Optimizing Network Resilience Using Domain-Specific Hardware Accelerator for Dynamic Programming

Ali Mazloum*, Sergio Elizalde*, Samia Choueiri*, Elie Kfoury*, Jose Gomez°, Ali AlSabeh†, Jorge Crichigno*

*University of South Carolina (USC)
°Fort Lewis College (FLC)
†University of South Carolina Aiken (USCA)

Innovating the Network for Data-Intensive Science (INDIS) Workshop

ST. Louis, MO
November 16, 2025

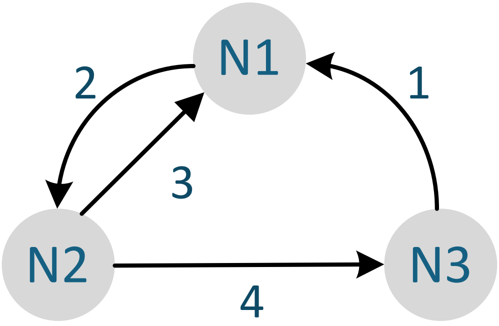
Domain-Specific Accelerators for Dynamic Programming

- Dynamic programming accelerators (DPAs) are devices that:
  - Adopt the many thread architecture
  - Incorporate hardware-accelerated instructions for dynamic programming (DP) workloads
- DPAs may enhance the performance of many networking functions, including:
  - Network resilience
  - Partial-matching deep packet inspection (DPI)

Network Resilience

- Network resilience can be expressed as an all-pairs shortest path problem
- Floyd-Warshall DP algorithm can be used to compute optimal paths
- The recurrence function of Floyd-Warshall is:

$$D_{i,j} \leftarrow \min\left(D_{i,j},\ D_{i,k} + D_{k,j}\right)$$



	N1	N2	N3
N1	0	2	∞
N2	3	0	4
N3	1	∞	0

	N1	N2	N3
N1	0	2	6
N2	3	0	4
N3	1	3	0

Network Resilience

- This function can be implemented using a fused function on DPA:
  - *__viaddmin_s32 (a, b, c):* Computes min(a + b, c)

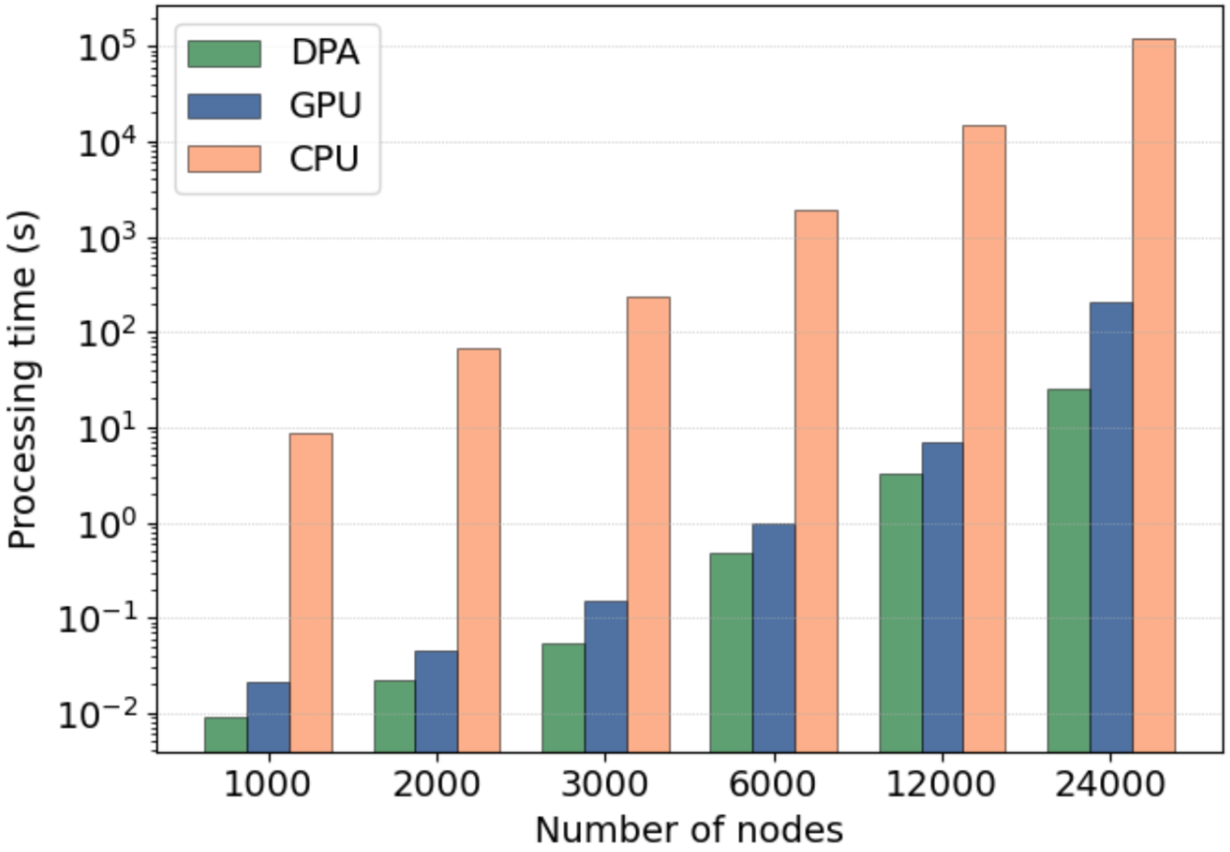
- DPA supports many other functions, such as:
  - *__vimax3_s32*
  - *__vimin3_u16x2*

