병렬처리를 위한 고성능 라이브러리의 구현과 성능 평가

(Implementation and Performance Analysis of High Performance Computing Library for Parallel Processing)

김 영 태 † 이 용 권 ††
(Youngtae Kim) (Yong Gwon Lee)

요 약 본 연구에서는 병렬프로그램을 효율적으로 개발할 수 있고 병렬처리 프로그램의 성능을 향상시키는 이식성을 갖는 고성능 병렬 라이브러리인 HPCL(High Performance Computing Library)을 구현하였다. HPCL은 C 언어와 Fortran 언어로 구현되었으며, Fortran 프로그램에서 메시지 전송 인터페이스인 MPI(Message Passing Interface)를 효율적으로 사용할 수 있도록 하였다. 성능 분석은 PC 클러스터와 상업용 슈퍼컴퓨터인 IBM SP4를 이용하여 병렬프로그램의 성능 향상 및 통신 오버헤드 등에 대하여 다양하게 이루어졌다.

키워드 : 메시지전송, 클러스터, 속도개선, 확장성

Abstract We designed a portable parallel library HPCL(High Performance Computing Library) with following objectives: (1) to provide a close relationship between the parallel code and the original sequential code that will help future versions of the sequential code and (2) to enhance performance of the parallel code. The library is an interface written in C and Fortran programming languages between MPI(Message Passing Interface) and parallel programs in Fortran. Performance results were determined on clusters of PC's and IBM SP4.

Key words : Message passing, Cluster, Speedup, Scalability

1. 서 론

오래 전부터 공학 및 과학 분야에서 컴퓨터의 무한한 속도에 관한 기대는 계속되어 왔다. 최근에 와서 컴퓨터 프로세서의 성능은 괄목할 만큼 발전하여 일반 개인용 컴퓨터의 계산 성능이 10여 년 전의 슈퍼컴퓨터를 능가하는 속도를 갖게 되었지만 컴퓨터의 성능이 향상될수록 더 빠른 초고속 컴퓨팅에 대한 기대는 더 높아지고 있다. 여러 개의 프로세서들을 동시에 사용하여 프로그램을 실행함으로써 현격하게 계산시간을 줄일 수 있는 병렬컴퓨팅은 이러한 기대를 해결할 수 있는 가장 이상적인 방식으로 인식되고 있다. 병렬컴퓨터는 각 프로세서와 주기억장치의 연결 방식에 의하여 각 프로세서가 각각 기억장치를 갖는 분산 메모리형 컴퓨터와 모든 프로세서들이 기억장치를 공유하는 공유 메모리형 컴퓨터가 있다. 병렬프로그램 방식으로는 표준방식인 OpenMP와 MPI(Message Passing Interface)를 이용한 메시지 전송 방식이 대표적이며, OpenMP 방식은 공유메모리형에서만 실행이 가능한 반면에 MPI를 이용하여 구현된 병렬프로그램은 분산메모리형의 병렬컴퓨터뿐만 아니라 공유메모리형의 컴퓨터에서도 실행이 될 수 있다[1]. 메시지 전송 방식은 분산메모리형의 컴퓨터에서 가장 많이 사용되는 방식으로서 프로세스가 데이타를 송수신하기 위한 명령을 함수의 형태로 호출하여 사용한다. 메시지의 교환을 위해서는 프로그램 작성 시에 사용자가 데이타의 크기, 형태 및 전송 방향 등을 전부 지정해 주어야 하기 때문에 프로그램의 개발이 어렵다는 단점이 있지만 정확한 메시지 교환에 대한 내용을 기술할 수 있기 때문에 프로그램의 성능이 향상된다는 장점이 있다. 이런 이유로 Pacheco는 [1]에서 메시지 전송 프로그래밍을 "assembly language of parallel computing"이라고 언급했다. 최근에는 초고속 네트워크를 이용한 범용 컴퓨터의 클러스터를 이용한 저가의 분산처리 시스템의

† 정 회 원 : 강릉대학교 컴퓨터공학과
ykim@kangnung.ac.kr
†† 비 회 원 : (주) 라딕스 연구원
ygsoft@knusun.kangnung.ac.kr
논문접수 : 2003년 9월 14일
심사완료 : 2004년 5월 4일

활용이 보편화되었고 이러한 클러스터 컴퓨팅에서는 메시지 전송 방식을 이용한 병렬프로그램만이 가능하기 때문에 메시지 전송 방식을 이용하는 것이 최근의 추세이다.

본 연구에서는 Fortran 프로그램에서 MPI(Message Passing Interface)를 효율적으로 사용할 수 있는 이식성을 가진 고성능 라이브러리를 구현하였다. MPI를 이용한 병렬처리 프로그래밍은 병렬컴퓨터의 종류에 상관없이 효율적으로 실행될 수 있다는 장점이 있지만 MPI의 각 함수가 기본적인 기능만 제공하기 때문에 프로그램의 계산 특성에 맞게 구현하는 것이 어렵다. 특히 대형 프로그램을 병렬프로그램으로 전환하는 병렬화는 프로그램 크기의 방대함으로 인하여 효율적인 병렬프로그램의 구현에 장시간이 소요되었으며, 아울러 디버깅이 극도로 어려웠다. HPCL은 이러한 어려움을 해결하기 위하여 병렬프로그램과 MPI 사이에서의 인터페이스 형태로 Fortran 프로그램에서 메시지 전송 함수를 효율적으로 호출할 수 있도록 개발되었으며, 동적메모리 할당과 같이 Fortran 언어에서 구현하기 어려운 기능을 제공하기 위하여 C 언어로 구현되었다.

