Optimising Financial Services HPC Workloads on AMD EPYC 4th and 5th Generation Processors

Hamza MIAN
James CUFF
Callum WARD

October 2025

HMX LABS

AMD
EPYC

Abstract

This paper provides an analysis of HPC workloads in financial services on AMD 4th generation (previously code-named Genoa) and 5th generation (previously code-named Turin) CPUs using the COREx benchmark. It covers:

- A performance comparison of enabling or disabling SMT
- Power utilisation comparisons across generations and SMT states
- COREx results across different generations of CPU and bare metal vs cloud
- Performance optimisation based on process to logical CPU ratios (slot counts)
- Optimisation for interactive (real time) versus end of day risk

Zen 4 to Zen 5 shows a per core increase in performance of around 22 to 25% for financial analytics workloads. Whilst some of this may be attributable to an increase in the all-core boosted clock speed between the CPUs tested, much of it will be due to other generational improvements.

Further, a 63% increase in performance in observed between the two parts tested. Only part of this (33%) may be attributed to the increased core count between parts (that are not a direct replacement for one another between generations). The remaining is due to improvements in the L3 cache, memory bandwidth and other process improvements. These factors become increasingly important as core density increases and are well-illustrated by the results of this work.

The results also illustrate outcomes that may be counterintuitive to some with specific cloud virtual machines exhibiting *higher* COREx scores than comparable bare metal systems using the same Zen 4 or Zen 5 cores. This was observed across both Azure (HBv4 and Fasv6) and Google (C3D, C4D and H4D) virtual machines scoring higher on COREx than the bare metal systems we tested.

Finally, this work provides insight into performance and power optimisation of financial services risk workloads on modern CPU architectures. Some results shown here, while consistent with previous benchmarking exercises using COREx, may be counterintuitive to many. This includes enabling SMT for the best throughput performance but conversely deliberate underutilisation of the CPU (and even disabling SMT) for interactive risk use cases.

CPUs Tested

This work focuses on the 4th generation AMD EPYC 9654 (96 cores, 3.7GHz) and 5th generation AMD EPYC 9755 (128 cores, 4.1GHz) processors. Further details may be found in AMD EPYC 9654 and 9755 Specifications.

It is worth noting that an EPYC 9655, a Zen 5 equivalent of the EPYC 9654 does exist, as does a EPYC 9754, a Zen4 equivalent of the Zen 5 9755 part under test. As such, whilst both provided CPUs are top of rack parts, they are not necessarily a direct replacement across generations.

Both CPUs were provided in a two socket systems.

Access to Hardware

Access to the bare metal EPYC 9654 and EPYC 9755 systems was provided by AMD at no charge to HMx Labs.

Access to all cloud resources was paid for by HMx Labs and done via HMx Labs accounts.

Test Methodology & Disclaimers

All tests were performed using the same operating system, Ubuntu 22.04 LTS, with the latest available updates applied. The same binaries for the benchmarks were used across machines and sourced from a version-controlled binary repository.

The entire testing process is automated using tooling developed by HMx Labs which ensures a consistent baseline not only for the operating system, benchmark binaries and its dependencies but also the installation process.

All benchmarking on bare metal systems was repeated a minimum of nine times. All benchmark results from cloud VMs are based on at least three runs but usually between five and nine.

This work was performed using the COREx benchmark which is available for use here at https://github.com/hmxlabs/corex.

All relevant information to reproduce these results is also provided. The performance characteristics of your own workload may differ from that observed with COREx. These benchmarks are “as run”; no implied performance outside of this work is assumed

Explaining The COREx Benchmark

Financial risk calculation is not only computationally expensive but also based on sophisticated proprietary mathematical algorithms; often relying heavily on specific CPU instruction sets such as AVX2 and AVX512. Therefore, comparison of CPU performance between any given financial analytic library may not always provide a good correlation to performance with a different codebase. Unfortunately, due to the proprietary nature of this code, it is not possible to produce a publicly available benchmark or to share performance data derived from it.

Whilst performance comparisons between financial risk analytics code, such as QuantLib are not perfect, it does provide a closer correlation than generic CPU benchmarks such as Passmark[i] or GeekBench[ii].

Although floating point operations per second (FLOPS) may provide a proxy for linear algebra codebases, the use of theoretical FLOPs shows a poor correlation to real world performance. While there are several Linpack[iii]-based benchmarks available, comparing results can be problematic.

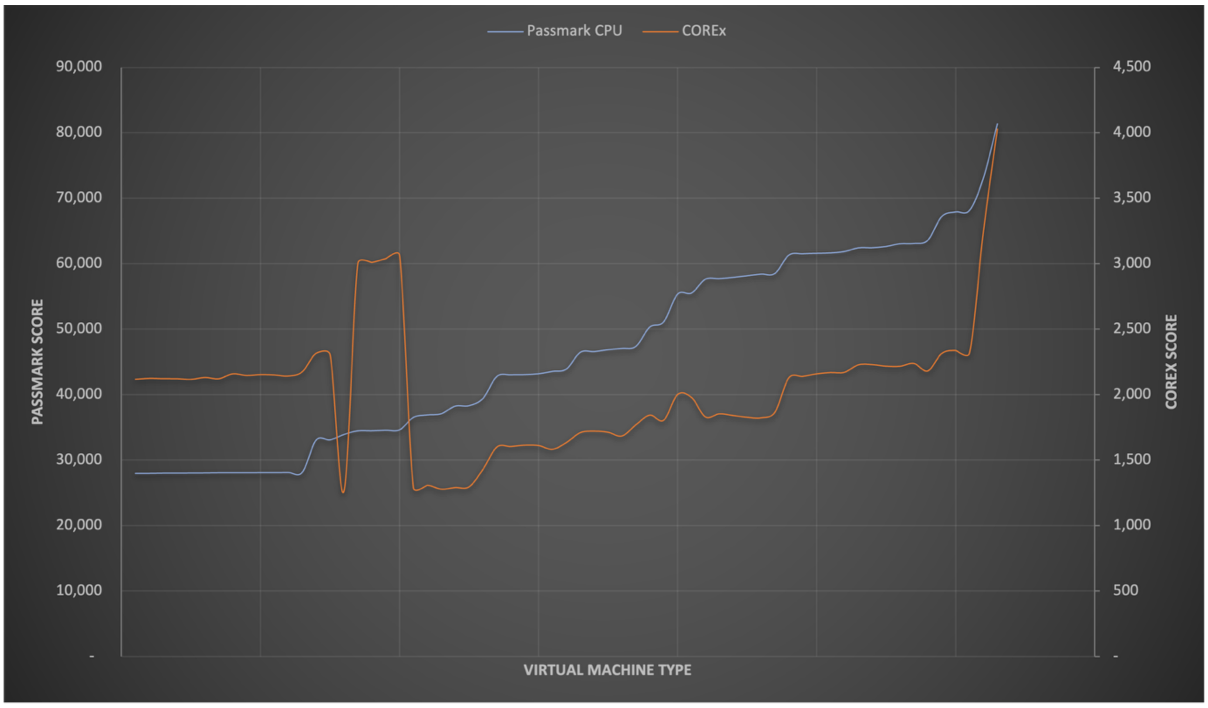


Figure 1 – x axis: Cloud virtual machines in order of ascending Passmark CPU score. Y axis: Passmark CPU and COREx scores. Note the lack of correlation, especially in the first third of X axis, between Passmark CPU and COREx despite COREx being a CPU bound benchmark

COREx uses the open-source financial analytics library QuantLib[iv]. Unlike other QuantLib based benchmarks, it runs more than just the test routines. We do not believe this to be representative of a real financial risk system.

Most benchmarking based on QuantLib takes the form of executing the unit tests that form part of its codebase. Whilst some effort may be made to create a blend of functions that represents real world proportions of workload for different financial products this does not emulate a real risk system. Financial risk systems are responsible for executing analytics libraries and marshalling the input data and results. Risk systems may also be required to co-ordinate multiple different subroutines within an analytics library such as for full revaluation value at risk, or certain risk metrics. COREx attempts to improve upon existing benchmarks by not relying upon the unit tests of a financial analytics library but instead using a complete risk engine.

COREx is based upon Open Source Risk Engine (ORE)[v], a complete open-source financial risk system capable of calculating trade level and portfolio risk metrics. While often used in the context of XVA calculations, in COREx it is used to calculate only market risk across fixed income, equity and commodity sectors and includes a number of trade types including, swaps, swaptions, FX forwards, FX options, equity options, credit default swaps, cap options, bonds, bond forwards, commodity forwards and commodity options.

Lastly, because a financial risk system is also responsible for marshalling highly sensitive data to and from the analytics library, it must comply with stringent security requirements to encrypt all input data and results. COREx uses OpenSSL routines that take advantage of a CPUs AES-NI instruction set if available. The presence (or lack of) AES-NI has a significant impact on the overarching performance of any modern risk system.

Benchmark Aims & Philosophy

"When a measure becomes a target, it ceases to be a good measure" – Goodhart's Law

Unsurprisingly, Goodhart's law applies to benchmarks too.

CPU benchmarks are often optimised to showcase the best possible performance from the hardware and may often be optimised for a particular manufacturer. While financial risk analytics may have been equally optimised, in general, this is no longer true. It is not the case across a range of different CPU manufacturers and models. Many financial analytics libraries are only optimised for Intel hardware and often rely on Intel specific math libraries such as MKL[vi].

COREx has deliberately been designed to be reflective of real risk systems. It has not been optimised for specific hardware and performs operations that closely mirror those of a real risk system running on commodity computing nodes. HMx Labs has deliberately decided to not optimise the benchmark for a particular CPU or ISA to ensure any results remain representative of real workloads **without** additional specific optimisations for a given architecture.

Controlling Execution Time & CPU Utilisation

Upon execution, COREx will read its encrypted input data, decrypt it, pass this to ORE, calculate the market risk factors, take the results and encrypt them. The time taken to do this is measured and a score calculated.

The benchmark code will (by default) run one process instance per logical CPU, i.e. one per thread if SMT is enabled, or one per core if SMT is disabled. A score is produced per process, and an overall COREx score for the system under test. The score is calculated as a function of the total number of simulations performed by the analytics library.

