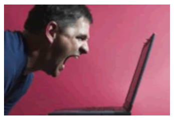
AutoBash: Improving Configuration Management with Operating System Causality Analysis

Ya-Yunn Su, Mona Attariyan, and Jason Flinn
University of Michigan

Presented by: Fahad Arshad

Motivation

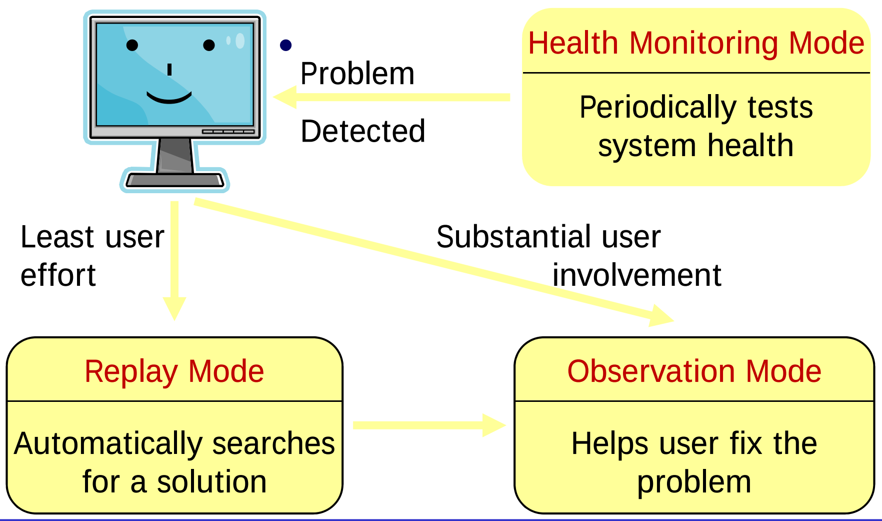
- Configuration management is frustrating!
- Users may have to
  - Change environment variables
  - Edit configuration files
  - Manage inter-application dependencies
- Current approach:
  - Ask friends, search on-line, read manual, …
  - Try potential solutions
  - Carefully undo wrong solutions



Problems with current approach
AutoBash solves these problems

- Applying solutions is time-consuming
  Automatically tries many solutions
- Undoing a wrong solution can be hard
  Provides undo capability
- Hard to know how a problem was solved
  Explains solution to user
- A "solution" may cause new problems
  Automatically runs regression tests

AutoBash overview

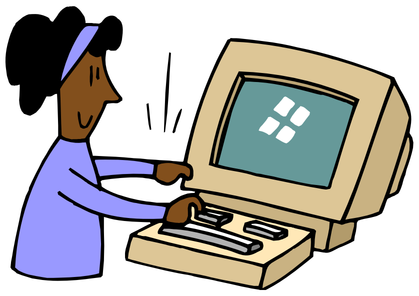

Problem
Detected
Health Monitoring Mode
Periodically tests system health
Least user effort
Substantial user involvement
Replay Mode
Automatically searches for a solution
Observation Mode
Helps user fix the problem

Outline

- Motivation
- AutoBash design and implementation
  - Observation mode
  - Replay mode
  - Health monitoring mode
- Evaluation
- Conclusion

Observation mode

- A modified bash shell
  - User types in commands to solve the problem



```
% command 1
% test if app works
% undo testing
% undo command 1
% command 2
```

Verifying a solution is tedious

- AutoBash automatically tests using *predicates*
- Predicate:
  - Tests if an application functions correctly
  - Returns true/false if the test passes/fails

```
% command 1
~~% test if app works~~
% undo testing
% rollback command 1
% command 2
```

wge … USER

Predica … o server

Undoing testing is tedious

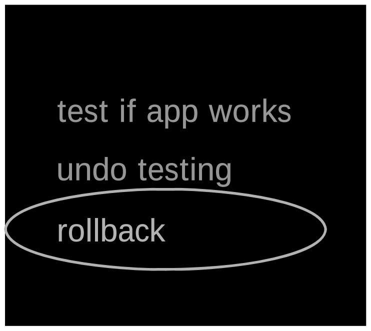
- Predicate testing has no side effects
  - Executed speculatively and rolled back
- Speculator [SOSP '05]
  - Process-level speculative execution

```
% command 1
% test if app works
~~% undo testing~~
% rollback command 1
% command 2
```

