Lightweight Memory Tracing

Mathias Payer*, Enrico Kravina, Thomas Gross
Department of Computer Science
ETH Zürich, Switzerland

* now at UC Berkeley

Memory Tracing via Memlets

Execute code (***memlets***) for every memory access

A memlet inspects a single memory access based on target ***address***, ***type*** of memory access, ***instruction***, or prior ***state***

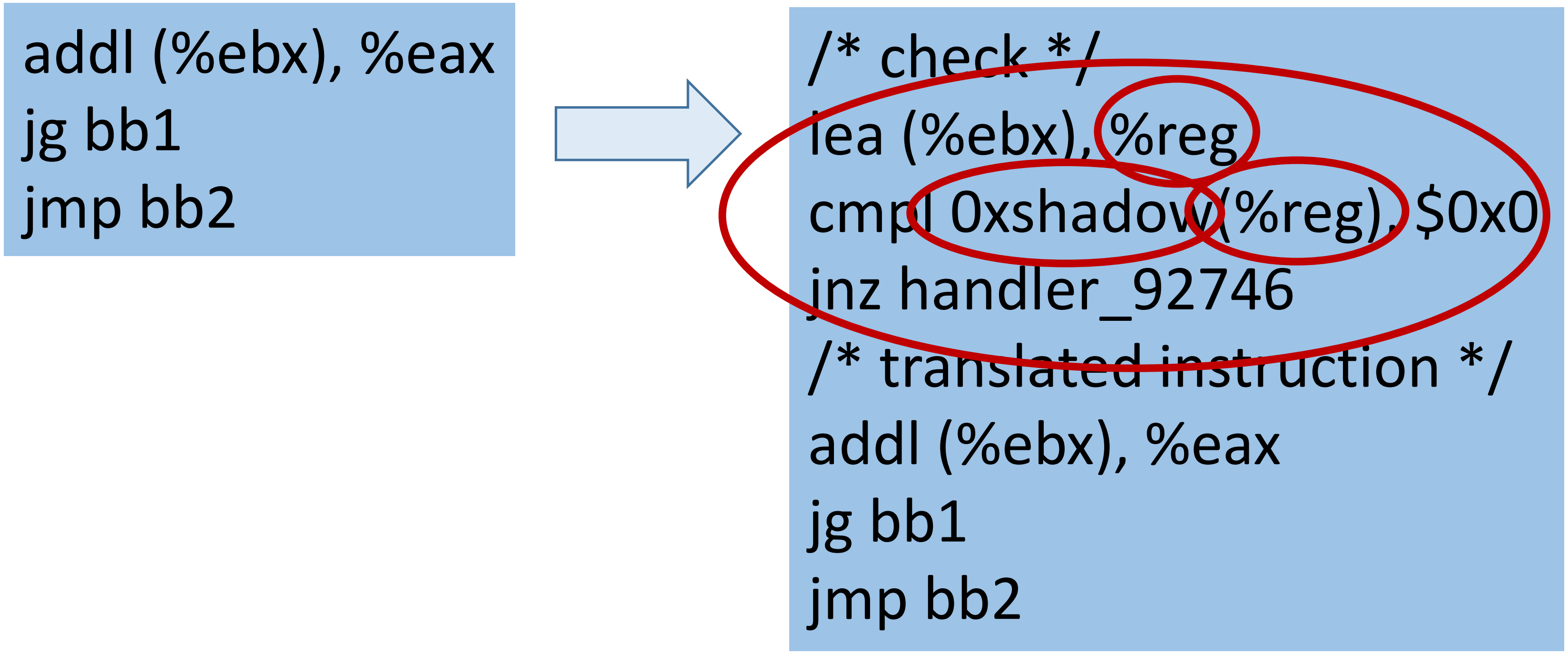
Memory tracing enables detailed memory access logs, debugging of memory accesses, security checks, privacy extensions

Memory Tracing by Example

Binary translation weaves memlets into executed code

memTrace is general, for talk let’s focus on example:

- Unlimited ***watchpoints***: check if R/W watchpoint is set



```
addl (%ebx), %eax
jg bb1
jmp bb2
```

```
/* check */
lea (%ebx), %reg
cmpl 0xshadow(%reg), $0x0
jnz handler_92746
/* translated instruction */
addl (%ebx), %eax
jg bb1
jmp bb2
```

