NICTA

Supporting Undoability in System Operations

I. Weber, H. Wada, A Fekete, A. Liu and L. Bass

Australian Government
Department of Broadband, Communications and the Digital Economy
Australian Research Council

NICTA Funding and Supporting Members and Partners

Australian National University
UNSW The University of New South Wales
NSW Government Trade & Investment
State Government Victoria
The University of Melbourne
The University of Sydney
Queensland Government
Griffith University
QUT Queensland University of Technology
The University Of Queensland Australia

System operation could go wrong

- Operation errors as the largest contributor to system failures [1]
- Configuration errors (28%) and human errors (13%) cause service disruptions [2]
- System operation plays a large role in overall system reliability

[1] D. Oppenheimer, et. al., "Why do internet services fail and what can be done about it?," USENIX, 2003
[2] L. Barroso, et. al., "The Datacenter as a Computer," 2013

Cloud increases the problem

NICTA

- More and more services operate on API-controlled infrastructures
  - e.g., IaaS cloud
  - Enabled frequent changes - increase chance of error

- Activities are performed without a safety net
  - Deleting a virtual disk results in a total loss
  - Stopping a VM changes relations to other resources. Not obvious how to fix it.

Project Goal and Scope

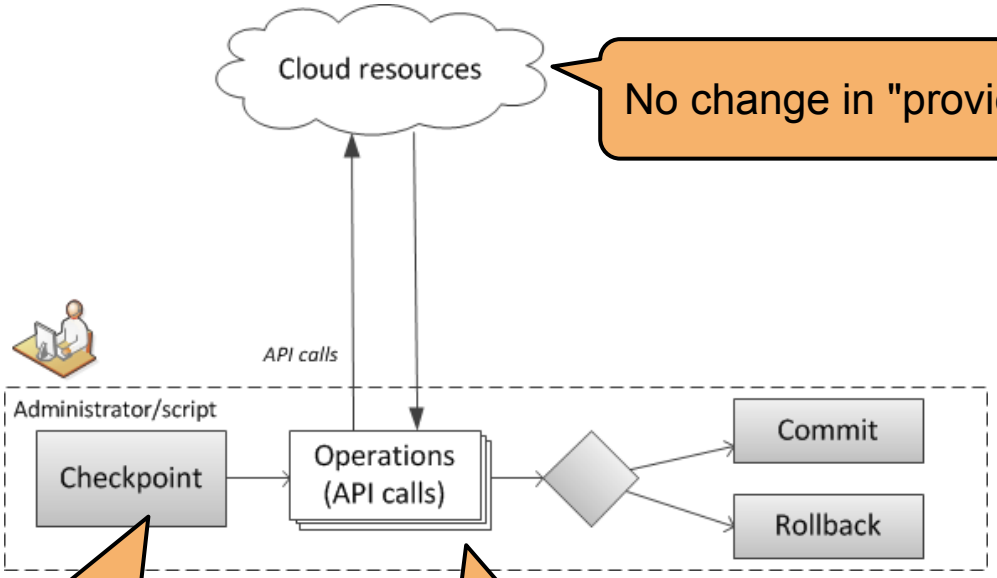
- Provide an "undo button" on Amazon Web Service
  - Facility to rollback changes is valuable support for dependability in various areas
  - Introduce "atomicity" in system operation

- Allow for restoring the state of cloud resources such as VM, disk volume, IP address and the structure

