2012 슈퍼컴퓨팅 경진대회 문제
- 학부팀-

0. 공통 사항

- 평가

○ 주어진 문제를 해결하여 정량적 평가 (80%)
- 각 제출한 결과에 대한 상대평가를 수행하여 진행함
- 문제별 점수 배점 표기하였음

○ 발표평가를 통한 정성적 평가 (20%)

- 제한 사항

○ 프로그래밍 언어: C, Fortran 모두 가능

○ 참가자들은 컴파일러(intel, gcc) 가능

○ 시간제한
- 10일 13:00 ~ 11일 13:00
- 제출시간 - 1, 2번 문제 : 10/11 00:00
  - 3번 문제 : 10/11 13:00
- 제출방법 : e-mail 송부 (edu@ksc.re.kr)
- 제한 시간 내 결과 제출

○ 특정 라이브러리 및 컴파일러 최적화 옵션 사용 가능

1. 시스템 성능 측정 문제 (점수 : 24점) _직접 작성

(1) 각 팀에 할당된 하나의 노드에 대한 이론 성능을 FLOPS으로 제출하시오

(2) 64bits(8 bytes) 실수 벡터 $\vec{a}$, $\vec{b}$에 대한 $s = \vec{a} \bullet \vec{b}$(dot product)를 계산하고, 실측성능을 FLOPS으로 제출하시오

※ 벡터의 크기 N=12*1024*1024*1024, $a_i = i,\ b_i = \frac{1}{i}, i = 1,...,N$

(3) 병렬 코드, 컴파일 옵션, 실행 파일, 결과 출력 후 제출할 것

※ 하나 또는 두 노드를 모두 활용할 수 있으며, 최대 실측 성능을 제출할 것

※ FLOPS : FLOating Point operations Per Second

※ 한 코어의 클럭 당 Flops = 4 Flops/clock

2. Matrix-Matrix Multiplication 문제 (점수 : 16점) _ source code 제공

(1) 주어진 NxN 행렬에 대하여 A·B를 구하고 모든 원소의 합을 구하는 순차 코드를 병렬 처리하여 최대 성능을 FLOPS으로 계산하여 제출 하시오

(2) 병렬 코드, 컴파일 옵션, 실행 파일 및 결과 출력 후 제출할 것

3. **HPL 최대 성능 (점수 : 40점)**

(1) 주어진 HPL 소스 코드에 대하여 최대 성능을 제출하시오

(2) 컴파일 옵션, 실행 파일 및 결과 파일 제출할 것

학부 문제 2 순차코드

matrix_mul.F90

```
program matrix
implicit none

integer :: i,j,k,n
real*8 sum, t1, t2
real*8,dimension(:,:),allocatable :: a,b,c

n = 1000
sum=0.0
c=0.0

allocate (a(n,n))
allocate (b(n,n))
allocate (c(n,n))

call cpu_time(t1)

do j=1,n
 do i=1,n
  a(i,j)=real(i)/real(j)
  b(i,j)=real(j)/real(i)
 end do
end do

do k=1,n
 do j=1,n
  do i=1,n
   c(i,j)=c(i,j)+a(i,k)*b(k,j)
  end do
 end do
end do

do j=1,n
 do i=1,n
  sum=sum+c(i,j)
 enddo
enddo

call cpu_time(t2)

print *, 'SUM = ',sum
print *, 'Elapse Time =',t2-t1
print FLOPS between time check routine

deallocate(a,b,c)

end program matrix
```

