NIVIDA Bluefield-3 DPA 상에서의 Allreduce 통신 설계

명훈주, 곽재혁, 정요상, 정기문
한국과학기술정보연구원 슈퍼컴퓨팅기술개발센터
hjmyung@kisti.re.kr, jhkwak@kisti.re.kr,
yosang.jeong@kisti.re.kr, kmjeong@kisti.re.kr

Design of Allreduce communication on NVIDIA Bluefield DPA

Hunjoo Myung, Jae-Hyuck Kwak, Yosang Jeong, Kimoon Jeong
Korea Institute of Science and Technology Information
Center for Supercomputing Technology and Development

요 약

본 논문은 고성능 컴퓨팅 환경에서 NVIDIA BlueField-3의 DPA를 활용한 MPI All-Reduce 통신 가속 방안을 제시한다. 본 연구에서는 BlueField-3의 DPA는 256개 스레드 기반의 강력한 병렬 처리 능력을 극대화하기 위해, 주요 All-Reduce 알고리즘의 DPA 환경 적용의 적합성을 분석했다. 분석 결과, Tree All-Reduce가 확장성과 DPA의 병렬성 활용이라는 두 가지 요구사항을 가장 효과적으로 충족하였다. 그리고, Tree All-Reduce의 Reduce 및 Scatter 단계에서 DPA 스레드가 다수 노드와의 통신을 병렬로 처리하도록 설계된 코드를 제시한다. 본 연구는 DPU의 병렬 가속 능력을 활용하여 분산 통신 지연 시간을 최소화하고 전체 분산 처리 성능을 극대화하는 실질적인 방법론을 제공한다.

1. 서론

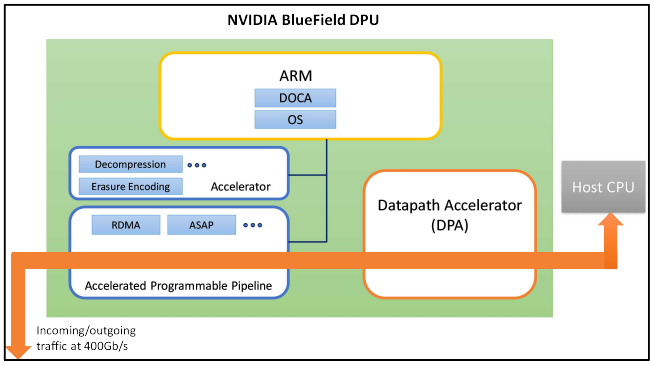
현대데이터센터의 인프라는 클라우드 서비스와 AI 워크로드 및 HPC 컴퓨팅 수요의 폭발적인 증가에 따라 급격히 복잡해지고 있다. 이러한 변화는 네트워크 가상화, 보안, 스토리지 기능 등 인프라 처리 부하를 호스트 CPU에 집중시키는 결과를 낳았다. 이러한 비효율성을 해소하고 서버 자원의 활용도를 극대화하기 위해, 이러한 오버헤드를 CPU로부터 분리하여 전용 가속기에서 처리하고자 하는 필요성이 대두되었다.

이러한 배경에서 SmartNIC 및 DPU (Data Processing Unit)이 출현하였다. 초기에는 NIC에 프로그래밍 가능한 기능을 추가하여 네트워킹 및 보안 인프라를 CPU로부터 오프로드하는 수준이었다[1]. 이후 NVIDIA BlueField와 같은 최신 DPU는 ARM CPU, 메모리, 다양한 가속 엔진을 장착하여, 단순한 네트워킹 가속을 넘어 인프라 컴퓨팅 전반의 범용 처리까지 담당하는 수준에 이르렀다[2].

본 논문에서는 이러한 DPU의 장점을 활용하여 MPI 집합통신의 성능을 극대화하는 코드를 설계한다. 2장에서는 본 논문에서 활용할 NVIDIA BlueField-3의 데이터경로 가속기(DPA: Data-Path Accelerator)에 대해 소개한다. 3장에서는 DPA 상에서 Allreduce MPI 통신 설계에 대해서 소개하고, 4장에서 결론을 맺는다.

2. 엔비디아 블루필드-3의 DPA 소개

DPA는 Bluefield-3 DPU에서 처음으로 도입된 16코어, 256 쓰레드 기반의 가속기이다[3]. 이 가속기는 NIC과 호스트 CPU 사이의 데이터경로에 위치하여 호스트 CPU의 개입 없이도 데이터경로 작업을 가속화하도록 설계되었다. DPA는 데이터경로를 통과하는 어떤 패킷이든 송수신할 수 있으며, DPU에 내장된 모든 가속기(암호화/복호화, 해시연산, 압축/압축해제)등을 활용할 수 있다[4]. 또한, DPA는 병렬처리를 위한 다수의 실행 유닛을 가지고 있어 지연시간을 최소화하고, 트래픽을 처리하는 총량을 크게 증가시킨다. 이러한 DPA의 특징으로 네트워크 패킷 처리이외에도 여러 분야에서 이를 활용하여 CPU의 부하를 줄이고, 가속화하려는 여러 연구가 시도되고 있다.



