

Introduction to Bioinformatics for Computer Scientists

Lecture 11

Course Beers

- Tonight, Oxford pub, reserved on the name of Paul
- Remember, drinks are on me

Plan for next lectures

- Today (Alexis):
 - More on Models
 - The Newton-Raphson Method
 - Data Structures for unrooted Phylogenetic Trees
 - Implementing and Optimizing Likelihood Calculations
 - Parallel Likelihood Calculations
- Lecture 11 (Alexis) Bayesian Statistics & MCMC methods

Basic Operations

Maximum Likelihood

- Compute Conditional Likelihood Vector 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 Vector 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

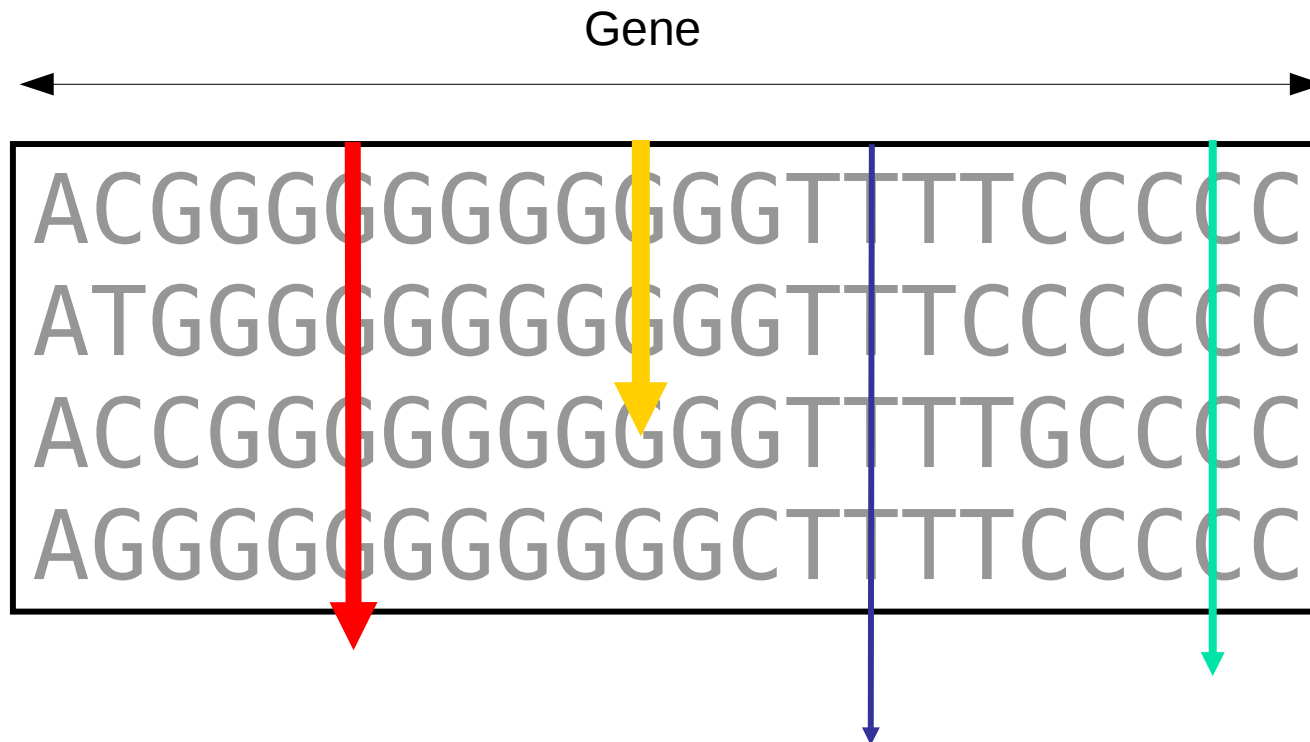


The optimizers are the tricky routines!

Protein Substitution Models

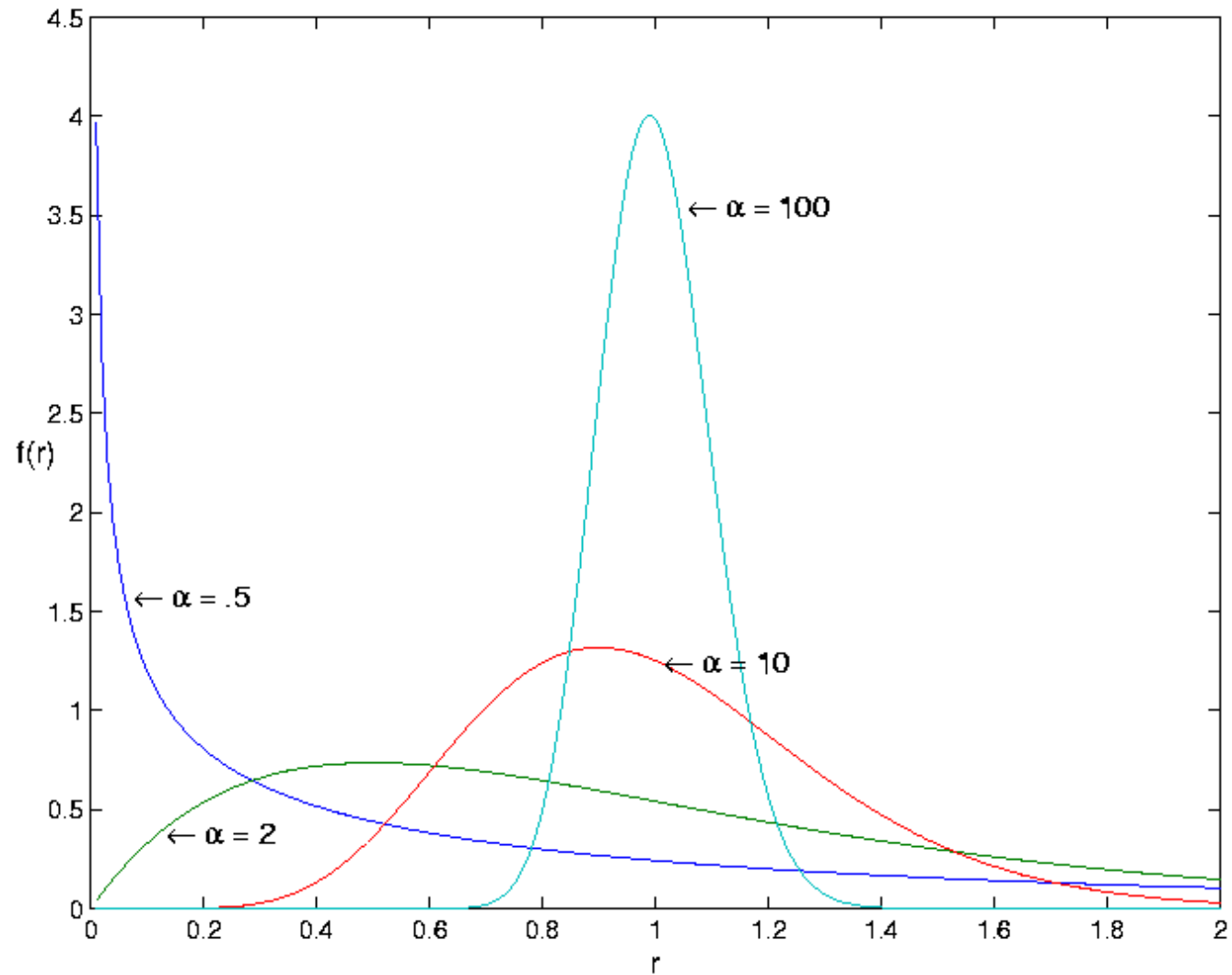
- The GTR Q matrix for protein data has 189 free parameters instead of just 5 (DNA)
- Estimating **189** rate parameters is difficult, time-consuming, and may lead to over-parameterizing the model
- Instead, empirical models such as JTT, LG, WAG, MTMAM, etc. are used
- The Q matrices are obtained by jointly optimizing model parameters on a large collection of reference alignments
- The models differ with respect to:
 - the amount of data used to obtain them
 - the type of data on which the models have been optimized
 - e.g., dedicated models for HIV, FLU, Mammals
 - the numerical optimization methods used
- Examples of general models:
 - **LG**: Le & Gascuel: “An Improved General Amino Acid Replacement Matrix”
 - **WAG**: Whelan & Goldman: “A General Empirical Model of Protein Evolution Derived from Multiple Protein Families Using a Maximum-Likelihood Approach”

Rate Heterogeneity among Sites



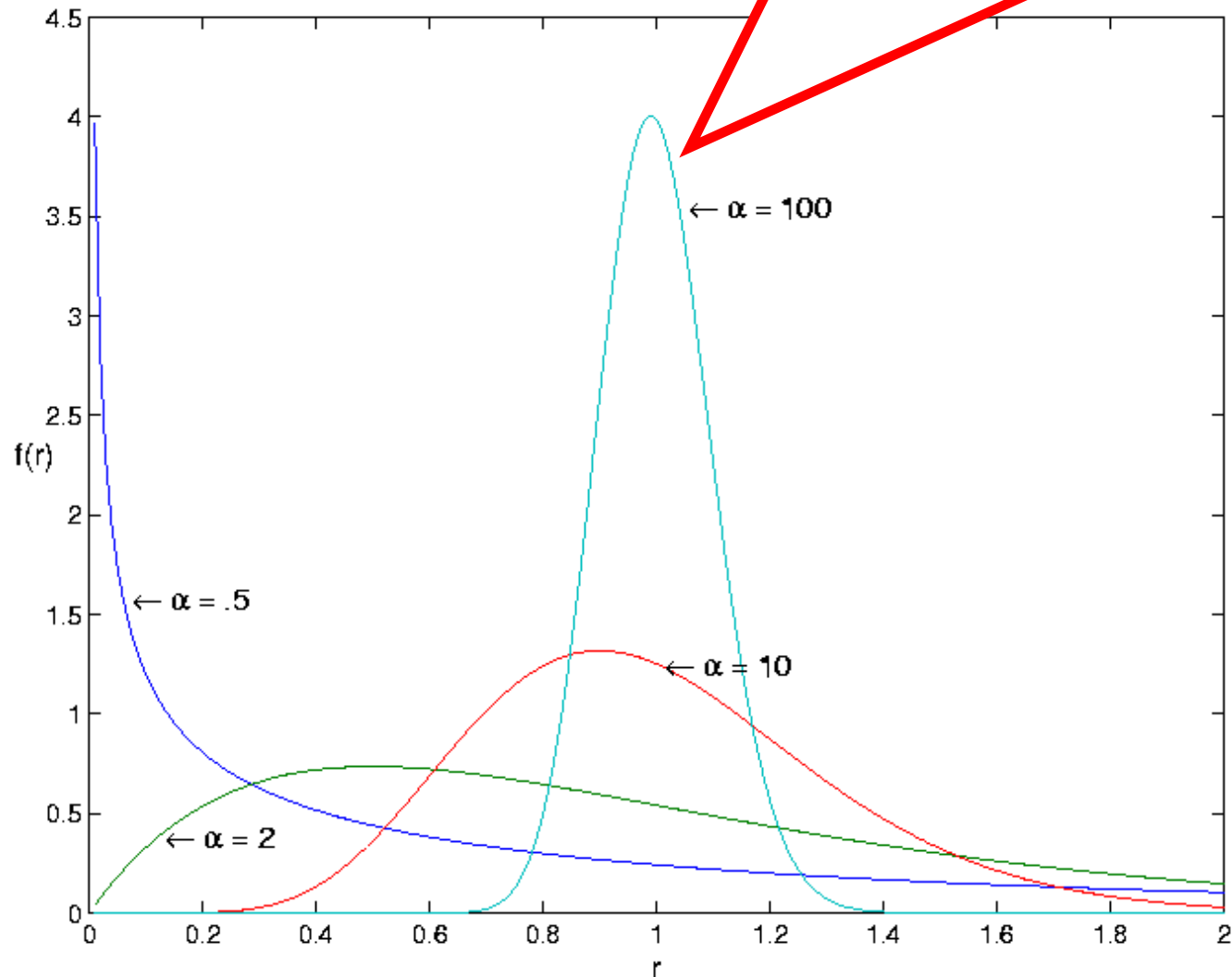
- Among-site rate heterogeneity
 - Biological phenomenon
 - different sites/columns evolve at different speeds
 - Need to accommodate this in our models

Γ -Distribution

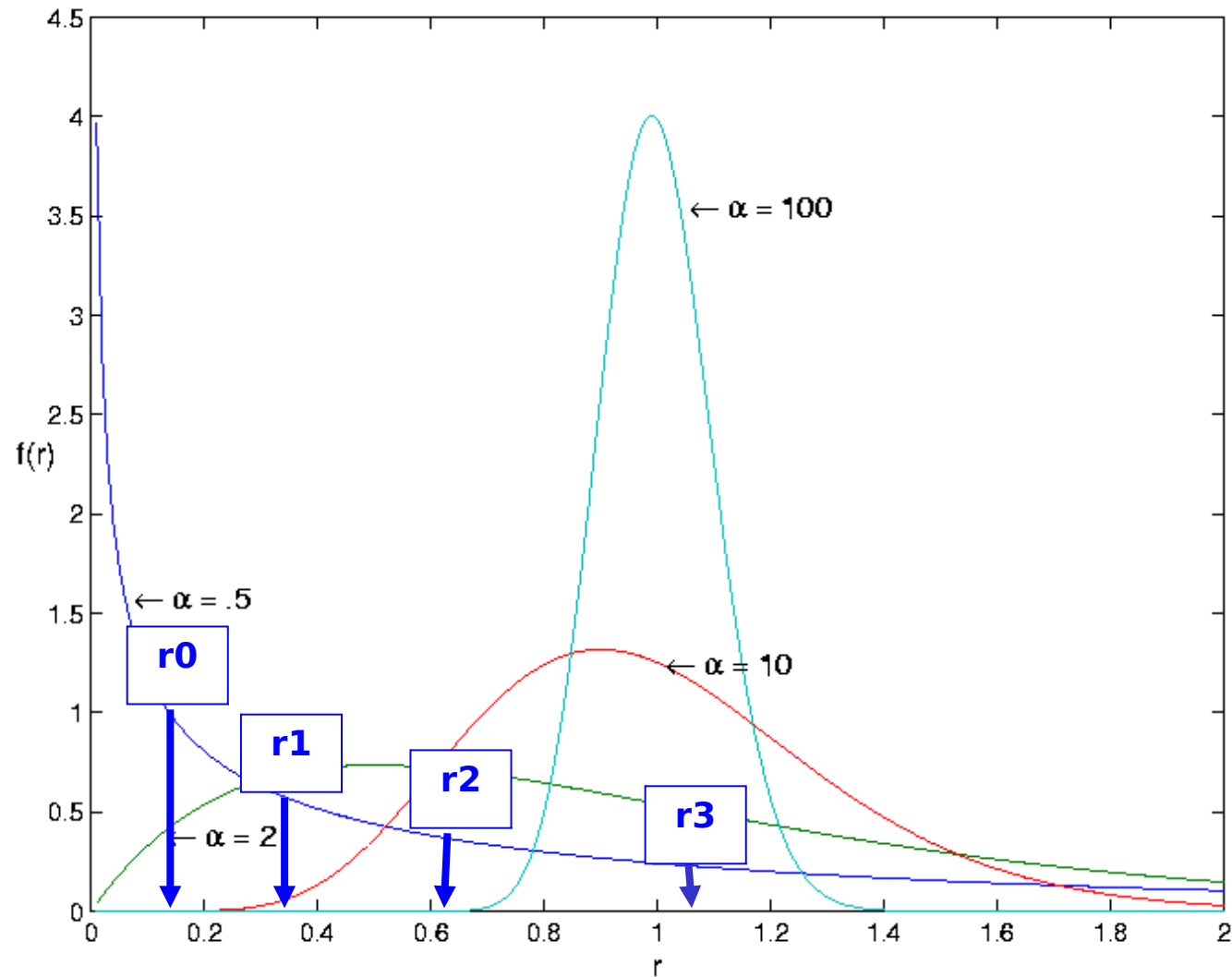


Γ -Distribution

Small $\alpha \rightarrow$ high rate heterogeneity
Large $\alpha \rightarrow$ low rate heterogeneity



Discrete Γ -Distribution



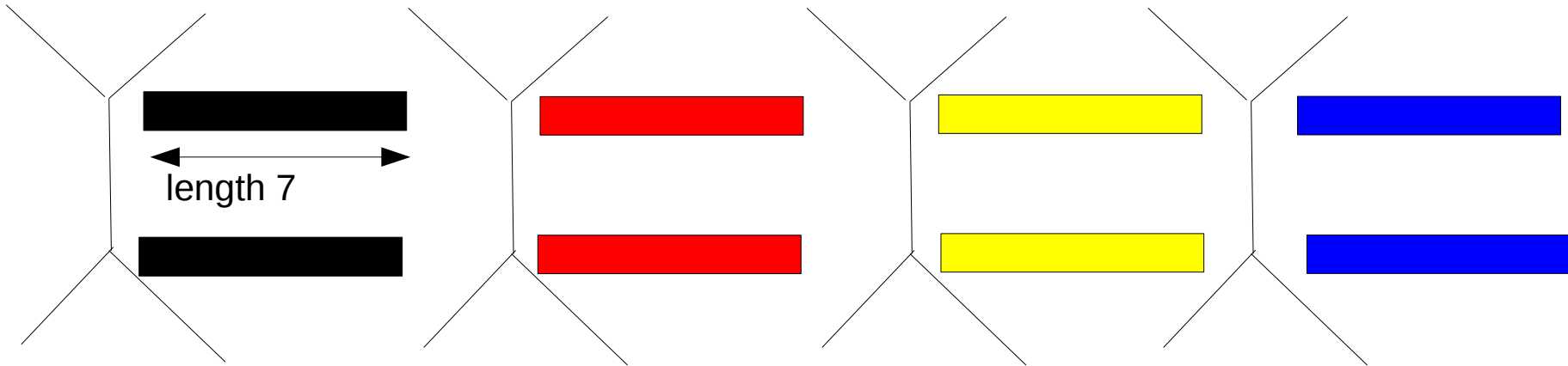
An Abstract View of Γ

rate 0
 $P(t) = e^{Qr_0 t}$

rate 1
 $P(t) = e^{Qr_1 t}$

rate 2
 $P(t) = e^{Qr_2 t}$

rate 3
 $P(t) = e^{Qr_3 t}$



This is the integral of the likelihood we approximate via discretization

$$\text{Ln}L(i) = \log\left(\frac{1}{4} * (L_0 + L_1 + L_2 + L_3)\right)$$

Log likelihood
 at site i
 11

All Γ rates have equal probability

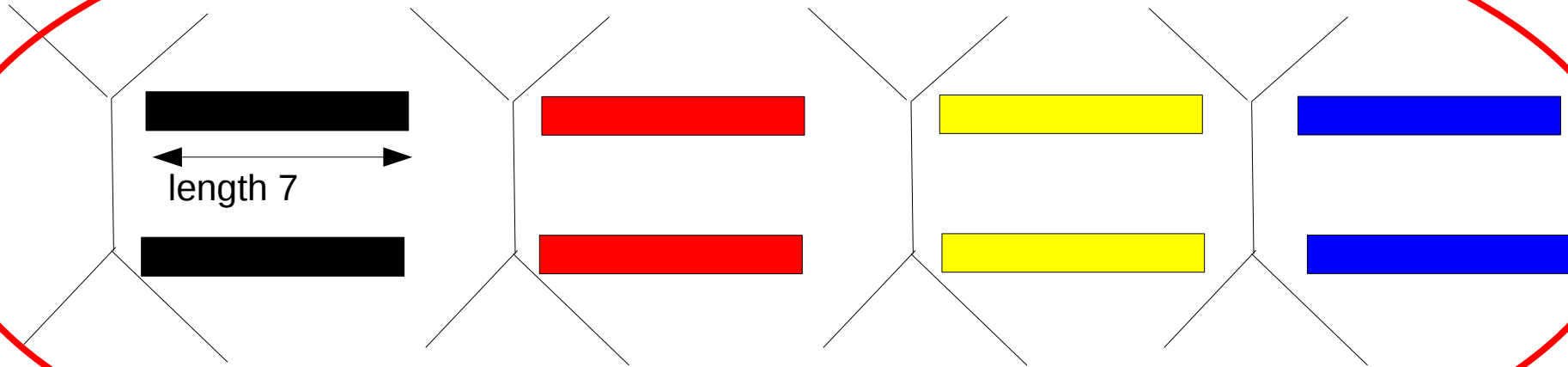
An Abstract View of Γ

rate 0
 $P(t) = e^{Qr_0 t}$

rate 1
 $P(t) = e^{Qr_1 t}$

rate 2
 $P(t) = e^{Qr_2 t}$

rate 3
 $P(t) = e^{Qr_3 t}$



4 times higher memory consumption

An Abstract View of Γ

rate 0

$$P(t) = e^{Qr_0 t}$$

rate 1

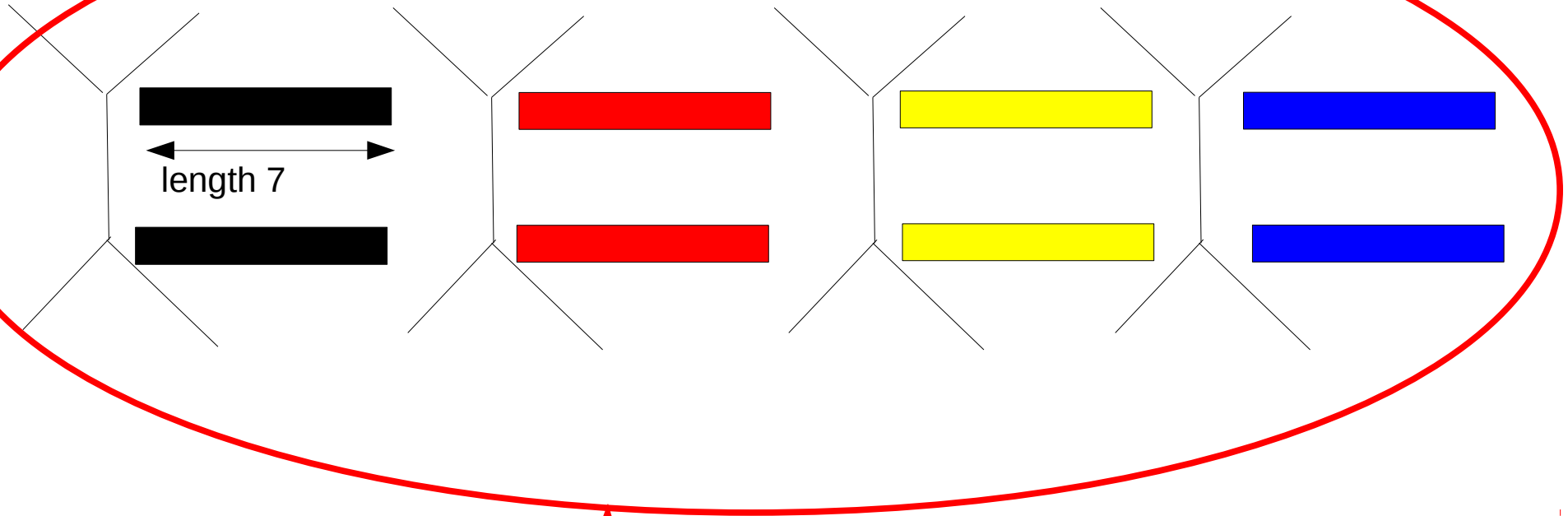
$$P(t) = e^{Qr_1 t}$$

rate 2

$$P(t) = e^{Qr_2 t}$$

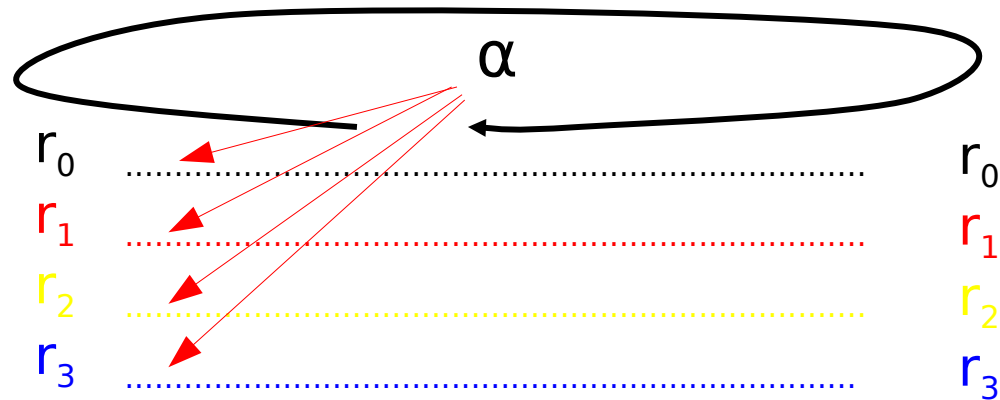
rate 3

$$P(t) = e^{Qr_3 t}$$



4 times more floating point operations

Γ Model of Rate Heterogeneity with 4 discrete rates



| | | | | |
|------------|------------|------------|------------|------------|
| Species209 | NNNNNNNNNN | NNNNNNNNNN | NNNNNNNNNN | NNNNNNNNNN |
| Species159 | UAUCUGGUUG | AUCCUGCCAG | UAGUAUNUGC | UUGUCUCAAA |
| Species109 | NNNNNNNNNN | NNNNNNNNNN | NNNNAUAUGC | UUGUCUC--A |
| Species025 | NNNNNNNNNN | NNNNNNNNNN | NNNNNNNNNN | NNNNNNNAAA |
| Species004 | NNNNNNNNNN | NNNNNNNNNN | NNNNNNNNNN | NNNNNNNNNN |
| Species186 | UACCUGGUUG | AUCCUGCCAG | UAGUAUAUGC | UUGUCUCAAA |
| Species034 | NNNNNNNNNN | NNNNNNNNNN | NNNNNNNNNN | NNNNNNNAAA |
| Species003 | NNNNNNNNNN | NNNNNNNNNN | NNNNNNNNNN | NNNNNNNAAA |
| Species192 | NNNNNNNNNN | NNNNNNNNNN | NNNNNNNNNN | NNNNNNCAAA |
| Species132 | NNNNNNNNNN | NNNNNNNNNN | UAGUAUAUGC | UUGUCUCAAA |
| Species172 | NNNNNNNNNN | NNNGAAUU-G | UAUUAANGC | -UGUUUCAAA |

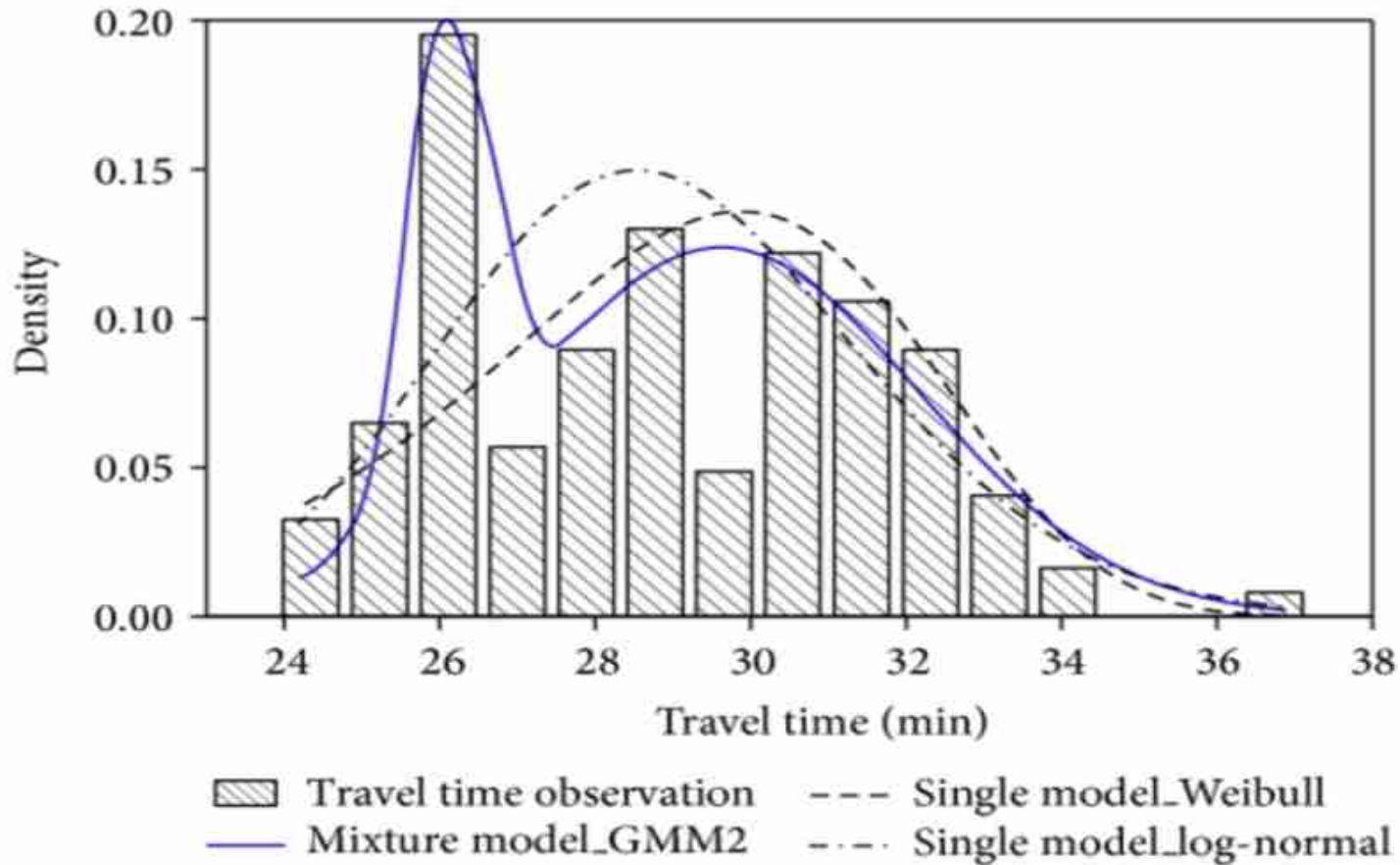
Mixture Models

- The of rate heterogeneity is a simple example of so-called mixture models
- From Wikipedia: “In statistics, a mixture model is a probabilistic model for representing the presence of subpopulations within an overall population, without requiring that an observed data set should identify the sub-population to which an individual observation belongs. Formally a mixture model corresponds to the mixture distribution that represents the probability distribution of observations in the overall population.”
- The Γ model gives us 4 discrete evolutionary rates over which we integrate (add) the likelihood for each site, without assigning a specific rate to a specific site

