AI-Assisted Design Space Analysis of High-Performance Arm Processors

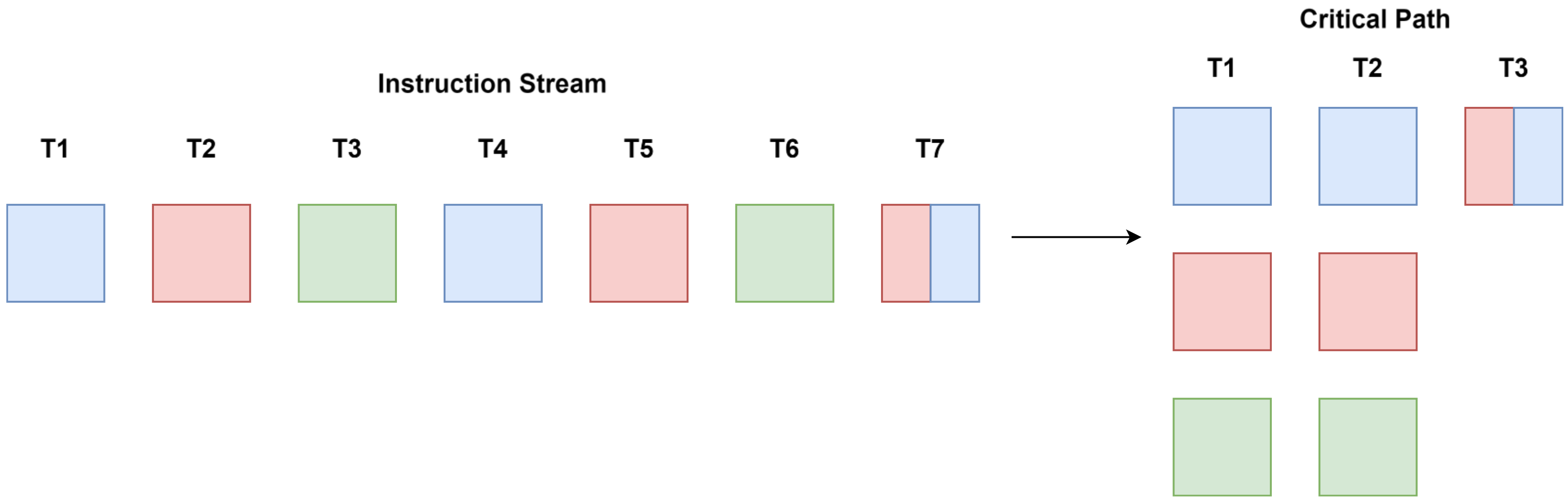
18th Nov 2024

Joseph Moore
Tom Deakin
Simon McIntosh-Smith

University of Bristol High Performance Computing Group

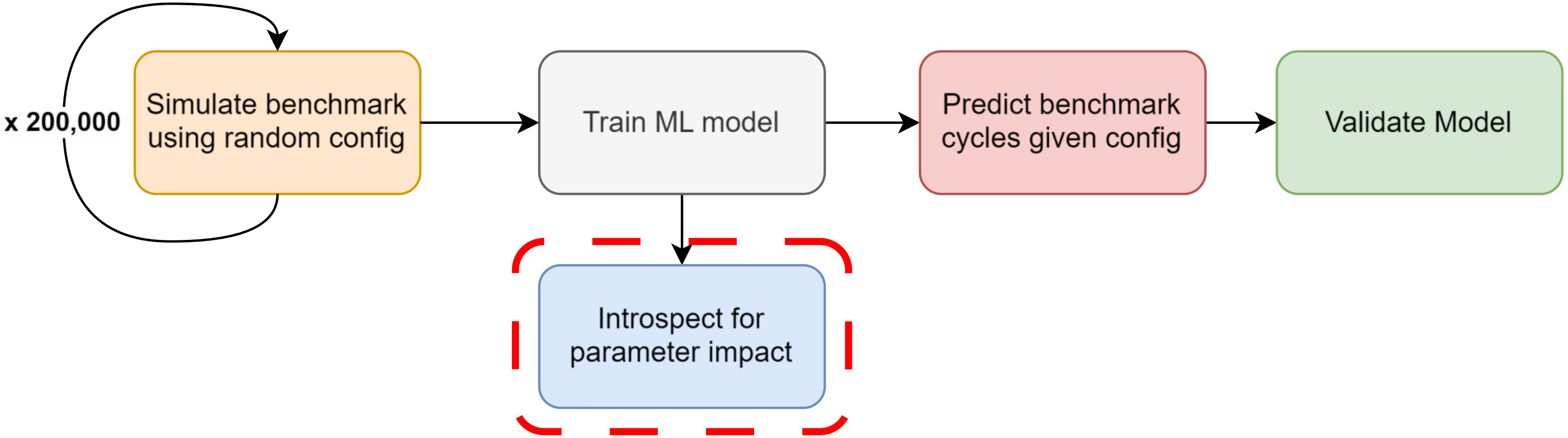
What’s the limit of a CPU?

- Consider the trace of executed instructions
  - Critical path = longest chain of data dependent instructions
- Cycles on “perfect” CPU = latency of critical path
  - Bottleneck is the program!
- How can hardware converge to this?



What this work does

- **Quantifies impact** of single-core bottlenecks
- **Predicts no. cycles** in known HPC applications for different hardware configurations
- Does so through **Machine Learning**
- Learn what the model learns – **how do parameters influence cycles**?


x 200,000
Simulate benchmark using random config
Train ML model
Predict benchmark cycles given config
Validate Model
Introspect for parameter impact

Related work

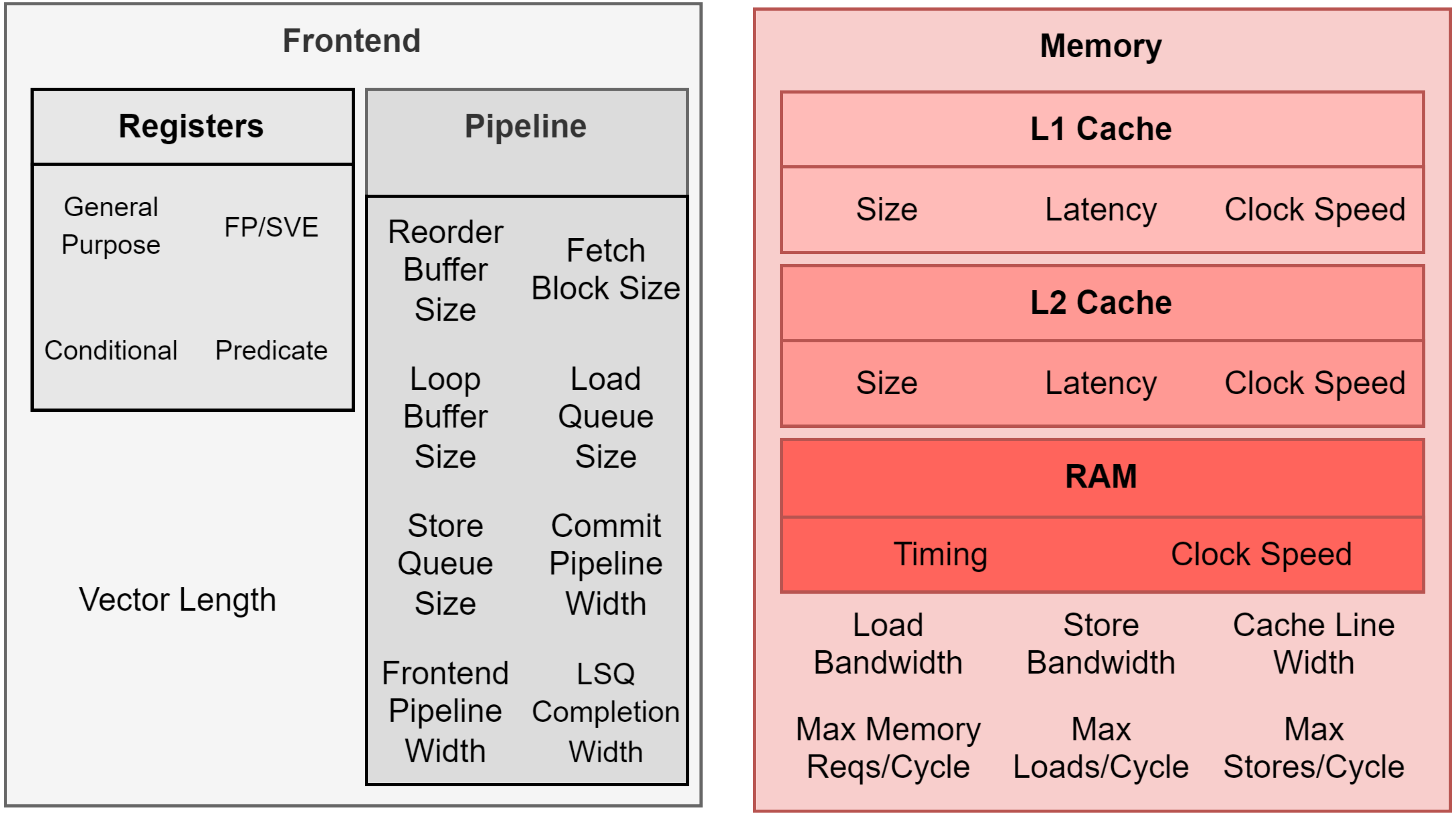
- 2006-2007 “Golden Age”
  - P.J.Joseph et al, Lee et al, Dubach et al
- Lots of work using traditional ML/AI for parameter searches on few parameters
- Significant jumps in computer architecture since
  - Vectors i.e. Scalable Vector Extension are now commonly used!
- Most work since is focused or models power, space etc.
  - Gap in updated, broad view of core architecture

