WAVEN: WebAssembly Memory Virtualization for Enclaves

Weili Wang[1*], Honghan Ji[2], Peixuan He[2], Yao Zhang[2], Ye Wu[2], Yinqian Zhang[1]

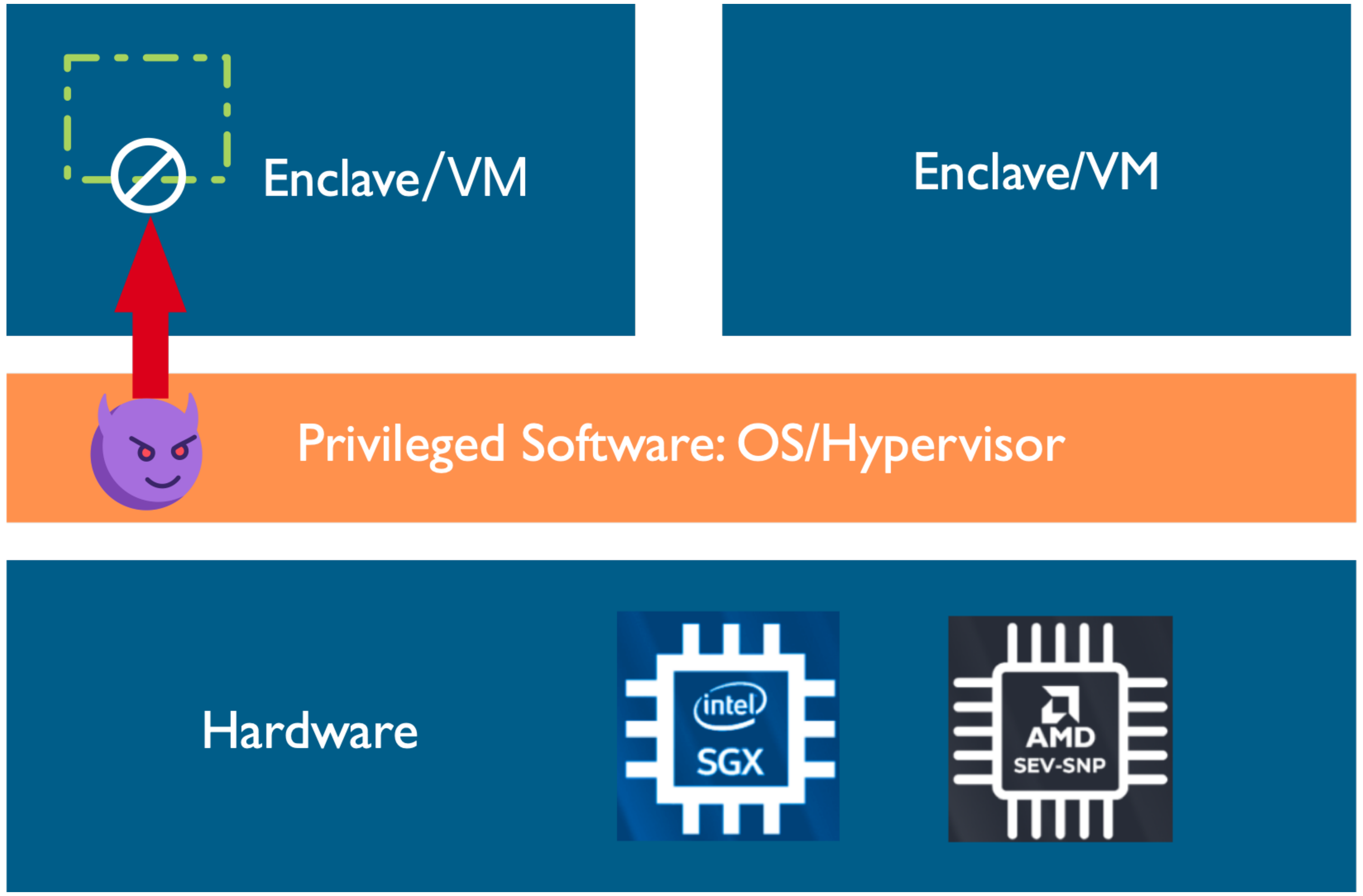
[1]Southern University of Science and Technology 南方科技大学 SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY Teecert Labs

[2]ByteDance Inc. ByteDance 字节跳动

[*] Currently a Ph.D. student at Duke University

Trusted Execution Environments (TEEs)

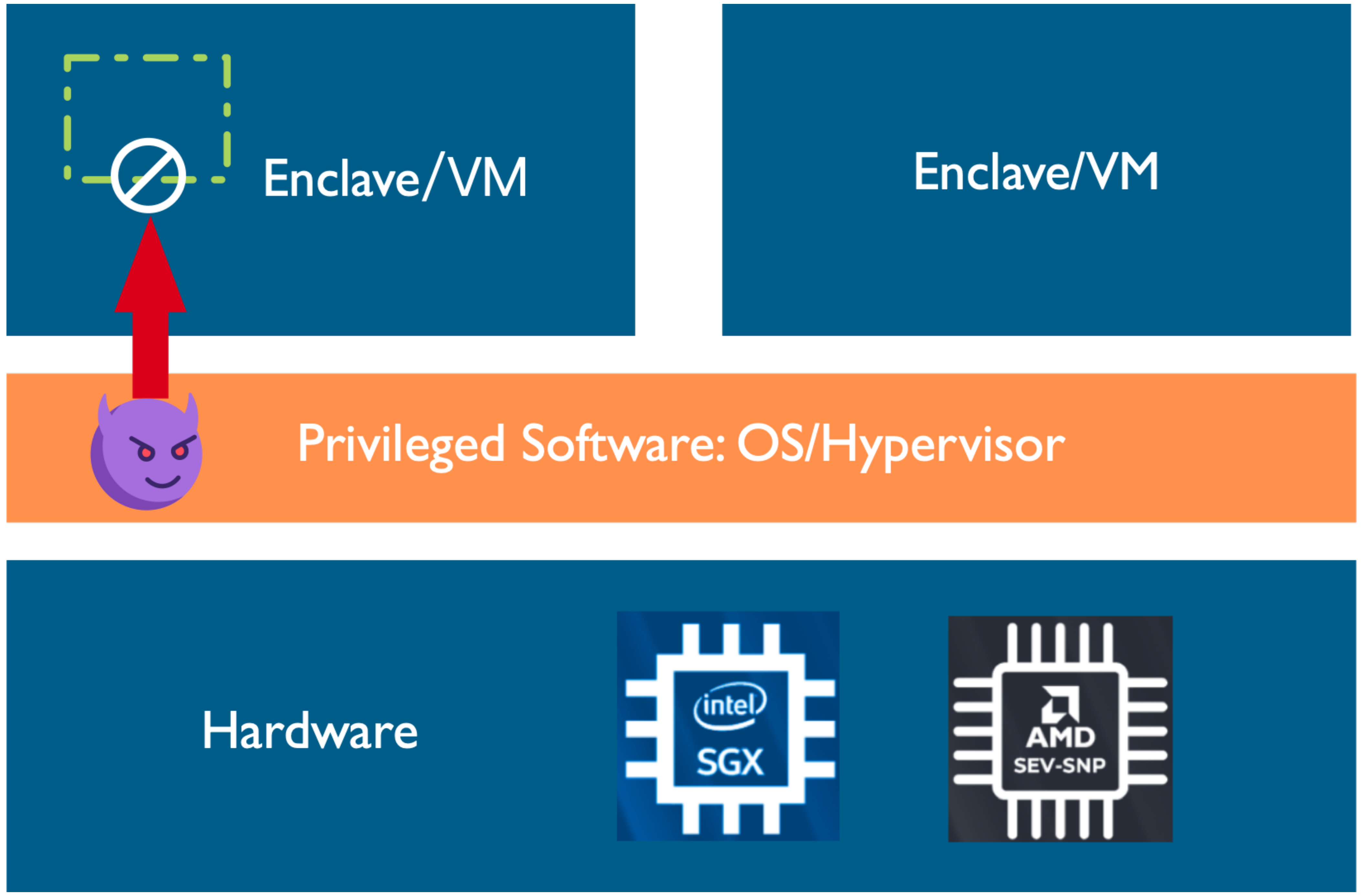
Secure containers immune to attacks from privileged software



- VM-based TEEs
  - AMD SEV, Intel TDX
  - VM-level abstraction
- Enclave-based TEEs
  - Intel SGX, Keystone, Sanctum, CURE…
  - **Significantly smaller TCB**

Trusted Execution Environments (TEEs)

Secure containers immune to attacks from privileged software

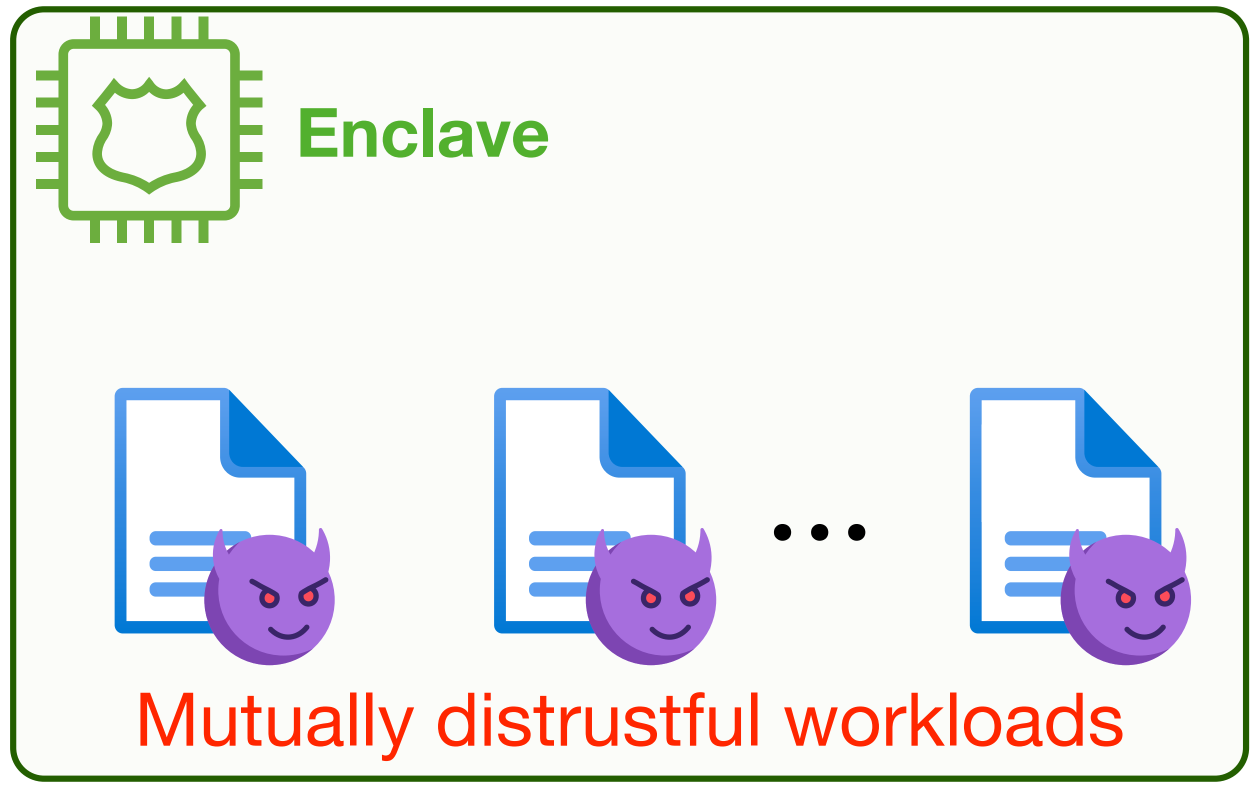


- VM-based TEEs
  - AMD SEV, Intel TDX
  - VM-level abstraction
- Enclave-based TEEs
  - Intel SGX, Keystone, Sanctum, CURE…
  - **Significantly smaller TCB**