Mixture Models

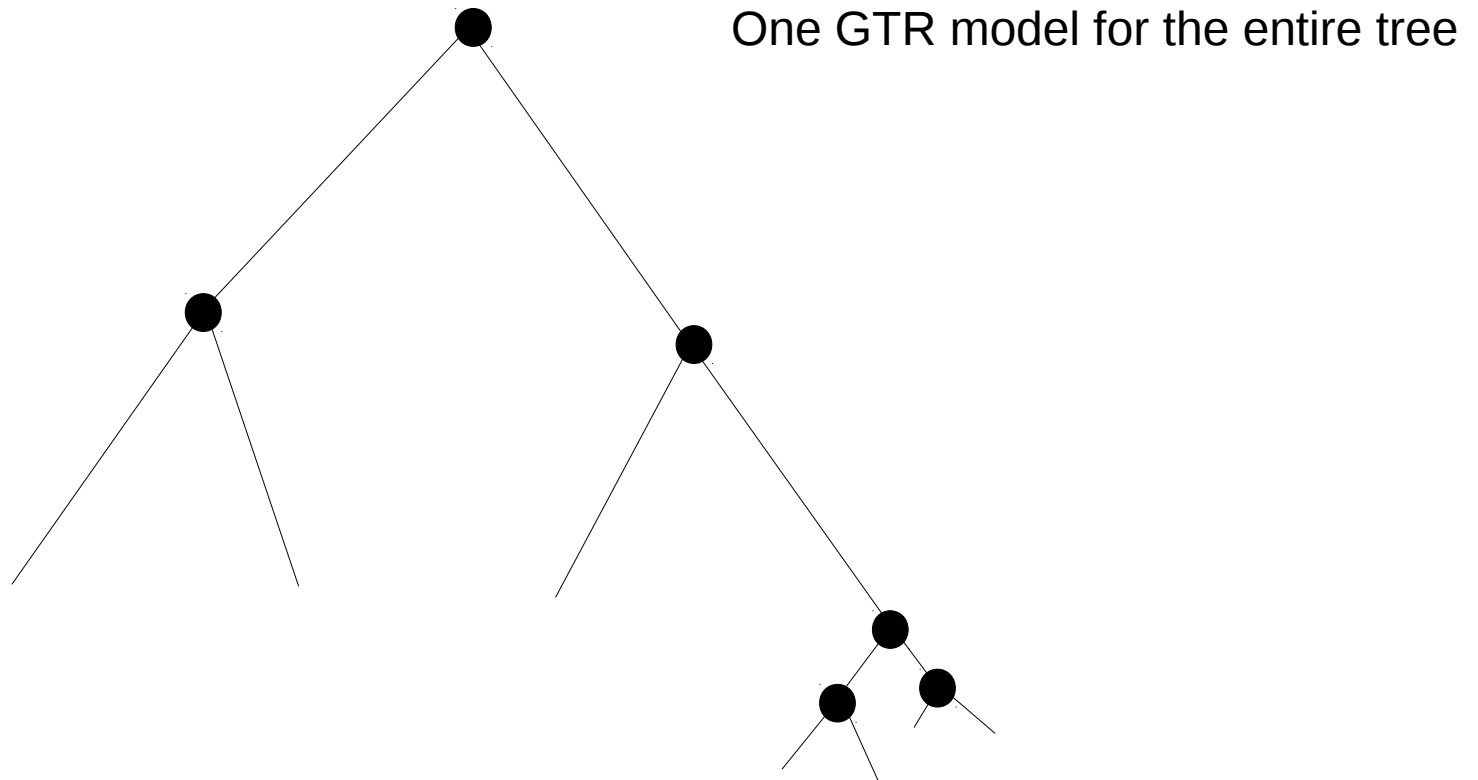
- We can also imagine to integrate the likelihood over a set of
 - distinct Q matrices
 - distinct base frequencies
 - or combinations thereof
- The LG protein substitution model is an example thereof:
- It uses 4 distinct empirical Q matrices and 4 distinct sets of base frequencies π over which we integrate just like for the Γ model

An example



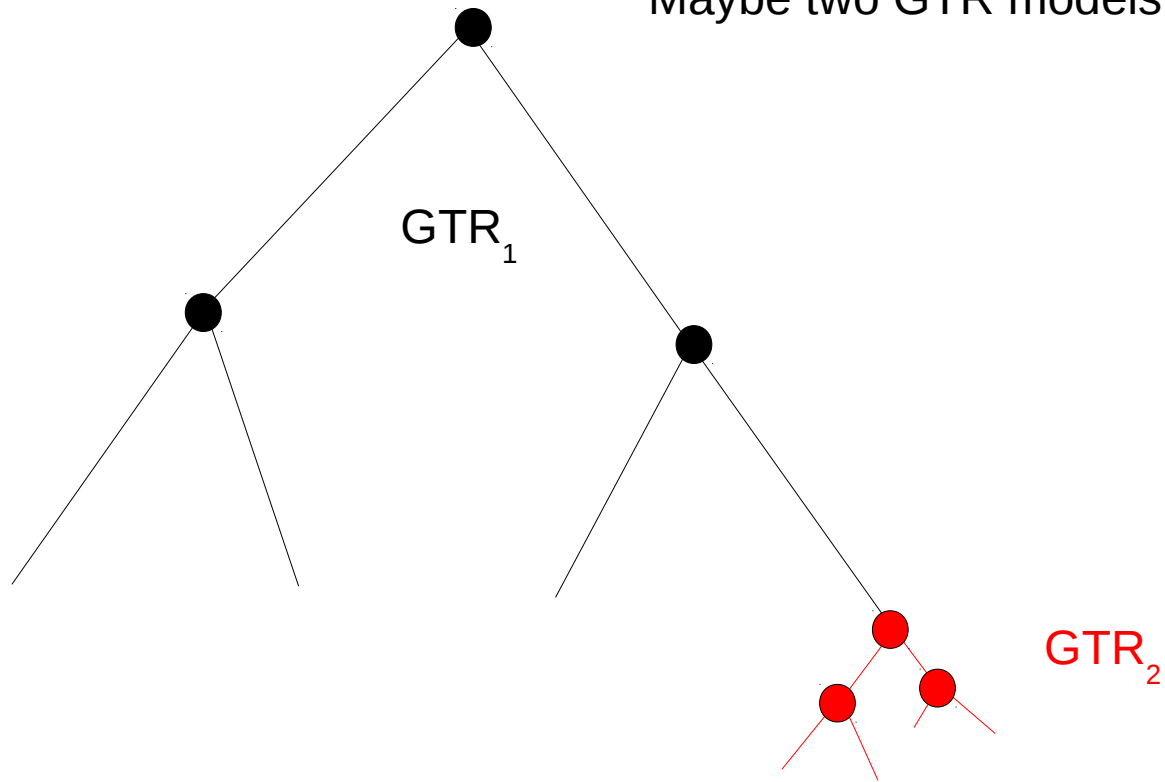
Taken from: "Measuring Service Reliability Using Automatic Vehicle Location Data"
→ bus service reliability

Heterotachous Models

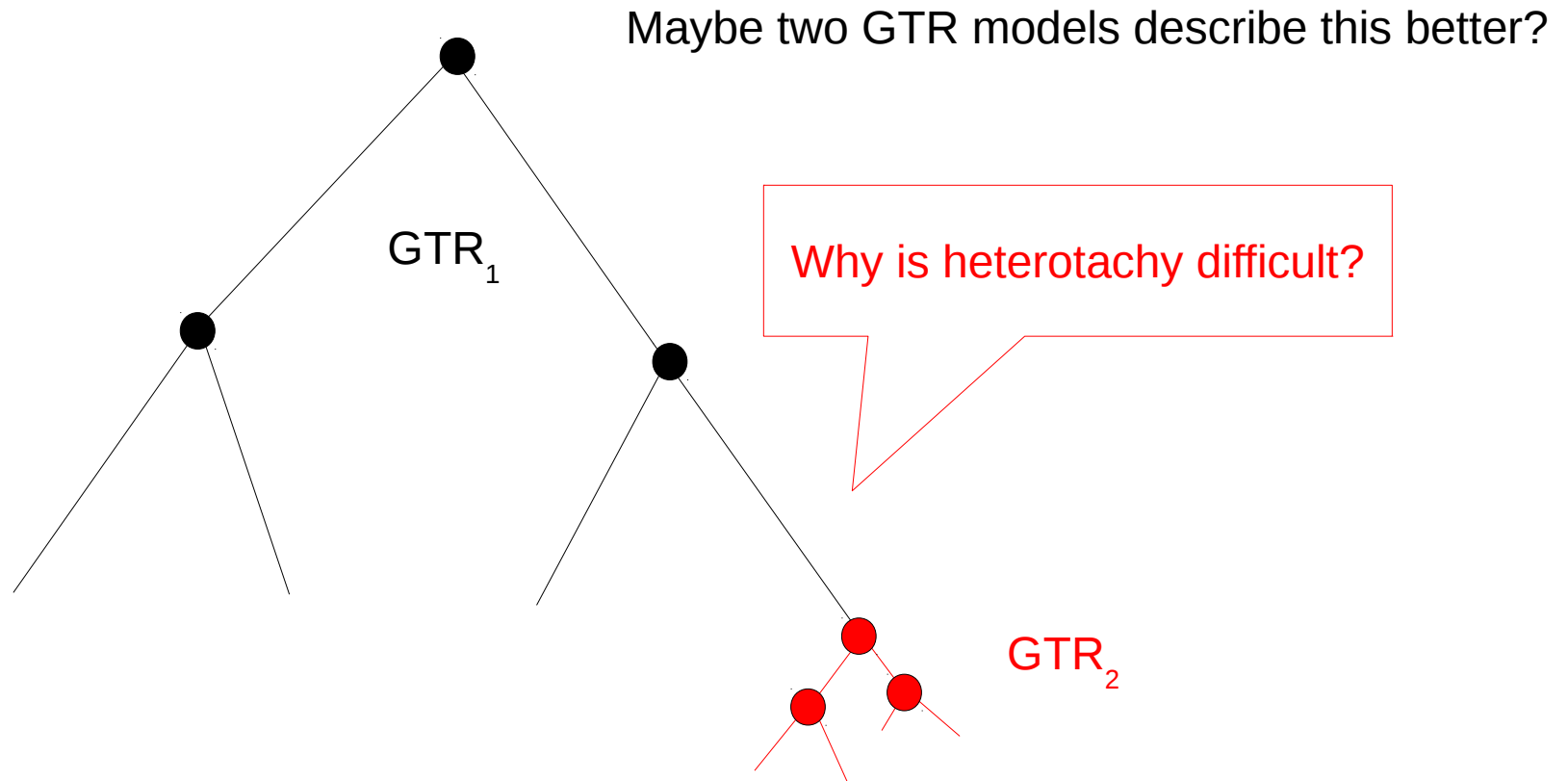


Heterotachous Models

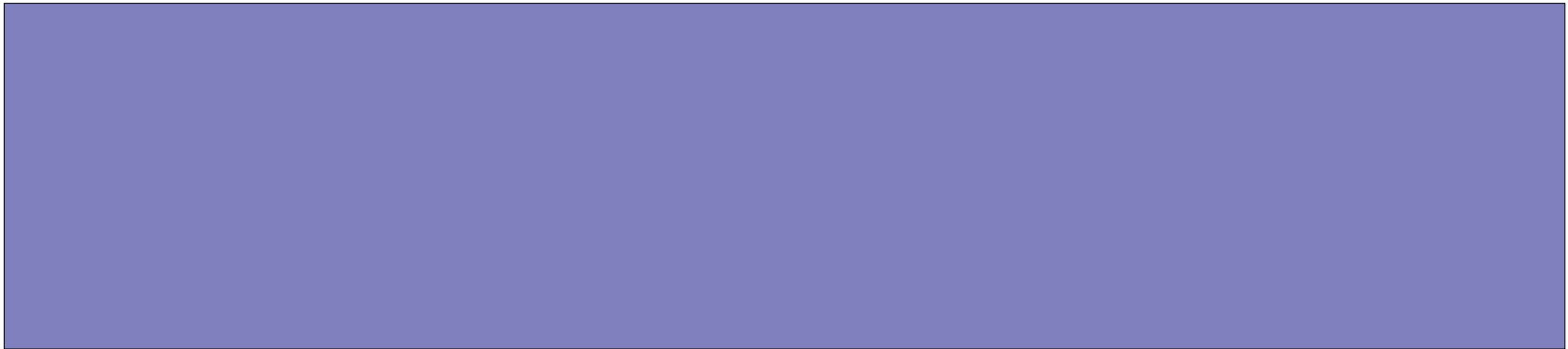
Maybe two GTR models describe this better?



Heterotachous Models

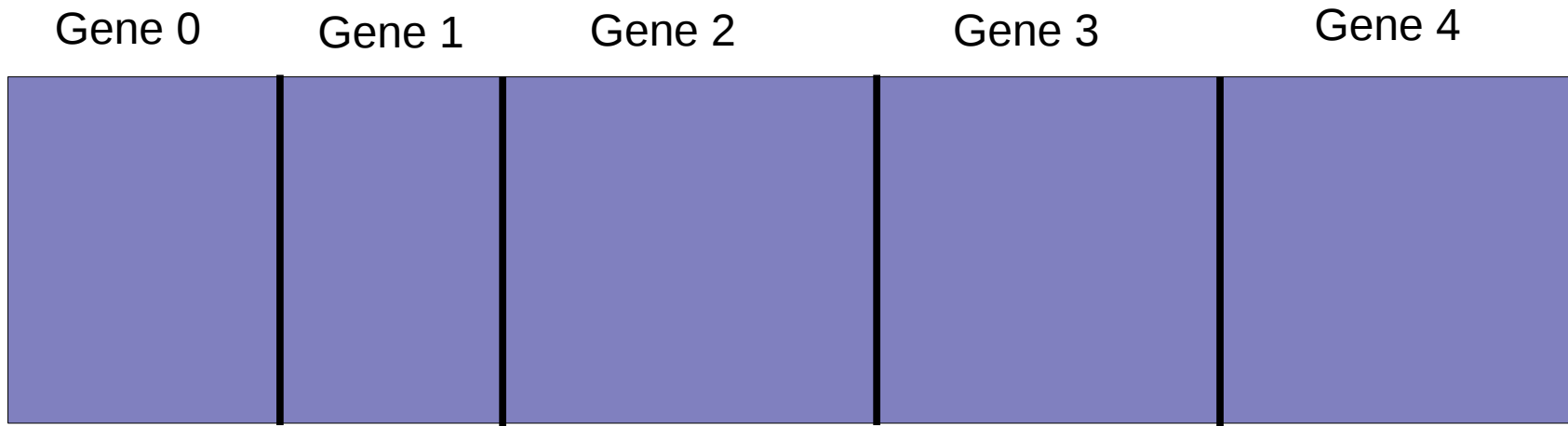


What is a partitioned dataset?



Multi-gene or whole-genome alignment

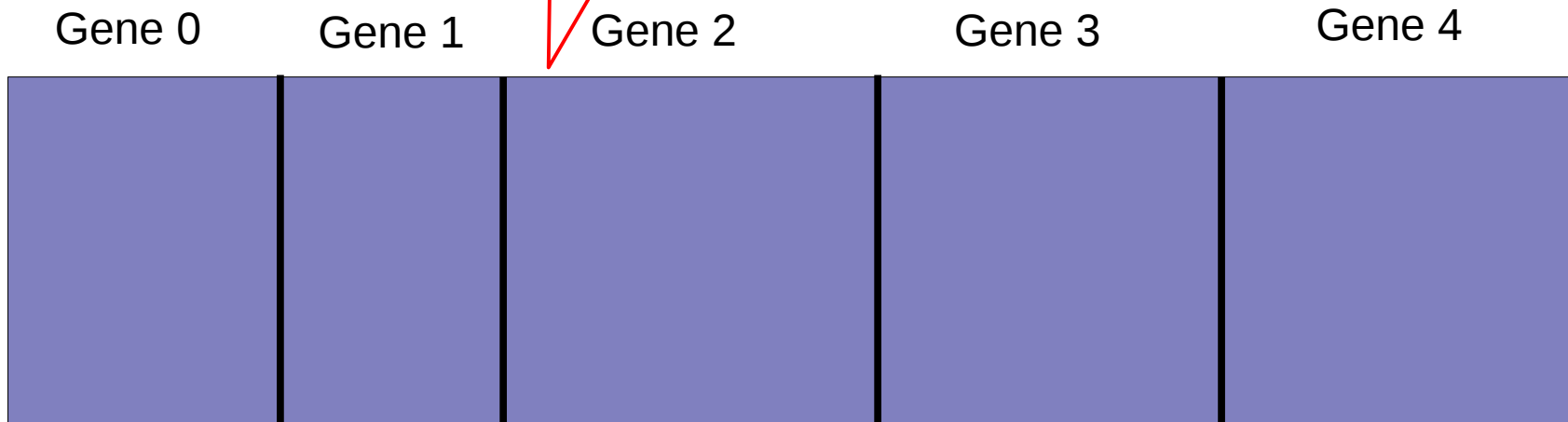
What is a partitioned dataset?



Multi-gene or whole-genome alignment

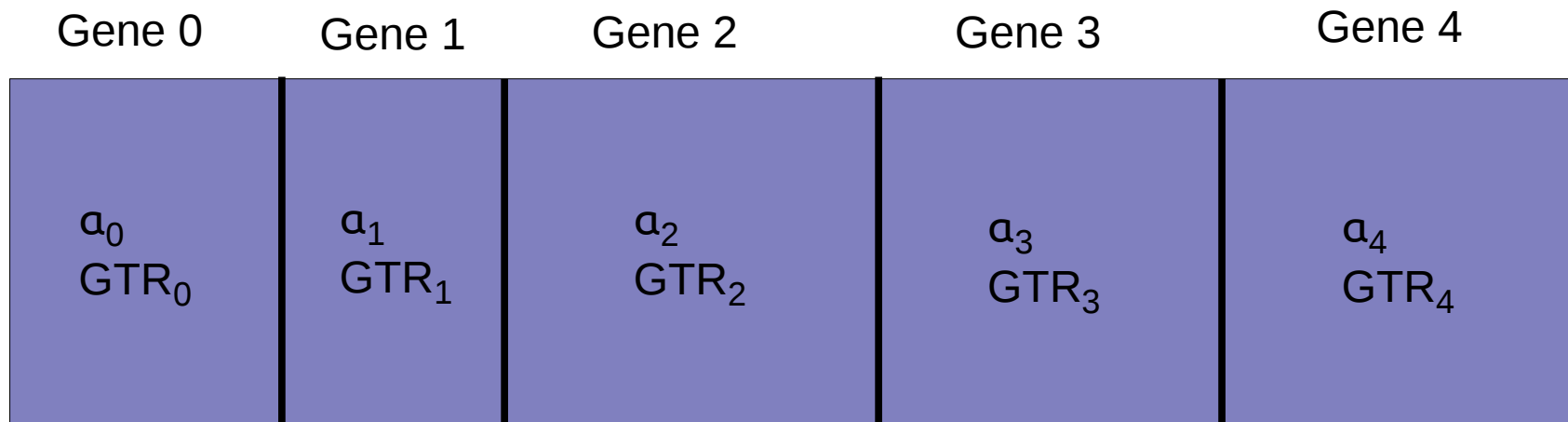
What is a partitioned dataset?

We may also partition
by 1st, 2nd, and 3rd
codon position

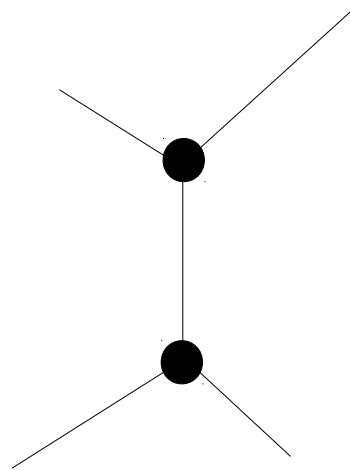
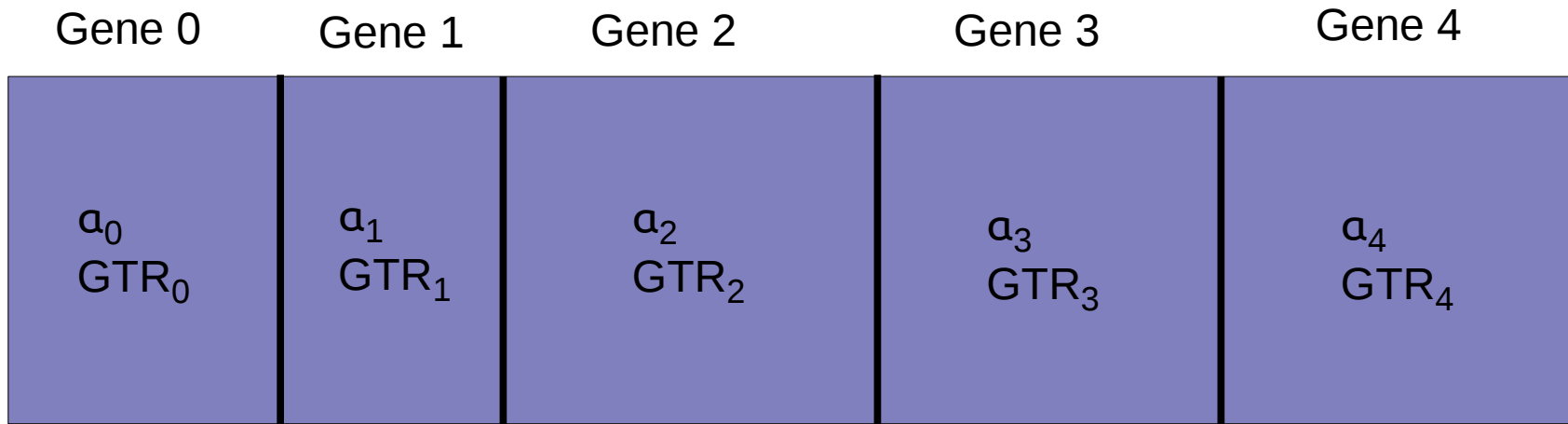


Multi-gene or whole-genome alignment

What is a partitioned dataset?

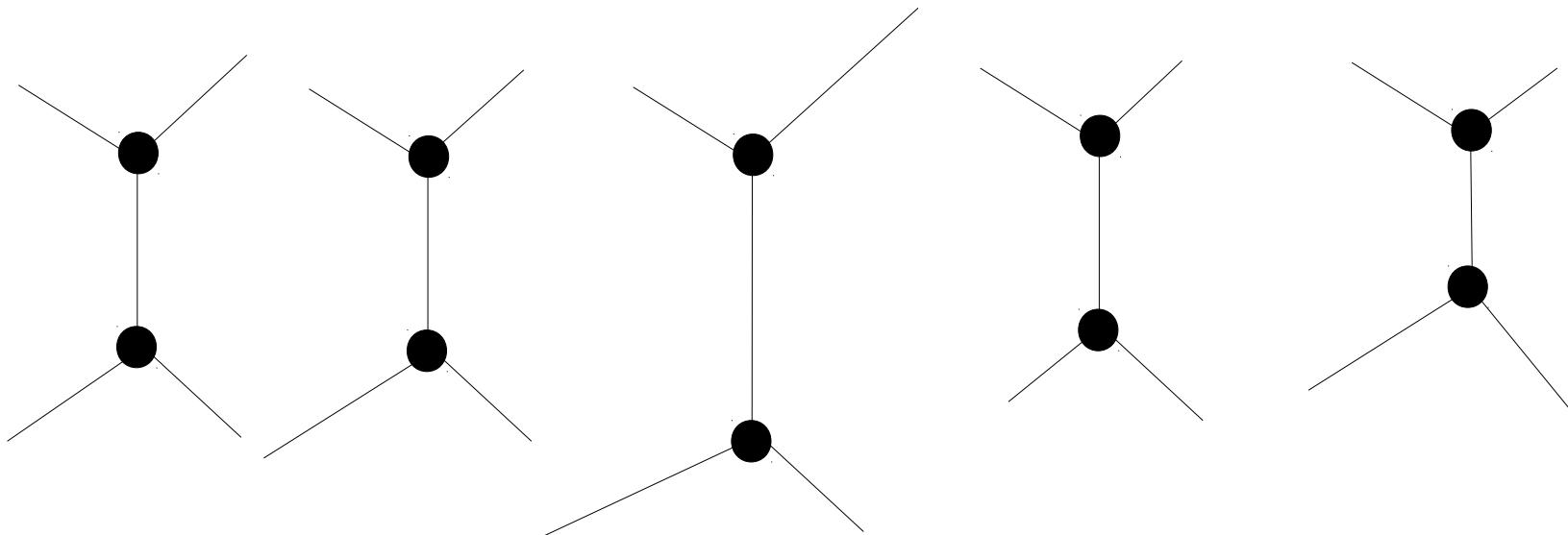
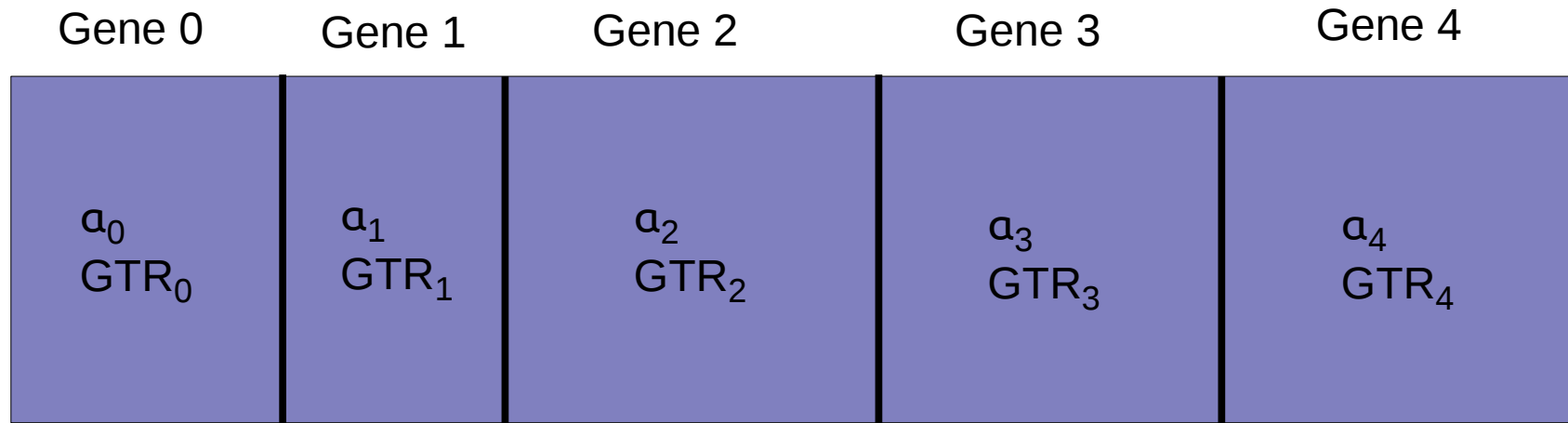


What is a partitioned dataset?

