A Theory of Auto-Scaling for Resource Reservation in Cloud Services

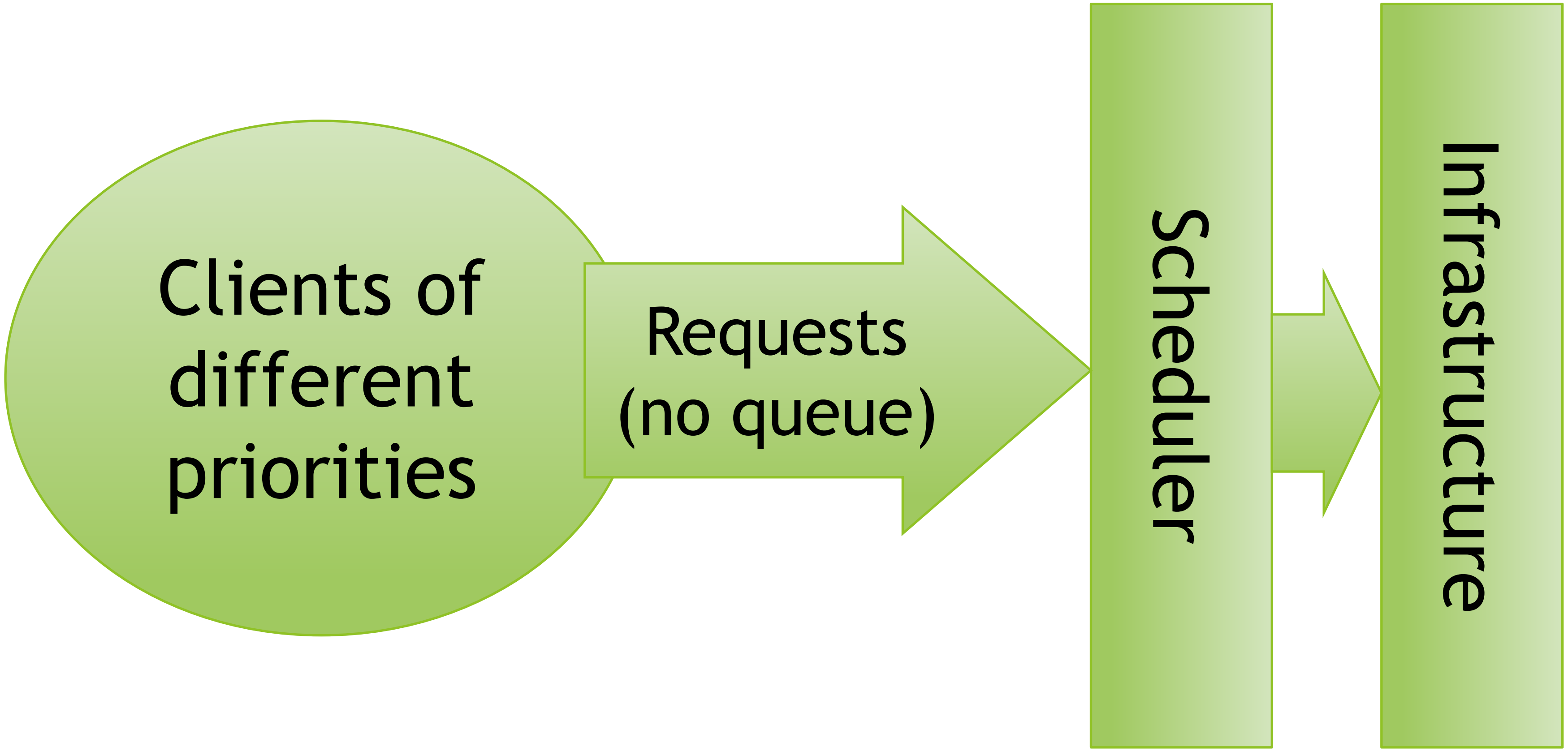
Konstantinos Psychas, Javad Ghaderi

Columbia University

Performance 2020: 38th International Symposium on Computer Performance, Modeling, Measurements and Evaluation

November 2-6 2020, Milan(Online)

