Decoupling Dynamic Information Flow Tracking with a Dedicated Coprocessor

Hari Kannan, Michael Dalton, Christos Kozyrakis

Presenter: Yue Zheng

Yulin Shi

Outline

- Motivation & Background
- Hardware DIFT overview
- DIFT Coprocessor Design
- Prototype System
- Evaluation
- Conclusion
- Discussion points

2

Outline

- Motivation & Background
- Hardware DIFT overview
- DIFT Coprocessor Design
- Prototype System
- Evaluation
- Conclusion
- Discussion points

3

Motivation & Background

Why is the pointer dereference potentially dangerous?

- Dynamic Information Flow Tracking (DIFT)
  - **Tag** data from untrusted source
  - **Track** tainted data propagation
  - **Check** unsafe tainted data usage

Vulnerable C Code

```
int idx = tainted_input;
buffer[idx] = x; // memory corruption
```

Tainted Pointer Dereference

TRAP

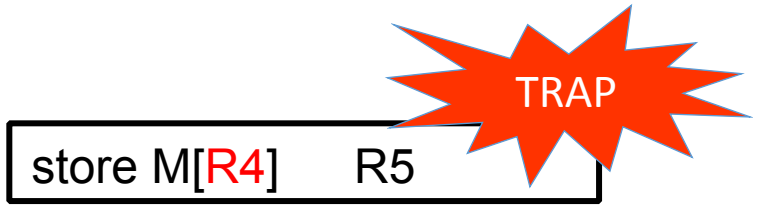
```
R1    &tainted_input
load R2    M[R1]
add R4    R2 + R3
re M[R4]    R5
```

#Reg	Data	Tag
R1	&tainted_input	
R2	idx (tainted_input)	
R3	&buffer	
R4	&buffer + idx	
R5	x	

4

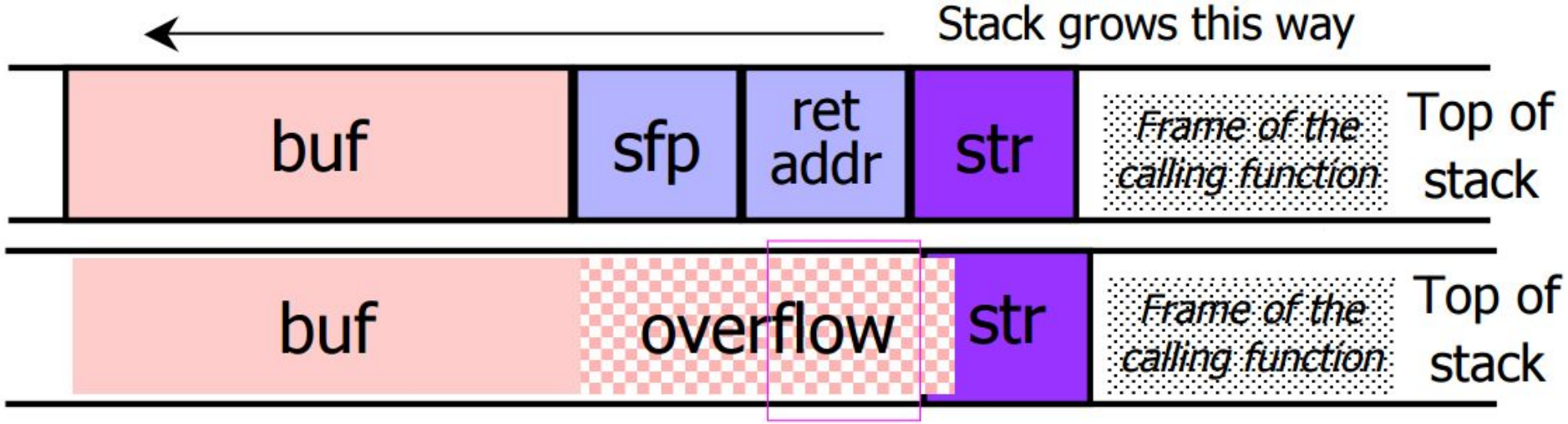
Why is the pointer dereference potentially dangerous?

