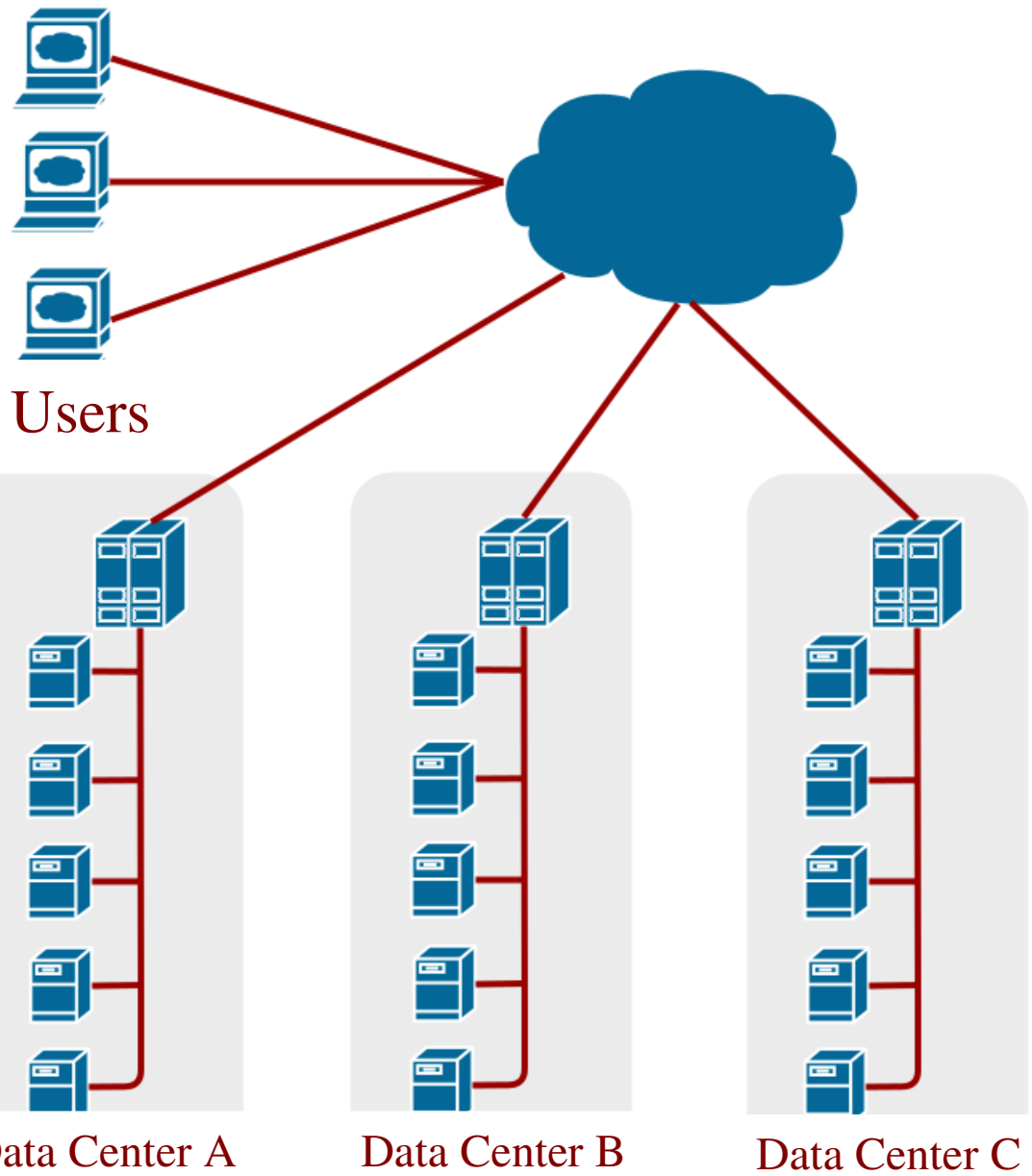
Carnegie Mellon

Managing Uncertainty in Self-* Systems with Plan Reuse and Stochastic Search

Cody Kinneer, Zack Coker, Joanna Wang,
David Garlan, Claire Le Goues

Cloud Web Server



Cloud Web Server

Data Centers

Data Center Properties

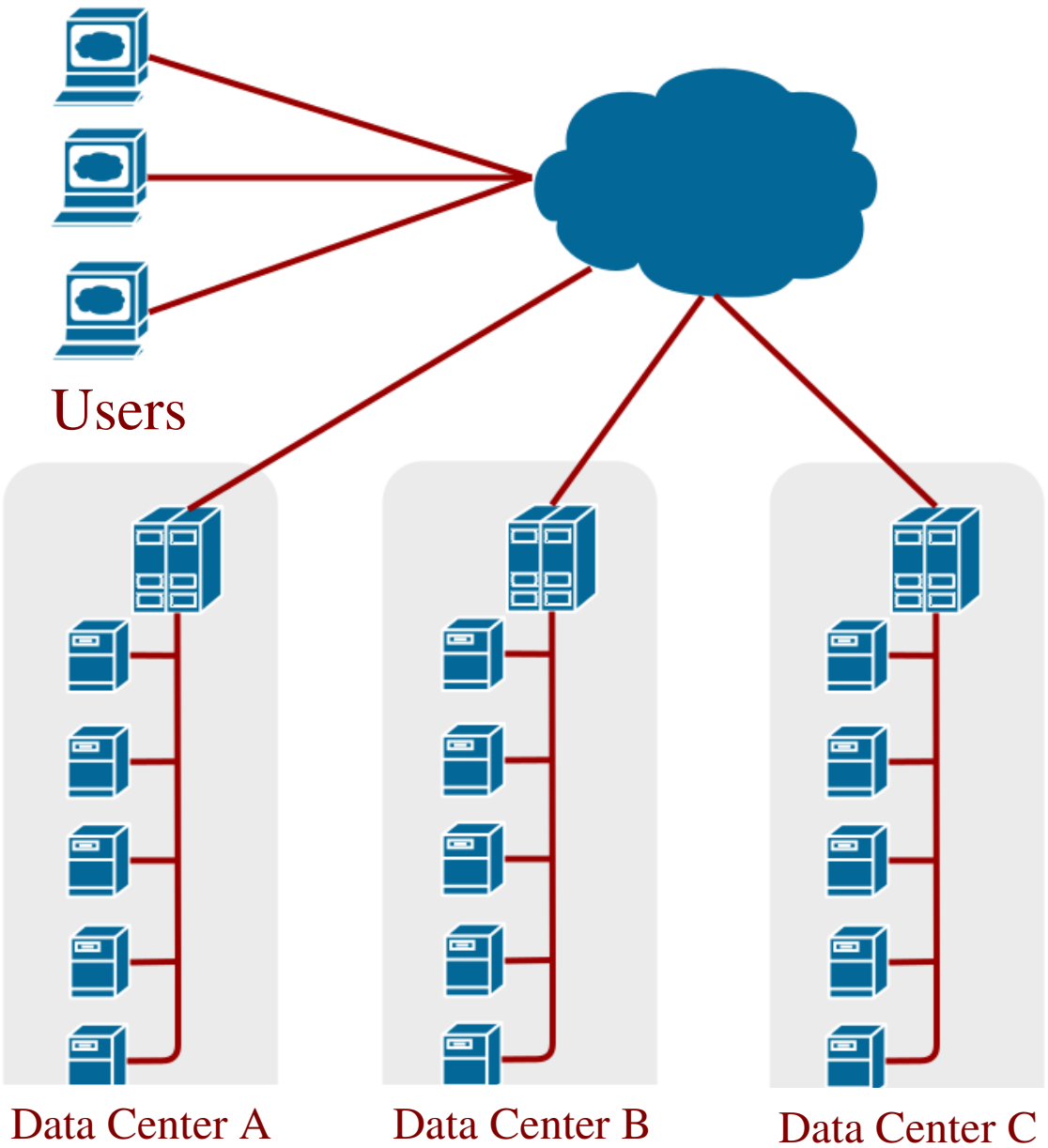
- Server type
- Traffic level
- Dimmer
- Number of servers

