Los Alamos NATIONAL LABORATORY

A Return to the Unusual Business of Tailored Computing

Galen M. Shipman

With major contributions from the Systematic Improvement by Design team
(Jered Dominguez-Trujillo · Kamalakkannan, Kamalavasan ·
Paul Henning · Kyle Roarty · Kevin Sheridan and Gary Grider)

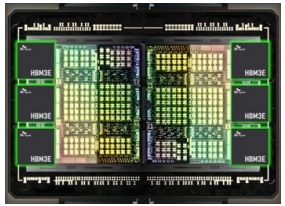
April 23rd 2024

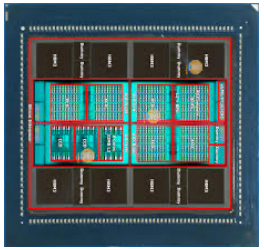
LA-UR-24-23732

Los Alamos NATIONAL LABORATORY

Why do we need to codesign tailored technologies?

Current technology trends – Increase width, cost, power!

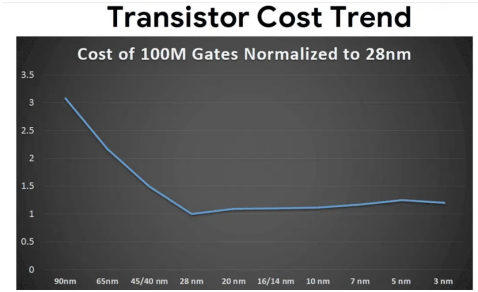




HBM4 goes wider

GPUs go wider (will approach 2KW) Heavily relying on advanced packaging technologies (CoWoS, Foveros, EMIB) designed to exceed 2.5KW

Operations get simpler (FP4!) -> increasing effective width!



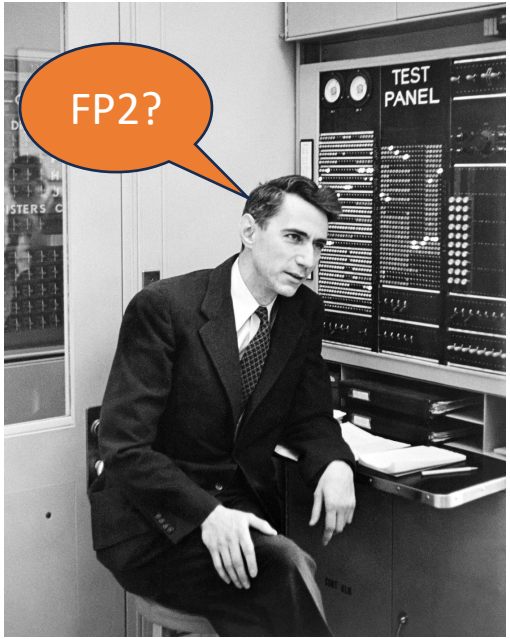
"Global Datacenter Critical IT power demand will surge from 49 Gigawatts (GW) in 2023 to 96 GW by 2026, of which AI will consume ~40 GW"

Every 8 cylinders consumes

1-1/2 gallons of nitromethane per second

AI Datacenter Energy Dilemma - Race for AI Datacenter Space

Less Signal?

