Efficient Resource Selection in Cloud Environments with Volume Discounts and Group Dependencies

Authors:
Victor Toporkov
Dmitry Yemelyanov
and Artem Bulkhak

Department of Computing Technologies
National Research University “MPEI”

Introduction

- Many modern companies and cloud computing providers offer increasingly sophisticated pricing plans for their services
  - bonuses, promotions and discounts
  - multi-tier offers, sustainability bonuses, quantity and volume discounts
  - natural groupings by resource types, network connectivity and geographic distribution
- Resources selection and scheduling algorithms should account for non-linear pricing models and emerging dependencies between the available resources both in price and utility criteria

Saving Plans and Discounts

AWS Discounts

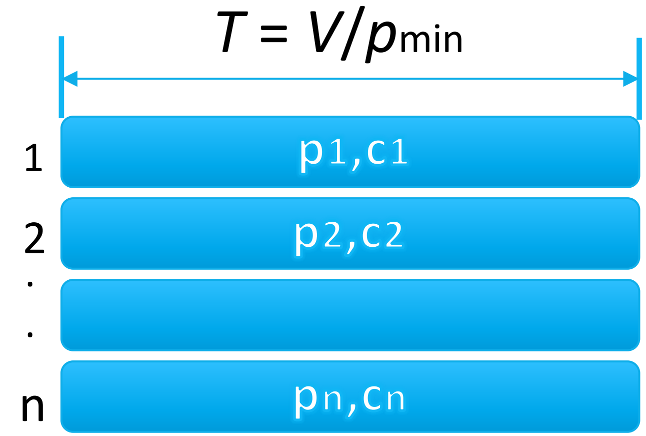
- Savings Plans
- Reserved Instances
- Reservation Marketplace
- Spot Market
- Economic Discount Program
- Private Pricing Program

All the ways you can receive discounts to help with AWS cost management

Resources Selection Request

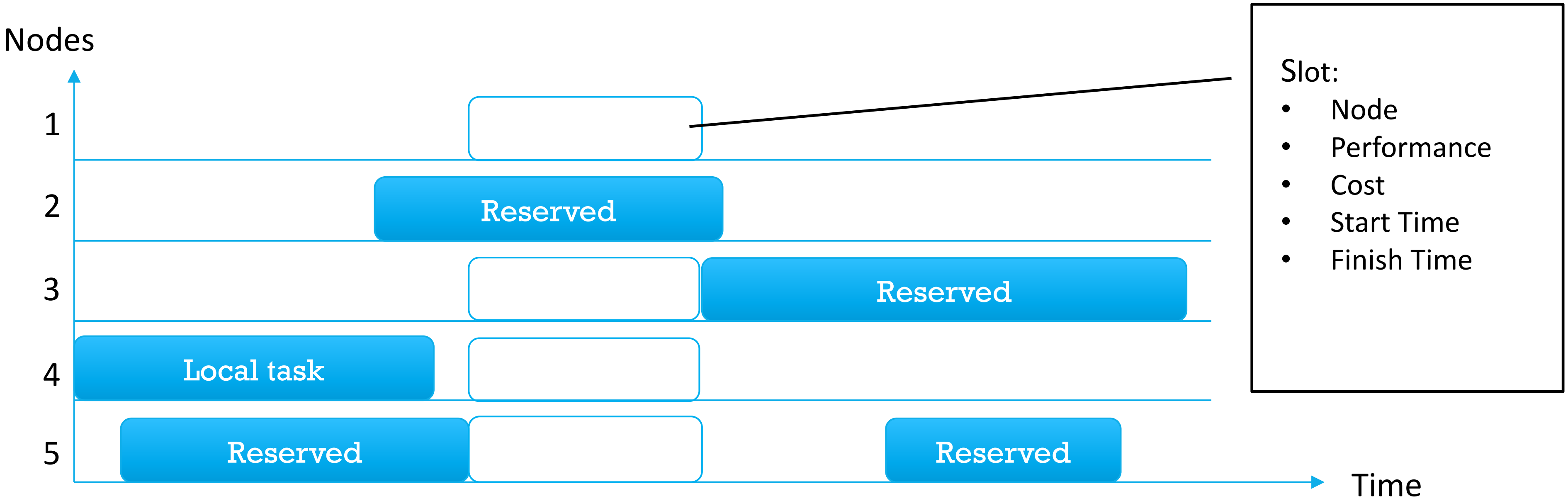
The resource requirements for a single service, parallel job or workflow are arranged into a resource request:

- *n* - number of simultaneously reserved computational nodes
- *p* - minimal performance requirement for each computational node
- *V* - computational volume for a single node task
- *C* - maximum total job execution cost (budget)