<그림 1>NVIDIA BlueField-3 DPU 아키텍처

3. DPA 상에서 Allreduce MPI 통신 설계

이 장에서는 MPI 집합통신을 DPA에 오프로딩하는 코드 설계과정을 기술한다. DPA는 앞서 설명한 대로 최대 256개의 쓰레드를 기반으로 병렬처리가 가능한 가속기이다. DPA의 병렬처리 장점을 극대화 극대화하기 위해 통신부하가 큰 집합통신을 주요 설계대상으로 삼았다.

3.1 주요 Allreduce 알고리즘 소개

(1) centralized Allreduce

이 알고리즘은 가장 단순한 형태로 논리적으로 star 위상에서 동작하며, 각 노드의 데이터를 한 노드로 수집하여(gather) 합산한 뒤, 이 결과를 모든 노드들에게 브로드캐스트한다.

(2) Ring Allreduce

이 알고리즘은 프로세스들이 논리적인 링 위상에서 작동하며, 각 프로세스는 자신의 이웃 프로세스와만 데이터를 주고 받는다. 이 알고리즘은 통신 효율을 극대화하기 위해 전체 데이터를 프로세스 수로 나눈 동일한 크기의 청크 단위로 처리하며, 크게 Reduce-scatter와 Allgather 두 단계로 구성된다. 첫 번째 단계에서는 각 프로세스가 전체 합산 결과 중 자신이 담당하는 청크를 완성하는 것이다. 즉, 자신의 청크와 이웃에서 받은 청크를 합산하여 이것을 링을 따라 다음 프로세스로 보낸다. 이것을 N번을 반복하면, 각 프로세스는 최종적으로 자신이 담당하는 합산된 청크를 확보한다. 두 번째 단계에서는 합산된 청크를 모든 프로세스에게 공유하여 전체 결과를 확보하는 것이다. 즉, 각 프로세스는 자신의 합산된 청크와 이웃에서 받은 청크를 링을 따라 N-1번 전달하면 모든 프로세스들이 합산된 전체 데이터를 가지게 된다.

(3) Recursive Doubling Allreduce

이 알고리즘은 논리적으로 하이퍼큐브 또는 버터플라이 위상에서 작동하며, 이것의 핵심은 각 통신 단계에서 데이터 교환 및 합산 범위를 재귀적으로 두 배씩 확장하는 것이다. k 단계에서 프로세스 i는 프로세스 ($i \oplus 2^{k-1}$)을 통신 파트너로 정해, 서로의 데이터를 주고 받아 합산한다. 총 log N (N은 총 프로세스 개수) 단계를 진행하면, 모든 프로세스는 최종 합산 결과를 얻게 된다.

(4) Tree Allreduce

이 알고리즘은 프로세스들이 논리적으로 이진 트리와 같은 계층적 구조로 연결된 기반에서 작동하며, 각 프로세스는 부모노드와 자식노드들과만 데이터를 주고 받는다. 리프(leaf) 노드에서 루트 노드 방향으로 데이터를 합치며 올라가 최종적으로 합계가 루트 노드에 모이게 된다. 루트 노드에서 계산된 최종 결과를 다시 리프 방향으로 브로드캐스트하여 Allreduce를 완성한다.

<표 1>주요 allreduce 알고리즘의 특징

알고리즘	위상	통신 단계	확장성	트래픽분산
centralized	star	2(P-1)	낮음	편중
Ring	ring	2(P-1)	높음	균형
Recursive Doubling	hypercube	$2\log_2 P$	중/높음	비교적균등
Tree	tree	$2\log_k P$ k=트리가지수	높음	트리구조로 균등분산

3.2 DPA 환경에서의 Allreduce 알고리즘 적용방안

본 연구에서는 DPA의 병렬처리 능력을 활용하여 All-Reduce 통신 성능을 극대화할 수 있는 방안을 모색하였다. DPA의 다중 쓰레드를 활용한 동시 처리 능력에 기반하여 주요 All-Reduce 알고리즘들의 적합성을 분석하였다.

Star 위상을 갖는 중앙 집중식 Allreduce 알고리즘은 Gather와 Broadcast 단계에서 DPA의 쓰레드를 활용하여 다수 통신을 동시에 처리하기에 적합하다. 그러나 이 알고리즘은 중앙 프로세스에 트래픽이 집중되어 통신 병목현상을 유발하며, 확장성이 낮아 대규모 시스템에서는 널리 사용되지 않는 단점이 있다.

Ring-Allreduce와 Doubling Recursive Allreduce는 트래픽이 균등하고 확장성이 높은 분산형 알고리즘이다. 하지만 이들은 각 통신 단계에서 프로세스당 동시에 처리되는 통신 수가 극히 제한적이다.(O(1)). 따라서 DPA의 다중 쓰레드 병렬처리 이

점을 충분히 활용하기 어려워 DPA 환경에 적용하기에는 적합하지 않다고 판단하였다.

tree Allreduce 알고리즘은 비교적 확장성을 유지하면서, 트리의 가지 수(k)를 조절함으로써 각 단계에서 DPA가 동시에 처리할 수 있는 통신 수를 증가시킬 수 있다.

