Efficient all-against-all protein similarity matrix computation using OpenCL

Genome-oriented bioinformatics lab - WS2013/2014

Uli Köhler & Anton Smirnov

LMU & TUM
Helmholtz-Zentrum München

Supervisor: Mathias Walter

February 24th, 2014

SIMAP I

Similarity Matrix of Proteins:

- Database of protein similarities
- Compares all-against-all
- Currently ~73 million protein sequences
  $\rightarrow 5.3 \cdot 10^{15}$ alignments
- BOINC-SIMAP: *distributed computing*

	p_1	p_2	p_3
p_1	—	5	$\cdots$
p_2	$\cdots$	—	$\cdots$
p_3	$\cdots$	170	—

SIMAP II

- Currently uses *FASTA* algorithm (fast, but suboptimal heuristics)
- For high-scoring hits, Smith-Waterman is currently in use
- Smith-Waterman provides better accuracy
- Requires efficient, parallelized implementation

Computational hardware

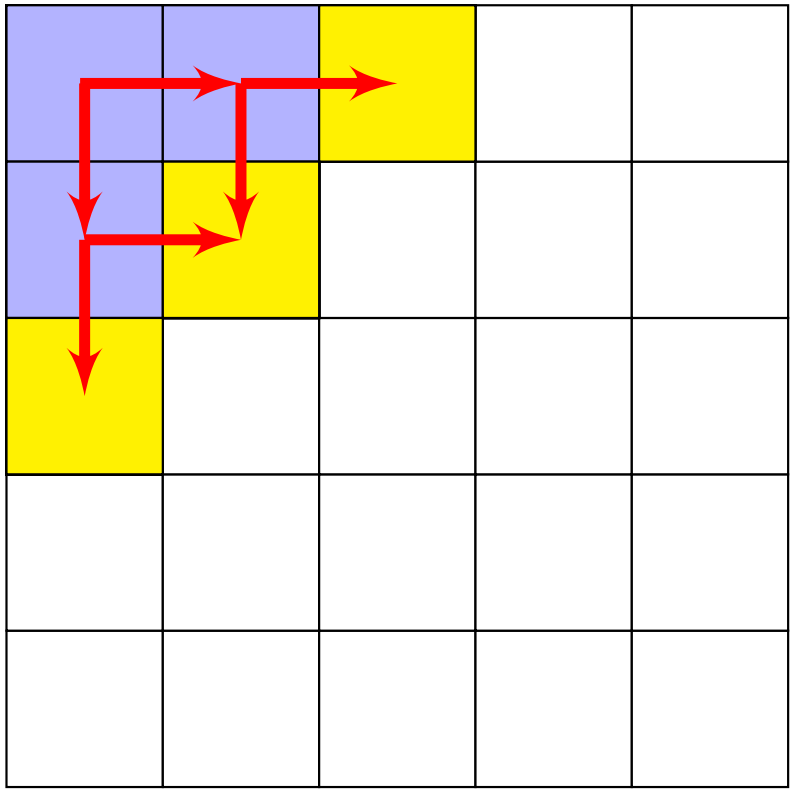
- CPU: ~1-12 cores, available anywhere
- GPU: 1000+ cores, good availability
- FPGA (*field programmable gate array*)
  - Configurable number of cores
  - Difficult to use
  - Expensive

