mpiBLAST

Open Source Parallel BLAST

mpiBLAST evolves
success, collaborations, and challenges

© Aaron Darling, BOSC 2005

THE UNIVERSITY WISCONSIN MADISON

Los Alamos NATIONAL LABORATORY

mpiBLAST
Open Source Parallel BLAST

Green Destiny

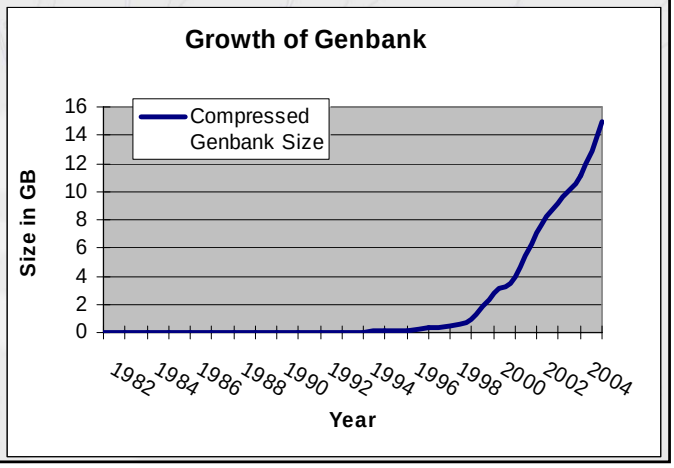
- 240 nodes of bladed-Beowulf bliss
- Each node has:
  - 933MHz Transmeta CPU
  - 640MB RAM
  - 20GB 4200RPM hard drive
  - 3 x 100Mbit ethernet
- One rack
- 5200 watts (low energy consumption)

BLAST on Green Destiny?

- The ASAP bacterial genome annotation system depends heavily on similarity searches
- Could Green Destiny be used for BLAST?
- Looked at existing parallel BLAST implementations
  - Query segmentation very common
  - nt database was 1.2GB (formatted)
  - Each blade has 640MB memory
  - BLAST is fastest when DB fits in memory

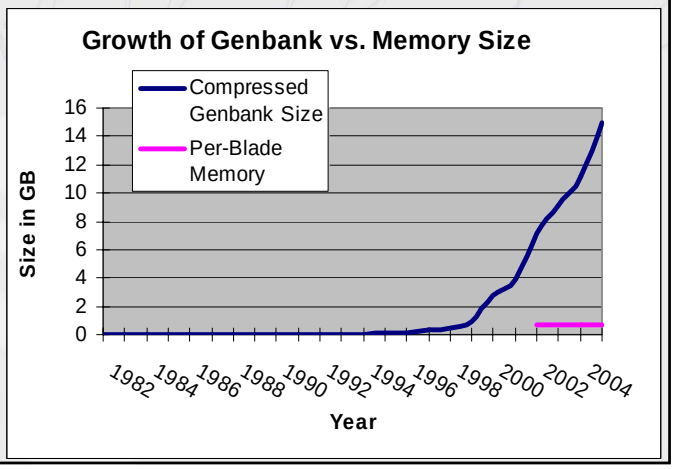
The growth of GenBank

- GenBank has been growing exponentially
- nt has tripled in size over three years



As the database grows…

- Memory per-blade doesn't grow



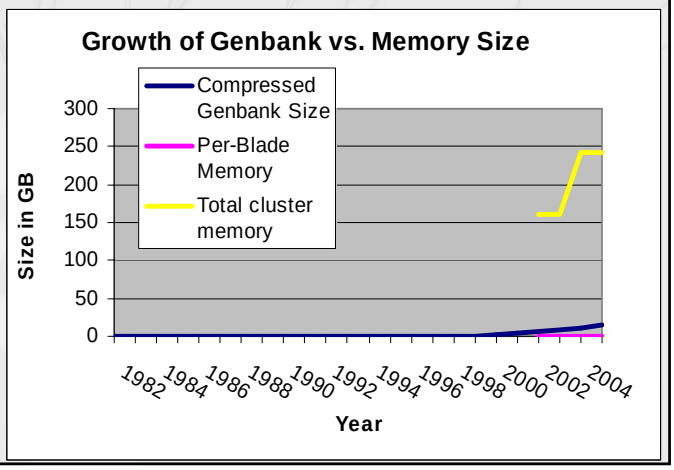
Look at aggregate cluster memory

- Big enough in 2002 and still big enough

Originally 156GB of cluster memory.