Key to *Lightweight* Memory Tracing

Modern CPUs support multiple ISAs: x86/x86_64

- Most programs still 32-bit x86

Cross-ISA binary translation allows the tracer to use additional hardware available in target ISA:

- Wider address space: isolation & performance
- Additional registers: flexibility & performance

Outline

Motivation and Introduction

Lightweight Memory Tracing

- Requirements
- User-defined Memlets
- Cross-ISA Binary Translation (BT)
- Implementation

Evaluation

Related Work

Conclusion

Tracing Requirements

Flexibility

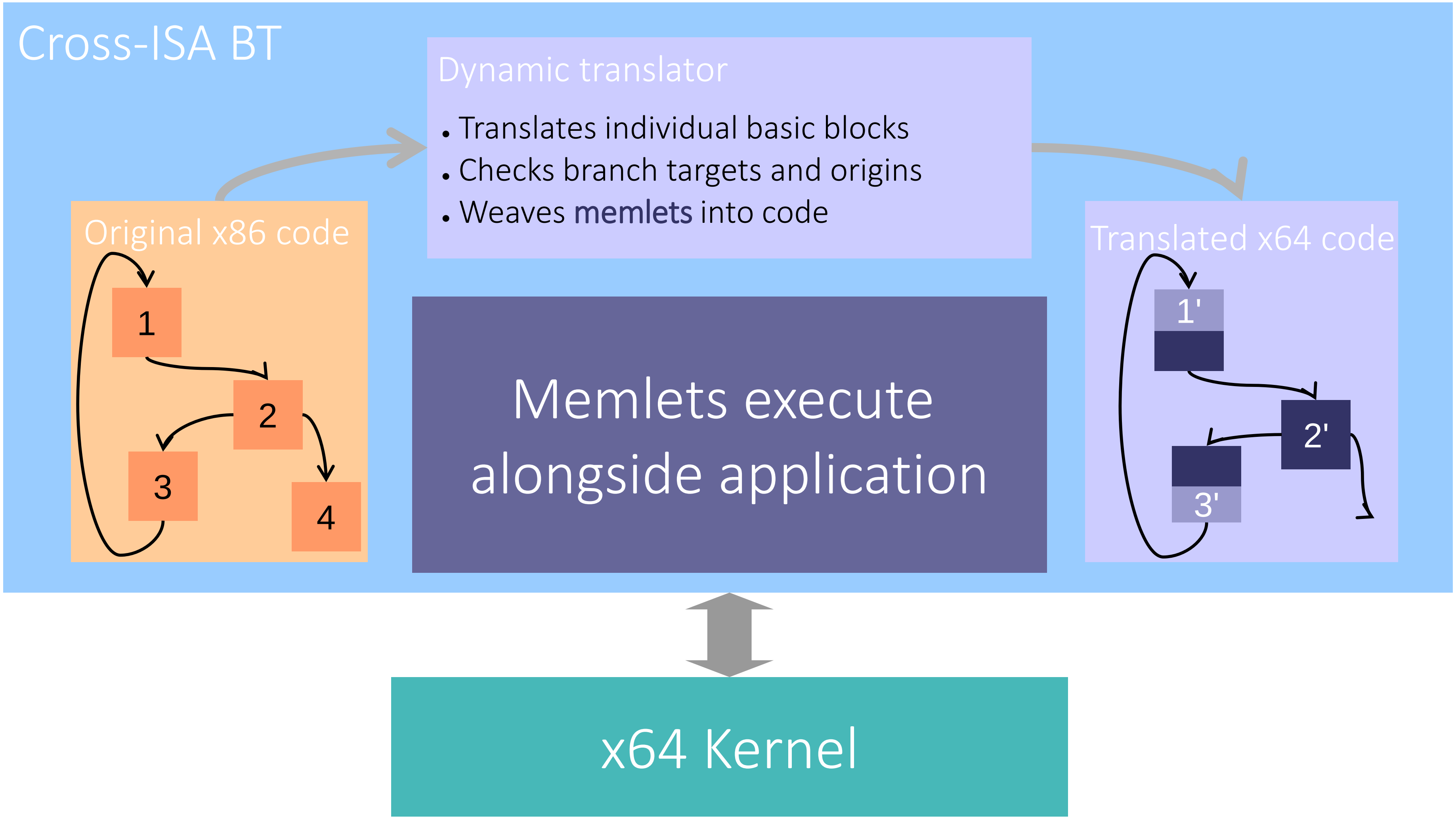
Isolation

Performance

http://blo

http://elie.im