HPCL은 크게 두 가지 병렬프로그램을 위한 기본 기능을 제공한다: (1) 인덱스 전환, (2) 통신 인터페이스. 인덱스 전환을 위해서는 HPCL이 병렬 코드의 루프를 분석하여 전역 인덱스와 지역 인덱스의 데이타베이스를 실시간으로 제공하며 통신 인터페이스에서는 계산 도메인의 분산 및 통합을 포함하여 계산 과정 중에 필요한 데이타 송수신에 대한 함수들을 제공한다. HPCL을 이용하는 가장 큰 장점은 병렬프로그램의 구현 시간을 현저하게 줄일 수 있고 아울러 병렬프로그램의 성능을 높일 수 있다는 점이다. 또한 병렬프로그램을 원래 프로그램의 형태와 유사하게 함으로써 병렬프로그램의 지식이 없는 현업의 사용자도 유지 및 보수를 쉽게 할 수 있다.

본 연구에서는 HPCL의 병렬프로그램의 구현과 성능 평가를 위하여 유한차분법(FDM: Finite Difference Method) 수치 모델 QPM(Qualitative Prediction Model)을 병렬화하였다. 대형 수치 모델의 병렬프로그램의 구현은 프로그램의 방대함으로 인하여 대부분 새로이 프로그램을 구현하기보다 기존의 프로그램을 병렬화 하고 있다. 따라서 본 논문에서는 기존의 프로그램의 병렬화 위주로 설명하고자 한다. HPCL을 이용한 병렬 QPM은 PC 클러스터와 상업용 슈퍼컴퓨터인 IBM SP4를 이용하여 프로그램의 정확도와 계산시간 및 통신에 대한 다양한 성능 분석을 하였다.

본 논문은 2장에서는 HPCL의 기능과 특성 및 구현, 3장에서는 병렬 QPM을 이용한 HPCL의 성능 분석, 그리고 4장의 결론으로 구성된다.

2. HPCL(High Performance Computing Library)

HPCL의 기능은 크게 병렬프로그램 초기화 및 계산 도메인 정의, 데이타 입력 및 분산, 데이타 통신 및 병렬 연산 그리고 데이타 통합 및 출력 등으로 구분된다. 이 장에서는 HPCL의 각 기능과 특성에 대하여 설명한다.

2.1 HPCL의 기본 구조

HPCL은 병렬프로그램과 MPI의 중간에 위치하여 병렬프로그램에서 MPI를 직접 호출하는 대신에 수치 모델에 맞도록 파일 입출력, 도메인 분산, 통신 데이타 정의 및 송수신, 그리고 기타 병렬프로그램에서 필요한 MPI의 함수들을 효율적으로 사용할 수 있게 해 주며, 구현된 코드의 재사용을 가능하게 한다. 또한 HPCL은 C 언어와 Fortran 언어로 구현되었기 때문에 동적 메모리 할당 및 파일 정형 입출력 등 두 언어의 특징을 이용할 수 있으며, 여러 함수들의 집합인 라이브러리 형태로써 최적의 병렬프로그램 개발 환경을 제공하여 계산 성능을 향상시킨다.

HPCL의 구조는 그림 1에서와 같이 데이타베이스 HPCL_DB를 가지고 있다. 여기에는 프로세스의 번호, 프로세스의 개수, 프로세스의 위상 정보, 메인/서브 도메인 정보, 통신 패턴 정보를 가지고 있다. 순차 프로그램의 병렬화는 HPCL 함수들의 호출을 통하여 이루어진다.

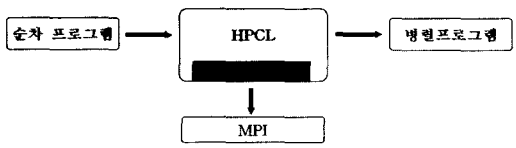

그림 1 HPCL의 기본 구조

일반적으로 수치 모델 프로그램은 그림 2의 왼쪽 부분과 같이 크게 프로그램 초기화, 파일로부터 입력 데이타 읽기, 반복 연산 그리고 파일로의 결과 출력으로 구분된다. 따라서 병렬프로그램도 기본적으로 동일한 구조를 가지며 프로세스들의 병렬 계산을 위하여 그림 2의 오른쪽 부분과 같이 구분한다. 이와 같은 병렬프로그램의 구조에 따라서 HPCL의 각 기능도 다음의 4 부분으로 나누어진다.

