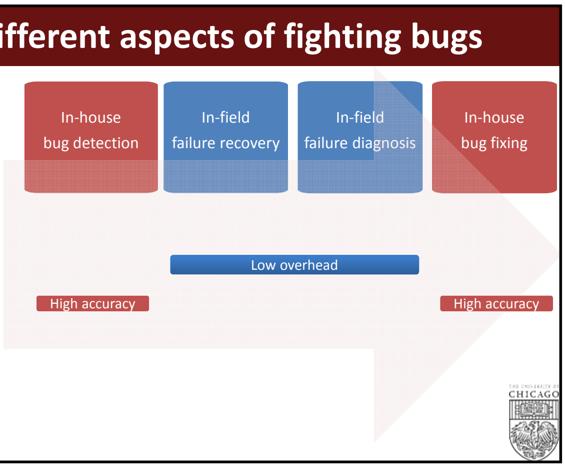
Production-Run Failures Diagnosis for Concurrency Bugs

Shan Lu
University of Chicago

SL41

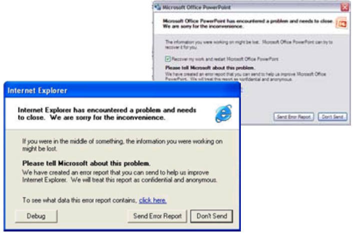
Different aspects of fighting bugs



SL35

Failure diagnosis is challenging

- Limited information
- Failures are difficult to repeat
- Root causes are difficult to reason about



Microsoft Office PowerPoint

Microsoft Office PowerPoint has encountered a problem and needs to close. We are sorry for the inconvenience.

The information you were working on might be lost. Microsoft Office PowerPoint can try to recover it for you.

☑ Recover my work and restart Microsoft Office PowerPoint

Please tell Microsoft about this problem.
We have created an error report that you can send to help us improve Microsoft Office PowerPoint. We will treat this report as confidential and anonymous.

Internet Explorer

Internet Explorer has encountered a problem and needs to close. We are sorry for the inconvenience.

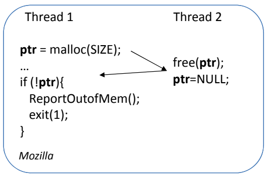
If you were in the middle of something, the information you were working on might be lost.

Please tell Microsoft about this problem.
We have created an error report that you can send to help us improve Internet Explorer. We will treat this report as confidential and anonymous.

To see what data this error report contains, click here.

A15

Example



Thread 1

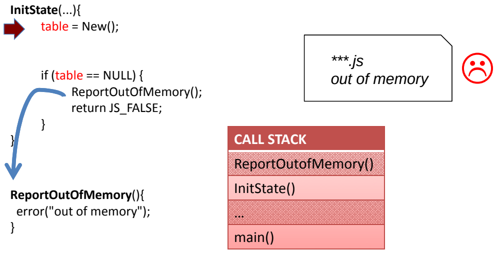
```
ptr = malloc(SIZE);
...
if (!ptr){
  ReportOutofMem();
  exit(1);
}
```

Thread 2

```
free(ptr);
ptr=NULL;
```

Mozilla

Example



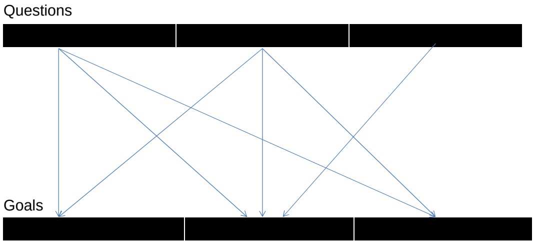
```
InitState(...){
    table = New();


    if (table == NULL) {
        ReportOutOfMemory();
        return JS_FALSE;
    }
}


ReportOutOfMemory(){
  error("out of memory");
}
```

****.js*
out of memory

☹

CALL STACK
ReportOutofMemory()
InitState()
...
main()

Design space



Slide 2

SL41 ideally, this should be a cycle, but ...

Shan Lu, 2014-1-7

Slide 3

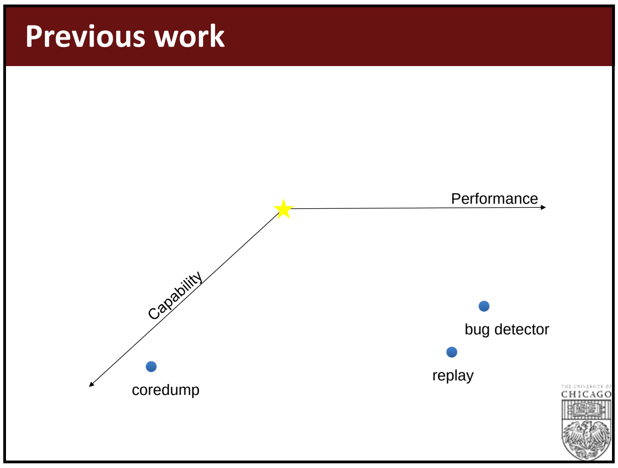
SL35 if i have time, i can turn these into developers quotes

Shan Lu, 2014-1-15

Slide 4

A15 i need to replace this with Joy's version

Administratr, 2014-3-5



Previous work

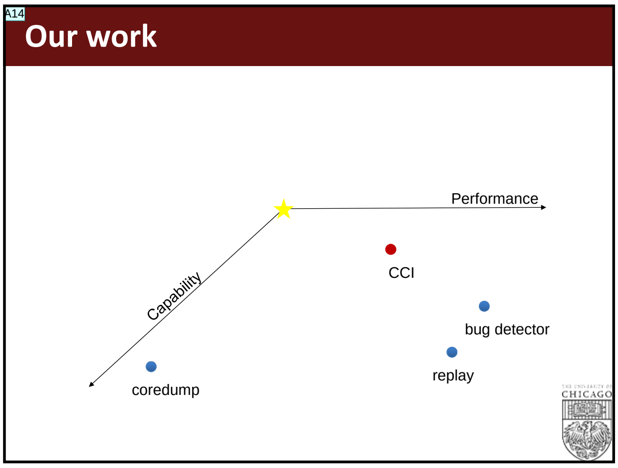
Performance

Capability

bug detector

replay

coredump



A14

Our work

Performance

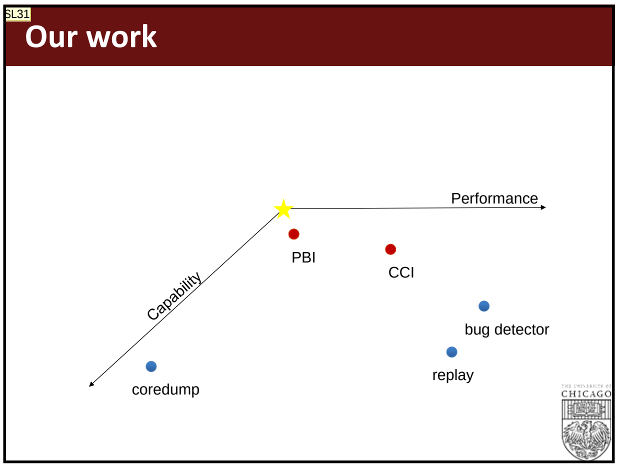
Capability

CCI

bug detector

replay

coredump



SL31

Our work

Performance

Capability

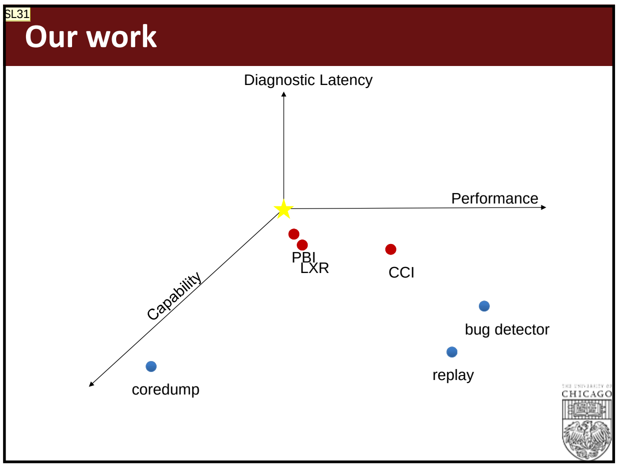
PBI

CCI

bug detector

replay

coredump



SL31

Our work

Diagnostic Latency

Performance

Capability

PBI

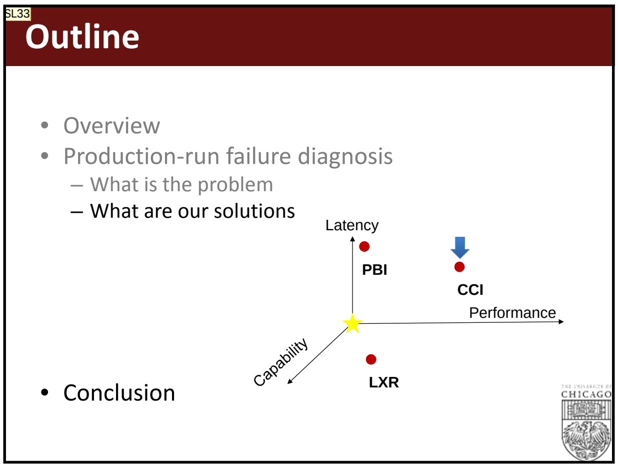
LXR

CCI

bug detector

replay

coredump



SL33

Outline

- Overview
- Production-run failure diagnosis
  - What is the problem
  - What are our solutions

Latency

PBI

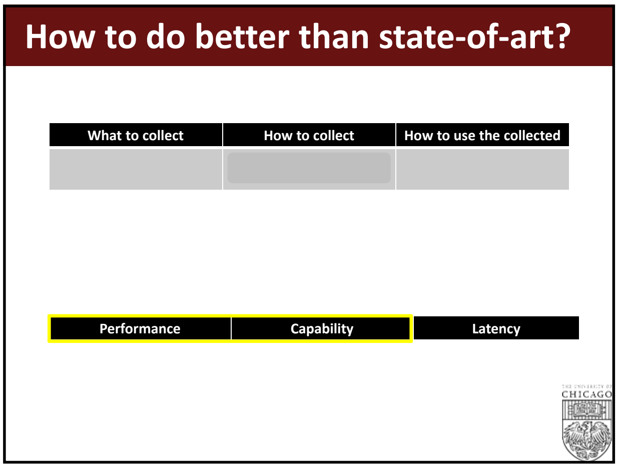
CCI

Performance

Capability

LXR

- Conclusion



How to do better than state-of-art?

