AMD EPYC

WHITE PAPER

# INTRODUCING A NEW QUANTLIB BENCHMARK VERSION

AMD

together we advance_data center computing

*First Edition*
*November 2024*

*Jacques du Toit and Sarina Sit*

# INTRODUCING A NEW QUANTLIB BENCHMARK

## CONTENTS

# INTRODUCTION

The majority of quantitative finance and risk management workloads are proprietary applications. Financial institutions guard their mathematical models and codes closely to maintain their competitive advantage. This limited access to suitable codes complicates benchmarking computer hardware for quantitative risk management workloads. However, a few commonly cited benchmarks do exist.

For example, STAC-A2™* has a rather narrow definition consisting of the Heston Model, Anderson QE sampling, and AMC regressing on a quadratic with no calibration. Hardware vendors have optimized the STAC-A2 benchmark extensively, including exploiting the Vandermonde structure of the regression matrix. STAC-A2™ benchmark on POWER8* presents a good overview (subscription required).

Other examples of quantitative risk management benchmarks include:

- FinanceBench* ports a small subset of simple pricing models (Bonds, Black-Scholes, one factor Monte Carlo) to OpenMP, CUDA and OpenCL.
- COREx* by HMX Labs is an XVA benchmark built on Open Risk Engine and executes the same single-threaded XVA calculation on all CPU cores.
- QuantLib* is a well-known open-source project that also ships with a benchmark*. More on this below.

*Note: This list does not include various finance-related codes available online that are occasionally mentioned in the context of benchmarking quantitative risk applications.*

Benchmarks are useful for highlighting technologies that warrant closer investigation. An overnight risk calculation in a bank involves many steps that include curve stripping, vol surface construction, calibration, simulation, and pricing for a wide variety of models and products. These calculations exercise a range of numerical methods from (semi)-closed form formulas to trees, Partial Differential Equations (PDEs), and Monte Carlo.

The software libraries that perform all these tasks are large object-oriented code bases that must manage the complexity of supporting multiple models and numerical methods, as well as navigate the infrastructure and grid/end user compute constraints within the institution. These software libraries typically also carry many legacy models and methods since financial contracts can be long lived and pricing and risk management needs to remain consistent across time frames.

An ideal quantitative risk benchmark should capture the essence of this modeling and algorithmic richness to present a more balanced view of the technology being tested. However, most of the benchmarks mentioned previously tend to consist of one or two relatively "classical" models (e.g., Black-Scholes, Heston, Hull-White, etc.) and usually only one numerical technique (e.g., Monte Carlo). COREx uses a production-quality XVA engine to perform real XVA calculations, but the portfolio is rather simple (Swaps, Swaptions, Caps/Floors, one Bermudan swaption) and the models are all one factor.

This white paper presents the Quantlib v1.35 benchmark version and then showcases performance results across generations of data center processors.

## THE QUANTLIB BENCHMARK

QuantLib is a free, open-source library of tools for modeling, trading, and managing risk in quantitative finance. Banks, asset managers, financial firms, regulatory agencies, and researchers can use QuantLib to build models, create new financial products, establish pricing, and mitigate risks. It is the only open-source project that comes close to encompassing the same scope as an overnight risk calculation. It stands to reason that a benchmark built on QuantLib could be useful in determining how different machines would perform relative to each other on the proprietary codes that financial institutions use.

This white paper introduces a new benchmark for quantitative risk analysis available in QuantLib v1.35 and explains the principles behind the design. The QuantLib package has always had a benchmark, but it was completely sequential until the release of QuantLib v1.31 in August 2023. QuantLib v1.31 leaned towards classical one factor models and was based around a series of QuantLib tests that performed option pricing and/or low-level numerics, such as computing Mersenne Twister discrepancy. The number of FLOPs that each test performed was measured beforehand and hard coded into the benchmark, and the overall score was the average Million FLoating-point OPerationS per second (MFLOPS/s) achieved across all the tests.

Basing the benchmark score on the achieved MFLOPS/s is intuitive but can lead to some surprising results. Table 1 shows a typical sample of QuantLib v1.31 benchmark output showing MFLOPS/s and runtime of each test. The top four tests account for almost 60% of the total MFLOPs yet account for only ~1.57% of the overall runtime.

| Test Name | MFLOPS/s | Runtime (s) | % of Total Runtime |
|---|---|---|---|
| InterpolationTest::testSabrInterpolation | 21,448.90 | 0.106 | ~0.39% |
| ConvertibleBondTest::testBond | 9,164.10 | 0.018 | ~0.07% |
| RandomNumber::MersenneTwisterDescrepancy | 6,189.90 | 0.154 | ~0.56% |
| ShortRateModel::Swaps | 6,072.90 | 0.075 | ~0.27% |
| DigitalOption::MCCashAtHit | 2,949.40 | 0.338 | ~1.23% |
| BatesModel::DAXCalibration | 2,920.80 | 0.683 | ~2.49% |
| RiskStatistics::Results | 2,424.40 | 0.124 | ~0.45% |
| BarrierOption::BabsiriValues | 2,160.00 | 0.408 | ~1.49% |
| AsianOption::MCArithmeticAveragePrice | 2,139.00 | 2.425 | ~8.86% |
| FdHestonTest::testFdmHestonAmerican | 2,131.30 | 0.110 | ~0.40% |
| EuropeanOption::FdMcEngines | 1,979.40 | 1.005 | ~3.67% |
| BasketOption::TavellaValues | 1,927.80 | 0.484 | ~1.77% |
| MarketModelSmmTest::testMultiSmmSwaptions | 1,829.80 | 6.146 | ~22.44% |
| MarketModelCmsTest::testCmSwapsSwaptions | 1,293.40 | 8.890 | ~32.46% |
| BasketOption::OddSamples | 1,272.00 | 0.505 | ~1.84% |
| HestonModel::DAXCalibration | 1,140.20 | 0.487 | ~1.78% |

