

Hardware-Aware Algorithms for Efficient Molecular Dynamics and Genome Data Analysis

Author

Lorién López Villellas

Supervisors

Jesús Alastruey Benedé and Pablo Ibáñez Marín



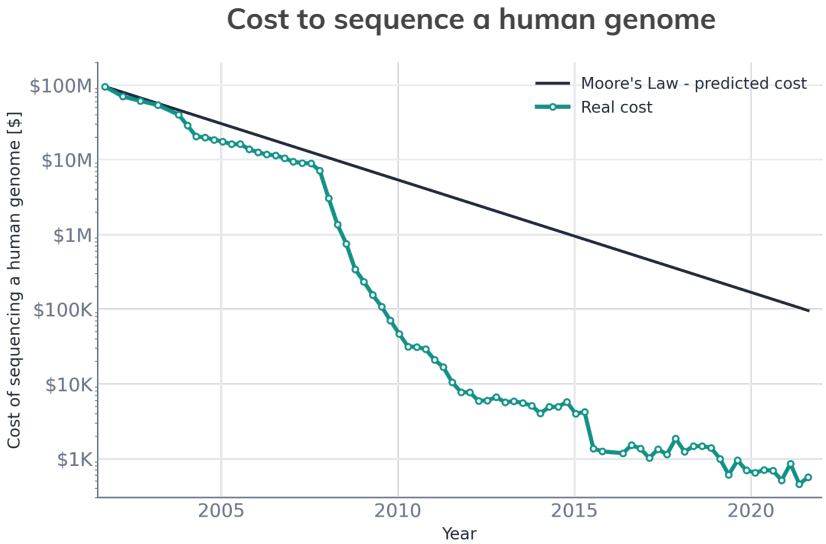
Universidad Zaragoza

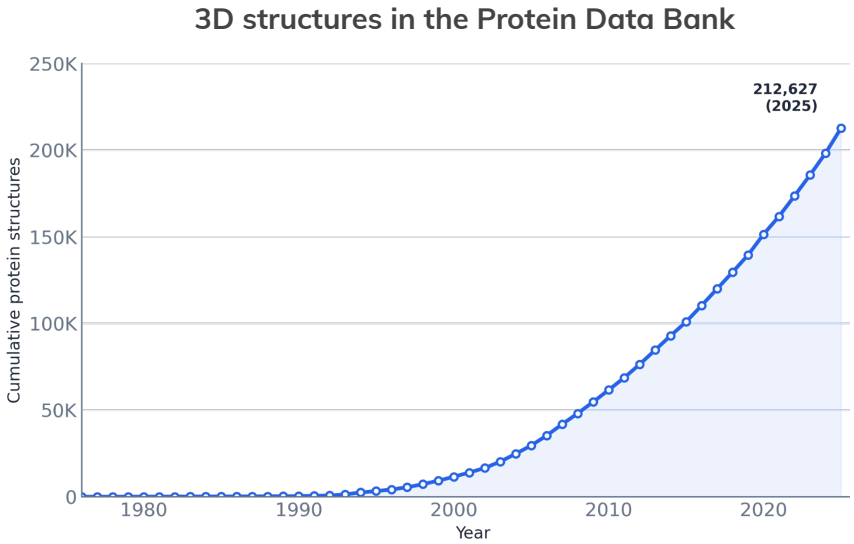


Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Compute-Dependent Science

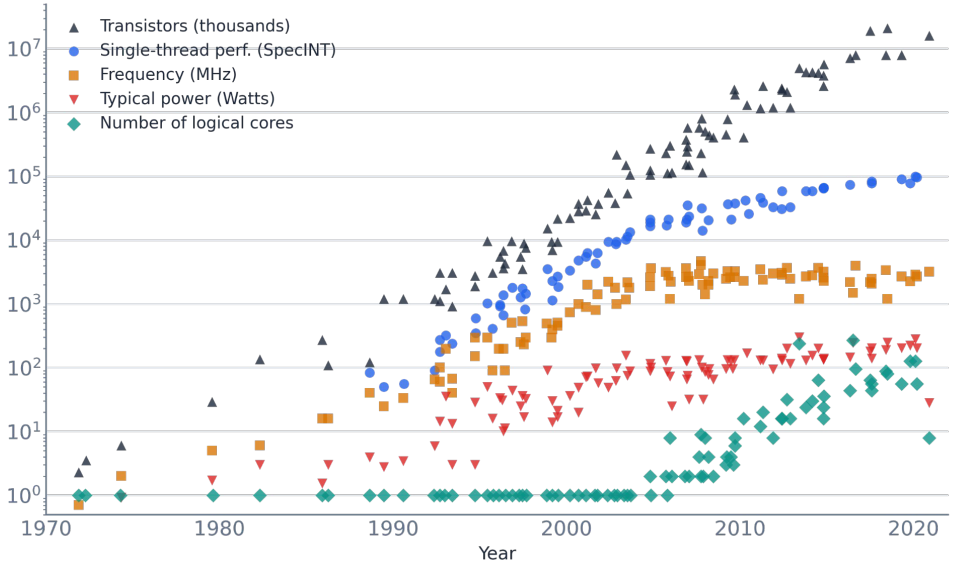
Science is **increasingly dependent** on High-Performance Computing (HPC) to perform large-scale simulations and data analysis





50 Years of Microprocessor Trends

For decades, Dennard scaling and Moore's law made code faster with every CPU generation with **little developer effort**



Clock speed & single-thread stalled

Dennard scaling ended; frequency and per-core gains flattened.

Parallelism is required for performance

Nodes, threads, ILP, SIMD, etc.

... and heterogeneity

GPUs, TPUs, domain specific accelerators, etc.

Heterogeneous HPC Landscape

TOP500 Reference

2016



2026

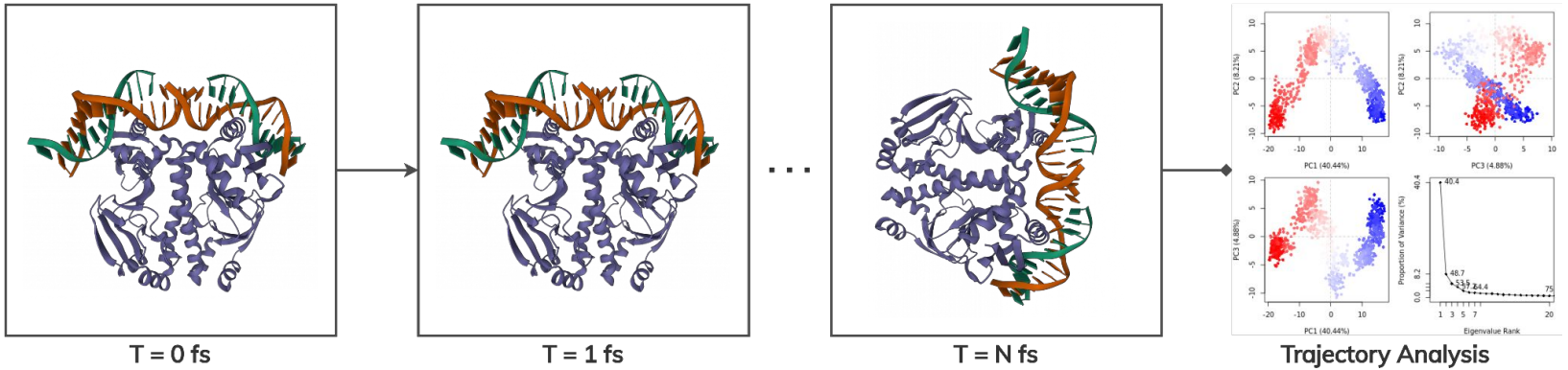


Hardware-Aware Software Design

- Algorithms must be designed and accelerated with the **underlying hardware in mind**
  - Only way to exploit the resources of modern microarchitectures
  - Example: quick sort vs. merge sort, tiled matrix multiplication, etc.
- The properties of the algorithm's **input and output also enable design choices**
  - Example: binary search, radix sort, etc.
- We apply this hardware-aware philosophy to two pillars of modern science
  - **Molecular Dynamics and Genome Data Analysis**
  - (we focus on the CPU, noting that several contributions can be extended to GPUs and other accelerators)

