High Performance Computing in Bioinformatics

Thomas Ludwig (t.ludwig@computer.org)
Ruprecht-Karls-Universität Heidelberg
Computer Science Department

Alexandros Stamatakis (stamatak@cs.tum.edu)
Technische Universität München
Computer Science Department

Tutorial Outline

- PART I: High Performance Computing
  Thomas Ludwig
- PART II: HPC Computing in Bioinformatics
  Alexandros Stamatakis
  - Grand Challenges for HPC Bioinformatics
  - HPC Bioinformatics by Example of Phylogenetic Inference

PART I
High Performance Computing

- Introduction
- Architecture
- Top Systems
- Programming
- Problems
- Own Research
- The Future

Outline

- Introduction
- Architecture
- Top Systems
- Programming
- Problems
- Own Research
- The Future

Introduction
Why High Performance Computing?

- Situation in science and engineering
  - Replace complicated physical experiments by computer simulations
  - Evaluate more fine-grained models
- User requirements
  - Compute masses of individual tasks
  - Compute complicated single tasks
- Available computational power
  - Single workstation is not sufficient

What Is It?

- High Performace Computing (HPC), Networking and Storage

  Deals with high and highest performance computers, with high speed networks, and powerful disk and tape storage systems
- Performance improvement
  - Compared to personal computers and small workstations:

    Factor 100...10.000

What Do I Need?

- Small scale high performance computing
  - Cheapest version: use what you have
    Workstations with disks and network
  - A bit more expensive: buy PCs
    - E.g. 16 personal computers with disks and gigabit ethernet
  - It´s mainly a human ressources problem
    - Network of workstations is time consuming to maintain
    - Software comes for free

What Do I Need...?

- Large scale high performance computing
  - Buy 10.000 PCs or a dedicated supercomputer
  - Buy special hardware for networking and storage
  - Add a special building
  - Add an electric power station

How Much Do I Have to Pay?

- Small scale (<64 nodes)
  - 1000€/node
- Medium scale (64-1024 nodes)
  - 2000€/node (multiprocessor, 64-bit)
  - 1500€/node for high speed network
  - 500€/node for high performance I/O
- Large scale (>1024 nodes)
  - Money for building
  - Money for power plant
  - Current costs range between 20...400 million Euros

Application Fields

- Numerical calculations and simulations
  - Particle physics
  - Computational fluid dynamics
  - Car crash simulations
  - Weather forecast
  - ...
- Non-numerical computations
  - Chess playing, theorem proving
  - Commercial database applications

Application Fields...

- All fields of Bioinformatics
  - Computational genomics
  - Computational proteomics
  - Computational evolutionary biology
  - …
- In general
  - Everything that runs beween 1 and 10.000 days
  - Everything that uses high volumes of data

Measures

Mega ($2^{20} \cong 10^6$) – Giga ($2^{30} \cong 10^9$) – Tera ($2^{40} \cong 10^{12}$)
Peta ($2^{50} \cong 10^{15}$) – Exa ($2^{60} \cong 10^{18}$)

- Computational Performance (Flop/s)
  Flop/s = floating point operations per second
  - Modern processor: 3 GFlop/s
  - Nr. 1 supercomputer: 35 TFlop/s (factor 10.000)
- Network performance (Byte/s)
  - Personal computer: 10/100 MByte/s
  - Supercomputer networks: gigabytes/s

Measures...

- Main memory (Byte)
  - Personal computer: 1 GByte
  - Nr. 1 supercomputer: 10 TByte (factor 10.000)
- Disk space (Byte)
  - Single disk 2004: 200 GByte
  - Nr. 1 supercomputer: 700 TByte (factor 3.500)
- Tape storage (Byte)
  - Personal computer: 200 GByte
  - Nr. 1 supercomputer: 1.6 Pbyte (factor 8.000)

Outline

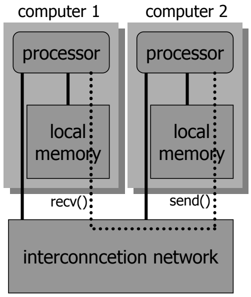
- Introduction
- Architecture
- Top Systems
- Programming
- Problems
- Own Research
- The Future

Architecture

- Basic classification concept:
  How is the main memory organized?
  - Distributed memory architecture
  - Shared memory architecture

- Available systems
  - Dedicated supercomputers
  - Cluster systems

Distributed Memory Architecture

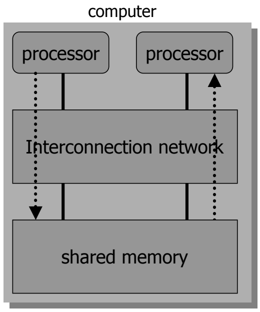


- Autonomous computers connected via network
- Processes on each compute node have access to local memory only
- Parallel program spawns processes over a set of processors
- Communication between computers (processes) via message passing
- Called: multi computer system

Distributed Memory Architecture...

- Advantages
  - Good scalability: just buy new nodes
    - Concept scales up to 10.000+ nodes
  - You can use what you already have
  - Extend the system when you have money and need for more power
- Disadvantages
  - Complicated programming: parallelization of formerly sequential programs
    (including complicated debugging, performance tuning, load blancing, etc.)

Shared Memory Architecture



- Several processors in one box (e.g. multiprocessor mother-board)
- Each process on a processor sees complete address space
- Communication between processes via shared variables
- Called: multiprocessor system, symmetric multi-processing system (SMP)

Shared Memory Architecture...

- Advantages
  - Much easier programming
- Disadvantages
  - Limited scalability: up to 64 processors
    Reason: interconnection network becomes bottleneck
  - Limited extensibility
  - Very expensive due to high performance interconnection network

---

Hybrid Architectures

- Use several SMP systems
  - Combination of shared memory systems and distributed memory system
- The good thing: scalable performance according to your financial budget
- The bad thing: programming gets even more complicated (hybrid programming)
- The reality: vendors like to sell these systems, because they are easier to build

---

Supercomputers vs. Clusters

- Supercomputers (Distributed/shared memory)
  - Constructed by a major vendor (IBM, HP, ...)
  - Use custom components (processor, network, ...)
  - Custom (Unix-like) operating systems
- Clusters (Network of workstations, NOWs)
  - Assembled by vendor or users
  - Commodity-of-the-shelf components (COTS)
  - Linux operating system

Supercomputers vs. Clusters...

- Supercomputers
  - Very expensive to buy
  - Usually high availability and scalability

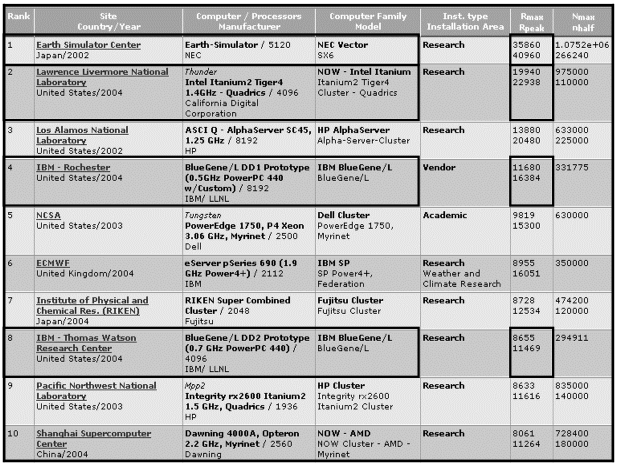
- Clusters
  - Factor 10 cheaper to buy, but:
  - Very expensive to own
  - Lower overall availability and scalability

Outline

- Introduction
- Architecture
- Top Systems
- Programming
- Problems
- Own Research
- The Future

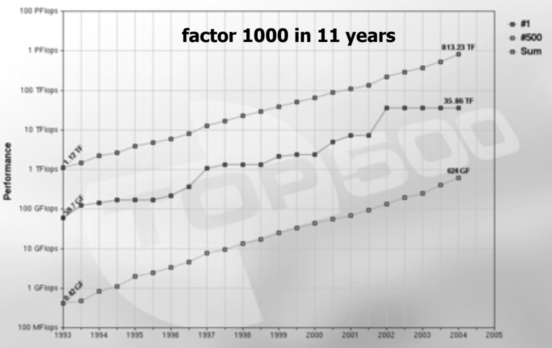
TOP500-List

- Lists world 500 most powerful systems
  www.top500.org
- Update in June and November
- Ranking based on numerical algorithm
- In 6 months almost half of the systems fall off the list
- The majority of systems now are clusters



