Whole genome alignment in High Performance Computing environments

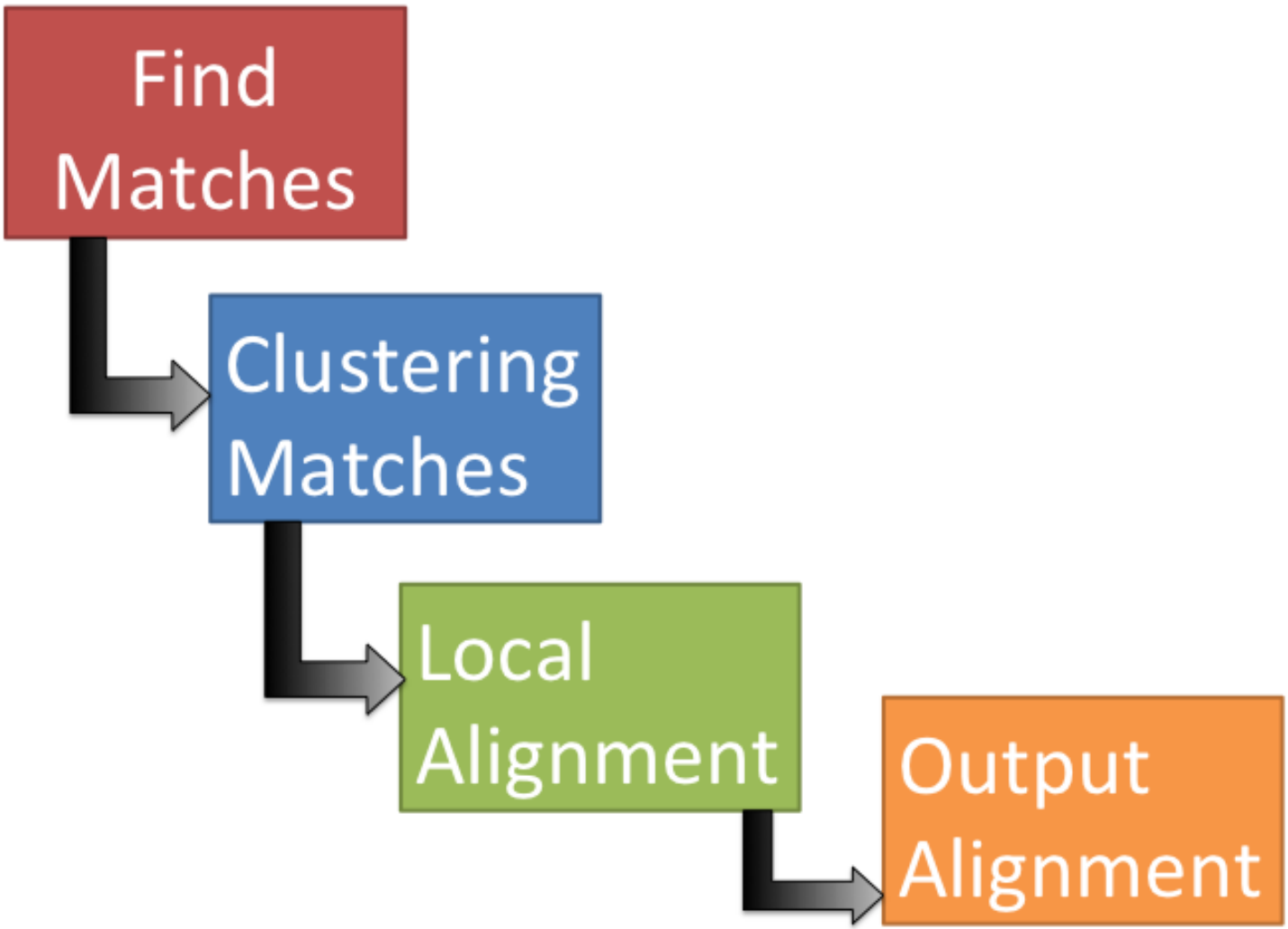
Julio César García Vizcaíno Directores: Antonio Espinosa, Juan Carlos Moure

Computer Architecture & Operating Systems Department
Universitat Autónoma de Barcelona