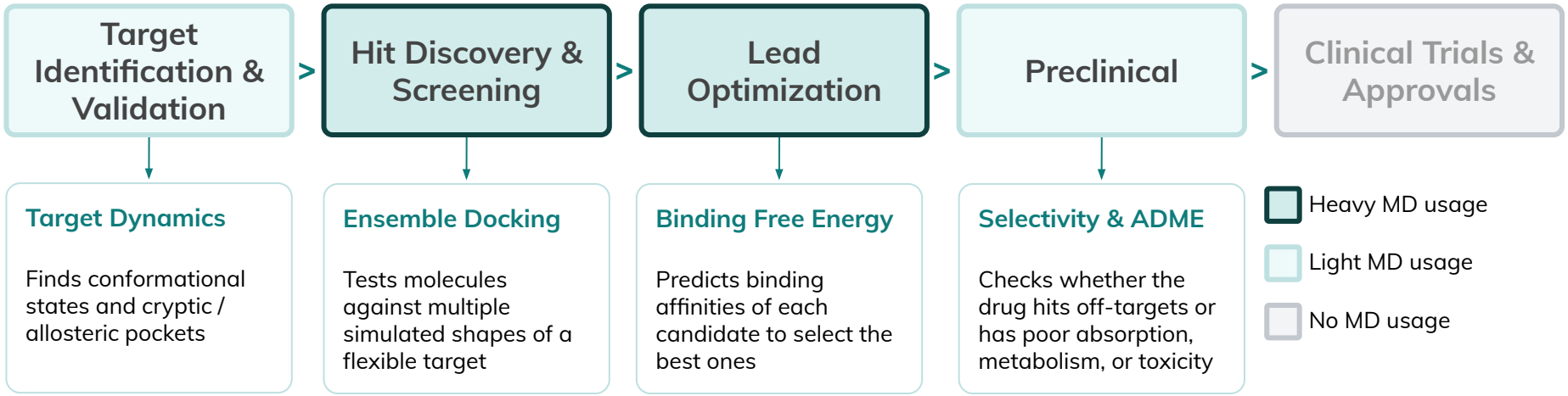
Molecular Dynamics (MD)

- Computer simulation of the evolution over time of interacting particles
  - Solve the equations of motion at discrete time steps (each step covers ~2 fs)
- **Low cost** and **high accessibility** vs. wet lab experiments
  - Applications: drug discovery, nanomaterials, catalysis, food science...



Molecular Dynamics (MD)

Motivation: Drug Discovery



$1-2.8 Billion & 10+ Years

AVERAGE COST & TIMELINE TO BRING A NEW DRUG TO MARKET

Molecular Dynamics (MD)

Simulation Costs and MD Packages

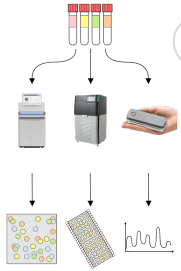
- MD is highly computationally demanding
  - Typical supercomputer throughput: tens to hundreds of ns/day
  - Many biological processes of interest occur at the **μs-ms timescale** (week- to month-long simulations)
- Several widely used MD packages: GROMACS, LAMMPS, NAMD, etc.
  - Different use cases, numerical implementation of the physics, software license…
- We target GROMACS
  - One of the most used HPC tools worldwide

Genome Data Analysis

Genome data analysis relies on **pipelines of multiple tools** that transform the sequencing data from a biological sample (DNA or RNA) into interpretable results

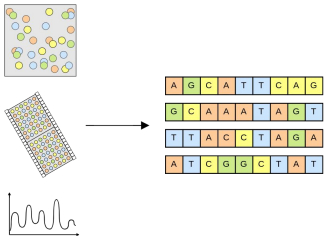
Sequencing

Sample to raw digital data



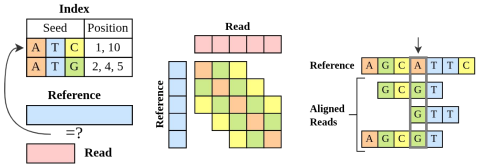
Primary Analysis

Raw signal to sequence reads with quality scores



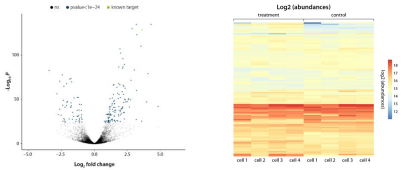
Secondary Analysis

Reads organized into genomic structure



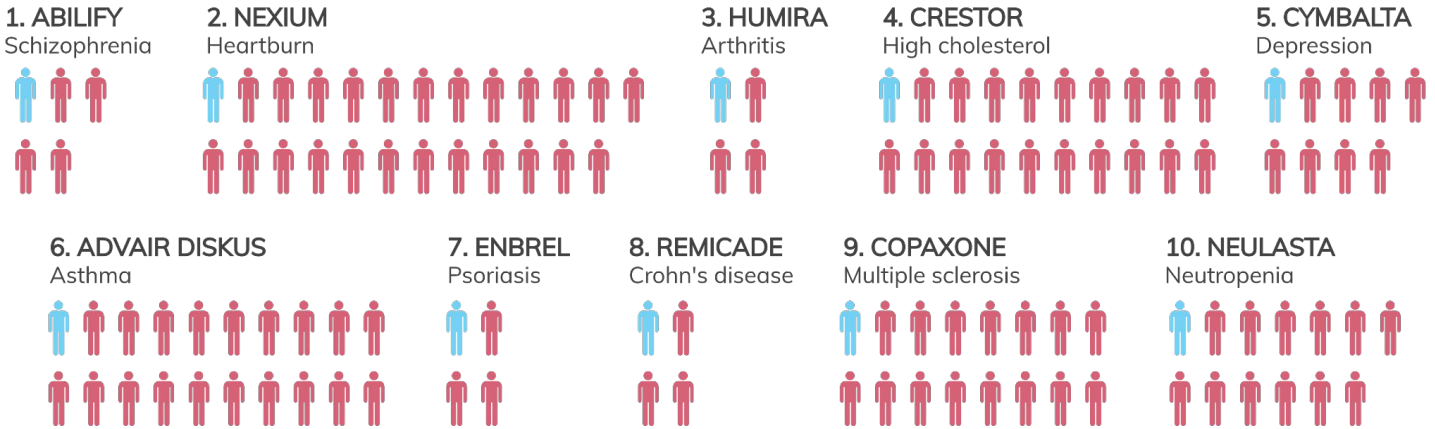
Tertiary Analysis

Genomic data to interpretation: biological insight or clinical conclusions



Genome Data Analysis

Motivation: Precision Medicine



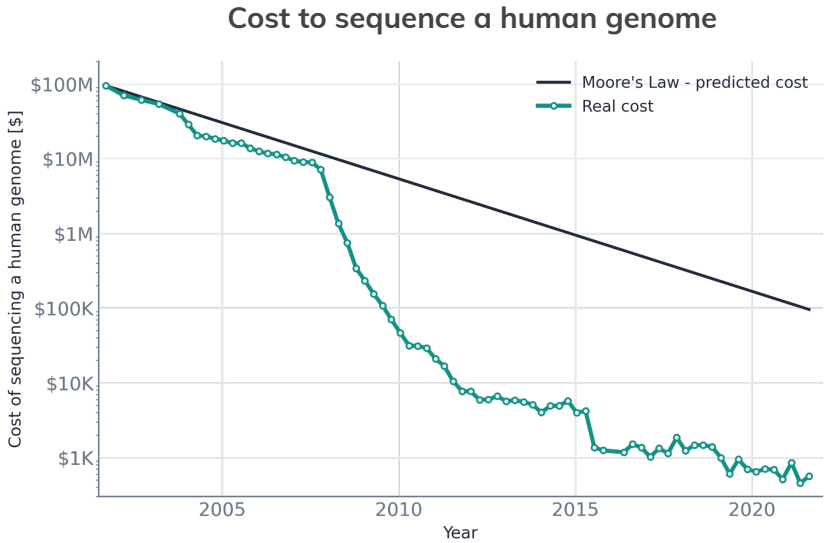
For every person they do help (**blue**), the **ten highest-grossing drugs in the USA fail** to improve the conditions of between 3 and 24 people (**red**)

Schork, Nicholas J.
"Personalized medicine: time for one-person trials." Nature.

Genome Data Analysis

Sequencing Technologies and the Computational Bottleneck

- Sequencing technologies have improved exponentially in the past decade
  - Reducing cost
  - Increasing yield
  - Simplifying methods
- We can sequence a complete individual in 24–48h for $200
  - In 2021: 72h and $1000
- **Current HPC struggles to keep pace** with the constant increase in genomic data to analyze



Contributions

Summary

1 ILVES

Molecular Dynamics

A family of parallel constraint solvers for bond and angle constraints

- Faster convergence than SOTA methods, enabling higher accuracy and stability
- Up to 1.65× simulation throughput using angle constraints

2 GenArchBench

Genome Data Analysis

13 genomics kernels ported and optimized for Arm and RISC-V

- (Micro-)architecture acceleration
- Comprehensive performance characterization
- Architecture-readiness evaluation

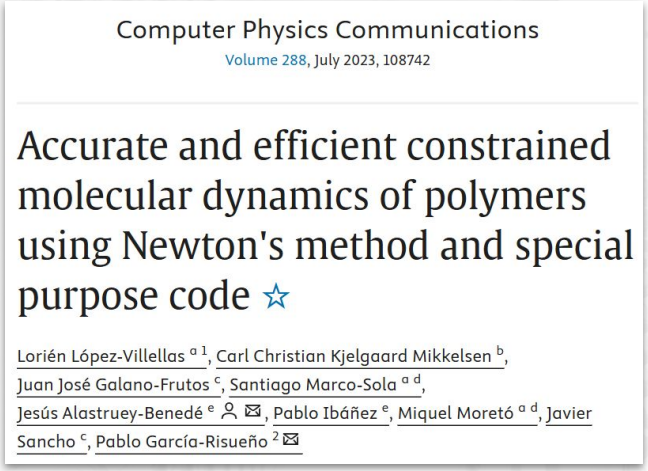
3 Singletrack

Genome Data Analysis

An exact gap-affine backtrace that reduces memory usage and pressure

- Between 2-5× less memory for SOTA algorithms
- Up to 2.1× speedups through better cache locality

