Dynamic Adaptation of Runtime Systems Based on Energy Consumption

IFL'24 – 27 August 2024

Jordy Aaldering
Bernard van Gastel
Sven-Bodo Scholz

Radboud University

Energy crisis

Greenhouse gas emissions of ICT projected to be 14% of total in 2040 [1]

We should take steps to decrease energy consumption of software!

However

- Measuring energy consumption is hard
  - Cannot isolate a single process
- Decreasing energy consumption is even harder!
  - No control over environment and background

[1] Lotfi Belkhir, Ahmed Elmeligi, Assessing ICT global emissions footprint: Trends to 2040 & recommendations

Adapt to runtime changes

Cannot isolate energy consumption of a single process

- Consider system's energy consumption as a whole

What users can measure

- CPU total energy consumption (RAPL)

What we can control

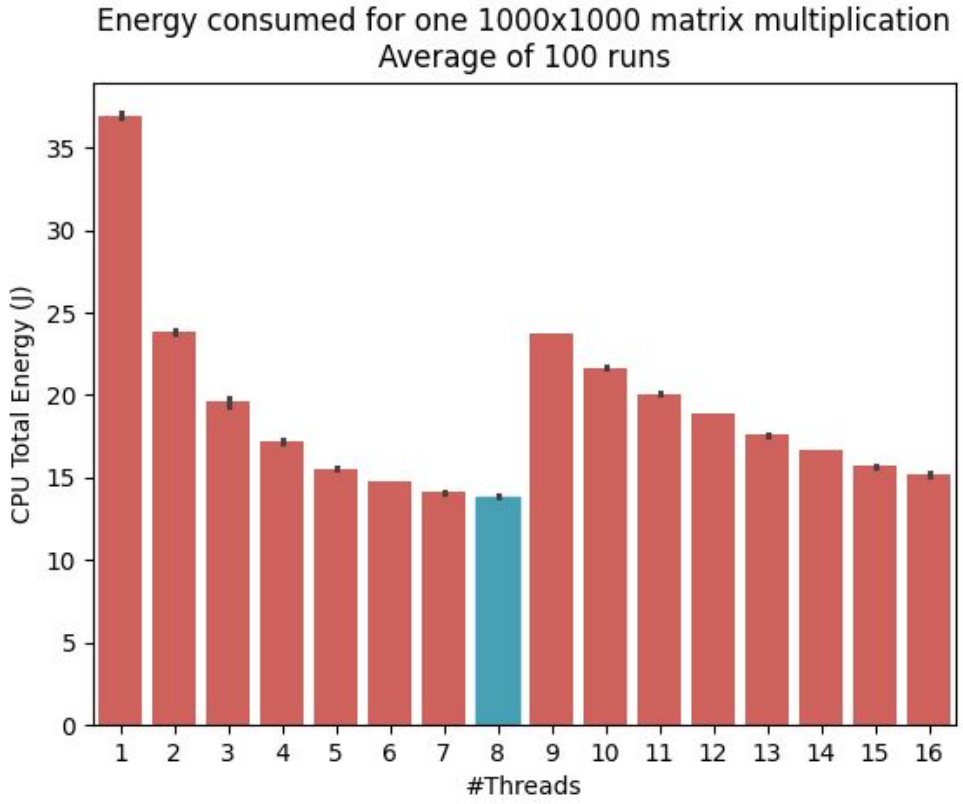
- Thread-count
- Perfect for HPC
- Interested in the variation between measurements

Why is a static choice not good enough?

Energy per thread-count

Intel Xeon E-2378
- 8 homogeneous cores
- 16 threads

What about
- Other running programs
- Different hardware
- Performance/efficiency cores
- Power profile
- Architecture
- Throttling
- Pinning
- etc.



Why is a static choice not good enough?

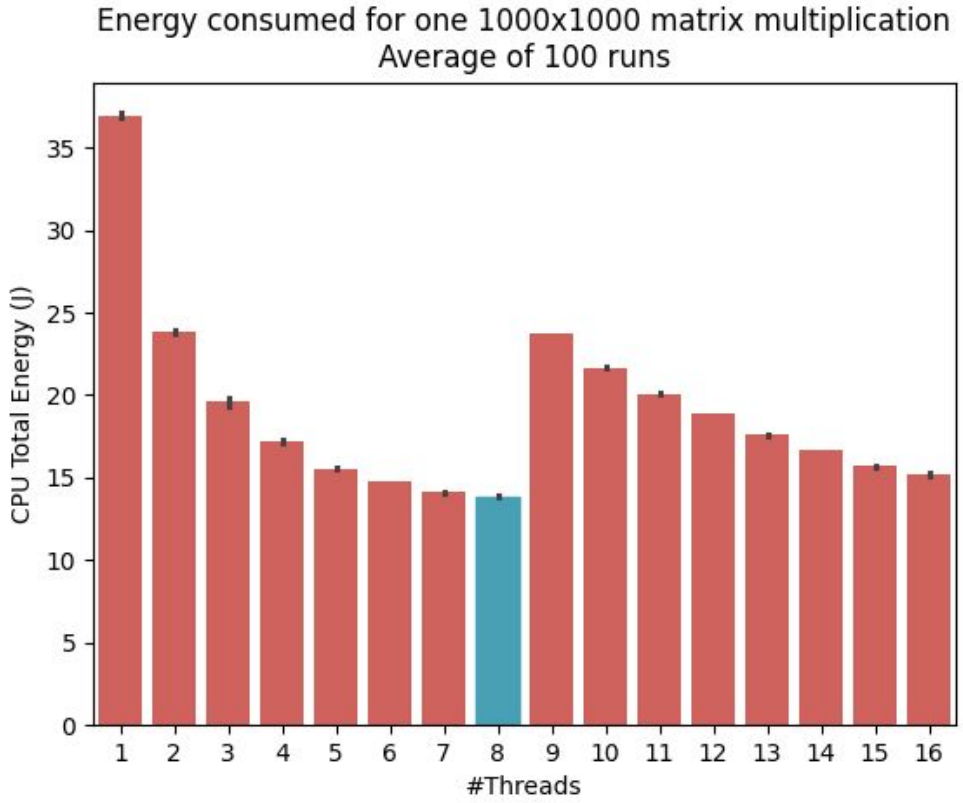
Energy per thread-count

Intel Xeon E-2378

- 8 homogeneous cores
- 16 threads

What about

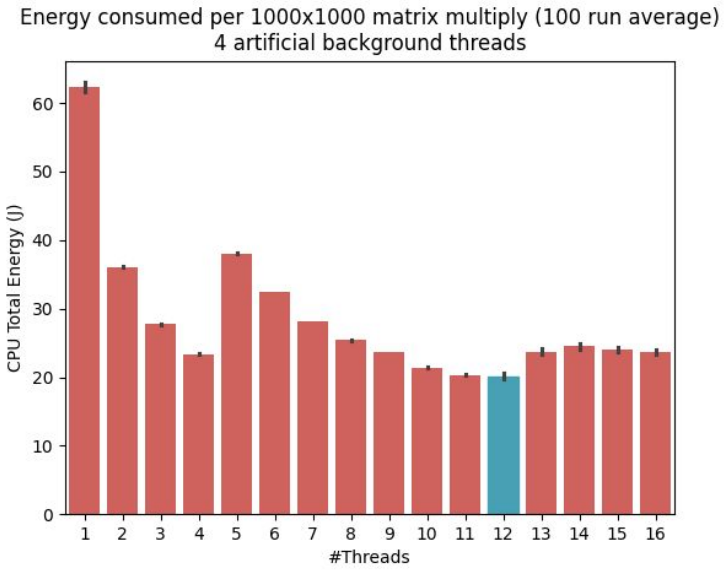
- Other running programs
- Different hardware
- Performance/efficiency cores
- Power profile
- Architecture
- Throttling
- Pinning
- etc.