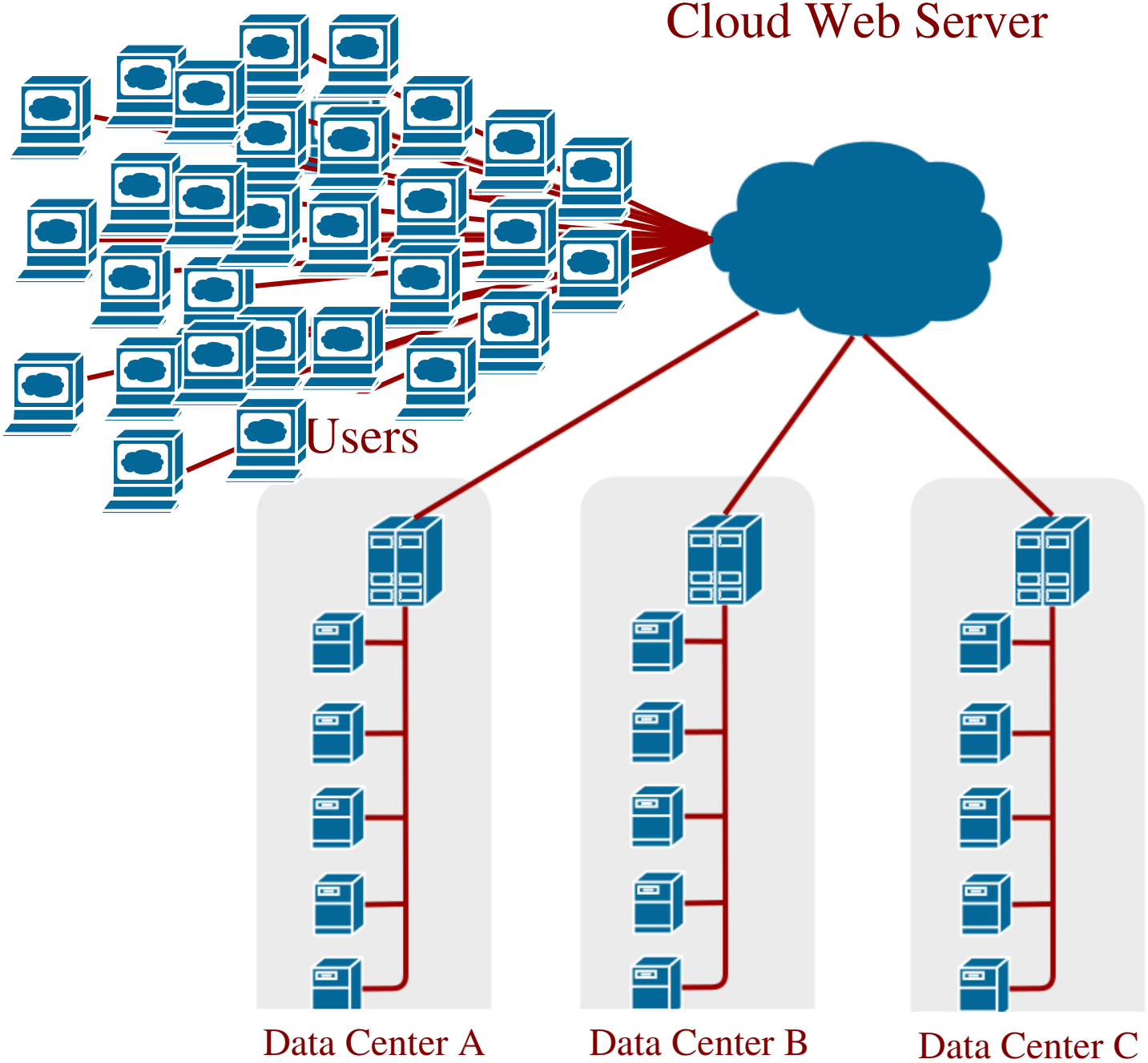
Servers

Server Properties

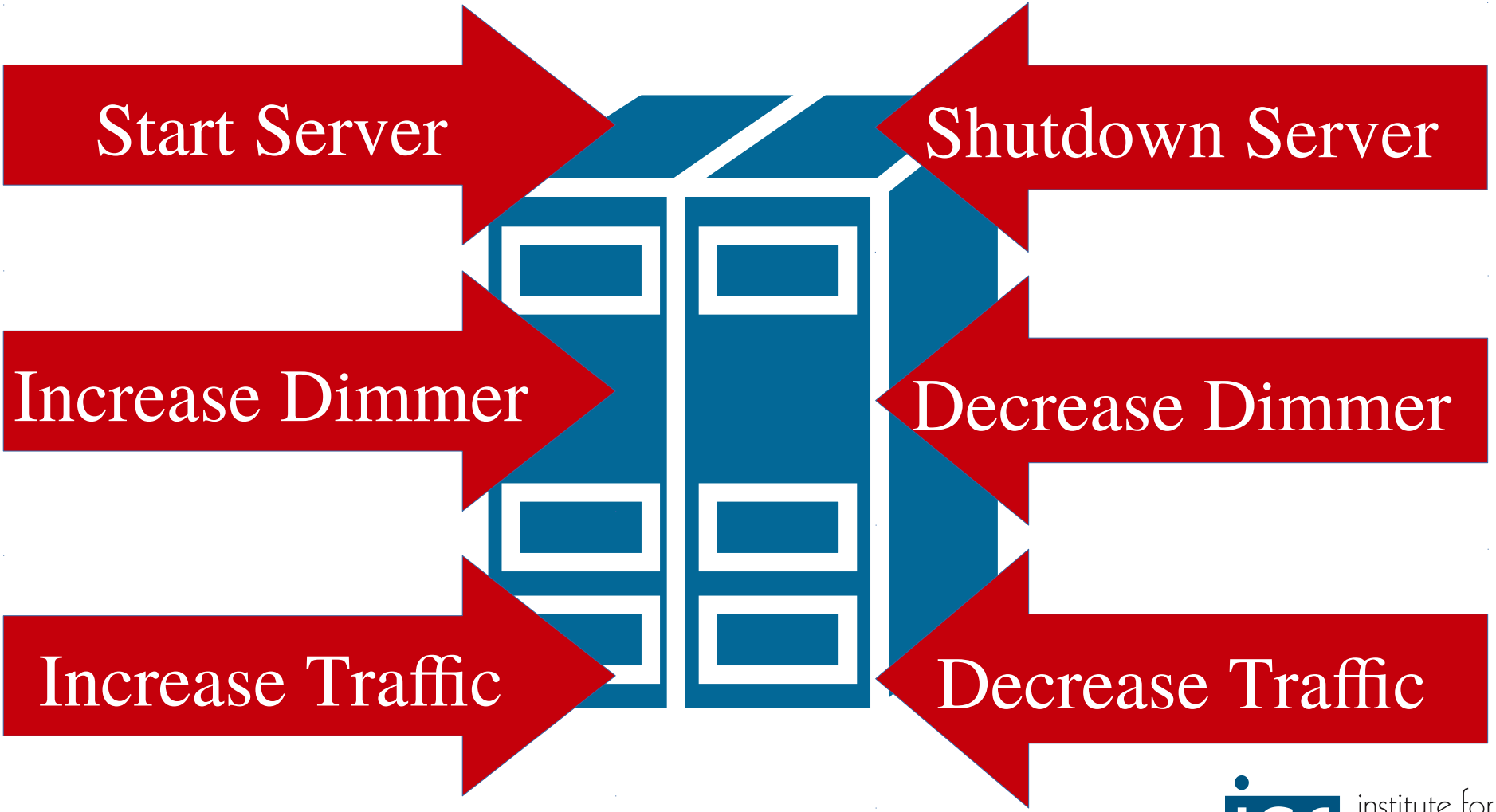
- Cost
- Max full requests
- Max dimmed requests

Cloud Web Server

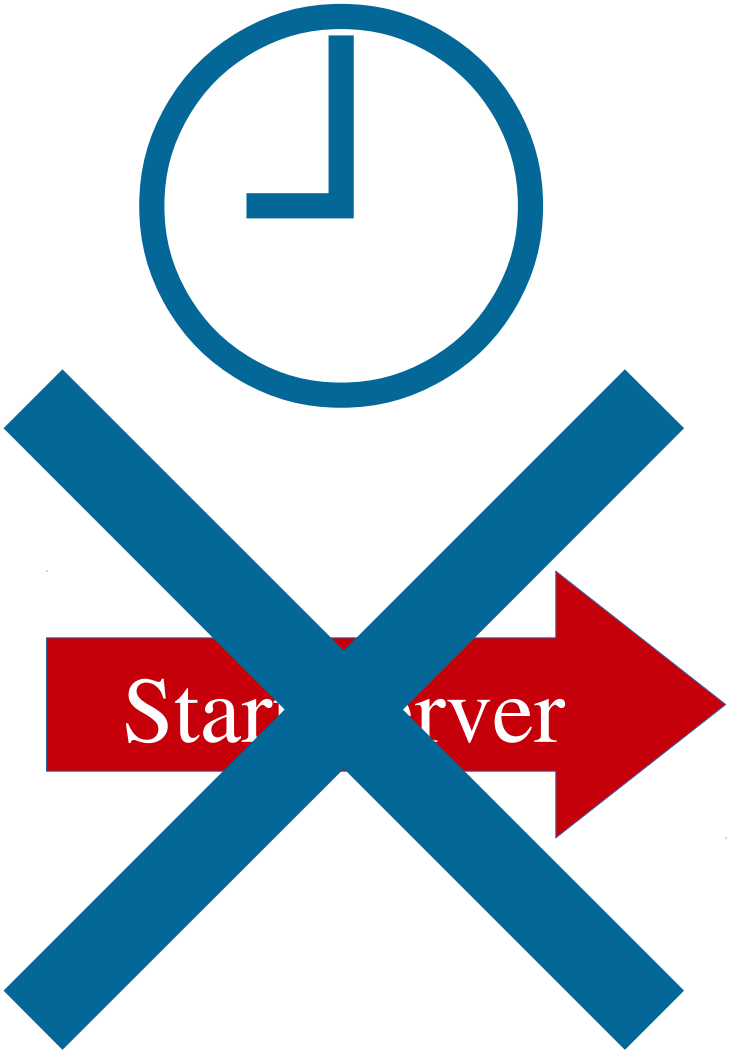


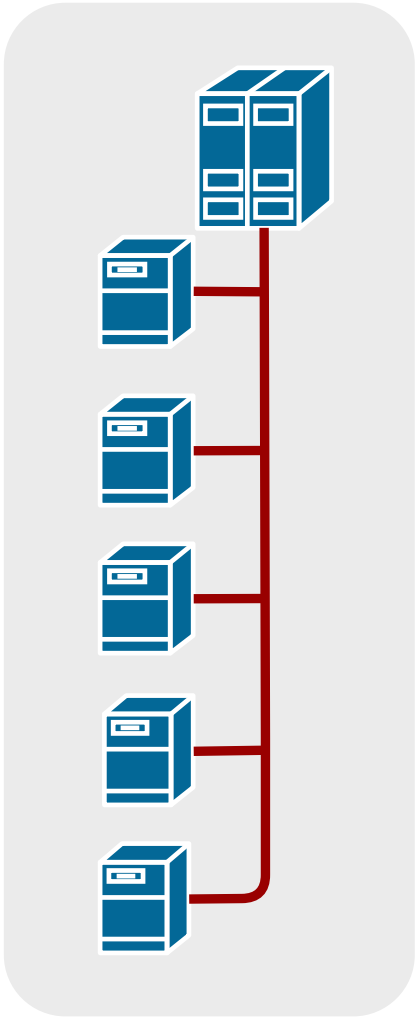

Cloud Web Server
Users
Data Center A
Data Center B
Data Center C

Tactics


Start Server
Shutdown Server
Increase Dimmer
Decrease Dimmer
Increase Traffic
Decrease Traffic

