Sustainable Smart Media Technology
and Innovation

2023
스마트미디어 심포지움

2023 Smart Media Symposium

일시: 2023년 10월 26일(목) ~ 10월28일(토)
장소: 순천대학교 70주년기념관&산학협력관
주최: (사)한국스마트미디어학회/(사)한국전자거래학회
후원: 순천시

본 사업은 기획재정부의 복권기금 및 과학기술정보통신부의 과학기술진흥기금으로
추진되어 사회적 가치 실현과 국가 과학기술 발전에 기여합니다.

GPU 내 CUDA 프로그램의 저전력 코드 패턴 식별

김현태*, 진예진**, 김영철***
홍익대학교 소프트웨어공학 연구실
e-mail : {hyuntaekim*, yejin_jin**}@g.hongik.ac.kr, bob@hongik.ac.kr***

Code Pattern Identification for Low Power Consumption of CUDA Program on GPU

Hyun-Tae Kim, Ye-Jin Jin, R. Young Chul Kim
SE Lab, Dept. of Software and Communications Engineering, Hongik University

요 약

기존에는 오픈 소스와 같이 AI SW 알고리즘을 사용하고 있다. 앞으로는 한국형 AI 반도체에 한국형 AI 소프트웨어 알고리즘 개발이 필요하다. 또한 현재 AI 관련 소프트웨어 및 학습 모델들은 전력 소비에 대해 문제 삼지 않고 있다. 이를 해결하기 위해 실제 GPU 구조 내에서 CUDA 프로그램 수행 메카니즘 및 절차에 대한 이해가 필요하다. 또한 GPU 내 저전력 코드로 전력 소비에 대한 연구가 필요하다. 본 연구에서는 CUDA 프로그래밍의 저전력 코드를 식별하고 정의한다. 이를 통해 AI 소프트웨어의 전력 소비를 줄일 수 있기를 기대한다.

1. 서 론

최근 GPU 기술이 발전함에 따라 GPU의 기능이 비약적으로 상승하며 AI의 관심과 사용률이 증가하고 있다. 그러나, 고성능 컴퓨팅으로 인해 전력 소모량 증가에 대한 문제들이 언급되고 있다[1]. 기존 연구[2]에서는 AI의 전력 문제를 해결하고, AI 소프트웨어의 전력 관점의 품질을 높일 수 있도록 효율적인 코드 패턴을 분석하였다. Code Building Block 중 Loop문의 전력을 측정하여 저전력 코드 패턴을 정의하였다.

본 연구에서는 기존 연구를 바탕으로 Code Building Block 중 Statement문과 Branch문의 전력 소비량을 측정한다. 이를 통해 CUDA에서의 저전력 코드 패턴을 정의한다.

2. 저전력 코드 패턴 연구

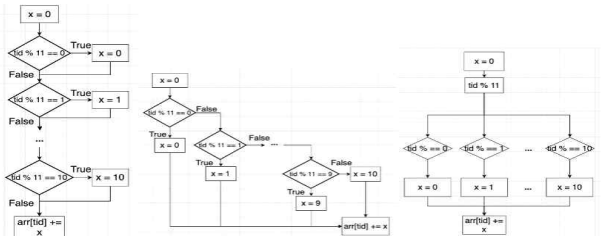
기존 C언어에서의 저전력 연구[3]에서는 Code Building Block 중 반복문과 함수 호출 방법 등을 측정 및 연구하였다. 측정한 결과는 Call by Value 방식과 If-then-else문이 전력 소모가 심하다. 이때, 전력 소모가 높은 코드를 Bad Energy Pattern을 정의하였다. 전력을 측정한 도구는 Nvidia-smi와 Intel Power Gadget을 사용하였다.

3. 본 연구

본 연구에서는 기존 연구[2]와 동일한 조건으로 Statement문과 Branch문의 전력을 측정한다.

