InstantCheck: Checking the determinism of parallel programs using on-the-fly incremental hashing

Adrian Nistor, Darko Marinov, Josep Torrellas

MICRO '10
Problem: Nondeterminism
Problem: Nondeterminism

- Parallel programming is difficult
  - One input → many different outputs
Problem: Nondeterminism

- Parallel programming is difficult
  - One input \(\rightarrow\) many different outputs
- However, programmers like serial thinking
  - One input \(\rightarrow\) one output
Problem: Nondeterminism

- Parallel programming is difficult
  - One input $\rightarrow$ many different outputs
- However, programmers like serial thinking
  - One input $\rightarrow$ one output
- The problem with one input, many outputs
  - **Uncertain** what your own code does
Problem: Nondeterminism

- Parallel programming is difficult
  - One input $\rightarrow$ many different outputs
- However, programmers like serial thinking
  - One input $\rightarrow$ one output
- The problem with one input, many outputs
  - Uncertain what your own code does
- The problem: Nondeterminism
  - Due to parallel execution
  - Presume input is fixed (like in serial execution)
External determinism

Capturing state w/ Incremental Hashing

Hardware system

Software system

Other uses of hardware primitive

Evaluation

Conclusions
Our proposal:  External Determinism
Our proposal: External Determinism

- Determinism that we would like (for relevant algorithms)
  - Same input MEM state $\rightarrow$ same output MEM state
Our proposal: External Determinism

- Determinism that we would like (for relevant algorithms)
  - Same input MEM state → same output MEM state

- Determinism that we are currently getting
  - Fix order of inter-thread communications
Our proposal: External Determinism

- Determinism that we would like (for relevant algorithms)
  - Same input MEM state $\rightarrow$ same output MEM state
- External Determinism – our proposal
- Determinism that we are currently getting
  - Fix order of inter-thread communications
Our proposal: External Determinism

- Determinism that we would like (for relevant algorithms)
  - Same input MEM state $\rightarrow$ same output MEM state
- **External Determinism** — our proposal
- Determinism that we are currently getting
  - Fix order of inter-thread communications
  - **Internal Determinism** — some of the current proposals
Our proposal: External Determinism

- Determinism that we would like (for relevant algorithms)
  - Same input MEM state → same output MEM state
- **External Determinism** – our proposal
- Determinism that we are currently getting
  - Fix order of inter-thread communications
- **Internal Determinism** – some of the current proposals
- External determinism includes Internal determinism

Adrian Nistor

Checking determinism with on-the-fly incremental hashing
Our proposal:  External Determinism

- Determinism that we would like (for relevant algorithms)
  - Same input MEM state $\rightarrow$ same output MEM state
- **External Determinism** – our proposal

- Determinism that we are currently getting
  - Fix order of inter-thread communications
- **Internal Determinism** – some of the current proposals

- External determinism includes Internal determinism

  - If fix order of communication
    - You implicitly get same output memory state

  Internal

  External
Our proposal: External Determinism

- Determinism that we would like (for relevant algorithms)
  - Same input MEM state $\rightarrow$ same output MEM state
  - **External Determinism** – our proposal

- Determinism that we are currently getting
  - Fix order of inter-thread communications
  - **Internal Determinism** – some of the current proposals

- External determinism includes Internal determinism
  - If fix order of communication
    - You implicitly get same output memory state
  - But can get same output memory state
    - Without enforcing inter-thread communication order
Example
Example

CODE
Example
Example

CODE

Global  G
Local   L
Example

CODE

Global  \( G \)
Local   \( L \)

\[ G = G + L \]
Example

CODE

Global  \( G \)
Local   \( L \)

LOCK

\[ G = G + L \]

UN__LOCK
Example

CODE

Global  G
Local   L

EXEC—1

LOCK

G = G + L

UN_LOCK
Example

CODE

Global G
Local L

EXEC—1

EXEC—2

LOCK

G = G + L

UN_LOCK
## Example

<table>
<thead>
<tr>
<th>CODE</th>
<th>EXEC—1</th>
<th>EXEC—2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global</td>
<td>G</td>
<td>G == 2</td>
</tr>
<tr>
<td>Local</td>
<td>L</td>
<td></td>
</tr>
<tr>
<td>LOCK</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**CODE**

- **Global:** \(G\)
- **Local:** \(L\)
- **LOCK**

**EXEC—1**

- \(G == 2\)

**EXEC—2**

- \(G == 2\)
<table>
<thead>
<tr>
<th>CODE</th>
<th>EXEC—1</th>
<th>EXEC—2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global G</td>
<td>G == 2</td>
<td>G == 2</td>
</tr>
<tr>
<td>Local L</td>
<td>L_0 == 7</td>
<td>L_0 == 7</td>
</tr>
<tr>
<td>LOCK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>G = G + L</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UN_LOCK</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Example

Adrian Nistor  Checking determinism with on-the-fly incremental hashing
### Example

<table>
<thead>
<tr>
<th>CODE</th>
<th>EXEC—1</th>
<th>EXEC—2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global $G$</td>
<td>$G == 2$</td>
<td>$G == 2$</td>
</tr>
<tr>
<td>Local $L$</td>
<td>$L_0 == 7$  $L_1 == 3$</td>
<td>$L_0 == 7$  $L_1 == 3$</td>
</tr>
</tbody>
</table>

---

**CODE**

- Global $G$
- Local $L$

**LOCK**

**G = G + L**

**UN__LOCK**
Example

<table>
<thead>
<tr>
<th>CODE</th>
<th>EXEC—1</th>
<th>EXEC—2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global</td>
<td>G</td>
<td>G == 2</td>
</tr>
<tr>
<td>Local</td>
<td>L</td>
<td>L_0 == 7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>L_1 == 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>thread 0</td>
</tr>
<tr>
<td>LOCK</td>
<td></td>
<td>thread 0</td>
</tr>
</tbody>
</table>

G = G + L
### Example

#### CODE

<table>
<thead>
<tr>
<th>Global</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local</td>
<td>L</td>
</tr>
</tbody>
</table>

#### EXEC—1

<table>
<thead>
<tr>
<th>Thread 0</th>
<th>Thread 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>G = G + L</td>
<td>G = 2</td>
</tr>
<tr>
<td>L_0 = 7</td>
<td>L_1 = 3</td>
</tr>
</tbody>
</table>

#### EXEC—2

<table>
<thead>
<tr>
<th>Thread 0</th>
<th>Thread 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>G = G + L</td>
<td>G = 2</td>
</tr>
<tr>
<td>L_0 = 7</td>
<td>L_1 = 3</td>
</tr>
</tbody>
</table>

---

Checking determinism with on-the-fly incremental hashing
Example

CODE

Global  G
Local   L

LOCK

G = G + L

EXEC—1

G == 2

L_0 == 7

thread 0

EXEC—2

G == 2

L_0 == 7

thread 0

L_1 == 3

thread 1

G = 2 + 7

Adrian Nistor  Checking determinism with on-the-fly incremental hashing  3 / 27
### Example

<table>
<thead>
<tr>
<th>CODE</th>
<th>EXEC—1</th>
<th>EXEC—2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global G</td>
<td>G == 2</td>
<td>G == 2</td>
</tr>
<tr>
<td>Local L</td>
<td>L_0 == 7, L_1 == 3</td>
<td>L_0 == 7, L_1 == 3</td>
</tr>
<tr>
<td>thread 0, thread 1</td>
<td>thread 0, thread 1</td>
<td></td>
</tr>
<tr>
<td>G = G + L</td>
<td>G = 2 + 7</td>
<td>G = 9 + 3</td>
</tr>
</tbody>
</table>
Example

CODE

Global  G
Local   L

LOCK

G = G + L

UN_LOCK

EXEC—1

G == 2
L_0 == 7  L_1 == 3
thread 0  thread 1

G = 2 + 7

EXEC—2

G == 2
L_0 == 7  L_1 == 3
thread 0  thread 1

G = 9 + 3

G == 12
Example

CODE

<table>
<thead>
<tr>
<th>Global</th>
<th>G</th>
<th>Local</th>
<th>L</th>
</tr>
</thead>
</table>

LOCK

G = G + L

EXEC—1

<table>
<thead>
<tr>
<th>thread 0</th>
<th>thread 1</th>
</tr>
</thead>
</table>

| G = 2 + 7 |

EXEC—2

<table>
<thead>
<tr>
<th>thread 0</th>
<th>thread 1</th>
</tr>
</thead>
</table>

| G = 2 + 3 |

G = 9 + 3

G == 12
<table>
<thead>
<tr>
<th>CODE</th>
<th>EXEC—1</th>
<th>EXEC—2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>G == 2</td>
<td>G == 2</td>
</tr>
<tr>
<td></td>
<td>L_0 == 7</td>
<td>L_0 == 7</td>
</tr>
<tr>
<td></td>
<td>L_1 == 3</td>
<td>L_1 == 3</td>
</tr>
<tr>
<td></td>
<td>thread 0</td>
<td>thread 0</td>
</tr>
<tr>
<td></td>
<td>thread 1</td>
<td>thread 1</td>
</tr>
<tr>
<td></td>
<td>G = 2 + 7</td>
<td>G = 2 + 3</td>
</tr>
<tr>
<td></td>
<td>G = 9 + 3</td>
<td>G = 5 + 7</td>
</tr>
<tr>
<td></td>
<td>UN__LOCK</td>
<td></td>
</tr>
</tbody>
</table>
Example

<table>
<thead>
<tr>
<th>CODE</th>
<th>EXEC—1</th>
<th>EXEC—2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Global</td>
<td>G</td>
<td>G</td>
</tr>
<tr>
<td>Local</td>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td>LOCK</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>G + L</td>
<td>G + L</td>
</tr>
<tr>
<td>LOCK</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>G = 2 + 7</td>
<td>G = 2 + 3</td>
</tr>
<tr>
<td>thread 0</td>
<td></td>
<td>thread 0</td>
</tr>
<tr>
<td>thread 1</td>
<td></td>
<td>thread 1</td>
</tr>
<tr>
<td></td>
<td>G = 9 + 3</td>
<td>G = 5 + 7</td>
</tr>
<tr>
<td></td>
<td>G = 12</td>
<td>G = 12</td>
</tr>
</tbody>
</table>

