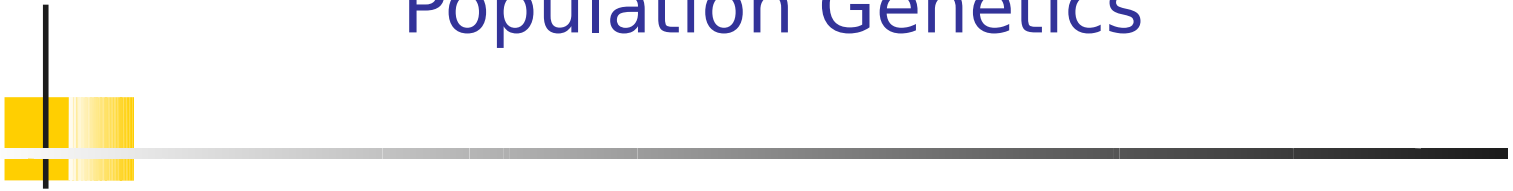
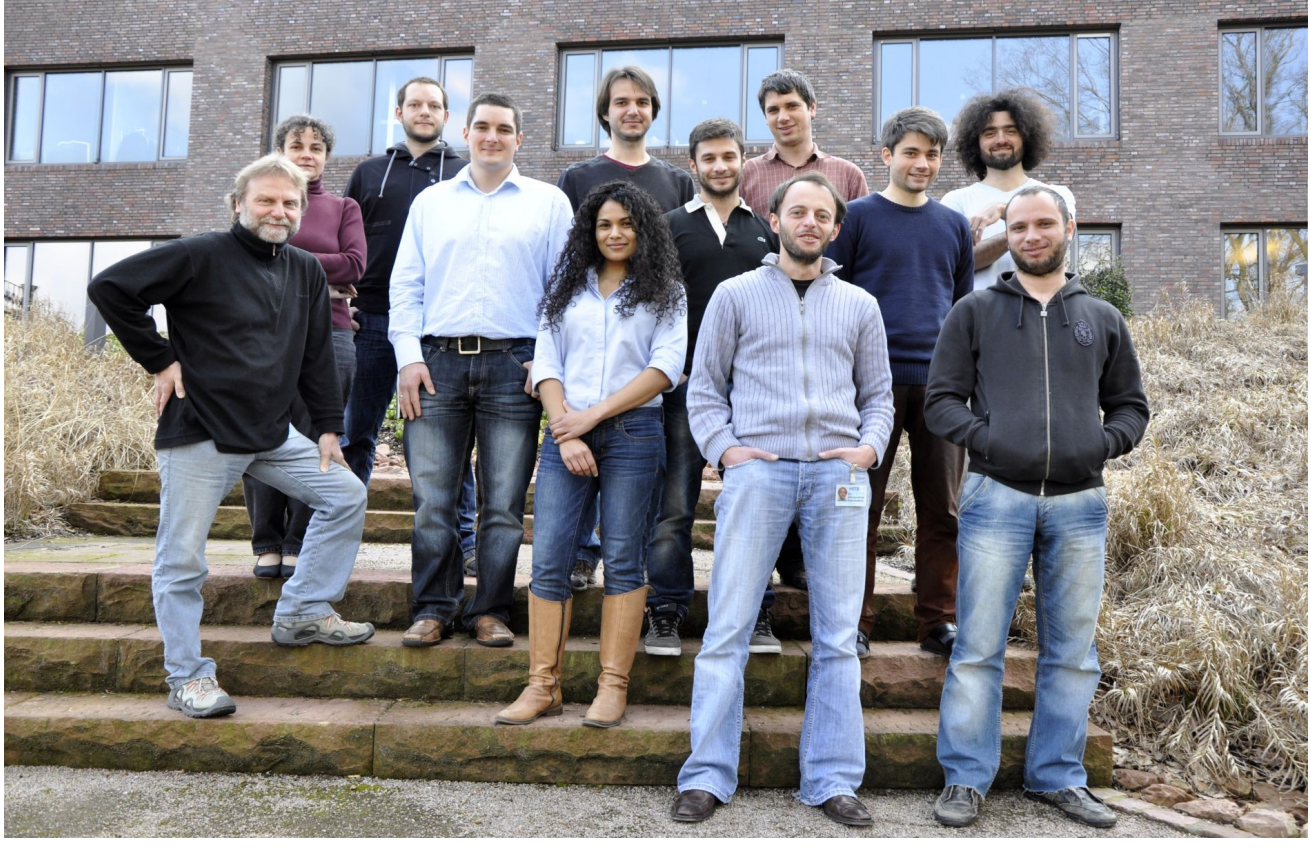
High Performance Phylogenetics & Population Genetics



Alexandros Stamatakis

Scientific Computing Group
Heidelberg Institute for Theoretical Studies

Alexandros.Stamatakis@h-its.org
www.exelixis-lab.org



- IT
  - Cluster with 2500 cores
  - Archiving of Scientific Data
  - 90 TB parallel I/O system
  - All other IT services
- Research
  - Method development
  - SW development
  - Emerging Parallel Architectures

Please ask Questions!

Why is this important?

- Datasets are growing
- Need to use supercomputers
- Transformation into a computational science
- Software/methods are widely used when they are fast
- Software/algorithm engineering is required

However

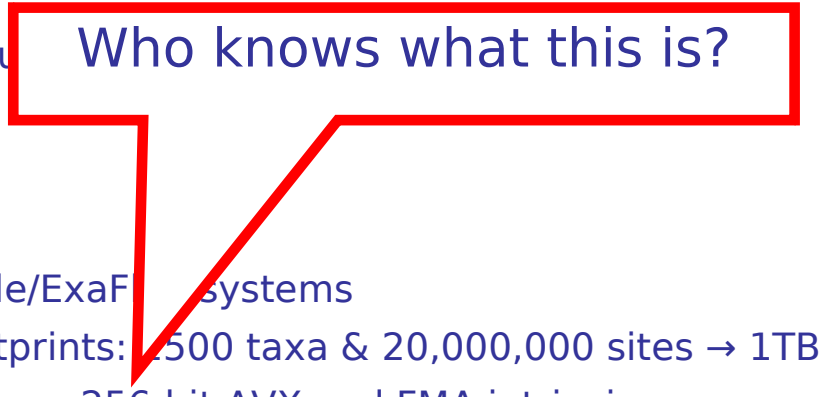
- Smarter algorithms are always better than brute-force HPC approaches

Toward a Computational Science

- **Example** 1Kite Project www.1kite.org to sequence 1000 insect transcriptomes
- **Challenges**
  - Data transfer
  - Data storage
  - Code correctness
  - Supercomputer resources
  - CO^2 footprints
- **Software Properties**
  - Checkpointing
  - Scalability → Exascale/ExaFlop systems
  - Reduce memory footprints: 1500 taxa & 20,000,000 sites → 1TB
  - Low-level optimizations: 256-bit AVX and FMA intrinsics
  - Load Balance Issues

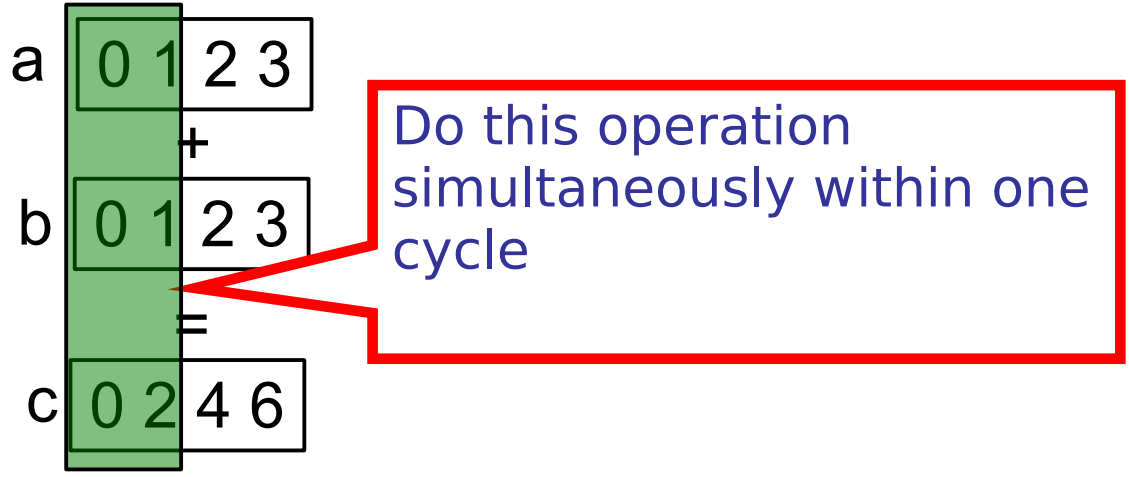
Toward a Computational Science

- **Example** 1Kite Project www.1kite.org to sequence 1000 insect transcriptomes
- **Challenges**
  - Data transfer
  - Data storage
  - Code correctness
  - Supercomputer resou
  - CO^2 footprints
- **Software Properties**
  - Checkpointing
  - Scalability → Exascale/ExaF systems
  - Reduce memory footprints: 2500 taxa & 20,000,000 sites → 1TB
  - Low-level optimizations: 256-bit AVX and FMA intrinsics
  - Load Balance Issues



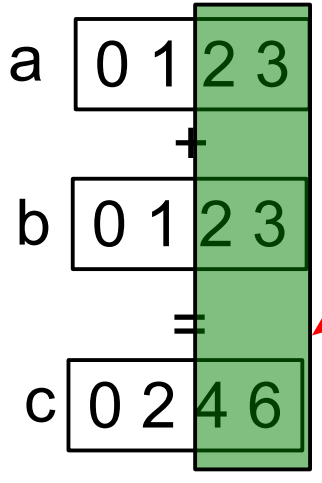
Vector Instructions

- e.g. RAxML SSE3/AVX versions
- A clock tick: execute one instruction
- 2.2 GHz 2.2 * 10^9 instructions per second



Vector Instructions

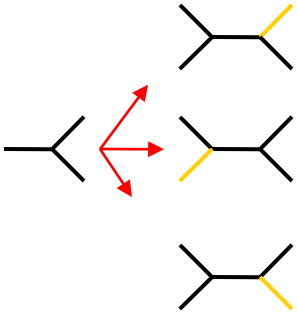
- e.g. RAxML SSE3/AVX versions
- A clock tick: execute one instruction
- 2.2 GHz 2.2 * 10^9 instructions per second



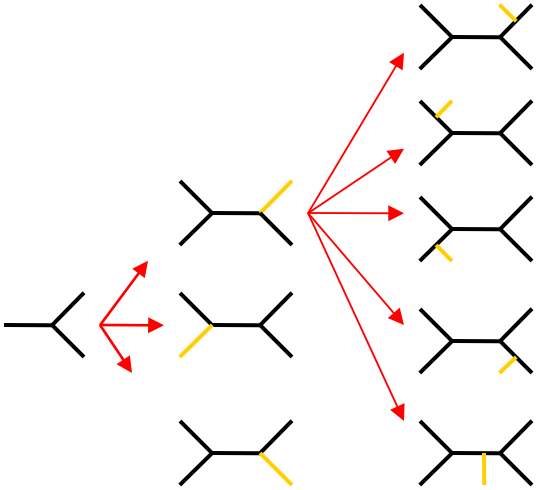
... and this operation simultaneously within one cycle: only two clock cycles (ticks) required

The algorithmic problem

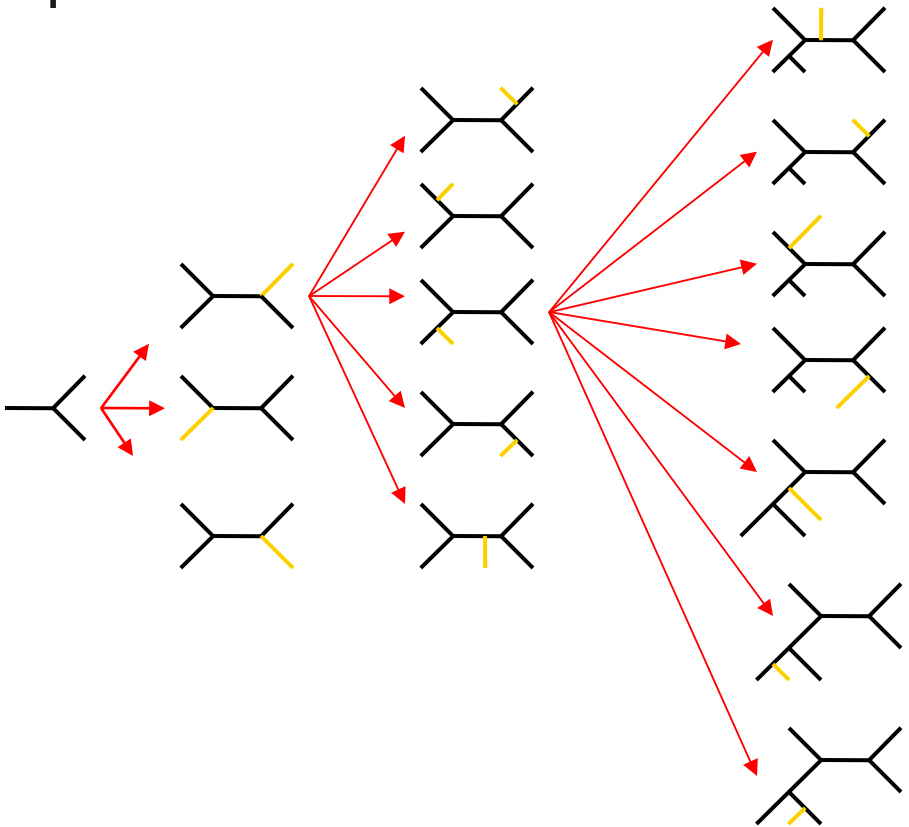
The number of trees



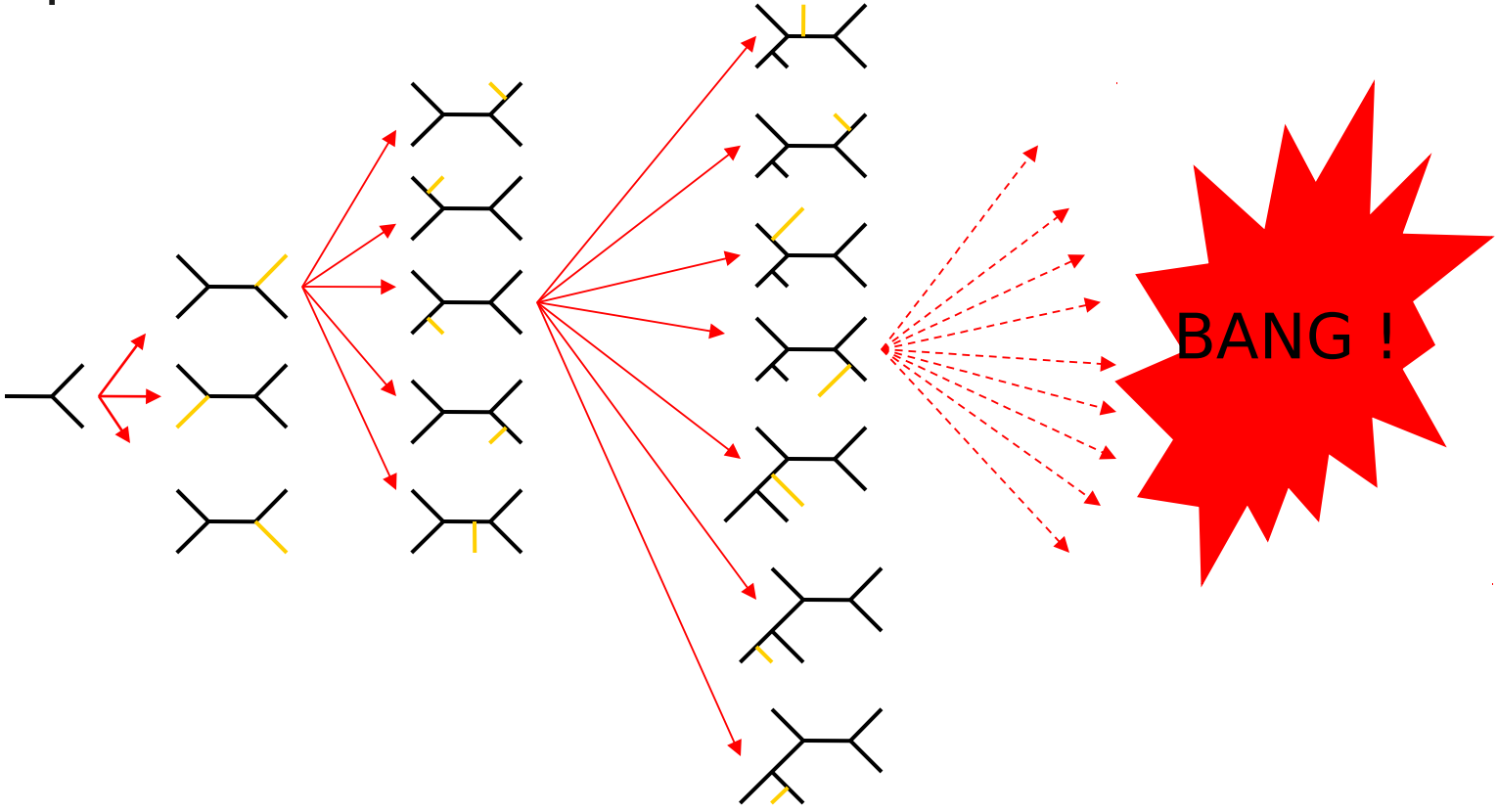
The number of trees



The number of trees



The number of trees explodes!



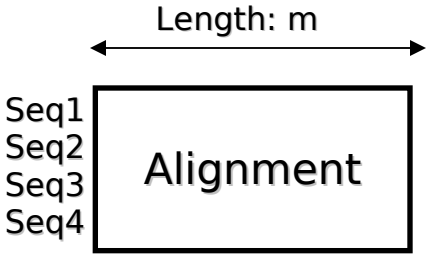
trees with 2000 tips

```
stamatak@exelixis:~/Desktop/GIT/TreeCounter$ ./treeCounter -n 2000


GNU GPL tree number calculator released June 2011 by Alexandros Stamatakis

Number of unrooted binary trees for 2000 taxa: 3004963817421165615163291006568181498137723207423701308950495404301263652525830821082768599668824700046435273521426563428829589150234460006314939691306329704360561848618774654822779912235368092334555631999108345976931267565250128998674331877528114019609916315223670306091217357097623798477054676677795324797182614385273338226727784250737252849916669687584403510579587020686505817687044666318123742901021438506432471360934491667021135969756940300666252646479269124551031494236619554282411827762511484875825458122791428980113264890267403376129471274576703626757908684316966071860984794181886595721455704474457228866172905358352074425368812312401066131569488619609411956467362003425752413352775750858291610964225757276997679914082833432101613274016528309938039045923276906900359729197099407393495634862038990107426872822975974655377102257672676842858011877224950106218117340523208265397342962227352536590515865631383272031119841987467599738646318290320383252308597997992216101227215780580524814583120684401676062393060097116167297155047284877996343375313489942303724373478791319890859537640701348494461138775725769524087024617201078742973804622750525457066893723194182064407068918840038705902897721975164544959758216621306205064617761099485663734168183584989329076993382067801052437284614924034229611551826097782286191926720712951895893600995913097423307231638251842811033057101744115688430513186587754437630850031145111072383703970746518223204040615470827307862995754933103127520861670066079129801426223005651235227180638195093358726517286235890205200161443617560756428647142216613004434807084067501589247673166341539540575074474994909831496473031080411401891849735912811228378774049888483405621024205664244638600938996508574296194726905430152812375265109658152846997970367921711290355680981807916958795161415928104952817985584729253444786442443599808531537204796814969465991768614533701051985928577157482455943377242369582576242663016946320482495182255939287403177623433881048604630975191556923871167513095213415098816715464307862352606237786406838680424690252749113931927680261151599058260388673317293071367390340361863746398060576483647467027444672788088533707425442192272667774700332940332010382880351126890262551830967919483586789293701637681753048206338943871497931152353698229625111163071482945992116208033026847620133356904410896681454361509051558775811679770012563912151116237444170497371704604029481104114822286466131918821997571383368352072526055202769823974613218495249264897050790398360256255606289852288839561357874156576648899926087328661263064254326024897922911356007164057398451637524523376943755857384725545564397599604255916404112221144755235573176239973057747183956531217416532295986675901294116123924072209325036967312488449155375921065065601541672077415923624086866767534828651296488873905970757880247339346347084815901163977279774748041731626870091672873505612164226846816068319895980126037648561531278161168958721512312330876006347338109725311842339640390937378395066835578735307886358646400563299499490631187424029092779272693300322445377595797224873456891511458557078385054168166766742581130195806362190750079029503108820909727174813643698947397107993277770067630173061756653873972603777717300844134394051236690555449324861650825399577950363267049478442934988531727973481777971465671751511788763964340693324580763461107342143281950499096808074207397688914704517472055439689639666874260147724189469321290245331733418828677319465354411330210086657508171324034264758048921862366346160795594372051639515409694980648624230979694721116966596158004178839922322646284989423522769263911243607675089976717896834593378907423463545577193530665615379900277916265636361861974048590938234062235459769733137213659343717585590664439646132830011367260193406870644233948991925130438528154165963011854942363486352458574664283460906291627956492658472300360855559898976191629324814009459248989946846886232258601705514689056498372760039274870695504637888197416994290491028531804107765160072633847216389037822700138435995997302657271988043246643123597585552171969605139210102265963247887830977405333131551762978152880718652603176327264828094499370456258099938053058497699570080289379808014902901005293898472279947167804821689424159118284257696464786505731532517830233630729825169221034658426589444746491612385468971850796817290913903218283411118482138476772831654865321231738200413199051051896702220188704958568718050959073036060934037291762633308898996258780062800956094441693237794495544103369658626155625601066939030320387897098367386087056641433585106111658314520424513208508589994932364831689671194951671619567622707090697388958885557956246664153656172354930180739400476052980172177139168678800027785196617307006128451730758250373564310206511244373082522962504045316059074134388187256347791383066059093188025223100853401768402614015396169891920751471080337577088497401418345997539720598786820648791160649698581776011539720584982226989071813494326918018211733188063653910893689811714891357456680542807485170175858266639633570189354449832669762835092657922201746372190273119641751489944010079636876017826747107019945473218887832742608896672437157471342060009370425130989363053745978427998040313298941726649229042573095836853441621564055729028206622400386323752638091023326989783886042375962560156797526269507986398681042948323331602672165551781208992646778049357413263871374084238855465338361586434513054396242813972795597259951107063143059926154956229583202327080576811566904895866105220300573725298472118747827136676085866927109487556397485848947591081972703387824439864486743456200958161930314727345961900499318424337975243662489363321244850597199252366852924930534625276413785341320894312890152373809255604598709091276666232967870332888205913494958007407447314338880072453232174730965974196711444145312713279020510100476710143506388579534784472553898015419233170275198961806351526825431731938329258919315301641305489723111286664654929719304792964328295567190928816920910423341220074542420499008726585046208051104875883059495990311188736668509414882172573457635523396403848131821316740835900691640005326225818478376506780445117771732865818989921535830944776535041796875

Approximately 3.00 times 10^6328
```

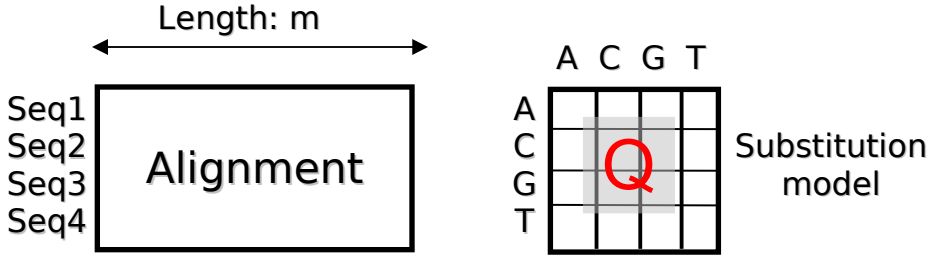
Outline

- Computing the Likelihood
- Optimizing & Parallelizing Likelihood Computations
- Saving Memory in Likelihood Computations
- HPC population genetics

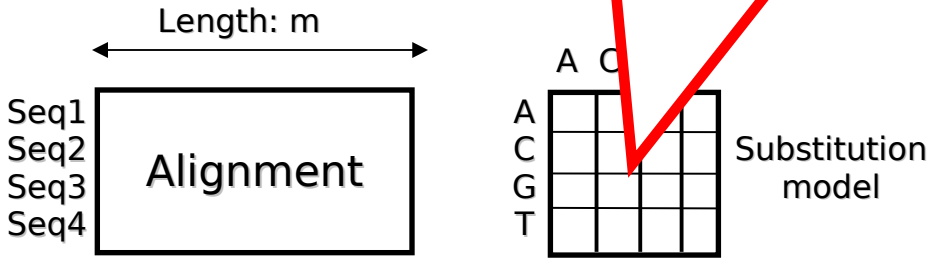
Maximum Likelihood



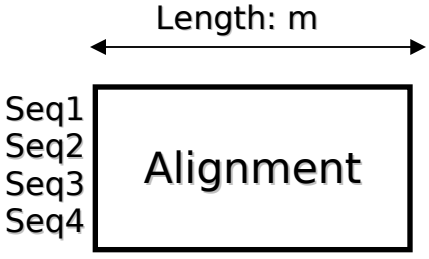
Maximum Likelihood

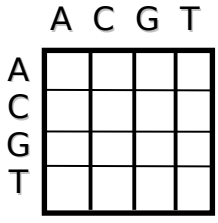


Maxi od


Length: m
Seq1
Seq2
Seq3
Seq4
Alignment
A C
A
C
G
T
Substitution model
P(t) = e^Qt
Important for load balance!

Maximum Likelihood



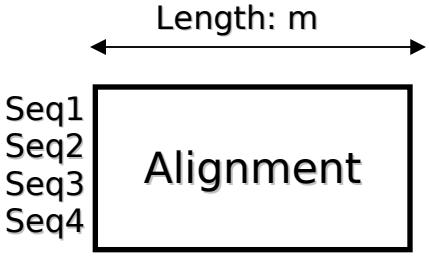


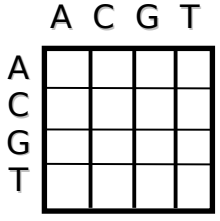
Substitution model

Prior probabilities,
Empirical base frequencies

π_A π_C π_G π_T

Maximum Likelihood

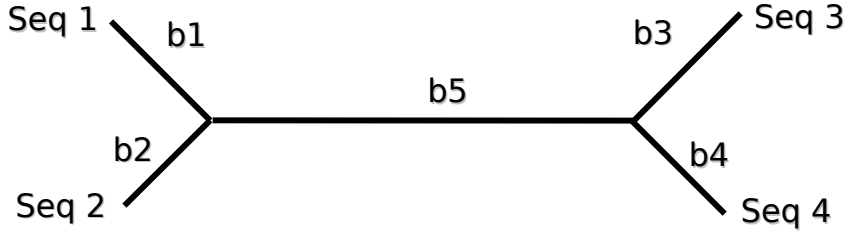




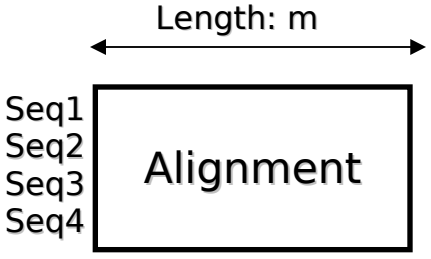
Substitution model

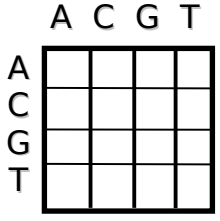
Prior probabilities,
Empirical base frequencies

π_A π_C π_G π_T



Maximum Likelihood

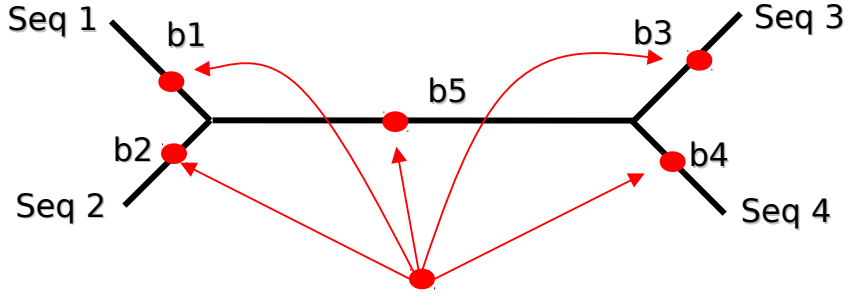




Substitution model

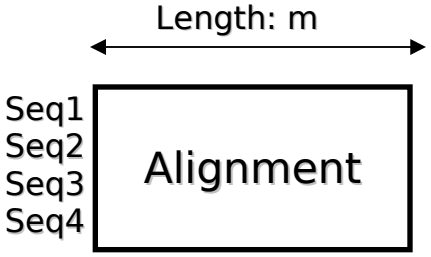
Prior probabilities,
Empirical base frequencies

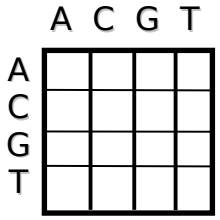
π_A π_C π_G π_T



virtual root: vr

Maximum Likelihood

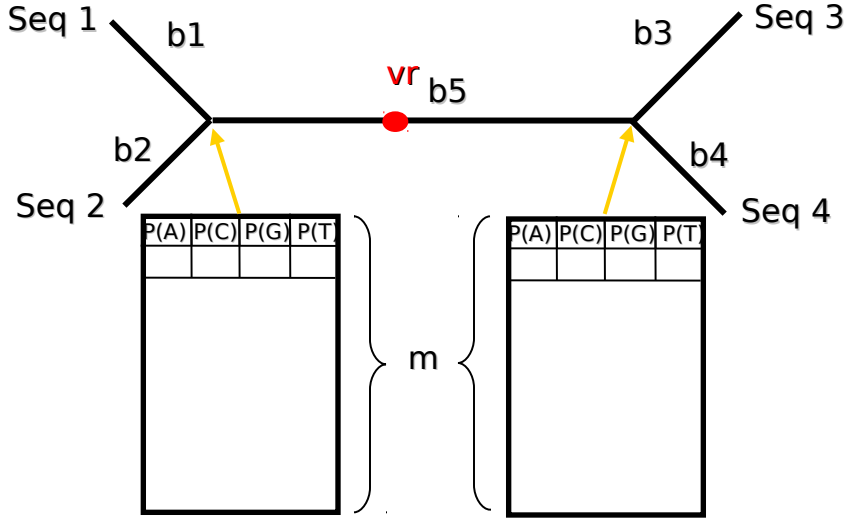




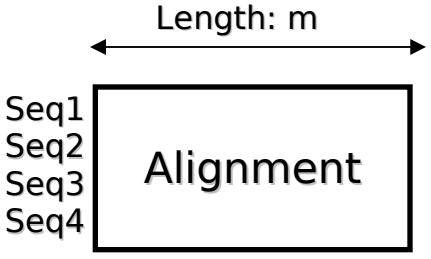
Substitution model

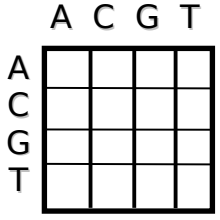
Prior probabilities,
Empirical base frequencies

π_A π_C π_G π_T



Maximum Likelihood

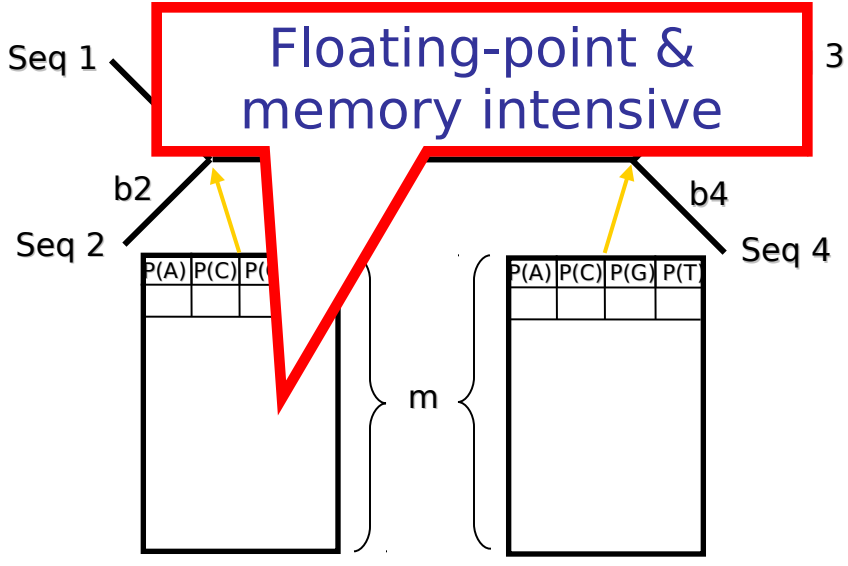




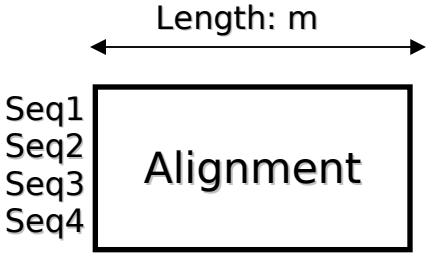
Substitution model

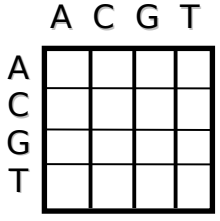
Prior probabilities,
Empirical base frequencies

π_A π_C π_G π_T



Maximum Likelihood

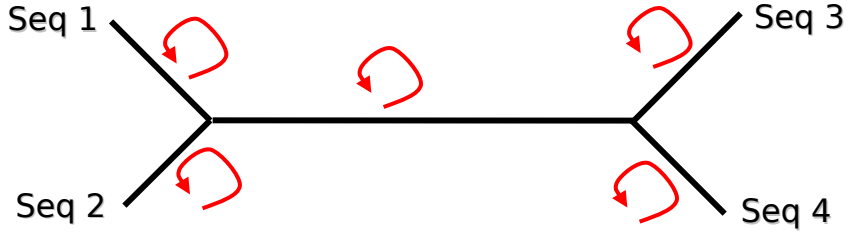




Substitution model

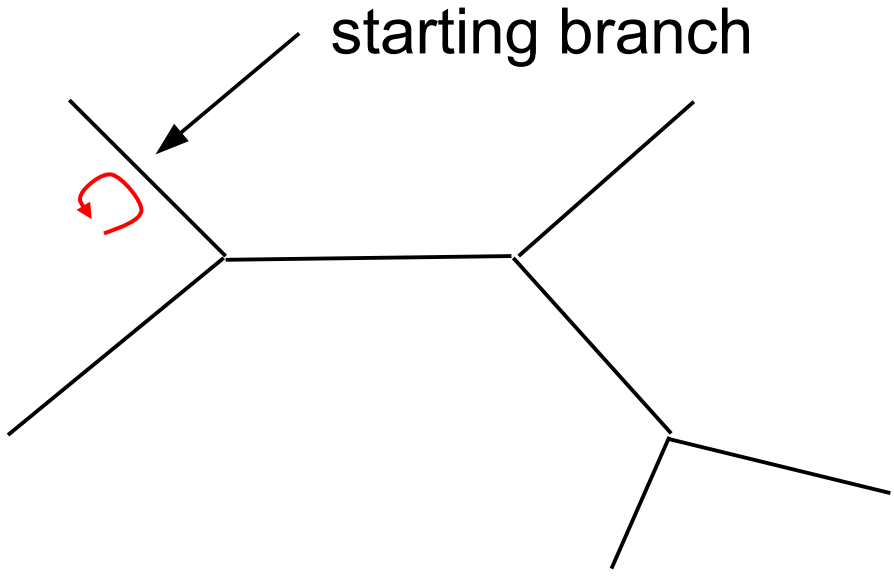
Prior probabilities,
Empirical base frequencies