3.1 Branch문

Branch문에서는 If, If-then-Else, Switch문의 저전력 코드 패턴을 분석하였다. 그림 1은 본 연구에서 측정한 CUDA의 커널 함수 Flow Chart이다.



(그림 1) Branch CUDA 커널 함수 Flow Chart

측정 코드는 1부터 1억까지의 수를 11로 나누었을 때의 나머지를 구하는 코드이다. 표 1은 Branch문 테스트 코드의 커널 함수와 전력 측정 결과이다.

(표 1) Branch문 테스트 코드와 전력 측정 결과

전력 단위 [Wh]	IF	IF-Then-Else	Switch
Kernel Function Code	__global__ void fn1(int* arr) { int tid = blockIdx.x * blockDim.x + threadIdx.x; int x = 0; if (tid % 11 == 0) { x = 0; } if (tid % 11 == 1) { x = 1; } … if (tid % 11 == 10) { x = 10; } arr[tid] += x; }	__global__ void fn1(int* arr) { int tid = blockIdx.x * blockDim.x + threadIdx.x; int x = 0; if (tid % 11 == 0) { x = 0; } else if (tid % 11 == 1) { x = 1; } … else { x = 10; } arr[tid] += x; }	__global__ void fn1(int* arr) { int tid = blockIdx.x * blockDim.x + threadIdx.x; int x = 0; switch (tid % 11) { case 0: x = 0; break; case 1: x = 1; break; … case 10 : x = 10; } arr[tid] += x; }
CPU	0.00192	0.00236	0.00250
GPU	0.00153	0.00102	0.00138
소모 전력	0.00344	0.00339	0.00363

CPU의 전력 소모량이 높은 패턴은 Switch문이고, GPU의 경우 If문이다. CPU와 GPU의 전력을 모두 합친 소모 전력이 가장 높은 패턴은 Switch문이다. Switch문을 Bad Energy Pattern으로 정의하고, If-Then-Else문을 저전력 소비 코드 패턴으로 정의한다. 해당 코드에 절차식 언어에서 사용하는 Cyclomatic 복잡도를 적용하면, If의 복잡도는 12, If-Then-Else문은 11, Switch문은 12이다. 저전력 소비 코드 패턴인 If-Then-Else문의 Cyclomatic 복잡도가 가장 낮은 것을 확인할 수 있다. 그러나, CUDA는 기존

CPU 프로그래밍 코드와 다르게 동작하기 때문에 코드의 복잡도나 품질을 계산하는 방법을 개선할 필요가 있다.

3.2 Statement문

Statement문은 3가지 유형으로 나누어 측정하였다. 각각 A, B, C 유형으로 나누어 유형별 저전력 소비 코드 패턴을 정의한다. 표 2는 A 유형에서 사용한 테스트 코드의 커널 함수와 전력 측정 결과이다.

(표 2) A유형 Statement문 테스트 코드와 전력 측정 결과

전력 단위 [Wh]	선언, 연산 반복	전체 선언 후, 연산

Kernel Function Code	global__ void a2(int* arr) {
	int tid = blockIdx.x * blockDim.x + threadIdx.x;
	int x = arr[tid];
	int sum = 0;
	int a = x + 1;
	sum += a;
	int b = x + 2;
	sum += b;
	…
	int j = x + 10;
	sum += j;
	arr[tid] = sum;
	}
		_global__ void a2(int* arr) {
		int tid = blockIdx.x * blockDim.x + threadIdx.x;
		int x = arr[tid];
		int sum = 0;
		int a = x + 1;
		int b = x + 2;
		…
		int j = x + 10;
		sum += a;
		sum += b;
		…
		sum += j;
		arr[tid] = sum;
		}

CPU	11.28450	12.53610
GPU	5.43082	6.04630
소모 전력	16.71483	18.58240

표 2의 좌측 코드는 선언과 연산을 반복적으로 하는 코드이고, 우측 코드는 선언 후 연산을 진행하는 코드이다. GPU와 CPU를 합한 결과를 바탕으로 저전력 코드 패턴은 선언, 연산 반복 패턴이다. 표 3은 B 유형에서 사용한 테스트 코드의 커널 함수와 전력 측정 결과이다.