Added another 120 blades in 2004. Total 360 blades, 230GB RAM.

Must exploit total cluster memory to BLAST effectively!



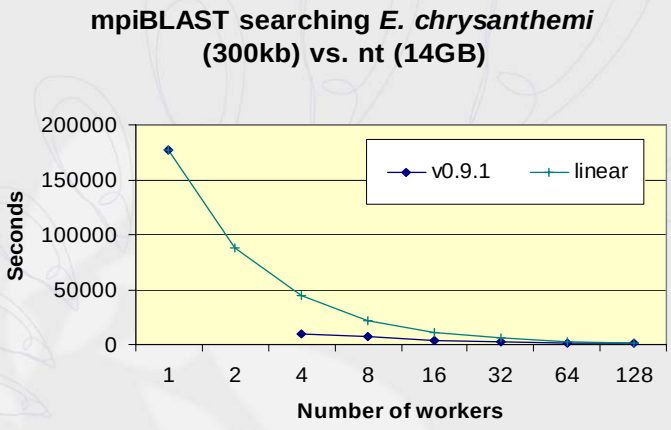
The birth of mpiBLAST

- First release version 0.9 11/2002
- Wrappers for blastall and formatdb that split the database and search each fragment on a cluster node

mpiBLAST is a mean little devil

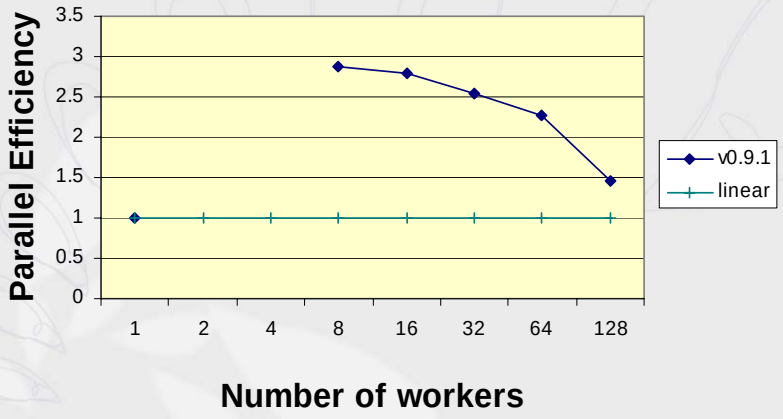
Results from each node were munged together using a text-parser from David Mathog

mpiBLAST 0.9 search times



Searching 441 queries (300KB) vs nt (14GB uncompressed)
Timed the second of two runs after the DB distribution
mpiBLAST v0.9 scales well: **187x speedup on 128 workers**

mpiBLAST 0.9 parallel efficiency



Searching 441 queries (300KB) vs nt (14GB uncompressed)

mpiBLAST v0.9 parallel efficiency is best with the fewest workers that can fit the entire database into core memory

Drawbacks of mpiBLAST 0.9

- Poor accuracy
  - E-values were approximate for blastn/p and way off for translated search types
- Poor fault-tolerance
  - 0.9 didn't adapt to changes in cluster resources. If a node went down the db had to be reformatted and re-distributed
- Poor usability
  - NCBI BLAST supports 12 different output formats, but Mathog's result-munger only did one text and one HTML format

Moving on…mpiBLAST 1.0

- Poor accuracy
  - E-values were approximate for blastn/p and way off for translated search types
- Better fault-tolerance
  - Master/Worker: dynamically distribute the database at job startup (if not already distributed to all worker nodes)
- Better usability
  - Nodes generate intermediate results in ASN.1 format. A master process uses the NCBI library to read the ASN.1 results and translates them to the requested output format.

mpiBLAST 1.0 parallel efficiency

Searching 441 queries (300KB) vs nt (14GB uncompressed)
mpiBLAST v1.0: add features, lose performance
Got us a "Best Paper" at the 4th Intl. Conference on Linux Clusters

mpiBLAST 1.2

- Jason Gans implements "distributed biosequence lookup"
  - When master needs a sequence for an alignment it pulls the sequence (ASN.1 encoded) from the worker instead of from shared storage
- Other contributions start rolling in
  - Bugfixes for various platforms (BSD/Solaris/IRIX)
  - RPM packaging by Joe Landman
  - BioBrew Linux cluster distribution by Glen Otero
  - Query segmentation and SGE submission scripts by Joe Landman
  - Userbase grows to include MGH, Eli Lilly, many more

mpiBLAST 1.2.1 parallel efficiency

Searching 441 queries (300KB) vs nt (14GB uncompressed)
Extra search options were: --disable-mpi-db
mpiBLAST v1.2.1 parallel efficiency is still worse than 0.9!