*Table 1: Example MFLOPS/s and runtime of various tests within the QuantLib v1.31 benchmark*

| Test Name | MFLOPS/s | Runtime (s) | % of Total Runtime |
|---|---|---|---|
| AmericanOption::FdAmericanGreeks | 1,002.40 | 0.517 | ~1.89% |
| JumpDiffusion::Greeks | 994.30 | 0.436 | ~1.59% |
| DividendOption::FdAmericanGreeks | 962.60 | 1.157 | ~4.23% |
| EuropeanOption::ImpliedVol | 885.00 | 0.149 | ~0.54% |
| DividendOption::FdEuropeanGreeks | 620.90 | 1.529 | ~5.58% |
| BasketOption::EuroTwoValues | 533.90 | 0.637 | ~2.33% |
| QuantoOption::ForwardGreeks | 451.90 | 0.201 | ~0.73% |
| EuropeanOption::FdEngines | 185.50 | 0.800 | ~2.92% |

*Table 1: Example MFLOPS/s and runtime of various tests within the QuantLib v1.31 benchmark (Continued)*

QuantLib v1.32 parallelized this benchmark by having each worker process execute a copy of the benchmark (on average). The results were summed to produce the overall system MFLOPS, but the discrepancy between wall time and MFLOPS remained.

Wall time is the key metric. Cloud bills and internal cross charges within institutions are based on the amount of time a resource was occupied and not on the number of FLOPs executed. In other words, time spent is the metric that correlates most closely with money spent. Measuring performance in terms of how long a machine takes to perform representative work is therefore a robust approach that most benchmarks adopt.

## BENCHMARK GOALS

An ideal quantitative benchmark should:

- Verify that calculations return correct answers. Some compilers can be too aggressive out of the box, and the numerics in financial simulations are sometimes delicate.
- Use system throughput as the overall performance metric. Overnight risk is a massively parallel throughput problem. Let's take a simple example where Machine A has 64 cores and each core can complete one VaR calculation in 160s, while Machine B has 40 cores and each core can complete one VaR calculation in 120s. Machine A is preferable because it completes 64/160=0.4 VaR calcs/s while Machine B only completes 40/120=0.33 VaR calcs/s. Additional variables such as power draw and system cost could affect this conclusion.
- Perform the same steps as an overnight risk calculation (curve stripping, calibration, pricing) for a range of products, models, and numerical methods that include industry-standard multi-factor models.
- Express a large volume of inhomogeneous work (tasks) and allow the machine to schedule tasks as it would in a production grid environment without imposing load balancing by design.
- Minimize tail effects.

These points formed the basis for re-working the benchmark in QuantLib v1.35 into an effective means of comparing machines for overnight risk calculations. The new benchmark comprises more than 80 QuantLib tests that range in duration from less than one second to approximately 15 seconds. The test suite includes:

- Models such as Heston, Bates, Heston SLV, Hull-White, multifactor Market Models, Markov functional, Zabr, Gaussian Copula, and Kluge Extended Ornstein-Uhlenbeck.
- Numerical methods including Monte Carlo, American Monte Carlo (Longstaff-Schwartz), PDEs, trees, and closed form formulas.
- Curve building, volatility interpolation, calibration, pricing, and select low-level mathematical and statistical tests.

This re-worked benchmark is part of the Phoronix Test Suite. Open Benchmarking* reports results from a range of systems.

## CONTROLLING THE AMOUNT OF WORK

The QuantLib v1.35 benchmark contains a size factor that determines how many times to execute the test set. If the size equals 1, then the queue contains one instance of each test, and each test becomes one task. A size of 10 places 10 instances of each test in the queue. The benchmark predefines a set of "T-shirt" sizes (i.e., "XXS", "XS", "S", "M", etc.), where the "S" size will keep most modern, large dual-socket servers busy for about 5 minutes.

The benchmark reports the overall wall time taken and the number of tasks per second that the system completed. This is the main metric to use when comparing different systems. The tasks/s metric remains relatively constant across benchmark sizes that fully loaded the system.

## PARALLELIZATION AND LOAD BALANCING

The benchmark contains an `nthreads` parameter that sets the number of forked child processes. The QuantLib library is not thread safe, meaning that parallelization must be implemented via multi-processing. By default, the benchmark forks one worker process for each core the operating system sees, which can be either 1x or 2x the number of physical processor cores depending on whether logical core threading is enabled.

- The Simultaneous Multithreading (SMT) BIOS setting controls logical core threading on AMD EPYC™ processors.
- The Hyperthreading BIOS setting controls logical core threading on Intel® processors.

*Note: In general, the benchmark benefits from enabling logical core threading on both AMD and Intel systems.*

The benchmark uses the Boost inter-process communication library to coordinate the master and worker processes. The master process performs basic setup, forks all worker processes, distributes work, and receives results. The worker processes perform the computations. This setup does impose some overhead at the start of the benchmark – more on this below.

Production quantitative risk systems keep machines busy for hours at a time by flowing a constant stream of tasks across the machines in the grid. A single grid machine does not have much of a "tail effect" wherein one process finishes long after all the others, but this is a real problem for benchmarking. The benchmark should return results in only a few minutes, and those results should not be skewed by tail effects. In practice, the last two goals listed in "Benchmark Goals" on page 5 are in conflict because only careful load balancing can control tail effects.

The benchmark handles this by hard coding a relative ranking for each test: high ranks for longer running tests and low ranks for short ones. (Rankings need not be precise.) It then sorts the task hopper by ranking and issues the most expensive tasks first.

The tail effect is the ratio of the average tail lifetime to the lifetime of the master process. Each worker process records timestamps on creation and on completion of its last task, and the difference between these timestamps defines the worker lifetime. Lifetimes are ranked from shortest to longest, and the average tail lifetime is the geomean of the shortest 10% of lifetimes. The tail effect is the ratio of the average tail lifetime to the master process lifetime. A tail effect of 1 means that all workers ran for the same amount of time as the master process. A tail effect significantly less than 1 indicates that several workers finished significantly before the master process, which indicates a nontrivial load imbalance. Running the benchmark executable with the `–verbose=1` flag will print out the tail effect ratio.