π_A π_C π_G π_T

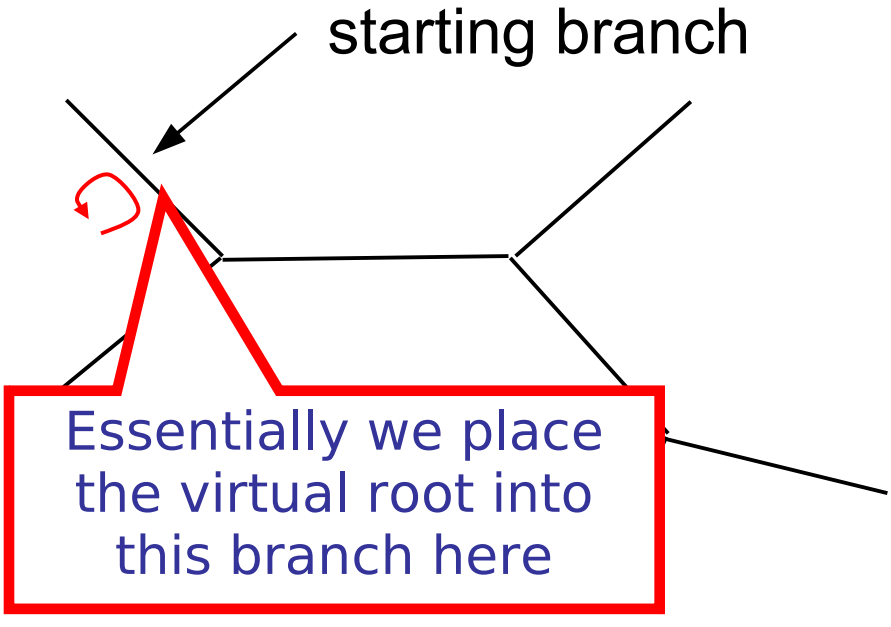


optimize branch lengths

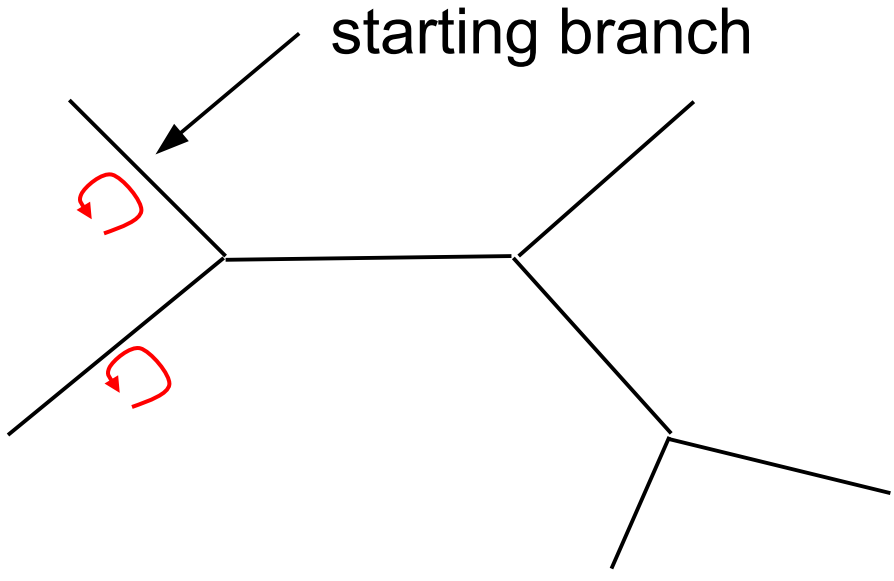
Branch Length Optimization



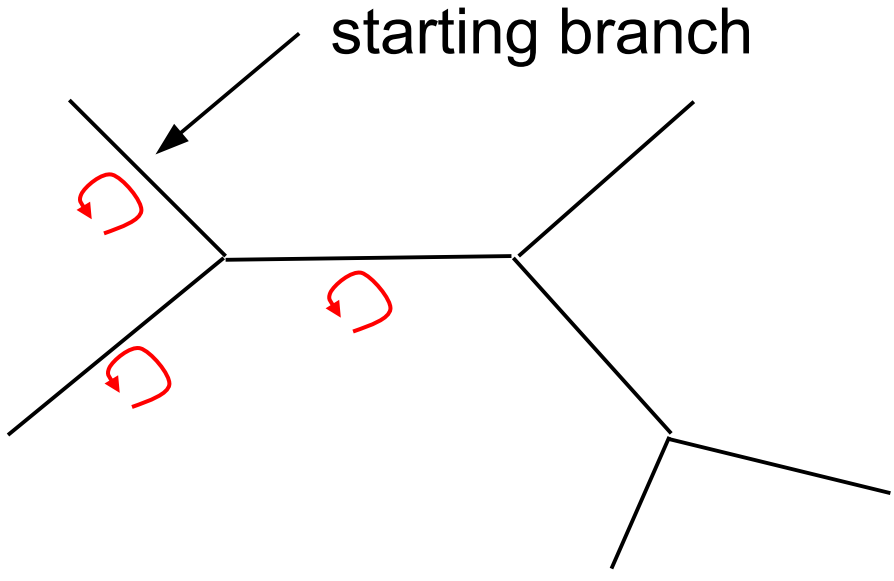
Branch Length Optimization



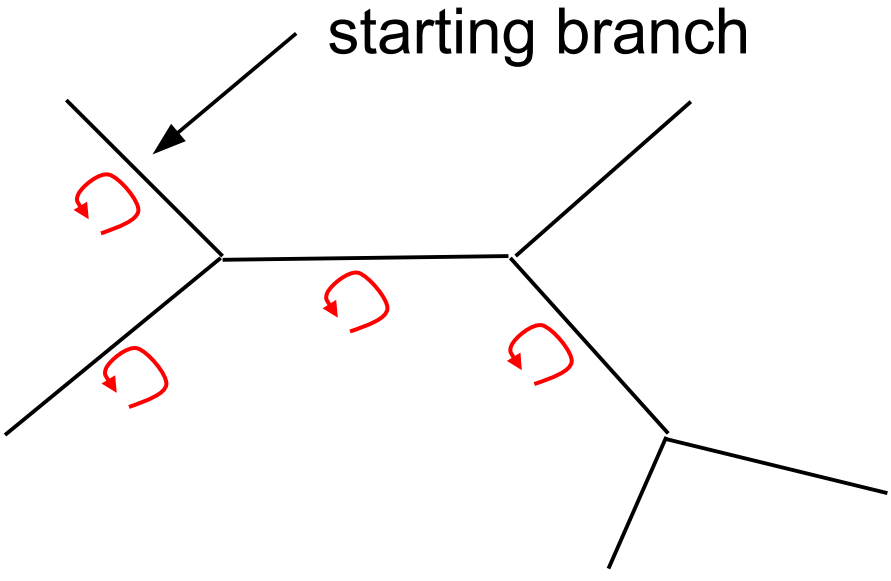
Branch Length Optimization



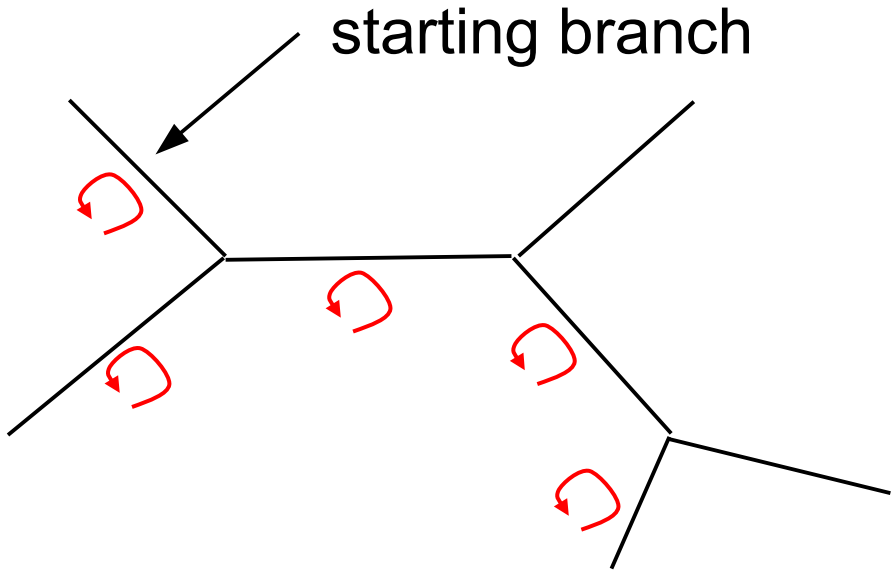
Branch Length Optimization



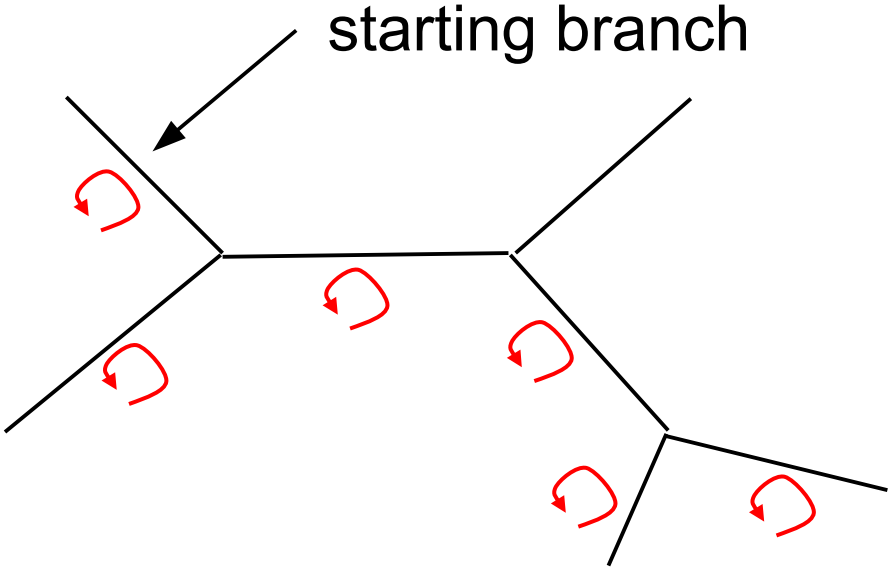
Branch Length Optimization



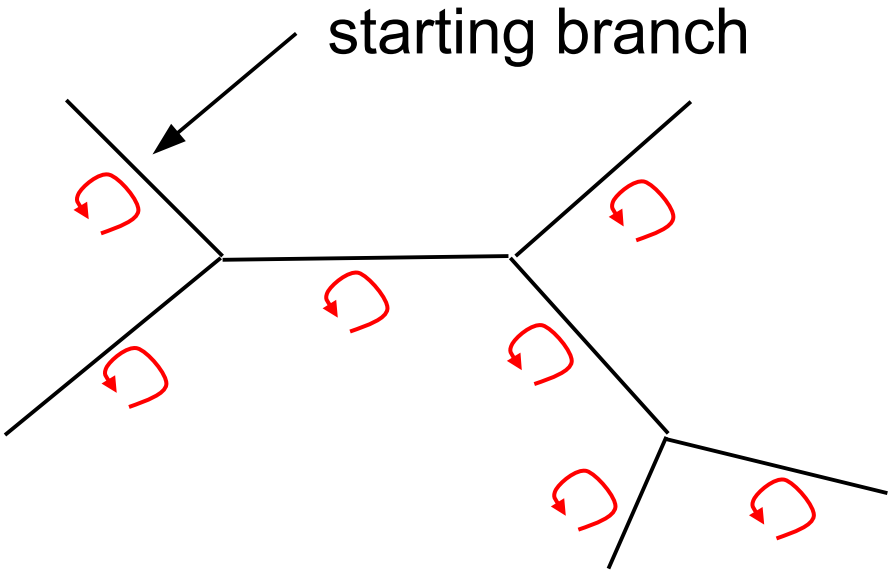
Branch Length Optimization



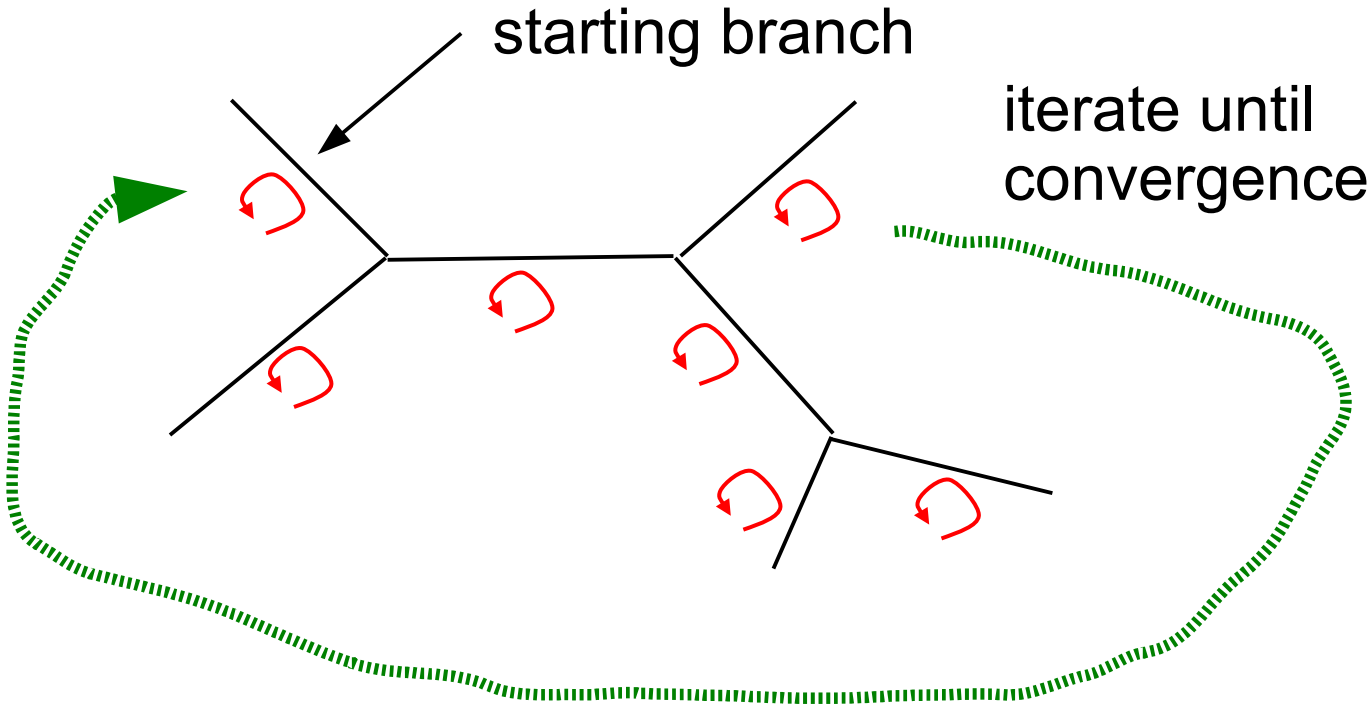
Branch Length Optimization



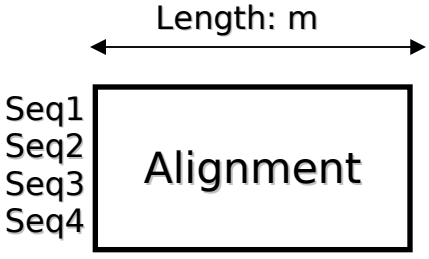
Branch Length Optimization

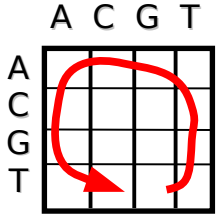


Branch Length Optimization



Maximum Likelihood



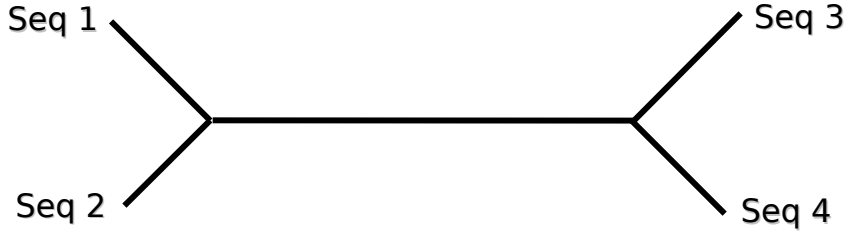


Substitution model

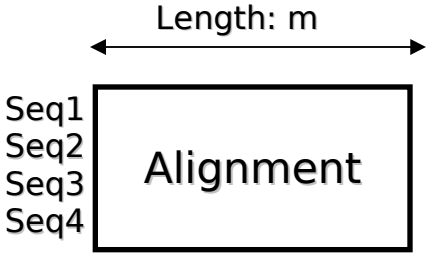
Prior probabilities,
Empirical base frequencies

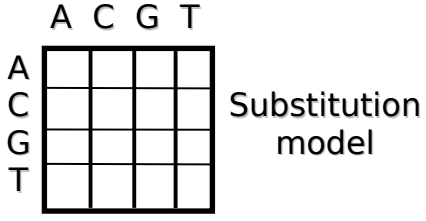
π_A π_C π_G π_T

optimize model parameters



Memory Requirements





Prior probabilities,
Empirical base frequencies

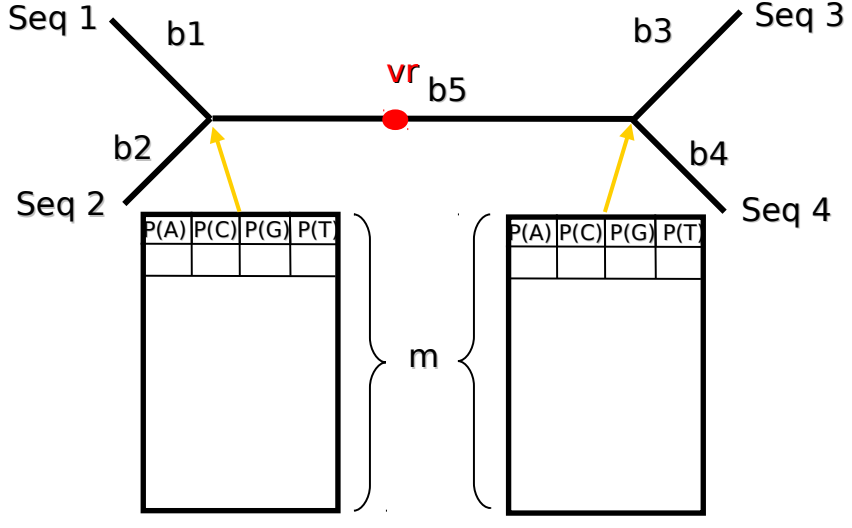
π_A π_C π_G π_T

Memory Consumption:
(n-2) * m * 4 * 1 * 8 bytes

DNA

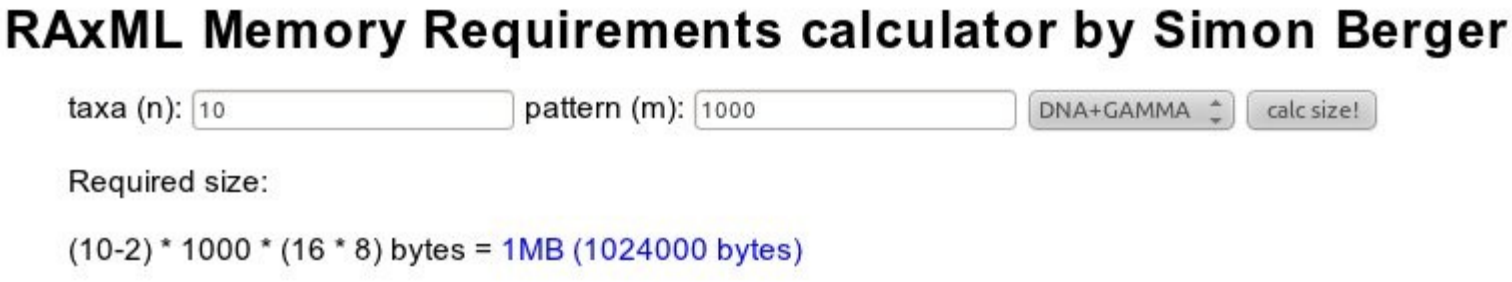
rate heterogeneity

double precision arithmetics

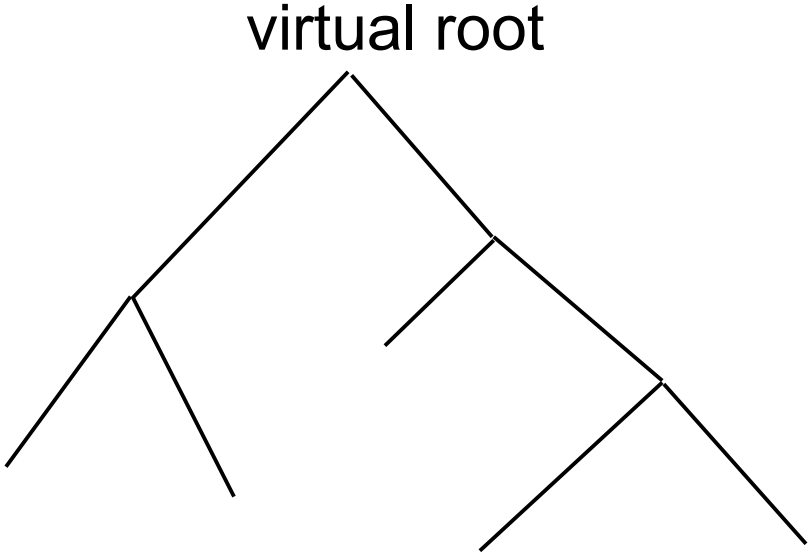


Memory Requirements

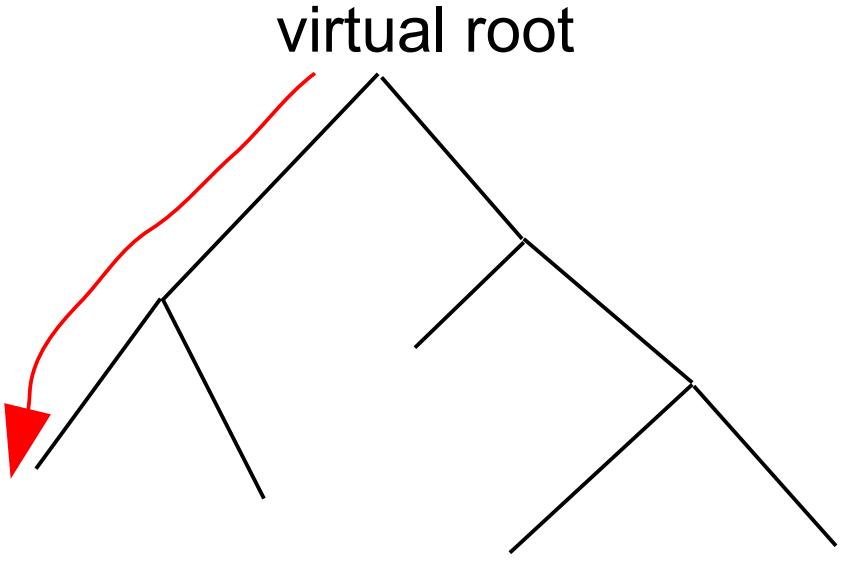
- For lazy people: mem. calculator at www.exelixis-lab.org/software.html



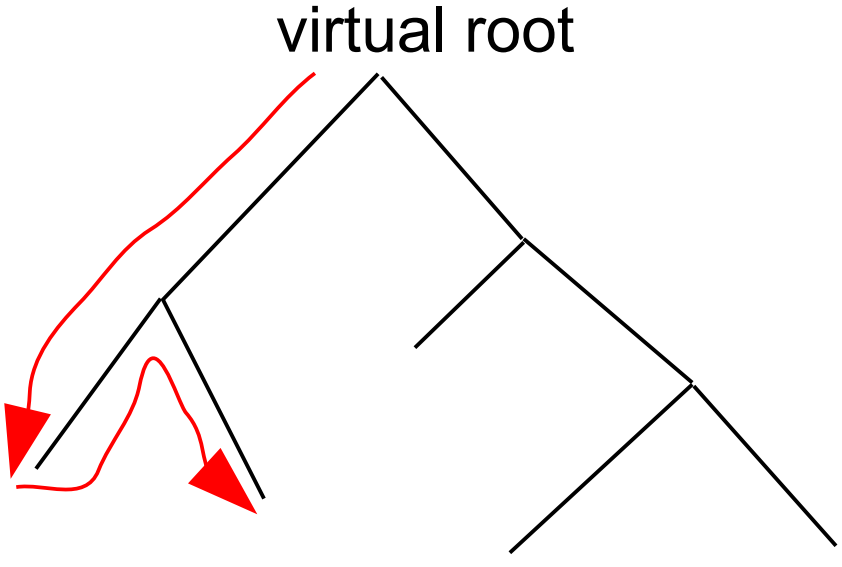
Post-order Traversal



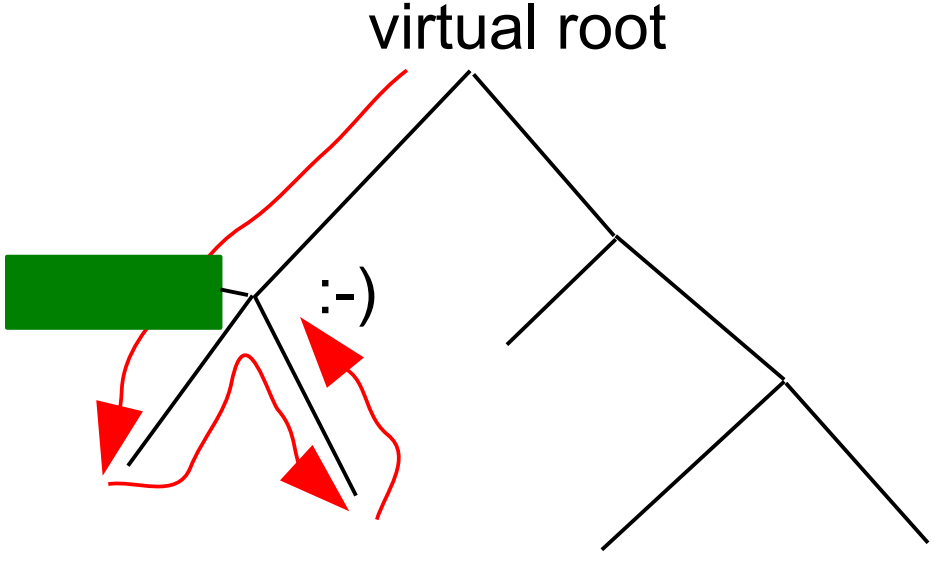
Post-order Traversal



Post-order Traversal

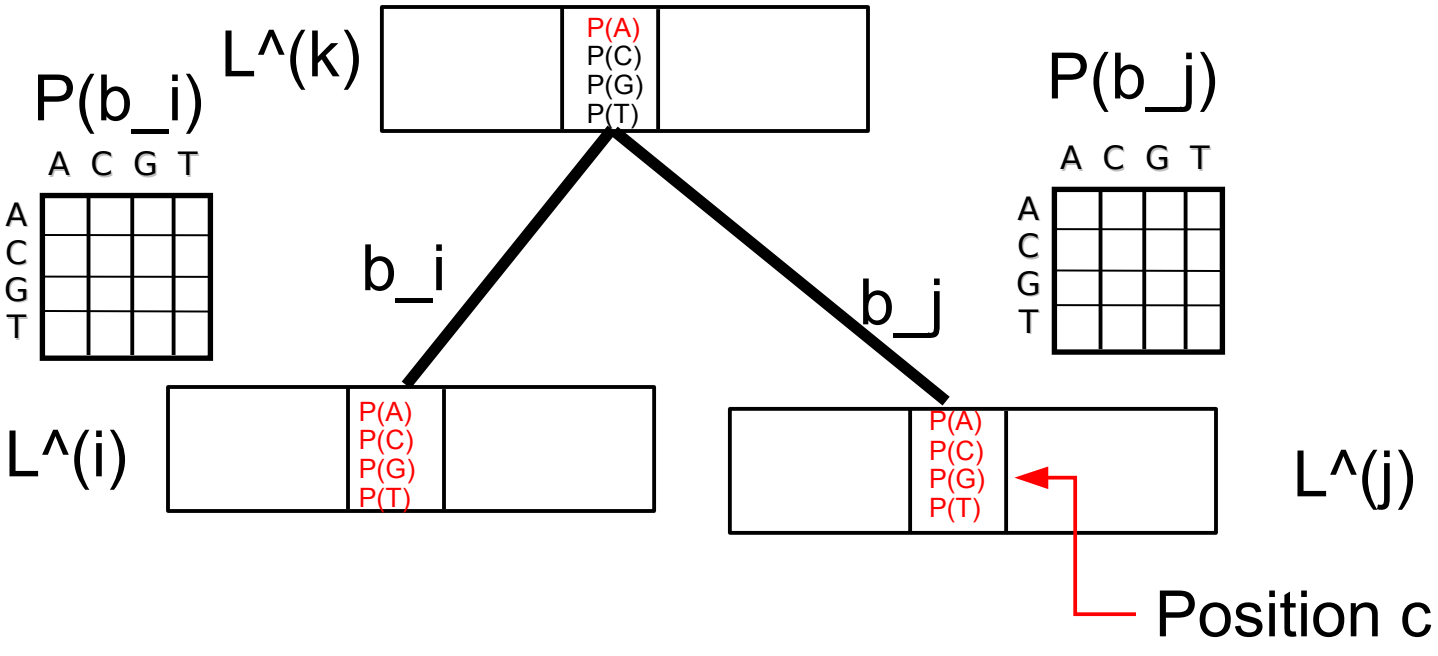


Post-order Traversal

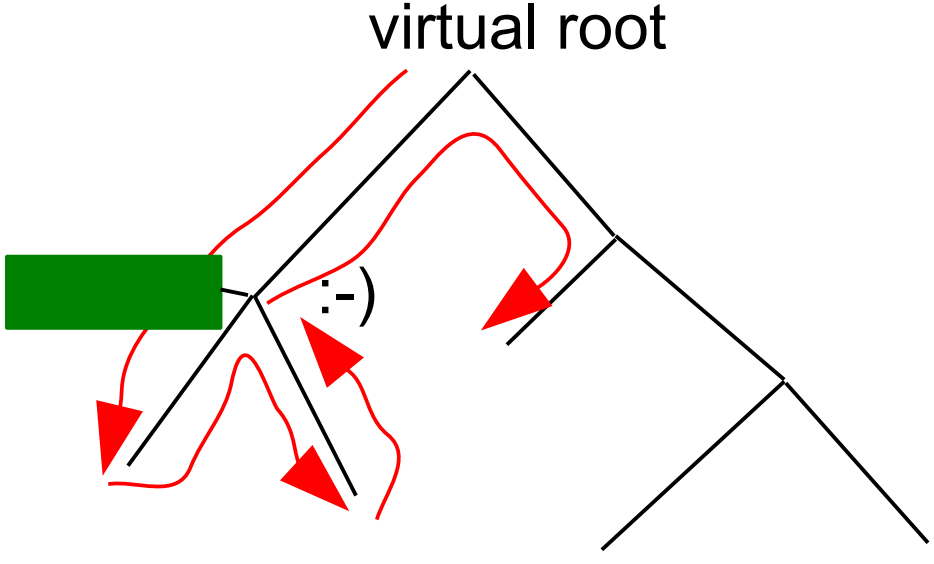


What happens when we compute this inner vector?

