Automatic Undo for Cloud Management via AI Planning

Ingo Weber, Hiroshi Wada, Alan Fekete, Anna Liu and Len Bass

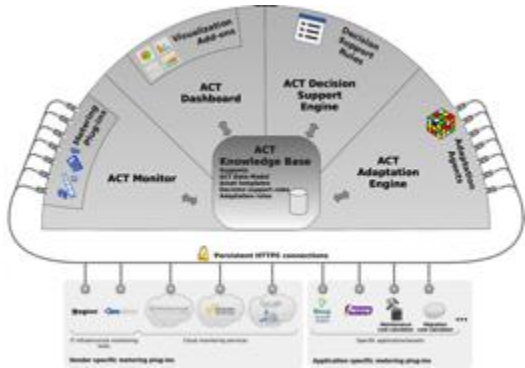
Based on presentation at USENIX HotDep '12

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

NICTA in Brief

- Australia's National Centre of Excellence in Information and Communication Technology
- Five Research Labs:
  - ATP: Australian Technology Park, Sydney
  - NRL: UNSW, Sydney
  - CRL: ANU, Canberra
  - VRL: Uni. Melbourne
  - QRL: Uni. Queensland and QUT
- 700 staff including 270 PhD students
- Budget: ~$90M/yr from Fed/State Gov and industry
- ~600 research papers/year, ~150 patents total



Yuruware our spin-out

NICTA

Bolt
your cloud deployment

Monitor
your cloud applications

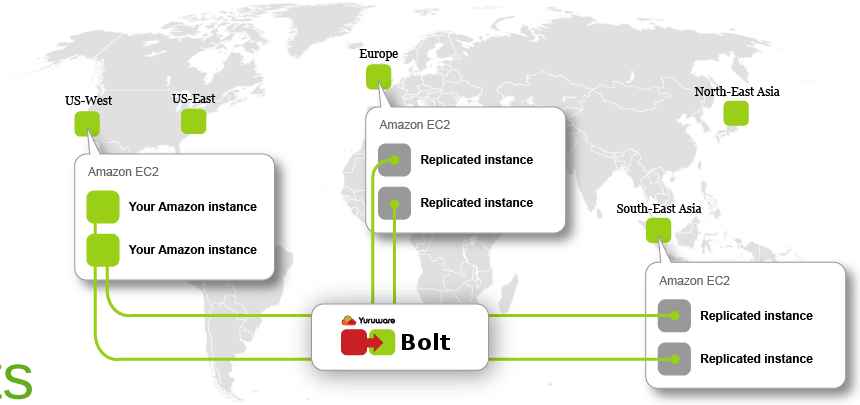
Compare
your cloud services

Clip
your idle resources

Yuruware can help…	Bolt	Monitor	Compare	Clip
Build business continuity	■			
Compare providers			■	
Monitor multiple clouds		■		
Find idle resources				■

Motivation of this work

- Yuruware Bolt: site-replication across regions
  - Executes long-running operations - create, delete and update variety resources via AWS API

