SmartHarvest: Harvesting Idle CPUs Safely and Efficiently in the Cloud

Yawen Wang[1], Kapil Arya[2], Marios Kogias[2], Manohar Vanga[3],
Aditya Bhandari[2], Neeraja Yadwadkar[1], Siddhartha Sen[2],
Sameh Elnikety[2], Christos Kozyrakis[1], Ricardo Bianchini[2]

[1]Stanford University [2]Microsoft [3]Nokia Bell Labs, Germany

EuroSys 2021

CPU Underutilization in Datacenter Servers

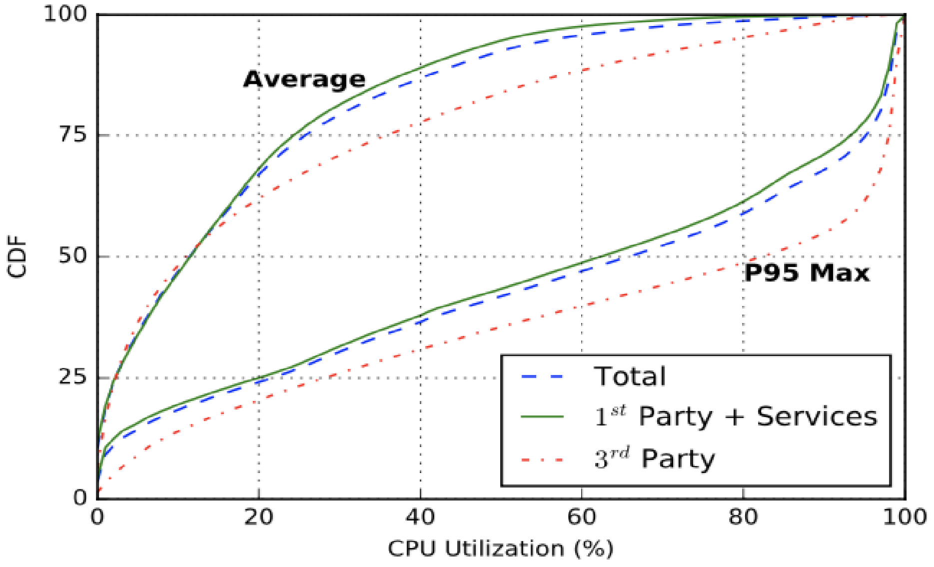


Figure 1: Average and P95 of max CPU utilizations. [1]

[1] Cortez et al. "Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms ." *SOSP'17*

CPU Underutilization in Datacenter Servers

60% of the virtual machines (VMs) in Azure have an average CPU utilization < 20%

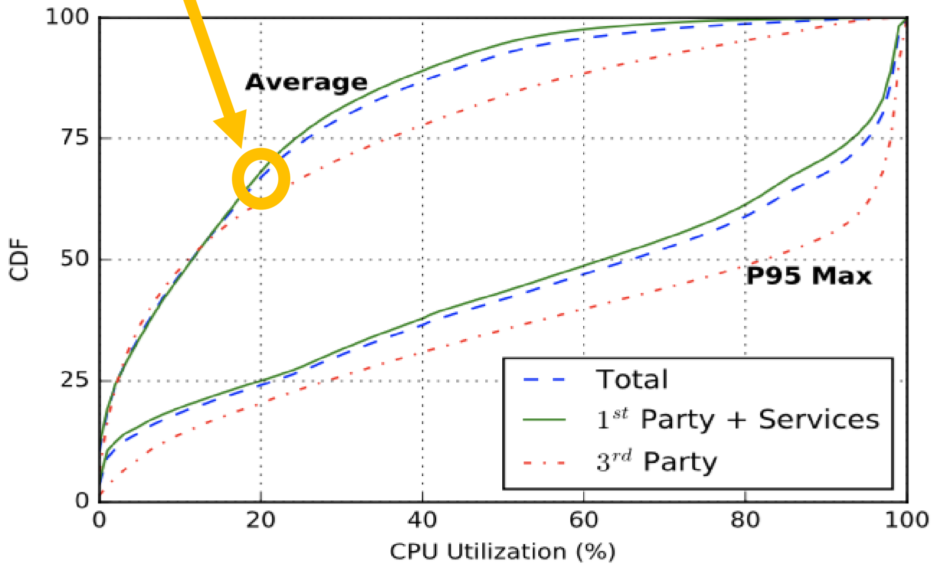


Figure 1: Average and P95 of max CPU utilizations. [1]

[1] Cortez et al. "Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms ." *SOSP'17*

CPU Underutilization in Datacenter Servers

60% of the virtual machines (VMs) in Azure have an average CPU utilization < 20%

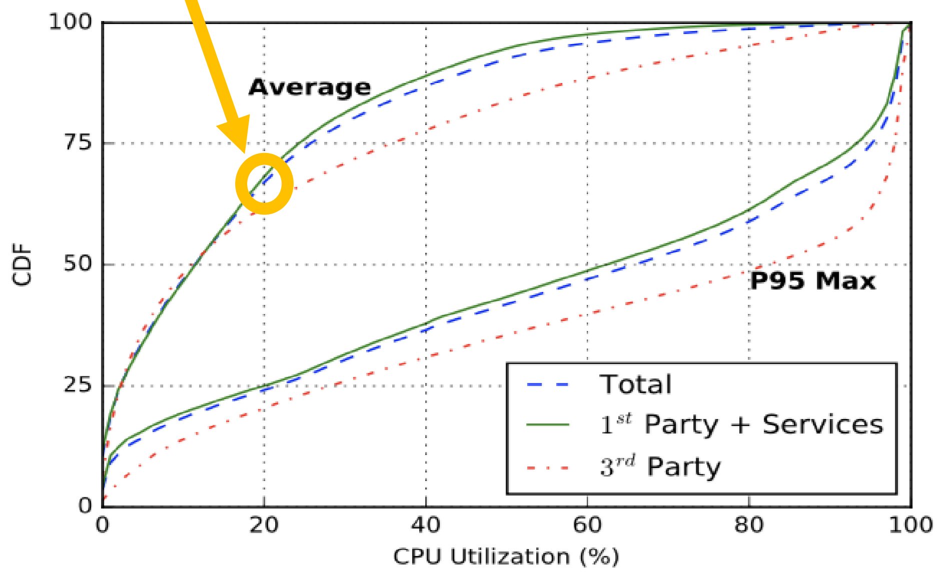


Figure 1: Average and P95 of max CPU utilizations. [1]

Reason for low CPU utilization in VMs

- VMs are often oversized for peak load
- Common for user-facing workloads

[1] Cortez et al. "Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms ." *SOSP'17*

Prior approaches to increase CPU utilization

- Use spare CPU resources from latency-sensitive workloads to run batch processing tasks
  - Extensive offline workload profiling (e.g. PerfIso[2])
  - Knowledge of application characteristics (e.g. Heracles[3], Shenango[4])

[2] Iorgulescu, Călin, et al. "Perfiso: Performance isolation for commercial latency-sensitive services." ATC'18
[3] Lo, David, et al. "Heracles: Improving resource efficiency at scale." ISCA'15
[4] Ousterhout, Amy, et al. "Shenango: Achieving high CPU efficiency for latency-sensitive datacenter workloads." NSDI'19

Prior approaches to increase CPU utilization

- Use spare CPU resources from latency-sensitive workloads to run batch processing tasks
  - Extensive offline workload profiling (e.g. PerfIso[2])
  - Knowledge of application characteristics (e.g. Heracles[3], Shenango[4])

Challenge: VMs are opaque boxes in public cloud

[2] Iorgulescu, Călin, et al. "Perfiso: Performance isolation for commercial latency-sensitive services." ATC'18
[3] Lo, David, et al. "Heracles: Improving resource efficiency at scale." ISCA'15
[4] Ousterhout, Amy, et al. "Shenango: Achieving high CPU efficiency for latency-sensitive datacenter workloads." NSDI'19

Prior approaches to increase CPU utilization

- Use spare CPU resources from latency-sensitive workloads to run batch processing tasks
  - Extensive offline workload profiling (e.g. PerfIso[2])
  - Knowledge of application characteristics (e.g. Heracles[3], Shenango[4])

Challenge: VMs are opaque boxes in public cloud

1. Rely only on monitoring of low-level proxies (e.g. CPU usage)
2. Assume any VM may be latency-sensitive

[2] Iorgulescu, Călin, et al. "Perfiso: Performance isolation for commercial latency-sensitive services." ATC'18
[3] Lo, David, et al. "Heracles: Improving resource efficiency at scale." ISCA'15
[4] Ousterhout, Amy, et al. "Shenango: Achieving high CPU efficiency for latency-sensitive datacenter workloads." NSDI'19

Proposed Solution: SmartHarvest

Proposed Solution: SmartHarvest

- Use **online learning** to continuously learn and predict future CPU utilization of opaque-box primary, customer VMs based on past CPU usage

Proposed Solution: SmartHarvest

- Use **online learning** to continuously learn and predict future CPU utilization of opaque-box primary, customer VMs based on past CPU usage
- **Safely harvest idle cores** to run batch processing jobs inside ElasticVM

Proposed Solution: SmartHarvest

- Use **online learning** to continuously learn and predict future CPU utilization of opaque-box primary, customer VMs based on past CPU usage
- **Safely harvest idle cores** to run batch processing jobs inside ElasticVM
- Employ a **two-level safeguard** to reduce performance impact on primary VMs when learning misbehaves

Proposed Solution: SmartHarvest

- Use **online learning** to continuously learn and predict future CPU utilization of opaque-box primary, customer VMs based on past CPU usage
- **Safely harvest idle cores** to run batch processing jobs inside ElasticVM
- Employ a **two-level safeguard** to reduce performance impact on primary VMs when learning misbehaves
- Dynamically allocate cores among VMs to
    1. **Minimize** impact on primary VMs (e.g., no more than 10%)
    2. **Maximize** harvested spare CPU resources

A new type of VM: ElasticVM

A new type of VM: ElasticVM

- Uses idle cores from primary VMs to runs batch workloads

A new type of VM: ElasticVM

- Uses idle cores from primary VMs to runs batch workloads
- Minimum guaranteed resources
  - E.g. 1 core, 8GB memory, 10GB SSD

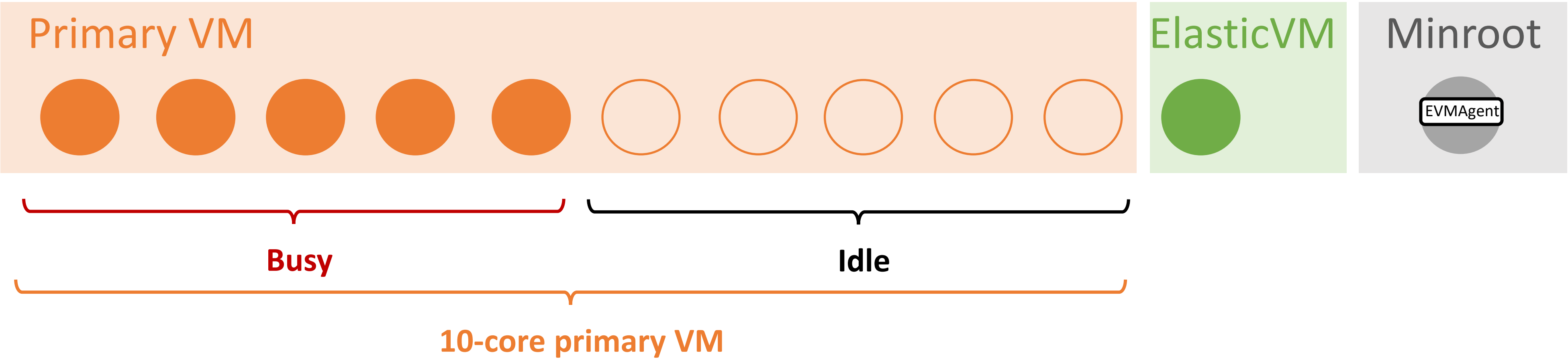
A new type of VM: ElasticVM

- Uses idle cores from primary VMs to runs batch workloads
- Minimum guaranteed resources
  - E.g. 1 core, 8GB memory, 10GB SSD
- Lower priority than primary VMs
  - Its number of assigned physical cores dynamically grows and shrinks