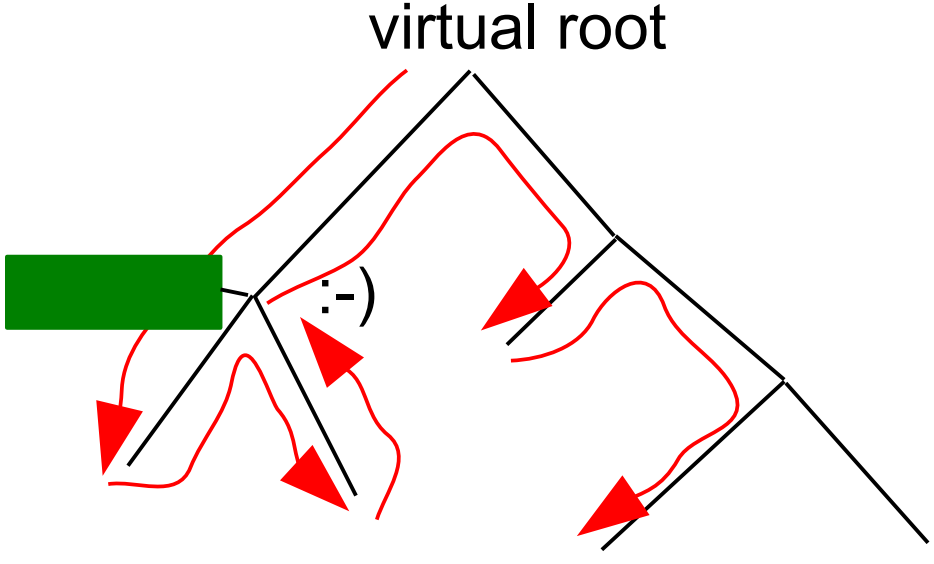
$$\vec{L}_A^{(k)}(c) = \Big(\sum_{S=A}^{T} P_{AS}(b_i)\vec{L}_S^{(i)}(c)\Big)\Big(\sum_{S=A}^{T} P_{AS}(b_j)\vec{L}_S^{(j)}(c)\Big)$$



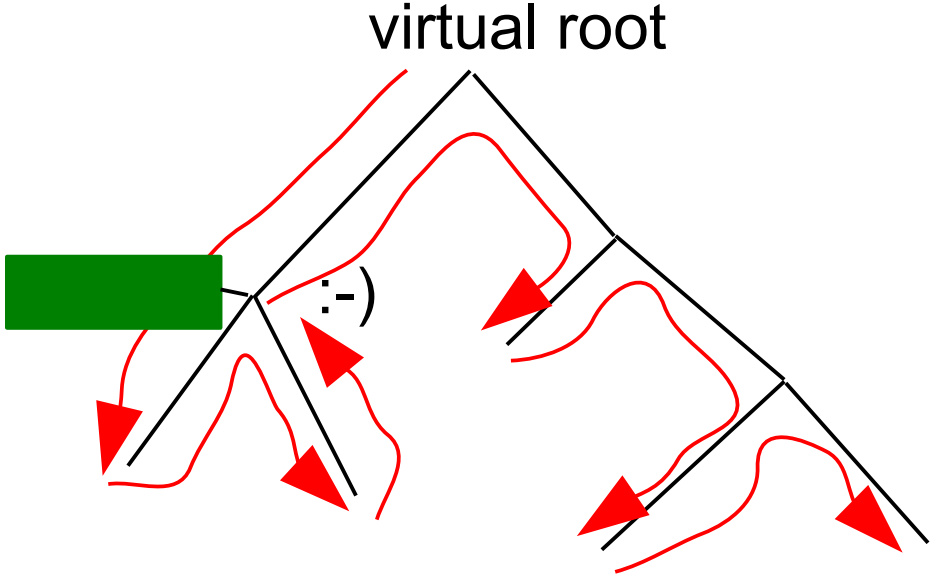
Post-order Traversal



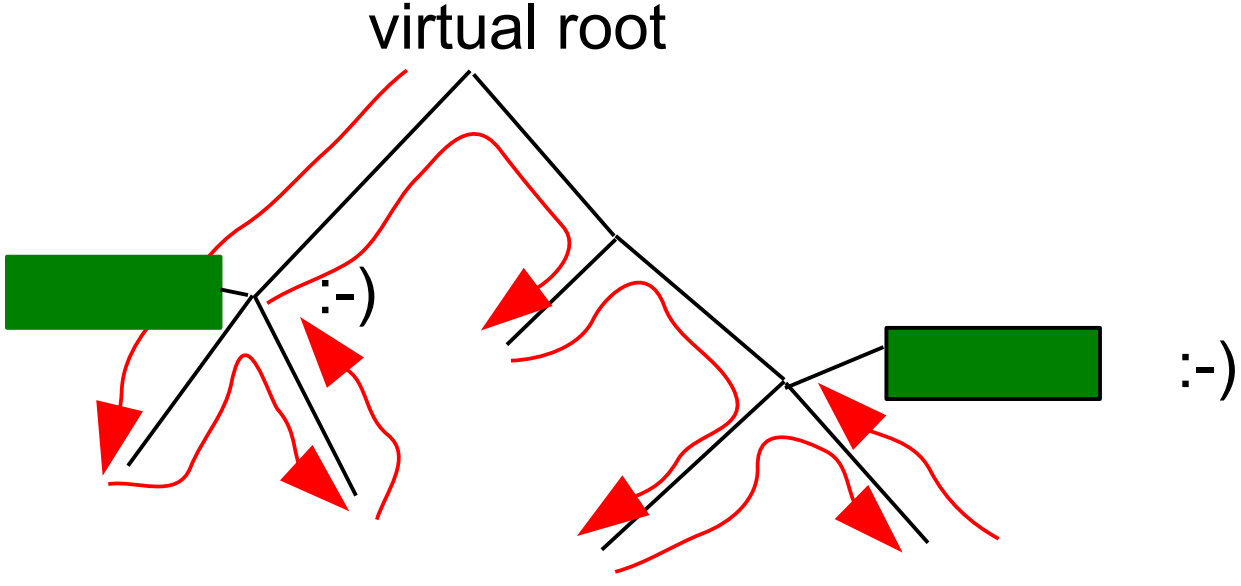
Post-order Traversal



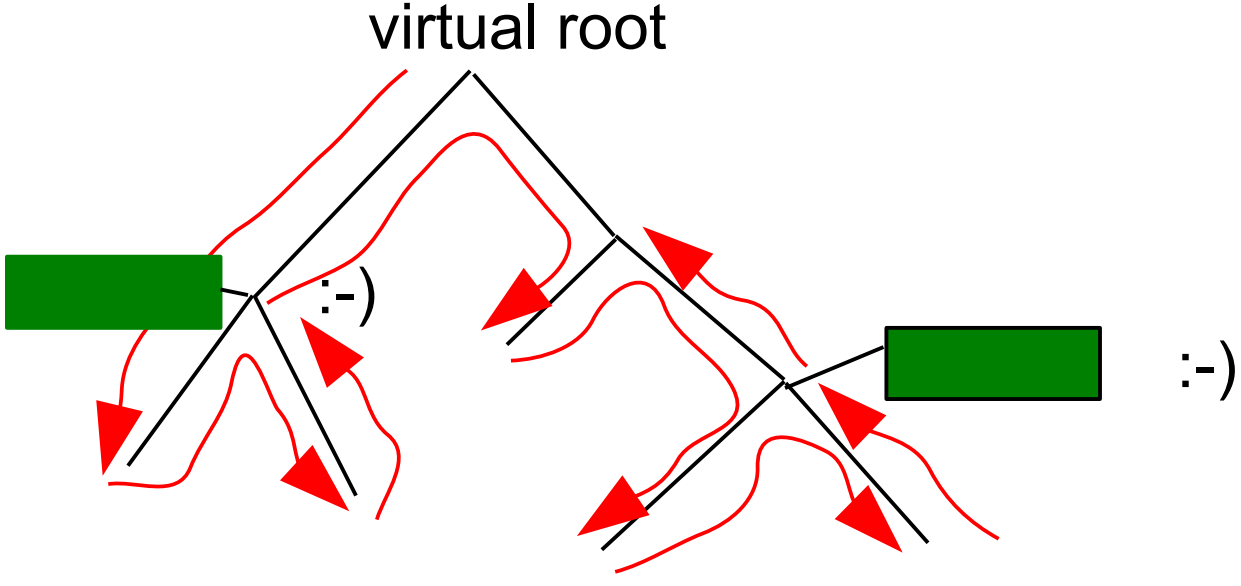
Post-order Traversal



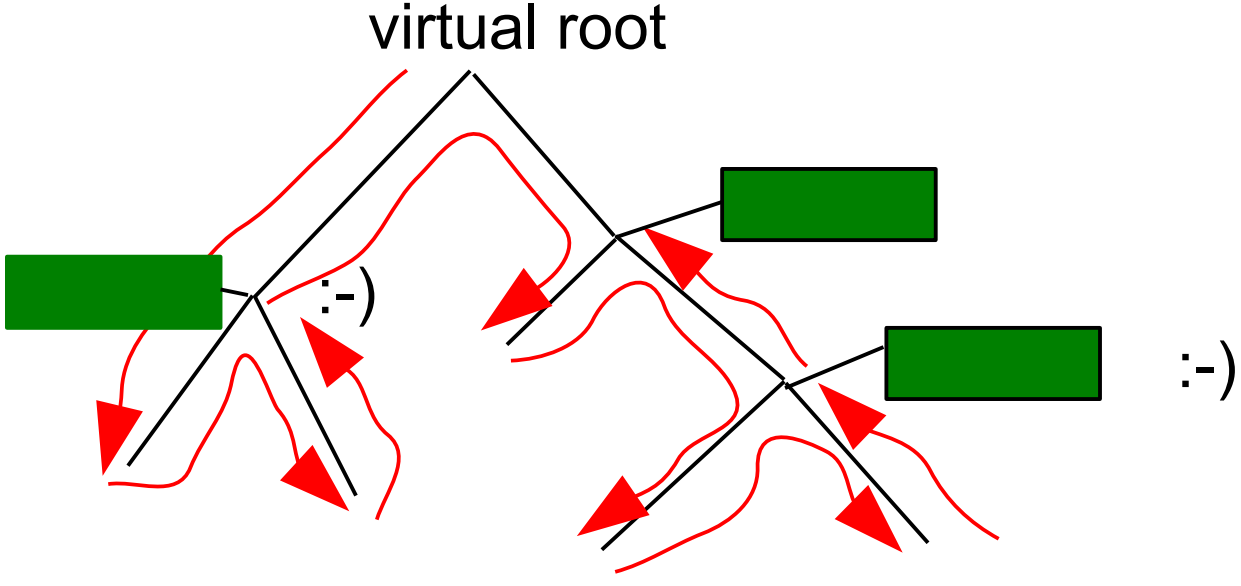
Post-order Traversal



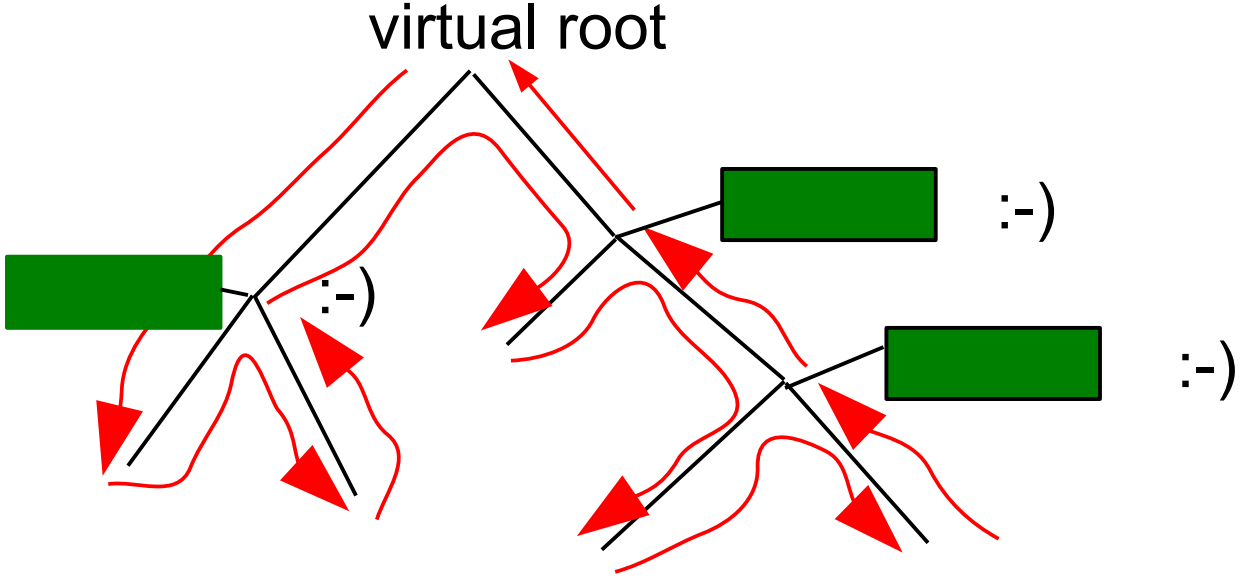
Post-order Traversal



Post-order Traversal

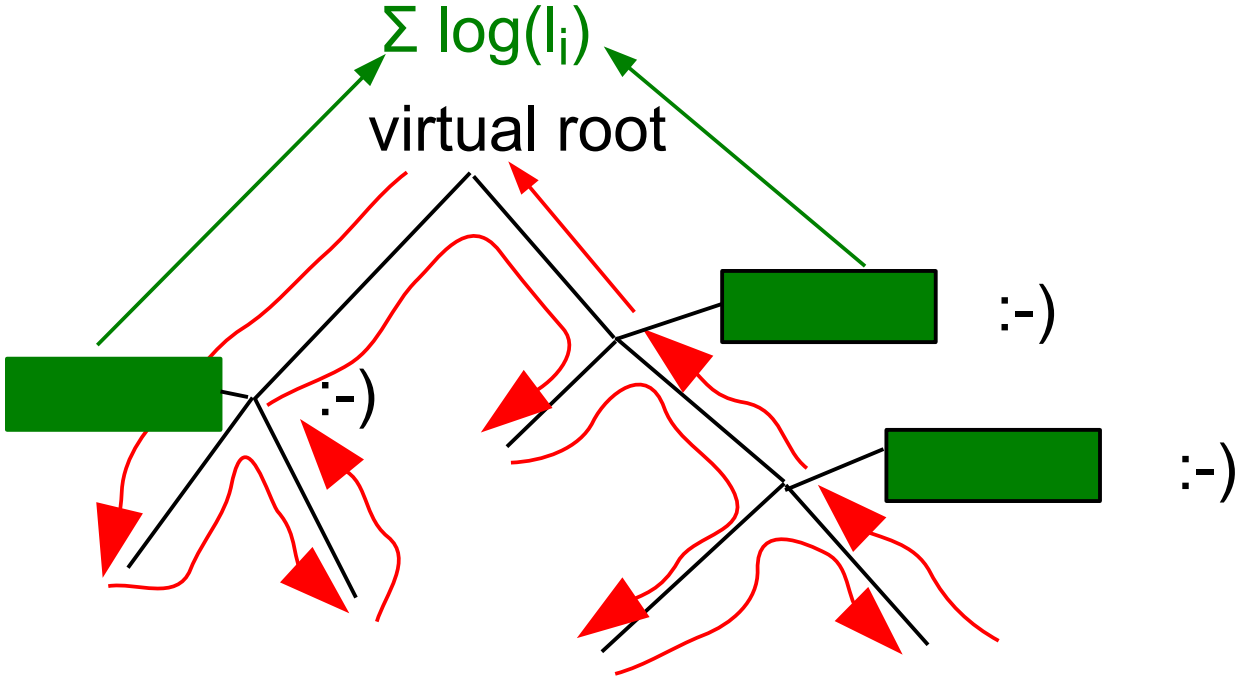


Post-order Traversal

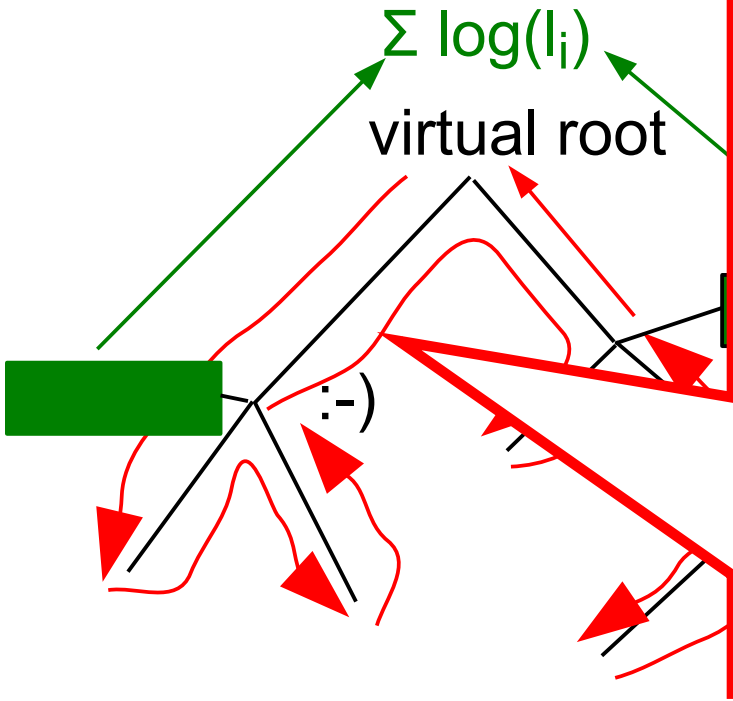


Post-order Traversal

tree score: sum over per-site log likelihoods



Post-order Traversal

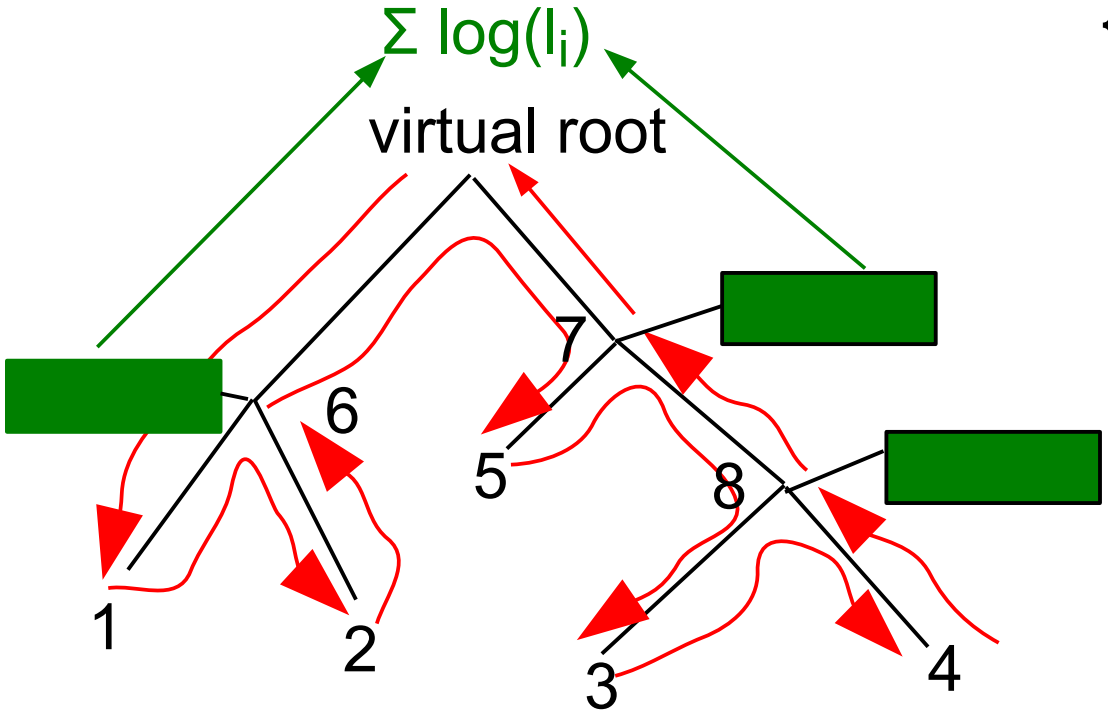


We need to communicate the traversal order to all parallel processes:
Shared-Memory: master thread computes and stores it: workers read it
Distributed-Memory: master process computes and broadcasts it as message to workers!

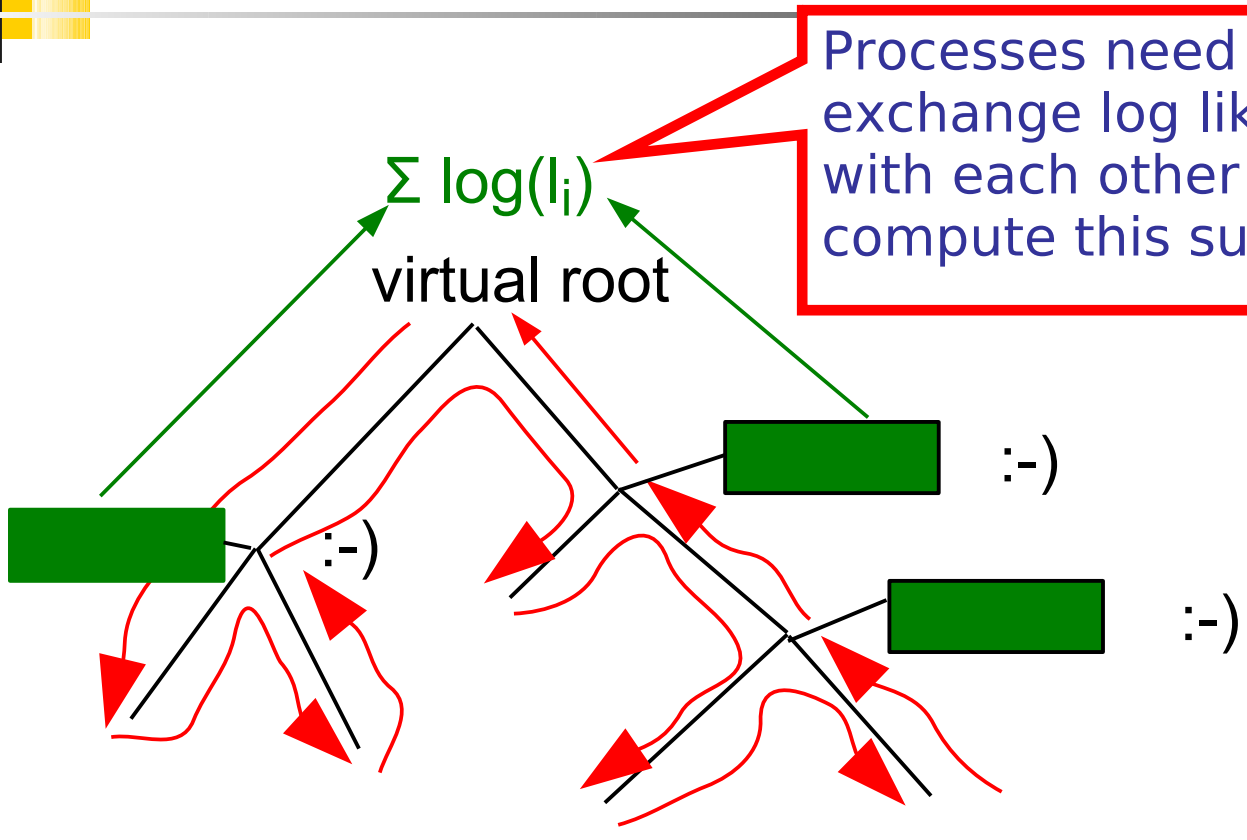
Post-order Traversal

Traversal Descriptor:

{(6 ← 1,2),
(8 ← 3,4),
(7 ← 5,8),
(L ← 6,7)}



Post-order Traversal



Basic Operations Maximum Likelihood

- Compute Conditional Likelihood Array at an inner node
- Compute Likelihood at Virtual Root
- Optimize a Branch Length for a given Branch
- Optimize all Branch Lengths
- Optimize other Model Parameters

Basic Operations
Maximum Likelihood

- Compute Conditional Likelihood Array at an inner node
- Compute Likelihood at Virtual Root
- Optimize a Branch Length for a given Branch
- Optimize all Branch Lengths
- Optimize other Model Parameters

Bayesian programs only require two operations

Outline

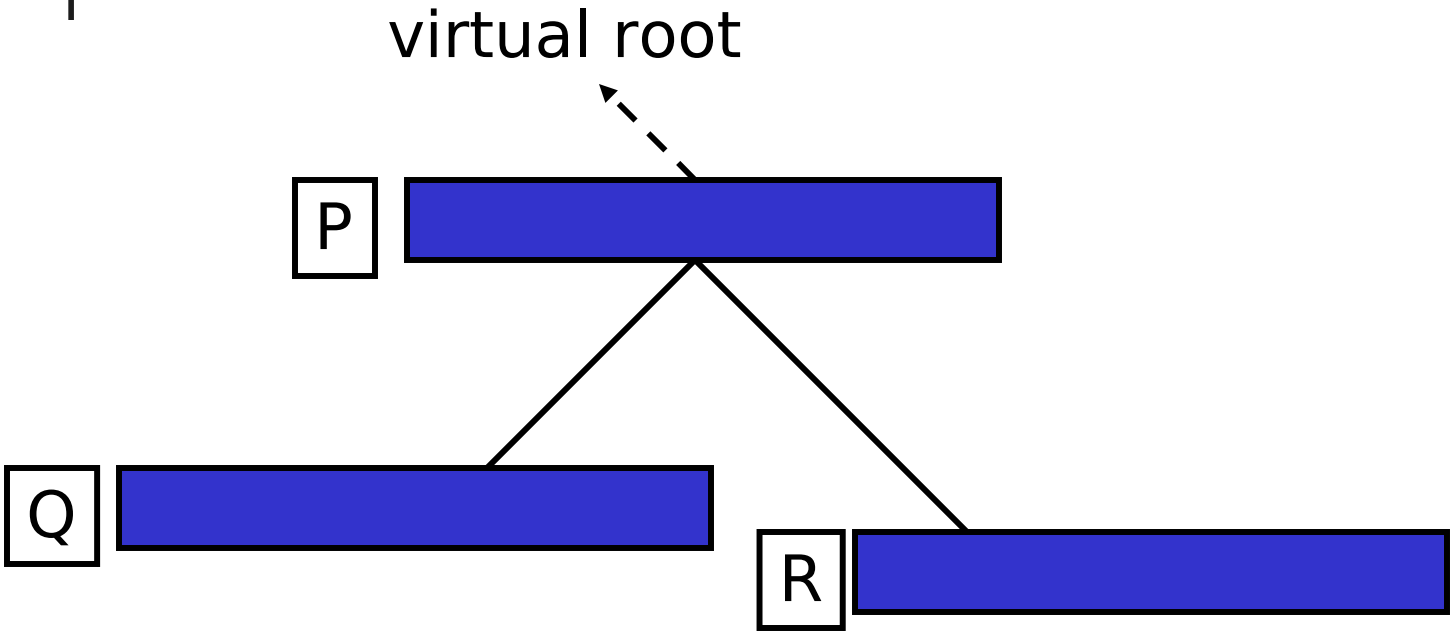
- Computing the Likelihood
- Optimizing & Parallelizing Likelihood Computations
- Saving Memory in Likelihood Computations
- HPC population genetics

Optimization

- Use vector intrinsics SSE3/AVX
- Special implementations for:

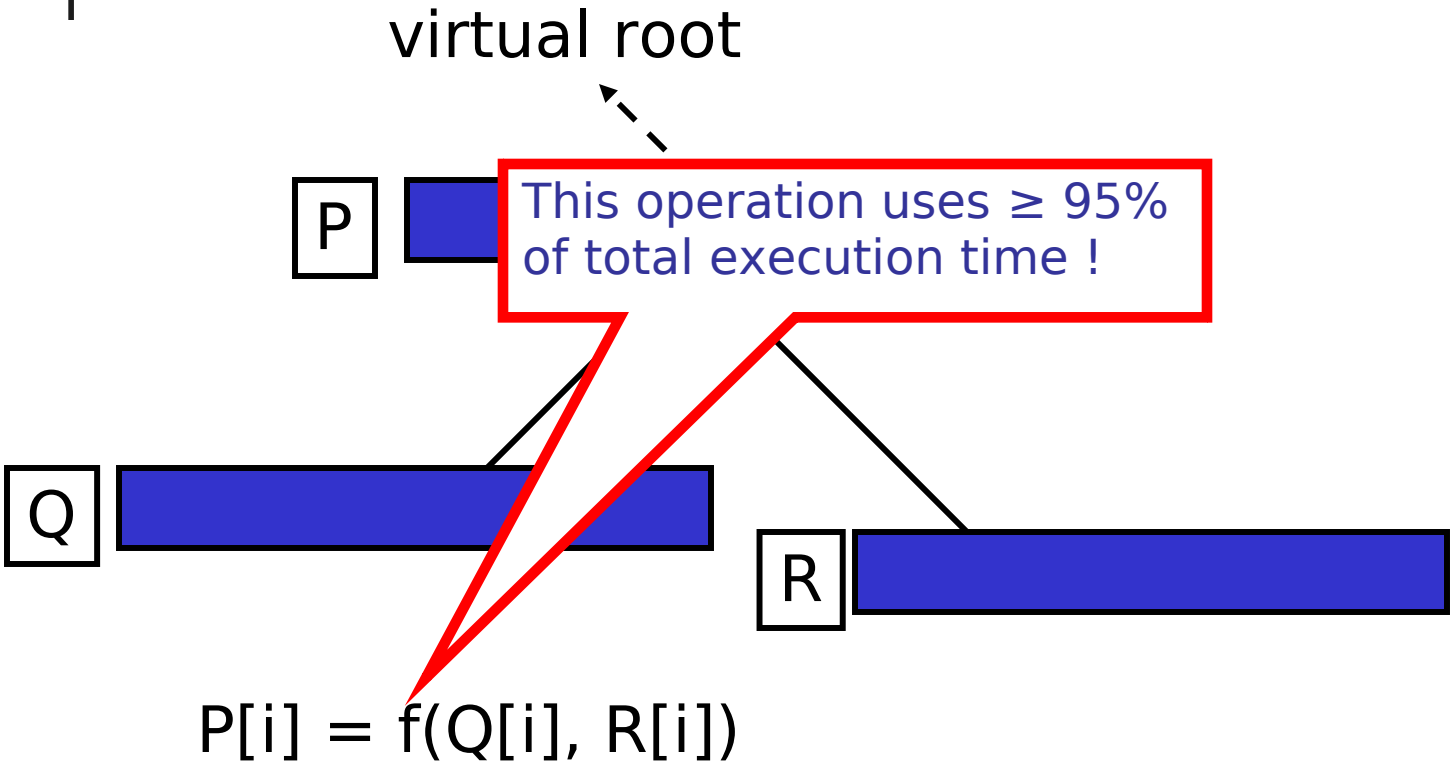
- I will spare you the details
- But, avoid redundant computations

Loop Level Parallelism

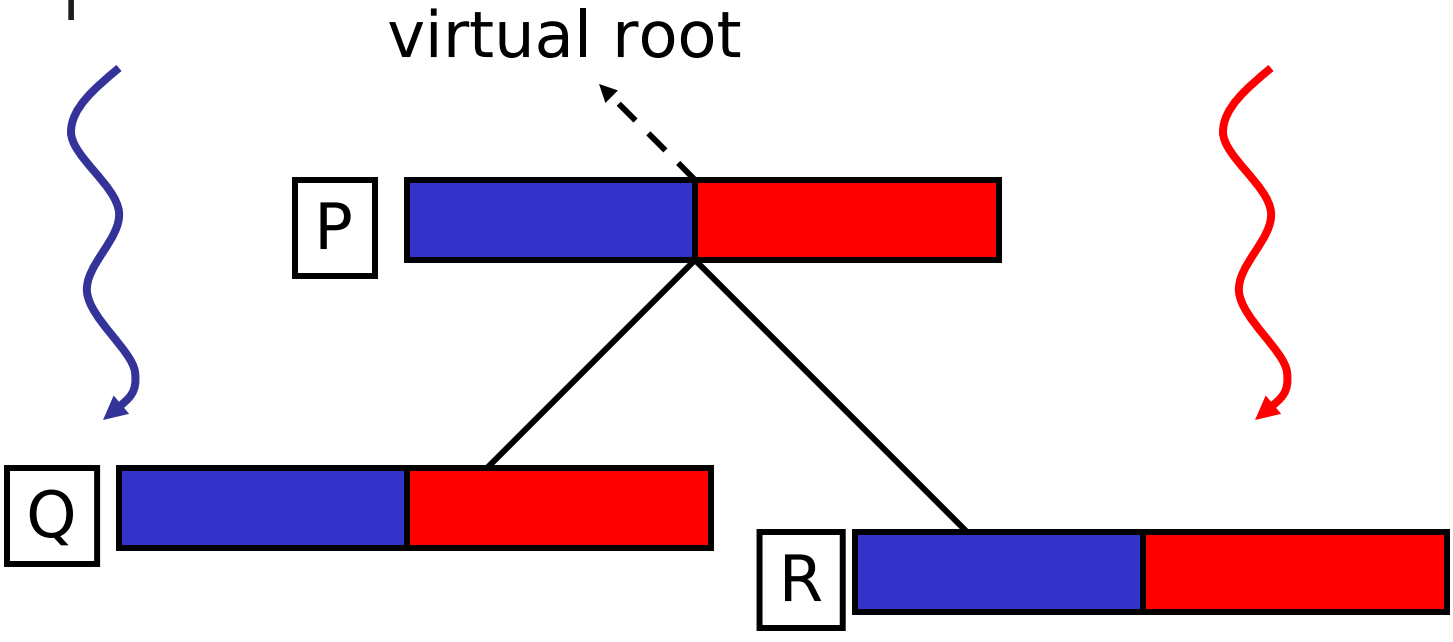


P[i] = f(Q[i], R[i])