SimEng – Our Simulation Framework

- Cycle-approximate Out-Of-Order CPU simulator
- Allows simulation of every stage of the pipeline
- SST integration for memory model
- ~1 MIPS on moderate hardware
- Easy to use - Simple YAML to define CPU properties

https://uob-hpc.github.io/SimEng

What we are modelling



Frontend

Registers

General Purpose

FP/SVE

Conditional

Predicate

Pipeline

Reorder Buffer Size

Fetch Block Size

Loop Buffer Size

Load Queue Size

Store Queue Size

Commit Pipeline Width

Frontend Pipeline Width

LSQ Completion Width

Vector Length

Memory

L1 Cache

Size

Latency

Clock Speed

L2 Cache

Size

Latency

Clock Speed

RAM

Timing

Clock Speed

Load Bandwidth

Store Bandwidth

Cache Line Width

Max Memory Reqs/Cycle

Max Loads/Cycle

Max Stores/Cycle

What we are **not** modelling

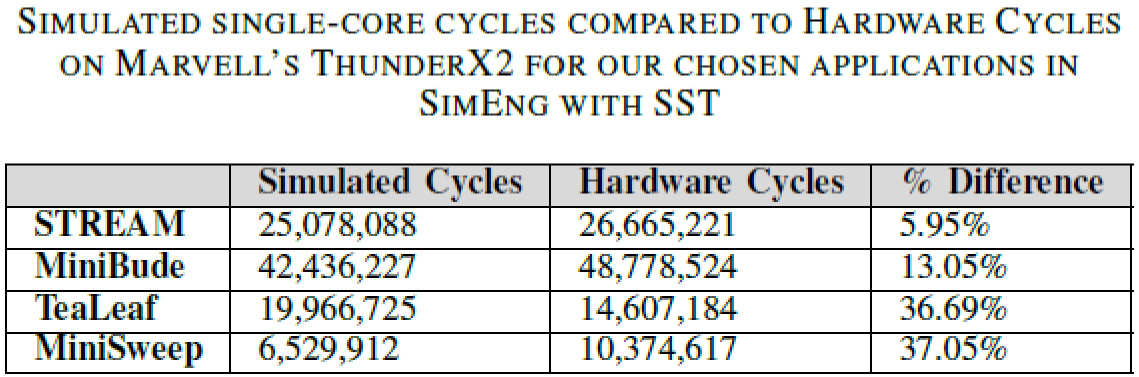
- Reservation Stations
- Execution Units
- No. Cores (just 1)
- Instruction Cache
- L3 Cache – just L1+L2+RAM
- Instruction Set Architecture – fixed to Armv8.4-a+sve

Benchmarks used

- STREAM – Memory Bound
  - Sustained memory bandwidth benchmark
- MiniBude – Compute Bound
  - Drug Screening Mini-app
- TeaLeaf – Memory Bound
  - Heat Conduction Mini-app
- MiniSweep – Compute bound for single core
  - Sn Radiation Transport Mini-app

Remarks on the benchmarks

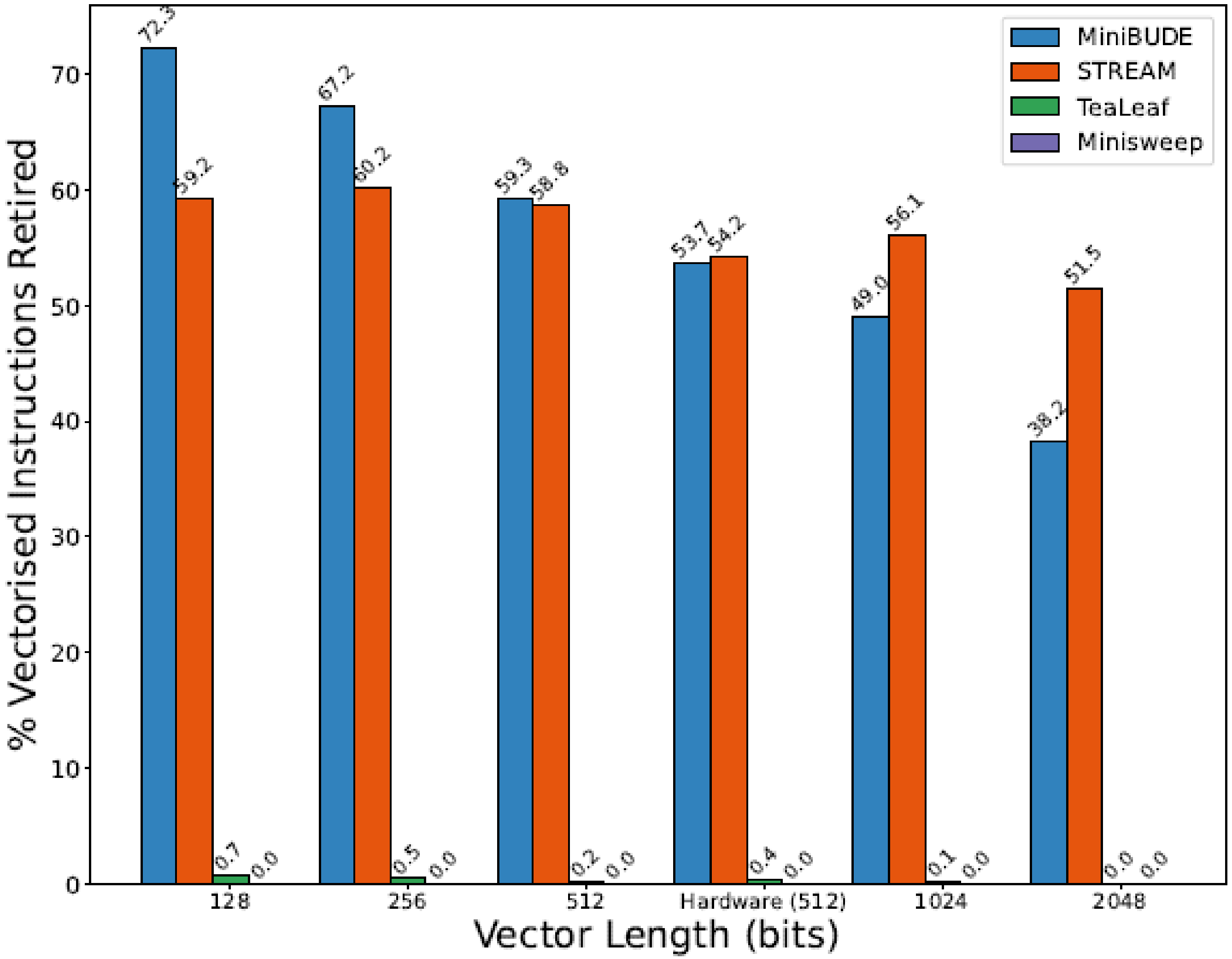
- All problems *mostly* fit into L1 or L2 cache (larger takes too much time)
  - For example, STREAM = ~600KiB
- All compiled with Arm Compiler for Linux v23.04.1
  - Compiled statically with –O3, OpenMP (single threaded), and no MPI
  - SVE Vector Length set to “scalable”



