# Reducing Load Imbalance of Virtual Clusters via Reconfiguration and Adaptive Job Scheduling

Sina M. Khorandi, Siavash Ghiasvand, Mohsen Sharifi

TAPEMS, MADRID, SPAIN
MAY 14TH - 17TH , 2017

# Outline

Introduction

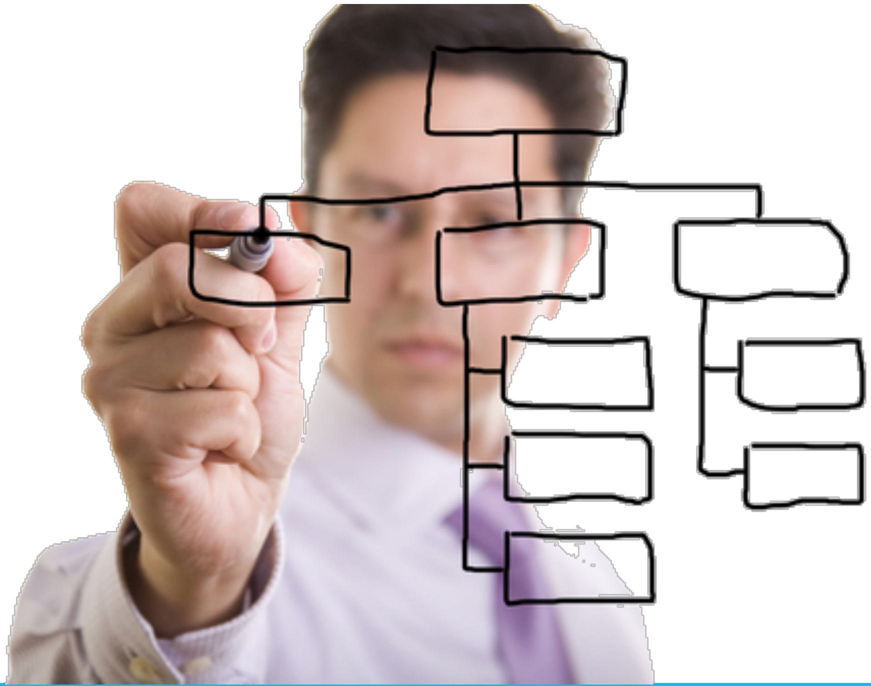
Challenges

Proposed solution

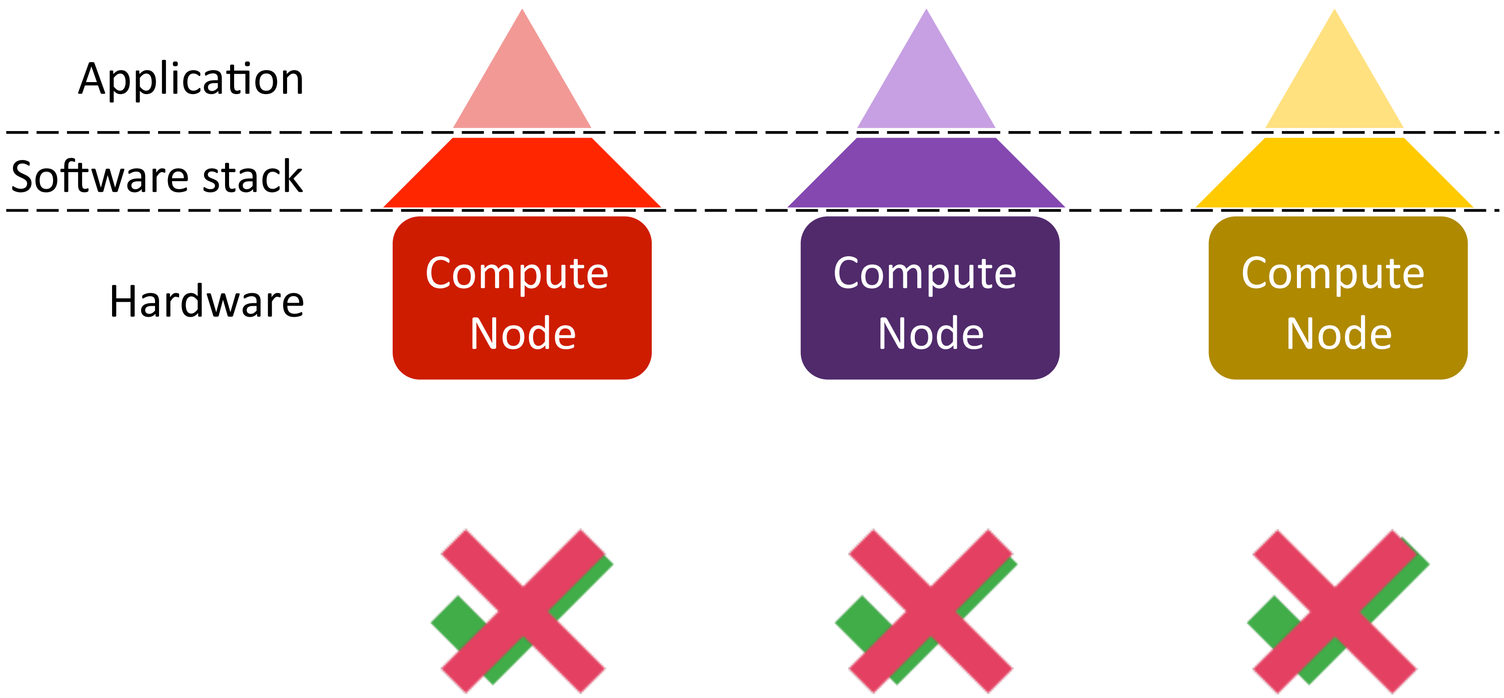
Formal proof

Experimental results

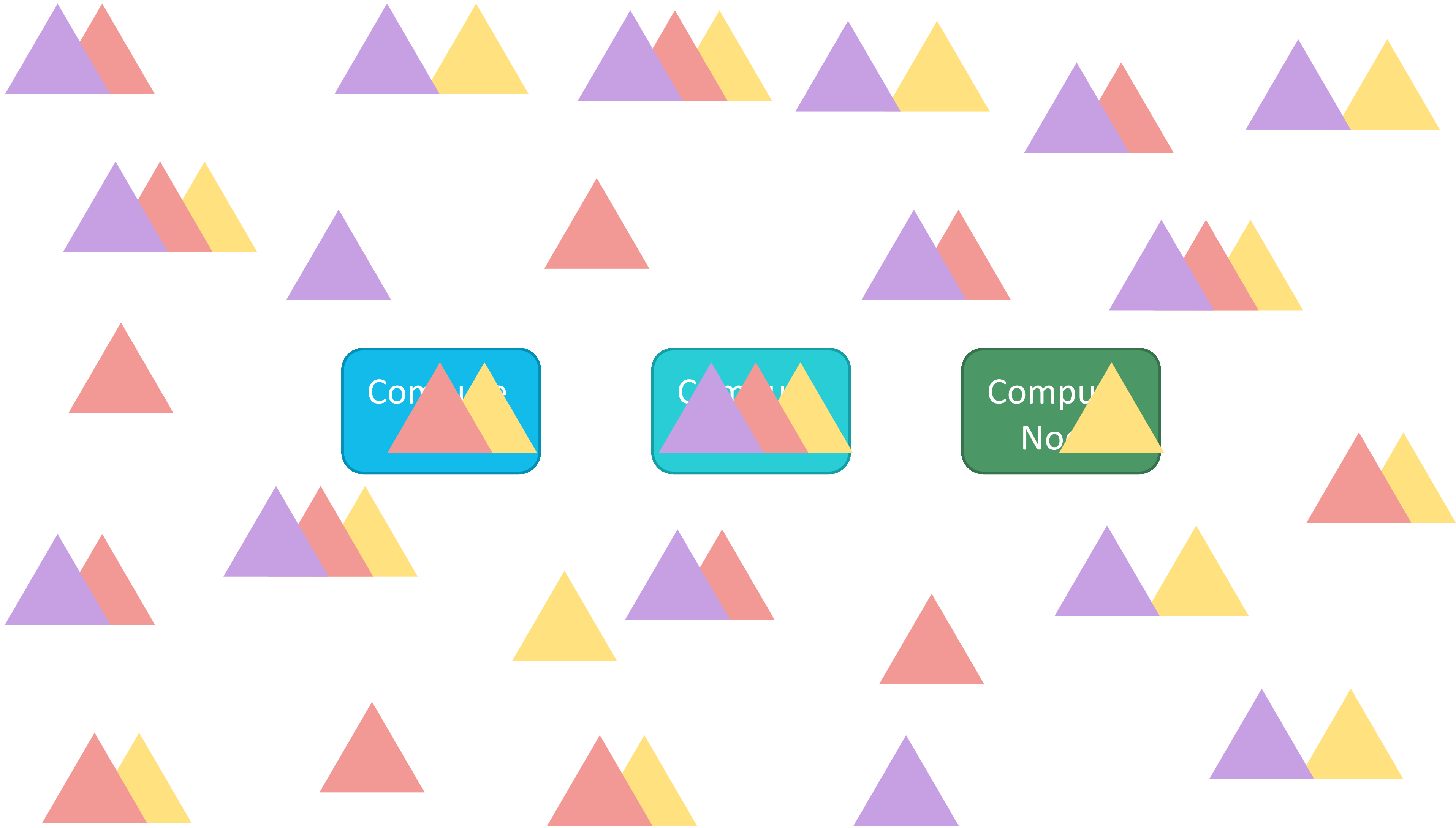
