Civet: An Efficient Java Partitioning Framework for Hardware Enclaves

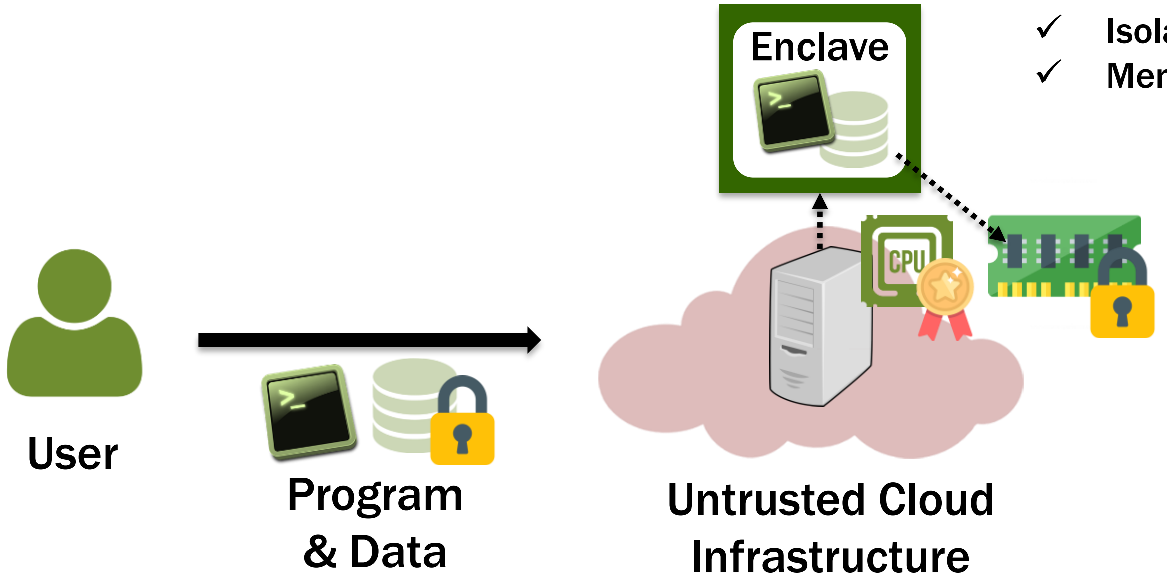
Chia-Che Tsai*†, Jeongseok Son†, Bhushan Jain‡, John McAvey¶, Raluca Ada Popa†, Donald E. Porter‡

**TAMU, †UC Berkeley, ‡UNC Chapel Hill, ¶Hendrix College*

Hardware Enclaves as Root of Trust

An abstraction for bootstrapping users' trust on untrusted platforms.

- ✓ Proof of a trustworthy CPU
- ✓ Isolated from untrusted OS & IO
- ✓ Memory encrypted in DRAM



Existing solutions:

- Intel SGX
- AMD SEV
- ARM TrustZone
- RISC-V Sanctum & Keystone

Cloud Platforms + Enclaves= Large Trusted Computing Base

Cloud applications are often
(1) complex
(2) multi-principal
(3) written in managed languages like Java.

Ex: Hadoop

mapper

reducer

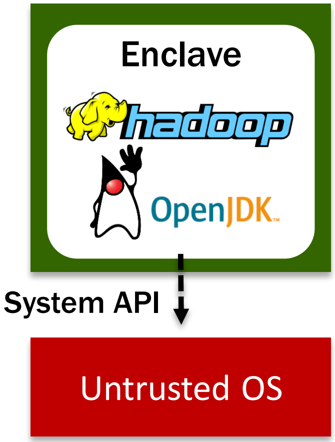
6.3 MLoC

- **Scheduler**
- **HDFS**
- **Workers**
- **Other mappers/reducers**

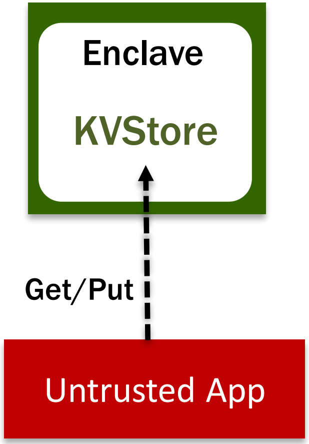
+ 2.3 MLoC (JARs)
+ 0.9 MLoC (JVM)

