RUNTIME ADAPTATION: A CASE FOR REACTIVE CODE ALIGNMENT

Michelle McDaniel and Kim Hazelwood
University of Virginia

Introduction

- Code alignment affects the performance of applications
- Poor code alignment can cause an increase in
  - Branch mispredictions
  - Caches misses
  - Memory stalls
  - Instruction fetches

Code Alignment Example

```
int a = 0;
int i;
int ii;

for(ii = 0; ii < 500000000; i++)
{
   for(i = 0; i < 5; i++)
   {
      a++;
   }
}
```

Code Alignment Example

```
for(ii = 0; ii < 500000000; i++)
{
        for(i = 0; i < 5; i++)
        {
                a++;
        }
}
```

0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
						addl	addl	addl	addl	addl	addl	addl	cmp	cmp	cmp
cmp	jle	jle	addl	addl	addl	addl	cmp	cmp	cmp	cmp	cmp	cmp	cmp	jle	jle

Code Alignment Example

```
for(ii = 0; ii < 500000000; i++)
{
        for(i = 0; i < 5; i++)
        {
                a++;
        }
}
```

0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
						addl	addl	addl	addl	addl	addl	addl	cmp	cmp	cmp
cmp	jle	jle	addl	addl	addl	addl	cmp	cmp	cmp	cmp	cmp	cmp	cmp	jle	jle

- 250M branch mispredictions

Code Alignment Example

```
for(ii = 0; ii < 500000000; i++)
{
        for(i = 0; i < 5; i++)
        {
                a++;
        }
}
```

0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
							addl	addl	addl	addl	addl	addl	addl	cmp	cmp
cmp	cmp	jle	jle	addl	addl	addl	addl	cmp	cmp	cmp	cmp	cmp	cmp	cmp	jle
jle															

Code Alignment Example

```
for(ii = 0; ii < 500000000; i++)
{
        for(i = 0; i < 5; i++)
        {
                a++;
        }
}
```

0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
							addl	addl	addl	addl	addl	addl	addl	cmp	cmp
cmp	cmp	jle	jle	addl	addl	addl	addl	cmp	cmp	cmp	cmp	cmp	cmp	cmp	jle
jle															

- 44K branch mispredictions

Code Alignment Example

```
for(ii = 0; ii < 500000000; i++)
{
        for(i = 0; i < 5; i++)
        {
                a++;
        }
}
```

0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
							addl	addl	addl	addl	addl	addl	addl	cmp	cmp
cmp	cmp	jle	jle	addl	addl	addl	addl	cmp	cmp	cmp	cmp	cmp	cmp	cmp	jle
jle															

- 44K branch mispredictions
- 21% faster than the original version

Code Alignment Example

0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
						addl	addl	addl	addl	addl	addl	addl	cmp	cmp	cmp
cmp	jle	jle	addl	addl	addl	addl	cmp	cmp	cmp	cmp	cmp	cmp	cmp	jle	jle

							addl	addl	addl	addl	addl	addl	addl	cmp	cmp
cmp	cmp	jle	jle	addl	addl	addl	addl	cmp	cmp	cmp	cmp	cmp	cmp	cmp	jle
jle															

- Branch prediction issue on Intel Core and Core2 processors
- Branch collisions cause increased branch misprediction rate

Limitations of Static Alignment

- Cannot align for specific microarchitectural features (branch collisions, loop stream detector, etc)
- Single alignment for all inputs
- Predicting branch behavior is hard

Limitations of Static Alignment

- Cannot align for specific microarchitectural features (branch collisions, loop stream detector, etc)
- Single alignment for all inputs
- Difficult to predict runtime branch behavior

Reactive realignment can avoid these limitations.