Problems with mpiBLAST 1.2

- Poor accuracy (still!)
  - E-values were approximate for blastn/p and way off for other search types
- "distributed biosequence lookup"
  - Some sequences are very large and can't all be stored in memory until the search has finished because the master will run out of memory
- Joe Landman's SGE submission script works around the out-of-memory problem by splitting up the query data

mpiBLAST 1.3

- Query segmentation & pipelined result output
  - Master writes out results for the first query while the second query is being searched. Overlaps result output with search, reduces memory load on the writer
- Exact e-value statistics
  - Master calculates effective query and database lengths using the entire database at job startup, broadcasts these values to workers
- Pipelined database fragment copy
  - When N > about 10 workers all try to copy from NFS at the same time…chaos ensues. Pipelining limits the number of concurrent copy operations.

mpiBLAST 1.3.1 parallel efficiency

Searching 441 queries (300KB) vs nt (14GB uncompressed)
Extra search options were: --disable-mpi-db --concurrent=4
mpiBLAST v1.3.1 parallel efficiency is still worse than 0.9!

mpiBLAST-g2, a grid-based mpiBLAST

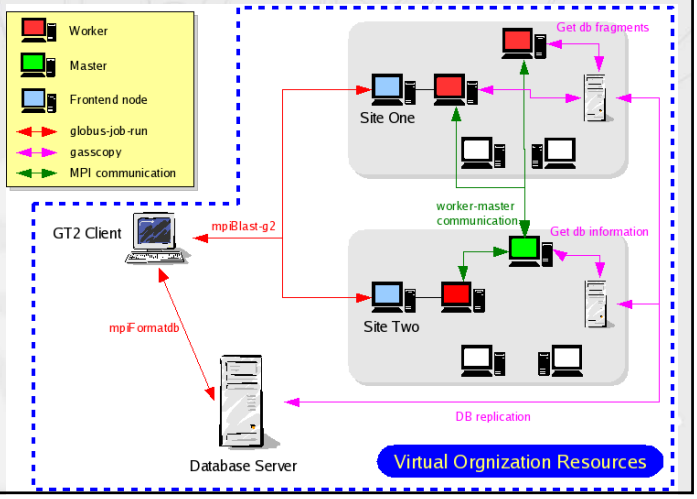
Using MPICH-g2 and the GT2 GASSCOPY API

Features:

-Multi-cluster job execution

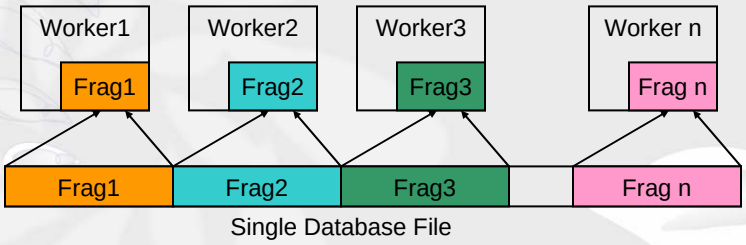
-Remote database sharing

-Implemented by ASCC (Taiwan)



pioBLAST – By NCSU/ORNL

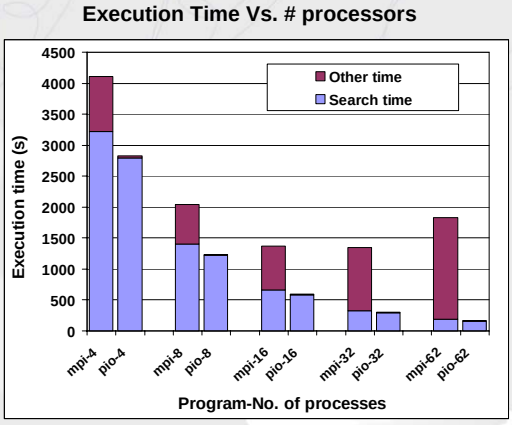
- Built out of mpiBLAST 1.2.1
  - Uses parallel/collective I/O (MPI File I/O + a parallel filesystem)
- No prepartitioning of the database
  - One single database image to search against
- Virtual fragments generated dynamically at run time
  - Workers read inputs in parallel with MPI-IO interface
- Fragment size configurable at run time
  - Easily supports dynamic load balancing



