APSEC 2014
THE 21ST ASIA-PACIFIC SOFTWARE ENGINEERING CONFERENCE
1ST-4TH DECEMBER 2014 / JEJU KOREA

Synchronization Error Detection of MPI Programs by Symbolic Execution

Xianjin Fu, Zhenbang Chen, Chun Huang, Wei Dong, and Ji Wang
zbchen@nudt.edu.cn

College of Computer
National University of Defense Technology
Changsha, China

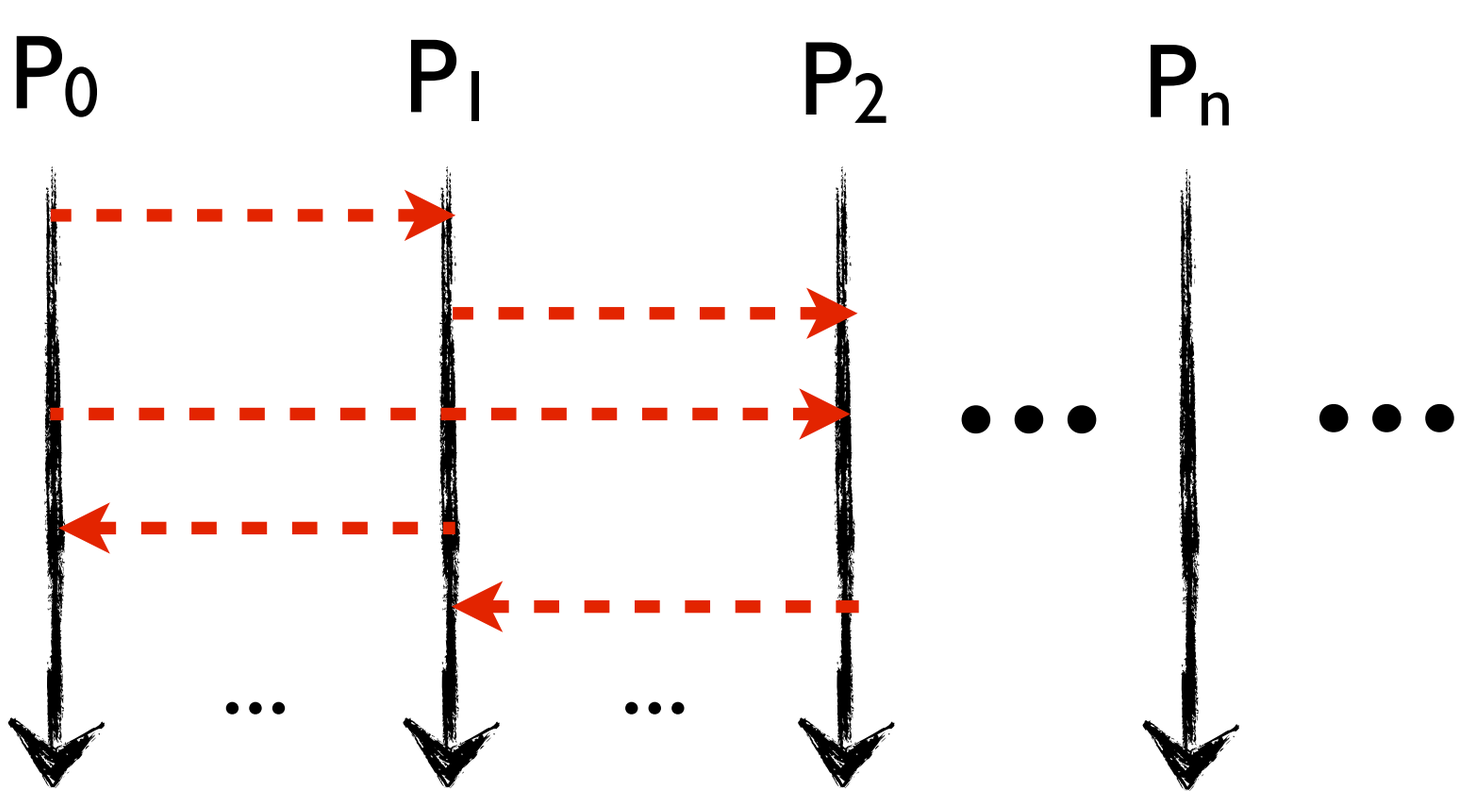
2014.12.02

High Performance Computing

- MPI is widely used for developing HPC applications
- MPI programs are *not* easy to develop and maintain

MPI Paradigm

- MPI implements a message-passing based parallel programming style



MPI Paradigm

- MPI implements a message-passing based parallel programming style
- Frequently used optimization trick
  - Overlapping of communication and computation
  - Asynchronous communication

Synchronization Error

- A buffer is written/read **before** *asynchronously* sent out/received

Synchronization Error

- A buffer is written/read **before** *asynchronously* sent out/received

Sending Example

```
ISend(..., buff, req)
buff[0] = 1
Wait(req)
```

Synchronization Error

- A buffer is written/read **before** *asynchronously* sent out/received

Sending Example

```
ISend(..., buff, req)
buff[0] = 1
Wait(req)
```

Receiving Example

```
IRecv(..., buff, req)
c = buff[0]
Wait(req)
```

Synchronization Error

- A buffer is written/read **before** *asynchronously* sent out/received
  - Incorrect computation and results
  - Crash the MPI application

Synchronization Error

- A buffer is written/read **before** *asynchronously* sent out/received
  - Incorrect computation and results
  - Crash the MPI application

How to detect synchronization errors in MPI programs?

Existing Approaches

- Dynamic methods
  - Runtime checking: SyncChecker[IPDPS'12], UMPIRE[SC'00], ...
    - Input coverage & Non-determinism
- No static methods
  - False alarms

Problem

- How to detect *input-related* synchronization error detection precisely?

P_0	P_1
ISend(P_1, *buff*, req) Wait(req)	IRecv(P_0, *buff*, req) if (!X) c = *buff*[0] Wait(req)

Problem

- How to detect *input-related* synchronization error detection precisely?

P_0	P_1
ISend(P_1, *buff*, req) Wait(req)	IRecv(P_0, *buff*, req) if (!X) c = *buff*[0] Wait(req)

