Quantitative Evaluation of Service Dependability in Shared Execution Environments

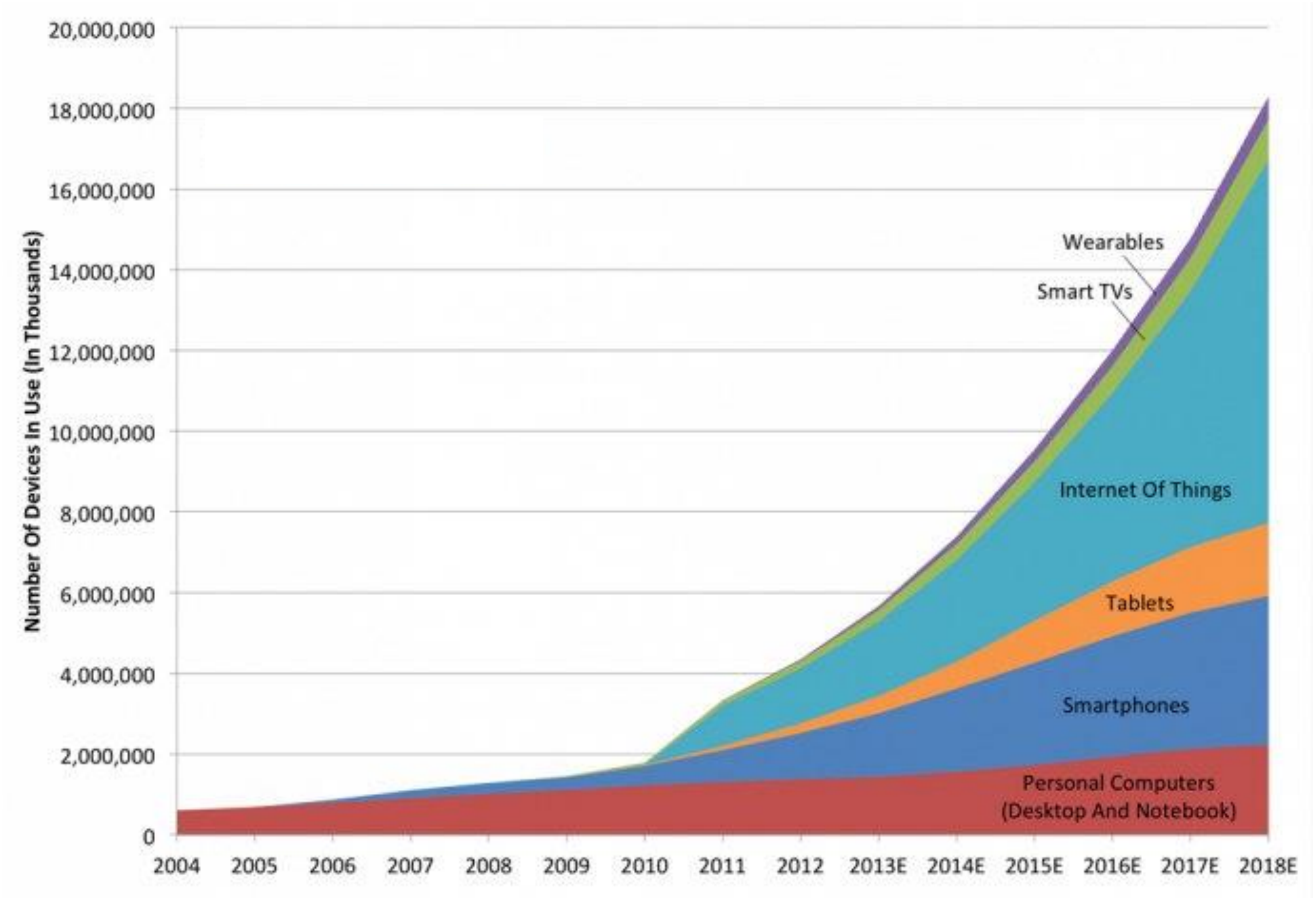
Samuel Kounev

Chair of Software Engineering
University of Würzburg

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

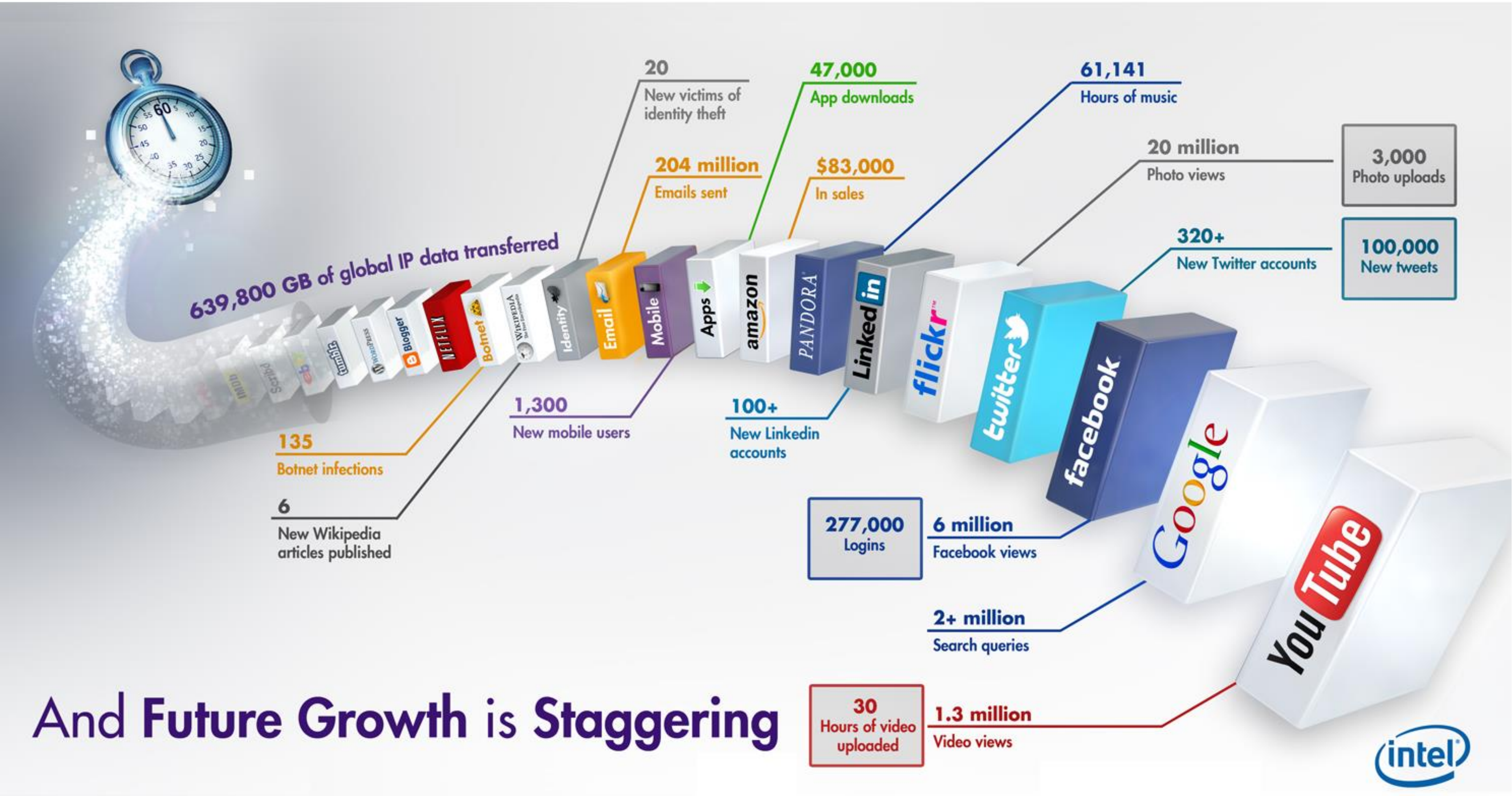
Keynote talk, QEST 2014, Florence, Italy, Sept. 10, 2014

Explosion of IT Service Clients



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

One Internet Minute



25,000+ new Apps added every month

Source: Intel, March 2012 (http://scoop.intel.com/what-happens-in-an-internet-minute)

Growing Data Centers

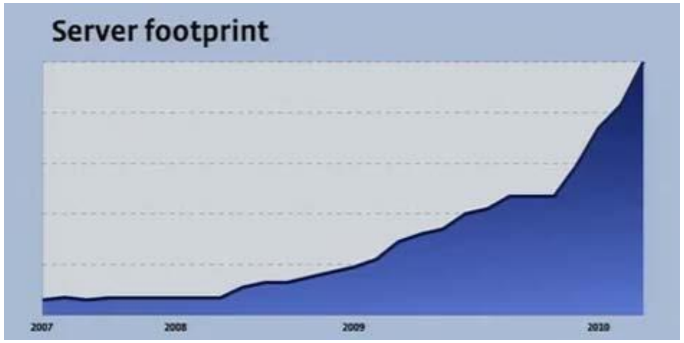
Maiden, North Carolina (Apple)
46 000 m^2

