TECHNISCHE UNIVERSITÄT DRESDEN

Department of Computer Science Institute for System Architecture, Systems-Engineering Group

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

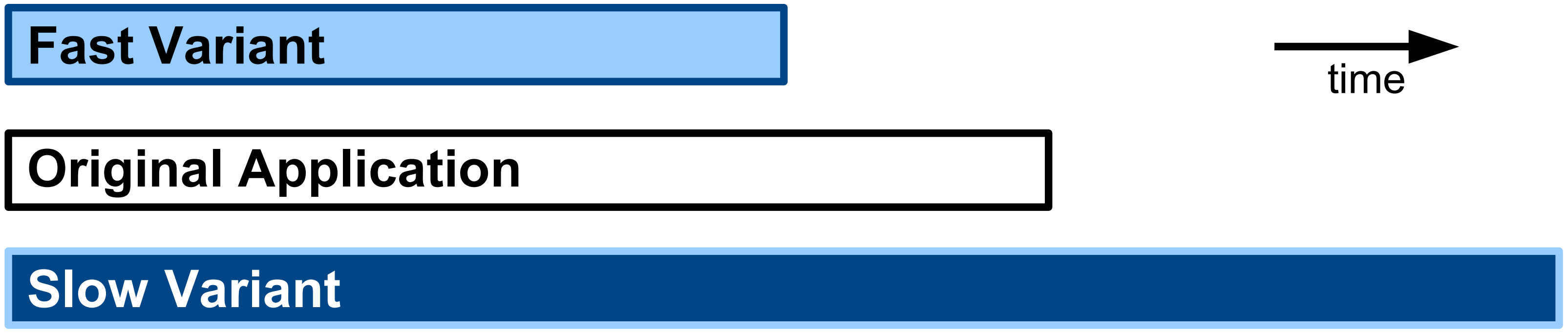
Multi-cores
do not help

Checks are single threaded

Predictor/Executor
Approach

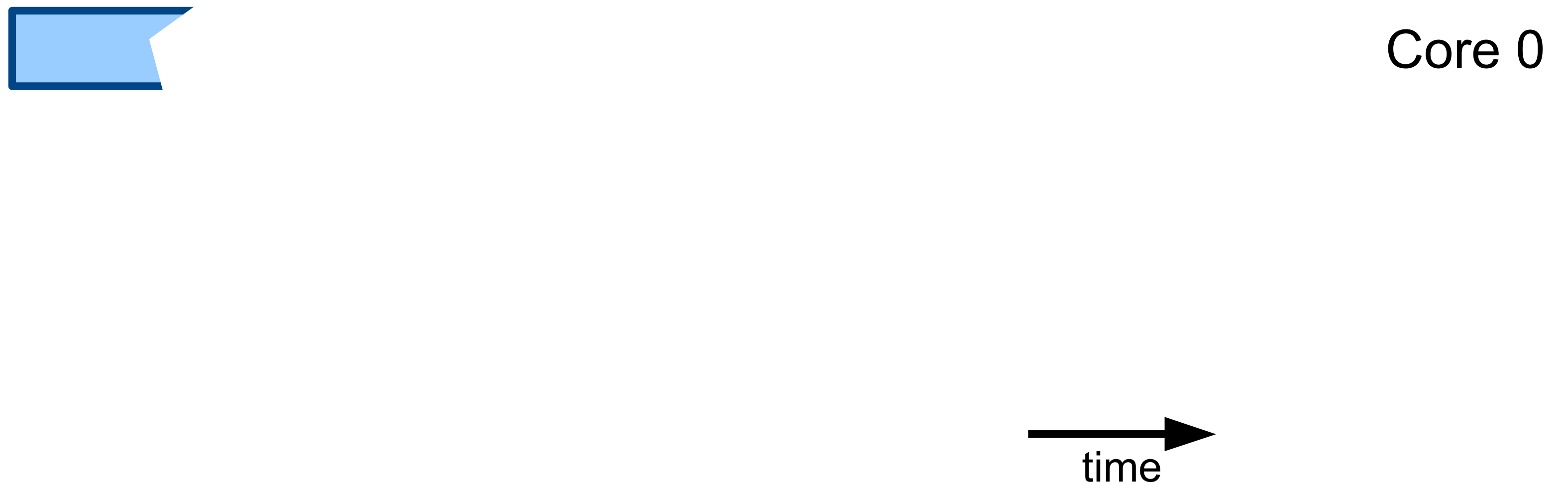
- Parallelizes runtime checks
- ***Prospect***

Predictor/Executor Approach



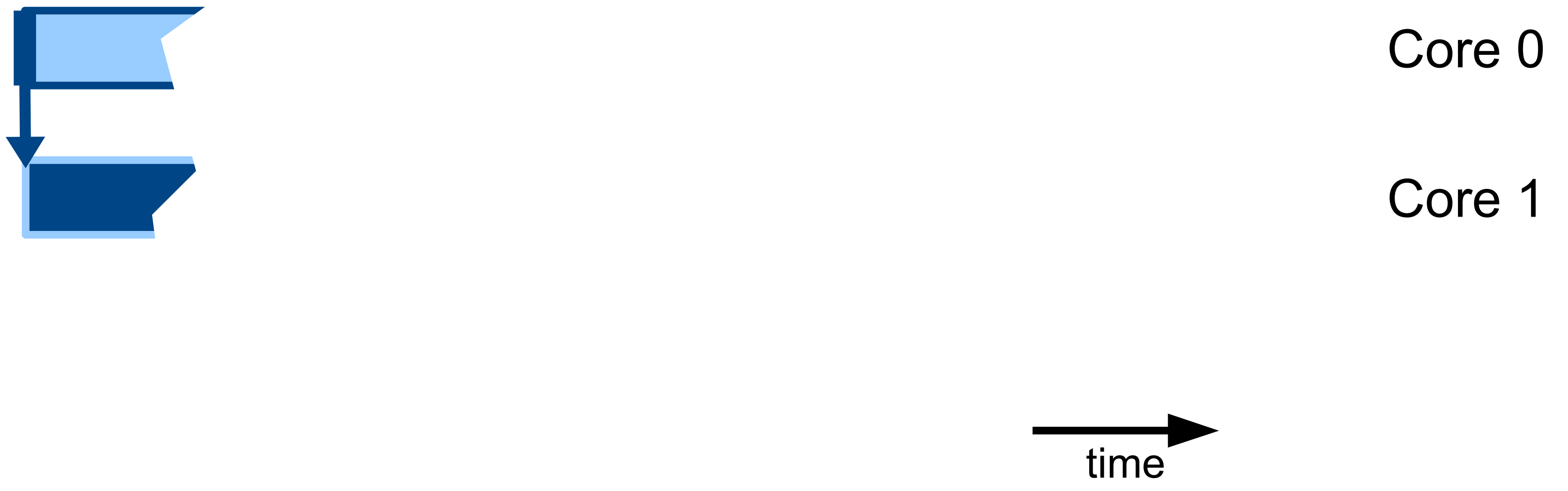
- Fast Variant: optimized or orig. App
- Slow Variant: with runtime checks or orig. App
- Goal:
  - Runtime of the fast variant
  - Functionality of the slow variant

Predictor/Executor: Details



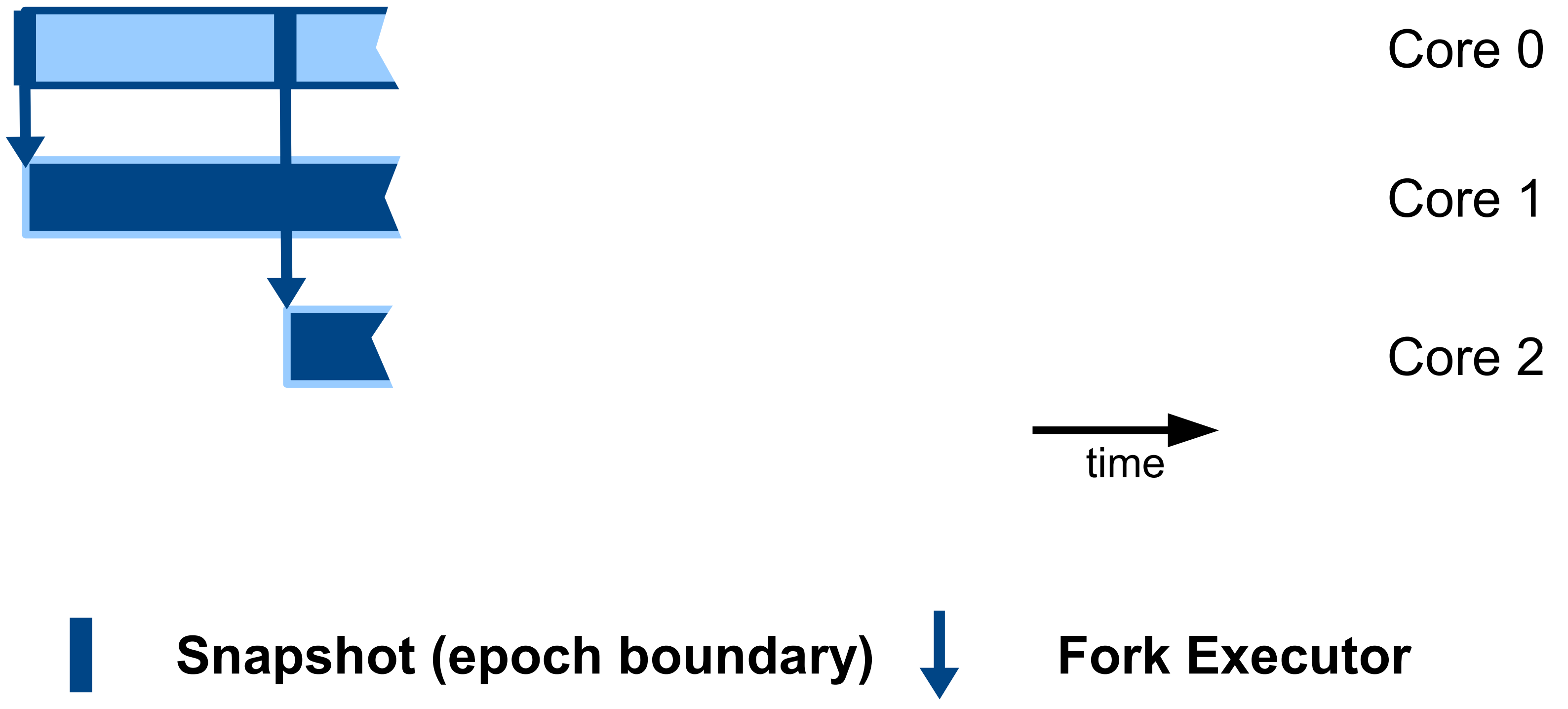
🟦 **Predictor – Fast Variant**

Predictor/Executor: Details



▮	**Snapshot (epoch boundary)**	↓	**Fork Executor**
▭	**Predictor – Fast Variant**	■	**Executor – Slow Variant**

Predictor/Executor: Details



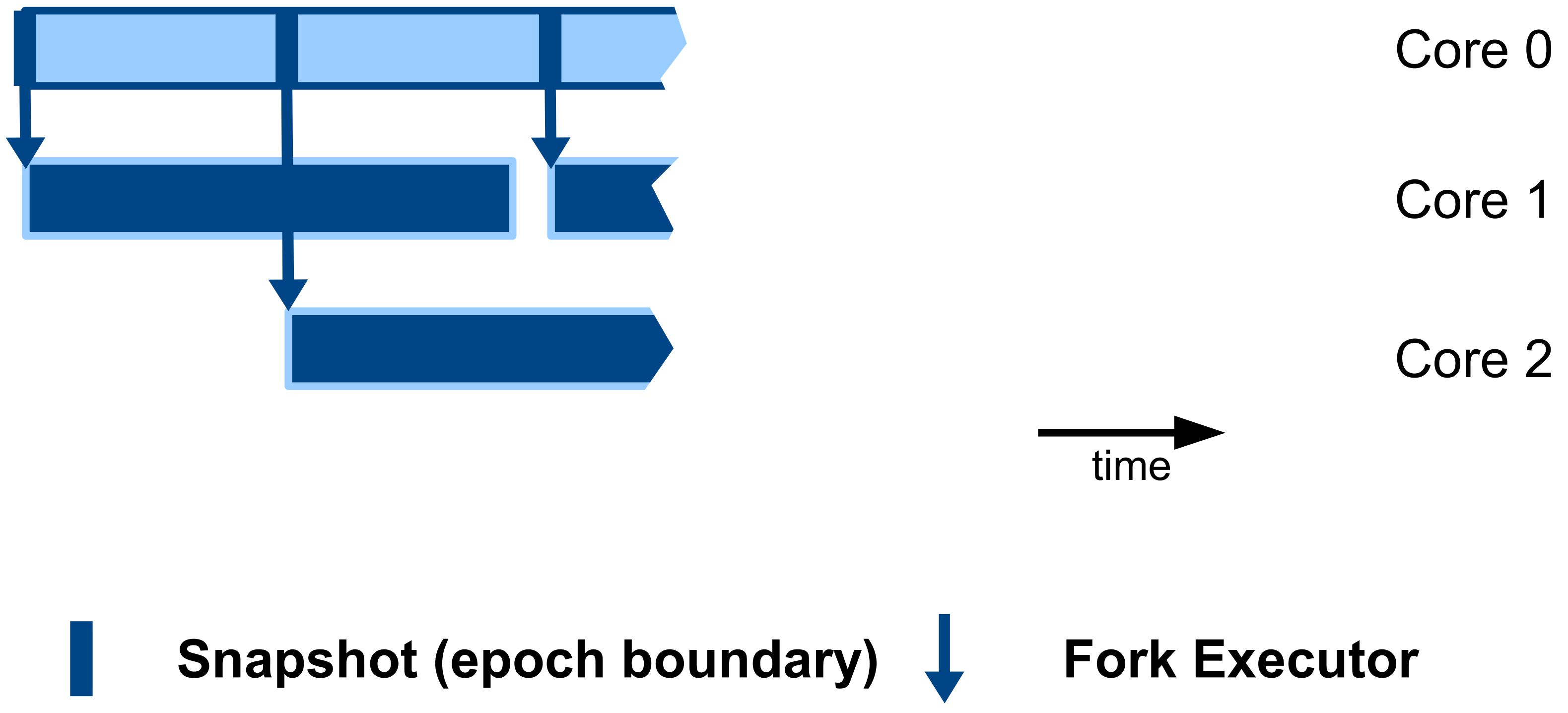
Snapshot (epoch boundary)