Evaluation

- The performance is assessed across three dimensions:
  - Processing time
  - Computational throughput (measured in billion cell updates per second, BCUPS)
  - Energy consumption
- The evaluation considers three platforms:
  - NVIDIA H100 (DPA)
  - NVIDIA A100 (GPU)
  - Intel Xeon Silver 4114 (CPU)

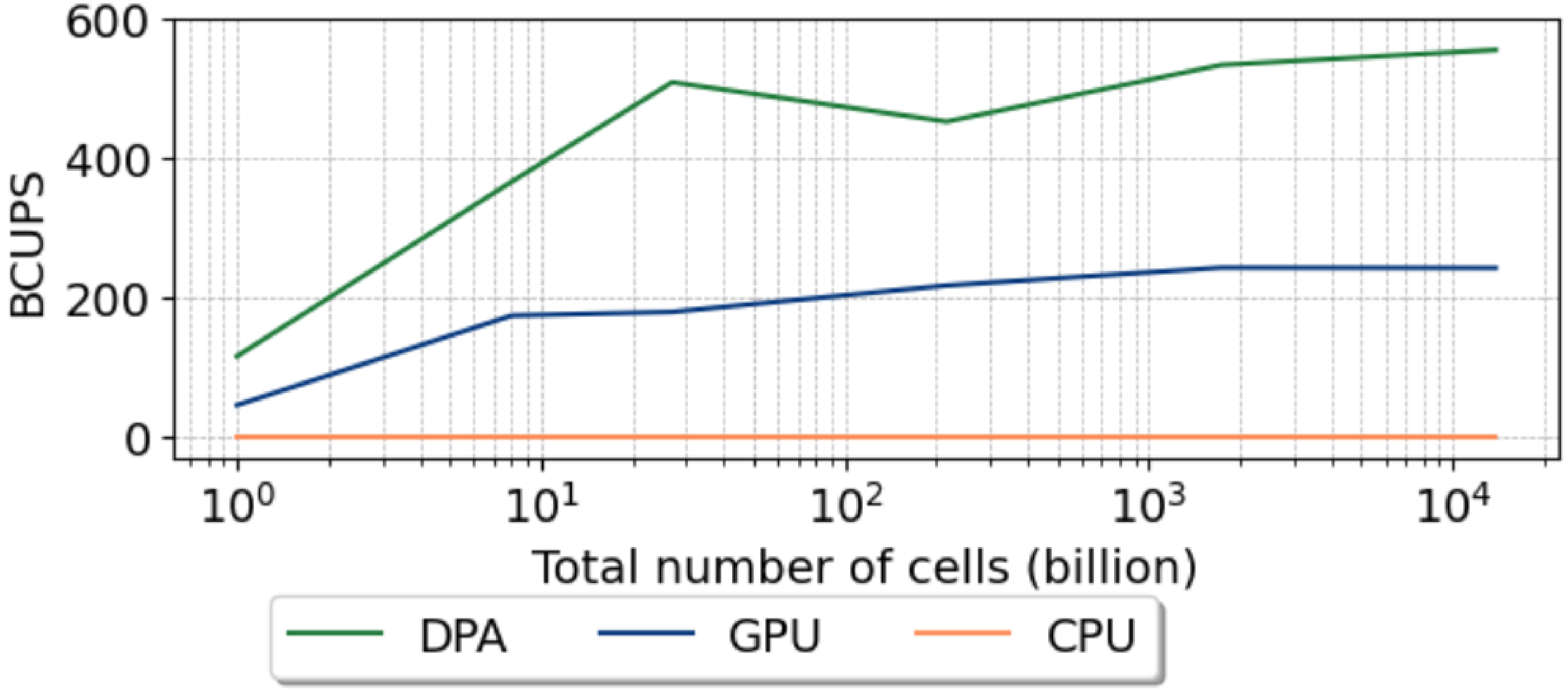
Evaluation Results: Processing Time

- Evaluation on three platforms using graphs with 1K, 2K, 3K, 6K, 12K, and 24K nodes
- DPA consistently achieved between 2x to 3x speedup compared to GPU
- DPA achieved between 1,000x and 4,600x speedup compared to CPU