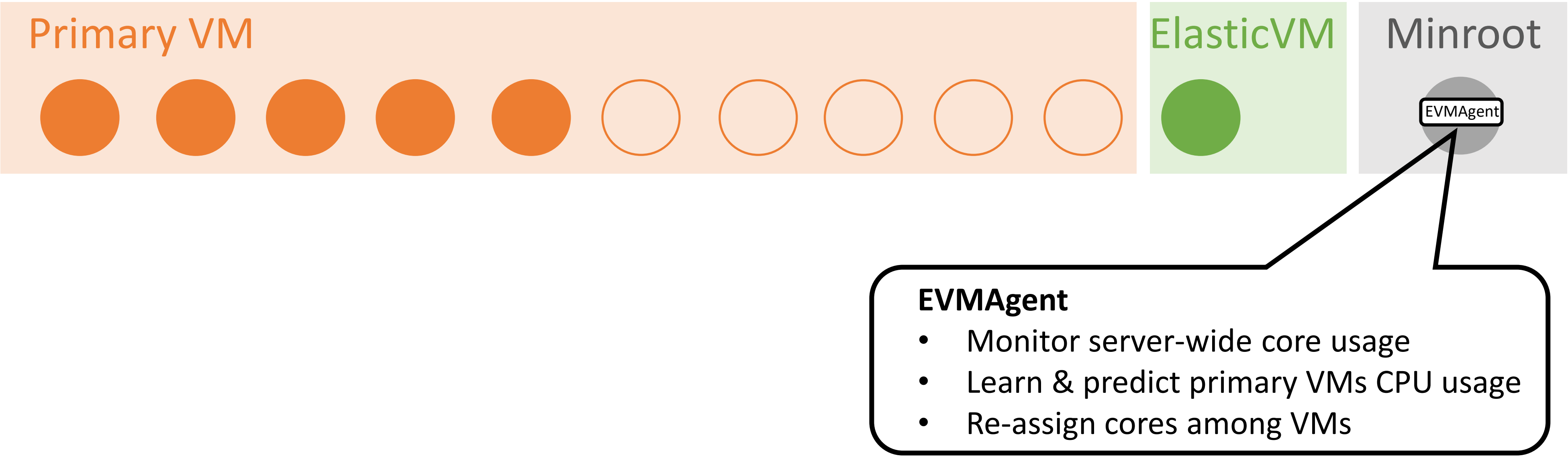
A new type of VM: ElasticVM

- Uses idle cores from primary VMs to runs batch workloads
- Minimum guaranteed resources
  - E.g. 1 core, 8GB memory, 10GB SSD
- Lower priority than primary VMs
  - Its number of assigned physical cores dynamically grows and shrinks
- Configured to have as many virtual cores as the total number of physical cores on the server