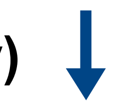
Fork Executor

Predictor – Fast Variant

Executor – Slow Variant

Predictor/Executor: Details

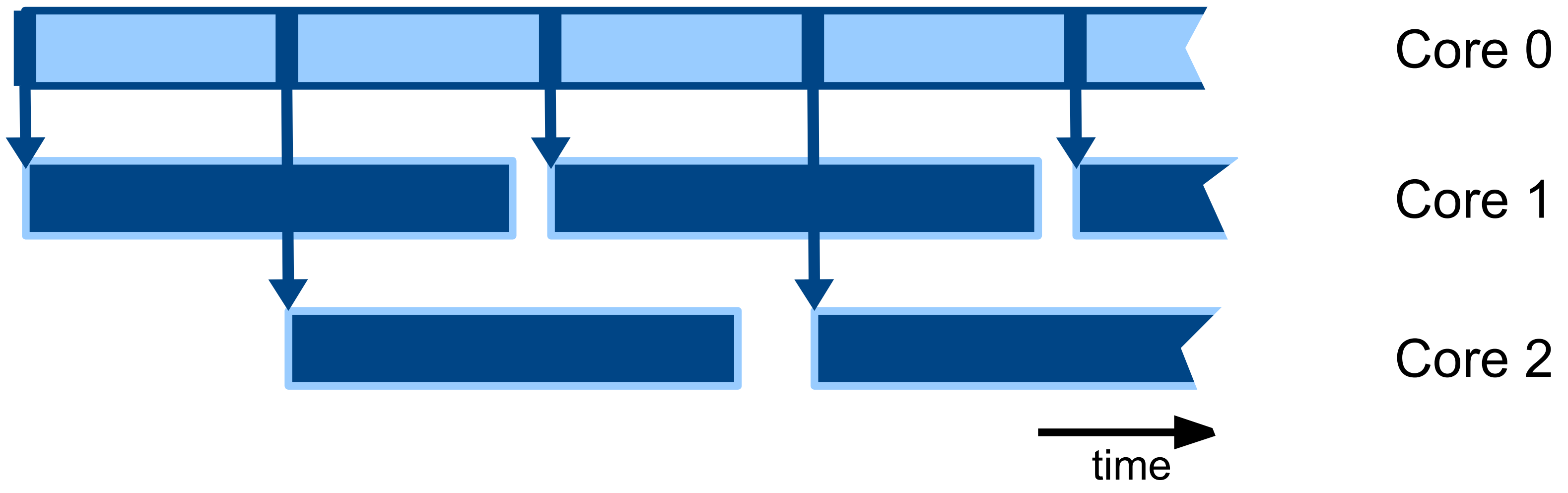


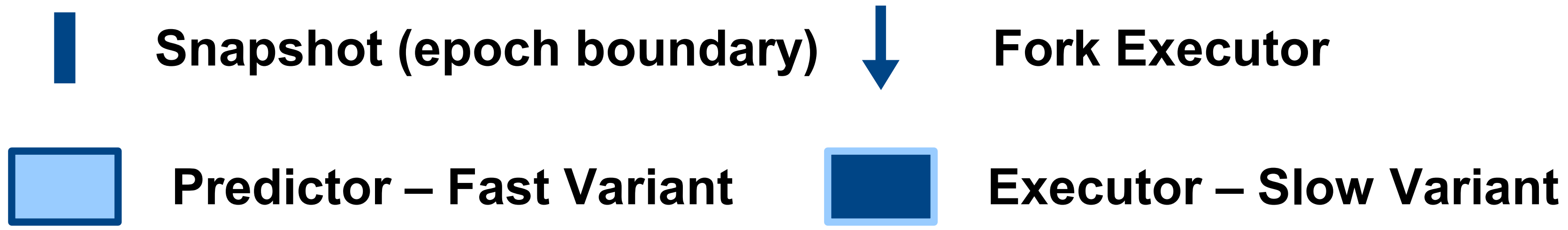


▮ **Snapshot (epoch boundary)** ↓ **Fork Executor**

Predictor – Fast Variant **Executor – Slow Variant**

Predictor/Executor: Details





Contributions

On-stack-replacement

StackLifter: application wide instrumentation at compile time

Contributions

On-stack-replacement

StackLifter: application wide instrumentation at compile time

Speculative variables

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

Contributions

On-stack-replacement

StackLifter: application wide instrumentation at compile time

Speculative variables

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

Speculative system calls

- Similar to Speck [2]
- But more modular

Contributions

On-stack-replacement

StackLifter: application
instrumentation at
time

Focus

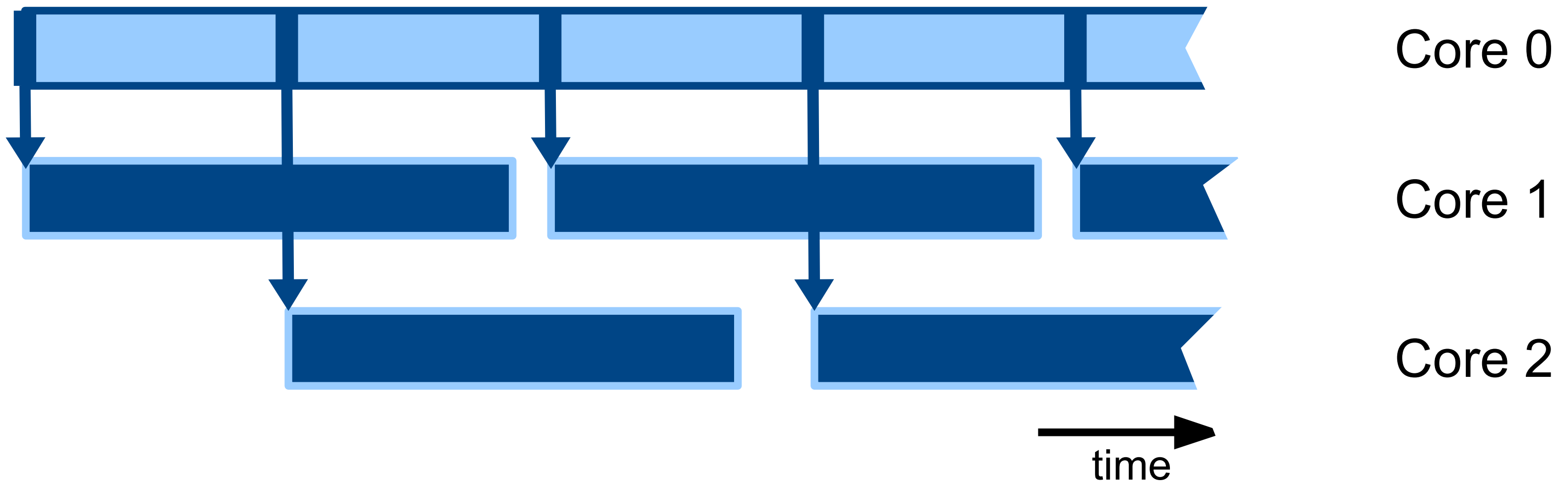
Speculative variables

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

Speculative system calls

- Similar to Speck [2]
- But more modular

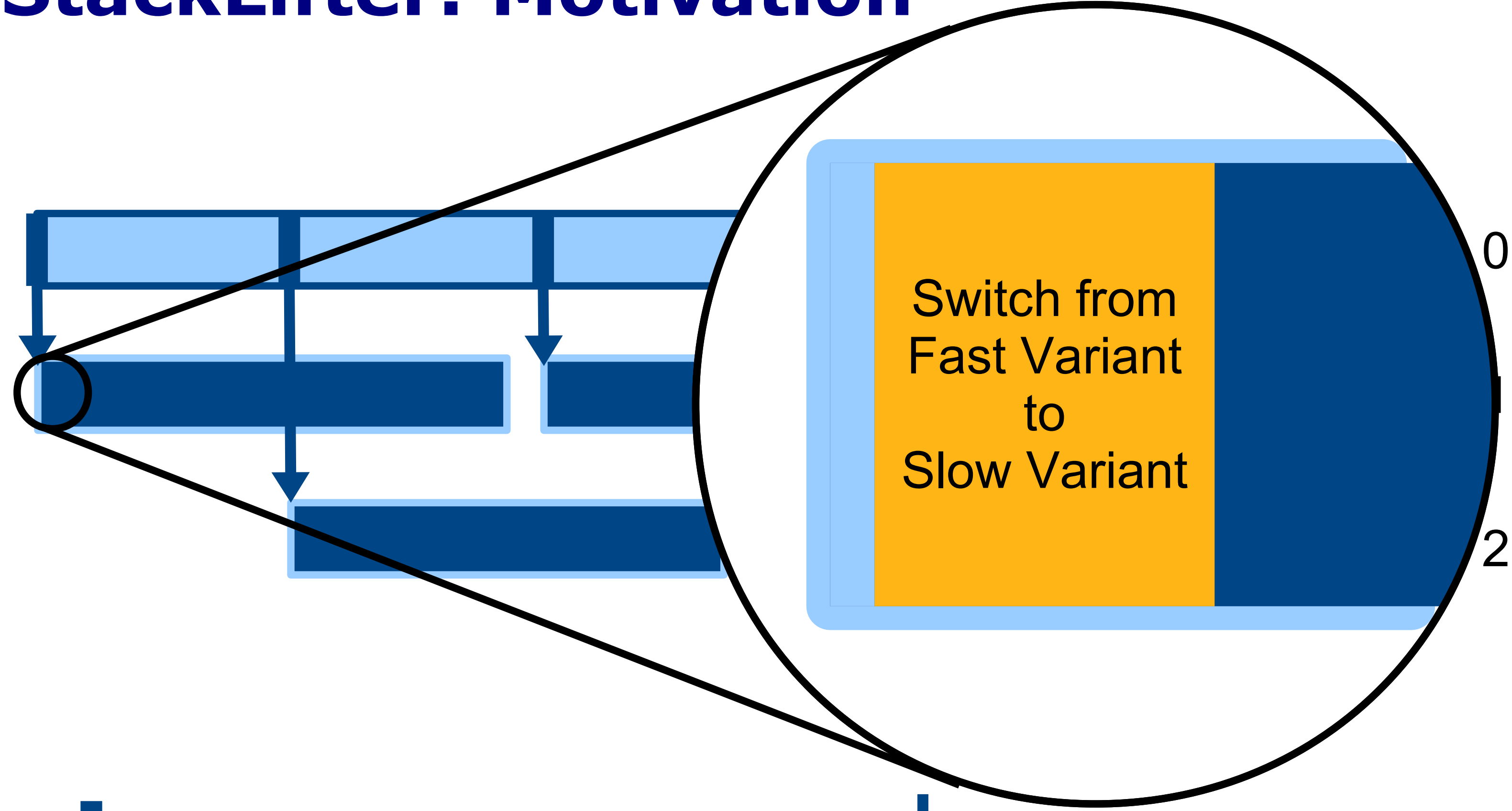
StackLifter: Motivation



▮ **Snapshot(epoch boundary)** ↓ **Fork executor**

▢ **Predictor – Fast Variant** ■ **Executor – Slow Variant**

StackLifter: Motivation



▮ **Snapshot(epoch boundary)** ↓ **Fork executor**

🟦 **Predictor – Fast Variant** ■ **Executor – 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

```
main () {
  foo ();
}

foo () {
  bar ();
}

bar () {
  chkpnt ();
}
```

StackLifter: Interface

```
main () {
  foo ();
}

foo () {
  bar ();
}

bar () {
  chkpnt ();
}
```

Switch epochs

StackLifter: Difficulty

Fast Variant

```
bar (int b) {
  int a = 2;
  chkpnt ();
}
```

StackLifter: Difficulty

Fast Variant

```
bar (int b) {
   int a = 2;
   chkpnt ();
}
```

Slow Variant

```
bar (int b, int c) {
   int check = ...
   int a = 2;
   chkpnt ();
}
```

StackLifter: Difficulty

Fast Variant

```
bar (int b) {
    int a = 2;
    chkpnt ();
}
```

Slow Variant

```
bar (int b, int c) {
    int check = ...
    int a = 2;
    chkpnt ();
}
```

StackLifter: Difficulty

Fast Variant

```
bar (int b) {
    int a = 2;
    chkpnt ();
}
```

Slow Variant

```
bar (int b, int c) {
    int check = ...
    int a = 2;
    chkpnt ();
}
```

Translate:

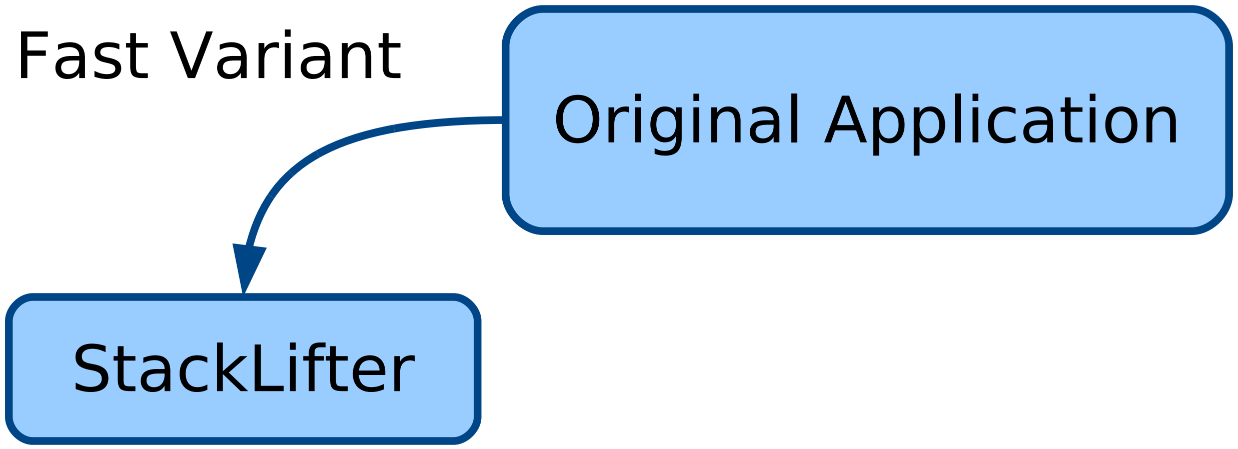
- Local variables + arguments
- Instruction pointer
- Return addresses on the stack