San Antonio (Microsoft)
43 000 m^2

Prineville, Oregon (Facebook)
28 000 m^2

Chicago (Digital Realty)
100 000 m^2

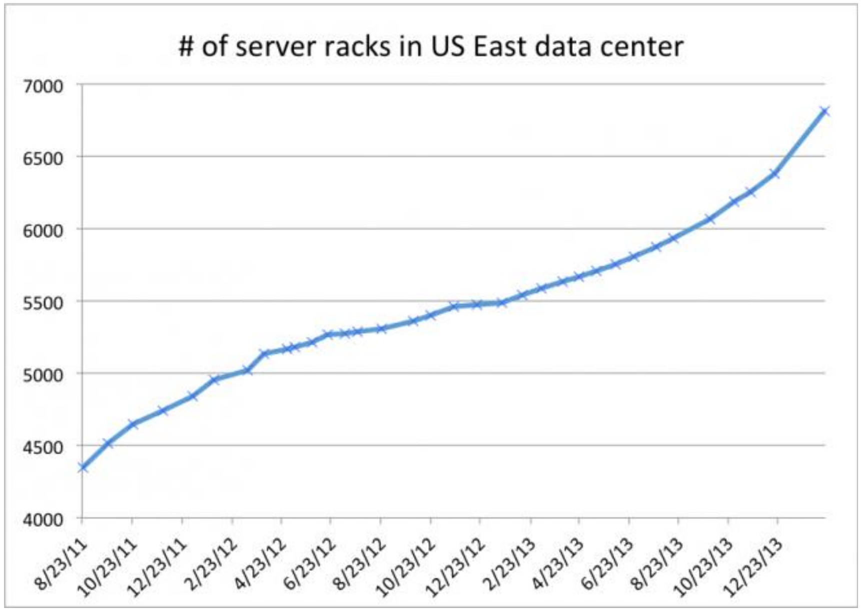
Growing Number of Servers



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)
- …

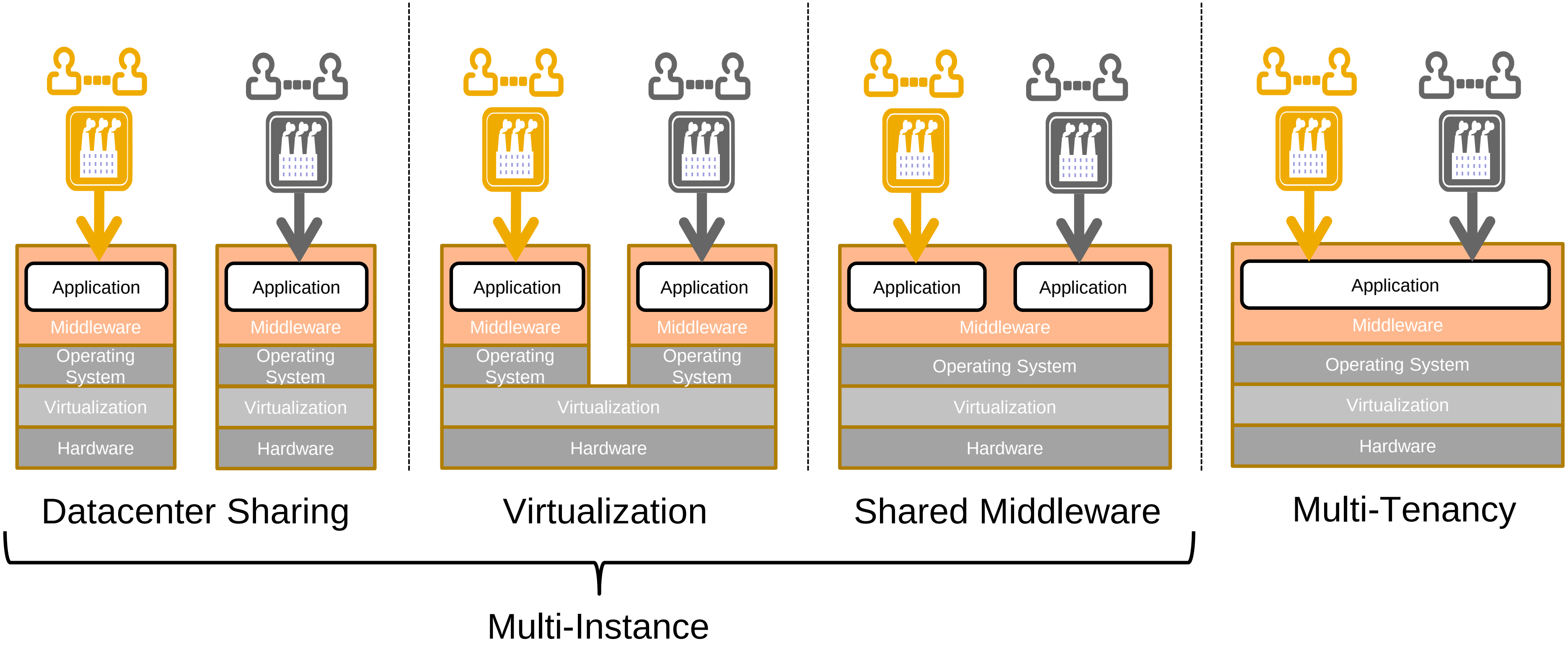
Source: http://www.datacenterknowledge.com



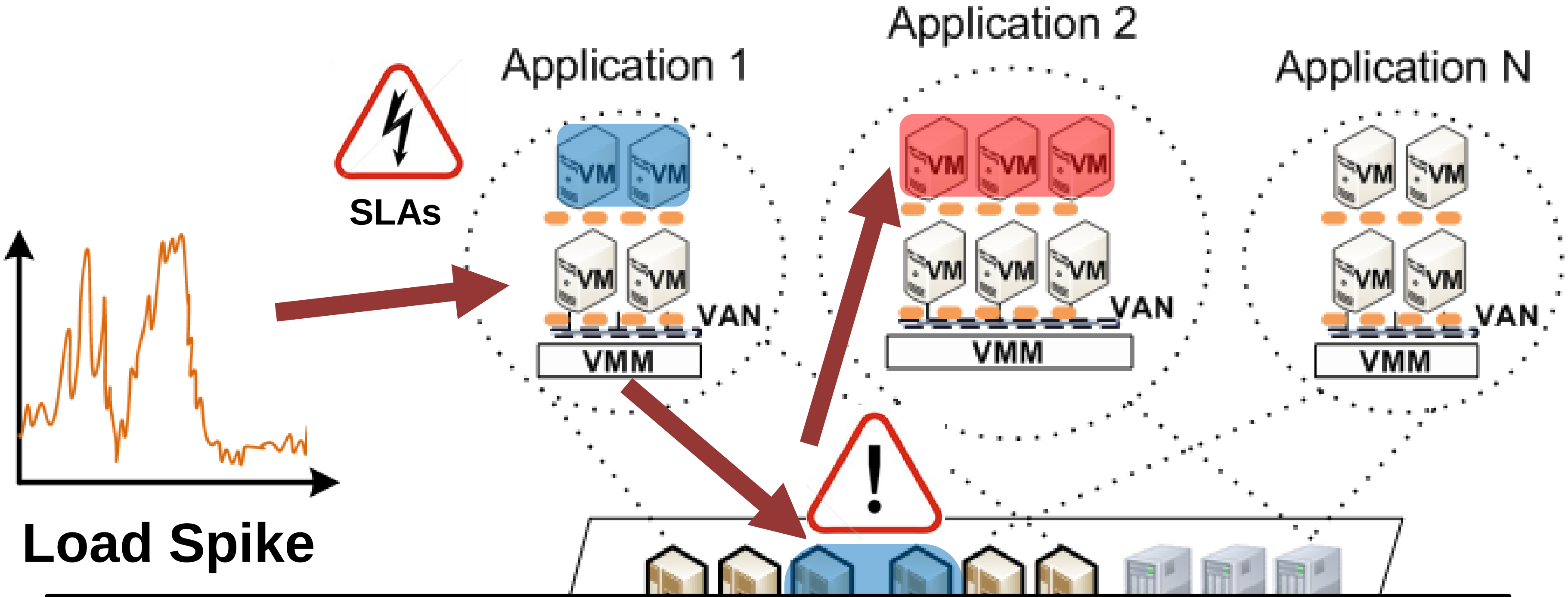
Amazon’s Virginia region [Src: Wired.com]

Increasing Pressure to Raise Efficiency

- Proliferation of shared execution environments
- Different forms of resource sharing (hardware and software)
  - Network, storage, and computing infrastructure
  - Software stacks