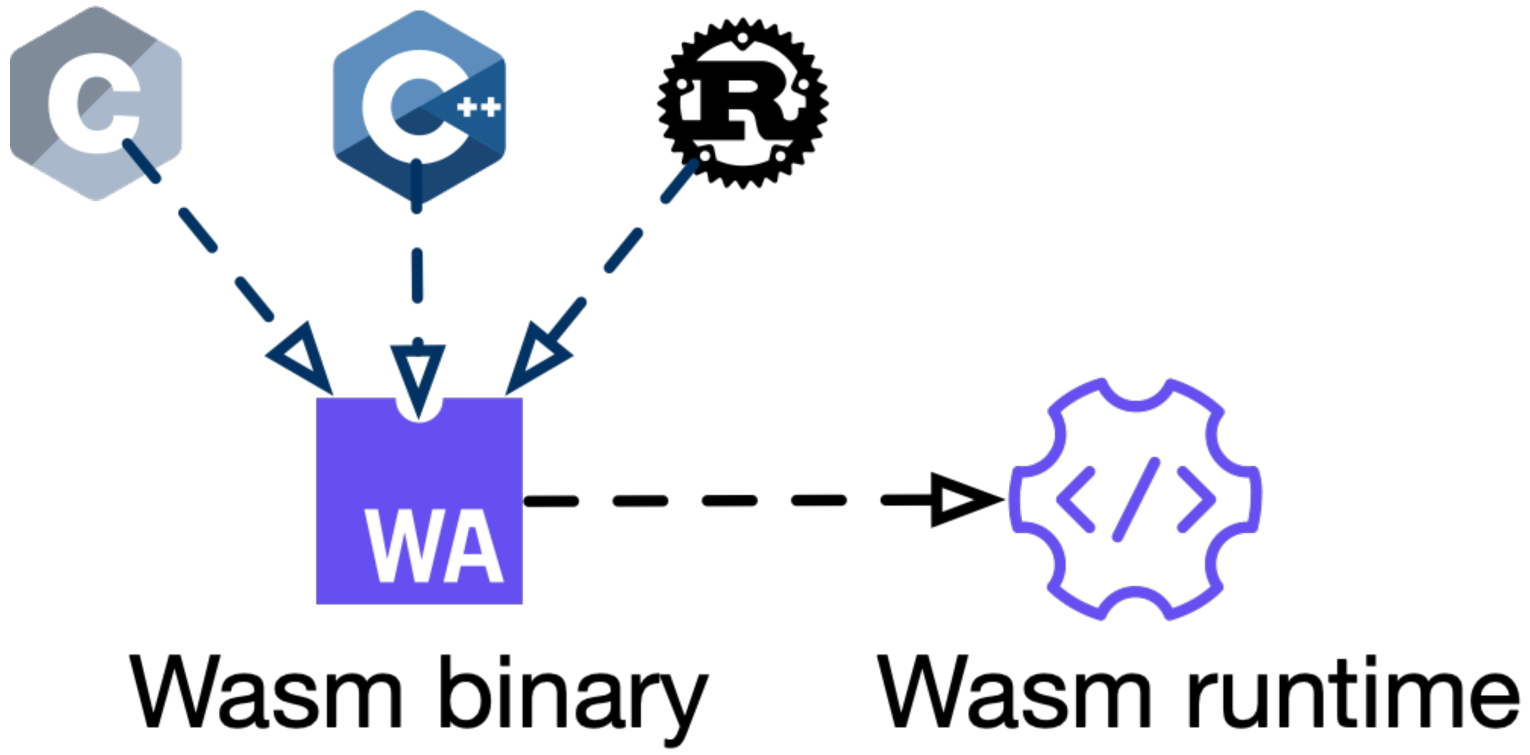
Enclave-based TEEs are here to stay

In-Enclave Multi-Tenancy for SGX

- In-enclave multi-tenancy
  - **Mutually distrustful workloads** in one enclave
    - Confidential Function-as-a-Service (FaaS)
    - Privacy-preserving data analysis

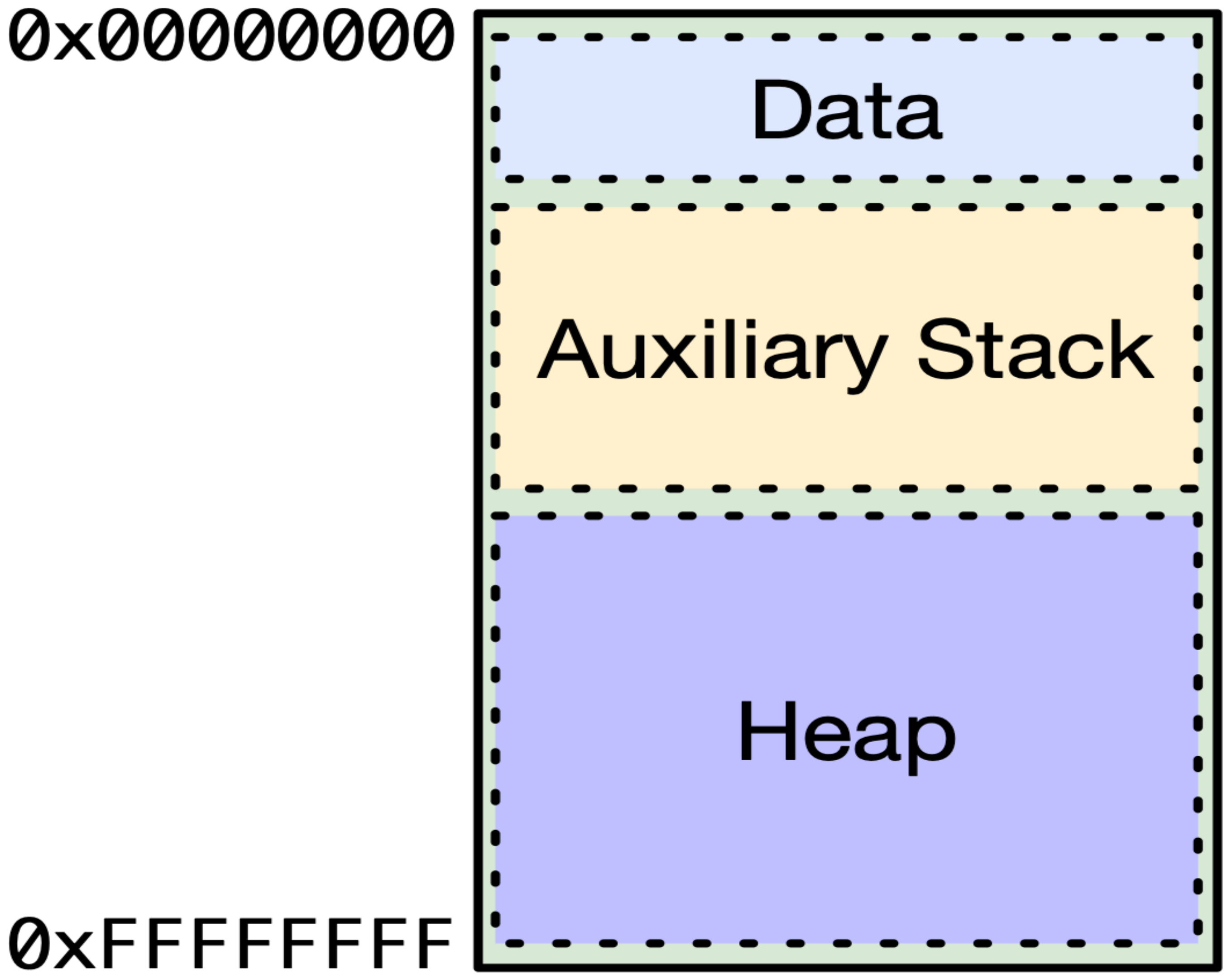


- WebAssembly (Wasm) as a solution
  - A novel portable and efficient binary format
  - Isolated **sandboxes** for Wasm modules
  - “Wasm+SGX” designs: TWINE, Reusable Enclaves…



