Motivations & Confessions

Biologist’s Perspective

Bacteria Appreciation

10 quadrillion human cells vs. 100 quadrillion bacterial cells

The upside of diarrhea

It’s a small world after all, in the soil, in the oceans …

Millions of bacteria species, plus the strains

Currently ~300 complete genomes vs less than 20 eukaryotic genomes available

Much can be learned from these, and more are being sequenced every week

1000 genomes in ~2008

So where do these bacteria hide these secrets?

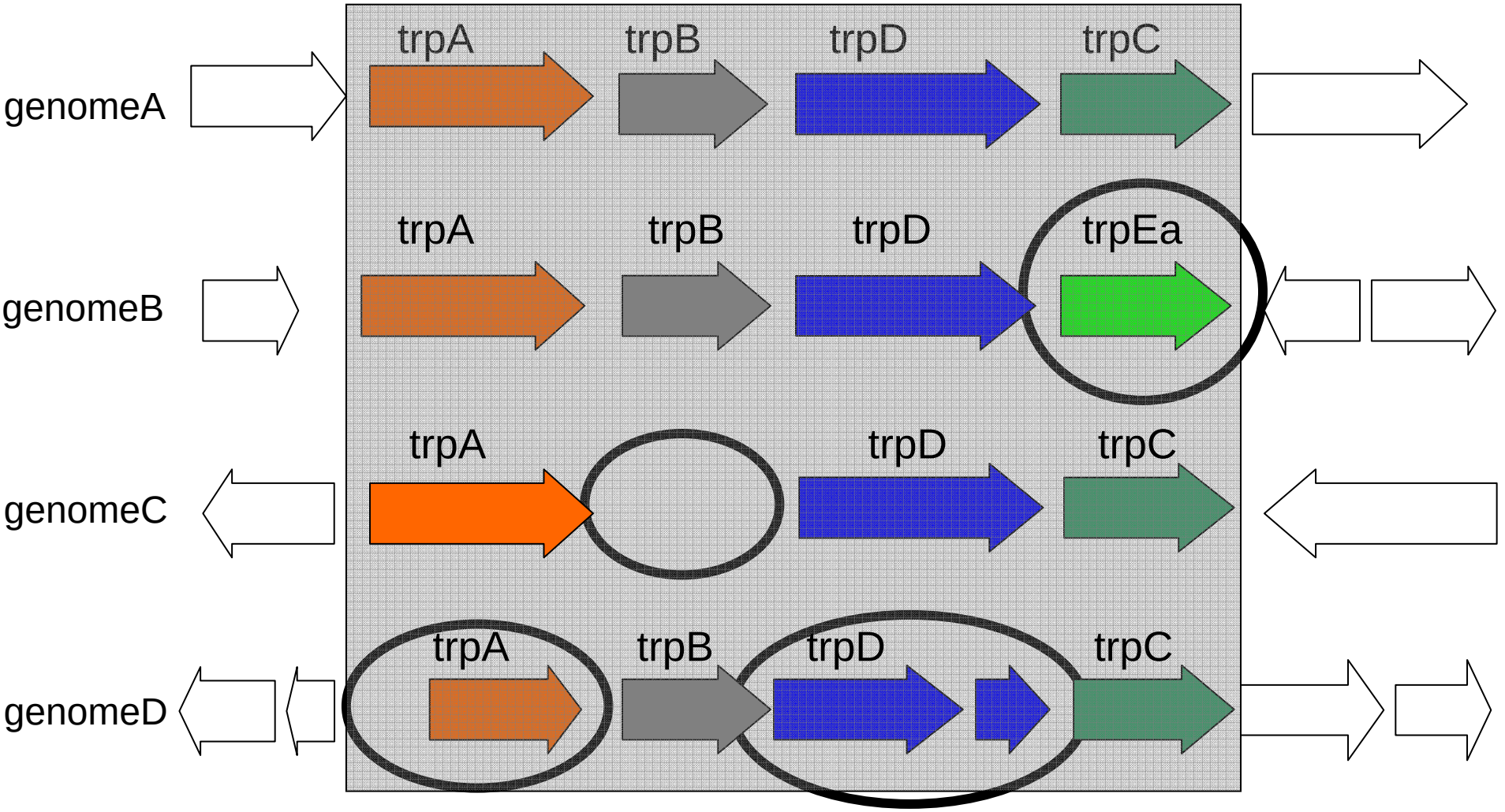
Beautiful Nuggets of TRUTH

In a thing we called CLUSTERS!

So what's a CLUSTER?

"functionally-related set of genes that code for proteins that perform a role in the same biological process or are sub-units in the same structure and are located near each other in the DNA"

Conserved Clusters



Overbeek *et al, PNAS, 1999*

The SIGNIFICANCE of CLUSTERS:

Genomic context gives us insight to unknown gene functions

~50% of genes in bacteria (euks don’t), good chance your gene clusters somewhere!

Why do they cluster?

Common regulation, Lateral Gene Transfer, Protein-Protein Interactions?

Am I in the right room?

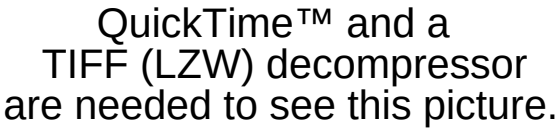
Detecting CLUSTERS requires computing the SIMILARITY between every protein in our database.

~2.5 million protein sequences in our DB

BLAST x Bunch of Computers = Similarities

.123 ms/per char X 440 char/per seq X 2.5M

> 4.25 years for a single processor


QuickTime™ and a
TIFF (LZW) decompressor
are needed to see this picture.

Drug Target Pipeline of NMPDR

Curators harvest proteins of interest from the literature