Checking determinism with on-the-fly incremental hashing
Example

CODE

Global G
Local L

LOCK

G = G + L

UN__LOCK

EXEC—1

G == 2

L_0 == 7

thread 0

G = 2 + 7

G = 9 + 3

EXEC—2

G == 2

L_0 == 7

L_1 == 3

thread 0

thread 1

G = 2 + 3

G = 5 + 7

SAME STATE

G == 12

G == 12
Example

<table>
<thead>
<tr>
<th>CODE</th>
<th>EXEC—1</th>
<th>EXEC—2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global</td>
<td>G</td>
<td>G</td>
</tr>
<tr>
<td>Local</td>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td>LOCK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>G = G + L</td>
<td>G = 2 + 7</td>
<td>G = 2 + 3</td>
</tr>
<tr>
<td>UN_LOCK</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Execution order does not matter

SAME STATE
### Example

<table>
<thead>
<tr>
<th>CODE</th>
<th>EXEC—1</th>
<th>EXEC—2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global G</td>
<td>G == 2</td>
<td>G == 2</td>
</tr>
<tr>
<td>Local L</td>
<td>L_0 == 7 L_1 == 3</td>
<td>L_0 == 7 L_1 == 3</td>
</tr>
<tr>
<td>LOCK</td>
<td>thread 0 thread 1</td>
<td>thread 0 thread 1</td>
</tr>
<tr>
<td>G = G + L</td>
<td>G = 2 + 7</td>
<td>G = 2 + 3</td>
</tr>
<tr>
<td>UN_LOCK</td>
<td>G = 9 + 3</td>
<td>G = 5 + 7</td>
</tr>
</tbody>
</table>

**SAME STATE = EXTERNAL DET**

Execution order does not matter

---

Adrian Nistor  
Checking determinism with on-the-fly incremental hashing
### Example

<table>
<thead>
<tr>
<th>CODE</th>
<th>EXEC—1</th>
<th>EXEC—2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global G</td>
<td>G == 2</td>
<td>G == 2</td>
</tr>
<tr>
<td>Local L</td>
<td>L_0 == 7</td>
<td>L_0 == 7</td>
</tr>
<tr>
<td></td>
<td>L_1 == 3</td>
<td>L_1 == 3</td>
</tr>
<tr>
<td>LOCK</td>
<td>thread 0</td>
<td>thread 0</td>
</tr>
<tr>
<td></td>
<td>thread 1</td>
<td>thread 1</td>
</tr>
<tr>
<td>G = G + L</td>
<td>G = 2 + 7</td>
<td>G = 2 + 3</td>
</tr>
<tr>
<td></td>
<td>G = 9 + 3</td>
<td>G = 5 + 7</td>
</tr>
</tbody>
</table>

**SAME STATE** = **EXTERNAL DET**

Execution order does not matter = **NO internal det**
Internal Determinism
Internal Determinism

- Internal determinism is extremely intuitive
Internal Determinism

- Internal determinism is extremely intuitive
- But has costs
Internal Determinism

- Internal determinism is **extremely intuitive**
- But has costs
- Hardware cost
  - Special HW that commits transactions in order
Internal Determinism

- Internal determinism is **extremely intuitive**
- But has costs
- Hardware cost
  - Special HW that commits transactions in order
- Programmability cost
  - Disallow some synchronization (e.g., locks)
  - Deterministic languages
Internal Determinism

- Internal determinism is extremely intuitive
- But has costs
- Hardware cost
  - Special HW that commits transactions in order
- Programmability cost
  - Disallow some synchronization (e.g., locks)
  - Deterministic languages
- Performance cost
  - Software runtime fixes order – slows down execution
• Check external determinism
InstantCheck

- Check external determinism
- Fast checking of memory–state identity
  - On-the-fly incremental hashing
InstantCheck

- Check external determinism
- **Fast checking of memory–state identity**
  - On-the-fly incremental hashing
- Do the checking during code testing
InstantCheck

- Check external determinism
- **Fast checking of memory–state identity**
  - **On-the-fly incremental hashing**
- Do the checking during code testing
- Parallel code testing already runs the code many times
  - Piggyback on these many runs
  - Insert some very fast checks for external determinism
InstantCheck

- Check external determinism
- **Fast checking of memory–state identity**
  - On-the-fly incremental hashing
- Do the checking during code testing
- Parallel code testing already runs the code many times
  - Piggyback on these many runs
  - Insert some very fast checks for external determinism
- Check that memory states are identical
  - Program end, barriers, other points in the program
External determinism

Capture state w/ Incremental Hashing

Hardware system

Software system

Other uses of hardware primitive

Evaluation

Conclusions
How it works
How it works

- Have a hash of the Memory State
How it works

- Have a hash of the Memory State
- Each time memory is updated (via Store)
How it works

- Have a hash of the Memory State
- Each time memory is updated (via Store)
- Update the hash
  - To reflect the update to memory
How it works

- Have a hash of the Memory State
- Each time memory is updated (via Store)
- Update the hash
  - To reflect the update to memory
- At any time the hash already summarizes state
  - No need to compute it from scratch
  - No need to traverse the entire memory state
How it works

- Have a hash of the Memory State
- Each time memory is updated (via Store)
- Update the hash
  - To reflect the update to memory
- At any time the hash already summarizes state
  - No need to compute it from scratch
  - No need to traverse the entire memory state
- At any time, *Instantly ready* for identity checking
Example
Example

EXEC—1
Example

EXEC—1

G == 2
Example

EXEC—1

thread 0

G == 2
Example

EXEC—1

G == 2

thread 0    thread 1
Example

EXEC—1

Thread Hash

\[ \text{TH}_0 = \]

\[ G == 2 \]

thread 0    thread 1
Example

EXEC—1

Thread Hash

G == 2

thread 0

TH_0 = 0

thread 1
Example

EXEC—1

Thread Hash

thread 0  thread 1
TH_0 = 0    TH_1 = 0

G == 2
Example

EXEC—1

Thread Hash

\[ G = 2 \]

\[ TH_0 = 0 \quad TH_1 = 0 \]

\[ G = 2 + 7 \]
Example

EXEC—1

Thread Hash

<table>
<thead>
<tr>
<th>thread 0</th>
<th>thread 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>TH_0 = 0</td>
<td>TH_1 = 0</td>
</tr>
</tbody>
</table>

G == 2

G = 2 + 7

TH_0 =
Example

EXEC—1

Thread Hash

G == 2

thread 0
TH_0 = 0

thread 1
TH_1 = 0

G = 2 + 7

TH_0 = TH_0
Example

**EXEC—1**

- Thread Hash
- Thread 0: \(\text{TH}_0 = 0\)
- Thread 1: \(\text{TH}_1 = 0\)
- **G == 2**
- **G = 2 + 7**
- \(\text{TH}_0 = \text{TH}_0 \oplus 2\)
- MINUS old
Example

**EXEC—1**

Thread Hash

thread 0  thread 1

\[ TH_0 = 0 \quad TH_1 = 0 \]

\[ G = 2 + 7 \]

\[ TH_0 = TH_0 \oplus 2 \oplus 9 \]

\[ G = 2 \]

MINUS old

PLUS new

Adrian Nistor  Checking determinism with on-the-fly incremental hashing
Example

EXEC—1

Thread Hash

G == 2

thread 0
TH_0 = 0

thread 1
TH_1 = 0

G = 2 + 7

TH_0 = TH_0 ⊕ 2

9

hash ( G , 2 )

MINUS old

PLUS new

Adrian Nistor Checking determinism with on-the-fly incremental hashing
Example

EXEC—1

Thread Hash

<table>
<thead>
<tr>
<th>Thread Hash</th>
<th>thread 0</th>
<th>thread 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>G == 2</td>
<td>TH_0 = 0</td>
<td>TH_1 = 0</td>
</tr>
</tbody>
</table>

G = 2 + 7

TH_0 = TH_0 ⊕ 2 + 9

G = 9 + 3

hash (G, 2)

MINUS old

PLUS new

Adrian Nistor

Checking determinism with on-the-fly incremental hashing
Example

EXEC—1

Thread Hash

G == 2

TH_0 = 0

TH_0 = TH_0 ⊕ 2

TH_0 = 0 ⊕ 2

TH_0 = 2

TH_0 = 2 ⊕ 9

TH_0 = 11

TH_1 = 0

G = 2 + 7

G = 9 + 3

TH_1 =
Example

EXEC—1

Thread Hash

G == 2

thread 0

TH_0 = 0

G = 2 + 7

TH_0 = TH_0 \oplus 2

TH_1 = TH_1

G = 9 + 3

TH_1 = TH_1

hash ( G , 2 )

MINUS old

PLUS new

Adrian Nistor  Checking determinism with on-the-fly incremental hashing
Example

EXEC—1

Thread Hash

G == 2
TH_0 = 0
TH_0 = TH_0 ⊕ 2 ⊕ 9

G = 2 + 7

G = 9 + 3
TH_1 = TH_1 ⊕ 9

hash ( G , 2 )

MINUS old
PLUS new
MINUS old

Adrian Nistor
Checking determinism with on-the-fly incremental hashing
EXEC—1

Thread Hash

thread 0

\[ TH_0 = 0 \]

\[ G = 2 + 7 \]

\[ TH_0 = TH_0 \oplus 2 \]

\[ TH_0 = 0 \oplus 2 \]

\[ TH_0 = 0 \oplus 9 \]

\[ TH_0 = 9 \]

thread 1

\[ TH_1 = 0 \]

\[ G = 9 + 3 \]

\[ TH_1 = TH_1 \oplus 9 \]

\[ TH_1 = 0 \oplus 9 \]

\[ TH_1 = 12 \]

\[ hash \ ( G , 2 ) \]

MINUS old

PLUS new

MINUS old

PLUS new

Example
Example

\[
G = 2
\]

Thread Hash

\[
\begin{align*}
G &= 2 + 7 \\
G &= 9 + 3
\end{align*}
\]

\[
TH_0 = TH_0 - 2 + 9
\]

\[
TH_1 = TH_1 - 9 + 12
\]

\[
TH_0 = 0
\]

\[
TH_1 = 0
\]

EXEC—1

hash ( \( G, 2 \) )

