SIM THEORY

Time and Energy Efficiency + Sim Theory's Thunder SDK: A Case Study

Executive Summary

In July 2025, Sim Theory set out to show how much time and energy savings Thunder SDK, our compute optimization solution, could achieve on commonly-available cloud and on-premise server infrastructure.

The test was conducted on-premises on three hardware configurations:

- ➔ Top-tier
- ➔ Mid-tier
- ➔ Low-tier

Sim Theory selected these configurations because they cover or approximate a wide variety of common processing scenarios. The top-tier system, for example, uses the same CPU architecture and memory as high-performance cloud data centers. 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

Sim Theory's Thunder SDK accomplished the same work in around **1/10th of the time** and achieved an average of **88.5% energy savings**. Compute cost savings are either realized as shown in the following table as directly proportional to the length of time the cloud instance is busy or in the ability to move to a cheaper instance type, depending on the needs of the customer.

Time, Energy, and Compute Cost Savings with Sim Theory's Thunder SDK

	Time Savings	Energy Savings	Compute Cost Savings[1]
Top-Tier	**96.7%**	93.1%	96.7%
Mid-Tier	**91.4%**	84.6%	91.4%
Low-Tier	**90.9%**	87.8%	90.9%

[1] On the closest equivalent AWS instance

Hypothesis

Sim Theory can show substantial time and energy savings on both cloud and on-premise hardware when we use the Thunder SDK to maximize the parallel execution of work on high-performance computing tasks like AI, simulations, media trans-coding, and other applications.

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.

The test was run twice per system, once using the default threading behavior of magick.NET[2] and once using the 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 the 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.
- The Thunder SDK contains C# bindings and magick.NET is implemented in C#. Neither code boundaries or data had to be managed.

[2] magick.NET (https://github.com/dlemstra/Magick.NET) is a commonly used image manipulation package which leverages the powerful ImageMagick(https://imagemagick.org/index.php) image manipulation library. It was selected due to the extensive feature set provided and the ease of integrating the Thunder SDK. magick.NET also allows Sim Theory to design a real-world test of the Thunder SDK cross-language bindings.

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 Threading Results

Completing testing using the default threading behavior of magick.NET took **21 minutes and 45.5 seconds**. On the equivalent c7a.8xlarge AWS instance, the compute portion would **cost $1.13**[4].

The total power consumption was:

	CPU	System-Wide
Median Total Use	110,017.1 watts	227,815.16 watts

28 Thread Results

Completing testing using the Thunder SDK to distribute work across 28 threads took **42.6 seconds**. The compute portion would **cost $0.04** to run in AWS.

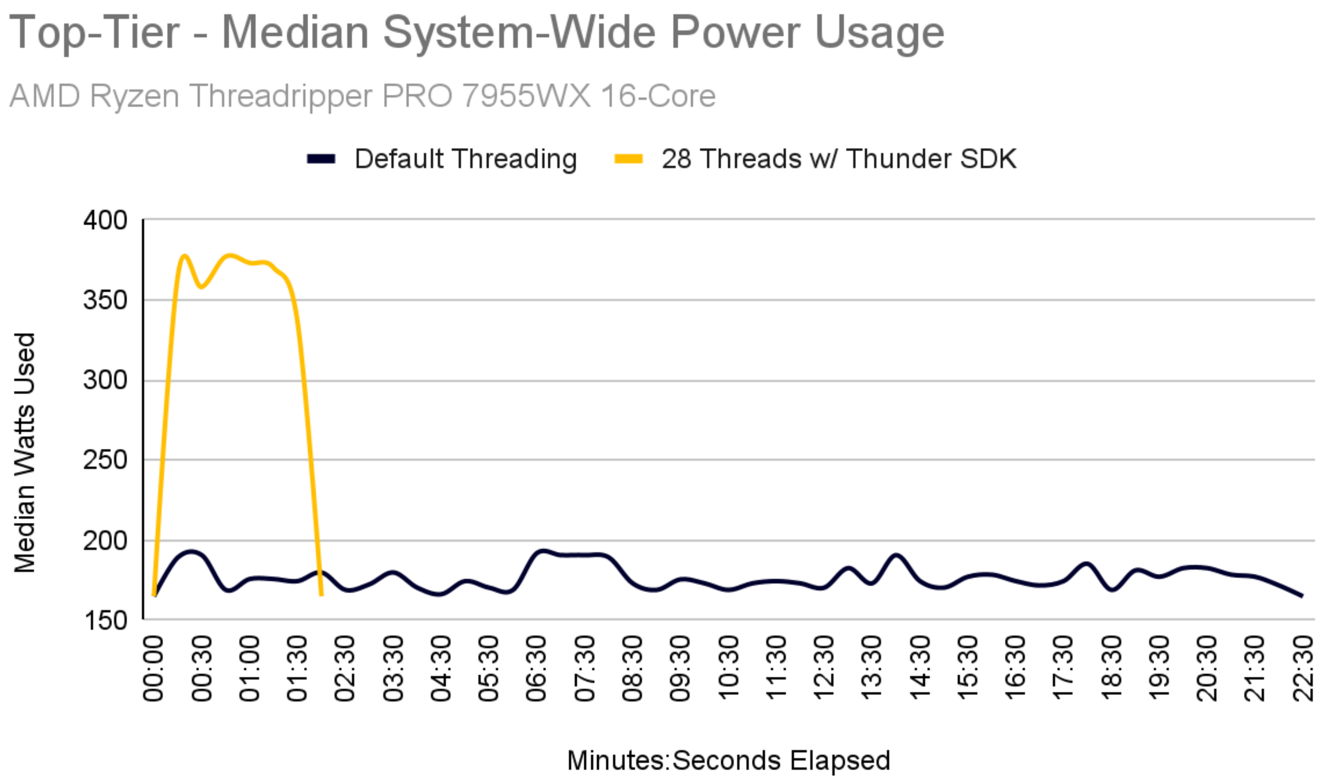
The Thunder SDK shortened the test by **21 minutes and 2.9 seconds, a 96.7% savings, and a compute cost savings in AWS of $1.09.**