Flexibility through BT



Cross-ISA BT

Original x86 code

Dynamic translator

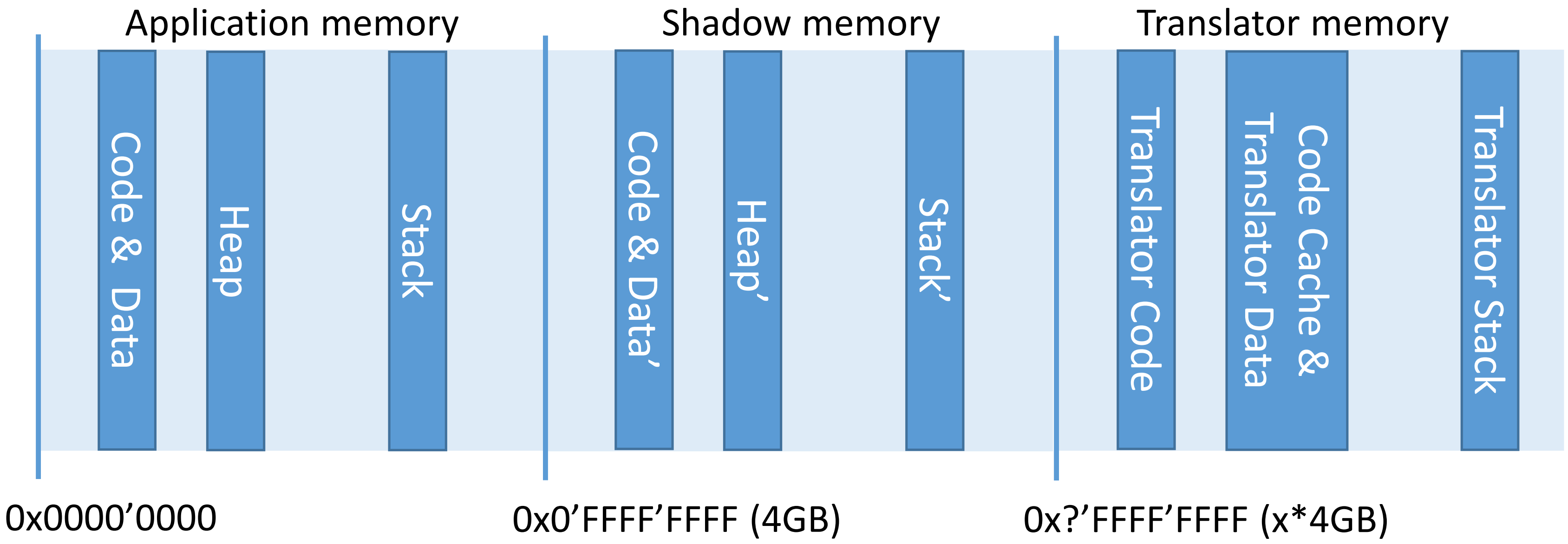
- Translates individual basic blocks
- Checks branch targets and origins
- Weaves **memlets** into code

Translated x64 code

Memlets execute alongside application

x64 Kernel

Isolation: Larger Memory Space



Wider memory space
Isolates tracer from application

Key to Low Overhead

Fast, efficient binary translation

Letting the hardware do most of the work…

- use 64-bit addressing (aligned 4GB blocks)
- keep state in additional/wider registers
- optimize for EFLAGS usage

Implementation

memTrace implementation (open source)

