제목: 비균일 작업들의 병렬 분산 계산 (task parallelism)

1. 목적

상당수 많은 과학 계산들이 작업량이 다른 다수의 작업들을 갖고 있다. 즉 요구되는 작업의 크기들이 비균일(inhomogeneous)하는 경우도 많아서 단순한 정적 도메인 나누기로는 병렬 효율성을 높이는 데에 문제가 많다. 따라서 이번 문제에서는 기존의 단순한 정적 도메인 나누기보다는 주어진 작업량에 따라서 동적 작업량 할당하는 기법을 이용해서 병렬 효율성을 높이는 방법을 고민하고자 한다.

2. 개요

우주공간에는 물질의 73%가 암흑물질로 채워져있다. 이는 눈으로 보이지 않지만 중력을 통해서 알수 있고, 또한 우주 공간의 물질의 역사를 지배한다는 것을 그동안 과학적 계산을 통해서 알고 있다. 이러한 헤일로에 대한 연구는 우주론적 다체 시뮬레이션을 통해서 널리 수행되고 있는데, 시뮬레이션의 분해능이 커지고 병렬 컴퓨터의 성능이 증가함에 따라서 시간이 갈수록 더 큰 대용량 암흑물질 중력진화 시뮬레이션들이 수행되고 있다.
예를 들면 우주론에서 암흑물질로 이루어진 헤일로들이 우주 공간에 수 천억개가 있고 시뮬레이션을 통해서 보통 수십만에서 수 십억개의 헤일로들이 만들어지는데, 이러한 헤일로들의 물리적 성질을 구하기 위해서 헤일로의 포텐셜을 계산해야 한다. 하지만 헤일로를 이루고 있는 암흑물질의 시뮬레이션 입자들의 개수는 헤일로 질량에 따라서 수 백만배 차이가 나기 때문에 헤일로 포텐셜 계산은 매우 비균일한 작업이 된다. 따라서 이러한 과학계산에 자주 마주치게 되는 비균일 계산 수행을 병렬기법을 통해서 적절히 여러 CPU에 분산하는 것이 병렬 효율성을 높이는 데에 큰 도움이 된다.

N개의 입자들로 이루어진 구조물의 포텐셜 계산 시간은 $O(N^2)$를 따른다. 즉 100개 입자로 이루어진 경우 1초의 시간이 필요했다면, 10000개의 입자의 경우 10000초 (대략 2.8시간)가 필요하다. 즉 한번에 수행해야 하는 시간이 입자의 개수에따라 크게 다르기 때문에 (또한 대부분의 경우 이의 관계를 미리 알기 힘든 경우도 존재) 작업의 동적 할당이 매우 중요하다.

3. 목표: 병렬화 방법

여러 가지 동적 할당 방법이 있겠지만, 이번에는 가장 단순한 동적 작업할당을 생각하기로 한다. 먼저 master rank 에서는 수행해야할 작업을 생성시키고 이를 쉬고 있는 slave rank에

보내서 작업하도록 지시한다. 만약 어떤 slave rank에서 작업이 끝났다면, 이를 master rank에서 결과를 가져와서 보관한다. 만약 slave rank에서 작업들이 다 끝났다면 이를 첫 번째 rank에 알려주어서 계산 결과를 출력하고 프로그램을 마치도록 한다 (그림 1).
아래에 나와있는 순차프로그램을 수행해서 시간을 구하고 이를 병렬화한다. 여기서 입자의 위치를 정하는 것은 master rank에서만 수행하는 것으로 한다. slave rank들에서는 이러한 데이터를 받아서 중력계산 (potential 함수)을 수행하고 이 결과를 master rank에 보낸다.

4. 평가방법

그리고 병렬효율을 구해서 가장 크게 나오는 팀이 우승하는 것으로 한다. 병렬 효율은 아래와 같은 수식으로 계산한다. 이렇게 하여 가장 병렬화 효율이 좋은 팀부터 순서를 매긴다.

$$P = \frac{T_{sequential}}{T_{wall} \times N_{cpus}} \times 100$$

5. 유의사항
아래는 프로그램은 문제로 제시한 serial program의 소스코드이다. 이 프로그램을 병렬화하는 것이 목적인데, 여기에 나와있는 함수들 (potential, getrandomnp, gettime)을 변형하지 않는다. 그리고 "struct Pos"를 변형하지 않는다. 또한 random number를 생성하여 입자 데이터에 값을 넣는 부분을 해손하지 않아야 한다. 즉 serial program의 결과

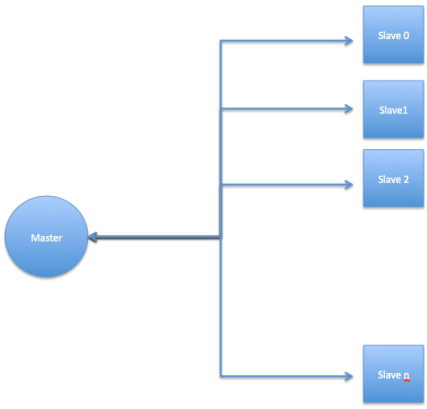

그림 1 master rank와 slave rank간의 통신 구성의 예. master rank는 필요한 작업들을 slave rank들에게 적절하게 분배하는 것이 이 테스트의 목적이다.

(totpotent 값)와 병렬 프로그램의 결과는 일치해야 한다.