WebAssembly Memory Isolation

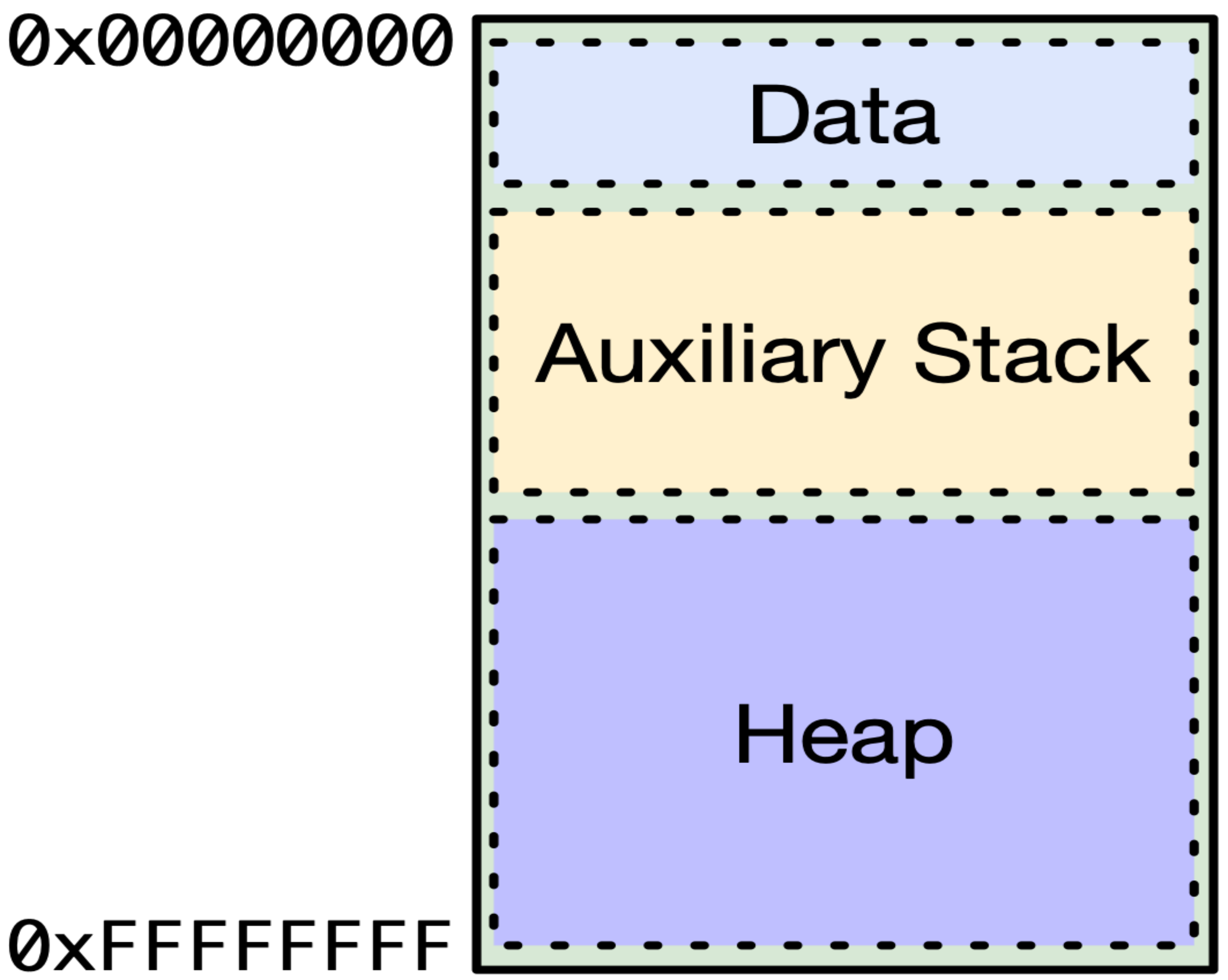
Wasm features a linear memory model isolating modules' memories



- Linear memory
  - A **contiguous byte array**
  - 32-bit Wasm addresses
  - One memory per module
  - **Boundary-check-based isolation**

WebAssembly Memory Isolation

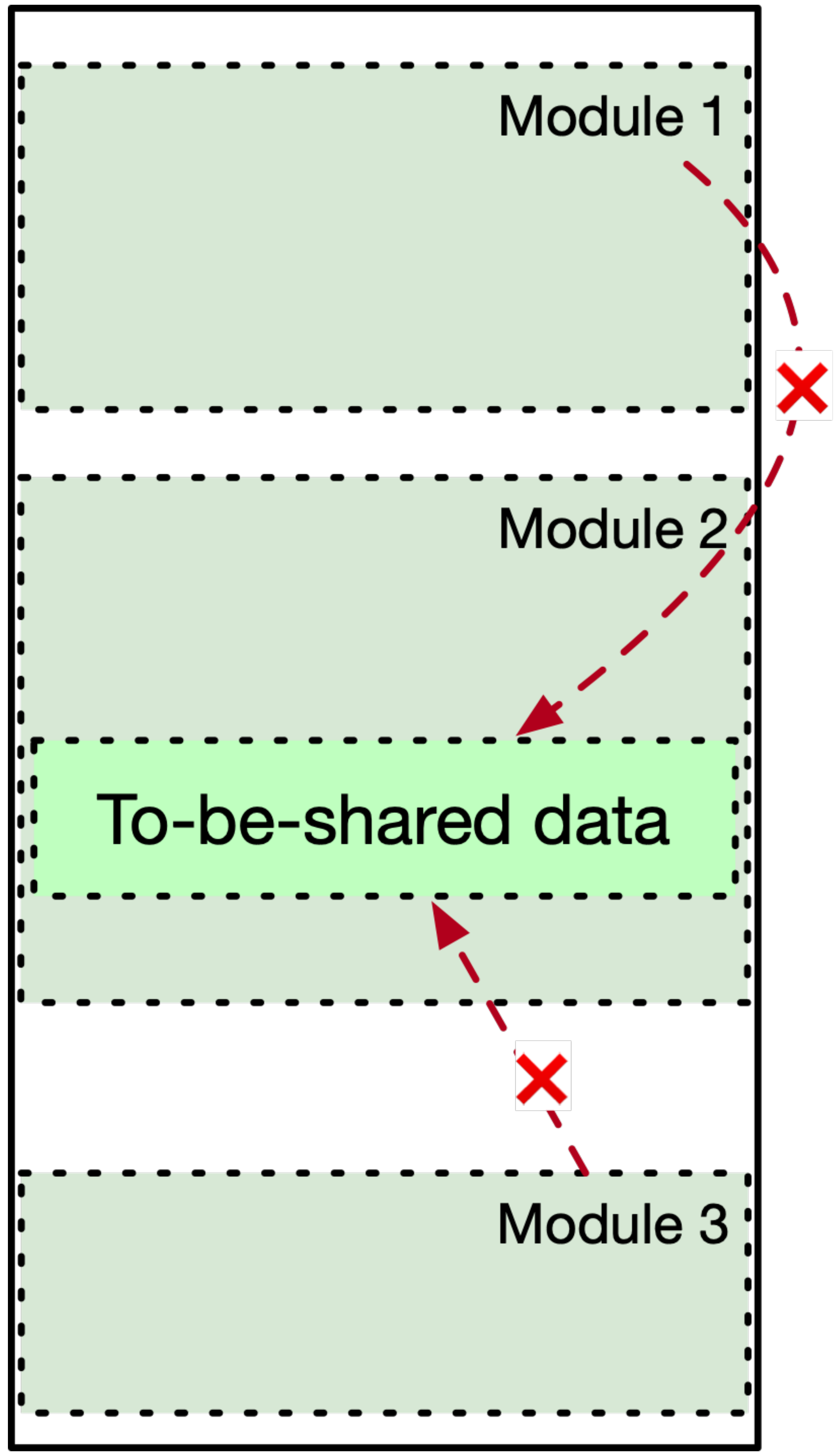
Wasm features a linear memory model isolating modules' memories



- Linear memory
  - A **contiguous byte array**
  - 32-bit Wasm addresses
  - One memory per module
  - **Boundary-check-based isolation**

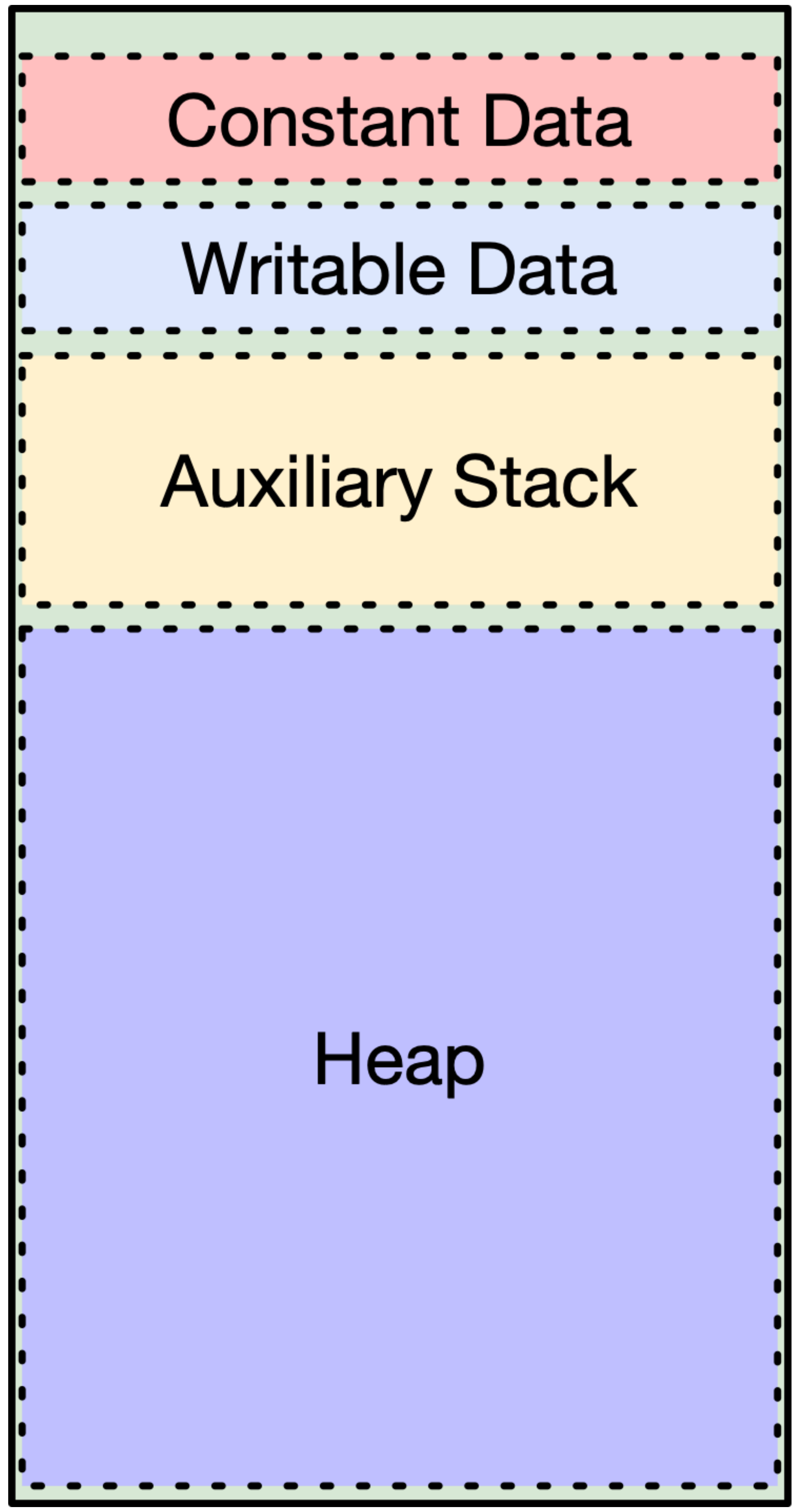
*Linear memory model is incompatible with confidential computing scenarios where **data sharing** and **access control** is important*

