From sequential to massively parallel Rethinking computational-biology applications

Jonathan Rouzaud-Cornabas

LIRIS – Inria Beagle

Overview

From Distributed Systems to Computational Biology

- Distributed Systems (Avalon) : Security, Heuristics, Simulation

From Distributed Systems to Computational Biology

- Distributed Systems (Avalon) : Security, Heuristics, Simulation
- Working with big company : Research is rarely used...

From Distributed Systems to Computational Biology

- Distributed Systems (Avalon) : Security, Heuristics, Simulation
- Working with big company : Research is rarely used...
- Wanted to be embedded in a team/laboratory from another discipline

From Distributed Systems to Computational Biology

- Distributed Systems (Avalon) : Security, Heuristics, Simulation
- Working with big company : Research is rarely used...
- Wanted to be embedded in a team/laboratory from another discipline

- Associate Professor position at INSA-Lyon/LIRIS-Inria Beagle team
- Working on computational biology
- Goal: fully integrated within the team and producing biological knowledge
- Integrating the team through the HPCization of the application

From Distributed Systems to Computational Biology

- Distributed Systems (Avalon) : Security, Heuristics, Simulation
- Working with big company : Research is rarely used...
- Wanted to be embedded in a team/laboratory from another discipline

- Associate Professor position at INSA-Lyon/LIRIS-Inria Beagle team
- Working on computational biology
- Goal: fully integrated within the team and producing biological knowledge
- Integrating the team through the HPCization of the application

- Teaching HPC (OpenMP, CUDA, MPI) : Computer Science and Computational Biology students (M2)

Beagle : Computational Biology at the cell level

- Joint research group : LIRIS, LBBE (Biometry and Evolutionary Biology), CarMeN (Cardio-Metabolism, Diabetes, Nutrition)
- Interdisciplinary team : Computer Science, Biology, Physics, Modelization
- Two main topics :
  - Computational Cell Biology
  - In silico Models of Evolution

Origins of Life Complexity

- Observation : The biological complexity has increased during geological time.
- Question : Why the life is so complex ? Why simple organism can be so complex ?

Origins of Life Complexity

- Observation : The biological complexity has increased during geological time.
- Question : Why the life is so complex ? Why simple organism can be so complex ?
- Two theories :
  - Complex organisms can outcompete simple ones
  - Complexity is due to the variation process that is biased toward an increase of complexity

Origins of Life Complexity

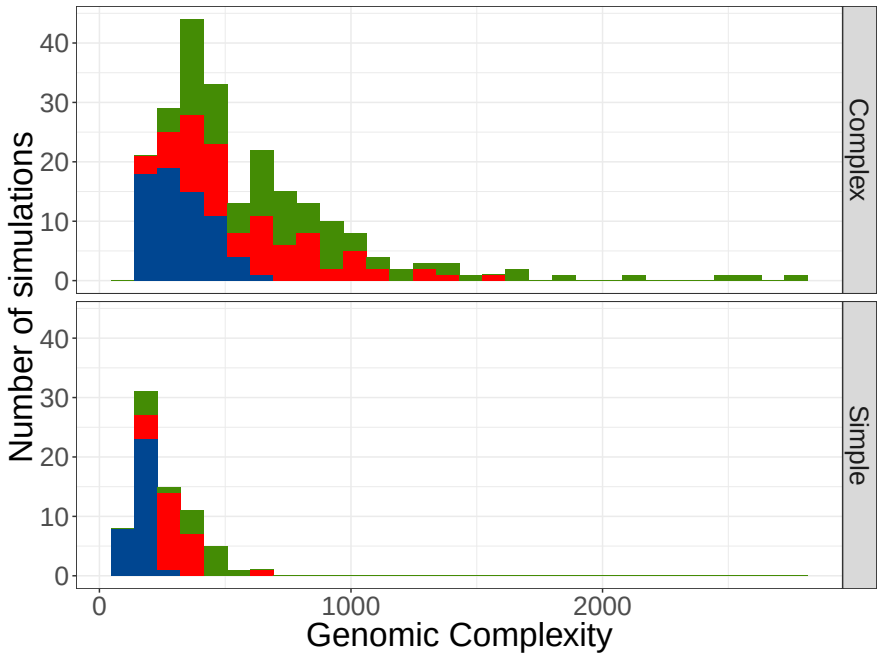
- Observation : The biological complexity has increased during geological time.
- Question : Why the life is so complex ? Why simple organism can be so complex ?
- Two theories :
  - Complex organisms can outcompete simple ones
  - Complexity is due to the variation process that is biased toward an increase of complexity
- Two paradoxes :
  - C-value paradox : Complexity is not linked to the size of DNA (many plants have larger genome than humans)
  - G-value paradox : Complexity is not linked to the number of genes (wheat have 5 times more genes than humans).

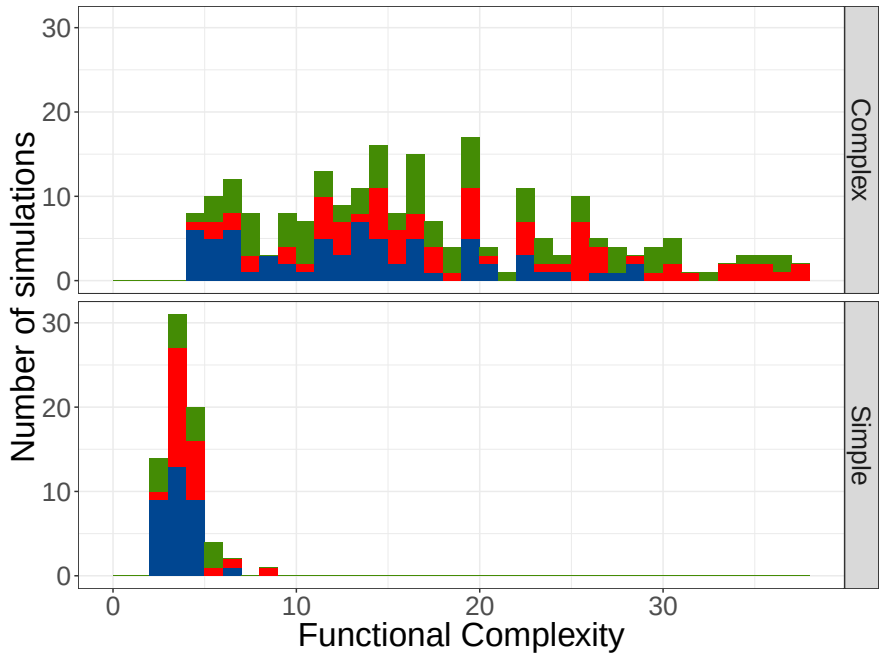
In-silico experimental evolution : Aevol

- How can we built an experiment to test it ? Almost impossible with in-vivo/vitro approaches
- We use *in-silico* experimental evolution to test it (*i.e.* a computational model)
- We define a target that can be fulfilled by a very simple organism
- We run our simulator with different parameters and observed the outcome

The Complexity Ratchet

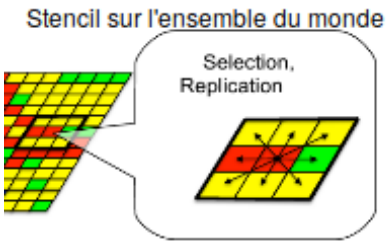
Mutation rate (μ)	Number of *Simples*	Number of *Complexes*
10^{-4} mut.bp^{-1}.gen^{-1}	32 [24 – 43]	68 [58 – 76]
10^{-5} mut.bp^{-1}.gen^{-1}	25 [18 – 34]	75 [66 – 82]
10^{-6} mut.bp^{-1}.gen^{-1}	14 [9 – 22]	86 [78 – 91]

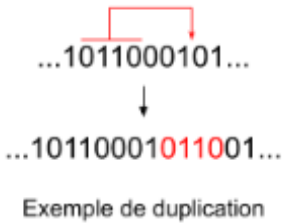


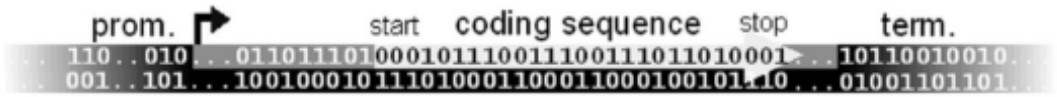


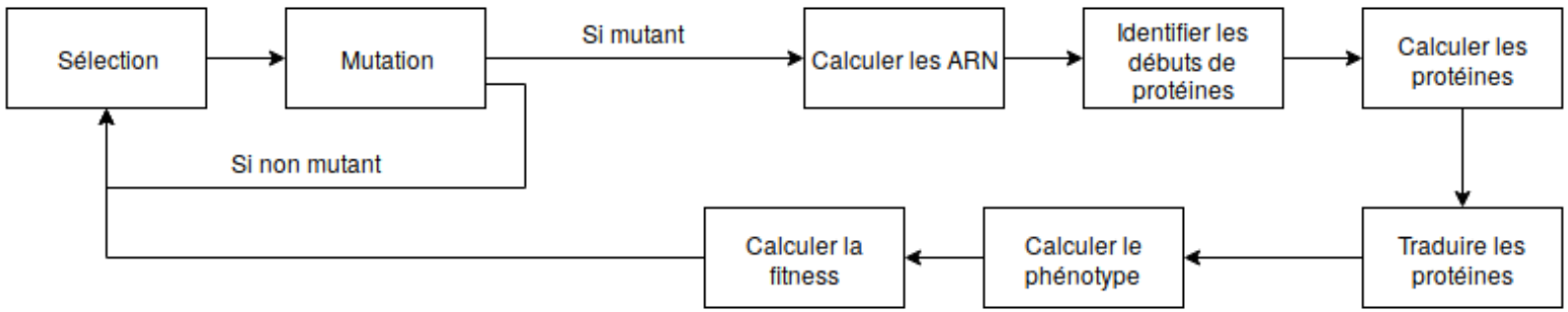
Conclusion : Once you start to become complex, it is almost impossible to go back to simple (accumulation)

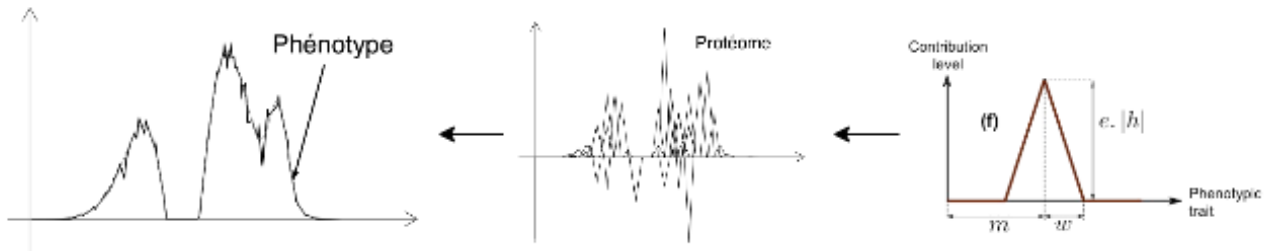
Aevol : Biological model







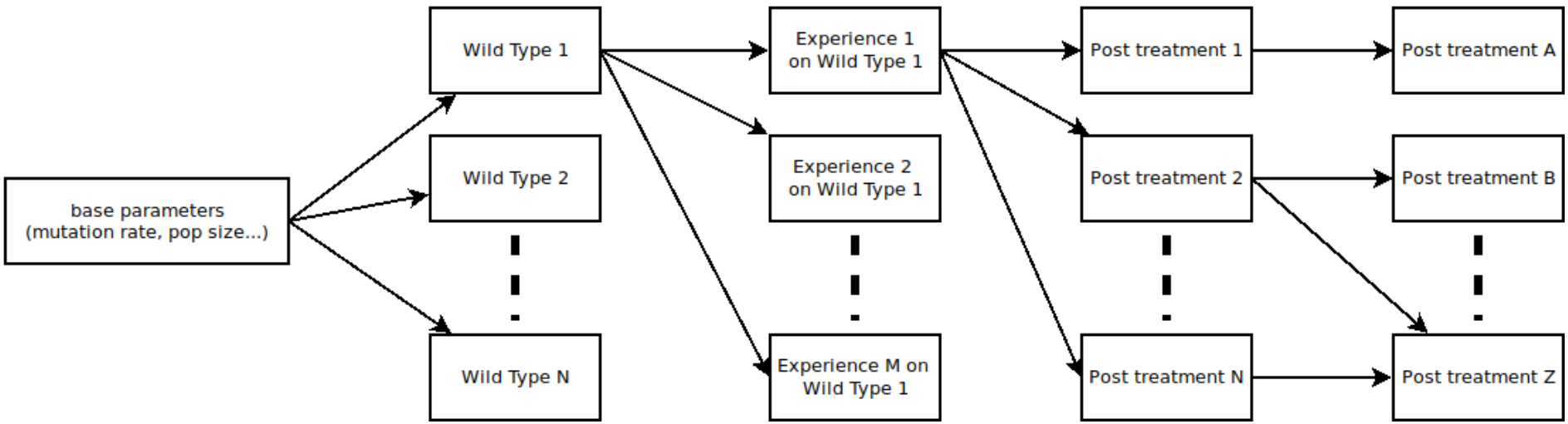




Aevol : some use cases

- How the mutation rate of DNA limit DNA size ?
- Does epigenetics accelerate the evolution ?
- Impact of the different type of mutations on the evolution ?
- How the genetic network are built and wrought ?
- Can we predict evolution ? (and how ?)