What to collect	How to collect	How to use the collected

Performance	Capability	Latency

Slide 8

A14 simplify these. put
statistical approach, compiler, cause-pattern
hardware support
hardware extension, effect-pattern
in one text box, keep growing.

change the cloud shape. simplify both the slide and the script
Administratr, 2014-3-4

Slide 9

SL31 maybe i should put 4-d/3-d coordinates here, and change the tables following
Shan Lu, 2014-1-15

Slide 10

SL31 maybe i should put 4-d/3-d coordinates here, and change the tables following
Shan Lu, 2014-1-15

Slide 11

SL33 change the bullets texts. things like "compiler-based" is strange.
Shan Lu, 2014-1-15

How to do better than state-of-art?

What to collect	How to collect	How to use the collected
	Sampling	

Performance	Capability	Latency

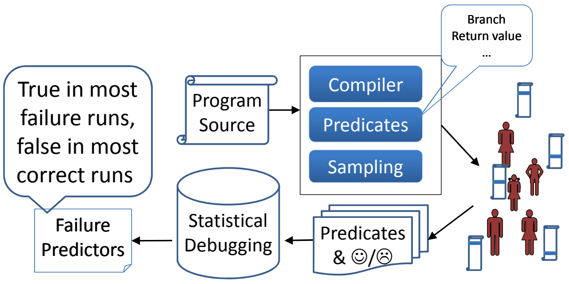
How to do better than state-of-art?

What to collect	How to collect	How to use the collected
	Sampling	Cooperative statistical analysis

Performance	Capability	Latency

SL20 SL34

Cooperative Bug Isolation (CBI)

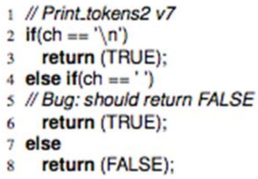


Performance	Capability
Good	??

A long story about CBI

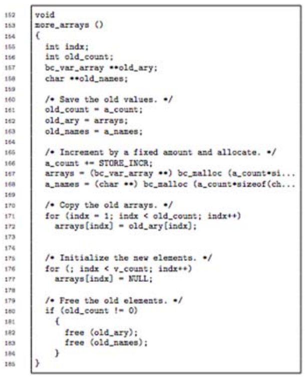
- Statistical fault localization, delta debugging
- Sampling based statistical fault localization

An example



```
// Print_tokens2 v7
if(ch == '\n')
    return (TRUE);
else if(ch == ' ')
// Bug: should return FALSE
    return (TRUE);
else
    return (FALSE);
```

Another example



```
void
more_arrays ()
{
  int indx;
  int old_count;
  bc_var_array **old_ary;
  char **old_names;

  /* Save the old values. */
  old_count = a_count;
  old_ary = arrays;
  old_names = a_names;

  /* Increment by a fixed amount and allocate. */
  a_count += STORE_INCR;
  arrays = (bc_var_array **) bc_malloc (a_count*si...
  a_names = (char **) bc_malloc (a_count*sizeof(ch...

  /* Copy the old arrays. */
  for (indx = 1; indx < old_count; indx++)
    arrays[indx] = old_ary[indx];


  /* Initialize the new elements. */
  for (; indx < v_count; indx++)
    arrays[indx] = NULL;

  /* Free the old elements. */
  if (old_count != 0)
    {
      free (old_ary);
      free (old_names);
    }
}
```

Slide 15

SL20 do i need to provide a sequential bug diagnosis example?
Shan Lu, 2014-1-10

SL34 should i add an overview slide before this saying: challenges; solutions: apply xxx to concurrency bug diagnosis.
Shan Lu, 2014-1-15

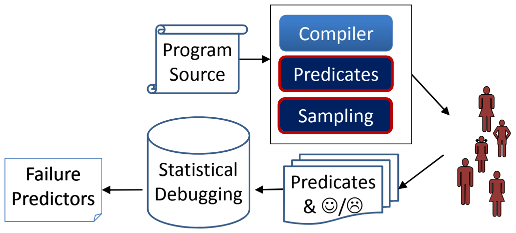
Does it work for concurrency bugs?

```
Thread 1                          Thread 2

ptr = malloc(SIZE);
…                                 free(ptr);
if (!ptr){ //b                    ptr=NULL;
  ReportOutofMem();
  exit(1);
}
```

Predicate
…
taken$_b$
!taken$_b$
…

Why does CBI not work?

Cooperative Con-Bug Isolation (CCI)



Performance	Capability
Mixed	Good

Instrumentation and Sampling Strategies for Cooperative Concurrency Bug Isolation, OOPSLA'10

SL42

What to collect? (predicate design)

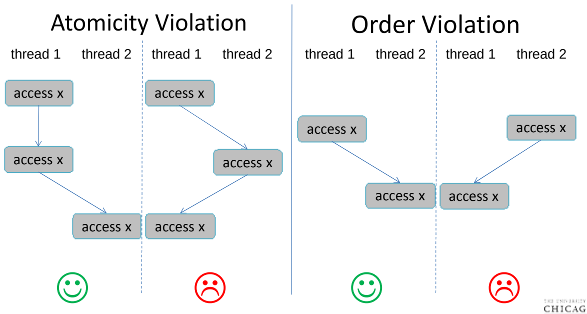
Concurrency bug root cause patterns

Atomicity Violation

Order Violation

Learning from Mistakes --- A Comprehensive Study on Real World Concurrency Bug Characteristics, ASPLOS'08

Concurrency bug root cause patterns



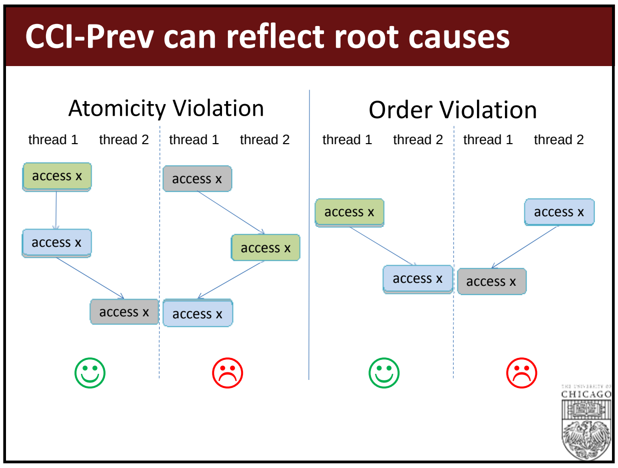
CCI-Prev predicate

Whether two successive accesses to a memory location were by two distinct threads or one thread

Slide 21

SL42 i need to redraw this to be consistent with earlier ...

Shan Lu, 2014-1-16



CCI-Prev can reflect root causes

Atomicity Violation

thread 1 thread 2 | thread 1 thread 2

access x
access x
access x

access x
access x
access x

☺ ☹

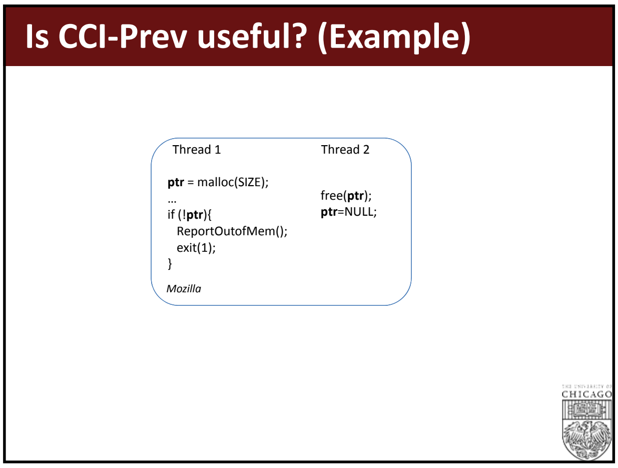
Order Violation

thread 1 thread 2 | thread 1 thread 2

access x
access x

access x
access x

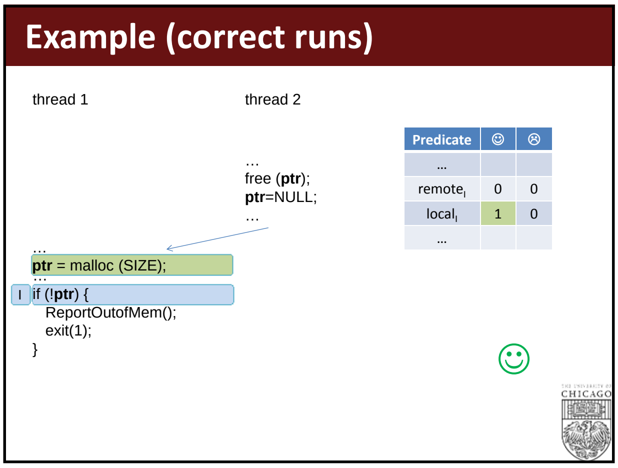
☺ ☹



Is CCI-Prev useful? (Example)

```
Thread 1                      Thread 2

ptr = malloc(SIZE);
                              free(ptr);
…                             ptr=NULL;
if (!ptr){
  ReportOutofMem();
  exit(1);
}
```