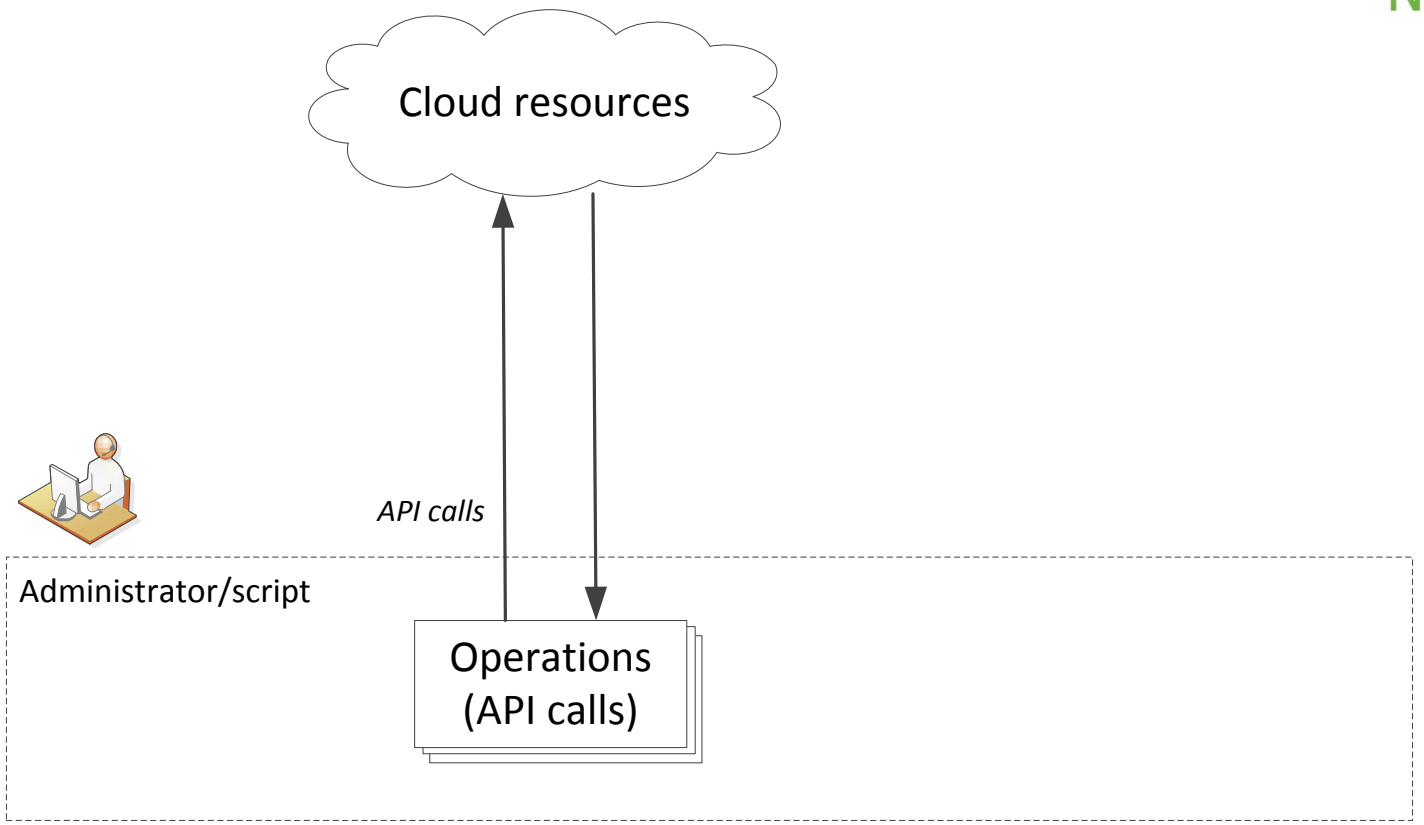


- Issues we face
  - High cost of writing unit tests
    - Preparing a test bed, reset after each test, and error recovery
    - “Why there is no DBUnit for cloud!”
  - High cost of error handling
    - Resources often get stuck in an unexpected state
    - Well-coordinated and tailored clean up for each case