Aevol : Experimental campaign workflow



Aevol : Computational model

- Searching for motifs (exact and with hamming distance) on circular strings
- Matching found motifs (match a start with the first next end)
- Decoding triplet of chars to double value
- Summing triangles
- Summing vectors
- A 2D stencil
- Generating random numbers (uniform distribution)

Before going parallel : Optimize, Optimize and Optimize

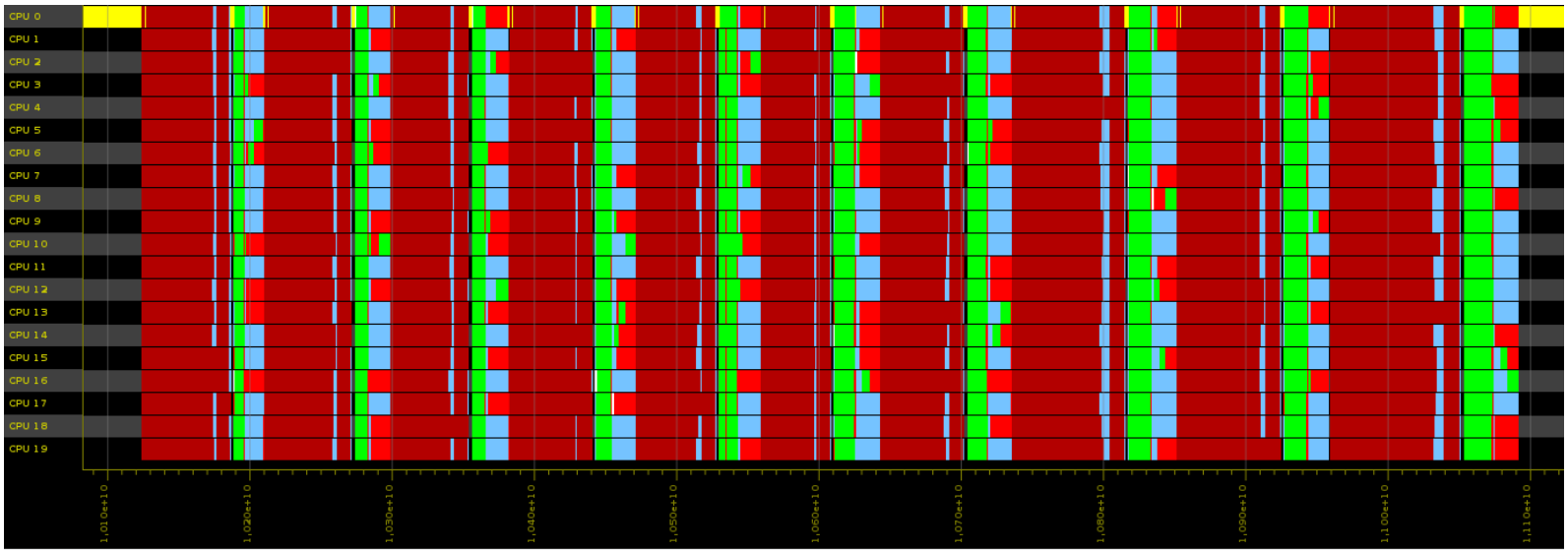
- A scientific application is often written by non-computer science researcher
- Performance aspects is not the first focus
- Code is structured closely to the model

- Profile the code (do not trust yourself or the others)
- Find the hotspot (*e.g.* arithmetics between constants)
- Fix them
- Loop

- Sound simple but mandatory before going further
- Help a lot to understand the code (and interact with the researchers who design the apps)

Aevol and OpenMP

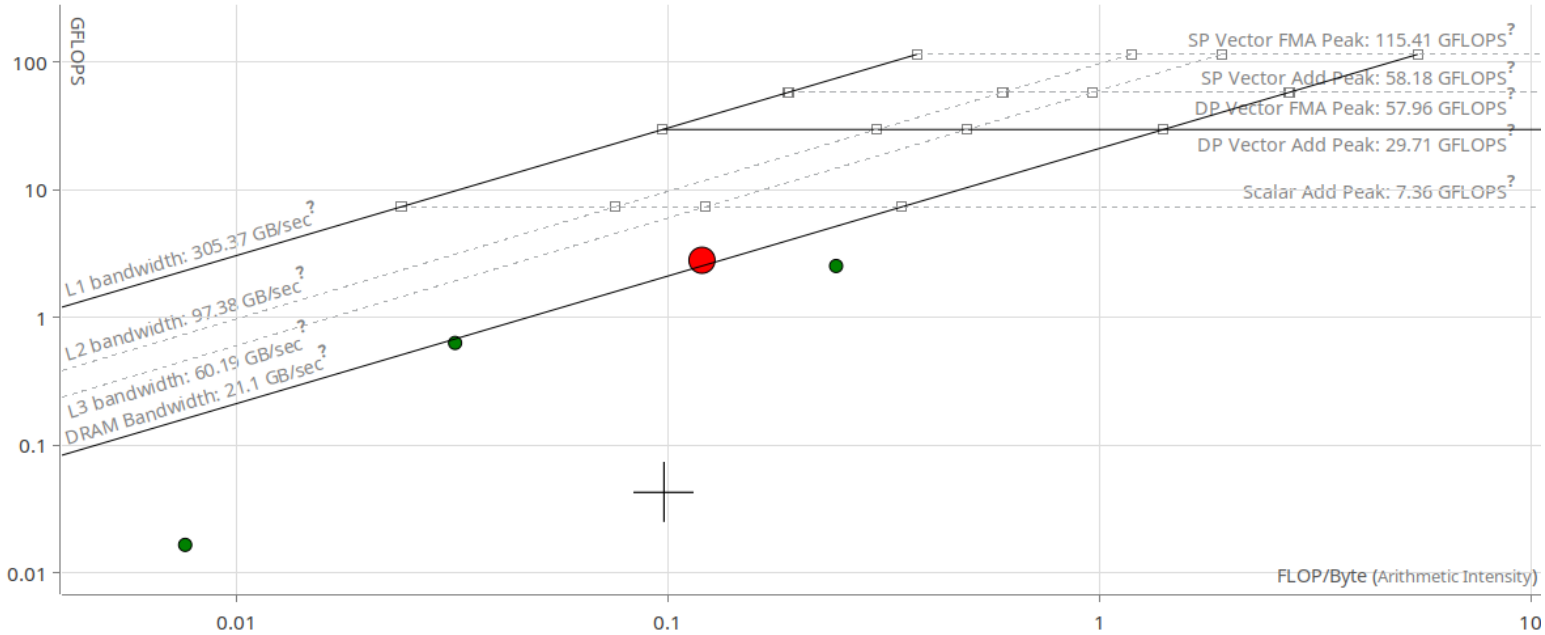
- 3 parallel loop pragma, 2 atomics (counter)



- We also parallelize statistics and checkpointing using task with dependencies
- We try to use task with dependencies everywhere, performance were very bad

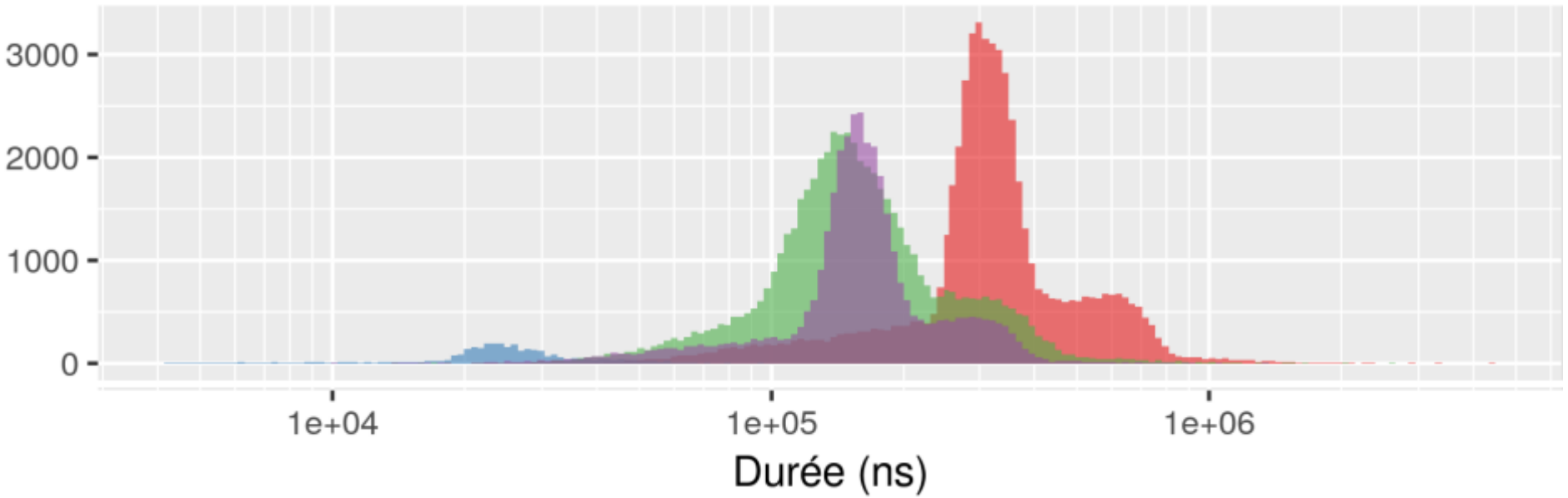
Limitations of the Aevol OpenMP implementation

- The speedup is pretty bad
- Aevol is memory bound (looking for motifs)

