Back to Bits

Extending Shannon's Framework for Communication Performance to Computation

Max Hawkins and Rich Vuduc

Computing Export Controls

"Who controls the ~~spice~~ compute, controls the world"

REPORTS *Mar 24, 2025* | Hudson Institute

AI, National Security, and the Global Technology Race: How US Export Controls Define the Future of Innovation

Nury Turkel

Exclusive: Nvidia modifies H20 chip for China to overcome US export controls, sources say

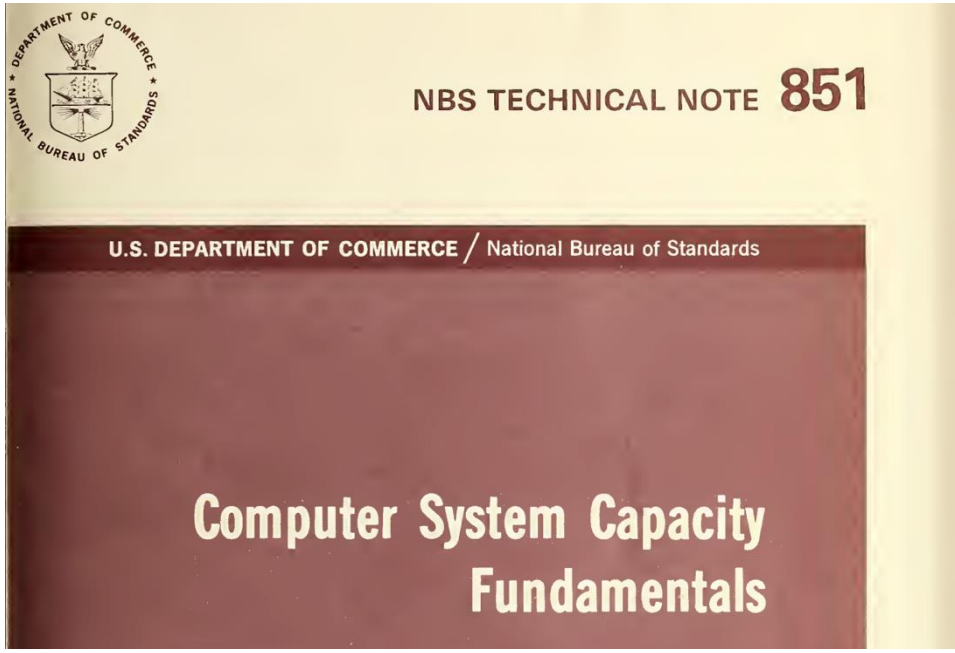
By **Liam Mo** and **Brenda Goh**

May 9, 2025 5:21 AM EDT · Updated May 9, 2025

Definitions of computing performance impact
billion-dollar decisions,
national security,
and the future of computing.

How to do this accurately, fairly, and generally?

What are flop/s?



- Floating-point operations per second
  - Historically multiplies and adds in IEEE-754 FP64
- Originated from US Dept. of Commerce
  - D. J. Kuck Oct. 1974
- 50 years later, it's still dominant
- Math matters - not pointer indexing or memory fetches

"People have often quoted computer speeds in …millions of instructions per second. But the execution of an 'instruction' yields rather different effects on various machines… Thus, megaflops…may be the important measure."

Which computer is more performant?

By how much?

Computer A:
1.44 Exaflop/s

Computer B:
1.35 Exaflop/s

Which computer is more performant?

By how much?

Computer A

- 1.44 Exaflop/s*
- Nvidia's GB200 NVL72
  - "The NVIDIA GB200 NVL72 is an exascale computer in a single rack." - https://www.nvidia.com/en-us/data-center/gb200-nvl72/

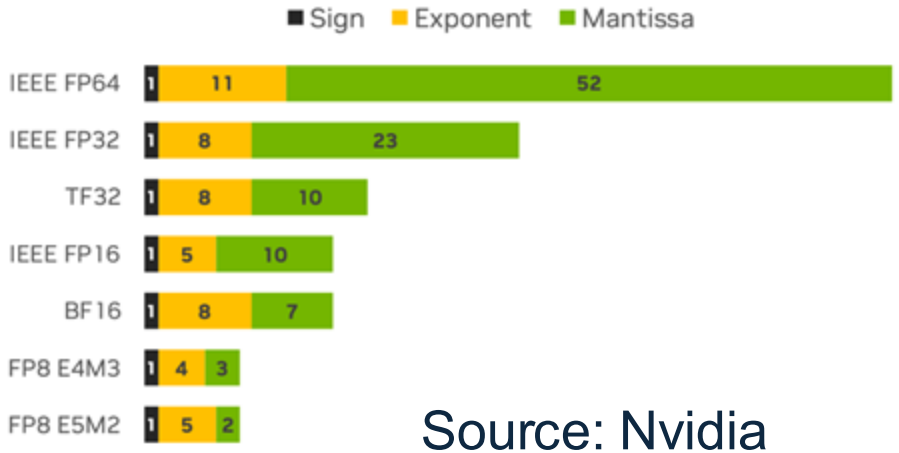
Computer B

- 1.35 Exaflop/s**
- Frontier
  - #2 most performant public supercomputer

*** With sparse FP4 tensor cores**

**** Dense FP64 with Linpack**

The Variety of Computing



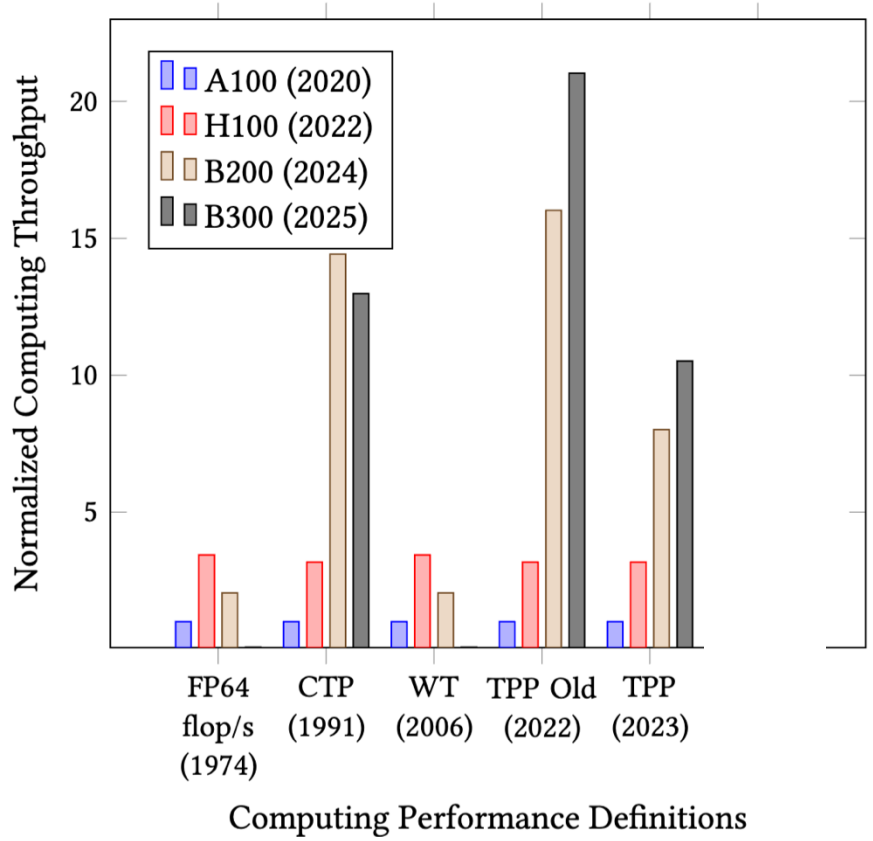
Source: Nvidia

- Data types
  - Bit widths
  - Bit allocation (e.g. mantissa and exponent bits)
  - Encoding schemes - integer vs floats vs posits…
  - Specifications (e.g. IEEE, OCP, vendor…)
- Operations
  - Add, subtract, negate, multiply, divide, compare, sqrt, tanh, …
  - Sparsity
  - Emulation (Ozaki and beyond)
  - Noise
  - Scalar vs vector vs matrix inputs

