Towards Understanding The Effects of Intermittent Hardware Faults on Programs

Layali Rashid, Karthik Pattabiraman and Sathish Gopalakrishnan

Department of Electrical and Computer Engineering

The University of British Columbia

Motivation: Why Intermittent Faults?

- Intermittent faults are likely to be a significant concern in future processors
  - Do not persist forever unlike permanent faults
  - Persist for longer duration than transient faults
  - May impact program more than transient faults

- **Assumption:**
  - An intermittent fault affects two or more consecutive instructions in the program.

Contributions

- Study the impact of intermittent faults on programs.
  - Model the propagation of intermittent faults in programs at the instruction-level.
  - Validate the model using fault injections.

Motivation: Why Model Error Propagation?

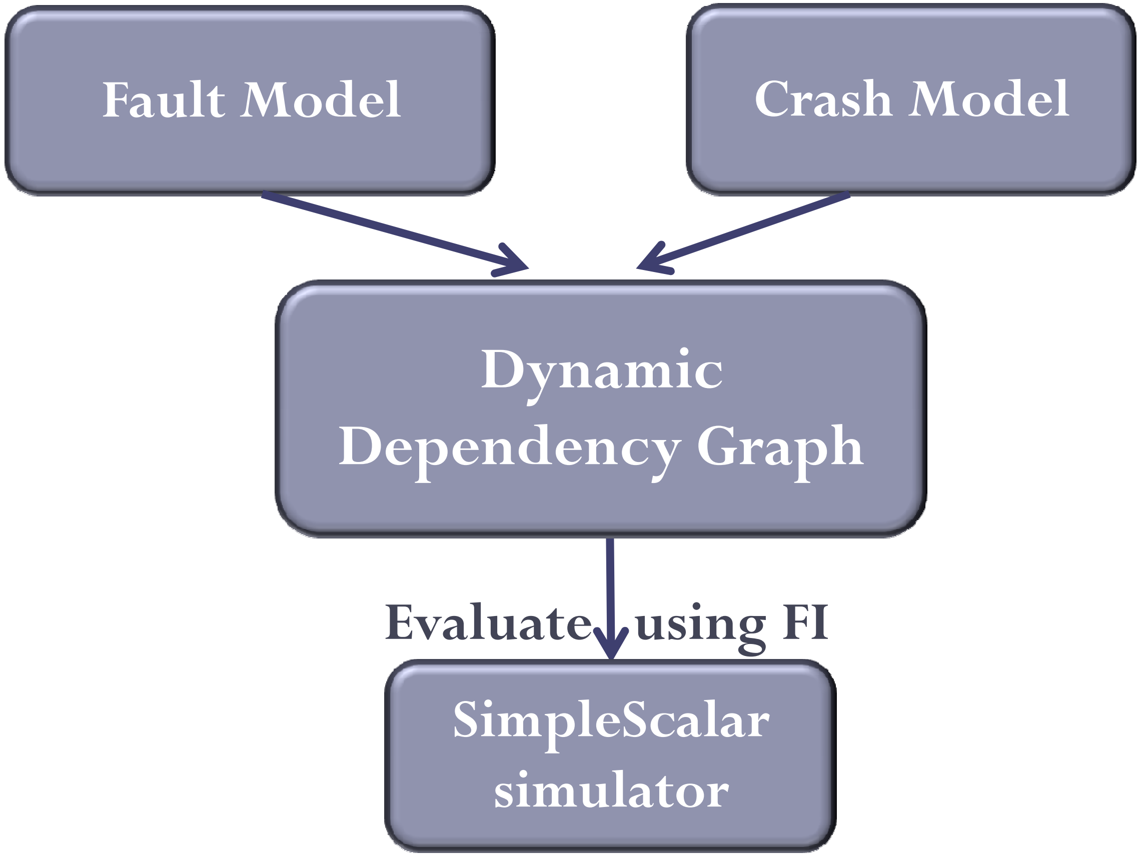
- Fault injection experiments are prohibitively expensive.
  - Intermittent faults vary in location and duration.
  - An order of magnitude slower than modeling.

- Modeling error propagation provides more insights that may help in tolerating faults.

