

Special Analysis

Demystifying GPU Arithmetic

Mike Heroux and Bob Sorensen
September 2026

OVERVIEW

For most of the history of computing, "doing the math right" meant picking a single, fixed floating-point format and using it everywhere. Today, that assumption no longer holds. Modern GPUs and other AI accelerators support a growing menu of numerical formats – FP64, FP32, FP16, BF16, FP8, and even FP4 – and vendors increasingly market their hardware around how fast it runs each one.

For technical decision-makers who are not hands-on developers, understanding what these formats mean and what the differences imply for technology and business decisions matters more than ever. Arithmetic precision has become another major lever for performance, alongside clock speed and parallelism, and that lever is now something software and algorithms can tune deliberately rather than something fixed by hardware alone.

PRECISION AS AN ARCHITECTURAL TUNING KNOB

Processor performance has historically improved along a small number of major axes. Clock frequency was the primary lever for decades, but clock speeds largely plateaued 15 to 20 years ago. Parallelism became the next major lever and remains critically important – it is the reason GPUs, with their thousands of simple cores, became so valuable in the first place.

Arithmetic precision has emerged as a third major lever. Hardware designers have found that using fewer bits, when an application does not require full FP64 or FP32 precision, buys substantially more computational throughput, lower energy consumption, and reduced data movement. That trade-off is why formats such as FP16, BF16, FP8, and FP4 have become strategically important to GPU vendors, rather than a niche concern for a handful of specialists. The unprecedented scale of investment in AI infrastructure is a major driver of this shift, pushing the industry to pursue low-precision arithmetic wherever possible to keep performance climbing and hold down costs.

RANGE AND PRECISION ARE DIFFERENT THINGS

A useful way to make sense of the alphabet soup of formats – FP64, FP32, BF16, FP16, FP8, FP4 – is a simple analogy: a ruler versus a yardstick.

- **Range is like the length of the measuring device.** A ruler and a yardstick cover different spans of physical length. In a floating-point format, the exponent plays this role – more exponent bits mean the format can represent a wider span of very large and very small numbers.
- **Precision is like the hash marks on the device.** How finely spaced those marks are determines how precisely you can read a measurement. In a floating-point format, this is the job of the significand, or mantissa – more mantissa bits mean finer distinctions between nearby values.

These are independent dimensions. A format can be long (wide range) without being finely marked (high precision), just as a yardstick need not have millimeter markings. "More bits" is not automatically "better," how those bits are split between range and precision matters as much as the total count.

Table 1 shows how several common formats allocate bits between the exponent (range) and the mantissa (precision). In each of these formats, one additional bit is reserved for the sign.

TABLE 1

Floating-Point and Low-Precision Format Comparison

Format	Exponent Bits	Fraction Bits	Precision (Significant Digits)	Normal-Range Dynamic (orders of magnitude)	Comments
FP4	2	1	~0.6 digit	~1	In practice, deployed as blocks of values sharing a common exponent, which extends the effective range well beyond a single element's raw span
FP16	5	10	~3.3 digits	~9	Same total width as BF16, but allocates more bits to precision and fewer to range
BF16	8	7	~2.4 digits	~76	Matches FP32's exponent width, which makes it a natural storage format paired with FP32 compute
FP32	8	23	~7.2 digits	~76	Long the default format in general-purpose and scientific computing
FP64	11	52	~15.9 digits	~600+	Often unnecessary for AI inference and training; many AI accelerators provide limited or no native support, while HPC-oriented processors retain it

Note: Ranges reflect normal (non-subnormal) values only; including subnormal numbers extends each format's effective range somewhat further. "Fraction Bits" reflects the stored mantissa; normal values also carry an implicit leading bit, which is why, for example, FP32's 23 stored bits yield 24 bits of effective precision

Source: Hyperion Research, 2026

The FP16-versus-BF16 comparison is a good illustration of the range/precision trade-off in practice. FP16 devotes more bits to the mantissa, yielding higher precision but a narrower range. BF16 gives up some of that precision to keep roughly FP32's exponent range, which makes it convenient in mixed

BF16/FP32 environments where preserving FP32-like dynamic range matters more than fine-grained precision.

STORAGE, COMPUTE, AND ACCUMULATOR: ONE CALCULATION, THREE PRECISIONS

A second source of confusion is the assumption that a single calculation has one precision. In practice, there are at least three distinct roles, and they need not use the same format:

- Storage format – how values sit in memory and move across the system. Compact formats reduce memory footprint and bandwidth demand.
- Compute format – the representation actually used when performing the arithmetic. It need not match the storage format; for example, a value stored in FP4 may be paired with a higher-precision scale factor and transformed before it enters the arithmetic unit.
- Accumulator format – the representation used to hold running sums and reductions. It is often given more range or precision than the operands themselves, because many small errors can accumulate across a long sum.