17 de septiembre de 2012

Contents

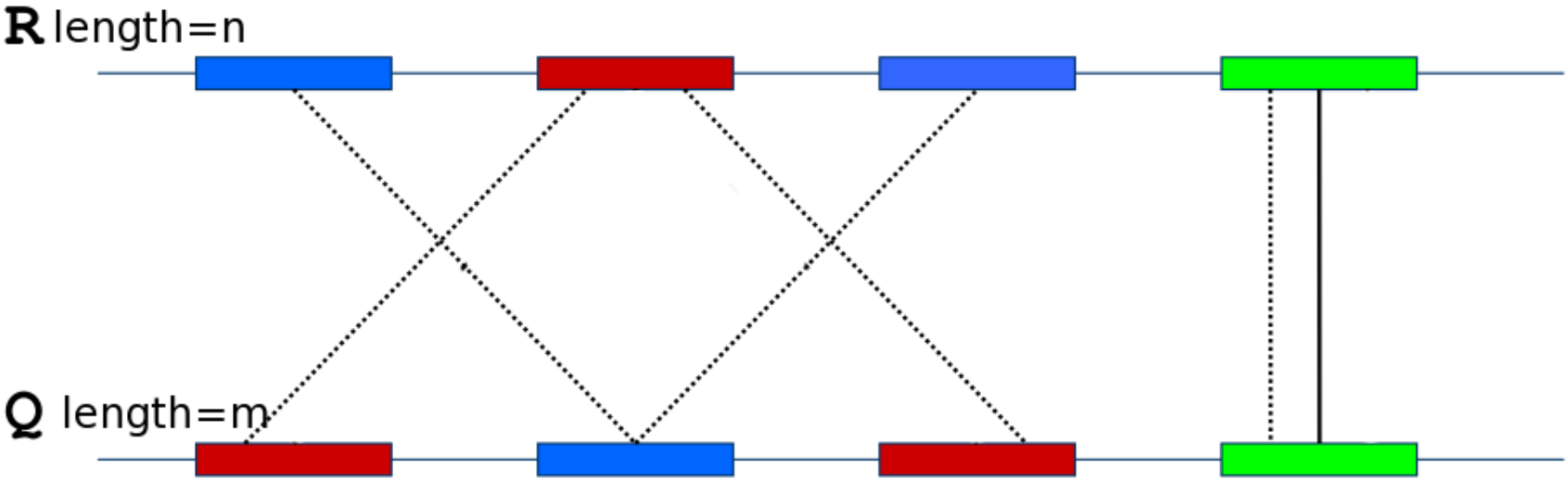
Whole Genome Alignment in MUMmer



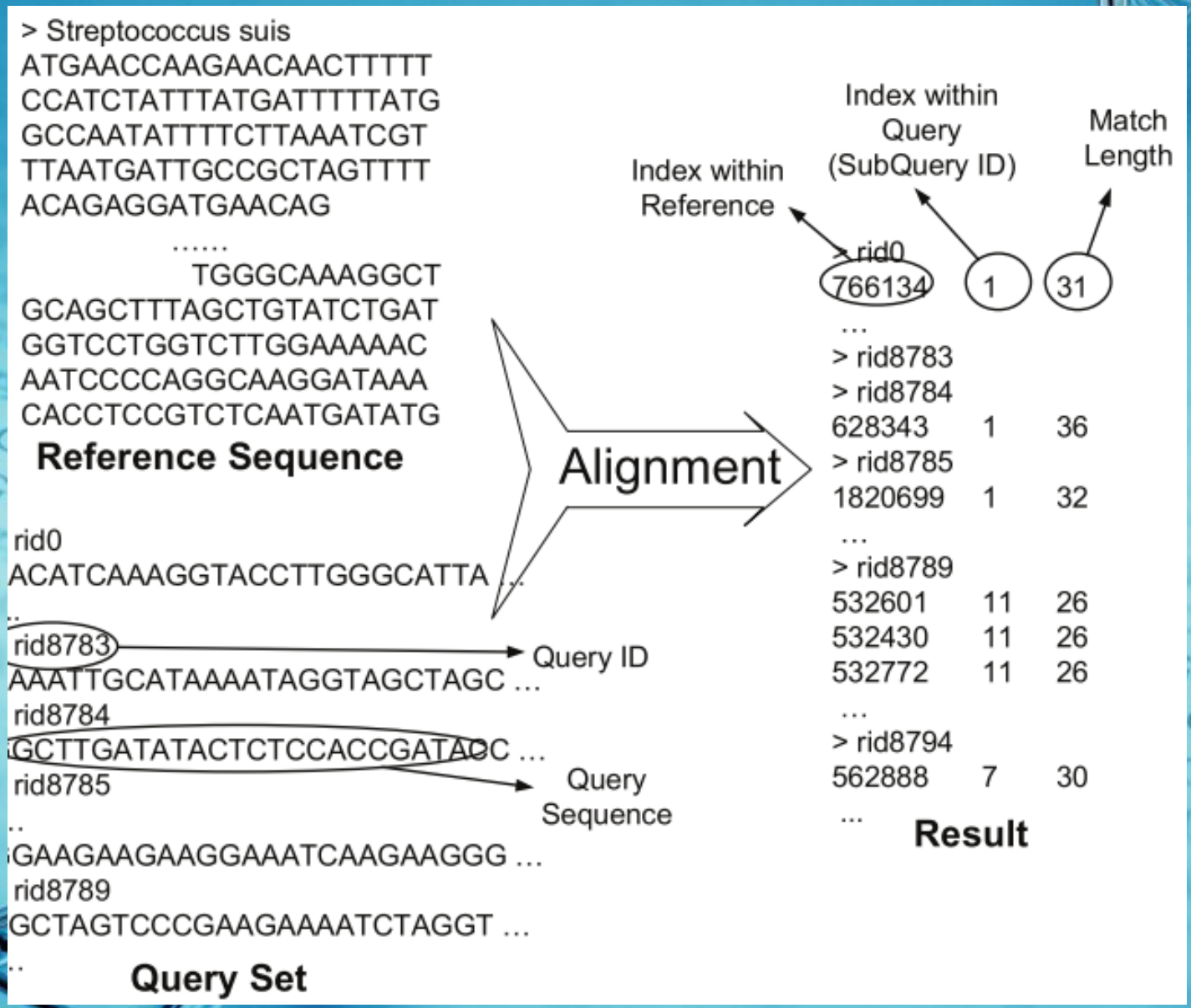
Search of Maximal Exact Matches

MUM: Maximal Unique Match

MEM: Maximal Exact Match



Genome alignment: search of Maximal Exact Matches



Ways of finding exact matches

Brute Force (3 GB)

```
BANANA
BAN
  ANA
    NAN
      ANA
```

O(nm)

Naive

Slow & Easy

Suffix Array (>15 GB)

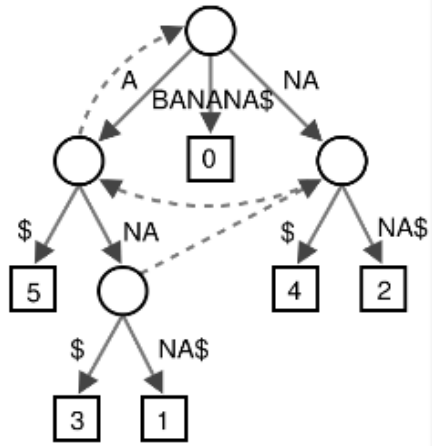
6	$
5	A$
3	ANA$
1	ANANA$
0	BANANA$
4	NA$
2	NANA$

O(m log n)

Vmatch, PacBio Aligner

Binary Search

Suffix Tree (>51 GB)

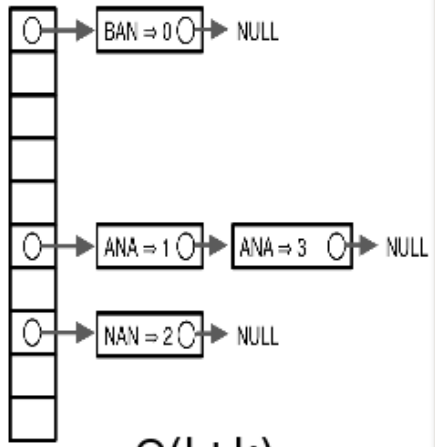


O(m+k)

MUMmer, MUMmerGPU

Tree Searching

Hash Table (>15 GB)

