

# Knowledge Discovery in a Virtual Universe



Data Intensive Scalable Computing  
strategies for large astrophysical datasets

Jeffrey P. Gardner  
Andrew Connolly  
Cameron McBride

*Pittsburgh Supercomputing Center  
University of Pittsburgh  
Carnegie Mellon University  
University of Washington*



# How to turn astrophysics simulation output into scientific knowledge

**Using 300 processors:**  
(circa 1995)



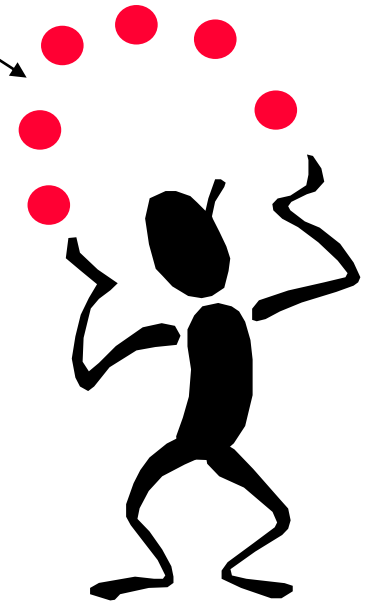
**Step 1:** Run simulation



**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)



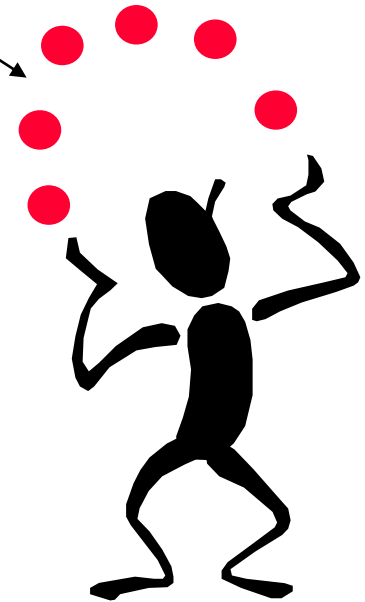
**Step 1:** Run simulation



**Step 2:** Analyze simulation  
on server (in serial)



(happy scientist)



**Step 3:** Extract meaningful  
scientific knowledge

# How to turn astrophysics simulation output into scientific knowledge

Using **10,000 cores:**  
(circa 2008)

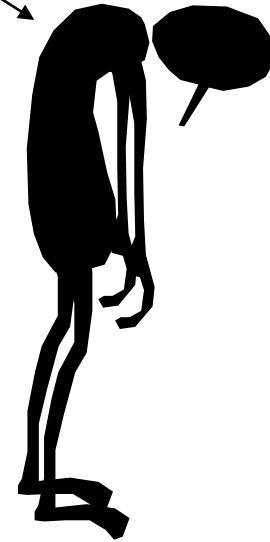


**Step 1:** Run simulation



**Step 2:** Analyze simulation  
on ???

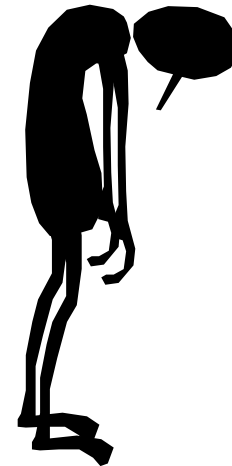
(unhappy scientist)



