Carbon Savings + Sim Theory's Thunder SDK: A Case Study

Executive Summary

In November 2025, Sim Theory set out to show how much carbon emissions can be reduced by integrating Thunder SDK into a compute-intensive application on commonly-available cloud and on-premise server infrastructure.

The test was conducted on-premises with three hardware configurations. Sim Theory selected these configurations because they correspond to a wide variety of common processing scenarios. The top-tier system uses the same CPU architecture and memory as modern, compute-optimized, high-performance cloud data center offerings. Mid-tier testing was done on a system equivalent to a slightly below average compute-optimized cloud instance. Our low-tier system is equivalent to an average office desktop capable of general purpose compute loads. Testing was done on on-premise hardware as cloud providers do not make total power draw or specific consumption statistics available to customers.

The Results

After integration with Thunder SDK, the test application completed the same work in around **1/10th of the time** and **cut carbon emissions by an average of 88.5%**. Carbon savings are realized by using Thunder SDK to harness the entire power of the CPU to complete the work in a significantly shorter period of time.

Time and Carbon Savings with Sim Theory's Thunder SDK

	Time Savings	Carbon Savings
Top-Tier	96.7%	**93.1%**
Mid-Tier	91.4%	**84.6%**
Low-Tier	90.9%	**87.8%**

Hypothesis

Sim Theory delivers significant carbon footprint reductions across both cloud and on-premise hardware when we use Thunder SDK to maximize the parallel execution of work on high-performance computing tasks such as AI, digital twins, engineering simulations, and media trans-coding. We expect to see a higher power draw by the CPU when it is operating at near-full capacity, but we expect the overall power requirements to shrink substantially as the CPU is processing the job for a much shorter period of time.

Testing

The testing process involved resizing 4,578 image files totaling 19.29GB. The files were each originally 4k by 4k pixels and were resized down to 100 by 100 pixels while maintaining the existing aspect ratio. The output format was png. Additionally, EXIF and XMP data was extracted if it was present in the original image.

magick.NET[1] was the application Sim Theory chose to complete this work. It is a commonly used image manipulation package which leverages the powerful ImageMagick[2] image manipulation library. It was selected due to the extensive feature set provided and the ease of integrating Thunder SDK. magick.NET also allows Sim Theory to design a real-world test of Thunder SDK cross-language bindings.

The test was run twice per system, once using the default threading behavior of magick.NET and once using Thunder SDK to schedule work across up to 85% of the total available threads. Testing was limited to 85% of the total available threads so as not to interfere with basic operating system function.

Integration

The work necessary to integrate Thunder SDK with the project source code was what Sim Theory refers to as a basic, high-level integration.

- ➔ The pre-compiled Thunder SDK was placed next to the existing project structure.
- ➔ Only the publicly available features and APIs of magick.NET were used.
- ➔ Sim Theory's optimized concurrency runtime library was integrated by adding the appropriate paths and using the STI namespace.
- ➔ A small library was written to manage application input and output and to set up the Sim Theory Scheduler to execute work in parallel.
- ➔ The testing data was highly parallelizable without data dependencies to define.
- ➔ Thunder SDK contains C# bindings and magick.NET is implemented in C#. Neither code boundaries or data had to be managed.

[1] https://github.com/dlemstra/Magick.NET
[2] https://imagemagick.org/index.php

Results

Top-Tier Results

The top-tier system used for testing was on loan from AMD and was an:

- ➔ AMD Ryzen Threadripper PRO 7955WX 16-Core[3] Processor @ 4.5GHz, Windows 11 system with 128GB of DDR5 RAM
- ➔ It is most equivalent to a c7a.8xlarge AWS instance.

This processor was chosen for testing specifically because it uses the Zen 4 architecture, which is the same architecture used in the EPYC server CPUs used by Amazon, Google, IBM, Microsoft, and Oracle in their high-performance cloud computing infrastructure. Additionally, the RAM is equivalent to what is used in that infrastructure.

Default magick.NET Results

Testing was completed using the default threading behavior of magick.NET, which consumed the following resources during a total runtime of **21 minutes and 45.5 seconds**:

	Joules per Second	**Joules to kWh per Second**	**Total Power Consumption kWh**
Median Total Use	174.5	.000048472	.063281989

	USA (400 gCO_2 produced per kWh)	**Germany** (315 gCO_2 produced per kWh)	**Australia** (444 gCO_2 produced per kWh)	**India** (559 gCO_2 produced per kWh)	**World** (473 gCO_2 produced per kWh)
Total Carbon Footprint[4]	**25.313 gCO_2**	**19.934 gCO_2**	**28.097 gCO_2**	**35.375 gCO_2**	**29.932 gCO_2**

[3] 16 physical cores with 2 logical counts per core - 32 total cores, test ran using 28 cores

[4] The country-specific grams of CO_2 per kWh come from electricitymaps.com using the average of the last 12 months of data. The world average of grams of CO_2 comes from the 2024 average according to electricitymaps.com.

magick.Net + Thunder SDK Results

Testing was completed using Thunder SDK to distribute work across 28 threads, which took **42.6 seconds** and consumed the following resources:

	Joules per Second	Joules to kWh per Second	Total Power Consumption kWh
Median Total Use	367.5	.000102083	.004346504
Delta Between Default and Thunder SDK	Increased by 193	Increased by .000053611	**Decreased by .058935485**

	USA (400 gCO_2 produced per kWh)	**Germany** (315 gCO_2 produced per kWh)	**Australia** (444 gCO_2 produced per kWh)	**India** (559 gCO_2 produced per kWh)	**World** (473 gCO_2 produced per kWh)
Total Carbon Footprint[5]	1.74 gCO_2	1.37 gCO_2	1.93 gCO_2	2.43 gCO_2	2.06 gCO_2
Total Reduction	**23.58 gCO_2**	**18.56 gCO_2**	**26.17 gCO_2**	**32.94 gCO_2**	**27.88 gCO_2**
93.13% Total Carbon Savings					

[5] The country-specific grams of CO_2 per kWh come from electricitymaps.com using the average of the last 12 months of data. The world average of grams of CO_2 comes from the 2024 average according to electricitymaps.com.