Simulated single-core cycles compared to Hardware Cycles on Marvell’s ThunderX2 for our chosen applications in SimEng with SST

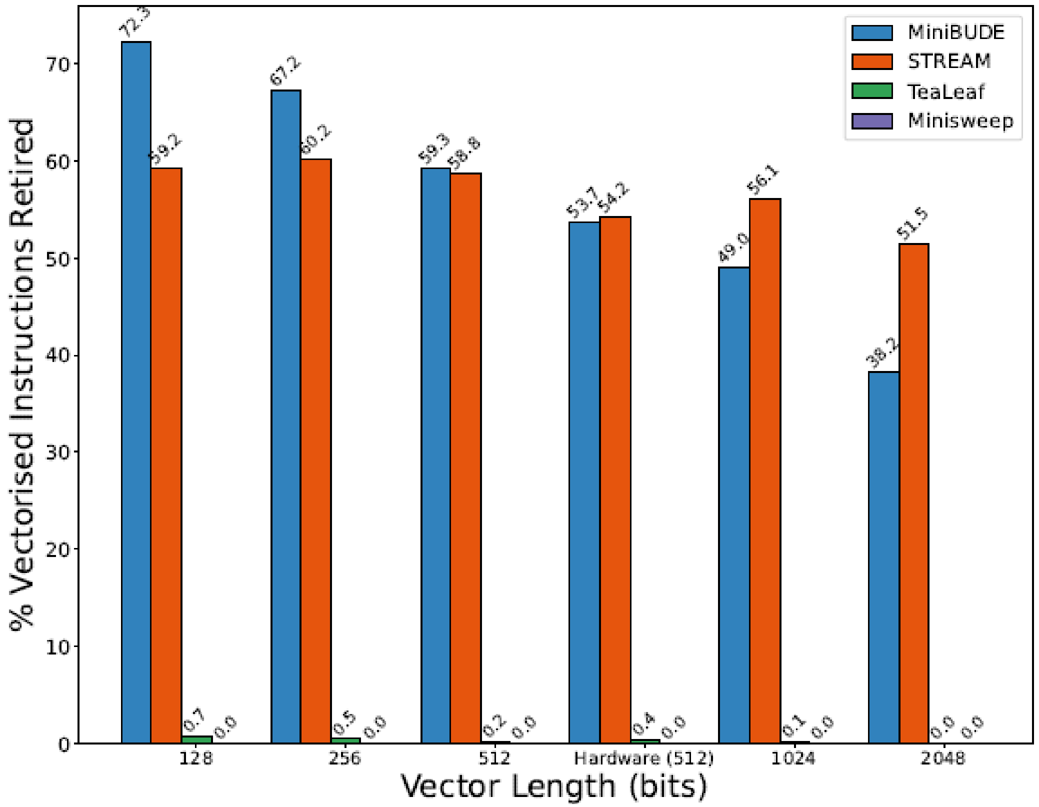
	Simulated Cycles	Hardware Cycles	% Difference
STREAM	25,078,088	26,665,221	5.95%
MiniBude	42,436,227	48,778,524	13.05%
TeaLeaf	19,966,725	14,607,184	36.69%
MiniSweep	6,529,912	10,374,617	37.05%

Code Vectorisation



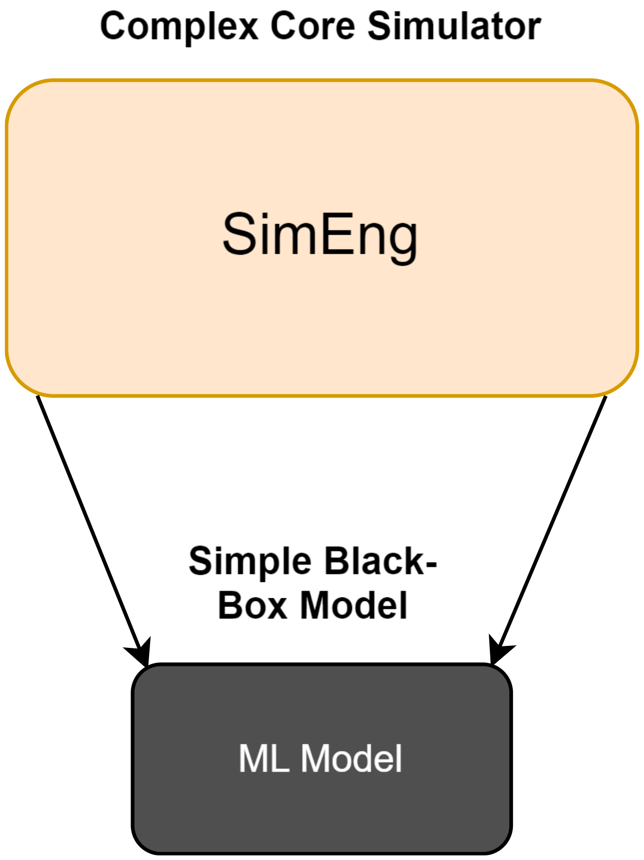
Poor Vectorisation?

- Compiler dependent, not the fault of the hardware!
- Some discrepancies between simulation vs hardware counting
- Huge performance implications
- Not the fault of *-march* flags etc
- Interesting to consider both well vs poorly vectorized performance

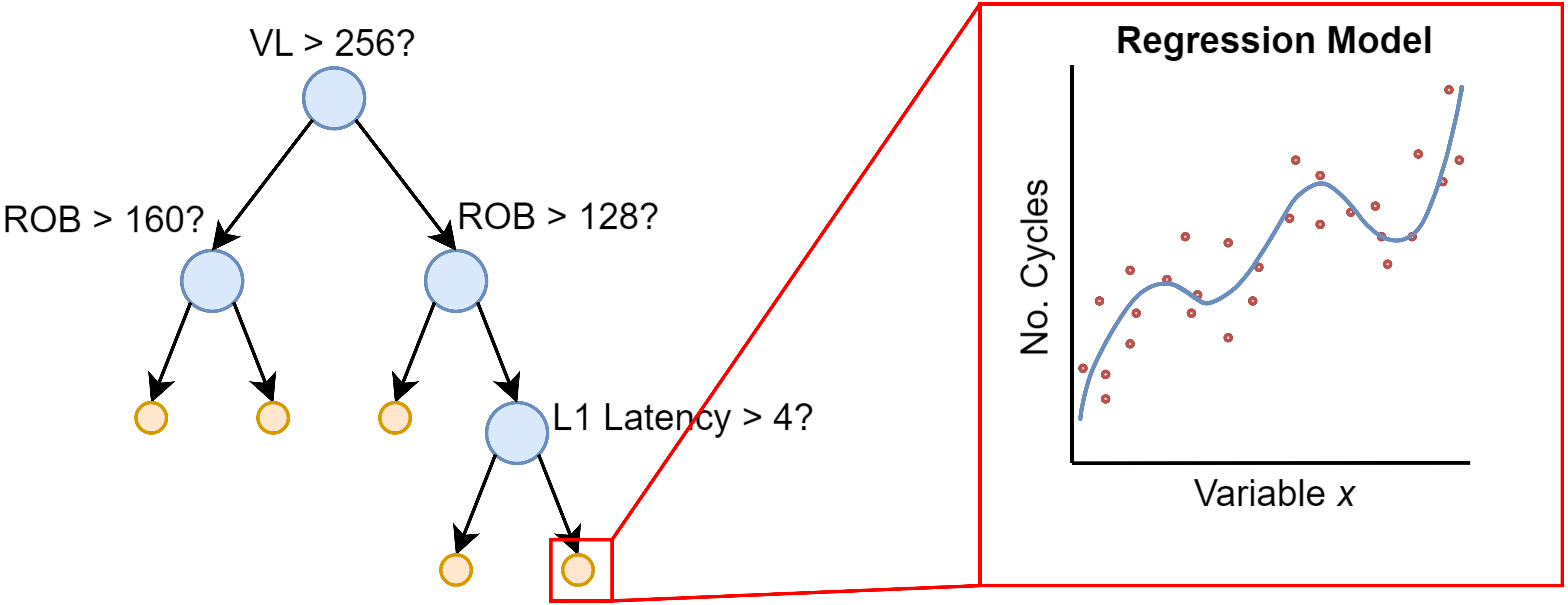


Machine Learning Model

- Surrogate model – map simulation to ML
- Model significantly faster but more constrained
- Lots of high dimensional data
- Predicting numerical output – regression
- Interested in learned data, not the usage



