Accelerating Polynomial Multiplication for Homomorphic Encryption on GPUs

Kaustubh Shivdikar[1], Gilbert Jonathan[3], Evelio Mora[4], Neal Livesay[1], Rashmi Agrawal[2], Ajay Joshi[2], Jose L. Abellan[4], John Kim[3], David Kaeli[1]

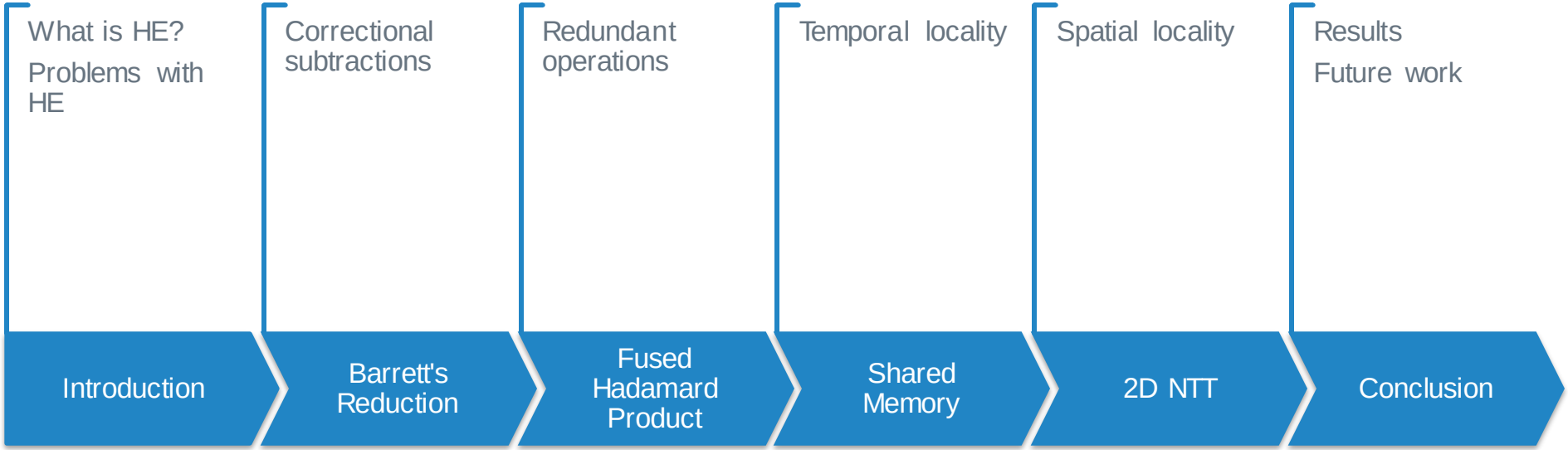
1 NORTHEASTERN UNIVERSITY

2 BOSTON UNIVERSITY

3 KAIST — Korea Advanced Institute of Science and Technology — 1971

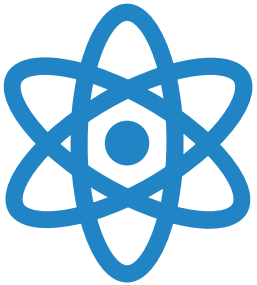
4 UCAM Universidad Católica de Murcia

Outline


What is HE?
Problems with HE
Correctional subtractions
Redundant operations
Temporal locality
Spatial locality
Results
Future work
Introduction
Barrett's Reduction
Fused Hadamard Product
Shared Memory
2D NTT
Conclusion