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

[4] On an instance in US West (Oregon) running Windows the hourly rate at time of publication is $3.11424/hour.

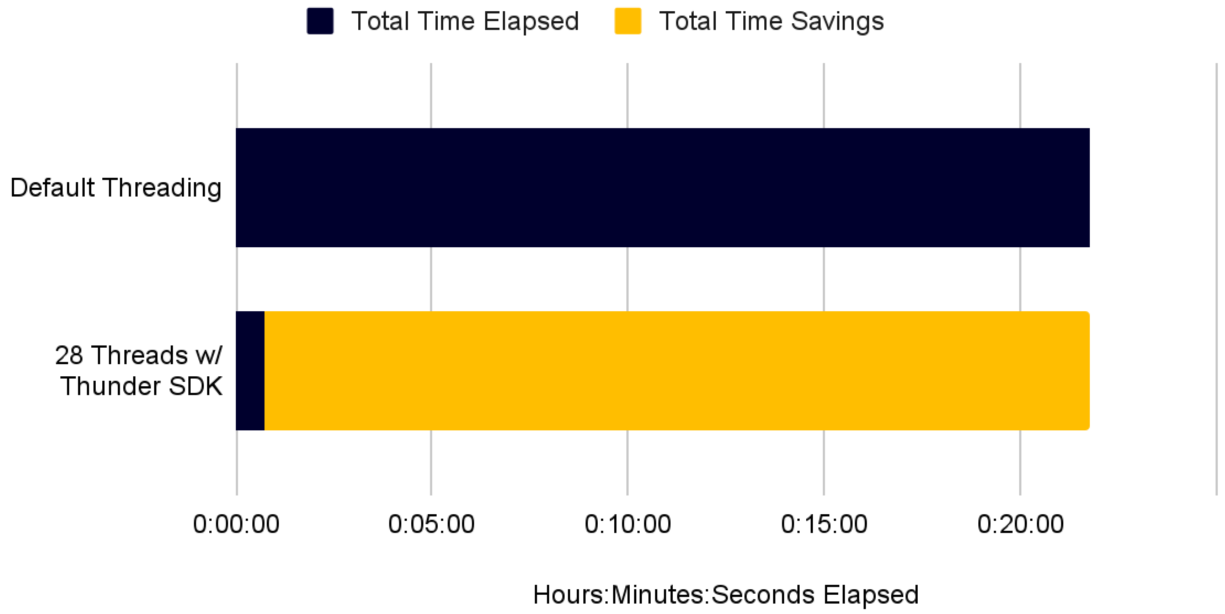
The total power consumption was:

	CPU	System-Wide
Median Total Use	9,893.64 watts	15,647.42 watts
Total Power Savings	**100,123.46 watts** **91%**	**212,167.74 watts** **93.1%**