StackLifter: Instrumentation Approach

Original Application

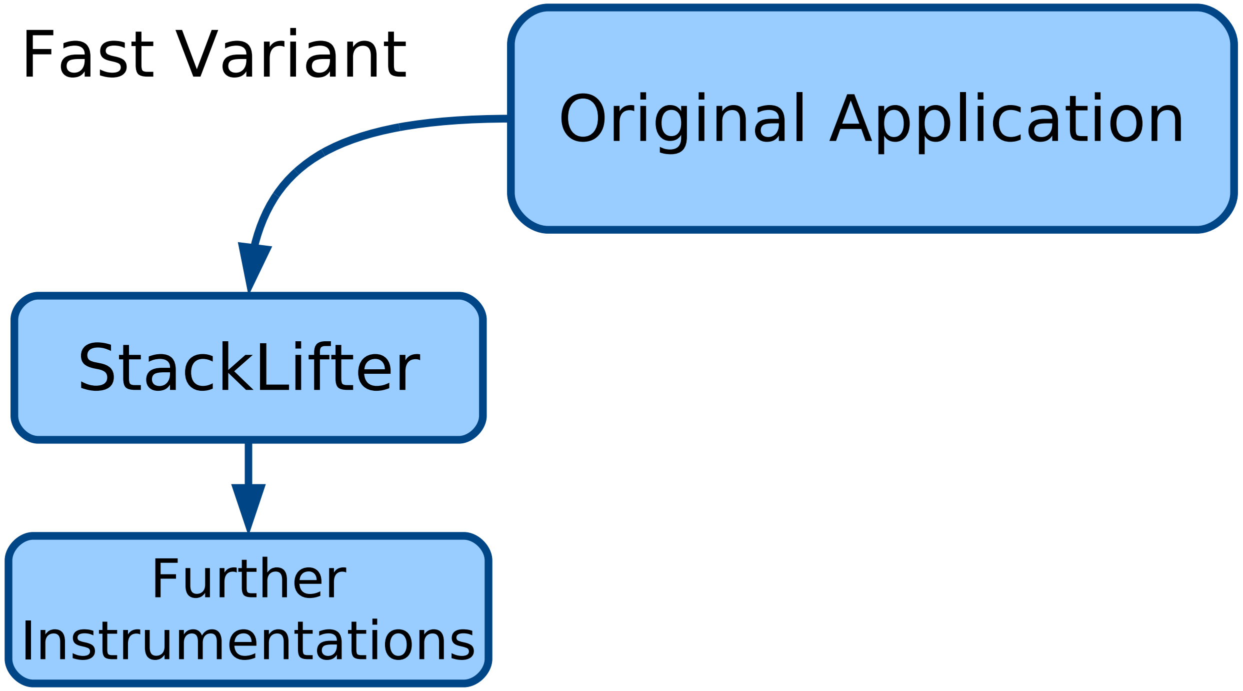
StackLifter: Instrumentation Approach

Fast Variant

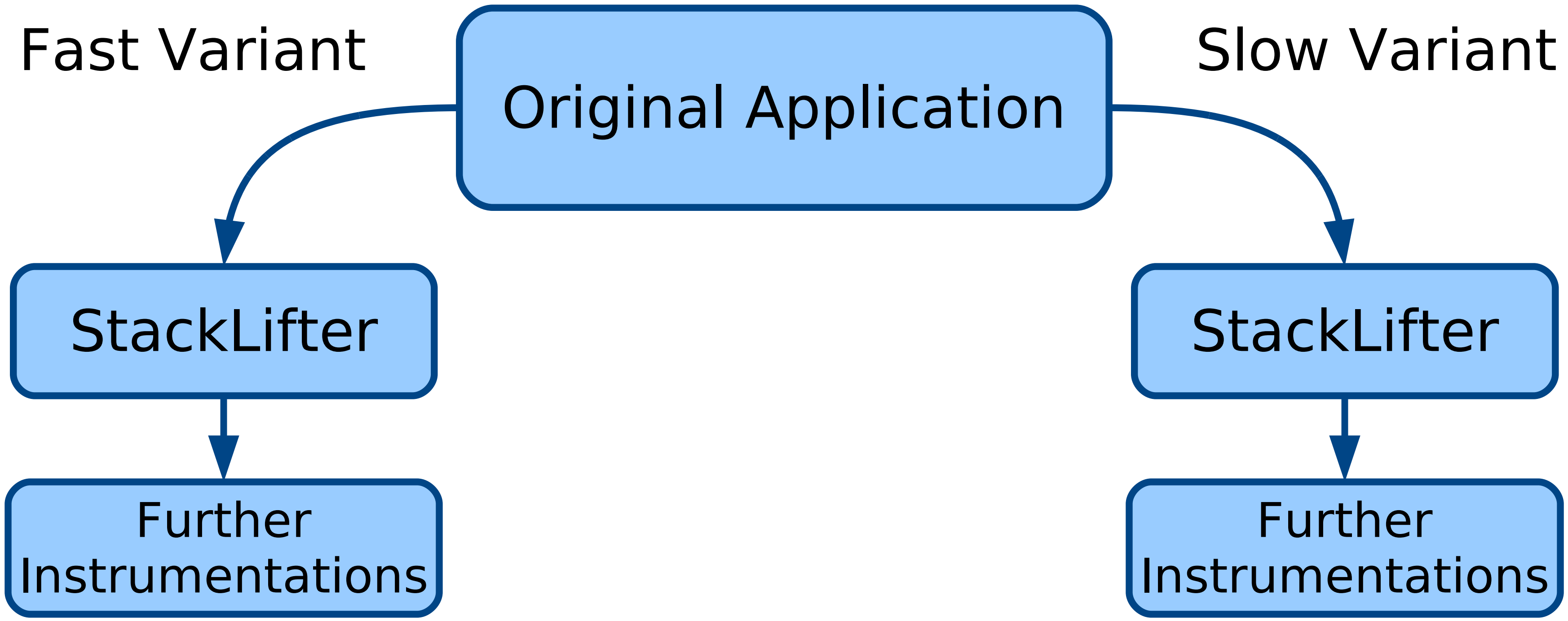


StackLifter: Instrumentation Approach

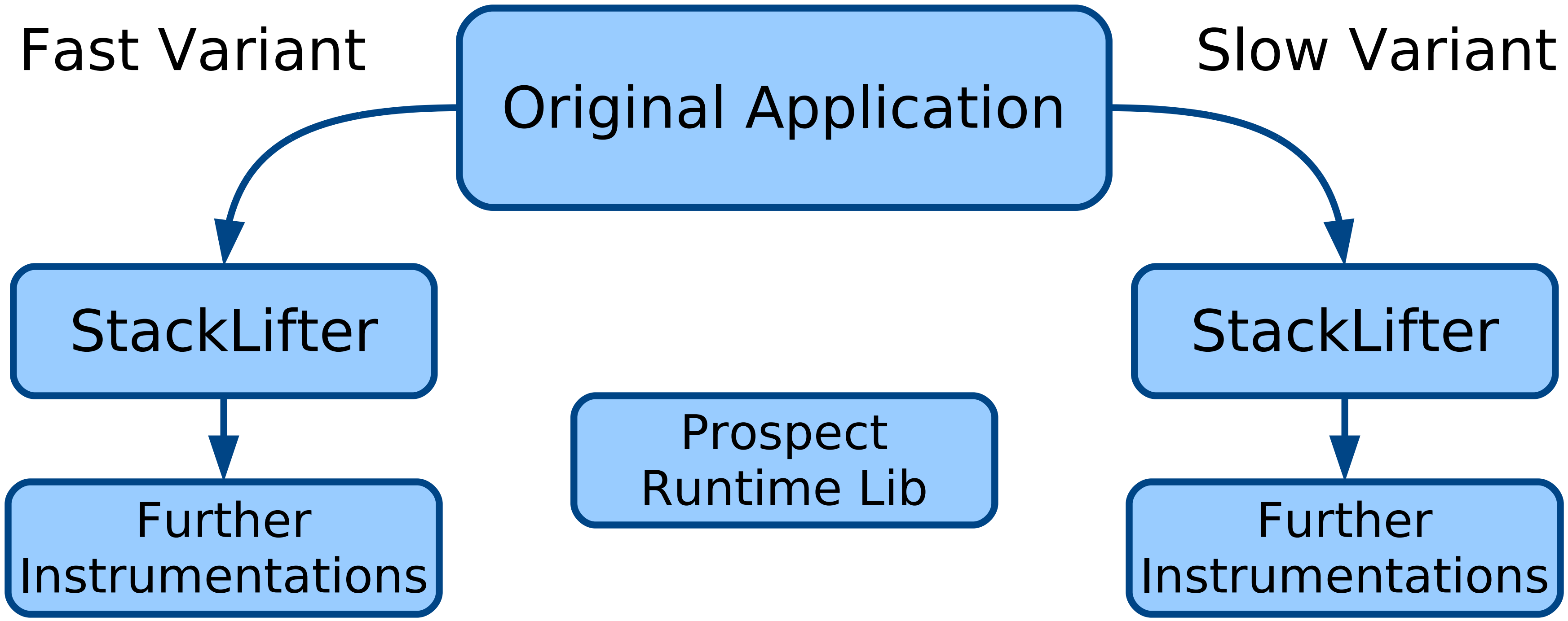
Fast Variant



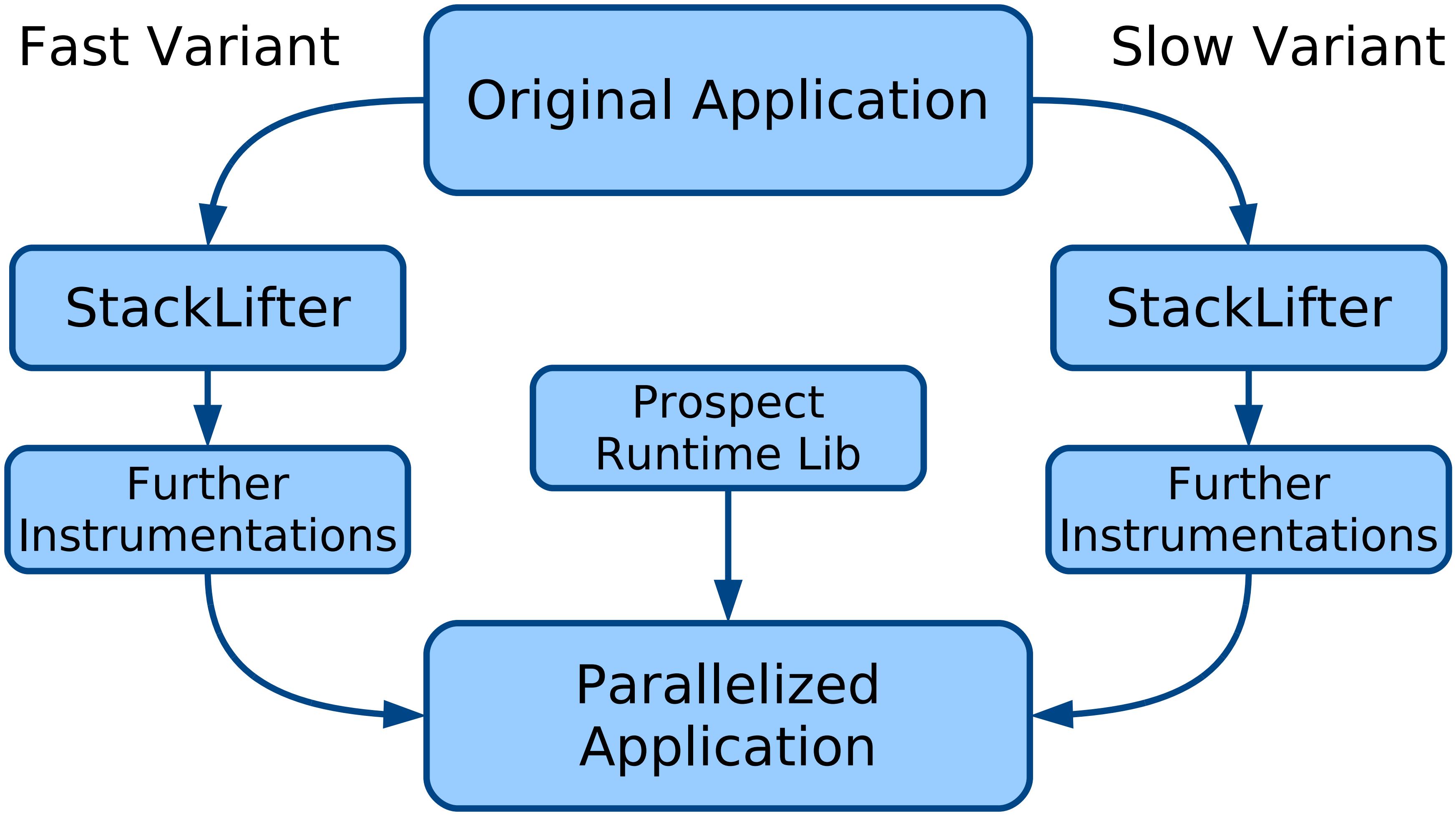
StackLifter: Instrumentation Approach



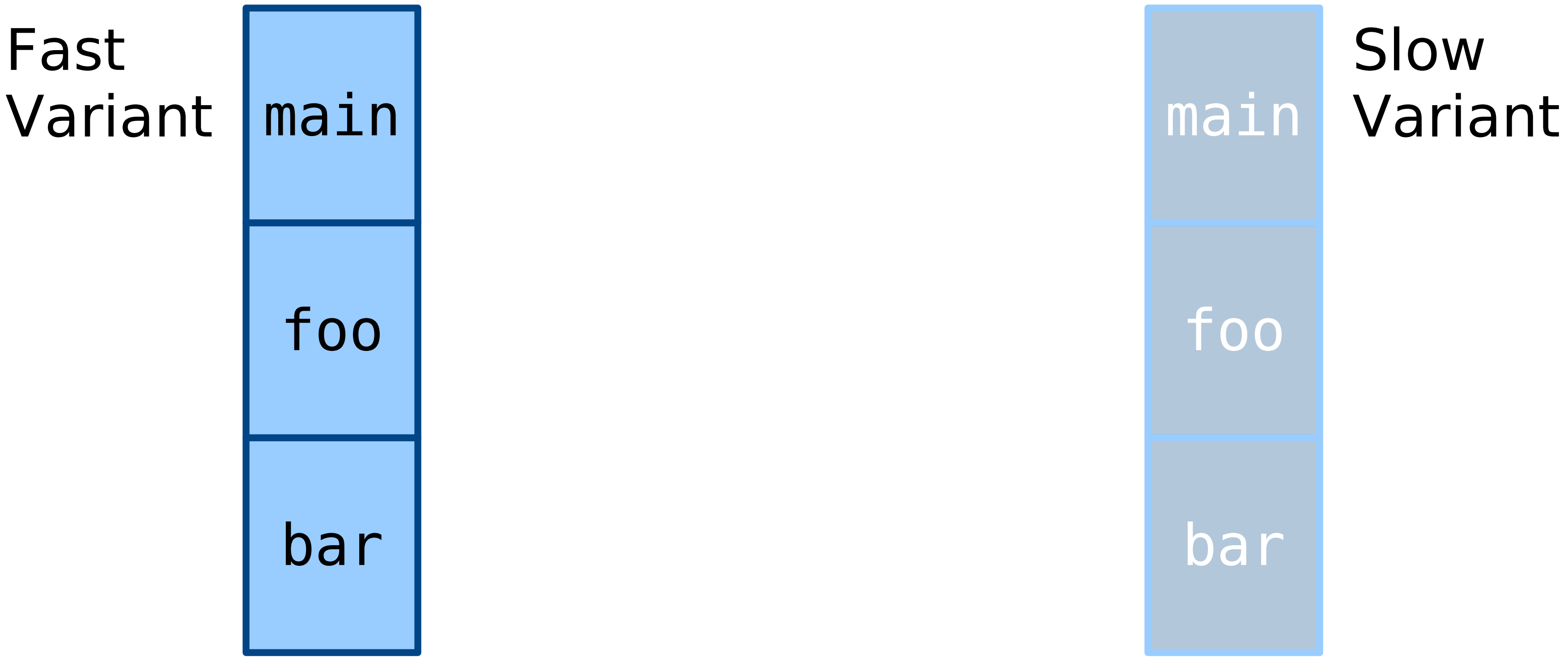
StackLifter: Instrumentation Approach



StackLifter: Instrumentation Approach

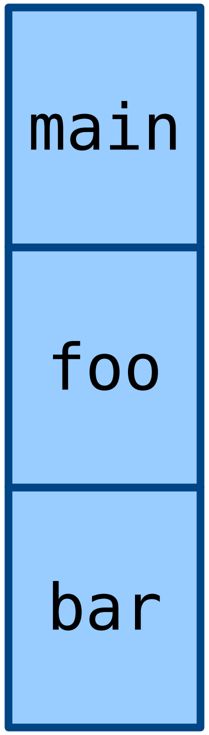


StackLifter: On-Stack Replacement



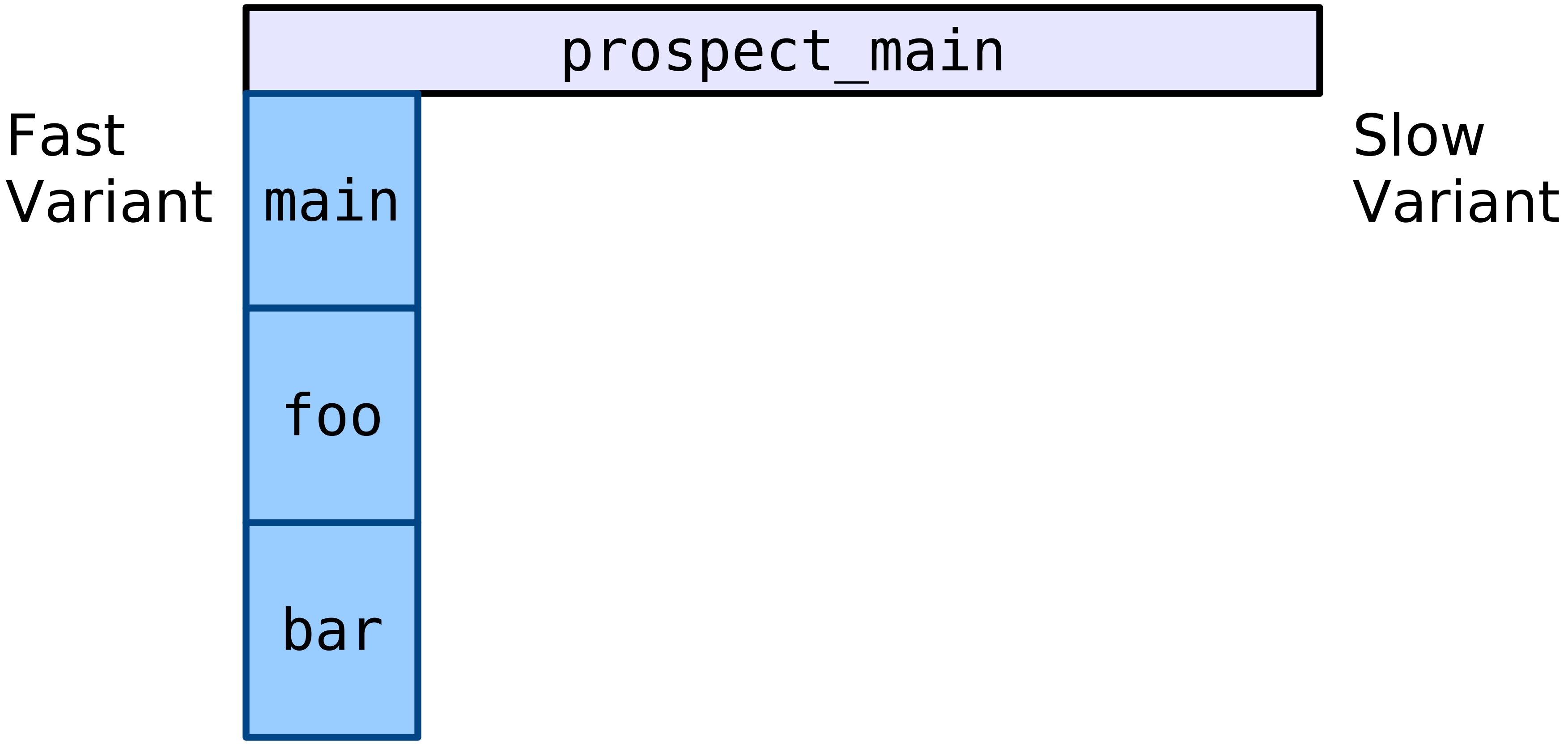
StackLifter: On-Stack Replacement

Fast Variant



Slow Variant

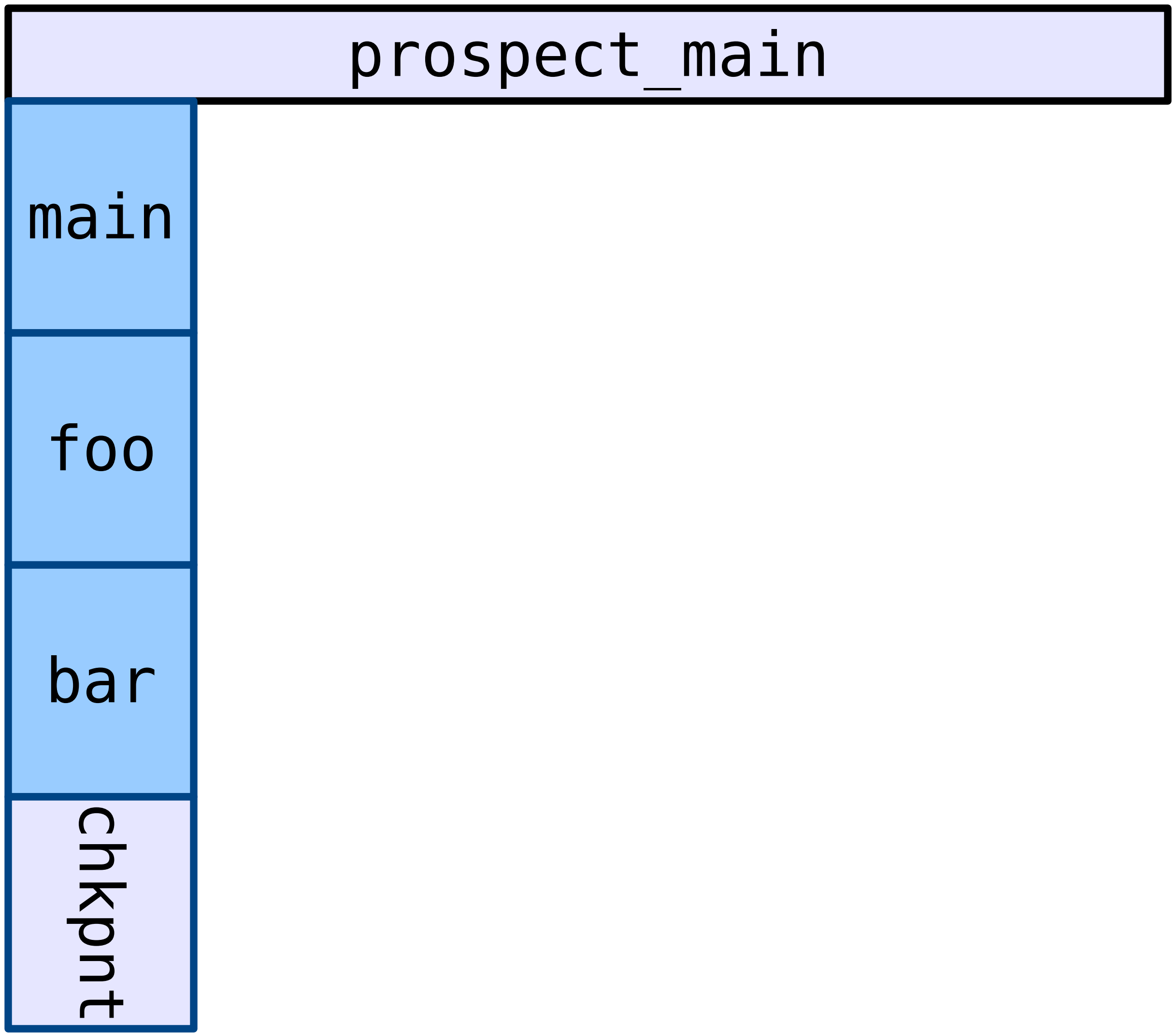
StackLifter: On-Stack Replacement



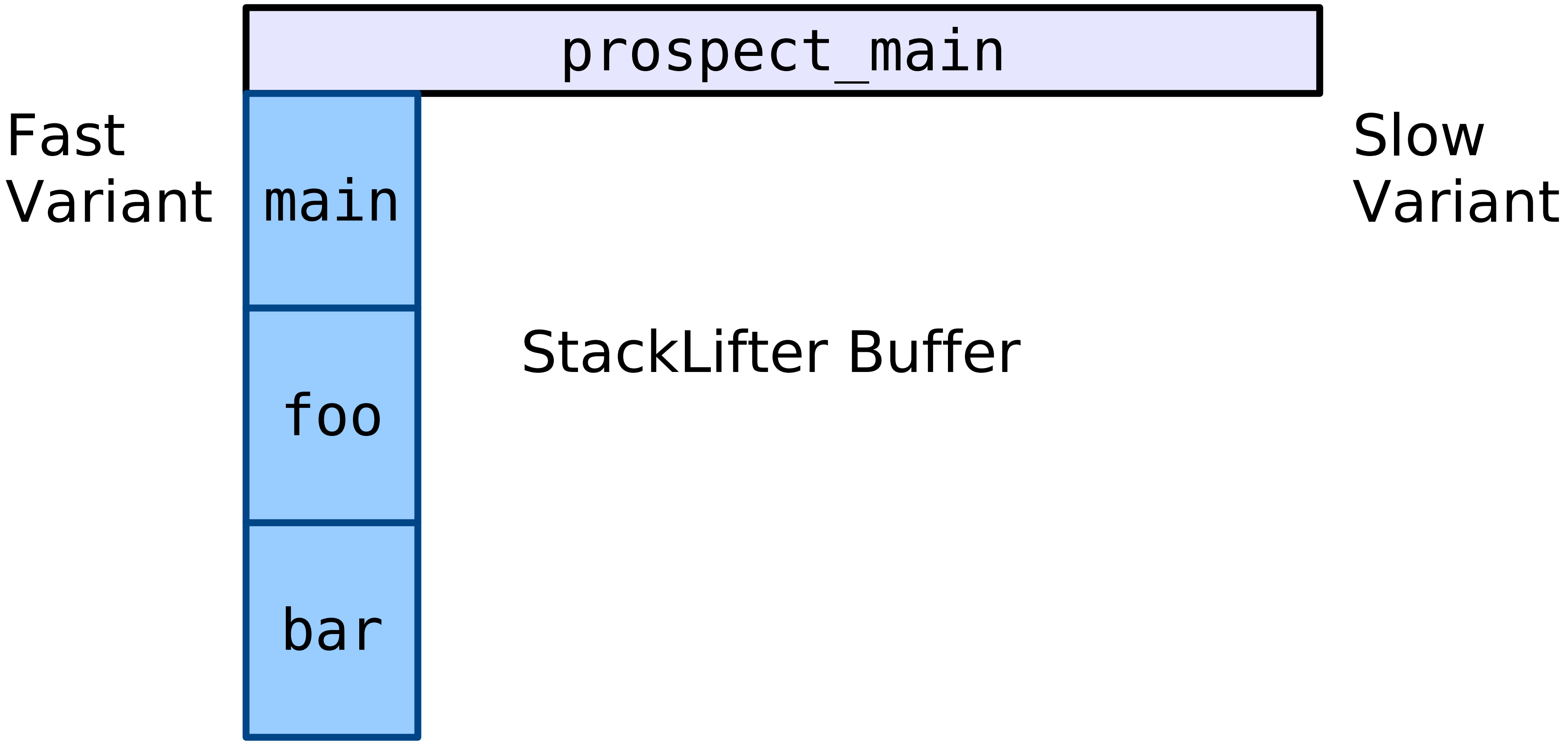
