# PageRankVM

## A PageRank Based Algorithm with Anti-Collocation Constraints for Virtual Machine Placement in Cloud Datacenters

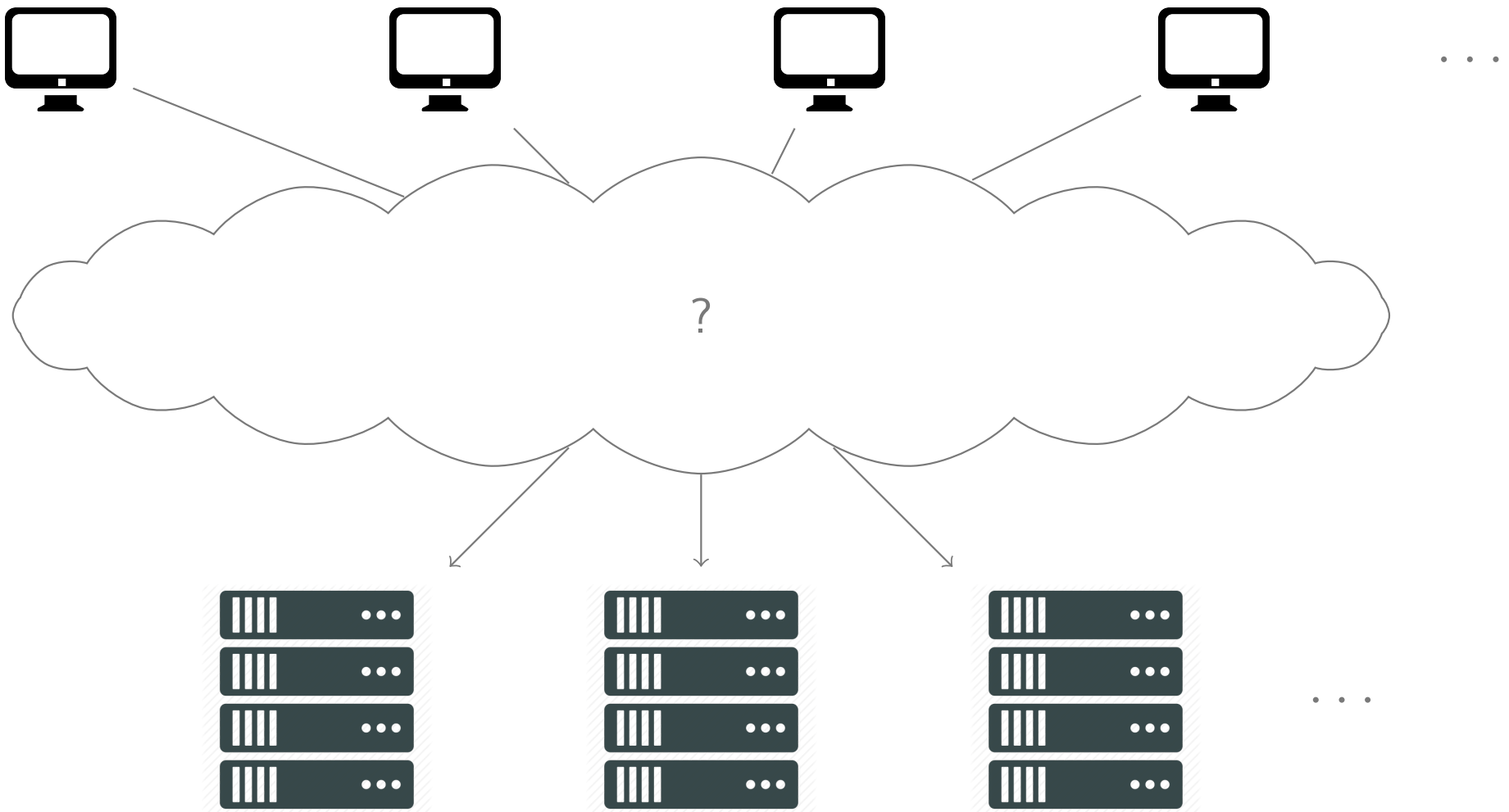
Zhuozhao Li  Haiying Shen  Cole Miles

University of Virginia

July 3, 2018

# Virtual Machine Placement

In a cloud datacenter, one wants to satisfy demands for varying types of virtual machines while utilizing as few physical machines as possible.



# Virtual Machine Placement

## Problem Statement

Model both VMs and PMs as vectors, where each dimension is a different resource.

Example: Write machine resources as $\begin{pmatrix} CPU \\ Memory \\ Disk \end{pmatrix}$

$$VMs = \left\{ \begin{pmatrix} 1 \\ 1 \\ 2 \end{pmatrix}, \begin{pmatrix} 2 \\ 3 \\ 2 \end{pmatrix} \right\} \qquad PMs = \left\{ \begin{pmatrix} 4 \\ 4 \\ 4 \end{pmatrix} \right\}$$

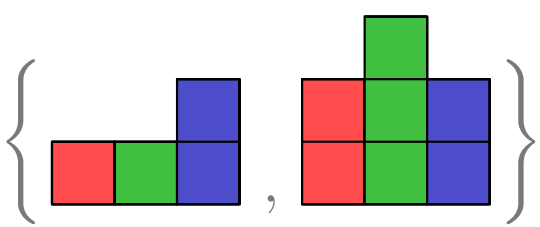
# Virtual Machine Placement

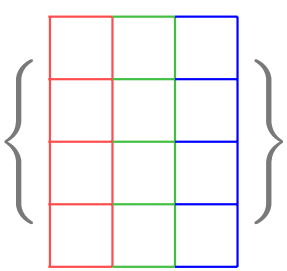
## Problem Statement

Model both VMs and PMs as vectors, where each dimension is a different resource.

Example: Write machine resources as $\begin{pmatrix} CPU \\ Memory \\ Disk \end{pmatrix}$

$$VMs = \left\{ \begin{pmatrix} 1 \\ 1 \\ 2 \end{pmatrix}, \begin{pmatrix} 2 \\ 3 \\ 2 \end{pmatrix} \right\} \qquad PMs = \left\{ \begin{pmatrix} 4 \\ 4 \\ 4 \end{pmatrix} \right\}$$





# Virtual Machine Placement

## Anti-Collocation Constraints

In addition, we want to satisfy sets of *anti-collocation constraints* on certain resources.

**Solution**

Treat anti-collocation constrained resources as multiple dimensions in the resource vector

# Virtual Machine Placement

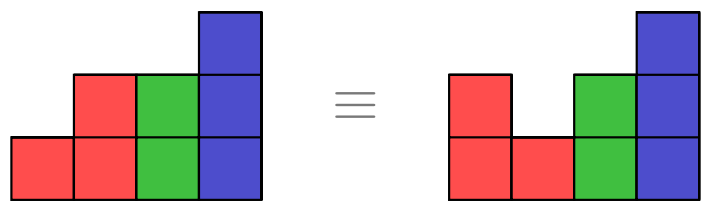
## Anti-Collocation Constraints

In addition, we want to satisfy sets of *anti-collocation constraints* on certain resources.

**Solution**

Treat anti-collocation constrained resources as multiple dimensions in the resource vector