Rank	Site Country/Year	Computer / Processors Manufacturer	Computer Family Model	Inst. type Installation Area	Rmax Rpeak	Nmax nhalf
1	Earth Simulator Center Japan/2002	**Earth-Simulator** / 5120 NEC	**NEC Vector** SX6	**Research**	35860 40960	1.0752e+06 266240
2	Lawrence Livermore National Laboratory United States/2004	*Thunder* **Intel Itanium2 Tiger4 1.4GHz - Quadrics** / 4096 California Digital Corporation	**NOW - Intel Itanium** Itanium2 Tiger4 Cluster - Quadrics	**Research**	19940 22938	975000 110000
3	Los Alamos National Laboratory United States/2002	**ASCI Q - AlphaServer SC45, 1.25 GHz** / 8192 HP	**HP AlphaServer** Alpha-Server-Cluster	**Research**	13880 20480	633000 225000
4	IBM - Rochester United States/2004	**BlueGene/L DD1 Prototype (0.5GHz PowerPC 440 w/Custom)** / 8192 IBM/ LLNL	**IBM BlueGene/L** BlueGene/L	**Vendor**	11680 16384	331775
5	NCSA United States/2003	*Tungsten* **PowerEdge 1750, P4 Xeon 3.06 GHz, Myrinet** / 2500 Dell	**Dell Cluster** PowerEdge 1750, Myrinet	**Academic**	9819 15300	630000
6	ECMWF United Kingdom/2004	**eServer pSeries 690 (1.9 GHz Power4+)** / 2112 IBM	**IBM SP** SP Power4+, Federation	**Research** Weather and Climate Research	8955 16051	350000
7	Institute of Physical and Chemical Res. (RIKEN) Japan/2004	**RIKEN Super Combined Cluster** / 2048 Fujitsu	**Fujitsu Cluster** Fujitsu Cluster	**Research**	8728 12534	474200 120000
8	IBM - Thomas Watson Research Center United States/2004	**BlueGene/L DD2 Prototype (0.7 GHz PowerPC 440)** / 4096 IBM/ LLNL	**IBM BlueGene/L** BlueGene/L	**Research**	8655 11469	294911
9	Pacific Northwest National Laboratory United States/2003	*Mpp2* **Integrity rx2600 Itanium2 1.5 GHz, Quadrics** / 1936 HP	**HP Cluster** Integrity rx2600 Itanium2 Cluster	**Research**	8633 11616	835000 140000
10	Shanghai Supercomputer Center China/2004	**Dawning 4000A, Opteron 2.2 GHz, Myrinet** / 2560 Dawning	**NOW - AMD** NOW Cluster - AMD - Myrinet	**Research**	8061 11264	728400 180000
11	Los Alamos National Laboratory United States/2003	*Lightning* **Opteron 2 GHz, Myrinet** / 2816 Linux Networx	**NOW - AMD** NOW Cluster - AMD - Myrinet	**Research**	8051 11264	761160 109208
12	Lawrence Livermore National Laboratory United States/2002	**MCR Linux Cluster Xeon 2.4 GHz - Quadrics** / 2304 Linux Networx/Quadrics	**NOW - Intel Pentium** NOW Cluster - Intel Pentium - Quadrics	**Research**	7634 11060	350000 75000
13	Lawrence Livermore National Laboratory United States/2000	**ASCI White, SP Power3 375 MHz** / 8192 IBM	**IBM SP** SP Power3 375 MHz high node	**Research**	7304 12288	640000
14	NERSC/LBNL United States/2002	*Seaborg* **SP Power3 375 MHz 16 way** / 6656 IBM	**IBM SP** SP Power3 375 MHz high node	**Research**	7304 9984	640000
15	NCSA United States/2004	**TeraGrid, Itanium2 1.3/1.5 GHZ, Myrinet** / 1776 IBM	**NOW - Intel Itanium** Titan Cluster Itanium2 - Myrinet	**Academic**	7215 10259	540000
16	Lawrence Livermore National Laboratory United States/2003	**xSeries Cluster Xeon 2.4 GHz - Quadrics** / 1920 IBM/Quadrics	**IBM Cluster** xSeries Cluster Xeon - Quadrics	**Research**	6586 9216	425000 90000
17	Lawrence Livermore National Laboratory United States/2004	*Lilac* **xSeries Xeon 3.06 GHz, Quadrics** / 1540 IBM	**IBM Cluster** xSeries Cluster Xeon - Quadrics	**Research**	6232 9425	
18	HPCx United Kingdom/2004	**eServer pSeries 690 (1.7 GHz Power4+)** / 1600 IBM	**IBM SP** SP Power4+, Federation	**Academic**	6188 10880	355000
19	Grid Technology Research Center, AIST Japan/2004	*AIST Super Cluster P-32* **AIST Super Cluster P-32, Opteron 2.0 GHz, Myrinet** / 2200 IBM	**IBM Cluster** eServer Cluster Opteron - Myrinet	**Research**	6155 8800	
20	Oak Ridge National Laboratory United States/2004	**Cray X1** / 504 Cray Inc.	**Cray X1** Cray X1	**Research**	5895 6451	494592 53760

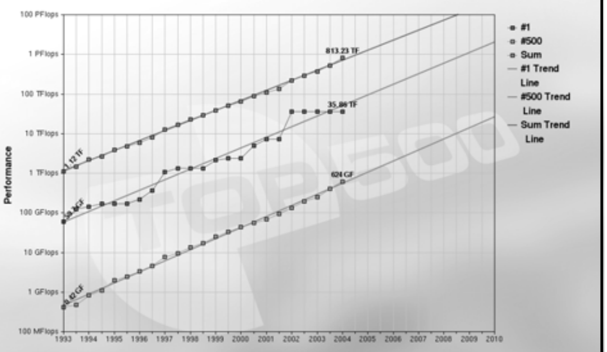
Performance Development



factor 1000 in 11 years

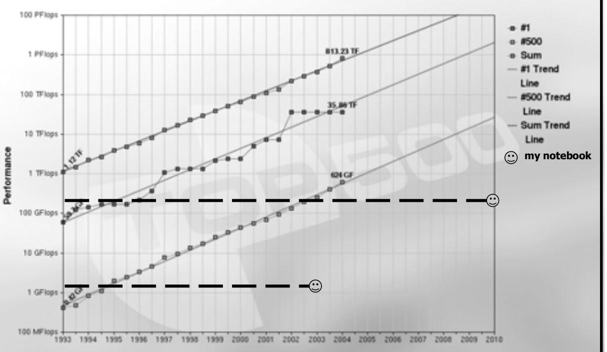
23rd List / June 2004 | http://www.top500.org/

Projected Performance Development



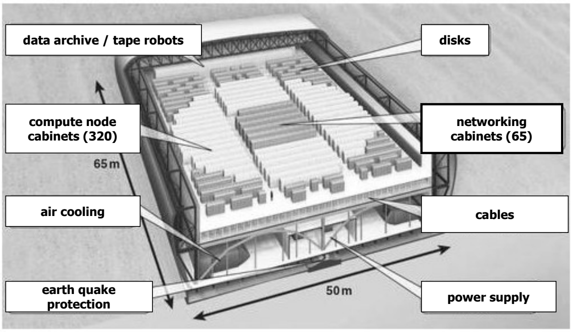
23rd List / June 2004 | http://www.top500.org/

Projected Performance Development



23rd List / June 2004 | http://www.top500.org/

EarthSimulator / NEC



© Thomas Ludwig, Alexandros Stamatakis, GCB'04

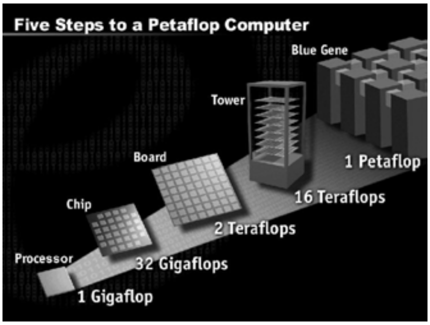
EarthSimulator / NEC

- 640 nodes x 8 processors = 5120 processors
- 200MioUSD for computer
- 200MioUSD for building and electric power station
- Building: 3250m^2 earth quake protected
- Power: 7MW

- 10TByte main memory
- 700TByte disk
- 1.6PBytes tapes
- 83.000 copper cables
- 2.800km / 220t of cabeling
- Application field: climate simulations

Computenic-Shock

BlueGene / IBM



BlueGene / IBM

- IBM intends to break the 1PFlop/s barrier by end of 2006
- Power consumption and floor space problems solved by new packing technology
  (less than 30 Earth-Simulators! ☺)

- Application fields: bioinformatics
  - Ab initio protein folding
  - Molecular dynamics on a millisecond to second time scale
  - Protein structure prediction

Outline

- Introduction
- Architecture
- Top Systems
- Programming
- Problems
- Own Research
- The Future

Programming
Parallelization Paradigms

- What do we have?
  - Many processors, one program, much data
- How do we proceed?
  - Start one instance of the program on each processor (called process)
  - Give it one part of the data
  - Collect the result
- Is this all?
  - Well, sometimes yes ☺, sometimes no ☹

Parallelization Paradigms...

- Remember the two categories
  1. Compute masses of individual tasks
  2. Compute complicated single taks
- Category 1: embarrassingly parallel
  - Master/worker concept:
    Start master on one processor
    Start workers on all other processors
    Master sends data to workers
    Master collects results from workers
  - Example: Analysis of different molecules

Parallelization Paradigms...

- Category 2: non-trivial parallelism
  - Usually: data partitioning

    Data is partitioned amongst the processes

    Each process computes results

    During computation processes have to coordinate with each other: done by communication

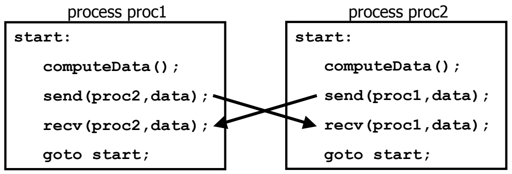
    A selected process organizes input/output
  - Example: Molecular dynamics simulations
  - Only for experienced programmers

Parallelization Support

- Parallelization via automatically parallelizing compilers
  - Complicated, seldom, inefficient, not scalable
- Parallelization for shared memory architectures
  - Language support via OpenMP
  - Only for small node numbers
- Parallelization for distributed memory architectures
  - Done manually
  - Decorate program with message passing calls

Message Passing

- Basic principle: use send and receive library calls to transfer data



process proc1

```
start:
   computeData();
   send(proc2,data);
   recv(proc2,data);
   goto start;
```

process proc2

```
start:
   computeData();
   send(proc1,data);
   recv(proc1,data);
   goto start;
```

Message Passing...

- Message passing standard:

  MPI (Message Passing Interface)
- Conceived in the mid 90s
  - Language bindings: C, C++, Fortran (others available)
- Divided into two parts
  - MPI: just message passing
  - MPI-2: process management, input/output, ...
- Available implementations
  - Vendor versions on all major supercomputers
  - Open source versions MPICH and LAME for workstations clusters

---

Message Passing...