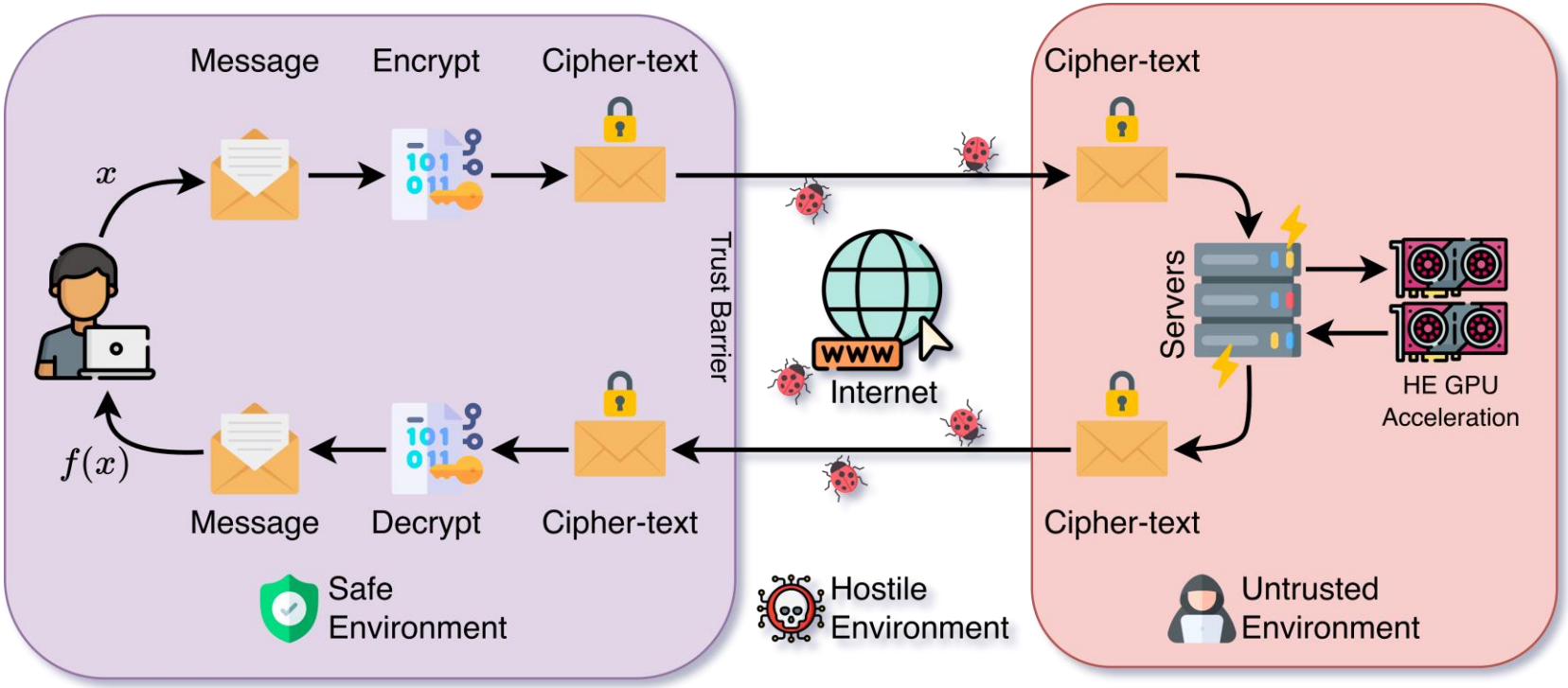
What is Homomorphic Encryption?

HE is a type of encryption that allows computation to be run on encrypted operands

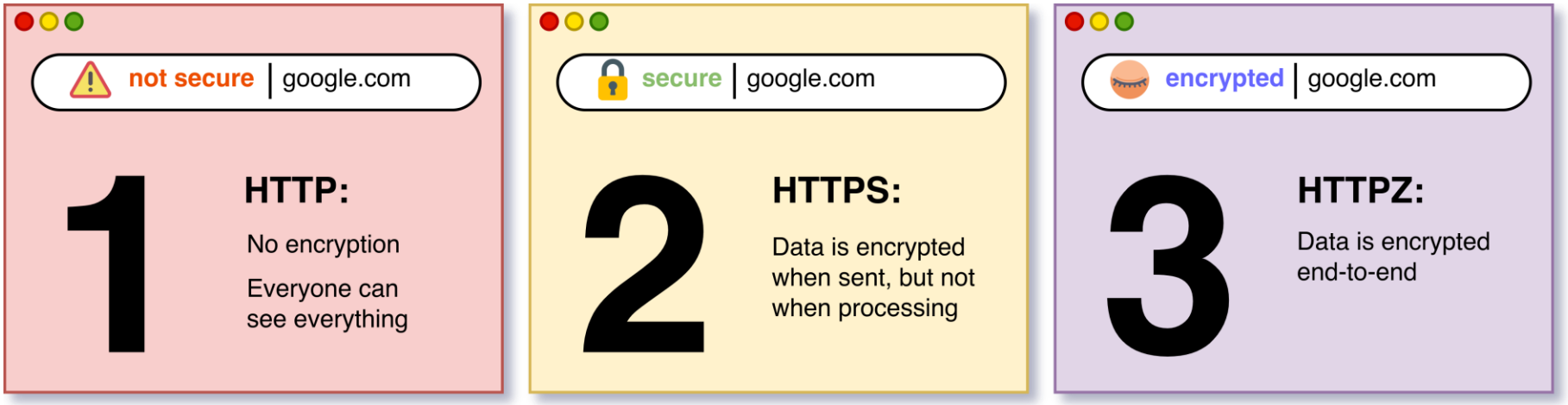


HE schemes are lattice-based, making them quantum resistant

What is Homomorphic Encryption?