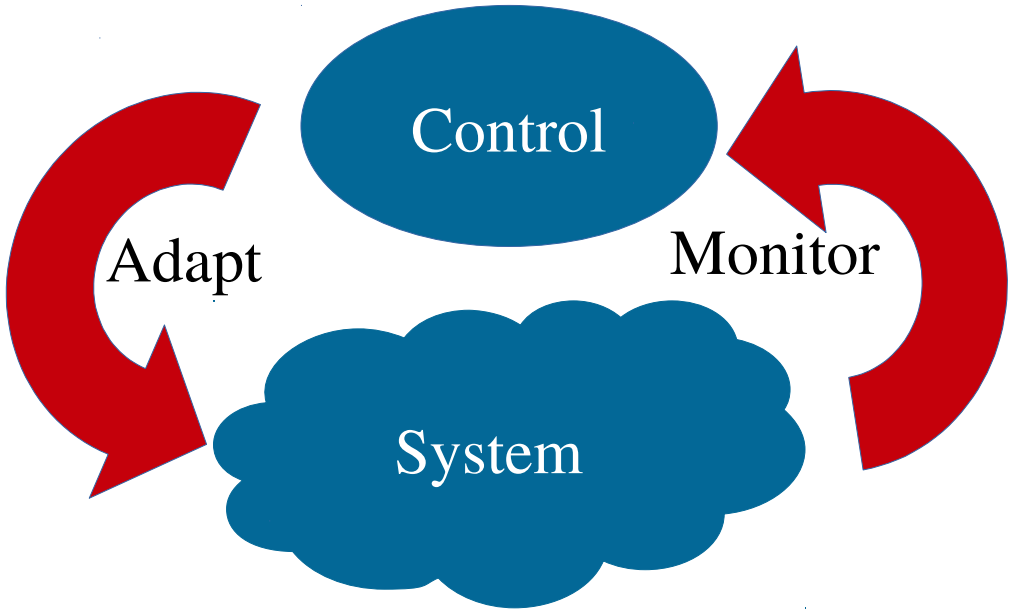
Time and Failure





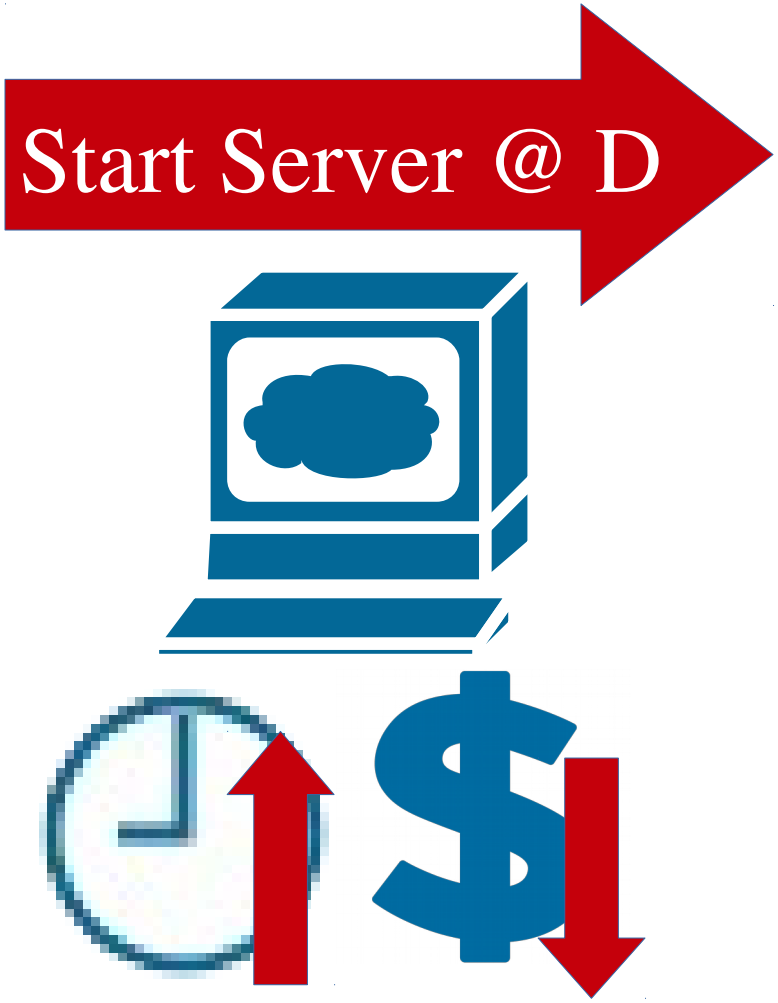
Existing Approaches

- Rainbow / Stitch
- Planning
  - Manual
  - Automated
    - PRISM



Handle Evolution?

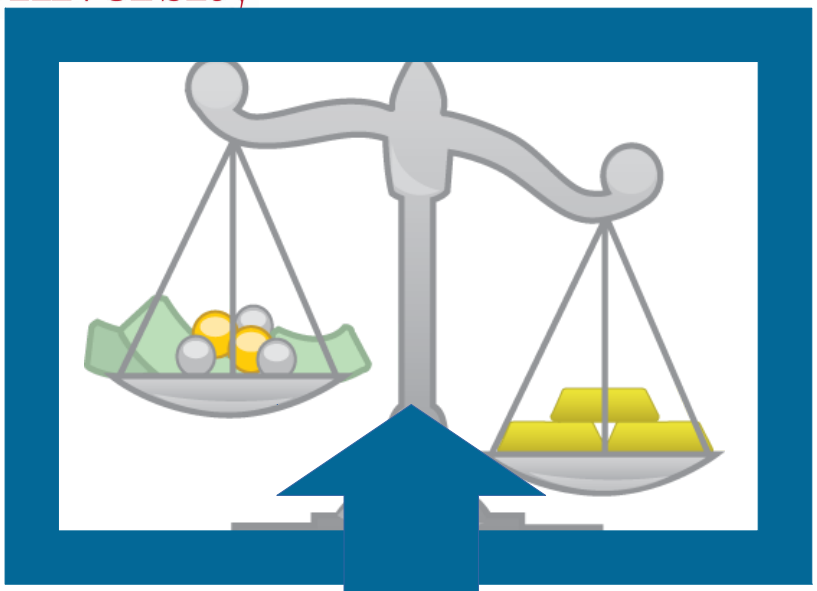
Evolution



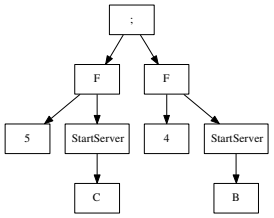
- Evolving Tactics
- Evolving Environment
- Evolving Quality Priorities

Our Approach

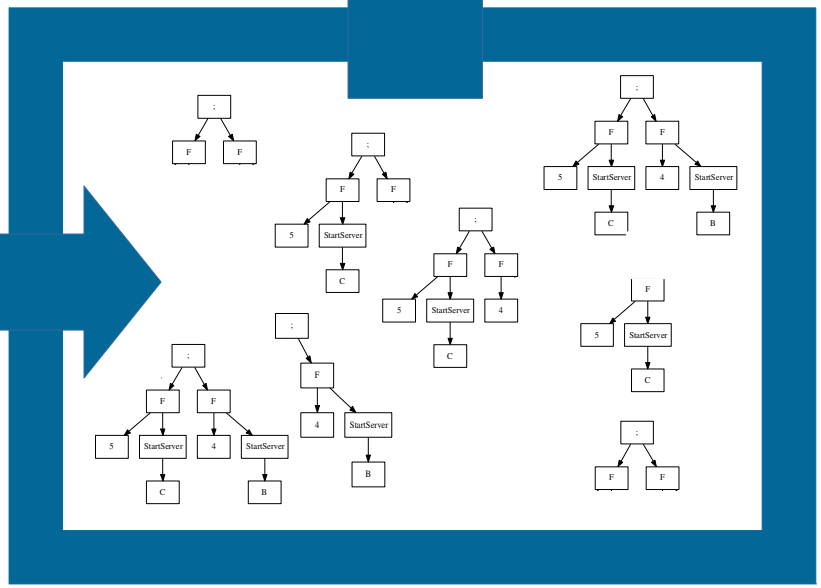
- Genetic Programming
  - Inspired by automated repair
- Simple planning language
- Reuse enabling techniques
- Implemented in Java using ECJ library from George Mason University



Evaluation



Initial Plan



Mutation
and
Crossover

$$\langle plan \rangle ::= \text{`(`}\ \langle operator \rangle\ \text{`)`}$$
$$\quad | \quad \text{`(`}\ \langle tactic \rangle\ \text{`)`}$$

$$\langle operator \rangle ::= \langle for\text{-}loop \rangle$$
$$\quad | \quad \langle try\text{-}catch \rangle$$
$$\quad | \quad \langle sequence \rangle$$

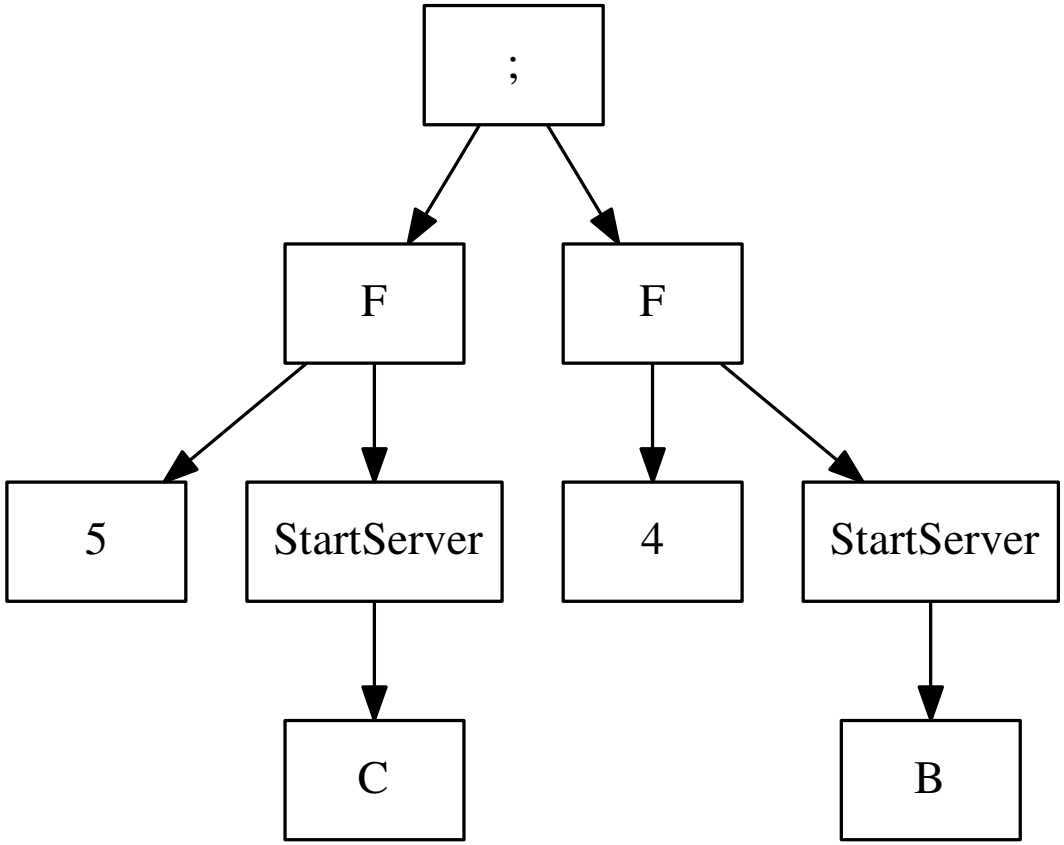
$$\langle for\text{-}loop \rangle ::= \text{`F`}\ \langle int \rangle\ \langle plan \rangle$$

$$\langle int \rangle ::= [2\text{-}10]$$

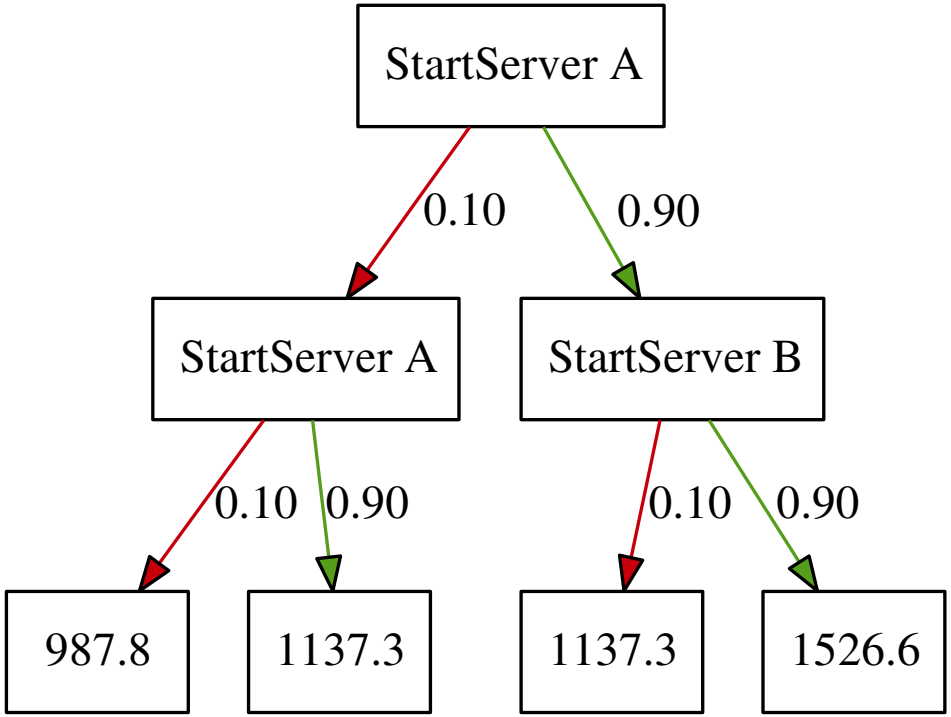
$$\langle sequence \rangle ::= \text{`;`}\ \langle plan \rangle\ \langle plan \rangle$$

$$\langle try\text{-}catch \rangle ::= \text{`T`}\ \langle plan \rangle\ \langle plan \rangle\ \langle plan \rangle$$