It is possible to manually specify the number of processes that are run and the number of simulations that each process will perform. Increasing the simulation count parameter will increase the time taken. The impact on the score is a function of the amount of time spent moving data in and out versus computing the financial analysis. A higher simulation count will generally result in a higher COREx score; however, this will always asymptotically approach an upper limit (Figure 2)

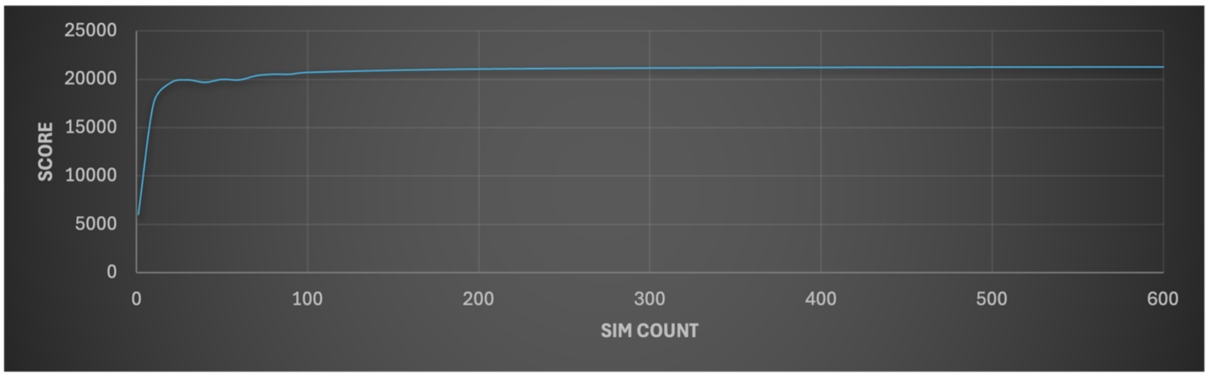


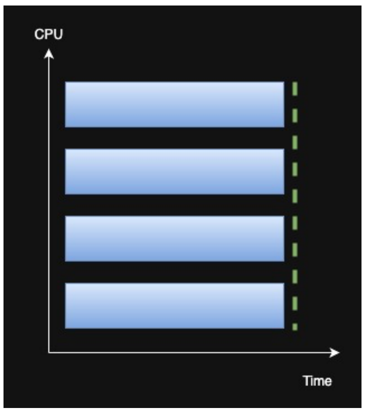
Figure 2 – X axis: Sim Count. Y axis: COREx score.

In our tests, COREx was run with the sim count set to 100, the same value used for benchmarking all existing cloud VMs to date, and was selected to ensure COREx remains CPU-bound, while providing a good balance of execution time and cost.

Each COREx process is capable of fully utilising a single logical CPU, accordingly all prior COREx benchmarks have been performed with the process count set to equal the logical CPU count. However, as part of this work, an analysis of the impact of reducing the process count (with SMT enabled) was also carried out and is detailed below.

How COREx Scores are Calculated

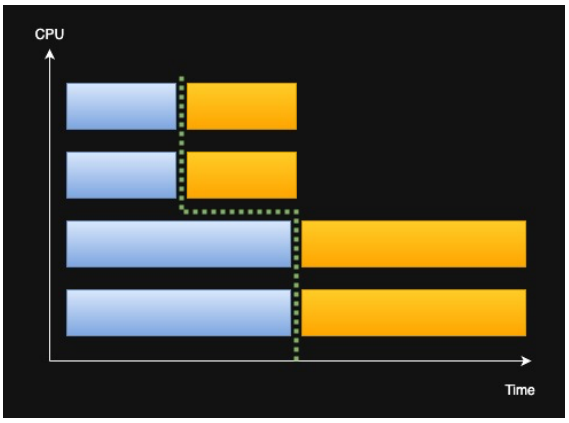
Most benchmarks are calculated using “wall time” for completion of the benchmark, i.e. the clock starts, and the completion time is the time at which all cores/threads complete.



While this is appropriate for interactive workloads or end user benchmarks, it does not provide an accurate reflection of performance when:

1) A CPU contains both performance and efficiency cores leading to a disparity in the wall clock completion time across cores.
2) The CPU is used in an environment where it is possible to schedule work onto each core/thread of the CPU independently such as in HPC and HTC workloads.

The first case is not relevant to the CPUs tested in this work. However, financial services risk calculations are performed using HPC and HTC schedulers and workload is scheduled onto cores/threads as soon as they are idle regardless of the status of work on other threads on the same CPU.



The diagram above illustrates two workloads, blue and yellow with the green line indicating the time measurement for calculating the benchmark score.

COREx takes the completion time for each process to calculate the overall score. This reflects the operational environment for financial risk calculations, i.e. that additional work will utilise free capacity, as illustrated by the top two tasks in the diagram above.

Metrics Analysis for COREx – SMT Enabled & Disabled

CPU Utilisation

Each COREx process is single threaded, and CPU-bound. The benchmark, in its default configuration, will execute one process per logical CPU on the system with the expectation that each process will be CPU-bound. This hypothesis is confirmed both when SMT is disabled and enabled.

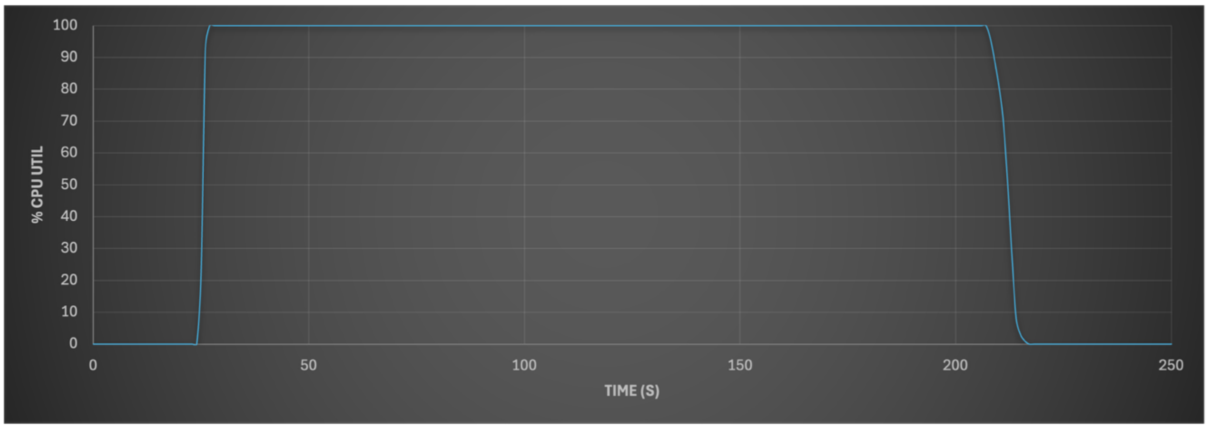


Figure 3 –SMT Disabled. X axis: Time (seconds). Y axis: Percentage CPU Utilisation

For clarity, on systems with SMT disabled this equates to running a single process per CPU core and for systems with SMT enabled this equates to running a single process per thread. For the AMD systems tested in this paper this means two processes per CPU core.

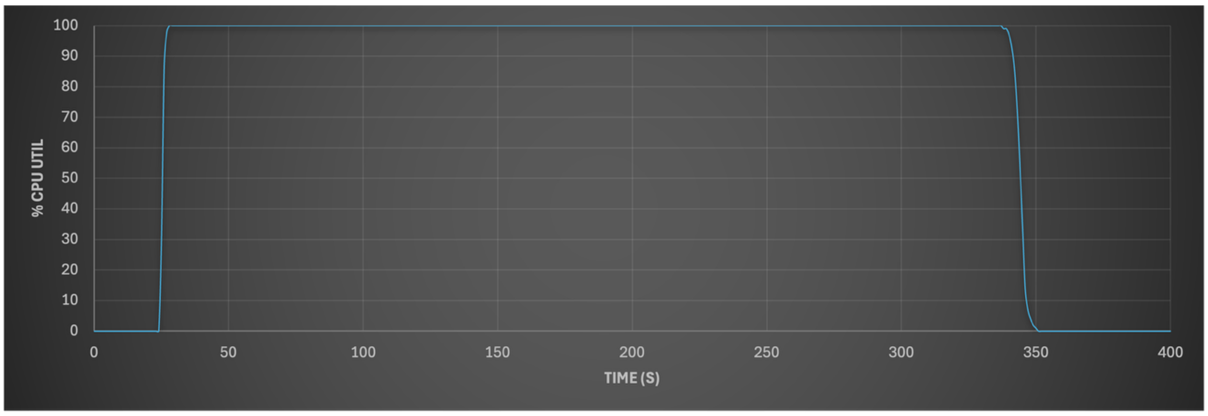


Figure 4 –SMT Enabled. X axis: Time (seconds). Y axis: Percentage CPU Utilisation

Context Switches & Interrupts

Enabling SMT, especially for HPC and HTC workloads, does not necessarily result in an increase in performance. In addition to the benchmark results there are other factors that result in changes to performance including the number of context switches and interrupts.

Enabling SMT does not directly increase the number of context switches, and this can be managed using logical CPU pinning. The availability of additional CPUs may result in increased context switching, especially if the workload performs I/O operations, for example, disk reads and writes, or network communication.

When running COREx with SMT it does not exhibit an increase in context switching, even without using CPU pinning. This indicates CPU-bound processes that will run to completion for the duration of the benchmark.