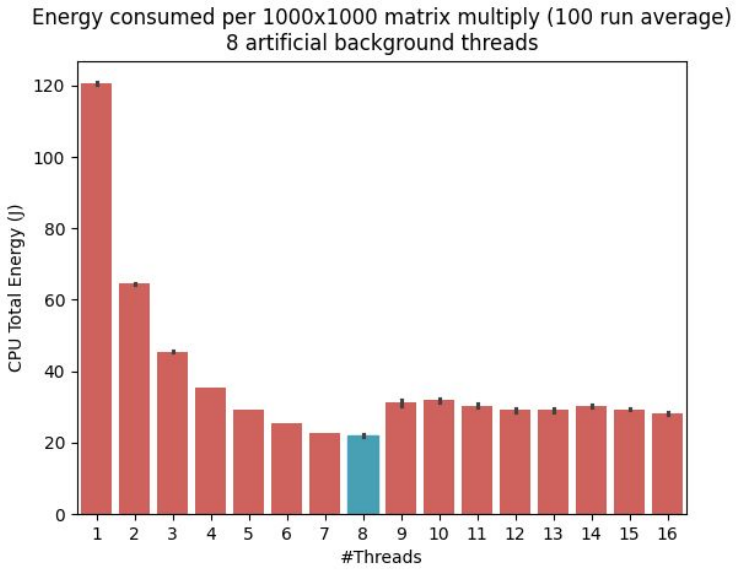


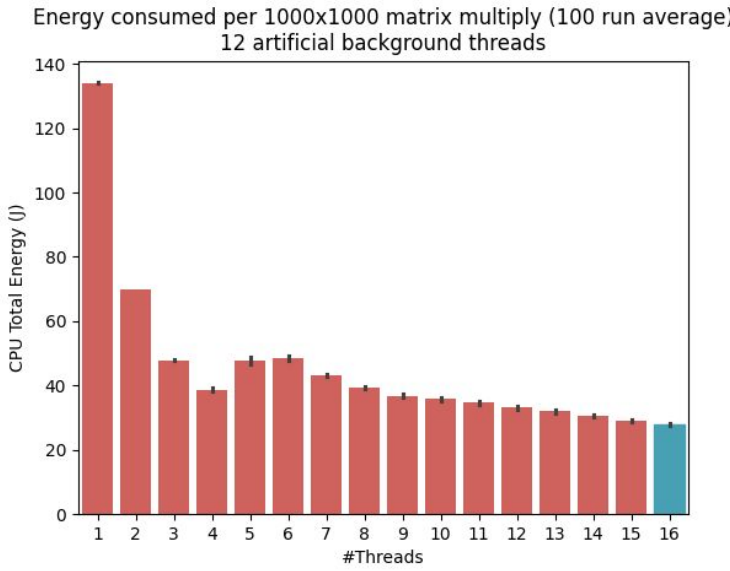
Why is a static choice not good enough?

Energy per thread-count

How does optimum change if we add a background load?





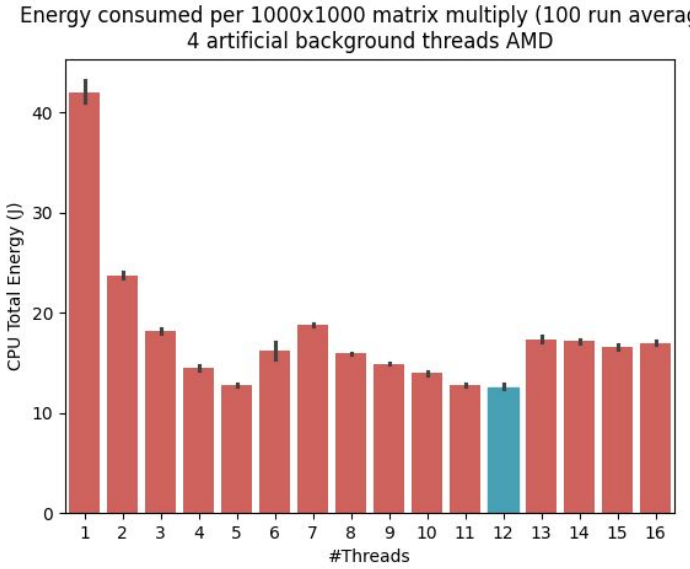


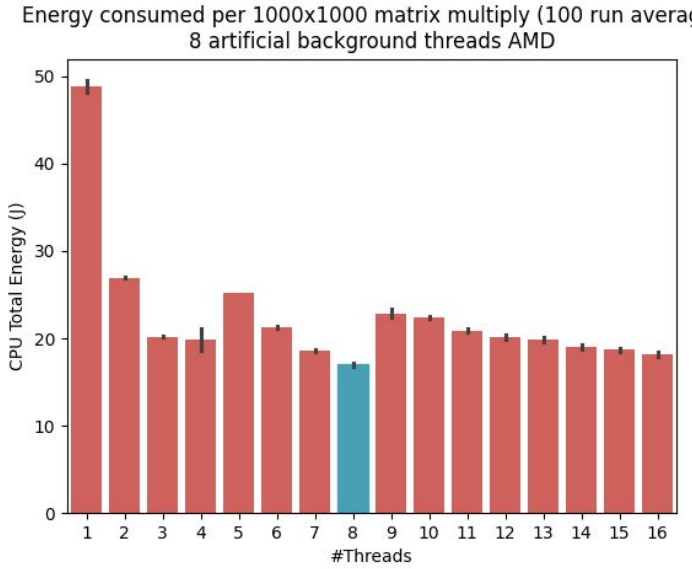
Why is a static choice not good enough?

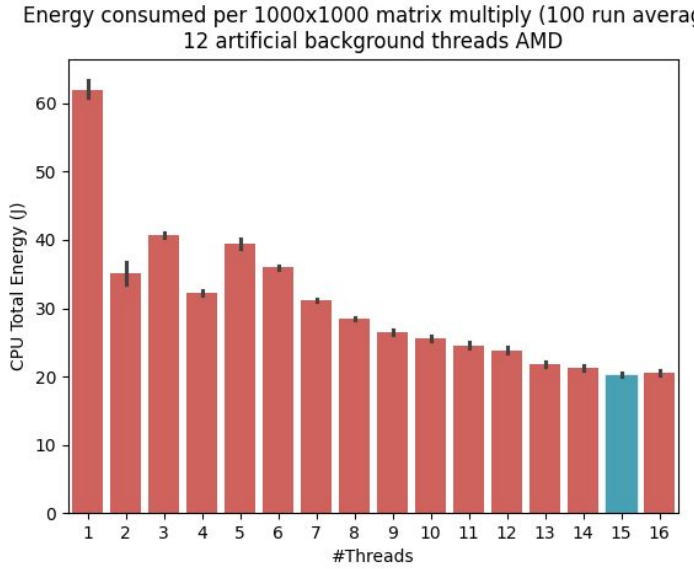
Energy per thread-count

AMD Ryzen 7 5800H

- 8 homogeneous cores
- 16 threads







Dynamic Adaptation

Shows why we need dynamic adaptation

Note that

- System changes relatively slow compared to measuring time
  - Other relatively long-running programs
  - Otherwise we consider it as noise
- Require a repeated parallel application
  - Matmul
  - Stencils
  - N-body
  - etc.

How?

Program provides energy statistics of the parallel iteration

- With-loop in the case of SaC
- Energy measured with RAPL

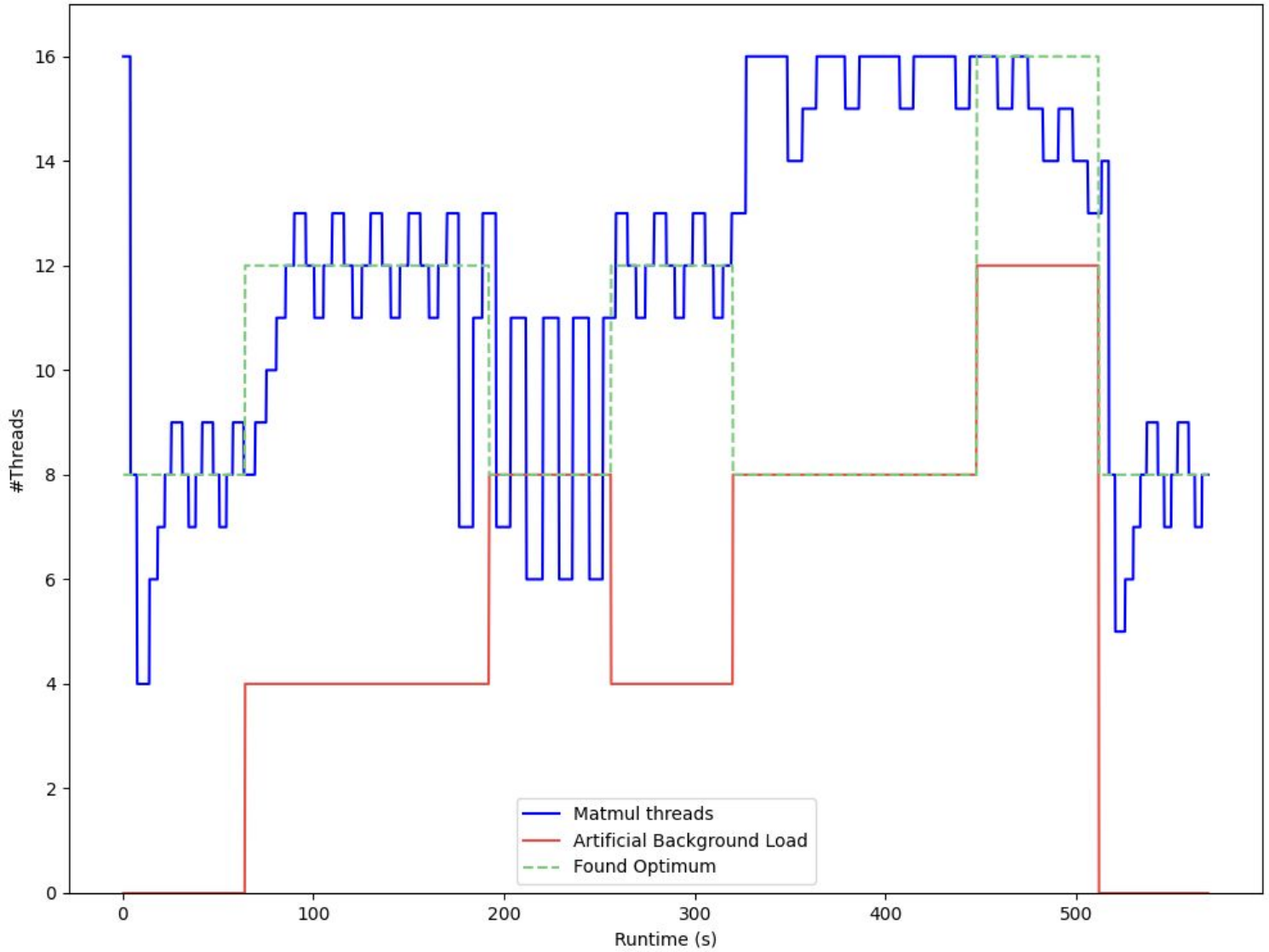
Framework updates optimal thread-count every 20 iterations