Conclusion



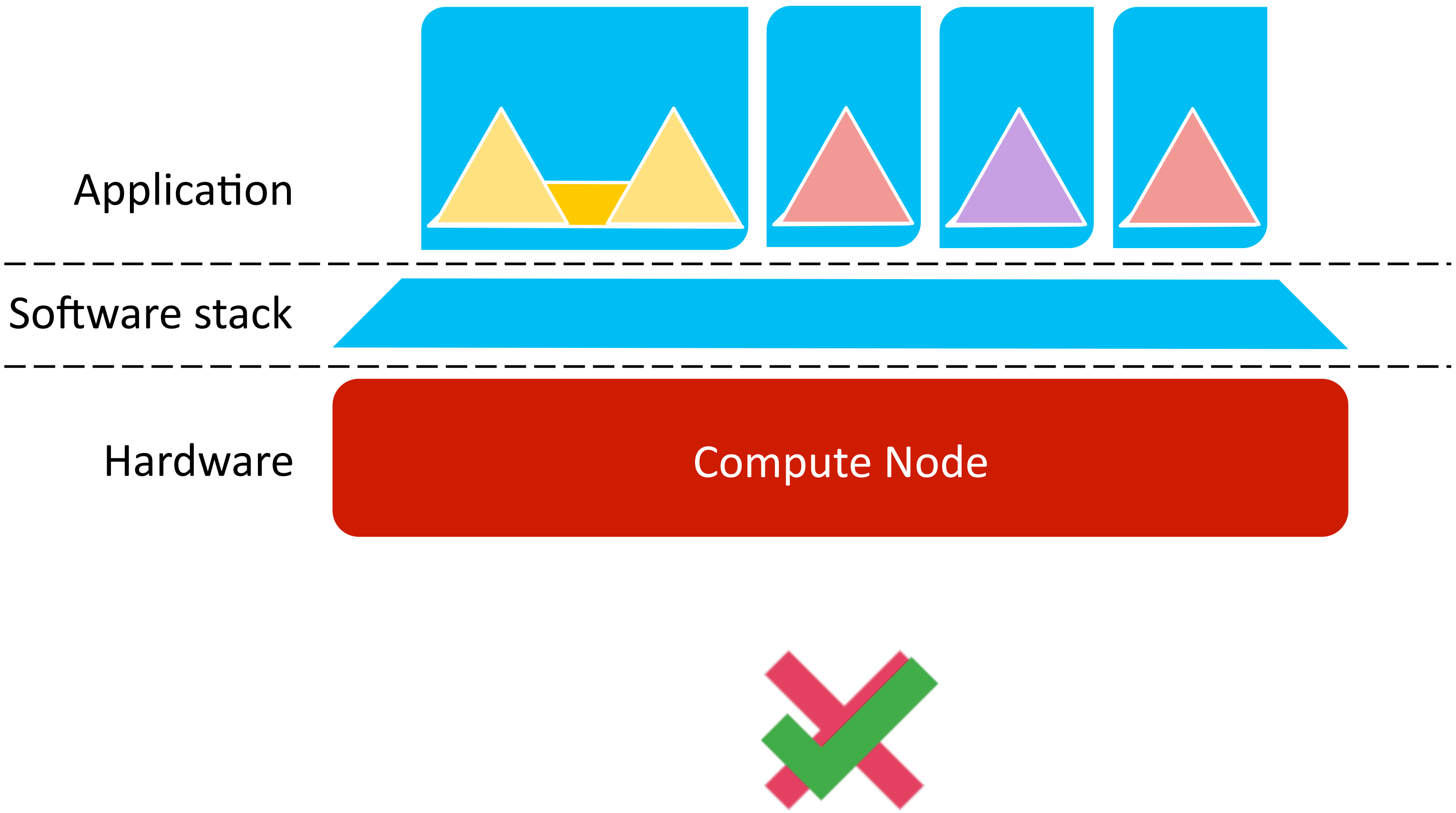
# Specific requirements



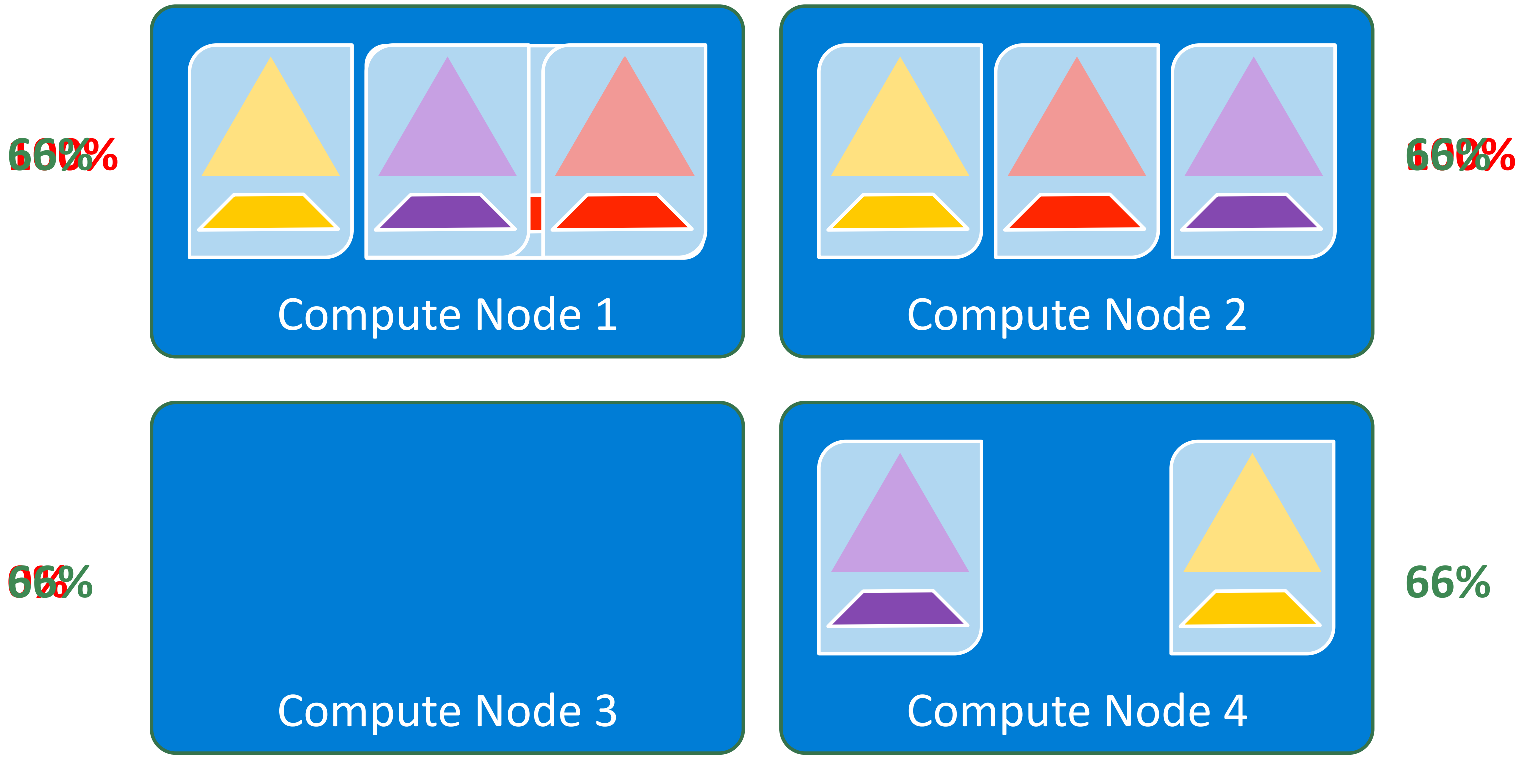
# Toward extreme scales



# Virtualization: specific requirements



# Virtualization: extreme scale




## 5% > Overhead

J. Lange et al., "Minimal-Overhead Virtualization of a Large Scale Supercomputer", ACM SIGPLAN Notices – VEE '11, vol. 46, no. 7, 2011