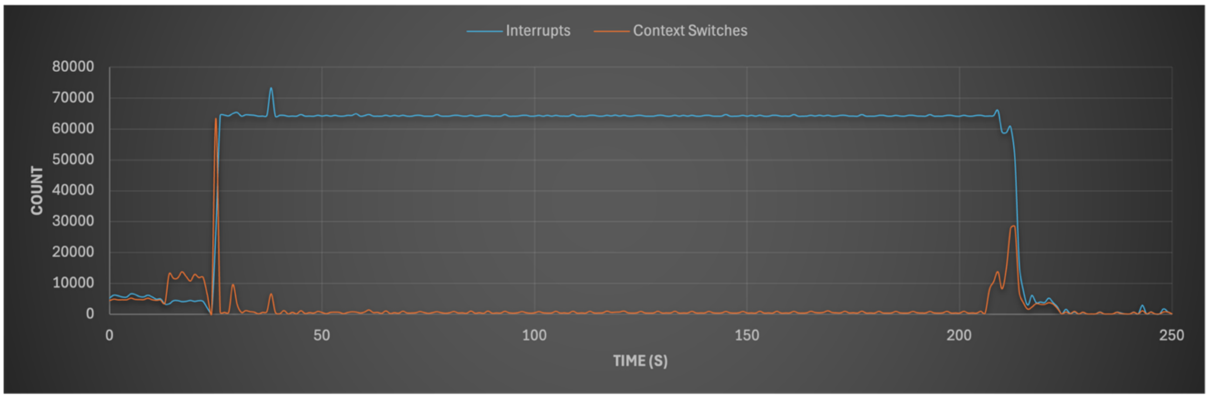


Figure 5 –SMT Disabled. X axis: Time. Y axis: CPU wait time

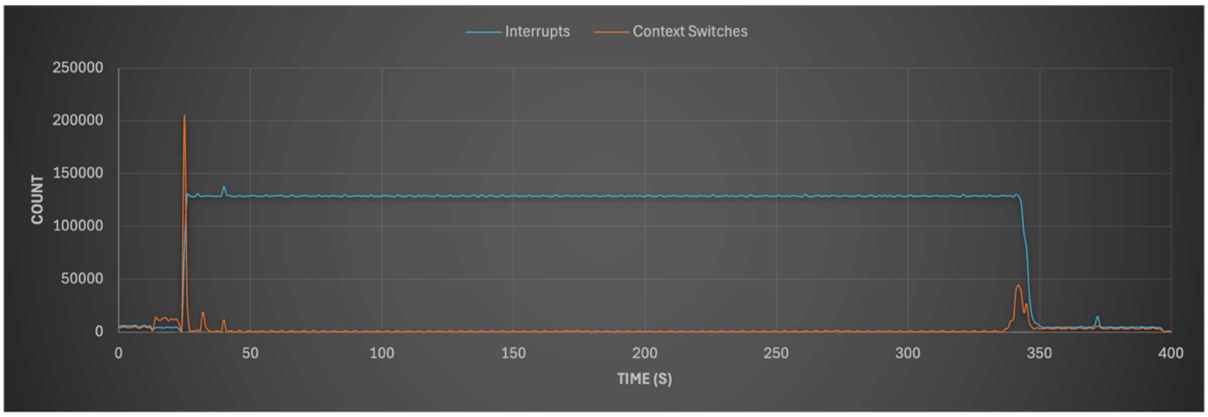


Figure 6 –SMT Enabled. X axis: Time. Y axis: CPU wait time

CPU System Usage and Wait Time

COREx exhibits a small amount of system level CPU utilisation (~10%) for a short duration during test startup. This can be attributed to the startup of a large number of processes within the COREx benchmark and their initial I/O operations. CPU Wait time remains at zero.

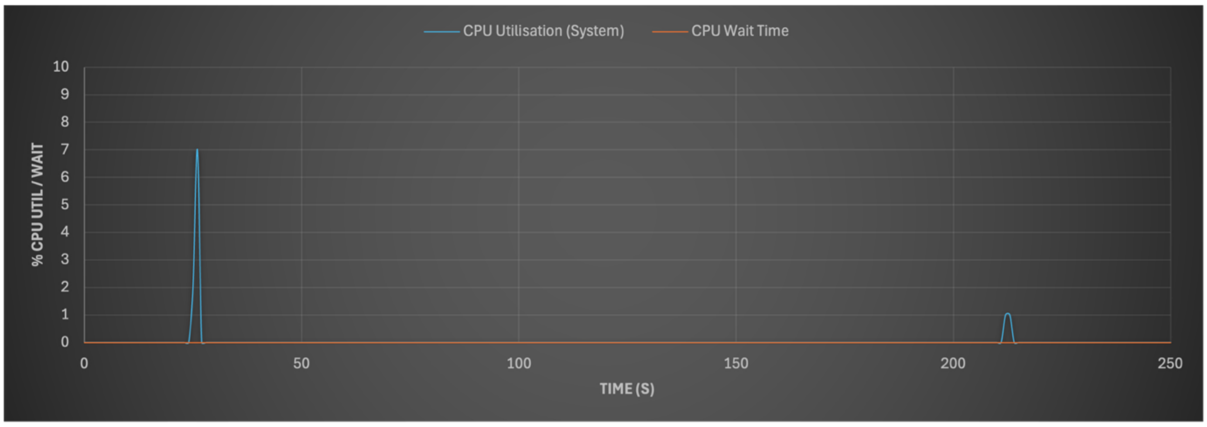


Figure 7 SMT Disabled – X axis: Time. Y axis: CPU system utilisation and wait time

These results would suggest COREx being a genuinely CPU-bound benchmark. However, the performance increase shown by enabling SMT may indicate otherwise. This is discussed further in the next section.

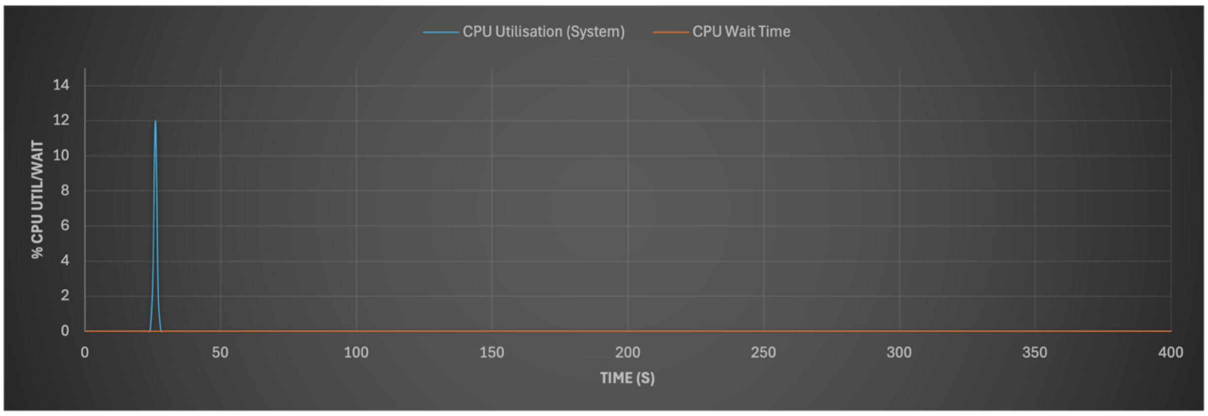


Figure 8 SMT Enabled – X axis: Time. Y axis: CPU system utilisation and wait time

COREx System Performance Scores

The COREx benchmark was executed on both systems in accordance with the methodology outlined earlier and achieved using HMx Labs' Cloudbench tooling.

In line with previous analysis,[vii] both 4th generation (EPYC 9654) and 5th generation (EPYC 9755) processors showed an improvement in system level performance with SMT enabled, 15% and 12%, respectively.

> *Key Point 1*
>
> *Enabling SMT, even when workloads may appear to be CPU-bound may result in improved throughput. There is no substitute for testing*

System Performance

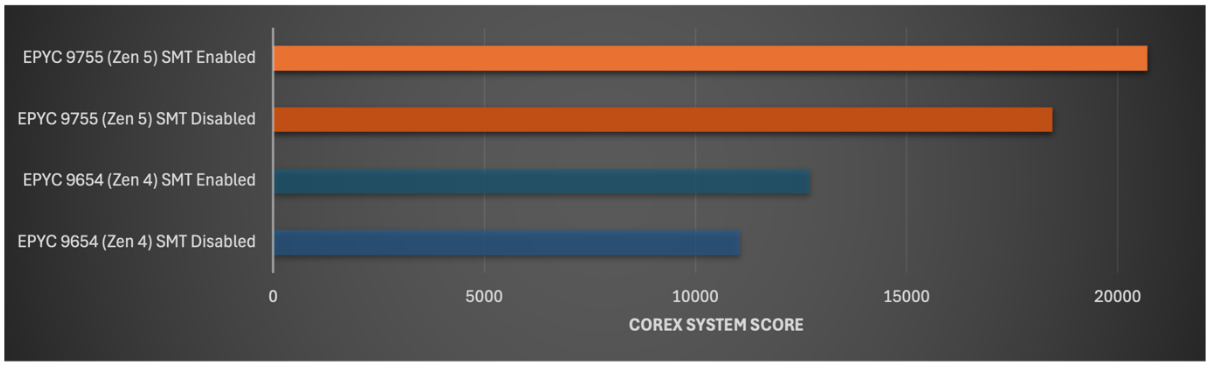


Figure 9: System COREx scores EPYC 9654 and EPYC 9755. Higher scores are better

There is a significant uptick in performance between the EPYC 9654 and EPYC 9755 at a system level with a 67% and 63% increase in the COREx score with SMT disabled and enabled, respectively. It should be noted, however, that the EPYC 9655 would be the direct replacement for a EPYC 9654. The EPYC 9755 has a higher core count (96 vs 128) which in itself would account for a 33% increase; the additional ~30% increase can be explained due to generational improvements and differences in cache sizes which is further confirmed below.

Core Performance

While a large proportion of the increase in system level performance can be attributed to the higher core count (128 vs 92) between the two processors, there is also a non-trivial increase is per core performance, 25% and 22% with SMT disabled and enabled, respectively. This may be attributed directly to improvements between the Zen 4 and Zen 5 CPU architectures including higher all core boost speeds.

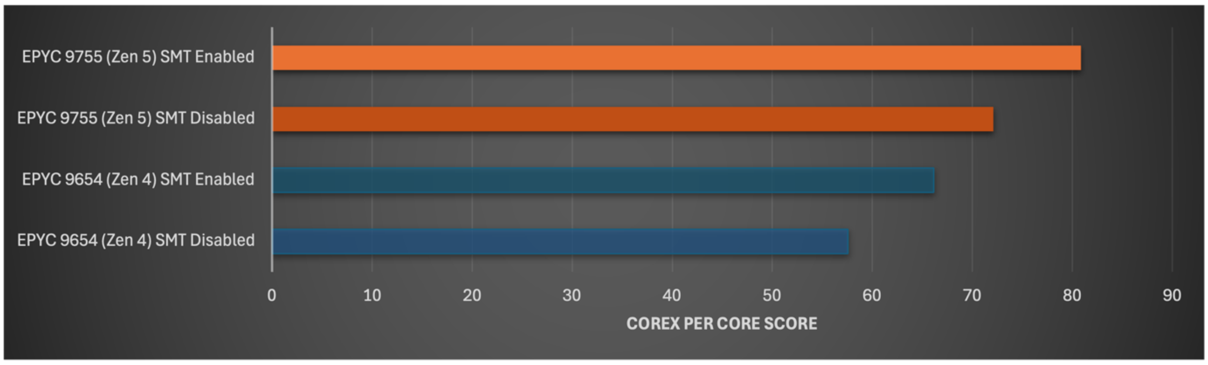


Figure 10: Per CPU core COREx Scores. EPYC 9654 and EPYC 9755. Higher scores are better

Key Point 2

Generational improvements between Zen 4 and Zen 5 show a ~25% performance uplift

Consistency

Both 4th and 5th generation processors were incredibly consistent across 15 executions of the COREx benchmark; with a coefficient of variation that remains below 0.5%. While a notable increase was observed for the EPYC 9654 with SMT enabled this remains very low in absolute terms.