- How big is MPI/MPI-2?
  - Several hundreds of library calls!
  - Don´t worry: half a dozen is enough to start with
- What makes MPI so copious?
  - All sorts of library calls to just have more comfort with programming:
    - Collective calls: e.g. compute a global sum over all processes and distribute it to all of them
    - Special communication calls: e.g. non-blocking calls provide means to overlap computation and communication

---

Message Passing... The Dark Side

- Parallel programming in general introduces new error category: *Time dependent errors*
- The reason for this
  - Processes run in parallel but are not synchronized; timing depends on concrete acitivites on the nodes
- The consequences!
  - An erroneous program crashes only sometimes
  - When you slow it down to observe it, it does not crash
  - The precondition for the crash cannot be reproduced (nondeterminism)
- The solution? Well, next question ☹

Performance Limitations: Amdahl´s Law

- How to evaluate parallel performance
  - Run with 1 and with n processors
  - Speedup: S(n) = t(1) / t(n)
  - Efficiency: E(n) = S(n) / n
- Theoretical maximum: $S_{max}=n$
- Practical maximum by Amdahl´s Law

Performance Limitations: Amdahl´s Law...

- Every parallal program has a non-parallel part (input/output, initialization, etc.)
- Fraction of non-parallel part is f
- Maximum speedup with Amdahl

  $S_{Amdahl} = 1 / (f + (1-f)/p)$

  Examples:
  - $F=0.01 \Rightarrow S_{Amdahl}=100$
  - $F=0.001 \Rightarrow S_{Amdahl}=1000$

Outline

- Introduction
- Architecture
- Top Systems
- Programming
- Problems
- Own Research
- The Future

Problems
Fault Tolerance

- What happens when during runtime a node crashes? (probability increases with number of components and execution time)
- Today
  - The program just crashes
  - The program crashes but can be restarted from a checkpoint
- In the future
  - The program continues with execution and uses a different set of nodes

---

High Performance Input/Output

- How can we handle terabytes and petabytes of input/output data?
- Today
  - Not yet too efficiently; often a master process handles all I/O
- In the future
  - All processes perform I/O-calls (parallel I/O)
  - All nodes have efficient access to high performance I/O hardware

---

Outline

- Introduction
- Architecture
- Top Systems
- Programming
- Problems
- Own Research
- The Future

Own Research
High Performance Parallel I/O

- High performance I/O is a major challenge today
- Research activities
  - Investigate load balancing mechanisms for parallel I/O systems (i.e. distribute data to disks according to load)
  - Provide performance measurement tools to *see* the influence of I/O library calls in the source code
  - Adapt parallel programs to parallel I/O concepts

Outline

- Introduction
- Architecture
- Top Systems
- Programming
- Problems
- Own Research
- The Future

The Future
Petaflops and Petabytes

- The new borders of supercomputing
  - Petaflops: maybe by end of 2006?
  - Petabytes: next year
  - Number of processors: 10.000 and more
- However: do we have the right algorithms and programs for that?

Gridcomputing

- The new hype of supercomputing
  - Idea: as with the power grid
    - Computational performance should be available everywhere
    - Computational performance can be produced at various places
  - Concept
    - Join parallel computers, cluster, compute centers
    - Offer their aggregate compute performance
  - Problems
    - Programming, management, security, ...

Limitations of High Performance Computing

- Reconsider: „everything that runs between 1 and 10000 days"
  - Current supercomputers can reduce the program runtime by a maximum factor of 5 orders of magnitude
- What to do, if you want to compute e.g. billions of molecular variations?
- There is only one answer:

  First: Improve the algorithm!

  Second: Use supercomputers

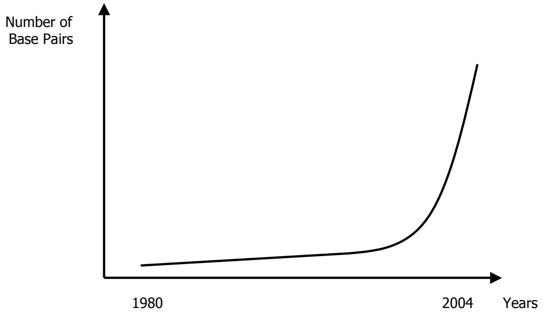
PART II: HPC in Bioinformatics

- Grand Challenges in HPC Bioinformatics
- HPC Bioinformatics by Example of Phylogenetic Inference
  - phylogenetic analysis
  - maximum likelihood
  - bayesian inference
  - sequential codes
  - parallel & distributed RAxML
  - future developments

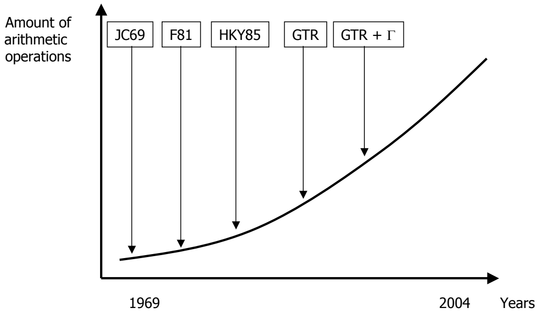
Outline

- Grand Challenges in HPC Bioinformatics
- HPC Bioinformatics by Example of Phylogenetic Inference
  - phylogenetic analysis
  - maximum likelihood
  - bayesian inference
  - sequential codes
  - parallel & distributed RAxML
  - future developments

The Classic Slide: GenBank Data Growth



Increase in Nucleotide Substitution Model Complexity



Grand Challenges

- Protein folding & structure prediction
- Homology search
- Multiple alignment
- Genomic sequence analysis
- Gene finding
- Gene expression data analysis
- Drug discovery
- Phylogenetic inference

Grand Challenges

- Protein folding & structure prediction
- Homology search
- Multiple alignment
- Genomic sequence ar
- Gene finding
- Gene expression data nalysis
- Drug discovery
- Phylogeny construction

Main focus of this talk!

Multiple Alignment

- **The** prerequisite for phylogenetic analysis
- Computational effort increases exponentially in time **and** space with standard dynamic programming approach and Sum-of-Pairs score →

Which is the adequate score function?

Multiple Alignment

- **The** prerequisite for phylogenetic analysis
- Computational effort increases exponentially in time **and** space with standard dynamic programming approach and Sum-of-Pairs score →
  - Good heuristics
  - Parallel algorithms
    - Fine-grained, e.g. on alignment matrix level
    - FPGA implementations for pairwise alignment
  - Heuristics & parallel algorithm
    - coarse-grained divide-and-conquer

Multiple Alignment

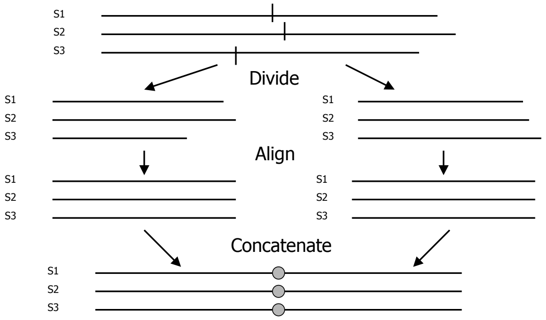
- **The** prere analysis
- Computati ponentially in time **an** ynamic programm of-Pairs score →
  - Good heu
  - Parallel a
    - Fine-grain g. on alignment matrix level
    - FPGA im lementations for pairwise alignment
  - Heuristics & parallel algorithm
    - coarse-grained divide-and-conquer

Type of reverse engineering: How can I change the algorithm to be able to parallelize it with MPI?

An example: DCA

- Stoye et al (1997) Divide-and-Conquer Alignment Algorithm (DCA)
  1. Divide sequences into smaller subsequence-sets
  2. If length of multiple subsequence-set < predifined threshold value L compute optimal subalignments in parallel
  3. Concatenate subalignments to whole alignment
- The Art consists in the design of intelligent decomposition heuristics to obtain near optimal concatenated alignment → non-trivial problem

DCA



Outline

- Grand Challenges in HPC Bioinformatics
- HPC Bioinformatics by Example of Phylogenetic Inference
  - phylogenetic analysis
  - maximum likelihood
  - bayesian inference
  - sequential codes
  - parallel & distributed RAxML
  - future developments

Phylogenetic Analysis

- Motivation
  - Tree-of-life
  - New insights in medical & biological research
  - CIPRES: NSF-funded 11.6 million $ tree-of-life project (www.phylo.org)
- Applications of phylogenetic trees
  - Bader et al (2001) Industrial applications of high-performance computing for phylogeny reconstruction.
  - Baker et al (1994) Which whales are hunted? A molecular genetic approach to whaling.

Phylogenetic Methods

- Input: “good” multiple Alignment
- Output: unrooted binary tree
- Various models for phylogenetic inference
    - Models differ in computational complexity & accuracy of final trees
    - Fast & simple models
        - Neighbor Joining
        - Parsimony (MP)
    - Slow & complex models
        - Maximum Likelihood (ML)
        - Bayesian Methods

We focus on ML & Bayesian Methods

Remember !

- Input need not be DNA or protein sequence data → gene order data
    - Moret et al (2001) GRAPPA: a high performance computational tool for phylogeny reconstruction from gene-order data.
- Model need not be a tree → networks
    - Gusfield et al (2003) Efficient reconstruction of phylogenetic networks with constrained recombination.
- Output need not be a strictly bifurcating tree → multifurcating tree

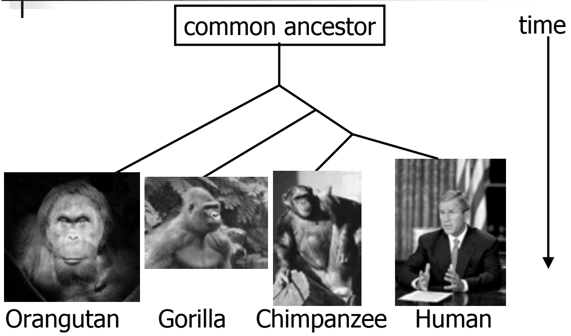
---

Remember !