The benchmark performs no load balancing beyond issuing the longest running tasks first. The reported tail effect warns users of a potential load imbalance that should be solved by increasing the benchmark size.

## CHECKING FOR THE RIGHT ANSWER

QuantLib uses the Boost unit test framework to execute tests and report failures, which imposes considerable overhead on running tests and is therefore not ideal for benchmarking. One can avoid this overhead by running the test bodies independently of the unit test framework, but this disables all the checking normally performed by the tests. Checking for the right answer must incur the significant overhead of enabling the unit test framework.

The QuantLib v1.35 benchmark mitigates this issue by running the first execution of every test through the unit test framework and propagating any errors back to the master process, which terminates the benchmark. All subsequent executions of the same test call the test body directly and bypass the unit test framework.

## BENCHMARK CHARACTERIZATION

The testing described in this white paper used the AMD uProf performance analysis tool to characterize the benchmark. Executing the following command on an AMD EPYC system generates an HTML report showing how the machine responds to the benchmark workload and includes graphs like the ones presented below.

```
AMDuProfPcm --html --collect-psi -O ./
<quantlib_install_path>/bin/quantlib-benchmark --
size=S –verbose=1
```

Figures 1-7 were collected from tests running on a 2P server powered by 4th Gen 96-core AMD EPYC 9654 processors (192 total physical cores).

*Note: The results shown in Figures 1-7 are for illustrative purposes only with no performance claims being made or implied.*

Figure 1 shows the traditional "top-down" pipeline analysis for the QuantLib v1.35 benchmark with a roughly equal division between retiring, frontend bound, backend bound, and SMT contention. See A Top-Down Method for Performance Analysis and Counters Architecture* for a description of the test methodology used.


*Figure 1: Top-down Quantlib v1.35 analysis*

"Retiring" represents the proportion of dispatch slots that resulted in retired instructions, and "Backend" represents the proportion of dispatch slots that were unable to issue instructions due to stalls on the processor backend. AMD adds the SMT contention category to represent the fraction of dispatch slots that this thread was unable to use because the other SMT thread was using them. In this example, all SMT threads are helping execute the benchmark and are, on average, seeing the same retiring/stall statistics as this thread, meaning that one can effectively ignore SMT contention.

Figure 2 shows a whole-machine aggregate CPU utilization plot produced by uProfPcm profiling the QuantLib v1.35 benchmark. Here, the benchmark remains virtually inactive for about the first 9 seconds.

This is the phase during which all 384 processes are launching and the Boost IPC communication infrastructure is being set up. The QuantLib v1.35 benchmark includes this startup cost, while QuantLib v1.36 and later place the timestamps so as to exclude this overhead. Startup time can thus be safely ignored.

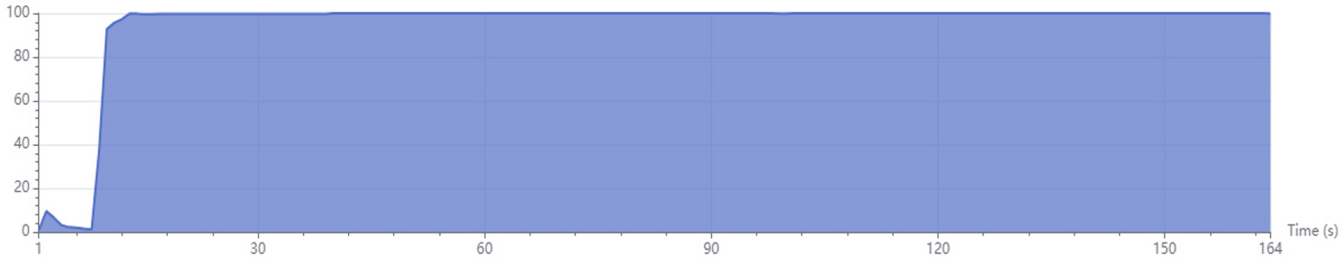



*Figure 2: Whole-machine aggregate CPU utilization plot*

Figure 3 shows an Instructions Per Cycle (IPC) plot produced by uProfPcm, throughout the duration of the QuantLib benchmark. In this example, IPC varies throughout the benchmark, hovering at about 1.5 and dipping as low as 0.6 on occasion.

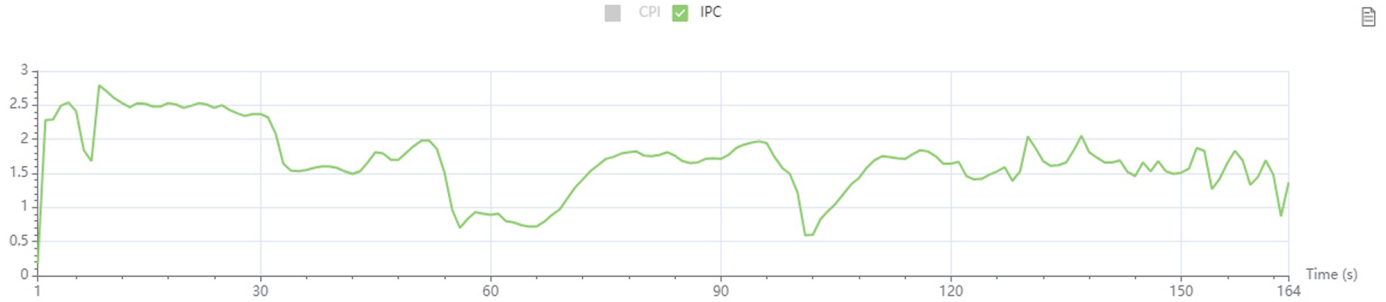



*Figure 3: IPC plot*

Figure 4 shows L1 and L2 cache misses Per Thousand Instructions (PTI). The benchmark has a very low L2 cache miss ratio that peaks at 52 PTI but generally remains around 20 PTI.