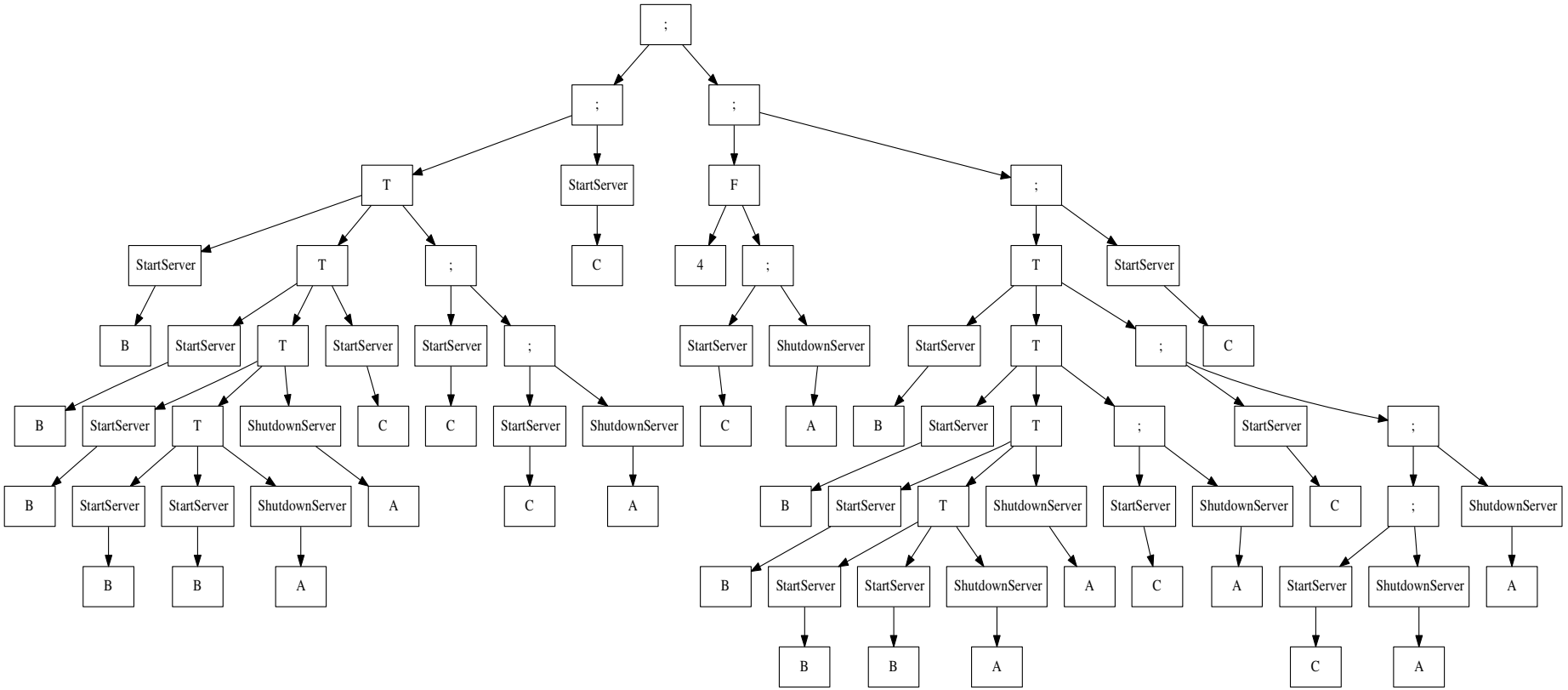
Plans



Plan Evaluation



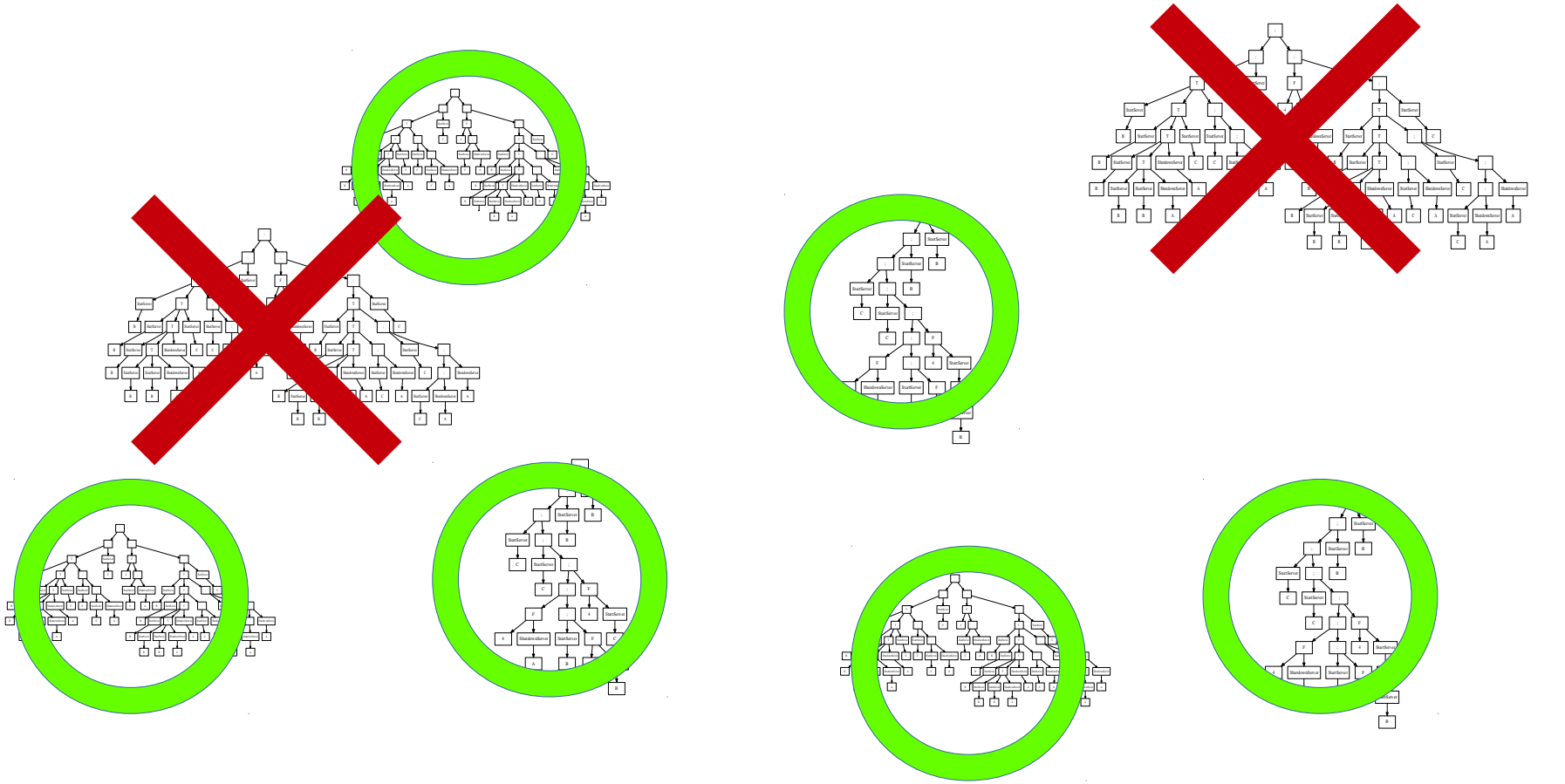
Good Plans are Big



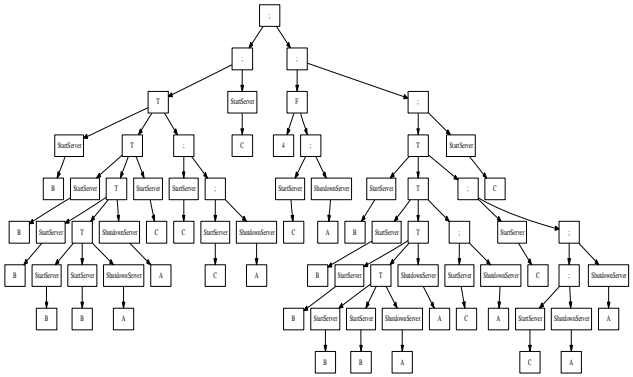
Reuse Enabling

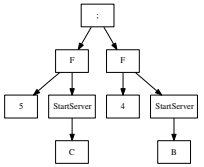
- Kill ratio
- Scratch ratio
- Plan Trimmer