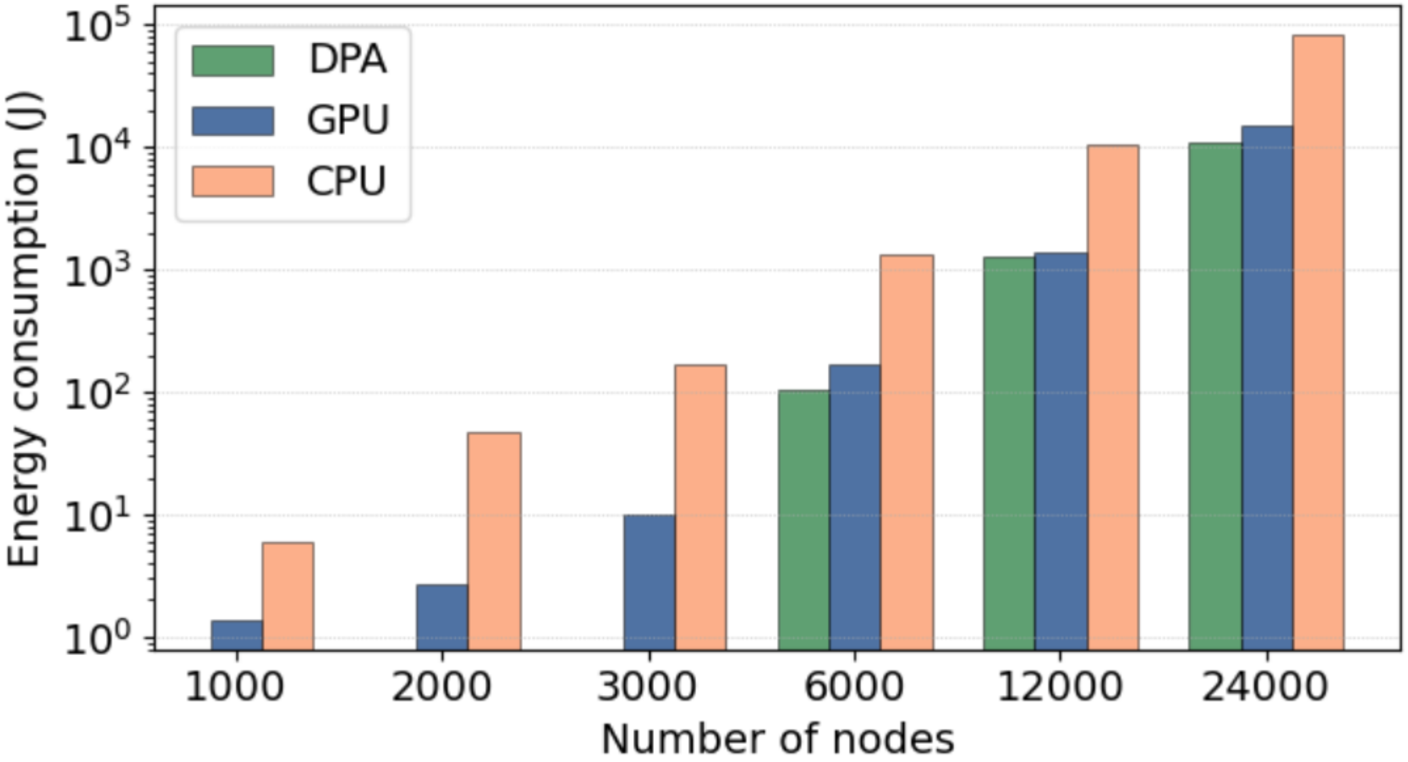
Evaluation Results: Throughput

- DPA throughput increases with the problem size
- DPA achieved peak performance at 555.79 BCUPS with 24K nodes
- GPU showed a similar increasing trend as the DPA
- CPU sustained around 0.116 BCUPS



Evaluation Results: Energy Consumption

- For 1K to 3K nodes, the kernel completion time is too short to get energy usage
- DPA consistently demonstrates the highest energy efficiency
- GPU has similar energy consumption compared to DPA
- CPU has one order of magnitude higher energy consumption compared to DPA