- Cross-ISA translator
- Sample memlets

Small, lean implementation

	Code	**Comments**
memTrace	13,800*	3,300
Memlets	150-200	100-200

*4,900 LOC for the translation tables

Outline

Motivation and Introduction

Lightweight Memory Tracing

Evaluation

- Unlimited Watchpoints
- Safe Memory Allocation

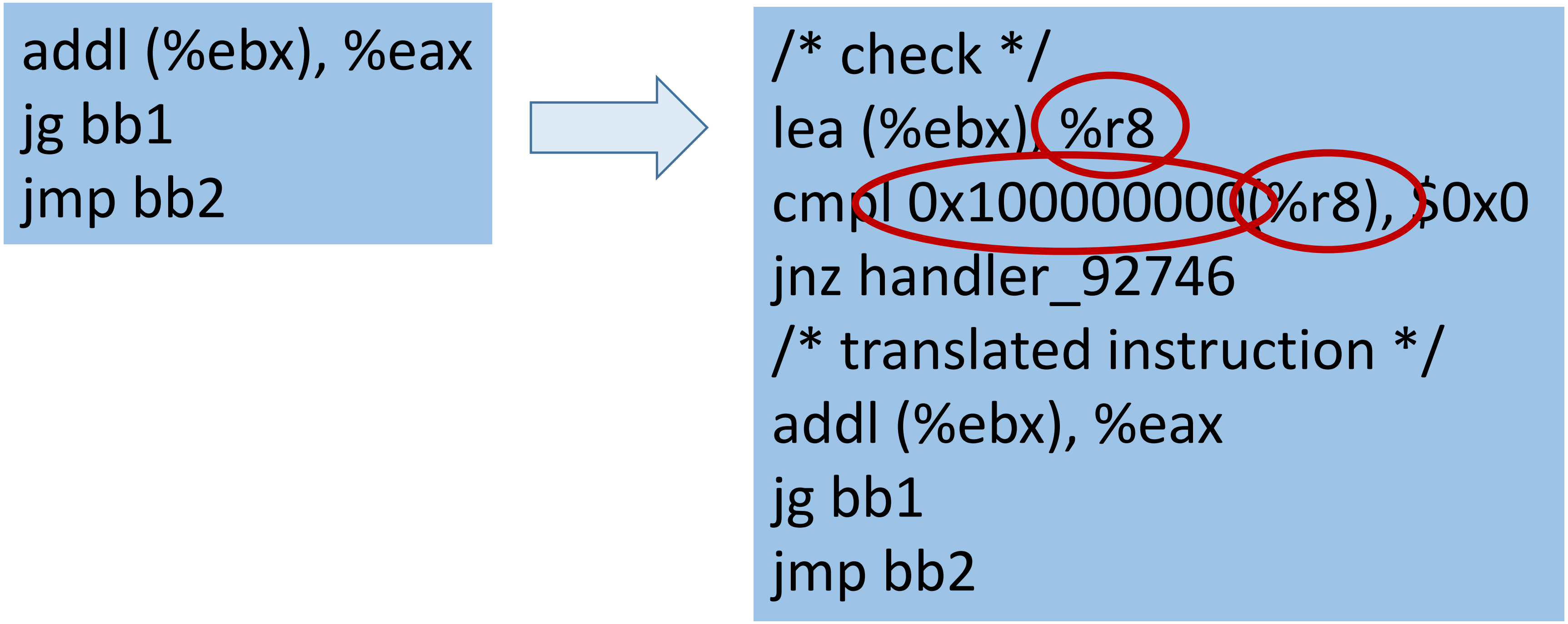
Related Work

Conclusion

Unlimited Watchpoints

Watchpoints trigger on memory reads/writes

Memlet checks if read/write watchpoint is set for each memory access



```
addl (%ebx), %eax
jg bb1
jmp bb2
```

```
/* check */
lea (%ebx), %r8
cmpl 0x100000000(%r8), $0x0
jnz handler_92746
/* translated instruction */
addl (%ebx), %eax
jg bb1
jmp bb2
```

