***N*Tropy**: A Framework for Analyzing Massive Astrophysical Datasets

Harnessing the Power of Parallel Grid Resources for Astrophysical Data Analysis

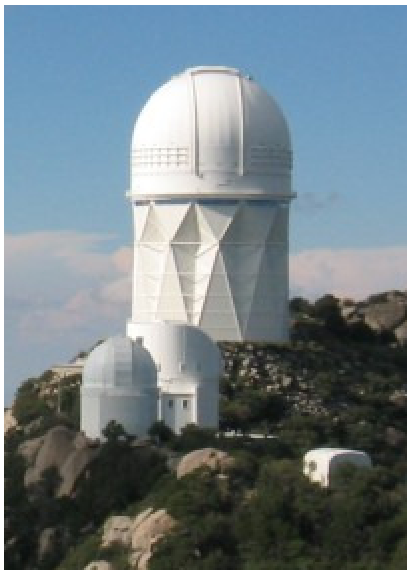
Jeffrey P. Gardner
Andrew Connolly
Cameron McBride

Pittsburgh Supercomputing Center
University of Pittsburgh
Carnegie Mellon University

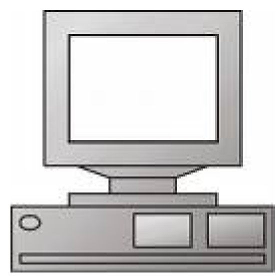
PITTSBURGH SUPERCOMPUTING CENTER

TERAGRID

How to turn observational data into scientific knowledge

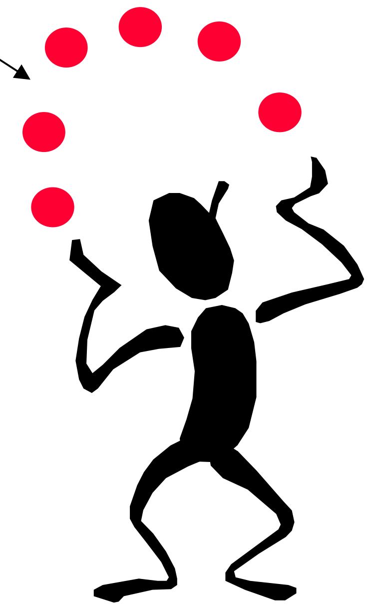


Step 1: Collect data



Step 2: Analyze data
on workstation

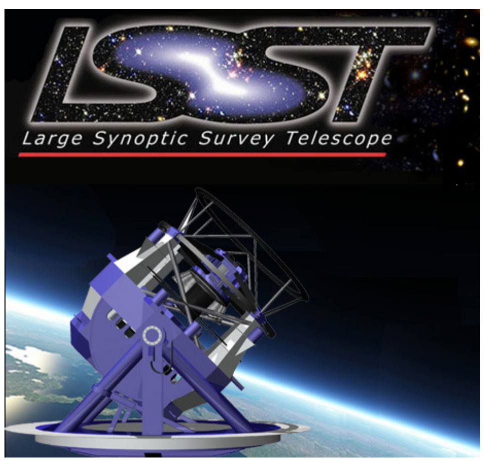
(happy astronomer)