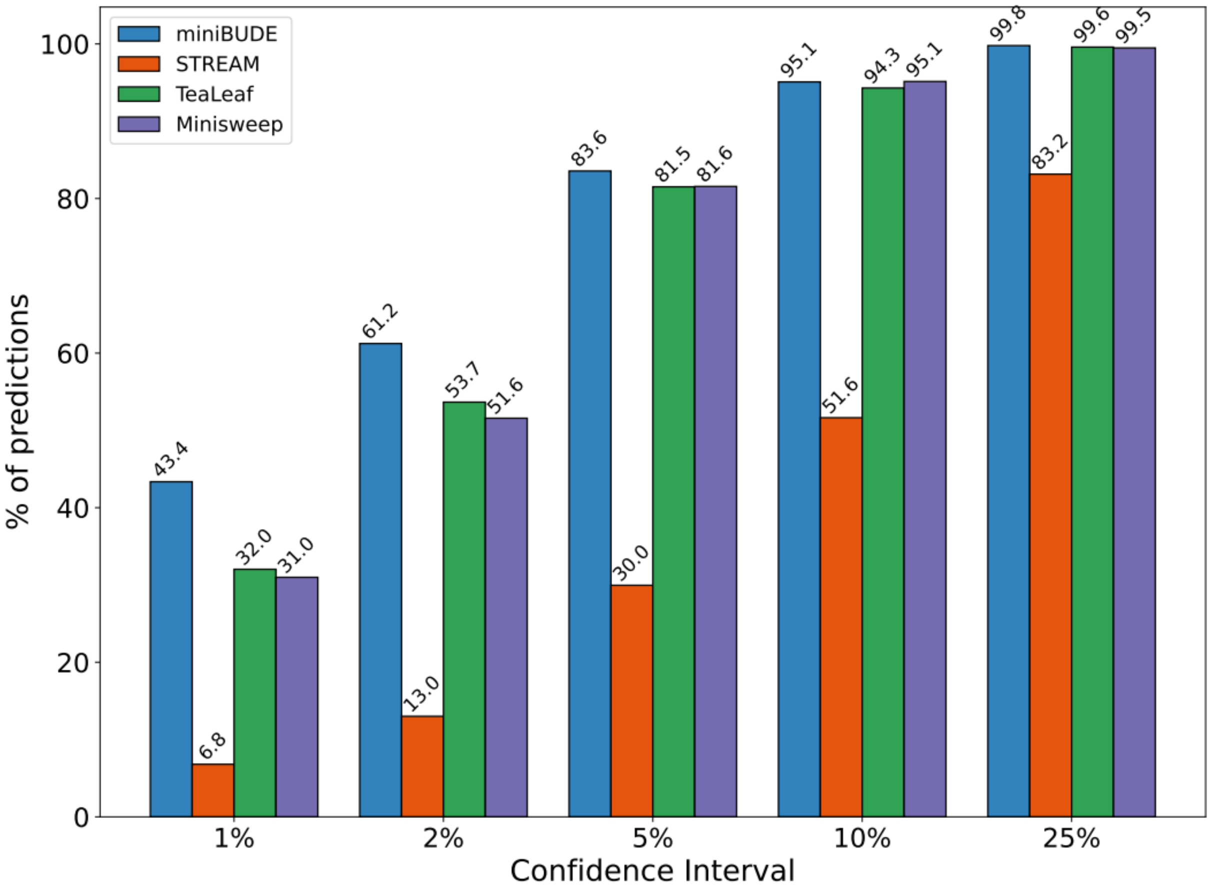
Decision Tree Regressor



Training the model

- 180,006 valid data entries
- Data sampled uniformly at random
- Collected across 10 Marvell Thunder-X2 nodes across ~3 days
- “Data” is runtime statistics for all applications + config
- 80/20 Train/Test split
- One tree per application

