Data Challenges in Astrophysical Simulation

Jeff Gardner

Dept. of Physics

Dept. of Astronomy

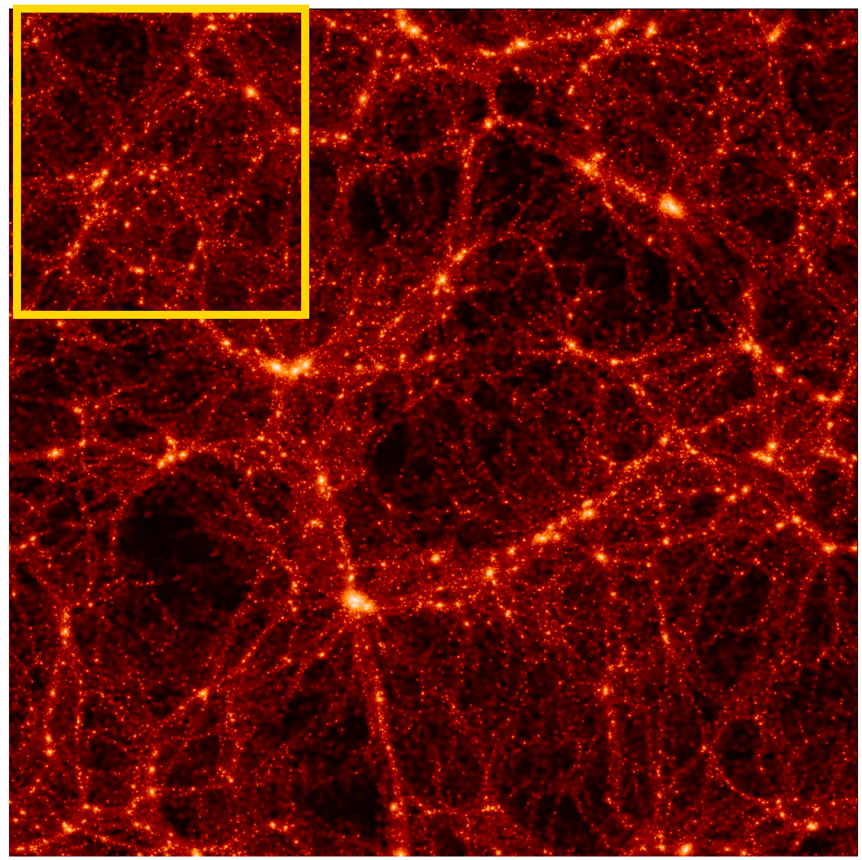
UW eScience Institute

gardnerj@phys

Galaxies: The Tip of the Iceberg

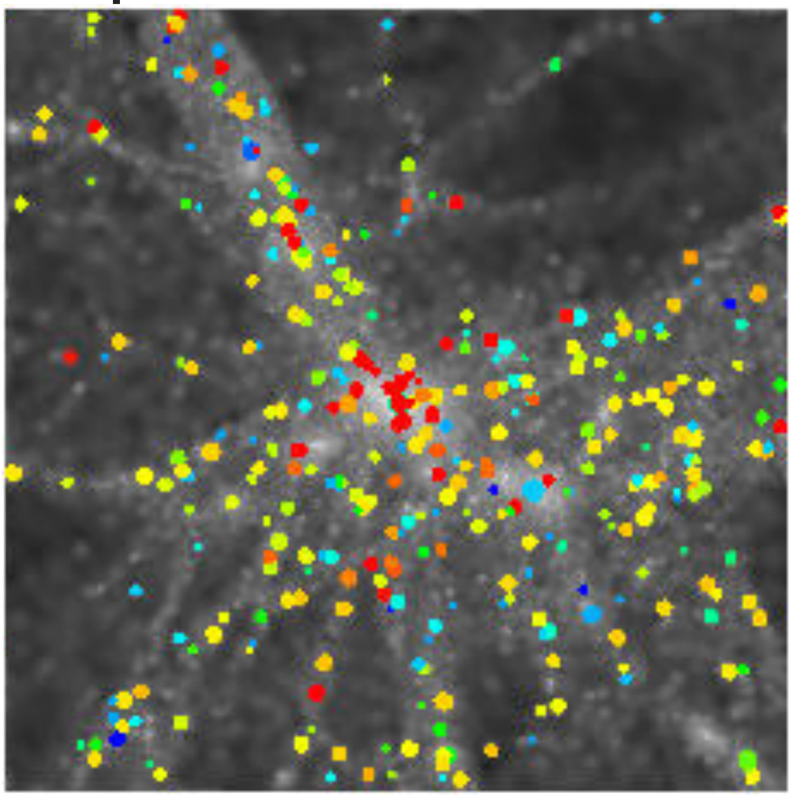
Luminous Matter
(Telescope)