MINUS old

PLUS new

CANCEL each other

MINUS old

PLUS new

Adrian Nistor

Checking determinism with on-the-fly incremental hashing
Example

EXEC—1

G == 2

thread 0
TH_0 = 0

thread 1
TH_1 = 0

G = 2 + 7

TH_0 = TH_0 \ominus 2 + 9

TH_1 = TH_1 \ominus 9 + 12

G = 9 + 3

TF_0 = TF_0 \ominus 2 + 9

TH_1 = TH_1 \ominus 9 + 12

hash (G, 2)

MINUS old
PLUS new
CANCEL each other
MINUS old
PLUS new

Checking determinism with on-the-fly incremental hashing
EXEC—1

Thread Hash

G == 2

thread 0

TH_0 = 0

G = 2 + 7

TH_0 = TH_0 ⊕ +

2

9

State Hash

TH_1 = TH_1 ⊕ +

G = 9 + 3

TH_1 = 0

G = 9 + 3

SH =

hash ( G , 2 )

MINUS old

PLUS new

CANCEL each other

MINUS old

PLUS new

Adrian Nistor

Checking determinism with on-the-fly incremental hashing
Example

EXEC—1

Thread Hash

thread 0

\[ G = 2 \]

\[ TH_0 = 0 \]

thread 1

\[ G = 2 + 7 \]

\[ TH_1 = 0 \]

\[ G = 9 + 3 \]

\[ TH_0 = TH_0 \oplus 9 \]

\[ TH_1 = TH_1 \oplus 12 \]

\[ SH = \]

\[ hash ( G, 2 ) \]

MINUS old

PLUS new

CANCEL each other

MINUS old

PLUS new

combine THs

= 2 + 9

= 2 + 9

= 2 + 9

= 2 + 9

Adrian Nistor

Checking determinism with on-the-fly incremental hashing
Example

EXEC—1

thread 0

TH_0 = 0

thread 1

TH_1 = 0

G = 2 + 7

G = 2

G = 9 + 3

TH_0 = TH_0 ⊕ 2

TH_1 = TH_1 ⊕ 9

SH = TH_0 ⊕ TH_1

TH_0 = 0

TH_1 = 0

hash ( G , 2 )

CANCEL each other

MINUS old

PLUS new

MINUS old

PLUS new

Adrian Nistor

Checking determinism with on-the-fly incremental hashing
**EXEC—1**

- **Thread Hash**
  - thread 0: $TH_0 = 0$
  - thread 1: $TH_1 = 0$

- **G = 2**
  - $G = G_0 + 7$
  - $TH_0 = TH_0 + 9$
  - $TH_1 = TH_1 + 12$

- **State Hash**
  - $SH = TH_0 \oplus TH_1 = 2$

- **Hash (G, 2)**
  - MINUS old
  - PLUS new
  - CANCEL each other
  - MINUS Initial Val

---

**Example**

Adrian Nistor

Checking determinism with on-the-fly incremental hashing
Example

**EXEC—1**

- **G == 2**
- **G = 2 + 7**
- **TH_0 = TH_0 \oplus 2 + 9**
- **TH_1 = TH_1 \oplus 9 + 12**
- **SH = TH_0 \oplus TH_1 = \ominus 2 + 12**

Thread Hash:
- **thread 0**
  - **TH_0 = 0**
- **thread 1**
  - **TH_1 = 0**

State Hash:
- Combine THs

MINUS old
PLUS new
CANCEL each other
MINUS Initial Val
PLUS Final Val

**hash (G, 2)**
Example

EXEC—1

Thread Hash

\[ G = 2 \]

thread 0

\[ TH_0 = 0 \]

thread 1

\[ TH_1 = 0 \]

\[ G = 2 + 7 \]

\[ G = 9 + 3 \]

\[ TH_0 = TH_0 \oplus 2 \]

\[ TH_1 = TH_1 \oplus 9 \]

\[ TH_0 = 0 \]

\[ TH_1 = 0 \]

\[ SH = TH_0 + TH_1 = 2 + 12 \]

hash \( (G, 2) \)

MINUS old

PLUS new

CANCEL each other

MINUS Initial Val

PLUS Final Val

INDEPENDENT of intermediate values

Adrian Nistor  Checking determinism with on-the-fly incremental hashing
Overview
Overview

• The computed hash is:
  \[ \text{MINUS (Initial\_State)} \quad \text{PLUS (Final\_State)} \]
The computed hash is:

\[
\text{MINUS (Initial\_State)} \quad \text{PLUS (Final\_State)}
\]

Independent of the intermediate steps
Overview

- The computed hash is:
  \[ \text{MINUS (Initial\_State)} \ \text{PLUS (Final\_State)} \]
- Independent of the intermediate steps
- At each Write operation do:
  \[ \text{MINUS (Old)} \ \text{PLUS (New)} \]
Overview

- The computed hash is:
  \[ \text{MINUS (Initial\_State)} \text{ PLUS (Final\_State)} \]
- Independent of the intermediate steps
- At each Write operation do:
  \[ \text{MINUS (Old)} \text{ PLUS (New)} \]
- Cancels the intermediate values
Advantages of Incremental Hashing
Advantages of Incremental Hashing

- Highly efficient hardware implementation
Advantages of Incremental Hashing

- Highly efficient hardware implementation
  - Thread Hash = locally computed = HW
Advantages of Incremental Hashing

- Highly efficient hardware implementation
  - Thread Hash = locally computed = HW
  - Global Hash = global operation = SW
    - Very rare ~ 10 – 10,000 times
Advantages of Incremental Hashing

• Highly efficient hardware implementation
  • Thread Hash = locally computed = HW
  • Global Hash = global operation = SW
    – Very rare ~ 10 – 10,000 times

• Associate & Commutative
  • The +/- operations can be: in Parallel & Out of Order
  • Flexible implementation choices for HW module
Advantages of Incremental Hashing

- Highly efficient hardware implementation
  - Thread Hash = locally computed = HW
  - Global Hash = global operation = SW
    - Very rare ~ 10 – 10,000 times
- Associate & Commutative
  - The +/- operations can be: in Parallel & Out of Order
- Flexible implementation choices for HW module
- Can delete some variables:
  \[ \oplus \text{initial\_value} \ominus \text{final\_value} \]
External determinism
Capturing state w/ Incremental Hashing

Hardware system
Software system
Other uses of hardware primitive
Evaluation
Conclusions
Basic design
Basic design

Write Buffer
Basic design

<table>
<thead>
<tr>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write</td>
</tr>
<tr>
<td>Buffer</td>
</tr>
</tbody>
</table>

Checking determinism with on-the-fly incremental hashing
Basic design

<table>
<thead>
<tr>
<th>Write Buffer</th>
<th>Data</th>
<th>P__addr</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Basic design

Write Buffer

Data

P__addr

WR Data__new
Basic design

Checking determinism with on-the-fly incremental hashing

Adrian Nistor
Basic design

Write Buffer

Data

P_addr

WR Data__new

L1 cache

L1 cache controller

Checking determinism with on-the-fly incremental hashing
Basic design

Write Buffer

Data

P_addr

WR Data_new

L1 cache

Memory – State Hashing Module (MHM)

L1 cache controller
Basic design

Checking determinism with on-the-fly incremental hashing
Basic design

Checking determinism with on-the-fly incremental hashing
Basic design

Write Buffer

Data

P_addr

VPN

WR Data__new

L1 cache

Memory – State Hashing Module (MHM)

L1 cache controller
Basic design

Write Buffer

Data

P_addr

VPN

V_addr

WR Data__new

L1 cache

Memory – State Hashing Module (MHM)

L1 cache controller

Checking determinism with on-the-fly incremental hashing
Basic design

Checking determinism with on-the-fly incremental hashing
Basic design

Checking determinism with on-the-fly incremental hashing
Basic design

Checking determinism with on-the-fly incremental hashing
Basic design

Checking determinism with on-the-fly incremental hashing
Basic design

Checking determinism with on-the-fly incremental hashing
Basic design

Checking determinism with on-the-fly incremental hashing
Basic design

Checking determinism with on-the-fly incremental hashing
Basic design

Writing Buffer

Write

Data

P_addr

VPN

V_addr

WR

Data_new

RD

Data_old

L1

cache

controller

L1 cache

controller

Memory – State Hashing Module (MHM)

Hash

FP

Round

CNTR

Check for determinism with on-the-fly incremental hashing
Basic design

Checking determinism with on-the-fly incremental hashing
Basic design

Checking determinism with on-the-fly incremental hashing
Basic design

Checking determinism with on-the-fly incremental hashing
Flexible Implementation Choices
Flexible Implementation Choices

Commutative
Associative
Flexible Implementation Choices

Commutative
Associative
Flexible Implementation Choices

- Commutative
- Associative
- Out of Order
- In parallel

Checking determinism with on-the-fly incremental hashing
Flexible Implementation Choices

MHM

Commutative
Associative
Out of Order
In parallel

MHM
Flexible Implementation Choices

MHM

Data__old
Data__new
V__addr

Commutative
Associative
Out of Order
In parallel

Checking determinism with on-the-fly incremental hashing
Flexible Implementation Choices

CLUSTERS
- Compute
- Intermediate
- Results

MHM

Data_old
Data_new
V_addr

Commutative
Associative
Out of Order
In parallel

Adrian Nistor
Flexible Implementation Choices

CLUSTERS
Compute Intermediate Results

Data__old
Data__new
V__addr

MHM

Commutative
Associative
Out of Order
In parallel

Checking determinism with on-the-fly incremental hashing
Flexible Implementation Choices

CLUSTERS
Compute
Intermediate
Results

MHM

Data__old
Data__new
V__addr

Commutative
Associative
Out of Order
In parallel

Adrian Nistor
Checking determinism with on-the-fly incremental hashing
Flexible Implementation Choices

CLUSTERS
Compute Intermediate Results

Data_old
Data_new
V_addr

MHM

Commutative
Associative
Out of Order
In parallel

Checking determinism with on-the-fly incremental hashing
Flexible Implementation Choices

CLUSTERS
Compute Intermediate Results

MHM

Commutative
Associative
Out of Order
In parallel

Checking determinism with on-the-fly incremental hashing
Flexible Implementation Choices