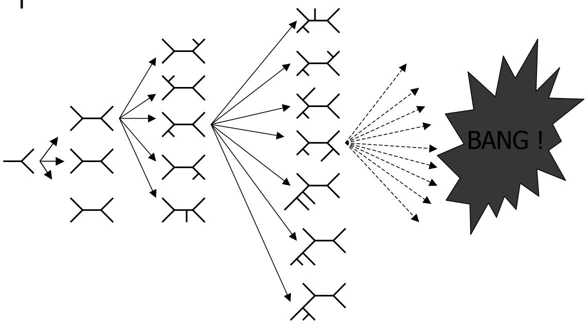
- Input need no be DNA or protein sequence data → gene er data
    - M mance c uction from g
- Model ks
    - G on of p recombination.
- Output need not be a strictly bifurcating tree → multifurcating tree

We focus on computation of strictly bifurcating phylogenetic trees with maximum likelihood for DNA and Protein sequence data !

Example: Phylogeny of great Apes



The number of trees explodes!



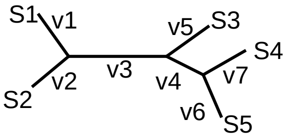
The Algorithmic Problem

- Number of potential trees grows exponentially

# Taxa	# Trees
5	15
10	2.027.025
15	7.905.853.580.62
50	2.84 * 10^76

This is ≈ the number of atoms in the universe 10^80

Maximum Likelihood



Maximum Likelihood calculates:

1. Topologies
2. Branch lengths v[i]
3. Likelihood of the tree

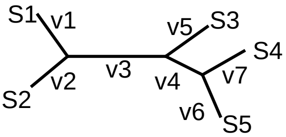
Goal: Obtain topology with maximum likelihood value

Problem I: Number of possible topologies is exponential in n

Problem II: Computation of likelihood function is expensive

Solution: algorithmic optimizations + new heuristics + HPC

Maximum Likelihood



Maximum Likelihood calculates:

1. Topologies
2. Branch lengths v[i]
3. Likelihood of the tree

Goal: Obtain topology with maximum likelihood value

Problem I: Number of possible topologies is exponential in n

Problem II: Computation of likelihood function is expensive

Solution: algorithmic optimizations + new heuristics + HPC

Bayesian Inference

- Uses bayesian statistics
- P (t + m|d) = P (d|t + m) x P (t + m) / P (d)

Likelihood of the tree

Bayesian Inference

- Uses bayesian statistics
- P (t + m|d) = P (d|t + m) x P (t + m) / P (d)

prior probability of the tree: must be assumed; all possible trees are usually considered to be equally probable

---

Bayesian Inference

- Uses bayesian statistics
- P (t + m|d) = P (d|t + m) x P (t + m) / P (d)

problematic term: P(d) equals the sum of the likelihood x prior probability of the tree + model for **all possible trees.**

Bayesian Inference

- Uses bayesian statistics
- P (t + m|d) = P (d|t + m) x P (t + m) / P (d)

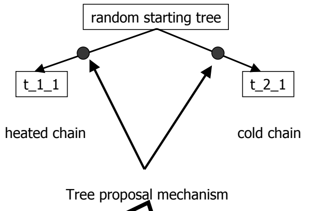
problemati the
sum of th obability
of the tree + **ossible**
trees.

impossible to calculate!

Bayesian Inference

- Solution: Metropolis-Coupled Markov Chain Monte Carlo Simulation (MC^3)
- Advantages compared to ML:
  - straightforward statistical measure of phylogeny
  - Avoids bootstrapping & provides straightforward support values
- Disadvantages compared to ML:
  - Requires prior probabilities for tree & model
  - MC^3 convergence problem
  - More difficult to parallelize: Feng et al (2003) Parallel algorithms for bayesian phylogenetic inference.

MC^3 Algorithm



random starting tree

t_1_1

t_2_1

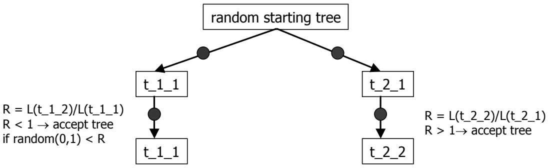
heated chain

cold chain

Tree proposal mechanism

The art in the design of bayesian phylogenetic analysis lies in the tree proposal mechanism.

MC^3 Algorithm



random starting tree

t_1_1

t_2_1

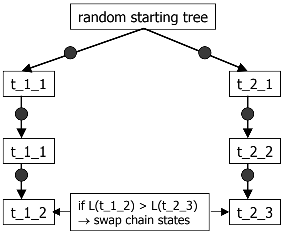
R = L(t_1_2)/L(t_1_1)
R < 1 → accept tree
if random(0,1) < R

t_1_1

t_2_2

R = L(t_2_2)/L(t_2_1)
R > 1→ accept tree

MC^3 Algorithm



random starting tree

t_1_1

t_2_1

t_1_1

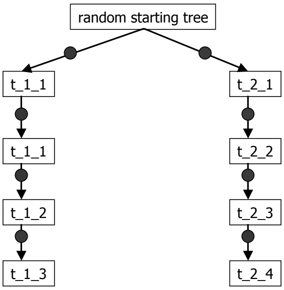
t_2_2

t_1_2

if L(t_1_2) > L(t_2_3)
→ swap chain states

t_2_3

MC^3 Algorithm



random starting tree

t_1_1

t_2_1

t_1_1

t_2_2

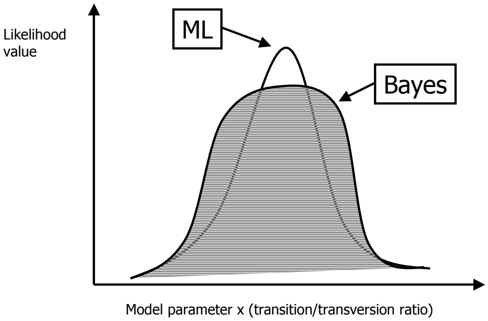
t_1_2

t_2_3

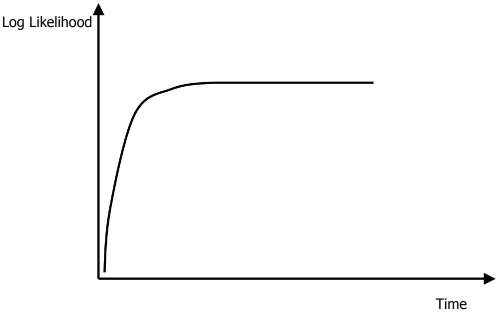
t_1_3

t_2_4

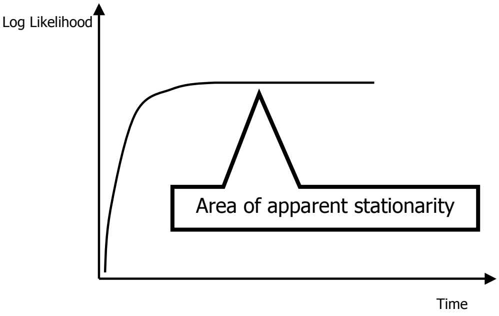
ML vs. Bayes



MC³ convergence problem

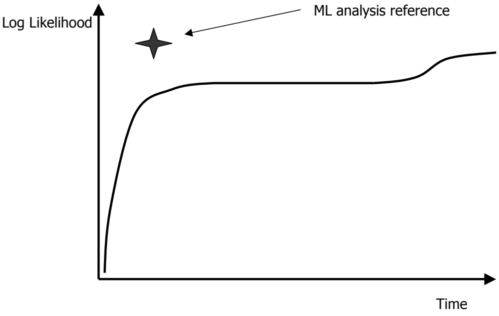


MC³ convergence problem



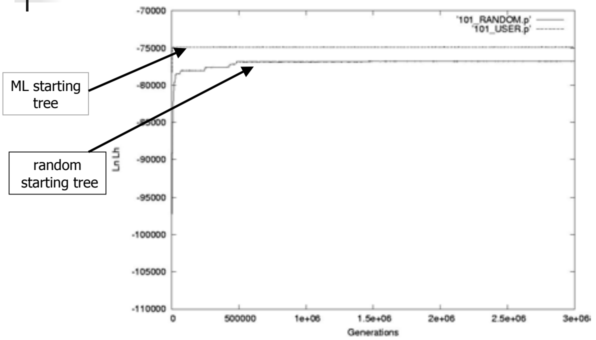
Area of apparent stationarity

MC³ convergence problem



ML analysis reference

Real-world example



ML starting tree

random starting tree

Outline

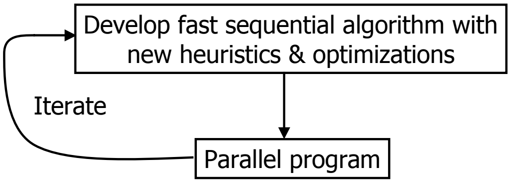
- Grand Challenges in HPC Bioinformatics
- HPC Bioinformatics by Example of Phylogenetic Inference
  - phylogenetic analysis
  - maximum likelihood
  - bayesian inference
  - sequential codes
  - parallel & distributed RAxML
  - future developments

