GridRPC의 DAG 기반 Co-scheduling을 위한 프로그래밍 인터페이스

최지현, 이동우, 김미옥, R.S.Ramakrishna
광주과학기술원 정보통신공학과
e-mail: {jhchoi80, leepro, mokim, rsr}@kijst.ac.kr

Programming Interface for DAG-based Co-scheduling of GridRPC

Ji-Hyun Choi, Dong-Woo Lee, Mi-Ok Kim, R.S.Ramakrishna
Dept. of Information and Communication, K-JIST

요 약

이 논문에서는 그리드환경에서 Remote Procedure Call(RPC) 프로그래밍 인터페이스를 위한 메커니즘인 GridRPC의 성능향상을 위해 DAG 기반의 Co-scheduling API를 제안한다. 네트워크 상의 통신없음을 중점으로 GridRPC를 최적화하기 위한 프로그래밍 인터페이스와 이를 가능하게 하는 서버구조를 제안한다. DAG 기반의 co-scheduling은 서버-클라이언트간의 연산에 사용되는 입력값과 출력값들의 흐름을 분석하여 사용자에 하여금 DAG(Directed Acyclic Graph)로 GridRPC call들을 구성하고 이를 기반으로 GridRPC call들을 최적화하는 방법이다. 또한, GridRPC가 Client Interface이기 때문에 생기는 문제점인 서버간의 지원의 문제점이나 SOAP 서버의 Wrapping을 통해 해결한다.

1. 서론

그리드 컴퓨팅 기술이 차세대 컴퓨팅기술의 기반이로 인정되고 있음에 따라, 그리드 분야에서 중요한 이슈들이 많이 생겨나고 있다. 자원을 신속하게 제공하는 방법이나 최적의 비용으로 계산 경로를 구하는 것은 중요한 문제이며 네트워크의 속도보다 빠른 반응 비용으로 발전하는 프로세스의 속도로 인해, 데이터 전송은 고성능 컴퓨팅 아키텍처에서 큰 오버헤드를 가지고 있다. 이 논문에서는 서버-클라이언트간의 통신을 최적화하여 아키텍처의 성능을 향상시키기 위해 GridRPC(1.5)의 call을 co-scheduling함으로써 얻을 수 있는 성능향상을 위해서 다룬다. GridRPC는 그리드 환경에서 원격 라이브러리 접근과 병렬처리 프로그래밍 모델을 지원하는 마들레어 인터페이스이며, 대표적인 시스템으로는 Ninf와 NetSolve등이 지원하고 있다. GridRPC는 그리드에 가장 적합한 서버에 명시된 자원을 활용하거나 여러 개의 서버에 존재하는 다른 파라미터들의 계산이나 다양한 형식의 병렬 처리 프로그래밍 레벨에서 사용자에게 프로그래밍 인터페이스의 투명성을 제공하면서 네트워크에 산재 되어 있는 컴퓨터 자원을 사용하도록 한다. 하지만 스케일링을 고려하지 않는 RPC의 경우 통신비용에 많은 오버헤드를 가지고 있다. 스케일링을 통해서 네트워크 통신에 들여가는 오버헤드를 줄일 수 있는 방법의 하나로 RPC call들을 sequencing하여 통신 비용을 줄이고자 하는 것이다. 이것이 바로 GridRPC의 Co-scheduling이며, 이것을 통하여 성능이 향상된 GridRPC를 사용하고자 한다.

2. GridRPC


2.1 GridRPC의 특성

그리드 환경에 맞게 GridRPC는 비동기성의 병렬처리
리터와 하이레벨의 프로그래밍 모델을 제공한다. 프로그래머들이부터 그래드의 분산성, 불확실성, 동작 성능 등에 구매하지 않도록 한다. Coarse-grained call을 대한 지원과 비동기적인 병렬처리 프로그래밍의 다양한 스타일을 지원해야 이미 만들어진 다양한 미들웨어로 구현된 서비스 할수록가 개발자가 이를 이용할 수 있도록 한다. 이와 같이 GridRPC는 개개의 애플리케이션이 분산되는 것뿐만 아니라 그리드에서 분산되고 결합된 계산을 요구하는 웹포너드들과 같은 높은 레벨의 소프트웨어 등의 기반을 제공할 수 있다.

2.2 GridRPC vs. RPC


3. 문제점


DAG[3]는 서로 의존성을 가진 프로세스의 집합으로 구성된 병렬 프로그램을 모델화 하는데 쓰이는 일반적인 방법이다. DAG에 있는 하나의 노드는 같은 프로세스에서 시작하여 순서대로 실행되며 구현된 병렬 프로그램의 모델화하는데 쓰이는 일반적 방법이다. DAG에 있는 하나의 노드는 같은 프로세스에서 시작하여 순서대로 실행되며 그 노드가 종료되면 그 노드의 작업을 표현한 것이다. 각 노드는 모든 임력값이 입력되었을 때, 실행이 시작된다. 그 그래프의 작업들의 중복 순서를 표현하는 단방향 edge들을 가진다. 이 중복 순서는 DAG의 실행 체계를 의미한다.