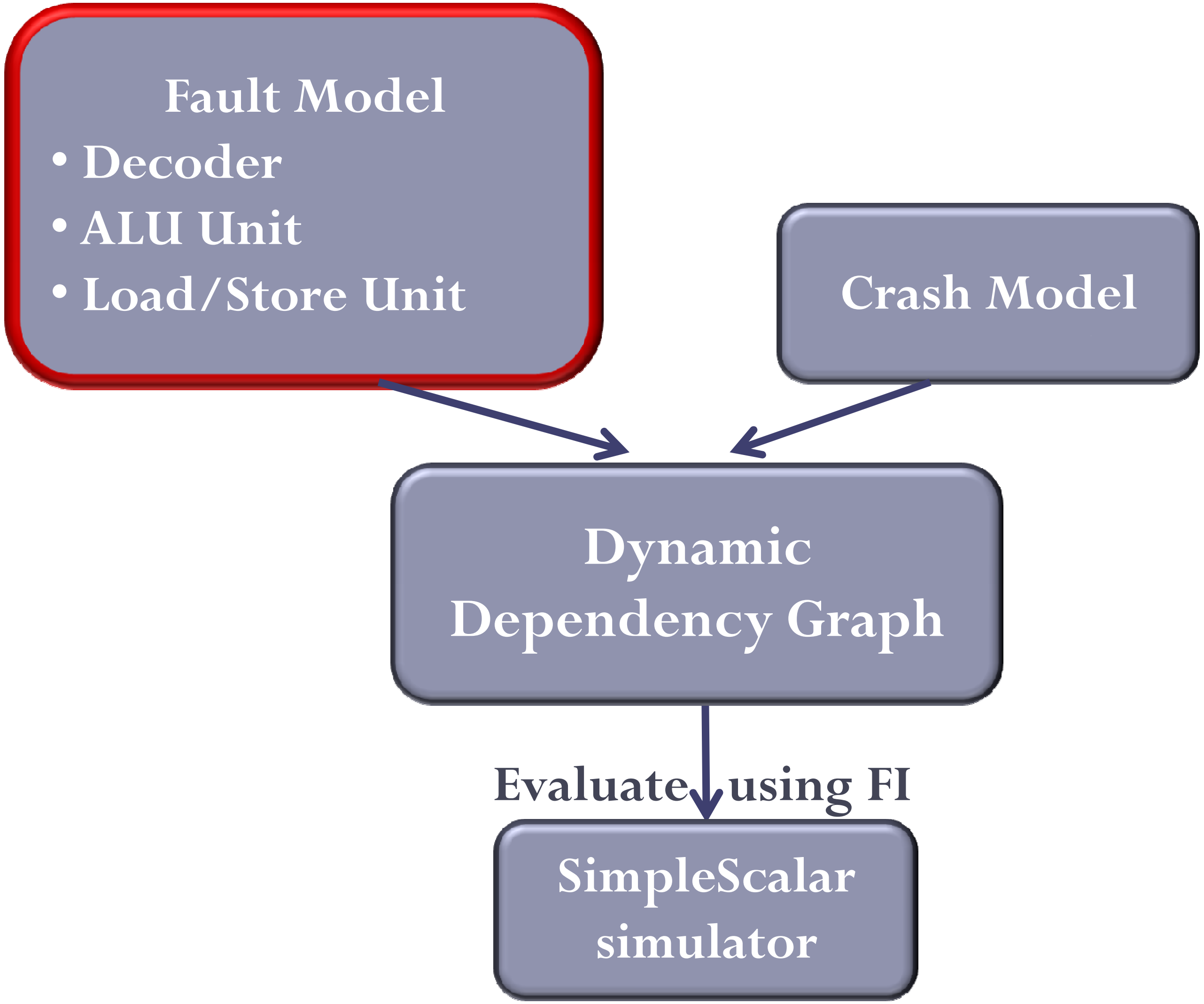
Primary Research Questions

- Do all intermittent faults lead to program crash?
- How many instructions are executed before the program crashes?
- How many variables are corrupted by the fault before the program crashes?

Approach


Fault Model
Crash Model
Dynamic Dependency Graph
Evaluate using FI
SimpleScalar simulator