Contribution 1: The ILVES Family of Algorithms



Computer Physics Communications

Volume 288, July 2023, 108742

Accurate and efficient constrained molecular dynamics of polymers using Newton's method and special purpose code ☆

Lorién López-Villellas [a 1], Carl Christian Kjelgaard Mikkelsen [b], Juan José Galano-Frutos [c], Santiago Marco-Sola [a d], Jesús Alastruey-Benedé [e], Pablo Ibáñez [e], Miquel Moretó [a d], Javier Sancho [c], Pablo García-Risueño [2]



ILVES: Accurate and Efficient Bond Length and Angle Constraints in Molecular Dynamics

Lorién López-Villellas*, Carl Christian Kjelgaard Mikkelsen, Juan José Galano-Frutos, Santiago Marco-Sola, Jesús Alastruey-Benedé, Pablo Ibáñez, Pablo Echenique, Miquel Moretó, Maria Cristina De Rosa, and Pablo García-Risueño*

Journal of Chemical Theory and Computation

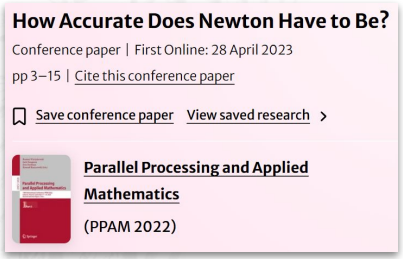
Cite this: *J. Chem. Theory Comput.* 2025, 21, 18, 8711–8719

https://doi.org/10.1021/acs.jctc.5c01376

Published September 4, 2025

Copyright © 2025 The Authors. Published by American Chemical Society. This publication is licensed under CC-BY 4.0 .

Article Views	Altmetric	Citations
4535	199	4



How Accurate Does Newton Have to Be?

Conference paper | First Online: 28 April 2023

pp 3–15 | Cite this conference paper

Save conference paper View saved research

Parallel Processing and Applied Mathematics

(PPAM 2022)



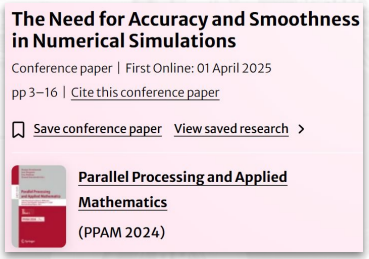
Concurrency and Computation: Practice and Experience / Volume 36, Issue 10 / e7853

SPECIAL ISSUE PAPER | Open Access

Newton's method revisited: How accurate do we have to be?

Carl Christian Kjelgaard Mikkelsen, Lorién López-Villellas, Pablo García-Risueño

First published: 03 July 2023



The Need for Accuracy and Smoothness in Numerical Simulations

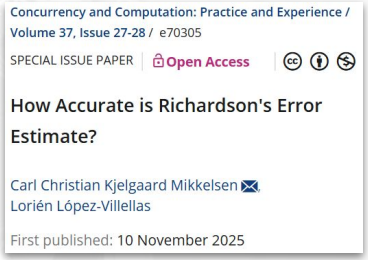
Conference paper | First Online: 01 April 2025

pp 3–16 | Cite this conference paper

Save conference paper View saved research

Parallel Processing and Applied Mathematics

(PPAM 2024)



Concurrency and Computation: Practice and Experience / Volume 37, Issue 27-28 / e70305

SPECIAL ISSUE PAPER | Open Access

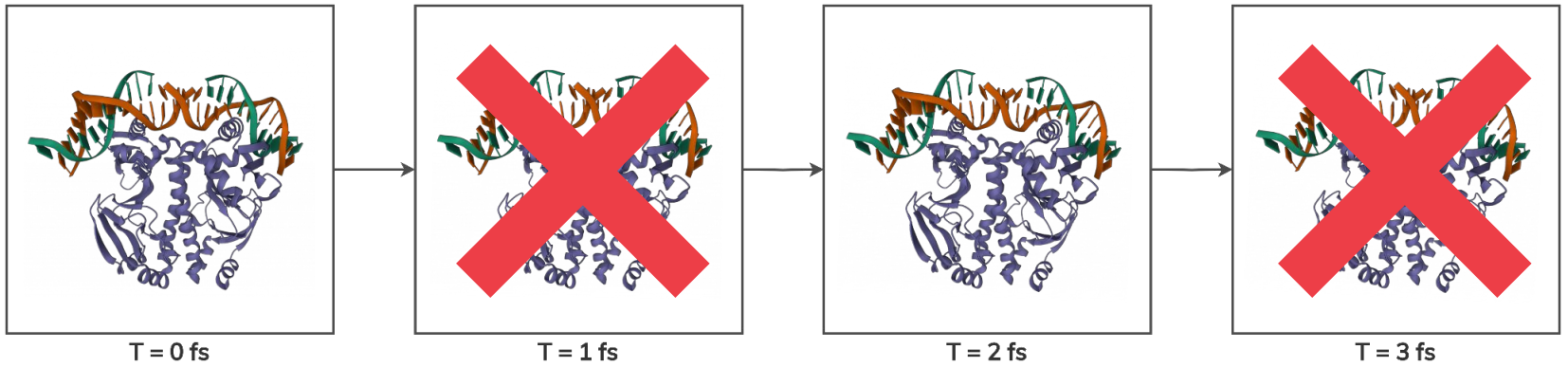
How Accurate is Richardson's Error Estimate?

Carl Christian Kjelgaard Mikkelsen, Lorién López-Villellas

First published: 10 November 2025

Constrained Molecular Dynamics

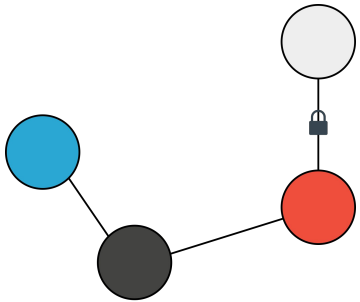
- In most cases, users are not interested in high-frequency vibrations
  - E.g., bond stretching between hydrogen atoms and heavy atoms
- These degrees of freedom set an upper bound for the time step
  - Time step: interval between “images” of the simulation
- They can be removed without affecting the physics by **imposing constraints**



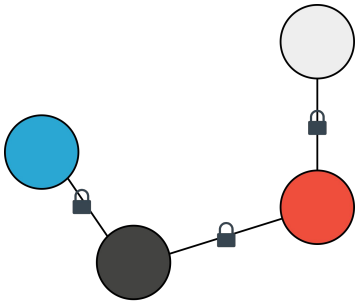
Constrained Molecular Dynamics

Classification of Constraints

Widely Used

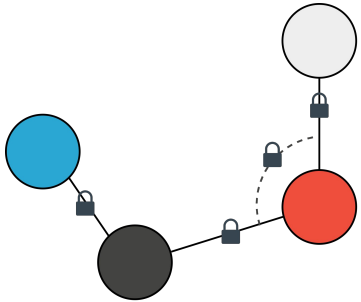


H-bonds
ts = 2 fs

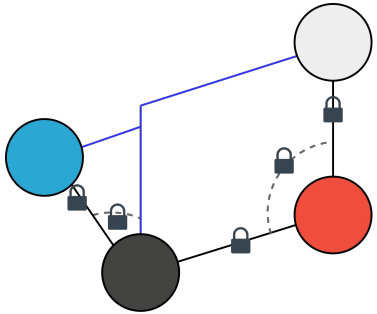


all-bonds
ts = ~2 fs
(enables HMR/VS)

Partially Used / Unexplored



H-angles
ts = ~4 fs

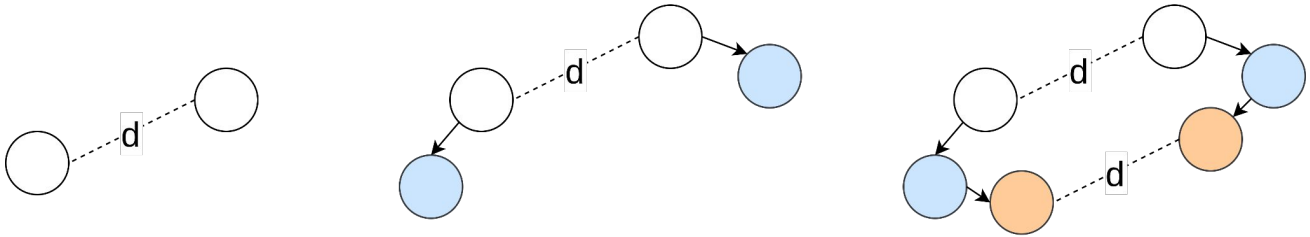


dihedral-angles
ts = ~6-7 fs

Constrained Molecular Dynamics

Imposing Constraints

- Applying constraints involves (numerically) solving a nonlinear system
  - Iteratively solve a linear system until a “good enough” solution is reached
- Imposing constraints inaccurately can result in **unrealistic simulations**