matrix_mul.c

```
#define N 512

int main()
{
        int i, j, k;
        double a[N][N], b[N][N], c[N][N] ;
        double sum=0.0, flops=0.0;
        struct timeval tv_start, tv_end, tv_diff;

        for (i=0; i<N; i++)
        for(j=0; j<N; j++)
        c[i][j]=0.0;

        printf("O.K");

        gettimeofday(&tv_start, NULL);  //    timecheck();

        for (i=0; i<N; i++){
        for (j=0; j<N; j++){
        a[i][j] = (double)(i+1)/(double)(j+1);
        b[i][j] = (double)(j+1)/(double)(i+1);
        }
        }

        for (i=0; i<N; i++)
        for (j=0; j<N; j++)
        for (k=0; k<N; k++)
        c[i][j] += a[i][k]*b[k][j];

        for (i=0; i<N; i++)
        for (j=0; j<N; j++)
        sum += sqrt(c[i][j]*c[i][j]);

        gettimeofday(&tv_end, NULL);  //    timecheck();

        timeval_subtract(&tv_diff, &tv_end, &tv_start);
        printf("Elapsed time : %ld.%06ld ms.₩n", tv_diff.tv_sec, tv_diff.tv_usec);

        print FLOPS between time check routine

        return 0;
}

/* Time function. */
int timeval_subtract(struct timeval *result, struct timeval *t2, struct timeval *t1)
{
    long int diff = (t2->tv_usec + 1000000 * t2->tv_sec) - (t1->tv_usec + 1000000 * t1->tv_sec);
    result->tv_sec = diff / 1000000;
    result->tv_usec = diff % 1000000;
    return (diff<0);
}
```

2012 슈퍼컴퓨팅 경진대회 문제
- 대학원팀-

0. 공통 사항

- **평가**
  - ○ 주어진 문제를 해결하여 정량적 평가 (80%)
    - 각 제출한 결과에 대한 상대평가를 수행하여 진행함
    - HPL 성능 측정 (40%), 문제 1 (16%), 문제 2(24%) 점수 배점
  - ○ 발표평가를 통한 정성적 평가 (20%)
- **제한 사항**
  - ○ 프로그래밍 언어: C, Fortran 모두 가능
  - ○ 참가자들은 컴파일러(intel, gcc) 가능
  - ○ 시간제한
    - 10일 13:00 ~ 11일 13:00
    - 제출시간 - 1, 2번 문제 : 10/11 00:00
      - 3번 문제 : 10/11 13:00
    - 제출방법 : e-mail 송부 (edu@ksc.re.kr)
    - 제한 시간 내 결과 제출
  - ○ 특정 라이브러리 및 컴파일러 최적화 옵션 사용 가능

1. Matrix-Matrix Multiplication 문제 (점수 : 16점)

(1) 각 팀에 할당된 하나의 노드에 대한 이론 성능을 FLOPS으로 제출하시오

(1) 주어진 NxN 행렬에 대하여 A·B를 구하고 모든 원소의 합을 구하는 순차 코드를 병렬 처리하여 최대 성능을 FLOPS으로 계산하여 제출하시오

※ 벡터의 크기 N=12*1024*1024*1024

$$a_{ij} = \frac{i}{j},\ b_{ij} = \frac{j}{i} \qquad i = 1,...,N \qquad j = 1,...,N$$

(3) 병렬 코드, 컴파일 옵션, 실행 파일, 결과 출력 후 제출할 것

※ FLOPS : FLOating Point operations Per Second

2. 2D FDM 문제 (점수 : 24점)

