Prospect: A Compiler Framework for Speculative Parallelization

Martin Süßkraut, Thomas Knauth, Stefan Weigert, Ute Schiffel, Martin Meinhold, and Christof Fetzer
{martin.suesskraut, thomas.knauth, stefan.weigert, ute.schiffel}@tu-dresden.de, m.meinhold@kontext-e.de, christof.fetzer@tu-dresden.de

CGO 2010, 27th April 2010
Motivation: Parallelization

Example: Runtime Checks

- Buffer overflow: <12x
- Encoded Processing: >40x
Motivation: Parallelization

Example: Runtime Checks

- Buffer overflow: <12x
- Encoded Processing: >40x

Multi-cores do not help

Checks are single threaded
Motivation: Parallelization

Example: Runtime Checks

- Buffer overflow: <12x
- Encoded Processing: >40x

Multi-cores do not help

Checks are single threaded

Predictor/Executor Approach

- Parallelizes runtime checks
- Prospect
**Predictor/Executor Approach**

- **Fast Variant**: optimized or orig. App
- **Slow Variant**: with runtime checks or orig. App
- **Goal:**
  - Runtime of the fast variant
  - Functionality of the slow variant
Predictor/Executor: Details

Predictor – Fast Variant

Core 0

time
Predictor/Executor: Details

Snapshot (epoch boundary) -> Fork Executor

Predictor – Fast Variant
Executor – Slow Variant

Core 0
Core 1

time
Predictor/Executor: Details

Snapshot (epoch boundary)  Fork Executor

Predictor – Fast Variant  Executor – Slow Variant

Core 0
Core 1
Core 2

time
Predictor/Executor: Details

Snapshot (epoch boundary) → Fork Executor

- Predictor – Fast Variant
- Executor – Slow Variant

Core 0
Core 1
Core 2
Predictor/Executor: Details

Snapshots (epoch boundary) → Fork Executor

- Predictor – Fast Variant
- Executor – Slow Variant

Core 0
Core 1
Core 2
Contributions

On-stack-replacement

StackLifter: application wide instrumentation at compile time
Contributions

**On-stack-replacement**

**Speculative variables**

**StackLifter**: application wide instrumentation at compile time

- Manage extra state in slow variant
- Published at SSS'09 [4]
Contributions

- **On-stack-replacement**
  - StackLifter: application wide instrumentation at compile time
  - Manage extra state in slow variant
  - Published at SSS'09 [4]

- **Speculative variables**
  - Similar to Speck [2]
  - But more modular

- **Speculative system calls**
Contributions

On-stack-replacement

Speculative variables

Speculative system calls

StackLifter: application instrumentation at compile time

• Manage extra state in slow variant
• Published at SSS'09 [4]

Focus

• Similar to Speck [2]
• But more modular
StackLifter: Motivation

Snapshot (epoch boundary) ➔ Fork executor

- Predictor – Fast Variant
- Executor – Slow Variant

Core 0
Core 1
Core 2

Time
StackLifter: Motivation

Snapshot (epoch boundary) → Fork executor

- Predictor – Fast Variant
- Executor – Slow Variant

Switch from Fast Variant to Slow Variant
StackLifter: Requirements

Intermediate Code

- LLVM
- After linking
StackLifter: Requirements

Intermediate Code

- LLVM
- After linking

Prepares code bases

- Fast and Slow Variant
- Before instrumentation of variants
StackLifter: Requirements

**Intermediate Code**
- LLVM
- After linking

**Prepares code bases**
- Fast and Slow Variant
- Before instrumentation of variants

**Developer Interface**
- `chkpnt` function
StackLifter: Interface

1:    main () {
2:        foo ();
3:    }
4:    
5:    foo () {
6:        bar ();
7:    }
8:    
9:    bar () {
10:        chkpnt ();
11:    }
StackLifter: Interface

```c
1:   main () {
2:     foo ();
3:   }
4:
5:   foo () {
6:     bar ();
7:   }
8:
9:   bar () {
10:    chkptn ();
11:   }
```