Problem Setting



- We will use terms **job** and **server**
- Both treated as abstract **multidimensional** resource vectors

Model

If and *where* to schedule

A set of J
job types $\mathcal{J}$

Type $j \in \mathcal{J}$ gives
reward u_j
per time unit

A set of L
homogeneous
servers

Normalized Workload
(jobs per server)
$\boldsymbol{\rho} := (\rho_1, \rho_2, \cdots, \rho_J)$

#jobs from each type that can
simultaneously fit in the server
$\boldsymbol{k} = (k_1, k_2, \cdots, k_J) \in \mathcal{K} \subset \mathbb{R}^J$

How to schedule when workload is known

$$\max_{\mathbf{X},\mathbf{Y}} \quad \sum_j u_j Y_j$$

$$\text{s.t.} \quad Y_j \le \hat{Y}_j^L,\ \forall j \in \mathcal{J}$$

$$\sum_{\mathbf{k}\in\mathcal{K}} X_{\mathbf{k}} k_j \ge Y_j,\ \forall j \in \mathcal{J}$$

$$\sum_{\mathbf{k}\in\mathcal{K}} X_{\mathbf{k}} = L, \quad X_{\mathbf{k}} \ge 0,\ \forall \mathbf{k} \in \mathcal{K}$$

Example $\widehat{\boldsymbol{Y}} = L\boldsymbol{\rho}$

- Even if relaxed still hard (exponential number of variables)
- Needs to be solved every time workload changes
- Unclear how to change assignment when servers are already full
- There can be consistent error in ρ causing loss

Solving Static Optimization: Ordering Configuration by total Reward

- Reward of configuration

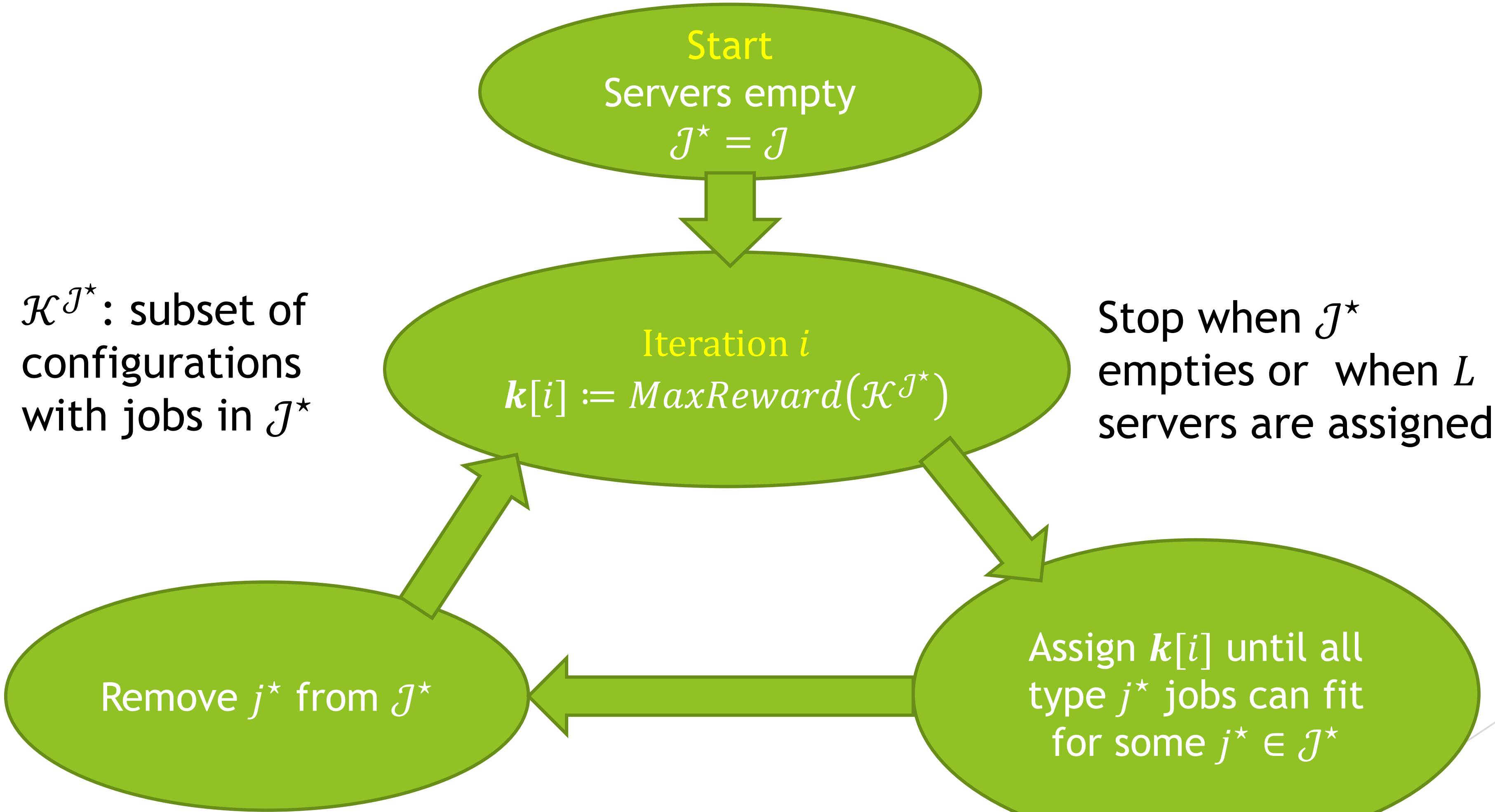
$$U(\boldsymbol{k}) = \sum_{j=1}^{J} u_j k_j$$