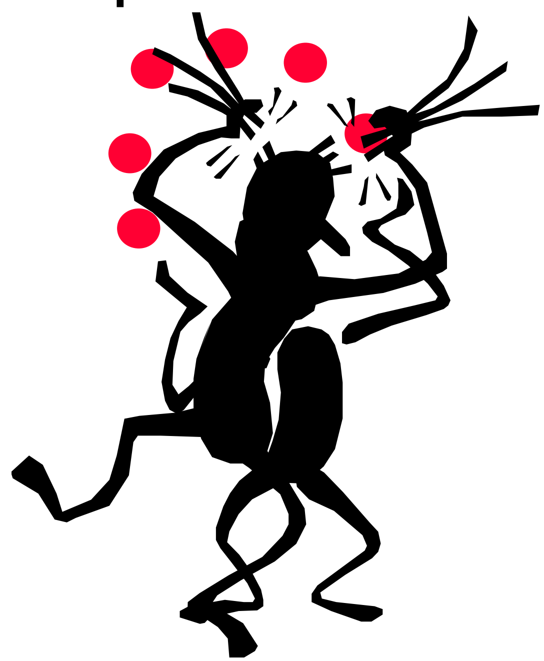


Step 3: Extract meaningful
scientific knowledge

The Era of Sky Surveys

- Paradigm shift in astronomy: Sky Surveys
  - Available data is growing at a much faster rate than computational power.



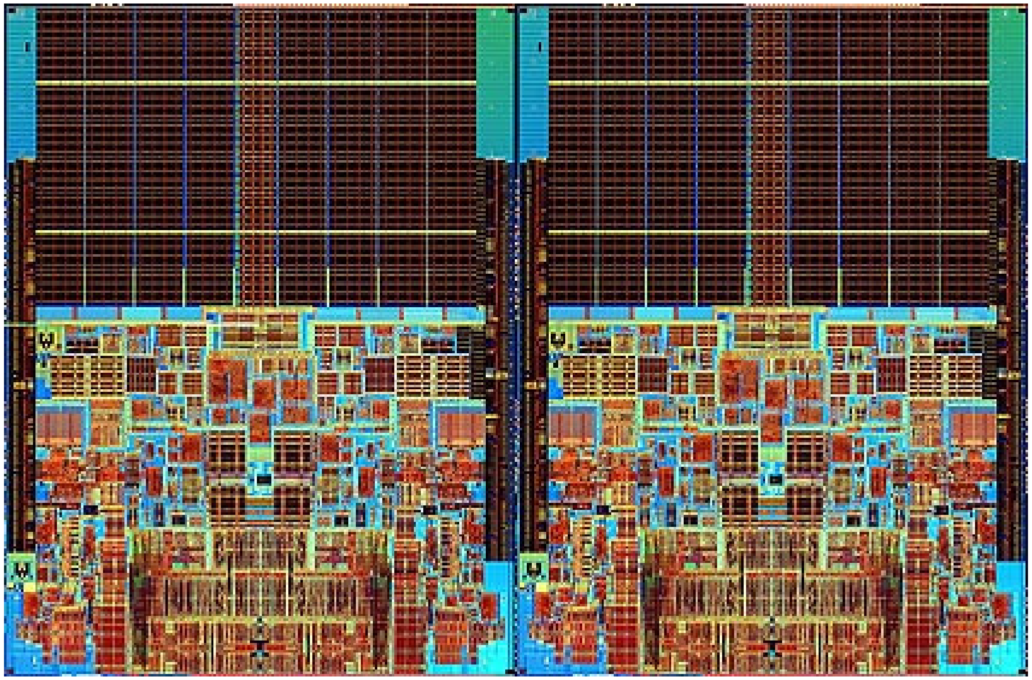


Mining the Universe can be (Computationally) Expensive

- In the future, there will be many problems that will be *impossible* without multiprocessor resources.
- There will be many more problems for which throughput can be substantially enhanced by multiprocessor resources.

Mining the Universe can be (Computationally) Expensive

- In 5 years, even your workstation may have *80 cores!!*



Intel's 4-core Quadro

Good News for "Data Parallel" Operations

- Data Parallel (or "Embarrassingly Parallel"):
  - Example:
    - 1,000,000 QSO spectra
    - Each spectrum takes ~1 hour to reduce
    - Each spectrum is computationally independent from the others
  - There are many workflow management tools that will distribute your computations across many machines.