StackLifter: On-Stack Replacement

Fast
Variant

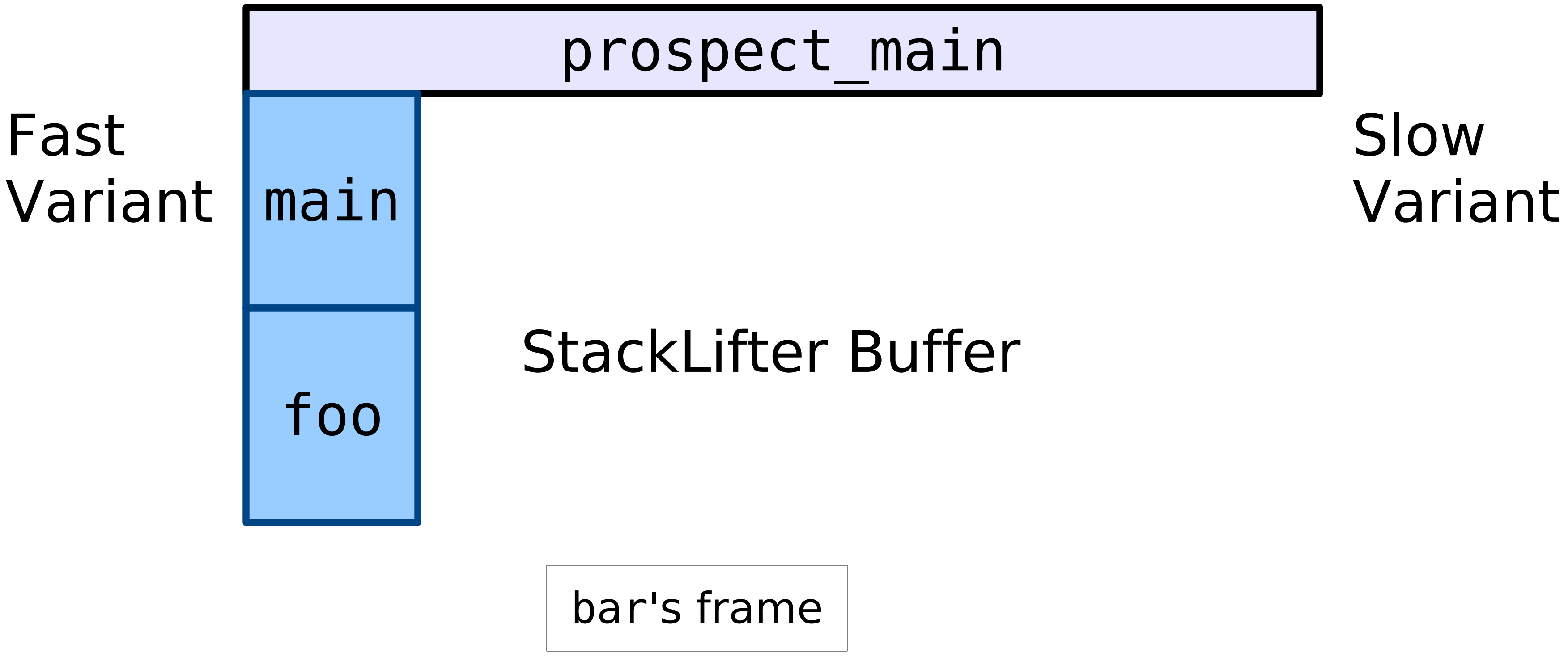
Slow
Variant



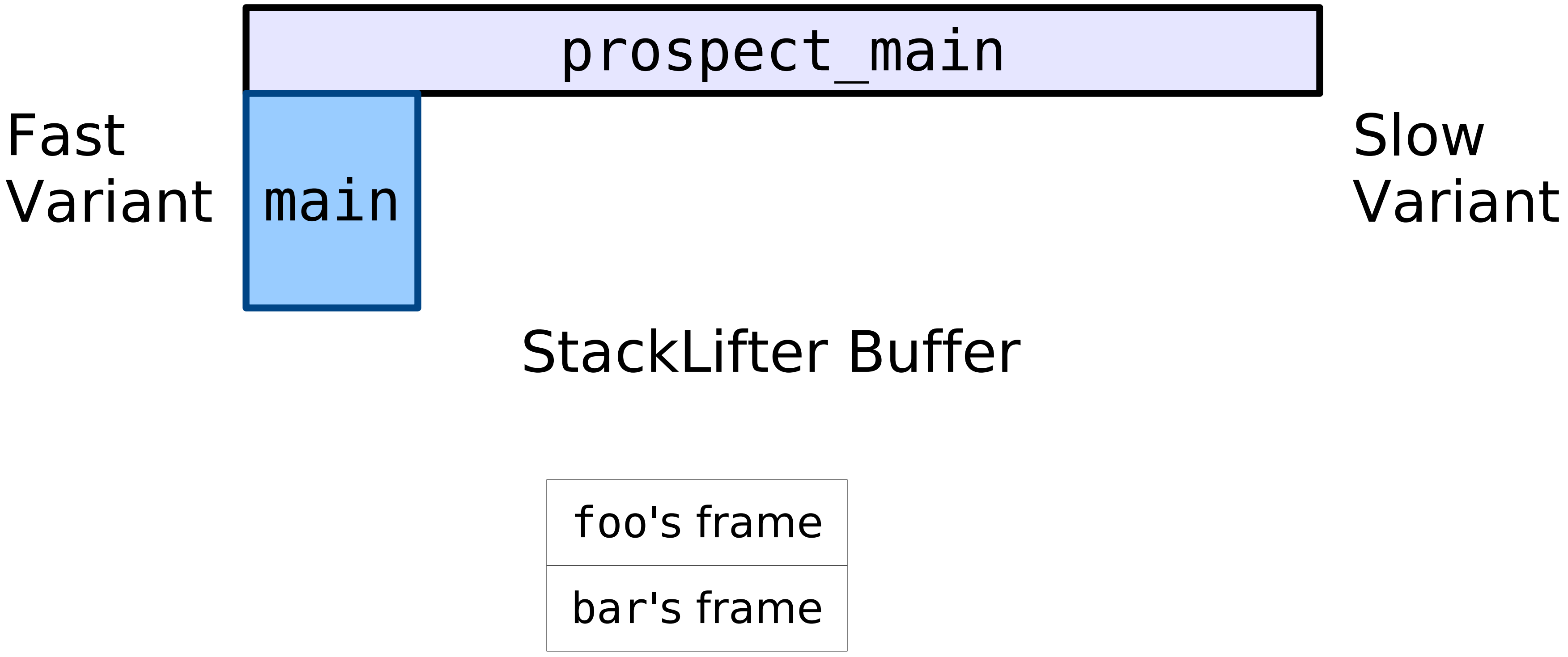
StackLifter: On-Stack Replacement



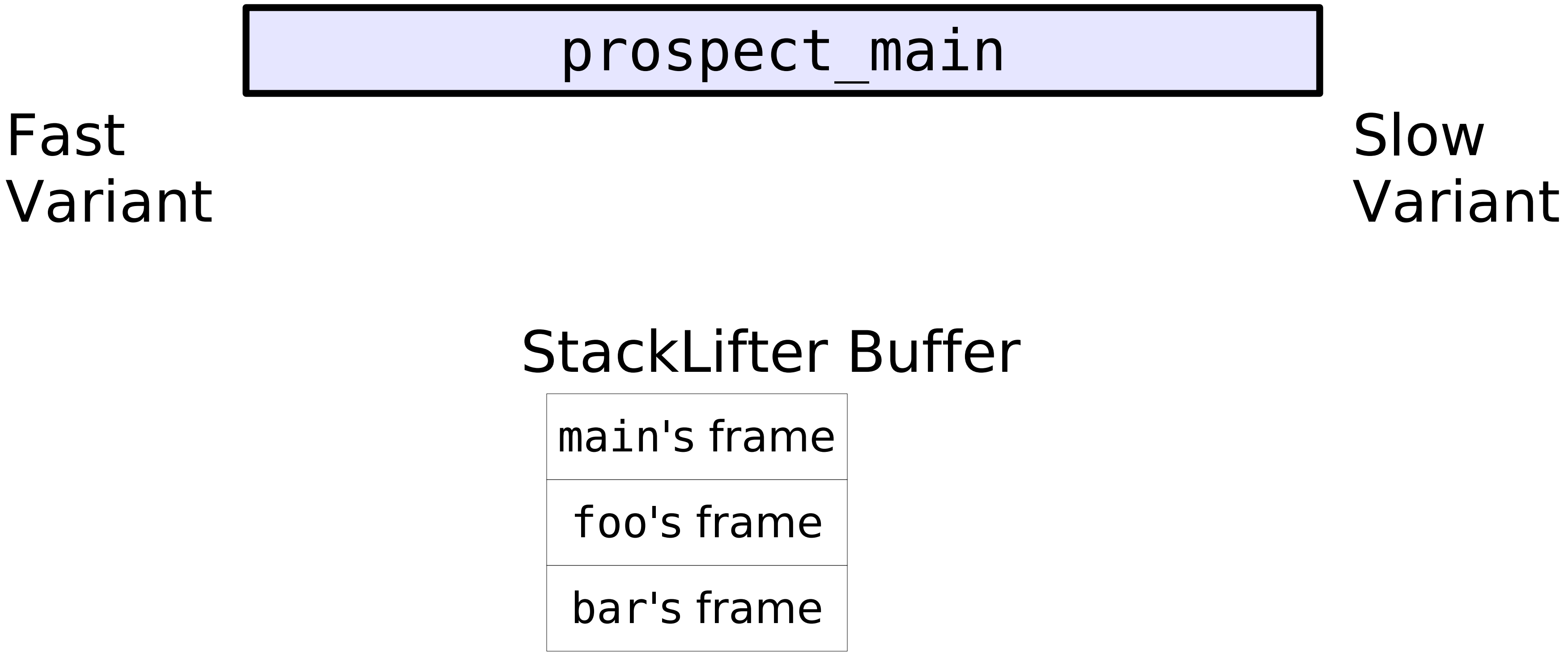
StackLifter: On-Stack Replacement



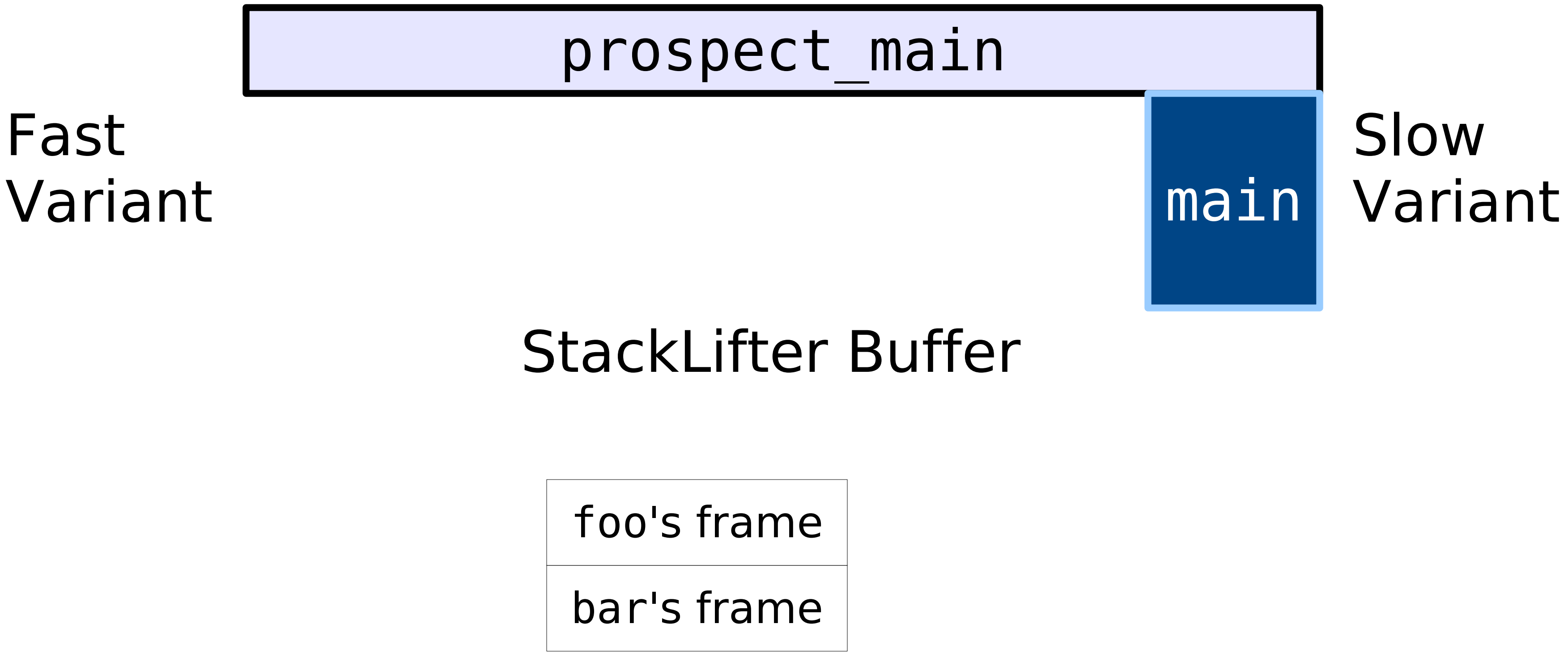
StackLifter: On-Stack Replacement



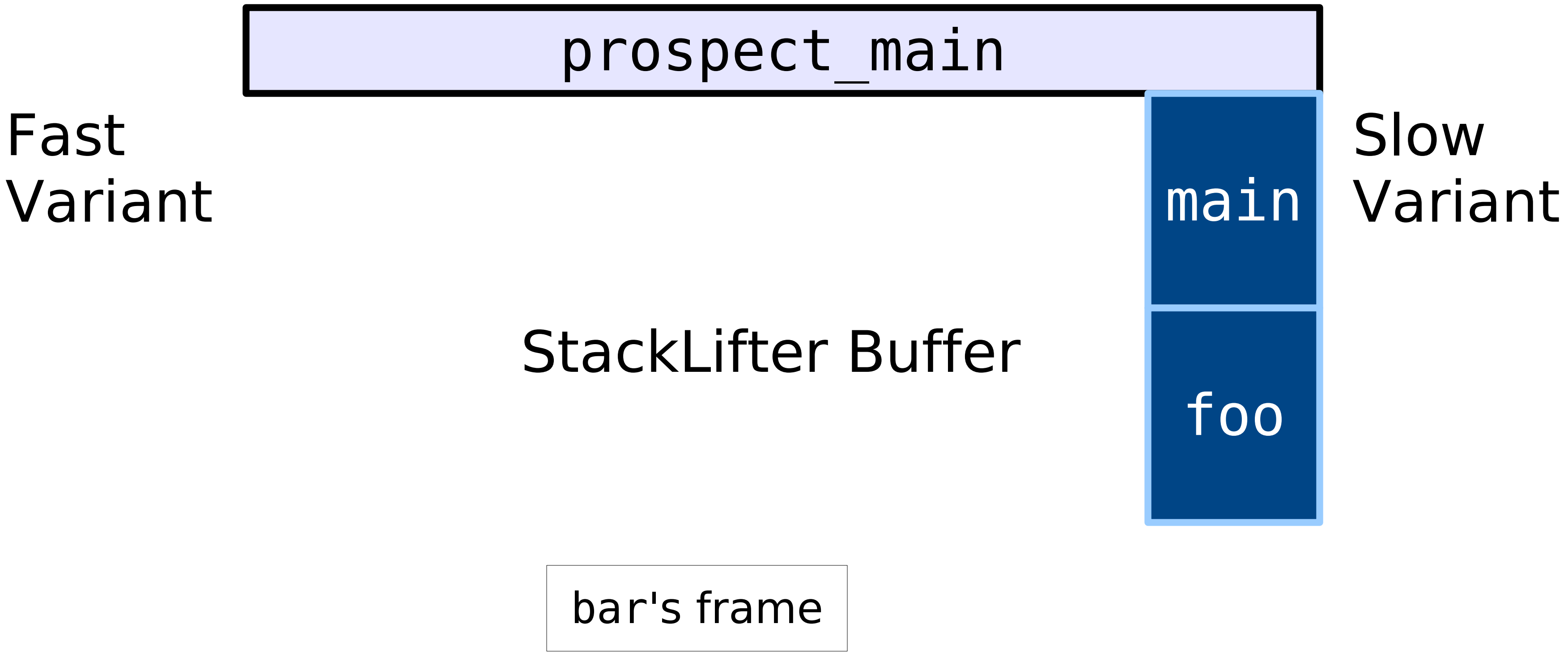
StackLifter: On-Stack Replacement



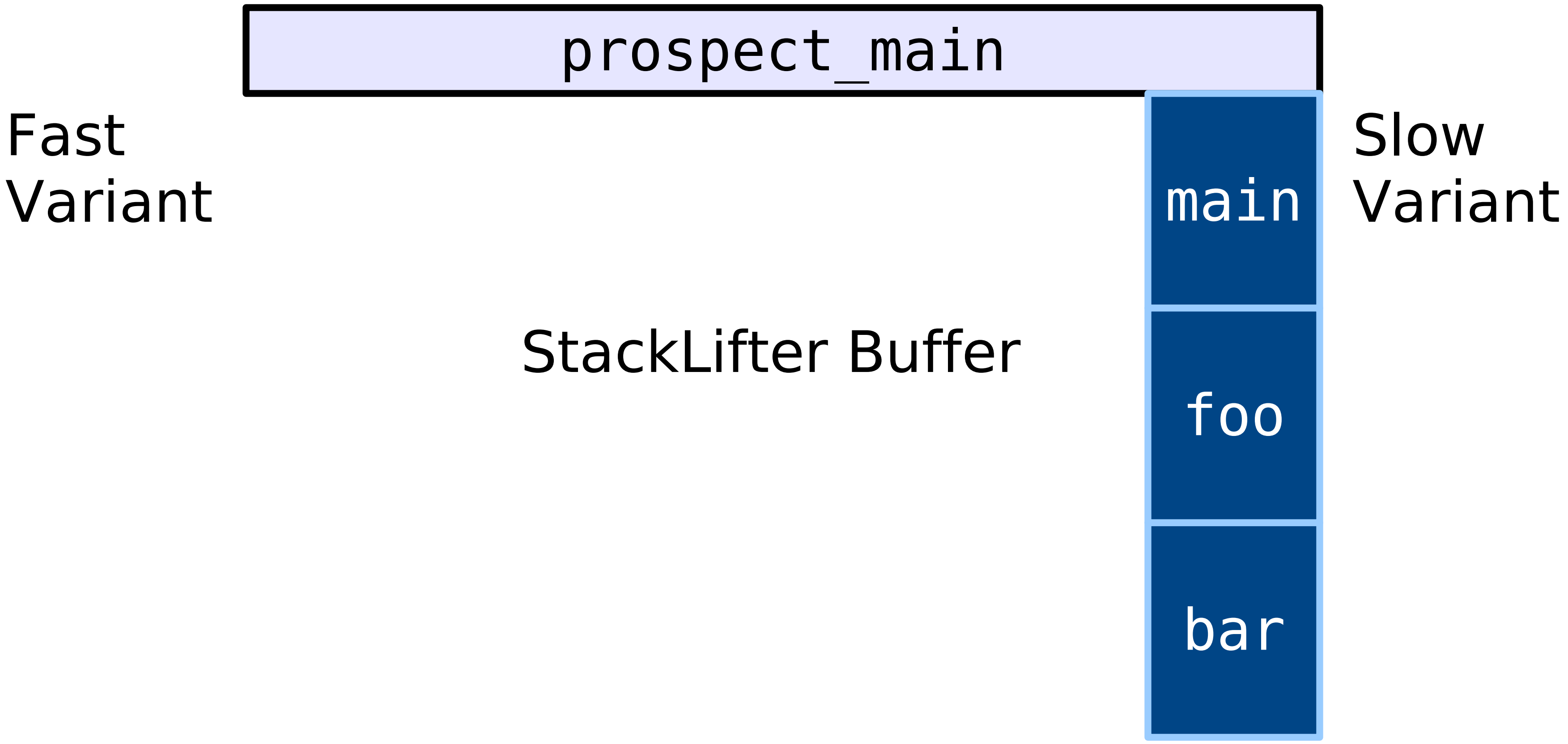
StackLifter: On-Stack Replacement



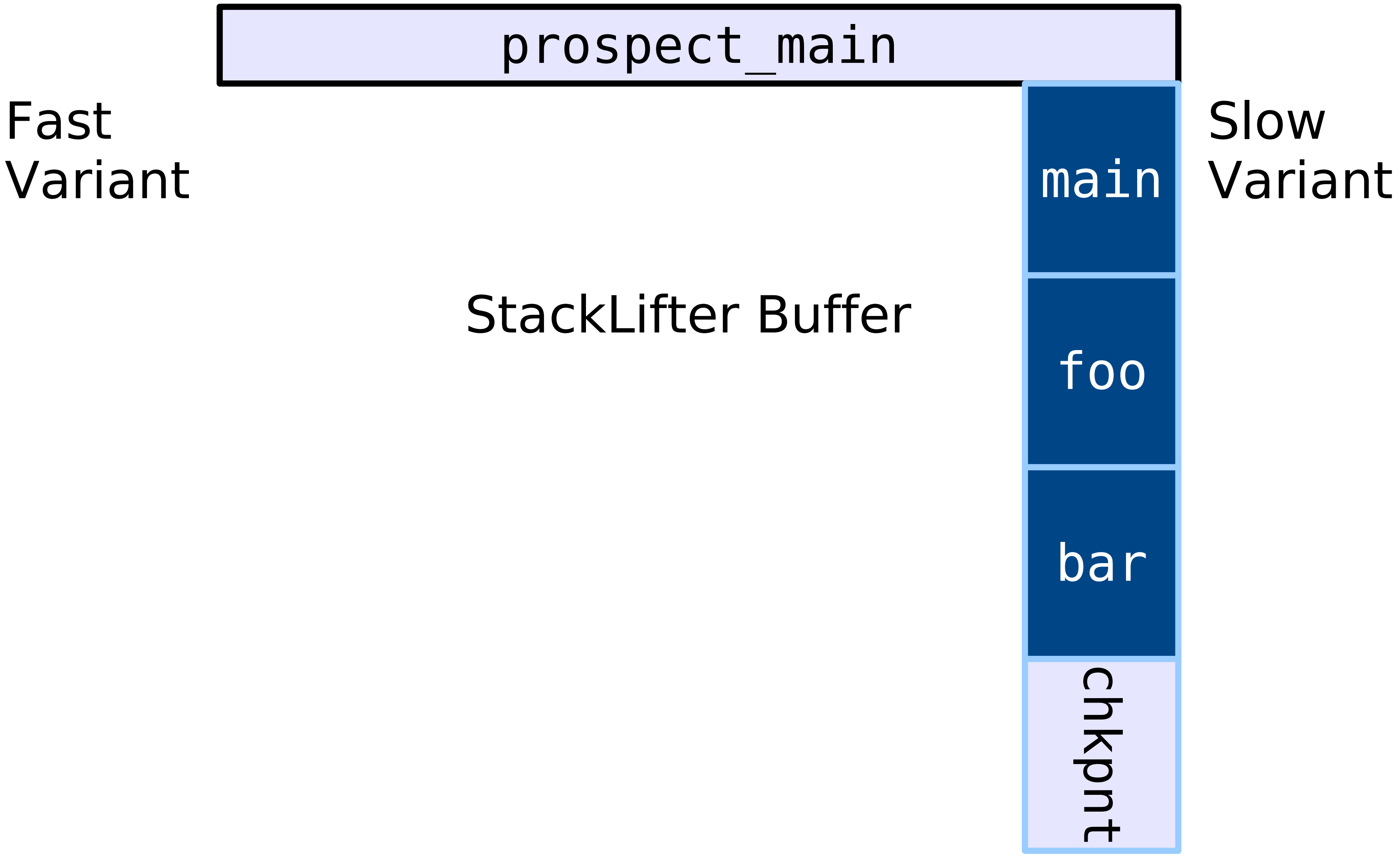
StackLifter: On-Stack Replacement



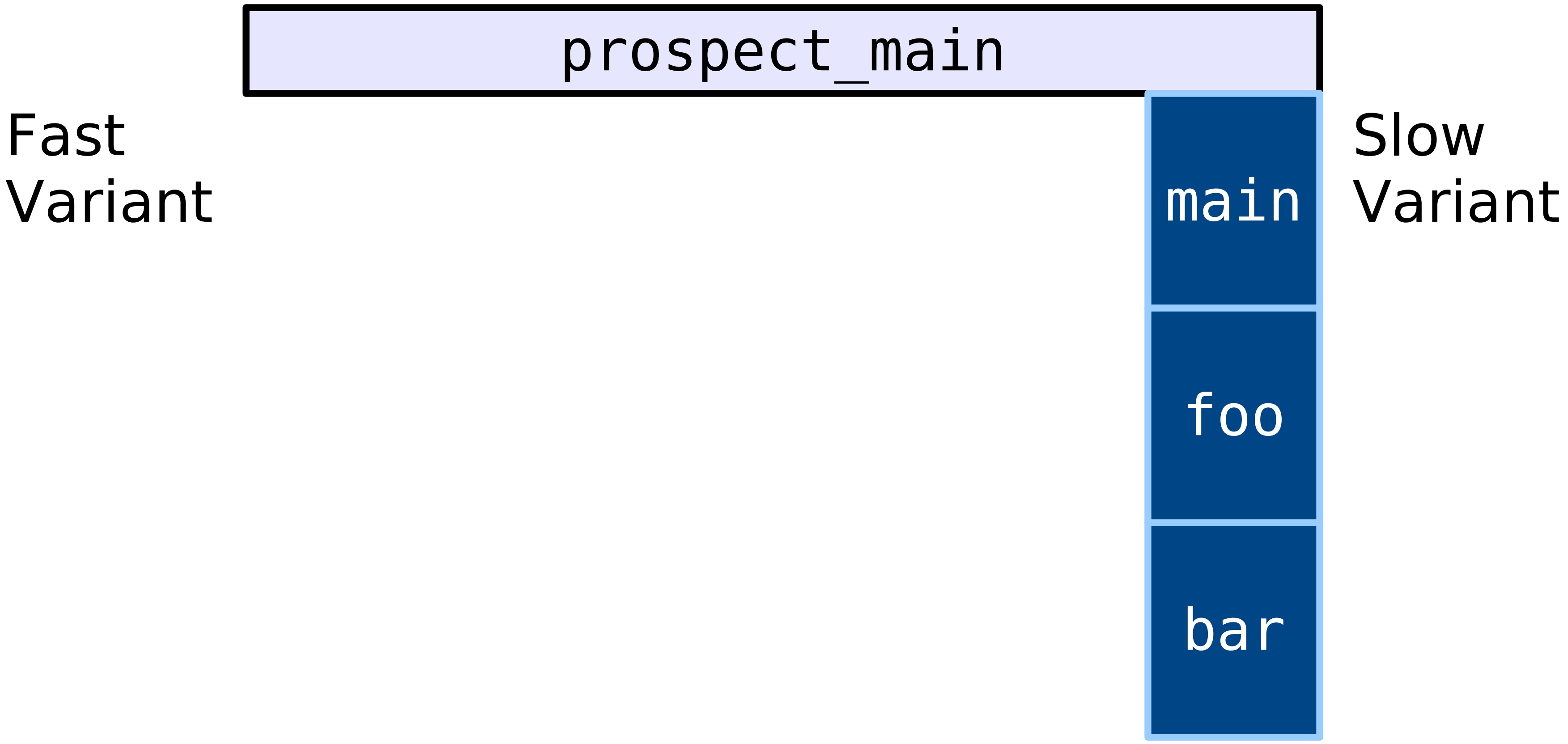
StackLifter: On-Stack Replacement



StackLifter: On-Stack Replacement

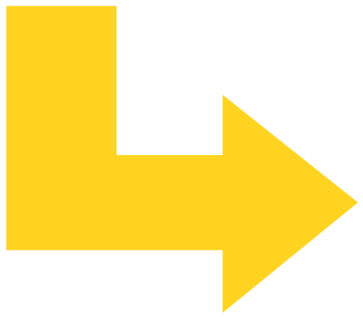


StackLifter: On-Stack Replacement



StackLifter: Instrumentation of Fast Variant