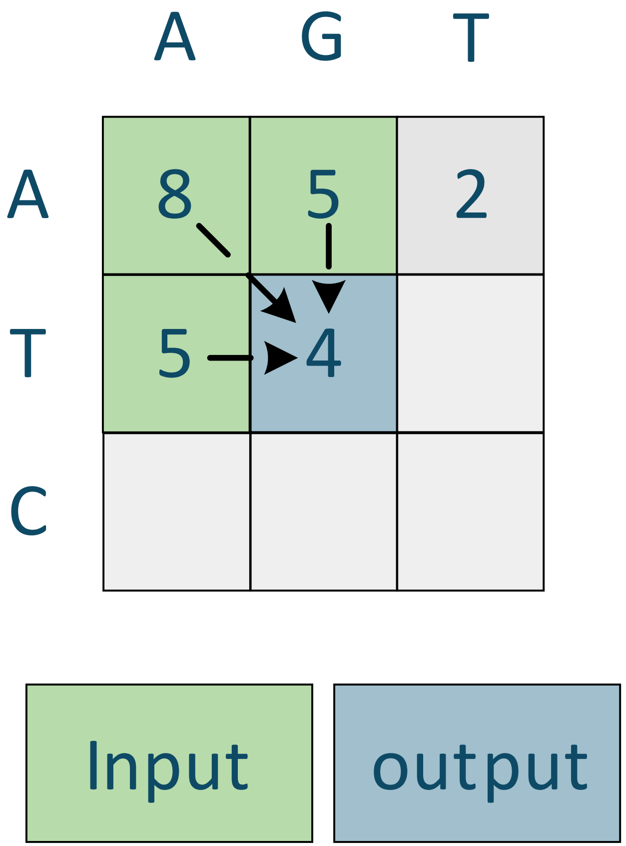
Partial-Matching DPI

- Partial-matching DPI can be used to mitigate signature obfuscation[1]
- It can be expressed as a local sequence alignment problem

[1] J. Drew, T. Moore, and M. Hahsler, “Polymorphic malware detection using sequence classification methods,” in 2016 IEEE Security andPrivacy Workshops (SPW), pp. 81–87, IEEE, May 2016.