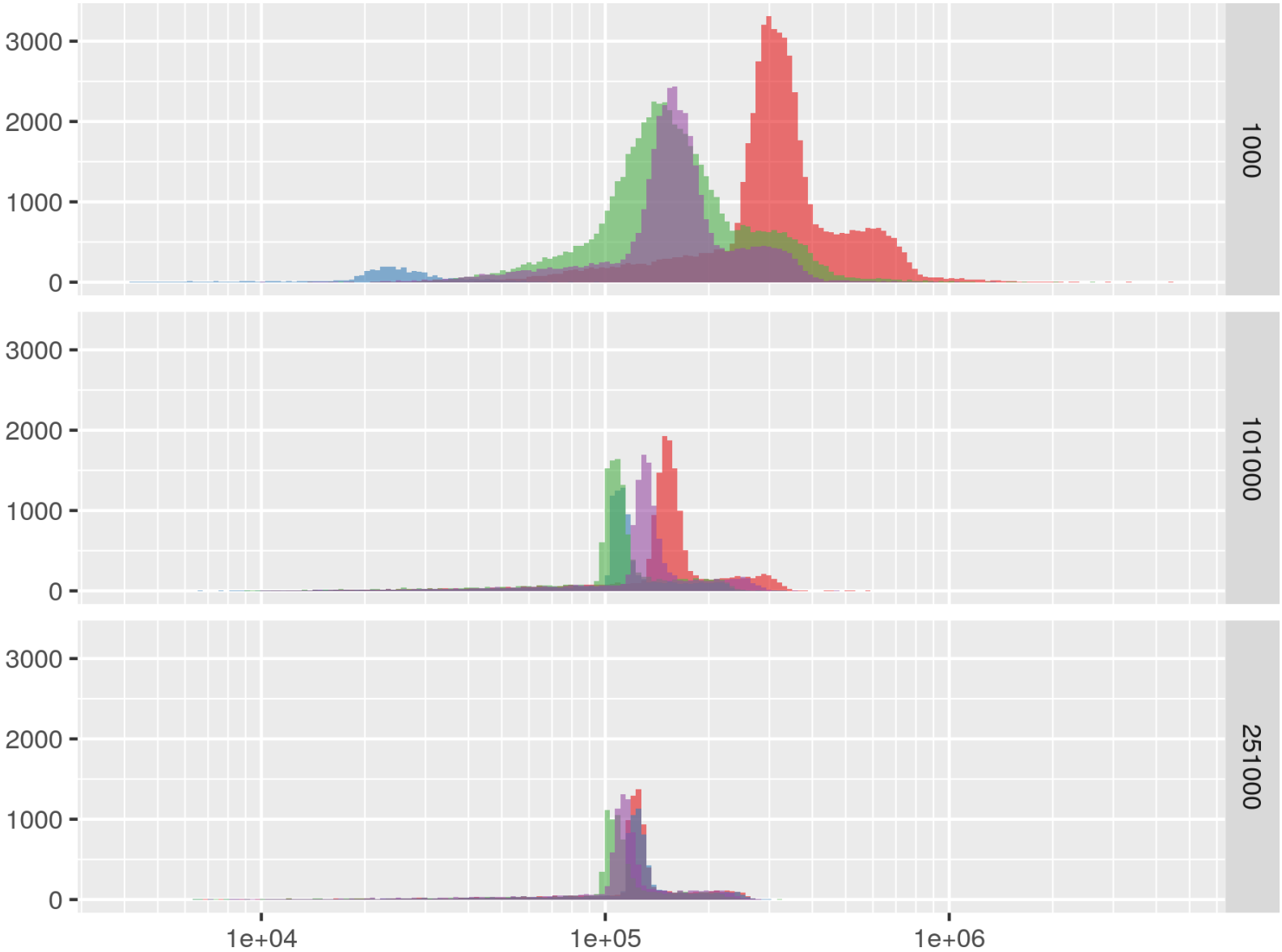


- But there is another issue

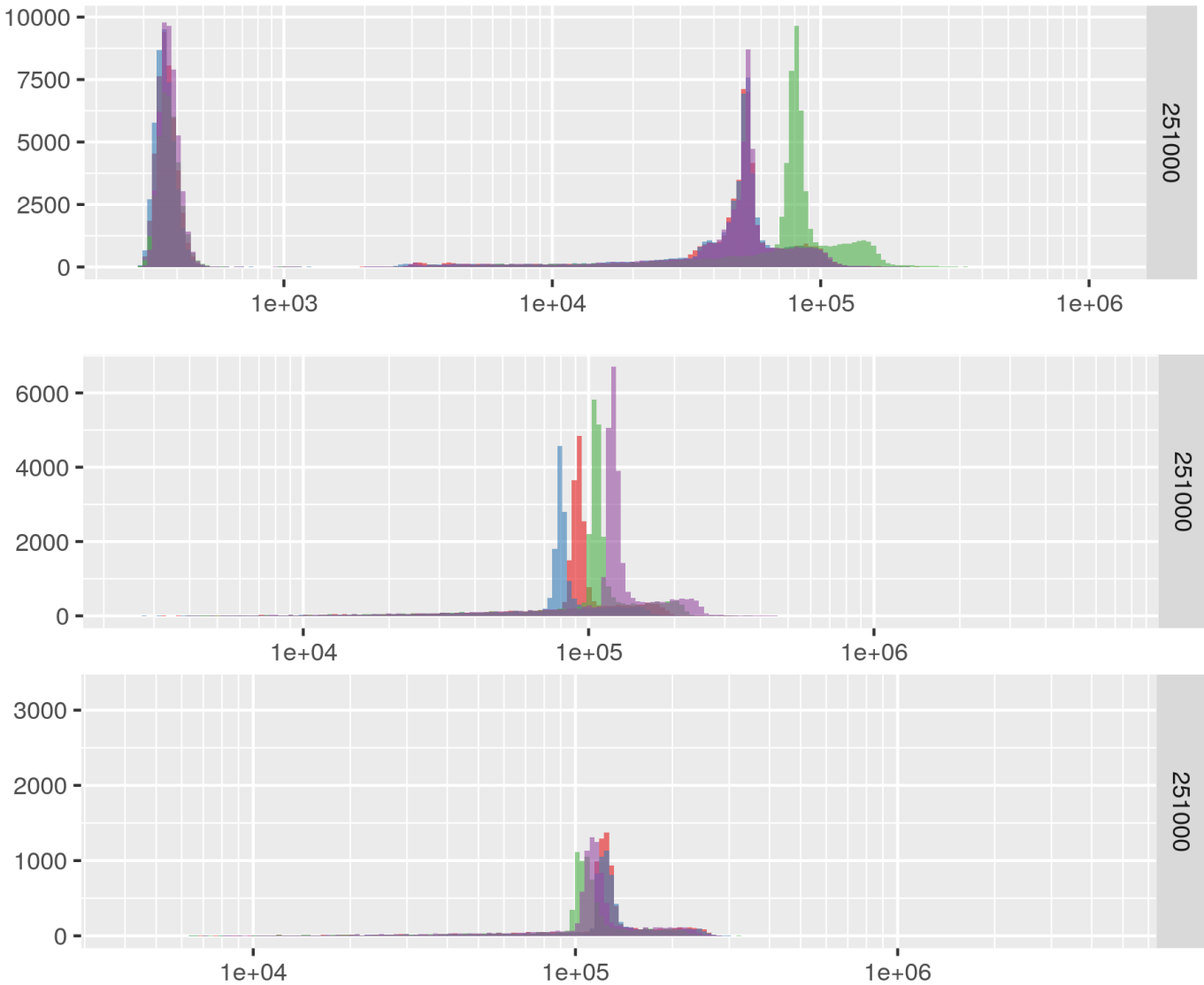
Task runtime distribution at a given timestep



Task runtime during the evolution



Task runtime with different simulation parameters

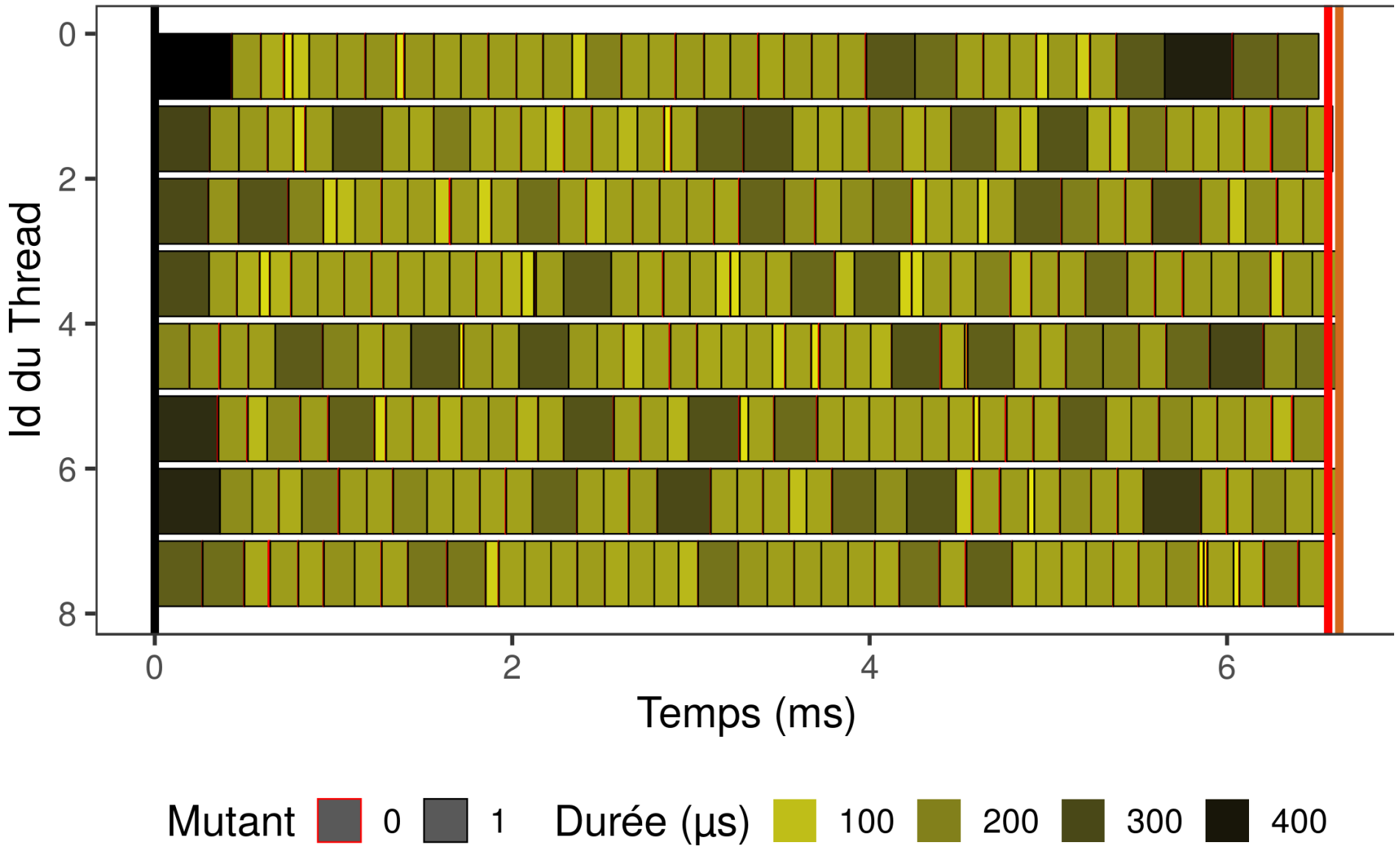


10-4

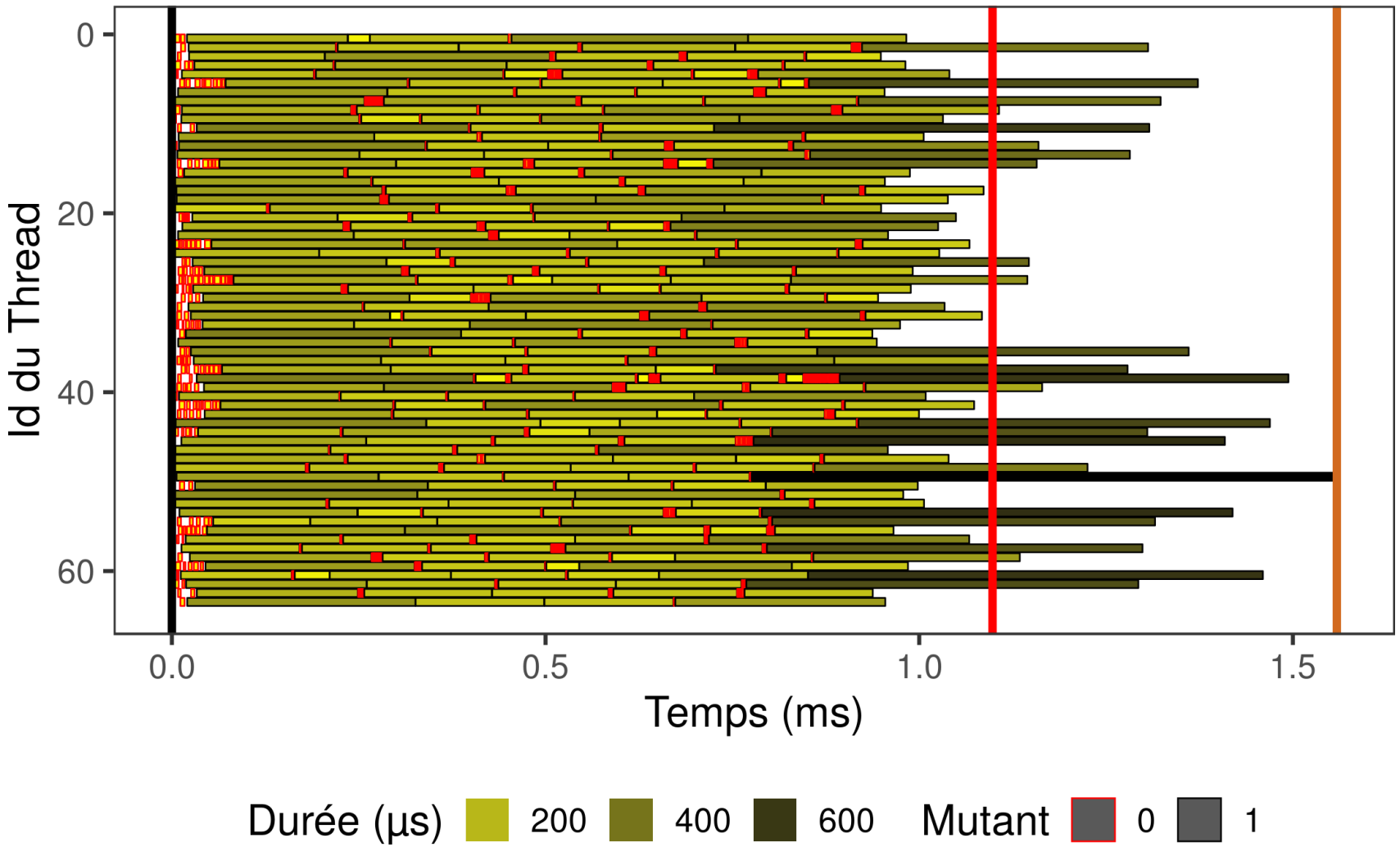
10-5

10-6

Irregularity and OpenMP

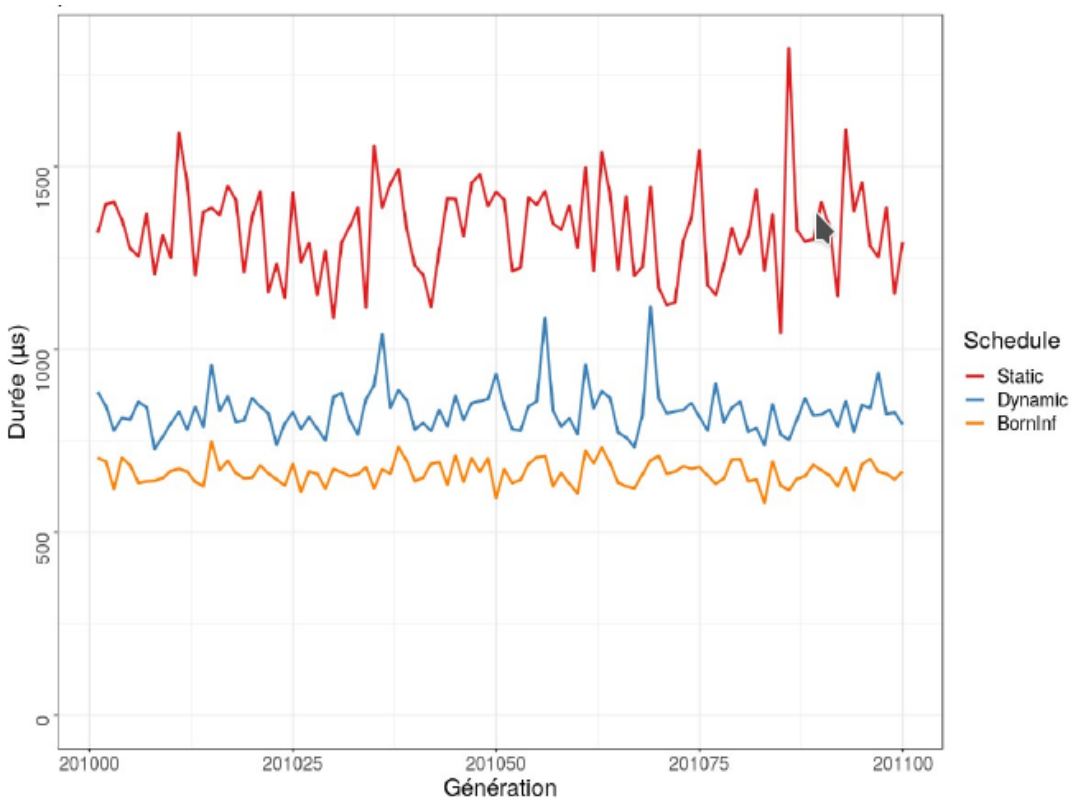


Irregularity and OpenMP



8x more cores, 4.4x speedup

OpenMP scheduling methods



Ordonnancement	Speed Up	Inactivité
Static	33.2	48.1 %
Dynamic	51.2	20 %
Borne Inf	64	0 %