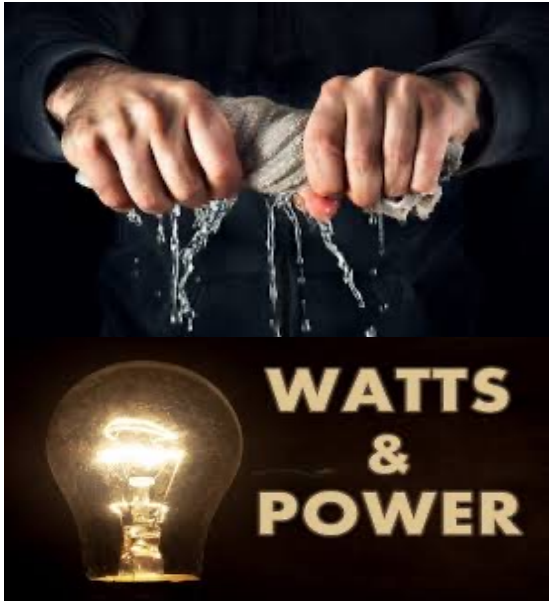


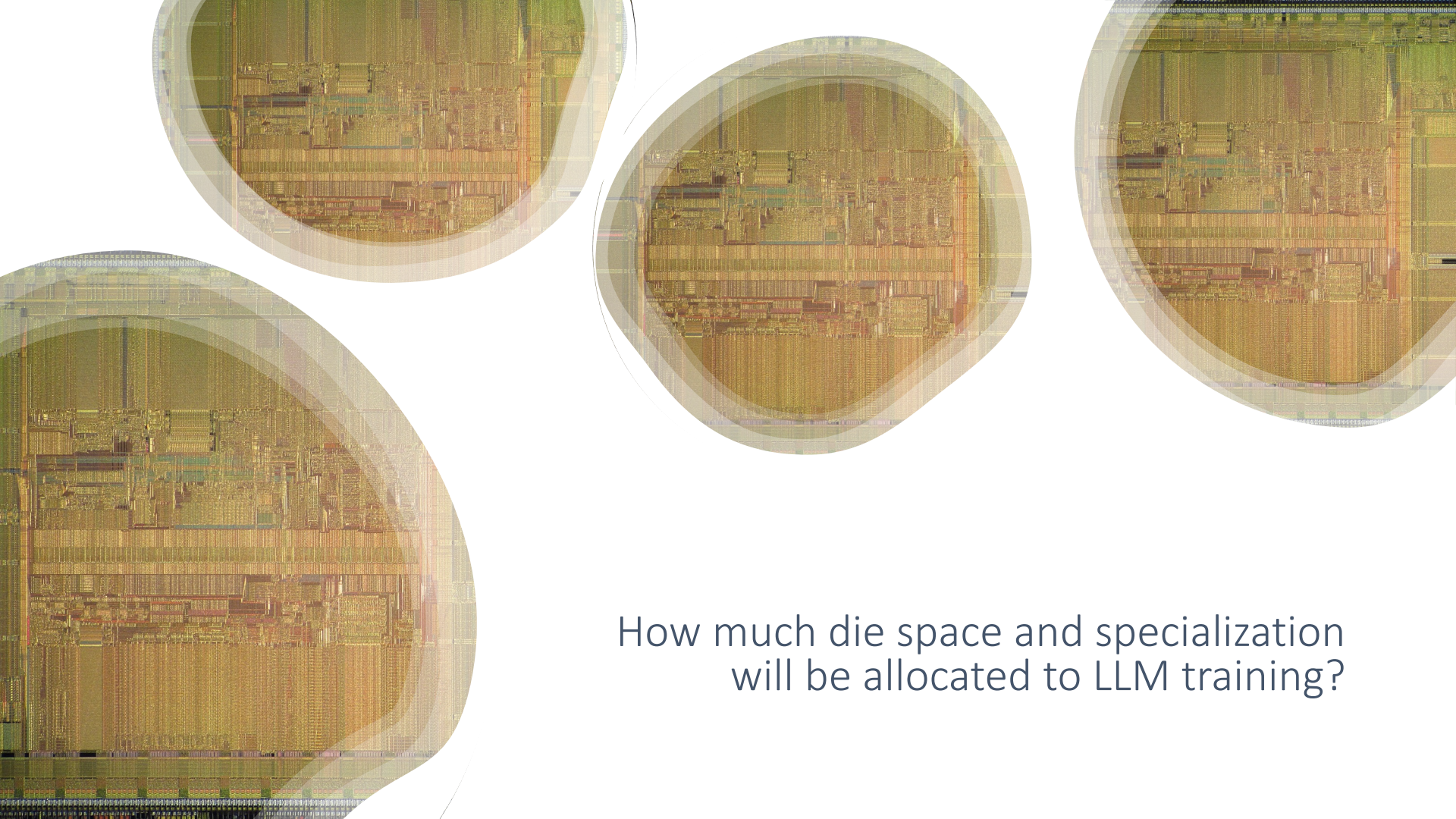
More Power?

6 x Grand Coulee Dam

2 x Three Gorges Dam

Less Waste?





How much die space and specialization will be allocated to LLM training?

So should our strategy be?

Advanced Porting (circa 2023)?

.. Or Codesign (circa 1996)?

Design Automation for Embedded Systems, 1, 121–145 (1996)

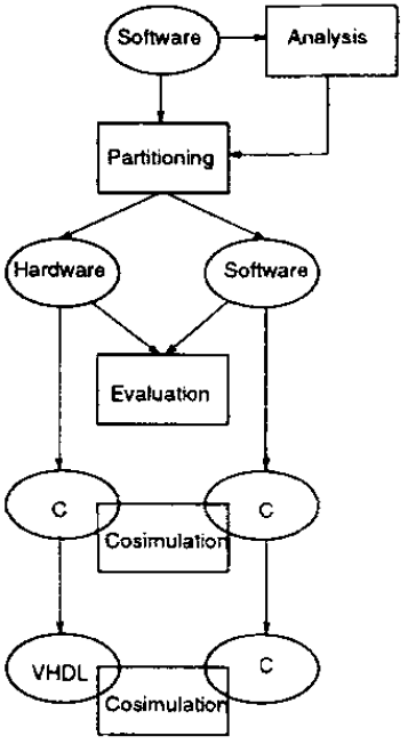
Codesign Analysis of a Computer Graphics Application

J. MADSEN AND J.P. BRAGE* jan@id.dtu.dk, brage@delta.dk
Department of Computer Science, Technical University of Denmark, DK-2800 Lyngby, Denmark