- $U(\boldsymbol{k})$ induces an ordering for all $\boldsymbol{k} \in \mathcal{K}$
- Define for $\mathcal{K}_s \subseteq \mathcal{K}$

$$\mathrm{MaxReward}(\mathcal{K}_s) = \arg\max_{\boldsymbol{k} \in \mathcal{K}_s} U(\boldsymbol{k})$$

Solving Static Optimization: Greedy Placement Algorithm (GPA)

$\mathcal{K}^{\mathcal{J}^\star}$: subset of configurations with jobs in $\mathcal{J}^\star$



Stop when $\mathcal{J}^\star$ empties or when L servers are assigned

Return $\widehat{\boldsymbol{X}} = \{\hat{X}_{\boldsymbol{k}[i]}, i = 1, \dots, I\}$

How good is greedy?

Consider normalized rewards ($L \to \infty$)

- Optimal Reward $U^{\star}[\boldsymbol{\rho}]$
- Greedy Reward $U^{(g)}[\boldsymbol{\rho}]$

Without extra assumptions $\Longrightarrow$ $\boxed{U^{(g)}[\boldsymbol{\rho}] \geq \frac{1}{2} U^{\star}[\boldsymbol{\rho}]}$

Monotone Greedy Property $\Longrightarrow$ $\boxed{U^{(g)}[\boldsymbol{\rho}] \geq (1 - e^{-1}) U^{\star}[\boldsymbol{\rho}]}$

- If $\boldsymbol{\rho}_1 \geq \boldsymbol{\rho}_2$ then $U^{(g)}[\boldsymbol{\rho}_1] \geq U^{(g)}[\boldsymbol{\rho}_2]$

