



The Convergence and Divergence of HPC and AI

Mohamed Wahib

RIKEN Center for Computational Science



Central Message

Money will go after AI

Market will MOSTLY spec for AI

HPC has no option but to “shop” in that market

*More divergence between HPC and AI than we like to think?**

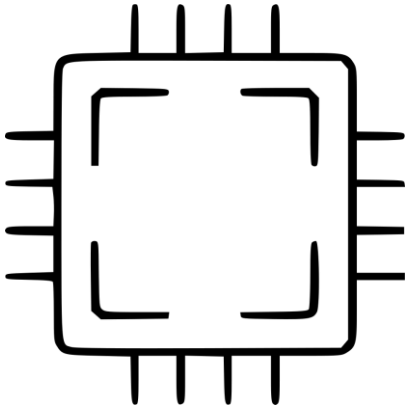
Convergence

- Too much analysis on how AI and HPC are:
 - “Converging”
 - “Two sides of the same coin”
 - “Aligned”

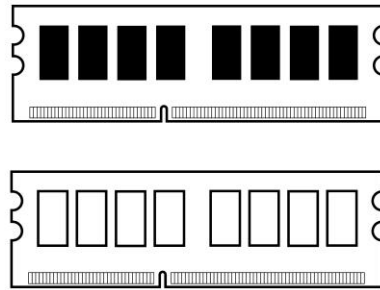
- And there is some truth to that, yet we have to acknowledge the differences
 - We focus on this talk however on **divergence**, since it is can be disruptive

Divergence

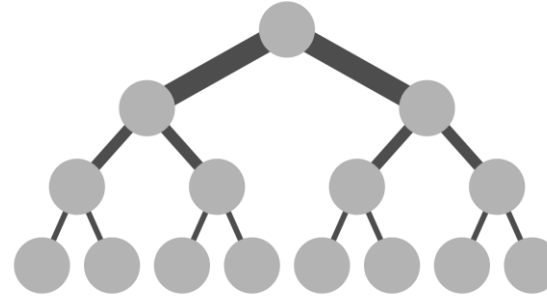
Compute



Memory



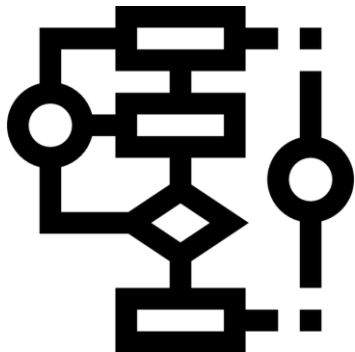
Network



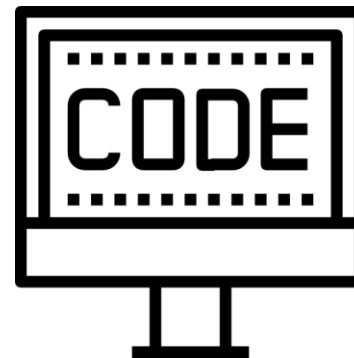
Storage



Algorithms



Programming

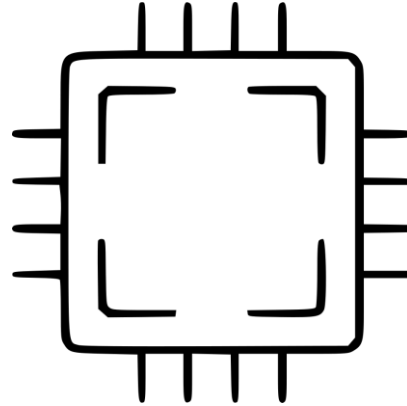


Libraries (Sys SW)

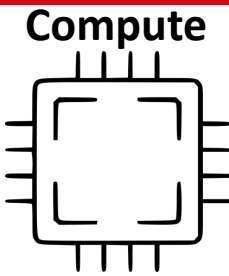


Divergence

Compute



Divergence



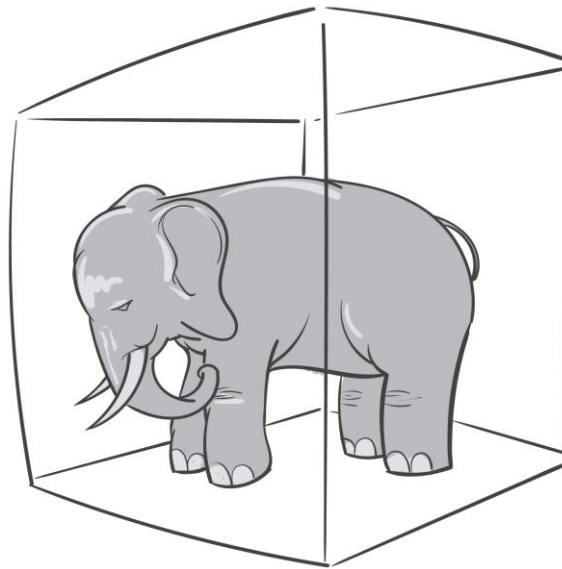
- Lower precision pushing out higher precision

- Libraries

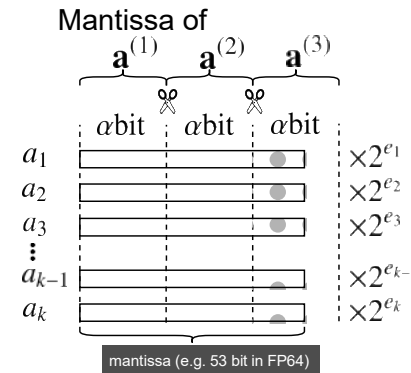
- Every library that can emulate, will emulate

