□ 기술해설 □

확장된 메시지 전달 표준 MPI-2

숭실대학교 최재영*·김명호*·김정훈**

1. 서 론

메시지 전달(Message Passing)이란 주로 분산 메모리를 가진 병렬컴퓨터에서 정보를 교환하기 위한 수단으로서 사용되며, 지난 15년동안 병렬컴퓨터에 이용되면서 많은 진보가 이루어졌다. 그러나 컴퓨터회사들이 자신들의 컴퓨터에 각기 다른 통신수단을 사용하였기 때문에 프로그램을 다른 컴퓨터에 이식할 수 없었다. 90년대 초반에 Express [1], Zipcode [2, 3], PVM [4, 5], Chameleon [6], PICL [7]과 같은 아주 효율적이면서도 이식가능한 메시지 전달 시스템이 개발되기 시작하였다. 이러한 다양한 많은 노력들을 통합하여, 메시지 전달의 기본이 되는 루틴들의 의미와 표현을 정의하여, 사용되는 거의 모든 컴퓨터에서 이용할 수 있는 표준을 제정하고자, 1992년 4월 미국 버지니아주 윌리엄스버그에서 첫번째 워크샵이 열렸다. MPI(Message Passing Interface) 포름은 컴퓨터회사와 대학교, 연구소에 소속된 60여명의 연구원들로 구성되었으며, 그 노력의 결과로써 1994년 6월에 버전 1.0이, 그리고 1995년 6월에 버전 1.1이 발표되었다.

MPI의 초기 목표들은 다음과 같았다.

- 응용 프로그램을 위한 인터페이스를 정의 (컴파일러 혹은 시스템으로 구현하여야 하는 라이브 러리를 배제하였음)
- 효율적인 통신(메모리간의 불필요한 복사과정을 생략하고, 계산과 통신을 효과적으로 중첩함)
- 이기종 컴퓨터 환경 지원
- 인터페이스로서 C와 Fortran 77 언어로 구현

MPI에는 일대일(point-to-point) 통신과 집합(collective) 통신과 함께, 병렬 라이브러리를 지원하기 위한 여러 가지 개념들을-통신문맥(context), 통신자(communiator), 통신그룹(group), 가상 토폴로지 등을-정의하였다. MPI에서 통신문맥이란 메시지들을 구별하는 부가적인 태그로서, 다른 라이브러리가 서로 다른 통신문맥을 사용함으로써 그 라이브러리 내부의 통신을 안전하게 보장해 준다. 통신자는 통신을 수행하기 위한 통신 문맥을 나타내며, 또한 통신문맥을 공유하는 프로세스들의 집합을 나타낸다. 통신그룹은 한 통신자에서 통신에 관여하는 프로세스들의 순서적인 집합으로, 통신자에 따라서 다르게 정의된다. MPI-COMM-WORLD는 미리 정의된 통신자로 MPI가 시스템으로부터 부여한다. MPI에서 통신자에는 내부통신자(intra-communicator)와 상호통신자(inter-communicator)가 있다. 내부통신자는 하나의 통신그룹 안에서 정의되며, 그 통신자 안에서 일대일 통신, 집합 통신을 수행할 수 있다. 그리고 상호통신자는 서로 중복되지 않는 두 개의 통신그룹 사이에서의 통신을 정의하였다.

한편 MPI의 첫 표준안이 완성되어 공표된 직후, 1995년 3월부터 MPI 포럼은 공표된 표준의 오류를 정정하고 그 표준의 확장을 논의하기 시작하였다. MPI 포럼에서는 MPI의 표준안을 제정하면서, 논란이 된 많은 새로운 개념들을 좀더 확장된 표준에 포함시키고자 하였

*종신회원
**학생회원

다. 원래의 MPI 표준을 MPI-1이라 하고, 새로 제정된 MPI 표준을 MPI-2로 명명하였다. MPI-2는 기존의 MPI-1을 대체하는 새로운 표준이 아니라 MPI-1에 많은 기능들이 추가된 확장된 표준이다.

MPI-2의 초안은 96년 11월에 열린 Supercomputing '96에 MPI-1의 오류를 정정한 버전 1.2와 함께 제출되었으며, 독자들이 본고를 읽고 있을 무렵이면 공식적으로 MPI-2가 공표되었을 것이다(1997년 6월에 공표될 예정으로 있다). MPI-2가 공식적으로 공표되기 전에 본고를 준비하는 것이 약간 성급한 것 같지만, 대체적인 내용은 Supercomputing '96에 제출된 초안과 크게 다르지 않을 것이다. MPI-2에서 새로 추가된 대표적인 내용을 보게 되면 아래와 같다.

1. 프로세스 생성 및 관리(Process Creation and Management)
2. 일방적인 통신(One-sided Communication)
3. 확장된 집합 통신(Extended Collective Communication)
4. 외부 인터페이스(External Interface)
5. MPI-IO(Parallel IO)

다음의 각 장에서는 MPI-2에서 다루는 새로운 기능 및 개념들을 기술한다.