Limitations of Linear Memory



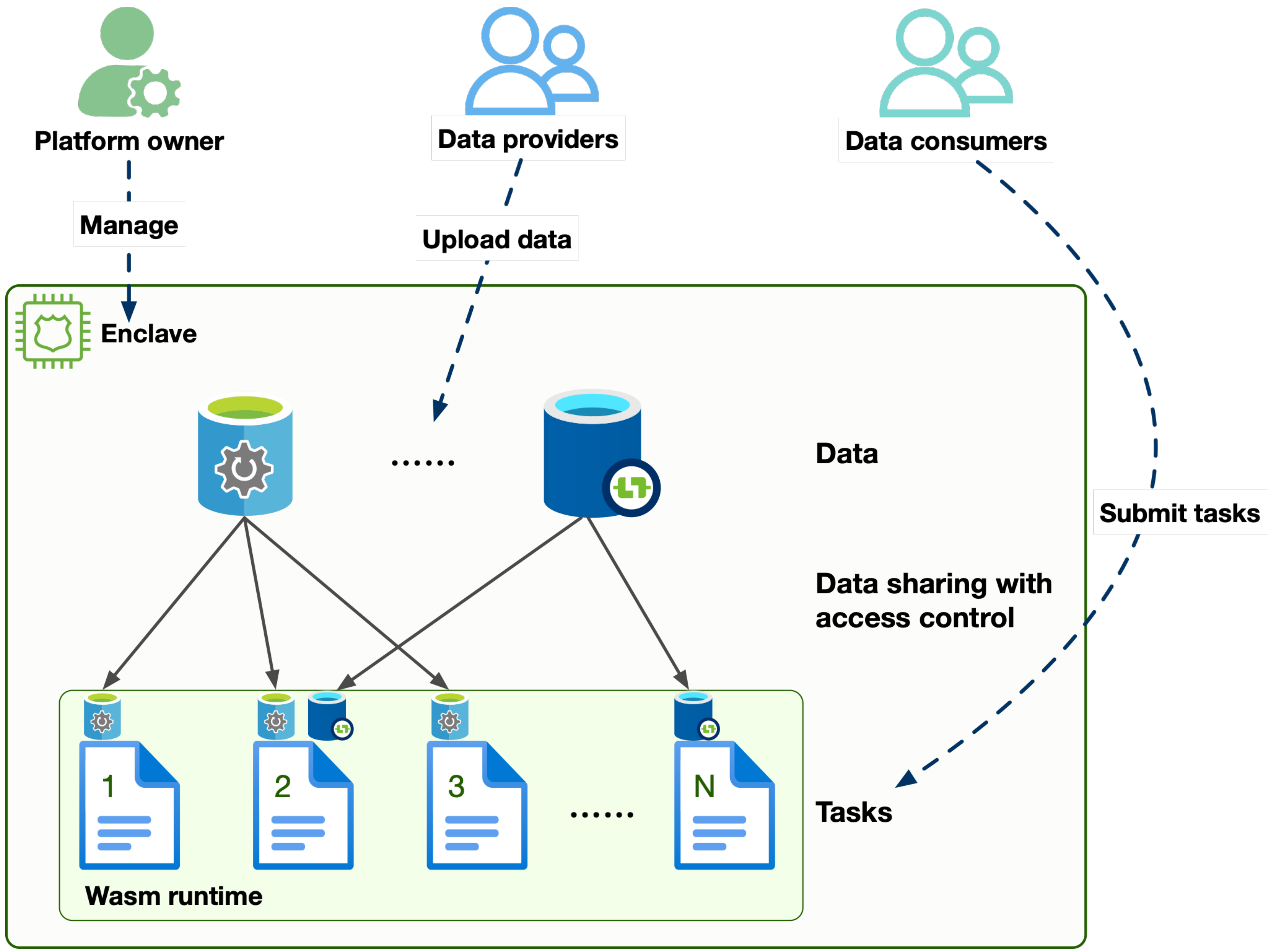
- Limitation: **Inefficient memory sharing**
  - One memory per module
    - Share by **exporting entire memory**
    - Inflexible and impractical
  - Multi-memory proposal
    - Coarse-grained sharing
    - **No compiler support**

Limitations of Linear Memory



- Limitation: **Lack of memory access control**
  - No read-only memory
  - All partitions are writable
  - **Not secure in memory sharing**
    - Shared data is entirely writable
    - Shared data can be tampered with

System Model



- **Roles**
  - **Platform owner** provides service
  - **Data providers** share data
  - **Data consumers** compute
- **Security goals**
  - Execution confidentiality
  - Execution integrity
  - Controlled data sharing

Example Use Cases

	Confidential stateful FaaS	Secure data marketplaces
Platform owner	Platform operator	Market operator
Data providers	FaaS users or dataset owners	Data sellers
Data consumers	FaaS users	Data buyers

1. **Confidential stateful FaaS**
    - A task uses parallel modules
    - Shared data across modules
    - Modules cannot modify the data
2. **Secure data marketplace**
    - Sellers share their data
    - Buyers compute on it
    - Buyer cannot modify the data

WebAssembly Memory Virtualization as a Solution

WAVEN: **W**eb**A**ssembly Memory **V**irtualization scheme for **EN**claves

- Experience in OS evolvement
  - Modules hosted in a Wasm runtime vs. Processes running in an OS
    - Alike an OS kernel, the runtime manages the memory of modules
  - OS memory management
    - From **direct allocation on physical memory** to **memory paging**

WebAssembly Memory Virtualization as a Solution

WAVEN: **W**eb**A**ssembly Memory **V**irtualization scheme for **EN**claves

- Experience in OS evolvement
  - Modules hosted in a Wasm runtime vs. Processes running in an OS
    - Alike an OS kernel, the runtime manages the memory of modules
  - OS memory management
    - From **direct allocation on physical memory** to **memory paging**

Inspired by OSs' evolvement, we propose a memory virtualization scheme for in-enclave Wasm runtimes, supporting ***memory sharing with access control***

Design Goals & Challenges

Goals

- **Practicality**: Comply with Wasm spec
- **Security**: Memory isolation guarantee
- **Performance**: Minimal overhead

Challenges

- **Complexity**: Design could be complex
- **Efficiency**: Software MMU is slow
- **Compatibility**: No linear memory

Design Goals & Challenges

Goals

- **Practicality**: Comply with Wasm spec
- **Security**: Memory isolation guarantee
- **Performance**: Minimal overhead

Challenges

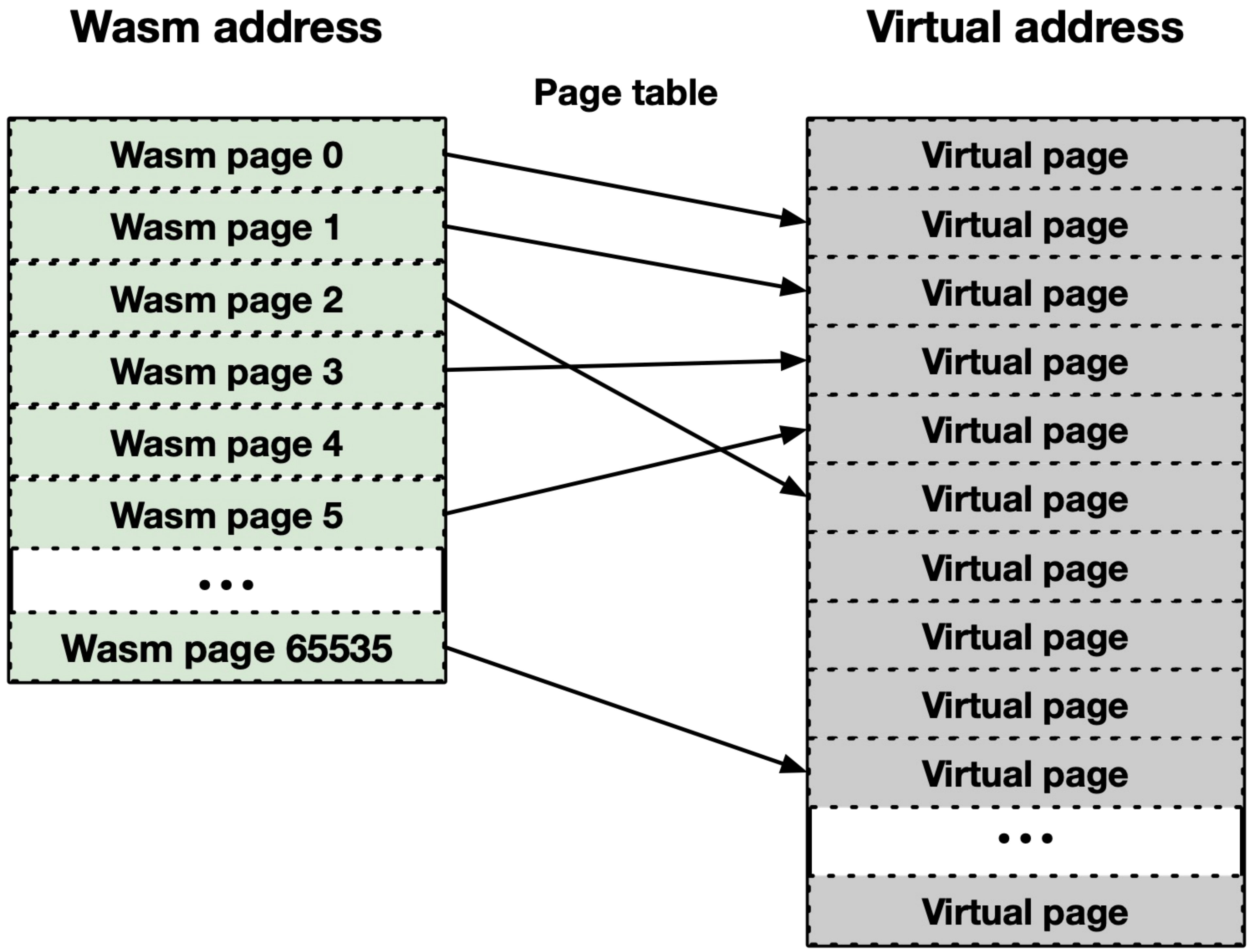
- **Complexity**: Design could be complex
- **Efficiency**: Software MMU is slow
- **Compatibility**: No linear memory