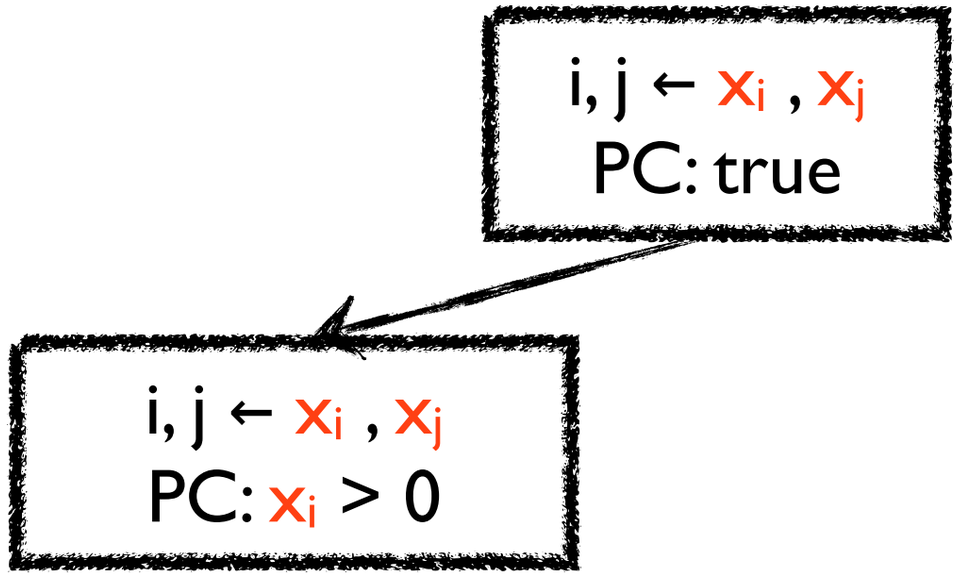
Symbolic execution based detection

Symbolic Execution

- A SAT/SMT based program analysis method
  - Execute a program with symbolic values
  - Convert a program into path conditions
  - A precise method
- Usages
  - Test generation, bug finding, *etc.*

Symbolic Execution

```
int main(int i, j) {
    if (i > 0) {
      i = i + j
    } else {
      i = i - j
    }
    return i
}
```

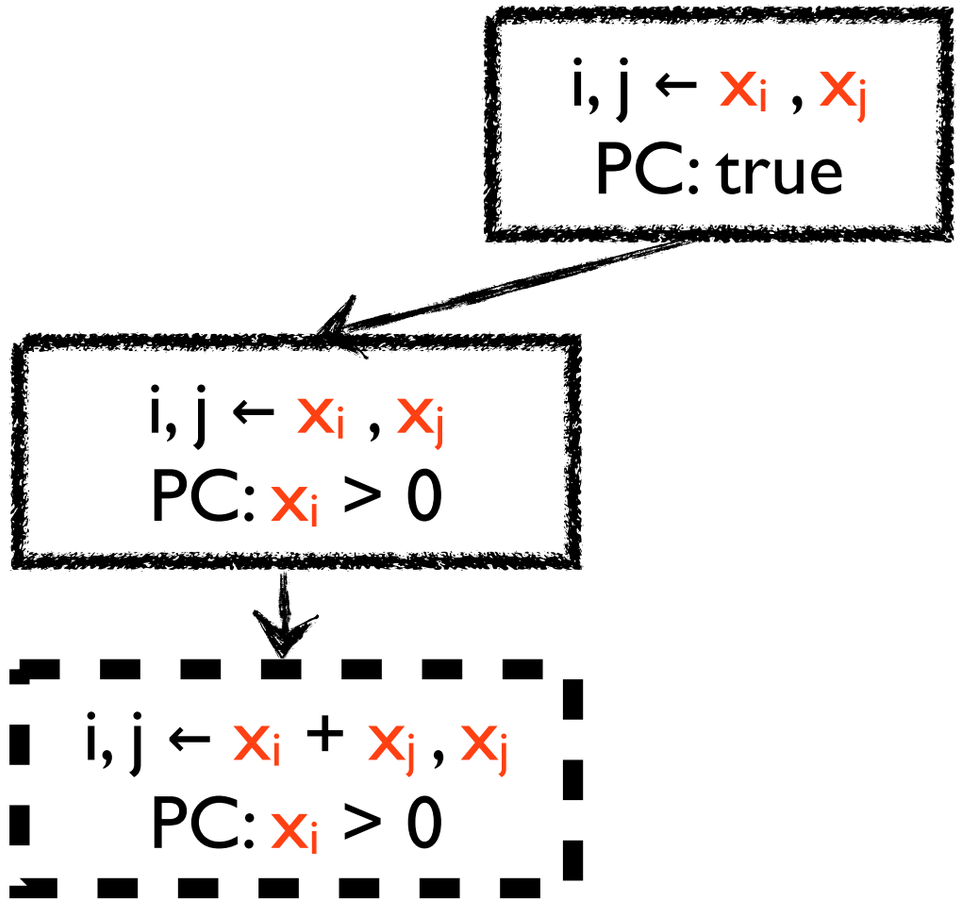


$x_i > 0$? ?

Solving

Symbolic Execution

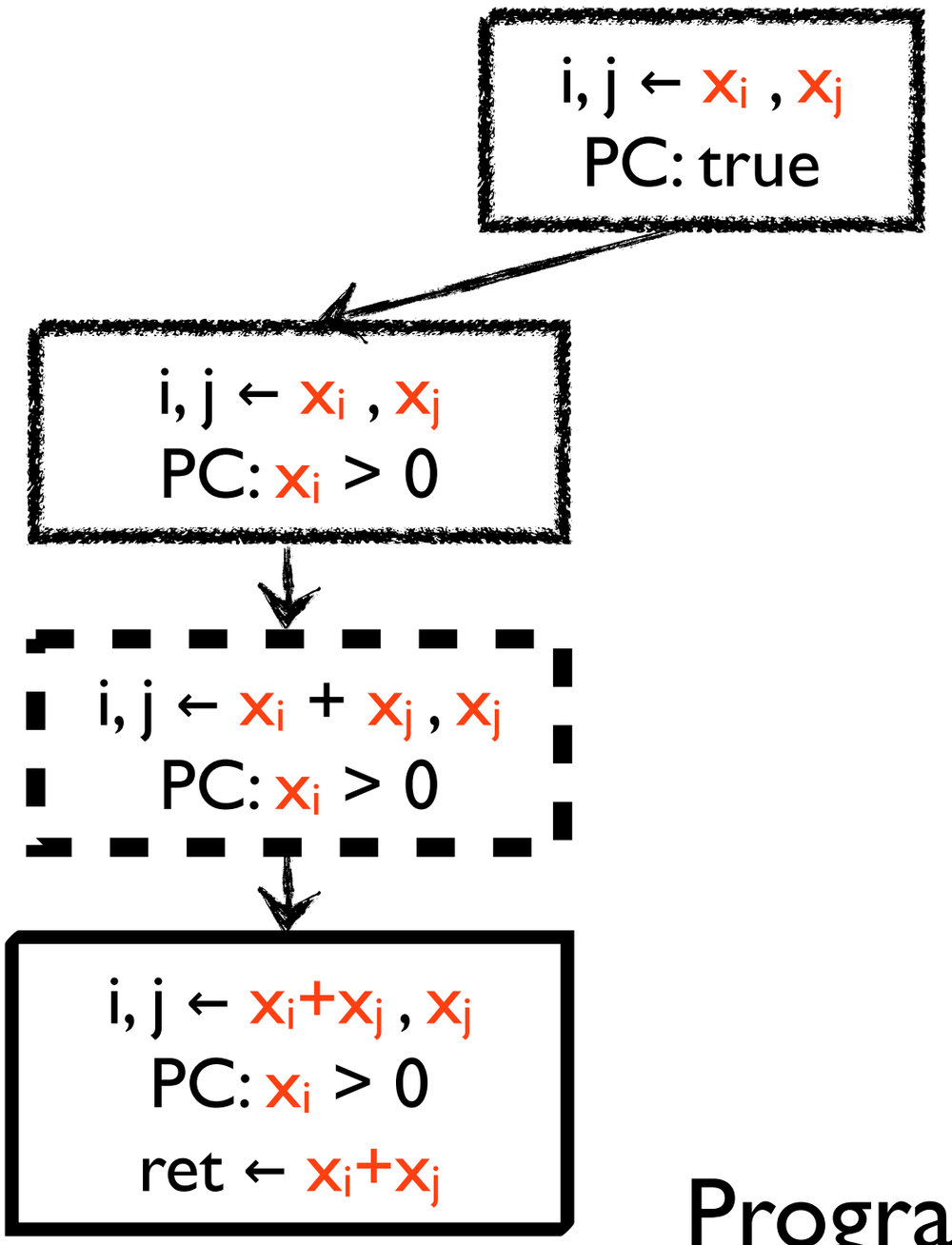
```
int main(int i, j) {
    if (i > 0) {
      i = i + j
    } else {
      i = i - j
    }
    return i
}
```