ML Phylogeny Program Development

Develop fast sequential algorithm with new heuristics & optimizations

Phylogenetics are an algorithmic discipline

ML Phylogeny Program Development



Develop fast sequential algorithm with new heuristics & optimizations

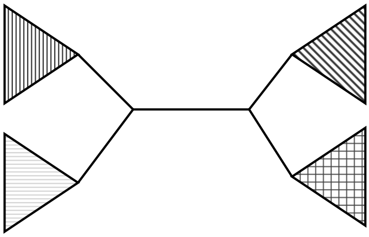
Parallel program

Iterate

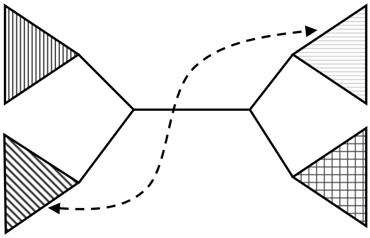
Basic Algorithms

- Two basic classes of algorithms
  - I. Progressive algorithms: progressive insertion of sequences into the tree e.g. stepwise addition
  - II. Global algorithms:
    - use NJ or parsimony starting tree
    - optimize tree by application of standard topological alterations
      - NNI: Nearest Neighbor Interchange
      - TBR: Tree Bisection Reconnection
      - Subtree Rearrangements

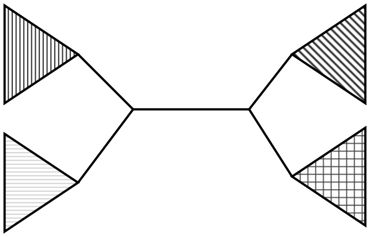
NNI



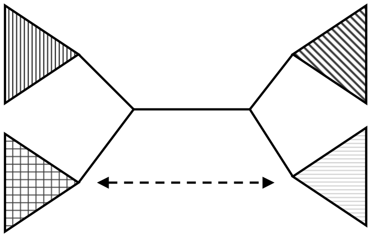
NNI



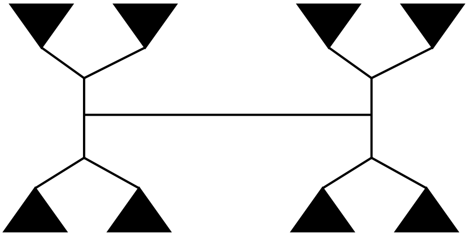
NNI



NNI

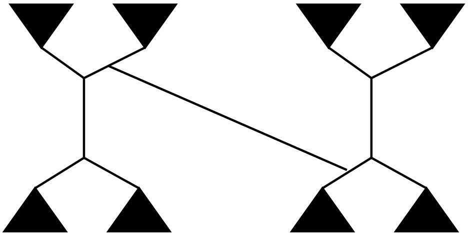


TBR



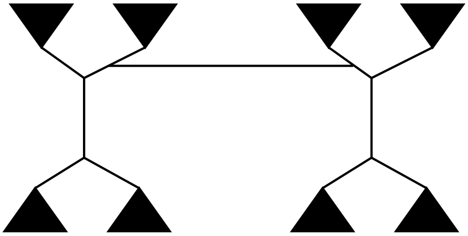
© Thomas Ludwig, Alexandros Stamatakis, GCB'04

TBR



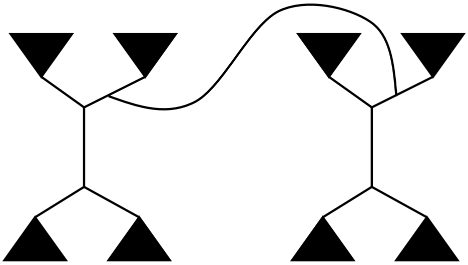
© Thomas Ludwig, Alexandros Stamatakis, GCB'04

TBR



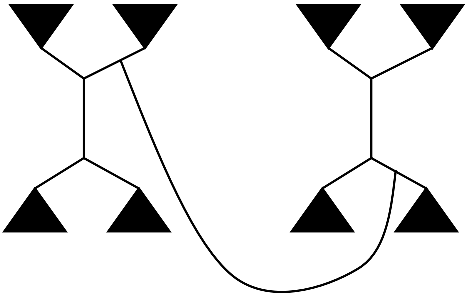
© Thomas Ludwig, Alexandros Stamatakis, GCB'04

TBR



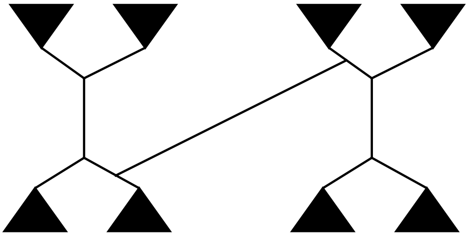
© Thomas Ludwig, Alexandros Stamatakis, GCB'04

TBR



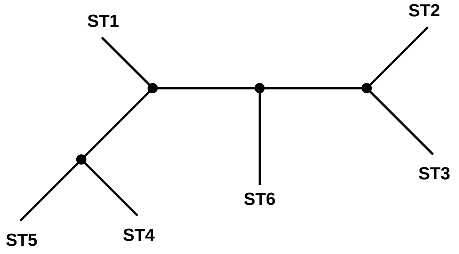
© Thomas Ludwig, Alexandros Stamatakis, GCB'04

TBR

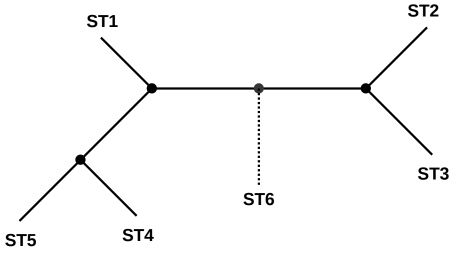


© Thomas Ludwig, Alexandros Stamatakis, GCB'04

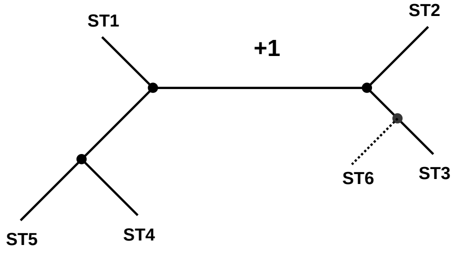
Subtree Rearrangements



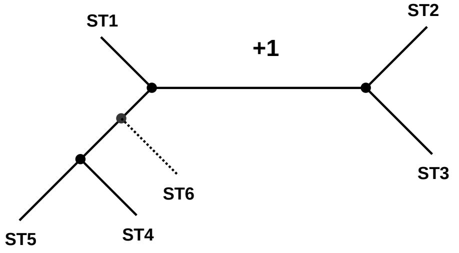
Subtree Rearrangements



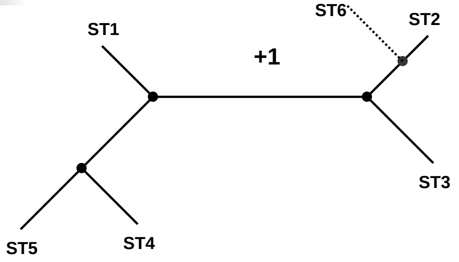
Subtree Rearrangements



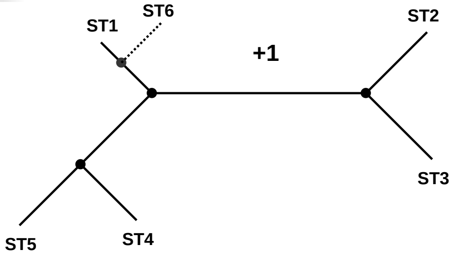
Subtree Rearrangements



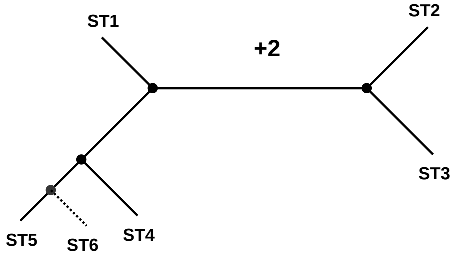
Subtree Rearrangements



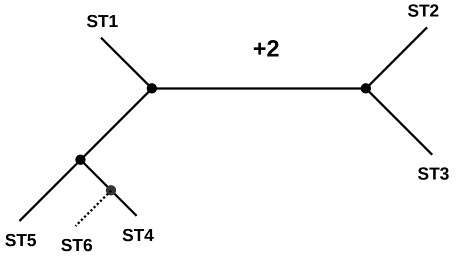
Subtree Rearrangements



Subtree Rearrangements



Subtree Rearrangements



State-of-the-Art sequential phylogeny programs I

- PHYML: fast & accurate on simulated data
  - Guindon et al (2003) A simple, fast, and accurate algorithm to estimate large phylogenies by maximum likelihood.
- RAxML-III: fast & accurate on real data
  - Stamatakis et al (2004) RAxML-III: A fast program for maximum likelihood-based inference of large phylogenetic trees.
- MetaPigA: fastest genetic search algorithm
  - Lemmon et al (2002) The metapopulation genetic algorithm: An efficient solution for the problem of large phylogeny estimation.

State-of-the-Art sequential phylogeny programs II

- IQPNNI: accurate on real & simulated data; slower than PHYML/RAxML
  - Vinh et al (2004) IQPNNI: Moving fast through tree space and stopping in time.
- PAUP*: Many options for MP & ML searches; very slow on ML, not available free of charge
  - Swofford (1998) PAUP* 4.0 - Phylogenetic Analysis Using Parsimony (*and Other Methods).
- MrBayes: fast bayesian inference
  - Huelsenbeck (2001) MrBayes: Bayesian inference of phylogenetic trees.

Performance of Phylogeny Programs

- Quantitative Measures
  - Accuracy
  - Time consumption
  - Memory requirements
- Qualitative Measures
  - Number of implemented evolutionary models
  - Ability to optimize evol. model parameters
  - Availability: negative examples TNT, DCM
  - Code for various platforms

Performance of Phylogeny Programs

- Quantitative Measures
  - Accuracy
  - Time consumption
  - Memory requirements
- Qualitative Measures
  - Number of implemented evolutionary models
  - Ability to optimize evol. model parameters
  - Availability: negative examples TNT, DCM
  - Code for various platforms

Memory consumption is an important
—often underestimated—
problem !