- General code

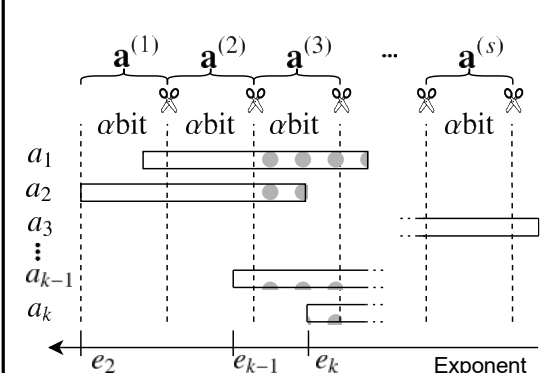
- Algorithmic
 - Automated mixed precision
 - User assisted



elementwise-place splitting



shared-place splitting (Ozaki scheme)

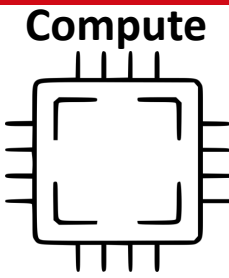


Emulate high precision using low precision* (Ex: Ozaki scheme)

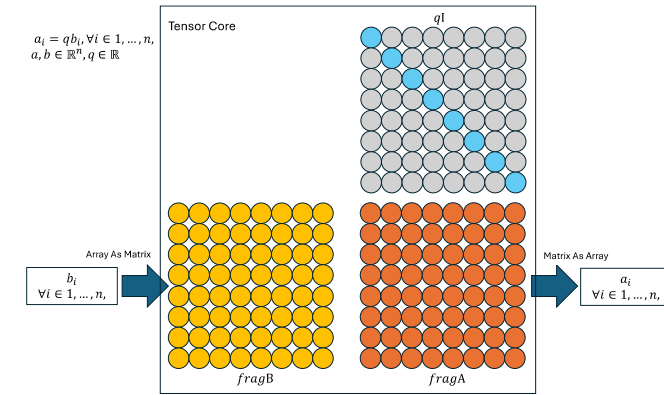
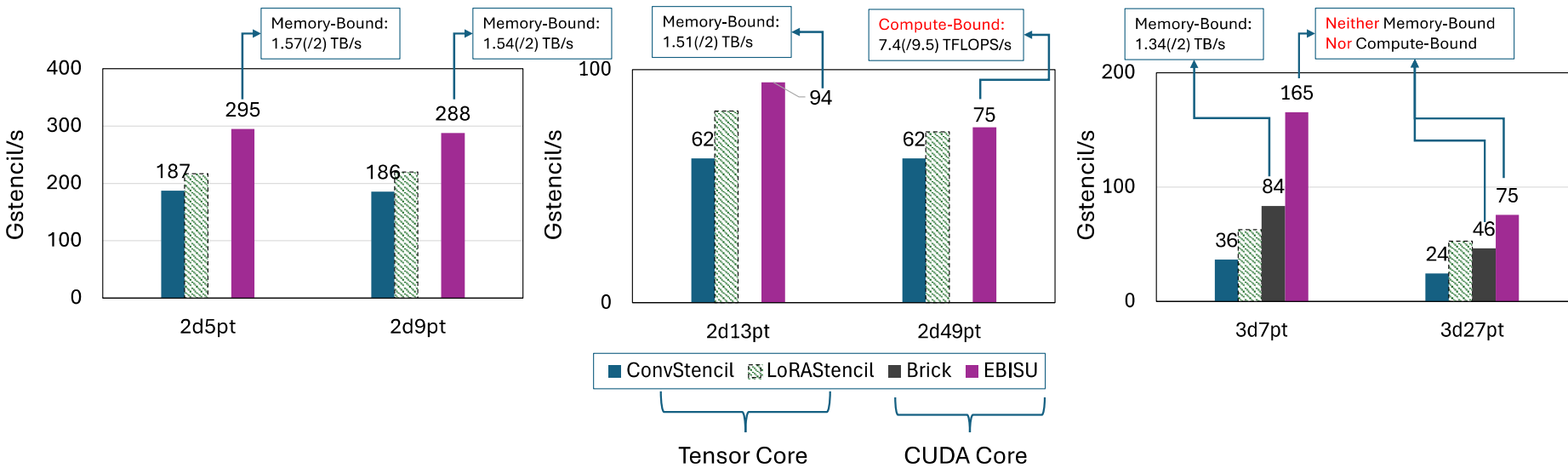
Risk: inefficient emulation/mixed precision (non-gemm) leading to small effective gain in perf. with new systems

* Ootomo et al., DGEMM on Integer Matrix Multiplication Unit, IJHPCA 2024

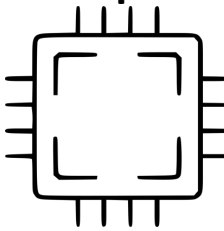
Divergence



- Memory-bound code: what does low precision do there?
 - Simple answer: “reduces the memory traffic!”
 - Complex answer: “low precision is mostly on matrix engines”