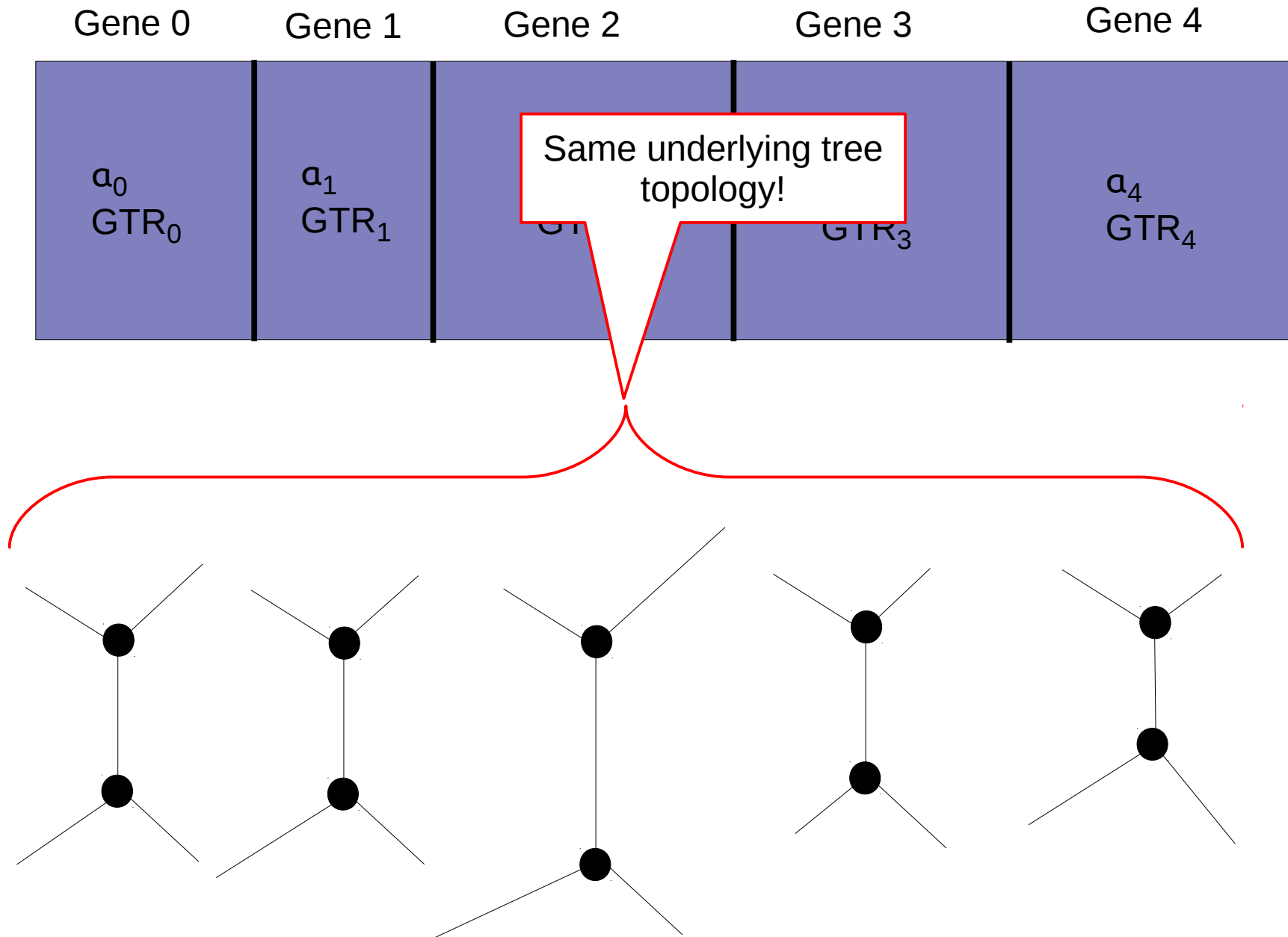


Joint branch length
estimate

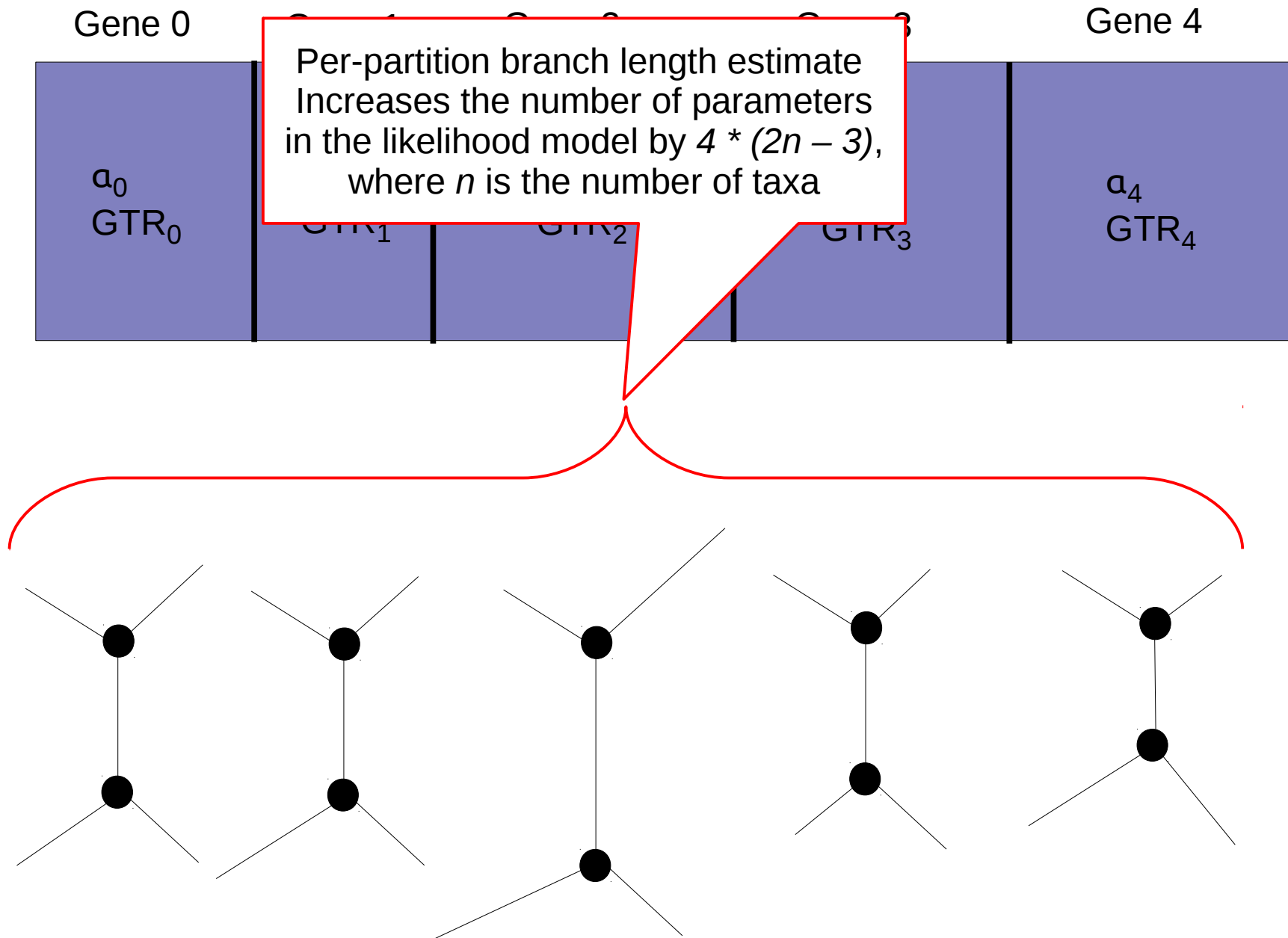
What is a partitioned dataset?



What is a partitioned dataset?



What is a partitioned dataset?



Models and Parameters

- If we add an additional parameter to a model, the likelihood will become better
- However, this does not mean anything, as
 - We might be over-parameterizing
 - The key question is if the more complex model yields a different tree topology
- So, how do we determine the best-fit model for a given dataset?

Model Testing

- If models are nested we can use a likelihood ratio test
- Model A is nested in model B if parameters in model A are a subset of the parameters in model B
- For instance: the *Jukes Cantor (JC)* model is nested in the *General Time Reversible (GTR)* model of nucleotide substitution
- $LR = P(\mathbf{D}|A) / P(\mathbf{D}|B) = L(A) / L(B)$
- $\Delta = \ln(LR^2) = 2 (\ln(L(A)) - \ln(L(B)))$
- Compare Δ to χ^2 distribution with $k_A - k_B$ degrees of freedom to determine if the Δ is significant or not
- The degrees of freedom difference is the difference in the number of free parameters in the models
- How many free parameters do the *JC* and *GTR* models have?

Model Testing

- If models are nested we can use a likelihood ratio test
- Model A is nested in model B if parameters in model A are a subset of the parameters in model B
- For instance, *General Time Reversible (GTR)* is a subset of *General* Time Reversible (GTR) model of nucleotide substitution
- We are only allowed to compare likelihoods on the same data **D**!
- $LR = P(\mathbf{D}|A) / P(\mathbf{D}|B) = L(A) / L(B)$
- $\Delta = \ln(LR^2) = 2 (\ln(L(A)) - \ln(L(B)))$
- Compare Δ to χ^2 distribution with $k_A - k_B$ degrees of freedom to determine if the Δ is significant or not
- The degrees of freedom difference is the difference in the number of free parameters in the models
- How many free parameters do the *JC* and *GTR* models have?

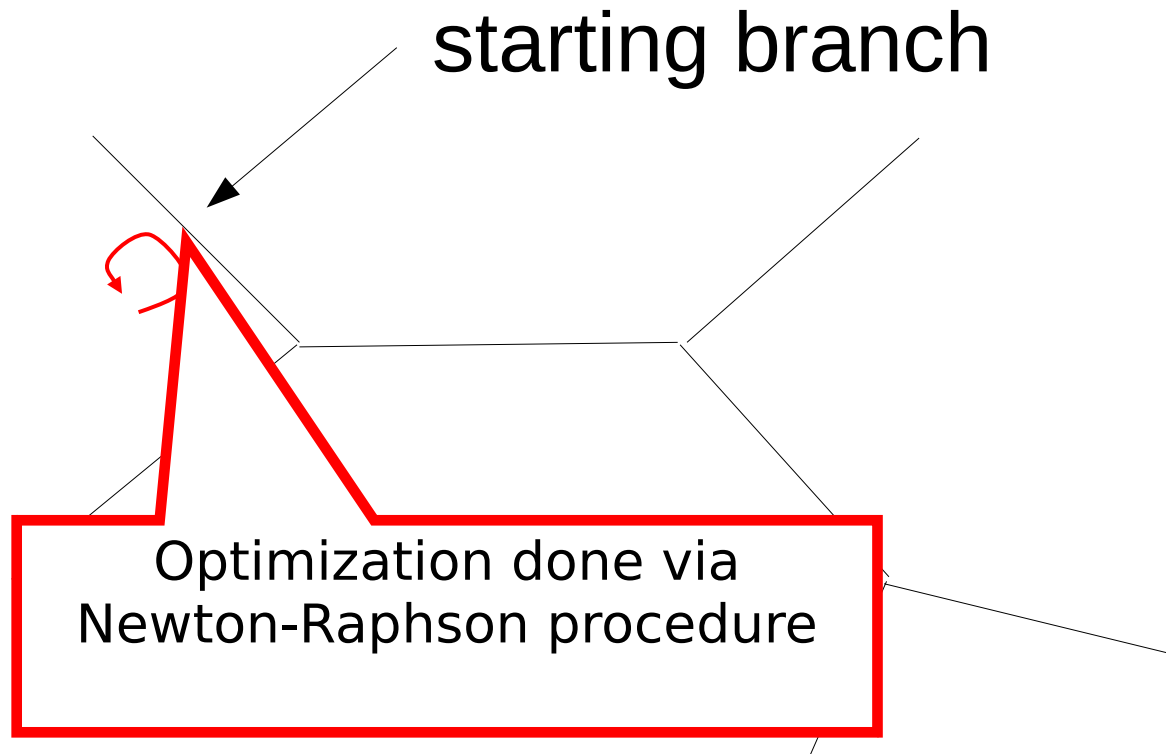
What if Models are not nested?

- One can use other criteria such as
 - *Akaike Information Criterion (AIC)*
 - *Bayesian Information Criterion (BIC)*
- I will spare you the details, but the basic idea always is:
 - Compute likelihood of alternative models
 - Penalize the more parameter-rich models

Outline

- More on Models
- The Newton-Raphson Method
- Data Structures for unrooted Phylogenetic Trees
- Implementing and Optimizing Likelihood Calculations
- Parallel Likelihood Calculations

Branch Length Optimization



Newton Raphson

- We want to find the branch length b that maximizes the likelihood $L(b)$ of the tree
- For this, we want to know where the *first* derivative of $L(b)$ is 0
- To achieve this numerically we use the Newton-Raphson procedure for root finding deploying the first and second derivative of the likelihood $L'(b)$ and $L''(b)$
- Note that, the likelihood only depends on branch b , all other model parameters (Q matrix, base frequencies, tree topology) remain fixed

Derivatives of $L(b)$

- To compute the derivatives of $L(b)$, we essentially need to be able to compute the derivatives of $P(b)$ since the rest is just sums and does not depend on b

- Recall

$$P(b) = e^{Qb} = Ue^{\Lambda b}U^{-1}$$

- thus

$$(P(b))' = U\Lambda e^{\Lambda b}U^{-1}$$

- and

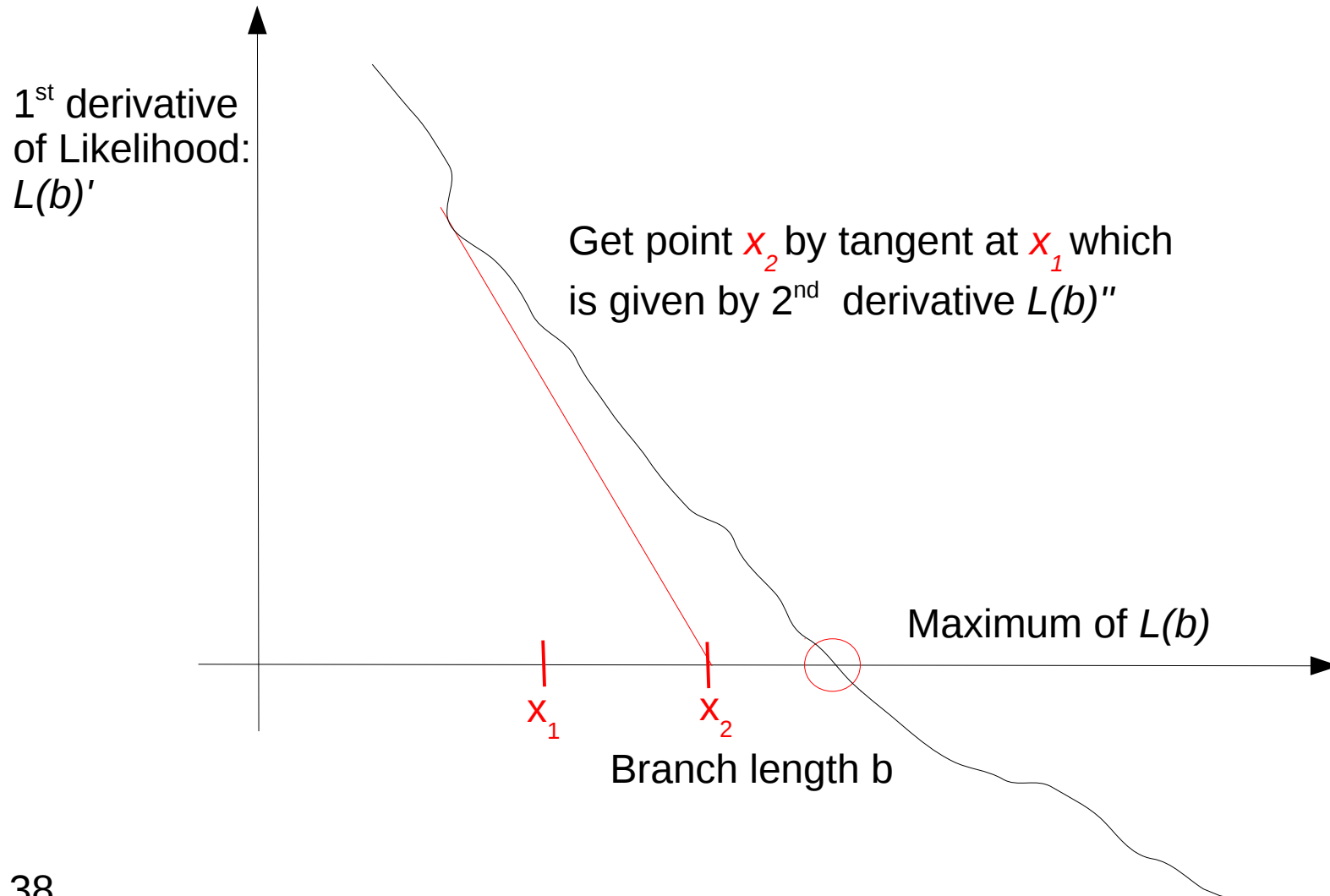
$$(P(b))'' = U\Lambda^2 e^{\Lambda b}U^{-1}$$

- In practice we compute the derivatives of the log likelihood $\log(L(b))$, but it is essentially the same

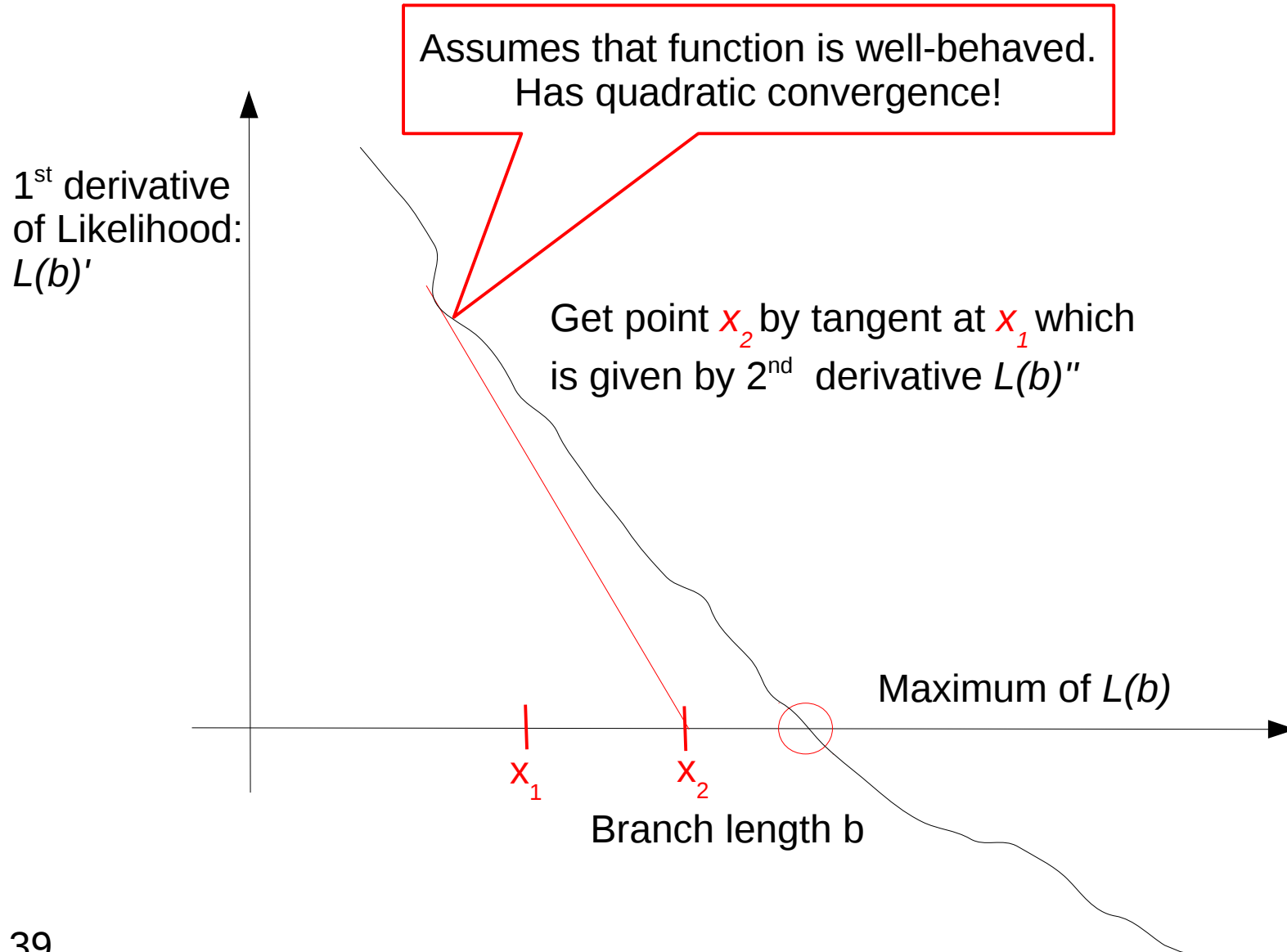
Derivatives of $\log(L(b))$

- 1st derivative: $L(b)' / L(b)$
- 2nd derivative: $(L(b) L(b)'' - (L(b)')^2) / L(b)^2$

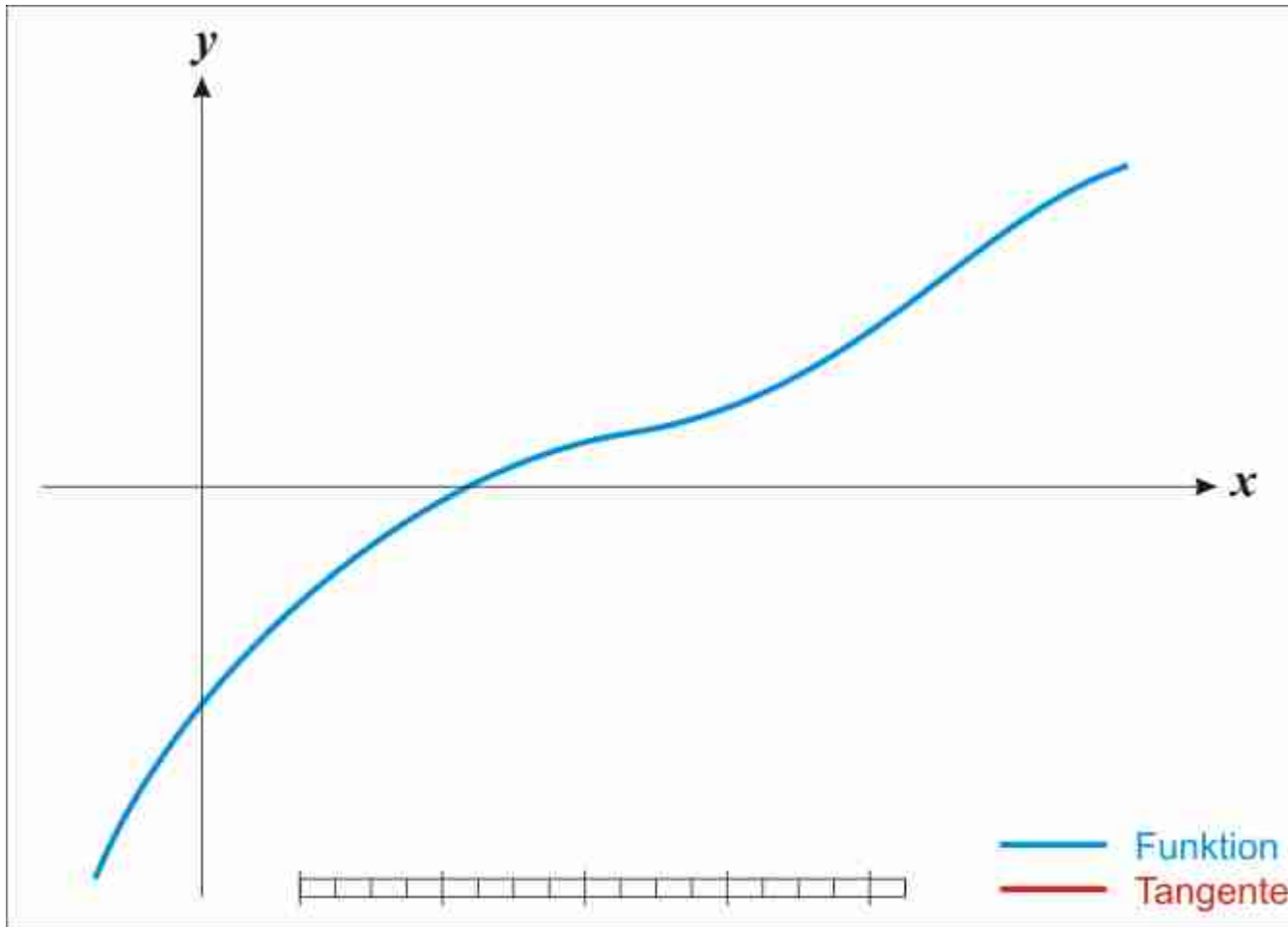
Newton Raphson



Newton Raphson



An animation



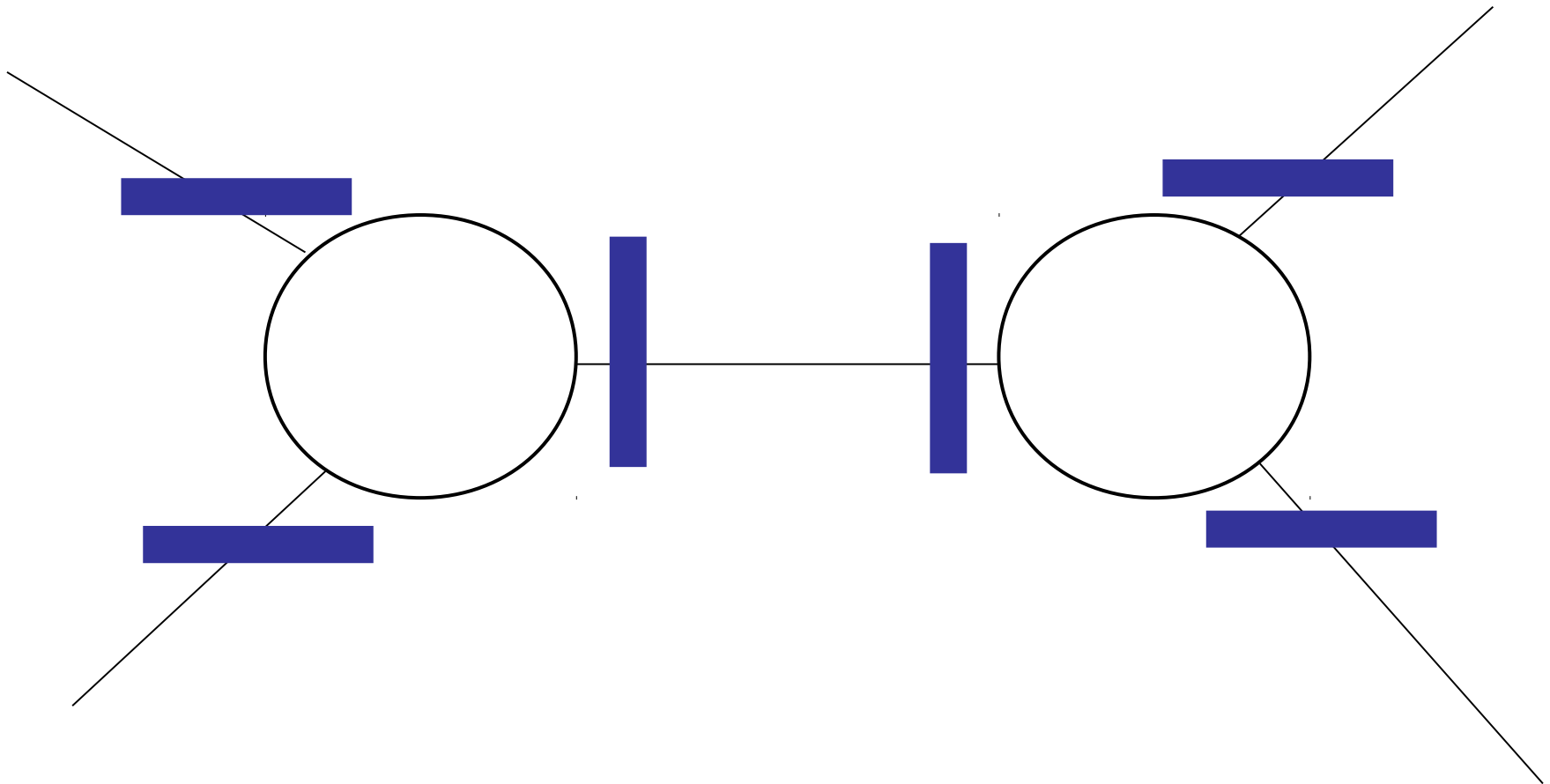
Outline

- More on Models
- The Newton-Raphson Method
- **Data Structures for unrooted Phylogenetic Trees**
- Implementing and Optimizing Likelihood Calculations
- Parallel Likelihood Calculations

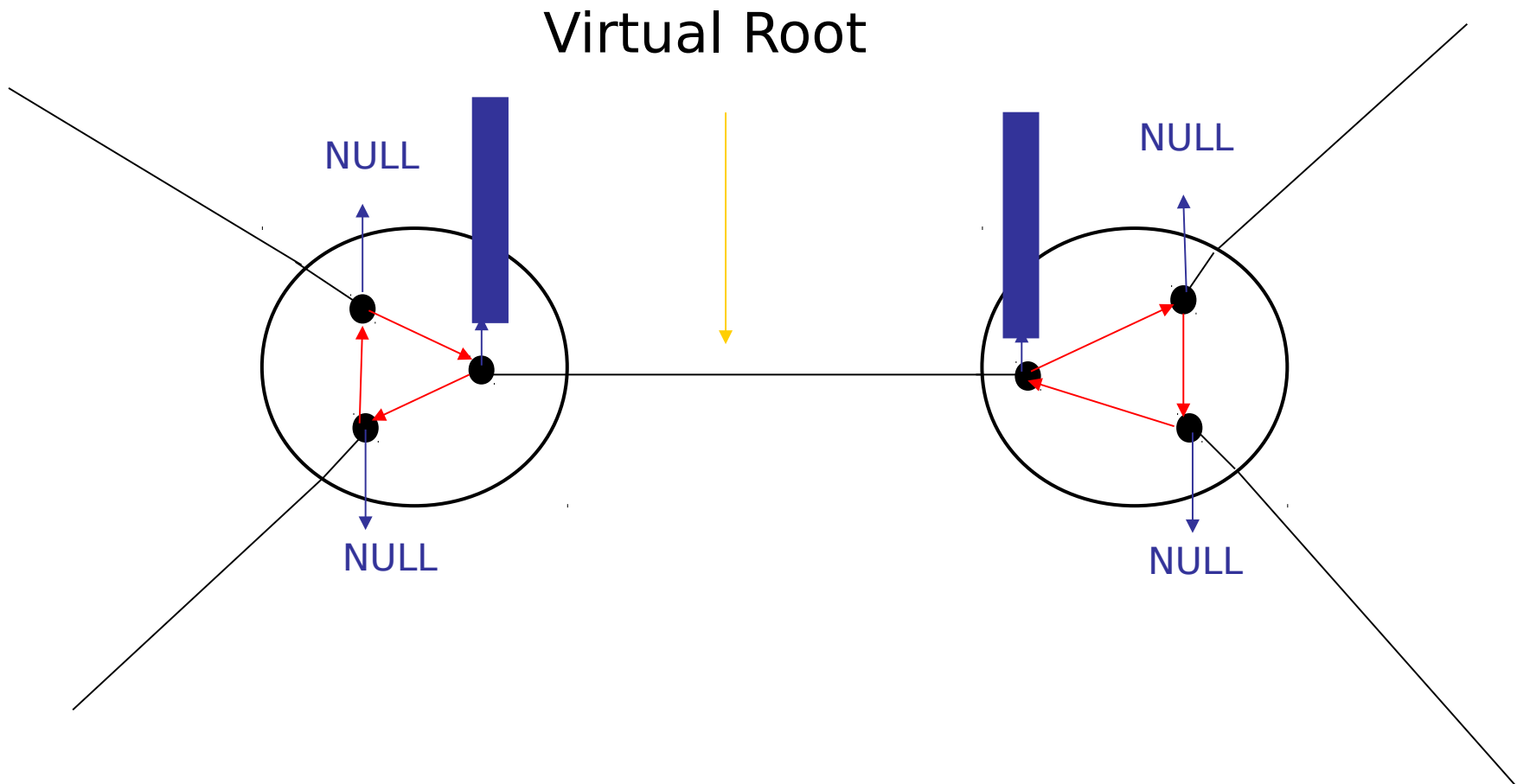
Data Structures for unrooted Trees

- Unrooted trees with dynamically changing virtual roots need a dedicated tree data structure
- Why can the virtual root positions change dynamically?
- If we apply a topological move (NNI, SPR, TBR) will we have to re-compute all conditional likelihood vectors?