Three Alignment Questions

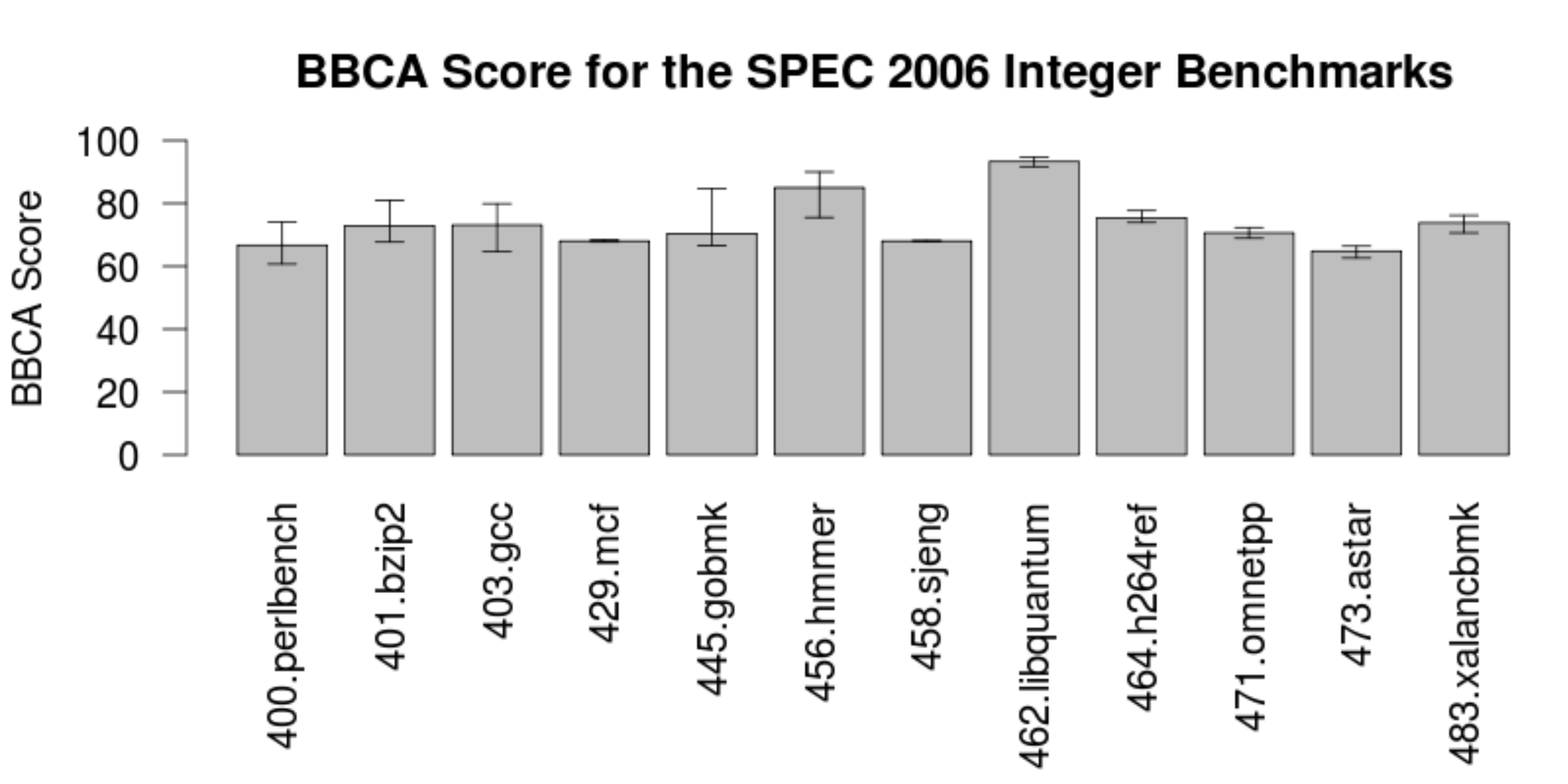
- How do we know an application is poorly aligned?
- What causes these alignment issues?
- What can we do about it?

How do we know that an application is poorly aligned?

How do we know?

- Basic Block Code Alignment Score
  - Static alignment measurement
- Runtime Triggers
  - Branch mispredictions
  - Fetches-per-instruction

BBCA Score


BBCA Score for the SPEC 2006 Integer Benchmarks
BBCA Score
100
80
60
40
20
0
400.perlbench
401.bzip2
403.gcc
429.mcf
445.gobmk
456.hmmer
458.sjeng
462.libquantum
464.h264ref
471.omnetpp
473.astar
483.xalancbmk

Runtime Triggers

- Triggers suggest an application is poorly aligned
- Issues to be monitored:
  - Increase in instruction fetches
  - Branch mispredictions

