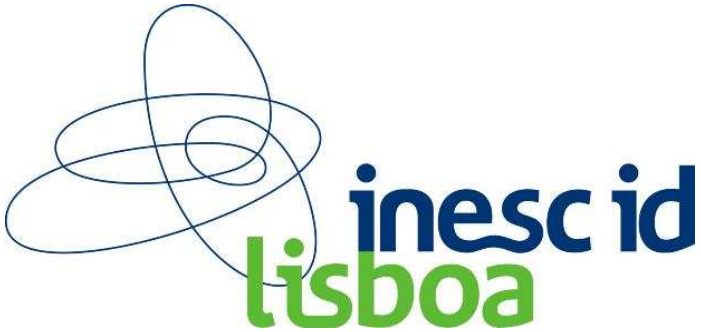
Heuristic for resources allocation on utility computing infrastructures

technology
from seed

João Nuno Silva, Luís Veiga, Paulo Ferreira
Distributed Systems Group – INESC-ID
Lisbon, Portugal



INSTITUTO SUPERIOR TÉCNICO

Motivation

- Lots of computation demanding tasks
  - Parameter sweep simulations, …
  - Image rendering, …
  - Everywhere …
- Most of them are Bag of Tasks (BoT)
  - Algorithmically complex
  - Simple w.r.t. task coordination
  - Easily parallelizable

Bag-of-Tasks Problems (BoT)

- Where to run them?
  - Single computer (bad performance)
  - Local Cluster (ownership / operation cost)
  - Grid (requires institutional membership)
  - Utility computer infrastructure
    - Mostly efficient
    - Cheap
    - Unrestricted public access

Utility computing (UC)

- Provider owns clusters of computers
  - Public use
- Users launch virtual machines
  - On demand
  - No reservation mechanisms
- No complex contracts
  - Like utilities, user pays what he spends

BoT on Utility computing Amazon EC2

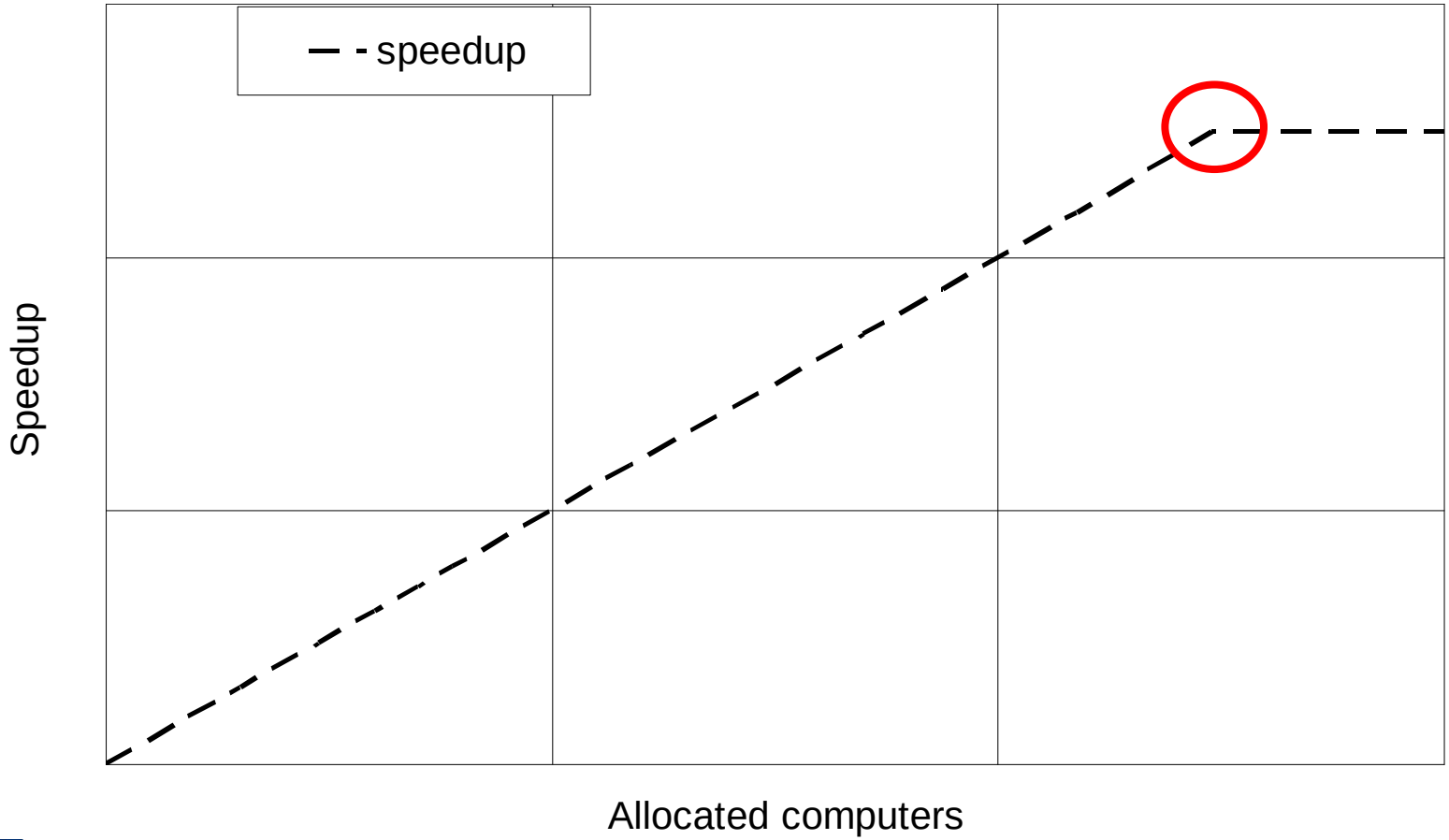
- User picks a task distribution middleware
  - Install it on virtual appliance
- Configures data download mechanisms
- Launches virtual machines
  - Deployed on the utility computing infrastructure
- Waits for completion
  - Pay only for the execution time