(i) 초기화 및 도메인 정의: 병렬프로그램에 필요한 변수들 초기화와 분산될 도메인 정의

(ii) 데이타 입력 및 분산: 제어 프로세스에서 파일로부터 데이타를 읽거나 임의로 생성하여 각 프로세스로 서브 도메인 분산

(iii) 데이타 통신 및 병렬 연산: 데이타의 일관성을 유지하기 위한 프로세스간의 데이타 통신 및 서브 도메인의 병렬 연산

(iv) 데이타 통합 및 출력: 서브 도메인을 제어 프로세스에서 통합하고 결과를 파일에 쓰거나 화면 출력 처리

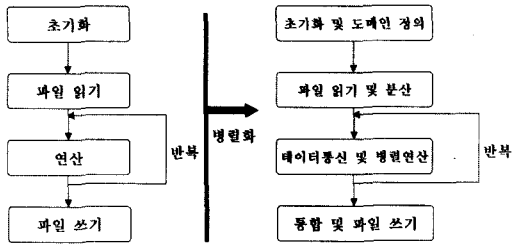

그림 2 수치모델 프로그램의 일반적인 형태

2.2 초기화 및 도메인 정의

HPCL에서는 초기의 도메인 정보 및 도메인 분산에 대한 위상 정의, 서브 도메인의 형태에 대한 정보 입력을 함수를 통하여 이를 데이타베이스에 저장한다. 먼저 HPCL은 도메인의 크기를 이용하여 분산될 서브 도메인의 크기를 자동으로 생성해 준다. 아래 예를 보면 연산시 필요한 경계영역의 통신을 위한 버퍼(나중에 설명)의 크기가 2이고 프로세스의 위상이 4×3이라고 할 때 도메인의 크기가 nx×ny×nz인 경우에 서브 도메인의 크기는 Lnx×Lny×nz가 된다. 이는 그림 3과 같이 각 프로세스에 할당될 도메인의 크기를 의미한다. 이 예에서는 nz는 고려하지 않고 2차원 방향으로의 분산만 하는 형태이다.

parameter (nx=231,ny=231,nz=20)
parameter (Lnx=**HPCL_PARALLEL(nx,4,2)**)
parameter (Lny=**HPCL_PARALLEL(ny,3,2)**)

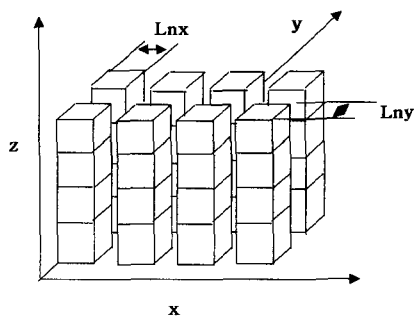

그림 3 2차원 도메인 분할

변수의 선언을 한 후에는 다음과 같이 HPCL의 데이타베이스 구축에 필요한 정보를 입력한다.

HPCL_INIT()
HPCL_DEFINE_COMM(comm_id,nxproc,nyproc)
HPCL_DEFINE_DOMAIN(form_id,comm_id,datatype,nx,ny,nz, ghost_area)

HPCL_INIT() 함수는 MPI 환경을 초기화하고 HPCL 환경에 필요한 데이타베이스 HPCL_DB의 정보를 초기화한다. 함수 HPCL_DEFINE_COMM()는 프로세스의 위상 정보를 HPCL_DB에 추가하며, HPCL_DEFINE_DOMAIN() 함수에서는 HPCL_DB에 저장이 되어 있는 프로세스의 위상 정보(comm_id)를 이용하여 주 도메인 크기인 nx, ny, nz와 통신 버퍼인 ghost_area를 이용하여 위상에 따라 각 프로세스에서의 서브 도메인의 형태가 정의되어 HPCL_DB에 저장된다. 이 때 정의된 서브 도메인에 관한 정보 form_id는 파일 읽기, 파일 쓰기, 분산 및 통신 시에 사용된다. HPCL을 이용한 병렬프로그램에서는 다양한 형태를 가진 프로세스의 위상과 각 위상에 맞게 정의되어진 데이타의 형태가 함께 존재할 수 있다.

2.3 데이타 입력과 분산

HPCL에서의 입출력 함수는 병렬 Fortran 프로그램에서의 입출력 함수와 호환을 갖게 하기 위하여 Fortran 언어의 함수를 이용한다. HPCL의 입력 함수를 다음과 같이 호출한다.

HPCL_READ(unit,format,variable,form_id)

HPCL_READ()를 호출하면 제어 프로세스는 form_id를 이용하여 HPCL_DB에 저장된 도메인 정보를 읽는다. 이 정보를 통하여 (i) 필요한 메모리를 동적으로 할당하고, (ii) 파일로부터 초기데이타를 읽어들이며, (iii) 이를 서브 도메인으로 분산하고, (iv) 할당된 메모리를 반환한다(그림 3 참조). 따라서 병렬프로그램에서는 HPCL의 입력 함수만 호출하면 입력 후 자동으로 도메인 분산을 실행한다.

이러한 방식은 데이타의 메모리 사용이 큰 순차 프로그램에서 효율적으로 메모리를 사용할 수 있으며 프로그램의 컴파일 시에 makefile에서 -DNYPROC=#nyproc -DNXPROC=#nxproc의 형태로 프로세스의 개수가 프로그램에 전달되어 프로그램이 자동으로 각 프로세스의 분산된 도메인을 계산하게 된다. 한편 이 방식은 프로세스의 개수가 달라질 때 병렬프로그램을 다시 컴파일을 해야 하는 단점이 있다.