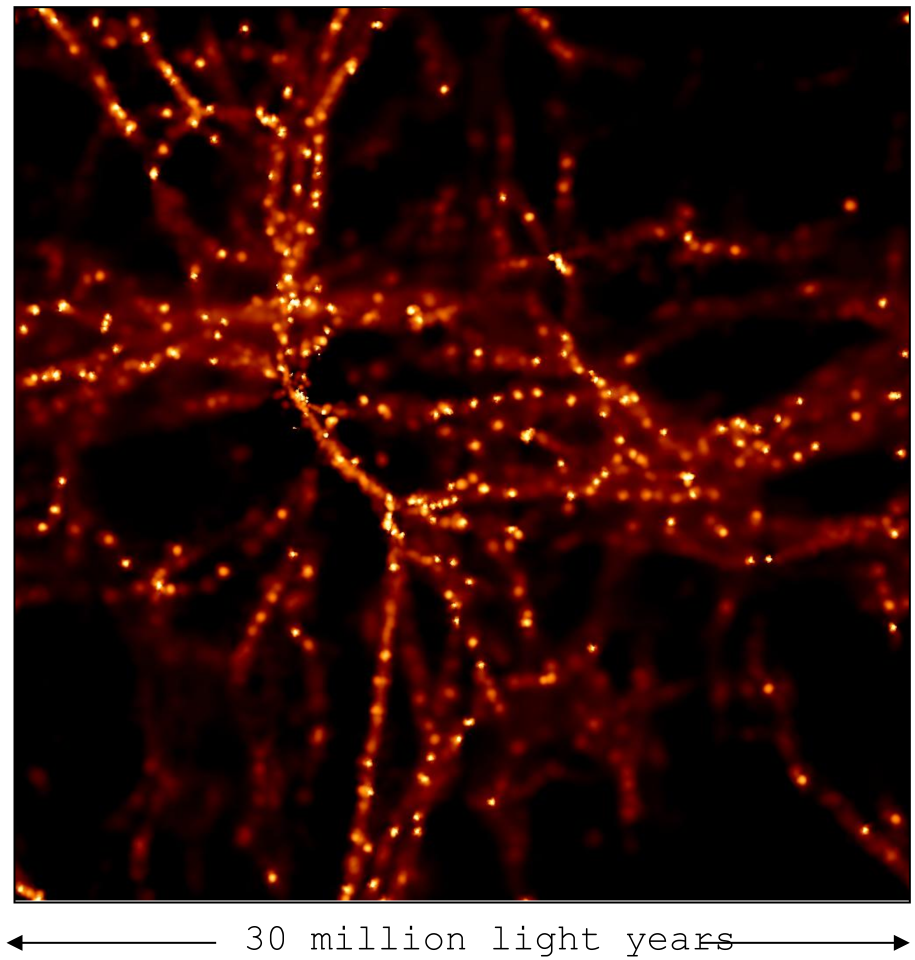
Dark Matter
(Simulation)