- Get lowest energy consumption those measurements
  - Frequency distribution
- Compare against
  - Previous energy consumption
  - Currently known best energy consumption
- Accordingly adjust step direction and step size

Changing background load

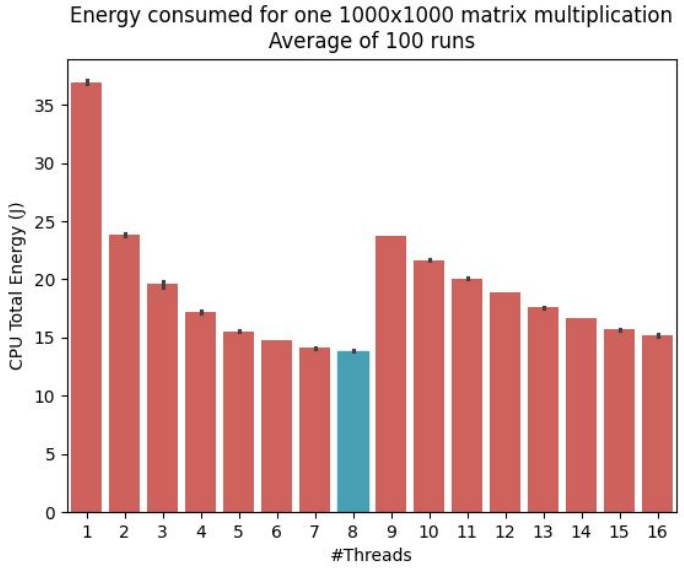
Results

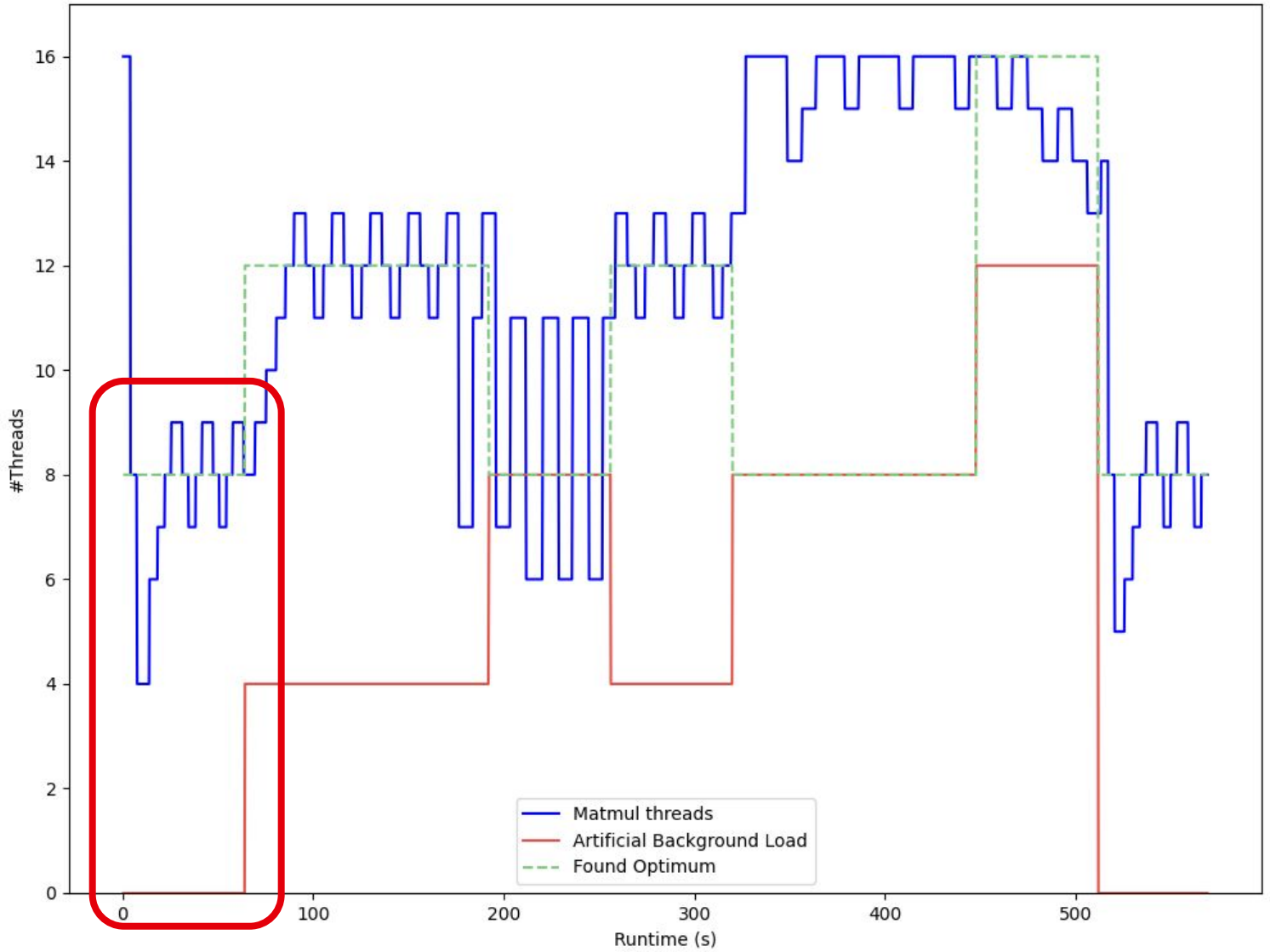
Able to adapt to runtime changes