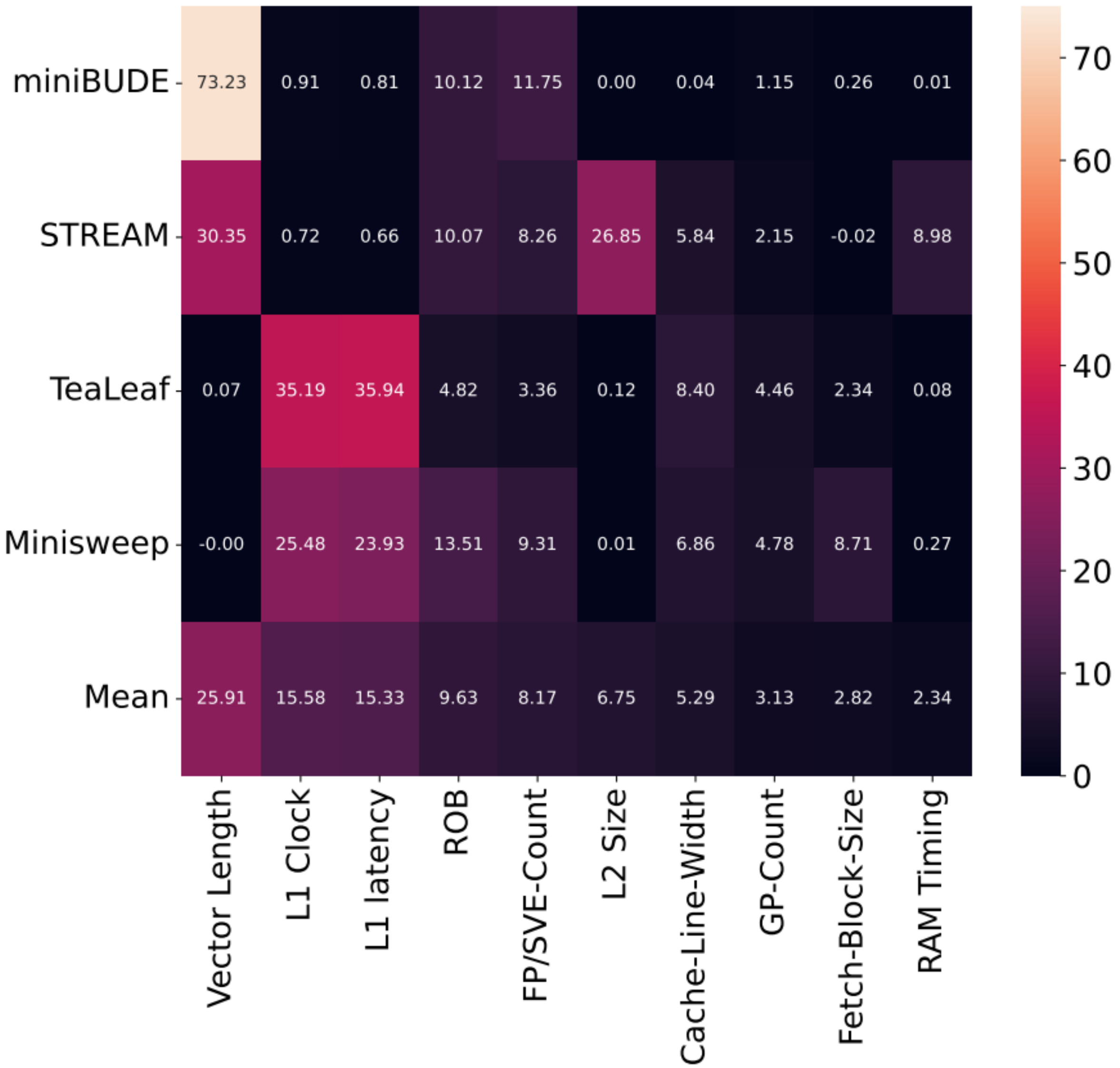
Model Validation



Metric of Importance

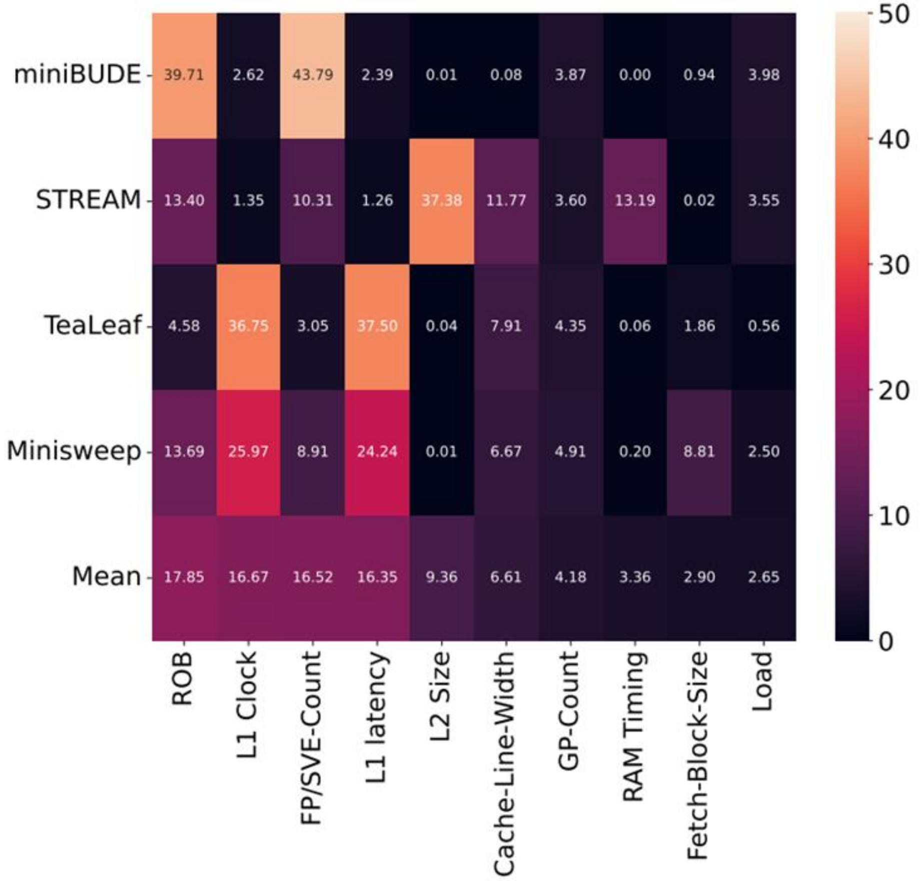
- Permutation Feature Importance
- Shuffle values of each column and predict
- Measure mean absolute error
- More error caused = more important feature
- Feature importance = percentage of summed error across all features

Feature Importance



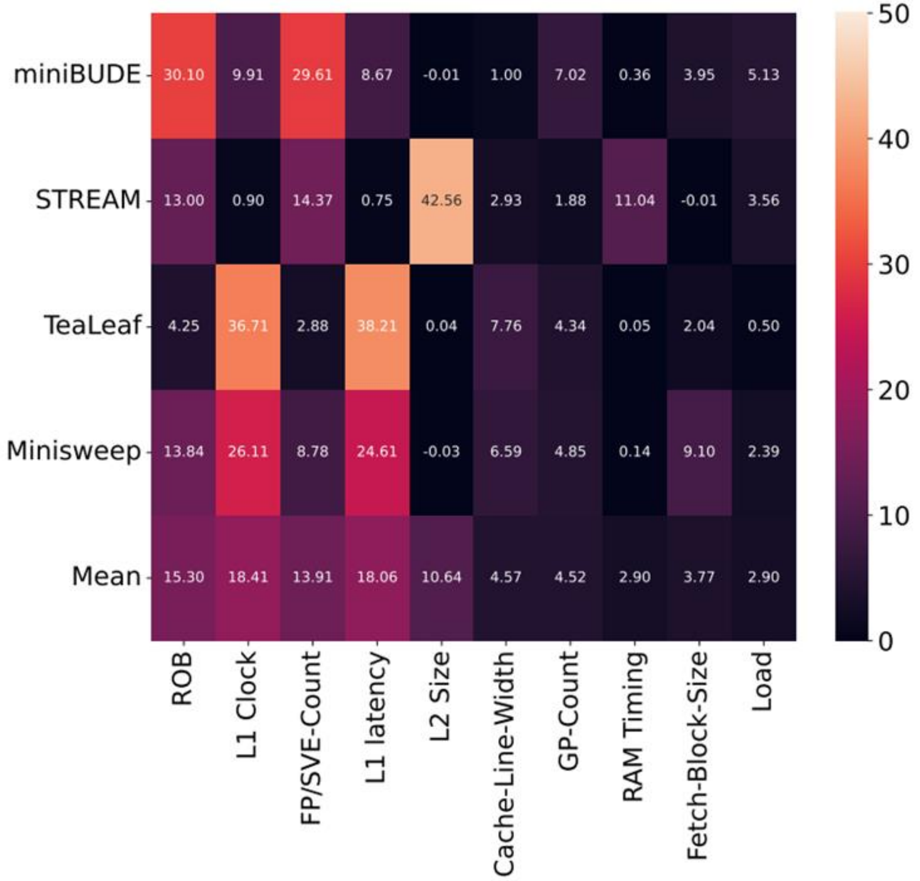
	Vector Length	L1 Clock	L1 latency	ROB	FP/SVE-Count	L2 Size	Cache-Line-Width	GP-Count	Fetch-Block-Size	RAM Timing
miniBUDE	73.23	0.91	0.81	10.12	11.75	0.00	0.04	1.15	0.26	0.01
STREAM	30.35	0.72	0.66	10.07	8.26	26.85	5.84	2.15	-0.02	8.98
TeaLeaf	0.07	35.19	35.94	4.82	3.36	0.12	8.40	4.46	2.34	0.08
Minisweep	-0.00	25.48	23.93	13.51	9.31	0.01	6.86	4.78	8.71	0.27
Mean	25.91	15.58	15.33	9.63	8.17	6.75	5.29	3.13	2.82	2.34

Feature Importance (Fixed Vector Length)



	ROB	L1 Clock	FP/SVE-Count	L1 latency	L2 Size	Cache-Line-Width	GP-Count	RAM Timing	Fetch-Block-Size	Load
miniBUDE	39.71	2.62	43.79	2.39	0.01	0.08	3.87	0.00	0.94	3.98
STREAM	13.40	1.35	10.31	1.26	37.38	11.77	3.60	13.19	0.02	3.55
TeaLeaf	4.58	36.75	3.05	37.50	0.04	7.91	4.35	0.06	1.86	0.56
Minisweep	13.69	25.97	8.91	24.24	0.01	6.67	4.91	0.20	8.81	2.50
Mean	17.85	16.67	16.52	16.35	9.36	6.61	4.18	3.36	2.90	2.65

