PreFail: Programmable and Efficient Failure Testing Framework

Cloud

Pallavi Joshi, Haryadi S. Gunawi, Koushik Sen
UC Berkeley

Motivation

- Large scale distributed systems face **frequent**, **multiple**, and **diverse** hardware failures
- Recovery protocols are often buggy
- Most of the previous work on failure testing focuses on single failures
- For multiple failures, brute force has to explore huge (e.g. **>40,000**) number of failure scenarios
- **Thus new challenge: combinatorial explosion of multiple failures**

Failure Testing

Example Program

Node A	Node B
L1. write(B, msg)	L1. write(A, msg)
L2. read(B, header)	L2. read(A, header)
L3. read(B, body)	L3. read(A, body)
L4. write(B, msg)	L4. write(A, msg)
L5. write(Disk, buf)	L5. write(Disk, buf)

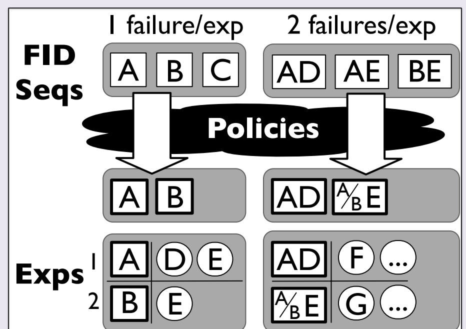
Failure ID (FID)

I/O ID Fields		Values
Static	func	write()
	src loc	Write.java (line L1)
Dynamic	node	A
	target	B
	stack	(the stack trace)
Domain Specific	network msg.	"Heartbeat Msg"
Failure ID = hash (I/O ID + Crash) = 2849067135		

Programmable Failure Testing

- Testers write policies to indicate the subset of failure space to explore
- Policies can be of varying complexities
- Policies can help achieve different coverage criteria
  - –code coverage, recovery coverage

Testing workflow



Policies

- **Filter policy**
  - –Express which failure sequences to exercise
- **Cluster policy**
  - –Express the equivalence of two failure sequences
  - –Only one failure sequence from an equivalence class is exercised

Recovery Coverage

```
def eqv (seq1, seq2):
  rPath1 = recoveryPath (seq1)
  rPath2 = recoveryPath (seq2)
  return rPath1 == rPath2

def recoveryPath(fSeq):
  a = allFids (fSeq)
  r = reducedFids (a, [`loc'])
  a0 = allFids ([])
  r0 = reducedFids (a0, [`loc'])
  rPath = r – r0
  return rPath
```

Code Coverage

```
def filter (fSeq):
    l = len(fSeq)
    last = fSeq [ l– 1 ]
    b = explored (last.loc)
    return not b

def cluster (fSeq1, fSeq2):
    l1 = len(fSeq1)
    l2 = len(fSeq2)
    last1 = fSeq1 [ l1– 1 ]
    last2 = fSeq2 [ l2– 1 ]
    b = (last1.loc == last2.loc)
    return b
```

Efficient Failure Testing

Optimizations

- Crashes before writes
- Read failures/corruption on first reads
- No crashes/network failures for dead nodes

Triaging

- Cluster according to root cause (bug)
- Sort according to bug type

Parallelization

- Experiments with failure sequences of a particular length i are distributed across m machines

Evaluation

- **Target systems** : HDFS, Zookeeper, Cassandra
- **6** new, 16 old bugs found
  - –data loss, unavailability
- Reduction of experiments with
  - –policies: 1 to 3 orders of magnitude
  - –optimizations: 5 times (average)