대용량 파일에 의한 프로세스간의 동기화

하성진[0*] 황선태* 정갑주** 이지수***
*국민대학교 컴퓨터학부
**건국대학교 인터넷미디어공학부
***한국과학기술연구소 컴퓨팅센터
{sjha[0*], sthwnag*}@cs.kookmin.ac.kr, jeongk**@konkuk.ac.kr jysoo***@hpcnet.ne.kr

Inter-Process Synchronization by Large Scaled File

Sungjin Ha[0*] Suntae Hwang* Karpju Jeong** JySoo Le***
*Department of Computer Science, Kookmin University
**College of Information and Communication, Konkuk University
*** Korea Institute of Science and Technology Information Computing Center

요 약

최근에 지역적으로 분산된 컴퓨팅 자원을 어디에서나 활용할 수 있도록 해주는 GRID가 많은 주목을 받고 있다. 특히 단백질 분자모사나 고에너지 물리학 분야 등과 같이 매우 많은 계산을 요구하는 분야에서는 GRID를 통해서 계산 자원을 제공받을 수 있다. GRID에서 제공되는 계산 능력을 잘 활용하기 위해서 각 분야에서 사용되는 어플리케이션을 병렬화할 수 도 있지만 이미 계산 방법이나 결과가 검증되어 있는 기존의 패키지를 활용하는 것도 매우 중요하므로 기존 패키지에 의한 직렬 또는 지역적으로 병렬인 프로세스를 매우 많이 생성하여 GRID를 채우는 것도 한 방법이라 하겠다. 일반적으로 이와 같은 패키지는 기동할 때에 패러미터 파일을 참조하게 되고 그 계산 결과는 매우 큰 파일로 출력이 되는데 본 논문에서는 대용량 파일에 의해서 프로세스 간에 동기화 및 통신을 이루어야할 때 발생하는 문제를 해결하는 방안을 제시한다. 동기화와 통신을 동시에 다루어야 하므로 Linda 개념을 도입하였으며 기존 Linda에서는 Tuple Space안에서 대용량 파일 처리를 고려하기 어려우므로 이에 대한 해결책을 제안하였다.

1. 서 론

최근에는 지역적으로 분산된 고성능 컴퓨팅 자원을 네트워크로 연결하여 마치 하나의 컴퓨터처럼 자원을 활용하는 GRID가 많은 주목을 받고 있다. 이런 GRID는 분산된 많은 계산 자원을 활용할 수 있다는 점에서 생물학이나 물리학, 화학 등의 많은 계산량을 요구하는 분야에서 특히 많은 관심을 가지고 있으며 그런 분야와 관련되어 많은 연구들이 진행되고 있다. 이런 연구를 통해 지역적으로 분산된 대용량의 데이터와 컴퓨팅 자원을 이용해서 결과를 얻고 이를 공유하는 애플리케이션이 다수 등장하게 되었다. 이런 애플리케이션으로는 DNA의 염기 배열순서 판별, 단백질의 분자 구조 모사, 고에너지 물리학 및 기후 모델링 등의 많은 분야에서 여러 형태로 구현되고 있다. 그러나 이런 애플리케이션이 가지는 공통적인 문제는 계산을 위한 대용량의 데이터와 대용량의 계산 결과를 어떻게 효율적으로 공유하고 이를 통해 원거리 프로세스간의 효율적인 동기화 수행에 있다. 또한 각각의 분산된 공유자원을 통해 이루어지는 계산과정상에 발생할 수 있는 장애에 대한 효율적인 복구 메커니즘에 있다[1][2].