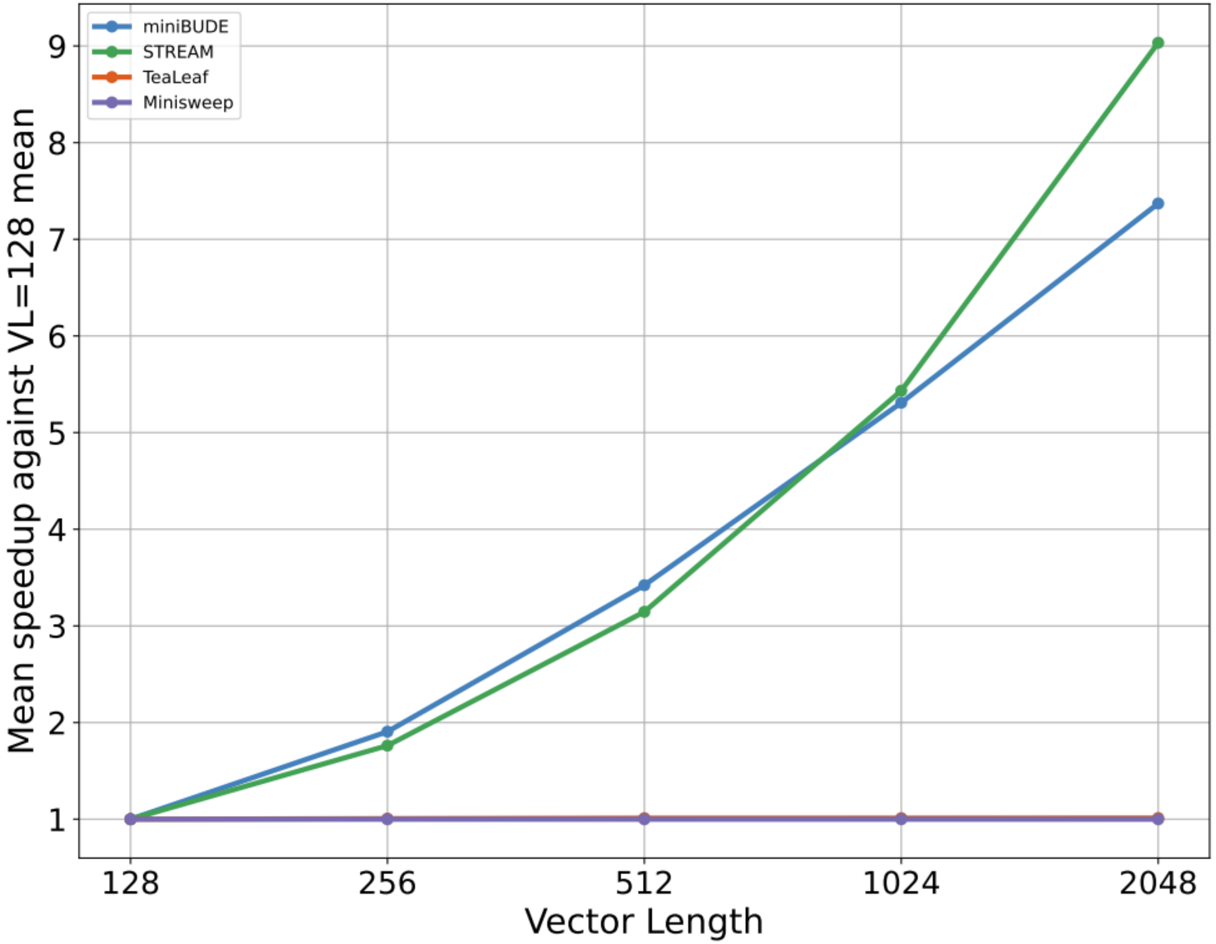
VL=128



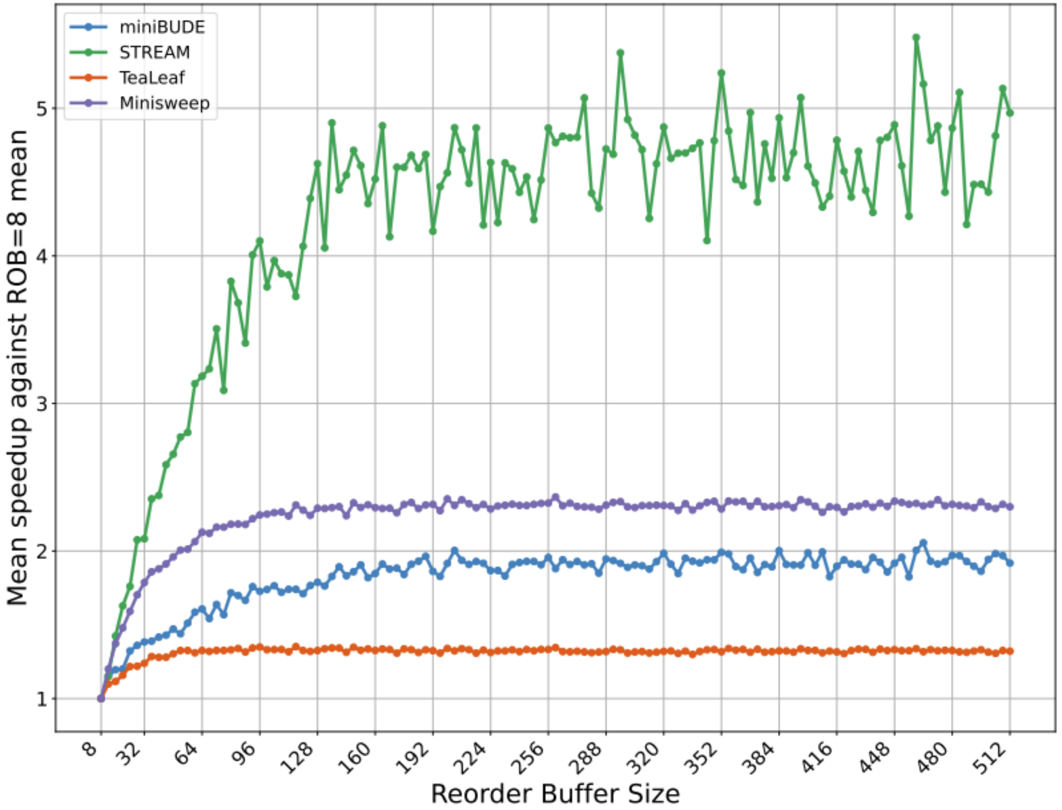
	ROB	L1 Clock	FP/SVE-Count	L1 latency	L2 Size	Cache-Line-Width	GP-Count	RAM Timing	Fetch-Block-Size	Load
miniBUDE	30.10	9.91	29.61	8.67	-0.01	1.00	7.02	0.36	3.95	5.13
STREAM	13.00	0.90	14.37	0.75	42.56	2.93	1.88	11.04	-0.01	3.56
TeaLeaf	4.25	36.71	2.88	38.21	0.04	7.76	4.34	0.05	2.04	0.50
Minisweep	13.84	26.11	8.78	24.61	-0.03	6.59	4.85	0.14	9.10	2.39
Mean	15.30	18.41	13.91	18.06	10.64	4.57	4.52	2.90	3.77	2.90