Runtime Triggers

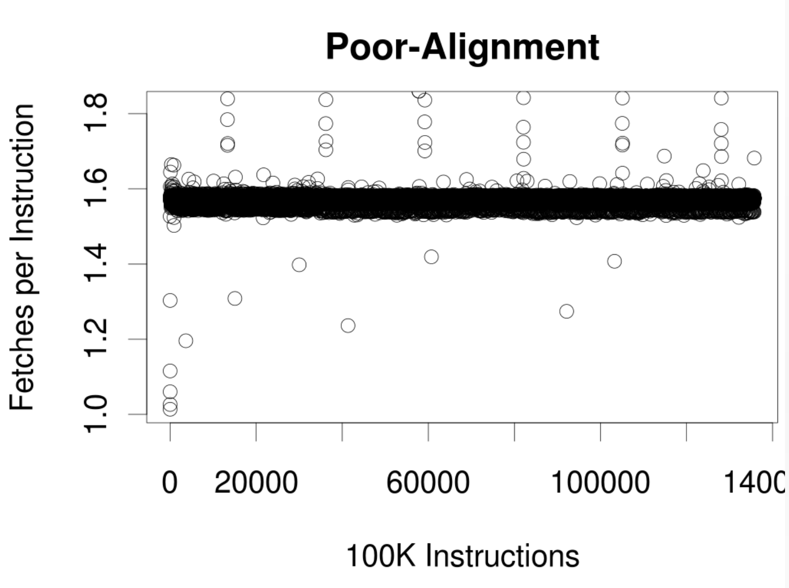
```
for(ii = 0; ii < 500000000; i++)
{
    for(i = 0; i < 5; i++)
    {
        a++;
    }
}
```

0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
						addl	addl	addl	addl	addl	addl	addl	cmp	cmp	cmp
cmp	jle	jle	addl	addl	addl	addl	cmp	cmp	cmp	cmp	cmp	cmp	cmp	jle	jle

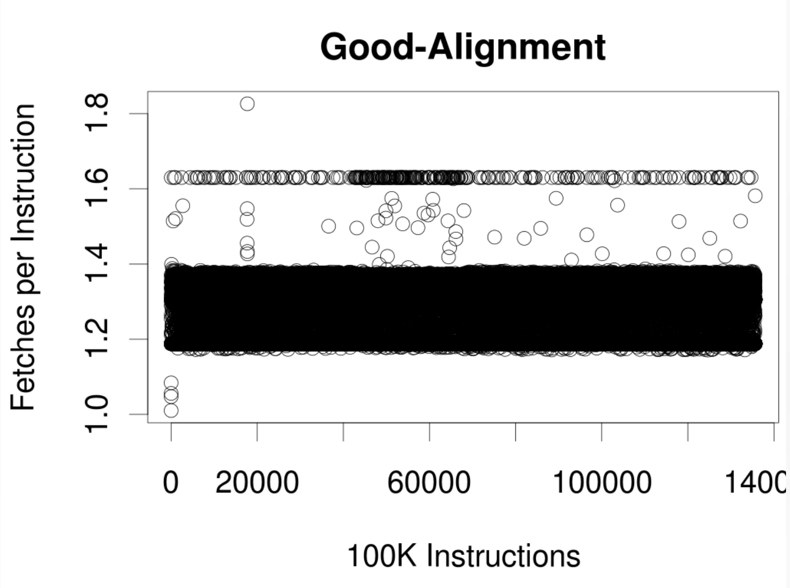
							addl	addl	addl	addl	addl	addl	addl	cmp	cmp
cmp	cmp	jle	jle	addl	addl	addl	addl	cmp	cmp	cmp	cmp	cmp	cmp	cmp	jle
jle															