OpenCL

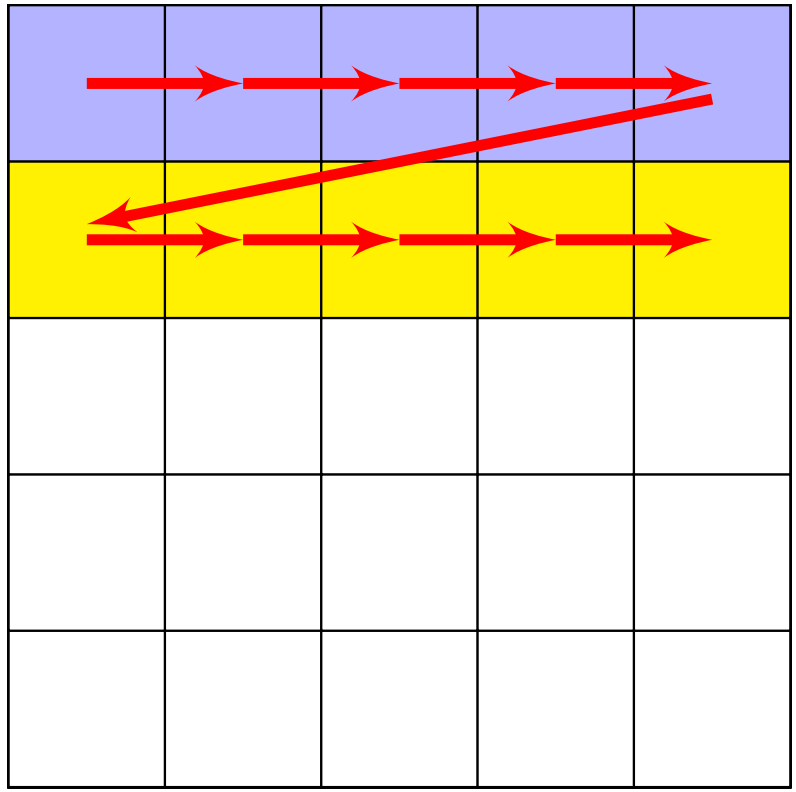
- Programming framework for parallel computing
- Top level abstraction for low level routines
- Runs on CPUs, GPUs & FPGAs without modification
- Driver optimizes code for specific devices

Smith-Waterman parallelization

Intra-task



Inter-task



Sequence length optimization

Maximal efficiency of Smith-Waterman implementation:

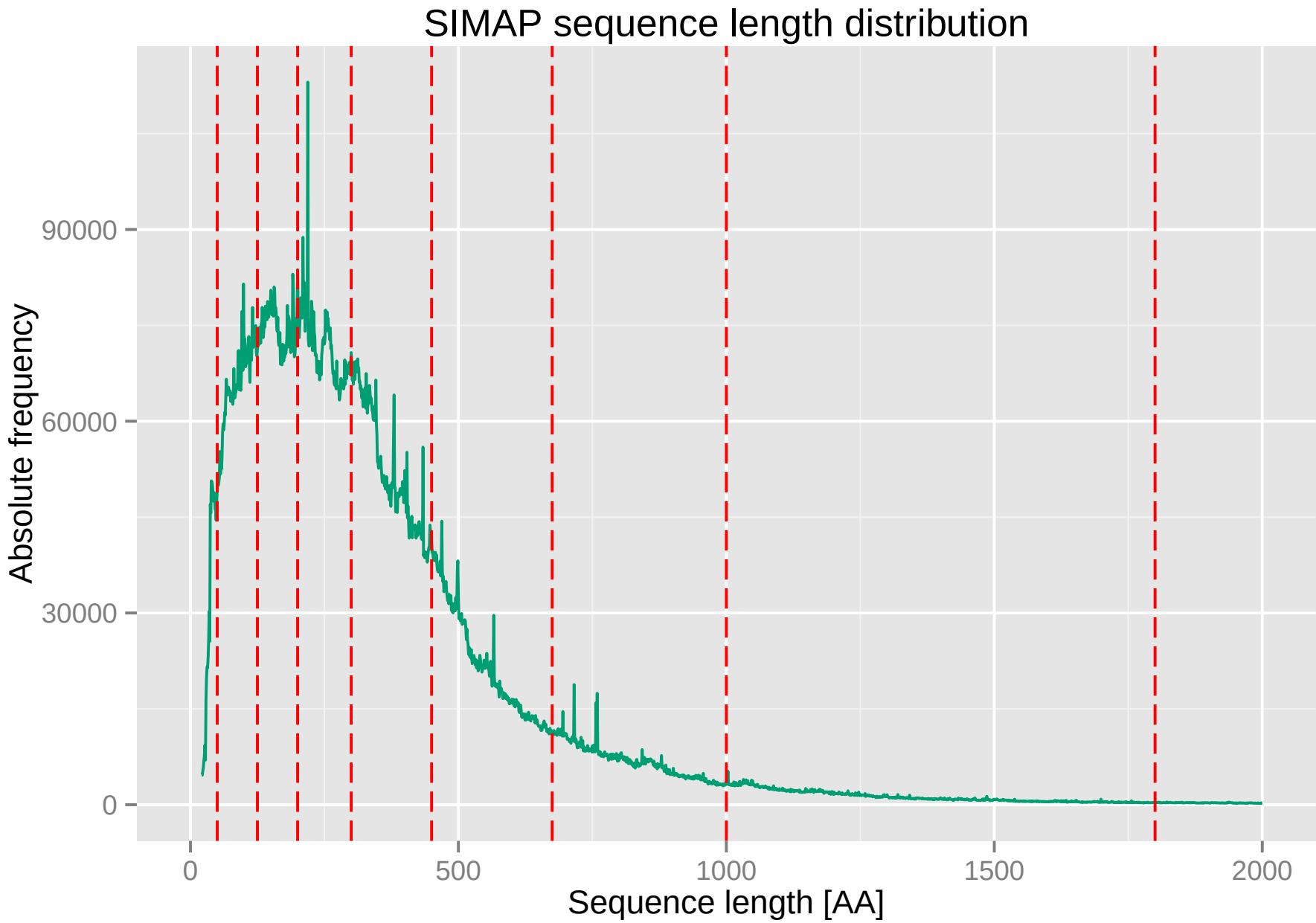
- For many optimizations, we need sequences with equal length
- Equal length can boost performance by multiple magnitudes
- Pad sequence with ε
- Alignment score must not change
  $\rightarrow$ Substitution score: $-\infty$
- Problem: Padding increases matrix size
  $\rightarrow$ Large overhead