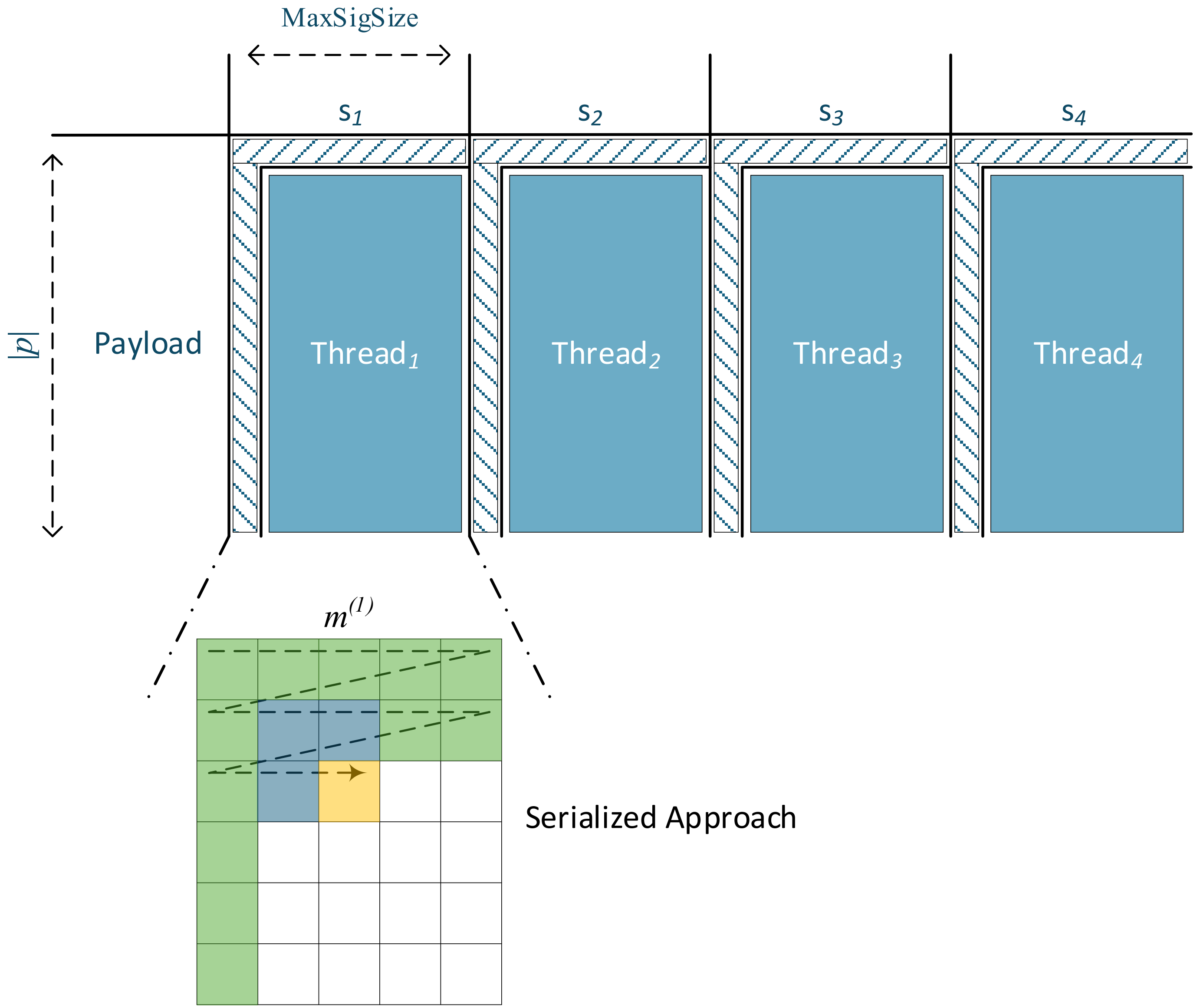
Partial-Matching DPI

- The Smith-Waterman (SW) algorithm is a local sequence alignment algorithm
- It is a DP algorithm to identify similar regions or substrings within two larger strings