Memory Organization: Conditional Likelihood Vectors with an Unrooted View



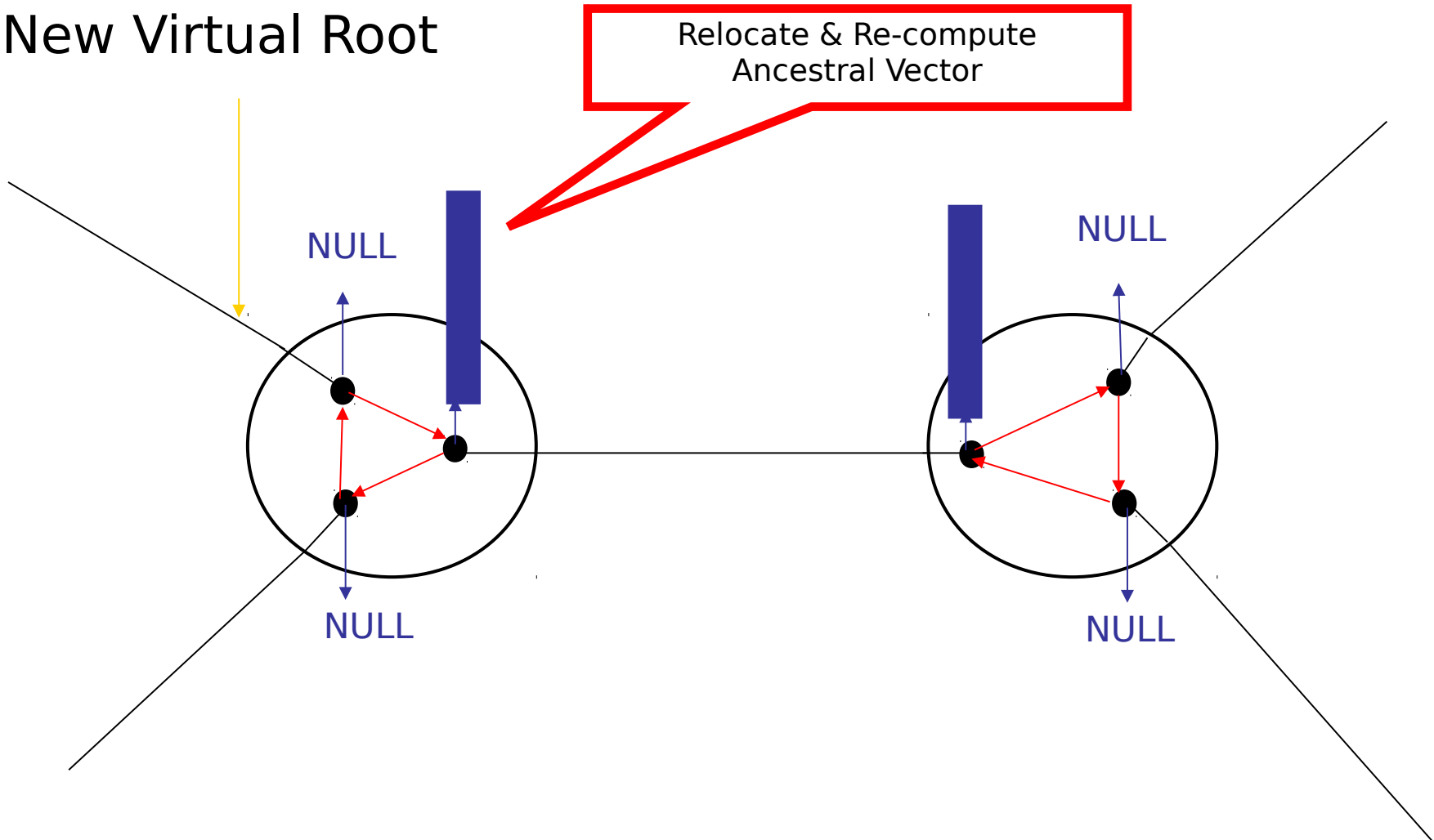
Memory Organization: Conditional Likelihood Vectors with a Rooted View



Memory Organization: CLVs with a Rooted View

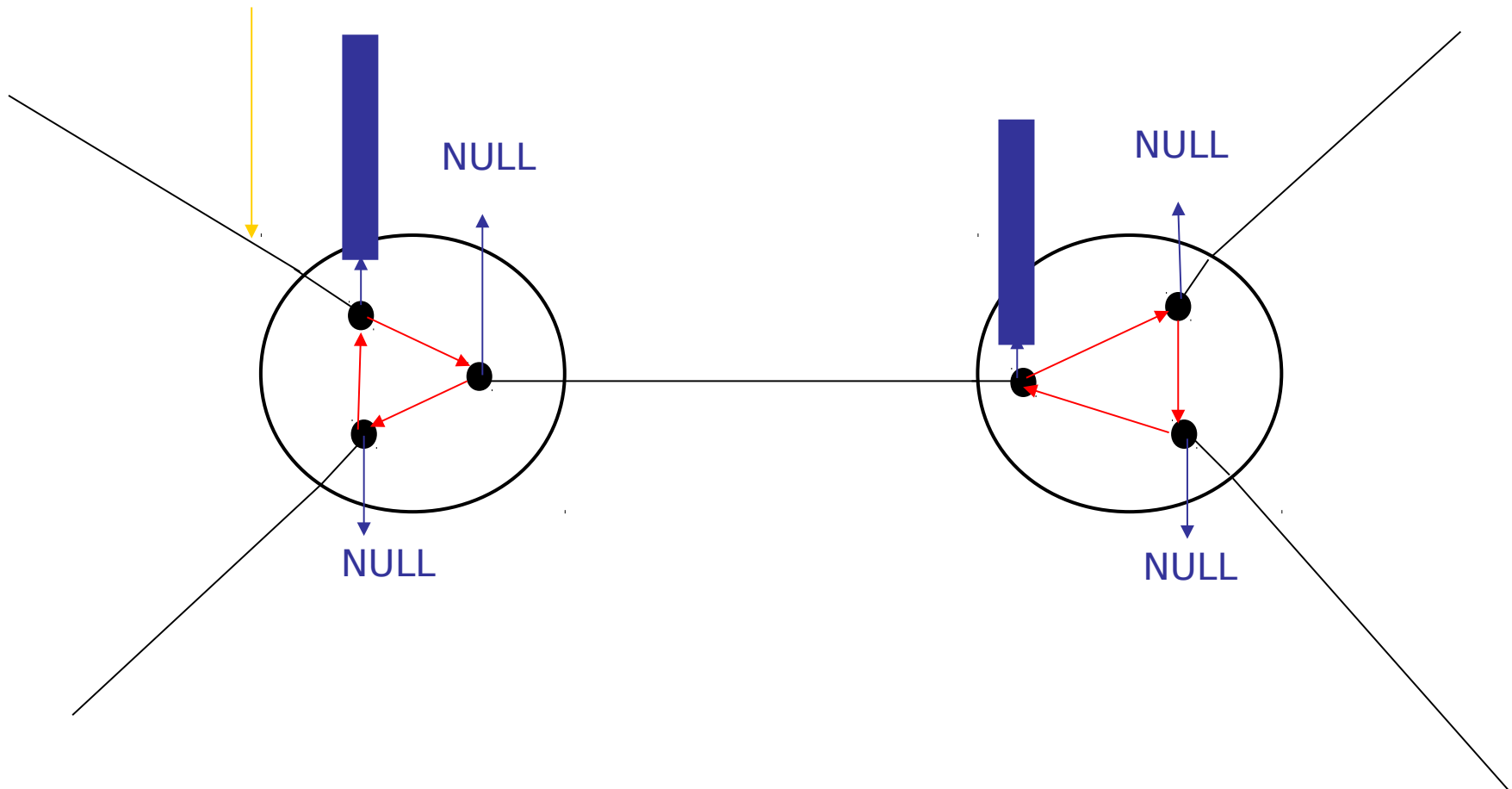
New Virtual Root

Relocate & Re-compute Ancestral Vector

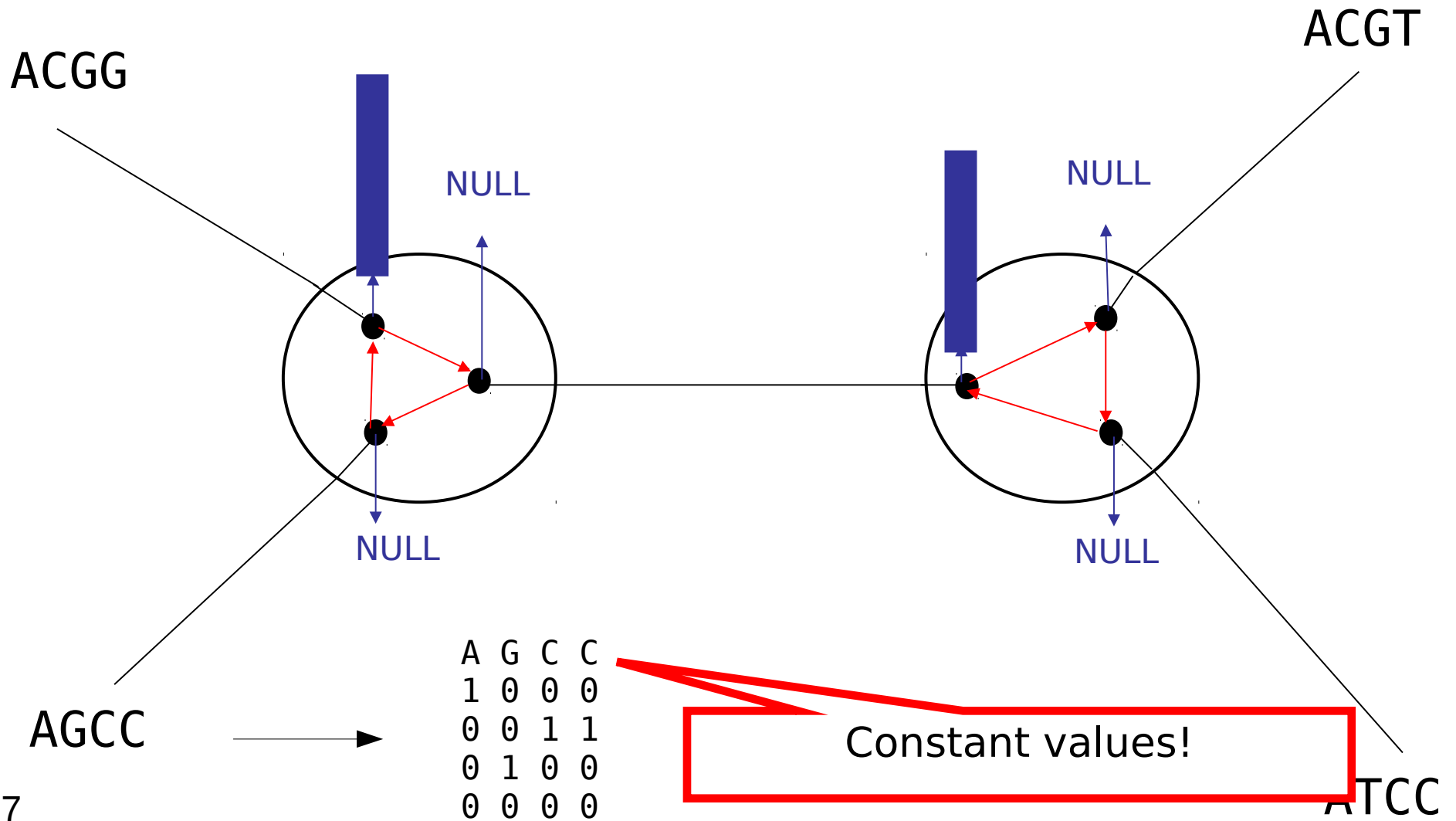


Memory Organization: Ancestral Vectors with a Rooted View

New Virtual Root



Memory Organization: Tip Vectors

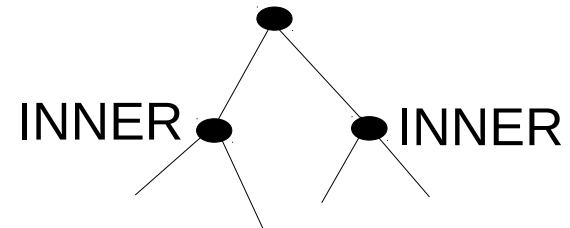
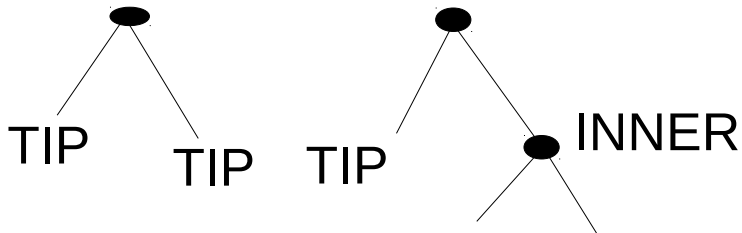


Outline

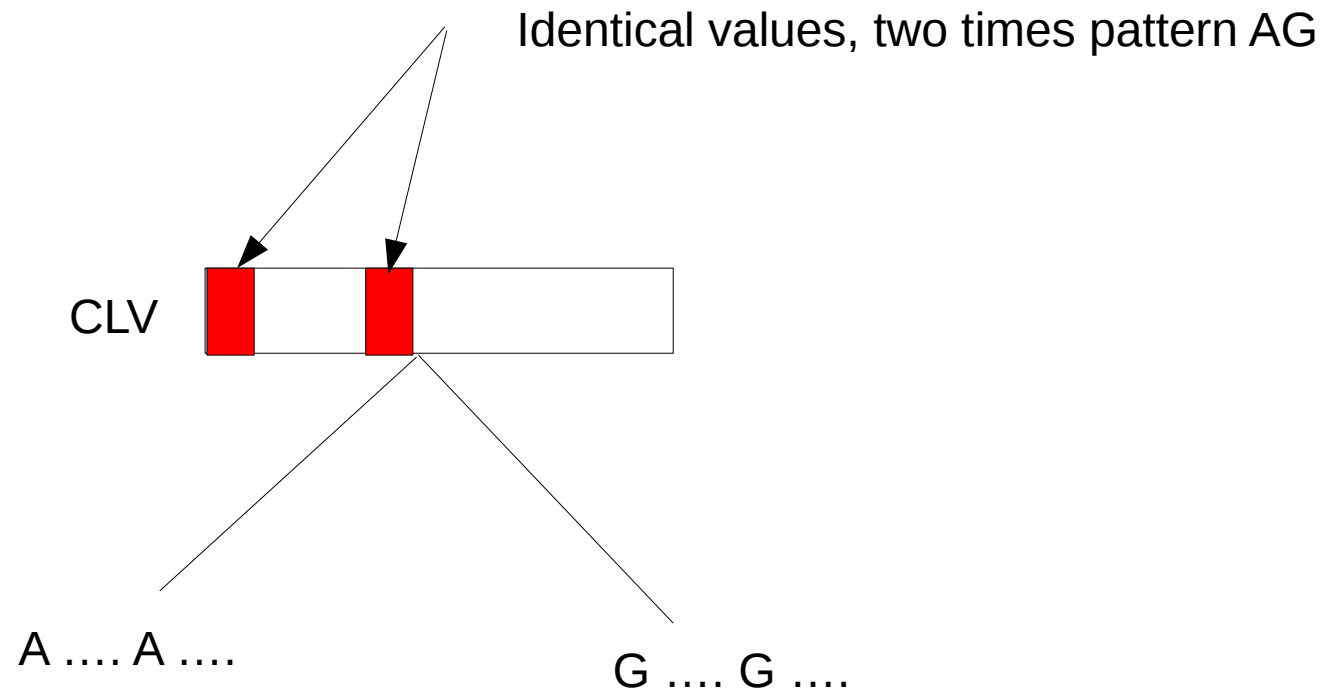
- More on Models
- The Newton-Raphson Method
- Data Structures for unrooted Phylogenetic Trees
- Implementing and Optimizing Likelihood Calculations
- Parallel Likelihood Calculations

Optimization of Likelihood Calculations

- Use SSE3 & AVX vector intrinsics
- Also: GPUs, FPGAs, the Intel Xeon PHI
- Special implementations (why?) for computing CLVs:

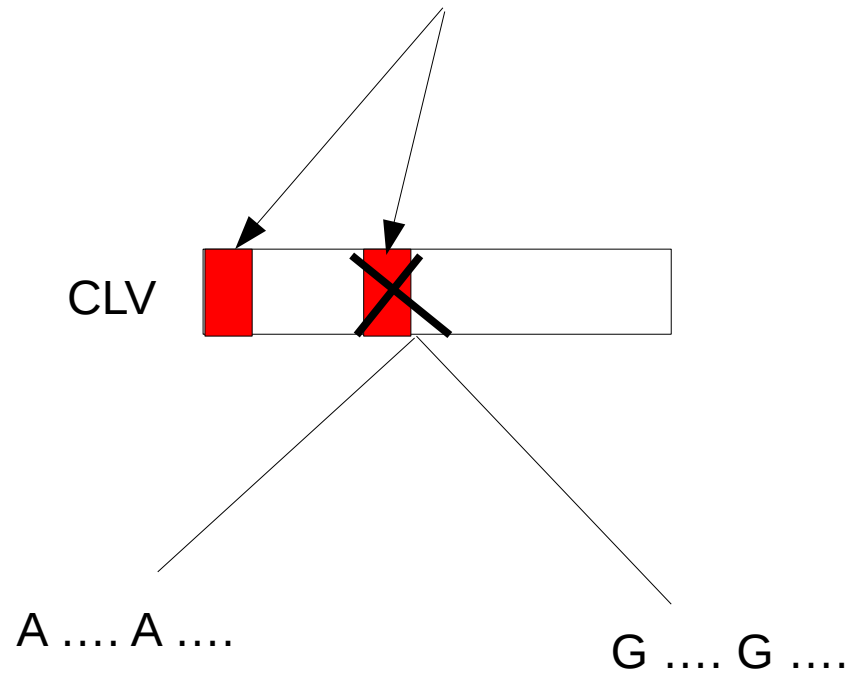


Repeating Patterns



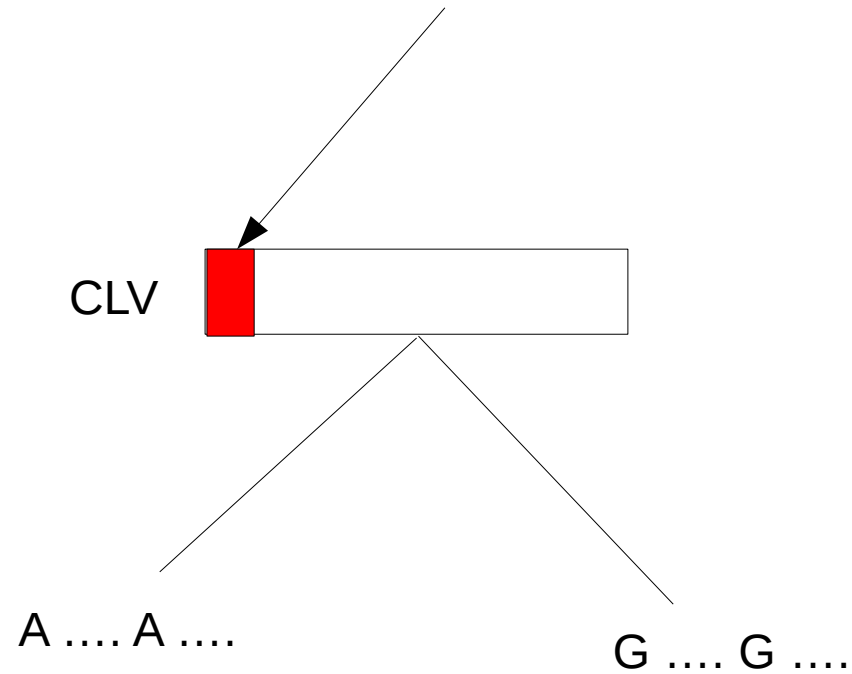
Repeating Patterns

Detect identical patterns and omit second computation



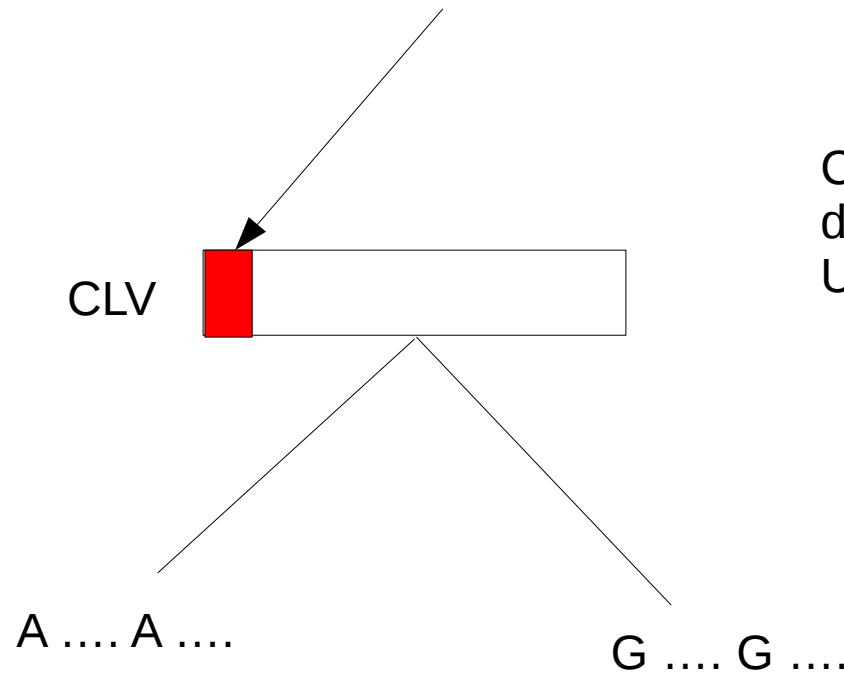
Repeating Patterns

Also, shorten CLV → less memory required



Repeating Patterns

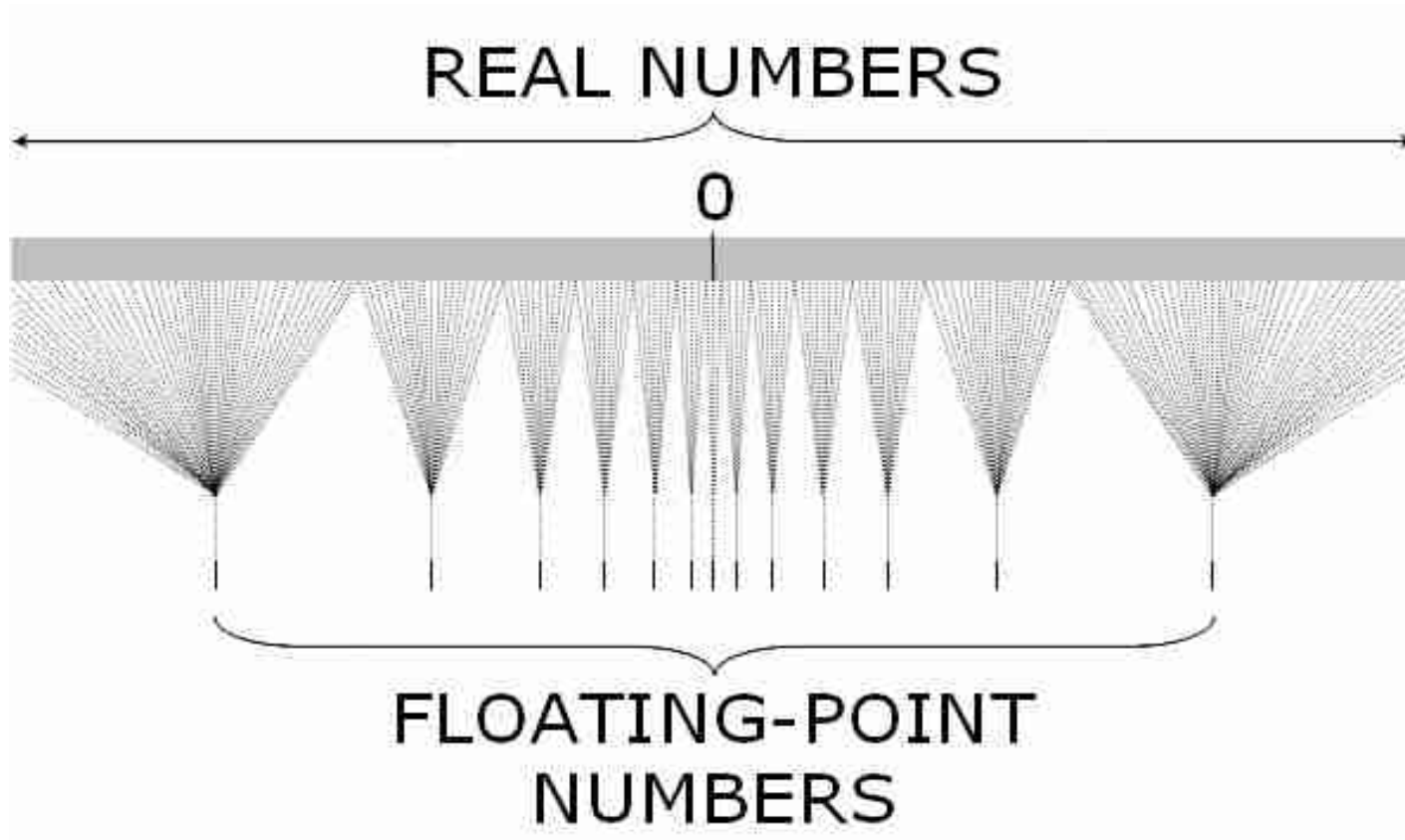
Also, shorten CLV → less memory required



Challenge: Efficient data structure to detect & store repeats
Up to 10-fold run-time improvements

Floating Point Numbers

- Machine numbers are an imperfect mapping of the infinite real numbers to a finite number of machine values!



Floating Point Arithmetics: The Root of All Evil

- Computational science mostly relies on floating-point intensive codes
- How do we verify these codes?
- We stand on shaky grounds
- Scientists using those codes assume that there are no bugs
- Double precision arithmetics required for certain applications
- Who knows what de-normalized floating point numbers are?

→ Please have a look at:

J. Björndalen, O. Anshus: “Trusting floating point benchmarks-are your benchmarks really data-independent?” Applied Parallel Computing. State of the art in Scientific Computing 2010; pp 178-188, Springer.

and at my micro-benchmark at:

<https://github.com/stamatak/denormalizedFloatingPointNumbers>

Floating Point Arithmetics: The Root of All Evil

- Computational science mostly relies on codes
- How do we verify these codes?
- We stand on shaky grounds
- Scientists using those codes assume that there are no bugs
- Double precision arithmetics required for certain applications
- Who knows what de-normalized floating point numbers are?