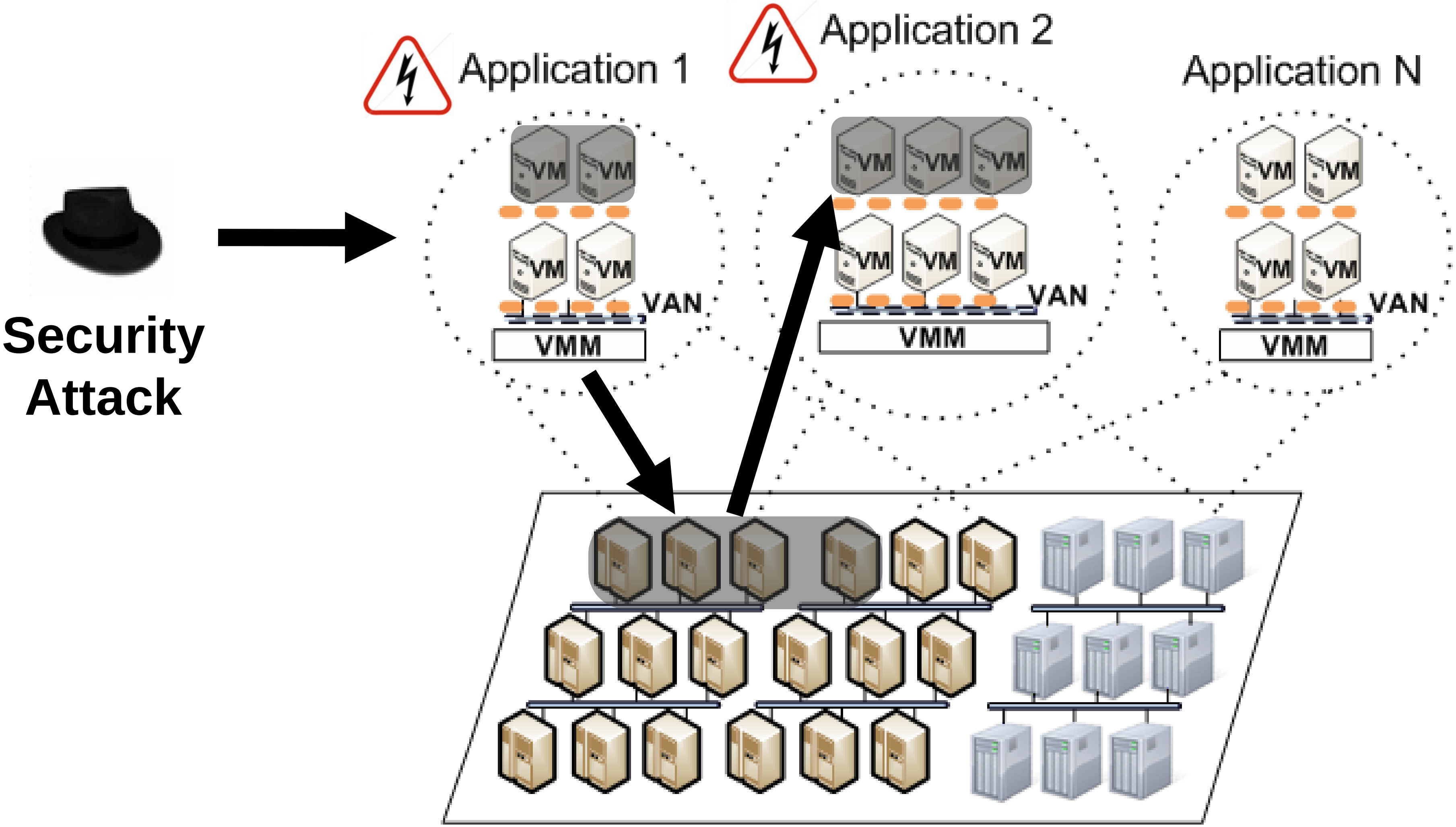
Challenges



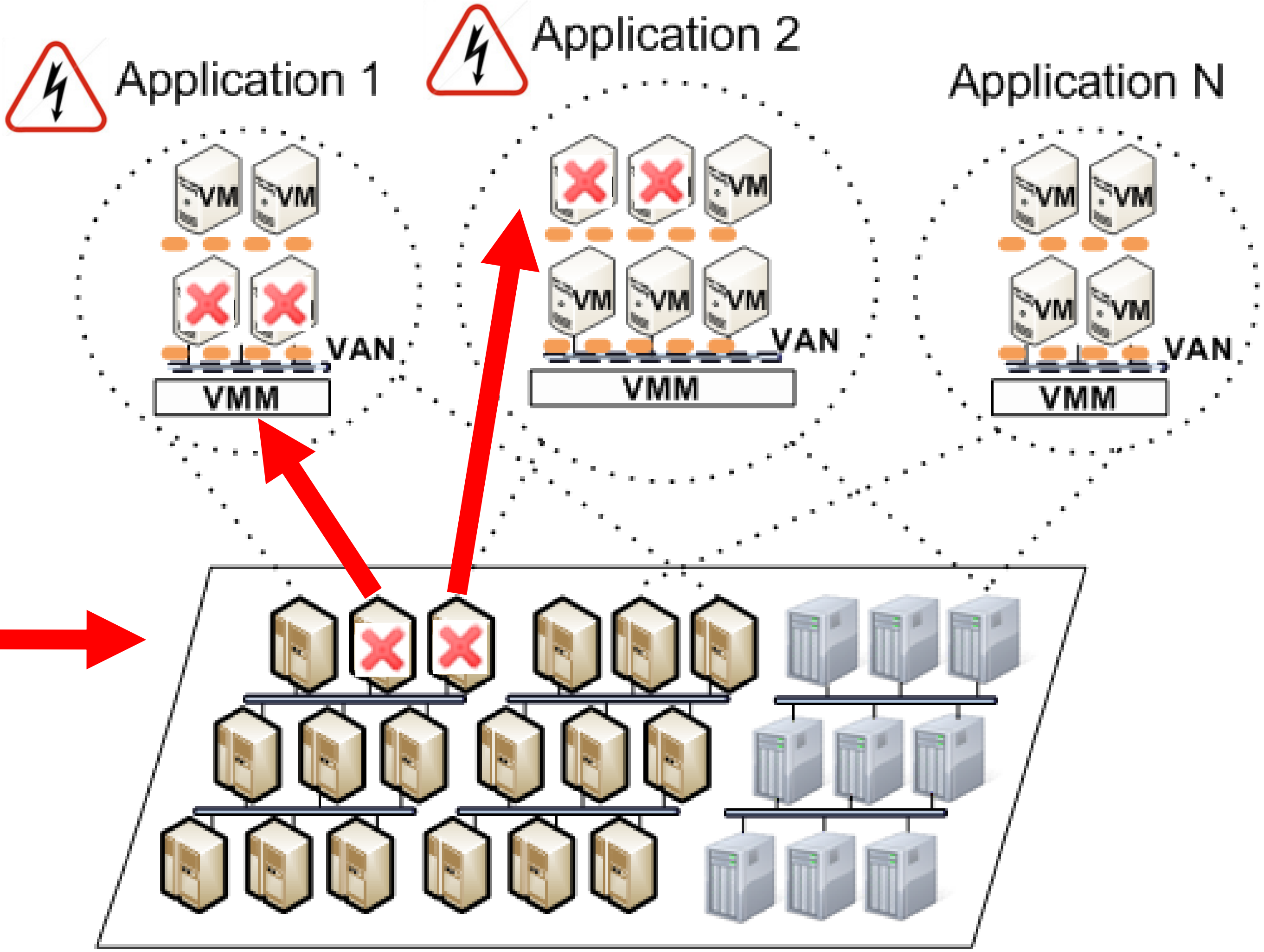
Expand / shrink resources on-the-fly

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

Challenges


Application 1
Application 2
Application N
Security Attack
VM
VAN
VMM

Challenges



Hardware Failure

Consequences

- Increased system complexity and dynamics
- Diverse vulnerabilities due to resource sharing
- Inability of to provide **dependability** guarantees
    - Availability, reliability (+ security, performance, …)

  ⇨ **Major distinguishing factor between service offerings**
- Lack of reliable benchmarks and metrics

"You can't ***control*** *what you can't measure?"* (DeMarco)

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

What is Needed?

Reliable Metrics

- What exactly should be measured and computed?

Representative Workloads

- For which scenarios and under which conditions?

Sound Measurement Methodology

- How should measurements be conducted?

“*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)

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
- Multi-tenant applications

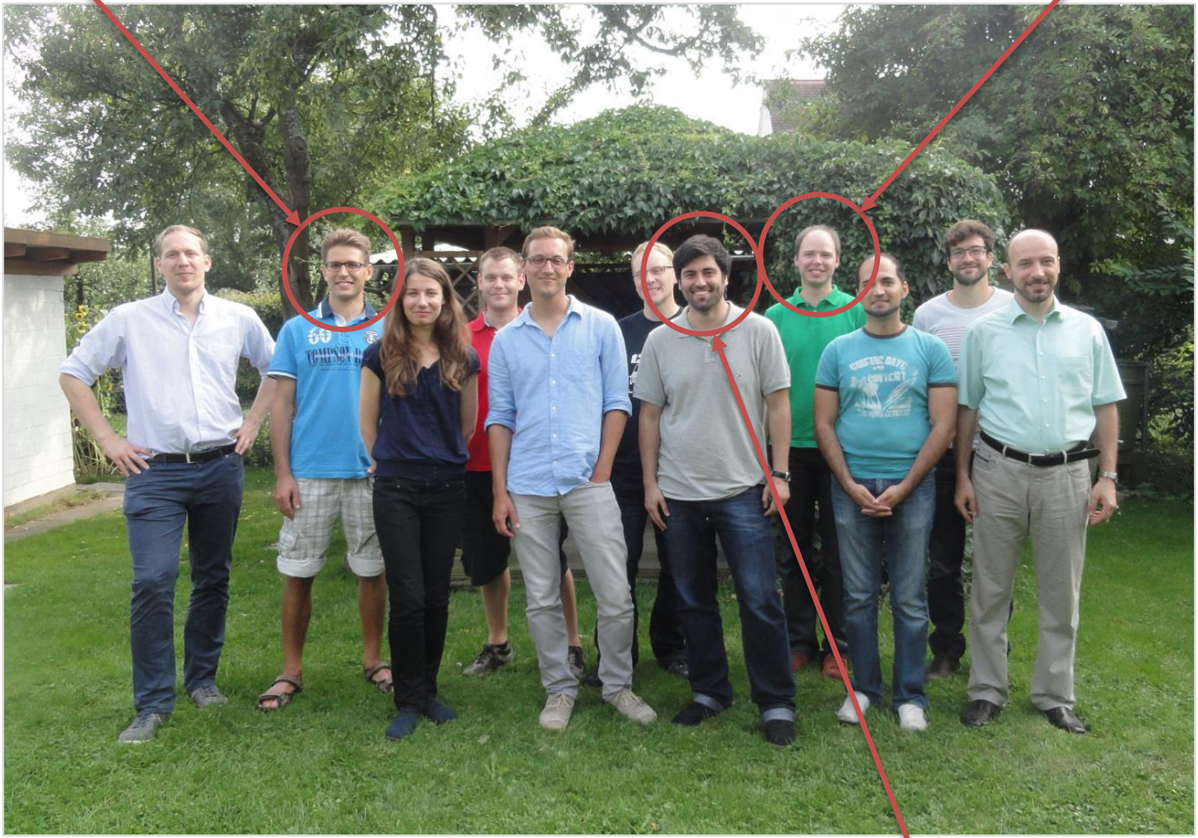
HAVE YOU TESTED YOUR CODE UNDER STRESS?
NO, BUT I'VE WRITTEN IT UNDER STRESS

[geek & poke]

Credits

Nikolas Herbst + MSc students (elasticity)

Rouven Krebs + MSc students (performance isolation)



Aleksandar Milenkoski (intrusion detection)

Part I: Resource Elasticity

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

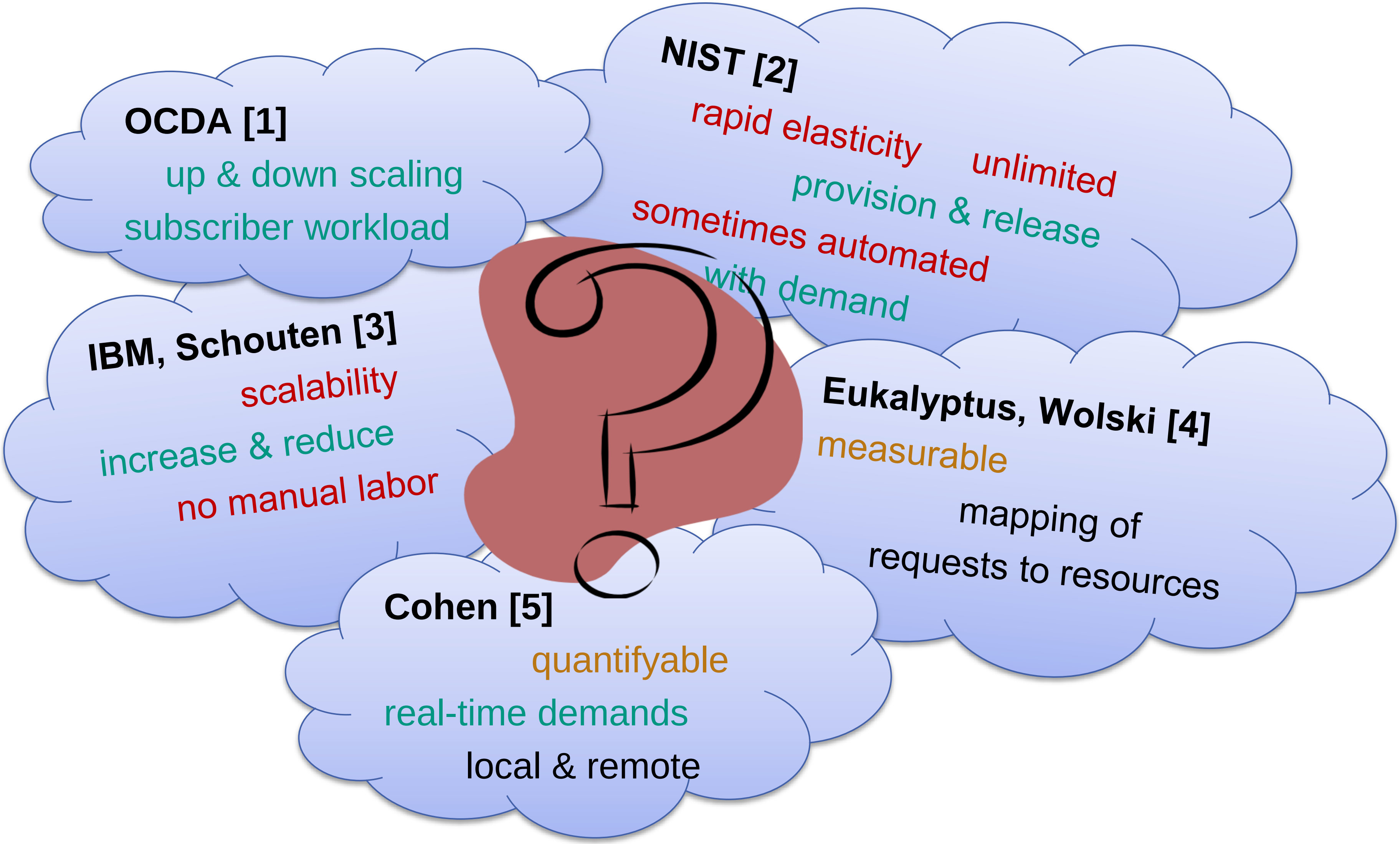
N. Herbst, N. Huber, S. Kounev and E. Amrehn. **Self-Adaptive Workload Classification and Forecasting for Proactive Resource Provisioning**. *Concurrency and Computation - Practice and Experience, John Wiley and Sons, Ltd.*, 26(12):2053-2078, 2014. [DOI | http]