(1) 주어진 2D 경계 조건에서 FDM을 이용한 최대 성능을 FLOPS으로 계산하여 제출하시오

(2) 병렬 코드, 컴파일 옵션, 실행 파일 및 결과 출력 후 제출할 것

3. HPL 최대 성능 (점수 : 40점)

(1) 주어진 HPL 소스 코드에 대하여 최대 성능을 제출하시오
(2) 컴파일 옵션, 실행 파일 및 결과 파일 제출할 것

05. 대학원팀 유형 3 코드

FDM2D.F90

```
program FDM2D
      integer im,jm,im1,jm1
      parameter(im=300,jm=300)
      integer is,ie,js,je
      integer iter,itermax,nprt
      real*8    tolerance , time_start, time_end, elapsed_time
      real*8    bc(4)                    !left,right,bottom,top
      real*8    u(0:im+1,0:jm+1)
      real*8    error

! read input data
      itermax=100000
      nprt=500
      tolerance = 1.0d-7
      bc(1) = 10.0
      bc(2) = 10.0
      bc(3) = 10.0
      bc(4) = 20.0

! initialize
      im1=im+1
      jm1=jm+1
      do j=0,jm1
      do i=0,im1
        u(i,j) = 0.0
      end do
      end do

! boundary conditions
      do j=0,jm1
        u(0   ,j)   = bc(1) !left
        u(im1,j)   = bc(2) !right
      end do

      do i=0,im1
        u(i,0   )   = bc(3) !bottom
        u(i,jm1)   = bc(4) !top
      end do

! set computation range
      is = 1
      ie = im
      js = 1
      je = jm

! main routine
      iter = 0
      error = 1000.

! ********************************************************************
call cpu_time(time_start)
      do while(iter.le.itermax.and.error.gt.tolerance)
        call jacobi(u,im,jm,is,ie,js,je,error)
        iter = iter + 1
      end do
call cpu_time(time_end)
! ********************************************************************

      print*,'Error=',error
      print*,'Converged after ',iter,'iteration'
100    format('Iteration=',i6,'  Error=',e9.4)
```

```
      elapsed_time = time_end - time_start
      print*,'elapsed time=',elapsed_time

      print FLOPS between time check routine

      stop
      end

      subroutine jacobi(u,im,jm,is,ie,js,je,error)
      integer im,jm,is,ie,js,je
      integer i,j
      real error
      real u(0:im+1,0:jm+1), uo(0:im+1,0:jm+1)

! store old data
      do j=0,jm+1
      do i=0,im+1
        uo(i,j) = u(i,j)
      end do
      end do

! jacobi
      do j=js,je
      do i=is,ie
        u(i,j) = (u(i-1,j)+uo(i+1,j)+u(i,j-1)+uo(i,j+1))/4.0
      end do
      end do

! error
      error = 0.0
      do j=js,je
      do i=is,ie
        error = error + (u(i,j) - uo(i,j))**2
      end do
      end do
      return
      end
```

FDM2D.c

```
#include <stdio.h>
#include <math.h>

#define im 100
#define jm 100
#define im1 101
#define jm1 101
#define ITER_MAX 100000

int main()
{
        int i,j,iter, nprt;

        double tolerance, error;
        double u[im1+1][jm1+1], uo[im1+1][jm1+1];
        double elapsed_time;
        struct timeval tv_start, tv_end, tv_diff;

        // Read Input
        iter=0;
        nprt = 500;
        tolerance = 1.e-7 ;
        elapsed_time=0.0;

        // initialize
        for (i=0; i<=im1; i++) {
        for (j=0; j<=jm1; j++) {
        u[i][j] = 0.0;
        uo[i][j] = 0.0;
        }
        }

        // BC
        for(j=0; j<=jm1; j++) {
        u[0][j] = 1.0;
        u[im1][j] = 1.0;
        }
        for(i=0; i<=im1; i++) {
        u[i][0] = 1.0;
        u[i][jm1] = 2.0;
        }

        do {
        gettimeofday(&tv_start, NULL);  //    timecheck();

        // store old data
        //memcpy(uo, u, sizeof(double) * im1 * jm1);
        for (i=0; i<=im1; i++)
        for (j=0; j<=jm1; j++)
        uo[i][j] = u[i][j] ;

        // jacobi
        for(i=1; i<=im; i++)
        for(j=1; j<=jm; j++)
        u[i][j] = (u[i-1][j] + uo[i+1][j] + u[i][j-1] + uo[i][j+1]) / 4.;

        // error
        error = 0.0;
        for(i=1; i<=im; i++)
        for(j=1; j<=jm; j++)
        error += ( (u[i][j] - uo[i][j]) * (u[i][j] - uo[i][j]) );

        gettimeofday(&tv_end, NULL);            // timecheck();

        printf("%d %.16e₩n", iter, error);
        // elapsed_time = elapsed_time + time_check(1)-time_check(2);
        } while( (iter++ < ITER_MAX) && (error > tolerance) );

        timeval_subtract(&tv_diff, &tv_end, &tv_start);
        printf("Elapsed time : %ld.%06ld ms.₩n", tv_diff.tv_sec, tv_diff.tv_usec);


        print FLOPS between time check routine


        return 0;
}
```

```
/* Time function. */
int timeval_subtract(struct timeval *result, struct timeval *t2, struct timeval *t1)
{
    long int diff = (t2->tv_usec + 1000000 * t2->tv_sec) - (t1->tv_usec + 1000000 * t1->tv_sec);
    result->tv_sec = diff / 1000000;
    result->tv_usec = diff % 1000000;

    return (diff<0);
}
```