# The Deconstruction of Dyninst

Andrew Bernat, Bill Williams

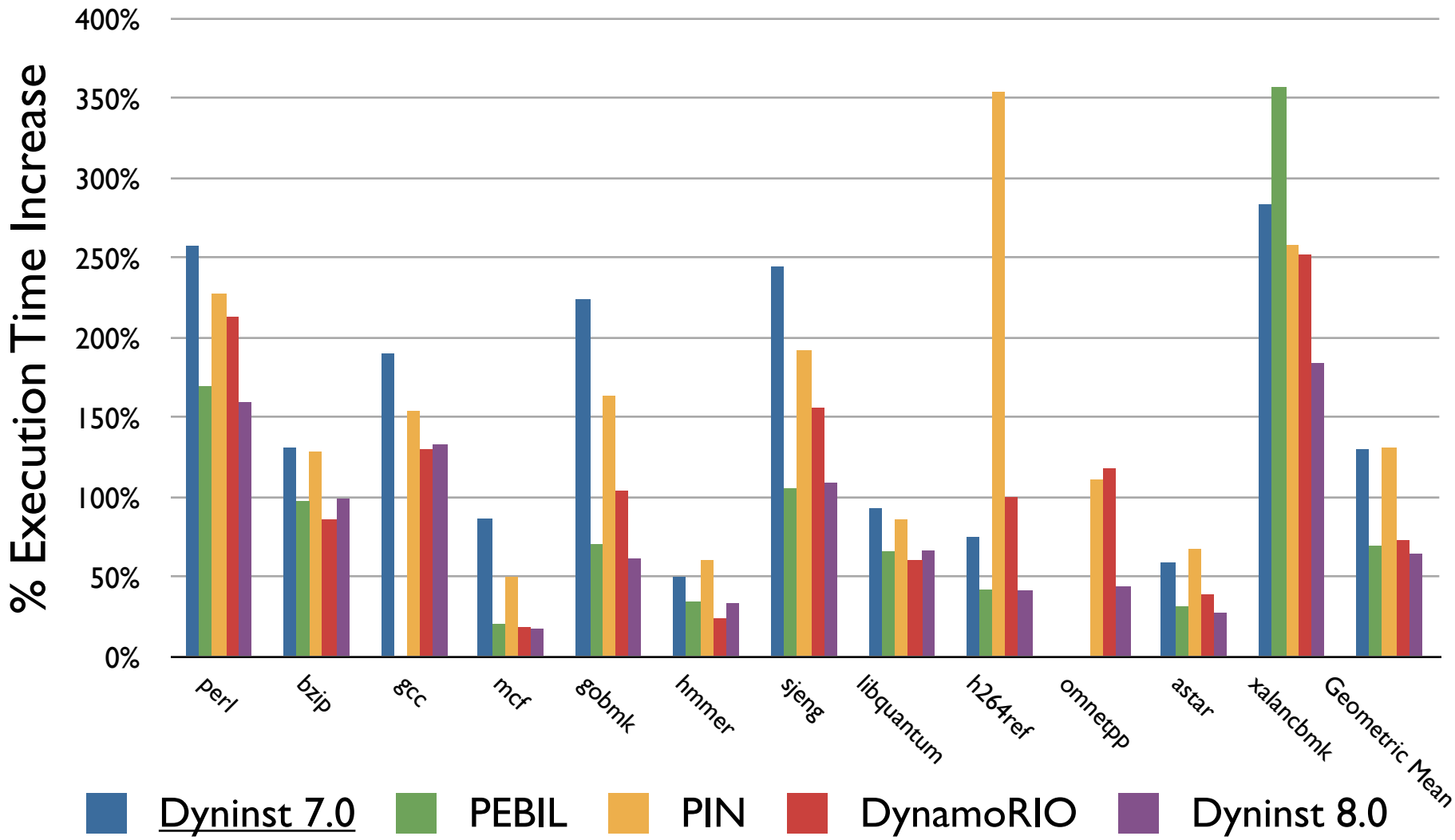
Paradyn Project

CScADS
June 26, 2012

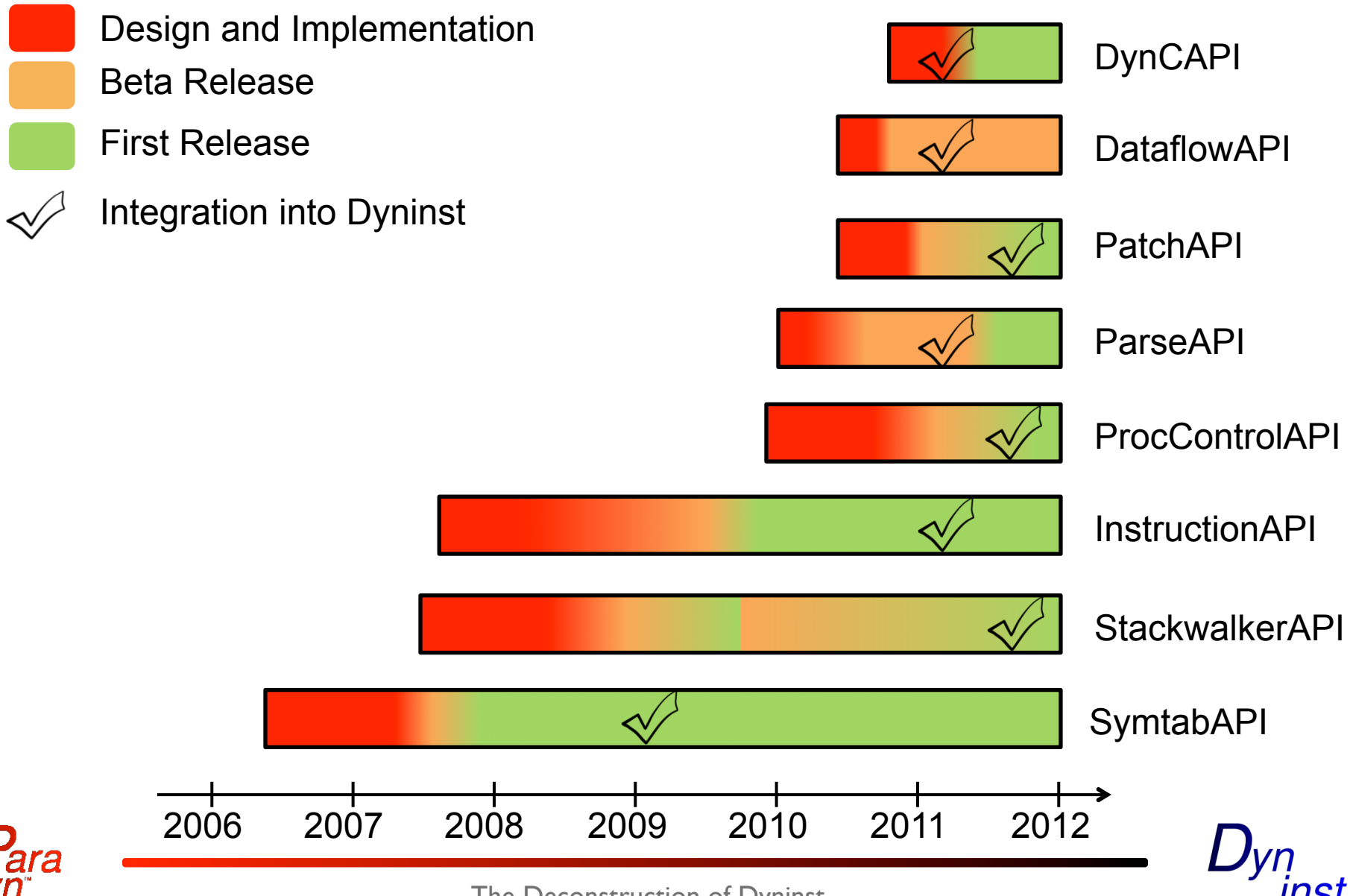

# Dyninst 8.0

- Component integration
  - ProcControlAPI
  - StackwalkerAPI
  - PatchAPI
- Additional analyses
  - Register liveness
  - Improved stack height
- Significantly reduced overhead
- Additional platforms: PPC-64, BlueGene

