

# Benchmarking Dependability of Virtual and Cloud Environments

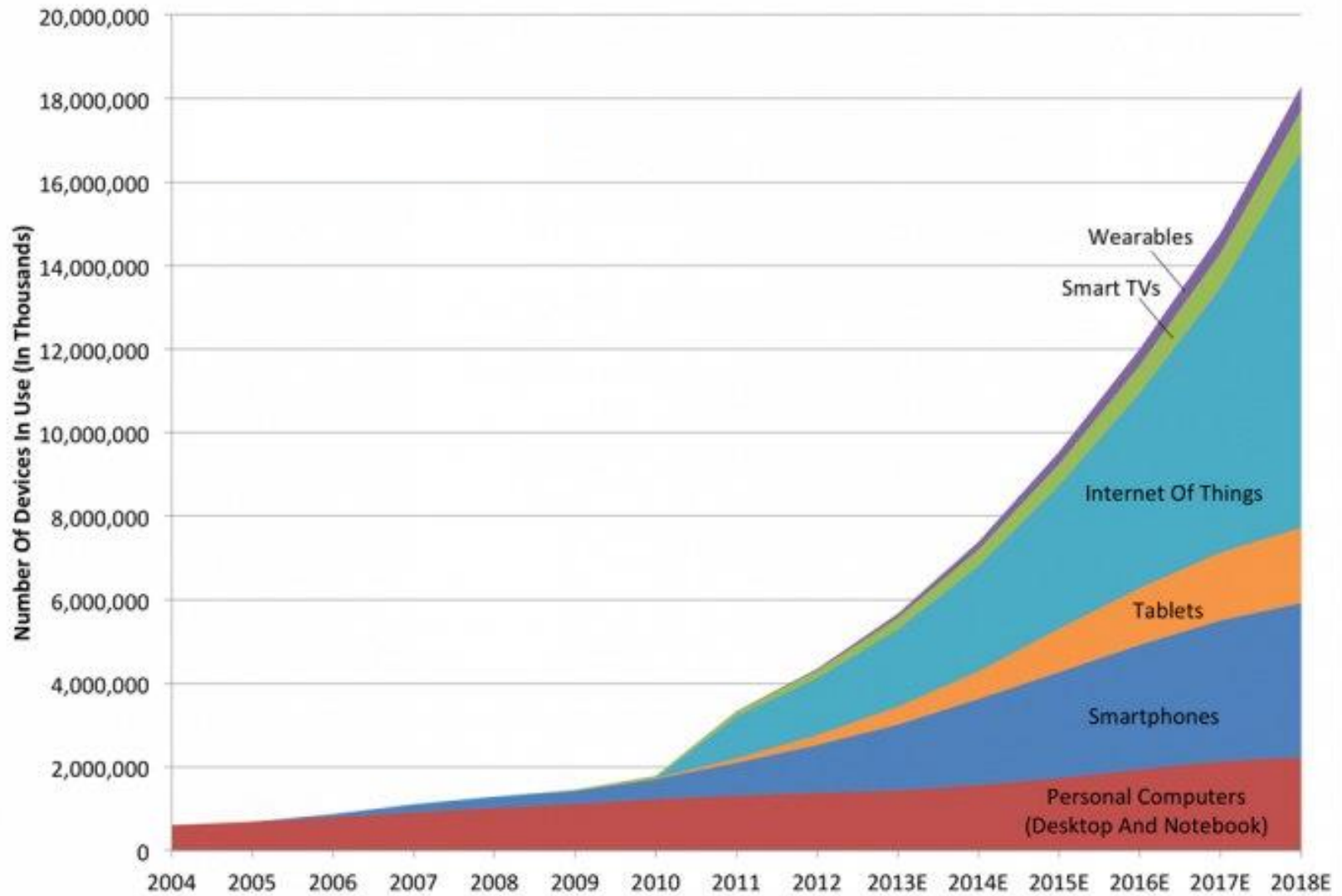
**Samuel Kounev**

Chair of Software Engineering  
University of Würzburg

<http://se.informatik.uni-wuerzburg.de/>

Charles University, Prague, May 4, 2016

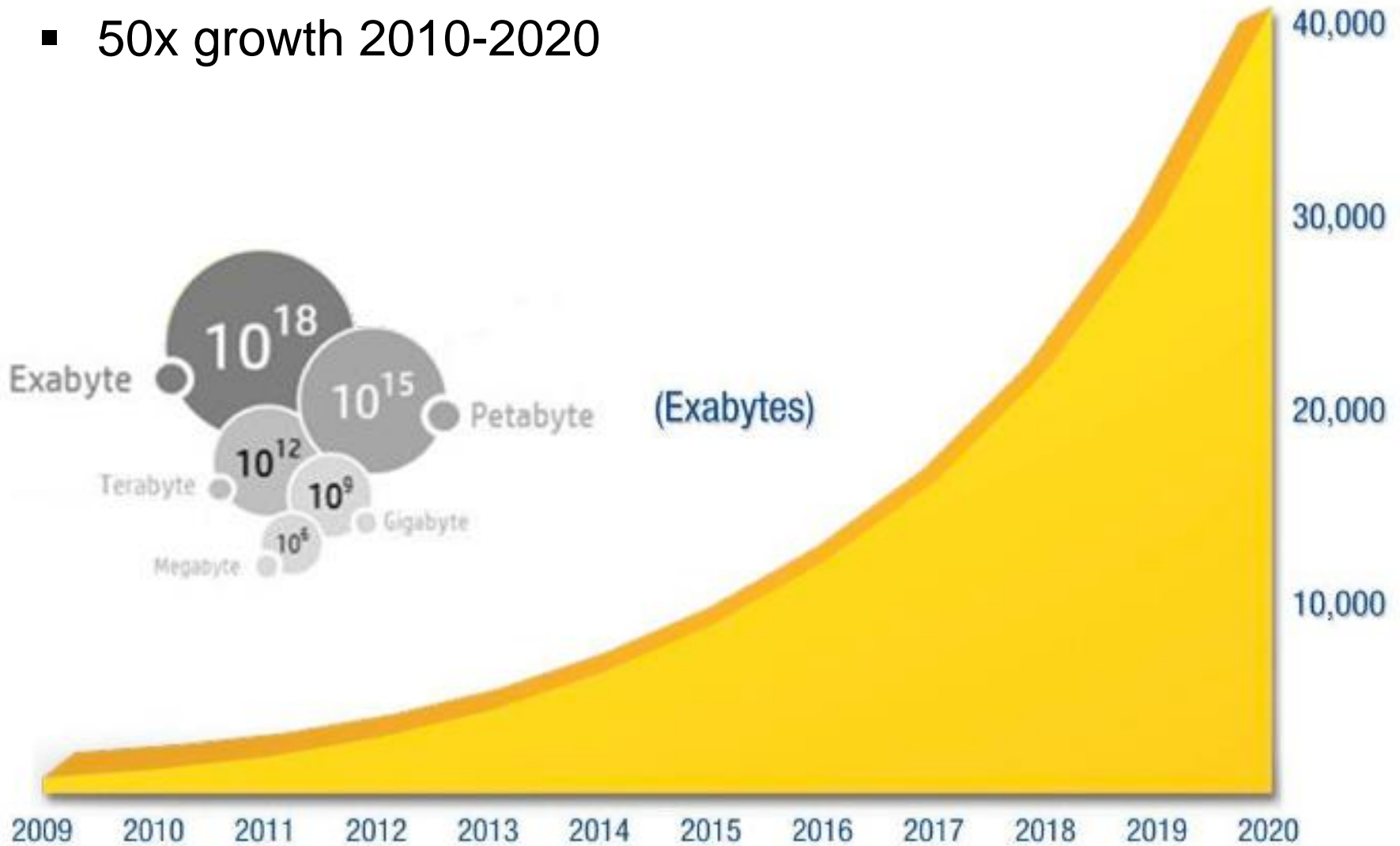
# Explosion of IT Services & Users



Source: Gartner, IDC, Strategy Analytics, Machina Research, company filings, BII estimates

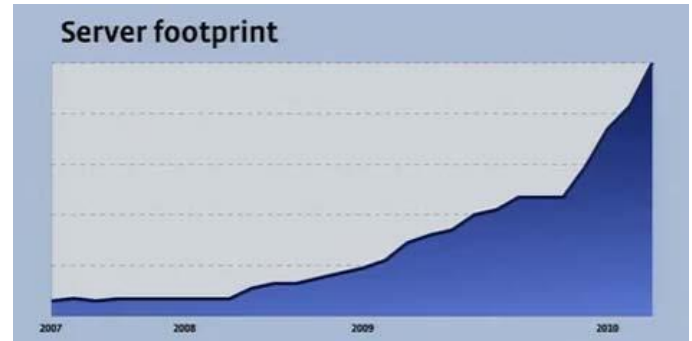
# Explosion of Data

- 50x growth 2010-2020



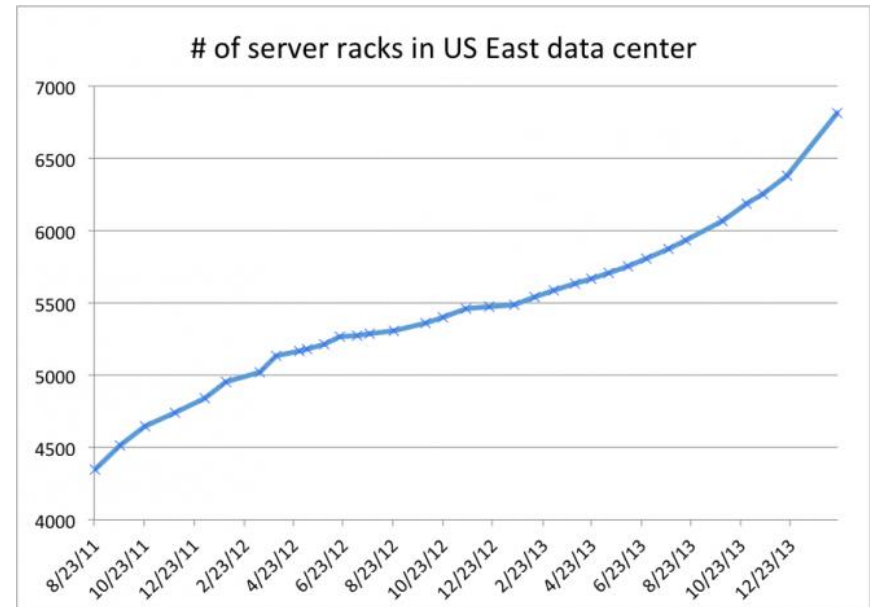
Source: IDC's Digital Universe Study, sponsored by EMC, December 2012

# Growing Number of Servers



Facebook Servers

- Google ~ 1 Mil. (2013)
- Microsoft ~ 1 Mil. (2013)
- Facebook ~ 180K (2012)
- OVH ~ 150K (2013)
- Akamai Tech. ~ 127K (2013)
- Rackspace ~ 94K (2013)
- 1&1 Internet ~ 70K (2010)
- eBay ~ 54K (2013)
- HP/EDS ~ 380K (2013)
- ...

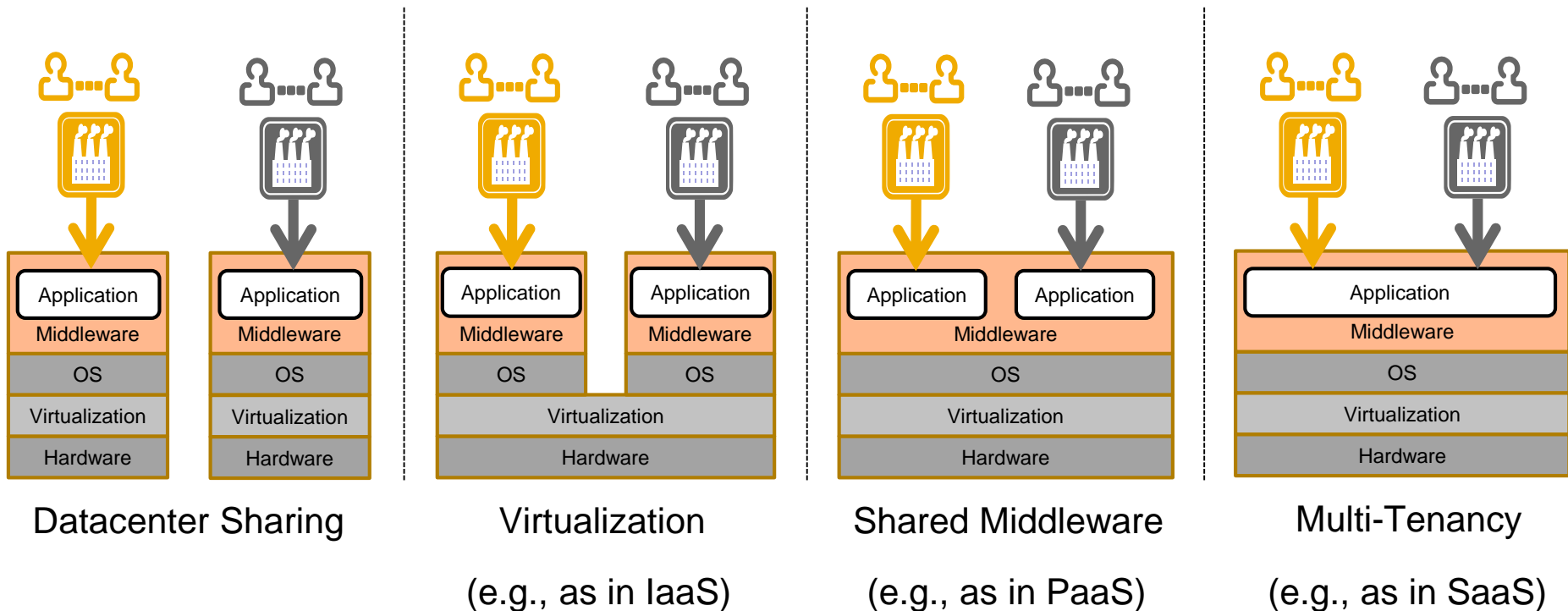


Amazon's Virginia region [Src: Wired.com]

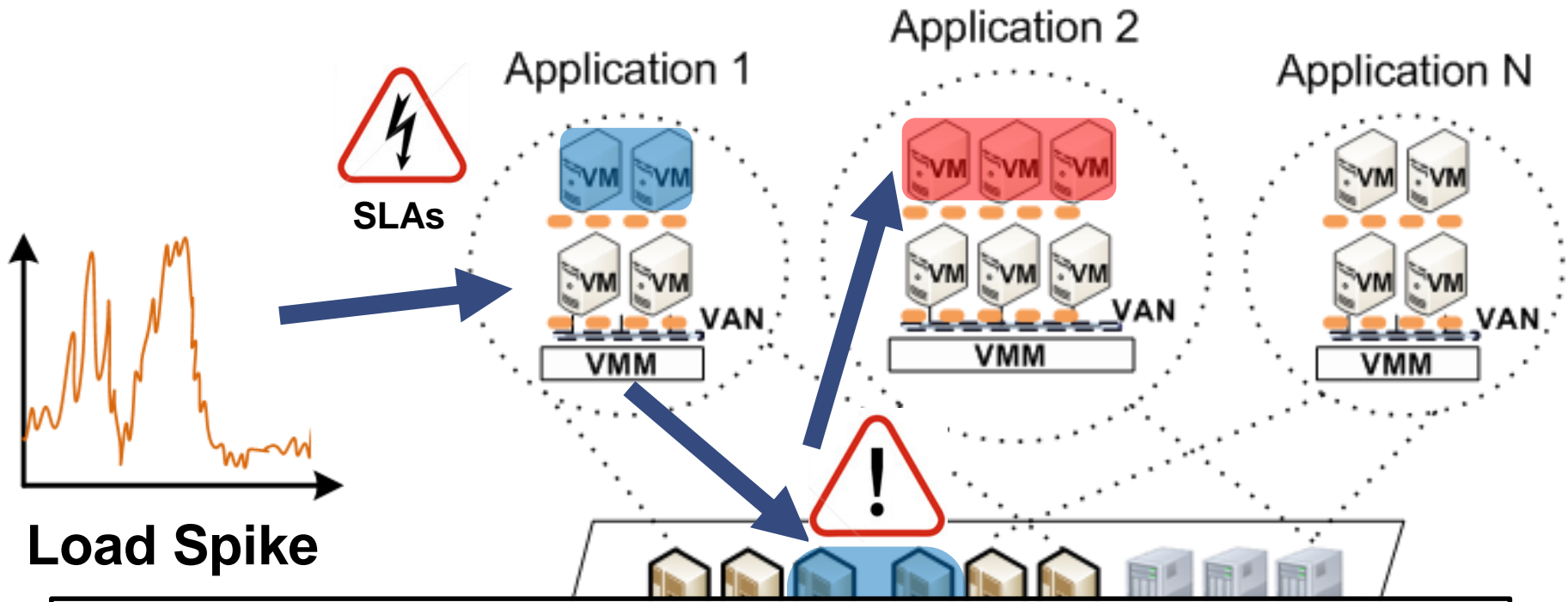
Source: <http://www.datacenterknowledge.com>

# Increasing Pressure to Raise Efficiency

- Proliferation of **shared IT infrastructures** → Cloud Computing
- Different forms of resource sharing (hardware and software)
  - Network, storage, and computing infrastructure
  - Software stacks



# Challenges

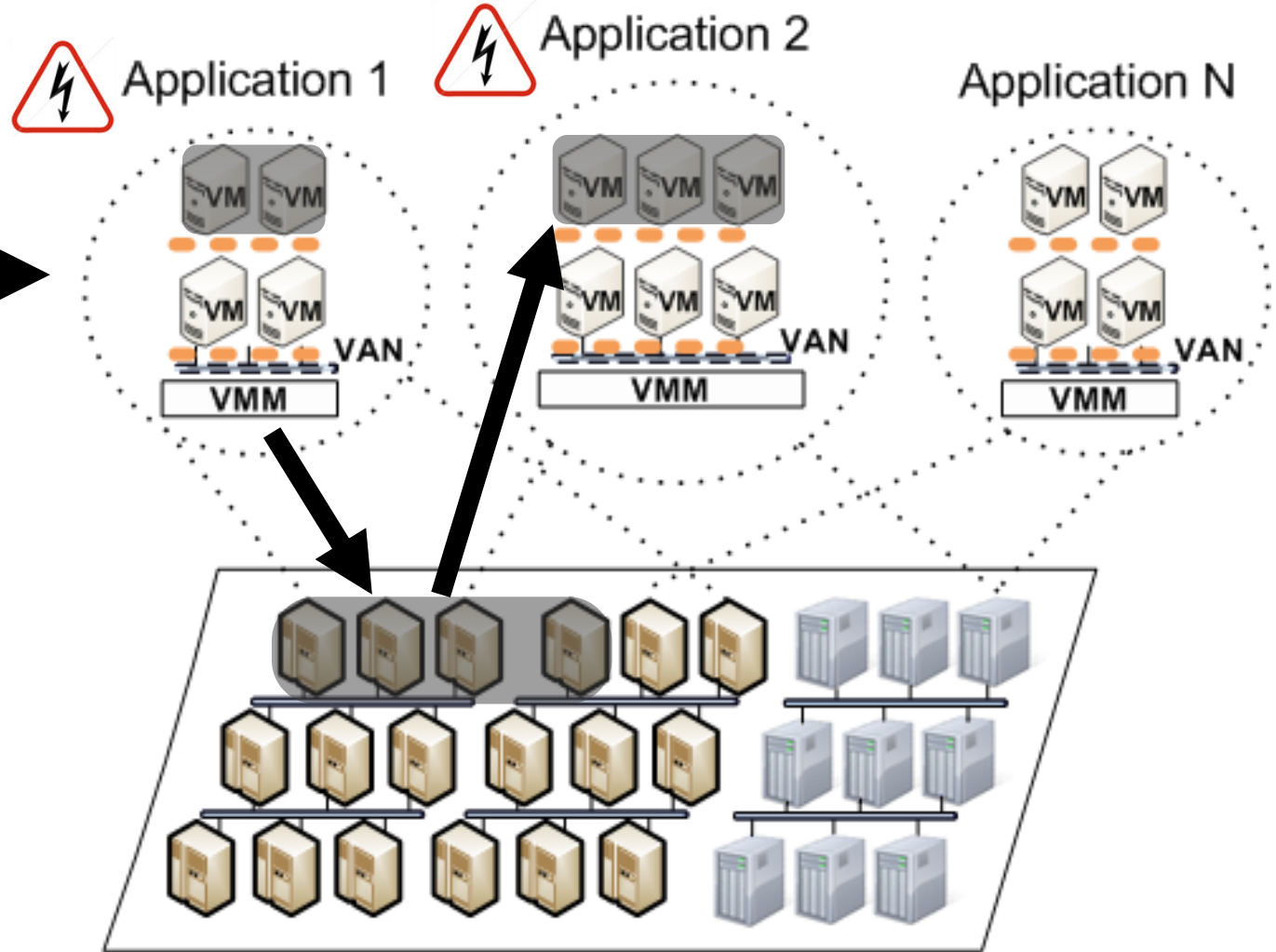


## Expand / shrink resources on-the-fly

- When exactly should a reconfiguration be triggered?
- Which particular resources should be scaled?
- How quickly and at what granularity?

