Data Locality를 활용한 VR환경에서의 대용량 데이터 가시화 시스템의 성능 개선

이세훈*, 김민아, 이중연, 허영주
한국과학기술정보연구원
sehooi@kisti.re.kr, petimina@kisti.re.kr, jylee@kisti.re.kr, popea@kisti.re.kr

Performance Enhancement of A Massive Scientific Data Visualization System on Virtual Reality Environment by Using Data Locality

Se-Hoon Lee*, Min-Ah Kim, Joong-Yeon Lee, Young-Ju Hur
*Korea Institute of Science and Technology Information

요 약

GLOVE(GLObal Virtual reality visualization Environment for scientific simulation)는 컴퓨팅 자원의 성능 향상으로 데이터 양이 급속히 증가한 응용 과학과 전산 시뮬레이션 분야의 대용량 과학 데이터를 효율적으로 가시화하여 분석하기 위한 도구이다. GLOVE의 데이터 관리자인 GDM(GLOVE Data Manager)은 대용량 데이터의 분산 병렬 가시화를 위해 분산 공유 메모리를 제공하는 GA(Global Array)를 이용해 테라 바이트 단위의 데이터를 실시간으로 처리한다. 그러나 대용량 과학 데이터를 가시화 하는 과정에서 기존의 Data Locality를 고려하지 않은 데이터 접근 방식으로 인한 성능 저하를 확인했다. 본 논문은 기존 GLOVE에서 발견한 성능 저하 현상을 밝히고, 이에 대한 해결 방법을 제시한다.

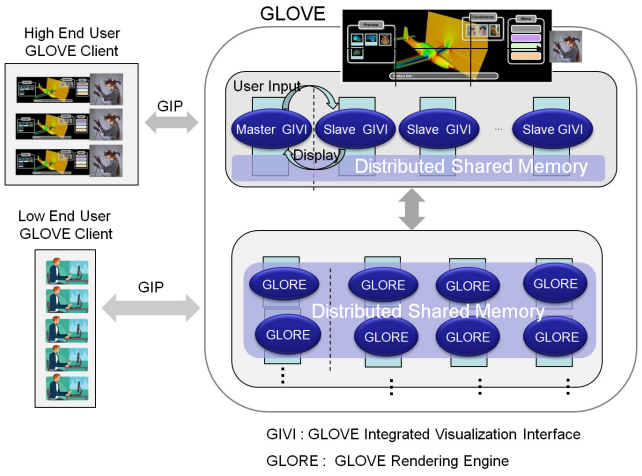
1. 서론

지구과학, 기상, 천체물리, 항공우주공학 등의 분야에서 복잡한 과학 현상을 분석하기 위해 사용되는 시뮬레이션 데이터의 분석 방법은 크게 그래프, 수치 데이터를 이용한 정량적 분석과 데이터 가시화를 통한 정성적 분석의 두 가지로 나누어진다. 가시화를 이용한 정성적 분석을 통하면 전체 데이터의 양상과 흐름을 알 수 있고, 시뮬레이션이 올바른 방향으로 진행되고 있는지에 대한 즉각적인 판단도 가능하다.

최근 컴퓨팅 자원의 급격한 성능 향상으로 시뮬레이션을 통해 더 정밀하고 규모가 큰 실험이 가능해 짐에 따라 시뮬레이션 결과 데이터의 양도 테라, 페타 바이트 단위로 기하급수적으로 증가하는 추세이다. 이에 따라 과학 데이터 가시화 관련 기술도 병렬 데이터 처리, 고해상도 렌더링, 복잡한 데이터를 알아보기 쉽게 보여주기 위한 표현 방법 등을 지원하도록 발전해왔다.

1) GLOVE

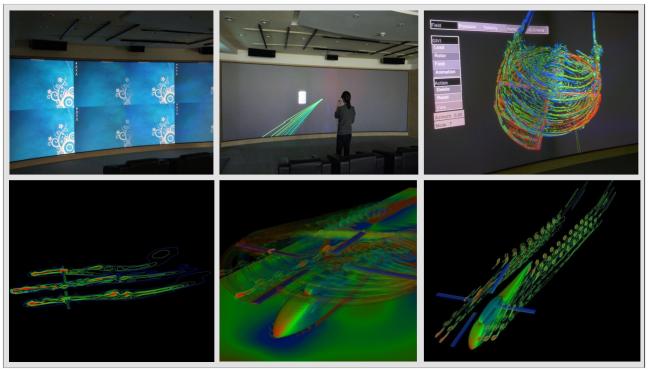
GLOVE[1]는 이러한 기술적 발전과 요구를 수용하기 위해 개발된 가시화 도구들 중 하나로, 고성능 컴퓨팅 환경에서 대용량 시뮬레이션 데이터를 효과적으로 분석하고 가시화하여 공유한다. 고해상도의 Tiled Display와 VR(Virtual Reality)환경은 응용 연구자에게 데이터 분석에 있어 직관적으로 다른 시각의 관점을 제공할 뿐만 아니라 다계층의 데이터 구조는 응용이 달라지더라도 최소한의 수정으로 응용에 맞는 하나의 새로운 인터페이스를 제공한다.