Divergence

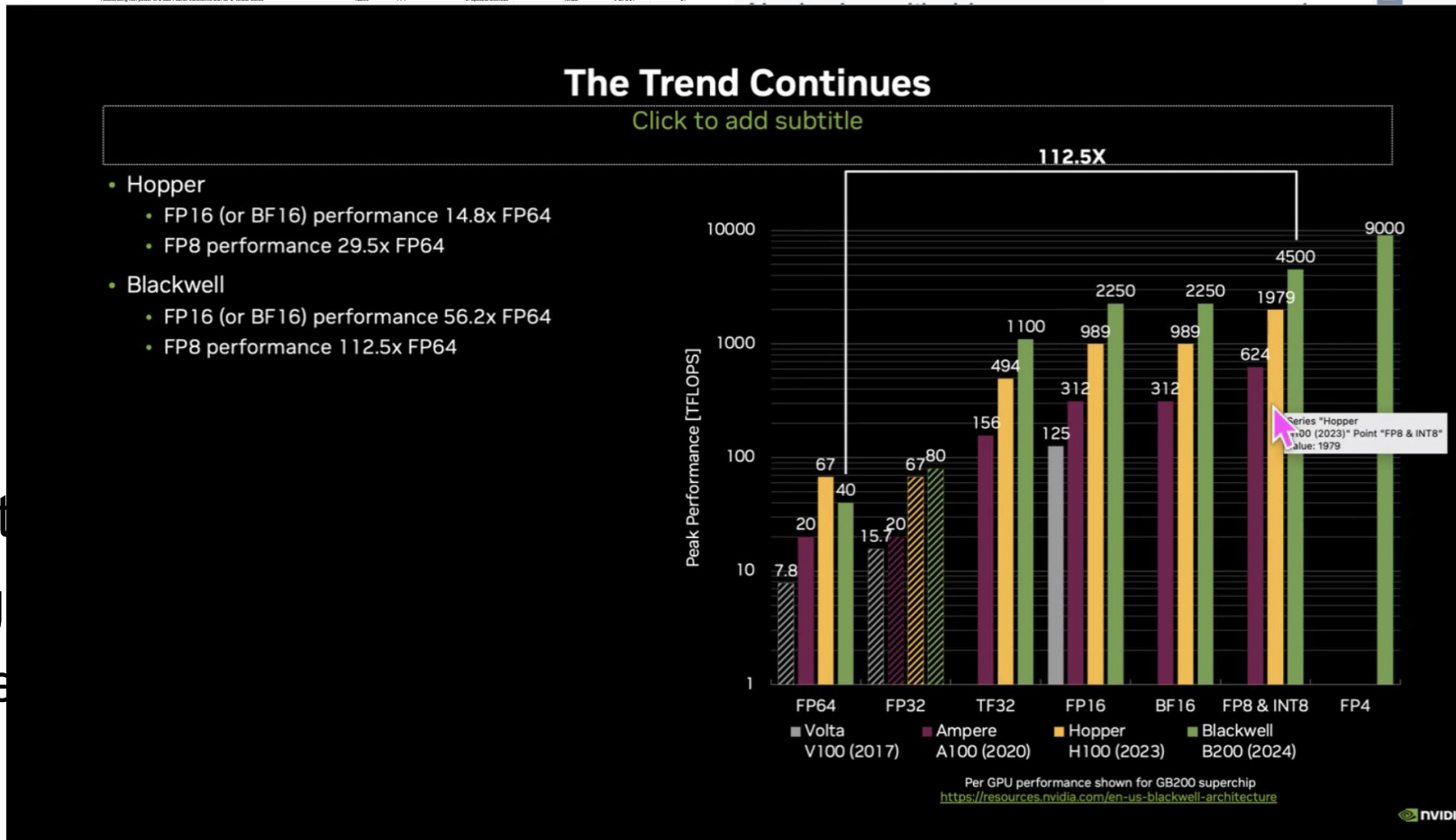


- Specialized hardware
 - Special compute units added to CPUs and GPUs: Matrix engines
 - ASICs

Title	Abstraction	Application	Device	Precision	Target Hardware	Venue	Year
Automatic Kernel Generation for 16bit Tensor Cores	Compiler	MathML DL operators (sum)	1. Denzel Core	None	ARMV9	ASPLOS	20
ANSI automatic kernel generation for neural processing units using polyhedral transformations	Compiler	MathML DL operators (sum)	1. Denzel Core	None	FLD21	ASPLOS	21
Polyhedral Transformation and Code Generation of Sparse Tensor Contraction with Coalescing	Compiler	Tensor (sparse)	2. Sparse Core	None	ARM M83	ASPLOS	21
Programming Tensor Cores from an Image Processing DSL	Compiler	Tensor	1. Denzel Core	None	SCOPES20	ASPLOS	20
AMDC: Streaming Automatic Mapping for Tensor Computations On Spatial Accelerators with Hardware Abstraction	Compiler	Tensor	1. Denzel Core	None	ARM	ISCP22	22
Accelerating Sparse and Sparse-Dense Transformations Using Tensor Cores and Strip Shuffles	Native	FFT	3. Sparse Methods	None	PLDI21	ASPLOS	21
Accelerating non-power-of-2 size Fourier transforms with GPU Tensor Cores	Native	FFT	3. Sparse Methods	None	ISCP21	ASPLOS	21

Application	COUNTA of Title	Year	Venue	COUNT of Year
-------------	-----------------	------	-------	---------------