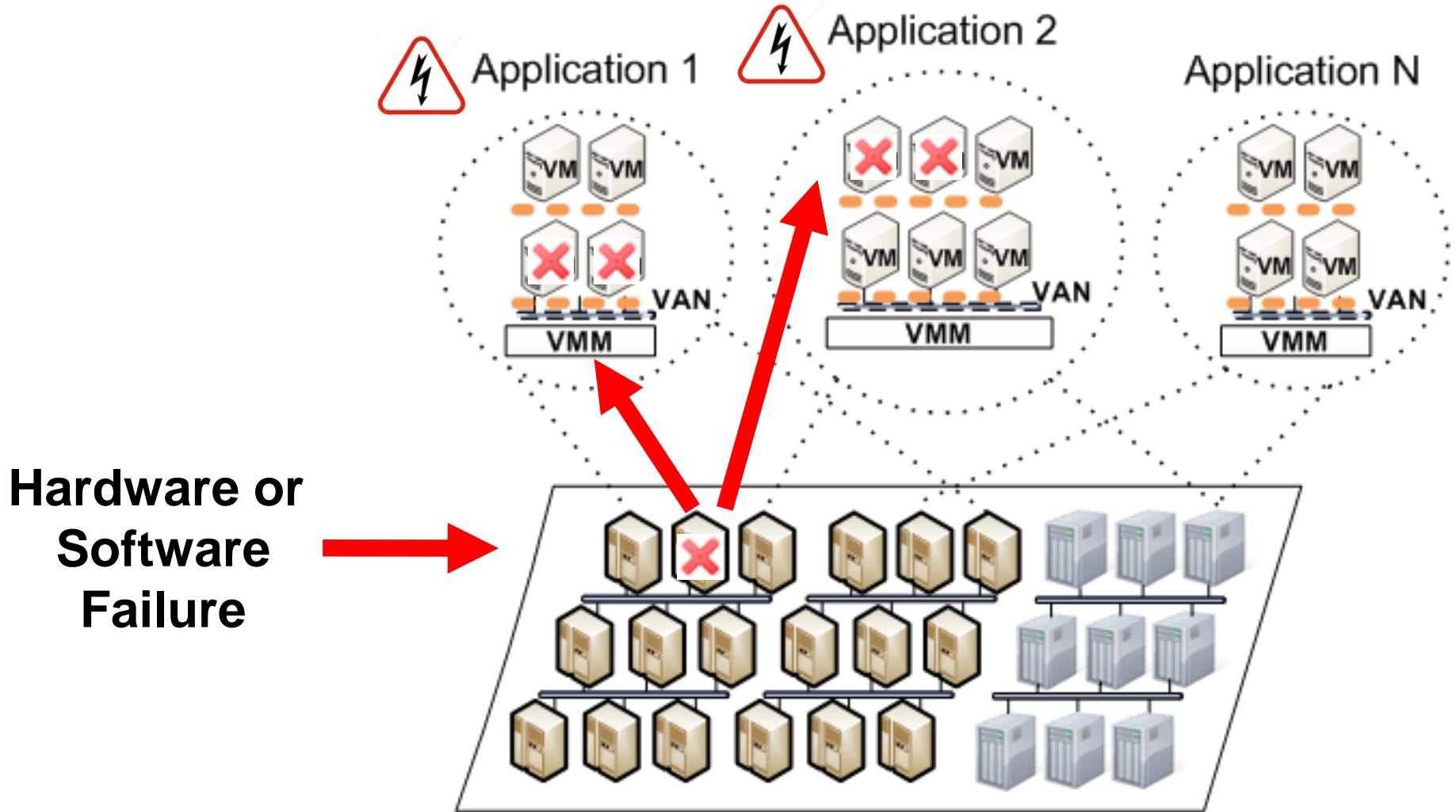
# Challenges

  
**Security Attack**





# Challenges





# Consequences

- Challenges
  - Increased system complexity and dynamics
  - Diverse vulnerabilities due to resource sharing
- **Service “dependability”** → major distinguishing factor between cloud platforms
  - Availability, reliability (+ security, performance, ...)
- **Lack of reliable benchmarks to evaluate dependability**



*“You can’t **control** what you can’t measure?” (DeMarco)*

*“If you cannot measure it, you cannot **improve** it” (Lord Kelvin)*

# Need for Benchmarks

*“To **measure** is to **know**.”* -- Clerk Maxwell, 1831-1879

*“It is much easier to make **measurements** than to **know** exactly what you are measuring.”* -- J.W.N.Sullivan (1928)

## 1. Reliable Metrics

- What exactly should be measured and computed?

## 2. Representative Workloads

- For which usage scenarios and under what conditions?

## 3. Sound Measurement Methodology

- How should measurements be conducted?

# The Focus of this Talk

- Metrics and benchmarks for quantitative evaluation of
  1. **Resource elasticity**
  2. Performance isolation
  3. Intrusion detection (and prevention)
- in shared execution environments
  - Virtualized infrastructures (e.g., as in IaaS)
  - Multi-tenant applications (e.g., as in SaaS)



## Main references

N. Herbst, A. Weber, H. Groenda and S. Kounev. **BUNGEE: Benchmarking Resource Elasticity of Cloud Environments**. Submitted to *6th ACM/SPEC Intl. Conf. on Performance Engineering (ICPE 2015)*.

N. Herbst, S. Kounev and R. Reussner. **Elasticity in Cloud Computing: What it is, and What it is Not**. In *Proc. of the 10th Intl. Conf. on Autonomic Computing (ICAC 2013)*, San Jose, CA, June 24-28, 2013. USENIX. [ [slides](#) | [http](#) | [.pdf](#) ]

## Further references

N. Herbst, N. Huber, S. Kounev and E. Amrehn. **Self-Adaptive Workload Classification and Forecasting for Proactive Resource Provisioning**. *Concurrency and Computation - Practice and Experience*, John Wiley and Sons, Ltd., 26(12):2053-2078, 2014. [ [DOI](#) | [http](#) ]

J. von Kistowski, N. Herbst and S. Kounev. **LIMBO: A Tool For Modeling Variable Load Intensities** (Demonstration Paper). In *Proc. of the 5th ACM/SPEC Intl. Conf. on Performance Engineering (ICPE 2014)*, Dublin, Ireland, March 22-26, 2014. ACM. [ [DOI](#) | [slides](#) | [http](#) | [.pdf](#) ]

J. von Kistowski, N. Herbst and S. Kounev. **Modeling Variations in Load Intensity over Time**. In *Proc. of the 3rd Intl. Workshop on Large-Scale Testing (LT 2014)*, co-located with ICPE 2014, Dublin, Ireland, March 22, 2014. ACM. [ [DOI](#) | [slides](#) | [http](#) | [.pdf](#) ]

A. Weber, N. Herbst, H. Groenda and S. Kounev. **Towards a Resource Elasticity Benchmark for Cloud Environments**. In *Proc. of the 2nd Intl. Workshop on Hot Topics in Cloud Service Scalability (HotTopiCS 2014)*, co-located with ICPE 2014, March 22, 2014. ACM. [ [slides](#) | [.pdf](#) ]

# What People Say Elasticity Is...

- **ODCA, Compute Infrastructure-as-a-Service:**

*"[...] defines elasticity as the configurability and expandability of the solution[...] Centrally, it is the ability to scale **up and** scale **down** capacity **based on subscriber workload**."*

- **NIST Definition of Cloud Computing**

*"**Rapid** elasticity: Capabilities can be elastically **provisioned and released**, in some cases **automatically**, to scale rapidly outward and inward **commensurate with demand**."*

- **IBM, Thoughts on Cloud, Edwin Schouten:**

*"Elasticity is basically a 'rename' of scalability [...]" and "removes any manual labor needed to **increase or reduce** capacity."*

- **Rich Wolski, CTO, Eucalyptus:**

*"Elasticity **measures** the ability of the cloud to map a single user request to different resources."*

- **Reuven Cohen:**

*Elasticity is "the **quantifiable** ability to manage, measure, predict and adapt responsiveness of an application **based on real time demands** placed on an infrastructure using a combination of local and remote computing resources."*

# What People Say Elasticity Is...

**OCDA [1]**

up & down scaling  
with subscriber workload

**NIST [2]**

rapid elasticity    unlimited  
provisioning & releasing  
sometimes automated  
with demand

**IBM, Schouten [3]**

scalability  
increasing & reducing  
no manual labor

**Eukalyptus, Wolski [4]**

measurable  
mapping of  
requests to resources

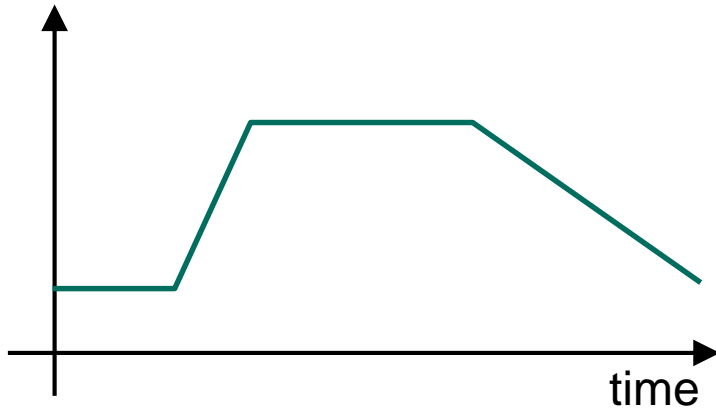
**Cohen [5]**

quantifiable  
real-time demands  
local & remote



# Elasticity (in Cloud Computing)

Workload intensity (e.g., # requests / sec)



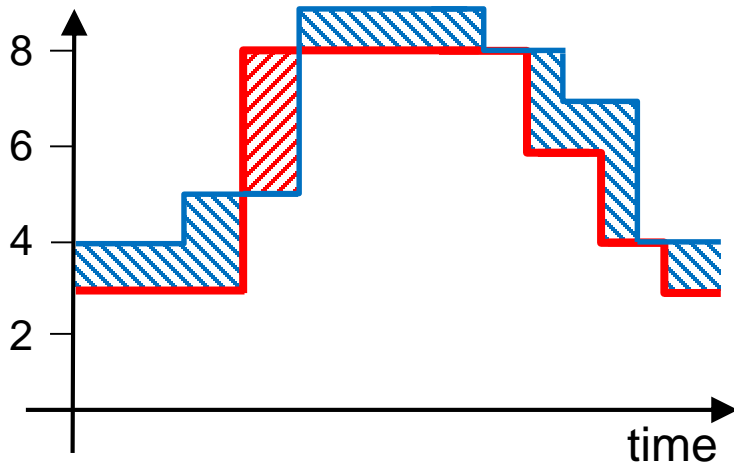
## Service Level Objective (SLO)





(e.g., resp. time  $\leq 2$  sec, 95%)

## Resource Demand

Minimal amount of resources required to ensure SLOs

Amount of resources (e.g., # VMs)



-  resource demand
-  underprovisioning
-  resource supply
-  overprovisioning



Def: The degree to which a system is able to **adapt** to **workload changes** by **provisioning and deprovisioning** resources in an **autonomic manner**, such that at each point in time the **available resources match** the **current demand** as closely as possible.

*N. Herbst, S. Kounev and R. Reussner*

*Elasticity in Cloud Computing: What it is, and What it is Not.*

*in Proceedings of the 10th International Conference on Autonomic Computing (ICAC 2013), San Jose, CA, June 24-28, 2013.*

[http://en.wikipedia.org/wiki/Elasticity\\_\(cloud\\_computing\)](http://en.wikipedia.org/wiki/Elasticity_(cloud_computing))

# Need for Benchmarks

*“To **measure** is to **know**.”* -- Clerk Maxwell, 1831-1879

*“It is much easier to make **measurements** than to **know** exactly what you are measuring.”* -- J.W.N.Sullivan (1928)

## 1. Reliable Metrics

- What exactly should be measured and computed?

## 2. Representative Workloads

- For which usage scenarios and under what conditions?

## 3. Sound Measurement Methodology

- How should measurements be conducted?

# Benchmarking Elasticity

## 1. Reliable Metrics

- What exactly should be measured and computed?

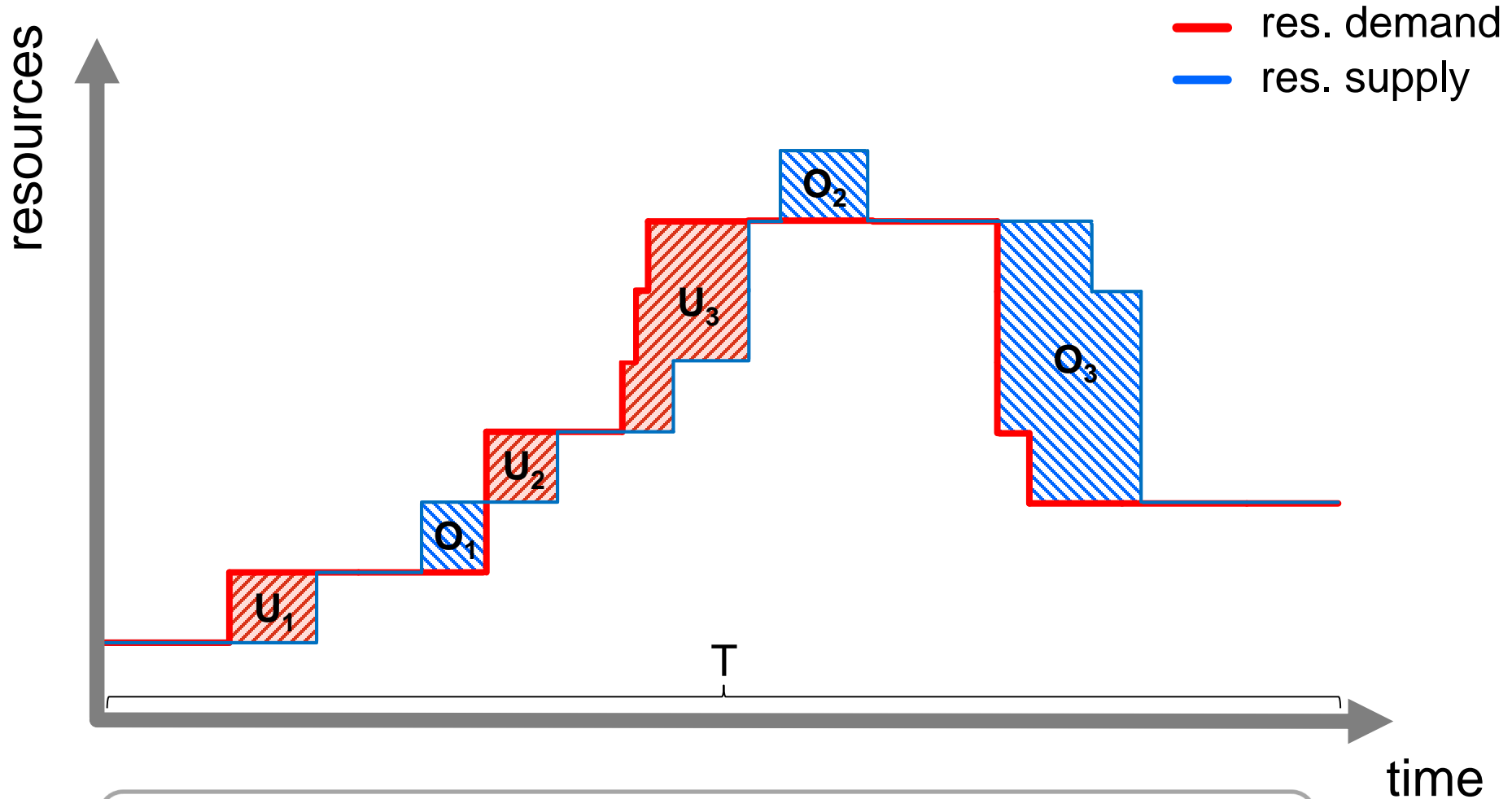
## 2. Representative Workloads

- For which usage scenarios and under what conditions?

## 3. Sound Measurement Methodology

- How should measurements be conducted?