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…



OCDA [1]

up & down scaling

subscriber workload

NIST [2]

rapid elasticity

unlimited

provision & release

sometimes automated

with demand

IBM, Schouten [3]

scalability

increase & reduce

no manual labor

Eukalyptus, Wolski [4]

measurable

mapping of requests to resources

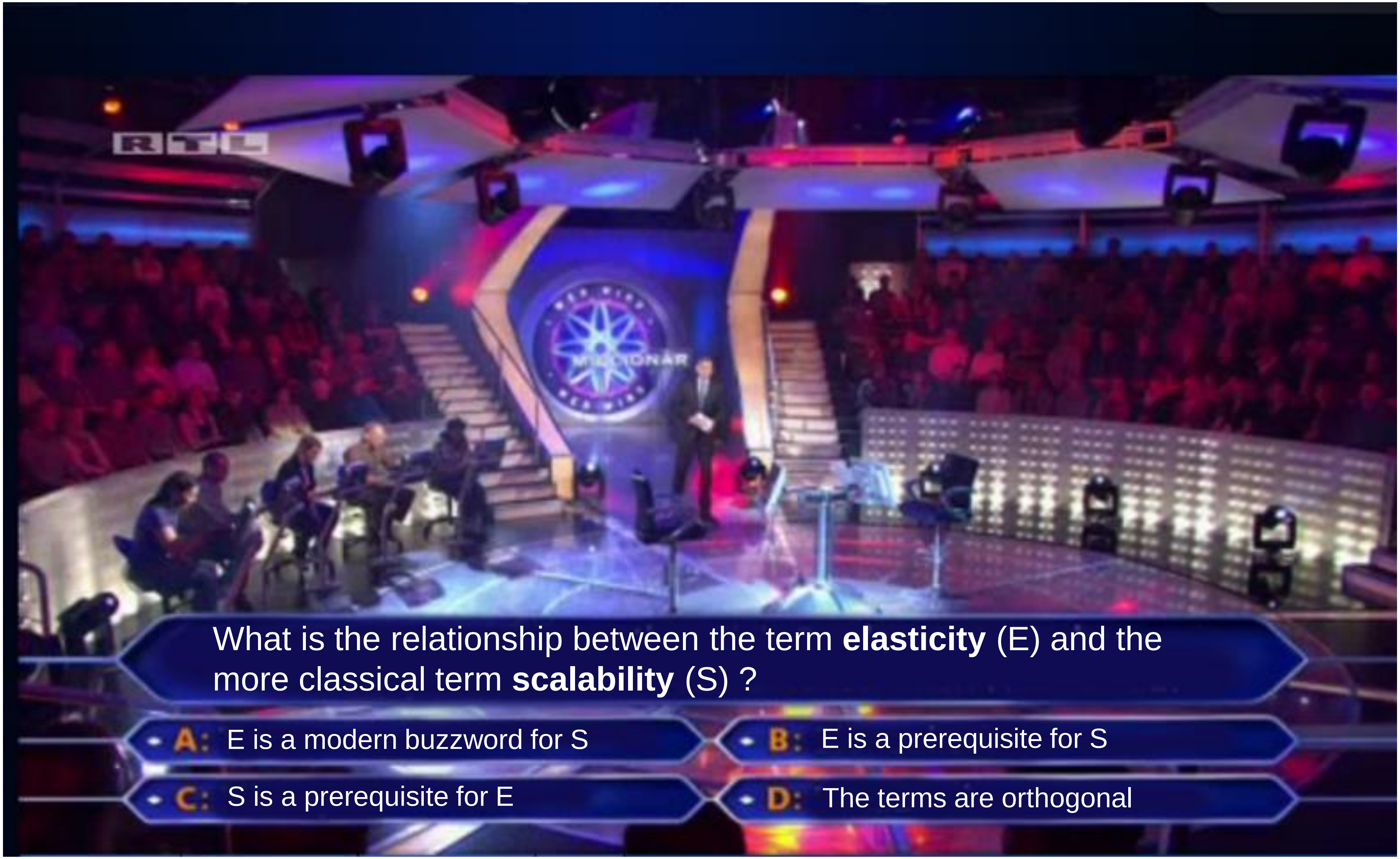
Cohen [5]

quantifyable

real-time demands

local & remote

Elasticity vs. Scalability

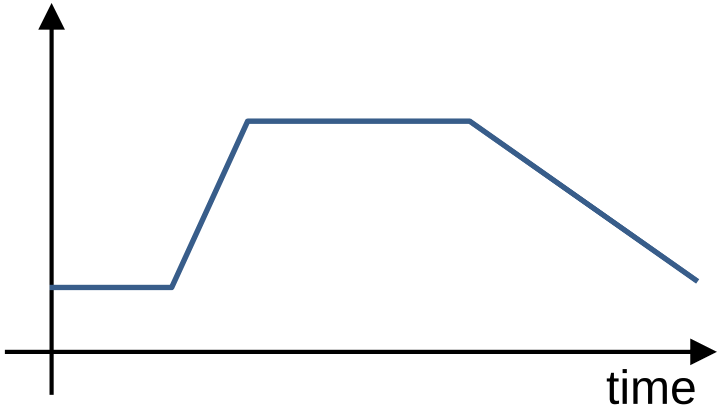


What is the relationship between the term **elasticity** (E) and the more classical term **scalability** (S) ?

- A: E is a modern buzzword for S
- B: E is a prerequisite for S
- C: S is a prerequisite for E
- D: The terms are orthogonal

Elasticity

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



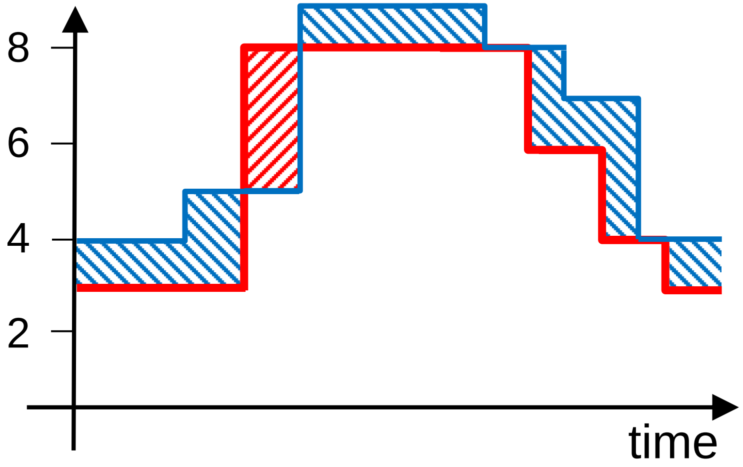
Service Level Objective (SLO)

(e.g., resp. time ≤ 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

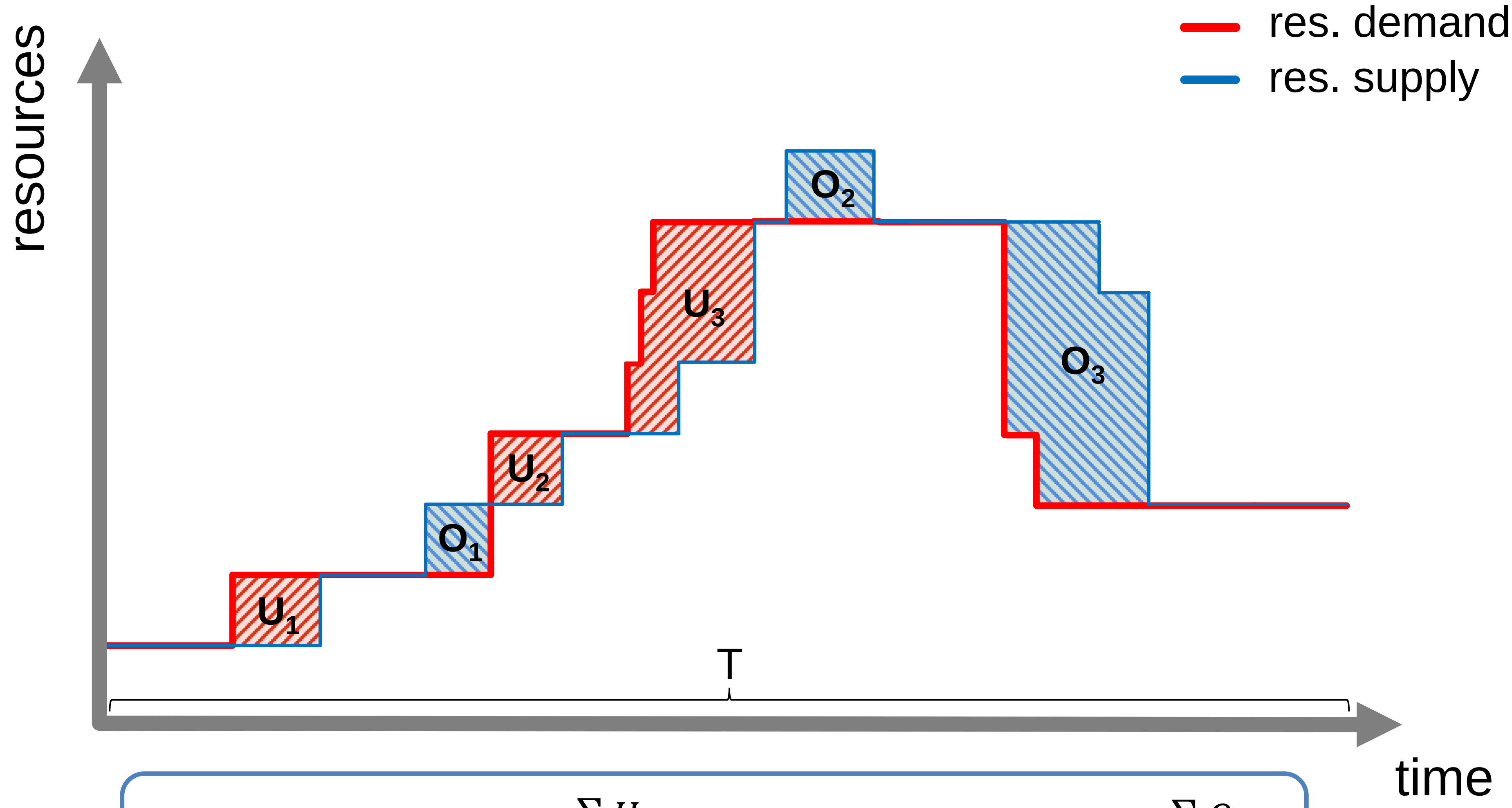
Elasticity

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: 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.