Quantum, analog, neuromorphic, reversible, …

How do we fairly measure and compare performance across this large design space?

The flop is dead. Long live the flop.



- US Dept. of Commerce hasn't used 'basic' flop/s in decades!
  - They need an accurate and generalizable computing performance metric
  - Applying changing, ad-hoc corrections to flop/s
- 1991 – Composite Theoretical Performance (CTP)
  - Scaling based on operand bit width: Scale = 1/3 + width / 96
- 2006 – Weighted TeraFLOP (WT)
  - Only 64 bit or larger math counts
  - Scaling based on vector (0.9) or non-vector (0.3)
- 2022 – Total Processing Performance (TPP)
  - Scaled based on max bit width of inputs and outputs
- 2023 to Current – Updated TPP
  - Like above but restricted to inputs only

Same Sh…tory, Different Day

flops

"People have often quoted computer speeds in …~~millions of instructions~~ per second.

flop

But the execution of a~~n 'instruction'~~ yields rather different effects on various machines…

???

Thus, ~~megaflops~~…may be the important measure."

These changing metrics reflect the **difficulty** in evaluating computing performance **without a principled foundation**.

Our Proposal: Back to Bits

Inspiration from Communication

"The fundamental problem of communication is that of reproducing at one point either exactly or approximately a message selected at another point”

– *A Mathematical Theory of Communication (Shannon 1948)*

- Established the field of information theory
- Generalized the concept of communication performance
  - Allowed for fair and generalized performance evaluation

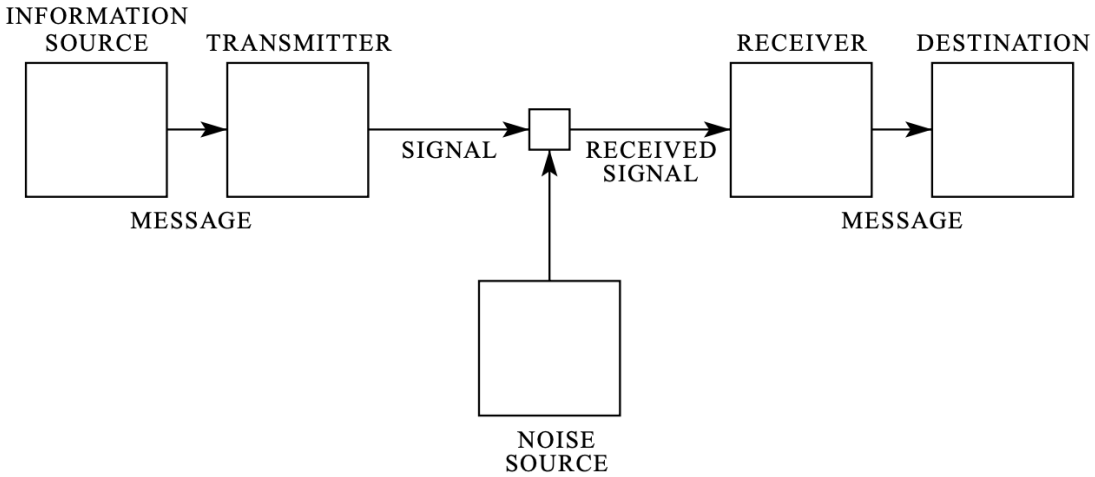
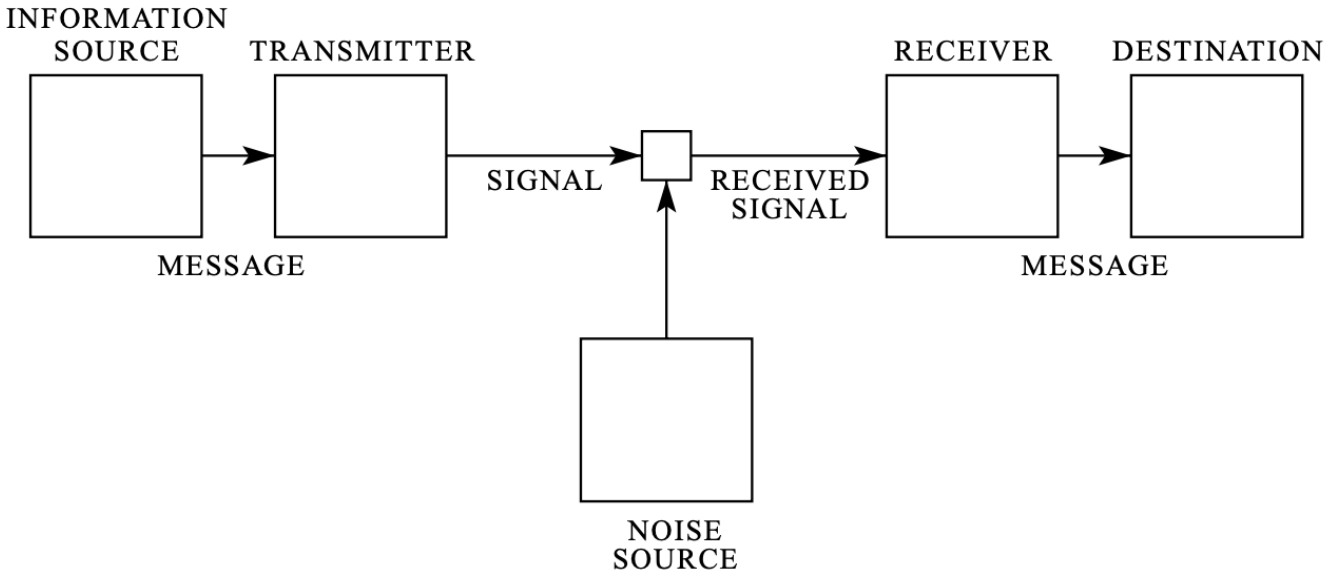
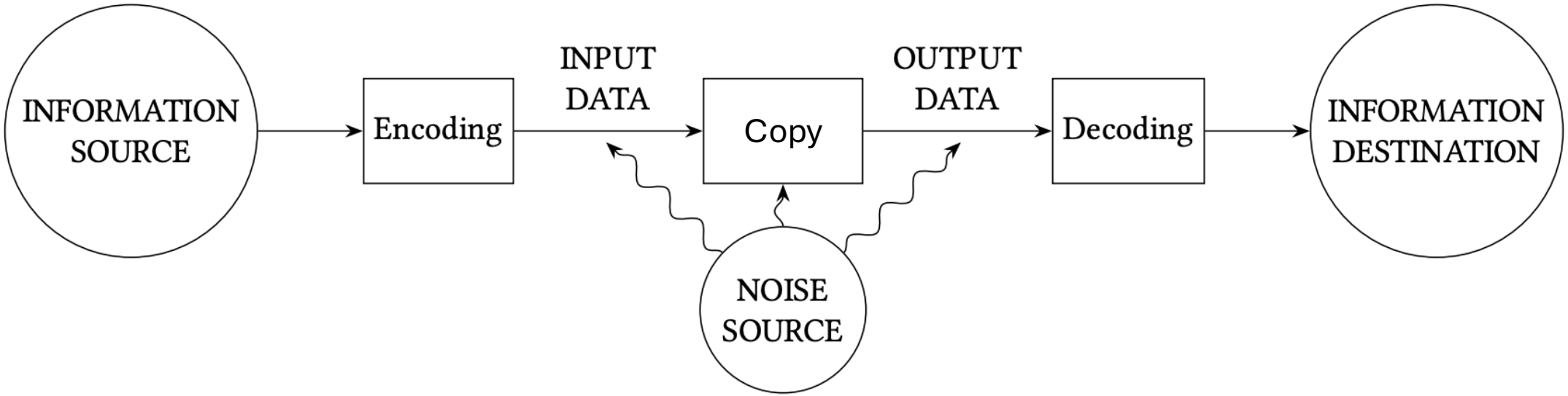