CLUSTERS
Compute Intermediate Results

merge / sum
The Intermediate Results

Commutative Associative
Out of Order In parallel

MHM

Data_old
Data_new
V_addr

Checking determinism with on-the-fly incremental hashing
Flexible Implementation Choices

CLUSTERS
Compute Intermediate Results

merge / sum
The Intermediate Results

Data_old
Data_new
V_addr

Commutative
Associative
Out of Order
In parallel

MHM

Checking determinism with on-the-fly incremental hashing
Flexible Implementation Choices

CLUSTERS
Compute Intermediate Results

merge / sum
The Intermediate Results

Data_old
Data_new
V_addr

64 bit Thread Hash register

Commutative Associative
Out of Order In parallel

MHM

Checking determinism with on-the-fly incremental hashing
Lightweight Hardware
Lightweight Hardware

✔️ Migrate
Lightweight Hardware

✔ Migrate
✔ Virtualize
Lightweight Hardware

✓ Migrate
✓ Virtualize
✓ Context-switch
Lightweight Hardware

- Migrate
- Virtualize
- Context-switch

\[
\begin{align*}
1 \text{ reg} & \quad = \quad \text{Save & Restore}
\end{align*}
\]
Lightweight Hardware

- ✓ Migrate
- ✓ Virtualize
- ✓ Context-switch
- ✓ Scalable

\[
\begin{align*}
1 \text{ reg} & = \text{ Save & Restore} \\
\text{Core – local operations}
\end{align*}
\]
Lightweight Hardware

✓ Migrate
✓ Virtualize
✓ Context-switch
✓ Scalable
✓ Flexible implementation choices

\[
\begin{align*}
1 \text{ reg} & = \text{ Save & Restore} \\
\text{Core – local operations}
\end{align*}
\]
External determinism
Capturing state w/ Incremental Hashing
Hardware system
Software system
Other uses of hardware primitive
Evaluation
Conclusions
Hashing in Software

- Can hash in SW
  - No special HW
  - Slower, but still reasonable
Hashing in Software

- Can hash in SW
  - No special HW
  - Slower, but still reasonable
- Two software only implementations
Hashing in Software

- Can hash in SW
  - No special HW
  - Slower, but still reasonable
- Two software only implementations
- Incremental hashing – SW-InstantCheck-Inc
  - Atomically get: Data_Old, Data_New
Hashing in Software

- Can hash in SW
  - No special HW
  - Slower, but still reasonable
- Two software only implementations
- Incremental hashing – SW-InstantCheck-Inc
  - Atomically get: Data_Old, Data_New
- Traversal base hashing – SW-InstantCheck-Tr
  - Need type information: float, double, other
Hashing in Software

• Can hash in SW
  • No special HW
  • Slower, but still reasonable
• Two software only implementations
• Incremental hashing – SW-InstantCheck-Inc
  • Atomically get: Data_Old, Data_New
• Traversal base hashing – SW-InstantCheck-Tr
  • Need type information: float, double, other
• I/O
  • Hash (e.g., CRC) the output stream
Sources of nondeterminism (1)
Sources of nondeterminism (1)

- We want to be aware and understand them
Sources of nondeterminism (1)

- We want to be aware and understand them
- Maybe even ignore some benign nondeterminism
Sources of nondeterminism (1)

- We want to be aware and understand them
- Maybe even ignore some benign nondeterminism
- Software bugs
Sources of nondeterminism (1)

- We want to be aware and understand them
- Maybe even ignore some benign nondeterminism
- Software bugs
  - **Important** source of nondeterminism
Sources of nondeterminism (1)

• We want to be aware and understand them
• Maybe even ignore some benign nondeterminism
• Software bugs
  • **Important** source of nondeterminism
  • But this is very positive:
    – Can detect software bugs
    – If they create nondeterminism
Sources of nondeterminism (1)

- We want to be aware and understand them
- Maybe even ignore some benign nondeterminism
- Software bugs
  - **Important** source of nondeterminism
  - But this is very positive:
    - Can detect software bugs
    - If they create nondeterminism
- Concurrency bugs create nondeterminism
Sources of nondeterminism (1)

- We want to be aware and understand them
- Maybe even ignore some benign nondeterminism
- Software bugs
  - *Important* source of nondeterminism
  - But this is very positive:
    - Can detect software bugs
    - If they create nondeterminism
- Concurrency bugs create nondeterminism
- In our experiments, we found real bug in PARSEC
Sources of nondeterminism (2)
Sources of nondeterminism (2)

- Floating point precision limitation
Sources of nondeterminism (2)

- Floating point precision limitation
  - Previous example, if we used FP:
    \[(G_0 + L_0) + L_1 \neq G_0 + (L_0 + L_1)\]
Sources of nondeterminism (2)

- Floating point precision limitation
  - Previous example, if we used FP:
    \[(G_0 + L_0) + L_1 \neq G_0 + (L_0 + L_1)\]
  - Maybe the user wants to ignore this (or not)
    - Different users may have different preferences
    - We offer several options
Sources of nondeterminism (2)

- Floating point precision limitation
  - Previous example, if we used FP:
    \[(G_0 + L_0) + L_1 \neq G_0 + (L_0 + L_1)\]
  - Maybe the user wants to ignore this (or not)
    - Different users may have different preferences
    - We offer several options
  - Default option: round to nearest 0.001
    - In our experiments (many FP) this was effective
Sources of nondeterminism (3)
Sources of nondeterminism (3)

- Small auxiliary data structures
Sources of nondeterminism (3)

- Small auxiliary data structures
  - e.g., 1 integer out of 10 MB of state
    - Reasonable to say that it is almost deterministic
Sources of nondeterminism (3)

- Small auxiliary data structures
  - e.g., 1 integer out of 10 MB of state
    - Reasonable to say that it is almost deterministic
  - Cholesky: list free “task nodes”
    - Truly nondeterministic
      - Different size, different values, different order
    - But just one structure -> ignore it
Sources of nondeterminism (3)

- Small auxiliary data structures
  - e.g., 1 integer out of 10 MB of state
    - Reasonable to say that it is almost deterministic
  - Cholesky: list free “task nodes”
    - Truly nondeterministic
      - Different size, different values, different order
      - But just one structure  ->  ignore it
  - PBZip2: dangling pointers to nondeterministic memory
    - The nondeterministic memory is deallocated
    - But the dangling pointers remain part of the state
Sources of nondeterminism (3)

- Small auxiliary data structures
  - e.g., 1 integer out of 10 MB of state
    - Reasonable to say that it is almost deterministic
  - Cholesky: list free “task nodes”
    - Truly nondeterministic
      - Different size, different values, different order
    - But just one structure -> ignore it
- PBZip2: dangling pointers to nondeterministic memory
  - The nondeterministic memory is deallocated
  - But the dangling pointers remain part of the state
- Solution: delete from hash (with the +/- technique)
Sources of nondeterminism (4)
Sources of nondeterminism (4)

• Truly nondeterministic algorithms
  • e.g.: Simulated Annealing in PARSEC
  • Expected to be nondeterministic
Sources of nondeterminism (4)

- Truly nondeterministic algorithms
  - e.g.: Simulated Annealing in PARSEC
  - Expected to be nondeterministic
- Monte-Carlo implementation in PARSEC
  - Maybe expect to be nondeterministic
Sources of nondeterminism (4)

- Truly nondeterministic algorithms
  - e.g.: Simulated Annealing in PARSEC
  - Expected to be nondeterministic
- Monte-Carlo implementation in PARSEC
  - Maybe expect to be nondeterministic
  - In fact, turns out this implementation is deterministic
    - From parallel execution point of view
    - Local random number generator (no shared state)
    - Number sequence: independent of other threads
Sources of nondeterminism (4)

• Truly nondeterministic algorithms
  • e.g.: Simulated Annealing in PARSEC
  • Expected to be nondeterministic
• Monte-Carlo implementation in PARSEC
  • Maybe expect to be nondeterministic
  • In fact, turns out this implementation is deterministic
    – From parallel execution point of view
    – **Local** random number generator (no shared state)
    – Number sequence: independent of other threads
• Input nondeterminism = like in serial program
External determinism
Capturing state w/ Incremental Hashing
Hardware system
Software system
Other uses of hardware primitive
Evaluation
Conclusions

Adrian Nistor  Checking determinism with on-the-fly incremental hashing
Instantly Compare Memory States (1)

Checking determinism with on-the-fly incremental hashing
Instantly Compare Memory States (1)

- Small hardware
Instantly Compare Memory States (1)

- Small hardware
- How can we use this?
Instantly Compare Memory States (1)

- Small hardware
- How can we use this?
- In techniques that use memory state comparison
Instantly Compare Memory States (1)

- Small hardware
- How can we use this?
- In techniques that use memory state comparison
- Detect bugs
Instantly Compare Memory States (1)

- Small hardware
- How can we use this?
- In techniques that use memory state comparison
- Detect bugs
  - Concurrency bugs create nondeterminism
Instantly Compare Memory States (1)

- Small hardware
- How can we use this?
- In techniques that use memory state comparison
- Detect bugs
  - Concurrency bugs create nondeterminism
  - Atom viol, order viol, data races, some semantic bugs
Instantly Compare Memory States (1)

- Small hardware
- How can we use this?
- In techniques that use memory state comparison
- Detect bugs
  - Concurrency bugs create nondeterminism
  - Atom viol, order viol, data races, some semantic bugs
  - Detect that a bug did occur = very fast
    - Keep bug detection always – on
Instantly Compare Memory States (1)

- Small hardware
- How can we use this?
- In techniques that use memory state comparison
- Detect bugs
  - Concurrency bugs create nondeterminism
  - Atom viol, order viol, data races, some semantic bugs
  - Detect that a bug did occur = very fast
    - Keep bug detection always – on
  - If bug occurred (hopefully rare case) => go pin–point
Instantly Compare Memory States (2)
**Instantly** Compare Memory States (2)

- Systematic testing
Instantly Compare Memory States (2)

- Systematic testing
  - CHESS, Microsoft Research
    - Found bugs that stress testing missed for *months*
Instantly Compare Memory States (2)

- Systematic testing
  - CHESS, Microsoft Research
    - Found bugs that stress testing missed for months
  - State comparison is very frequent operation
Instantly Compare Memory States (2)