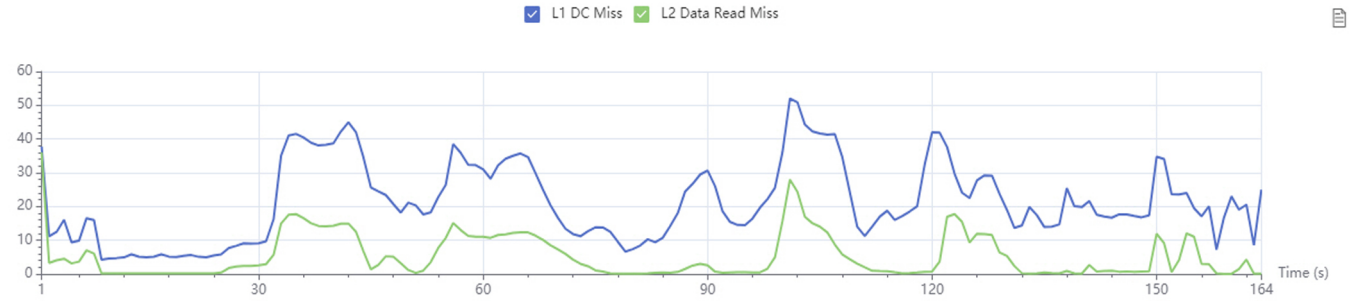



*Figure 4: L1 and L2 cache misses.*

Figure 5 shows where data was fetched from (data cache fills per thousand instructions). This example shows the vast majority of data coming from caches local to the CCX (processor tile).

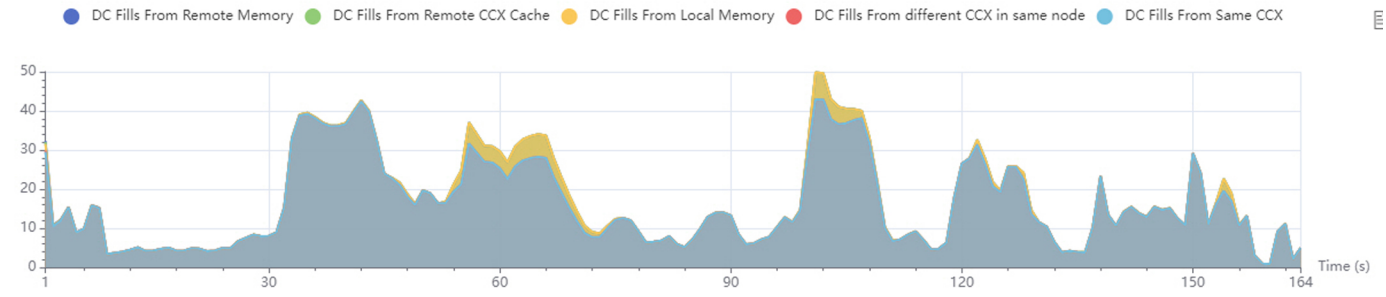



*Figure 5: Diagram 5. Data cache (DC) fill plot from uProfPcm.*

Comparing Figure 5 with the achieved memory bandwidth plotted in Figure 6 illustrates that some parts of the benchmark show very high DRAM bandwidth. The low L2 misses and high DRAM bandwidth imply that the prefetchers are doing a good job of fetching data ahead of need.

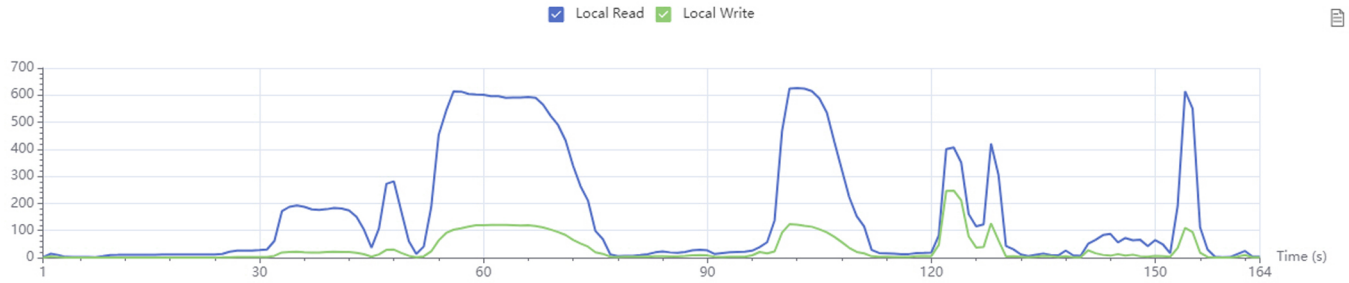



*Figure 6: Local read and local write memory speeds, per uProfPcm.*

Figure 7 shows the instruction mix between scalar, SSE, AVX, and AVX-512 and reveals that QuantLib v1.35 mainly consists of scalar and SSE instructions.

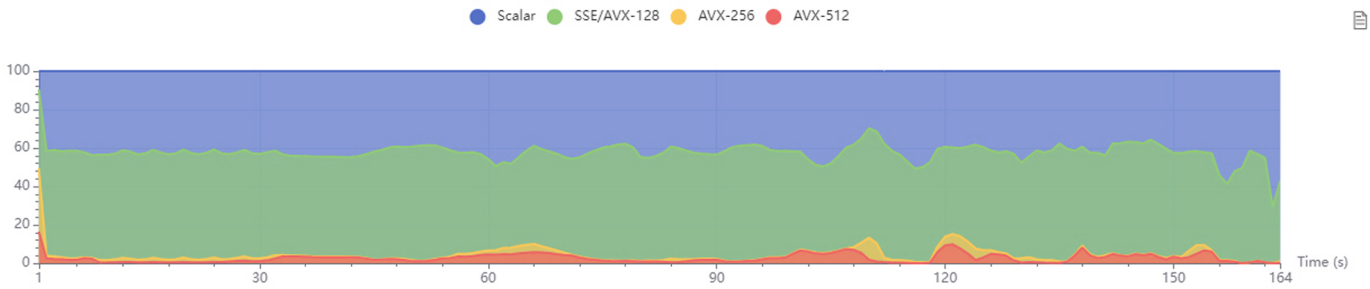



*Figure 7: Instruction mix*

AMD compiled this benchmark binary using AOCC v4.2 with the following flags set:

```
-O3 -zopt -march=znver4 -fveclib=AMDLIBM -lamdlibm
```

This benchmark has significant optimizations enabled but does not use many AVX or AVX512 instructions. This suggests many loop structures where the compiler is either:

- pessimistic on the value of wider vectors, or
- inserting both AVX and SSE versions of a loop where the low trip count at runtime indicates that the SSE2 version is preferable.