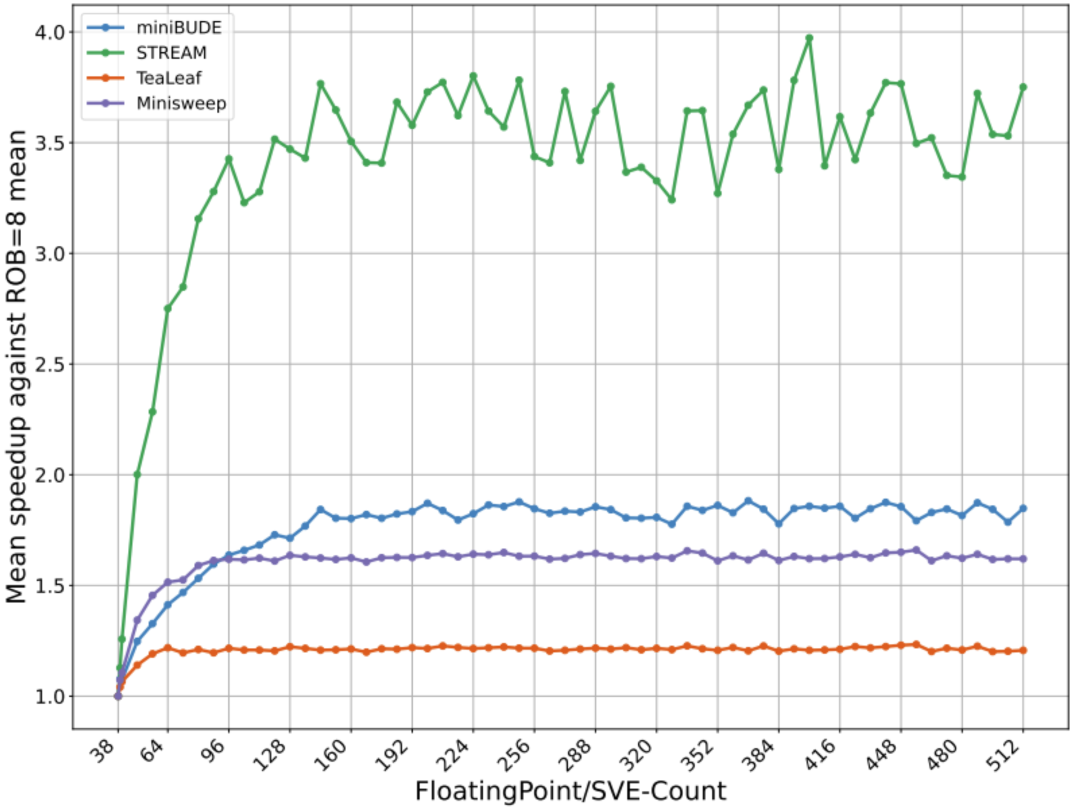
VL=2048

Vector Length



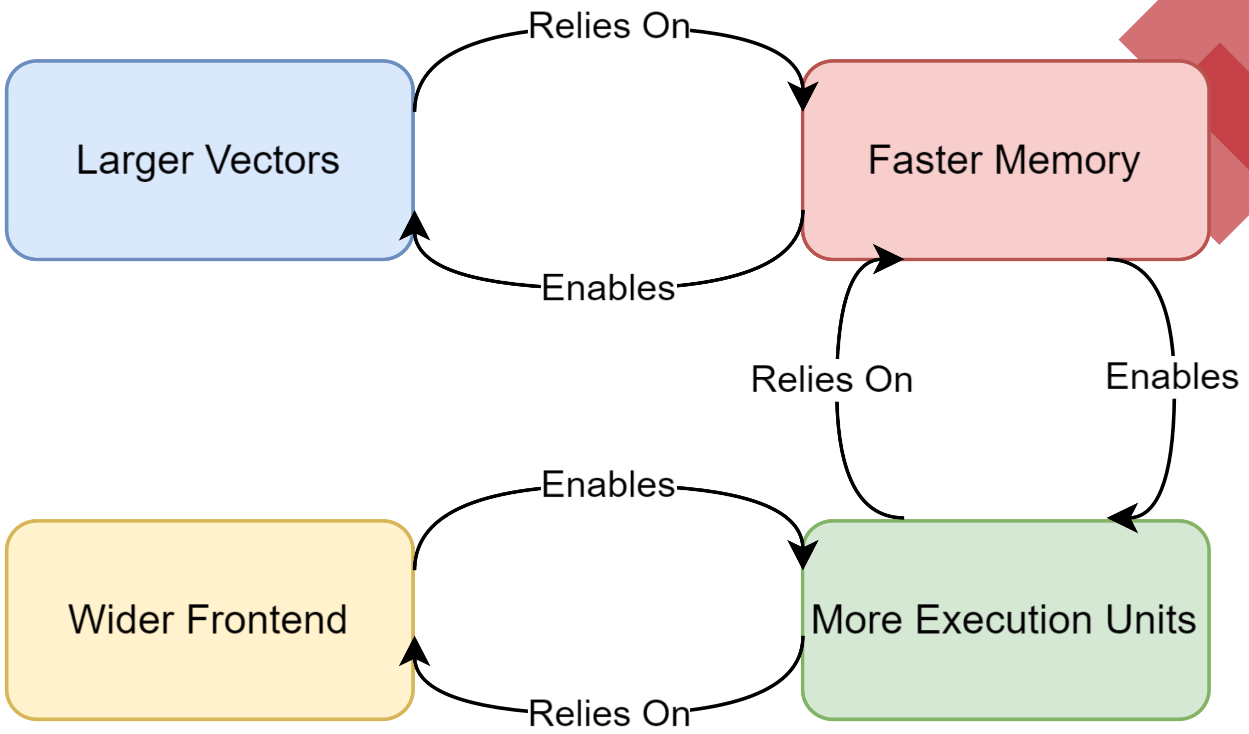
ROB Size / FP/SVE Register Count





What we found

- Vector Length unlocks huge Data-Level-Parallelism (when it's used)
- Memory speed (and capacity) is key
- Frontend throttles, not accelerates



More interestingly…

- We can accurately map out known codes across a large search space
- Faster simulators and machines make data collection cheaper
- Decision Tree Regressors work nicely for modelling these high-dimensional relationships
- Relatively easy to map new codes against a defined architecture space
- Reduces one context of simulation down to a faster surrogate

Future Work

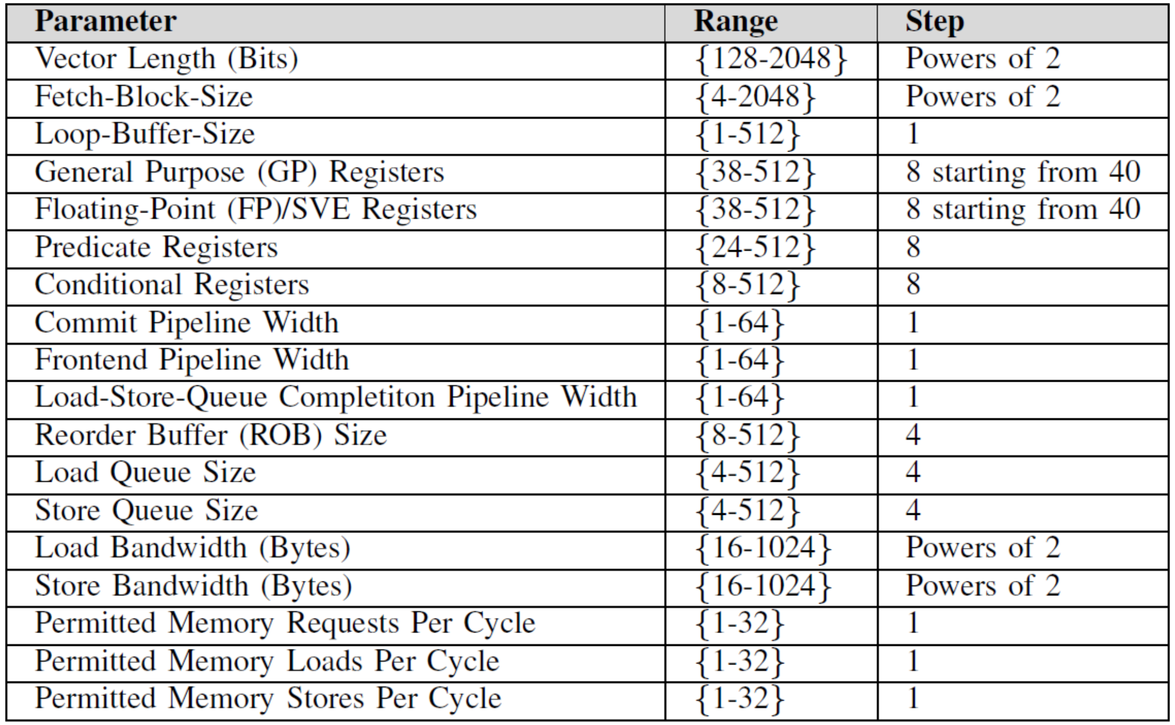
- Multi-core/multi-node modelling to consider communication
- Modelling execution unit design
- Prediction of unseen codes with higher-capacity models
- Improved compiler cost-modelling to fully utilise hardware

