Let-Me-In: (Still) Employing In-pointer Bounds Metadata for Fine-grained GPU Memory Safety

Jaewon Lee[†], Euijun Chung[†], Saurabh Singh[†], Seonjin Na[†], Yonghae Kim[‡], Jaekyu Lee[§], and Hyesoon Kim[†]

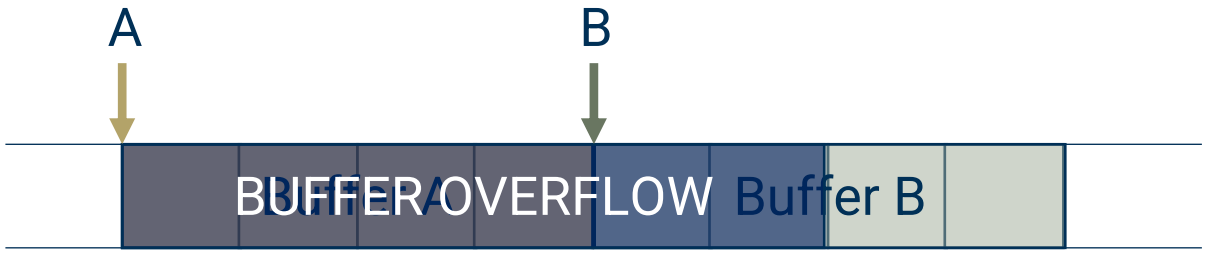
Georgia Institute of Technology[†], Arm[‡], Intel[§]

arm intel Georgia Tech

Memory Safety

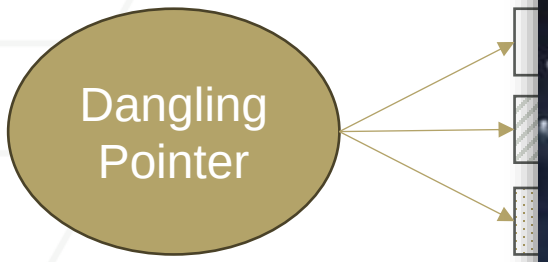
- **Provide a protection** against the **unauthorized memory access** which leads to system vulnerability.
- Ex.1) Buffer overflow.
  - e.g. Stack smashing attack

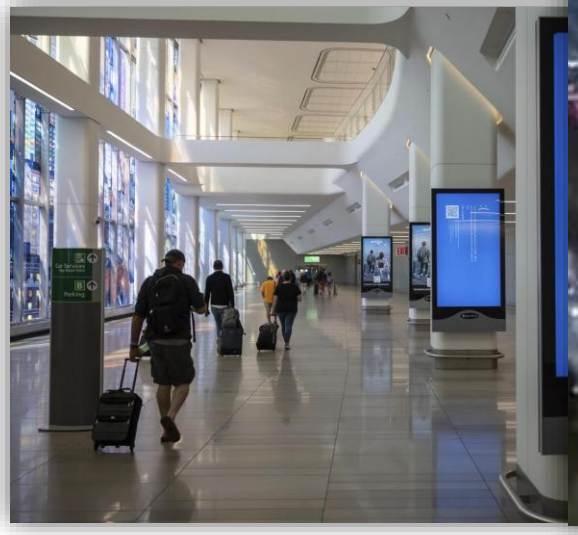
Allocate adjacent buffers A and B.
Trigger the buffer overflow on buffer A.
Read on B will obtain the **incorrect** value.



Memory Sa

- Ex.2) Dangling P





Arise of The Concern on GPU Security

Previous GPU

Passive Device	Limited Usage	Generic Data

Research on GPU Memory Safety is actively going on!

Now

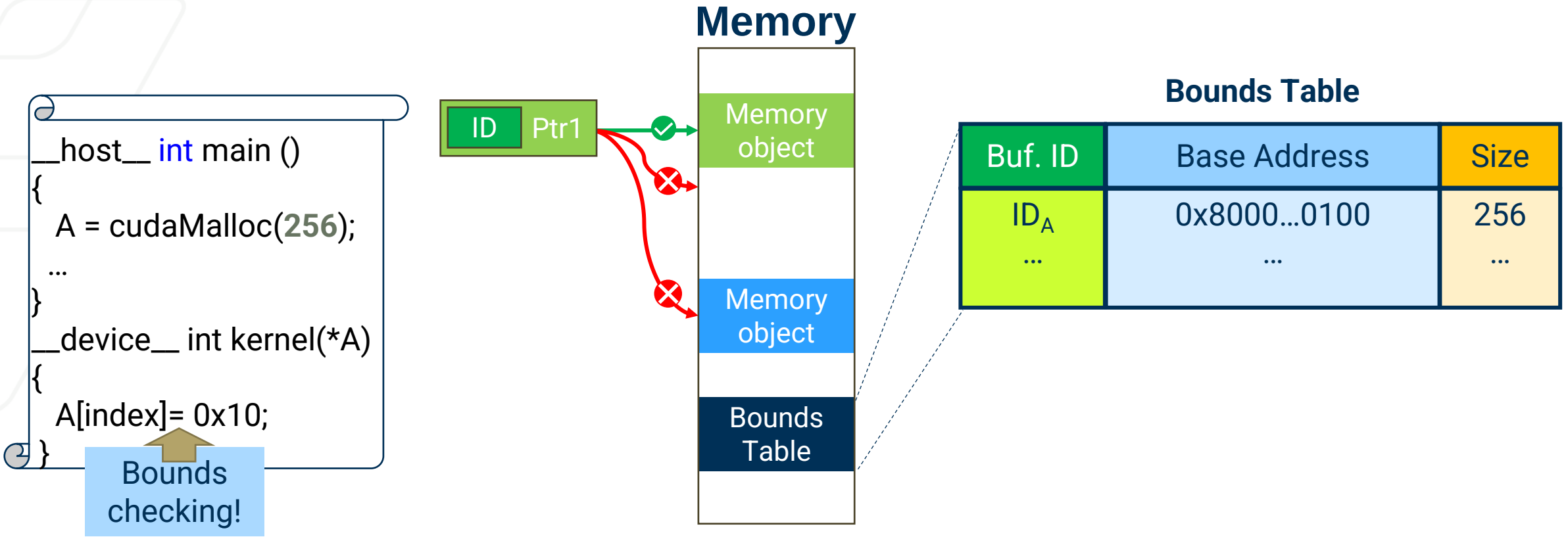
Active Device	General Purpose	Private Data