Existing Approaches for Enclave Development

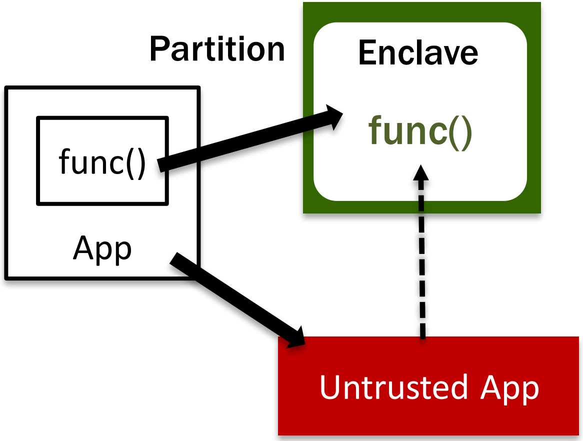
- **Entire Application**
  (Haven'14, SCONE'16, Graphene-SGX'17,SGK-LKL'20)
- **API Engine Only**
  (VC3'15, SecureKeeper'16)
- **Partitioned / Partial Re-development**
  (Glamdring'17, GOTEE'19)



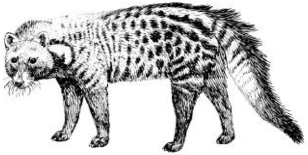
No code modification
but large TCB



Small TCB
but little flexibility



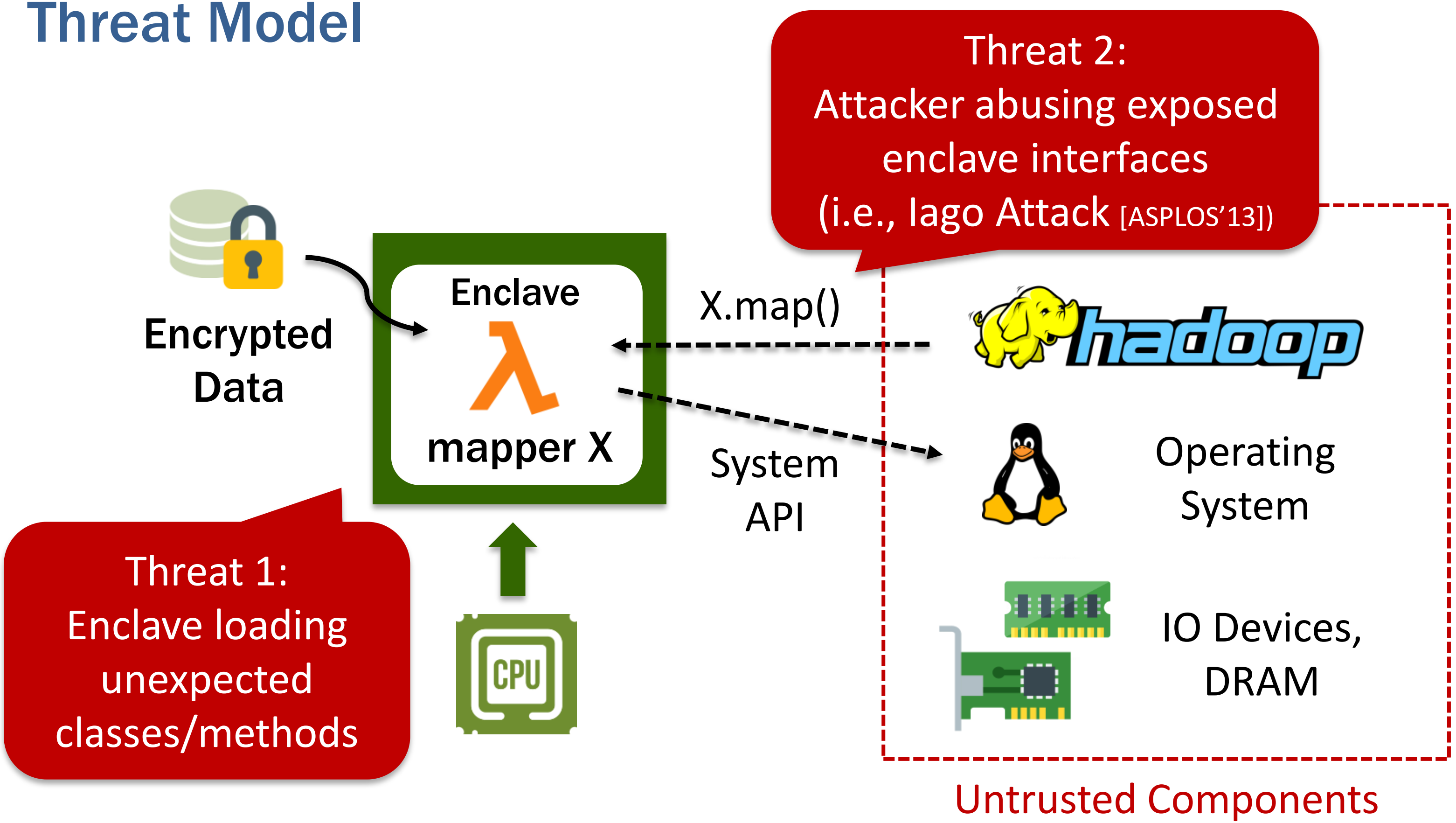
Small TCB
but lack support for partition
with object-oriented interfaces