2. 프로세스 생성 및 관리

MPI-1은 병렬 프로그램에서 프로세스들이 서로간에 통신할 수 있는 인터페이스를 제공한다. MPI-1은 어떻게 프로세스들이 생성되고 통신을 설정하는지 정의하고 있지 않다. 더욱이 MPI-1 프로그램들은 정적(static)으로, 프로그램이 시작된 후에 프로세스를 추가하거나 삭제할 수 없다. 그러나 MPI-2에서는 프로그램이 시작된 후에도 프로세스를 추가하고 관리할 수 있다. 이 기능은 PVM의 영향으로 구현되었으며, PVM과는 달리 자원(resource)을 제어하지 않으며, 단지 프로세스들을 관리한다. 자원들에 관한 제어는 MPI-2의 이식성을 제한하기 때문에 지원하지 않는다. MPI-2에서는 프로세스 관리에 대한 기능을 지원하므로, PVM을 이용한 기존의 많은 프로그램들을 MPI를 사용하는 시스템으로 이식시키기가 용이하다.

MPI-2는 MPI 프로그램이 시작한 이후에도 프로세스들을 생성(spawn)하고 제거할 수 있다. 그리고 새로이 생성된 프로세스들과 실행중인 프로그램과의 통신을, 그리고 실행중인 두 프로그램사이의 통신을 설정할 수 있다.

MPI-SPAWN은 새로운 프로세스들을 생성하고 그들간에 통신을 설정하며, 상호통신자를 되돌려 준다. 즉 원하는 프로그램을 실행하는 프로세스를 원하는 갯수만큼 생성시킨다. 생성된 프로세스들을 자식(child) 프로세스라고하고, 자식 프로세스들은 그들 자신만의 내부통신자인 MPI-COMM-WORLD를 갖는다. 생성된 자식 프로세스들은 MPI-INIT을 호출하여 그들 부모 프로세스와 집합 통신을 수행할 수 있는데, 집합 통신은 상호통신자를 통해 이루어진다. 부모 프로세스에서 MPI-SPAWN를 호출한 후 상호통신자를 되돌려 받으며, 자식 프로세스들은 MPI-COMM-PARENT을 호출하여 그 상호통신자를 알아낼 수 있다.

PVM 프로그램처럼 MPI에서도, 먼저 하나의 프로세스를 시작시킨 후에, 그 프로세스가 MPI-SPAWN을 이용하여 나머지 프로세스들을 생성시키는, 정적인 SPMD 혹은 MPMD 응용프로그램이 가능하게 되었다. 그러나 가능하면 모든 프로세스들을 한번에 생성하는 것이 더 효율적이다. MPI-SPAWN-MULTIPLE은 MPI-SPAWN을 여러번 한 것과 동일한 효과를 얻게 되지만, 완전히 같지는 않다. MPI-SPAWN을 여러번 호출하면 자식 프로세스들이 서로 다른 MPI-COMM-WORLD를 갖지만, MPI-SPAWN-MULTIPLE에서 생성되는 자식 프로세스들은 단 하나의 MPI-COMM-WORLD를 갖는다. 그리고 MPI-SPAWN-MULTIPLE을 이용하면 더 효율적이며, 또한 동시에 생성된 프로세스간 통신이 따로 생성된 프로세스간의 통신보다 빠르다. 한편 프로세스들을 생성하는 것은 상당히 시간이 많이 걸리는 과정이다. 따라서 MPI는 프로세스를 생성하는 동안 그 프로세스가 유용한 작업을 할 수 있는 비지연(non-blocking)형식의 루틴들(MPI-

ISPAWN과 MPI-ISPAWN-MULTIPLE)이 있다.

한편, MPI-1에서는 같은 통신자를 공유하고 있지 않는 두 개의 MPI 프로세스 집합사이에 서로간 통신을 할 수 없었다. 그러나 두 응용 프로그램이 각기 다르게 시작되었거나, 가시화 도구를 수행중인 프로세스에 연결시키고자 할 때, 그리고 서버가 여러 클라이언트들로부터 연결을 요청받을 때, 서로간에 통신을 필요로 한다. 같은 통신자를 공유하고 있지 않는 두 집합의 프로세스들을 연결시키는 것은 집합적이고 비대칭적인 과정이다. 한 집합의 프로세스들이 다른 집합의 프로세스들에 연결시키고자 한다면, 요청하는 프로세스들을 클라이언트(client), 요청받는 프로세스들을 서버(server)라고 정의한다. MPI-2에서 클라이언트-서버라는 이름을 사용하지만, 이것은 전형적인 클라이언트-서버 시스템은 아니다.

서버는 두개의 루틴을 사용하여 그 구실을 한다. 서버는 MPI-PORT-OPEN를 호출하여 연결될 포트를 설정한 후에 MPI-PORT-ACCEPT를 호출하여 클라이언트와 연결시키며, 연결을 해지하기 위한 MPI-PORT-CLOSE 루틴이 있다. 클라이언트는 MPI-CONNECT를 사용하여 서버의 지정된 포트와 연결할 수 있다. 한편 포트명은 서버가 연결될 수 있는 네트웍 주소로서 시스템이 제공한다. 서버는 MPI-NAME-PUBLISH 루틴을 사용하여 포트명을 알기 쉬운 서비스명으로 대체시키며, 클라이언트는 MPI-NAME-GET 루틴을 사용하여 난해한 포트명 대신 서비스명을 얻어 사용할 수 있다. 서버와 클라이언트에서 MPI-ACCEPT와 MPI-CONNECT가 각각 수행되면 그들은 같은 상호통신자를 가지게 되고, 이 상호통신자를 사용하여 서로간에 통신할 수 있다. 한편 비지연을 지원하는 MPI-IACCEPT, MPI-ICONNECT 루틴들이 있다. 그림 1은 클라이언트와 서버를 연결하는 간단한 프로그램 예를 보여준다. 중요한 MPI-2의 루틴들을 진하게 나타내었다.