Now GPUs Need New Memory Safety Solution

- Previous studies focused on **per-kernel** memory safety solution.
  - Memory safety on buffers allocated by host, accessed by threads
- Not sufficient for recent attacks
  - e.g. Stack-smashing attacks on GPUs[1].
- Need a **per-thread** memory safety solution.
  - Protection on memory chunks allocated and used by each thread at **RUNTIME**

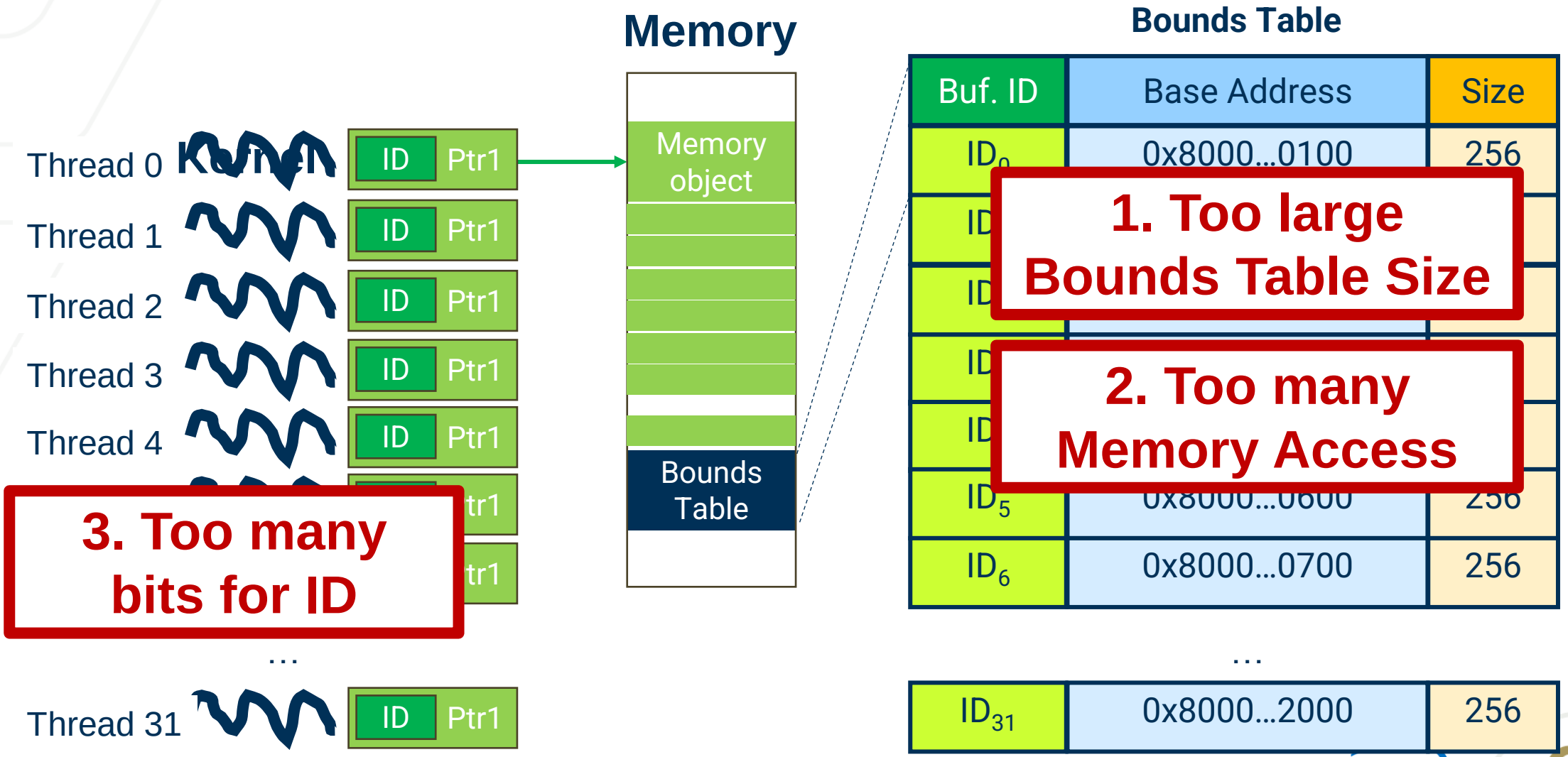
[1] Guo, Yanan, Zhenkai Zhang, and Jun Yang. "GPU Memory Exploitation for Fun and Profit." In *33rd USENIX Security Symposium*