~~...with **Live** migration and dedicated resources!~~

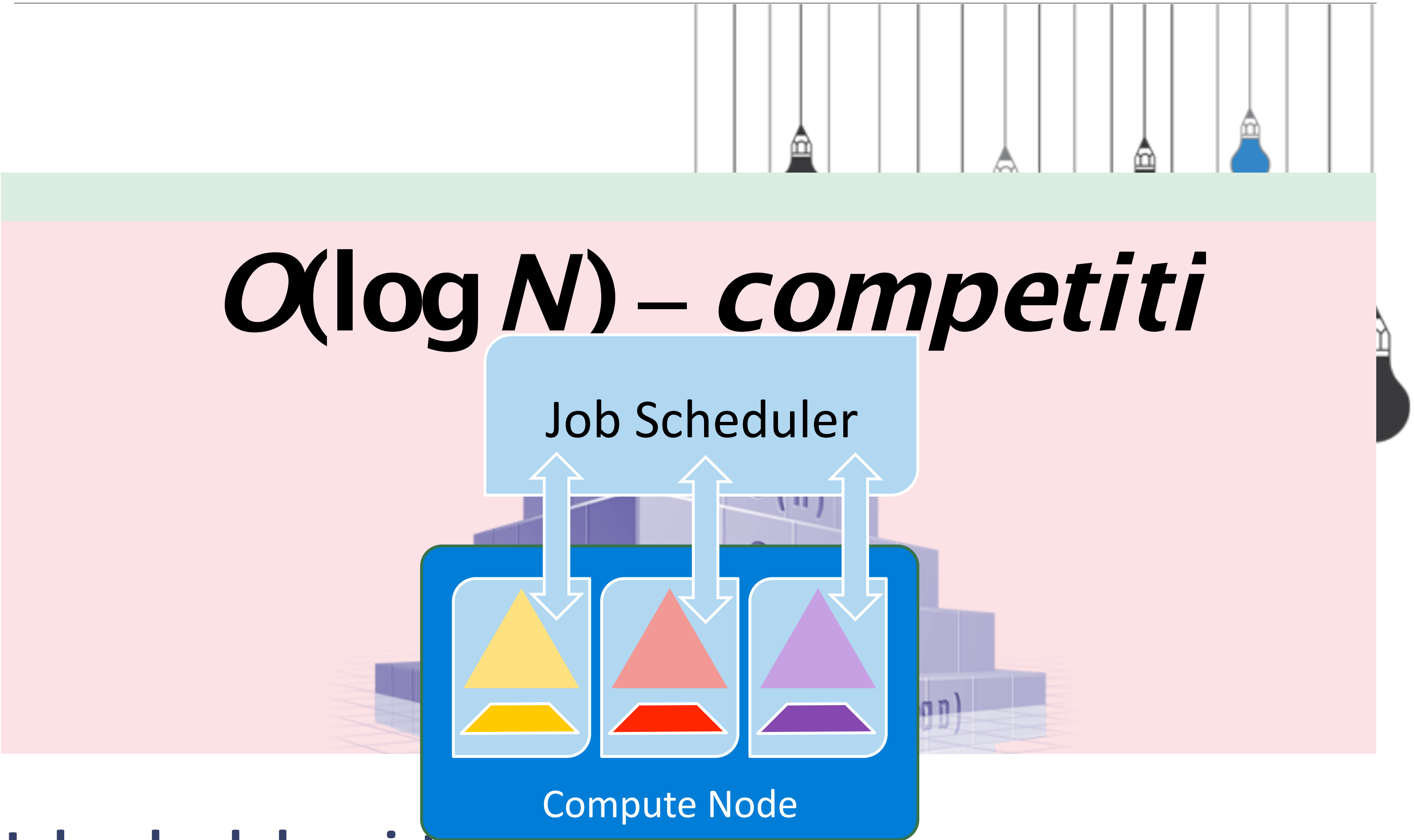
# Problem Definition


**Optimal load imbalance**

**Near-to-optimal scheduling**

**Job scheduler virtualization unawareness**

# Problem Definition




**Job scheduler virtualization unawareness**

# Approach and Goal

## Virtual circuits routing (VCR) theory

-- B. Awerbuch, et al., “Competetive Routing of Virtual Circuits with Unknown Duration”, Journal of Computer and System Sciences, 2001

## Opportunity cost algorithm

-- S.M. Khorandi, et al., “Scheduling of Online Compute-intensive Synchronized Jobs on High Performance Virtual Clusters”, Journal of Computer and Systsem Sciences, 2016

-- A. Keren, et al., “Opportunity Cost Algorithms for Reduction of I/O and Interprosess Communication Overhead in a Computer Cluster”, IEEE Transactions on Parallel and Distributed Systems, 2003

## A Scheduling technique

|___ Online manner
|___ Optimal load imbalance
|___ Low time complexity

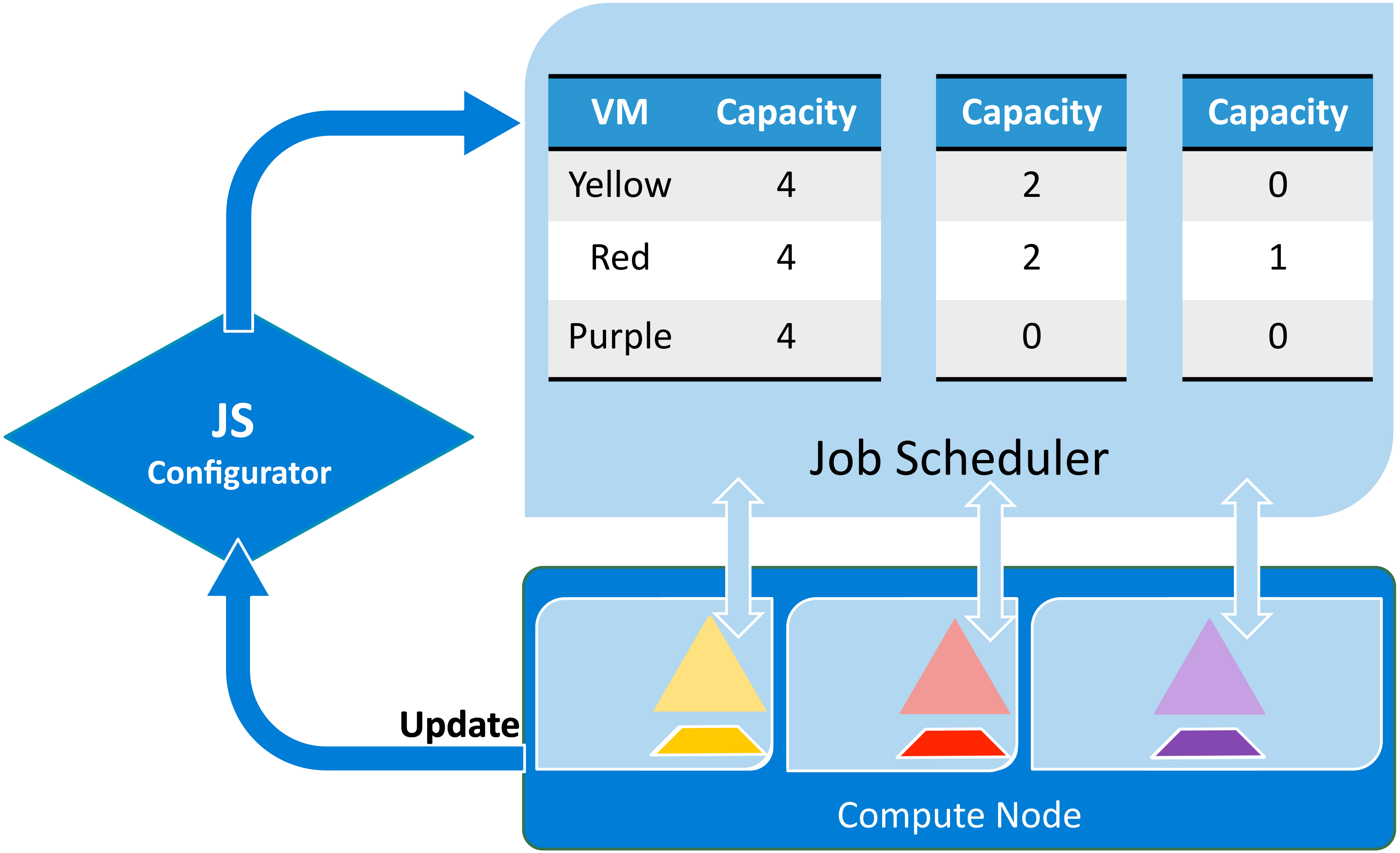
# Proposed solution

1. Statically assigned VMs

2. Check the stability condition of *ASSIGN-ROUTE* algorithm, with the physical core of physical machines as the focused resource