Approach



Fault Model

- Decoder
- ALU Unit
- Load/Store Unit

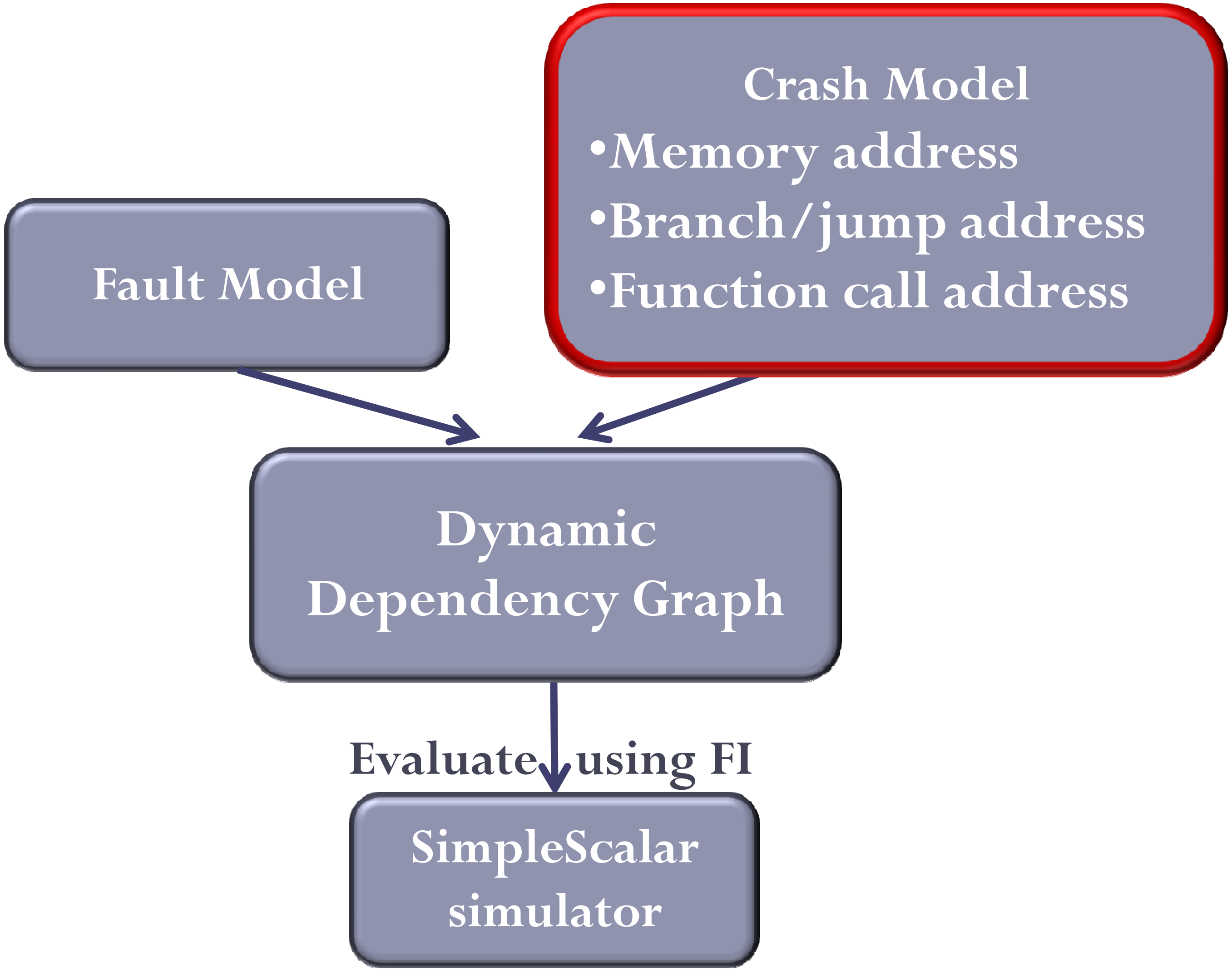
Crash Model

Dynamic Dependency Graph

Evaluate using FI

SimpleScalar simulator

Approach



Fault Model

Crash Model

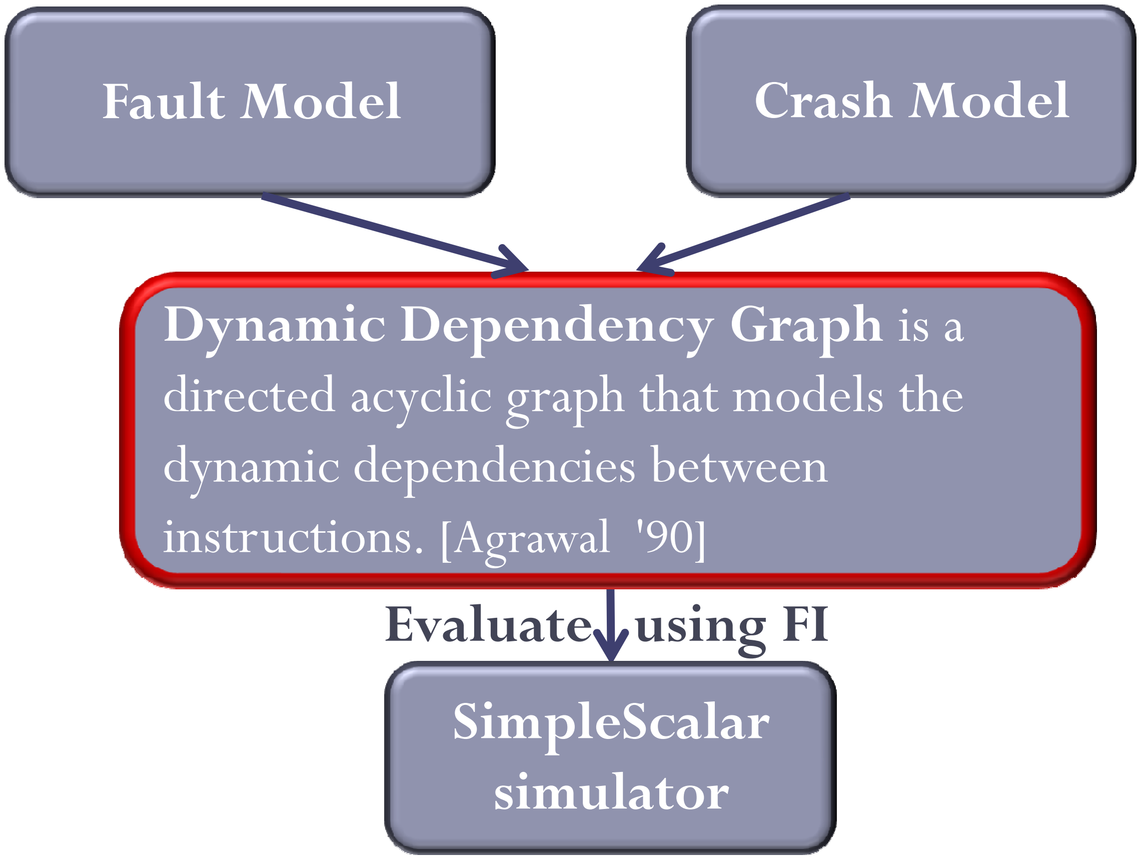
- Memory address
- Branch/jump address
- Function call address

Dynamic Dependency Graph

Evaluate using FI

SimpleScalar simulator

Approach



Fault Model

Crash Model

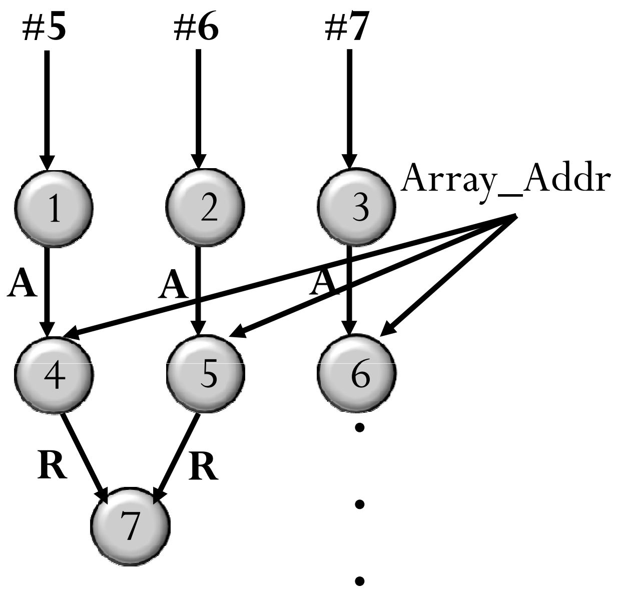
Dynamic Dependency Graph is a directed acyclic graph that models the dynamic dependencies between instructions. [Agrawal '90]

Evaluate using FI

SimpleScalar simulator

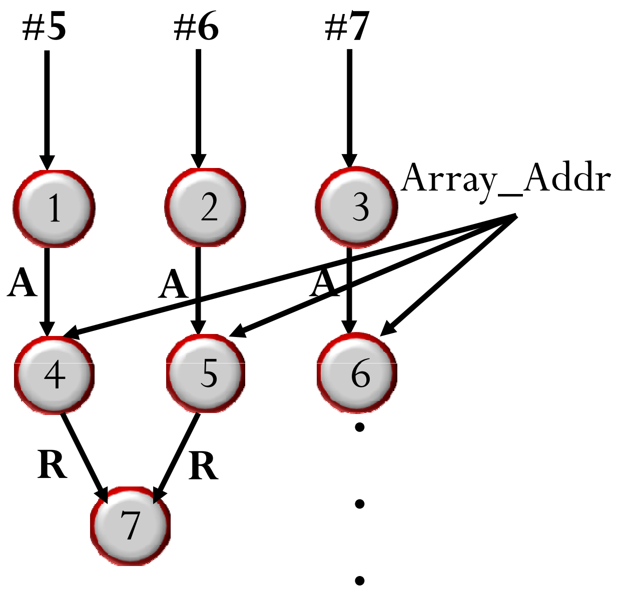
Example

Code Fragment	Node
mov R1, #5	1
mov R2, #6	2
mov R3, #7	3
ld R4, R1, Array_Addr	4
ld R5, R2, Array_Addr	5
ld R6, R3, Array_Addr	6
mult R7, R5, R4	7



Example

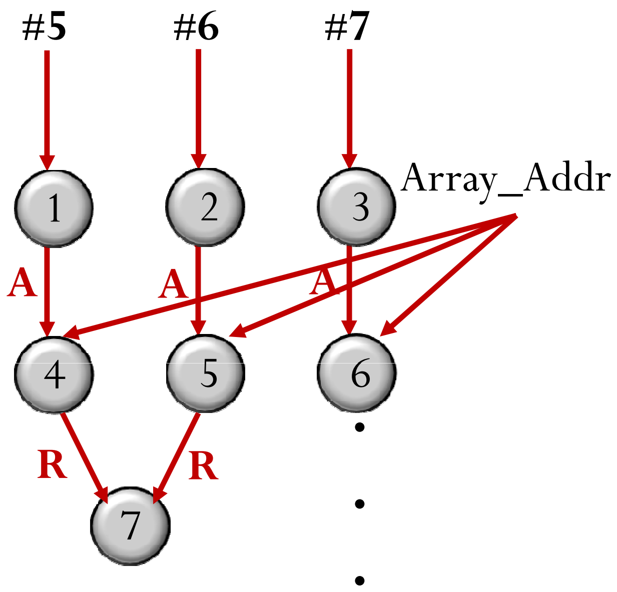
Code Fragment	Node
mov R1, #5	1
mov R2, #6	2
mov R3, #7	3
ld R4, R1, Array_Addr	4
ld R5, R2, Array_Addr	5
ld R6, R3, Array_Addr	6
mult R7, R5, R4	7