검토된 네 가지 Allreduce 알고리즘 중 Tree allreduce가 확장성과 DPA의 병렬처리능력 활용이라는 두가지 요소를 가장 효과적으로 충족시킨다. 이에 따라, 본 연구는 DPA 환경에서 Allreduce 통신을 수행하기 위한 가장 적합한 알고리즘으로 tree Allreduce를 최종 선정하였다.

3.3.DPA 환경에서의 tree Allreduce 알고리즘 구현

<그림 2>은 tree 위상에서 각 노드에서 처리해야 할 allreduce 통신의 핵심부분을 DPA에게 오프로드한 kernel 코드이다. 제시된 코드에서는 최대 성능을 달성하기 위해 다음과 같이 DPA 쓰레드의 병렬성을 사용한다.

① reduce 단계: 데이터를 취합하는 reduce 단계에서, 부모 노드는 여러 DPA 쓰레드를 사용하여 동시에 다수의 자식노드로부터 데이터를 병렬로 읽어온다.

② scatter 단계: 합산된 결과를 전파하는 scatter 단계에서 DPA 쓰레드들이 자식노드들에게 결과를 병렬로 전달한다.

이러한 DPA의 멀티쓰레딩 기능을 통해 통신비용을 줄이고, 데이터 이동 및 처리를 가속화하여 전체 분산처리 성능을 극대화한다.

```
/*****************/
/* reduce phase  */
/*****************/
thread_id =doca_dpa_dev_thread_rank(); //dpa thread id 할당
num_children; /* 자식노드 개수 */
 if (자신이 leaf 노드가 아니면){

 /* DPA 쓰레드로 자식노드의 데이터 읽기를 병렬로 수행 */
 for(i=thread_id; i<num_children;i+=num_threads){
     doca_dpa_dev_sync_event_wait_gt(...)
          /*자식노드의 계산이 끝나기를 기다림*/
     doca_dpa_dev_rmda_post_read(...);
          /* 자식 노드에서 값을 가져옴 */
     doca_dpa_dev_rdma_synchronize(...)
          /* 비동기 통신이 끝나기만을 기다림 */
 }

/* DPA 쓰레드간의 동기화 코드 필요 */

 for(i=0;i<num_children ;i++)
  reduce_value=reduce_value+data[i];
 /*자식노드로부터 얻는 값과 자신의 값을 합산한다*/.
if(자신이 root 노드가 아니라면)
  doca_dpa_dev_rdma_signal_set(....)
      /* 합산이 다 이루어졌음을 부모노드에게 알린다*/

/*****************/
/* scatter phase  */
/*****************/
if (root 노드가 아니면)
  doca_dpa_sync_event_wait_gt(...)
    /* 부모노드의 전송이 완료되기를 기다림 */

/* DPA 쓰레드로 자식노드로의 전송을 병렬로 수행 */
for(i=thread_id;i<num_children;i+=num_threads){
      doca_dpa_dev_rmda_post_write(...);
      /*자식노드에게 부모노드로부터 받은 값을 전달한다 */
      doca_dpa_dev_rdma_synchronize(...);
          /* 비동기 함수의 완료를 기다림 */
      doca_dpa_dev_rdma_signal_set(...);
       /*자식노드에게 값을 전달했음을 알린다 */
}
```

<그림 2>allreduce 통신을 위한 DPA 커널 psuedo 코드

4. 결론

본 논문에서는 NVIDIA BlueField-3의 DPA을 활용하여 MPI 집합통신의 성능을 극대화하는 코드 설계에 중점을 두었다. 이를 위해 DPA 환경의 특성에 가장 적합한 Allreduce 알고리즘을 비교 및 분석하였다. 나아가, 이 분석을 통해 선정된 최적의 알고리즘을 NVIDIA DOCA 프로그래밍 모델을 사용하여 구현할 수 있도록 의사(pseudo) 코드로 제시하였다. 본 연구는 고성능 컴퓨팅 환경에서 DPU의 병렬처리 능력을 집합 통신 가속에 효과적으로 접목할 수 있는 실질적인 설계 방향을 제시하는 데 기여한다.

이 논문은 대한민국 정부(과학기술정보통신부)의 재원으로 한국과학기술정보연구원 수요자 맞춤형 연구환경 제공을 위한 초고성능컴퓨팅 기술개발 사업의 지원을 받아 수행된 연구임(과제번호: K25L1M2C2)

참고문헌

[1] https://aws.amazon.com/ko/blogs/tech/2024-aws-reinvent-ec2-instance-recap/

[2] https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/documents/datasheet-nvidia-bluefield-3-dpu.pdf

[3] Michael Beebe, et al, “OpenSHMEM Performance on Bluefield-3 Data Processing Units (DPUs)”, PEARC '25: Practice and Experience in Advanced Research Computing 2025: The Power of Collaboration

[4] https://docs.nvidia.com/doca/sdk/doca+dpa/index.html