Changing background load

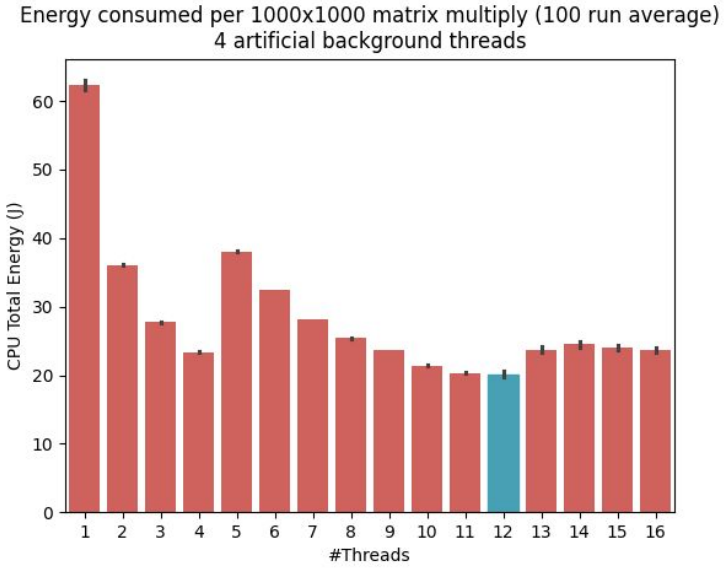
Results

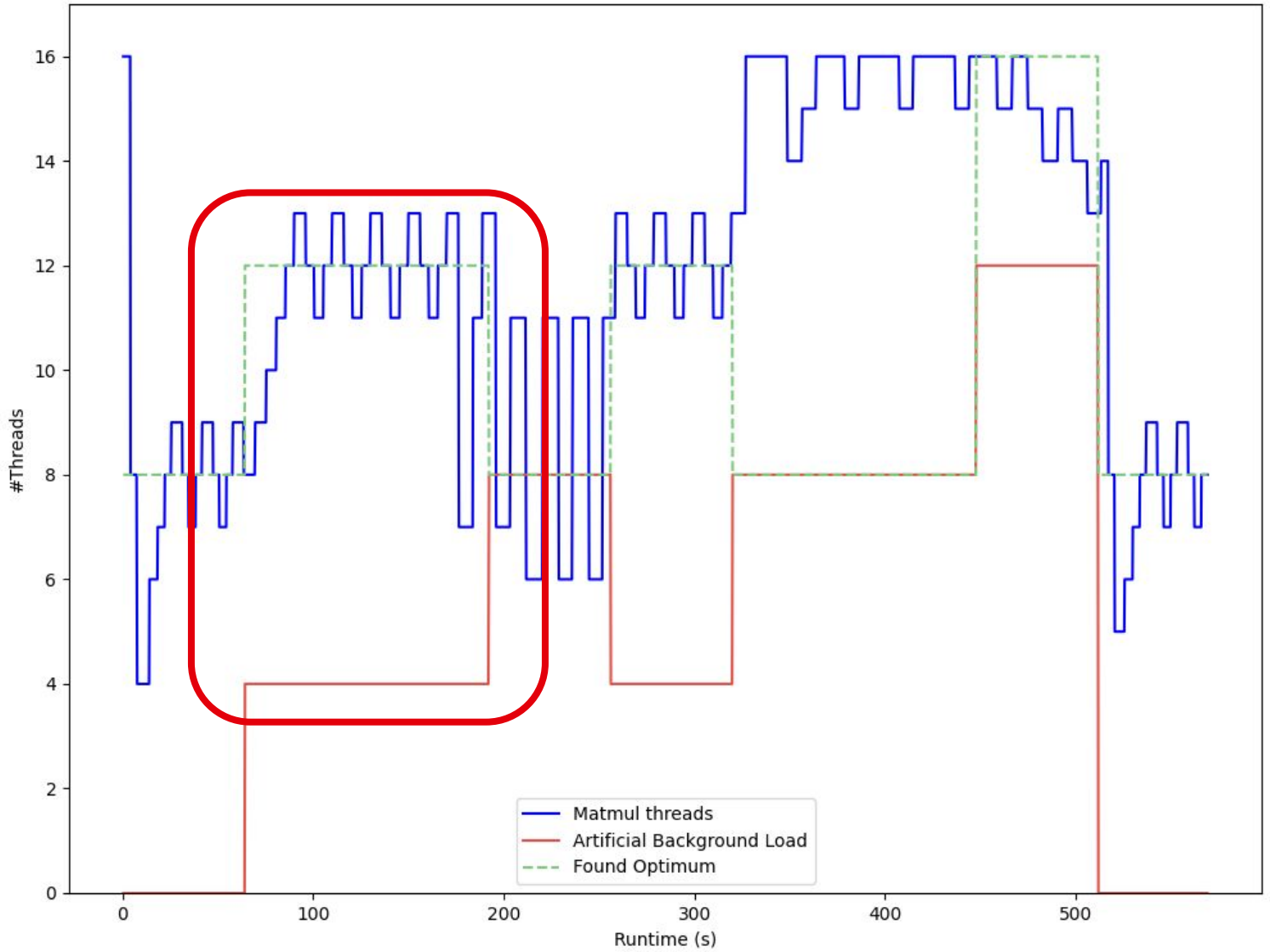




Changing background load

Results

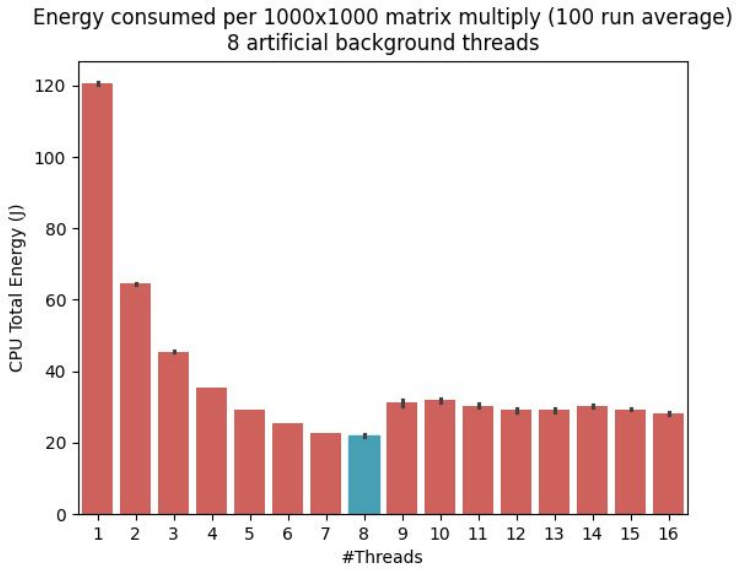


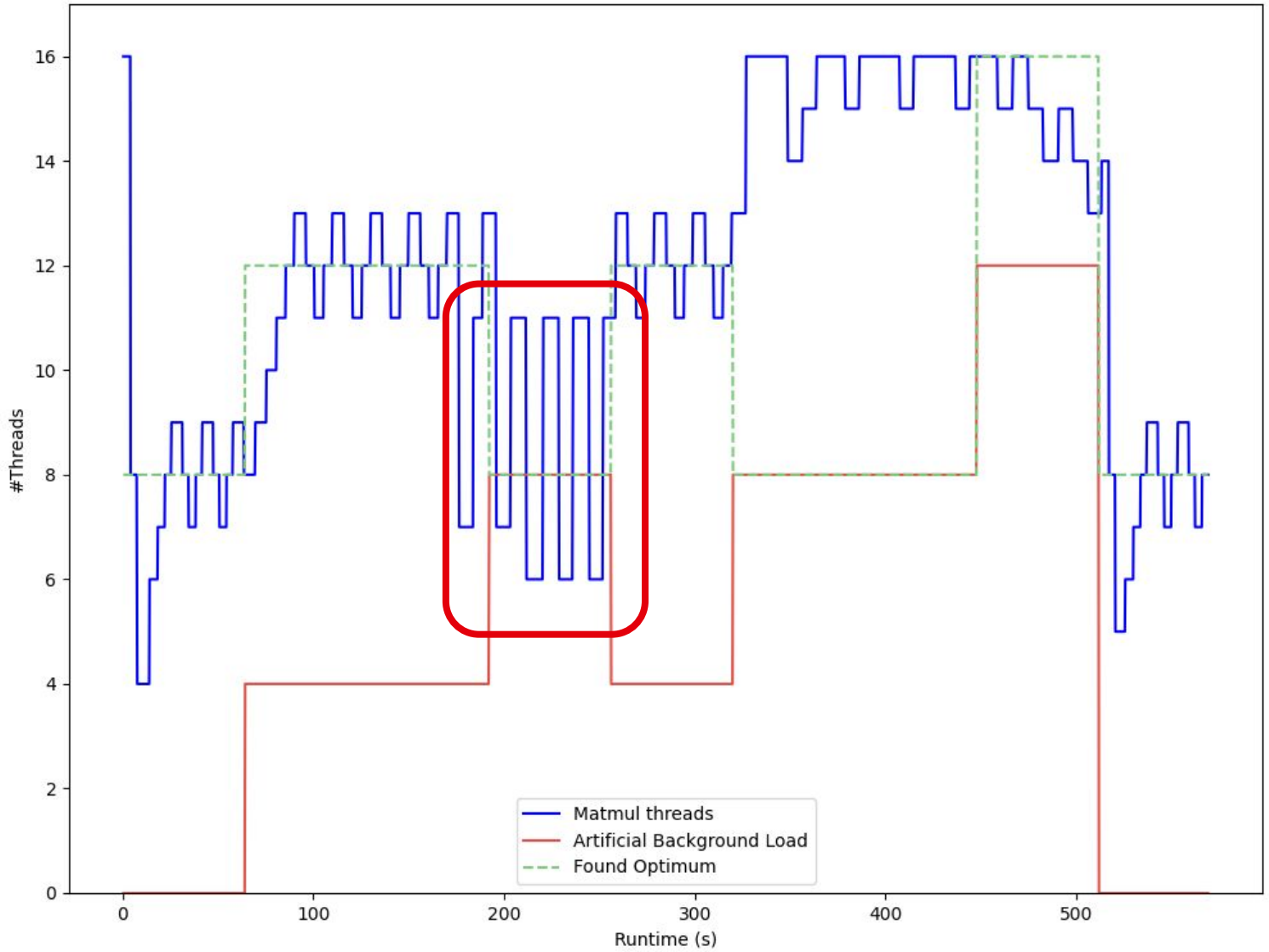


Changing background load

Results

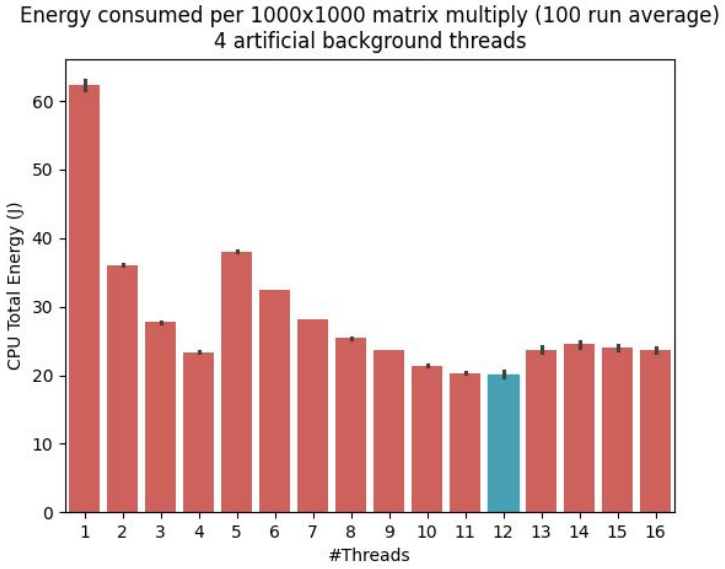
We keep jumping too far past the optimum