Per-kernel Solution: Pointer Tagging Method



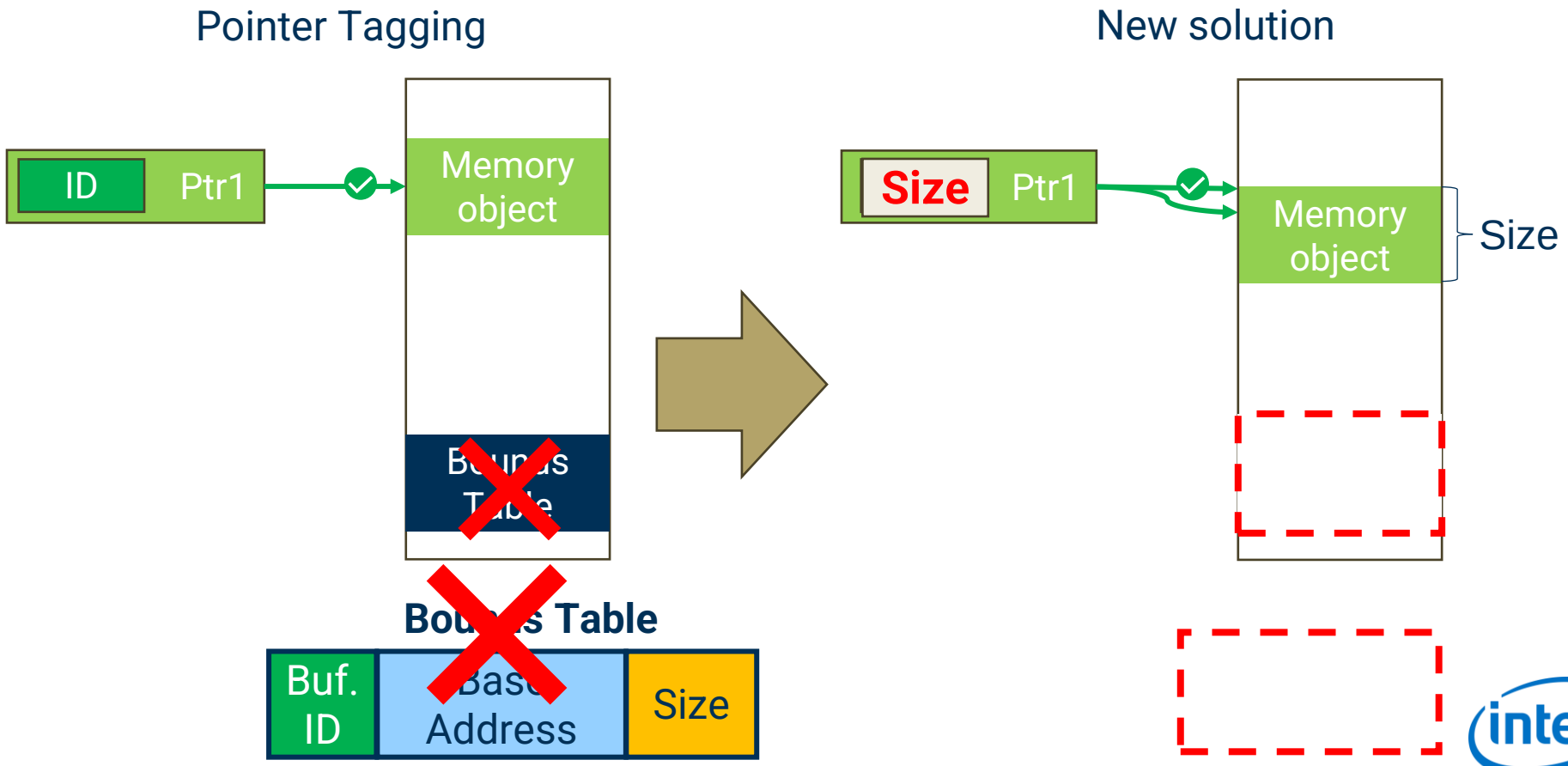
- **Buffer ID** to identify the memory chunk.
- **Buffer Base Address** to specify the starting address of the buffer.
- **Buffer Size** to store the size to determine the end address.
- Utilize **unused** upper bits in pointer as a tag(= **Buffer ID)** storage**.**