3. When the stability condition is violated, reduce the CPU load capacity of each VM of the overloaded physical core

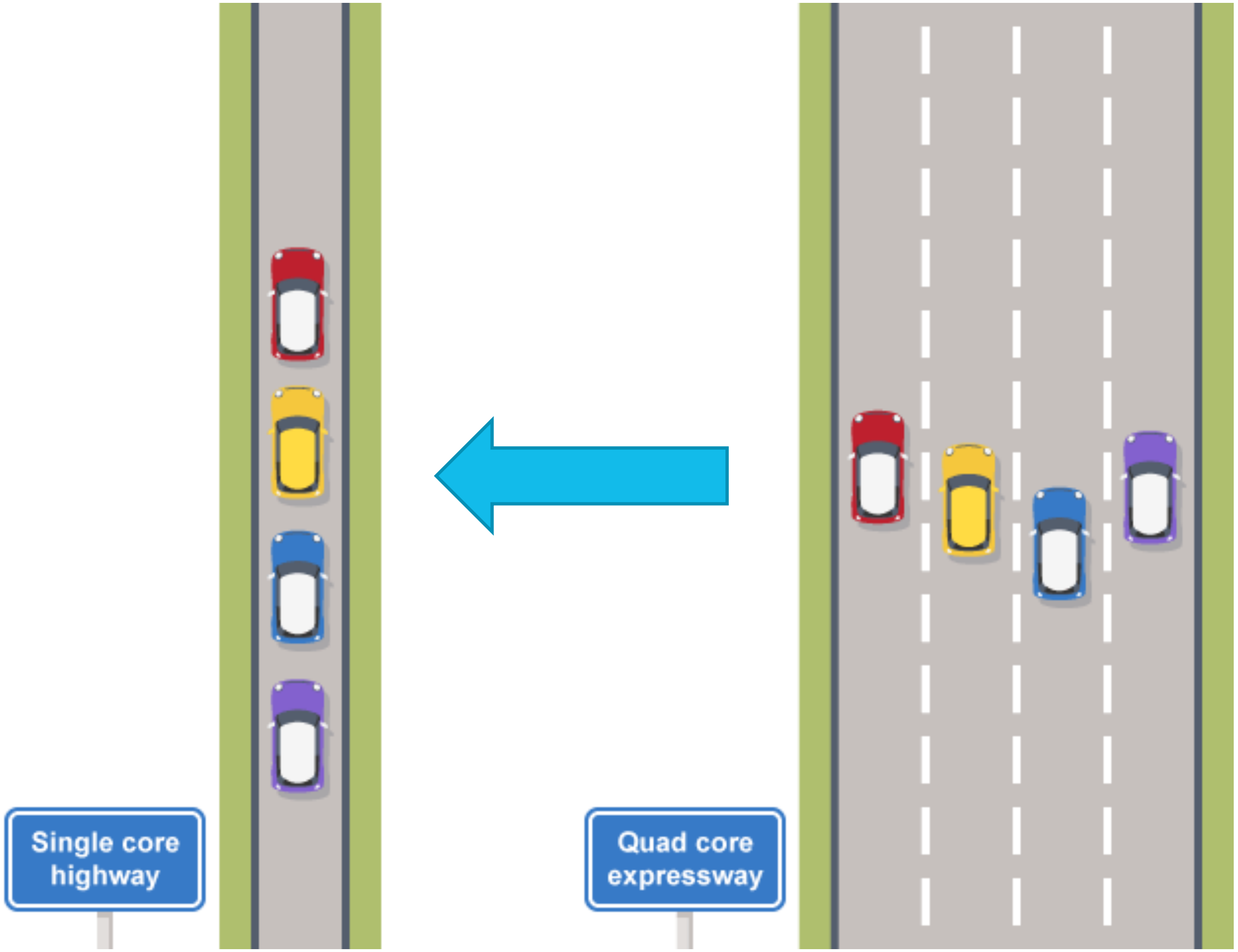
# Proposed solution




| VM | Capacity | Capacity | Capacity |
|---|---|---|---|
| Yellow | 4 | 2 | 0 |
| Red | 4 | 2 | 1 |
| Purple | 4 | 0 | 0 |

# Assumption

Each VM has only a single-core CPU



# Parameters

| | |
|---|---|
| $pc_i$ | The i[th] physical core |
| $vm_j$ | The j[th] virtual machine |
| $s$ | The processor's speed of a VM or physical core |
| $l_{vm_j}(t)$ | The load of $vm_j$ at time t |
| $l_{pc_i}(t)$ | The load of $pc_i$ at time t |
| $c_{vm_j}$ $c_{pc_i}$ | VM or physical core capacity, respectively |
| $u_j(t)$ | CPU cycles required by job *j* at time *t* |
| $u_{vm}(t)$ | CPU cycles required by job *vm* at time *t* |

# ASSIGN-ROUTE Algorithm


| | |
|---|---|
| | $pc_i$ |
| | $vm_j$ |
| | $s$ |
| | $l_{vm_j}(t)$ |
| | $l_{pc_i}(t)$ |
| | $c_{vm_j}\ c_{pc_i}$ |
| | $u_j(t)$ |
| | $u_{vm}(t)$ |

Marginal cost:

$$H_P(j) = \sum_{e \in P} a^{l_e(j)+p_e(j)} - a^{l_e(j)}$$

Stability condition:

$$\forall P', j: \sum_{e \in P} a^{l_e(j)+p_e(j)} - a^{l_e(j)} \leq 2 \sum_{e \in P'} a^{l_e(j)+p_e(j)} - a^{l_e(j)}$$

# Lemma 1

$pc_i$

$vm_j$

$s$

$l_{vm_j}(t)$

$l_{pc_i}(t)$

$c_{vm_j}$ $c_{pc_i}$

$u_j(t)$

$u_{vm}(t)$

Jobs scheduler is always reconfigurable, to preemptively reassign jobs.

$$k > \max_{\forall j \mapsto vm} \left( \frac{l_{vm}(j)}{l_{vm}(j) + p_{vm}(j)} \right) \times \log_a^{(\gamma \times N)}$$

# Lemma 2


$pc_i$

$vm_j$

$s$

$l_{vm_j}(t)$

$l_{pc_i}(t)$

$c_{vm_j}$ $c_{pc_i}$

$u_j(t)$

$u_{vm}(t)$

Reconfiguration of all VMs assigned to instable physical core prevents assignment/reassignment of jobs to those VMs.

$$\tau_{pc} = \max_{\forall vm \mapsto pc}\left(\tau_{vm}\right) \qquad \tau_{vm} = \max_{\forall j \mapsto vm}\left(\frac{l_{vm}(j)}{l_{vm}(j) + p_{vm}(j)}\right)$$

$$k > \tau_{pc} \times \log_a^{\gamma} N$$

# Lemma 3

The combination of **job scheduler** and **configurator** is always stable.

# Theorem

The combination of job scheduler and scheduler configurator is stable and:

$$\log(\beta) + O(\log(M)) - competitive \qquad \beta = \frac{N}{M}$$

# Experimental results

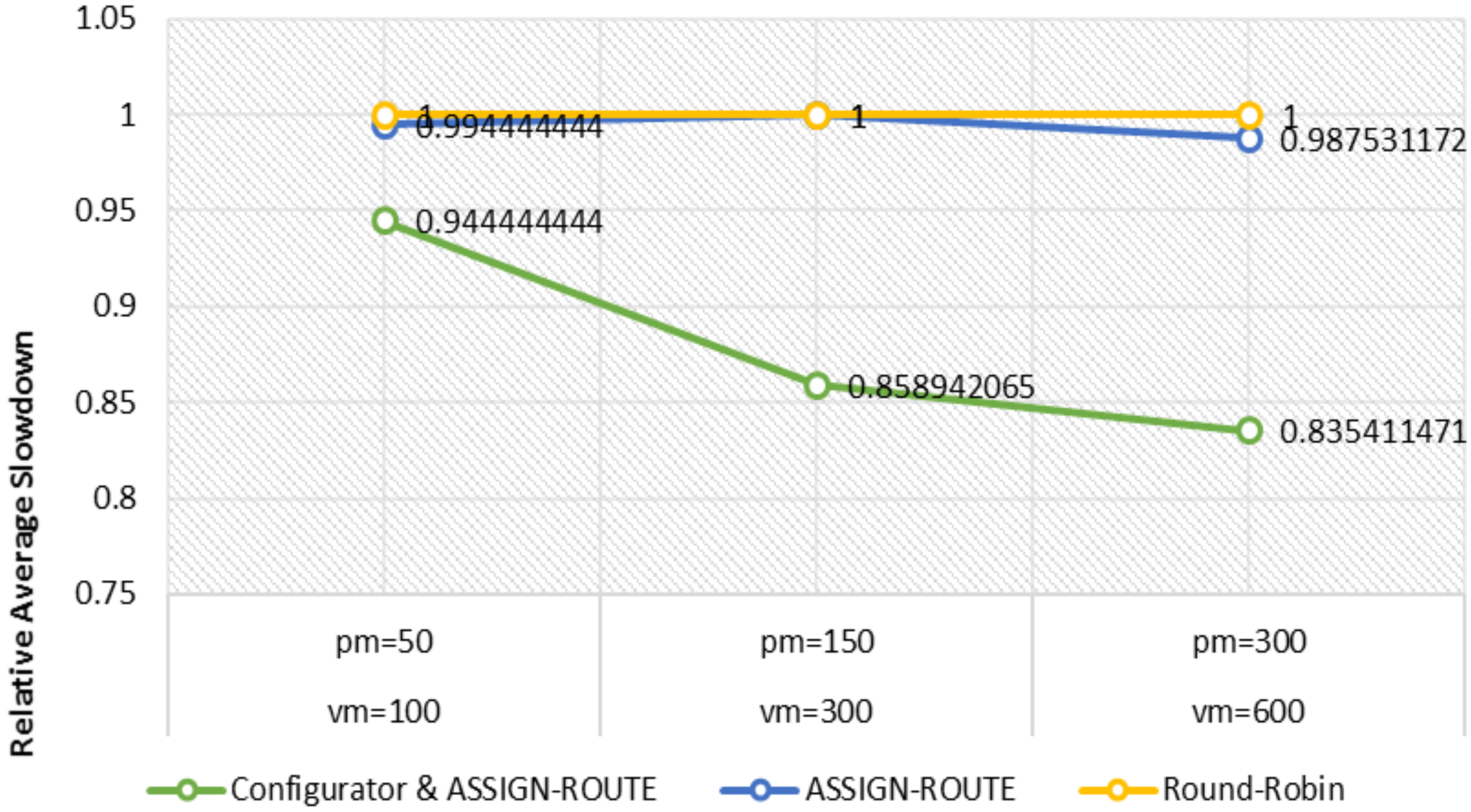
## Synthetic Jobs

- 2/r seconds of CPU time on the fastest CPU of a physical machine
  - 5% of the jobs required 20/r seconds
  - r is a random uniform number between 0 and 1
- Total number of jobs was 50N

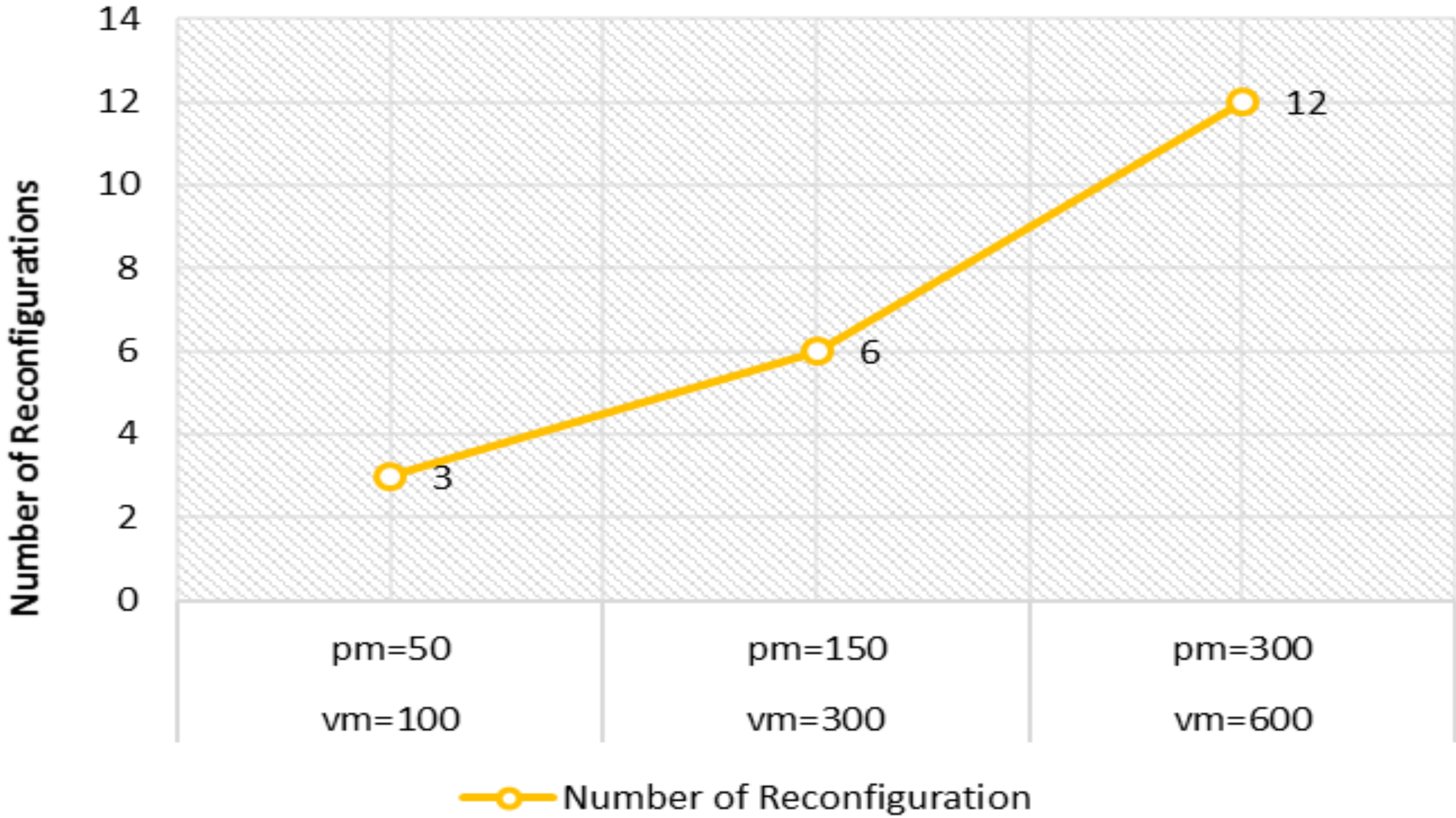
## 50 physical machine

- A single core 4 GHz CPU
- 16 GB of RAM
- Each physical machine contained 2 VMs
  - 1 GB of RAM
  - Full capacity CPU