(표 3) B유형 Statement문 테스트 코드와 전력 측정 결과

	x=x+1	x+=1	x++	++x

Kernel Function Code	_global__ void fn(int* arr) {
	int tid = blockIdx.x * blockDim.x + threadIdx.x;
	int x = arr[tid];
	x = x + 1;
	x = x + 1;
	x = x + 1;
	x = x + 1;
	x = x + 1;
	x = x + 1;
	arr[tid] = x;
	}
		_global__ void fn(int* arr) {
		int tid = blockIdx.x * blockDim.x + threadIdx.x;
		int x = arr[tid];
		x += 1;
		x += 1;
		x += 1;
		x += 1;
		x += 1;
		x += 1;
		arr[tid] = x;
		}
			_global__ void fn(int* arr) {
			int tid = blockIdx.x * blockDim.x + threadIdx.x;
			int x = arr[tid];
			x++;
			x++;
			x++;
			x++;
			x++;
			x++;
			arr[tid] = x;
			}
				_global__ void fn(int* arr) {
				int tid = blockIdx.x * blockDim.x + threadIdx.x;
				int x = arr[tid];
				++x;
				++x;
				++x;
				++x;
				++x;
				++x;
				arr[tid] = x;
				}

CPU [Wh]	19.023057	18.520483	20.270457	18.470592
GPU [Wh]	8.9813402	8.4189692	8.6354106	9.0097644
소모전력 [Wh]	28.004397	26.939452	28.905868	27.480356

표 3의 GPU와 CPU의 소모 전력을 합한 결과로부터 x++ 패턴을 Bad Energy Pattern으로 정의하고, x+=1 패턴을 저전력 코드 패턴으로 정의한다. 표 4는 C 유형에서 사용한 테스트 코드의 커널 함수와 전력 측정 결과이다.

(표 4) C유형 Statement문 테스트 코드와 전력 측정 결과

전력 단위 [Wh]	감소 덧셈	증가 덧셈

Kernel Function Code	global__ void a2(int* arr) {
	int tid = blockIdx.x * blockDim.x + threadIdx.x;
	int x = arr[tid];
	int sum = 0;
	int a = x + 1;
	sum += a;
	int b = x + 2;
	sum += b;
	…
	int j = x + 10;
	sum += j;
	arr[tid] = sum;
	}
		_global__ void a2(int* arr) {
		int tid = blockIdx.x * blockDim.x + threadIdx.x;
		int x = arr[tid];
		int sum = 0;
		int a = x + 1;
		int b = x + 2;
		…
		int j = x + 10;
		sum += a;
		sum += b;
		…
		sum += j;
		arr[tid] = sum;
		}

CPU	11.28450	12.53610
GPU	5.43082	6.04630
소모 전력	16.71483	18.58240

표 4의 GPU의 소모 전력을 비교하면 GPU의 소모 전력이 가장 많은 패턴은 감소 덧셈 패턴이고 CPU에서 소모 전력이 가장 많은 패턴은 증가 덧셈 패턴이다. GPU와 CPU의 소모 전력을 합한 결과는 감소 덧셈 패턴이 더 높게 측정되어 Bad Energy Pattern으로 정의하고, 증가 덧셈 패턴을 저전력 코드 패턴으로 정의한다.

3.3 Loop문

CUDA의 저전력 연구[2]에서는 Code Building Block 중 Loop문의 소모 전력을 통해 저전력 코드 패턴을 정의하였다. 표 5는 Loop문의 커널 함수와 전력 측정 결과이다.