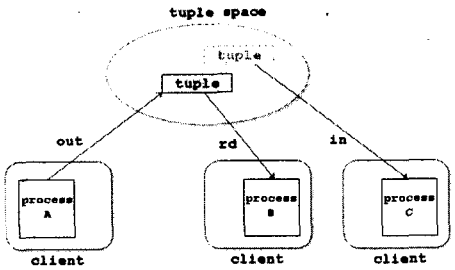
이 논문에서는 지역적으로 분산된 대용량의 데이터를 수집해서 계산을 수행하고 수행된 결과를 대용량 파일을 통한 애플리케이션간의 이동시 분산 자원간의 공유 메모리를 통한 프로세스간의 동기화를 수행하는 모델의 제시에 있다. 이런 모델의 구현을 위해 일종의 가상 공유 메모리를 가지는 병렬 프로그램 모델인 Linda 모델을 대용량 파일을 통한 동기화 구조로 확장한다. 이를 통해 원거리 프로세스간의 대용량 데이터를 이용한 동기화 모델을 제시하고 향후 현재 그리드 환경을 구축하기 위해 이용되는 미들웨어로 확장에 그 목적을 두고 있다.

2. 관련연구

2.1 Linda

Linda는 tuple space라고 불리는 가상 공유 메모리를 가지는 병렬 프로그램 모델이다. Linda는 tuple space내에 tuple이라고 불리는 데이터 객체의 생성을 통해 프로세스간의 통신 및 동기화를 수행한다. tuple은 기본타입을 가지는 데이터들의 연속적인 필드의 집합이다. 따라서 각각의 프로세스들 간의 데이터 통신은 데이터를 보내려는 프로세스에 의해 tuple space에 tuple을 생성하며 이렇게 생성된 tuple은 이것을 요구하는 다른 프로세스들에게 데이터베이스 스타일의 일종에 패턴 매칭 메커니즘을 통해 원하는 tuple을 선택하고 가져가는 과정을 통해 데이터 전송이 이루어진다. 이런 데이터 전송 과정에서 매칭이 되는 tuple이 tuple space내에 존재하지 않으면 원하는 tuple이 들어올 때까지 프로세스는 잠들게 되고 원하는 tuple이 생성된 후 깨어나 남은 과정을 수행한다. 이런 과정을 통해 프로세스간의 상호 동기화를 수행할 수 있다.

Linda는 tuple을 제어하기 위해 다음과 같이 4개의 기본 연산을 제공한다. out 연산은 tuple space에 새로운 tuple 객체를 생성, eval 연산은 tuple space에 새로운 tuple 객체를 생성하고 생성된 tuple을 처리하기 위한 프로세스를 생성, in 연산은 tuple space에서 매칭 되는 tuple을 가져오고 tuple space에서 삭제, rd는 in과 같은 연산을 수행하지만 tuple space에서 삭제하지 않는다. 또한 4개의 연산을 tuple space에서 수행될 때 상호 배제적이기 때문에 서로의 연산에 영향을 주지 않는다.



[그림2] tuple space를 경유한 연산

따라서 out과 in 연산을 통해 두 프로세스간의 동기화된 일대일 통신을 수행한다[].

out ("foo", a, b)
in ("foo", ?x, ?y)

[그림3] out, in 연산

out 연산을 통해 "foo"문자열 필드와 정수 a, b를 가지는 두개의 필드로 구성된 tuple을 tuple space에 생성한다. in 연산을 통해 문자열 "foo"를 첫 번 필드로 가지고두개의 정수 필드를 가지는 tuple을 tuple space에서 매칭 메커니즘을 통해 찾은 후 정수변수 x, y에 정수상수 a, b의 값을 넣는다. 이때 tuple space에 존재하던 tuple은 제거된다.

Linda 모델이 다른 병렬프로그램 모델에 비해 가지는 장점은 구현이 간단하고 높은 유연성을 가진다는데 있다. 단지 위에 언급된 4개의 기본 연산을 통해 tuple space에 접근이 가능하고 특히 분산 환경에서 모든 프로세스들이 하나의 가상 공유 메모리인 tuple space를 공유하고 있기 때문에 프로세스들 간의 동기화 메커니즘을 구현하는데 많은 유연성을 가진다[3].