Switch epochs
StackLifter: Difficulty

Fast Variant

1: bar (int b) {
2:   int a = 2;
3:   chkptnt ();
4: }

StackLifter: Difficulty

Fast Variant

1: bar (int b) {
2:  int a = 2;
3:  chkptnt();
4:  }

Slow Variant

1: bar (int b, int c) {
2:  int check = ... 
3:  int a = 2;
4:  chkptnt();
5:  }
StackLifter: Difficulty

Fast Variant
1:    bar (int b) {
2:       int a = 2;
3:         chkpnt ();
4:    }

Slow Variant
1:    bar (int b, int c) {
2:       int check = ...
3:       int a = 2;
4:         chkpnt ();
5:    }

StackLifter: Difficulty

Fast Variant

1: bar (int b) {
2:   int a = 2;
3:   chkptn ();
4: }

Slow Variant

1: bar (int b, int c) {
2:   int check = ...
3:   int a = 2;
4:   chkptn ();
5: }

Translate:
- Local variables + arguments
- Instruction pointer
- Return addresses on the stack
StackLifter: Instrumentation Approach

Original Application
StackLifter: Instrumentation Approach

Fast Variant

Original Application

StackLifter
StackLifter: Instrumentation Approach

Fast Variant

Original Application

StackLifter

Further Instrumentations
StackLifter: Instrumentation Approach

Fast Variant

Original Application

StackLifter

Further Instrumentations

Slow Variant

StackLifter

Further Instrumentations
StackLifter: Instrumentation Approach

Fast Variant

Original Application

StackLifter

Further Instrumentations

Slow Variant

StackLifter

Further Instrumentations

Prospect Runtime Lib
StackLifter: Instrumentation Approach

Fast Variant

Original Application

StackLifter

Further Instrumentations

Parallelized Application

Prospect Runtime Lib

Further Instrumentations

Slow Variant

StackLifter

Original Application
StackLifter: On-Stack Replacement

Fast Variant

```
main
foo
bar
```

Slow Variant

```
main
foo
bar
```
StackLifter: On-Stack Replacement

Fast Variant

main

foo

bar

Slow Variant
StackLifter: On-Stack Replacement

Fast Variant

prospect_main

main

foo

bar

Slow Variant
StackLifter: On-Stack Replacement

- Fast Variant
  - main
  - foo
  - bar
  - chkptnt
- Slow Variant

prospect_main
StackLifter: On-Stack Replacement

prospect_main

Fast Variant

main

foo

bar

Slow Variant

StackLifter Buffer
StackLifter: On-Stack Replacement

Fast Variant

StackLifter Buffer

main

foo

prospect_main

Slow Variant

bar's frame
**StackLifter: On-Stack Replacement**

StackLifter Buffer

<table>
<thead>
<tr>
<th>Fast Variant</th>
<th>Slow Variant</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>main</strong></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>StackLifter Buffer</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>foo's frame</strong></td>
</tr>
<tr>
<td><strong>bar's frame</strong></td>
</tr>
</tbody>
</table>
StackLifter: On-Stack Replacement

Fast Variant

Slow Variant

StackLifter Buffer

- main's frame
- foo's frame
- bar's frame

prospect_main
StackLifter: On-Stack Replacement

Fast Variant

StackLifter Buffer

foo's frame

bar's frame

Slow Variant
StackLifter: On-Stack Replacement

prospect_main

Fast Variant

StackLifter Buffer

Slow Variant

foo

main

bar's frame
StackLifter: On-Stack Replacement

StackLifter Buffer

Fast Variant

Slow Variant

prospect_main

main

foo

bar
StackLifter: On-Stack Replacement

Fast Variant

StackLifter Buffer

Slow Variant

Fast Variant

Slow Variant

StackLifter Buffer
StackLifter: On-Stack Replacement

Fast Variant

Slow Variant

prospect_main

main

foo

bar
StackLifter: Instrumentation of Fast Variant

1: bar () {
2:   int a = 2;
3:   chkptn ();
4: }

1: bar () {
2:   int a = 2;
3:   chkptn ();
4:   ...
5:   ...
6:   ...
7:   ...
8:   ...
9:   ...
10: }