A "simple" list-scheduling issue

- List scheduling (OpenMP dynamic) : (2-1/m)-approximation, O(n log m)
- LPT (Longest Processing Time first) : (4/3 -1/(3m))-approximation, O(n log n + n log m)

A "simple" list-scheduling issue

- List scheduling (OpenMP dynamic) : (2-1/m)-approximation, O(n log m)
- LPT (Longest Processing Time first) : (4/3 -1/(3m))-approximation, O(n log n + n log m)
- But can we model task runtime ?

A "simple" list-scheduling issue

- List scheduling (OpenMP dynamic) : (2-1/m)-approximation, O(n log m)
- LPT (Longest Processing Time first) : (4/3 -1/(3m))-approximation, O(n log n + n log m)
- But can we model task runtime ?
- We try but too difficult and some hardware artefacts

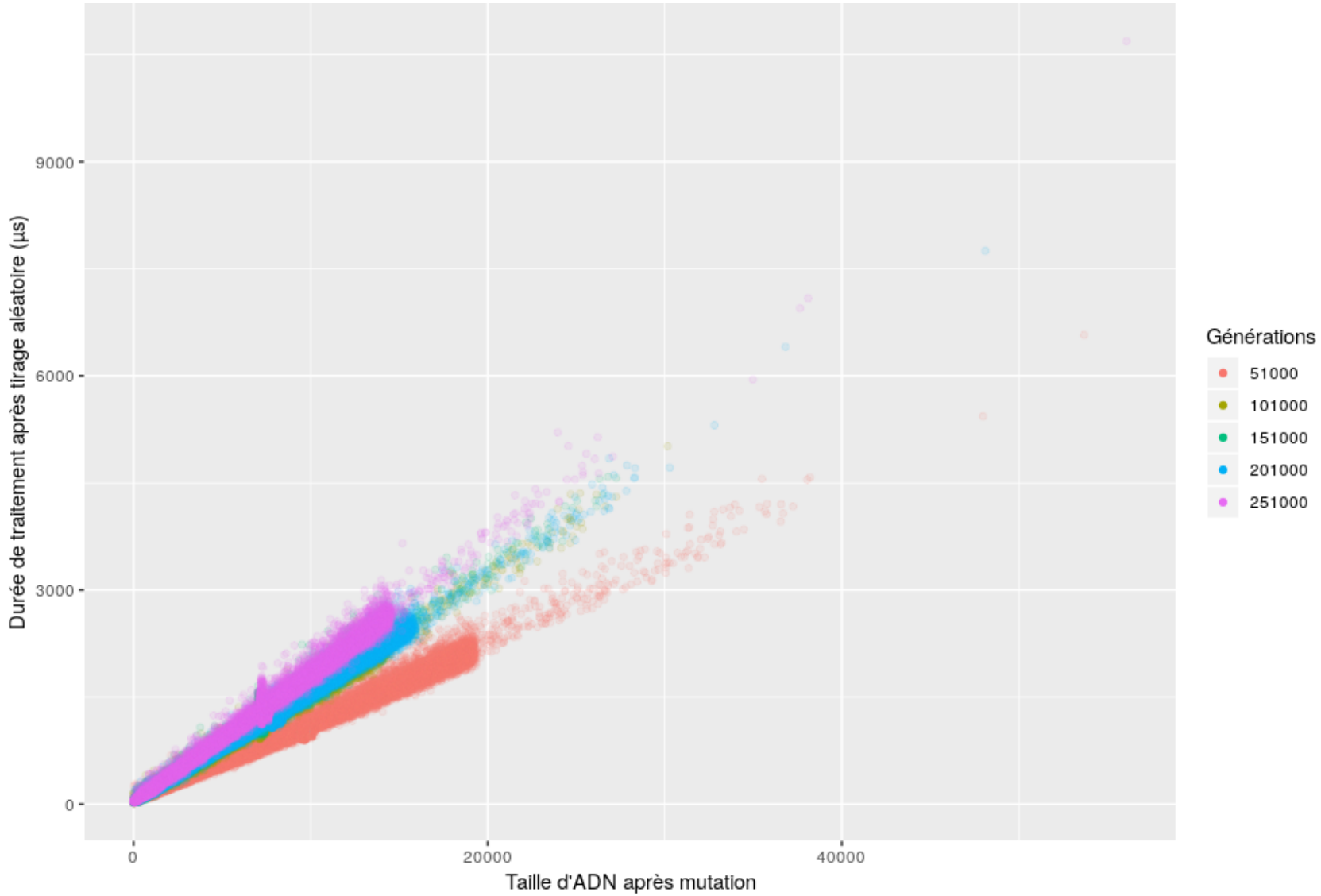
A "simple" list-scheduling issue

- List scheduling (OpenMP dynamic) : (2-1/m)-approximation, O(n log m)
- LPT (Longest Processing Time first) : (4/3 -1/(3m))-approximation, O(n log n + n log m)
- But can we model task runtime ?
- We try but too difficult and some hardware artefacts

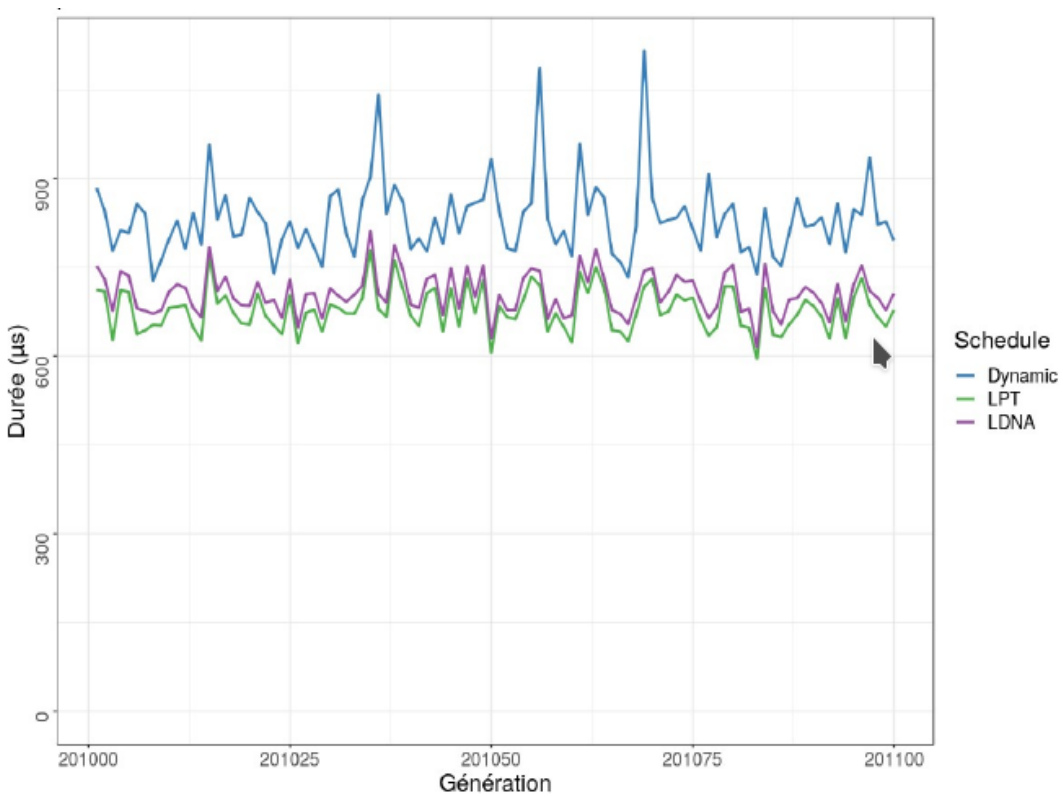
A "simple" list-scheduling issue

- List scheduling (OpenMP dynamic) : (2-1/m)-approximation, O(n log m)
- LPT (Longest Processing Time first) : (4/3 -1/(3m))-approximation, O(n log n + n log m)
- But can we model task runtime ?
- We try but too difficult and some hardware artefacts
- We do not need to predict runtime only to predict tasks order

How to predict task order with Aevol



Preliminary results



Ordonnancement	Speed Up	Inactivité
Dynamic	51.2	20 %
LDNA	59.8	3.4 %*
LPT	62.8	1.9 %

Aevol VS mini-Aevol

- Aevol VS mini-Aevol SLOC : 87,000 VS 2,000
- Simplify model
  - Only a strand of DNA (and not too) *i.e.* the DNA is read one-way and not both-way
  - A reduce mutation sets: no translocation
  - A lot of advances features are missing: no plasmid, 4bp DNA, ...
- Implementation
  - Lot less robust to input errors
  - Can not change model parameters during an evolution
  - No phylogenetic tree
  - No postprocessing tool

Université Claude Bernard

Lyon 1

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES LYON

LIRIS

EuroHack 2018
Lugano – October 2018

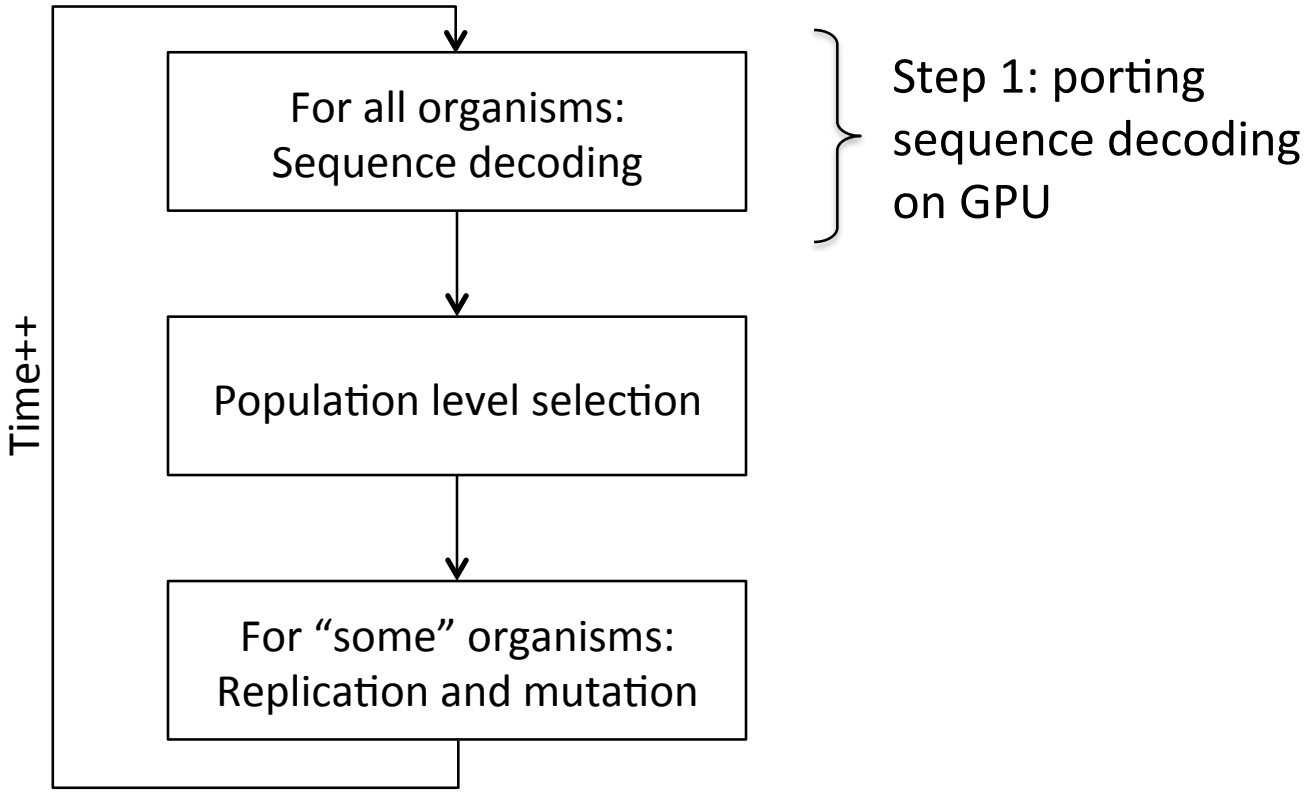
GPU-Aevol

Guillaume Beslon – Computational Biology
David P. Parsons – Software engineering
Jonathan Rouzaud-Cornabas – High Performance Computing
Mentors: Vasileios Karakasis (ETH Zurich), Jeffrey Kelling (HZDR, Dresden)