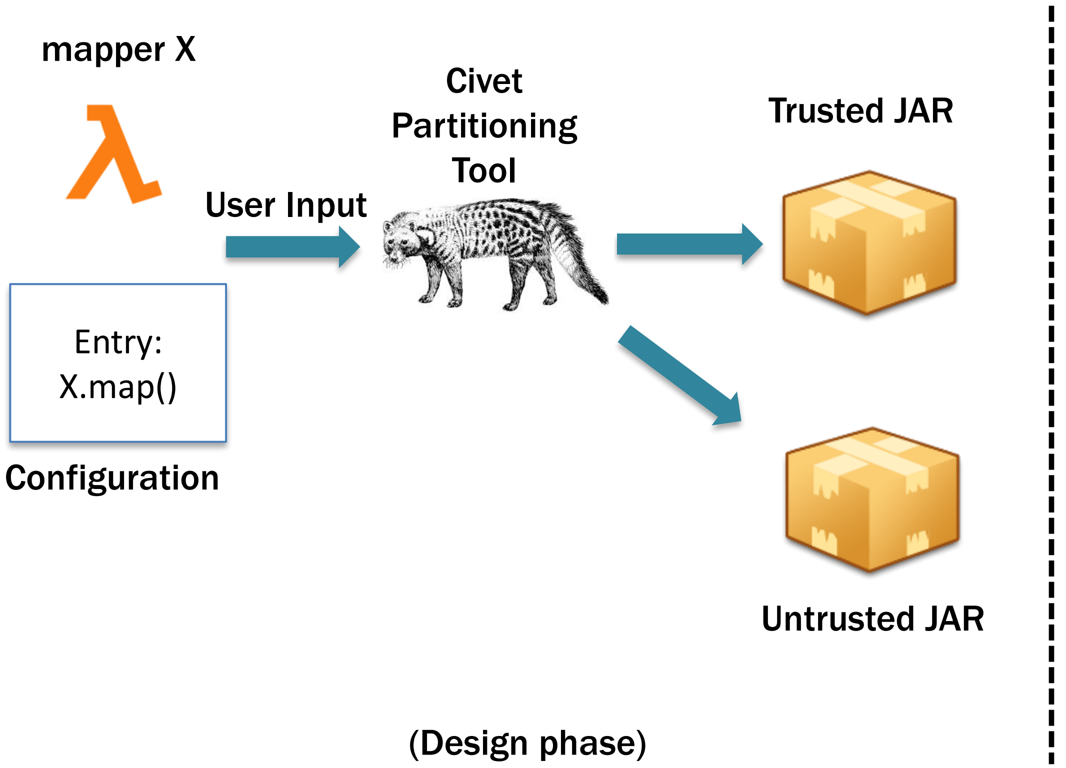
Civet: Partitioning Java-based Applications for Enclaves

