Variability-Guided Optimization

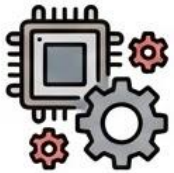
Eitan Frachtenberg, Viyom Mittal, Mohammed Baydoun, Aditya Dhakal, Izzat El Hajj, Dejan Milojicic

HPE Labs

AMERICAN UNIVERSITY OF BEIRUT

Modern systems performance is non-deterministic

Multiple factors and optimizations combine to make performance unpredictable



Hardware

Memory hierarchy, speculative execution, hyperthreading, frequency scaling

Operating systems

Time sharing and space sharing, interrupts, scheduling decisions

Runtimes and Applications

Garbage collection, randomized algorithms, asynchronous coroutines and concurrency

External dependencies

Accelerators, variable I/O, environmental conditions

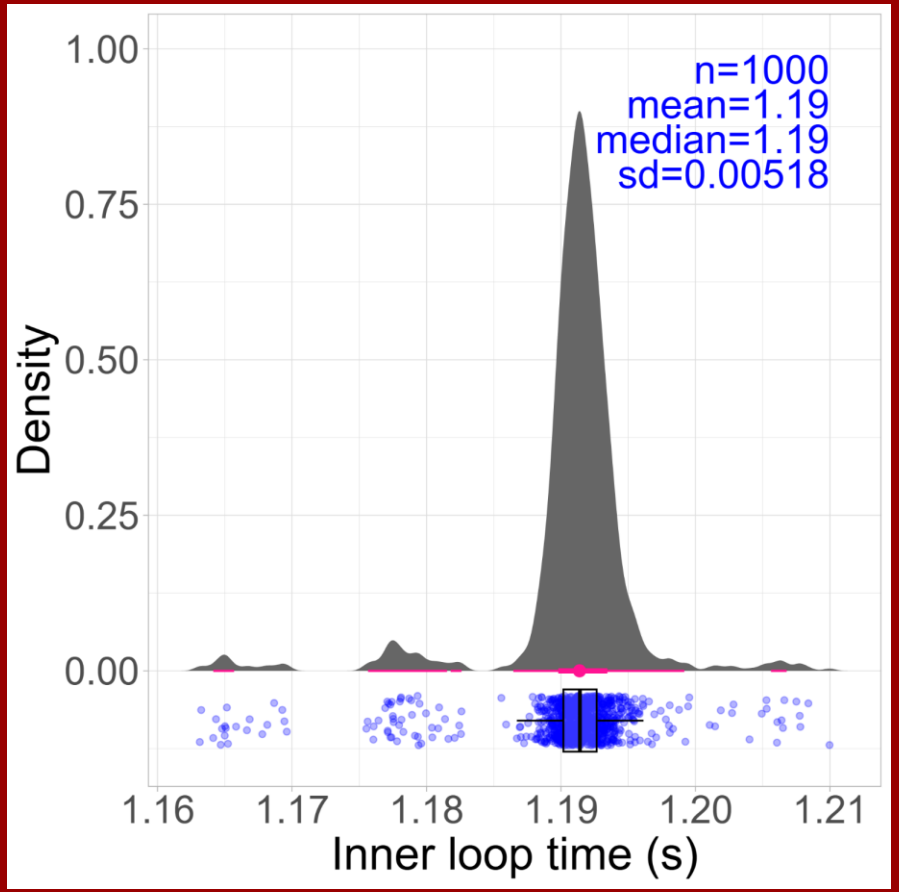
As good as it gets nowadays:

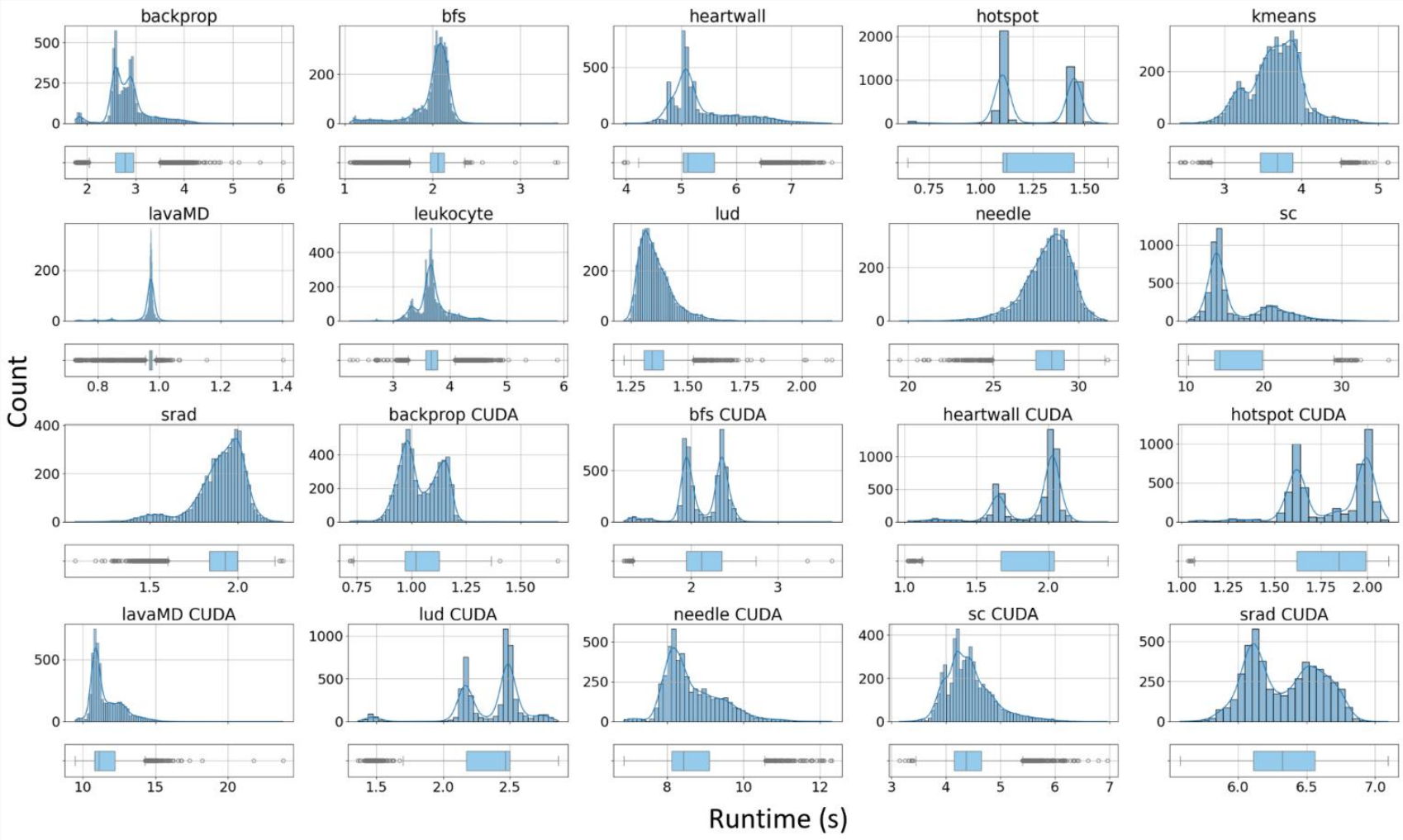
Experiment: 1,000 repetitions of a single-threaded numpy array increment (10M integers) on a dedicated, high-performance dual Xeon server.

These are ideal conditions for minimum variability: no interference, no speculative execution.

The performance distribution is highly consistent, but nevertheless shows:

- **Inherent Variability:** Even the simplest, most unassuming code exhibits performance variability on modern hardware, despite having no resource contention or interference.
- **Suboptimal Consistency:** The presence of the left tail proves that the system often fails to consistently reach its maximum attainable performance, even in an idealized environment.




backprop
bfs
heartwall
hotspot
kmeans
lavaMD
leukocyte
lud
needle
sc
srad
backprop CUDA
bfs CUDA
heartwall CUDA
hotspot CUDA
lavaMD CUDA
lud CUDA
needle CUDA
sc CUDA
srad CUDA
Count
Runtime (s)

Reshaping distributions

- **The question**: How can we exploit the information hidden in performance variability to reduce it and to improve performance?
- **The answer**: Variability-guided optimization

Traditional Approach: “Peak” Optimization

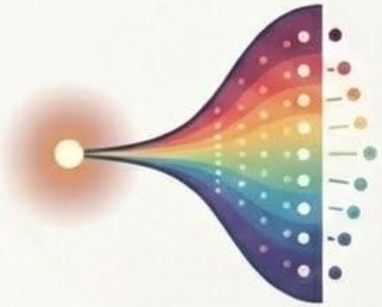
Target:
Minimize the arithmetic mean or median execution time.