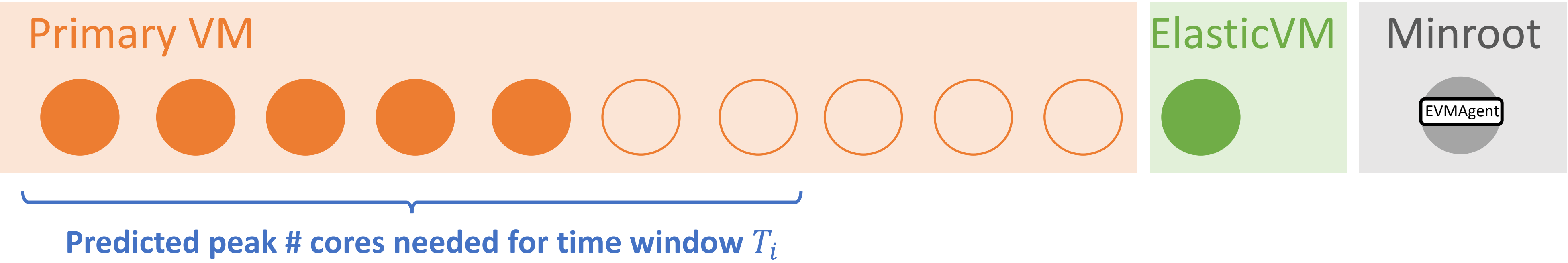
High-Level Design of SmartHarvest



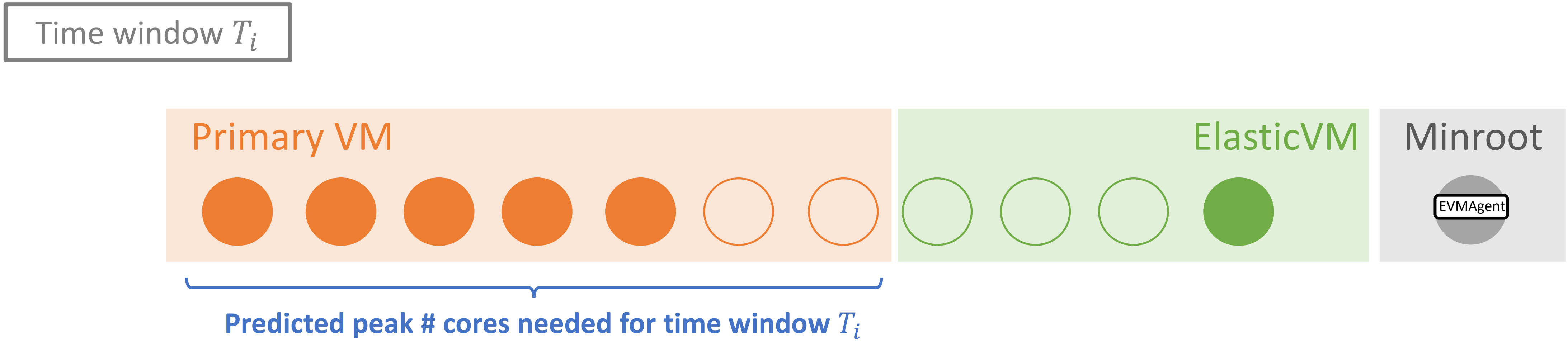
High-Level Design of SmartHarvest



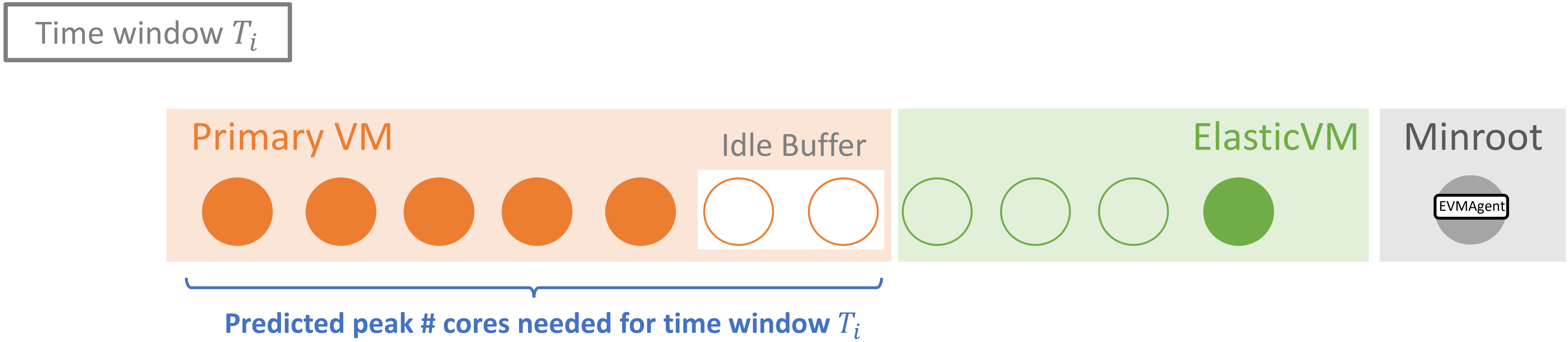
High-Level Design of SmartHarvest



High-Level Design of SmartHarvest

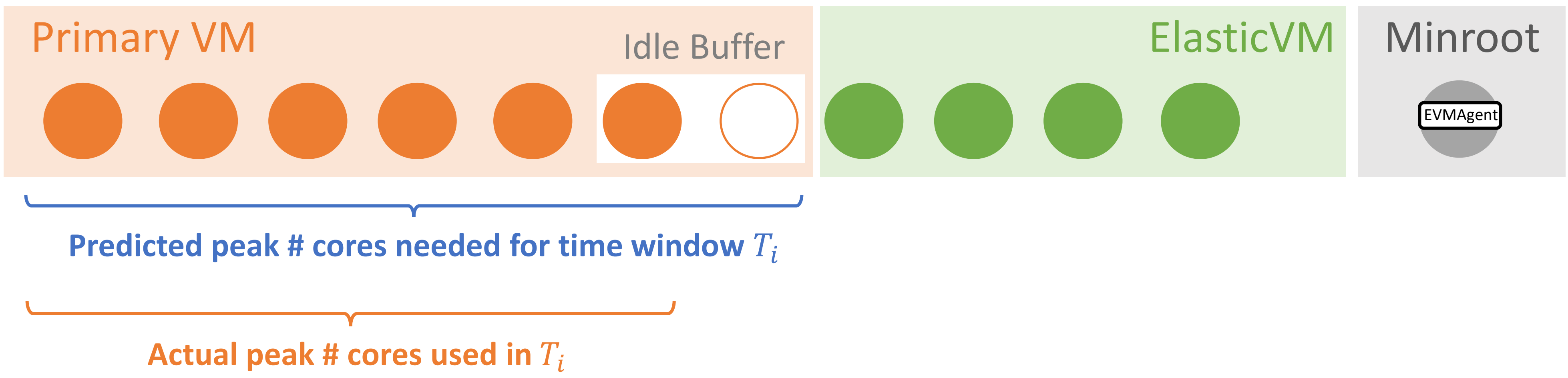


High-Level Design of SmartHarvest



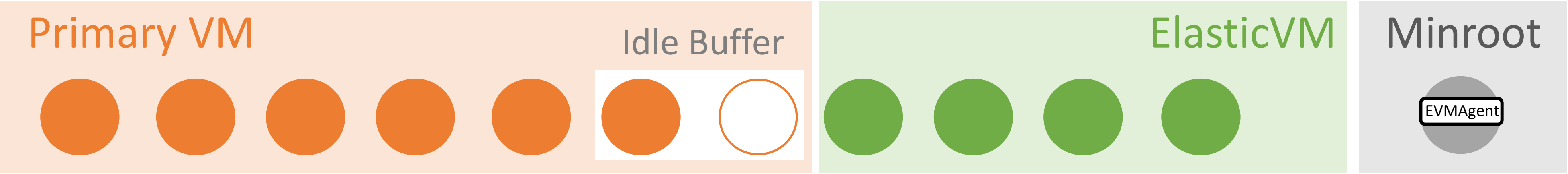
High-Level Design of SmartHarvest

End of time window T_i



High-Level Design of SmartHarvest

End of time window T_i

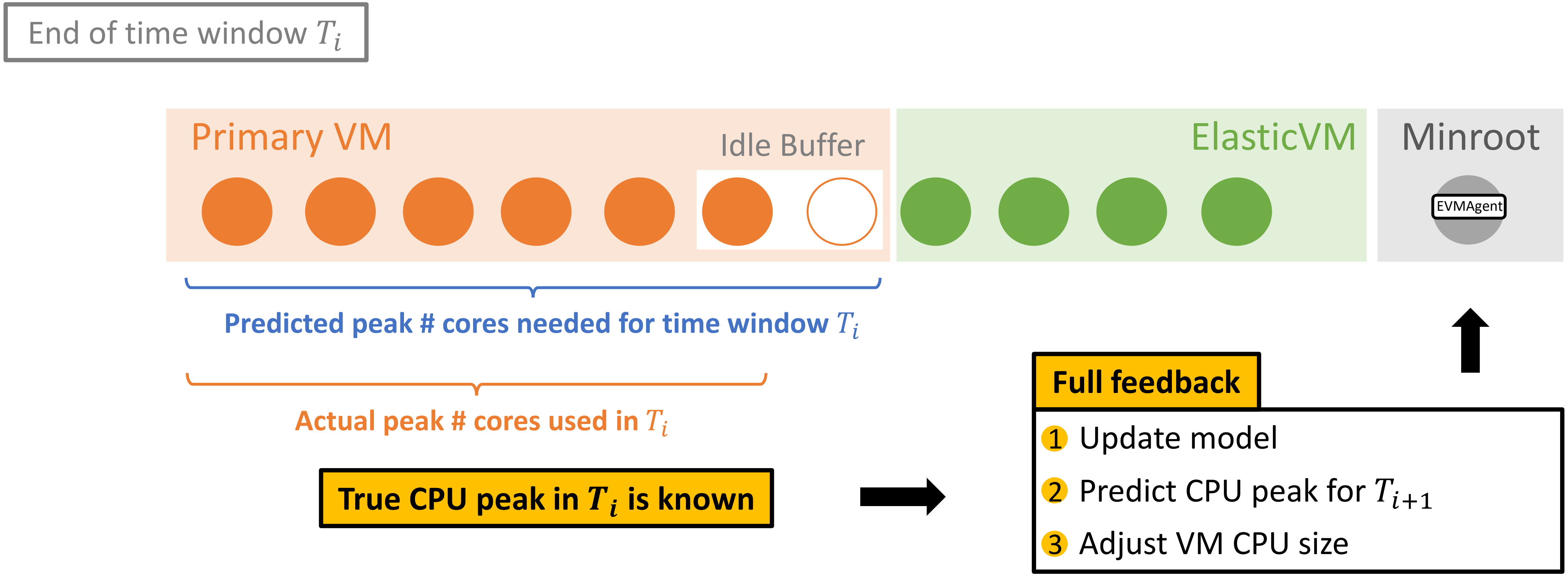


Predicted peak # cores needed for time window T_i

Actual peak # cores used in T_i

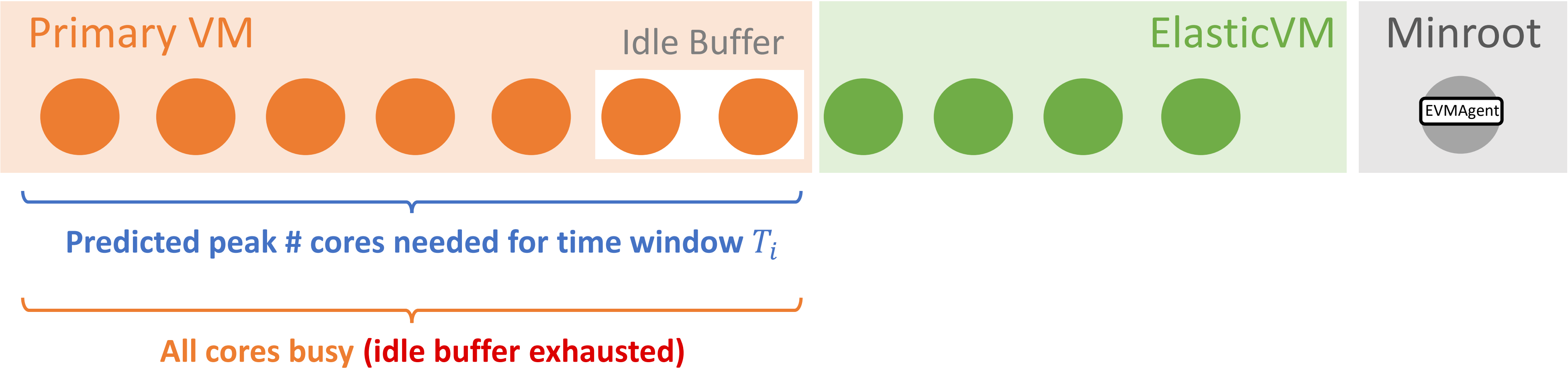
True CPU peak in T_i is known

High-Level Design of SmartHarvest

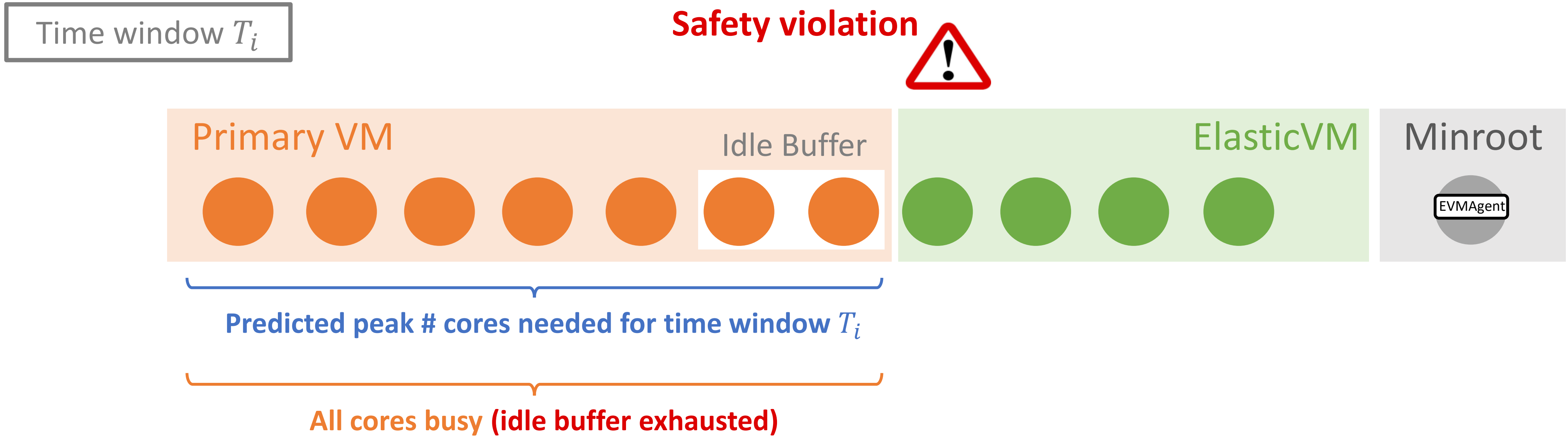


High-Level Design of SmartHarvest

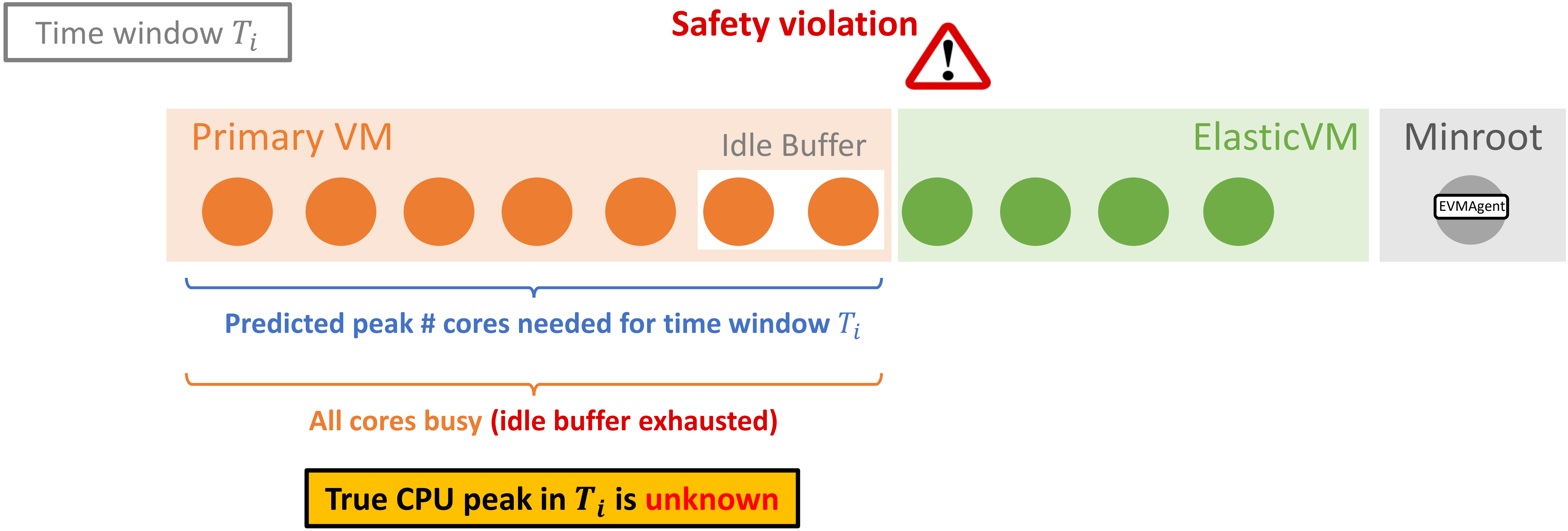
Time window T_i



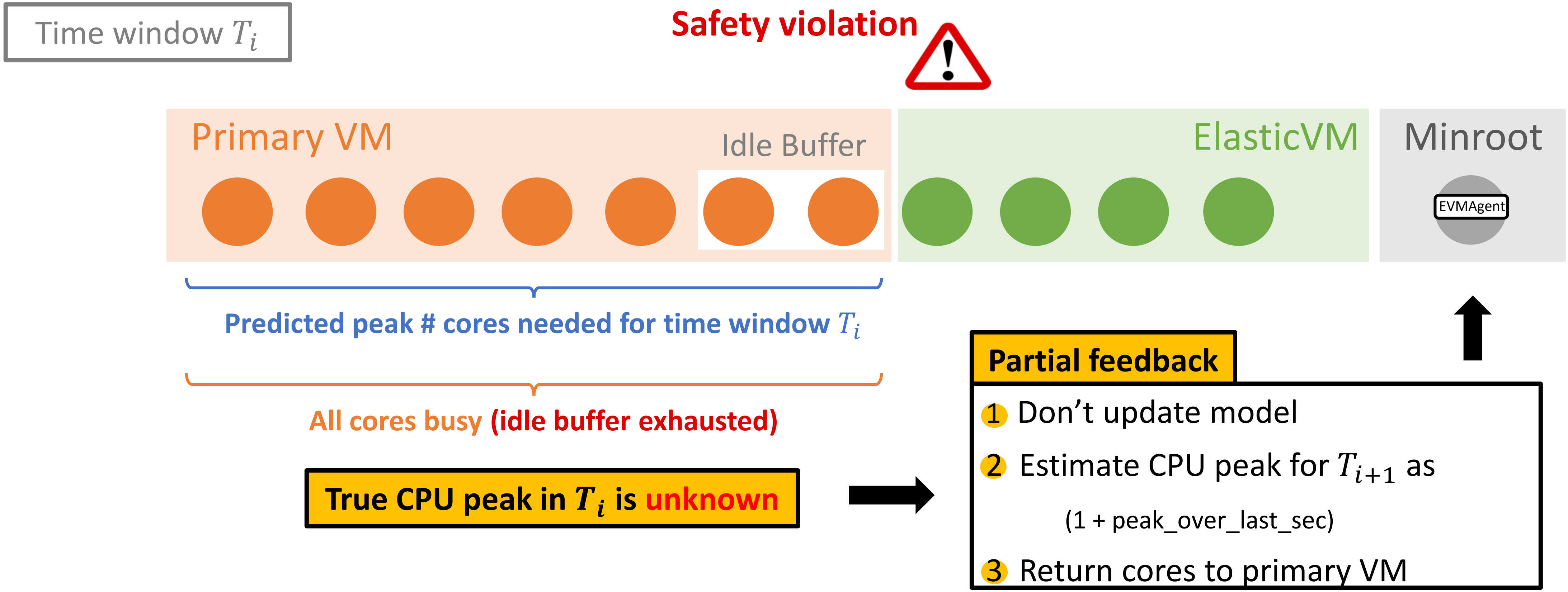
High-Level Design of SmartHarvest



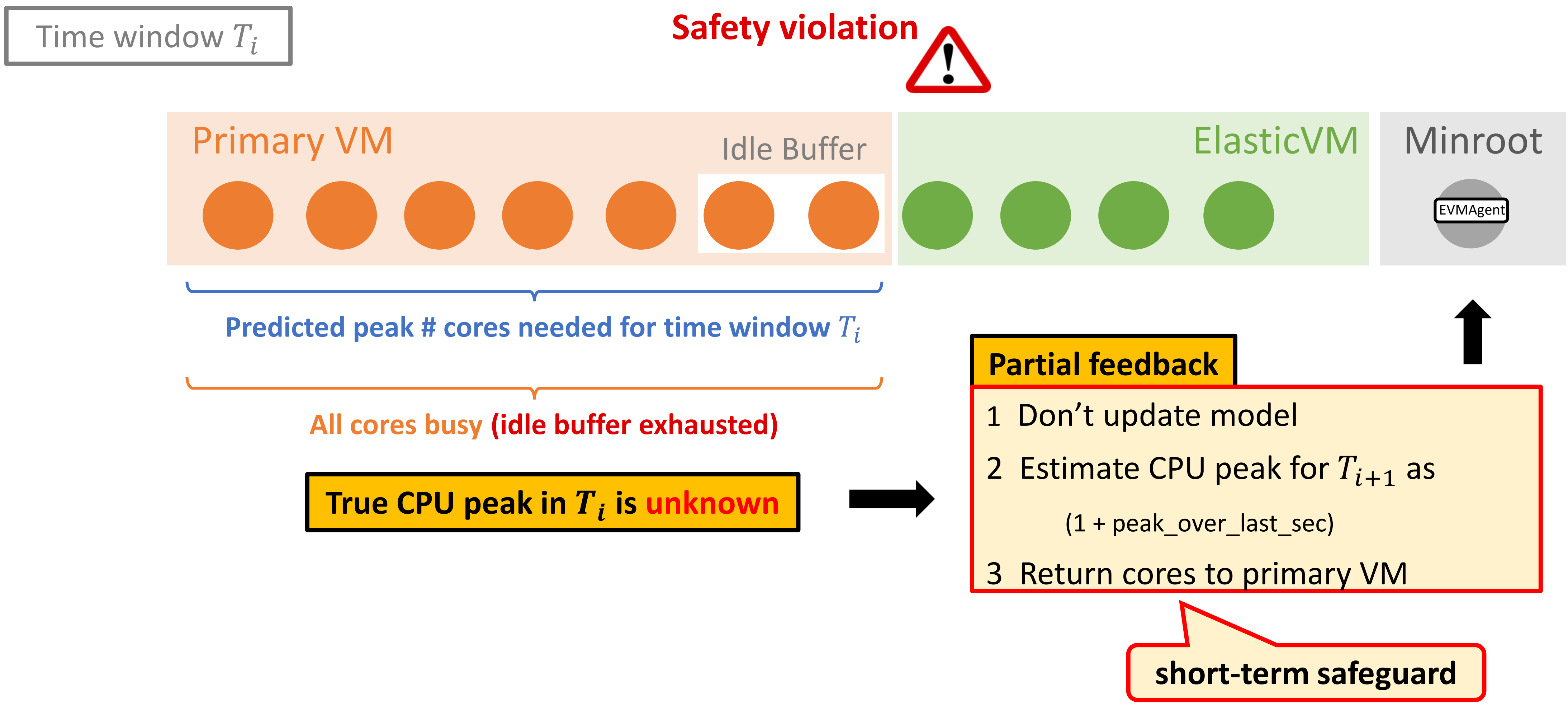
High-Level Design of SmartHarvest



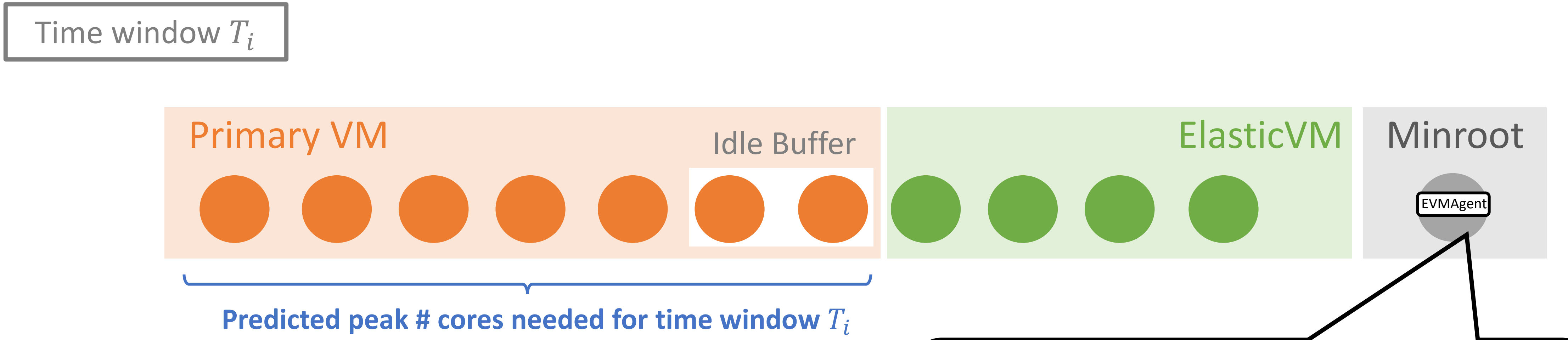
High-Level Design of SmartHarvest



High-Level Design of SmartHarvest



High-Level Design of SmartHarvest

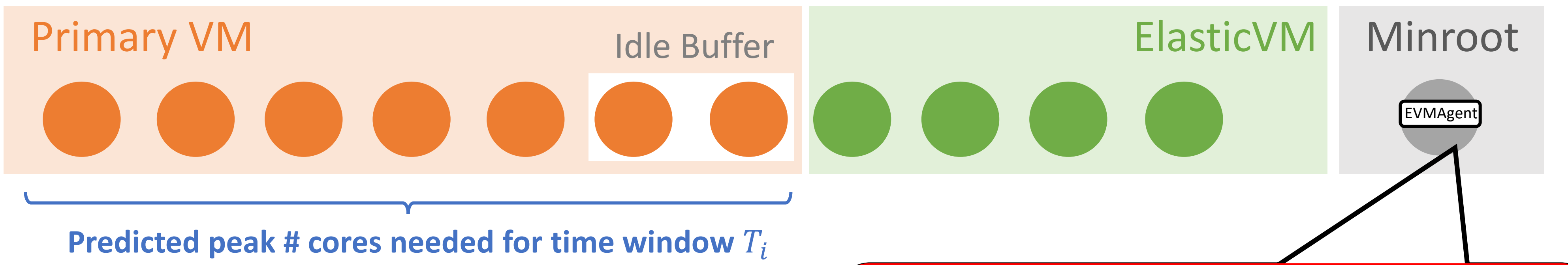


EVMAgent

- Disable harvesting when primary VM experiences long vCPU wait time → proxy for VM performance
  - e.g., more than 1% of vCPU wait times > 50 μs

High-Level Design of SmartHarvest

Time window T_i



EVMAgent

- Disable harvesting when primary VM experiences long vCPU wait time
  - e.g., more than 1% of vCPU wait times > 50 μs

Long-term safeguard

Predicting CPU Utilization

