Overhead-Aware-Best-Fit(OABF) Resource Allocation Algorithm for Minimizing VM Launching Overhead

Hao Wu, Shangping Ren, Steven Timm, Gabriele Garzoglio, Seo-Young Noh

Presenter: Hao Wu

Work supported by the U.S. Department of Energy under contract No. DE-AC02-07CH11359
And by joint CRADA FRA 2014-0002 / KISTI-C14014 between KISTI and Fermilab
And supported in part by NSF under grant number CAREER 0746643 and CNS 1018731

Outline

- **Introduction**
  - FermiCloud
- **Cloud Bursting Implementation**
  - vcluster
- **VM Launching Overhead Reference Model Training**
  - VM Launching Overhead Reference Model
- **Overhead-Aware-Best-Fit (OABF) Algorithm**
- **Evaluation**
- **Conclusion**

Introduction: Fermilab

Fermi National Accelerator Laboratory:

- Lead United States particle physics laboratory

Introduction: Fermilab

Fermi National Accelerator Laboratory:

- Lead United States particle physics laboratory
- Data center for:

Introduction: Fermilab

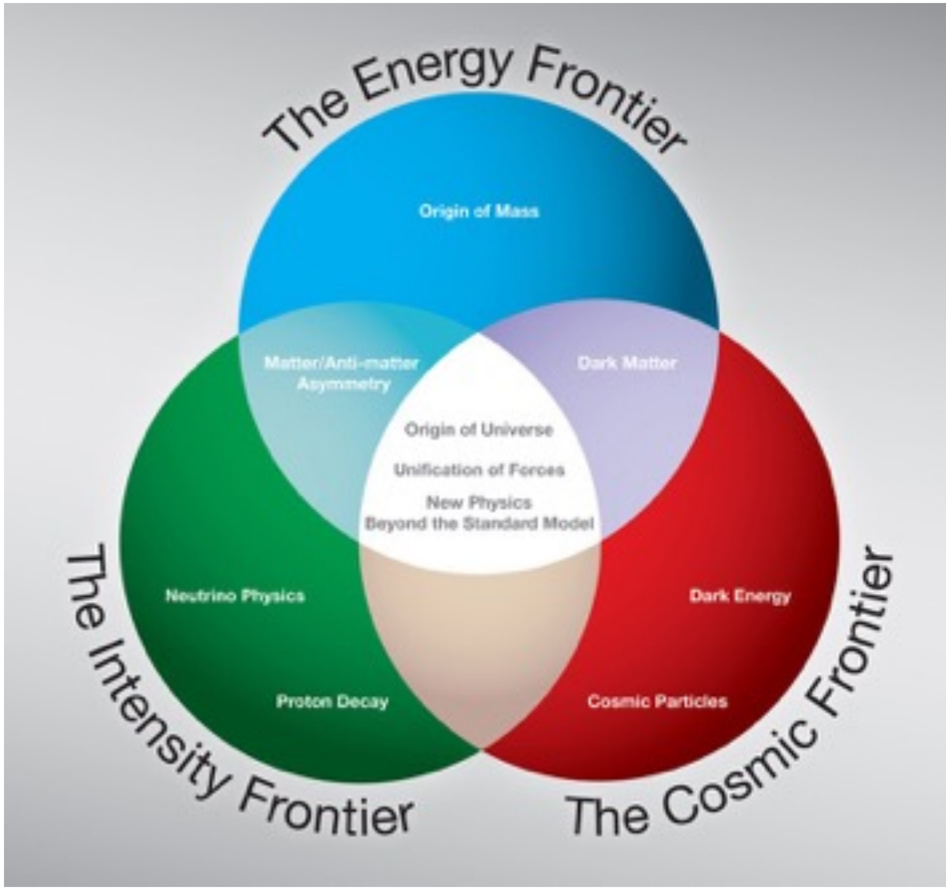
Fermi National Accelerator Laboratory:

- Lead United States particle physics laboratory
- Data center for:

Introduction: Fermilab

Fermi National Accelerator Laboratory:

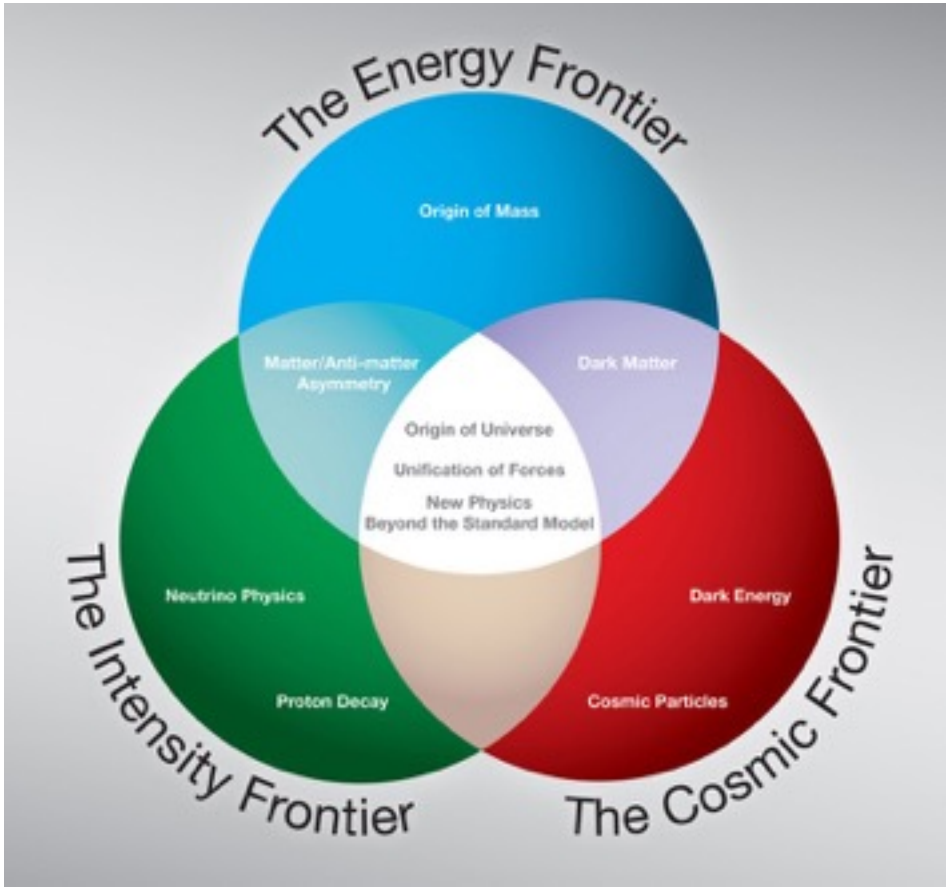
- Lead United States particle physics laboratory
- Data center for:



Introduction: Fermilab

Fermi National Accelerator Laboratory:

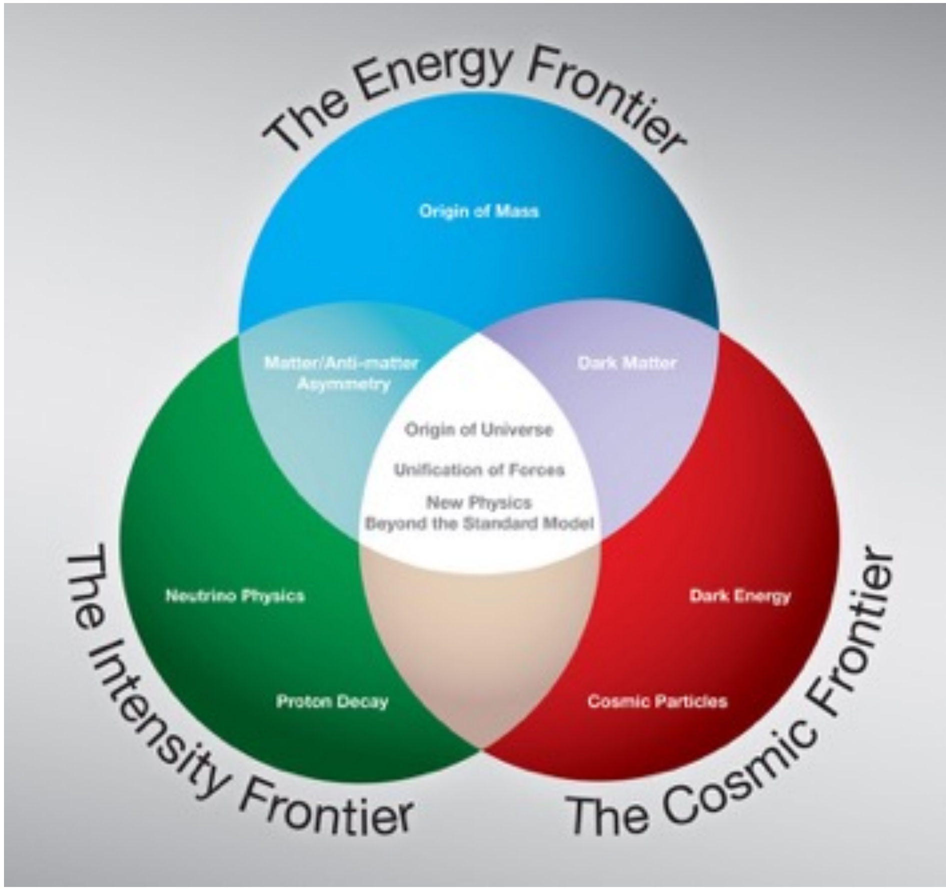
- ☐ Lead United States particle physics laboratory
- ☐ Data center for:
  - Energy Frontier (CDF, D0, CMS)