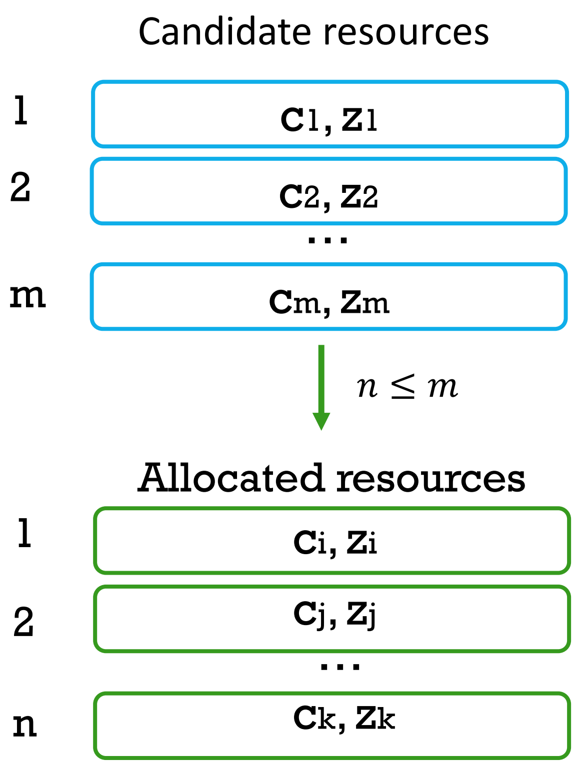


Window Search Problem

Allocate a window of computing four nodes for a time T, with requirements on nodes performance and total cost. Minimize window start time:



0-1 Knapsack Slots Subset Allocation



$$Z = \sum_{i=1}^{m} z_i \, x_i \rightarrow \max$$

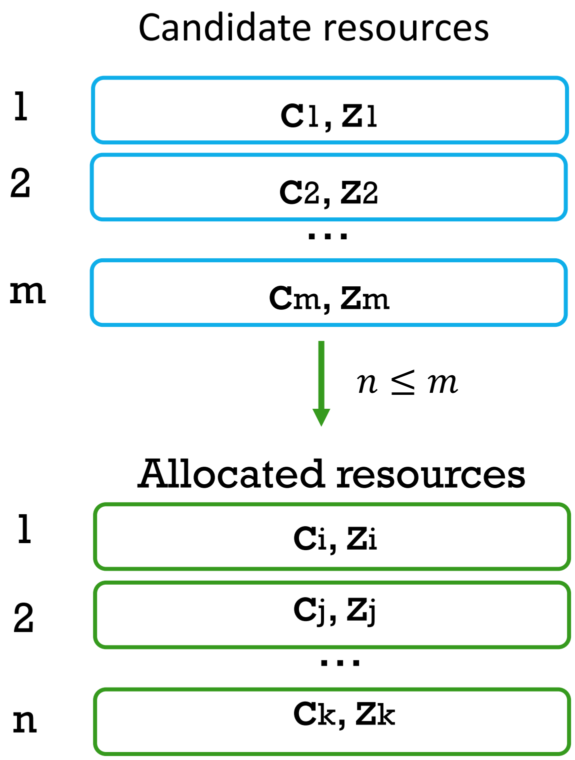
$$\sum_{i=1}^{m} c_i \, x_i \le C_j,$$

$$x_i \in \{0,1\}, i = 1..m$$

Number n of allocated resources is not limited: $n \in [0; m]$

Classic 0-1 knapsack problem

Limited Size Slots Subset Allocation



$$Z = \sum_{i=1}^{m} z_i \, x_i \rightarrow \max$$

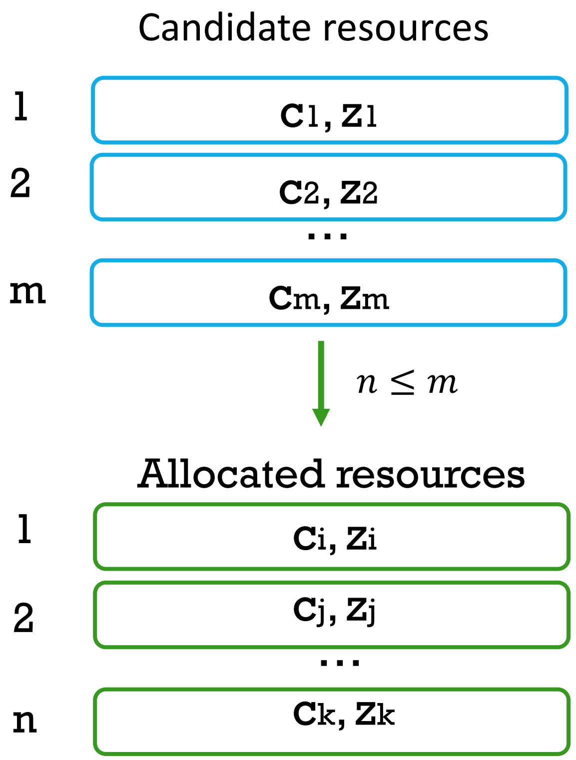
$$\sum_{i=1}^{m} c_i \, x_i \le C_j,$$

$$\sum_{i=1}^{m} x_i = n,$$

$$x_i \in \{0,1\}, i = 1..m$$

Number n of simultaneously required resources is predetermined

Interval-based Slots Subset Allocation



$$Z = \sum_{i=1}^{m} z_i \, x_i \to \max$$

$$\sum_{i=1}^{m} c_i \, x_i \leq C,$$

$$\sum_{i=1}^{m} x_i \geq n_{\min},$$

$$\sum_{i=1}^{m} x_i \leq n_{\max},$$

$$x_i \in \{0,1\}, i = 1..m$$

Interval of permissible values $[n_{\min}; n_{\max}]$ is defined for n

This problem describes a more generic resources allocation scenario

Recurrent Solution for Interval Problem

$$f_i(c,k) = \max\{f_{i-1}(c,k), f_{i-1}(c-c_i, k-1) + z_i\},$$

$$i = 1,..,m, c = 1,..,C_j, k = 1,..,n_{\max}$$

$$Z_{\max} = \max_n f_m(C,n)$$

i →

C ↓

k=1	c=3 z=3	c=2 z=1	c=4 z=6
0	-	-	-
1	-	-	-
2	-	1	1
3	3	3	3
4	3	3	6
5	3	3	6
6	3	3	6

k=2	c=3 z=3	c=2 z=1	c=4 z=6
0	-	-	-
1	-	-	-
2	-	-	-
3	-	-	-
4	-	-	-
5	-	4	4
6	-	4	7

$$O(m * n_{\max} * C)$$

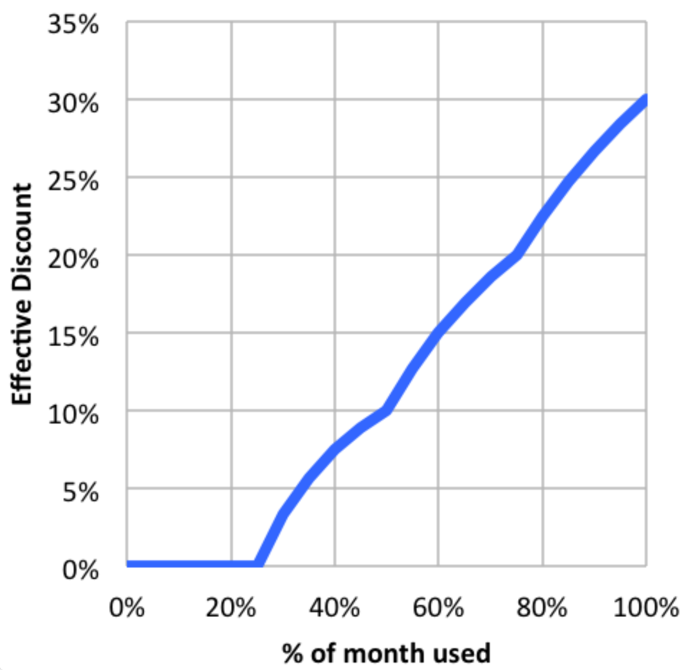
Real Life Knapsack Example

Item	Worth	Weight
Lenovo X1 Carbon (5th Gen)	40	112
10 pairs thongs	39	80
5 Underarmour Strappy	38	305
1 pair Uniqlo leggings	37	185
2 Lululemon Cool Racerback	36	174
Chargers and cables in Mini Bomber Travel Kit	35	665
The Roost Stand	34	170
ThinkPad Compact Bluetooth Keyboard with trackpoint	33	460
Seagate Backup PlusSlim	32	159
1 pair black denim shorts	31	197

https://dev.to/victoria/knapsack-problem-algorithms-for-my-real-life-carry-on-knapsack-33jj

Volume and Quantity Discounts

- Volume discount: resource provider offers a reduced price for a larger quantity of services or resources
  - Google sustained use discounts
  - Yandex commited volume of services
  - Amazon S3 data transfer

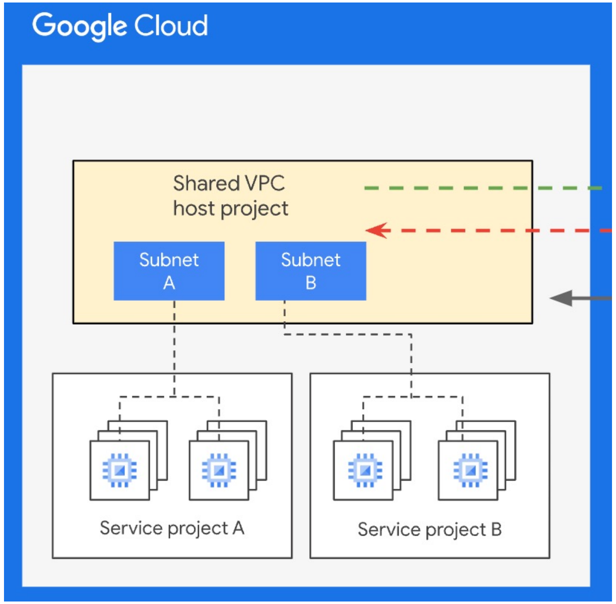


- Oracle Siebel CRM example:

A volume discount is configured as 10% discount for 5-10 items, 20% discount for 11-20 items and 30% discount for 21+ items. When ordering 23 items a customer gets no discount on items 1-4, a 10% discount on items 5-10, a 20% discount on items 11-20, and a 30% discount on items 21-23

Group Dependencies

- Locally grouped resources within a single datacenter, may have greater connectivity, consistency and greater group performance for data-intensive workloads
- Multi-cloud strategy can help manage risks, increase flexibility, optimize costs and avoid vendor lock-in
- We describe and consider aggregated price and performance benefits in terms of group dependencies
- We study a problem of an efficient multi-cloud resources selection by taking into account local group dependencies between them



Problem Statement

- The set R of cloud resources consists of multiple VM instances possibly available from different datacenters and resource providers

- Each group $G_i \in G$ is a subset of resources $r_j \in R$ with a common group dependency expressed as a special rule for aggregate cost and utility values
  - we assume that in a general case the *aggregate* cost and utility of the selected resources may differ from their *total sum* of cost and utility
  - quantity and volume discounts, connectivity benefits, enumerations

- Given the set R of VM resources and set G of non-intersecting resource groups defined over R, select a subset of $[n_{\min}; n_{\max}]$ of resources with the aggregate cost $C_a < C_{\max}$ while optimizing the aggregate utility value $U_a \rightarrow \max$.

Localized Problems and Allocation Variants

- The main idea of our approach is to solve resources selection problems for each group independently and combine their results in a higher-level recurrent solution
- For each group we calculate a Pareto-optimal set of possible allocation variants
- Each allocation variant describes one possible subset of group resources which provides aggregate utility u for an aggregate cost C:

$$Var_i = \{n_i, C_i, u_i\}$$

- Pareto-optimal set of variants can be obtained by knapsack algorithms, greedy algorithms, heuristics or brute force enumeration

Group Knapsack Algorithm GKA

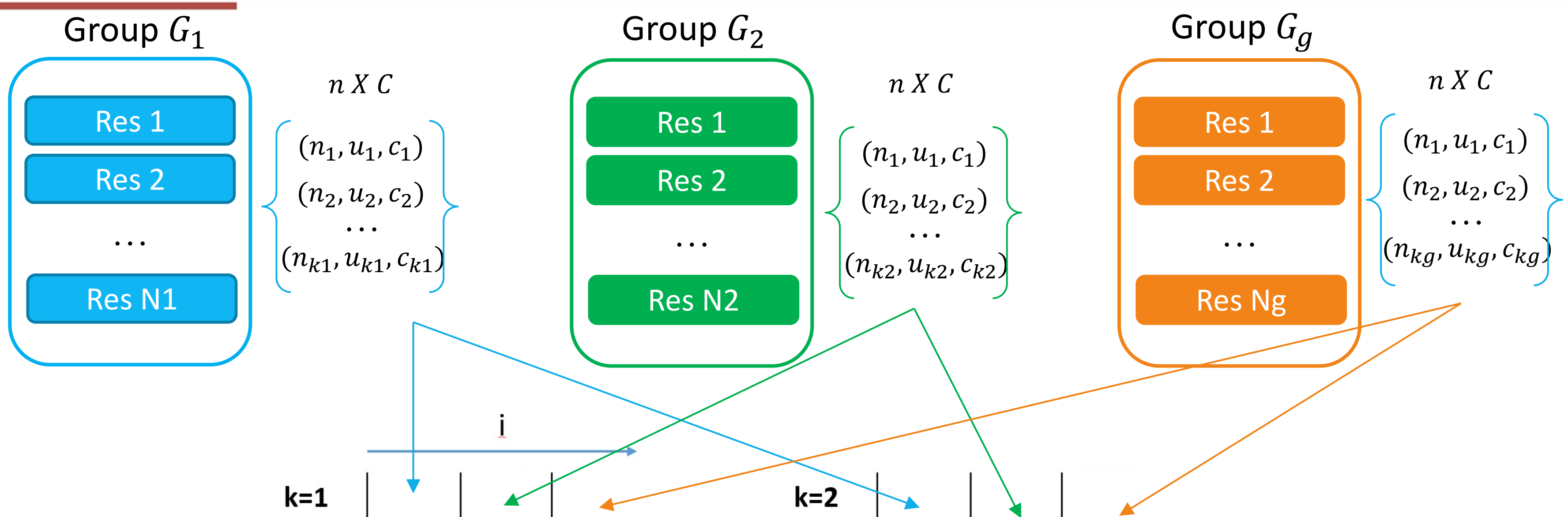
- GKA considers groups of resources G_i as enumeration items instead of individual VMs
- Instead of a single pair of characteristics u_i and c_i, each group item G_i provides a list of NV_i possible resource allocation *variants* $Var_j = (n_j, u_j, c_j)$
- GKA iterates over groups $G_i \in G$ and their variants $\{Var_j\}$ to calculate the following recurrent scheme:

$$f_i(c,n) = \max\{f_{i-1}(c,n), f_{\mathrm{i}-1}(c - c_j, n - \mathrm{n_j}) + u_j\},$$

$$i = 1,..,|G|, j = 1,..,NV_i, c = 1,..,C_{\max}, n = 1,..,n_{\max}$$

- $f_i(c,n)$ then maintains the maximum possible *aggregate* utility U achievable for a subset of n VMs combined from different variants from groups $\{G_1,..,G_i\}$ for a budget c
- Estimated computational complexity is bounded by $O(N * \mathrm{n}_{\max} * C_{\max}^2)$

Group Knapsack Algorithm GKA



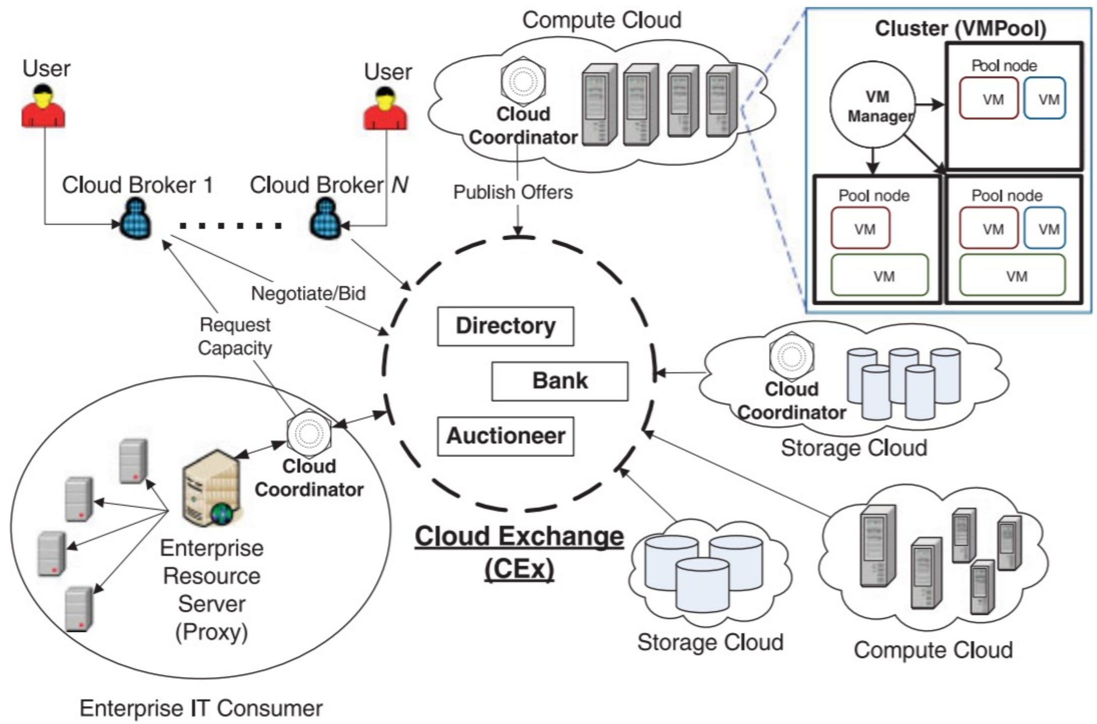
k=1			
0	-	-	-
1	-	-	-
2	-	1	1
3	3	3	3
4	3	3	6
5	3	3	6
6	3	3	6

k=2			
0	-	-	-
1	-	-	-
2	-	-	-
3	-	-	-
4	-	-	-
5	-	4	4
6	-	4	7

Simulation Environment

The simulation study is implemented in CloudSim v6 environment to comply with modern cloud resource provisioning model Physical resources

- Virtual resources
- Datacenters
- Users and cloud brokers
- Different VM allocation policies
- Pricing models
- Event-based simulation
- Extensions
  - CloudAuction, pricing models



Simulation Environment Setup

- Up to 1000 VMs and their characteristics (performance, price, RAM, etc.) are generated randomly in each simulation cycle in accordance with pre-defined tier levels