Method:
Profile code, identify the "hot path" and apply deterministic transforms (unrolling, vectorization).

Assumption:
Performance is a single, stable point; variability is “noise” to be averaged out or ignored.

The Result:
Significant gains in the average case, but often at the cost of unpredictable tail behavior or system jitter.

VGO Approach: Distribution Optimization

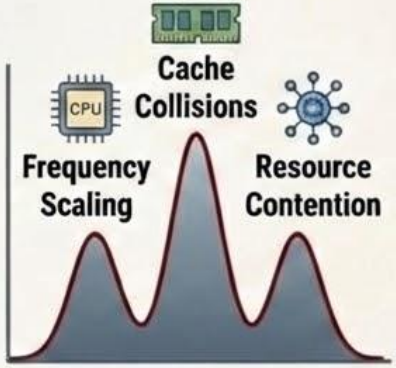


The Premise

Modern systems are non-deterministic. The same code on the same chip produces a spectrum of results, not a point.

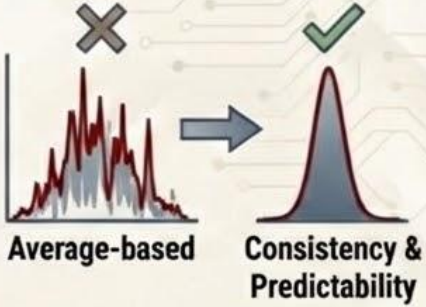
Latent Information

Why did the system run 15% faster at the 5th percentile? Can we stabilize those "lucky" hardware states?



The Modes

Multimodal distributions often signal hidden resource contention or state-switching.

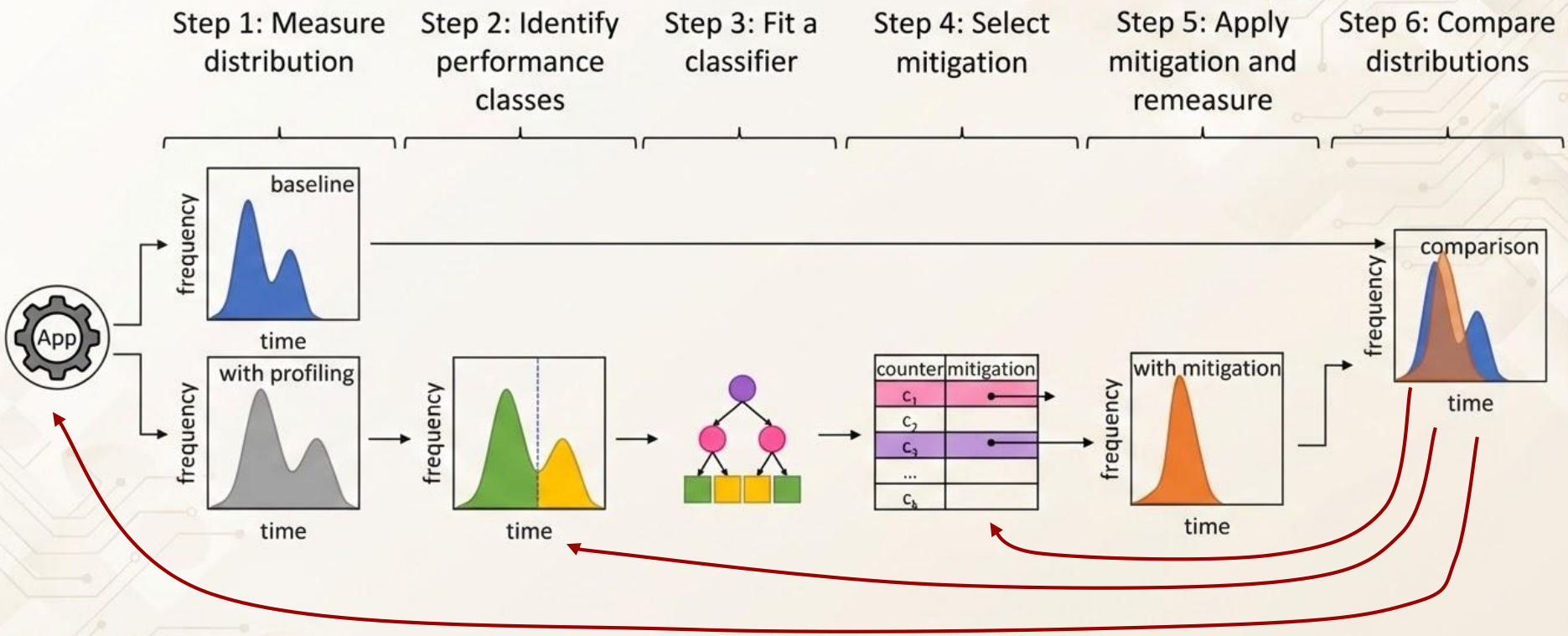


The VGO Thesis

Treating the distribution as a first-class citizen moves us from optimizing for "average" to optimizing for consistency and predictability.