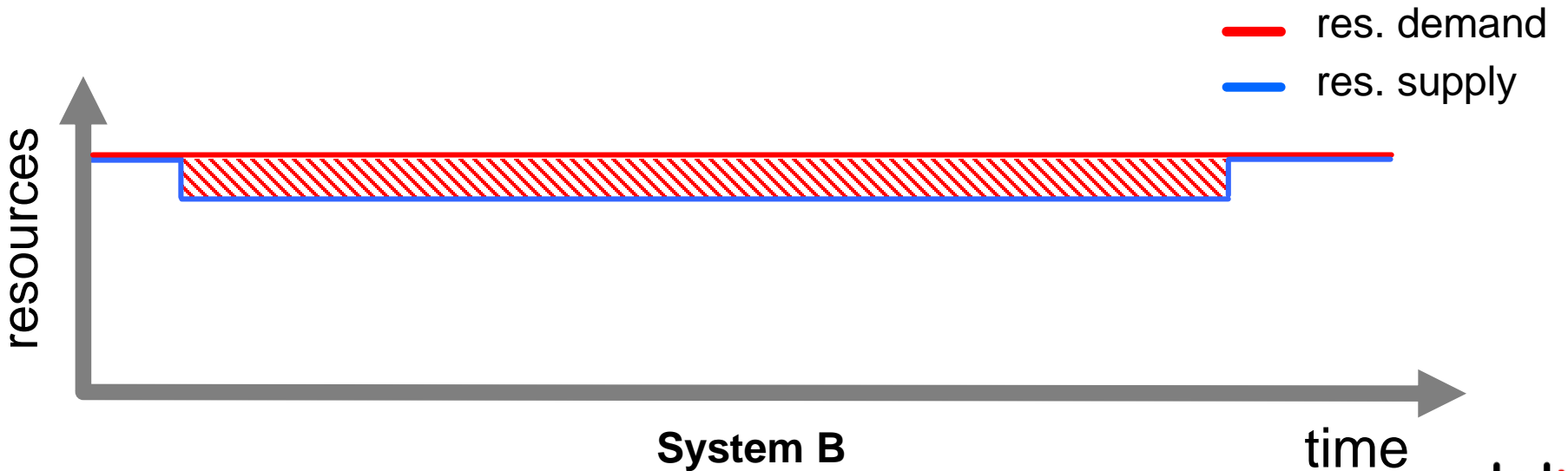
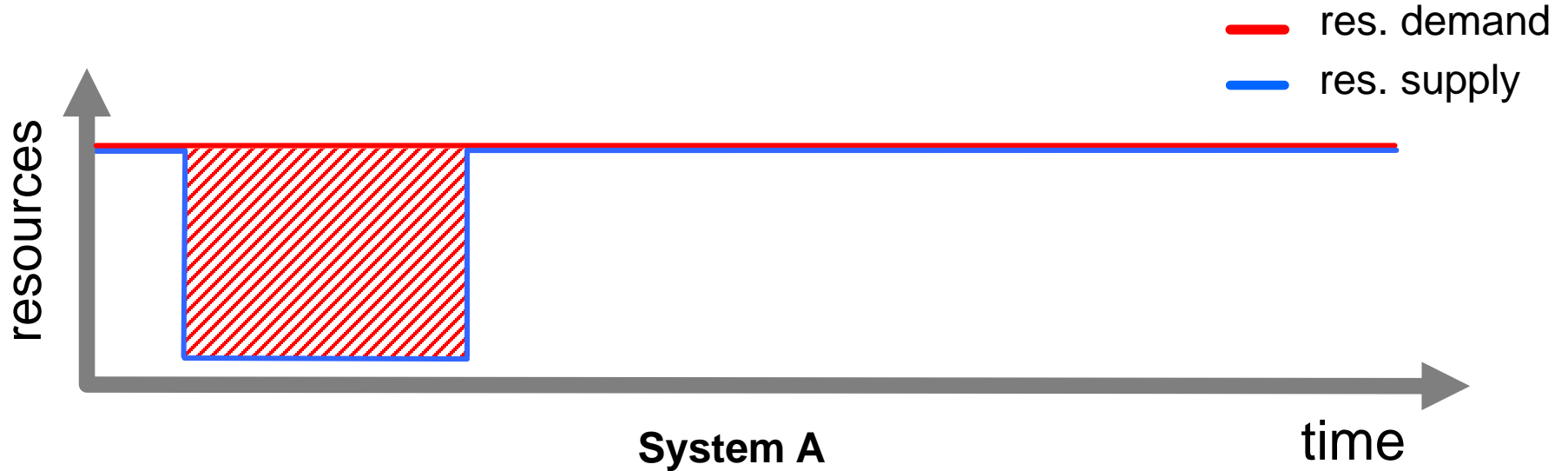
# Metric 1: Accuracy



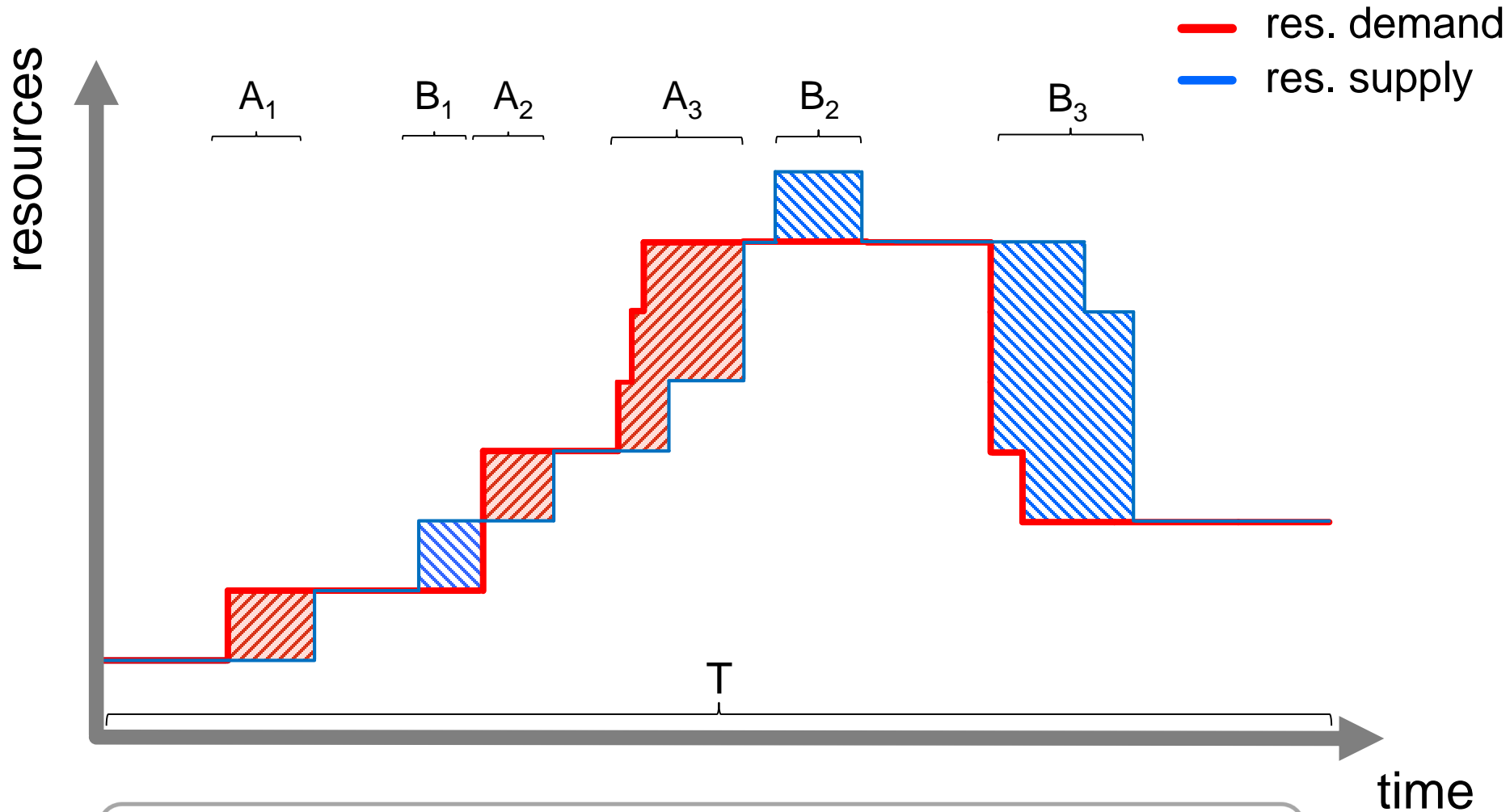
(1) accuracy<sub>U</sub>:  $\frac{\sum U}{T}$

(2) accuracy<sub>O</sub>:  $\frac{\sum O}{T}$

# Same Metric Values - Different Behavior!



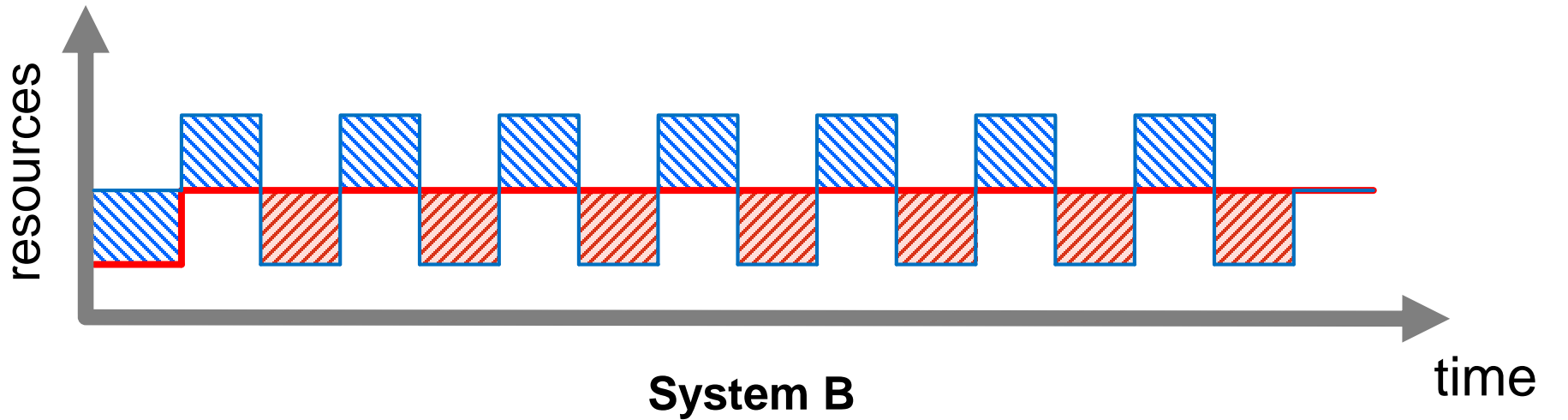
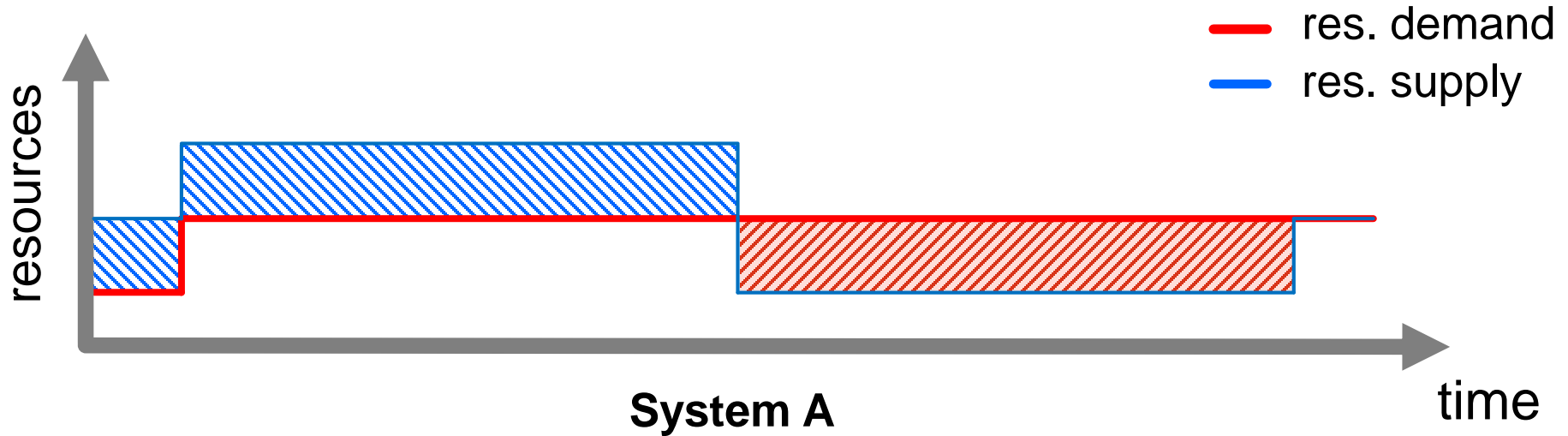
# Metric 2: Timeshare



(3)  $\text{timeshare}_U: \frac{\sum A}{T}$

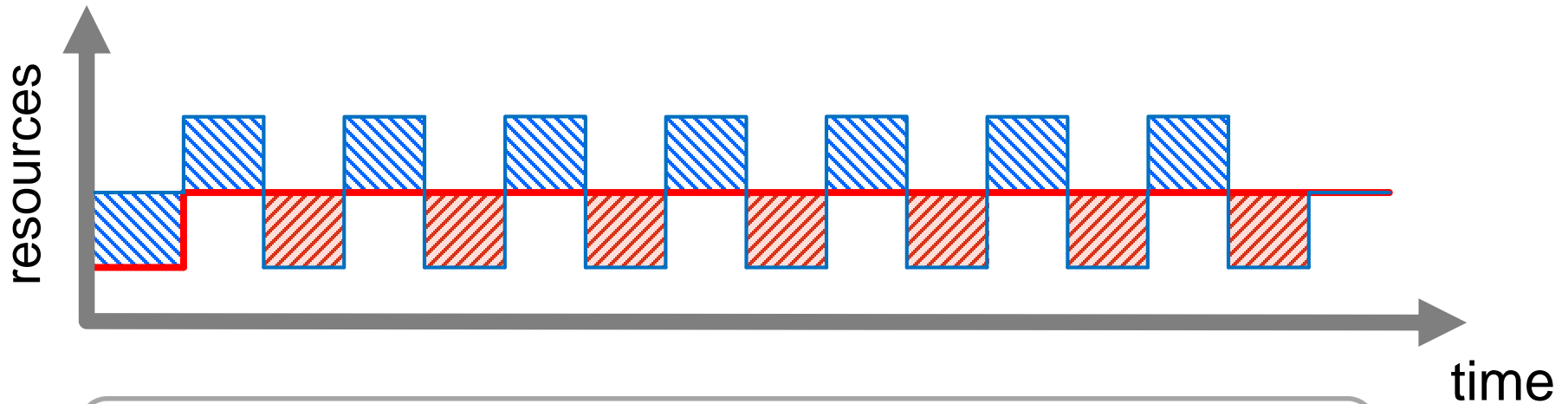
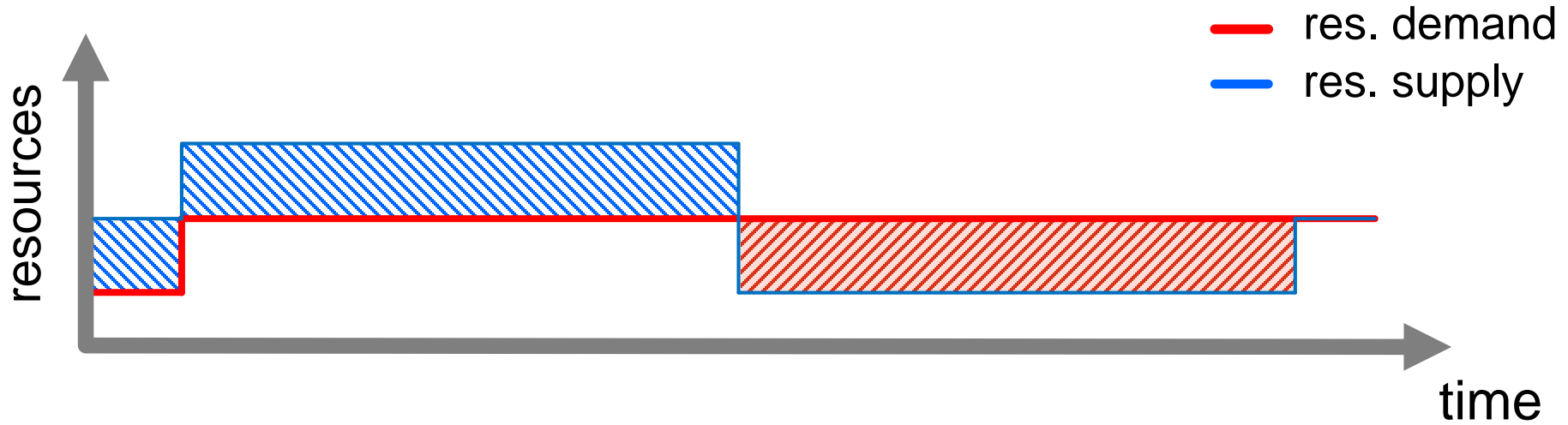
(4)  $\text{timeshare}_O: \frac{\sum B}{T}$

# Same Metric Values - Different Behavior!





# Metric 3: Jitter



$$(5) \text{ jitter: } \frac{E_S - E_D}{T}$$

$E_D$ : # demand changes

$E_S$ : # supply changes

# Benchmarking Elasticity

## 1. Reliable Metrics

- What exactly should be measured and computed?

## 2. Representative Workloads

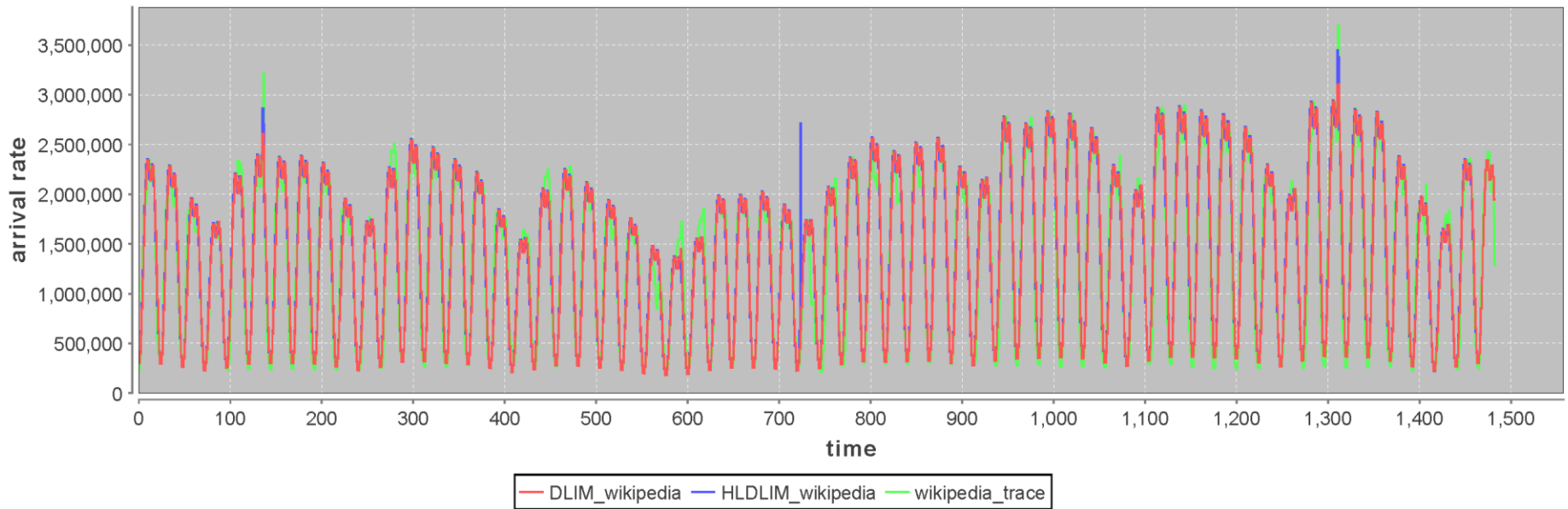
- For which usage scenarios and under what conditions?

## 3. Sound Measurement Methodology

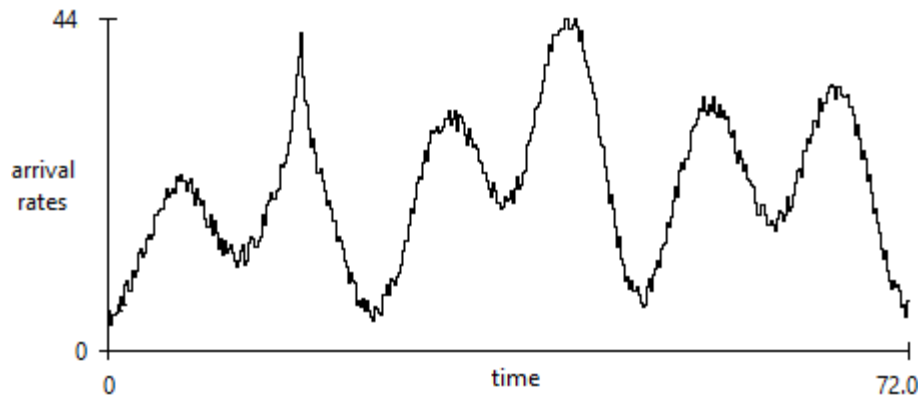
- How should measurements be conducted?