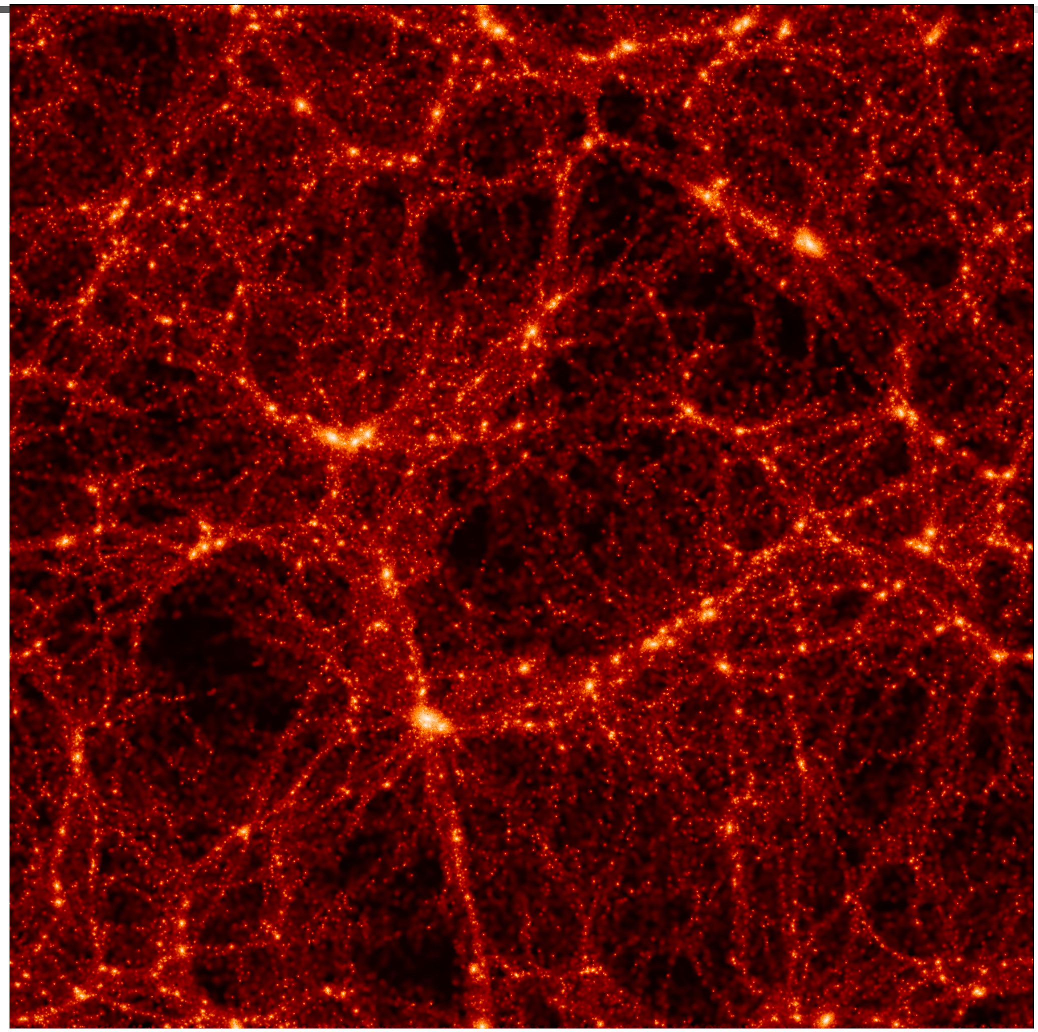
100 million light years

Galaxies: The Tip of the Iceberg





Simulations of Dark Matter tell us what the Universe really “looks like”



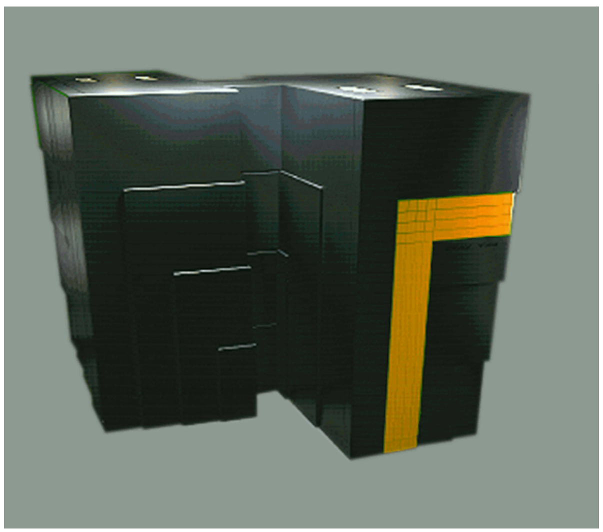
100 million light years

Dataset properties

- 10^6 to 10^{11} particles
  - Dark matter
  - Stars
  - Gas
- Each particles has roughly a dozen properties:
  - Position
  - Velocity
  - Mass
  - Density
  - Temperature

How to turn astrophysics simulation output into scientific knowledge

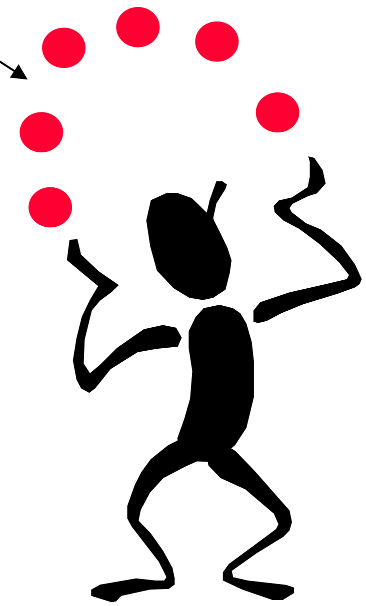
Using 300 processors:
(circa 1995)