The opportunity: VGO and traditional optimizations are complementary, working together for superior system performance.

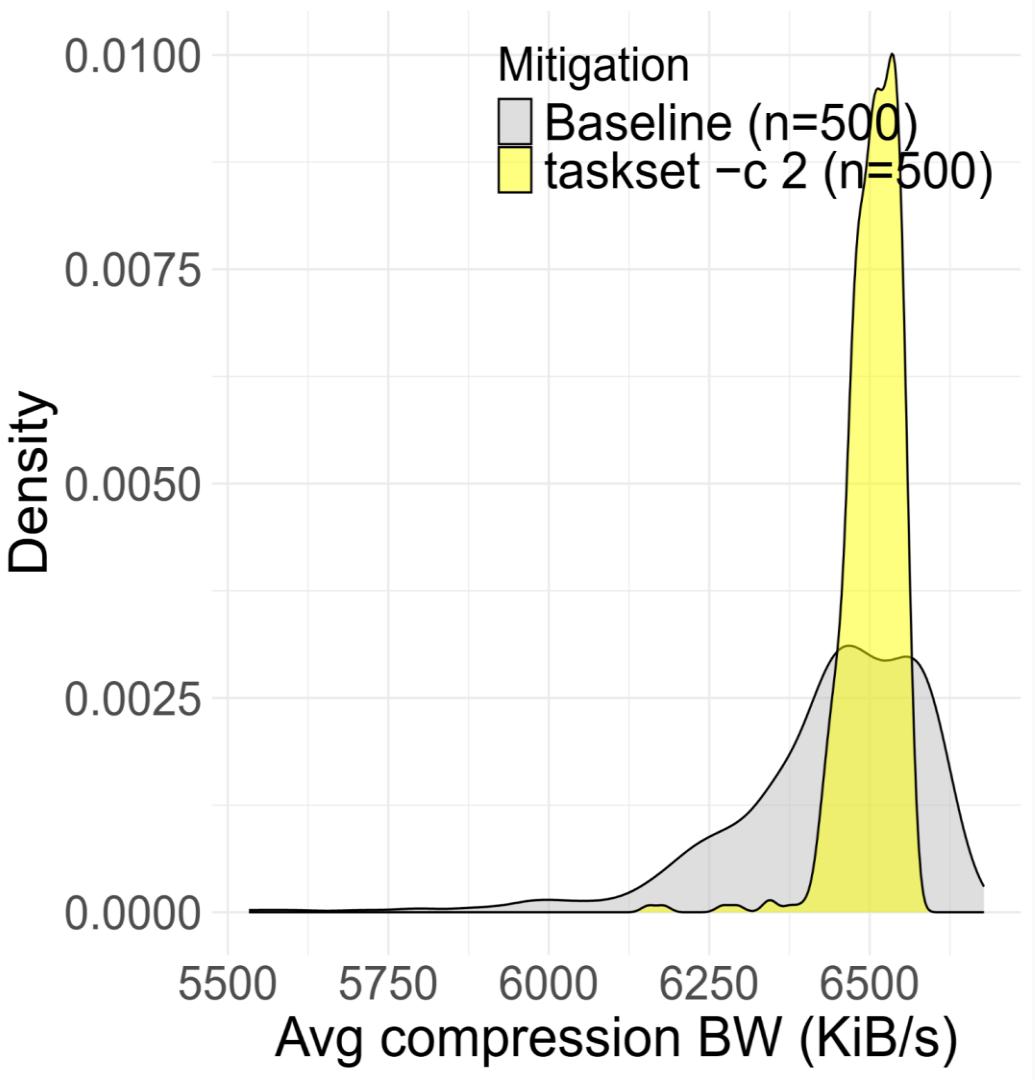