Issues in Per-thread Pointer Tagging Method



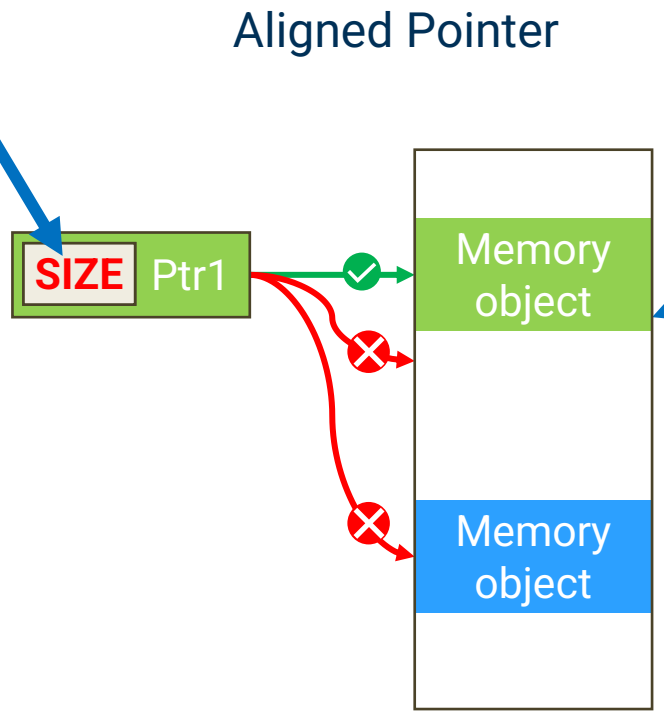
(1) Let's Remove Bounds Table.

- Immune to memory access issues.
- Verify the bounds per thread only with its size information instead of ID



There are (Still) Problems

2) How to **minimize** the size bits?