BoT on Utility computing Amazon EC2

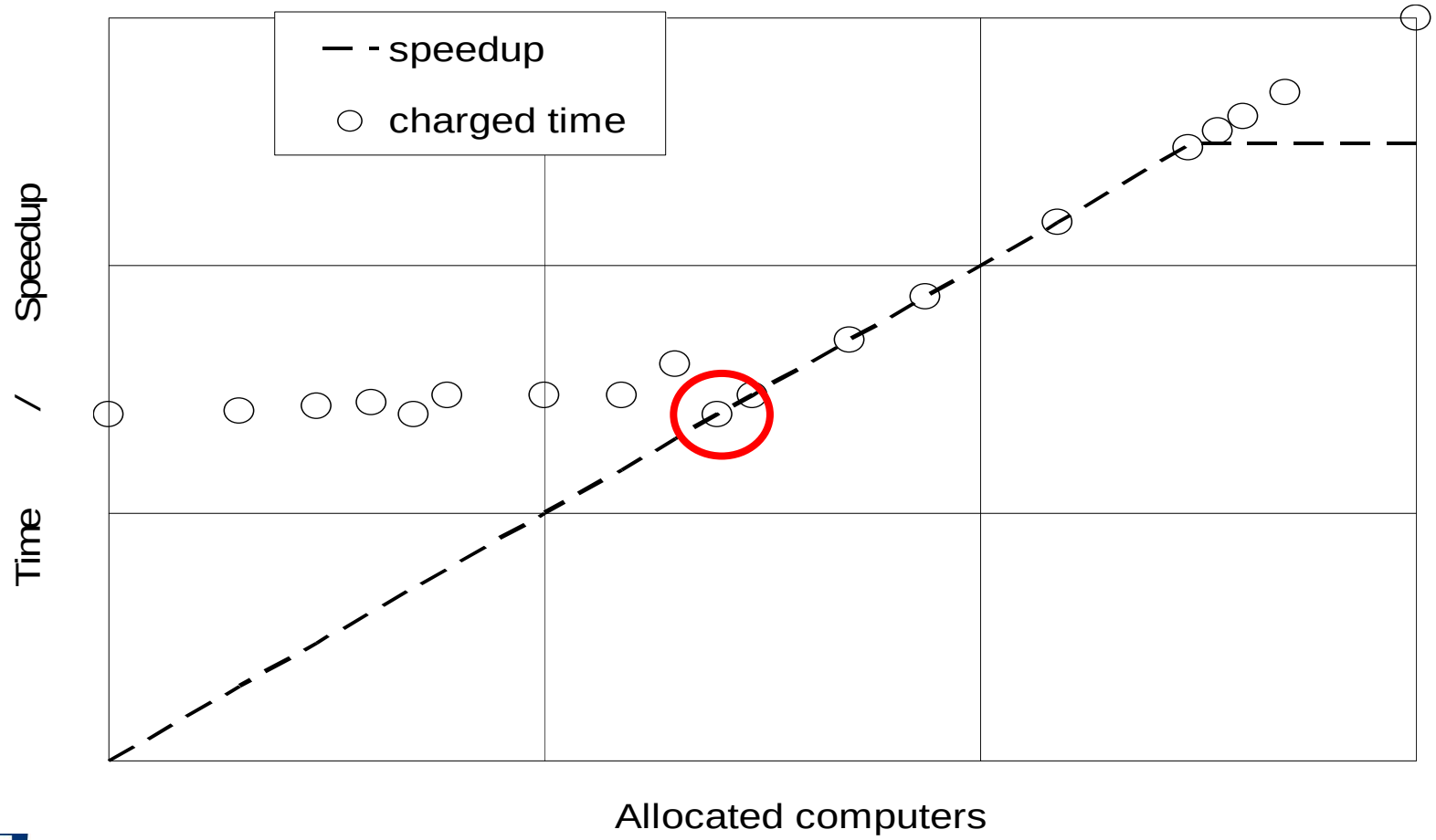
- 32 Tasks / 15m each / 8h total
  - User charged by the hour