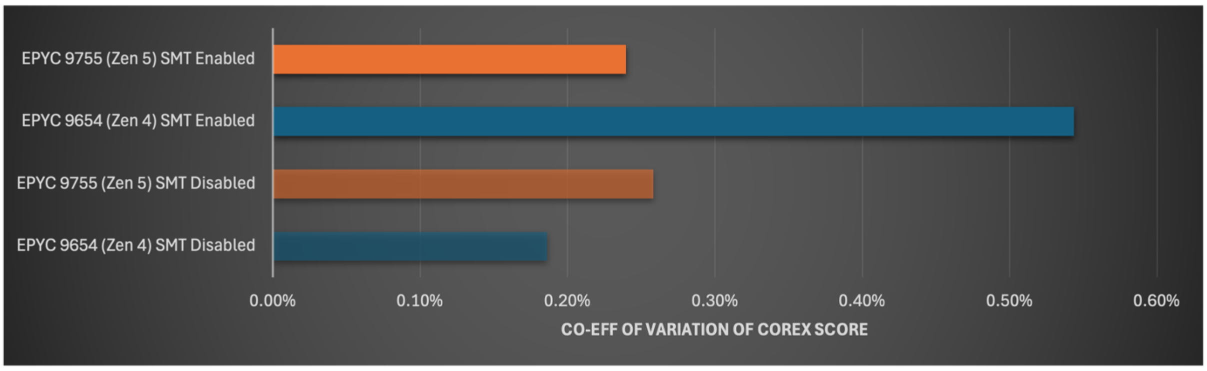


Figure 11: Coefficient of Variation of COREx with SMT enabled and disabled. Lower values are better

Increased SMT Performance: Is COREx CPU-Bound?

It is interesting to note here that whilst the initial metrics shared would point towards a CPU-bound benchmark, the clear increase in performance obtained by enabling SMT indicates it may not be.

It should be noted here that this is consistent with our previous experience of proprietary financial analytics libraries also, i.e. that they will typically benefit from an uptick in performance by enabling SMT.

Whilst COREx does have an IO component to the benchmark, with the relatively high sim count selected and the significant increase in performance provided by SMT this may point to the analytics code being not limited purely by the arithmetic logic units (ALU), floating point units (FLU)s and single instruction multiple data (SIMD) units within the CPU but other factors such as memory bandwidth and CPU cache sizes.

Indeed, other results in this paper point to the size of the L3 cache having a significant impact on the performance, which points to some aspects of the benchmark being memory access-bound.

This is discussed further below in Variation with Instance Count

Key Point 3

COREx, may not be limited only by the performance of the CPU's ALU/FLU/SIMD units and exhibits characteristics that benefit from increased memory bandwidth and CPU cache sizes

Comparison to Cloud VMs

There lingers a perception that access to bare metal hardware is more performant than virtualised machines.

The data in this case do not support this expectation. The following cloud virtual machines were tested:

Cloud Service Provider	VM Instance Type	vCPUs	CPU Model	Generation	SMT Status
AWS	c7a.48xlarge	192	EPYC 9R14	Zen 4	Disabled
AWS	hpc7a.96xlarge	192	EPYC 9R14	Zen 4	Disabled
Azure	HB_176rs_v4	176	EPYC 9V33X	Zen 4X	Disabled
Azure	D160as_v6	160	EPYC 9V74	Zen 4	Enabled
Azure	F64as_v6	64	EPYC 9V74	Zen 4	Disabled
Google	c3d-standard-360	360	EPYC 9B14	Zen 4	Enabled
Google	c4d-highcpu-384	384	EPYC 9B45	Zen 5	Enabled
Google	h4d-standard-192	192	EPYC 9B45	Zen 5	Disabled

Note: Google H4D was all available in public preview and was not generally available. As such, results shown here may not reflect those with the final specification of these VM types. This performance comparison also does not account for pricing and prices may vary substantially between different VMs.

It is important to note that this test is not a like-for-like comparison of the same CPU model in a bare metal system and on cloud. Cloud service providers, due to their scale, operate their services with customised CPUs that are not available for purchase by other parties. This can be confirmed using `lscpu` output. Further, the exact specification of the CPUs, including clock frequencies (base and boosted) and L3 cache sizes are not available to allow an adjusted comparison or draw meaningful comparisons between the results.

However, this does reflect the choice available to HPC users in the evaluation of on-premises or cloud-based systems. As such, and in line with the philosophy behind the COREx benchmark, the tests have been conducted to reflect as closely as possible real-world usage. To ensure parity with the bare metal systems, the largest available size (vCPUs) in each VM family has been tested to mitigate the noisy neighbour effect (see also the section below on Variation with Instance Count).

Key Point 4

It is possible to run HPC workloads in the cloud without a performance penalty compared to bare metal on premises systems

This may not provide academic answers to questions such as the impact of a hypervisor, but it does provide useful data in the selection of infrastructure for real world workload.

Moreover, comparison of the EPYC 9654 and EPYC 9755 systems to cloud virtual machines (logical CPU scores compared) yields some interesting results.

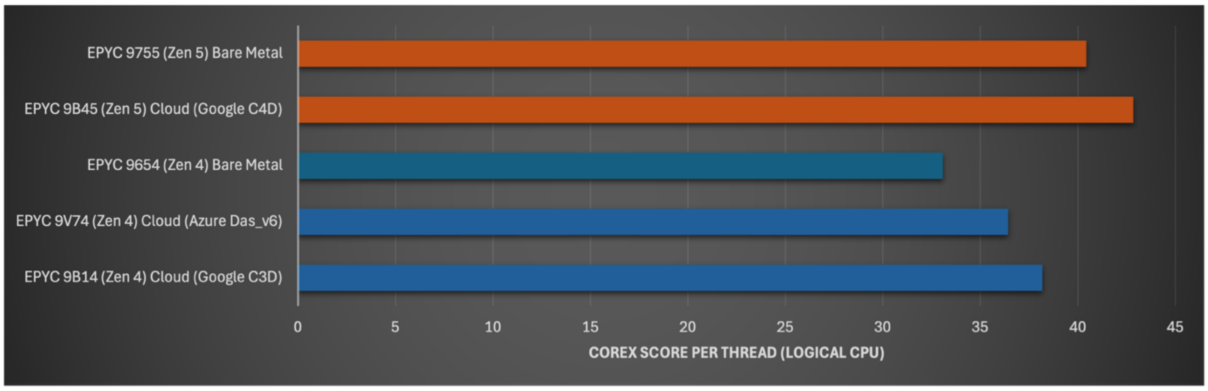


Figure 12: COREx scores per thread with SMT enabled for EPYC 9654, 9755, Google C3D and C4D and Azure Das_v7. Higher scores are better

We see higher per logical CPU scores on both 4th and 5th generation EPYC CPUs in cloud virtual machines than the top of stack processors with both SMT enabled (above) and disabled (see below).

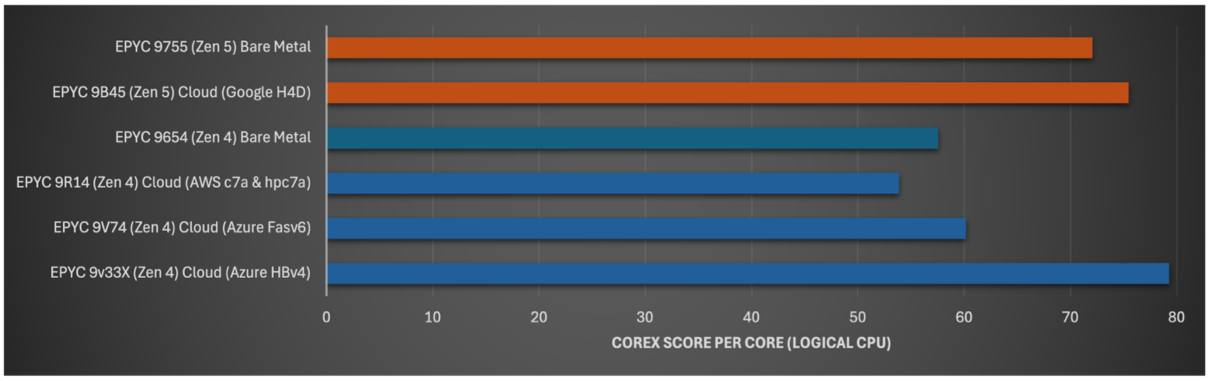


Figure 13: COREx scores per core with SMT disabled for EPYC.9654, 9755, Azure HBv4, Fas_v6, Fas_v7 and AWS hpc7a and c7a, Google H4D. Higher scores are better.

As stated above, the cloud providers do not provide any details on the specifications of the CPUs. Further, no analysis has been conducted thermal throttling or the environmental conditions that these systems are operating in.

We can however speculate a little. The higher performance of the Azure HBv4, which uses the EPYC 9004X processers, may be attributed to its larger 3D V cache as this is based on the Zen4c architecture not Zen 4. The other differences such as the C4D and H4D could possibly be attributed to all core boosted higher clock speed which may be possible due to the lower core counts.

Whilst we did not access to it, testing and comparing results for the EPYC 9655, a 96 core part with a higher all core boost of 4.5GHz but a lower L3 cache size of 384MB would provide some interesting insights.

Scores for the AWS c7a and hpc7a are lower, the difference is small at ~7%. Again, without publicly available details it is difficult to attribute this difference in performance to differences in hardware configuration, changes in the CPU specification or a hypervisor or simply thermal throttling.

Variation with Instance Count

Optimising for PnL and Realtime Risk vs End of Day

This investigation was carried out only with SMT enabled for all COREx runs.

There is a clear correlation between COREx process count and overall score (increasing) until a limit of one process per thread (logical CPU), i.e. 512 processes, is reached. At this point a significant degradation of performance is evident.