- Guided partitioning for experimentation of partition boundary
- White-listing class loading & polymorphism
- Tailored Java runtime for enclave performance patterns (e.g., GC)

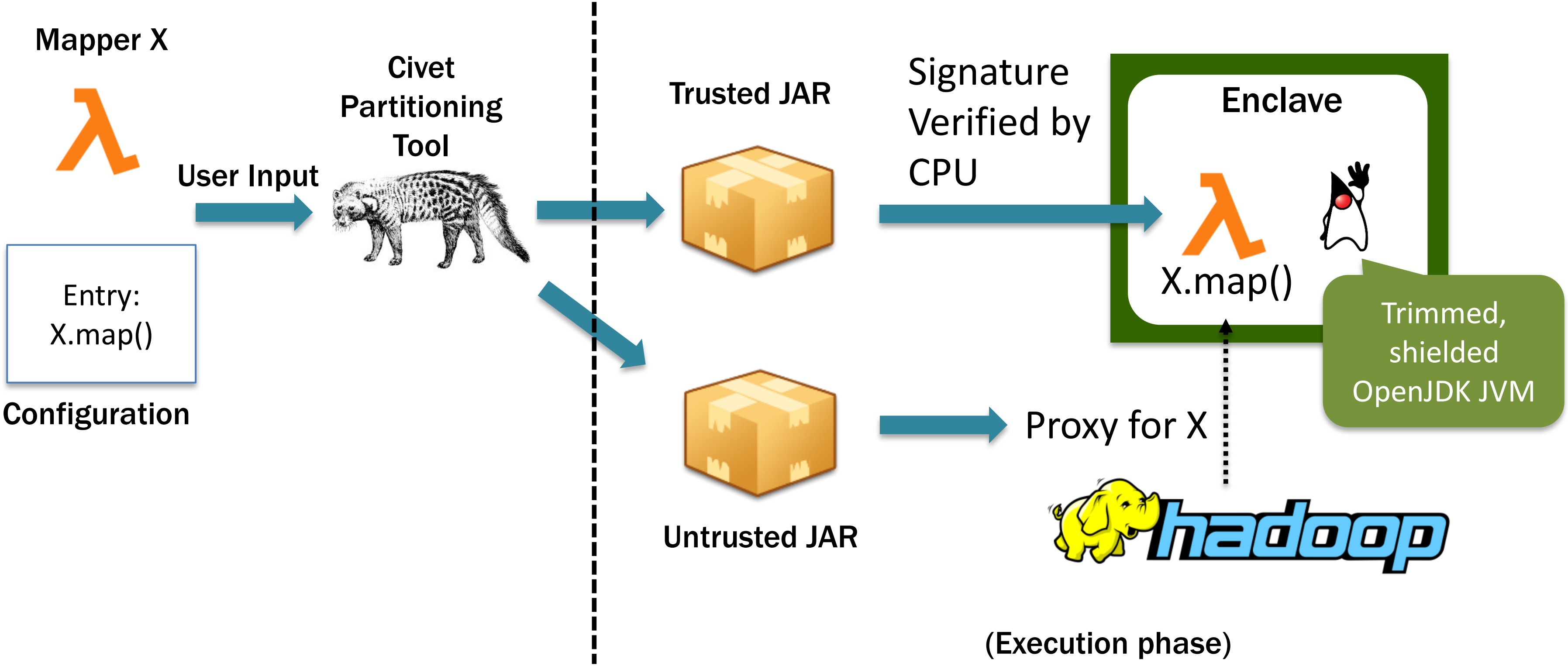
Threat Model



Partitioning Tool + Java Runtime for Enclaves



Partitioning Tool + Java Runtime for Enclaves

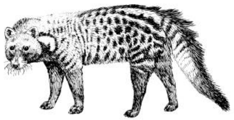


Determining Boundary for TCB

mapper X

User Input

Civet
Partitioning
Tool



Trusted JAR

Entry:
X.map()