A node is a value produced by a dynamic instruction

Example

Code Fragment	Node
mov R1, #5	1
mov R2, #6	2
mov R3, #7	3
ld R4, R1, Array_Addr	4
ld R5, R2, Array_Addr	5
ld R6, R3, Array_Addr	6
mult R7, R5, R4	7



The edges represent the instructions' operands:

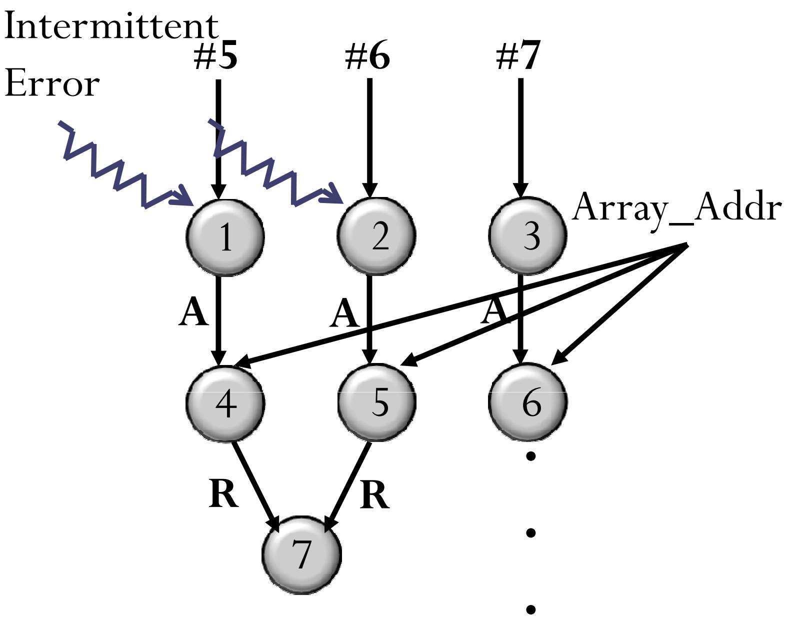
- A is an address operand
- R is a regular operand.

DDG Metrics

- **Intermittent Propagation Set (IPS)**: set of program values to which an intermittent fault propagates,
- **Crash Distance (CD)**: number of instructions that execute from the time an intermittent fault occurs until the program crashes (due to fault).

Example

Code Fragment	Node
mov R1, #5	1
mov R2, #6	2
mov R3, #7	3
ld R4, R1, Array_Addr	4
ld R5, R2, Array_Addr	5
ld R6, R3, Array_Addr	6
mult R7, R5, R4	7

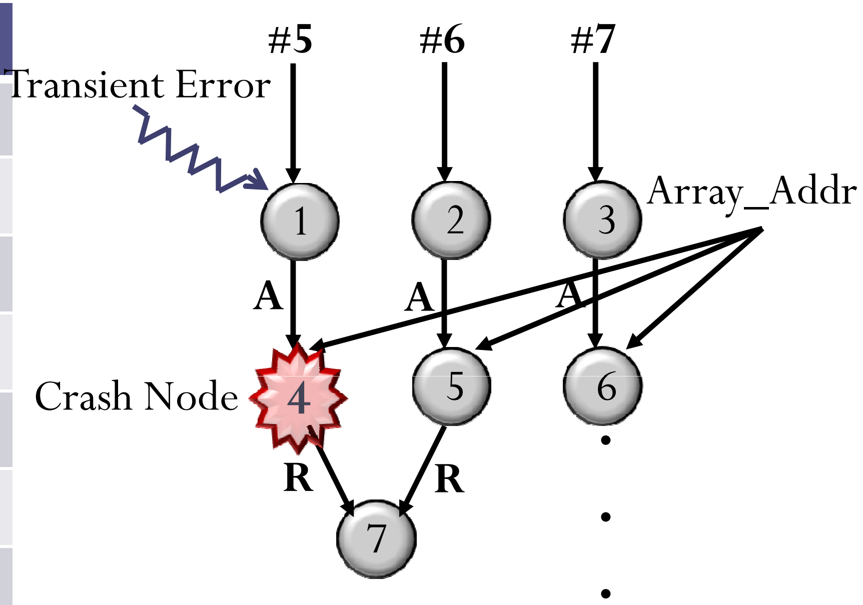


Intermittent Propagation Set (1,2) = {?}

Crash Distance (1, 2) = ?

Example

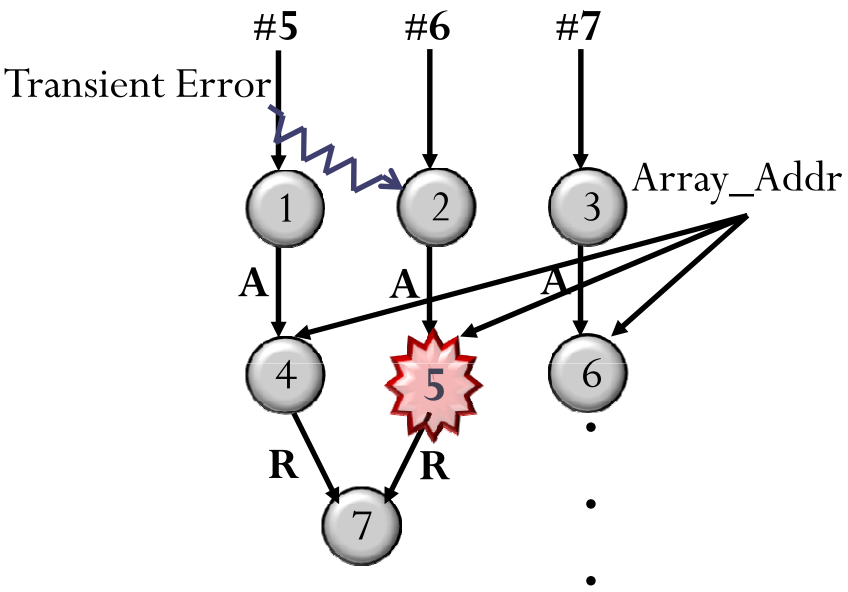
Code Fragment	Node
mov R1, #5	1
mov R2, #6	2
mov R3, #7	3
ld R4, R1, Array_Addr	4
ld R5, R2, Array_Addr	5
ld R6, R3, Array_Addr	6
mult R7, R5, R4	7