서버

```
MPI_Port_open(NULL, &port_name),
MPI_Name_publish(port_name, "ocean");

MPI_Accept(port_name, NULL, 0,
           MPI_COMM_SELF, &intercomm),
if('MPI_NAMES_ARE_PUBLISHED) {
    if(batch_mode) error(FATAL,
        "can't transfer port name");
}
MPI_Name_delete(port_name, "ocean");
```

클라이언트

```
if(!MPI_NAMES_ARE_PUBLISHED) {
    if(batch_mode) error(FATAL,
        "can't transfer port name"),
    printf("low quality MPI
        implementation encountered \\
        enter port name:");
    gets(port_name);
}
else MPI_NAME_GET("ocean", port_name),
MPI_Connect(port_name, NULL, 0,
            MPI_COMM_SELF, &intercomm);
```

그림 1 간단한 클라이언트와 서버 프로그램

3. 일방적인 통신 (One-Sided Communications)

RMA(Remote Memory Access)는 보내는 쪽이나 받는 쪽에서, 한 프로세스가 모든 통신 파라미터들을 정의할 수 있다. 이 방식을 이용하면, 각 프로세스는 다른 프로세스가 가지고 있는 데이타에 접근할 수 있고, 또 그 데이타를 변경시킬 수 있다. 하지만 프로세스들은 자신의 메모리에 있는 어떤 데이타를 다른 프로세스가 가져가고 혹은 변경시키는지 알지 못할 뿐 아니라, 어떤 프로세스가 그 일을 하는지조차 모를 수 있다. 일반적인 메시지를 주고 받는(message passing) 방식에서는 보내는 쪽과 받는 쪽이 서로 존재하여야 통신이 가능하지만, RMA에서는 통신 파라미터들이 한 쪽에서만 정의된다. 이를 위해 PUT과 GET 호출을 사용한다. 그림 2에서 보듯이, PUT은 한 프로세스가 다른 프로세스의 메모리에 값을 보내고, GET은 한 프로세스가 다른 프로세스의 메모리에 있는 값을 얻는 것이다. 이 모델은 공유 메모리(shared memory)를 대치하는 것이 아니고 여전히 분산 메모리 기반이며 전통적인

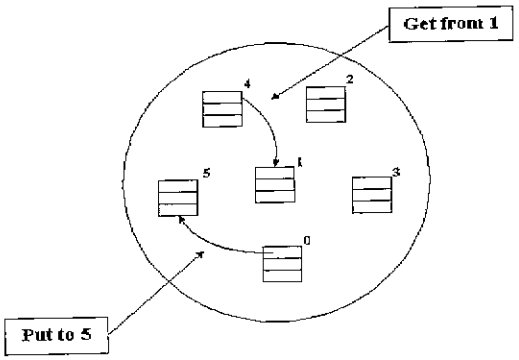

그림 2 GET와 PUT의 원격 메모리 접근

메시지 전달보다 하드웨어의 이용면에서 더 효율적이다.

메시지를 주고 받는 통신에서는 보내는 쪽에서 받는 쪽으로 데이타를 전달하고, 이와 더불어 동기화하는 효과가 있지만, RMA에서는 이 두가지 기능을 분리하였다. RMA에는 MPI-PUT, MPI-GET, MPI-ACCUMULATE의 세 개의 통신 루틴이 있으며, 동기화하기 위한 여러 루틴들이 있다. MPI-2에서는 루틴을 호출하는 프로세스를 오리진(origin)으로, 그리고 메모리에 접근한 다른 프로세스를 타겟(target)으로 나타내는데, PUT에서는 source=origin이고 destination=target이지만, GET에서는 그 반대가 된다.

MPI-PUT은 호출하는 프로세스의 메모리로부터(이것이 오리진이다) 타겟 메모리로 데이타를 전달하며, MPI-GET은 타겟 메모리에서 호출 프로세스의 메모리로 데이타를 가져온다. MPI-ACCUMULATE은 호출하는 프로세스의 메모리에서 보내지는 데이타로 타겟 프로세스 메모리에 있는 데이타를 단순히 교체하는 것이 아니고, 타겟 프로세스 메모리에 있는 데이타와 사용자가 정의하는 연산을 한다. 이 루틴들은 비지연으로 동작한다. 즉 루틴이 호출되면 메시지 전달이 시작되고, 루틴은 즉시 프로그램으로 리턴되어 그 다음 루틴들을 수행하지만, 메시지 전달은 끝까지 계속 진행된다. 오리진과 타겟에서 메시지 전달이 끝나면, 호출 프로세스는 동기화루틴을 호출하여 작업 종료를 확인한다.

일방적인 통신을 지원하기 위하여 초기화 과정이 필요하다. 즉, 원격 프로세스가 접근할 수 있는 메모리 윈도우를 각 프로세스에 설정해주어야 한다. 이러한 기능을 위해 통신자에 속한 모든 프로세스들이 MPI-WIN-INIT을 집합적으로 호출하며, 프로세스들은 RMA 연산을 수행하는데 필요한 윈도우 객체를 되돌려 받는다. 한편 MPI-WIN-FREE로 설정된 윈도우 객체가 해지된다.