2.4 데이타 통신 및 병렬 연산

2.4.1 인덱스 전환

병렬프로그램에서는 분산된 서브 도메인을 처리하기 위해서 각 프로세스가 지역 인덱스를 이용하지만 메인 도메인에서의 정보를 이용하기 위하여 전역 인덱스가 필요한 경우가 있다. 따라서 일반적으로 테이블을 이용하여 2개의 인덱스를 동시에 사용한다. HPCL에서는 연산에 필요한 인덱스를 얻어오기 위한 함수 및 서브 도

메인을 처리하기 위한 함수들을 지원한다.

HPCL_GET_INDICES() 함수는 HPCL_DB에 저장되어 있는 메인/서브 도메인 정보를 이용하여 전역 인덱스를 얻어오기 위한 함수로서, 1차원 및 2차원에 해당되는 지역 인덱스와 전역 인덱스를 계산한다. 그림 4는 메인 도메인이 1차원으로 3개의 프로세스로 분산되었을 때 각 서브 도메인이 정해지는 결과를 보여준다. HPCL_X는 x축 방향으로의 1차원 분산을 나타내는 입력값이며, begin, end는 지역 인덱스의 시작과 끝을 가리킨다.

call HPCL_GET_INDICES(form_id,HPCL_X,indices, begin,end)

전역 인덱스에서 일정 범위 내의 서브 도메인을 처리하기 위해 연산 반복문 DO문 대신 사용하기 위한 HPCL_DO을 제공한다. HPCL_ENDDO는 병렬화된 HPCL_DO를 종료한다. 그림 5는 HPCL_DO와 HPCL_ENDDDO문이 DO문에 적용된 사례이다. 병렬프로그램에서 HPCL_ENDDO를 호출하면 이 함수는 인덱스에 관한 정보를 이용하여 서브도메인에 대한 반복문을 자동으로 처리한다. 한편 중간 부분의 if문들에서의 값들은 전역 인덱스 값에 대한 처리이기 때문에 지역 인덱스 값을 전역 인덱스 값으로 대치하여 처리하기 위한 HPCL_GLOBAL()을 사용하였다. HPCL_GLOBAL()을 사용하지 않을 경우는 프로세스들이 각 서브도메인을 모두 계산하게 되므로 잘못된 계산 결과가 나오게 된다.

2.4.2 경계 영역의 데이타 통신 및 전역 통신

병렬프로그램에서 가장 많이 필요한 통신으로서 경계 영역의 데이타 통신이 있다. 이는 연산 과정에서 서브 도메인의 값이 수정되었을 때 인접한 프로세스의 값의 일관성을 유지하기 위한 것으로서 특히 유한차분법을 기본으로 하는 계산에서는 많이 발생하게 된다[2,3]. 예를 들어, 모든 프로세스가 다음과 같은 연산을 수행하였을 경우에 a를 계산하기 위해서 도메인의 경계부분에 있는 경우에는 그림 6과 같이 인접한 다른 프로세스로부터 데이타 b를 수신 받아야 일관성을 유지하게 된다. 이 때 수신 받는 버퍼가 ghost_area로 정의되며 이 버퍼는 병렬프로그램에서는 초기화 시에 크기만 지정하고 HPCL이 자동으로 이용하여 병렬프로그램에서는 보이지 않게 된다.

$$a = f(b(i-1,j-1), b(i-1,j), b(i,j-1))$$

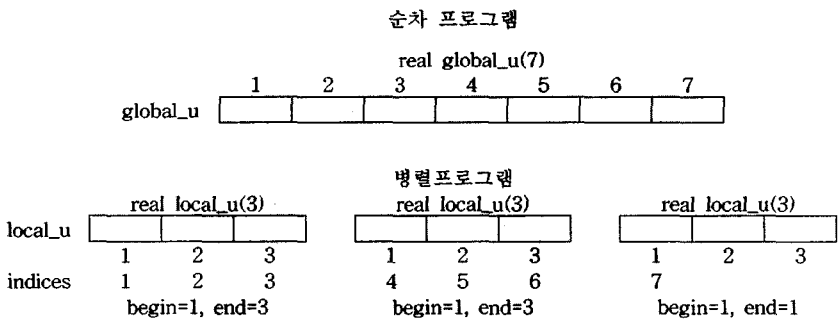

그림 4 3개의 프로세스로의 도메인 분할의 예

```
[redacted]
  g12(i,j)=(rhtop-zs(i,j))/rhtop
  if([redacted]) then
    dxval=(zs(i+1,j)-zs(i,j))/dx
  else
    dxval=(zs(i,j)-zs(i-1,j))/dx
  endif
  if([redacted]) then
    dyval=(zs(i,j+1)-zs(i,j))/dy
  else
    dyval=(zs(i,j)-zs(i,j-1))/dy
  endif
  g13(i,j)=(rhtop-zs(i,j))*dxval
  g14(i,j)=(rhtop-zs(i,j))*dyval
  [redacted]
[redacted]
```

```
[redacted]
  g12(i,j)=(rhtop-zs(i,j))/rhtop
  if([redacted]) then
    dxval=(zs(i+1,j)-zs(i,j))/dx
  else
    dxval=(zs(i,j)-zs(i-1,j))/dx
  endif
  if([redacted]) then
    dyval=(zs(i,j+1)-zs(i,j))/dy
  else
    dyval=(zs(i,j)-zs(i,j-1))/dy
  endif
  g13(i,j)=(rhtop-zs(i,j))*dxval
  g14(i,j)=(rhtop-zs(i,j))*dyval
  [redacted]
[redacted]
```