- Buffer Overflow Attack



Vulnerable C Code

```
char buf[126];
strcpy(buf,str);
```



5

Motivation & Background

- Software DIFT
  - Use Dynamic Binary Translation (DBT) to implement DIFT
    - Avoid recompilation
    - Introduce **significant overheads**
  - Limitation
    - **Incompatible** with self-modifying and multithreaded programs

6

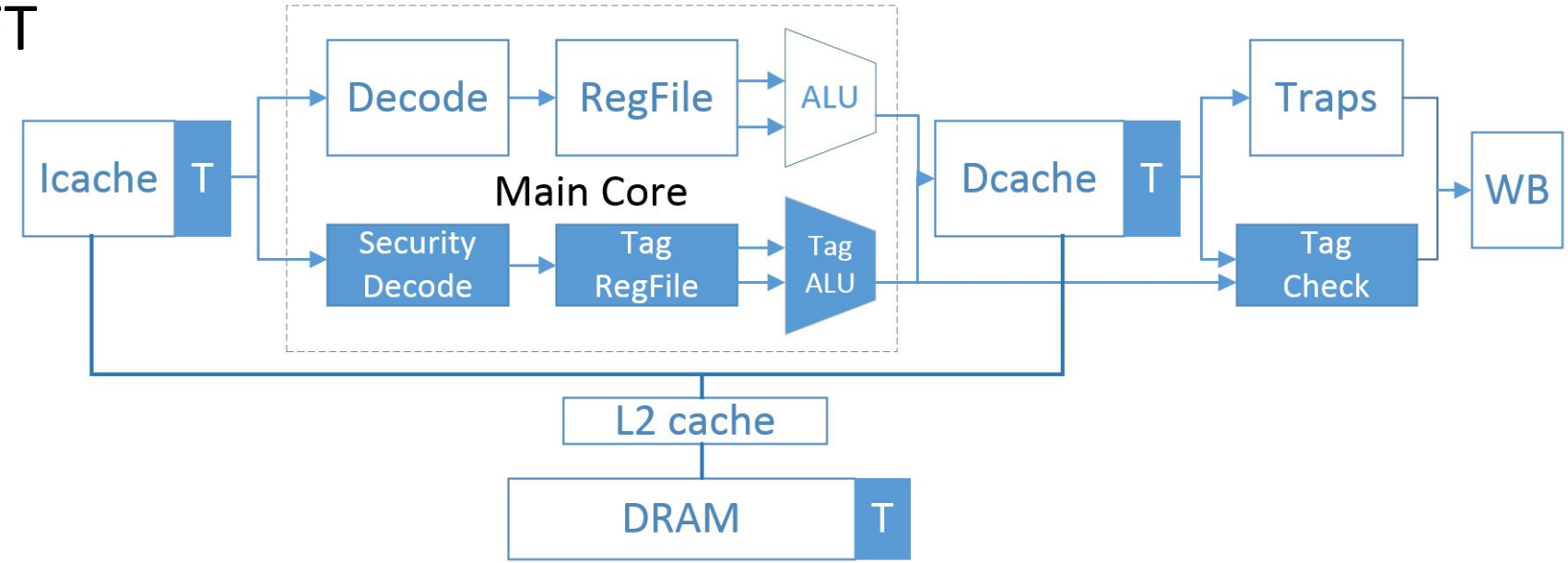
Outline

- Motivation & Background
- Hardware DIFT overview
- DIFT Coprocessor Design
- Prototype System
- Evaluation
- Conclusion
- Discussion points

7

Hardware DIFT overview

- In-core DIFT

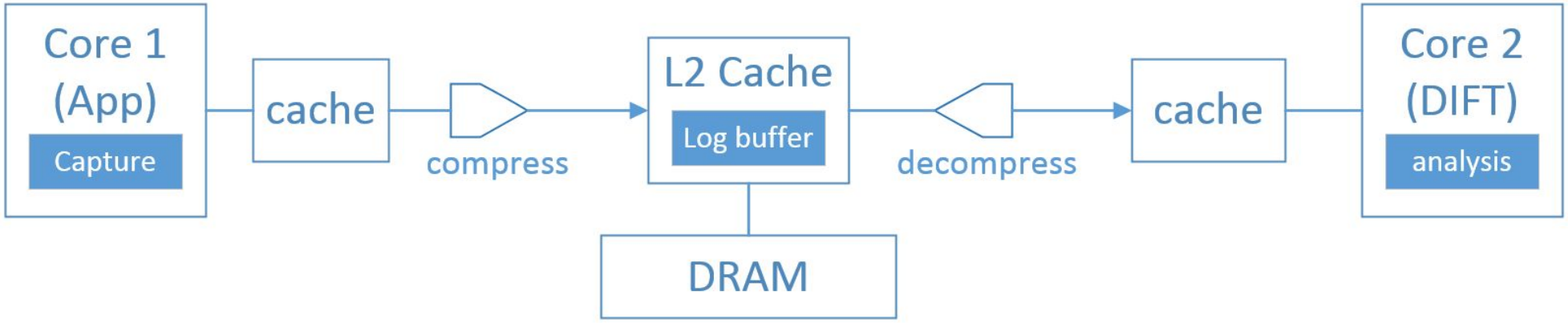


- Minimize runtime overhead
- Require significant modifications to processor structure