VGO Overview



Benchmark:
7z compression,
Single-thread CPU

Influential factor:
Context switches

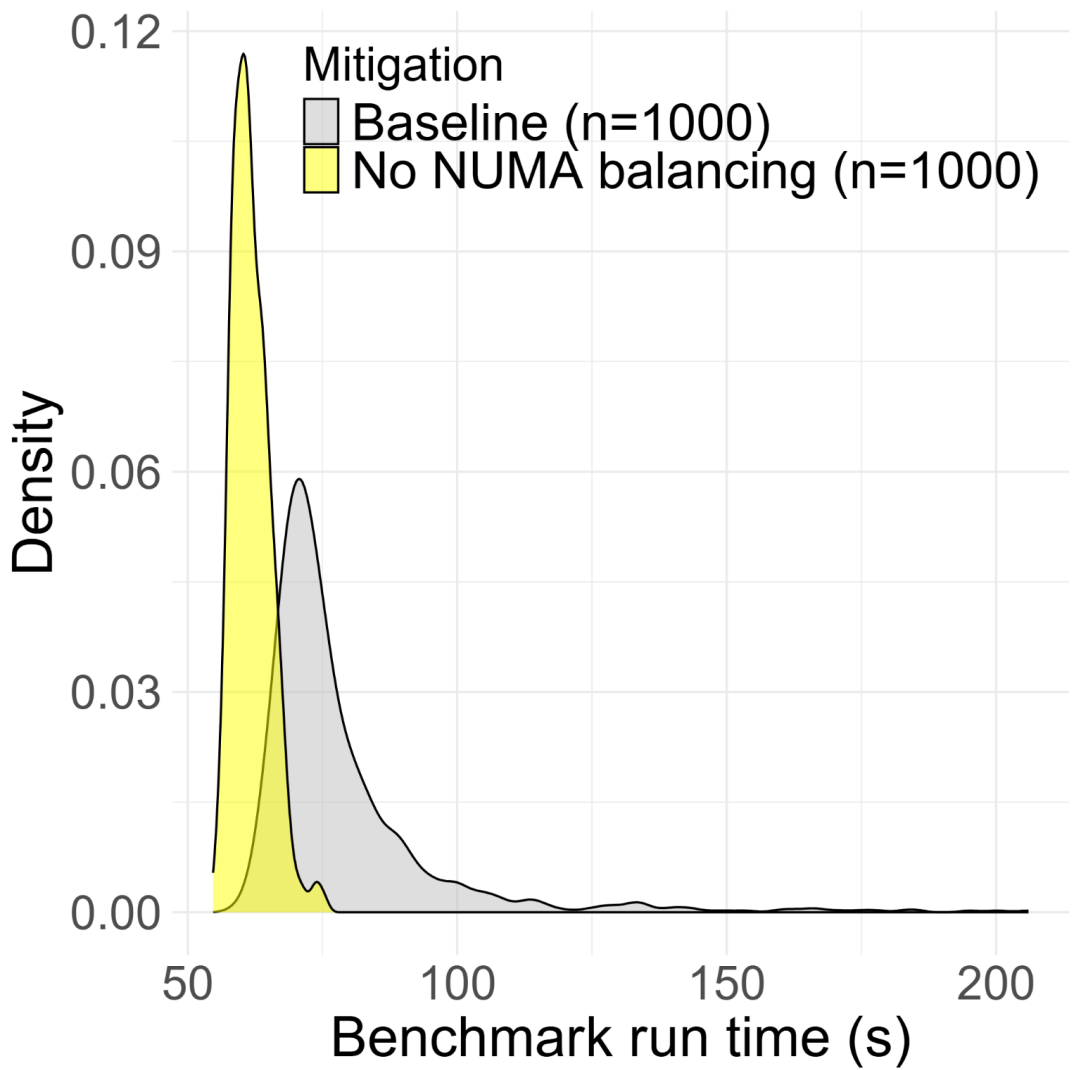
Mitigation:
Thread pinning