Sizeclasses

Solution: Extension sizeclasses / Adaptive binning

- Divide sequence length into different classes
- Pad only within one sizeclass
- Multiple sizeclasses reduce overall padding

	A	K	L	ε	ε
A	$\ddots$	$\ddots$	$\ddots$	0	0
C	$\ddots$	$\ddots$	$\ddots$	0	0
M	$\ddots$	$\ddots$	$\ddots$	0	0
M	$\ddots$	$\ddots$	$\ddots$	0	0
L	$\ddots$	$\ddots$	$\ddots$	0	0


SIMAP sequence length distribution
Absolute frequency
90000
60000
30000
0
0
500
1000
1500
2000
Sequence length [AA]

CLSW: OpenCL Smith-Waterman

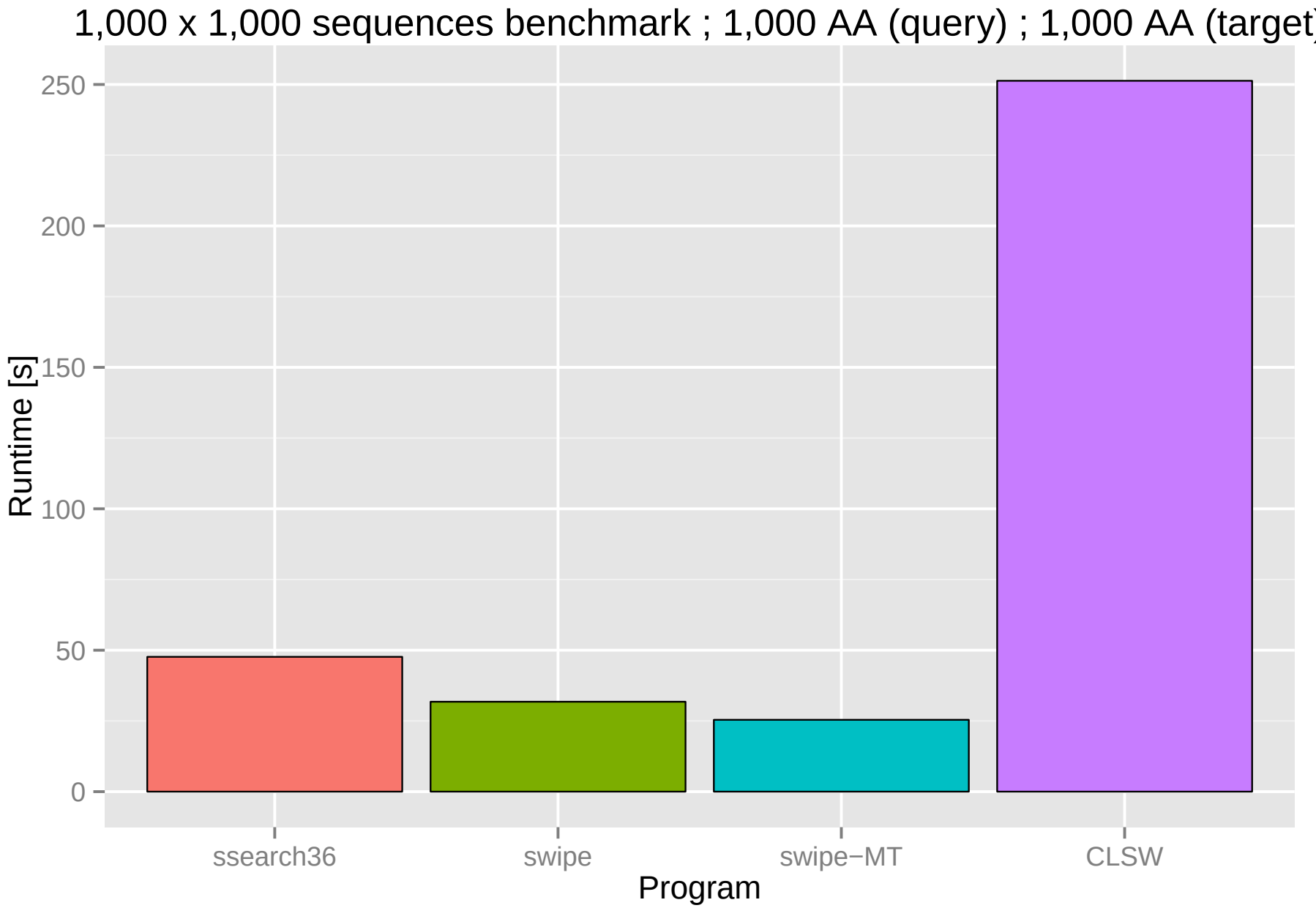
- Objective: Develop proof-of-concept score-only OpenCL Smith-Waterman
- Use inter-task parallelization
- All-against-all with affine gap costs
- Can be used to build vendor-independent fast Smith-Waterman implementation