# Example: Wikipedia Workload Trace

DLIM\_wikipedia Arrival Rates



# Extracting Models of Real-Life Traces






<http://descartes.tools/limbo>

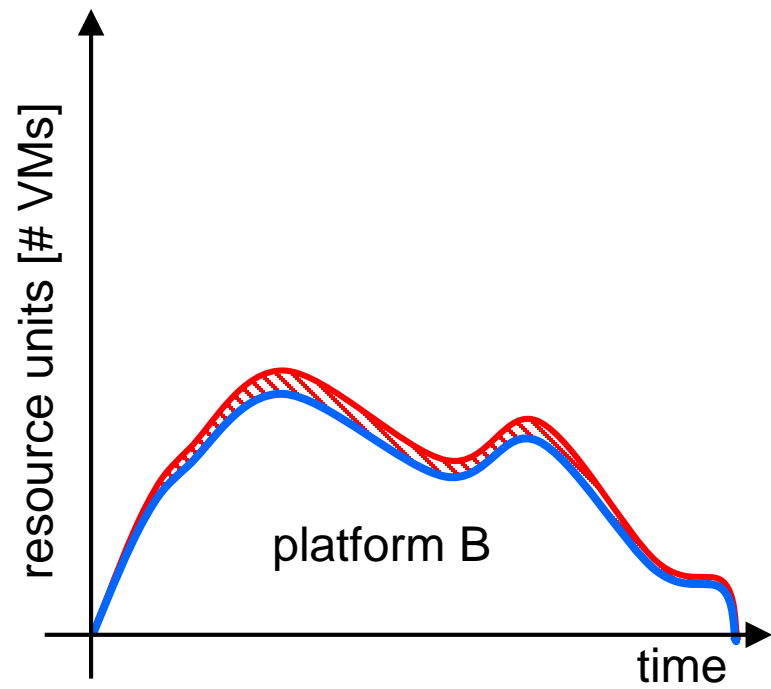
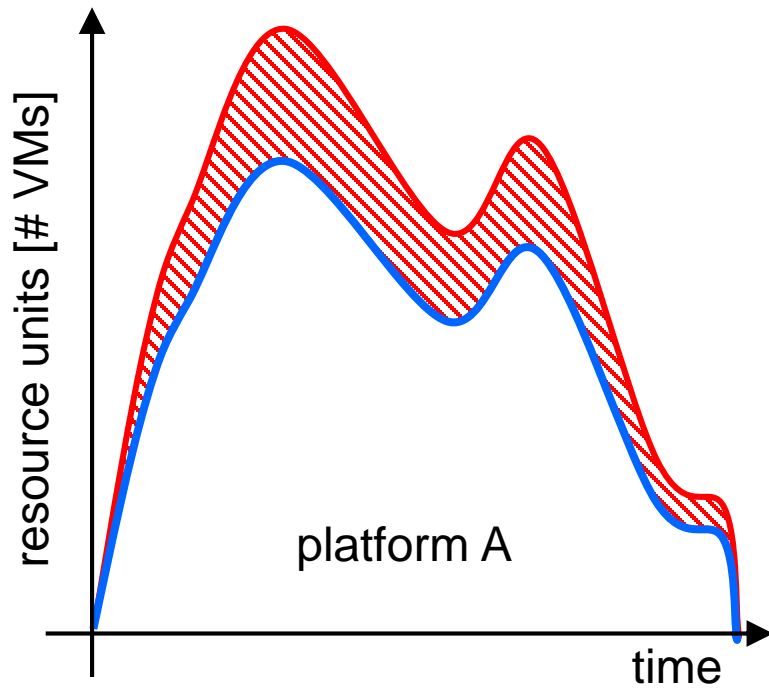
J. von Kistowski, N. Herbst and S. Kounev. **LIMBO: A Tool For Modeling Variable Load Intensities** (Demonstration Paper). In *Proc. of the 5th ACM/SPEC Intl. Conf. on Performance Engineering (ICPE 2014)*, Dublin, Ireland, March 22-26, 2014. ACM. [ [DOI](#) | [slides](#) | [http](#) | [.pdf](#) ]

J. von Kistowski, N. Herbst and S. Kounev. **Modeling Variations in Load Intensity over Time**. In *Proc. of the 3rd Intl. Workshop on Large-Scale Testing (LT 2014)*, Dublin, Ireland, March 22, 2014. ACM. [ [DOI](#) | [slides](#) | [http](#) | [.pdf](#) ]




# Same Workload on Two Platforms

-  Resource demand
-  Resource supply
-  Underprovisioning

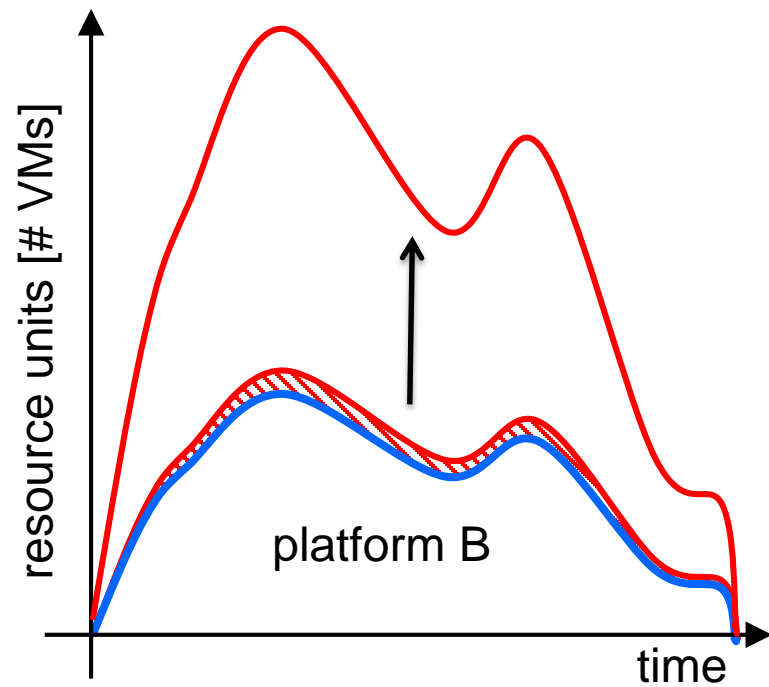
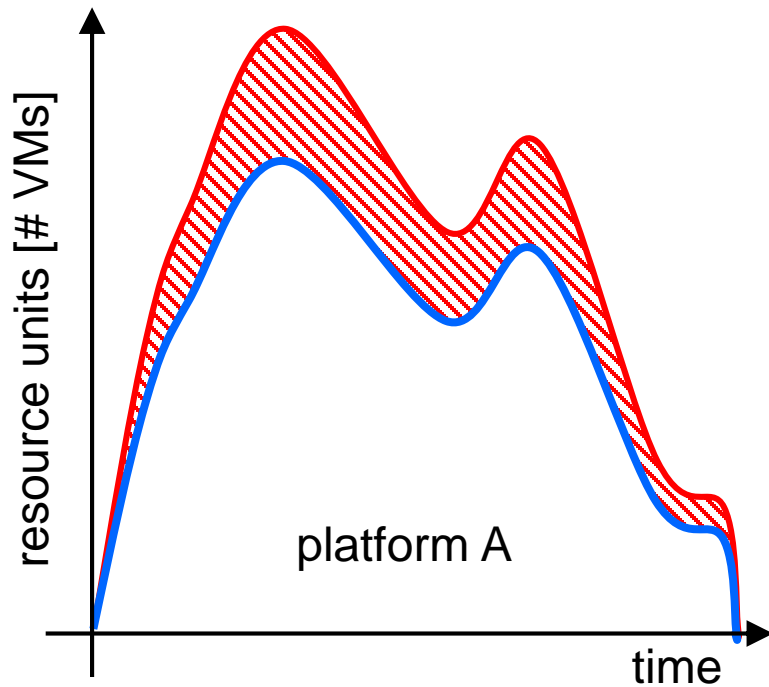
Same user workload on system B






# Same Workload on Two Platforms

-  Resource demand
-  Resource supply
-  Underprovisioning

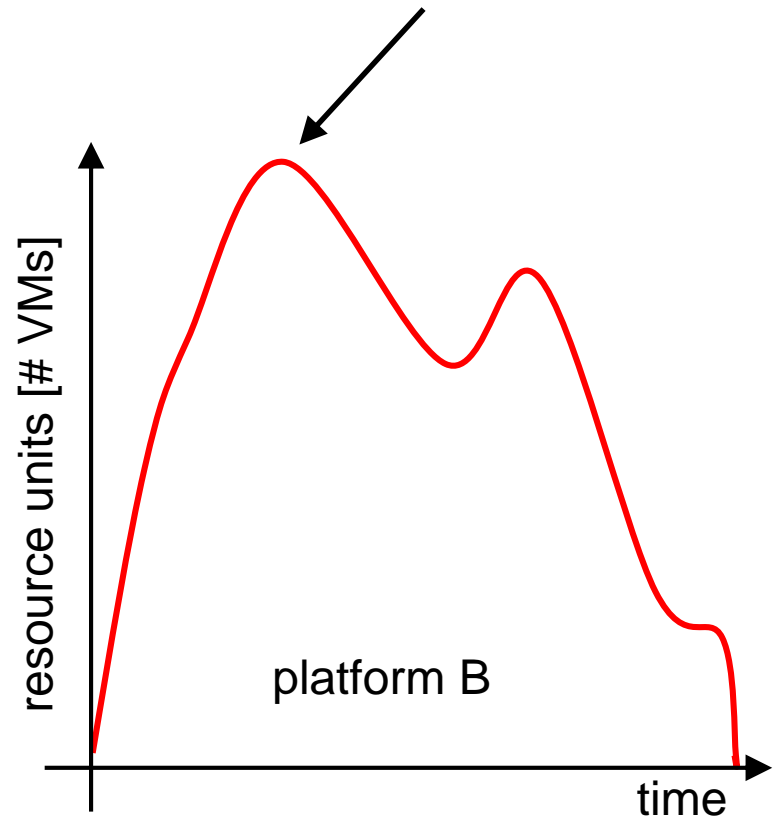
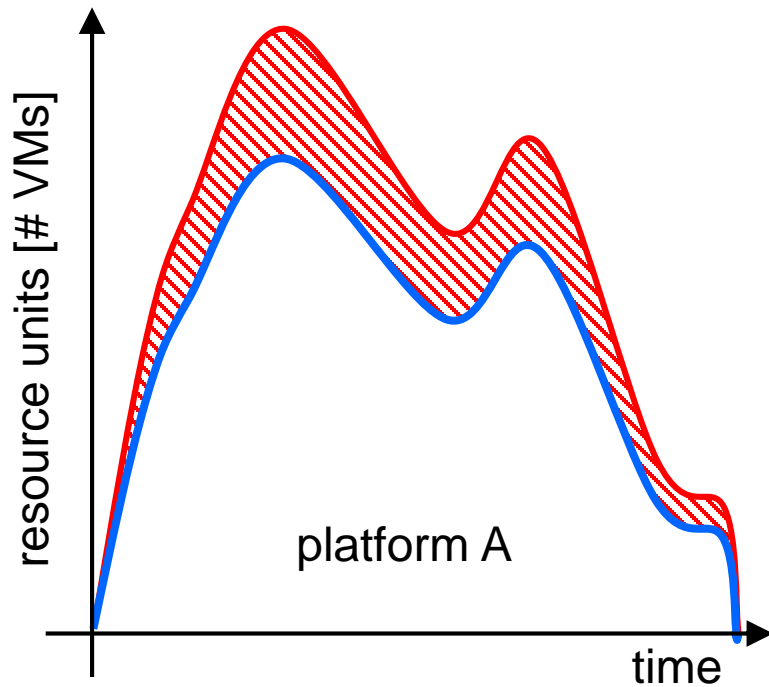
Load intensity adjusted to induce the same demand curve as for platform A



# Same Demand Variations on Two Platforms




-  Resource demand
-  Resource supply
-  Underprovisioning

System B at a user workload adjusted to induce the same demand curve

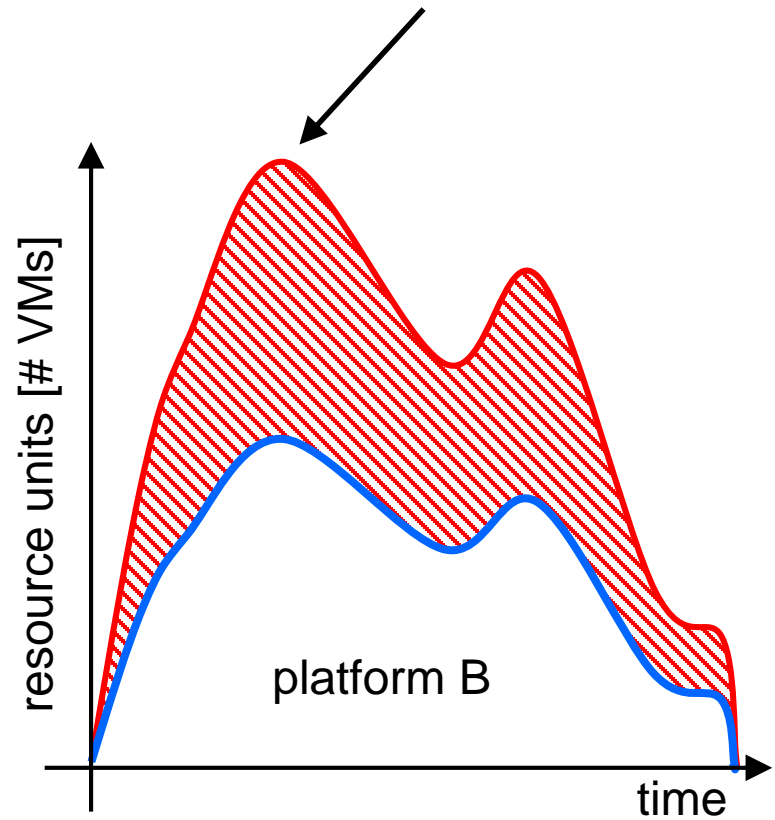
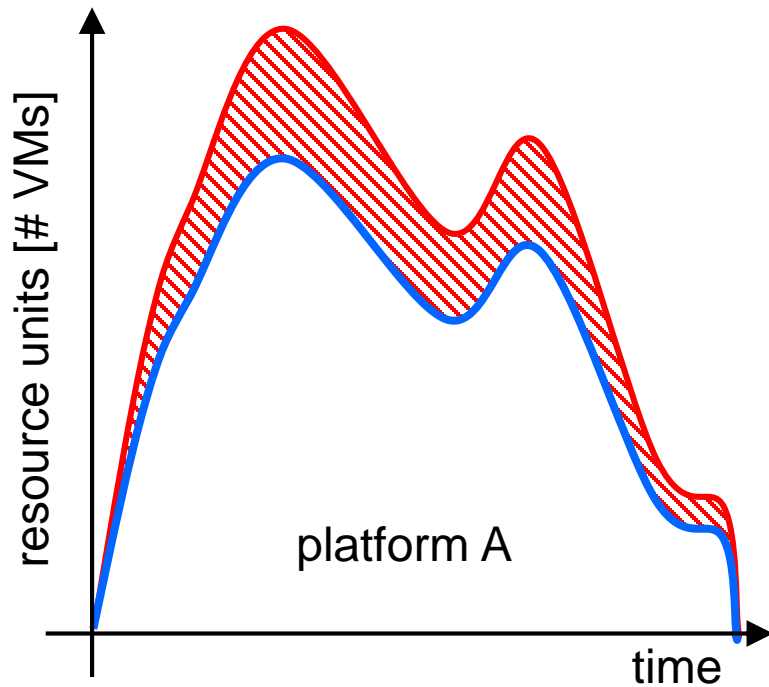




# Same Demand Variations on Two Platforms

-  Resource demand
-  Resource supply
-  Underprovisioning

System B at a user workload adjusted to induce the same demand curve



# Benchmarking Elasticity

## 1. Reliable Metrics

- What exactly should be measured and computed?

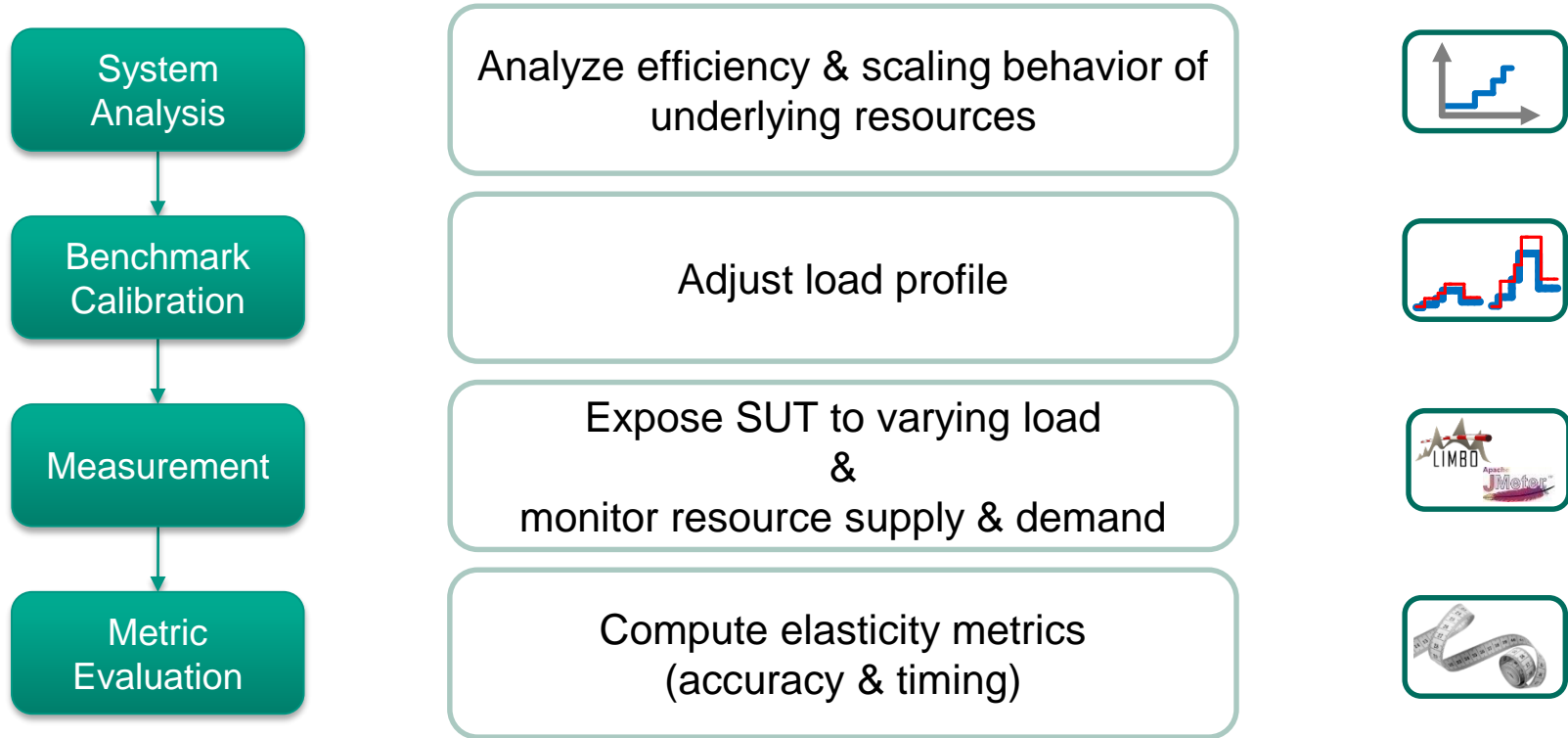
## 2. Representative Workloads

- For which usage scenarios and under what conditions?

