Why Is Random Testing Effective for Partition Tolerance Bugs?

Rupak Majumdar, Filip Niksic

Max Planck Institute for Software Systems (MPI-SWS)
Despite Many Formal Approaches...
Despite Many Formal Approaches…

…practitioners test their code
Despite Many Formal Approaches…

…practitioners **test** their code

…by providing **random** inputs.
Despite Many Formal Approaches…

…practitioners test their code

…by providing random inputs.

And despite our best judgement,
Despite Many Formal Approaches…

...practitioners test their code

...by providing random inputs.

And despite our best judgement,

...testing is surprisingly effective in finding bugs.
Despite Many Formal Approaches...

...practitioners test their code

...by providing random inputs.

And despite our best judgement,

...testing is surprisingly effective in finding bugs.

We explore this unexpected effectiveness in testing distributed systems under partition faults.
Jepsen: Call Me Maybe

A framework for black-box testing of distributed systems by randomly inserting network partition faults

1. General Random Testing Framework

2. Randomly Testing Distributed Systems

3. Wider Context: Combinatorial Testing
Tests and Goal Coverage

Tests $T$

Goals $G$
Tests and Goal Coverage

Tests $T$ cover some goals $G$. A test covers some goals.
Tests and Goal Coverage

Covering family = Set of tests that cover all goals
Tests and Goal Coverage

Covering family = Set of tests that cover all goals

“Small” covering families = Efficient testing
Random Testing

Pick a random test from $T$

Fix a goal from $G$

Suppose $P[\bullet \text{ covers } \circ] \geq p$

Characterize covering families with respect to $p$ and $|G|$
Probabilistic Method

Let $G$ be the set of goals and $P[\text{random } \bullet \text{ covers } \bullet] \geq p$

**Theorem.** There exists a covering family of size $p^{-1} \log|G|$. 
Probabilistic Method

Let $\mathbf{G}$ be the set of goals and $P[\text{random}\ \bullet\ \text{covers}\ \bullet] \geq p$

**Theorem.** There exists a covering family of size $p^{-1} \log|\mathbf{G}|$.

**Proof.**

$P[\text{random}\ \bullet\ \text{does not cover}\ \bullet] \leq 1 - p$
Probabilistic Method

Let $G$ be the set of goals and $P[\text{random } \bigcirc \text{ covers } \bigcirc] \geq p$

**Theorem.** There exists a covering family of size $p^{-1} \log |G|$.

**Proof.**

$P[\text{random } \bigcirc \text{ does not cover } \bigcirc] \leq 1 - p$

$P[\text{K independent } \bigcirc \text{ do not cover } \bigcirc] \leq (1 - p)^K$
Probabilistic Method

Let $G$ be the set of goals and $P[\text{random } \bullet \text{ covers } \bullet] \geq p$

**Theorem.** There exists a covering family of size $p^{-1} \log |G|$.

**Proof.**

$P[\text{random } \bullet \text{ does not cover } \bullet] \leq 1 - p$

$P[\text{K independent } \bullet \text{ do not cover } \bullet] \leq (1 - p)^K$

$P[\text{K independent } \bullet \text{ are not a covering family}] \leq |G| (1 - p)^K$
Let $G$ be the set of goals and $\Pr[\text{random test covers $G$}] \geq p$.

Theorem. There exists a covering family of size $p^{-1} \log |G|$.

Proof.

$\Pr[\text{random test does not cover $G$}] \leq 1 - p$

$\Pr[\text{K independent tests do not cover $G$}] \leq (1 - p)^K$

$\Pr[\text{K independent tests are not a covering family}] \leq |G|(1 - p)^K$

For $K = p^{-1} \log |G|$, this probability is strictly less than 1. Therefore, there must exist $K$ tests that are a covering family!
Probabilistic Method

Let $G$ be the set of goals and $P[\text{random covers } G] \geq p$

**Theorem.** There exists a covering family of size $p^{-1} \log|G|$.

**Theorem.** For $\epsilon > 0$, a random family of $p^{-1} \log|G| + p^{-1} \log \epsilon^{-1}$ tests is a covering family with probability at least $1 - \epsilon$. 
Random Testing Framework

1. What are tests?

2. What are testing goals?

3. What is the notion of coverage?

4. Can we bound $P[\text{random} \text{ covers}]$?
1. General Random Testing Framework

2. Randomly Testing Distributed Systems

3. Wider Context: Combinatorial Testing
Ninjas in Training

In a dojo in Kaiserslautern, \( n \) ninjas are in training.
Training is complete if for every pair of ninjas, there is a round where they are in opposing teams.