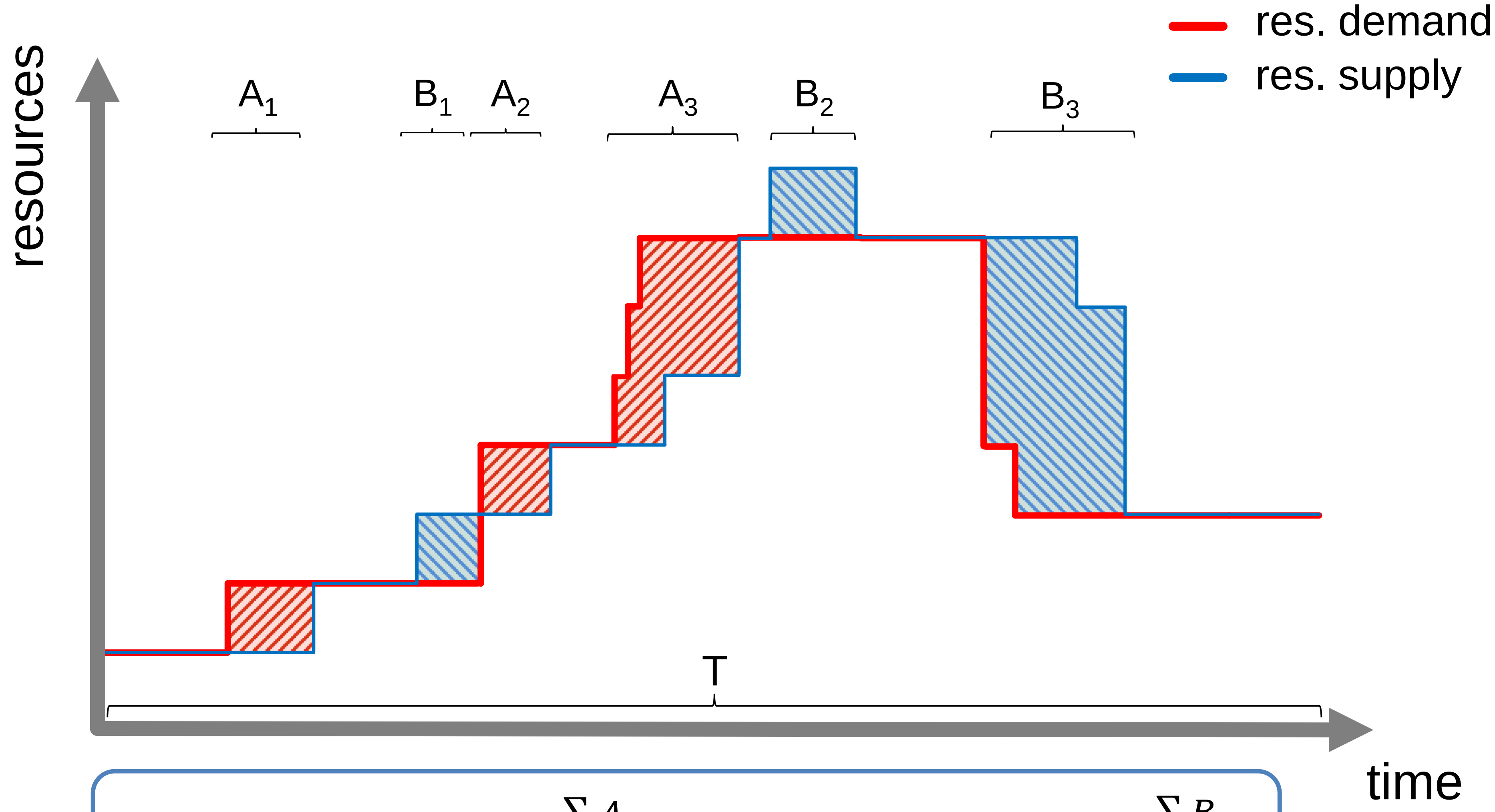
http://en.wikipedia.org/wiki/Elasticity_(cloud_computing)