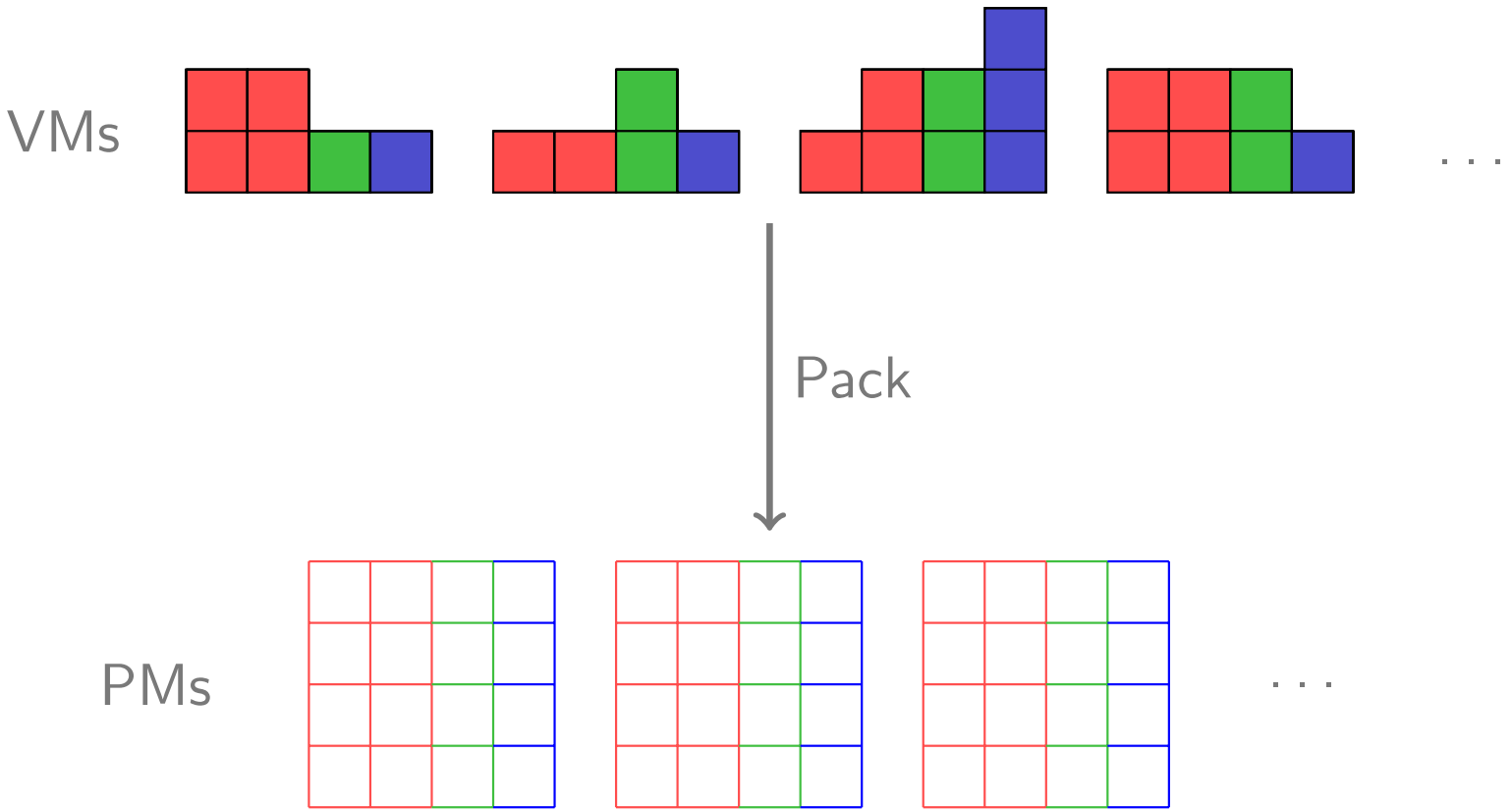
Permuting same resource type OK:



# Virtual Machine Placement

## Vector Bin Packing

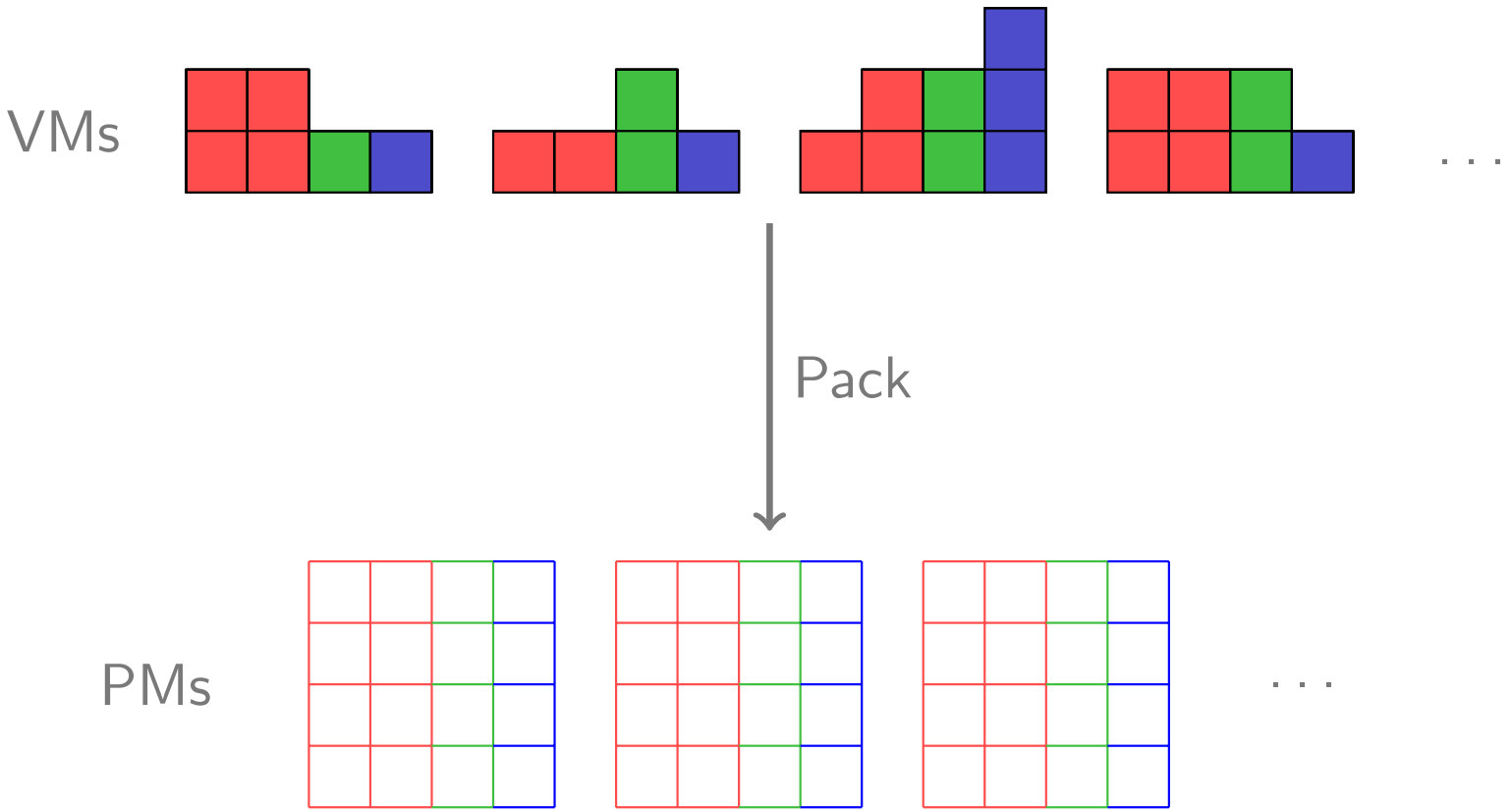
The problem of allocating VMs to physical machines in datacenters is analogous to the vector bin packing problem



# Virtual Machine Placement

## Vector Bin Packing

The problem of allocating VMs to physical machines in datacenters is analogous to the vector bin packing problem




Issue: Vector Bin Packing is an NP-Hard problem!

# Virtual Machine Placement

## Vector Bin Packing - Differences

However, the VM packing problem deviates from standard vector bin packing in a few ways

- Don't know full set of vectors to pack at a single time
- Vectors are pulled from a fixed, known set we decide upon
- Allowed to permute some dimensions of the vector (CPU cores)

# Previous Solutions

Try to maximize PM usage or minimize PM variance

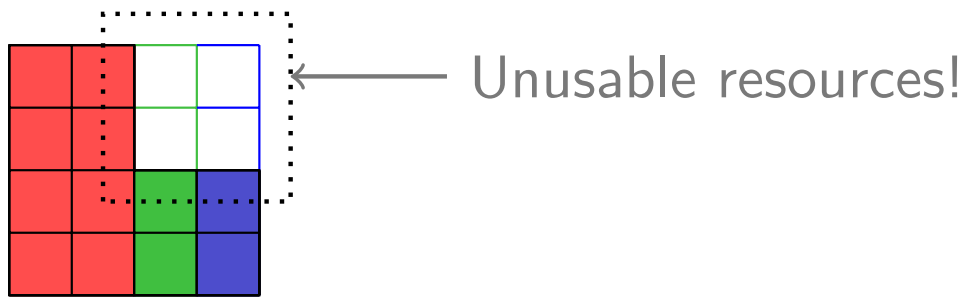
- First Fit - Assign VM to first PM it fits into
- First Fit Decreasing Sum - Assign VM to the PM whose resource utilization is highest
- CompVM (Dot Product)
- Integer Linear Programming

# Previous Solutions

Try to maximize PM usage or minimize PM variance

- First Fit - Assign VM to first PM it fits into
- First Fit Decreasing Sum - Assign VM to the PM whose resource utilization is highest
- CompVM (Dot Product)
- Integer Linear Programming

Do not account for resource imbalances



# Previous Solutions

Try to maximize PM usage or minimize PM variance

- First Fit - Assign VM to first PM it fits into
- First Fit Decreasing Sum - Assign VM to the PM whose resource utilization is highest
- CompVM (Dot Product)
- Integer Linear Programming

Assigns VMs to complementary PMs, but is not aware of the set of VM types or the best final PM state

# Previous Solutions

Try to maximize PM usage or minimize PM variance

- First Fit - Assign VM to first PM it fits into
- First Fit Decreasing Sum - Assign VM to the PM whose resource utilization is highest
- CompVM (Dot Product)
- Integer Linear Programming

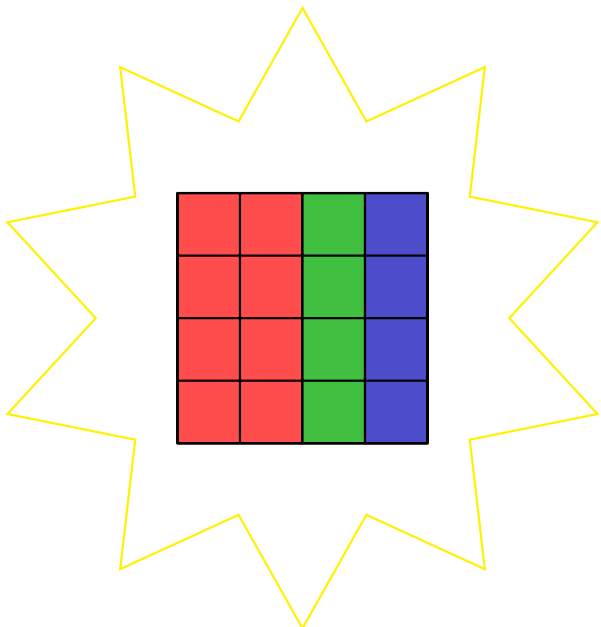
Too slow for large problem instances

# Desired Solution

**Goal**

Minimize the number of wasted resources in each dimension.

