High-performance Computing Methods for Computational Genomics

Srinivas Aluru, Iowa State University
David A. Bader, Georgia Institute of Technology
Ananth Kalyanaraman, Washington State University

IPDPS’07 Tutorial Handouts
7:00pm-10:00pm, 3/27/2007

1-4244-0910-1/07/$20.00 ©2007 IEEE

Srinivas Aluru

IOWA STATE UNIVERSITY

- Stanley Chair in Interdisciplinary Engineering,
  Professor, Department of Electrical and Computer Engineering,
  Chair, Bioinformatics and Computational Biology,
  Iowa State University, Ames, IA

- Contact:
  - Email: aluru@iastate.edu
  - Website: http://www.ee.iastate.edu/~aluru

- Ph.D., 1994, Iowa State University

- Research Interests:
  - Parallel Algorithms and Applications
  - Computational Biology and Bioinformatics
  - Combinatorial Scientific Computing

David Bader

- Associate Professor, College of Computing
  - Computational Science and Engineering
  - Executive Director of High-Performance Computing
  - Georgia Institute of Technology, Atlanta, GA
- Contact:
  - Email: bader@cc.gatech.edu
  - Website: http://www.cc.gatech.edu/~bader
- Ph.D., 1996, University of Maryland, College Park
- Research Interests:
  - Parallel algorithms and applications
  - High-performance computing for computational biology and genomics
  - Large-scale phylogeny reconstruction
- HPC Thrust Leader of CIPRES

Ananth Kalyanaraman

- Assistant Professor,
  School of Electrical Engineering and Computer Science,
  Washington State University,
  Pullman, WA

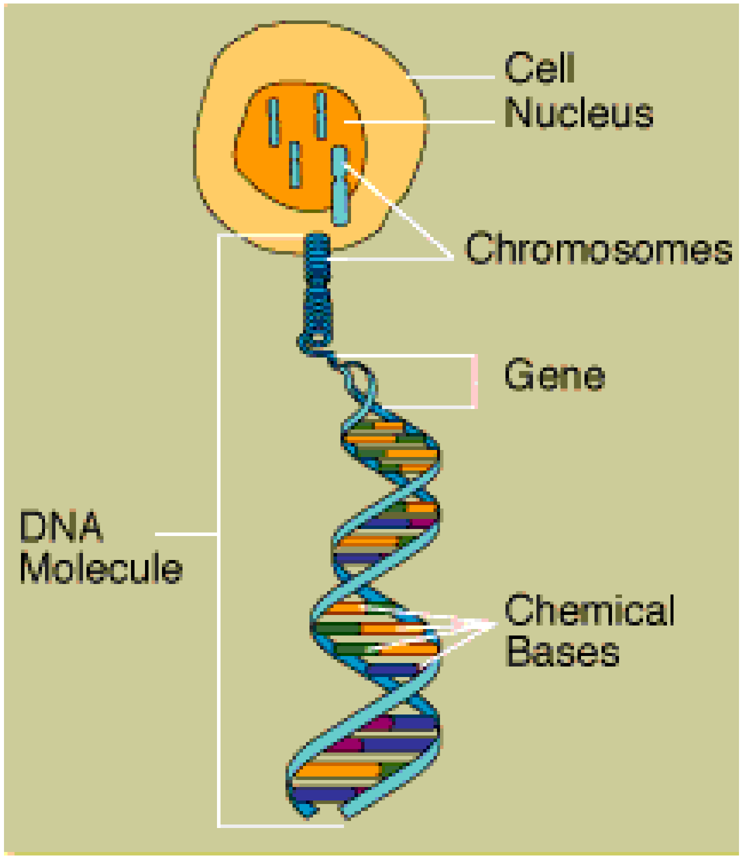
- Contact:
  - Email: ananth@eecs.wsu.edu
  - Website: http://www.eecs.wsu.edu/~ananth

- Ph.D., 2006, Iowa State University

- Research Interests:
  - Computational Biology and Bioinformatics
  - Parallel Algorithms and Applications
  - String Algorithms and Combinatorial Pattern Matching

Background

- *DNA*
  - Computationally, a string over alphabet *{A,C,G,T}*
- *Genome*
  - Collection of all DNA in a cell
- *Gene*
  - Encodes the recipe for producing proteins
- *Protein*
  - A sequence of amino acids



Source: *http://rex.nci.nih.gov/behindthenews/ugt/05ugt/ugt05.htm*

Sequence Discovery

Genome
Gene
Regulatory elements
Proteins

Function

Gene to protein annotation
Gene expression analysis
Microarray experiments
RNA interference
Metabolic networks/pathway

Structure

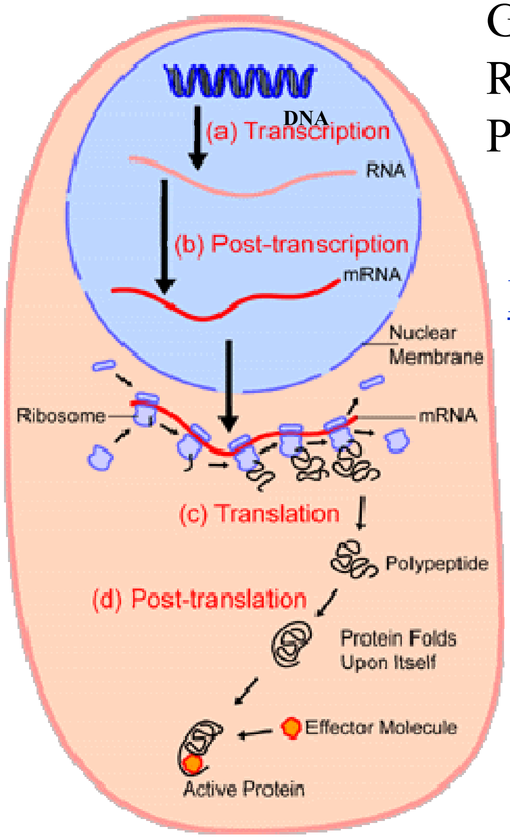
Gene structure prediction
RNA structure prediction
Protein structure prediction

Evolutionary Studies

Tree of life
Speciation

Population Genetics

Haplotype analysis
Nucleotide polymorphism



Protein Synthesis in an Eukaryotic Cell
Source: Science Primer, NCBI, NIH.
http://en.wikipedia.org/wiki/Image:Proteinsynthesis.png

GenBank

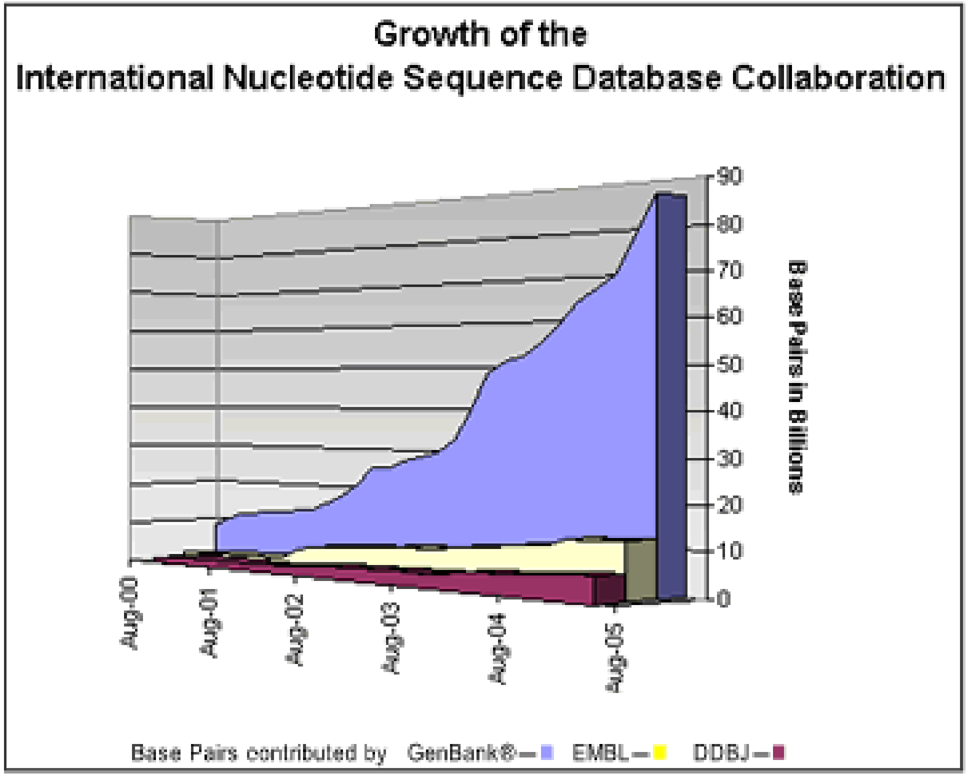
"*An annotated collection of all publicly available nucleotide and amino acid sequences*."

As of October 2005, the NCBI's public collection contained:

- **109.8 G bases**, and
- **60.3 million** sequences,

obtained from over

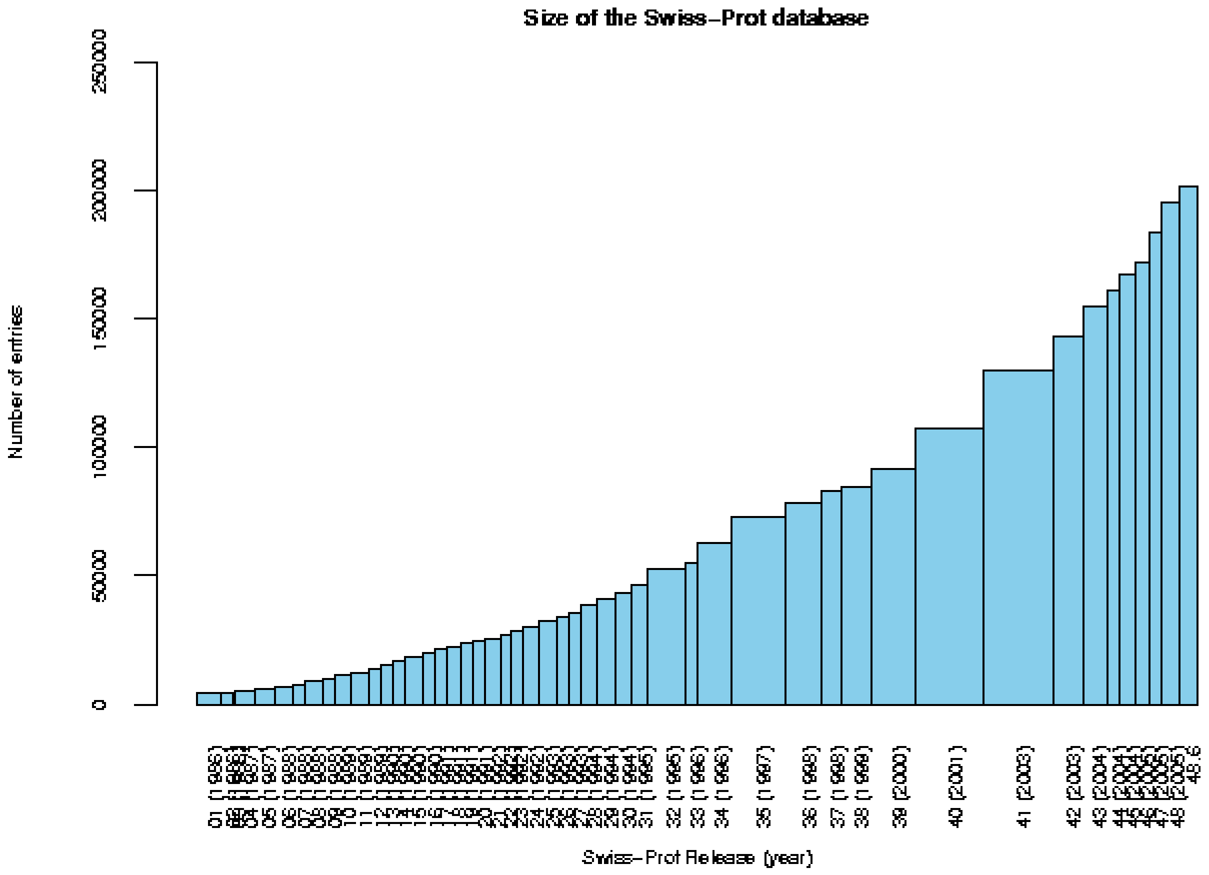
- **165,000 organisms**



Source: NCBI GenBank http://www.ncbi.nih.gov/GenBank/index.htm

UniprotKB/Swiss-Prot

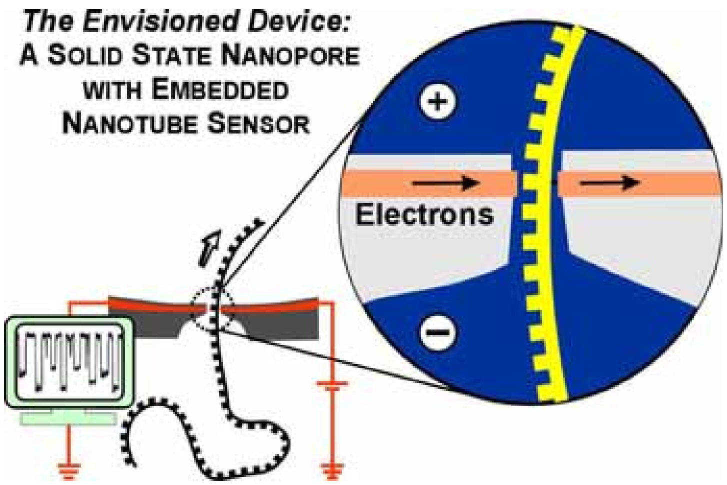
- A knowledge base for protein sequences.
- Contains annotated protein sequences
- Contains 201,594 sequences, 73,123,101 amino acids.



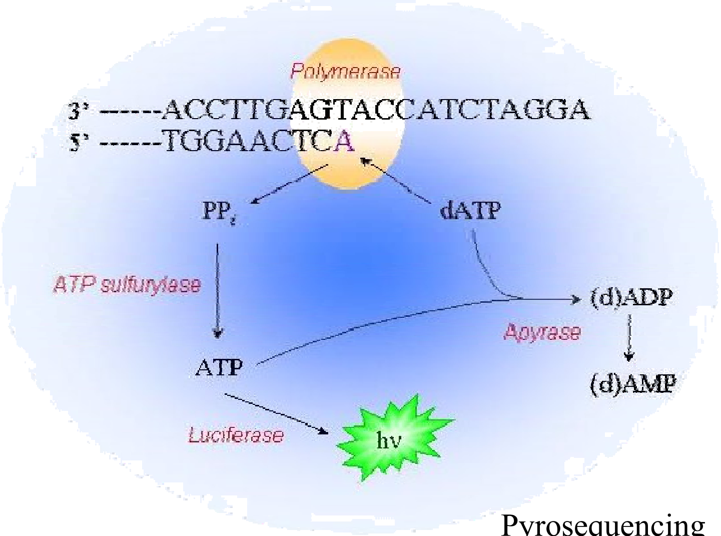
Source: *http://ca.expasy.org/sprot/relnotes/relstat.htm*

Primary HPC Uses in Biology

- Massive Parallelism
  - Sequencing: Pyrosequencing, Nanopore, Polony
  - Data assembly and mining
  - Databases of genomes and derived information
  - Sequence comparisons (Smith-Waterman, BLAST)



http://www.mcb.harvard.edu/branton/index.htm



Pyrosequencing

Primary HPC Uses in Biology

- NP-hard Problems
  - Intractable (computationally expensive)
    - Exact solutions for small inputs
    - Approximate solutions for moderate to large inputs
  - Structure prediction and functional analysis
    - protein folding
  - Reconstructing evolutionary histories
    - Phylogenetic Relationships
    - Comparative Genomics

Topics for this Tutorial

- Review high-performance methods in computational genomics that belong one of the following classes

Part I

1. Compare one sequence vs. another sequence
   - Application: Sequence alignment
2. Compare one sequence against many sequences
   - Application: Querying a database

Part II

3. Analyze multiple sequences
   - Applications: Clustering, Genome Assembly

Part III

4. Reconstruction of Evolutionary Histories

Tutorial Schedule

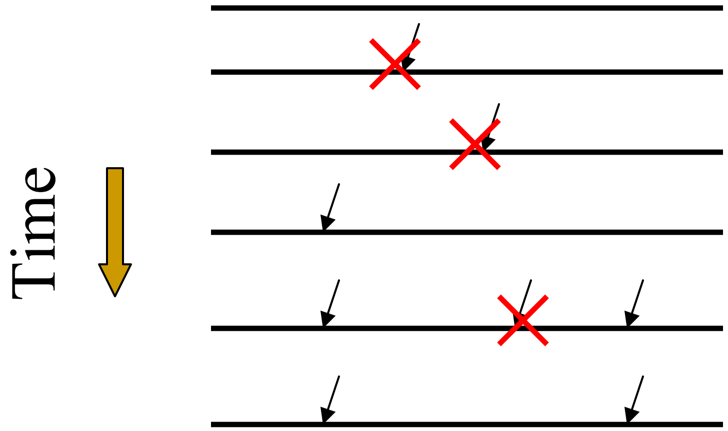
- Tuesday 7pm-10pm
  - 7:00pm – 7:10pm: Welcome and Introduction
  - 7:10pm – 7:40pm: Part I
  - 7:40pm – 8:30pm: Part II
  - 8:30pm – 8:45pm: Break
  - 8:45pm – 9:00pm: Part II
  - 9:00pm – 9:55pm: Part III
  - 9:55pm – 10:00pm: Conclusion

Part I:
Sequence Alignment and Database Querying

Why Compare One Sequence to Another?

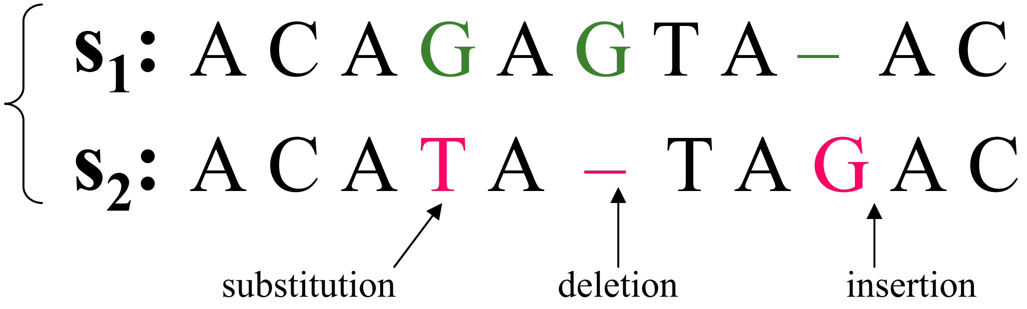
- *Mutation* ➔ natural genetic variations

A genome mutating over generations



- Mutations are random events
- The effect of only some mutation events carry over to future generations
- Sequence comparison key for evolutionary studies

Alignment between s_1 and s_2


s_1: A C A G A G T A – A C

s_2: A C A T A – T A G A C

How to Compare Two Sequences?

- **Problem**:
  - Given two sequences s_1 and s_2 over a fixed alphabet Σ, what is the set of variations that best describes the genetic transformation from s_1 to s_2 (or equivalently, from s_2 to s_1)?

Combinatorial Optimality

• Based on either maximizing an *alignment score* or minimizing *edit distance*

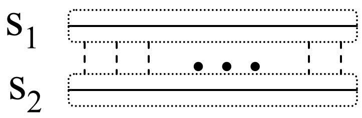
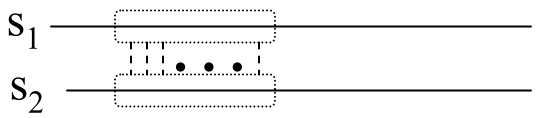
• Standard dynamic programming techniques

Probabilistic Optimality

• Based on finding a most *probable* set of changes in aligning two sequences

• Hidden-Markov Model (HMM) techniques

Two Important Types of Alignments

		Preferred Applications
Global	Alignment between s_1 and s_2  	For detecting two highly similar sequences (eg., two homologous proteins)
Local	Alignment between a substring of s_1 and a substring of s_2  	For detecting highly conserved regions (eg., genes) between two sequences (eg., genomes)

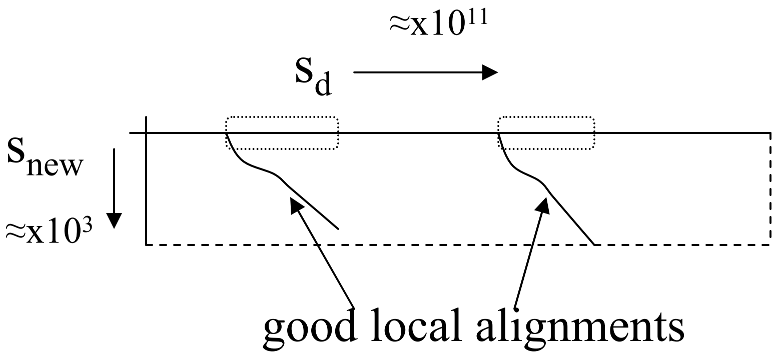
Optimal global and local alignments can be computed in $O(|s_1|.|s_2|)$ run-time and $O(|s_1|+|s_2|)$ space

Need for a Fast Alignment Method

- Let us say, we have a newly found gene candidate, s_{new}, in an arbitrary organism. Next, we want to locate "similar" genes in other organisms.

One Approach:

1. Concatenate all sequences in our genomic database into one sequence, say s_d
2. Compute the local alignment between s_{new} and s_d
3. Report all "significant" local alignments



Run-time: $O(|s_d|.|s_{new}|)$

⇓

Very long query time !!

Basic Local Alignment Search Tool (BLAST)

- Altschul *et al.* (1990) developed a program called BLAST to quickly query large sequence databases

- **Input:**
  - Query sequence q and a sequence database D

- **Output:**
  - List of all significant local alignment hits ranked in increasing order of *E-value* (aka *p-value*, which is the probability that a random sequence scores more than q against D).

BLAST Algorithm

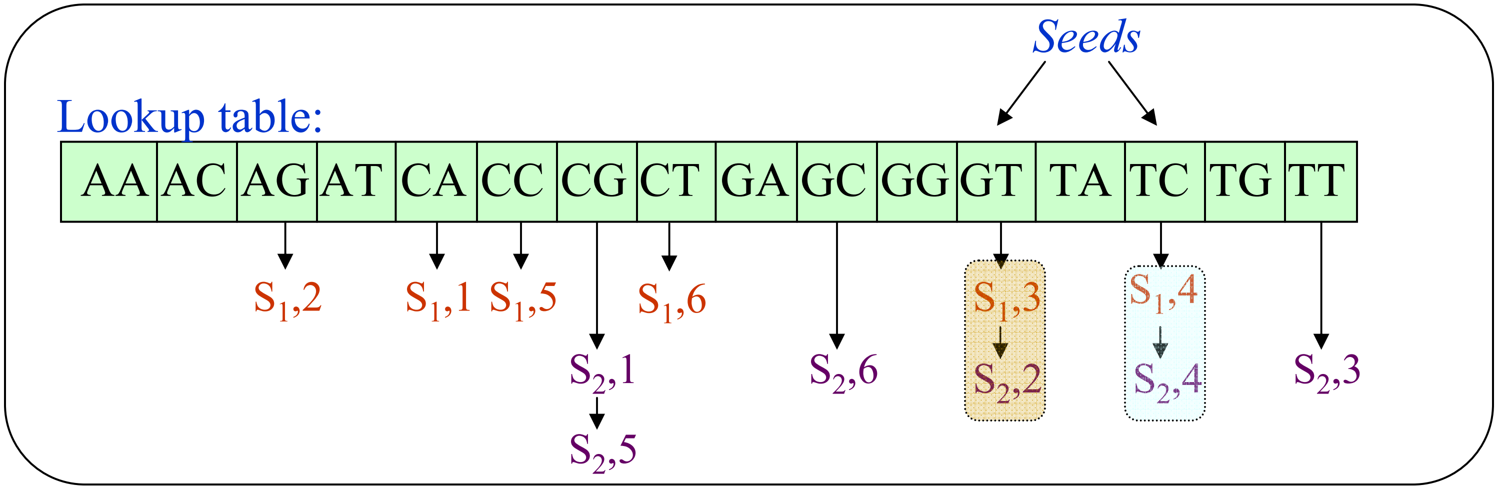
0. Preprocess: Build a *lookup table* of size $|\Sigma|^w$ for all *w*-length words in D

	1	2	3	4	5	6	7
S_1:	C	A	G	T	C	C	T
S_2:	C	G	T	T	C	G	C

$\Sigma=\{A,C,G,T\}$

$w = 2$

→ 4^2 (=16) entries in lookup table



AA	AC	AG	AT	CA	CC	CG	CT	GA	GC	GG	GT	TA	TC	TG	TT
		S_1,2		S_1,1	S_1,5	S_2,1 S_2,5	S_1,6		S_2,6		S_1,3 S_2,2		S_1,4 S_2,4		S_2,3

Preprocessing is a one time activity

BLAST Algorithm …

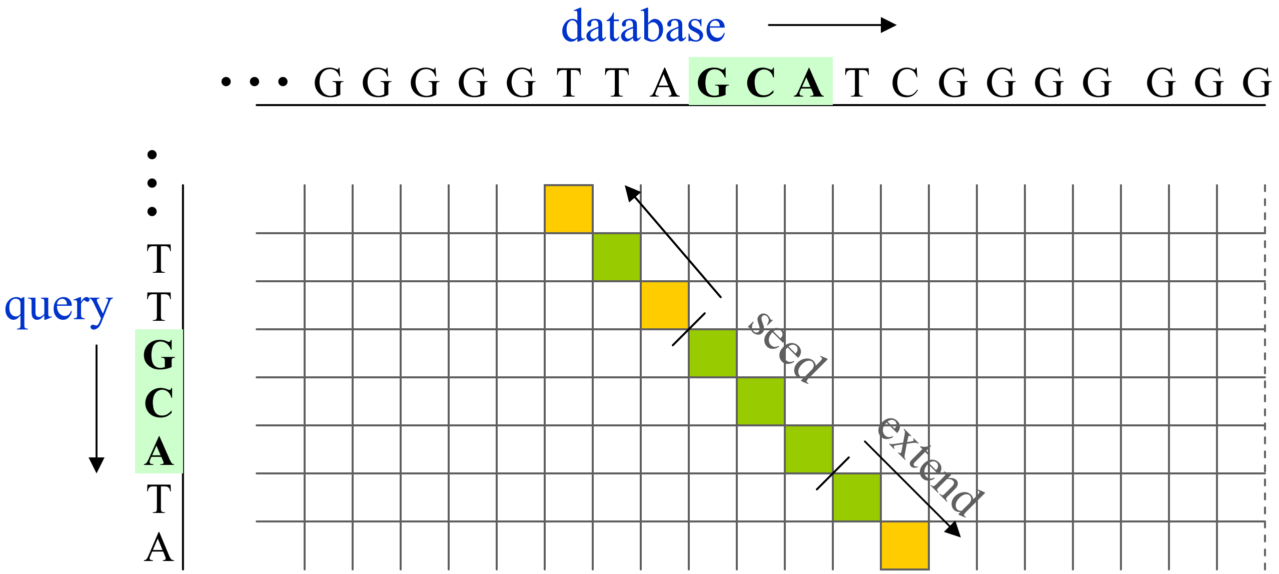
1. Identify Seeds*:* Find all *w*-length substrings in *q* that are also in D using the lookup table

2. Extend seeds: Extend each seed on either side until the aggregate alignment score falls below a threshold
   - Ungapped: Extend by only either matches or mismatches
   - Gapped: Extend by matches, mismatches or a limited number of insertion/deletion gaps

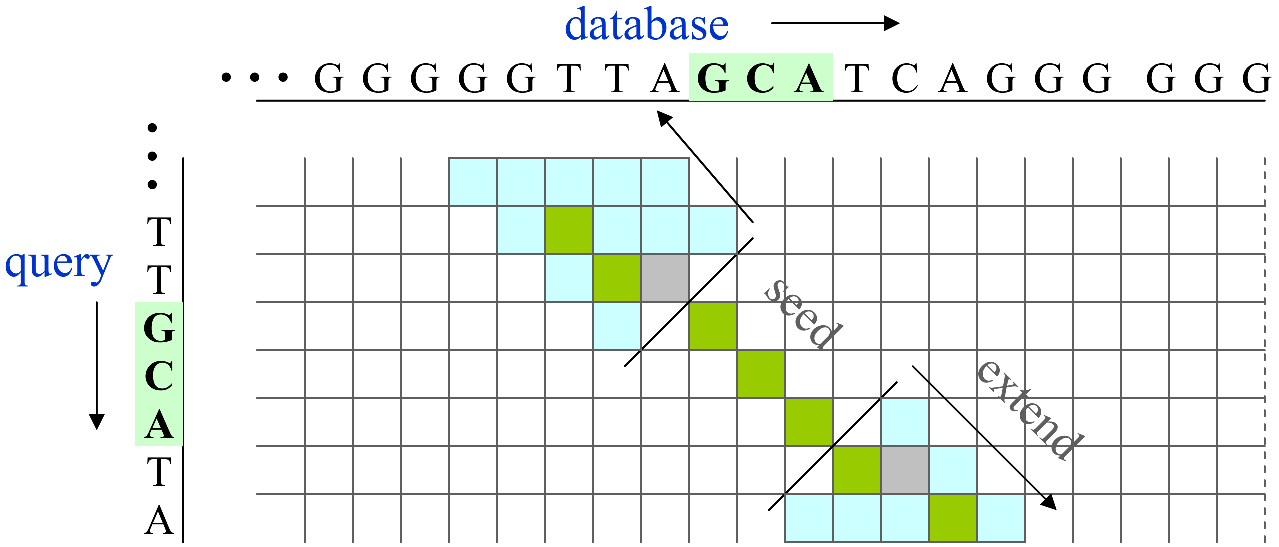
3. Record all local alignments that score more than a certain statistical threshold