Mozilla

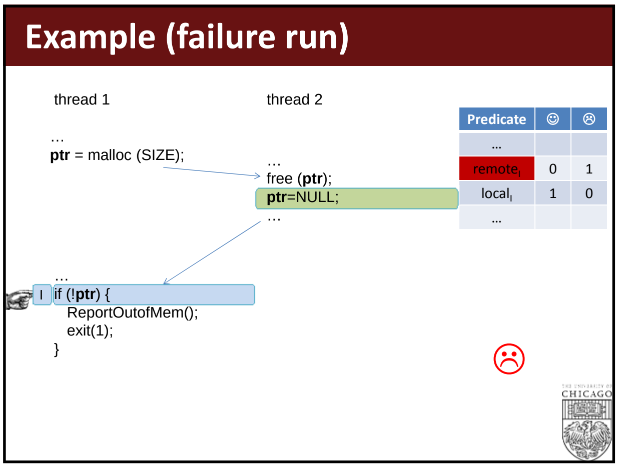


Example (correct runs)

```
thread 1                      thread 2

                              …
                              free (ptr);
                              ptr=NULL;
                              …
…
ptr = malloc (SIZE);
…
I  if (!ptr) {
     ReportOutofMem();
     exit(1);
   }
```

Predicate	☺	☹
…		
remote$_l$	0	0
local$_l$	1	0
…		

☺

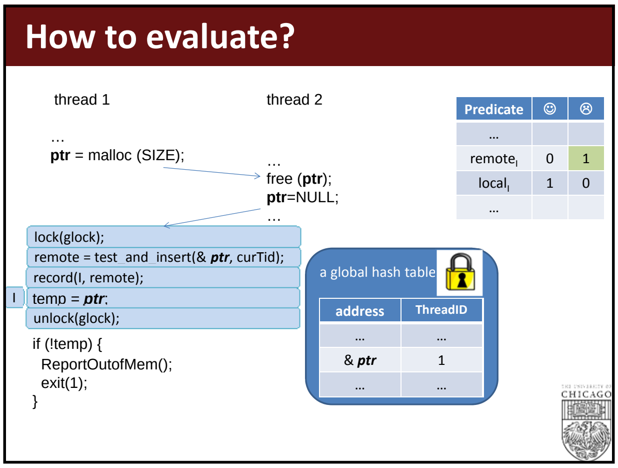


Example (failure run)

```
thread 1                      thread 2

…
ptr = malloc (SIZE);
                              …
                              free (ptr);
                              ptr=NULL;
                              …
…
I  if (!ptr) {
     ReportOutofMem();
     exit(1);
   }
```

Predicate	☺	☹
…		
remote$_l$	0	1
local$_l$	1	0
…		

☹



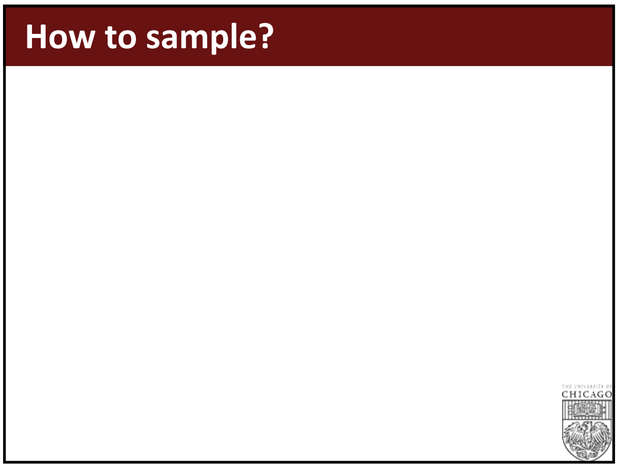
How to evaluate?

```
thread 1                      thread 2

…
ptr = malloc (SIZE);
                              …
                              free (ptr);
                              ptr=NULL;
                              …
lock(glock);
remote = test_and_insert(& ptr, curTid);
record(I, remote);
I  temp = ptr;
unlock(glock);
if (!temp) {
  ReportOutofMem();
  exit(1);
}
```

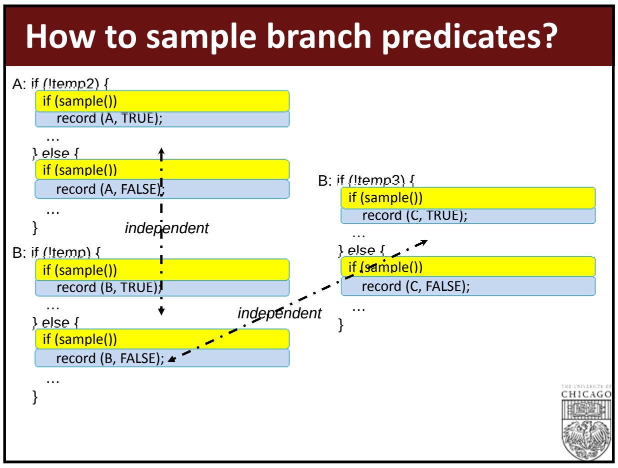
Predicate	☺	☹
…		
remote$_l$	0	1
local$_l$	1	0
…		

a global hash table

address	ThreadID
…	…
& ***ptr***	1
…	…



How to sample?



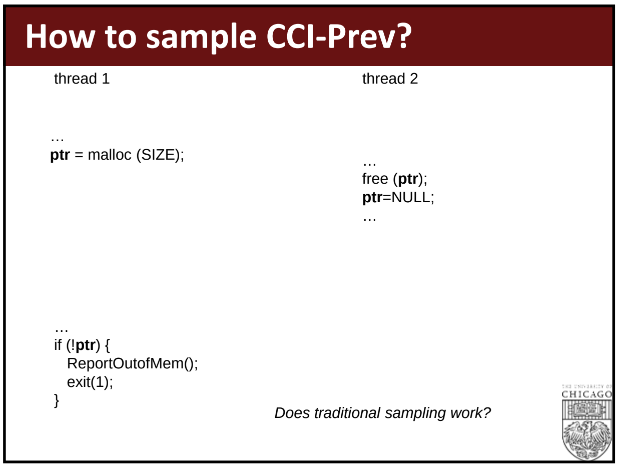
How to sample branch predicates?

```
A: if (!temp2) {
     if (sample())
       record (A, TRUE);
     …
   } else {
     if (sample())
       record (A, FALSE);
     …
   }
B: if (!temp) {
     if (sample())
       record (B, TRUE);
     …
   } else {
     if (sample())
       record (B, FALSE);
     …
   }
```

```
B: if (!temp3) {
     if (sample())
       record (C, TRUE);
     …
   } else {
     if (sample())
       record (C, FALSE);
     …
   }
```

independent

independent



How to sample CCI-Prev?

thread 1

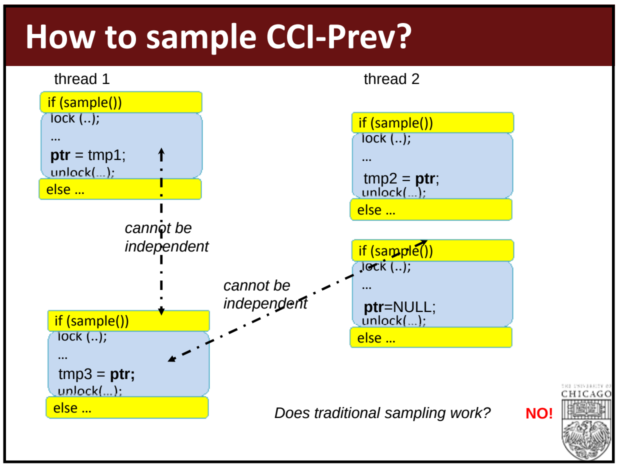
```
…
ptr = malloc (SIZE);
```

thread 2

```
…
free (ptr);
ptr=NULL;
…
```

```
…
if (!ptr) {
   ReportOutofMem();
   exit(1);
}
```

Does traditional sampling work?