The full uProfPcm report provides many other interesting insights, such as heat maps of processor activity, L3 cache misses and latencies over time, and time-aggregated metrics.

Executing the following command reveals that floating point operations comprise approximately 11% of all retired operations:

```
perf stat -e fp_ops_retired_by_type.all,ex_ret_ops
quantlib-benchmark --size=S -verbose=1
```

HPC codes typically see 20%-40% of floating point operations at retirement, but finance codes tend to have significant object creation/management, small loops, memory handling, and control flow in addition to floating point math. It would be interesting to compute these metric and graphs for in-house quantitative libraries and compare those results against the benchmark.

# QUANTLIB PERFORMANCE RESULTS

AMD tested the relative performance and performance per Watt of a range of dual-socket systems powered by 3rd-5th Gen AMD EPYC and 1st-5th Gen Intel Xeon Platinum processors running the QuantLib v1.35 benchmark.

AMD used the following compilers to generate optimized binaries:

- AOCC v4.2 with the `-O3 -zopt -fveclib=AMDLIBM -lamdlibm` flags to test the AMD EPYC systems described in Table 2, below. The 3rd Gen AMD EPYC 7763 system also used the `–march=znver3` flag, while the 4th Gen and 5th Gen AMD EPYC systems used the `–march=znver4` flag.
- Intel® oneAPI Compiler Suite 2024* with the `–O3 –xCORE-AVX512 -fp-model=precise` flags to test the Intel® Xeon® systems described in Table 3, below. This compiler enables aggressive floating-point optimizations by default. The QuantLib benchmark tests fail unless the compiler is instructed to use a precise floating-point model.

Tables 2 and 3 describe the AMD EPYC and Intel Xeon systems used for the testing described in this section.

| AMD System Configurations | | | |
|---|---|---|---|
| CPUs | 2 x 3rd Gen AMD EPYC 7763 | 2 x 4th Gen AMD EPYC 9654 | 2 x 5th Gen AMD EPYC 9755 |
| TDP/socket (W) | 280 | 400 | 500 |
| Frequency: Base \| Boost[1] | 2.45 GHz \| 3.5 GHz | 2.40 GHz \| 3.70 GHz | 2.70 GHz \| 4.10 GHz |
| Cores per CPU | 64 | 96 | 128 |
| L3 Cache per CPU | 256 MB | 384 MB | 512 MB |
| Memory | 2.0 TB (16 x 128 GB DDR4-3200) | 768 GB (24 x 32 GB DDR5-4800) | 1.5 TB (24 x 64 GB DDR5-6000) |
| Storage | Dell YDHYX Samsung SM883 MZ7KH960HAJR0D3 SSD | Western Digital Black SN850 WDS100T1X0E-00AFY0 NVMe | Samsung PM9A1 MZVL2512HCJQ-00B00 NVMe |
| BIOS | 2.7.3 | RTI1006C | TVOT0090A |
| BIOS Settings | Virtualization Technology=OFF, Kernel DMA Protection=OFF, L3 cache as NUMA domain=OFF, SMT=OFF, System Profile= Performance, CPU Power Management=Max Performance, Memory Frequency=Max, Turbo Boost=Enabled, C states= Disabled, Memory Patrol Scrub=Standard, PCI ASM Link Power Management=OFF, Determinism Slider=Power, Efficiency Optimized Mode= Disabled, ABPDIS=Disabled | TDP Control=400W, PPT Control=400W, Determinism=Power, Power Profile=High Performance, NPS=4, L3 Cache as NUMA Domain= Disabled | Determinism=Power, Probe Filter Organization=Shared, GMI Folding = Disabled, Memory Clock Speed=4800/5600/6000, EDC Controller=Disabled, GMI DLWM=Disabled, FP Di/Dt= Disabled, L3 DFLL Stretch Disable=Enabled, DRAM UEC Retry=Enabled |
| OS | Ubuntu 20.04 LTS | Ubuntu 22.04 LTS | |

*Table 2: AMD system configurations used for testing*

| AMD System Configurations | | |
|---|---|---|
| Runtime Tunings | echo 3 > /proc/sys/vm/drop_caches | vm.dirty_ratio=8, vm.swappiness=1, vm.zone_reclaim_mode=1, kernel.randomize_va_space=0, Transparent huge pages = ON, Page defrag = ON, echo 3 > /proc/sys/vm/drop_caches |