Implementation aspects

- Written in pure C++11 & OpenCL 1.1
- No external dependencies, compact binary
- Tested with SIMAP subset
- Verified using *SeqAn* library

Core advantages

- *SWIPE*: Integer $\leftrightarrow$ *CLSW*: Floating point
  $\rightarrow$ Composition based score adjustment
  $\rightarrow$ Higher accuracy
- Concise codebase:
  $<$ 1,000 C++ lines of code
  OpenCL Smith-Waterman: $<$50 lines of code
  (SWIPE: 10,000 lines of code)
- Existing implementations are based on CUDA
  $\rightarrow$ Only runs on NVidia GPUs


1,000 x 1,000 sequences benchmark ; 1,000 AA (query) ; 1,000 AA (target)
Runtime [s]
250
200
150
100
50
0
ssearch36
swipe
swipe−MT
CLSW
Program


4000x1000 sequences benchmark, 20 AA (query), 1.000 AA (target)
Runtime [s]
0
20
40
60
ssearch36
swipe
swipe−MT
CLSW
Program

Integration into SIMAP

- Since 2005, only CPU clients
- Since 2014, also ARM client for Android
- Users ask for GPU clients regularly since 2005
- CLSW was built to be integratable into BOINC $\rightarrow$ Leverage huge amount of computing power
- Still, a lot of work needs to be done...

Other uses

- 3-4 times faster than SWIPE for short query sequences
  $\rightarrow$ Shotgun proteomics, NGS?
- Huge optimization potential
  $\rightarrow$ Reduce overhead, 5-10x speedup
- Platforms unsupported by SWIPE (e.g. 32 bit platforms)