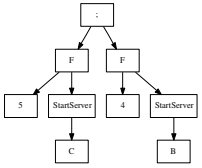
Kill Ratio

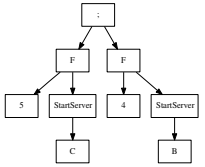


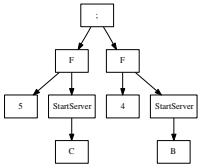
Scratch Ratio



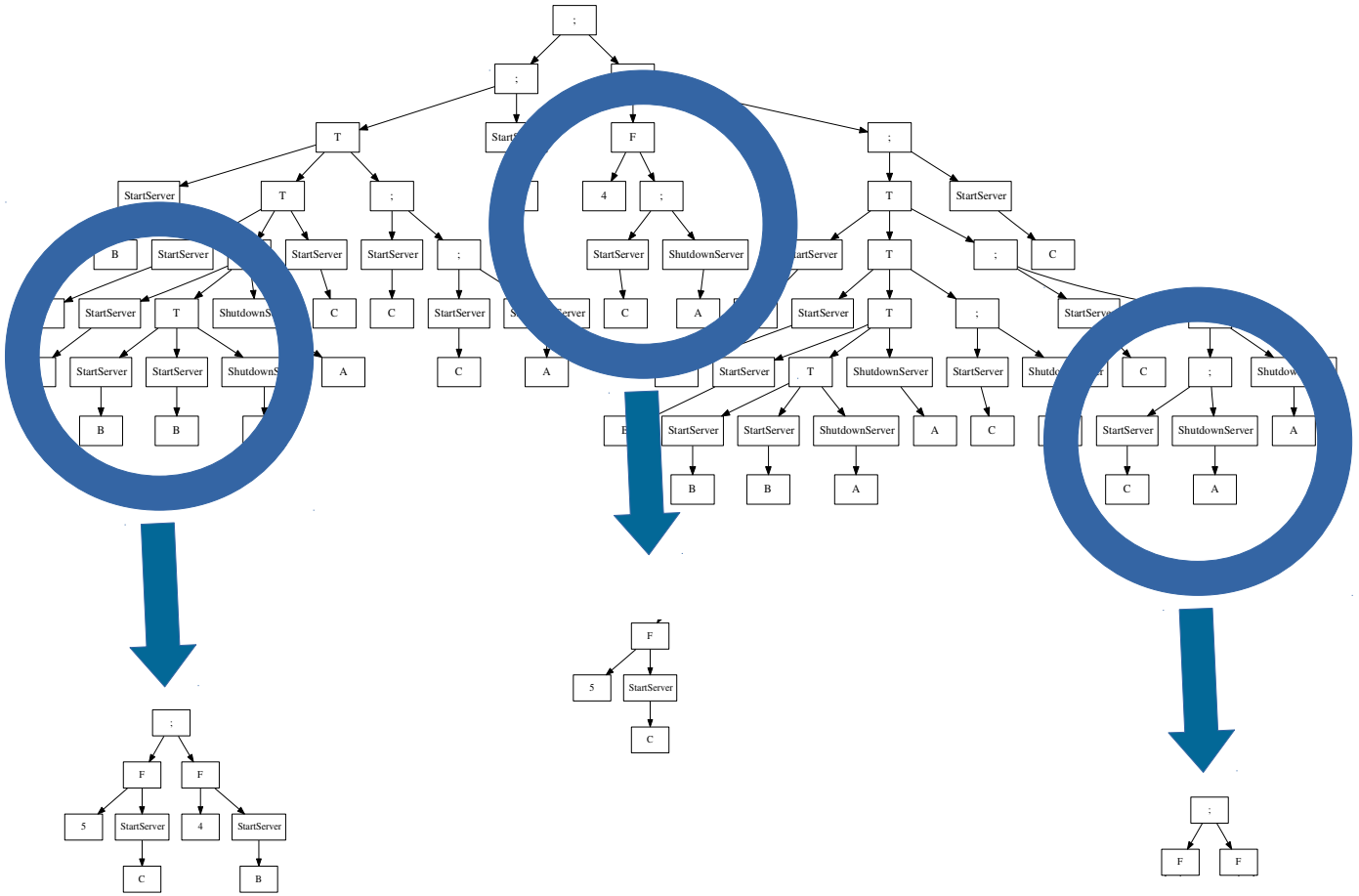








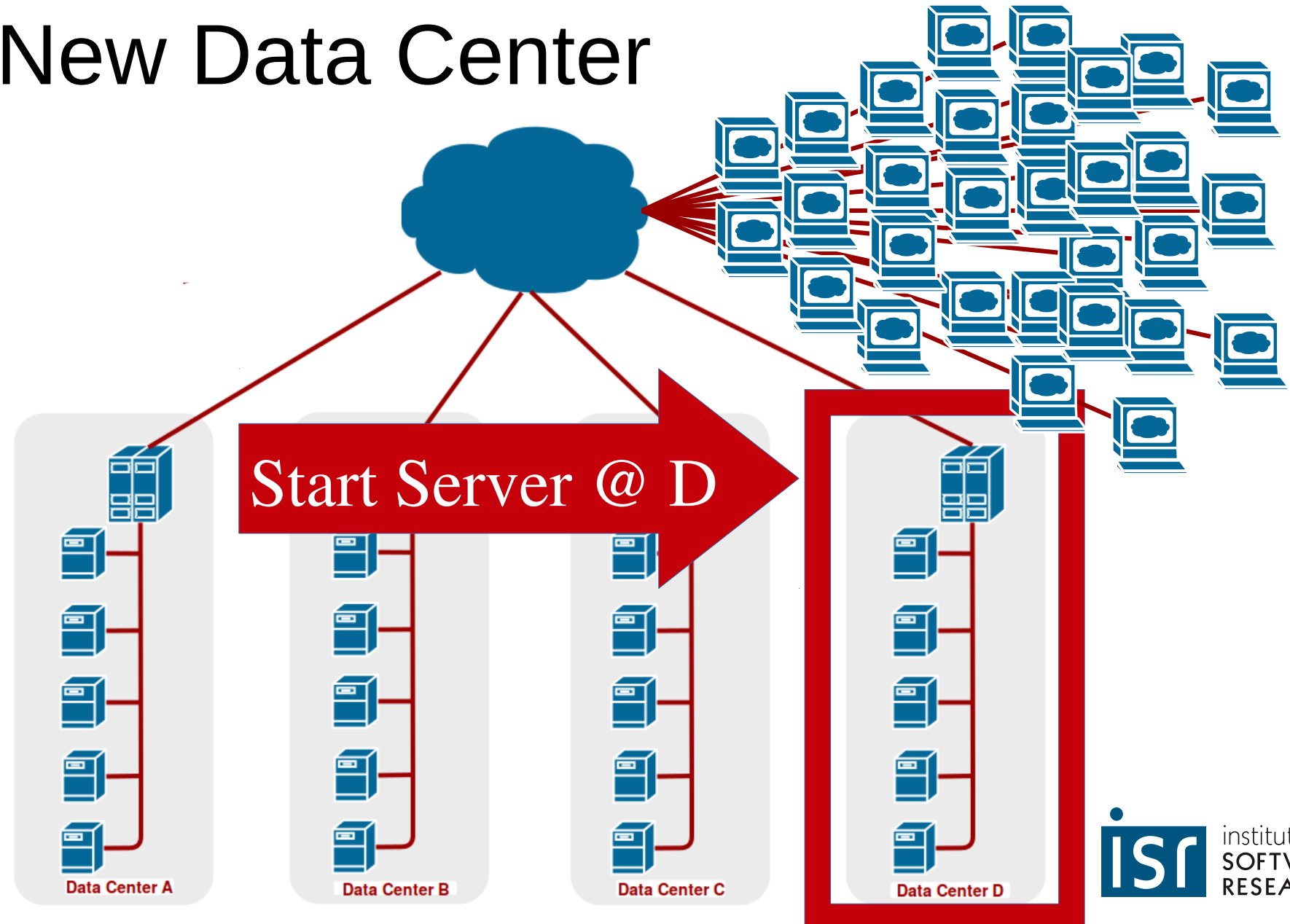
Trimmer



Empirical Evaluation

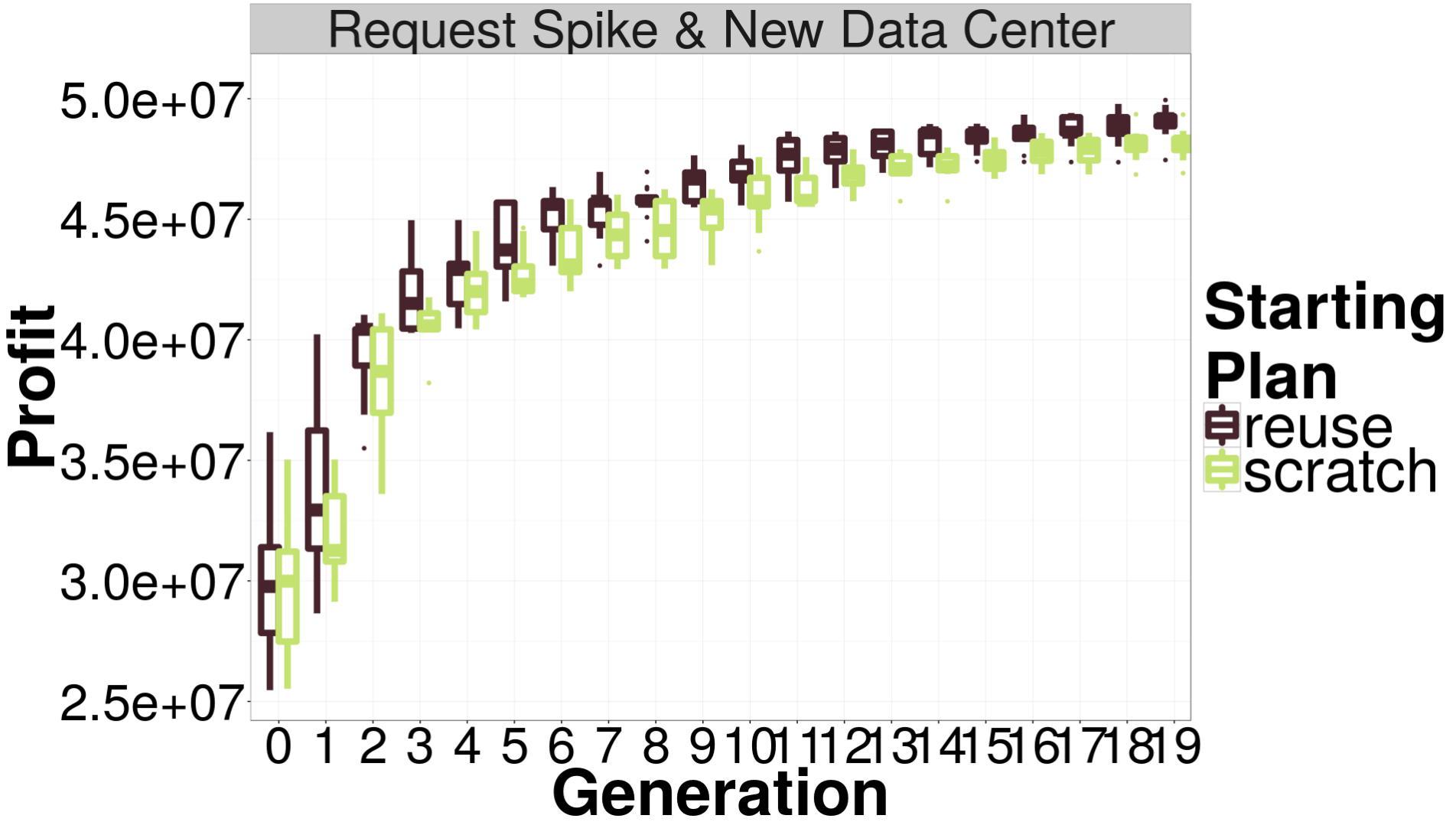
- Does plan reuse result in improved fitness?
- Change model and plan for the new situation
- All results involving randomness are the median of 10 trials
- P values obtained by Wilcoxon rank sum test