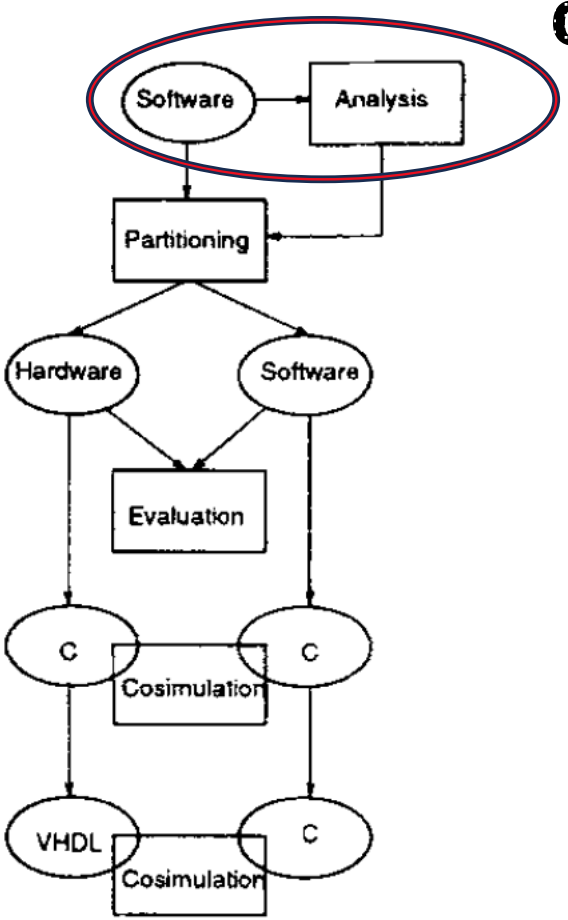
Editor:

Abstract. This paper describes a codesign case study where a computer graphics application is examined with the intention to speed up its execution. The application is specified as a C program, and is characterized by the lack of a simple compute-intensive kernel. The hardware/software partitioning is based on information obtained from software profiling and the resulting design is validated through cosimulation. The achieved speed-up is estimated based on an analysis of profiling information from different sets of input data and various architectural options.

Keywords: Codesign, Analysis, Speed-up, Cosimulation, Computer Graphics

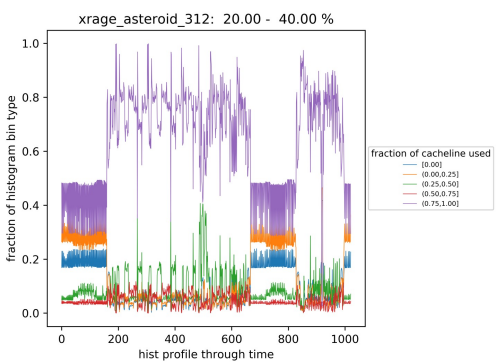


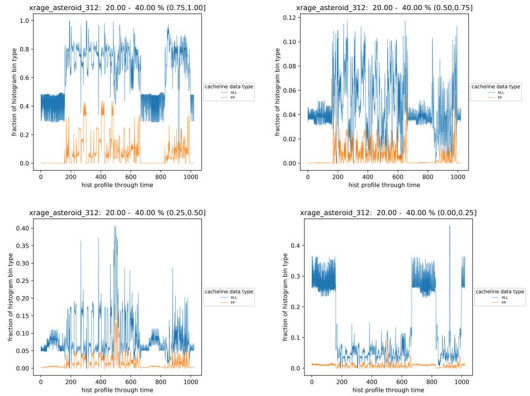
Codesign (circa 1996) - Analysis



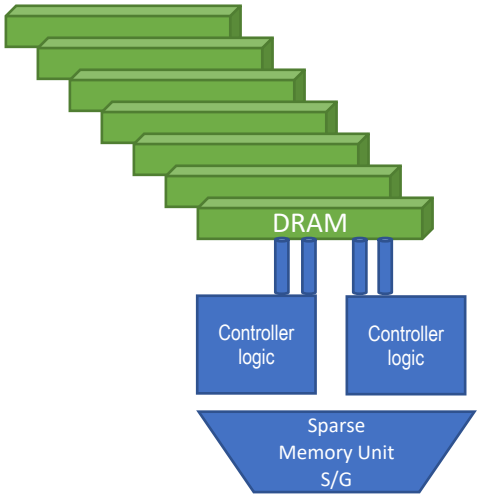
Instruction	Count	Percentage
Load	6,775,030,849	18%
Branching	6,063,697,707	16%
Integer Add	5,334,155,682	14%
Array Indexing	4,855,537,532	13%
Conditional	3,299,248,274	9%
Store	2,599,966,427	7%
Type cast	1,959,938,043	5%
Sign extension	1,541,094,404	4%
Stack frame allocation	1,221,694,311	3%
FP multiplication	1,171,615,897	3%
FP comparison	1,141,415,386	3%
INT multiplication	991,524,374	3%

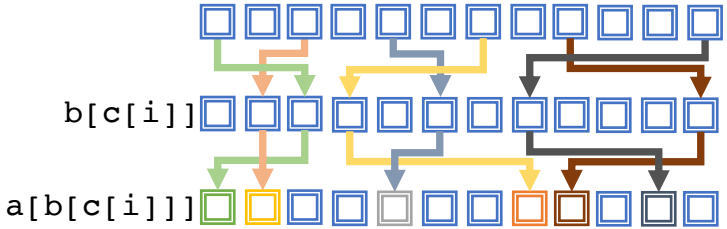
Memory subsystem						Floating Point		
L1	L2	L3	DRAM	DRAM BW	Mem Latency	DP FLOPs	Vectorization	Non-FP
						2.50%	7.10%	97.50%
						26.20%	90.40%	73.80%
						14.30%	0.20%	85.70%
						6.50%	14.00%	93.50%
						7.80%	19.20%	92.20%
						8.10%	17.60%	91.90%





Architecture? Can memory tech get smarter?

