Shining Light on Dark Matter

Adventures in Supercomputing Challenge
Final Report
April 6, 2005

Bosque School
Team #010

Team Members
James Scantlan
Chloe Williams
Colin Williams

Teacher
Thomas I. Allen

Project Advisor
Christopher Hoppe
Dr. Dave Westpfahl

Table of Contents

Executive Summary

For decades, even centuries, it has been believed that some sort of matter remains in the universe that humans have been unable to confirm to exist even though it has been detected. Eventually, this matter came to be called dark matter, and was detected through the gravitational pull that this matter has on the universe as a whole even though the substance of dark matter remains a mystery to this day. In addition to the gravitational effect that dark matter has on the universe, dark matter is believed to be the source of other cosmic phenomena. If any one of these cosmic phenomena could be determined to be the result of the presence of dark matter, than from this discovery, there would be the potential to discover the substance of dark matter.

Today there are two theories of what dark matter is made up of, MACHOs or WIMPs (see background information). Each of these types of dark matter would result in different reactions, but our program was designed to work only with WIMPs. This project was originally designed to model the interaction of dark matter particles flying through interstellar cloud of cosmic gases and determine how this interaction affected the temperature of the gas clouds as a whole. However, the project that we originally intended to work with was much to complicated and time consuming for what we could accomplish in one year, so the project design was modified.

Introduction

The Issue

According to the Particle Physics and Astronomy Research Council, dark matter can be defined as “the matter which is postulated to account for at least 90% of the mass of the Universe, but which has yet to be directly detected. The evidence for its existence comes mainly from observations of the dynamics of stars in galaxies and of galaxies in clusters of galaxies, from gravitational lensing and from cosmological models. Candidates for dark matter range from brown dwarfs and black holes to weakly-interacting elementary particles such as low-mass, fast-moving neutrinos or massive, slow-moving cold dark matter particles”. Yet even with the definition at hand, the question remains, what is dark matter made up of? And why, if it is seemingly obvious that such a matter must exist, is this matter still unable to be directly detected? These are just some of the problems that scientists have faced when dealing with the cosmic phenomena that appear to be directly related to an interaction with dark matter.

So while the substance of dark matter remains a mystery, the question of what the cosmic phenomena would be without dark matter remains a mystery too. What would the temperature of interstellar clouds of atomic hydrogen be without the dark matter particles flying through the clouds? Or, how would the effect of black holes change if no dark matter were present? These answers among many others are unknown and can only be hypothesized until the mystery of dark matter is unveiled.

The Project

The original purpose of this project was to model the interaction of dark matter particles flying through interstellar clouds of cosmic gases (namely atomic hydrogen) and determine how this interaction affected the temperature of the gas clouds as a whole. In addition, we believed that after determining the specifics of this interaction we could then further determine what the dark matter particles we were working with would have had to be made up of (i.e. the density) to have this reaction occur. However, a project such as this would be much more complicated than we could complete in one year and would require much more knowledge than we had at the outset of the project. Thus we modified the purpose of the project to a much less complicated and time-consuming project.

The second version of our project was just a simplified version of the first design. Instead of determining what dark matter particles would have to be made up of to have the reaction that we did, we simply narrowed our focus to determining how the interaction of dark matter particles flying through interstellar clouds of atomic hydrogen affected the temperature of the clouds of cosmic gases.

Background Information

The Mystery Matter

Throughout history, astronomical advances have come and gone, but still one mystery has yet to be uncovered: the mystery of the substance of dark matter. Although it has not always been known as dark matter, since before Galileo, even before Copernicus, suspicions had arisen as to there being an unknown matter in the universe that could not, for an unknown reason, be seen because it emitted no visible light. As time passed, the knowledge that astronomers and astrophysicists had increased and thus the information about this matter increased to such a point as is present today. Yet still, the question remains what is this mystery matter that is known as dark matter? How can we be sure that this matter doesn't make up even 99% of the overall mass of the universe, or only 80%? The answer, at least for now, is that we can't. But we do know for certain that it does exist. Nevertheless, the term 'dark matter' still encompasses a wide range of cosmic objects, as dark matter is "any non-luminous astronomical object or particle that is detected only by its gravitational influence".

Cosmic gases

Cosmic gases are not essential to the universe as a whole; they basically just sit there, moving only when an outside factor affects them. But for some reason, these gas clouds are never found to go below a certain temperature, approximately 1000-Kelvin or725° Celsius. It has been hypothesized that this is due to an outside factor, which stirs up the particles of gas in the gas clouds and due to the friction caused by this interaction heats up the gas clouds and essentially does not let them cool down because this interaction is continuous. Most scientists believe, or hypothesize that this outside factor is the mysterious substance of dark matter. These scientists believe that without this interaction the temperature of these gas clouds would be approximately 3 Kelvin. To date, there has been no formal studies to show that this actually is the case, but even with studies there would be no way to prove or disprove this hypothesis because this interaction is always present.