(표 5) Loop문 테스트 코드와 전력 측정 결과

	인수 증가 For	인수 감소 For	While	Do-While

Kernel Function Code	__global__ void fn1(int* arr) {
	int tid = blockIdx.x * blockDim.x + threadIdx.x;
	int x = tid;
	for (int i = 0; i < 100; i++) {
	x *= x;
	}
	arr[tid] += x;
	}
		__global__ void fn1(int* arr) {
		int tid = blockIdx.x * blockDim.x + threadIdx.x;
		int x = tid;
		for (int i = 100; i > 0; i--) {
		x *= x;
		}
		arr[tid] += x;
		}
			__global__ void fn1(int* arr) {
			int tid = blockIdx.x * blockDim.x + threadIdx.x;
			int x = tid;
			int i = 1;
			while (i <= 100) {
			x *= x;
			i++;
			}
			}
				__global__ void fn1(int* arr) {
				int tid = blockIdx.x * blockDim.x + threadIdx.x;
				int x = tid;
				int i = 1;
				do {
				x *= x;
				i++;
				}
				while (i <= 100);
				arr[tid] += x;
				}

CPU	0.00320	0.00261	0.00275	0.00250
GPU	0.00125	0.00118	0.00117	0.00138
소모 전력	0.00445	0.00380	0.00391	0.00388

Bad Energy Pattern은 인수 증가 for문, 저전력 소비 코드 패턴은 인수 감소 for문으로 정의하였다. Bad Energy Pattern을 저전력 소비 코드 패턴으로 개선할 때, 14.6% 감소 효과를 보였다.

4. 결론 및 향후 연구

CUDA 프로그래밍에서의 Branch문과 Statement문의 저전력 소비 코드 패턴과 Bad Energy 패턴을 정의하였다. Branch문에서는 Bad Energy Pattern을 저전력 코드 패턴으로 개선 시 6.6%의 전력 소모를 감소할 수 있다. Statement A유형의 경우, 선언 후 연산 패턴을 선언, 연산 반복 패턴으로 개선 하면 11%의 전력 개선 효과를 기대할 수 있다. B유형에서 Bad Energy 패턴을 저전력 소비 패턴으로 개선 시 7.3%의 전력 감소 효과를 기대할 수 있다. C유형에서 Bad Energy Pattern을 저전력 에너지 패턴으로 개선 시 전력 소모값은 0.3%로 세가지 유형 중 가장 소모 값이 적다.

향후 연구에서는 Code Building Block 중 본 연구에서 진행하지 않은 부분을 연구할 것이다. 또한 CUDA에서 복잡도를 계산하기 위해 Cyclomatic 외의 복잡도를 연구할 것이다.

ACKNOWLEDGMENT

본 연구는 2023년도 문화체육관광부의 재원으로 한국콘텐츠진흥원(과제명: 인공지능 기반 사용자 대화형 멀티모달인터랙티브 스토리텔링 3D장면 저작 기술 개발, 과제번호: RS-2023-00227917,기여율:50%) 지원과 2023년도 정부(교육부)의 재원으로 한국연구재단 기초연구사업(과제명: NLP BERT Model 기반 자동 리팩토링을 통한 무결점 코드화 연구, 과제번호: No.2021R1I1A3050407,기여율:50%)의 지원을 받아 수행된 연구임.

참 고 문 헌

[1] Huang, Yanhui, Bing Guo, and Yan Shen. GPU energy consumption optimization with a global-based neural network method, IEEE Access 7, 2019.

[2] 진예진, 김현태, 김영철, “CUDA 프로그래밍의 코드 빌딩 블록에서 저전력 소비 코드 패턴 식별,” ICT 플랫폼, vol. 10, no. 1, pp. 63-66, July 2023.

[3] 박경문, 김병서, 김영철, “초소형 IoT 디바이스의 저전력 최적운용 기술 분석 및 알고리즘 요소기술 연구”, 한국전자통신연구원 보고서, 2018.