StackLifter
StackLifter: Instrumentation of Fast Variant

```
1: bar () {
2:   int a = 2;
3:   chkptnt ();
4: }
```

StackLifter

```
1: bar () {
2:   int a = 2;
3:   call_01:
3:     chkptnt ();
4: }
```

```
1: bar () {
2:   int a = 2;
3:   call_01:
4:     chkptnt ();
5:     
6:     
7:     
8:     
9:     
10:     
11: }
```
StackLifter: Instrumentation of Fast Variant

Original Code:
1: bar () {
2:   int a = 2;
3:   chkpnt ();
4: }

Instrumented Code:
1: bar () {
2:   int a = 2;
3:   call_01:
4:   chkpnt ();
5:   if (stackLifting)
6:     {
7:       push (a);
8:       push (call_01);
9:       return;
10:   }
11: }

StackLifter
StackLifter: Instrumentation of Slow Variant

1: bar () {
2:   int a = 2;
3:   chkpnt ();
4: }

1: bar () {
2:   int a = 2;
3:   chkpnt ();
4: }

StackLifter
StackLifter:
Instrumentation of Slow Variant

```
1: bar () {
2:   int a = 2;
3:   chkptnt ();
4: }
```

```
1:  bar () {
2:   int a = 2;
3:   chkptnt ();
4: }
```

StackLifter
StackLifter:
Instrumentation of Slow Variant

```c
1: bar () {
2:   int a = 2;
3:   chkptn ();
4: }
```

```c
1: bar () {
2:   int a = 2;
3:   chkptn ();
4: }
5: entry:
6:   int a = 2;
7: call_01:
8:   chkptn ();
9:   restore_01:
10:  a = pop ();
11:  goto call_01;
12: }
```
StackLifter:
Instrumentation of Slow Variant

```
1: bar () {
2:   int a = 2;
3:   chkpnt ();
4: }

1: bar () {
2:   if (stackLifting) {
3:     switch (pop ()) {
4:       case call_01:
5:         goto restore_01;
6:       }
7:   }
8:   else goto entry;
9:   restore_01:
10:  a = pop ();
11:  goto call_01;
12:  entry:
13:  int a = 2;
14:  call_01:
15:  chkpnt ();
16: }
```
StackLifter: Features

Instrumentations do not need to be aware of StackLifter

Transparent
StackLifter: Features

- **Transparent**
  - Instrumentations do not need to be aware of StackLifter

- **Completeness**
  - Indirect Calls
  - Support arbitrary LLVM
StackLifter: Features

- **Transparent**
  - Instrumentations do not need to be aware of StackLifter

- **Completeness**
  - Indirect Calls
  - Support arbitrary LLVM

- **Fast**
  - Restrict StackLifter with Call Graph Analysis
Evaluation: Overview

<table>
<thead>
<tr>
<th>Speedup</th>
<th>Of Prospect:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• Out-of-bounds checker</td>
</tr>
<tr>
<td></td>
<td>• FastAssert</td>
</tr>
</tbody>
</table>
Evaluation: Overview

- Speedup:
  - Out-of-bounds checker
  - FastAssert

- Overhead:
  - Of StackLifter
Evaluation: Overview

- **Speedup**:
  - Out-of-bounds checker
  - FastAssert

- **Overhead**:
  - Of StackLifter

- **Setup**:
  - Intel Xeon 8-core CPU
  - 6 benchmark applications
Evaluation: Out-of-bounds

![Graph showing speedup for different benchmarks with Prospects and upper bounds.](image)
Evaluation: FastAssert

![Diagram showing runtime and speedup comparisons with and without FastAssert]

- Runtime in s (log scale)
- Speedup

- Boost words without assertions: 0.2
- Boost words with assertions without FastAssert: 87.0
- Boost words with FastAssert: 12.2
- Upper bound speedup: 7.2
Evaluation: StackLifter (Vacation benchmark)
# Comparison