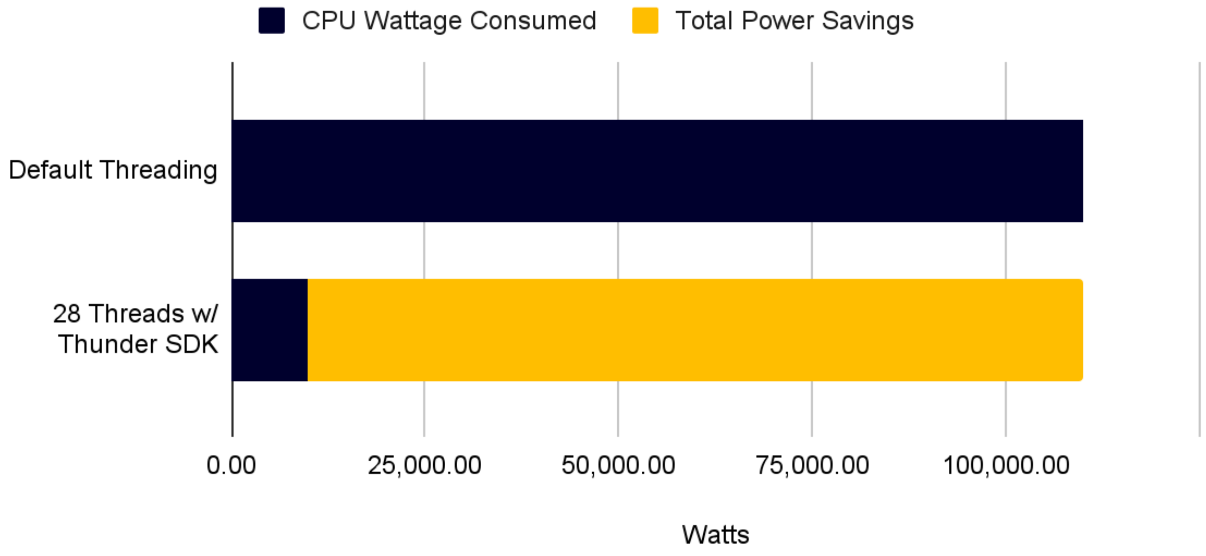
Top-Tier - Total Time Elapsed During Testing

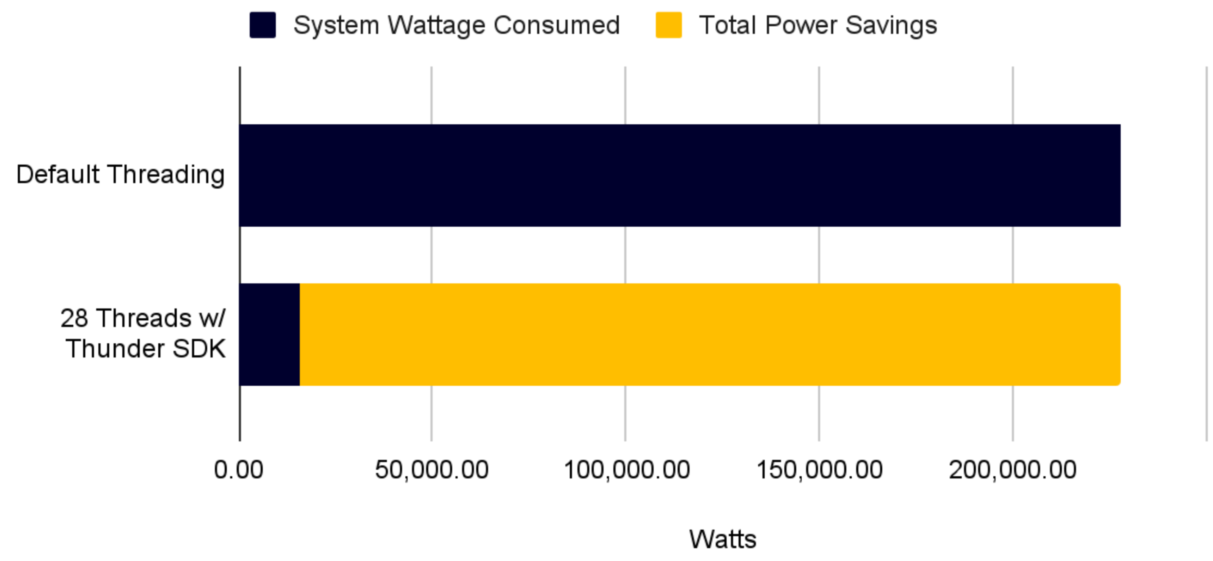
AMD Ryzen Threadripper PRO 7955WX 16-Core



Top-Tier - Median Total Watts Consumed by CPU During Testing

AMD Ryzen Threadripper PRO 7955WX 16-Core





Top-Tier - Median Total Watts Consumed by Whole System During Testing

AMD Ryzen Threadripper PRO 7955WX 16-Core

Mid-Tier Results

The mid-tier system used for testing was an:

- ➔ AMD Ryzen Threadripper 2990WX 32-Core[5] Processor @ 3.0GHz, Windows 10 system with 128GB of DDR4 RAM
- ➔ It is most equivalent to a c5a.16xlarge AWS instance.