Runtime Triggers

Alignment for poorly aligned branch test

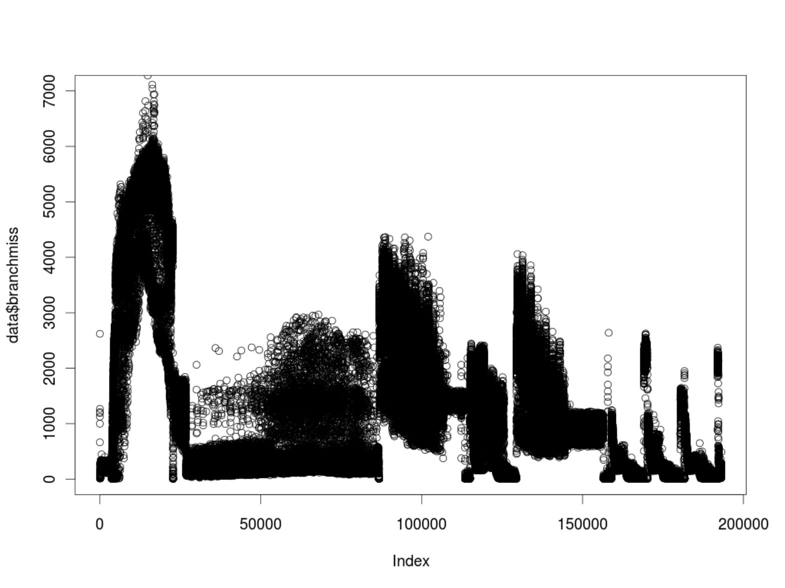


Alignment for correctly aligned branch test

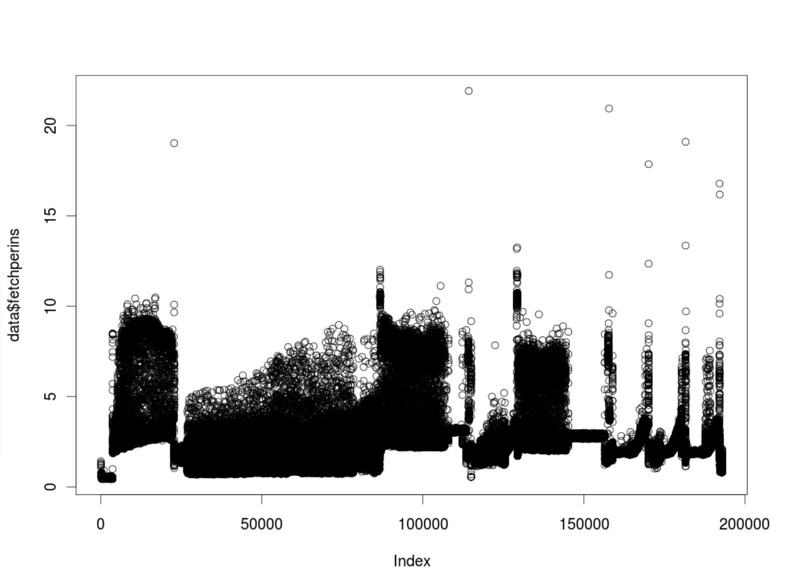


Runtime Triggers

Branch mispredictions for 429.mcf



Fetches per instruction for 429.mcf



Spikes tell us that alignment is poor at that particular code section

What causes these alignment issues?

What causes alignment issues?

- Microarchitectural differences
- Program inputs
  - Different execution paths
- Dynamic branch behavior
  - Indirect branches
  - Phase changes

Microarchitectural Differences

- Microarchitectural features can either help or hurt program performance
- Code alignment must adapt to the changes in microarchitecture to exploit or accommodate certain microarchitectural features

Microarchitectural Differences

```
for(ii = 0; ii < 500000000; i++)
{
    for(i = 0; i < 5; i++)
    {
        a++;
    }
}
```

0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
						addl	addl	addl	addl	addl	addl	addl	cmp	cmp	cmp
cmp	jle	jle	addl	addl	addl	addl	cmp	cmp	cmp	cmp	cmp	cmp	cmp	jle	jle

							addl	addl	addl	addl	addl	addl	addl	cmp	cmp
cmp	cmp	jle	jle	addl	addl	addl	addl	cmp	cmp	cmp	cmp	cmp	cmp	cmp	jle
jle															

Microarchitectural Differences

0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
						addl	addl	addl	addl	addl	addl	addl	cmp	cmp	cmp
cmp	jle	jle	addl	addl	addl	addl	cmp	cmp	cmp	cmp	cmp	cmp	cmp	jle	jle

							addl	addl	addl	addl	addl	addl	addl	cmp	cmp
cmp	cmp	jle	jle	addl	addl	addl	addl	cmp	cmp	cmp	cmp	cmp	cmp	cmp	jle
jle															