4. Rank and report all local alignments in non-decreasing order of *E-value*

Illustration of BLAST Algorithm



Ungapped Extension



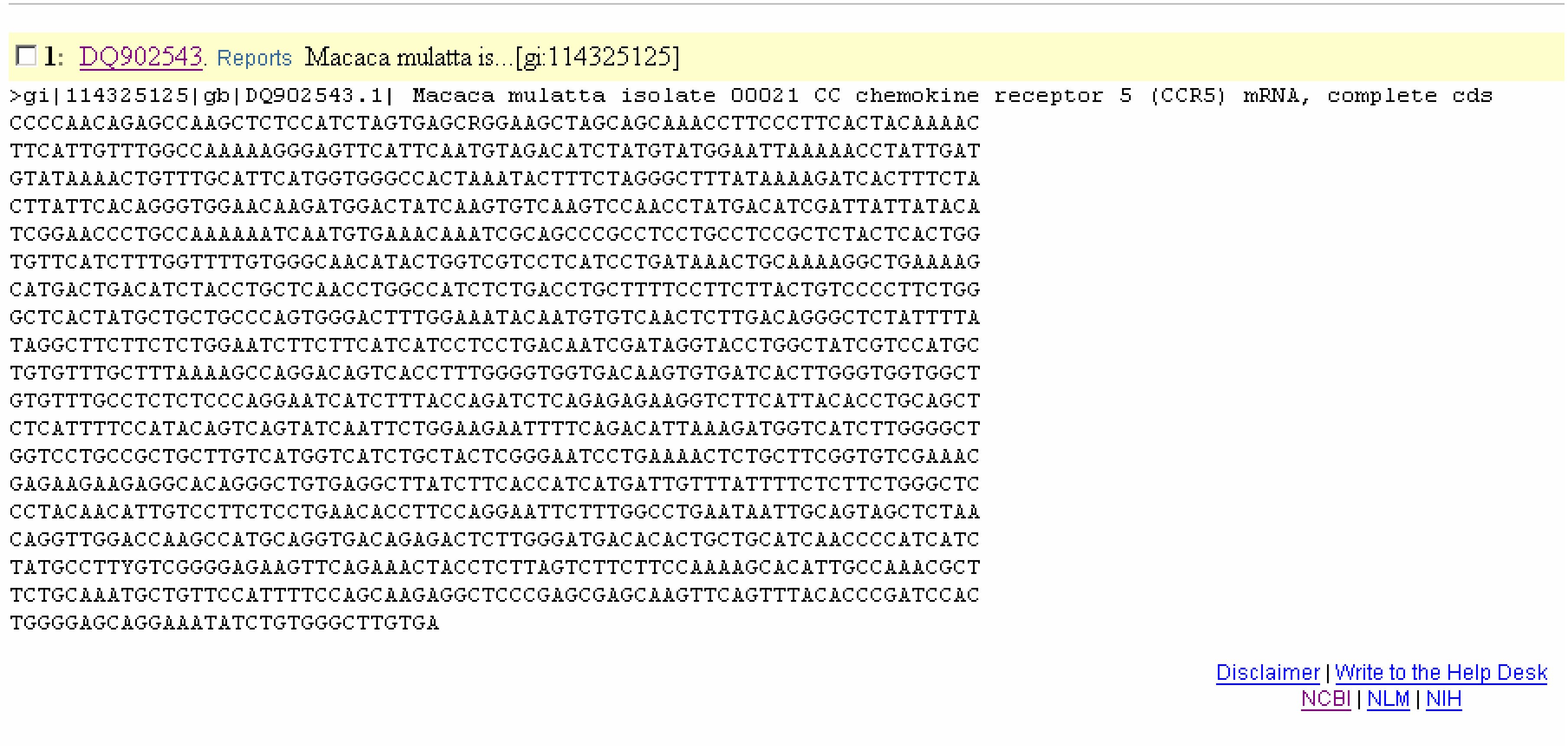
Gapped Extension (over a band of diagonals)

Different Types of BLAST Programs

Program	**Query**	**Database**
blastn	nucleotide	nucleotide
blastp	protein/peptide	protein/peptide
blastx	nucleotide	protein/peptide
tblastn	protein/peptide	nucleotide
tblastx	nucleotide	nucleotide

http://www.ncbi.nlm.nih.gov/blast

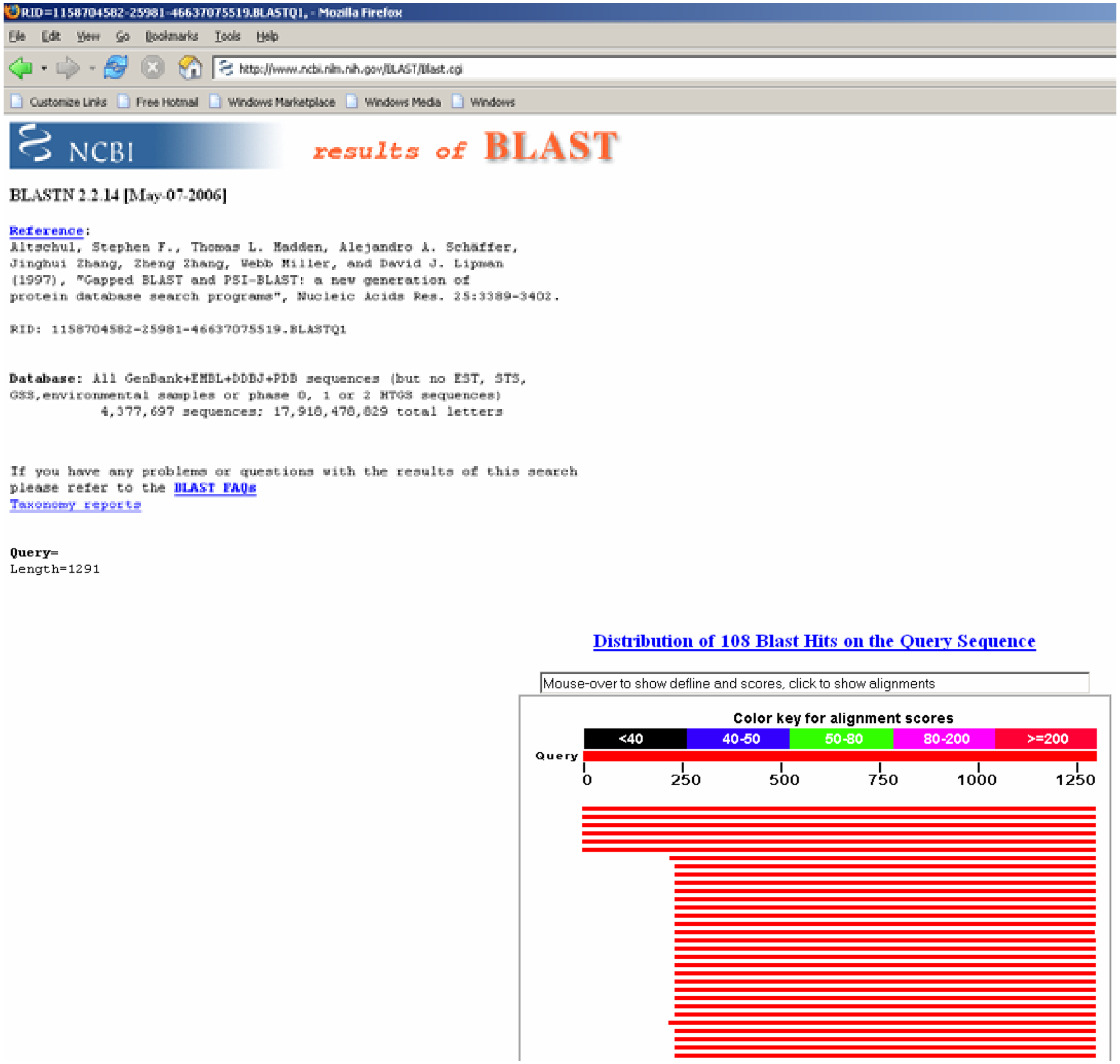
An Example: Querying gene CCR5 against GenBank



☐ **1:** DQ902543. Reports Macaca mulatta is...[gi:114325125]

```
>gi|114325125|gb|DQ902543.1| Macaca mulatta isolate 00021 CC chemokine receptor 5 (CCR5) mRNA, complete cds
CCCCAACAGAGCCAAGCTCTCCATCTAGTGAGCRGGAAGCTAGCAGCAAACCTTCCCTTCACTACAAAAC
TTCATTGTTTGGCCAAAAAGGGAGTTCATTCAATGTAGACATCTATGTATGGAATTAAAAACCTATTGAT
GTATAAAACTGTTTGCATTCATGGTGGGCCACTAAATACTTTCTAGGGCTTTATAAAAGATCACTTTCTA
CTTATTCACAGGGTGGAACAAGATGGACTATCAAGTGTCAAGTCCAACCTATGACATCGATTATTATACA
TCGGAACCCTGCCAAAAAATCAATGTGAAACAAATCGCAGCCCGCCTCCTGCCTCCGCTCTACTCACTGG
TGTTCATCTTTGGTTTTGTGGGCAACATACTGGTCGTCCTCATCCTGATAAACTGCAAAAGGCTGAAAAG
CATGACTGACATCTACCTGCTCAACCTGGCCATCTCTGACCTGCTTTTCCTTCTTACTGTCCCCTTCTGG
GCTCACTATGCTGCTGCCCAGTGGGACTTTGGAAATACAATGTGTCAACTCTTGACAGGGCTCTATTTTA
TAGGCTTCTTCTCTGGAATCTTCTTCATCATCCTCCTGACAATCGATAGGTACCTGGCTATCGTCCATGC
TGTGTTTGCTTTAAAAGCCAGGACAGTCACCTTTGGGGTGGTGACAAGTGTGATCACTTGGGTGGTGGCT
GTGTTTGCCTCTCTCCCAGGAATCATCTTTACCAGATCTCAGAGAGAAGGTCTTCATTACACCTGCAGCT
CTCATTTTCCATACAGTCAGTATCAATTCTGGAAGAATTTTCAGACATTAAAGATGGTCATCTTGGGGCT
GGTCCTGCCGCTGCTTGTCATGGTCATCTGCTACTCGGGAATCCTGAAAACTCTGCTTCGGTGTCGAAAC
GAGAAGAAGAGGCACAGGGCTGTGAGGCTTATCTTCACCATCATGATTGTTTATTTTCTCTTCTGGGCTC
CCTACAACATTGTCCTTCTCCTGAACACCTTCCAGGAATTCTTTGGCCTGAATAATTGCAGTAGCTCTAA
CAGGTTGGACCAAGCCATGCAGGTGACAGAGACTCTTGGGATGACACACTGCTGCATCAACCCCATCATC
TATGCCTTYGTCGGGGAGAAGTTCAGAAACTACCTCTTAGTCTTCTTCCAAAAGCACATTGCCAAACGCT
TCTGCAAATGCTGTTCCATTTTCCAGCAAGAGGCTCCCGAGCGAGCAAGTTCAGTTTACACCCGATCCAC
TGGGGAGCAGGAAATATCTGTGGGCTTGTGA
```

An Example: Querying Result (Page I)


RID=1158704582-25981-46637075519.BLASTQ1, - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://www.ncbi.nlm.nih.gov/BLAST/Blast.cgi
Customize Links
Free Hotmail
Windows Marketplace
Windows Media
Windows
NCBI
results of BLAST

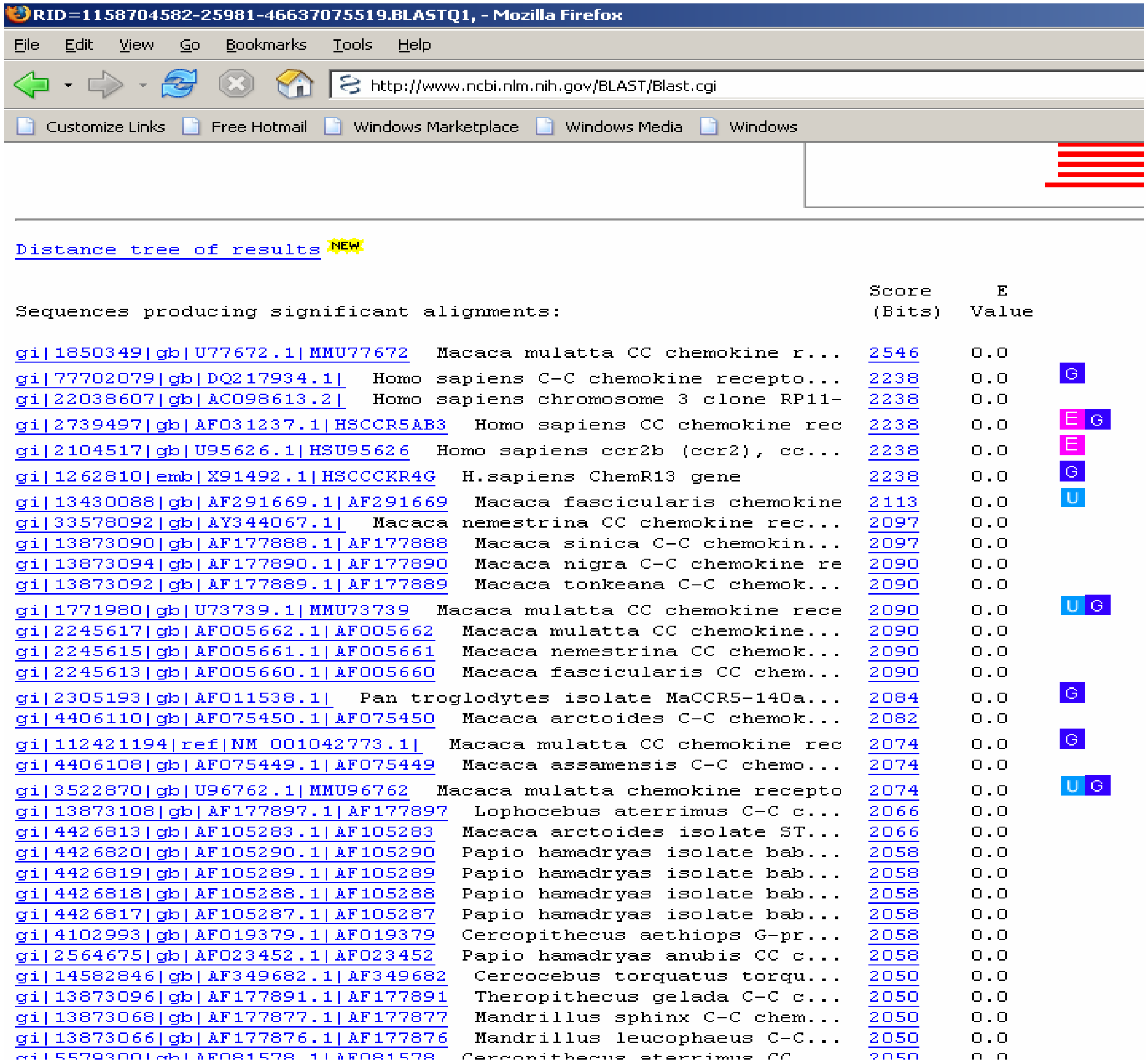
BLASTN 2.2.14 [May-07-2006]

```
Reference:
Altschul, Stephen F., Thomas L. Madden, Alejandro A. Schäffer,
Jinghui Zhang, Zheng Zhang, Webb Miller, and David J. Lipman
(1997), "Gapped BLAST and PSI-BLAST: a new generation of
protein database search programs", Nucleic Acids Res. 25:3389-3402.

RID: 1158704582-25981-46637075519.BLASTQ1


Database: All GenBank+EMBL+DDBJ+PDB sequences (but no EST, STS,
GSS,environmental samples or phase 0, 1 or 2 HTGS sequences)
           4,377,697 sequences: 17,918,470,829 total letters


If you have any problems or questions with the results of this search
please refer to the BLAST FAQs
Taxonomy reports


Query=
Length=1291
```

Distribution of 108 Blast Hits on the Query Sequence

Mouse-over to show defline and scores, click to show alignments

An Example: Querying Result (Page II)



RID=1158704582-25981-46637075519.BLASTQ1, - Mozilla Firefox

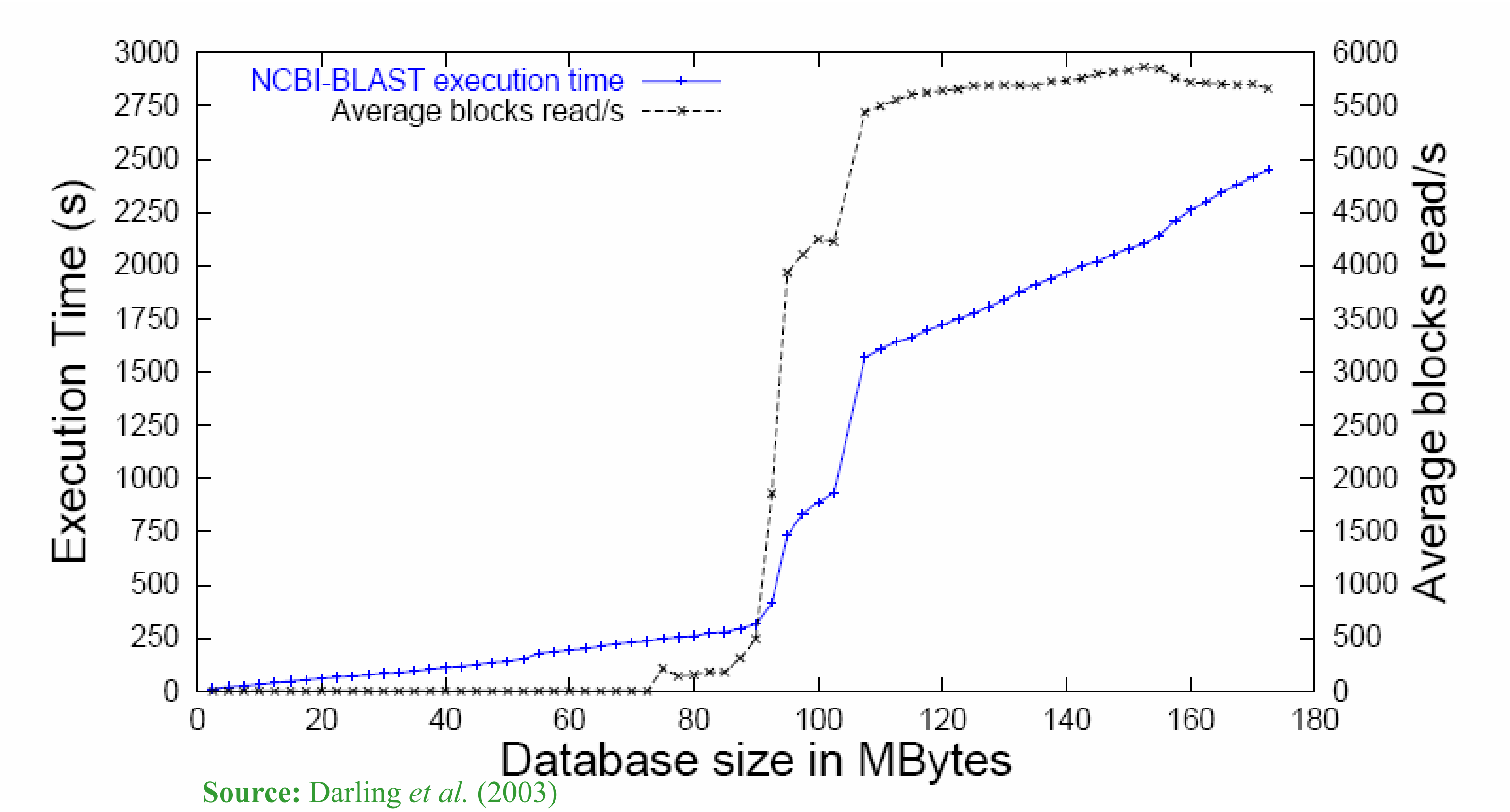
http://www.ncbi.nlm.nih.gov/BLAST/Blast.cgi

Distance tree of results NEW

```
                                                                   Score     E
Sequences producing significant alignments:                        (Bits)  Value

gi|1850349|gb|U77672.1|MMU77672  Macaca mulatta CC chemokine r...   2546    0.0
gi|77702079|gb|DQ217934.1|  Homo sapiens C-C chemokine recepto...   2238    0.0       G
gi|22038607|gb|AC098613.2|  Homo sapiens chromosome 3 clone RP11-   2238    0.0
gi|2739497|gb|AF031237.1|HSCCR5AB3  Homo sapiens CC chemokine rec   2238    0.0       E G
gi|2104517|gb|U95626.1|HSU95626  Homo sapiens ccr2b (ccr2), cc...   2238    0.0       E
gi|1262810|emb|X91492.1|HSCCCKR4G  H.sapiens ChemR13 gene           2238    0.0       G
gi|13430088|gb|AF291669.1|AF291669  Macaca fascicularis chemokine   2113    0.0       U
gi|33578092|gb|AY344067.1|  Macaca nemestrina CC chemokine rec...   2097    0.0
gi|13873090|gb|AF177888.1|AF177888  Macaca sinica C-C chemokin...   2097    0.0
gi|13873094|gb|AF177890.1|AF177890  Macaca nigra C-C chemokine re   2090    0.0
gi|13873092|gb|AF177889.1|AF177889  Macaca tonkeana C-C chemok...   2090    0.0
gi|1771980|gb|U73739.1|MMU73739  Macaca mulatta CC chemokine rece   2090    0.0       U G
gi|2245617|gb|AF005662.1|AF005662  Macaca mulatta CC chemokine...   2090    0.0
gi|2245615|gb|AF005661.1|AF005661  Macaca nemestrina CC chemok...   2090    0.0
gi|2245613|gb|AF005660.1|AF005660  Macaca fascicularis CC chem...   2090    0.0
gi|2305193|gb|AF011538.1|  Pan troglodytes isolate MaCCR5-140a...   2084    0.0       G
gi|4406110|gb|AF075450.1|AF075450  Macaca arctoides C-C chemok...   2082    0.0
gi|112421194|ref|NM_001042773.1|  Macaca mulatta CC chemokine rec   2074    0.0       G
gi|4406108|gb|AF075449.1|AF075449  Macaca assamensis C-C chemo...   2074    0.0
gi|3522870|gb|U96762.1|MMU96762  Macaca mulatta chemokine recepto   2074    0.0       U G
gi|13873108|gb|AF177897.1|AF177897  Lophocebus aterrimus C-C c...   2066    0.0
gi|4426813|gb|AF105283.1|AF105283  Macaca arctoides isolate ST...   2066    0.0
gi|4426820|gb|AF105290.1|AF105290  Papio hamadryas isolate bab...   2058    0.0
gi|4426819|gb|AF105289.1|AF105289  Papio hamadryas isolate bab...   2058    0.0
gi|4426818|gb|AF105288.1|AF105288  Papio hamadryas isolate bab...   2058    0.0
gi|4426817|gb|AF105287.1|AF105287  Papio hamadryas isolate bab...   2058    0.0
gi|4102993|gb|AF019379.1|AF019379  Cercopithecus aethiops G-pr...   2058    0.0
gi|2564675|gb|AF023452.1|AF023452  Papio hamadryas anubis CC c...   2058    0.0
gi|14582846|gb|AF349682.1|AF349682  Cercocebus torquatus torqu...   2050    0.0
gi|13873096|gb|AF177891.1|AF177891  Theropithecus gelada C-C c...   2050    0.0
gi|13873068|gb|AF177877.1|AF177877  Mandrillus sphinx C-C chem...   2050    0.0
gi|13873066|gb|AF177876.1|AF177876  Mandrillus leucophaeus C-C...   2050    0.0
gi|5579300|gb|AF081578.1|AF081578  Cercopithecus aterrimus CC       2050    0.0
```

This query takes roughly 10 seconds

What if the Database Does Not Fit in the Main Memory?



Source: Darling *et al.* (2003)

- Darling et al. (2003) show the effect by performing a blastn search when run on a system with 128 MB RAM. The increase in run-time is due to I/O .

HPC for BLAST

- Sequential BLAST is suitable for small number of queries
- HPC solutions for BLAST were developed to cater to large number of queries and also to address the rapid growth in database sizes
- We will review two HPC solutions for BLAST:

1. **mpiBLAST**:

   Darling *et al.* (2003), "The Design, Implementation, and Evaluation of mpiBLAST", *Proc. ClusterWorld.*

2. **ScalaBLAST**:

   Oehmen and Nieplocha (2006), "ScalaBLAST: A Scalable Implementation of BLAST for High-Performance Data-Intensive Bioinformatics Analysis", *IEEE Transactions on Parallel and Distributed Systems*, 17(8):740-749.

mpiBLAST

- Input
  - Set of Queries, $Q=\{q_1,q_2,\ldots,q_m\}$, and
  - Database $D=\{s_1,s_2,\ldots,s_n\}$

- Let p denote the number of processors, $M=\Sigma_{1\leq i\leq m}|q_i|$, and $N=\Sigma_{1\leq i\leq n}|s_i|$

- Algorithm follows the master-worker paradigm (1 master, *p-1* workers)

- Assumption:
  - Q is small enough to fit in the main memory of each worker

- Preferred:
  - Each worker processor has access to a local disk storage supporting contention-free local I/O

mpiBLAST: The Parallel Algorithm

Master

Time

- The database D is *fragmented* into numerous disjoint pieces:

  $F=\{f_1,f_2,\ldots,f_k\},\ \ k>>p$

- The master processor broadcasts all queries in Q to workers

- The master processor records the list of “owners” for each database fragment

- The master then marks all fragments as *unassigned*

Worker

- Each worker p_i *reads* a subset F_i of F into its local storage, s.t., $F=\mathrm{U}_{1\leq i\leq p-1}F_i$

- Each worker sends the list of its local fragments to the master for housekeeping, and also reports that it is *idle*

mpiBLAST: Algorithm …

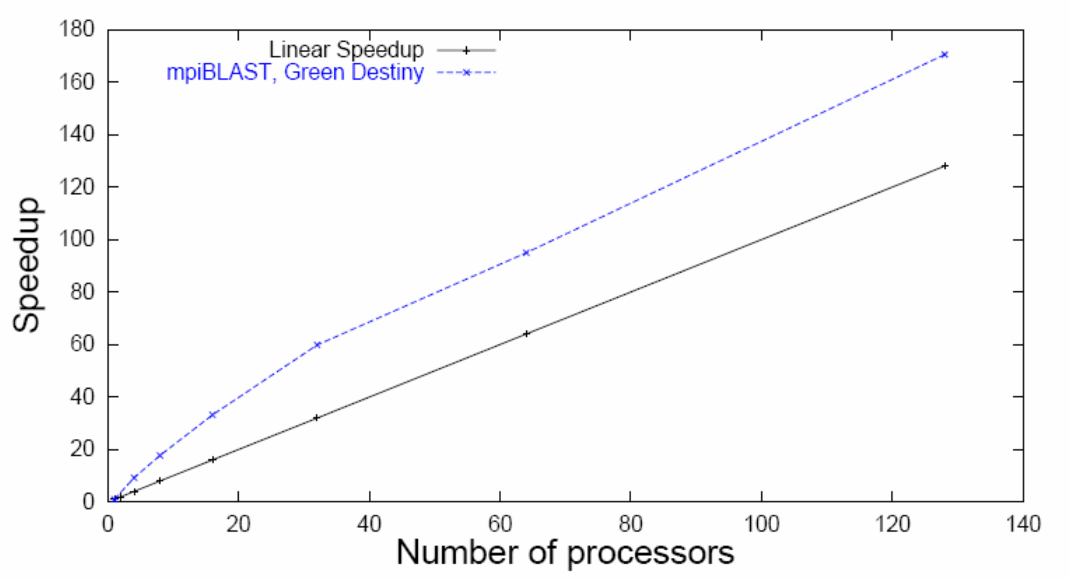
Master

- The master *assigns* each database fragment to one worker. The fragment and order in which to assign is dynamically determined in a "greedy" fashion, as follows:
  - Each p_i is allocated all its unique fragments first
  - Once such unique fragments are exhausted, a fragment f is assigned to p_i, if $f \in F_i$ and f is duplicated in least number of other workers
  - Finally, the remaining unassigned fragments are assigned to workers in decreasing order of their degrees of duplication
- The master processor ranks and *outputs* the hits for each BLAST query

Worker

- Each worker processor *searches* (ie., performs serial BLAST of *Q* against*)* a database fragment assigned by the master.
  - If a fragment is not present in the local storage, it is copied from the corresponding worker that has it
- After searching each fragment, the results are communicated to the master processor

mpiBLAST: Run-time



Source: from Darling *et al.* (2003)

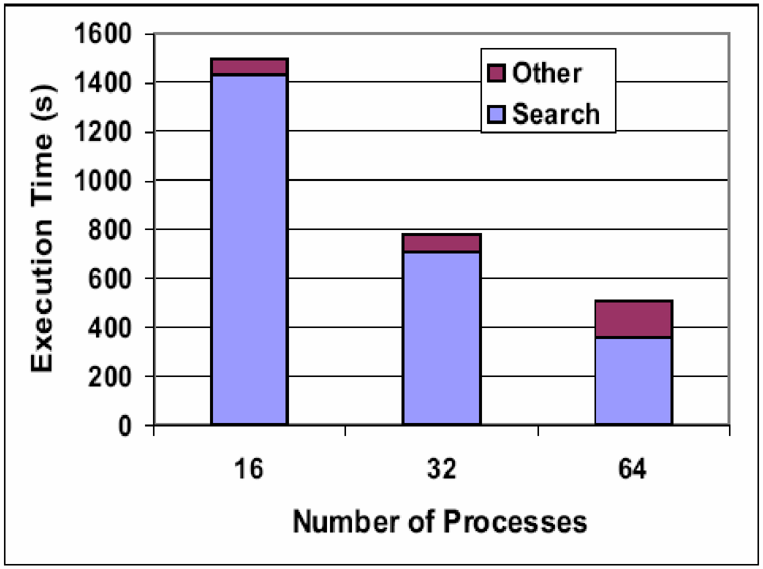
"Green Destiny":