Default Threading Results

Completing testing using the default threading behavior of magick.NET took **39 minutes and 39.9 seconds**. On the equivalent c5a.16xlarge AWS instance, the compute portion would **cost $3.57**[6].

[5] 32 physical cores with 2 logical counts per core - 64 total cores, test ran using 56 cores

[6] On an instance in US West (Oregon) running Windows the hourly rate at time of publication is $5.4048/hour.

The total power consumption was:

	CPU	System-Wide
Median Total Use	164,953.16 watts	357,227.94 watts

56 Thread Results

Completing testing using the Thunder SDK to distribute work across 56 threads took **3 minutes and 25.6 seconds**. The compute portion would **cost $0.31** to run in AWS.

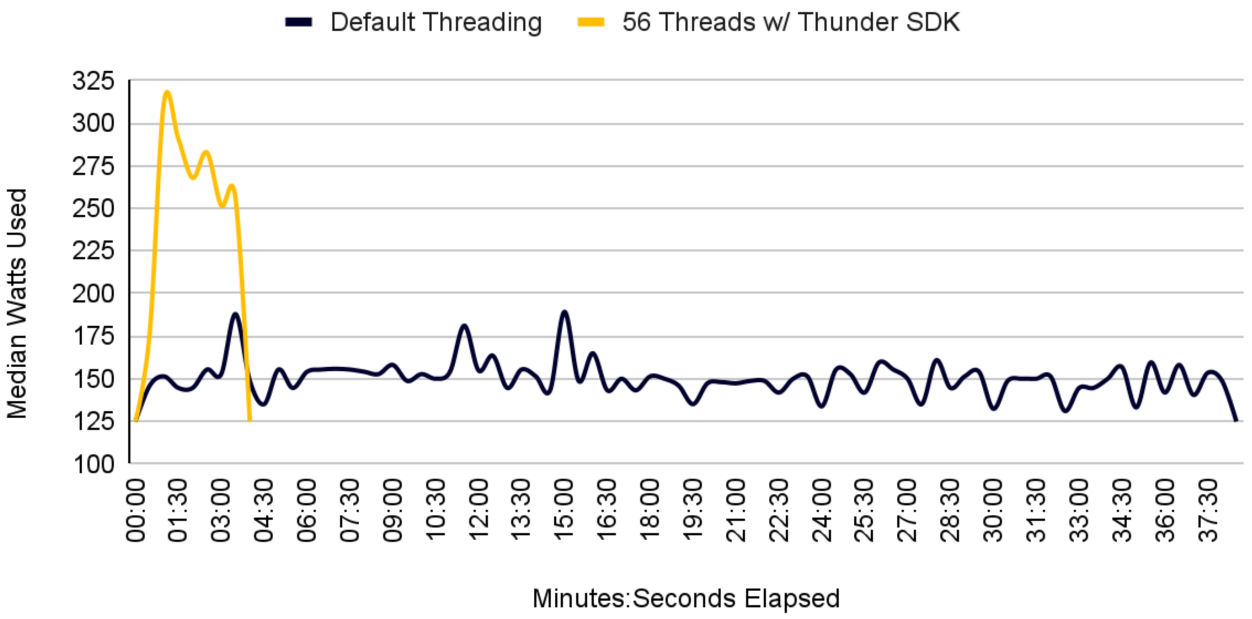
The Thunder SDK shortened the test by **36 minutes and 14.3 seconds, a 91.4% savings, and a compute cost savings in AWS of $3.26.**

The total power consumption was:

	CPU	System-Wide
Median Total Use	32,993.21 watts	55,053.79 watts
Total Power Savings	**131,959.95 watts 80%**	**302,174.15 watts 84.6%**