## 3. Sound Measurement Methodology

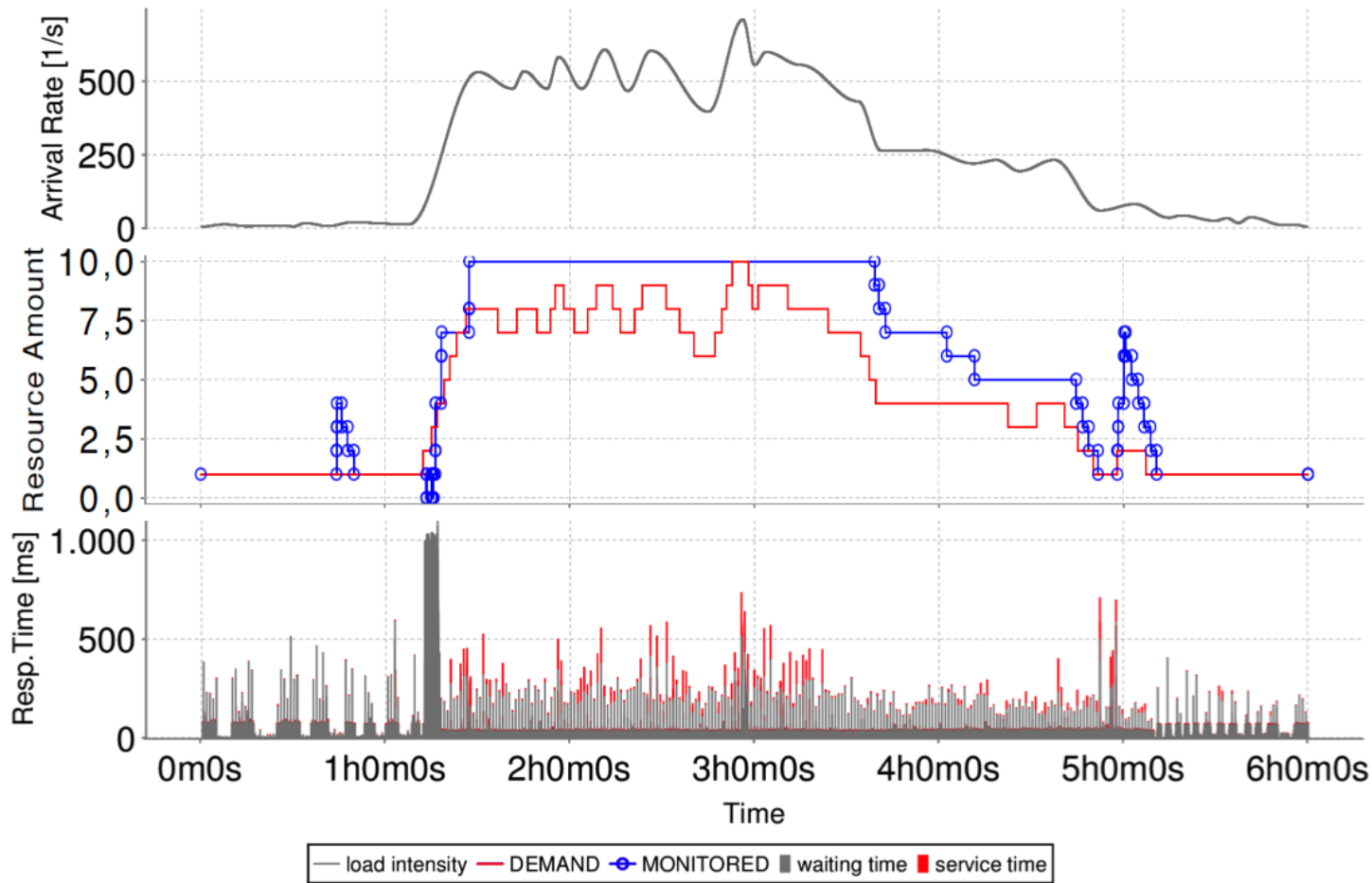
- How should measurements be conducted?

# Elasticity Benchmarking Approach



N. Herbst, S. Kounev, A. Weber and H. Groenda. **BUNGEE: An Elasticity Benchmark for Self-Adaptive IaaS Cloud Environments**. In *10th Intl. Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2015)*, Firenze, Italy, May 18-19, 2015.

# Case Study: Amazon Web Services vs. CloudStack



Configuration	accuracy <sub>o</sub> [res. units]	accuracy <sub>u</sub> [res. units]	timeshare <sub>o</sub> [%]	timeshare <sub>u</sub> [%]	jitter [adap/min.]	elastic speedup	violations [%]
CS – 1Core	2.423	0.067	66.1	4.8	-0.067	1.046	7.6
CS – 2Core adjusted	2.508	0.061	67.1	4.5	-0.044	1.025	8.2
AWS - m1.small	1.340	0.019	61.6	1.4	0.000	1.502	2.5

## Main references

R. Krebs, C. Momm and S. Kounev. **Metrics and Techniques for Quantifying Performance Isolation in Cloud Environments**. *Elsevier Science of Computer Programming Journal (SciCo)*, Vol. 90, Part B:116-134, 2014, Elsevier B.V. [ [bib](#) | [.pdf](#) ]

R. Krebs, A. Wert and S. Kounev. **Multi-Tenancy Performance Benchmark for Web Application Platforms**. In *Proc. of the 13th Intl. Conf. on Web Engineering (ICWE 2013)*, Aalborg, Denmark, July 8-12, 2013. Springer-Verlag. [ [.pdf](#) ]

R. Krebs, C. Momm and S. Kounev. **Metrics and Techniques for Quantifying Performance Isolation in Cloud Environments**. In *Proc. of the 8th ACM SIGSOFT Intl. Conf. on the Quality of Software Architectures (QoSA 2012)*, Bertinoro, Italy, June 25-28, 2012. ACM. [ [http](#) | [.pdf](#) ]

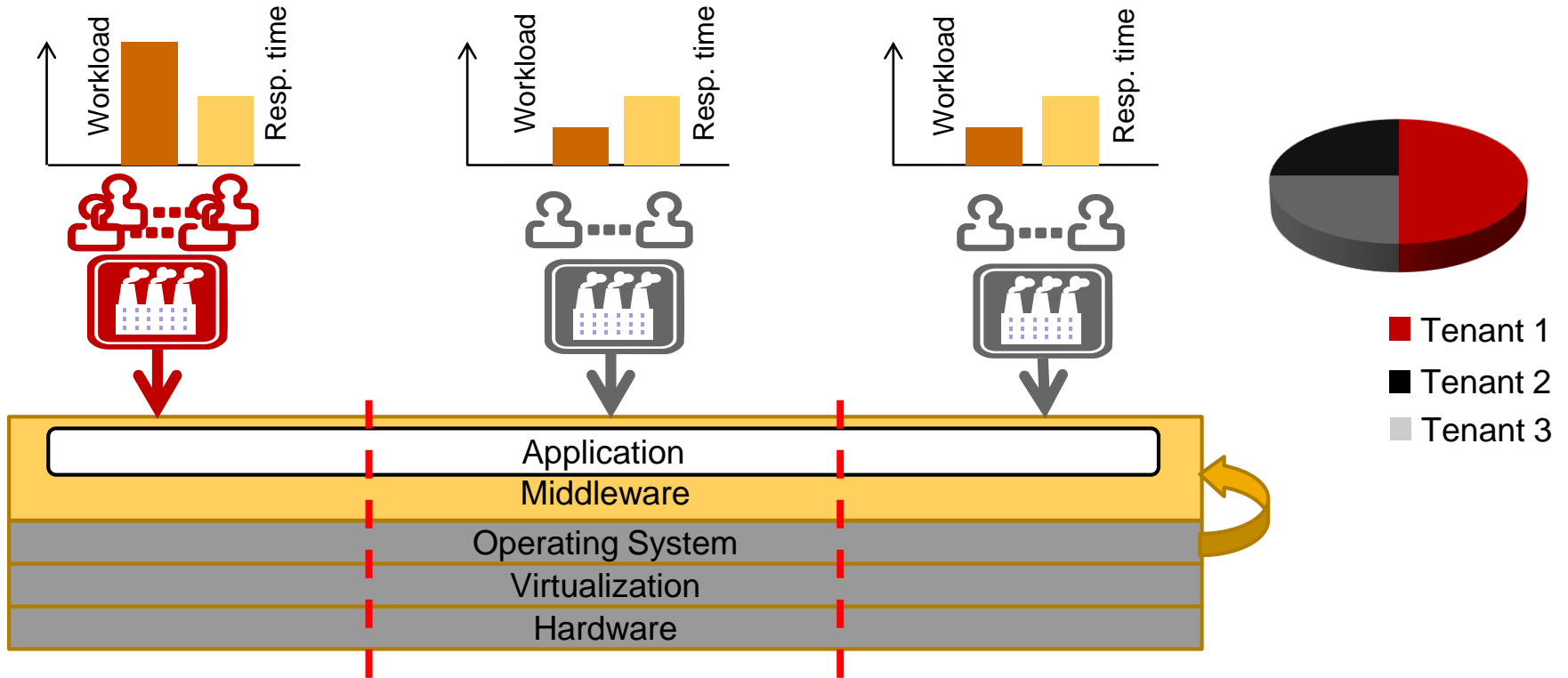
## Further references

R. Krebs, S. Spinner, N. Ahmed and S. Kounev. **Resource Usage Control In Multi-Tenant Applications**. In *Proc. of the 14th IEEE/ACM Intl. Symp. on Cluster, Cloud and Grid Computing (CCGrid 2014)*, Chicago, IL, USA, May 26, 2014. IEEE/ACM. [ [.pdf](#) ]

R. Krebs, M. Loesch and S. Kounev. **Platform-as-a-Service Architecture for Performance Isolated Multi-Tenant Applications**. In *Proc. of the 7th IEEE Intl. Conf. on Cloud Computing*, Anchorage, USA, July 2, 2014. IEEE.

R. Krebs, C. Momm and S. Kounev. **Architectural Concerns in Multi-Tenant SaaS Applications**. In *Proc. of 2nd Intl. Conf. on Cloud Computing and Services Science (CLOSER 2012)*, Setubal, Portugal, April 18-21, 2012. [ [.pdf](#) ]

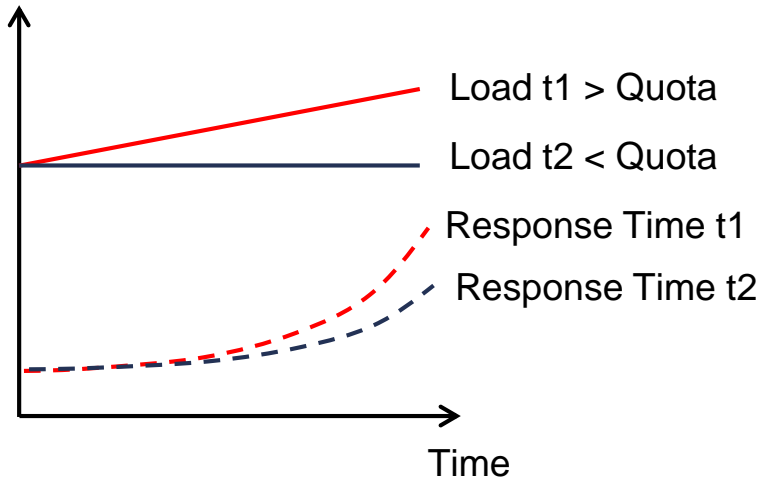
# Example Scenario: Multi-Tenant Environments



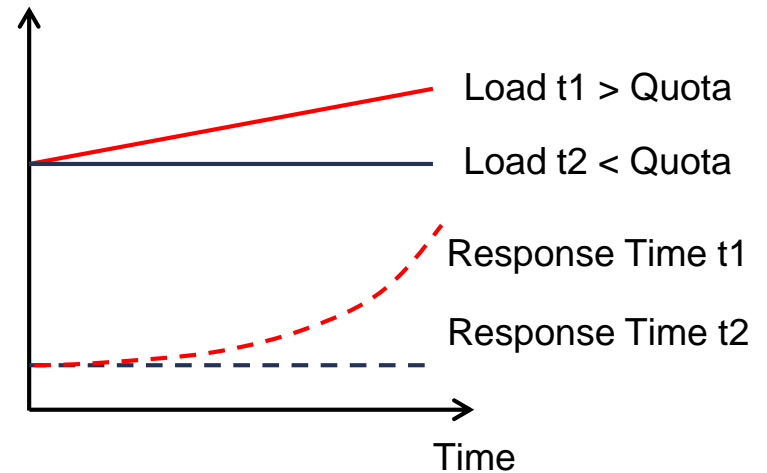
Tenants working within their assigned quota (e.g., # users) should not suffer from tenants exceeding their quotas.

# Definition of Performance Isolation

- Tenants working within their assigned quota (e.g., # users) should not suffer from tenants exceeding their quotas.



Non-Isolated System



Isolated System



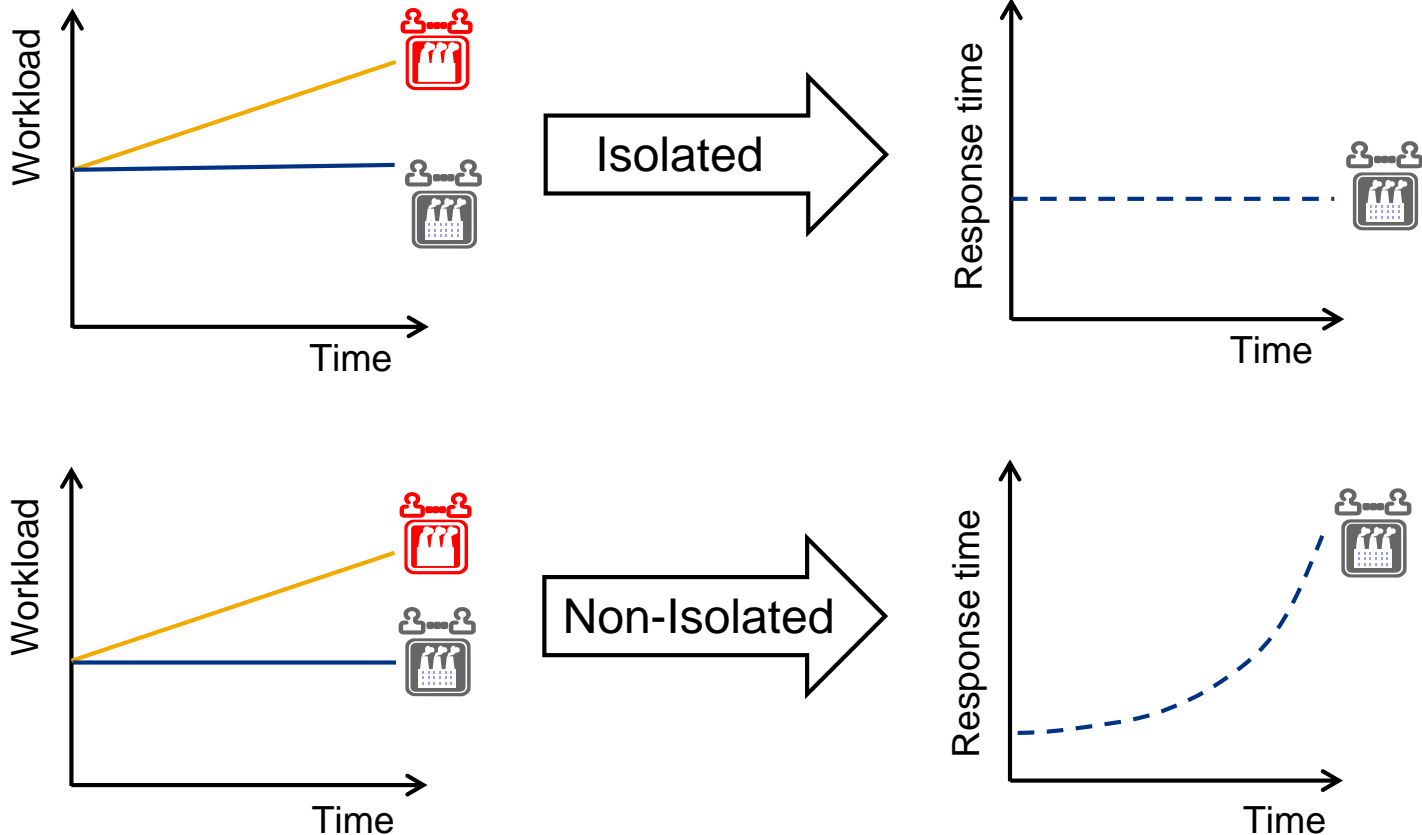
# Definition of Performance Isolation



D is a set of **disruptive tenants** exceeding their quotas.



A is a set of **abiding tenants** not exceeding their quotas.



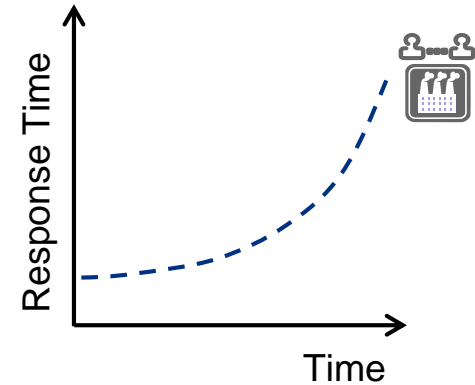
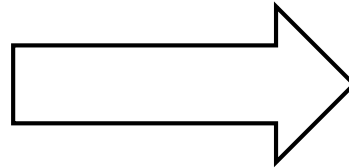
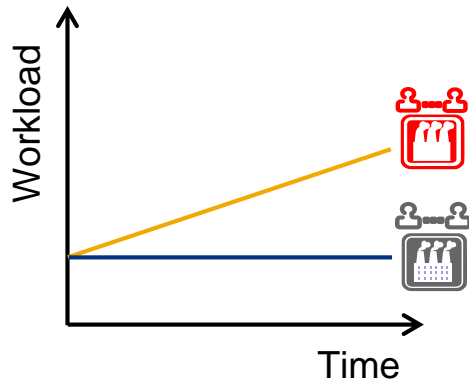
# Performance Isolation Metrics



D is a set of **disruptive tenants** exceeding their quotas.