본 논문의 목표는 GridRPC에서 네트워크 전송을 줄이고 전체 request의 response time을 줄이도록 하기 위해서 Grid의 새로운 인터페이스를 제안하는 것이다. 이를 통해 불필요한 데이터를 전송하지 않으면서 또한 모든 필요한 데이터를 전송하도록 해야 한다는 것이 요구된다. 일부의 request의 입력과 출력 파라미터의 세부적인 분석을 수행하고 각 작업과 그 작업의 실행 의존성을 표현한 DAG를 생성하는 과정을 통해서 위와 같은 설계 목표를 이룰 수 있다. 사용자의 의해 생성된 DAG는 작업의 실행을 위해 request들이 스택을 필 시스템의 서버에 전송되어 실행을 최적화하기 위한 정보로 활용된다.

[그림1] Request sequencing을 이용한 클라이언트와 서버간의 데이터 흐름 비교


4. GridRPC에서 DAG 기반의 Co-scheduling을 위한 API와 Wrapper 서비스 구조 제안


4.1 일련의 call에 대한 DAG 생성

다음에 그림2의 일련의 GridRPC Call에 의해 생성된 DAG가 일련의 그림2[2]는 일련의 작업을 위한 일련의 GridRPC call들에 대한 서버간의 데이터 흐름을 표현한 DAG이다. 여기서 DAG의 역할은 클라이언트-서버간의 연산의 진행조차를 명시하고 클라이언트가 요청한 일련의 작업을 위해서 서버간의 Call에 대한 입력 데이터와 출력데이터를 분석하여 데이터의 흐름이 각각 다른 서버에서 일어나는 계산의 결과값들을 클라이언트에게 반환하지 않고 중간 결과값이 되는 출력 값들
4.2. 사용자 DAG로부터 생성된 GridRPC call

하나의 작업을 위해서 수행되어야 하는 일련의 call들이 시작을 선언하기 이전에 모든 call에 대한 입력 값이 들어있는 이후에 실행하도록 한다. 일련의 call들이 끝날 때까지 입력 변수와 출력 변수들은 클라이언트에게 반환되지 않고 서버에 저장되어 있어야 한다는 서버의 사용에 사용된다.

여기서 제안된 API는 DAG의 구성, 수행, 결과확인으로 구성된다.

**DAG의 구성**

GridRPC_DAG_Begin( “DAG폴더” ) : DAG 클록 시작을 표시하는 것으로 “DAG폴더” 로 관리된다.

GridRPC_DAG_Call( “함수명” , “결과값” , 입력 파라미터 1, 2, …, 입력파라미터n) : 하나의 GridRPC Call 을 의미하는 것으로 “함수명” 은 서버의 구현에 있는 함수명을 의미한다. “결과값”은 입력 파라미터들과 지정한 서버함수에 의해 수행된 결과를 의미하는 값이다. 이 “결과값”은 다음과 나오는 DAG Call의 입력 파라미터로 서서하여 DAG를 구성 할 수 있도록 한다.

GridRPC_DAG_End( “DAG폴더” ) : “DAG폴더” 의 DAG Call의 클록을 의미한다. 전송이 가능하도록 marshalling한다.

**DAG클의 실행**

GridRPC_DAG_Submit( “DAG폴더” ) : 구성된 DAG클을 전송하고 DAG ID를 반환한다.

**결과확인**

GridRPC_DAG_result( “DAG폴더” ) : DAG클의 수행의 완료를 알려는 것으로 false와 true값을 반환한다.

GridRPC_DAG_GetResult( “DAG폴더” ) : 지정된 “DAG폴더” 과 그에 해당하는 마지막 “결과값”로 DAG클의 결과를 반환한다.


4.3 서버 지원의 문제점

GridRPC를 이용한 Co-scheduling에서는 GridRPC가 클라이언트와 서버가 각각 프로그래밍됨으로 인한 서버지원의 문제점이 존재한다.

GridRPC는 클라이언트와 서버의 간의 인터페이스로서 클라이언트가 서버와의 통신을 제어할 수 있는 레터널이 부재한다. 그러므로 그러므로 여기서 서버의 서버의의 수 행을 위해 서버 와 패치를 제안한다.

[그림 4] 클라이언트와 서버간의 GridRPC call의 Co-scheduling을 위한 Server Wrapper로 SOAP Server와 SOAP Client


서버를 런타임으로 생성하는 SOAP서버로 인해 클라이언트가 보낸 일련의 call들을 수행할 서버간의 정보를 얻고 그 서버들간의 데이터를 전송하고 데이터의 흐름을 관리할 수 있도록 한다.
6. 결론

GridRPC는 loosely-coupled 시스템에서 이기종 시스템간의 통신을 요구하므로 서버들과 클라이언트간에 고성능의GridRPC call이 중요하며 서버간의 통신을 위한 Co-scheduling이 상호작용을 위해 필요하다. 서버들간의 스케줄링은 DAG를 기반으로 하여 네트워크에서 생성되는 병렬요소로 정록되는 데이터를 제거하여 최적의 네트워크 전송비용을 가능하게 한다. 클라이언트측의 인터페이스를 가지고 있는 GridRPC가 있어 서버간의 Co-scheduling을 요구하는 API 제공을 위 한 문제가 되는 서버간의 지원받을 SOAP 서버를 이용하여 Server wrapper를 이용함으로써 클라이언트가 서버간의 통신을 자유롭게 할 수 있도록 하고 제산된 DAG 기반의 GridRPC의 Co-scheduling을 위한 서버간의 통신에 있어 사용자에게 편리함을 제공하여 최적의 네트워크 전송비용으로 GridRPC call을 가능하게 한다.

참고문헌