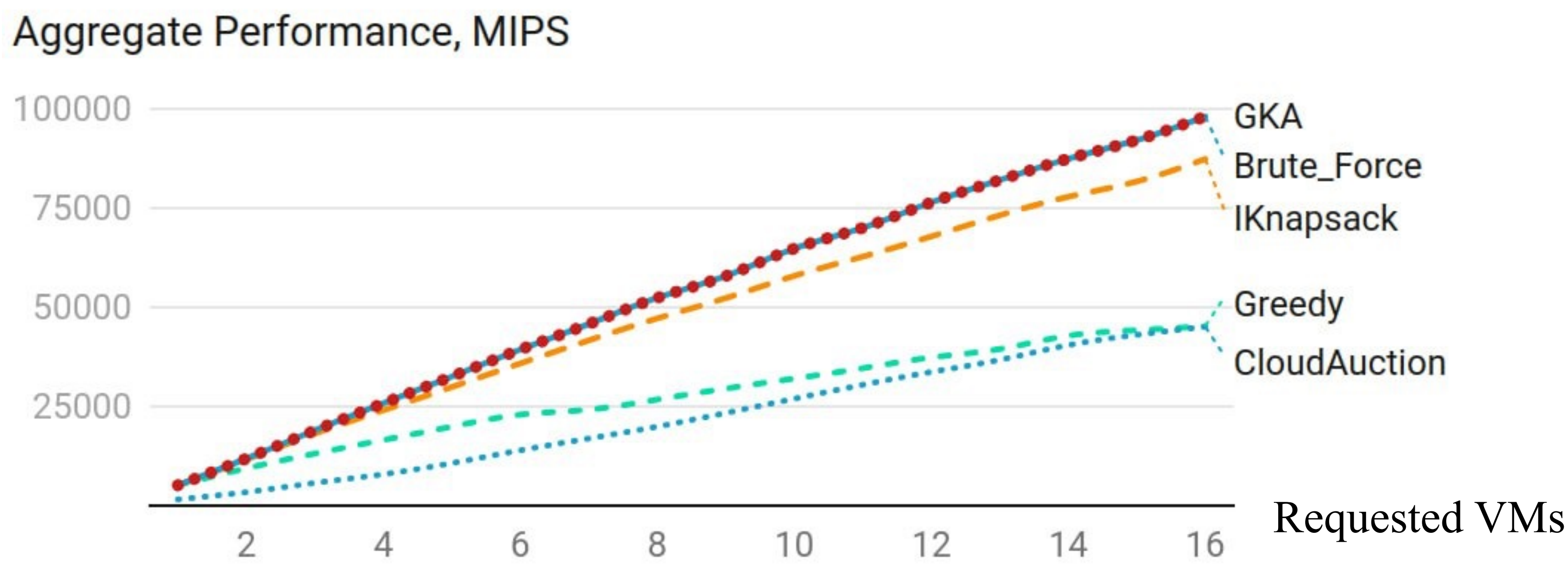
- We consider the following types of the grouping rules:
  1) groups without any quantity advantages or discounts
  2) groups with significant quantity discounts for up to 30%
  3) groups with significant quantity performance bonuses for up to 20%
  4) groups with both quantity discounts up to 20% and performance bonuses up to 10%
  - For example, for a group of 10 VMs we model quantity discount as 10% for 2-4 items, 20% for 5-7 items and 30% for 8-10 items purchased

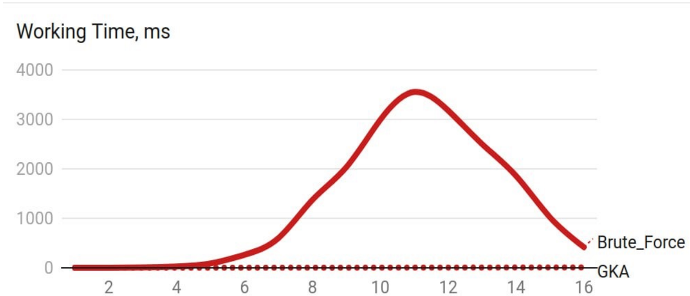
- Types of grouping rules are uniformly assigned and are equally represented in the simulated cloud environment

Considered Algorithms

- *Group Knapsack algorithm* (*GKA*) implements proposed approach to optimize VMs selection considering the resource groupings
- *Brute Force* explores all feasible combinations of available VMs and selects the best combination considering the resource groupings
- *0-1 Knapsack (lKnapsack*) implements VMs selection without information about the resource groups; group bonuses and discounts are applied to the resulting selection
- *Greedy* algorithm implements approximation of 0-1 Knapsack problem above; does not consider resource groupings; group bonuses and discounts are applied to the resulting selection
- *CloudAuction* is CloudSim extension which implements double auction algorithm; group bonuses and discounts are applied to the resulting selection