(그림 1) GLOVE 시스템 구조

GLOVE가 데이터로 주로 활용한 로터 시뮬레이션 데이터는 헬기가 이착륙할 때 발생하는 기류의 흐름을 시뮬레이션 해보고 소음, 진동, 로터의 변형 등 새로운 헬기 로터 설계 시 발생할 수 있는 상황을 예측하는데 활용된다. 로터 시뮬레이션 데이터의 주요 현상은 주로 헬기 로터 주변의 유동 데이터에서 관찰할 수 있는데 가시화 도

구를 통해 이러한 현상들을 인지할 수 있도록 직관적으로 보여줄 수 있어야 한다.



(그림 2) 로터 데이터 가시화 결과

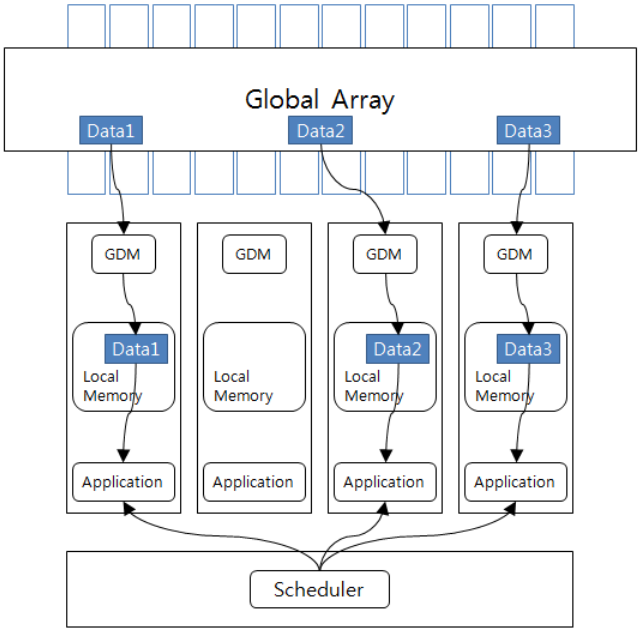
2) GDM

대용량 데이터를 처리하는 가시화 도구는 성능이 가장 중요하다. GLOVE에서는 분산 공유 메모리 데이터 관리자인 GDM이 대용량 데이터를 효율적으로 관리 및 처리하는 역할을 맡고 있다.

GDM은 기본적으로 데이터에 대한 입출력을 병렬 처리함으로써 대용량 데이터에 대한 처리를 가능하게 한다. 일반적으로 사용자들은 최대한 많은 데이터를 메모리에 올려 작업을 수행하고 싶어 하지만 노드 하나의 메모리는 한정되어 있기 때문에 대용량 데이터의 경우 오랜 로딩 및 처리 시간을 참을 수 밖에 없다. 특히나 가시화에 있어 많은 시간이 소요되는 부분이 데이터 로딩이다. GDM을 활용한 GLOVE는 한 노드에서는 처리 불가능한 1.21TB의 데이터를 158.4초에 로딩한다[1].

GDM은 대용량 데이터 처리를 위한 부하 분산 기능을 수행한다. 대용량 데이터는 일반적으로 하나의 노드에서 처리가 힘들기 때문에 데이터를 여러 노드에 미리 나누고 병렬처리 하는 Static Data Decomposition 방식을 활용한다. 하지만 컴퓨팅 자원이 부족해도 데이터가 없는 노드에는 작업을 분배할 수 없는 Static Data Decomposition 방식의 한계를 극복하기 위해 GDM은 분산 공유 메모리의 하나인 GA[2]의 기능을 이용해 데이터 존재 유무에 관계없이 모든 노드에 작업을 분배함으로써 공정한 부하 분산 효과를 얻는다.