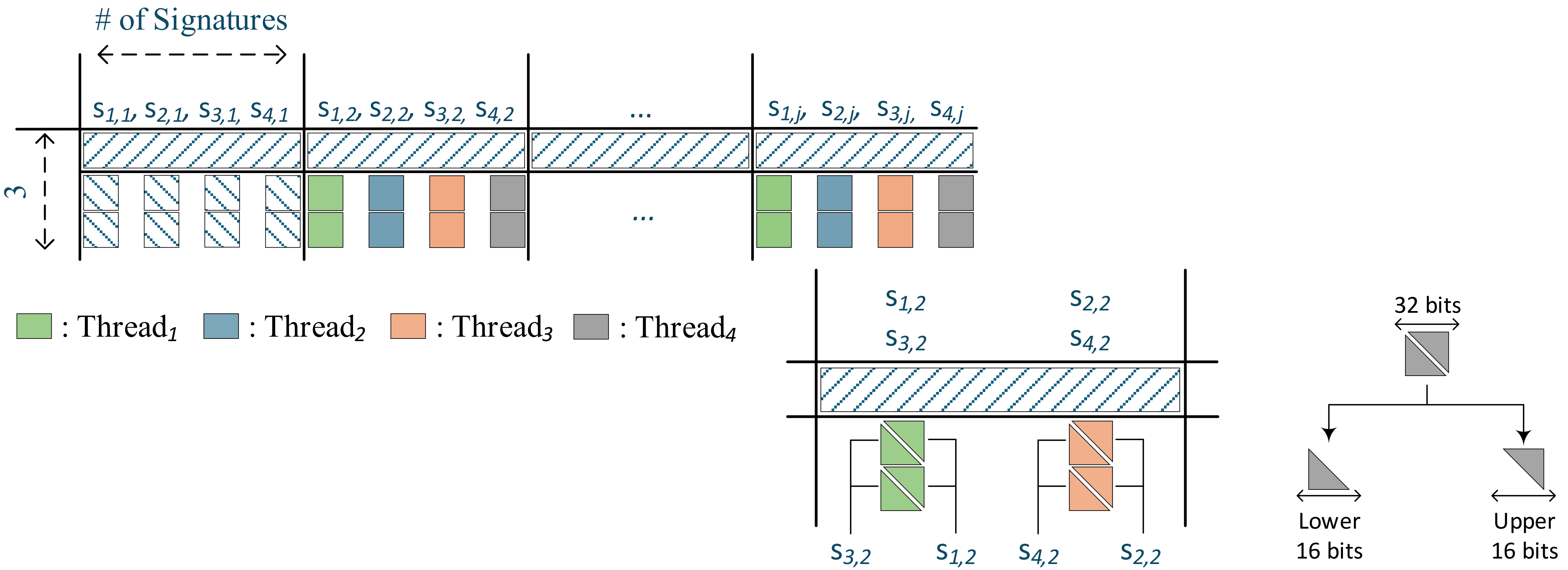
Partial-Matching DPI

- Formulate the DPI with partial matching as a variant of the SW algorithm



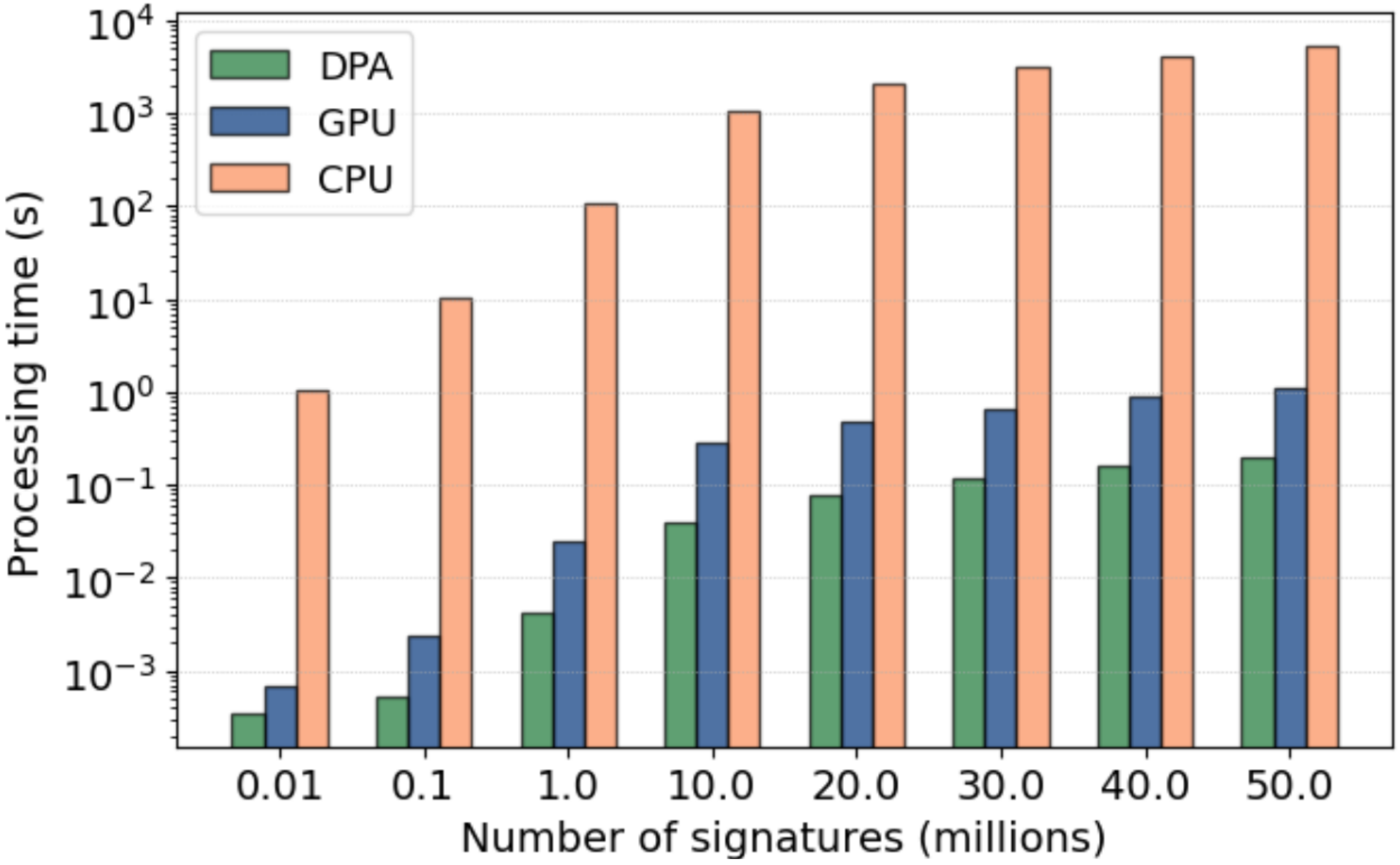
Partial-Matching DPI

- Formulate the DPI with partial matching as a variant of the SW algorithm and use DPA to process two signatures simultaneously per thread



Evaluation – Processing Time

- Evaluating the processing time of the three platforms for different signature counts
- DPA consistently achieved between 5x to 7.5x speedup compared to GPU
- DPA achieved between 3,000x and 26,000x speedup compared to CPU



Conclusion

- DPAs can be used to accelerate multiple networking applications
- For network resilience application, DPA achieves 2x to 3x speedup over GPU and up to 4,600 speedup over CPU
- For partial-matching DPI, DPA achieves 5x to 7.5x speedup over GPU and up to 26,000 speedup over CPU
- DPA consistently demonstrates the smallest energy usage per task

UNIVERSITY OF
SOUTH CAROLINA

Why DPA on GPUs?

	T1	T2	T3	T4	A1	A2	A3	A4	S1	S2
T1	0	∞	∞	∞	2	3	∞	∞	∞	∞
T2	∞	0	∞	∞	3	2	∞	∞	∞	∞
T3	∞	∞	0	∞	∞	∞	2	3	∞	∞
T4	∞	∞	∞	0	∞	∞	3	2	∞	∞
A1	3	4	∞	∞	0	∞	∞	∞	3	4
A2	4	3	∞	∞	∞	0	∞	∞	4	3
A3	∞	∞	3	4	∞	∞	0	∞	3	4
A4	∞	∞	4	3	∞	∞	∞	0	4	3
S1	∞	∞	∞	∞	1	2	1	2	0	∞
S2	∞	∞	∞	∞	2	1	2	1	∞	0

Why DPA on GPUs?