Request Spike & New Data Center



Reuse Enabling Techniques

Planning Technique	Utility	P Value
Scratch	1.000	
Scratch & Kill Ratio	1.044	< 0.01
Naive Reuse	0.962	0.06
Reuse & Kill Ratio	1.072	< 0.01
Reuse & Kill Ratio & Scratch Ratio	1.077	0.63
Reuse & Kill Ratio & Scratch Ratio & Trimmer	1.112	< 0.01


Request Spike & New Data Center
Profit
5.0e+07
4.5e+07
4.0e+07
3.5e+07
3.0e+07
2.5e+07
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
Generation
Starting Plan
reuse
scratch

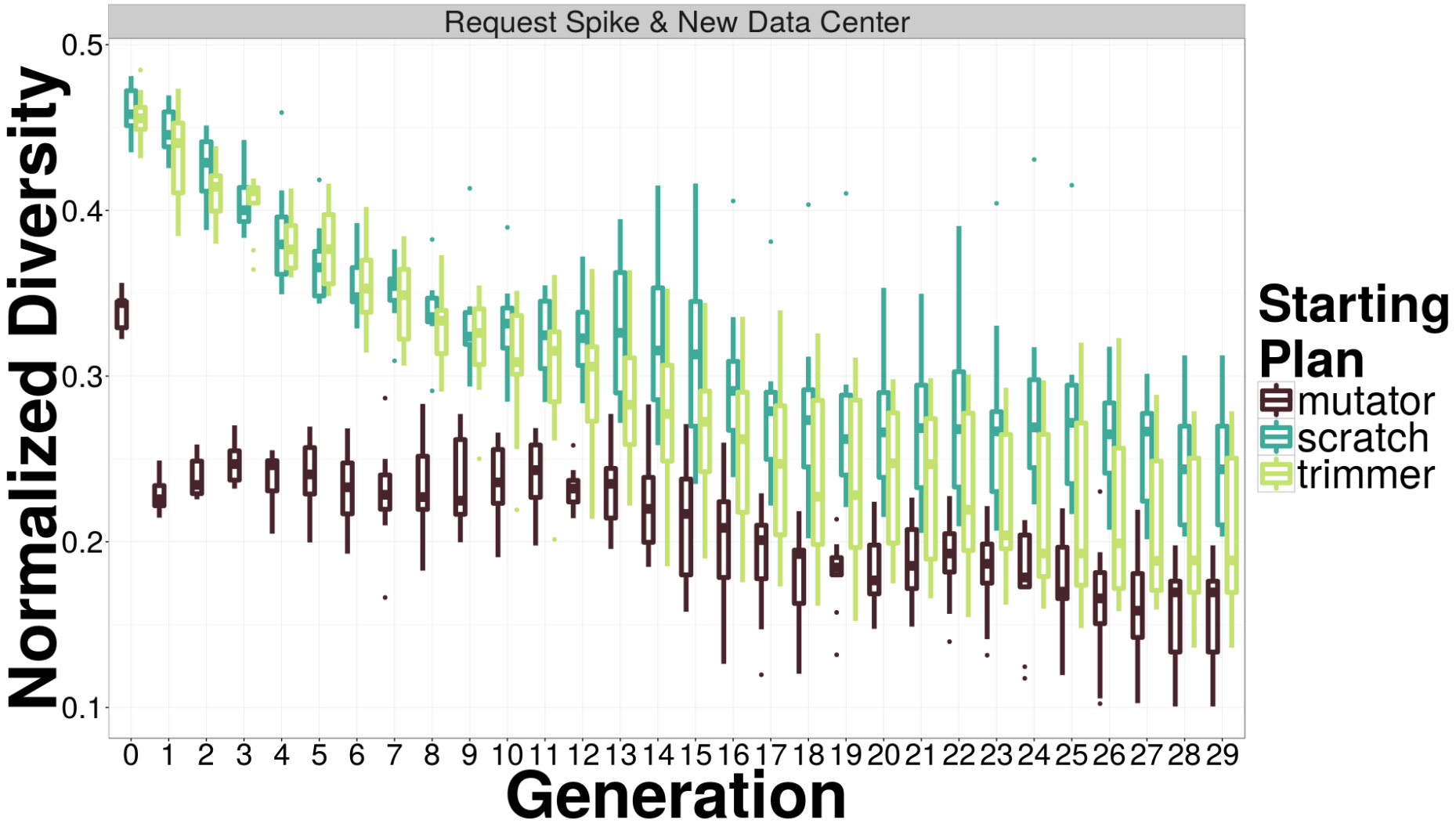
Reuse Improvement

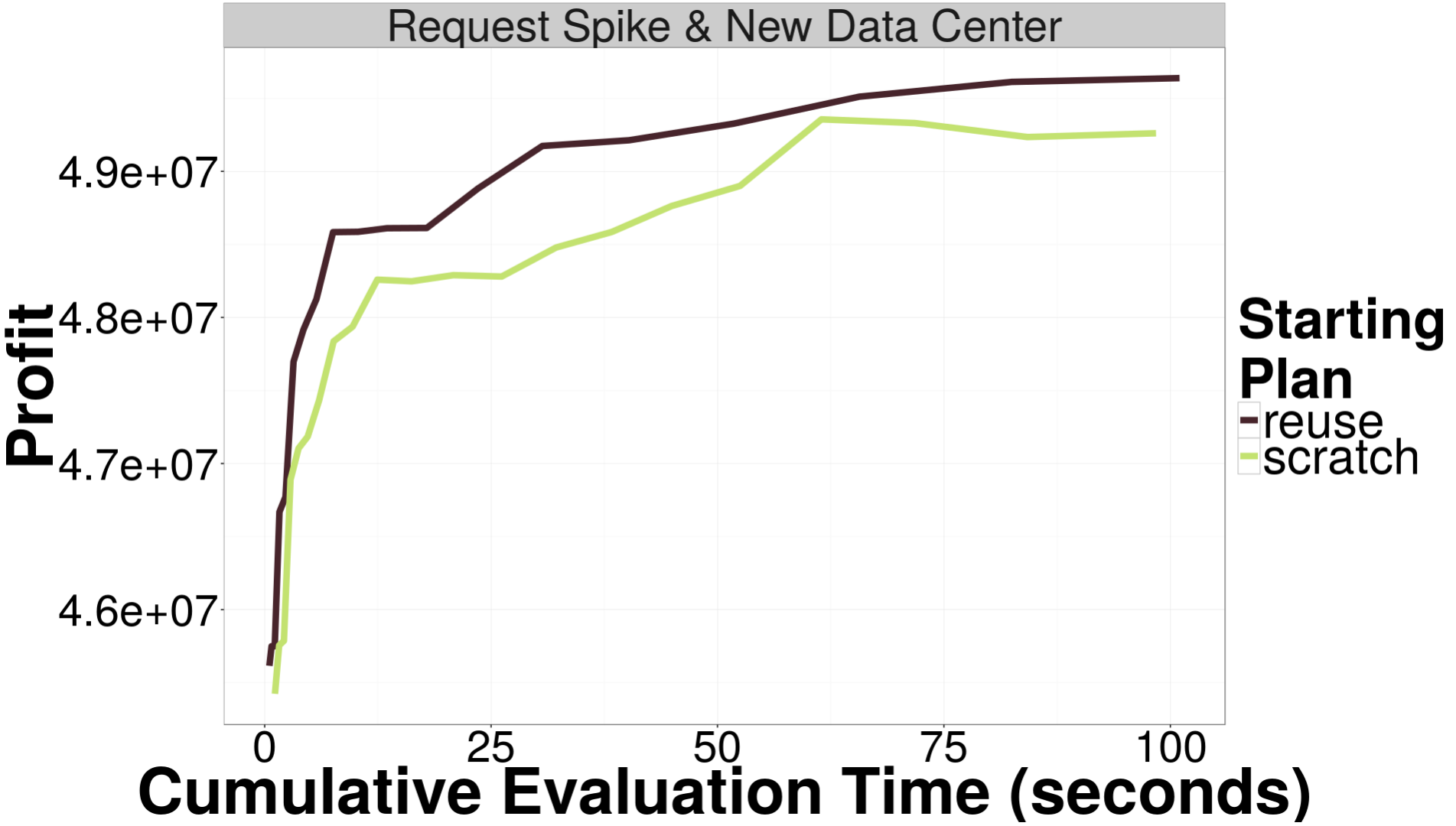
Scenario	1k	10k
Increased Costs	0.02	0.81
Network Unreliability	0.01	**0.10**
Failing Data Center	-0.02	**0.14**
Request Spike	-0.14	-0.01
New Data Center	-0.63	0.28
Request Spike & New Data Center	-0.47	**1.54**