-Beowulf cluster with a 100 Mb/s Ethernet

-Each compute node has a 667 MHz TM5600 CPU, 640 MB RAM, and a 20 GB local hard drive

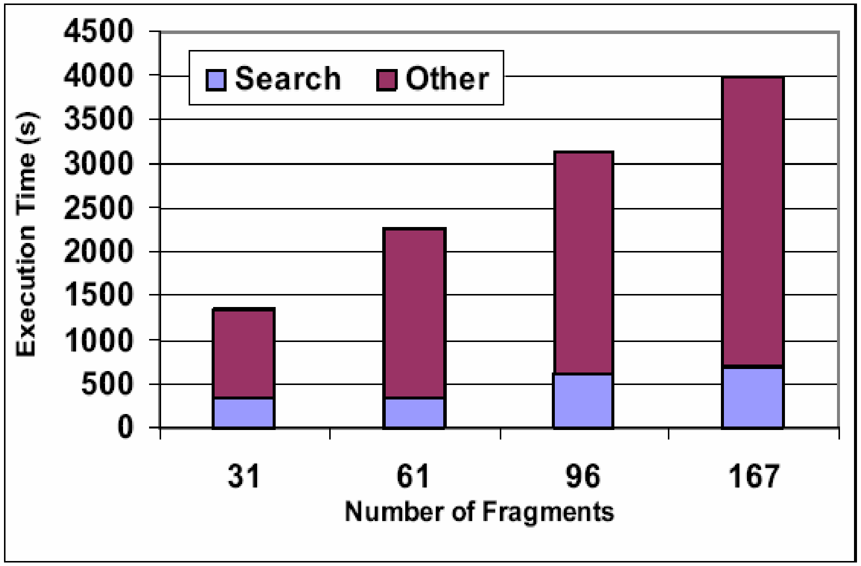
- Database size is 5.1 GB
- Super-linear speedup observed as more memory becomes available for caching a bigger chunk of the local database fragments
- However, efficiency drops because of serial processing of output (during the final reporting step)

mpiBLAST: Effect of the Input and Output Processing Overhead



Source: Lin *et al.* (2005)

Lin *et al.* (2005) observed an almost linear speedup



Source: Lin *et al.* (2005)

Lin *et al.* (2005) also observed a steep rise in the non-search time when the number of database fragments was increased (keeping number of processors fixed).

mpiBLAST: Recent Improvements and Updates

- Parallel I/O for output processing (*mpiBLAST-PIO*)

  Parallel I/O

  - (Local sorting + global merging) for all output records corresponding to each query
  - Improved scalability

- *http://mpiblast.lanl.gov/*

ScalaBLAST: Main Ideas

- Removes I/O dependency by loading the entire target database into (distributed) memory

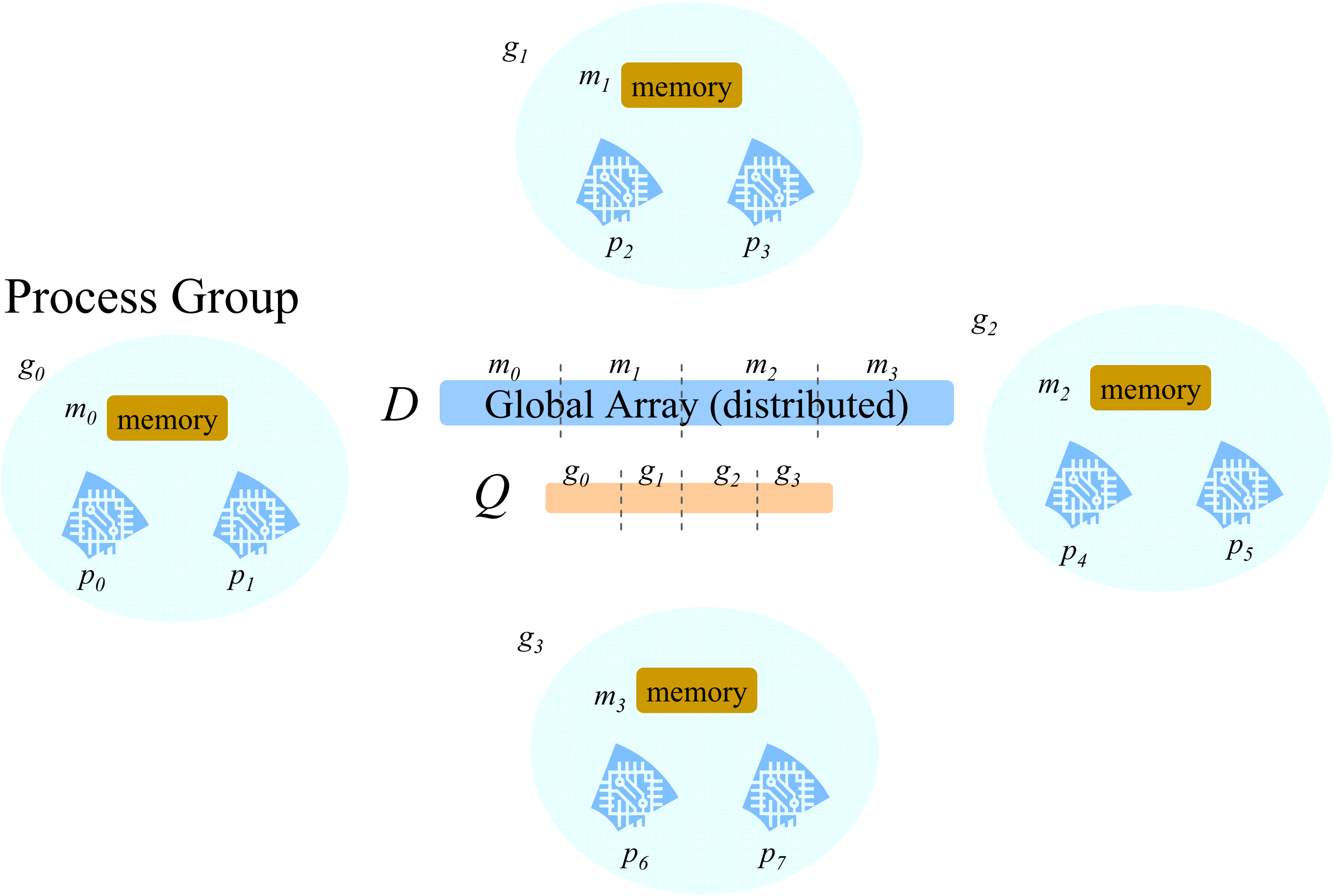
- All processors can access the entire database through *Global Array*, which is an interface for non-uniform memory access

- A query is evaluated entirely by a single processor group to avoid the serialization of reporting results later

- Supports layered parallelism:
  - The work related to each query is shared by processors in a MPI *process group* (compute nodes of an SMP node)
  - The query list itself is partitioned among the process groups

ScalaBLAST: Data and Processor Organization

An example with 8 processors:



ScalaBLAST: The Algorithm

1. Both the database *D* and query list *Q* are evenly *partitioned* across processor groups over their sizes

2. In each process group g_i, the corresponding p_0' and p_1' perform BLAST search on the local query list, one query at a time. For a given query *q*,
   - p_0' performs the BLAST operation on the first half on the database while p_1' performs BLAST operation on the second half
   - Results for *q* are then trivially merged, ranked and reported by one of the processors

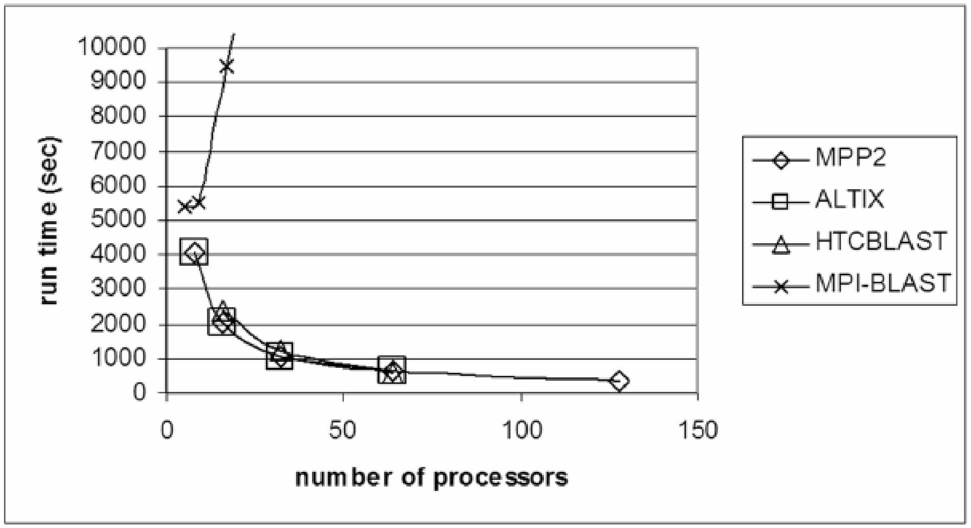
3. Each process element posts a non-blocking request for the next portion of database resident in a remote memory, *before* starting to compute BLAST operation on the current portion of database. This pre-fetching masks communication overhead with computation

ScalaBLAST: Performance Results

- Database: 1.5 million protein sequences ≈ 503 characters
- Query: 1,000 sequences of total size 709 Kbytes
- Experimental Platforms:
  - MPP2, a distributed memory machine with 1.5 GHz Itanium II processors and Quadrics Elan-4 interconnect, 6 to 8 GB RAM/per node
  - SGI Altix, an SMP with 128 1.5 GHz Itanium II processors and with 256 GB.

Phase-wise Run-time

	Setup %	Query %	Output %
|Q|=100 p=8	~ 2.5	~ 95	~ 2.5
|Q|=1000 p=8	< 0.1	~ 98.5	~ 1.4
|Q|=1000 p=32	< 0.3	~ 98.3	~ 1.5



Source: Oehman and Nieplocha (2006)

More information about ScalaBLAST

- *http://hpc.pnl.gov/projects/scalablast/*

Selected Bibliography for Alignment Topics

Papers

- S. Needleman and C. Wunsch (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins, *J. Molecular Biology*, 48:443-453.
- D. Hirschberg (1975). A linear space algorithm for computing maximal common subsequences. *Communications of the ACM*, 18(6):341-343.
- T. Smith and M. Waterman (1981). Overlapping genes and information theory, *J. Theoretical Biology*, 91:379-380.
- O. Gotoh (1982). An improved algorithm for matching biological sequences. *J. Molecular Biology*, 162(3):705-708.
- J. Fickett (1984). Fast optimal alignment. Nucleic Acids Research, 12(1):175-179.
- M.S. Gelfand *et al.* (1996). Gene recognition via spliced alignment. *Proc. National Academy of Sciences*, 93(17):9061-9066.
- A. Delcher *et al.* (1999). Alignment of whole genomes. *Nucleic Acids Research*, 27(11):2369-2376.
- X. Huang and K. Chao (2003). A generalized global alignment algorithm. *Bioinformatics*, 19(2):228-233.
- S. Rajko and S. Aluru (2004). Space and time optimal parallel sequence alignments. *IEEE Transactions on Parallel and Distributed Systems*, 15(12):1070-1081.

Books

- D. Gusfield (1997). Algorithms on strings, trees and sequences: Computer Science and Computational Biology. Cambridge University Press, Cambridge, London.
- J. Setubal and J. Meidanis (1997). Introduction to computational molecular biology. PWS Publishing Company, Boston, MA.
- B. Jackson and S. Aluru (2005). Chapter: “Pairwise sequence alignment” in Handbook of computational molecular biology, Ed. S. Aluru, Chapman & Hall/CRC Press.

Selected Bibliography for BLAST Related Topics

Serial BLAST

- S. Altschul *et al.* (1990). Basic Local Alignment Search Tool, *J. Molecular Biology*, 215:403-410.
- W. Gish and D.J. States (1993). Identification of protein coding regions by database similarity search. *Nature Genetics*. 3:266-272.
- T.L. Madden *et al.* (1996). Applications of network BLAST server. *Meth. Enzymol*. 266:131-141.
- S. Altschul, *et al.* (1997). Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 25:3389-3402.
- Z. Zhang *et al.* (2000). A greedy algorithm for aligning DNA sequences. *J. Computational Biology*, 7(1-2):203-214.

HPC BLAST

- T. Rognes (2001). ParAlign: A parallel sequence alignment algorithm for rapid and sensitive database searches, Nucleic Acids Research, 29:1647-1652.
- R. Bjornson *et al. (*2002). TurboBLAST®: A parallel implementation of BLAST built on the TurboHub, *Proc. International Parallel and Distributed Processing Symposium.*
- A. Darling, L. Carey and W.C. Feng (2003). The design, implementation, and evaluation of mpiBLAST, *Proc. ClusterWorld.*
- D. Mathog (2003). Parallel BLAST on split databases, *Bioinformatics*, 19:1865-1866.
- J. Wang and Q. Mu (2003). Soap-HT-BLAST: High-throughput BLAST based on web services, *Bioinformatics*, 19:1863-1864.
- H. Lin *et al.* (2005). Efficient data access for parallel BLAST, *Proc. International Parallel and Distributed Processing Symposium*.
- K. Muriki, K. Underwood and R. Sass (2005). RC-BLAST: Towards a portable, cost-effective open source hardware implementation, *Proc. HiCOMB 2005.*
- M. Salisbury (2005). Parallel BLAST: Chopping the database, *Genome Technology*, pp 21-22.

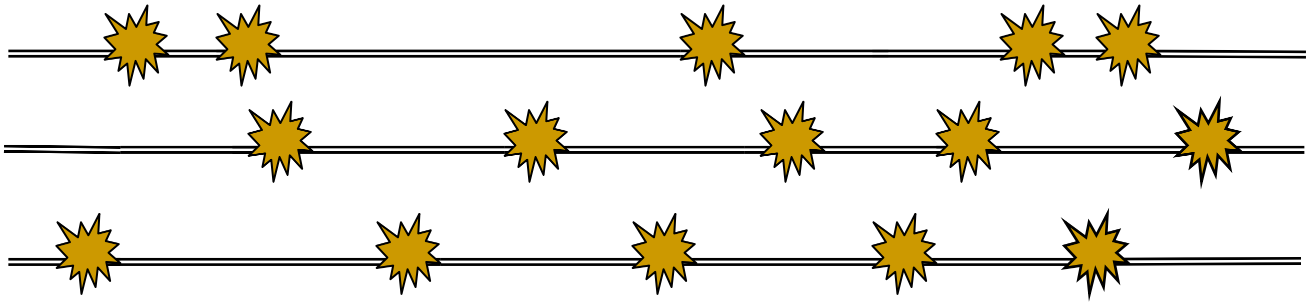
NCBI BLAST - Web Resources

- NCBI BLAST Webpage:
  http://www.ncbi.nlm.nih.gov/BLAST/
- For a comprehensive list of BLAST related references:
  http://www.ncbi.nlm.nih.gov/blast/blast_references.shtml

Part II:
Large-Scale Sequence Analysis

Genome Assembly

Input: Multiple copies of the same genome



Process: Randomly fragment each copy

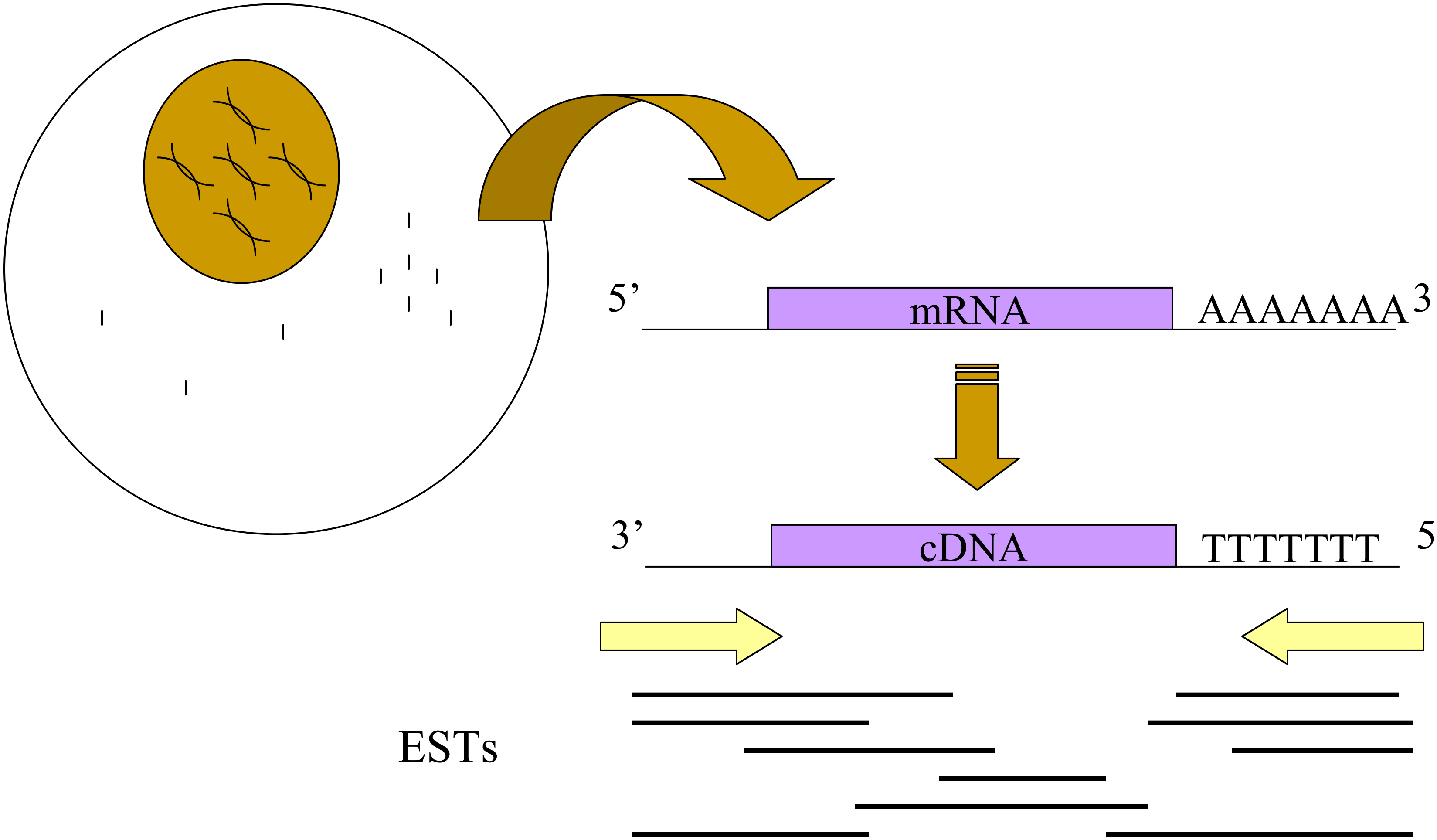
Genome Assembly

Output: Unordered genome fragments

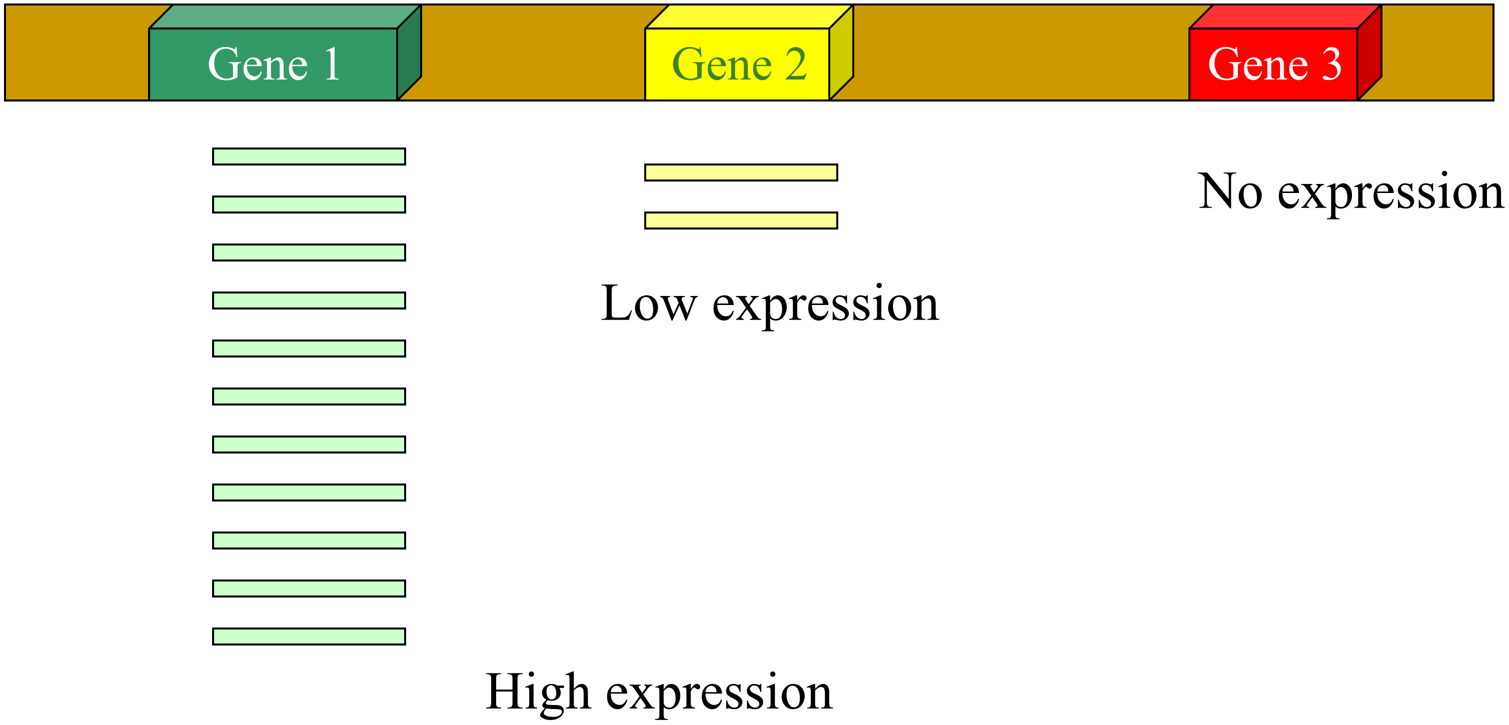
Sequence Assembly Required!

PHASE TWO : INTERPRETATION
©SHENEMAN The Star-Ledger
I THINK I FOUND A CORNER PIECE.
3 BILLION PIECES
GENOME

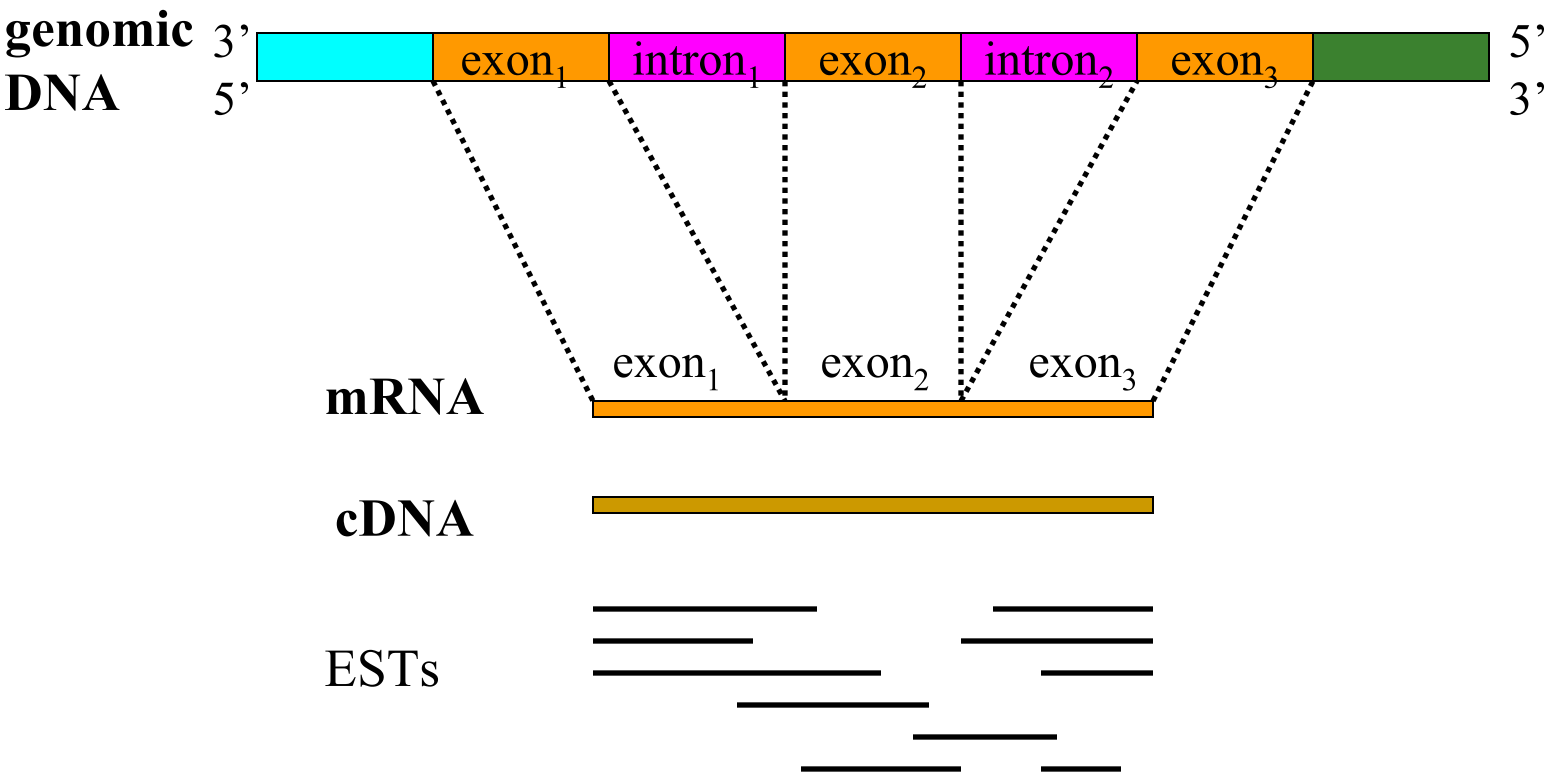
EST Clustering



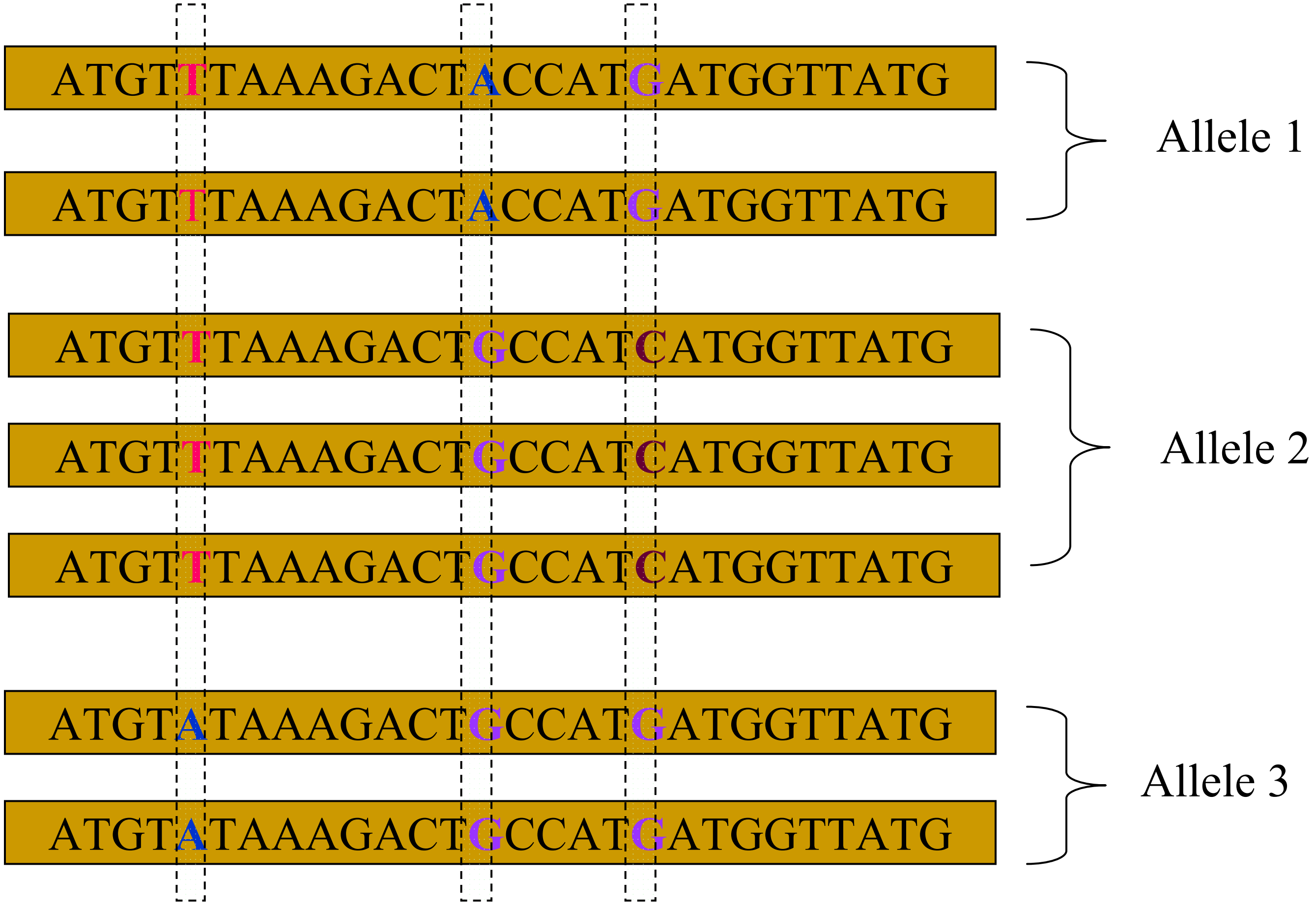
Genes Are Not Uniformly Sampled



EST Based Gene Discovery



Single Nucleotide Polymorphisms (SNPs)