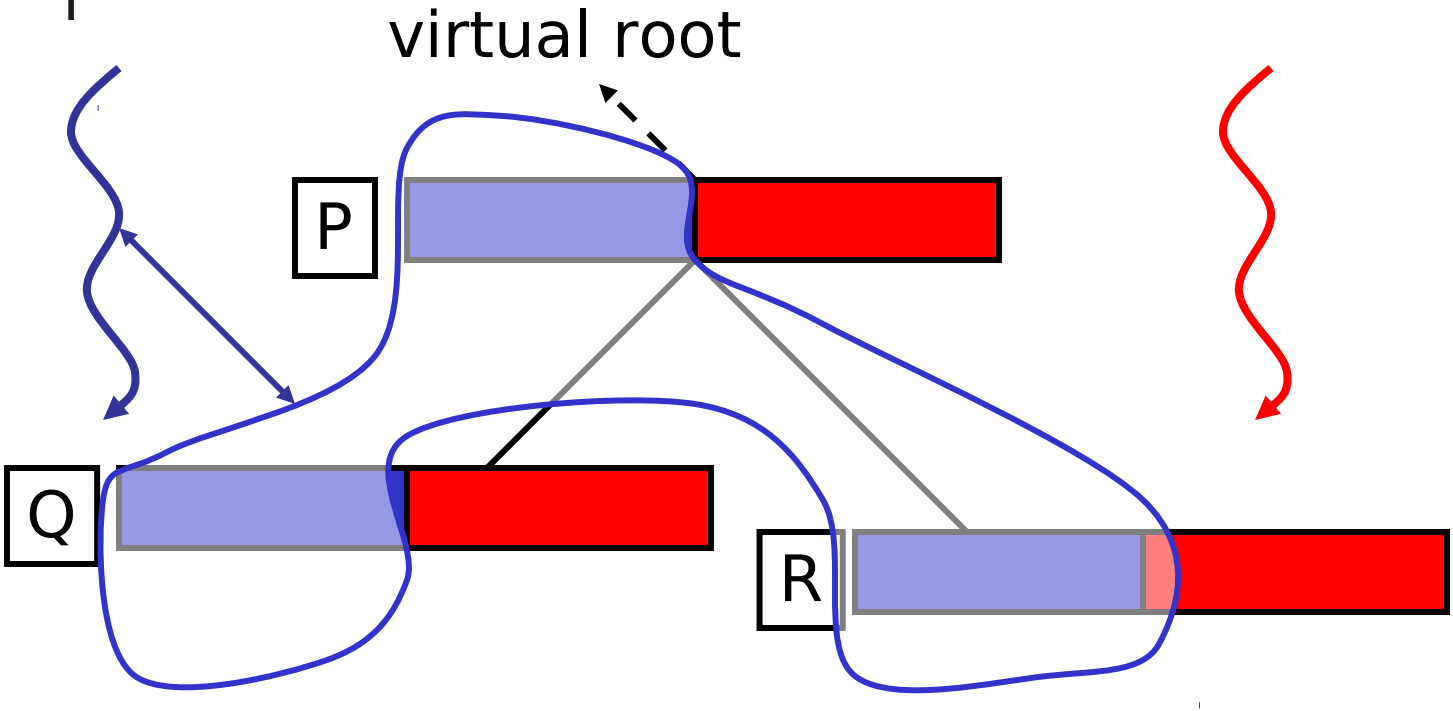
Loop Level Parallelism



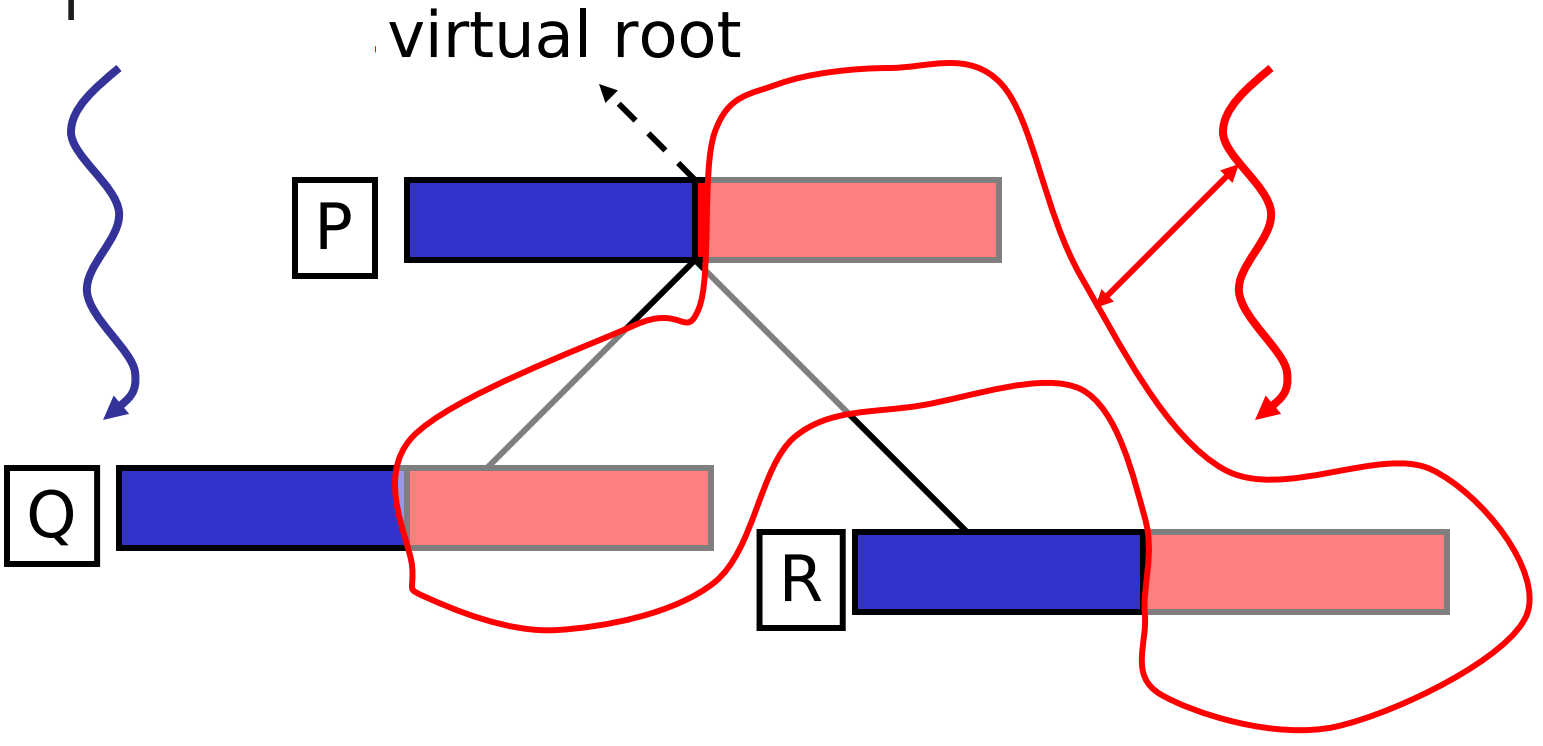
Loop Level Parallelism



Loop Level Parallelism

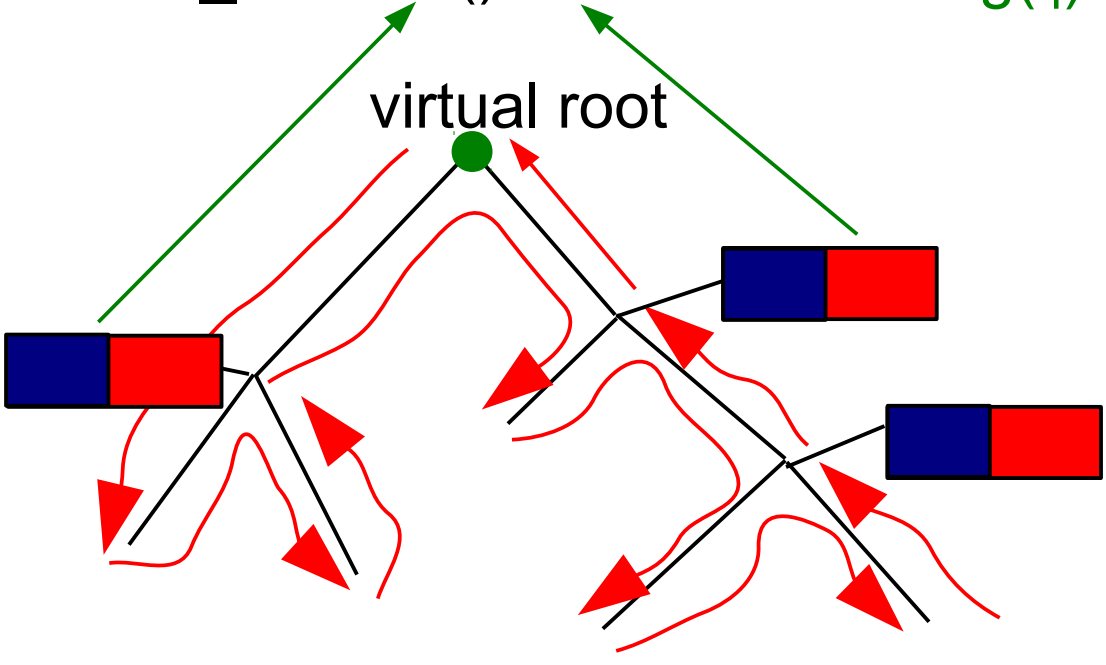


Loop Level Parallelism

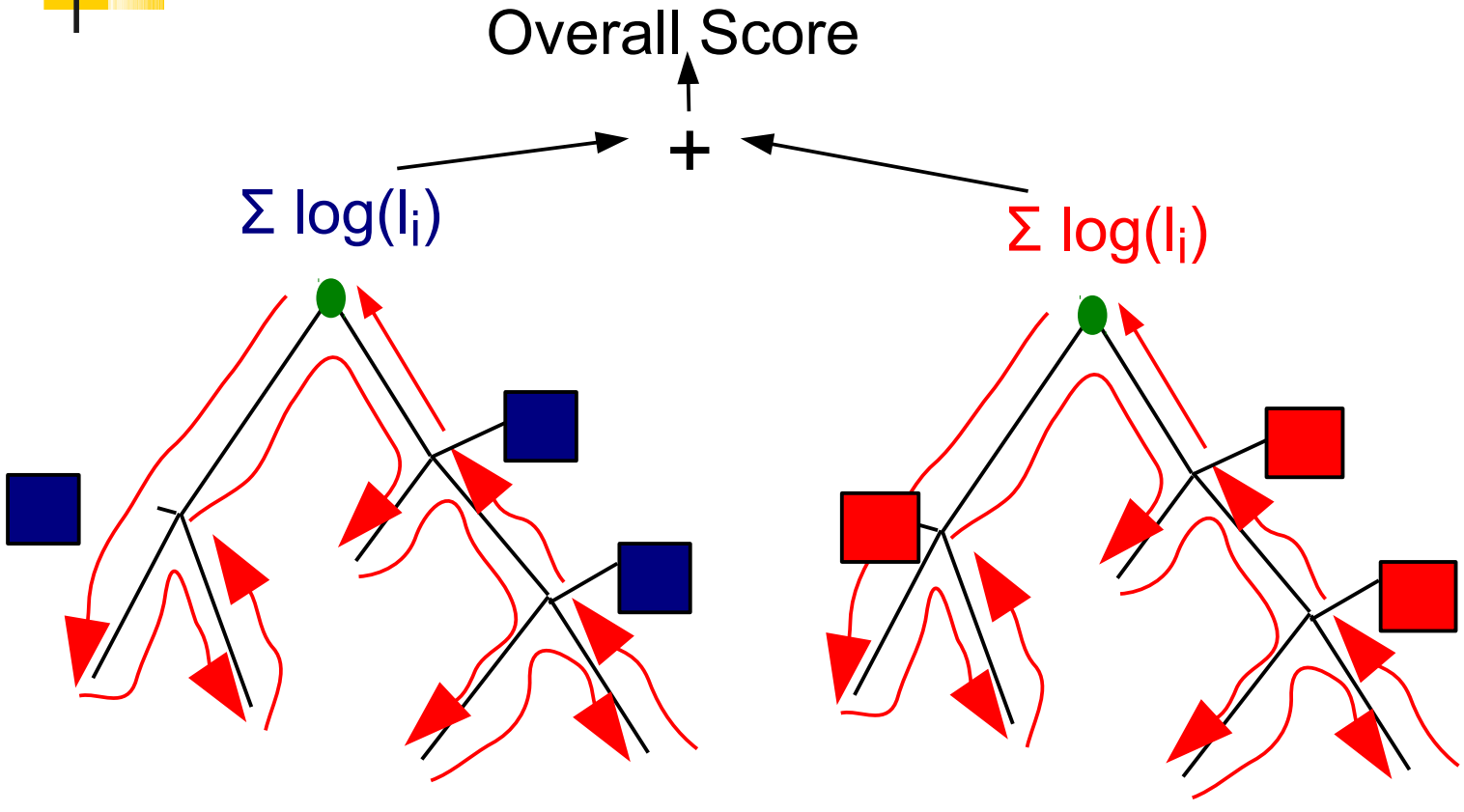


Parallel Post-order Traversal

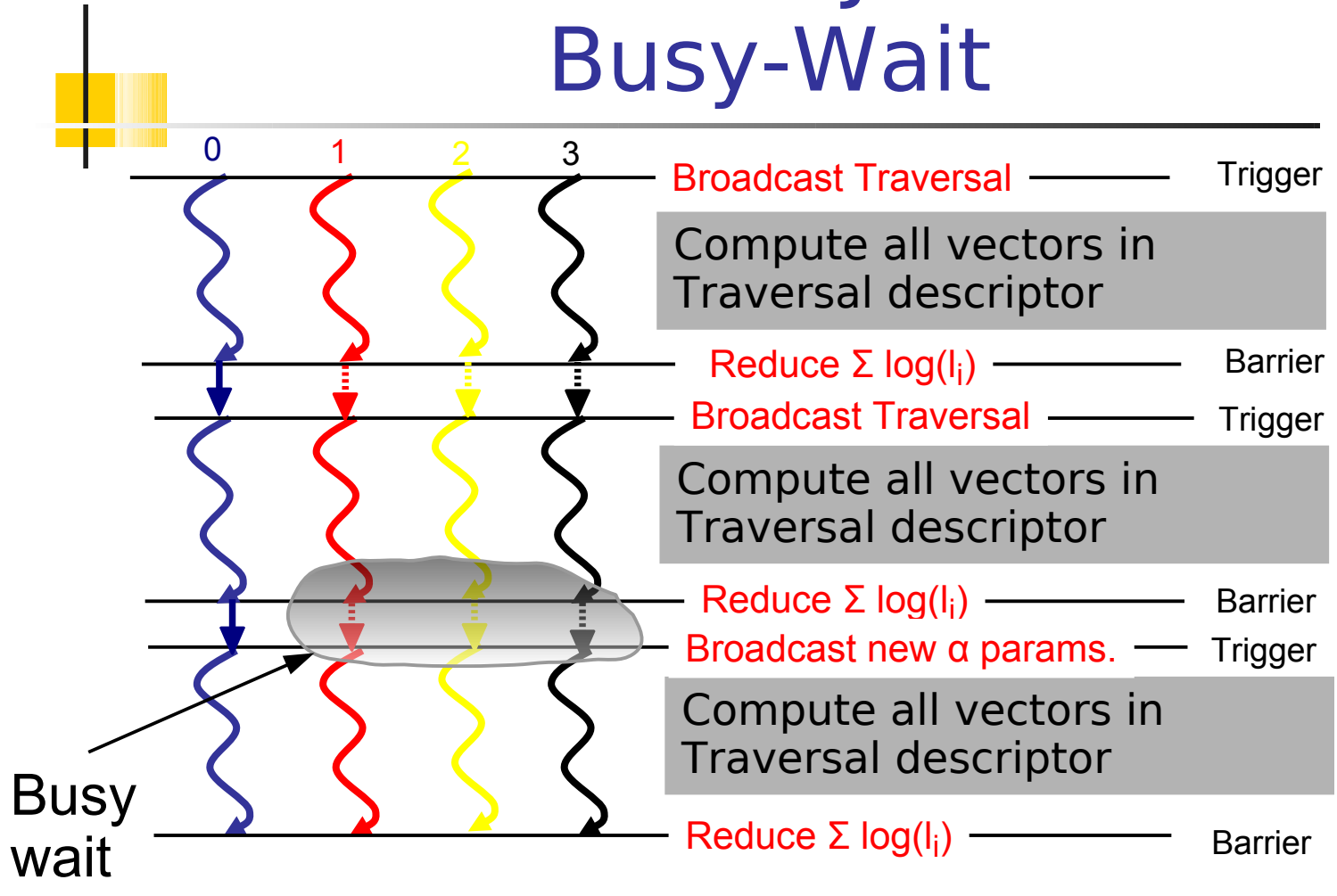
Only need to synchronize at the root
→ MPI_Reduce() to calculate: $\Sigma \log(l_i)$



Parallel Post-order Traversal



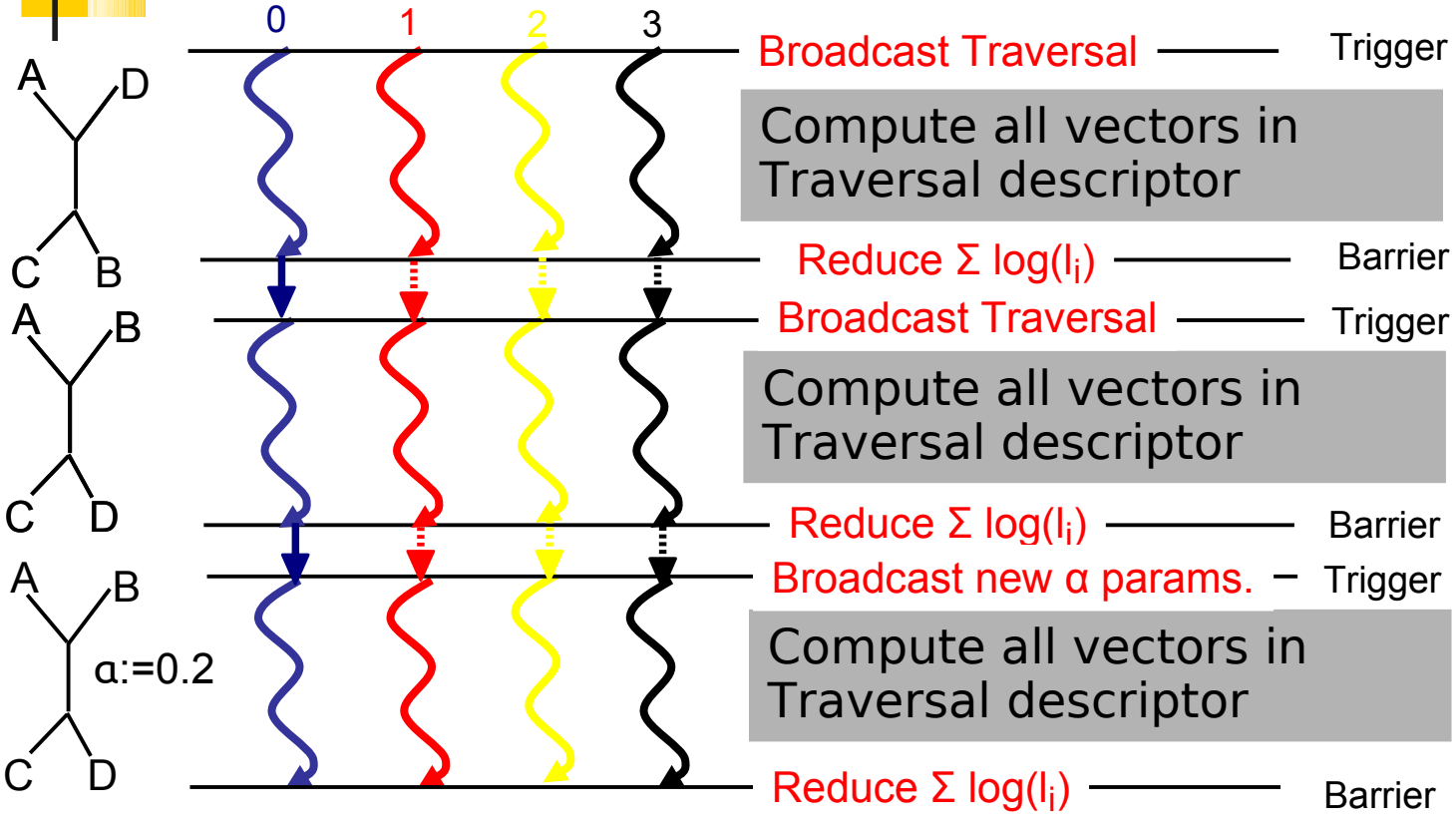
Classic Fork-Join with Busy-Wait



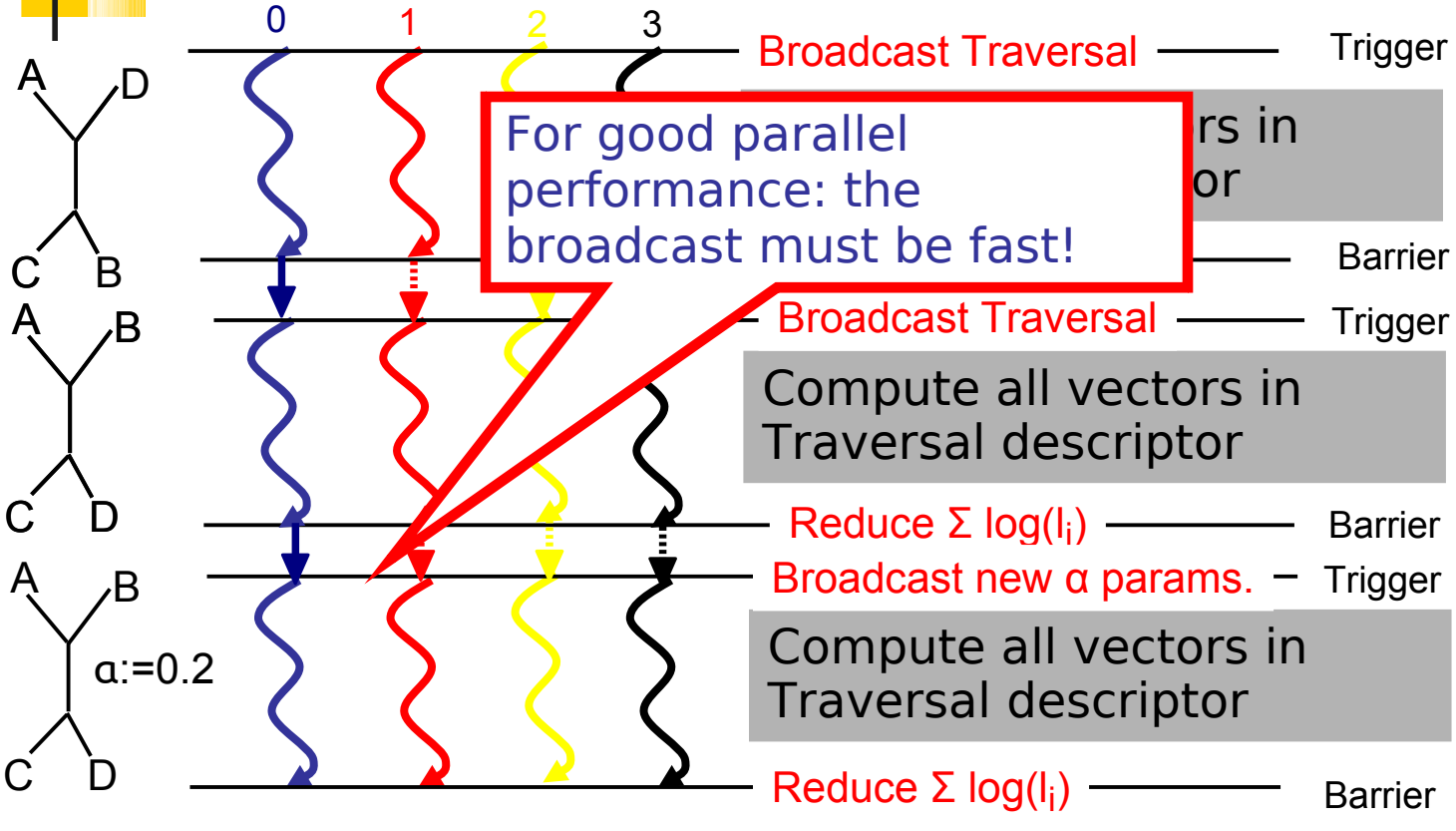
Synchronizations in RAxML with Pthreads

- RAxML Pthreads for a run time of about **10 seconds** on 16 Cores/16 Threads
- 404 taxa 7429 sites: **194,000** Barriers
- 1481 taxa 1241 sites: **739,000** Barriers

Classic Fork-Join with Busy-Wait



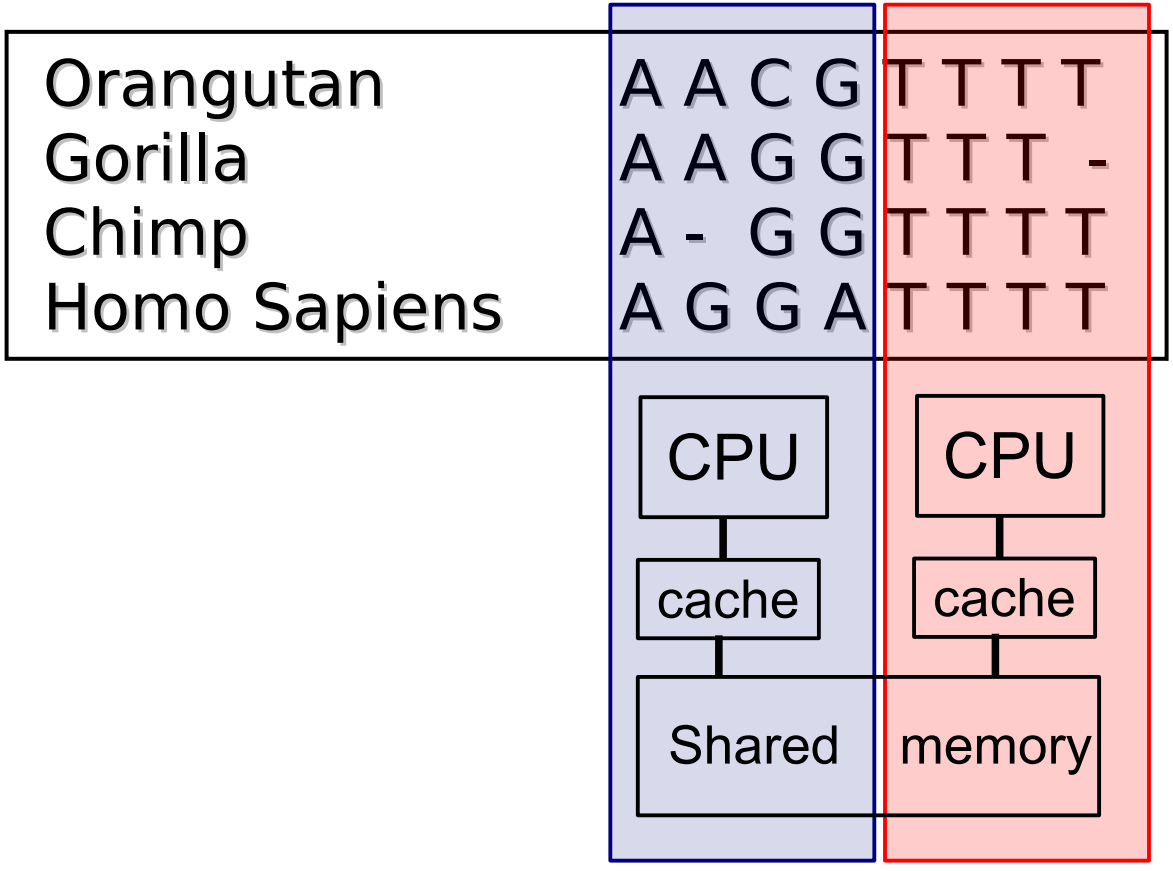
Classic Fork-Join with Busy-Wait



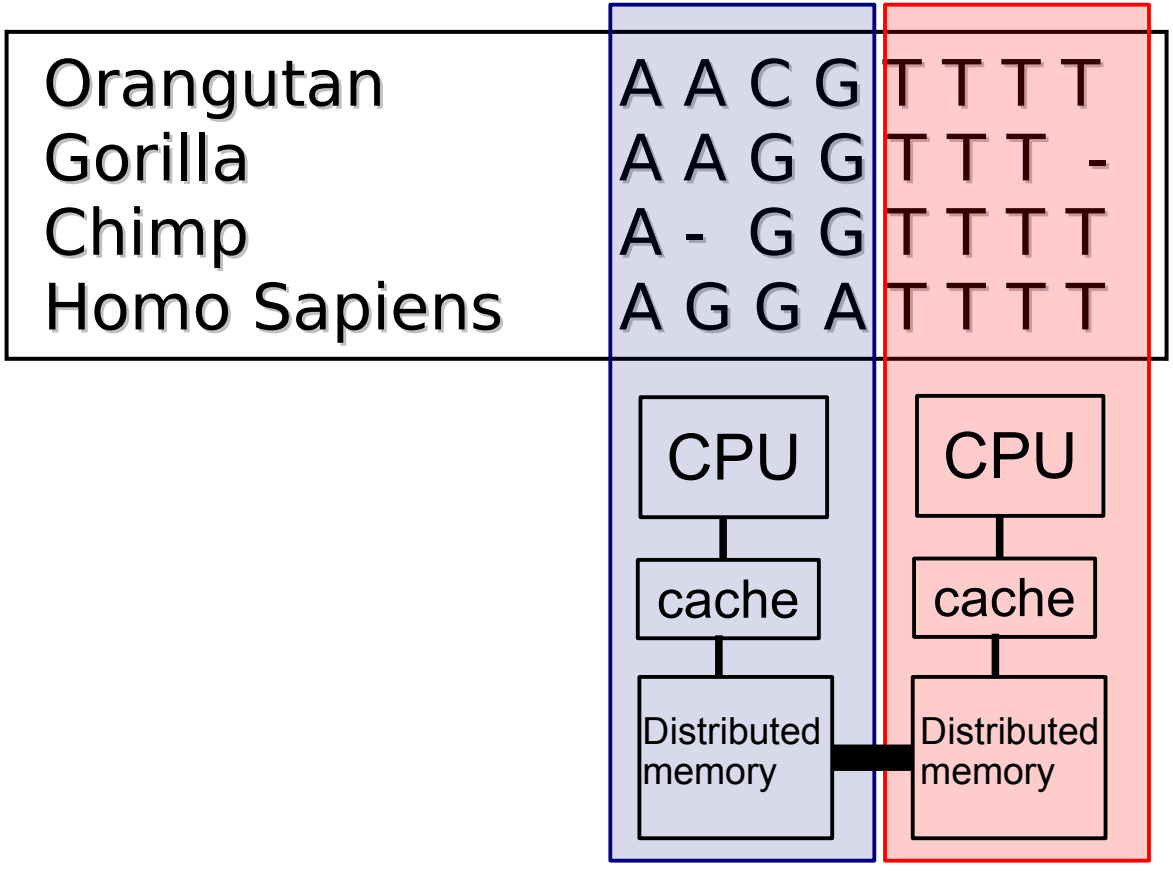
Parallel Performance Problems

- They all start with partitioned datasets!
- How do we distribute partitions to processors?
- How do we calculate parameter changes?
- How much time does our Broadcast take?
- Goal: Keep all processors busy all the time → reduce amount of communication and synchronization!

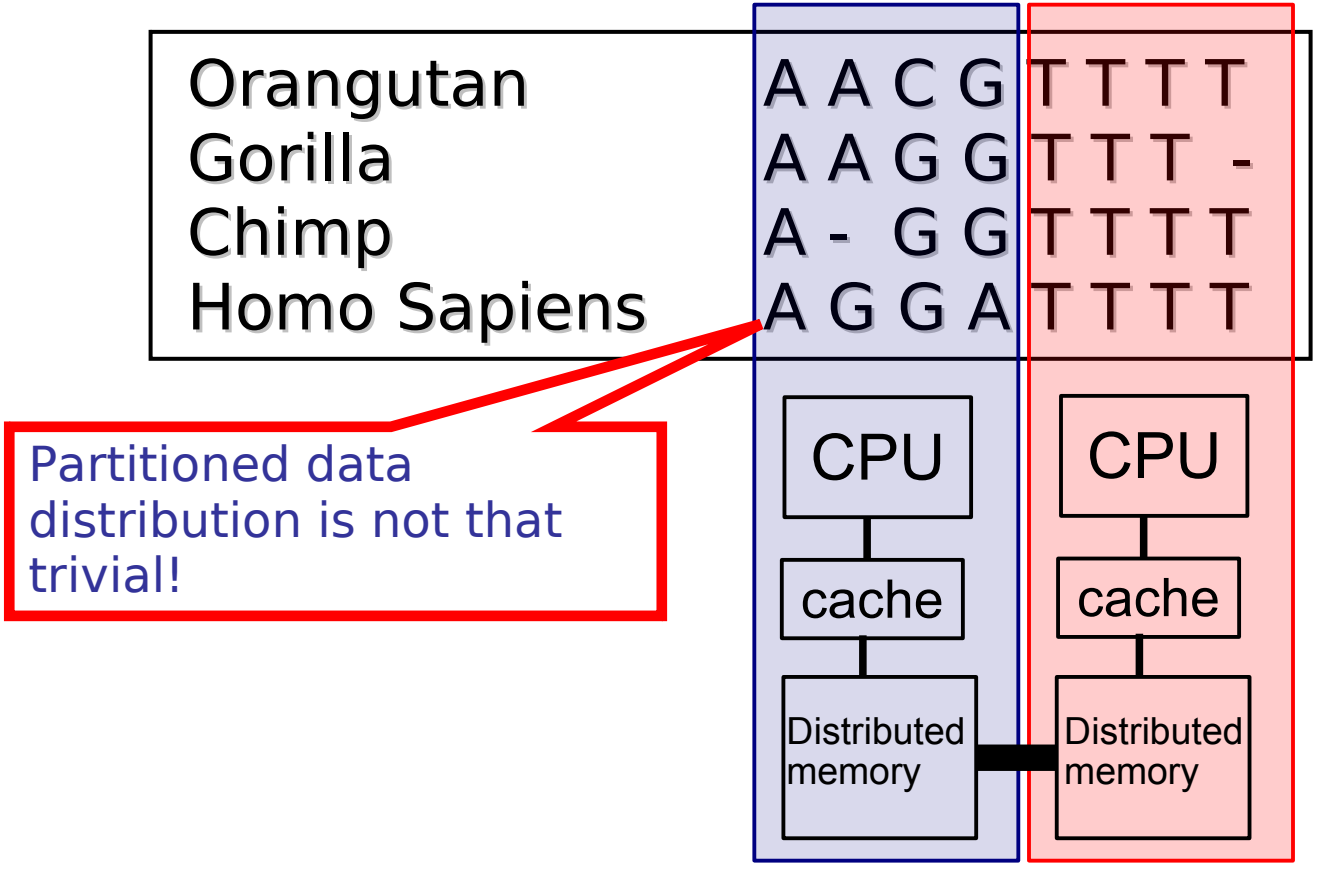
Data Distribution



Data Distribution

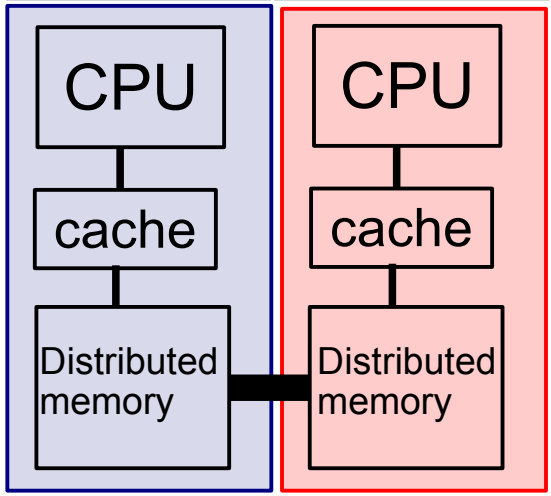


Data Distribution



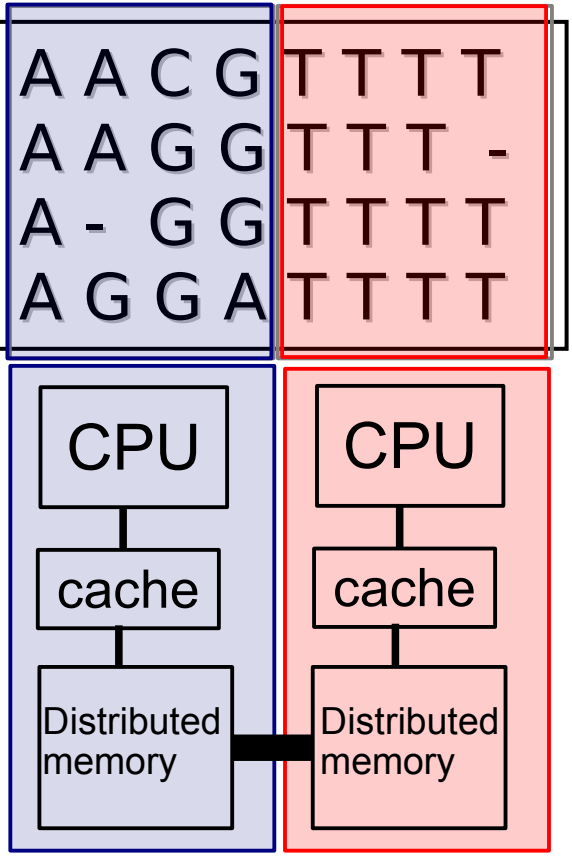
Data Distribution

Orangutan	A A C G	T T T T
Gorilla	A A G G	T T T -
Chimp	A - G G	T T T T
Homo Sapiens	A G G A	T T T T



Data Distribution I

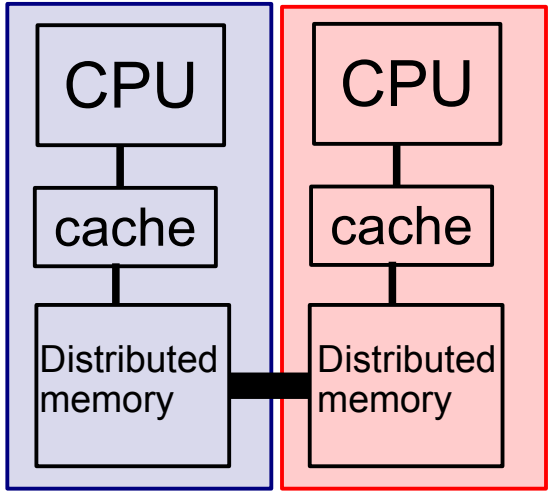
Orangutan	A A C G	T T T T
Gorilla	A A G G	T T T -
Chimp	A - G G	T T T T
Homo Sapiens	A G G A	T T T T



Works well when we have more partitions than processors:
May lead to load imbalance not all processors obtain equal number of sites!

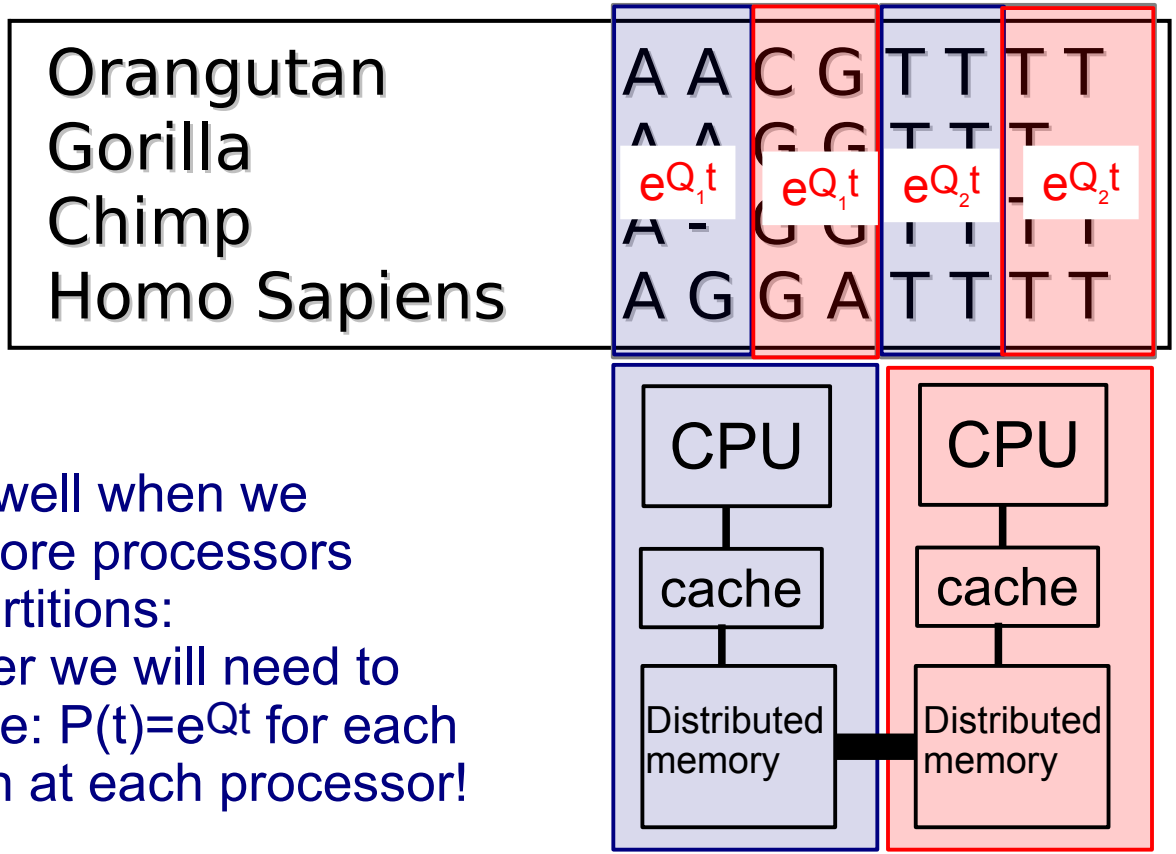
Data Distribution II

Orangutan	A	A	C G	T T	T T
Gorilla	A	A	G G	T T	T -
Chimp	A	-	G G	T T	T T
Homo Sapiens	A	G	G A	T T	T T



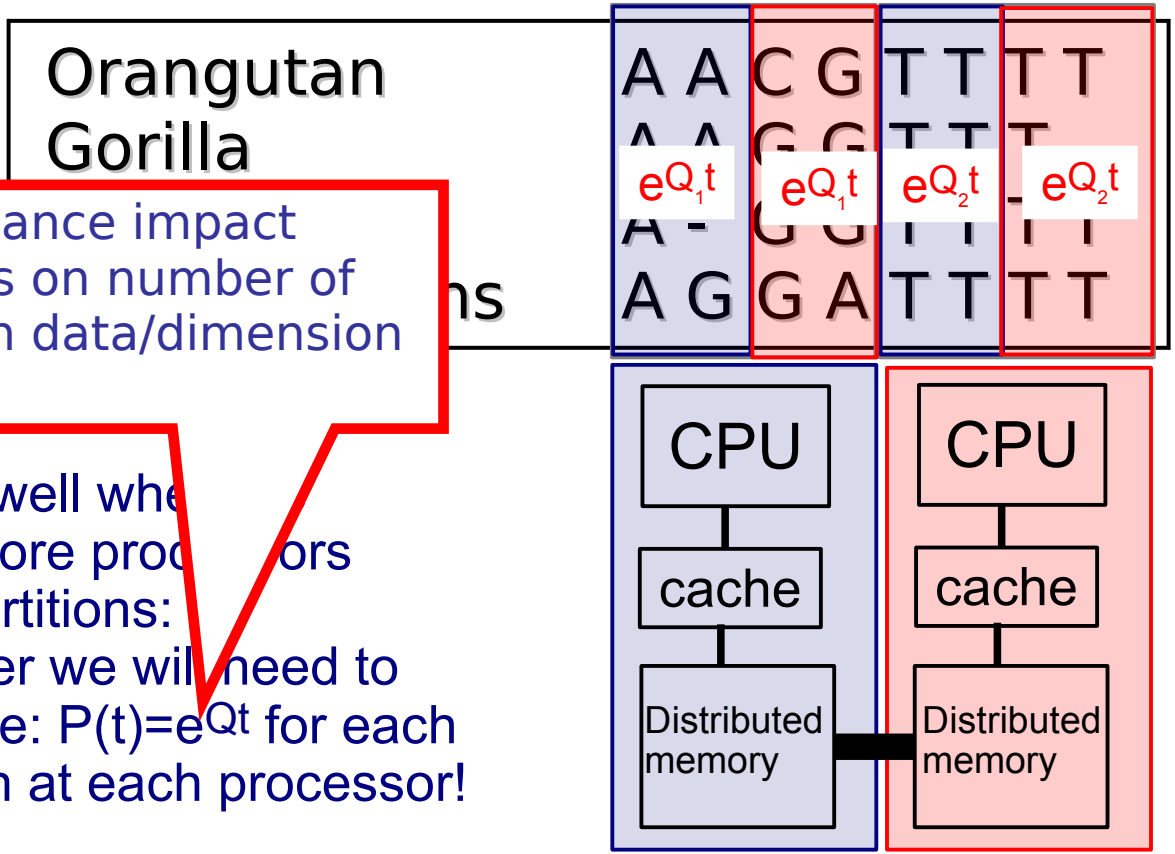
Works well when we have more processors than partitions:
However we will need to compute: $P(t)=e^{Qt}$ for each partition at each processor!