- Core and Core2
  - First alignment is 21% slower
  - Branch collisions

Microarchitectural Differences

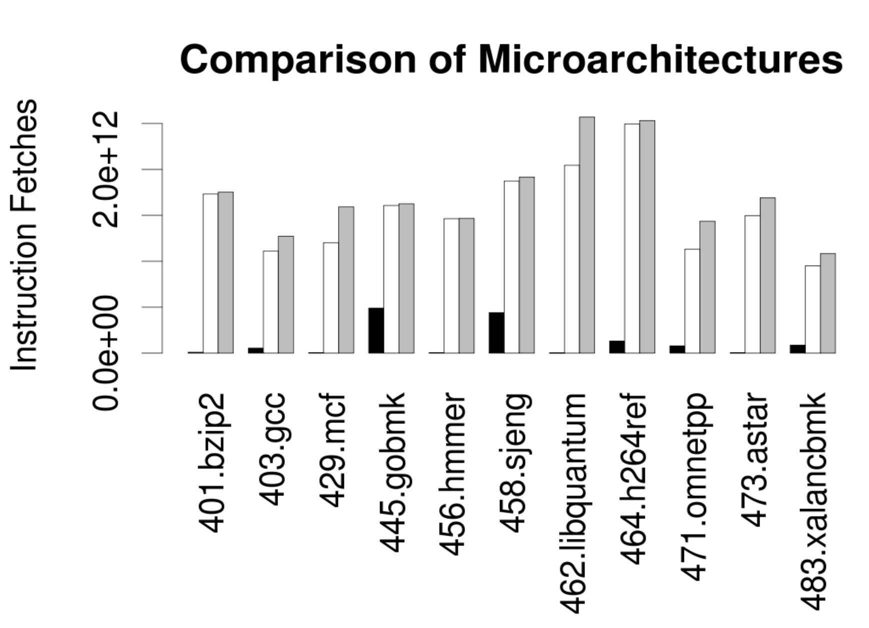
0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
						addl	addl	addl	addl	addl	addl	addl	cmp	cmp	cmp
cmp	jle	jle	addl	addl	addl	addl	cmp	cmp	cmp	cmp	cmp	cmp	cmp	jle	jle

							addl	addl	addl	addl	addl	addl	addl	cmp	cmp
cmp	cmp	jle	jle	addl	addl	addl	addl	cmp	cmp	cmp	cmp	cmp	cmp	cmp	jle
jle															

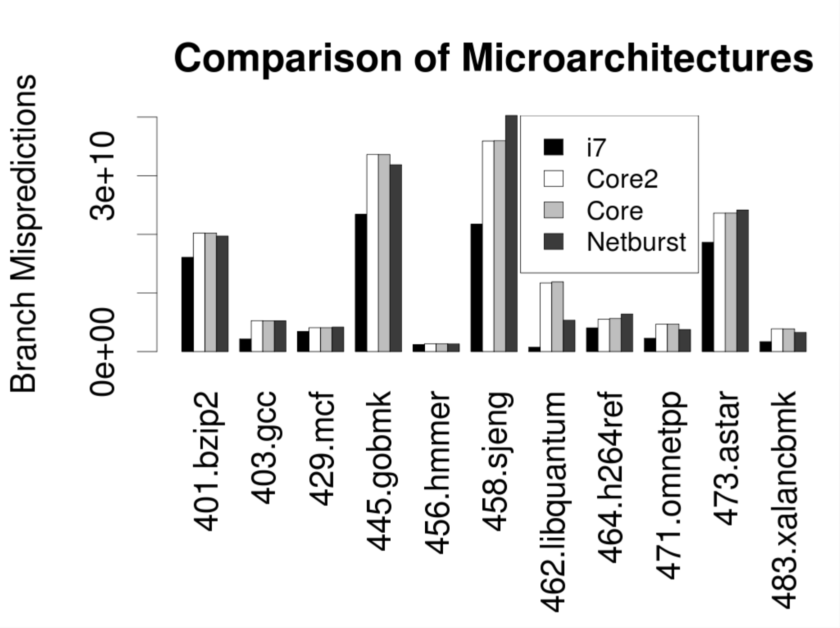
- Core and Core2
  - First alignment is 21% slower
  - Branch collisions
- Netburst and i7
  - Second alignment is 2% slower
  - No branch collisions