Our Goal

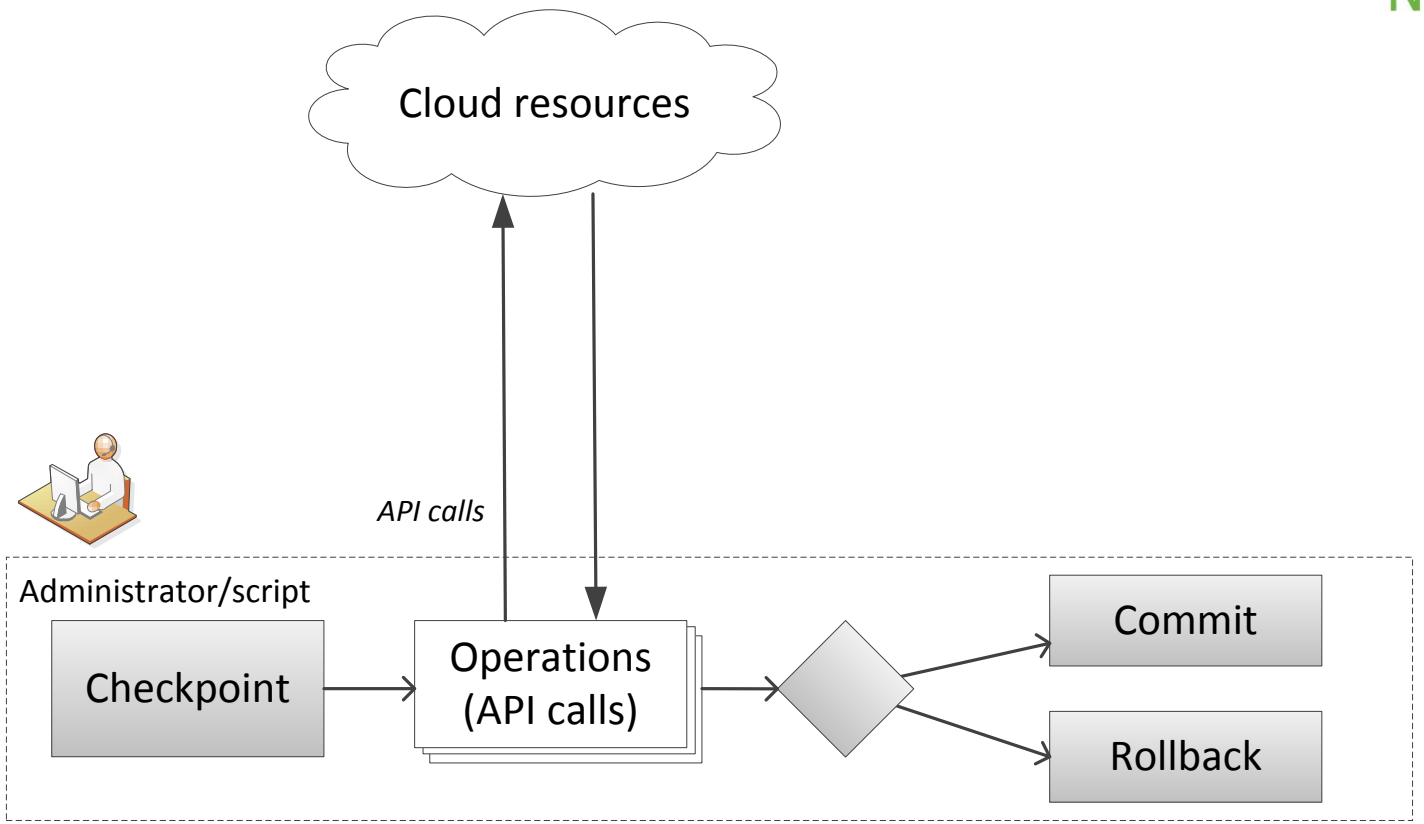
- Provide an “undo” button to cloud users
  - Allow for rolling back to a previous state
  - e.g., undelete deleted resources and reconstruct the relations among resources
- We’re users - cannot alter API or implementation
  - Some operations are undoable, e.g., detach a VIP
  - Some operations are semi-undoable, e.g., stop a VM
  - Some operations are not undoable, e.g., delete a disk
- Less invasive
  - Minimum changes in existing code or scripts