How many rounds make the training complete?
Ninjas in Training

In a dojo in Kaiserslautern, \( n \) ninjas are in training. Training is complete if for every pair of ninjas, there is a round where they are in opposing teams.

How many rounds make the training complete?
Ninjas in Training

In a dojo in Kaiserslautern, \( n \) ninjas are in training. Training is **complete** if for every pair of ninjas, there is a round where they are in opposing teams.

How many rounds make the training complete?
In a dojo in Kaiserslautern, \( n \) ninjas are in training. Training is **complete** if for every pair of ninjas, there is a round where they are in opposing teams.

**How many rounds make the training complete?**

Round 1:

Round 2:
In a dojo in Kaiserslautern, \( n \) ninjas are in training. Training is \textbf{complete} if for every pair of ninjas, there is a round where they are in opposing teams.

**How many rounds make the training complete?**
Ninjas in Training

In a dojo in Kaiserslautern, \( n \) ninjas are in training. Training is **complete** if for every pair of ninjas, there is a round where they are in opposing teams.

**How many rounds make the training complete?**
Ninjas in Training

In a dojo in Kaiserslautern, \( n \) ninjas are in training. Training is \textbf{complete} if for every pair of ninjas, there is a round where they are in opposing teams.

\textbf{How many rounds make the training complete?}
Ninjas in Training

In a dojo in Kaiserslautern, $n$ ninjas are in training. Training is **complete** if for every pair of ninjas, there is a round where they are in opposing teams.

How many rounds make the training complete?

- Naïve: $O(n^2)$

- Can you do it in $\log n$ rounds?
Ninjas in Training

More generally, \( n \) ninjas are training in \( k \) teams. Training is **complete** if for every choice of \( k \) ninjas, there is a round where they are each in different team.

**How many rounds make the training complete?**

Round 1:

- \( 1 \)
- \( 2 \)
- \( 3 \)
- \( \ldots \)
- \( n \)

Round 2:

- \( \ldots \)
- \( \ldots \)
Ninjas in Training

More generally, $n$ ninjas are training in $k$ teams. Training is **complete** if for every choice of $k$ ninjas, there is a round where they are each in different team.

**How many rounds make the training complete?**
Ninjas in Training

More generally, \( n \) ninjas are training in \( k \) teams. Training is complete if for every choice of \( k \) ninjas, there is a round where they are each in different team.

How many rounds make the training complete?
Ninjas in Training

More generally, \( n \) ninjas are training in \( k \) teams. Training is \textbf{complete} if for every choice of \( k \) ninjas, there is a round where they are each in different team.

How many rounds make the training complete?

Round 1:

Round 2:
More generally, \( n \) ninjas are training in \( k \) teams. Training is **complete** if for every choice of \( k \) ninjas, there is a round where they are each in different team.

**How many rounds make the training complete?**
Ninjas in Training

More generally, \( n \) ninjas are training in \( k \) teams. Training is **complete** if for every choice of \( k \) ninjas, there is a round where they are each in different team.

**How many rounds make the training complete?**

- Naïve: \( O(n^k) \)

- Can you do it in \( k^{k+1} \frac{(k!)^{-1}}{\log n} \) rounds?
From Training Ninjas to Distributed Systems with Partition Faults

- ninjas
- teams
- rounds
- complete training

- nodes in a network
- blocks in a partition
- partitions
- covering family
Splitting Coverage

Given $n$ nodes and $k \leq n$:

- Tests are partitions of nodes into $k$ blocks: $P = \{B_1, \ldots, B_k\}$
- Testing goals are sets of $k$ nodes: $S = \{x_1, \ldots, x_k\}$
- $P$ covers $S$ if $P$ splits $S$: $x_1 \in B_1, \ldots, x_k \in B_k$

Covering families are called $k$-splitting families here.
A Bug in Chronos

- A distributed fault-tolerant job scheduler
- Works in conjunction with Mesos and Zookeeper
- Three special nodes: Chronos leader, Mesos leader, Zookeeper leader
A Bug in Chronos

- A distributed fault-tolerant job scheduler
- Works in conjunction with Mesos and Zookeeper
- Three special nodes: Chronos leader, Mesos leader, Zookeeper leader
A partition isolating Chronos from the ZK leader can cause a crash #513

aphyr opened this issue on 7 Aug 2015 · 20 comments

aphyr commented on 7 Aug 2015

When a network partition isolates a Chronos node from the Zookeeper leader, the Chronos process may exit entirely, resulting in downtime until an operator intervenes to restart it.