2.2 PLinda

Linda 모델이 가지는 tuple space를 통한 통신과 동기화는 익명성을 가진다. 따라서 이런 익명성을 통해 tuple space내에 존재하는 tuple은 특정 프로세스에 종속되지 않는다. 따라서 어느 프로세스라도 기본 연산을 이용하여 tuple space내의 매칭 tuple에 접근할 수 있다. 이런 특징을 이용해 Linda 모델을 트랜잭션처리와 장애허용을 위해 확장된 모델이 PLinda 모델이다. 또한 Plinda모델은 분산 환경을 기반으로 설계되었기 때문에 [그림6]과 같이 서버를 통해 네개의 기본연산에 대한 처리를 수행한다.

PLinda 모델에서 트랜잭션을 수행할 때 커미트를 위한 상태정보 등을 안정적인 tuple space에 out 연산을 통해 유지하고 프로세스 장애 발생시 다시 실행된 프로세스는 tuple space의 커미트 tuple을 in 연산을 통해 얻은 정보를 이용해 작업을 계속 수행할 수 있는 메커니즘을 제공한다. 이런 트랜잭션처리 및 장애허용 메커니즘은 기존에 Linda가 가지는 기본 연산을 통해 쉽게 구현이 된다는 장점을 가진다[4].

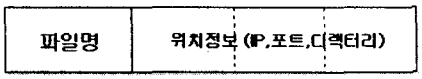
3. 대용량 파일을 통한 프로세스 동기화

Linda 모델은 다른 병렬 프로그램 모델에 비해 간단하고 분산환경에서 손쉬운 동기화 메커니즘을 제공한다. 따라서 이런 장점을 이용해 그리드 환경에 적용할 수 있다. 그러나 현재 Linda 모델의 tuple space는 단지 서버내의 메모리에 존재하기 때문에 대용량의 파일을 통한 프로세스간의 동기화된 통신에는 많은 한계점을 가진다. 따라서 이런 한계점을 극복하기 위한 확장이 요구된다.

대용량 파일을 통한 동기화는 작업을 수행하기위한 입력데이터, 작업의 결과데이터, 작업 중에 교환되는 중간데이터를 프로세스 상호간에 교환 시 요구되며 Linda모델에서는 이런 동기화를 통한 통신은 in, out 연산을 통해 수행한다. 따라서 확장을 위한 우선적인 고려사항은 대용량 파일을 통한 동기화 수행을 위해 in, out 연산을 수행할 때 tuple space와 물리 저장매체간의 신뢰할 수 있는 구조의 구축에 있다.

3.1 메타정보

대용량 파일을 통한 프로세스간의 동기화 메커니즘 구축을 위해 tuple space내의 tuple과 저장소의 대용량 파일간의 신뢰할 수 있는 구조를 가지는 메타 정보가 요구 된다. 이런 메타정보를 통해 사용자는 in, out 연산의 하나의 인터페이스를 가지고 대용량 파일을 통한 동기화를 수행할 수 있다. 메타 정보는 tuple내에 대용량 파일에 대한 정보를 가지는 tuple의 하나의 필드로 표현된다. 메타 tuple은 다음과 같은 필드를 가진다.

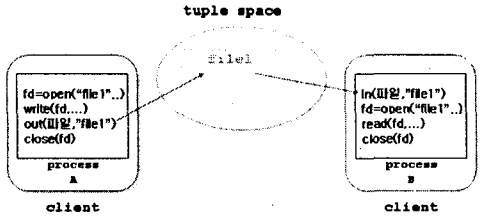


[그림4] 메타 tuple의 필드 구성

파일명 필드를 통해 통신하려는 파일명을 명시하며 위치정보를 가지는 필드를 통해 통신하려는 양쪽의 주소, 포트, 디렉터리 위치정보 등을 tuple의 하나의 필드 값으로 가지는 tuple을 구성한다.