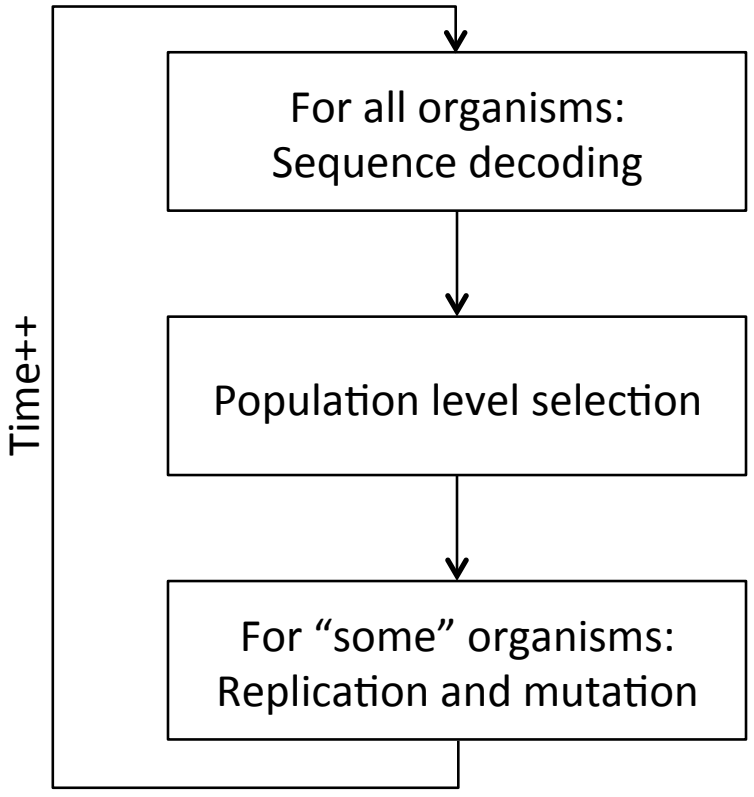
GPU-Aevol – Starting point
(Monday morning)

- Aevol had never been ported on GPU
- Mini-Aevol
  - Aevol simplified and not optimized
  - ~2,000 C++ lines (vs. ~67,000 C++ lines for aevol)
  - One evident parallel scheme: the individual level (but very high heterogeneity)
  - No clear idea on how to efficiently run Aevol on GPU
- Preparation step
  - Optimize Mini-Aevol to enable fair comparisons
  - Replace random-generator by a GPU compatible one

GPU integration in 4 steps



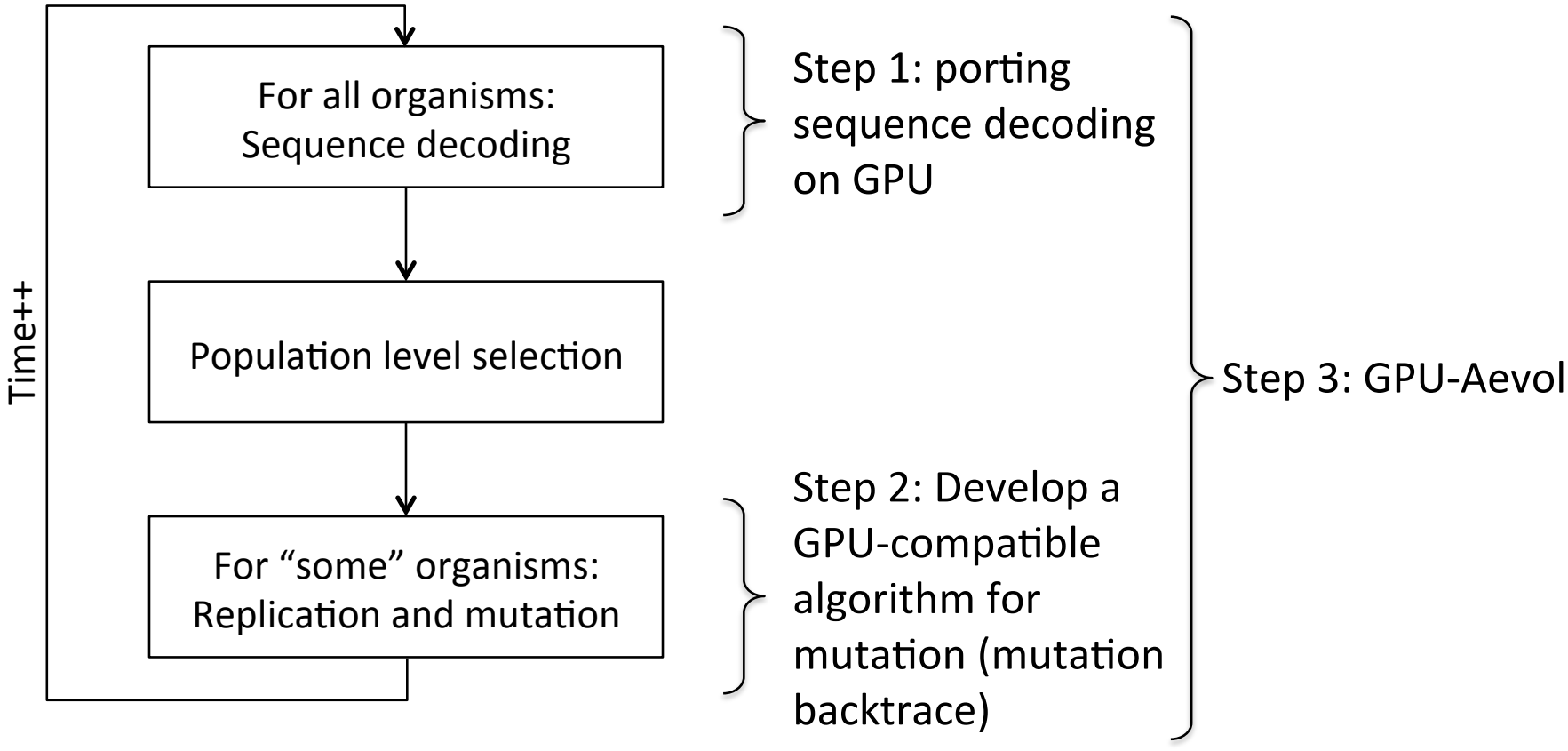
GPU integration in 4 steps



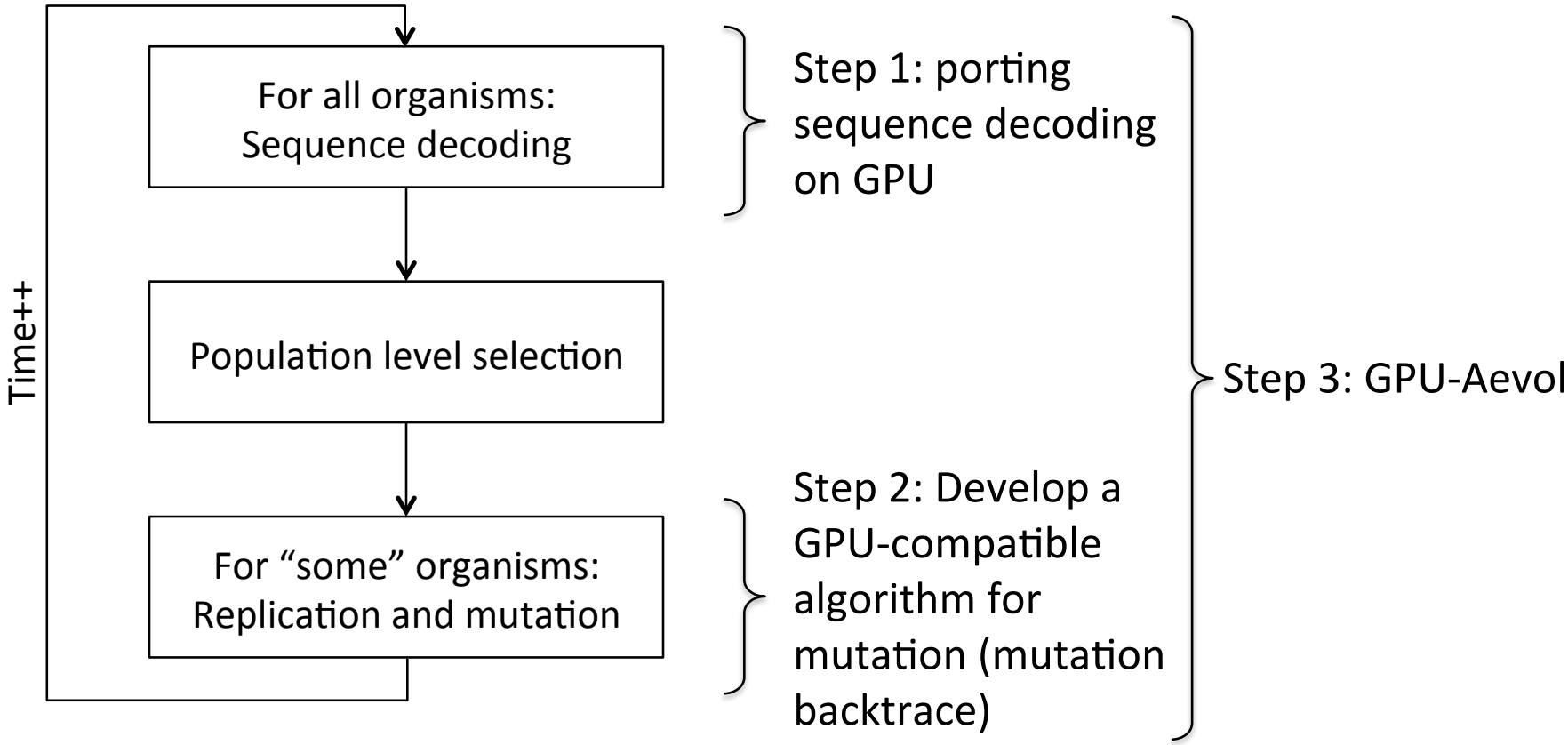
Step 1: porting sequence decoding on GPU

Step 2: Develop a GPU-compatible algorithm for mutation (mutation backtrace)

GPU integration in 4 steps



GPU integration in 4 steps



Step 4: Debugging Optimizing Debugging Optimizing Debugging Optimizing Debugging Optimizing Debugging Optimizing Debugging Optimizing Debugging Debugging Debug Debug Debug Debug