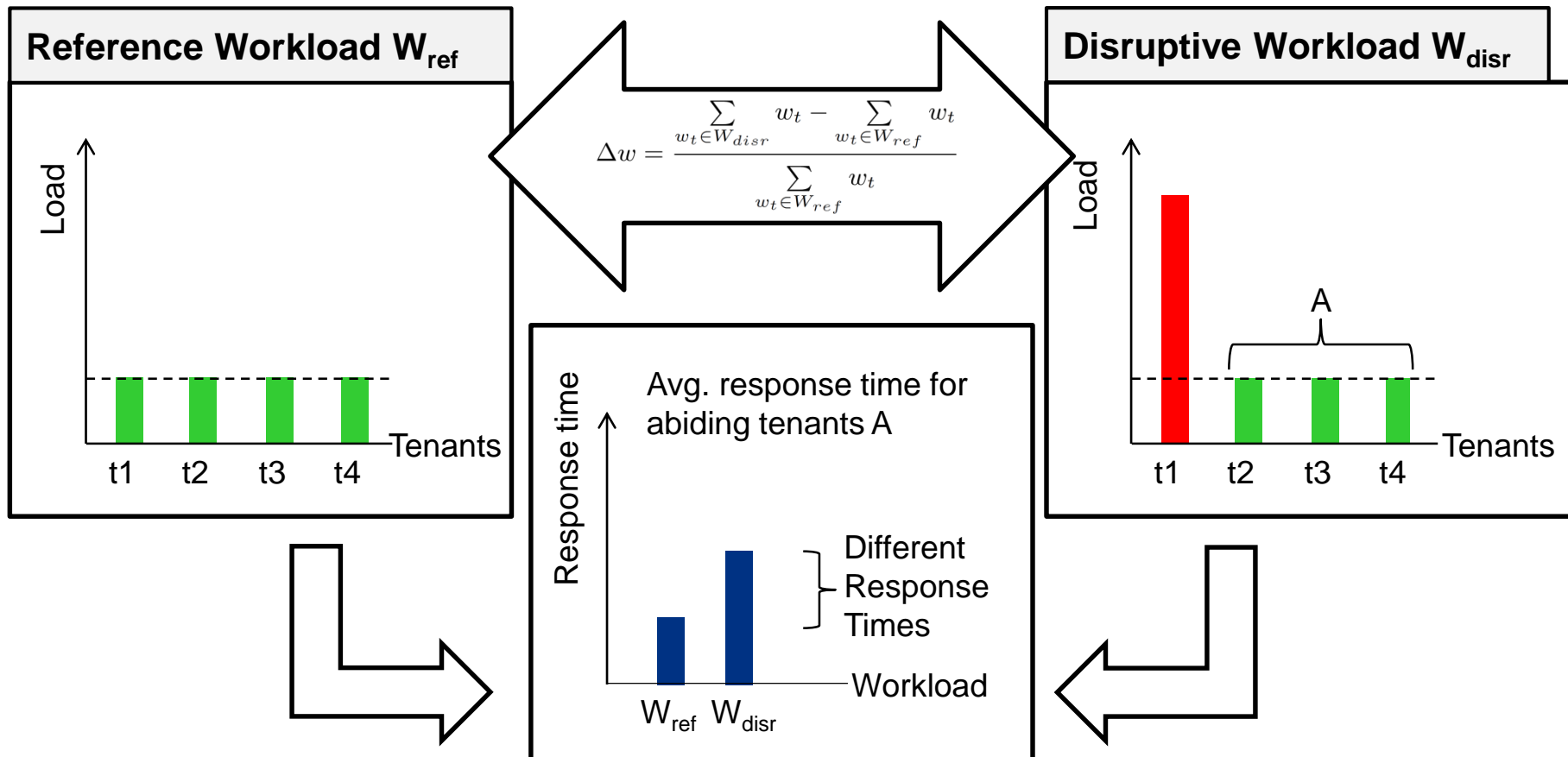


A is a set of **abiding tenants** not exceeding their quotas.



Approach: Quantify impact of increasing workload of the disruptive tenants on the performance of the abiding ones.

# Metrics Based on QoS Impact



$$\Delta z_A = \frac{\sum_{t \in A} [z_t(W_{disr}) - z_t(W_{ref})]}{\sum_{t \in A} z_t(W_{ref})}$$

# Example Metric

$$I_{QoS} = \frac{\Delta z_A}{\Delta w}$$

Difference in response time

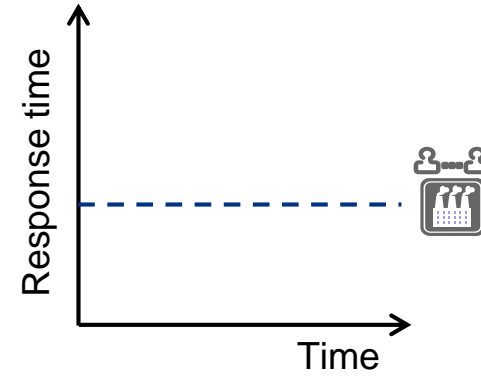
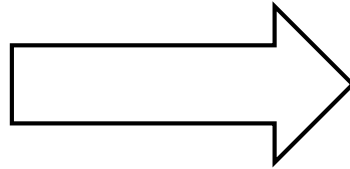
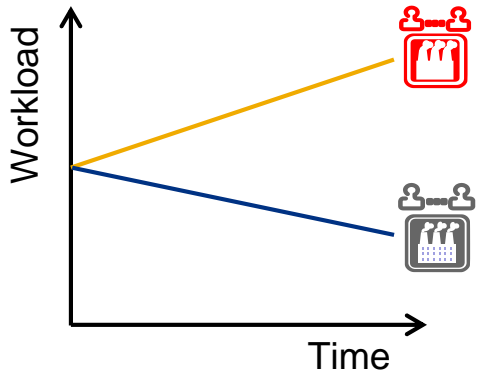
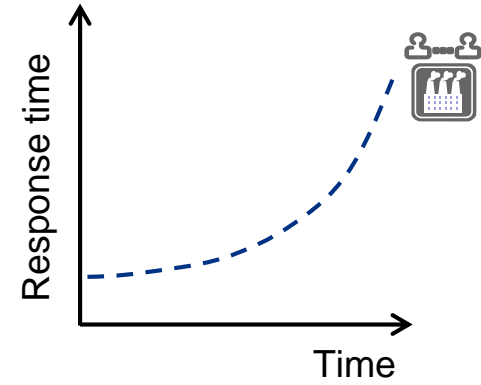
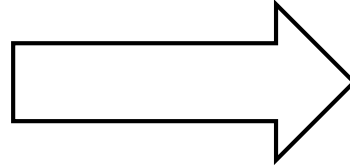
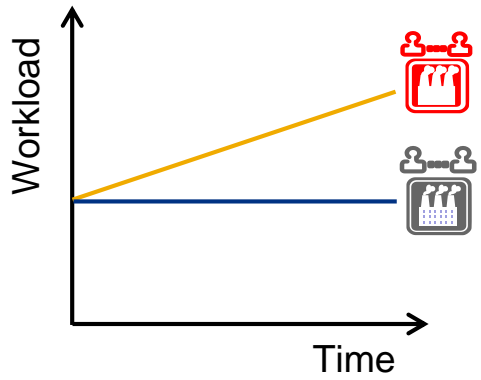
Difference in workload

**Perfectly Isolated = 0**

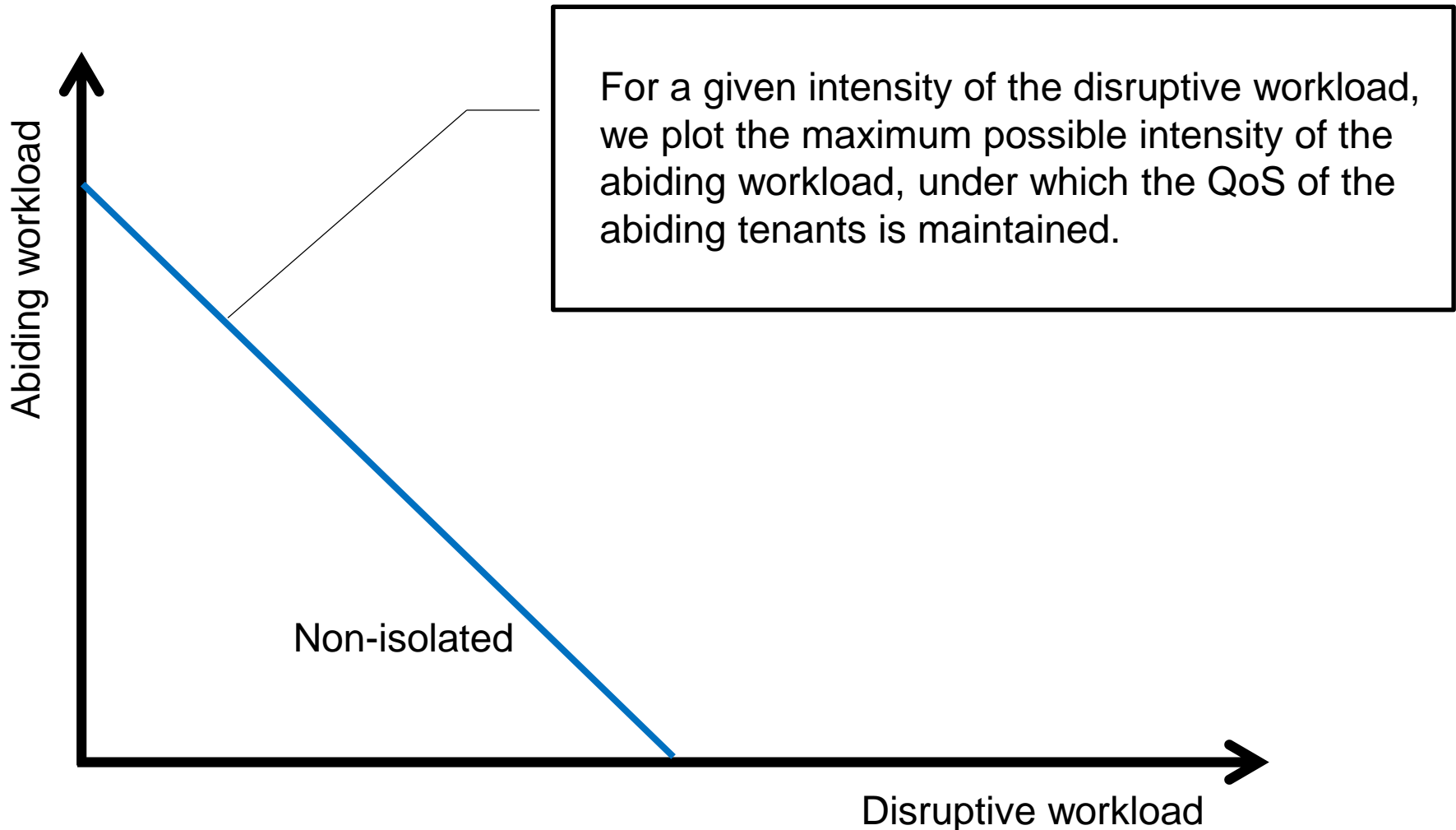
**Non-Isolated = ?**

**Answers: How strong is a tenant's influence on the others?**

# Metrics Based on Workload Ratio

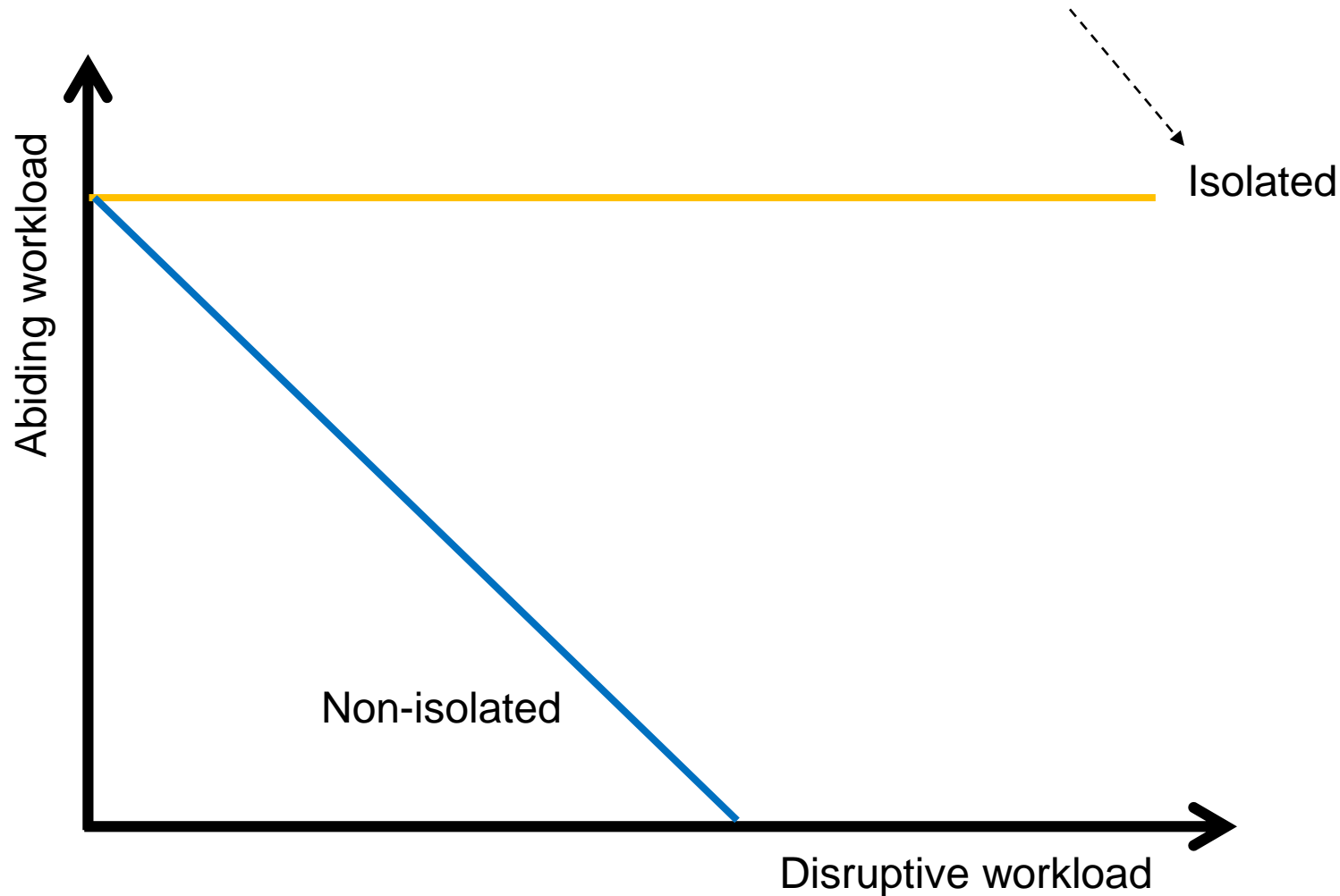


# Metrics Based on Workload Ratio

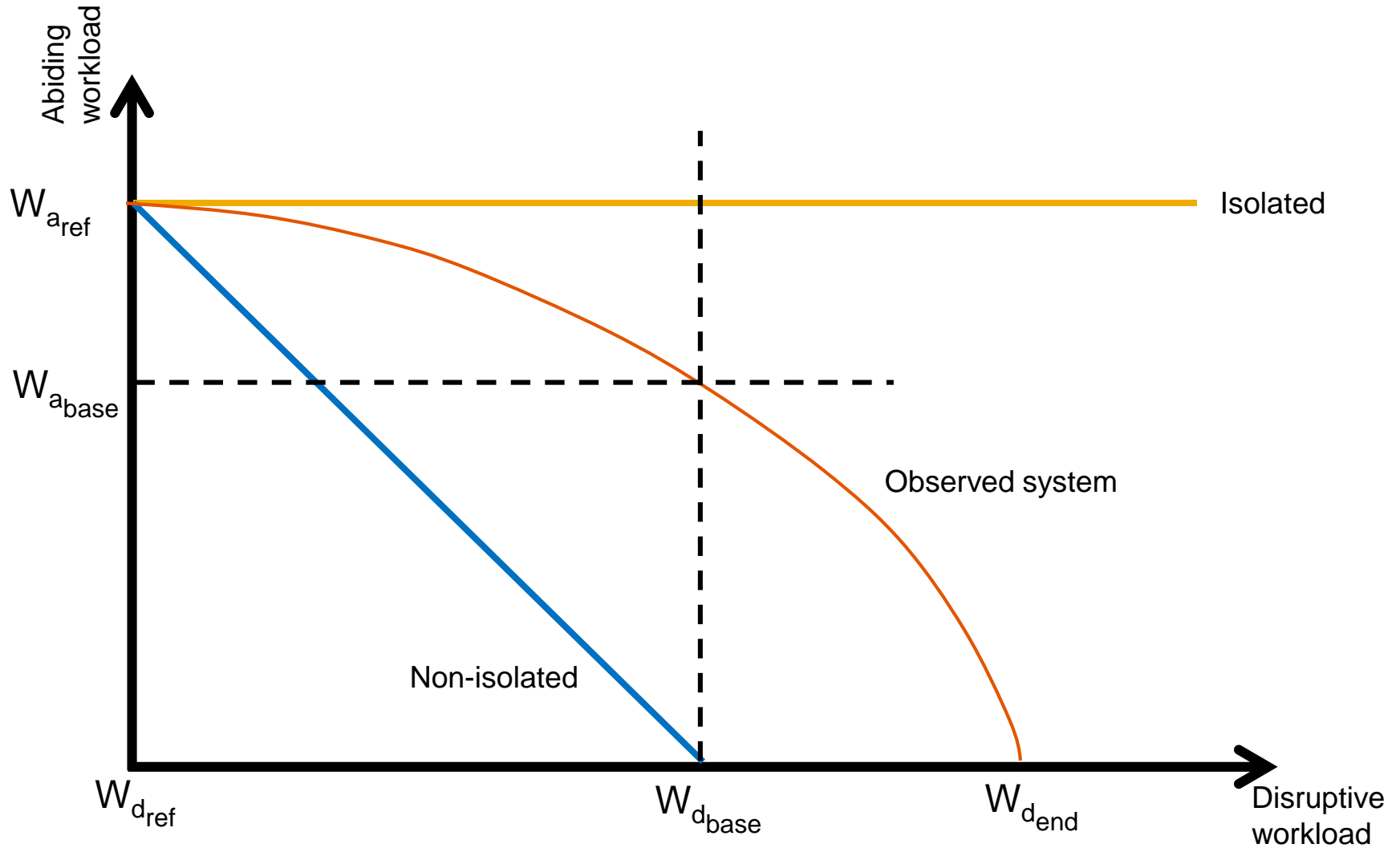


# Metrics Based on Workload Ratio

We can maintain the QoS for the abiding tenant without decreasing his workload.