Fig. 1—Schematic diagram of a general communication system.

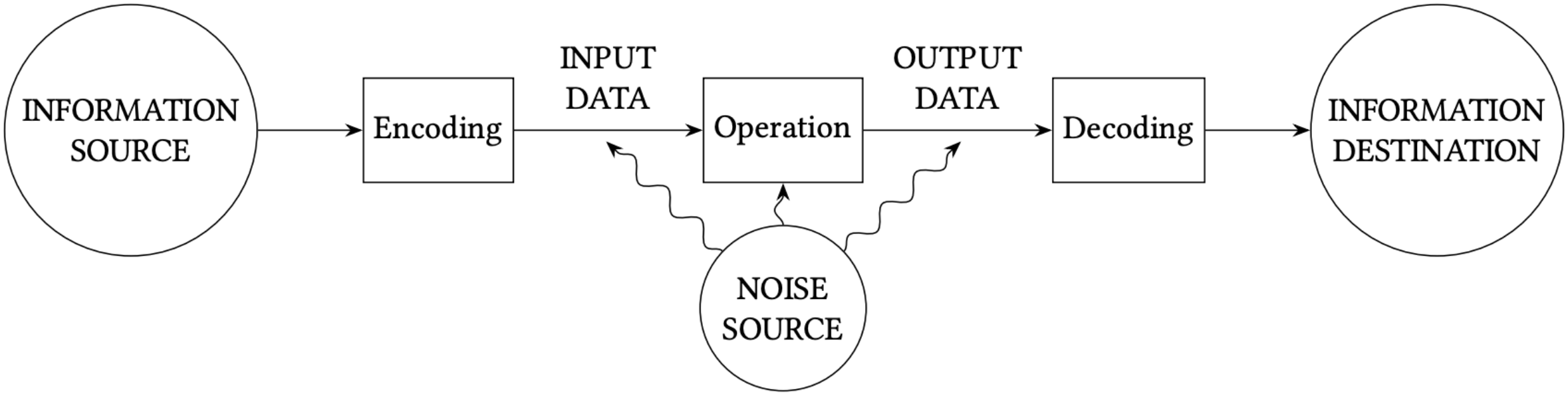
Can we use information theory for computation performance evaluation?

Communication: Copies inputs to outputs (identity operation)





Expand beyond the identity: Computation



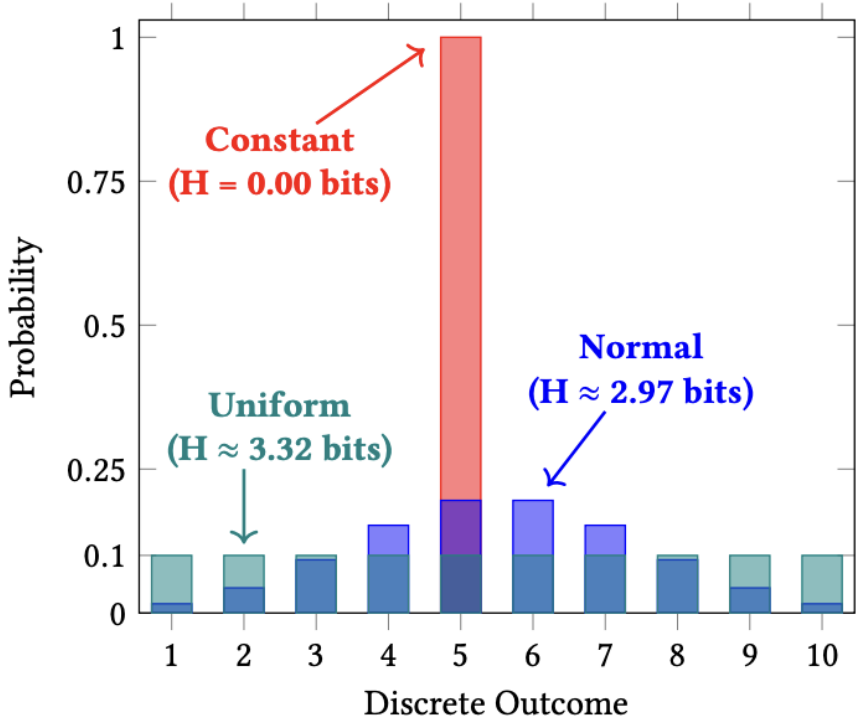
A Short Intro to Information Theory

- Uncertainty
- Flipping a coin: Heads or Tails
- Rolling a die: 1,2,3,4,5,6 (or generally M faces)

How much uncertainty does each act have?