Benchmark:
srad,
Multi-thread CPU

Influential factor:
CPU migrations

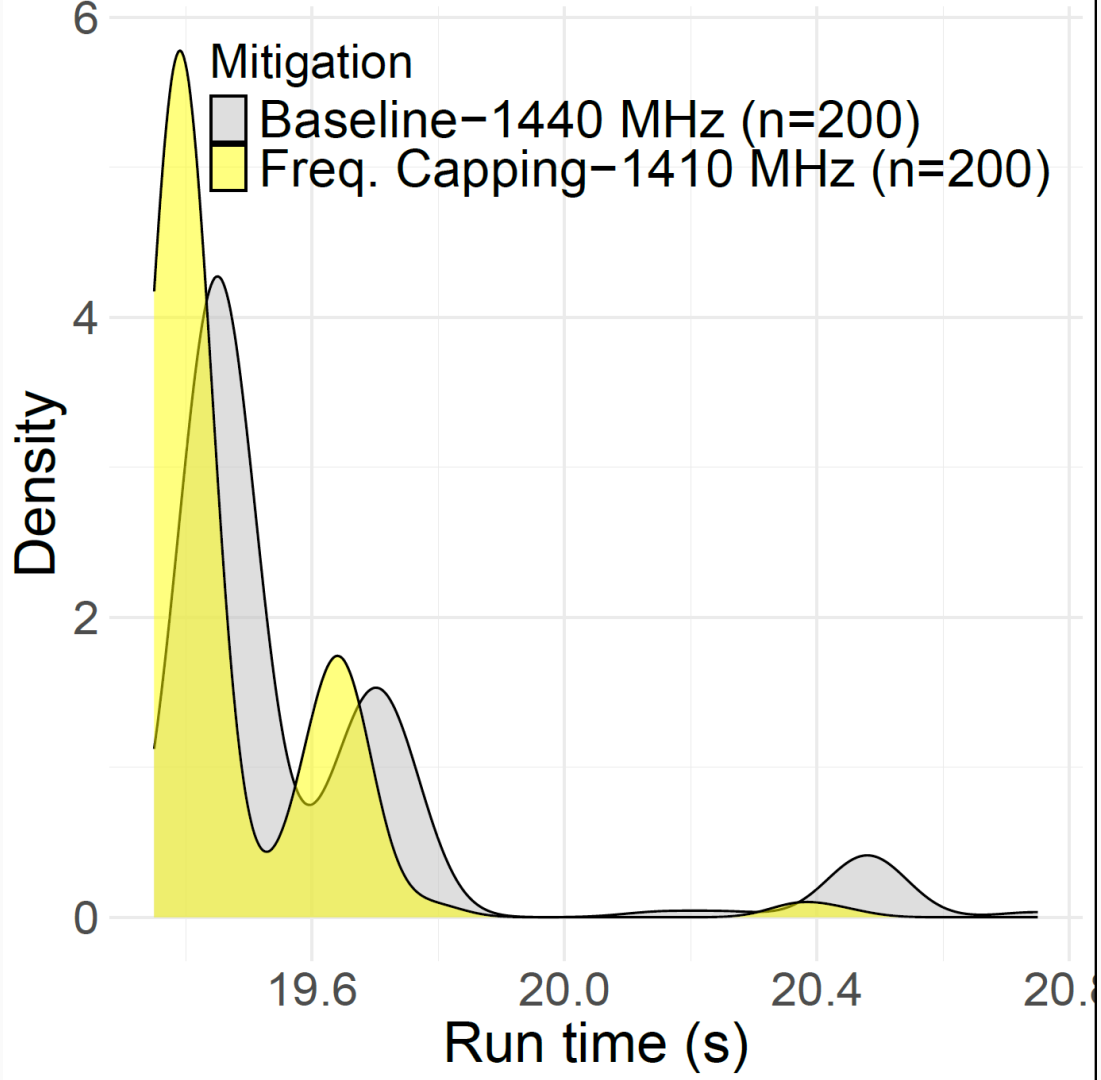
Mitigation:
Disable NUMA rebalancing in kernel