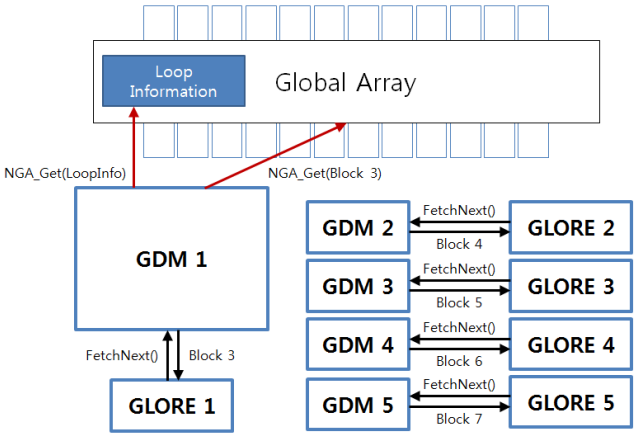
GDM에 데이터를 로딩하면 각 노드에 이를 분산한다. 가시화 요청을 받은 스케줄러는 데이터의 위치에 관계없이 모든 노드에 작업을 분배하고, 각 노드에 분산된 가시화 엔진들이 각 노드에 병렬 처리 작업을 스케줄링하여 요청하면 각 노드에 있는 GDM이 데이터가 자신의 노드에 있지 않더라도 GA 영역에서 찾아 로컬 메모리에 복사한다. 이를 통해 데이터의 위치에 관계없이 공정한 부하 분산이 가능하게 된다. (그림 3)은 스케줄러가 작업을 분배할 때 GDM이 이를 어떻게 처리하는지 보여준다.



(그림 3) GDM 데이터 처리 동작

2. 기존 기법의 문제점

GDM은 응용에 맞는 다양한 패턴의 데이터 접근 방식을 제공한다. 그 중 하나인 Loop-mode Data Fetch의 경우 전체 데이터 블럭을 라운드 로빈 방식으로 순차적으로 제공하는 방식이다. 각 노드는 공급받은 데이터에 대한 가시화 연산이 끝나는 대로 경쟁적으로 GDM으로부터 데이터를 공급받아 처리함으로써 Dynamic Data Distribution을 수행한다.

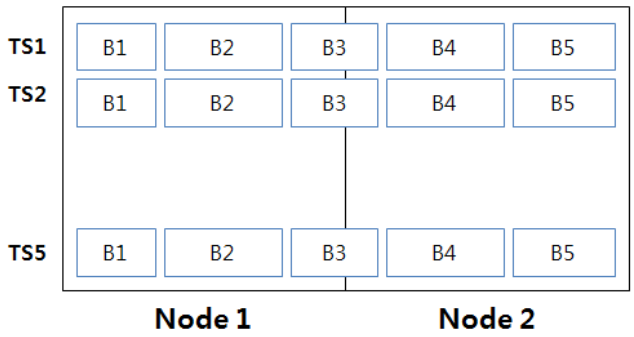


(그림 4) Loop-mode Data Fetch 동작

(그림 4)는 Loop-mode Data Fetch의 동작을 보여 준다. GDM이 Loop-mode Data Fetch 요청을 받으면 GA 영역에 있는 정보를 통해 다음에 처리해야 할 블럭 번호를 가져오고 해당하는 블럭을 GA API를 통해 가져온다. Loop에 대한 정보는 GA 영역에 존재하며 다른 노드에 있는 GDM과 공유할 수 있다. 따라서 다른 노드에서 요청이 들어오면 순차적으로 블럭 번호를 할당해준다. (그림 4)와

같이 각 노드에서는 요청한 순서대로 각각 블럭 3, 4, 5, 6, 7을 받는다. 이러한 방식으로 Loop-mode Data Fetch 기법은 할당된 모든 노드들이 GA 영역에 저장된 Loop 정보에 대한 접근만으로 스케쥴러나 명시적인 노드 간 정보 교환 없이 라운드 로빈 방식의 Load Balancing 효과를 제공한다.