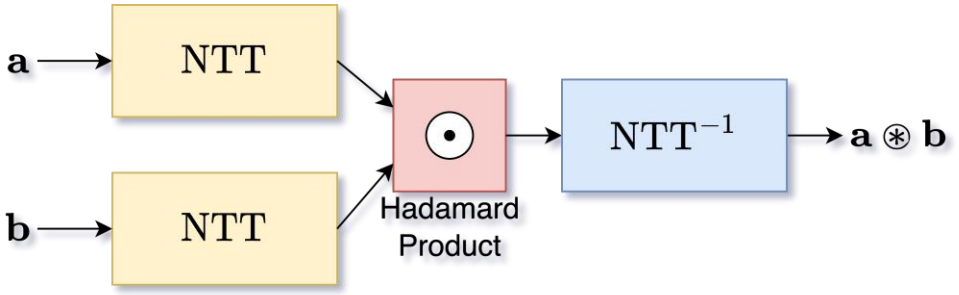
"HE could give rise to a new internet protocol, **HTTPZ**, that would standardize end-to-end encryption and replace HTTPS as the default protocol"



--- By R. Hindi, "*People shouldn't care about privacy*", ZAMA, Aug 2021

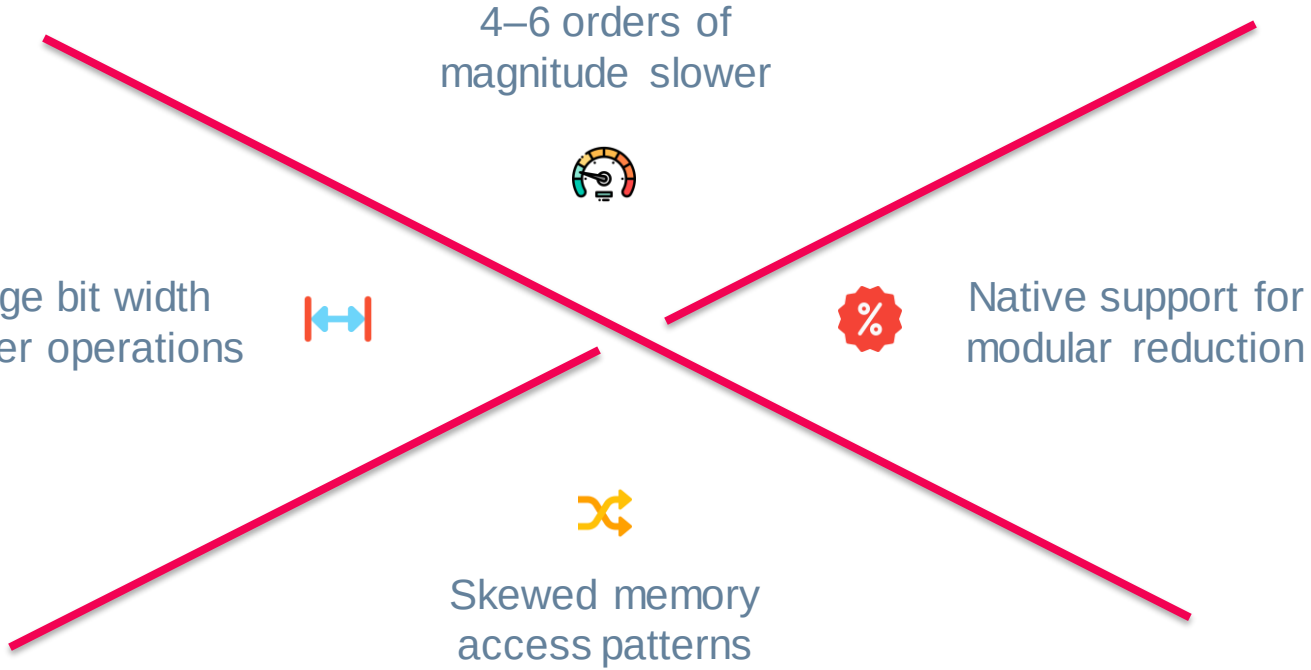
HE Bottlenecks

- **Polynomial multiplication** is the key bottleneck for lattice-based cryptography and HE
- Poly-mult is typically implemented with Number Theoretic Transform (**NTT**)
- NTT relies heavily on **modular reduction** operation



Polynomial Multiplication

Problems with HE



Contributions

Optimized Barrett's Reduction

- Fewer Correctional Subtractions

Fused Hadamard Product

- Reduces Operational Complexity

Persistent Shared Memory

- Increased Temporal Locality

Mixed Radix 2D NTT

- Increased Spatial Locality

Optimized Barrett's Reduction

- NTT is dominated by **modulo** operation
- Modulo computation involves expensive **division** operation
- **Barrett's reduction** replaces division with a set of bit-shift and multiplication operations

$$\text{Remainder} = x \,\%\, q$$

$$\text{Remainder} = x - \left(\left\lfloor \frac{x}{q} \right\rfloor \times q \right)$$