Configuration

- Determine effective TCB based on partitioning decisions
- Prevent loading unnecessary code

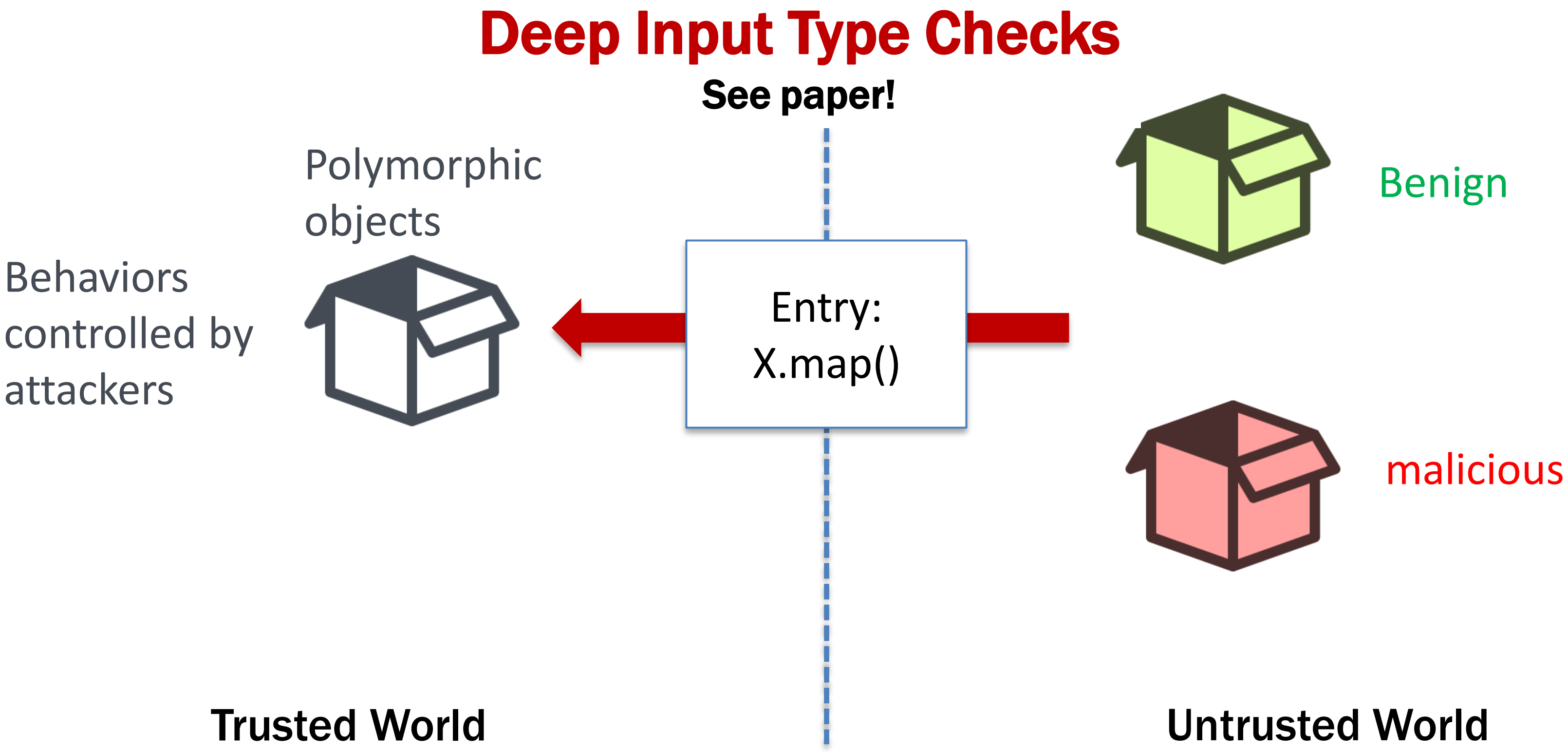
Code Reachability Analysis

(Based on bytecode-level, call graph + points-to analysis)

Class & method shredding

See paper!