- Speculative … dicate testing safe

Undo can be hard

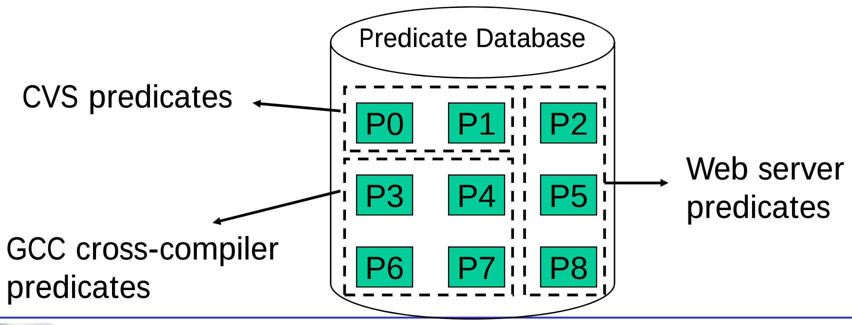
- AutoBash speculatively executes each action
  - Light-weight checkpoint and rollback



- Speculative execution makes undo easy

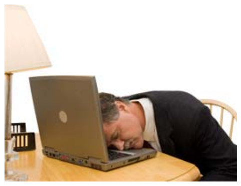
Regression testing is hard

- AutoBash automatically runs regression tests
  - Executes predicates in the predicate database
  - Ensures all predicates pass



Regression tests can be slow

- Problem: running all predicates can be slow



- Only need to run predicates affected by an action
  - Uses causality tracking to find affected predicates