Evaluation Setup

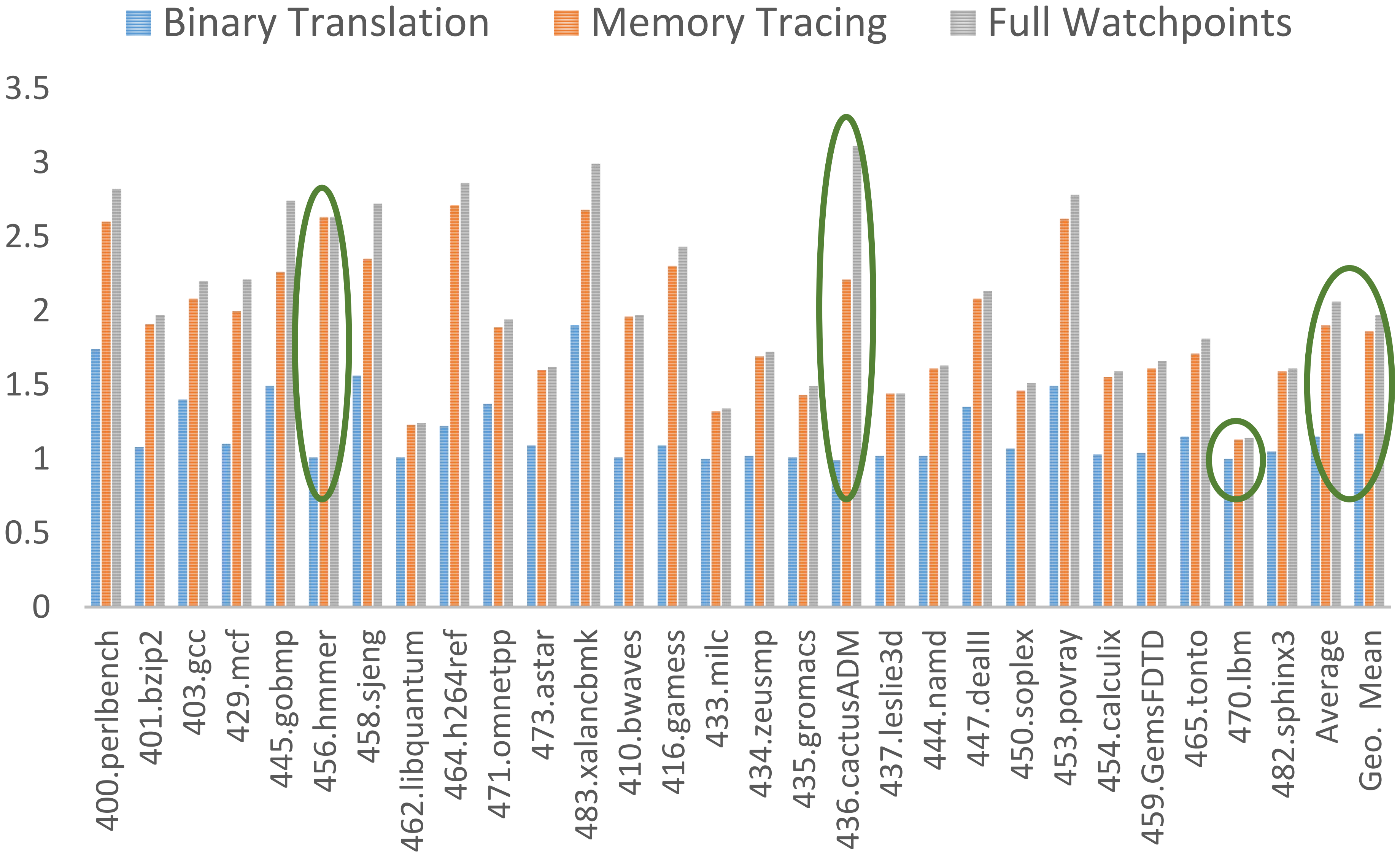
SPEC CPU2006 benchmarks evaluated

- System: Ubuntu 12.04, GCC 4.6.3 (64bit)
- Intel Core i7-2640M @ 2.80GHz, 4GB RAM