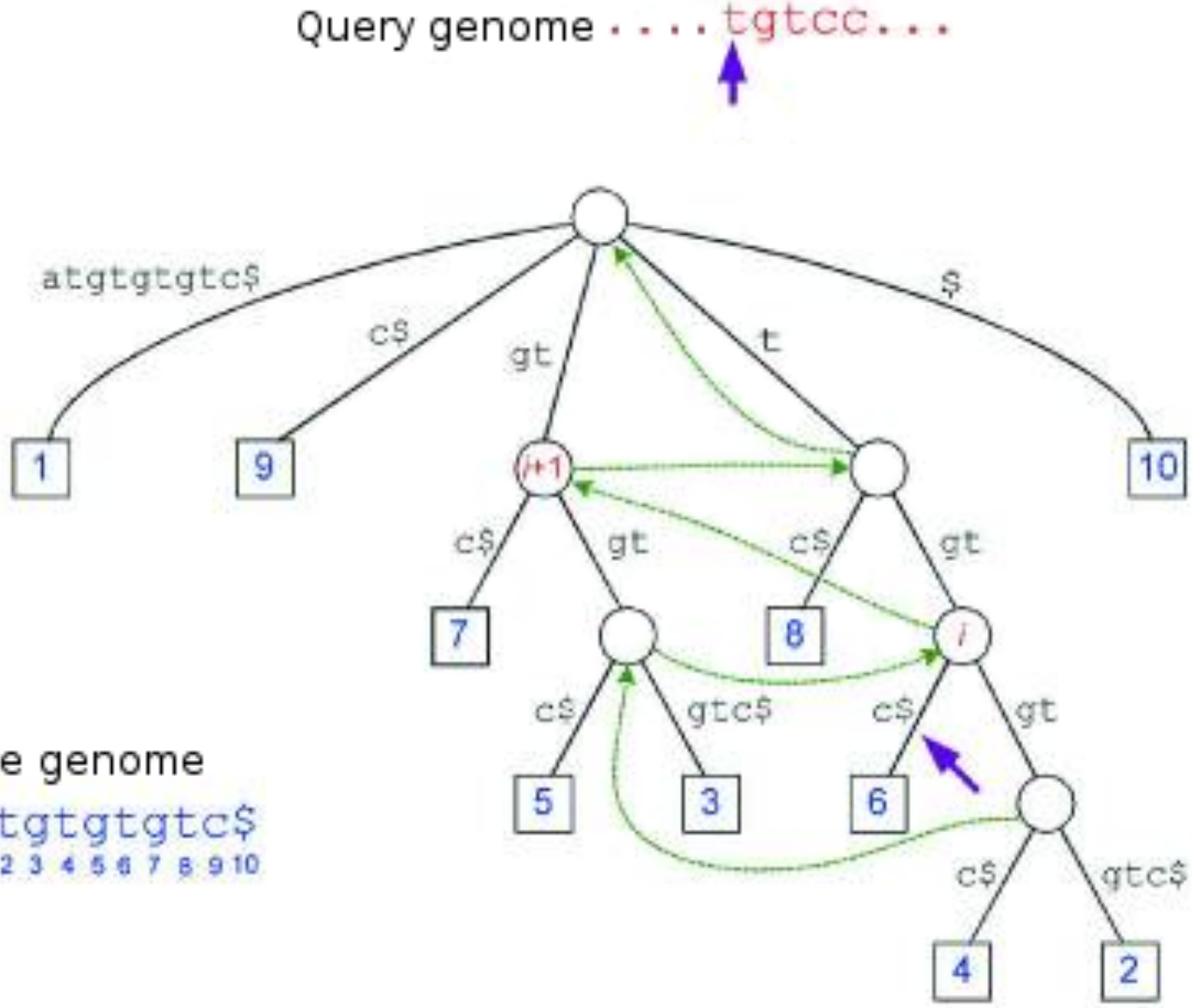


O(l+k)