<table>
<thead>
<tr>
<th></th>
<th><strong>Prospect</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Instrumentation</strong></td>
<td>compiler</td>
</tr>
<tr>
<td><strong>Slow variant</strong></td>
<td>+</td>
</tr>
<tr>
<td><strong>Fast variant</strong></td>
<td>+</td>
</tr>
<tr>
<td><strong>Application Wide</strong></td>
<td>yes</td>
</tr>
<tr>
<td><strong>Syscall support</strong></td>
<td>speculative</td>
</tr>
<tr>
<td><strong>Developer interface</strong></td>
<td><strong>chkpnt</strong></td>
</tr>
<tr>
<td><strong>Additional state in Slow Variant</strong></td>
<td>speculative</td>
</tr>
</tbody>
</table>
# Comparison

<table>
<thead>
<tr>
<th></th>
<th>Prospect</th>
<th>SuperPin [1]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Instrumentation</strong></td>
<td>compiler</td>
<td>DBI</td>
</tr>
<tr>
<td><strong>Slow variant</strong></td>
<td>+</td>
<td>o</td>
</tr>
<tr>
<td><strong>Fast variant</strong></td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td><strong>Application Wide</strong></td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td><strong>Syscall support</strong></td>
<td>speculative</td>
<td>replay only</td>
</tr>
<tr>
<td><strong>Developer interface</strong></td>
<td><strong>chkpnt</strong></td>
<td>transparent</td>
</tr>
<tr>
<td><strong>Additional state in Slow Variant</strong></td>
<td>speculative</td>
<td>non-speculative</td>
</tr>
</tbody>
</table>
## Comparison

<table>
<thead>
<tr>
<th></th>
<th><strong>Prospect</strong></th>
<th><strong>SuperPin</strong> [1]</th>
<th><strong>Speck</strong> [2]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instrumentation</td>
<td>compiler</td>
<td>DBI</td>
<td>DBI</td>
</tr>
<tr>
<td>Slow variant</td>
<td>+</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>Fast variant</td>
<td>+</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Application Wide</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Syscall support</td>
<td>speculative</td>
<td>replay only</td>
<td>speculative</td>
</tr>
<tr>
<td>Developer interface</td>
<td><strong>chkpnt</strong></td>
<td>transparent</td>
<td>transparent</td>
</tr>
<tr>
<td>Additional state in Slow Variant</td>
<td>speculative</td>
<td>non-speculative</td>
<td>non-speculative</td>
</tr>
</tbody>
</table>
# Comparison

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Instrumentation</td>
<td>compiler</td>
<td>DBI</td>
<td>DBI</td>
<td>compiler</td>
</tr>
<tr>
<td>Slow variant</td>
<td>+</td>
<td>o</td>
<td>o</td>
<td>+</td>
</tr>
<tr>
<td>Fast variant</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Application Wide</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Syscall support</td>
<td>speculative</td>
<td>replay only</td>
<td>speculative</td>
<td>forbidden</td>
</tr>
<tr>
<td>Developer interface</td>
<td>chkpt</td>
<td>transparent</td>
<td>transparent</td>
<td>FastTrack region</td>
</tr>
<tr>
<td>Additional state in Slow Variant</td>
<td>speculative</td>
<td>non-speculative</td>
<td>non-speculative</td>
<td>no</td>
</tr>
</tbody>
</table>
Conclusion

Prospect

- Application wide instrumentation
- At compile time
Conclusion

Prospect

• Application wide instrumentation
• At compile time

StackLifter

Switch from Fast variant to Slow Variant
Conclusion

Prospect

- Application wide instrumentation
- At compile time

StackLifter

Switch from Fast variant to Slow Variant

On-stack replacement

- Runtime optimization
- Dynamic updates
- Need bi-directionality
References


StackLifter: On-Stack Replacement

StackLifter Buffer

Fast Variant

- main
- foo
- bar
- chkptnt

Slow Variant

- main
- foo
- bar
- chkptnt

prospect_main

main's frame

foo's frame

bar's frame
Evaluation: Out-of-bounds (1/2)