How to sample CCI-Prev?

thread 1

```
if (sample())
  lock (..);
  …
  ptr = tmp1;
  unlock(…);
else …
```

```
if (sample())
  lock (..);
  …
  tmp3 = ptr;
  unlock(…);
else …
```

thread 2

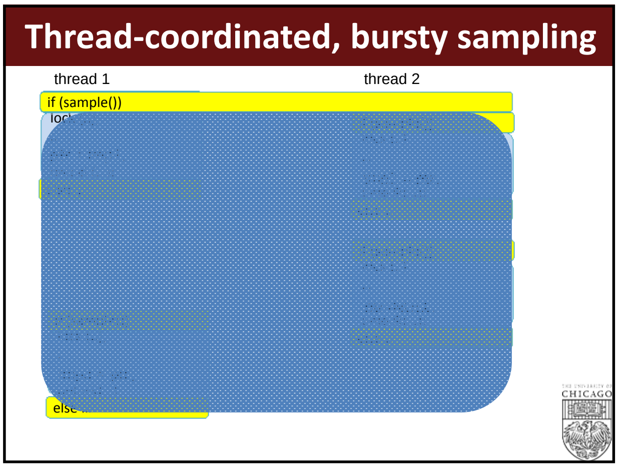
```
if (sample())
  lock (..);
  …
  tmp2 = ptr;
  unlock(…);
else …
```

```
if (sample())
  lock (..);
  …
  ptr=NULL;
  unlock(…);
else …
```

cannot be independent

cannot be independent

Does traditional sampling work? **NO!**

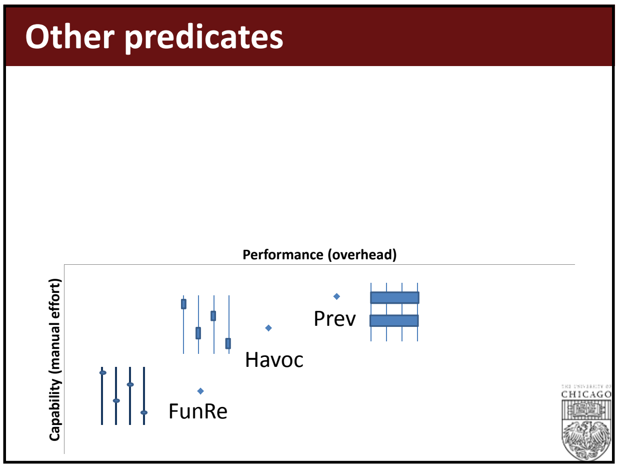


Thread-coordinated, bursty sampling

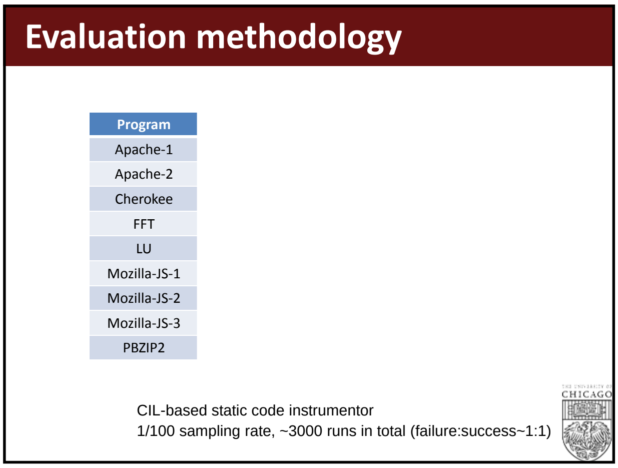
thread 1

thread 2

```
if (sample())
  loc
else
```



Other predicates

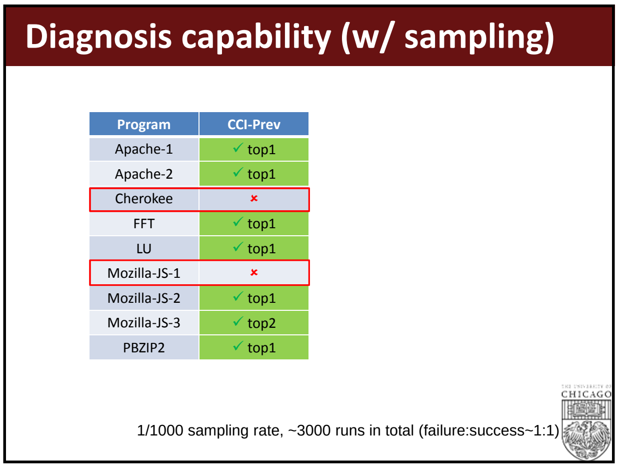


Evaluation methodology

Program
Apache-1
Apache-2
Cherokee
FFT
LU
Mozilla-JS-1
Mozilla-JS-2
Mozilla-JS-3
PBZIP2

CIL-based static code instrumentor

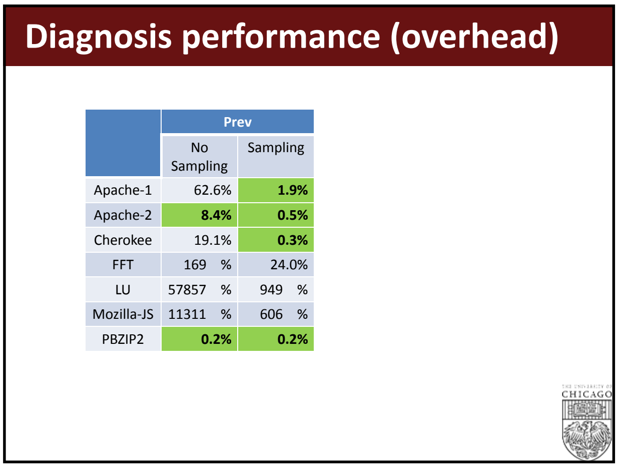
1/100 sampling rate, ~3000 runs in total (failure:success~1:1)



Diagnosis capability (w/ sampling)

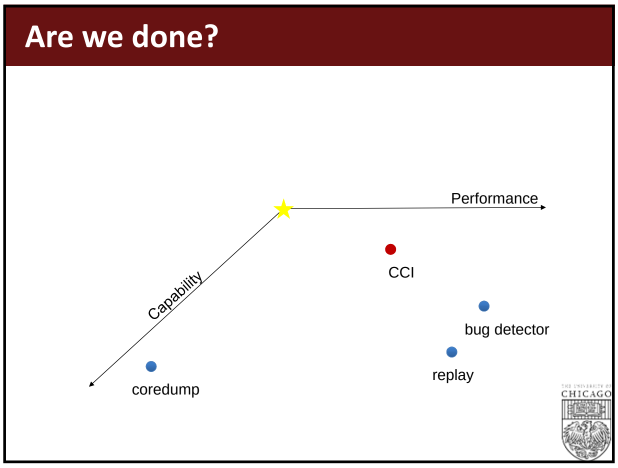
Program	CCI-Prev
Apache-1	✓ top1
Apache-2	✓ top1
Cherokee	×
FFT	✓ top1
LU	✓ top1
Mozilla-JS-1	×
Mozilla-JS-2	✓ top1
Mozilla-JS-3	✓ top2
PBZIP2	✓ top1

1/1000 sampling rate, ~3000 runs in total (failure:success~1:1)



Diagnosis performance (overhead)

	Prev	
	No Sampling	Sampling
Apache-1	62.6%	**1.9%**
Apache-2	**8.4%**	**0.5%**
Cherokee	19.1%	**0.3%**
FFT	169 %	24.0%
LU	57857 %	949 %
Mozilla-JS	11311 %	606 %
PBZIP2	**0.2%**	**0.2%**



Are we done?

Performance

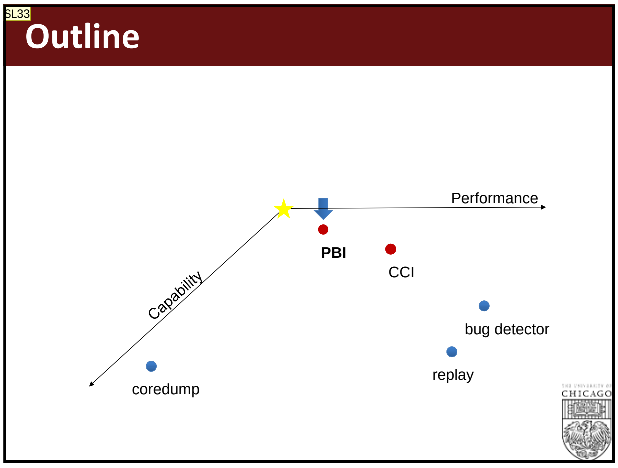
Capability

CCI

bug detector

replay

coredump



SL33

Outline

Performance

Capability

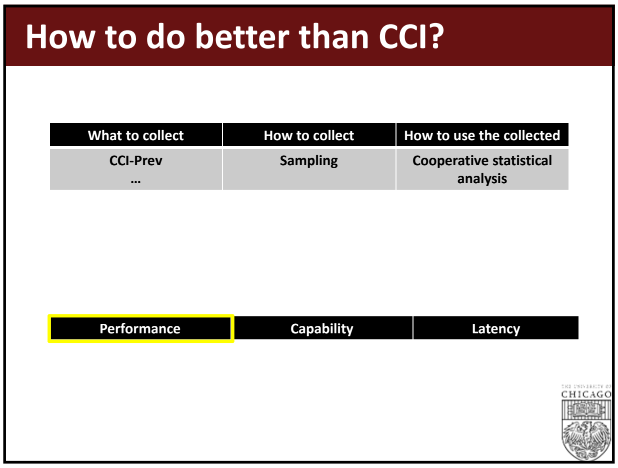
PBI

CCI

bug detector

replay

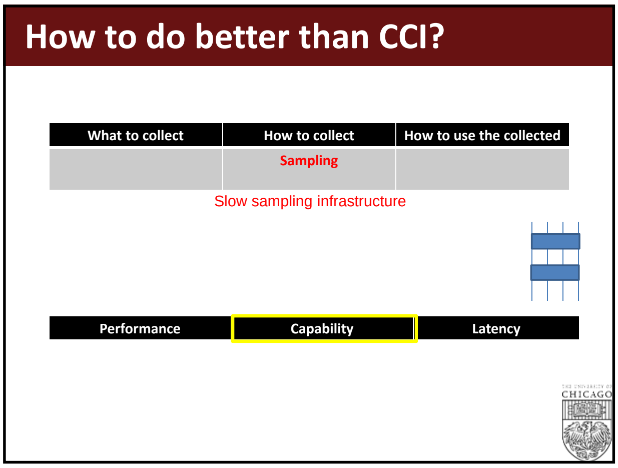
coredump



How to do better than CCI?