Introduction: Fermilab

Fermi National Accelerator Laboratory:

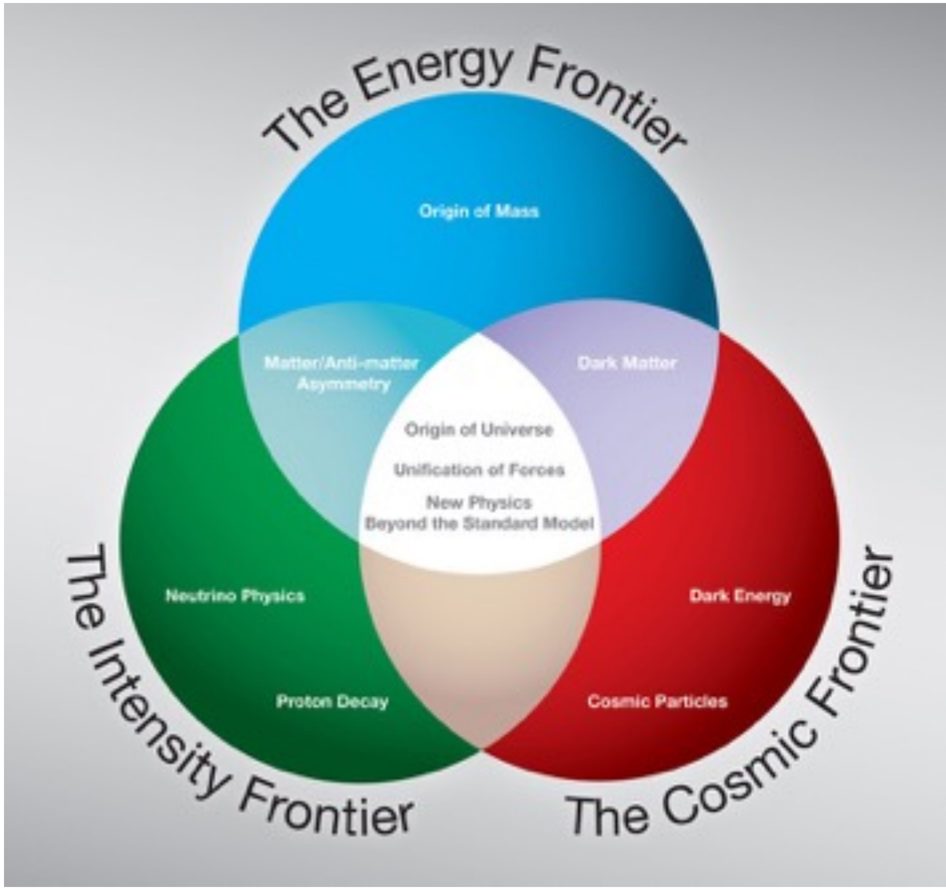
- Lead United States particle physics laboratory
- Data center for:
  - Energy Frontier (CDF, D0, CMS)
  - Cosmic Frontier (DES, LSST, Darkside, etc.)



Introduction: Fermilab

Fermi National Accelerator Laboratory:

- ☐ Lead United States particle physics laboratory
- ☐ Data center for:
  - Energy Frontier (CDF, D0, CMS)
  - Cosmic Frontier (DES, LSST, Darkside, etc.)
  - Intensity Frontier (LBNE, Mu2e, MINOS, etc.)



Introduction: FermiCloud

- FermiCloud Project was established in 2009 with the goal of developing and establishing Scientific Cloud capabilities for the Fermilab Scientific Program.
- FermiCloud Infrastructure as a Service (IaaS) running since 2010.
- FermiCloud Project now focusing on On-demand Services for Scientific Users (PaaS, SaaS).
- The FermiCloud project is a program of work that is split over several overlapping phases.
  - Each phase builds on the capabilities delivered as part of the previous phases.

Introduction: KISTI

Korea Institute of Science and Technology Information:

- Provision of World-Class Infrastructure for Global Collaboration
- Leading Nation-wide Super Computing Infrastructure
- Leading Nation-wide Scientific IaaS Infrastructure