Polymorphic Attacks on Enclave Entries

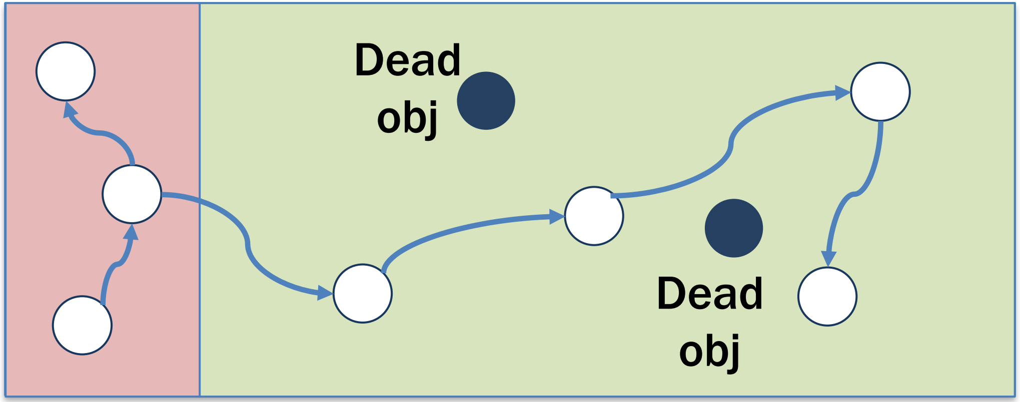


Garbage Collection in Enclaves (1/2)

Insight: Memory overhead in enclaves heavily impacted by cache misses & page faults.

Young Gen.
(Frequently GC'ed)

2-6X
caching overhead



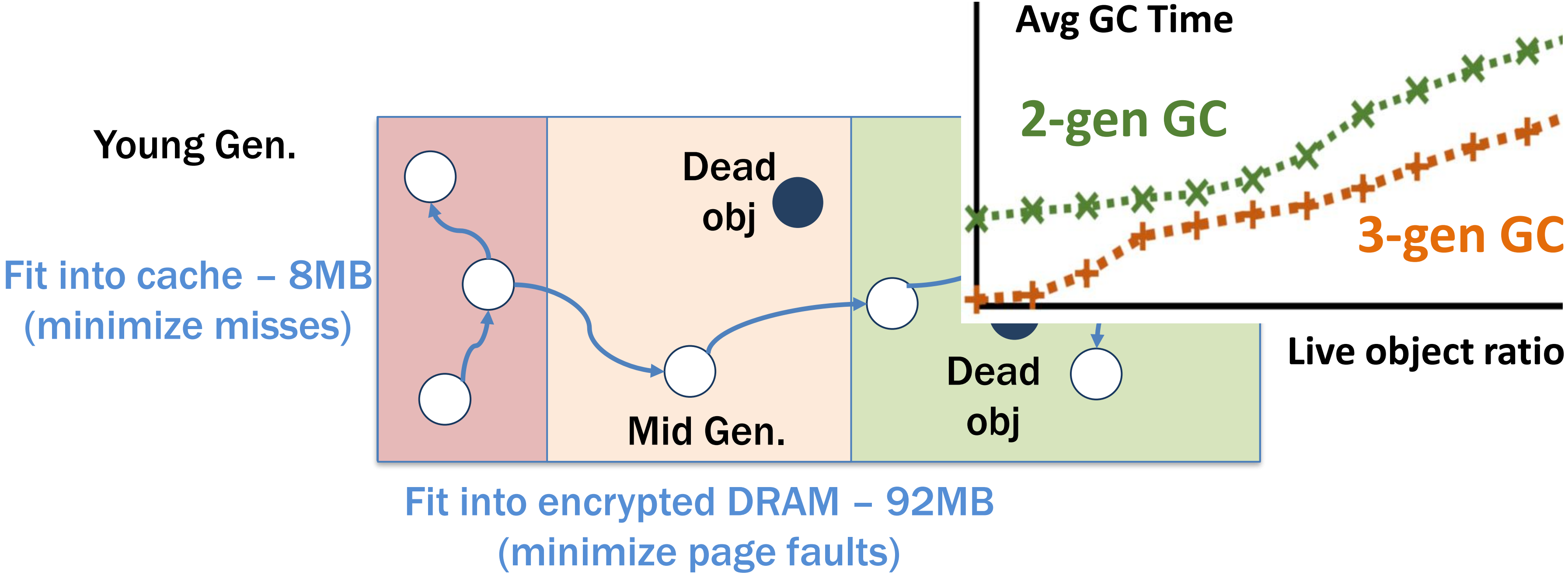
Old Gen.
(Rarely GC'ed)