Mid-Tier Results

The mid-tier system used for testing was an:

- ➔ AMD Ryzen Threadripper 2990WX 32-Core[6] Processor @ 3.0GHz, Windows 10 system with 128GB of DDR4 RAM
- ➔ It is most equivalent to a c5a.16xlarge AWS instance.

Default magick.NET Results

Testing was completed using the default threading behavior of magick.NET, which consumed the following resources during a total runtime of **39 minutes and 39.9 seconds**:

	Joules per Second	Joules to kWh per Second	Total Power Consumption kWh
Median Total Use	150.1	.000041694	.099229984

	USA (400 gCO_2 produced per kWh)	Germany (315 gCO_2 produced per kWh)	Australia (444 gCO_2 produced per kWh)	India (559 gCO_2 produced per kWh)	World (473 gCO_2 produced per kWh)
Total Carbon Footprint[7]	**39.692 gCO_2**	**31.257 gCO_2**	**44.058 gCO_2**	**55.470 gCO_2**	**46.936 gCO_2**

magick.Net + Thunder SDK

Testing was completed using Thunder SDK to distribute work across 56 threads, which took **3 minutes and 25.6 seconds** and consumed the following resources:

	Joules per Second	Joules to kWh per Second	Total Power Consumption kWh
Median Total Use	267.8	.000074389	.015292719
Delta Between Default and Thunder SDK	Increased by 117.7	Increased by .000032695	**Decreased by .083937265**

[6] 32 physical cores with 2 logical counts per core - 64 total cores, test ran using 56 cores

[7] The country-specific grams of CO_2 per kWh come from electricitymaps.com using the average of the last 12 months of data. The world average of grams of CO_2 comes from the 2024 average according to electricitymaps.com.

	USA (400 gCO_2 produced per kWh)	Germany (315 gCO_2 produced per kWh)	Australia (444 gCO_2 produced per kWh)	India (559 gCO_2 produced per kWh)	World (473 gCO_2 produced per kWh)
Total Carbon Footprint[8]	6.12 gCO_2	4.82 gCO_2	6.79 gCO_2	8.55 gCO_2	7.23 gCO_2
Total Reduction	**33.57 gCO_2**	**26.44 gCO_2**	**37.27 gCO_2**	**46.92 gCO_2**	**39.7 gCO_2**
84.59% Total Carbon Savings					

Low-Tier Results

The low-tier system used for testing was an:

- ➔ AMD Ryzen 9 6900HS 8-Core[9] Processor @ 3.3GHz, Windows 11 laptop with 40GB of DDR4 RAM
- ➔ It is most equivalent to a m6a.4xlarge AWS instance.

Default magick.NET Results

Testing was completed using the default threading behavior of magick.NET. which consumed the following resources during a total runtime of **67 minutes and 52.6 seconds**:

	Joules per Second	Joules to kWh per Second	Total Power Consumption kWh
Median Total Use	44.23	.000012286	.050036465

[8] The country-specific grams of CO_2 per kWh come from electricitymaps.com using the average of the last 12 months of data. The world average of grams of CO_2 comes from the 2024 average according to electricitymaps.com.
[9] 8 physical cores with 2 logical counts per core - 16 total cores, test ran using 14 cores

	USA (400 gCO_2 produced per kWh)	Germany (315 gCO_2 produced per kWh)	Australia (444 gCO_2 produced per kWh)	India (559 gCO_2 produced per kWh)	World (473 gCO_2 produced per kWh)
Total Carbon Footprint[10]	**20.015 gCO_2**	**15.761 gCO_2**	**22.216 gCO_2**	**27.970 gCO_2**	**23.667 gCO_2**

magick.Net + Thunder SDK Results

Testing was completed using Thunder SDK to distribute work across 14 threads, which took **6 minutes and 12.2 seconds** and consumed the following resources:

	Joules per Second	Joules to kWh per Second	Total Power Consumption kWh
Median Total Use	58.97	.000102083	.004346504
Delta Between Default and Thunder SDK	Increased by 14.74	Increased by .000089797	**Decreased by .045689961**

	USA (400 gCO_2 produced per kWh)	Germany (315 gCO_2 produced per kWh)	Australia (444 gCO_2 produced per kWh)	India (559 gCO_2 produced per kWh)	World (473 gCO_2 produced per kWh)
Total Carbon Footprint[11]	2.44 gCO_2	1.92 gCO_2	2.71 gCO_2	3.41 gCO_2	2.88 gCO_2
Total Reduction	**17.58 gCO_2**	**13.84 gCO_2**	**19.51 gCO_2**	**24.56 gCO_2**	**20.78 gCO_2**
87.82% Total Carbon Savings					

[10] The country-specific grams of CO_2 per kWh come from electricitymaps.com using the average of the last 12 months of data. The world average of grams of CO_2 comes from the 2024 average according to electricitymaps.com.
[11] The country-specific grams of CO_2 per kWh come from electricitymaps.com using the average of the last 12 months of data. The world average of grams of CO_2 comes from the 2024 average according to electricitymaps.com.

Sim Theory would like to acknowledge and thank AMD for partnering with us and providing test hardware for this case study.

Author: Randy Culley, CTO
sales@simtheory.com
simtheory.com