BLAST, MAQ, ZOOM, RMAP, CloudBurst

Seed-and-extend

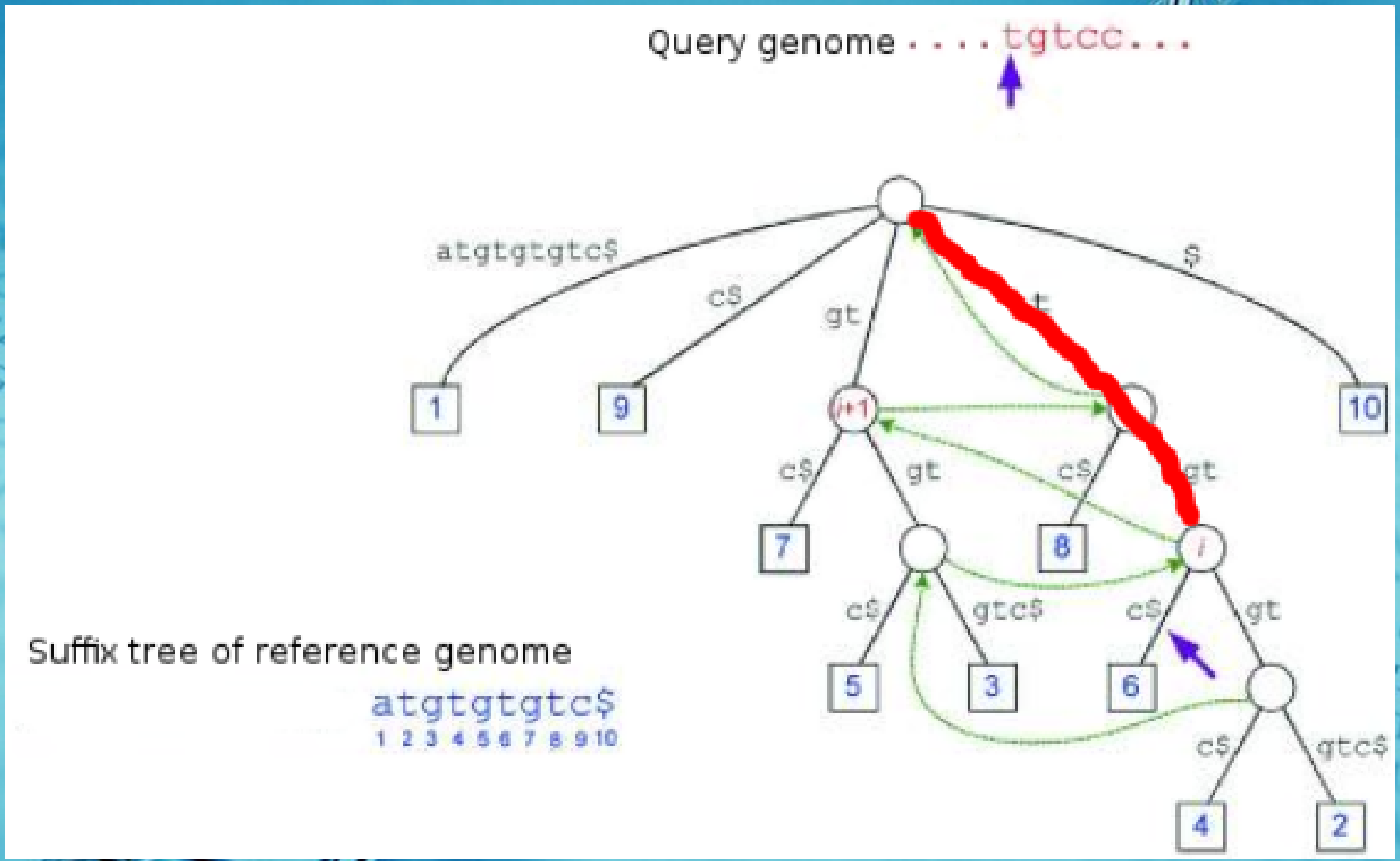
Traversal of suffix tree



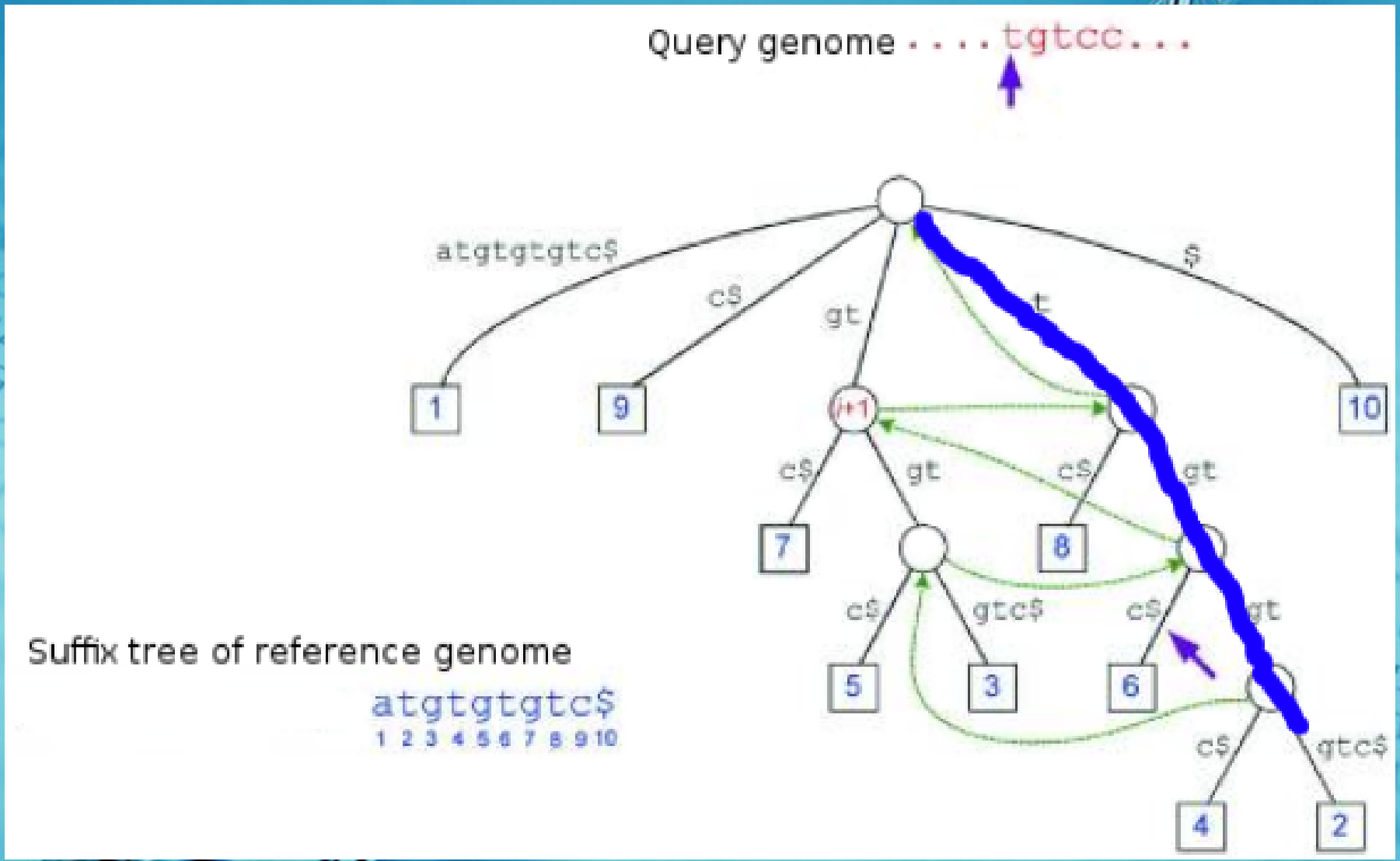
Suffix tree of reference genome

atgtgtgtc$
1 2 3 4 5 6 7 8 9 10

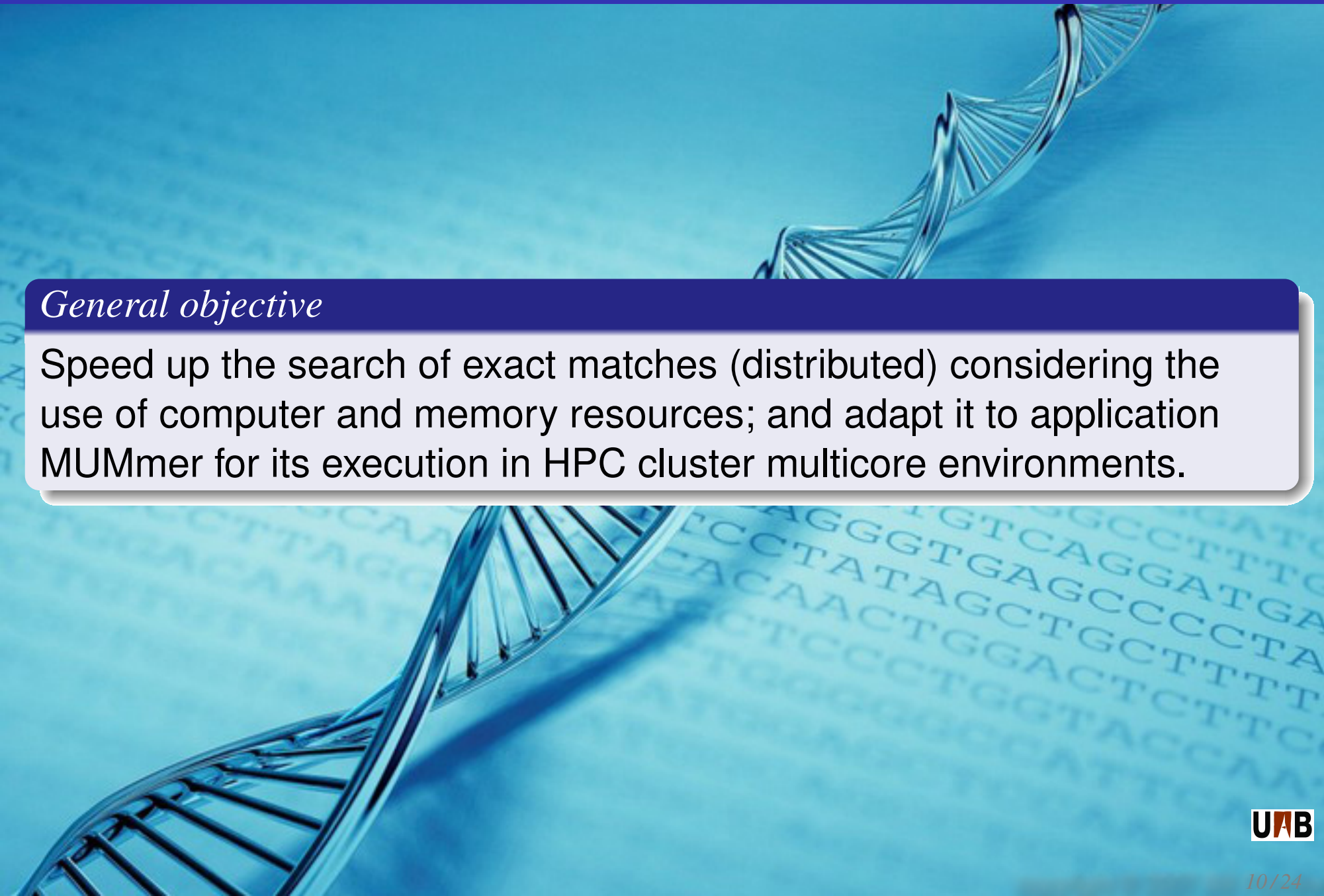
Find MEM in suffix tree



Find MUM in suffix tree



General objective



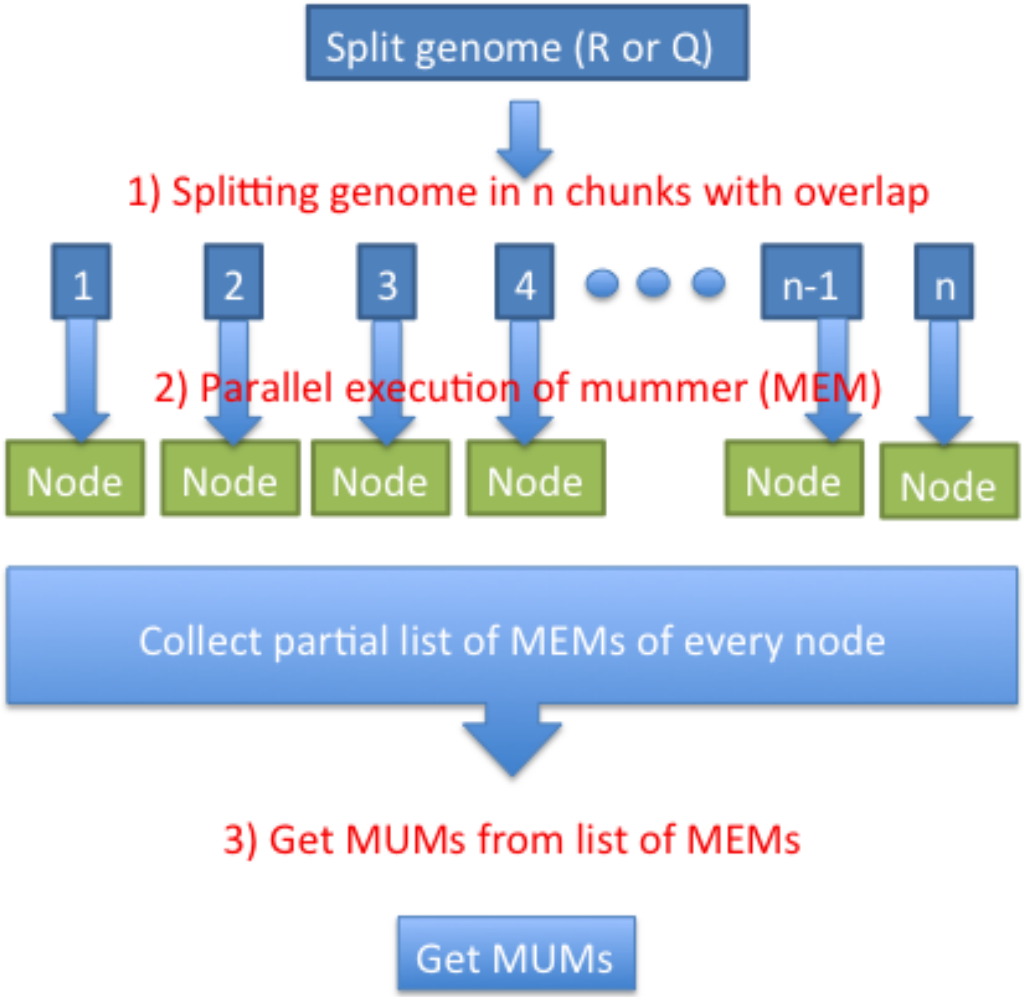
General objective