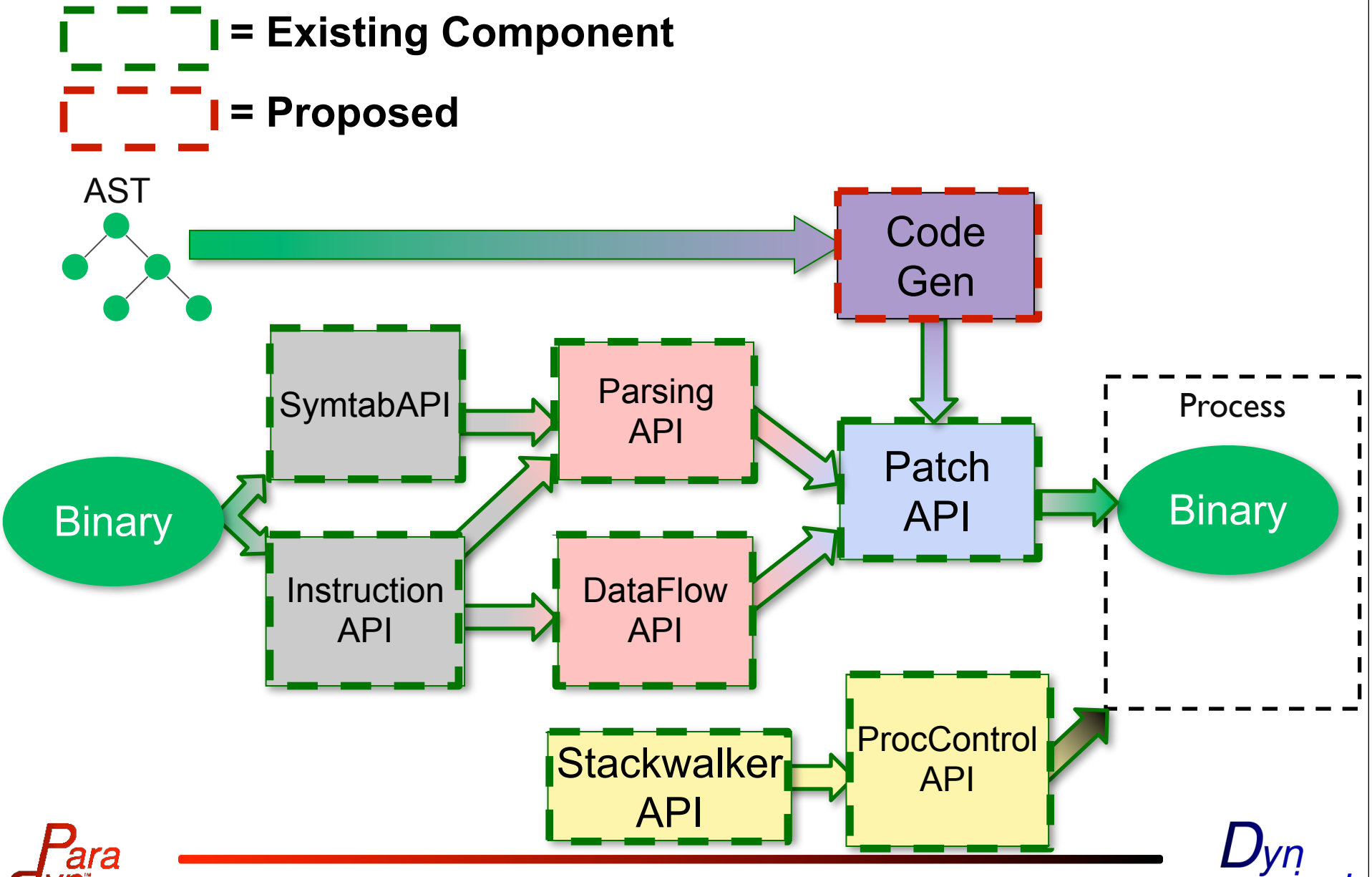
# Performance Improvements



# Dyninst Components Timeline



# Dyninst and the Components



# Programming with Dyninst and Components

- Dyninst user interface is backwards compatible
- Component interfaces are more capable
- Goal: Dyninst as thin veneer over components

```
PatchMgrPtr PatchAPI::convert(BPatch_addressSpace *);

PatchBlock *PatchAPI::convert(BPatch_basicBlock *);

Block *ParseAPI::convert(BPatch_basicBlock *);

Symtab *SymtabAPI::convert(BPatch_module *);
```