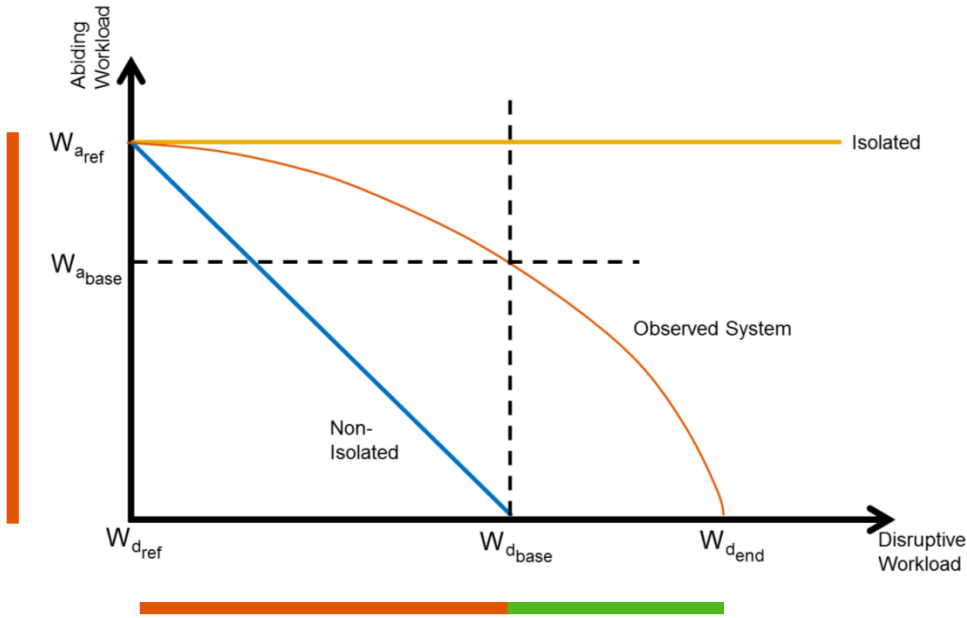


# Metrics Based on Workload Ratio





# Example Metric: $I_{end}$



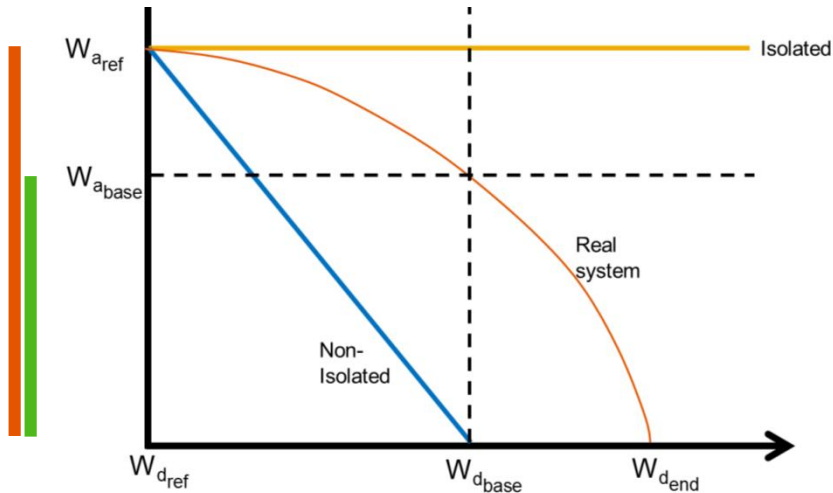
$$I_{end} = \frac{W_{d_{end}} - W_{d_{base}}}{W_{a_{ref}}}$$

**Perfectly Isolated = ?**

**Non-Isolated = 0**

**Answers: How isolated is the system compared to a non-isolated system?**

# Example Metric: $I_{base}$



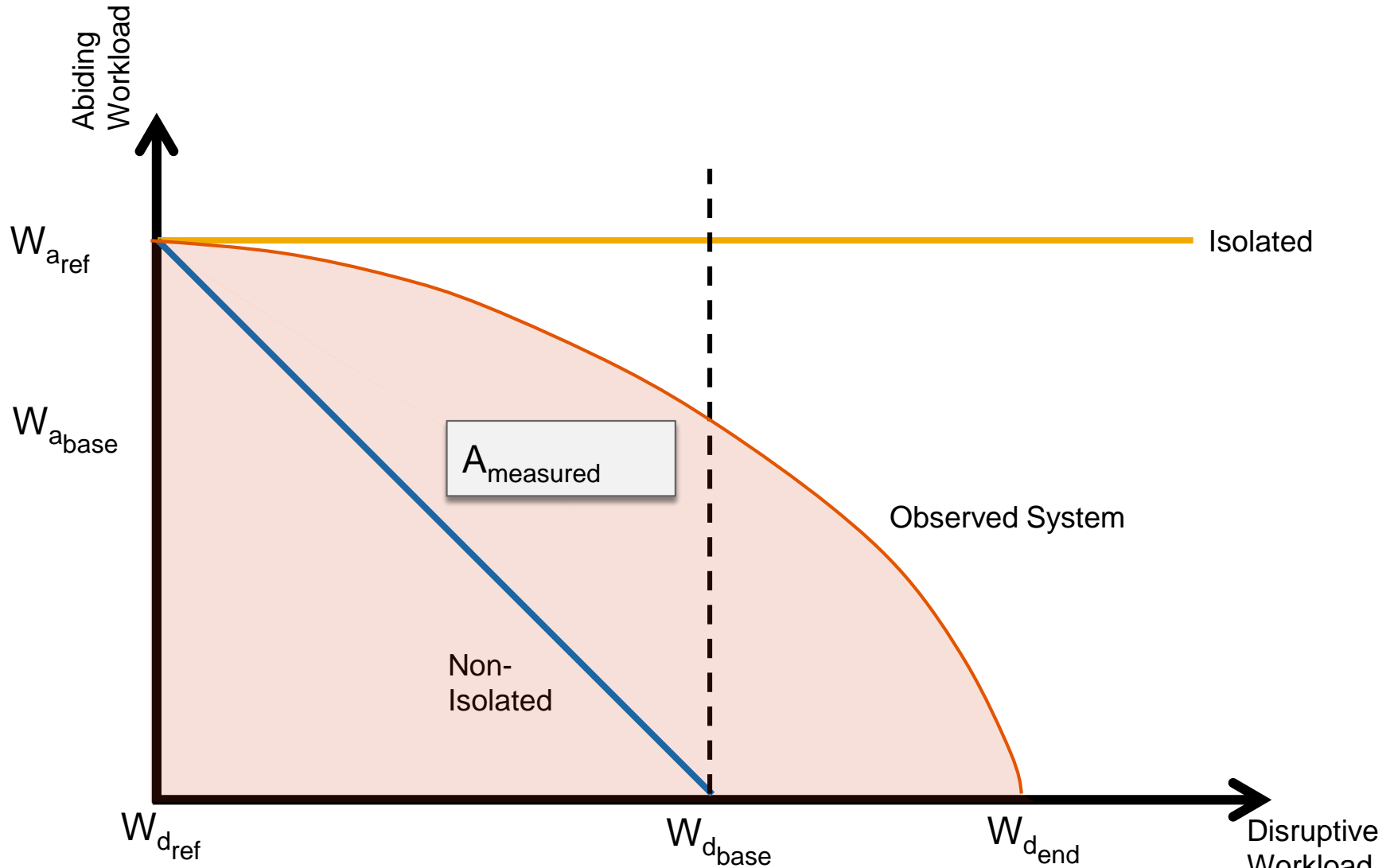
$$I_{base} = \frac{W_{a_{base}}}{W_{a_{ref}}}$$

**Perfectly Isolation = 1**

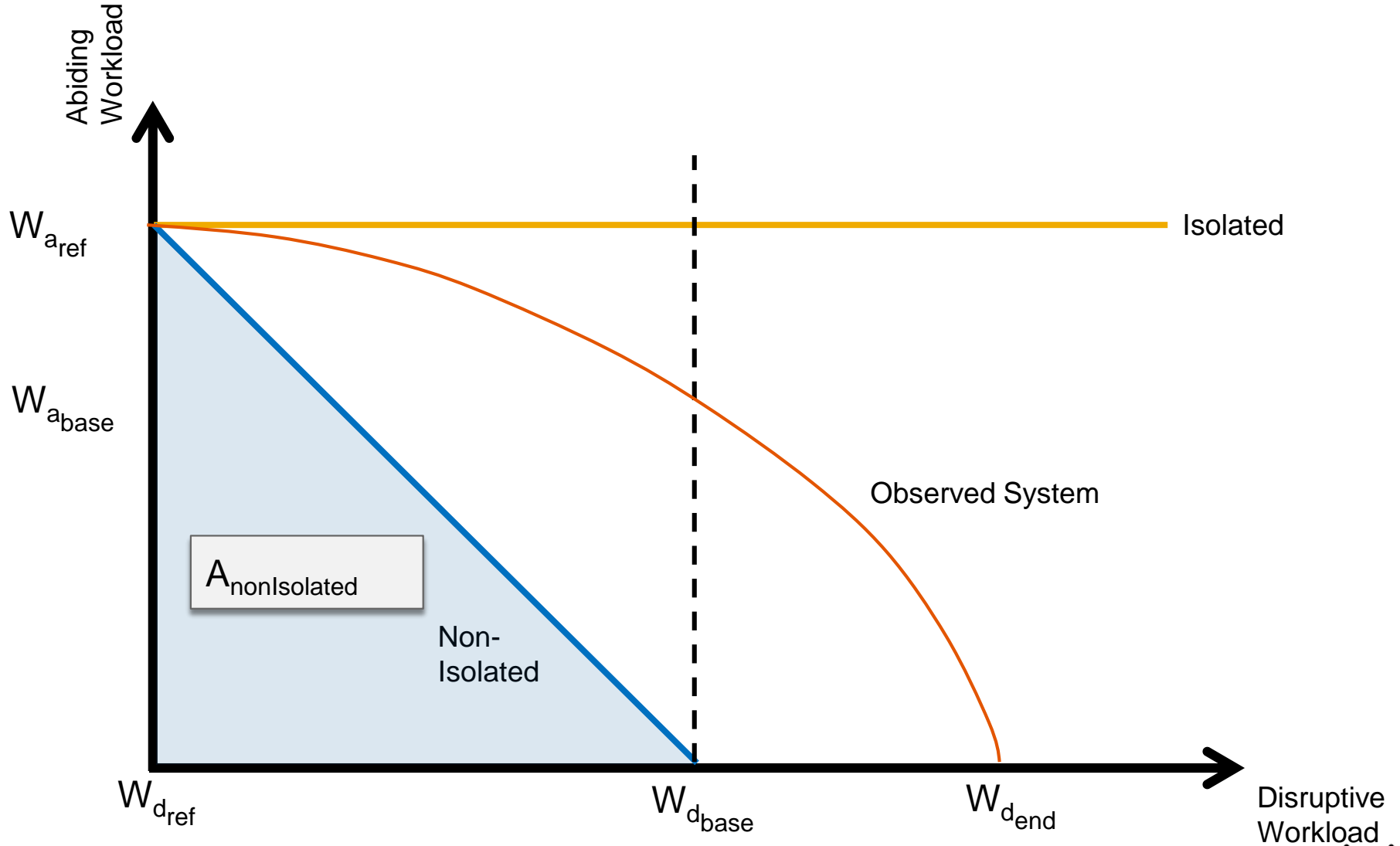
**Non-Isolated = 0**

**Describes the decrease of abiding workload at the point at which a non-isolated systems abiding load is 0.**

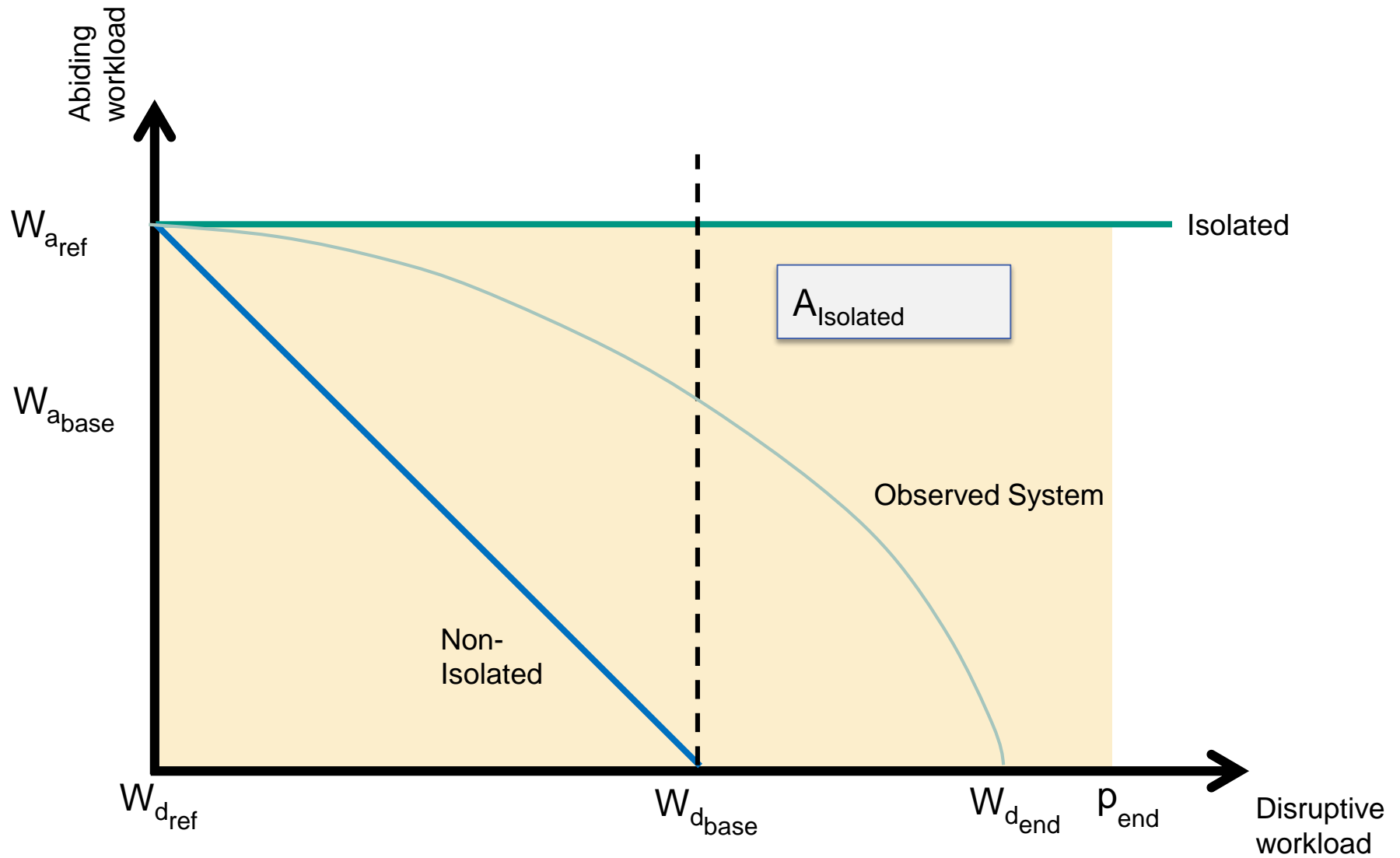
# Metrics Based on Workload Ratio Integrals



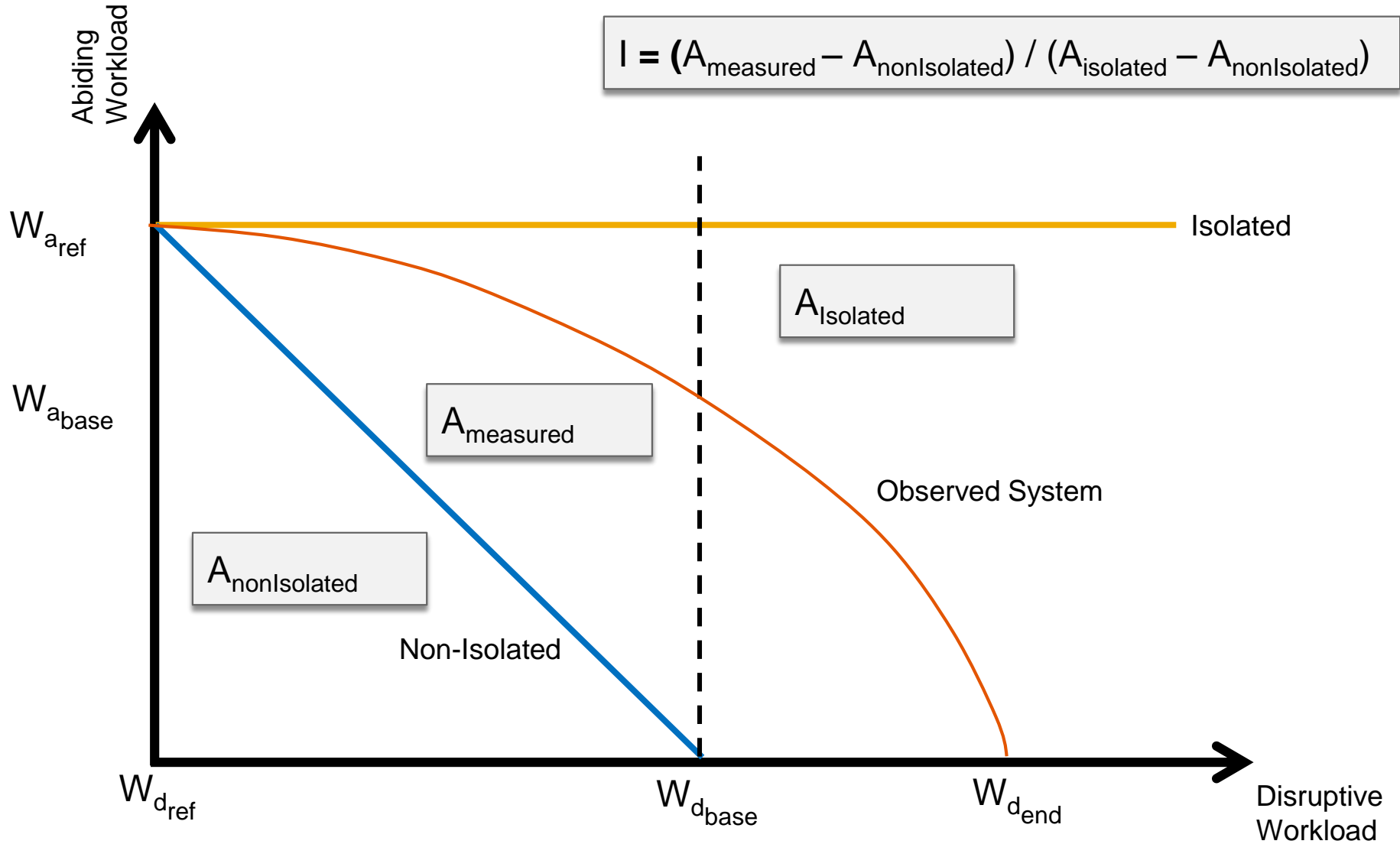
# Metrics Based on Workload Ratio Integrals



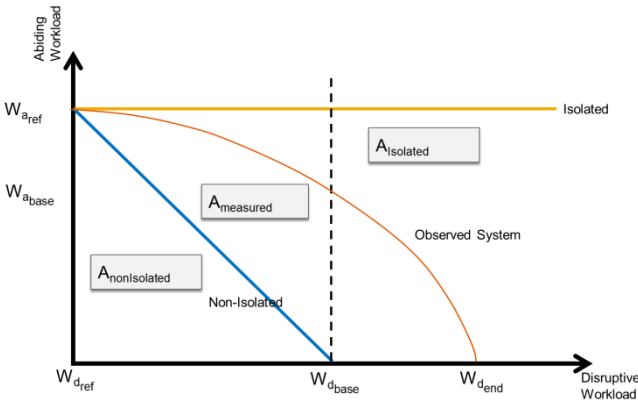
# Metrics Based on Workload Ratio Integrals



# Metrics Based on Workload Ratio Integrals



# Example Metrics: $I_{intBase}$ and $I_{intFree}$



$$I_{intBase} = \frac{\left( \int_{W_{dref}}^{W_{dbase}} f_m(W_d) dW_d \right) - W_{dref}^2/2}{W_{dref}^2/2}$$

Areas within  $W_{dref}$  and  $W_{dbase}$

$$I_{intFree} = \frac{\left( \int_{W_{dref}}^{pend} f_m(W_d) dW_d \right) - W_{dref}^2/2}{W_{dref} \cdot (pend - W_{dref}) - W_{dref}^2/2}$$

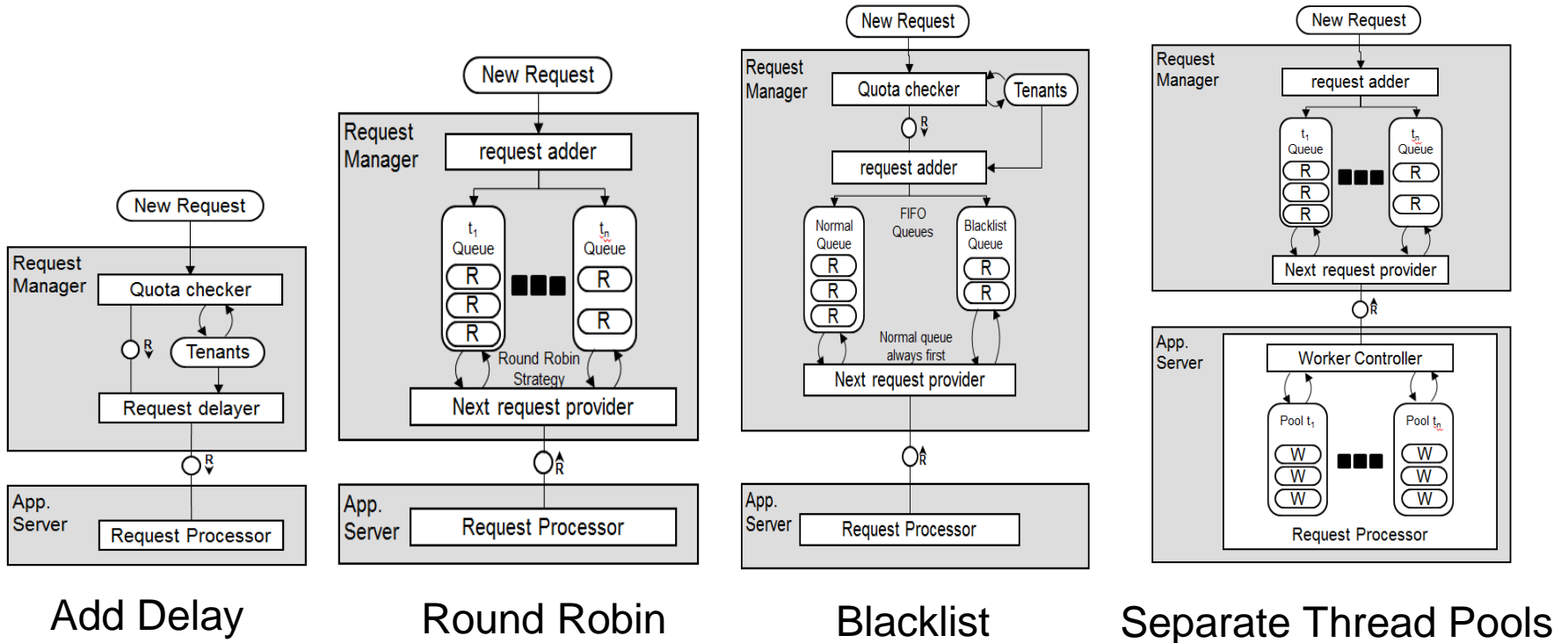
Areas within  $W_{dref}$  and predefined bound.