Speed up the search of exact matches (distributed) considering the use of computer and memory resources; and adapt it to application MUMmer for its execution in HPC cluster multicore environments.

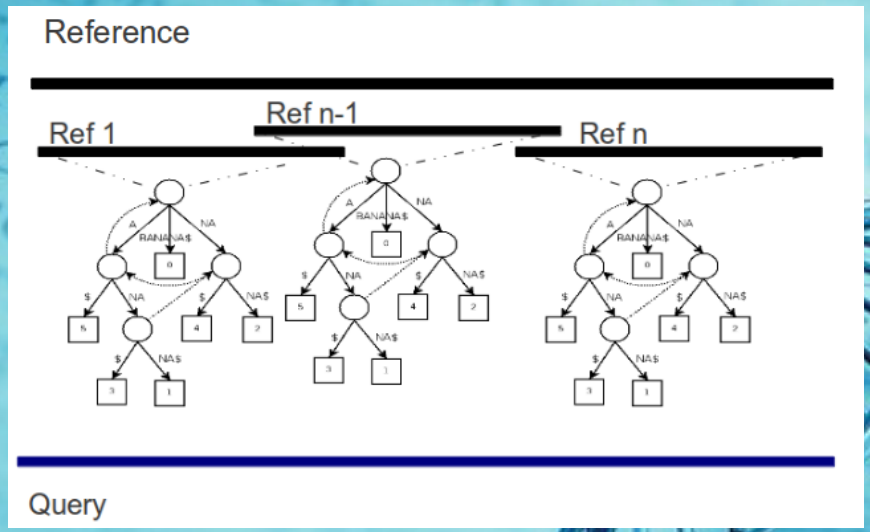
Specific objective

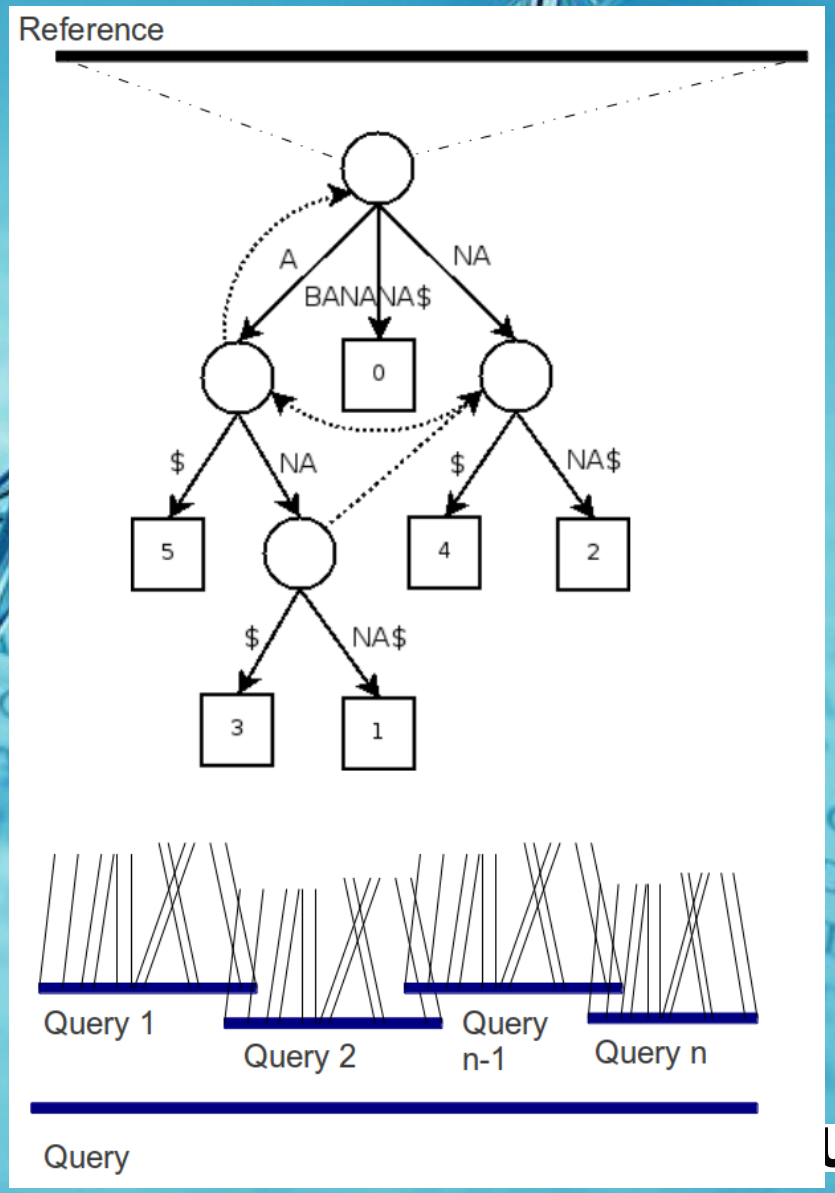
- To have a data structure, efficient usage of memory and processor, that allows a quick search of maximal exact matches.
  - Save relevant information for the search of matches.
  - Be able to nimbly check the data structure.