8

Hardware DIFT overview

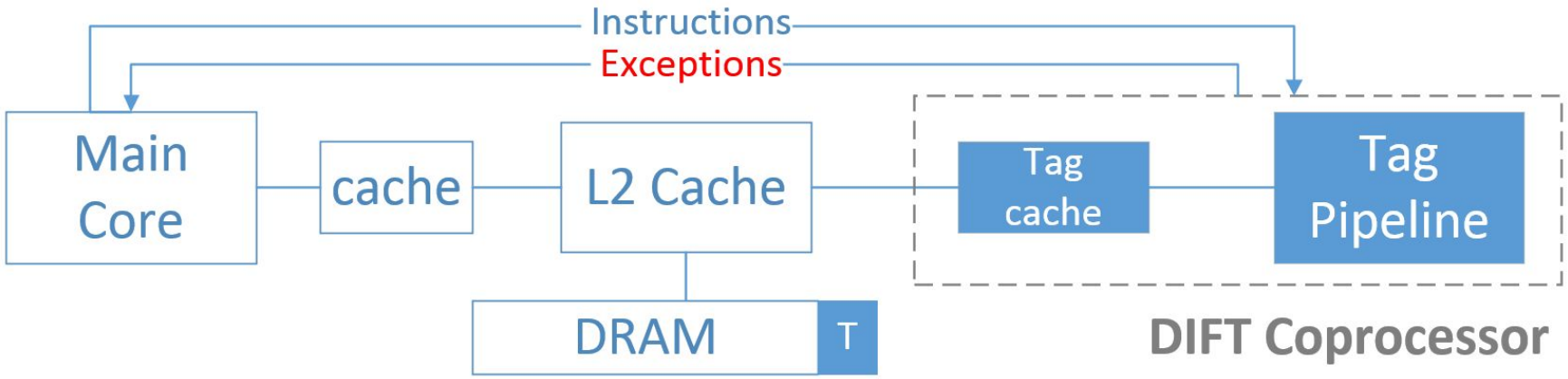
- Offloading DIFT



  - Offer the flexibility of analysis in software
  - Introduce significant overheads
    - halve the throughput, double the power consumption
  - Require pipeline changes

Hardware DIFT overview

- Off-core DIFT



- DIFT synchronizes with main core only on system calls
- Eliminate the changes to processor
- Allow pairing with multiple processor designs