# VM's	Cost (charge units)	Wall Time	Speedup
1	1*8h = 8	8h	1
4	4*2h = 8	2h	4
8	**8*1h = 8**	**1h**	**8**
12	12x1h =12	45m	10.6
16	16*1h =16	30m	16
32	**32*1h =32**	**15m**	**32**
64	64*1h =64	15m	32

How many computers?



How many computers?



Objective

- Development of heuristic for
  - VM allocation on UC infrastructures
    - tailored to BoT problems
  - Reducing waste of charged VM time
  - Guaranteeing best speedups
  - For a fixed payment decided by the user

Resource managment on Utility Computing

- System centralized backend
  - Aimed at global efficiency
    - Physical CPU allocation, node idleness, network traffic, heat dissipation
- User frontend scheduling
  - Simple API to handle
    - VM allocation, accounting, monitoring
  - No optimization mechanisms

Shortcomings of Current Utility Computing

- User frontend scheduling
  - Unable to determine appropriate/optimal
    - Number of VMs to allocate
    - Number of cores on each VM
    - Amount of memory
    - Network behavior
- Cost-efficient use of Utility Computing?
  - No framework, guidelines, principled approach

How many computers?

- Should the user decide?
  - Execution times may not be known
  - The less knowledge the user has
    - More prone to error he is
- Automatic allocation
  - Run-time calculation
  - Guarantees:
    - optimal usage of the charged time

Heuristic

- User decides how much to pay
  - *As-if-serial* processing time
    - Minimum fixed cost
  - Relative urgency overhead
    - e.g. minimum+50%
- System guarantees:
  - Best speedups for that amount

Heuristic

- How to reduce wasted time

$$\#VMs = \frac{averageTime \times ntasks}{chargedTimeUnit}$$

- But *averageTime* not known till the end

Heuristic

- Heuristic should use
  - Average time of finished tasks
  - nTasks not completed
  - Available time on Running hosts