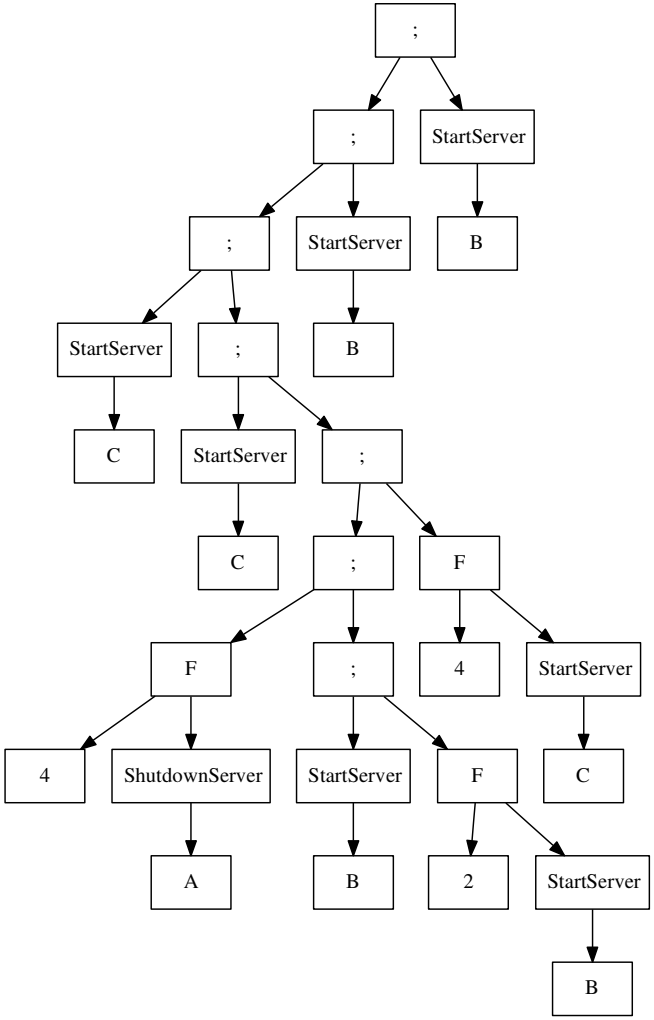
Key Contributions

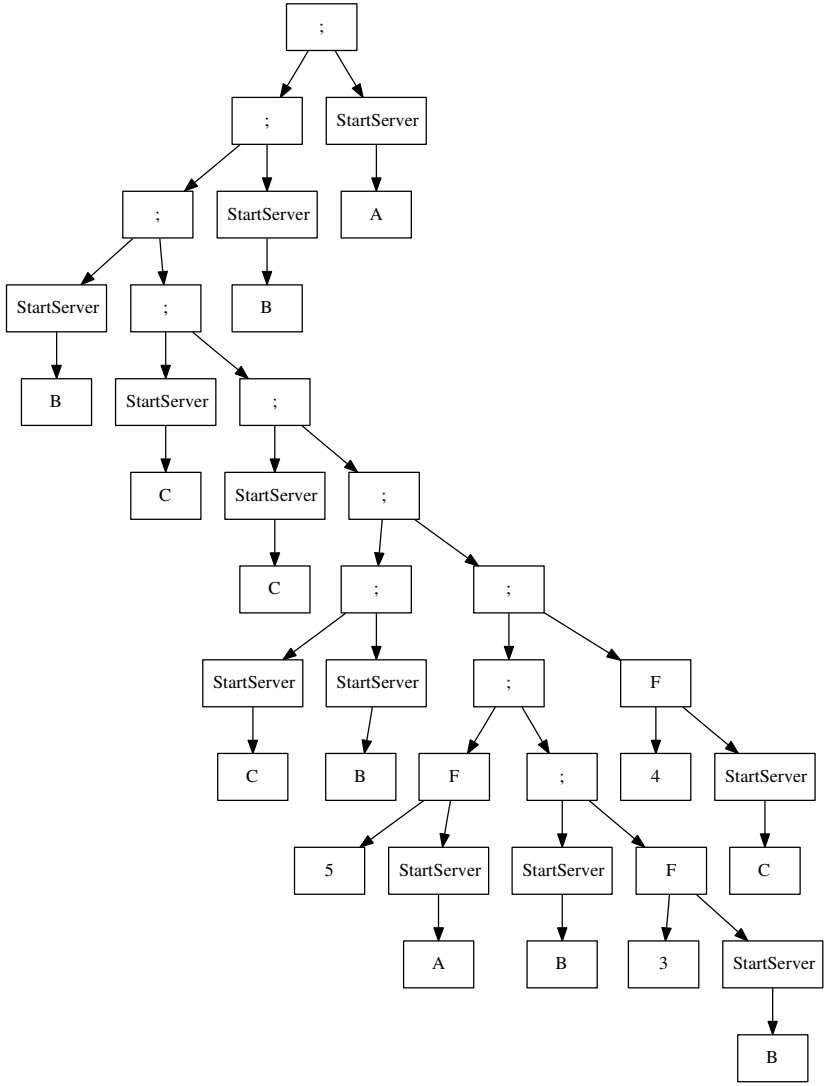
- A planner based on genetic programming for self-adaptive systems
- A set of techniques for enabling more efficient plan reuse in GP
- Enhancing existing plans can result in improved fitness compared to planning from scratch

Cody Kinneer ckinneer@andrew.cmu.edu

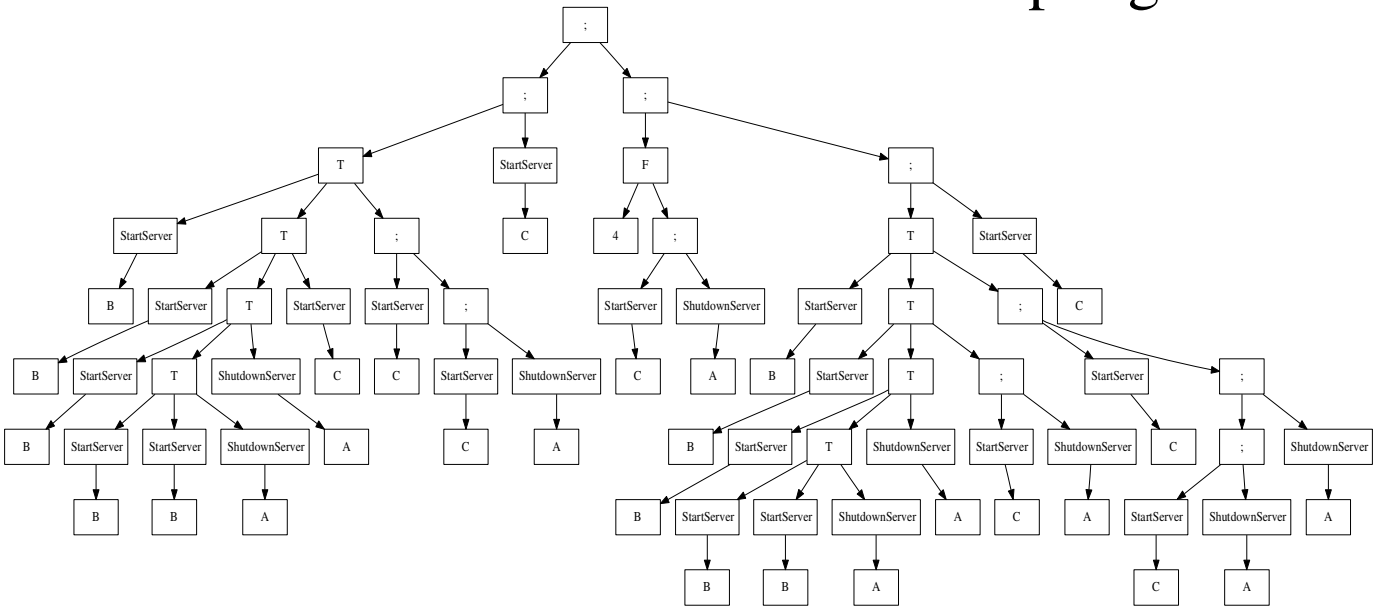

Request Spike & New Data Center
Normalized Diversity
0.5
0.4
0.3
0.2
0.1
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29
Generation
Starting Plan
mutator
scratch
trimmer

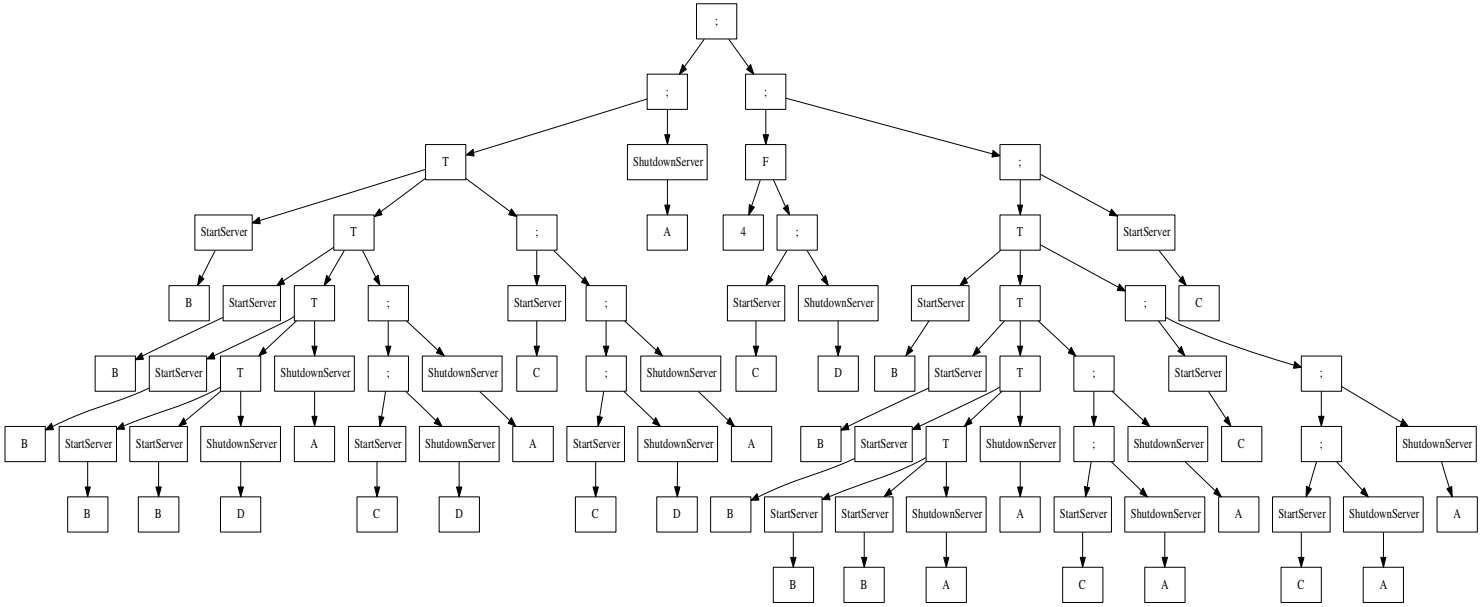

Request Spike & New Data Center
Profit
4.9e+07
4.8e+07
4.7e+07
4.6e+07
0
25
50
75
100
Cumulative Evaluation Time (seconds)
Starting Plan
reuse
scratch


;
;
StartServer
;
StartServer
B
StartServer
;
B
C
StartServer
;
C
;
F
F
;
4
StartServer
4
ShutdownServer
StartServer
F
C
A
B
2
StartServer
B


;
;
StartServer
;
StartServer
A
StartServer
;
B
B
StartServer
;
C
StartServer
;
C
;
;
StartServer
StartServer
;
F
C
B
F
;
4
StartServer
5
StartServer
StartServer
F
C
A
B
3
StartServer
B