Performance of Phylogeny Programs

- Simulated data
  - generate simulated "true" tree
    - Standard program: r8s
  - generate simulated alignment for the tree
    - Standard program: Seq-Gen
  - compute tree with phylogeny program
  - measure topological distance to true tree
    - Standard measure: Robinson-Foulds distance
- Problems
  - perfect world: no gaps, no sequencing errors
  - evolutionary model known a priori

Performance of Phylogeny Programs

- Real data alignments
  - compute tree with phylogeny programs
  - compare final tree scores
  - significance of small δ between final ML scores
    - apply likelihood ratio tests
    - remember that programs return log-likelihood values
    - High score-accuracy required: 99.99%
- Problems
  - real tree not known
  - evolutionary model not known
  - application to one class of model (ML, MP) only

Performance of Phylogeny Programs

- Real data alignments
  - ompute tree with phylogeny programs
  - npare final tree scores
  - δ between final ML scores
    - tests
    - ams return log-likelihood values
    - equired: 99.99%
- Problems
  - real tree not known
  - evolutionary model not known
  - application to one class of model (ML, MP) only

The optimization of trees for real data is generally significantly harder than for simulated data!

Performance of Phylogeny Programs

- Real data alignments
  - logeny programs
  - res
  - between final ML scores
    - apply like ratio tests
    - remember t programs return log-likelihood values
    - High score-accuracy required: 99.99%
- Problems
  - real tree not known
  - evolutionary model not known
  - application to one class of model (ML, MP) only

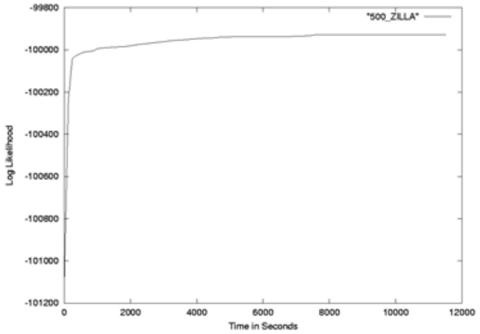
Current research issue: When to stop the analysis?

Performance of Phylogeny Programs

- Real data alignments
  - compute t with phylogeny programs
  - cores
  - δ between final ML scores
    - tests
    - grams return log-likelihood values
    - High score-accuracy required: 99.99%
- Problems
  - real tree not known
  - evolutionary model not known
  - application to one class of model (ML, MP) only

Current issue: Standard real-data benchmark set required!

When to stop the analysis?



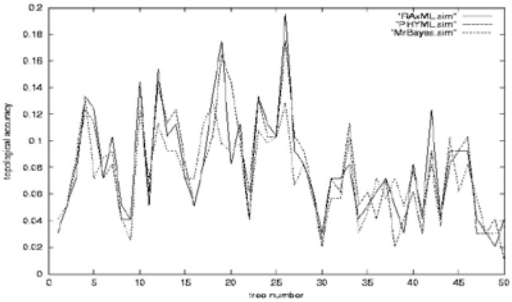
When to stop the analysis?

Is this improvement worth the extra time?

Survey of sequential Programs

- Comparison on small simulated alignemnts
  - Williams et al (2003) An investigation of phylogenetic likelihood methods.
- Evaluated RAxML, PHYML, MrBayes on 50 simulated alignments with 100 taxa
- Used 9 real-world alignments with 101-1000 taxa
- Results:
  - RAxML best & fastest on real data
  - MrBayes best on simulated data
  - MrBayes significantly slower than PHYML, RAxML

Sequential Results: Simulated Data



PHYML: 0.0796 / 35.21 secs **RAxML: 0.0808 / 131.05 secs**
MrBayes: 0.0741 / 945.32 secs **RAxML: 0.0818 / 29.27 secs**

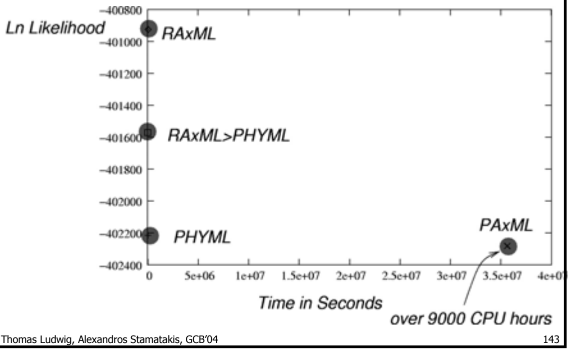
Sequential Results: Real Data

data	PHYML	secs	MrBayes	secs	RAxML	secs	R > PHY secs	PAXML	hrs
101_SC	-74097.6	153	-77191.5	40527	-73919.3	617	31	-73975.9	47
150_SC	-44298.1	158	-52028.4	49427	-44142.6	390	33	-44146.9	164
150_ARB	-77219.7	313	-77196.7	29383	-77189.7	178	67	-77189.8	300
200_ARB	-104826.5	477	-104856.4	156419	-104742.6	272	99	-104743.3	775
250_ARB	-131560.3	787	-133238.3	158418	-131468.0	1067	249	-131469.0	1947
500_ARB	-253354.2	2235	-263217.8	366496	-252499.4	26124	493	-252588.1	7372
1000_ARB	-402215.0	16594	-459392.4	509148	-400925.3	50729	1893	-402282.1	9898
218_RDPII	-157923.1	403	-158911.6	138453	-157526.0	6774	244	n/a	n/a

Sequential Results: Real Data

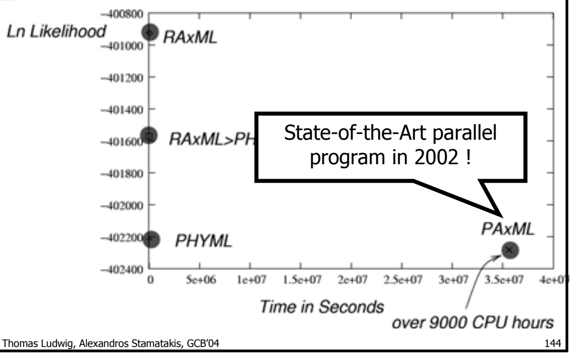
data	PHYML	secs	MrBayes	secs	RAxML	secs	R > PHY secs	PAXML	hrs
101_SC	-74097.6	153	-77191.5	40527	-73919.3	617	31	-73975.9	47
150_SC	-44298.1	158	-52028.4	49427	-44142.6	390	33	-44146.9	164
150_ARB	-77219.7	313	-77196.7	29383	-77189.7	178	67	-77189.8	300
200_ARB	-104826.5	477	-104856.4	156419	-104742.6	272	99	-104743.3	775
250_ARB	-131560.3	787	-133238.3	158418	-131468.0	1067	249	-131469.0	1947
500_ARB	-253354.2	2235	-263217.8	366496	-252499.4	26124	493	-252588.1	7372
1000_ARB	-402215.0	16594	-459392.4	509148	-400925.3	50729	1893	-402282.1	9898
218_RDPII	-157923.1	403	-158911.6	138453	-157526.0	6774	244	n/a	n/a

Sequential Results: Real Data



over 9000 CPU hours

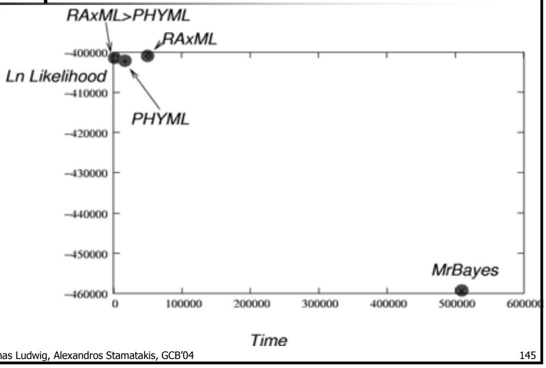
Sequential Results: Real Data



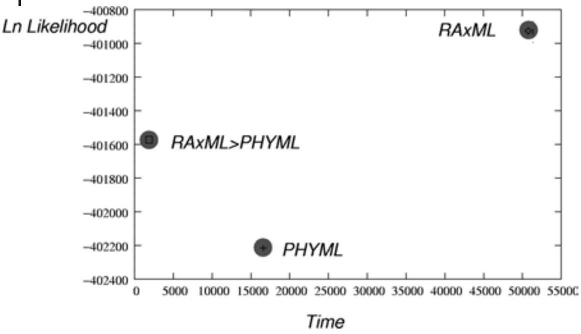
State-of-the-Art parallel program in 2002 !

over 9000 CPU hours

Sequential Results: Real Data



Sequential Results: Real Data



Memory requirements

	1000 taxa	10000 taxa
RAxML	200MB	750MB
PHYML	900MB	8.8GB
MrBayes	1150MB	not available

Sequential RAxML

Compute randomized parsimony starting tree with dnapars from PHYLIP

Advantage of RAxML: search starts from distinct points in search space

Sequential RAxML

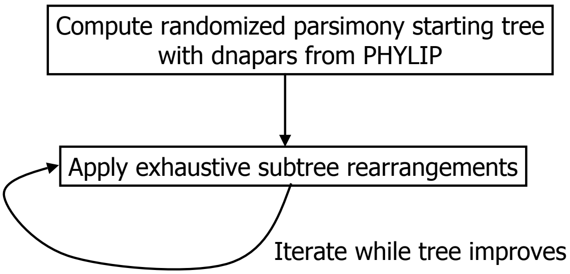
Compute randomized parsimony starting tree with dnapars from PHYLIP

↓

Apply exhaustive subtree rearrangements

RAxML performs fast lazy rearrangements

Sequential RAxML

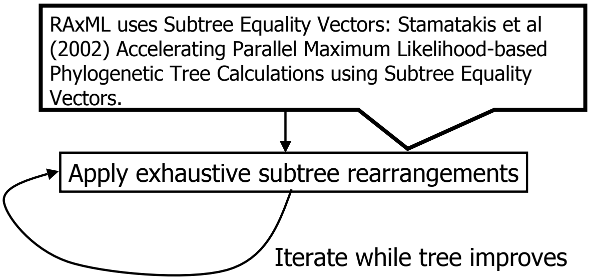


Compute randomized parsimony starting tree with dnapars from PHYLIP

Apply exhaustive subtree rearrangements

Iterate while tree improves

Sequential RAxML



RAxML uses Subtree Equality Vectors: Stamatakis et al (2002) Accelerating Parallel Maximum Likelihood-based Phylogenetic Tree Calculations using Subtree Equality Vectors.