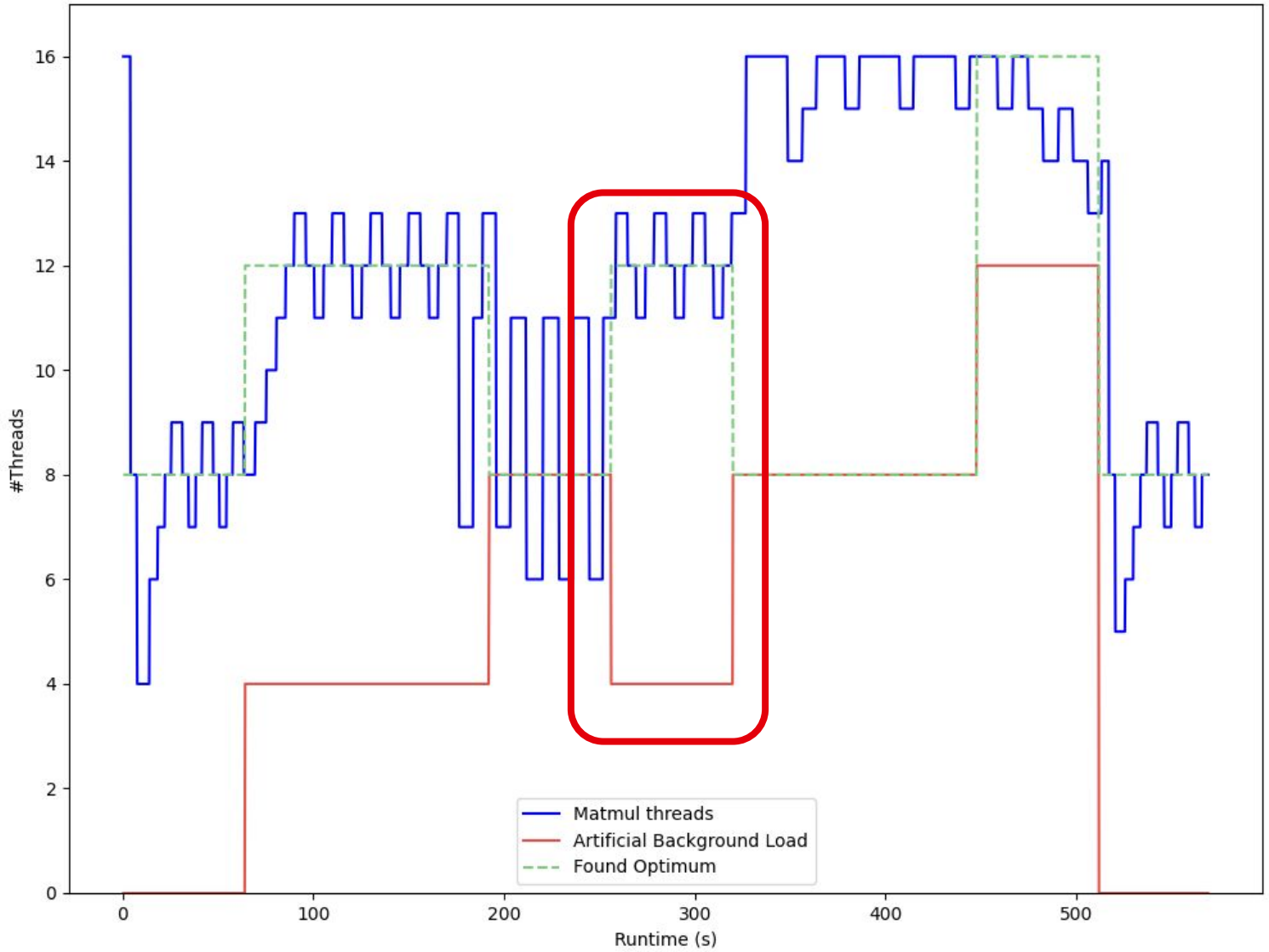




Changing background load

Results

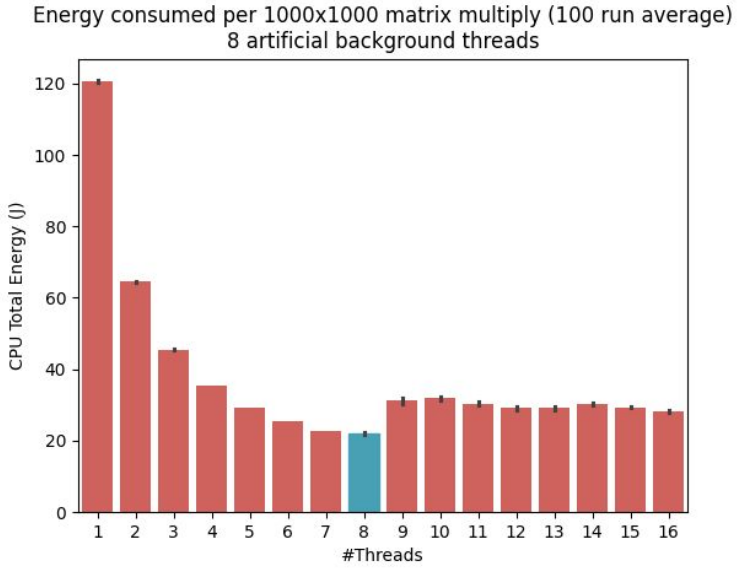


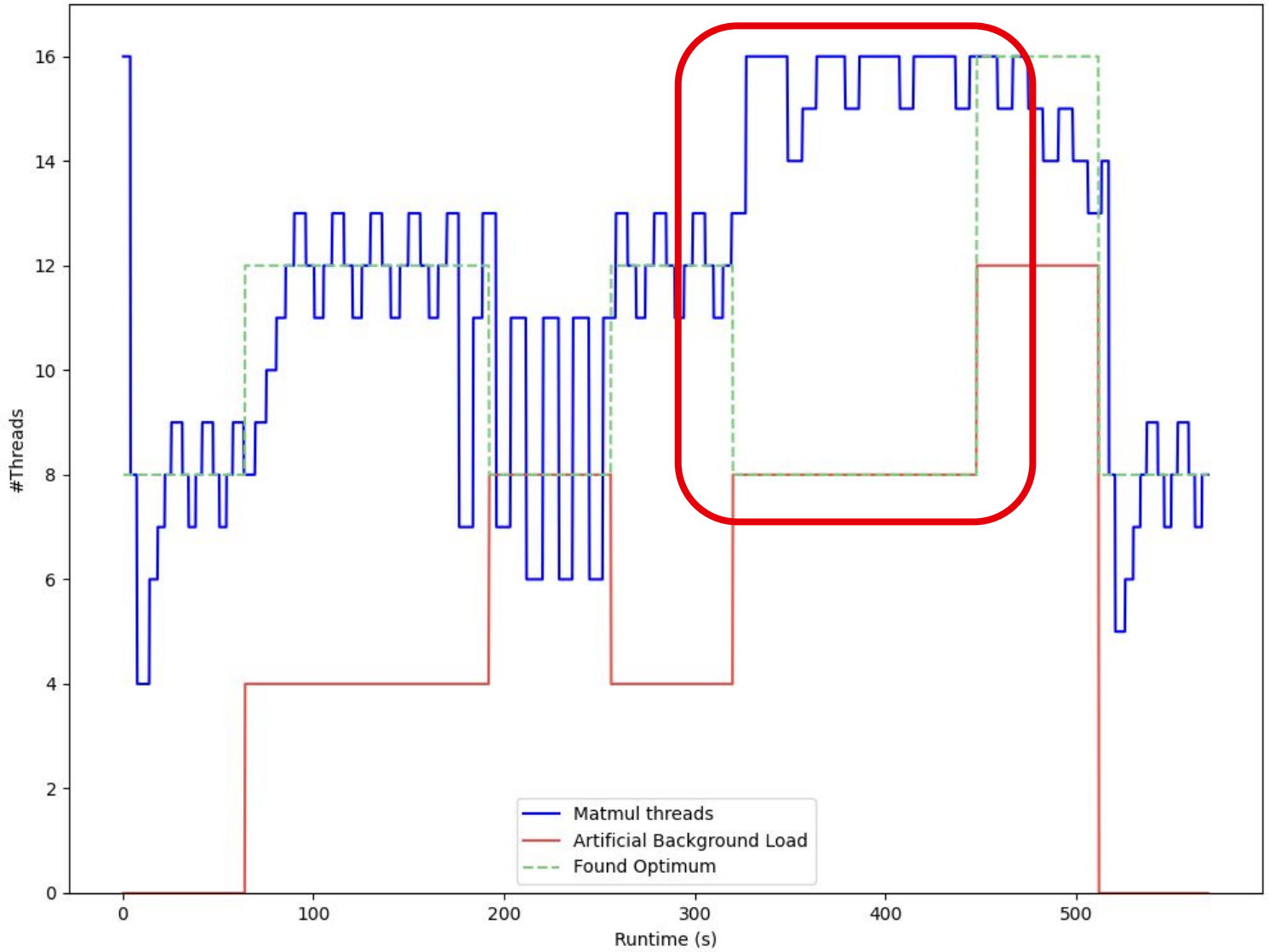


Changing background load

Results

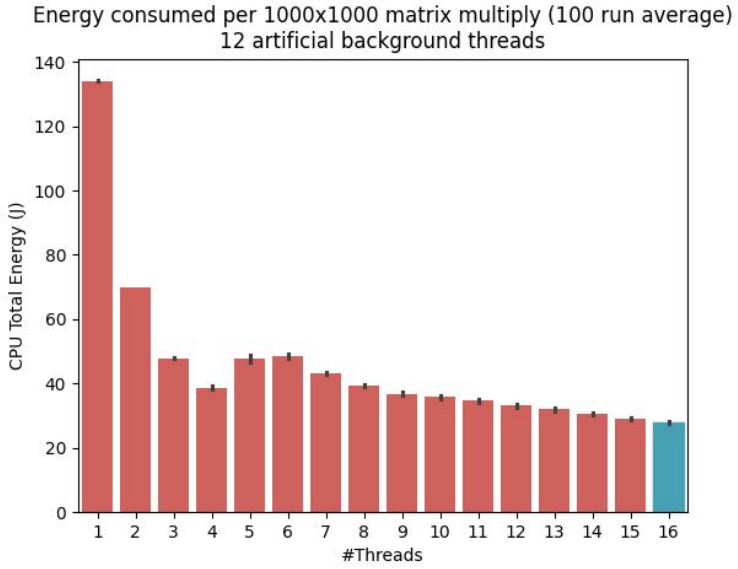
Local optimum

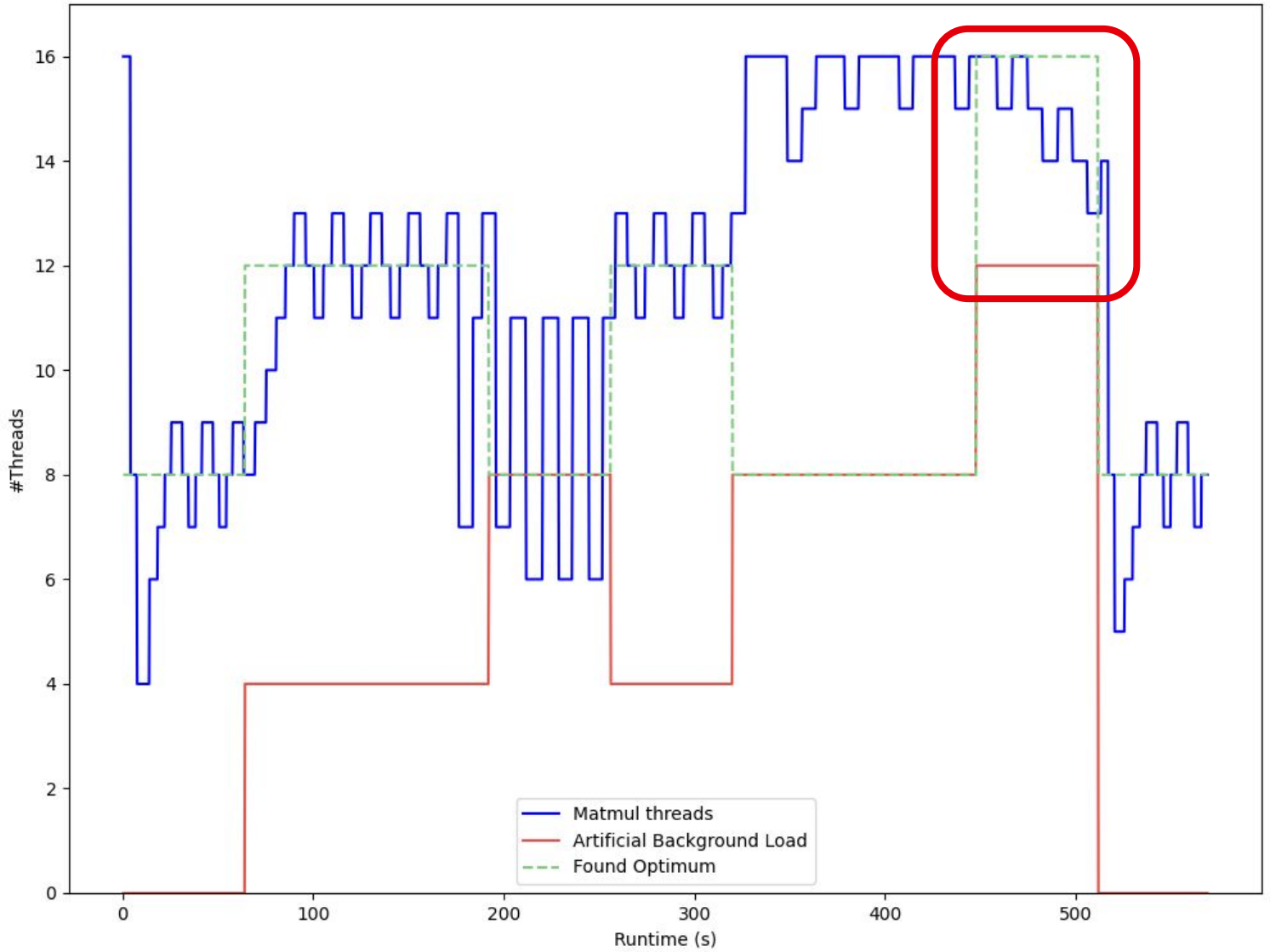




Changing background load