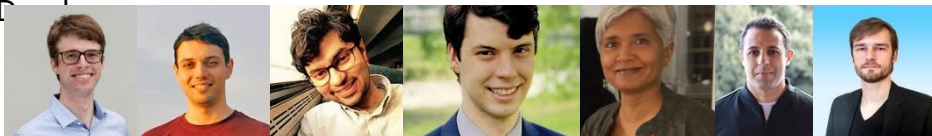
18	SC'18	1
19	ICS'19	1
20	ArXiv'20	1
	IPDPS'20	1
	SCOPES'20	1
21	Cluster'21	1
	IPDPS'21	1
	PACT'21	1
	PLDI'21	1
	SC'21	1
	TPDS	1
22	ACM TACO	1
	ICS'22	1
	IJHPCA	1
	ISCA'22	1
	PACT'22	1
	SC'22	1
23	SC'23	1
	TPDS	1
24	ArXiv	1
	ASPLOS'24	1
	ICS'24	2
	IJHPCA	1
	PPoPP'24	1
	SC'24	3



- Matrix eng
- Rigid: not
- Ex: using
- We need

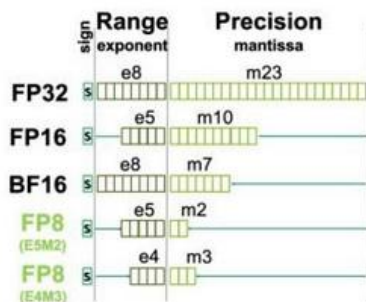
AI-HPC SW: RAPTOR – Numerical Profiling of Scientific Applications

Faveo Hoerold, Ivan Ivanov, Akash Dhruv, William Moses, Anshu Dubey, Mohamed Wahib, Jens



Motivation:

- Historically, easy choice: FP32 enough or is FP64 necessary?
- AI induced HW trends: low-precision units & FP64 capabilities reduced



Goals/Features:

- Easily applicable, support arbitrary precisions & numerical error analysis

Design Approach:

- LLVM compiler pass and a supporting runtime using MPFR

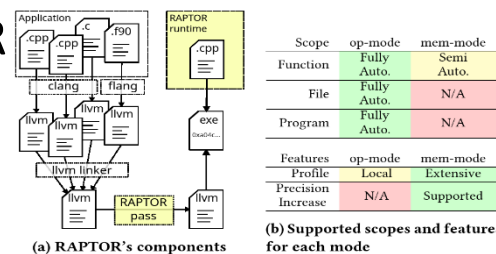


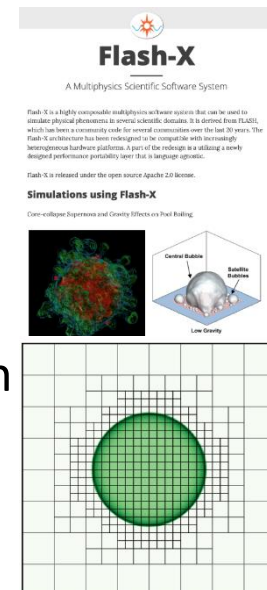
Figure 2: Overview of RAPTOR and config. matrix



SC'25 Best Reproducibility Advancement Award

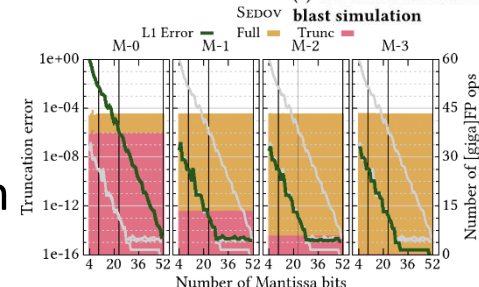
Flash-X: multi-physics multi-domain application software

- Simulation domain divided into blocks with **AMR**
- Shock wave** in radial direction
- Hypothesis 1:** using reduced precision in less-refined AMR blocks w/o quality loss



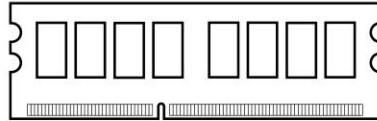
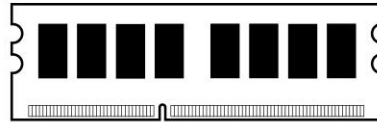
(a) The radial shock in the Sedov blast simulation

➔ Sedov problem is very robust to low-precision



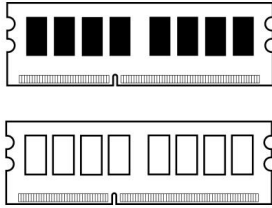
Divergence

Memory



Divergence

Memory

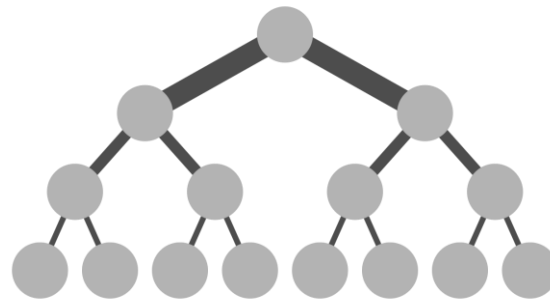


- High bandwidth is a requirement in both HPC and AI
 - Good!
 - Potential PIM innovations driven by AI would also be welcome by HPC
- Divergence: memory capacity
 - AI needs a lot of memory (arguably the main reason people buy more GPUs)
 - HPC doesn't need as much memory
 - Current HPC algorithms don't benefit from more memory
 - **Rewrite algorithms for abundant capacity (ex: aggressive use of look up tables?)**