# Component Challenges

Concurrency

+

Incomplete and inconsistent interfaces

=

High-performance process control

# ProcControlAPI

- Entirely reengineered stop/continue logic
- Simplified RPC interface
- Process group support
- Hardware breakpoint support

- Platform support
  - BlueGene
  - Windows

# StackwalkerAPI

- Binary analysis frameStepper
  - Improves stack walk accuracy in frameless functions
  - Fallback option if cheaper steppers fail

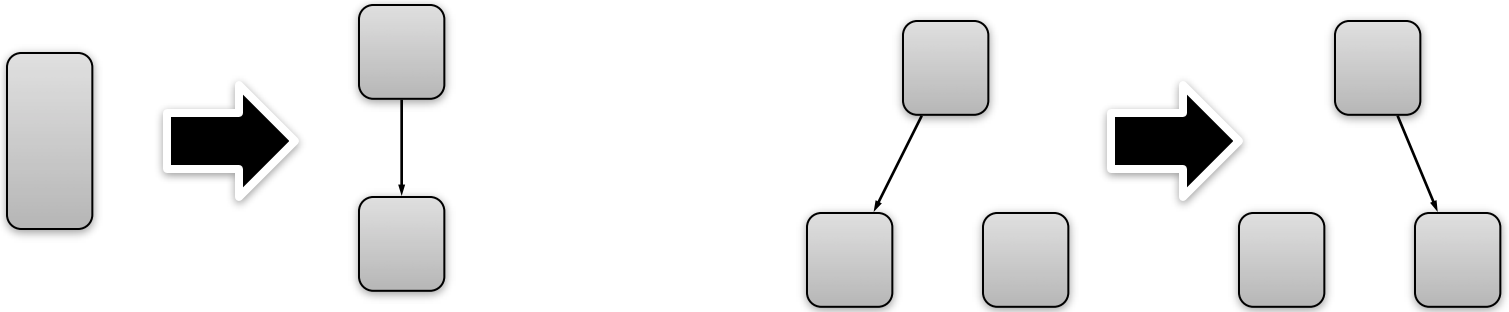
- 3rd party stack walking through ProcControlAPI
  - Improved portability
  - More capable process control interface

# PatchAPI – Binary Modification

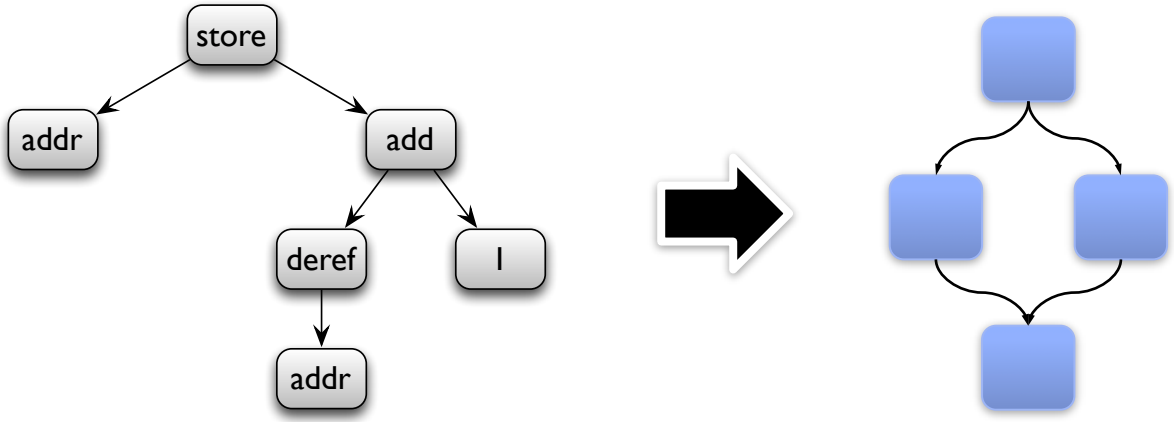
- Use familiar abstractions
  - CFG
  - Snippets
- Interactive
  - Inserted code becomes part of the CFG and can be modified further
  - Instrument modified code
- Safe
  - Avoid unexpected side-effects
  - Preserve correct control flow