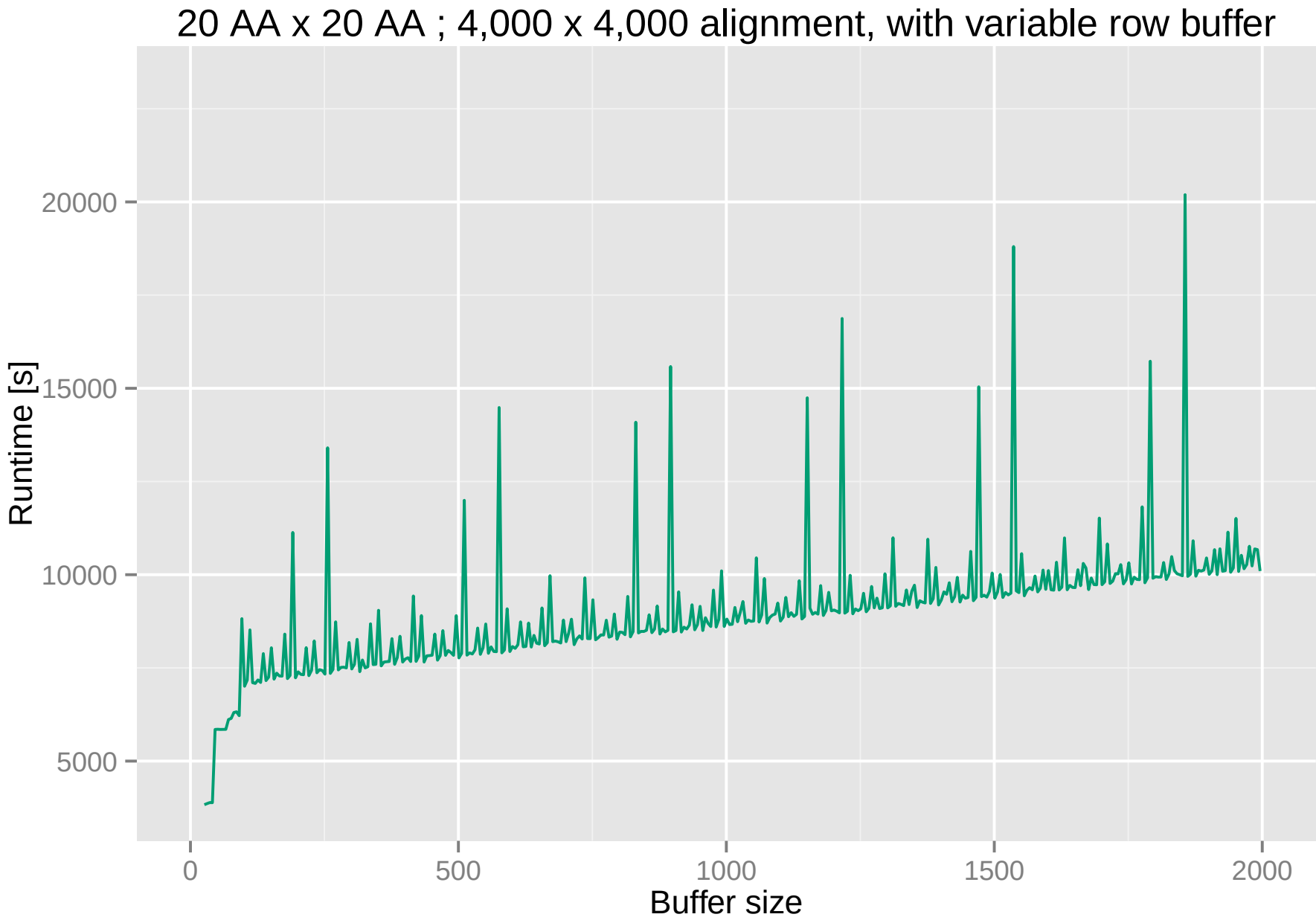
Conclusion

- CLSW: Portable, GPU-based Smith-Waterman
- Fast for small queries, can be optimized for large queries
- Floating point score calculation
  $\rightarrow$ Composition-based score adjustment
- GPU computing is underestimated in computational biology

Thank you for your attention!

Special thanks to Mathias Walter & Thomas Rattei who made this project possible!

Questions?


20 AA x 20 AA ; 4,000 x 4,000 alignment, with variable row buffer
Runtime [s]
20000
15000
10000
5000
0
500
1000
1500
2000
Buffer size

- Sizeclass: $(\alpha \cdot \sum \textit{sizeclass penalty}) + (\beta \cdot |\textit{sizeclass}|)$
- Difficult to determine optimal values for α and β
- Idea: Use population quantiles (e.g. $q_{0.01\%}$ to $q_{100\%}$) as sizeclass boundaries.
- Postprocessing:
  Divide sizeclasses with penalty $>$ threshold