Solutions

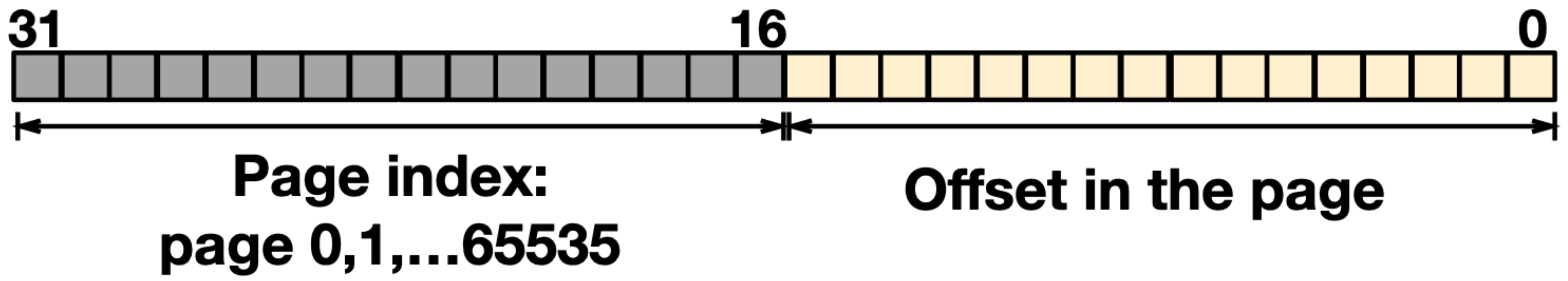
- **Complexity and efficiency**: Single-level page table and dual page tables
- **Efficiency**: Exception page and page padding
- **Compatibility**: Only require modifications to Wasm runtimes

南方科技大学 SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY ByteDance 字节跳动

WebAssembly Paging



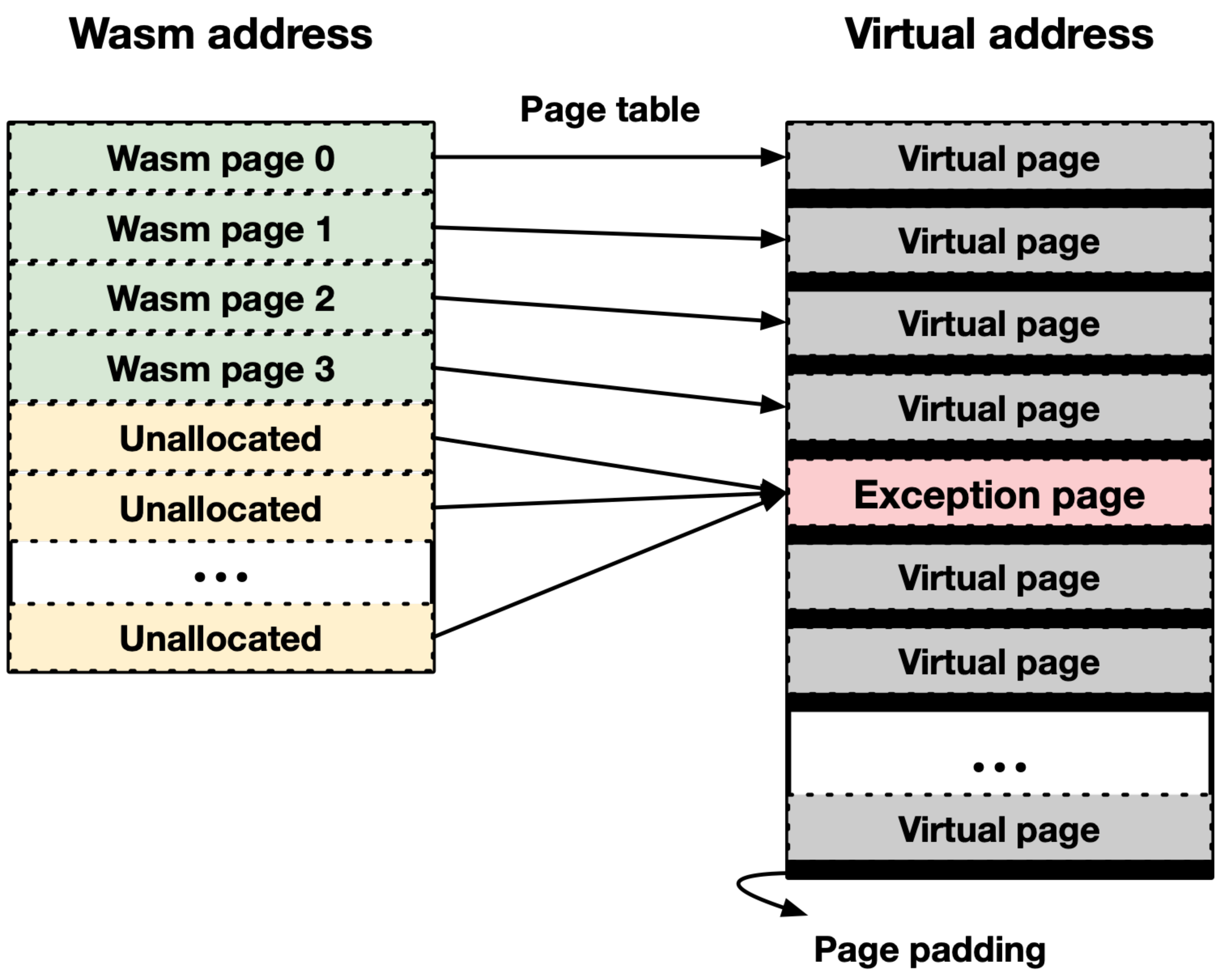
- Memory virtualization
  - 64KB page size
  - "**Virtual address**": Wasm address (32 bits)
  - "**Physical address**": Runtime virtual address
- Single-level page table: **Minimal** page table walk
- Address translation for memory instructions



Memory Isolation

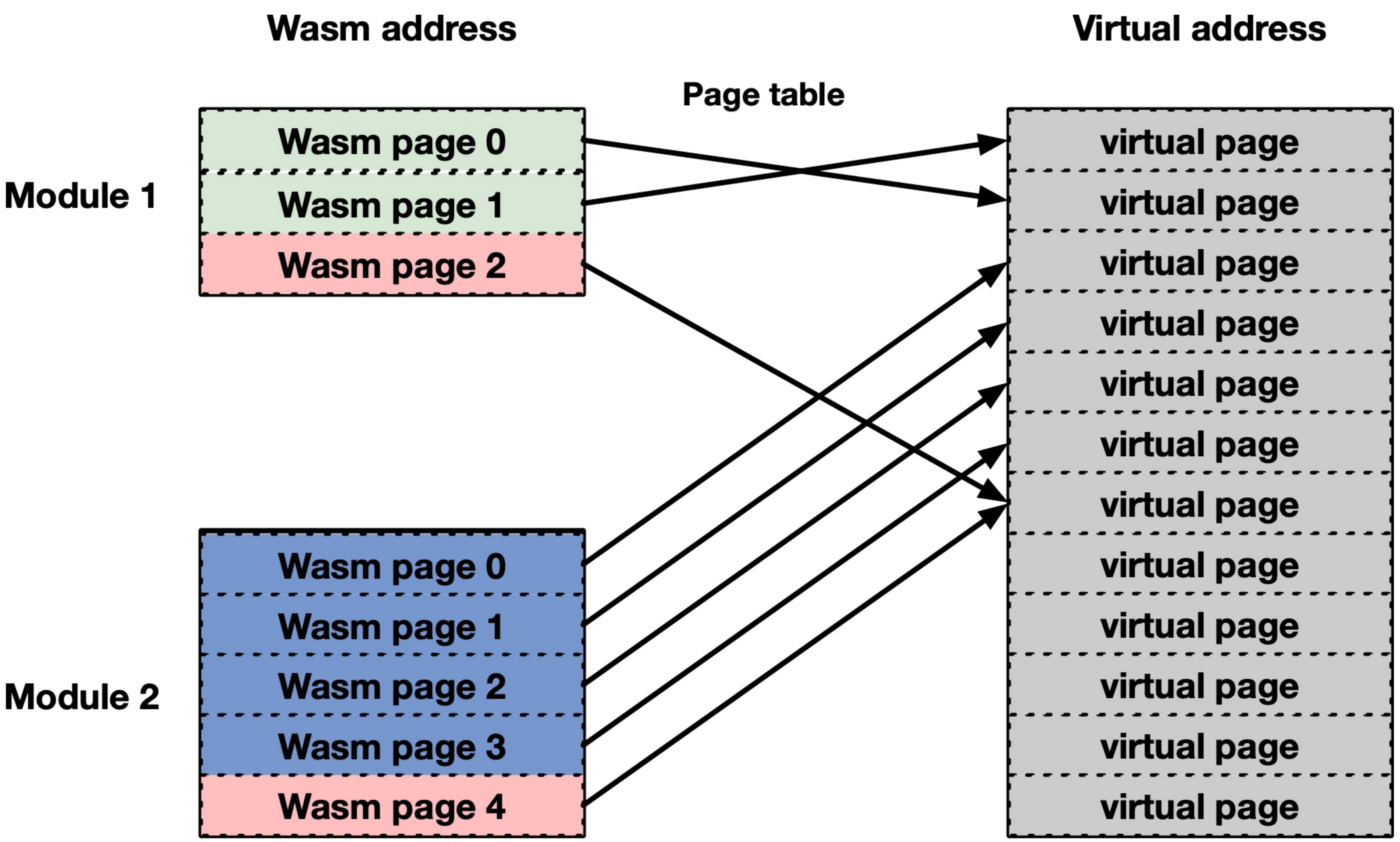
- Linear memory model’s approach
    - Use boundary checks
    - In-SGX Wasm only supports **expensive software checks**
- WAVEN’s approach
    - Optimize the address translation
    - Prevent illegal accesses **without explicit checks**

Memory Isolation



- Exception pages
    - One **empty exception page** for a module
    - Out-of-bound accesses → exception page
- Page paddings
    - Every page **is padded with extra bytes**
        - Cross-page accesses → padding
    - Minimum padding size: **7 bytes**