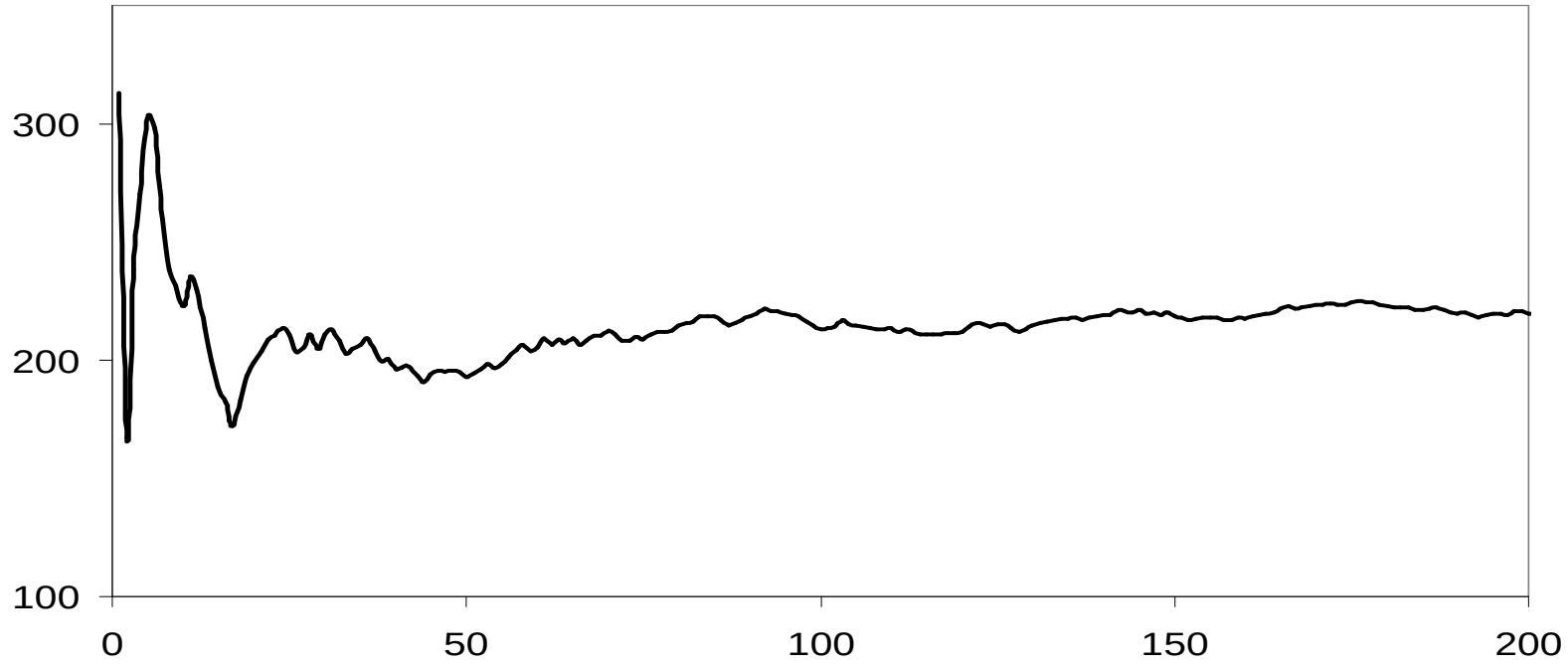
$$\#VMs = \frac{averageTime \times remainingTasks - remainingTime}{chargedTimeUnit}$$

Heuristic

- If first tasks take too long?
  - Too many VMs are allocated
- If first tasks are too fast ?
  - Too few VMs are allocated
- Approach: select tasks randomly
  - Disperses locality complexity
  - Evens-out variations