What to collect	How to collect	How to use the collected
CCI-Prev …	Sampling	Cooperative statistical analysis

Performance	Capability	Latency



How to do better than CCI?

What to collect	How to collect	How to use the collected
	Sampling	

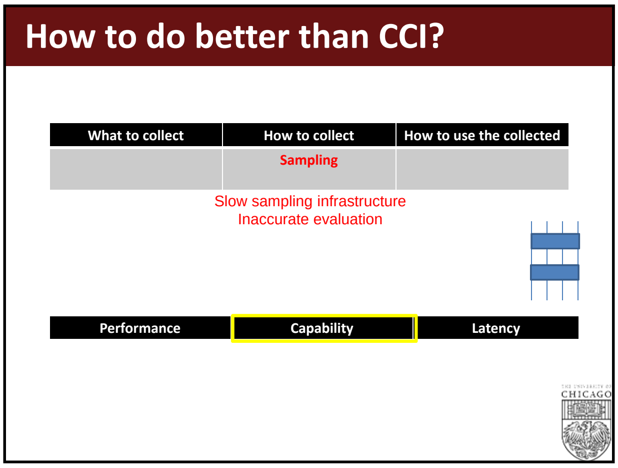
Slow sampling infrastructure

Performance	Capability	Latency

Slide 40

SL33 change the bullets texts. things like "compiler-based" is strange.

Shan Lu, 2014-1-15

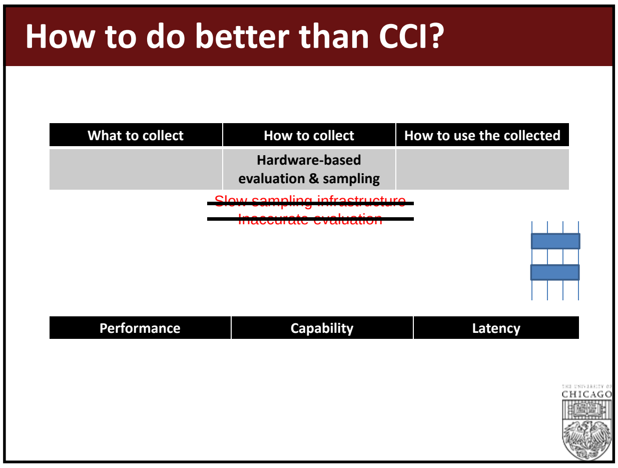


How to do better than CCI?

What to collect	How to collect	How to use the collected
	Sampling	

Slow sampling infrastructure
Inaccurate evaluation

Performance	Capability	Latency

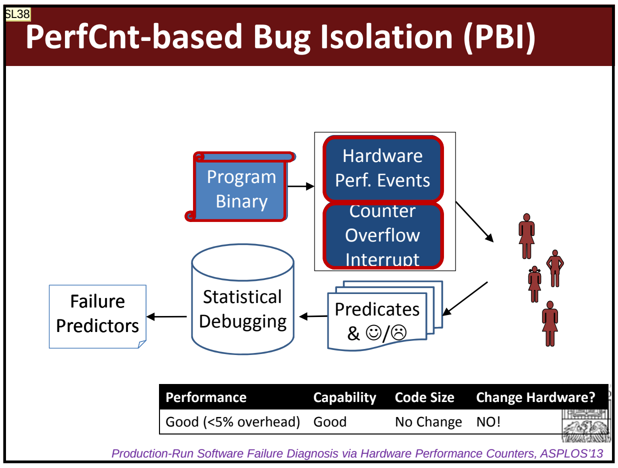


How to do better than CCI?

What to collect	How to collect	How to use the collected
	Hardware-based evaluation & sampling	

~~Slow sampling infrastructure~~
~~Inaccurate evaluation~~

Performance	Capability	Latency

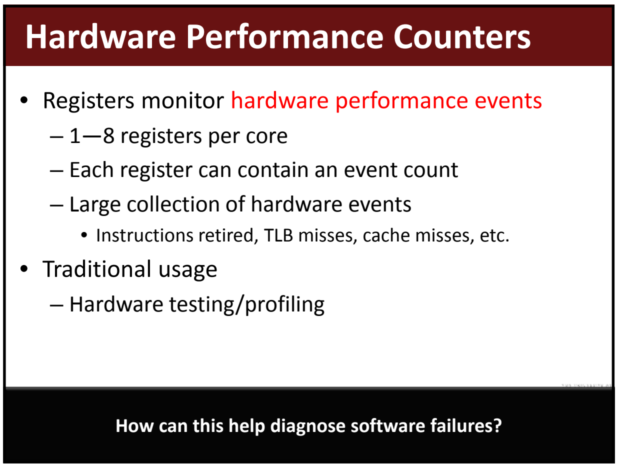


SL38

PerfCnt-based Bug Isolation (PBI)

Performance	Capability	Code Size	Change Hardware?
Good (<5% overhead)	Good	No Change	NO!

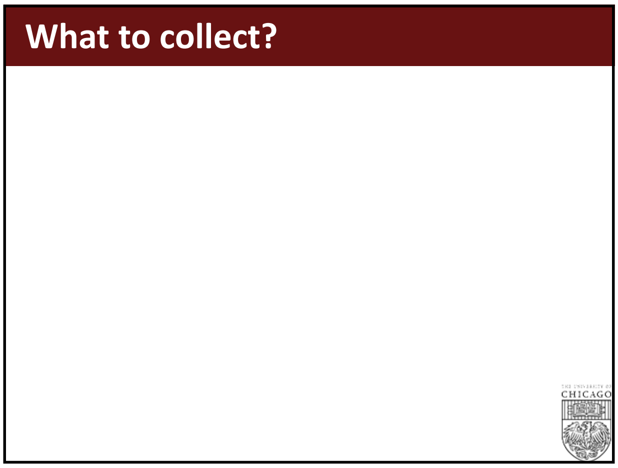
Production-Run Software Failure Diagnosis via Hardware Performance Counters, ASPLOS'13



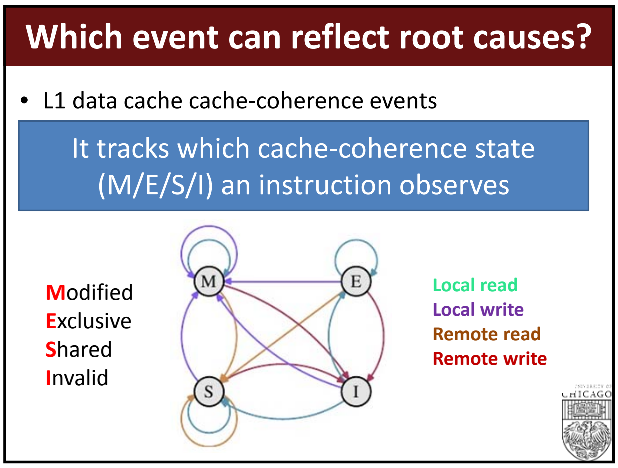
Hardware Performance Counters

- Registers monitor hardware performance events
  - 1—8 registers per core
  - Each register can contain an event count
  - Large collection of hardware events
    - Instructions retired, TLB misses, cache misses, etc.
- Traditional usage
  - Hardware testing/profiling

How can this help diagnose software failures?



What to collect?



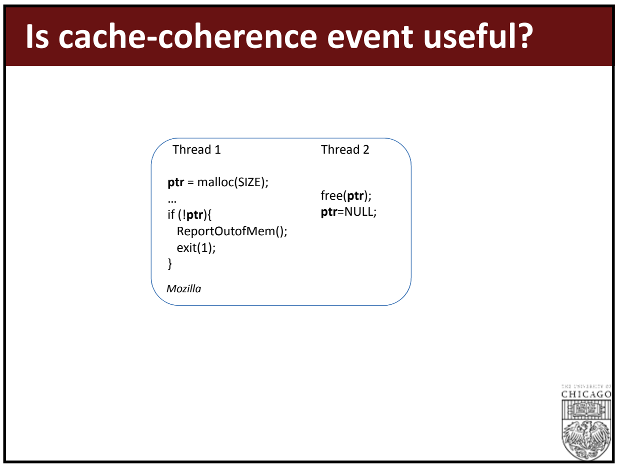
Which event can reflect root causes?