그림 5 HPCL_DO과 HPCL_ENDDO의 사용 예

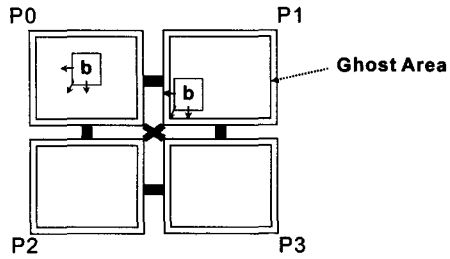

그림 6 인접 프로세서로의 통신을 위한 버퍼(ghost area) 정의

HPCL_ADD_PATTERN() 함수는 경계 영역의 데이타 통신을 위한 정보를 만드는 함수로 form_id로부터 얻은 메인/서브 도메인 정보를 통해 변수에 대해 어느 방향으로 얼마만큼의 경계 영역을 통신하는지를 정의하는 함수이다. HPCL에서 X(i,j)를 기준으로 방향은 표 1과 같다.

표 1 HPCL의 통신 방향표

HPCL_NW X(i-1,j-1)	**HPCL_N** X(i,j-1)	**HPCL_NE** X(i+1,j-1)
HPCL_W X(i-1,j)	X(i,j)	**HPCL_E** X(i+1,j)
HPCL_SW X(i-1,j+1)	**HPCL_S** X(i,j+1)	**HPCL_SE** X(i+1,j+1)

예를 들어, 연산 내에 다음과 같은 수식이 있다고 가정하자.

X(i,j) = X(i-1,j) + X(i,j) + X(i+1,j) + X(i,j+2)

Y(i,j) = Y(i,j-1) + Y(i,j) + Y(i,j+1)

X값은 X의 HPCL_W, HPCL_E 방향으로 1만큼, HPCL_S 방향으로 2만큼의 경계 영역 값을 필요로 하고 Y값은 Y의 HPCL_N, HPCL_S 방향으로 1만큼의 경계 영역 값을 필요로 한다. 정의 후 실제 통신을 하기 위해서는 HPCL_SYNC() 함수를 호출하면 된다. HPCL_SYNC()은 패턴에 정의된 데이타들을 모아 한번에 송수신하기 때문에 성능 향상을 기대할 수 있다. 위의 수식에 대한 통신을 정의하기 위한 함수 호출은 다음과 같다.

```
call HPCL_ADD_PATTERN(pattern_id, X, form_id, HPCL_E, 1)
call HPCL_ADD_PATTERN(pattern_id, X, form_id, HPCL_W, 1)
call HPCL_ADD_PATTERN(pattern_id, X, form_id, HPCL_S, 2)
call HPCL_ADD_PATTERN(pattern_id, Y, form_id, HPCL_S, 1)
call HPCL_ADD_PATTERN(pattern_id, Y, form_id, HPCL_N, 1)
...
call HPCL_SYNC(pattern_id)
```

HPCL_REDUCE() 함수는 각 프로세스의 지역 변수에 대한 전체 합(HPCL_SUM), 곱(HPCL_PROD), 최대값(HPCL_MAX), 최소값(HPCL_MIN) 및 복사(HPCL_DUP) 연산을 지원한다.

2.5 데이타 통합 및 출력

HPCL에서의 출력 함수는 입력 함수와 유사하며, 다음과 같이 HPCL의 출력 함수를 호출한다.

HPCL_WRITE(unit,format,variable,form_id)

HPCL_WRITE()를 호출하면 form_id를 이용하여 HPCL_DB에서 메인/서브 도메인 정보를 얻어 제어 프로세스는 (i) 필요한 만큼의 메모리를 할당하여 (ii) 각 프로세스의 서브 도메인을 통합하고, (iii) 파일 출력을 수행 후, (iv) 메모리를 반환한다. HPCL의 입력 함수와 마찬가지로 병렬프로그램에서는 출력 함수만 호출하면 여러 과정들이 자동으로 수행된다.

3. 성능 분석

본 연구에서는 유한차분법을 기본 계산 형태로 하는 기상모델인 QPM을 HPCL을 이용하여 병렬화하였다. 이 장에서는 병렬 QPM의 성능 분석을 통하여 HPCL의 성능을 알아본다. 성능을 분석하기 위해서 병렬프로그램에서 발생하는 통신 오버헤드를 먼저 분석하고 프로세서의 증가에 따른 속도개선 및 확장성을 분석하였다. HPCL의 정확도는 병렬프로그램의 결과와 병렬화 이전의 프로그램의 결과와 비교하여 정확하게 일치하였으므로 여기서는 생략하기로 한다. 성능 분석에는 병렬컴퓨터인 PC 클러스터와 상업용 병렬컴퓨터인 IBM SP4를 사용하였다.