Do computations to determine if its druggable

Identify characteristics of a druggable pocket:
size, shape, key residues

Suggest more focused candidates for chemical screening and docking applications

Exploitable Observations*

Proteins structures descending from a common ancestor are remarkably similar

Residues in the active sites of these proteins are under evolutionary pressure to maintain their functional integrity, and therefore do not mutate as frequently as less functionally important residues

*J. Mol.Biol.(1996) 257, 342-358, Lichtarge et al

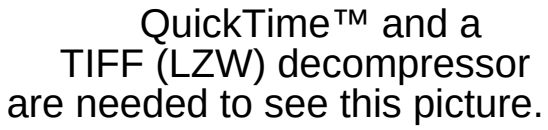
Some of the Tools

SCOPmap provides us with structural classifications of our targets.

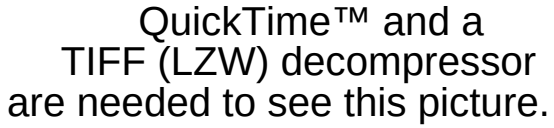
We take sets of proteins with the same structural classification and a representative structure as input to a evolutionary trace program.

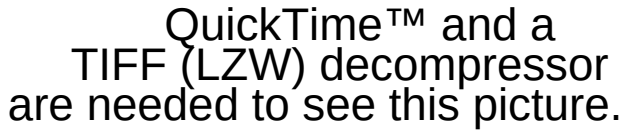
The ET predicts binding pockets and key residues


QuickTime™ and a
TIFF (LZW) decompressor
are needed to see this picture.



*J. Mol.Biol.(1996) 257, 342-358, Lichtarge et al





*J. Mol.Biol.(1996) 257, 342-358, Lichtarge et al

Problem Solving Approach

- Problem
  - Applications Consist of many distinct but related computations.
  - Use same data base, same code, output collected and integrated
  - Computations are independent, time not consistent
  - Tens of thousands of computations
  - Multiple heterogeneous clusters available
    - Different schedulers
    - Different security models
- Challenge
  - Efficiently distribute the computations across the resources, monitor their execution and collect the output.