Data Distribution II



Works well when we have more processors than partitions:
However we will need to compute: $P(t)=e^{Qt}$ for each partition at each processor!

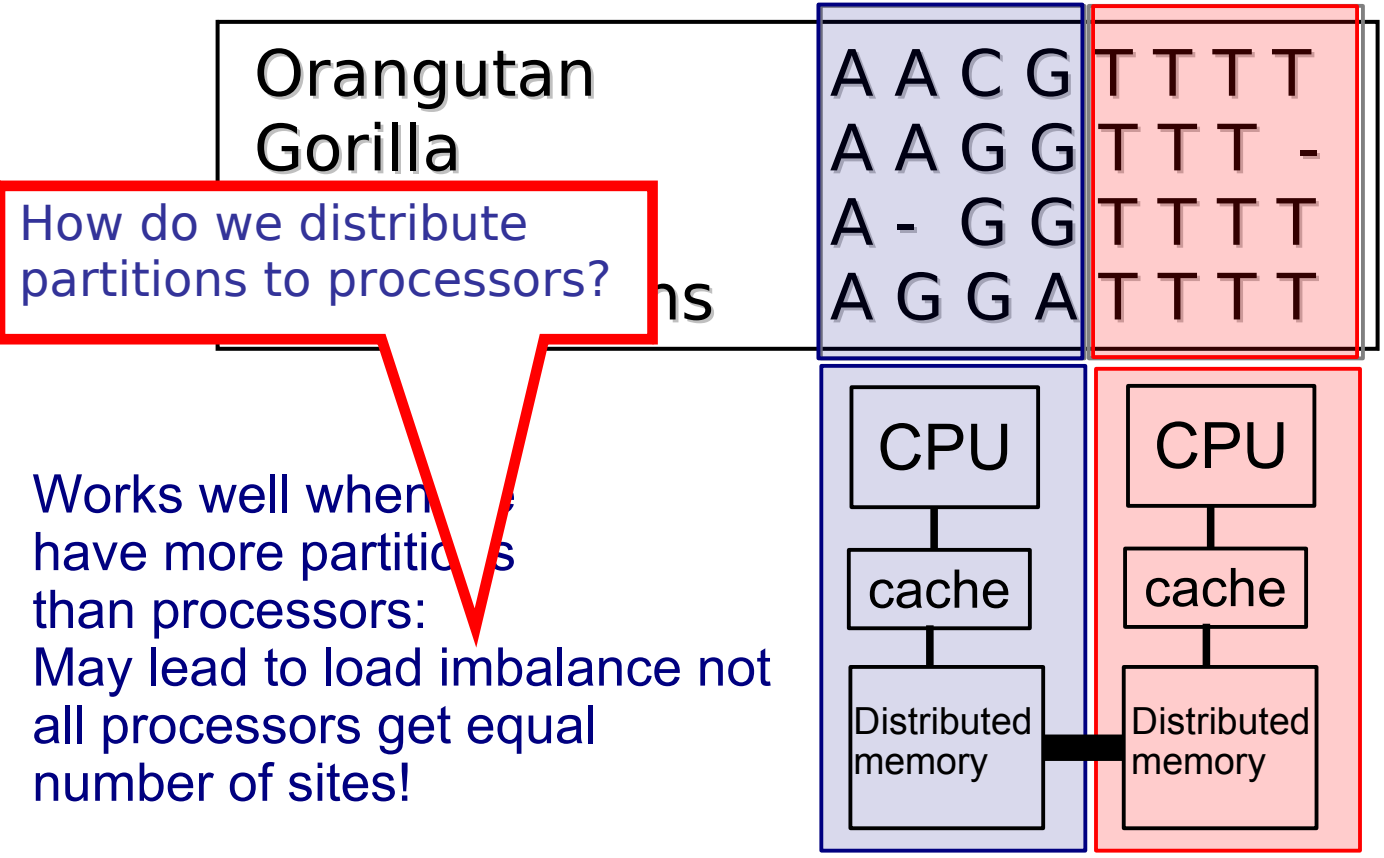
Data Distribution II



Performance impact depends on number of states in data/dimension of Q

Works well whe
have more proc ors
than partitions:
However we wil need to compute: $P(t)=e^{Qt}$ for each partition at each processor!

Data Distribution I

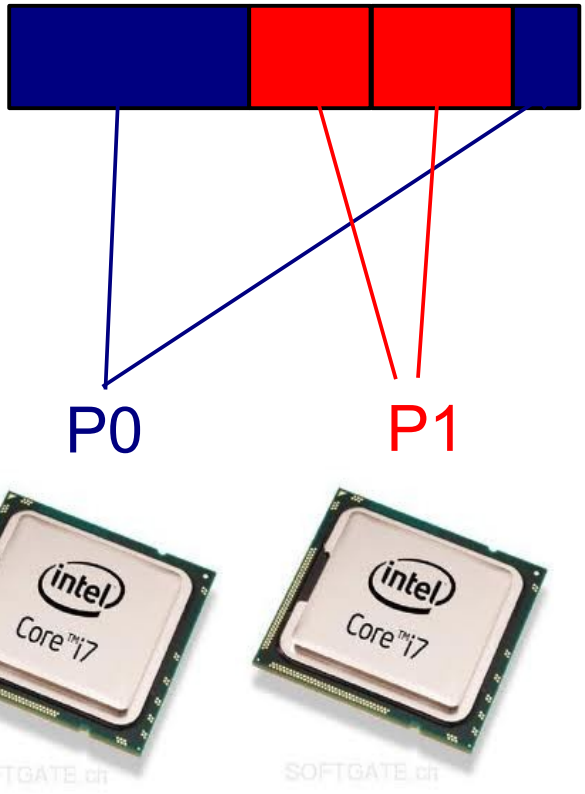


Works well when we have more partitions than processors:
May lead to load imbalance not all processors get equal number of sites!

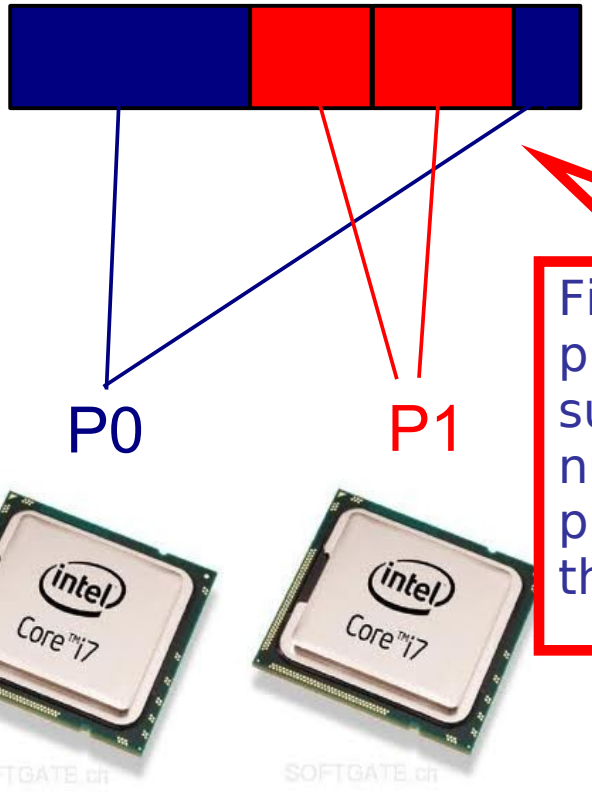
Load Balance I

P0 P1

Load Balance I



Load Balance I



Find the partition-to-processor assignment such that the maximum number of sites per processor is minimized → this is NP-hard

Load Balance I

- The **multiprocessor job scheduling problem** in phylogenetics
  - Problem when #partitions >> #cores
  - Tested per-site data distribution versus Longest Processing Time (LPT) heuristics
  - 25 taxa, 220,000 sites, 100 genes
    - GAMMA model

      naïve: **613** secs

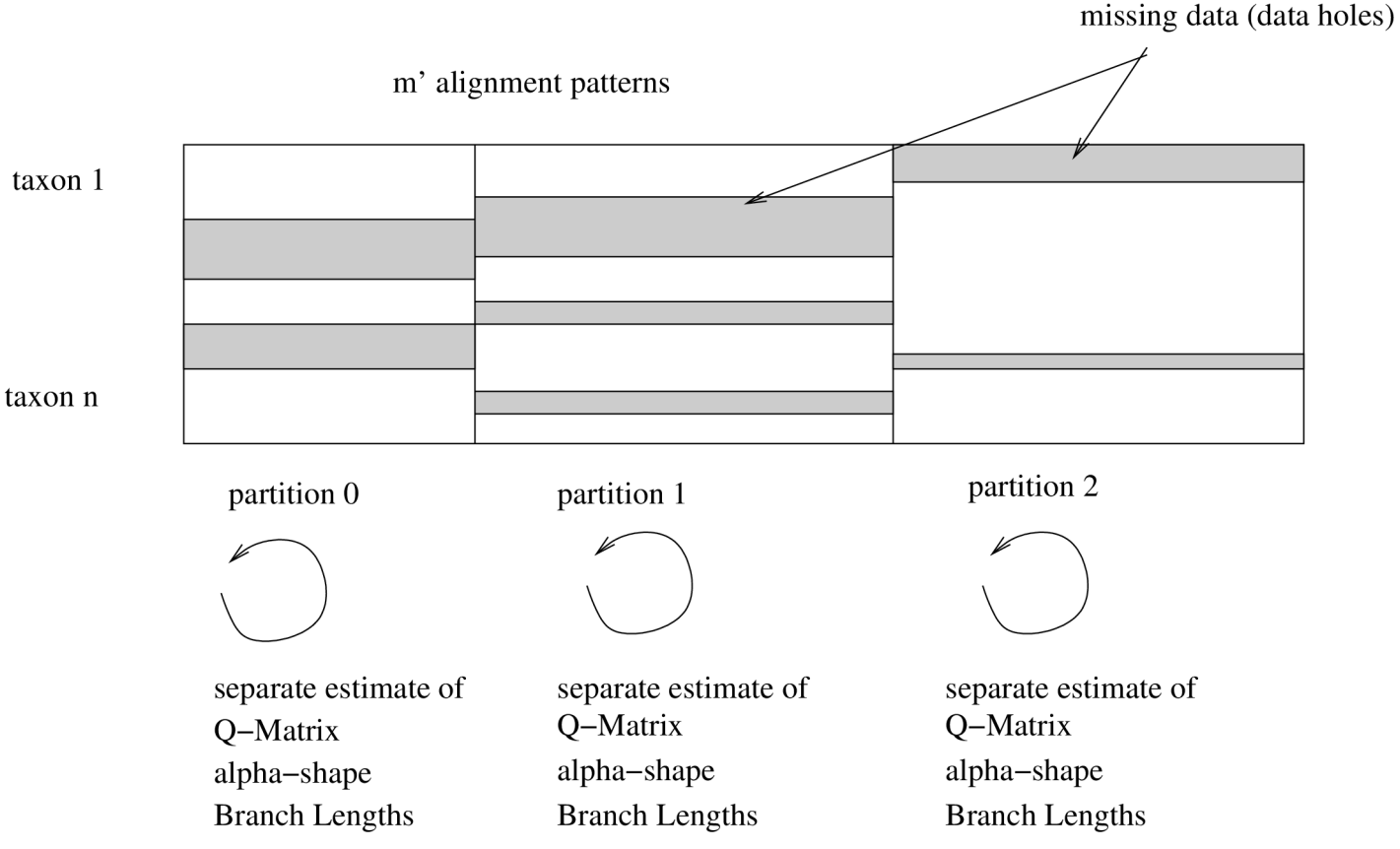
      LPT: **550** secs
    - CAT model

      naïve: **298** secs

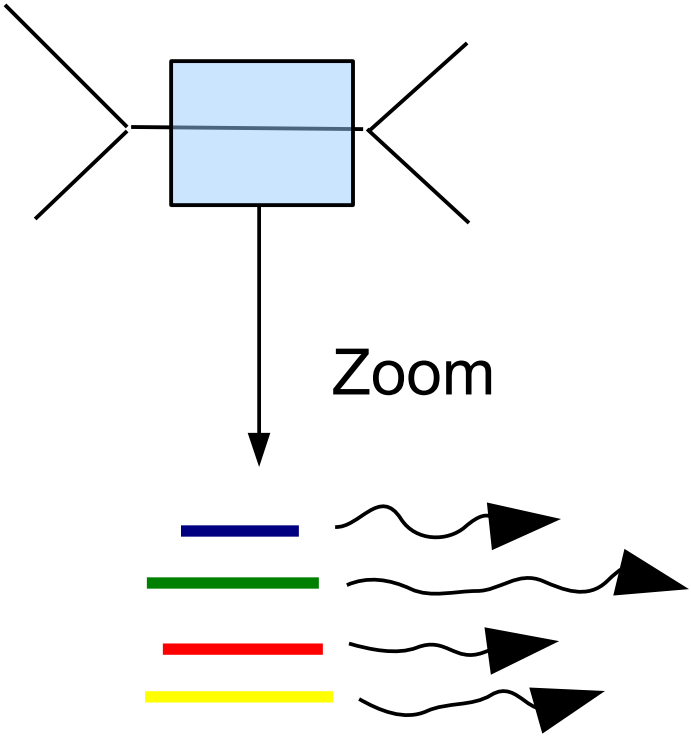
      LPT: **127** secs
  - Larger protein dataset under GAMMA: 10-fold performance improvement!

J. Zhang, A. Stamatakis: "The Multi-Processor Scheduling Problem in Phylogenetics", 11th IEEE HICOMB workshop (in conjunction with IPDPS 2012).

Partitioned Branch Lengths & other parameters



Load-Balance II



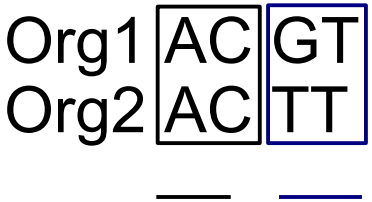
Synchronization Points

- Assume 10 branches
- Each branch requires 10 Newton-Raphson Iterations
- Each NR Iteration requires a syncronization via a reduction operation
- One branch/partition at a time: 100 sync. points, less work (only one partition) per sync. point
- All branches concurrently: 10 sync. points, more work per sync. point
- Branches will need distinct number of operations
- Add convergence state → bit vector

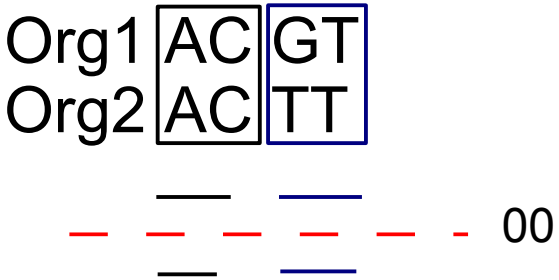
Synchronization Points

Org1 AC GT
Org2 AC TT

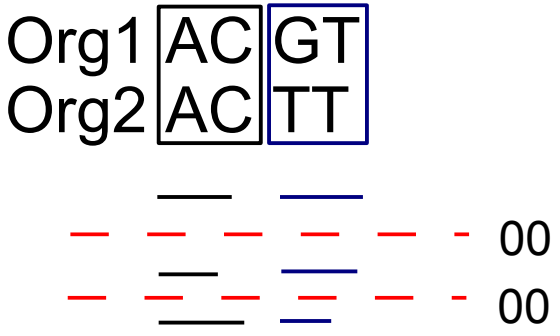
Synchronization Points



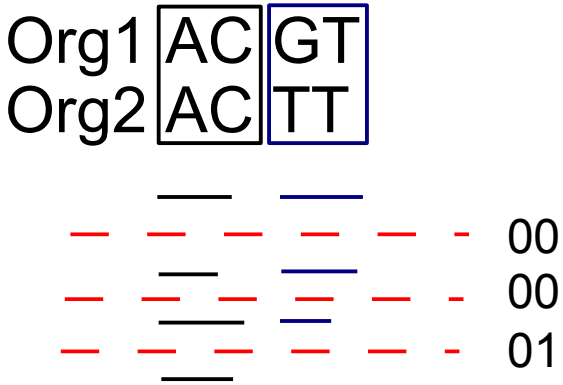
Synchronization Points



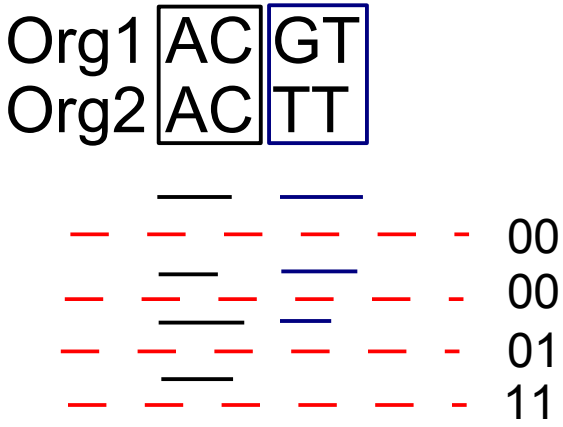
Synchronization Points



Synchronization Points

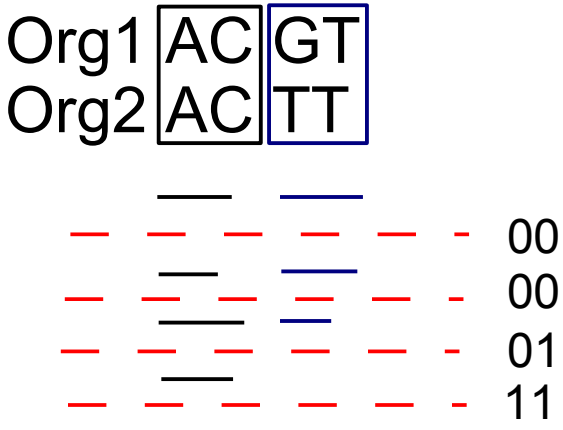


Synchronization Points



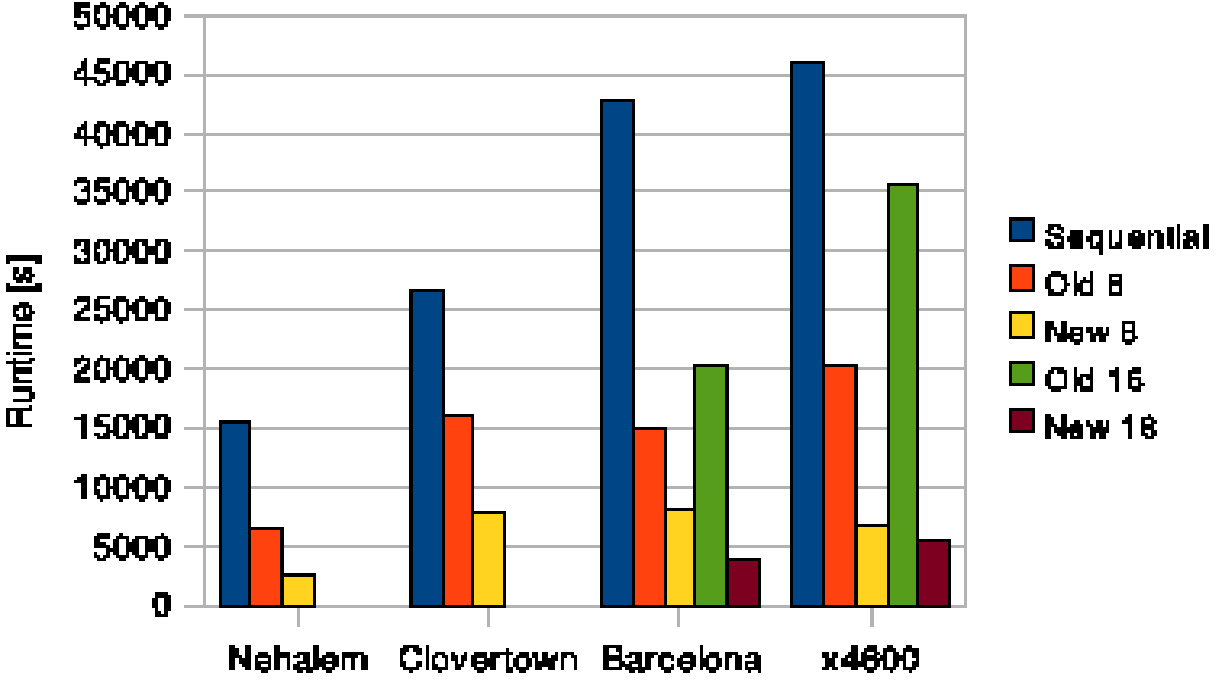
In this example: 4 instead of 7 sync points!

Synchronization Points



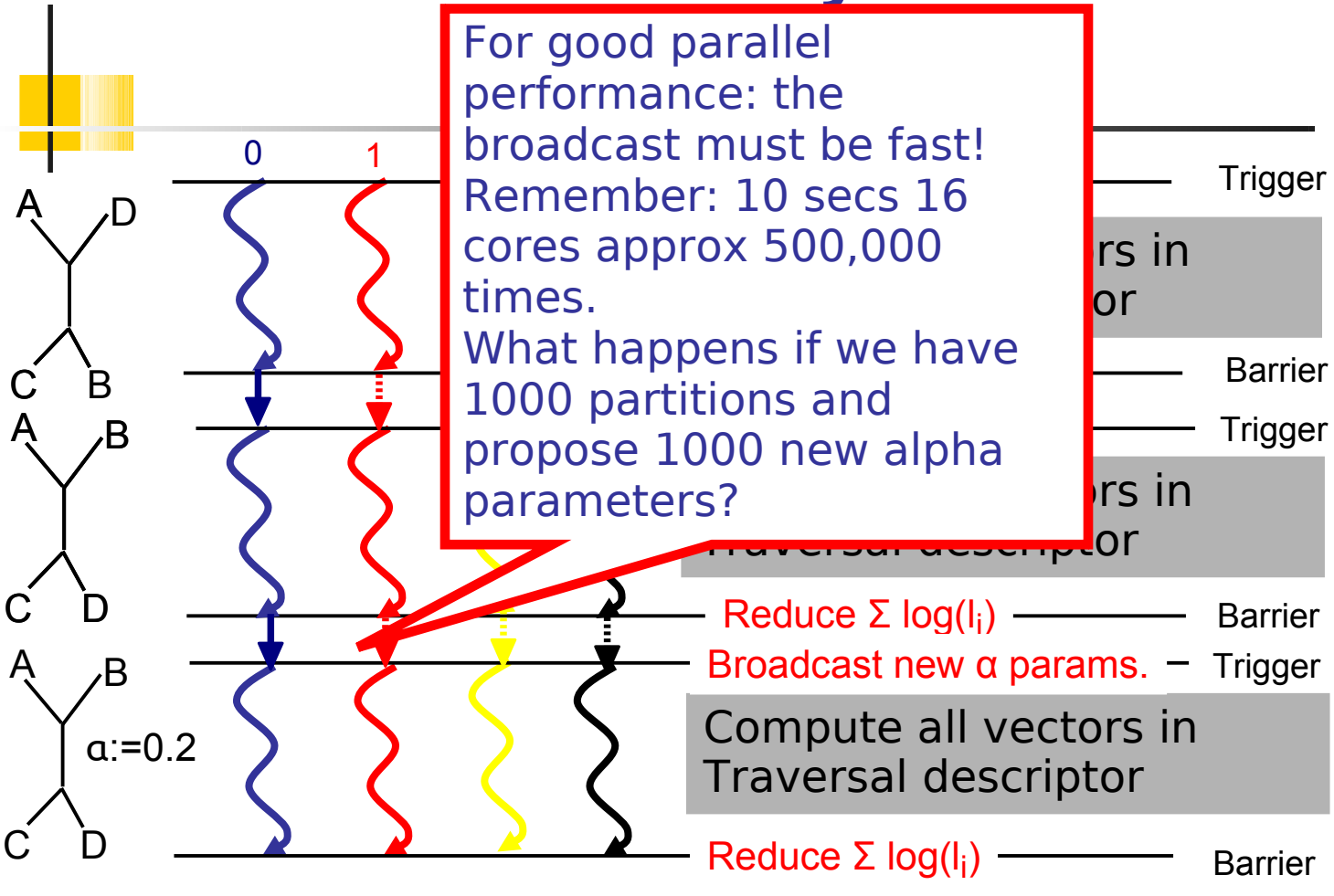
Implications for Bayesian programs: Propose e.g., new α parameters or new branch lengths simultaneously for all partitions!

Load Balance II

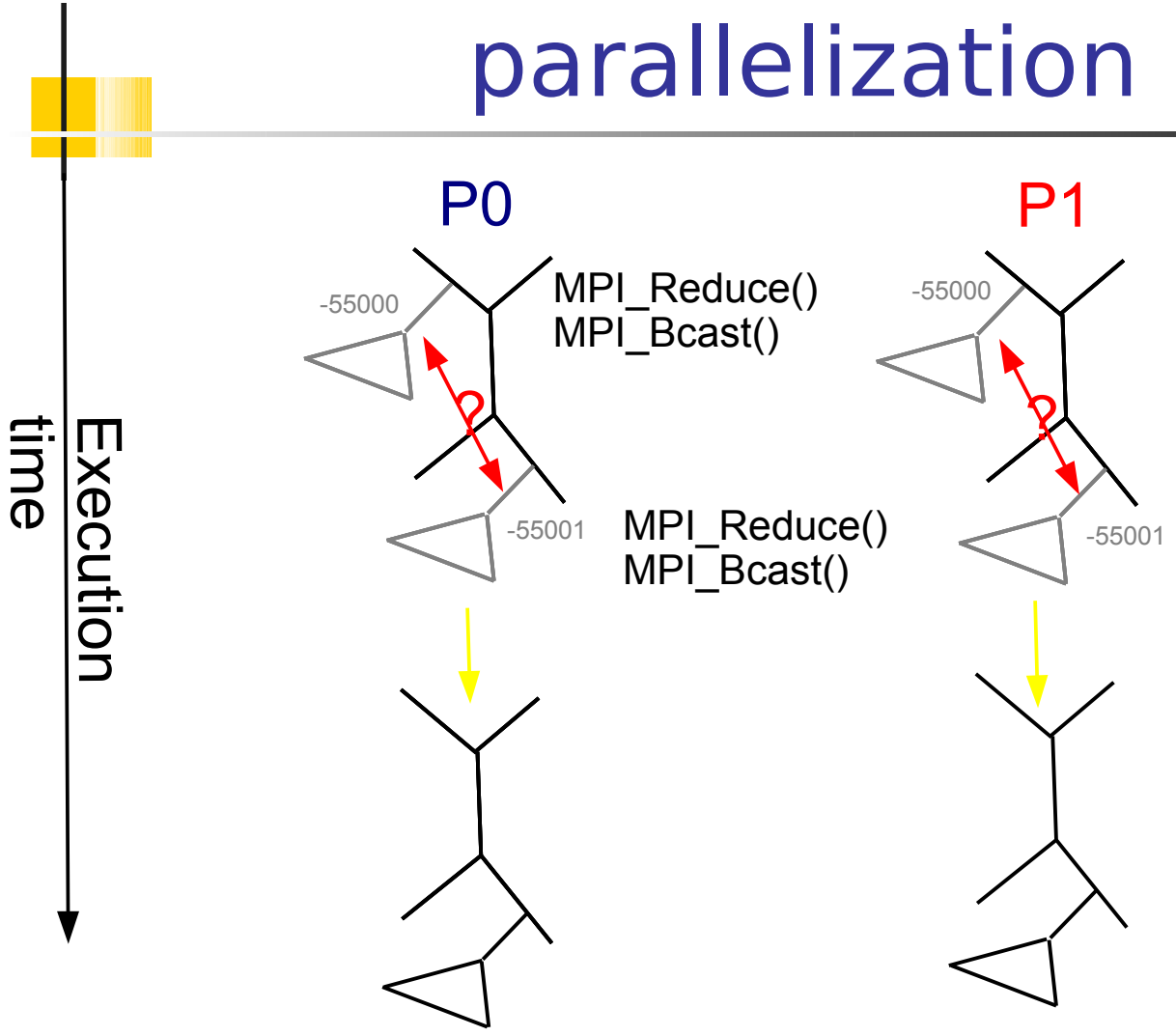


A. Stamatakis, M. Ott: "Load Balance in the Phylogenetic Likelihood Kernel". Proceedings of ICPP 2009, Vienna, Austria, September 2009.