Transient Propagation Set (1) = {1, 4}
Transient Crash Distance (1) = 4

Example

Code Fragment	Node
mov R1, #5	1
mov R2, #6	2
mov R3, #7	3
ld R4, R1, Array_Addr	4
ld R5, R2, Array_Addr	5
ld R6, R3, Array_Addr	6
mult R7, R5, R4	7



Transient Propagation Set (1) = {1, 4}

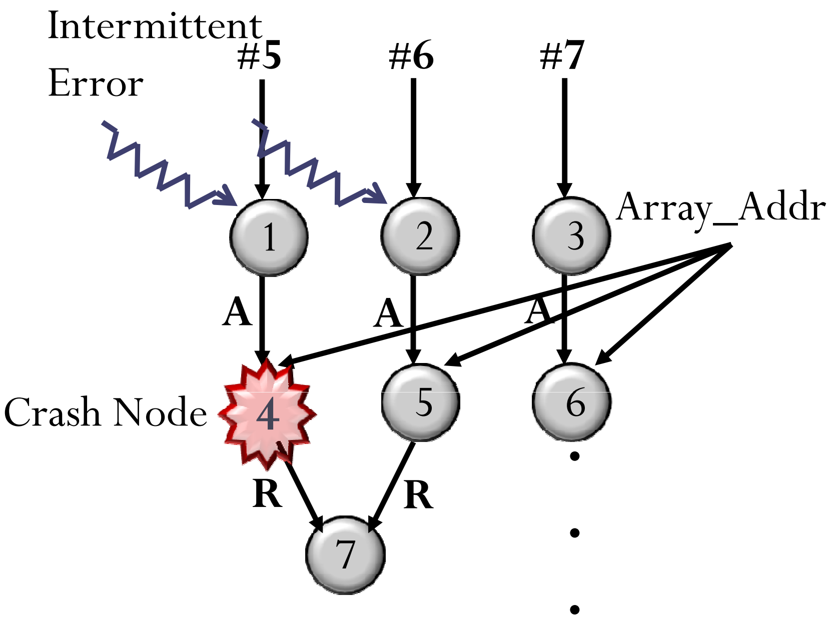
Transient Crash Distance (1) = 4

Transient Propagation Set (2) = {2, 5}

Transient Crash Distance (2) = 4

Example

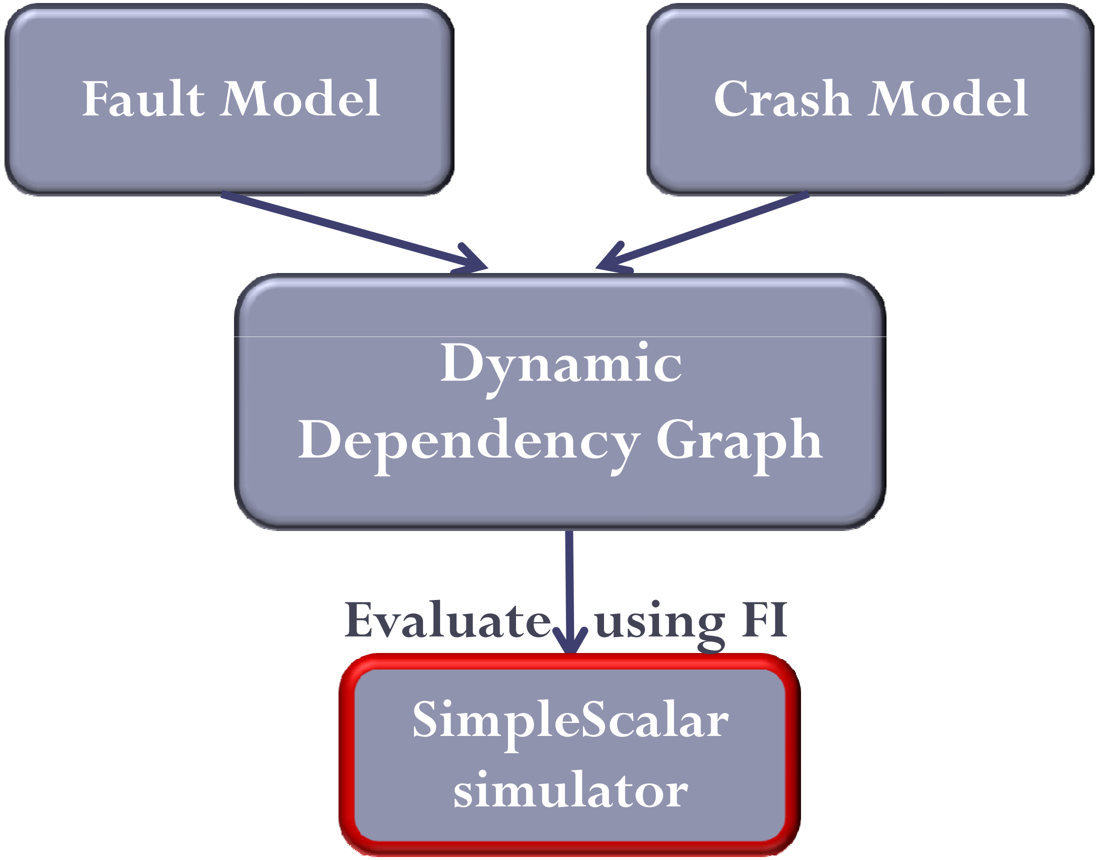
Code Fragment	Node
mov R1, #5	1
mov R2, #6	2
mov R3, #7	3
ld R4, R1, Array_Addr	4
ld R5, R2, Array_Addr	5
ld R6, R3, Array_Addr	6
mult R7, R5, R4	7