1. Initial State

2. Movement of Atoms due to external forces

3. Imposing Constraints

State-of-the-Art Constraint Solvers

SHAKE 1977

- Implemented in most MD packages
- Nonlinear Gauss-Seidel method
- Slow linear convergence
- Unsuitable for H-angles/dihedral constraints
- No general parallel implementation exists
- MD packages rely on communication-free implementations

LINCS/P-LINCS 1997/2008

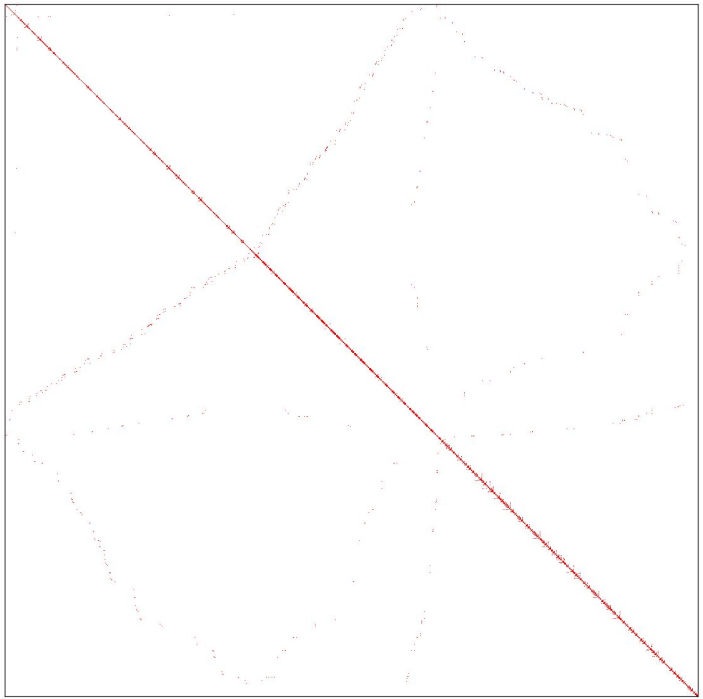
- Implemented in GROMACS
- Truncated Neumann series approximation
- Linear convergence
- Faster than SHAKE in general
- Parallel (P-LINCS)
- Bond-length constraints only (by design)
- Can fail to converge in some cases
- Accumulates rounding errors

The ILVES Constraint Solvers

Idea

- Can we use Newton's method to solve the nonlinear system?
  - **Quadratic convergence!**
- Problem: We need a direct solver with time complexity $O(n^3)$
  - E.g. LU factorization
- Observation: Molecules are intrinsically linear
  - E.g. proteins are linear chains of amino acids
- Direct solvers can be applied in **O(n)** in this case!
  - We need a custom parallel linear solver

Barnase linear system structure



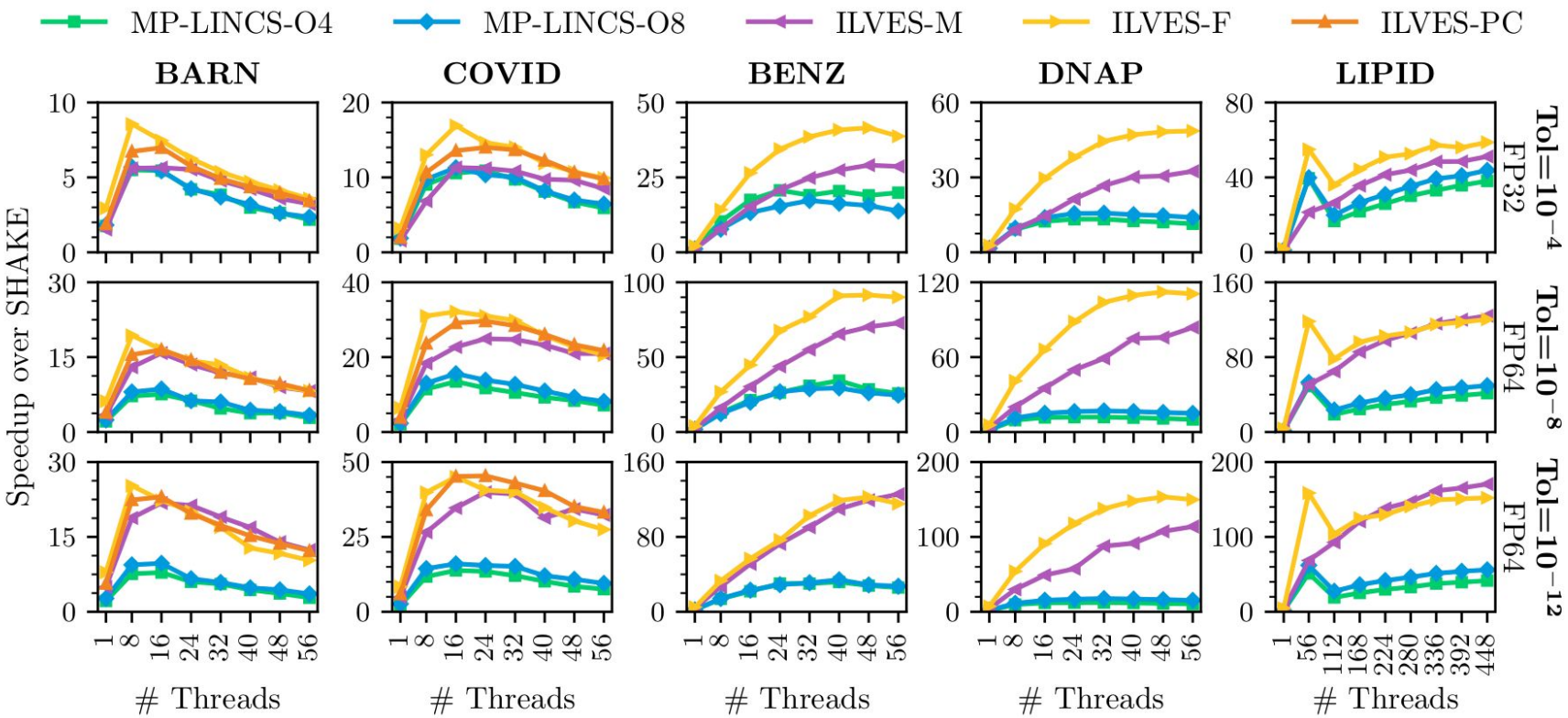
The ILVES Constraint Solvers

Versions

- **ILVES-PC** (***P**eptide **C**hains*):
  - Special-purpose code for the part of the linear system corresponding to each amino acid
  - Programmatic detection of amino acids for partitioning work
  - Parallelization with Schur's complement (OpenMP)
  - Peptide chains, all-bonds, and maintainability is challenging (+40K lines of code)
- **ILVES-M** (***M**ain*):
  - Full generalization of algorithms: custom greedy algorithm for work-partitioning, modified Approximate Minimum Degree (AMD) for minimizing fill-ins, custom parallel LU solve, etc.
  - Any particle/bond constraint (angle constraints through virtual bonds)
- **ILVES-F** (***F**ast*):
  - Reformulation of ILVES-M (ILVES-PC and SHAKE) equations → One LDLT solve per step
  - Quasi-Newton but extremely fast linear convergence (minimum error bounded by data type)
  - ~2× faster than ILVES-M

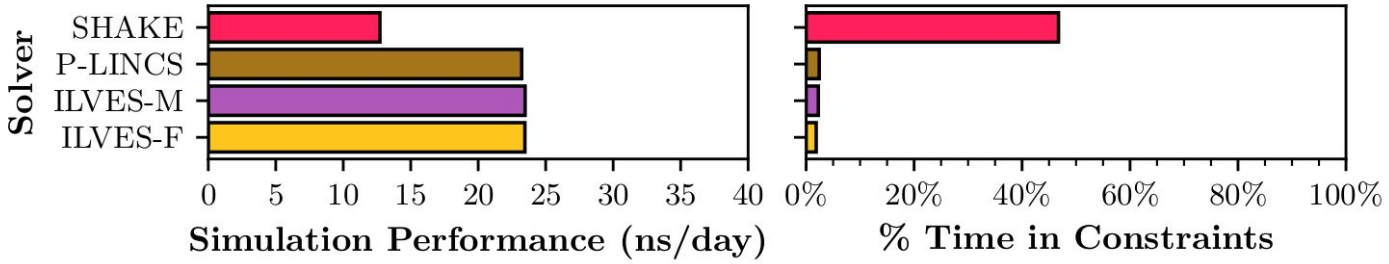
Main Results (GROMACS)

Speedup over SHAKE - Bond Constraints (all-bonds)