Problem Solving Approach

- Our requirements:
  - Dynamically acquire resources during the run
  - Provide different credentials
  - Detect failed jobs, rerun them
  - Distribute a private database to all computations
- In other words, we need a Grid Metascheduler

Grid Metascheduler

- Two examples
  - Community Scheduler Framework
    - Platform Inc. provides framework for building community scheduler
    - No metascheduler deployed on our resources
  - GT3 Blast Grid Service
    - Excellent for single researcher with small set of sequences
    - Static allocation
    - Assumes single credentials
    - Assumes small number of sequences (We have 2.4 million)
    - Assume use of public NR (We have our own)

Traditional Scheduling

- Divide the input in to N pieces and schedule N jobs, or schedule 1 job on N processors
  - N can be too large (10's of thousands)
  - Inefficient distribution of reference datasets
  - Failure of an individual computation causes entire job to fail
  - Difficult to prevent time-outs while still requesting reasonable amounts of time.

The self Scheduling Model

- Divide the computations into N pieces of work (The pool of work)
- Create a master that owns the work and hands it out, one piece at a time, to workers that ask for it. Master keeps track of available, completed and to-be-finished work
- Create clients that ask for units of work, runs the computation, delivers results and ask for more work.
- We call this the Askfor
- A variant of traditional master-worker

Askfor

- Use each cluster scheduler to dynamically schedule as many workers as possible as often as possible.
- Scheduling scheme depends on individual scheduler policies and keen observation of the scheduler queue dynamics over time.
- Each worker runs for the full scheduled time, continually asking for work from the master and returning solutions.

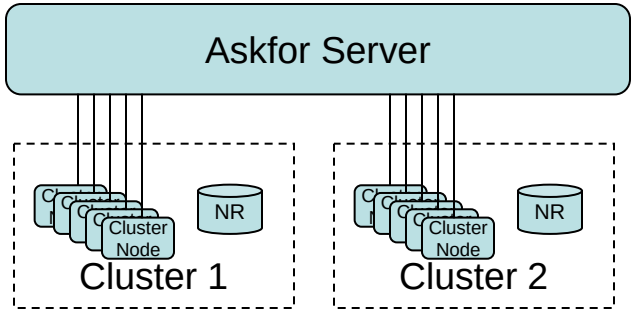
Resources

- Jazz Cluster at ANL
  - 350 nodes
    - 2.4GHz Intel Xeon
    - 1 - 2 GB RAM
  - 10TB Shared disk Global File System (GFS)
  - 10TB Parallel Virtual File System (PVFS)

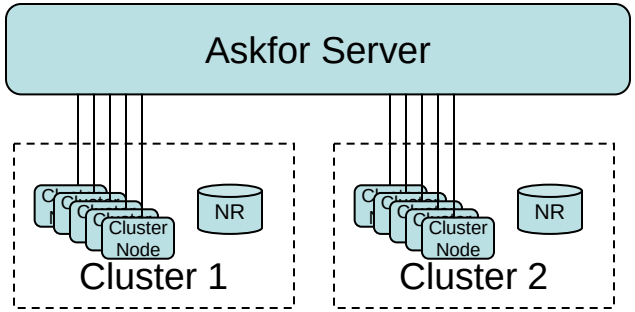
Resources

- Teragrid Distributed Clusters
  - 9 clusters, at different sites
  - 40Gbs network
- Typical Cluster (ANL)
  - 62 Nodes
    - Dual 1.3 GHz Intel Itanium 2
    - 4GB RAM
  - 96 Nodes
    - Dual 2.4GHz Intel Xeon
    - 4GB RAM
  - 15TB General Parallel File System (GPFS)
  - 5TB PVFS

Version 1: Persistent server



Version 1: Persistent Server



- Custom for large BLAST computation
- Python-based askfor server
- NR manually distributed to cluster nodes
- First run at the problem, simple and straightforward implementation
- Server is single point-of-failure (death of server releases all cluster nodes)
  - However, server maintains persistent state and is restartable

V1, cont.