Intermittent Propagation Set (1,2) = {1, 2, 4}

Crash Distance (1, 2) = 4

Approach



Fault Model

Crash Model

Dynamic Dependency Graph

Evaluate using FI

SimpleScalar simulator

Experimental Setup

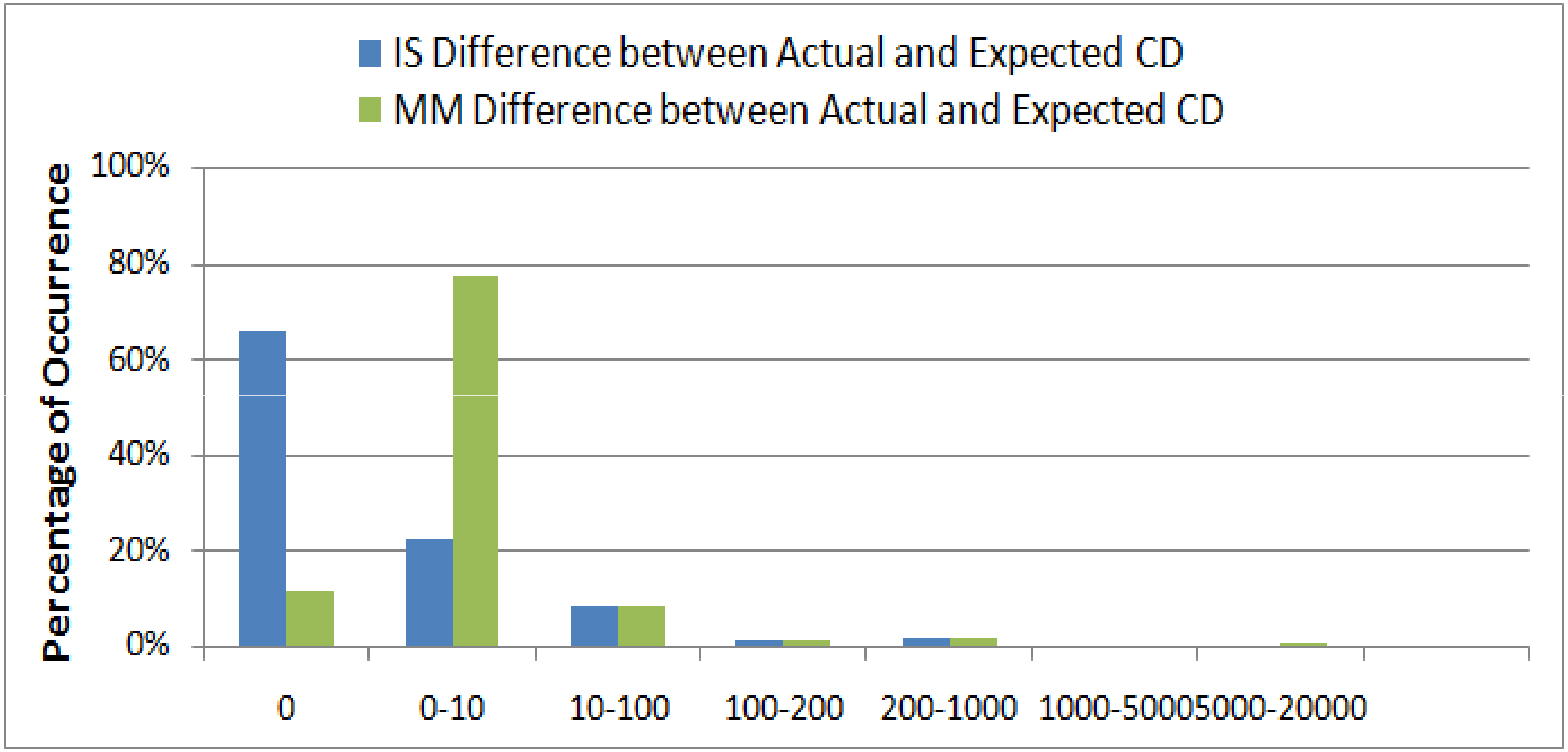
- **Evaluating the Model's Accuracy**
  - Intermittent fault injections in instruction level simulator (SimpleScalar)
  - Measure the difference between the predicted and the actual CD for crashes

- **Computation of Intermittent Fault Propagation**
  - Construct the DDG of each program.
  - Find the IPS and the CD for each fault

Benchmarks

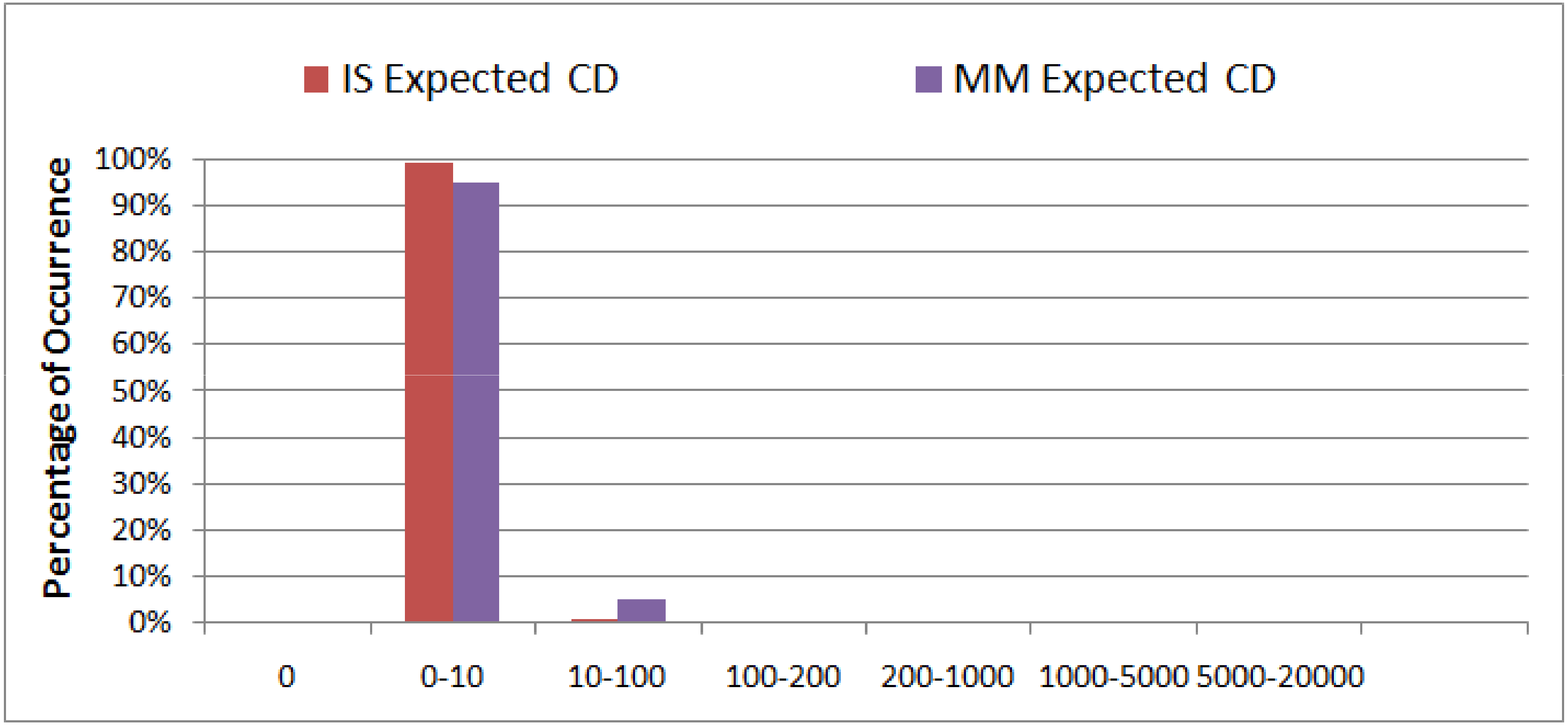
- Preliminary results for two programs: Matrix Multiply and Insertion Sort.
- Each program has about 11,000 static MIPS instructions.