Metrics: Accuracy



(1) accuracy$_U$: $\frac{\sum U}{T}$ (2) accuracy$_O$: $\frac{\sum O}{T}$

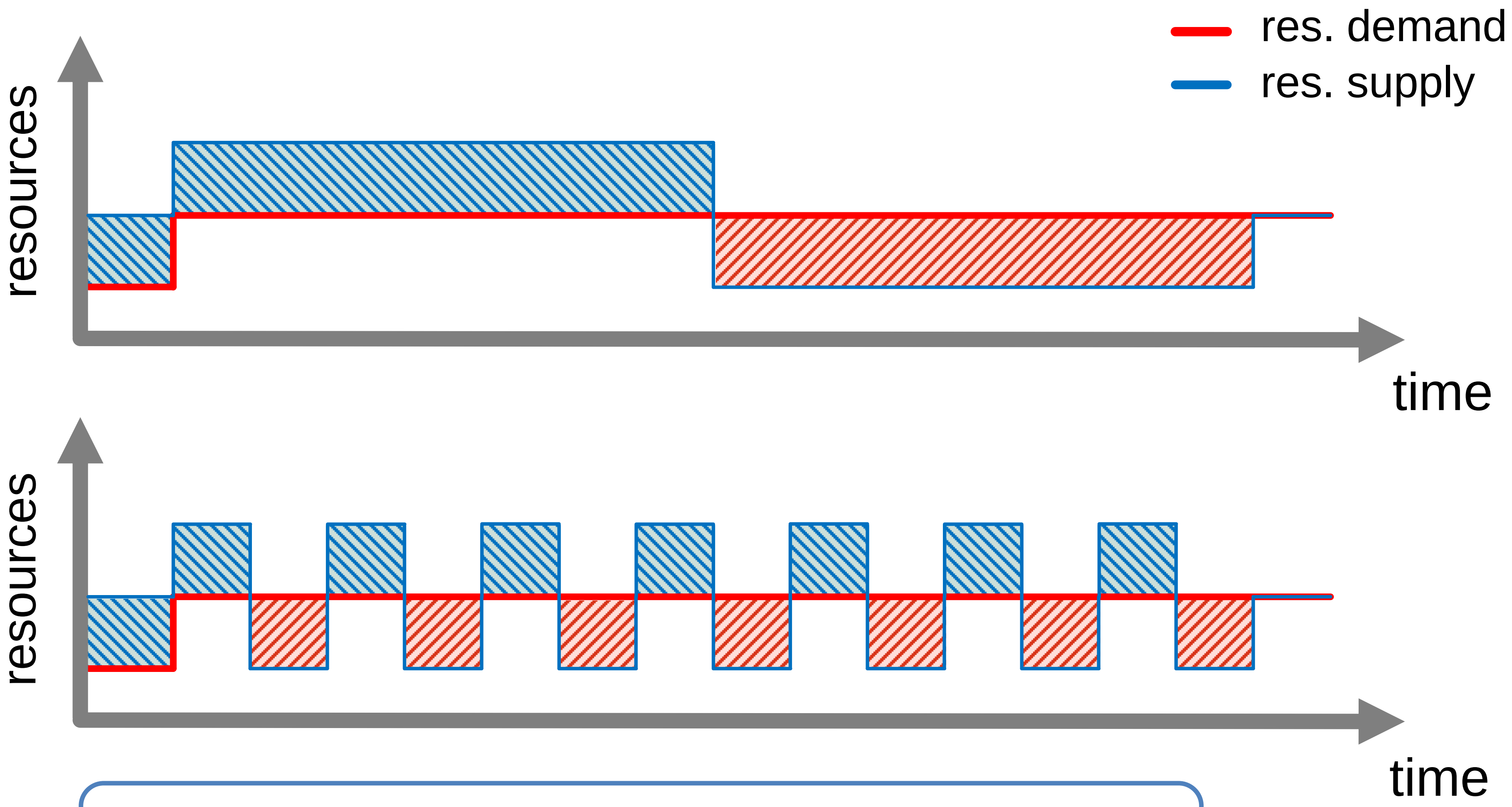
Metrics: Timeshare



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

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

Metrics: Jitter



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

E_D: # demand changes

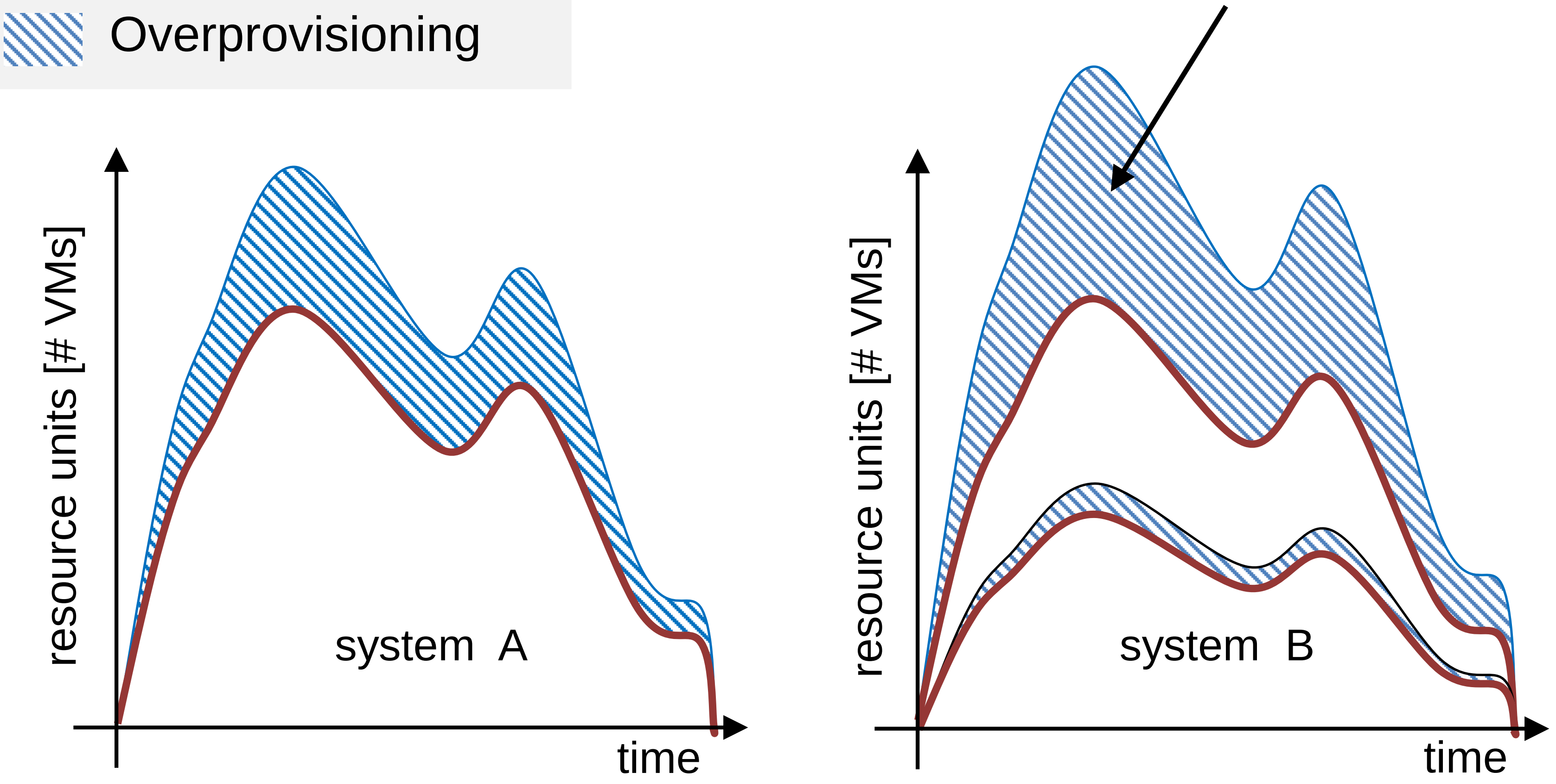
E_S: # supply changes

Elasticity Benchmarking

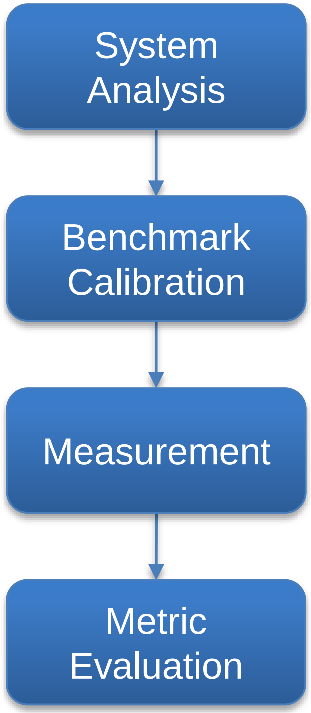
- ━ Resource demand
- ━ Resource supply
- ▨ Overprovisioning

Same user workload on system B

System B at a doubled user workload



Elasticity Benchmarking Approach



System Analysis

Benchmark Calibration

Measurement

Metric Evaluation

Analyze efficiency & scaling behavior of underlying resources

Adjust load profile

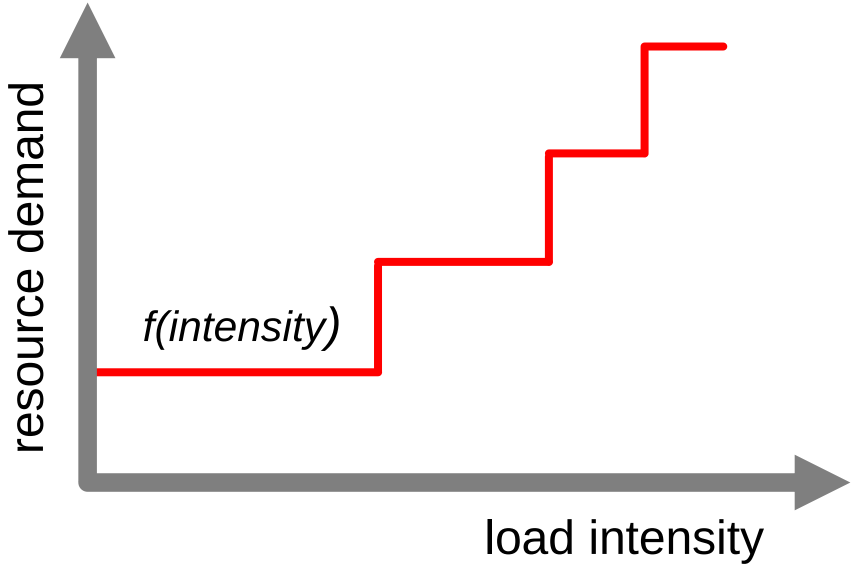
Expose SUT to varying load
&
monitor resource supply & demand

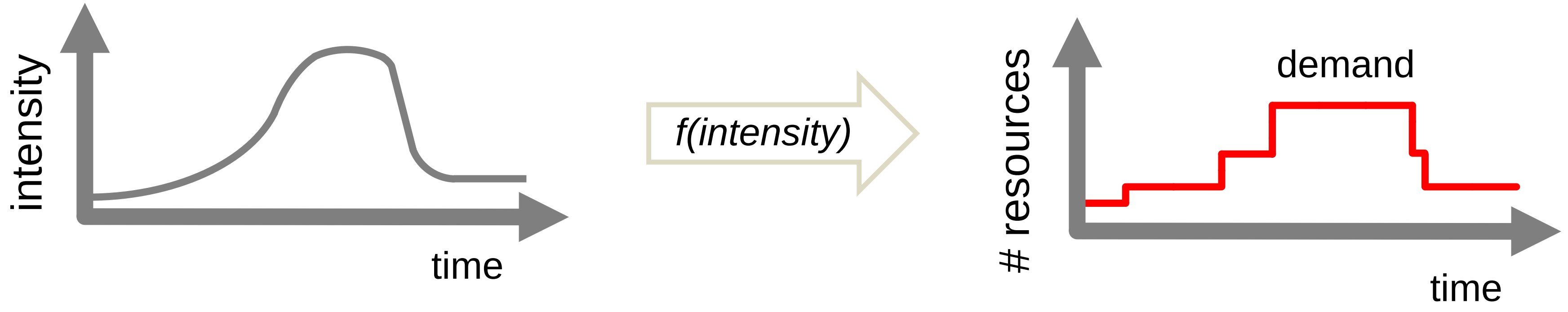
Compute elasticity metrics (accuracy & timing)

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)*.

Step 1: System Analysis

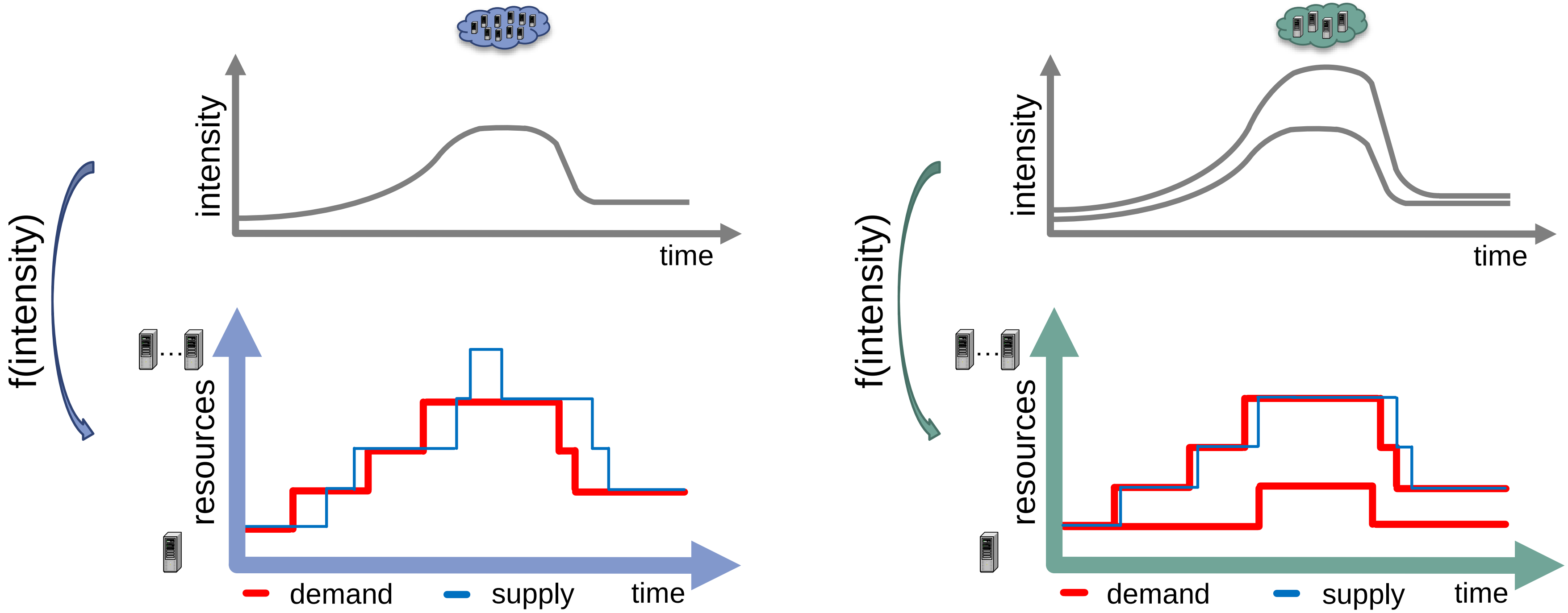
- Evaluate system separately at each scale
- Find maximal intensity that the system can withstand without violating SLO (binary search)
- Derive demand step function: *resourceDemand = f(intensity)*





Step 2: Benchmark Calibration

- Goal: Induce same resource demand on all systems



- Approach: Adjust load intensity profile to overcome
  - Different efficiency of underlying resources
  - Different scalability

Step 3: Measurement

- Requirement: Stress SUT in a representative manner
  - Realistic variability of load intensity
  - Adaptability of load profiles to suit different domains