Memory Sharing

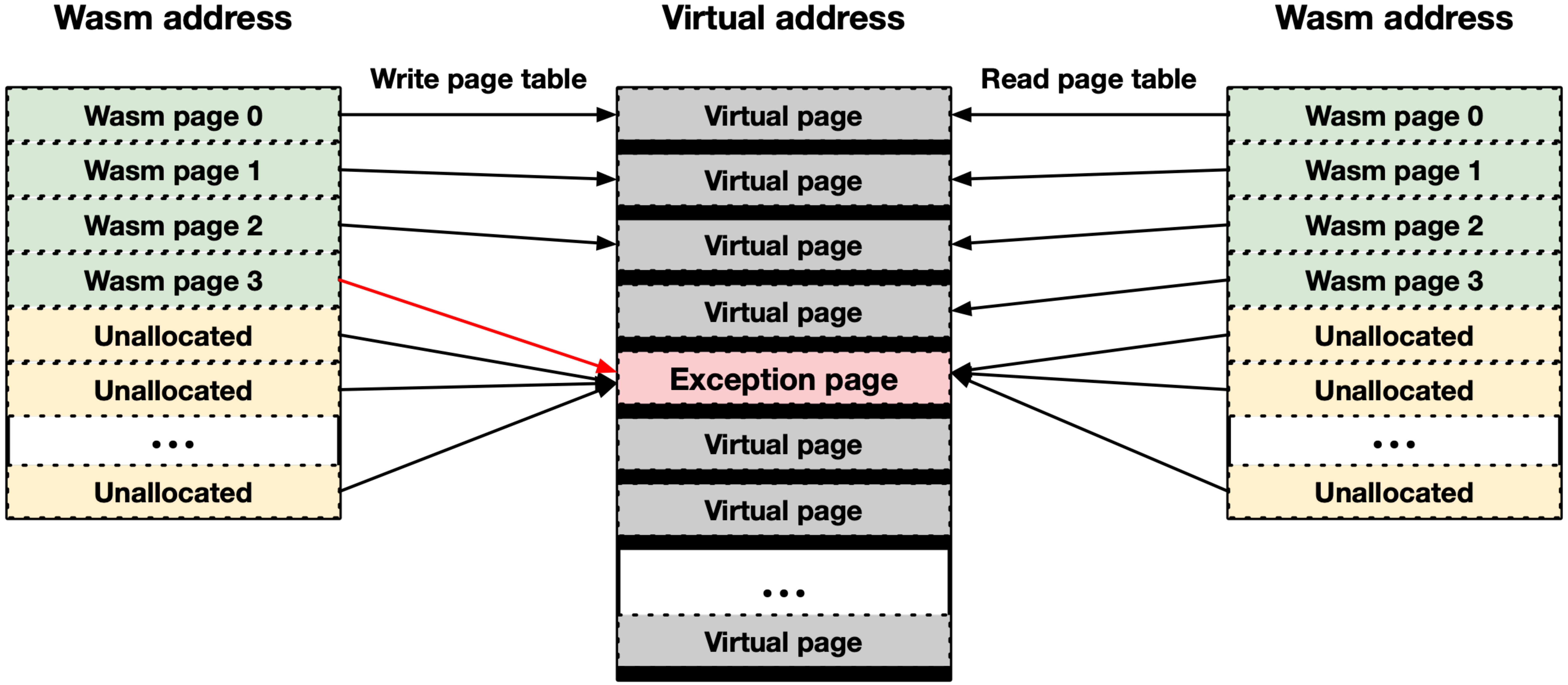


- Sharing by page table manipulation
  - Entries point to the same page
  - Flexible shared memory
    - **Page-granularity**
    - Easy to share
    - Easy to revoke shared data

Memory Access Control

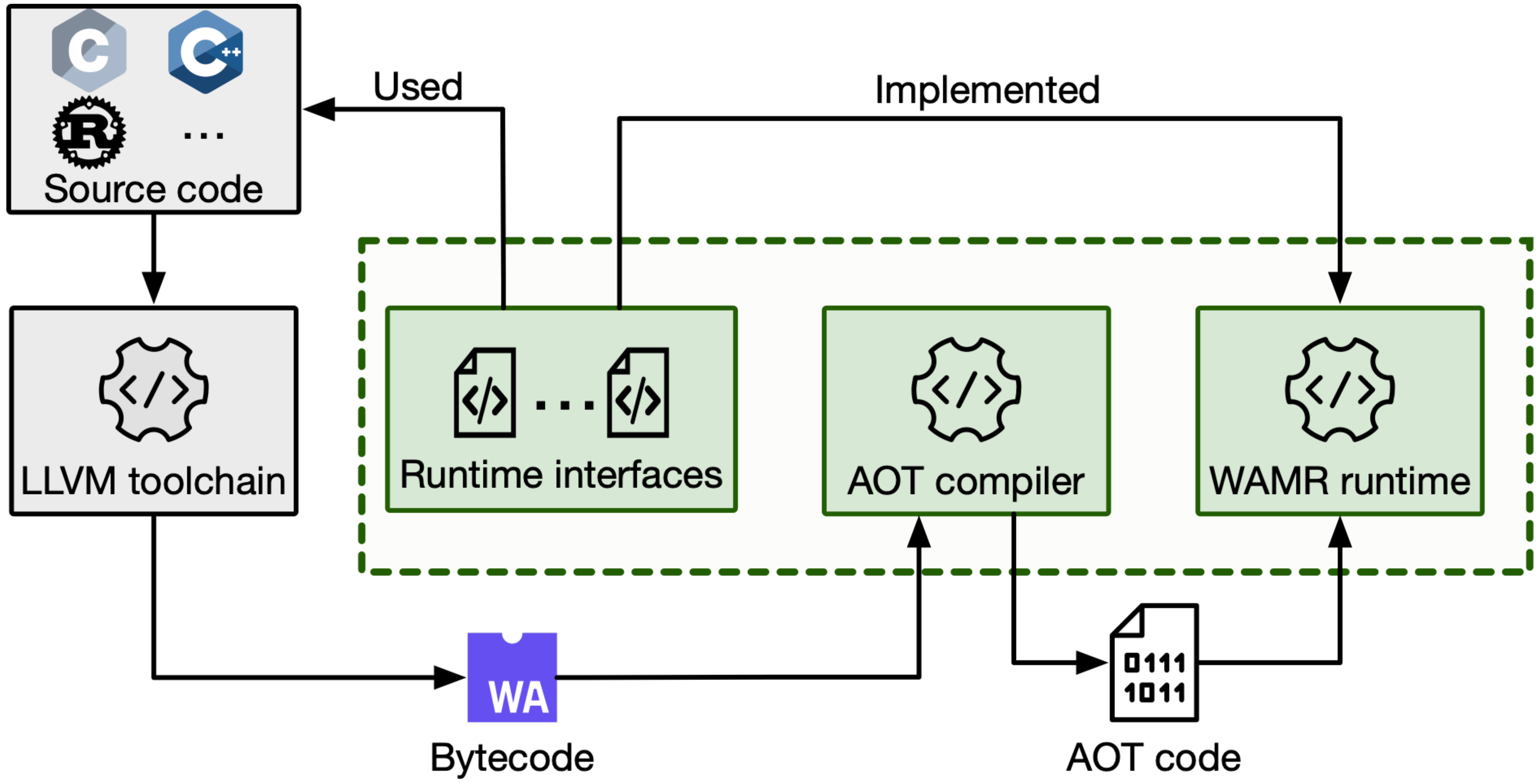
- Approaches for Wasm memory access control
  - Hardware primitives
    - Intel MPK: **Require a trusted OS**
  - Software permission checks
    - Check before accessing: **High overhead**
- Dual-page-table design in WAVEN
  - Read page table for **memory reads**, write page table for **memory writes**
  - Address translation **without expensive permission checks**

Memory Access Control



- In the write page table, **entries of read-only pages point to the exception page**
  - Any memory writes on page 3 is redirected to the exception page

Implementation



Implemented atop WAMR, a runtime with native SGX support

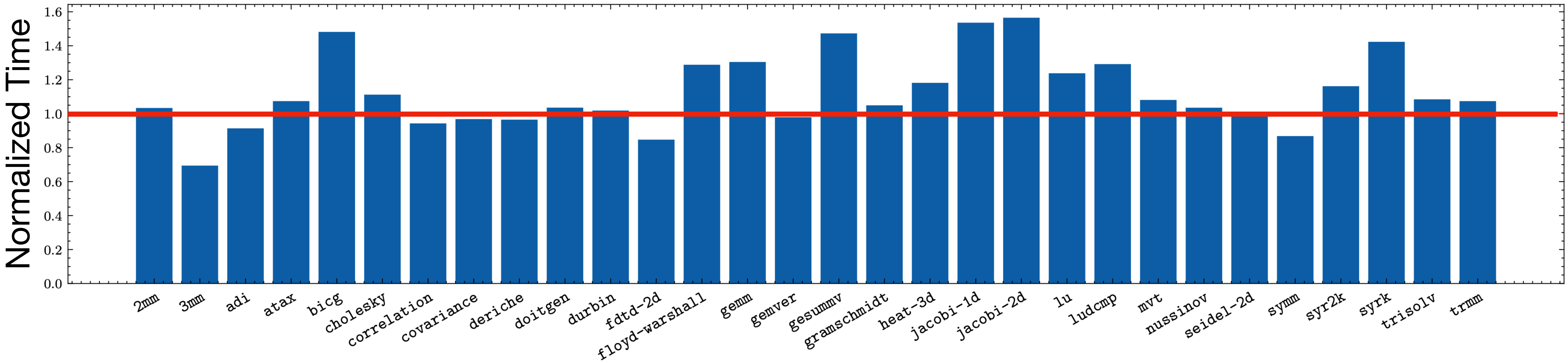
- Support Ahead-of-time compilation
  - Modify the compiler to support address translation
  - Modify the runtime to manage page tables during execution
  - Specify shared memory interfaces

Evaluation

- Benchmarks
  - PolyBench: Scientific computing tasks
  - STREAM: Memory stress tests
  - Confidential workloads: Database and machine learning inference
  - Memory sharing scenarios: **Multi-write multi-read and multi-read settings**
- Evaluation questions
  - What’s the **performance of WAVEN**?
  - What’s the **effectiveness of memory sharing**?

Performance on PolyBench

What’s the performance of WAVEN in scientic computation tasks?



- Vanilla WAMR uses software boundary checks
- WAVEN incurs extra memory reads for page table lookups
- Overheads are dependent on the memory access patterns
- The geometric mean of overheads is **10.42%**

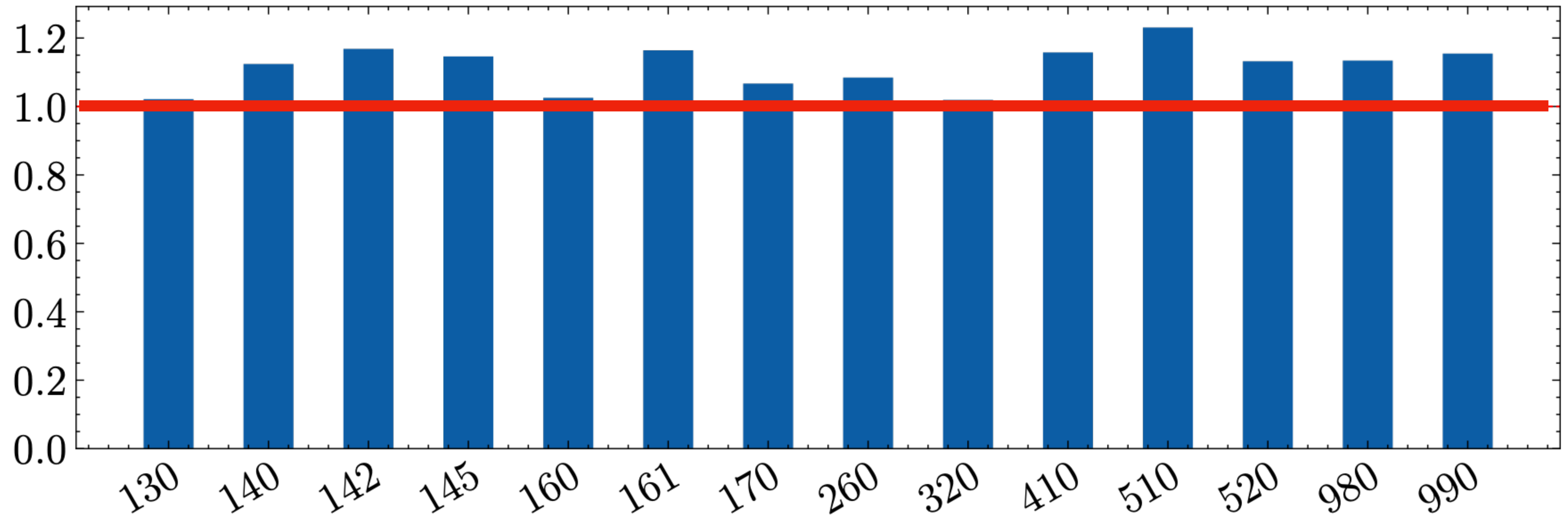
南方科技大学 SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