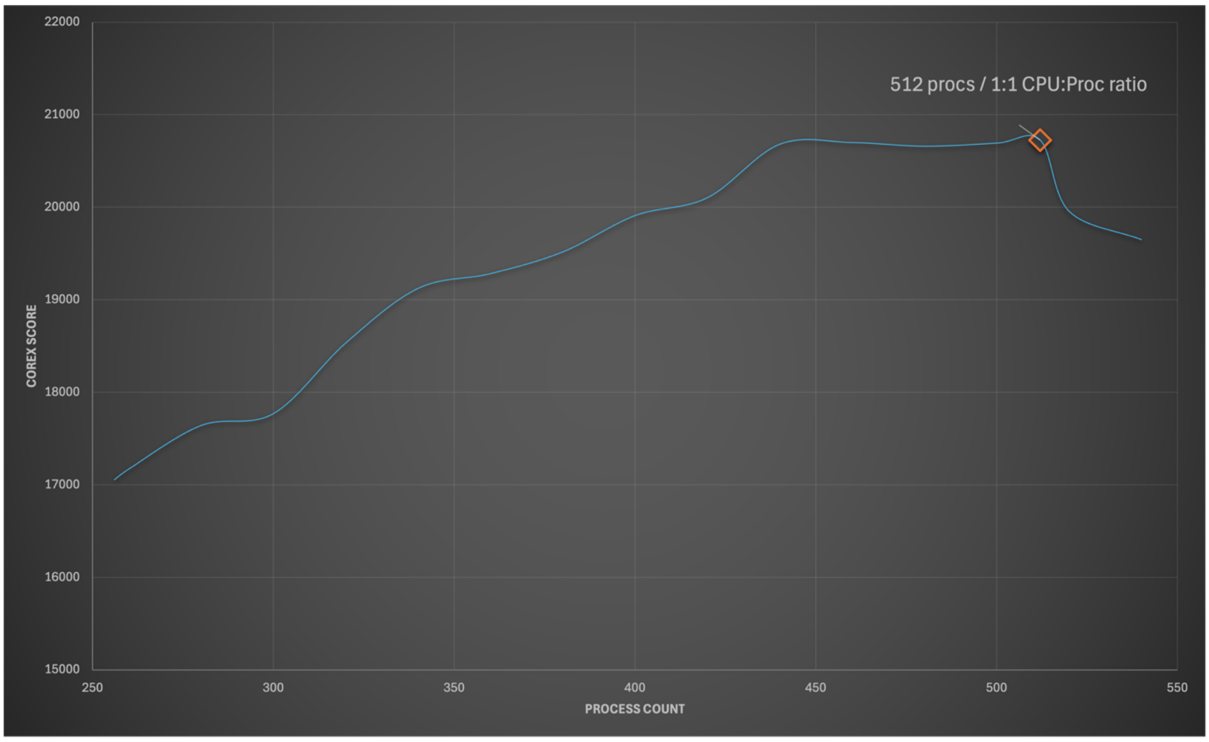


Figure 14: X axis: Number of COREx processes. Y axis: COREx score (higher scores are better). Increasing the process count results in higher scores as expected up to the number of logical CPUs available on the system followed by a sharp drop off.

> *Key Point 5*
>
> *Throughput performance of COREx increases until a ratio of 1 process to 1 thread, beyond which point it degrades substantially*

This is in line with the expected behaviour for most HPC applications, and in fact most CPU-bound processes.

However, whilst financial risk systems are generally optimised for high throughput, i.e. to complete a large volume of risk calculations in the shortest time possible, this is no longer their only use case.

Through a process of risk system consolidation undertaken by virtually all banks, most financial risk systems are now required to perform not only end of day risk calculations but also interactive risk calculations such as intraday risk and pre-trade scenarios. Unlike end of day risk calculations these use cases present a relatively low overall compute demand, several orders of magnitude lower than end of day risk. Conversely, they present a much higher sensitivity to latency.

Intraday risk results are often useful for time periods as low as five minutes and pre-trade scenarios are generally expected to return results sub second (trader expectation), but must present results within 10 seconds (upper limit to retain user attention).

Although risk systems for interactive risk use cases may often have dedicated resource groups, the configuration of this compute capacity is generally the same as that used for end of day risk (to maximise overall compute utilisation).

COREx was designed primarily with an end of day risk use case in mind (the score methodology confirms this).

Optimising for latency rather than throughput requires analysis of metrics other than headline score. Specifically, the wall time and the logical CPU score (both also reported in the COREx results).

While the overall throughput of the system is seen to increase until the process and logical CPU count reaches parity, the wall time also increases and there is a steady decrease in the logical CPU COREx score.

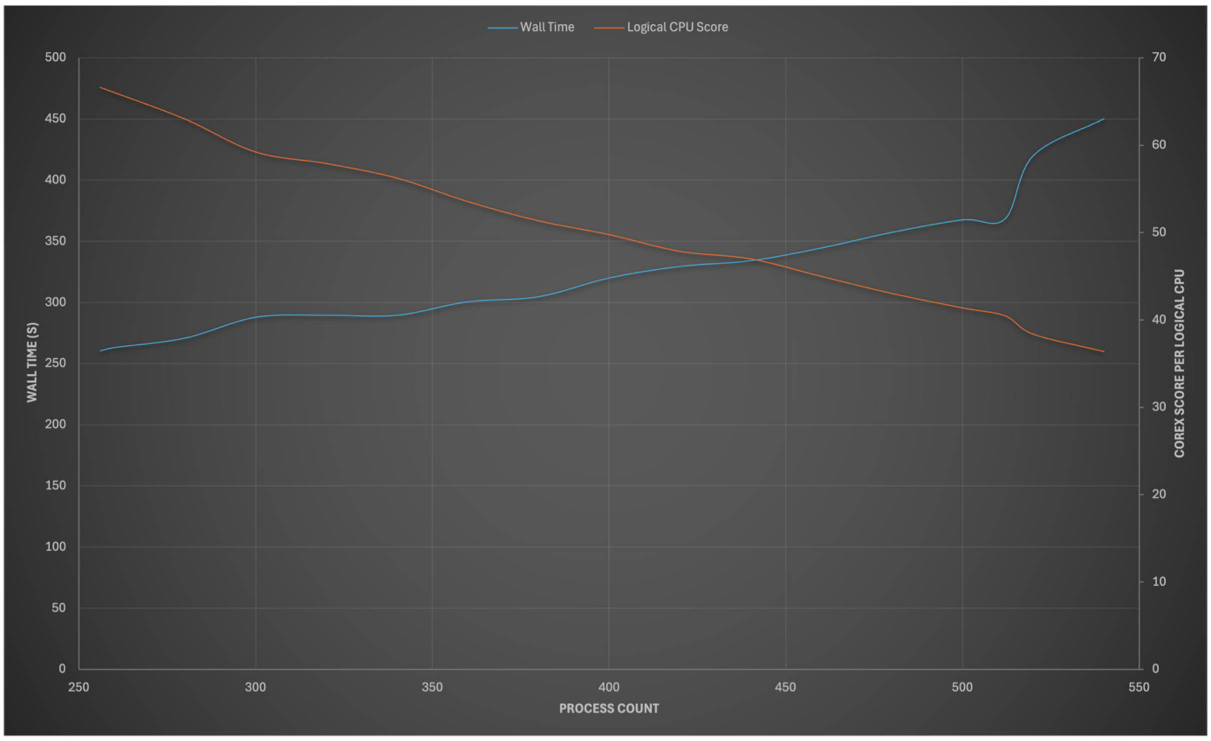


Figure 15: X axis: Number of COREx processes. Y axes: Logical CPU COREx score and Wall Time (seconds).

This phenomenon is neither new, nor unexpected. In previous testing on cloud VMs, it has been found that the logical CPU score is significantly higher for machines in the same VM family but with lower vCPU counts. The decrease in performance may be attributed to several factors including memory bandwidth and CPU cache limitations. AMD refer to this as the "noisy neighbour" issue[viii]; steps have been taken by AMD to minimise this effect, but it is impossible to eliminate. Further details on the "noisy neighbour" effect can be found in the AMD contributed section in the book *An Introduction to Supercomputing and HPC*[ix]

Although this effect is expected, its impact against the optimisation of risk systems for interactive versus end of day risk has not been widely discussed.

Critically, the trade-off involves accepting lower overall CPU utilisation (increased hardware costs) for improved response times. This is generally the antithesis of HPC optimisation techniques.

Key Point 6

Although throughput improves with higher process to thread count ratios, latency also increases. For time sensitive workloads such as interactive or intraday risk it is preferable to underutilise the CPU for improved latency. This may include disabling SMT for these workloads.

An interesting point to note here is at 256 processes (i.e. 1 process per core) the logical CPU score of 67 remains ~7% below the SMT disabled score. When reducing the slot count to 0.5 processes per logical CPU, it would also be beneficial to run with SMT disabled. Although enabling SMT provides a higher throughput for time-sensitive calculations, running with SMT disabled may be preferable. It would be interesting to repeat this exercise with SMT disabled to obtain a more complete picture.

Practical Application

Practically, this could result in allocating resource groups for interactive risk that under allocate CPU slots. However, while this will improve the responsiveness of interactive risk it comes at the cost of not only reduced CPU utilisation within the dedicated resource group but also lower overall CPU utilisation (due to increased partitioning of the compute capacity).

In a purely diurnal system where capacity is idle during periods of demand for interactive risk this is not necessarily problematic. However, many organisations have attempted to consolidate regional use and increase utilisation over a 24-hour period leading to a reduction in the diurnal nature of risk calculation.

Beyond One Process per Logical CPU

As shown in Figure 14, there is a sharp drop in performance once the process to logical CPU ratio exceeds 1 (i.e. more than one process per thread). This is as expected and due to the decrease in performance attributed to additional context switching (the processes were not pinned).

The non metered access to these systems allowed the exploration of performance beyond this drop off point. An area that is not generally investigated.

Further COREx benchmark runs were conducted with a ratio of up to 10:1 (processes to logical CPUs) with some rather interesting effects.

To further understand these results, it was necessary to change the way in which the COREx score is calculated. As explained above, it is the elapsed time of each process, not the overall wall time of the benchmark that drives the score. This assumed that each logical CPU is idle and available to execute other workloads once the process completes. In this scenario, when there are more processes than logical CPUs this is not true; each logical CPU will have additional work to process after completion until such time that the number of processes is less than or equal to the number of logical CPUs.

As such, an alternative score based on the overall wall time (Alt Score) was also calculated.