1) How can we check bounds **without BASE address?**

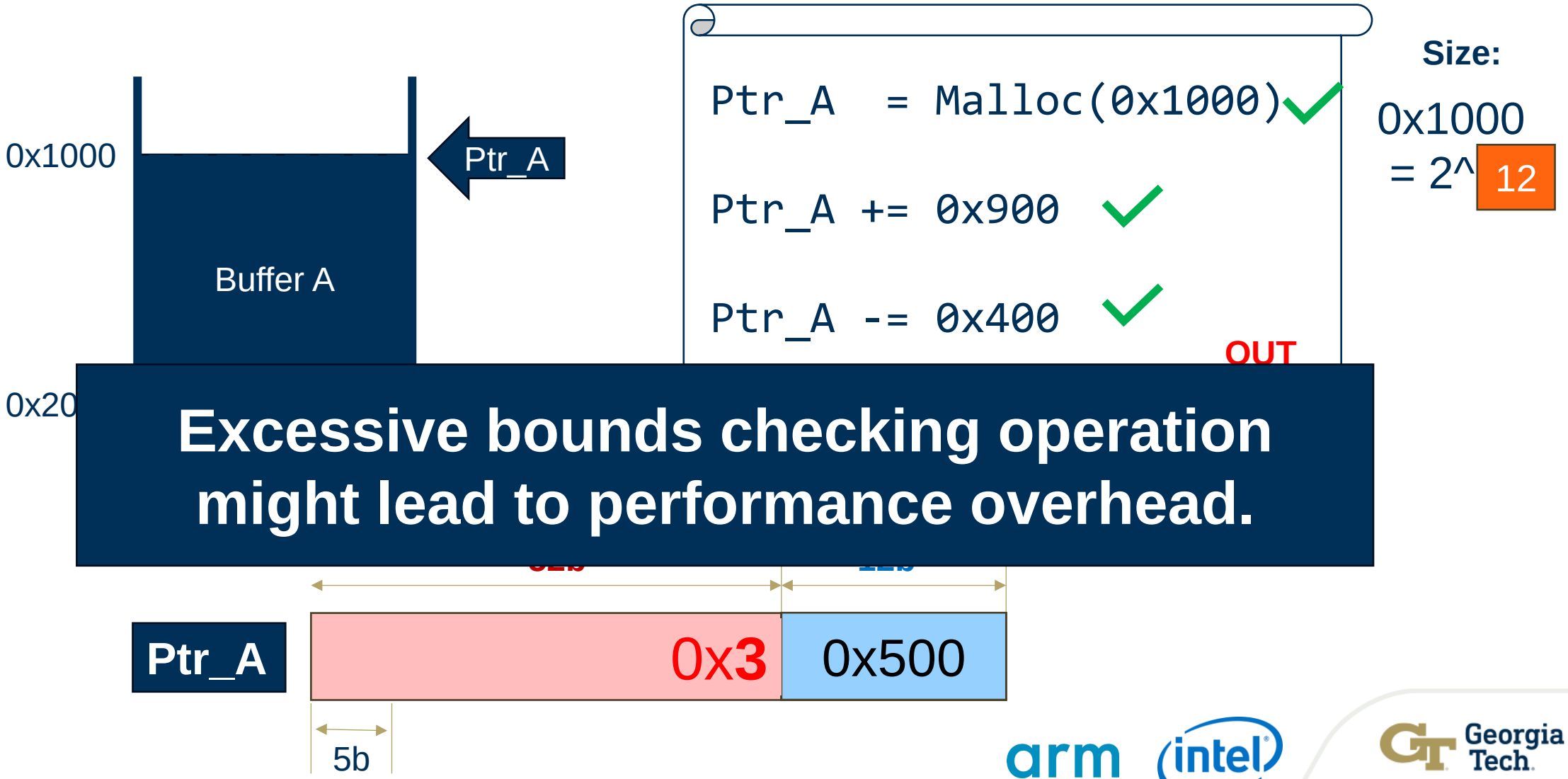
Need a New Approach: LMI

(2) Let buffer be aligned to a power-of-two size

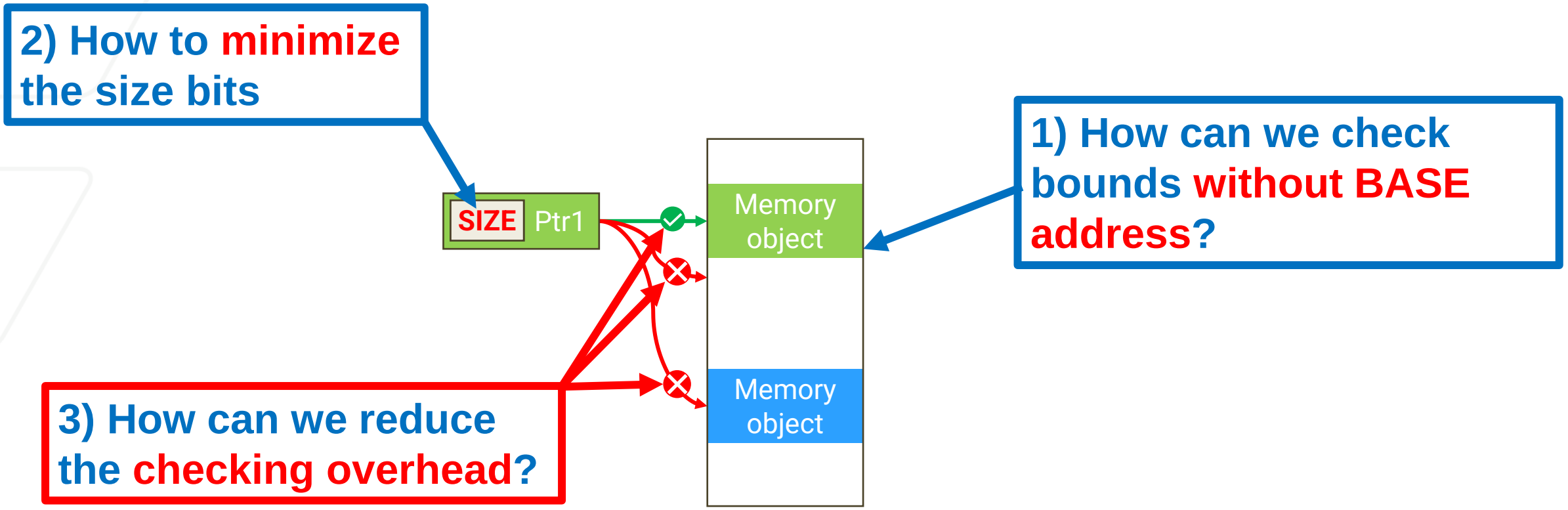
- **We can remove BASE address.**
  - Keep **base address information in address itself**.
    - Make the base address part unmodifiable.
    - Check if it is modified or not.

- **We can minimize the size bits.**
  - Store the **extent part** only.
    - 32bit size can be represented within 5 bit.

Memory Checking Without Base Address Info.



Feasibility of All-Time Bounds Checking

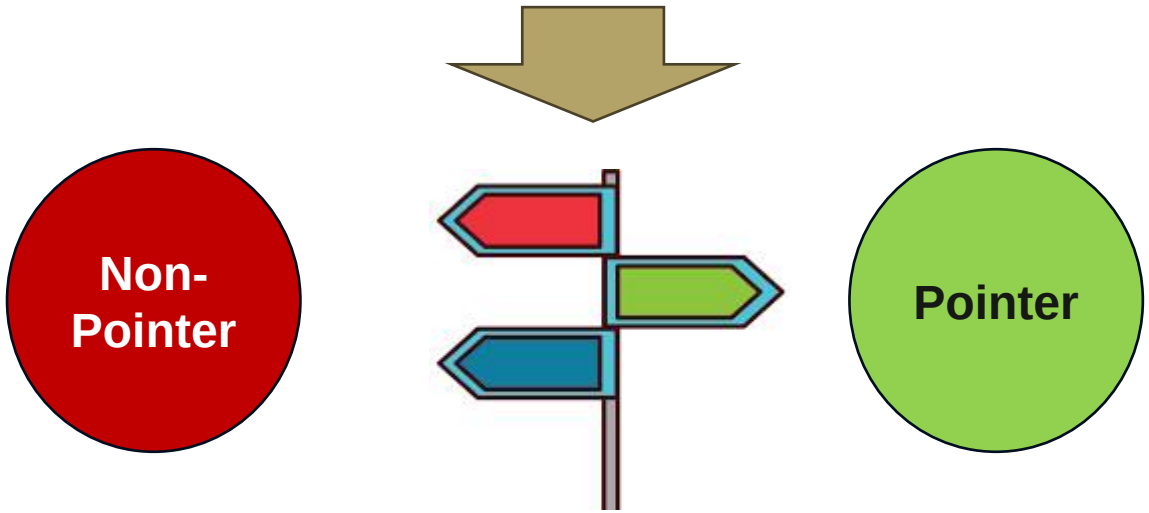


Not all Arithmetic Operation are Address Operation.