# Relative slowdown

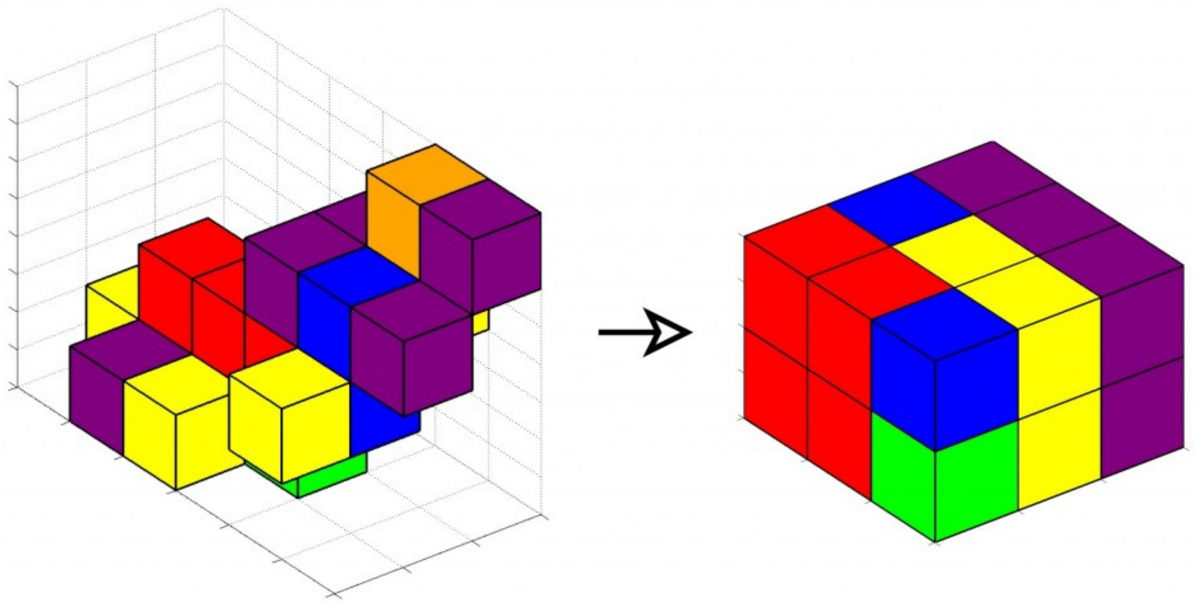


# Reconfigurations



# Conclusion

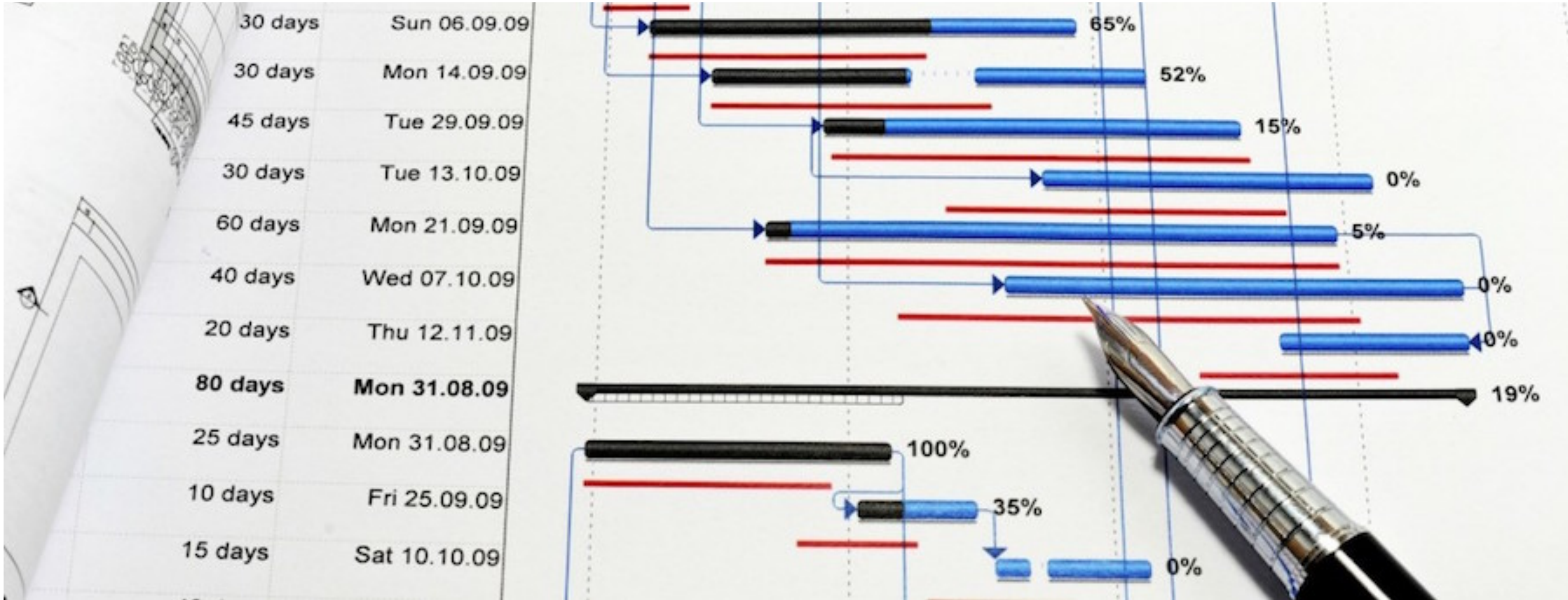
Use of Reconfiguration instead of VM migration



# Conclusion

Use of Reconfiguration instead of VM migration

A new combinatory scheduling technique



# Conclusion

Use of Reconfiguration instead of VM migration

A new combinatory scheduling technique

Formal proof of Log(n)-competitiveness

$$\forall j \in vm, vm' :$$
$$a^{l_{vm}(j)+p_{vm}(j)} - a^{l_{vm}(j)} \le 2(a^{l_{vm'}(j)+p_{vm'}(j)} - a^{l_{vm'}(j)}) \quad (6)$$

Using the proof of Lemma 2.2 in [16], and according to the fact that $\forall x \in [0,1] : 2(a^x - 1) \le \gamma x$, and since the maximum load is *O(log N)*, we get Eq. (7) for all jobs on *vm*.

$$a^{k\times(l_{vm}(j)+p_{vm}(j))} - a^{k\times l_{vm}(j)} \le \gamma N \quad (7)$$

In Eq. (7), $\gamma$ is a real value between 0 and 1, and $a=1+\gamma/2$. Using logarithm on both side of the inequality of Eq. (7), and contradictory of inequality for implying violation of stability condition, we get $\frac{l_{vm}(j)+p_{vm}(j)}{l_{vm}(j)} > \log_{a^k}^{\gamma N}$. By changing the base of the logarithm we get $\frac{l_{vm}(j)+p_{vm}(j)}{l_{vm}(j)} > \frac{\log_a^{\gamma N}}{k}$. So, Eq. (8) shows the condition for *k* on *vm* for each job *j*.

$$k > \max_{\forall j \mapsto vm}\left(\frac{l_{vm}(j)}{l_{vm}(j)+p_{vm}(j)}\right)\times\log_a^{\gamma N} \quad (8)$$

# Future Works

## Synchronized Jobs

- I/O style synchronization
  - Global machine-level barriers
- IPC-based synchronization
  - Iterative parallel synchronized processes