10

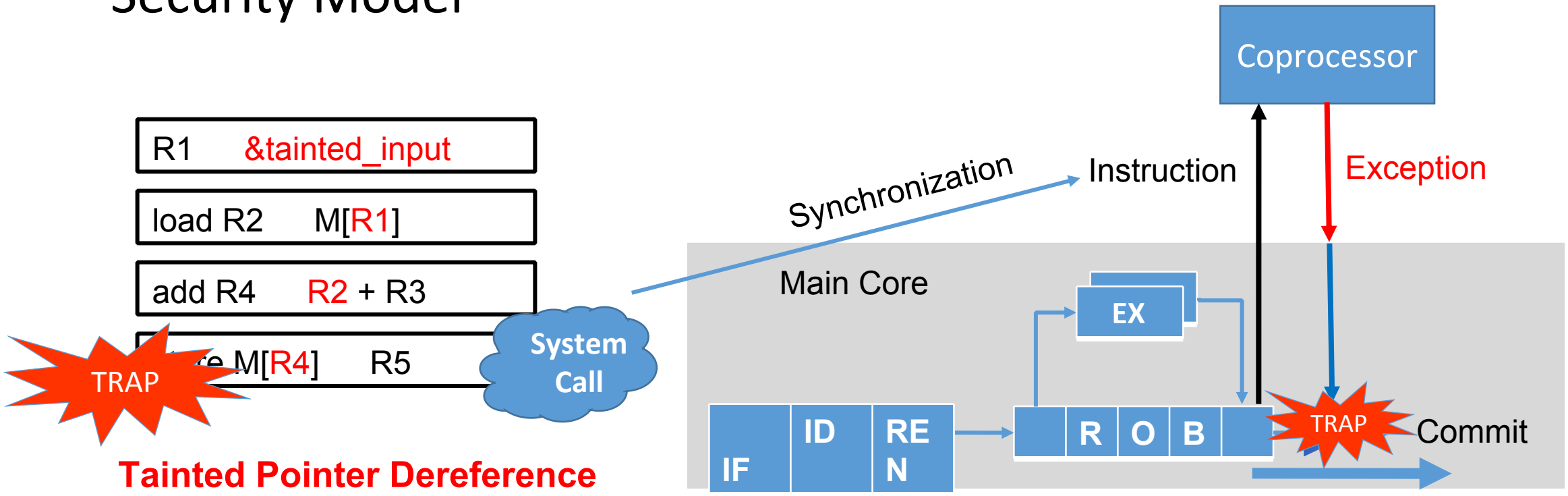
Outline

- Motivation & Background
- Hardware DIFT overview
- DIFT Coprocessor Design
- Prototype System
- Evaluation
- Conclusion
- Discussion points