Naive solution

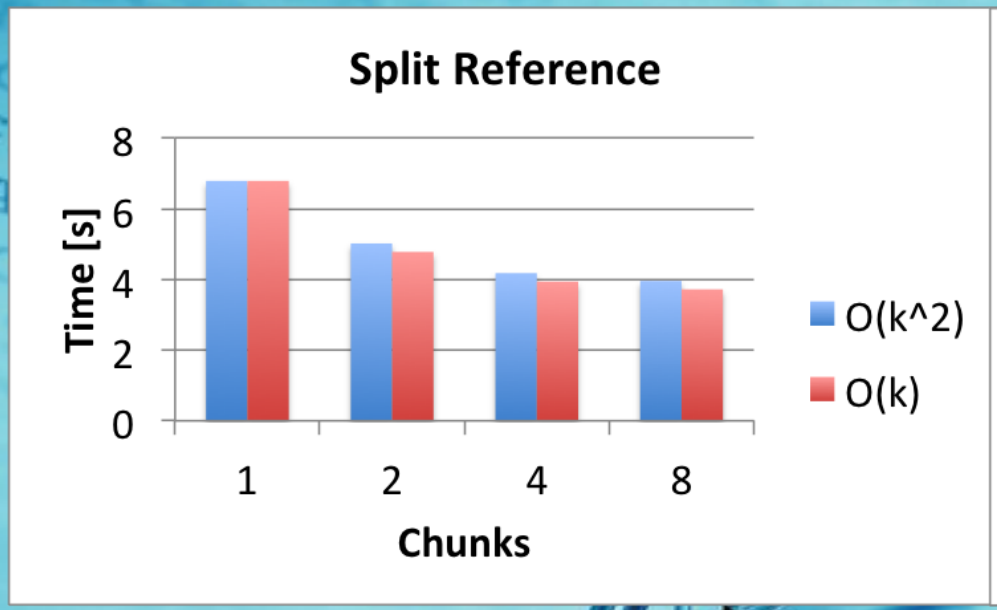


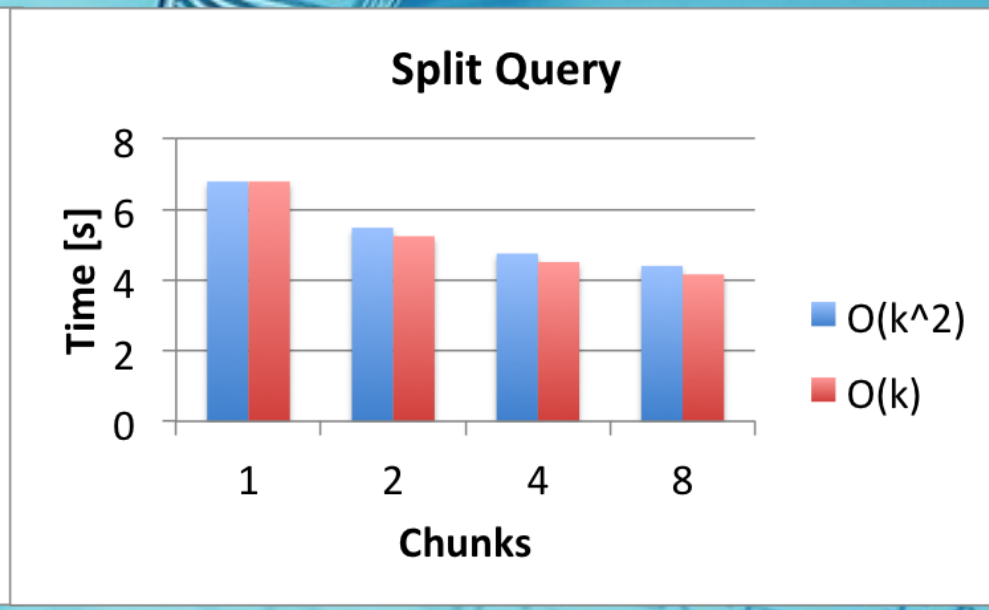
Split sequence





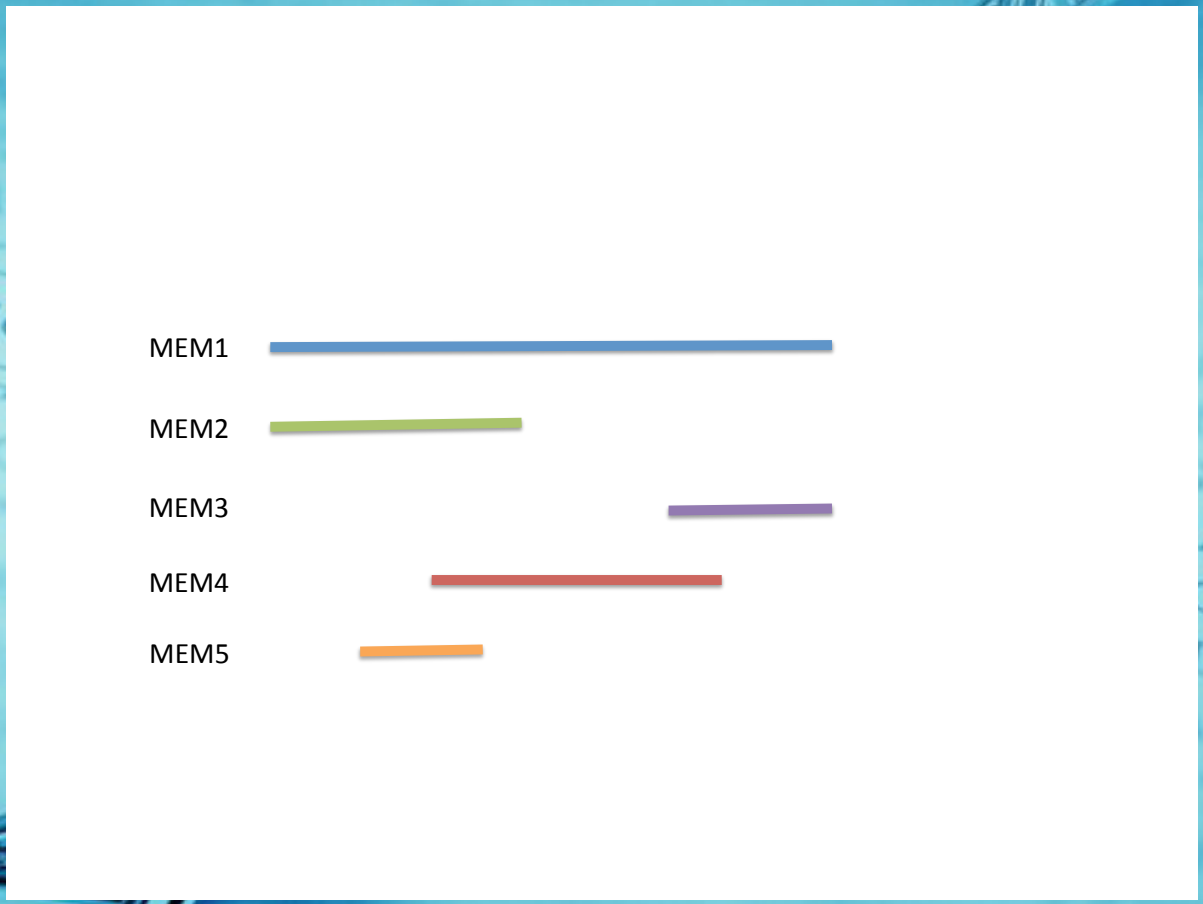
Naive solution cont.