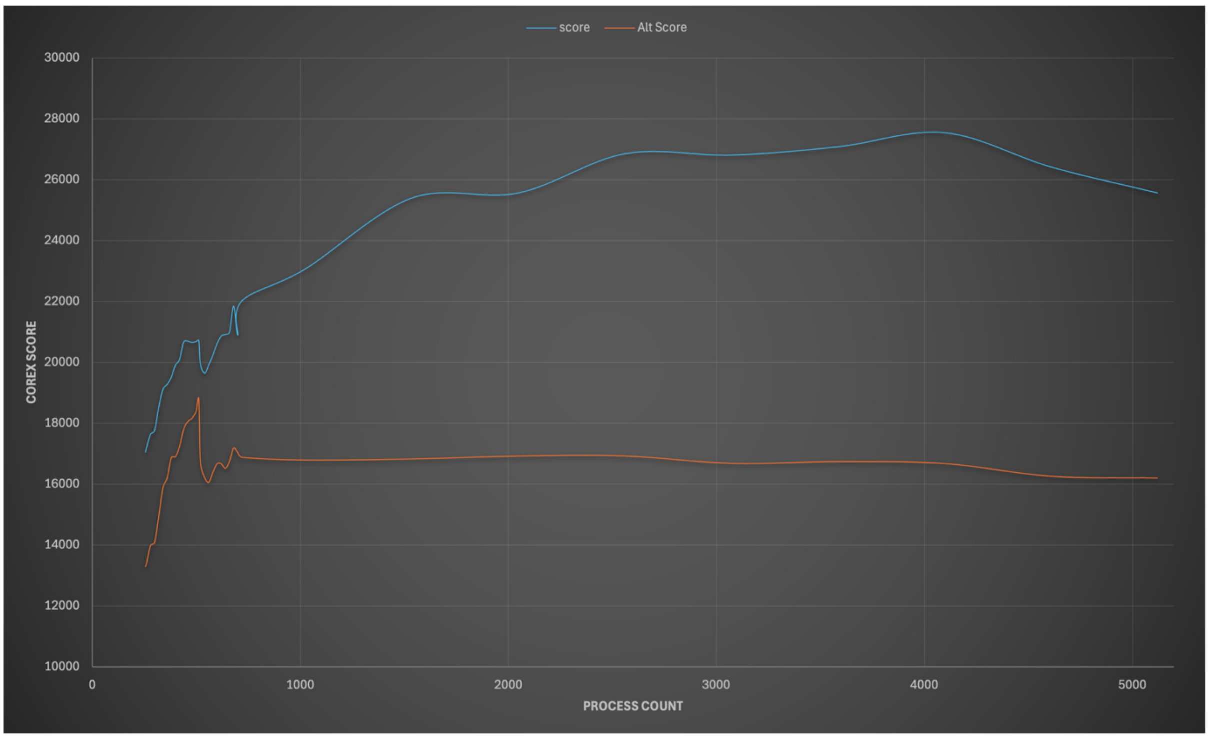


Figure 16: X axis: Number of COREx processes. Y axis: COREx score (higher scores are better). Whilst the sharp drop-off above the number of threads is expected, the slow increase in scores as the process count continues to increase is surprising.

It is interesting to note that at whole number ratios (e.g. 2:1, 3:1) the performance measured by the standard COREx score continued to increase and held constant (but at a lower level) as measured by the alternative score.

Looking at the context switches during a run with 1024 processes (2:1 ratio), it is interesting to note that there is a spike in the context switches at the halfway mark. It would appear as though the operating system was able to effectively queue the processes.

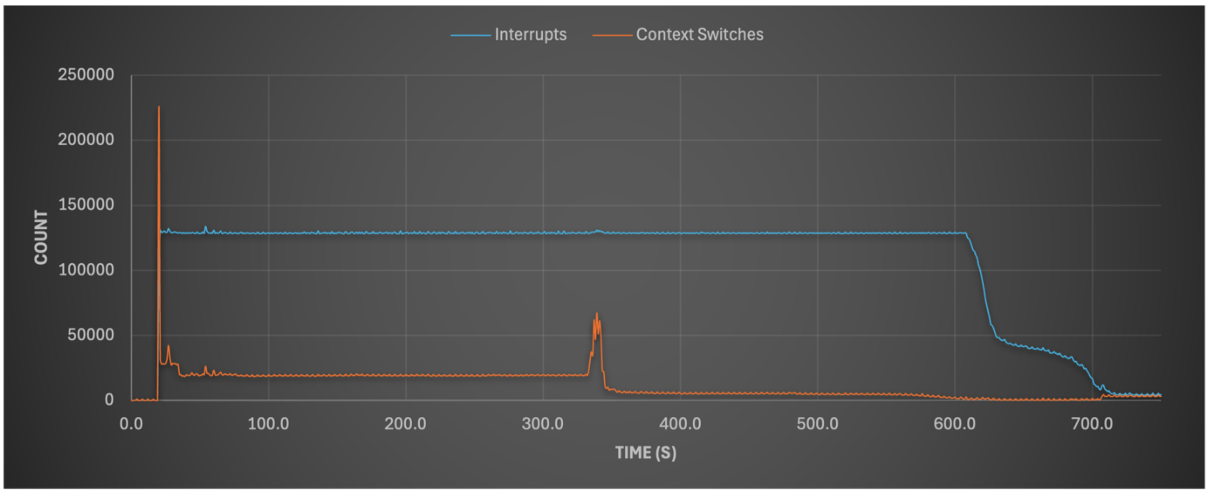


Figure 17: X axis: Time. Y axis: Number of interrupts and context switches. Process count: 1024

Looking at the context switch activity for 5120 processes, (10:1) the behaviour seems far less reminiscent of a true queuing mechanism. which also correlates to a drop in the performance.

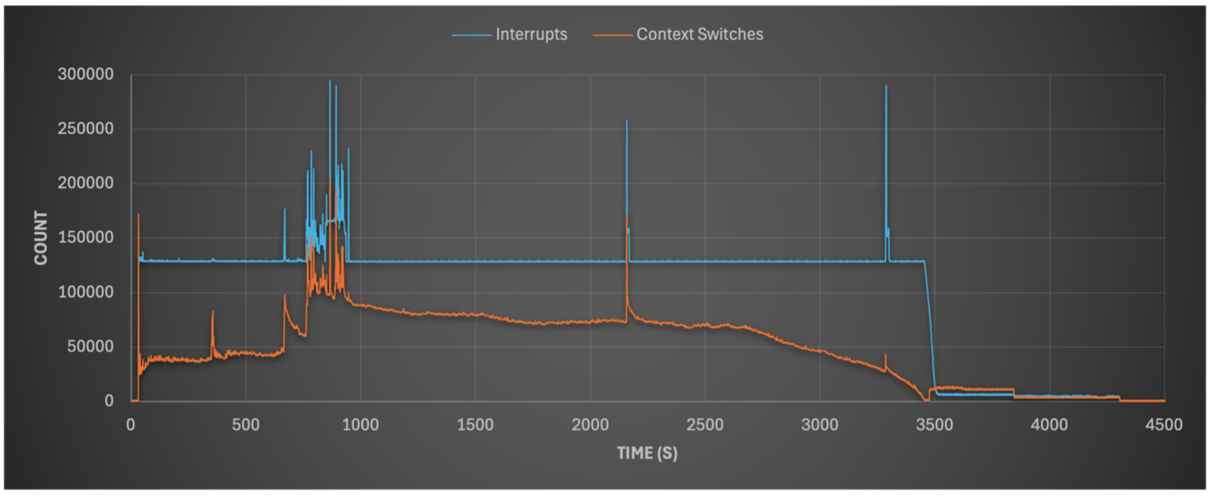


Figure 18: X axis: Time. Y axis: Number of interrupts and context switches. Process count: 5120

This increase in performance through oversubscription is another indicator that COREx may not be bound purely by the ALU/FPU/SIMD units in the CPU but also by its ability to access memory and CPU cache sizes.

Key Point 7
Where latency is not a concern, such as end of day risk, higher throughput may be possible at ratios beyond one process per thread even where code

Practical Application

Although this behaviour is interesting, and points to the possibility of exploring higher levels of throughput beyond the traditional limit of one process per logical CPU even in CPU bound workloads, extrapolating oversubscription to other workloads without testing would be inadvisable. The ratio of non-CPU activity to CPU activity within the workload, and the points at which this occurs, is likely to play a significant role in the benefits of adopting oversubscribing workload.

It does, however, show that where workload is not latency sensitive, there may be some value in deliberately oversubscribing the CPUs, i.e. allowing the OS to schedule workload instead of the HPC scheduler in certain cases.

Performance per Watt: Power usage and Energy Efficiency

Having access to physical machines rather than cloud VMs for benchmarking allows measurement of power utilisation in addition to performance. This provides another valuable data point required to calculate total cost of ownership for on-premises HPC operation.

Power utilisation for both the 4th generation EPYC 9654 and 5th generation EPYC 9755 CPUs running COREx with SMT enabled and disabled was measured using a Yokogawa WT310 power meter connected to a Windows machine via USB to record voltage, current draw and, therefore, power consumption.

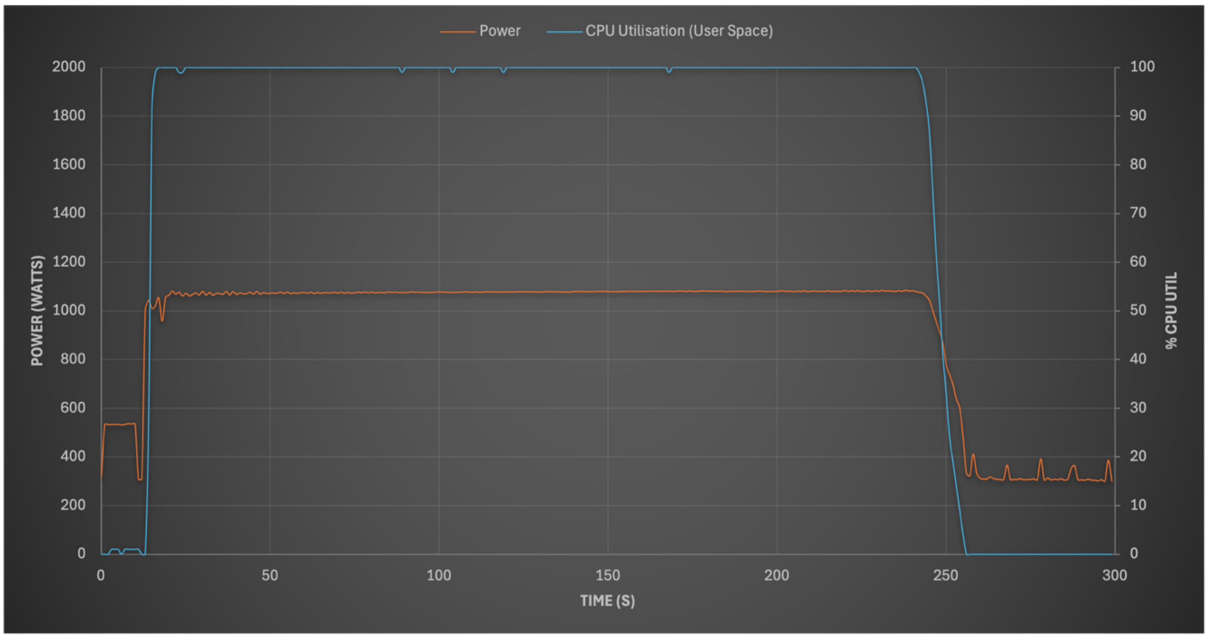


Figure 20: Y axes: Power and CPU utilisation. X axis: Time. SMT Disabled. EPYC 9654 (4th gen) running COREx