11

DIFT Coprocessor Design – Security Model

- Security Model



12

DIFT Coprocessor Design - Architecture

Main Core Processor VS Coprocessor

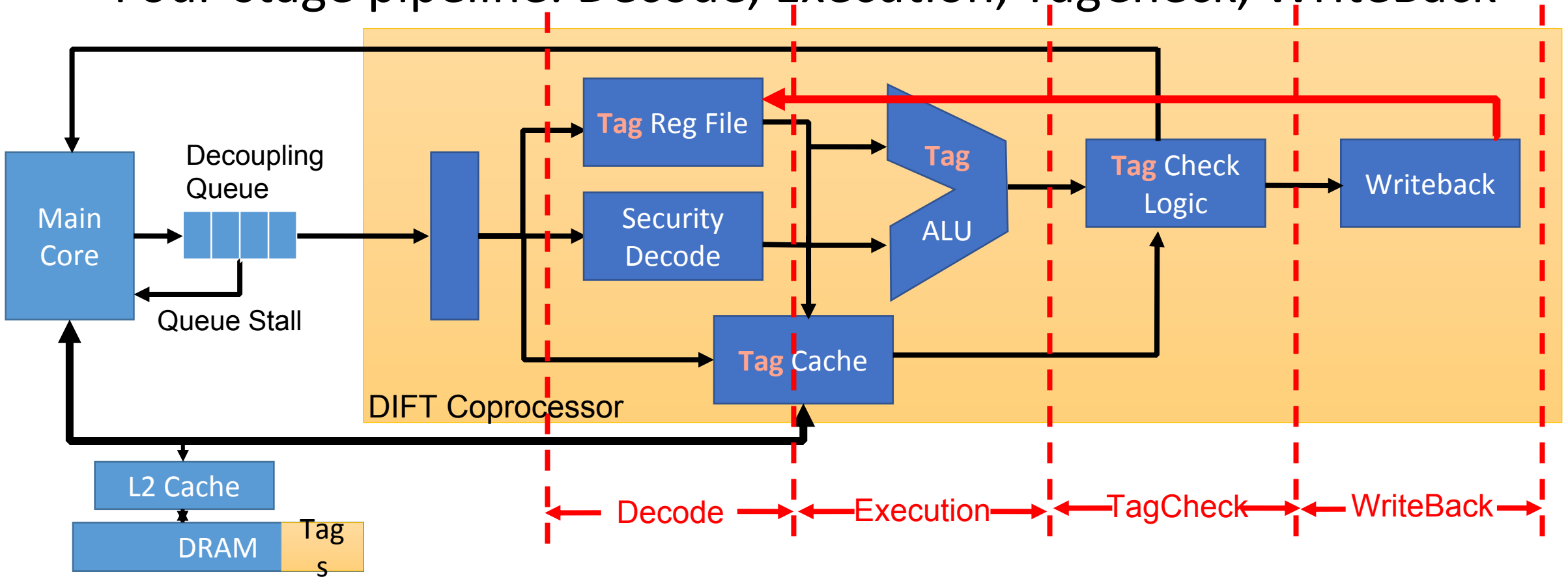
- Handle data
- 32 bits
- Complexity

- Handle **tag** propagation and check
- **4 bits**
- Simple

13

DIFT Coprocessor Design - Architecture

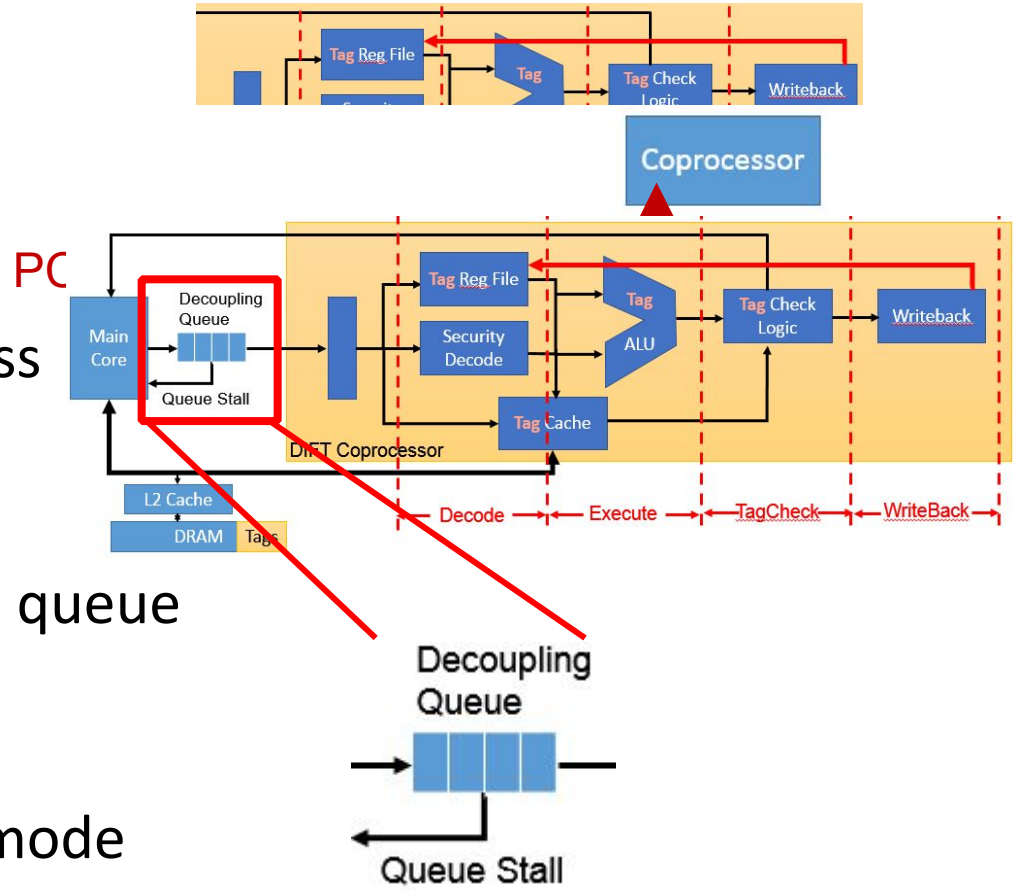
- Four-stage pipeline: Decode, Execution, TagCheck, WriteBack



14

DIFT Coprocessor Design - Interface

- Coprocessor Setup
  - Software control security policies
- Instruction Flow Information
  - PC, Instruction Encoding and Memory Address
- Decoupling
  - Greater or equal processing rate to avoid full queue
- Security Exceptions
  - Asynchronous interrupts and run in trusted mode



15

Outline

- Motivation & Background
- Hardware DIFT overview
- DIFT Coprocessor Design
- Prototype System
- Evaluation
- Conclusion
- Discussion points

16

Prototype

- Hardware
  - SPARC V8 Processor
  - FPGA Board
- Software
  - Gentoo Linux 2.6
- Design Statistics
  - 7.64% area overhead

17

Outline

- Motivation & Background
- Hardware DIFT overview
- DIFT Coprocessor Architecture
- Prototype System
- Evaluation
- Conclusion
- Discussion points

18

Evaluation

- Security Evaluation

Program	Lang.	Attack	Detected Vulnerability
Gzip	C	Directory traversal	Open tainted directory
Tar	C	Directory traversal	Open tainted directory
Scry	PHP	Cross-site scripting	Tainted <script> tag
Htdig	C++	Cross-site scripting	Tainted <script> tag
Polymorph	C	Buffer overflow	Tainted code ptr dereference
Sendmail	C	Buffer overflow	Tainted data ptr dereference
Quotactl syscall	C	User/kernel pointer dereference	Tainted pointer to kernelspace
SUS	C	Format string bug	Tainted '%n' in syslog
WU_FTPD	C	Format string bug	Tainted '%n' in vfprintf