ATGTTTAAAGACTACCATGATGGTTATG

ATGTTTAAAGACTACCATGATGGTTATG

Allele 1

ATGTTTAAAGACTGCCATCATGGTTATG

ATGTTTAAAGACTGCCATCATGGTTATG

ATGTTTAAAGACTGCCATCATGGTTATG

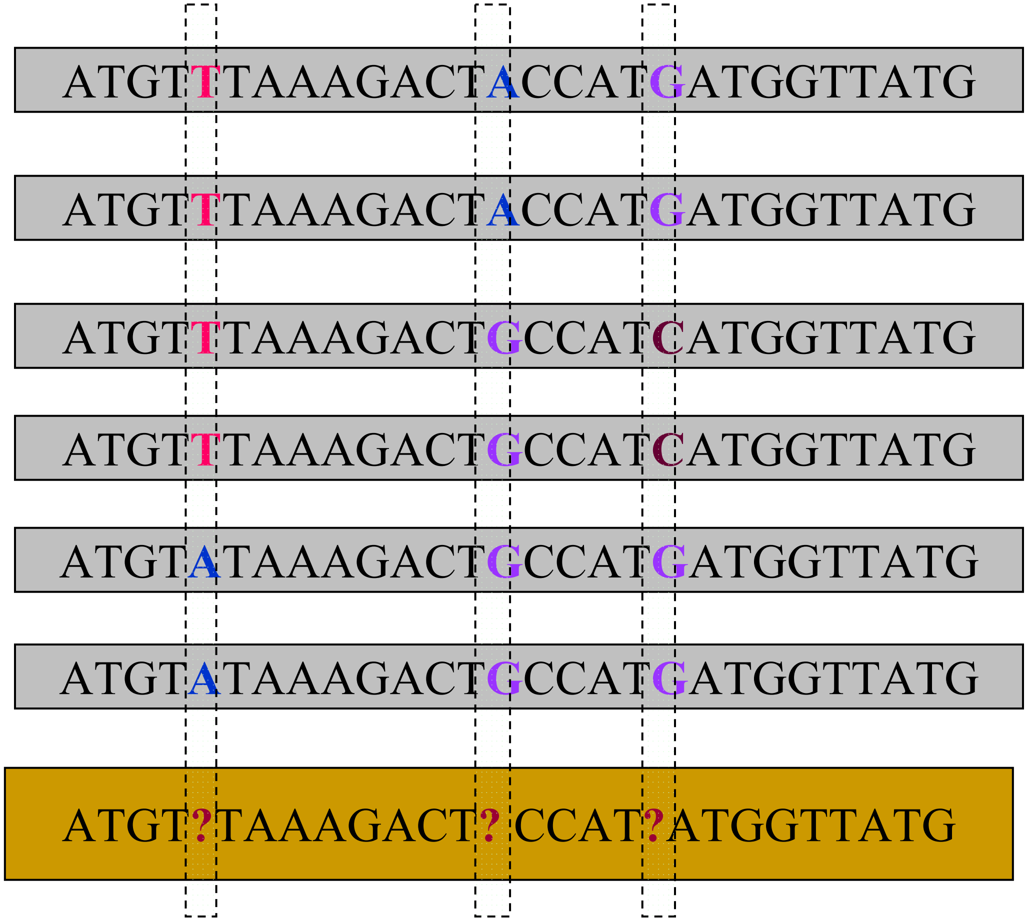
Allele 2

ATGTATAAAGACTGCCATGATGGTTATG

ATGTATAAAGACTGCCATGATGGTTATG

Allele 3

SNPs Based on Assembly



ATGTTTAAAGACTACCATGATGGTTATG

ATGTTTAAAGACTACCATGATGGTTATG

ATGTTTAAAGACTGCCATCATGGTTATG

ATGTTTAAAGACTGCCATCATGGTTATG

ATGTATAAAGACTGCCATGATGGTTATG

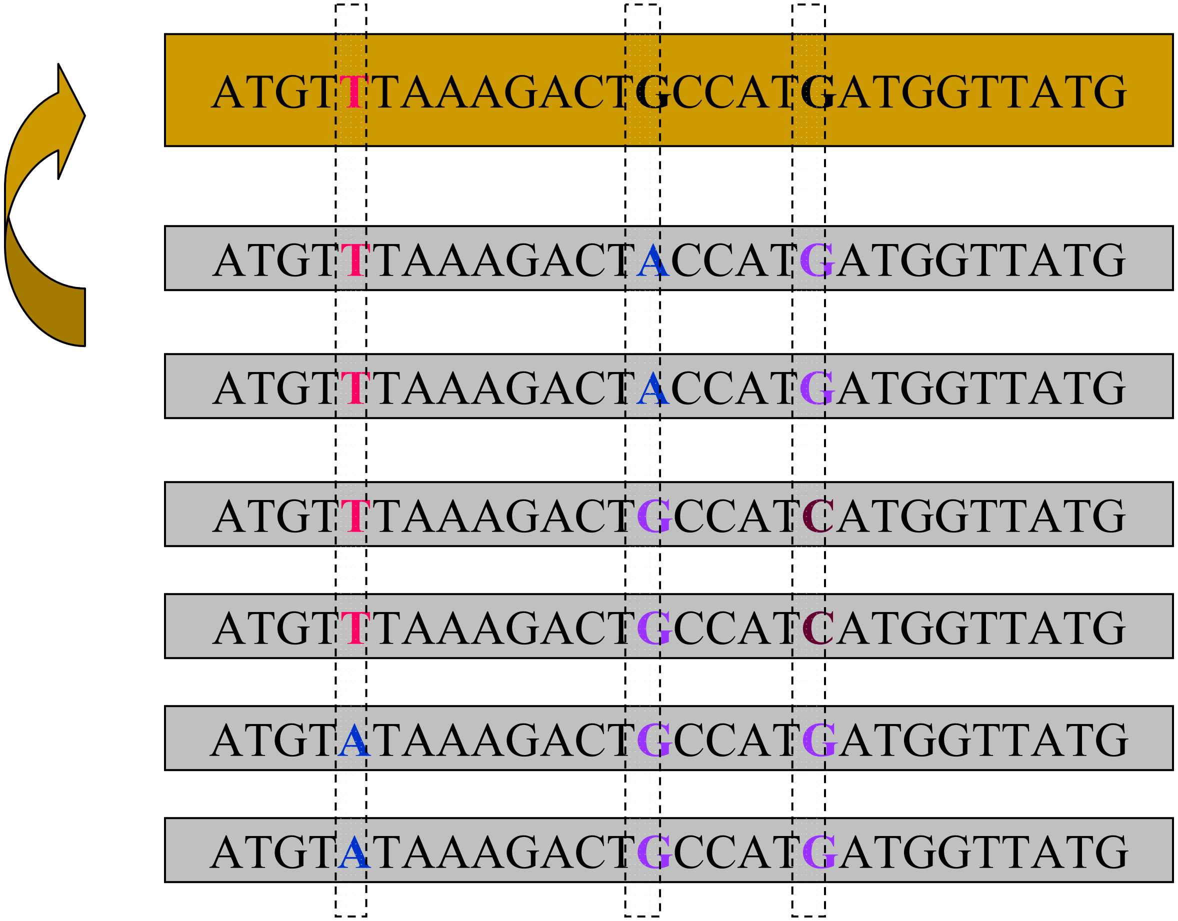
ATGTATAAAGACTGCCATGATGGTTATG

ATGT?TAAAGACT?CCAT?ATGGTTATG

Alignment of related genomic sequences

Consensus

SNPs Based On Clustering



ATGTTTAAAGACTGCCATGATGGTTATG

ATGTTTAAAGACTACCATGATGGTTATG

ATGTTTAAAGACTACCATGATGGTTATG

ATGTTTAAAGACTGCCATCATGGTTATG

ATGTTTAAAGACTGCCATCATGGTTATG

ATGTATAAAGACTGCCATGATGGTTATG

ATGTATAAAGACTGCCATGATGGTTATG

Genome

Samples that
are aligned
to the consensus

Naïve Approach

All vs. All alignments + post processing

Compute-intensive and wasteful!

- 33 million fragments for mouse assembly
- 7+ million human ESTs

Typical Methodology

- Identify pairs of fragments that have a good exact match (promising pairs).

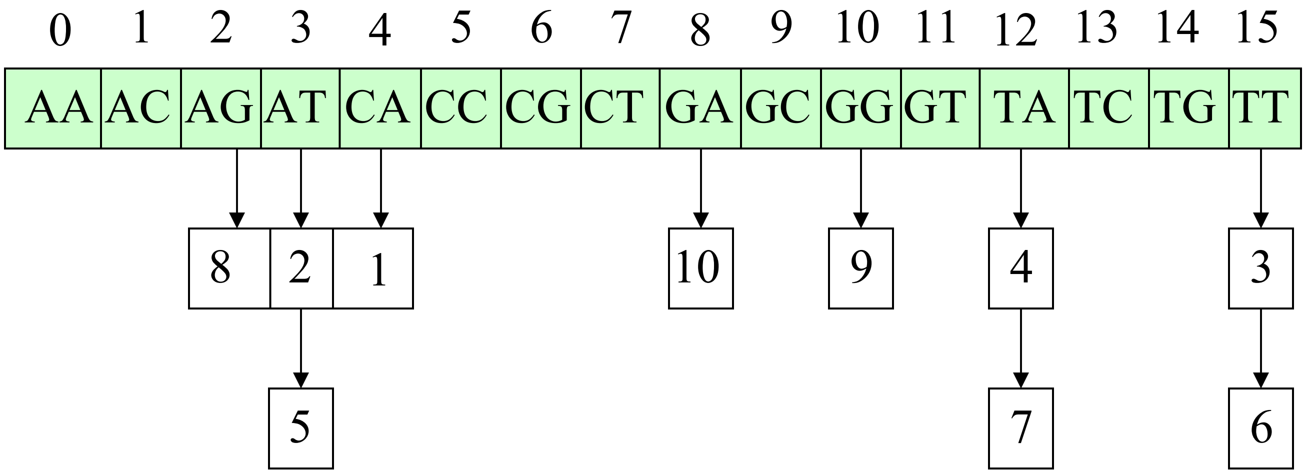
- Restrict alignment computations to promising pairs.

- Perform post-processing.

Lookup Table Pair Generation

1 2 3 4 56 7 8 9 10 11

CATTATTAGGA



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
AA	AC	AG	AT	CA	CC	CG	CT	GA	GC	GG	GT	TA	TC	TG	TT

Problems for Large-scale Analysis

- Longer matches are revealed as multiple short matches in a lookup table based approach.

- Matches are arbitrarily generated.

- Linear space for uniformly random overlaps with constant coverage but worst-case quadratic in the non-uniform case.

PaCE Methodology

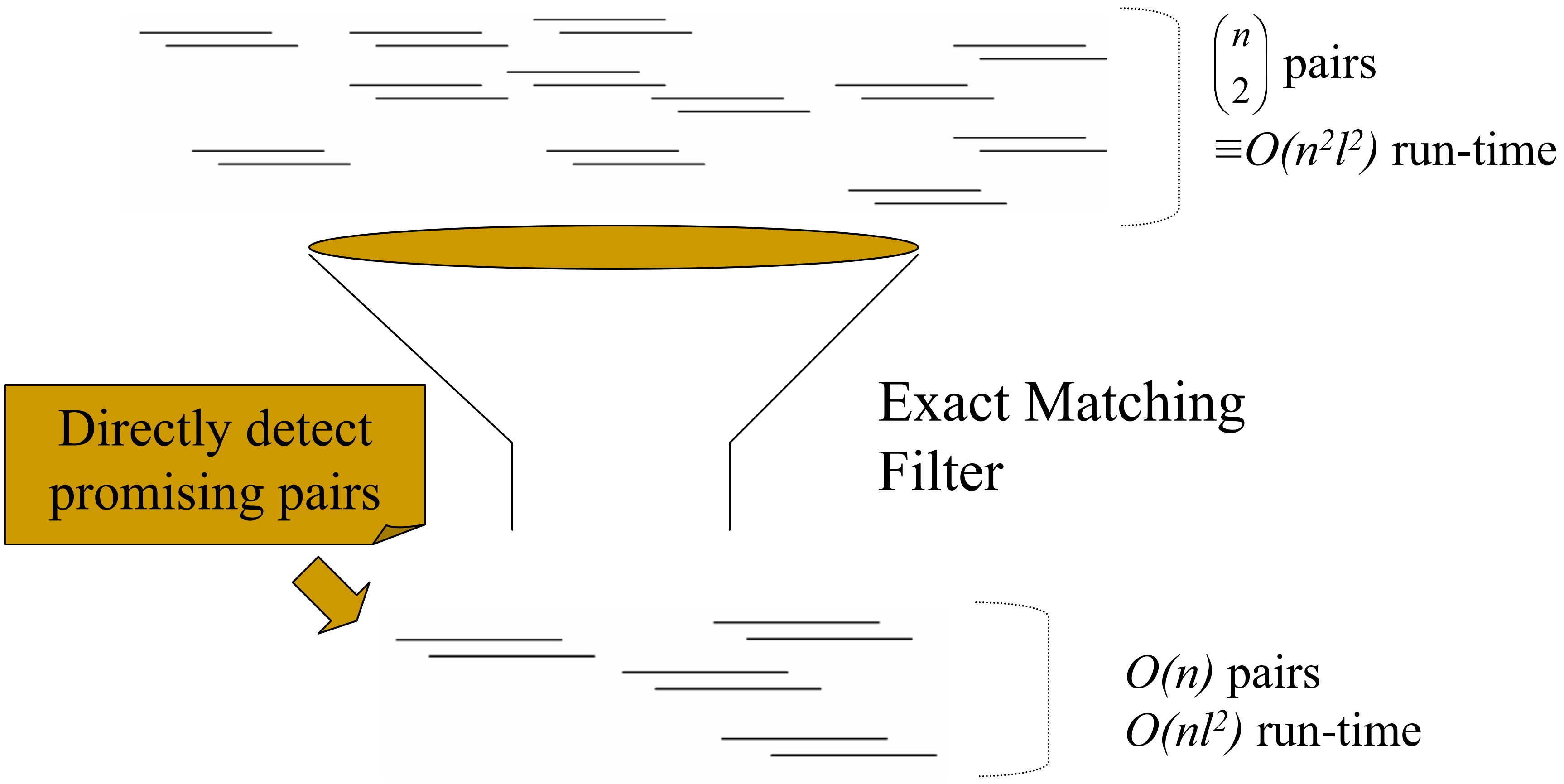
- Reduce space requirement from quadratic to linear.
- Generate promising pairs in decreasing order of maximal common substring length.
- Constant time per generation of a pairwise maximal common substring.
- Significantly reduce number of alignments without affecting quality.
- Massively parallel processing – reduce run-time; increase available memory.

A Specific Application: Maize Genome Assembly

Why sequence the maize genome?

- Maize (*i.e.*, corn) is an economically important crop.
- Best studied model organism for the cereal crops.
- Just as the human genome project will intensify upcoming medical advances, cereal genomes (rice and maize) will help improve worldwide food production.

Typical Assembly Strategy



$\binom{n}{2}$ pairs

$\equiv O(n^2 l^2)$ run-time

Exact Matching Filter

Directly detect promising pairs

$O(n)$ pairs

$O(nl^2)$ run-time

Genome Assembly Example

- Human Genome Assembly (Venter *et al*. 2001):
  - Input: 27 million fragments
  - Program: Celera Assembler
  - 10,000 CPU hours for detecting overlaps
    - Parallelized to run on 64 GB shared memory machine + 10 4-processor SMPs with 4-GB memory
  - 10,000 CPU hours for the rest

Maize Genome Assembly

- Maize genome is comparable in size to the human genome (2.5 GB) but is highly repetitive (65-80%). About 15-20% is gene space.

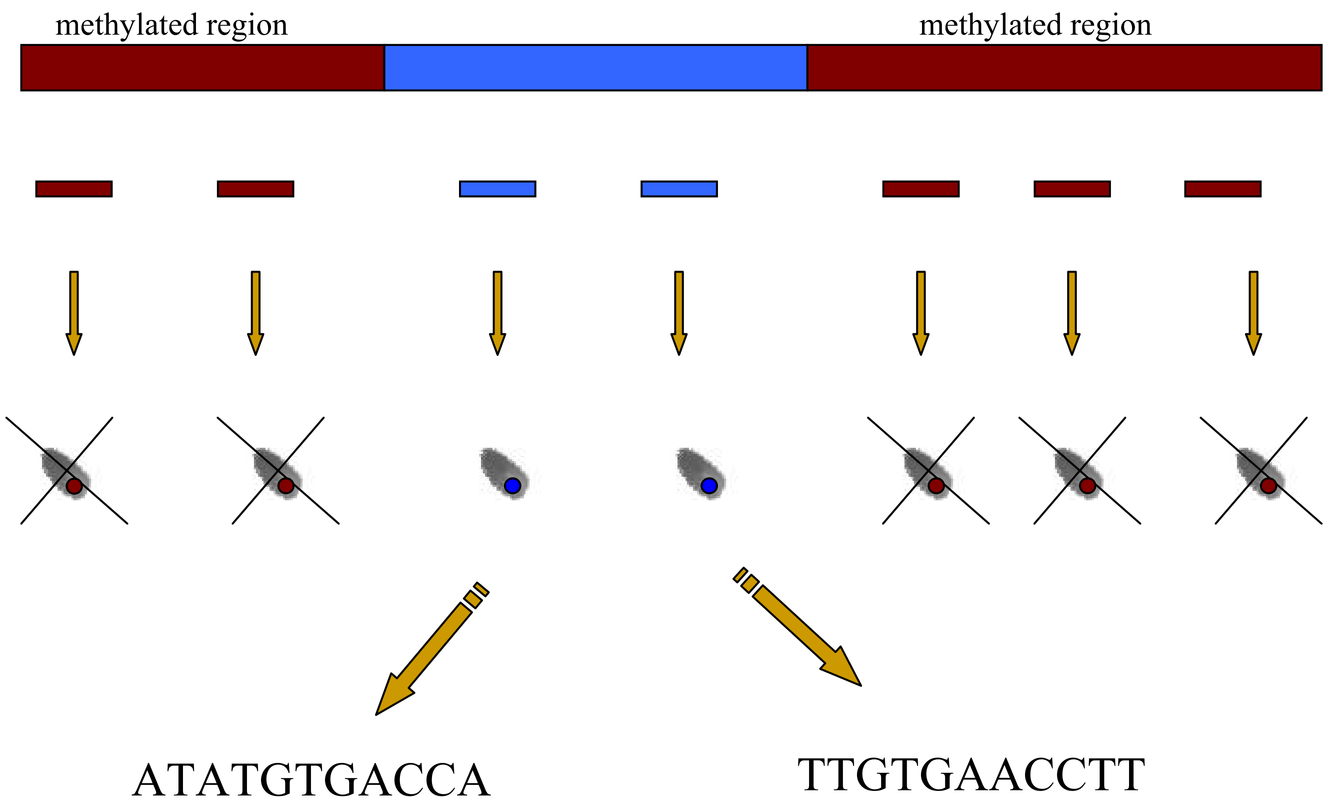
- NSF Workshop in July 2001 to debate sequencing strategies

Maize Genome Assembly

NSF funded pilot projects (2002; $10.2 million):

- "gene-enrichment" – Consortium for Maize Genomics (Danforth Center, TIGR, Purdue & Orion Genomics)
  - Methylation filtration (MF)
  - High $C_o t$ selection (HC)
- BAC sequencing – Rutgers & Univ. of Arizona.
- Dept. of Energy (DOE) added about 2.4 million fragments.

Methylation Filtration



methylated region

methylated region

1.) *Fragment*

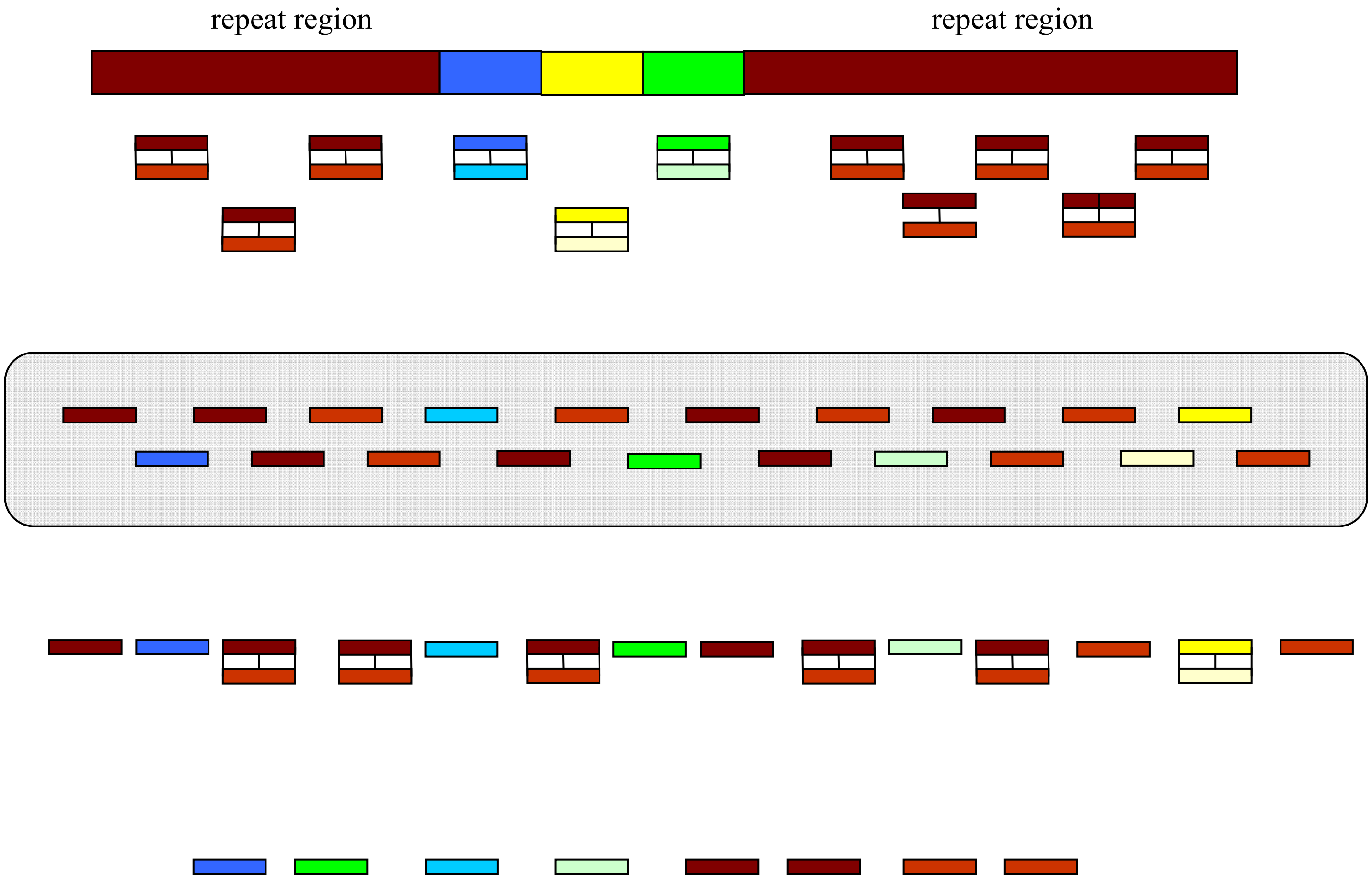
2.) *Clone into special bacteria*

3.) *Sequence*

ATATGTGACCA

TTGTGAACCTT

High C_ot Selection



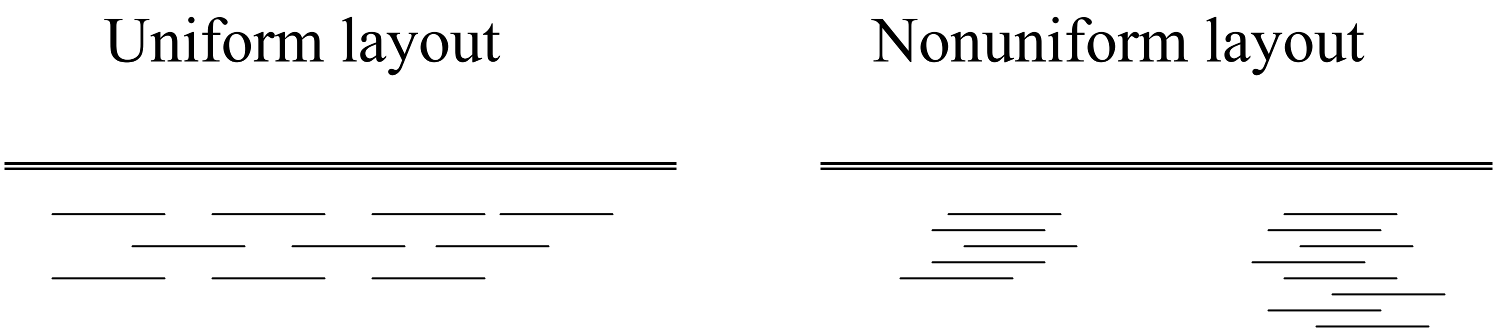
1.) *Fragment dsDNA*

2.) *Denature into ssDNA*

3.) *Slowly reform dsDNA*

4.) *Sequence remaining ssDNA*

Random vs. Biased Sampling



- Uniform case – *O(n)* overlaps
- Non-uniform case – *O(n^2)* overlaps

PaCE Methodology

- First cluster, then assemble.
- Two sequences fall in the same cluster if there is a chain of overlaps that leads from one sequence to the other.
- Each cluster can be assembled into a contig.

Clustering Strategy

- Initially, treat each sequence as a cluster by itself.
- If two sequences from two different clusters show significant overlap, merge the clusters.
- Use union-find data structure.

Processing High-quality Overlaps first is important!

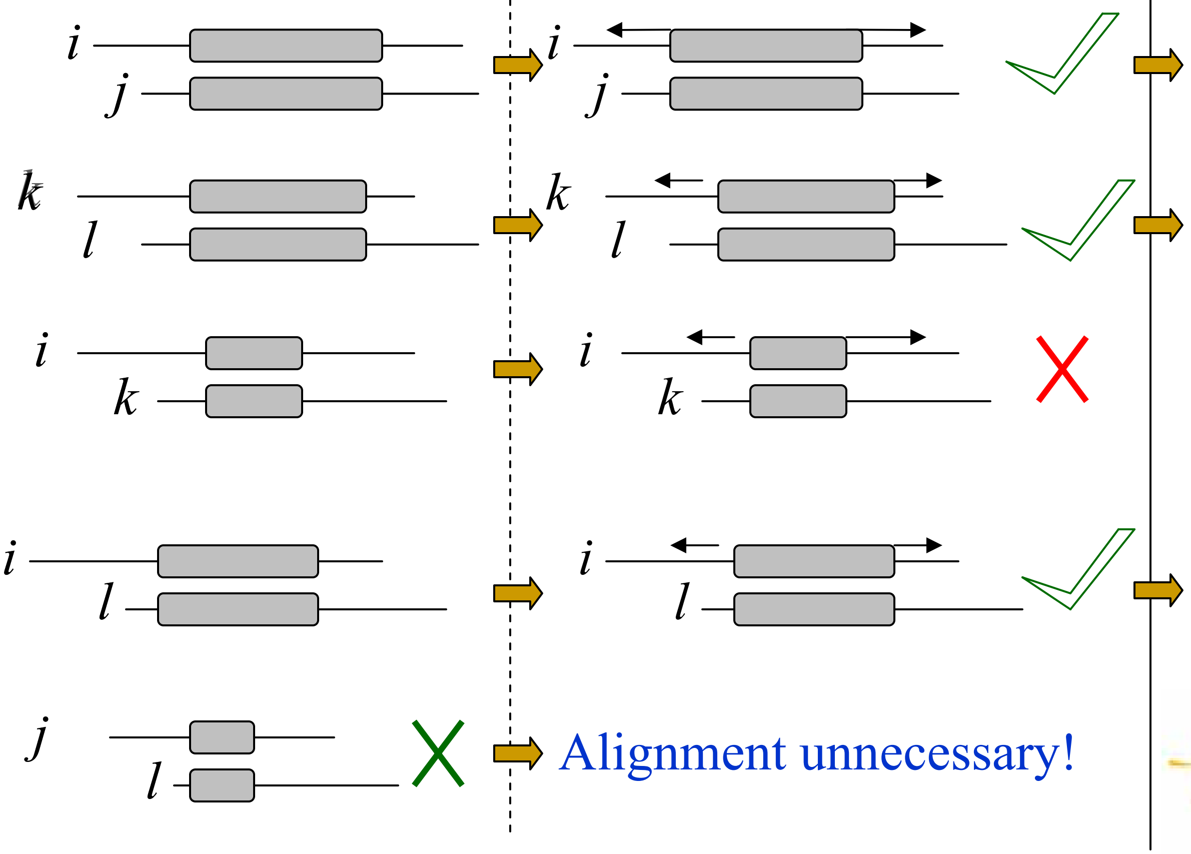
Successful overlap results in

- Merging of two clusters.
- No need to test other promising pairs of fragments where a member of the pair comes from each constituent cluster.