Project Description

The Initial Plan

Before beginning the project we intended to create a program that would model the interaction of dark matter particles flying through interstellar clouds of cosmic gases and then modify the program to determine what the dark matter particles would have had to be made up of to acquire the reaction that we did. This, however, would have been so incredibly much more difficult than we could accomplish, so we revised our project. In addition to the difficulty level, so many factors would have had to be considered to accomplish even the smallest part of this goal that not even if we had two years could we have accomplished this.

Results

We hope to be able to complete our model, and run it until the point of no return. This point will be the point at which the data simulated by the model is either very close or exactly the same as the data which can be found in the actual clouds of atomic hydrogen. We plan to finish this by the Expo, and we will have more results to present then.

Conclusions

We do not have conclusions based on data at this time because we have not finished the project, but will have conclusions at the expo.

Most Significant Achievement

Our most significant achievement would be actually having running code that works. This would be our most significant achievement because for the longest time we didn’t have any working code and could not figure our why it was not working. While at the moment our code is not entirely done, it will be finished by the expo and thus we will have further results and conclusions to go with our completion of the code.

Acknowledgements

Challenge team 010 would like to acknowledge the following people for their contributions to this project:

Thomas I. Allen, for his support and guidance on the project;

Dr. Dave Westpfahl, for his help on our project scope and additional resources for our research;

Greg Scantlen, for providing the laptops for our cluster and initial help setting up the cluster.

References

1. Dark Matter, Cosmology, and Large-Scale Structure of the Universe. 2004 <http://www.astro.queensu.ca/~dursi/dm-tutorial/dm0.html>. Peebles, P.J. E..

2. Dark Matter, Cosmology, and Large-Scale Structure of the Universe. 2004 <http://www.astro.queensu.ca/~dursi/dm-tutorial/dm0.html>.

3. White, Martin. Dark Matter. 2004 <http://astron.berkeley.edu/~mwhite/darkmatter/dm.html>.

4. Verkindt, Didier. Neutrino History. April 11, 1999. 2004 <http://wwwlapp.in2p3.fr/neutrinos/aneut.html>.

5. Wikipedia, the free encyclopedia. Dark matter. December 6, 2004. 2004 <http://en.wikipedia.org/wiki/Dark_matter>.

6. Hernquist, Lars. TPM. 2004 <http://www.astro.princeton.edu/~bode/TPM />.

7. Krauss, Lawrence. Quintessence: The Mystery of the Missing Mass. Basic Books, 2001.

8. NRAO Glossary. National Radio Astronomy Observatory. March 29, 2005 <http://wwwmain.cv.nrao.edu/imagegallery/glossary.shtml>.

Appendices:

Appendix A: Mathematical Model

$$F = G\frac{(m_1 m_2)}{r^2}$$

This is the first of our equations, it is the universal gravitational law.

F is the force between the two particles.

m_1 and m_2 are the masses of the two particles.

G is the gravitational constant, which is approximately 6.6300 x 10^{-11} m^3 kg^{-1} s^{-2}

R is the distance between the two particles.

$$c_i''(t) = \gamma \sum 1 \le j \le n, i \ne j \, m_j \frac{c_j(t) - c_i(t)}{|c_j(t) - c_i(t)|^3}$$

This is the second of our equations, this solves for the acceleration of the i^{th} particle.

γ represents the gravitational constant, which is approximately 6.6300 x 10^{-11} m^3 kg^{-1} s^{-2} .

m_j is the mass of the j^{th} particle.

$c_j(t)$-$c_i(t)$ is the directed distance between the i^{th} particle and the j^{th} particle.

$$F_i = c_i''(t) m_i$$

This is the third of our equations, and is just the basic force equals mass times acceleration.

F_i represents the force of the i^{th} particle

c_i”(t) is the acceleration of the i^{th} particle at any given moment in time.

m_i is the mass of the i^{th} particle.

$$KE = \frac{1}{2} m_i v^2$$

This is the fourth of our equations, and is the equation for the kinetic energy of the i^{th} particle.

KE represents the kinetic energy of the i^{th} particle

m_i is the mass of the i^{th} particle

v is the velocity of the i^{th} particle

$$E = tk_b$$

This is the last of our equations, and is the equation for conversion of temperature to energy.