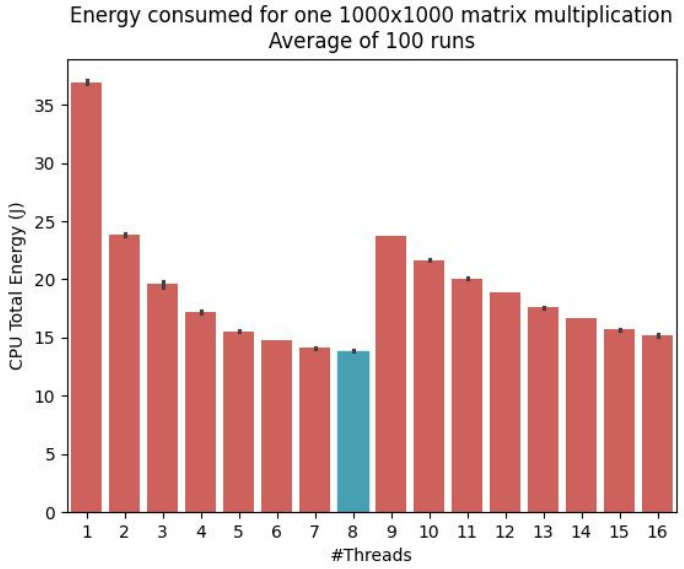
Results

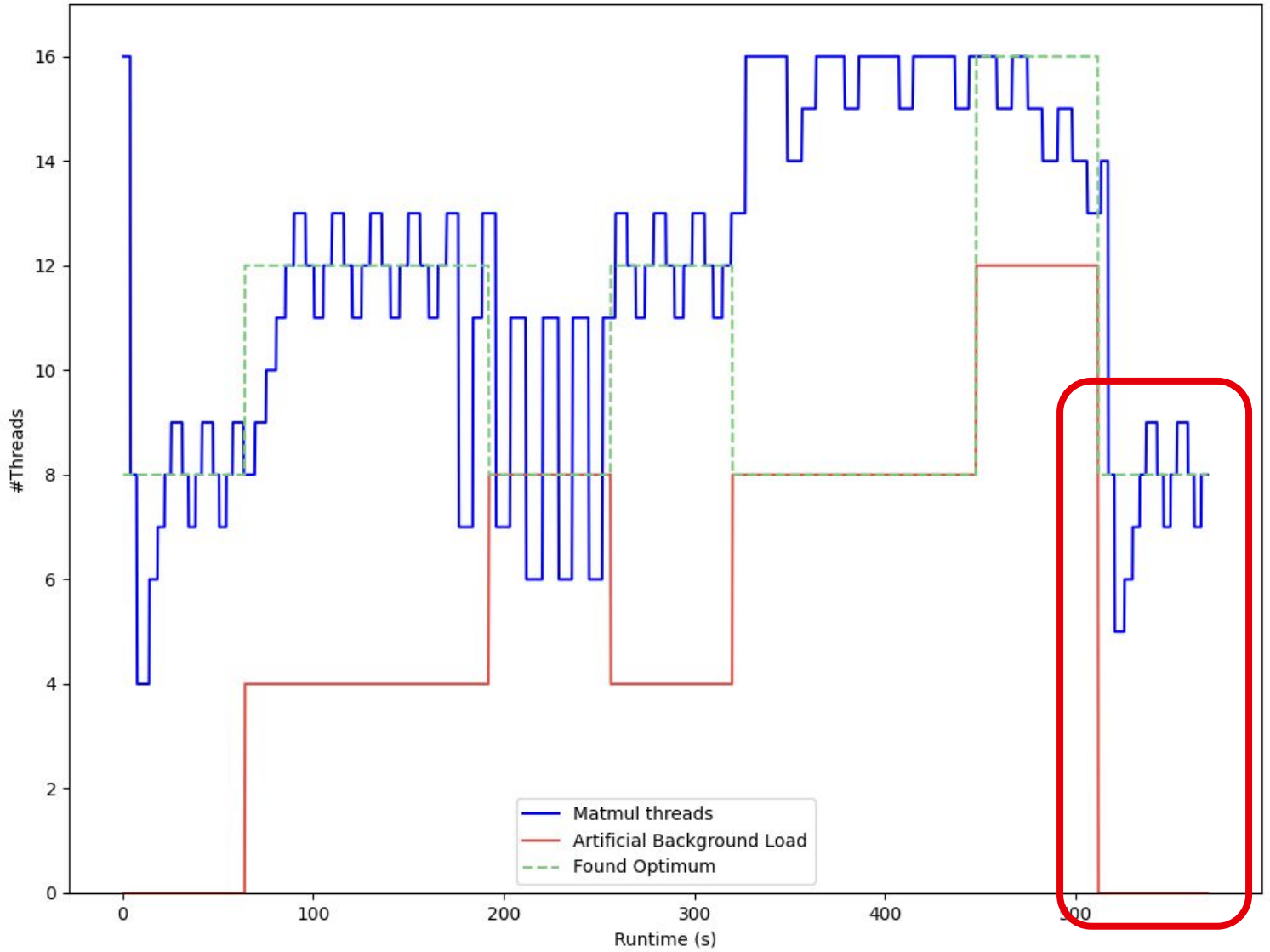




Changing background load

Results

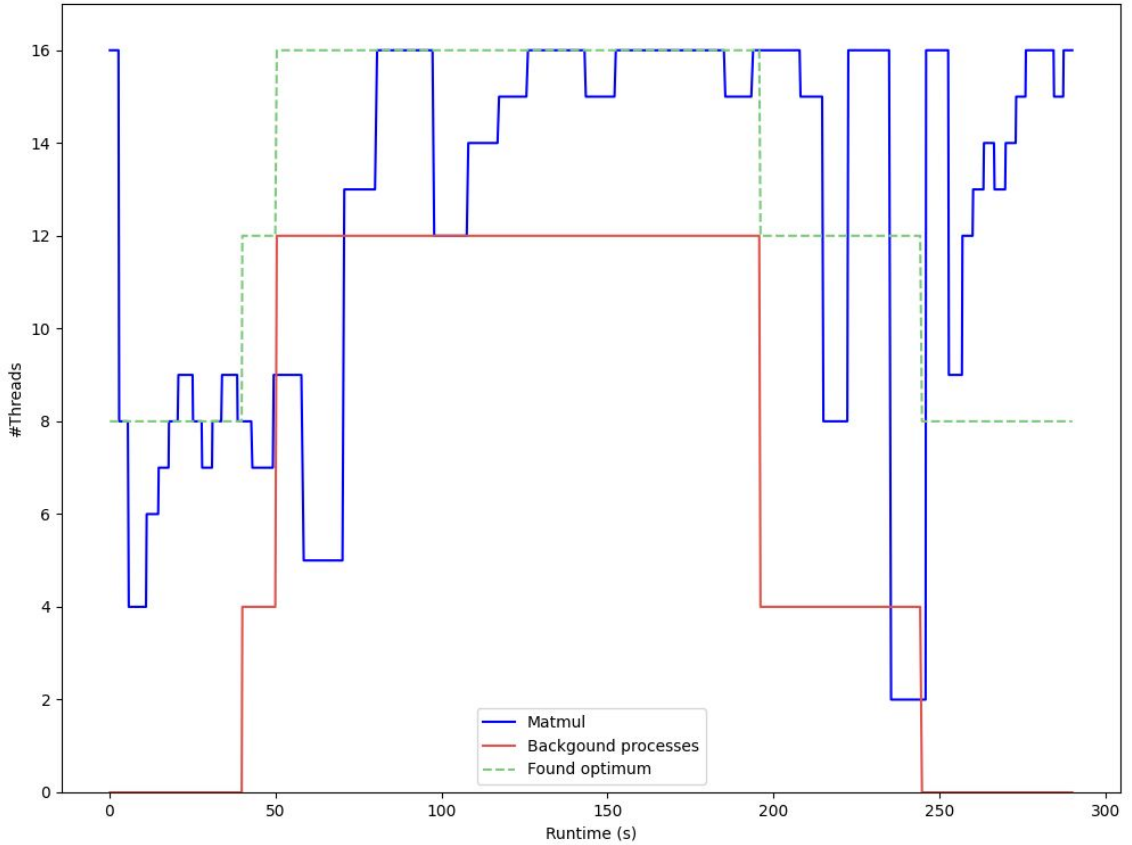




Results

Energy needed for 1300 matrix multiplications

- Changing background load
- Background load no longer artificial