Status quo



- Administrators and/or scripts talk to cloud via API

Goal

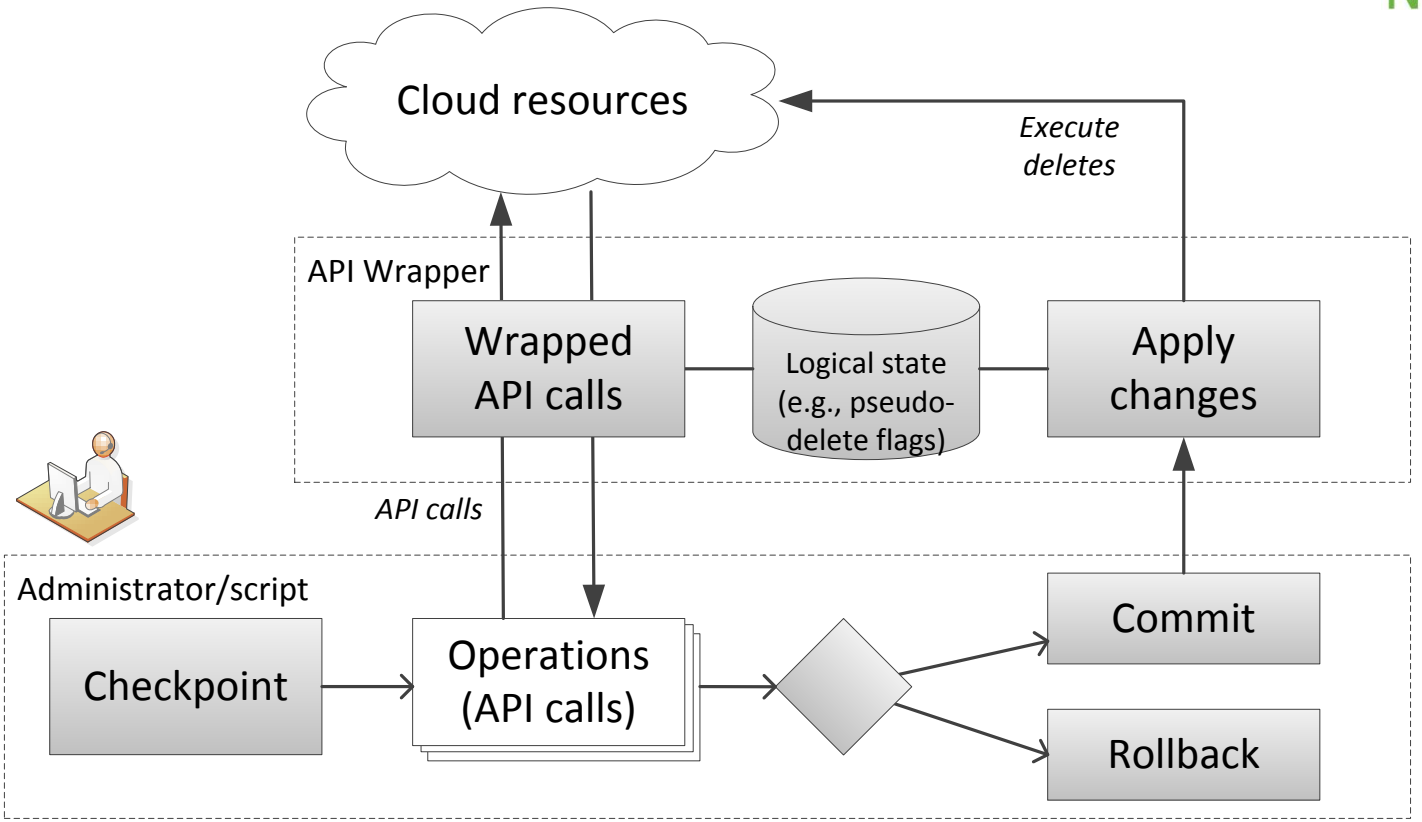


- Provide the ability to go back to a checkpoint
- No change in scripts except checkpoint and commit