MPI에는 동기화하는 3가지 방식이 있다. 첫째로 MPI-WIN-BARRIER는 단순하고 집합적인 동기화 루틴으로, 모든 프로세스들이 연산과정과 통신과정을 전체적으로 번갈아 수행하는 병렬 연산에서 주로 사용된다. 이 방식은 프로세스들이 통신하는 통로가 빈번하게 변경되거나, 혹은 각 프로세스가 많은 다른 프로세스들과 통신을 필요로하는 곳에서 유용하게 사용될 수 있다. 그리고 MPI-WIN-START, MPI-WIN-COMPLETE, MPI-WIN-POST, MPI-WIN-WAIT의 4개 함수는 동기화를 최소한으로 제한하는데 사용한다. 즉 통신하는 프로세스들의 서로간에 동기화하거나, 프로세스내의 메모리 접근과 비교하여 순차적으로 RMA의 메모리에 접근하기위해 동기화를 필요로할 때 이들 함수를 사용한다. 이 방식은 각 프로세스가 단지 몇 개의 프로세스들과 통신하거나, 혹은 통신하는 통로가 거의 고정되어 있을 경우에 효과적이다. 마지막으로 MPI-WIN-LOCK과 MPI-WIN-UNLOCK은 공유되고 배타적인 록(lock) 방식이다. 록을 이용한 동기화는 MPI 호출을 이용하여 공유메모리 모델을 에뮬레이션하는데 유용하다.

4. 확장된 집합 통신

MPI-2에서는 MPI-1에서 지원하지 않았던 내부 통신자에서의 비지연 기능을 제공하며, 내부 통신자와 상호 통신자에서의 영속적인(persistent) 집합 통신 기능을 지원한다. 영속적인 통신이란,병렬 프로그램의 내부 루프에서 똑같은 파라미터를 가진 통신이 반복적으로 호출될 때, 그 통신 파라미터들을 하나의 영속적인 통신 요청으로 묶어놓고, 데이타를 전송시키기 위해 반복적으로 그 요청을 사용하므로써, 그러한 통신을 효과적으로 수행시키는 것이다. 그리고 MPI-2는 상호통신자의 생성을 위한 추가적인 루틴, MPI-1의 상호통신자에 대한 집합 통신을 확장하였다. 또한 MPI-1에서은 집합 통신을 위해 수신과 송신용 버퍼를 분리하지만, MPI-2에서는 내부통신자의 집합 연산을 위해 'in-place' 버퍼를 정의하여, 수신과 송신 버퍼를 하나로 쓸 수 있도록 한다.

MPI-1은 상호통신자를 생성하기 위한 두 개의 루틴을 제공한다. MPI-INTERCOMM-CREATE는 두 개의 내부통신자로 부터 하나의 상호통신자를 생성하는 루틴이고, MPI-

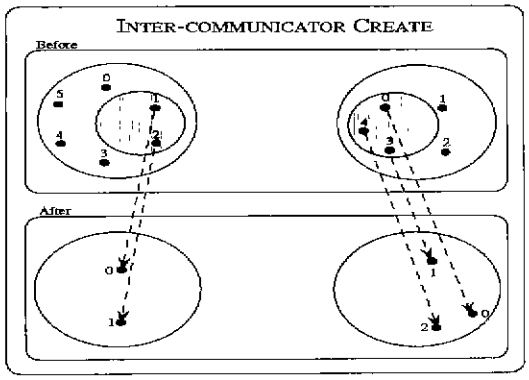

그림 3 MPI-COMM-CREATE를 이용한 상호통신자의 생성

COMM-DUP는 한 내부통신자를 복사하는 루틴이다. MPI-2에서는 MPI-COMM-CREATE를 사용하여 기존의 상호통신자를 이용하여 새로운 상호통신자를 만들 수 있도록 상호통신자의 생성을 확장하였다. 그림 3은 기존의 상호통신자에서 MPI-COMM-CREATE를 사용하여 새로운 상호통신자를 생성시키는 것을 보여주고 있다. 그림에서 Before는 MPI-COMM-CREATE이전의 상호통신자를 보이고 있다. 이 상호통신자는 두 개의 통신자로 이루어져 있는데, 왼쪽의 통신자는 6개의 프로세스로 이루어져 있으며, 오른쪽의 통신자는 5개의 프로세스로 이루어져 있다. 여기서 MPI-COMM-CREATE를 사용하여 Before의 각 통신자에서 음영된 프로세스들로만 이루어진 새로운 상호통신자를 만들 수 있다. 즉, After는 각각 2개와 3개의 프로세스로 이루어진 통신자들의 상호통신자를 보이고 있다. 또한 MPI-2는 기존의 상호통신자로부터 특정한 값으로 여러개의 분리된 상호통신자로 분할할 수 있게 하는 MPI-COMM-SPLIT를 제공한다.

MPI-2는 다음과 같은 집합 연산을 위한 함수를 정의하고 있다.