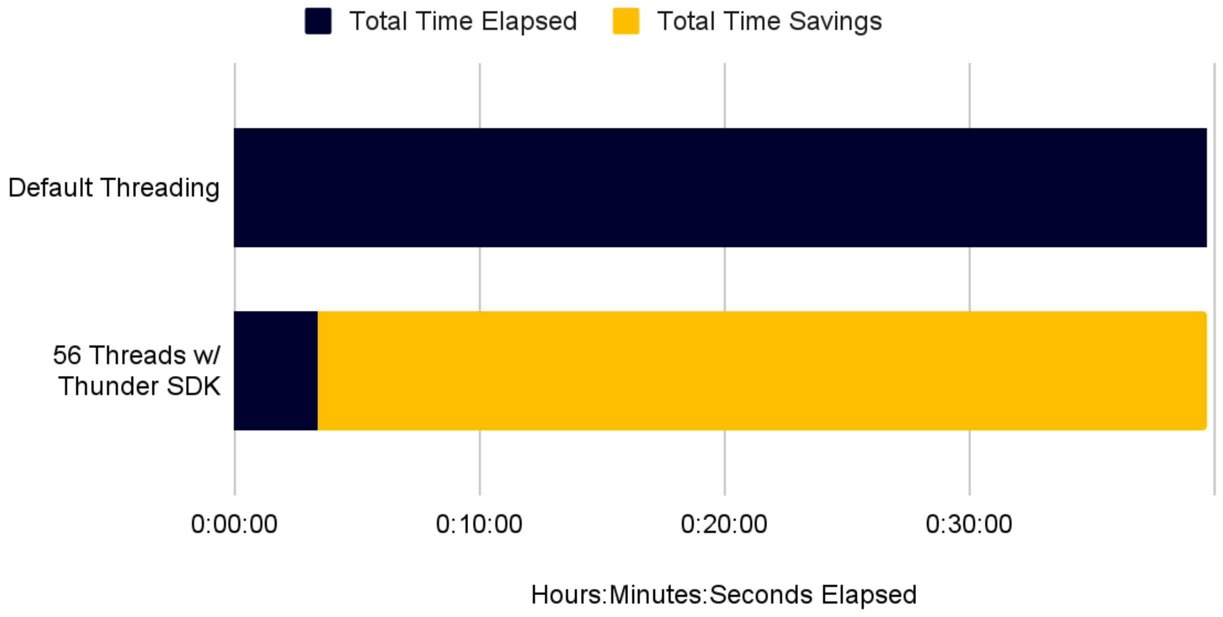
Mid-Tier - Median System-Wide Power Usage

AMD Ryzen Threadripper 2990WX 32-Core



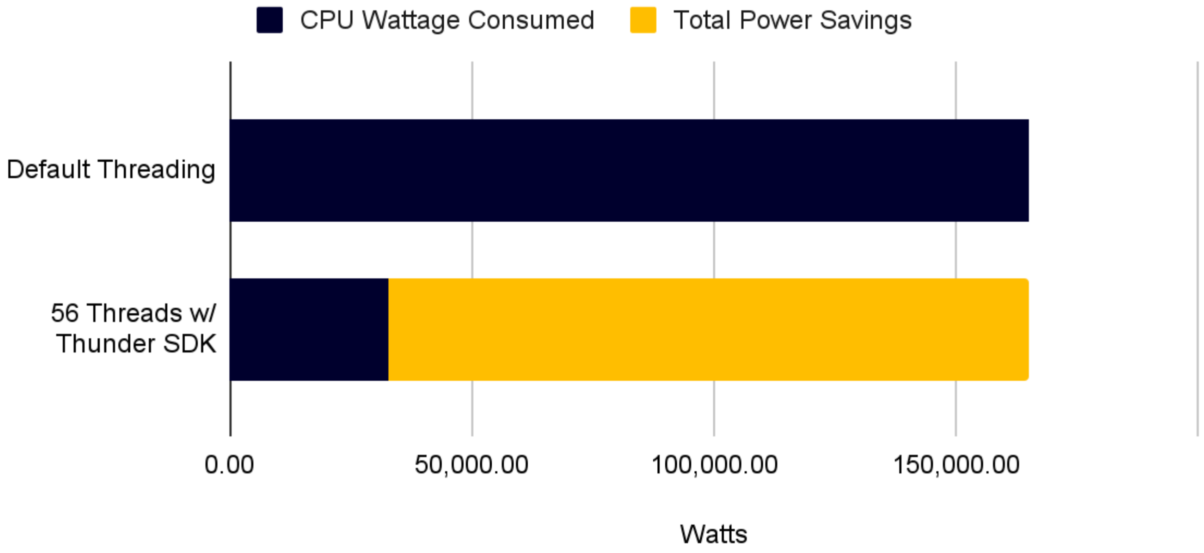
Mid-Tier - Total Time Elapsed During Testing