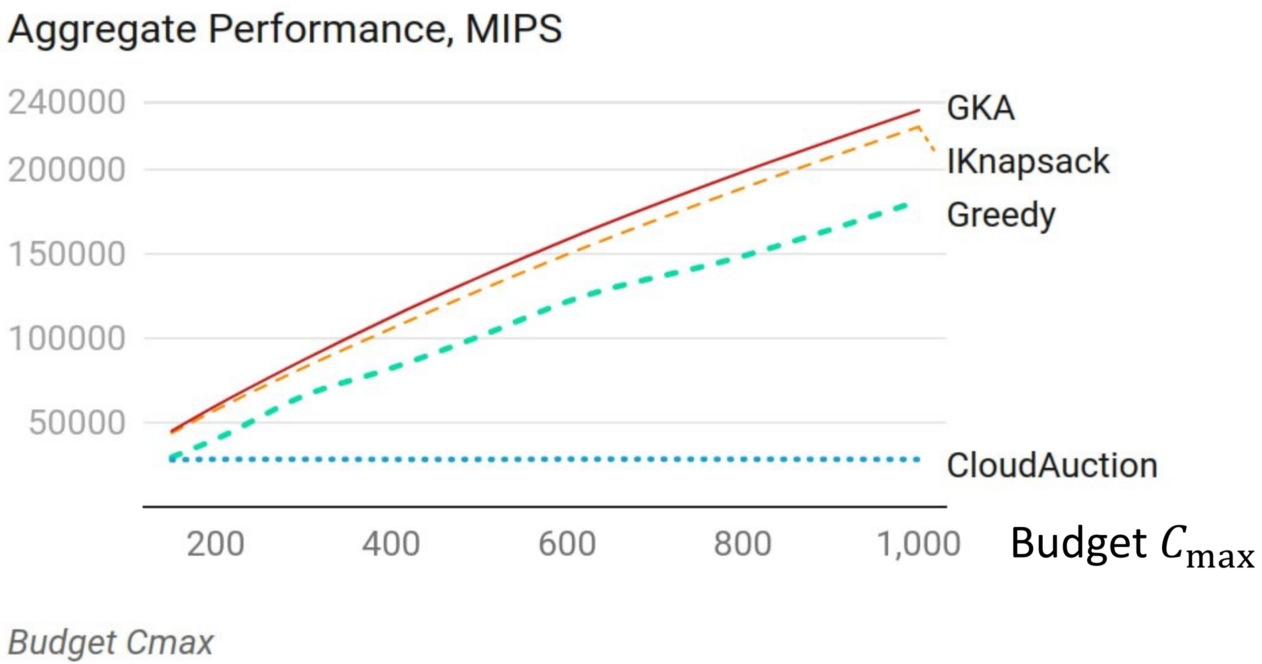
Brute Force Comparison (small environment with N = 25 VMs)





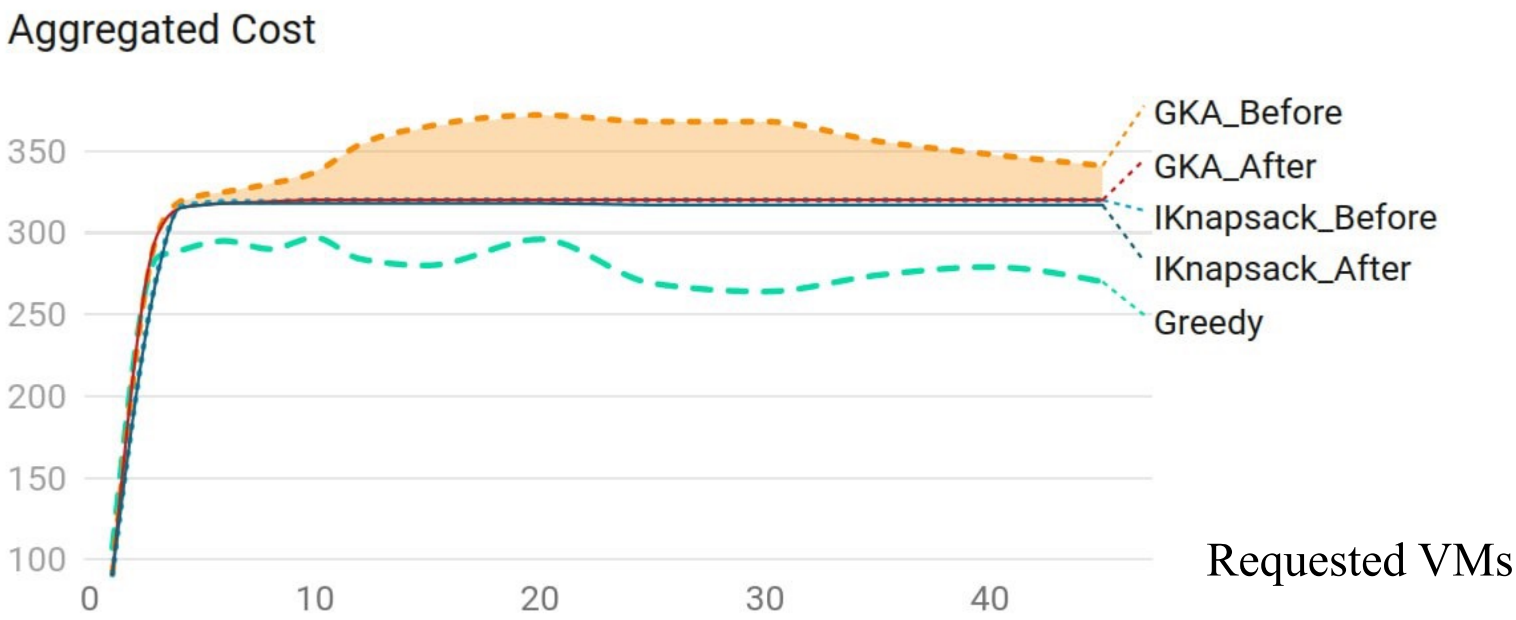
- *Brute Force* and *GKA* showed the identical best results in all simulation runs
- *IKnapsack* provided up to 12% less aggregate performance; *Greedy* – up to 50% less
- Brute Force required 1000 times more working time