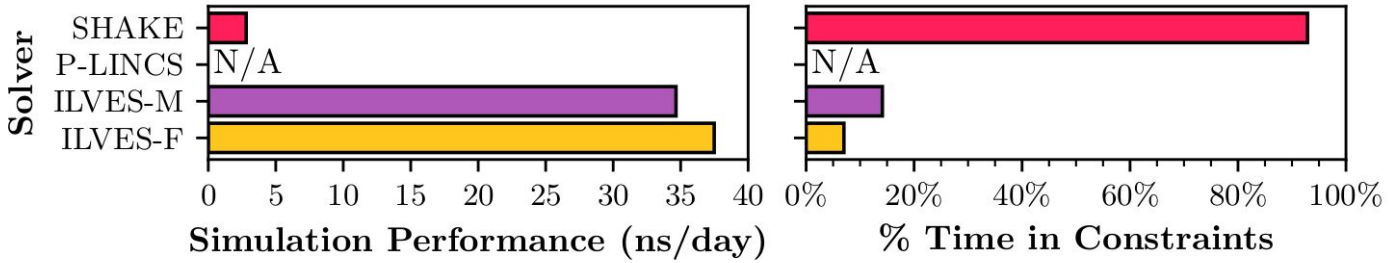


Main Results (GROMACS)

Simulation Throughput - Angle Constraints (H-angles)



(a) H-bonds, 2 fs



(b) H-angles, 3.5 fs

Conclusion

- The ILVES algorithms outperform state-of-the-art algorithms in most simulations when constraining bond lengths (GROMACS)
  - Speedups up to **158× compared to SHAKE** and up to **14× compared to P-LINCS**
  - Equally fast or faster in low-accuracy simulations
  - **More accurate simulations at the same or lower cost** than SOTA
- ILVES enables efficient constraining of additional degrees of freedom
  - Unachievable with state-of-the-art constraint solvers like SHAKE and P-LINCS
  - **1.65× more simulation throughput** using a time step of 3.5 fs and H-angle constraints

Future Work

- Upstream ILVES into main MD packages
  - Already part of LAMMPS, work in progress for NAMD
- Research larger time-steps by constraining additional degrees of freedom (DOF)
  - Stability and performance of simulations constraining additional DOF (DONE; this work)
  - Analytical study of max time-step per constrained DOF (WIP)
  - Validation of candidates against force-field parameterization data (planned)
- GPU acceleration
  - Trivial for H-bonds constraints (1 independent cluster per thread; no communication)
  - Performance bounded by communication for all-bonds or additional DOF
    - ILVES on the CPU and rest on the GPU?
    - New iteration of the algorithm?

Contribution 2: The GenArch Benchmark Suite

ELSEVIER

Future Generation Computer Systems

Volume 157, August 2024, Pages 313-329



GenArchBench: A genomics benchmark suite for arm HPC processors

Lorién López-Villellas [d 1], Rubén Langarita-Benítez [a], Asaf Badouh [a], Víctor Soria-Pardos [a], Quim Aguado-Puig [b], Guillem López-Paradís [a], Max Doblas [a], Javier Setoain [e], Chulho Kim [f], Makoto Ono [g], Adrià Armejach [a c], Santiago Marco-Sola [a c], Jesús Alastruey-Benedé [d], Pablo Ibáñez [d], Miquel Moretó [a c]

Conferences > 2023 38th Conference on Design of Circuits and Integrated Systems (DCIS 2023)

RISC-V for Genome Data Analysis: Opportunities and Challenges

Publisher: IEEE

Cite This

Lorién López-Villellas ; Esteve Pineda-Sánchez ; Asaf Badouh ; Santiago Marco-Sola ; Pablo Ibáñez ; Jesús Alastruey-Benedé ; Miquel Moretó

1 Cites in Paper

206 Full Text Views

Genome Data Analysis CPU-Intensive Pipelines

- Official versions of widely used software are usually CPU-only
  - Examples: minimap2, BWA-MEM, Bowtie2, GATK HaplotypeCaller, Flye, Canu, etc.
- The algorithms are a moving target
  - Short reads (2007-2008) → Long reads (2011-2014) → Pangenomes (2023)
- Some key kernels do not fit accelerators (FPGA, GPU, TPU...)
  - Examples: FM-index, k-mer-counting, De-Bruijn Graph Construction, etc.
  - Irregular and data-dependent accesses
- Genome data analysis kernels are historically optimized for x86

The Emergence of Arm and RISC-V in HPC

2016

arm

- **Mont-Blanc project**
  Arm HPC prototypes
- **Cavium Thunder X**
  First Armv8 server chip

RISC-V

- **RISC-V Foundation**
  Open ISA and first spec
- **Rocket / BOOM**
  Academic prototypes

2026

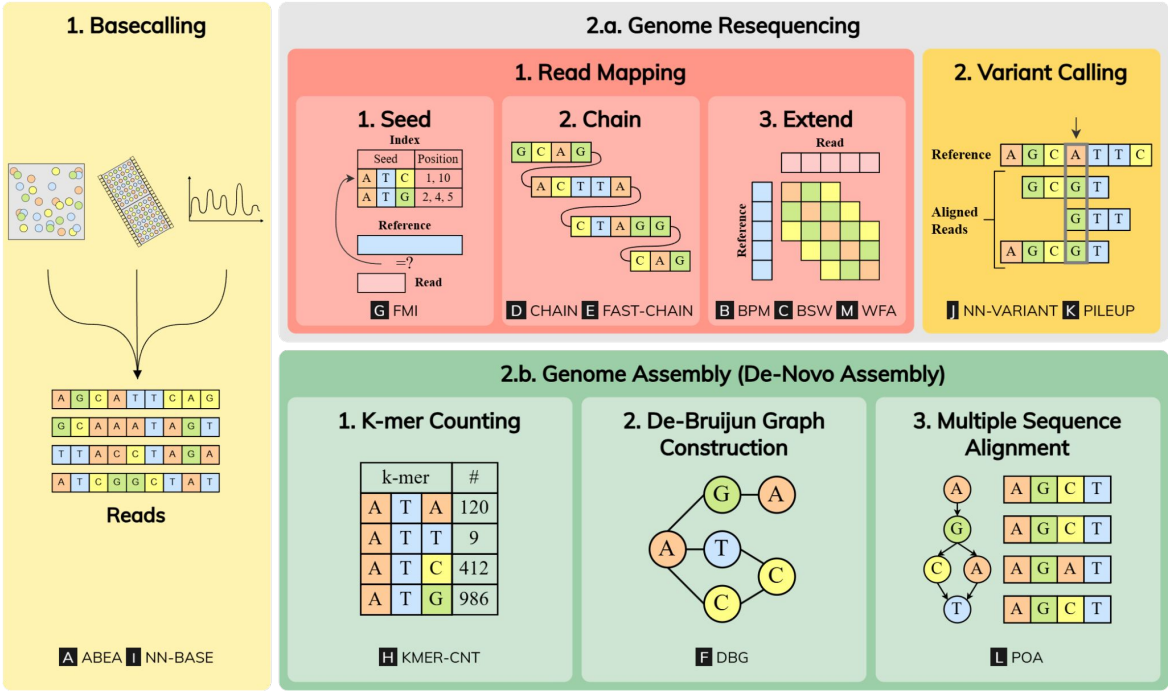
arm

- **Fugaku A64FX**
  Top500#1 (2020-2022)
- **AWS Graviton 1-4**
  Cloud HPC
- **NVIDIA Grace**
  Grace-Hopper Superchip
- **Ampere Altra**
  Azure - Google Cloud
- **SiPearl Rhea**
  JUPITER exascale (exp.)
- **Microsoft Cobalt**
  Azure - Google Cloud

RISC-V

- **Milk-V Pioneer**
  64-core server board
- **XuanTie C910**
  Out-of-order core
- **SiFive Unmatched**
  4-core board (2021)
- **Allwinner D1**
  SIMD core (2021)
- **RVV 1.0 Ratified**
  SIMD ext. (2021)

The GenArch Benchmark Suite



13 Kernels **A-M** from the most widely-used tools

Primary and secondary analysis

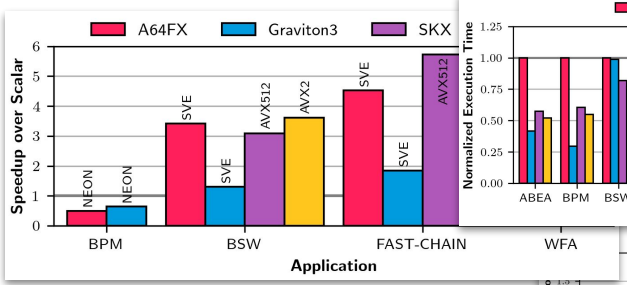
Multi-threaded and **computationally demanding**

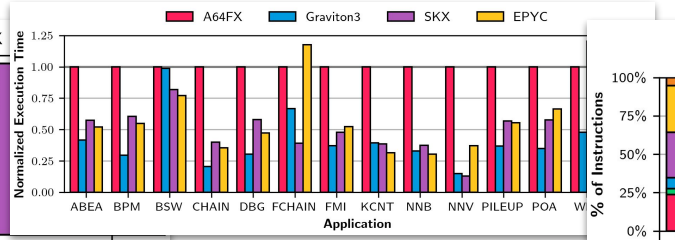
Efficient porting to **arm** and **RISC-V**