Predicting CPU Utilization

Challenge: VM CPU utilizations are constantly changing

Predicting CPU Utilization

Challenge: VM CPU utilizations are constantly changing

- Online learning to continuously learn without offline training

Predicting CPU Utilization

Challenge: VM CPU utilizations are constantly changing

- Online learning to continuously learn without offline training
- Fine-grained prediction (25ms) to quickly adapt to changes

Predicting CPU Utilization

Challenge: VM CPU utilizations are constantly changing

- Online learning to continuously learn without offline training
- Fine-grained prediction (25ms) to quickly adapt to changes

Prediction target

- peak CPU utilization (# cores) of the primary VM for the next 25ms

Predicting CPU Utilization

Challenge: VM CPU utilizations are constantly changing

- Online learning to continuously learn without offline training
- Fine-grained prediction (25ms) to quickly adapt to changes

Prediction target

- peak CPU utilization (# cores) of the primary VM for the next 25ms

Features

- {avg, stddev, min, max, median} of # CPU cores (actively monitored) used by the primary VM in the past 25ms

Predicting CPU Utilization

Challenge: VM CPU utilizations are constantly changing

- Online learning to continuously learn without offline training
- Fine-grained prediction (25ms) to quickly adapt to changes

Prediction target

- peak CPU utilization (# cores) of the primary VM for the next 25ms

Features

Most useful set of features

- {avg, stddev, min, max, median} of # CPU cores (actively monitored) used by the primary VM in the past 25ms