AMD Ryzen Threadripper 2990WX 32-Core



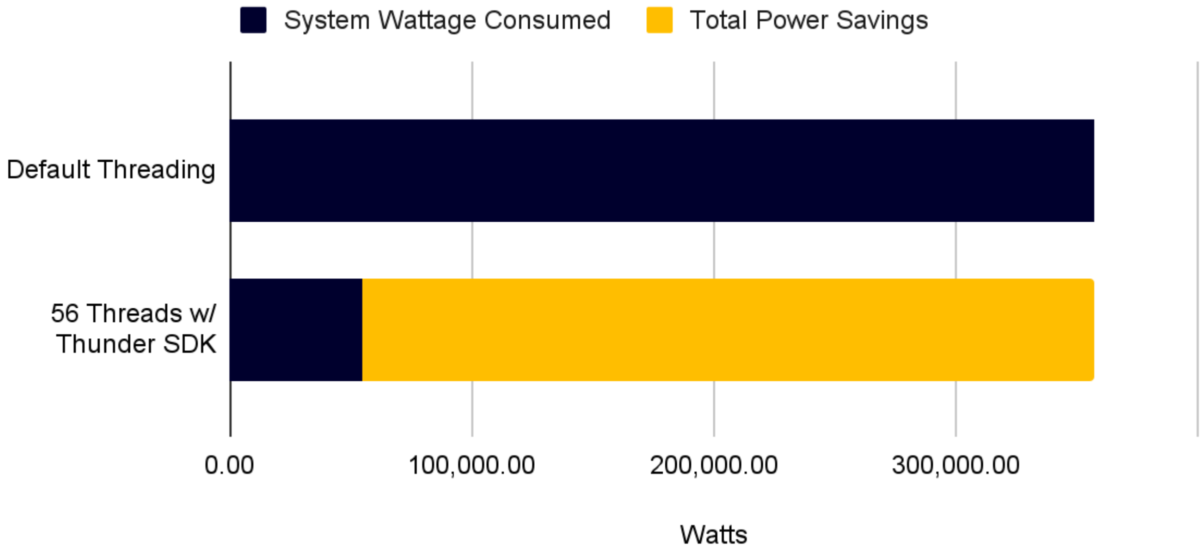
Mid-Tier - Median Total Watts Consumed by CPU During Testing

AMD Ryzen Threadripper 2990WX 32-Core



Mid-Tier - Median Total Watts Consumed by Whole System During Testing

AMD Ryzen Threadripper 2990WX 32-Core



Low-Tier Results

The low-tier system used for testing was an:

- ➔ AMD Ryzen 9 6900HS 8-Core[7] Processor @ 3.3GHz, Windows 11 laptop with 40GB of DDR4 RAM
- ➔ It is most equivalent to a m6a.4xlarge AWS instance.

Default Threading Results

Completing testing using the default threading behavior of magick.NET took **67 minutes and 52.6 seconds**. On the equivalent m6a.4xlarge AWS instance, the compute portion would **cost $1.61**[8].

The total power consumption was:

	CPU	System-Wide
Median Total Use	40,624.22 watts	180,131.27 watts

14 Thread Results

Completing testing using the Thunder SDK to distribute work across 14 threads took **6 minutes and 12.2 seconds**. The compute portion would **cost $0.15** to run in AWS.

The Thunder SDK shortened the test by **61 minutes and 40.4 seconds, a 90.9% savings, and a compute cost savings in AWS of $1.47**.