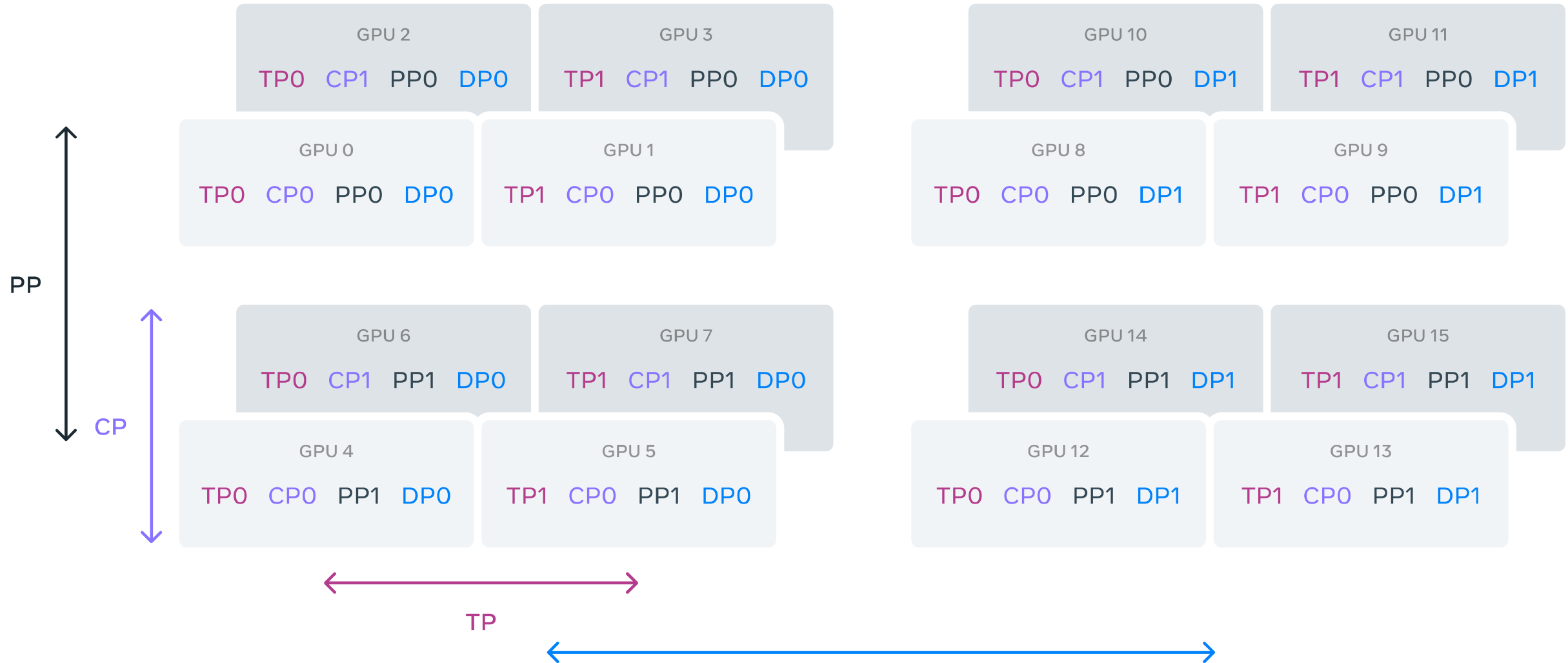
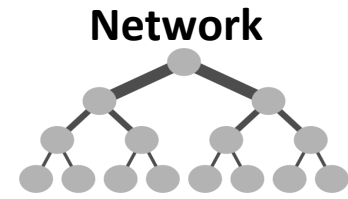
Risk: HPC has to pay high price for memory capacity it doesn't need; low return on perf.

Divergence

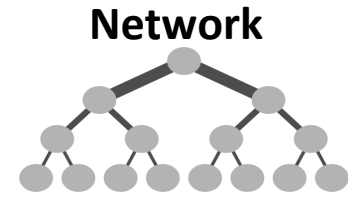
Network



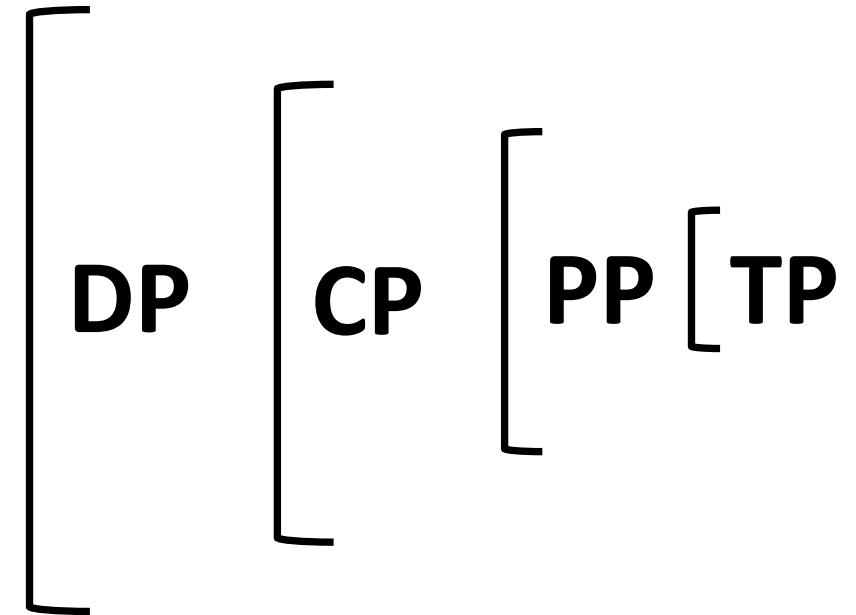
4D Parallelism: TP+CP+PP+DP



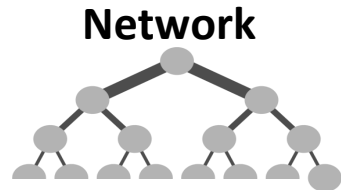
4D Parallelism: TP+CP+PP+DP



- Tensor Parallel [TP]
 - The more you split the layer via TP \rightarrow less compute and more comm
 - TP \rightarrow strong scaling
 - Conclusion: do TP inside the node (on a multi-GPU system)
 - In practice, we observe TP = 2~8
 - Depends on intra-node interconnect
- Context Parallel [CP]
 - Necessary evil
- Pipeline Parallel [PP]
 - Necessary evil
 - There is always inefficiency (bubble in a pipeline)
 - Used when running into the limits of TP, CP, and DP
- Data Parallel [DP]