- Requires compiler support for instruction marking

```
ADD I1, I2, I3
ADD P1, P1,     0x0
ADD P2, P2, -0x100
```

I: Integer
P: Pointer

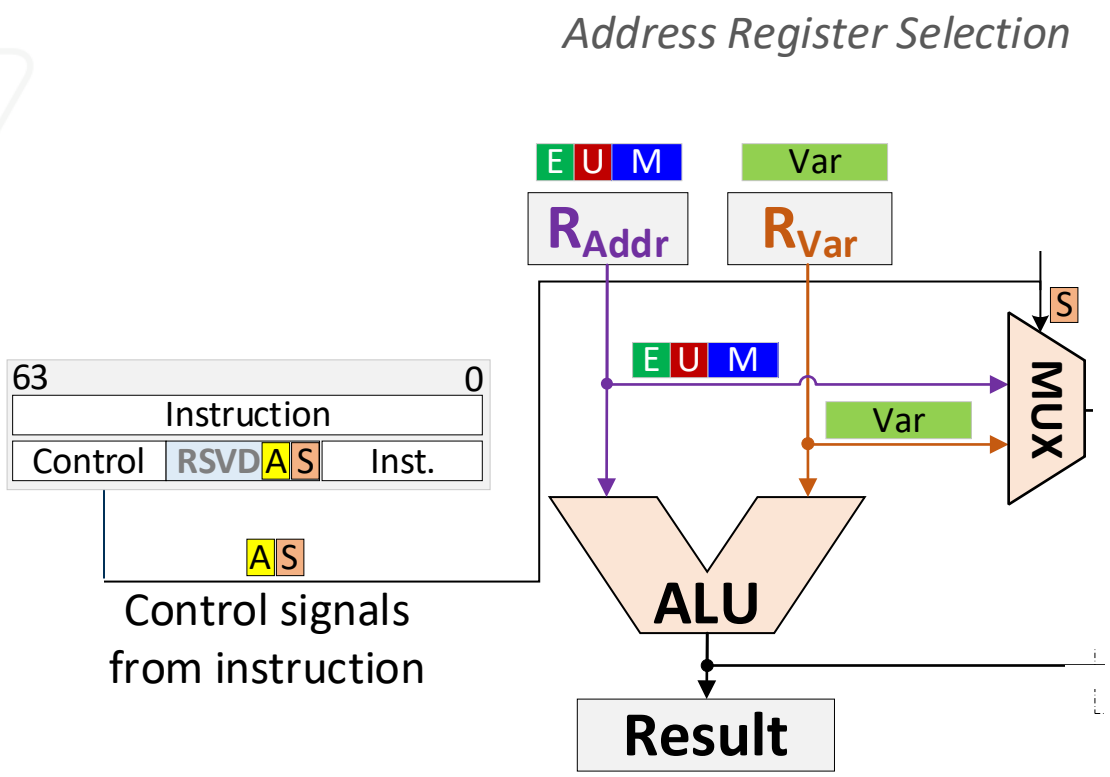


Compiler Analysis

arm intel Georgia Tech

Low Overhead Overflow Checking Unit (OCU)