www.kisti.re.kr

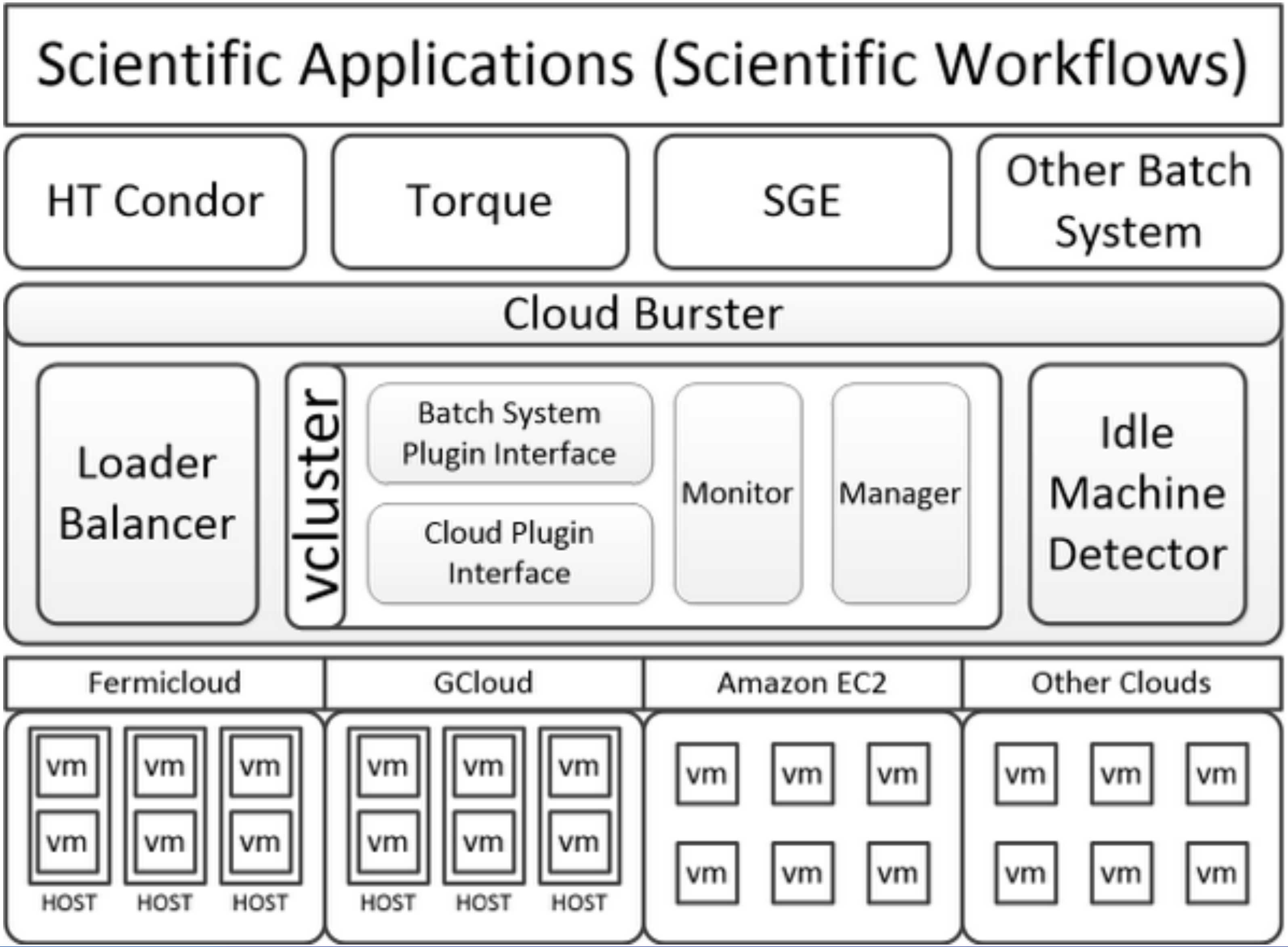
한국과학기술정보연구원

Korea Institute of Science and Technology Information

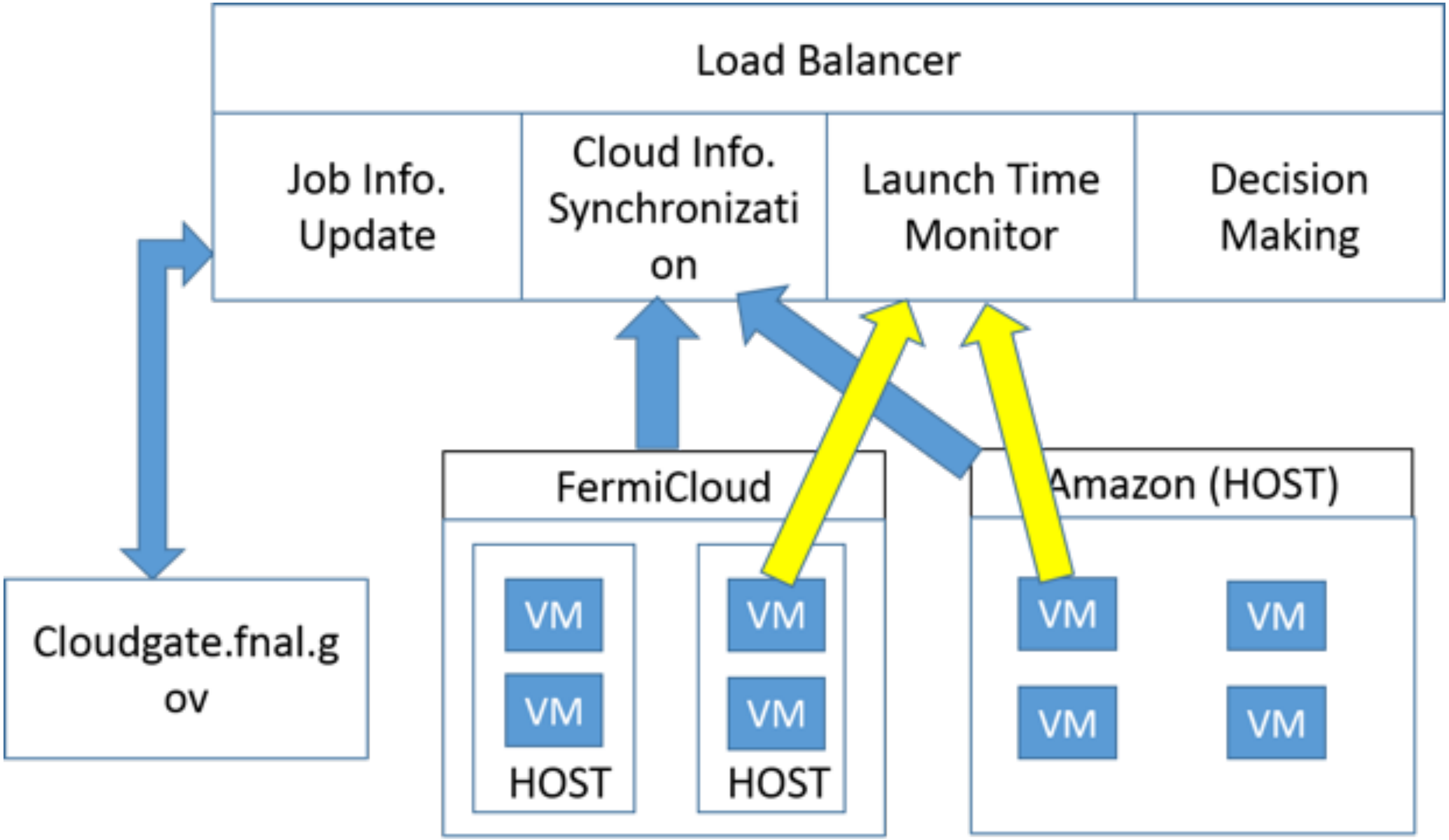
Introduction: Collaboration on Cloud Computing

- Fermilab and KISTI both support global physics collaborations, (CMS, CDF, STAR, ALICE, etc.)
- IIT has joint activities on cloud computing with Fermilab
- These experiments identify cloud computing as key technology for
  - ✓Data preservation
  - ✓Burst capacity
  - ✓Software distribution
- Global collaborations will not converge on single cloud solution.
- Three key components for success:
  - ✓Federation—Group of users run transparently on many resources
  - ✓Interoperability—Find common method that works on all resources
  - ✓Throughput—Data flow to and from resources across WAN.
- Goal is to enable real scientific stakeholders to get their computing done.

Cloud Bursting Implementation: vcluster



Design of the Load Balancer



Cloud Bursting

Cloud Bursting

- Load Balancing

Cloud Bursting

- Load Balancing
  - When to burst

Cloud Bursting

- Load Balancing
  - When to burst
  - Where to burst

Cloud Bursting

- Load Balancing
  - When to burst
  - Where to burst
  - What to burst

Cloud Bursting

- Load Balancing
  - When to burst
  - Where to burst
  - What to burst
- Impact of VM Launching Overhead

Cloud Bursting

- Load Balancing
  - When to burst
  - Where to burst
  - What to burst
- Impact of VM Launching Overhead
- Scenario 1: VM Launched after Job finished

Cloud Bursting

- Load Balancing
  - When to burst
  - Where to burst
  - What to burst
- Impact of VM Launching Overhead
- Scenario 1: VM Launched after Job finished

  Resource Waste!

Cloud Bursting

- Load Balancing
  - When to burst
  - Where to burst
  - What to burst
- Impact of VM Launching Overhead
- Scenario 1: VM Launched after Job finished

  Resource Waste!
- Scenario 2: Simultaneous Launch

Cloud Bursting

- Load Balancing
  - When to burst
  - Where to burst
  - What to burst
- Impact of VM Launching Overhead
- Scenario 1: VM Launched after Job finished

  Resource Waste!
- Scenario 2: Simultaneous Launch

  Over Provisioning!

Cloud Bursting

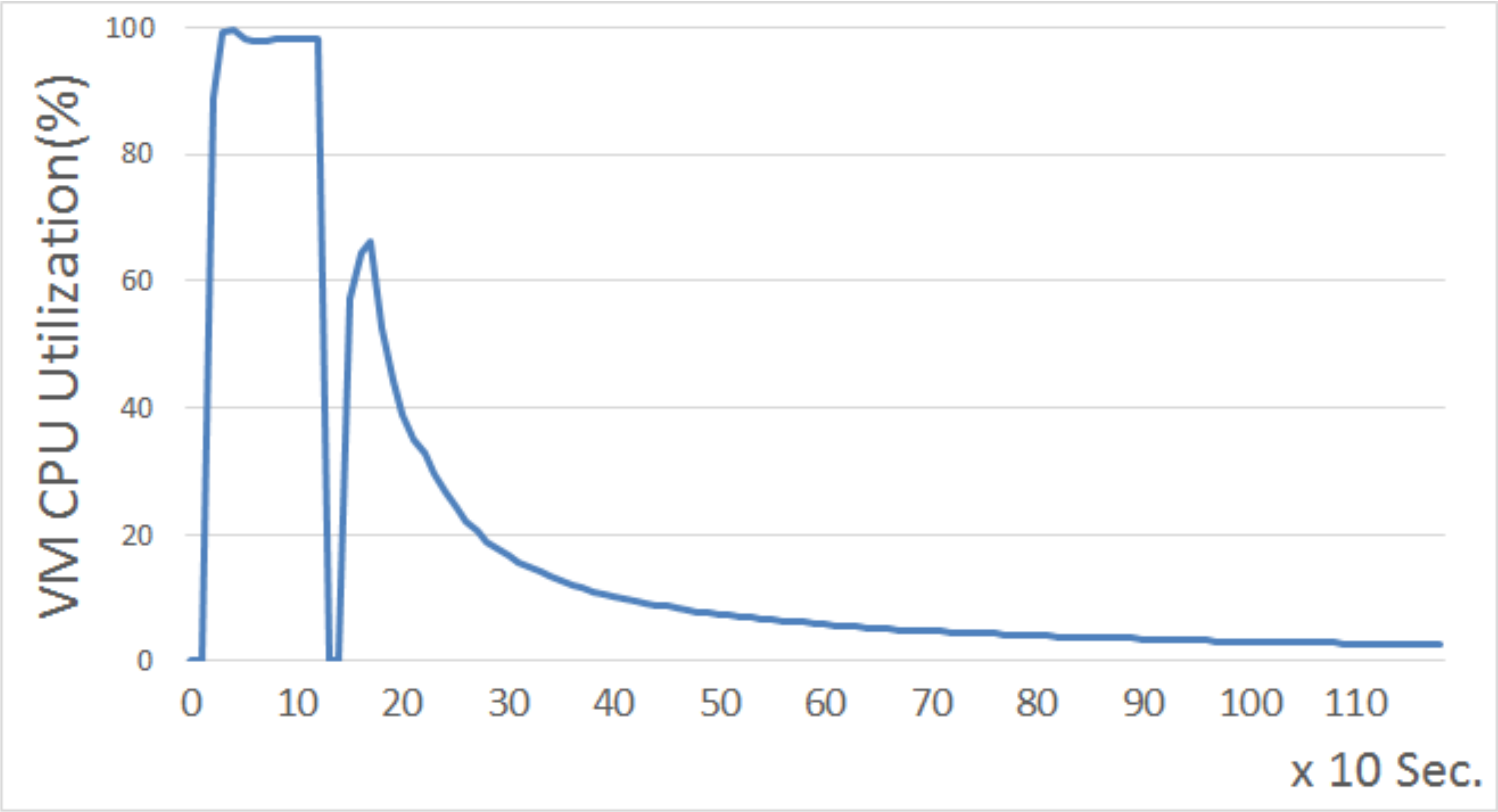
- Load Balancing
  - When to burst
  - Where to burst
  - What to burst
- Impact of VM Launching Overhead
- Scenario 1: VM Launched after Job finished