A Short Intro to Information Theory

- Uncertainty
- Flipping a coin: Heads or Tails
- Rolling a die with M faces



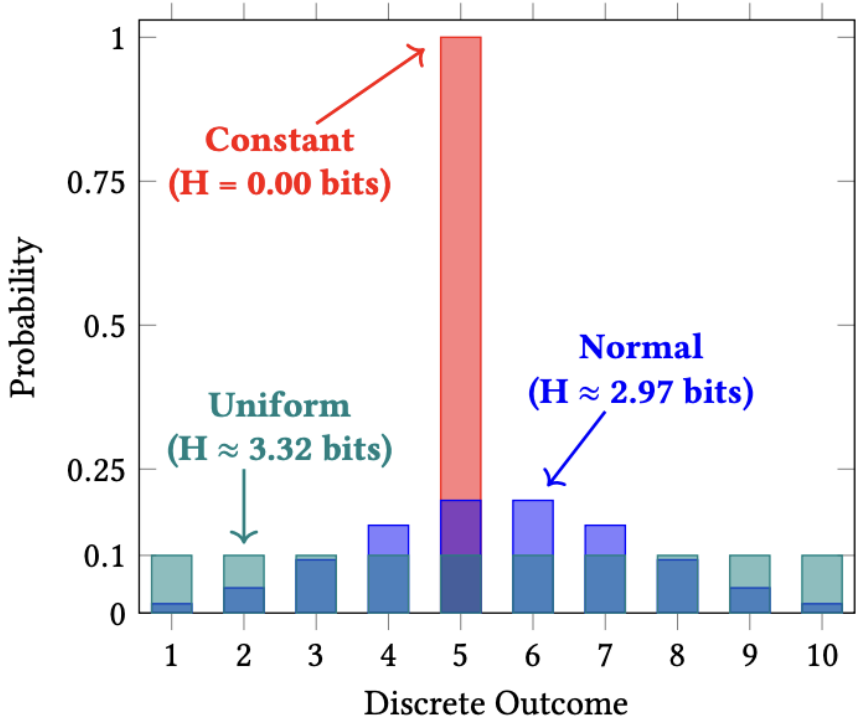
- Shannon entropy (H): A measure of uncertainty
  - Discrete random variable X with probability distribution p(x)
  - Measured in bits if b = 2

$$\mathrm{H}(X) = -\sum_{x \in \mathcal{X}} p(x) \log_b p(x)$$

A Short Intro to Information Theory

- Uncertainty
- Flipping a coin: Heads or Tails ➔ 1 bit
- Rolling a die with M faces ➔ $log_2(M)$

- Shannon entropy (H): A measure of uncertainty
  - Discrete random variable X with probability distribution p(x)
  - Measured in bits if b = 2

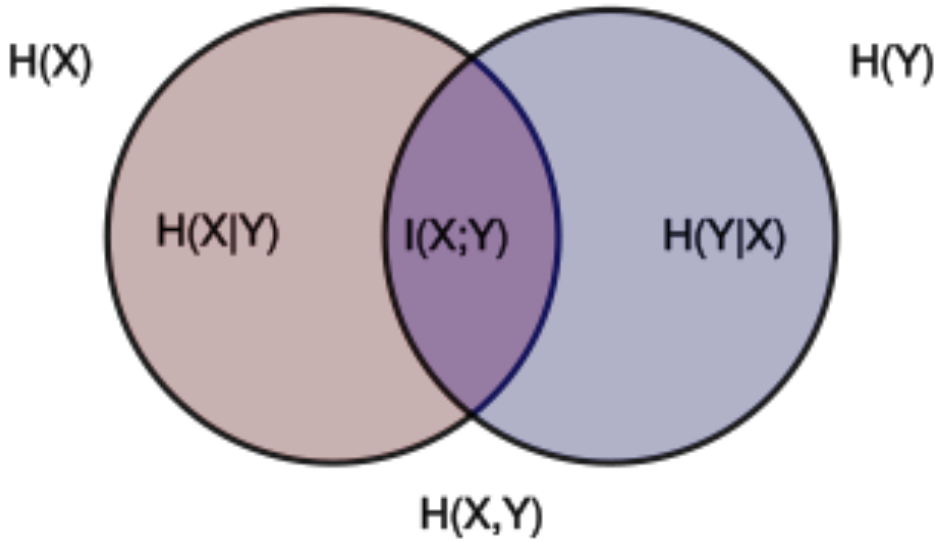


$$\mathrm{H}(X) = -\sum_{x \in \mathcal{X}} p(x) \log_b p(x)$$

Mutual Information (I)

- Measures the ‘shared’ information of random variables
  - Considers properties of the operation and noise
- ‘Operational’ quantity
- Application and runtime-dependent

$$I(X;Y) = H(X) - H(X|Y)$$



Channel Capacity (C)

- Maximum MI over possible distributions
- Upper bound/ideal quantity
- Application-agnostic

$$C = \max_{p(x)} I(X;Y)$$

Summary of Our Proposal

- Expand Shannon's communication performance model
  - Identity operator (communication) → Arbitrary operator (computation)
  - Mutual info and channel capacity naturally handle this
- Flop/s → Bit/s
  - Base Measure: Information in bits
  - Operational: Mutual Information
  - Ideal/Peak: Channel capacity
- Enable generalized and fair performance evaluation
  - Just like communication has had for >70 years

It's what we already implicitly do!

How do you weight across bit widths?

- Example from before:
  - 1.35 Exaflop/s at FP64
  - 1.44 Exaflop/s at FP4
- Bit width multiplier?
  - FP64 = FP4 * X
  - What is your X?

Using Logarithms is Natural

- Implicitly what most HPC users do currently
- $FP64 = \frac{64}{4} * FP4 = 16 * FP4$
- Example from before:
  - 1.35 Exaflop/s at FP64 ➔ 86.4 Exabits/s
  - 1.44 Exaflop/s at FP4 ➔ 5.76 Exabits/s
  - *Note: Sparsity and operation still need to be accounted for*
- The absurdity of linear scaling:
  - $\frac{2^{64}}{2^4} = 1{,}152{,}921{,}504{,}606{,}846{,}976$

Info Theory Already in Use: Redundant Encodings

- IEEE-754 redundant NaN encodings
  - FP64 has roughly $9 * 10^{15}$bitstrings encoding NaNs
  - Yet a ‘canonical’ NaN often exists in languages!
- Redundancy reduces information capturing potential
  - Magnified by smaller number of bitstrings – low bit widths
  - Quantified with encoding efficiency: $\eta = \frac{H_{encoding}}{bit\_width}$
- Modern data types are removing this redundancy – even IEEE
  - OCP, hardware vendor data types, posits, and IEEE P3109

Modern data type design already considers information theory.

Redundant NaN Encodings in Practice

Float Type	Bit Width	Entropy	Encoding Efficiency (%)
IEEE-754	64	63.974	99.960
IEEE-754	32	31.906	99.707
TF32	19	18.957	99.774
IEEE-754	16	15.657	97.854
BF16	16	15.969	99.806
OCP (E4M3)	8	7.992	99.902
IEEE-754 (E4M5)*	8	7.792	97.397
OCP (E2M3)	6	6	100
IEEE-754 (E2M3)*	6	5.167	86.119
IEEE-P3109	Any	Ideal	100

** Theoretical – Does not exist.*

Info Theory Already in Use: John Gustafson's Rules for Data Formats

- John Gustafson
  - Creator of Gustafson's Law, unums, and posits
  - A leading expert in data format design
  - Dr. Srinivas Aluru's PhD advisor

<u>Gustafson's Top-Two Rules:</u>

1. "The distribution of the values represented should resemble the distribution of numbers needed in calculations…this is the principle of 'maximum entropy'"
2. "There should be no redundant bit patterns to mean the same thing; every bit counts."

(Source: Page 97 of "Every Bit Counts: Posit Computing")

Aside: I'll be giving a talk on this at the Conference on Next Generation Arithmetic in November.