- Needs to be efficient at assignment time
- Capable of handling anti-collocation constraints
- Minimizes the total number of PMs used



# PageRankVM

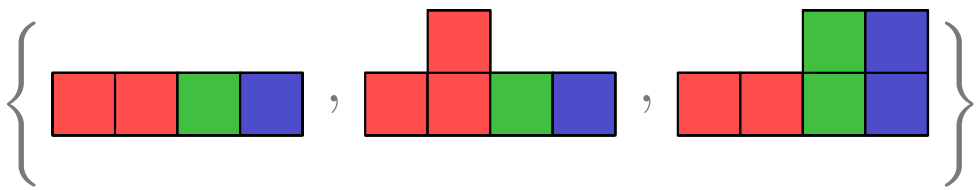
## Profile Graph

**Idea**

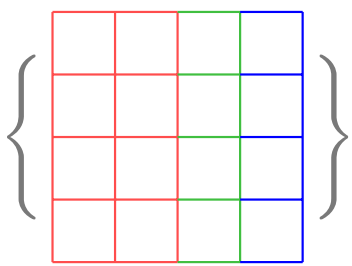
Describe each possible state of a PM as a *profile*. Link these together into a graph based on possible transitions between states.

**Example**

Let $VMs = \{[1,1,1,1],[1,2,1,1],[1,1,2,2]\}$.

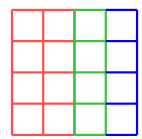


Let $PMs = \{[4,4,4,4]\}$.