Apply exhaustive subtree rearrangements

Iterate while tree improves

Outline

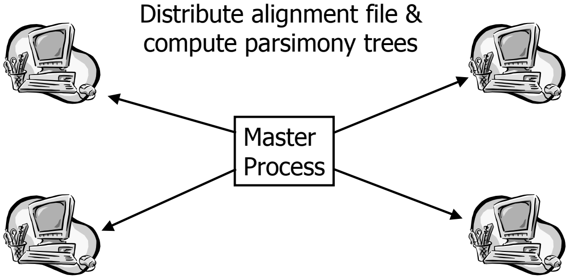
- Grand Challenges in HPC Bioinformatics
- HPC Bioinformatics by Example of Phylogenetic Inference
  - phylogenetic analysis
  - maximum likelihood
  - bayesian inference
  - sequential codes
  - parallel & distributed RAxML
  - future developments

Parallel & Distributed RAxML

- Design goals
  - minimize communication overhead
  - attain good speedup
- Master-Worker architecture
- 2 computational phases
  1. Computation of # workers parsimony trees
  2. Rearrangement of subtrees at each worker
- Program is non-deterministic → every run yields distinct result, even with fixed starting tree

---

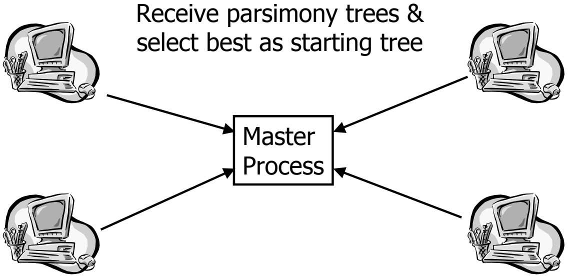
Parallel RAxML: Phase I



Distribute alignment file & compute parsimony trees

Master Process

Parallel RAxML: Phase I

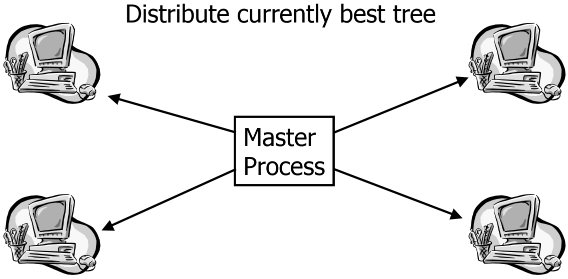


Receive parsimony trees & select best as starting tree

Master Process

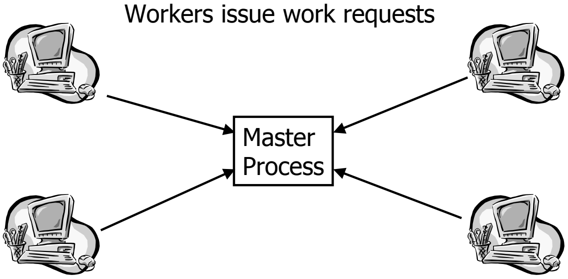
Parallel RAxML: Phase II

Distribute currently best tree


Master
Process

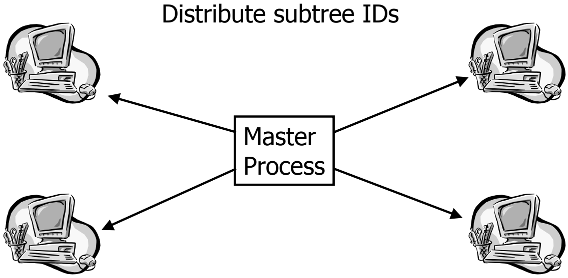
Parallel RAxML: Phase II

Workers issue work requests


Master
Process

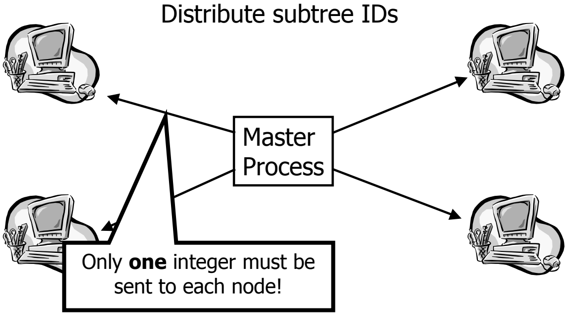
Parallel RAxML: Phase II

Distribute subtree IDs


Master
Process

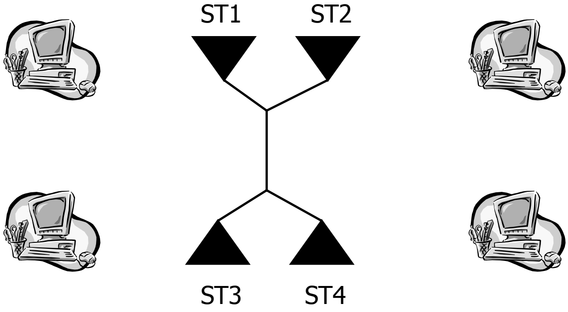
Parallel RAxML: Phase II

Distribute subtree IDs

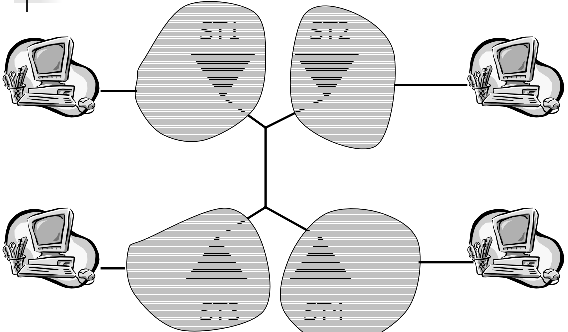

Master
Process

Only **one** integer must be sent to each node!

Parallel RAxML: Phase II

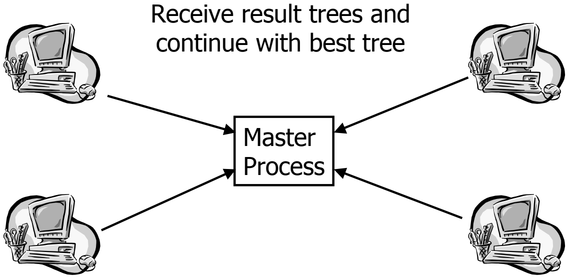


Parallel RAxML: Phase II



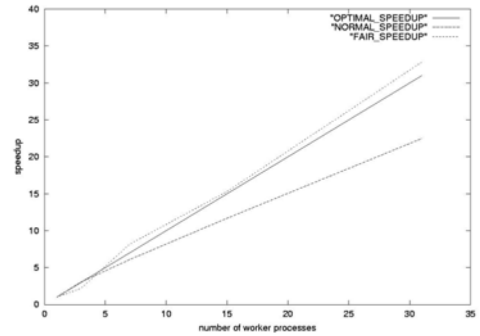
Parallel RAxML: Phase II

Receive result trees and continue with best tree

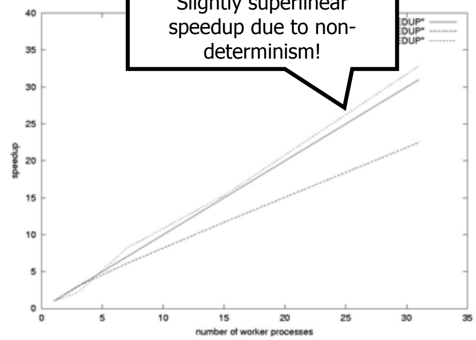


Master Process

Speedup



Speedup



Slightly superlinear speedup due to non-determinism!