Tightly-Coupled Parallelism
(what this talk is about)

- Data and computational domains overlap
- Computational elements must communicate with one another
- Examples:
  - Group finding
  - N-Point correlation functions
  - New object classification
  - Density estimation

The Challenge of Data Analysis in a Multiprocessor Universe

- Parallel programs are difficult to write!
  - Steep learning curve to learn parallel programming
- 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

The Challenge of Data Analysis in a Multiprocessor Universe

- Data Analysis does not work this way:
  - Rapidly changing scientific inqueries
  - Less code reuse
- Simulation groups do not even write their analysis code in parallel!
- Data Mining paradigm mandates rapid software development!

The Challenge of Data Analysis in a Multiprocessor Universe

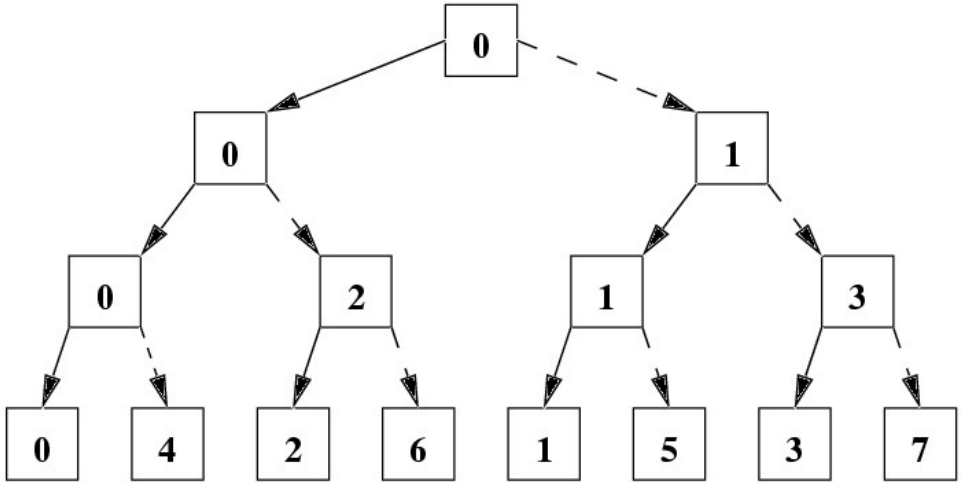
- Build a framework that is:
  - Sophisticated enough to take care of all of the nasty parallel bits for you
  - Flexible enough to be used for your own particular data analysis application

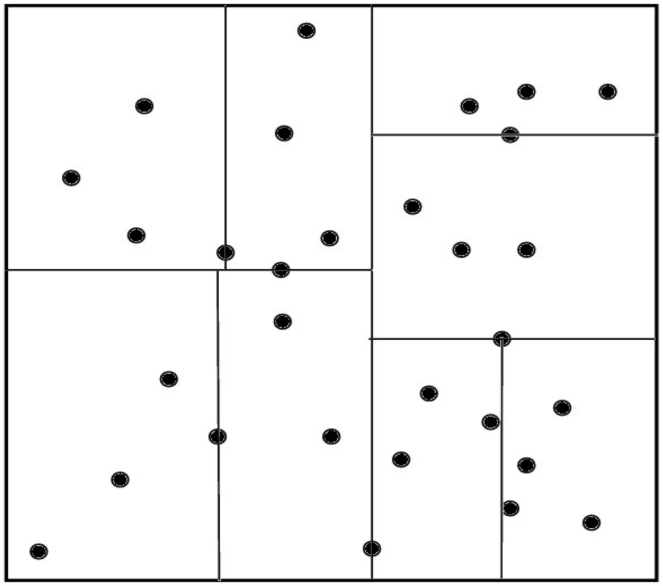
N tropy: A framework for multiprocessor development

- GOAL: Minimize development time for parallel applications.
- GOAL: Enable scientists with no parallel programming background (or time to learn) to still implement their algorithms in parallel by writing only serial code.
- GOAL: Provide seamless scalability from single processor machines to massively parallel resources.
- GOAL: Do not restrict inquiry space.