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

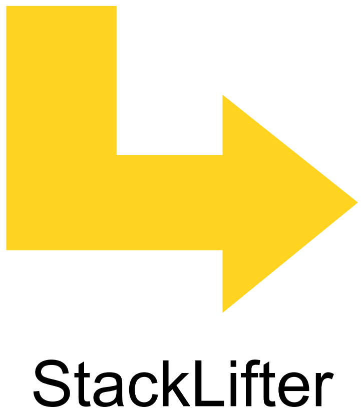


StackLifter

```
1:  bar () {
2:    int a = 2;
3:
4:    chkpnt ();
5:
6:
7:
8:
9:
10:
11: }
```

StackLifter: Instrumentation of Fast Variant

```
bar () {
  int a = 2;
  chkpnt ();
}
```



```
bar () {
  int a = 2;
call_01:
  chkpnt ();






}
```

StackLifter: Instrumentation of Fast Variant

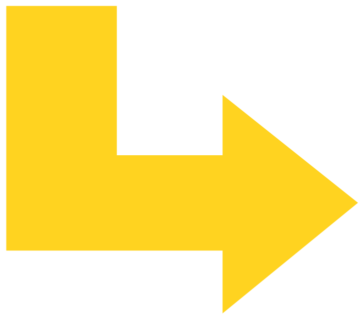
```
bar () {
  int a = 2;
  chkpnt ();
}
```

StackLifter

```
bar () {
  int a = 2;
call_01:
  chkpnt ();
  if (stackLifting)
  {
    push (a);
    push (call_01);
    return;
  }
}
```

StackLifter: Instrumentation of Slow Variant

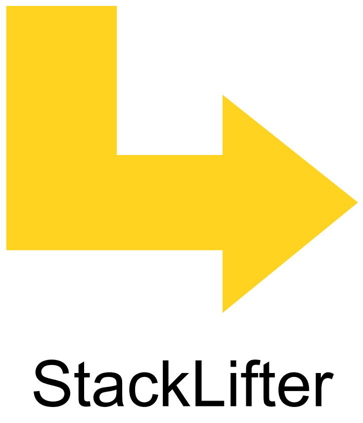
```
bar () {
  int a = 2;
  chkpnt ();
}
```



```
bar () {