Where we are today

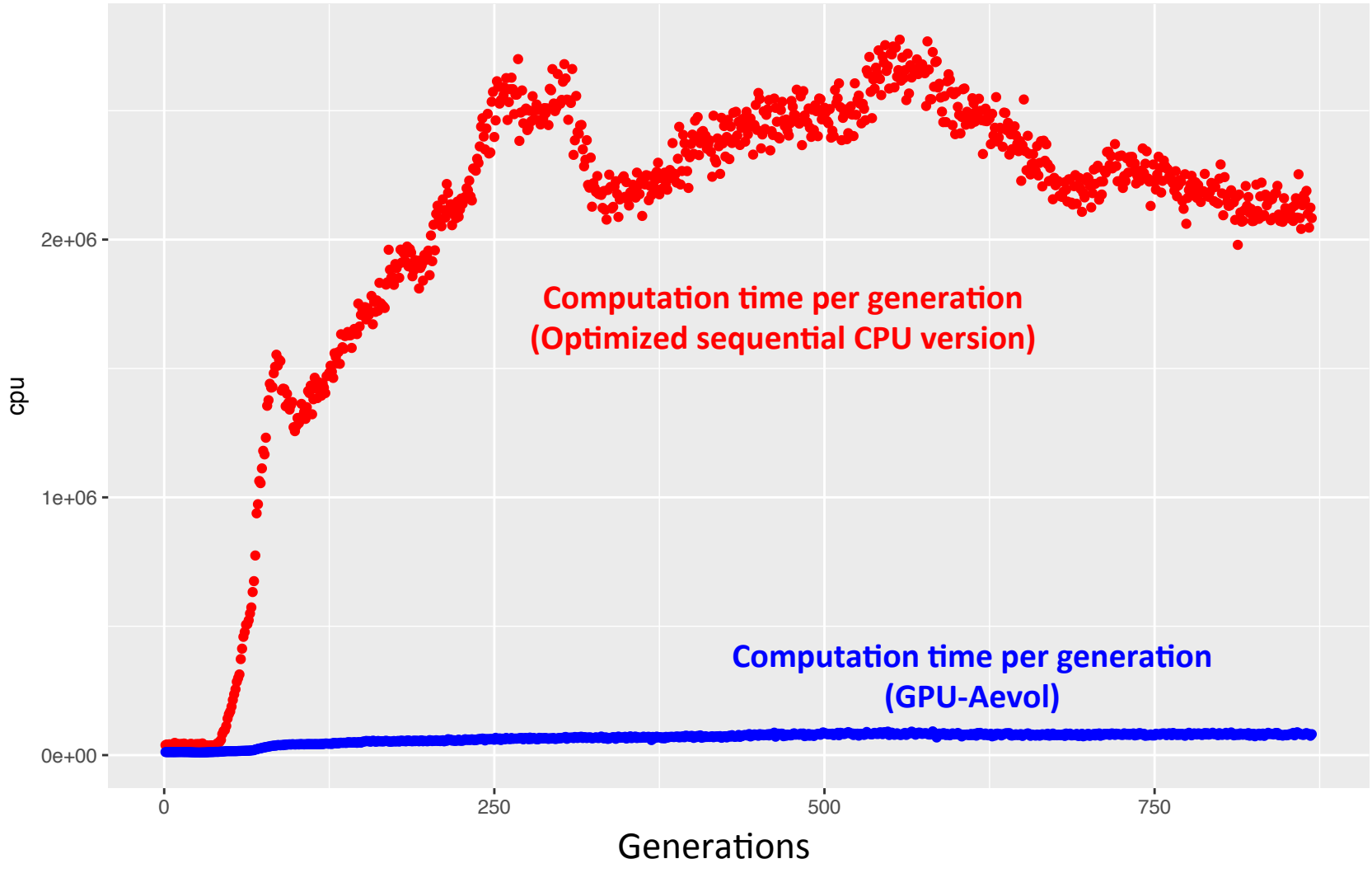
- Full-GPU implem
  - we didn't expect that!
- Speed-up ~X25 **on classical pop sizes**
  - Not much better than CPU openmp speed-up ☹

But

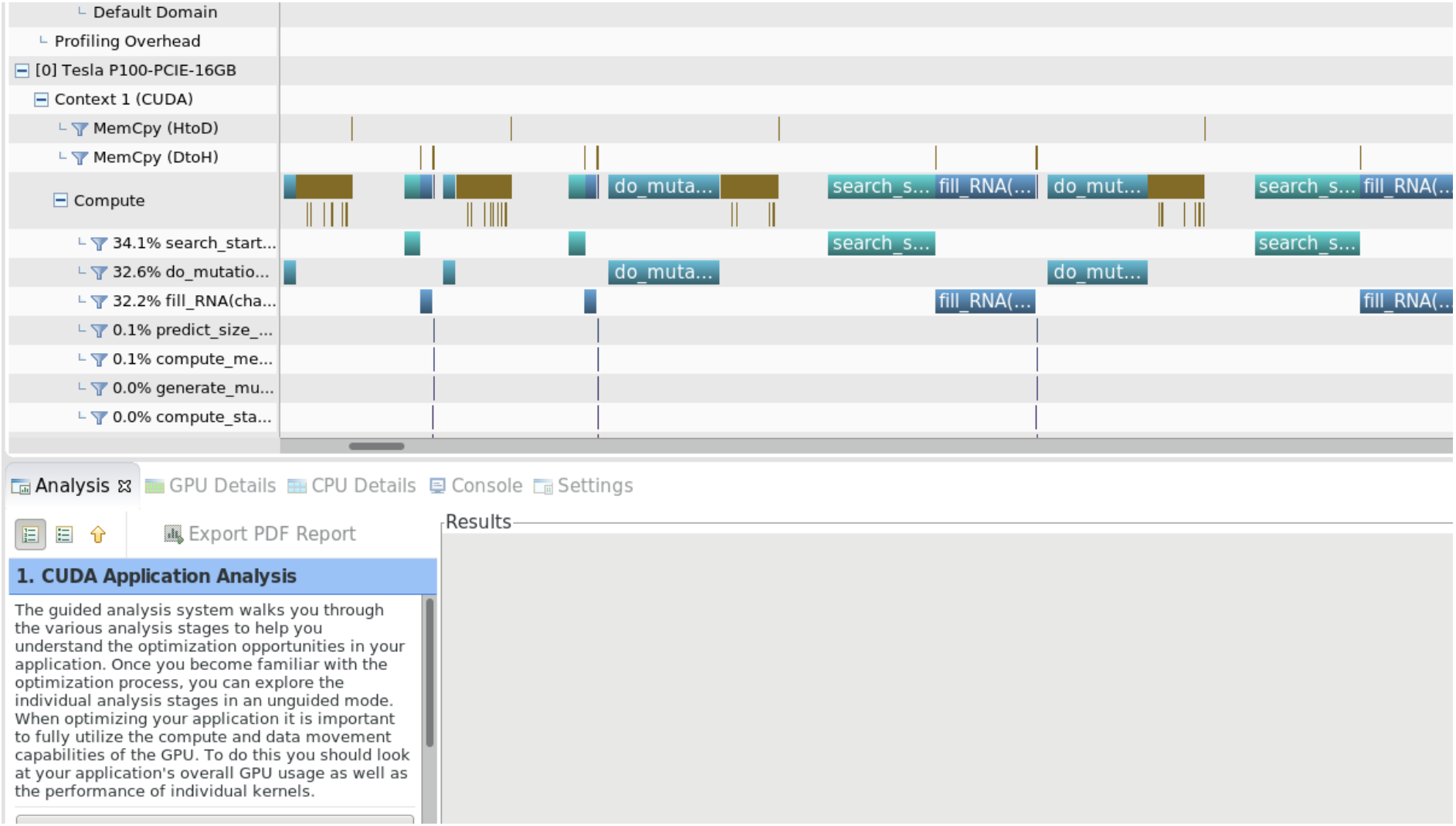
- Large populations run ~as fast as classical ones
  - Far better than CPU openmp speed-up ☺

  → Parallelization scheme seems "reasonable"

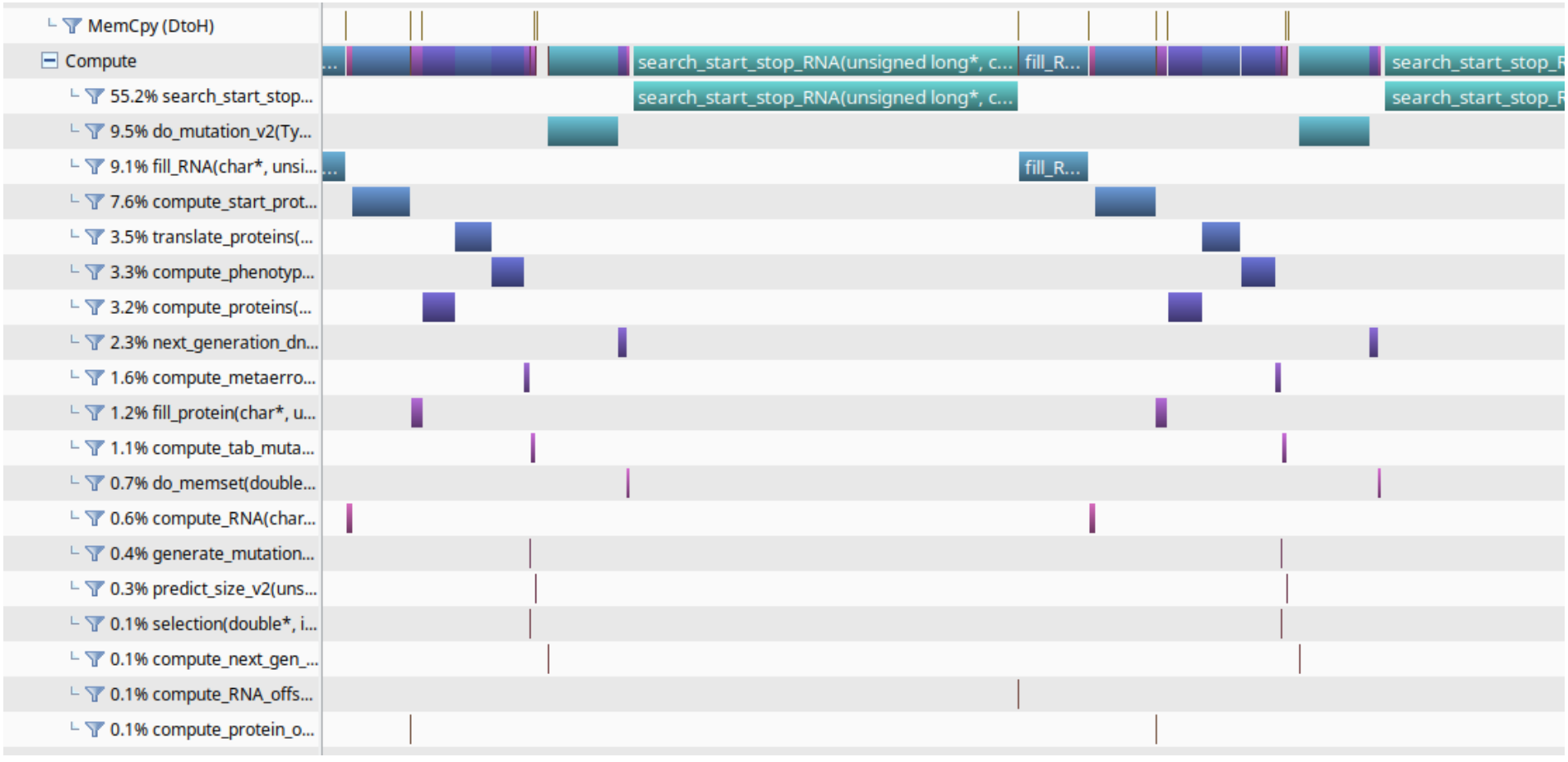
Speedup compared to GPU-optimized Mini-Aevol



Ongoing: profiling of GPU-Aevol (Thursday)



Ongoing: profiling of GPU-Aevol (Friday)



Still many optimization possibilities

- Optimization ideas:
  - Assemble all genomes into a metagenome (suppress heterogeneity)
    - →Done on DNA (estimated gain: >50% in the decoding kernels)
    - →To be done on RNAs
  - Merge decoding and mutation kernels → Easy; To be done
  - Compress genome and metadata
  - Track metadata to avoid "recomputation" (i.e. +/- same optimization idea as on CPU)
    - →Thought to be incompatible with GPU mutation algo.
    - →GPU-compatible algorithm proposed; to be implemented and tested
  - …

Candid feedback on GPU

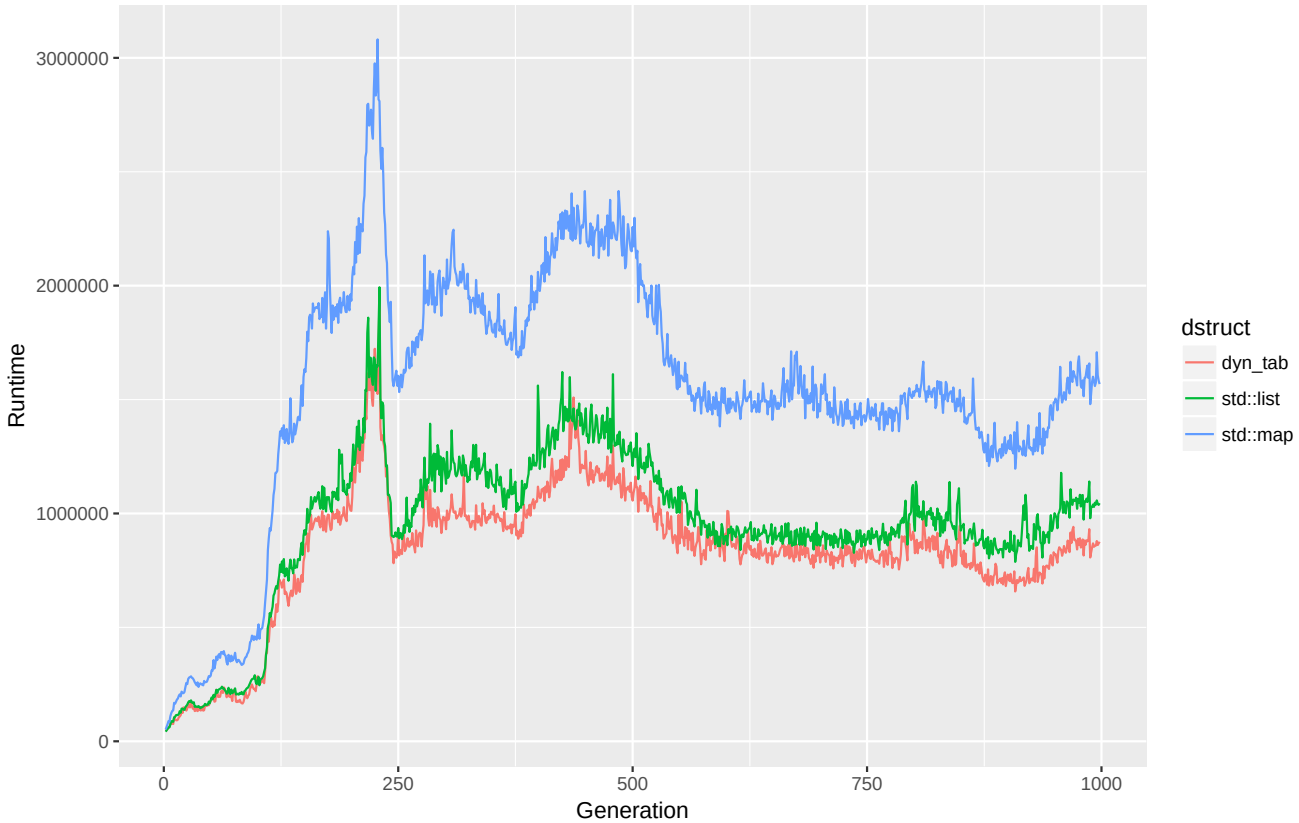
- Among the three of us…
  - Guillaume had no (recent) experience in programming
    - And was actually wondering what he was doing here!
  - David had no experience in GPU
    - But was eager to learn
  - Jonathan had limited experience on GPU
    - But had a theoretical understanding of the concepts
- Conclusion
  - CUDA is surprisingly easy to dive into but…
  - As GPU noobs, we had to change our vision of prog & algorithmics
  - Debugging is a nightmare… Only for dummies ?
  - Also, in depth knowledge on biology and evolution has revealed essential all along the week to find efficient parallel algorithms