	T1	T2	T3	T4	A1	A2	A3	A4	S1	S2
T1	0	5	∞	∞	2	3	∞	∞	∞	∞
T2	∞	0	∞	∞	3	2	∞	∞	∞	∞
T3	∞	∞	0	∞	∞	∞	2	3	∞	∞
T4	∞	∞	∞	0	∞	∞	3	2	∞	∞
A1	3	4	∞	∞	0	∞	∞	∞	3	4
A2	4	3	∞	∞	∞	0	∞	∞	4	3
A3	∞	∞	3	4	∞	∞	0	∞	3	4
A4	∞	∞	4	3	∞	∞	∞	0	4	3
S1	∞	∞	∞	∞	1	2	1	2	0	∞
S2	∞	∞	∞	∞	2	1	2	1	∞	0

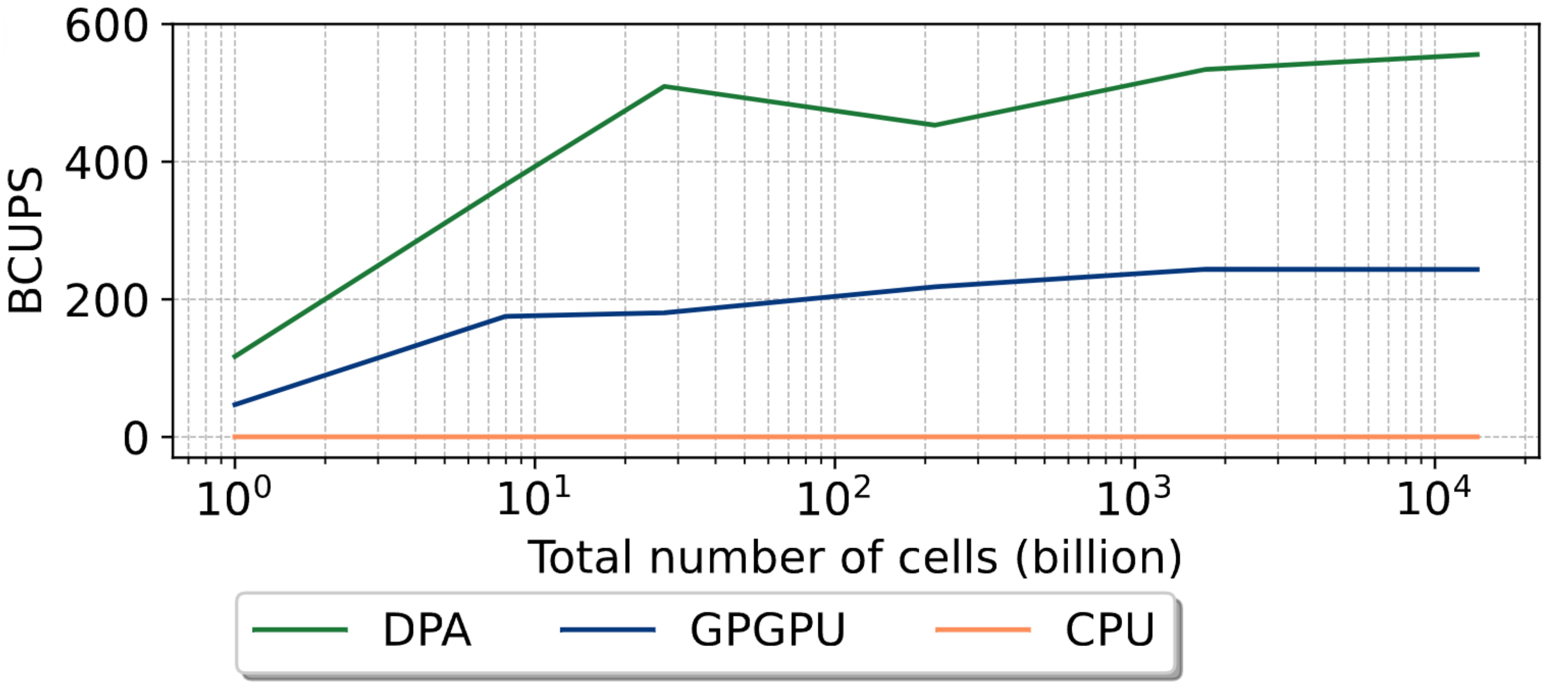
Why DPA on GPUs?

	T1	T2	T3	T4	A1	A2	A3	A4	S1	S2
T1	0	5	∞	∞	2	3	∞	∞	∞	∞
T2	∞	0	∞	∞	3	2	∞	∞	∞	∞
T3	∞	∞	0	∞	∞	∞	2	3	∞	∞
T4	∞	∞	∞	0	∞	∞	3	2	∞	∞
A1	3	4	∞	∞	0	∞	∞	∞	3	4
A2	4	3	∞	∞	∞	0	∞	∞	4	3
A3	∞	∞	3	4	∞	∞	0	∞	3	4
A4	∞	∞	4	3	∞	∞	∞	0	4	3
S1	∞	∞	∞	∞	1	2	1	2	0	∞
S2	∞	∞	∞	∞	2	1	2	1	∞	0