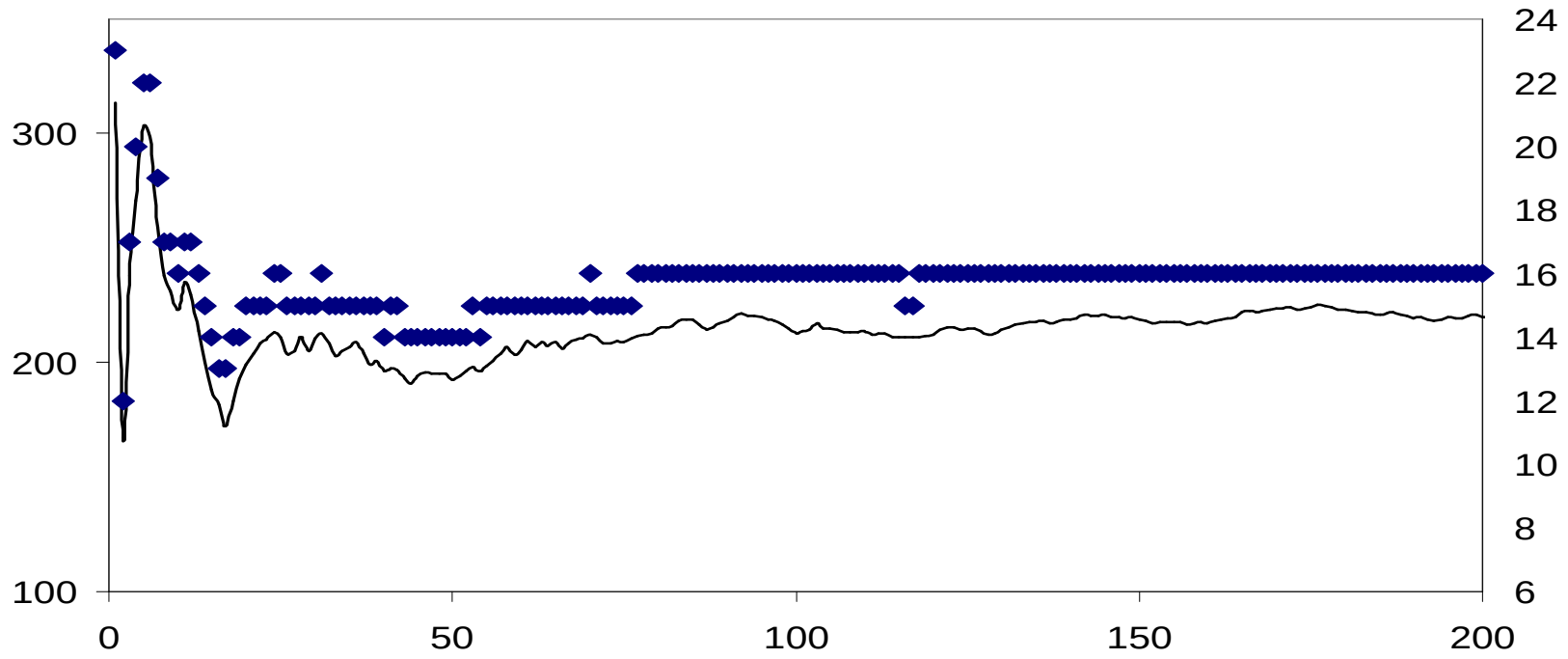
Heuristic

- Tasks Average processing time



Heuristic

- Tasks Average processing time
- Predicted VMs

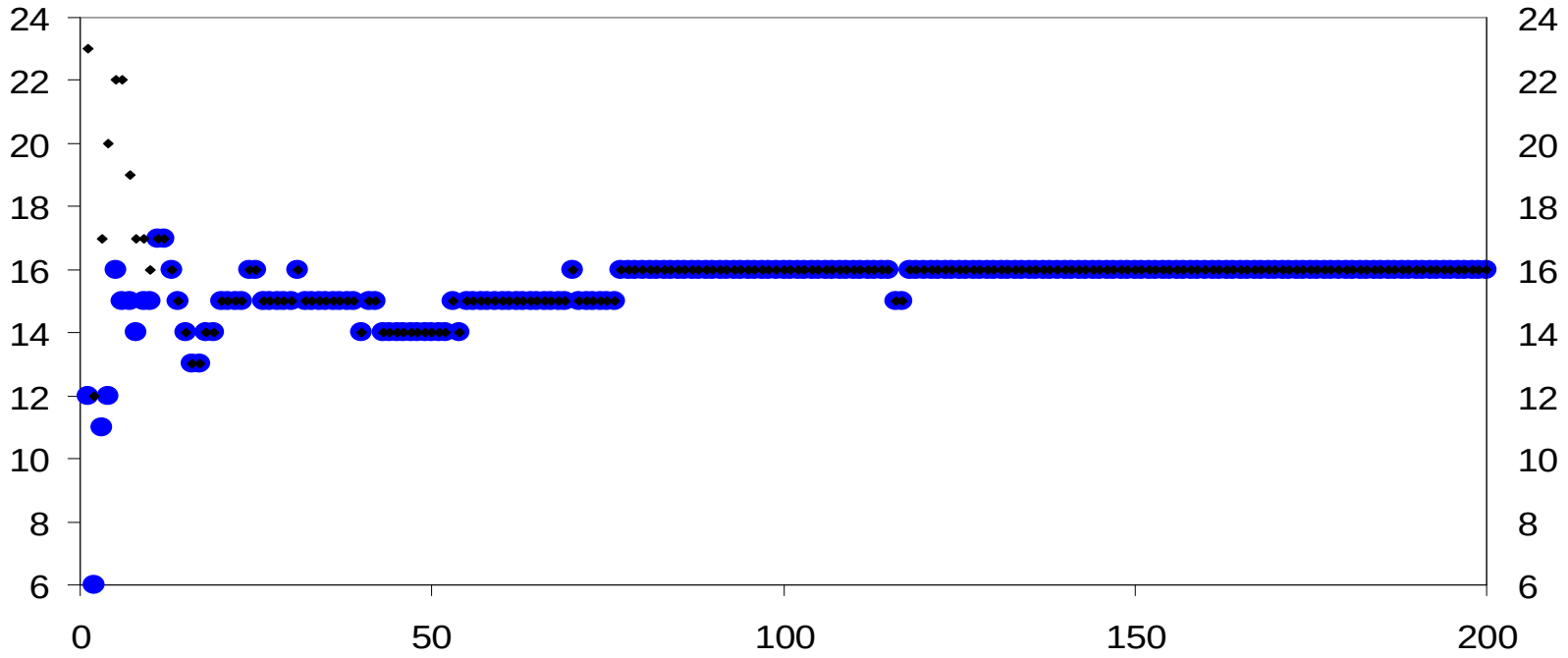


Heuristic

- Use of correction factor (creationRatio)
  - To reduce over allocation
    - On first estimations
  - Initially lower than 1
    - Fewer VMs than predicted are created