  Resource Waste!
- Scenario 2: Simultaneous Launch

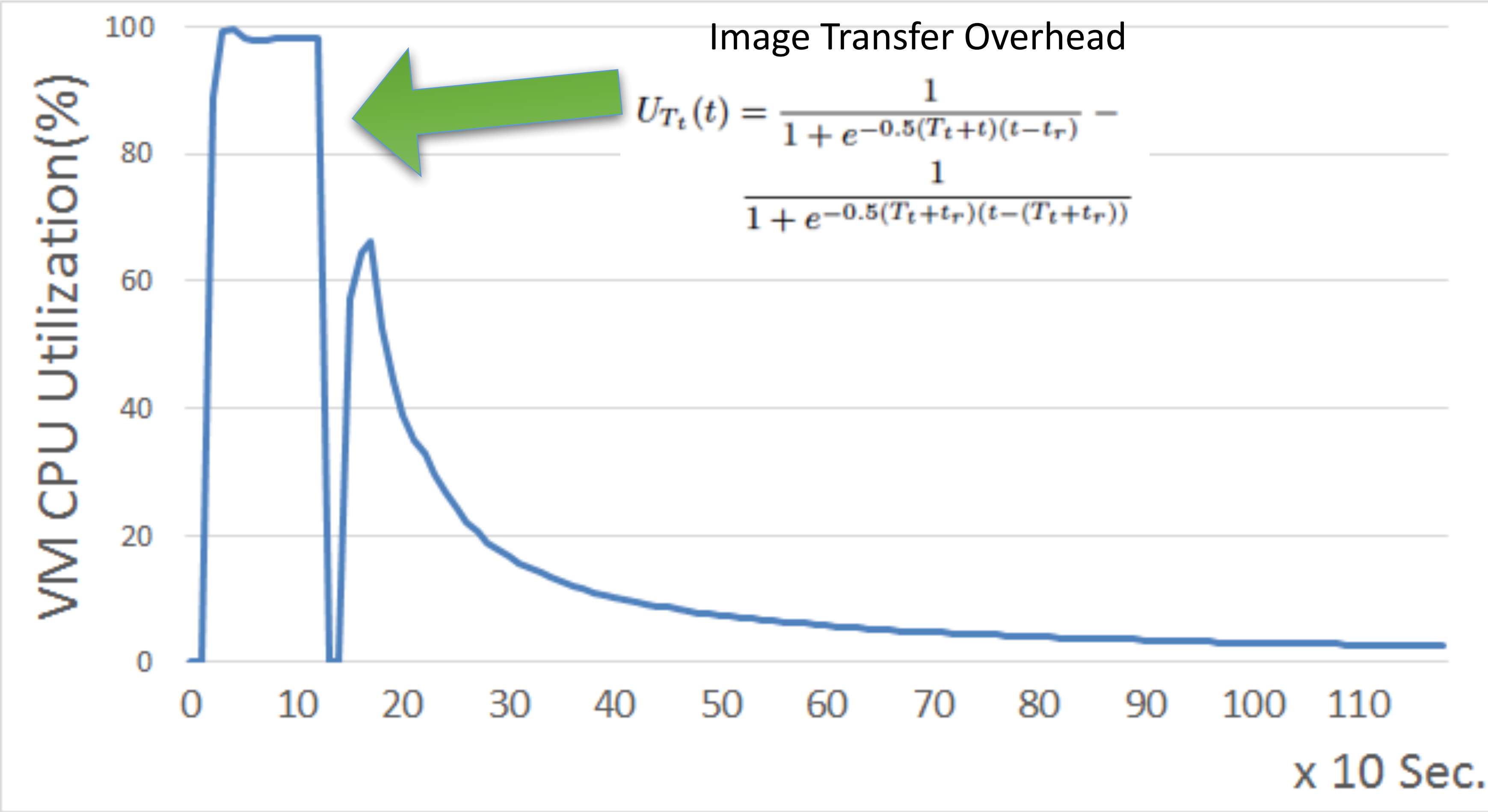
  Over Provisioning!

To minimize the VM launching overhead!