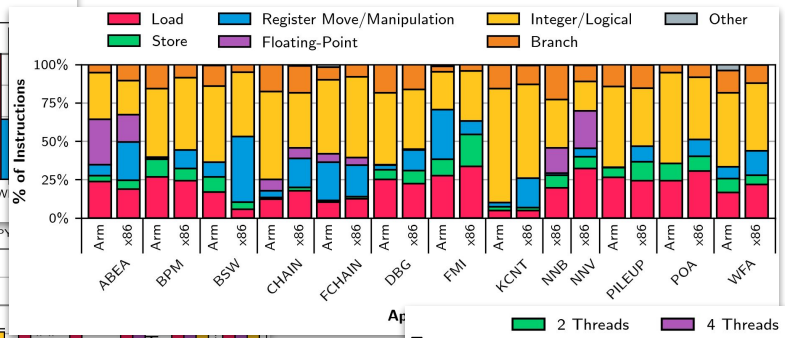
Architecture/ecosystem insights to guide software/CPU design

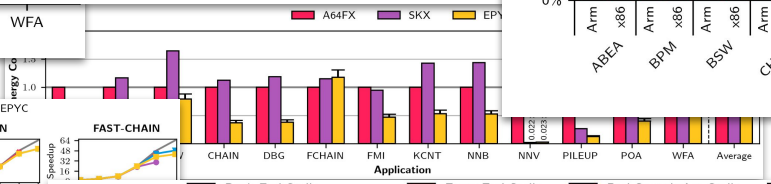
Target Machines

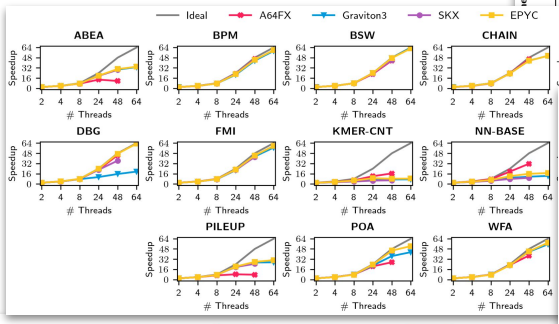
	arm		RISC-V		intel	AMD
	A64FX Fujitsu	Graviton 3 Amazon AWS	SiFive Unmatched	Allwinner D1	SKX 2 x Intel Xeon Platinum 8160	EPYC AMD EPYC 7742
Date	2019	2022	2021	2021	2017	2019
Freq	2.2 GHz	2.6 GHz	1.2 GHz	1 GHz	1-2.1 GHz	1.5-2.25 GHz
Power	120 W	N/A	N/A	N/A	2 x 150 W	225 W
SIMD	NEON/SVE	NEON/SVE	N/A	RVV 0.7	SSE/AVX2/AVX512	SSE/AVX2/AVX512
# Cores	4 x 12	1	4 + 1	1	24	64
Pipeline	out-of-order	out-of-order	in-order	in-order	out-of-order	out-of-order
Commit	4 µOP	N/A	2 instr.	1 instr.	4-8 µOP	8 MOP
L1	64 KB	64 KB	32 KB	32 KB	32 KB	32 KB
L2	—	1 MB	—	—	1 MB	512 KB
LLC	4 x 8 MB	32 MB	2 MB	—	33 MB	16 x 16 MB
Memory	32 GB HBM2	128 GB DDR5	16 GB DDR4	2 GB DDR3	48 GB DDR4	1 TB DDR4

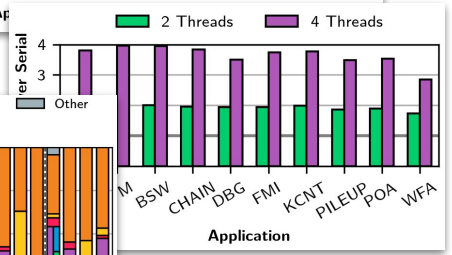

A64FX
Graviton3
SKX
Speedup over Scalar
NEON
SVE
AVX512
AVX2
BPM
BSW
FAST-CHAIN
WFA
Application

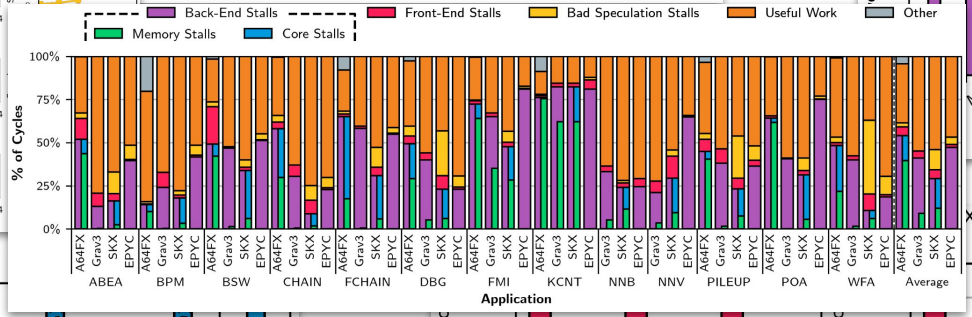

A64FX
Graviton3
SKX
EPYC
Normalized Execution Time
ABEA
BPM
BSW
CHAIN
DBG
FCHAIN
FMI
KCNT
NNB
NNV
PILEUP
POA
Application

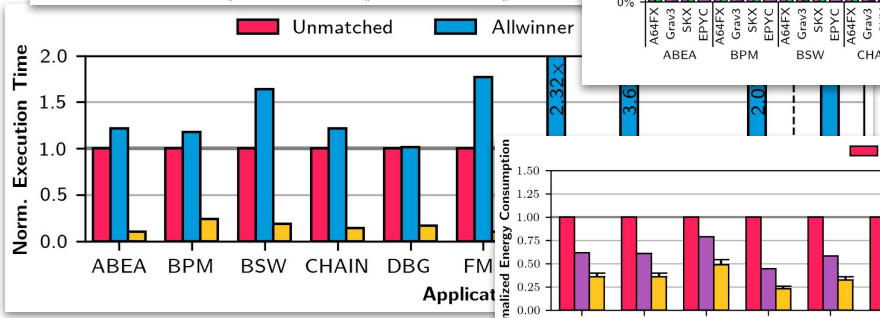

Load
Store
Register Move/Manipulation
Floating-Point
Integer/Logical
Branch
Other
% of Instructions
Arm
x86
ABEA
BPM
BSW
CHAIN
FCHAIN
DBG
FMI
KCNT
NNB
NNV
PILEUP
POA
WFA

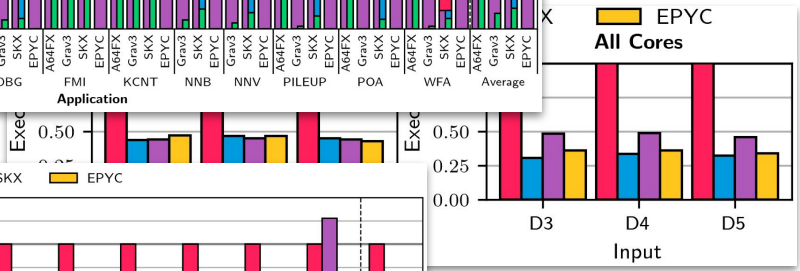

A64FX
SKX
CHAIN
DBG
FCHAIN
FMI
KCNT
NNB
NNV
PILEUP
POA
WFA
Average
Application

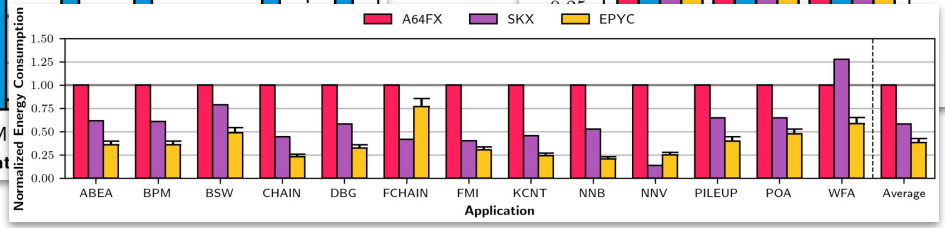

Ideal
A64FX
Graviton3
SKX
EPYC
ABEA
BPM
BSW
CHAIN
FAST-CHAIN
DBG
FMI
KMER-CNT
NN-BASE
PILEUP
POA
WFA
Speedup
Threads


2 Threads
4 Threads
Application


Back-End Stalls
Front-End Stalls
Bad Speculation Stalls
Useful Work
Other
Memory Stalls
Core Stalls
% of Cycles
A64FX
Grav3
SKX
EPYC
ABEA
BPM
BSW
CHAIN
FCHAIN
DBG
FMI
KCNT
NNB
NNV
PILEUP
POA
WFA
Average
Application