# Thank you!


SINA_MAHMOODI

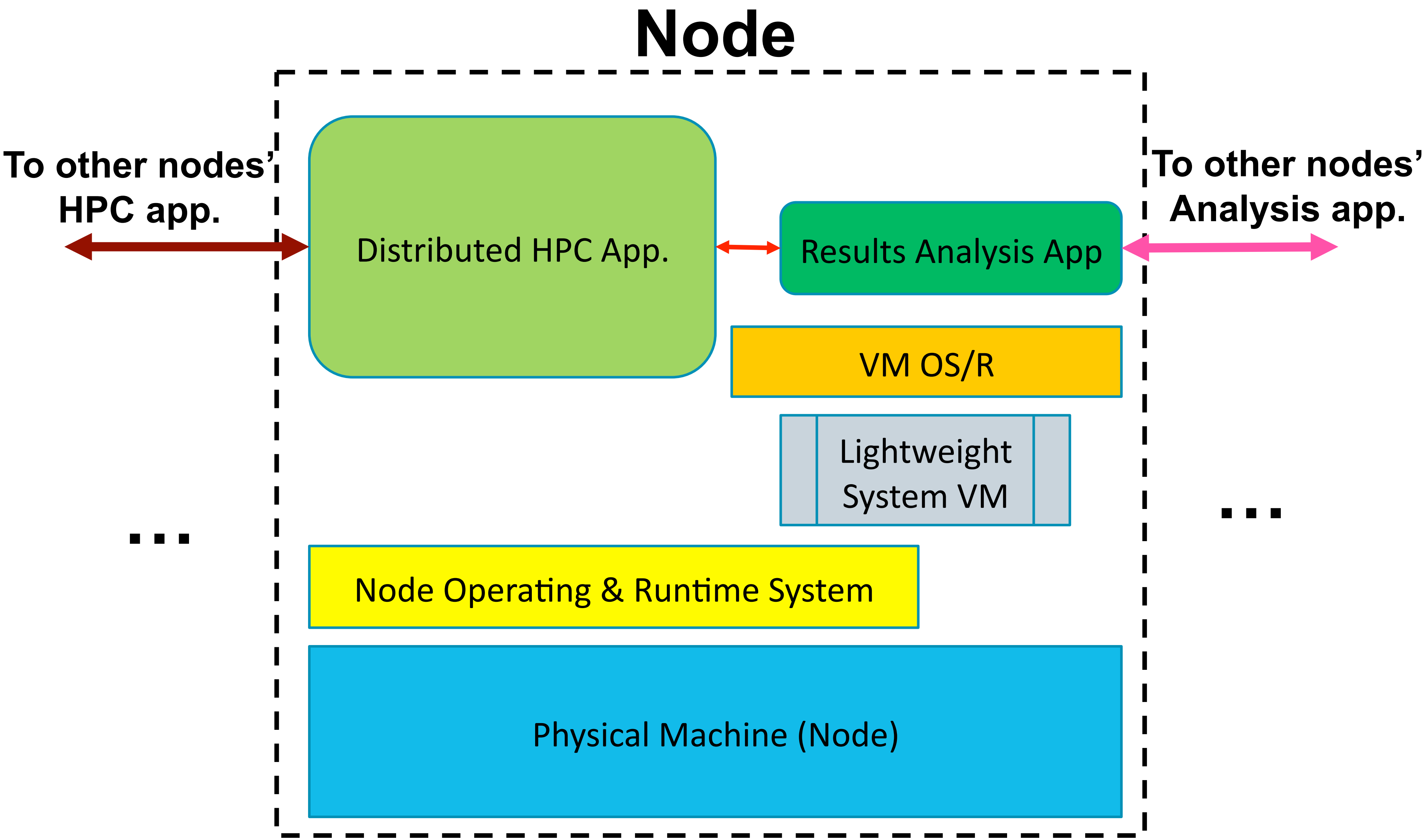
MSHARIFI

SIAVASH.GHIASVAND

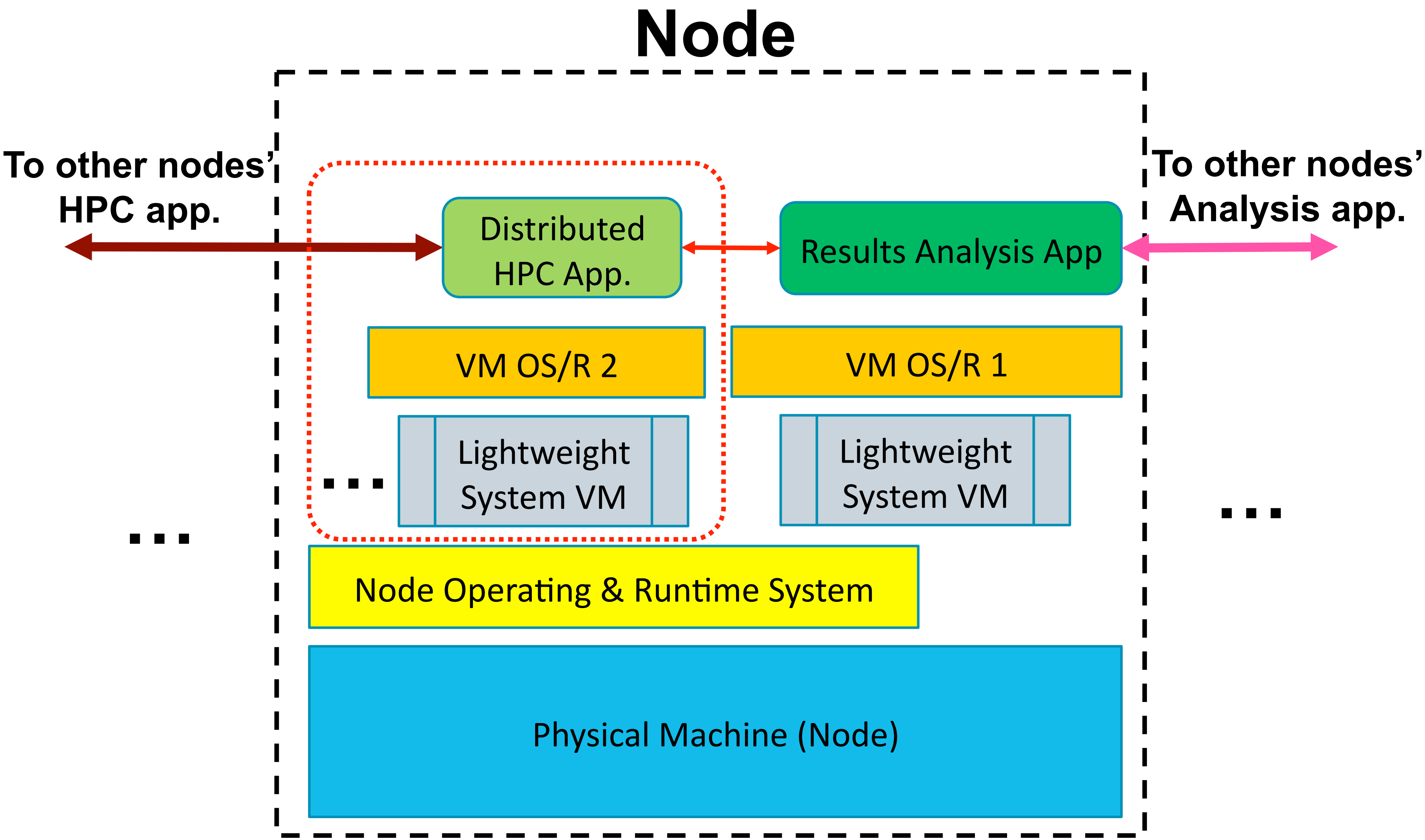
@COMP.IUST.AC.IR

@TU-DRESDEN.DE

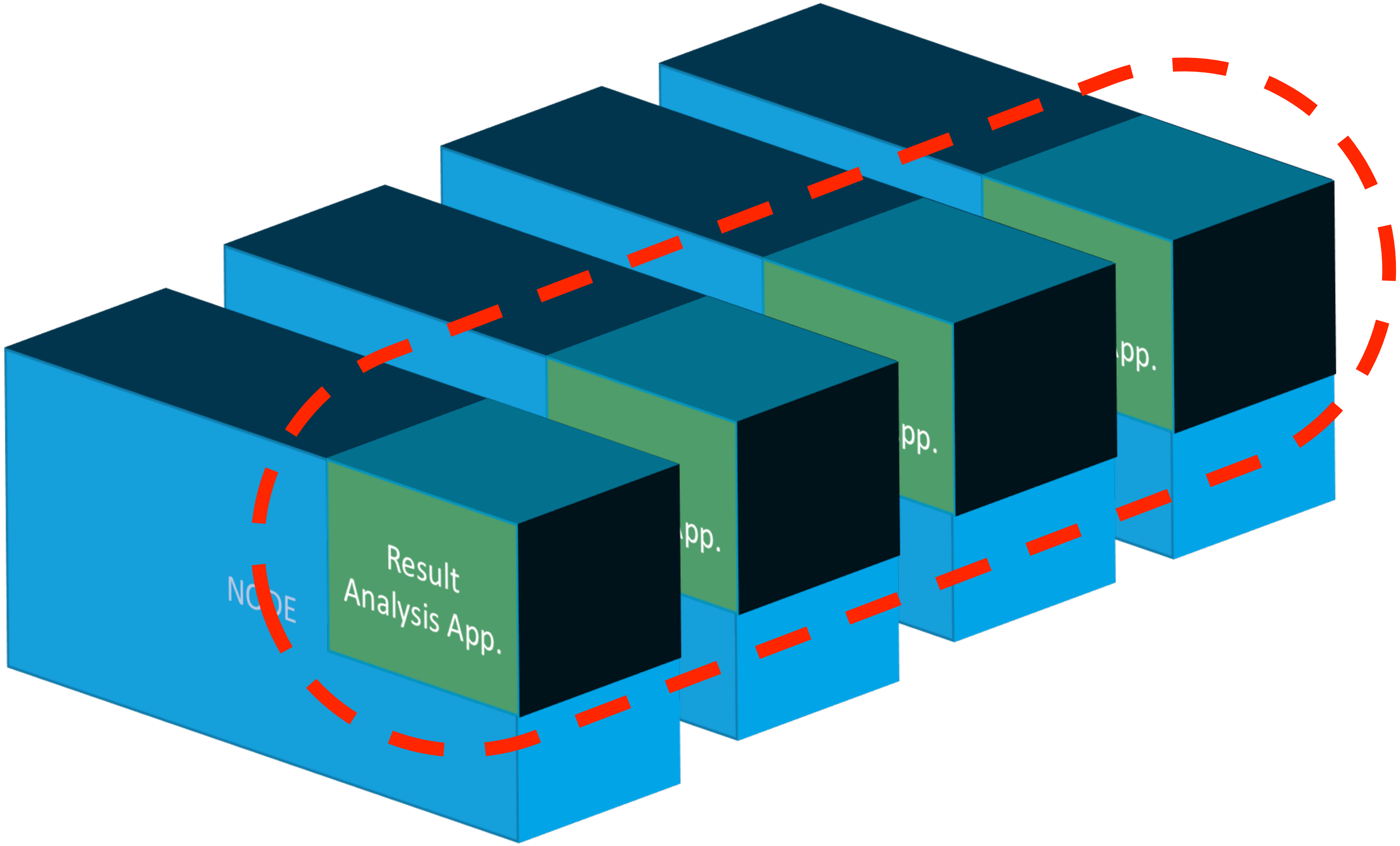
# Virtualization



# Virtualization



# Motivations



# Goals

Solving theoretical problem of *virtualization-unaware*, *adaptive*, and *near-to-optimal*, scheduling of online jobs on virtual clusters