- L1 data cache cache-coherence events

It tracks which cache-coherence state (M/E/S/I) an instruction observes

Slide 45

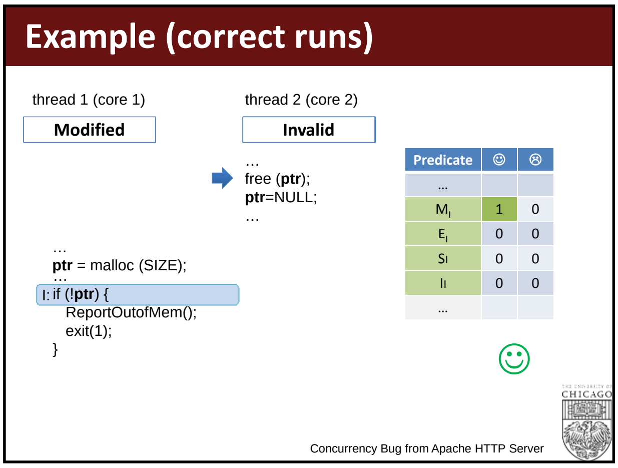
SL38 should i bring in secret sauce here?
Shan Lu, 2014-1-16



Is cache-coherence event useful?

```
Thread 1                      Thread 2

ptr = malloc(SIZE);
…                             free(ptr);
if (!ptr){                    ptr=NULL;
  ReportOutofMem();
  exit(1);
}
```

Mozilla



Example (correct runs)

thread 1 (core 1): **Modified**

thread 2 (core 2): **Invalid**

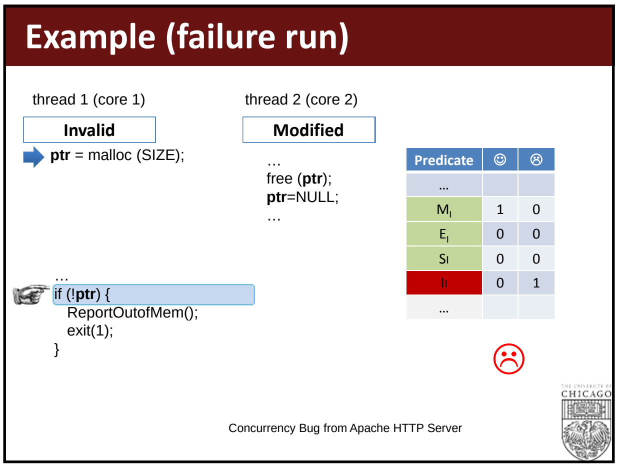
```
…
free (ptr);
ptr=NULL;
…
```

```
…
ptr = malloc (SIZE);
…
I: if (!ptr) {
    ReportOutofMem();
    exit(1);
}
```

Predicate	☺	☹
…		
M_I	1	0
E_I	0	0
S_I	0	0
I_I	0	0
…		

☺

Concurrency Bug from Apache HTTP Server



Example (failure run)

thread 1 (core 1): **Invalid**

thread 2 (core 2): **Modified**

```
ptr = malloc (SIZE);
```

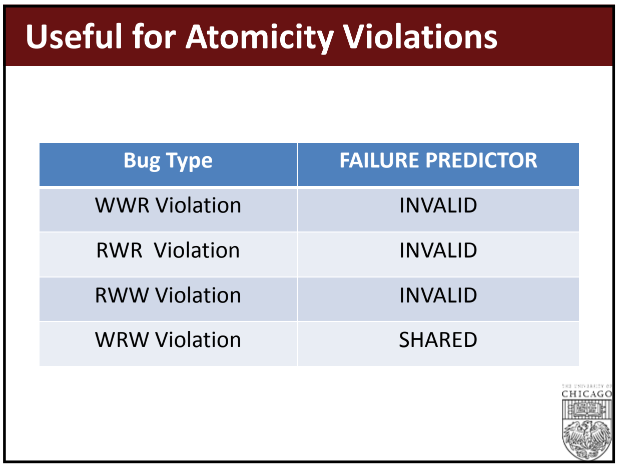
```
…
free (ptr);
ptr=NULL;
…
```

```
…
if (!ptr) {
    ReportOutofMem();
    exit(1);
}
```

Predicate	☺	☹
…		
M_I	1	0
E_I	0	0
S_I	0	0
I_I	0	1
…		

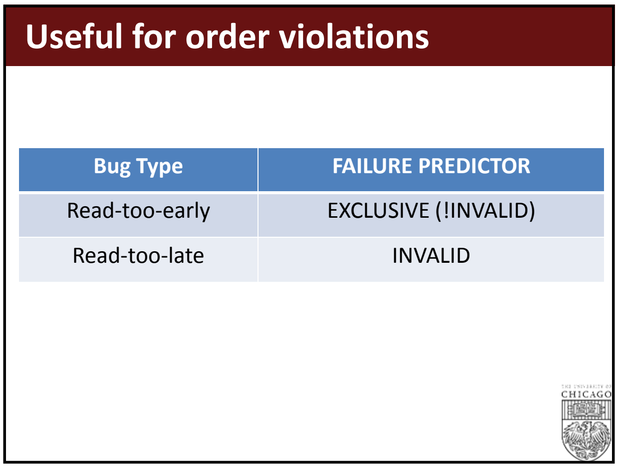
☹

Concurrency Bug from Apache HTTP Server



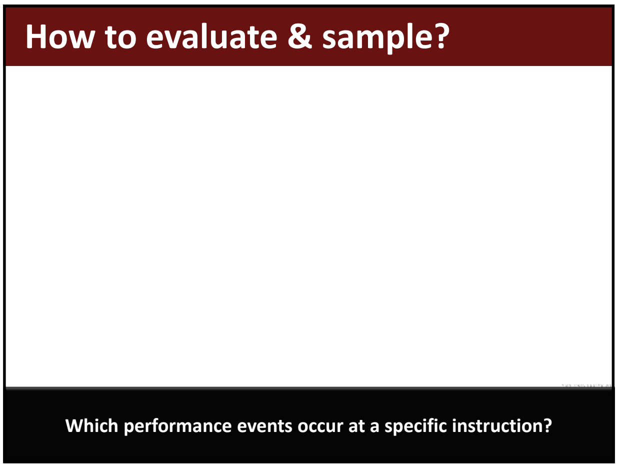
Useful for Atomicity Violations

Bug Type	FAILURE PREDICTOR
WWR Violation	INVALID
RWR Violation	INVALID
RWW Violation	INVALID
WRW Violation	SHARED



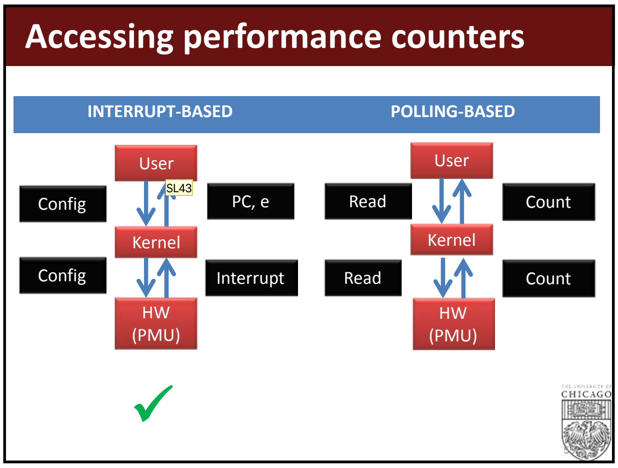
Useful for order violations

Bug Type	FAILURE PREDICTOR
Read-too-early	EXCLUSIVE (!INVALID)
Read-too-late	INVALID



How to evaluate & sample?

Which performance events occur at a specific instruction?



Accessing performance counters

INTERRUPT-BASED

POLLING-BASED

Config

User

SL43

PC, e

Kernel

Config

Interrupt

HW (PMU)

✓

Read

User

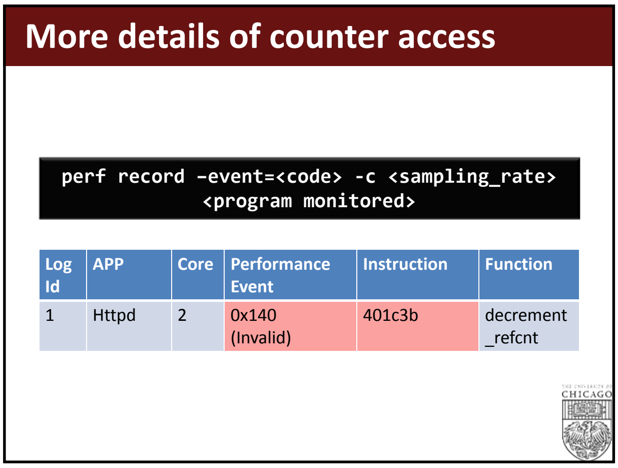
Count

Kernel

Read

Count

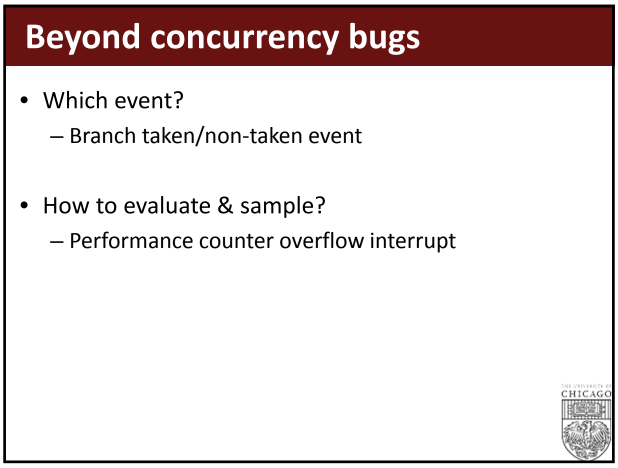
HW (PMU)



More details of counter access

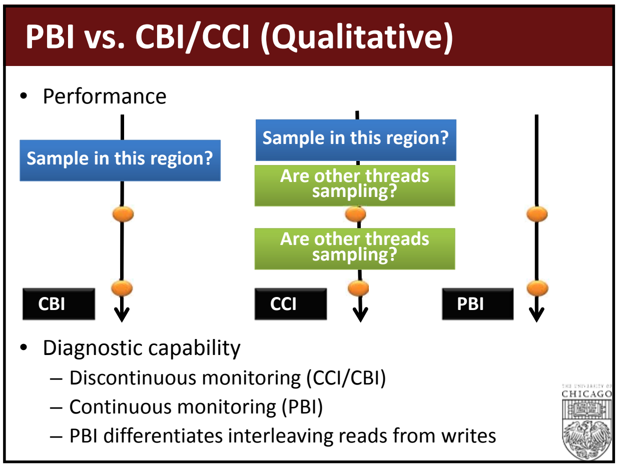
```
perf record -event=<code> -c <sampling_rate>
<program monitored>
```

Log Id	APP	Core	Performance Event	Instruction	Function
1	Httpd	2	0x140 (Invalid)	401c3b	decrement _refcnt



Beyond concurrency bugs

- Which event?
  - Branch taken/non-taken event
- How to evaluate & sample?
  - Performance counter overflow interrupt



PBI vs. CBI/CCI (Qualitative)

- Performance

Sample in this region?