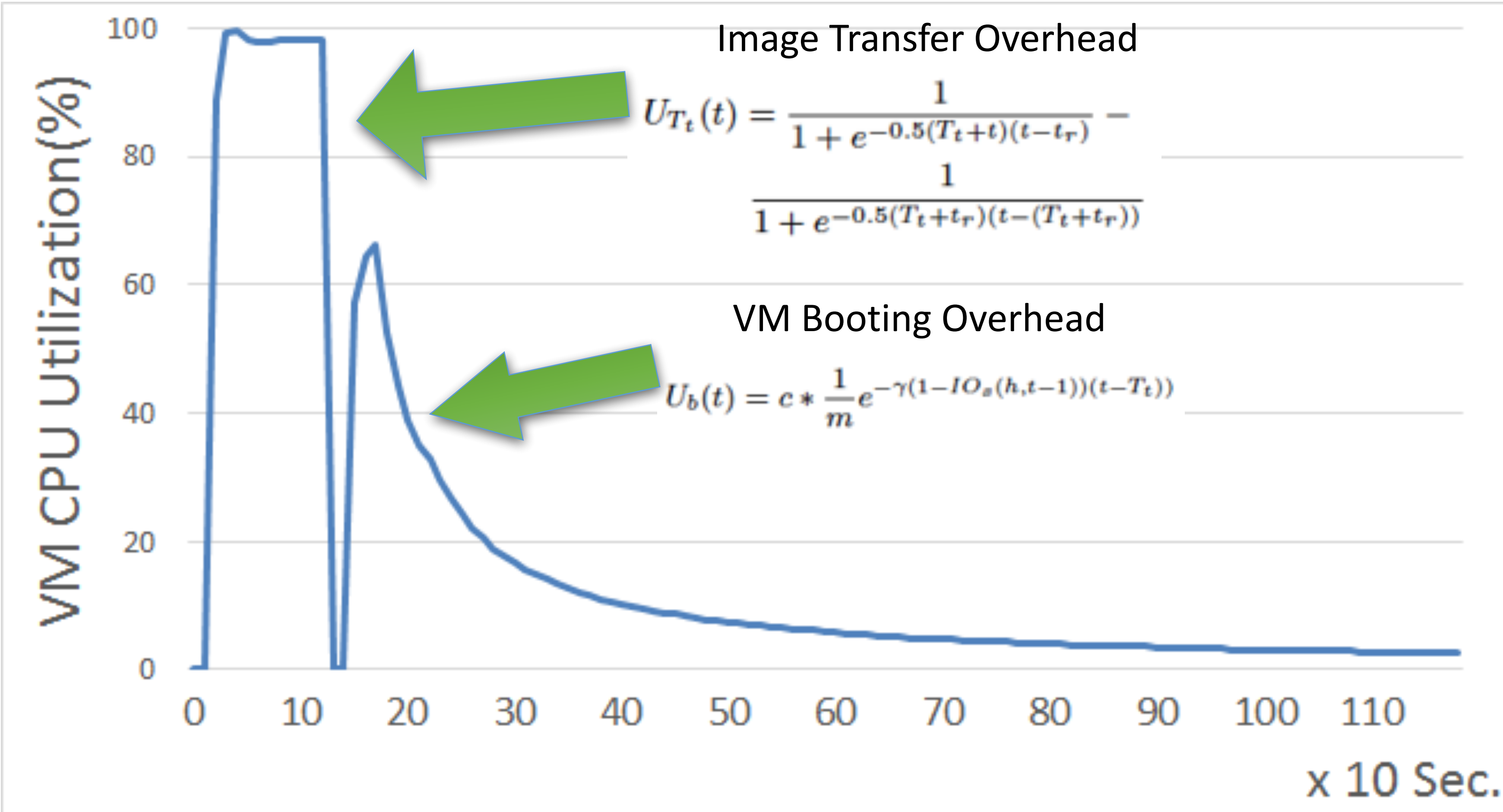
VM Launching Overhead



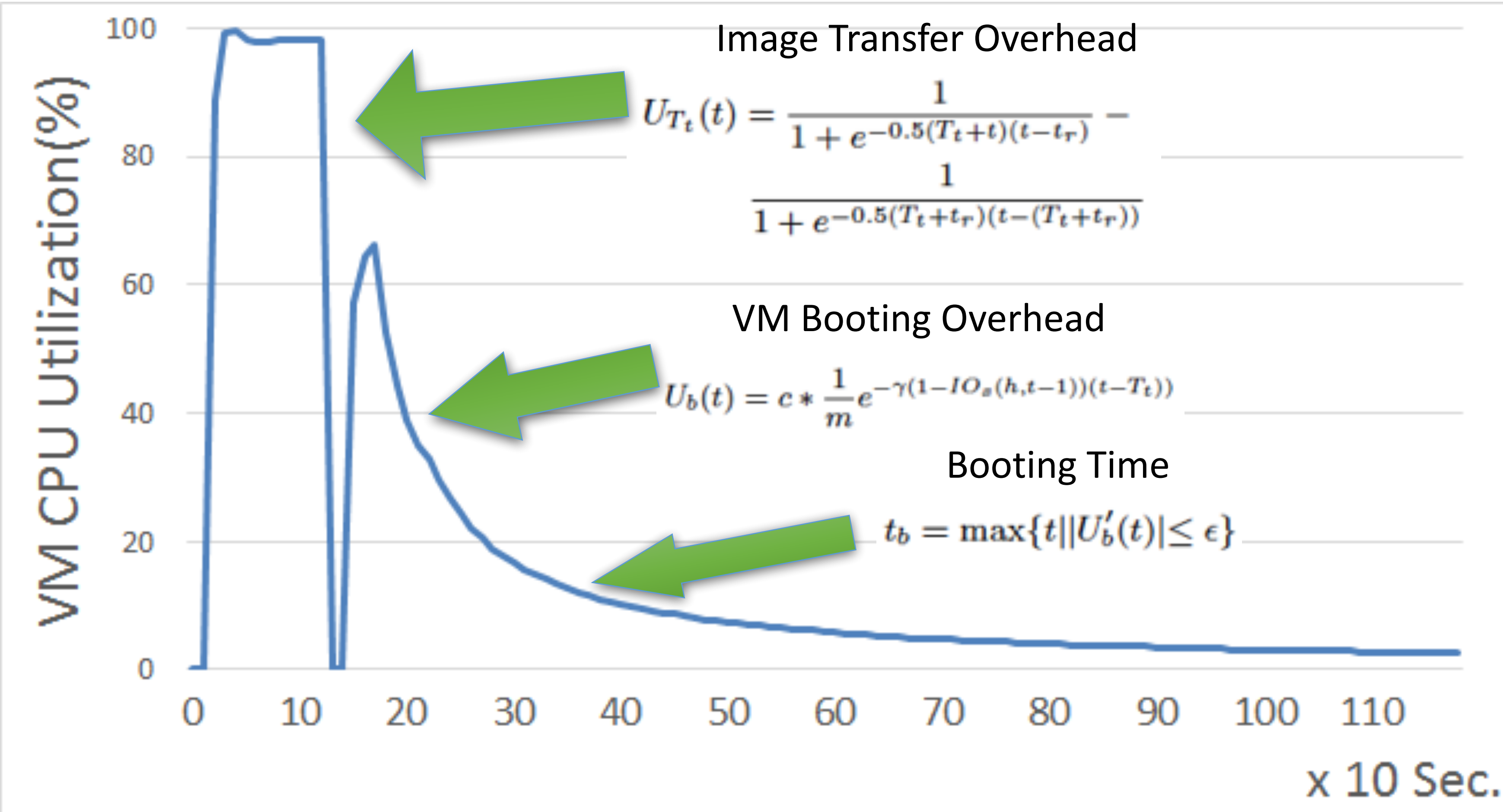
VM Launching Overhead



VM Launching Overhead



VM Launching Overhead



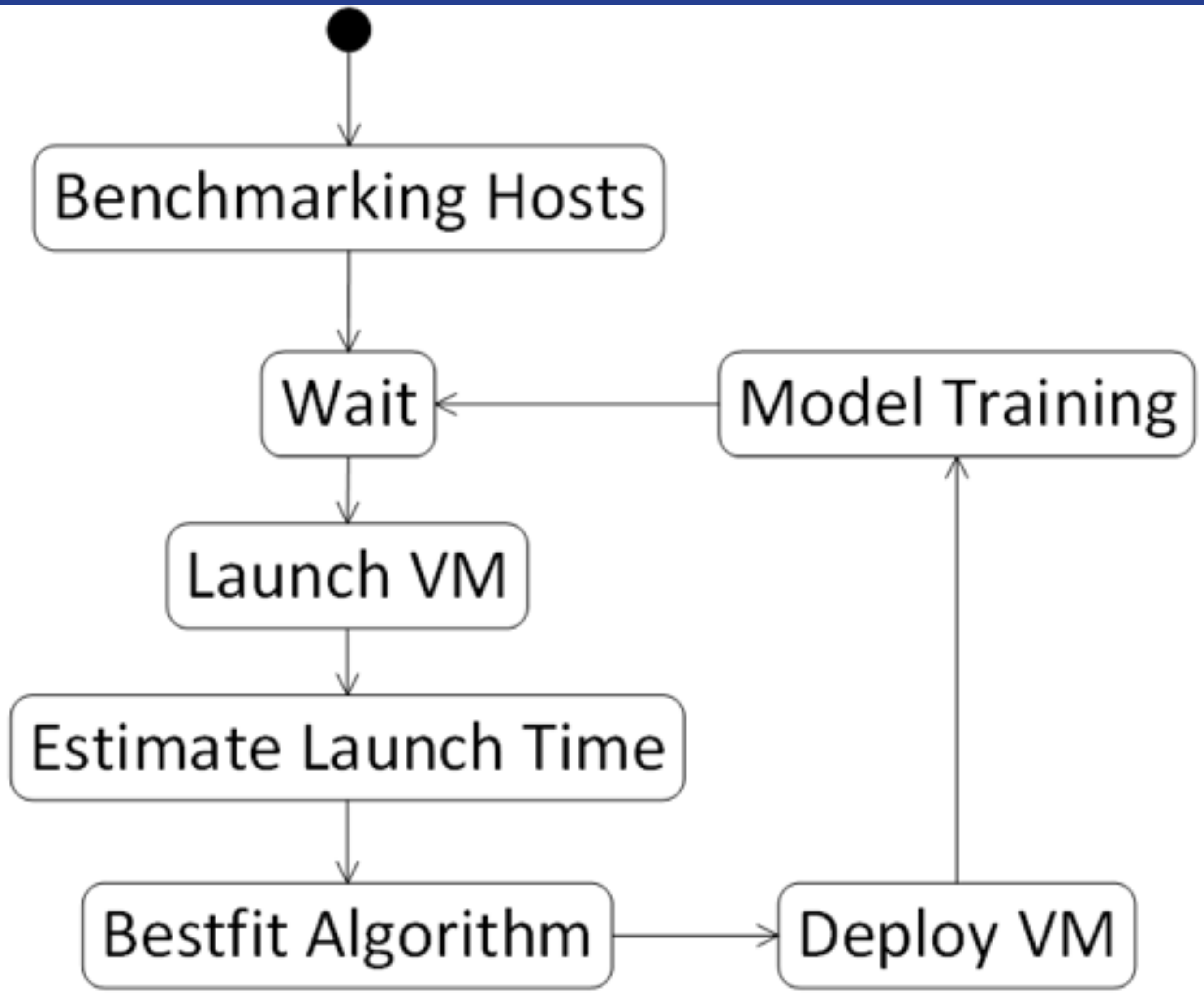
VM Launching Overhead Model Training

- Prediction Accuracy Evaluation
- Mean Absolute Scaled Error:

$$MASE = mean(|q_t|)$$

$$q_t = \frac{e_t}{\frac{1}{n-1}\sum_{i=2}^{n}|Y_i - Y_{i-1}|}$$

Resource Allocation Automation Workflow



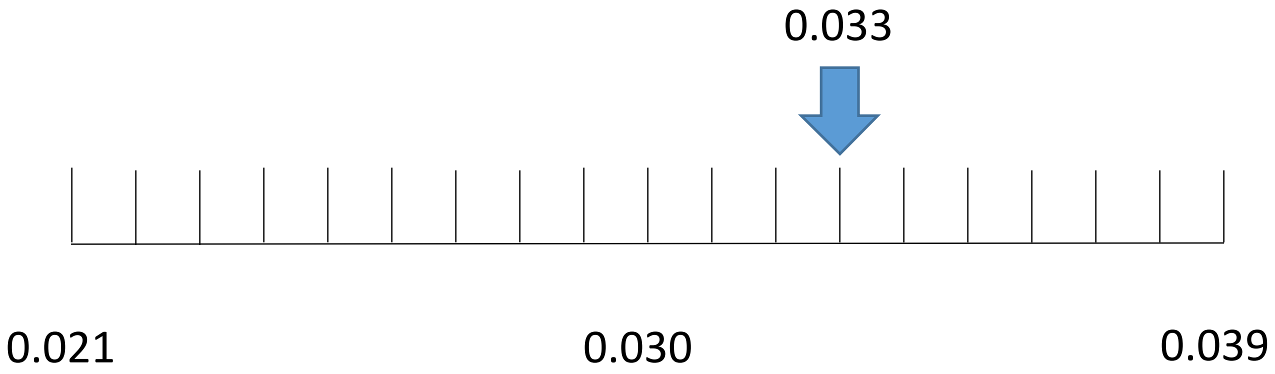
Model Training Algorithm

Model Training Algorithm

Assume the calibrated error from last round training is 0.032

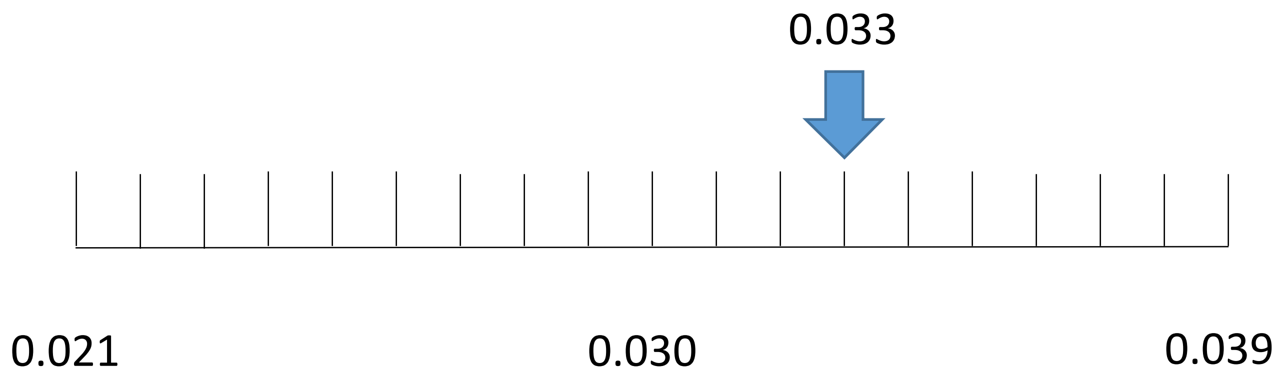
Model Training Algorithm

Assume the calibrated error from last round training is 0.032
Step1: Round the error to 0.030



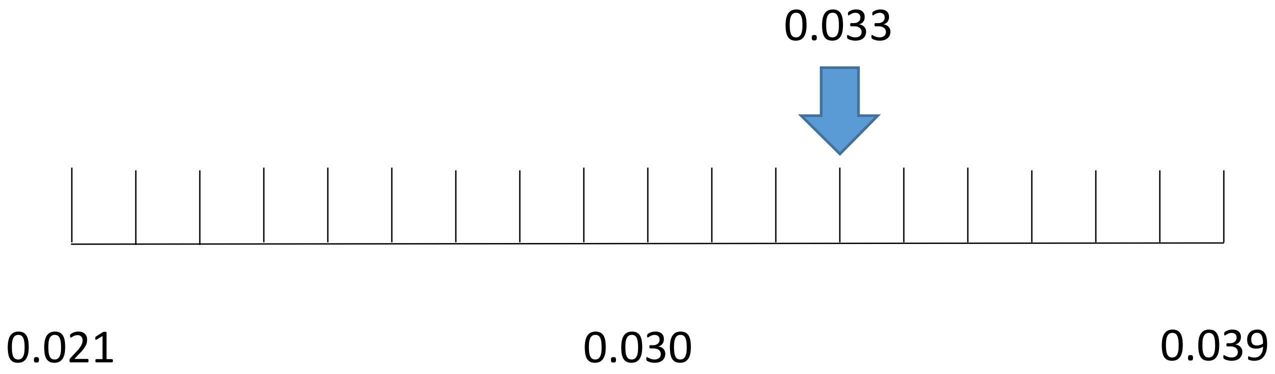
Model Training Algorithm

Assume the calibrated error from last round training is 0.032
Step1: Round the error to 0.030
Step2: Calculate the precision of the error: 0.001



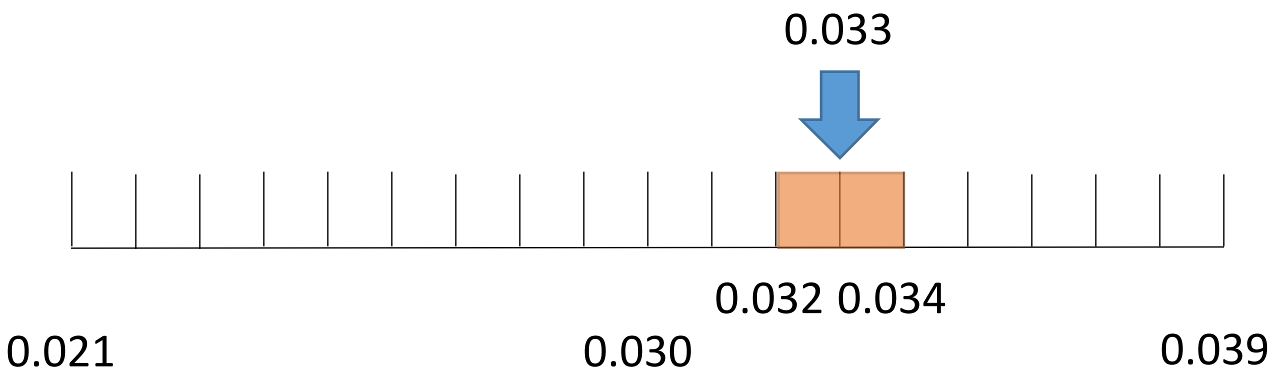
Model Training Algorithm

Assume the calibrated error from last round training is 0.032
Step1: Round the error to 0.030
Step2: Calculate the precision of the error: 0.001
Step3: Calculate MASE for each error within range 0.021 - 0.039



Model Training Algorithm

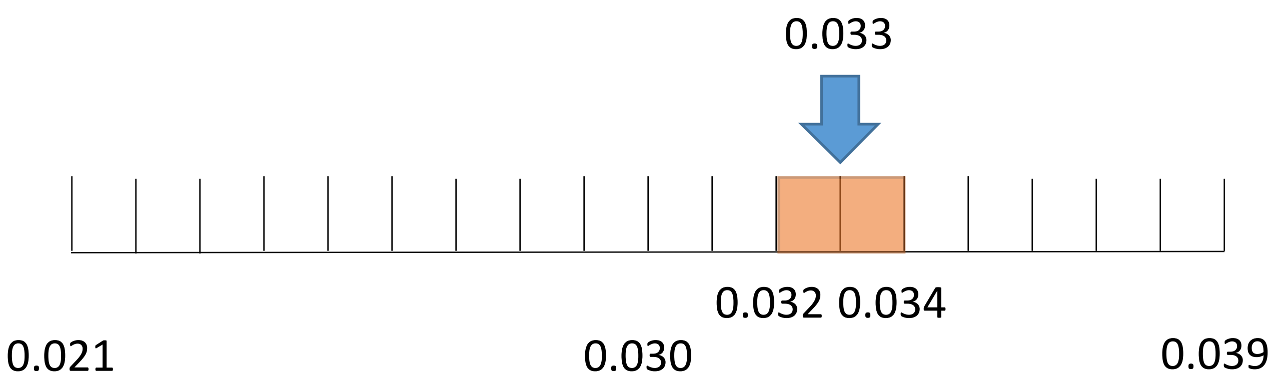
Assume the calibrated error from last round training is 0.032
Step1: Round the error to 0.030
Step2: Calculate the precision of the error: 0.001
Step3: Calculate MASE for each error within range 0.021 - 0.039
Step4: Choose the new error with minimal MASE: 0.033



Fermilab KISTI IIT

Model Training Algorithm

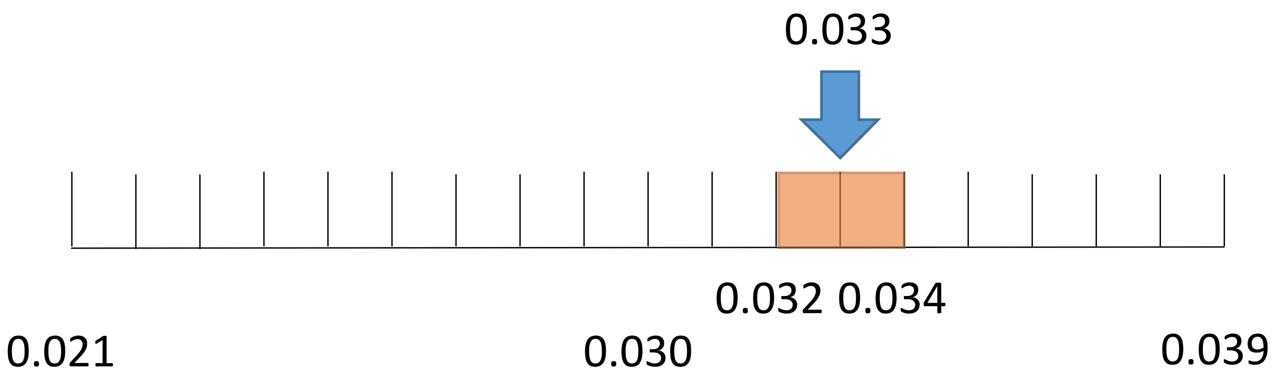
Assume the calibrated error from last round training is 0.032
Step1: Round the error to 0.030
Step2: Calculate the precision of the error: 0.001
Step3: Calculate MASE for each error within range 0.021 - 0.039
Step4: Choose the new error with minimal MASE: 0.033
Termination Condition:



Fermilab KISTI IIT

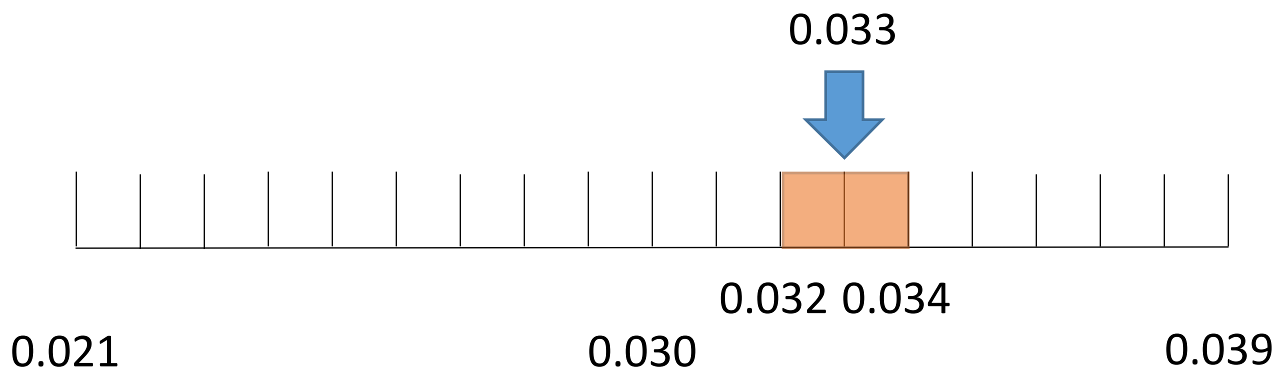
Model Training Algorithm

Assume the calibrated error from last round training is 0.032
Step1: Round the error to 0.030
Step2: Calculate the precision of the error: 0.001
Step3: Calculate MASE for each error within range 0.021 - 0.039
Step4: Choose the new error with minimal MASE: 0.033
Termination Condition:
1: Error is smaller than threshold



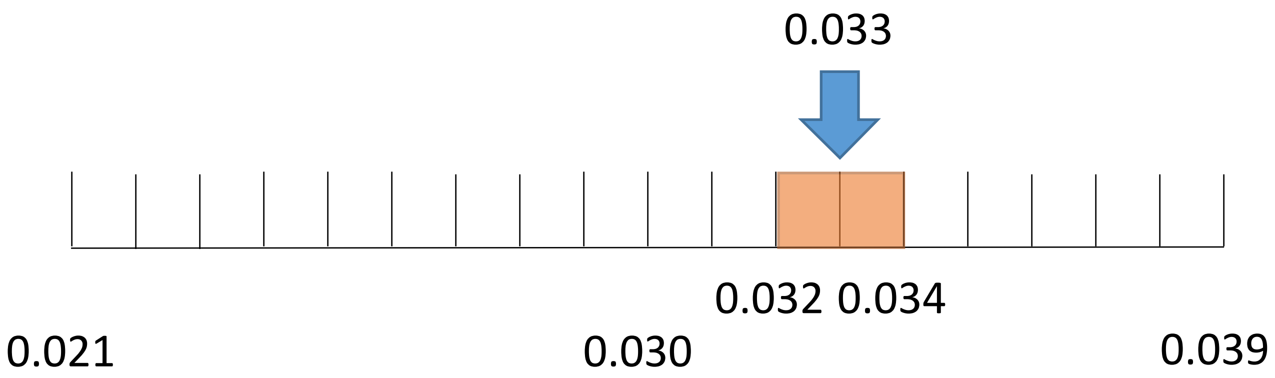
Model Training Algorithm

Assume the calibrated error from last round training is 0.032
Step1: Round the error to 0.030
Step2: Calculate the precision of the error: 0.001
Step3: Calculate MASE for each error within range 0.021 - 0.039
Step4: Choose the new error with minimal MASE: 0.033
Termination Condition:
1: Error is smaller than threshold
2: Training round is smaller than threshold



Model Training Algorithm

Assume the calibrated error from last round training is 0.032
Step1: Round the error to 0.030
Step2: Calculate the precision of the error: 0.001
Step3: Calculate MASE for each error within range 0.021 - 0.039
Step4: Choose the new error with minimal MASE: 0.033
Termination Condition:
1: Error is smaller than threshold
2: Training round is smaller than threshold



Overhead-Aware-Best-Fit (OABF) Algorithm

Algorithm 2: overhead-aware-best-fit Algorithm

Input : Empty Virtual Machine $v = \{t_r, h, t_w\}$, Host set $H = \{h_1, \ldots, h_n\}$, VM waiting queue $Q = \{v_1^q, \ldots, v_m^q\}$

Output: Virtual Machine $v = \{t_r, h, t_w\}$ with host and waiting time information

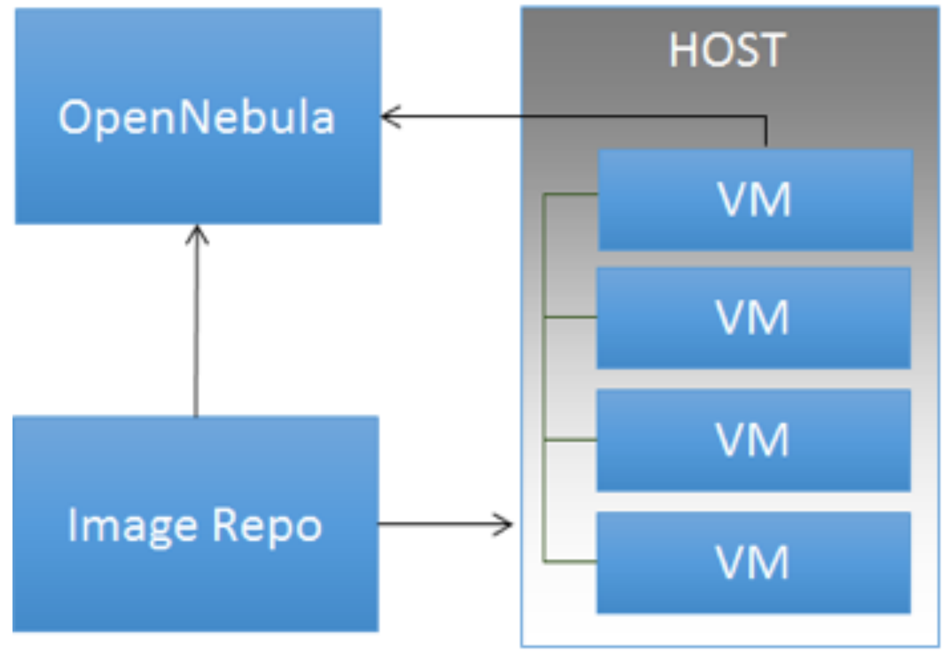
$v.h \leftarrow$ null; $v.t_w \leftarrow 0$; $h' \leftarrow null$
$t'_w \leftarrow 0$; $t_b \leftarrow \infty$; $t'_r \leftarrow v.t_r$
for $i \leftarrow 1$ *to* n **do**
    $v.t_r \leftarrow t'_r$
    $t_p \leftarrow$ calculatePredictLaunchTime(h_i, v)
    if $t_p \leq t_b$ **then**
        $t_b \leftarrow t_p$; $h' \leftarrow h_i$
    end
    for $j \leftarrow 1$ *to* m **do**
        if v_j^q *is deployed on* h_i **then**
            $t_t \leftarrow v_j^q$'s predicted image transfer time
            if $t_t + v_j^q.t_r + v_j^q.t_w \geq v.t_r$ **then**
                $v.t_r \leftarrow t_t + v_j^q.t_r + v_j^q.t_w$
                $t_p \leftarrow$ calculatePredictLaunchTime(h_i, v)
                if $t_p + t_t + v_j^q.t_r + v_j^q.t_w - v.t_r \leq t_b$ **then**
                    $t_b \leftarrow t_p$; $h' \leftarrow h_i$
                    $t'_w \leftarrow t_t + v_j^q.t_r + v_j^q.t_w - v.t_r$
                end
            end
        end
    end
end
$v.t_r \leftarrow t'_r$; $v.h \leftarrow h'$; $v.t_w \leftarrow t'_w$
return v