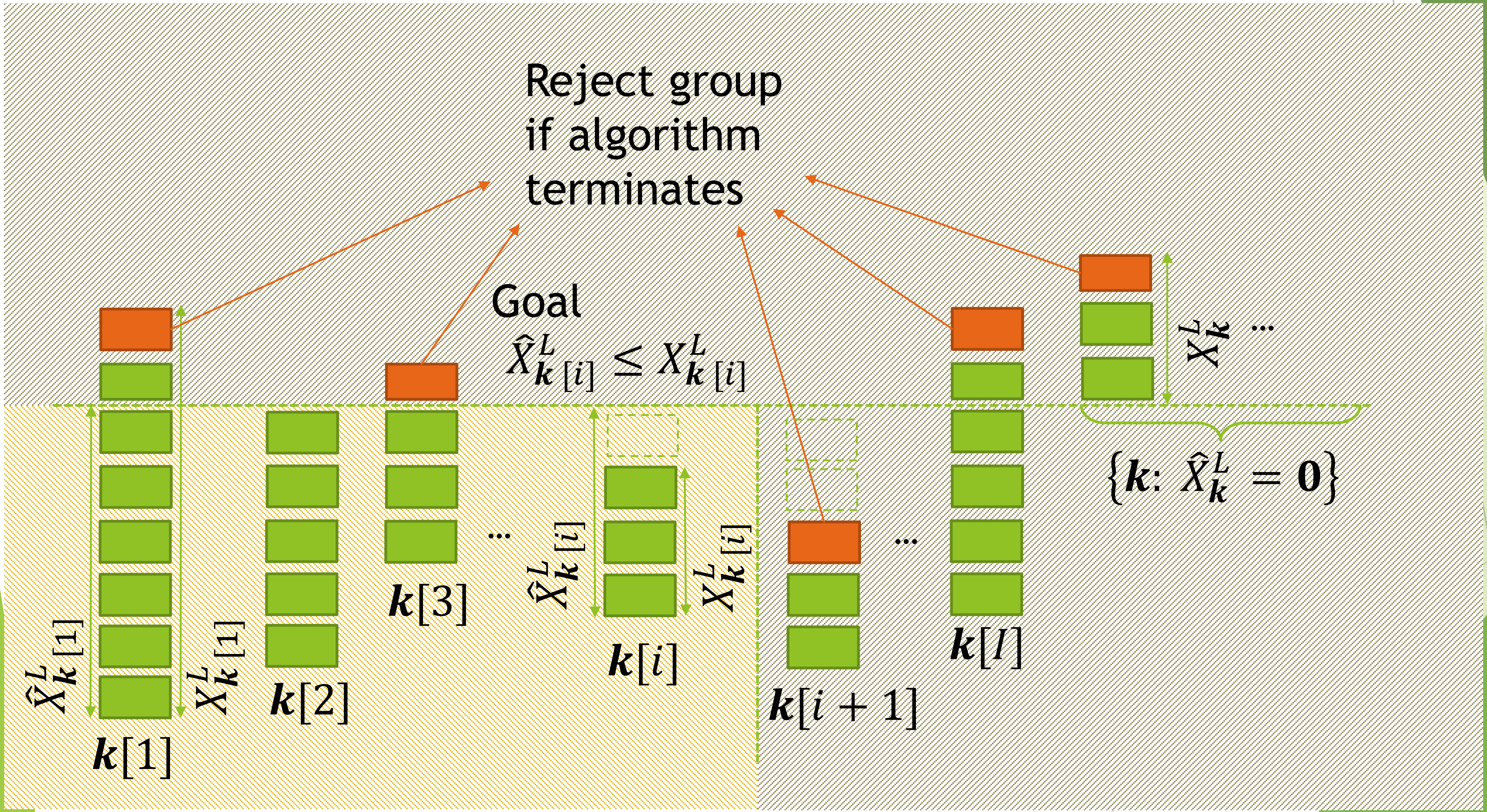
Online Algorithm: Server Groups

Server Group	Accept	Reject
Goal	Try to Fill	Try to Empty
Schedule in it	Yes	No
Migration	Migrate from a Reject Group filled slot to an Accept Group slot that empties	

Online Algorithm: (CRA) Classification and Reassignment Algorithm

- Get solution of GPA
  - Input $\widehat{\boldsymbol{Y}}^L = \boldsymbol{Y}^L$ +/- arrival/departure + $g(L)\mathbf{1}_J$
  - $g(L) = \omega(\log L)$: reservation factor
- Match server assignment to GPA solution
  - Matches configurations in decreasing reward order

CRA Example [Iteration i]



Dynamic Reservation Algorithm (DRA)

On arrival

- Run CRA
- In which slot to deploy the job arrived
- Answer: Any empty slot in Accept Group if exists

On departure

- Run CRA
- Which job to migrate in the slot that emptied
- Answer: Any job in a slot of Reject Group if exists