Model for Online Learning

Requirements

Model for Online Learning

Requirements

1. Simple and lightweight ~~neural network~~

Model for Online Learning

Requirements

1. Simple and lightweight ~~neural network~~
2. Differentiating under-/over-predictions

Model for Online Learning

Requirements

1. Simple and lightweight ~~neural network~~
2. Differentiating under-/over-predictions

→ Cost-sensitive multi-class classification

Model for Online Learning

Requirements

1. Simple and lightweight ~~neural network~~
2. Differentiating under-/over-predictions

➔ **Cost-sensitive multi-class classification**

- Trains a separate linear regression model for each class

Model for Online Learning

Requirements

1. Simple and lightweight ~~neural network~~
2. Differentiating under-/over-predictions

→ **Cost-sensitive multi-class classification**

- Trains a separate linear regression model for each class
- Selects the class with the lowest predicted cost

Model for Online Learning

Requirements

1. Simple and lightweight ~~neural network~~
2. Differentiating under-/over-predictions

→ **Cost-sensitive multi-class classification**

- Trains a separate linear regression model for each class
- Selects the class with the lowest predicted cost

☆ Allows flexible cost assignment to update model

Model for Online Learning

Requirements

1. Simple and lightweight ~~neural network~~
2. Differentiating under-/over-predictions

→ **Cost-sensitive multi-class classification**

- Trains a separate linear regression model for each class
- Selects the class with the lowest predicted cost

☆ Allows flexible cost assignment to update model

☆ Offers fast prediction and update times (e.g. <15μs)

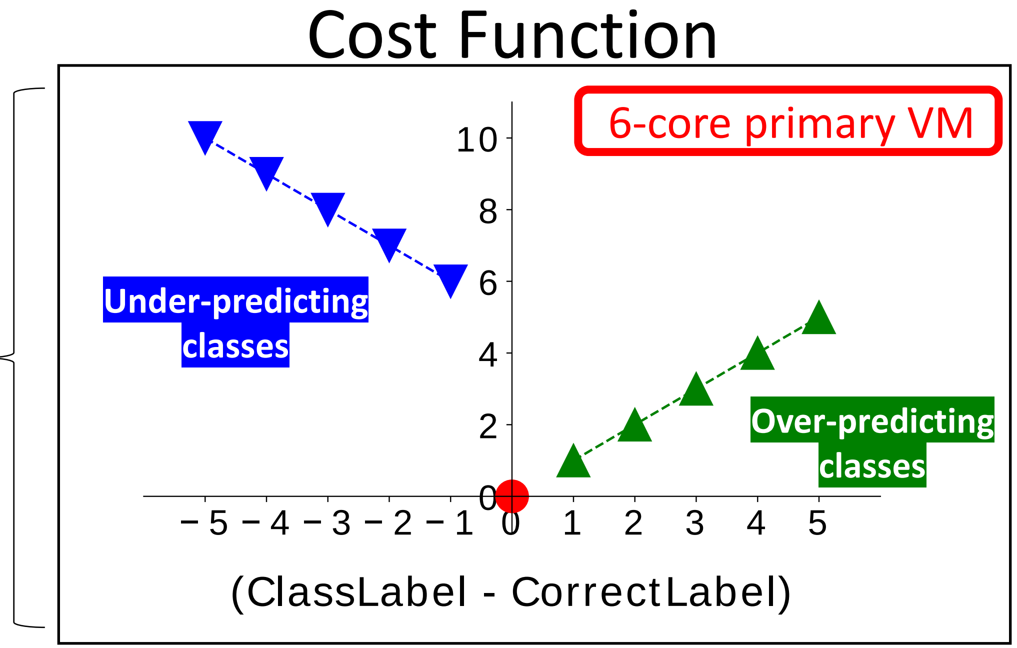
Full-feedback model update

```
if pred_peak > observed_peak:
        true_peak = observed_peak
```

Full-feedback model update

```
if pred_peak > observed_peak:
      true_peak = observed_peak
```

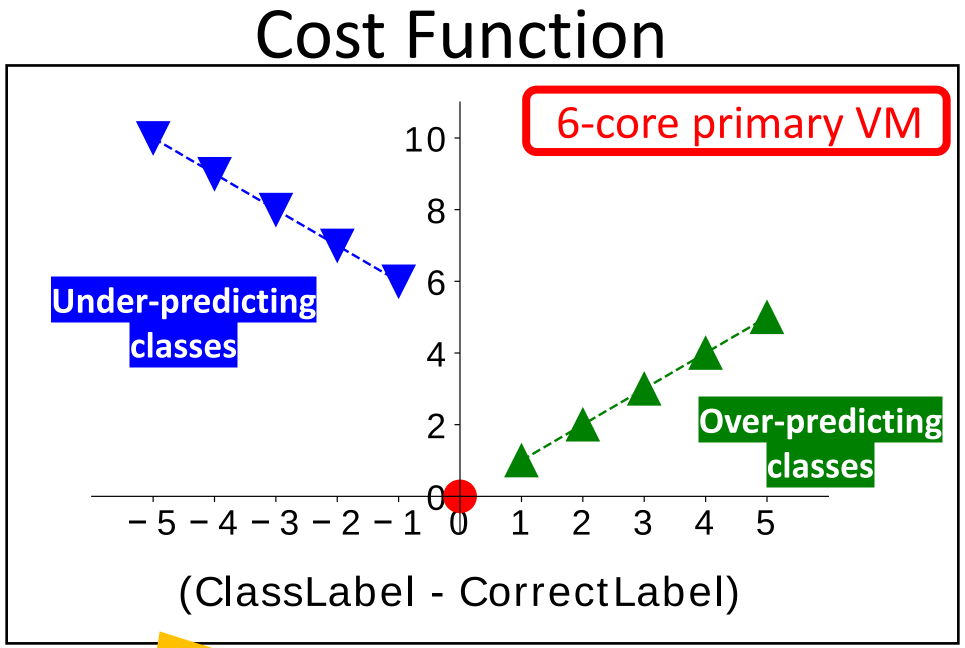
Update model with a cost for each class



Full-feedback model update

```
if pred_peak > observed_peak:
      true_peak = observed_peak
```

Update model with a cost for each class



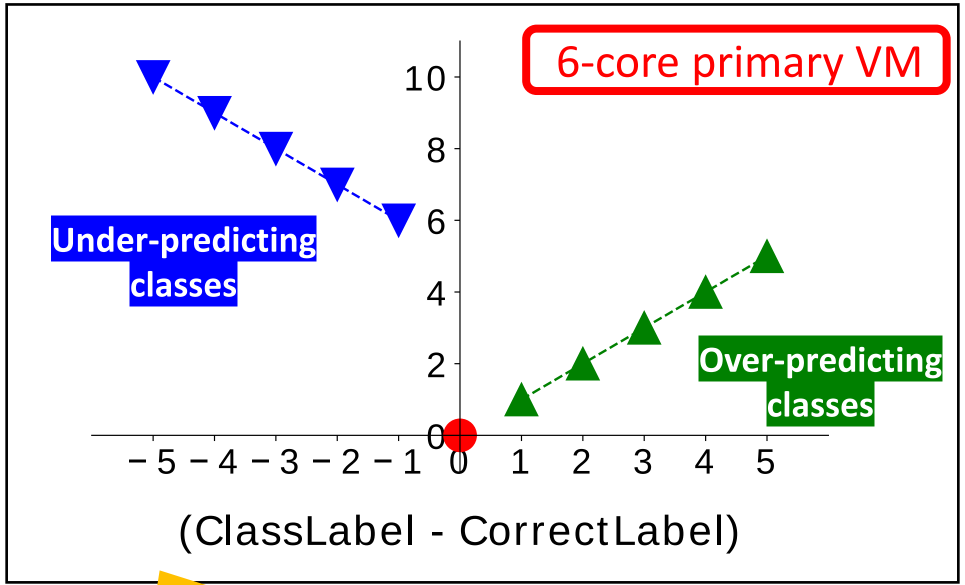
Penalize under-predicting classes more to skew away from aggressive harvesting

Full-feedback model update

```
if pred_peak > observed_peak:
      true_peak = observed_peak
```

Cost Function

Update model with a cost for each class



Penalize under-predicting classes more to skew away from aggressive harvesting

E.g. **true_peak = 3 (correct class label)**

- Class 1: cost = |1-3|+5=7
- Class 2: cost = |2-3|+5=6

Classes that were under-predicting

- Class 3: cost = |3-3|=0 → Correct class
- Class 4: cost = |4-3|=1
- Class 5: cost = |5-3|=2
- Class 6: cost = |6-3|=3

Classes that were over-predicting

Evaluation

- **Primary VM workloads**
  - Microsoft Bing IndexServe
  - Memcached: in-memory key-value store
  - moses: machine translation application
  - img-dnn: image recognition application

P99 Latency

Evaluation

- **Primary VM workloads**
  - Microsoft Bing IndexServe
  - Memcached: in-memory key-value store
  - moses: machine translation application
  - img-dnn: image recognition application

  } P99 Latency
- **ElasitcVM workloads**
  - CPUBully (synthetic CPU-bound workload) } Avg. # of cores harvested

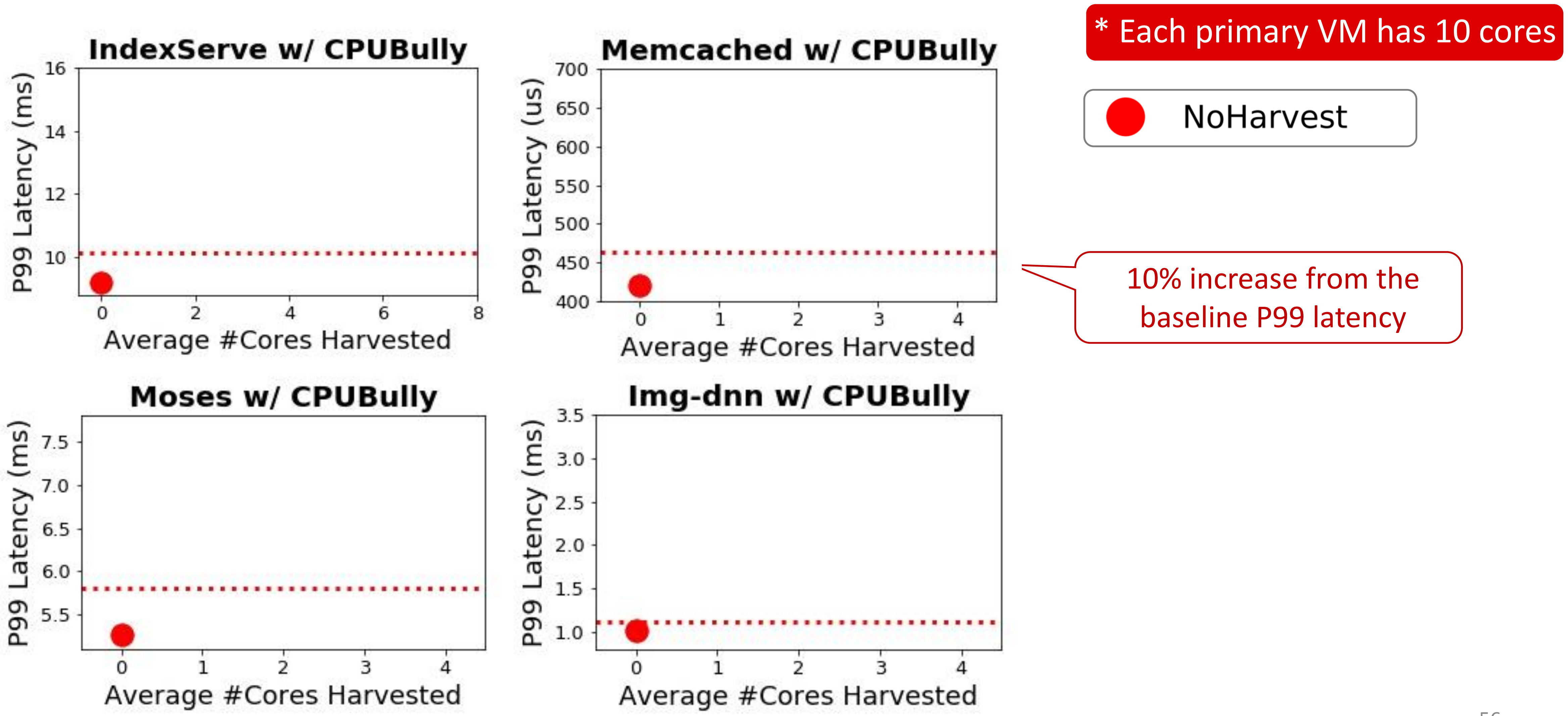
Evaluation (cont’d)