Clustering Heuristic

Promising pairs | Pairs aligned | Clustering

Initialize: {i}, {j}, {k}, {l}



Alignment unnecessary!

{i,j}, {k}, {l}

{i,j}, {k,l}

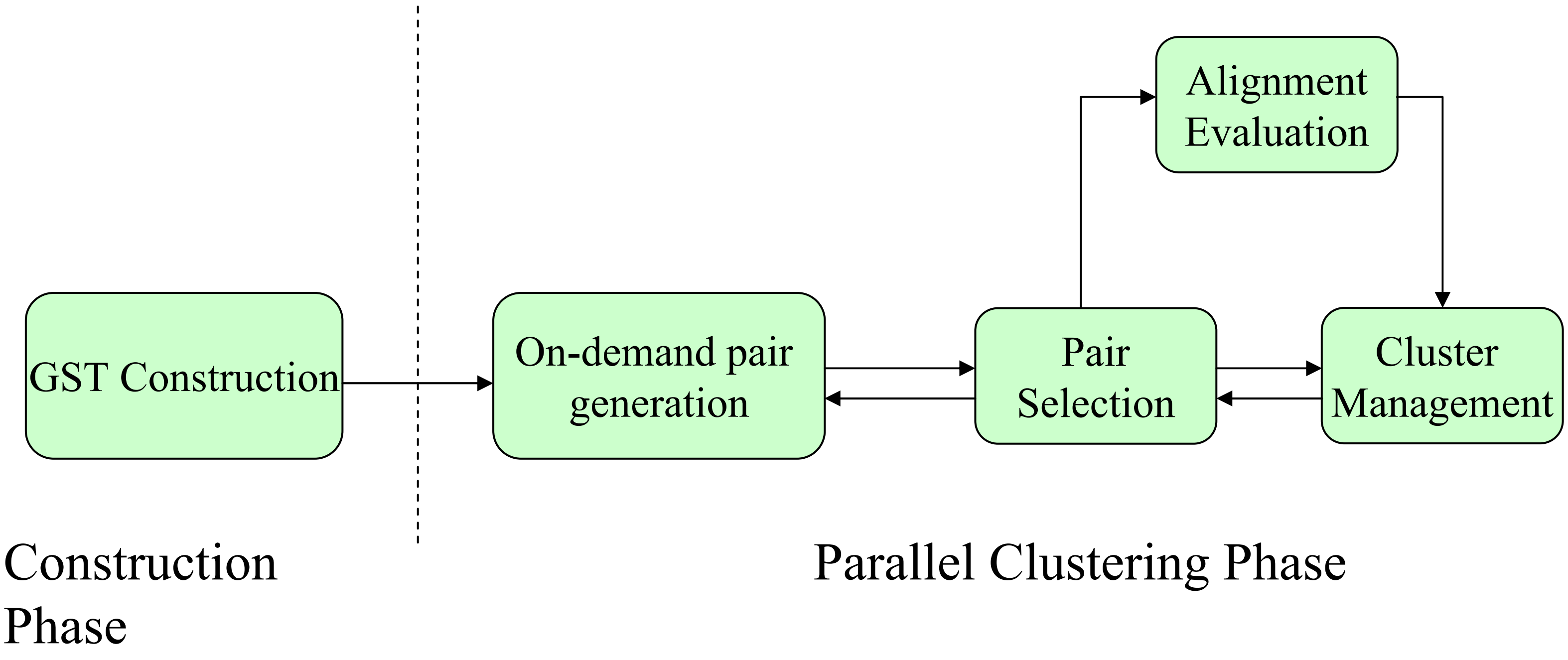
{i,j,k,l}

Pair generation order matters !

Pair Generation Methodology

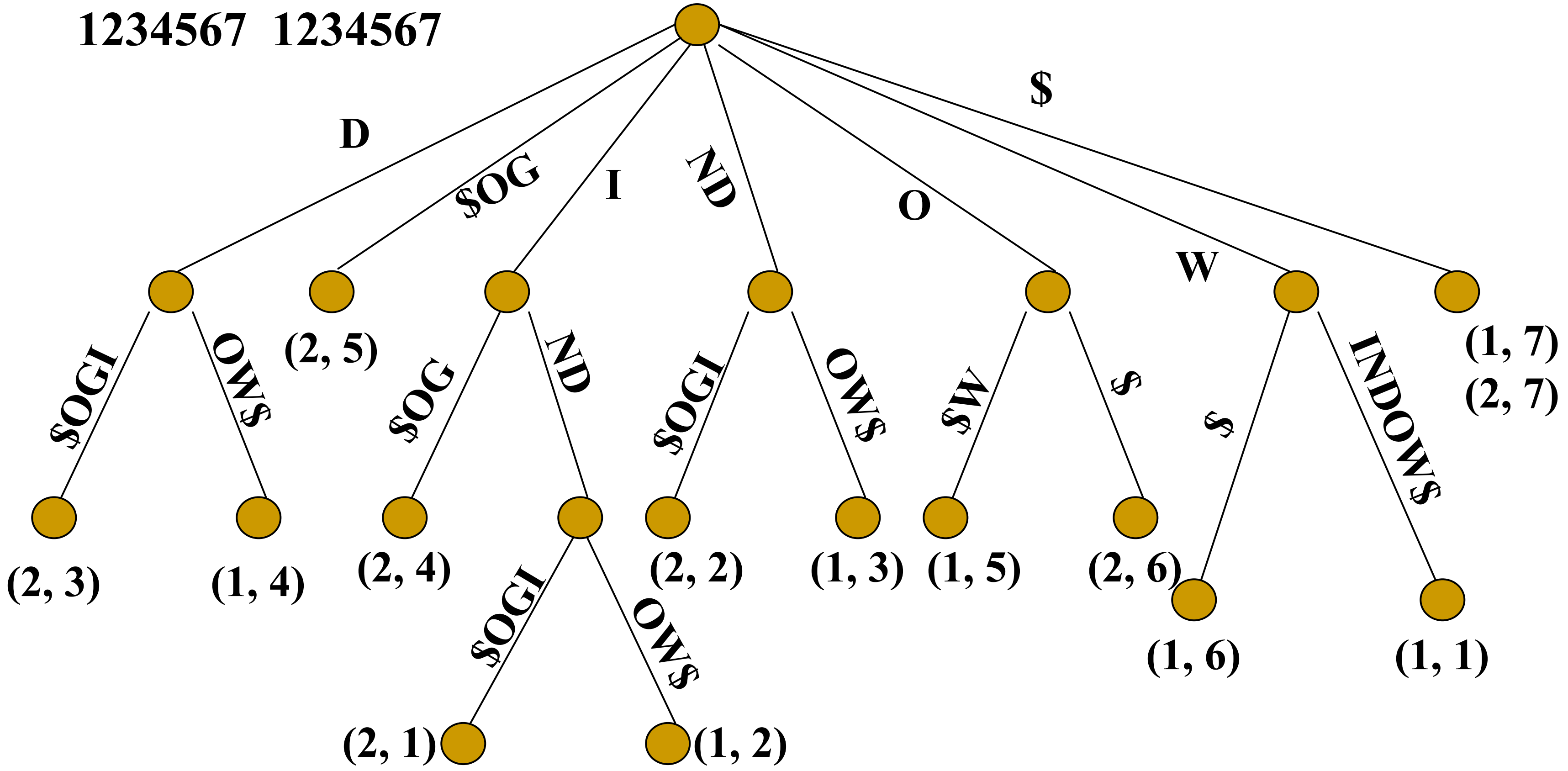
- Generate pairs
  - In non-increasing order of maximal common substring length
  - On-demand without storing previously generated pairs
  - O(1) amortized time per pair
  - Using linear space

PaCE Software Architecture

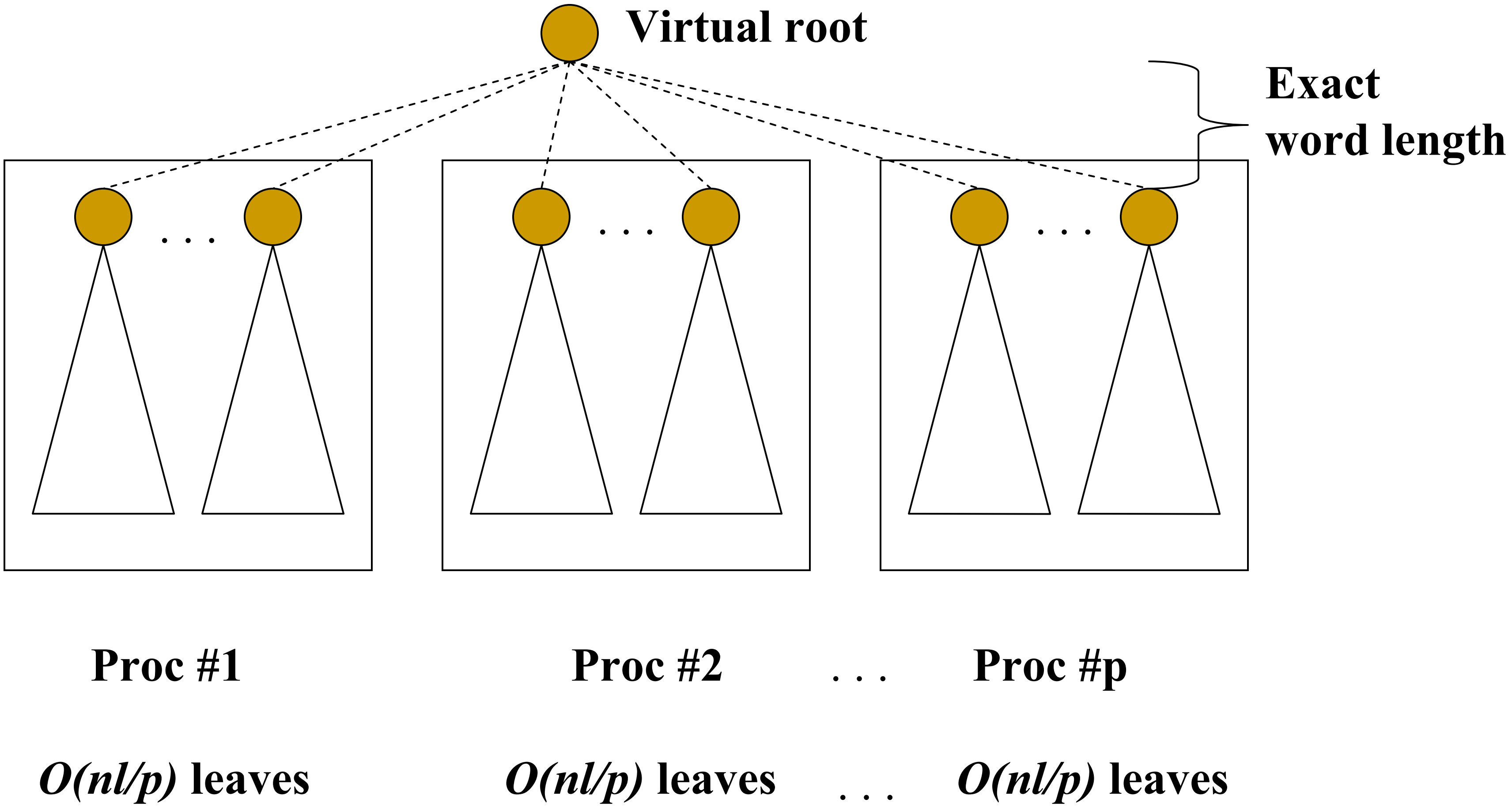


Generalized Suffix Tree (GST)

WINDOW$ INDIGO$
1234567 1234567



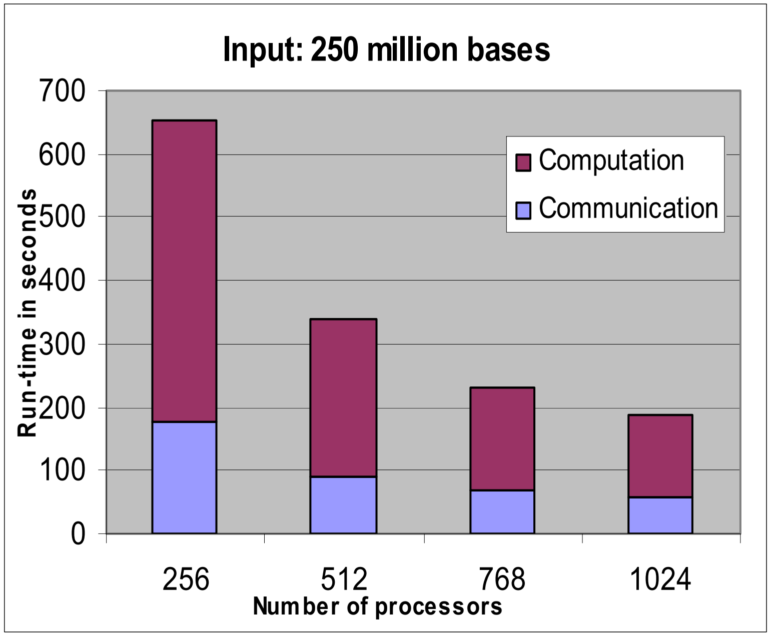
Parallel Construction of GST

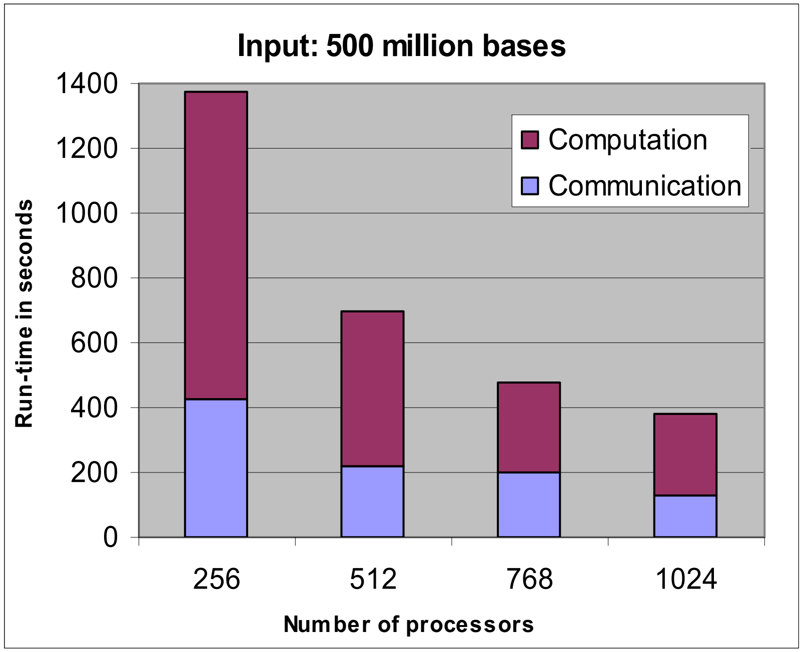


Parallel Construction of GST

- Bucket the suffixes of the sequences based on the first *k* bases.
- Redistribute the suffixes in parallel such that each processor owns a set of buckets.
- Build GST locally in each processor.
- In each processor, #leaves = $O(nl/p)$
- Run-time = $O(nl^2/p)$

GST Construction on BlueGene/L





Pair Generation Algorithm

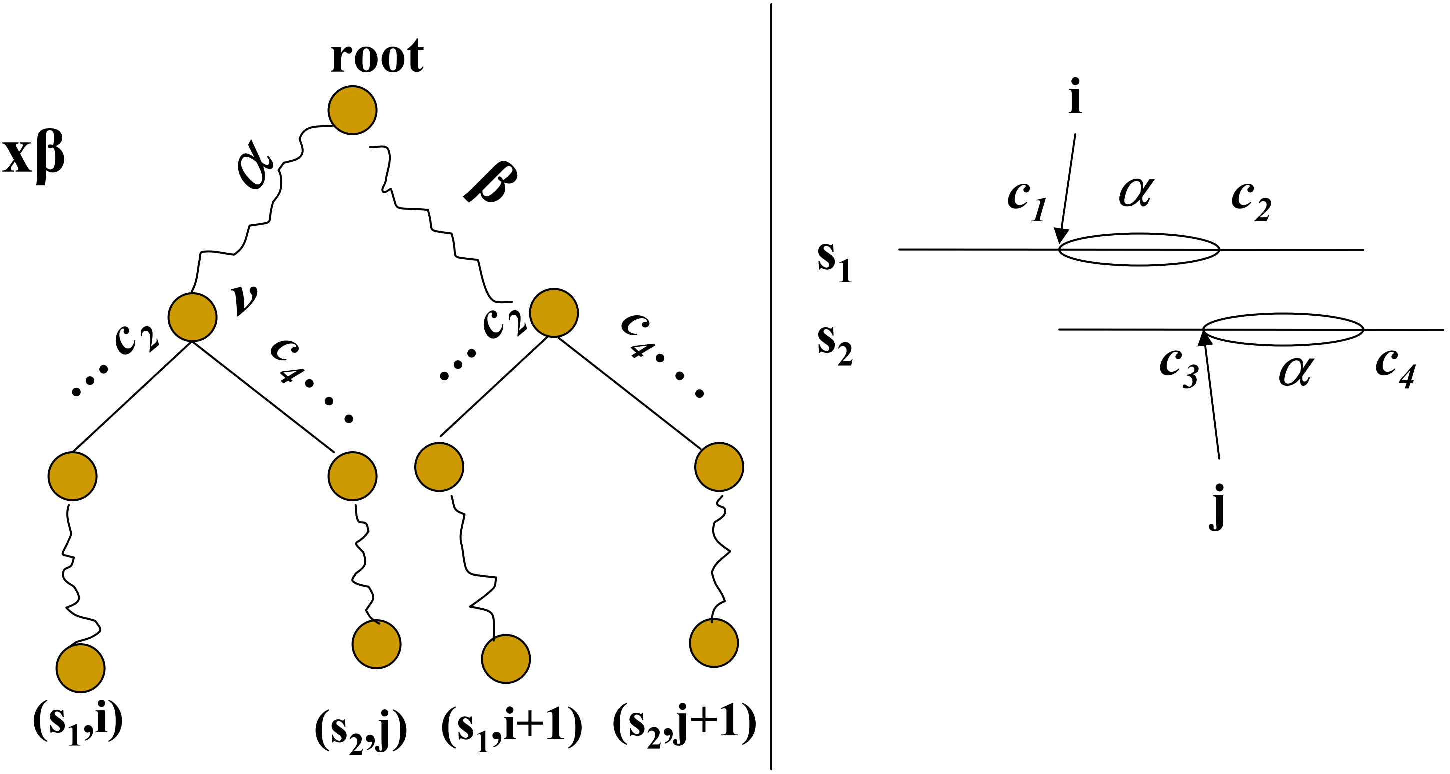
- Process the nodes in the local GST in the decreasing order of string-depth and generate pairs at each node.

- Generate a pair at a node only if the corresponding overlap is maximal.

Main Idea of the Algorithm

- *Maximal match*

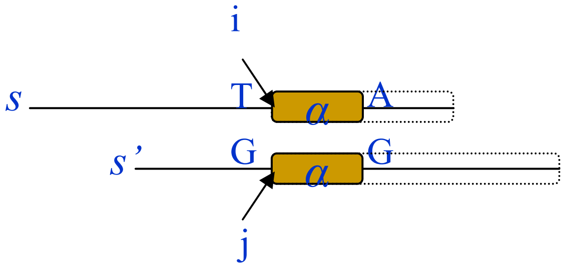
$\alpha = x\beta$



Left Character Sets (*lsets*)

- *leaf-set(v)* = set of strings whose suffixes are present in the subtree of *v*.
- *lset (v)* = partition of *leaf-set(v)* into $|\Sigma|+1$ subsets, $l_A(v)$, $l_C(v)$, $l_G(v)$, $l_T(v)$, $l_\lambda(v)$.

Maximal Match Detection



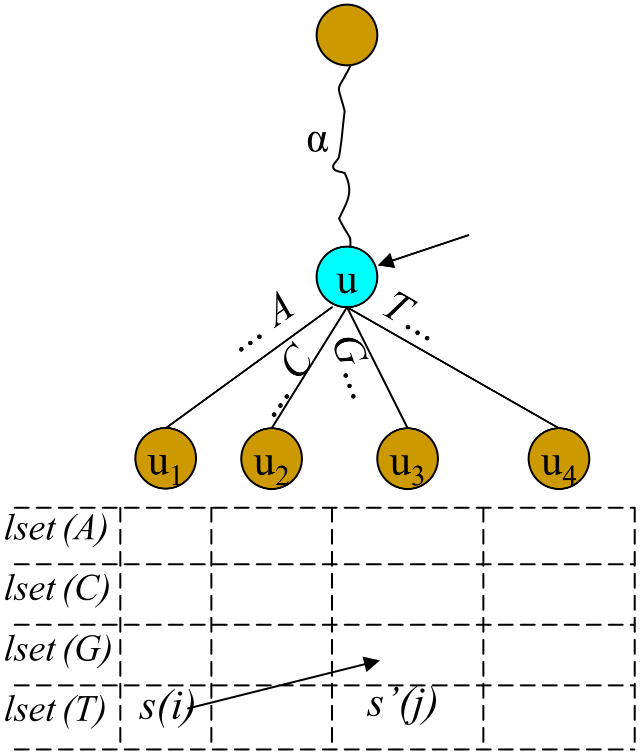
- **Right Maximality**

  ≡ *s(i)* and *s'(j)* are in subtrees of two different children of *u*

- **Left Maximality**

  ≡ $s[i-1] \neq s'[j-1]$, if $i>1$ and $j>1$

Pair generation at an internal node u



lset (A)				
lset (C)				
lset (G)				
lset (T)	*s(i)*		*s'(j)*	

Run-time: *O(1)* per pair

Run-time for Pair Generation

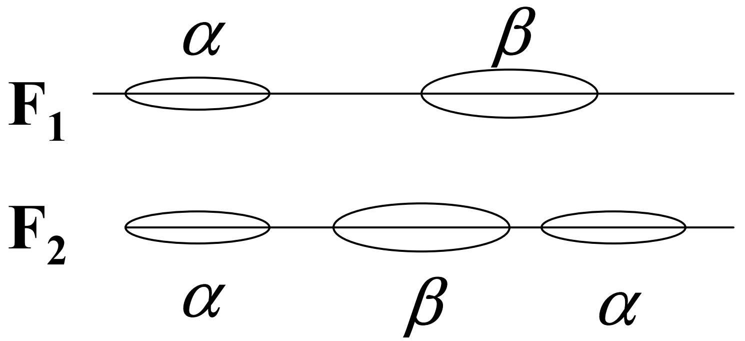
- Sorting of nodes in the local GST

  $= O(nl/p)$

- Processing of all nodes in the local GST

  $= O(\#\ pairs\ generated)$

Number of Duplicates



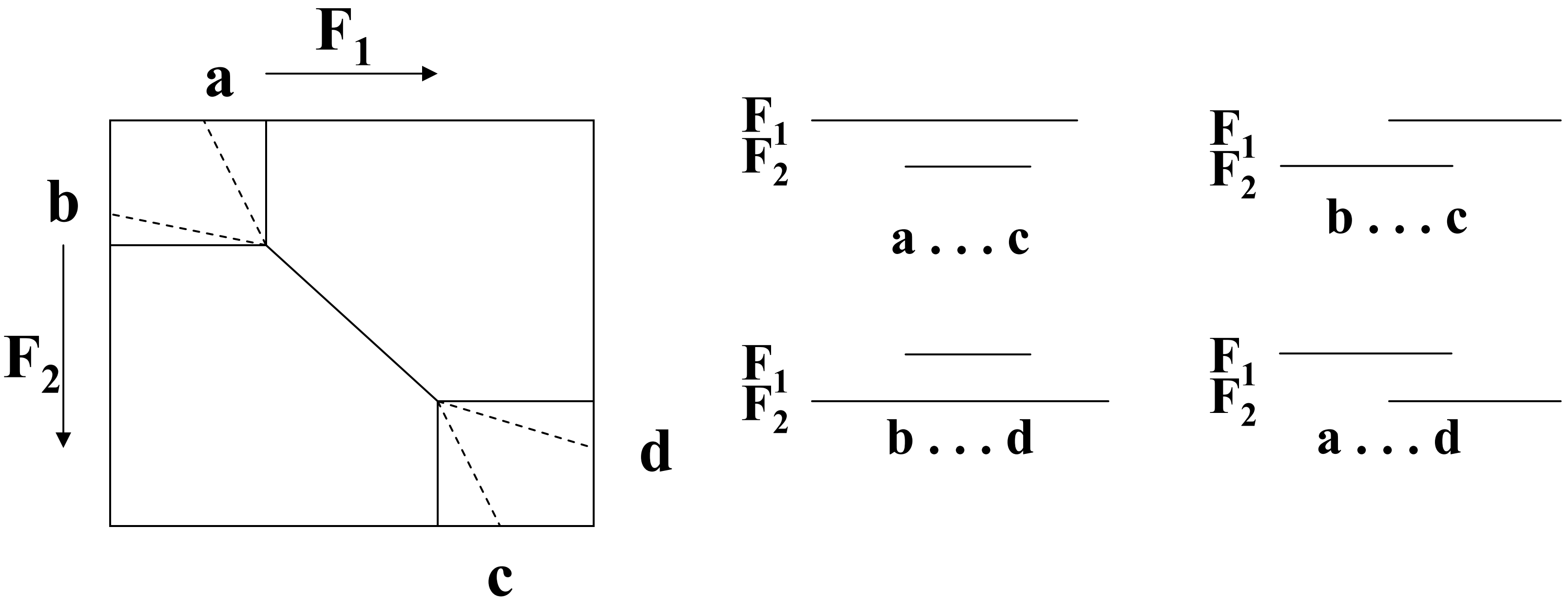
$|\alpha|, |\beta| \geq \psi$

eg., (F_1, F_2) is generated at most twice.