Bit Widths

- Provides a theoretical foundation for our implicit comparison
- Simple scaling factor: $log_2(num\_unique\_values)$
  - With efficient data types, approximates bit width
- Mutual Information Constraint
  - Maximum of sum of input bit widths or sum of output bit widths
  - Essentially what TPP export control already does

$$I_{\max} \approx \min\left(\sum_i bw_{\text{in},i},\ \sum_j bw_{\text{out},j}\right)$$

Shannon's Framework Beyond the Identity Operator

- Mutual info aligns with our intuition
- Bit width multiplier
- 'Work'/utility of $f(x) = y$
  - x is known to be constant
  - y is constant regardless of x

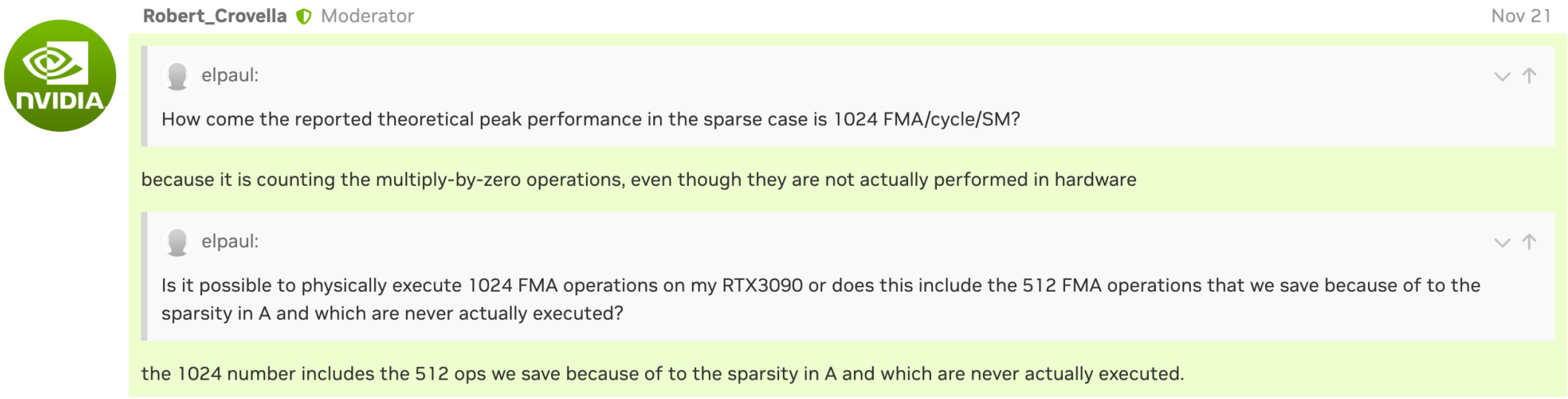
  } Optimized away by compilers $H(x) = 0 \rightarrow I = 0$, $H(y) = 0 \rightarrow I = 0$

  - y is random regardless of x

  } Useless $H(y|x) = H(y) \rightarrow I = 0$

Sparsity

- A known value carries 0 information
  - Don’t count operations on known zeroes!
  - Just like standard HPC practice (and HPCG)
- HW vendor counting these when reporting sparse performance:



Robert_Crovella Moderator — Nov 21

> elpaul:
>
> How come the reported theoretical peak performance in the sparse case is 1024 FMA/cycle/SM?

because it is counting the multiply-by-zero operations, even though they are not actually performed in hardware

> elpaul:
>
> Is it possible to physically execute 1024 FMA operations on my RTX3090 or does this include the 512 FMA operations that we save because of to the sparsity in A and which are never actually executed?

the 1024 number includes the 512 ops we save because of to the sparsity in A and which are never actually executed.

Source: https://forums.developer.nvidia.com/t/dense-vs-sparse-tensor-core-performance-fp16/314261

Noise

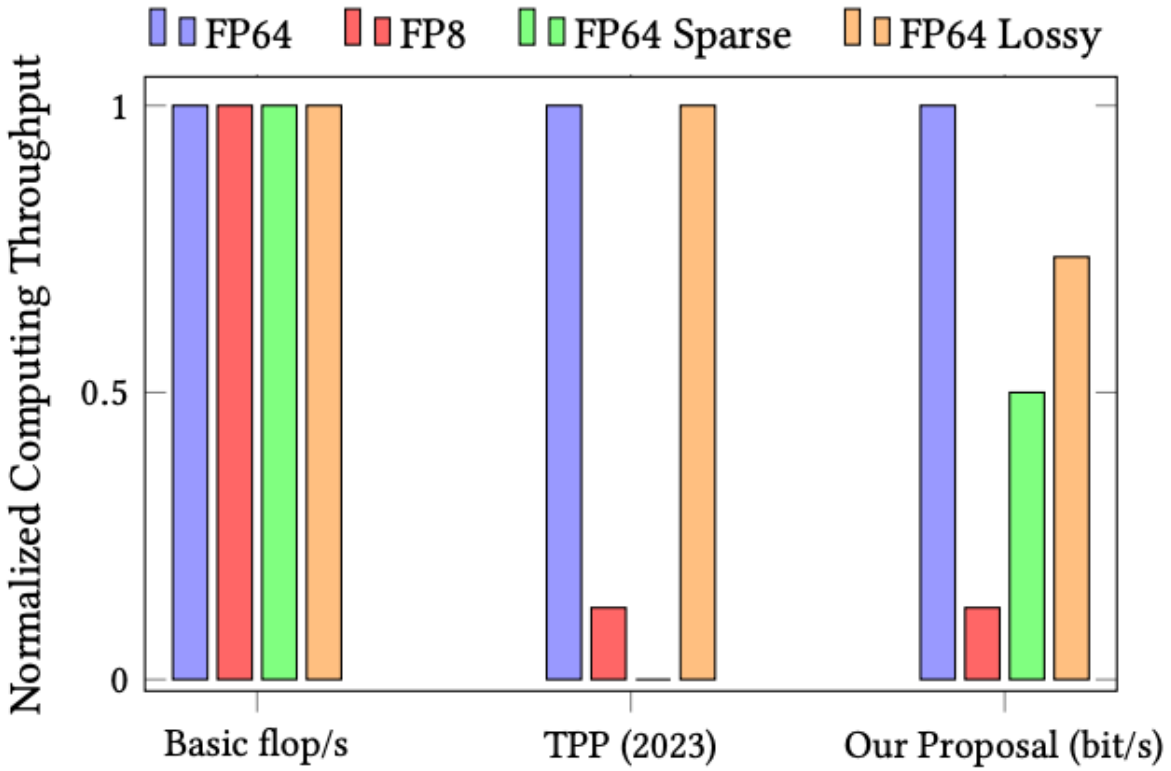
- Errors in digital computing are rare
  - Not considered in current performance metrics
- This assumption may not hold
  - Probabilistic, analog, quantum, …
- Mutual information naturally accounts for noise
  - Communication errors abound!

Operations

- Hardware specialization
  - ALUs don’t drive peak GPU performance anymore
  - Vector/matrix pipelines for specific ops (e.g. matmul/tensor cores)
- Not all operations are created equal
  - Compare vs negate vs XOR vs add vs multiply vs sqrt…
- Why restrict the subset of operations evaluated?