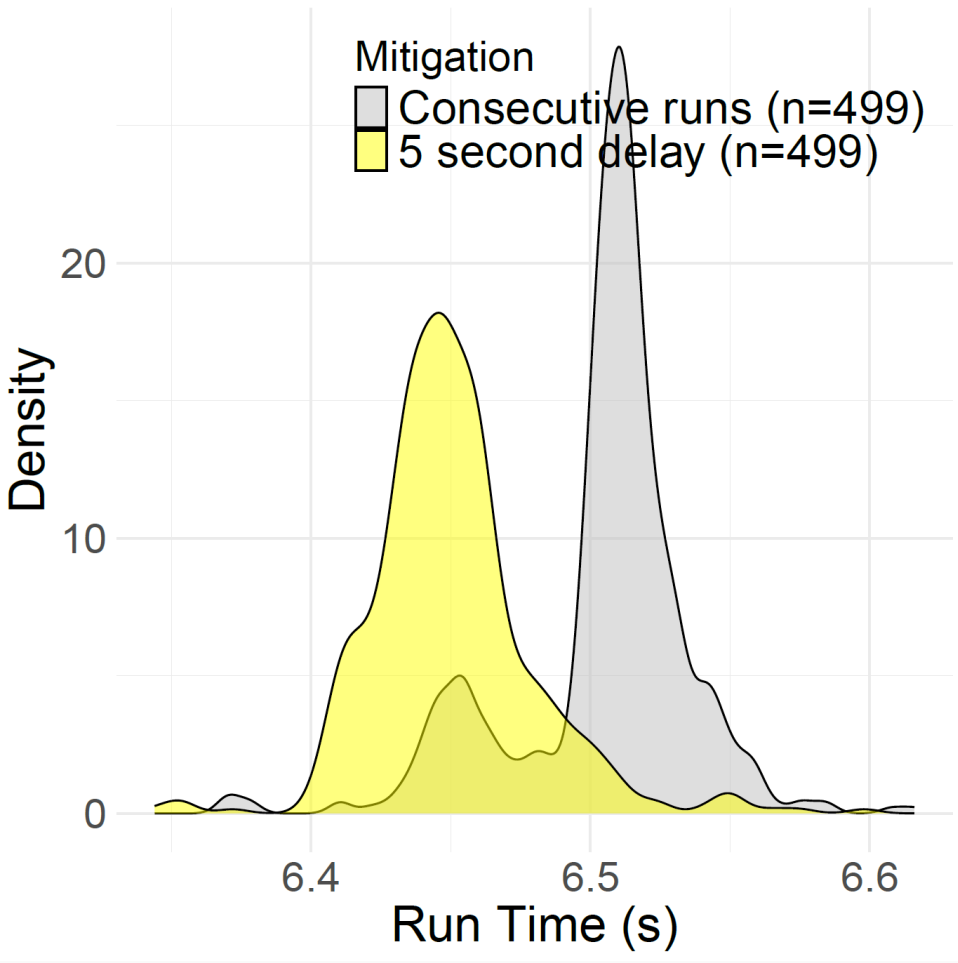


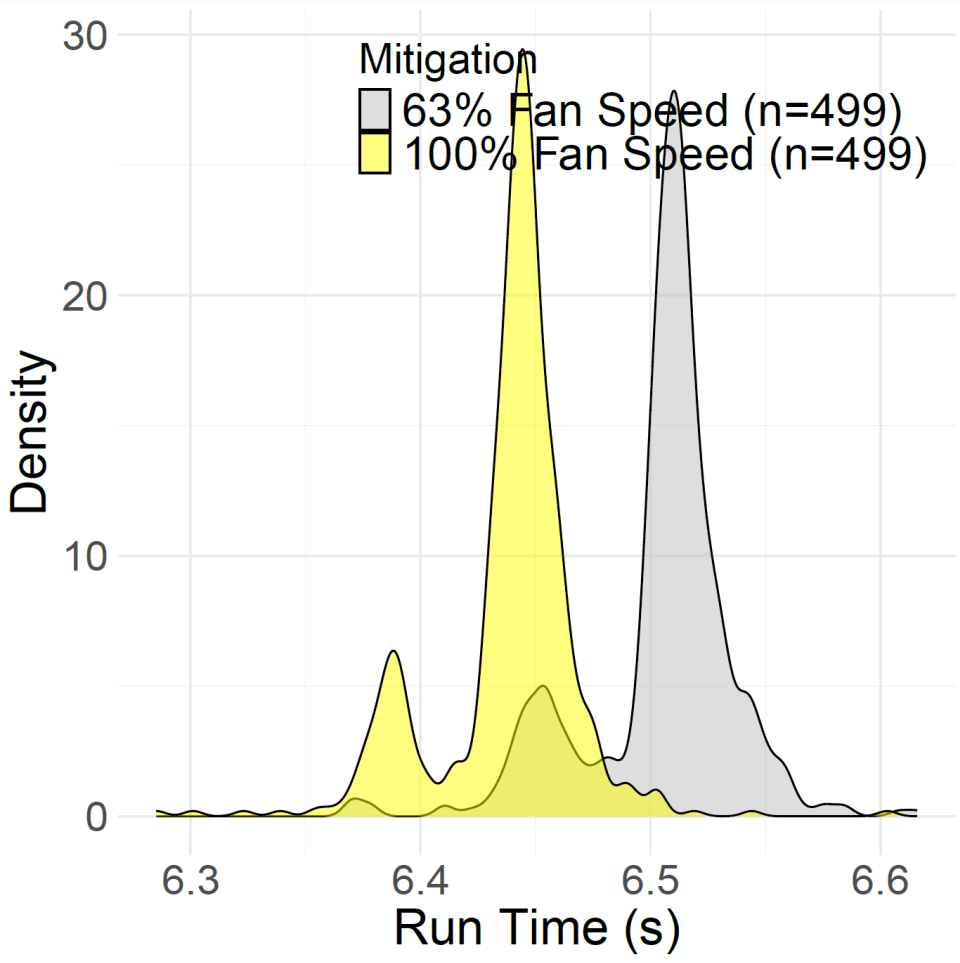
Benchmark:
gpt-oss-20b LLM, A100 GPU

Influential factor:
GPU clock frequency variance

Mitigation:
Frequency capping







Benchmark: Stable diffusion, A100 GPU. **Influential factor**: temperature. **Mitigations**: Cooldown delay; server fan speed

Selected benchmark summary

HW	Benchmark (release year)	System factor	Mitigation	Before mitigation			After mitigation			Reduction ratio		
				Mean	SD	CV	Mean	SD	CV	Mean	SD	CV
CPU	7z (2016)	Context switches	Pin threads	27.553	0.352	0.013	27.416	0.056	0.002	0.995×	0.159×	0.160×
	lbm (2012)	Context switches	Transparent huge pages	13.258	0.803	0.061	11.141	0.272	0.024	0.840×	0.339×	0.403×
	sad (2012)	TLB misses	Disable NUMA balancing	2.385	0.018	0.058	2.245	0.015	0.007	0.941×	0.833×	0.885×
	Xapian (2011)	CPU migrations	Disable NUMA balancing	78.542	17.541	0.223	13.266	3.500	0.043	0.981×	0.200×	0.254×
	srad (2009)	CPU migrations	Disable NUMA balancing	6.657	0.817	0.123	5.669	0.318	0.054	0.882×	0.045×	0.441×
	lavaMD (2009)	Cache misses	Disable SMT	3.465	0.817	0.011	3.465	0.034	0.037	0.997×	0.045×	0.651×
GPU	srad (2009)	Kernel launch overhead	Optimize streams usage	42.645	0.625	0.015	20.065	0.065	0.004	0.471×	0.138×	0.292×
	Retinanet (2018)	CUDA initialization	Avoid re-initialization	16.381	1.395	0.085	11.755	0.480	0.041	0.718×	0.344×	0.779×