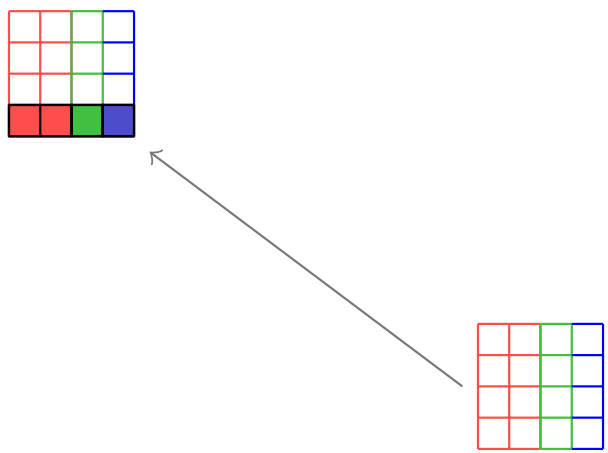
# PageRankVM

## Profile Graph



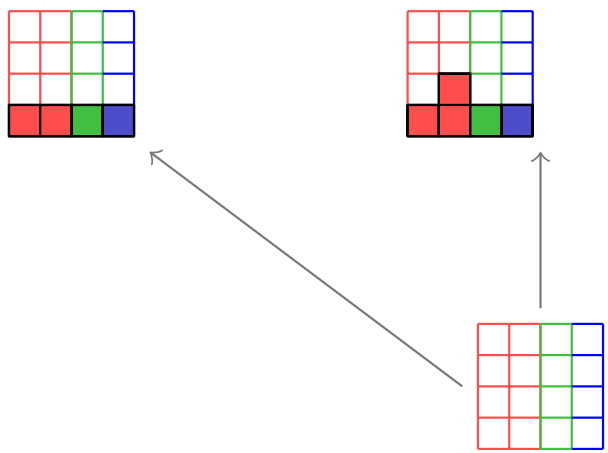
# PageRankVM

## Profile Graph



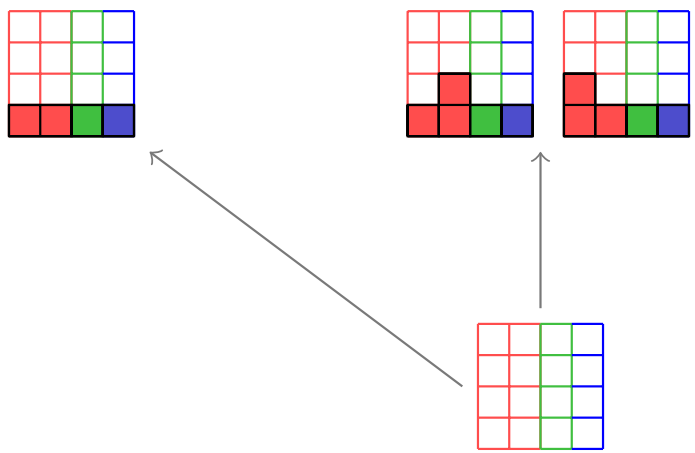
# PageRankVM

## Profile Graph



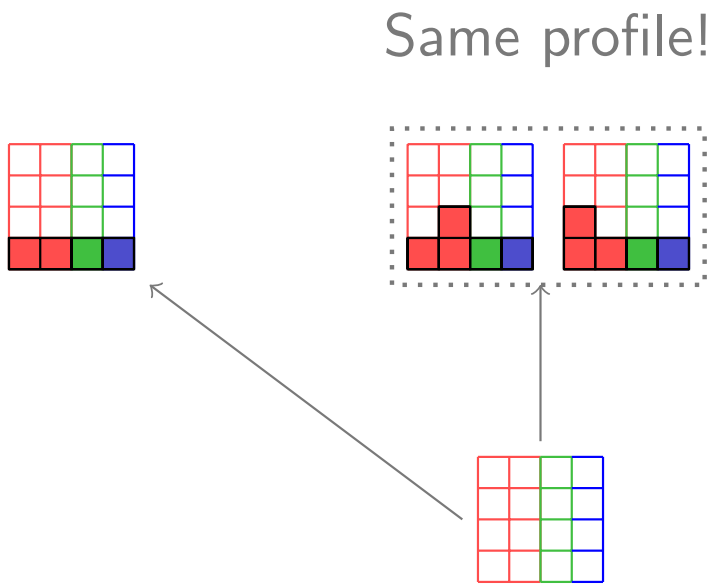
# PageRankVM

## Profile Graph



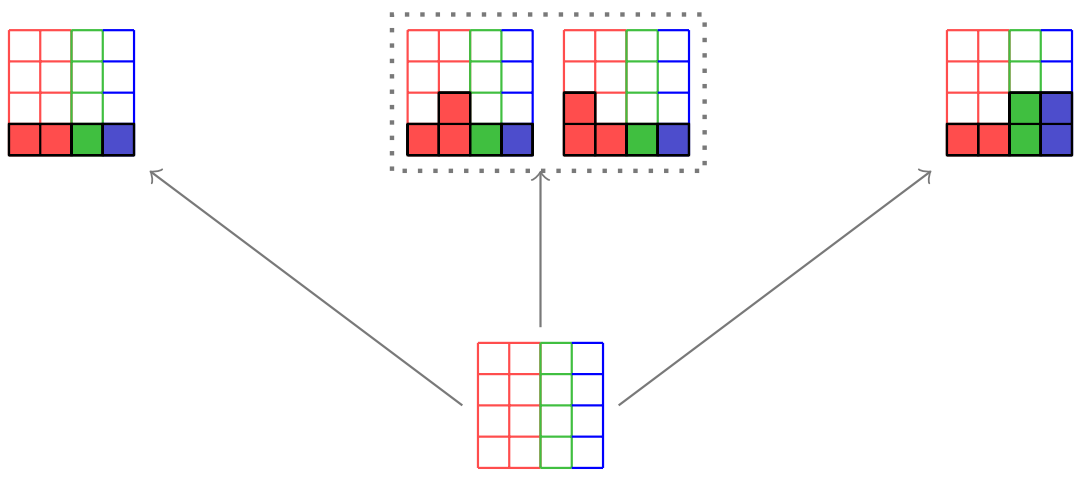
# PageRankVM

## Profile Graph



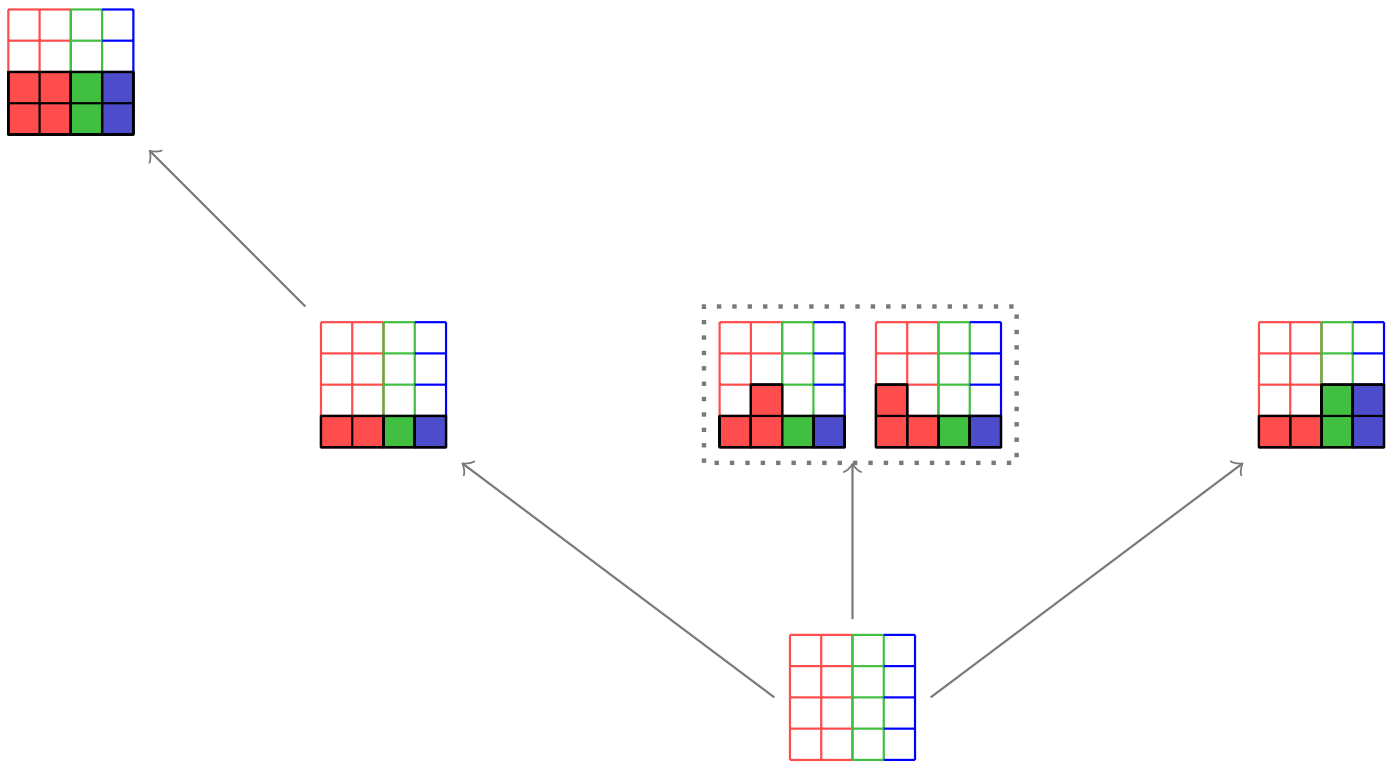
# PageRankVM

## Profile Graph



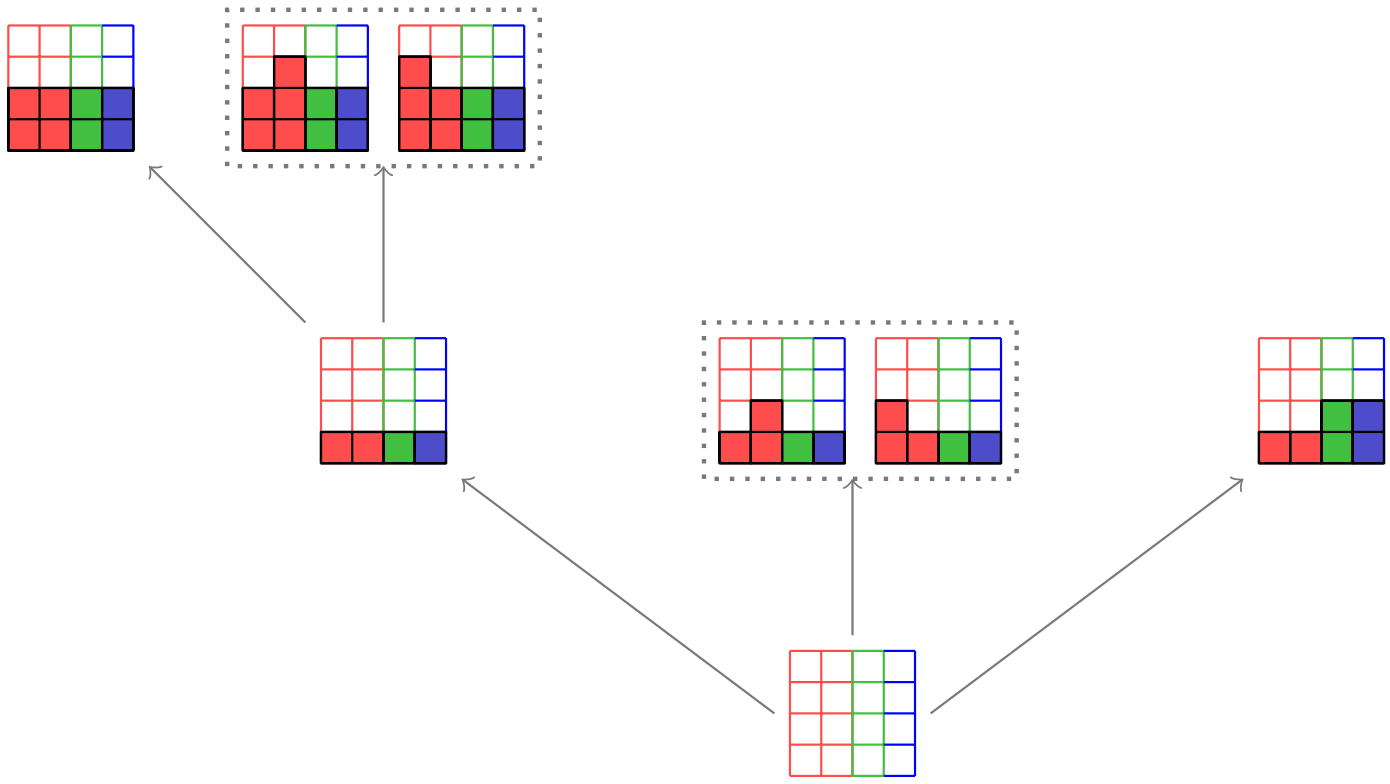