# of times a pair is generated

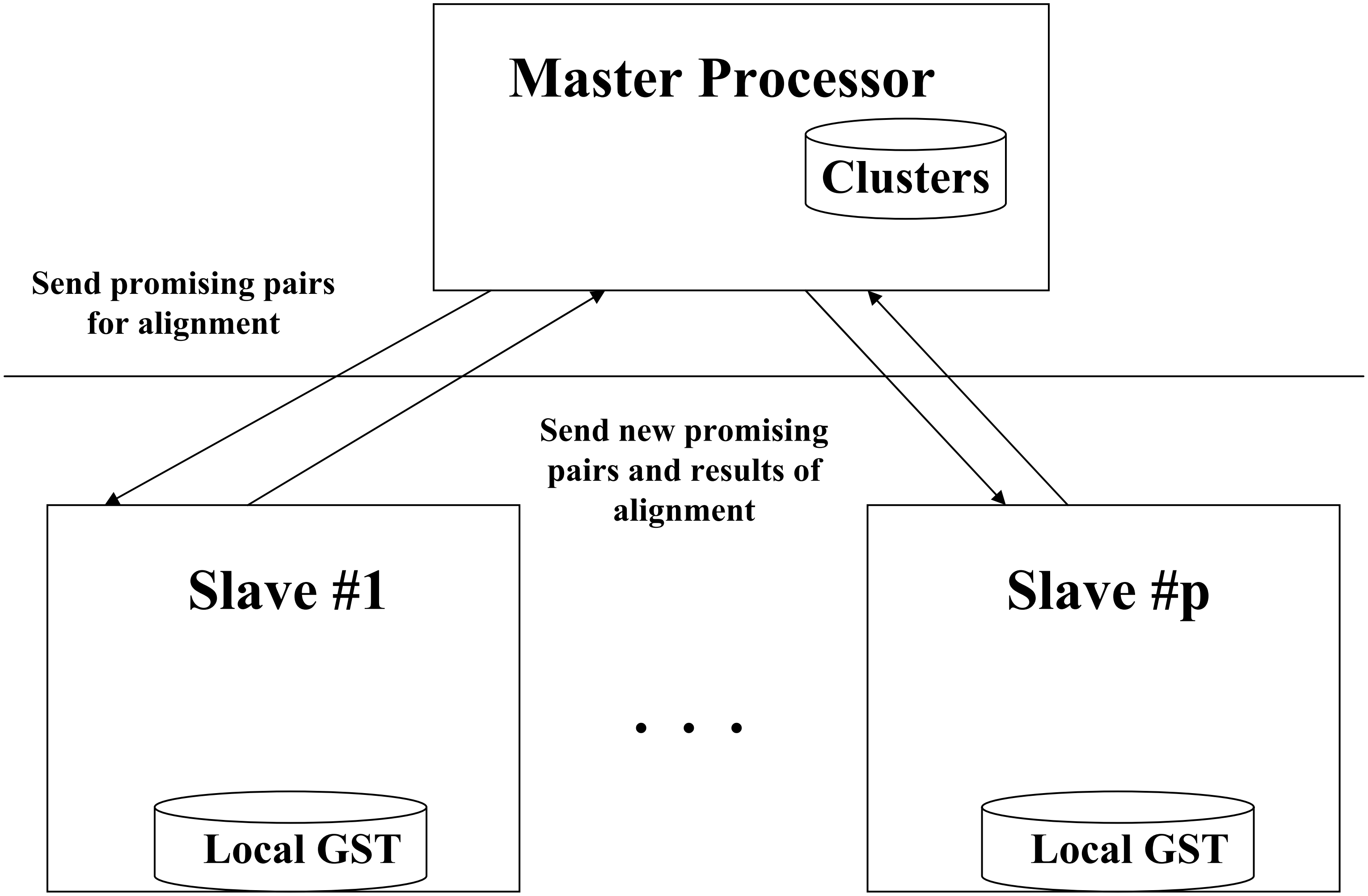
$\leq$ # of distinct maximal common substrings
(of length $\geq \psi$)

Possible Fragment Overlaps

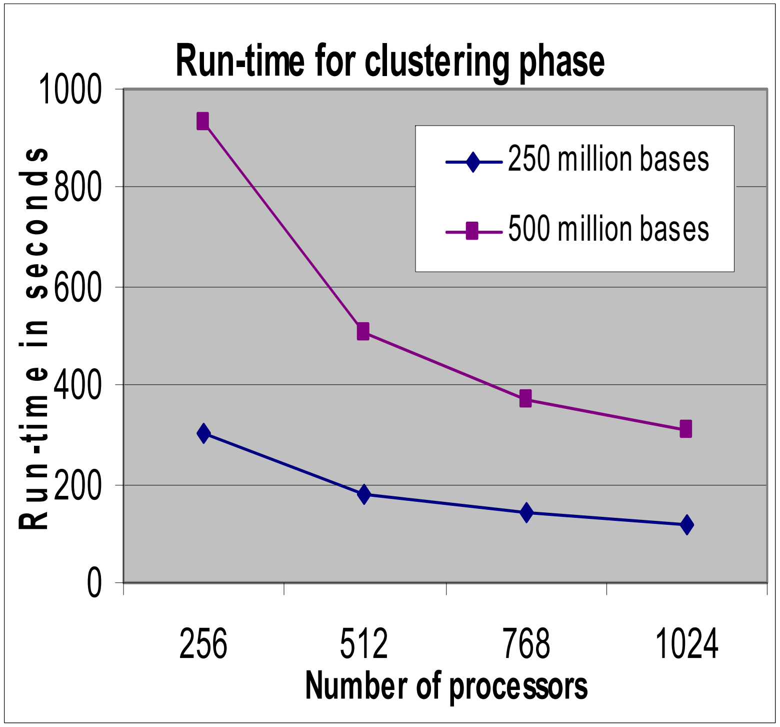


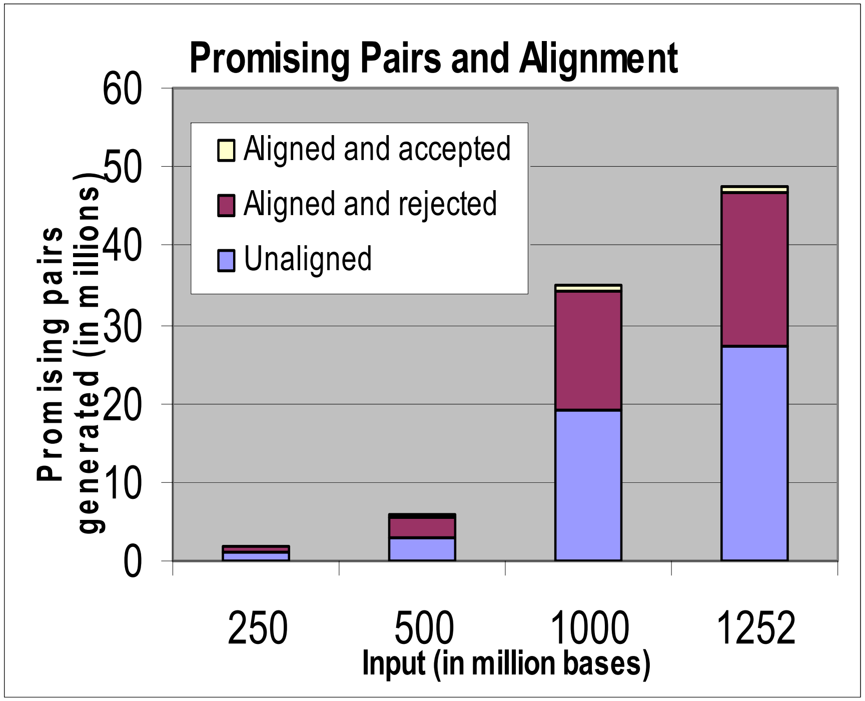
- Compute only lower and upper rectangles
- Do banded dynamic programming

Parallel Clustering Phase


Master Processor
Clusters
Send promising pairs for alignment
Send new promising pairs and results of alignment
Slave #1
Local GST
. . .
Slave #p
Local GST

Clustering Phase Performance on BlueGene/L





Overview of Assembly Pipeline

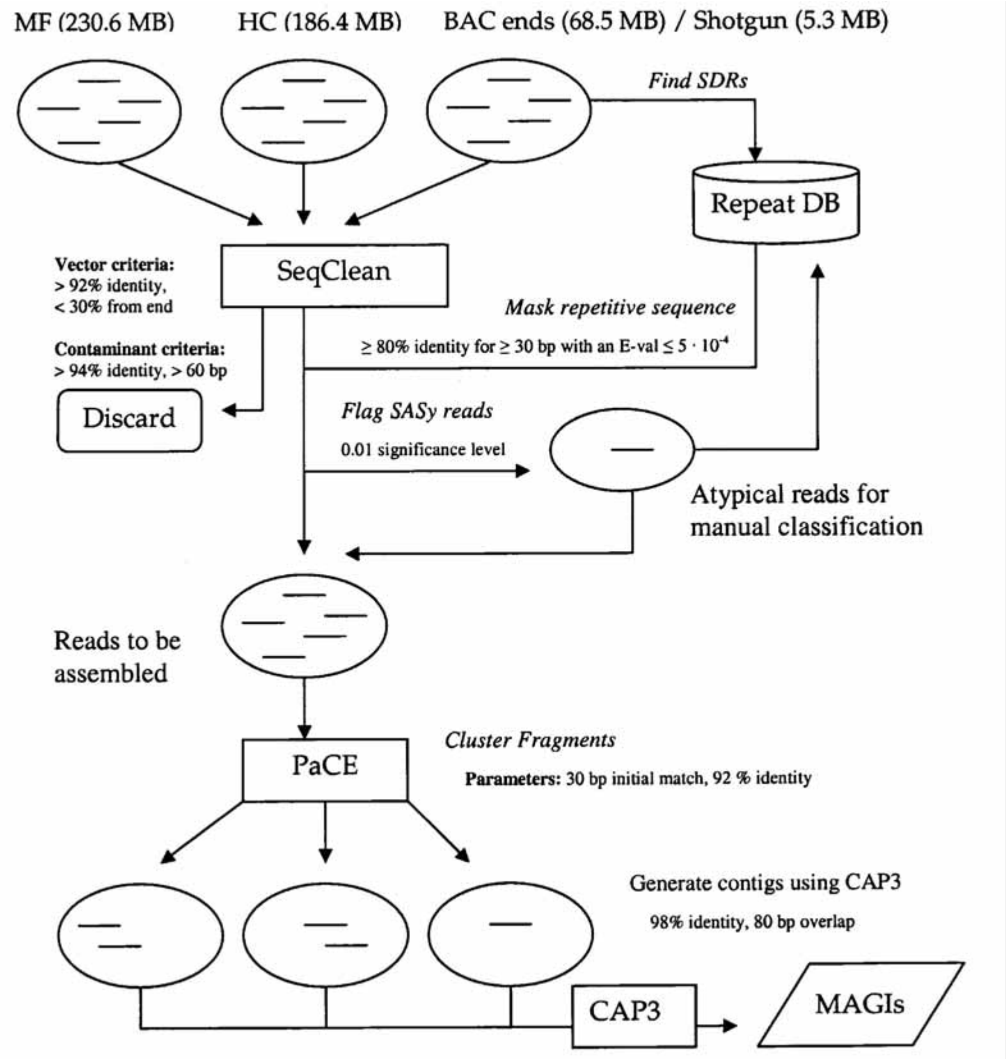
1.) Collect data

2.) Clean up data

3.) Mask repeats

4.) Cluster data

5.) Assemble smaller subproblems



Maize Assembly on BlueGene/L

Number of Input Bases (in billions)	Number of nodes	PaCE Runtimes (in minutes)		
		Tree Construction	Clustering	**Total**
1.25	1,024	13	89	**102**

Maize Assembly on BlueGene/L

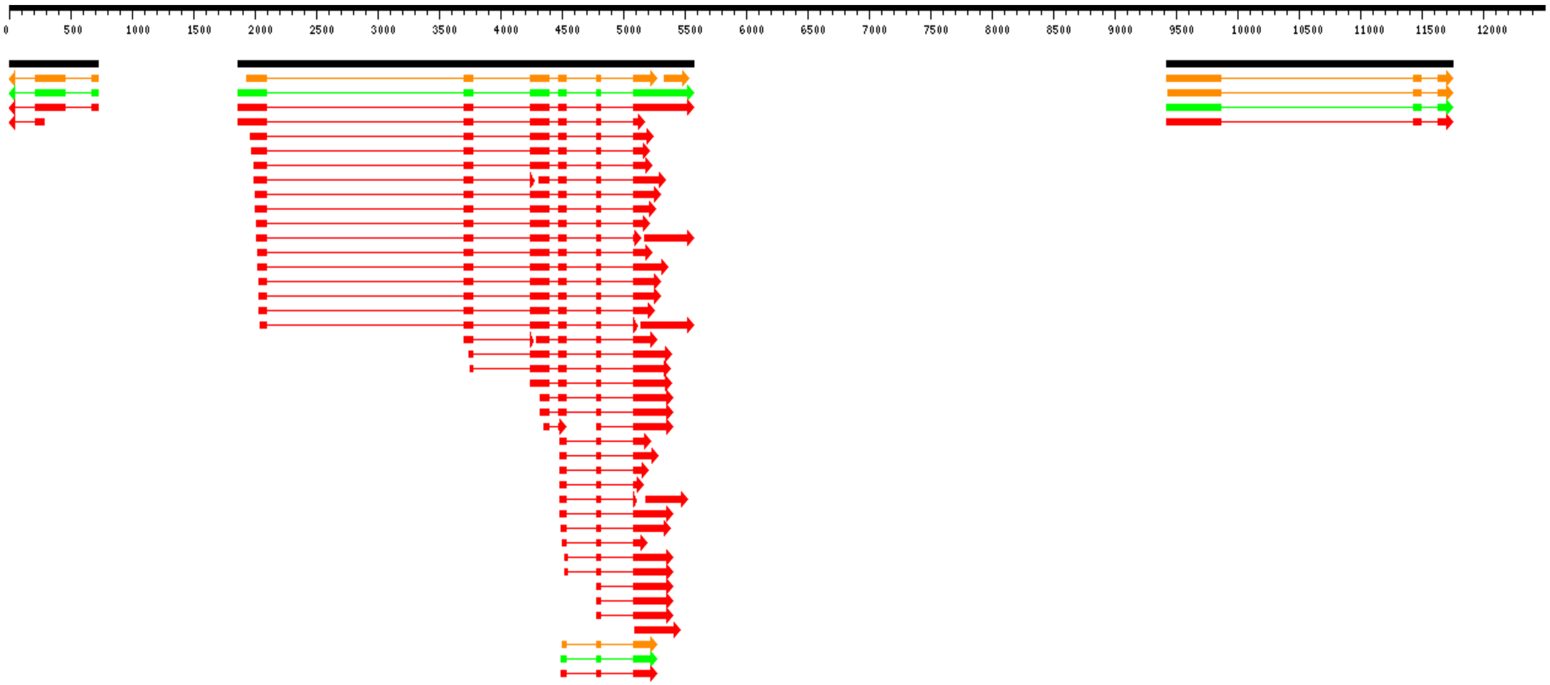
Number of Input Bases (in billions)	**Number of processors**	**PaCE Runtimes (in minutes)**		
		Tree Construction	Clustering	**Total**
0.5	8,192	1.2	11	**13**
1.15	8,192	2.3	72	**75**

Maize Assembled Genomic Islands (MAGIs)

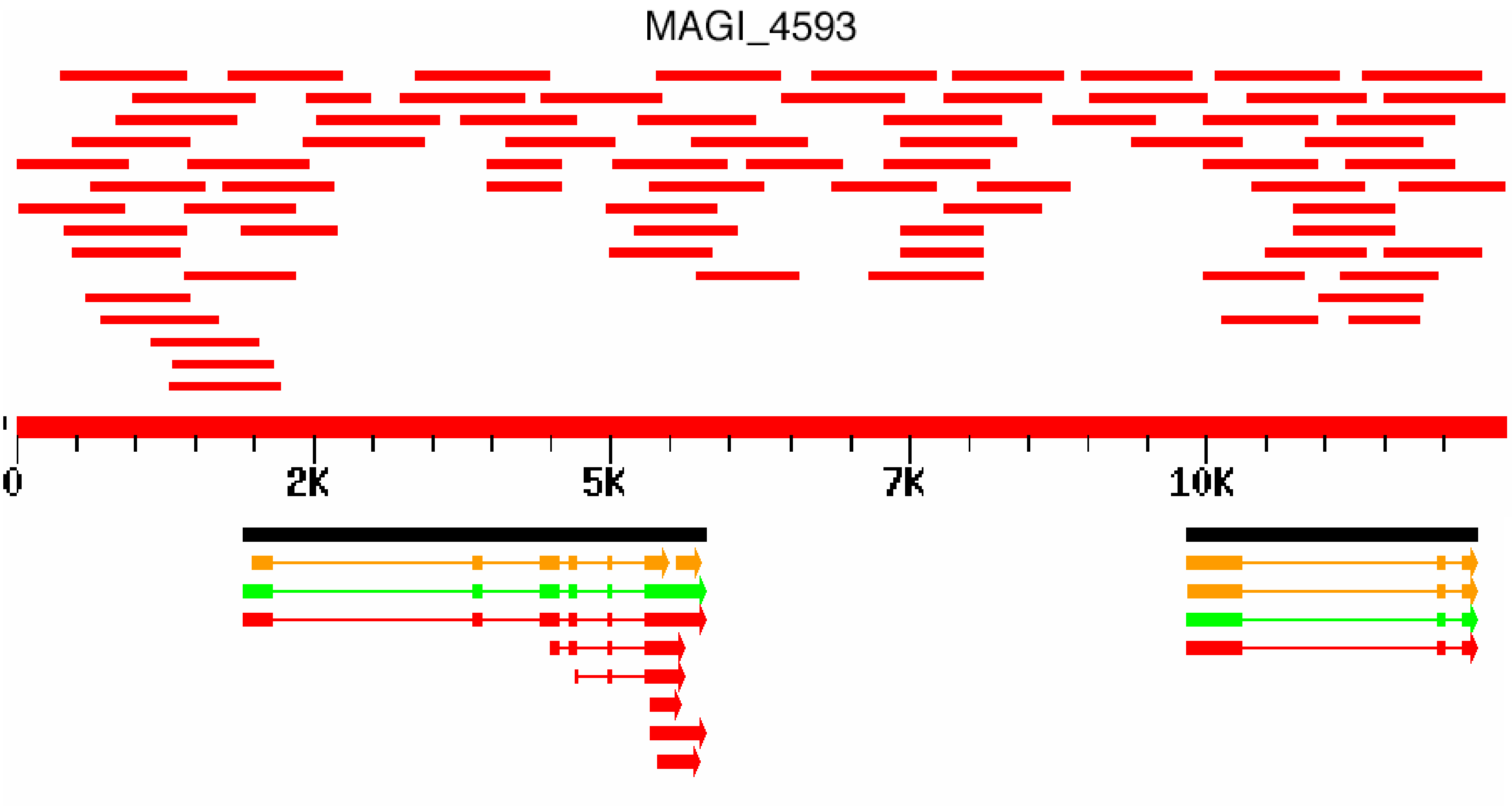
	MAGI v4.0
Input Sequences	3,202,268
Assembly Size	329.61 MB
GC Content	44.9%
Contigs	217,106
Non-repetitive Singletons	567,797
Avg contig len	1,518
Avg GSS per contig	4.78

Gene “archipelagoes”

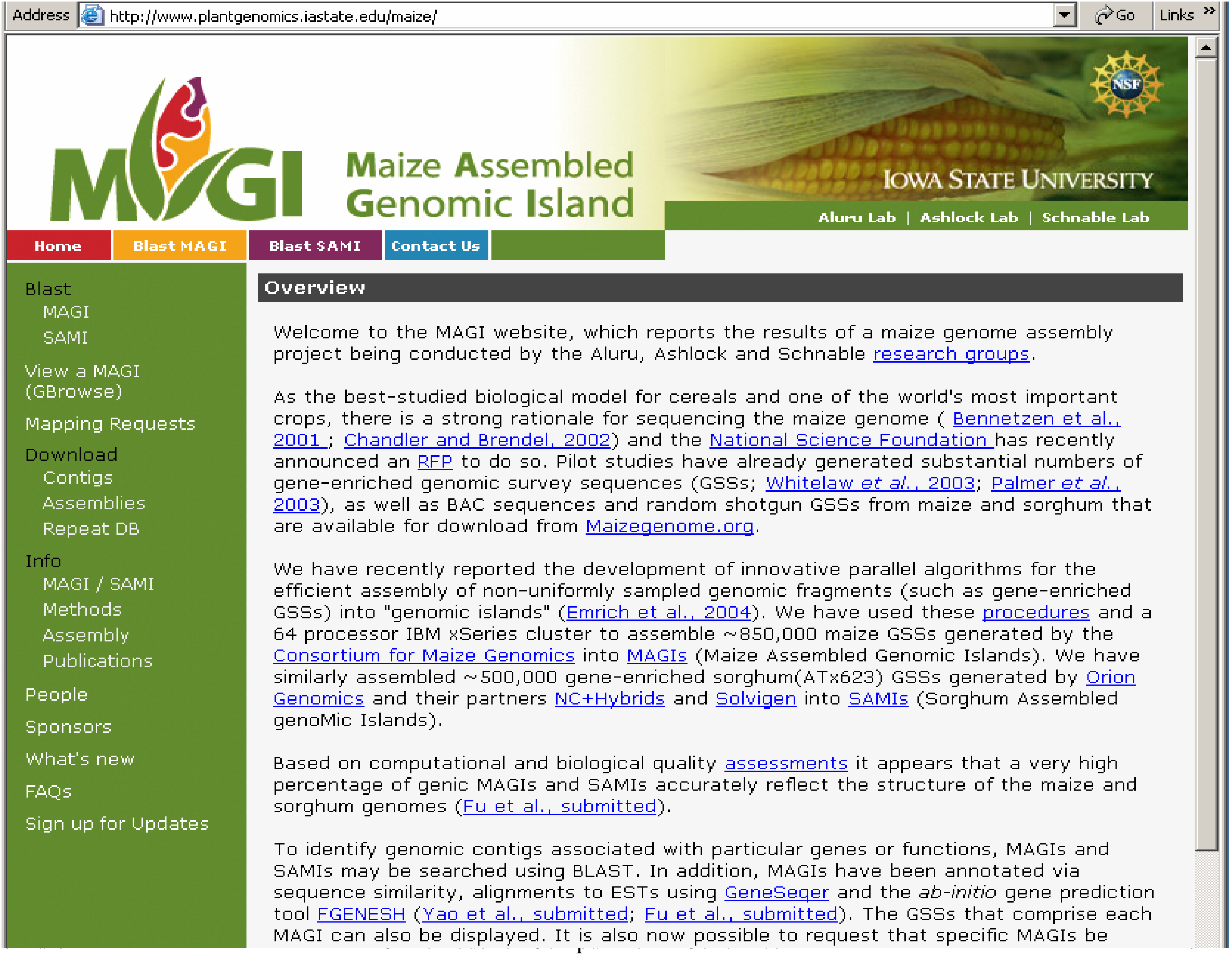
MAGI3.1_4593 (12,498 bp)



Gene "archipelagoes"



Maize assembly Portal



Overview

Welcome to the MAGI website, which reports the results of a maize genome assembly project being conducted by the Aluru, Ashlock and Schnable research groups.

As the best-studied biological model for cereals and one of the world's most important crops, there is a strong rationale for sequencing the maize genome (Bennetzen et al., 2001 ; Chandler and Brendel, 2002) and the National Science Foundation has recently announced an RFP to do so. Pilot studies have already generated substantial numbers of gene-enriched genomic survey sequences (GSSs; Whitelaw *et al.*, 2003; Palmer *et al.*, 2003), as well as BAC sequences and random shotgun GSSs from maize and sorghum that are available for download from Maizegenome.org.

We have recently reported the development of innovative parallel algorithms for the efficient assembly of non-uniformly sampled genomic fragments (such as gene-enriched GSSs) into "genomic islands" (Emrich et al., 2004). We have used these procedures and a 64 processor IBM xSeries cluster to assemble ~850,000 maize GSSs generated by the Consortium for Maize Genomics into MAGIs (Maize Assembled Genomic Islands). We have similarly assembled ~500,000 gene-enriched sorghum(ATx623) GSSs generated by Orion Genomics and their partners NC+Hybrids and Solvigen into SAMIs (Sorghum Assembled genoMic Islands).

Based on computational and biological quality assessments it appears that a very high percentage of genic MAGIs and SAMIs accurately reflect the structure of the maize and sorghum genomes (Fu et al., submitted).

To identify genomic contigs associated with particular genes or functions, MAGIs and SAMIs may be searched using BLAST. In addition, MAGIs have been annotated via sequence similarity, alignments to ESTs using GeneSeqer and the *ab-initio* gene prediction tool FGENESH (Yao et al., submitted; Fu et al., submitted). The GSSs that comprise each MAGI can also be displayed. It is also now possible to request that specific MAGIs be

More information …

- **PaCE software download**

  http://www.ece.iastate.edu/~aluru/software/PaCE

  - Over 50 academic/governmental/non-profit users from 10 countries.
  - 2 companies.

- **Maize Assembly Website**

  http://www.plantgenomics.iastate.edu/maize

  - Used by researchers from Berkeley, Cornell, Purdue, Penn State, Dupont, BASF etc.

More Information …

Publications:

- **On Maize Assembly**
  - S.J. Emrich *et al.* (2004). A strategy for assembling the maize (Zea mays L.) genome, *Bioinformatics*, 20(2):140-147.
  - A. Kalyanaraman *et al.* (2006). Assembling genomes on large scale parallel computers, *Proc. International Parallel and Distributed Processing Symposium*.
- **On PaCE**
  - A. Kalyanaraman *et al. (2003).* Space and time efficient parallel algorithms and software for EST clustering, *IEEE Transactions on Parallel and Distributed Systems*, 14(12):1209-1221.
  - A. Kalyanaraman *et al. (2003).* Efficient clustering of large EST data sets on parallel computers, *Nucleic Acids Research*, 31(11):2963:2974.
- **On Maize Genomics**
  - Y. Fu *et al.* (2005). Quality assessment of maize assembled genomic islands (MAGIs) and large-scale experimental verification of predicted novel genes. *Proceedings of the National Academy of Sciences,* 102(34):12282-12287.

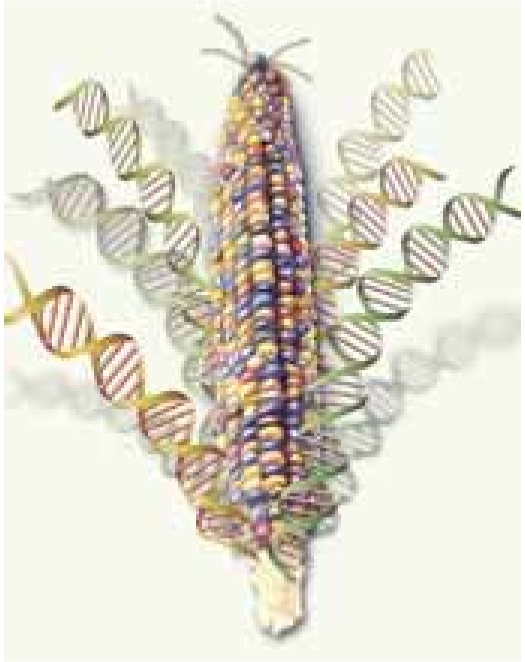
Future of Maize Genome Sequencing Project

- US $32 million project by NSF, DOE, and USDA for large-scale sequencing.
- Goal is to sequence all genes, determine their order and orientation, and anchor them to genetic/ physical maps.
- Projects started November 15, 2005.

$32 million B73 maize genome sequencing consortium

Washington University*

Iowa State University



University of Arizona

Cold Spring Harbor

Courtesy of the NSF

Another Application: Mouse EST Clustering

Mouse EST clustering

- Input:
  - A random subset of 56,470 UniGene clusters downloaded in March 2006
  - 3.78 million total entries including ESTs and full-length cDNAs

- Output:
  - 60,862 clusters with more than one sequence
  - Average cluster = 55; Largest = 807,671
  - 83% of clusters are composed of a single UniGene

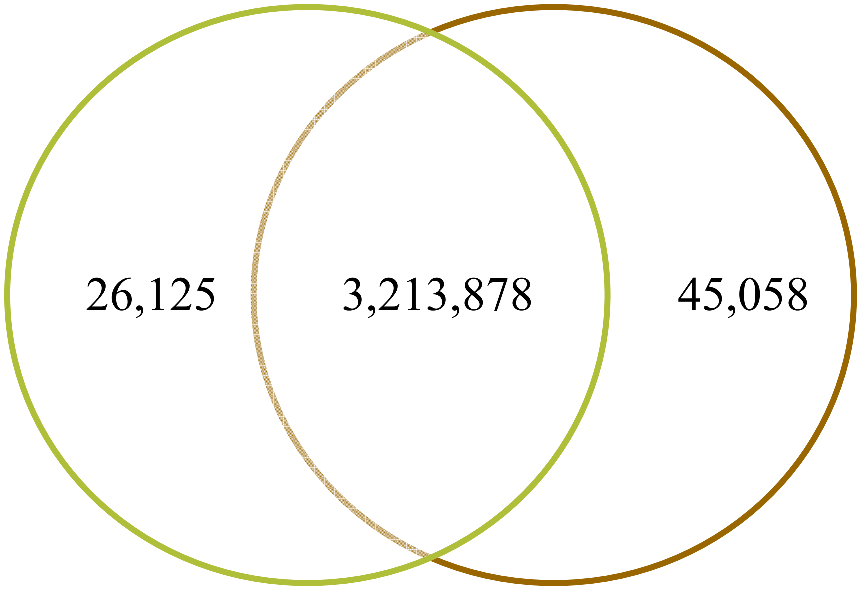
Validation

- Single-linkage clustering performs at most *n* merges.
- When comparing to UniGene, one measure of accuracy is the number of additional or missed merges performed.
- Ignoring clusters of size 1, our data suggest that over 98% of the links in UniGene were correctly determined by PaCE.