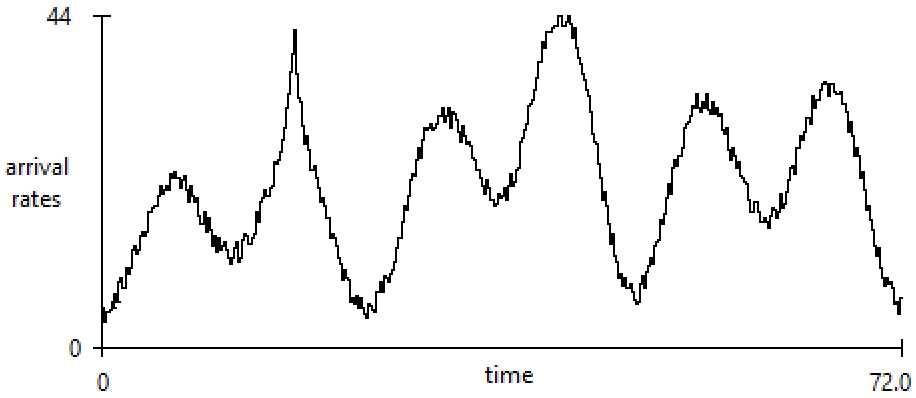
- Approach:
  - Open workload model
  - Model load variations with the LIMBO toolkit

    Facilitates creation of new load profiles
    - Derived from existing traces
    - With desired properties (e.g. seasonal pattern, bursts)
  - Execute load profile using JMeter

    Timer-Plugin delays requests according to timestamp file created by LIMBO

https://github.com/andreaswe/JMeterTimestampTimer

LIMBO: A Tool For Modeling Variable Load Intensities

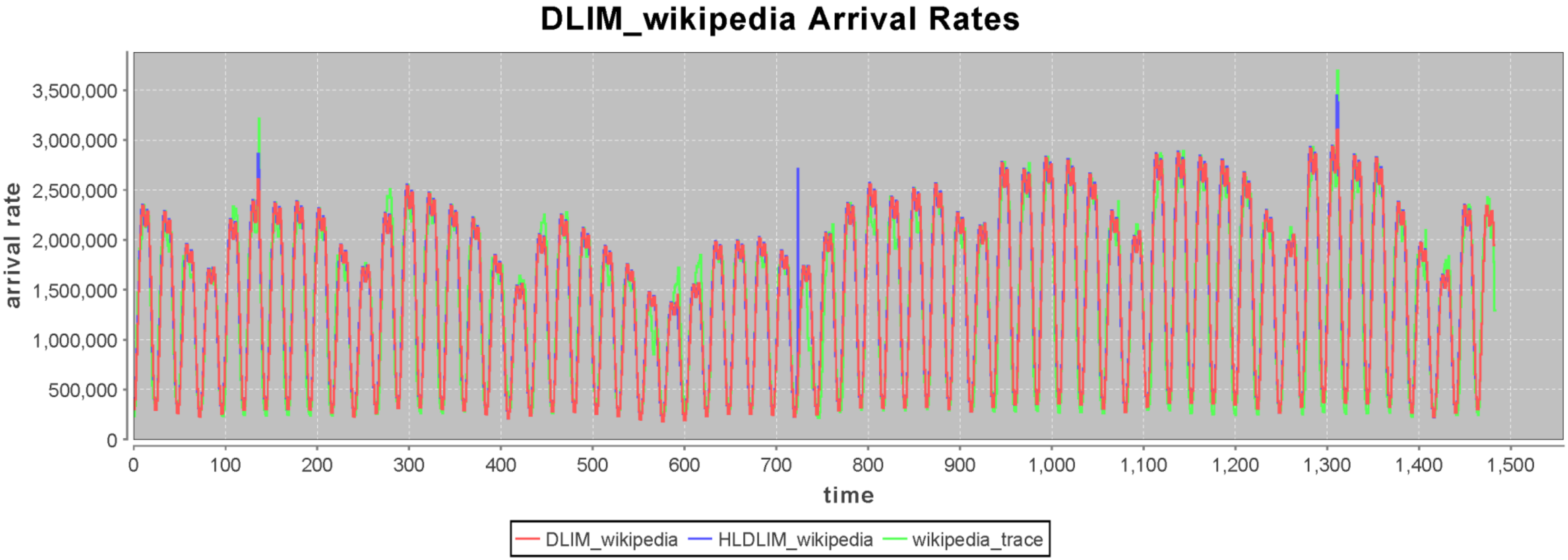


http://www.descartes-research.net/tools/

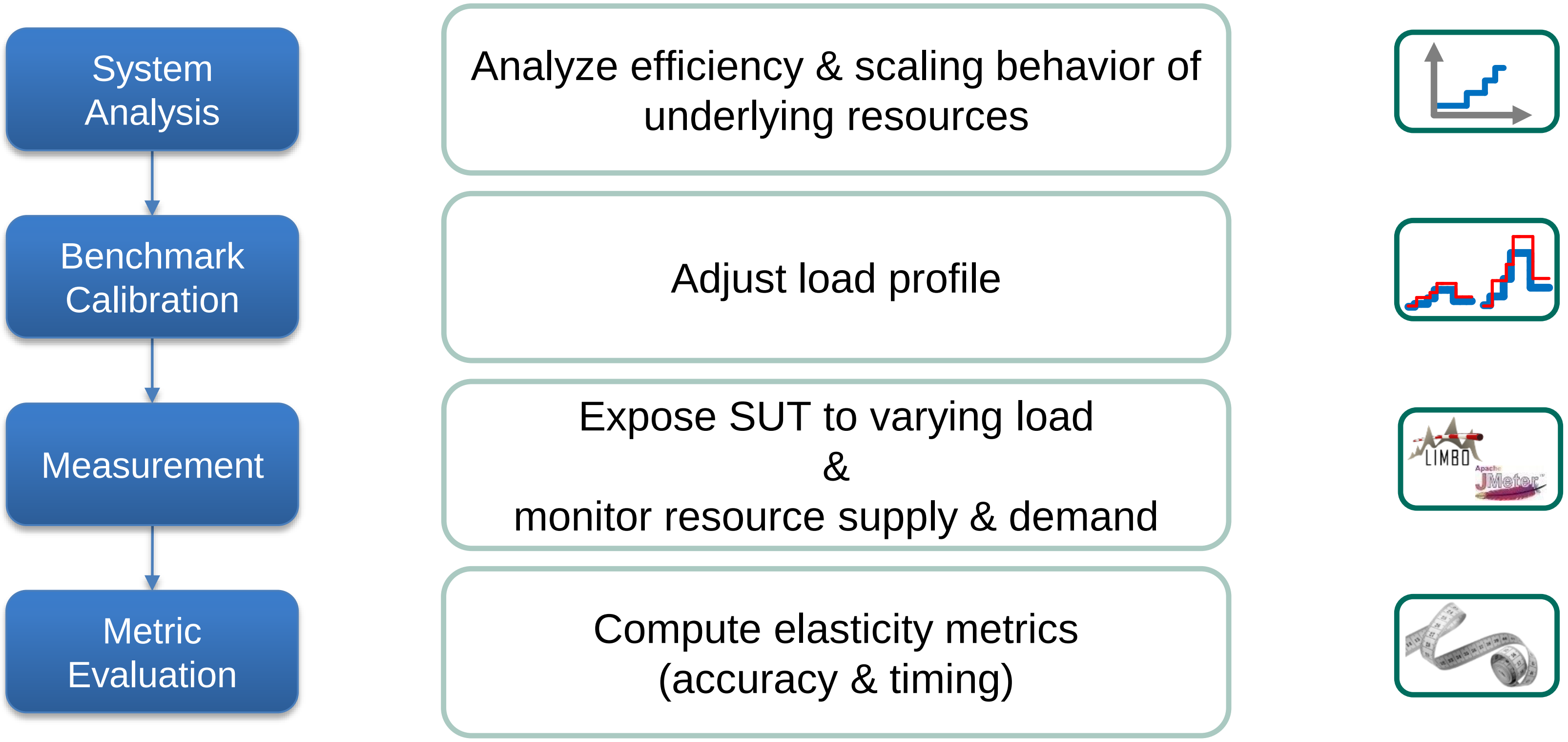
J. von Kistowski, N. Herbst and S. Kounev. **LIMBO: A Tool For Modeling Variable Load Intensities** (Demonstration Paper). In *Proc. of the 5th ACM/SPEC Intl. Conf. on Performance Engineering (ICPE 2014)*, Dublin, Ireland, March 22-26, 2014. ACM. [DOI | slides | http | .pdf]

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]

Example: Wikipedia Workload



Elasticity Benchmarking Approach



System Analysis

Analyze efficiency & scaling behavior of underlying resources

Benchmark Calibration

Adjust load profile

Measurement

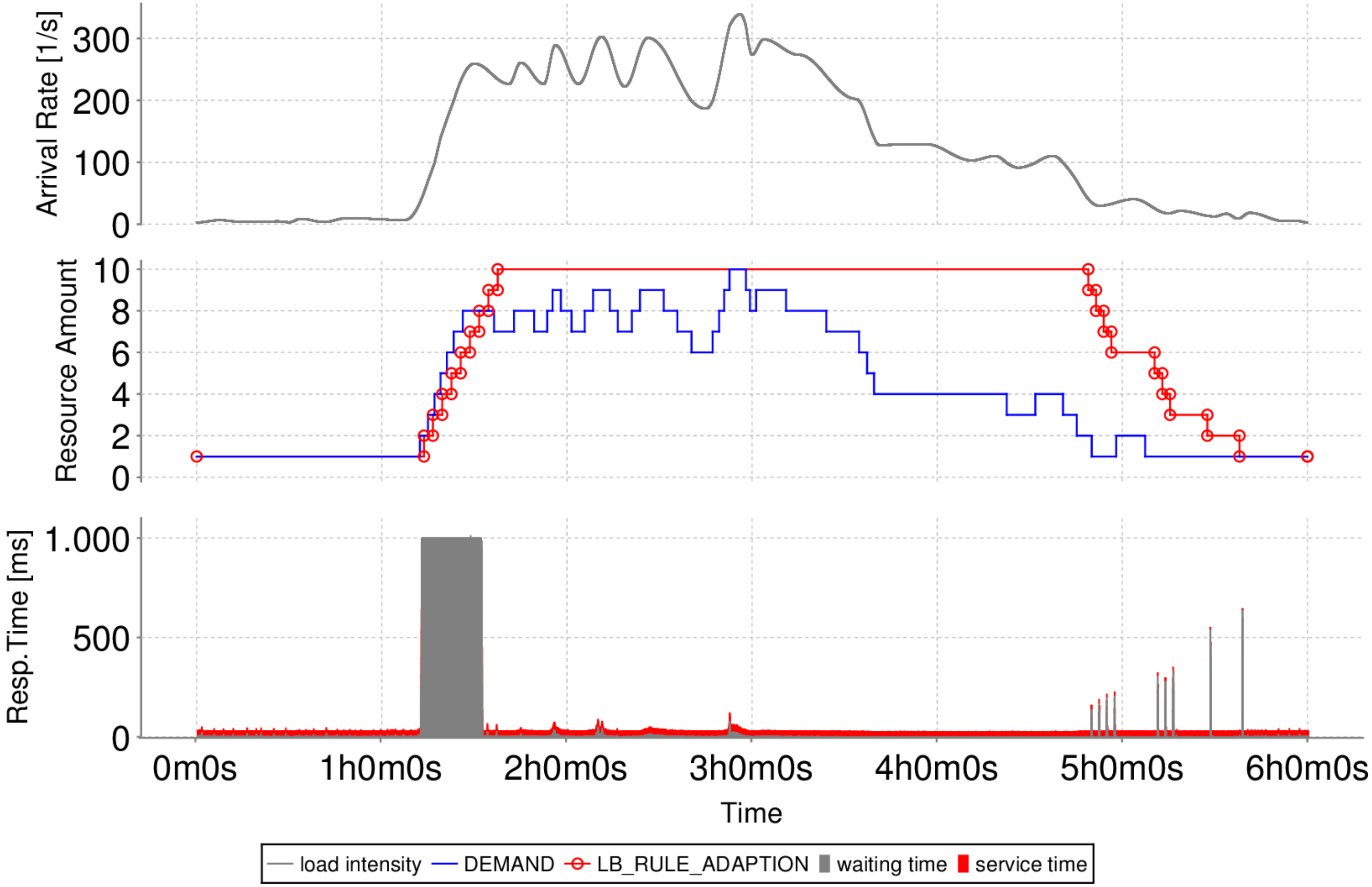
Expose SUT to varying load
&
monitor resource supply & demand