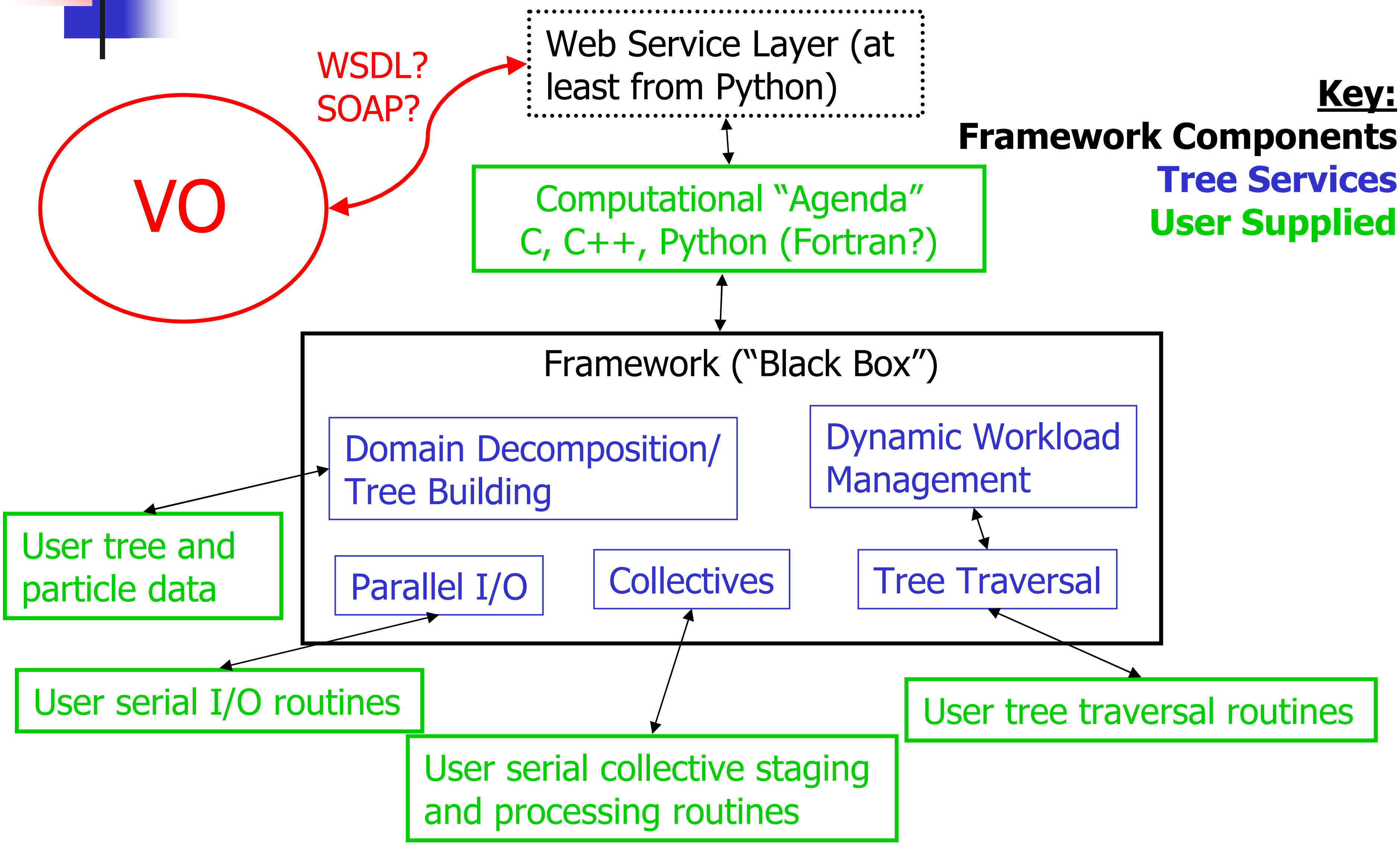
N tropy Methodology

- Limited 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.
- Limited Methods:
  - Analysis methods perform a limited number of fundamental operations on these data structures.