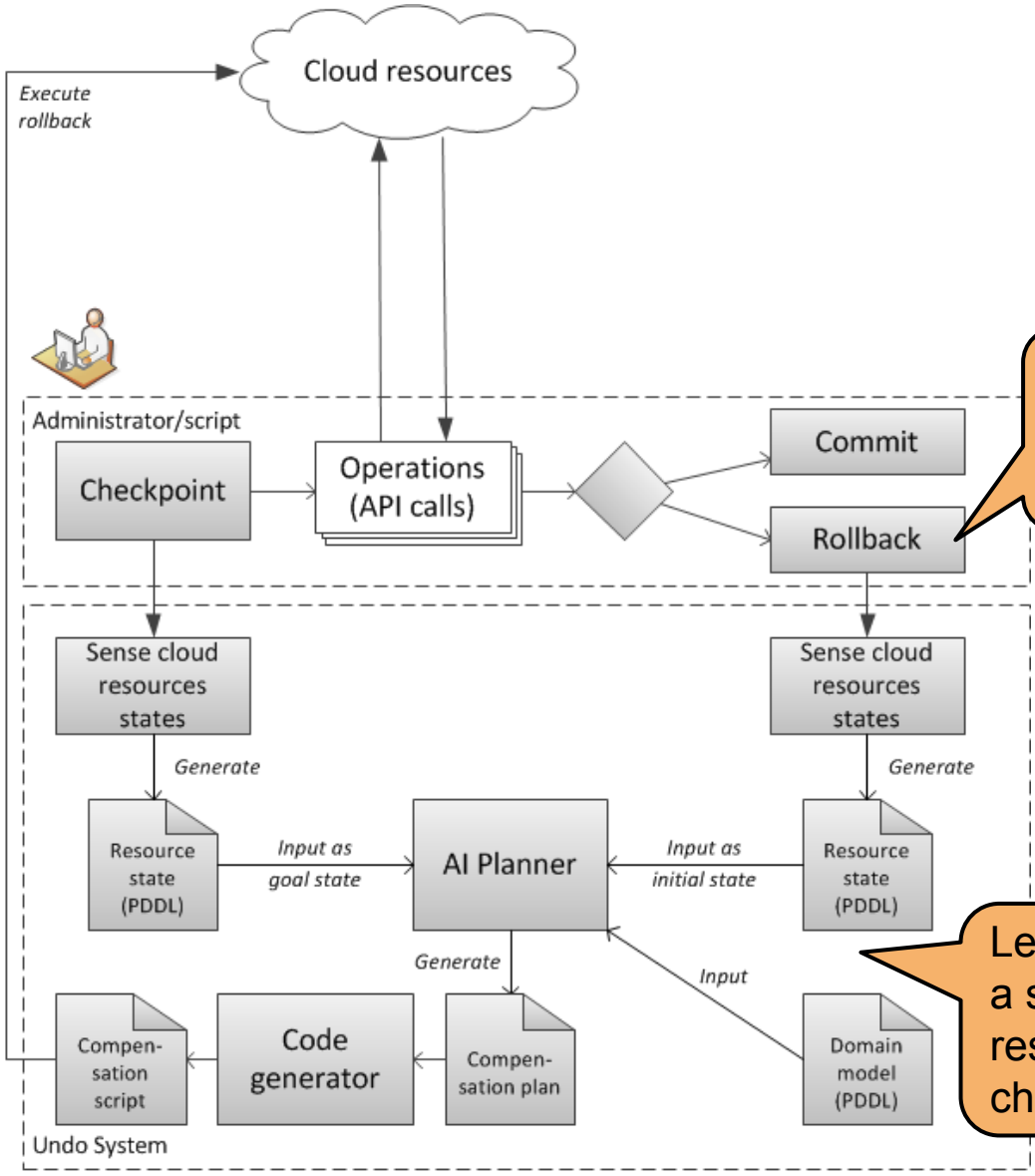
Challenges

- Not possible to "copy back snapshot" directly
  - Can only manipulate through *given API*
  - Need to find a right sequence of operations
- "Revert one-by-one" may not work
  - Constraints among operations/resources
  - Error-prone operations require "backup" plans
- Completely non reversible operations
  - e.g., deleting resources
- Partly reversible operations
  - Not all resource properties can be restorable, e.g., id, creation timestamp, ...

Our Approach: Checkpoint



Our Approach: Rollback by AI Planning



"Rollback" command triggers system restoration

Leverage AI planning to find a sequence of operations to restore the state at the checkpoint

Why AI Planning?