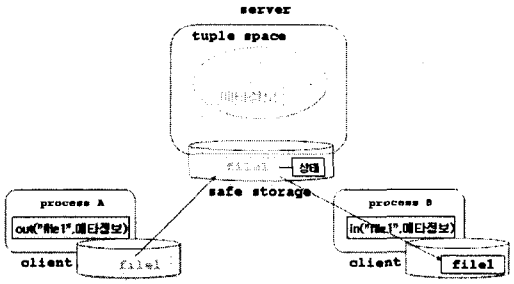
3.2 메타정보와 대용량 파일을 통한 동기화

out, in연산은 사용자의 관점에서 볼 때 기존의 Linda 모델과 대용량 파일을 통한 동기화를 위해 확장된 모델사이에 같은 인터페이스를 가진다. 단지 사용자는 파일 과 매치되는 메타정보를 가지는 tuple에 대한 in, out연산의 수행을 통해 tuple space를 통한 논리적인 통신만을 고려하면 된다. 그러나 시스템은 물리적으로는 두 프로세스간의 대용량 파일 통신을 수행한다.



[그림5] 사용자관점의 추상적 파일 통신

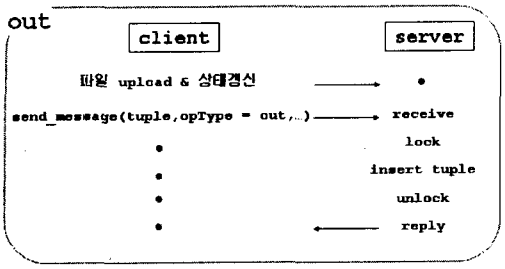
[그림5]에서처럼 사용자관점의 논리적 파일 동기화 통신은 단지 out, in연산을 통한 일반적인 방법과 같은 인터페이스를 사용한다.



[그림6] 시스템관점의 물리적 파일 통신

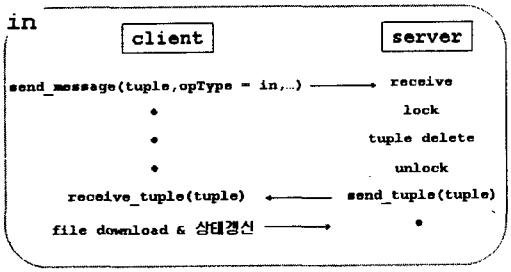
[그림6]에서처럼 시스템 관점의 물리적 대용량 파일 통신은 클라이언트와 서버의 safe storage간의 물리적 파일 전송이 이

루어진다. 이때 서버는 일종의 FTP서버의 역할을 수행한다.



[그림7] out 연산을 위한 구현

out연산을 수행하기 위한 구현은 [그림7]과 같다. 클라이언트는 우선 파일통신을 위한 대용량 파일을 서버의 safe storage에 업로드를 한다. 이때 파일 업로드에 대한 상태정보를 체크하고 out연산에 의한 tuple을 서버에 보낸다. 서버는 받은 tuple을 tuple space에 넣기 위해 tuple space를 잠근 후 tuple을 넣는다.



[그림8] in 연산을 위한 구현

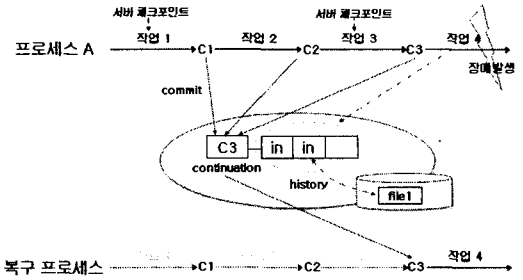
in연산을 수행하기 위한 구현은 [그림8]과 같다. 클라이언트는 tuple space에서 얻기를 원하는 tuple을 서버에게 보낸다. 서버는 클라이언트로부터 받은 tuple과 매칭되는 tuple을 tuple space에서 찾아 tuple space에서 제거한다. 이때 tuple space는 락 된다. 따라서 서버는 매칭되는 tuple을 클라이언트에게 돌려주고 클라이언트는 서버로부터 받은 tuple에서 파일에 대한 메타정보를 통해 서버에서 대용량 파일을 다운로드하고 서버에 있는 파일 상태 정보를 갱신한다.