In both cases, Zen 4 and Zen 5, enabling SMT increases the peak power usage

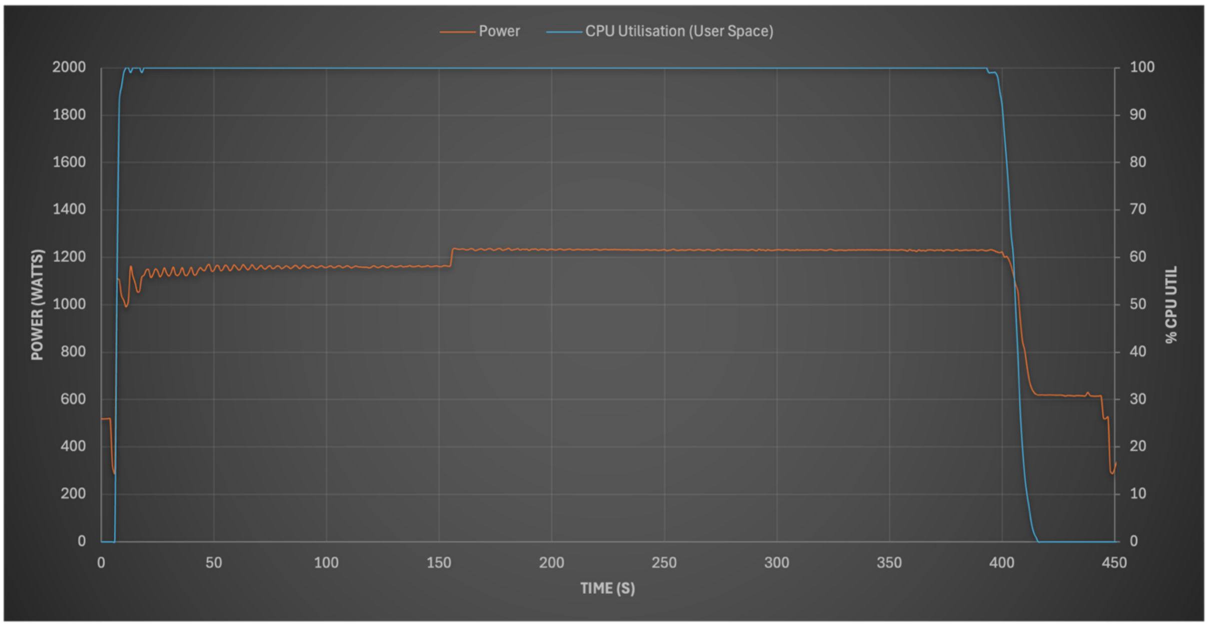


Figure 21: Y axes: Power and CPU utilisation. X axis: Time. SMT Enabled. EPYC 9654 (4th gen) running COREx

The peak power usage of the EPYC 9755 (5th gen) is also higher than the EPYC 9654 (4th gen). In fact, the 9755 with SMT enabled consumes more power than the 9654 with SMT disabled.

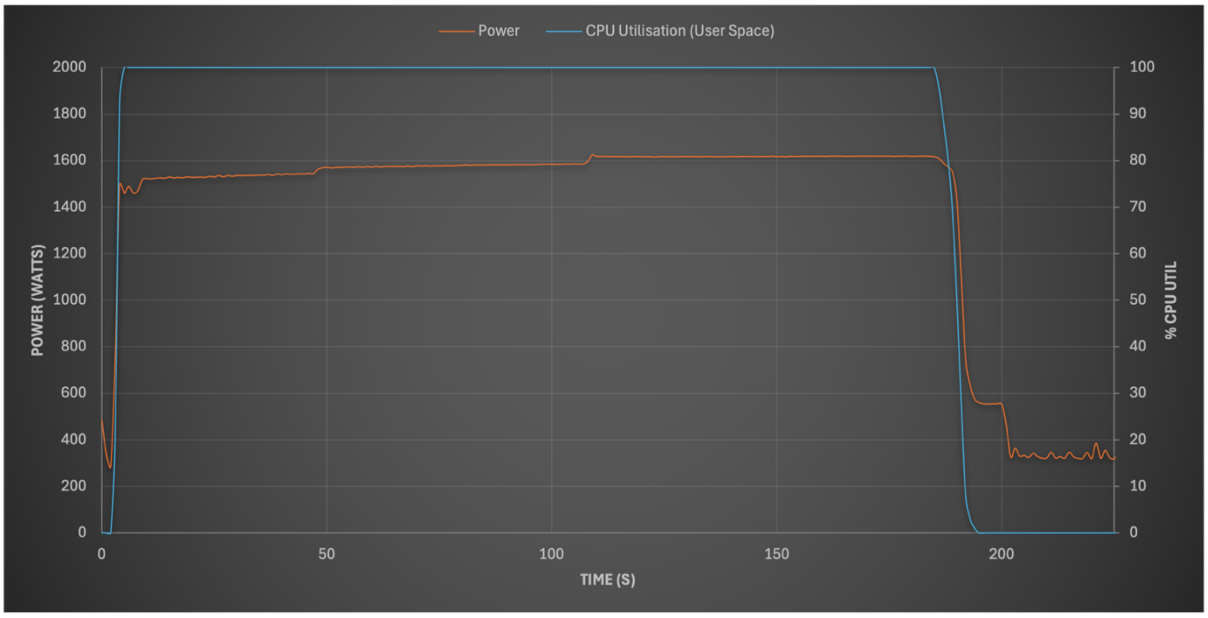


Figure 22: Y axes: Power and CPU utilisation. X axis: Time. SMT Disabled. EPYC 9755 running COREx

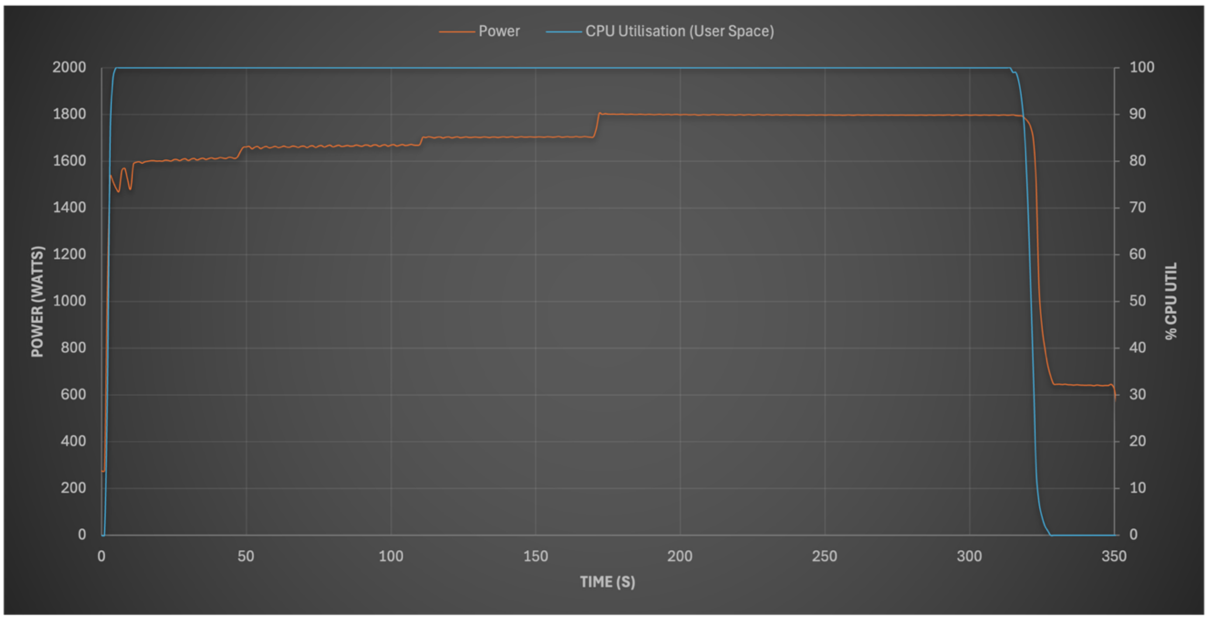


Figure 23: Y axes: Power and CPU utilisation. X axis: Time. SMT Enabled. EPYC 9755 running COREx

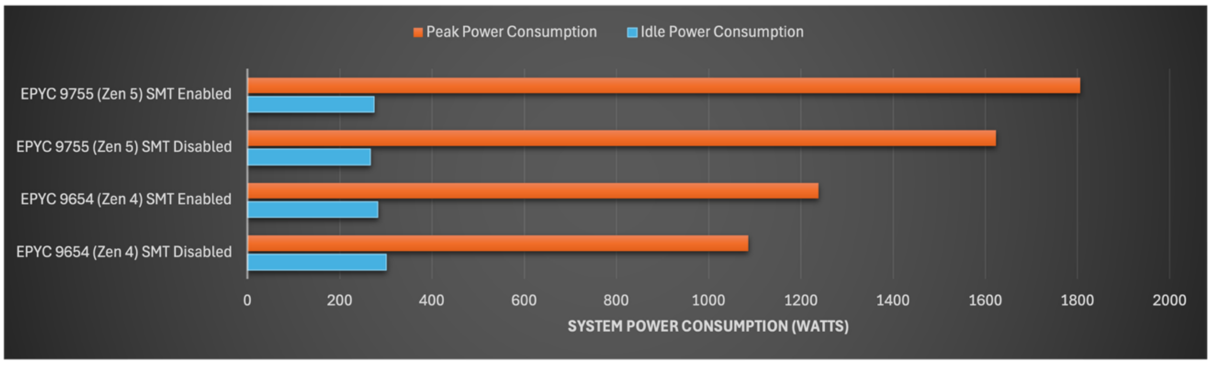


Figure 24: Power consumption at idle and peak load whilst running COREx.

Peak power utilisation per logical CPU (both with SMT enabled and disabled) is lower on the EPYC 9755 (5th gen) than the EPYC 9654 (4th gen) CPUs.