*Table 2: AMD system configurations used for testing (Continued)*

| Intel System Configurations | | | | | |
|---|---|---|---|---|---|
| CPUs | 2 x 1st Gen Intel Xeon Platinum 8180 | 2 x 2nd Gen Intel Xeon Platinum 8280 | 2 x 3rd Gen Intel Xeon Platinum 8380 | 2 x 4th Gen Intel Xeon Platinum 8490H | 2 x 5th Gen Intel Xeon Platinum 8592+ |
| TDP/socket (W) | 205 | 205 | 270 | 350 | 350 |
| Frequency: Base \| Boost[1] | 2.50 GHz \| 3.80 GHz | 2.70 GHz \| 4.00 GHz | 2.30 GHz \| 3.40 GHz | 1.90 GHz \| 3.50 GHz | 1.90 GHz \| 3.90 GHz |
| Cores per CPU | 28 | 28 | 40 | 60 | 64 |
| L3 Cache per CPU | 38.5 MB | 38.5 MB | 60 MB | 112.5 MB | 320 MB |
| Memory | 384 MB (12 x 32 GB DDR4-2666) | 768 MB (12 x 64 GB DDR4-2933) | 512 MB (16 x 32 GB DDR4-3200) | 1.0 TB (16 x 64 GB DDR5-4800) | 1.0 TB (24 x 64 GB DDR5-5600) |
| Storage | Dell Enterprise D1F14 Seagate Exos ST600MM0238 SATA | Dell X31G3 Intel SSDSC2KG960G8R SSD | Dell PERC H355 Front SCSI | Intel SSDPF2KE016T10 NVMe | Micron MTFDDAK480TGA-1B SSD |
| BIOS Settings | Memory Operating Mode= Optimizer, CPU Interconnect Speed=Maximum, Virtualization Technology=Disabled, Prefetchers=Enabled, Dell Controlled Turbo=Enabled, System Profile=Custom, CPU Power Management=Maximum Performance, | Virtualization Technology=Disabled, DCU Streamer Prefetcher=Disabled, Sub NUMA Cluster=Disabled, LLC Prefetch=Disabled, Dell Controlled Turbo=Disabled, System Profile=Custom, CPU Power Management=Maximum Performance, | System Profile=Custom, CPU Power Management=Max Performance, Memory Freq=Max Performance, Turbo Boost=Enabled, C1E=Disabled, C States=Autonomous, Memory Patrol Scrub=Disabled, Uncore Freq=Max, Energy Profile=Performance, | Memory Speed=Max Performance, Socket Interleave=NUMA, Patrol Scrub=Disabled, Memory Data Scrambling=Disabled, Page Policy=Adaptive, Operating Mode=Custom, P-state Control=None, C1 Enhanced Mode=Disabled, UPI Link Frequency=Max, | CPU Interconnect Speed=Max, Intel VT=Enabled, Kernel DMA=Disabled, Sub NUMA Cluster=Disabled, Prefetchers=Enabled, Directory AtoS=Disabled, Intel SST-CP=Disabled, System Profile=Performance, Optimized Power-Disabled, CPU Power Management=Max Performance, |

*Table 3: Intel system configurations used for testing*

| Intel System Configurations | | | | | |
|---|---|---|---|---|---|
| BIOS Settings (Continued) | *Memory Frequency=Maximum Performance, Turbo Boost=Enabled, C1E=Disabled, C States=Autonomous, Write Data CRC=Disabled, Memory Patrol Scrub=Standard, Uncore Frequency=Max, Monitor/ MWait=Enabled, PCI ASPM L1 Link Power Management=Disabled* | *Memory Frequency=2933MHz, Turbo Boost=Enabled, C1E=Disabled, C States=Autonomous, Write Data CRC=Disabled, Memory Patrol Scrub=Disabled, Energy Efficiency Policy=Performance, Turbo Boost=Enabled, CPU Interconnect Bus Link Power Management=Disabled, PCI ASPM L1 Link Power Management=Disabled* | *CPU Interconnect Bus Link Power Management=Disabled, PCI ASPM L1 Link Power Management=Disabled, Virtualization= OFF, Hyperthreading=ON, Prefetchers= Enabled, SNC=2* | *UPI Link Disable=Disabled, 1 Link Turbo=Enabled, Energy Efficient Turbo=Disabled, C-states=Legacy, Power-Performance Bias=Platform Controlled, Platform Controller=Max performance, Power Performance Bias=Platform Controlled, Platform Controlled Type=Max Performance, Workload Config=Balanced, Hyperthreading=Enabled, Prefetchers= Enabled, Intel Virtualization= Disabled, DCA= Enabled, SNC= SNC4, P-State Hysteresis=500us, CPU Frequency Limits=Full Turbo Uplift, Rocket Mode=Disabled* | *Memory Frequency=Max Performance, Turbo Boost=Enabled, Energy Efficient Turbo=Enabled, C1E=Disabled, C-states=Disabled, Memory Patrol Scrub=Standard, Uncore Freq=Max, Energy Efficiency Policy=Performance, Workload Profile=not configured* |
| OS | *Ubuntu 20.10* | *Ubuntu 22.04 LTS* | *CentOS 9* | *Fedora 38* | *Ubuntu 22.04 LTS* |
| Runtime Tunings | *echo 3 > /proc/sys/vm/drop_caches* | | | | |

*Table 3: Intel system configurations used for testing (Continued)*

Figure 8 (see following page) shows QuantLib performance across 1st through 5th Gen Intel Xeon processors compared to 3rd through 5th Gen AMD EPYC processors with all results normalized to the 1st Gen Intel Xeon Platinum 8180 system. The 5th Gen AMD EPYC 9755 system outperforms the 5th Gen Intel Xeon Platinum 8592+ system by ~2.39x (~8.30x/~3.47x). Data centers are under constant pressure to increase power efficiency in the face of climbing energy costs and sustainability goals. Overnight quantitative risk calculations are massively parallel, meaning that the time required to obtain results depends on the amount of allocated power and rack space. Financial institutions therefore pay close attention to performance per Watt when selecting systems.

Figure 9 (see following page)shows the relative QuantLib v1.35 performance per Watt across the same AMD EPYC and Xeon systems described in Tables 2 and 3 calculated using twice the Thermal Design Power (TDP) of each processor to account for the dual-socket servers. Do not confuse this with wall power consumed or processor power consumed. Here again, all results are normalized to the 1st Gen Intel Xeon Platinum 8180 system.

The 4th and 5th Gen AMD EPYC processors tested have TDPs of 400 and 500 Watts, which are higher than the 350 Watt TDPs of 4th and 5th Gen Intel Xeon Platinum systems tested. Even so, they demonstrate outstanding performance per Watt for quantitative risk workloads.