  int a = 2;

  chkpnt ();
}
```

StackLifter: Instrumentation of Slow Variant

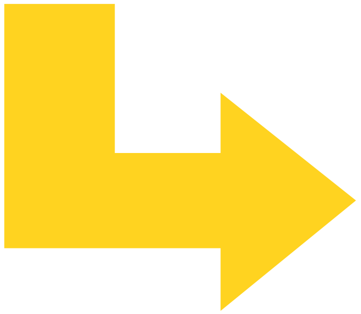
```
bar () {
  int a = 2;
  chkpnt ();
}
```



```
bar () {










entry:
  int a = 2;
call_01:
  chkpnt ();
}
```

StackLifter: Instrumentation of Slow Variant

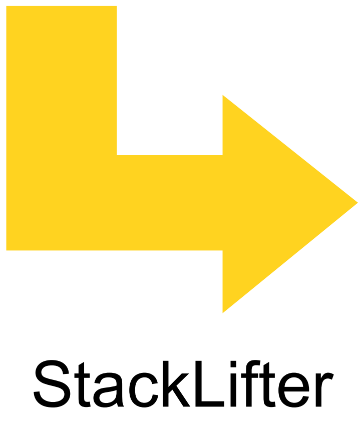
```
bar () {
  int a = 2;
  chkpnt ();
}
```



```
bar () {







restore_01:
  a = pop ();
  goto call_01;
entry:
  int a = 2;
call_01:
  chkpnt ();
}
```

StackLifter: Instrumentation of Slow Variant

```
bar () {
    int a = 2;
    chkpnt ();
}
```



```
bar () {
  if (stackLifting) {
    switch (pop ()) {
    case call_01:
      goto restore_01;
    }
  }
  else goto entry;
restore_01:
  a = pop ();
  goto call_01;
entry:
  int a = 2;
call_01:
  chkpnt ();
}
```