Results: DDG Model Vs. SimpleScalar



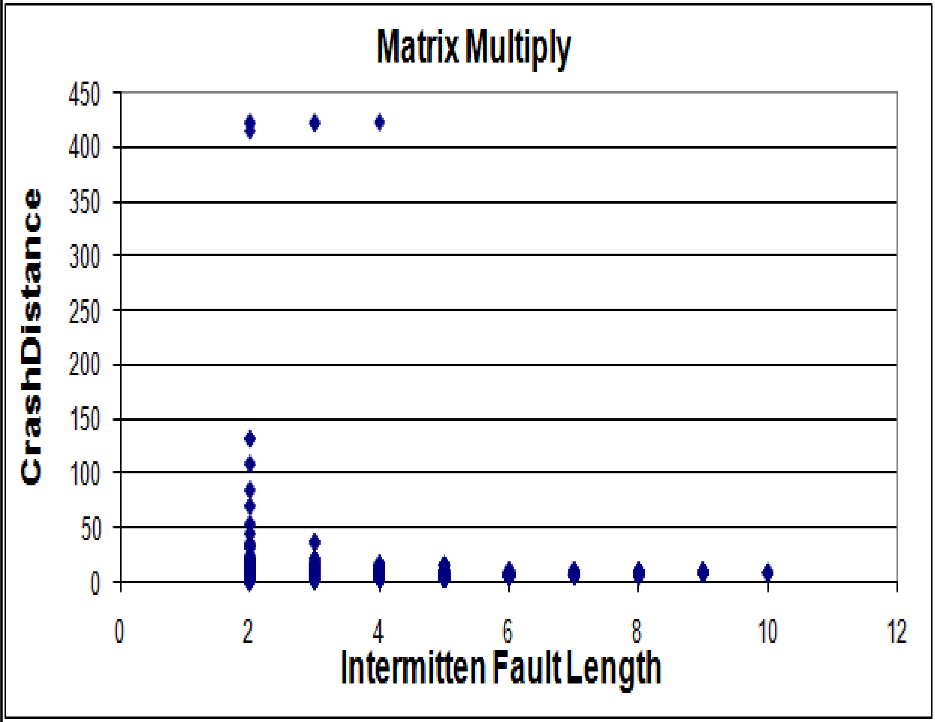
- 88% of the expected CD fall within 10 nodes from the actual ones and 97% fall within 100 nodes.

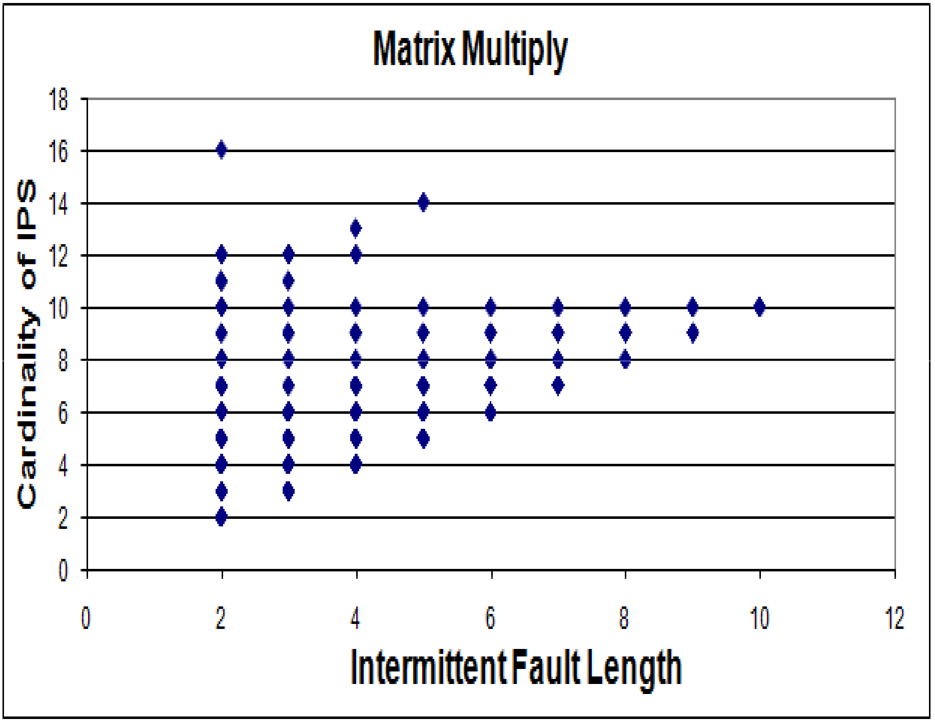
Results: CD Absolute values



- 95% of the faults cause program to crash within 10 nodes of the fault's start.