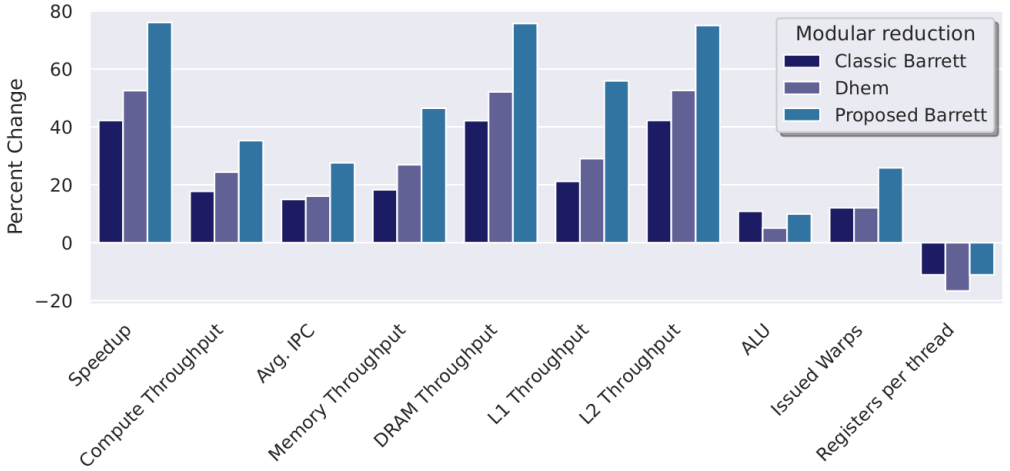
Algorithm 2 Classical Barrett reduction

Require: $m = \text{len}(q) \leq \beta - 2,\ 0 \leq x < 2^{2m},\ \mu = \lfloor \frac{2^{2m}}{q} \rfloor$
Ensure: $\text{rem} = x \bmod q$

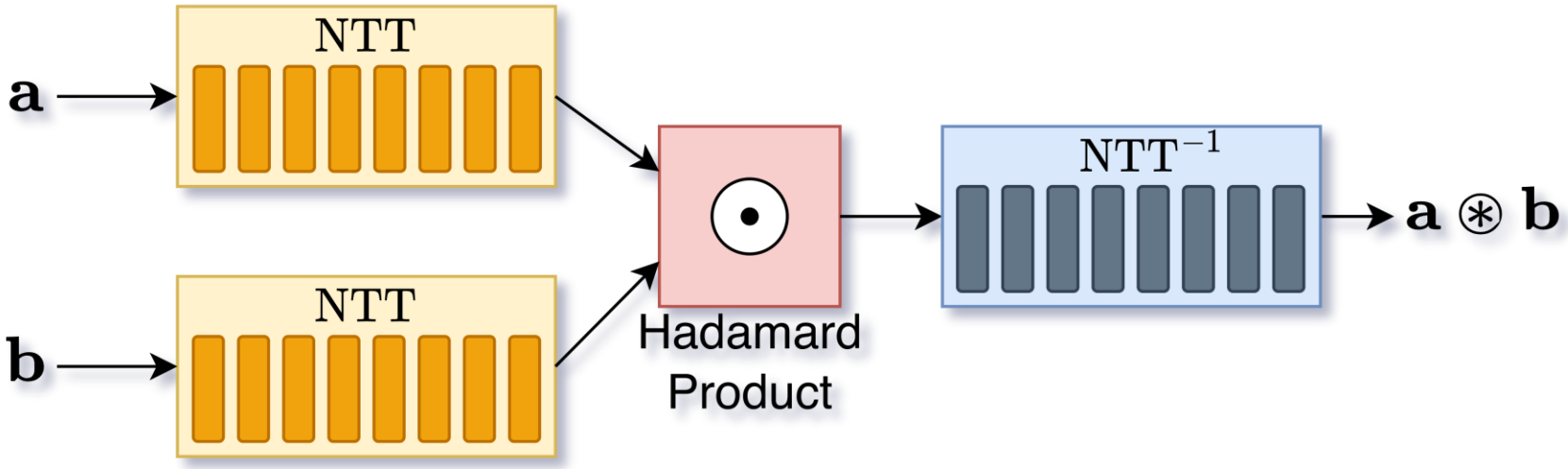
1: $c \leftarrow x \gg (m-1)$
2: $\text{quot} \leftarrow (c \times \mu) \gg (m+1)$
3: $\text{rem} \leftarrow x - \text{quot} \times q$
4: **if** $\text{rem} \geq q$ **then**
5: $\quad \text{rem} \leftarrow \text{rem} - q$
6: **if** $\text{rem} \geq q$ **then**
7: $\quad \text{rem} \leftarrow \text{rem} - q$
8: **return** rem