  - Converges to 1
    - At a given ratio (increaseRatio)

Heuristic

- Predicted VMs
- Corrected number of VMs

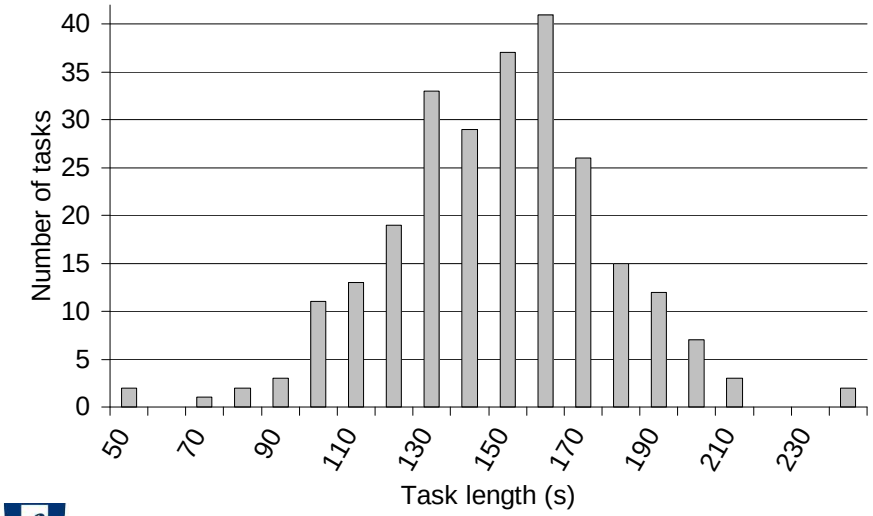


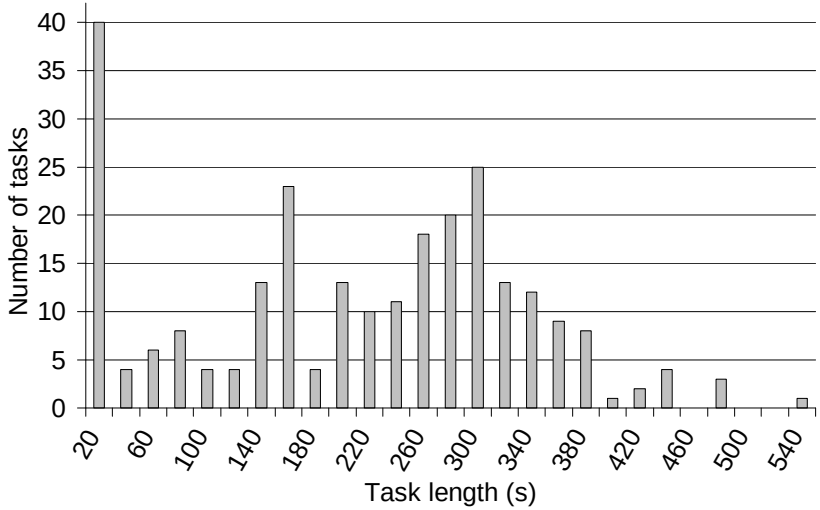
Heuristic

- Calculates needed host VMs
  - Initial corrections mask variability
- Creates host VMs
  - If more hosts are needed
- Turns hosts VMs off
  - Before increases in payment