데이터를 로딩하게 되면 우선 GA 영역이 만들어진다. GA 영역은 각 노드 별로 균등하게 나누어 생성된다. GDM에 저장되는 필드 데이터는 세로 TimeStep, 가로 Block번호의 2차원 배열 형태로 구성되는데 예를 들어 5개의 TimeStep, 5개의 블럭으로 이루어진 필드를 2개의 노드로 구성된 GA 영역에 로딩하면 (그림 5)와 같이 저장된다. 각 블럭의 크기는 서로 다를 수 있다. 현재 노드1은 블럭1~3을, 노드2는 블럭3~5를 가지고 있다. 이 데이터에 대해 병렬적으로 데이터 요청이 들어오는 경우 기존의 Loop-mode Data Fetch 기법은 Data Locality를 고려하지 않고 순차적으로 블럭 번호를 할당한다.



(그림 5) GA 영역에 저장된 데이터의 구조

예를 들어 N1~5까지 5개의 노드로 부터 N3, N2, N4, N1, N5의 순서로 데이터 요청이 들어온 경우 N3이 블럭1을, N2가 블럭2, N4가 블럭3, N1이 블럭4, N5가 블럭5를 가져가게 된다. 이 경우 자신의 노드에 데이터가 있음에도 불구하고 다른 노드에 있는 데이터를 가져오게 되어 GA 영역에서 로컬 메모리로 복사를 하게 되고, 이에 따른 네트워크 비용이 발생하게 된다. 특정 노드로부터 데이터 요청이 들어왔을 때 그 노드에 있는 블럭을 우선적으로 할당할 수 있다면 데이터를 가져오는 시간을 줄일 수 있고 대용량 데이터의 실시간 처리가 더 빨라져 가시화 도구의 성능 향상 효과를 볼 수 있다.

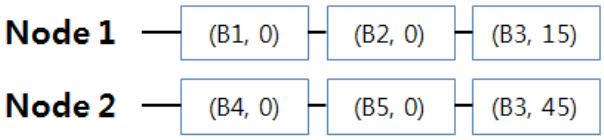
3. 개선사항

본 연구에서는 앞서 설명한 기존 GDM의 Loop-mode Data Fetch 기법에서 Data Locality를 고려하지 않고 데이터를 가져오면서 발생하는 네트워크 비용을 최소화하기 위해 다음과 같은 방법을 제안한다.

먼저 데이터를 로딩할 때 각 노드별로 가지고 있는 블럭 정보를 계산하여 유지한다. GA가 제공하는 NGA_Distribution() API를 이용하면 각 노드가 가지고 있는 GA 영역의 범위를 알 수 있다. 이 영역 범위와 각 블럭의 크기 및 위치를 비교하여 계산하면 각 노드가 어떤 블럭을 가지고 있는지 계산 가능하다. (그림 5)의 예를 살펴보면 노드1이 블럭1~3, 노드2가 블럭 3~5를 가지고 있다는 정보를 얻게 된다.

GDM은 블록의 일부분이 아닌 전체를 가져와서 처리한다. 때문에 블럭이 여러 노드에 나누어 저장되더라도 처리할 때는 하나의 노드가 독점적으로 가져가야 한다. 블럭 3과 같이 두 노드에 나누어 저장되어 있는 경우가 이에 해당한다. 이를 처리하기 위해 자신의 노드에 속하지 않은, 즉 다른 노드가 소유한 부분의 크기 정보를 함께 유지한다. 따라서 네트워크 비용을 최소화 하기 위해서는 블럭의 많은 부분을 가지고 있는 노드가 이 블럭을 가져가 처리해야 한다.

결론적으로 한 프로세스가 블럭을 가져가 처리할 때 자신만이 소유한 블럭을 우선적으로 가져오고, 더 이상 없을 경우에는 다른 노드가 소유한 부분의 크기가 작은 순서대로 가져와 처리한다. 이를 위해 (블럭번호, 다른 노드가 소유한 영역의 크기)를 리스트에 저장하고, 다른 노드가 소유한 영역의 크기가 작은 순으로 정렬하여 유지한다. (그림 6)은 (그림 5)의 경우 계산된 각 노드별 블록 리스트를 보여준다.



(그림 6) 각 노드별 블록 리스트

요청이 들어올 때마다 리스트에서 정렬한 순서대로 하나씩 제공하면 GA 영역에서 로컬 메모리로 복사하는 네트워크 비용을 최소화 할 수 있다.

리스트에 블록이 없는 경우는 가지고 있는 블록이 모두 처리된 상태이거나 블록을 가지고 있지 않은 노드가 데이터 처리를 요청하는 경우이다. 이 경우에는 GA 영역에서 무조건 로컬 메모리로 복사를 할 수 밖에 없기 때문에 기존의 방식과 동일하게 아직 처리되지 않은 블록을 순차적으로 할당한다.