Microarchitectural Differences

Instruction Fetches



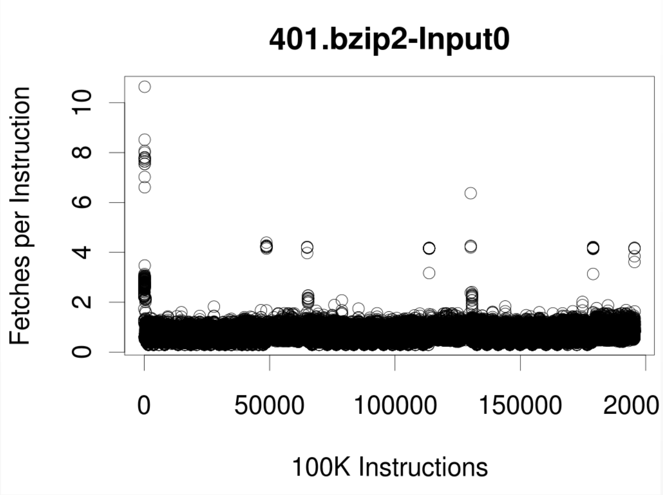
Branch Mispredictions

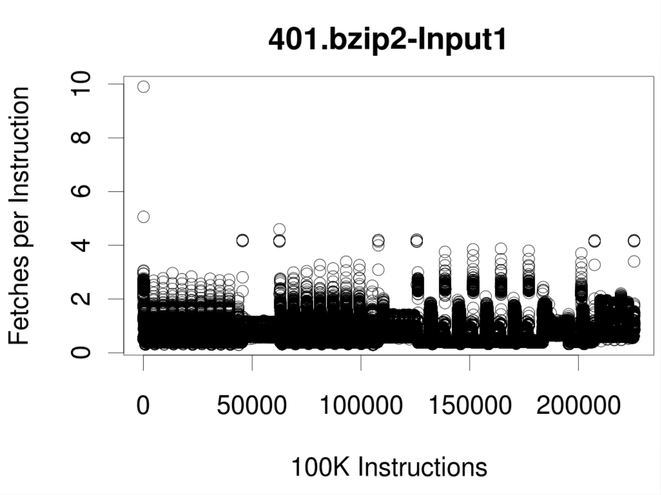


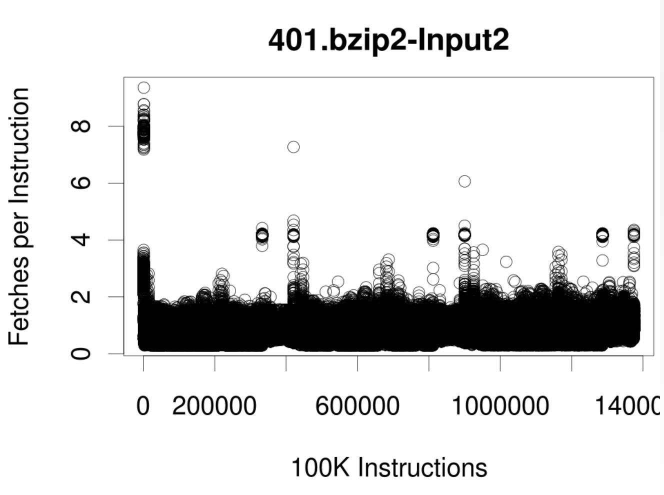
Program Inputs

- At compile time, we know little about the most frequent paths of a program
- Most frequently executed paths change based on the input
- Dynamically, we can react to the current execution to make up for these compiler limitations