Step 1: Run simulation

Step 2: Analyze simulation on workstation

(happy scientist)



Step 3: Extract meaningful scientific knowledge

Data Analysis

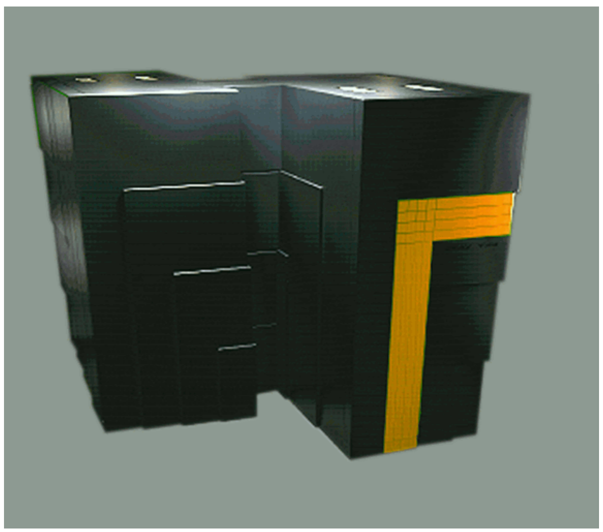
- Each simulation generates many “snapshots”.
- Each snapshot is a single file.
- To analyze, astrophysicists write programs in C or Fortran.
- Usually, these programs read in an entire snapshot, then operate on that snapshot in memory.

Why analyze in RAM?

1. Dataset is tightly-coupled
    1. Operations are typically not data-parallel
    2. Cannot break up a snapshot into smaller pieces to be analyzed separately
2. One rarely selects subsets of data
    1. It's hard for a DBMS to minimize I/O when you need everything anyway
    2. When subset selection is possible, it tends to be in non-trivial ways
3. *Lots of math*
    1. Analysis typically utilizes fairly complex analytical models.
    2. Historically, a highly optimizable compiled language has been required

How to turn astrophysics simulation output into scientific knowledge

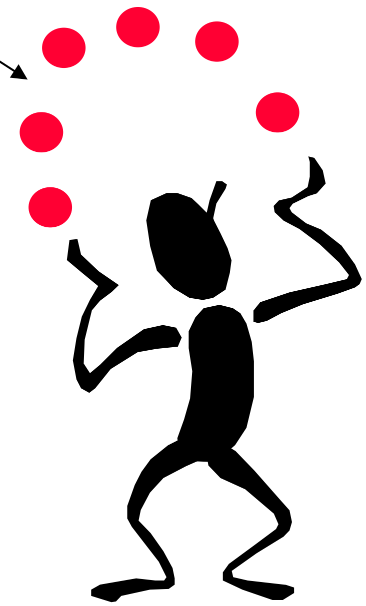
Using 300 processors:
(circa 1995)



Step 1: Run simulation

Step 2: Analyze simulation on workstation

(happy scientist)



Step 3: Extract meaningful scientific knowledge

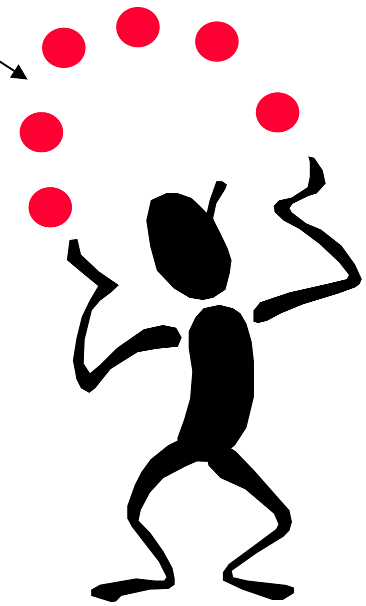
How to turn astrophysics simulation output into scientific knowledge

Using 1000 processors:
(circa 2000)

(happy scientist)

Step 1: Run simulation

Step 2: Analyze simulation on server (in serial)



Step 3: Extract meaningful scientific knowledge

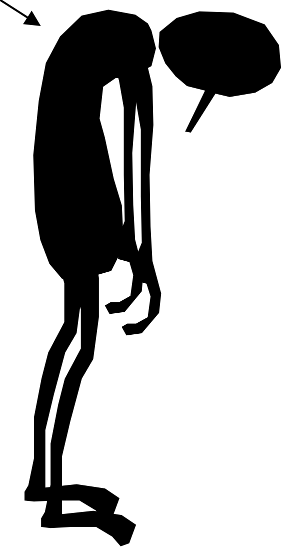
How to turn astrophysics simulation output into scientific knowledge

Using 10,000 cores:
(circa 2008)

Step 1: Run simulation

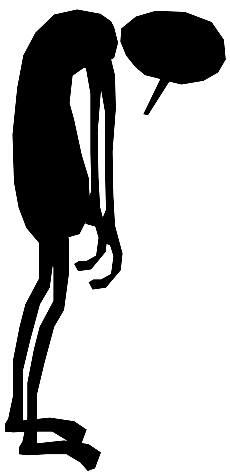
Step 2: Analyze simulation on ???