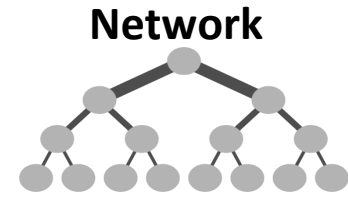
Divergence



For GTP3-175B parameterized as: $B = 16$, $E = 12K$, $S = 32K$, $\$N_p\$ = 175B$, $L = 96$, $W = 2$. Model_FLOPS is empirically measured (Model_FLOPS = 467.9 x 96 TF)

	FLOPS per Worker	Total FLOPS	Payload Size per Worker (Bytes: logical)	Agg. Payload Workers (Bytes: logical)	Real Comm. injected per Worker (All_reduce ring implementation)
TP only (T workers)	Model_FLOPS / T	Model_FLOPS (constant)	$(W \times B \times E \times S \times 4 \times L) / T$ (constant)	$W \times B \times E \times S \times 4 \times L$	$W \times B \times E \times S \times 4 \times L \times (2 \times (T-1)/T)$ (All_reduce)
Example: T = 8	5,614.8 TF	44,918.4 TF	589.8 GB	4,718.4 GB	8,257.2 GB
PP only (P workers)	Model_FLOPS / P	Model_FLOPS (constant)	$W \times B \times E \times S \times 2 \times P$ (constant)	$W \times B \times E \times S \times 2 \times P \times (P-1)$	$W \times B \times E \times S \times 2 \times P$ (P2P communication)
Example: P = 8	5,614.8 TF	44,918.4 TF	196.8 GB	1,377.6 GB	196.8 GB
DP only (D workers)	Model_FLOPS	Model_FLOPS x D (linear)	$(W \times \$N_p\$)$ (constant)	$W \times \$N_p\$ \times D$	$W \times \$N_p\$ \times (2 \times (D-1)/D)$ (All_reduce)
Example: D = 8	44,918.4 TF	359,347.2 TF	350 GB	2,800 GB	612.5 GB
TP+PP+DP (T x P x D workers)	Model_FLOPS / (T x P)	Model_FLOPS x D (linear)	$(W \times B \times E \times S \times 4 \times L) / (T \times P)$ + $(W \times B \times E \times S \times 2 \times P) / T$ + $W \times \$N_p\$ / (T \times P)$	$((W \times B \times E \times S \times 4 \times L) \times D + (W \times B \times E \times S \times 2 \times P \times (P-1) \times D) / T + W \times \$N_p\$ \times D)$	$(W \times B \times E \times S \times 4 \times L) / (T \times P) \times (2 \times (T-1)/T) + (W \times B \times E \times S \times 2 \times P) / T + W \times \$N_p\$ / (T \times P) \times (2 \times (D-1)/D)$
Example: 8 x 8 x 8	701.85 TF	359,347.2 TF	73.7 + 24.6 + 5.5 = 103.8 GB	18,873.6 + 1,377.6 + 2,800 = 23,051.2 GB	129 + 24.6 + 9.6 = 163.2 GB

Divergence



Fabric options/alternatives for 4x MI300A config

- Target: 1T model LLM training in BF16:
 - Training: 10-20x of working memory vs model parameters
 - 1/10 of ram * TP * PP * ~two bytes
 - need 20+ TB or 40-80 MI nodes
 - TP inside node, CP among 4 doing ops similar to allgather; PP is low volume P2P comm. betw 2 or 4 nodes
 - DP needs (TP*SP*PP) allreduce rings among #DP nodes
 - order of parall. dim. [TP, CP, PP, DP]

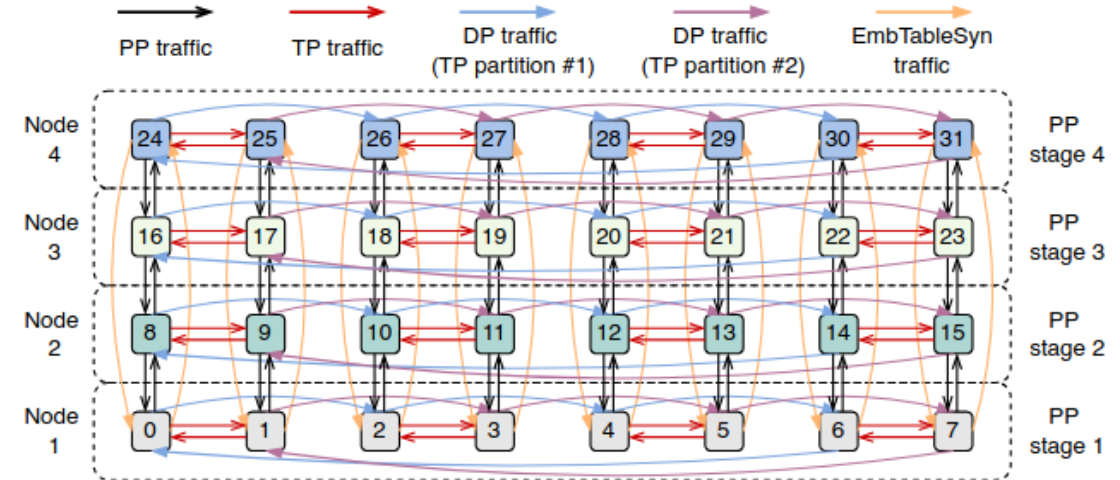


Figure 3: DeepSpeed's default parallelism mapping from $(p, t, d) = (4, 2, 4)$ to 32 GPUs (4 nodes).