Optimized Barrett's Reduction

- Proposed Barrett's reduction reduces number of correctional subtractions
- Future Work: Natively supported modular reduction with hardware implementation

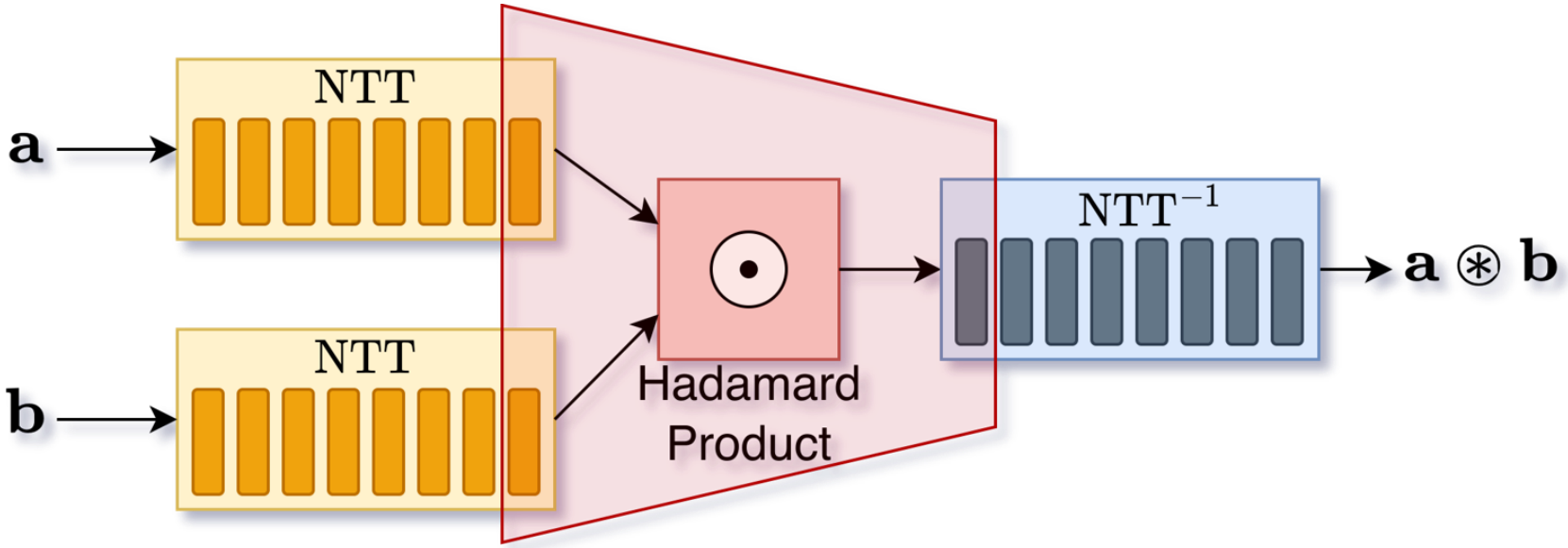


Fused Hadamard Product