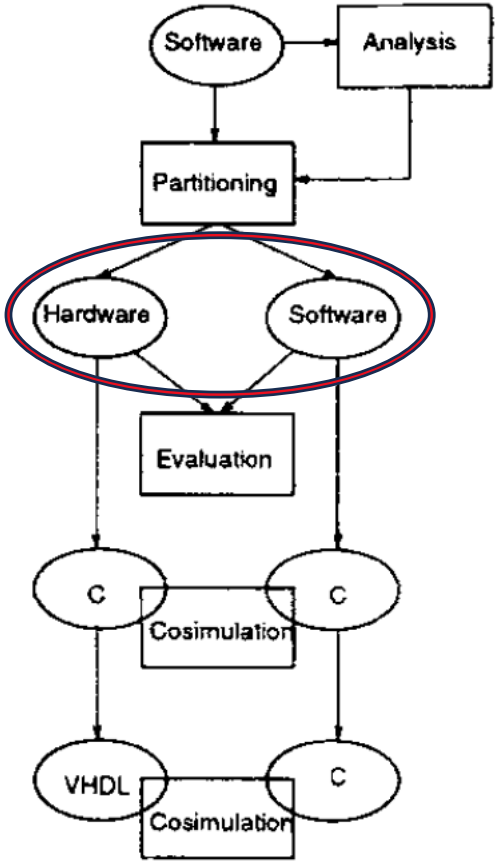




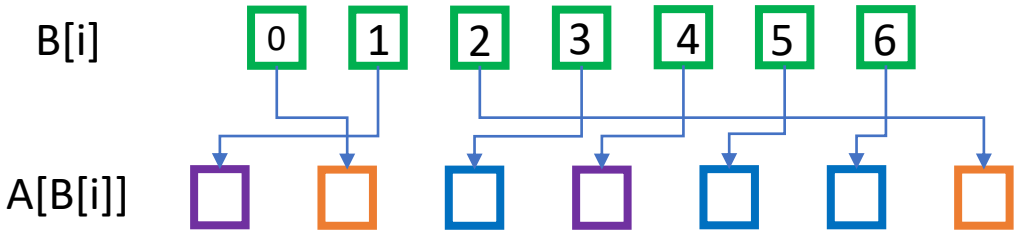
Potential: Integrate scatter/gather accelerations into controller logic allowing coalescing of small grain accesses to maximize bandwidth

But many challenges:

1) Explicit programming scatter/gather?
2) Changes to MMU / Virtual memory / Caches?
3) Changes to ISA and/or µarch
4) Scratchpads or caches?