Symbolic Calculation

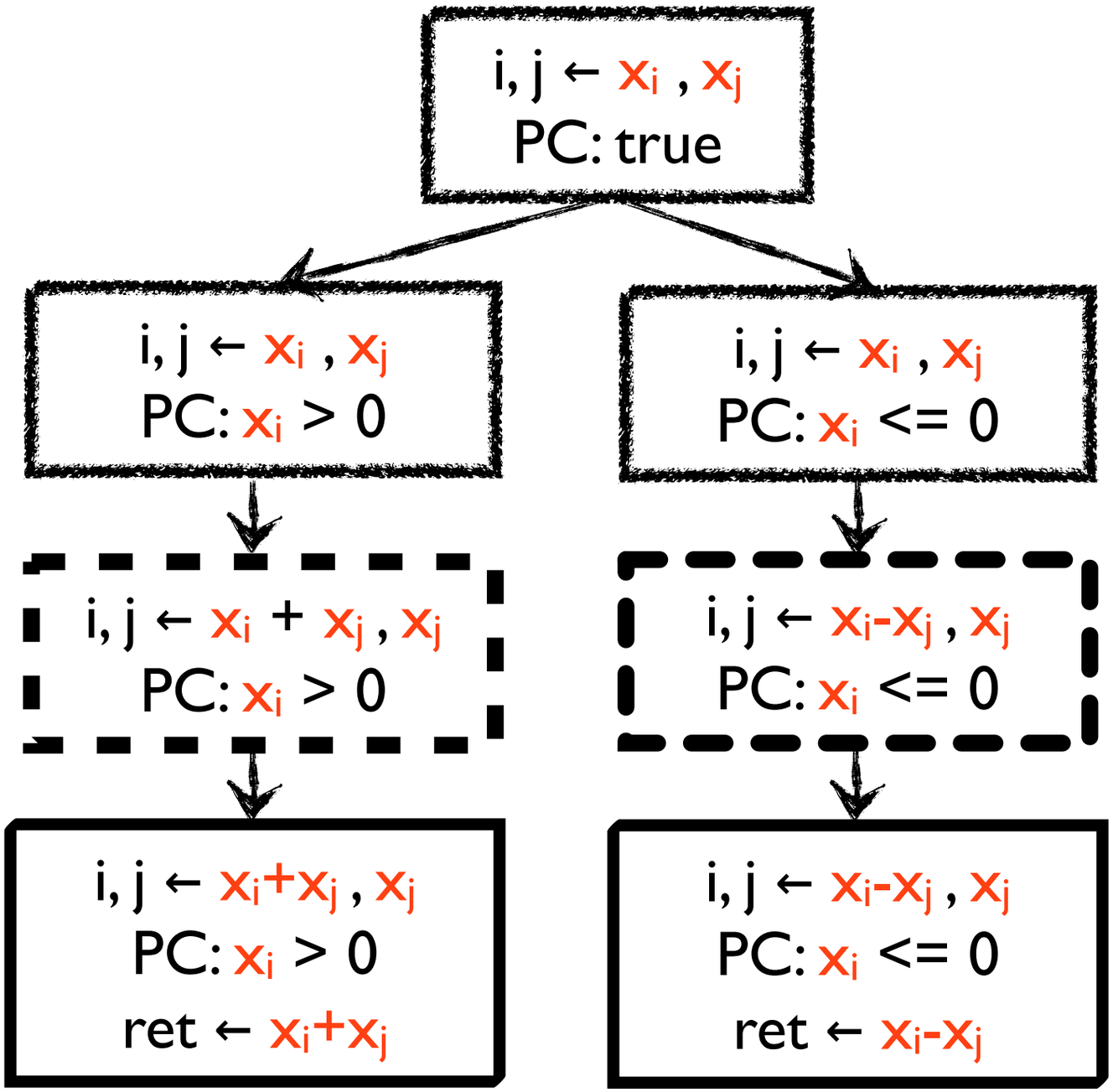
Symbolic Execution

```
int main(int i, j) {
    if (i > 0) {
      i = i + j
    } else {
      i = i - j
    }
    return i
}
```



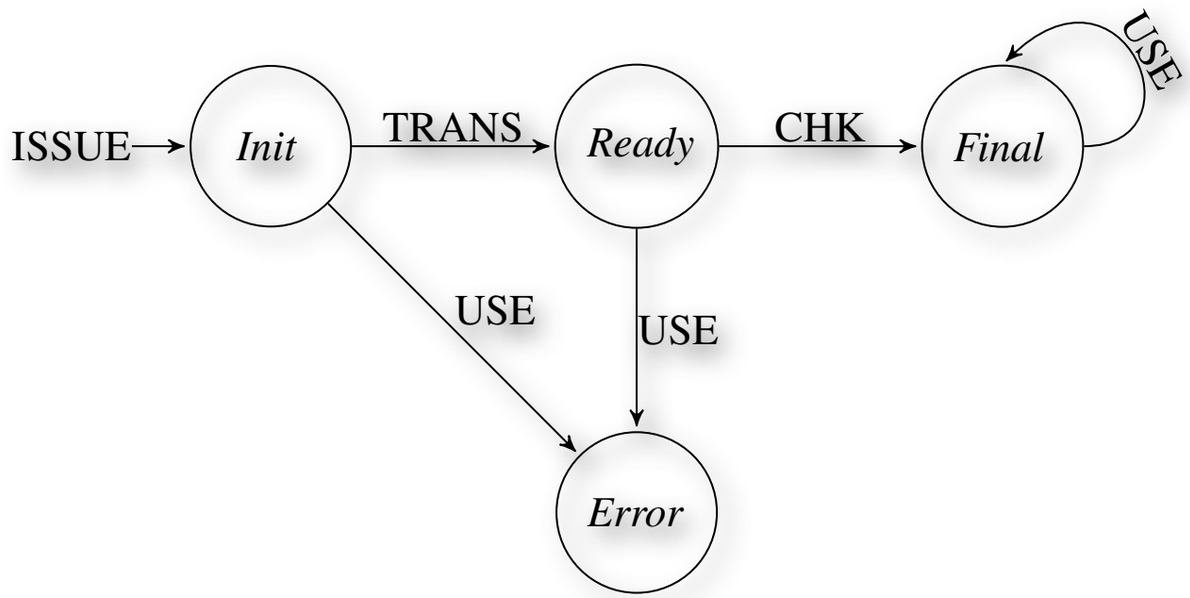
Symbolic Execution

```
int main(int i, j) {
    if (i > 0) {
      i = i + j
    } else {
      i = i - j
    }
    return i
}
```