ByteDance 字节跳动

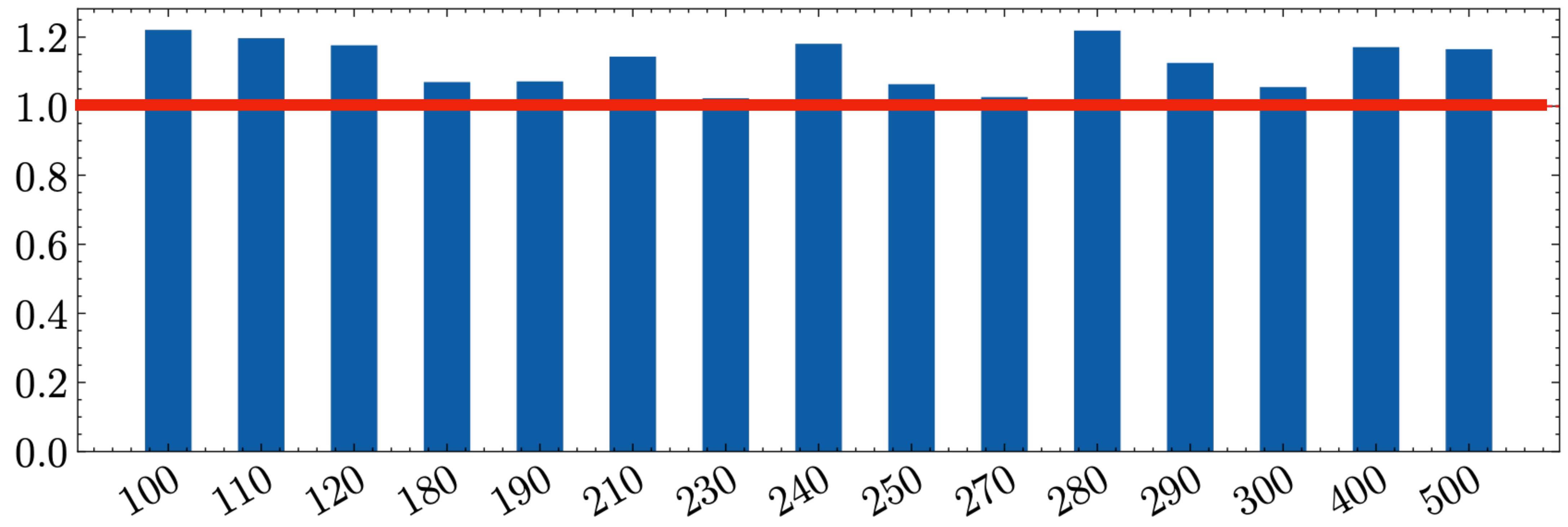
Performance on Typical Confidential Workloads

What's the performance of WAVEN in confidential workloads?

Confidential database (SQLite)



Average overhead of database query: **11.47%**



Average overhead of database update: **12.52%**

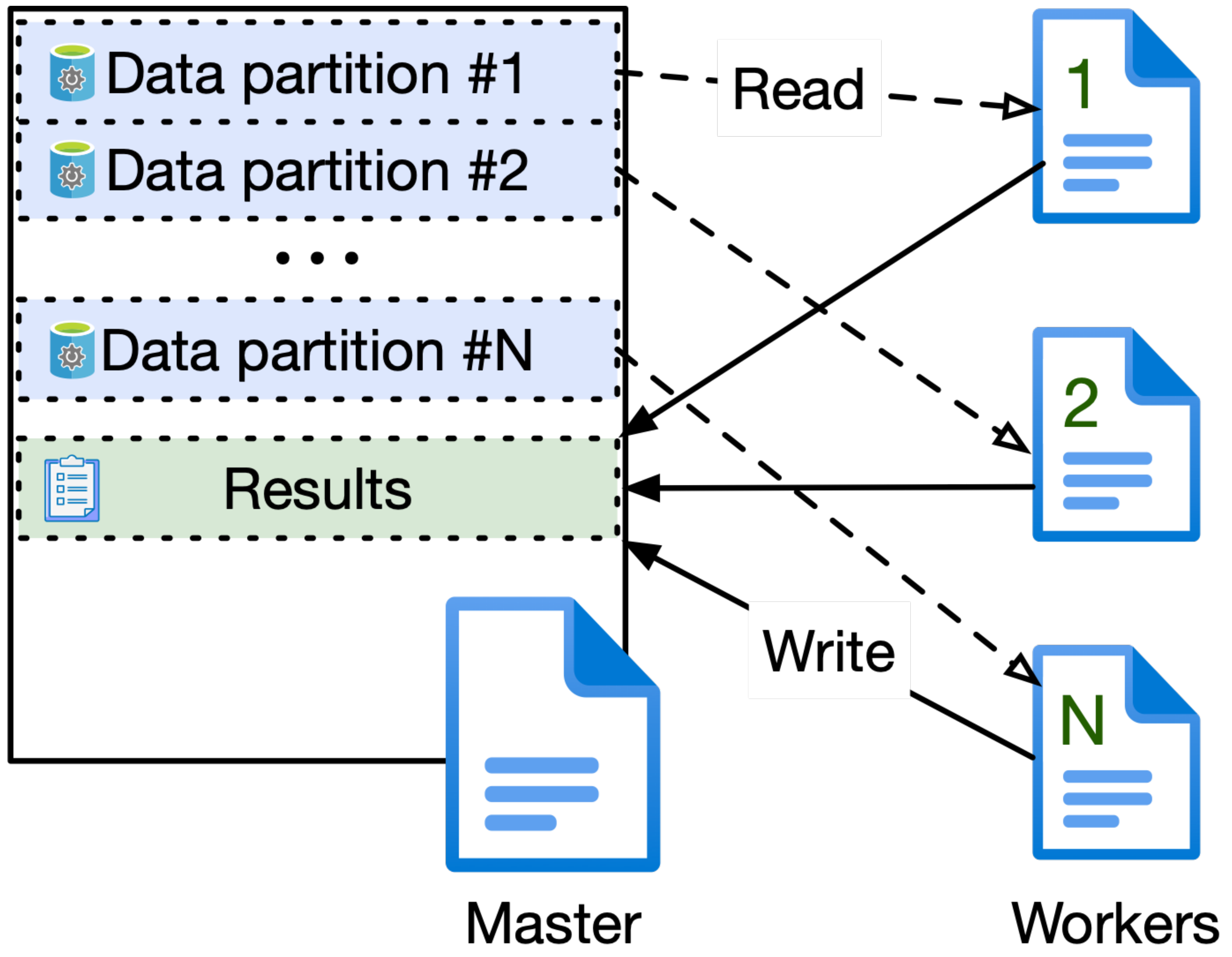
Privacy-preserving ML inference

- Run a face detection model
- Measure the time used in detection
- WAVEN only exhibits **6.14%** overhead
  - WAVEN takes 176.06s to process
  - Vanilla WAMR takes 165.87s

