernels Math Libraries 와 Kokkos Profiling, Debugging Tools 로 구성되어 있으며 각 요소는 멀티/매니코어 디바이스 상에서의 공유 메모리 관리와 HPC 상에서의 커널 소프트웨어 라이브러리 제공 및 프로파일링/디버깅 툴로써 역할을 수행한다.

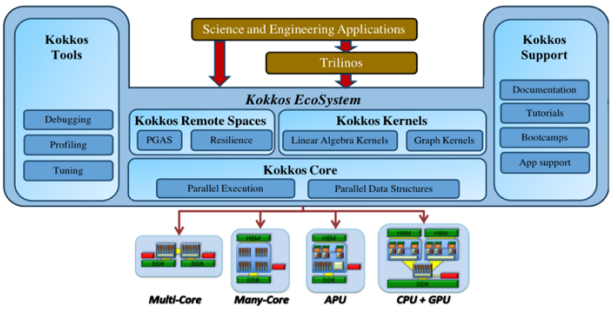

Figure 1 Kokkos 프로그래밍 모델[4]

2.2 Kokkos Views

Kokkos 에서는 특별히 사용자가 메모리를 명시적으로 관리하는 부담을 줄이기 위해 자체적으로 View 라는 추상화를 제공한다. 이는 다차원 배열에 대해 편리한 메모리 할당과 관리, 병렬처리를 위한 데이터 통신 최적화를 기대할 수 있다.

2.3 숄레스키 분해

숄레스키 분해는 양의 정부호 에르미트 행렬을 대상으로 하는 선형대수 연산을 의미한다. 숄레스키 분해는 위 행렬을 수식 1 의 (1)처럼 하삼각행렬과 하삼각행렬의 전치행렬의 곱으로 나타내도록 한다. 이는 수식 1 의 (2)처럼 상삼각행렬에도 적용할 수 있다.

$$A = LL^* \quad \dots(1)$$
$$A = U^*U \quad \dots(2)$$

Equation 1 숄레스키 분해 공식

수식 1 의 (2)로 분해하게 되면 각 원소는 대각원소들은 수식 2 의 (3)과 같이, 나머지 원소들은 수식 2 의 (4)처럼 계산된다[5].

$$u_{ii} = \sqrt{a_{ii} - \sum_{k=1}^{i-1} u_{ki}^* u_{ki}} \quad \dots(3)$$
$$u_{ij} = \frac{1}{u_{ii}}\left(a_{ij} - \sum_{k=1}^{i-1} u_{kj}^* u_{kj}\right) \quad \dots(4)$$

Equation 2 분해 시 원소 별 계산식

이처럼 다양한 원소에 지속적으로 메모리에 접근해야 하는 분해 연산의 알고리즘 특성 상 병렬처리를 구현할 때 성능 향상이 나타날 것이라고 판단하여 숄레스키 알고리즘 프로그램을 실험에 활용했다.

3. 실험

성능을 비교하기 위한 기본적인 직렬 숄레스키 분해 코드와 Kokkos-CUDA 기반 병렬 숄레스키 분해 코드를 구현했고, 동일한 1000*1000 크기의 양의 정부호 에르미트 행렬을 직접 생성해 각 프로그램별 커널 실행시간을 비교하여 성능 향상을 분석하였다. 실험 환경은 표 1 과 동일하다.

CPU	AMD Ryzen 7 7800x3D
GPU	NVIDIA RTX 4070ti
OS	Windows
Data size (2D Matrix Size)	1000*1000 2-dimensional array

Table 1 실험 환경

3.1 실험 방식

직렬 숄레스키 분해 코드와 병렬 숄레스키 분해 코드로 작성된 프로그램을 각각 100 회 실행 후 평균 커널 실행 시간을 합산해 산술 평균을 계산하여 가속화 수준을 측정했다.

3.2 실험 결과

	Serial	Kokkos-CUDA
Time(ms)	15.459	0.031
Speedup(Ts/Tp)	1	498.68

Table 2 실험 결과

표 2 와 같이 Kokkos-CUDA 상에서 기존 $O(n^3)$의 연산을 수행해야 하는 직렬 숄레스키 연산을 단순하게 적절한 블록 단위로 연산을 GPU 의 코어에 할당하는 병렬화를 진행했음에도 불구하고, Kokkos View 를 통해 진행된 자체적인 메모리 관리를 통해 직렬 프로그램 대비 약 500 배에 가까운 연산 가속화를 보이는 것을 확인할 수 있었다.

4. 결론

본 연구의 목적은 직렬 숄레스키 알고리즘을 Kokkos 환경에서 구현했을 때 가속화 수준을 통해 프로그래밍 모델을 평가하기 위함이었다. 실험을 통해 사용자가 명시적으로 메모리 관리를 진행하지 않음에도 불구하고 좋은 가속화 수준을 확인했고, 이를 통해 Kokkos 프로그래밍 모델에 Kokkos Views 메모리 추상화의 강점과 CUDA 코드 이식성이 높음을 확인할 수 있었다.

참고문헌

[1] Jack Dongarra et al, "The International Exascale Software Project Roadmap 1", The International Journal of High Performance Computing Applications 25, p3-60, 2011

[2] “Kokkos”, (Aug 01, 2024), https://kokkos.org

[3] 권오경, “Kokkos 프로그래밍 모델”, 국가슈퍼컴퓨팅연구소, Nov. 2015

[4] “The Kokkos EcoSystem”, (Jul 26, 2020), Kokkos tutorial https://github.com/kokkos/kokkos-tutorials/blob/main/Intro-Short/KokkosTutorial_Short.pdf

[5] Aravindh Krishnamoorthy, Deepak Menon, “Matrix Inversion Using Cholesky Decomposition”, [Signal Processing Algorithms, Architectures, Arrangements and Applications(SPA)], Poland, 2013, p70-72

감사의 글

본 연구는 과학기술정보통신부의 재원으로 한국연구 재단의 지원 사업(RS-2023-00321688)과 정보통신기획 평가원의 정보통신방송혁신인재양성(메타버스융합대학원)사업(IITP-2024-RS-2023-00256615)의 연구 결과로 수행되었음