E is the energy of the particle

t is the temperature (approx. 1975 K)

k_b is the Boltzmann constant {approx. 1.30807×10^{-23} Joules (1 watt/second) per Kelvin}

Appendix B: Algorithms

For every particle in the simulation, we must compute the net force that is acting upon that particle. To do this, we can break the net force down into three separate forces: the forces along the x-, y-, and z-axes. We can then calculate each force separately, and compound the three after we have finished computing them. This process reduces the complexity of the program by allowing us to avoid working with vector mathematics. To calculate the net force acting upon one particle along one dimension, we can further break the problem down by calculating the force for just one other particle, then adding all these forces together. Once we have calculated the net force for one particle, we must then perform this computation for every other particle in the simulation. Once we have calculated the net force for each particle in the simulation, we can derive the acceleration of each particle from the net force. We can then add this new acceleration to the old velocity of the particle to get the new velocity for the particle. Using this new velocity, we can move each particle to its new position. Finally, we must repeat this entire process again for however long we want to progress the simulation.

Appendix C: Code

Our code performs a basic simulation of the progression of a system of particles according to their gravitational interaction. Right now it will only run on one processor. We had planned on porting it to a multi-processor environment, but we were held up by limitations imposed on us by the compilers and systems we were using to test the model with a single processor.

mainserial.cpp

```
#include <iostream>
#include <cstdlib>
#include <ctime>
#include "PARTICLE.H"

#define NUM_PARTICLES 1000  //The number of particles in the simulation
#define NUM_CYCLES 100      //The number of total cycles to progress the simulation


int main() {
  srand(static_cast<unsigned>(time(0)));
  Particle theParticles[NUM_PARTICLES];
  long double tempposx, tempposy, tempposz, tempvelx, tempvely, tempvelz, tempMass;
            long double avgvelx, avgvely, avgvelz, avgvelall;

  //Display the beginnings
  std::cout << "Particle:\tX Position:\tY Position:\tZ Position:\tX Velocity:\tY Velocity:\tZ Velocity:" <<
std::endl;
  for(int i = 0; i < NUM_PARTICLES; i++) {
    std::cout<<  i << "\t"<< theParticles[i].getPosition(DIRECTION_X) << "\t" <<
theParticles[i].getPosition(DIRECTION_Y) << "\t" << theParticles[i].getPosition(DIRECTION_Z) << "\t"<<
theParticles[i].getVelocity(DIRECTION_X) << "\t" << theParticles[i].getVelocity(DIRECTION_Y) << "\t" <<
theParticles[i].getVelocity(DIRECTION_Z) << std::endl;
  }
  std::cout << std::endl << std::endl;

  //Perform the simulation
  for(int i = 0; i < NUM_CYCLES; i++) {
    for(int j = 0; j < NUM_PARTICLES; j++) {
      //Reset the forces on this particle
      theParticles[j].resetForce(DIRECTION_ALL);
      //Loop through the rest of the particles
      for(int k = 0; k < NUM_PARTICLES; k++) {
        //Make sure we're not calculating the force for the same particle
        if(k == j) break;
        //Add the force for the other particle
        theParticles[j].addForce(DIRECTION_ALL, theParticles[k]);
      }
```

```
        }
        //The second loop ensures we calculate the forces for every particle before moving any of them
        for(int j = 0; j < NUM_PARTICLES; j++) {
          //Update the positions and velocities for the particles
          theParticles[j].update();
        }
      }
      //Display the results
      std::cout << "Particle:\tX Position:\tY Position:\tZ Position:\tX Velocity:\tY Velocity:\tZ Velocity:" << std::endl;
      for(int i = 0; i < NUM_PARTICLES; i++) {
        std::cout<<  i << "\t"<< theParticles[i].getPosition(DIRECTION_X) << "\t" << theParticles[i].getPosition(DIRECTION_Y) << "\t" << theParticles[i].getPosition(DIRECTION_Z) << "\t"<< theParticles[i].getVelocity(DIRECTION_X) << "\t" << theParticles[i].getVelocity(DIRECTION_Y) << "\t" << theParticles[i].getVelocity(DIRECTION_Z) << std::endl;
      }


              //Calculate the average velocities
              for(int i = 0; i < NUM_PARTICLES; i++) {
                                      avgvelx += theParticles[i].getVelocity(DIRECTION_X);
                                      avgvely += theParticles[i].getVelocity(DIRECTION_Y);
                                      avgvelx += theParticles[i].getVelocity(DIRECTION_Z);
              }
              avgvelx = avgvelx / NUM_PARTICLES;
              avgvely = avgvely / NUM_PARTICLES;
              avgvelz = avgvelz / NUM_PARTICLES;
              avgvelall = sqrt(abso( avgvelx * avgvelx + avgvely * avgvely + avgvelz + avgvelz ) );

              std::cout << "The average velocity of the particles is: " << avgvelall << std::endl;

      return 0;
    }
```

particle.h:

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

#define DIRECTION_X 56
#define DIRECTION_Y 57
#define DIRECTION_Z 58
#define DIRECTION_ALL 59
```