Effectiveness of Memory Sharing

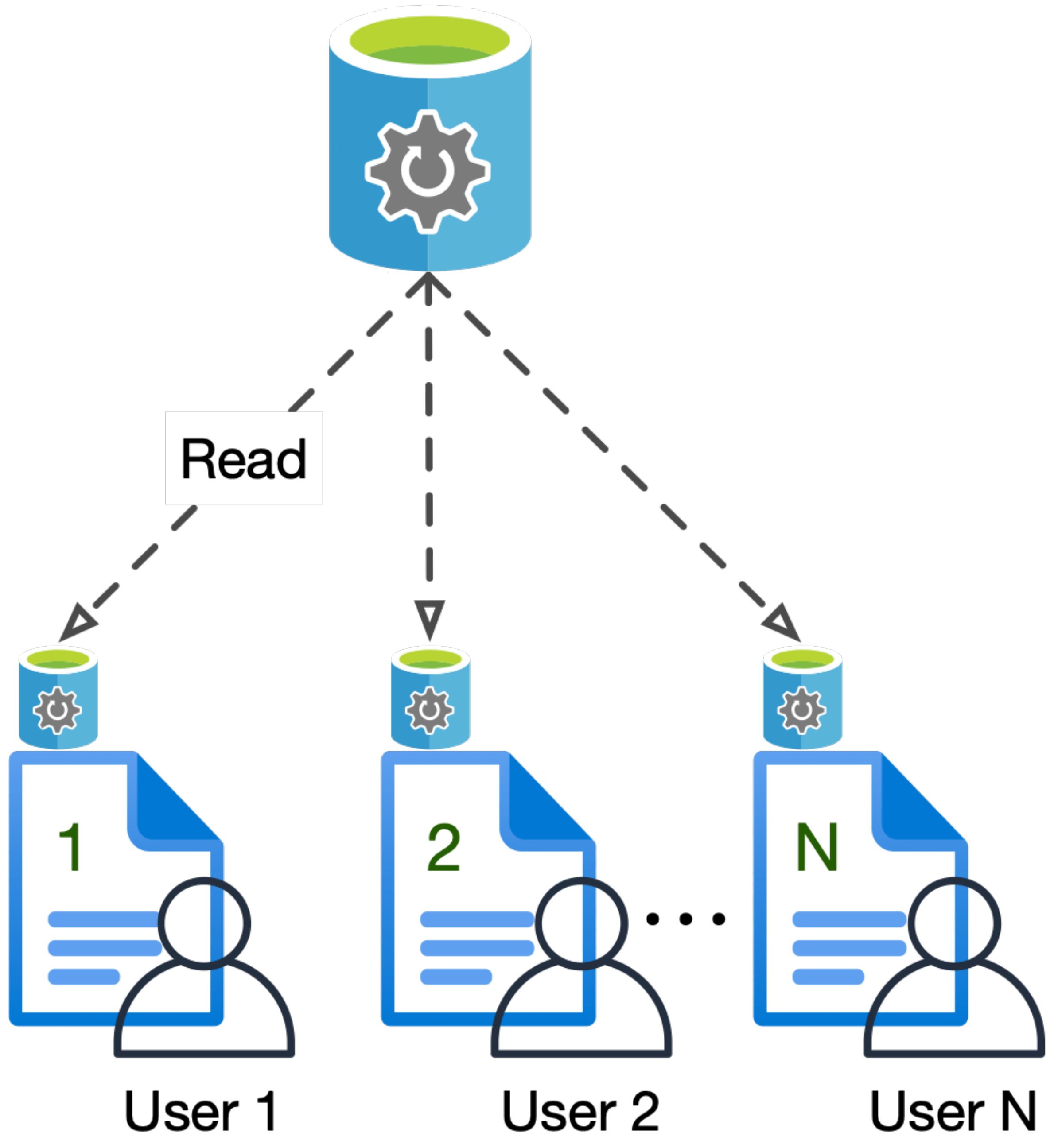
Multi-write multi-read setting

- Typical in confidential stateful FaaS
- Master and worker functions



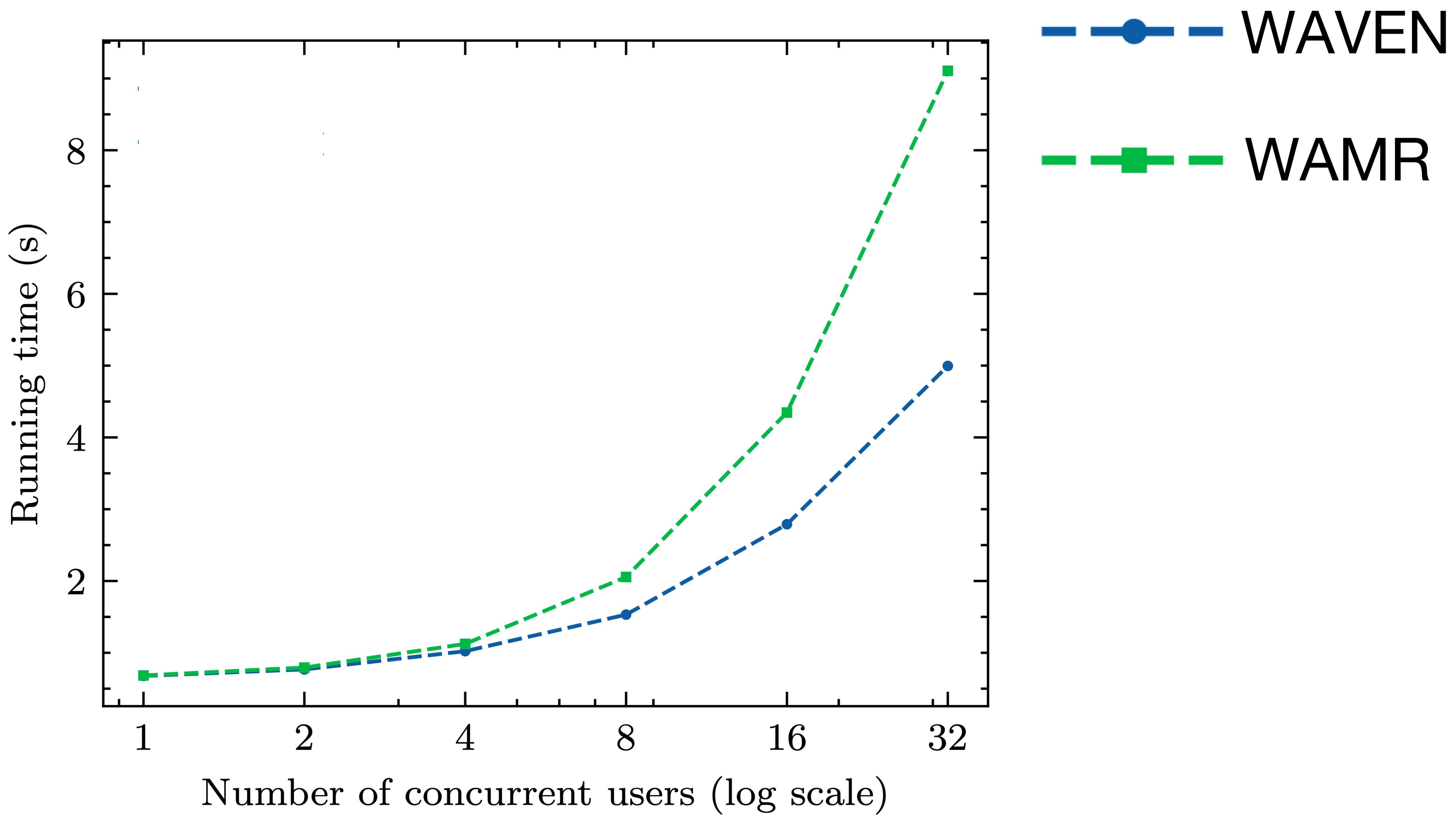
Multi-read setting

- Typical in secure data marketplaces
- Users compute on the same data



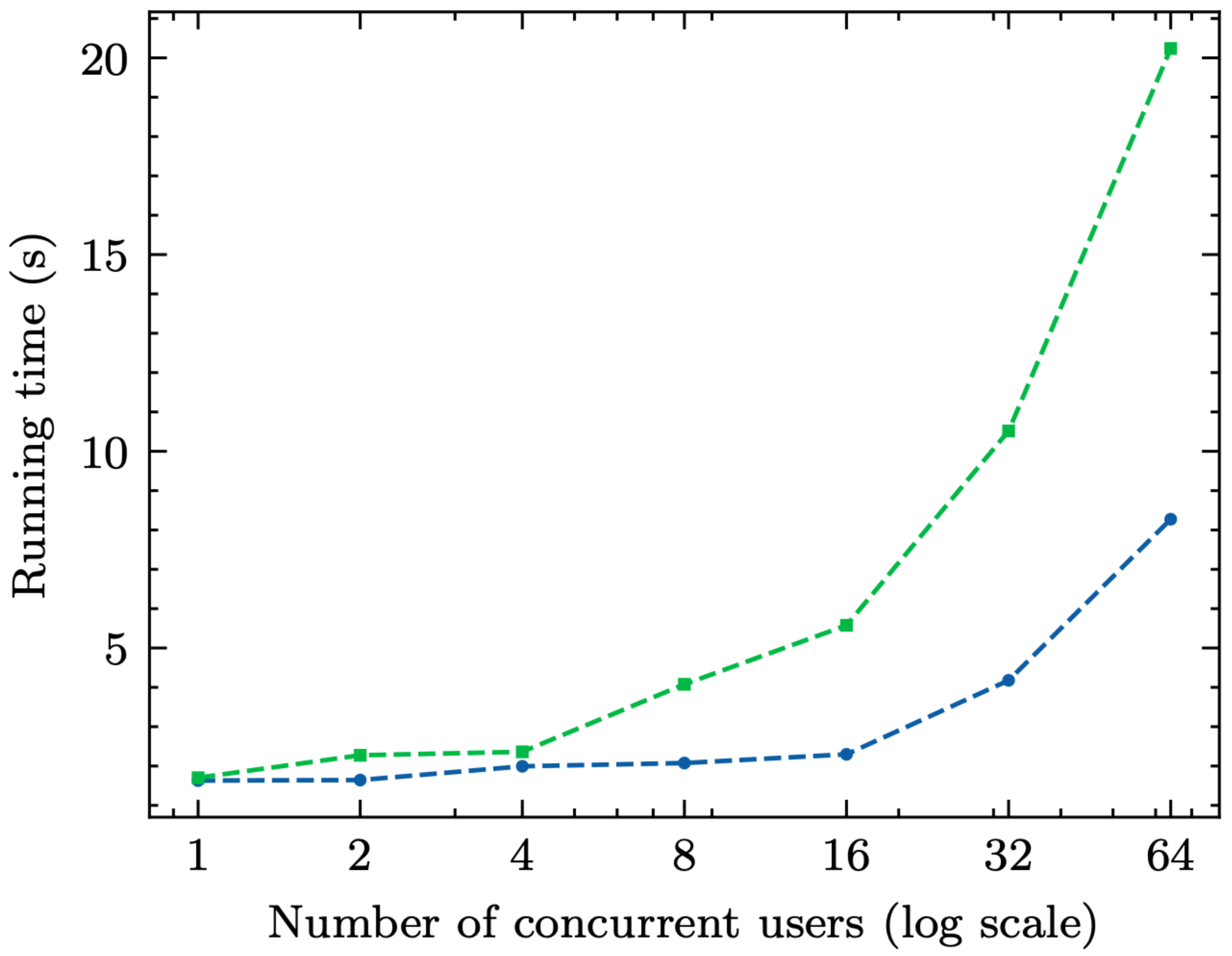
Effectiveness of Memory Sharing

Multi-write multi-read setting



Peak speedup: **1.56×-1.82×**

Multi-read setting



Peak speedup: **2.4×-2.5×**

南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY
ByteDance
字节跳动

Discussion

- Prevelance of in-enclave WebAssembly
  - Wasm is **formally specified and verified**
  - Wasm has a **strong ecosystem**
- Generalization to other TEEs
  - SGX is still important: WAVEN is **useful in the long run**
  - WAVEN can also be adapted to other enclave-based TEEs
- TLB implementation
  - Tried implementing software TLB
  - Extra overhead (~21%) due to the “miss or hit” checks

Related Work

- Intra-enclave isolation
  - Other software fault isolation approach
    - **Not as flexible as Wasm**: CHANCEL (NDSS ’21), users cannot execute their code
  - Hardware-based approach
    - **Deprecated technique** (Intel MPX): MPTEE (EuroS&P ’20) and Occlum (ASPLOS ’20)
    - **Require hardware modification** to use Intel MPK: LightEnclave (Security ’22)
- Confidential computing with WebAssembly
  - Two-way sandboxes: TWINE (ICDE ’21) and AccTEE (Middleware ’19)
  - FaaS: Reusable enclaves (Security ’23) and Se-Lambda (SecureComm ’18)

*WAVEN is the first approach that enables **cross-module memory sharing with fine-grained memory access control** for in-enclave Wasm*

Conclusion

- WebAssembly + SGX
  - A popular design paradigm that provides in-enclave multi-tenancy
  - Linear memory model impedes important confidential computing scenarios where controlled data sharing is highly needed
- WAVEN
  - In-enclave memory virtualization as a solution
  - Page-granularity memory sharing with access control
  - Much better performance in data sharing scenarios

Refer to our paper for more details!