- Systematic testing
  - CHESS, Microsoft Research
    - Found bugs that stress testing missed for months
  - State comparison is very frequent operation
  - Currently, very conservative
    - This results in unnecessary exploring some states
    - state == synchronization – order
    - previous example: same state, different sync – order
Instantly Compare Memory States (2)

- Systematic testing
  - CHESS, Microsoft Research
    - Found bugs that stress testing missed for months
  - State comparison is very frequent operation
  - Currently, very conservative
    - This results in unnecessary exploring some states
    - state == synchronization – order
    - previous example: same state, different sync – order
  - May miss states (coverage):
    - Sync – order the same,
    - but if races => not same state
Instantly Compare Memory States (3)
Instantly Compare Memory States (3)

- Filtering out benign data races
• Filtering out benign data races
  • You know about a race, and ask:
    – “is this race benign?”
Instanly Compare Memory States (3)

- Filtering out benign data races
  - You know about a race, and ask:
    - “is this race benign?”
  - If you flip that race
Instantly Compare Memory States (3)

- Filtering out benign data races
  - You know about a race, and ask:
    - “is this race benign?”
  - If you flip that race
  - But after some time you reach the same state
Filtering out benign data races

- You know about a race, and ask:
  - “is this race benign?”
- If you flip that race
- But after some time you reach the same state
- This means the race is benign
Instantly Compare Memory States (4)
Instantly Compare Memory States (4)

- Deterministic replay
Instantly Compare Memory States (4)

- Deterministic replay
  - Classical approach:
    - save a very precise execution log
    - Replay using that log
Instantly Compare Memory States (4)

- Deterministic replay
  - Classical approach:
    - save a very precise execution log
    - Replay using that log
  - Several recent proposals
    - Save partial log (e.g., just SYNCs)
    - At replay, search total log that generated the state
Instantly Compare Memory States (4)

- Deterministic replay
  - Classical approach:
    - save a very precise execution log
    - Replay using that log
  - Several recent proposals
    - Save partial log (e.g., just SYNCs)
    - At replay, search total log that generated the state
  - Can have hashes as part of the partial log
    - Guide the search at replay time
    - Detect replay failure
External determinism
Capturing state w/ Incremental Hashing
Hardware system
Software system
Other uses of hardware primitive

Evaluation
Conclusions
Setup
Setup

• Simulate the HW hashing module with PIN
Setup

- Simulate the HW hashing module with PIN
- 17 applications:
  - Sphinx3, PBZip2, PARSEC, SPLASH–2
Setup

- Simulate the HW hashing module with PIN
- 17 applications:
  - Sphinx3, PBZip2, PARSEC, SPLASH–2
- Run each application 30 times
Setup

- Simulate the HW hashing module with PIN
- 17 applications:
  - Sphinx3, PBZip2, PARSEC, SPLASH–2
- Run each application 30 times
- 8 Threads
Setup

- Simulate the HW hashing module with PIN
- 17 applications:
  - Sphinx3, PBZip2, PARSEC, SPLASH–2
- Run each application 30 times
- 8 Threads
- Compare memory state at:
  - Program end, Barriers, Loop iteration (for 2 apps)
Setup

- Simulate the HW hashing module with PIN
- 17 applications:
  - Sphinx3, PBZip2, PARSEC, SPLASH–2
- Run each application 30 times
- 8 Threads
- Compare memory state at:
  - Program end, Barriers, Loop iteration (for 2 apps)
- Randomizing thread scheduler
  - Like tools from Microsoft Research: PCT, CHESS
  - But simple scheduling policy: random
Determinism Characteristics
<table>
<thead>
<tr>
<th>Application</th>
<th>NDet</th>
<th>Run (b-b-b)</th>
</tr>
</thead>
<tbody>
<tr>
<td>blackscholes</td>
<td>101</td>
<td></td>
</tr>
<tr>
<td>fft</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>lu</td>
<td>68</td>
<td></td>
</tr>
<tr>
<td>radix</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>streamcluster</td>
<td>12928</td>
<td></td>
</tr>
<tr>
<td>swaptions</td>
<td>2501</td>
<td></td>
</tr>
<tr>
<td>volrend</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>fluidanimate</td>
<td>41</td>
<td></td>
</tr>
<tr>
<td>ocean</td>
<td>3871</td>
<td></td>
</tr>
<tr>
<td>waterNS</td>
<td>321</td>
<td></td>
</tr>
<tr>
<td>waterSP</td>
<td>221</td>
<td></td>
</tr>
<tr>
<td>cholesky</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>pbzip2</td>
<td>21</td>
<td></td>
</tr>
<tr>
<td>sphinx3</td>
<td>4265</td>
<td></td>
</tr>
<tr>
<td>barnes</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>canneal</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>radiosity</td>
<td>19</td>
<td></td>
</tr>
<tr>
<td>cholesky</td>
<td>64</td>
<td></td>
</tr>
<tr>
<td>barnes</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>canneal</td>
<td>64</td>
<td></td>
</tr>
<tr>
<td>radiosity</td>
<td>19</td>
<td></td>
</tr>
</tbody>
</table>

Determinism Characteristics

Checking determinism with on-the-fly incremental hashing
## Determinism Characteristics

<table>
<thead>
<tr>
<th>Det Type</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>blackscholes</td>
<td></td>
</tr>
<tr>
<td>fft</td>
<td></td>
</tr>
<tr>
<td>lu</td>
<td></td>
</tr>
<tr>
<td>radix</td>
<td></td>
</tr>
<tr>
<td>streamcluster</td>
<td></td>
</tr>
<tr>
<td>swaptions</td>
<td></td>
</tr>
<tr>
<td>volrend</td>
<td></td>
</tr>
<tr>
<td>fluidanimate</td>
<td></td>
</tr>
<tr>
<td>ocean</td>
<td></td>
</tr>
<tr>
<td>waterNS</td>
<td></td>
</tr>
<tr>
<td>waterSP</td>
<td></td>
</tr>
<tr>
<td>waterSP</td>
<td></td>
</tr>
<tr>
<td>waterSP</td>
<td></td>
</tr>
<tr>
<td>cholesky</td>
<td></td>
</tr>
<tr>
<td>pbzip2</td>
<td></td>
</tr>
<tr>
<td>sphinx3</td>
<td></td>
</tr>
<tr>
<td>barnes</td>
<td></td>
</tr>
<tr>
<td>canneal</td>
<td></td>
</tr>
<tr>
<td>radiosity</td>
<td></td>
</tr>
</tbody>
</table>

Checking determinism with on-the-fly incremental hashing
## Determinism Characteristics

<table>
<thead>
<tr>
<th>Det Type</th>
<th>Application</th>
<th>Run (b-b-b)</th>
<th>NDet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit</td>
<td>blackscholes</td>
<td>–</td>
<td>101</td>
</tr>
<tr>
<td>Bit</td>
<td>fft</td>
<td>–</td>
<td>13</td>
</tr>
<tr>
<td>Bit</td>
<td>lu</td>
<td>–</td>
<td>68</td>
</tr>
<tr>
<td>By</td>
<td>radix</td>
<td>–</td>
<td>12</td>
</tr>
<tr>
<td>Bit</td>
<td>streamcluster</td>
<td>–</td>
<td>12928</td>
</tr>
<tr>
<td>Bit</td>
<td>swaptions</td>
<td>–</td>
<td>2501</td>
</tr>
<tr>
<td>Bit</td>
<td>volrend</td>
<td>–</td>
<td>6</td>
</tr>
<tr>
<td>Bit</td>
<td>fluidanimate</td>
<td>–</td>
<td>2</td>
</tr>
<tr>
<td>Bit</td>
<td>ocean</td>
<td>–</td>
<td>3871</td>
</tr>
<tr>
<td>Bit</td>
<td>waterNS</td>
<td>–</td>
<td>321</td>
</tr>
<tr>
<td>Bit</td>
<td>waterSP</td>
<td>–</td>
<td>221</td>
</tr>
<tr>
<td>Bit</td>
<td>cholesky</td>
<td>–</td>
<td>4</td>
</tr>
<tr>
<td>Bit</td>
<td>pbzip2</td>
<td>–</td>
<td>2</td>
</tr>
<tr>
<td>Bit</td>
<td>sphinx3</td>
<td>–</td>
<td>4265</td>
</tr>
<tr>
<td>Bit</td>
<td>barnes</td>
<td>–</td>
<td>3</td>
</tr>
<tr>
<td>Bit</td>
<td>canneal</td>
<td>–</td>
<td>2</td>
</tr>
<tr>
<td>Bit</td>
<td>radiosity</td>
<td>–</td>
<td>16</td>
</tr>
<tr>
<td>Bit</td>
<td>canneal</td>
<td>–</td>
<td>0</td>
</tr>
<tr>
<td>Bit</td>
<td>cholesky</td>
<td>–</td>
<td>3</td>
</tr>
</tbody>
</table>

**Det Type**
- blackscholes
- fft
- lu
- radix
- streamcluster
- swaptions
- volrend
- fluidanimate
- ocean
- waterNS
- waterSP
- cholesky
- pbzip2
- sphinx3
- barnes
- canneal
- radiosity

Checking determinism with on-the-fly incremental hashing
## Determinism Characteristics

<table>
<thead>
<tr>
<th>Det Type</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>blackscholes</td>
</tr>
<tr>
<td></td>
<td>fft</td>
</tr>
<tr>
<td>Bit</td>
<td>lu</td>
</tr>
<tr>
<td>By</td>
<td>radix</td>
</tr>
<tr>
<td>Bit</td>
<td>streamcluster</td>
</tr>
<tr>
<td></td>
<td>swaptions</td>
</tr>
<tr>
<td></td>
<td>volrend</td>
</tr>
<tr>
<td></td>
<td>fluidanimate</td>
</tr>
<tr>
<td>FP</td>
<td>ocean</td>
</tr>
<tr>
<td>prec</td>
<td>waterNS</td>
</tr>
<tr>
<td></td>
<td>waterSP</td>
</tr>
<tr>
<td></td>
<td>cholesky</td>
</tr>
<tr>
<td></td>
<td>pbzip2</td>
</tr>
<tr>
<td></td>
<td>sphinx3</td>
</tr>
<tr>
<td></td>
<td>barnes</td>
</tr>
<tr>
<td></td>
<td>canneal</td>
</tr>
<tr>
<td></td>
<td>radiosity</td>
</tr>
</tbody>
</table>
## Determinism Characteristics