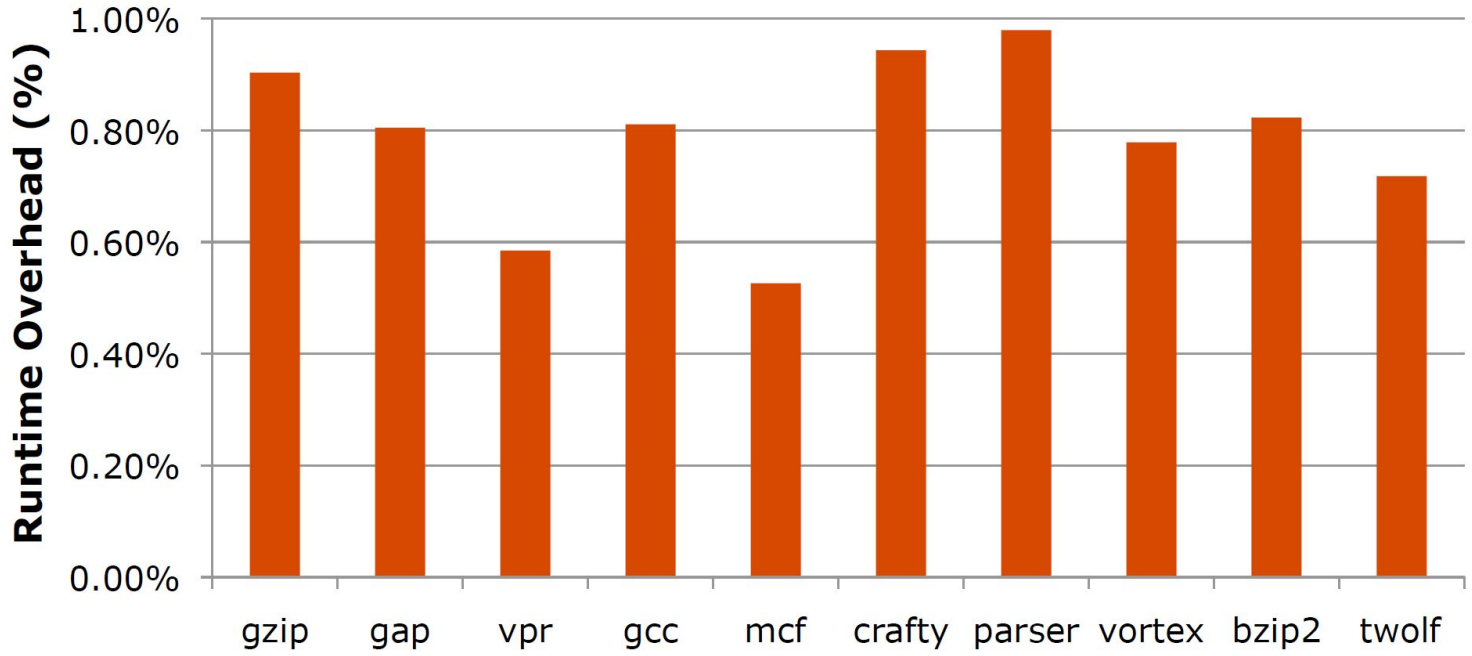
Common Utilities

Servers
Web apps
Kernel code

19

Evaluation

- Performance Evaluation – Execution time



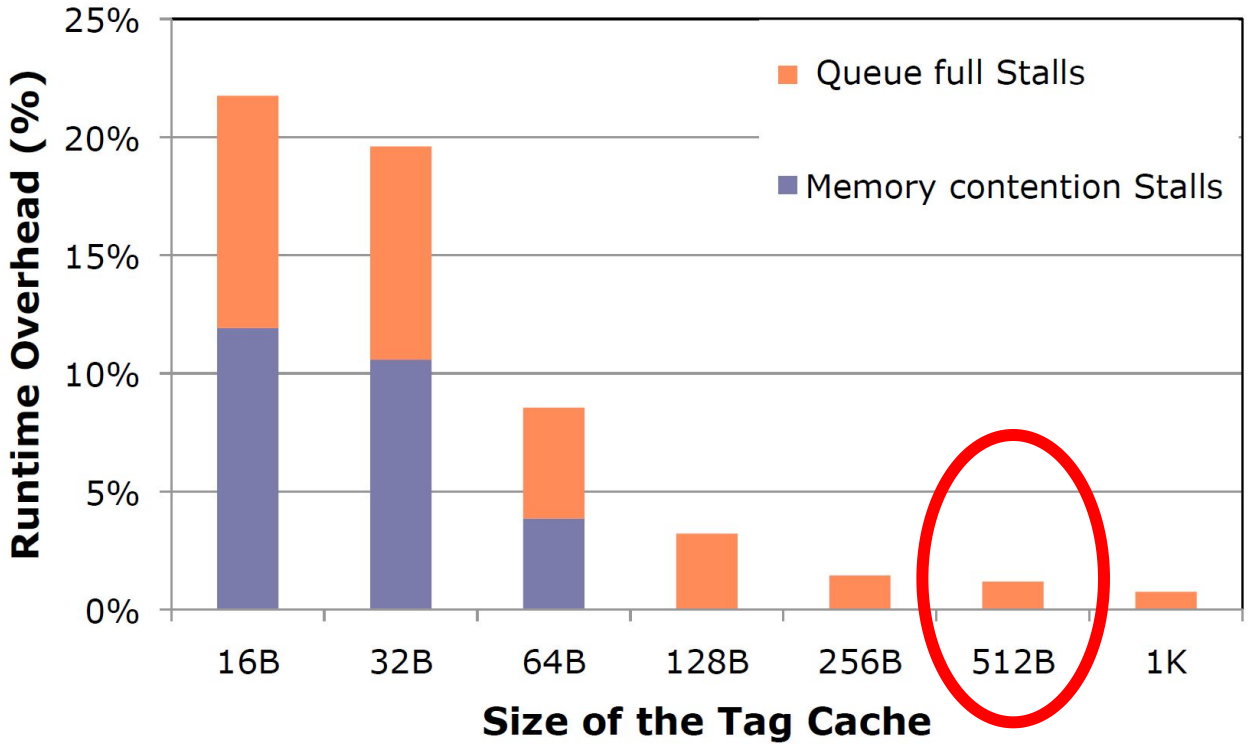