NICTA

- Traditional techniques to rollback long-running transactions do not apply or are sub-optimal:
  - There may be no direct reverse operation
  - More than one operations required to reverse one
  - Hand-coding for all possible cases is tedious

- AI Planning:
  - Given start state, goal state, set of actions, searches a solution in the state of possible plans
  - Our variant finds maximal contingency plans
    - If one action fails, but the goal is still reachable, a backup plan will be executed

Domain Model: Supported Resources

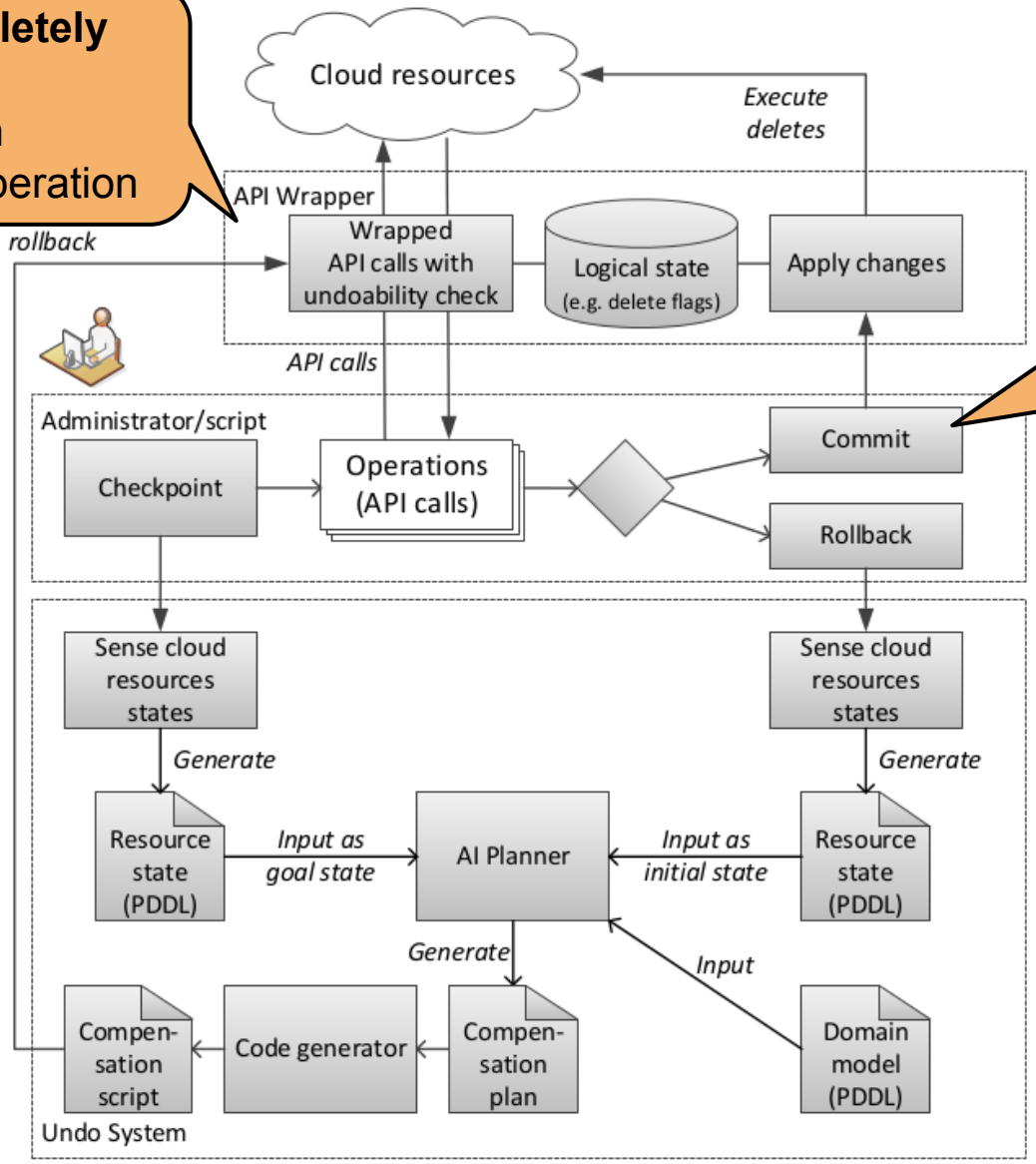
NICTA

Resource Type in AWS	API operations
Virtual machine	launch, terminate, start, stop, change size