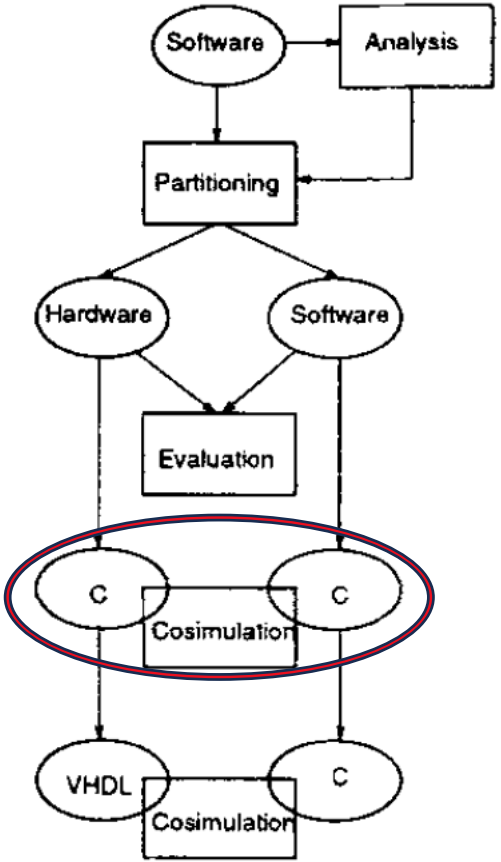


Hardware

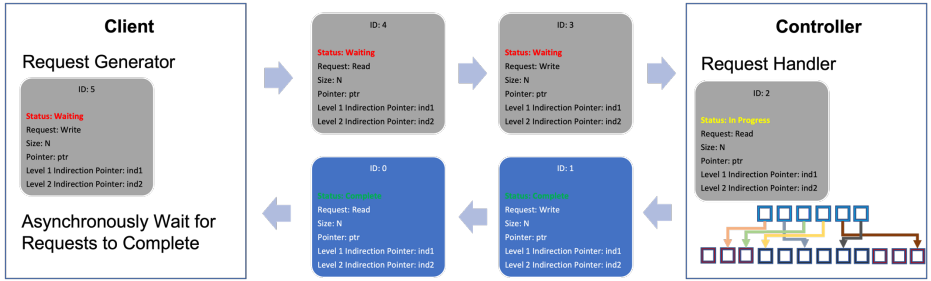


Software

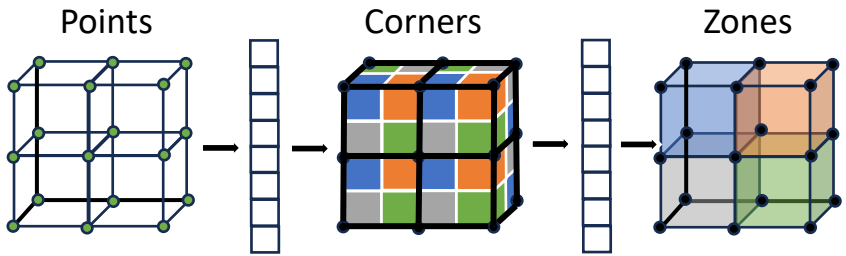
```
// Original implementation
for(int i = BEGIN; i < END; i++){
    sum += A[B[i]];
}
//********** Transformation *************//
// Assuming BEGIN and END are multiple of
SG_accelerator SG_acc;
auto dH = SG_acc.add_dataStructure(A, B, BEGIN, END);
SG_acc.start();

VecType sum_v; // initialize it to zeros
// Assuming BEGIN and END are multiple of VecSize
for(int i = BEGIN/VecSize; i < END/VecSize; i++){
    sum_v =  sum_v + SG_acc.getNextVecData(dH);
}
auto sum = sum_v.sum();
```



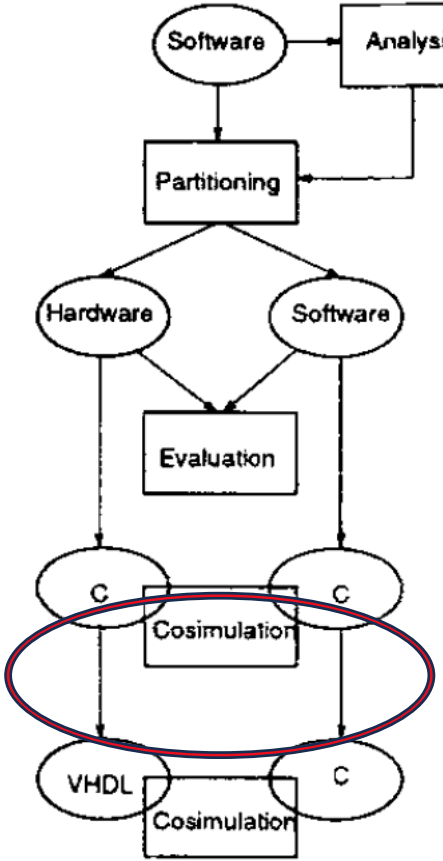
ScorIA: Sparse Memory Acceleration Testbed

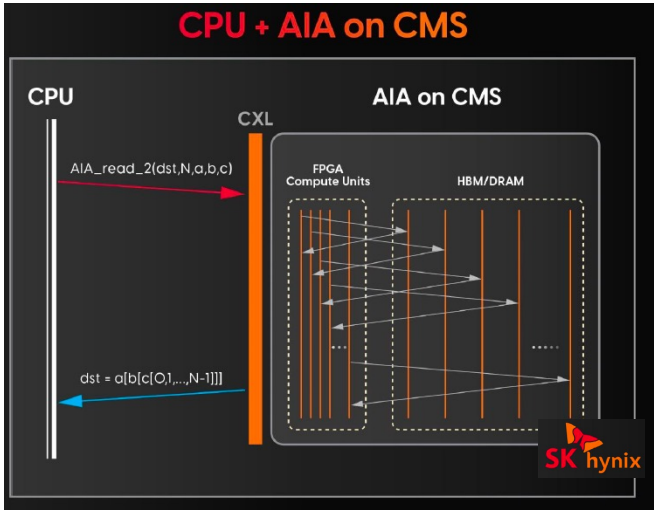


Unstructured Mesh Explorations (LAP Proxy)