PURDUE UNIVERSITY

Tracking causality

- Output set
  - kernel objects an action causally affects

Action: touch foo

Output set = {file foo}

- Input set
  - kernel objects a predicate causally depends on

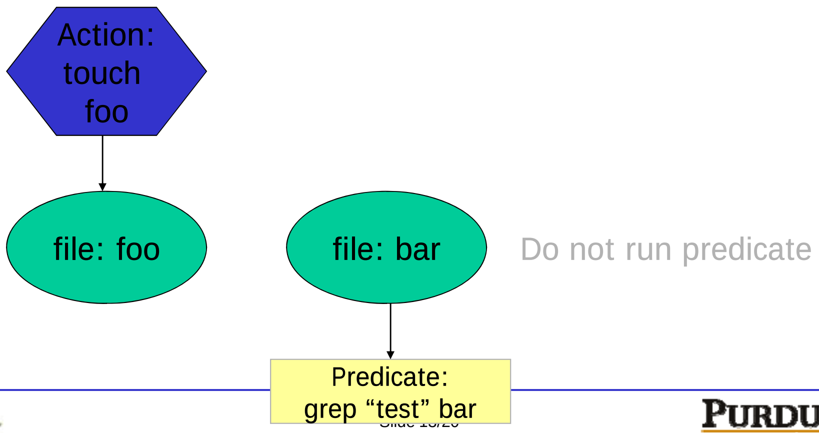
Predicate: grep "test" bar

Input set = {file bar}

PURDUE UNIVERSITY

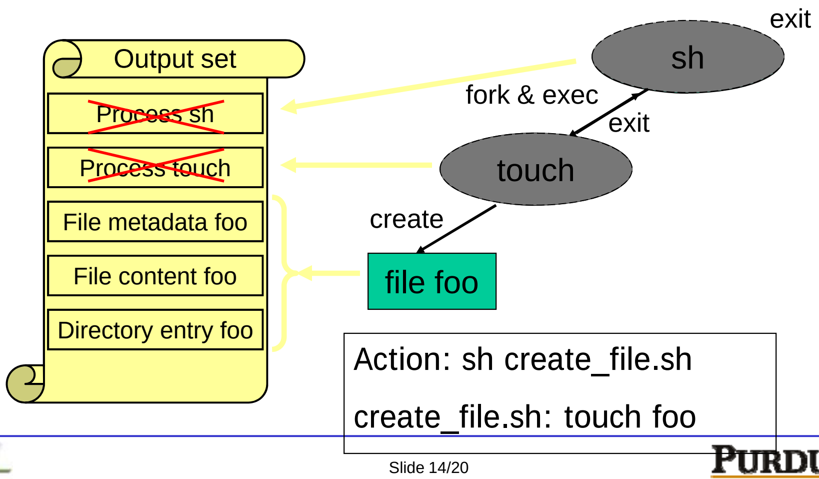
Analyzing causality

- AutoBash calculates the intersection
  - Determines which predicates to run



Tracking output sets

- An output set is tracked for each action

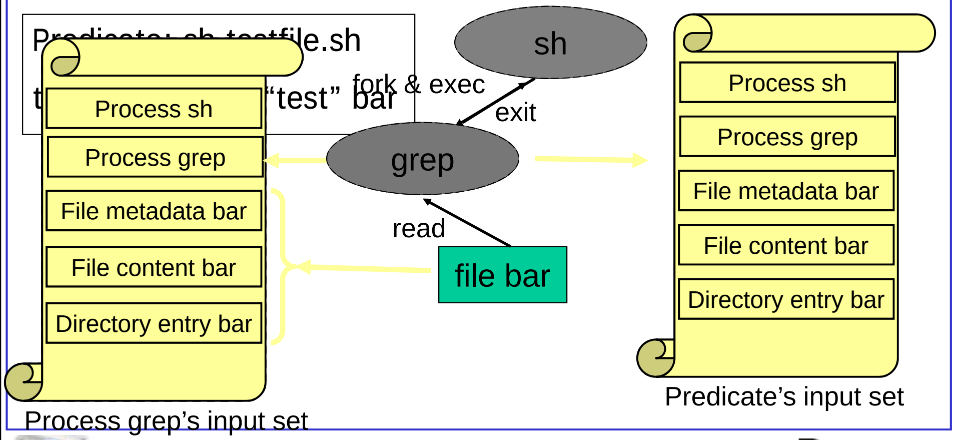


Action: sh create_file.sh

create_file.sh: touch foo

Tracking input sets

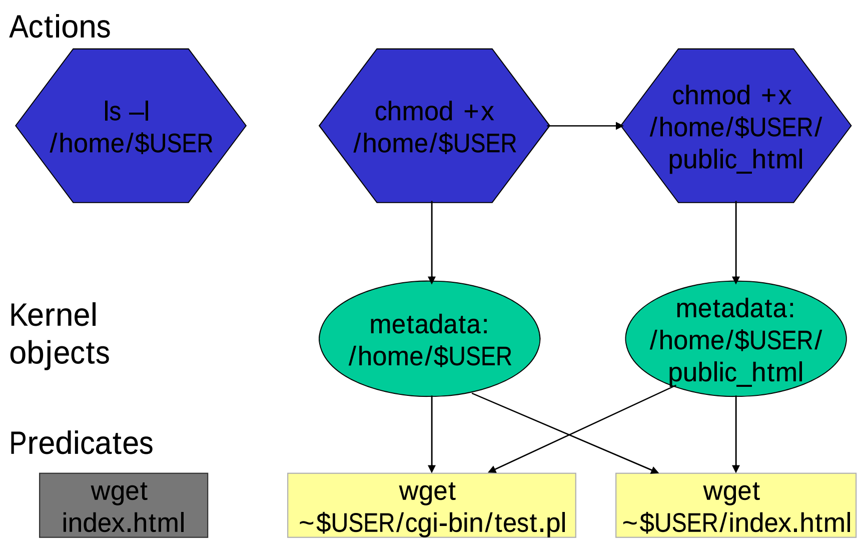
- An input set is tracked for each predicate