- 512-byte tag cache
- 6-entry queue

Runtime overhead < 1%

20

Evaluation

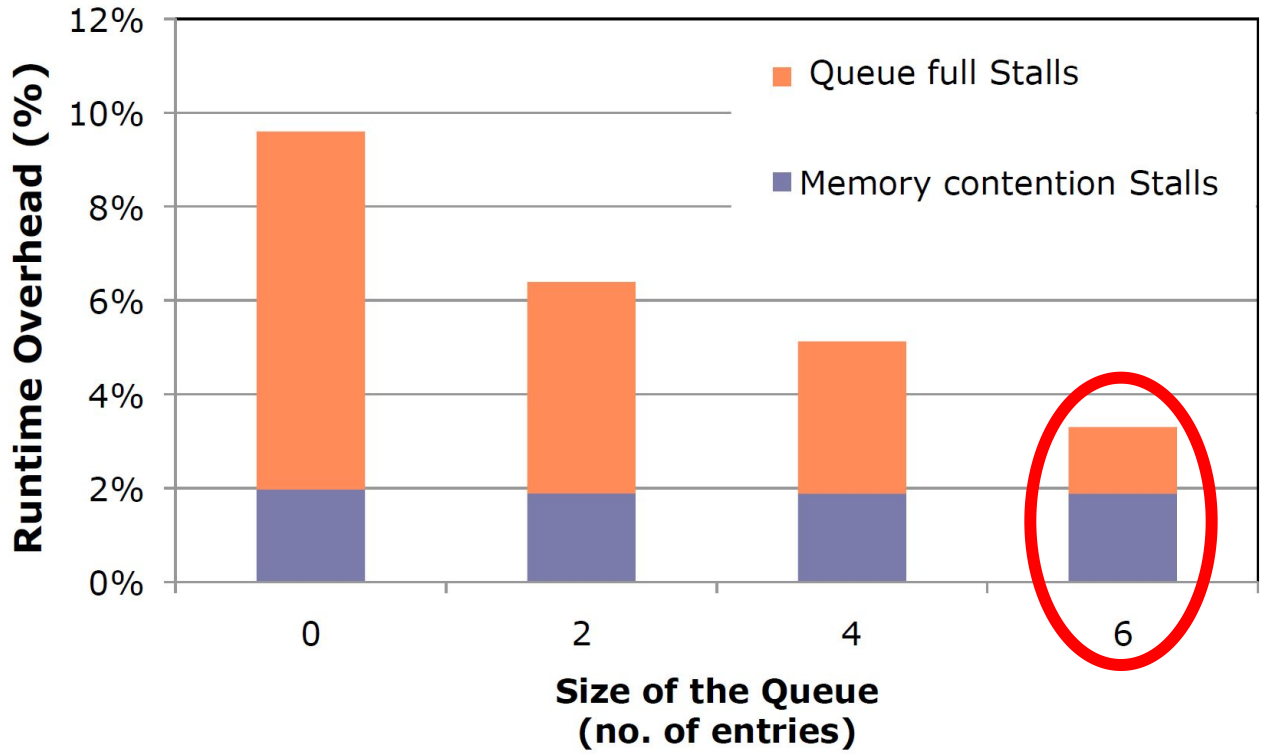
- Performance Evaluation – Scaling the tag cache