# Exploring the Universe can be (Computationally) Expensive

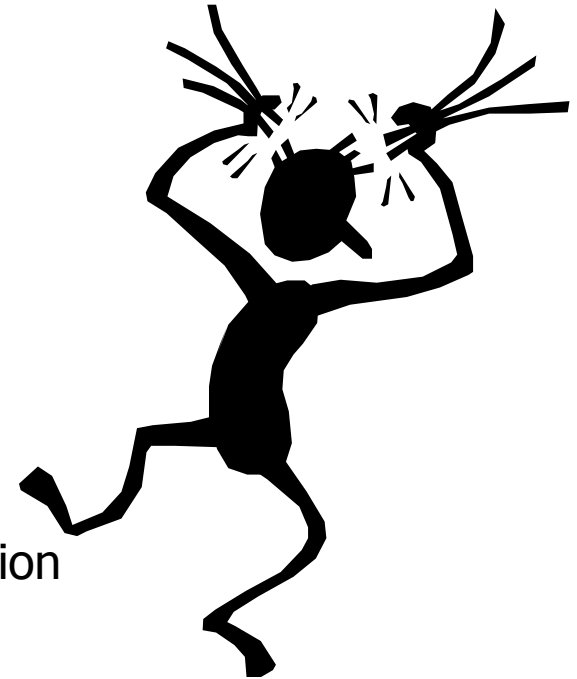
---

- The size of simulations is no longer limited by computational power
- It is **limited by the parallelizability of data analysis tools**
- This situation, will only get worse in the future.



# How to turn astrophysics simulation output into scientific knowledge

Using **1,600,000 cores?**:  
(circa 2012)

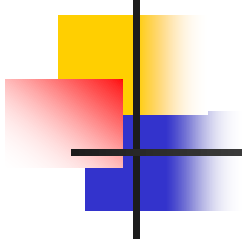


**Step 1:** Run simulation

**Step 2:** Analyze simulation  
on ???

(Single snapshot: 200TB)

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



# Astrophysical applications are often tightly-coupled

---

- Data and computational domains overlap in the extreme
- A single node may need to access data on most (if not all) other nodes during the computation
- Examples:
  - Group finding
  - N-Point correlation functions
  - New object classification
  - Density estimation

# The Challenge of Data Analysis in a Multiprocessor 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 FTE-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*







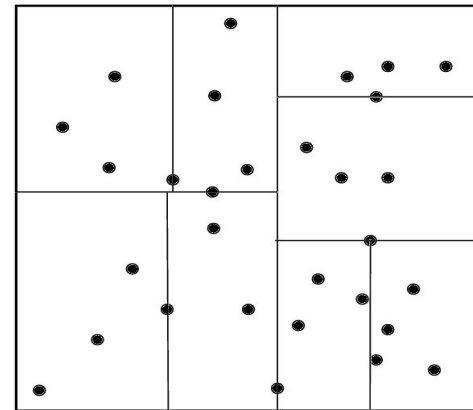
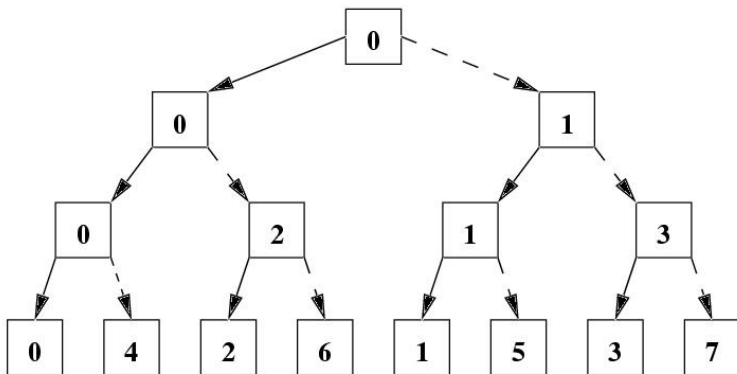
# Research Questions

---

1. Can we substantially reduce the development time of tightly-coupled astrophysical data analysis applications?
2. Can we do this using tools that are
  1. Familiar to HPC researchers
  2. Able to run on existing HEC platforms

# The Challenge of Astrophysics Data Analysis in a Multiprocessor Universe

- Astrophysics uses dynamic, irregular data structures:
  - Astronomy deals with point-like data in an N-dimensional parameter space
  - Most efficient methods on these kind of data use **space-partitioning trees**.
  - The most common data structure is a **kd-tree**.