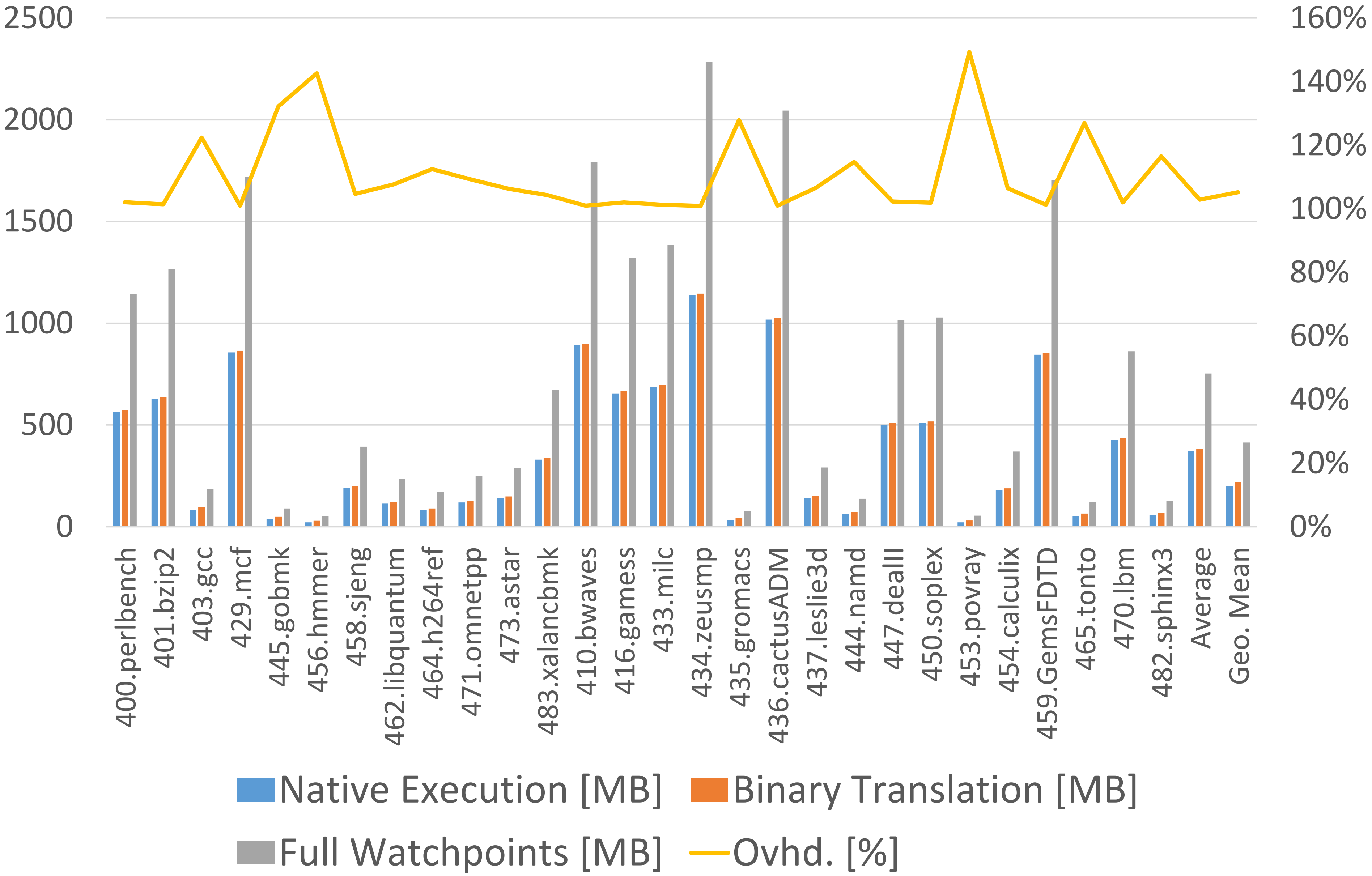
Four configurations:

- Native
- Binary translation (BT) only
- Memory Tracing
- Full Watchpoints

SPEC CPU 2006: Low Perf. Impact



Memory Overhead: 2x



Safe Memory Allocation

Check for use-after-free bugs and heap corruption

Intercept calls to `malloc` and `free`

- Protect metadata of allocated blocks
- Check for read/write accesses to freed blocks until they are reused

Outline

Motivation and Introduction

Lightweight Memory Tracing

Evaluation

Related Work

Conclusion

Related work

Valgrind allows high-level transformations on machine code with performance cost (~7x for nullgrind, ~26x for memcheck)

GDB/Hardware watchpoints allow a limited set of watchpoints with negligible overhead

Limitations of other dynamic tracing systems are (i) limited ISA support, (ii) high overhead, or (iii) limited flexibility

Outline

Motivation and Introduction

Lightweight Memory Tracing

Evaluation

Related Work

Conclusion

Conclusion

memTrace enables lightweight, low-overhead <90% memory inspection for unmodified applications

- Use resources of modern CPUs

Memlets allow user-configurable checks for each memory access

- Flexible framework for memory tracing

Source:

- http://nebelwelt.net/projects/memTrace/
- https://github.com/gannimo/memTrace