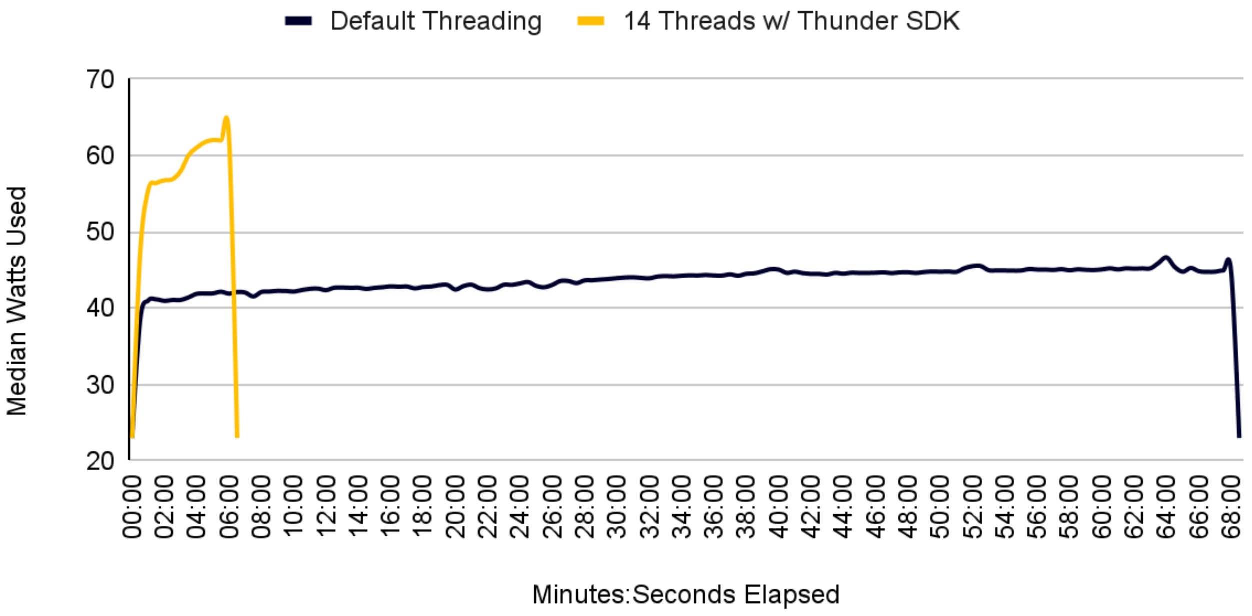
The total power consumption was:

	CPU	System-Wide
Median Total Use	8,526.48 watts	21,945.18 watts
Total Power Savings	**32,097.74 watts 79%**	**158,186.09 watts 87.8%**

[7] 8 physical cores with 2 logical counts per core - 16 total cores, test ran using 14 cores
[8] On an instance in US West (Oregon) running Windows the hourly rate at time of publication is $1.4272/hour.

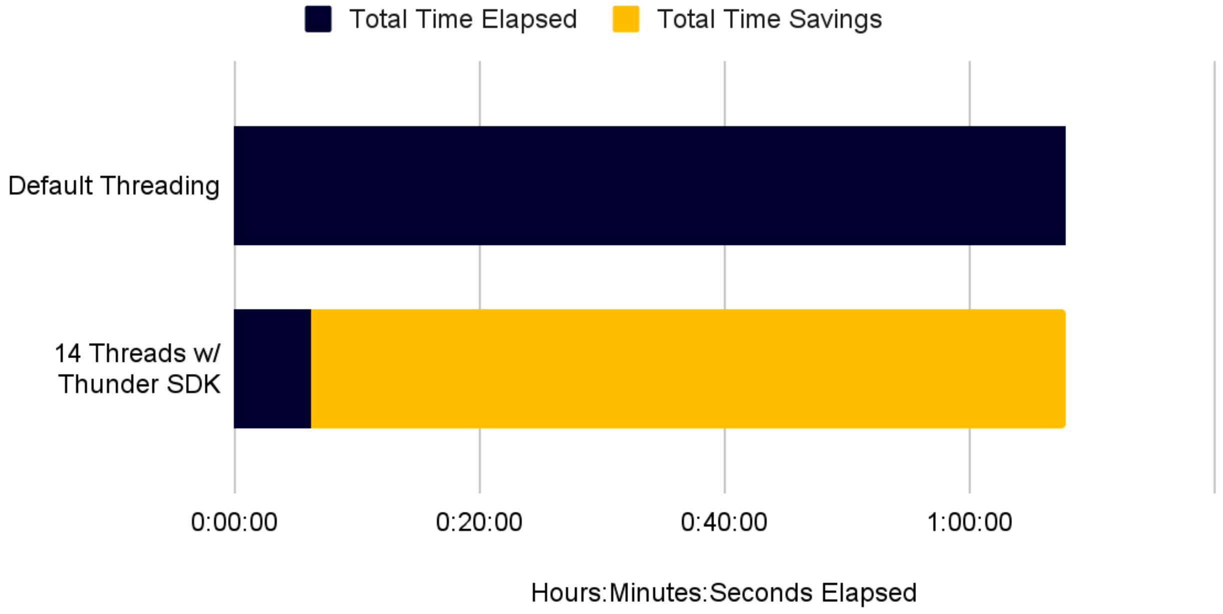
Low-Tier - Median System-Wide Power Usage