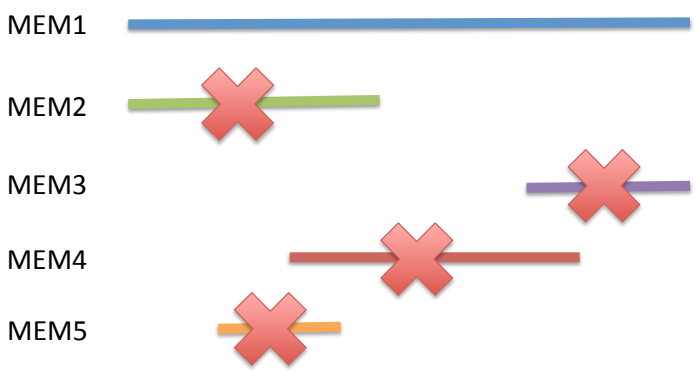
Merge phase algorithm

We require to drop those MEMs that are covered by a bigger MEM.



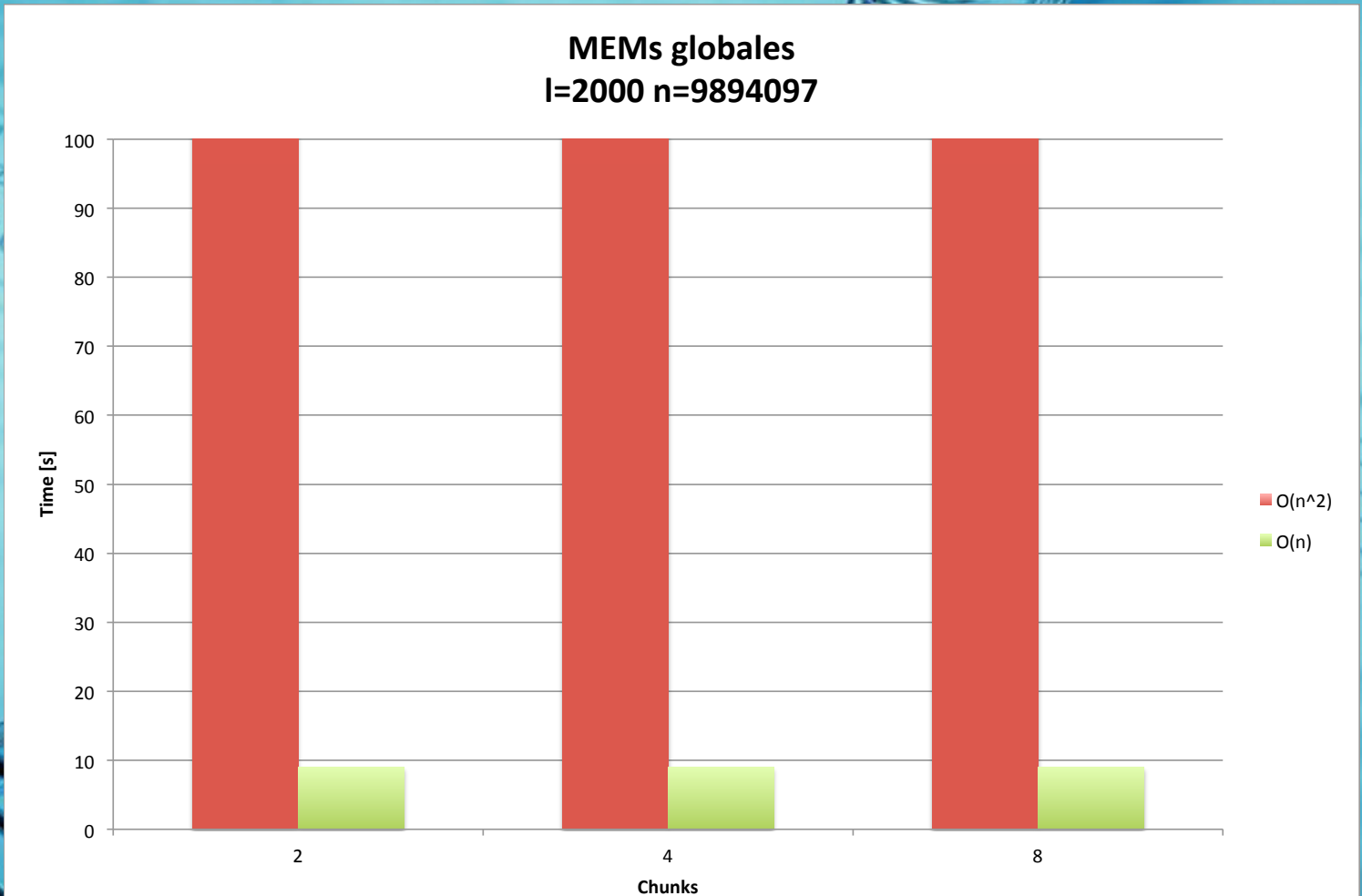
Merge phase algorithm

Drop MEMs that are covered by the MEM of reference.