# PageRankVM

## Profile Graph



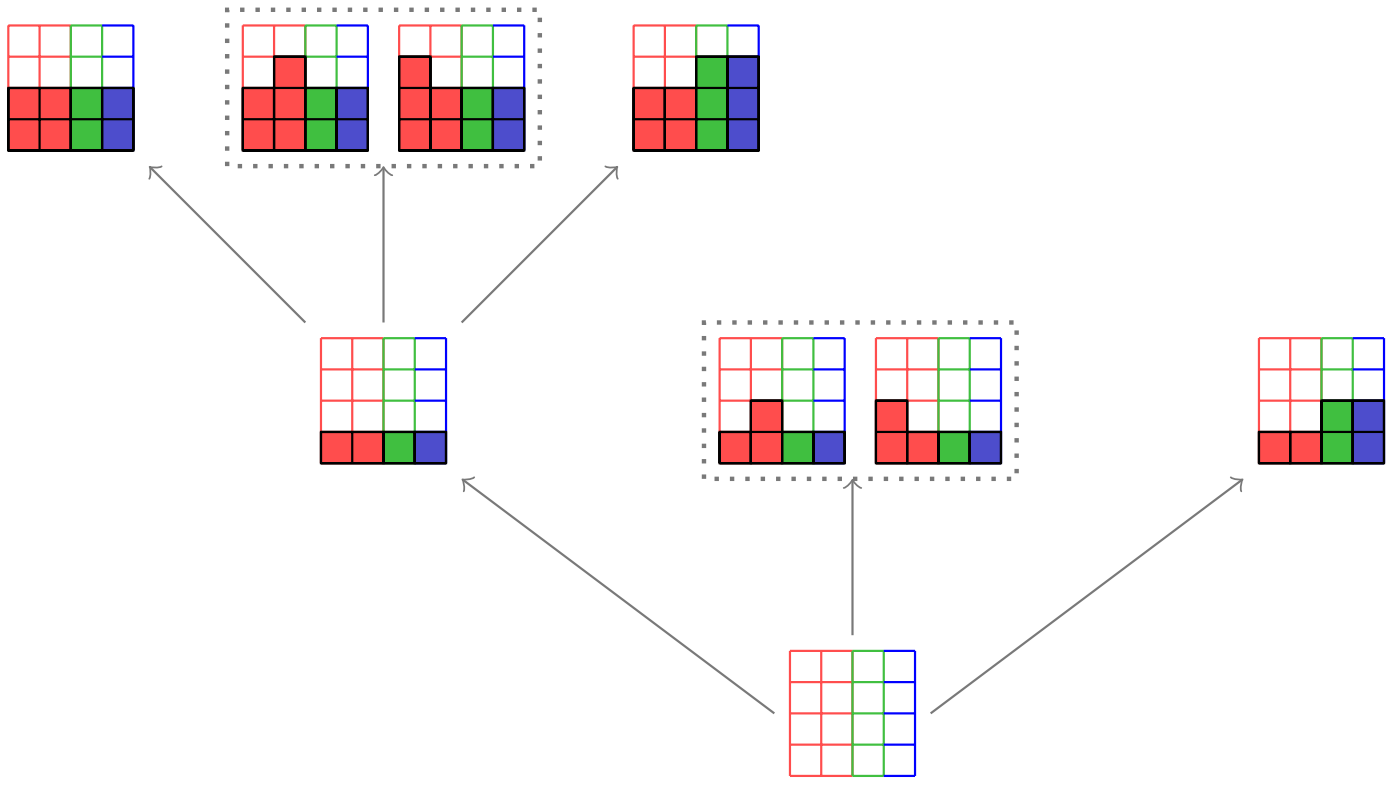
# PageRankVM

## Profile Graph



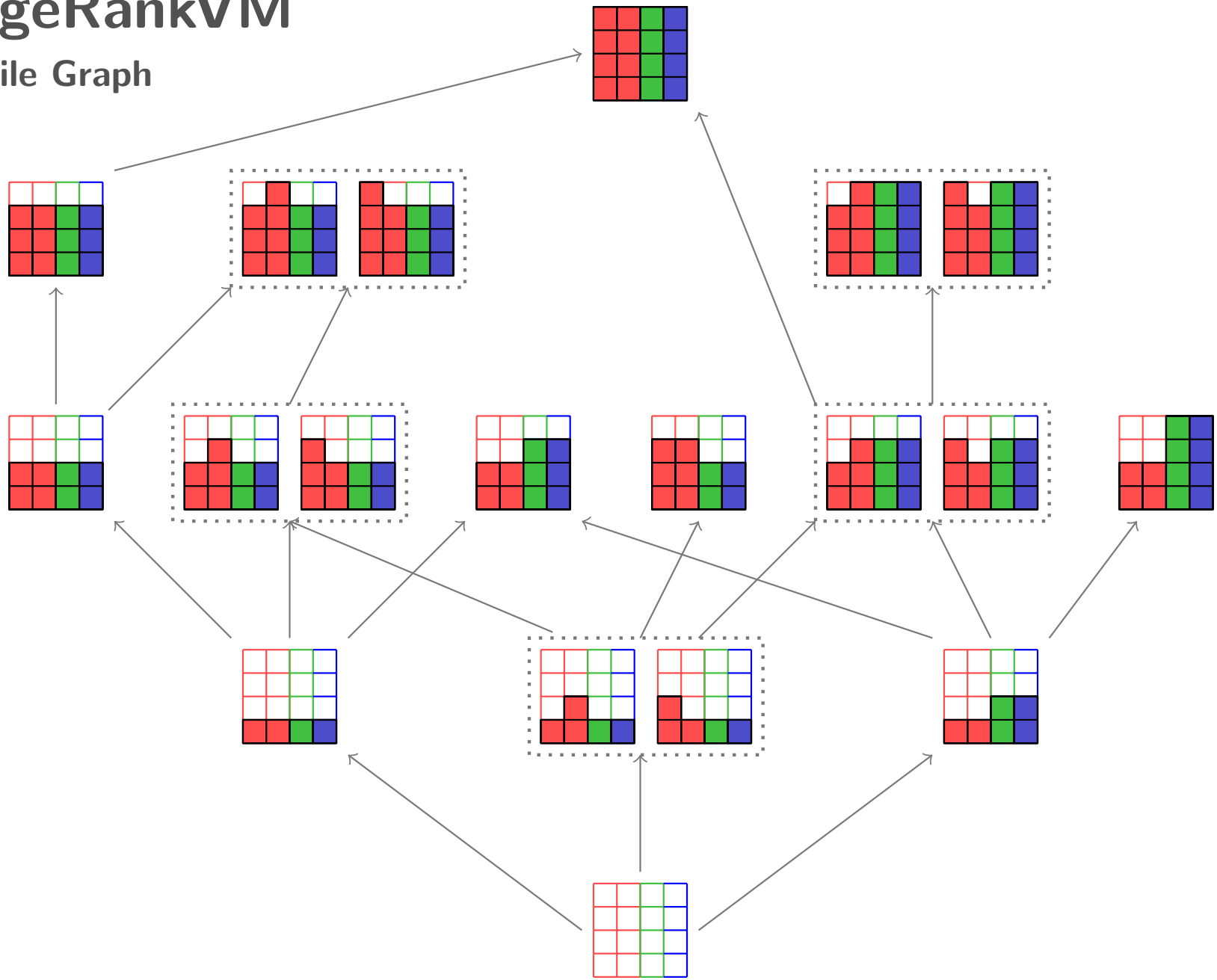
# PageRankVM

## Profile Graph

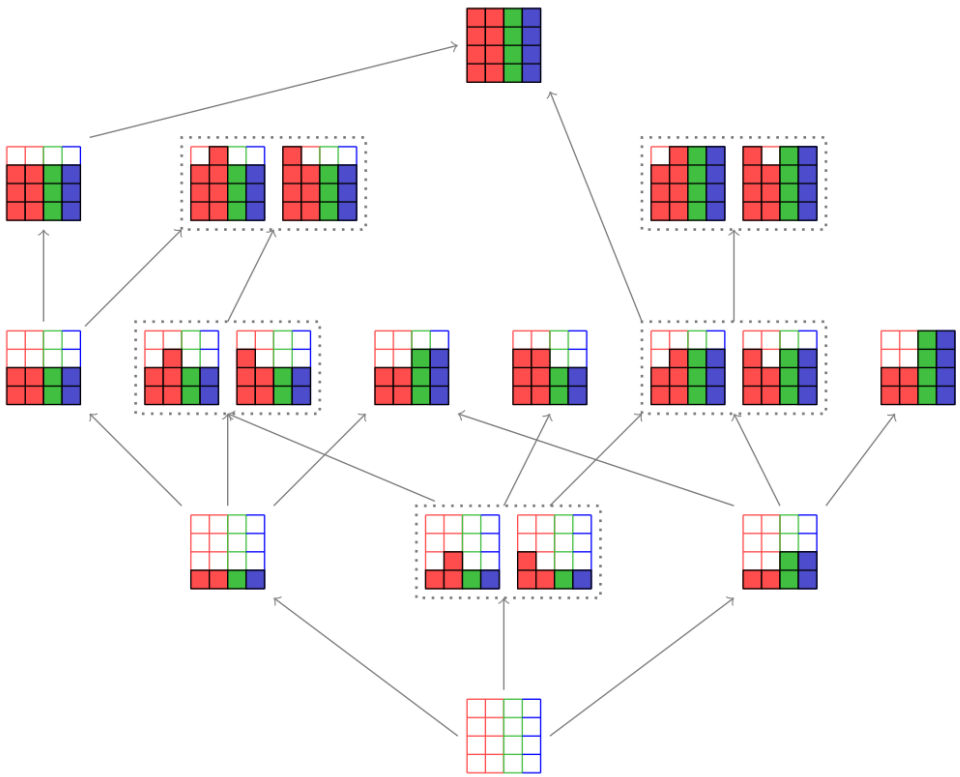


# PageRankVM

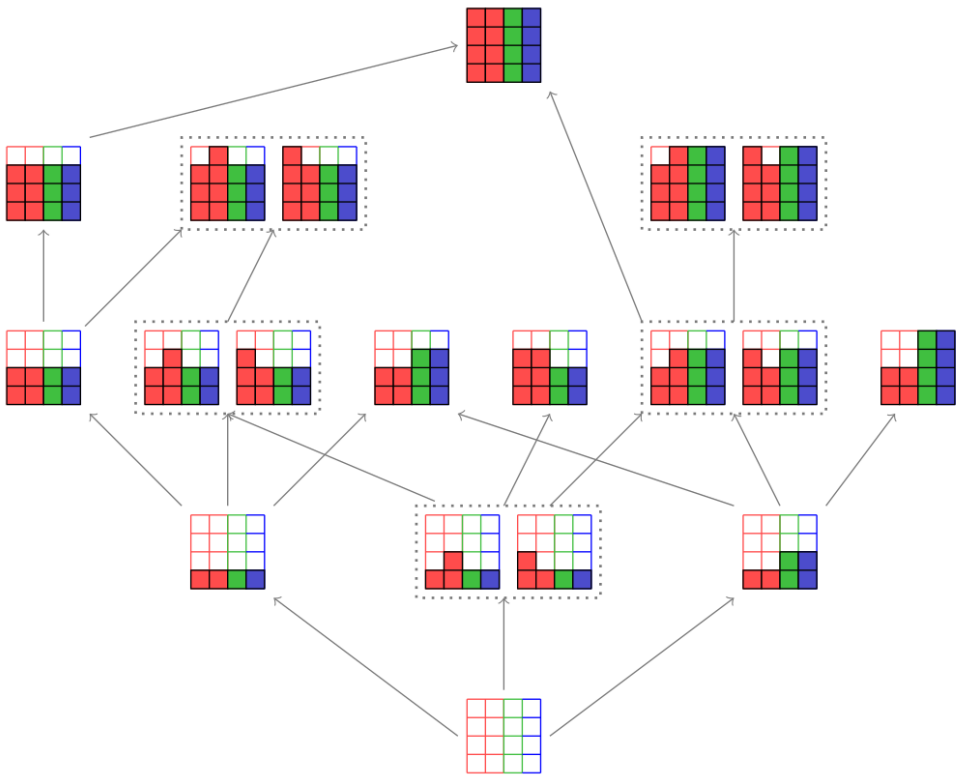
## Profile Graph



# Desired Metric Qualities



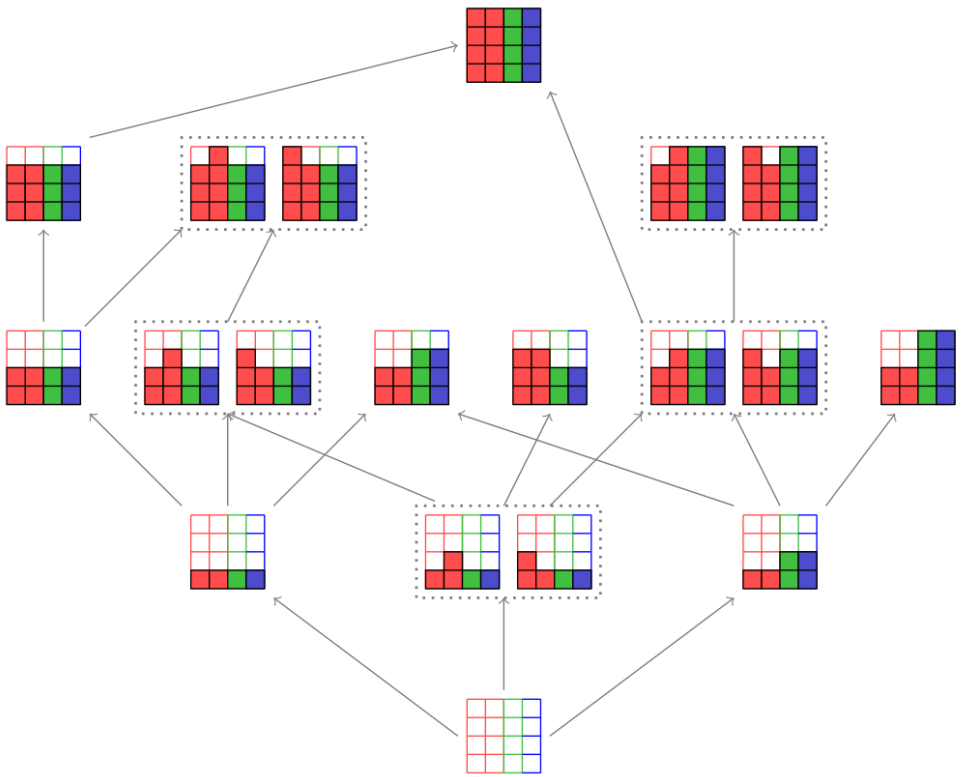
# Desired Metric Qualities



Want to rank each profile based on

- how connected it is to higher-utilization profiles
- how flexible the profile is to accommodating various VM types