Our approach

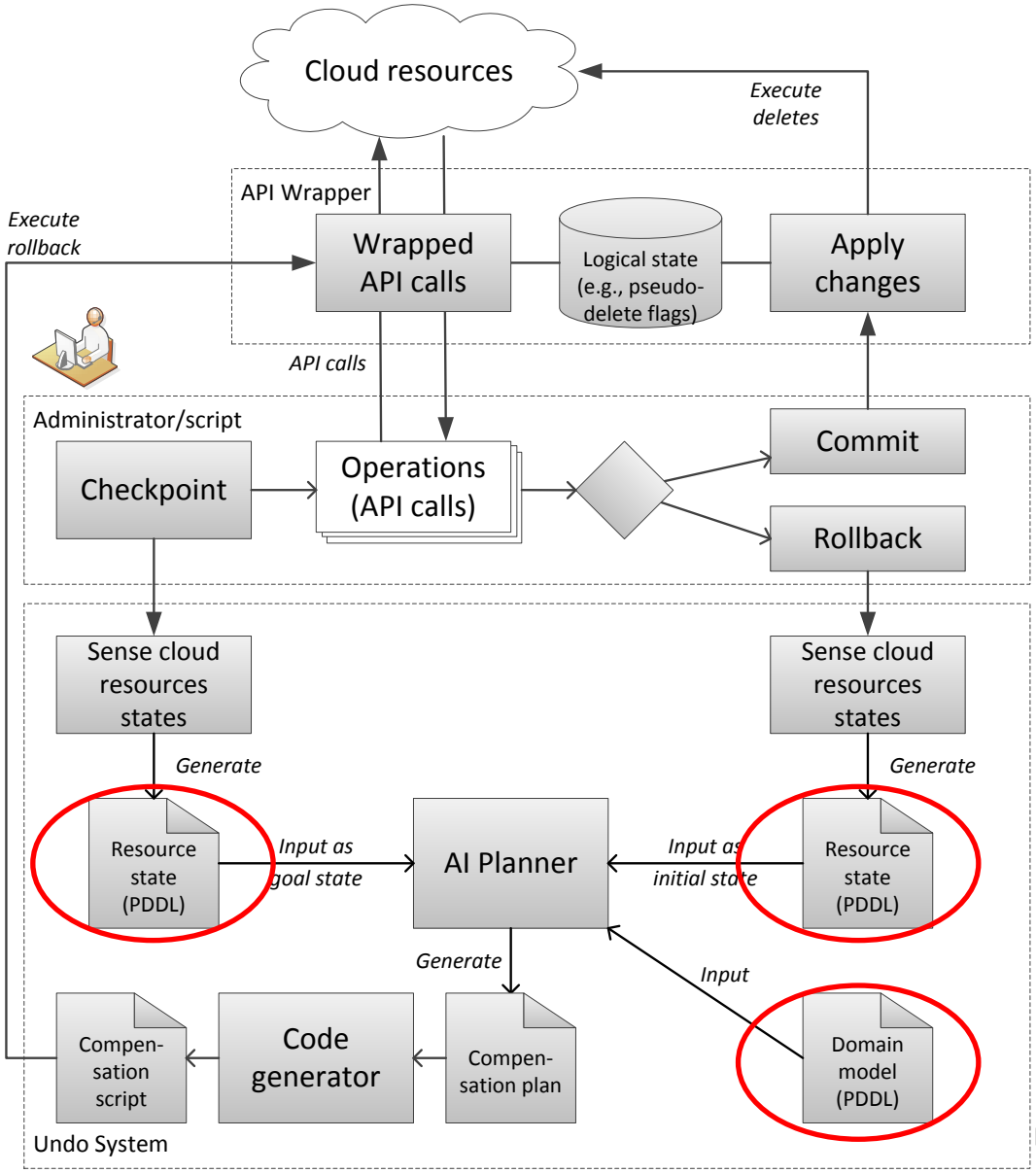
- “undo one by one in reverse order“ does not always work
  - No undo action is available
    - e.g., no “undeleteing a deleted resouce “
  - Undo requires a different sequence of API calls
    - e.g., “deleteing an autoscaling group“ does not work
  - Simple undo could results in a different state
    - e.g., Undo “stopping an instance with a VIP“
  - Undo operation may fail
    - e.g., detaching a volume could fail. Need an alternative.
  - Not optimal
    - e.g., Bolt‘s operations (creating many temporary resouces)
- Tedious to hand-code for all possible cases!

Pseudo-delete for not-undoable ops



- Execute API calls if they are undoable
- Defer the execution of non-undoable calls until commit