Why is this relevant when
Talking about Maximum
Likelihood?

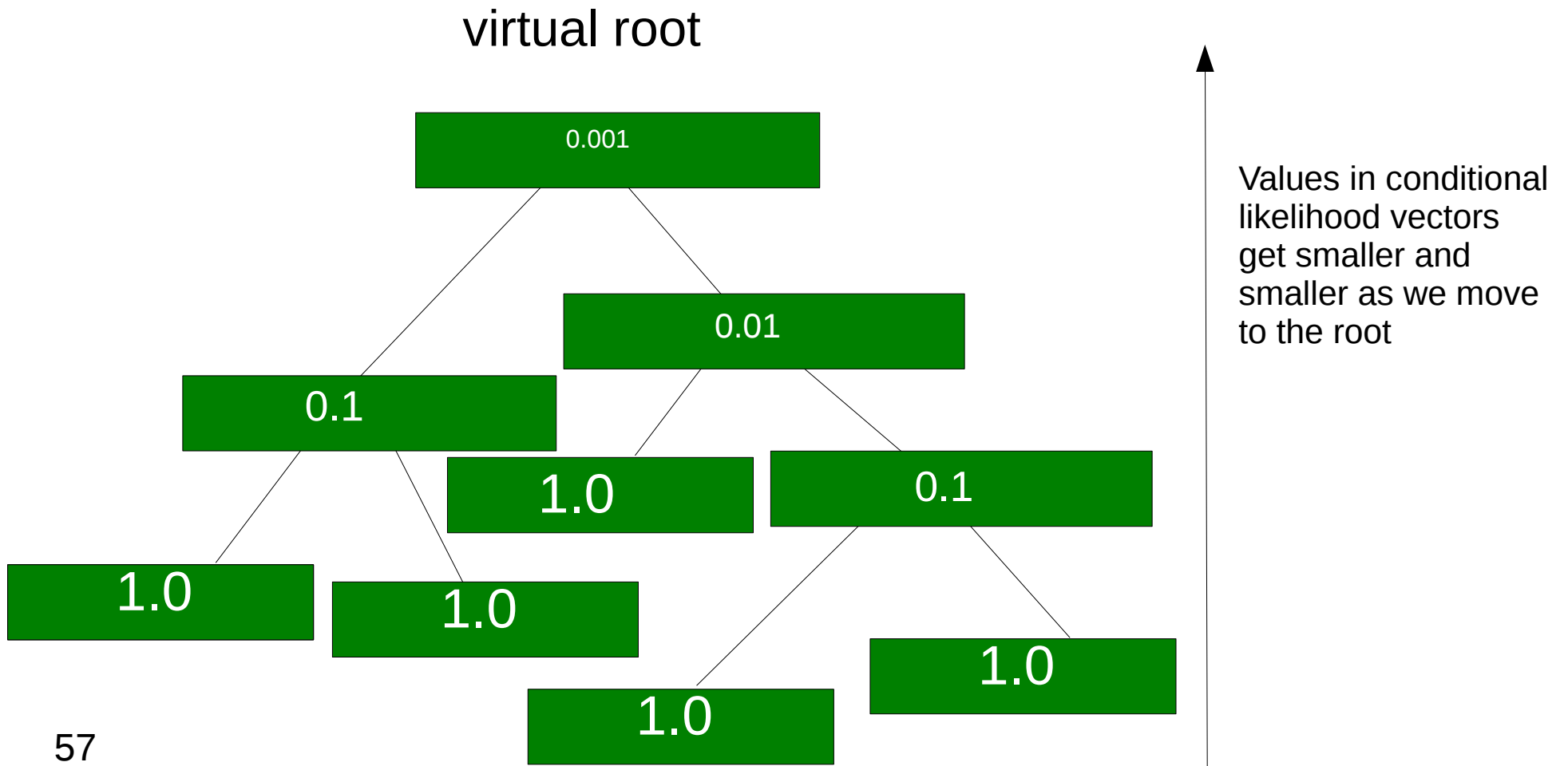
→ Please have a look at:

J. Björndalen, O. Anshus: “Trusting floating point benchmarks-are your benchmarks really data-independent?” Applied Parallel Computing. State of the art in Scientific Computing 2010; pp 178-188, Springer.

and at my micro-benchmark at:

<https://github.com/stamatak/denormalizedFloatingPointNumbers>

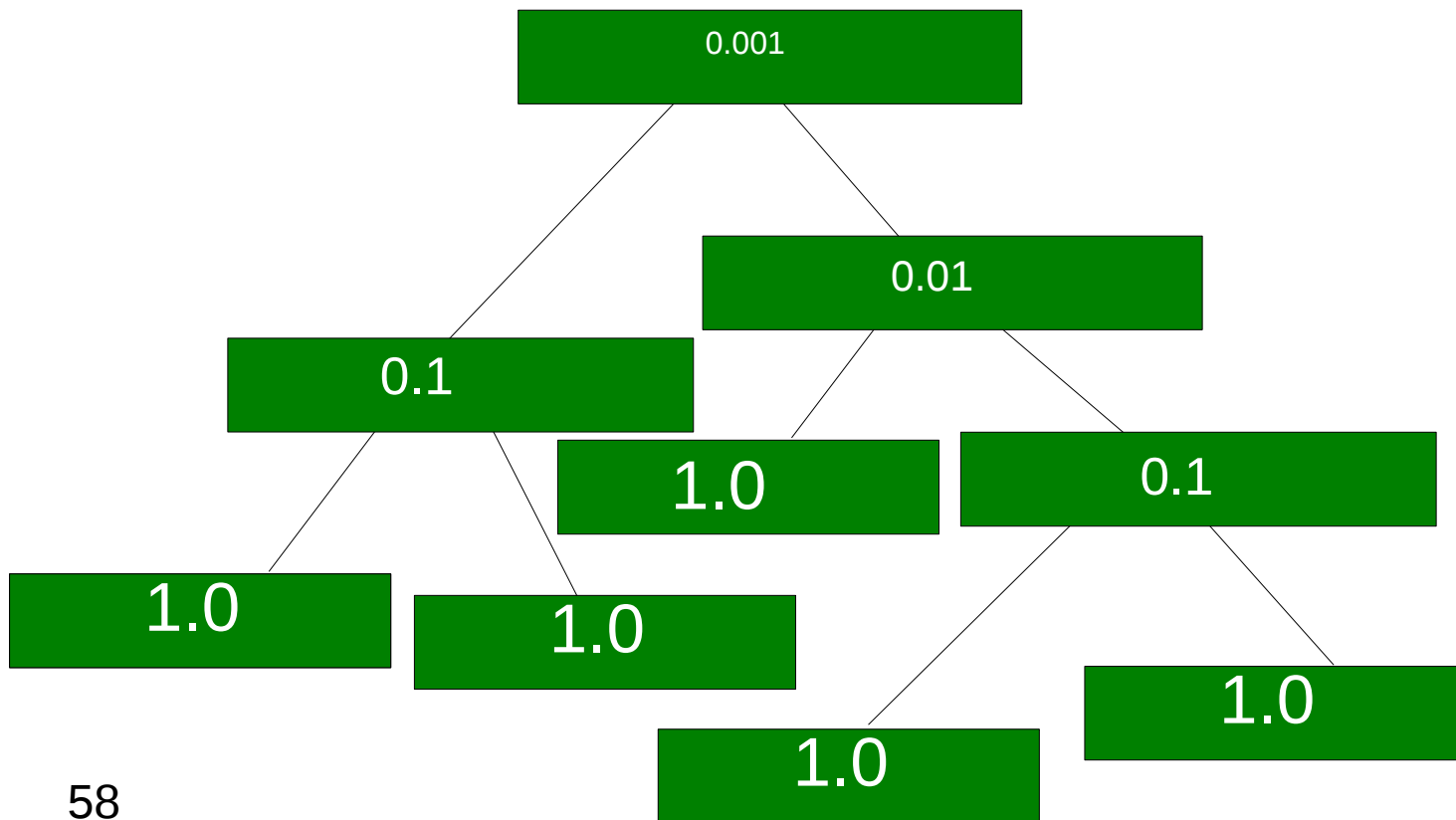
Post-order Traversal



Post-order Traversal

We need to apply numerical scaling techniques to avoid underflow!

virtual root

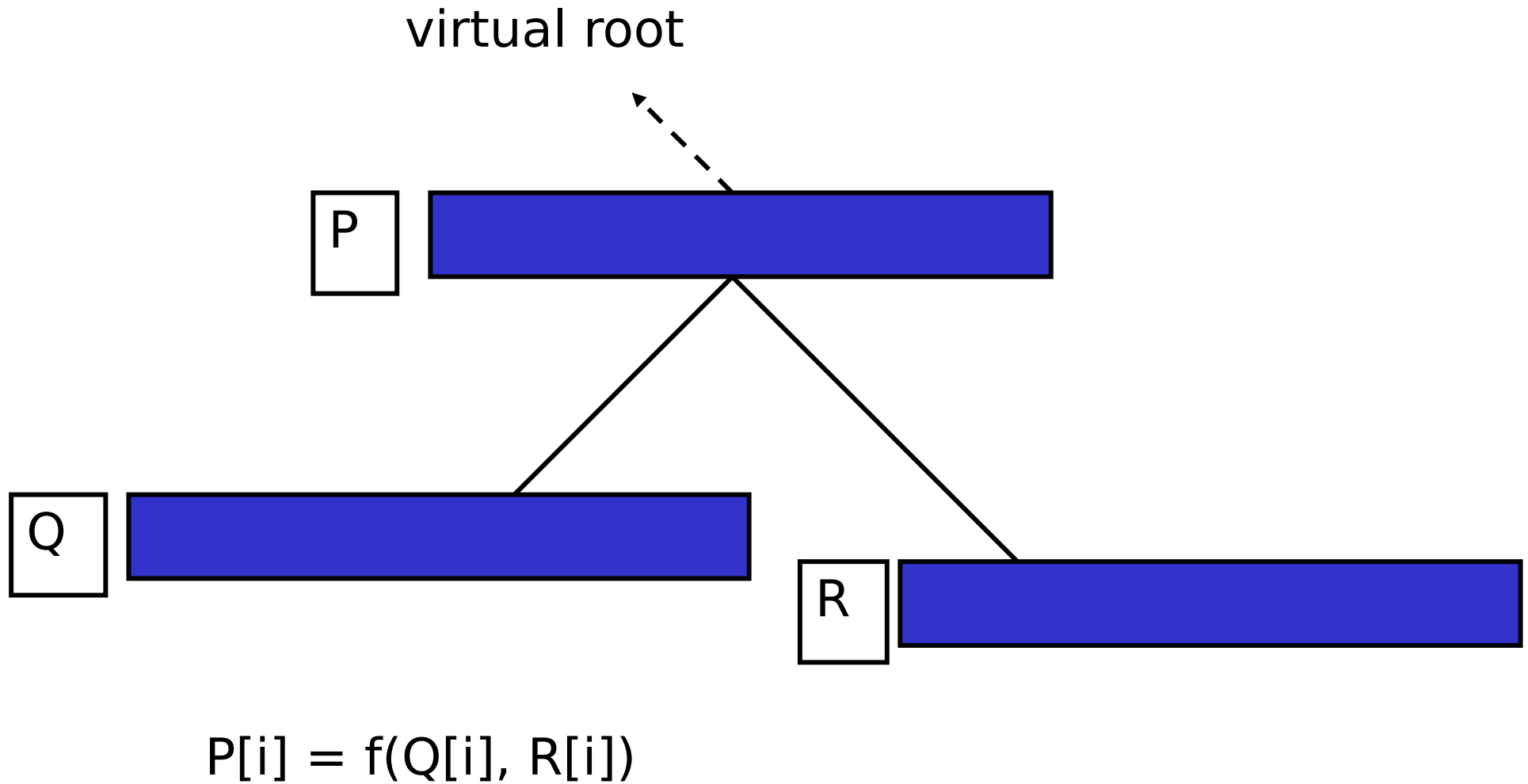


Values in conditional likelihood vectors get smaller and smaller as we move to the root → this needs to be handled!

Outline

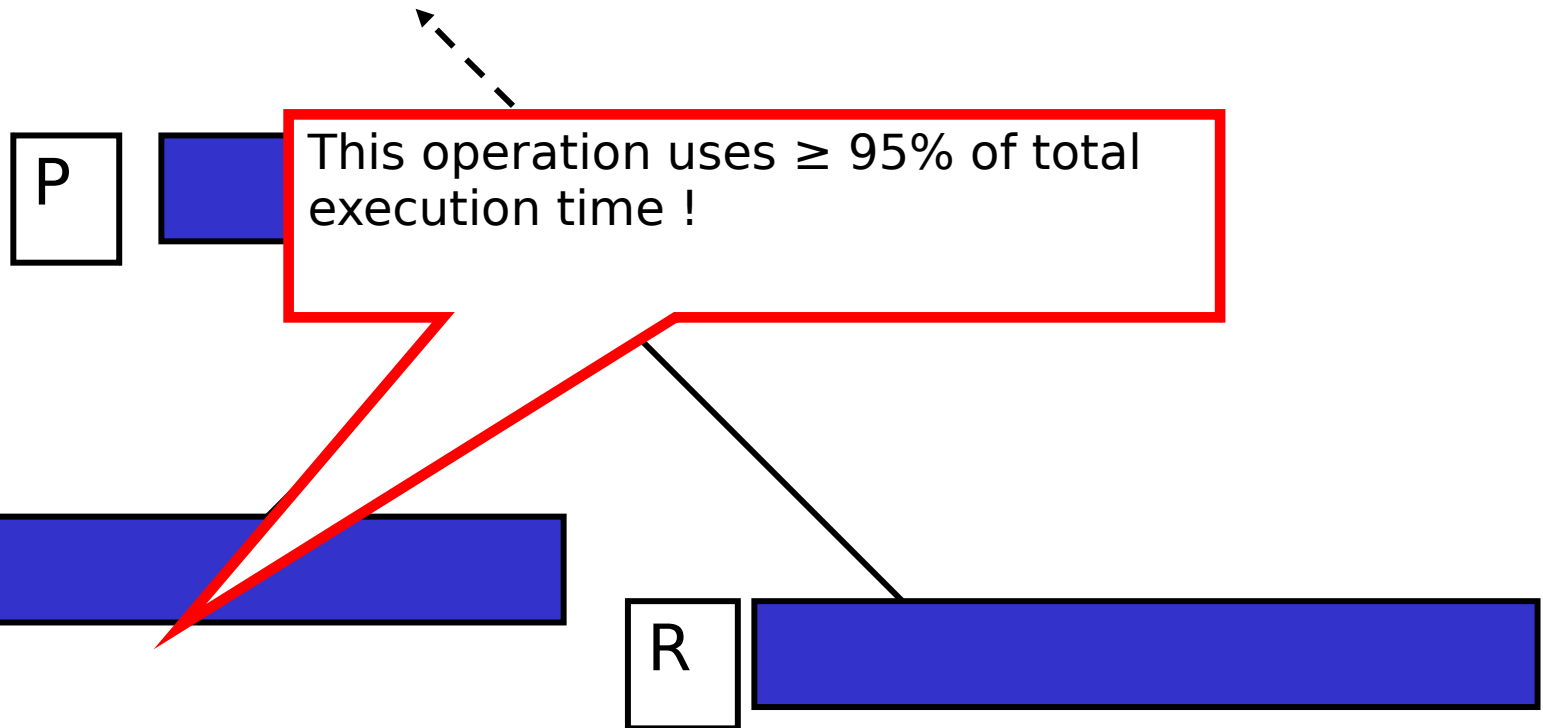
- More on Models
- The Newton-Raphson Method
- Data Structures for unrooted Phylogenetic Trees
- Implementing and Optimizing Likelihood Calculations
- **Parallel Likelihood Calculations**