StackLifter: Features

Transparent

Instrumentations do not need to be aware of StackLifter

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

Speedup

Of Prospect:
- Out-of-bounds checker
- FastAssert

Evaluation: Overview

Speedup

Of Prospect:
- Out-of-bounds checker
- FastAssert

Overhead

Of StackLifter

Evaluation: Overview

Speedup

Of Prospect:
- Out-of-bounds checker
- FastAssert

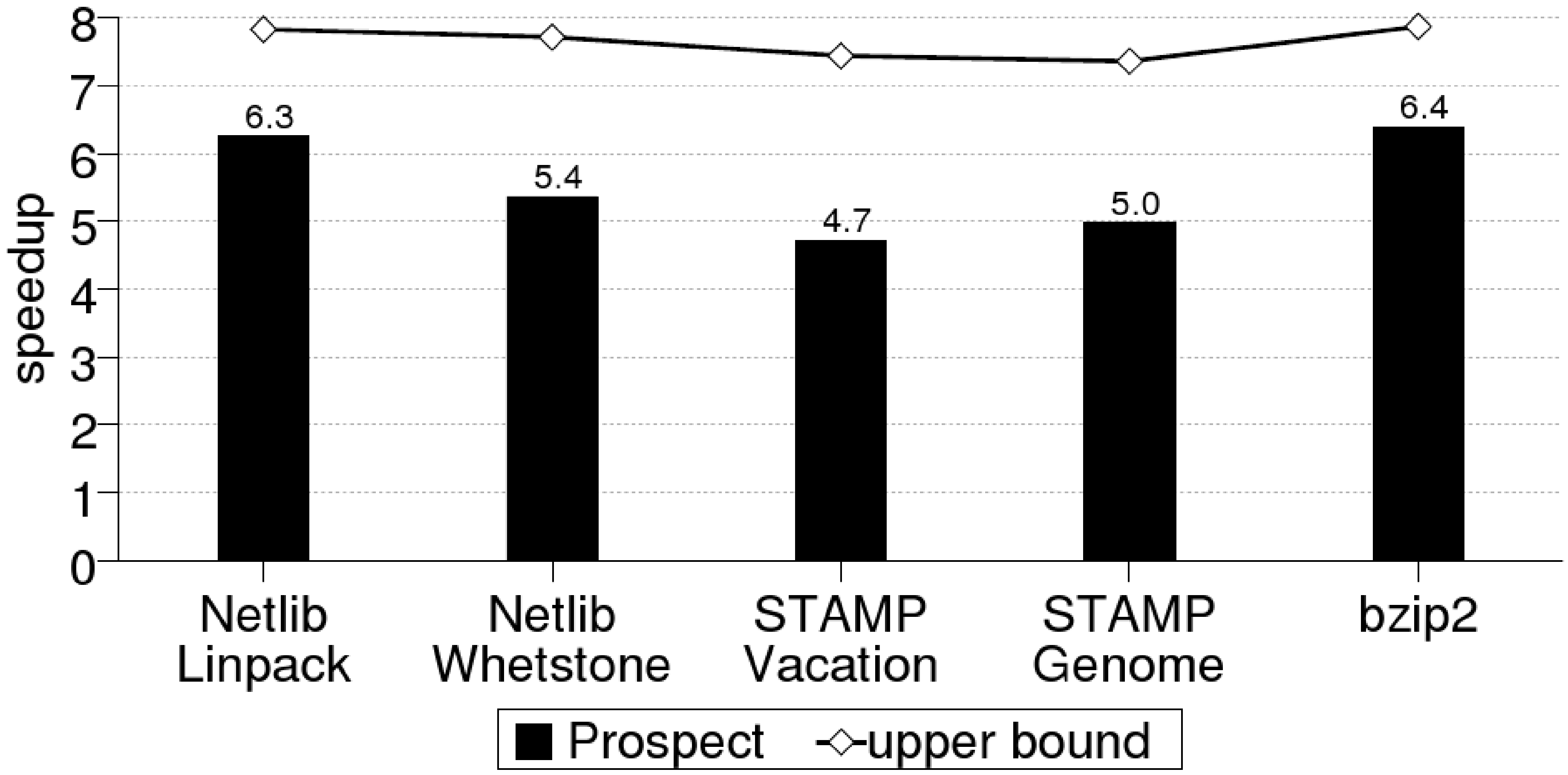
Overhead

Of StackLifter

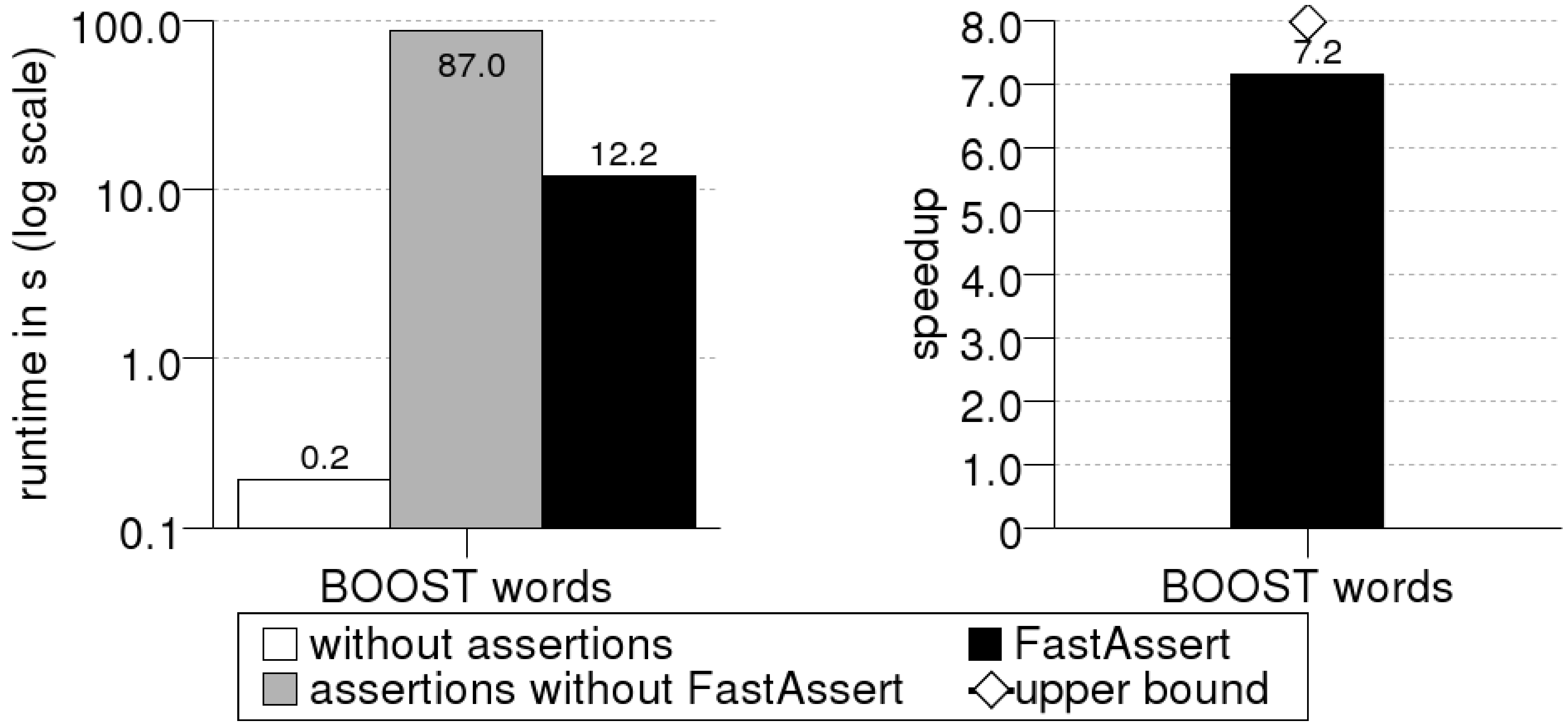
Setup

- Intel Xeon 8-core CPU
- 6 benchmark applications

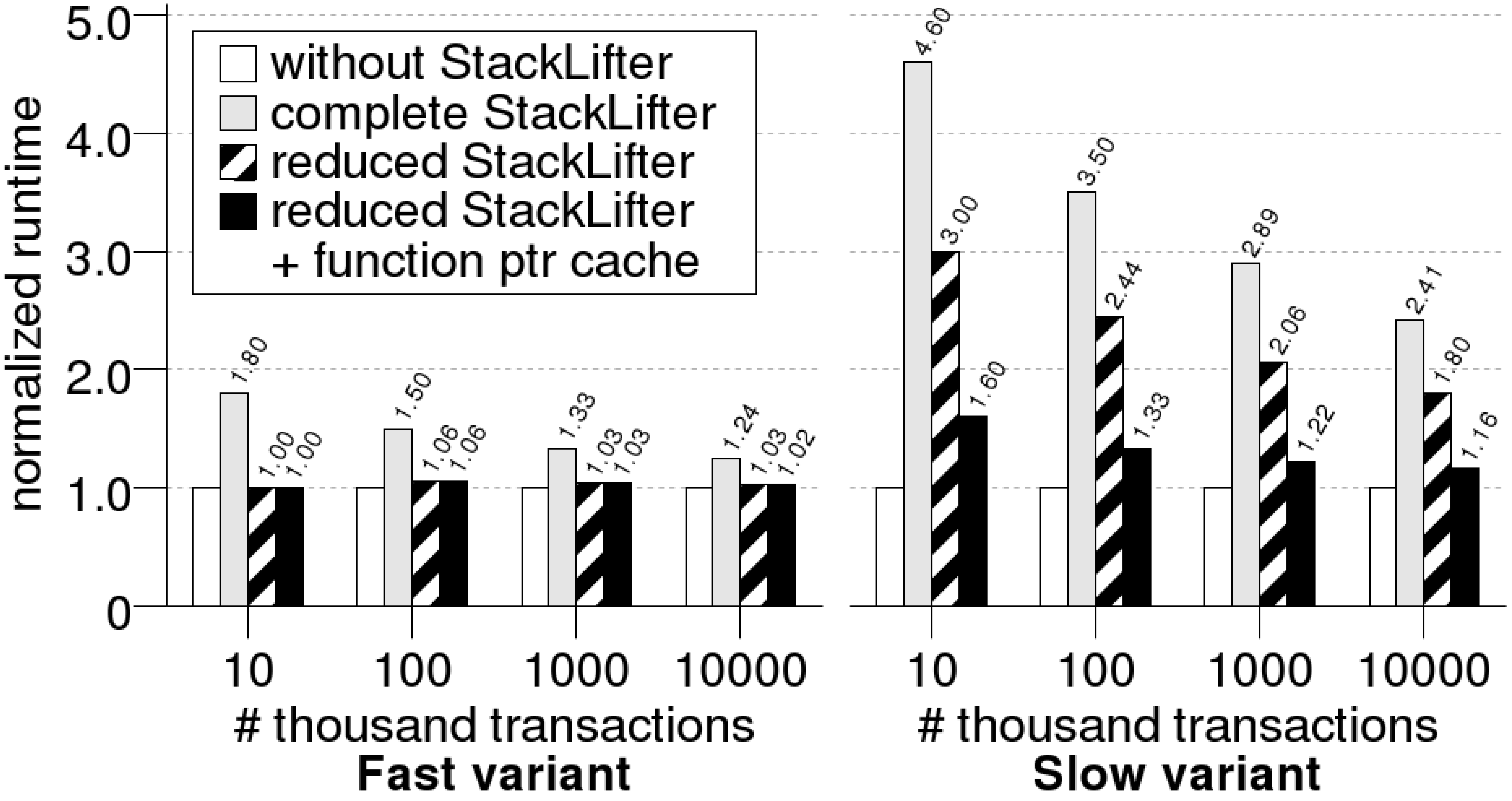
Evaluation: Out-of-bounds



Evaluation: FastAssert



Evaluation: StackLifter (Vacation benchmark)



Comparison

	Prospect
Instrumentation	compiler
Slow variant	**+**
Fast variant	**+**
Application Wide	yes
Syscall support	speculative
Developer interface	**`chkpnt`**
Additional state in Slow Variant	speculative

Comparison

	Prospect	**SuperPin** [1]
Instrumentation	compiler	DBI
Slow variant	**+**	**o**
Fast variant	**+**	**-**
Application Wide	yes	yes
Syscall support	speculative	replay only
Developer interface	**`chkpnt`**	transparent
Additional state in Slow Variant	speculative	non-speculative

Comparison

	Prospect	**SuperPin** [1]	**Speck** [2]
Instrumentation	compiler	DBI	DBI
Slow variant	**+**	**o**	**o**
Fast variant	**+**	**-**	**-**
Application Wide	yes	yes	yes
Syscall support	speculative	replay only	speculative
Developer interface	**`chkpnt`**	transparent	transparent
Additional state in Slow Variant	speculative	non-speculative	non-speculative

Comparison

	Prospect	**SuperPin** [1]	**Speck** [2]	**FastTrack** [3]
Instrumentation	compiler	DBI	DBI	compiler