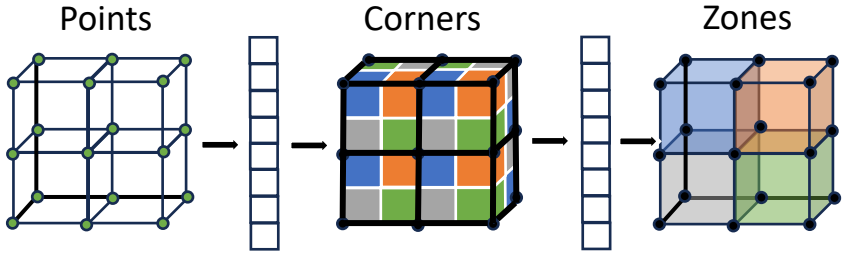


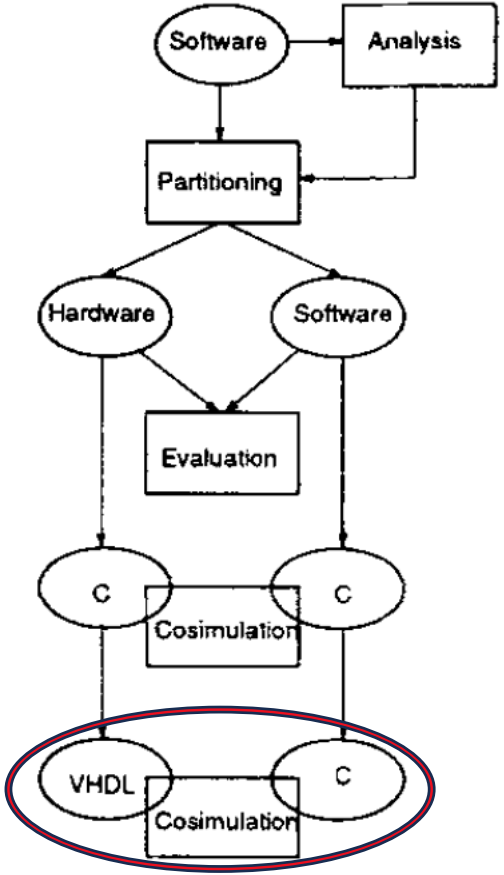
Accelerate Indirect memory Access – ScorIA on FPGA



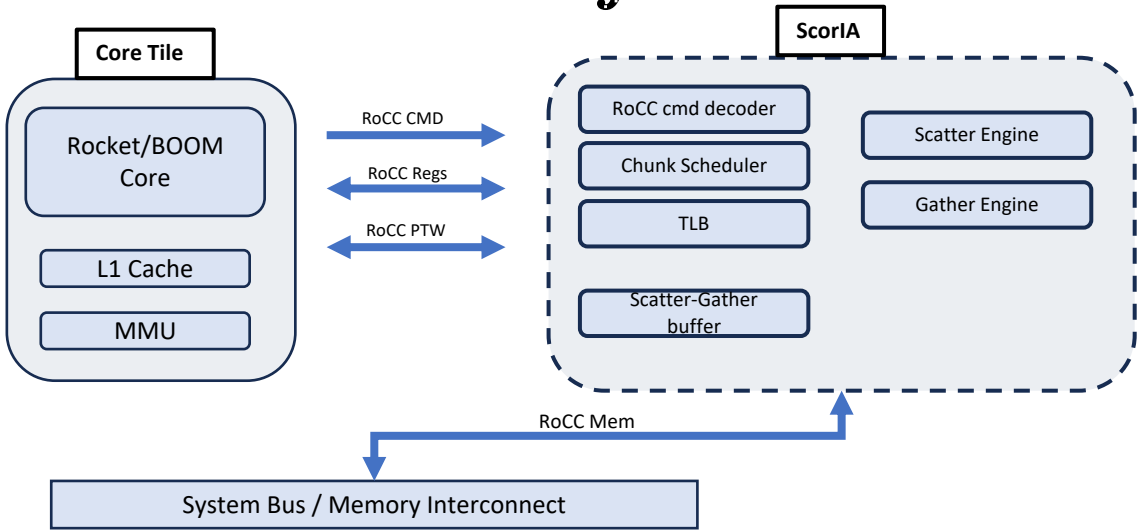


Unstructured Mesh Explorations (LAP Proxy)