Loop Level Parallelism



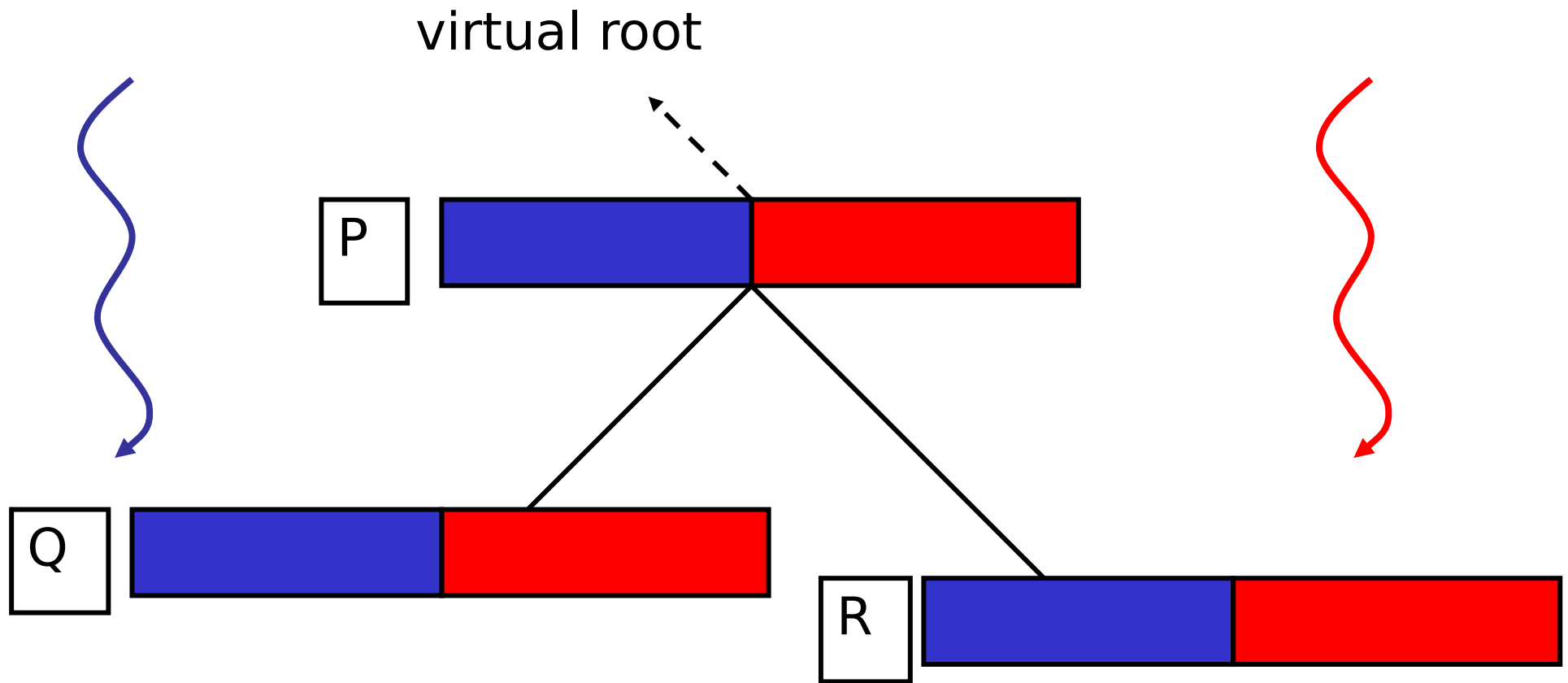
Loop Level Parallelism

virtual root

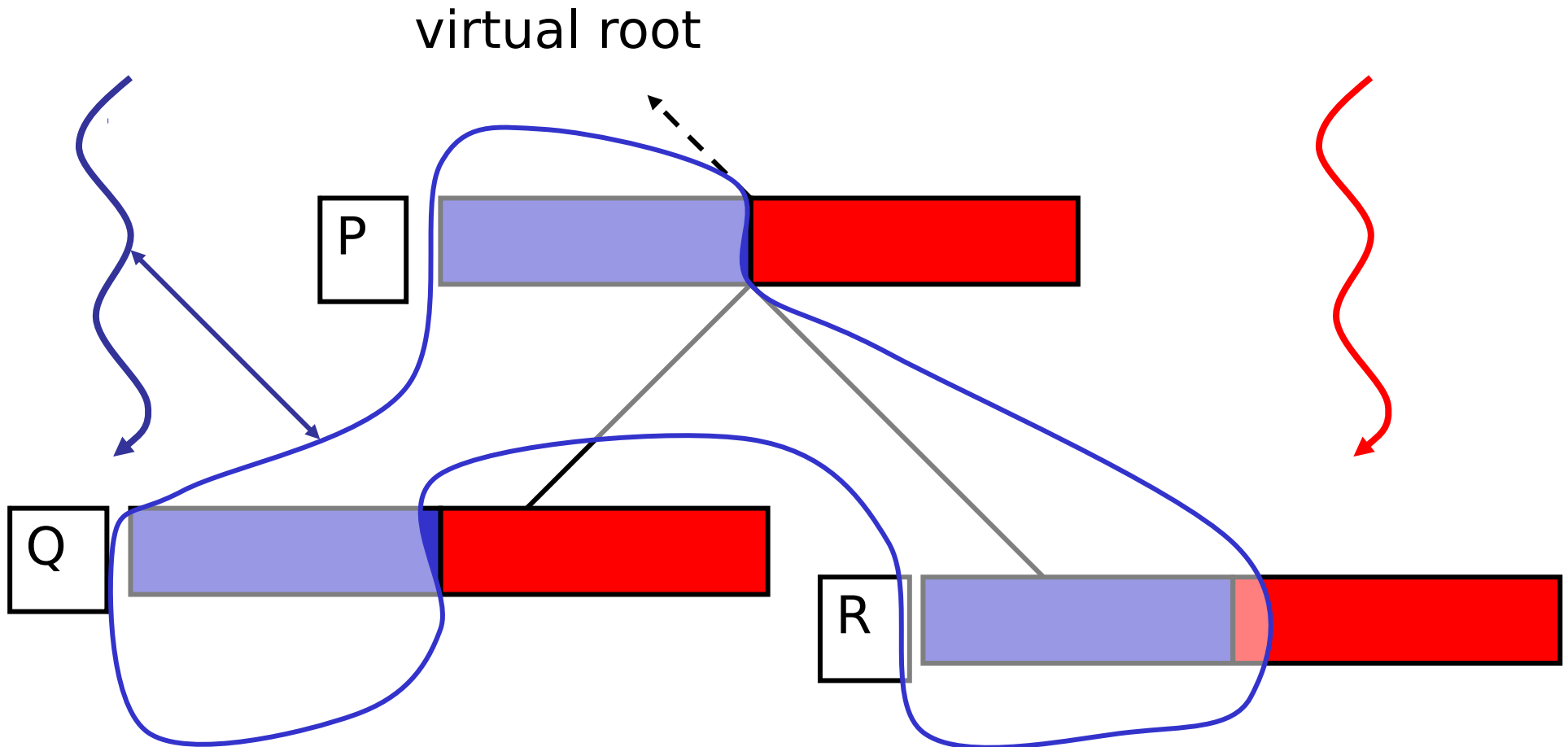


$$P[i] = f(Q[i], R[i])$$

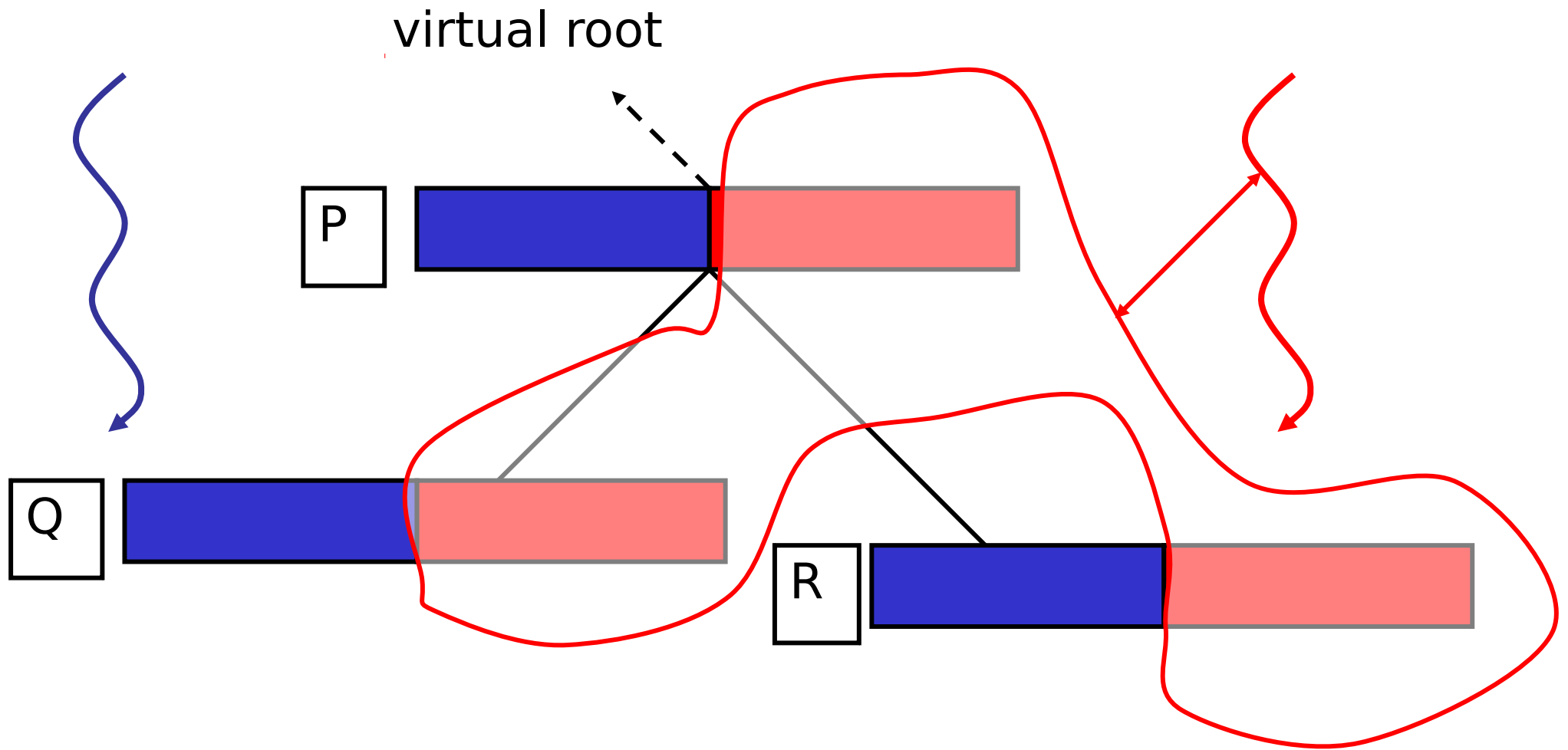
Loop Level Parallelism



Loop Level Parallelism



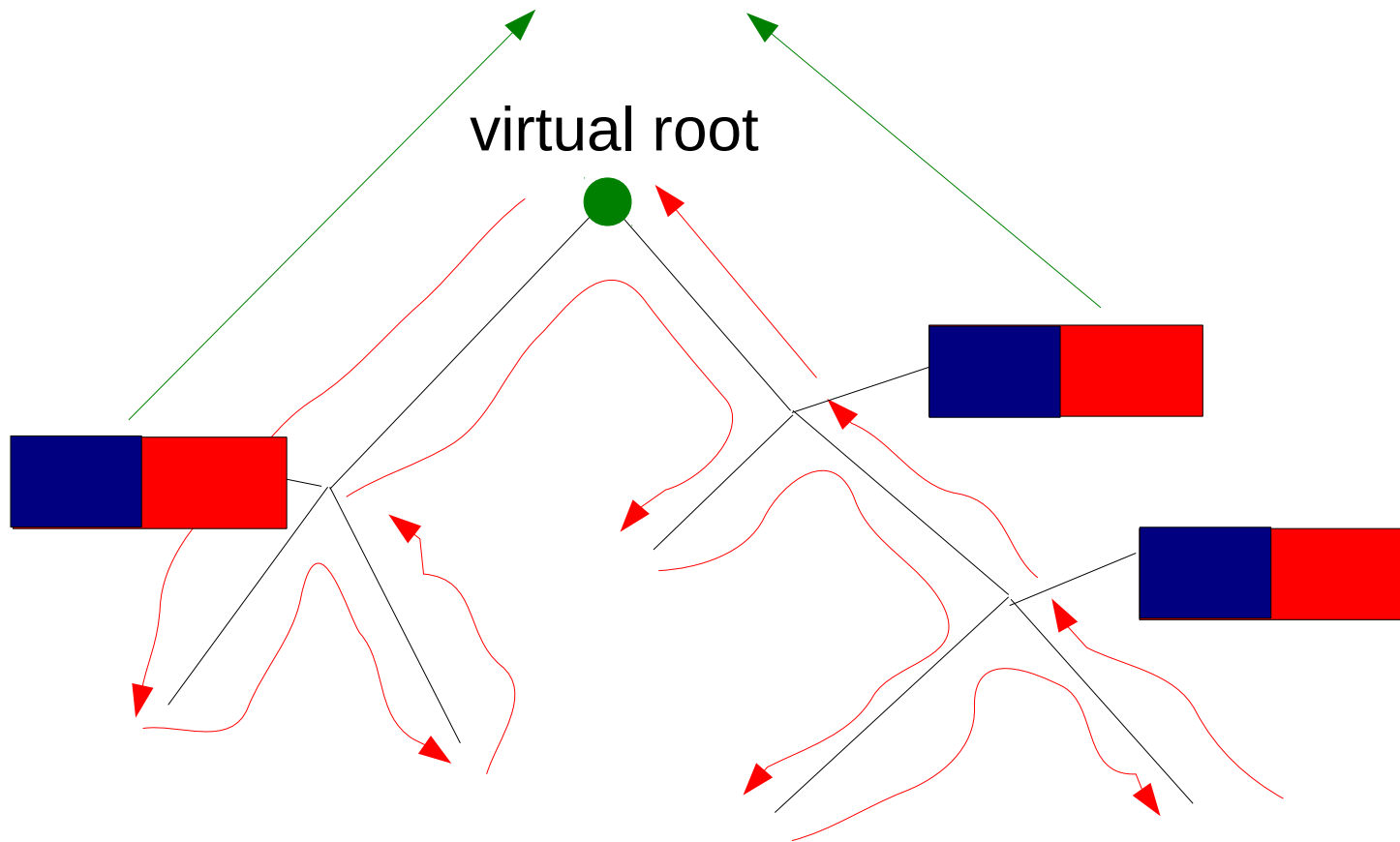
Loop Level Parallelism



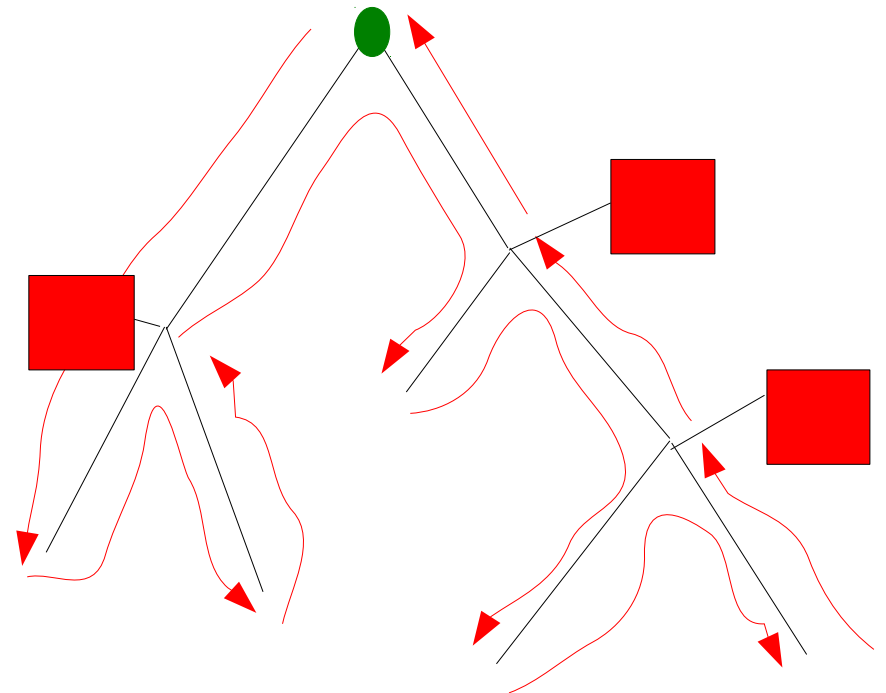
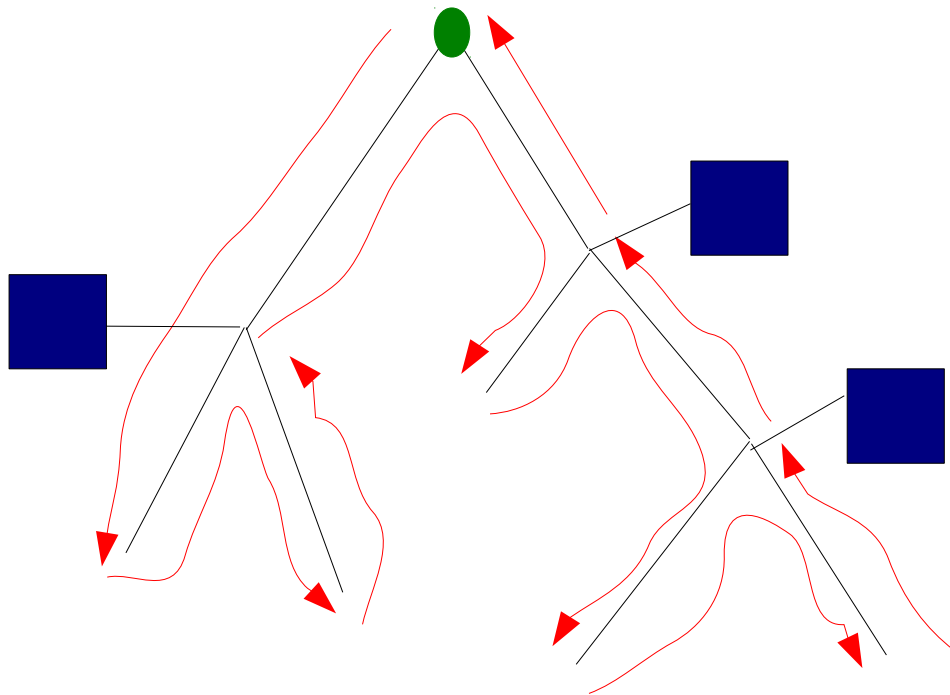
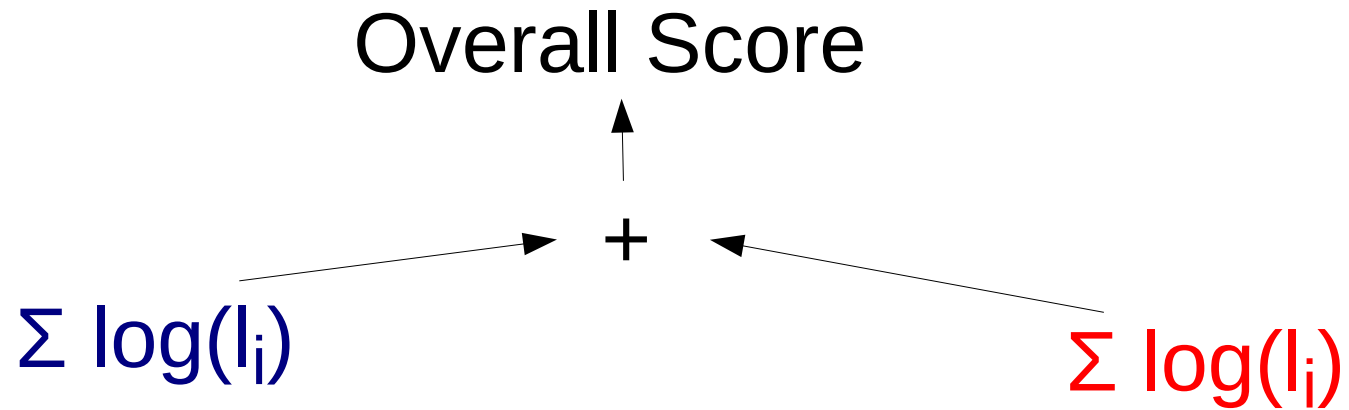
Parallel Post-order Traversal

Only need to synchronize at the root
→ MPI_Reduce() to calculate:

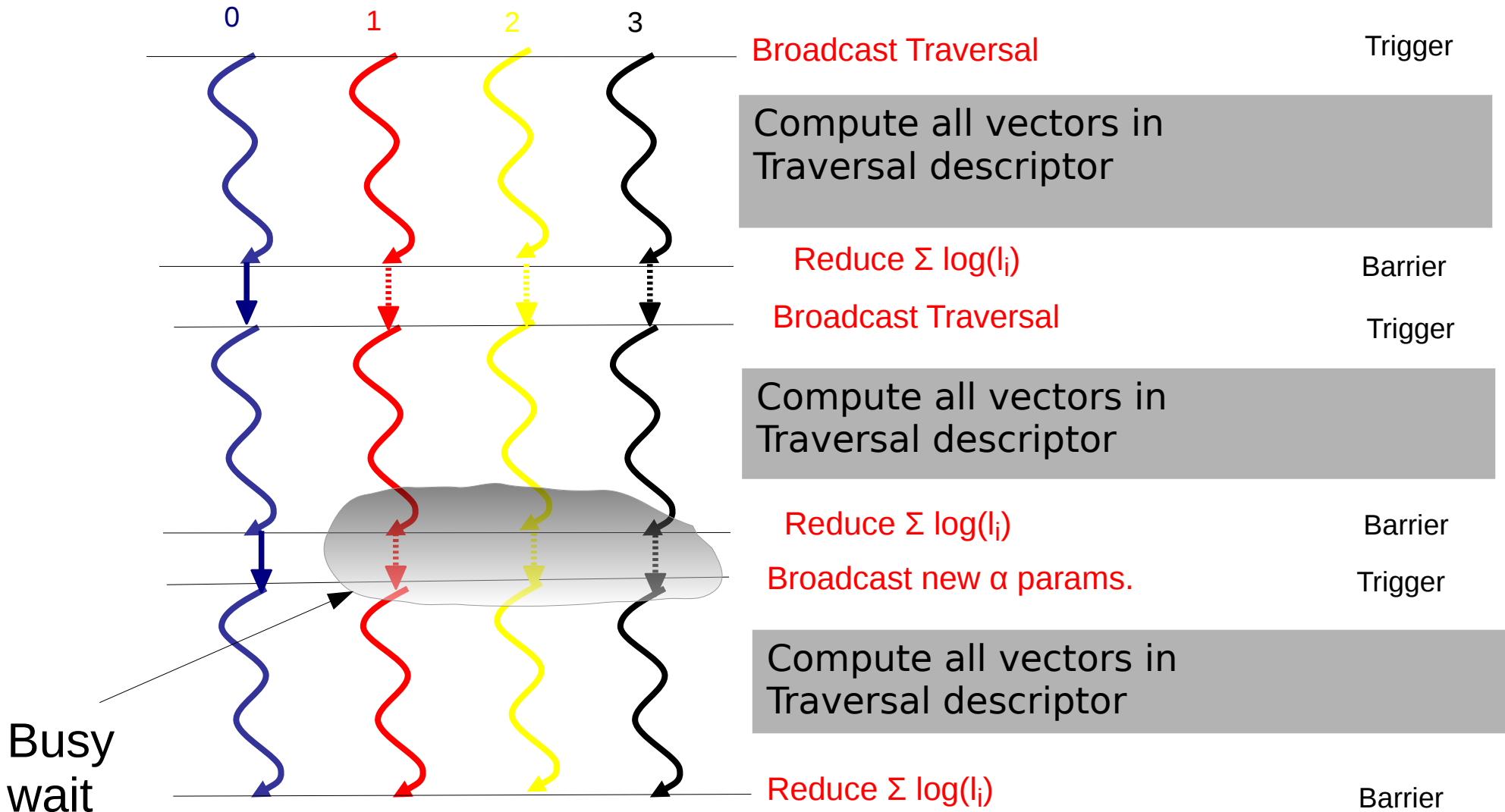
$$\sum \log(l_i)$$



Parallel Post-order Traversal



Classic Fork-Join with Busy-Wait

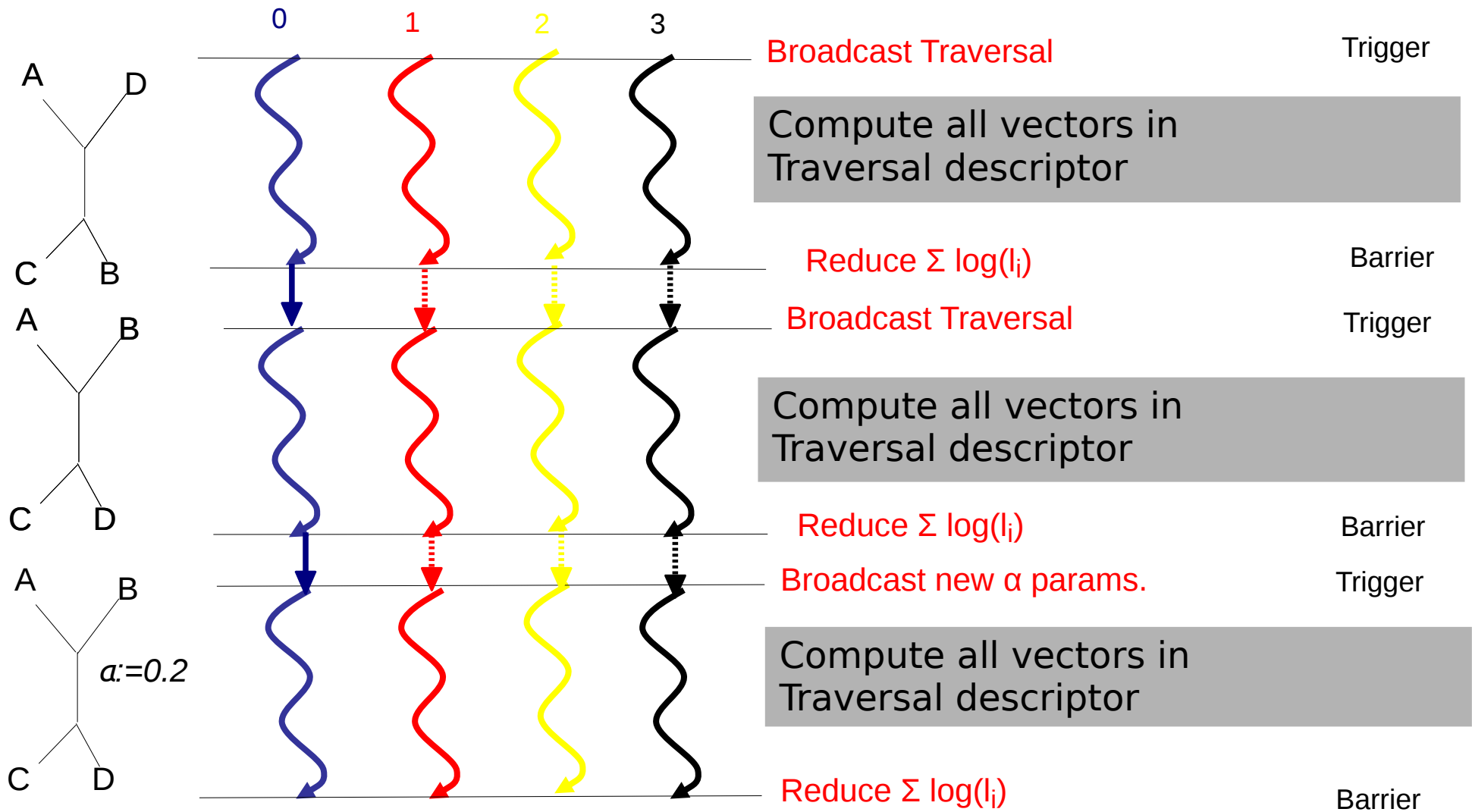


Synchronizations in RAxML with Pthreads

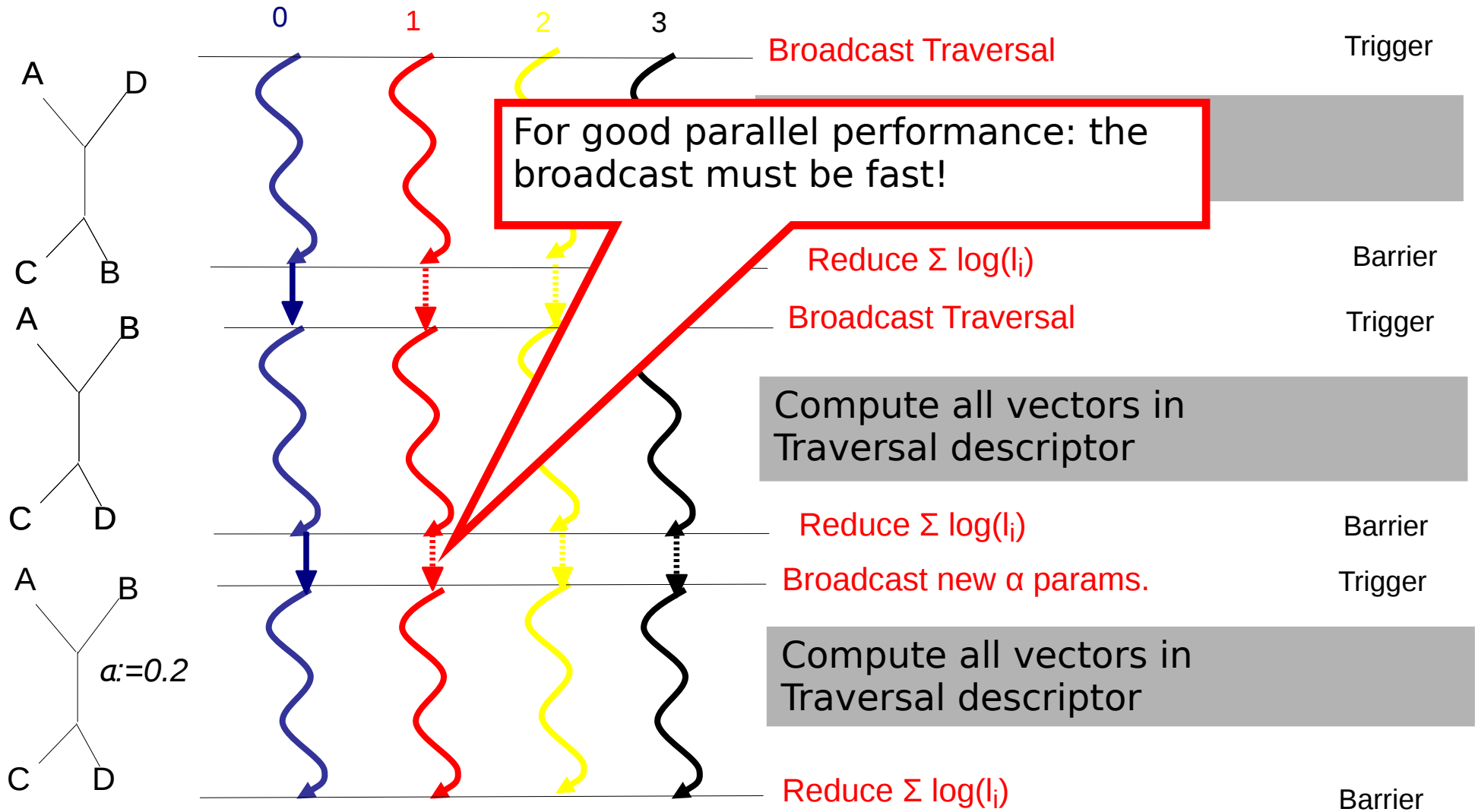
- RAxML Pthreads for a run time of about **10 seconds** on 16 cores/threads
- 404 taxa 7429 sites: **194,000** Barriers
- 1481 taxa 1241 sites: **739,000** Barriers
- A paper on performance of alternative PThreads barrier implementations:

S.A. Berger, A. Stamatakis: "Assessment of Barrier Implementations for Fine-Grain Parallel Regions on Current Multi-core Architectures", *IEEE Cluster* 2010.

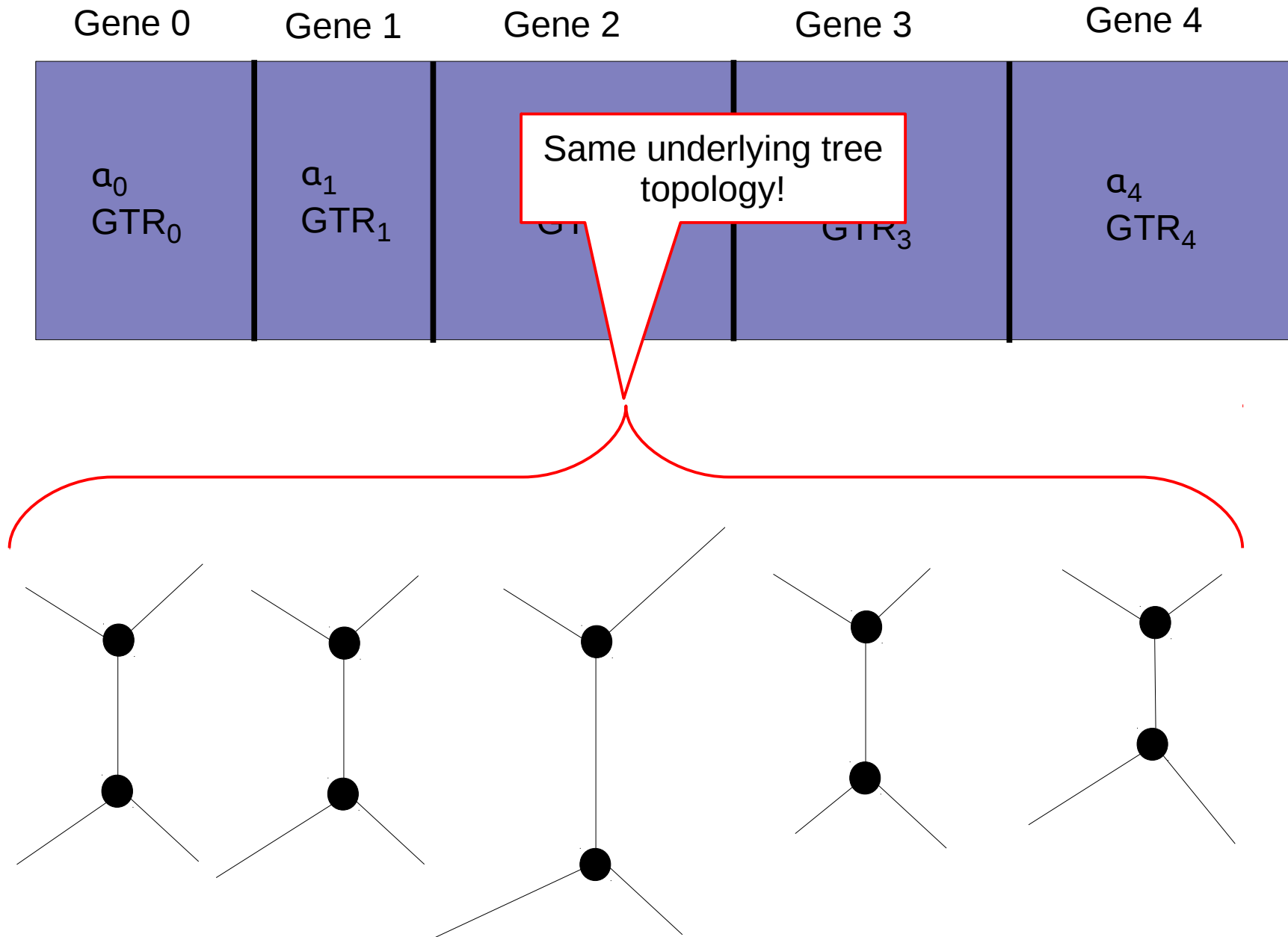
Classic Fork-Join with Busy-Wait



Classic Fork-Join with Busy-Wait



Problems start with partitioned datasets!



Parallel Performance Problems

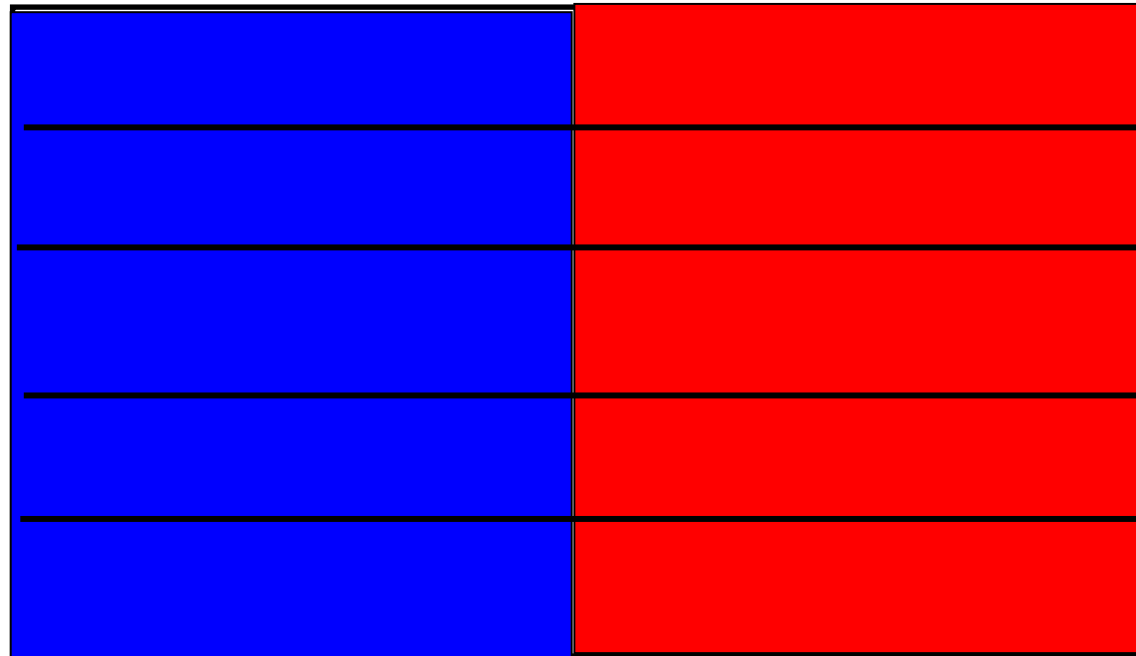
- They all start with partitioned datasets!
- How do we distribute partitions to processors?
- How do we calculate parameter changes?
- How much time does our broadcast take?
- Goal: Keep all processors busy all the time
 - minimize communication and synchronization!