- **Alternative policies**
  - **FixedBuffer policy**
    - Adjusts primary CPU size to maintain a fixed buffer of idle cores
  - **PrevPeak policy**
    - Estimates primary CPU peak usage based on the peak from last 25ms

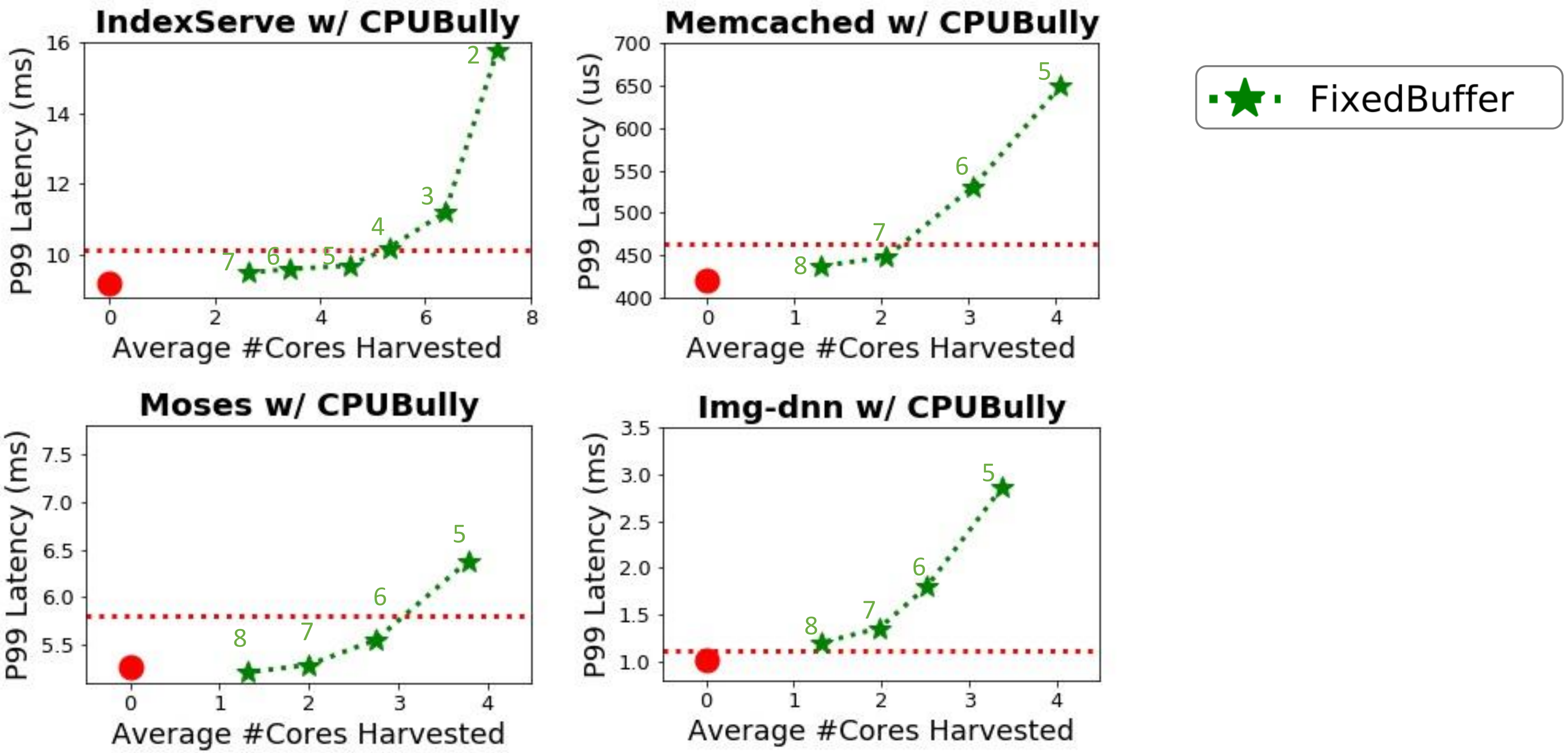
Evaluation (cont'd)

- **Alternative policies**
  - **FixedBuffer policy**
    - Adjusts primary CPU size to maintain a fixed buffer of idle cores
  - **PrevPeak policy**
    - Estimates primary CPU peak usage based on the peak from last 25ms
- **Testbed**
  - Two-socket Intel server with Xeon Platinum 8160 processor
  - 2.10GHz, 24 cores per socket, 255GB DRAM
  - Running the Hyper-V hypervisor

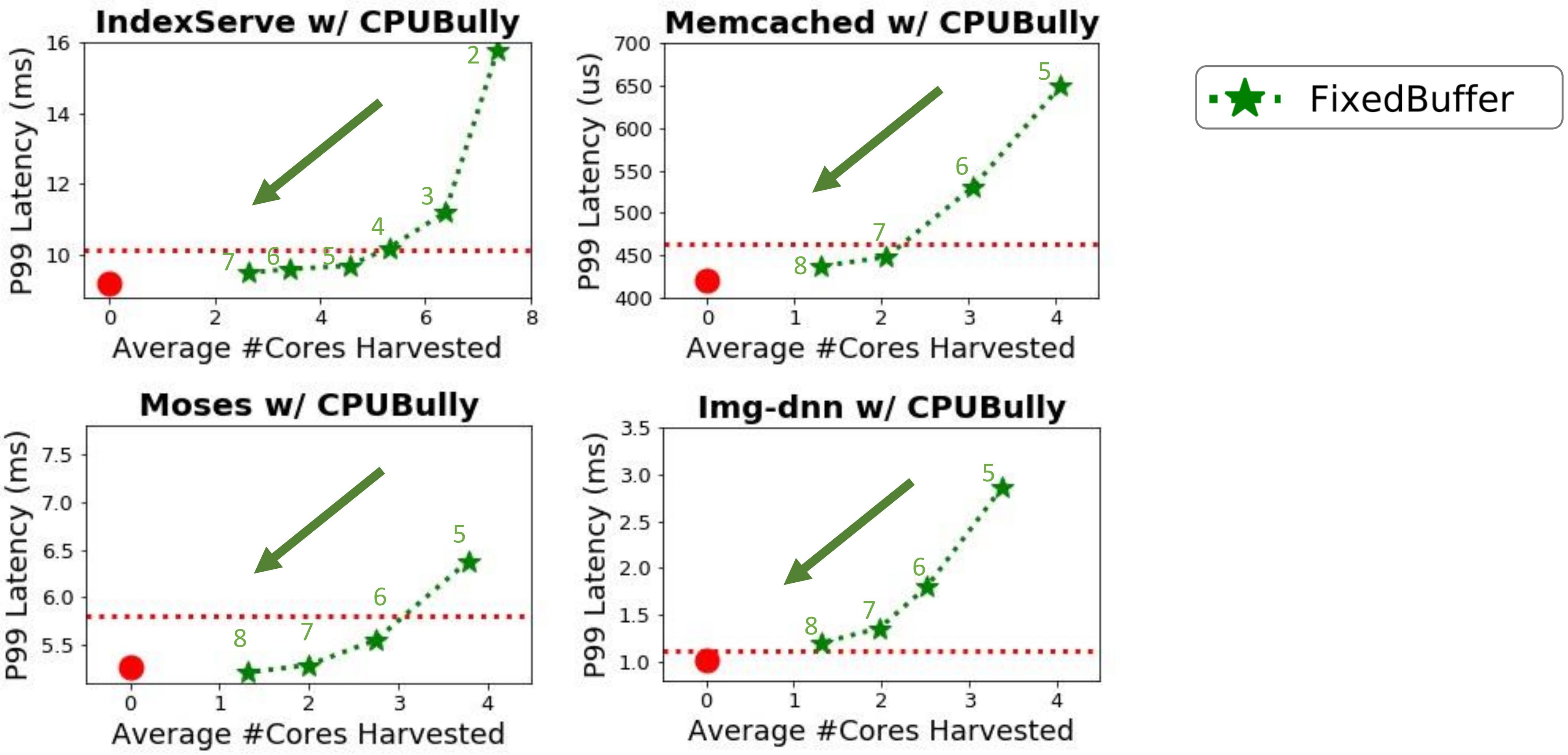
Single primary VM co-located with CPUBully



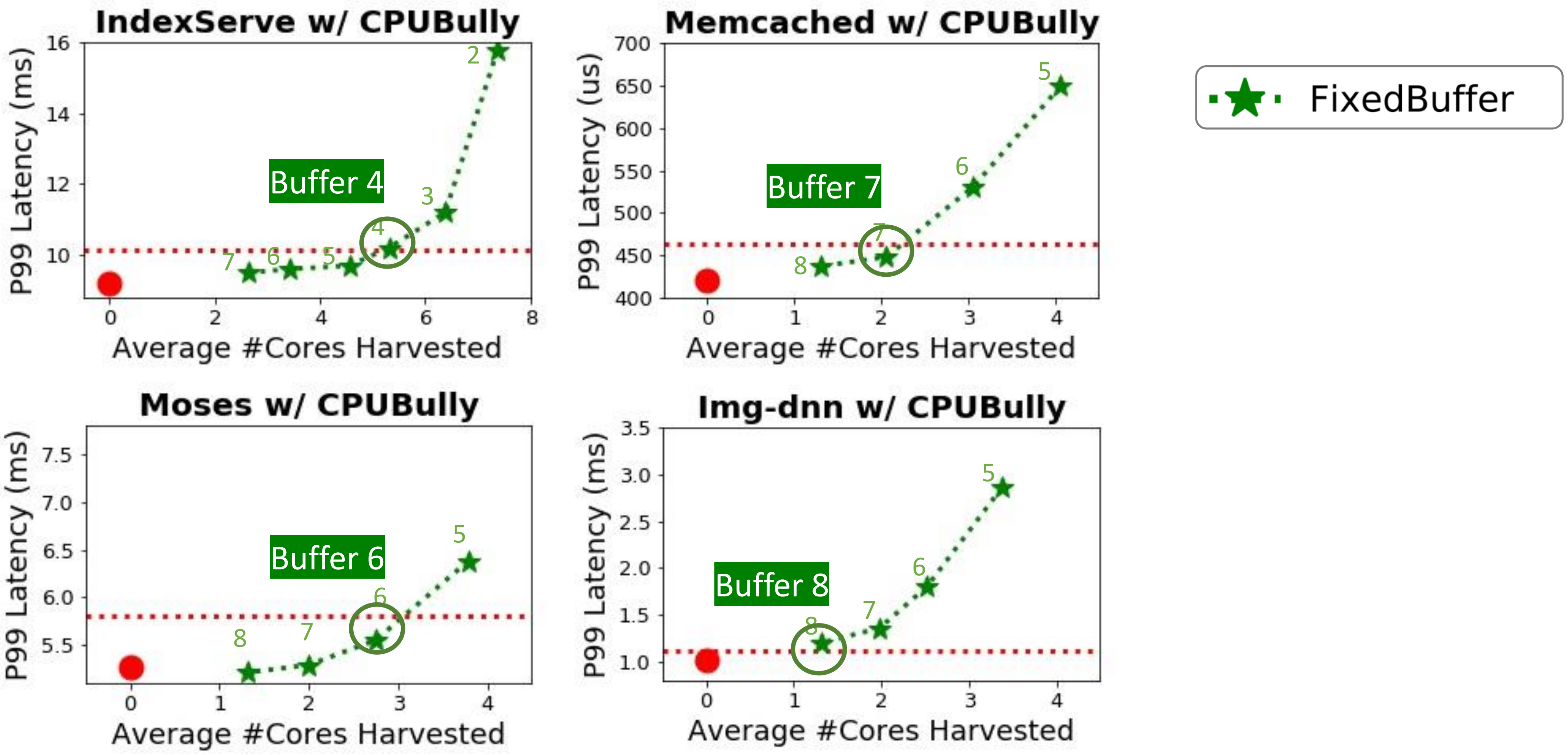
Single primary VM co-located with CPUBully