Results: Effect of Fault Length





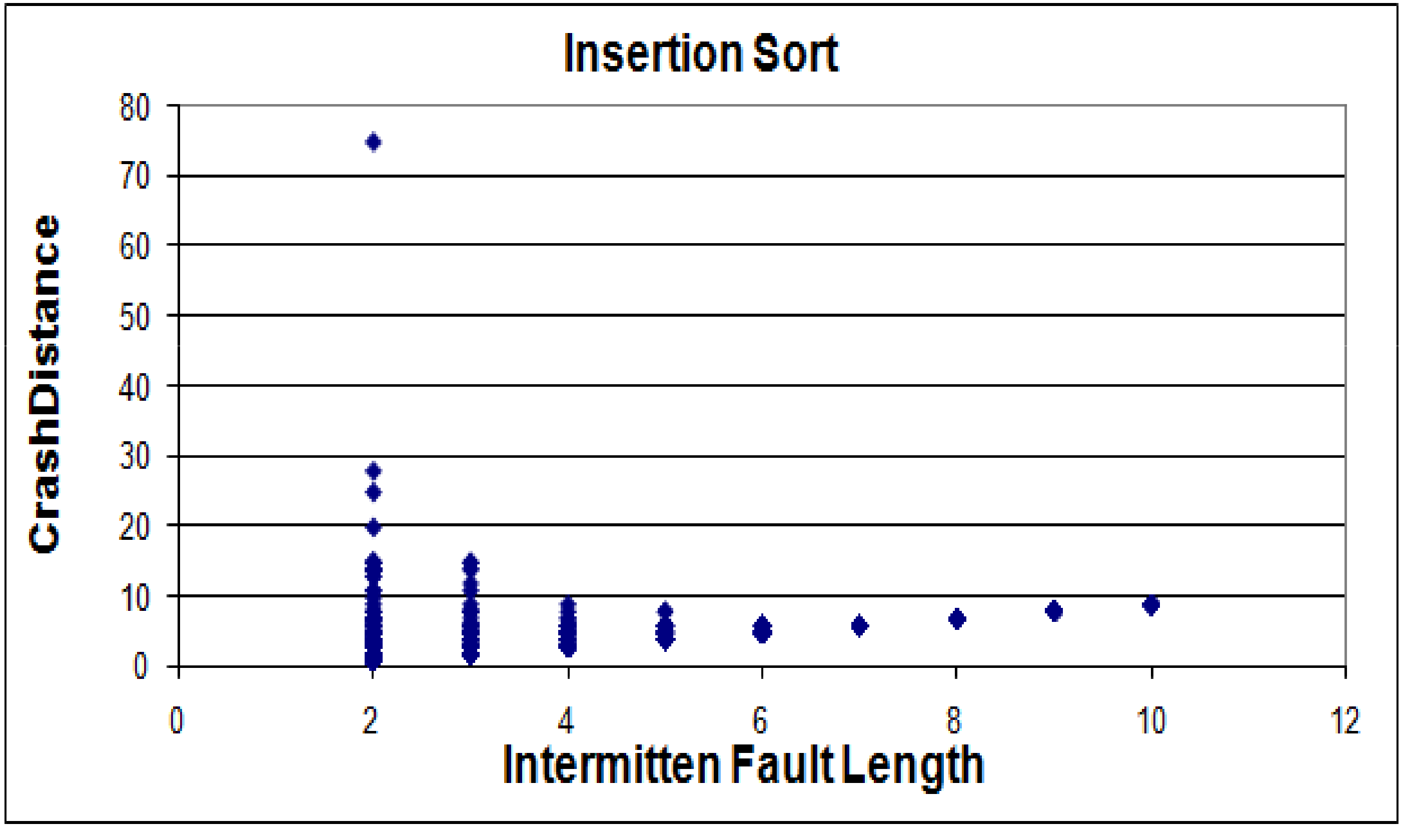
Conclusions and Discussion

- We enhanced Dynamic Dependency Graph to model intermittent fault propagation in programs.
- 88% of the expected faults' CDs fall within 10 nodes of the actual CDs.
- The majority of the intermittent faults cause programs to crash within few hundreds of dynamic instructions.
- Discussion
  - **Detection** using software-based techniques of intermittent faults can be efficient.
  - **Diagnosis** of intermittent faults is possibly feasible using software-based techniques.
  - **Recovery** using check-pointing techniques on the order of thousands of instructions will be effective.

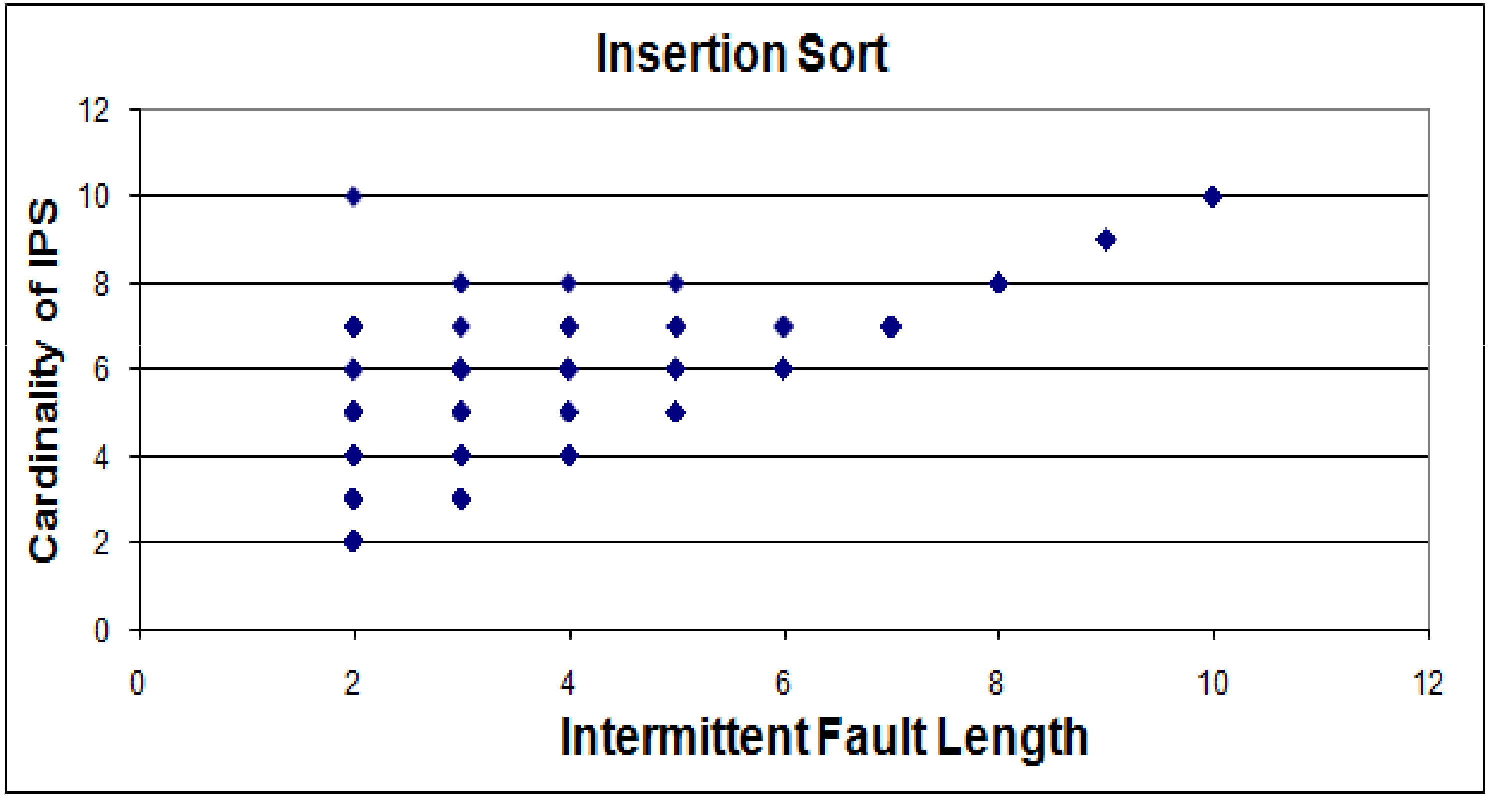
Thank You

Backup Slides

Insertion Sort CD



Insertion Sort IPS


Insertion Sort
Cardinality of IPS
Intermittent Fault Length