Evaluation

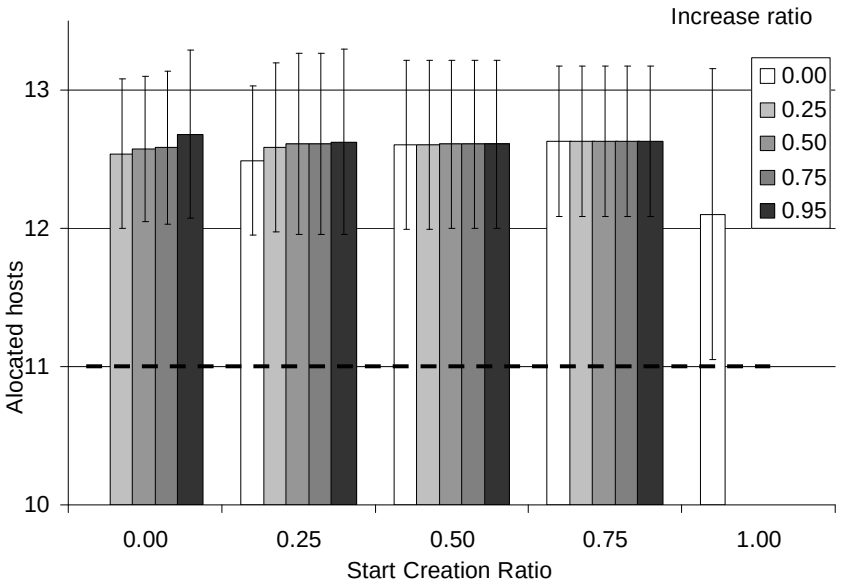
- Allocate optimal number of host VMs
  - Ensure payment is minimum
    - getting maximal speedups
  - Normal distribution
  - Image rendering