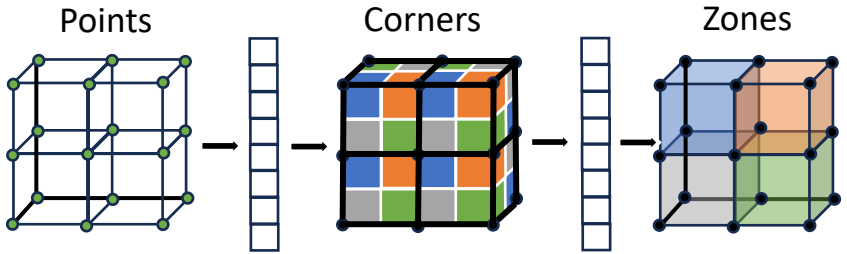




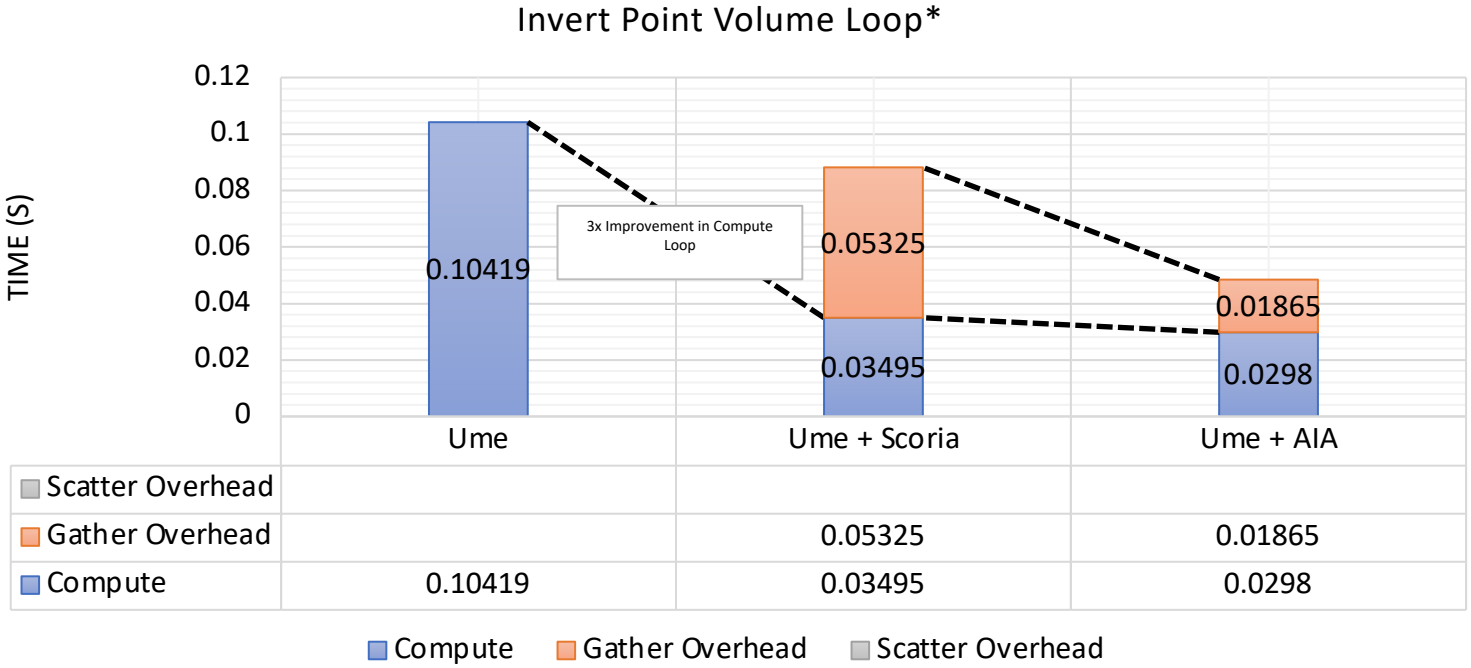
ScorIA hardware prototype in ChipYard, DRAMsim3 and DRAMSys



Unstructured Mesh Explorations (LAP Proxy)



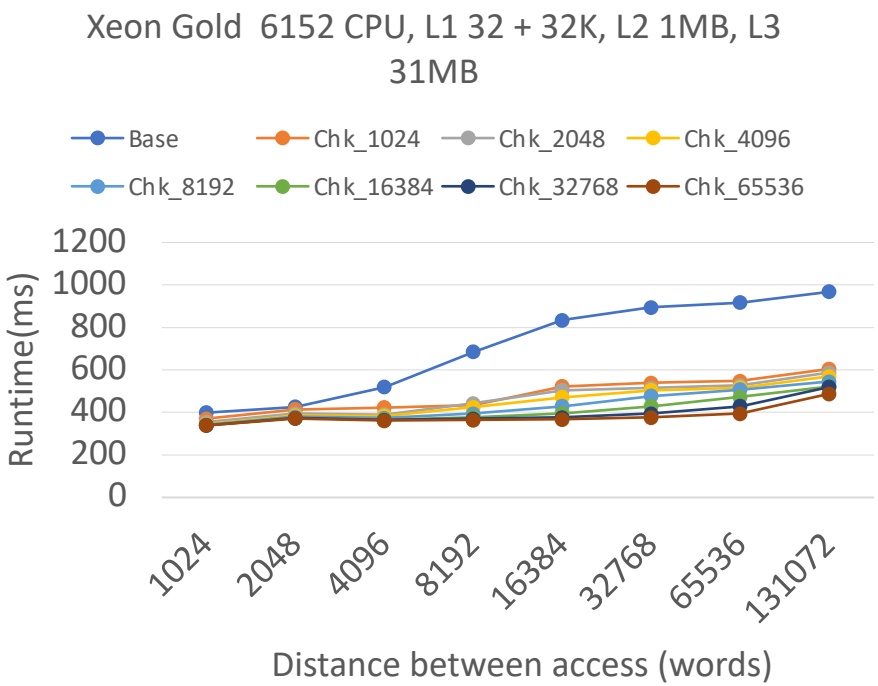
Performance Improvements with Software and FPGA prototypes