	T1	T2	T3	T4	A1	A2	A3	A4	S1	S2
T1	0	5	∞	∞	2	3	∞	∞	5	6
T2	5	0	∞	∞	3	2	∞	∞	6	7
T3	∞	∞	0	∞	∞	∞	2	3	∞	∞
T4	∞	∞	∞	0	∞	∞	3	2	∞	∞
A1	3	4	∞	∞	0	∞	∞	∞	3	4
A2	4	3	∞	∞	∞	0	∞	∞	4	3
A3	∞	∞	3	4	∞	∞	0	∞	3	4
A4	∞	∞	4	3	∞	∞	∞	0	4	3
S1	4	5	∞	∞	1	2	1	2	0	∞
S2	5	6	∞	∞	2	1	2	1	∞	0

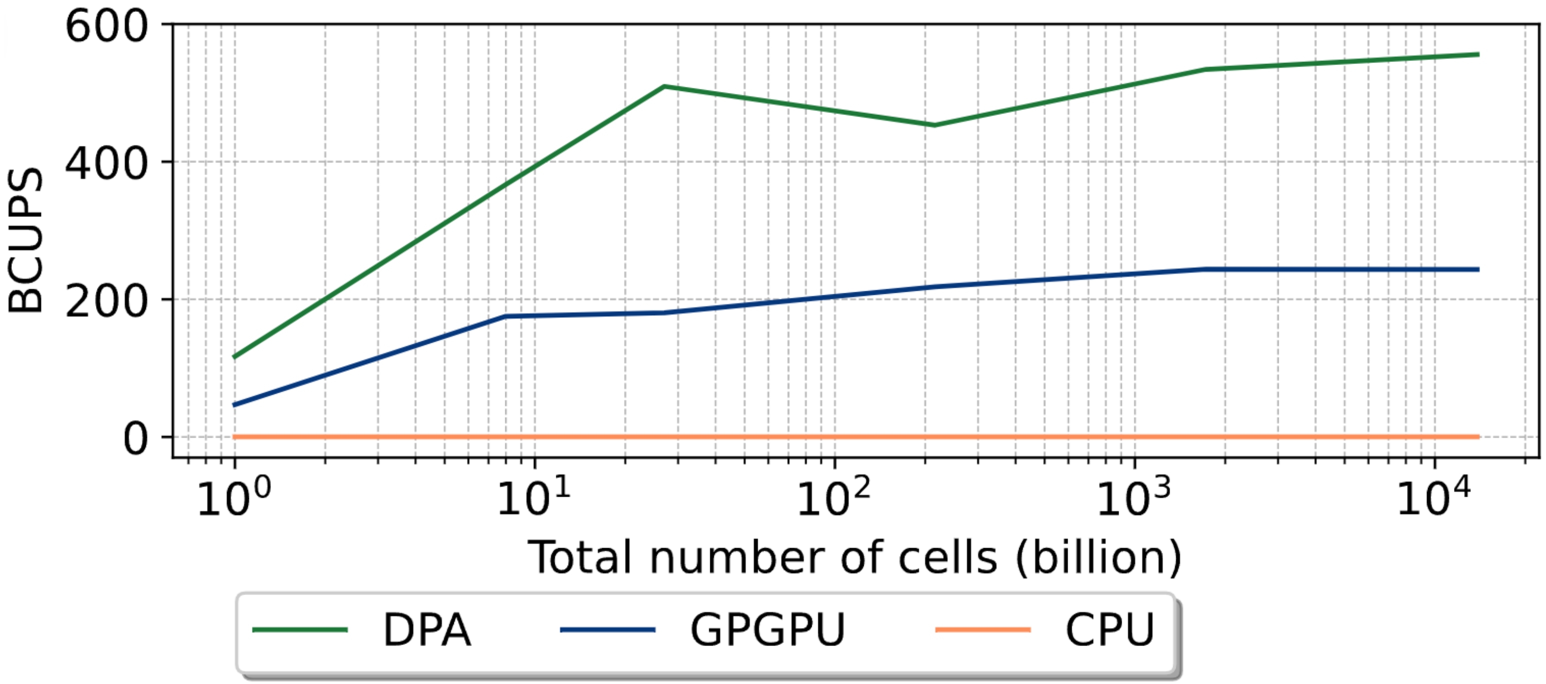
Evaluation Results: Throughput

- This experiment evaluates the throughput calculated as the total number of cell updates ($|V|^3$) divided by the processing time
- DPA throughput increases with the problem size
- DPA achieved peak performance at 555.79 BCUPS with 24K nodes



Evaluation Results: Throughput

- GPGPU showed a similar increasing trend as the DPA
- CPU sustained around 0.116 BCUPS



Evaluation Results: Energy Consumption

- This experiment measures the energy consumed by each platform using the NVIDIA NVML library for the GPUs and intel-rapl for the CPU
- Energy consumption is calculated as the product of average power usage and execution time