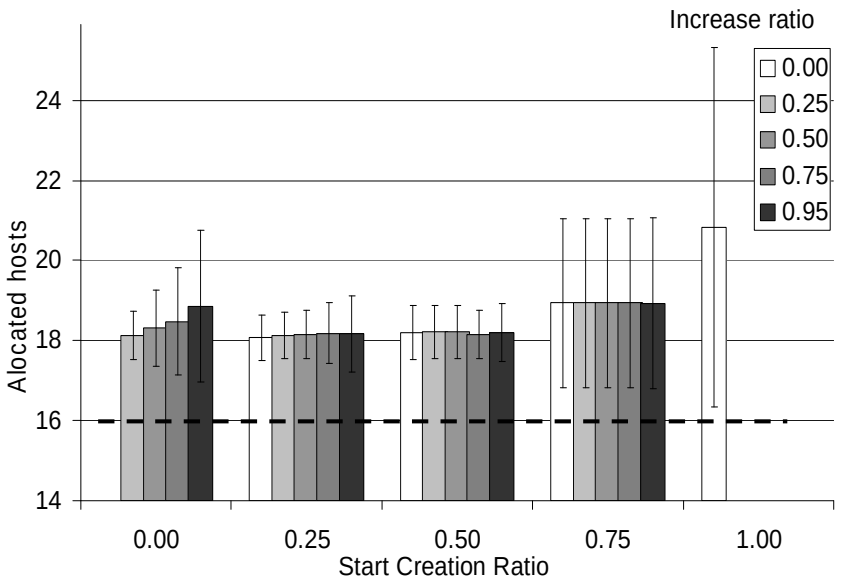




Evaluation

- Allocated hosts and speedups
  - Normal distribution
  - Image rendering



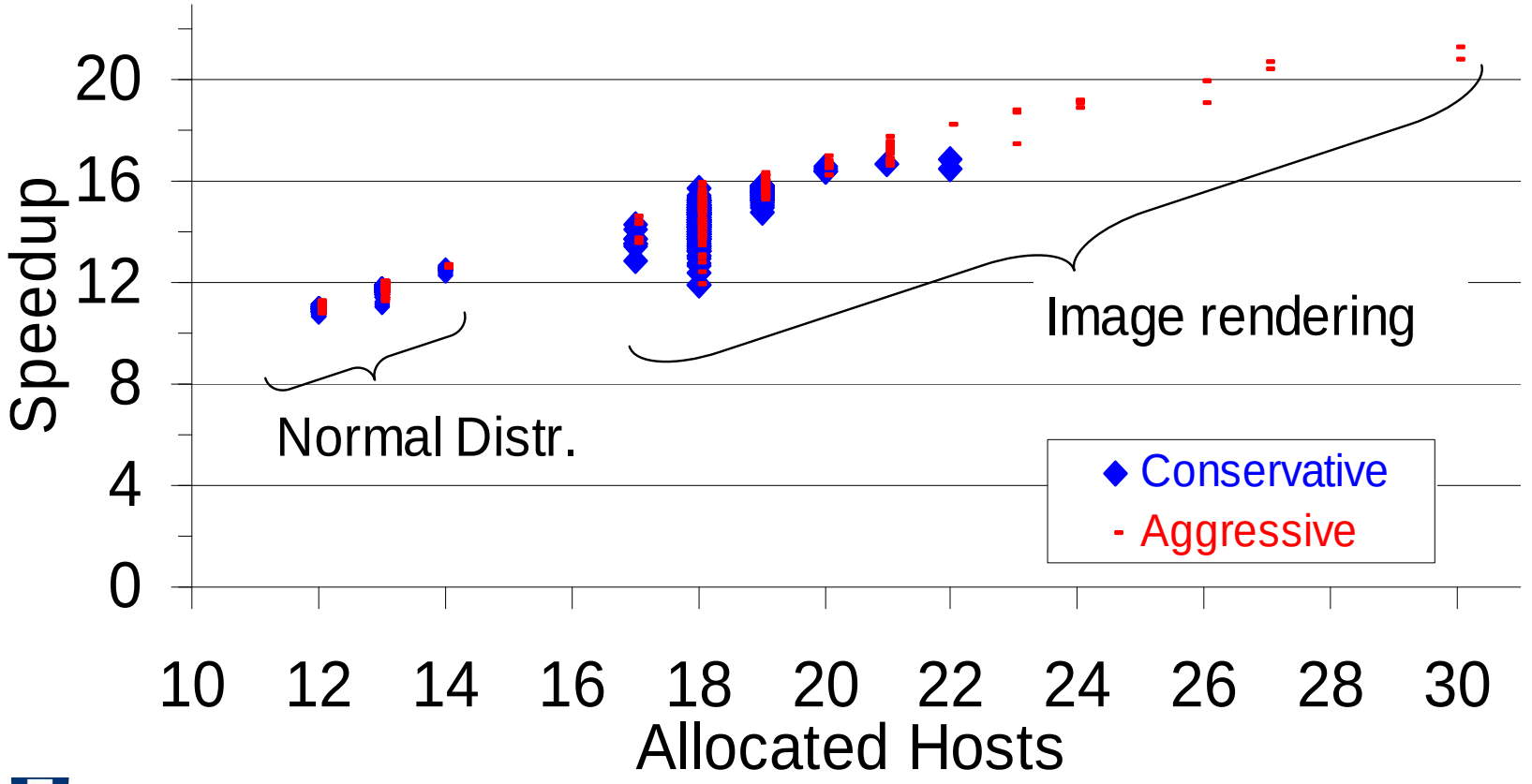


Evaluation

- Allocated hosts vs. speedups
  - Conservative
    - creationRatio 0.5
    - increaseRatio 0.5
  - Aggressive
    - creationRatio 0.75
    - increaseRatio 0.75

Evaluation

- Allocated hosts vs. speedups



Conclusions

- Heuristic to use utility computing infrastructures efficiently
- Guarantees a desired cost
  - Based on equivalent-serial processing time
  - Achieves high speedups
  - Without user intervention
- Protects user from overshooting penalties
  - Prevents over-allocation beyond optimal point

Future Work

- User hints
  - number of initially allocated VMs
    - In order to get better results
- Use previous execution times
  - Based on application, data, user, etc.
  - To predict current execution times
  - Optimize task distribution


Questions?
www.gsd.inesc-id.pt/~jnos
technology
from seed
inesc id
lisboa
Heuristic for resources allocation on utility computing infrastructures