Example

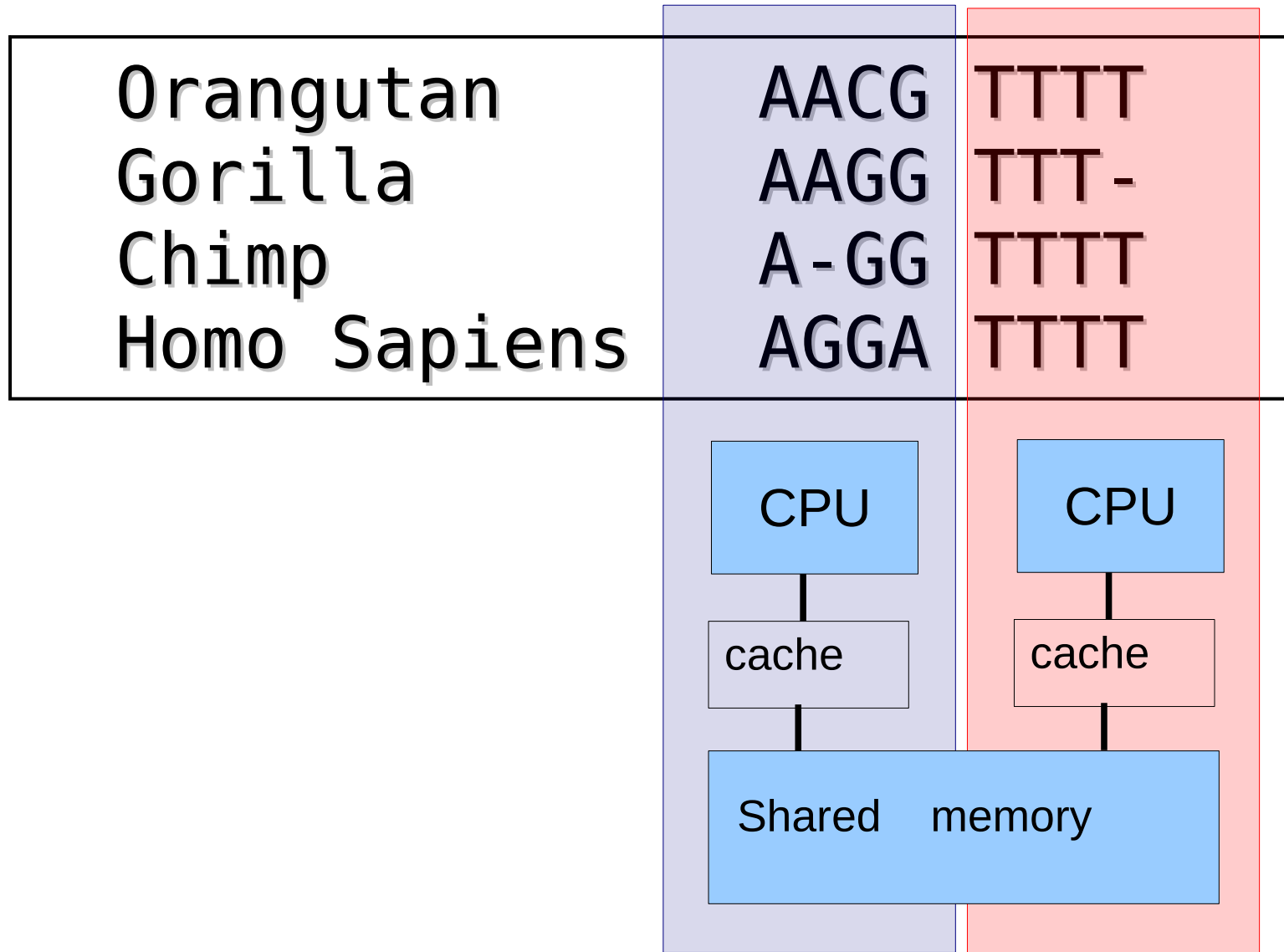
Blue Gene

Red Gene

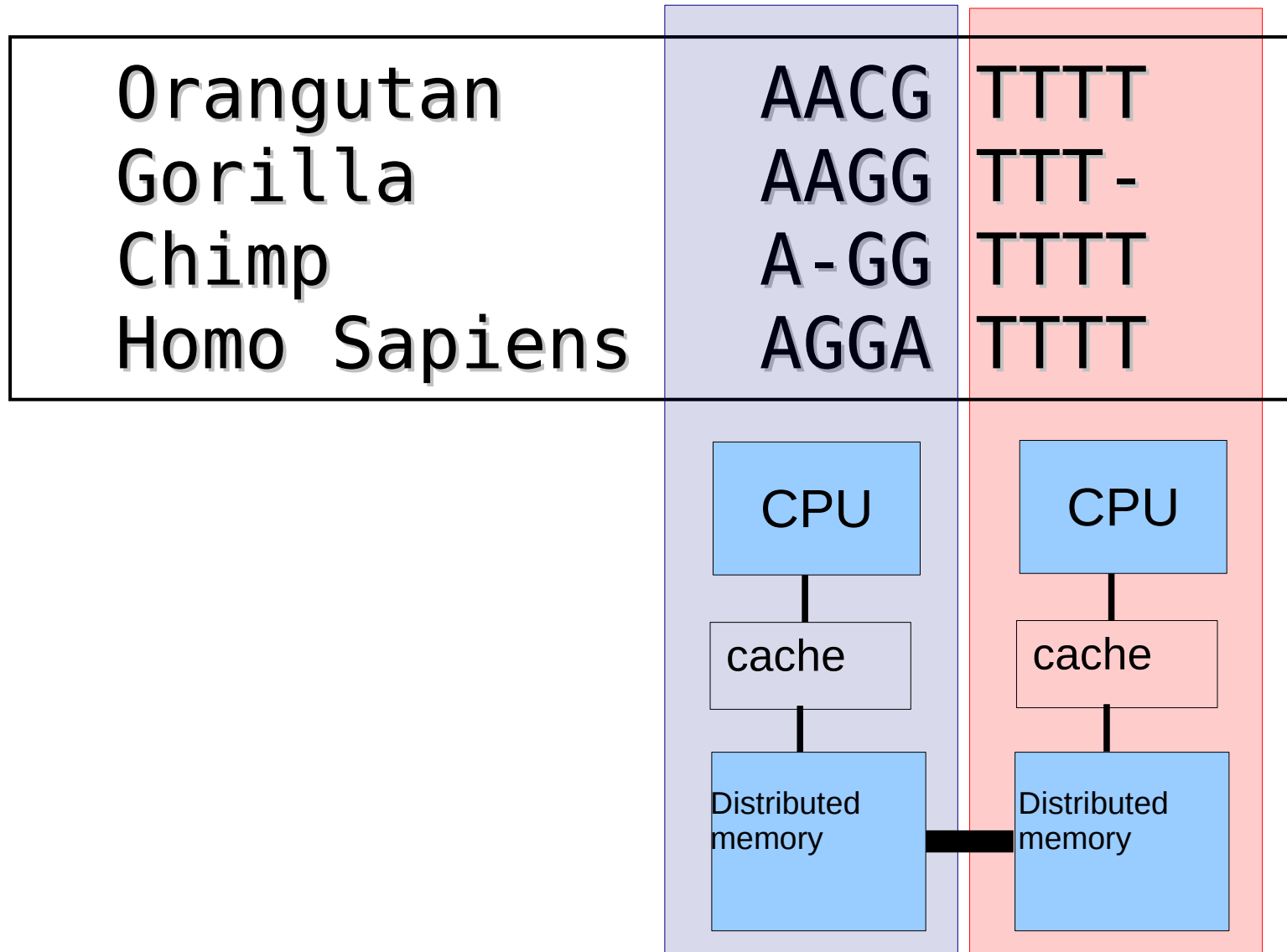
Sequence 1



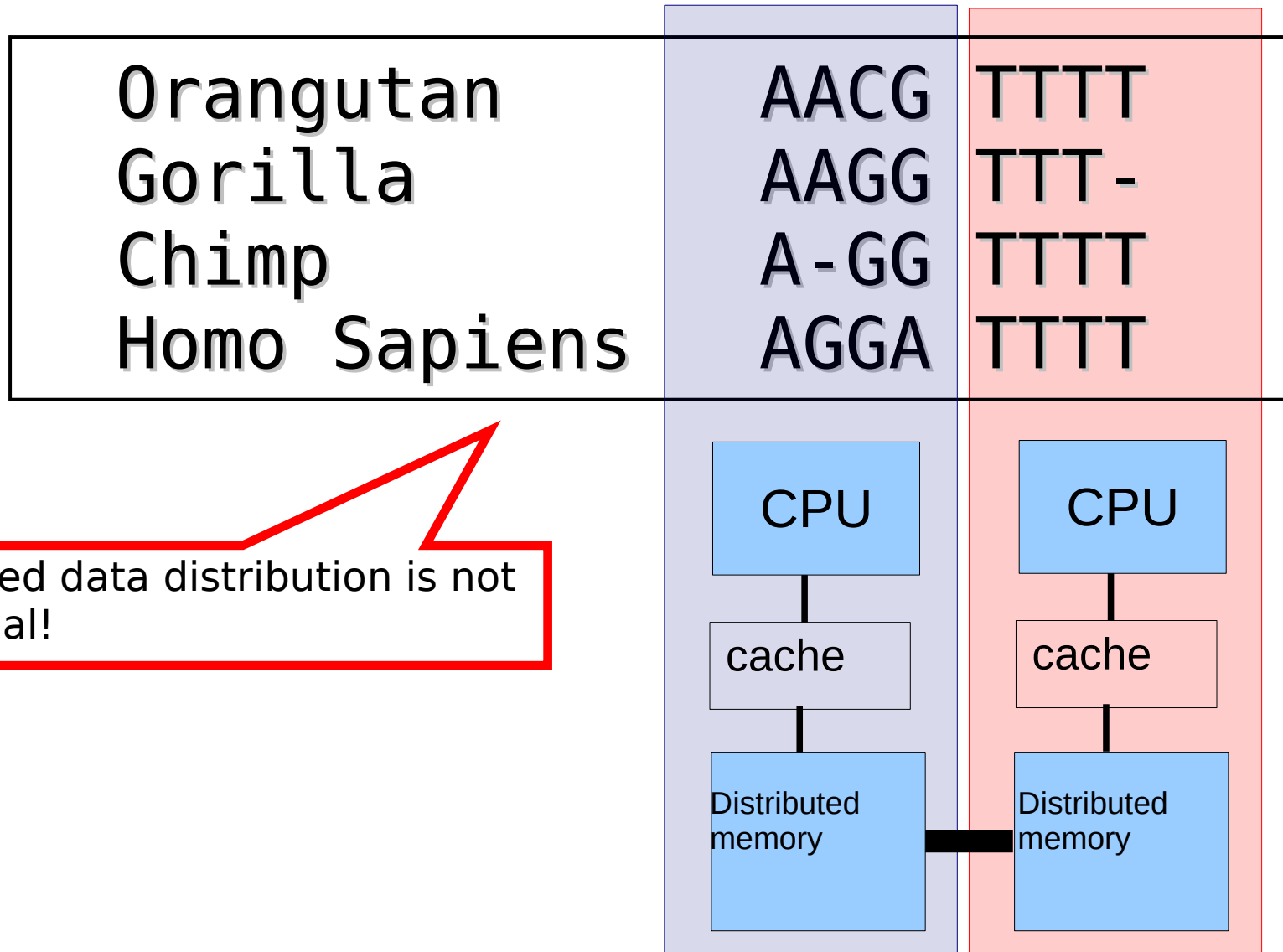
Data Distribution



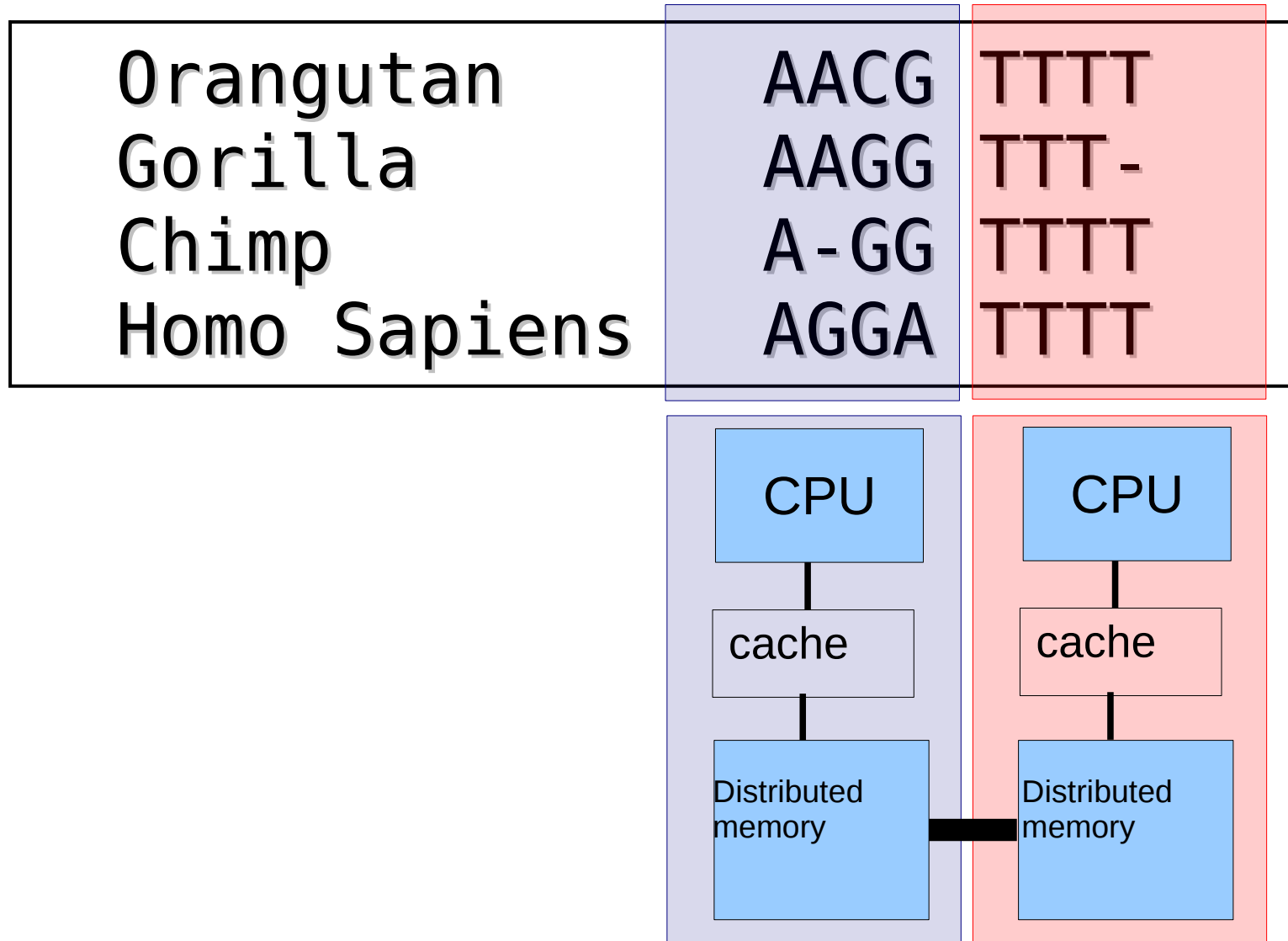
Data Distribution



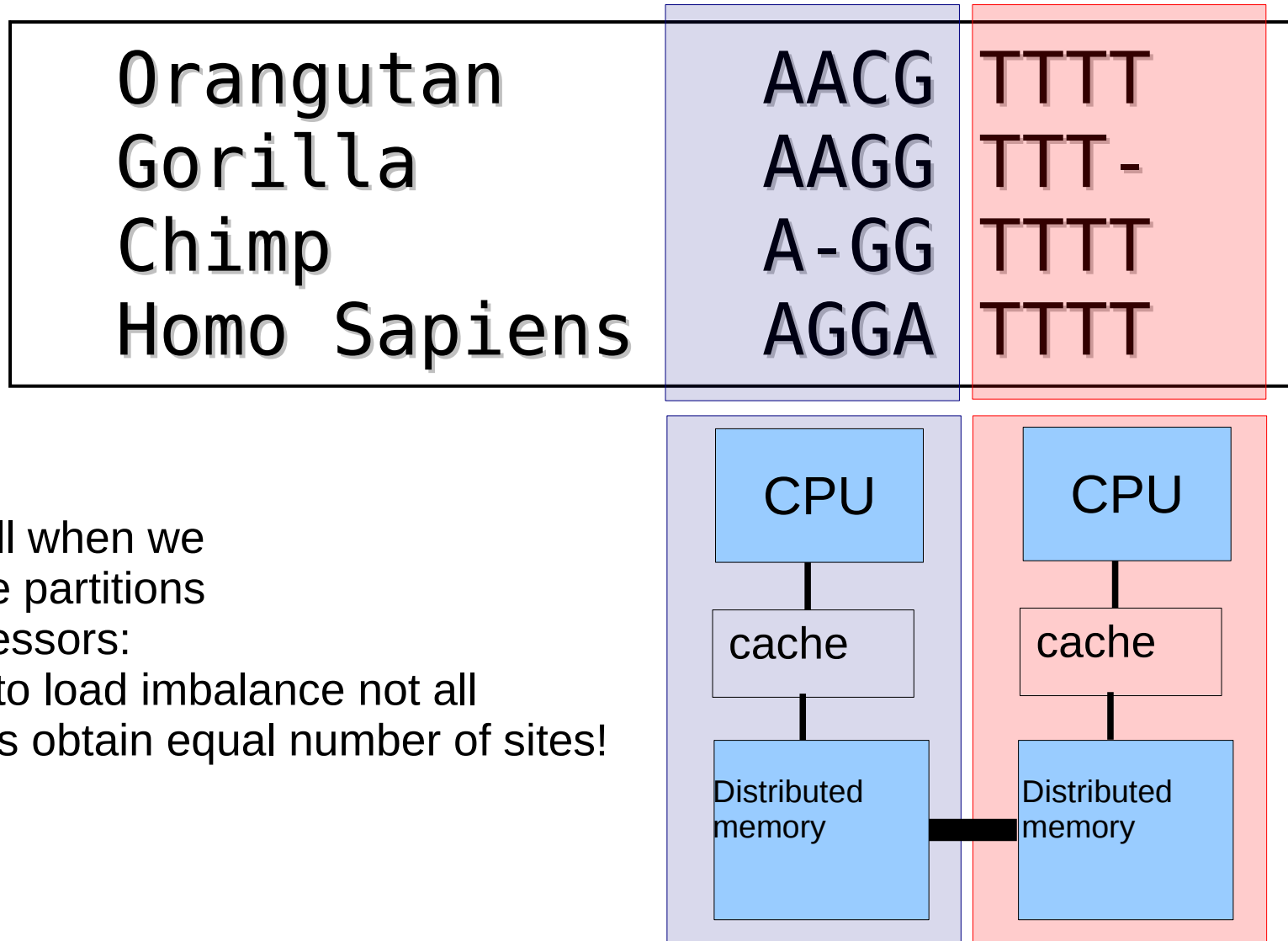
Data Distribution



Data Distribution I



Data Distribution I



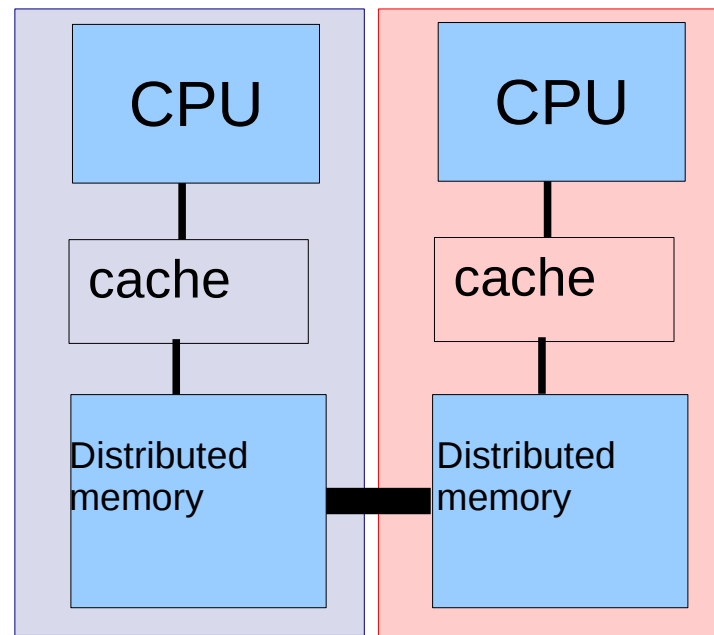
Works well when we have more partitions than processors:
May lead to load imbalance not all processors obtain equal number of sites!

Data Distribution II

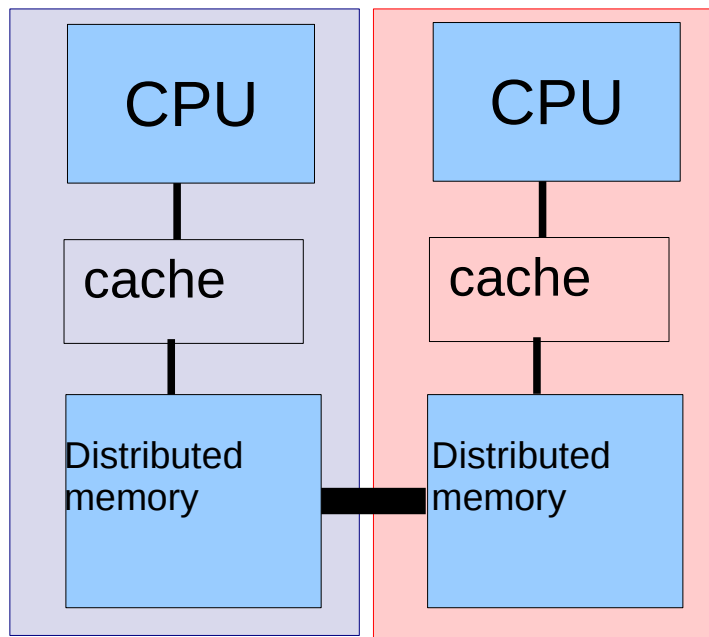
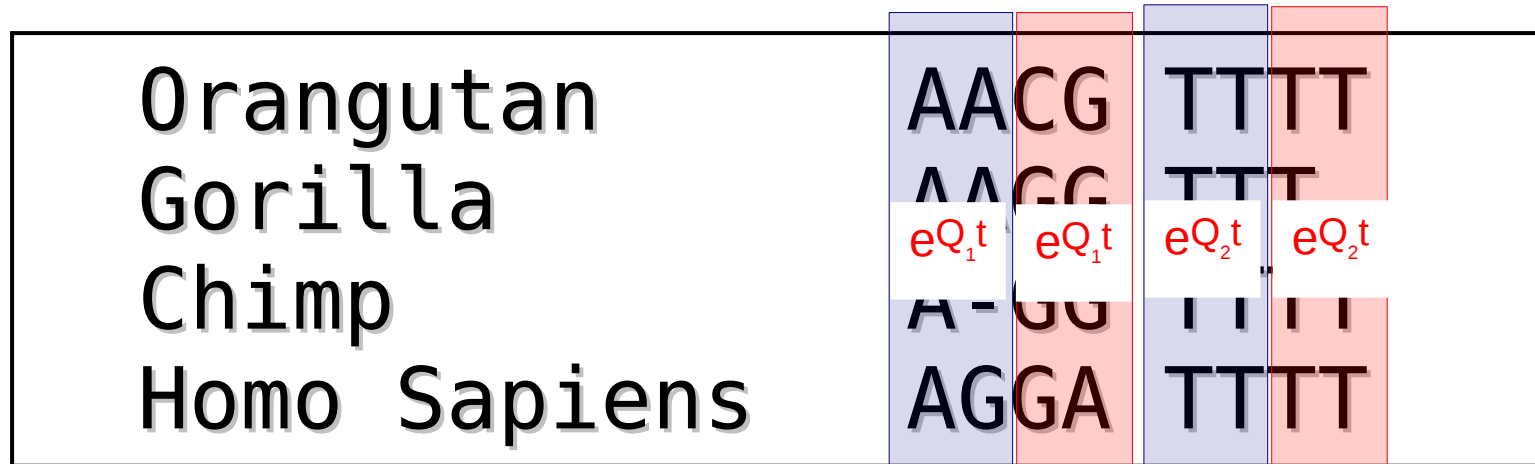
| | | |
|--------------|------|------|
| Orangutan | AACG | TTTT |
| Gorilla | AAGG | TTT- |
| Chimp | A-GG | TTTT |
| Homo Sapiens | AGGA | TTTT |

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!



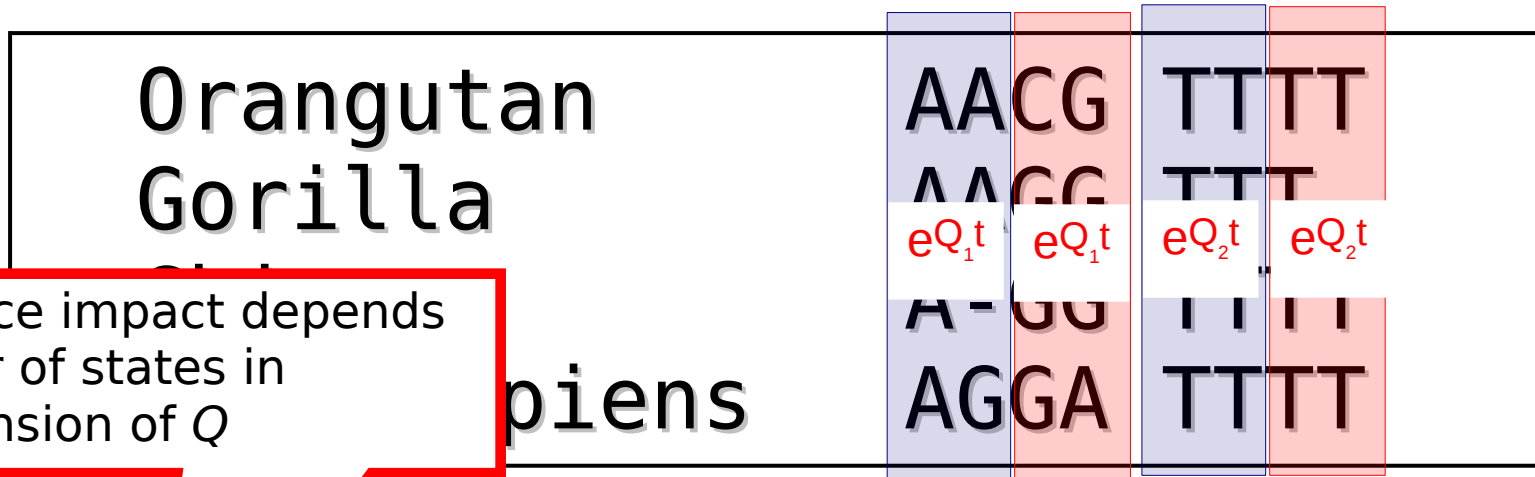
Data Distribution II



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!

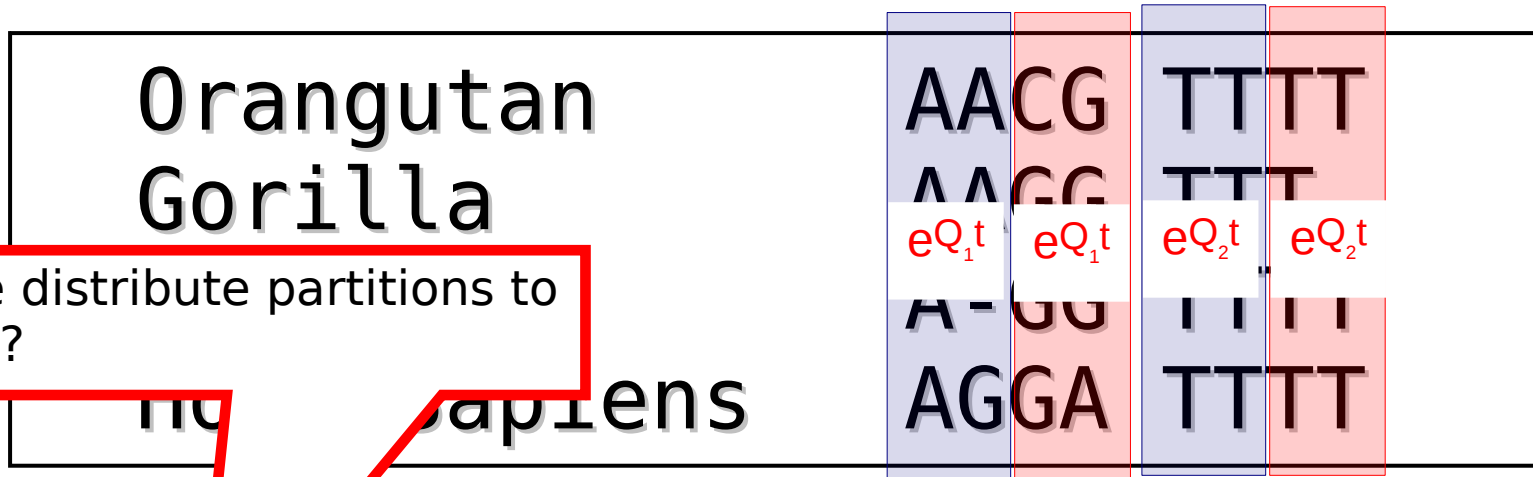
Data Distribution II



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

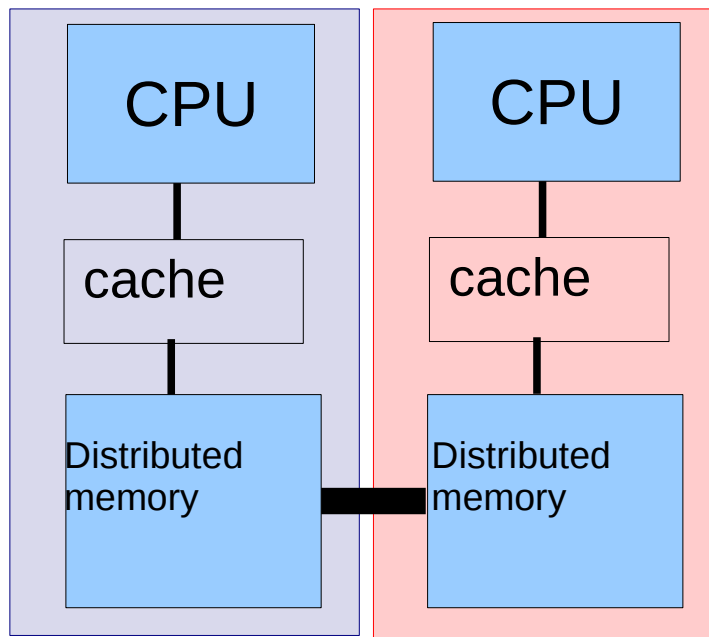
Works well when we have more processors than partitions:
However we will need to compute: $P(t) = e^{Qt}$ for each partition at each processor!

Data Distribution II

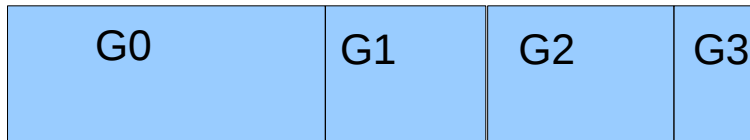


How do we distribute partitions to processors?

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!