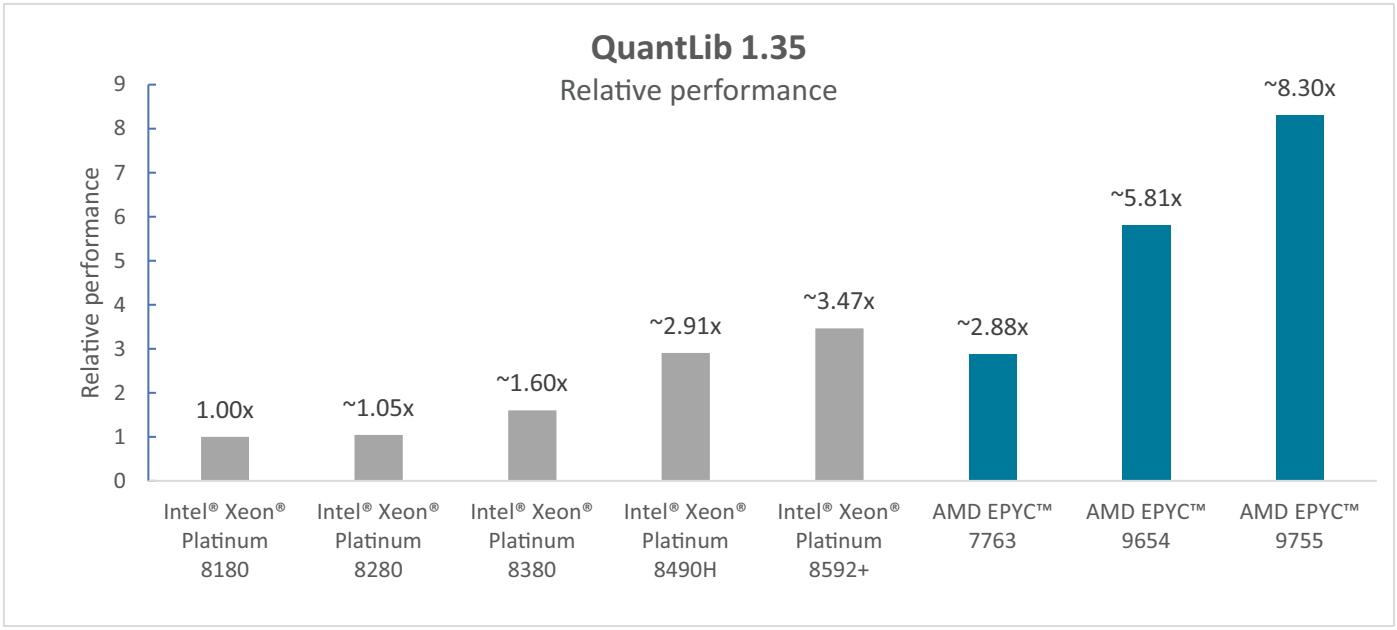



*Figure 8: Normalized relative QuantLib v1.35 performance*

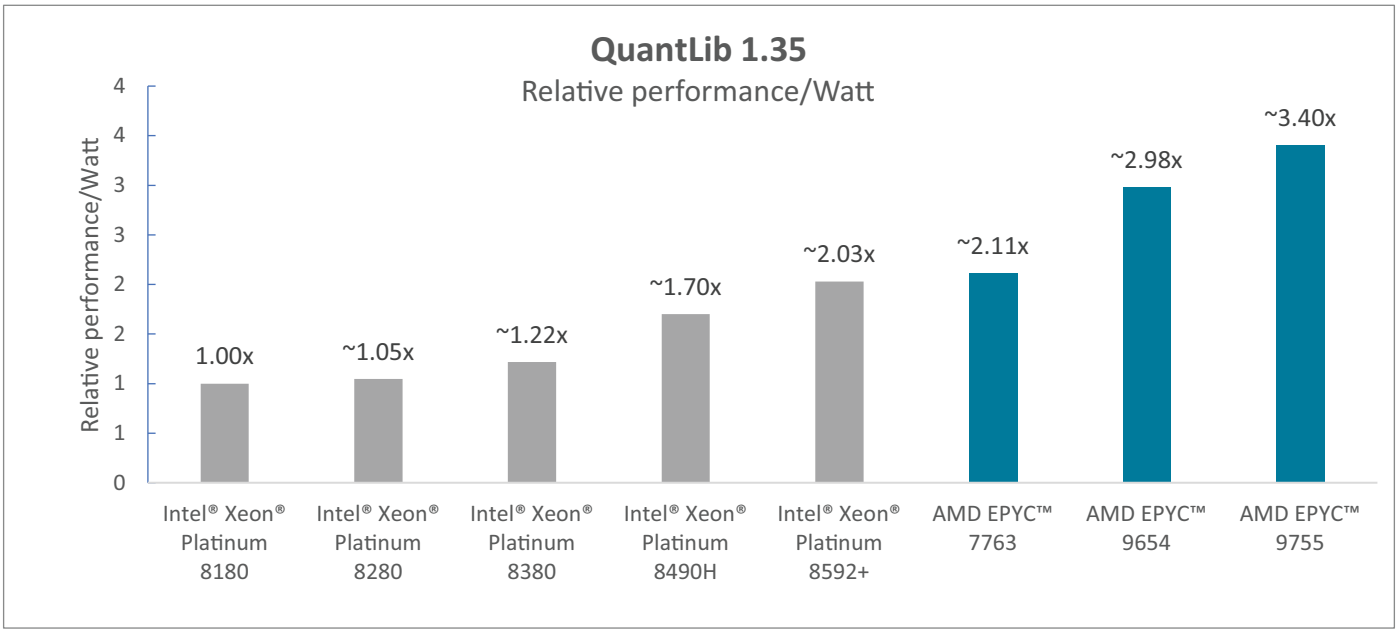



*Figure 9: Normalized relative QuantLib v1.35 performance per TDP Watt*

# CONCLUSIONS

The first part of this white paper described how the new QuantLib v1.35 benchmark builds on previous versions by more closely resembling an overnight risk calculation running within a financial institution. The performance scores generated by this version of the benchmark are thus better indicators of real-world system performance than prior versions. In other words, a high QuantLib v1.35 benchmark score carries more weight than a high score based on a previous benchmark version for the reasons described above.