Unmatched
Allwinner
Norm. Execution Time
2.32×
3.6
2.0
ABEA
BPM
BSW
CHAIN
DBG


EPYC
All Cores
D3
D4
D5
Input

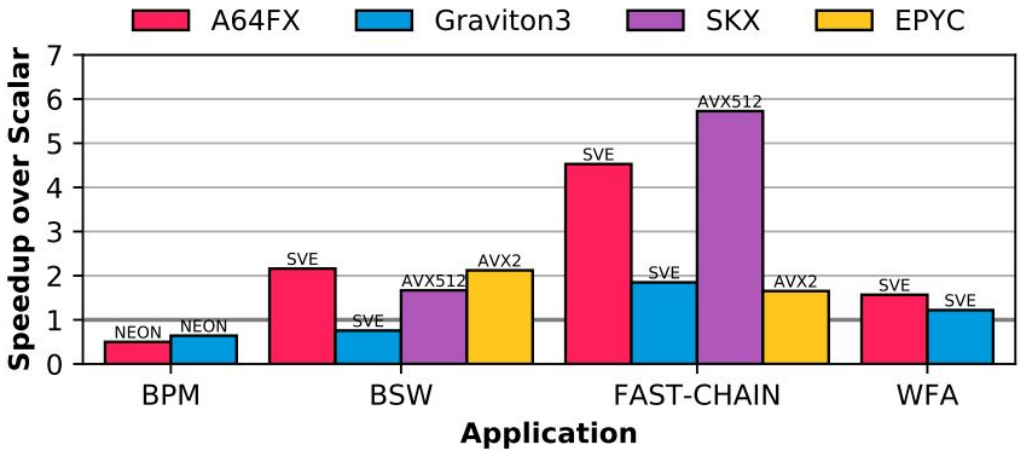

A64FX
SKX
EPYC
Normalized Energy Consumption
ABEA
BPM
BSW
CHAIN
DBG
FCHAIN
FMI
KCNT
NNB
NNV
PILEUP
POA
WFA
Average
Application

The Arm and RISC-V Ecosystem

- Arm is maturing
  - Mainstream toolchain runs natively: GDB, GCC, perf, vendor builds of PyTorch/TensorFlow…
  - Low-level/SVE tooling still lags: DynamoRIO, SIMDe, Valgrind…
  - A64FX relies on custom Fujitsu tools (FAPP) for profiling
  - Graviton3 is closed (scarce counters, no freq/power)
- RISC-V is still immature
  - Core tools only partially ported: Valgrind, GCC, LLVM, GDB, MemorySanitizer…
  - SIMD dev needs QEMU + Vehave + LLVM-MC
  - Libraries often need manual compilation. Unfeasible with heavy deps (PyTorch, TensorFlow)
  - Profiling is extremely limited: Unmatched exposes only cycles/instructions, Allwinner none

Exploiting SIMD in Arm

- 4 kernels migrated to Arm SVE vector-agnostic extension (up to 4.5× speedup)
  - Missing horizontal SIMD instructions for BPM (e.g., bit shift between elements of the vreg)
  - Short loops: vector-length-agnostic removes costly scalar tails
  - Long loops: per-iteration cost outweighs the saved tail



Exploiting SIMD in RISC-V

- Some of the required instructions are not part of RVV
  - `vclz`: Count leading zeros (part of the scalar B extension)
  - `vctz`: Count trailing zeros (part of the scalar B extension)
  - `vreverse`: Reverse the elements of a vector (proposed generalization of `vid`)
- Significant performance loss without them
  - Can be emulated in software with scalar instructions, but kills performance
- Despite supporting RVV-0.7, Allwinner raises an exception when executing SIMD instructions for 64-bit element width
  - Compilers generate these instructions even for 32- or 16-bit elements
- No SIMD kernel was successfully ported to RISC-V

Arm Microarchitectural Insights

- 2 kernels accelerated leveraging Arm performance libraries (up to 6.5× speedup)
- 2 kernels accelerated via μarch tuning and algorithmic optimizations (up to 3×)
- A64FX requires aggressive tuning efforts to exploit performance
  - Energy-efficient at full load at ~2× slowdown compared to Graviton3 and x86 (constrained on OoO resources)
  - Frequent memory stalls even though it implements 4× more memory bandwidth than the other systems (limited memory hierarchy and high latencies)
- Graviton3 is competitive out-of-the-box, no aggressive tuning needed
  - Wide out-of-order pipeline
  - Faster than A64FX, SKX and EPYC in single-thread

RISC-V Microarchitectural Insights

- Allwinner and Unmatched cannot compete with Arm and x86 HPC machines
  - Scalar, in-order, small memory hierarchy, low frequency, etc.
  - Development/test oriented: e.g., Allwinner does not fully support RVV specification
- New-generation RISC-V CPUs are a leap forward
  - Out-of-order cores, many-core server boards, and RVV 1.0 SIMD
  - E.g., XuanTie C910, Milk-V Pioneer
- RISC-V extensibility is ideal for developing new domain-specific extensions and accelerators
  - Genomics and also molecular dynamics

Conclusion

- Porting and acceleration of 13 widely used genomics kernels to Arm and RISC-V
  - Exploiting SIMD when possible
- Deep evaluation of Arm, RISC-V and x86 machines
  - Graviton3 competes with x86
  - A64FX and RISC-V are behind (OoO resources and memory hierarchy)
- SIMD intrinsics are not good friends with architecture heterogeneity
  - Standardized abstraction layer? C++ Data-parallel types (SIMD) (C++26)?
  - More powerful and general auto-vectorization? OpenMP simd?
- Hidden performance counters hinder optimizations (kudos to Fujitsu and Intel)
  - Not only missing counters but also bad documentation
  - I believe this will be extremely important in the Agentic AI era

Future Work

- Update GenArchBench with new software and evaluation on new hardware
  - Add new applications (e.g., pangenomics)
  - Reevaluation on modern machines (especially interesting for RISC-V)
- Further acceleration of kernels
  - Some kernels remain non-vectorized
- Multiple offshoot lines of work related to performance characterization
  - Community-driven database for performance counters
  - Analysis of agentic AI software optimization performance under different architectures (and different performance counters/documentation detail)
  - Improve microarchitecture characterization methodology

Contribution 3: The Singletrack Backtrace Algorithm

JOURNAL ARTICLE

Singletrack: an algorithm for improving memory consumption and performance of gap-affine sequence alignment

Lorién López-Villellas ✉, Cristian Iñiguez, Albert Jiménez-Blanco, Quim Aguado-Puig, Miquel Moretó, Jesús Alastruey-Benedé, Pablo Ibáñez, Santiago Marco-Sola

Bioinformatics, Volume 42, Issue 5, May 2026, btag183,
https://doi.org/10.1093/bioinformatics/btag183
Published: 13 April 2026 **Article history** ▾

JOURNAL ARTICLE ACCEPTED MANUSCRIPT

Theseus: Fast and Optimal Affine-Gap Sequence-to-Graph Alignment

Albert Jiménez-Blanco ✉, Lorién López-Villellas, Juan Carlos Moure, Miquel Moreto, Santiago Marco-Sola

Bioinformatics, btag561, https://doi.org/10.1093/bioinformatics/btag561
Published: 27 July 2026 **Article history** ▾

Sequence Alignment

Introduction

- Sequence alignment is one of the **most time-consuming** steps in genomics
  - Read mapping, variant calling...
- Given target (t) of size m, and query (q) of size n, arrange q and t to minimize a given cost function
  - Adding gaps to the sequences
- Aligning two sequences of 100K → 40 GB of memory



Sequence Alignment

Alignment Phase

- Algorithms are based on **dynamic programming** (DP)
  - Compute a matrix M of size ≈ n × m
  - Each cell contains partial alignments between prefixes
- Recurrence relations to compute each cell

	-	G	C	C	A	A
-	0	2	4	6	8	10
G	2	0	2	4	6	8
C	4	2	0	2	4	6
A	6	4	2	2	4	

$$M_{i,j} = \min \begin{cases} M_{i,j-1} + e \\ M_{i-1,j} + e \\ M_{i-1,j-1} + S(i,j) \end{cases}$$

Min{4 + match, 6 + gap, 4 + gap}

Sequence Alignment

Alignment Phase

- Algorithms are based on **dynamic programming** (DP)
  - Compute a matrix M of size ≈ n × m
  - Each cell contains partial alignments between prefixes
- Recurrence relations to compute each cell

	-	G	C	C	A	A
-	0	2	4	6	8	10
G	2	0	2	4	6	8
C	4	2	0	2	4	6
A	6	4	2	2	4	4

$$M_{i,j} = \min \begin{cases} M_{i,j-1} + e \\ M_{i-1,j} + e \\ M_{i-1,j-1} + S(i,j) \end{cases}$$

Sequence Alignment

Backtrace

- Given the computed matrices, follow the path that led to $M_{n,m}$ to obtain the alignment
  - **Linear algorithm**
  - Tiny part of the total execution time

M

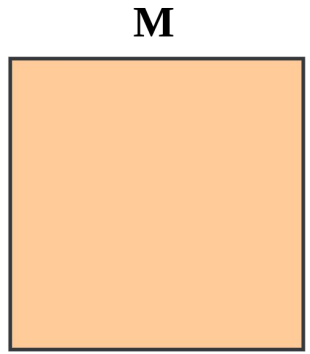
		- (j=0)	G (j=1)	C (j=2)	C (j=3)	A (j=4)	A (j=5)
-	i=0	0	2	4	6	8	10
G	i=1	2	0	2	4	6	8
C	i=2	4	2	0	2	4	6
A	i=3	6	4	2	2	4	4

$$M_{i,j} = \min \begin{cases} M_{i,j-1} + e \\ M_{i-1,j} + e \\ M_{i-1,j-1} + S(i,j) \end{cases}$$