Providing a Needed Framework

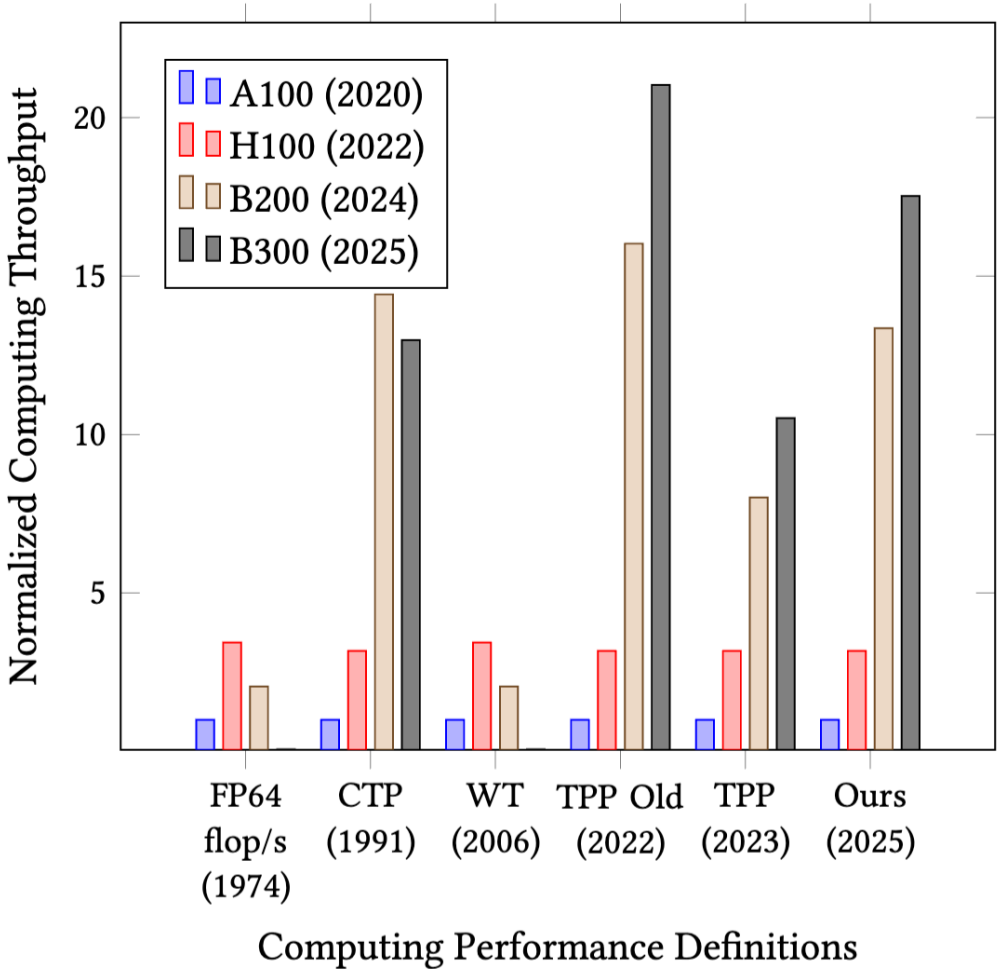
- At worst, this framework grounds existing heuristics
  - US export controls (TPP)
  - Bit-width multiplier (FP64 vs FP4)
  - Redundant value encodings
- “We measure computing performance this way because…”
  - “…it’s what we do.”
  - “…of this theoretical framework.”
- Can adapt to future hardware



Flop/s flattens performance to create false equivalence.
Measuring computing with information throughput (bit/s) captures the true diversity.

Providing a Needed Framework

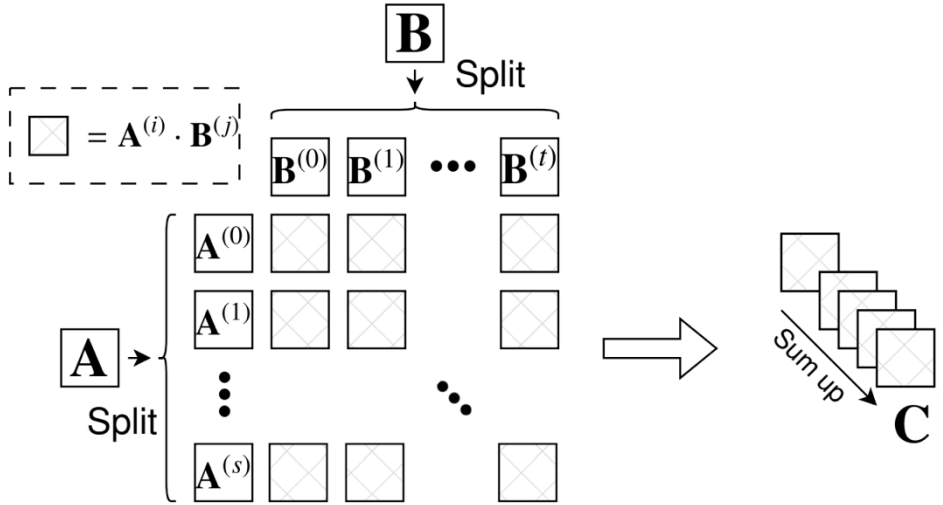
- At worst, this framework grounds existing heuristics
  - US export controls (TPP)
  - Bit-width multiplier (FP64 vs FP4)
  - Redundant value encodings
- “We measure computing performance this way because…”
  - “…it’s what we do.”
  - “…of this theoretical framework.”
- Can adapt to future hardware

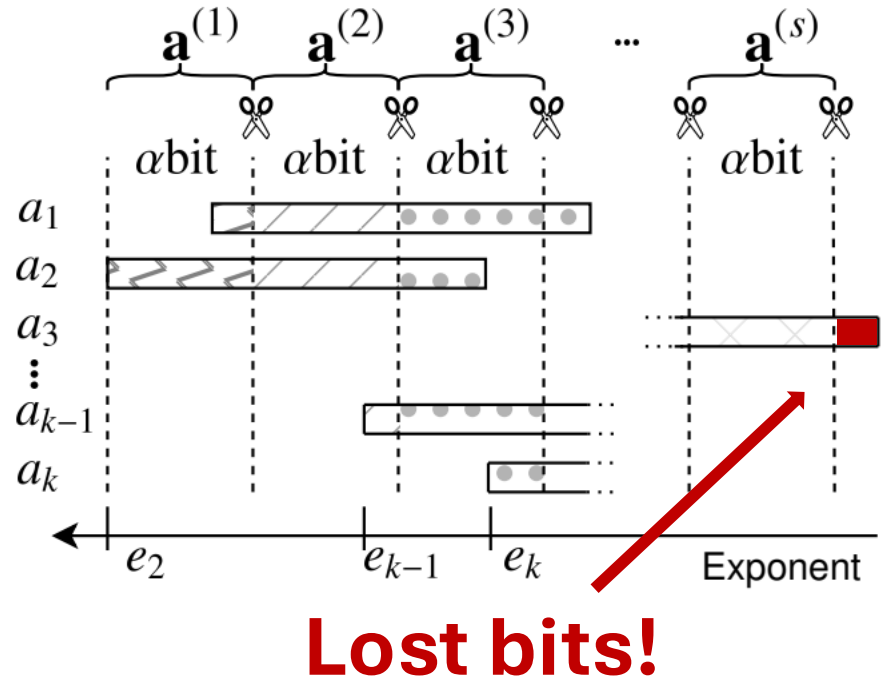


Flop/s flattens performance to create false equivalence.
Measuring computing with information throughput (bit/s) captures the true diversity.