- MPI-BCAST,
- MPI-GATHER, MPI-GATHERV,
- MPI-SCATTER, MPI-SCATTERV,
- MPI-ALLGATHER, MPI-ALLGATHERV,
- MPI-ALLTOALL, MPI-ALLTOALLV, MPI-ALLTOALLW
- MPI-REDUCE, MPI-ALLREDUCE,
- MPI-REDUCE-SCATTER,
- MPI-BARRIER.

각 함수의 인자 리스트 및 기능은 상응하는 MPI-1의 연산들과 유사하다. 차이점은 MPI-2는 집합 상호통신자 연산을 위해 전이중(full-duplex) 방식을 선택하였고, 또한 대부분의 집합 통신은 다음과 같은 두 개의 특성을 포함한다는 것이다.

1) 집합 연산이 'in place'로 호출될 수 있다. 이것을 제공하므로써 수신 버퍼를 송신 버퍼로 사용할 수 있으며(즉, 송신과 수신버퍼를 하나로 사용할 수 있음), 또한 MPI와 사용자에 의한 불필요한 메모리 이동을 피할 수 있다.
2) 집합 연산이 상호 통신자에도 적용될 수 있다.

MPI-ALLTOALLW 함수는 MPI-1에서 정의되어 있지 않는 이름이다. 이 함수는 수신과 송신 버퍼에서의 블록 변위를 바이트 단위로 기술할 수 있게하여 최대한의 유연성을 제공하기위한 것이다.

5. 외부 인터페이스

MPI-2에서는 외부 인터페이스를 지원하기 위한 많은 기능들이 제공된다. 먼저 일반적인 요청을 생성할 수 있는 기능이 있고, 오류 코드와 클래스를 추가하는 기능, 데이타형을 복호화하고 전송하고 저장하는 기능, 스레드를 다루는 방법 등이 있다. 일반화된 요청, 데이타형의 접근, 스레드에 대해서만 간략하게 기술한다.

복잡한 집합 연산이나 I/O처럼 MPI에서 지원하지 않는 연산을 사용자가 요청할 수 있도록 하는데 MPI-2를 확장하는 목적이 있다. 또한 지연, 비지연, 그리고 영속적인 일반화된 요청을 생성할 수 있다. 일반화된 요청에 대한 기능은 그것과 관련된 콜백(callback) 함수로 정의된다. 콜백 함수는 특정한 시간에 호출되어 주어진 임무를 수행하게 된다. 정의된 콜백 함수들은 일반화된 요청의 시작, 진행 및 완료 감지, 완료, 취소로 분류되고, 각각 start-fn,

push-fn, complete-fn, cancel-fn으로 나타낸다. 일반화된 요청은 MPI-META-REQUEST-CREATE로 생성되고, MPI-META-REQUEST-FREE로 해지된다. 영속적인 일반화된 요청의 생성과 해제를 위해서는 MPI-META-REQUEST-INIT과 MPI-REQUEST-FREE를 사용한다. 지연과 비지연 일반화된 요청은 각각 MPI-DO와 MPI-START{ALL}로 시작된다. 일반화된 요청이 시작되면, start-fn 콜백 함수가 호출되고, 그 요청의 진행과 완료 감지를 위해 push-fn을 사용한다. MPI는 일반화된 요청의 완료를 complete-fn을 사용하여 알아내고, 사용자 프로그램에서는 MPI-{TEST | WAIT}{ANY | SOME | ALL}을 사용하여 알 수 있다. 일반화된 요청이 완료되면 cancel-fn 콜백이 호출된다.

MPI-1은 사용자가 메모리에 있는 임의의 데이타 구도를 설정할 수 있도록 데이타형식의 객체를 제공한다. 한번 데이타형식에 들어간 구도 정보는 데이타형으로부터 복호화될 필요는 없다. 그러나 모호한 데이타형 객체의 구도에 대한 정보에 접근하는 것을 필요로 할 때가 종종 있다. 첫 번째로, MPI-PUT과 MPI-GET에서와 같이 데이타형이 프로세스간에 전달될 필요가 있을 경우, 두 번째로, 이식가능한 MPI 프로파일러나, 트레이서, 디버거를 구현하는데 데이타형이 사용될 때이다. 세번째로는 메모리가 아닌 다른 데서 데이타 구도를 설정하려고 하는 경우이다. MPI-2는 데이타형에 접근할 수 있는 여러 방법을 제공한다. 먼저 저수준 데이타형을 접근할 수 있는 방법으로는 MPI-GET-TYPE-ENVELOP, MPI-GET-TYPE-CONTENTS, MPI-TYPE-COMPARE가 있다. MPI-GET-TYPE-ENVELOP은 주어진 데이타형의 형태와 이것을 구성하는 원소의 갯수를 알려준다. MPI-GET-TYPE-CONTENTS는 주어진 데이타형의 다양한 정보를 알려준다. MPI-TYPE-COMPARE는 기본적 데이타형에만 사용되고, 두 개의 데이타형을 받아들여 이 둘이 같은 데이타형인지를 알려준다.