<table>
<thead>
<tr>
<th>Det Type</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>blackscholes</td>
<td>fft</td>
</tr>
<tr>
<td>Bit</td>
<td>lu</td>
</tr>
<tr>
<td>By</td>
<td>radix</td>
</tr>
<tr>
<td>Bit</td>
<td>streamcluster</td>
</tr>
<tr>
<td>swaptions</td>
<td>volrend</td>
</tr>
<tr>
<td>fluidanimate</td>
<td></td>
</tr>
<tr>
<td>ocean</td>
<td>waterNS</td>
</tr>
<tr>
<td>waterSP</td>
<td></td>
</tr>
<tr>
<td>cholesky</td>
<td></td>
</tr>
<tr>
<td>pbzip2</td>
<td></td>
</tr>
<tr>
<td>sphinx3</td>
<td></td>
</tr>
<tr>
<td>barnes</td>
<td></td>
</tr>
<tr>
<td>canneal</td>
<td></td>
</tr>
<tr>
<td>radiosity</td>
<td></td>
</tr>
</tbody>
</table>
### Determinism Characteristics

<table>
<thead>
<tr>
<th>Det Type</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>blackscholes</td>
<td>bl</td>
</tr>
<tr>
<td>fft</td>
<td>ft</td>
</tr>
<tr>
<td>Bit</td>
<td>lu</td>
</tr>
<tr>
<td>By</td>
<td>radix</td>
</tr>
<tr>
<td>Bit</td>
<td>streamcluster</td>
</tr>
<tr>
<td></td>
<td>swaptions</td>
</tr>
<tr>
<td></td>
<td>volrend</td>
</tr>
<tr>
<td>fluidanimate</td>
<td>fluidanimate</td>
</tr>
<tr>
<td>ocean</td>
<td>ocean</td>
</tr>
<tr>
<td>waterNS</td>
<td>waterNS</td>
</tr>
<tr>
<td>waterSP</td>
<td>waterSP</td>
</tr>
<tr>
<td>Ignore</td>
<td>cholesky</td>
</tr>
<tr>
<td>Small</td>
<td>pbzip2</td>
</tr>
<tr>
<td>Structs</td>
<td>sphinx3</td>
</tr>
<tr>
<td>barnes</td>
<td>barnes</td>
</tr>
<tr>
<td>canneal</td>
<td>canneal</td>
</tr>
<tr>
<td>radiosity</td>
<td>radiosity</td>
</tr>
</tbody>
</table>

Free list

- barnes
- canneal
- radiosity

Adrian Nistor

Checking determinism with on-the-fly incremental hashing
## Determinism Characteristics

<table>
<thead>
<tr>
<th>Det Type</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit</td>
<td>blackscholes</td>
</tr>
<tr>
<td>Bit</td>
<td>fft</td>
</tr>
<tr>
<td>By</td>
<td>lu</td>
</tr>
<tr>
<td>Bit</td>
<td>radix</td>
</tr>
<tr>
<td>Bit</td>
<td>streamcluster</td>
</tr>
<tr>
<td>Bit</td>
<td>swaptions</td>
</tr>
<tr>
<td>Bit</td>
<td>volrend</td>
</tr>
<tr>
<td>FP</td>
<td>fluidanimate</td>
</tr>
<tr>
<td>prec</td>
<td>ocean</td>
</tr>
<tr>
<td>prec</td>
<td>waterNS</td>
</tr>
<tr>
<td>prec</td>
<td>waterSP</td>
</tr>
<tr>
<td>Ignore</td>
<td>cholesky</td>
</tr>
<tr>
<td>Small</td>
<td>pbzip2</td>
</tr>
<tr>
<td>Structs</td>
<td>sphinx3</td>
</tr>
<tr>
<td>barnes</td>
<td></td>
</tr>
<tr>
<td>canneal</td>
<td></td>
</tr>
<tr>
<td>radiosity</td>
<td></td>
</tr>
</tbody>
</table>
## Determinism Characteristics

<table>
<thead>
<tr>
<th>Det Type</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>blackscholes</td>
</tr>
<tr>
<td></td>
<td>fft</td>
</tr>
<tr>
<td>Bit</td>
<td>lu</td>
</tr>
<tr>
<td>By</td>
<td>radix</td>
</tr>
<tr>
<td>Bit</td>
<td>streamcluster</td>
</tr>
<tr>
<td></td>
<td>swaptions</td>
</tr>
<tr>
<td></td>
<td>volrend</td>
</tr>
<tr>
<td></td>
<td>fluidanimate</td>
</tr>
<tr>
<td>FP</td>
<td>ocean</td>
</tr>
<tr>
<td>prec</td>
<td>waterNS</td>
</tr>
<tr>
<td></td>
<td>waterSP</td>
</tr>
<tr>
<td>Free list</td>
<td></td>
</tr>
<tr>
<td>Dangling ptr</td>
<td></td>
</tr>
<tr>
<td>4% mem</td>
<td></td>
</tr>
<tr>
<td>Ignore</td>
<td>cholesky</td>
</tr>
<tr>
<td>Small</td>
<td>pbzip2</td>
</tr>
<tr>
<td>Structs</td>
<td>sphinx3</td>
</tr>
<tr>
<td></td>
<td>barnes</td>
</tr>
<tr>
<td></td>
<td>canneal</td>
</tr>
<tr>
<td></td>
<td>radiosity</td>
</tr>
</tbody>
</table>
### Determinism Characteristics

<table>
<thead>
<tr>
<th>Det Type</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit</td>
<td>blackscholes, fft</td>
</tr>
<tr>
<td>By</td>
<td>lu</td>
</tr>
<tr>
<td>Bit</td>
<td>radix, streamcluster, swaptions, volrend</td>
</tr>
<tr>
<td>FP</td>
<td>fluidanimate</td>
</tr>
<tr>
<td>prec</td>
<td>ocean, waterNS, waterSP</td>
</tr>
<tr>
<td>Free list</td>
<td>cholesky</td>
</tr>
<tr>
<td>Dangling ptr</td>
<td>pbzip2</td>
</tr>
<tr>
<td>4% mem</td>
<td>sphinx3</td>
</tr>
<tr>
<td>NDet</td>
<td>barnes</td>
</tr>
<tr>
<td></td>
<td>canneal</td>
</tr>
<tr>
<td></td>
<td>radiosity</td>
</tr>
</tbody>
</table>
## Determinism Characteristics

<table>
<thead>
<tr>
<th>Det Type</th>
<th>Application</th>
<th>First Ndet Run (b-b-b)</th>
</tr>
</thead>
<tbody>
<tr>
<td>blackscholes</td>
<td>ft</td>
<td></td>
</tr>
<tr>
<td>fft</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit</td>
<td>lu</td>
<td></td>
</tr>
<tr>
<td>By</td>
<td>radix</td>
<td></td>
</tr>
<tr>
<td>Bit</td>
<td>streamcluster</td>
<td></td>
</tr>
<tr>
<td>Bit</td>
<td>swaptions</td>
<td></td>
</tr>
<tr>
<td>Bit</td>
<td>volrend</td>
<td></td>
</tr>
<tr>
<td>fluidanimate</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FP</td>
<td>ocean</td>
<td></td>
</tr>
<tr>
<td>prec</td>
<td>waterNS</td>
<td></td>
</tr>
<tr>
<td>prec</td>
<td>waterSP</td>
<td></td>
</tr>
<tr>
<td>Ignore</td>
<td>cholesky</td>
<td></td>
</tr>
<tr>
<td>Small</td>
<td>pbzip2</td>
<td></td>
</tr>
<tr>
<td>Small</td>
<td>sphinx3</td>
<td></td>
</tr>
<tr>
<td>Structs</td>
<td>barnes</td>
<td></td>
</tr>
<tr>
<td>NDet</td>
<td>canneal</td>
<td></td>
</tr>
<tr>
<td>NDet</td>
<td>radiosity</td>
<td></td>
</tr>
</tbody>
</table>
## Determinism Characteristics

<table>
<thead>
<tr>
<th>Det Type</th>
<th>Application</th>
<th>First Ndet Run (b-b-b)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>blackscholes</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>fft</td>
<td>–</td>
</tr>
<tr>
<td>Bit</td>
<td>lu</td>
<td>–</td>
</tr>
<tr>
<td>By</td>
<td>radix</td>
<td>–</td>
</tr>
<tr>
<td>Bit</td>
<td>streamcluster</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>swaptions</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>volrend</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>fluidanimate</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ocean</td>
<td></td>
</tr>
<tr>
<td>FP</td>
<td>waterNS</td>
<td></td>
</tr>
<tr>
<td>prec</td>
<td>waterSP</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Ignore</td>
<td></td>
</tr>
<tr>
<td></td>
<td>cholesky</td>
<td></td>
</tr>
<tr>
<td>Small</td>
<td>pbzip2</td>
<td></td>
</tr>
<tr>
<td>Structs</td>
<td>sphinx3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>barnes</td>
<td></td>
</tr>
<tr>
<td>NDet</td>
<td>canneal</td>
<td></td>
</tr>
<tr>
<td></td>
<td>radiosity</td>
<td></td>
</tr>
</tbody>
</table>

### Free list
- Dangling ptr: 4% mem
<table>
<thead>
<tr>
<th>Det Type</th>
<th>Application</th>
<th>First Ndet Run (b-b-b)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>blackscholes</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>fft</td>
<td>–</td>
</tr>
<tr>
<td>Bit</td>
<td>lu</td>
<td>–</td>
</tr>
<tr>
<td>By</td>
<td>radix</td>
<td>–</td>
</tr>
<tr>
<td>Bit</td>
<td>streamcluster</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>swaptions</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>volrend</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>fluidanimate</td>
<td>2</td>
</tr>
<tr>
<td>FP</td>
<td>ocean</td>
<td>3</td>
</tr>
<tr>
<td>prec</td>
<td>waterNS</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>waterSP</td>
<td>2</td>
</tr>
<tr>
<td>Ignore</td>
<td>cholesky</td>
<td>3</td>
</tr>
<tr>
<td>Small</td>
<td>pbzip2</td>
<td>2</td>
</tr>
<tr>
<td>Structs</td>
<td>sphinx3</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>barnes</td>
<td>2</td>
</tr>
<tr>
<td>NDet</td>
<td>canneal</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>radiosity</td>
<td>2</td>
</tr>
</tbody>
</table>