Thank you for listening

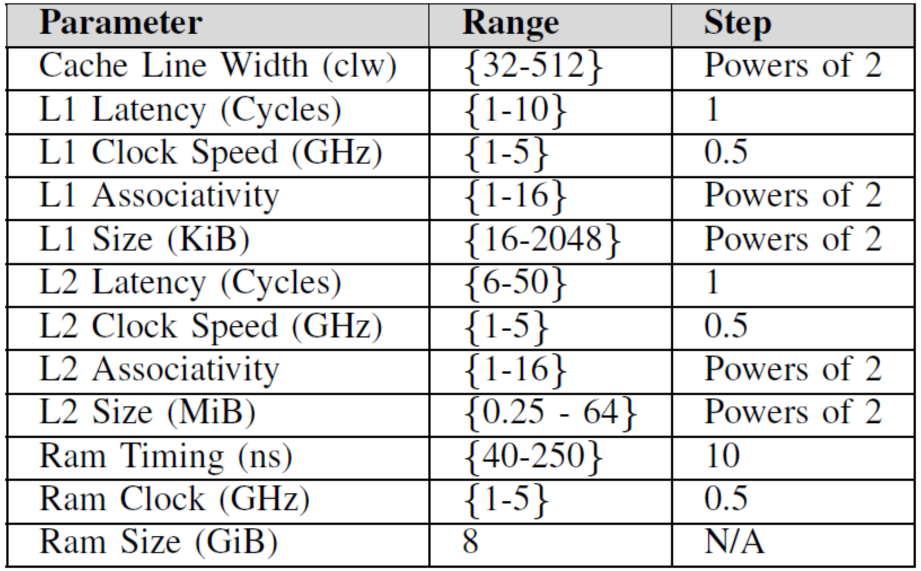
Any questions?

zi23956@bristol.ac.uk

Full Search Space

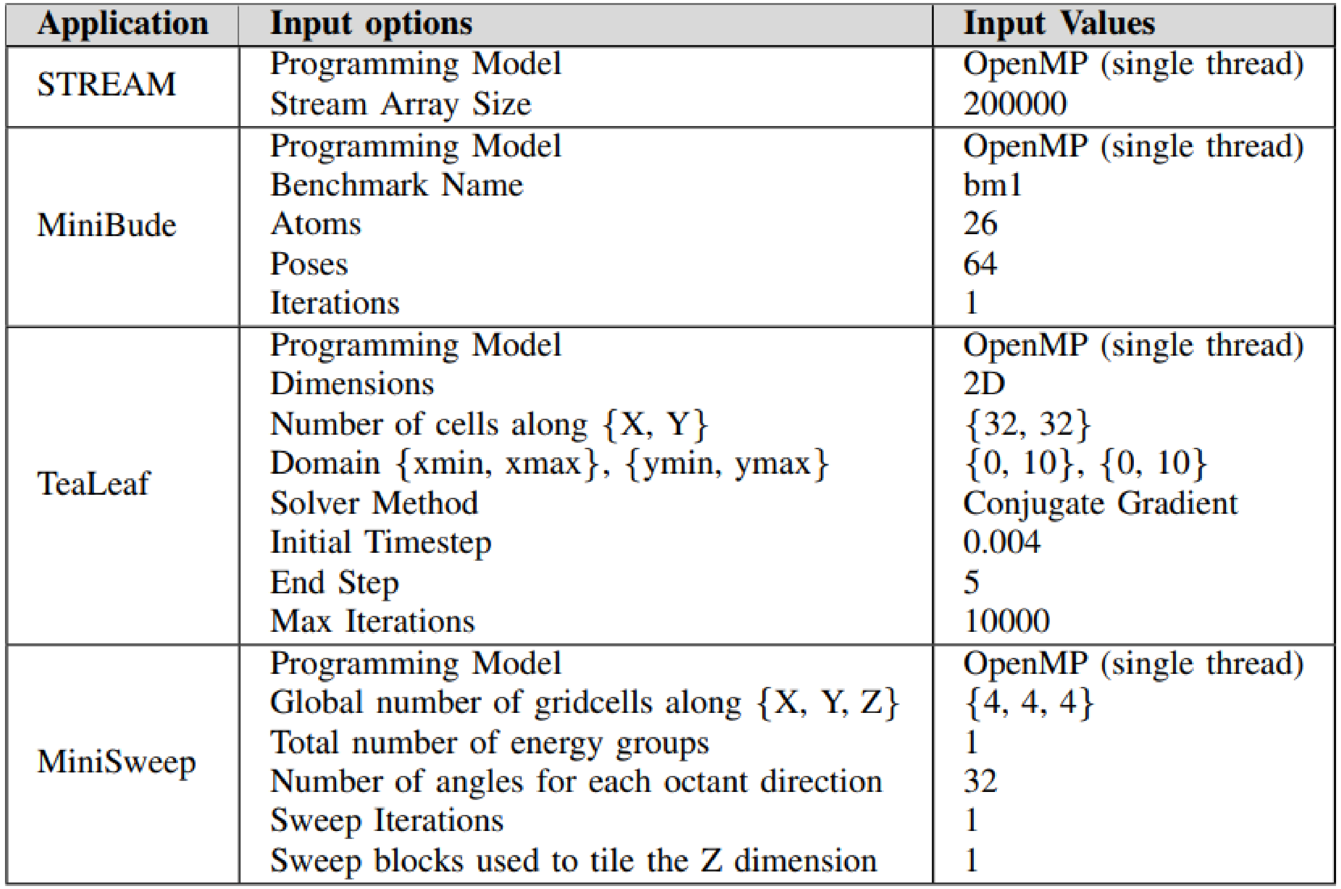


Parameter	Range	Step
Vector Length (Bits)	{128-2048}	Powers of 2
Fetch-Block-Size	{4-2048}	Powers of 2
Loop-Buffer-Size	{1-512}	1
General Purpose (GP) Registers	{38-512}	8 starting from 40
Floating-Point (FP)/SVE Registers	{38-512}	8 starting from 40
Predicate Registers	{24-512}	8
Conditional Registers	{8-512}	8
Commit Pipeline Width	{1-64}	1
Frontend Pipeline Width	{1-64}	1
Load-Store-Queue Completiton Pipeline Width	{1-64}	1
Reorder Buffer (ROB) Size	{8-512}	4
Load Queue Size	{4-512}	4
Store Queue Size	{4-512}	4
Load Bandwidth (Bytes)	{16-1024}	Powers of 2
Store Bandwidth (Bytes)	{16-1024}	Powers of 2
Permitted Memory Requests Per Cycle	{1-32}	1
Permitted Memory Loads Per Cycle	{1-32}	1
Permitted Memory Stores Per Cycle	{1-32}	1



Parameter	Range	Step
Cache Line Width (clw)	{32-512}	Powers of 2
L1 Latency (Cycles)	{1-10}	1
L1 Clock Speed (GHz)	{1-5}	0.5
L1 Associativity	{1-16}	Powers of 2
L1 Size (KiB)	{16-2048}	Powers of 2
L2 Latency (Cycles)	{6-50}	1
L2 Clock Speed (GHz)	{1-5}	0.5
L2 Associativity	{1-16}	Powers of 2
L2 Size (MiB)	{0.25 - 64}	Powers of 2
Ram Timing (ns)	{40-250}	10
Ram Clock (GHz)	{1-5}	0.5
Ram Size (GiB)	8	N/A

Benchmark Parameters



Application	Input options	Input Values
STREAM	Programming Model	OpenMP (single thread)
	Stream Array Size	200000
MiniBude	Programming Model	OpenMP (single thread)
	Benchmark Name	bm1
	Atoms	26
	Poses	64
	Iterations	1
TeaLeaf	Programming Model	OpenMP (single thread)
	Dimensions	2D
	Number of cells along {X, Y}	{32, 32}
	Domain {xmin, xmax}, {ymin, ymax}	{0, 10}, {0, 10}
	Solver Method	Conjugate Gradient
	Initial Timestep	0.004
	End Step	5
	Max Iterations	10000
MiniSweep	Programming Model	OpenMP (single thread)
	Global number of gridcells along {X, Y, Z}	{4, 4, 4}
	Total number of energy groups	1
	Number of angles for each octant direction	32
	Sweep Iterations	1
	Sweep blocks used to tile the Z dimension	1