Key Idea

- Use symbolic execution to ensure input coverage
- Track the state transition of each transferred buffer



Motivation Example

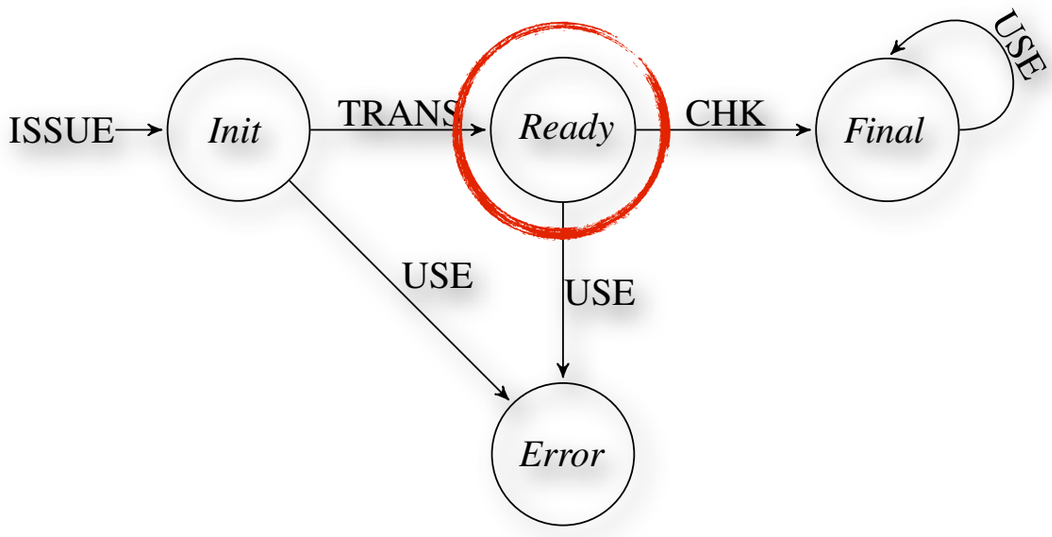
P_0

ISend(P_1, *buff*$_0$, req)
Wait(req)

P_1

IRecv(P_0, *buff*$_1$, req)
if (!X) c = *buff*$_1$[0]
Wait(req)

$x \leftarrow x_i$
PC: true



buff$_0$

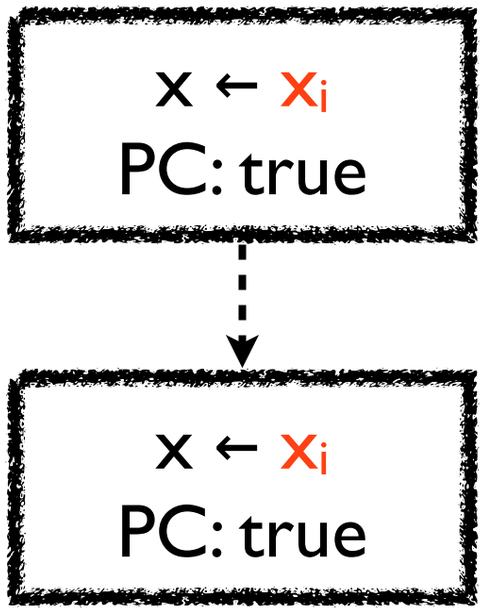
Motivation Example

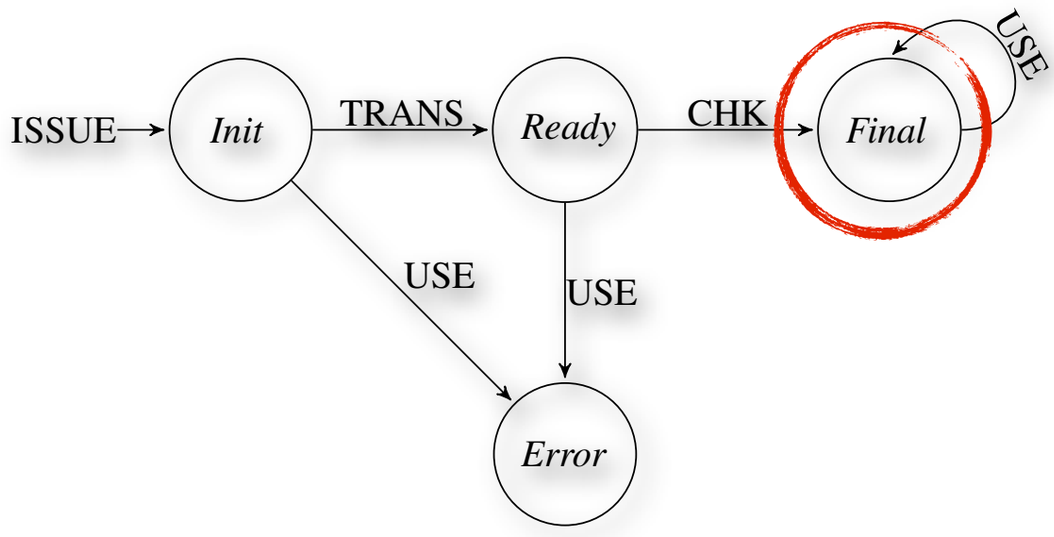
P_0

```
ISend(P1, buff0, req)
Wait(req)
```