Adapting Plan to New Tactic

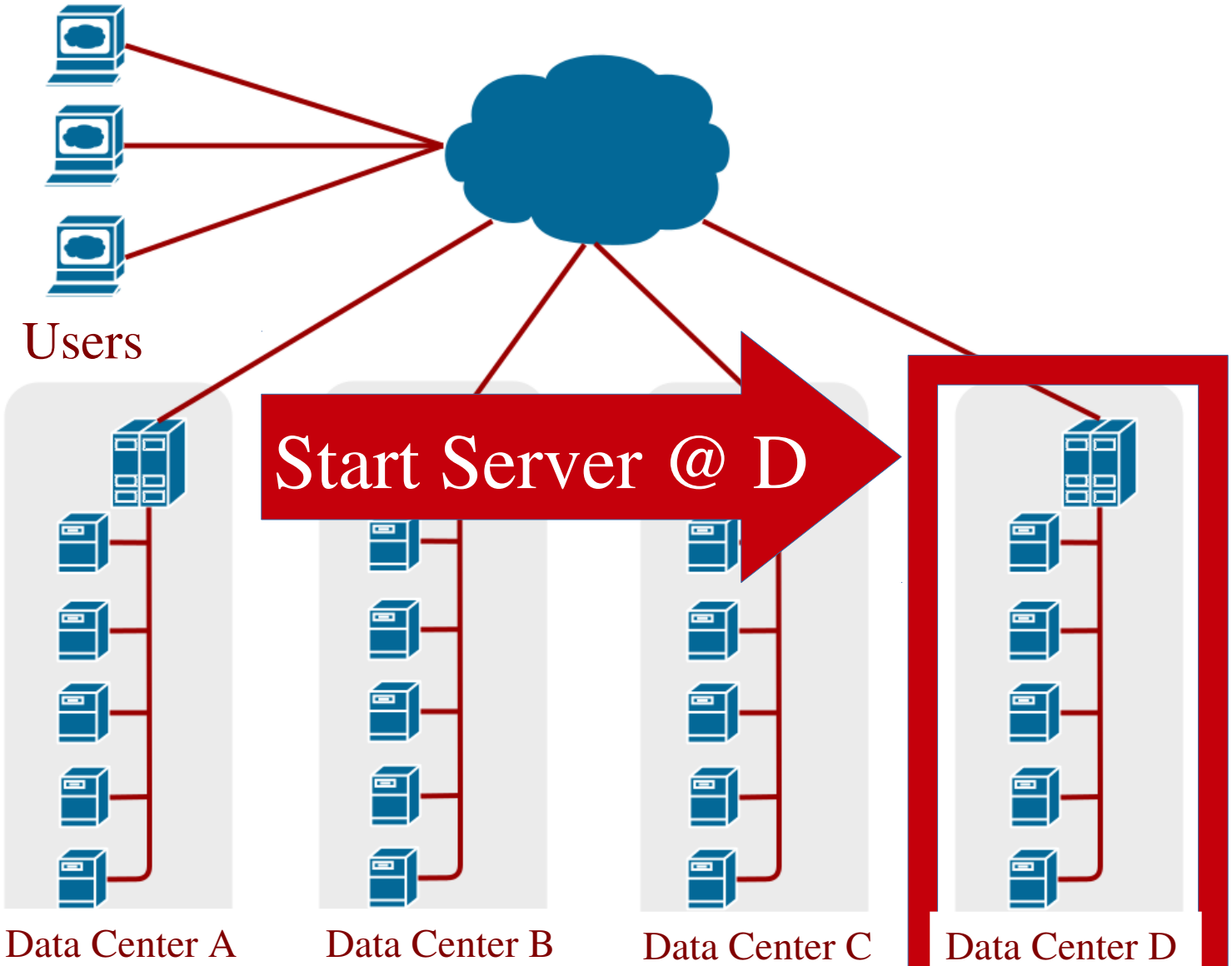


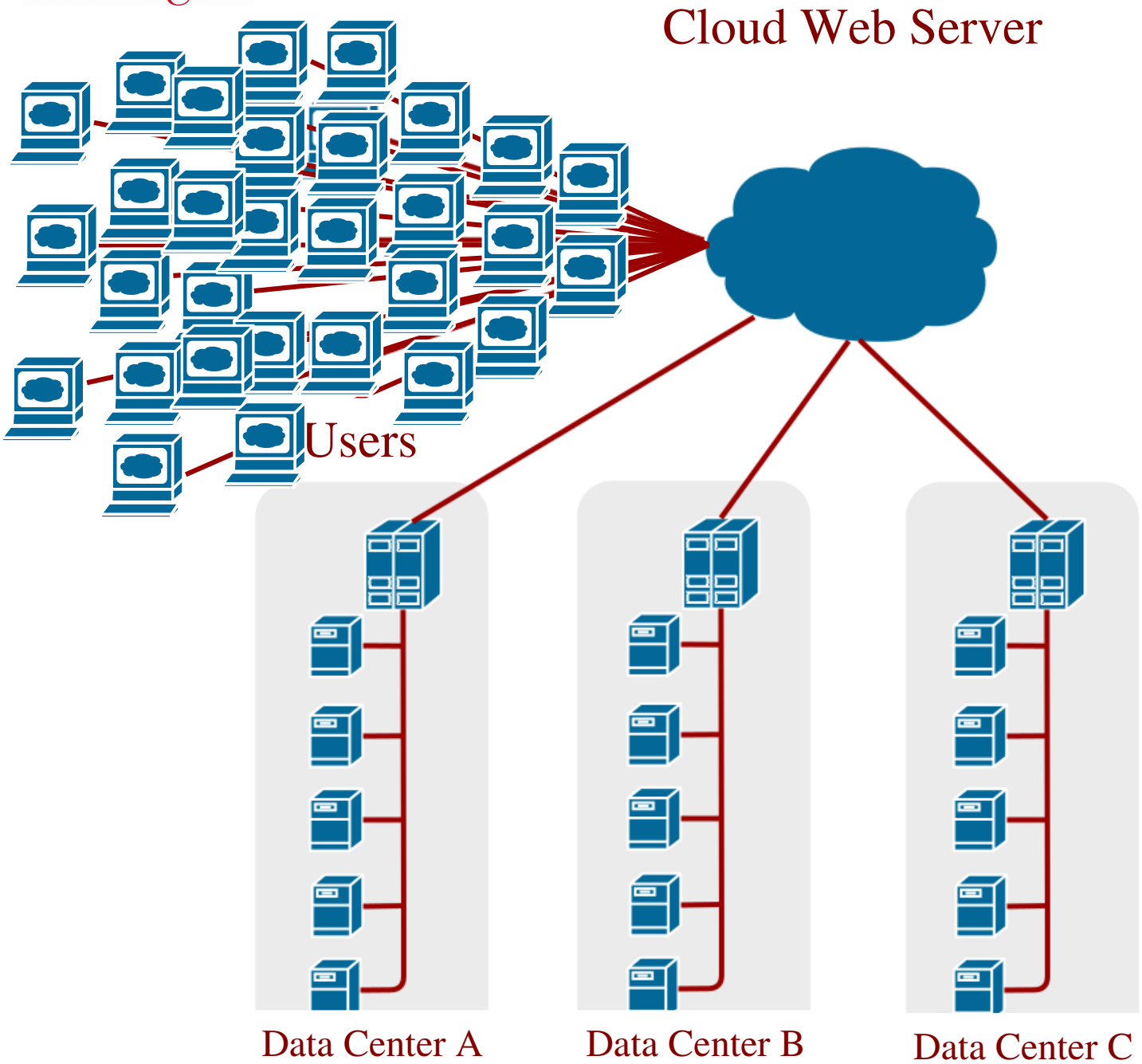


Empirical Results

Planning Technique	Utility	P Value
Scratch	1.000	
Reuse	0.962	0.06

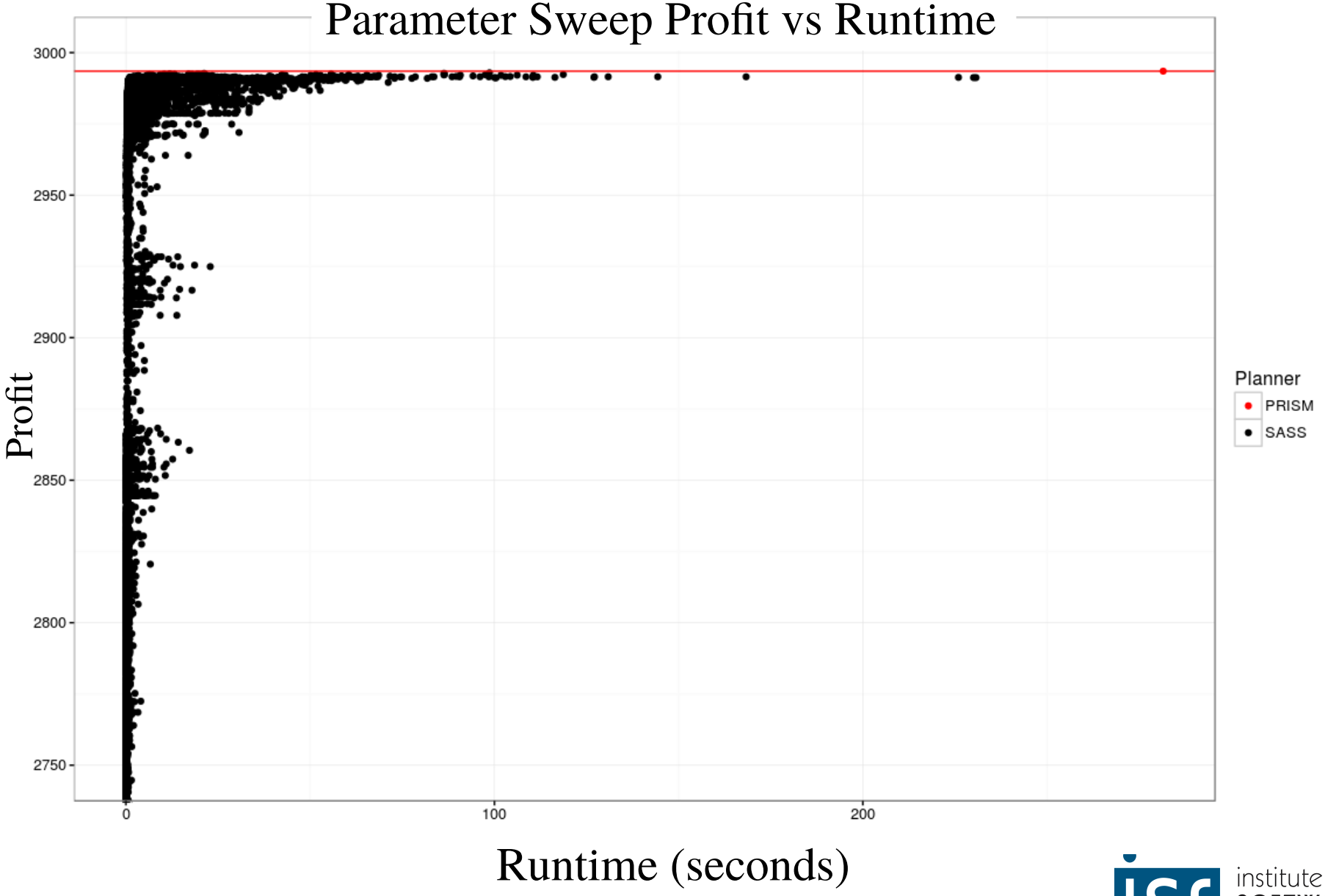
Evolving Tactics




Cloud Web Server
Users
Data Center A
Data Center B
Data Center C

Empirical Evaluation

- Sanity check: compare planner to PRISM probabilistic model checker
- Is the planner close to optimal?
- How much faster?


Parameter Sweep Profit vs Runtime
Profit
3000
2950
2900
2850
2800
2750
0
100
200
Runtime (seconds)
Planner
PRISM
SASS