또한 데이타형을 부호화하고 복호화하는 방법이 있다. 이것은 한 프로세스가 다른 프로세스와 데이타형을 통신할 수 있도록 한다. 두 데이타형이 동치라고 하는 것은, 두 데이타형이 데이타형 생성자를 같은 순서로 호출하여 구성되었다는 것을 의미한다. 데이타형 통신은 한 프로세스에서 정의된 데이타형과 동치인 데이타형을 다른 프로세스에서도 생성될 수 있도록 해야 한다. 이를 위해서는 한 프로세스가 데이타형을 부호화하여 다른 프로세스에게 보낼 수 있어야 하고, 그것을 받은 프로세스는 그 데이타형을 복호화하여 사용할 수 있어야 한다. 이 기능을 위해 MPI-2는 MPI-TYPE-ENCODE, MPI-TYPE-ENCODE-SIZE, MPI-TYPE-DECODE를 제공한다. 여기서 MPI-TYPE-ENCODE는 주어진 데이타형을 부호화하고, MPI-TYPE-ENCODE-SIZE는 데이타형을 부호화했을 때의 크기를 계산하며, MPI-TYPE-DECODE는 부호화된 데이타형을 복호화한다.

마지막으로 MPI 호출과 선점형 Posix 스레드간의 상호작용에 대해서 설명한다. 스레드를 지원하기위한 다양한 구현방법이 있겠지만, MPI는 스레드를 지원하지 않는 환경에서도 구현될 수 있다. 따라서 모든 MPI의 구현이 명시된대로 작용하지는 않아도 된다. 이러한 요구조건은 스레드에 순응적인 구현방법에 적용된다. 스레드에 순응적인 구현에서의 MPI 프로세스는 Posix 프로세스이다. 이 프로세스는 스레드를 생성하고 동기화하기 위해 Posix의 pthread를 사용한다. 각 스레드는 MPI 호출을 할 수 있지만, 스레드는 분리되어 다룰 수 없고, 송신과 수신에 있어서의 통신번호(rank)는 스레드가 아닌 프로세스로 식별된다. 한 프로세스로 보내진 메시지는 그 프로세스의 어떤 스레드도 받을 수 있다.

이제 MPI-2에서 스레드를 다루기위한 함수들을 살펴보자. 먼저 스레드 초기화와 관련된 것은 MPI-THREAD-INIT와 MPI-THREAD-INITIALIZED, MPI-THREAD-FINALIZE가 있다. 스레드가 MPI 호출을 사용하기 위해서는 MPI에 등록해야 하는데 이것을 수행하는 것이 MPI-THREAD-INIT이고, 스레드가 MPI-THREAD-INIT를 호출하였는지를 검사하는 것이 MPI-THREAD-INITIALI-

ZED이며, 스레드를 MPI에서 제거하는 것이 MPI-THREAD-FINALIZE이다. 스레드가 MPI-THREAD-FINALIZE를 수행한 이후에는 MPI 호출을 사용할 수 없다. 많은 경우에 통신이 완료되면 특정한 처리 루틴이 수행되기를 요구한다. 이를 위해 MPI-2는 MPI-POST-HANDLER를 사용하여 사용자가 작성한 처리 루틴을 수행할 수 있도록 해 준다. 이 함수는 수행될 요청과 처리루틴을 인자로 갖는데, 이 요청이 완료되는 즉시 처리 루틴이 수행된다. 이 함수는 또한 비지연 통신이나 타스크 생성과 같은 새로운 비지연 호출을 구현하는데 사용될 수 있고, 일반화된 요청 안에서 사용될 수도 있다.

6. MPI-IO

여러가지 병렬 파일 시스템은 서로 다른 인터페이스를 지원하며, 표준의 이식가능한 인터페이스가 없기 때문에, 병렬 파일 시스템을 지원하는 이식가능한 병렬 프로그램을 만들기가 불가능하다.

UNIX는 널리 사용할 수 있는 이식가능한 파일 시스템의 모델을 제시하지만, 병렬 I/O에 필요한 이식성과 적합성을 전형적인 UNIX로 얻을 수 없다.

효율적인 I/O를 구성하기위하여는 그룹핑[8], 집합 버퍼링[9, 10, 11, 12, 13], 디스크성(disk-directed) I/O [14]와 같이 많은 최적화 노력이 요구되지만, 그러한 것들이 프로세스들 간에 파일 데이타의 분산을 나타내는 상위 인터페이스를 지원하고, 프로세스 메모리와 파일 사이에서의 전역적인 자료 구조들을 완전히 변환시킬 수 있는 집합적인 인터페이스를 제공한다고 하여도, 병렬 입출력 환경의 일부분으로만 구현될 수 있다. 더욱이 비동기적인 I/O, 일정한 간격으로 떨어져있는 자료의 요청, 디스크와 같은 저장 매체에서의 실제적인 파일 배치에 대한 제어 등을 지원함으로써 더 효율적인 I/O 환경을 구축할 수 있다.

MPI-2에서는 방송(broadcast), 축약(reduction), 배포(scatter), 모음(gather)과 같은 공유된 파일들에 접근하기 위한 일반적인 양식을 정의하기보다는 유도 데이타 형식(derived datatype)을 통한 데이타 분산을 나타낼 수 있는 새로운 방식을 지원한다. 유도 데이타 형식이란 사용자가 정의한 데이타 형식으로, 비연속적인 데이타들을 정수, 실수 등을 구분하는 기본적인 데이타 양식과 함께 데이타간의 간격을 나타내는 정수값으로 표현된다. 기존의 제한된 미리 정의된 접근 양식과 비교하여, MPI-2에서 이같은 새로운 방식은 유연성과 표현성을 쉽게 지원한다.