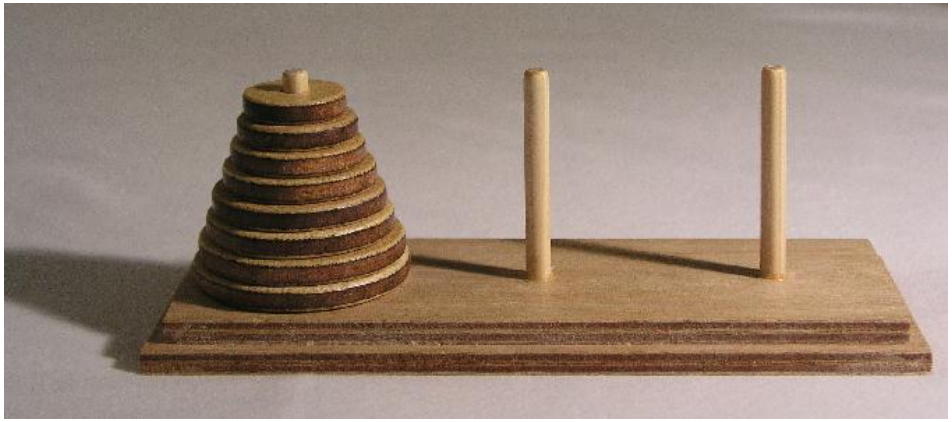
AI planning for the rest of cases



- Changes made by (semi-)undoable API calls are compensated by an AI planner
- AI planner finds ways to handle errors potentially occur during undo as well

AI Planning 101

- Given the initial state of the world, goal state, and a set of available actions, find a sequence of actions that leads from the initial to the goal



http://en.wikipedia.org/wiki/Tower_of_Hanoi

- Highly optimized heuristics tame the problem for practical purposes

Planning under uncertainty

- In our problem
  - Initial state: state at rollback
  - Goal state: state at checkpoint
  - Actions: AWS APIs
- We use FF [*] with an extension to handle actions with alternative outcomes
- Finds “maximal“ contingency plans
  - e.g., if detachnig a volume fails, stop the attached instance if possible. If a planner cannot solve, ask human intervention

[*] H OFFMANN , J., and N EBEL , B. “The FF planning system: Fast plan generation through heuristic search.” Journal of Artificial Intelligence Research, 14 (2001),