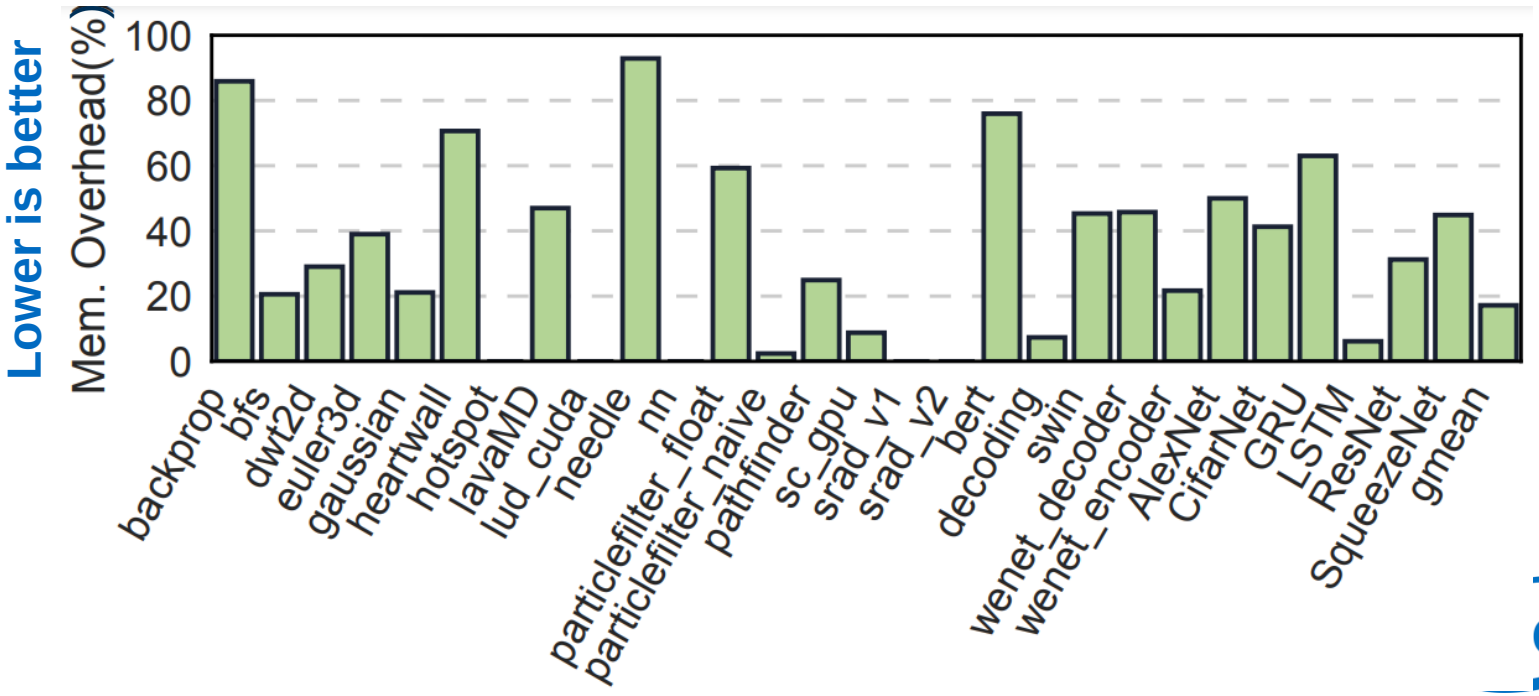
- Utilizing marking bits in the instruction set for static-time analysis.
- Hardware checking unit to operate within a single cycle.



A : Activation
S : Selection
E : Extent
U : Unmodifiable
M : Modifiable
: *Optional*

GPUs: More Room to Maneuver Than CPUs

1. Lower fragmentation overhead from power-of-two alignment.
   - Fragmentation is a critical issue in CPU programs.
   - Fortunately, the GPU fragmentation is low enough (19.7%)
     - This is a key trait of GPU programs, which mostly align data sizes to powers of two.



GPUs: More Room to Maneuver Than CPUs

1. Lower fragmentation overhead from power-of-two alignment.
   - Fragmentation is a critical issue in CPU programs.
   - Fortunately, the GPU fragmentation is low enough (19.7%)
     - This is a key trait of GPU programs, which mostly align data sizes to powers of two.
2. Simpler memory operation
   1. (Almost) no immediate number assign to pointers
   2. (Almost) no pointer load/store

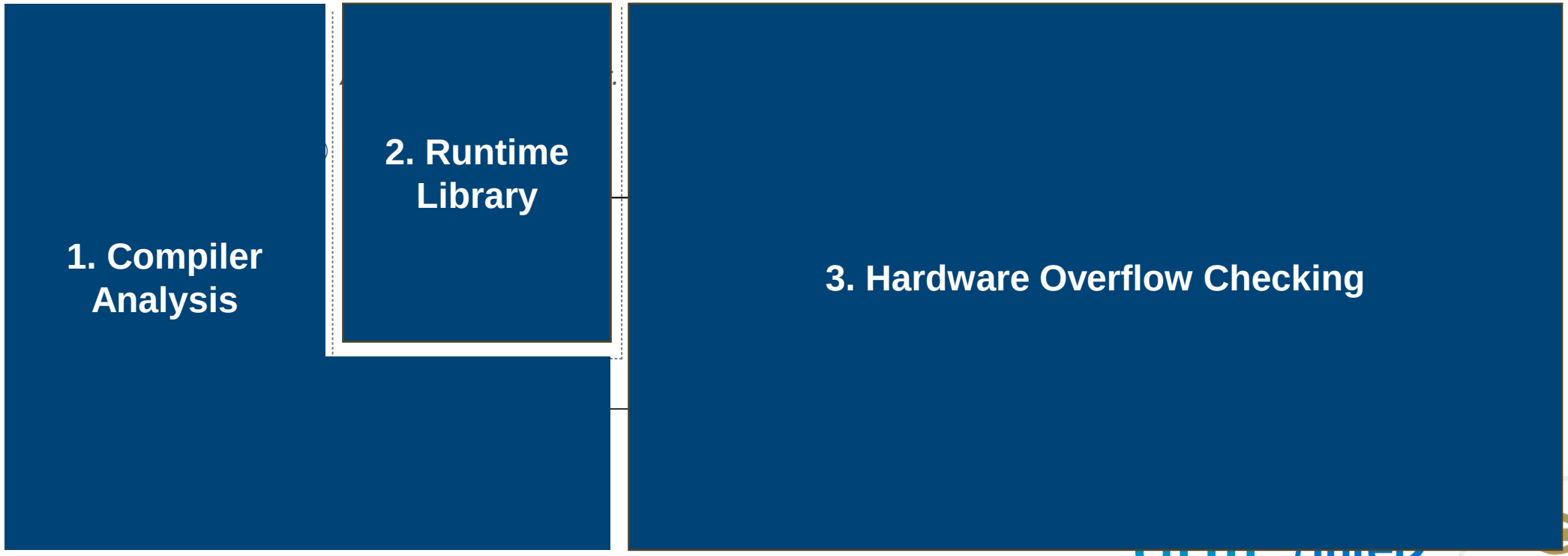
More In Our Paper!