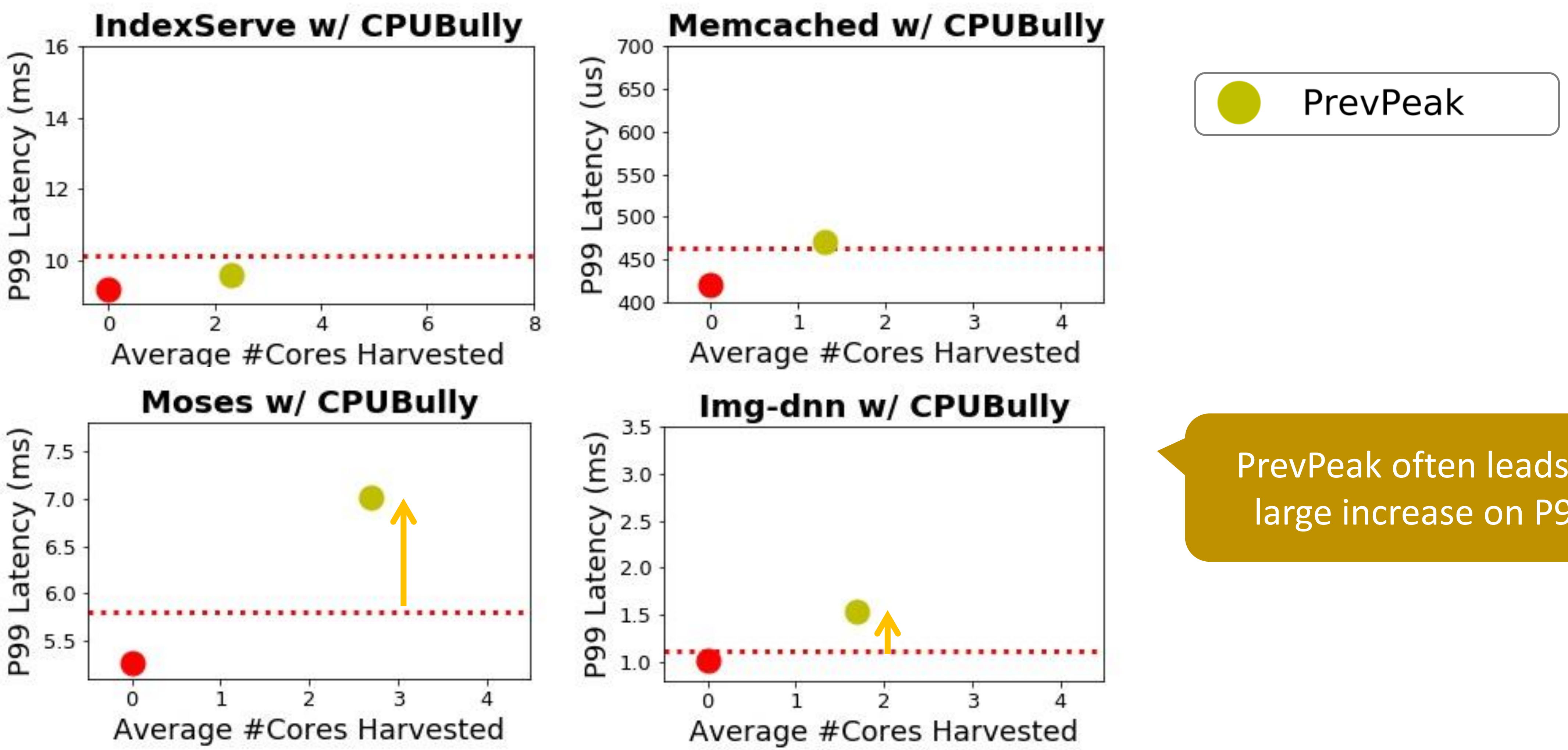
Single primary VM co-located with CPUBully