Domain model: example

- 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)))
```

Domain model: actions

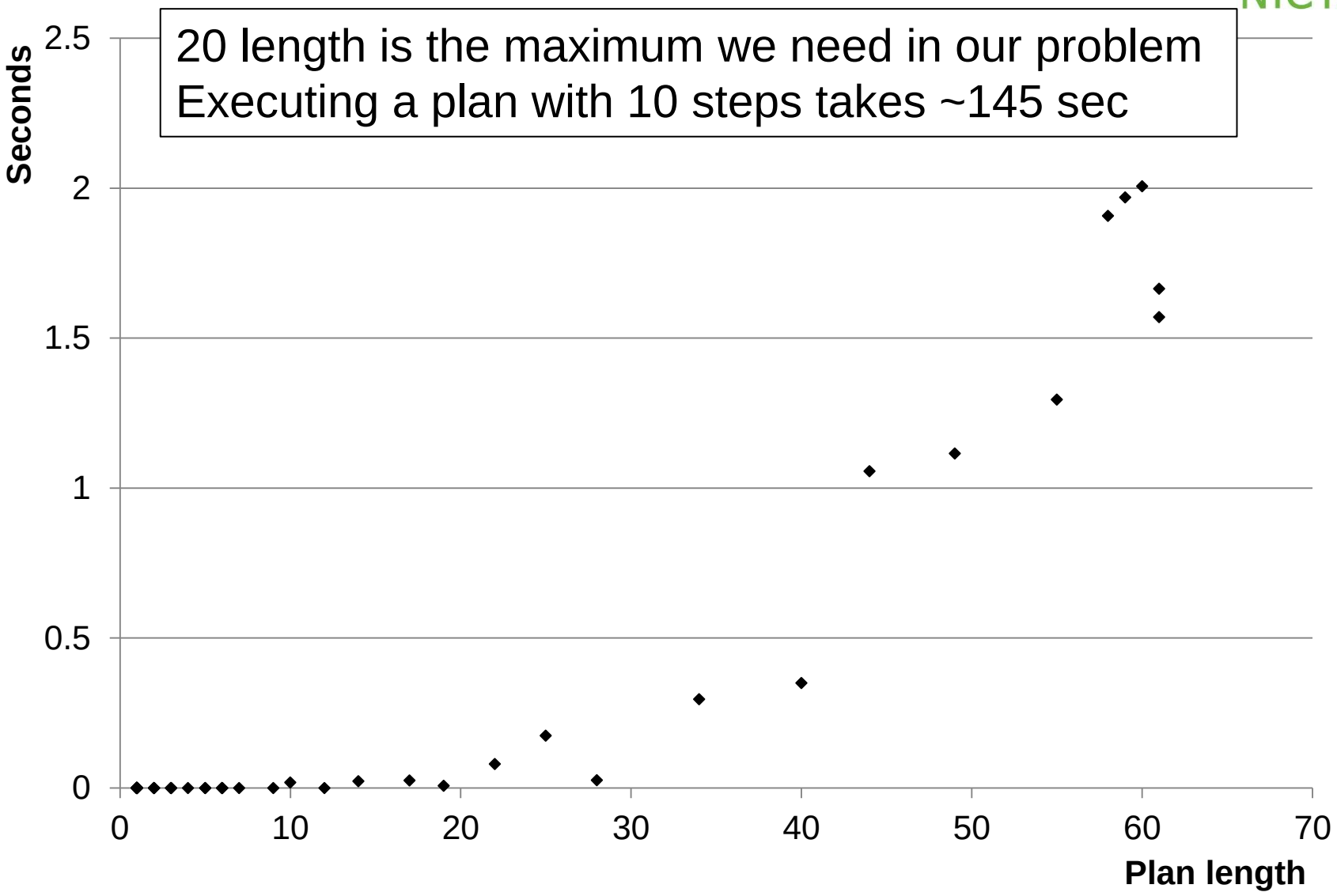
Resource type	Actions
Virtual machine	launch, terminate, start, stop, change VM size
Disk volume	create, delete, create-from-snapshot, attach, detach
Disk snapshot	create, delete
Elastic IP address	allocate, release, associate, disassociate
Security group	create, delete
Auto-scaling group	create, delete, change-sizes, change-launch-config
Auto-scaling launch config	create, delete
Tag	create, delete
Others	AZ, cluster online/offline, ...

Evaluation

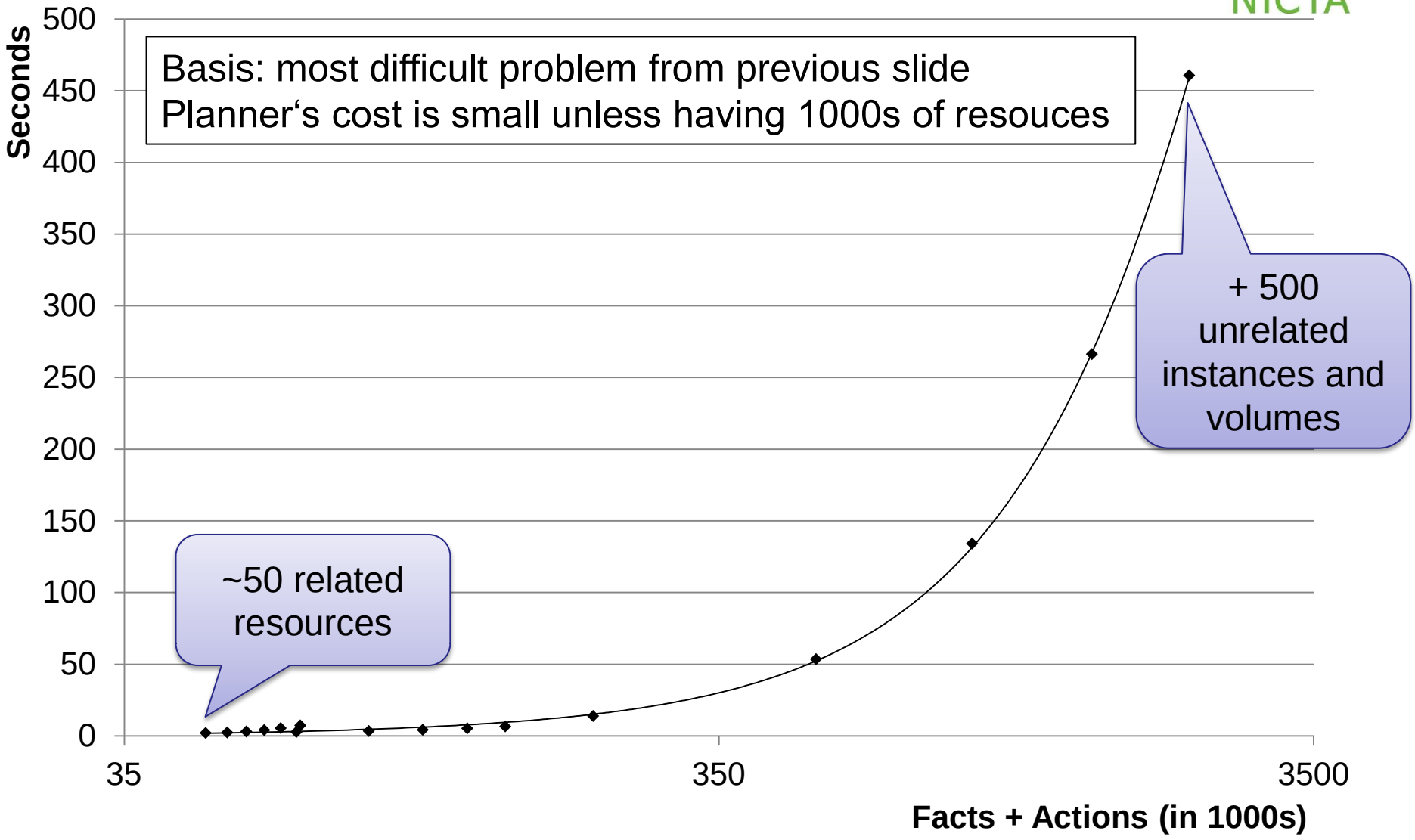
- Scalability of the planner based on an internally released prototype