- Compiler and Runtime Library Support.
- Hardware Implementation on Vortex Project[1] and Power/Area Analysis.
- Temporal Safety.

[1] Tine, Blaise, Krishna Praveen Yalamarthy, Fares Elsabbagh, and Kim Hyesoon. "Vortex: Extending the RISC-V ISA for GPGPU and 3D-graphics." In *MICRO-54*

LMI Recap

1. Compiler Analysis
2. Runtime library support
3. Hardware Overflow Checking Unit (OCU)



Evaluation Method

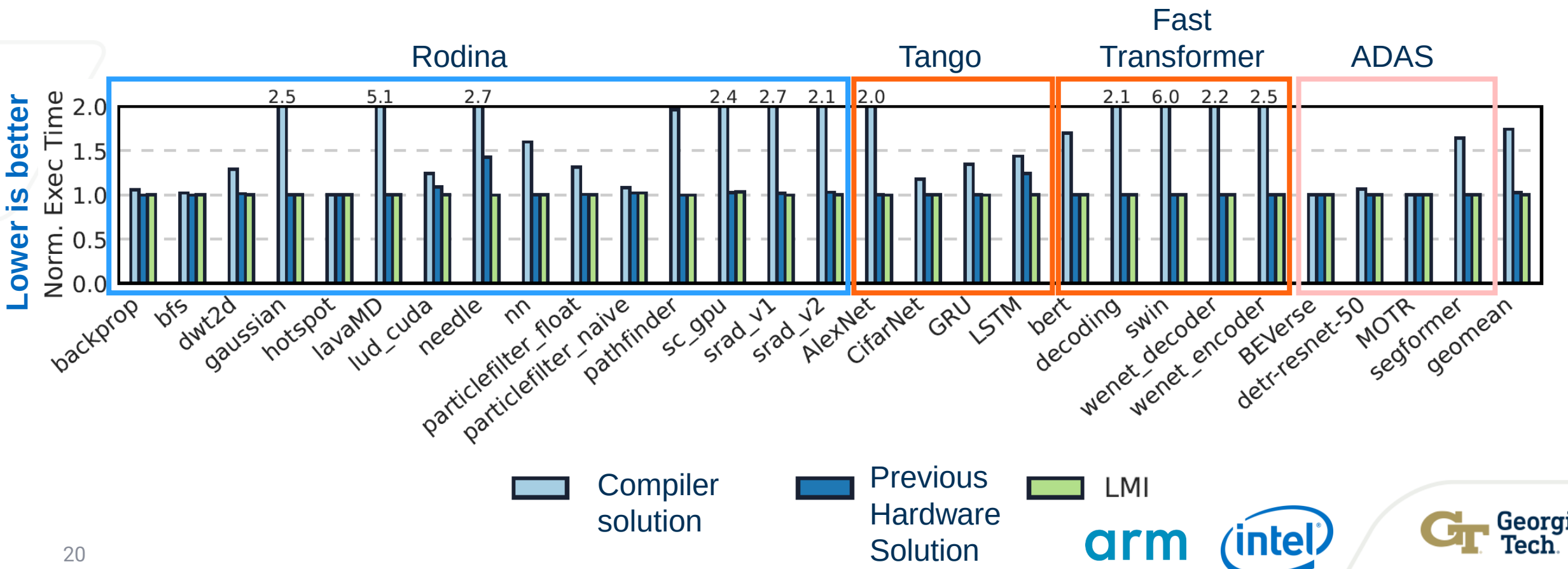
- Simulation Environment.
  - Macsim[1] Simulator.
  - With trace generated with NVBit.
- Target Benchmarks
  - HPC , ML, LLM, and ADAS applications

[1] Kim, Hyesoon, Jaekyu Lee, Nagesh B. Lakshminarayana, Jaewoong Sim, Jieun Lim, and Tri Pho. "Macsim: A cpu-gpu heterogeneous simulation framework user guide." *Georgia Institute of Technology* (2012): 1-57.

arm intel Georgia Tech

Evaluation for HW/Compiler solutions

- LMI Shows the better performance with wider security coverage on various benchmarks.



Conclusion

- Proposes an efficient bounds-checking solution with **in-pointer meta-data** for fine-grained GPU memory safety.

- Through employing **power-of-two-sized** buffer allocation,
  - Minimized the metadata so that it can be embedded into pointers
  - Extremely low bounds checking overhead
  - Enable to implement correct-by-construction concept, so that LMI guarantee the integrity of pointer from pointer creation to pointer deallocation.

THANK YOU!!
Questions?

Georgia Tech