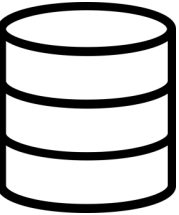
- Assume 1-partition layout MI300A (single node: 4* 128 GB)
 - => TP=4 & SP/CP=8 & PP=8 => allows 64 nodes or 32TB HBM memory
 - ==> DP=15 ← #nodes in each DP allredu is 15 and we have 256 of them
- => **need 256 non-overlapping rings embedded into topology** → **bisec.BW \geq 256x**
 (bisec.BW \geq 128x should be enough assuming 800G switch-to-switch links)

Divergence

Storage



Divergence



- AI/ML
 - Mostly reading (no need for strong Posix compliance)
 - Many small files vs. few large files (in HPC)
 - Favors local storage vs. shared storage (in HPC)
 - Different modalities

	GPFS			Burst Buffer		
Job Size	RI	WI	RW	RI	WI	RW
<i>Flagship</i>	78.57	21.43	0	88.89	11.11	0
<i>Medium</i>	55.94	14.29	29.77	37.04	62.50	0.46
<i>Small</i>	56.43	30.71	12.86	52.24	46.76	0.99

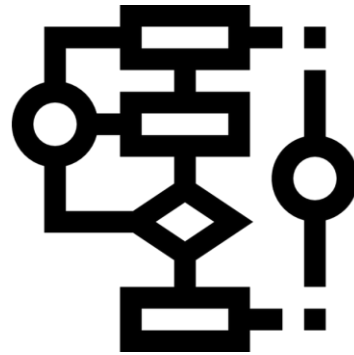
AI/ML jobs on Summit Supercomputer (ORNL)*

RI = Read Intensive; WI = Write Intensive; RW = Read-Write Intensive

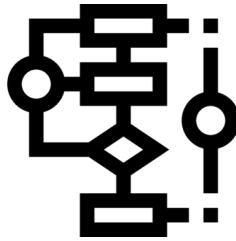
- **Object storage**

Divergence

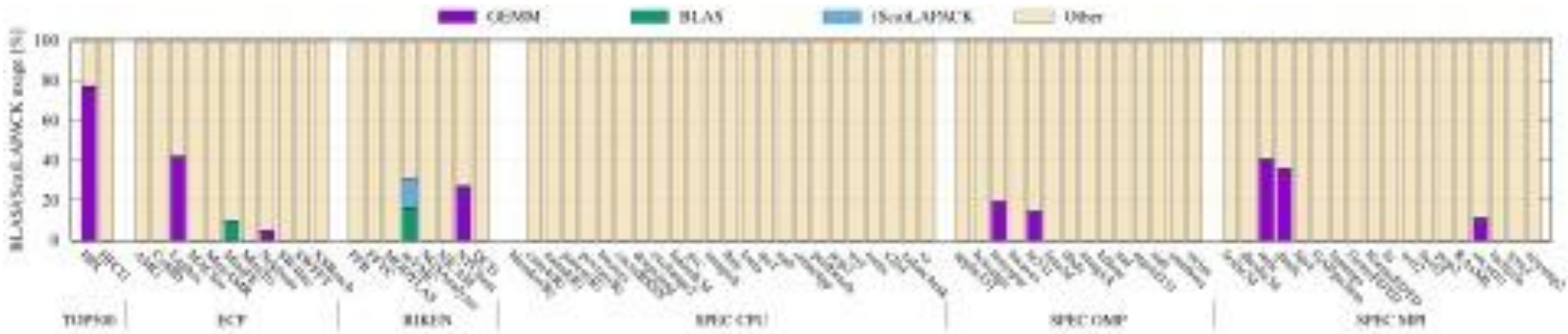
Algorithms



Divergence

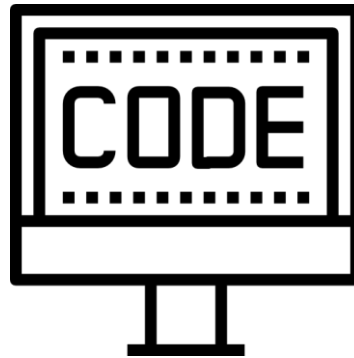


- *When you have a hammer, everything is a nail*
 - Dense LA drives AI/ML
 - How much of HPC can be “matrixfied”
 - Supercomputers have favored dense LA since the 90s
 - Yet most solvers are not matrixfied yet*

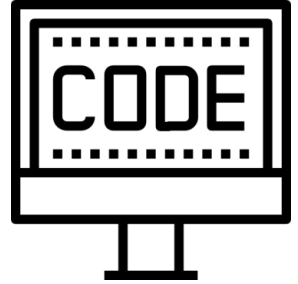


Divergence

Programming



Divergence



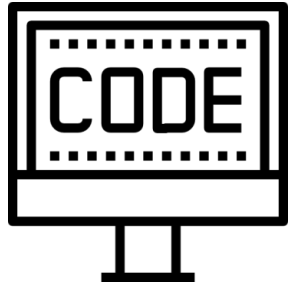
ML Approached to Correctness in Code Generation