3.1 QPM(Qualitative Prediction Model)

QPM은 주어진 시간 내에 공간 강수량 성분의 변화를 모의하는 전형적인 형태의 기상 모의 프로그램으로서 도메인의 기본적인 변수의 변화값은 유한차분법을 이용하여 계산한다[4]. 유한차분법의 메시지 전송을 이용한 병렬화는 주로 인접한 프로세스간의 데이타 송수신을 필요로 하며, 일반적으로 병렬프로그램이 뛰어난 성능을 보여준다[5,6]. QPM의 프로그램의 구조는 그림 2와 같이 프로그램 초기화, 파일로부터의 자료 입력, 반복 계산 그리고 계산 결과의 파일 출력으로 구성되어 있으며, 병렬 QPM은 메인/서브 도메인 정의, 파일로부터 입력하여 분산, 데이타 통신 및 연산, 연산 결과를 통합하여 파일 출력하는 부분으로 구성된다.

3.2 실행 환경

병렬 QPM의 성능 분석을 위해서 실행한 각 병렬컴퓨터의 사양은 표 2와 같다.

2대의 PC 클러스터 컴퓨터는 AMD Athlon CPU를 사용하였으며 각 노드는 2개의 CPU로 구성이 되어 있다. 통신 장비가 병렬프로그램에 미치는 영향을 알아보기 위하여 100Mbps의 통신속도를 가진 범용 스위치 허브 장비를 사용하는 컴퓨터와 클러스터 전용 통신 장비인 Myrinet을 사용하는 컴퓨터에서 각각 실행하였다. Myrinet는 속도가 빠르고 확장성이 우수하여 클러스터 컴퓨터에서 우수한 성능을 보이는 통신 장비이다[7,8]. 상업용 병렬컴퓨터인 IBM SP4는 8개의 CPU로 구성된 Multi-Chip Module(MCM)로 구성되어 있으며 공유메모리와 분산메모리의 혼합형 병렬컴퓨터이다[9].

3.3 통신 오버헤드

통신 오버헤드는 단일 프로세서를 이용하면 발생하지 않고 분산 메모리형 병렬컴퓨터에서 메시지 전송을 이용하는 경우에만 발생한다. 따라서 병렬프로그램에서 통신 오버헤드를 줄이는 것이 성능 향상에 가장 중요한 요소이다. 그림 7, 8, 9는 각 병렬컴퓨터에서의 프로세서 간의 데이타 통신에 소요되는 오버헤드를 전체 실행시간의 퍼센트로 나타내었다. 통신 시간에서 입출력 시에 발생하는 도메인 분산 및 통합에 걸리는 시간은 각 변수마다 한번씩 발생하여 성능에 거의 영향을 미치지 않으므로 제외하였다. 그림은 HPCL에서 제공하는 통신 함수가 통신 오버헤드가 낮아서 병렬프로그램의 성능이 우수하다는 것을 알 수 있다. 특히 16개나 32개의 프로세서를 사용했을 경우에 프로세서의 속도에 비하여 통

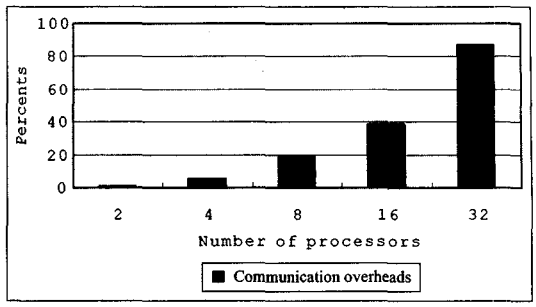

그림 7 Ethernet을 이용한 클러스터에서의 병렬 QPM의 통신부하

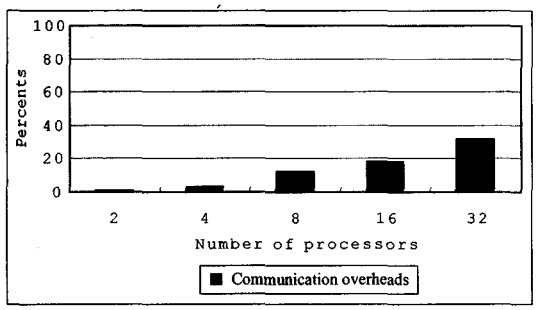

그림 8 Myrinet을 이용한 클러스터에서의 병렬 QPM의 통신부하

신 속도가 현저히 느린데도 불구하고 20~30%의 통신 오버헤드를 보이는 것은 병렬프로그램이 통신에 사용하는 시간이 매우 낮다는 것을 보여 준다([5] 참조). 한편, 그림 8과 9의 경우에는 통신 오버헤드가 비교적 낮게