A vendor can accurately describe a processor as "doing FP4 computation" even though part of the arithmetic pipeline is running at higher precision. The claim is about the compute format, not the whole calculation. Common combinations illustrate the pattern:

- FP32 storage with FP32 compute and FP32 accumulation is the fastest, simplest option on much hardware
- FP16 storage with FP32 accumulation is common in memory-bound GPU workloads, trading storage compactness for a more stable sum
- BF16 storage with FP32 compute and accumulation is a typical pattern on TPUs and other accelerators that store compactly but compute and accumulate at higher precision for stability

OZAKI SCHEMES: PRECISION AS AN ALGORITHM, NOT JUST A FORMAT

The formats above are hardware-defined choices. Another family of approaches, often called Ozaki schemes, treats precision as a tunable algorithmic parameter rather than a fixed hardware format.

The core idea is decomposition. Instead of computing a matrix product directly in FP64, an Ozaki-style method splits each FP64 matrix into several lower-precision – often integer – components. It then computes a collection of products between those low-precision pieces, using fast integer or low-precision matrix hardware, and combines the partial results with appropriate scaling to reconstruct a high-accuracy answer. The end user can still supply FP64 inputs and receive an FP64 result; the decomposition and reconstruction happen internally.

This approach is particularly attractive for dense matrix-matrix multiplication, where the cost of decomposing the input matrices is amortized over a much larger volume of arithmetic – dense matrix multiplication performs on the order of N^3 arithmetic operations on only N^2 data, so the one-time cost of decomposition is spread across a great deal of computation. That same economic argument does not hold for lower-intensity operations such as dot products, or even dense matrix-vector multiplication, where there is little arithmetic to amortize the decomposition cost against; Ozaki techniques offer no real speed or cost advantage there. One notable side benefit, however, is that

because the underlying arithmetic is performed using integers, Ozaki-based methods can produce bitwise-reproducible parallel sums, a property that is difficult to achieve with ordinary floating-point summation, where the result can depend on the order in which values are added.

The strategic implication is that Ozaki-style techniques turn precision into a continuously adjustable algorithmic choice. Depending on how a computation is structured, the same underlying low-precision hardware can be used to emulate FP64 accuracy, to deliberately trade some accuracy for speed, or to exceed FP64 accuracy where an application demands it. This flexibility also makes Ozaki schemes attractive for fine-tuning precision to a specific application's needs: the same decomposition approach can emulate accuracy levels between FP32 and FP64, or even precision beyond what FP64 itself provides, without requiring dedicated hardware for each intermediate level.

ANALYST COMMENT

Arithmetic precision has joined parallelism and data movement as a first-class architectural and algorithmic design dimension. A vendor's headline "FP64 teraflops" or "FP4 petaflops" number increasingly needs context: what is stored, what format the arithmetic actually runs in, how results are accumulated, and how much data has to move to support the calculation. A GPU's raw peak numbers for one format say relatively little about performance on a workload that spends most of its time in a different format or that is limited by memory bandwidth rather than arithmetic throughput in the first place.

This context is useful for interpreting a notable trend in recent GPU designs: reduced peak FP64 throughput compared with prior generations. Taken alone, that could look like a retreat from scientific computing. But it is better read as an acknowledgment of two things discussed above: most scientific applications are limited by memory system performance rather than by the number of FP64 arithmetic units available, and Ozaki-style techniques now offer a credible path to emulating FP64 accuracy using much cheaper low-precision hardware for some applications. Scaling back dedicated FP64 arithmetic capacity, in that light, can be a reasonable architectural trade rather than an abandonment of double precision.

A related and more provocative argument comes from Dongarra, Hoefler, and Matsuoka's "Do We Still Need GPUs?", which contends that CPUs incorporating matrix engines, high-bandwidth memory, and native mixed-precision support – down to FP4 – may in some cases rival GPUs for AI and scientific workloads, precisely because precision and memory bandwidth, not raw FP64 flops, are increasingly the variables that matter.

Going forward, the HPC-AI community can expect mixed precision to be the norm rather than the exception. Most practical AI pipelines already blend formats – low precision for bulk computation, higher precision for accumulation or sensitive layers – and that pattern is likely to extend further as FP4 matures. Traditional HPC applications, while still typically anchored by FP64, are also exploring and transitioning to mixed precision to take advantage of performance and storage benefits when used carefully.

About Hyperion Research, LLC

Hyperion Research provides data-driven research, analysis and recommendations for technologies, applications, and markets in high performance computing and emerging technology areas to help organizations worldwide make effective decisions and seize growth opportunities. Research includes market sizing and forecasting, share tracking, segmentation, technology, and related trend analysis, and both user and vendor analysis for multi-user technical server technology used for HPC, AI, quantum computing, and cloud computing. Hyperion Research provides thought leadership and practical guidance for users, vendors, and other members of the advanced computing community by focusing on key market and technology trends across government, industry, commerce, and academia.

Headquarters

365 Summit Avenue

St. Paul, MN 55102

USA

612.812.5798

www.HyperionResearch.com and www.hpcuserforum.com

Copyright Notice

Copyright 2026 Hyperion Research LLC. Reproduction is forbidden unless authorized. All rights reserved. Visit www.HyperionResearch.com to learn more. Please contact 612.812.5798 and/or email info@hyperionres.com for information on reprints, additional copies, web rights, or quoting permission.