# Desired Metric Qualities



Want to rank each profile based on

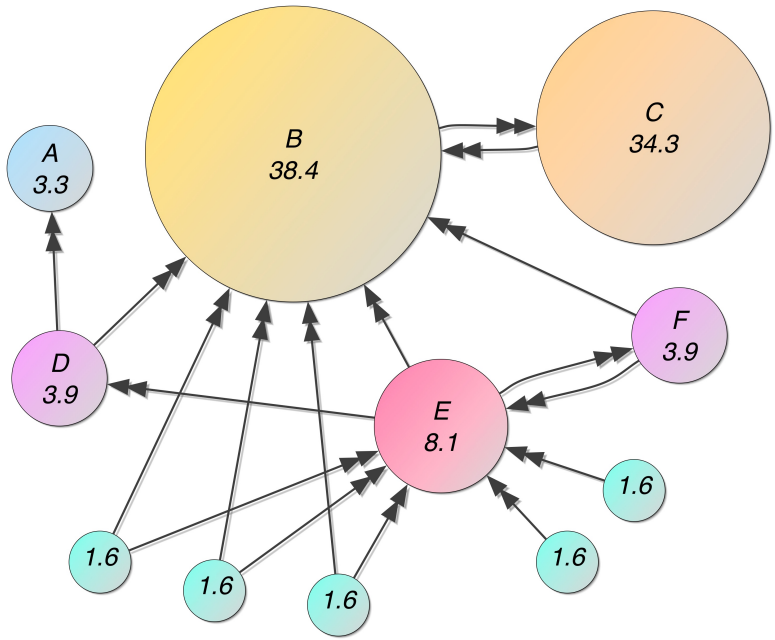
- how connected it is to higher-utilization profiles
- how flexible the profile is to accommodating various VM types

$\Longrightarrow$ PageRank!

# PageRank

Initialize PageRanks uniformly

$$PR(p_i; 0) = \frac{1}{N}$$

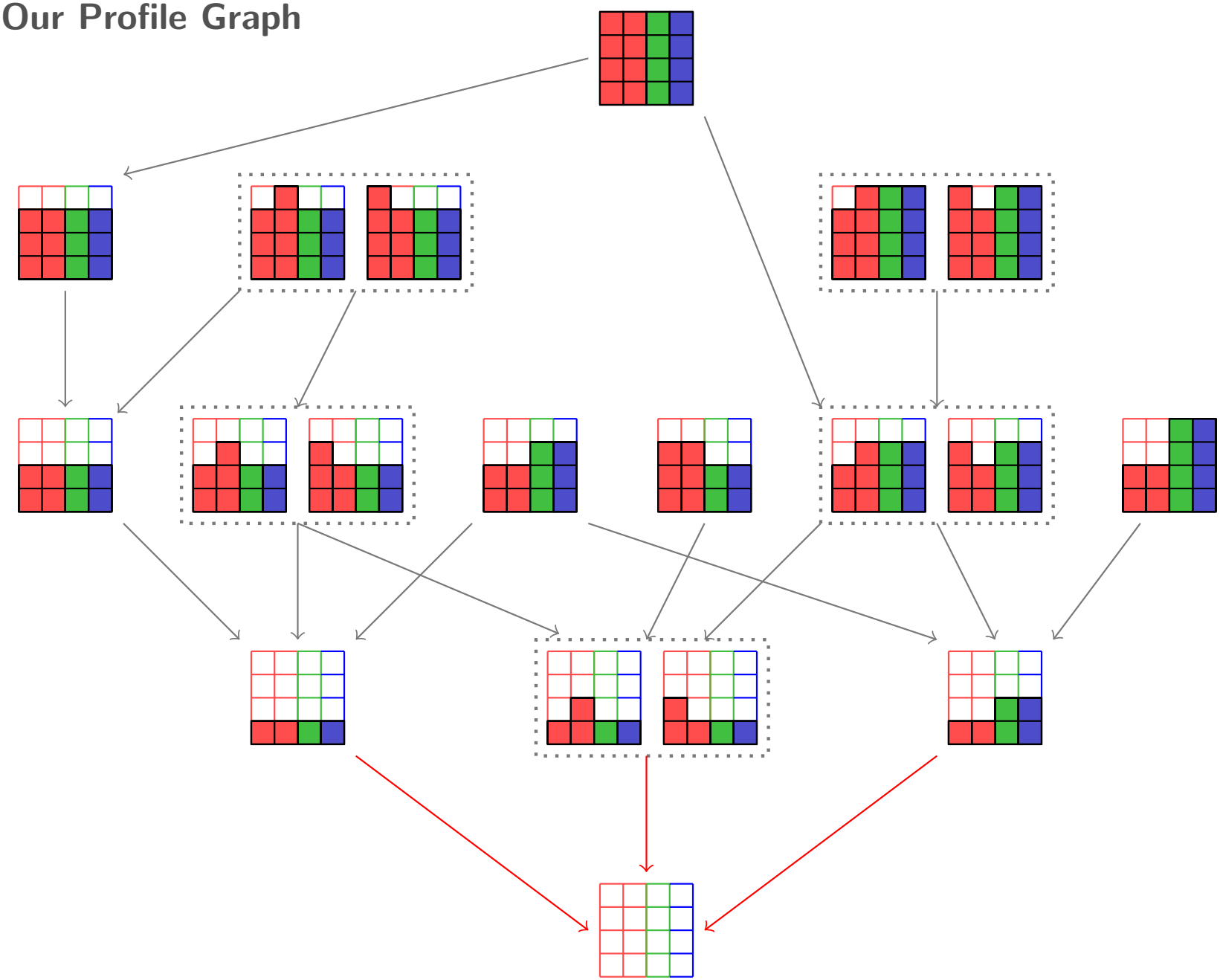



At each time step update as

$$PR(p_i; t+1) = \frac{1-d}{N} + d \sum_{p_j \in M(p_i)} \frac{PR(p_j; t)}{L(p_j)}$$