CBI

Sample in this region?

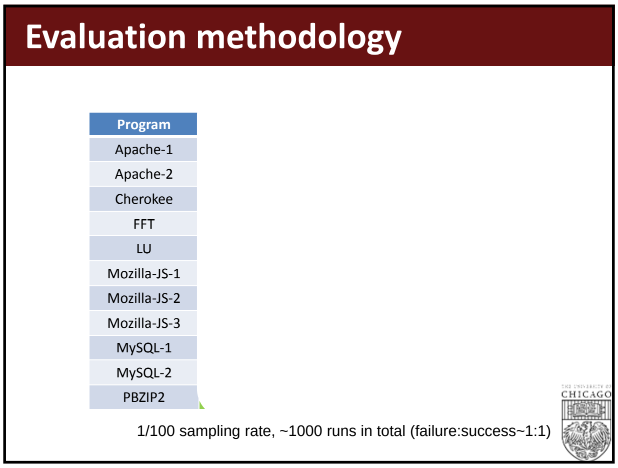
Are other threads sampling?

Are other threads sampling?

CCI

PBI

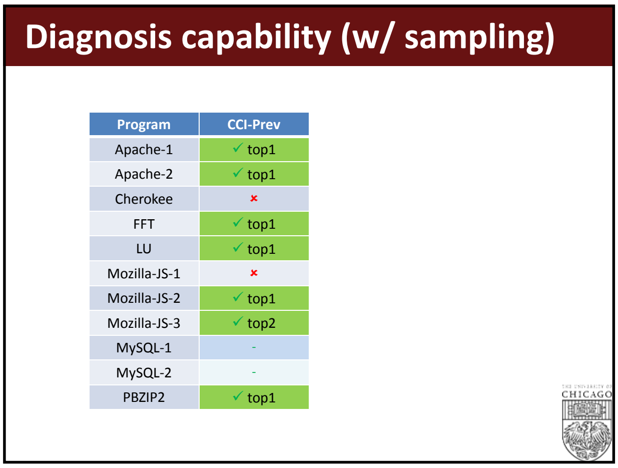
- Diagnostic capability
  - Discontinuous monitoring (CCI/CBI)
  - Continuous monitoring (PBI)
  - PBI differentiates interleaving reads from writes



Evaluation methodology

Program
Apache-1
Apache-2
Cherokee
FFT
LU
Mozilla-JS-1
Mozilla-JS-2
Mozilla-JS-3
MySQL-1
MySQL-2
PBZIP2

1/100 sampling rate, ~1000 runs in total (failure:success~1:1)



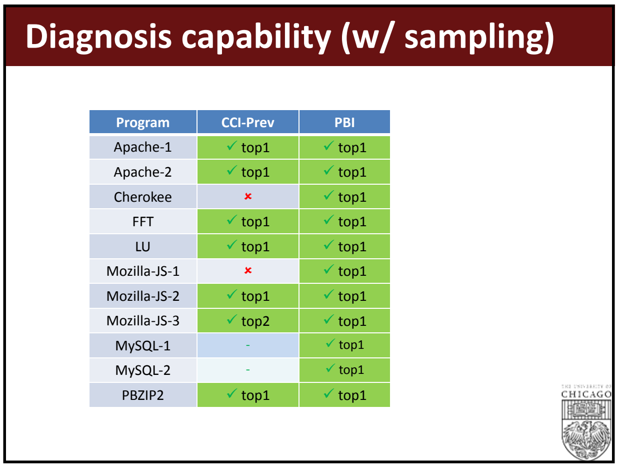
Diagnosis capability (w/ sampling)

Program	CCI-Prev
Apache-1	✓ top1
Apache-2	✓ top1
Cherokee	×
FFT	✓ top1
LU	✓ top1
Mozilla-JS-1	×
Mozilla-JS-2	✓ top1
Mozilla-JS-3	✓ top2
MySQL-1	-
MySQL-2	-
PBZIP2	✓ top1

Slide 55

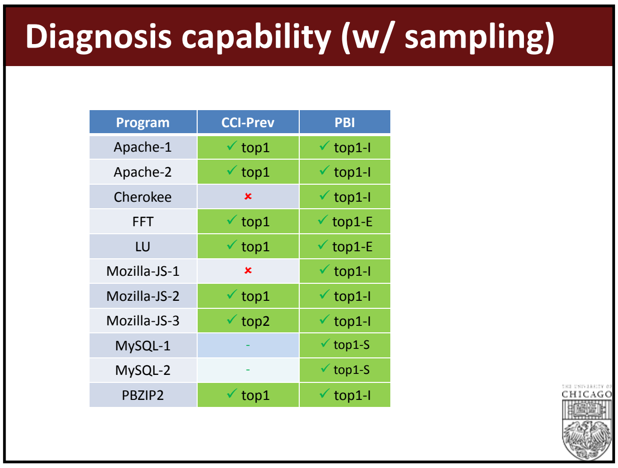
SL43 double check if polling needs to go through kernel

Shan Lu, 2014-1-16



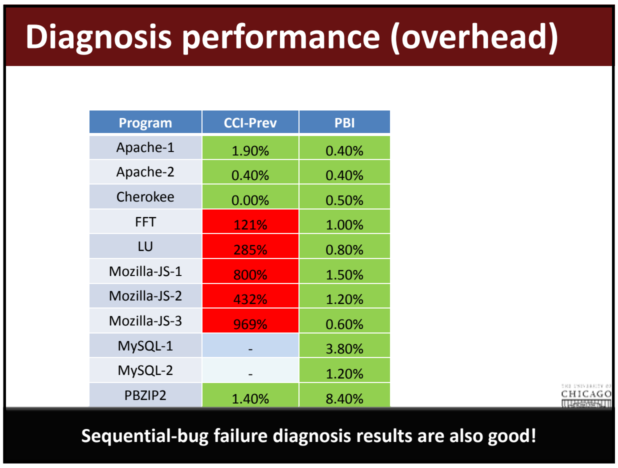
Diagnosis capability (w/ sampling)

Program	CCI-Prev	PBI
Apache-1	✓ top1	✓ top1
Apache-2	✓ top1	✓ top1
Cherokee	×	✓ top1
FFT	✓ top1	✓ top1
LU	✓ top1	✓ top1
Mozilla-JS-1	×	✓ top1
Mozilla-JS-2	✓ top1	✓ top1
Mozilla-JS-3	✓ top2	✓ top1
MySQL-1	-	✓ top1
MySQL-2	-	✓ top1
PBZIP2	✓ top1	✓ top1



Diagnosis capability (w/ sampling)

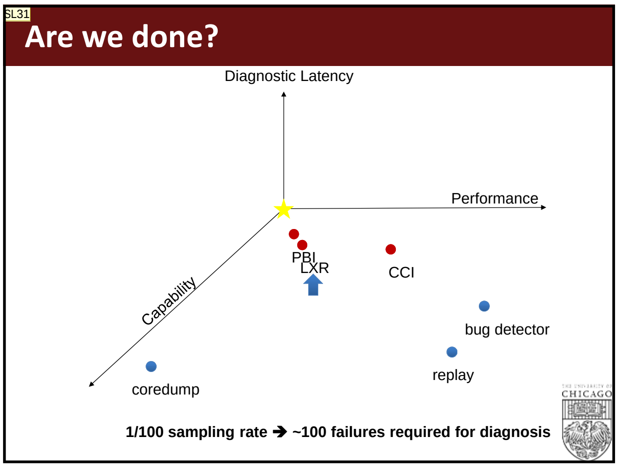
Program	CCI-Prev	PBI
Apache-1	✓ top1	✓ top1-I
Apache-2	✓ top1	✓ top1-I
Cherokee	×	✓ top1-I
FFT	✓ top1	✓ top1-E
LU	✓ top1	✓ top1-E
Mozilla-JS-1	×	✓ top1-I
Mozilla-JS-2	✓ top1	✓ top1-I
Mozilla-JS-3	✓ top2	✓ top1-I
MySQL-1	-	✓ top1-S
MySQL-2	-	✓ top1-S
PBZIP2	✓ top1	✓ top1-I



Diagnosis performance (overhead)

Program	CCI-Prev	PBI
Apache-1	1.90%	0.40%
Apache-2	0.40%	0.40%
Cherokee	0.00%	0.50%
FFT	121%	1.00%
LU	285%	0.80%
Mozilla-JS-1	800%	1.50%
Mozilla-JS-2	432%	1.20%
Mozilla-JS-3	969%	0.60%
MySQL-1	-	3.80%
MySQL-2	-	1.20%
PBZIP2	1.40%	8.40%

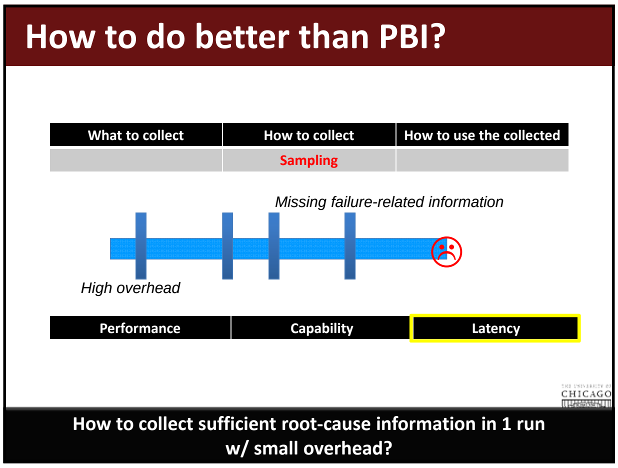
Sequential-bug failure diagnosis results are also good!