Informal Main Result

- Fraction of servers in each configuration of DRA $\rightarrow$ Fraction of servers in each configuration of GPA for input $\boldsymbol{\rho}L$ when $L \rightarrow \infty$
- Normalized Reward of DRA $\rightarrow$ Greedy Reward $U^{(g)}[\boldsymbol{\rho}]$

Simulations (Testing GPA approximation)

Memory: GB per CPU		
Memory-Opt	CPU-Opt	Regular
8 or 16	1 or 2	4

Generated 50 random setups

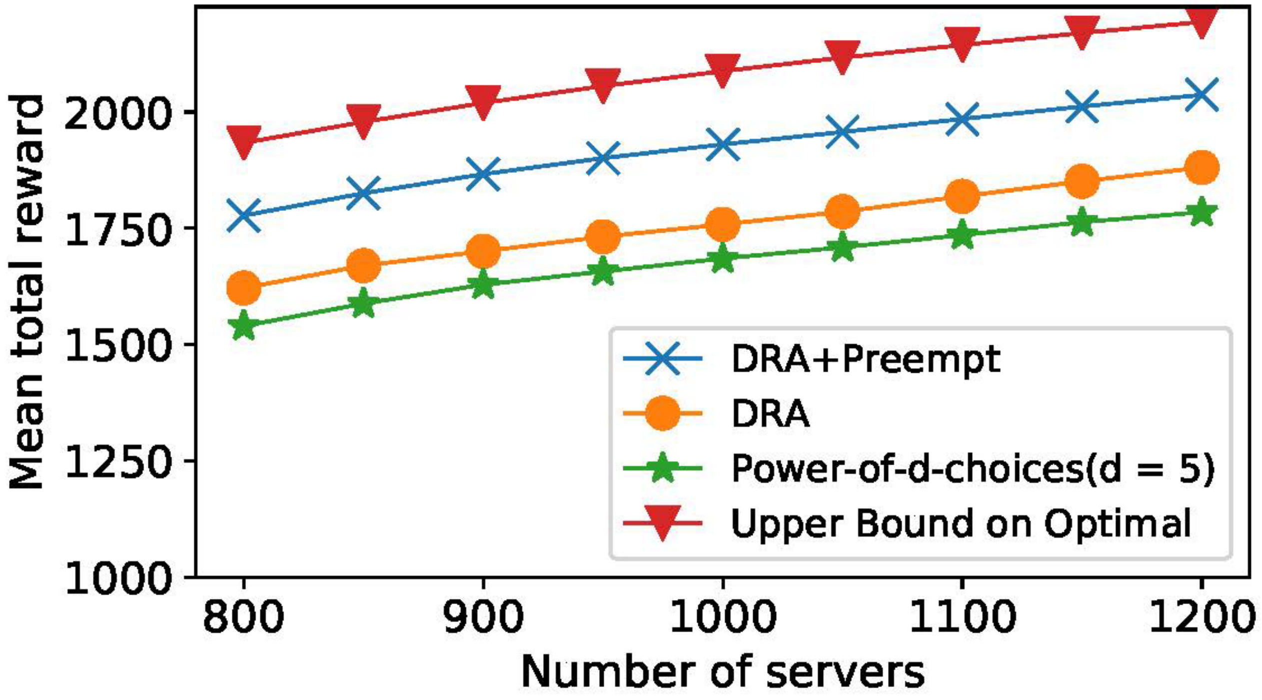
- 6VM types one per pair
- Rewards: $8\mathrm{vCPU} + \mathrm{GB}$
- Servers: $80\mathrm{vCPU}, 640\mathrm{GB}$
- Normalized workload ρ_j in $[0.2, 2.0]$

vCPU	
Small	Large
2 or 4 or 8	32 or 64

Computed $U^{(g)}[\boldsymbol{\rho}]/U^{\star}[\boldsymbol{\rho}]$ for each setup

- Worst ≈ 0.86 [worst in theory 0.5]
- Average ≈ 0.97
- Optimal $= 1.00$ [$23/50$ setups]

Simulations (Testing with Google Trace)



- 1 million tasks
- 3 priorities
- 8 different sizes
- $3 \times 8 = 24$ types

Thank You