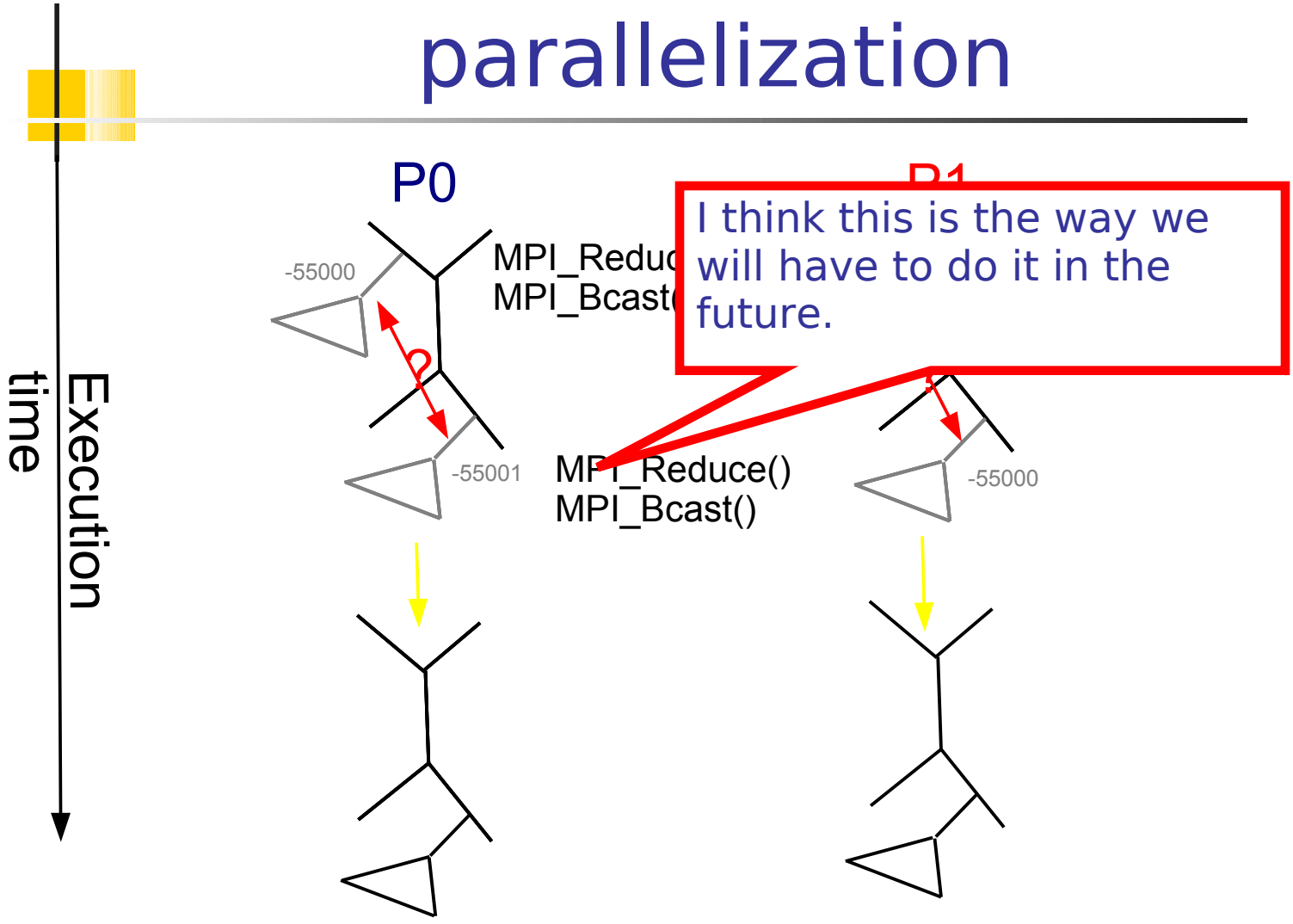
Classic Fork-Join with



Alternative MPI parallelization



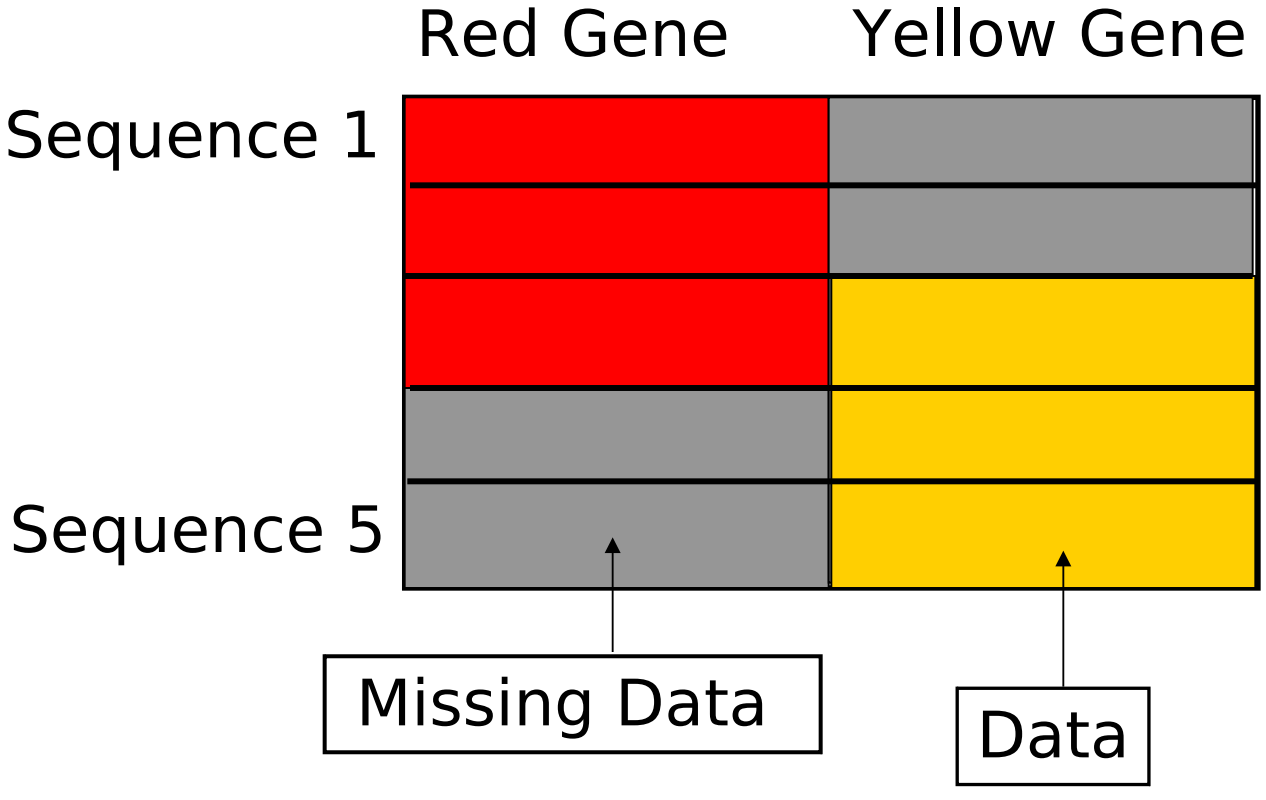
Alternative MPI parallelization



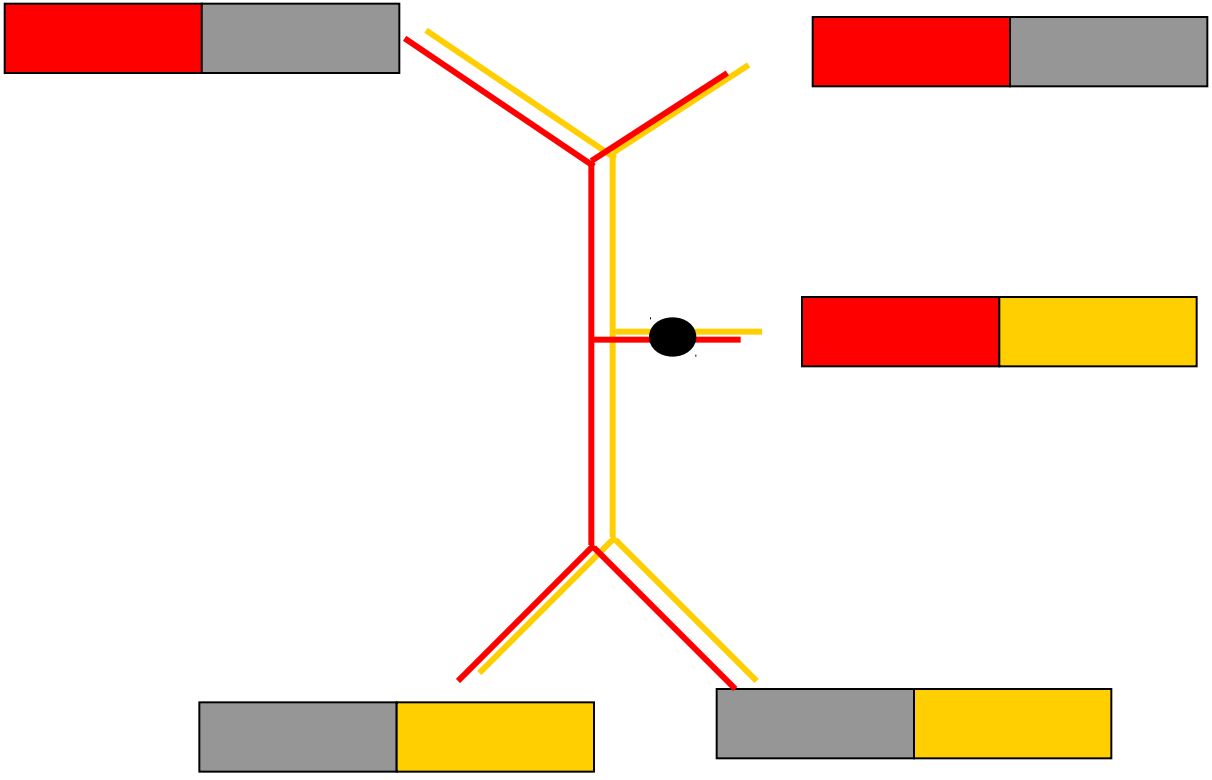
Outline

- Computing the Likelihood
- Optimizing & Parallelizing Likelihood Computations
- Saving Memory in Likelihood Computations
- HPC population genetics

Gappy Phylogenomic Alignments



A Multi-Gene Model



Initial mesh-based Approach

A. Stamatakis, N. Alachiotis: "Time and memory efficient likelihood-based tree searches on gappy phylogenomic alignments", Proceedings of ISMB 2010, Boston, Massachusetts, July 2010. In Bioinformatics, 26(12):i132-i139.

- Good performance
- Only works for per-partition branch length estimates
- Very high code complexity

  → only proof-of-concept implementation

  → production abandoned because too complex
- Does not work for dense datasets!
- How long will datasets stay gappy?

Initial mesh-based Approach

A. Stamatakis, N. Alachiotis: "Time and memory efficient likelihood-based tree searches on gappy phylogenomic alignments", Proceedings of ISMB 2010, Boston, Massachusetts, July 2010. In Bioinformatics, 26(12):i132-i139.

- Good performa
- Only w s
- Very hi
  → only
  → prod
- Does n
- How lo

Closely-related theoretical work on gappy datasets:
M. Sanderson, M. McMahon, M. Steel: "Terraces in Phylogenetic Tree Space", *Science* 2011.
The RAxML prrof-of-concept implementation now prints out the number of terrace moves encountered during a search.

Easier Memory-saving Techniques

- Resurrection of subtree equality vectors for gappy alignments

  A. Stamatakis, T. Ludwig, H. Meier, M.J. Wolf: “AxML: A Fast Program for Sequential and Parallel Phylogenetic Tree Calculations Based on the Maximum Likelihood Method”. In Proceedings of 1st IEEE Computer Society Bioinformatics Conference (CSB2002), 21–28, Palo Alto, California, August 2002.

- Out-of-core execution/external memory algos

  F. Izquierdo-Carrasco, A. Stamatakis: "Computing the Phylogenetic Likelihood Function Out-of-Core", accepted for publication at IEEE HICOMB 2011 workshop (held in conjunction with IPDPS 2011), Anchorage, USA, May 2011.

- Trading memory for computations

  F. Izquierdo-Carrasco, J. Gagneur, A. Stamatakis: "Trading Memory for Running Time in Phylogenetic Likelihood Computations", Bioinformatics 2012 conference, Vilamoura, Portugal, February 2012.

Easier Memory-saving Techniques

- Resurrection of subtree ...uality vectors for gappy alignments

  A. Stamatakis, T. Ludwig, H. Meier, M.J. V
  Parallel Phylogenetic Tree Calculations B
  Proceedings of 1st IEEE Computer Socie
  Palo Alto, California, August 2002.

These techniques can be used in all likelihood-based programs!

- Out-of-core execution/external memory algos

  F. Izquierdo-Carrasco, A. Stamatakis: "Computing the Phylogenetic Likelihood Function Out-of-Core", accepted for publication at IEEE HICOMB 2011 workshop (held in conjunction with IPDPS 2011), Anchorage, USA, May 2011.

- Trading memory for computations

  F. Izquierdo-Carrasco, J. Gagneur, A. Stamatakis: "Trading Memory for Running Time in Phylogenetic Likelihood Computations", Bioinformatics 2012 conference, Vilamoura, Portugal, February 2012.

Subtree Equality Vectors An old idea revisited

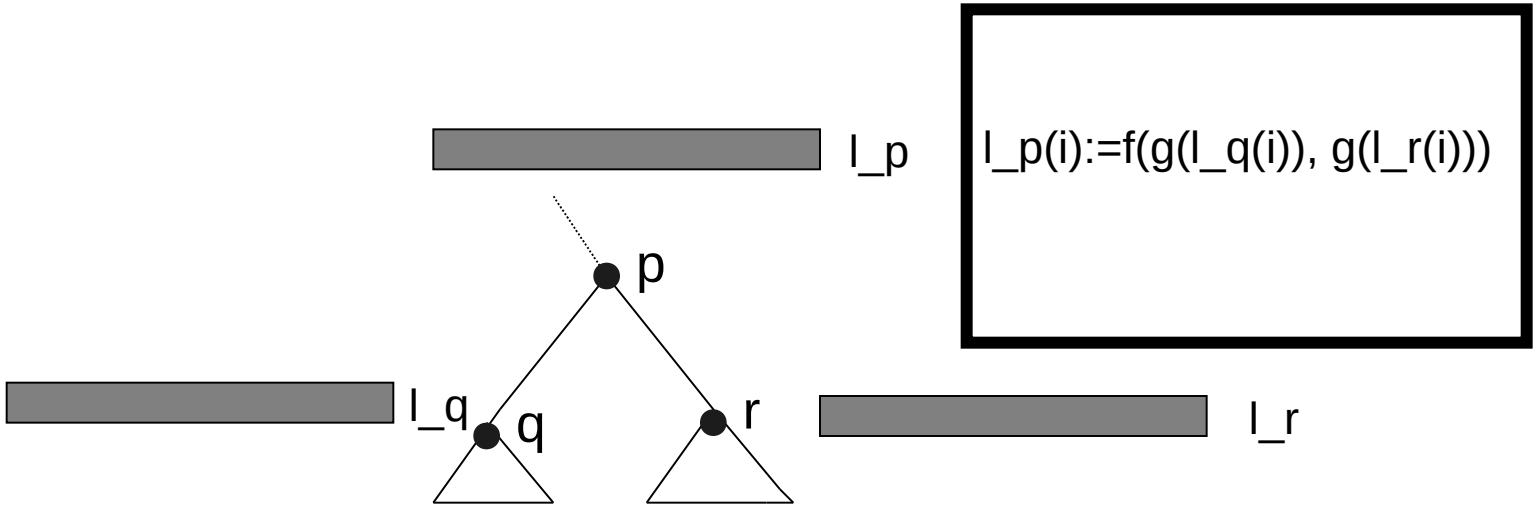
- Acceleration of Likelihood-function by detecting (and avoiding recomputation) of identical patterns in subtrees
- Detecting identical patterns may induce high bookkeeping overhead
- Conditional in innermost likelihood kernel loop may perturb pre-fetching and branch prediction
- Ancient papers

  Stamatakis *et al.* “Accelerating Parallel Maximum Likelihood-based Phylogenetic Tree Calculations using Subtree Equality Vectors”, Supercomputing **2002**.

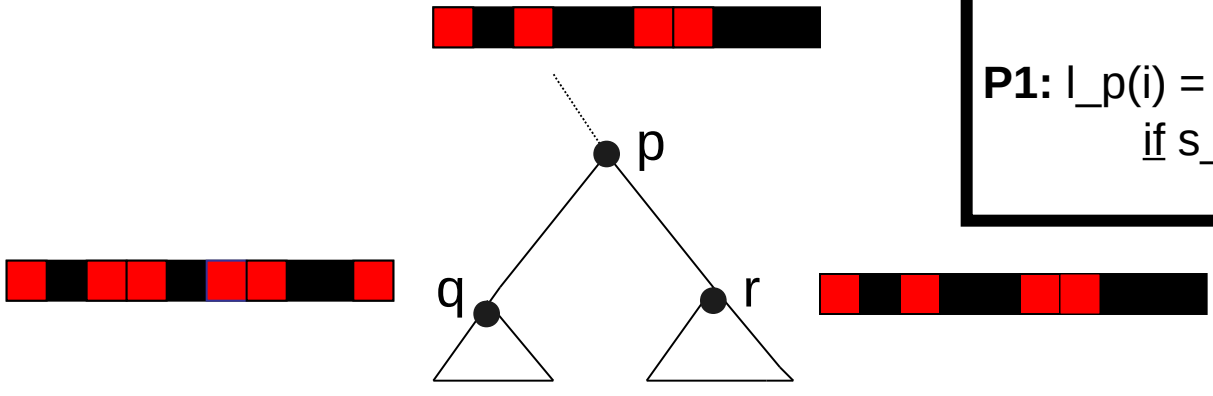
  Stamatakis *et al.* “AxML: A Fast Program for Sequential and Parallel Phylogenetic Tree Calculations based on the Maximum Likelihood Method”, **CSB2002**.

  Some old paper by Fredrik from the 90ies?

Subtree Equality Vectors



Subtree Equality Vectors



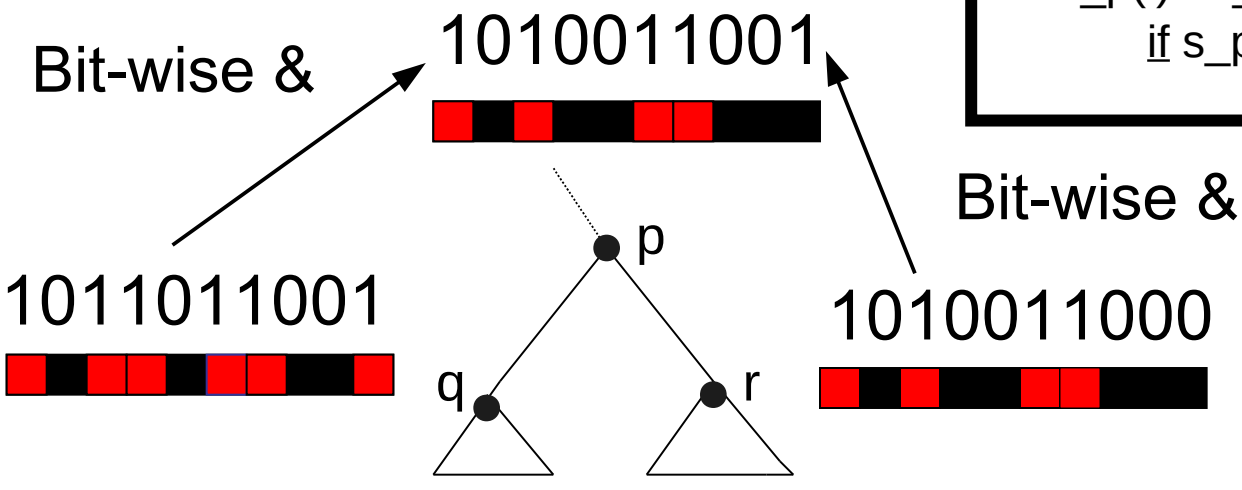
l_p(i):=f(g(l_q(i)), g(l_r(i)))

P1: l_p(i) = l_p(j)
if s_p(i) = s_p(j)

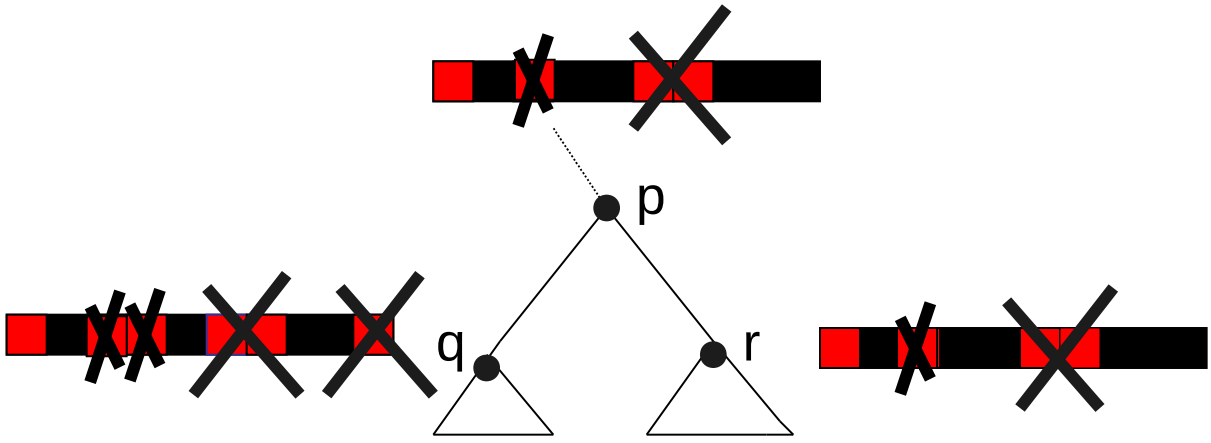
Subtree Equality Vectors

l_p(i):=f(g(l_q(i)), g(l_r(i)))

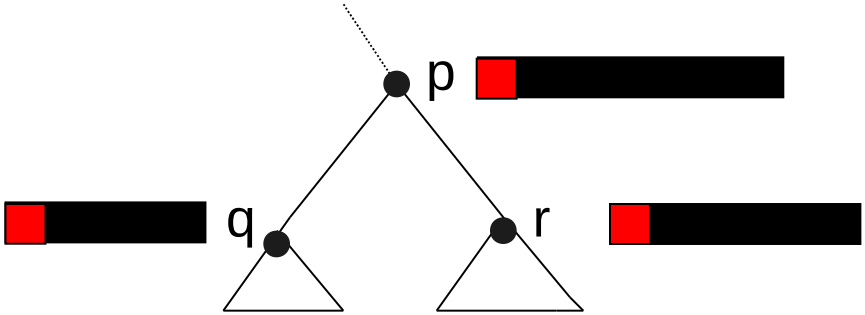
P1: l_p(i) = l_p(j)
if s_p(i) = s_p(j)



Subtree Equality Vectors Implementation Option 1



Subtree Equality Vectors Implementation Option 2

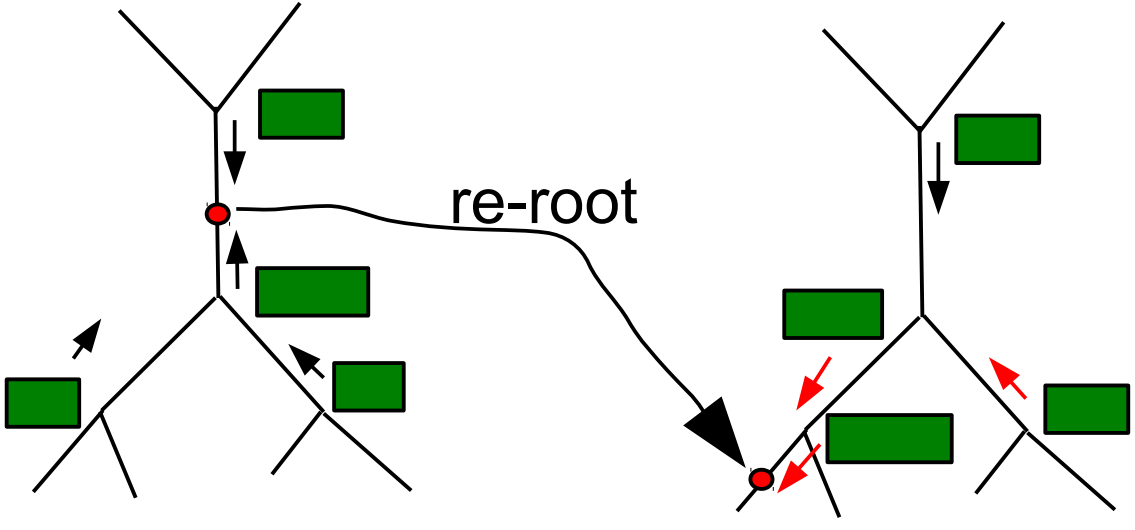


Initial Results
Dataset Gappyness ≈ 80%

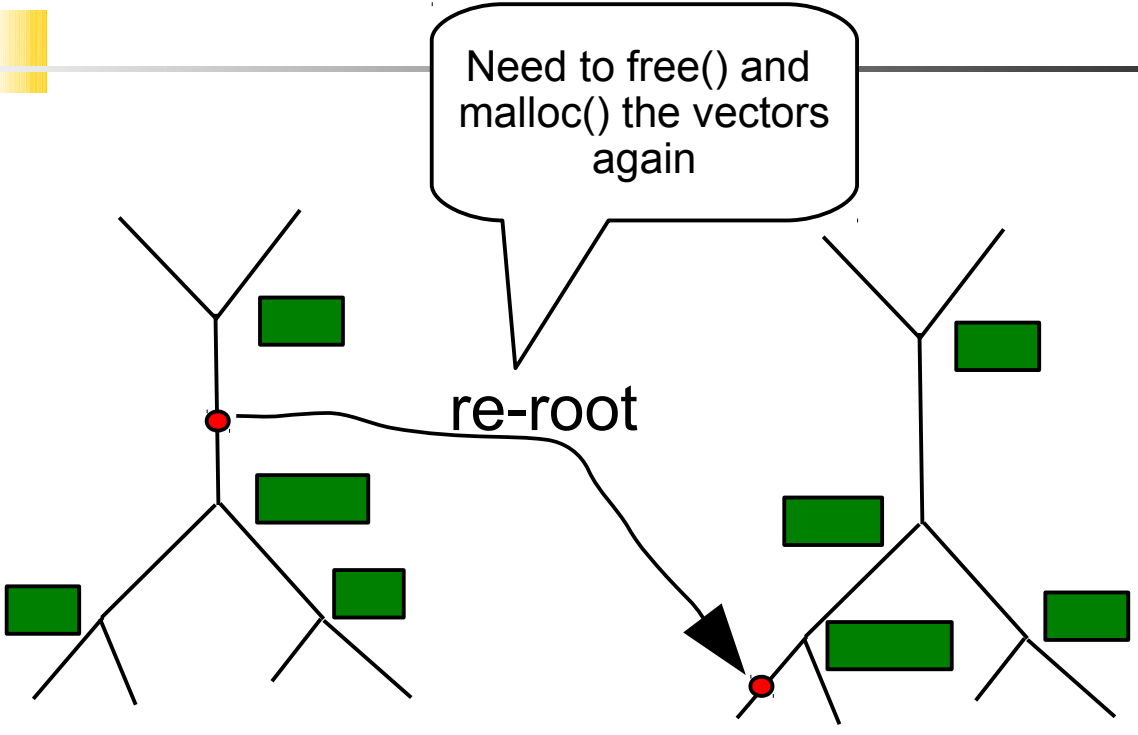
37831 taxa			
	SEVs	SEVs with memory saving	standard
Runtime (s)	4125.1	4116.8	6541.1
Memory (GB)	42	15	41
LogLikelihood	-5528590	-5528590	-5528590

55593 taxa			
	SEVs	SEVs with memory saving	standard
Runtime (s)	7145.2	8095.1	11181.4
Memory (GB)	67	29	67
log likelihood	-7059556	-7059556	-7059556

Memory Allocation SEVs



Memory Allocation SEVs



Frequent free() and malloc() in Pthreads version

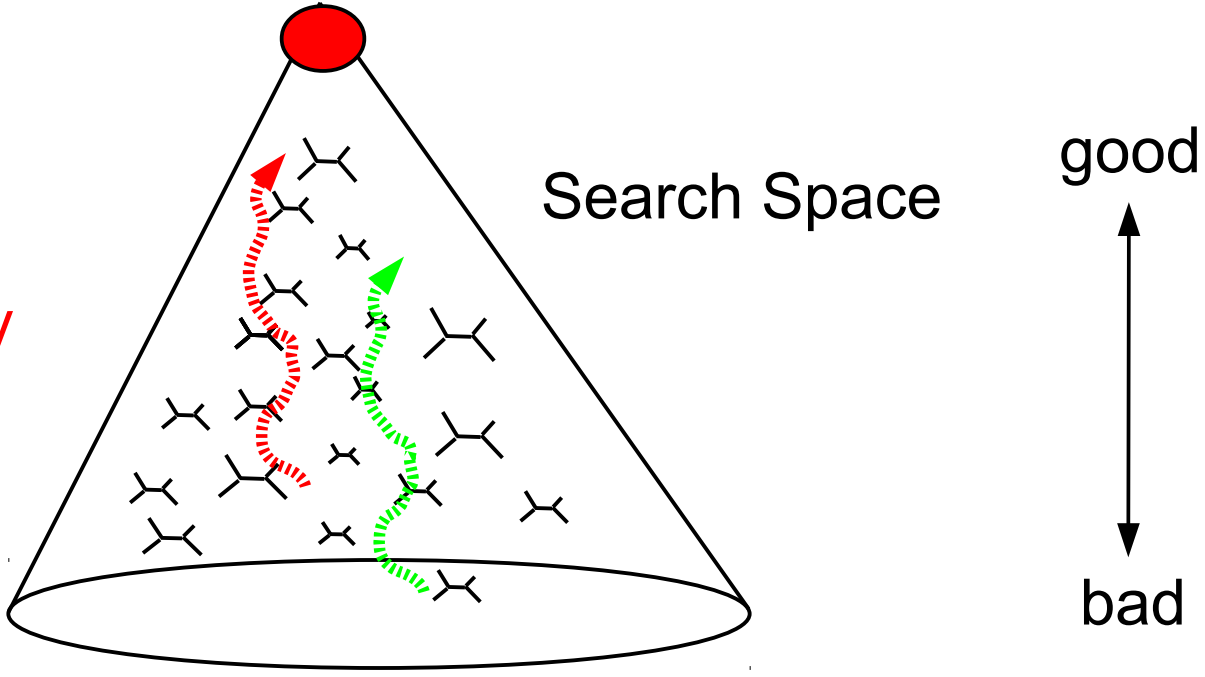
- Can lead to performance degradation → global lock
- Solution: use multi-core allocators
  - AMD Magny 48-core server 256GB
  - 33,000 species, 8000 sites
  - Standard malloc(): 33,400 secs
  - Facebook jemalloc(): 18,200 secs

Subtree Equality Vectors Summary

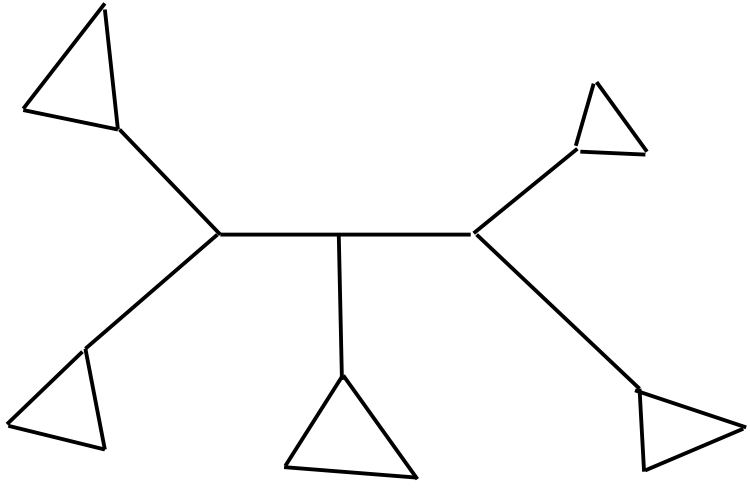
- Memory savings depend on
  - Proportion of missing data in input
- Speedups depend on
  - Flops saved versus bookkeeping overhead
  - Implementation of malloc() and free() on multicores when memory saving is used in the Pthreads version