Lessons learn on GPU transfer back to CPU

- Modern multi-core processor are more and more closer to GPU
- Vectorization is actually harder to use than going to GPU
- A lot of GPU optimizations have been re-used successfully on CPU

Data and memory bound: THE critical aspect (so far)

- On GPU, no STL
- Dynamic array oversized on GPU
- Working well on CPU too (avoid a lot of memory allocation/copy)
- We need to index position of motifs



How Aevol can use the NVidia DGX-1

- How can we support irregular applications on multi-GPU ? How can we do list-scheduling like optimization ? Mixed precision ?
- Aevol model is evolving, How multi-GPU can help to limit the increase of computing time link to these changes
  - BQR INSA BF2i/LIRIS : To simulate the transmission of bacteries from an organism to another one, we need to simulate dynamic population size (reducing from few thousands to just few ones then growing back to full population size)
  - ANR Evoluthon (Inria Beagle/LBBE) : To simulate multiple population living in different environment and exchanging DNA pieces (horizontal transfer) : Large population (at least x1000) and adding a 1/3-D stencil (DNA exchange)

Scaling ODE : Why ?

- ODE Solver used by a large scope of models in Computational Biology
- On top of it, Deep Learning could also benefit from HPC ODE Solver

- But ODE Solver are limited to small-medium size of ODE system (few thousands)

- ODE Solver does not scale : performance issues

- ExODE : new numerical methods and high performance computing (HPC) approaches to scale ODE solver (at least millions of equations).

Why scalability is hard ?

- N differential equations
- In biology, strongly connected equations : computing time $O(N^2)$

- For the moment, systems of few thousands equations but need to scale the system to a factor of 100-1,000
- Scalability issue : Multipling the size of the systems by 100-1,000, increase the number of equations to solve by $10^4 - 10^6$

- ExODE : Using advances in arithmetic computing unit to scale ODE solver ?
  - New numerical methods
  - New parallel algorithms

Use case #1 : Neuroscience (1/2)

- Synaptic plasticity model : 27 equations for one synapse
- Adding astrocytes : add 15 equations
- A microscopic neural network : 1,000 synapses (in order of 1,000,000 equations)

- 1 differential equations : 10^3 FLOPS
- Microscopic neural network : 10^{12} FLOPS $\Rightarrow$ 1TFLOPS

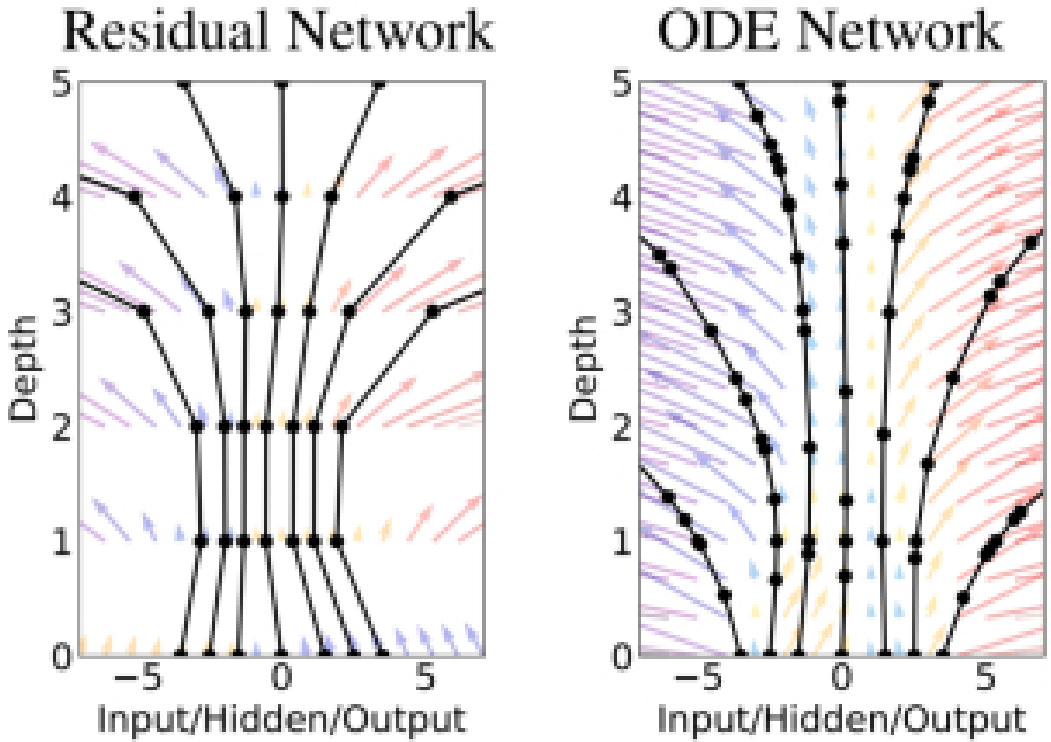
- Solving it for 1,000 time steps : 1PFLOPS

Use case #1 : Neuroscience (2/2)

- Computing microscopic neural network on NVidia Tesla V100 (10k€)
  - 120 seconds double precision (double)
  - 65 seconds single precision (float)
  - 8 seconds half precision (half)
- Targeted platform : DGX-1 with 8-NVidia Tesla V100 ($\Rightarrow$ 1sec)
- Real scale neural network (Summit 1.88 exaflops, could go up to 3.3)
  - Bee : 1ExaFlops
  - Mouse : 1 ZetaFlops
  - Human : 1 YottaFlops
- Computing time can be multiply by 2-6x as we need to solve 2 (RK2) to more time the same system (6 for RK4 with adaptative time step)

Use case #4 : NeuralODE/Deep Learning (1/2)

- Using ODE to replace Recurrent Neural Network (RNN)
- RNNs are one of the key component of Convolution Neural Network that are one of the reason of the hype around Deep Learning
- RNN can be modeled as an Euler solver with discrete time



R. T. Q. Chen, Y. Rubanova, J. Bettencourt, D. Duvenaud. "Neural Ordinary Differential Equations." Advances in Neural Processing Information Systems. 2018

Use case #4 : NeuralODE/Deep Learning (2/2)

- NeuralODE can work on continious (and discrete) data
- More precise and/or stable than RNN
- Decades of works around ExODE

- Need to scale to extreme scale (hundreds of millions of parameters)
- Need to resolve multiple system (learning batches)

Reduced precisions and ODE Solver

- 3 types of floating point number:
  double (64bit), float (32bit), half (16bit)
- Each type is twice as small as the previous one

- Require less time to transfer
- Can computed twice more for each CPU cycle (remember vectorization)

- But rounding errors can be (very) large !
- Can lead to dramatically different results (could be an issue or not)

Multi-precision ODE Solver: Predictor-Corrector

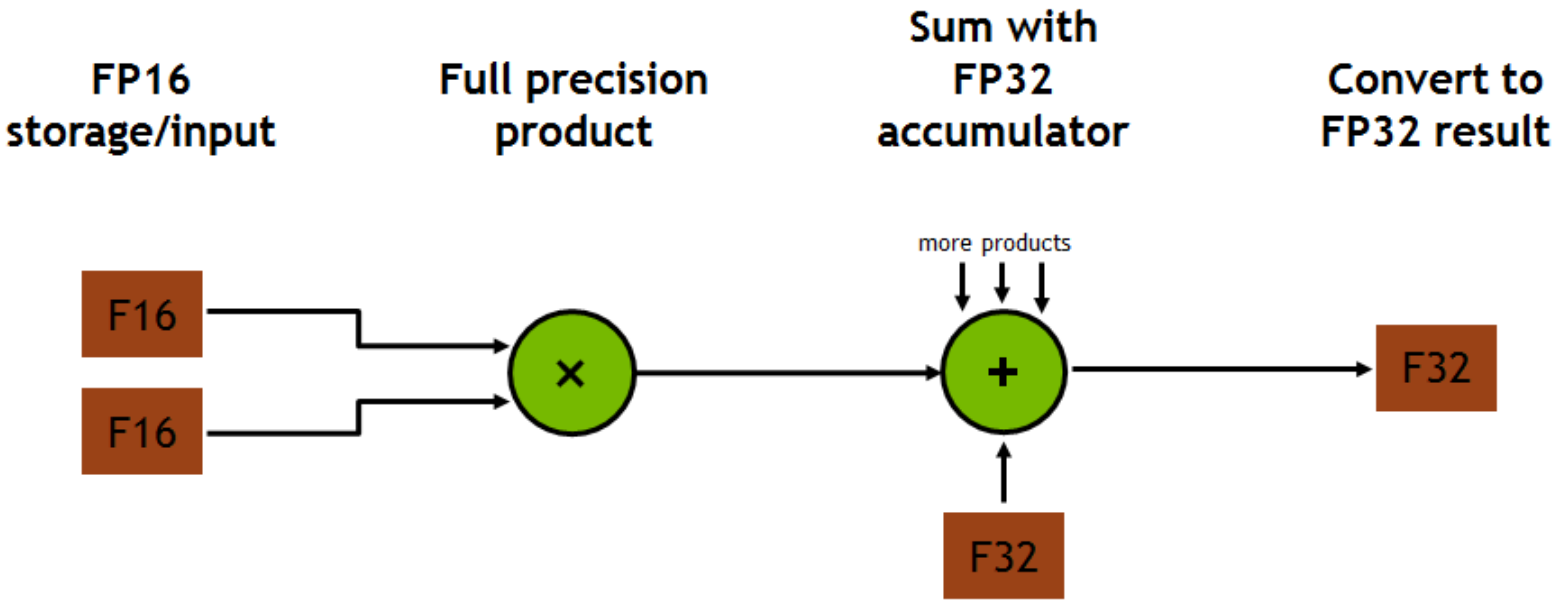
- Idea : Use smaller precision floating point number for most of the compute and larger one to correct the error
- Prediction with reduce precision, correction in full precision

- Classically used with float/double
- Never used with half float

- As we are interested only by statistical properties, reduce precision could be enough

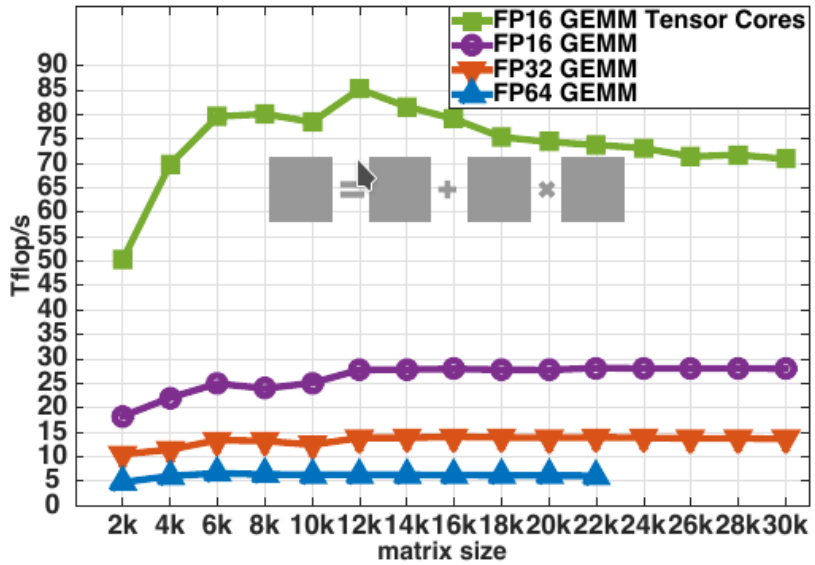
Mixed Precision Arithmetic

- Issue with low precision floating point number : large rounding errors
- Mixed precision within the computation (different floating point type)
- Popular approaches for CNN/Deep Learning
- Input are coded in half precision, accumulation in single precision (limit rounding errors)