A partition isolating Chronos from the ZK leader can *not* cause a crash #522

aphyr opened this issue on 14 Aug 2015 · 7 comments

aphyr commented on 14 Aug 2015

Per #513, Chronos is expected to crash when a leader loses its Zookeeper connection. In this test case, Chronos detects the loss of its Zookeeper connection and, instead of crashing, sleeps quietly and reconnects when the partition heals. #513 argues that to keep running would violate unspecified correctness constraints. To preserve safety, should Chronos also crash here?

air commented on 15 Aug 2015

Hi - you're referring to a statement that doesn't represent the design (it wasn't expressed carefully enough). Please disregard it and refer to the clarification in the thread. Make sense?
Splitting Coverage

Given $n$ nodes and $k \leq n$:

- Number of partitions with $k$ blocks: $\binom{n}{k} \approx \frac{k^n}{k!}$

- Number of sets of $k$ nodes: $\binom{n}{k} \approx \frac{n^k}{k!}$

- Splitting a set with a random partition: $p = \frac{k^{n-k}}{\binom{n}{k}} \approx \frac{k!}{k^k}$

By the general theorem, there exists a $k$-splitting family of size $k^{k+1} (k!)^{-1} \log n$
Effectiveness of Jepsen

**Theorem.** For $\epsilon > 0$, a random family of partitions of size $k^{k+1} (k!)^{-1} \log n + k^k (k!)^{-1} \log \epsilon^{-1}$ is a $k$-splitting family with probability at least $1 - \epsilon$.

For Chronos, with $n = 5$, $k = 2$, $\epsilon = 0.2$: a family of 10 randomly chosen partitions is splitting with probability 80%
Other Coverage Notions

**k,l-Separation**
- Tests: Bipartitions
- Goals: Two disjoint sets of \( k \) and \( l \) nodes
- Coverage notion: The two sets included in different blocks
- Size of covering families: \( \mathcal{O}(f(k,l) \log n) \)

**Minority isolation**
- Tests: Bipartitions
- Goals: Nodes
- Coverage notion: The node is in the smaller block
- Covering families: \( \mathcal{O}(\log n) \)
Other Coverage Notions

**k,l-Separation**

- Tests: Bipartitions
- **Goals**: Two disjoint sets of \( k \) and \( l \) nodes
- **Coverage notion**: The two sets included in different blocks
- **Size of covering families**: \( O(f(k, l) \log n) \)

**Minority isolation**

- Tests: Bipartitions
- **Goals**: Nodes
- **Coverage notion**: The node is in the smaller block
- **Covering families**: \( O(\log n) \)

- **k-Splitting**, **k,l-separation**, and **minority isolation** explain most bugs found by Jepsen
Other Coverage Notions

**k,l-Separation**

- **Tests:** Bipartitions

- **k-Splitting, k,l-separation, and minority isolation**
  explain most bugs found by Jepsen

- **With high probability, O(log n) random partitions simultaneously provide full coverage for all these notions**

- **Goals:** Nodes

- **Coverage notion:** The node is in the smaller block

- **Covering families:** O(log n)
1. General Random Testing Framework

2. Randomly Testing Distributed Systems

3. Wider Context: Combinatorial Testing
General Testing Framework

1. What are tests?
2. What are testing goals?

Tests $T$

Goals $G$

3. What is the notion of coverage?
4. How to construct covering families?
Distributed Systems with Network Partitions

1. Partitions with \( k \) blocks

2. Sets of \( k \) nodes

Tests \( T \)

Goals \( G \)

3. \( k \)-splitting coverage

4. Random families of size \( O(\log n) \) are \( k \)-splitting w.h.p.
Concurrent Programs

Program = Partially ordered set of events

1. Schedules (interleavings)
2. Ordered sets of $k$ events

Tests $T$
Goals $G$

3. $k$-hitting coverage: Schedule “hits” events $e_1 < \ldots < e_k$
4. Hitting families of size $O(\log n)$, $O(\log n)^{k-1}$, $O(n^{k-1})$

Chistikov, Majumdar, Niksic. Hitting families of schedules for asynchronous programs. CAV 2016
Burckhardt et al. A randomized scheduler with probabilistic guarantees of finding bugs. ASPLOS 2010
Combinatorial Testing

3. Input coincides with the chosen values on the $k$ features
4. Various constructions of covering arrays

3. What is the notion of coverage?
4. How to construct covering families?
3. What is the notion of coverage?
4. How to construct covering families?

Where else can we apply this approach?