Sequence Alignment

Score Functions

- **Gap-Linear (1970):** match, mismatch or gap
  - one n x m matrix
- Gap-Affine (1981): add a gap opening cost
  - Prefer long gaps over short gaps
  - Better captures evolutionary events (long gaps)
  - three n x m matrices for the recurrences
- Dual Gap-Affine (1990): two types of gaps
  - Cheap opening and expensive extension
  - Expensive opening and cheap extension
  - Favors either short or long gaps
  - Sequencing errors result in short gaps
  - five n x m matrices for the recurrences

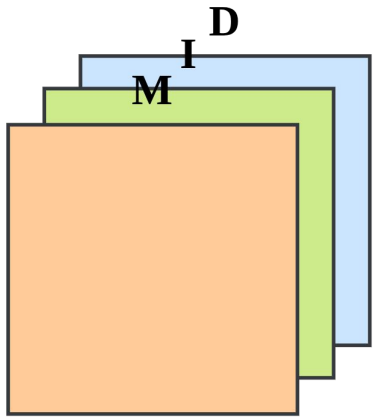


CA-C-TGTC-AA
CAACCT-TCCAA

Sequence Alignment

Score Functions

- Gap-Linear (1970): match, mismatch or gap
  - one n x m matrix
- **Gap-Affine (1981):** add a gap opening cost
  - Prefer long gaps over short gaps
  - Better captures evolutionary events (long gaps)
  - three n x m matrices for the recurrences
- Dual Gap-Affine (1990): two types of gaps
  - Cheap opening and expensive extension
  - Expensive opening and cheap extension
  - Favors either short or long gaps
  - Sequencing errors result in short gaps
  - five n x m matrices for the recurrences



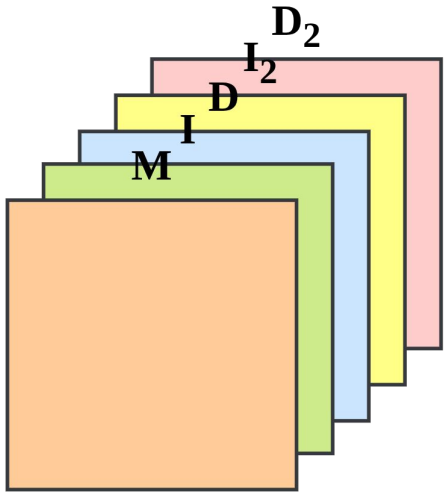
GCTATAGAG

GCTA----G

Sequence Alignment

Score Functions

- Gap-Linear (1970): match, mismatch or gap
  - one n x m matrix
- Gap-Affine (1981): add a gap opening cost
  - Prefer long gaps over short gaps
  - Better captures evolutionary events (long gaps)
  - three n x m matrices for the recurrences
- **Dual Gap-Affine (1990):** two types of gaps
  - Cheap opening and expensive extension
  - Expensive opening and cheap extension
  - Favors either short or long gaps
  - Sequencing errors result in short gaps
  - five n x m matrices for the recurrences



GCTACA-----A

G-TACACAATAA

Singletrack

Idea

During gap-affine backtrace...

1. If the current cell is in I (or D)
   - Only two possible previous cells: I or M
   - Always check against M
   - If we did not come from M, the correct path is the other one (I)

$$I_{i,j} = \min \begin{cases} I_{i,j-1} + e \\ M_{i,j-1} + o + e \end{cases}$$

$$D_{i,j} = \min \begin{cases} D_{i-1,j} + e \\ M_{i-1,j} + o + e \} \end{cases}$$

$$M_{i,j} = \min \begin{cases} I_{i,j} \\ D_{i,j} \\ M_{i-1,j-1} + S(i,j) \end{cases}$$

Singletrack

Idea

During backtrace...

2. If the current cell is in **M**
   - Three possible previous cells: **I**, **D**, **M**
   - If the previous cell is **M**, continue
   - Otherwise, open 2 tentative paths, one assuming we came from **I** and the other from **D** (state 1.)
   - Stepwise follow the two paths: eventually one will find a matching **M** cell → commit path and go to 2.
   - **We have never accessed a cell of I or D...**
   - **Only store M!**

$$I_{i,j} = \min \begin{cases} I_{i,j-1} + e \\ M_{i,j-1} + o + e \end{cases}$$

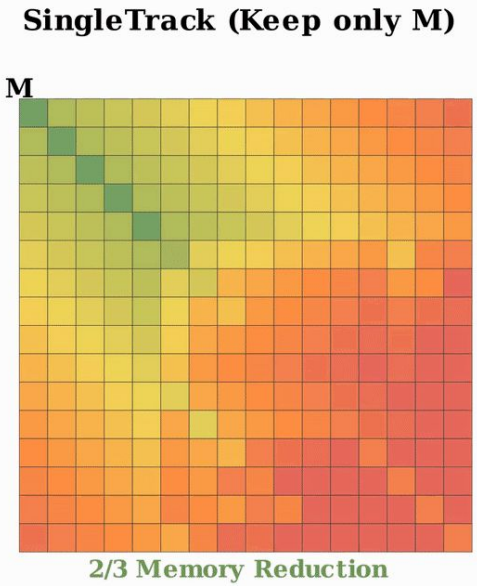
$$D_{i,j} = \min \begin{cases} D_{i-1,j} + e \\ M_{i-1,j} + o + e \} \end{cases}$$

$$M_{i,j} = \min \begin{cases} I_{i,j} \\ D_{i,j} \\ M_{i-1,j-1} + S(i,j) \end{cases}$$

Singletrack

Implications

- Analogous for dual gap-affine: four tentative paths
- Singletrack maintains linear execution time
  - Up to 2× more cells explored when many gaps
  - Cheap since the backtrace is a very cheap operation
- **Memory consumption reduction** of 3× for gap-affine and 5× for dual gap-affine
  - Equivalent to gap-linear alignments
- **Enables performance optimizations**
  - Fewer cache misses and fewer memory accesses (e.g., the single cell of I can be stored in a register)



Main Results

Single-Threaded Performance and Memory Consumption

		Time (µs) / Kbp			Mem (MiB)		
		Illumina	PacBio HF	PromethION	Illumina	PacBio HF	PromethION
aff	KSW2	65.8	2,980.6	7,849.8	**3.8**	10,463.6	4,534.5
	KSW2+ST	60.8	3,634.5	10,063.8	4.0	21,864.9	9,065.2
	WFA	4.8	13.3	7,447.2	5.3	427,9	6,337.0
	WFA+ST	**3.4**	**10.9**	**5,233.2**	26.0	**179.5**	**2,123.8**
2aff	KSW2	110.5	4,887.9	12,479.2	**3.8**	10,463.6	4,537.9
	KSW2+ST	77.7	4,108.7	**11,472.8**	4.0	21,864.9	9,066.2
	WFA	11.3	50.9	44,884.7	6.8	2,557.0	37,653.1
	WFA+ST	**5.4**	**39.7**	28,782.6	62.4	**543.8**	**7,546.1**

Conclusion

- Sequence alignment is one of the **main bottlenecks** of genomics pipelines
- Significantly **constrained in memory**
  - Pangenomes becoming increasingly common, which require even more memory
- Singletrack is a general technique to virtually achieve the same memory consumption in gap-affine and dual gap-affine as in gap-linear
  - Can be applied to any sequence alignment algorithm
  - Extremely easy to implement or integrate into SOTA algorithms
  - It enables performance improvements
- KSW2+ST and WFA+ST:
  - Consistent speedups over the original versions
  - Trade-offs to reduce memory requirements compared to SOTA techniques

Future Work

- Good fit for sequence-to-graph (pangenome) alignment
  - Memory usage explosion
  - Not all sequence-to-sequence memory reduction techniques are transferable
  - Theseus: fast and optimal affine-gap sequence-to-graph alignment (Bioinformatics, 2026)
- Also a good fit for accelerators (GPUs, FPGAs, ASICs, etc.)
  - Usually more constrained in memory than the CPU
  - Alignment phase remains unchanged, no new control logic added

Wrap Up

Wrap Up

- **ILVES**: constraint solver that enables accurate simulations and angle constraints
  - Exploit input characteristics (molecules) in custom parallel linear solver
  - Exploit hardware-precision floor (ILVES-F)
  - Distributed-memory parallelism, shared-memory parallelism and vectorization
- **GenArchBench**: genomics benchmark suite for Arm and RISC-V
  - Exploit machine-specific characteristics to accelerate widely-used genomics kernels
  - Deep μarch characterization to guide software optimization and hardware/ISA design
- **Singletrack**: backtrace technique that reduces memory usage and pressure
  - Exploit equations to remove memory requirements
  - Trade backtrace overhead for overall performance gain by reducing memory pressure



Hardware-Aware Algorithms for Efficient Molecular Dynamics and Genome Data Analysis

Author

Lorién López Villellas

Supervisors

Jesús Alastruey Benedé and Pablo Ibáñez Marín



Universidad Zaragoza



Barcelona Supercomputing Center
Centro Nacional de Supercomputación