Clustering accuracy

PaCE clustering decisions

UniGene clustering decisions



26,125

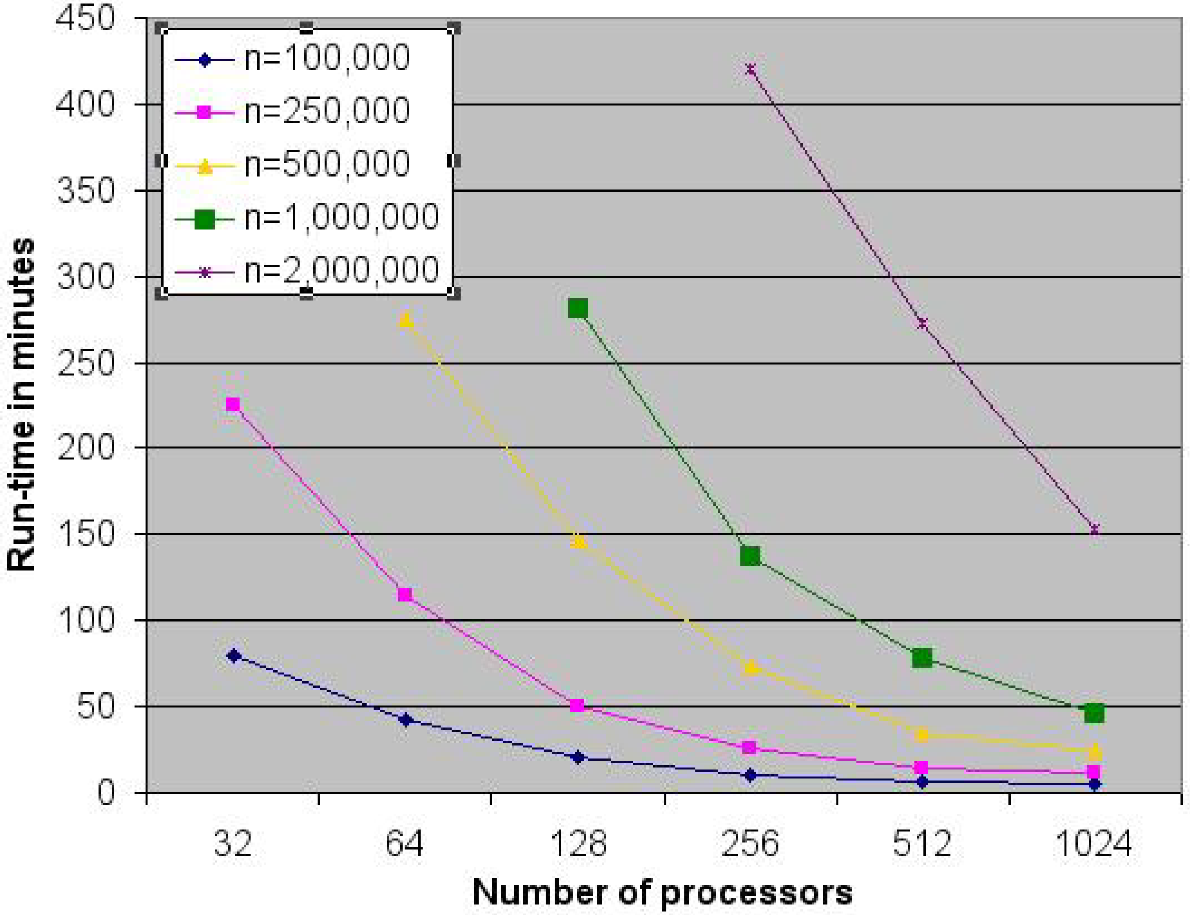
3,213,878

45,058

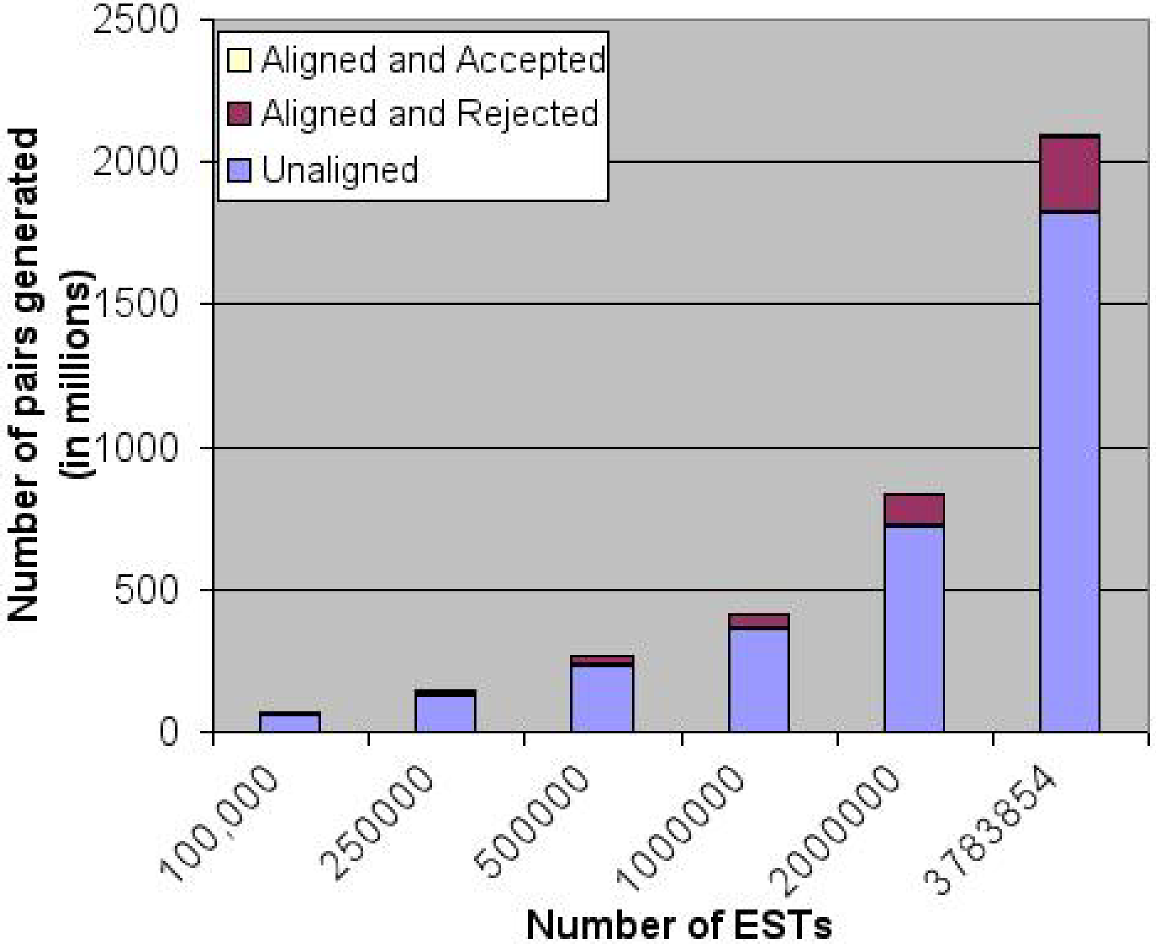
False positives

False negatives

Run-time Scaling: Mouse EST Clustering



PaCE: Promising Pairs Statistics



Selected Bibliography on EST Clustering and Genome Assembly

- P. Green (1994). Phrap - the assembler. http//www.phrap.org.
- J.D. Kececioglu and E.W. Myers (1995). Combinatorial algorithms for DNA sequence assembly, *Algorithmica*, 13(1):7-51.
- G. Sutton *et al.* (1995). TIGR assembler: A new tool for assembling large shotgun sequencing projects, *Genome Science and Technology*, 1:9-19.
- X. Huang and A. Madan (1999). CAP3: A DNA sequence assembly program, *Genome Research*, 9(9):868-877.
- F. Liang *et al.* (2000). An optimized protocol for analysis of EST sequences, *Nucleic Acids Research*, 28(18):3657-3665.
- E.W. Myers *et al.* (2000). A whole-genome assembly of *drosophila*, *Science*, 287(5461):2196-2204.
- P.A. Pevzner *et al.* (2001). An Eulerian path approach to DNA fragment assembly, *Proceedings of the National Academy of Sciences USA*, 98(17):9748-9753.
- A. Christoffels *et al.* (2001). STACK: Sequence Tag Alignment and Consensus Knowledgebase, *Nucleic Acids Research*, 29(1):234-238.
- K. Pedretti (2001). Accurate, parallel clustering of EST (gene) sequences, Master's Thesis, University of Iowa.
- S. Batzoglou *et al.* (2002). ARACHNE: A whole-genome shotgun assembler, *Genome Research*, 12(1):177-189.

Selected Bibliography on EST Clustering and Genome Assembly …

- J. Pontius *et al.* (2003). UniGene: a unified view of the transcriptome. The NCBI Handbook.
- G. Pertea *et al.* (2003). TIGR Gene Indices clustering tool (TGICL): A software system for fast clustering of large EST datasets, *Bioinformatics*, 19(5):651-652.
- K. Malde *et al.* (2003). Fast sequence clustering using a suffix array algorithm, *Bioinformatics*, 19(10):1221-1226.
- J.C. Mullikin and Z. Ning (2003). The phusion assembler, *Genome Research*, 13(1):81-90.
- X. Huang *et al.* (2003). PCAP: A whole-genome assembly program, *Genome Research*, 13(9):2164-2170.
- A. Kalyanaraman *et al. (2003)*. Space and time efficient parallel algorithms and software for EST clustering, *IEEE Transactions on Parallel and Distributed Systems*, 14(12):1209-1221.
- A. Kalyanaraman *et al. (2003)*. Efficient clustering of large EST data sets on parallel computers, *Nucleic Acids Research*, 31(11):2963:2974.
- P.Havlak *et al.* (2004). The atlas genome assembly system, *Genome Research*, 14(4):721-732.
- A. Kalyanaraman *et al.* (2006). Assembling genomes on large scale parallel computers, *Proc. International Parallel and Distributed Processing Symposium*.

Part III:
HPC for Phylogenetics

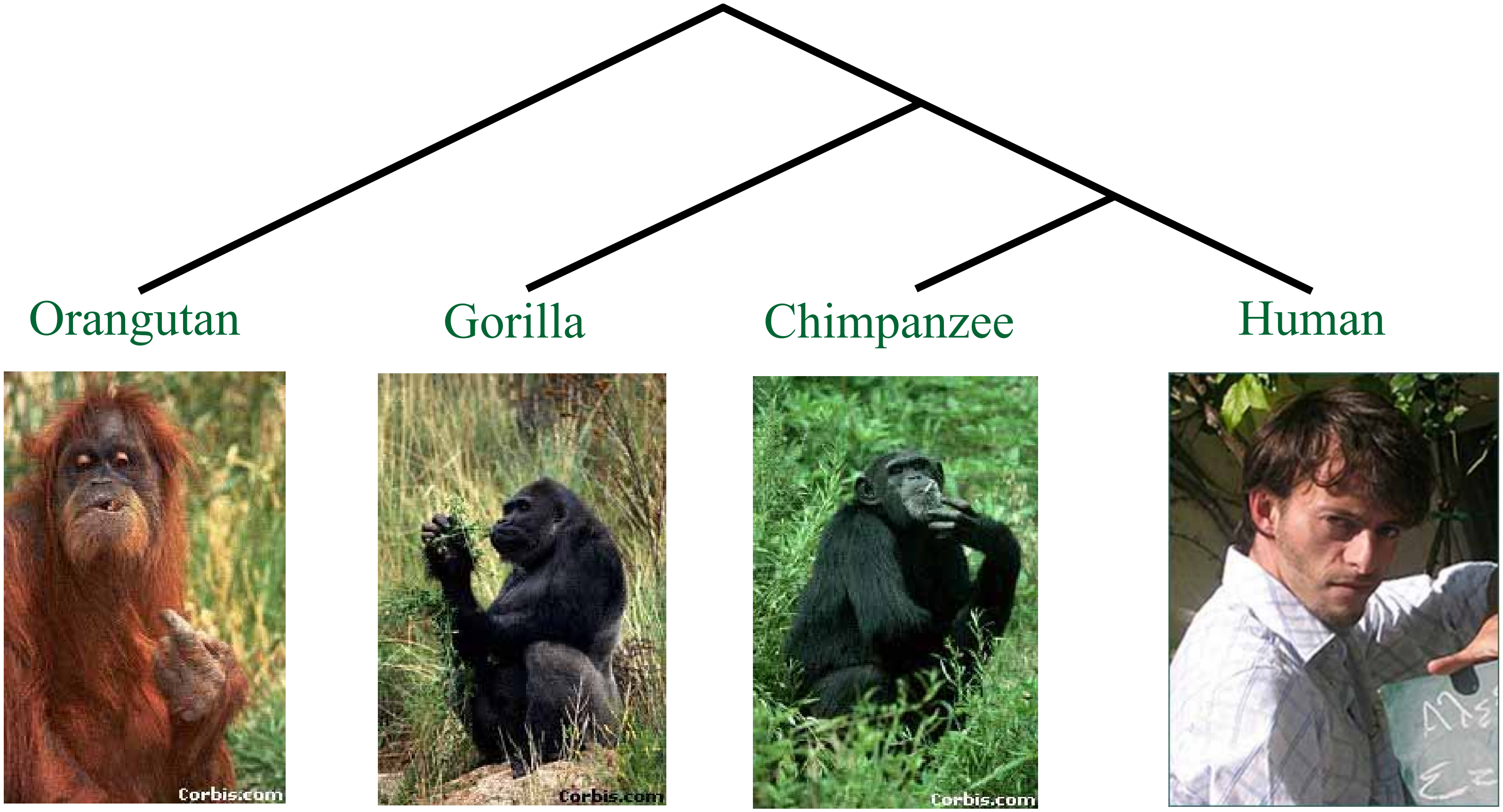
CIPRES

Cyber Infrastructure for Phylogenetic Research

- **www.phylo.org**
- A community project, funded by an $11.6M NSF Information Technology Research grant

- Georgia Institute of Technology: D.A. Bader
- University of New Mexico: B.M.E. Moret, T. Williams
- UC San Diego: F. Berman, P. Bourne
- Yale: M. Donoghue
- U Texas-Austin: T. Warnow, D.M. Hillis, W. Hunt, D. Miranker, L. Meyers
- U Pennsylvania: J. Kim
- U Connecticut: P. Lewis
- U Arizona: D. Maddison, W. Maddison
- UC Berkeley: B. Mischler, E. Myers, S. Rao, S. Russell
- Florida State U: D. Swofford
- American Museum of Natural History: W. Wheeler

Phylogeny



Orangutan

Gorilla

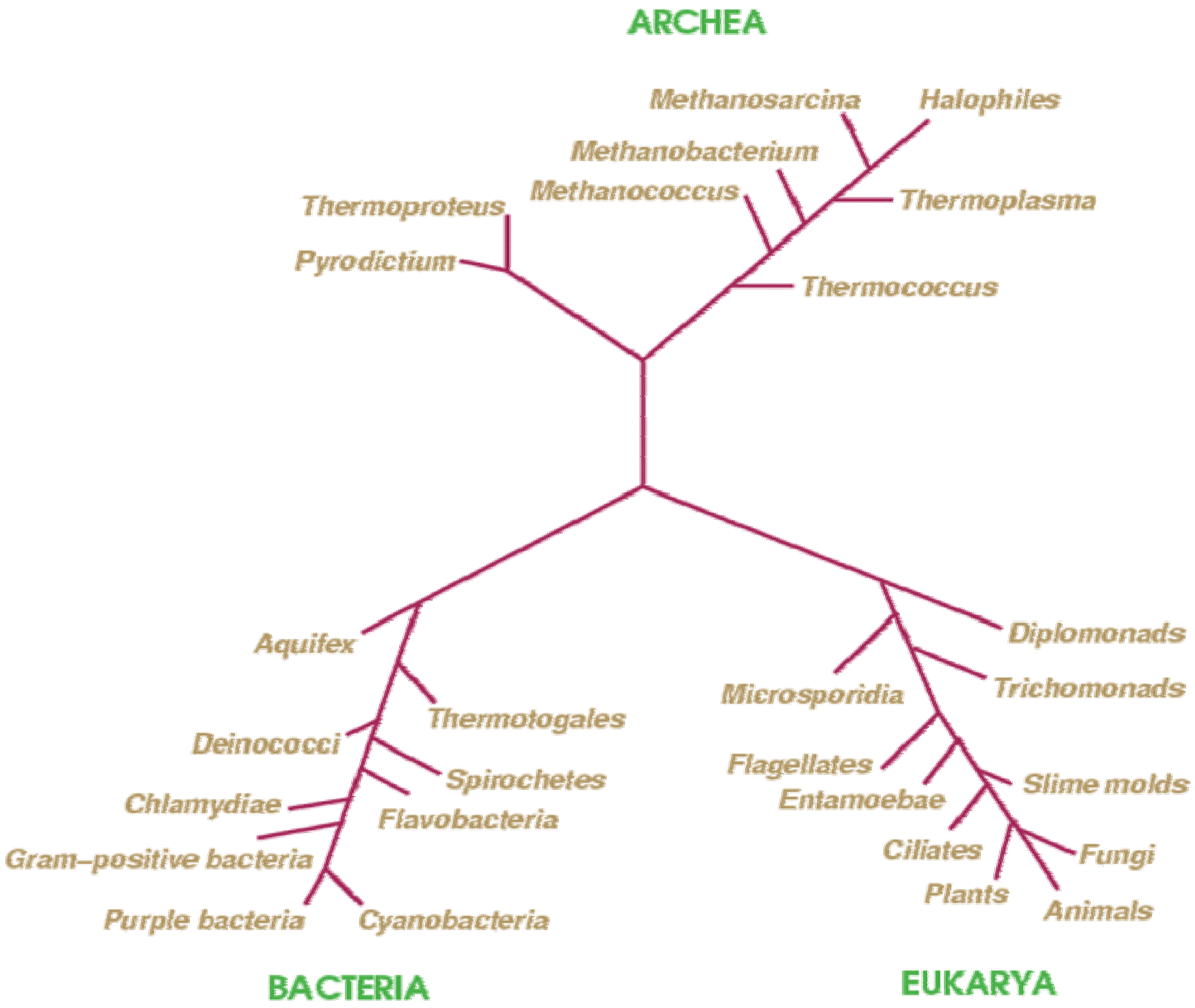
Chimpanzee

Human

Phylogeny informs everything in biology

- It relates organisms and genes.
- It helps us understand and predict:
  - interactions between genes (genetic networks)
  - functions of genes
  - relationship between genotype and phenotype
  - drug and vaccine development
  - origins and spread of disease
  - origins and migrations of humans

The Tree of Life: The Ultimate Phylogeny



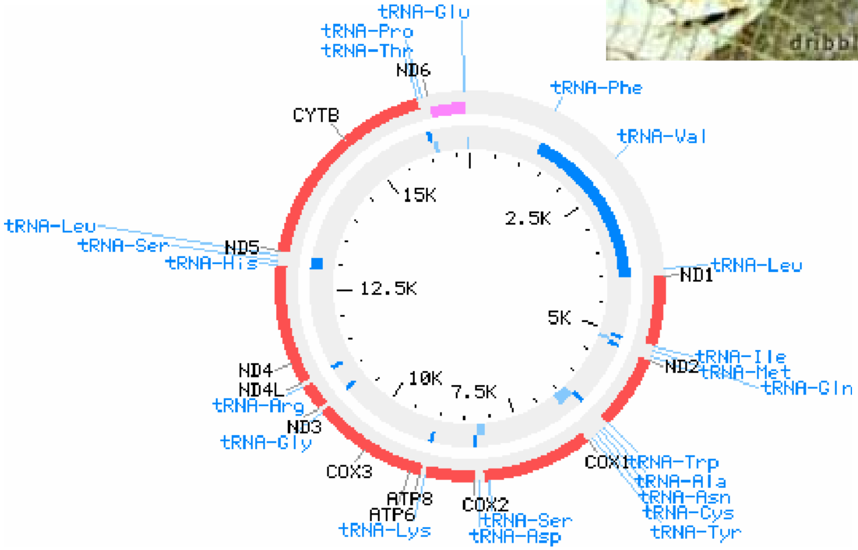
- CIPRES aims to establish the cyber infrastructure (platform, software, database) required to attempt a reconstruction of the Tree of Life

(10-100M organisms)

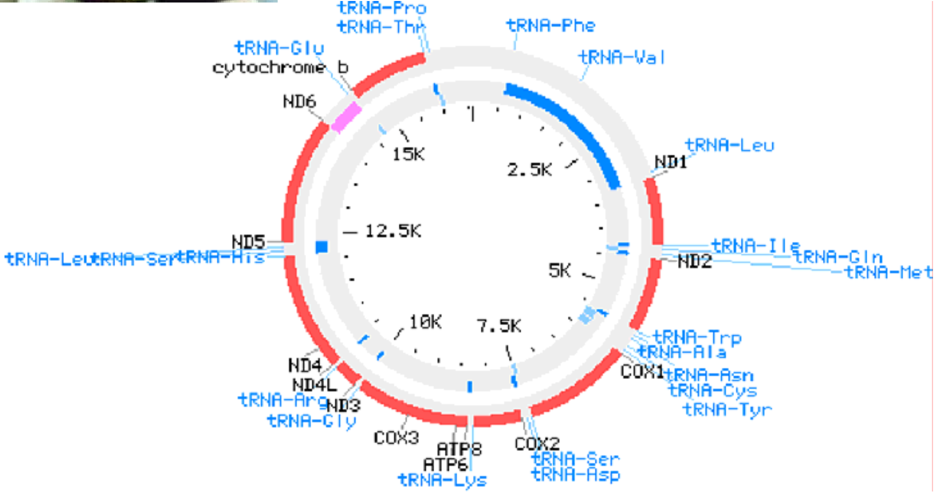
Comparative Genomics

Chicken

Human



NCBI accession #NC_001323

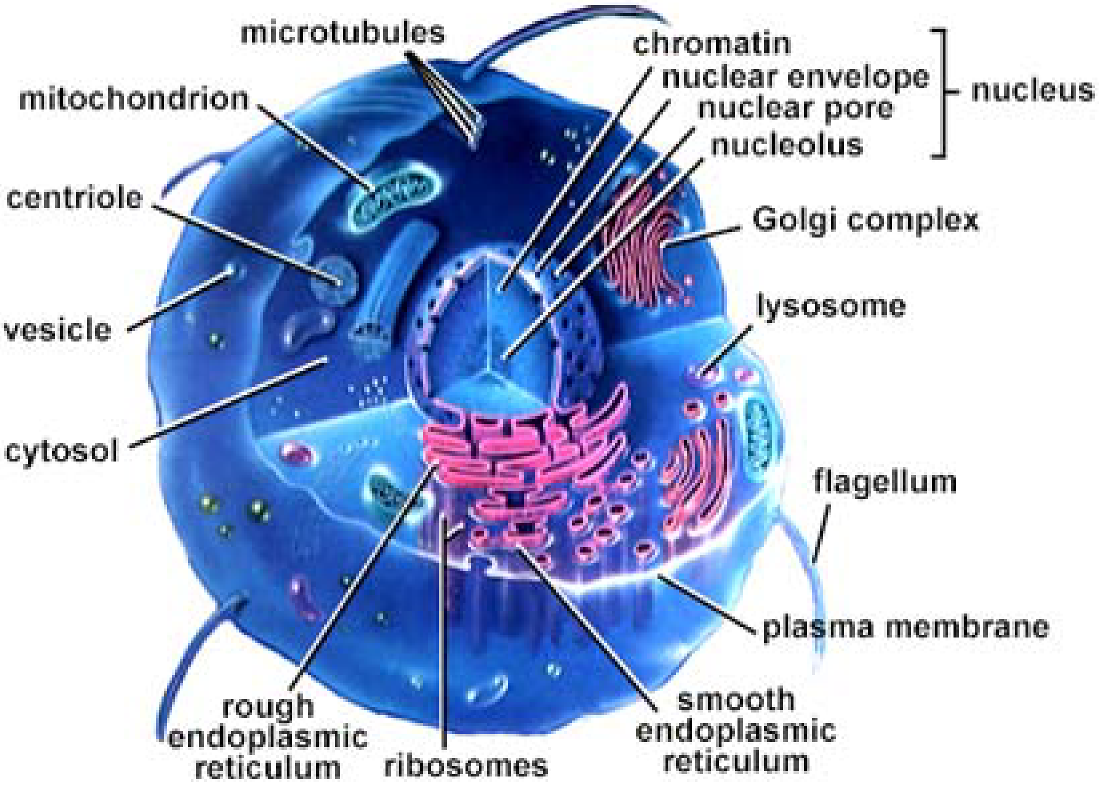


NCBI accession #NC_001807

Phylogenetic Trees

- Represents evolutionary relationships
- Leaves of the tree contain known taxa
- Internal vertices represent ancestral species
- Edges represent evolutionary events

Eukaryotic Cell

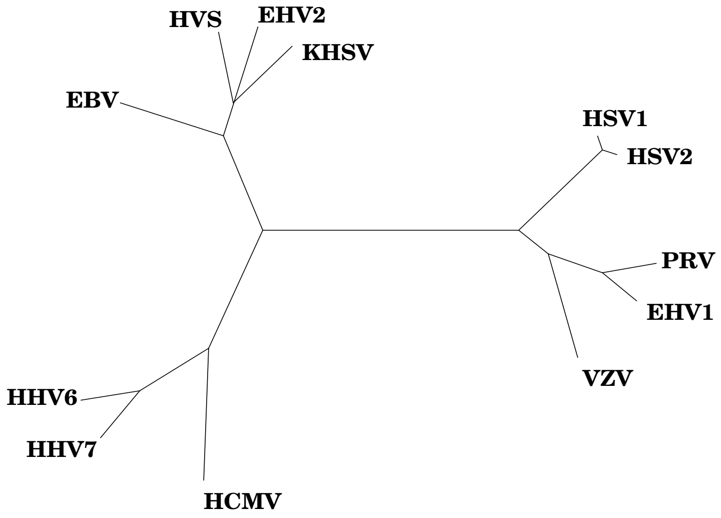


Types of Phylogenies

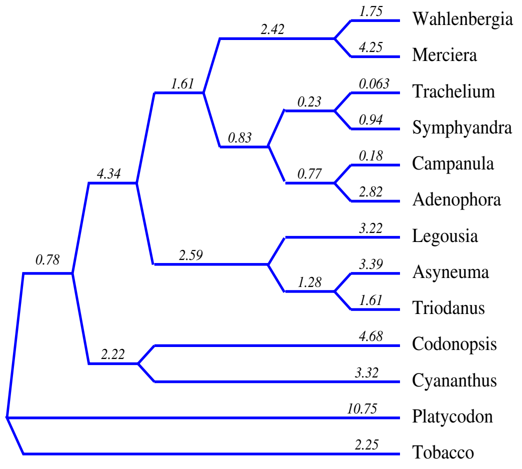
- Relationships between taxa
  - Species Trees
  - Gene Trees
- Data
  - Morphological
    - Tree of Life Web (Maddison/Maddison): http://tolweb.org/
  - Nuclear Genome
  - Organelle Genome

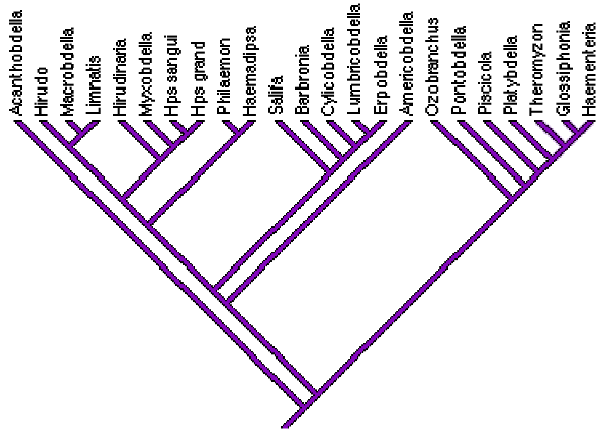
Example Phylogenies

Some herpesvirus known to affect humans



Campanulaceae (Bluebell Flowers)





Leeches

Commercial Aspects of Phylogeny Reconstruction

- Identification of microorganisms
  - public health entomology
  - sequence motifs for groups are patented
  - example: differentiating tuberculosis strains
- Dynamics of microbial communities
  - pesticide exposure: identify and quantify microbes in soil
- Vaccine development
  - variants of a cell wall or protein coat component
  - porcine reproductive and respiratory syndrome virus isolates from US and Europe were separate populations
  - HIV studied through DNA markers
- Biochemical pathways
  - antibacterials and herbicides
    - Glyphosate (Roundup™, Rodeo ™, and Pondmaster ™): first herbicide targeted at a pathway not present in mammals
  - phylogenetic distribution of a pathway is studied by the pharmaceutical industry before a drug is developed
- Pharmaceutical industry
  - predicting the natural ligands for cell surface receptors which are potential drug targets
  - a single family, G protein coupled receptors (GPCRs), contains 40% of the targets of most pharm. companies

Techniques

- Maximum parsimony
  - Occam’s razor: simplest explanation for evolution, minimizes the sum of the number of evolutionary events along the tree branches
- Maximum likelihood
  - Statistical methods that use an evolutionary model such as the transition/transversion rate ratio for the nuclear genome

Exploiting data about gene content and gene order

- has proved extremely challenging from a computational perspective
  - tasks that can easily be carried out in linear time for DNA data have required entirely new theories (such as the computation of inversion distance) or appear to be NP-hard

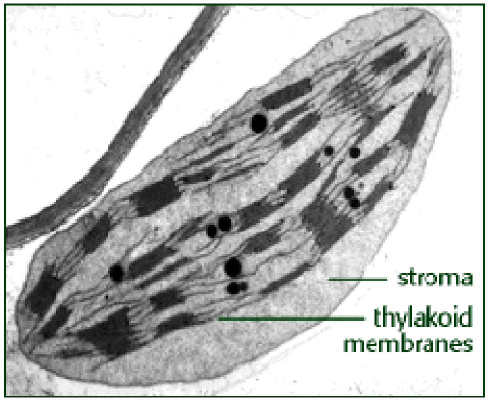
- The focus has thus been on simple genomes, preferably genomes
  - consisting of a single chromosome, and
  - where evolution can reasonably be assumed to have been driven mostly through gene order changes.

Cell Organelles

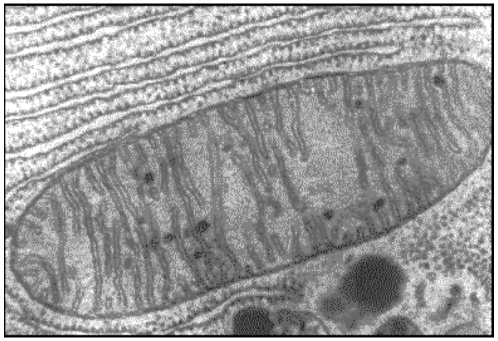
- Chloroplasts and mitochondria have such genomes: around 120 genes for the chloroplasts of higher plants and typically 37 genes for the mitochondria of multicellular animals, in both cases packed onto a single chromosome.

- The gene content of these genomes is fairly constant across a wide phylogenetic range, differences are mostly in the ordering of the genes.