Slow variant	**+**	**o**	**o**	**+**
Fast variant	**+**	**-**	**-**	**+**
Application Wide	yes	yes	yes	no
Syscall support	speculative	replay only	speculative	forbidden
Developer interface	**`chkpnt`**	transparent	transparent	FastTrack region
Additional state in Slow Variant	speculative	non-speculative	non-speculative	no

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

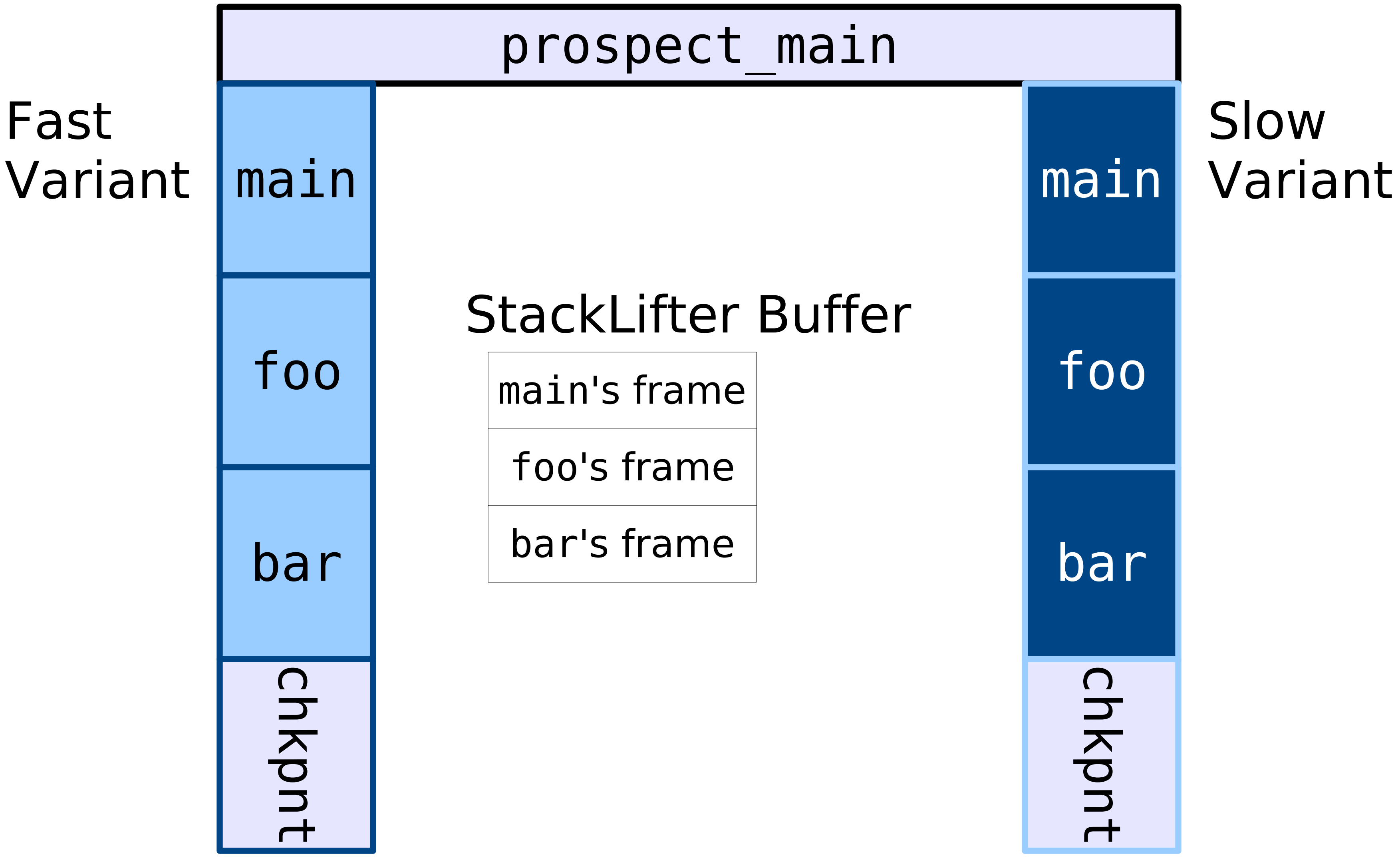
[1] S. Wallace and K. Hazelwood. *Superpin: Parallelizing dynamic instrumentation for real-time performance.* In CGO ’07: Proceedings of the International Symposium on Code Generation and Optimization, Washington, DC, USA, 2007.

[2] E. B. Nightingale, D. Peek, P. M. Chen, and J. Flinn. *Parallelizing security checks on commodity hardware.* SIGARCH Comput. Archit. News, 36 (1):308–318, 2008.

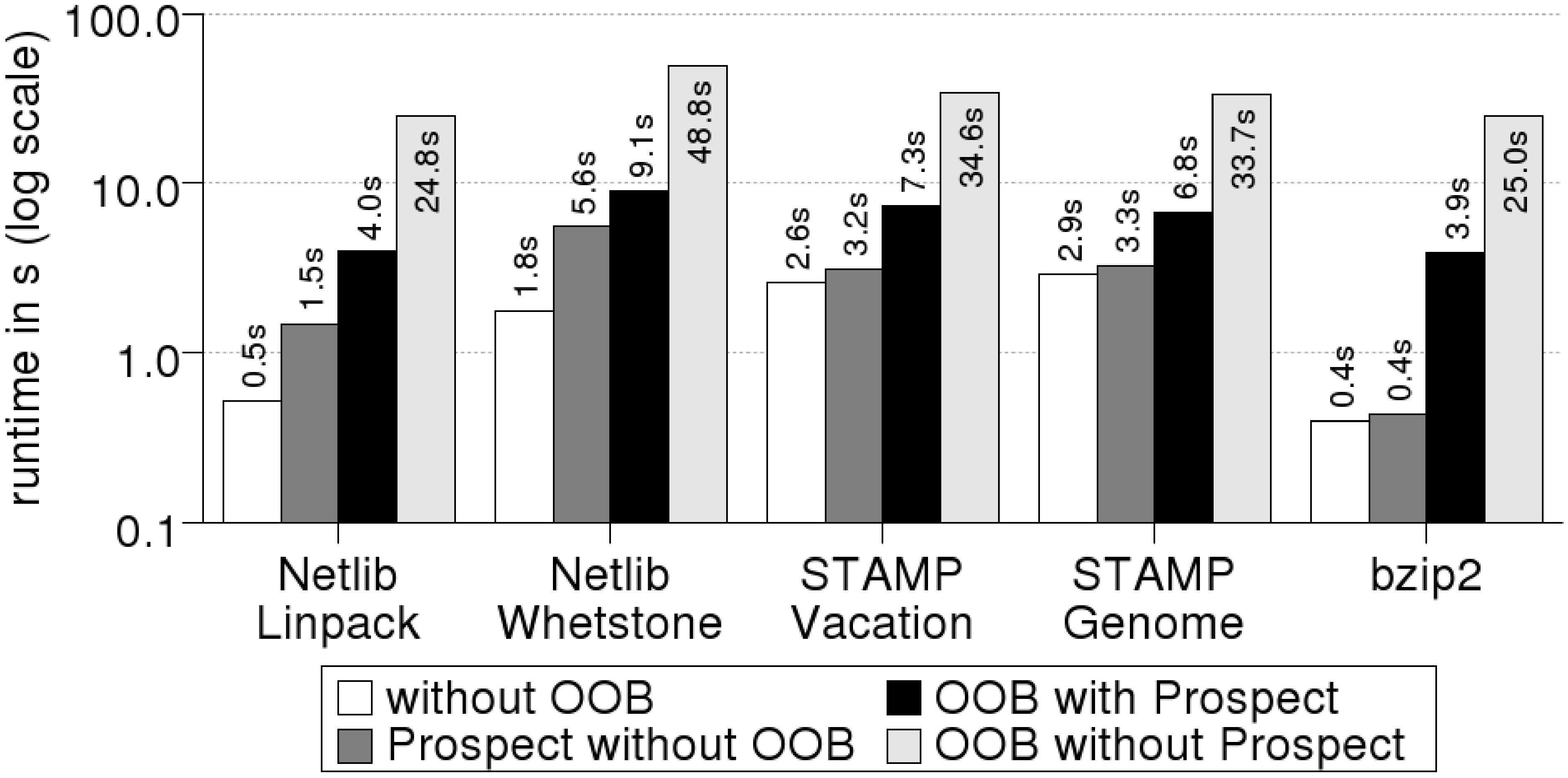
[3] K. Kelsey, T. Bai, C. Ding, and C. Zhang. *Fast track: A software system for speculative program optimization.* In CGO ’09: Proceedings of the 2009 International Symposium on Code Generation and Optimization, Washington, DC, USA, 2009.

[4] M. Süßkraut, S. Weigert, U. Schiffel, T. Knauth, M. Nowack, D. Becker de Brum, and C. Fetzer. *Speculation for parallelizing runtime checks.* In Proceedings of the 11th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS 2009), 2009.

StackLifter: On-Stack Replacement



Evaluation: Out-of-bounds (1/2)


runtime in s (log scale)
0.5s
1.5s
4.0s
24.8s
1.8s
5.6s
9.1s
48.8s
2.6s
3.2s
7.3s
34.6s
2.9s
3.3s
6.8s
33.7s
0.4s
0.4s
3.9s
25.0s
Netlib Linpack
Netlib Whetstone
STAMP Vacation
STAMP Genome
bzip2
☐ without OOB
■ OOB with Prospect
▨ Prospect without OOB
☐ OOB without Prospect