21

Evaluation

- Performance Evaluation – Scaling the decoupling queue

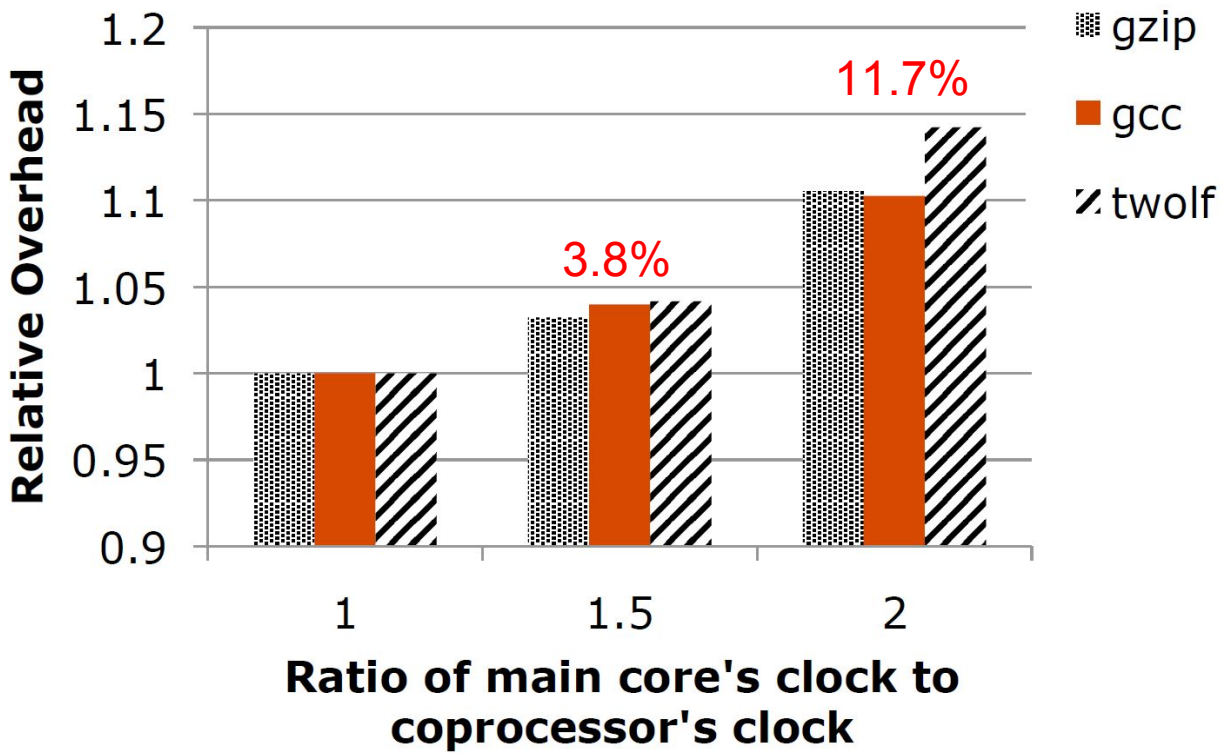


- 16-byte tag cache
  - Tag miss ↑

22

Evaluation

- Processor/Coprocessor Performance Ratio



- 16-entry queue

Can be paired with various main cores

23

Outline

- Motivation & Background
- Hardware DIFT overview
- DIFT Coprocessor Architecture
- Prototype System
- Evaluation
- Conclusion
- Discussion points

24

Conclusion

- DIFT: a promising security technique
- Proposed an off-core, decoupling coprocessor for DIFT
  - Provide the same security features as in-core DIFT
  - Reduce DIFT implementation cost drastically
  - Has low area and performance overheads
- Developed a full-system prototype
  - Protect real-world Linux applications

25

Questions?

26

Debate

- Is a wider-issue coprocessor better than a single-issue coprocessor for 3-way superscalar processors?

- Is it worth to add a checkpoint scheme to DIFT to provide reliable recovery? (A checkpoint scheme allows the system to rollback for recovery)