Free list Dangling ptr 4% mem
# Determinism Characteristics

<table>
<thead>
<tr>
<th>Det Type</th>
<th>Application</th>
<th>First Ndet Run (b-b-b)</th>
<th># Dyn check points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit</td>
<td>blackscholes</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>By</td>
<td>fft</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Bit</td>
<td>lu</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>By</td>
<td>radix</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Bit</td>
<td>streamcluster</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>swaptions</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>volrend</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>fluidanimate</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ocean</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>waterNS</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>waterSP</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>fluidanimate</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ocean</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>waterNS</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>waterSP</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Ignore</td>
<td>cholesky</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Small</td>
<td>pbzip2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Structs</td>
<td>sphinx3</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>barnes</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>NDet</td>
<td>canneal</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>radiosity</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

Determinism Characteristics

Adrian Nistor  Checking determinism with on-the-fly incremental hashing
# Determinism Characteristics

<table>
<thead>
<tr>
<th>Det Type</th>
<th>Application</th>
<th>First Ndet Run (b-b-b)</th>
<th># Dyn check points</th>
<th>NDet</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>blackscholes</td>
<td>–</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td></td>
<td>fft</td>
<td>–</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>Bit</td>
<td>lu</td>
<td>–</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>By</td>
<td>radix</td>
<td>–</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>Bit</td>
<td>streamcluster</td>
<td>–</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td></td>
<td>swaptions</td>
<td>–</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td></td>
<td>volrend</td>
<td>–</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td></td>
<td>fluidanimate</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FP</td>
<td>ocean</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>prec</td>
<td>waterNS</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>waterSP</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ignore</td>
<td>cholesky</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Small</td>
<td>pbzip2</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Structs</td>
<td>sphinx3</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>barnes</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NDet</td>
<td>canneal</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>radiosity</td>
<td>2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Free list
dangling ptr
4% mem

Checking determinism with on-the-fly incremental hashing
## Determinism Characteristics

<table>
<thead>
<tr>
<th>Det Type</th>
<th>Application</th>
<th>First Ndet Run (b-b-b)</th>
<th># Dyn check points</th>
<th>Det</th>
<th>NDet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit</td>
<td>blackscholes</td>
<td>–</td>
<td>101</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>fft</td>
<td>–</td>
<td>13</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>By</td>
<td>lu</td>
<td>–</td>
<td>68</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Bit</td>
<td>radix</td>
<td>–</td>
<td>12</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>streamcluster</td>
<td>–</td>
<td>12928</td>
<td>74</td>
<td></td>
</tr>
<tr>
<td></td>
<td>swaptions</td>
<td>–</td>
<td>2501</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>volrend</td>
<td>–</td>
<td>6</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>fluidanimate</td>
<td>2</td>
<td>41</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>FP</td>
<td>ocean</td>
<td>3</td>
<td>871</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>prec</td>
<td>waterNS</td>
<td>3</td>
<td>21</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>waterSP</td>
<td>2</td>
<td>21</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>cholesky</td>
<td>3</td>
<td>4</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Ignore</td>
<td>pbzip2</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Small</td>
<td>sphinx3</td>
<td>2</td>
<td>4265</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Structs</td>
<td>barnes</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>canneal</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>radiosity</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Free list**

- Dangling ptr 4% mem

- Ignore
- Small
- Structs

- NDet

**PARSEC BUG**
## Determinism Characteristics

<table>
<thead>
<tr>
<th>Det Type</th>
<th>Application</th>
<th>First Ndet Run (b-b-b)</th>
<th># Dyn check points Det</th>
<th>NDet</th>
</tr>
</thead>
<tbody>
<tr>
<td>blackscholes</td>
<td>–</td>
<td>101</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>fft</td>
<td>–</td>
<td>13</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>Bit</td>
<td>lu</td>
<td>–</td>
<td>68</td>
<td>0</td>
</tr>
<tr>
<td>By</td>
<td>radix</td>
<td>–</td>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td>Bit</td>
<td>streamcluster</td>
<td>–</td>
<td>12928</td>
<td>74</td>
</tr>
<tr>
<td></td>
<td>swaptions</td>
<td>–</td>
<td>2501</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>volrend</td>
<td>–</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>fluidanimate</td>
<td>2</td>
<td>41</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>FP</td>
<td>ocean</td>
<td>3</td>
<td>871</td>
<td>0</td>
</tr>
<tr>
<td>prec</td>
<td>waterNS</td>
<td>3</td>
<td>21</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>waterSP</td>
<td>2</td>
<td>21</td>
<td>0</td>
</tr>
<tr>
<td>Ignore</td>
<td>cholesky</td>
<td>3</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>Small</td>
<td>pbzip2</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Structs</td>
<td>sphinx3</td>
<td>2</td>
<td>4265</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>barnes</td>
<td>2</td>
<td>2</td>
<td>16</td>
</tr>
<tr>
<td>NDet</td>
<td>canneal</td>
<td>2</td>
<td>0</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td>radiosity</td>
<td>2</td>
<td>0</td>
<td>19</td>
</tr>
</tbody>
</table>
## Determinism Characteristics

<table>
<thead>
<tr>
<th>Det Type</th>
<th>Application</th>
<th>First Ndet Run (b-b-b)</th>
<th># Dyn check points</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Det</td>
</tr>
<tr>
<td>blackscholes</td>
<td>–</td>
<td>101</td>
<td>0</td>
</tr>
<tr>
<td>fft</td>
<td>–</td>
<td>13</td>
<td>0</td>
</tr>
<tr>
<td>Bit</td>
<td>lu</td>
<td>68</td>
<td>0</td>
</tr>
<tr>
<td>Bit</td>
<td>radix</td>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td>Bit</td>
<td>streamcluster</td>
<td>12928</td>
<td>74</td>
</tr>
<tr>
<td>Bit</td>
<td>swaptions</td>
<td>2501</td>
<td>0</td>
</tr>
<tr>
<td>Bit</td>
<td>volrend</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>fluidanimate</td>
<td>2</td>
<td>41</td>
</tr>
<tr>
<td></td>
<td>ocean</td>
<td>3</td>
<td>871</td>
</tr>
<tr>
<td>FP</td>
<td>waterNS</td>
<td>3</td>
<td>21</td>
</tr>
<tr>
<td>FP</td>
<td>waterSP</td>
<td>2</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>cholesky</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>pbzip2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>sphinx3</td>
<td>2</td>
<td>4265</td>
</tr>
<tr>
<td></td>
<td>barnes</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>NDet</td>
<td>canneal</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>NDet</td>
<td>radiosity</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

14 out of 17 apps are externally deterministic
## Determinism Characteristics

<table>
<thead>
<tr>
<th>Det Type</th>
<th>Application</th>
<th>First Ndet Run (b-b-b)</th>
<th># Dyn check points</th>
</tr>
</thead>
<tbody>
<tr>
<td>blackscholes</td>
<td>–</td>
<td>101</td>
<td>0</td>
</tr>
<tr>
<td>fft</td>
<td>–</td>
<td>13</td>
<td>0</td>
</tr>
<tr>
<td>Bit</td>
<td>lu</td>
<td>–</td>
<td>68</td>
</tr>
<tr>
<td>By</td>
<td>radix</td>
<td>–</td>
<td>12</td>
</tr>
<tr>
<td>Bit</td>
<td>streamcluster</td>
<td>–</td>
<td>12928</td>
</tr>
<tr>
<td></td>
<td>swaptions</td>
<td>–</td>
<td>2501</td>
</tr>
<tr>
<td></td>
<td>volrend</td>
<td>–</td>
<td>6</td>
</tr>
<tr>
<td>fluidanimate</td>
<td>2</td>
<td>41</td>
<td>0</td>
</tr>
<tr>
<td>FP</td>
<td>ocean</td>
<td>3</td>
<td>871</td>
</tr>
<tr>
<td>prec</td>
<td>waterNS</td>
<td>3</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>waterSP</td>
<td>2</td>
<td>21</td>
</tr>
<tr>
<td>Ignore</td>
<td>cholesky</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Small</td>
<td>pbzip2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Structs</td>
<td>sphinx3</td>
<td>2</td>
<td>4265</td>
</tr>
<tr>
<td>barnes</td>
<td>2</td>
<td>2</td>
<td>16</td>
</tr>
<tr>
<td>NDet</td>
<td>canneal</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>radiosity</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

14 out of 17 apps are externally deterministic
Even though they were not written for determinism (but high performance)
<table>
<thead>
<tr>
<th>Det Type</th>
<th>Application</th>
<th>First Ndet Run (b-b-b)</th>
<th># Dyn check points</th>
<th>Det</th>
<th>NDet</th>
</tr>
</thead>
<tbody>
<tr>
<td>blackscholes</td>
<td>–</td>
<td>101</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>fft</td>
<td>–</td>
<td>13</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit</td>
<td>lu</td>
<td>–</td>
<td>68</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>By</td>
<td>radix</td>
<td>–</td>
<td>12</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Bit</td>
<td>streamcluster</td>
<td>–</td>
<td>12928</td>
<td>74</td>
<td></td>
</tr>
<tr>
<td></td>
<td>swaptions</td>
<td>–</td>
<td>2501</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>volrend</td>
<td>–</td>
<td>6</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>fluidanimate</td>
<td>2</td>
<td>41</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FP</td>
<td>ocean</td>
<td>3</td>
<td>871</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>prec</td>
<td>waterNS</td>
<td>3</td>
<td>21</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>waterSP</td>
<td>2</td>
<td>21</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Ignore</td>
<td>cholesky</td>
<td>3</td>
<td>4</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Small</td>
<td>pbzip2</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Structs</td>
<td>sphinx3</td>
<td>2</td>
<td>4265</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>barnes</td>
<td>2</td>
<td>2</td>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NDet</td>
<td>canneal</td>
<td>2</td>
<td>0</td>
<td>64</td>
<td></td>
</tr>
<tr>
<td></td>
<td>radiosity</td>
<td>2</td>
<td>0</td>
<td>19</td>
<td></td>
</tr>
</tbody>
</table>