MPI-2 I/O에서의 뷰(view)란 기본적인 데이타 형식으로 정렬된 집합으로, 열려진 파일로 부터 접근할 수 있는 현재의 데이타 집합을 정의한다. 각 프로세스는 간격(displacement), 기본 데이타 형식(etype), 파일 형식(filetype)으로 정의되는 그 자신의 파일의 뷰를 갖고 있다. 이 뷰는 실행되는 동안에도 사용자가 변경시킬 수 있으며, 기본적인 뷰는 간격이 0인 일렬로 연결된 선형 바이트 스트림이다. 프로세스들의 그룹은 배포와 모음 형식과 같은 전체적인 데이타 분산을 얻기위한 다른 뷰들을 사용할 수 있다. 그림 4에서 프로세스들의 그룹들은 방송, 축약, 배포, 모음과 같은 방법을 사용하여 전체적인 데이타 분포를 얻는 보충의 view들을 사용할 수 있음을 보여준다.

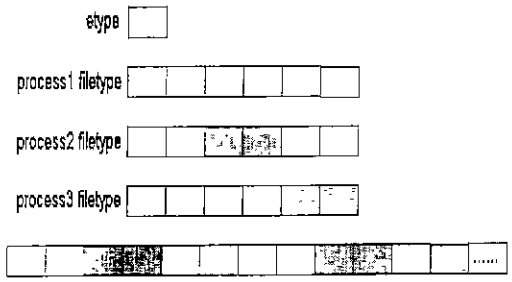

그림 4 병렬 프로세스들 사이에서 파일 분포

MPI-2에서 제공하는 병렬 I/O를 위한 루틴들을 알아보자. MPI-OPEN 루틴은 지정된 통신자 그룹안에서 있는 모든 프로세스들이 같이 사용하는 파일을 연다. MPI-OPEN은 집합 연산으로, 통신자 그룹안에서 있는 모든 프로세스들이 함께 같은 파라미터 값을 가지고 호출한다. MPI-OPEN을 호출한 후 되돌아오는 파일 핸들러 fh는 MPI-CLOSE를 사용하여 그

표 1 MPI-2의 데이타 접근 루틴

positioning	synchronism	coordination	
		non-collective	collective
explicit offsets	blocking (synchronous)	MPI-Read-explicit MPI-Write-explicit	MPI-Read-explicit-all MPI-Write-explicit-all
	nonblocking (asnynchronous)	MPI-Iread-explicit MPI-Iwrite-explicit	MPI-Iread-explicit-all MPI-Iwrite-explicit-all
individual file pointers	blocking (synchronous)	MPI-Read MPI-Write	MPI-Read-all MPI-Write-all
	nonblocking (asynchronous)	MPI-Iread MPI-Iwrite	MPI-Iread-all MPI-Write-all
shared file pointer	blocking (synchronous)	MPI-Read-shared MPI-Write-shared	MPI-Read-shared-ordered MPI-Write-shared-ordered
	nonblocking (asynchronous)	MPI-Iread-shared MPI-Iwrite-shared	MPI-Iread-shared-ordered MPI-Iwrite-shared-ordered

파일이 닫힐 때까지 계속하여 파일에 접근하는 데 사용할 수 있다.

MPI-FILE-SET-VIEW 루틴은 파일에 있는 데이타에 대한 프로세스의 뷰를 변경시킬 뿐만 아니라, 독자적인 파일 포인터들과 공유하고 있는 파일 포인터들을 0으로 재설정시킨다. MPI-FILE-SET-VIEW 루틴은 집합적인 함수로서 그룹안에 있는 모든 함수들이 같은 파라미터 값을 가지고 호출하여야 한다. MPI-FILE-GET-VIEW 루틴은 파일에 있는 데이타에 대한 프로세스의 뷰를 알려준다.

한편 데이타는 파일과 프로세스 사이에서 읽거나 쓰는 함수를 호출함으로써 이동된다. 데이타 접근에는 성질이 다른 3가지 방식이 있다. 위치잡기(positioning) 방식에는 명시적인 오프셋(explicit offset positioning)을 이용하거나 개별적인 파일 포인터, 혹은 공유된 파일 포인터를 사용한다. 동기화 방식에는 지연 동기화 및 비지연 동기화가 있으며, 협력(coordination) 방식에는 집합적 협력과 비집합적 협력이 있다. 이러한 데이타 접근 방식에 대한 모든 조합이 가능하다. 데이타 접근 루틴들은 표 1에 요약되어 있다.

7. 맺는말

MPI-1은 이기종 컴퓨터간의 메시지 교환 방법으로 이식성이 뛰어나고 성능이 좋아서 표준으로 자리잡고 있다. 그러나 MPI-1은 프로세스 관리 기능과 입출력 기능 등을 포함하지 않고 있고, C와 Fortran 언어로만의 바인딩을 제공하고 있으며, 집합통신에 제약이 있는 등 여러 가지 제한점을 가지고 있다. 따라서 MPI-1의 확장된 기능을 갖는 MPI-2를 제정하기에 이르렀다.