Chloroplast



Mitochondria



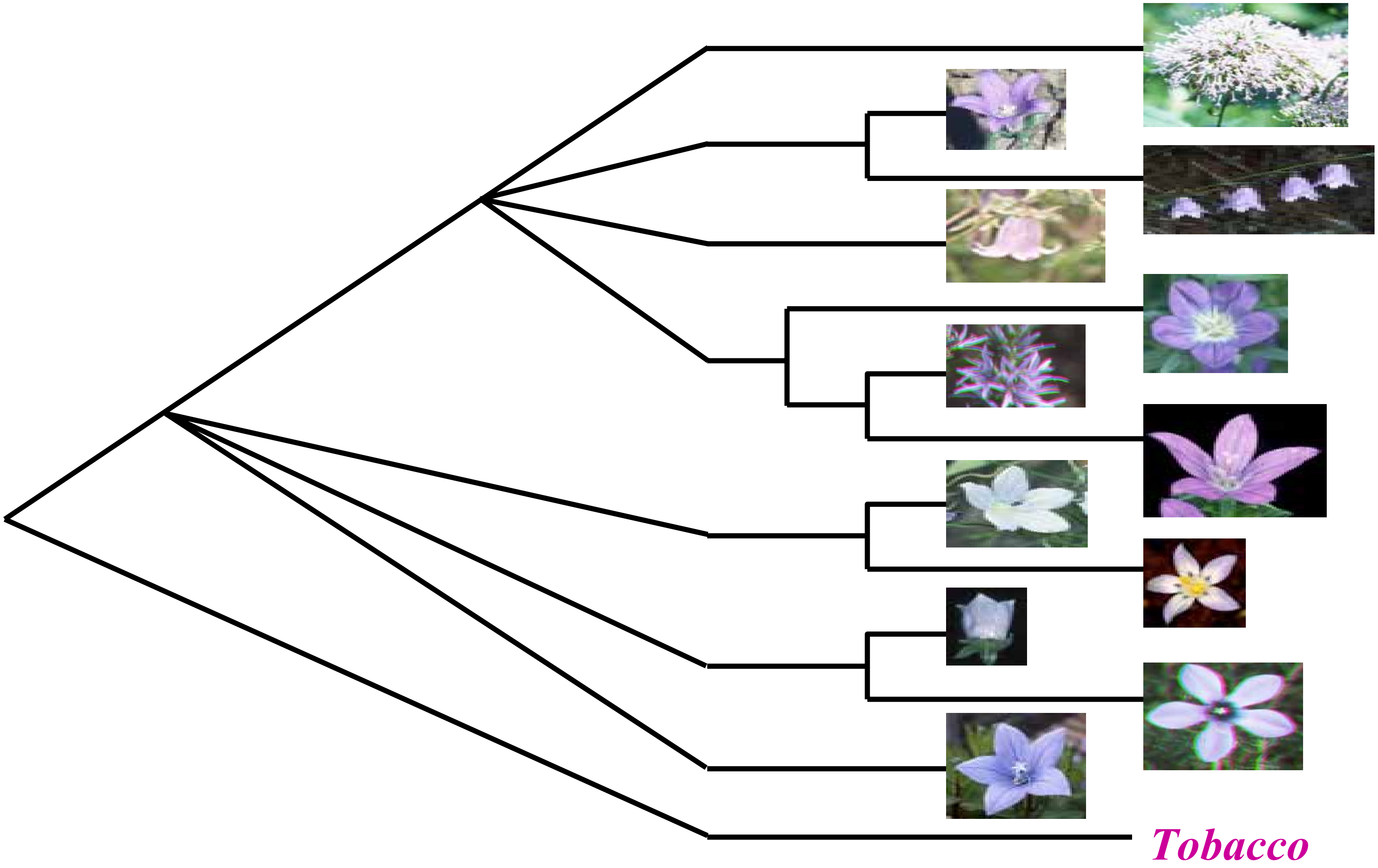
GRAPPA: Genome Rearrangements Analysis

- Genome Rearrangements Analysis under Parsimony and other Phylogenetic Algorithms
  - http://www.cc.gatech.edu/~bader/code.html
  - Freely-available, open-source, GNU GPL
  - already used by other computational phylogeny groups, Caprara, Pevzner, LANL, FBI, Smithsonian Institute, Aventis, GlaxoSmithKline, PharmCos.
- Gene-order Phylogeny Reconstruction
  - Breakpoint Median
  - Inversion Median
- over one-billion fold speedup from previous codes
- Parallelism scales linearly with the number of processors

Using GRAPPA to solve *Campanulaceae* Phylogeny

- On the 512-processor IBM Linux cluster, we ran the full analysis (all 14 billion trees) in under 1.5 hours – a 1,000,000-fold speedup (and using true inversion distance) compared with the best previous code BPanalysis
  - 256 IBM Netfinity 4500R nodes of dual 733MHz Intel Pentium III processors, interconnected with Myrinet 2000
- Current release of GRAPPA (v. 1.6) now takes minutes to solve the same problem on several processors

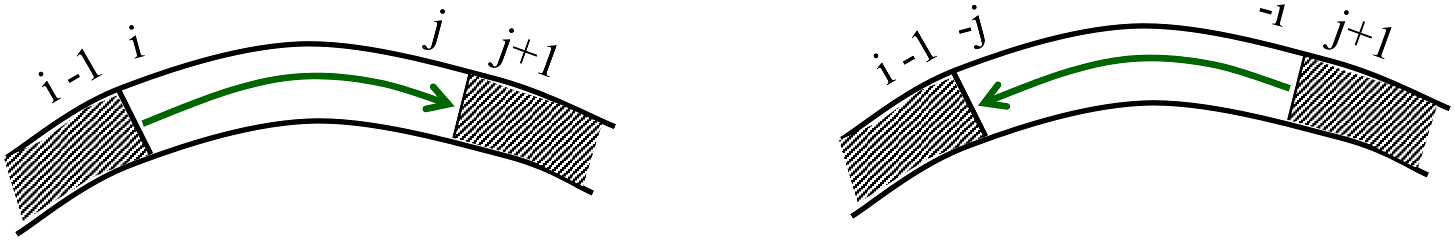
Campanulaceae



- Bob Jansen, UT-Austin;
- Linda Raubeson, Central Washington U

Gene Order Phylogeny

- Many organelles appear to evolve mostly through processes that simply rearrange gene ordering (inversion, transposition) and perhaps alter gene content (duplication, loss).
- Chloroplast have a single, typically circular, chromosome and appear to evolve mostly through inversion:



The sequence of genes i, i+1, …, j is inverted and every gene is flipped.

Breakpoint Analysis (Sankoff & Blanchette 1998)

$(2n-5)!! = (2n-5)\,(2n-7) \ldots \bullet 5 \bullet 3$ trees

unknown iterative heuristic

NP-hard

- For each tree topology do
  - somehow assign initial genomes to the internal nodes
  - repeat
    - for each internal node do
      - compute a new genome that minimizes the distances to its three neighbors
      - replace old genome by new if distance is reduced
  - until no change

Sankoff & Blanchette implemented this in a C++ package
This is NP-hard, even for just three taxa!

Algorithm Engineering Works!

- We reimplemented everything –
  - the original code is too slow and not as flexible as we wanted.
- Our main dataset is a collection of chloroplast data from the flowering plant family Campanulaceae (bluebells):
  - 13 genomes of 105 gene segments each
- On a Pentium III Linux PC:
  - BPAnalysis processes 10-12 trees/minute
  - Our implementation processes over 50,000 trees/minute
- Speedup ratio is over 5,000!!
- On synthetic datasets, we see speedups from 300 to over 50,000…

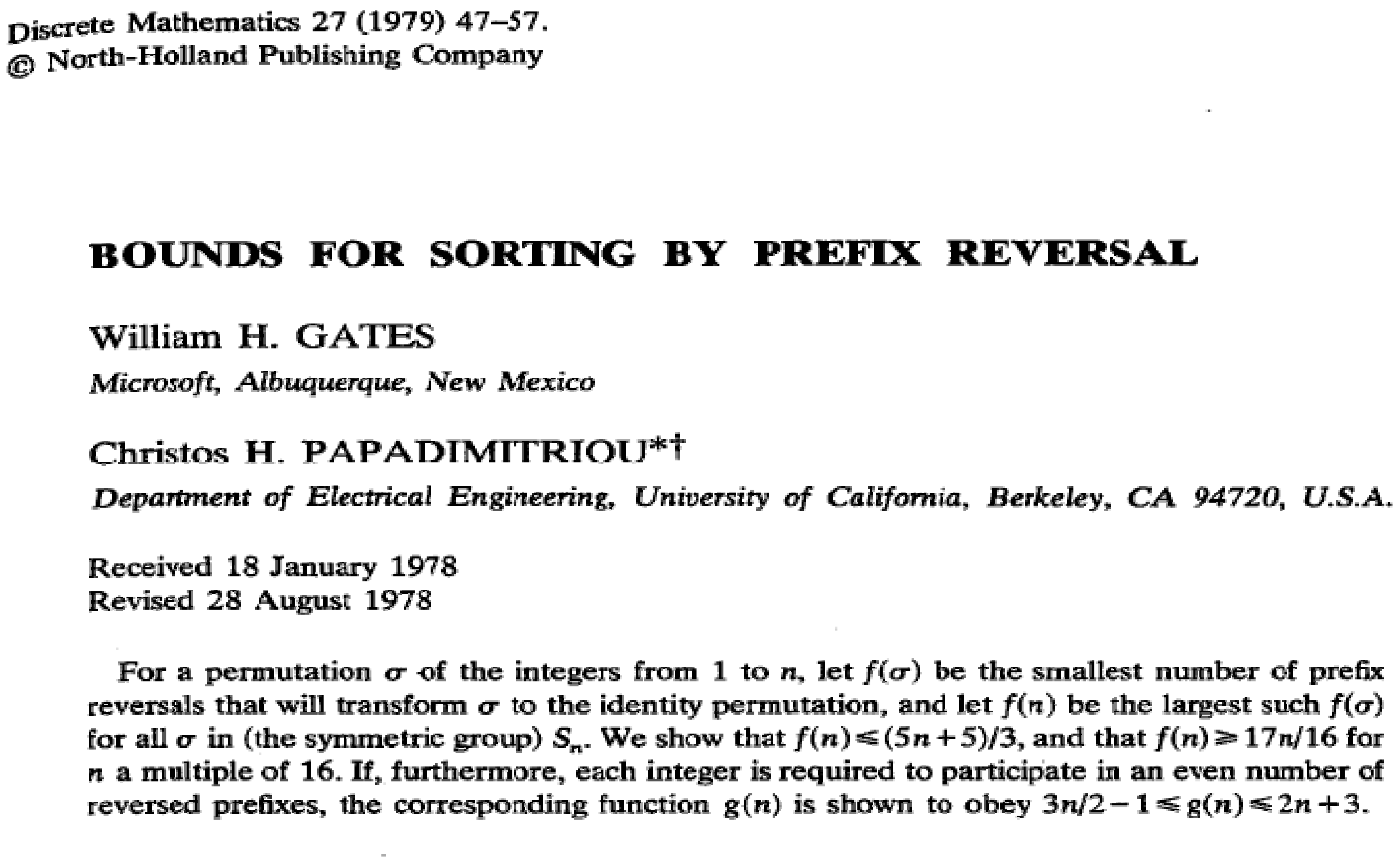
High-Performance Computing Techniques

- Availability of hundreds of powerful processors
- Standard parallel programming interfaces
  - Message passing interface (MPI)
  - OpenMP or POSIX threads
- Algorithmic libraries for SMP clusters
  - SIMPLE
- Goal: make efficient use of parallelism for
  - exploring candidate tree topologies
  - sharing of improved bounds

Parallelization of the Phylogeny Algorithm

- Enumerating tree topologies is pleasantly parallel and allows multiple processors to independently search the tree space with little or no overhead
- Improved bounds can be broadcast to other processors without interrupting work
- Load is evenly balanced when trees are cyclically assigned (e.g. in a round-robin fashion) to the processors
- Linear speedup

How Bill Gates's Only Journal Paper Relates to Computational Biology



Discrete Mathematics 27 (1979) 47–57.
© North-Holland Publishing Company

BOUNDS FOR SORTING BY PREFIX REVERSAL

William H. GATES
Microsoft, Albuquerque, New Mexico

Christos H. PAPADIMITRIOU*†
Department of Electrical Engineering, University of California, Berkeley, CA 94720, U.S.A.

Received 18 January 1978
Revised 28 August 1978

For a permutation σ of the integers from 1 to n, let $f(\sigma)$ be the smallest number of prefix reversals that will transform σ to the identity permutation, and let $f(n)$ be the largest such $f(\sigma)$ for all σ in (the symmetric group) S_n. We show that $f(n) \leq (5n+5)/3$, and that $f(n) \geq 17n/16$ for n a multiple of 16. If, furthermore, each integer is required to participate in an even number of reversed prefixes, the corresponding function $g(n)$ is shown to obey $3n/2 - 1 \leq g(n) \leq 2n + 3$.

Inversion Distance (Hannenhalli-Pevzner Theory)

- NP-hard for unsigned permutations [Caprara 97]

- Polynomial for signed permutations [Hannenhalli & Pevzner 95]
  - Compute combinatorial terms from the cycle graph
  - $d = b - c + h + f$ [Bafna & Pevzner 93, Setubal & Meidanis 97]
    - b = number of breakpoints
    - c = number of cycles
    - h = number of hurdles
    - f = (0/1) Is there a fortress?
  - $O(n\ \alpha(n))$ time, [Berman and Hannenhalli 96]
    - where $\alpha(n)$ is the inverse Ackerman function (practically a constant no greater than 4)

- New result: O(n) inversion distance, [Bader, Moret, Yan 01]
  - faster and simpler algorithm, both in theory and in practice

- High Impact work: already cited over 125 times!

GRAPPA Remarks

- Our reimplementation led to numerous extensions as well as to new theoretical results
  - GRAPPA has been extended to inversion phylogeny, with linear-time algorithms for inversion distance and a new approach to exact inversion median-of-three.
- High-performance implementations enable:
  - better approximations for difficult problems (MP, ML)
  - true optimization for larger instances
  - realistic data exploration (e.g., testing evolutionary scenarios, assessing answers obtained through other means, etc.)
- Our analysis of the Campanulaceae dataset confirmed the conjecture of Robert Jansen et al. – that inversion is the principal process of genome evolution in cpDNA for this group.

Reconstruction Software: single chromosome, organellar size (< 200 genes)

- 1998 BP Analysis
  - Sankoff
  - 8 taxa → 1 day
  - 13 taxa → 250 years
- 2000 GRAPPA
  - 13 taxa → 1 day (512 proc. cluster)
  - (200 serial, 100,000 parallel)
- 2001 GRAPPA
  - 13 taxa → 1 hour (laptop)
  - (2,000,000 serial)
  - 20 taxa → 3 million years
- 2003 DCM-GRAPPA
  - 1,000 taxa → 2 days
  - (effectively unbounded speedup)
- 2004 DCM-GRAPPA
  - Handles unequal gene content
  - (first method with this capability)

Challenges in Phylogeny

- Network evolution
  - Recombination events
- Large-scale phylogeny reconstruction
  - Scalability and Accuracy
- Comparison and accuracy of techniques and heuristics

Cyberinfrastructure Challenges

- Current HPC systems are designed for physics-based simulations that use
  - Floating-point, linear algebra
    - Top 500 List measures Linpack!
  - Regular operations (high-degrees of locality)
    - e.g., Matrices, FFT, CG
  - Low-order polynomial-time algorithms
- Focus of current HPC systems:
  - Dense linear algebra
  - Sparse linear algebra
  - FFT or multi-grid
  - Global scatter-gather operations
  - Dynamically evolving coordinate grids
  - Dynamic load-balancing
  - Particle-based or lattice-gas algorithms
  - Continuum equation solvers

- Computational biology and bioinformatics often require
  - Integer performance
    - Strings, trees, graphs
  - Combinatorics
    - Optimization, LP
    - Computational geometry
  - Irregular data accesses
  - Heuristics and solutions to NP-hard problems
- Next-generation cyberinfrastructure must take Biology into consideration

Parsimony Codes

- Phylip (Felsenstein)
  - http://evolution.genetics.washington.edu/phylip.html
- Hennig86 (Farris)
  - http://www.cladistics.org/
- Nona (Goloboff) and TNT (Goloboff, Farris, Nixon)
  - http://www.cladistics.com/
- PAUP* (Swofford)
  - http://paup.csit.fsu.edu/
- MEGA (Kumar, Tamura, Jakobsen, Nei)
  - http://www.megasoftware.net/
- GRAPPA (Bader, Moret, Warnow)
  - http://www.phylo.unm.edu/

Likelihood Codes

- Phylip (Felsenstein)
  - http://evolution.genetics.washington.edu/phylip.html
- PAUP* (Swofford)
  - http://paup.csit.fsu.edu/
- PAML (Yang)
  - http://abacus.gene.ucl.ac.uk/software/paml.html
- FastDNAml (Olsen, Matsuda, Hagstrom, Overbeek)
  - http://geta.life.uiuc.edu/~gary/programs/fastDNAml.html

- Felsenstein’s List of Software:
  - http://evolution.genetics.washington.edu/phylip/software.html

Part III References

Bader Publications (1/3)

- **A New Implementation and Detailed Study of Breakpoint Analysis**, B.M.E. Moret, S. Wyman, D.A. Bader, T. Warnow, M. Yan, *Sixth Pacific Symposium on Biocomputing 2001*, pp. 583-594, Hawaii, January 2001.
- **High-Performance Algorithm Engineering for Gene-Order Phylogenies**, D.A. Bader, B. M.E. Moret, T. Warnow, S.K. Wyman, and M. Yan, *DIMACS Workshop on Whole Genome Comparison*, DIMACS Center, Rutgers University, Piscataway, NJ, March 2001.
- **Variation in vegetation growth rates: Implications for the evolution of semi-arid landscapes**, C. Restrepo, B.T. Milne, D. Bader, W. Pockman, and A. Kerkhoff, *16th Annual Symposium of the US-International Association of Landscape Ecology*, Arizona State University, Tempe, April 2001.
- **High-Performance Algorithm Engineering for Computational Phylogeny**, B. M.E. Moret, D.A. Bader, and T. Warnow, *2001 International Conference on Computational Science*, San Francisco, CA, May 2001.
- **New approaches for using gene order data in phylogeny reconstruction**, R.K. Jansen, D.A. Bader, B. M. E. Moret, L.A. Raubeson, L.-S. Wang, T. Warnow, and S. Wyman. *Botany 2001*, Albuquerque, NM, August 2001.
- **GRAPPA: a high-performance computational tool for phylogeny reconstruction from gene-order data**, B. M.E. Moret, D.A. Bader, T. Warnow, S.K. Wyman, and M. Yan. *Botany 2001*, Albuquerque, NM, August 2001.
- **Inferring phylogenies of photosynthetic organisms from chloroplast gene orders**, L.A. Raubeson, D.A. Bader, B. M.E. Moret, L.-S. Wang, T. Warnow, and S.K. Wyman. *Botany 2001*, Albuquerque, NM, August 2001.
- **Industrial Applications of High-Performance Computing for Phylogeny Reconstruction**, D.A. Bader, B. M.E. Moret, and L. Vawter, *SPIE ITCom: Commercial Applications for High-Performance Computing*, Denver, CO, SPIE Vol. 4528, pp. 159-168, August 2001.
- **A Linear-Time Algorithm for Computing Inversion Distance Between Two Signed Permutations with an Experimental Study**, D.A. Bader, B. M.E. Moret, and M. Yan, *Journal of Computational Biology*, 8(5):483-491, October 2001.
- **High-Performance Algorithm Engineering for Computational Phylogeny**, B. M.E. Moret, D.A. Bader, and T. Warnow, *The Journal of Supercomputing*, 22:99-111, 2002.

Bader Publications (2/3)

- **Comparative chloroplast genomics of seed plants: integrating computational methods, phylogeny, and molecular evolution,** T.J. Warnow, J.L. Boore, H.M. Fourcade, R.K. Jansen, R. Haberle, T.W. Chumley, L. Raubeson, S. Wyman, C. dePamphilis, B. Moret, D. Bader, W. Miller, (Poster Session), *Evolution 2003*, Chico, CA, June 20-24, 2003.
- **Scalable Graph Algorithms for Shared Memory, D.A. Bader**, *11th SIAM Conference on Parallel Processing for Scientific Computing (PP04)*, San Francisco, CA, February 25-27, 2004.
- **High-Performance Computing for Reconstructing Evolutionary Trees from Gene-Order Data, D.A. Bader**, Minisymposium on Parallel Computational Biology, *11th SIAM Conference on Parallel Processing for Scientific Computing (PP04)*, San Francisco, CA, February 25-27, 2004.
- **Fast, Sparse Graph Algorithms using Symmetric Multiprocessors, D.A. Bader**, Minisymposium on Combinatorial Algorithms and Parallel Computing, *11th SIAM Conference on Parallel Processing for Scientific Computing (PP04)*, San Francisco, CA, February 25-27, 2004.
- **BioSPLASH: A sample workload from bioinformatics and computational biology for optimizing next-generation high-performance computer systems, D.A. Bader**, V. Sachdeva, A. Trehan, V. Agarwal, G. Gupta, and A.N. Singh, (Poster Session), *13th Annual International Conference on Intelligent Systems for Molecular Biology (ISMB 2005)*, Detroit, MI, June 25-29, 2005.
- **Incorporating life sciences applications in the architectural optimizations of next-generation petaflop-system, D.A. Bader**, V. Sachdeva, (Poster Session), *The 4th IEEE Computational Systems Bioinformatics Conference (CSB 2005)*, Stanford University, CA, August 8-11, 2005.
- **BioPerf: A Benchmark Suite to Evaluate High-Performance Computer Architecture on Bioinformatics Applications, D.A. Bader**, Y. Li, T. Li, V. Sachdeva, *The IEEE International Symposium on Workload Characterization (IISWC 2005)*, Austin, TX, October 6-8, 2005.
- **Opportunities and Challenges in Computational Biology,** S. Aluru and **D.A. Bader**, *The IEEE and ACM Supercomputing Conference 2002 (SC2002)* Tutorial, Baltimore, MD, November 17, 2002.
- **High-Performance Algorithm Engineering for Large-Scale Graph Problems and Computational Biology, D.A. Bader**, *Proc. 4th International Workshop on Efficient and Experimental Algorithms (WEA)*, *Lecture Notes in Computer Science*, 3503:16-21, May 2005.
- **Computational Biology and High-Performance Computing, D.A. Bader**, Special Issue on Bioinformatics, *Communications of the ACM*, 47(11):34-41, 2004.

Bader Publications (3/3)

- **Computational Grand Challenges in Assembling the Tree of Life: Problems & Solutions,** David A. Bader, Usman Roshan, and Alexandros Stamatakis, The IEEE and ACM Supercomputing Conference 2005 (SC2005), Seattle, WA, November 13, 2005.
- **ExactMP: An Efficient Parallel Exact Solver for Phylogenetic Tree Reconstruction Using Maximum Parsimony,** D.A. Bader, V. Chandu, and M. Yan, *The 35th International Conference on Parallel Processing (ICPP 2006),* Columbus, OH, August 14-18, 2006.
- **Parallel Algorithms for Evaluating Centrality Indices in Real-world Networks,** D.A. Bader and K. Madduri, *The 35th International Conference on Parallel Processing (ICPP 2006),* Columbus, OH, August 14-18, 2006.
- **Parallel Algorithm Design for Branch and Bound, David A. Bader**, William E. Hart, and Cynthia A. Phillips, in H.J. Greenberg, editor, *Tutorials on Emerging Methodologies and Applications in Operations Research*, Kluwer Academic Press, Chapter 5, pp. 1-44, 2004.
- **High Performance Algorithms for Phylogeny Reconstruction with Maximum Parsimony, David A. Bader** and Mi Yan, in S. Aluru, editor, *Handbook of Computational Molecular Biology*, Chapman & Hall / CRC Computer and Information Science Series, Chapter 22, pp. 1-19, 2006.
- **High-Performance Phylogeny Reconstruction Under Maximum Parsimony, David A. Bader**, Bernard M.E. Moret, Tiffani L. Williams, and Mi Yan, in A.Y. Zomaya, editor, *Parallel Computing for Bioinformatics and Computational Biology*, Wiley, Chapter 16, 2006.
- **Parallel Computational Biology,** Srinivas Aluru, Nancy Amato, **David A. Bader**, Suchendra Bhandarkar, Laxmikant Kale, and Dan Marinescu, in M.H. Heroux, P. Raghavan, and H.D. Simon, editors, *Frontiers of Scientific Computing*, SIAM Series on Software, Environments, and Tools, 2006, to appear.
- **Computational Grand Challenges in Assembling the Tree of Life: Problems & Solutions,** David A. Bader, Usman Roshan, and Alexandros Stamatakis, in C.-W. Tseng, editor, *Advances in Computing: Bioinformatics and Computational Biology*, Elsevier, 2006, to appear.

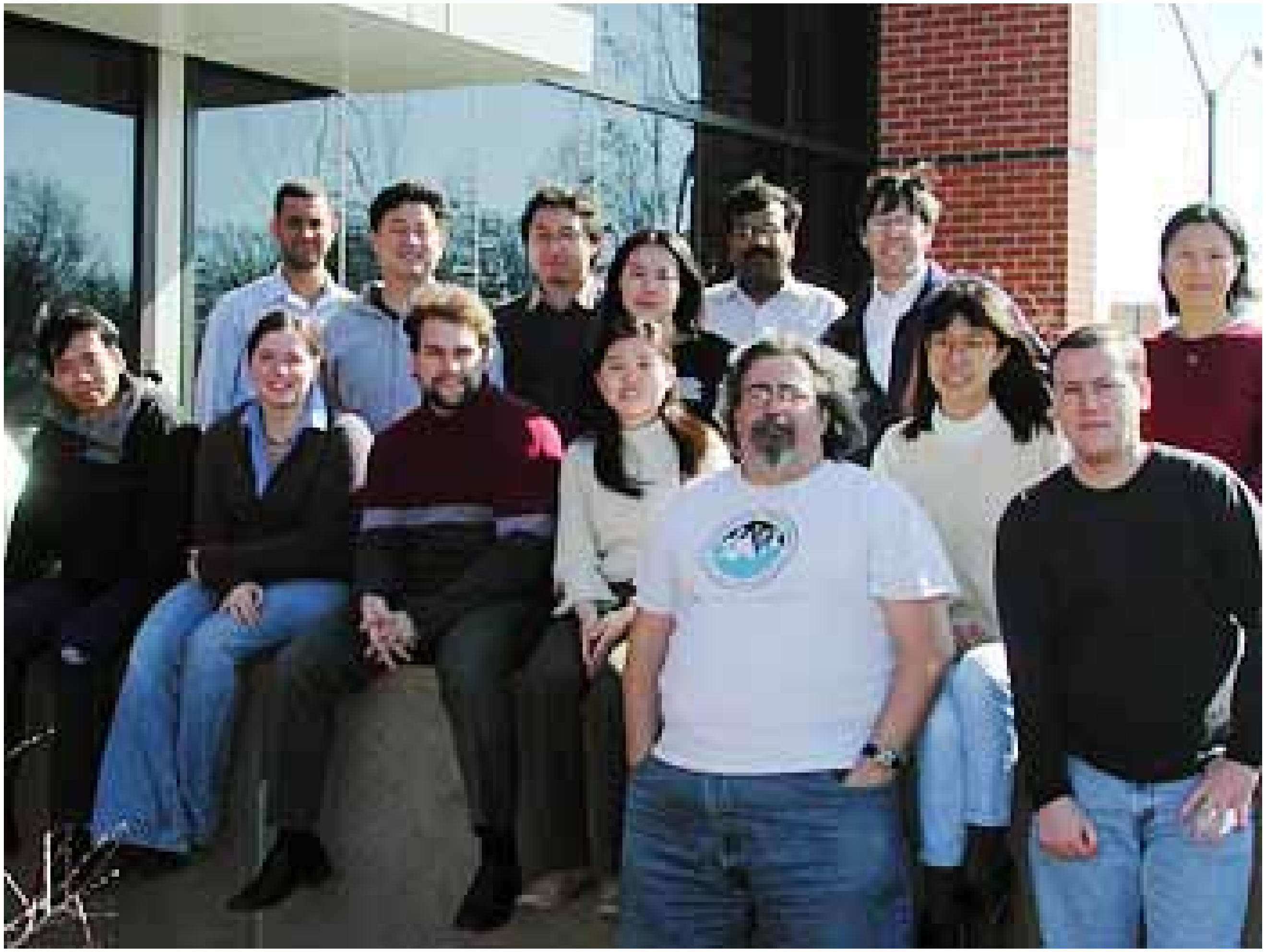
Acknowledgements for Part I

- mpiBLAST authors: A. Darling, L. Carey, W.Feng
- ScalaBLAST authors: C. Oehmen, J. Nieplocha
- H. Lin, X. Ma, P. Chandramohan, A. Geist, N. Samatova

Acknowledgements for Part II

- S.J. Emrich, P.S. Schnable @ Iowa State University
- Y. Fu @ Donald Danforth Plant Science Center
- D. Ashlock @ University of Guelph
- S.D. Ellis, K. Pinnow, B. Smith @ IBM, Rochester, MN

Acknowledgements for Part II …



Acknowledgement of Support (for Part II)

- NSF/DOE/USDA Grant:
  - Sequencing the Maize Genome
- NSF Grants:
  - Acquisition of a 512-node BlueGene/L Supercomputer for Large-Scale Applications in Genomics and Systems Biology
  - Efficient Representation and Manipulation of Large-Scale Biological Sequence Data
  - CREST: Center for Research Excellence in Bioinformatics and Computational Biology
  - Parallel Algorithms and Software for EST Clustering
- IBM
- SUN

Acknowledgment of Support (for Part III)

- National Science Foundation
  - **CSR**: A Framework for Optimizing Scientific Applications (06-14915)
  - **CAREER**: High-Performance Algorithms for Scientific Applications (06-11589; 00-93039)
  - **ITR**: Building the Tree of Life -- A National Resource for Phyloinformatics and Computational Phylogenetics (EF/BIO 03-31654)
  - **ITR/AP:** Reconstructing Complex Evolutionary Histories (01-21377)
  - **DEB** Comparative Chloroplast Genomics: Integrating Computational Methods, Molecular Evolution, and Phylogeny (01-20709)
  - **ITR/AP(DEB):** Computing Optimal Phylogenetic Trees under Genome Rearrangement Metrics (01-13095)
  - **DBI:** Acquisition of a High Performance Shared-Memory Computer for Computational Science and Engineering (04-20513).
- IBM PERCS / DARPA High Productivity Computing Systems (HPCS)
  - DARPA Contract NBCH30390004
- IBM Shared University Research (SUR) Grant
- Sun Academic Excellence Grant (AEG)

Thank You