- ▶ Extensive unit testing
 - ▶ Coverage
 - ▶ Writing tests is not trivial
- ▶ Surrogate ML model
- ▶ Round-trip
 - ▶ Trust issues
- ▶ Limit ML to short sequences of instructions
- ▶ Limit ML to a set of determined transformations
 - ▶ Not general
- ▶ Symbolic execution
 - ▶ Not well established

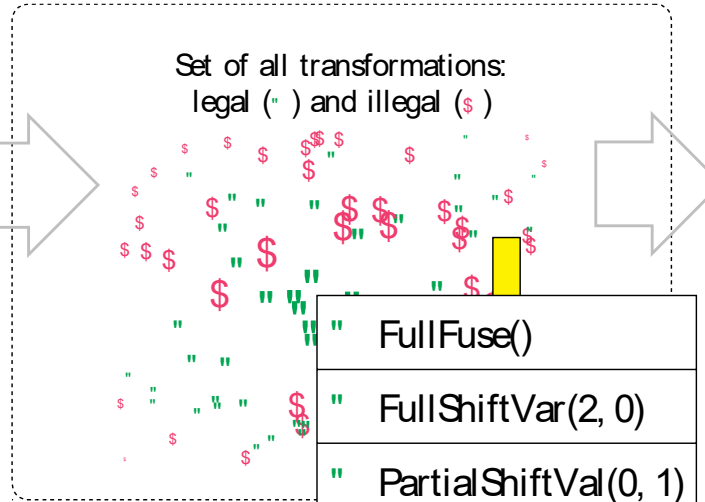
Edsger W. Dijkstra

“Program testing can be used to show the presence of bugs, but never to show their absence!”

Divergence

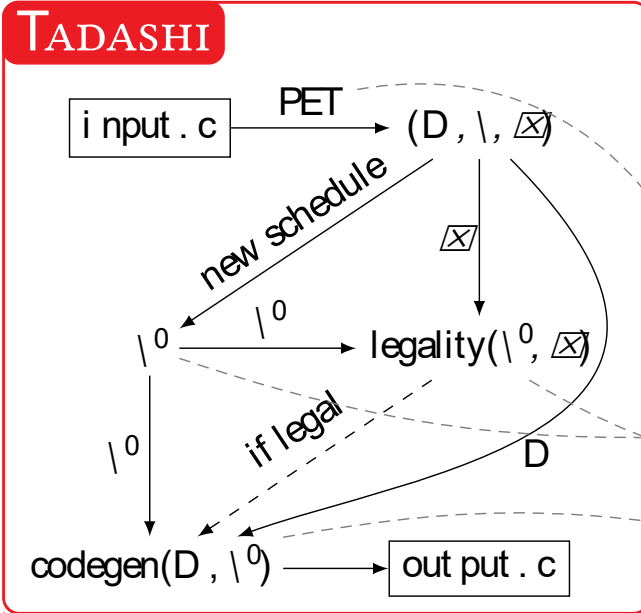


```
input.c
for (int t=0; t<T; t++) {
  for (int i=2; i<N-1; i++)
    b[i] = (a[i-1] + a[i] + a[i+1])/3;
  for (int j=2; j<N-1; j++)
    a[j] = b[j];
}
```



```
output.c
for (int c0=0; c0<T; c0+=1) {
  b[2] = (a[1] + a[2] + a[3]) / 3;
  for (int c1=2*c0+3; c1<N+2*c0-1; c1+=2*c0+1) {
    b[-2*c0+c1] = (a[-2*c0+c1-1] + a[-2*c0+c1]
                  + a[-2*c0+c1+1]) / 3;
    a[-2*c0+c1-1] = b[-2*c0+c1-1];
  }
  a[N-2] = b[N-2];
}
```

SEAL OF CORRECTNESS
Q.E.D.



- " FullFuse()
- " FullShiftVar(2, 0)
- " PartialShiftVal(0, 1)
- " SetLoopOpt(0, 3)

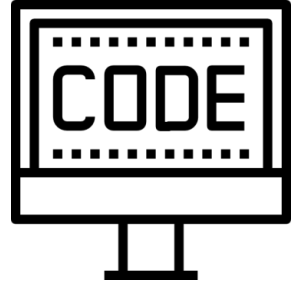
```
train.py
app = Simple("./input.c")
loop_nests = tadashi.Scops(app)

model = Model()
for _ in range(10):
  tr = model.sample_tran(loop_nests)
  legal = loop_nests.transform(tr)
  if legal:
    loop_nests.generate_code()
    t = app.compile().measure()
    model.update(tr, t)
```

Any ML model

- Choice of ML models:
- Supervised learning
 - Sample: Random primitive transformation
 - Unsupervised learning (LLMs)
 - Sample: Random composite transformations for a dataset
 - Reinforcement learning
 - Sample: Neighbouring primitive transformations
 - Evolutionary algorithms
 - Sample: Random sequence of primitive transformations
 - Auto-tuning
 - Sample: Grid/interval of composite transformations

Divergence



- ▶ i. <https://vatai.github.io/>
- ▶ ii. <https://github.com/vatai/tadashi/>
- ▶ iii. <https://arxiv.org/abs/2410.03210>