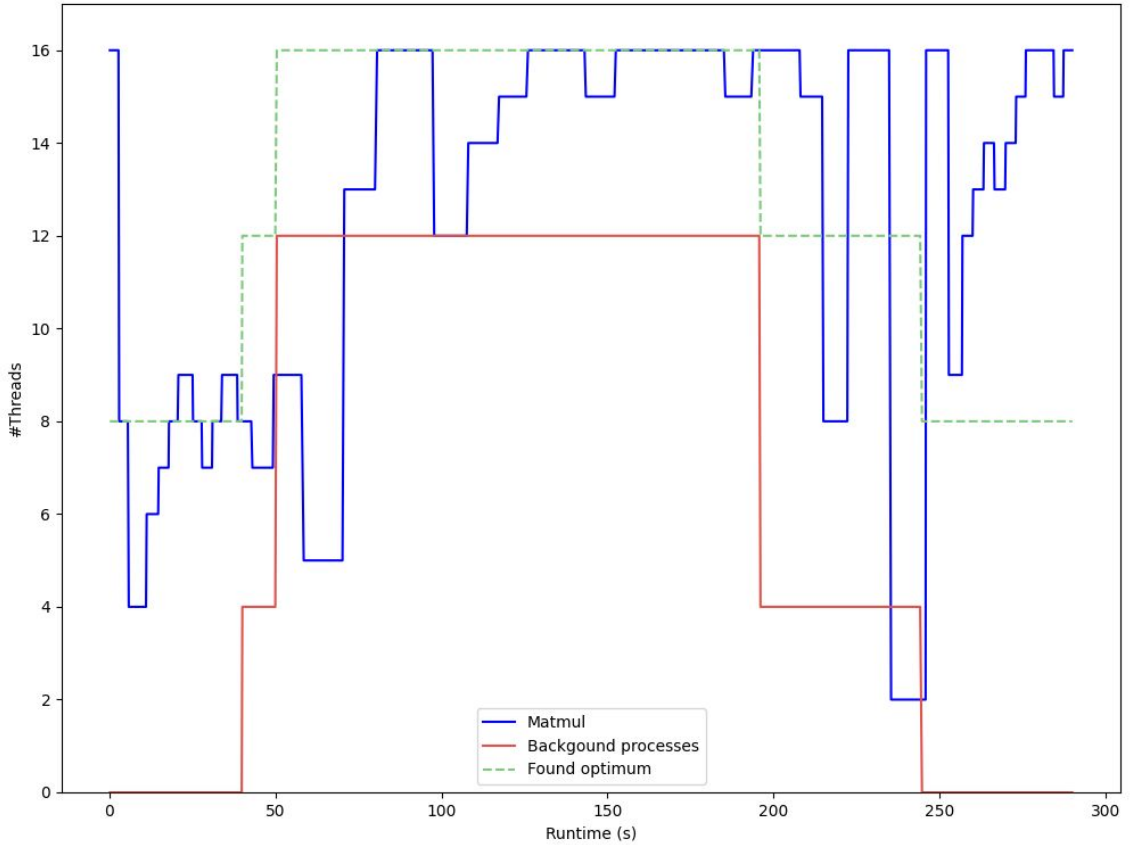


	Dynamic	Static 8 Threads	Static 12 Threads	Static 16 Threads
Energy	12.8kJ	13.6kJ	14.0kJ	14.8kJ
Runtime	290s	302s	303s	322s

Results

Energy needed for 1300 matrix multiplications

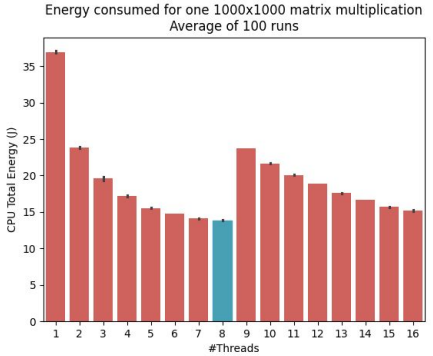
- Changing background load
- Background load no longer artificial