AMD Ryzen 9 6900HS 8-Core



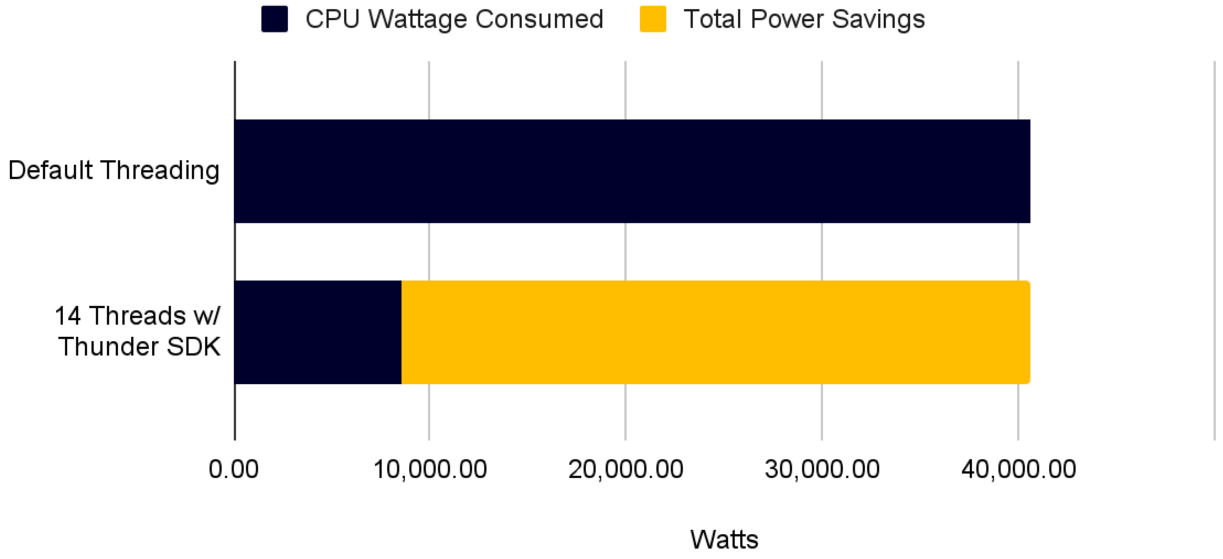
Low-Tier - Total Time Elapsed During Testing

AMD Ryzen 9 6900HS 8-Core



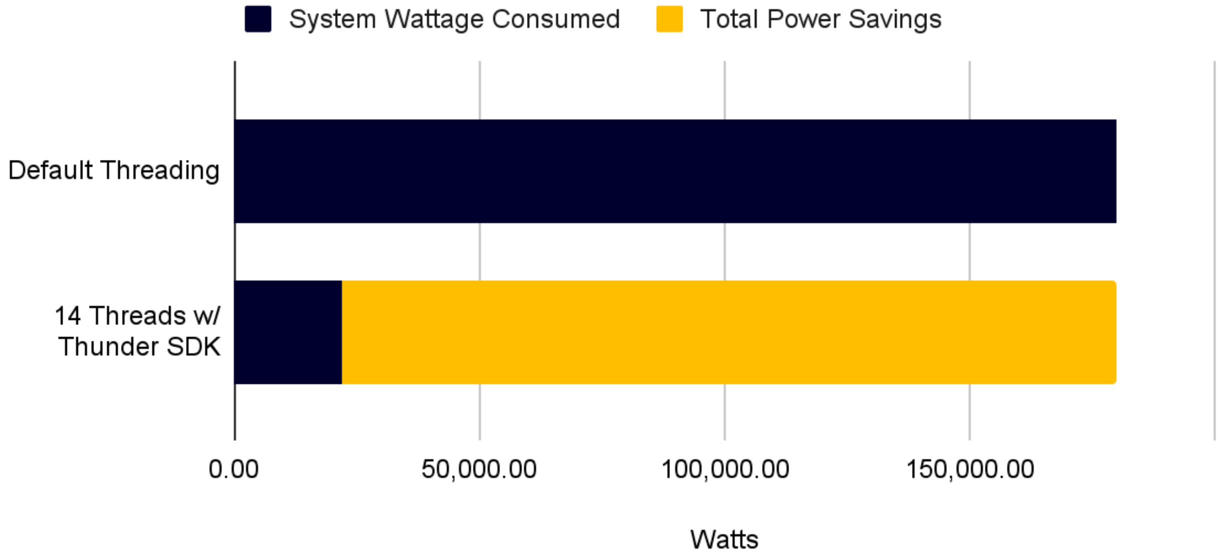
Low-Tier - Median Total Watts Consumed by CPU During Testing

AMD Ryzen 9 6900HS 8-Core



Low-Tier - Median Total Watts Consumed by Whole System During Testing

AMD Ryzen 9 6900HS 8-Core



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