Disk volume	create, delete, create-from-snapshot, attach, detach
Disk snapshot	create, delete
Elastic IP address	allocate, release, associate, disassociate
Security group	create, delete
Autoscaling group	create, delete, change size, change config, create config, delete config
Tag	create, delete

Our Approach: Wrap non reversible ops

NICTA

Replace **completely** non reversible operations with "pseudo-do" operation



"Commit" command applies changes

I. Weber, et. al., "Automatic undo for cloud management via AI planning," HotDep 2012.

Partly reversible operations

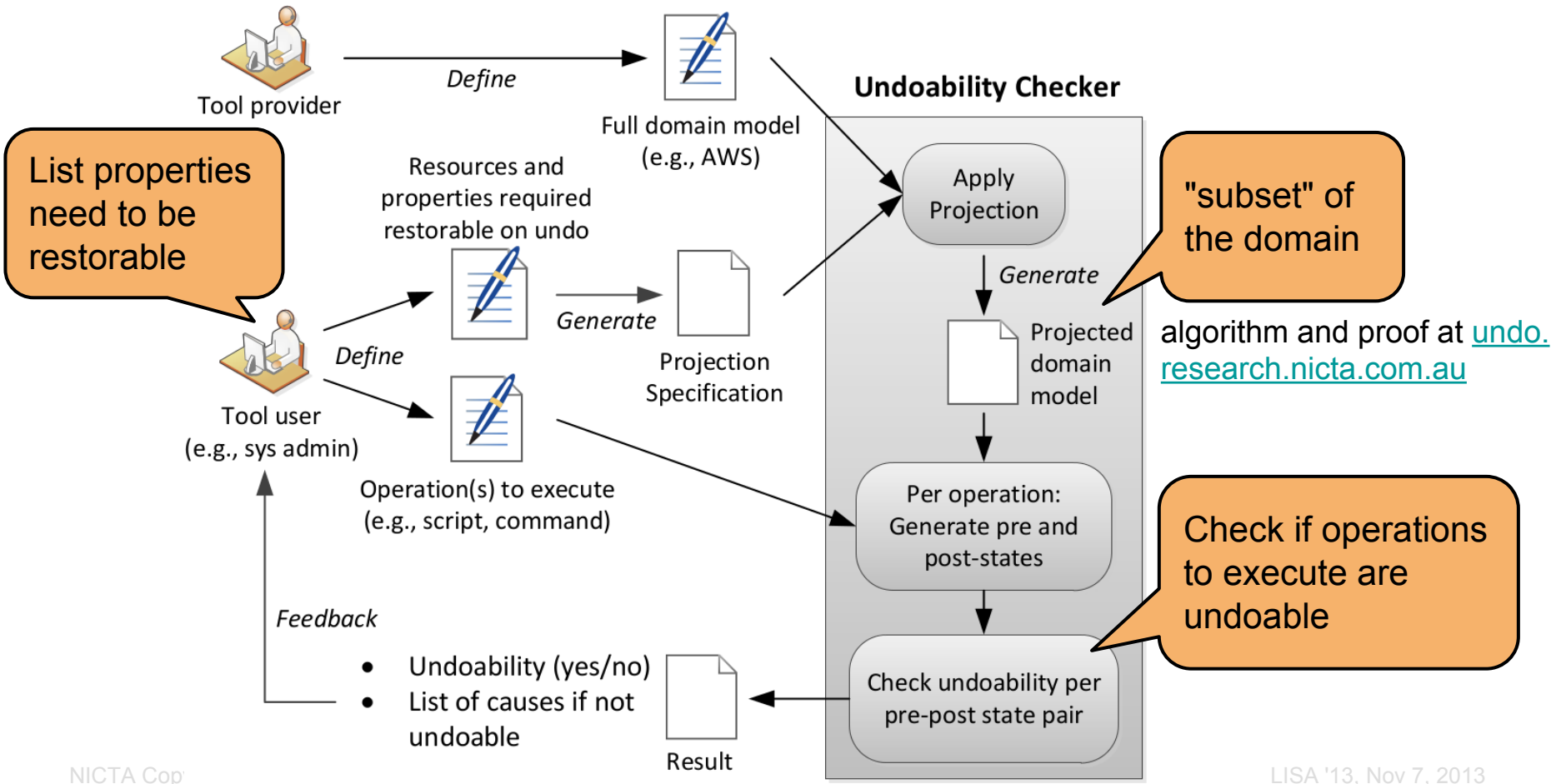
- Many operations seem to be undoable - but **not**