(unhappy scientist)



Exploring the Universe can be (Computationally) Expensive

- The size of simulations is no longer limited by computational power
- It is limited by the parallelizability of data analysis tools
- This situation is only getting worse.



Exploring the Universe can be (Computationally) Expensive

- The size of simulations is no longer limited by computational power
- It is limited by the parallelizability of data analysis tools
- This situation is only getting worse.
    1. Not only are we limited by the size of shared RAM
    2. We are also limited by I/O

(In fact, CPU speed is almost a second-order effect)

How to turn astrophysics simulation output into scientific knowledge

Using 10,000 cores:
(circa 2008)

Step 1: Run simulation

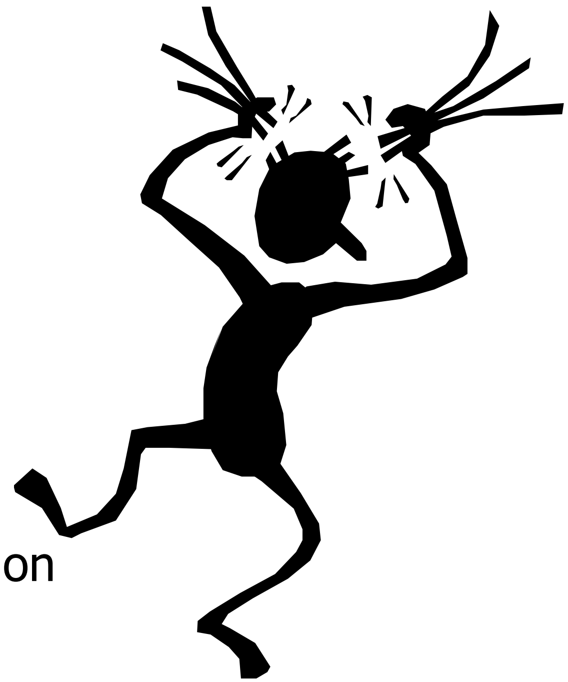
Step 2: Analyze simulation on ???

(unhappy scientist)

How to turn astrophysics simulation output into scientific knowledge

Using 500,000 cores?:

(circa 2012)



Step 2: Analyze simulation on ???

Step 1: Run simulation

(Single snapshot: 200TB)

By 2012, we will have machines that will have millions of processor cores!

The Challenge of Data Analysis in a Massively Parallel Universe

- Parallel programs are expensive to write!
  - Lengthy development time
- Parallel world is dominated by simulations:
  - Code is often reused for many years by many people
  - Therefore, you can afford to invest lots of time writing the code.
- *Example:* GASOLINE (a cosmology N-body code)
  - Required *10 person-years* of development
- Data Analysis does not work this way:
  - Rapidly changing scientific queries
  - Queries are specific to individual researchers
  - Less code reuse

Speed of scalable application development = speed of science

The fundamental challenge:

1. Physicists and astronomers do lots of math, and have historically required a *language*:
   1. Flexible, general-purpose
      1. A more special-purpose language is usually too restrictive.
   2. Procedural
      1. Other paradigms tend to be slower, although OO compilers are getting pretty good
   3. Imperative
      1. Math-driven view of computation
2. Despite hundreds of attempts, *nobody* has developed a general-purpose imperative programming language.

- For this reason, programs are written using message-passing

OK, so what's different this time?

1. Although the math is still there, CPUs are so fast that floating-point performance is becoming less important.
2. Data volume and I/O bandwidth are the main limiting factors.
3. The scientist can adopt a more data-driven view of their workflow (i.e., not math-driven and imperative).

Summary

- "*High-Performance Computing*" (HPC) is what we have been doing for the last 20 years
- Now we are entering the era of *Data Intensive Scalable Computing* (DISC)
- Implicit in DISC is the minimization of development time.
  - How do I express my scientific workflow to the computer so that it can optimize it in a scalable manner?
- The human component is what differentiates DISC from HPC:
  1. Need, on scalable resources, for *short development times*.
  2. Need, on scalable resources, for *interactivity*.