SL31

Are we done?

1/100 sampling rate ➔ ~100 failures required for diagnosis

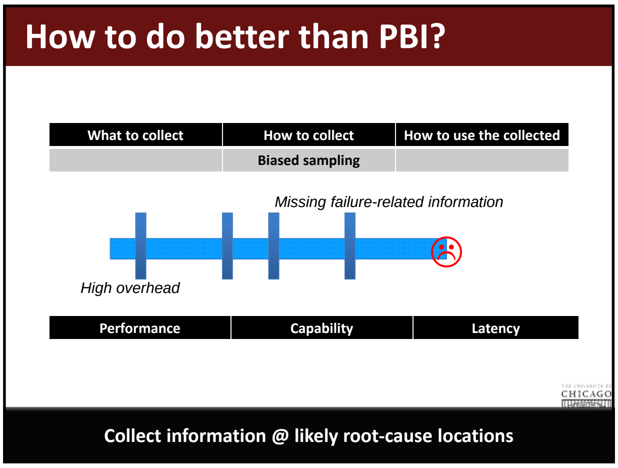


How to do better than PBI?

What to collect	How to collect	How to use the collected
	Sampling	

Performance	Capability	Latency

How to collect sufficient root-cause information in 1 run w/ small overhead?



How to do better than PBI?

What to collect	How to collect	How to use the collected
	Biased sampling	

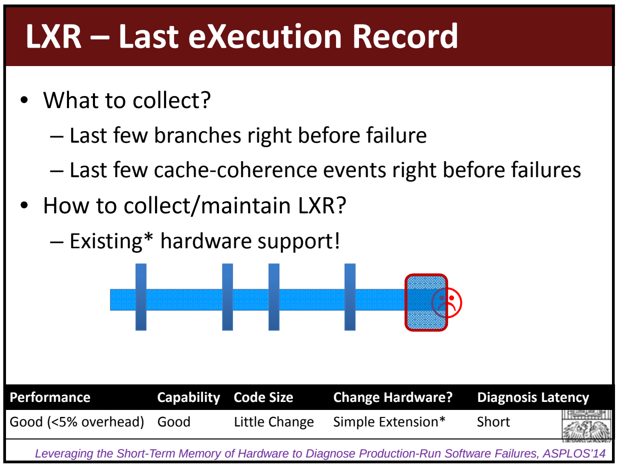
Performance	Capability	Latency

Collect information @ likely root-cause locations

Slide 64

SL31 maybe i should put 4-d/3-d coordinates here, and change the tables following

Shan Lu, 2014-1-15

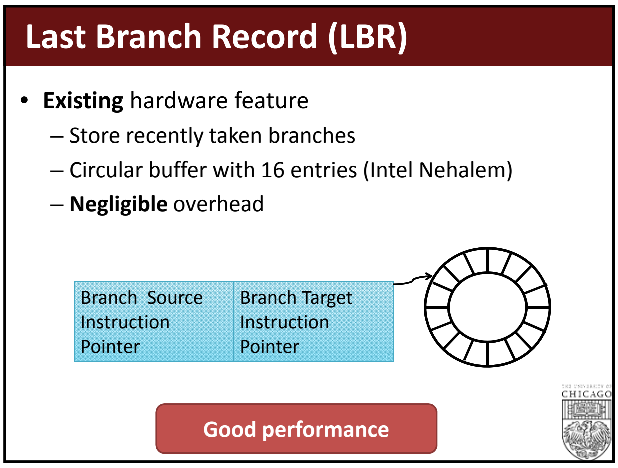


LXR – Last eXecution Record

- What to collect?
  - Last few branches right before failure
  - Last few cache-coherence events right before failures
- How to collect/maintain LXR?
  - Existing* hardware support!

Performance	Capability	Code Size	Change Hardware?	Diagnosis Latency
Good (<5% overhead)	Good	Little Change	Simple Extension*	Short

Leveraging the Short-Term Memory of Hardware to Diagnose Production-Run Software Failures, ASPLOS'14

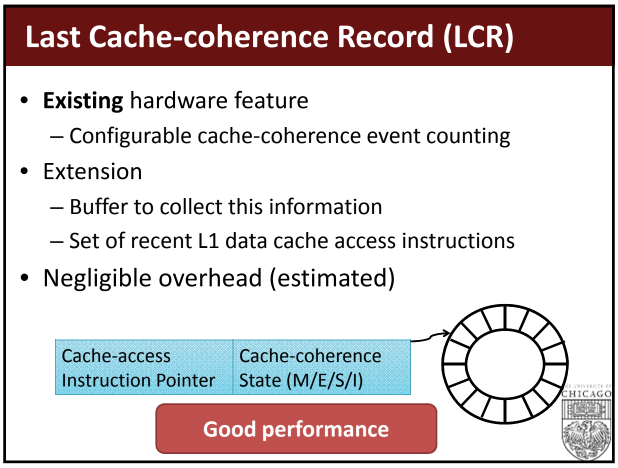


Last Branch Record (LBR)

- **Existing** hardware feature
  - Store recently taken branches
  - Circular buffer with 16 entries (Intel Nehalem)
  - **Negligible** overhead

Branch Source Instruction Pointer	Branch Target Instruction Pointer

Good performance

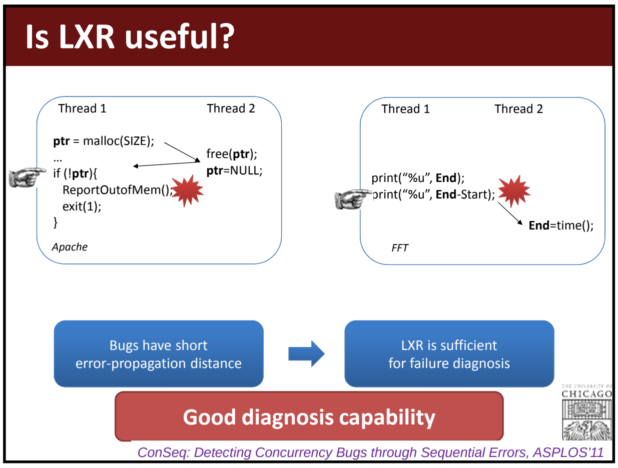


Last Cache-coherence Record (LCR)

- **Existing** hardware feature
  - Configurable cache-coherence event counting
- Extension
  - Buffer to collect this information
  - Set of recent L1 data cache access instructions
- Negligible overhead (estimated)

Cache-access Instruction Pointer	Cache-coherence State (M/E/S/I)

Good performance



Is LXR useful?

```
ptr = malloc(SIZE);
…
if (!ptr){
  ReportOutofMem();
  exit(1);
}
```

```
free(ptr);
ptr=NULL;
```

Apache

```
print("%u", End);
print("%u", End-Start);
```

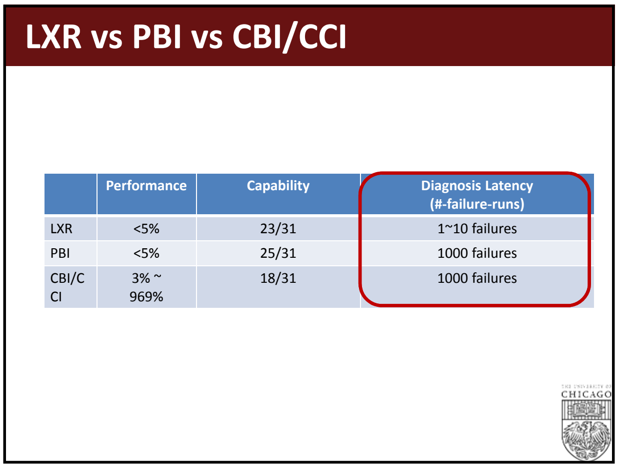
```
End=time();
```

FFT

Bugs have short error-propagation distance → LXR is sufficient for failure diagnosis

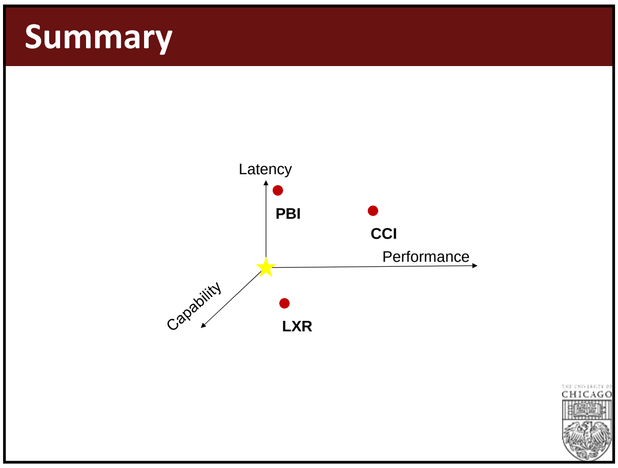
Good diagnosis capability

ConSeq: Detecting Concurrency Bugs through Sequential Errors, ASPLOS'11



LXR vs PBI vs CBI/CCI

	Performance	Capability	Diagnosis Latency (#-failure-runs)
LXR	<5%	23/31	1~10 failures
PBI	<5%	25/31	1000 failures
CBI/CCI	3% ~ 969%	18/31	1000 failures



Summary