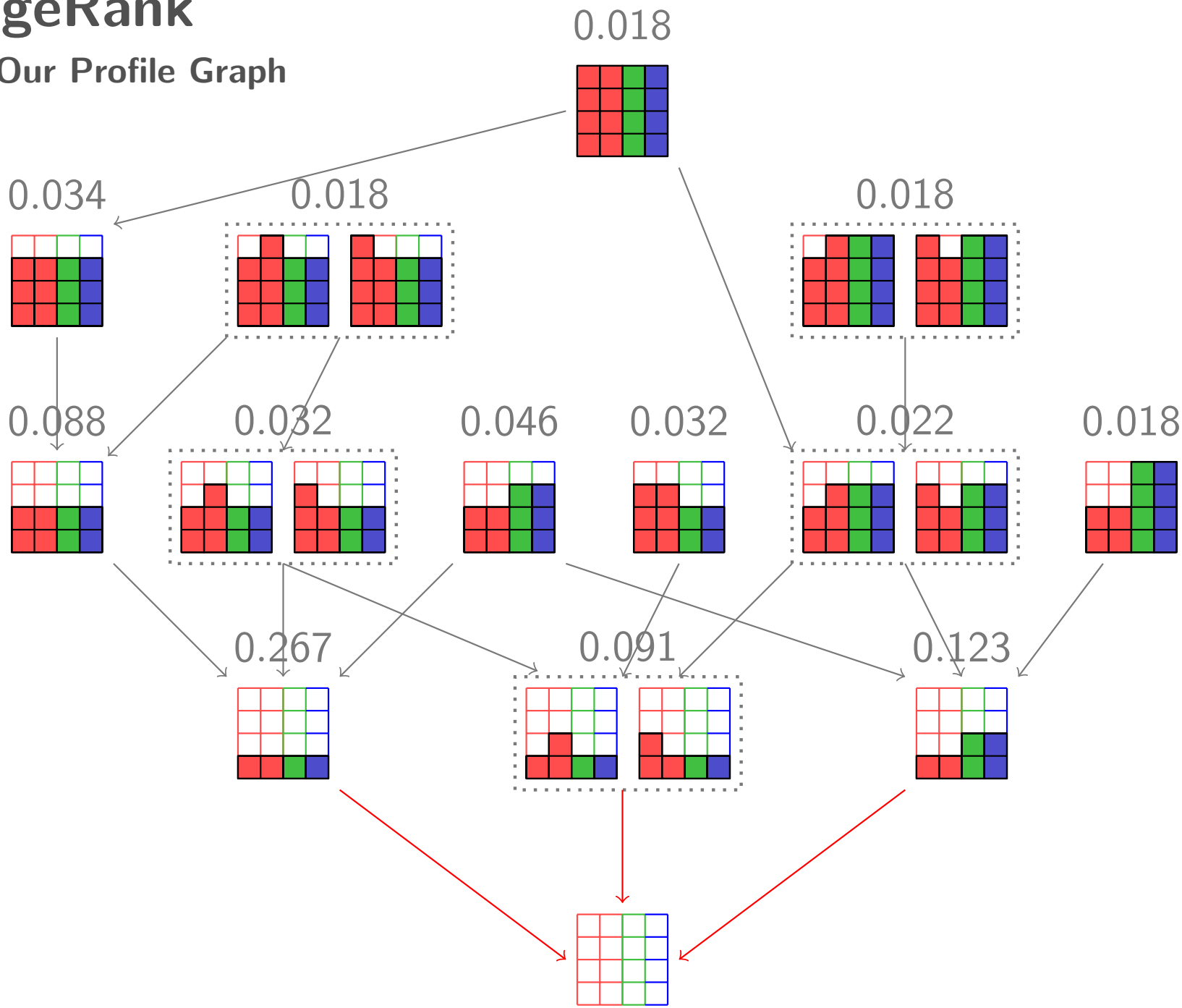
# PageRank

## On Our Profile Graph

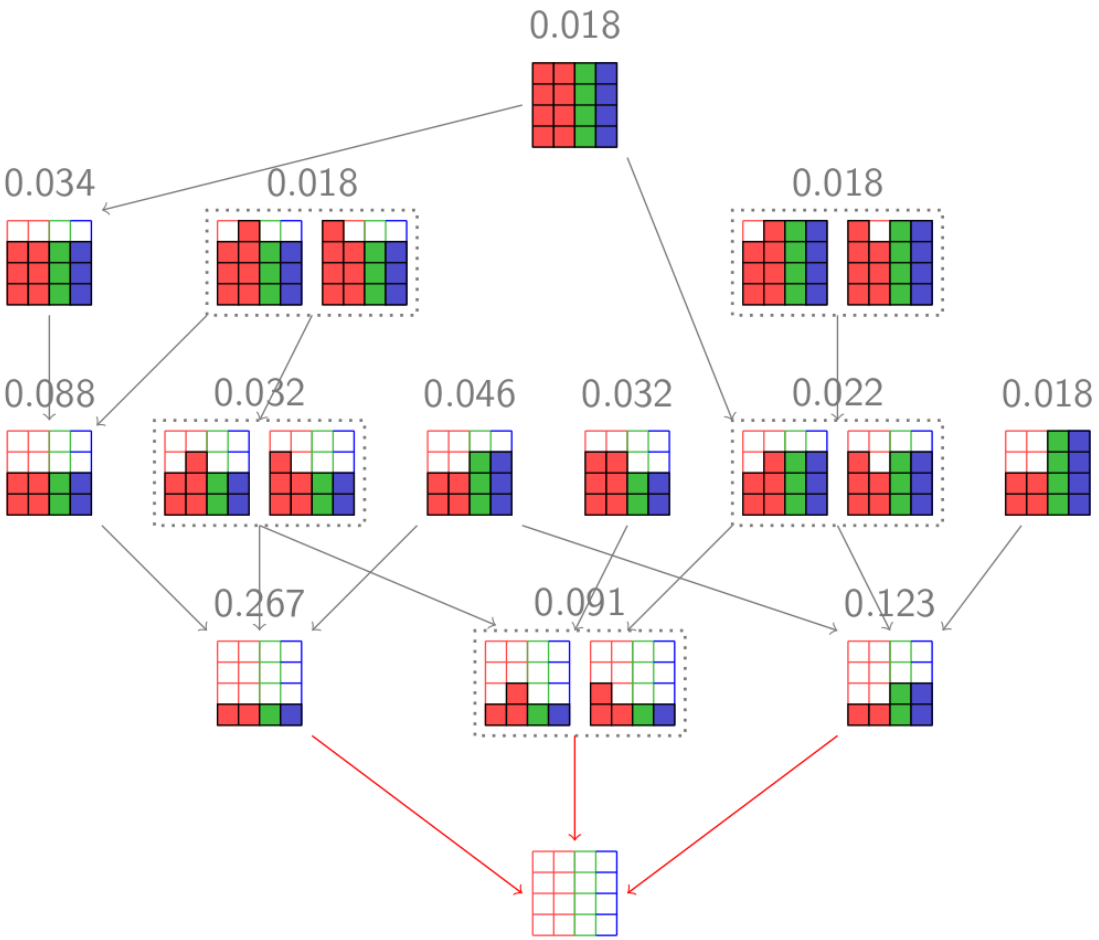


# PageRank

## On Our Profile Graph



# PageRankVM

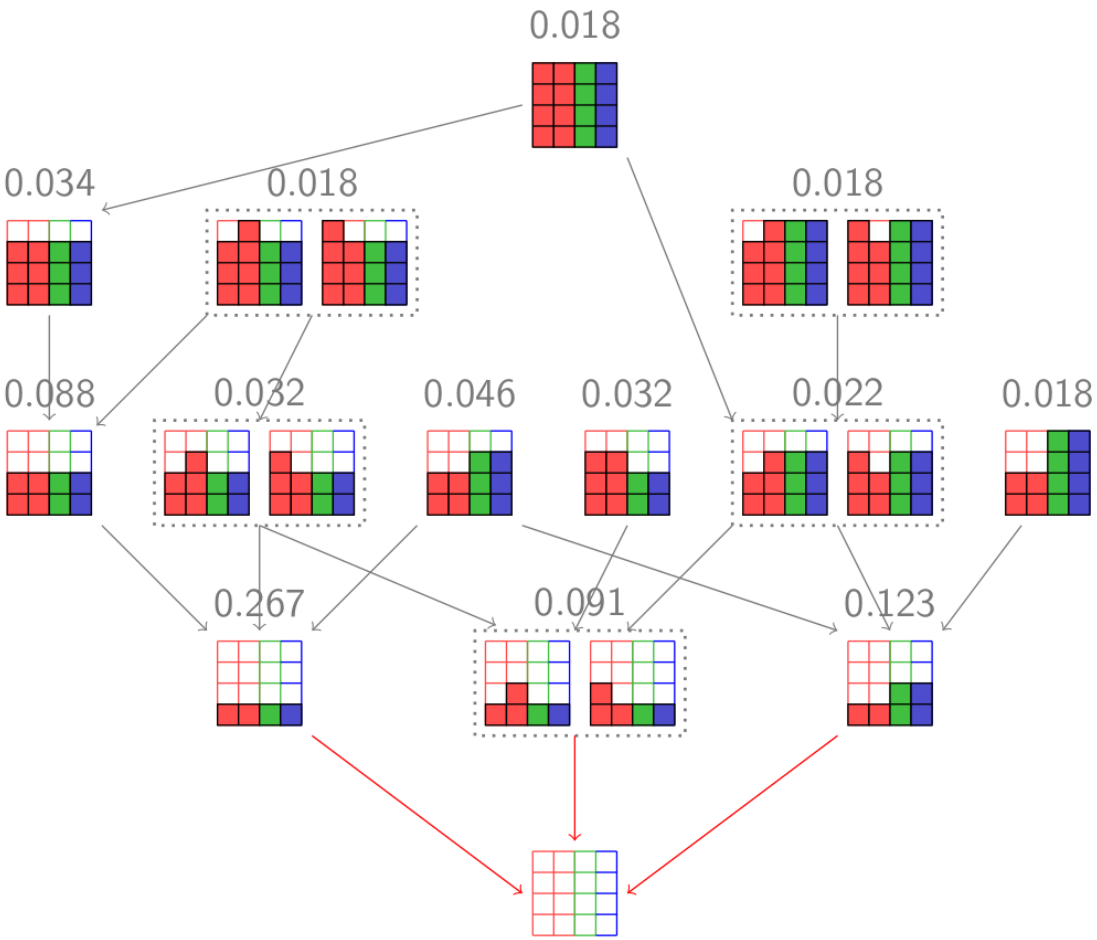



**At assignment time**

Out of your *active* PMs, assign the VM to the PM whose resulting profile has the highest PageRank.

Activate more PMs if current set of PMs gets too full

# PageRankVM

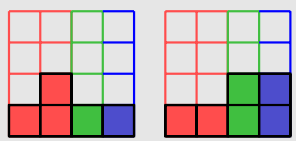



**At assignment time**

Out of your *active* PMs, assign the VM to the PM whose resulting profile has the highest PageRank.

Activate more PMs if current set of PMs gets too full

**Example**



← PMs

← VM to place

# Experiments

**Simulation**

- PM Usage
- VM Migrations
- Energy Consumption
- SLO Violations

**Real Testbed**

- PM Usage
- VM Migrations
- SLO Violations

# Experiment

## Simulation

Simulation performed on CloudSim using VM and PM types below.

TABLE I
DESCRIPTION OF VM TYPES

| VM Types | Virtual cores | | Memory (GiB) | Virtual Disk | |
|---|---|---|---|---|---|
| | # | Speed (GHz) | | # | Size (GB) |
| m3.medium | 1 | 0.6 | 3.75 | 1 | 4 |