Evaluation: Environment

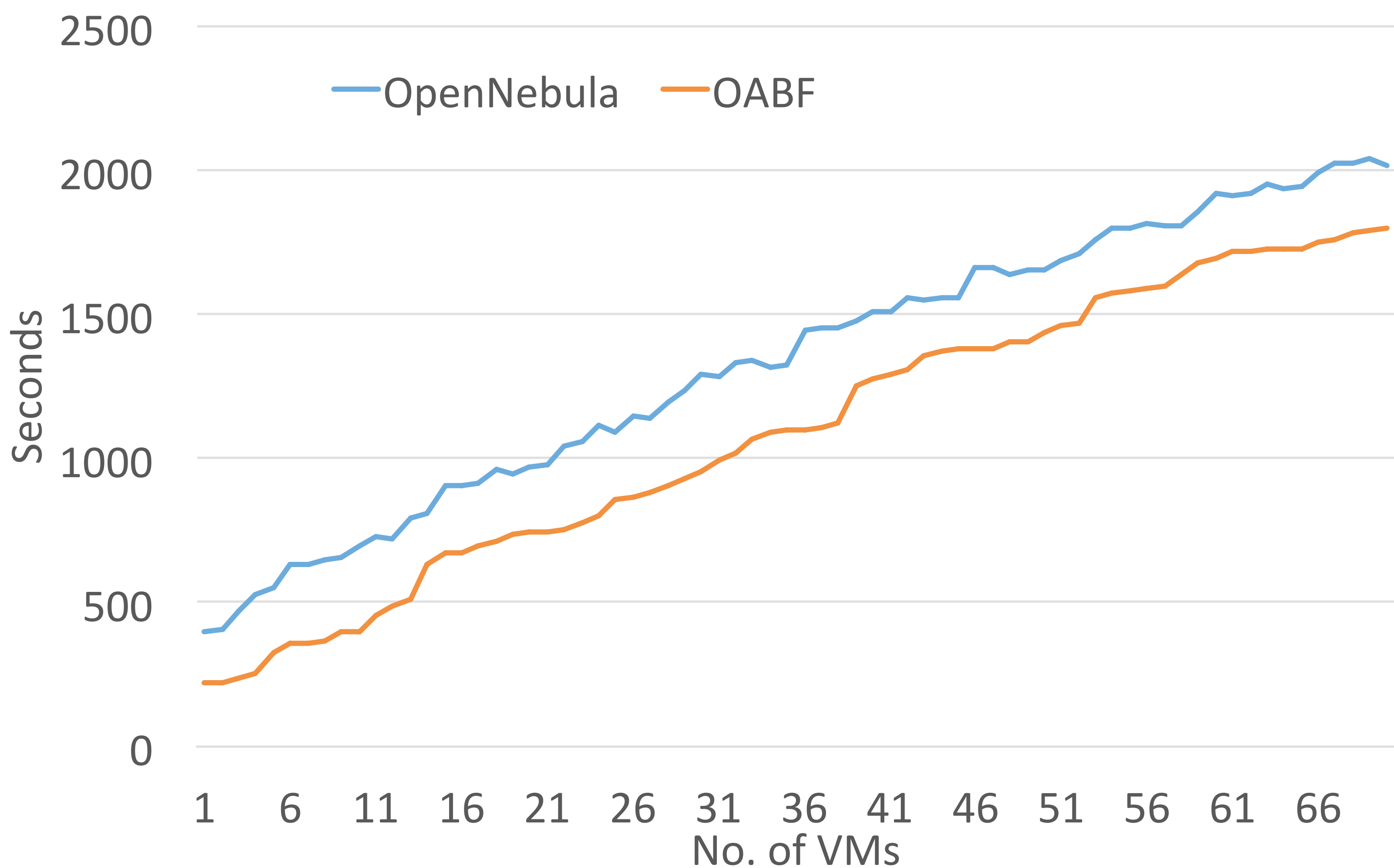
System Configuration:

- Front-end: 16-core Intel(R)Xeon(R) CPU E5640 @ 2.67GHz, 48GB memory.
- 15 VM hosts: 8-core Intel(R) Xeon(R) CPU X5355 @ 2.66GHz and 16GB memory

- Cloud Platform: OpenNebula

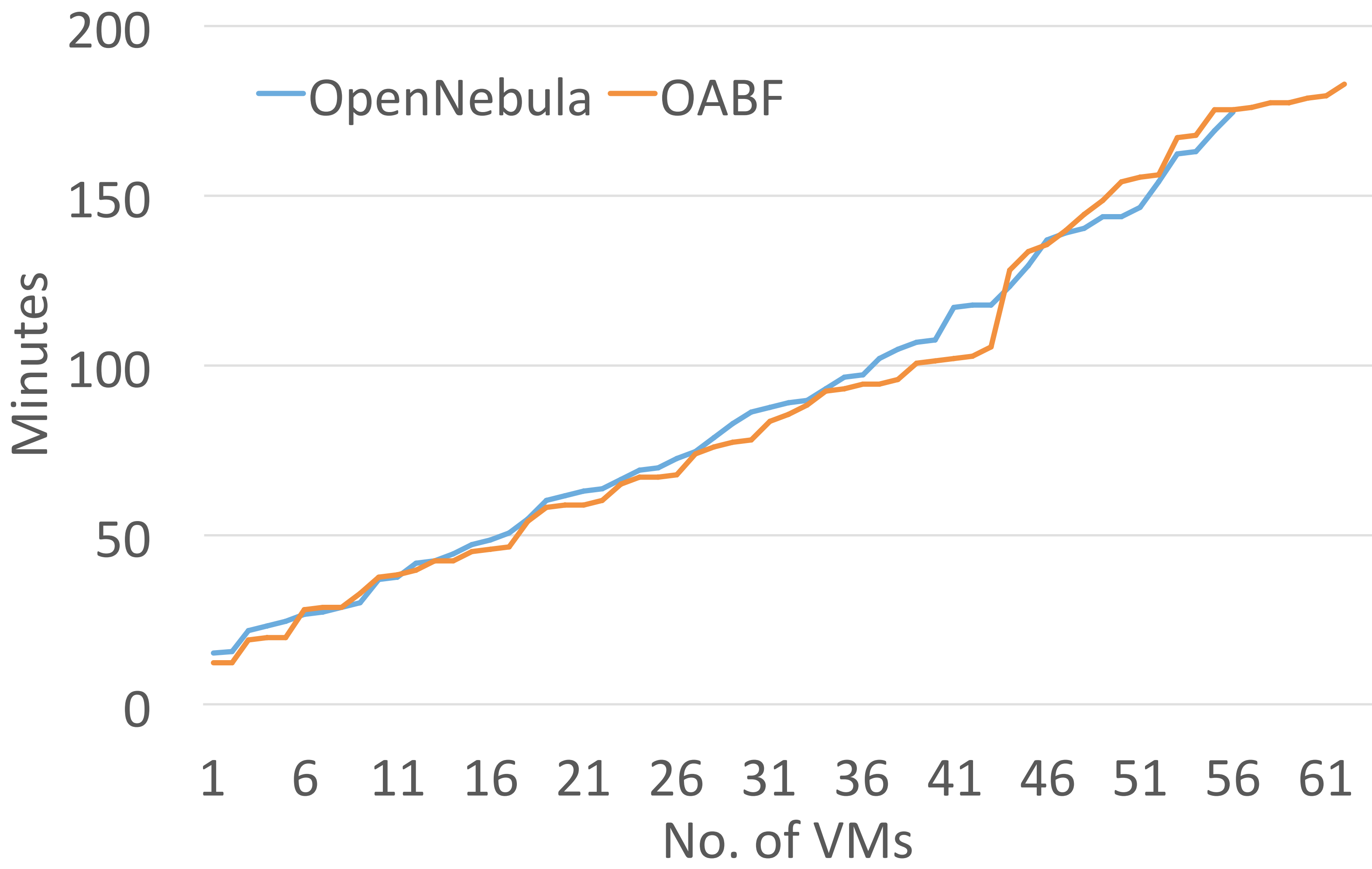


Evaluation: Small Image (2.6GB QCOW2)



Fermilab KISTI IIT

Evaluation: Large Image (16GB RAW)



Fermilab KISTI IIT

Conclusion

- Present a mechanism to automatically train the VM launching overhead reference model that we previously developed.
- Based on the virtual machine launching overhead reference model, we have developed in this paper an overhead-aware-best-t resource allocation algorithm to help the cloud reduce the average VM launching time.
- Presented an implementation of the developed OABF algorithm on FermiCloud
- For relatively small VM images commonly used in FermiCloud, OABF provides significant improvement in launch time.

Thanks!

Questions?