GKA Performance Study ($n = 20$, N = 1000 VMs)



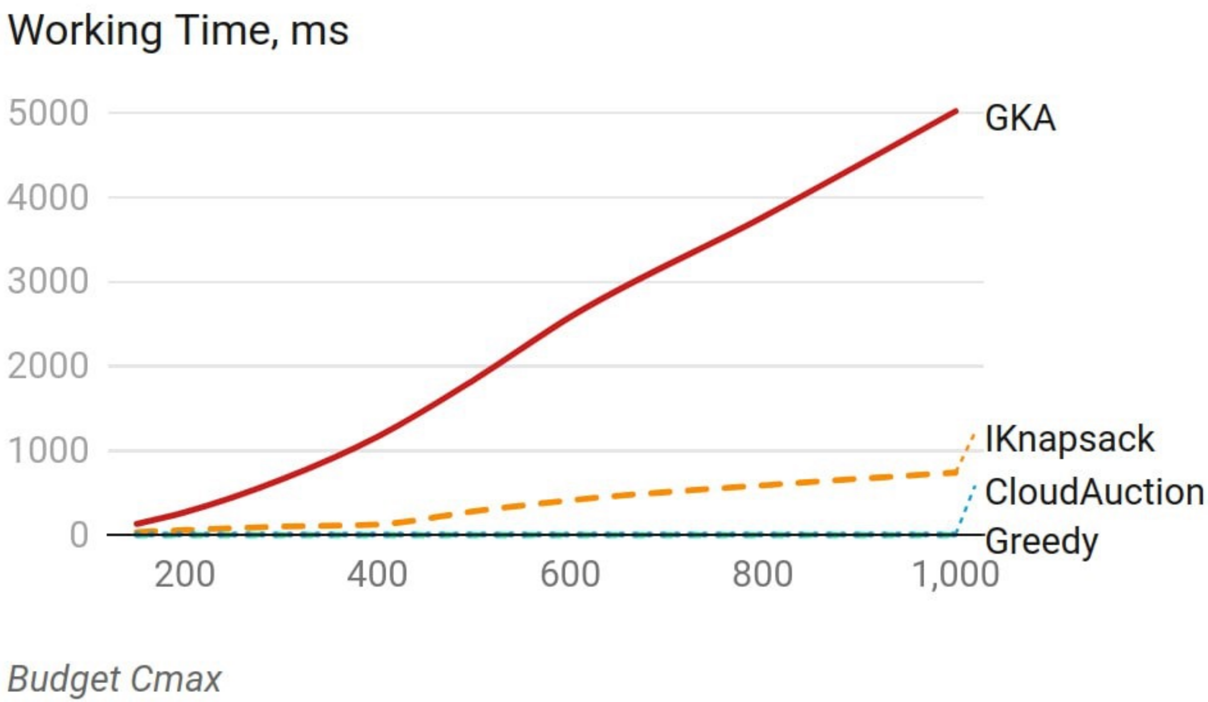
- *GKA* provides nearly 6% higher aggregate performance compared to *IKnapsack* and 25% higher compared to *Greedy*
- *CloudAuction* selects the minimally suitable VMs in terms of price/quality ratio

GKA Cost Study (N = 1000 VMs)



- *GKA* used up to 116% of $C_{\max}$ budget, which resulted in 100% after the group discounts
- *GKA* used 100% of allocated budget resulting in 99% after the group discounts
- Greedy heuristic failed to use even the entire budget $C_{\max}$

GKA Time Study (N = 1000 VMs)



- The working time of the *GKA* grows faster than that of *IKnapsack*
- The dependence on $C_{\max}$ is less than quadratic

Conclusion

- We studied the problem of efficient selection of cloud resources considering their localized group relations and dependencies
- CloudSim package was used to simulate cloud environments with up to 30% quantity discounts and and 20% performance bonuses when selecting VMs from a single datacenter group
- The proposed Group Knapsack algorithm (GKA) provides accurate solution identical to the brute force, while the advantage over other algorithms reaches 5-25% by the target optimization criterion

In further research will address and analyze more complex relationships between the available cloud resources, including non-localized dependencies

Thank you for attention!

Russian Supercomputing Days
2024

International Scientific Conference
September 23 – 24

Second Extremely Parallel Day