**Perfectly Isolated = 1**

**Non-Isolated = 0**

**Answers: How much potential has the isolation method to improve?**

# Case Study



R. Krebs, C. Momm and S. Kounev. **Metrics and Techniques for Quantifying Performance Isolation in Cloud Environments.** *Elsevier Science of Computer Programming Journal (SciCo)*, Vol. 90, Part B:116-134, 2014, Elsevier B.V. [ bib | [.pdf](#) ]



# Part III: Intrusion Detection

## Collaboration with

Marco Vieira and Nuno Antunes, University of Coimbra, Portugal

Bryan D. Payne, Department of Security Research, Nebula Inc.

Alberto Avritzer, Siemens Corporate Research, USA

## Main references

A. Milenkoski, B. Payne, N. Antunes, M. Vieira and S. Kounev. **An Analysis of Hypercall Handler Vulnerabilities**. In *Proc. of 25th IEEE Intl. Symp. on Software Reliability Engineering (ISSRE 2014) - Research Track*, Naples, Italy, November 2014. IEEE.

A. Milenkoski, B. Payne, N. Antunes, M. Vieira and S. Kounev. **HInjector: Injecting Hypercall Attacks for Evaluating VMI-based Intrusion Detection Systems** (Poster Paper). In *2013 Annual Computer Security Applications Conf. (ACSAC 2013)*, New Orleans, Louisiana, USA, 2013. [ [.pdf](#) ]

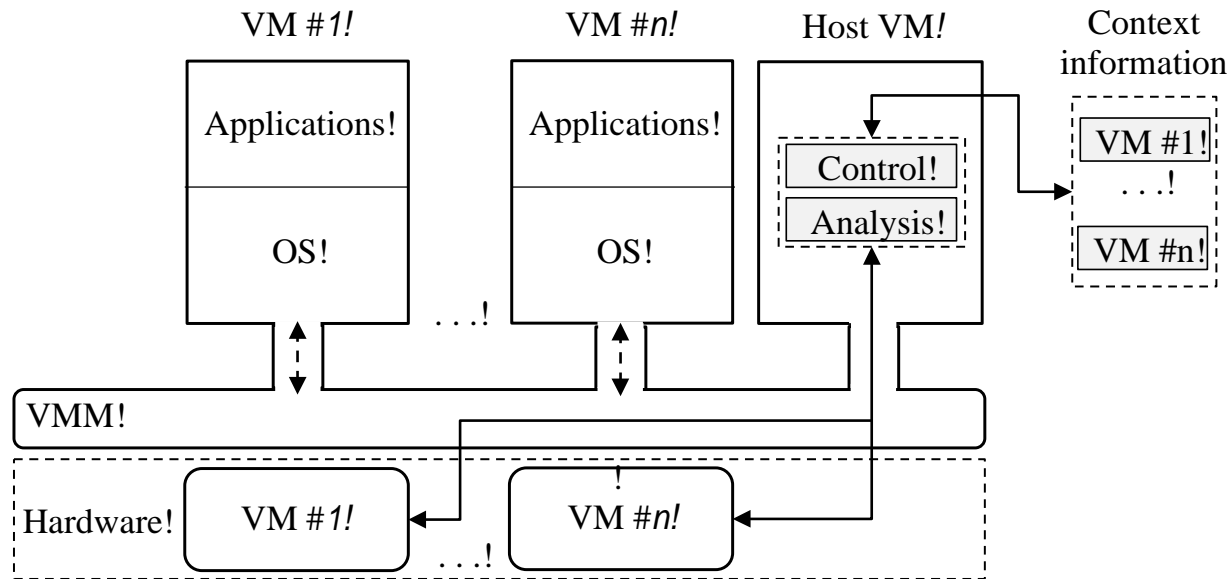
## Further references

A. Milenkoski, S. Kounev, A. Avritzer, N. Antunes and M. Vieira. **On Benchmarking Intrusion Detection Systems in Virtualized Environments**. Technical Report SPEC-RG-2013-002 v.1.0, SPEC Research Group - IDS Benchmarking Working Group, Standard Performance Evaluation Corporation (SPEC), June 2013. [ [.pdf](#) ]

A. Milenkoski, M. Vieira, B. Payne, N. Antunes and S. Kounev. **Technical Information on Vulnerabilities of Hypercall Handlers**. Technical Report SPEC-RG-2014-001 v.1.0, SPEC Research Group - IDS Benchmarking Working Group, Standard Performance Evaluation Corporation (SPEC), August 2014. [ [.pdf](#) ]

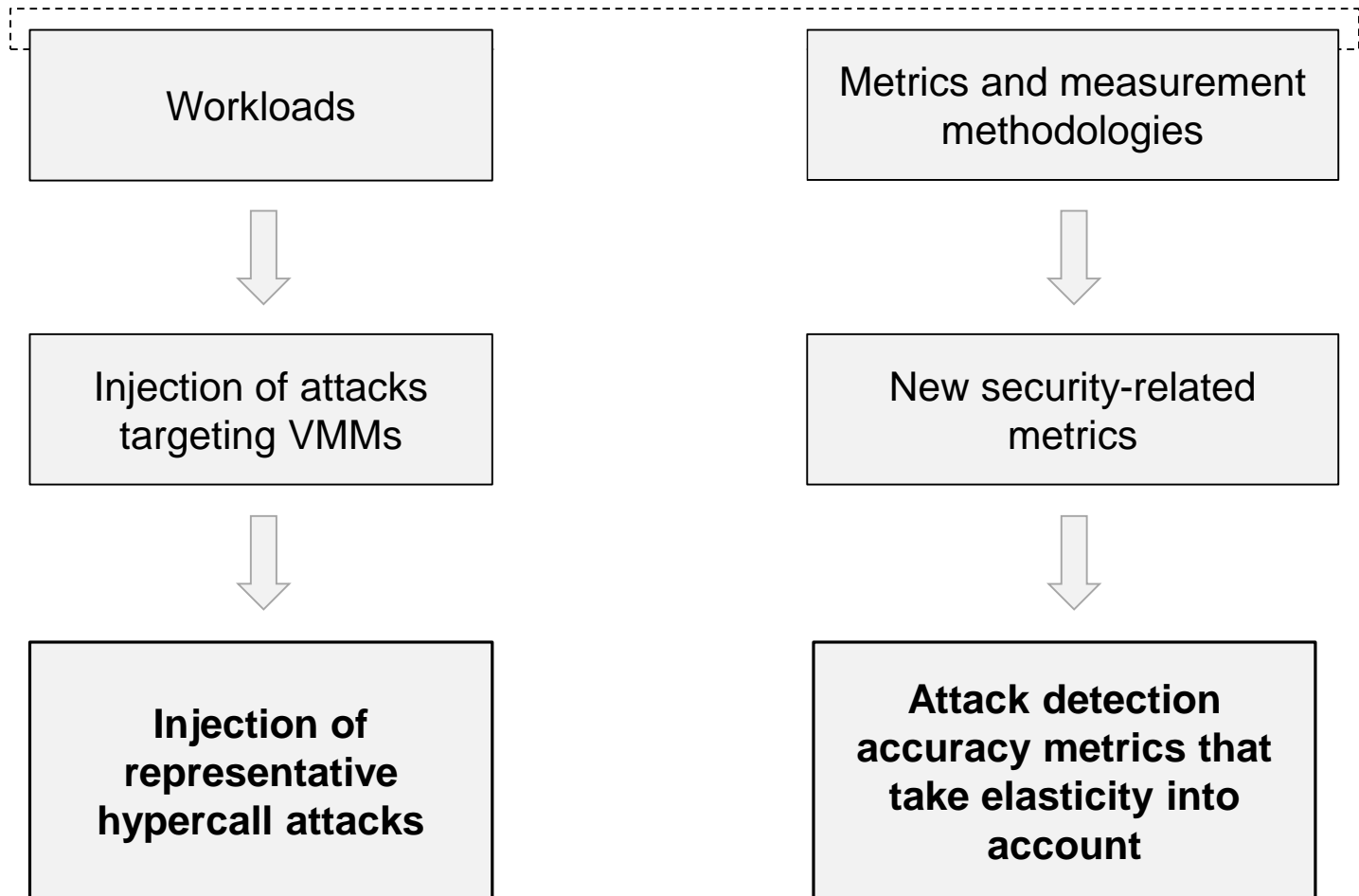
# IDS Evaluation

- Evaluation of intrusion detection systems (IDSes)
  - Enables the comparison of IDSes
  - Enables the improvement of the configuration of a deployed IDS
- IDSes for virtualized environments → many designs possible
  - Network intrusion detection by monitoring the virtual network bridge
  - Host intrusion detection through Virtual Machine Introspection (VMI)



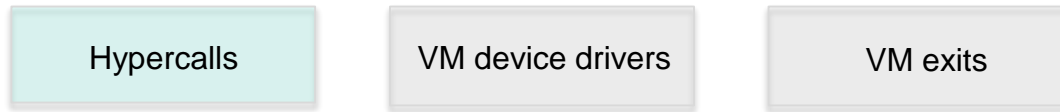
# Focus of our Work

## IDS evaluation in virtualized environments



# Malicious Workloads: Generating Attacks

- Focus: VMMs as attack surfaces
  - Attack scenario: “malicious guest VM attacks the underlying VMM”
    - Attack vectors



- Hypercalls
  - Routines / software traps invoked by kernels of paravirtualized, or HV with paravirtualized device(s), guest VMs for performing system management operations (e.g., sharing memory pages)



- Vulnerabilities in VMMs' hypercall handling routines are **critical!**

# Malicious Workloads: Generating Attacks

- Defining representative/realistic attack scenarios
  - Attack models
    - Identify characteristics of hypercall attacks (e.g., specific hypercall parameter values, hypercall order, ....)
  - No attack scripts/proof-of-concept code available ...
    - ... however, patches are available!
- Approach:

1. Select a set of hypercall vulnerabilities

2. Reverse-engineer the patches of the selected vulnerabilities

2.1 Develop proof-of-concept code

3. Characterize hypercall attacks

# Malicious Workloads: Generating Attacks

- *Artificial injection* of hypercall attacks based on *representative attack models*
  - Reason: Lack of publicly available attack scripts

- Attack models

1. Analysis of relevant CVE reports

2. Identification of patterns of VM activities

3. Categorization of VM activity patterns into attack models

- Attack patterns

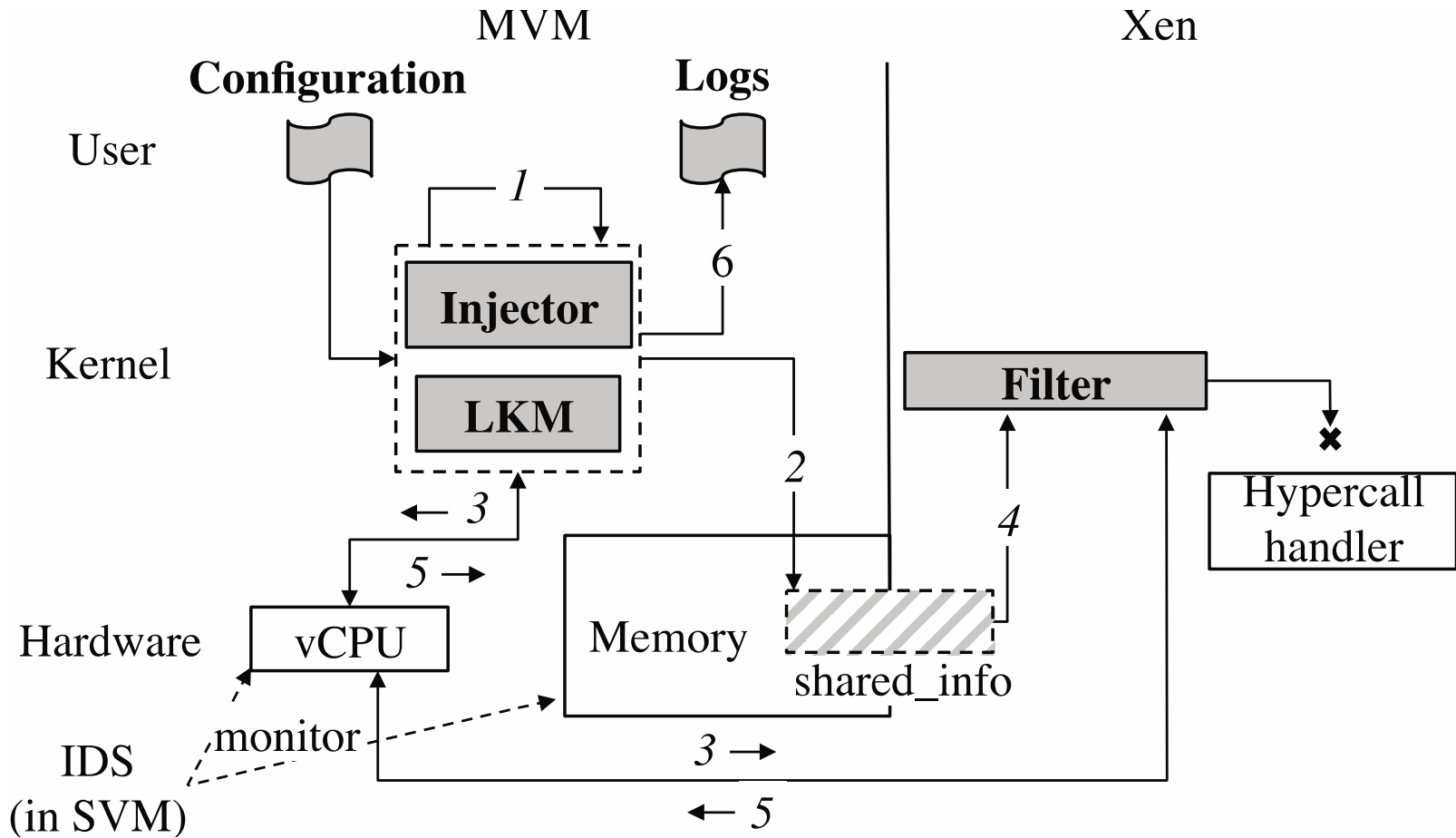
1. Invoking hypercalls from irregular call sites

2. Hypercalls with anomalous parameter values a) outside the valid value domains, or b) crafted for exploiting specific vulnerabilities (not necessarily outside the valid value domains)

3. A series of hypercalls in irregular order, including repetitive execution of a single or multiple hypercalls

More later ...

# HInjector: Framework for Injecting Hypercall Attacks



# Field Study on Hypercall Vulnerabilities

- Goals
  - Characterization and classification of hypercall vulnerabilities
  - Identification of causes of hypercall vulnerabilities
  - Provide technical information on hypercall vulnerabilities
  
- Benefits
  - Can we prevent future vulnerabilities?
    - Hypercall programming practices
    - Vulnerability discovery techniques
  - Can we detect and prevent the exploitation of existing vulnerabilities?
    - Hypercall attack detection and prevention mechanisms



# Field Study on Hypercall Vulnerabilities

CVE	Hypercall	Vulnerable Platform
CVE-2012-3497 / CVE-2012-6036	tmem_op	>= Xen 4.0.x
CVE-2012-5513	memory_op	< Xen 4.1.4
CVE-2008-3687	flask_op	< Xen 3.3
CVE-2013-0154	mmu_update	Xen 4.2.x
<b>CVE-2013-1964</b>	<b>grant_table_op</b>	<b>Xen 4.1.x – 4.1.5</b>
<b>CVE-2012-4539</b>	<b>grant_table_op</b>	<b>Xen 4.1.x – 4.1.4</b>
<b>CVE-2012-5525</b>	<b>mmuext_op</b>	<b>Xen 4.2.x</b>
<b>CVE-2012-5515</b>	<b>memory_op</b>	<b>Xen 3.4.x – 4.1.4</b>
<b>CVE-2012-3494</b>	<b>set_debugreg</b>	<b>&lt; Xen 4.1.4 (4.1 ser.), Xen 4.2.0 (4.2 ser.)</b>
<b>CVE-2012-3496</b>	<b>memory_op</b>	<b>Xen 3.9.x – 4.1.4</b>
<b>CVE-2012-5514</b>	<b>memory_op</b>	<b>Xen 3.4.x – 4.1.4</b>
<b>CVE-2012-3495</b>	<b>physdev_op</b>	<b>Xen 4.1.x</b>
CVE-2013-0154	mmuext_op	Xen 4.2.x
<b>CVE-2012-5513</b>	<b>memory_op</b>	<b>Xen 4.1.x</b>
CVE-2013-4553	domctl	> Xen 3.4.x
CVE-2013-0151	hvm_op	Xen 4.2.x
<b>CVE-2013-4494</b>	<b>grant_table_op</b>	<b>All versions of Xen up to the current date</b>
<b>CVE-2012-5510</b>	<b>grant_table_op</b>	<b>&lt; Xen 4.1.4 (4.1 ser.), Xen 4.2.0 (4.2 ser.)</b>
CVE-2013-3898	unknown	Windows 8 / Windows Server 2012

# Observations

- Errors causing hypercall vulnerabilities
  - Implementation errors (missing value validation, incorrect value validation, and incorrect implementation of inverse procedures)
  - Hypervisor design errors
- Most implementation errors are missing value validation errors
  - Internal variables (e.g., return codes) !
  - Eliminating missing value validation errors by adding program code verifying variable values →
    - Reduces hypercall execution speed → increased frequency of continuations → performance overhead →
    - Programming practices for boosting hypercall execution speed → vulnerabilities (e.g., CVE-2012-5535)

# Summary

- Pressure to raise efficiency by sharing IT resources
- Resource sharing poses challenges
- Service “dependability” → major distinguishing factor
- Need for reliable benchmarks:
  - metrics, workloads and measurement methodologies
- **Multiple metrics** needed to understand system behavior
- **Choice of workloads** is critical for fair comparisons!

www.VADLO.com

***"It is easy to lie  
with statistics.  
It is hard to tell the truth  
without statistics."***

**-- Andrejs Dunkels**

"I can prove it or disprove it! What do you want me to do?"

# Thank You!

[skounev@acm.org](mailto:skounev@acm.org)

<http://se.informatik.uni-wuerzburg.de>