Hardware mixed precision

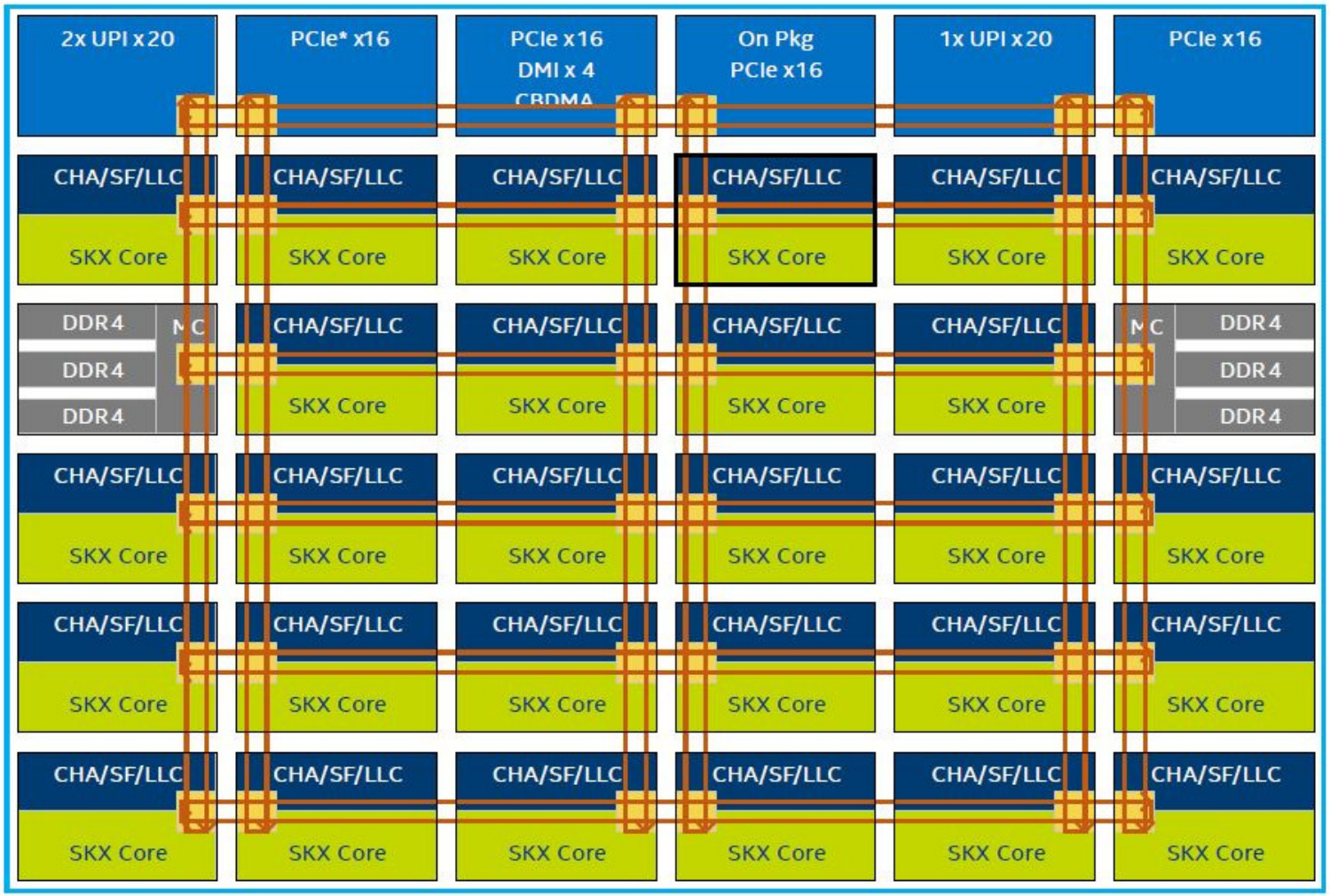
- Convolution of matrix are the base of Deep Learning
- Thanks to it, hardware implementation of
  - half precision floating point numbers
  - mixed precision arithmetic
- Core ideas behind ExODE : Taking advantages of
  - mixed precision within the ODE system
  - half precision and mixed precision arithmetic within PC methods



Parallel methods (1/2)

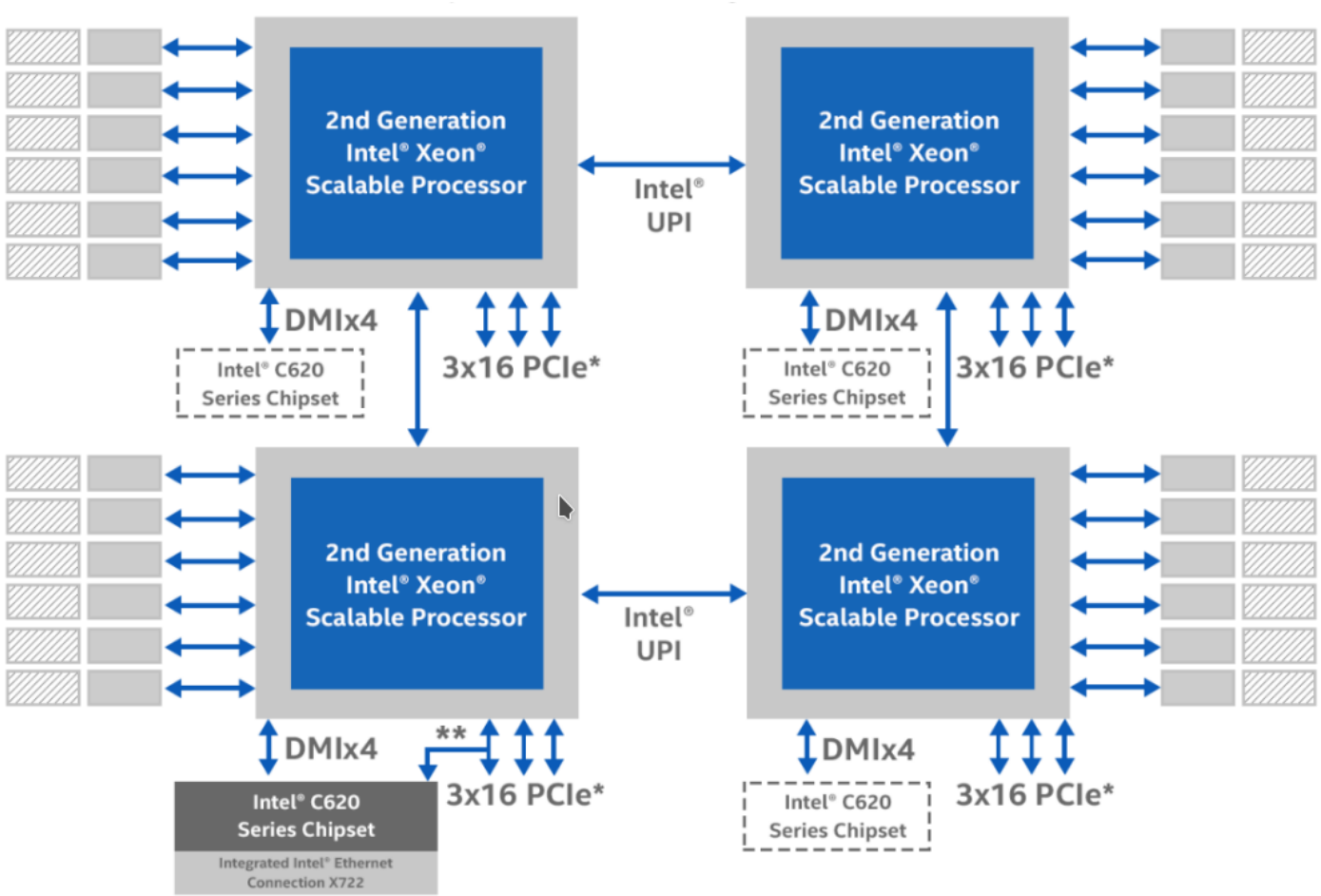
- Mixed precision ODE solving approaches on GPU / CPU
- Parallel solving of the system and/or the method (Euler, RK2, RK45, adaptative time step)
- Not only compute, also transfer time (CPU-GPU, CPU-CPU, GPU-GPU)
- Depending of the ODE system and the wanted precision/performance/accuracy, we should change the resolution methods, floating point number type and prediction-correction methods

Parallel architecture (1/4)



CHA – Caching and Home Agent ; SF – Snoop Filter; LLC – Last Level Cache;
SKX Core – Skylake Server Core; UPI – Intel® UltraPath Interconnect

Parallel architecture (2/4)



Parallel architecture (3/4)

TESLA V100

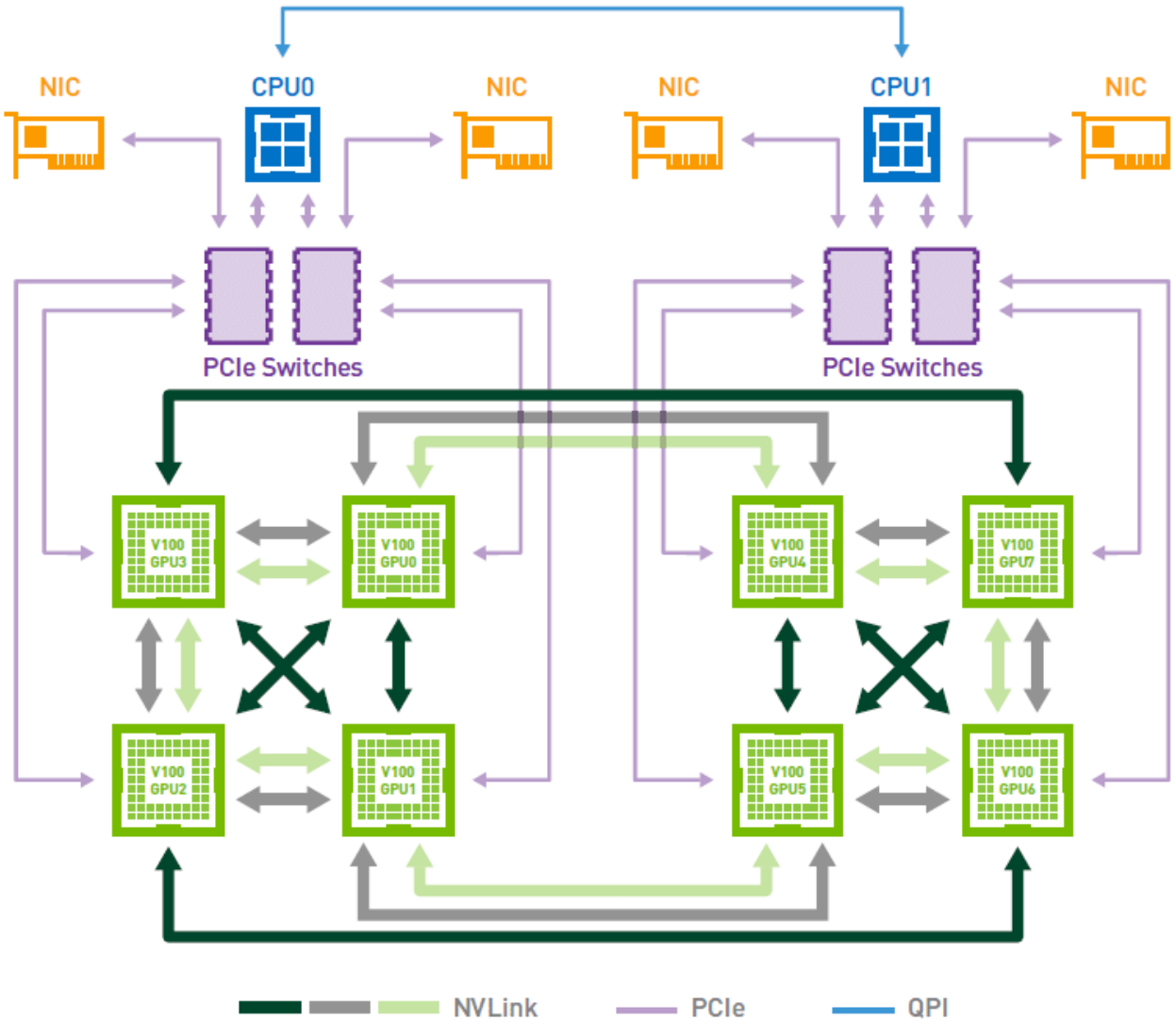
21B transistors
815 mm²

80 SM
5120 CUDA Cores
640 Tensor Cores

16 GB HBM2
900 GB/s HBM2
300 GB/s NVLink

*full GV100 chip contains 84 SMs

Parallel architecture (4/4)



Parallel methods (2/2)

- Mixed precision ODE solving approaches on GPU / CPU
- Parallel solving of the system and/or the method (Euler, RK2, RK45, adaptative time step)
- Taking into account platform heterogeneity (N CPUs, M GPUs, see DGX-1): locality, computing modes, etc.
- Automatically adapting ODE Solver : selecting the "best" (performance VS precision VS accuracy) solving methods based on the properties of the system and user requirements

AEx ExODE

- 3-years
- 3 Inria teams : Beagle, Dracula, Avalon

- Focus on solving ODE, not on producing biology results

- 1 PhD Student
- 2-year engineer