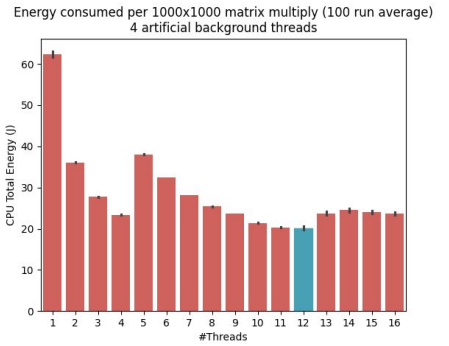


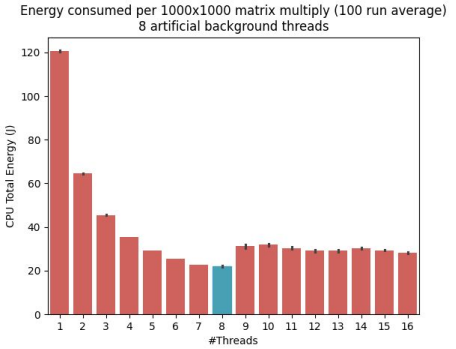
	Dynamic	Static 8 Threads	Static 12 Threads	Static 16 Threads
Energy		-6.3%	-9.4%	-15.6%
Runtime		-4.1%	-4.5%	-11.0%

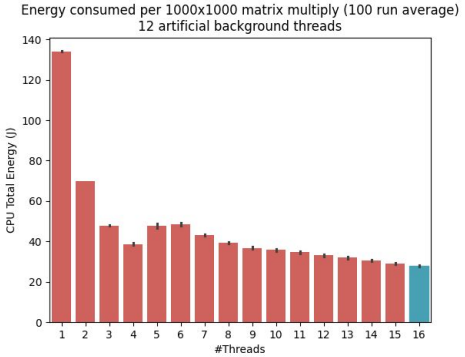
How much do we lose for being dynamic?

Overhead









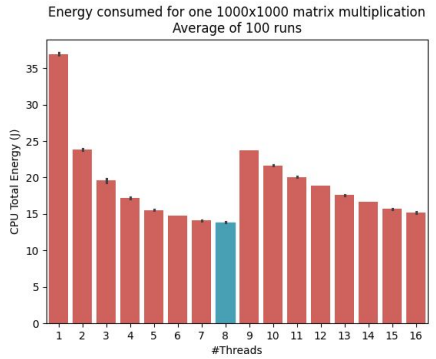
Magic oracle that knows the optimal thread count

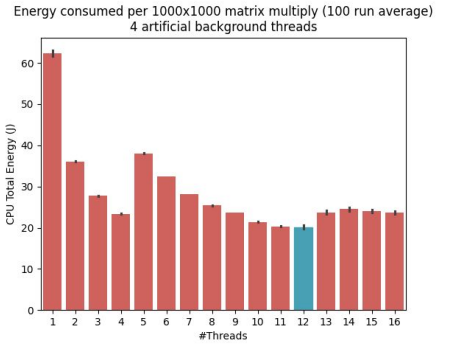
- For the task at hand
- For the given system
- For the given (artificial) background load

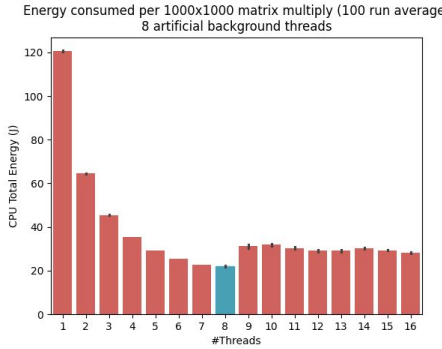
		No BG load	4 BG threads	8 BG threads	12 BG threads
Dynamic	**Energy**	7.11kJ	**11.17kJ**	**8.94kJ**	14.19kJ
	Runtime	165s	278s	197s	342s
Oracle	**Energy**	- *	**10.72kJ**	**7.51kJ**	13.88kJ
	Runtime	- *	236s	157s	331s

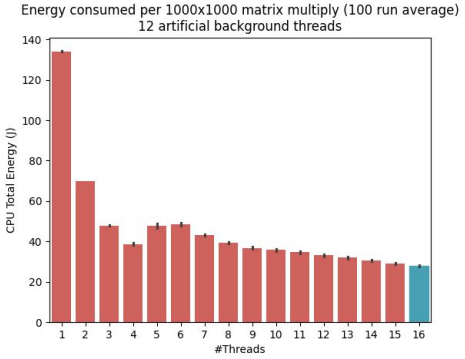
How much do we lose for being dynamic?

Overhead









Magic oracle that knows the optimal thread count

- For the task at hand
- For the given system
- For the given (artificial) background load

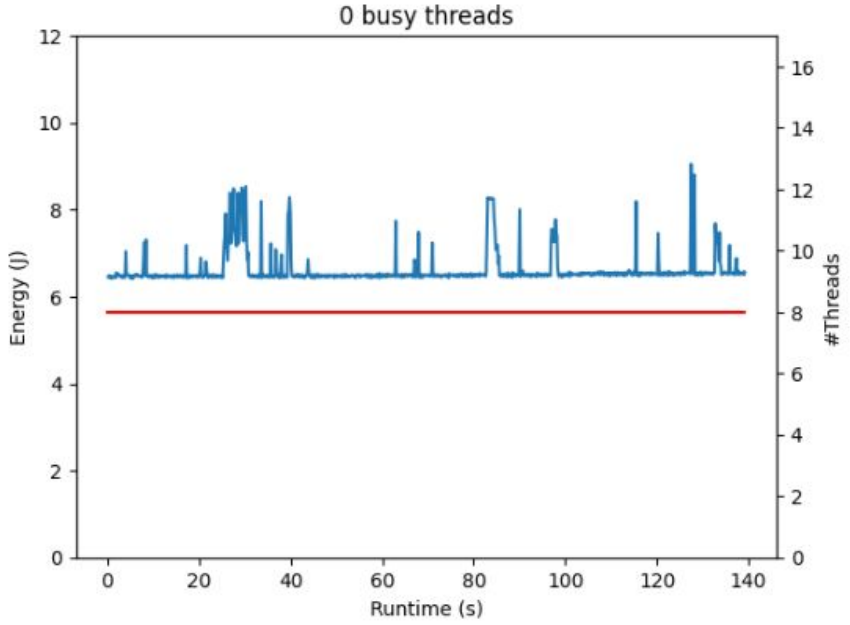
		No BG load	4 BG threads	8 BG threads	12 BG threads
Dynamic	**Energy**	- *	-4.2%	-19.0%	-2.2%
	Runtime	- *	-17.8%	-25.5%	-3.3%

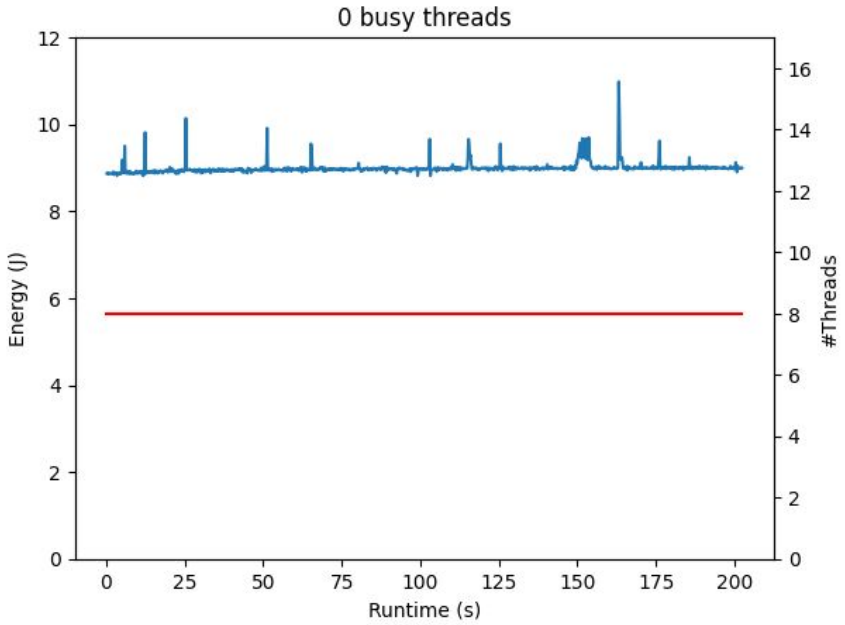
How much do we lose for being dynamic?

Overhead *

Even on a relatively well controlled device we have a lot of variation

- Why? Hard to tell!
- Future work: pinning threads might reduce variation
- Anecdotally: less variation with dynamic adaptation





Conclusion

Language-independent
- Functional languages especially suitable
- POC in SaC

Already shows potential to save energy
- Able to adapt to runtime changes
- Reasonable compared to magic oracle
- Other types of programs?
- Other ways to measure?

6 - 15% less energy consumed vs. static