Understanding solutions can be hard

- AutoBash generates causal explanation
  - Analyzes input and output sets

Causal explanation



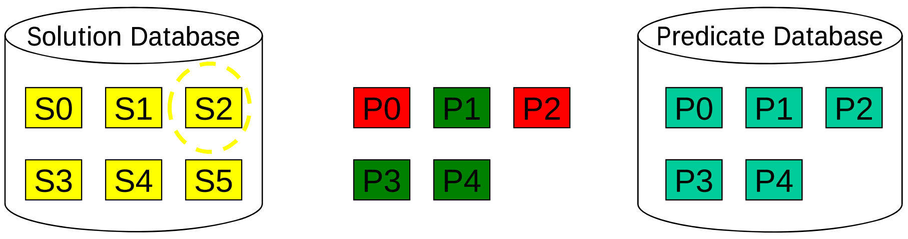
Outline

- *Motivation*
- AutoBash design and implementation
  - *Observation mode*
  - Replay mode
  - Health monitoring mode
- Evaluation
- Conclusion

Replay mode

- Problem: finding a solution is time-consuming
- Automatically searches for a solution
  - No user input needed
- Speculative execution provides isolation
  - User continues foreground task
  - AutoBash runs replay mode in background

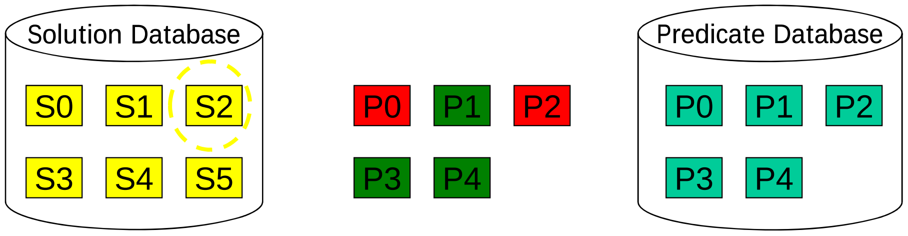
How replay mode works



(1) Initial predicate testing:

- Tracks input set for each predicate
- Determines passed/failed predicates

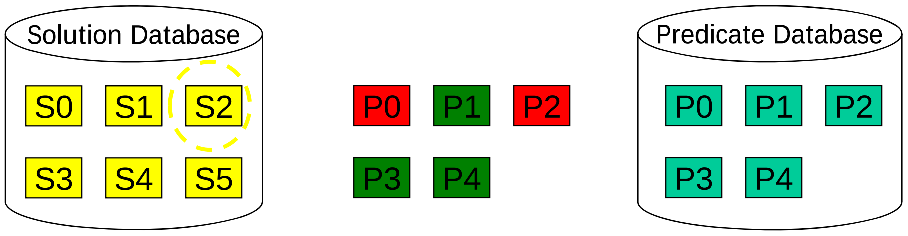
How replay mode works

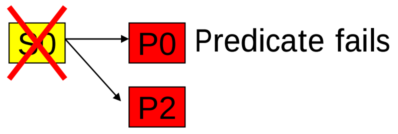


(2) Solution execution:

- Speculatively executes a solution
- Tracks solution output set

How replay mode works

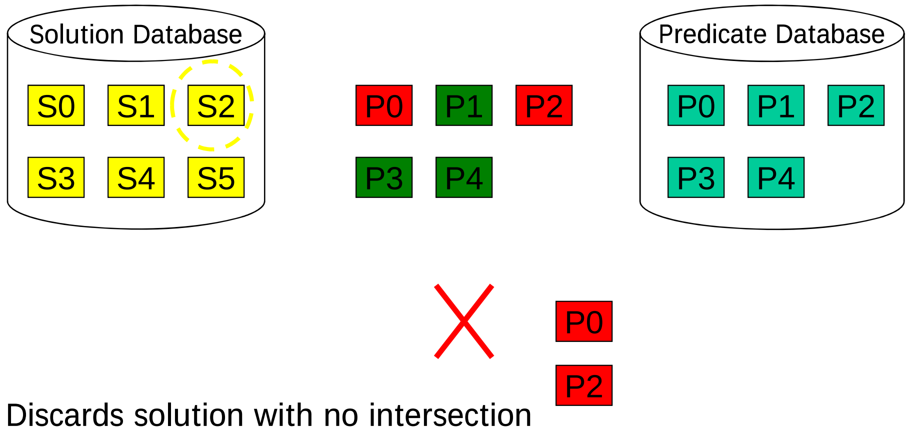




(3) Verifying solution:

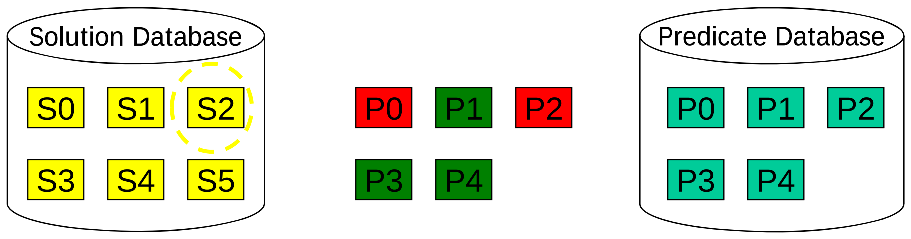
- Calculates intersection
- Runs predicates with intersection

How replay mode works



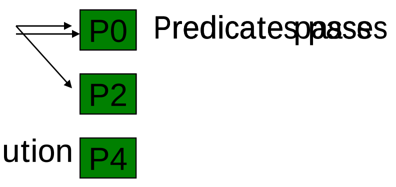
Discards solution with no intersection

How replay mode works

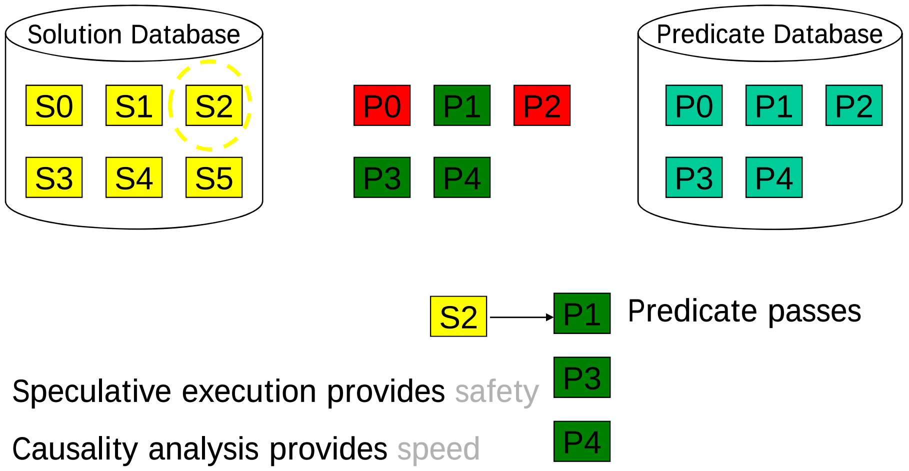


(4) Regression tests:

- Calculates intersection
- Runs predicates affected by solution



How replay mode works



- Speculative execution provides safety
- Causality analysis provides speed

Health monitoring mode

- Periodically executes all predicates
- If any predicate fails, AutoBash
  - Runs replay mode to search for a solution
  - Reports to the user to run observation mode