2-12X
paging overhead

"Mark-and-Sweep"

Garbage Collection in Enclaves (2/2)

Insight: Memory overhead in enclaves heavily impacted by cache misses & page faults.

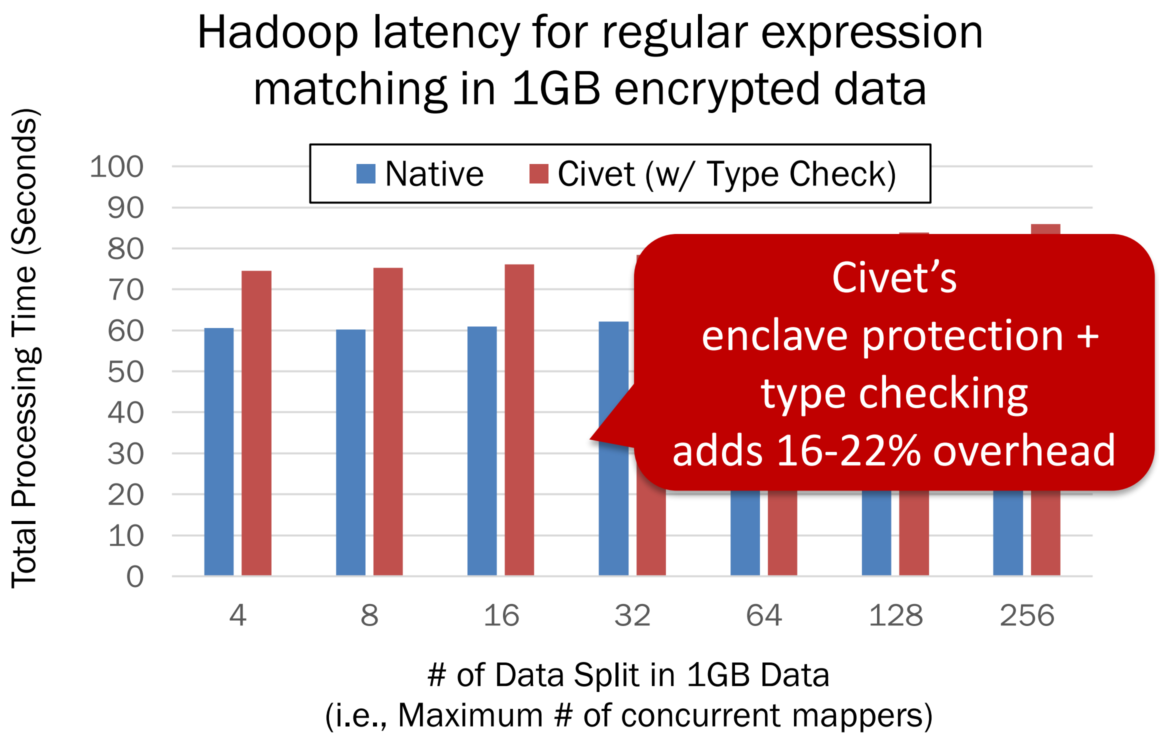


Partitioning Effectiveness + Performance

Hadoop Regular Expression Matching:

Before partitioning:
589K methods, 7.2MLoC

After partitioning:
12K methods, 248KLoC (-96%)



Conclusion

- Java workloads don’t fit into enclave programming paradigms
  - Dynamic and polymorphic behaviors
  - Monolithic runtimes and expensive resource management

- *Civet*: partitioning, refining and hardening with reachability analysis, deep type checking, and enclave-specific runtime design.

Questions or feedback: Chia-Che Tsai <chiache@tamu.edu>