    ≈ 50% faster with dedicated allocators

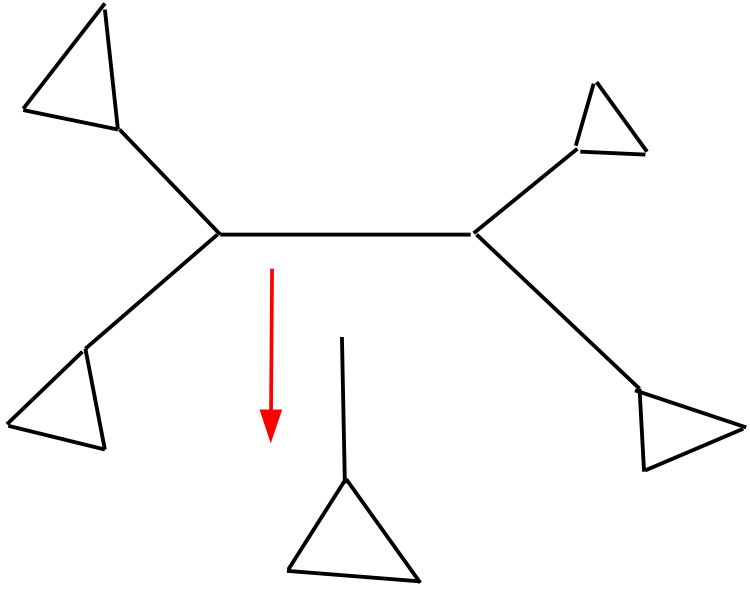
Search Strategies
ML Analyses



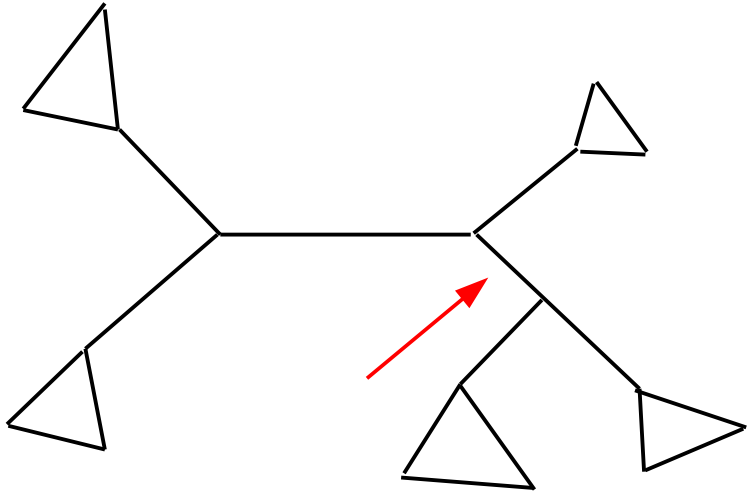
Tree Search



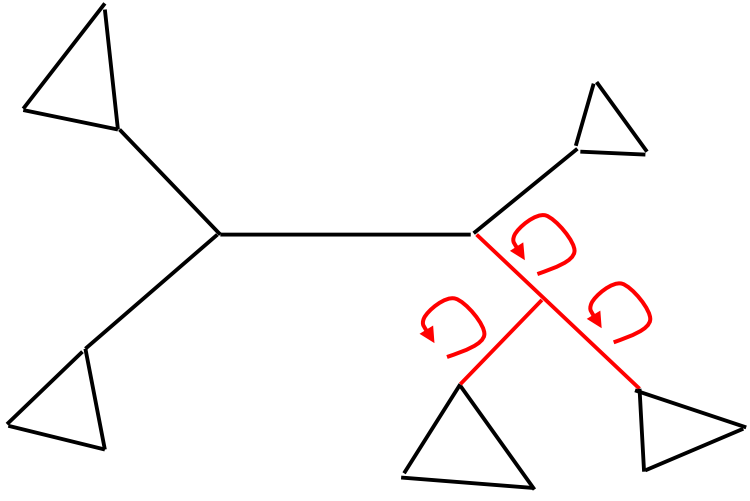
Tree Search



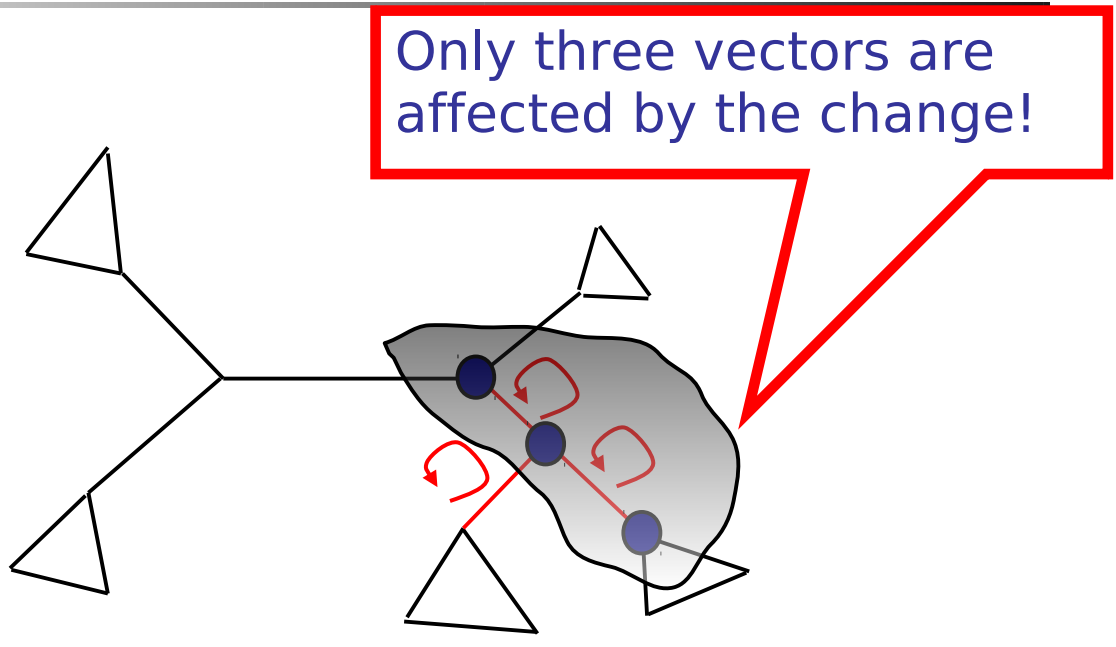
Tree Search



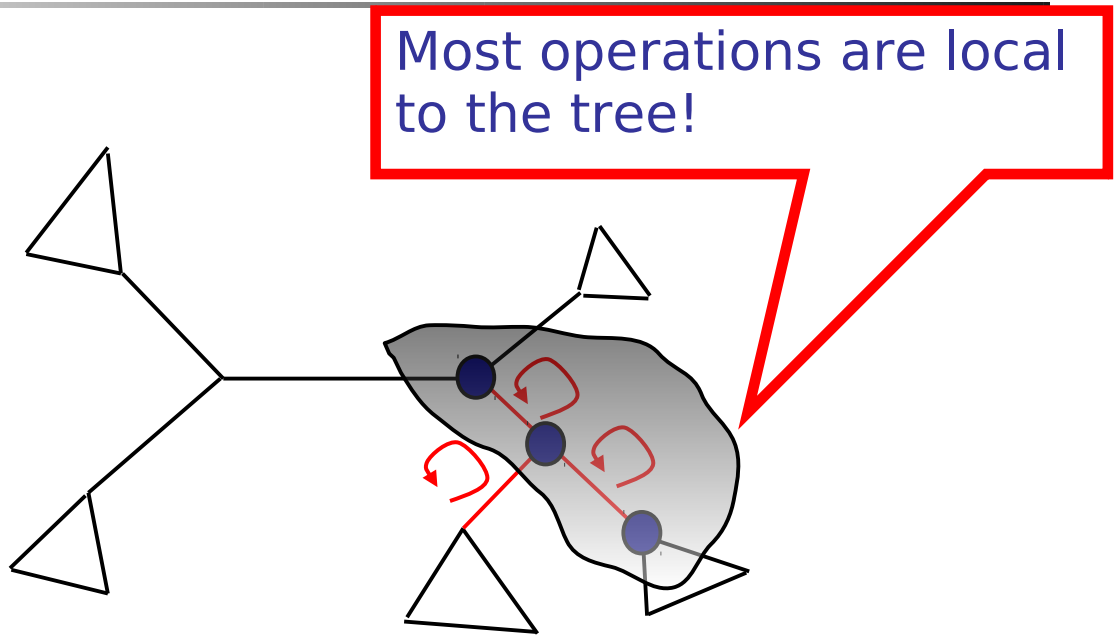
Tree Search



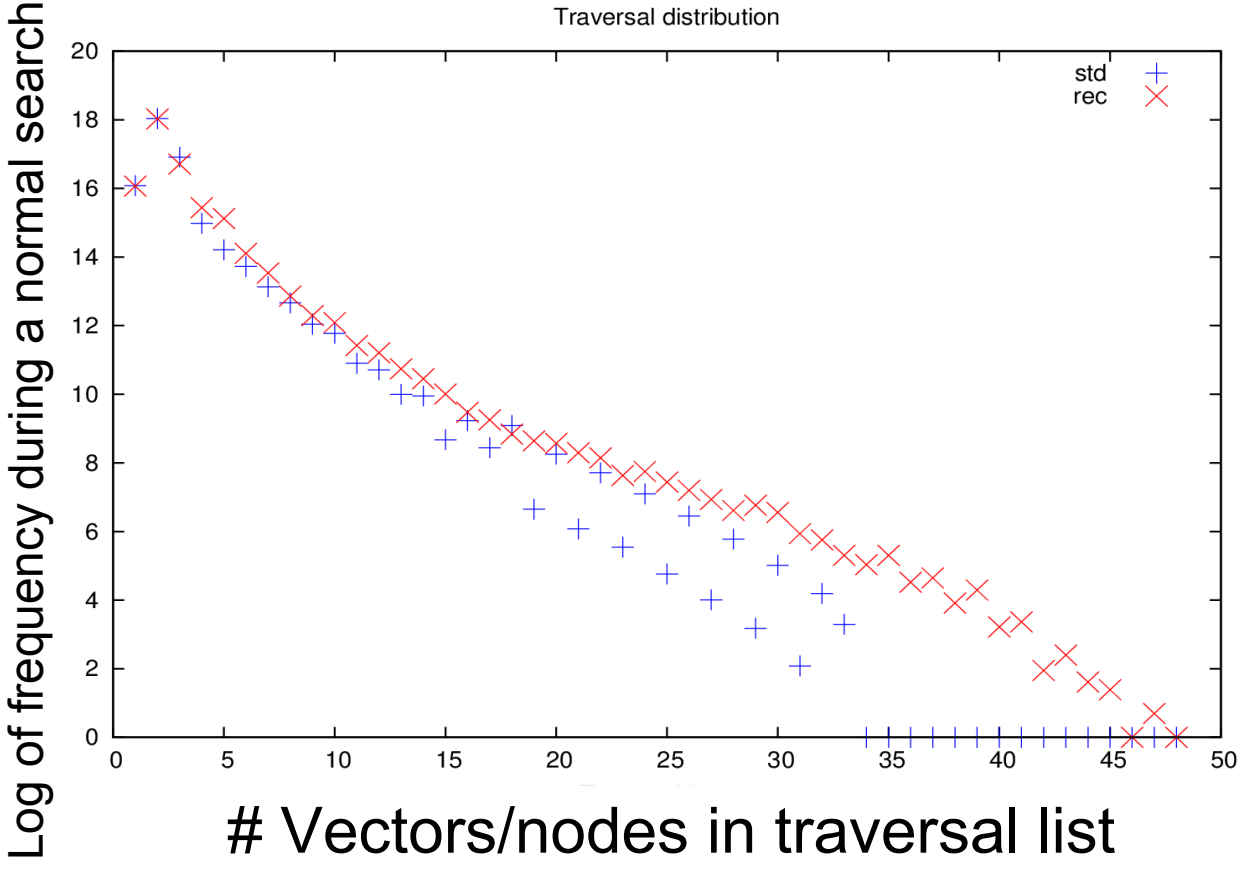
Tree Search



Tree Search



Traversal Lengths



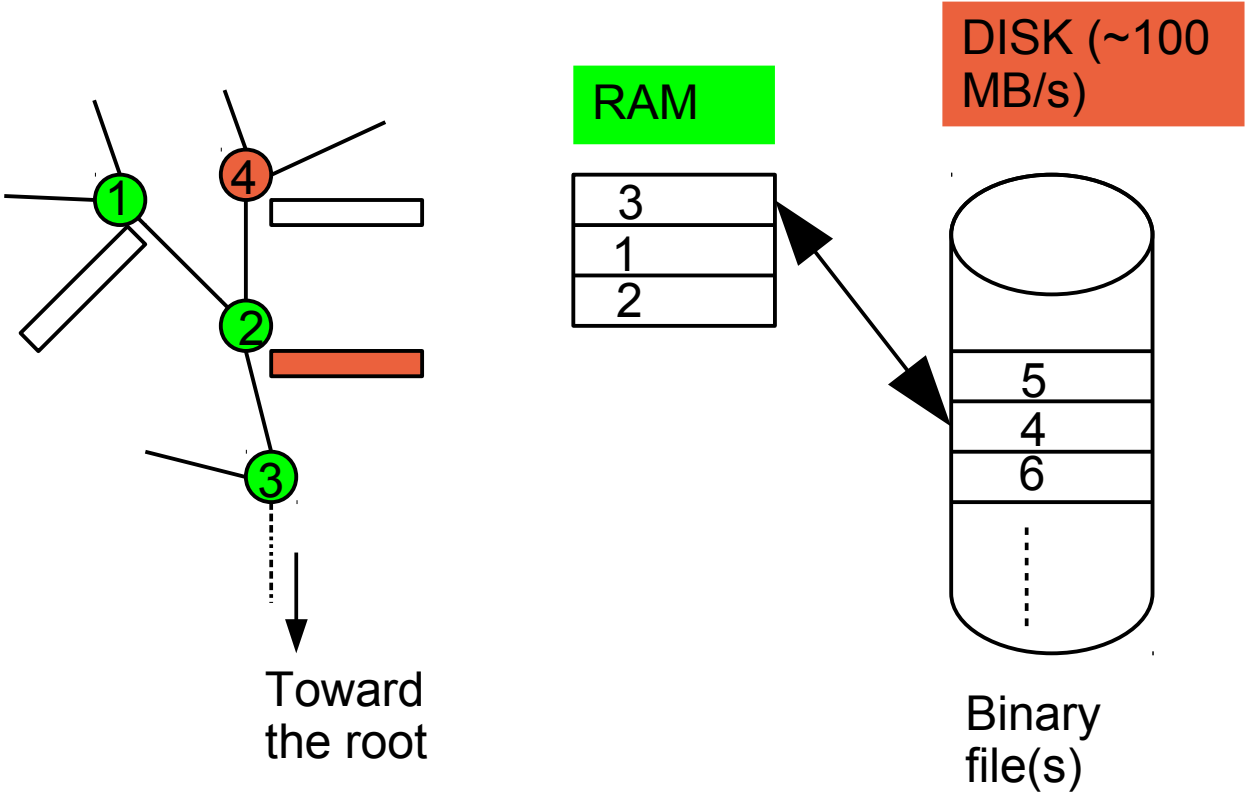
Out-of-core

- Likelihood computations exhibit high data locality

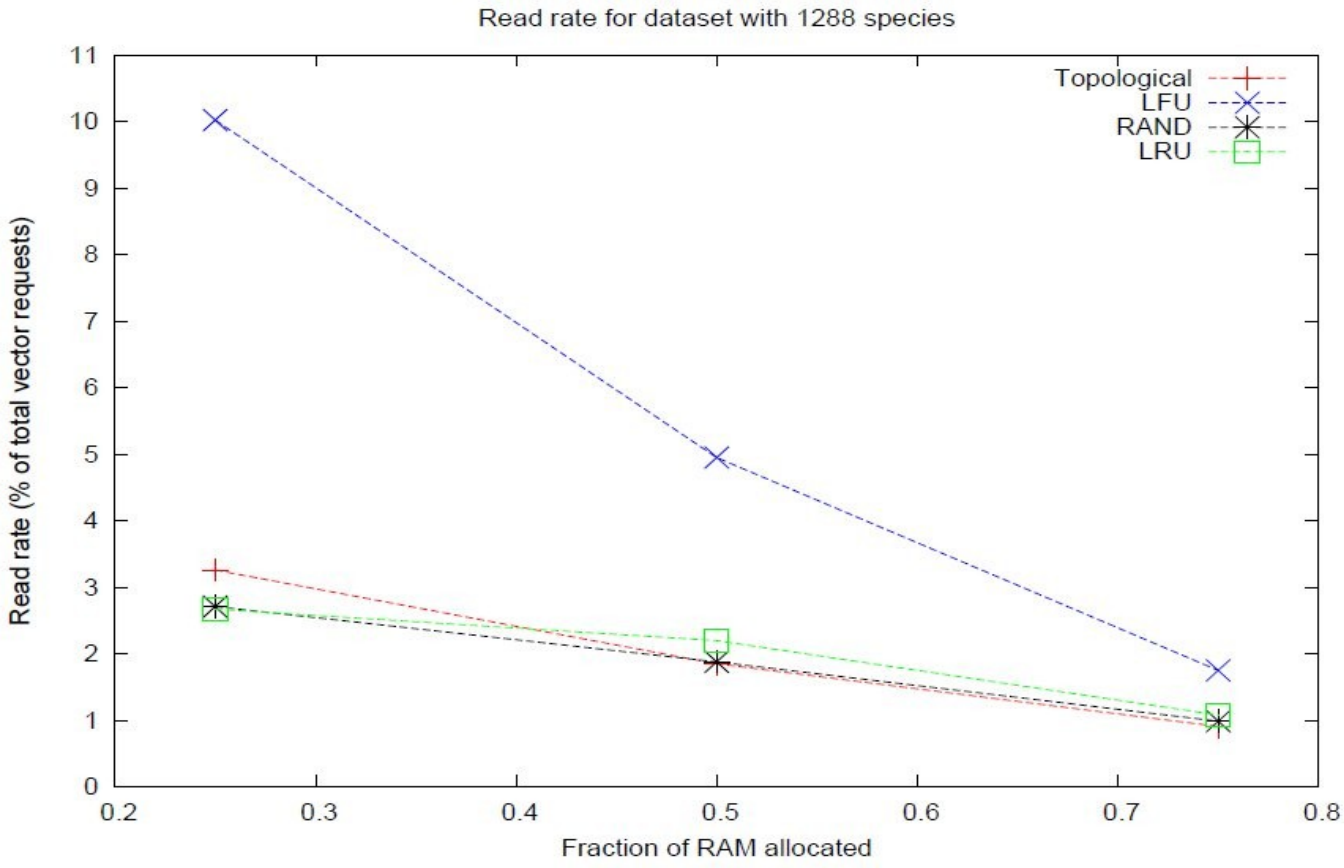
  → mostly short local traversals

  → expect low miss rate

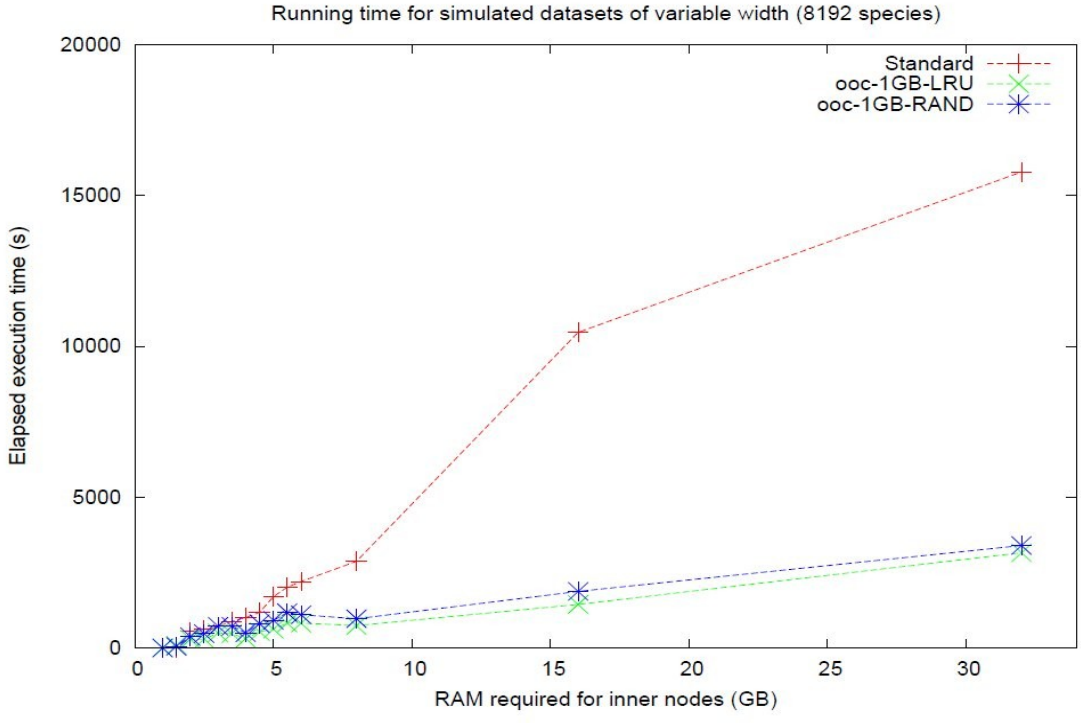
Out of core



Miss Rate


Read rate for dataset with 1288 species
Topological
LFU
RAND
LRU
Read rate (% of total vector requests)
Fraction of RAM allocated
0
1
2
3
4
5
6
7
8
9
10
11
0.2
0.3
0.4
0.5
0.6
0.7
0.8

Performance



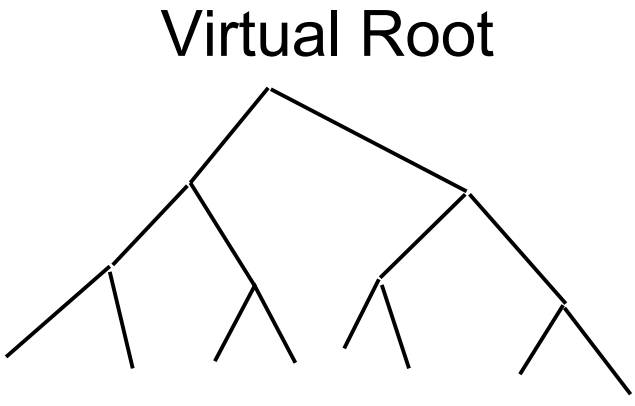
Summary: Out-of-core

- Performance substantially better than for paging
- … but still disappointing, not practical
- Works for dense datasets!
- **Future Work**: test with solid state disks

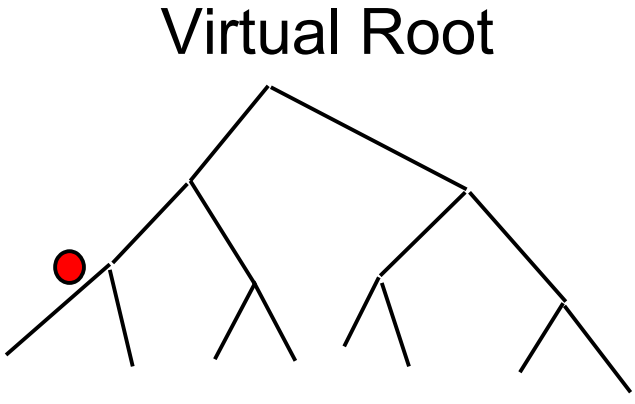
Recomputing Vectors instead of storing them

- Out of core: high data locality of vectors
- Proof: we only need ***log(#taxa) + 2*** vectors in memory to compute the likelihood on any unrooted binary tree
- Performance does not depend on gappyness!
- Can be combined with the orthogonal SEV approach