Results of Merge phase

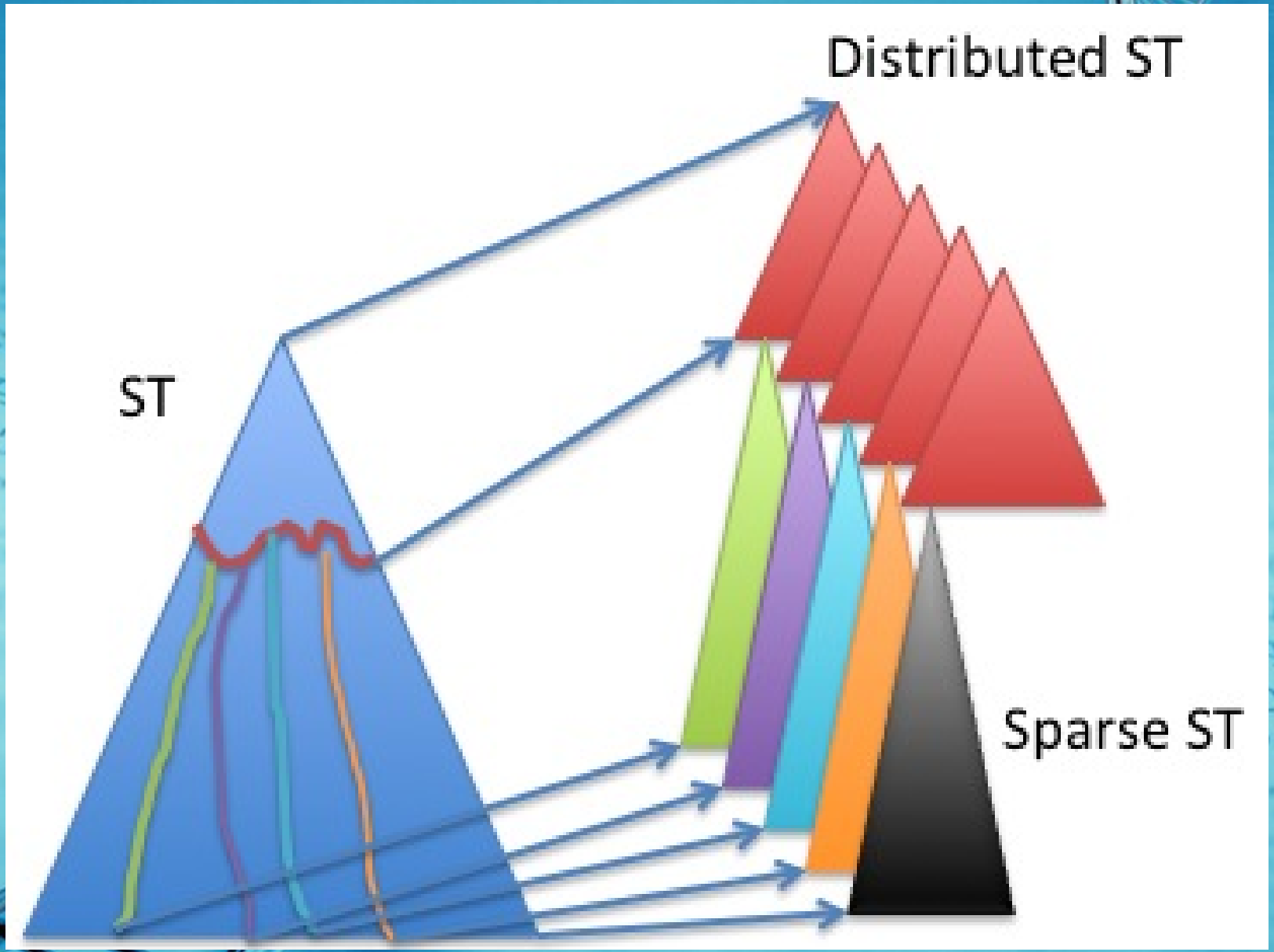
Merge of MEMs for chromosome 19 of homo sapiens and chimpanzee.



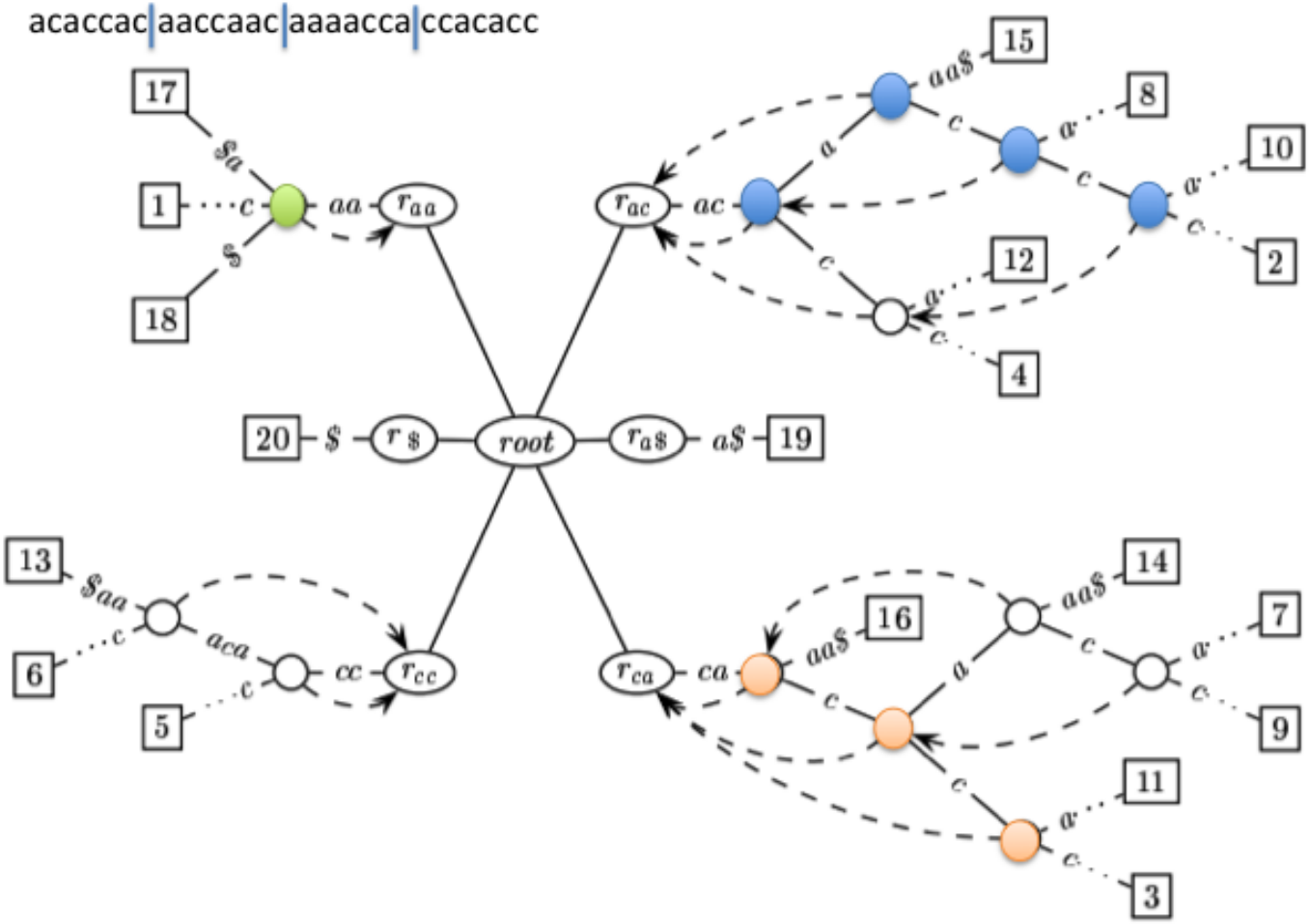
Distributed suffix tree

- New variant of the suffix tree.
- Handle of large strings efficiently.
- Based on linear time construction algorithm for subtrees of a suffix tree.
- It tackles the memory bottleneck problem by constructing these subtrees indepently and in parallel.

Distributed suffix tree



Distributed and parallel search of maximal matches



Conclusions

First year

It has been improved the merge phase of naive parallelization.
It has been adapted a data structure which can be deployed in HPC environments.
It may be used to implement parallel and distributed techniques for search of maximal matches.
This data structure is able of handling large input sequences to search maximal exact matches.

Future work

Design and test the search of maximal matches in multicore environments.
Implement Distributed Suffix Tree.
Perform massive searches of maximal matches: design and test a parallel and distributed algorithm to perform the search of maximal matches in Distributed Suffix Tree for HPC multicore environments.


Thanks!

Whole genome alignment in High Performance Computing environments

Julio César García Vizcaíno Directores: Antonio Espinosa, Juan Carlos Moure

Computer Architecture & Operating Systems Department
Universitat Autónoma de Barcelona

17 de septiembre de 2012