- e.g., starting a VM is **NOT** reverse of stopping it
  - Properties cannot be restorable: resource id, dynamic IP address, or creation timestamp

- Restoring **ALL** properties is not feasible for many scenarios; however, it may not be required
  - e.g., undoing "change the size of the web layer" can safely ignore changes to the creation timestamps

Our Approach: Undoability Checking

- Facility to examine whether a collection of operations to execute are undoable



algorithm and proof at undo.research.nicta.com.au

Two Usage Models of Checker

- Offline: check the undoability of all operations to execute under any situations
  - Much stronger than needed since it checks situations that may not occur
    - e.g., delete a volume if resource X is in state Z

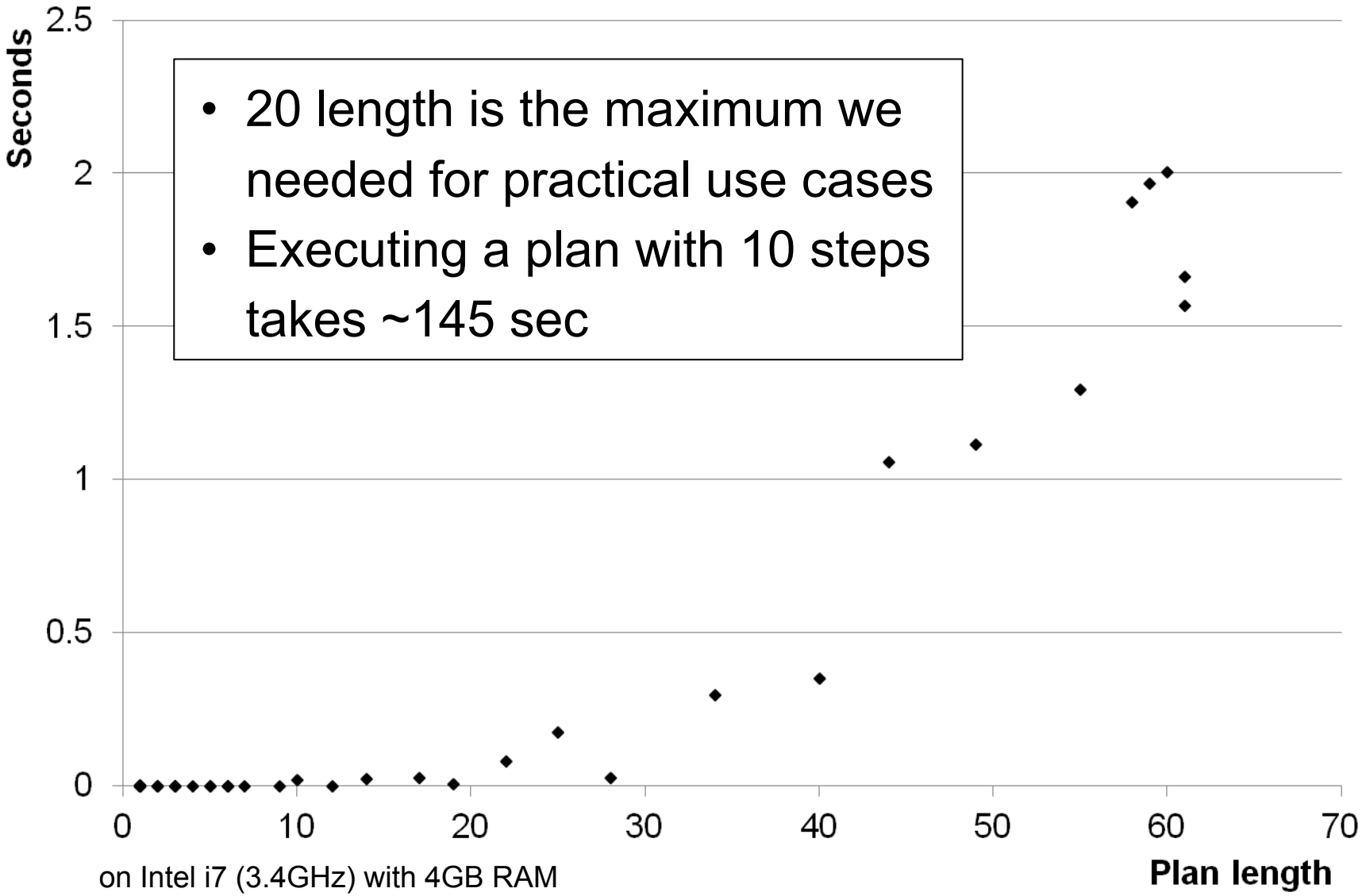
- Online: check the undoability of an operation before execution under the current situation
  - Check the undoability exactly as needed
  - Performance penalty do to state sensing