  - AWS cmd line tool replacement
- Use cases: ~70 different planning settings tested

- Evaluation 1: against plan length
- Evaluation 2: against # of unrelated resources

Evaluation 1: vs plan length



Evaluation 2: vs # of unrelated resources



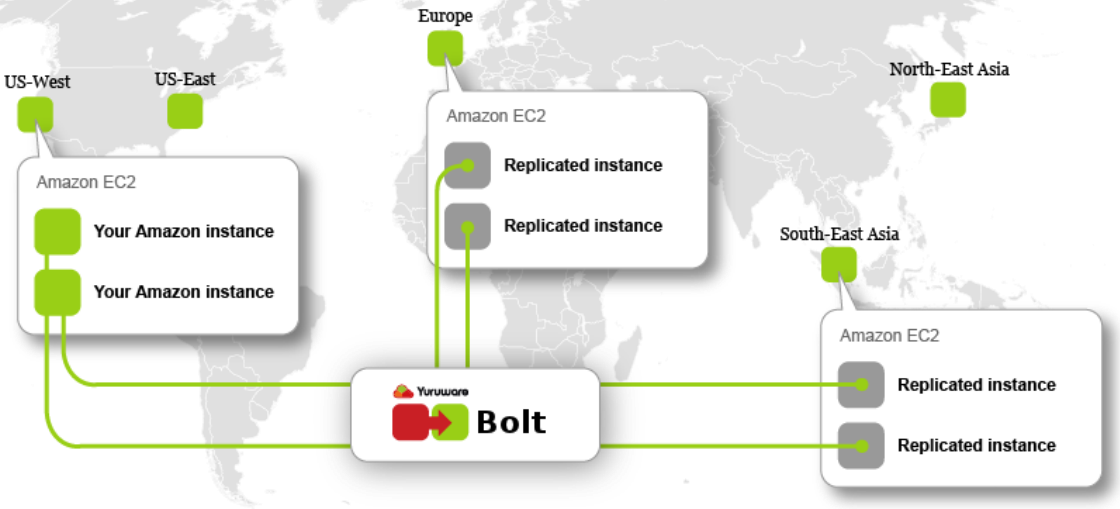
Conclusion & future work

- Rollback in cloud using AI planner
  - Formalized part of AWS APIs in a planning domain
  - Planning execution time is marginal for practical system sizes
- Future work
  - Extending checkpoints to capture internal resource state
  - Parallelizing plans
  - Finding forward plans with constraints

Questions?

NICTA

Yuruware



The paper is available at www.nicta.com.au/pub?doc=5994