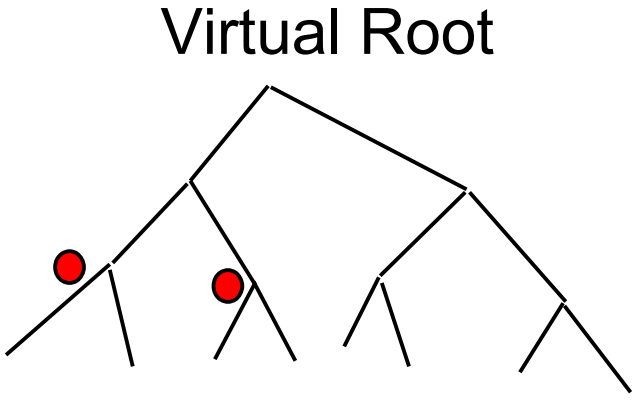
Tree Shapes



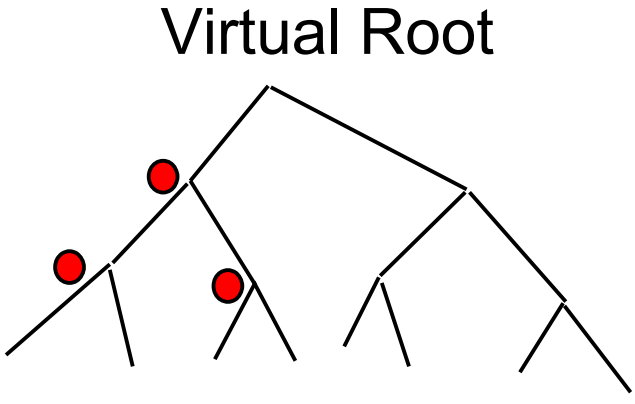
Tree Shapes



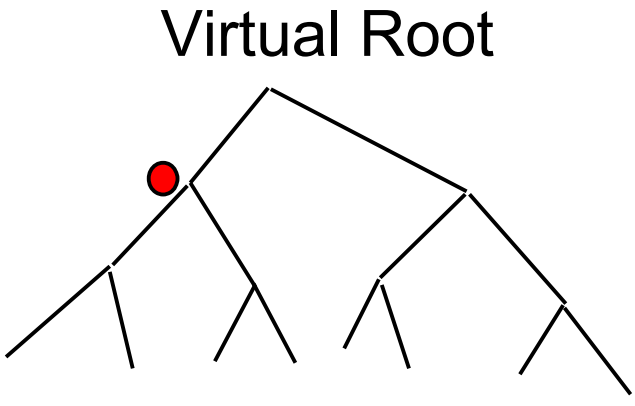
Tree Shapes



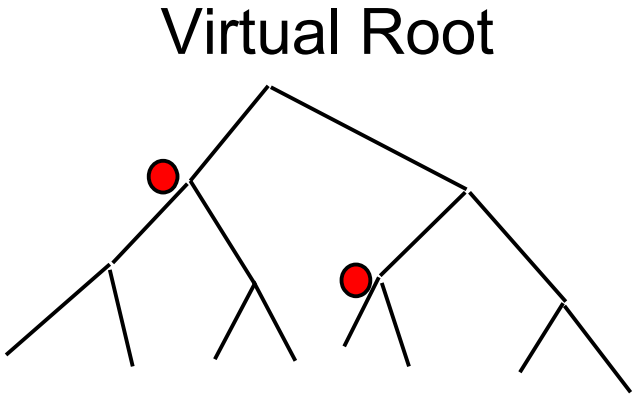
Tree Shapes



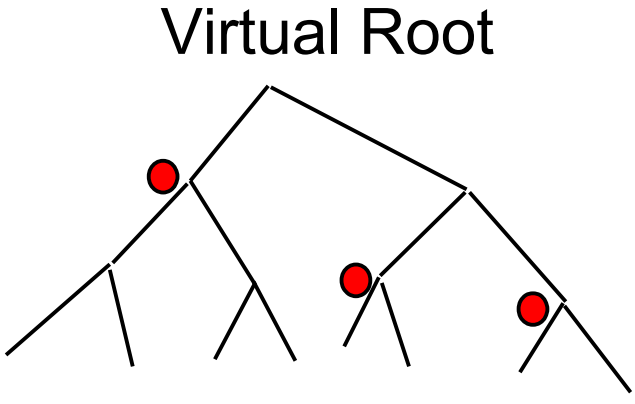
Tree Shapes



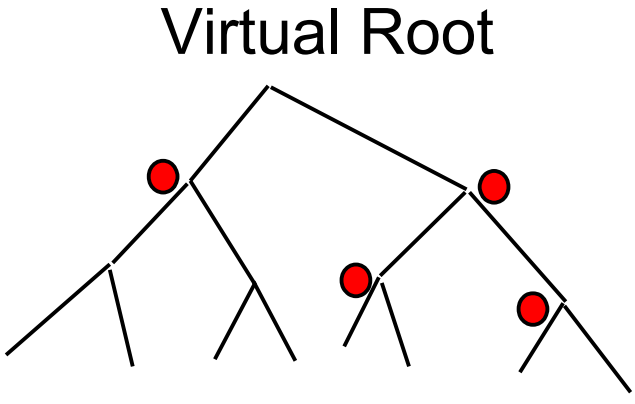
Tree Shapes



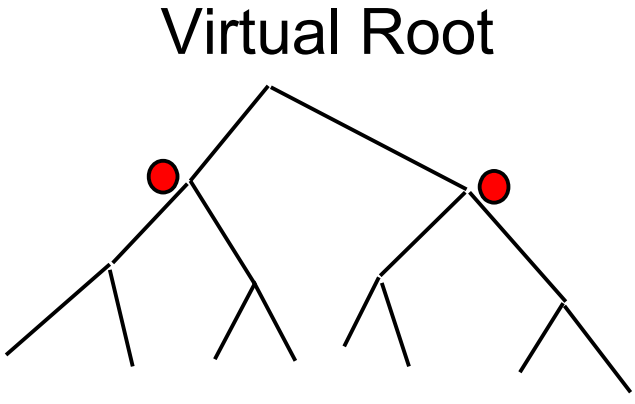
Tree Shapes



Tree Shapes

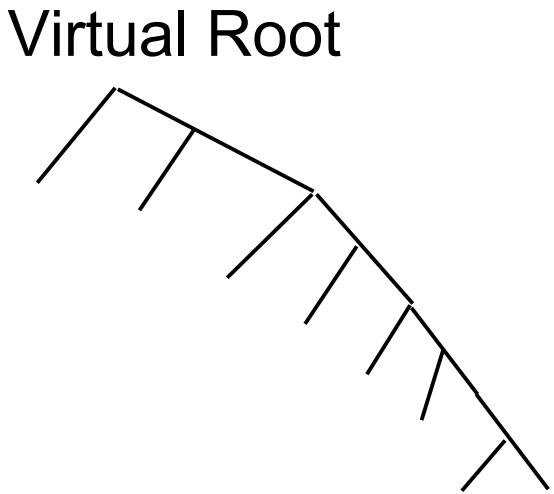


Tree Shapes

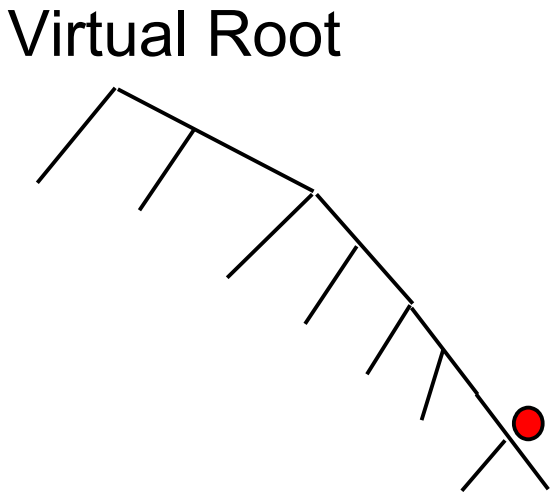


Max number of required vectors: 4

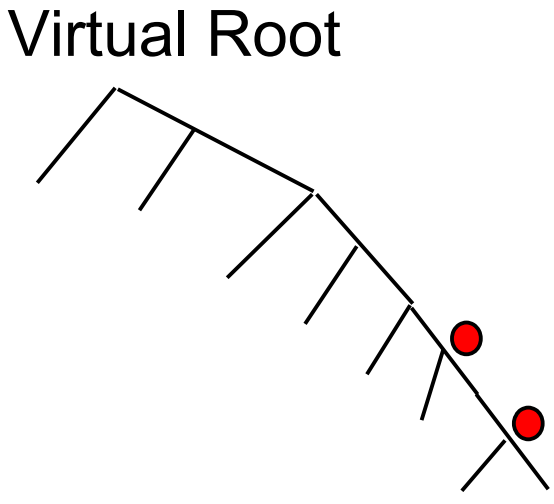
Tree Shapes



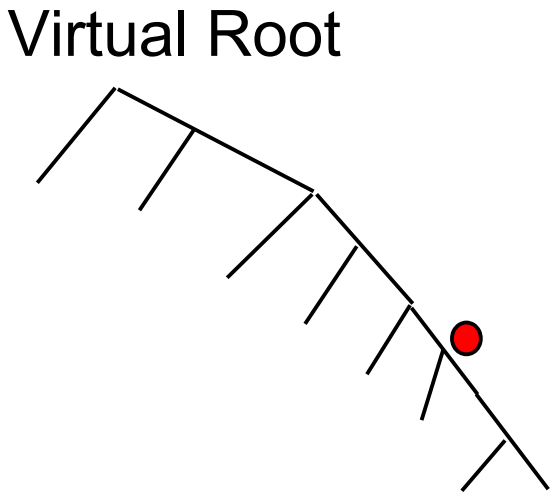
Tree Shapes



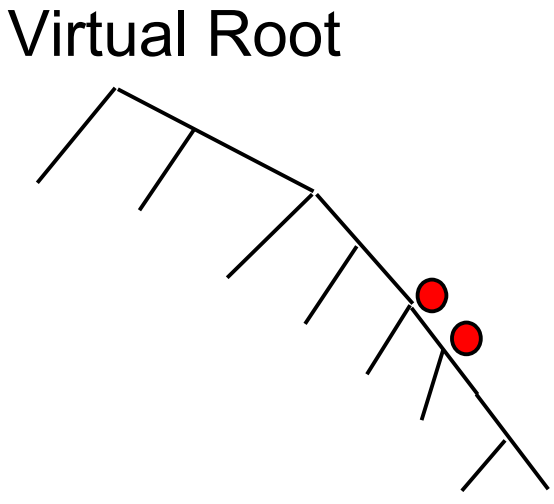
Tree Shapes



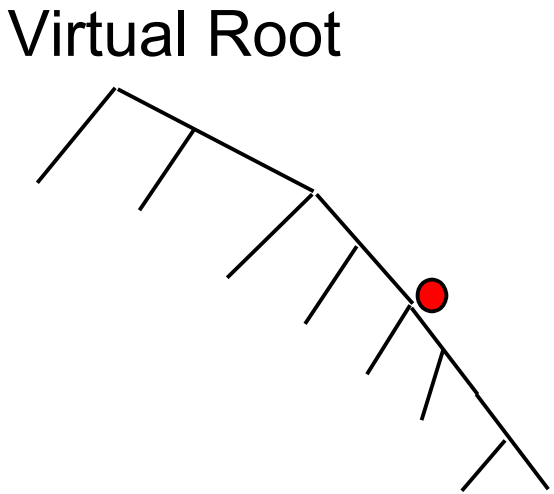
Tree Shapes



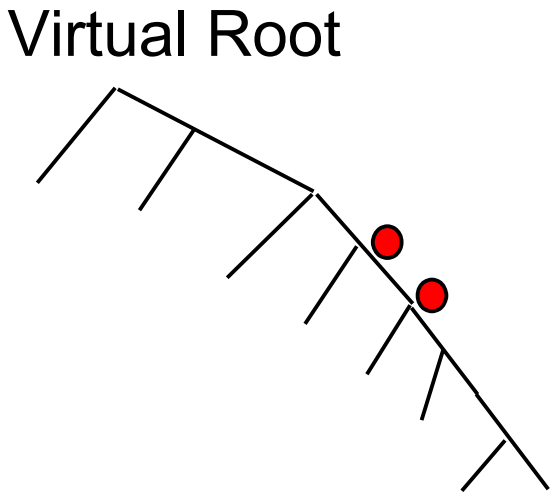
Tree Shapes



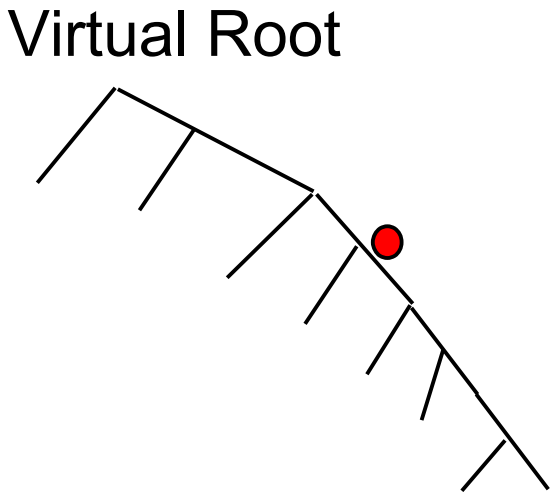
Tree Shapes



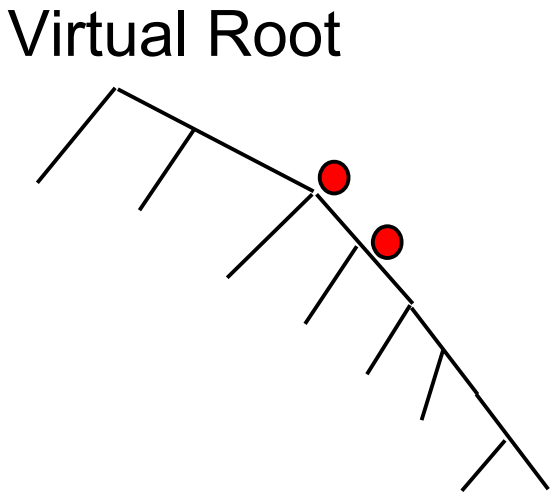
Tree Shapes



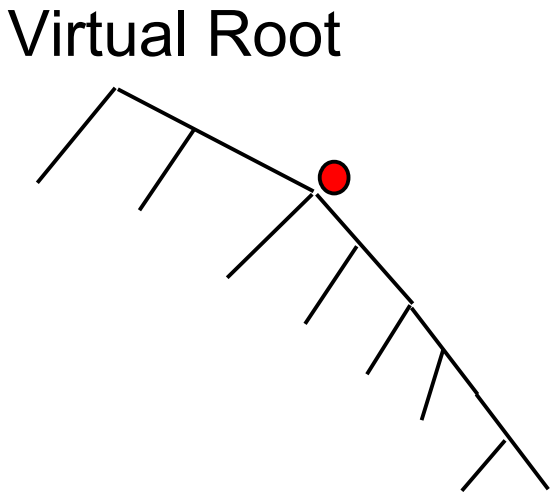
Tree Shapes



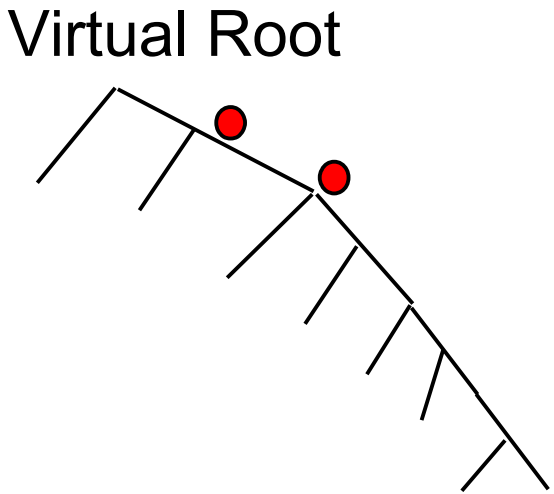
Tree Shapes



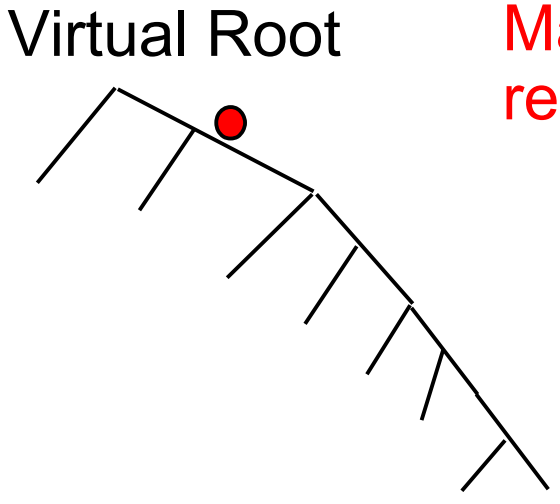
Tree Shapes



Tree Shapes

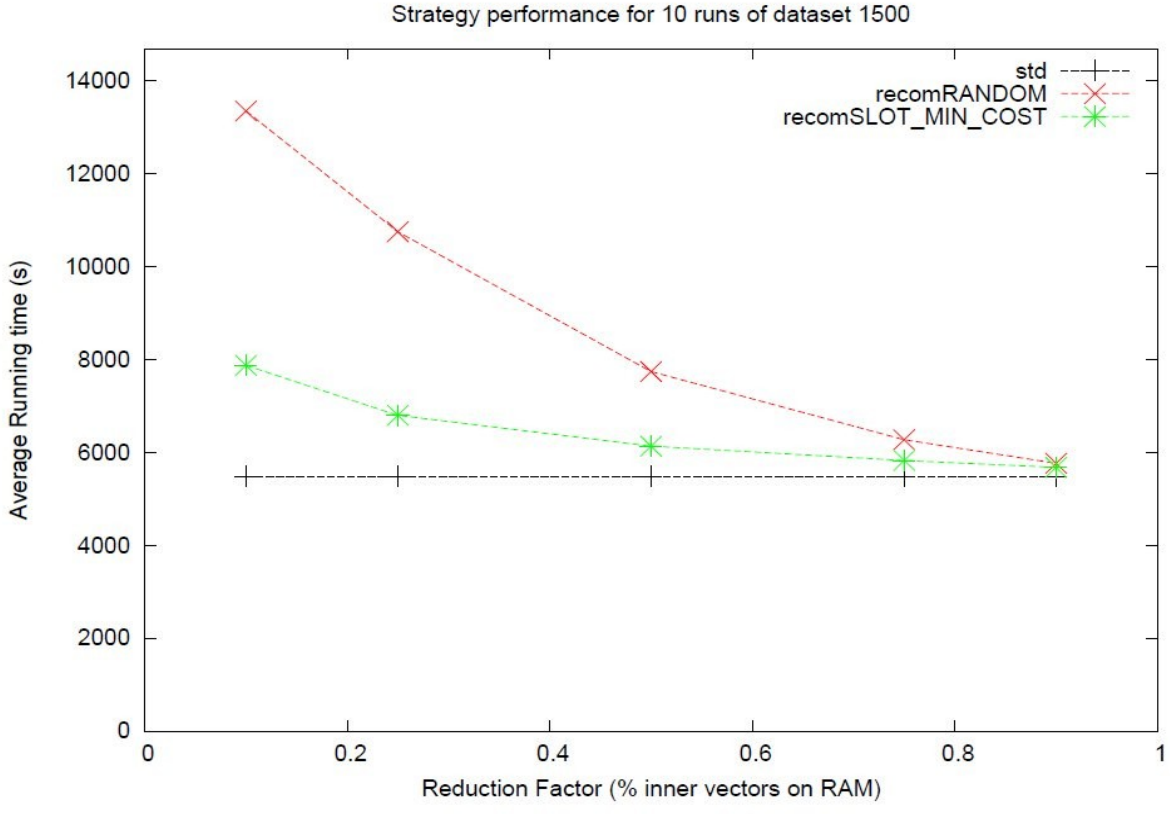


Tree Shapes



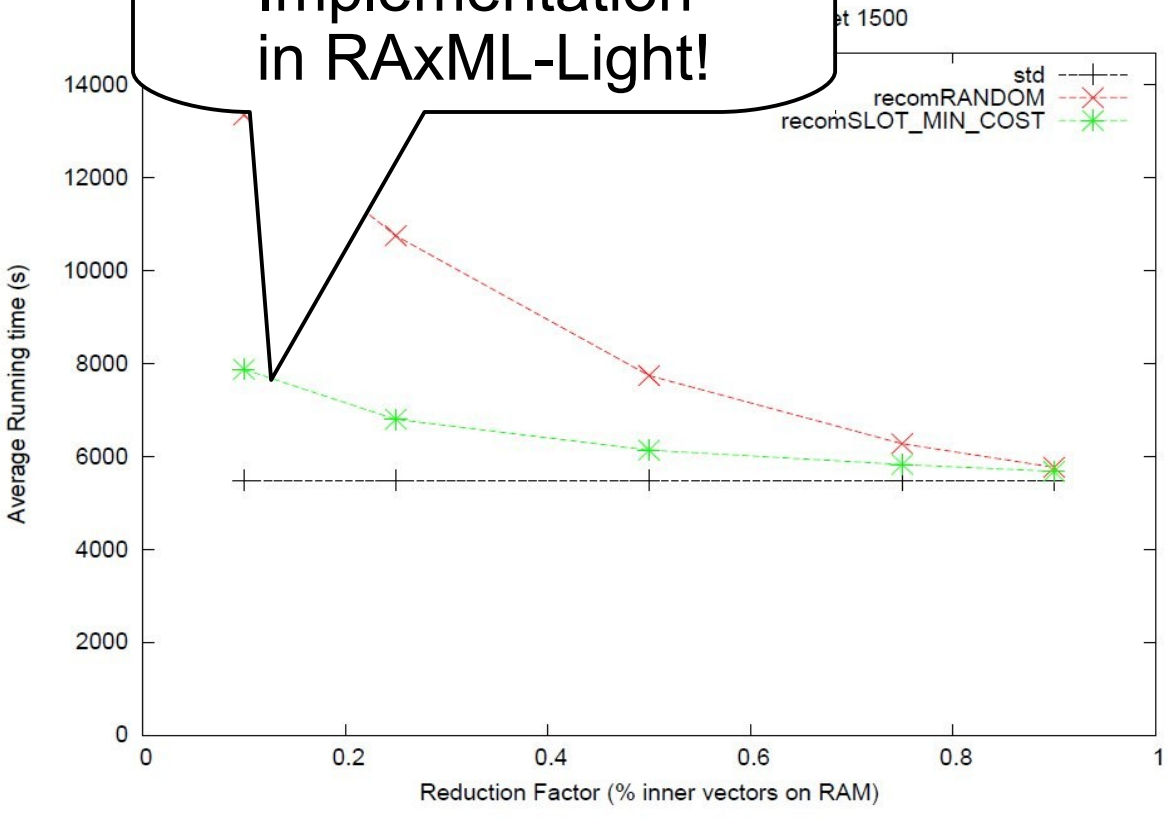
Max number of required vectors: 2

Results

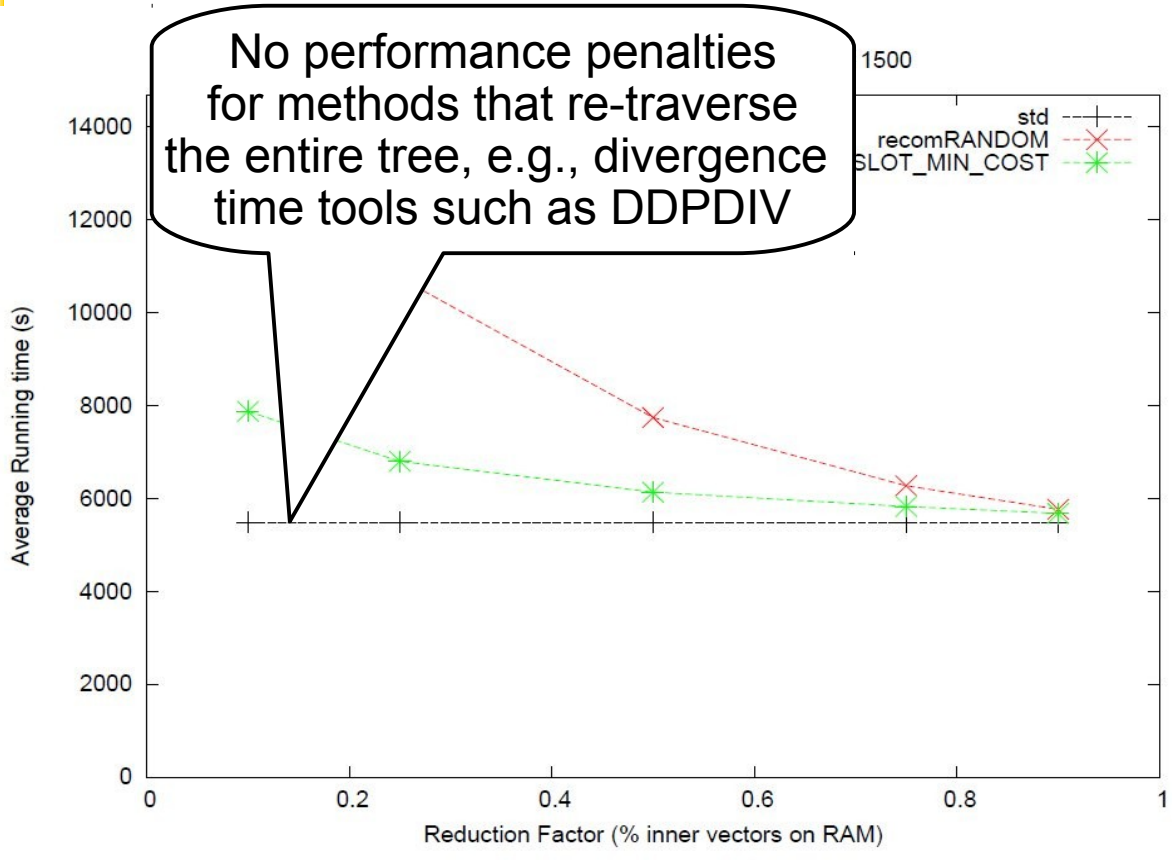

Strategy performance for 10 runs of dataset 1500
std
recomRANDOM
recomSLOT_MIN_COST
Average Running time (s)
14000
12000
10000
8000
6000
4000
2000
0
0
0.2
0.4
0.6
0.8
1
Reduction Factor (% inner vectors on RAM)

Results

Production-Level Implementation in RAxML-Light!



Results


No performance penalties for methods that re-traverse the entire tree, e.g., divergence time tools such as DDPDIV
1500
std
recomRANDOM
SLOT_MIN_COST
Average Running time (s)
14000
12000
10000
8000
6000
4000
2000
0
0
0.2
0.4
0.6
0.8
1
Reduction Factor (% inner vectors on RAM)

Outline

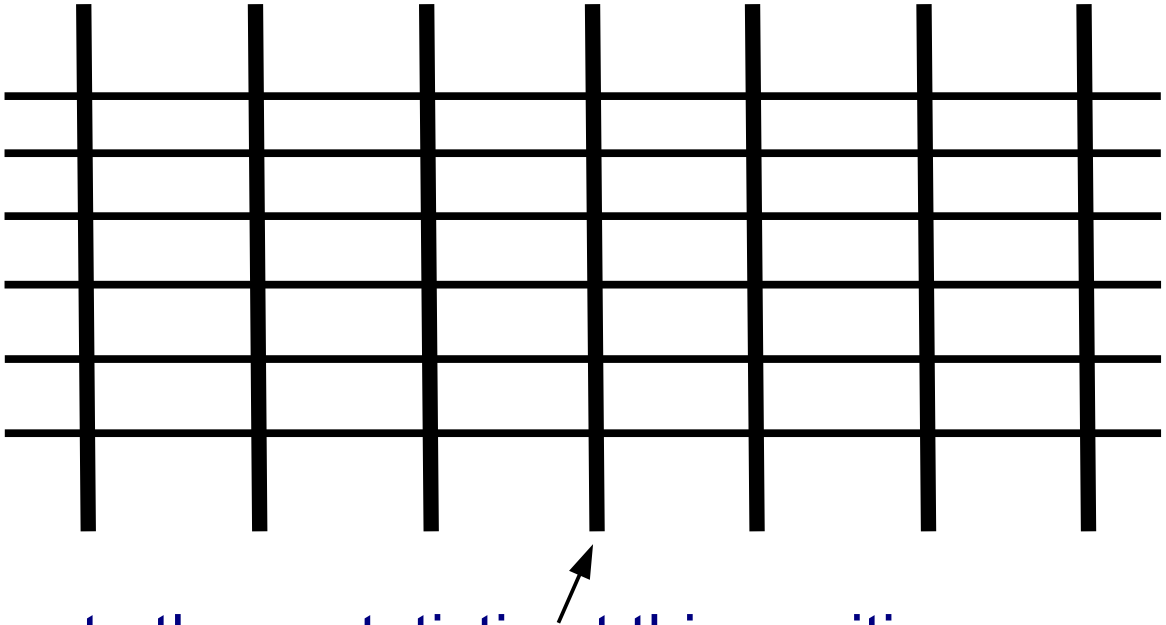
- Computing the Likelihood
- Optimizing & Parallelizing Likelihood Computations
- Saving Memory in Likelihood Computations
- HPC population genetics

HPC Pop. Gen.

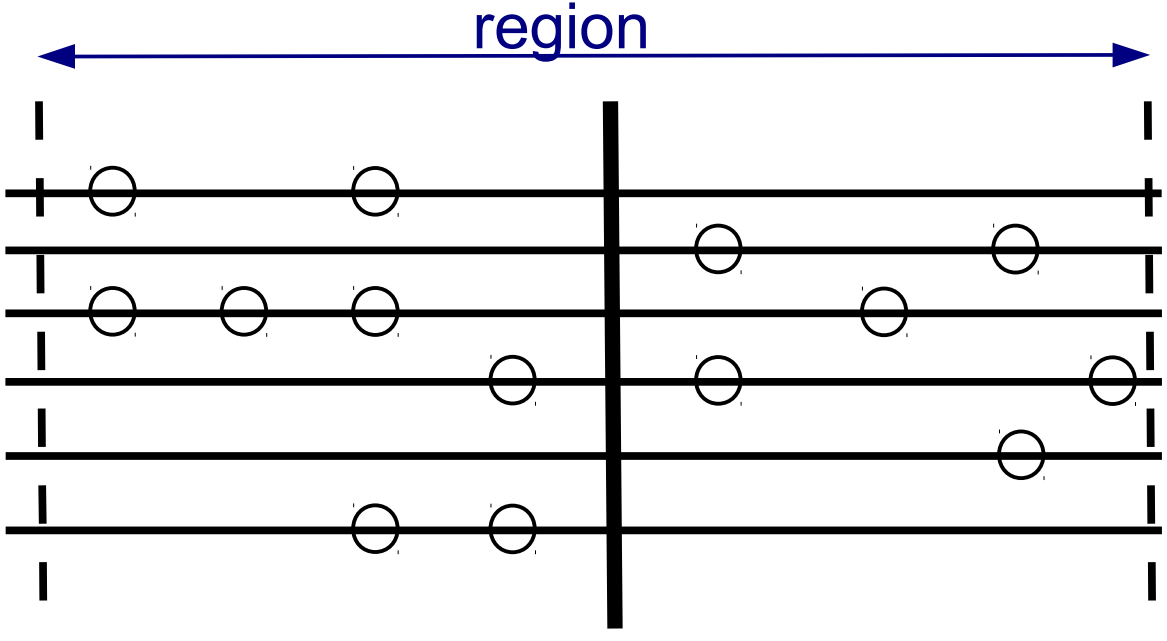
- Omega statistics code
  - Dynamic programming algorithm
  - Up to two orders of magnitude:
    - faster than existing codes
    - less memory
  - Multi-grain parallelization
- Forward in time simulator
  - Algorithmic engineering
  - New data structures
  - Vector instructions
  - 1-2 orders of magnitude faster than SFS
  - Work in progress: Parallelization

OmegaPlus: Selective Sweep Detection using the ω Statistic

OmegaPlus computes the ω statistic at N equidistant positions in the alignment.

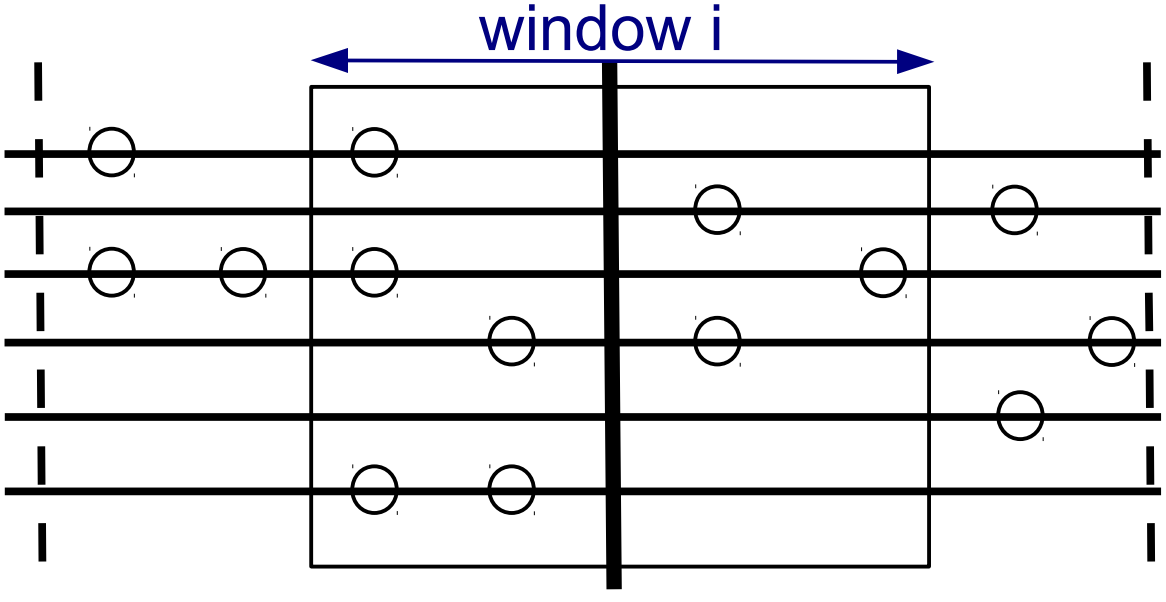


OmegaPlus



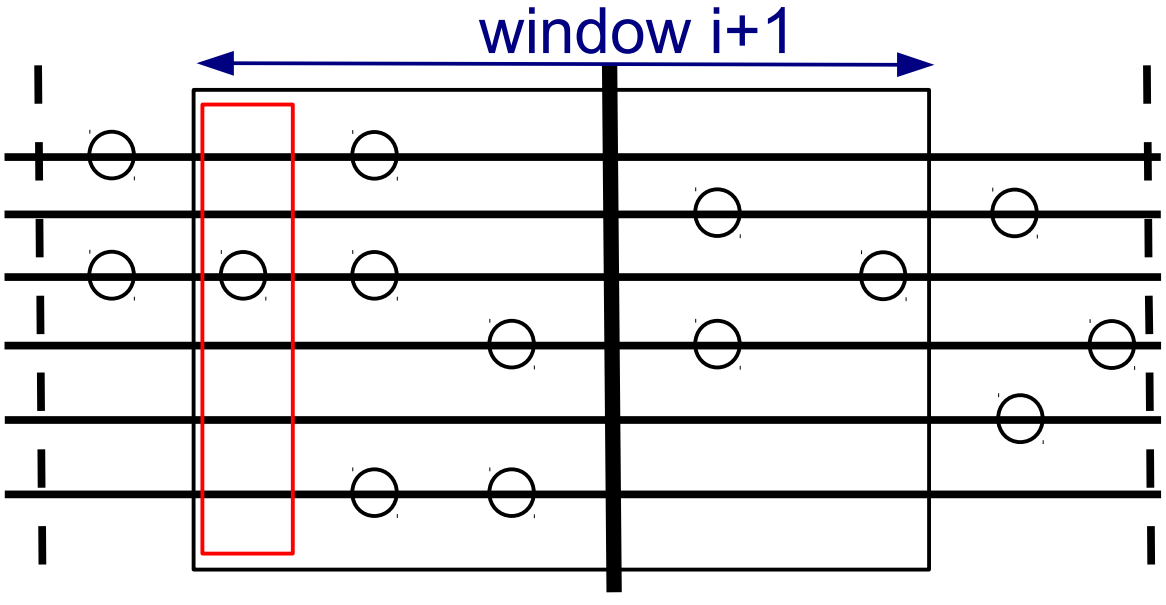
For each position, a maximum user-defined region that the sweep may have affected is analyzed.

OmegaPlus

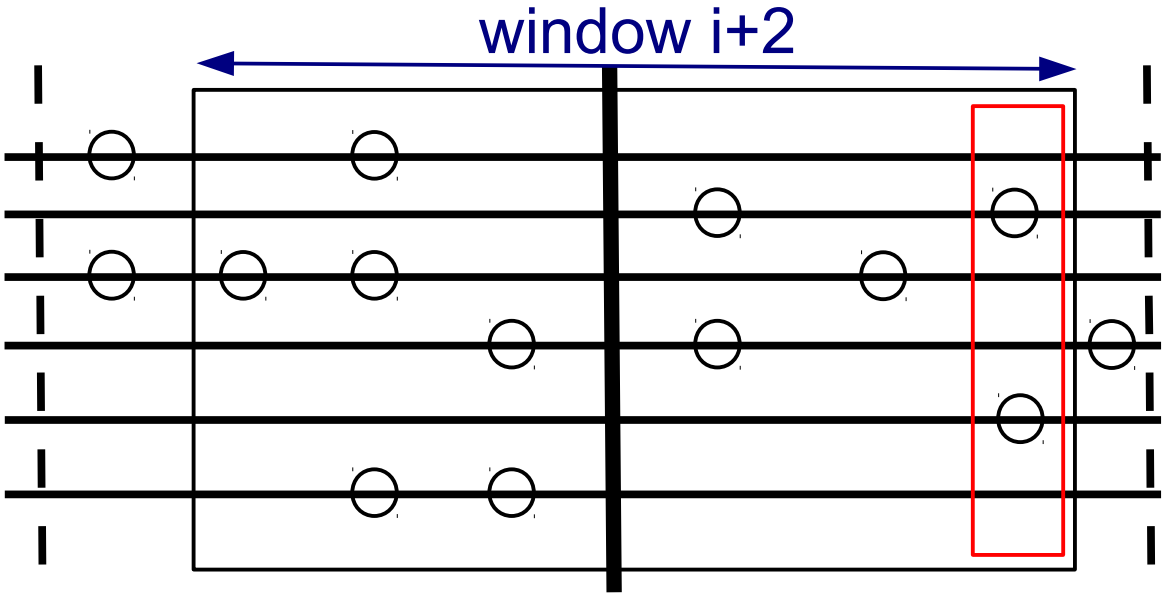


For each position, OmegaPlus computes ω for all possible window sizes within the user-defined region.
Each window i+1 contains one SNP more than the previous window i.

OmegaPlus



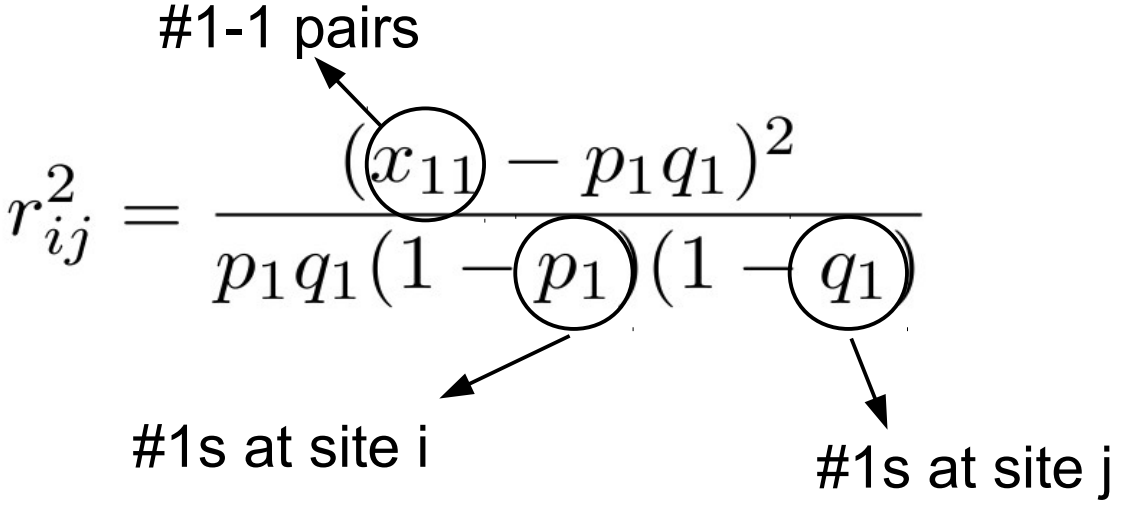
OmegaPlus



OmegaPlus

The ω statistic uses linkage-disequilibrium (LD) information to detect non-random associations of alleles at different alignment positions.

The LD at each site can be represented as a binary vector to calculate the squared correlation coefficient r_{ij}^2 between two sites i and j.

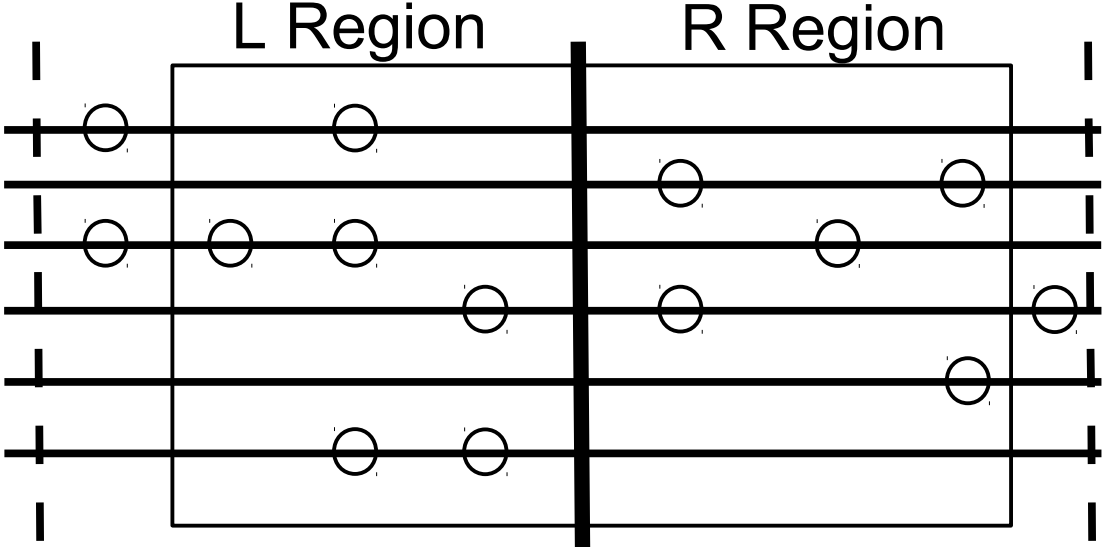


OmegaPlus

Number of SNPs in L

Number of SNPs in R

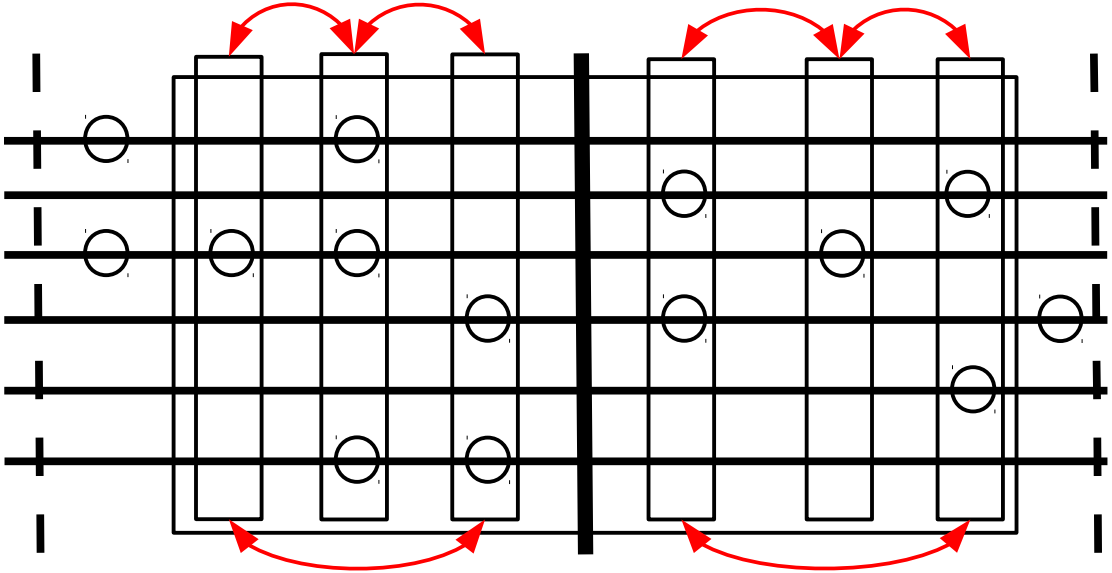
$$\omega = \frac{\left(\binom{l}{2} + \binom{S-l}{2}\right)^{-1}\left(\sum_{i,j \in L} r_{ij}^2 + \sum_{i,j \in R} r_{ij}^2\right)}{\left(l(S-l)\right)^{-1} \sum_{i \in L, j \in R} r_{ij}^2}$$



OmegaPlus

Selective sweep theory predicts LD increase at both sides of a selective sweep …

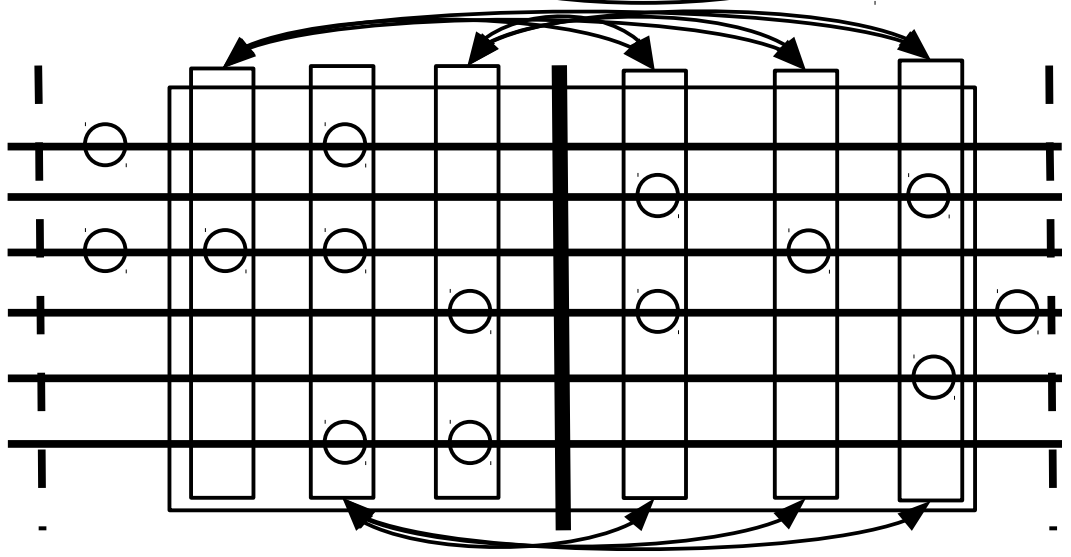
$$\omega = \frac{\left(\binom{l}{2} + \binom{S-l}{2}\right)^{-1}\left(\sum_{i,j\in L} r_{ij}^2 + \sum_{i,j\in R} r_{ij}^2\right)}{(l(S-l))^{-1}\sum_{i\in L, j\in R} r_{ij}^2}$$



OmegaPlus

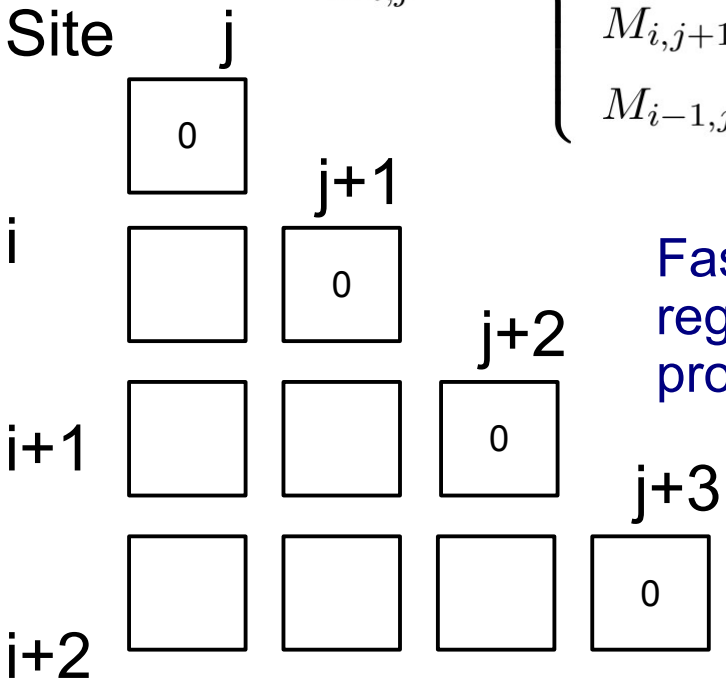
… but not across the selected site.

$$\omega = \frac{\left(\binom{l}{2} + \binom{S-l}{2}\right)^{-1}\left(\sum_{i,j\in L} r_{ij}^2 + \sum_{i,j\in R} r_{ij}^2\right)}{(l(S-l))^{-1}\sum_{i\in L, j\in R} r_{ij}^2}$$



OmegaPlus

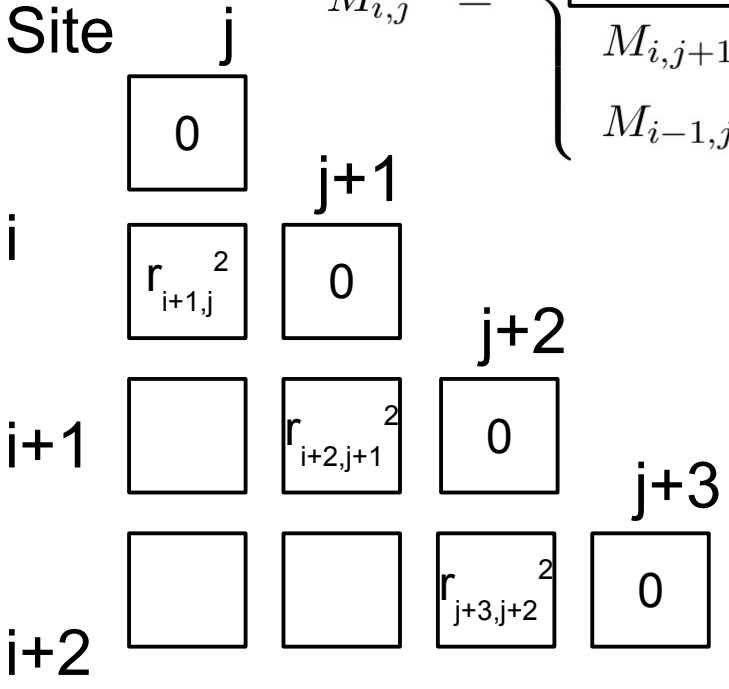
$$M_{i,j} = \begin{cases} 0 & 1 \leq i \leq S,\ j = i \\ r_{ij}^2 & 2 \leq i \leq S,\ j = i-1 \\ M_{i,j+1} + M_{i-1,j} + & \\ M_{i-1,j+1} + r_{ij} & 3 \leq i < S,\ i-1 > j \geq 0 \end{cases}$$



Fast computation of sums for a region using dynamic programming.

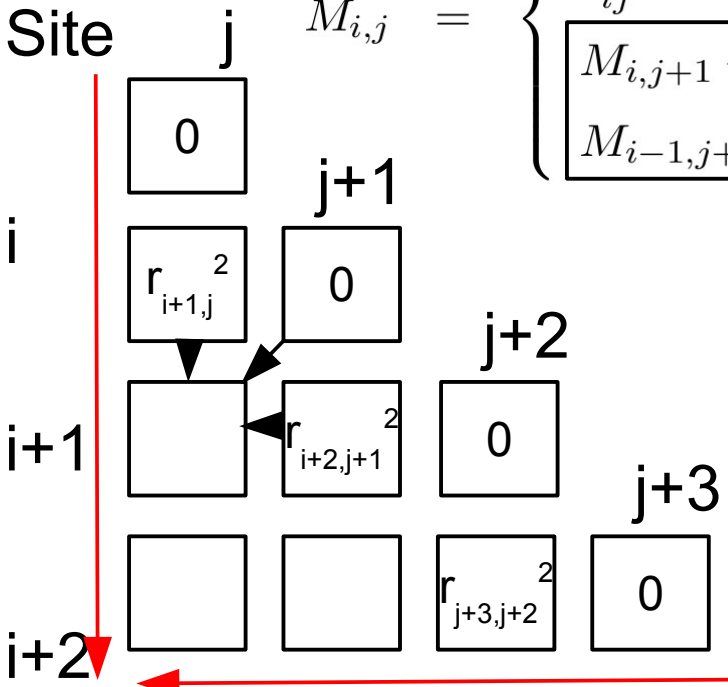
OmegaPlus

$$M_{i,j} = \begin{cases} 0 & 1 \leq i \leq S,\ j = i \\ r_{ij}^2 & 2 \leq i \leq S,\ j = i-1 \\ M_{i,j+1} + M_{i-1,j} + & \\ M_{i-1,j+1} + r_{ij} & 3 \leq i < S,\ i-1 > j \geq 0 \end{cases}$$



OmegaPlus

$$M_{i,j} = \begin{cases} 0 & 1 \leq i \leq S,\ j = i \\ r_{ij}^2 & 2 \leq i \leq S,\ j = i-1 \\ M_{i,j+1} + M_{i-1,j} + M_{i-1,j+1} + r_{ij} & 3 \leq i < S,\ i-1 > j \geq 0 \end{cases}$$

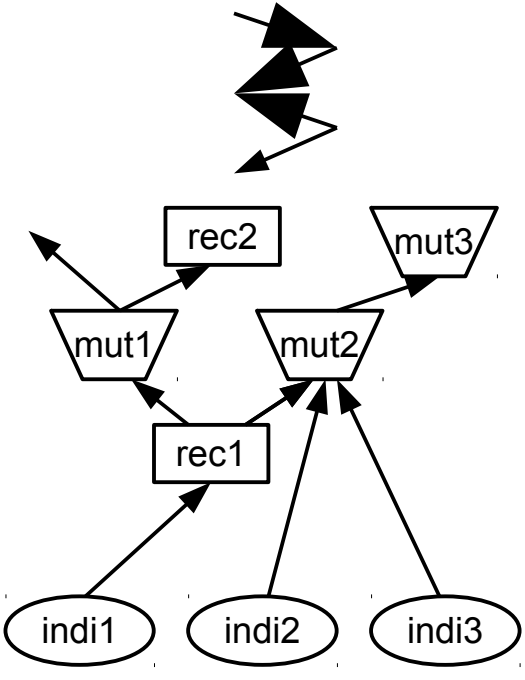


Forward-in-Time Simulation (FiTS)

- Most accurate method for simulating pop. Gen. datasets
- Works for small population sizes
- Straight-forward method to simulate evolution of multiple non-neutral mutations
- Combines various evolutionary forces under complex demographic scenarios
- **Problem:** orders of magnitude slower than backward simulation

Toward Whole-Genome FiTS

```
0:   n: 000001001000 s:   fitness: 1.01748
1:   n: 000001001101 s:   fitness: 1.02114
2:   n: 000011001010 s:   fitness: 1.02083
3:   n: 000001001010 s:   fitness: 1.02286
4:   n: 000001001010 s:   fitness: 1.02286
```



- Simulate non-neutral mutations forward in time
- Keep track of ancestry
- Progressively create ancestral graph
- Add neutral mutations in the end “backward-like”

Towards Whole-Genome FiTS

- Advantage: yields complete history for all simulated sequences in the population
- Uses:
  - high-quality random number generators
  - efficient data structures
  - SIMD/SSE3 instructions
- At present: 1-2 orders of magnitude faster than SFS code
- Planned:
  - parallelization on shared-memory machines
  - trait-based fitness function

Some other projects

- Protein model assignment in partitioned datasets with joint branch length estimates
  - Is this NP-hard?
  - Can we design good heuristics?
- RAxML goes Bayesian
- Phylogenetic short read identification and alignment in metagenomics
- Discrete algorithms on tree sets → building consensus trees and identifying rogue taxa
- Species delimitation in metagenomic samples
- Novel protein substitution models

Tonight

Thank you for your Attention !

Psiloritis peak, Crete, Greece, April 28, 2012