Outline

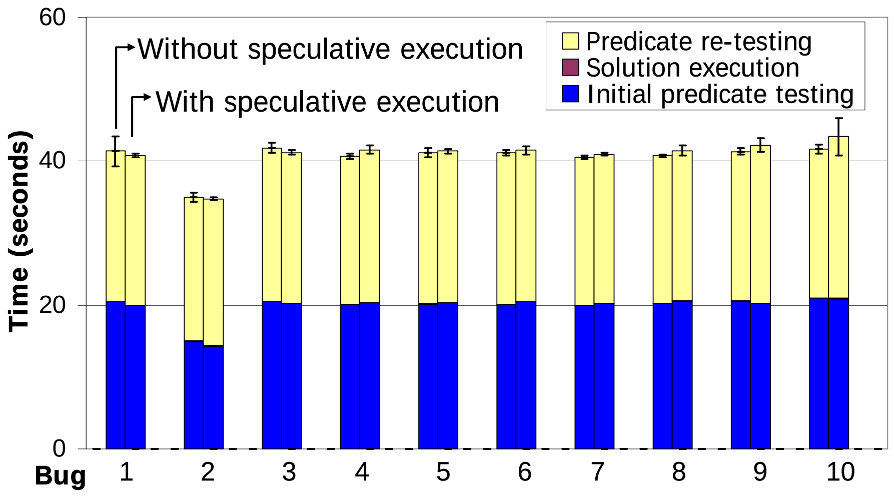
- *Motivation*
- *AutoBash Design and Implementation*
  - *Observation mode*
  - *Replay mode*
  - *Health monitoring mode*
- Evaluation
- Conclusion

Evaluation

- Questions:
  - What is the overhead of speculative execution?
  - How effective is causality analysis?

- Methodology:
  - Evaluated CVS, gcc cross compiler, web server
  - Manually created 10 bugs and 10 solutions
  - Manually created 5-8 predicates

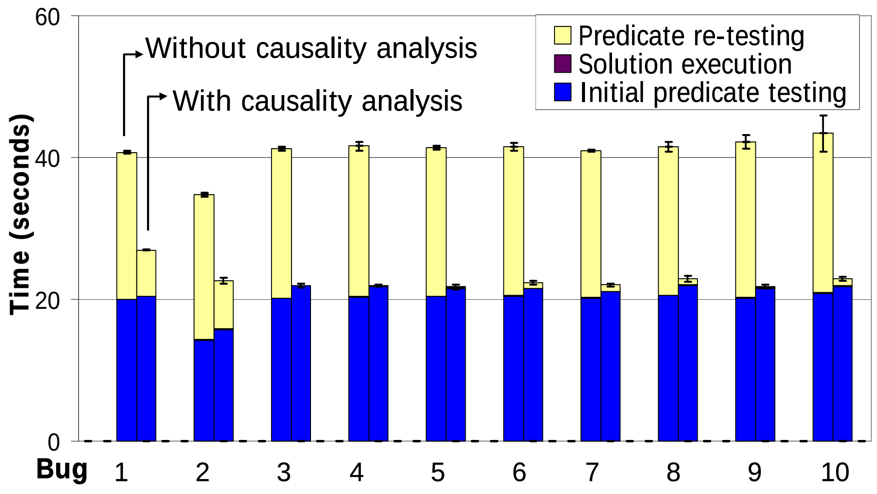
Total replay time (GCC)



▪ Speculative execution overhead is negligible

PURDUE UNIVERSITY

Total replay time (GCC)



▪ Causal analysis improves predicate re-testing time by 67-99%

PURDUE UNIVERSITY

Conclusion

- Configuration management is frustrating
- AutoBash automates most tedious parts
- Speculative execution makes AutoBash safe
- Causality analysis makes AutoBash fast