- Immediate feedback when a client dies
  - Connection goes away
  - Manager can reassign work
- Potential for resource overload
  - Requires file descriptor per cluster node in use

Road Tested

This model has been used regularly for the past 18 months to support several genome annotation projects.

We have consumed over 32,000 node hours on ANL's Jazz cluster alone using this model.

V2: Scopmap via Apache

- Custom for Scopmap protein structure computation
- Asynchronous client/server comms
  - Apache server
  - XMLRPC/CGI interface
  - Computation state maintained in relational db
- No file distribution required
  - Work units and results small enough to carry in XMLRPC messages
  - Required databases installed as part of Scopmap program installation

V2, cont.

- Async interface results in more reliability (no dependency on single server)
- However, lose information about status of clients
  - Scopmap computation can run very long (some pieces of work exhausted the 20-hour node allocation time)
  - Without heartbeat, cannot tell if job is dead or taking a long time

Performance of V2 Askfor Model

Attempted to classify all proteins in E.coli K12 with SCOPmap

Successful for over a 1000 proteins

3.5 days - aggregate of 196 nodes

VS

176 days for the serial computation

V3: General tool

- Builds on earlier tools:
  - Apache-based server
  - SOAP-based communications
  - Relational database for state maintenance
- Intended to be more general purpose
  - Pluggable application logic
  - Supports file staging

V3, cont.

- Addresses other weaknesses:
  - Client heartbeat provides mgr with complete state information
  - Careful signal handling and abnormal termination support
- Results in more server traffic, with some spectacular load spikes on the creation of 100-node worker pools
  - But computers are highly capable, so brute-force solution works well

  -Successfully used for large BLAST computations

Conclusion & Future

Our production and experimental usage experiences validate the AskFor model for use in high-volume distributed computations

Envision a web services interface to a centralized AskFor Manager that presents units of work to large-scale, distributed, back-end processing engines for applications in computational biology.

Computational Genomics

Complete Genomes

Semi-Automated Feature Identification And Annotation

Evolutionary Analysis

Organism Phylogeny

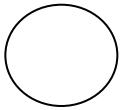
Reconstructing Gene Histories

Horizontal Gene Transfer Studies

Chromosomal Cluster Phylogeny

Comparative Analysis

Subsystems Analysis



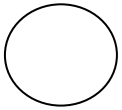
Gene Function Clarification

Encoding Pathway Variants

Subsystem Phylogenetics

Model Development

Reaction Set Determination



Compound Database

Reaction Database

Encoding Reaction Variants

S Matrix Generation

Core Metabolism

Spontaneous Reactions

Transport

Transcription Regulation Signaling

Flux Based Models

Constraints Observations

Systems Analysis

Phenotype Prediction

Dynamic modeling

Reaction Rates

Kinetics Based Models

Allosteric Regulation

Evolution of Kinetics

Personal Tools

Toolkits

Frameworks

Standalone Suites

Personal clusters

Examples:

BioLinux
EMBOSS
BioPerl

Community Access

FTP Data Access

Centralized Websites

Portals

Examples:

PubMed
EMBL web
BLAST severs
My*name it*

P2P Micropublishing

Mirroring

Publish/Subscribe

Blogs and Podcasting

Clearing Houses

Curation and Review

Examples:

The SEED
Alliance for Cell Signaling

Web Services

WS interfaces To Public Servers

WS to standalone Applications

Re-architected Tools

LW Clients

Workflow Tools

Open Grids

Service Oriented Architectures

Services Based Access to Resources

Tool and Content Providers

Service Oriented Programming tools

Acknowledgements

This work was supported by the Mathematical, Information and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research,Office of Science, U.S. Department of Energy, under Contract W-31-109-ENG-38