The next section showed dual-socket systems powered by 4th Gen and 5th Gen AMD EPYC processors delivering compelling QuantLib v1.35 quantitative risk performance and performance per Watt. These uplifts make replacing legacy Intel Xeon systems with modern systems powered by AMD EPYC processors a compelling option for the financial services industry.

The findings collated in this paper demonstrate that AMD EPYC processors are well suited for risk analytics applications. Ongoing AMD innovations in CPU technology and contributions to the open-source ecosystem will continue optimizing existing and emerging workloads for financial industry developers and beyond.

## AMD EPYC 9005 PROCESSORS

5th Gen AMD EPYC processors are the newest generation of the powerful and efficient AMD EPYC processor family for servers that have set hundreds of world records for performance and efficiency. The AMD EPYC 9005 processor family is built on the breakthrough high performance, highly efficient "Zen 5" processor core architecture and advanced microprocessor process technologies to better meet the needs of the modern AI-enabled data center. The complete line of 5th Gen AMD EPYC processor offerings include a wide range of core counts (up to 192 cores and 384 threads per processor), max boost frequencies up to 5 GHz[2], generous L3 cache capacities, high energy efficiency, and competitive cost points. These cutting-edge technologies and features are all backed by the familiar x86 software compatibility that allows servers powered by AMD EPYC 9005 processors to readily support almost any business need.

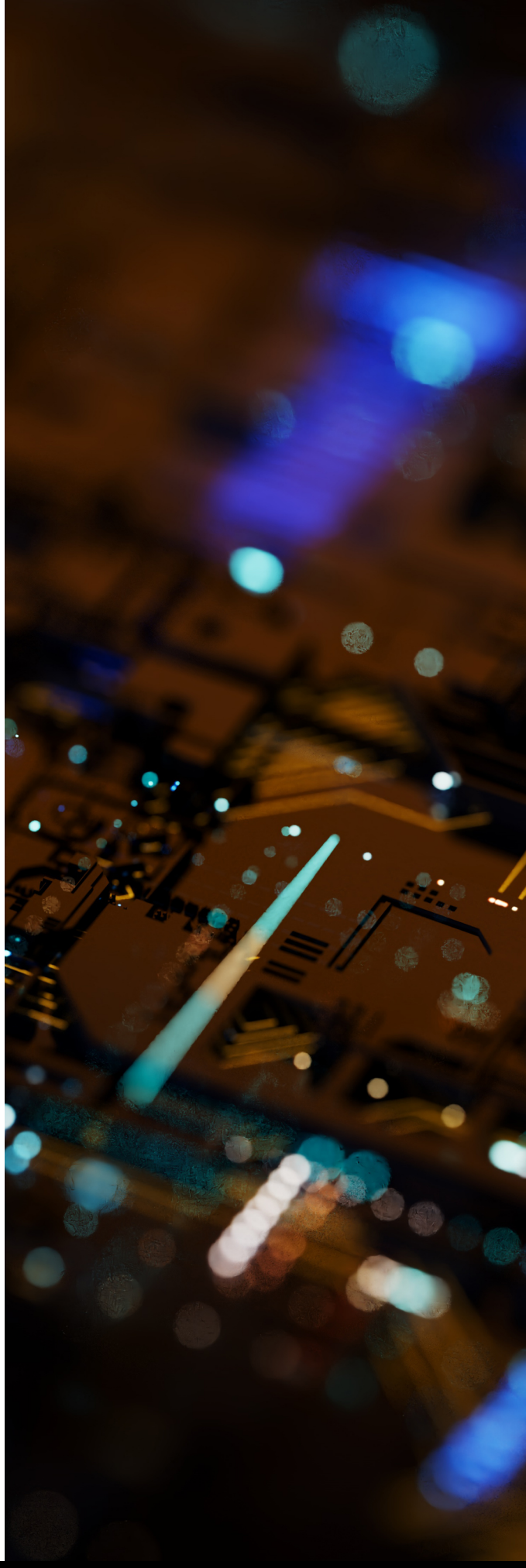
AMD EPYC 9005 Series Processors support up to:

- 192 physical cores, 384 threads
- Up to 512 MB of L3 cache per CPU
- 32 MB of L3 cache per CCD
- 9 TB of DDR5-6000 memory
- Up to 128 (1P) or 160 (2P) PCIe® Gen 5 lanes
- Infinity Guard security features[2]
  - Secure Boot
  - Encrypted memory with SME

# REFERENCES


1. **EPYC-18:** Maximum boost for AMD EPYC processors is the maximum frequency achievable by any single core on the processor under normal operating conditions for server systems.
2. **GD-183A:** AMD Infinity Guard features vary by EPYC™ Processor generations and/or series. Infinity Guard security features must be enabled by server OEMs and/or Cloud Service Providers to operate. Check with your OEM or provider to confirm support of these features. Learn more about Infinity Guard at https://www.amd.com/en/technologies/infinity-guard.

*Jacques du Toit is a PMTS Software System Design Engineer at AMD.*
*Sarina Sit is a Product Manager at AMD.*



## DISCLAIMERS

*Links to third party sites are provided for convenience and unless explicitly stated, AMD is not responsible for the contents of such linked sites and no endorsement is implied.

The information contained herein is for informational purposes only and is subject to change without notice. While every precaution has been taken in the preparation of this document, it may contain technical inaccuracies, omissions and typographical errors, and AMD is under no obligation to update or otherwise correct this information. Advanced Micro Devices, Inc. makes no representations or warranties with respect to the accuracy or completeness of the contents of this document, and assumes no liability of any kind, including the implied warranties of noninfringement, merchantability or fitness for particular purposes, with respect to the operation or use of AMD hardware, software or other products described herein. No license, including implied or arising by estoppel, to any intellectual property rights is granted by this document. Terms and limitations applicable to the purchase or use of AMD products are as set forth in a signed agreement between the parties or in AMD's Standard Terms and Conditions of Sale. GD-18u