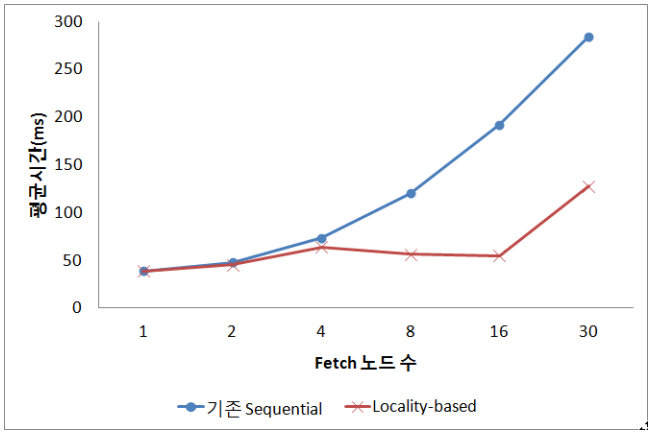
4. 실험결과

개선된 기법의 성능을 검증하기 위해 로터 시뮬레이션 데이터 가시화를 수행하고, 기존의 순차적으로 할당하는 방식과 Data Locality를 고려한 방식을 비교했다. 실험은 1.21TB의 데이터의 일부를 30개의 노드(렌더링 노드별 메인 메모리 64GB, CPU Clock 3.0GHz)에 로드하고, 이에 대해 가시화 병렬처리를 수행하며 성능 측정을 위해 각 노드별 평균 Fetch 시간을 측정했다.

(그림 7)은 그 결과를 보여준다. 1개의 노드로 Fetch를 수행하는 경우는 두 방식이 모두 동일하게 동작하기 때문

에 두 기법의 성능 역시 동일하게 나타남을 알 수 있다.

Fetch 병렬처리를 수행하는 노드 수가 늘어날수록 평균 소요시간이 더 많이 줄어드는 모습을 보이는데 이는 현재 30노드에 데이터가 균등하게 분배되어 있기 때문에 Fetch 병렬처리를 수행하는 노드 수가 늘어날수록 자신의 로컬 메모리에 있는 데이터를 바로 처리하는 비율이 높아지기 때문이다. 30노드로 동시에 Fetch를 수행한 경우 기존 방식의 평균 소요시간 284.21ms에 비해 127.63ms로 약 2.23배의 성능 개선 효과가 있음을 알 수 있다.



(그림 7) Fetch 노드 수 증가에 따른 성능 비교

5. 결론

GLOVE는 고성능 컴퓨터 환경에서 대용량 시뮬레이션 데이터를 실시간으로 병렬 가시화하기 위한 도구로 데이터를 효율적으로 처리할 수 있는 성능이 중요하다. 본 연구에서는 기존의 Loop-mode Data Fetch 기법에 Data Locality를 고려한 데이터 접근 방식을 도입해 성능 개선을 위한 방안을 제시했고, 이에 대한 실험 결과, 다수의 노드를 활용해 병렬 가시화를 수행할 경우 기존 GLOVE의 성능을 개선할 수 있음을 확인했다.

향후 GLOVE는 Particle Tracing을 지원할 예정이다. 비슷한 위치의 데이터에 자주 접근하는 특성을 활용하여 이미 가져간 이웃의 데이터를 Pre-Fetch하는 기법을 도입하여 병렬 가시화 성능을 더 개선하고자 한다.

참고문헌

[1] Min-Ah Kim et al, "GLOVE(GLObal Virtual reality visualization Environment for scientific simulation): VR 환경에서의 대용량 데이터 가시화 시스템" 2010 한국컴퓨터종합학술대회 논문집 제37권 제2호(B), 267-271

[2] Jarek Nieplocha , Bruce Palmer , Vinod Tipparaju , Manojkumar Krishnan , Harold Trease , Edoardo Aprà, “Advances, Applications and Performance of the Global Arrays Shared Memory Programming Toolkit”, International Journal of High Performance Computing Applications, vol.20 No. 2, pp.203-231, May 2006

[3] H. Shan and J. P. Singh, "A Comparison of Three Programming Models for Adaptive Applications on the Origin2000," in proceedings of Supercomputing, 2000

[4] Laurent C., Chrisophere M, Xavier C. and Bruno Levy, "Distributed Shared Memory for Roaming Large Volumes", IEEE Transactions on Visualization and Computer Graphics, vol. 12, No.5, Sept/Oct 2006