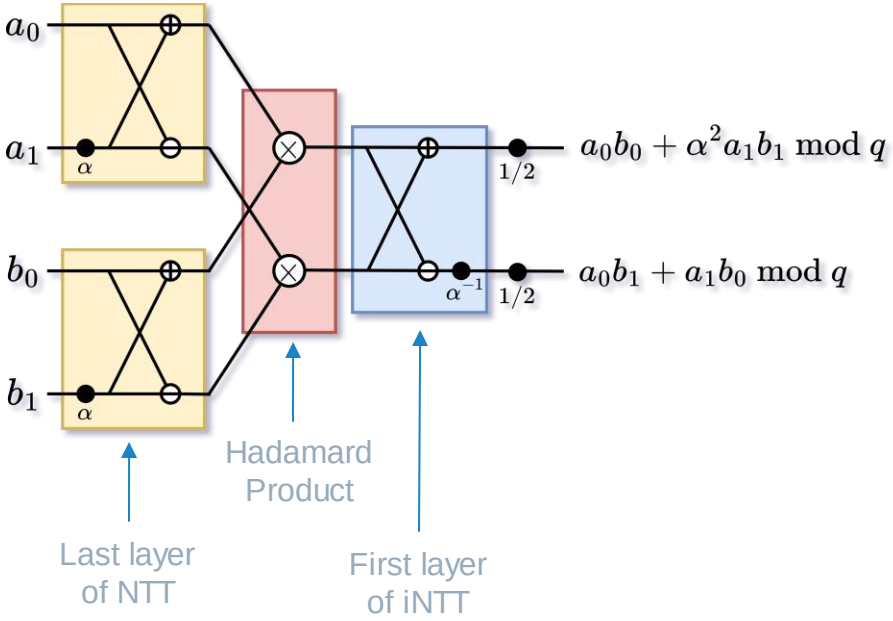
Polynomial Multiplication

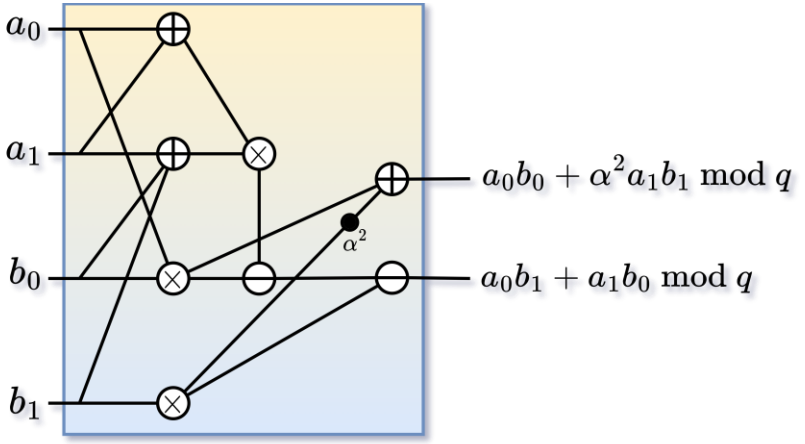
Fused Hadamard Product



Polynomial Multiplication

Fused Hadamard Product

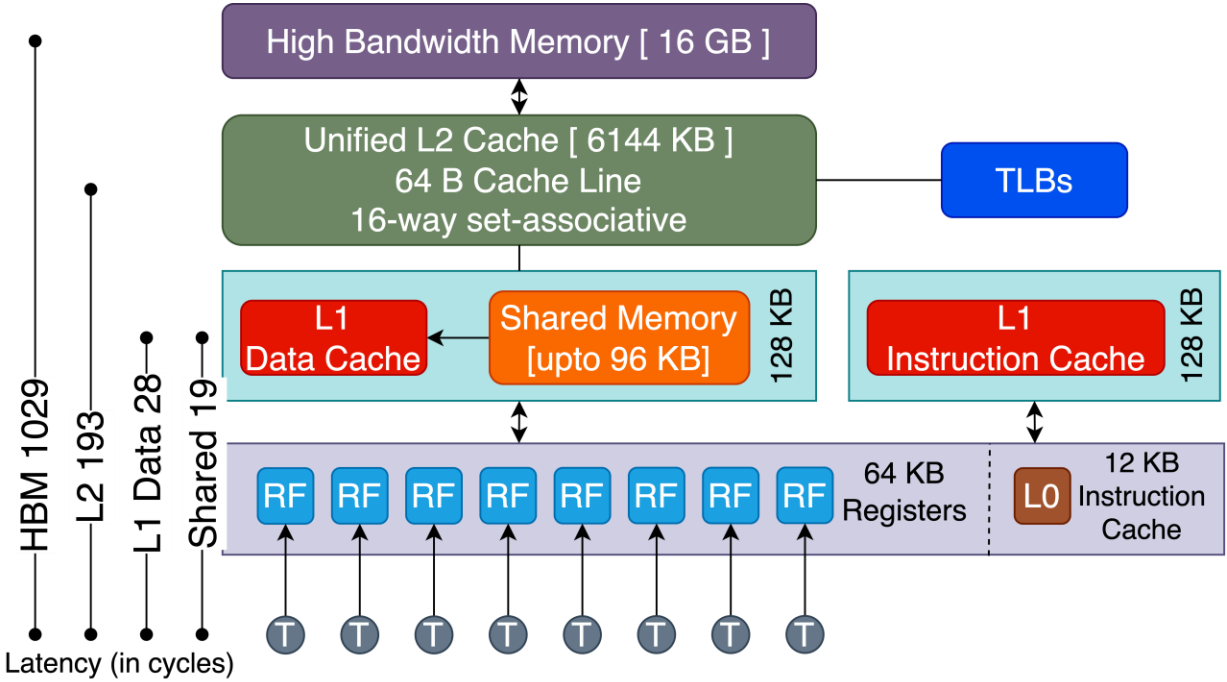




Fused NTT, iNTT, and Hadamard Product

Persistent Shared Memory