P_1

```
IRecv(P0, buff1, req)
if (!X) c = buff1[0]
Wait(req)
```





$buff_0$

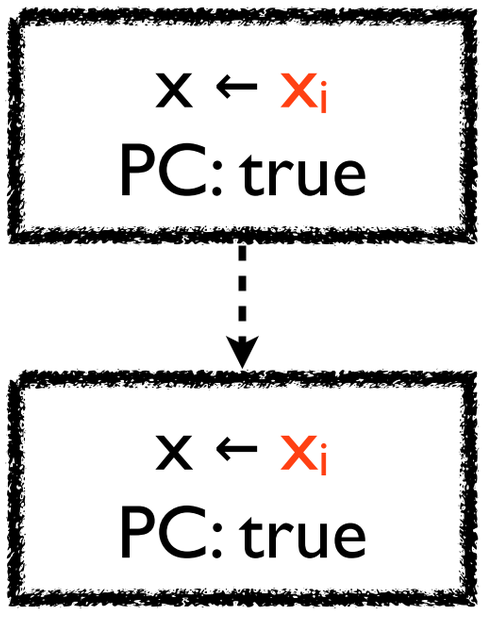
Motivation Example

P_0

```
ISend(P1, buff0, req)
Wait(req)
```

P_1

```
IRecv(P0, buff1, req)
if (!X) c = buff1[0]
Wait(req)
```



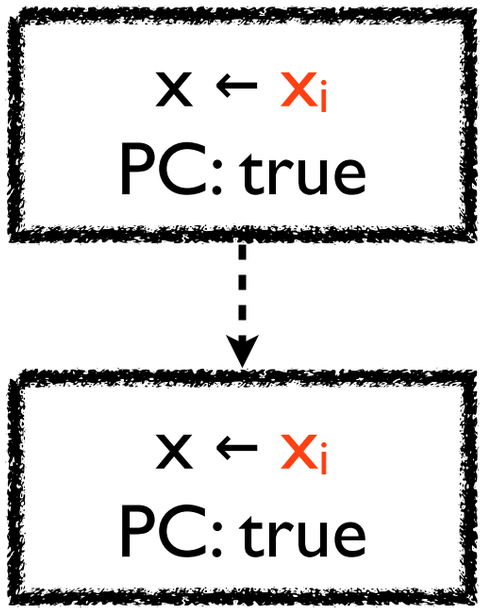
Motivation Example

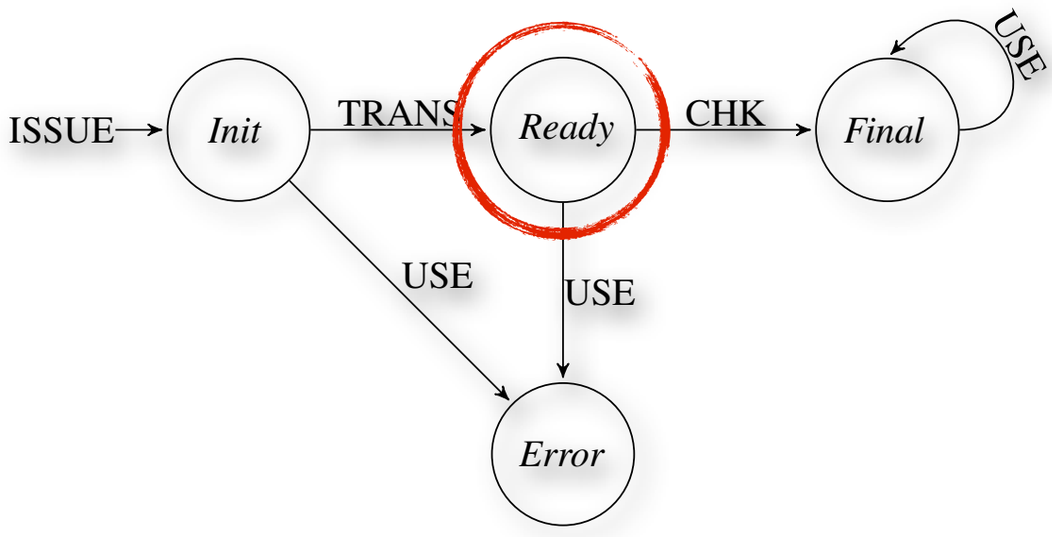
P_0

```
ISend(P1, buff0, req)
Wait(req)
```

P_1

```
IRecv(P0, buff1, req)
if (!X) c = buff1[0]
Wait(req)
```





$buff_1$

Motivation Example

P_0

```
ISend(P1, buff0, req)
Wait(req)
```

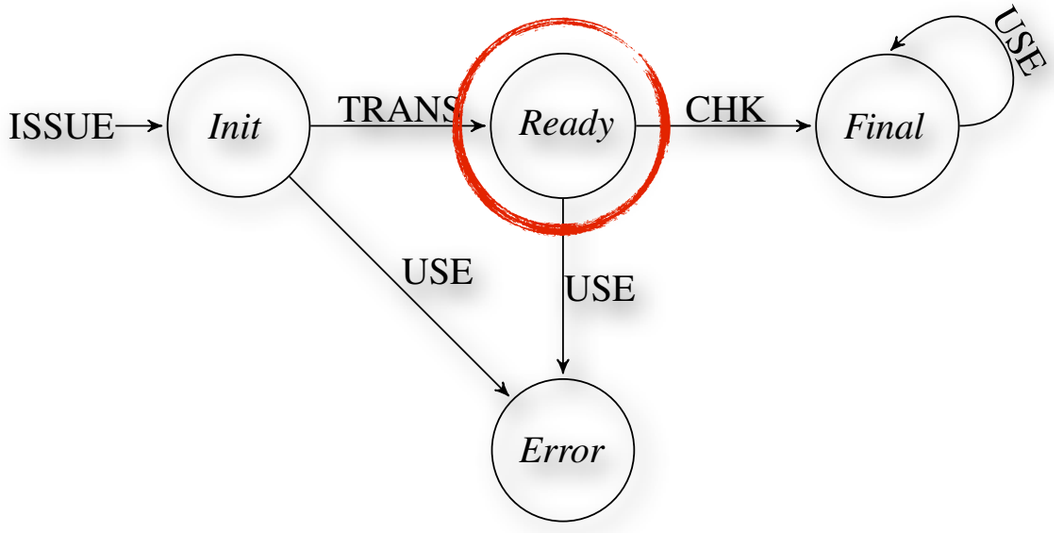
P_1

```
IRecv(P0, buff1, req)
if (!X) c = buff1[0]
Wait(req)
```

$x \leftarrow x_i$
PC: true

$x \leftarrow x_i$
PC: true

$x \leftarrow x_i$
PC: x_i != 0



$buff_1$

Motivation Example

P_0

```
ISend(P1, buff0, req)
Wait(req)
```

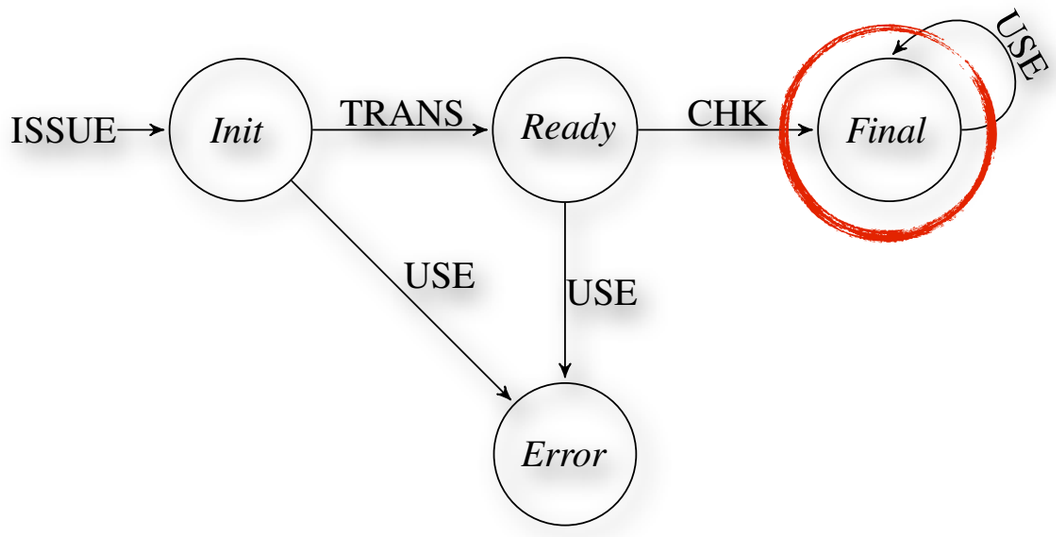
P_1

```
IRecv(P0, buff1, req)
if (!X) c = buff1[0]
Wait(req)
```