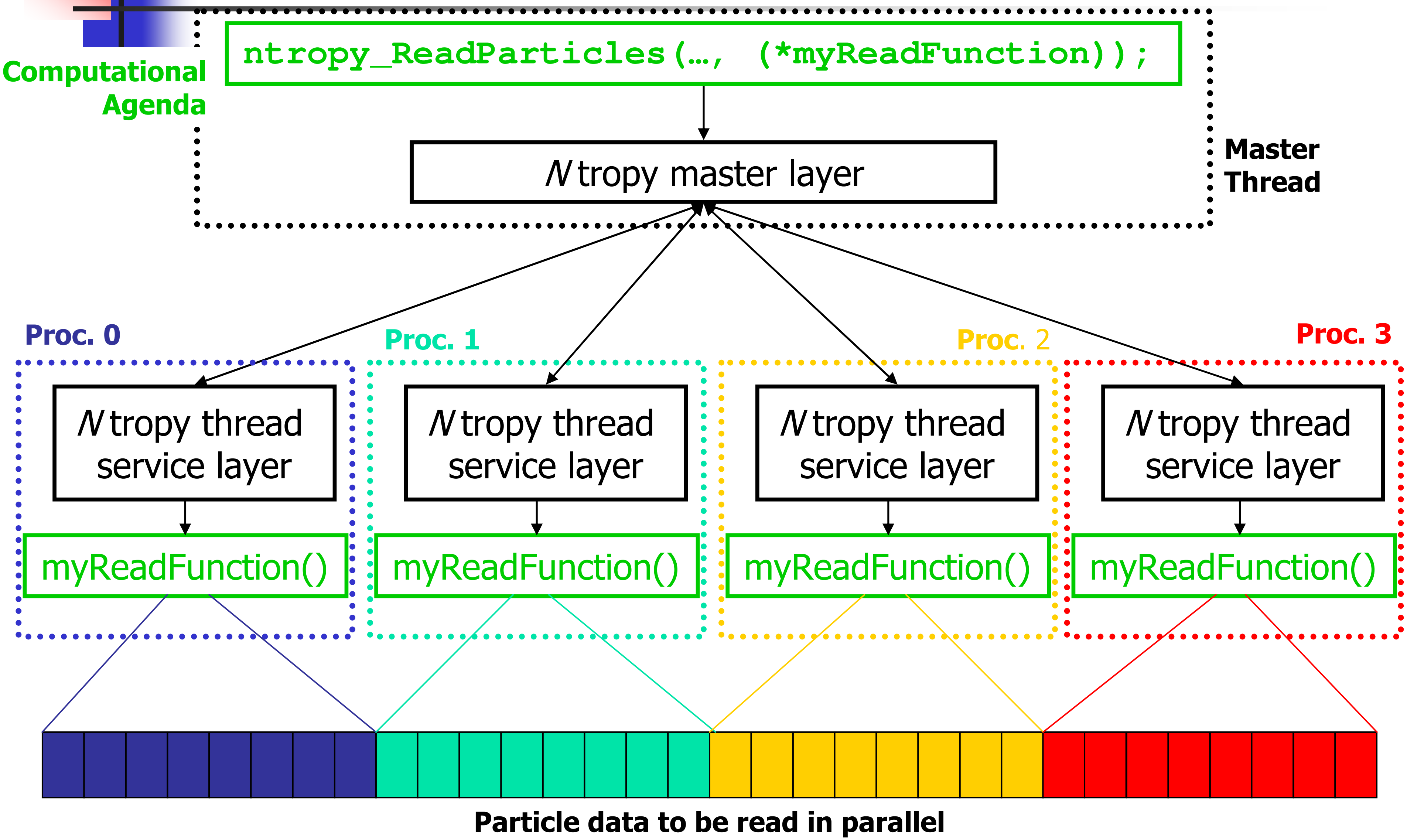




N tropy Conceptual Schematic



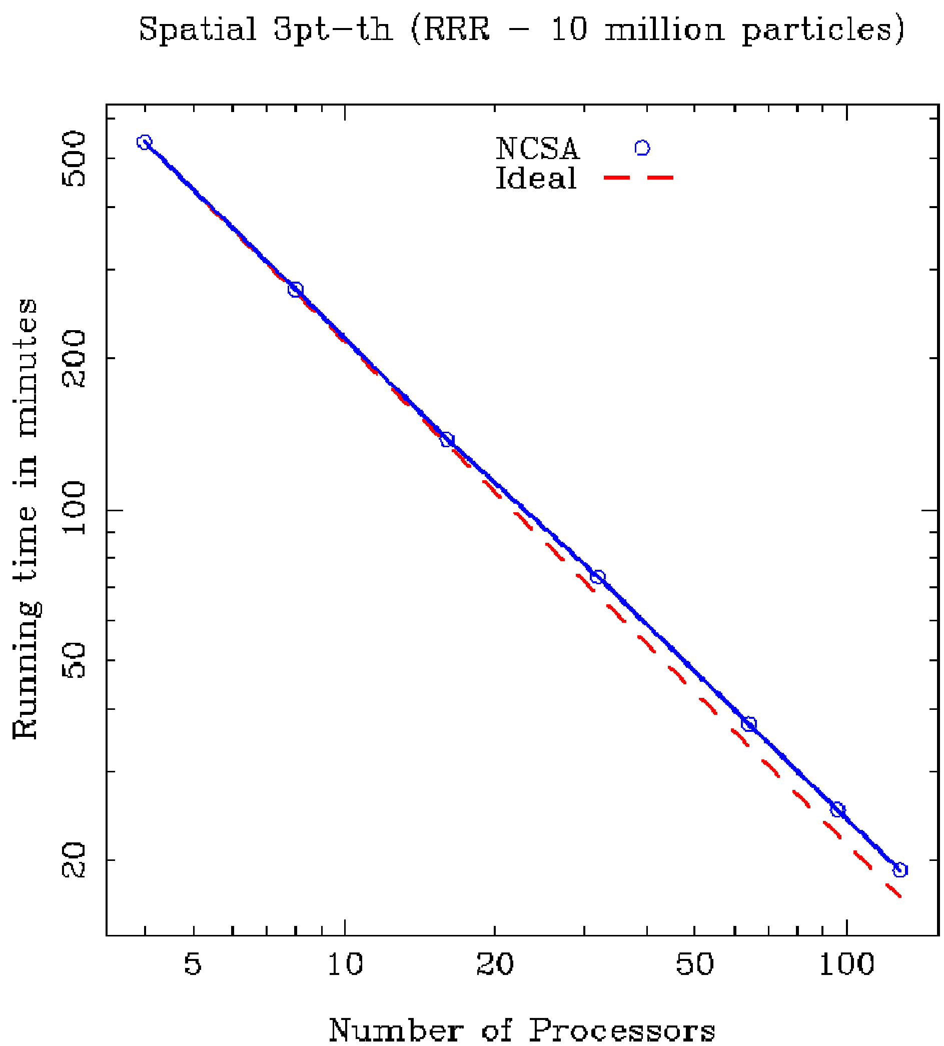
A Simple *N*tropy Example



N tropy Performance

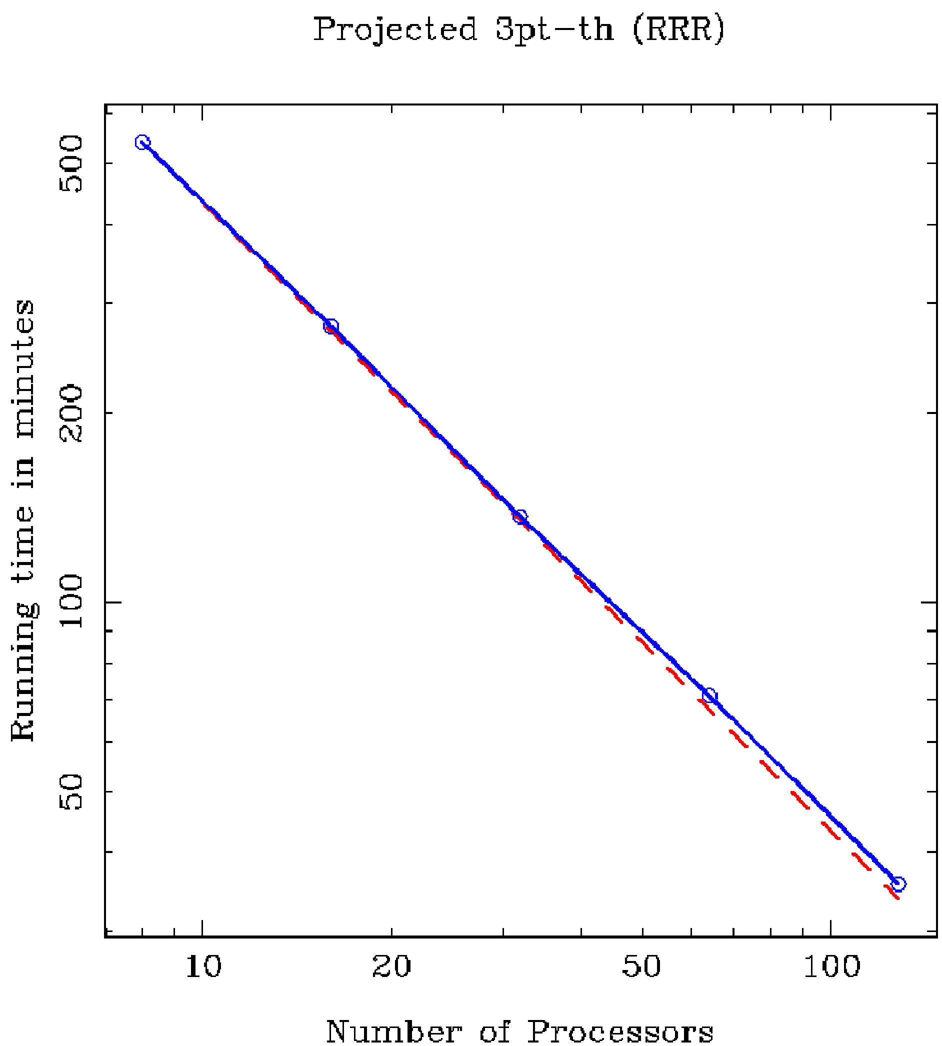
10 million particles
Spatial 3-Point
3->4 Mpc

(SDSS DR1 takes less than 1 minute with perfect load balancing)



N tropy Performance

10 million particles
Projected 3-Point
0->3 Mpc



Serial Performance

- *N* tropy vs. an existing serial n-point correlation function calculator:

N tropy is 6 to 30 times faster *in serial!*

- In short:

1. Not only does it takes much less time to write an application using *N* tropy,
2. You application may run *faster* than if you wrote it from scratch!

N tropy “Meaningful” Benchmarks

- The purpose of this framework is to minimize development time!
- Development time for:
  1. *Parallel* N-point correlation function calculator
     - 3 months
  2. *Parallel* Friends-of-Friends group finder
     - 3 weeks
  3. *Parallel* N-body gravity code
     - 1 day!*

*(OK, I cheated a bit and used existing serial N-body code fragments)

Conclusions

- Astrophysics, like many sciences, is facing a deluge of data that we must rely upon multiprocessor compute systems to analyze
- With *N* tropy, you can develop a tree-based algorithm in less time than it would take you to write one from scratch:
  1. The implementation may be even faster than a "from scratch" effort
  2. It will scale across many distributed processors
- More Information:
  - Go to Wikipedia and seach "Ntropy"