3.3 장애허용 및 트랜잭션

지금까지 다루었던 대용량 파일 통신을 통한 동기화 기법은 많은 연구가 있어왔다. 그러나 Linda 기본 연산을 바탕으로 tuple space의 성질을 통해 장애허용 및 트랜잭션을 수행할 수 있게 확장된 Plinda 모델을 이용해 대용량 파일 통신을 구현함으로써 대용량 파일을 통한 프로세스간의 동기화 메커니즘 구축이 용이하고 무엇보다도 장애발생시의 복구 및 연속적인 수행을 할 수 있는 구조의 구현이 용이하다는 장점을 가진다.

장애복구를 위해 요구되는 상태 정보 등을 tuple space에 유지하고 히스토리를 관리함으로써 장애복구 시 중단된 작업을 계속 수행할 수 있는 배경을 제공한다. 특히 하나의 작업 내에서 수행된 연산의 히스토리관리를 통해 대용량 파일에 대해 in 연산 수행 후 바로 제거되지 않고 해당 작업에 대한 커미트 트랜잭션 후에 통신수행종료 로 파일의 상태를 유지한다. 이런 유지를 통해 시스템 장애발생시 복구를 할 수 있다[5].

이런 파일을 통한 통신에 있어서 우선적으로 고려될 사항은 통신을 수행한 파일을 언제 safe storage에서 제거하느냐에 있다. 특히 이 Plinda 모델에서는 장애허용을 위한 두 단계의 방법을 제공한다. 첫째는 서버의 tuple space를 위한 체크 포인트를 두어 체크 포인트 순간의 모든 tuple space의 내용을 디스크에 덤프를 통해 서버 장애를 복구하는 방법과, 각각의 클라이언트의 수행에 대한 커미트 트랜잭션을 tuple space에 유지함으로써 클라이언트의 장애를 복구하는 방법을 제공한다. 따라서 이 두 가지 장애 복구 방법간에 적절한 시기에 파일이 삭제되어야한다.



[그림7] 장애발생에 따른 continuation

4. 결론 및 향후과제

지금까지 PLinda모델의 out, in연산의 확장을 통해 대량의 데이터를 요구하는 응용 애플리케이션을 지원하기 위한 확장된 PLinda모델을 제시했다. 이런 확장된 PLinda 모델을 통해 그리드 환경에 적합한 모델로 확장하기 위해서는 아직도 많은 연구가 요구된다. 특히 그리드환경에서 이용되는 생물학이나 에너지 물리학 등의 응용 애플리케이션이 요구하는 데이터는 수십 기가 이상의 데이터 크기를 가질 수 있기 때문에 하나의 중앙 저장소에서 다루기 어렵다. 따라서 이런 대용량의 데이터를 지원하기 위한 저장소의 분산된 모델의 연구가 요구된다. 또한 저장소에 있는 파일을 수반하는 통신 시 서버 및 클라이언트의 장애에 따른 장애 복구를 위한 메커니즘 또한 연구되어져야 하며 특히 장애복구에 따른 대용량 파일의 삭제주기에 대한 연구도 수반이 되어야 한다. 추가적으로 메타 tuple과 그에 상응하는 대용량 파일의 전송의 동기화에 따른 성능 향상에도 관심을 가져야 한다. 따라서 이런 연구를 통해 궁극적으로는 PLinda 모델이 가지는 장점인 프로세스들 간의 동기화 구축과 장애허용 및 트랜잭션구축의 용이함 을 그리드 환경에 접목하기 위한 노력이 계속적으로 요구된다.

5. 참고문헌

[1] I. Foster, C. Keselman, S. Tuecke, "The Anatomy of the Grid", International J. Supercomputer Applications. 2001

[2] G.Allen, E.Seidel, J.Shalf "Scientiic Computing on the Grid", Scientific Computing, spring. 2002

[3] N.Carriero, D.Gelernter "Linda in Context", Communication of ACM, vol32. April. 1989

[4] K.Jeong, D.Shasha "PLinda 2.0 : A Transactional/Checkpointing Approach to Fault Tolerant Linda", Proc. the 13th International Symposium on Reliable Distributed Systems. Oct. 1994

[5] Strom.R, Yemini.S "Optimistic Recovery in Distributed Systems", ACM Transactions on Computer System, vol3. pp.204-226, 1985