표 2 병렬컴퓨터의 사양

AMD Athlon with 100Mbps switch	CPU	AMD Athlon 1660MHz (Cache 256KB) * 2 - L1 Cache 128KB (64D/64I) - L2 Cache 256KB
	Memory / Swap	2GB / 2GB
	Network	Ethernet (100Mbps)
	HDD	65GB(SCSI)
AMD Athlon with Myrinet	CPU	AMD Athlon 1660MHz (Cache 256KB) * 2 - L1 Cache 128KB (64D/64I) - L2 Cache 256KB
	Memory / Swap	2GB / 2GB
	Network	Myrinet (1000Mbps)
	HDD	65GB(SCSI)
IBM SP4	CPU	IBM POWER4 1.3GHz * 32 - L1 Cache 96KB (32D/64I) - L2 Cache 1.44MB over 100GB/s - L3 Cache 32MB over 10GB/s
	Memory / Swap	160GB / 800GB
	Network	Gigibit Ethernet (1000Mbps)
	HDD	576GB

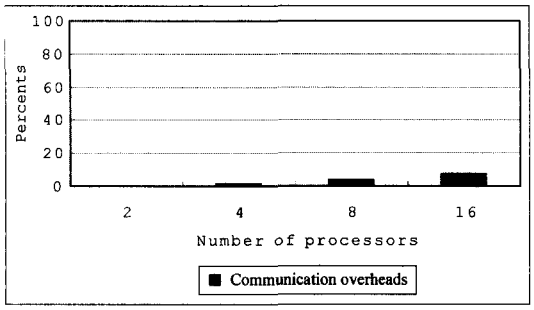

그림 9 IBM SP4에서의 병렬 QPM의 통신부하

나오는데 반하여 그림 7은 많은 부분의 통신 오버헤드가 발생한다. 특히 32개의 프로세서를 사용하는 경우에는 통신에 따른 부하가 많은 부분을 차지하고 있는 것을 볼 수 있다. 이는 범용 통신 장비는 많은 수의 프로세서를 사용할 경우에 Myrinet에 비교하여 성능 개선과 확장성에 크게 영향을 미치는 것을 보여 준다. 이와 같이 최근의 병렬컴퓨터에서는 통신 속도에 비하여 프로세서의 성능이 뛰어나므로 통신 장비의 선택이 병렬프로그램의 실행 성능에 절대적인 영향을 미치게 된다.

3.4 속도개선(Speedup)과 확장성(Scalability)

HPCL의 효율적인 성능을 알 수 있는 가장 큰 요소는 프로세서의 개수의 증가에 따른 속도개선과 확장성이다. 다시 말하면, 병렬프로그램의 구현에 있어서는 특정 병렬컴퓨터의 속도와 상관이 없이 병렬화를 통한 성능의 향상이 가장 중요한 요소이다. 속도개선을 알아보기 위하여 각 병렬컴퓨터에서 단일 프로세서를 사용한 병렬화 이전 프로그램의 성능과 다른 개수의 프로세서에서의 병렬프로그램의 성능과 비교하였다. 본 연구에서는 성능을 비교하기 위하여 실행시간을 이용하였다. 그림 10, 11, 12는 2대의 클러스터 컴퓨터와 IBM SP4에서의 성능 비교를 나타낸다. 그림에서 'Ideal execution time'는 단일컴퓨터의 병렬화 이전의 프로그램의 실행시간을 프로세서의 개수로 나눈 시간으로서 통신 오버헤드와 부하 불균형을 고려하지 않은 병렬프로그램의 가장 이상적인 실행시간이다.