RAxML: Biological Research

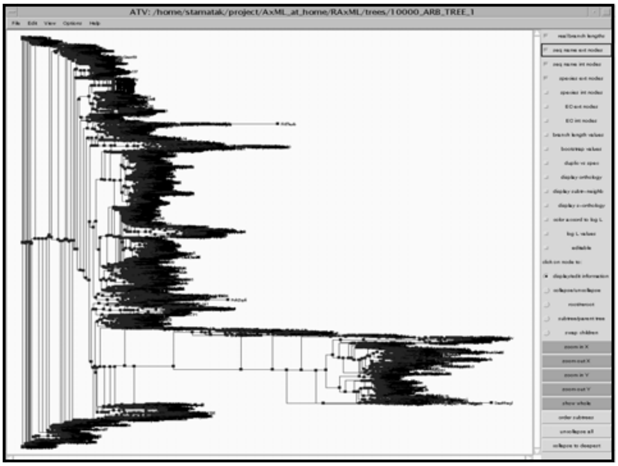
1. Parallel inference of 5 10.000 taxon trees containing Bacteria, Eukarya, Archaea on a Linux PC Cluster
    - Accumulated CPU hours per tree ≈ 3200
    - Largest ML-analysis to date
    - Major clades correctly identified
2. Sequential analysis of 2415 mammals (cytochrome-b sequences)
    - "Traditional" reference tree available
    - Error about 10-13%

Outline

- Grand Challenges in HPC Bioinformatics
- HPC Bioinformatics by Example of Phylogenetic Inference
    - phylogenetic analysis
    - maximum likelihood
    - bayesian inference
    - sequential codes
    - parallel & distributed RAxML
    - future developments

Future Developments

- Visualization


ATV: /home/stamatak/project/AxML_at_home/RAxML/trees/10000_ARB_TREE_1

Future Developments

- Visualization
- Divide-and-conquer algorithms
  I. Divide alignment into sub-alignments
  II. Infer subtrees for sub-alignments
  III. Merge subtrees into comprehensive tree by application of supertree methods
  IV. Resolve multifurcations & optimize supertree

Fut ts

Problem 1: Currently only 2 methods available for alignment division:
RAxML & DCM

- Visuali
- Divid -and-conquer algorithms
  I. Divide alignment into sub-alignments
  II. Infer subtrees for sub-alignments
  III. Merge subtrees into comprehensive tree by application of supertree methods
  IV. Resolve multifurcations & optimize supertree

Future Developments

- Visualization
- gorithms
  ub-alignments
  alignments
  mprehensive tree by
  plication of supertree methods
  IV. Resolve multifurcations & optimize supertree

Problem 2:Resolving multifurcations is hard optimization of the entire tree required

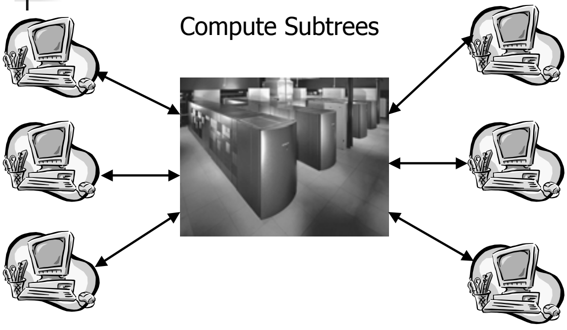
---

Distributed D & C

Construct Guide Tree & perform tree-based alignment divison

Distributed D & C

Compute Subtrees

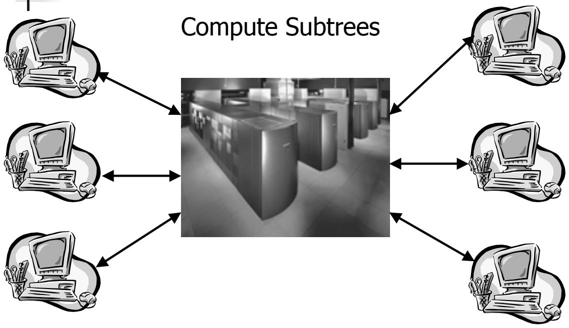


Distributed D & C

Merge into Guide Tree & Re-Divide Alignment

Distributed D & C

Compute Subtrees



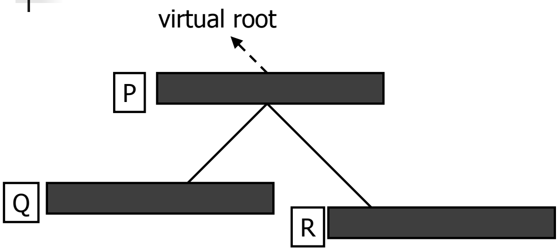
Recent Work on D & C

- Rec-I-DCM3: Very fast on parsimony
  - Roshan (2004) Rec-I-DCM3: A fast algorithmic technique for reconstructing large phylogenetic trees.
- PhyNav: Zoom-in zoom-out technique
  - Vinh et al (2004) PhyNav: A novel approach to reconstruct large phylogenies.
- BWD: New supertree reconstruction method using distances
  - Stephen J. Willson (2004) Constructing rooted supertrees using distances.

Future Developments

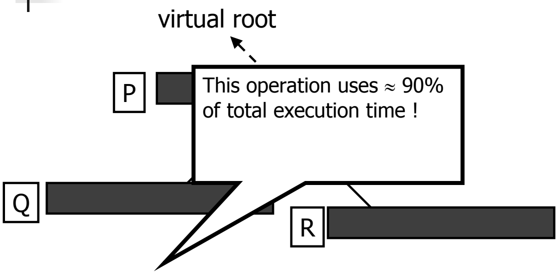
- Visualization
- Divide-and-conquer algorithms
  - I. Divide alignment into sub-alignments
  - II. Infer subtrees for sub-alignments
  - III. Merge subtrees into comprehensive tree by application of supertree methods
  - IV. Resolve multifurcations & optimize supertree
- Shared memory or vector processors

Shared Memory Parallelism



P[i] = f(g(Q[i]) , g(R[i]))

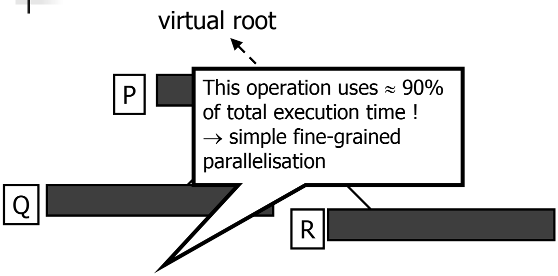
Shared Memory Parallelism



This operation uses ≈ 90% of total execution time !

P[i] = f(g(Q[i]) , g(R[i]))

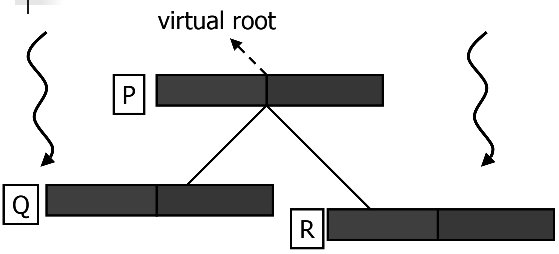
Shared Memory Parallelism



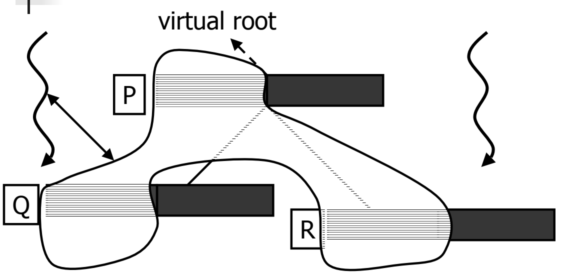
This operation uses ≈ 90% of total execution time !
→ simple fine-grained parallelisation

P[i] = f(g(Q[i]) , g(R[i]))

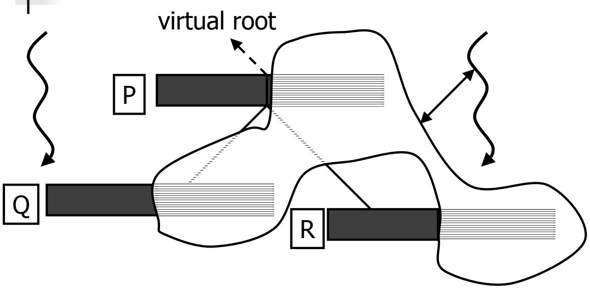
Shared Memory Parallelism



Shared Memory Parallelism



Shared Memory Parallelism



Future Developments

- Visualization
- Divide-and-conquer algorithms
  - I. Divide alignment into sub-alignments
  - II. Infer subtrees for sub-alignments
  - III. Merge subtrees into comprehensive tree by application of supertree methods
  - IV. Resolve multifurcations & optimize supertree
- Shared memory or vector processors
- Combination of ML hill-climbing and MC³-like sampling algorithms

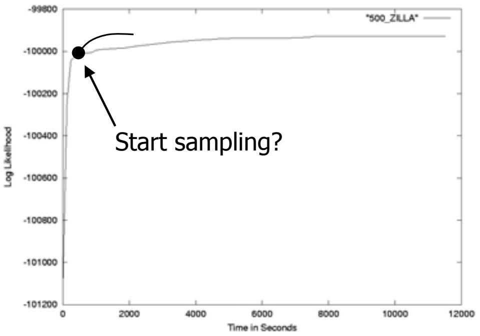
---

Future Developments

- Visualization
- Divid
  - I. Div ents
  - II. Inf s
  - III. Me ve tree by ap
  - IV. Resolve arcations & optimize supertree
- Shared emory or vector processors
- Combination of ML hill-climbing and MC³-like sampling algorithms

Potential Advantages:
1. Large number of sampled trees to build consensus tree
2. Faster convergence to best-known likelihood?

Combined method



Start sampling?

Recent HW-specific optimization

- Like every ML program RAxML makes many calls to exp() and log() in vectors → expensive
- Utilisation of Intel™ MKL (Math Kernel Library) on a Xeon 2.4GHz CPU
- Performance boost of over 30% …
- … in half a day of work

On-Line Resources

HPC Resources

- TOP500 List
  www.top500.org
- BlueGene
  www.research.ibm.com/bluegene/
- EarthSimulator
  www.es.jamstec.go.jp
- MPI-Forum
  www.mpi-forum.org
- Global Grid Form
  www.gridforum.org
- MPICH
  www-unix.mcs.anl.gov/mpi/mpich/
- OpenMP
  www.openmp.org

Bioinformatics Resources

- CIPRES project
  www.phylo.org
- Felsenatein's phylogeny page
  evolution.genetics.washington.edu/phylip/software.html
- PHYML download
  www.lirmm.fr/~guindon/phyml.html
- RAxML download
  wwwbode.cs.tum.edu/~stamatak/research.html
- IQPNNI & PhyNav download
  www.bi.uni-duesseldorf.de/software