```
#define G 22.1450761 //The gravitational constant.  Units are (m^3)/(GeV*s^2)
#define HYDROGEN_MASS .938  //The mass of each hydrogen particle.  This is in GeV's.
#define DIMENSION 1000      //The dimension of the system.  It's a cube.  Higher is better, but slower. Units are meters.

class Particle {
private:
  long double posx, posy, posz; //Position variables
  long double velx, vely, velz; //Velocity variables
  double forx, fory, forz; //Force variables
  double mass;          //The mass of the particle
public:
  Particle();
  Particle(long double ipx, long double ipy, long double ipz, long double ivx, long double ivy, long double ivz, double imass);
  double getDistance(int direction, Particle o);
  void resetForce(int direction);
  void addForce(int direction, Particle o);
  void update();
  long double getPosition(int direction);
  long double getVelocity(int direction);
  double getMass(void);
};

long double abso(long double value);
```

particle.cpp:

```
#include <math.h>
#include <ctime>
#include <cstdlib>
#include "PARTICLE.H"

Particle::Particle() {
  posx = (long double)(DIMENSION * rand() / ( RAND_MAX + 1.0 ) );
  posy = (long double)(DIMENSION * rand() / ( RAND_MAX + 1.0 ) );
  posz = (long double)(DIMENSION * rand() / ( RAND_MAX + 1.0 ) );
  velx = (long double)(DIMENSION * rand() / ( RAND_MAX + 1.0 ) );
  vely = (long double)(DIMENSION * rand() / ( RAND_MAX + 1.0 ) );
  velz = (long double)(DIMENSION * rand() / ( RAND_MAX + 1.0 ) );
```

```
        mass = HYDROGEN_MASS;
    }
    Particle::Particle(long double ipx, long double ipy, long double ipz, long double ivx, long double ivy,
long double ivz, double imass) {
        posx = ipx;
        posy = ipy;
        posz = ipz;
        velx = ivx;
        vely = ivy;
        velz = ivz;
        mass = imass;
    }
    double Particle::getDistance(int direction , Particle o) {
        //direction=DIRECTION_ALL;
        double distance;
        switch (direction) {
            case DIRECTION_ALL:
                distance = sqrt( abso( ( posx - o.posx ) * ( posx - o.posx ) + ( posy - o.posy ) * ( posy - o.posy ) +
( posz - o.posz ) * ( posz - o.posz ) ) );
                break;
            default:
                distance = getPosition(direction) - o.getPosition(direction);
                break;
        }
        return distance;
    }
    void Particle::resetForce(int direction = DIRECTION_ALL) {
        switch(direction) {
            case DIRECTION_X:
                forx = 0.0;
                break;
            case DIRECTION_Y:
                fory = 0.0;
                break;
            case DIRECTION_Z:
                forz = 0.0;
                break;
            case DIRECTION_ALL:
            default:
                forx = fory = forz = 0.0;
                break;
        }
```

```
}
void Particle::addForce(int direction , Particle o) {
  //direction=DIRECTION_ALL;
  double distance = getDistance(direction, o);
  double force = ( G * getMass() * o.getMass()) / ( distance * distance ) ;
  switch(direction) {
       case DIRECTION_X:
         forx += force;
         break;
       case DIRECTION_Y:
         fory += force;
         break;
       case DIRECTION_Z:
         forz += force;
         break;
       case DIRECTION_ALL:
       default:
         addForce(DIRECTION_X, o);
         addForce(DIRECTION_Y, o);
         addForce(DIRECTION_Z, o);
         break;
  }
}
void Particle::update(void) {
  double accx = forx / mass;
  double accy = fory / mass;
  double accz = forz / mass;
  //Update velocity
  velx += accx;
  vely += accy;
  velz += accz;
  //Update position
  posx += velx;
  posy += vely;
  posz += velz;
}
long double Particle::getPosition(int direction) {
  switch(direction) {
        case DIRECTION_X:
          return posx;
          break;
```

```
          case DIRECTION_Y:
            return posy;
            break;
          case DIRECTION_Z:
            return posz;
            break;
          default:
            return -1.1;
            break;
  }
}
long double Particle::getVelocity(int direction) {
  switch(direction) {
          case DIRECTION_X:
            return velx;
            break;
          case DIRECTION_Y:
            return vely;
            break;
          case DIRECTION_Z:
            return velz;
            break;
          default:
            return -1.1;
            break;
  }
}
double Particle::getMass(void) {
  return mass;
}

long double abso(long double value) {
    if(value < 0) return ( -1 * value );
    return value;
}
```

Appendix D: The Cluster

We built a cluster of computer specifically for use with this project. Unfortunately, we have not been able to use it yet, because we have not been able to port our code to be able to work on multiple processors. When we are able to get it up and running, the cluster will be able to run Octave or MPI on anywhere from two to eight nodes.