# CFG Transformations

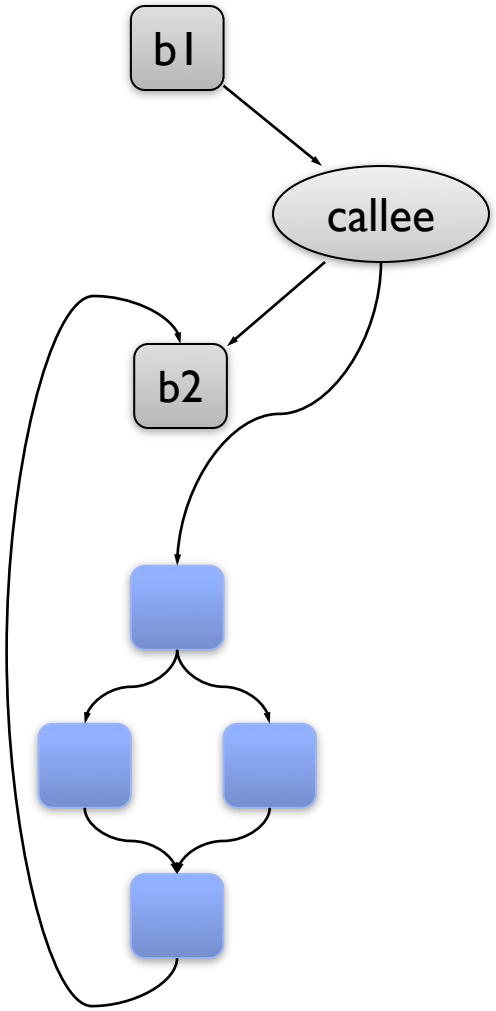
- Modifying code: block split, edge redirection



- Inserting code: snippets



# Code Insertion (Apache hotpatch tool)



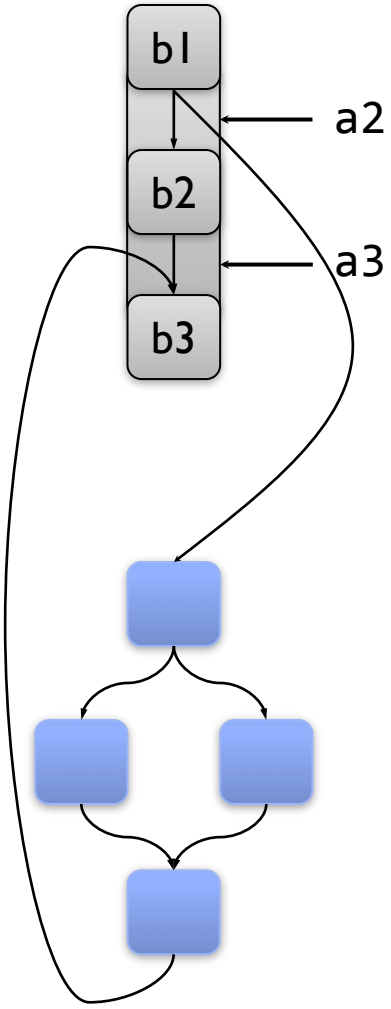

```
PatchBlock *b1, *b2;


Snippet::Ptr snip;
IC::Ptr code = PatchModifier::insert(b1->obj(),
                                     snip,
                                     b1->exit());


PatchModifier::redirect(getEdge(b1, CALL_FT),
                        code->entry());


for (iterator iter = code->exits().begin();
       iter != code->exits().end(); ++iter) {
    PatchModifier::redirect(*iter, b2);
}
```