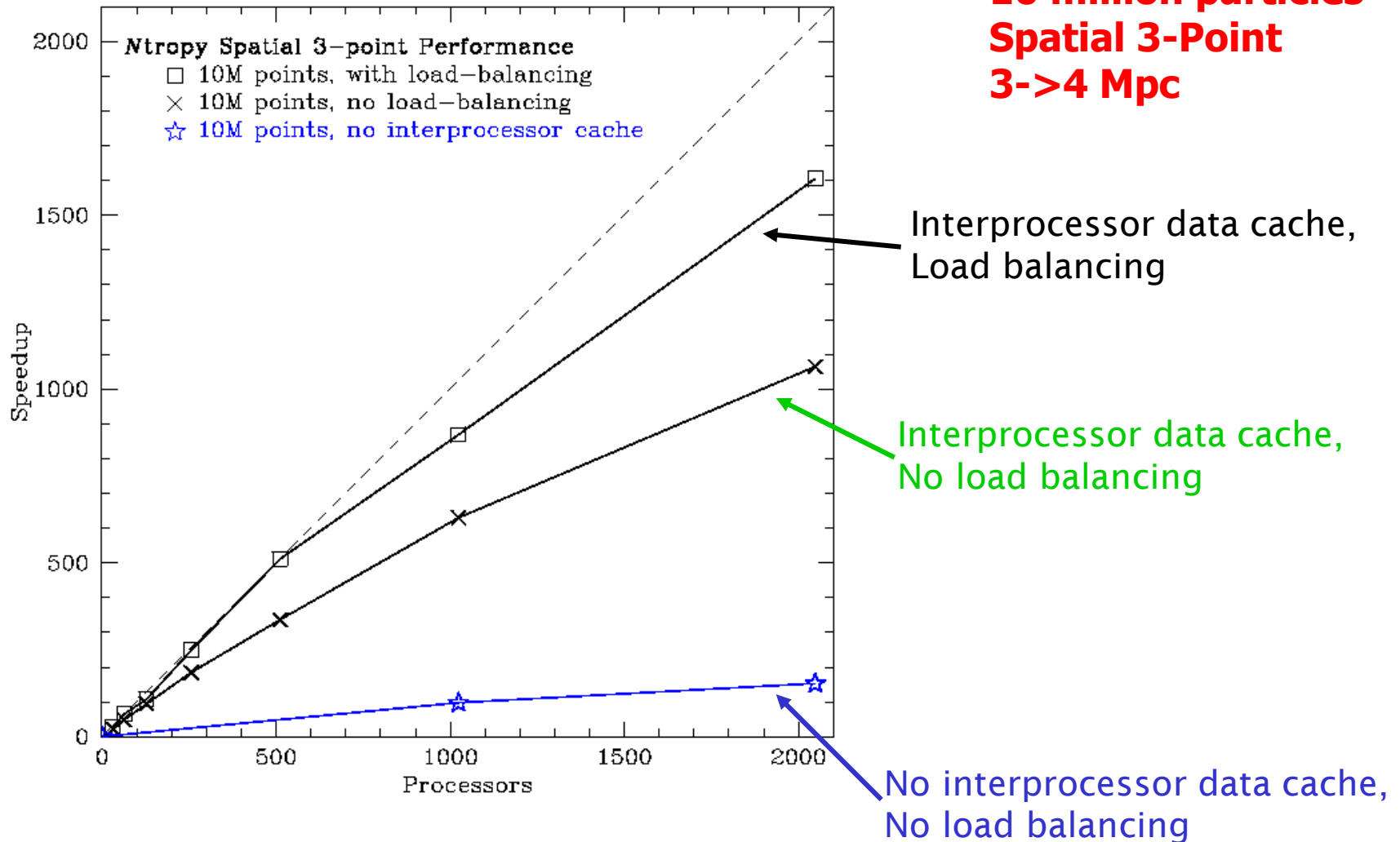
# The Challenge of Astrophysics Data Analysis in a Multiprocessor Universe

---

- Build a **targeted** library for distributed-memory kd-trees that is scalable to *thousands* of processing elements
  - Lightweight
  - Easy to learn
  - Language independent
  - Platform independent
- Library is application specific
  - Useful only for low-dimensional kd-trees

# Ntropy Performance

**10 million particles  
Spatial 3-Point  
3->4 Mpc**





# N trophy “Meaningful” Benchmarks

---

- The purpose of this library is to minimize development time!
- Development time for:
  1. *Parallel* N-point correlation function calculator
    - 2 years -> 3 months
  2. *Parallel* Friends-of-Friends group finder
    - 8 months -> 1 month



# Conclusions

---

- An implicit assumption in Data Intensive Scalable Computing is the minimization of development time.
- 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*.
- Deployment of lightweight libraries targeted towards specific domains is a viable means of enabling DISC.