Evaluation

- Evaluate prototype
  - Undo engine released at undo.research.nicta.com.au
- Execution time to search undo plans
  - Use FF [3] with an extension for planning
- Execution time to check undoability of a domain (offline)
- Validation with real-world scenarios

[3] J. Hoffmann, et. al, "The FF planning system: Fast plan generation through heuristic search," J. of Artifical Intelligence Research, 14 (2001)

Evaluation: planning performance



I. Weber, et. al., "Automatic undo for cloud management via AI planning," HotDep 2012.

Evaluation: undoability check

NICTA

- Checking a full domain (35 actions)
  - Projection: remove unrecoverable errors and few properties, and adding pseudo-delete and undelete
  - 11.0 seconds to check 1330 planning problems
  - Result can be reusable unless the projection changes

- Examine our day-to-day manual tasks
  - e.g., adding a slave to a database server, expand the disk size, scaling up/down web layer, and upgrading app layer
  - Confirmed the undoability given a domain projection

Conclusion

- Facility to offer "undo" and check undoability
  - Provide rollback on a set of API operations
  - Check if operations are undoable under in certain context (projected domain)
- Prototype shows the execution time is marginal and the algorithm scales well
  - http://undo.research.nicta.com.au
- Future work
  - Capture internal resource state
  - Parallelizing plans to speed up execution

Appendix

Domain Model: Example

NICTA

- Formally captures operations in domain
  - e.g., a set of operations that AWS supports
- Action to delete a disk volume in PDDL

```
(:action Delete-Volume
 :parameters (?vol - tVolume)

 :precondition
 (and
   (volumeAvailable ?vol)
   (not (unrecoverableFailure ?vol)))

 :effect
 (oneof
   (and
     (deleted ?vol)
     (not (volumeAvailable ?vol)))
   (unrecoverableFailure ?vol)))
```

Volume to be deleted must be available

Volume will become deleted and unavailable, or failure

Domain Projection and Undoability Check

NICTA

- Domain projection
  - Remove properties not required to be restorable
  - Add pseudo-delete for each delete operations

- Undoability checking algorithm

```
foreach operation op to execute:
  foreach pre-state:
    find possible post-states by applying op:
    foreach post-state:
      let AI planner finds a plan from pre to post-state
      if no plan found
        op is not undoable
```

  - pre-states is an infinite set. We obtain sufficient pre-states from domain model