$x \leftarrow x_i$
PC: true

$x \leftarrow x_i$
PC: true

$x \leftarrow x_i$
PC: x_i != 0



$buff_1$

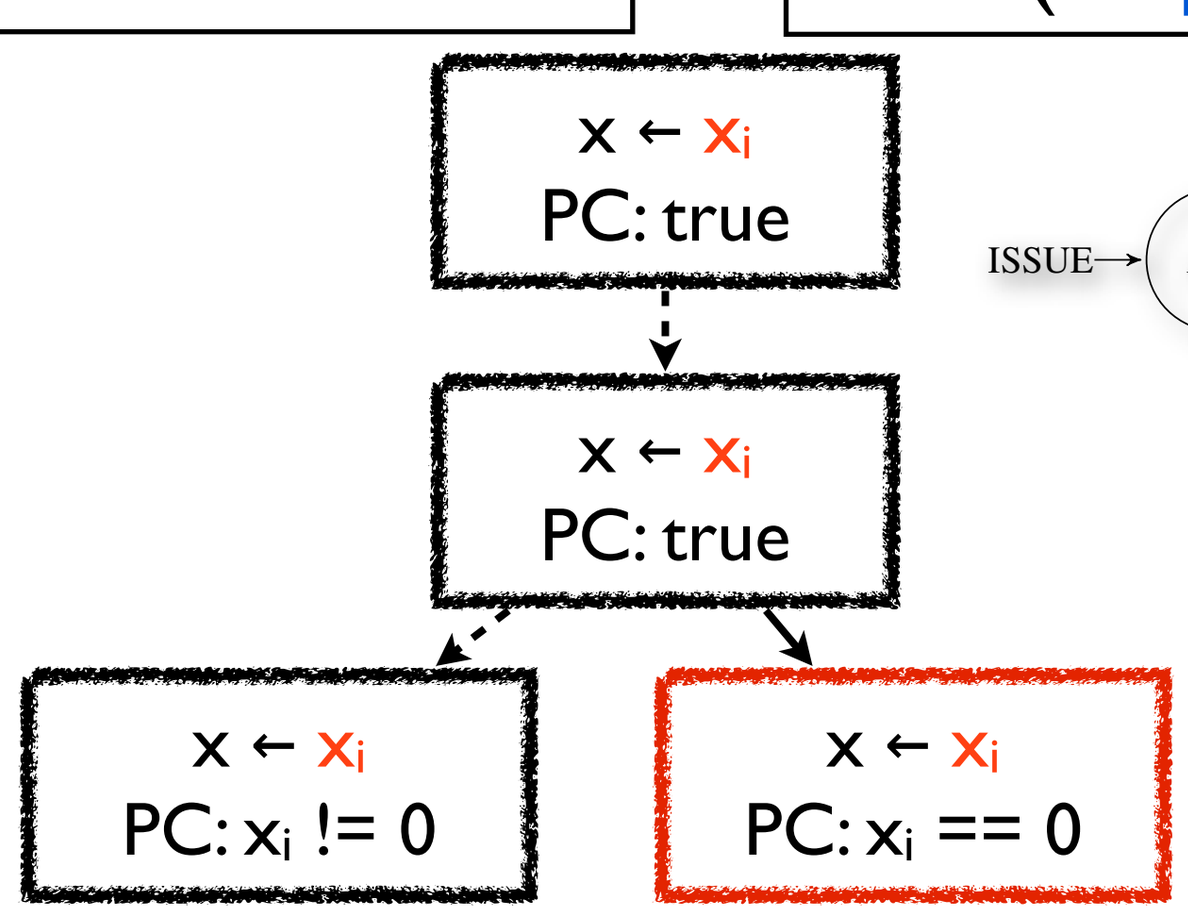
Motivation Example

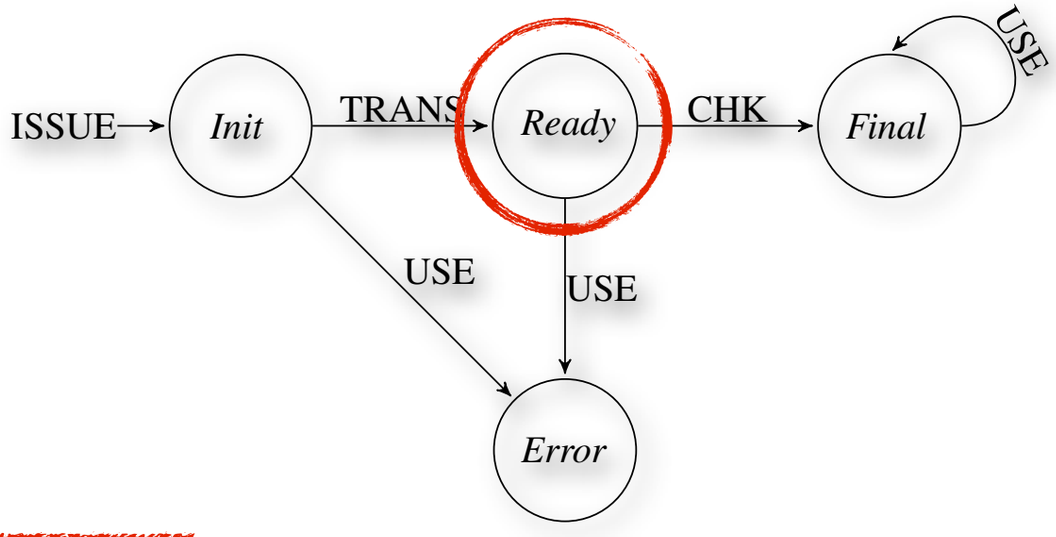
P_0

```
ISend(P1, buff0, req)
Wait(req)
```