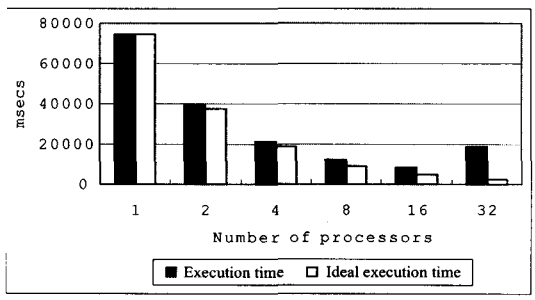

그림 10 Ethernet을 이용한 클러스터에서의 병렬 QPM의 성능개선과 확장성

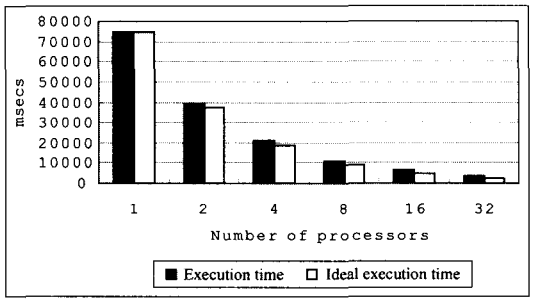

그림 11 Myrinet을 이용한 클러스터에서의 병렬 QPM의 성능개선과 확장성

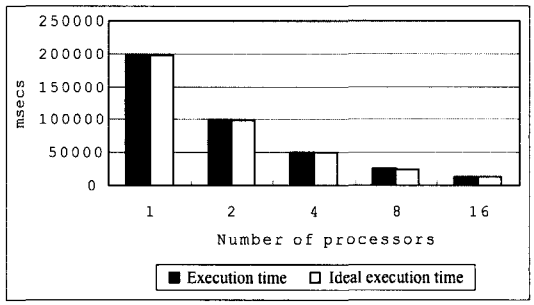

그림 12 IBM SP4를 이용한 클러스터에서의 병렬 QPM의 성능개선과 확장성

병렬 QPM의 프로세서의 개수의 증가에 따른 속도 확장성은 그림에서 알 수 있듯이 범용 통신 장비와 많은 수의 프로세서(32개)를 사용하는 경우를 제외하고는 이상적인 실행시간에 비교해서 매우 우수한 성능을 나타내고 있다. 이러한 성능은 앞에서 설명한 바와 같이 적은 통신 오버헤드에 따른 것이다. 따라서 HPCL을 이용하여 병렬프로그램을 구현하였을 경우에 프로그램의 속도개선과 확장성이 뛰어나다는 것을 알 수 있다.

5. 결 론

본 연구에서는 병렬프로그램에서 MPI 함수를 효율적으로 호출함으로써 병렬프로그램을 쉽게 구현할 수 있도록 HPCL을 개발하였다. HPCL은 유사한 계산을 기본으로 하는 수치 모델들에서 공통적으로 사용하는 MPI의 함수들을 HPCL의 함수를 통하여 호출하고 또한 이를 재사용할 수 있도록 하였다. HPCL은 병렬프로그램을 효율적으로 개발할 수 있을 뿐 아니라 프로그램의 성능도 뛰어난 것을 보여주었다. HPCL을 이용하여 병렬화된 병렬프로그램은 병렬화 이전의 프로그램과 구조가 거의 유사하기 때문에 현업에서도 프로그램의 유

지 및 보수가 용이하다. 따라서 HPCL은 병렬프로그램의 성능 향상과 아울러 이식성을 가진 병렬화를 효율적으로 할 수 있도록 하는데 있어서 매우 중요하다. 또한 HPCL을 이용한 병렬프로그램은 어떤 형태의 병렬컴퓨터에서도 실행이 가능하기 때문에 다양한 형태의 슈퍼컴퓨터 계산환경 및 슈퍼컴퓨터의 새로운 대체형으로 인식되고 있는 클러스터 컴퓨터의 계산환경에서도 운영될 수 있도록 함으로써 향후의 새로운 계산환경에도 쉽게 적용할 수 있게 된다.

현재 HPCL의 이식성 확장을 위하여 이를 이용한 다른 유한차분법 계산 방식의 병렬프로그램들을 구현하고 있으며, 필요에 따라 HPCL을 개선하고 있다. 향후에는 유한원소법(Finite Element Method) 및 스펙트랄 방식 등의 다른 계산 방식을 이용하는 수치모델에서도 구현할 수 있도록 HPCL을 확장하고자 한다.

참 고 문 헌

[1] Pacheco P., *Parallel Programming with MPI*, San Francisco, CA: Morgan Kaufmann, 1997.

[2] Kim, Y., Kothari, S., Takle, E. and Pan, Z., "A run-time library and load balance analysis for parallel atmospheric models," *Symposium on Regional Weather Prediction on Parallel Computer Environments*, Athens, Greece, 1997.

[3] Michalakes, J., *RSL: A parallel runtime system library for regular grid finite difference models using multiple nests*, Tech. Rep. ANL/MCS-TM-197, MCS Division, Argonne National Laboratory, Argonne, Illinois, 1994.

[4] Misumi, R., Bell, V. A and Moore, R. J., "River flow forecasting using a rainfall disaggregation model incorporating small-scale topographic effects," *Meteorol. Appl.* 8, 297-305, 2001.

[5] Kim, Y., Pan, Z., Takle, E. and Kothari, S., "Parallel Implementation of Hydrostatic MM5 (Mesoscale Model)," *The 8th SIAM Conference on Parallel Processing for Scientific Computing*, Minneapolis USA, Mar. 14-17, 1997.

[6] Michalakes, J., Canfielf, T., Nanjundiah, R. and Hammond, S., "Parallel implementation, validation, and performance of MM5," *Proc. 6th Workshop on the use of Parallel Processors in Meteorology*, Reading, U. K., European Center for Medium Range Weather Forecasting, 1994.

[7] Hsieh, J., Leng T., Mashayekhi V. and Roohollamini R., "Architectural and Performance Evaluation of GigaNet and Myrinet Interconnects on Clusters of Small-Scale SMP Servers", Proc. *Supercomputing Conference*, Dallas, USA, 2000.

[8] 김영태, 이용희, 최준태, 오재호. 초고속 네트워크를 이용한 PC 클러스터의 구현과 성능 평가, 정보과학회논문지:시스템 및 이론, 제29권 2호, pp. 57-64, Feb. 2002.

[9] 한국과학기술정보원(KISTI) 슈퍼컴퓨팅센터, IBM 시스템 사용자 지침서 v.1.6, 2003. 3.

김 영 태

1986년 연세대 수학 학사. 1996년 미국 Iowa State Univ. 컴퓨터과학 Ph.D., 1998년~현재 강릉대학교 교수

이 용 권

2001년 강릉대학교 컴퓨터공학 학사. 2004년 강릉대학교 컴퓨터공학 석사. 2004년~현재 (주) 라딕스 연구원