| m3.large | 2 | 0.6 | 7.5 | 1 | 32 |
| m3.xlarge | 4 | 0.6 | 15 | 2 | 40 |
| m3.2xlarge | 8 | 0.6 | 30 | 2 | 80 |
| c3.large | 2 | 0.7 | 3.75 | 2 | 16 |
| c3.xlarge | 4 | 0.7 | 7.5 | 2 | 40 |

TABLE II
DESCRIPTION OF PM TYPES

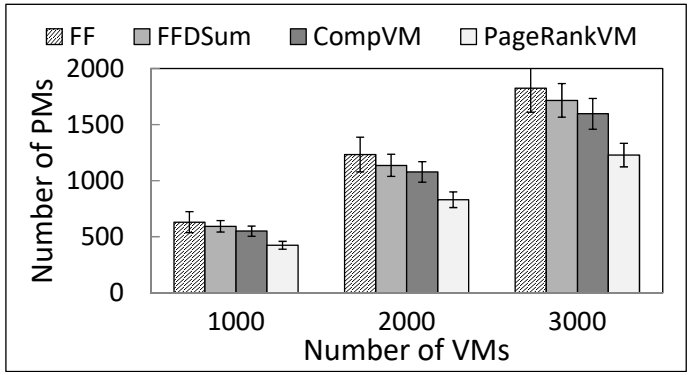
| PM Types | Physical Cores | | Memory (GiB) | Physical Disk | |
|---|---|---|---|---|---|
| | # | Speed (GHz) | | # | Size (GB) |
| M3 | 8 | 2.6 | 64 | 4 | 250 |
| C3 | 8 | 2.8 | 7.5 | 4 | 250 |

# Experiment

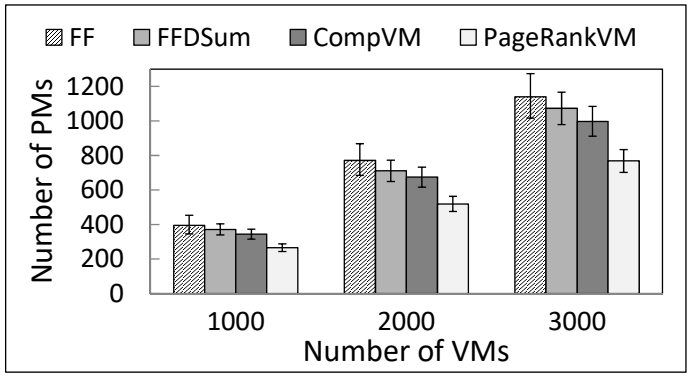
## CloudSim

3000 VMs were allocated in batches of 1000. CPU utilization of VMs were given by two public traces. Results are averaged over 100 trials.

PM Usage

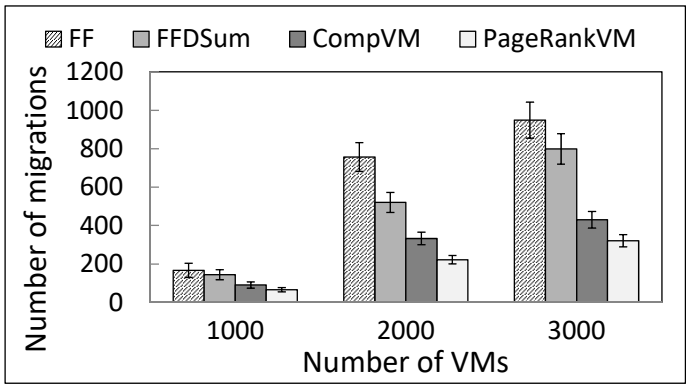



**(a)** PlanetLab

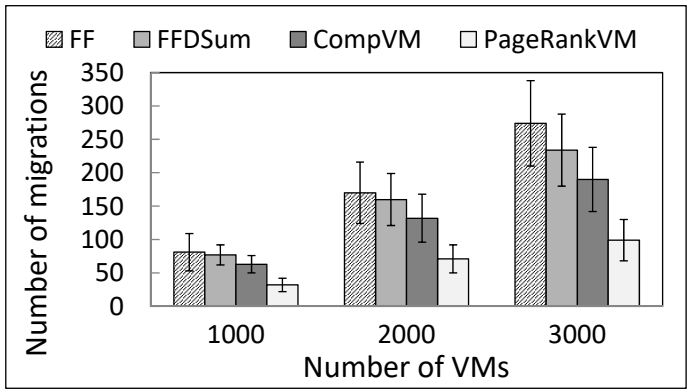



**(b)** Google Cluster

VM Migrations




**(a)** PlanetLab




**(b)** Google Cluster

# Summary

PageRankVM is a heuristic VM placement algorithm that:

- Reduces the total number of PMs needed to host VM demands
- Satisfies anti-collocation constraints by considering VM permutations
- Has very low placement-time cost

# Questions

?