Load Balance I



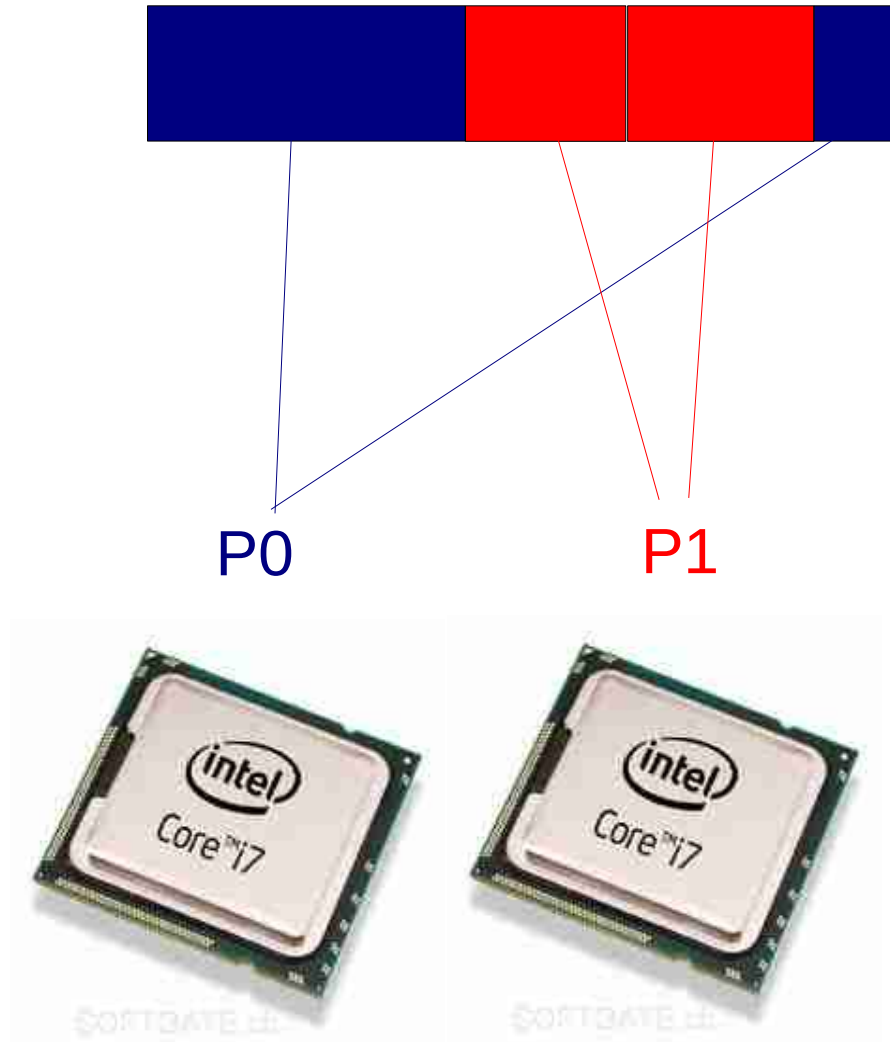
P0



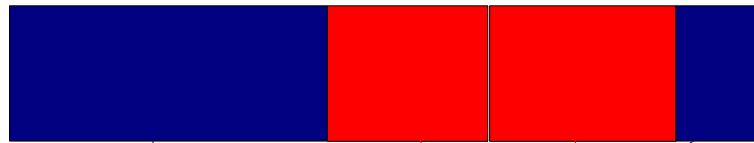
P1



Load Balance I



Load Balance I



P0

P1

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



Load Balance I

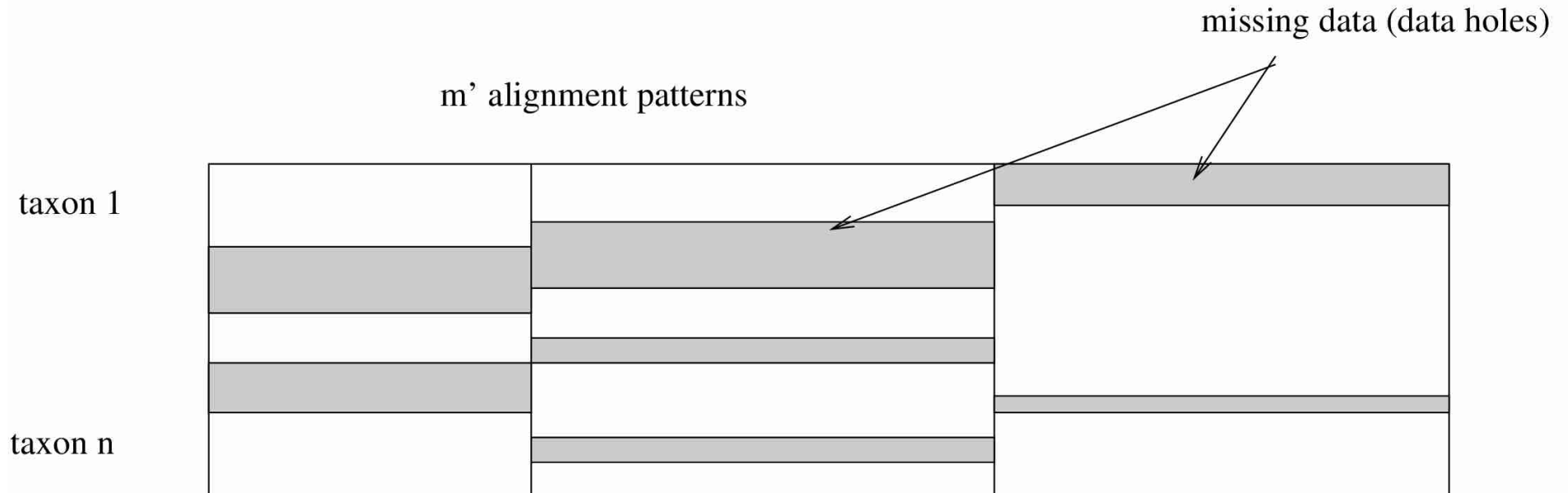
- The **multiprocessor job scheduling problem** in phylogenetics
 - Problem when #partitions \gg #cores
 - Tested per-site (cyclic/modulo) data distribution versus per partition data distribution
 - We used the Longest Processing Time (LPT) heuristics for assigning partitions to processors
 - 25 taxa, 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 Γ model of rate heterogeneity: 10-fold performance improvement!

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

LPT heuristics for multi-processor scheduling

- Sort jobs (partitions) by processing length (partition length) in decreasing order
- Remove a job (partition) from the sorted list and assign it to the processor with the earliest end time (the smallest sum of partition lengths)
- Repeat until the sorted list is empty
- Upper bound: $4/3 - 1/(3p) * OPT$, where p is the number of processors
- Graham, R. L.: "Bounds on Multiprocessing Timing Anomalies". *SIAM Journal on Applied Mathematics* 17 (2): 416–429, 1969.
- Remark: LPT works surprisingly well (see our paper on the phylogenetic problem where we also tested other heuristics)

Partitioned Branch Lengths & other parameters

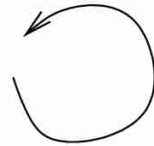


partition 0



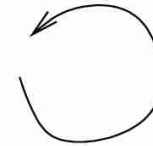
separate estimate of
Q-Matrix
alpha-shape
Branch Lengths

partition 1



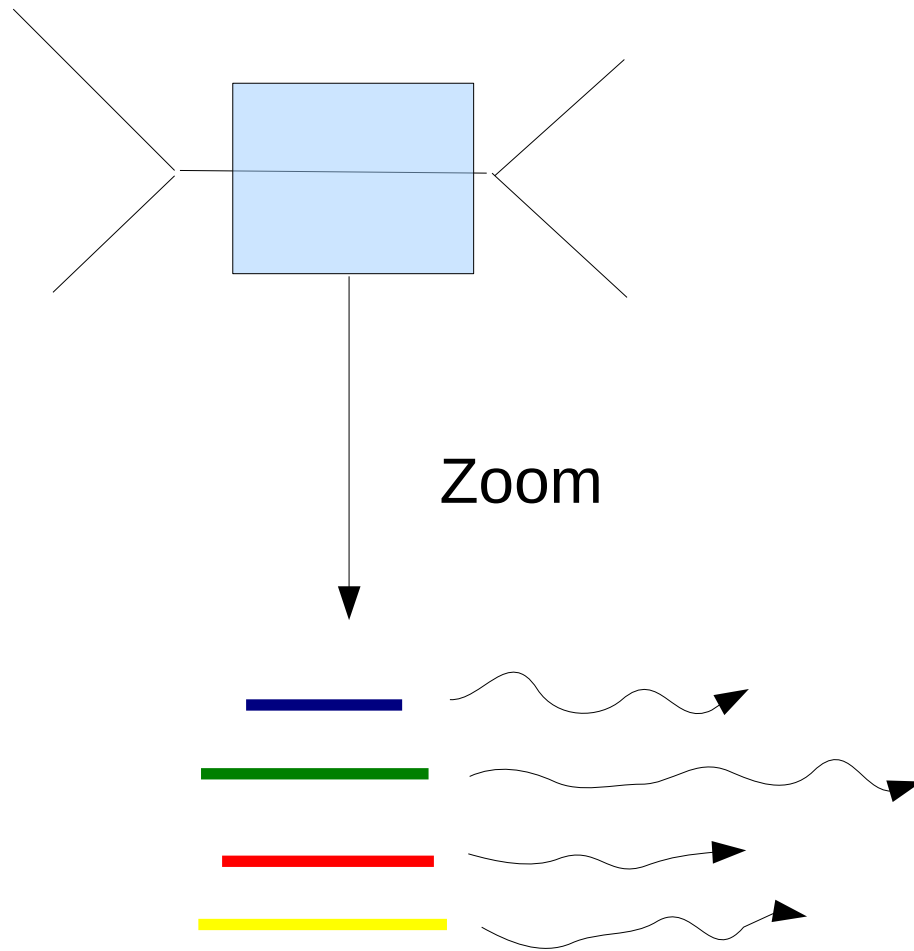
separate estimate of
Q-Matrix
alpha-shape
Branch Lengths

partition 2



separate estimate of
Q-Matrix
alpha-shape
Branch Lengths

Load-Balance II



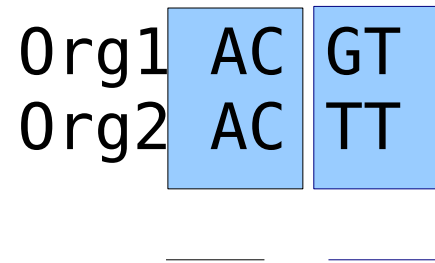
Synchronization Points

- Assume 10 branches
- Each branch requires 10 Newton-Raphson Iterations
- Each NR Iteration requires a synchronization 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

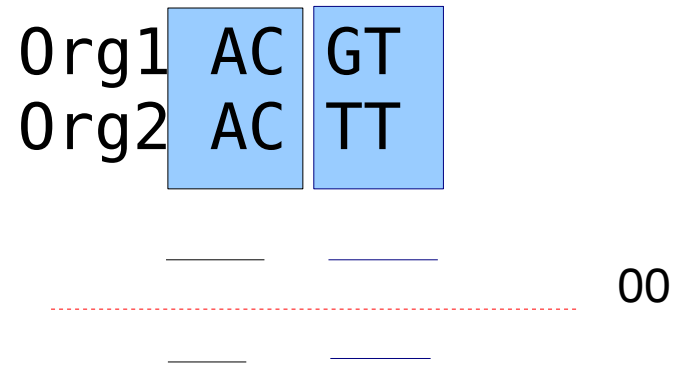
Synchronization Points

Org1 AC GT
Org2 AC TT

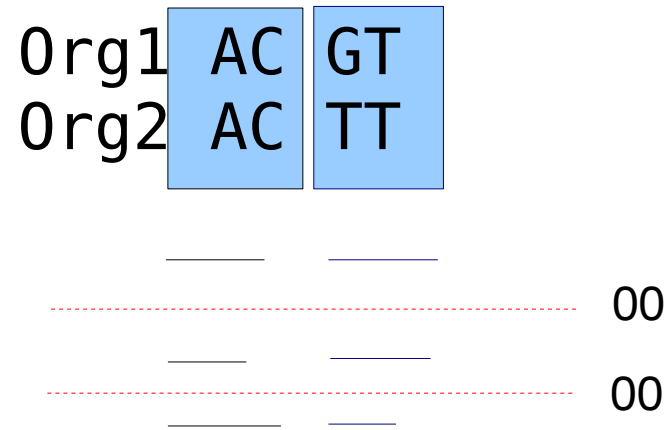
Synchronization Points



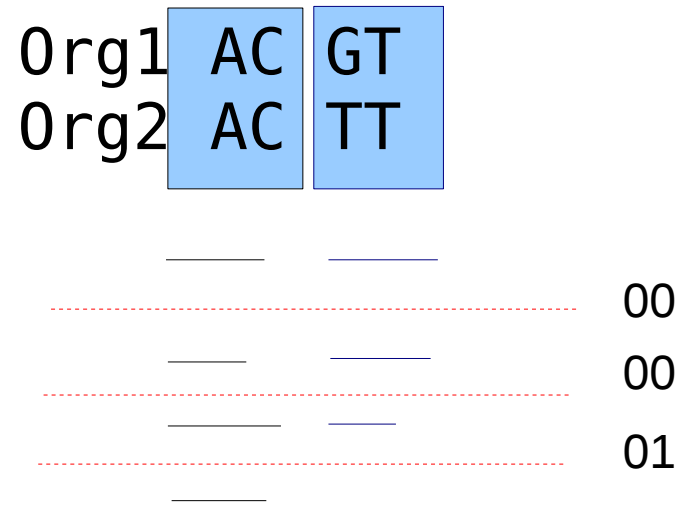
Synchronization Points



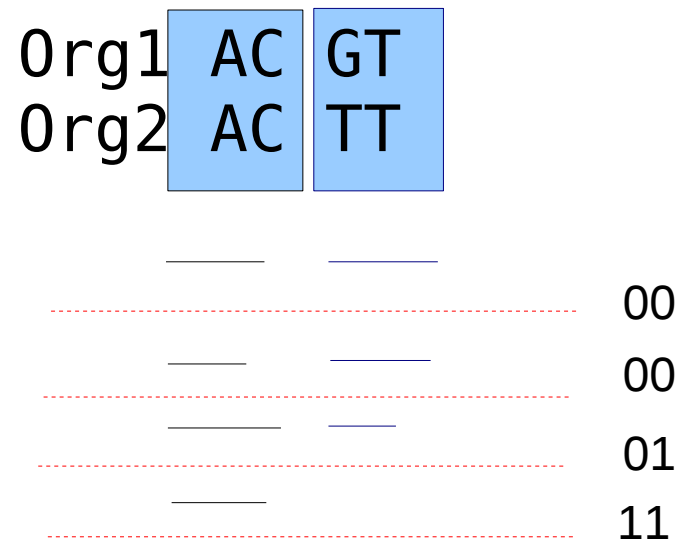
Synchronization Points



Synchronization Points

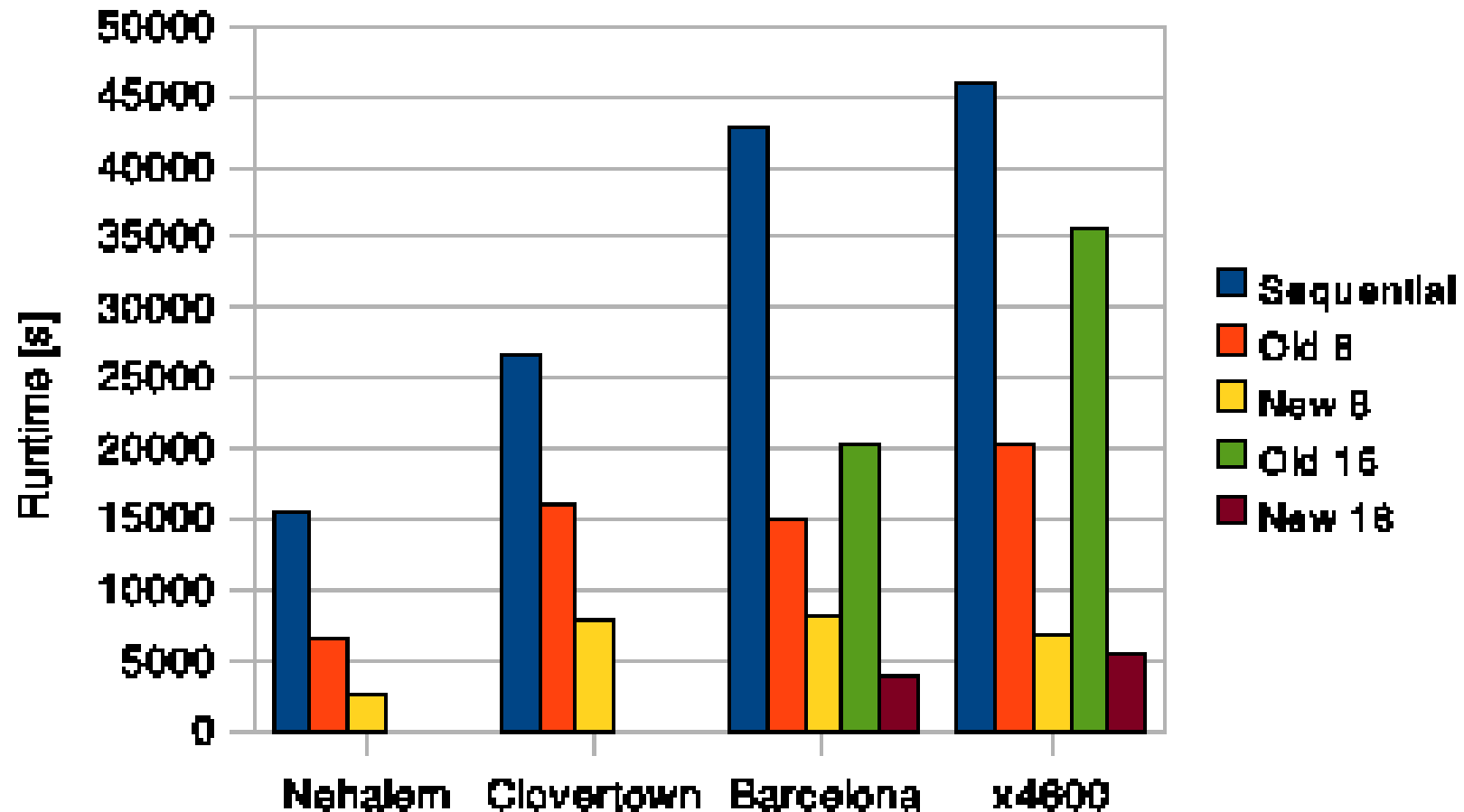


Synchronization Points



In this example: 4 instead of 7 sync points!

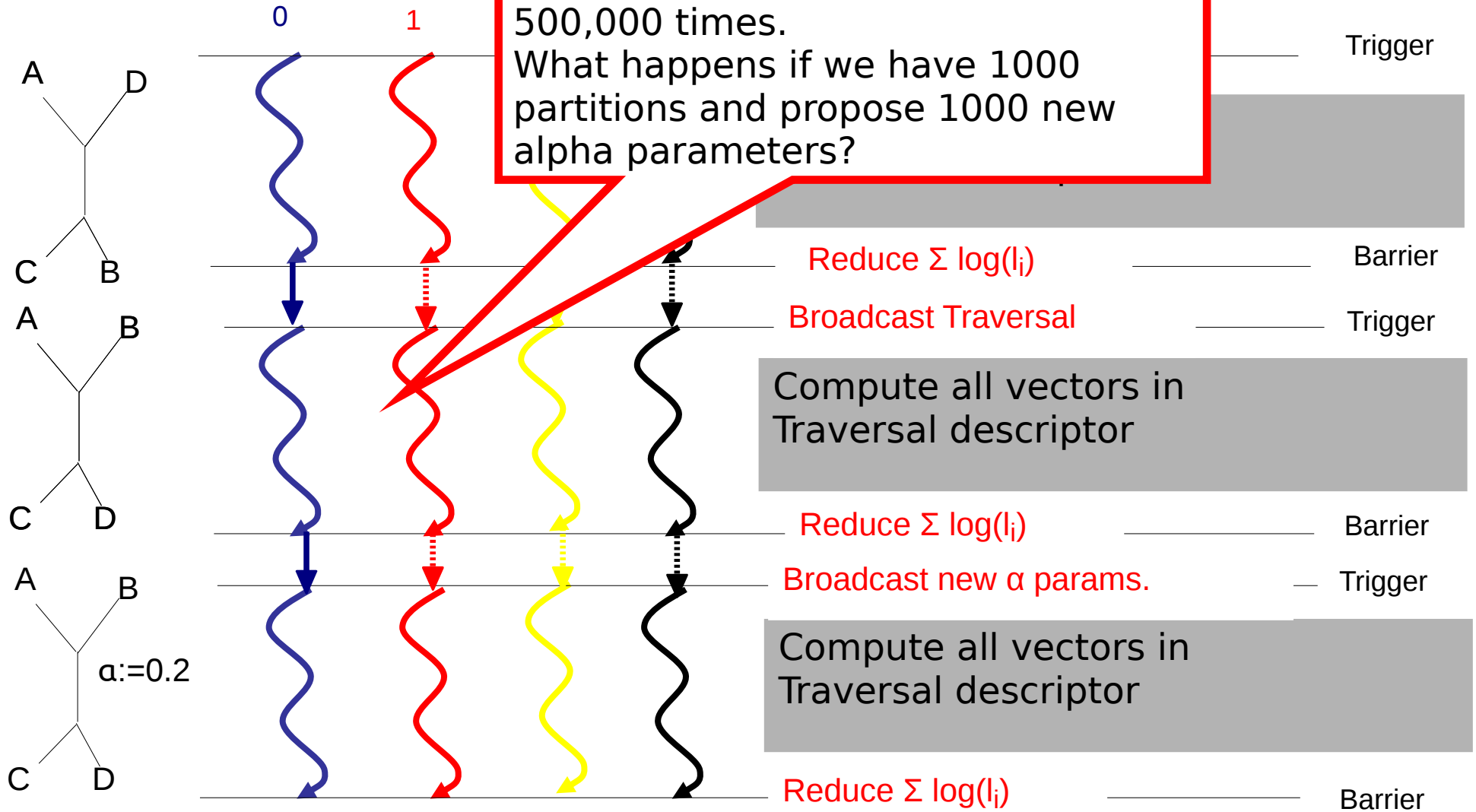
Load Balance II



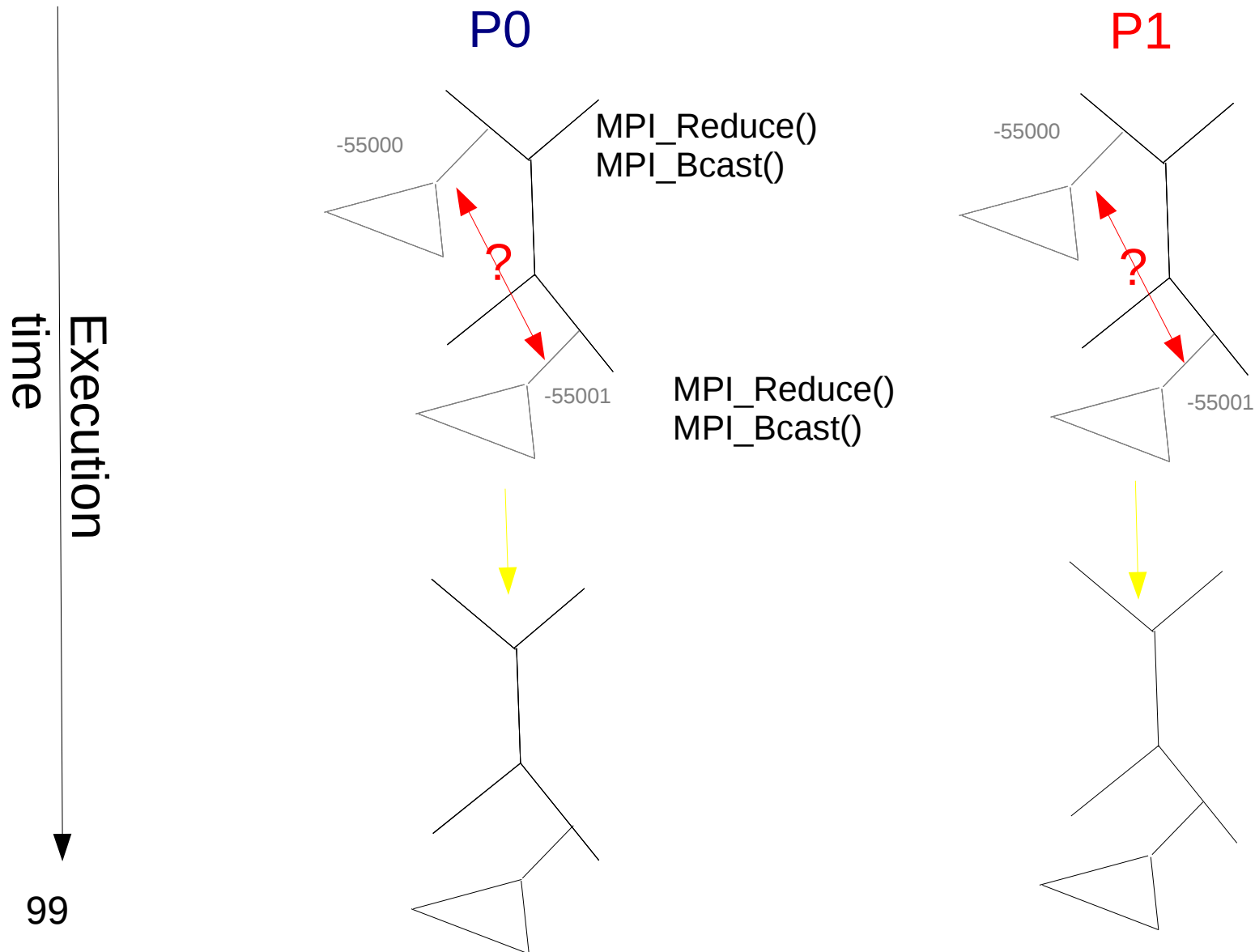
A. Stamatakis, M. Ott: "Load Balance in the Phylogenetic Likelihood Kernel".
Proceedings of ICPP 2009, Vienna, Austria, September 2009.

Classic Fork-Join with

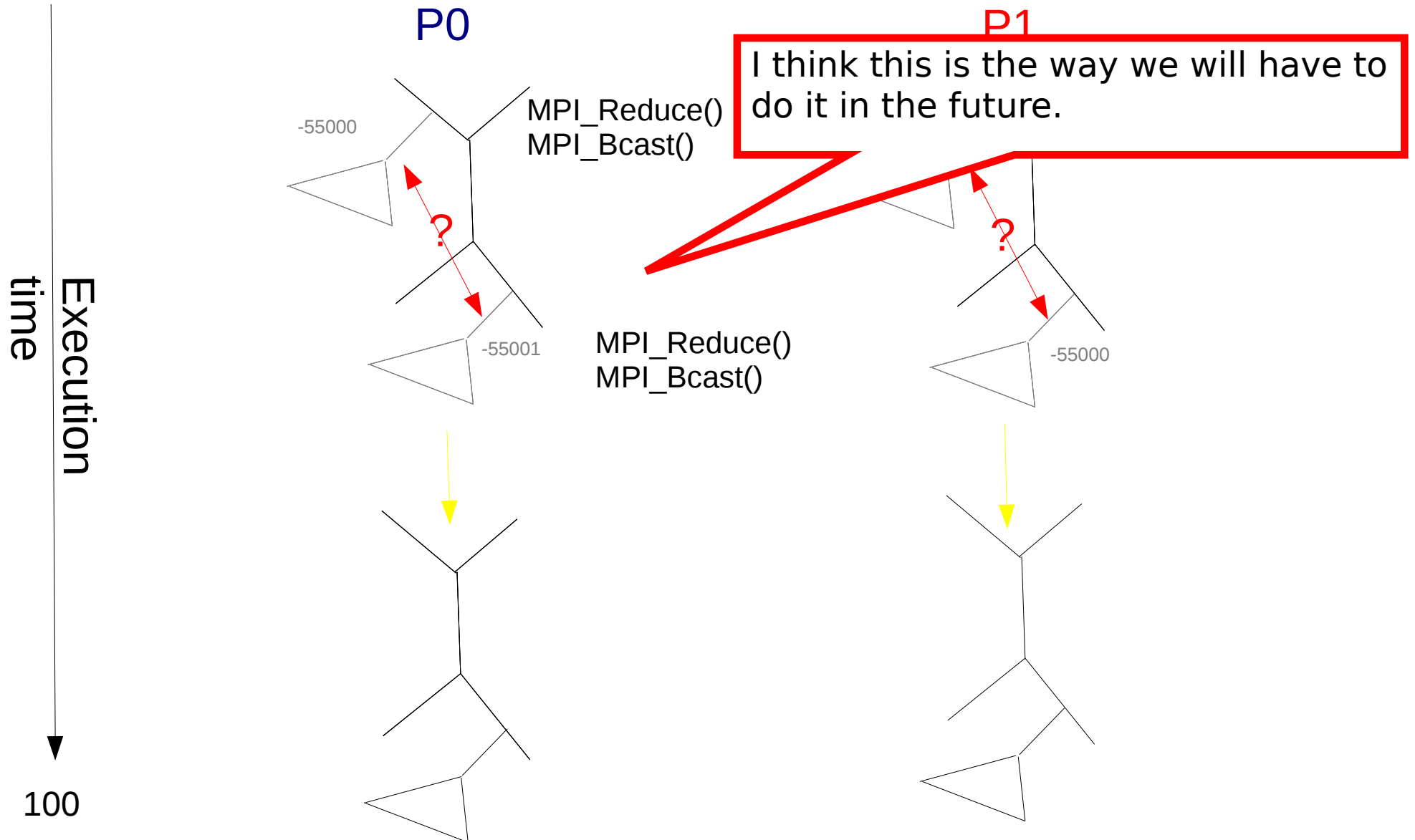
For good parallel performance: the broadcast must be fast!
 Remember: 10 secs 16 cores approx 500,000 times.
 What happens if we have 1000 partitions and propose 1000 new alpha parameters?



Alternative MPI parallelization



Alternative MPI parallelization



ExaML

- New code implementing this new parallelization scheme
- <https://github.com/stamatak/ExaML>
- A. Stamatakis, A. J. Aberer: "Novel Parallelization Schemes for Large-Scale Likelihood-based Phylogenetic Inference", accepted for publication at *IPDPS 2013*, Boston, USA, 2013.
- Up to 3 times faster than RAxML-Light (2012) on large, partitioned datasets
- Tested with up to 1536 cores on our cluster at HITS
- Future developments
 - 20,000,000 CPU hours on SuperMUC for
 - Improving scalability
 - Implementing fault tolerance
 - Execute 1KITE tree inferences
 - Further details → ask Andre Aberer