# Code Replacement (CRAFT, Michael Lam)




```
PatchBlock *b;
Address a2, a3;

PatchBlock *b3 = PatchModifier::split(b, a3);
PatchBlock *b2 = PatchModifier::split(b, a2);
PatchBlock *b1 = b;

IC::Ptr code = PatchModifier::insert(b->obj(),
                                     snip,
                                     b2->entry());

PatchModifier::redirect(getEdge(b1, FT),
                        code->entry());

for (iterator iter = code->exits().begin();
       iter != code->exits().end(); ++iter) {
    PatchModifier::redirect(*iter, b2);
}

PatchModifier::remove(b2);
```

# CFG Modification Callbacks

- Interface class for CFG modification updates
- Register one (or more) child classes
- Notify on CFG element:
  - Creation
  - Destruction
  - Block splitting
  - New in-edge or out-edge
  - Removed in-edge or out-edge
- Notify on Point creation, destruction, or change

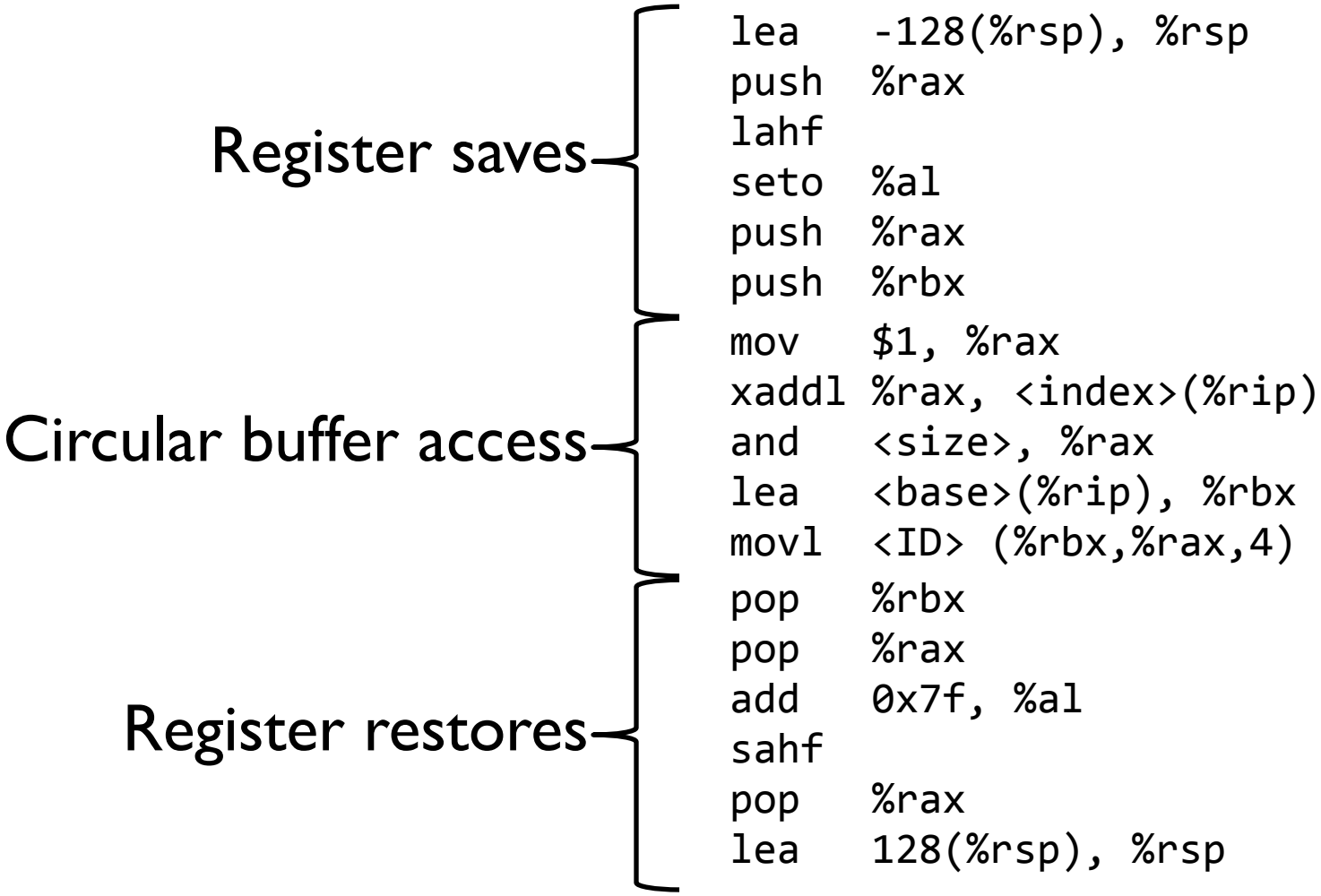
# PatchAPI – User-defined snippets

- Allow users to insert their own code
  - Floating point
  - Access to complex data structures
  - Platform-specific optimizations
  - Precompiled binary blobs
- Simple interface
  - Extensible for better code generation efficiency