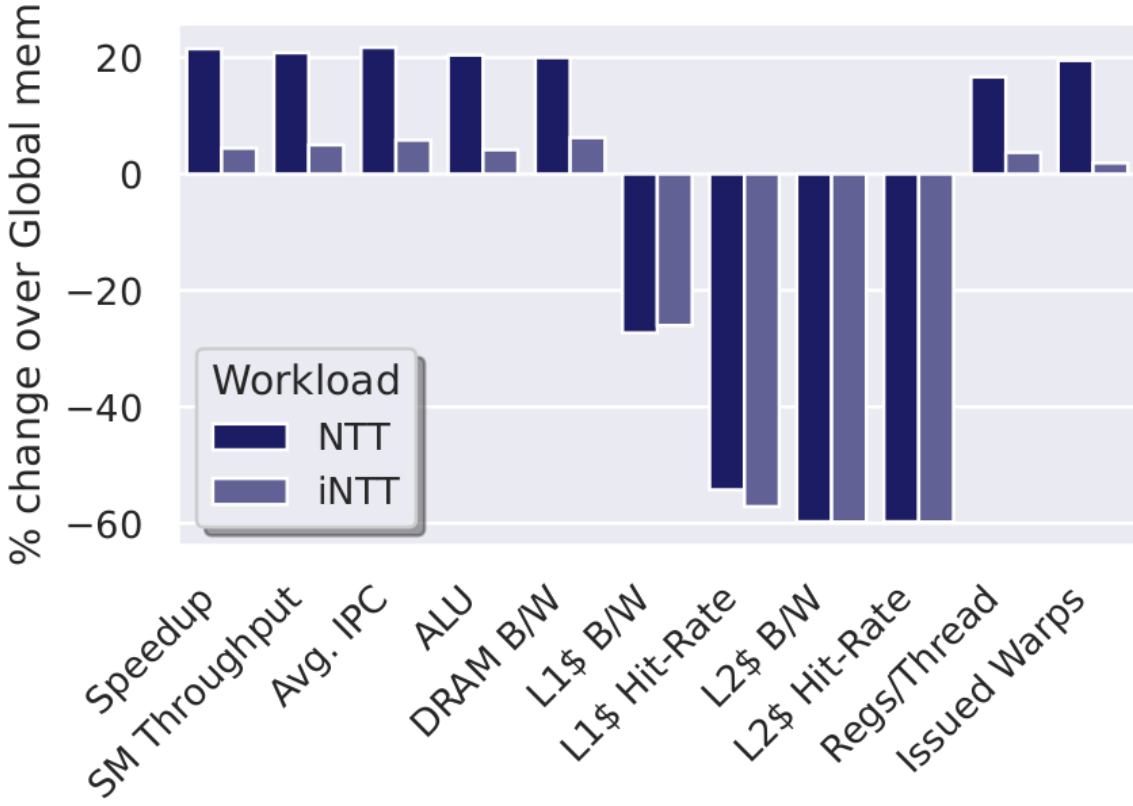
- NTT is a **memory-bound** kernel
- Each stage of NTT generates **intermediate results** for subsequent stages
- Intermediate results of each stage are **cached** on shared memory
- Removes **redundant** global memory accesses