P_1

```
IRecv(P0, buff1, req)
if (!X) c = buff1[0]
Wait(req)
```





$buff_1$

Motivation Example

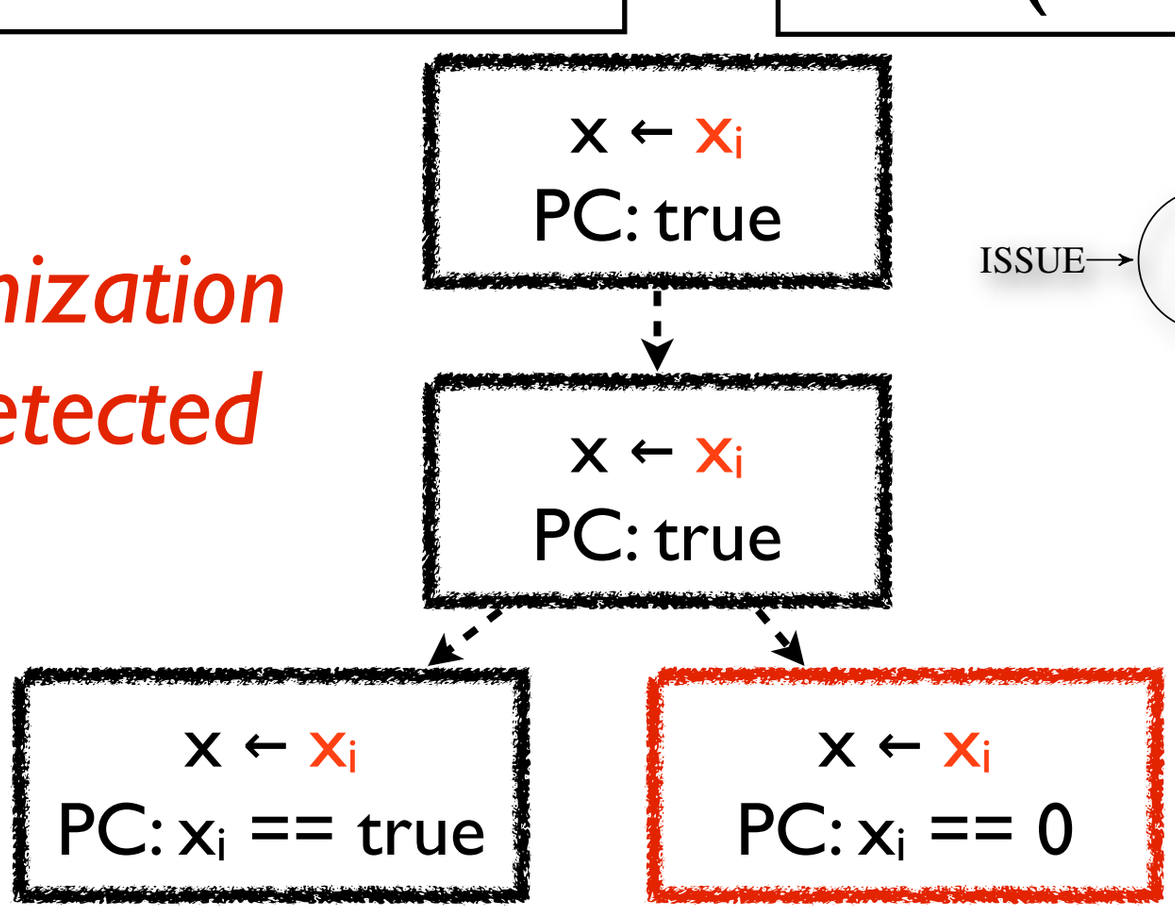
P_0

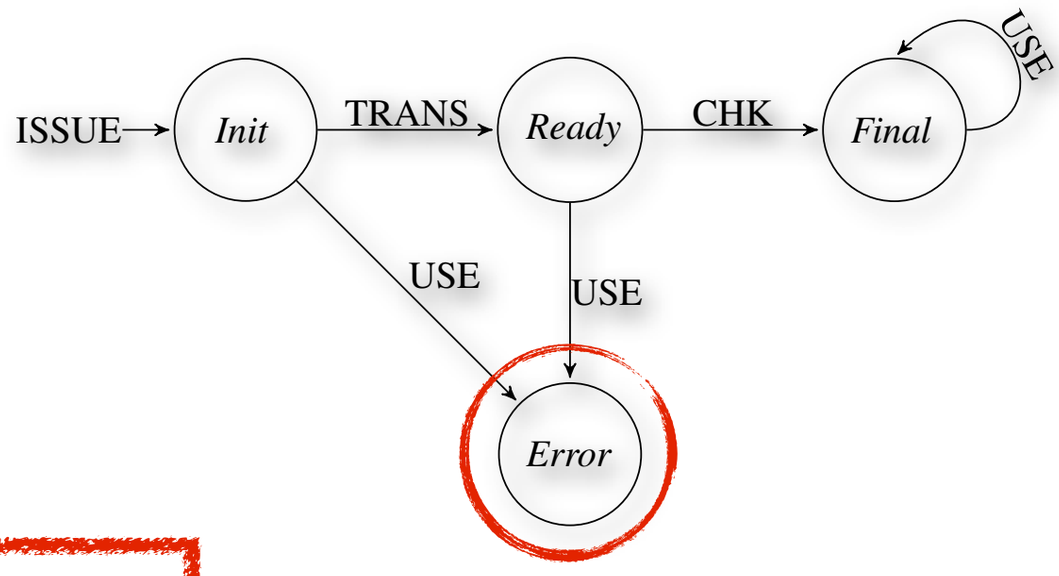
```
ISend(P1, buff0, req)
Wait(req)
```

P_1

```
IRecv(P0, buff1, req)
if (!X) c = buff1[0]
Wait(req)
```

A synchronization error is detected





buff1

Internals of Our Method

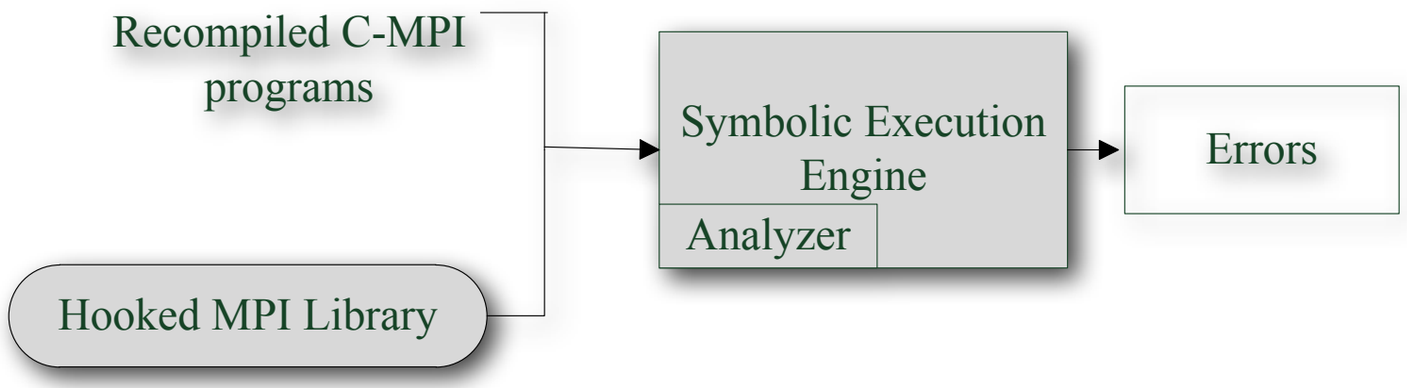
- Symbolic execution framework
  - Fixed number of processes
    - A round-robin style schedule
    - A process is preempted when being blocked
- Synchronization error checking as a dynamic typestate analysis