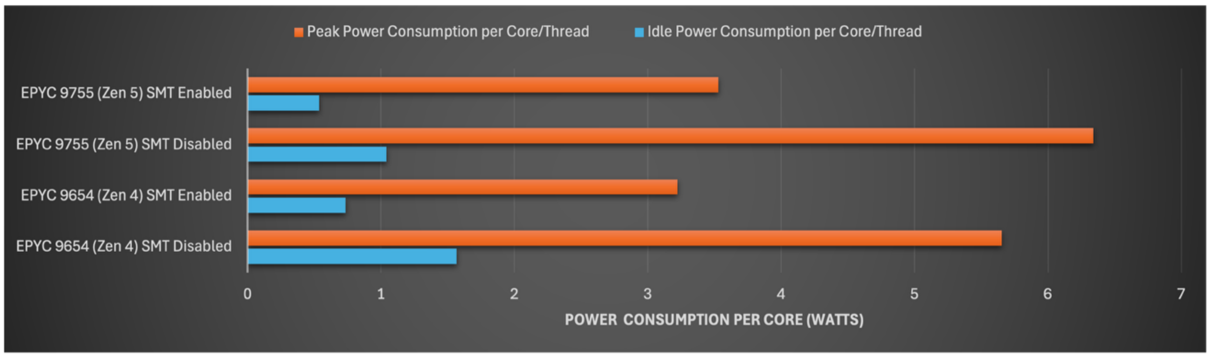


Figure 25: Power consumption at idle and peak load whilst running COREx per logical CPU

More importantly, the total energy required to execute one COREx process is lower with an EPYC 9755 (5th gen) CPU than a EPYC 9654 (4th gen) CPU, i.e. the 9755 is not only more performant but also more energy efficient by 14% with SMT enabled and 13% with SMT disabled.

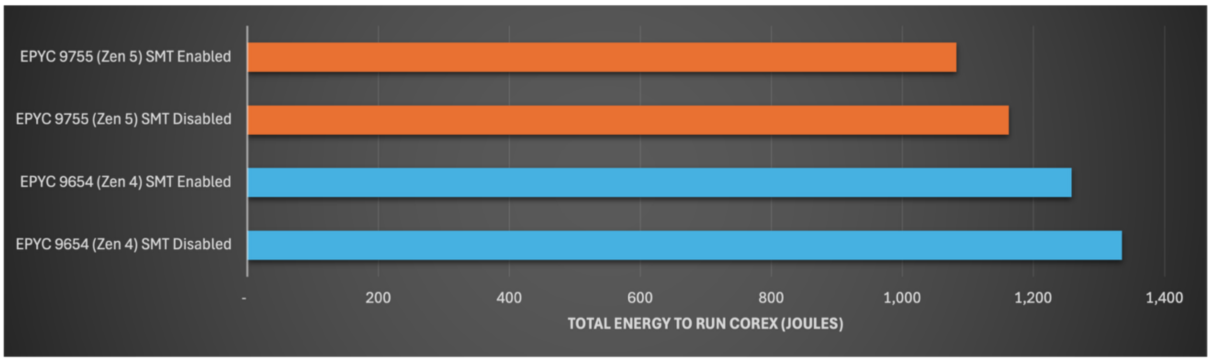


Figure 26: Total energy consumed in Joules (lower is better) per COREx process for EPYC 9654 (4th gen) and EPYC 9755 (5th gen). Lower is better.

Key Point 8

Enabling SMT increases peak power usage but also energy efficiency for COREx workloads with Zen 5 using 14% less energy for the same work.

Power Usage: COREx vs Passmark

While Passmark CPU, a CPU performance benchmark, does exhibit 100% CPU utilisation (at least under the EPYC 9654), it does have a lower peak power utilisation than COREx. This could be due to the shorter benchmark execution time, but this difference was important to recognise. It points to the possibility of using peak power demand in addition to CPU utilisation as a metric to determine how hard the system is being driven. However, the practicality will likely limit this approach to bench testing only, as opposed to forming a part of the observability metrics that are routinely collected. Having said that, smaller devices such as a Shelly Pro 3EM integrated into each rack's PDU could allow for power consumption to form part of the observability metrics of a HPC cluster.

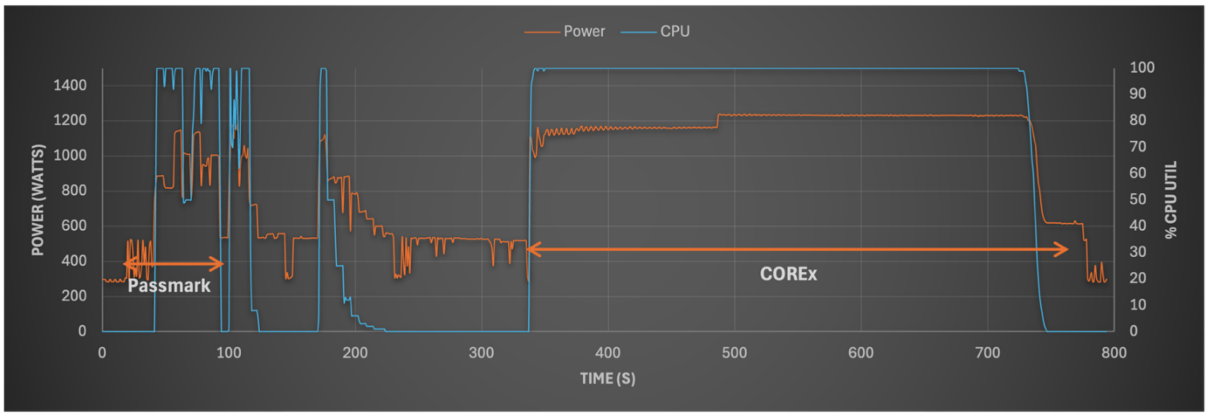


Figure 27: Power Usage under Passmark and COREx. Y axis: Power consumption. X axis: Time. Power usage during COREx vs Passmark. SMT disabled

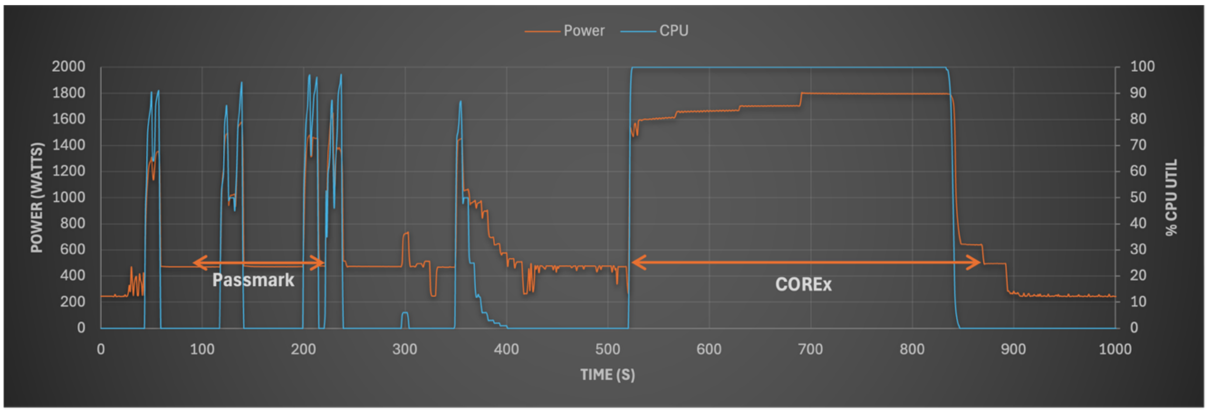


Figure 28: Power Usage under Passmark and COREx. Y axis: Power consumption. X axis: Time. Power usage during COREx vs Passmark. SMT disabled

AMD EPYC 9654 and 9755 Specifications

The 5th generation EPYC 9755[x] is a top of rack 128 core CPU with SMT providing a possible 256 threads per socket. The CPU operates at a base clock frequency of 2.7GHz with a max and all core boost speed of 4.1GHz. The tests were performed on a dual socket system, i.e. a total of 256 cores and 512 threads (with SMT enabled). The L3 cache is specified as 512MB. No data are provided by AMD for the L1 and L2 cache sizes.

The 4th generation EPYC 9654[xi] is a 96 core CPU with SMT providing a possible 192 threads per socket. The CPU operates at a base clock frequency of 2.4GHz with an all core boost speed of 3.55GHz and a max boost speed of 3.7GHz. The tests were performed on a dual socket system, i.e. a total of 192 cores and 384 threads (with SMT enabled). The L3 cache size is specified as 384MB. It should be noted that the equivalent part to the above would be an EPYC 9754 but this was not available for us to test.

The above specifications were largely confirmed in the `lscpu` output, except for the minimum and maximum clock frequencies, which were reported as 1.5GHz and 2.7GHz for the 9755 and 1.5GHz and 2.4GHz for the 9654. Further L3 cache for the 9654 is reported as 512MB as opposed to the stated 384MB.

The disparity in the observed clock speed has been explained by AMD. The EPYC CPUs have three power states:

- P0: The rated base frequency (also the default full-performance state)
- P1: A mid-level frequency below the base
- P2: The lowest active-state frequency (typically used when idle)

The observed clock frequencies were always at idle and, therefore, the lowest active state (P2). These have been confirmed by AMD as 1.5GHz for both CPUs.

	EPYC 9654 (4th gen)		**EPYC 9755 (5th gen)**	
	Specification	Observed	Specification	Observed
Core Count	96	96	128	128
Base Clock	2.4GHz	1.5GHz	2.7GHZ	1.5GHz
All Core Boosted Clock	3.7GHz		4.1GHz	
L3 Cache	384MB	512MB	512MB	512MB

Acknowledgements

The authors would like to thank Sarina Sit, Dat Ho and the rest of the team at AMD for access to the 4th and 5th generation AMD EPYC CPU systems and providing additional details where required.

We'd like to further thank the team at HMx Labs and other disinterested parties for their input and reviewing this work.

References

[i] https://www.passmark.com/products/performancetest/pt_advcpu.php?srsltid=AfmBOordpYxBHbgdXiRKwQFsGpbOvua9OxC3IRuOiU3q9NOGms7DHECO

[ii] https://www.geekbench.com

[iii] https://top500.org/project/linpack/
[iv] https://www.quantlib.org/
[v] https://www.opensourcerisk.org/
[vi] https://www.intel.com/content/www/us/en/developer/tools/oneapi/onemkl.html
[vii] https://cloudhpc.news/hyperthreading-in-hpc/
[viii] https://www.amd.com/en/blogs/2025/driving-amd-epyc-cpu-based-storage-solutions-for-enterprise.html
[ix] https://cloudhpc.news/intro-to-hpc-2/
[x] https://www.amd.com/en/products/processors/server/epyc/9005-series/amd-epyc-9755.html

[xi] https://www.amd.com/en/products/processors/server/epyc/4th-generation-9004-and-8004-series/amd-epyc-9654.html