Program Inputs – Case Study







Runtime Branch Behavior

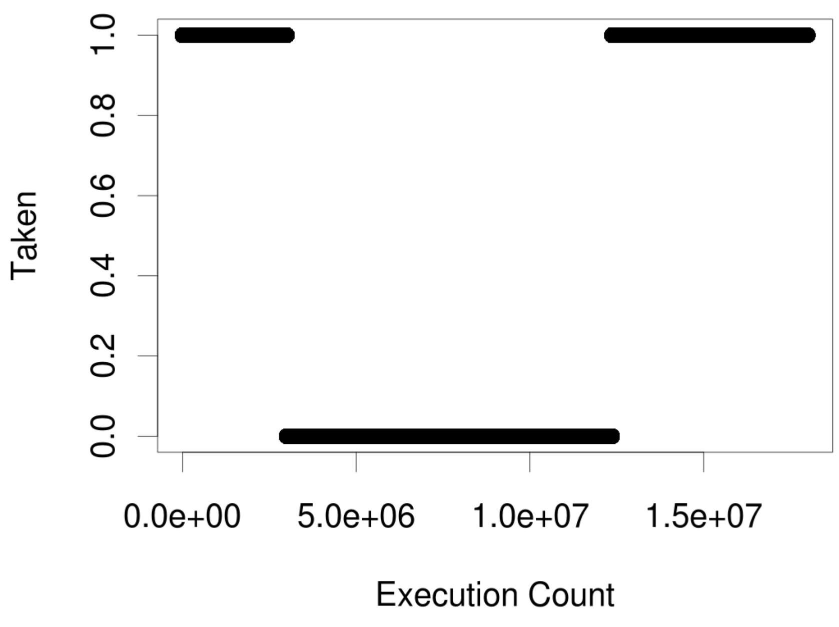
- Branch behavior is hard to predict at compile time
- Indirect branches
- Phase changes

Indirect Branch Behavior

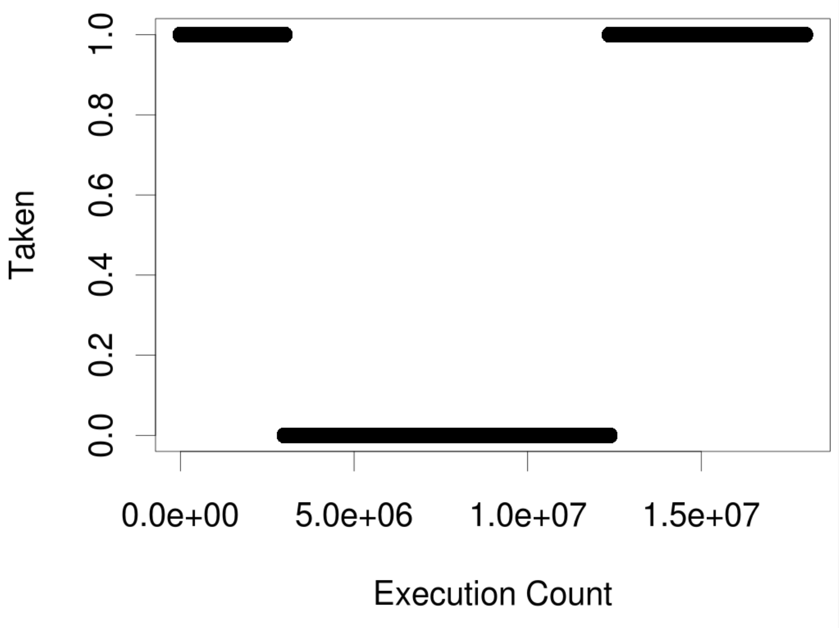
Benchmark	Average Static Indirect Branch Targets	Average Dynamic Indirect Branch Targets
400.perlbench	19.7	7.8
401.bzip2	24.0	7.7
403.gcc	13.3	7.6
445.gobmk	17.9	2.8
456.hmmer	9.2	1
458.sjeng	8.7	7.8
464.h264ref	6.8	4.7
471.omnetpp	7.9	3.1
483.xalancbmk	10.7	1.2

Phase Changes – Case Study

429.mcf



458.sjeng



- 429.mcf: branch nested in a while loop
  - Jump target is not well aligned due to compiler heuristic limitation
- 458.sjeng: case in a switch statement

What can we do about poor code alignment?

What can we do about poor code alignment?

Reactive Realignment

What can we do about it?

- Runtime realignment
- Monitor runtime triggers
- Adapt alignment as we notice symptoms of poor alignment
- **Future work**: Incorporate a reactive realignment system into dynamic optimization schemes (JIT)

Conclusions

- Alignment is important to the performance of applications
- Static alignment techniques have several limitations they cannot overcome
- Reactive alignment systems can align for microarchitectural differences, program inputs, and dynamic branch behavior
- We can use fetches-per-instruction as a trigger for a reactive systems

Questions?