CLOUD9:
A SOFTWARE TESTING SERVICE

Liviu Ciortea, Cristian Zamfir,
Stefan Bucur, Vitaly Chipounov, George Candea
SOFTWARE TESTING

- Software testing is laborious and expensive
- Bugs are still very common
- Human testing is prone to errors
- Current automatic test case generation is limited
GOALS

• **Autonomy**: Minimize intervention in test generation
• **Usability**: Minimize configuration effort
• **Performance**: Maximize results relevance
OVERVIEW

• System Interface
• Parallel Symbolic Execution
• Cloud9 Design
• Preliminary Results
WHAT IS CLOUD9?

- Web service for automated testing
  - *Easy to use* interface
- Relies on thorough testing technique
  - Can operate *autonomously*
- Massive parallelization in the cloud
  - Brings scalable *performance*
Program: ReleaseCandidate.tar.gz

Testing goal: Code Coverage

Resource policy: Optimize for budget: 1000 $
Optimize for time: 5 hours

Run
SERVICE INTERFACE

Testing Report for ReleaseCandidate.tgz:

- 25 Bugs Identified
- 18 Crashes
- 7 Failed Assertions

45 Total Generated Tests
44% Total Code Coverage

Program: ReleaseCandidate.tar.gz
Testing goal: Code Coverage
Resource policy: Optimize for budget: 1000 $, Optimize for time: 5 hours
Testing Report for ReleaseCandidate.tgz:

- 25 Bugs Identified
- 18Crashes
- 7 Failed Assertions

1345 Total Generated Tests
94% Total Code Coverage

Download Test Cases
OVERVIEW

• System Interface
• Parallel Symbolic Execution
• Cloud9 Design
• Preliminary Results
void read(int x) {
    if (x < 0) {
        if (x > -3)
            foo(x);
        else {
            ...
        }
    } else {
        ...
    }
} else {
    if (x < 5)
        bar(x);
    else {
        ...
    }
}

Concrete value:

x = -2
void read(int x) {
    if (x < 0) {
        if (x > -3)
            foo(x);
        else {
            ...
        }
    } else {
        if (x < 5)
            bar(x);
        else {
            ...
        }
    }
}

Concrete value:

x = 3
void read(int x) {
    if (x < 0) {
        if (x > -3)
            foo(x);
        else {
            ...
        }
    } else {
        if (x < 5)
            bar(x);
        else {
            ...
        }
    }
}

Symbolic execution tree:

Symbolic execution tree:

\[
\begin{align*}
\lambda &< 0 \\
\lambda &> = 0 \\
\lambda &< 0 \land \lambda > -3 \\
\lambda &> = 0 \land \lambda < 5
\end{align*}
\]

\[\begin{align*}
\lambda &< 0 \land \lambda > -3 & \xrightarrow{\text{Constraint solver}} & \lambda = -2 \text{ (a solution)}
\end{align*}\]
PATH EXPLOSION

Large memory consumption
CPU Intensive
Large memory consumption
CPU Intensive
We massively parallelize in the cloud
CHALLENGES

- Tree structure is not known a priori
CHALLENGES

- Tree structure is not known a priori
- *Naive approach:* pre-allocate workers equally on the tree
CHALLENGES

- Tree structure is not known a priori
- *Naive approach*: pre-allocate workers equally on the tree
- *Slightly better*: examine the first $h$ levels, then decide work allocation
CHALLENGES

- State transfer
- Avoid work and memory redundancy
- Coordination
OVERVIEW

- System Interface
- Parallel Symbolic Execution
- Cloud9 Design
- Preliminary Results
TREE EXPLORATION

Exploration frontier

Exploration strategy = which node to expand next?
Exploration strategy = which node to expand next?
LOAD BALANCING
LOAD BALANCING

negotiate
LOAD BALANCING
STATE TRANSFER DECISIONS

• State copying vs. state reconstruction

• Reconstruction optimizations

Copying
(Network intensive)

Reconstruction
(CPU intensive)
STRATEGY PORTFOLIO

• No "one size fits all" exploration strategy

• Different workers with different strategies

• Invest in few workers, then select successful methods
OVERVIEW

• System Interface
• Parallel Symbolic Execution
• Cloud9 Design
• Preliminary Results
CLOUD9 PROTOTYPE

• We built Cloud9 on KLEE
  • State-of-the-art sequential symbolic execution engine
  • Tested real programs and found bugs
• Use Amazon EC2 as cloud computing platform
TESTING METHODOLOGY

- We compare with KLEE for testing Coreutils
  - ls, cat, chmod, cp, mv, etc.
- Cloud9 and KLEE run for **1 hour**
- **16 workers** for Cloud9
CLOUD9 SPEEDUP

Fix code coverage and measure time

Tool

Speedup ($t_{\text{KLEE}}/t_{\text{Cloud9}}$)

(Linear) 16
Fix resources (CPU time) and measure code coverage
CONCLUSIONS

✓ Autonomy:
  • Symbolic execution

✓ Usability:
  • Web service interface
  • No local setup overhead

✓ Performance: up to 250x speedup
  • Parallel symbolic execution
  • Dynamic load balancing
  • Adaptive state transfer
  • Strategy portfolio