첨부) serial program 예제

주의) 위 코드중에서 굵게 표시된 부분은 수정하지 않아야 한다. 즉 사용자가 미리 random number를 생성하고

C 코드

```c
#include<stdio.h>
#include<stdlib.h>
#include<stddef.h>
#include<string.h>
#include<math.h>
#include<sys/time.h>

typedef struct Pos{
        float x,y,z;
}Pos;

float ran2(long *);
long iseed=-9;

double potential(Pos *r, int np){
        int i,j;
        double potent = 0;
        for(j=0;j<np;j++){
                for(i=j+1;i<np;i++){
                        float x= r[i].x-r[j].x;
                        float y= r[i].y-r[j].y;
                        float z= r[i].z-r[j].z;
                        float dist = sqrtf(x*x+y*y+z*z);
                        if(dist > 0.) potent += 1./dist;
                }
        }
        return potent;
}

int getrandomnp(int istep, int niter){
        if(istep < niter/10) {
                return (int)(3000+5000*(ran2(&iseed)));
        }
        else {
                return (int)(100*(ran2(&iseed)));
        }
}
struct timeval tv;
float gettime(){
        static int startflag = 1;
        static double tsecs0, tsecs1;
        if(startflag) {
                (void ) gettimeofday(&tv, NULL);
                tsecs0 = tv.tv_sec + tv.tv_usec*1.0E-6;
                startflag = 0;
        }
        (void) gettimeofday(&tv, NULL);
        tsecs1 = tv.tv_sec + tv.tv_usec*1.0e-6;
        return (float) (tsecs1 - tsecs0);
}

int main(int argc, char **argv){
        int i, j;
        int np,niter,maxnp=5000000;
        Pos *r;
        double totpotent=0;

        niter = 4000*5;
        r = (Pos*)malloc(sizeof(Pos)*maxnp);
        ran2(&iseed);

        float time1, time2;
        time1 = gettime();

        for(i=0;i<niter;i++){
                np =  getrandomnp(i,niter);
                for(j=0;j<np;j++){
                        r[j].x = 2.*(ran2(&iseed))-1.;
                        r[j].y = 2.*(ran2(&iseed))-1.;
                        r[j].z = 2.*(ran2(&iseed))-1.;
                }
                totpotent += potential(r, np);
        }
        time2 = gettime();
        printf("Total potential is %20.10g in wallclock time = %g second\n",totpotent, (time2-time1));
}
```

포트란 코드

```
c234567
      function potential(x,y,z, np)
      real*8 potential
      real*4 x(np),y(np),z(np),dist,tmpx,tmpy,tmpz
      potential = 0
      do j = 1, np
      do i = j+1, np
         tmpx = x(i)-x(j)
         tmpy = y(i)-y(j)
         tmpz = z(i)-z(j)
         dist=sqrt(tmpx*tmpx + tmpy*tmpy + tmpz*tmpz)
         if(dist .gt.0) potential = potential + 1.d0/dist
      enddo
      enddo
      return
      end

      function getrandomnp(istep, niter)
      integer getrandomnp,istep,niter
      integer iseed
          common /seed/ iseed
      if(istep-1 .lt. niter/10) then
         getrandomnp = (3000+5000*ran2(iseed))
      else
         getrandomnp = (100*ran2(iseed))
      endif
      return
      end

      program main
      parameter(maxnp= 5000000)
      integer i,j,np,niter
      real*8 totpotent,potential
      real x(maxnp),y(maxnp),z(maxnp)
      real timearray(2),time1,time2,walltime
          integer getrandomnp
      external getrandomnp,potential
      external real etime
      integer iseed
      common /seed/ iseed

      totpotent = 0
      iseed = -9
      niter = 4000*5
      xxx=ran2(iseed)


      time1 = etime(timearray)
      do i = 1, niter
         np = getrandomnp(i,niter)
         do j = 1, np
            x(j) = 2.*ran2(iseed)-1.
            y(j) = 2.*ran2(iseed)-1.
            z(j) = 2.*ran2(iseed)-1.
         enddo
         totpotent = totpotent + potential(x,y,z,np)
      enddo
      time2 =  etime(timearray)
      walltime = time2-time1
      print *,'total potential =',totpotent,' and wallclock
time=',
     &            walltime
      stop
      end
```

이를 분석해서 효율성이 좋도록 데이터를 조작하는 것은 금지한다. 그리고 sequential 코드와 parallel 코드의 결과는 일치(round off error를 감안)해야 한다. 그리고 하나의 헤일로 데이터를 전체 계산 rank들에게 broadcast하여 중력계산하는 방식인 “정적할당”하는 것을 금지한다.

6 병렬화 된 PSEUDO CODE

병렬화된 코드는 아래와 같은 형식으로 되어 있을 수 있다.

```
...
if master then
    for jobs begin
        Random_Generate_Particles_of_a_halo
        while search_for_idle_slave begin
            if an_idle_slave found then
                send_particle_data_to_an_idle_slave
            else if slave_wants_to_report_potential then
                get_potential_data_from_the_slave
            endif
         end while
    end for
    while potential_report begin
        get_potential_from_a_slave
    end while
    send_terminating_signal_to_slaves
else
    send_idl_signal_to_master
    while job_request_from_master begin
        get_data_from_master
        calculate_potential_using_data
        send_potential_to_master
        send_idl_signal_to_master
    endwhile
endif
...
```