H. Lin et. al. "Efficient Data Access for Parallel BLAST" In proceedings of IPDPS 2005

Performance Results

- Platform: SGI Altix at ORNL
  - 256 processors 1.5GHz Itanium2 processors
  - 8GB memory per processor
  - XFS
- Database: nr (1GB)
- Node scalability
  - mpiBLAST 1.2.1: **non-search** time increase from **21%** (4 node) to **90%** (62 node)
  - pioBLAST: **non-search** time remains low: **1%** (4 nodes) to **8%** (62 nodes)



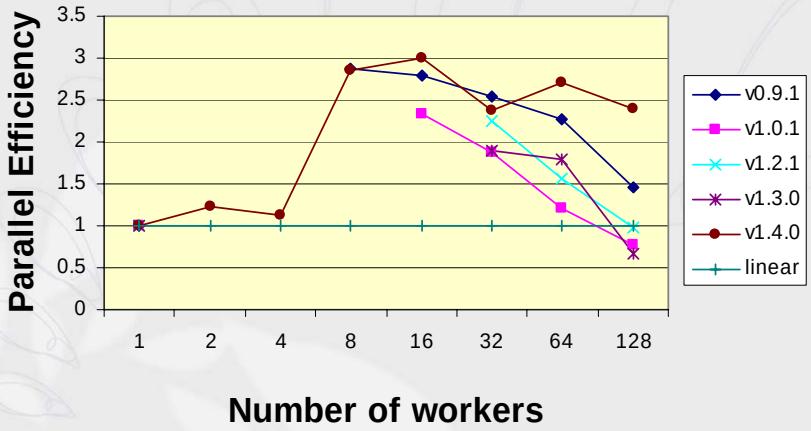
mpiBLAST needs to be fixed

- Motivation: lots of angry users
  - Lik Mui at Stanford writes to bioclusters:
  - "…using 1 single head node, I get a blastall output in about 20 seconds. When I feed an input of 20 input sequences to MPIBLAST on a 24 node cluster, the result takes 3 minutes to get back. This is hardly super-linear"
  - Other unsatisfied users from PNNL, INL, and LLNL
- **My problem: They are right**. I botched the fragment copy scheduler in 1.3.0 and the distributed sequence lookup (a default option in 1.2.1 and 1.3.0) wasn't ready for prime-time

mpiBLAST 1.4.0 – The retribution release

- Fixed distributed sequence lookup
  - Send *partial* biosequences
    - Only the part of the biosequence used in the alignment goes over the wire
    - Huge impact on human chromosome database searches
- Fixed 1.3.0 fragment copy algorithm
  - Predistribute N complete copies of the database across the cluster. N defaults to 1.
- Fixed worker->writer communication
  - 1.3.0 writer receive buffers would fill up and bog down the writer
  - 1.4.0 reduces the number of worker->writer send operations from roughly 2 x DB hits + 4 x queries down to 2 x queries

mpiBLAST 1.4.0 parallel efficiency



Number of workers

Searching 441 queries (300KB) vs nt (14GB uncompressed)
mpiBLAST v1.4.0 beats 0.9! **305x speedup on 128 workers!**
Search time drops from 2 days to under 10 minutes.

Acknowledgements

Presented by:

Aaron Darling <darling@cs.wisc.edu>

Primary code contributors:

Lucas Carey (SUNY SB), Jason Gans (LANL), Jeremy Archuleta (LANL)

Other contributors:

Wu-chun Feng (LANL), Joe Landman (Scalable Informatics)
Glen Otero, Heshan Lin (ORNL), Avery Ching (Northwestern)
Mike Firpo (LLNL), Adam Moody (LLNL),
Bruno Nyffeler (SIB), Mike Cariaso (SAIC/Celera)

This work was funded in part by NLM Grant 5T15LM007359-04