	Stable diffusion	Temperature	Increase fan speed	6.505	0.032	0.005	6.438	0.030	0.005	0.990×	0.938×	0.947×
	(2023)	Temperature	Frequency capping	6.438	0.030	0.005	6.422	0.026	0.004	0.990×	0.867×	0.869×
	gpt-oss-20b (2025)	Frequency	Frequency capping	19.516	0.030	0.005	19.469	0.160	0.008	0.998×	0.453×	0.544×
		Frequency	Frequency capping	19.516	0.333	0.055	19.469	0.160	0.008	0.998×	0.830×	0.828×
	Yi-6B (2025)	Frequency	Frequency capping	20.553	0.477	0.023	20.433	0.391	0.019	0.994×	0.820×	0.824×
Geometric mean				**20.553**	**0.477**	**0.023**	**20.433**	**0.391**	**0.019**	**0.843×**	**0.374×**	**0.444×**

Conclusion & Future work

- **The distribution is the performance**: Summarizing results into a single number isn't just an approximation—it's often a misrepresentation of system behavior.
- **Variability is the signal**: Non-determinism is an opportunity. When we treat variability as a first-class citizen, we can diagnose and **mitigate** the root causes of performance jitter.
- **The future is automation**: Much of the process of root-cause analysis of performance variability, hypothesis falsification, and effective mitigations can be automated, partially with the help of LLMs.

Thank you!

For follow-up questions:
eitan.frachtenberg@hpe.com

To use SHARP/VGO
github.com/HewelettPackard/sharp

“Information is the resolution of uncertainty”

-- Claude Shannon