# class Snippet

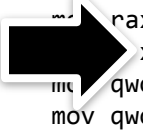
- bool generate(Point *point, Buffer &buffer);
  - point: identifies location of code generation
  - buffer: container of generated code

# Data structure accesses (boar)



# Single-precision floating point (CRAFT)

```
addsd xmm0, xmm1
```



```
addss xmm0, xmm1
```

```
nop

mov qword ptr [rsp-0xb8], rax
mov rax, 0x0
lahf
seto al
mov qword ptr [rsp-0xc0], rax
mov qword ptr [rsp-0xd0], rax
mov qword ptr [rsp-0xd8], rbx
mov qword ptr [rsp-0xe0], rcx
movq rax, xmm0
mov rbx, 0x7fffffff00000000
and rax, rbx
mov rbx, 0x7ff4dead00000000
cmp rbx, rax
jz 0x7fff1d923f6c

movq rax, xmm1
mov rbx, 0x7fffffff00000000
and rax, rbx
mov rbx, 0x7ff4dead00000000
cmp rbx, rax
jz 0x7fff1d923f6c

cvtsd2ss xmm0, xmm0
mov eax, 0x7ff4dead
mov rcx, 0xffffffff
and rax, rcx
rol rax, 0x20
movlpd qword ptr [rsp-0xe8], xmm0
mov rcx, 0xffffffff
and qword ptr [rsp-0xe8], rcx
or qword ptr [rsp-0xe8], rax
movlpd xmm0, qword ptr [rsp-0xe8]
```

```
cvtsd2ss xmm1, xmm1
mov eax, 0x7ff4dead
mov rcx, 0xffffffff
and rax, rcx
rol rax, 0x20
movlpd qword ptr [rsp-0xe8], xmm1
mov rcx, 0xffffffff
and qword ptr [rsp-0xe8], rcx
or qword ptr [rsp-0xe8], rax
movlpd xmm1, qword ptr [rsp-0xe8]

mov rax, 0x605000
mov rbx, qword ptr [rax]
inc qword ptr [rbx+0xf8]
mov rcx, qword ptr [rsp-0xe0]
mov rbx, qword ptr [rsp-0xd8]
mov rax, qword ptr [rsp-0xd0]
mov rax, qword ptr [rsp-0xb8]
movq xmm0, xmm1
mov qword ptr [rsp-0xb8], rax
mov qword ptr [rsp-0xd0], rax
mov qword ptr [rsp-0xd8], rcx
mov eax, 0x7ff4dead
mov rcx, 0xffffffff
and rax, rcx
rol rax, 0x20
movlpd qword ptr [rsp-0xe0], xmm0
mov rcx, 0xffffffff
and qword ptr [rsp-0xe0], rcx
or qword ptr [rsp-0xe0], rax
movlpd xmm0, qword ptr [rsp-0xe0]
mov rcx, qword ptr [rsp-0xd8]
mov rax, qword ptr [rsp-0xd0]
mov rax, qword ptr [rsp-0xc0]
add al, 0x7f
sahf
mov rax, qword ptr [rsp-0xb8]
```

# Dyninst 8.0

- Coming soon!
  - Individual component integration complete
  - Final merging in progress
- Great features and new platform support
- Beta access upon request

# Research Status

- Recently finished:
  - Binary editing (Bernat)
  - Extreme scale process control and inspection (Brim)
  - Analyzing and instrumenting malicious code (Roundy)
- In flight:
  - Analysis and visualization of large systems (Fang)
  - Return address tamper detection (Jacobson)
  - Binary authorship (Meng)

Papers at www.paradyn.org