Two Optimizations

- Optimization 1
  - Only track the buffers used on application level
- Optimization 2
  - Remove the buffer from checking list when its state reaches the final state

Implementation

- Cloud9 based implementation
  - A multi-thread MPI library (*azequiaMPI*) as the environment model for MPI



Experiments

- MPI Programs

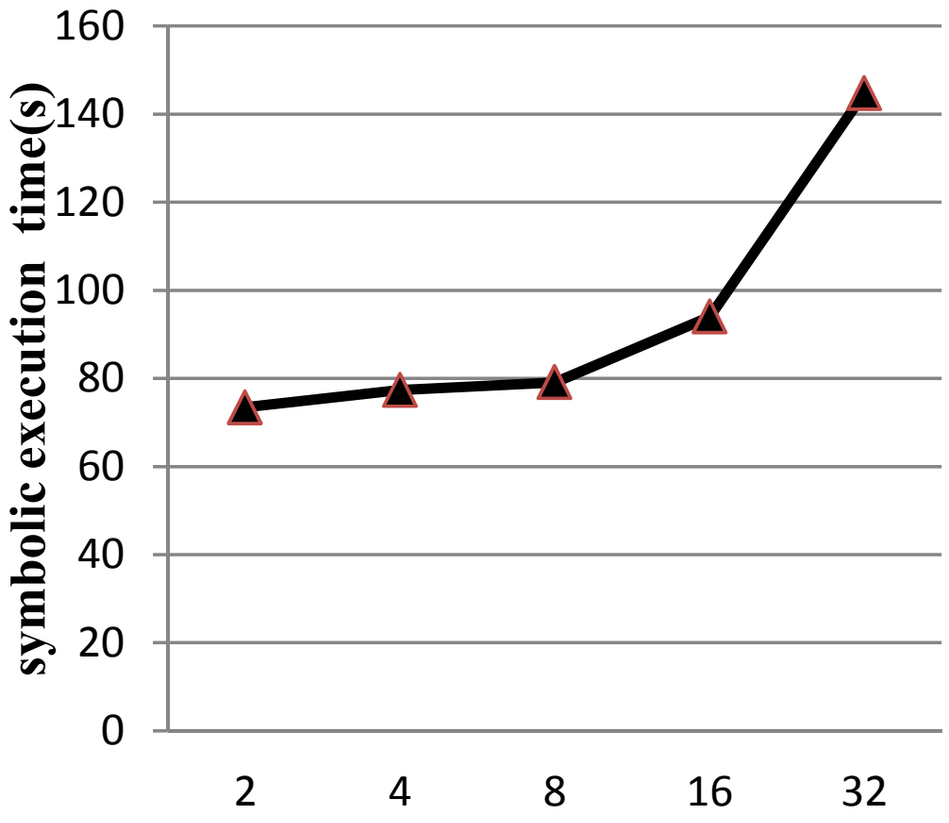
Programs	**Description**
change-send-buffer	Programs from Umpire benchmark
vector-isend	
noerror-wait	
irecv-isend	
athena 4.0	Astrophysical magneto hydrodynamics
heat-errors	The equation of heat conduction
IS	Integer sort from NPB

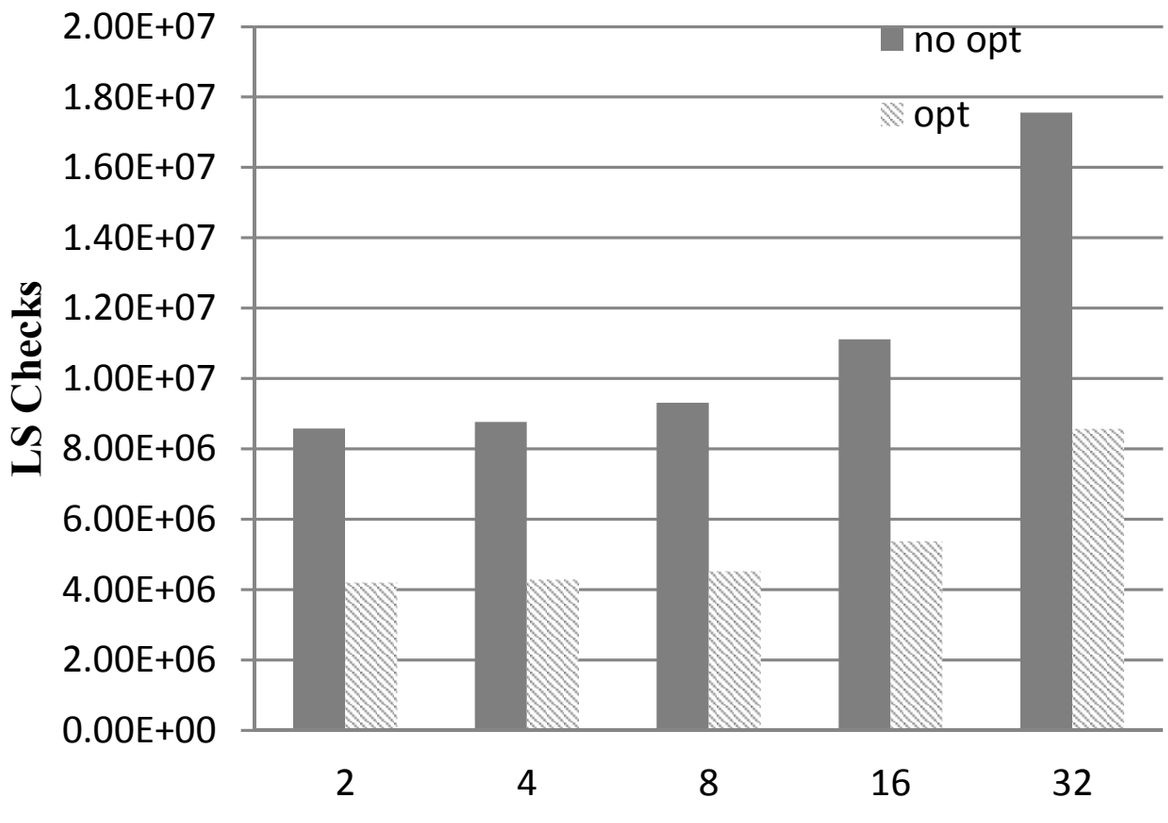
Results (1/2)

Program	#proc	Error	#ins	LS checks	
				no opt	opt
change-send-buffer	2	send	29990456	4322220	17960
vector-isend	2	send	30008052	4346419	42091
noerror-wait*	2	recv	30074743	4323203	18943
irecv-isend*	2	recv	30082632	4325064	20606
athena4.0*	2	recv	37285238	5559386	1249438
heat-errors	2	send	31373864	4411571	103350
IS*	16	recv	71232393	11117094	5384417

Results (2/2)

- Analyzing NPB IS with different numbers of processes





Conclusion

- A symbolic execution based method for detecting synchronization errors in MPI programs
  - Two optimizations
- A prototype and the experiments on real-world MPI programs

Work in progress

- Sound method for analyzing asynchronous MPI programs
- Applications on more real-world MPI programs
- Improvements and optimizations on tool

Thank you!
Q&A