14 out of 17 apps are externally deterministic
Even though they were not written for determinism (but high performance)
Variety of thread communication, global variables, benign races, floating point
Distribution of Nondeterminism
Distribution of Nondeterminism

- Previous slide: find NonDet in 2 – 3 runs
Distribution of Nondeterminism

- Previous slide: find NonDet in 2 – 3 runs
- But maybe just luck: what is the distribution of NonDet?
Distribution of Nondeterminism

• Previous slide: find NonDet in 2 – 3 runs
• But maybe just luck: what is the distribution of NonDet?
  • Scattered – high changes of finding nondeterminism
Distribution of Nondeterminism

- Previous slide: find NonDet in 2 – 3 runs
- But maybe just luck: what is the distribution of NonDet?
  - Scattered – high changes of finding nondeterminism
  - Biased – low chances of finding nondeterminism
Distribution of Nondeterminism

- Previous slide: find NonDet in 2 – 3 runs
- But maybe just luck: what is the distribution of NonDet?
  - Scattered – high changes of finding nondeterminism
  - Biased – low chances of finding nondeterminism
Distribution of Nondeterminism

- Previous slide: find NonDet in 2 – 3 runs
- But maybe just luck: what is the distribution of NonDet?
  - Scattered – high changes of finding nondeterminism
  - Biased – low chances of finding nondeterminism

Previous slide: find NonDet in 2 – 3 runs

But maybe just luck: what is the distribution of NonDet?

- Scattered – high changes of finding nondeterminism
- Biased – low chances of finding nondeterminism
Distribution of Nondeterminism

- Previous slide: find NonDet in 2 – 3 runs
- But maybe just luck: what is the distribution of NonDet?
  - Scattered – high chances of finding nondeterminism
  - Biased – low chances of finding nondeterminism

![Bar chart showing distribution of nondeterminism]

- 30 runs
- State Blue – 14 runs
Distribution of Nondeterminism

- Previous slide: find NonDet in 2 – 3 runs
- But maybe just luck: what is the distribution of NonDet?
  - Scattered – high chances of finding nondeterminism
  - Biased – low chances of finding nondeterminism

- 30 runs
  - State Red – 11 runs
  - State Blue – 14 runs

Adrian Nistor
Checking determinism with on-the-fly incremental hashing
Distribution of Nondeterminism

- Previous slide: find NonDet in 2 – 3 runs
- But maybe just luck: what is the distribution of NonDet?
  - Scattered – high changes of finding nondeterminism
  - Biased – low chances of finding nondeterminism

<table>
<thead>
<tr>
<th>State</th>
<th>Runs</th>
</tr>
</thead>
<tbody>
<tr>
<td>State Yellow</td>
<td>3 runs</td>
</tr>
<tr>
<td>State Red</td>
<td>11 runs</td>
</tr>
<tr>
<td>State Blue</td>
<td>14 runs</td>
</tr>
</tbody>
</table>

30 runs
Distribution of Nondeterminism

- Previous slide: find NonDet in 2 – 3 runs
- But maybe just luck: what is the distribution of NonDet?
  - Scattered – high changes of finding nondeterminism
  - Biased – low chances of finding nondeterminism

![Bar chart showing distribution of nondeterminism states]

- State Green – 2 runs
- State Yellow – 3 runs
- State Red – 11 runs
- State Blue – 14 runs

30 runs
Distribution of Nondeterminism

- Previous slide: find NonDet in 2 – 3 runs
- But maybe just luck: what is the distribution of NonDet?
  - Scattered – high chances of finding nondeterminism
  - Biased – low chances of finding nondeterminism

- State Green – 2 runs
- State Yellow – 3 runs
- State Red – 11 runs
- State Blue – 14 runs

30 runs
Distribution of Nondeterminism

- Previous slide: find NonDet in 2 – 3 runs
- But maybe just luck: what is the distribution of NonDet?
  - Scattered – high chances of finding nondeterminism
  - Biased – low chances of finding nondeterminism
- Scattered => high chances of finding Nondeterminism

State Green – 2 runs
State Yellow – 3 runs
State Red – 11 runs
State Blue – 14 runs

30 runs
Overhead
Overhead

Checking determinism with on-the-fly incremental hashing
Overhead

Checking determinism with on-the-fly incremental hashing
Checking determinism with on-the-fly incremental hashing
Overhead

Native
Overhead

Checking determinism with on-the-fly incremental hashing
Overhead

SW-Incremental (Ideal)
Overhead

SW-Traversal (Ideal)
Overhead

Checking determinism with on-the-fly incremental hashing
Overhead

- HW: 0.3% -> always-on
Overhead

- HW: 0.3 % -> always-on
- Geometric Mean: SW-Incremental = 3X, SW-Traversial = 5X
• HW: 0.3% -> always-on

• Geometric Mean: SW-Incremental = 3X , SW-Traversal = 5X
  • Very reasonable if you don't have the HW support
Overhead

- HW: 0.3% -> always-on
- Geometric Mean: SW-Incremental = 3X, SW-Traversal = 5X
  - Very reasonable if you don't have the HW support
  - For some apps, one of SW-Incremental or SW-Traversal is clearly better
Overhead

- HW: 0.3 % -> always-on

- Geometric Mean: SW-Incremental = 3X , SW-Traversal = 5X
  - Very reasonable if you don't have the HW support
  - For some apps, one of SW-Incremental or SW-Traversal is clearly better
  - Should choose the appropriate one
External determinism
Capturing state w/ Incremental Hashing
Hardware system
Software system
Other uses of hardware primitive
Evaluation

Conclusions
Hardware Hashing
Hardware Hashing

- Powerful, versatile and lightweight primitive
Hardware Hashing

- Powerful, versatile and lightweight primitive
- This paper — execution state
Hardware Hashing

- Powerful, versatile and lightweight primitive
- This paper — execution state, write (addr, data)
Hardware Hashing

- Powerful, versatile and lightweight primitive
- This paper — execution state, write (addr, data)
  - 5 applications
Hardware Hashing

- Powerful, versatile and lightweight primitive
- This paper — execution state, write (addr, data)
  - 5 applications
  - Checking determinism, detecting software bugs
  - Systematic testing, benign races, deterministic replay
Hardware Hashing

- Powerful, versatile and lightweight primitive
- This paper — execution state, write (addr, data)
  - 5 applications
  - Checking determinism, detecting software bugs
  - Systematic testing, benign races, deterministic replay
- Our previous paper — execution history
Hardware Hashing

- Powerful, versatile and lightweight primitive
- This paper — execution state, write (addr, data)
  - 5 applications
  - Checking determinism, detecting software bugs
  - Systematic testing, benign races, deterministic replay
- Our previous paper — execution history, read (data)
Hardware Hashing

- Powerful, versatile and lightweight primitive
- This paper — execution state, write (addr, data)
  - 5 applications
  - Checking determinism, detecting software bugs
  - Systematic testing, benign races, deterministic replay
- Our previous paper — execution history, read (data)
  - Detect data races during systematic testing
Hardware Hashing

- Powerful, versatile and lightweight primitive
- This paper — execution state, write (addr, data)
  - 5 applications
  - Checking determinism, detecting software bugs
  - Systematic testing, benign races, deterministic replay
- Our previous paper — execution history, read (data)
  - Detect data races during systematic testing
- Bond & McKinley — execution context
Hardware Hashing

- Powerful, versatile and lightweight primitive
- This paper — **execution state**, write (addr, data)
  - 5 applications
  - Checking determinism, detecting software bugs
  - Systematic testing, benign races, deterministic replay
- Our previous paper — **execution history**, read (data)
  - Detect data races during systematic testing
- Bond & McKinley — **execution context**, RetAddr

Checking determinism with on-the-fly incremental hashing
Hardware Hashing

- Powerful, versatile and lightweight primitive

- This paper — **execution state**, write (addr, data)
  - 5 applications
  - Checking determinism, detecting software bugs
  - Systematic testing, benign races, deterministic replay

- Our previous paper — **execution history**, read (data)
  - Detect data races during systematic testing

- Bond & McKinley — **execution context**, RetAddr
  - Java serial programs
  - Several testing and debugging tasks
Hardware Hashing

- Powerful, versatile and lightweight primitive
- This paper — execution state, write (addr, data)
  - 5 applications
  - Checking determinism, detecting software bugs
  - Systematic testing, benign races, deterministic replay
- Our previous paper — execution history, read (data)
  - Detect data races during systematic testing
- Bond & McKinley — execution context, RetAddr
  - Java serial programs
  - Several testing and debugging tasks
- Other operations, other information, other applications?
Checking determinism with on-the-fly incremental hashing
InstantCheck

- Check external determinism with low overhead
InstantCheck

- Check external determinism with low overhead
- Lightweight hardware
  - *Instantly* compare memory state
InstantCheck

- Check external determinism with low overhead
- Lightweight hardware
  - **Instantly** compare memory state
- Incremental hashing
  - High – performance hardware implementation
  - Enables flexible implementation choices
InstantCheck

- Check external determinism with low overhead
- Lightweight hardware
  - *Instantly* compare memory state
- Incremental hashing
  - High – performance hardware implementation
  - Enables flexible implementation choices
- Present four other applications of HW primitive

Checking determinism with on-the-fly incremental hashing
InstantCheck

- Check external determinism with low overhead
- Lightweight hardware
  - *Instantly* compare memory state
- Incremental hashing
  - High – performance hardware implementation
  - Enables flexible implementation choices
- Present four other applications of HW primitive
- Some applications are externally deterministic
  - Not written for determinism (but high performance)
InstantCheck

- Check external determinism with low overhead
- Lightweight hardware
  - Instantly compare memory state
- Incremental hashing
  - High – performance hardware implementation
  - Enables flexible implementation choices
- Present four other applications of HW primitive
- Some applications are externally deterministic
  - Not written for determinism (but high performance)
- Helped us find a real bug in PARSEC
InstantCheck: Checking the determinism of parallel programs using on-the-fly incremental hashing

Adrian Nistor, Darko Marinov, Josep Torrellas

MICRO '10