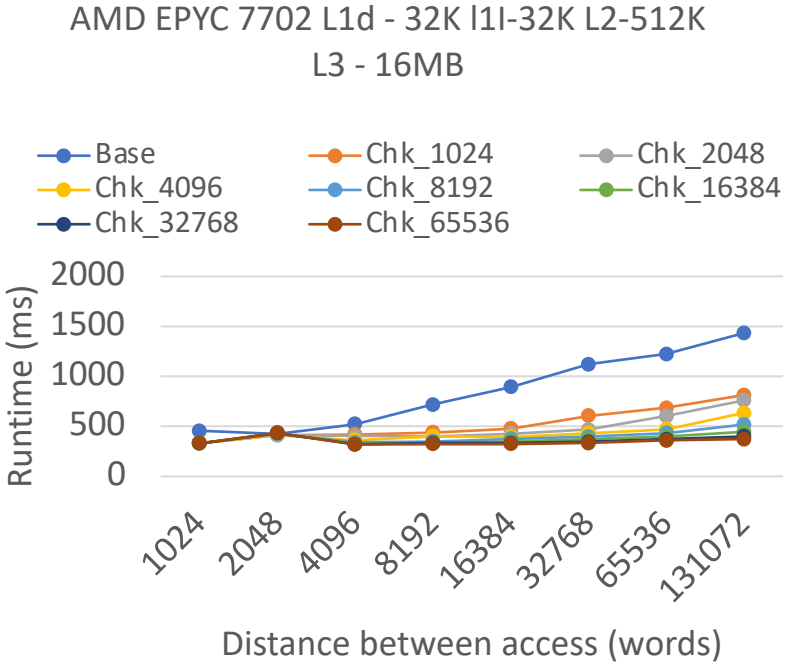


Observed Benefits:

- Compute loop performance from removing indirection (ScorIA)
- Bandwidth performance from gather/scatter accelerator FPGA prototype (AIA)

Accelerations can significantly improve performance

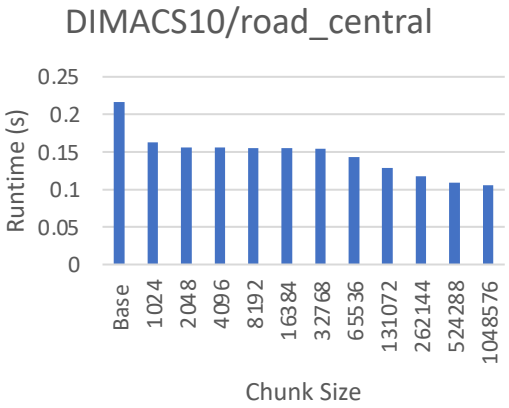


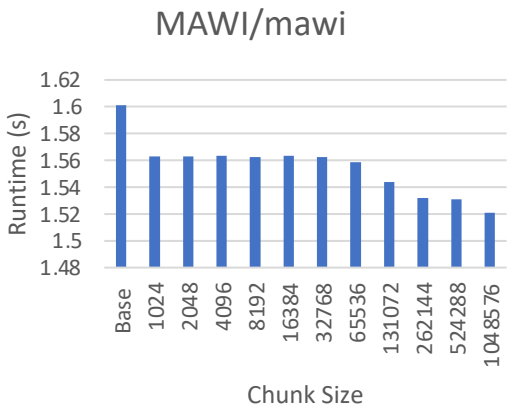


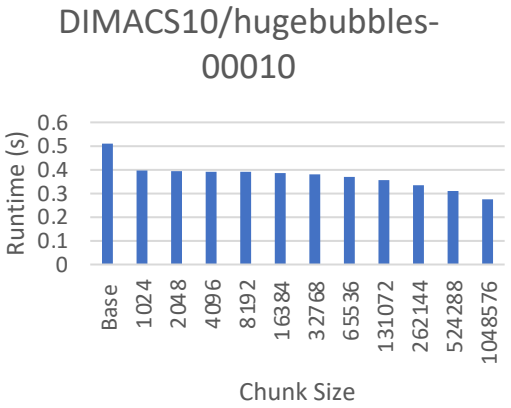
Analysis using manual acceleration of memory accesses on actual hardware show significant uplift

And Carry Across to Adjacent Workloads

Execution time of SpMV (from Suite Sparse)







Initial analysis shows memory accelerations can improve performance across a broad set of sparse matrix workloads

So are we a unique snowflake?
or is our memory access something like..

- High-dimensional sparse embeddings
-
- In-memory table-joins with sparse keys

dataset	#users	#items	#interactions	density
MovieLens-20M	136677	20108	9990030	0.0036
Netflix dataset	463435	17769	56880037	0.0069
Million Song Dataset	571355	41140	33633450	0.0014

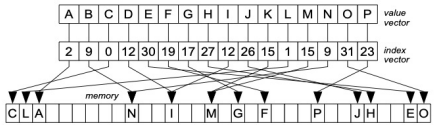
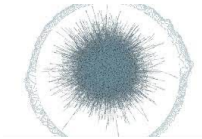


Figure 4: Scatter operation

- Large scale graph analysis
  - Multiple exemplars, but special purpose (Intel PIUMA, Cray Urika-XA)



- ***Can we codesign processor and memory tailors that address these use-cases in tandem?***

*Rethinking SIMD Vectorization for In-Memory Databases
http://www.cs.columbia.edu/~orestis/sigmod15.pdf


Questions?