본 고에서는 MPI-2의 새로운 주요 기능 및 개념들을 간략하게 소개하였다. 소개된 내용은 프로세스 생성 및 관리, 일방적인 통신, MPI-IO, 외부 인터페이스, 확장된 집합 통신 등으로 MPI를 더욱 효율적으로 사용할 수 있게 하는 기능들이다. 이 외에도 MPI-2에서는 MPI를 위한 C++ 언어 인터페이스를 제공한다.

MPI-2 이후에 추가될 기능으로 현재 논의되고 있는 실시간 MPI, 독립 프로세스 생성, 확장된 스레드 관리, MPI 핸들에서의 캐싱 등이 있다. 이에 대한 자세한 내용 및 MPI-1과 MPI-2에 대한 자료는 각각 http://www.mcs.anl.gov/mpi/index.html와 http://www.mcs.anl.gov/Projects/mpi/ mpi2/mpi2.html에서 찾을 수 있다.

참고문헌

[1] Parasoft Corporation, Pasadena, CA. *Express User's Guide*, version 3.2.5 edition, 1992.

[2] A. Skjellum and A. Leung. Zipcode : a portable ulticomputer communication library atop the reactive kernel. In D. W. Walker and Q. F. Stout, editors, *Proceedings of the Fifth Distributed Memory Concurrent Computing Conference,* pages 767-776. IEEE Press, 1990.

[3] A. Skjellum, S. Smith, C. Stil, A. Leung, and M. Morari. The Zipcode message passing system. Technical report, Lawrence Livermore National Laboratory, September 1992.

[4] A. Beguelin, J. Dongarra, A. Geist, R. Manchek, and V. Sunderam. Visualization and debugging in a heterogeneous environment. *IEEE Computer,* 26(6) : 88-95, June 1993.

[5] J. Dongarra, A. Geist, R. Manchek, and V. Sunderam. Integrated PVM framework supports heterogeneous network computing. *Computers in Physics,* 7(2) : 166-75, April 1993.

[6] William D. Gropp and Barry Smith. Chameleon parallel programming tools users manual. Technical Report ANL-93/23, Argonne National Laboratory, March 1993.

[7] G. A. Geist, M. T. Heath, B. W. Peyton, and P. H. Worley. A user's guide to PICL : a portable instrumented communication library. Technical Report TM-11616, Oak Ridge National Laboratory, October 1990.

[8] Bill Nitzberg. Performance of the iPSC /860 Concurrent File System. Technical Report RND-92-020, NAS Systems Division, NASA Ames, December 1992.

[9] Rajesh Bordawekar, Juan Miguel del Rosario, and Alok Choudhary. Design and evaluation of primitives for parallel I/O. *In Proceedings of Supercomputing '93,* pages 452-461, 1993.

[10] Juan Miguel del Rosario, Rajesh Bordawekar, and Alok Choudhary. Improved parallel I/O via a two-phase run-time access strategy. In *IPPS '93 Workshop on Input/Outpul in Parallel Computer Systems,* pages 56-70, 1993. Also published in Computer Architecture News 21(5), December 1993, pages 31-38.

[11] William J. Nitzberg. *Collective Parallel I/O.* PhD thesis, Department of Computer and Information Science, University of Oregon, December 1995.

[12] K. E. Seamons, Y. Chen, P. Jones, J. Jozwiak, and M. Winslett. Server-directed collective I/O in Panda. In *Proceedings of Supercomputing '95,* December 1995.

[13] Rajeev Thakur and Alok Choudhary. An extended two-phase method for accessing sections of out-of-core arrays. Technical Report CACR-103. Scalable I/O Initiative, Center for Advanced Computing Research, Caltch, June 1995. Revised November 1995.

[14] David Kotz. Disk-directed I/O for MIMD multiprocessors. In *Proceedings of the 1994 Symposium on Operating Systems Design and Implemeniation,* pages 61-74, November 1994. Updated as Dartmouth TR PCS-TR94-226 on Nobemberg 8, 1994.

최 재 영

1984 서울대학교 제어계측공학과 학사
1986 미국 남가주대학교 전기공학과 석사(컴퓨터공학)
1991 미국 코넬대학교 전기공학부 박사(컴퓨터공학)
1992~1994 미국 국립 오크리지연구소 연구원
1994~1995 미국 테네시 주립대학교 연구교수
1995~현재 숭실대학교 정보과학대학 컴퓨터학부 조교수
관심분야 : 병렬/분산처리, 병렬알고리즘, 초고속계산론, 시스템 소프트웨어

김 명 호

1984 숭실대학교 전자계산학과 학사
1991 포항공과대학교 전자계산학과 석사
1995 포항공과대학교 전자계산학과 박사, 한국전자통신연구원 선임연구원
1995~현재 숭실대학교 정보과학대학 컴퓨터학부 전임강사
관심분야 : 병렬처리, 병렬알고리즘, 초고속계산론, 시스템 소프트웨어

김 정 훈

1996 수원대학교 전자계산학과 학사
1996~현재 숭실대학교 대학원 전자계산학과 석사
관심분야 : 병렬/분산처리, 망관리, 시스템 소프트웨어

● 제24회 전국전문대학 전산관련학과 교수세미나 ●

- 일 자 : 1997년 7월 9~11일
- 장 소 : 낙산비치호텔
- 주 제 : 차세대 정보기술
- 주 최 : 전문대학전산교육연구회
- 문 의 처 : 인하공업전문대학 전자계산과 민태홍 교수
  Tel. 032-870-2323