Includes only loosely-coupled and sequential jobs

→ which only require CPU intensively

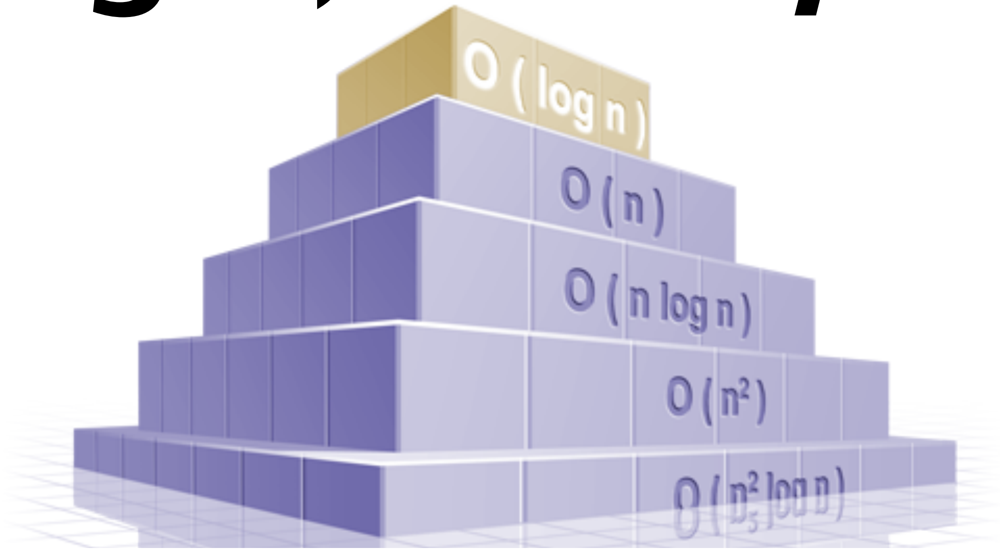
# Competitive Analysis

Algorithm *ALG* is *c-competitive* if for any sequence of input *I* we have:

$$ALG(I) \leq c \times OPT(I) + \alpha$$

# Low time complexity

$O(\log N)$ – *competiti*



# Background

## Situation

Online and adaptive scheduling is **NP-hard**

## Our goal

Minimizing the makespan

## Challenge

VMs and PCs are unrelated (not dependent to load and CPU speed)

No prior information about jobs CPU requirements

Opportunity cost approach

# Related Works

*Meta-Scheduling:* determining when and where to invoke the scheduler

- H. Menon, et al., "Automated Load Balancing Invocation Based on Application Characteristics," Proc. IEEE International Conference on Cluster Computing, (CLUSTER), IEEE, 2012, pp. 373 - 381.
- E. Huedo, et al., "Grid Architecture from a Metascheduling Perspective," Computer, vol. 43, no. 7, 2010, pp. 51 – 56.
- M. Beltran and A. Guzman, "How to Balance the Load on Heterogeneous Clusters," International Journal of High Performance Computing Applications, vol. 23, no. 1, 2009, pp. 99-118.

*Auto-Tuning:* selecting the scheduling policy, automatically

- A. Streit, "A Self-Tuning Job Scheduler Family with Dynamic Policy Switching," Proc. 8th International Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP '02), 2002, pp. 1 – 30.
- C. Clauss, et al., "Dynamic Process Management with Allocation-Internal Co-Scheduling Towards Interactive Supercomputing," Proc. the 1st Workshop on Co-Scheduling of HPC Applications (COSH 2016), 2016.

# 3- Job and Machine Model 2

Identical physical CPU and cores

VM Interference

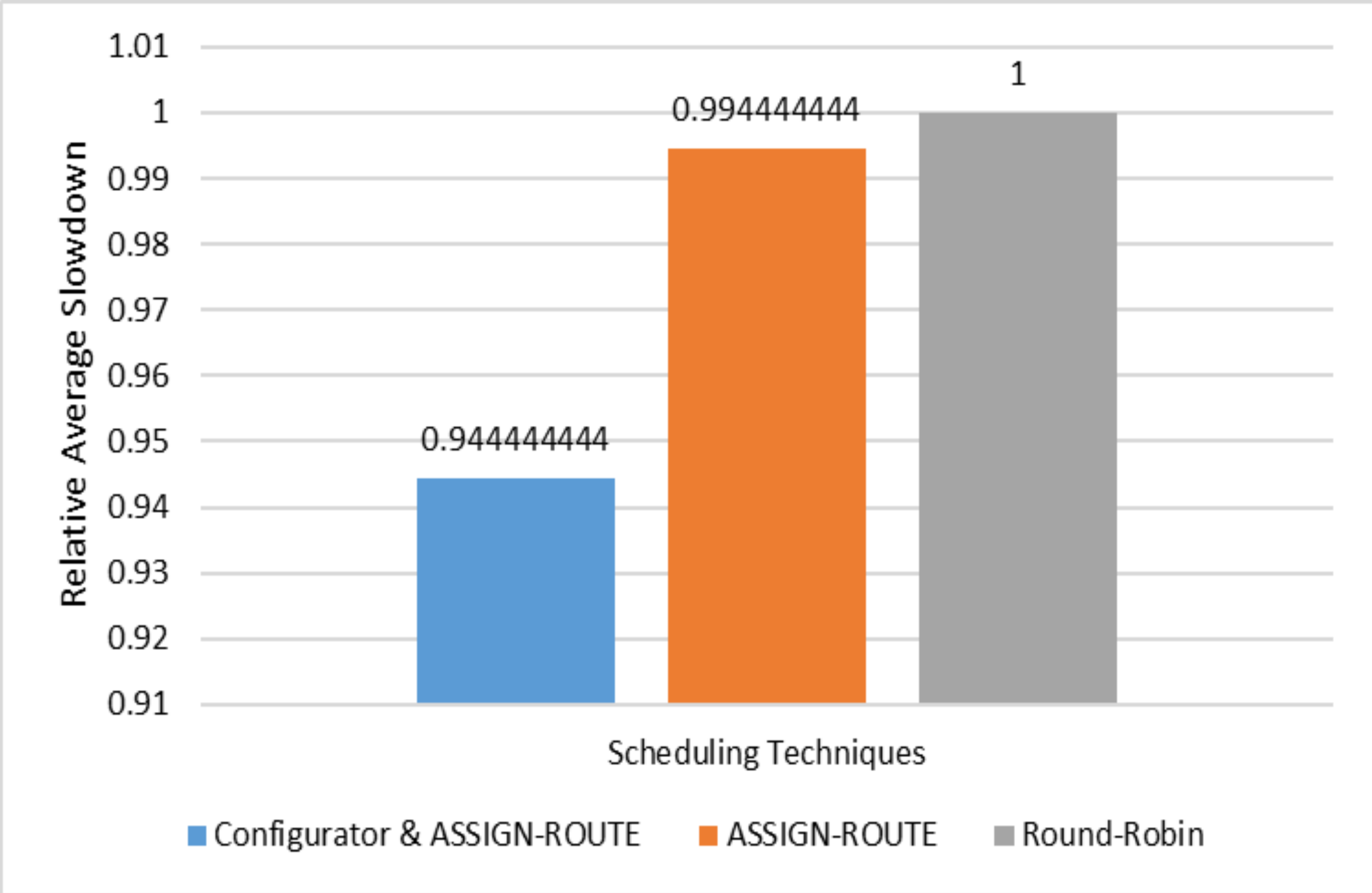
- Unrelated Machines

$$p_j^{vm}(t) = u_j(t) / c_{vm} \quad 0 \le p_j^{vm}(t) \le 1$$

$$p_{vm}^{pc}(t) = u_{vm}(t) / c_{pc} \quad 0 \le p_{vm}^{pc}(t) \le 1$$

$$l_{pc}(t) = \sum_{\forall vm \mapsto pc} l_{vm}(t) + I(\forall vm \mapsto pc, t)$$

# Results 1



1.01
1
0.99
0.98
0.97
0.96
0.95
0.94
0.93
0.92
0.91
Relative Average Slowdown
0.944444444
0.994444444
1
Scheduling Techniques
Configurator & ASSIGN-ROUTE
ASSIGN-ROUTE
Round-Robin

# Conclusion

Application of Composition
- Virtualization-based Approach

Interference Impact

Load Balancing Goal
- Virtual Circuits Routing
- Opportunity Cost Approach

Reconfiguration

Formal Proof of Competitiveness