Emulation: Ozaki Scheme

- FP64 matrix multiplication emulation
  - Using lower-precision formats and math
  - Int8 but also FP16, FP8, FP6, …
- Number of slices
  - Configurable parameter
  - Affects performance and accuracy
- Mantissa Space
  - Bits = Slices * Bits_per_slice
  - Bits per slice: Int8=7, FP16=11
- Data-dependent error!
  - Ozaki assumes a small input magnitude range





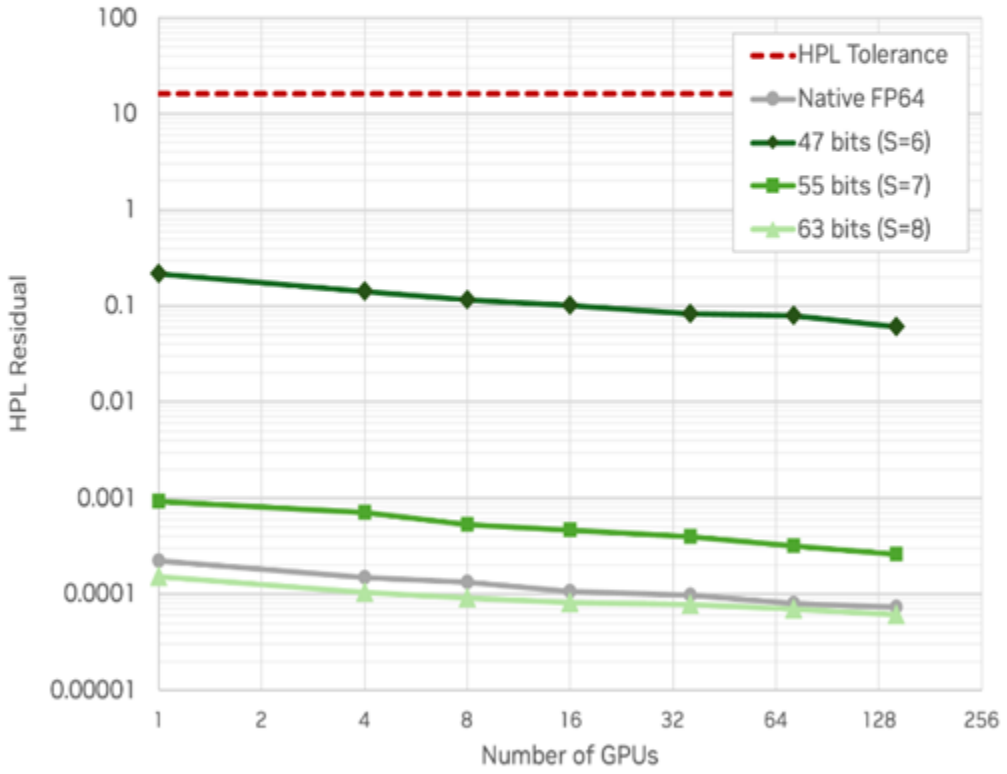
Lost bits!

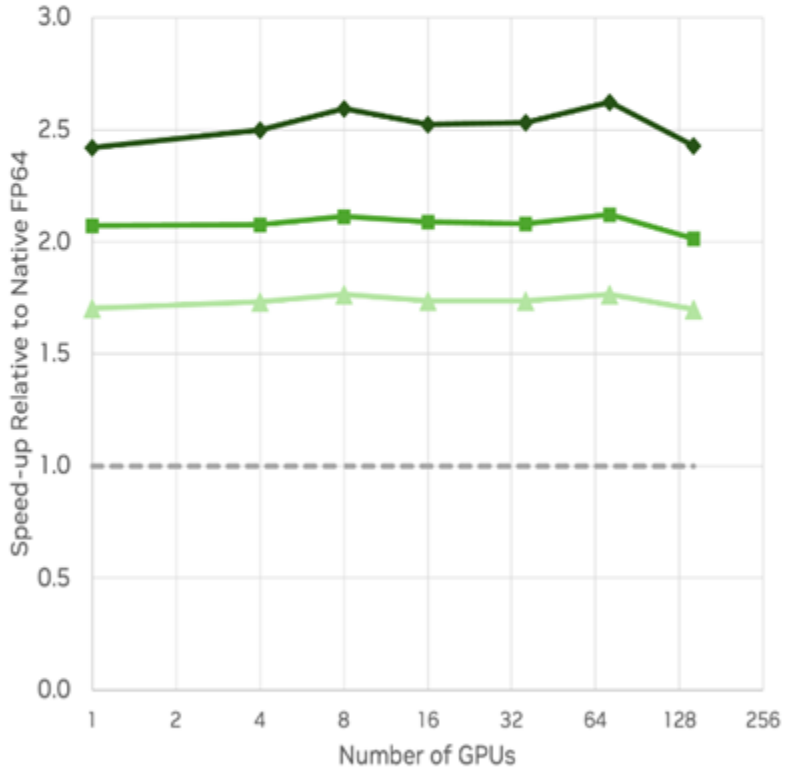
Source: https://arxiv.org/pdf/2306.11975

HPL with Double Precision Floating Point Emulation

Accuracy & Performance on GB200 NVL72

More slices decreases error.... **But also performance**





HPL checks if *Residual* < *Threshold*, where:

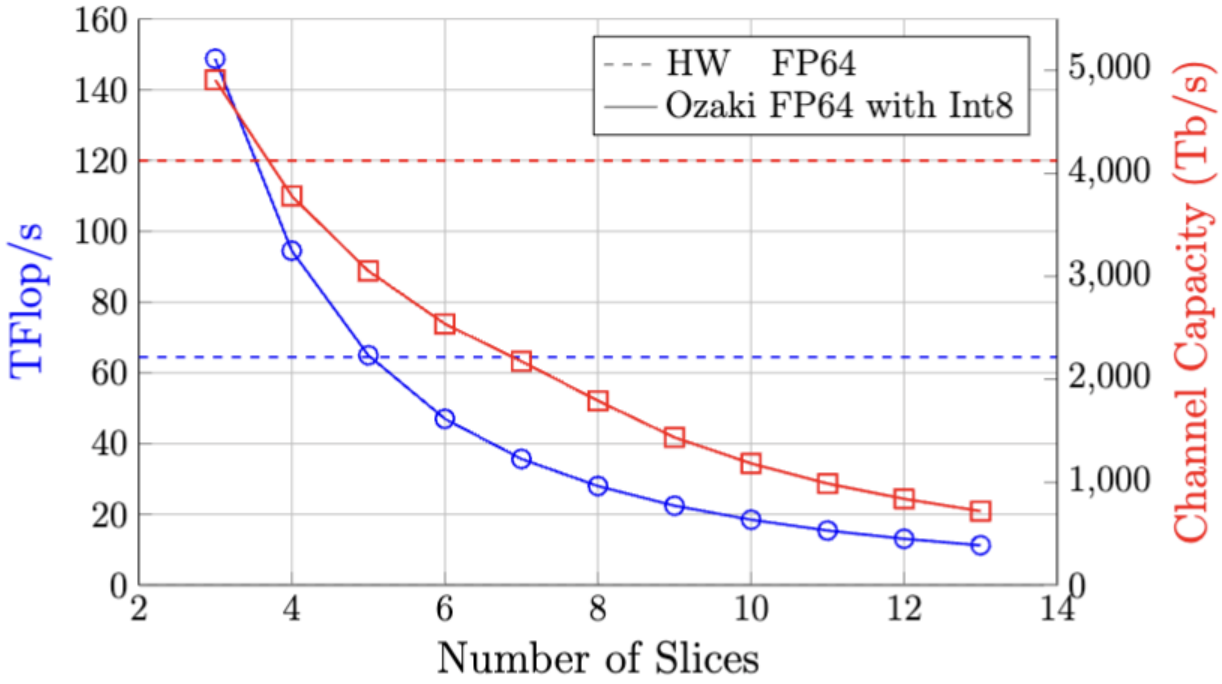
Residual=||*Ax-b*||_∞/(*eps**(||*A*||_∞*||*x*||_∞+||*b*||_∞)**N*)

Threshold = *16.0* (default value)

Source: Nvidia

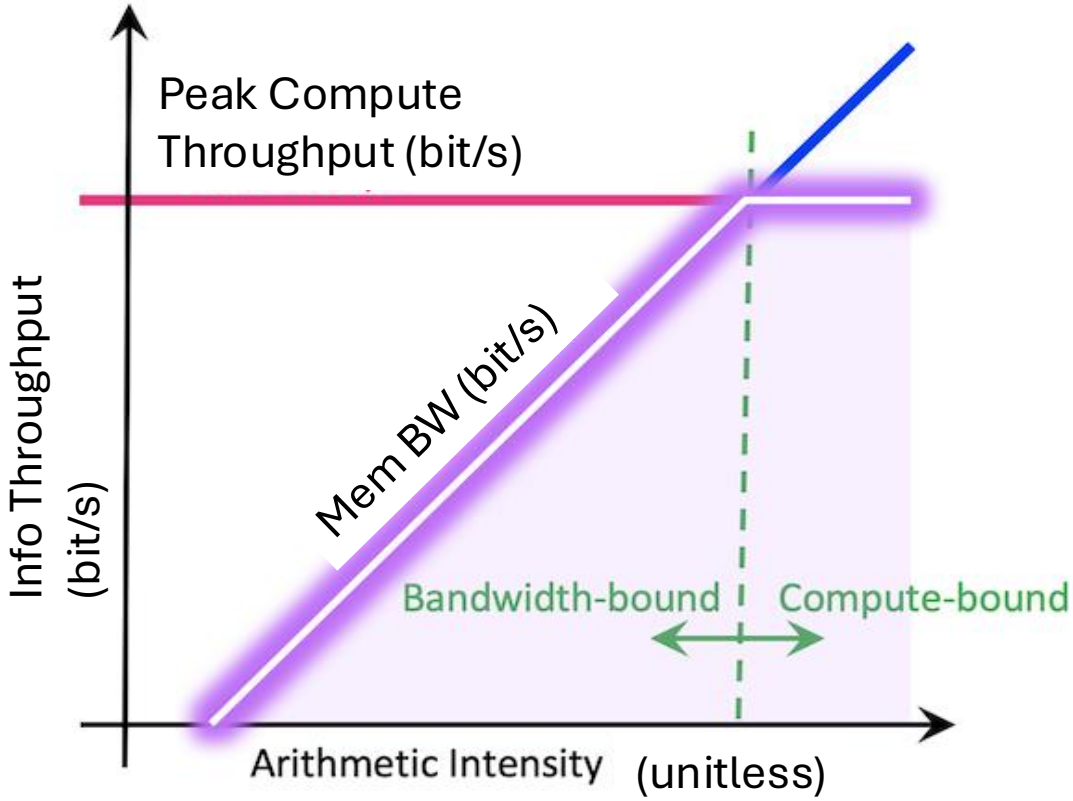
Ozaki Scheme Encourages Our Framework

- “Bit-width multiplier” isn’t so simple anymore
  - Configuration space is even larger
  - Low-precision data type and number of slices
- Calculating state space of Ozaki implementation:
  - Bits/op = bits_sign + bits_exp + bits_mantissa_space
  - OzIMMU: bits/op ~= 1 + 11 + min(52, 7 * num_slices)
- Experimental results:
  - Running CuBLAS dense GEMM on GH200
  - HW FP64 vs Ozaki-emulated with Int8



Communication and Computation

- Communication as a subset of computation
- Potentially unifies analyses
  - Critical for von Neumann architectures
  - Exascale vs exaflop discussions
- Unitless arithmetic intensity
  - $\frac{bits_{computation}}{bits_{communication}}$
- Information Roofline Analysis
  - Incorporates optimization of data types
  - Useful for emulation: HW pipeline used != Data type
  - A timely tool for HPC and AI application developers



The Importance of Agnosticism

- Hardware agnosticism
  - Assumptions don’t hold across hardware regimes/models
- Application agnosticism
  - My application isn’t your application
  - Accuracy/error analysis isn’t general

We will always need a hardware and application-agnostic computing performance metric.

A Mathematical Theory of Communication - Shannon 1948

"The fundamental problem of communication is that of reproducing at one point either exactly or approximately a message selected at another point.

Frequently the messages have meaning;

that is they refer to or are correlated according to some system with certain physical or conceptual entities.

These **semantic aspects of communication are irrelevant to the engineering problem**.

The significant aspect is that the actual message is one selected from a set of possible messages.

The system must be designed to operate for each possible selection, not just the one which will actually be chosen since this is unknown at the time of design."

Future Work

- Extend Ozaki scheme to highlight its approach
- Vector/matrix pipeline considerations
- Quantum, analog, neuromorphic, reversible, etc
- Application/runtime data type optimization
  - Increasingly important research area
  - Can I use FP32? FP16? Block-scaled FP8? Posits?
  - “Every model is wrong. Some are useful”
    - e.g. Discrete samples of continuous, real values

How will we fairly and generally measure computing performance in the future?

- Innovation will continue
  - Loss of Moore's law/Dennard scaling
- Data type and hardware variety will grow
- How many asterisks is too many?
  - *Flop/s in FP32, with sparsity, using tensor cores, emulated with Ozaki scheme 1 using two slices of Int8
- Export controls need a fundamental grounding
- HPC + Quantum systems

Information Theory Enables a Useful and General Framework for Computing Performance

"Let's get **Back to Bits**"

4-page preprint:
arxiv.org/pdf/2508.05621

Thank you to everyone in CSE who has helped!