Single primary VM co-located with CPUBully

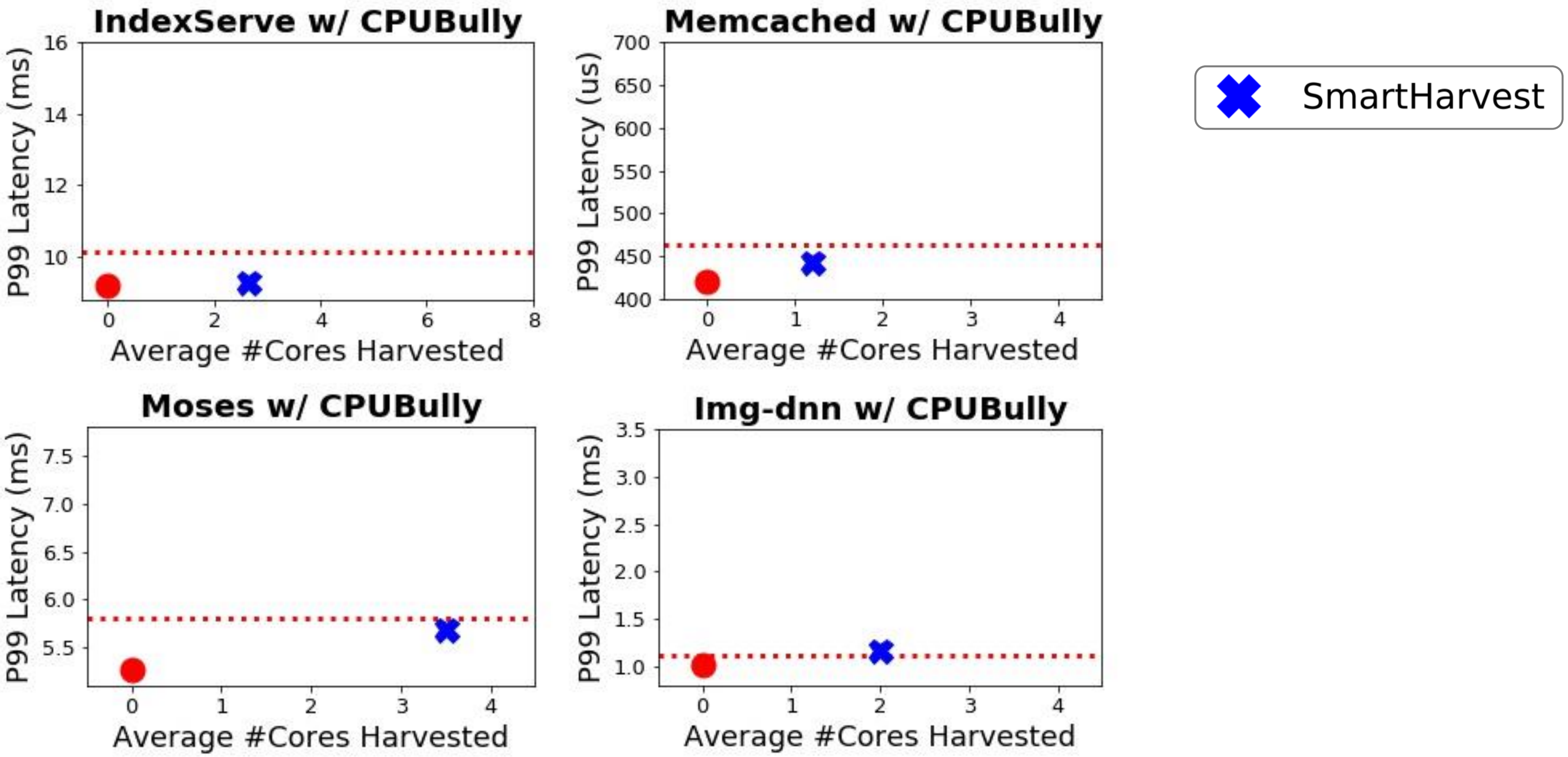


Single primary VM co-located with CPUBully

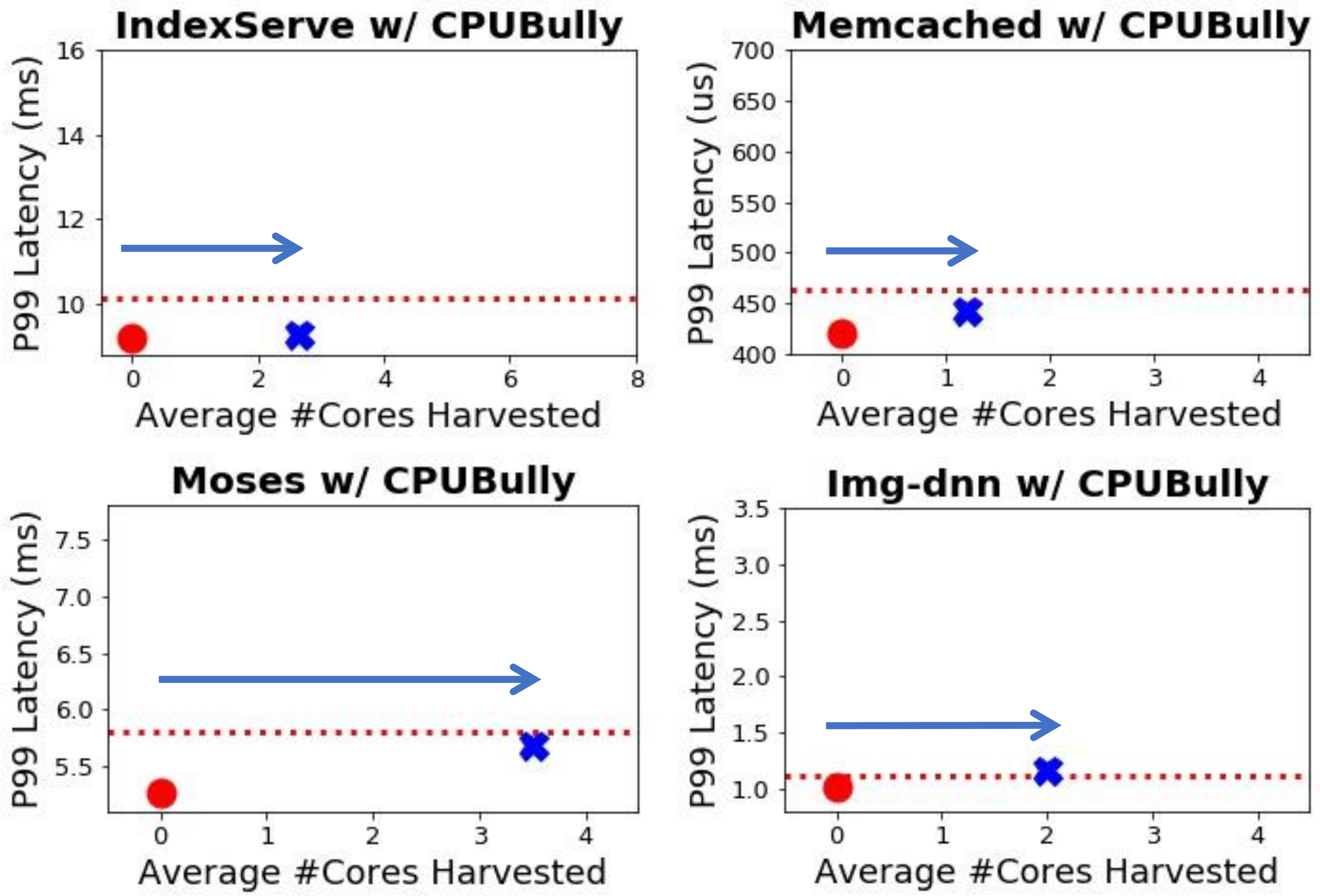


PrevPeak often leads to large increase on P99

Single primary VM co-located with CPUBully

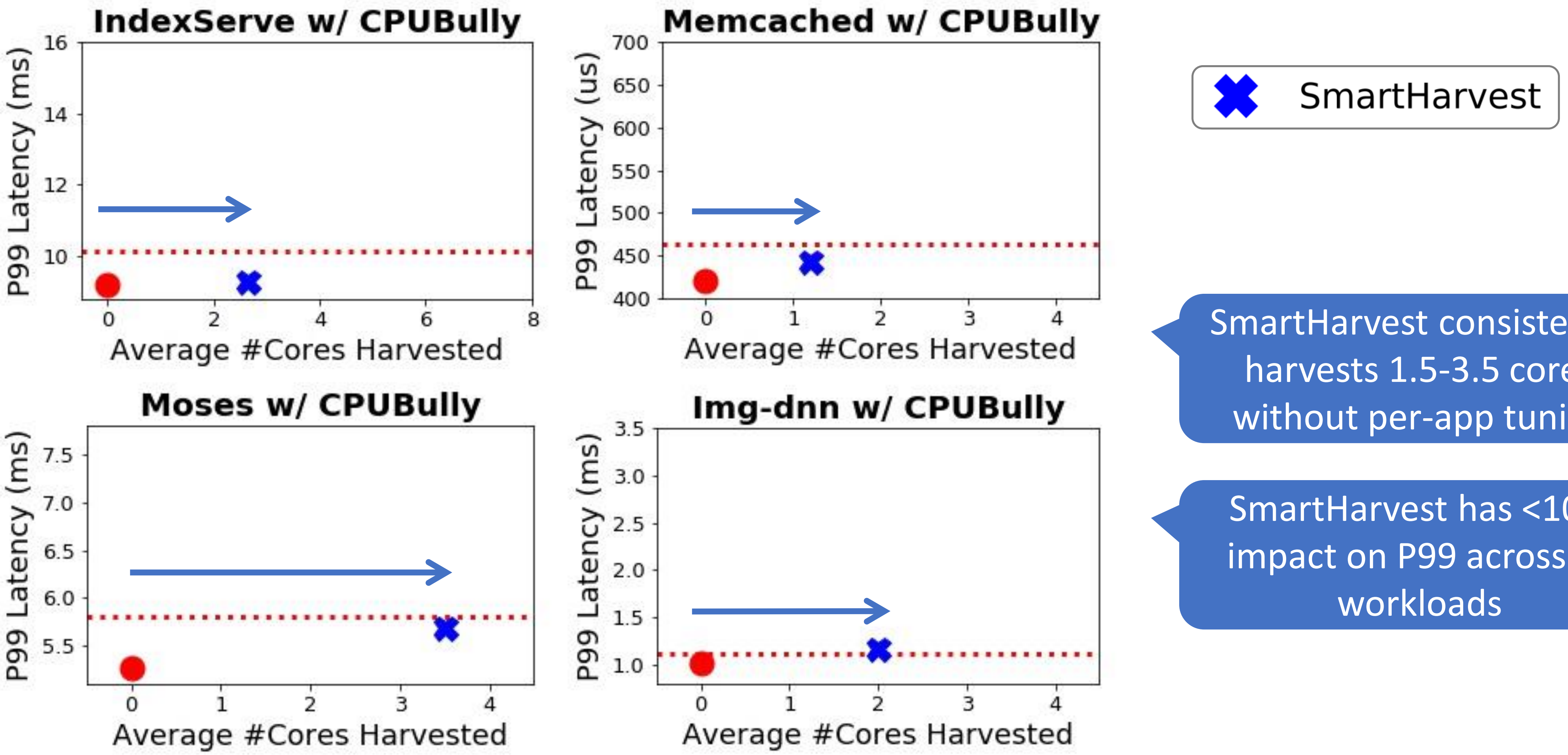


Single primary VM co-located with CPUBully



SmartHarvest consistently harvests 1.5-3.5 cores without per-app tuning

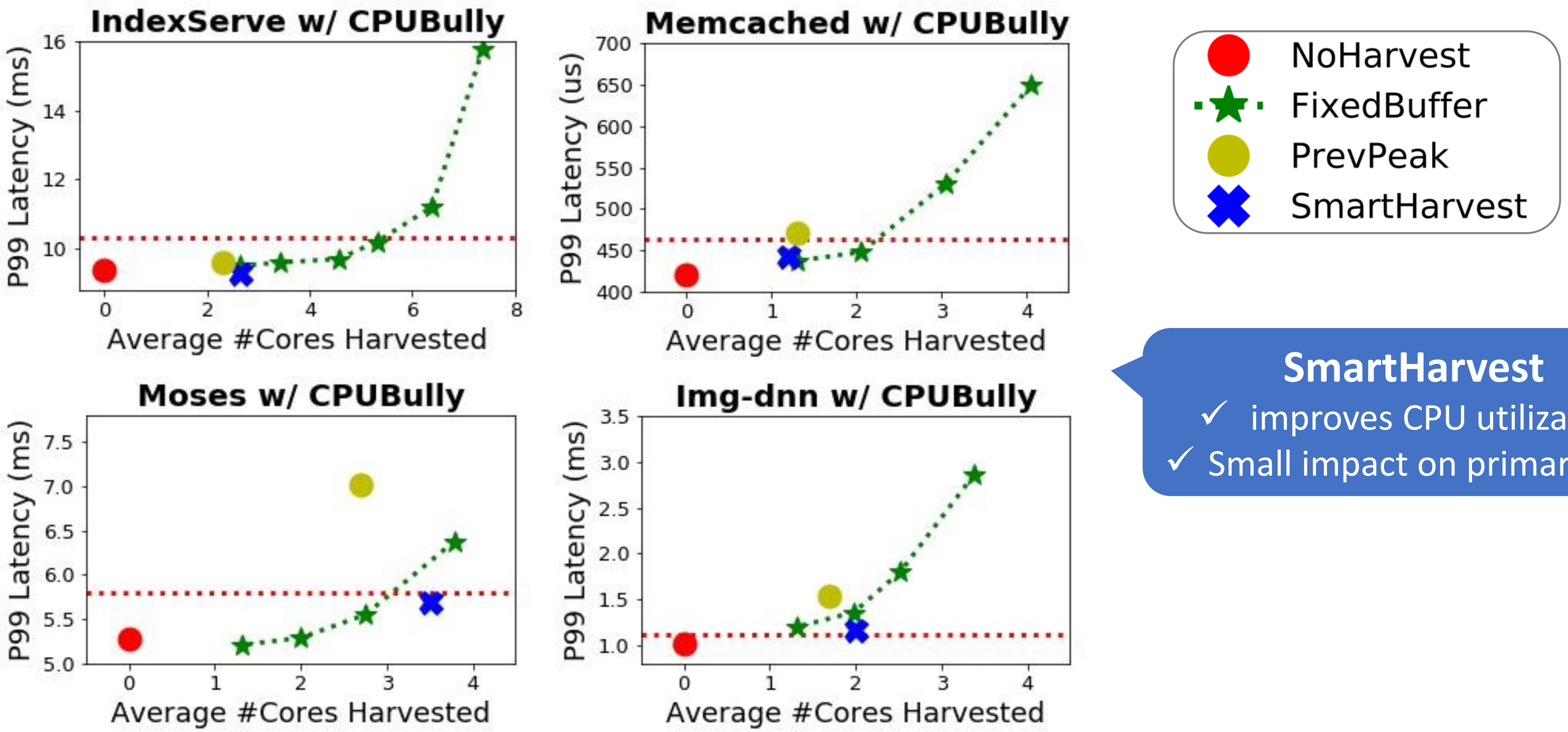
Single primary VM co-located with CPUBully



SmartHarvest consistently harvests 1.5-3.5 cores without per-app tuning

SmartHarvest has <10% impact on P99 across all workloads

Single primary VM co-located with CPUBully



SmartHarvest

✓ improves CPU utilization

✓ Small impact on primary VM

More evaluation results in the paper

- Running realistic batch workloads in ElasticVM
- Harvesting from multiple primary VMs
- Learning window selection
- Cost function comparison
- Effectiveness of safeguards
- System responsiveness vs benefit of learning

Conclusion

- **SmartHarvest** leverages online learning to manage cloud CPU resources

Conclusion

- **SmartHarvest** leverages online learning to manage cloud CPU resources
  - ➢ Automatically learns and adapts to CPU usage of different applications and varying load patterns inside opaque-box primary VMs

Conclusion

- **SmartHarvest** leverages online learning to manage cloud CPU resources
  - ➢ Automatically learns and adapts to CPU usage of different applications and varying load patterns inside opaque-box primary VMs
  - ➢ Harvests spare CPU cores while minimizing performance impacts on primary VMs

Conclusion

- **SmartHarvest** leverages online learning to manage cloud CPU resources
  - ➢ Automatically learns and adapts to CPU usage of different applications and varying load patterns inside opaque-box primary VMs
  - ➢ Harvests spare CPU cores while minimizing performance impacts on primary VMs
  - ➢ Achieves improved CPU utilization in the cloud

Conclusion

- **SmartHarvest** leverages online learning to manage cloud CPU resources
  - ➢ Automatically learns and adapts to CPU usage of different applications and varying load patterns inside opaque-box primary VMs
  - ➢ Harvests spare CPU cores while minimizing performance impacts on primary VMs
  - ➢ Achieves improved CPU utilization in the cloud

Thank you!

Contact: yawenw@stanford.edu