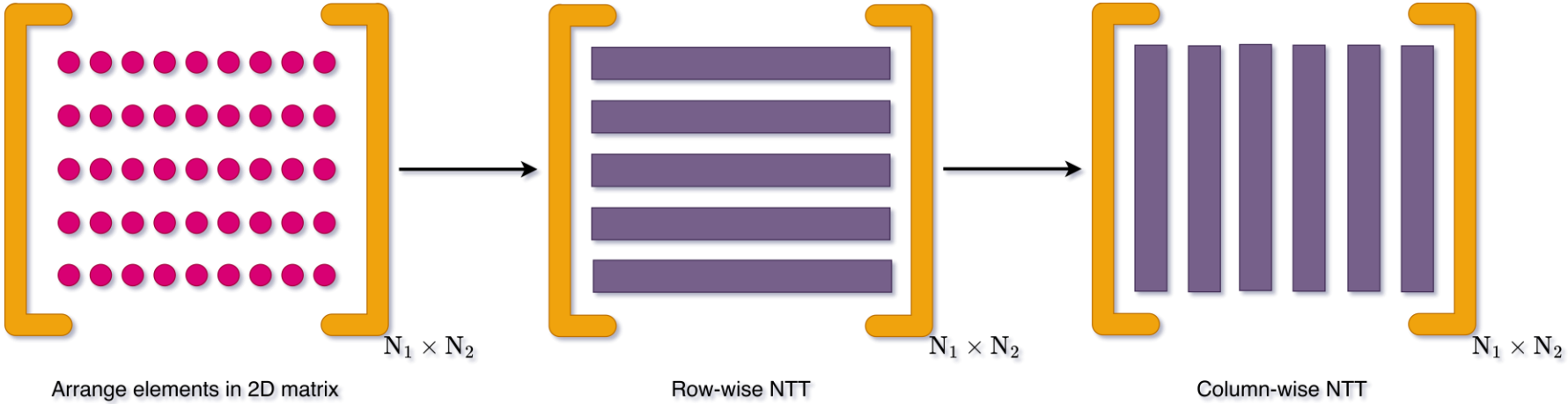
V100 GPU Memory Hierarchy

Persistent Shared Memory

- Shared memory use provides **25%** speedup over global memory
- **L1** and **L2** cache memory pressure drops significantly
- Only works for NTT sizes that fit in shared **memory size** ($N<2^{11}$)

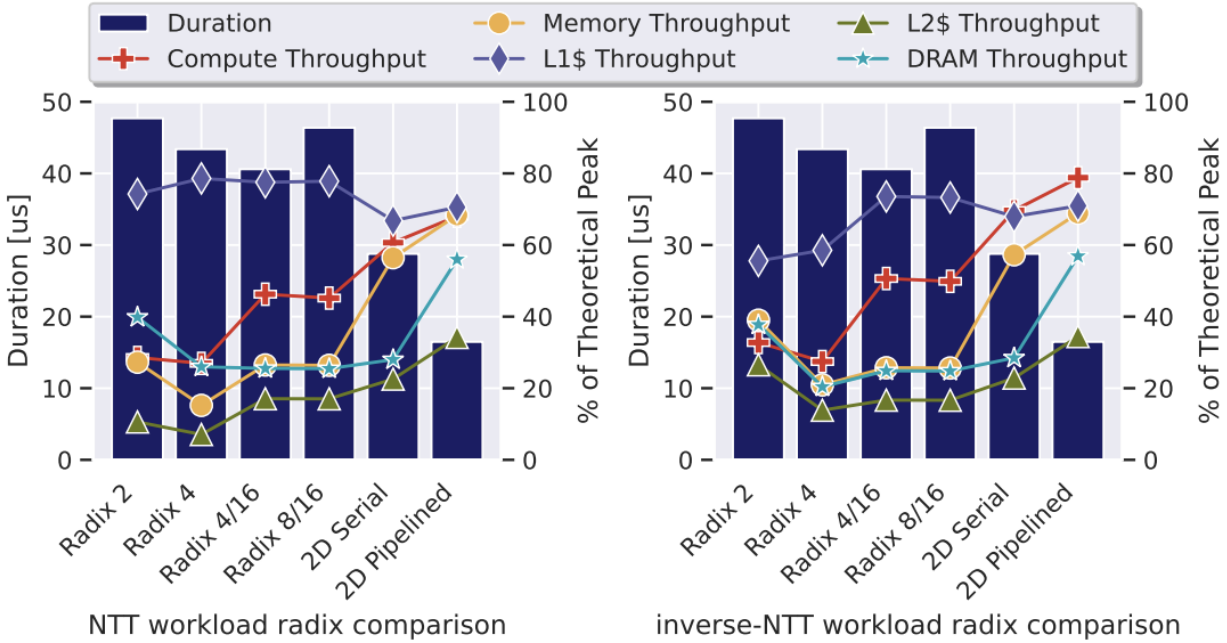


Mixed Radix 2D NTT



Mixed Radix 2D NTT

- 2D NTT increases **compute** costs
- Reduces **memory** pressure
- Preserves **spatial locality**
- Row-wise and Column-wise NTT can be **pipelined**

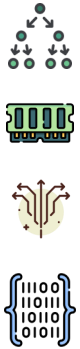


Concluding Remarks

HE is a **popular** memory-intensive workload with high computational demands

Explored key **bottlenecks** in HE

- Algorithmic improvements
- Low-level kernel improvements



Presented four optimizations

Achieved speedup

- CPU: **123.13x**
- GPU: **2.37x**

Future Work / Architectural Feature Requests

- Larger integer **bit width** support for GPU
- Native **modular reduction** support

Thank you!

Any questions?

--- This work was supported in part by the Institute for Experiential AI, the Harold Alfond Foundation, the NSF IUCRC Center for Hardware and Embedded Systems Security and Trust (CHEST), the RedHat Collaboratory, and project grant PID2020-112827GBI00 funded by MCIN/AEI/10.13039/501100011033.