Metric Evaluation

Compute elasticity metrics
(accuracy & timing)

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)*.

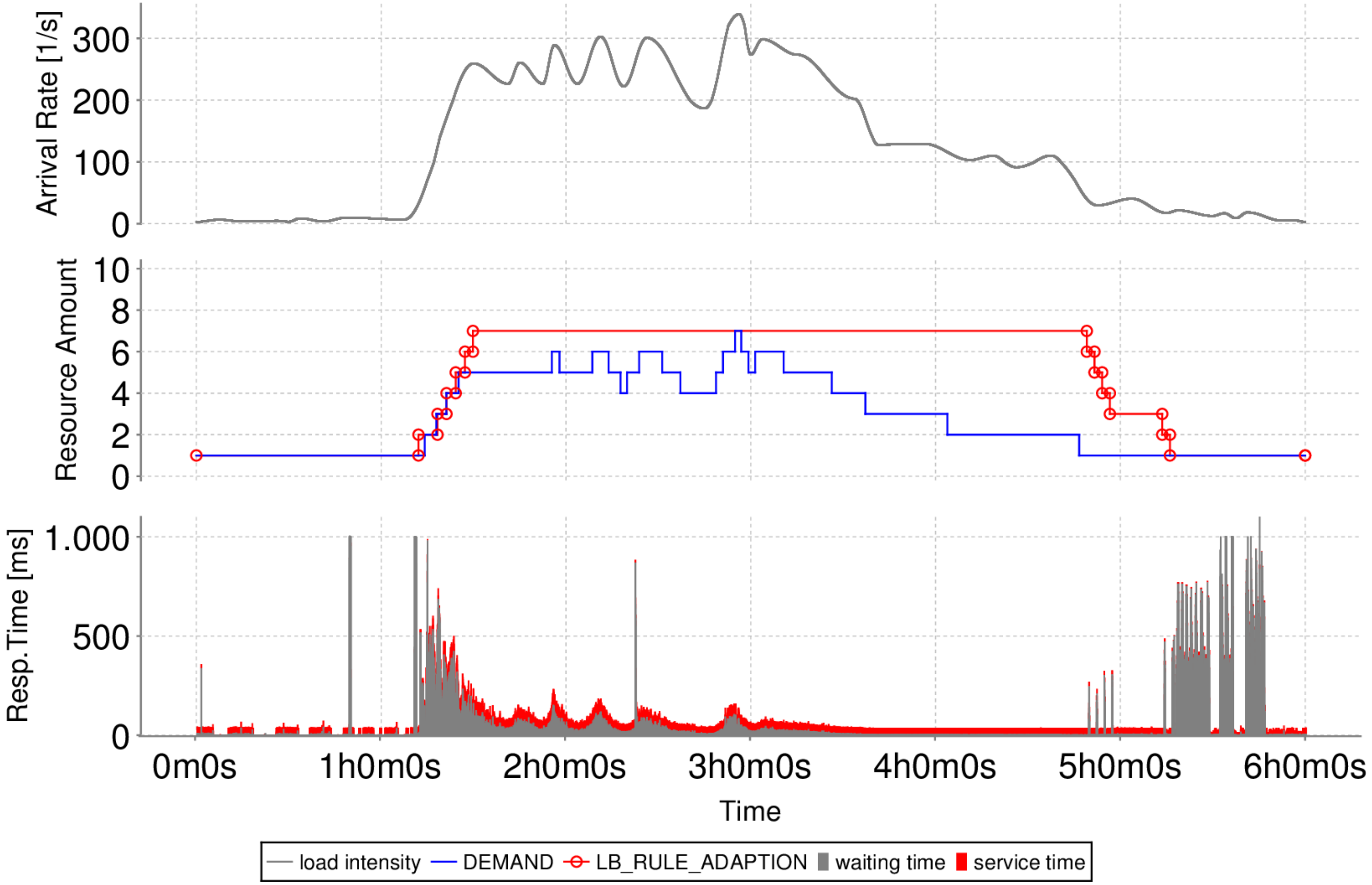
Case Study: CloudStack (CS) - 1Core



CloudStack Settings
quietTime
120s
condTrueDur
30s
threshUp
65%
threshDown
10%

Configuration	accuarcy$_O$ [res. units]	accuracy$_U$ [res. units]	timeshare$_O$ [%]	timeshare$_U$ [%]	jitter [adap/min.]	elastic speedup	violations [%]
CS – 1Core	2.423	0.067	66.1	4.8	-0.067	**1.046**	7.6

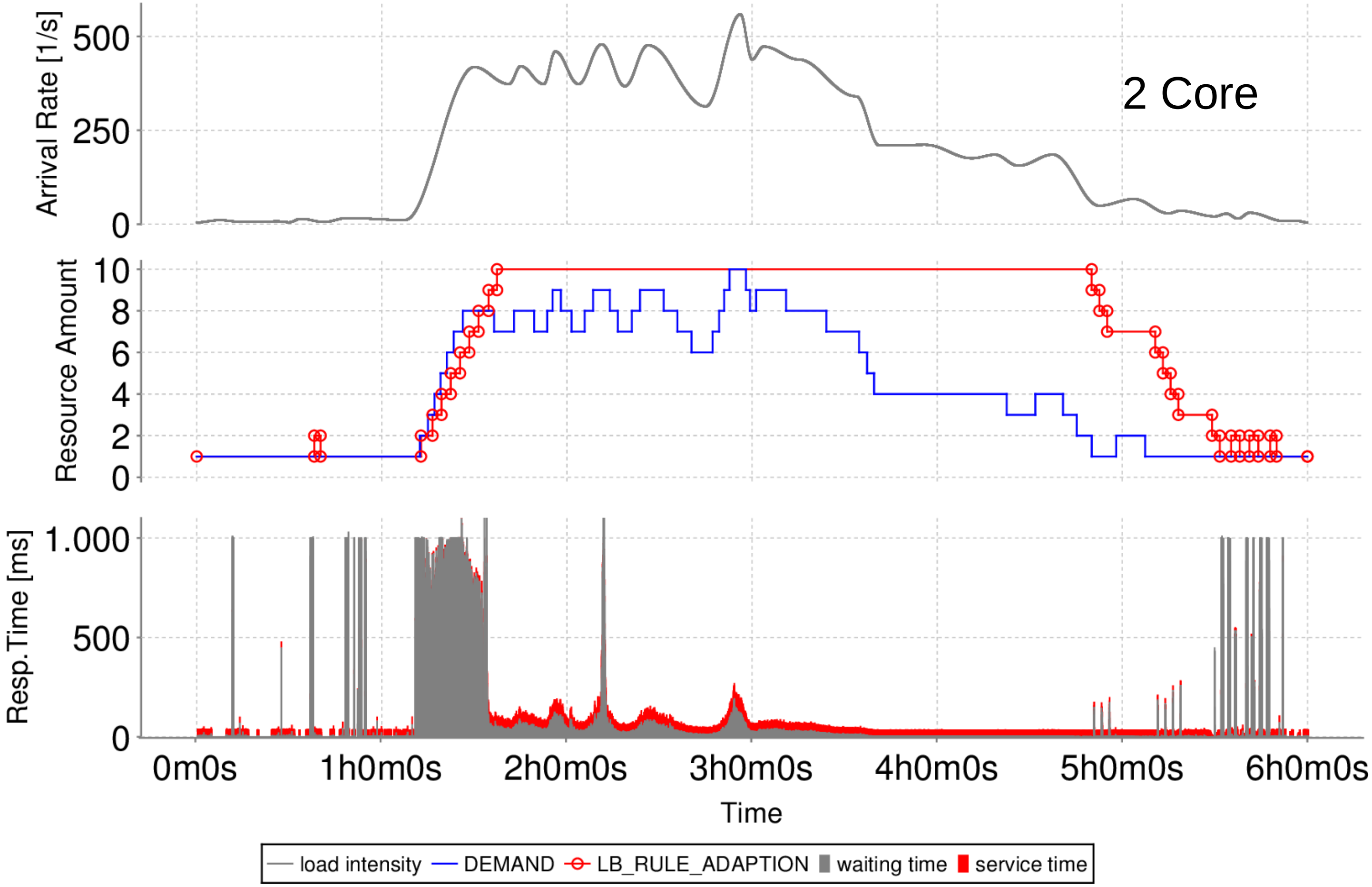
CloudStack (CS) – 2 Core – no adjustment



CloudStack Settings
quietTime
120s
condTrueDur
30s
threshUp
65%
threshDown
10%

Configuration	accuarcy$_O$ [res. units]	accuracy$_U$ [res. units]	timeshare$_O$ [%]	timeshare$_U$ [%]	jitter [adap/min.]	elastic speedup	violations [%]
CS – 1Core	2.423	0.067	66.1	4.8	-0.067	**1.046**	7.6
CS – 2Core no adjustment	1.811	0.001	63.8	0.1	-0.033	**1.291**	2.1

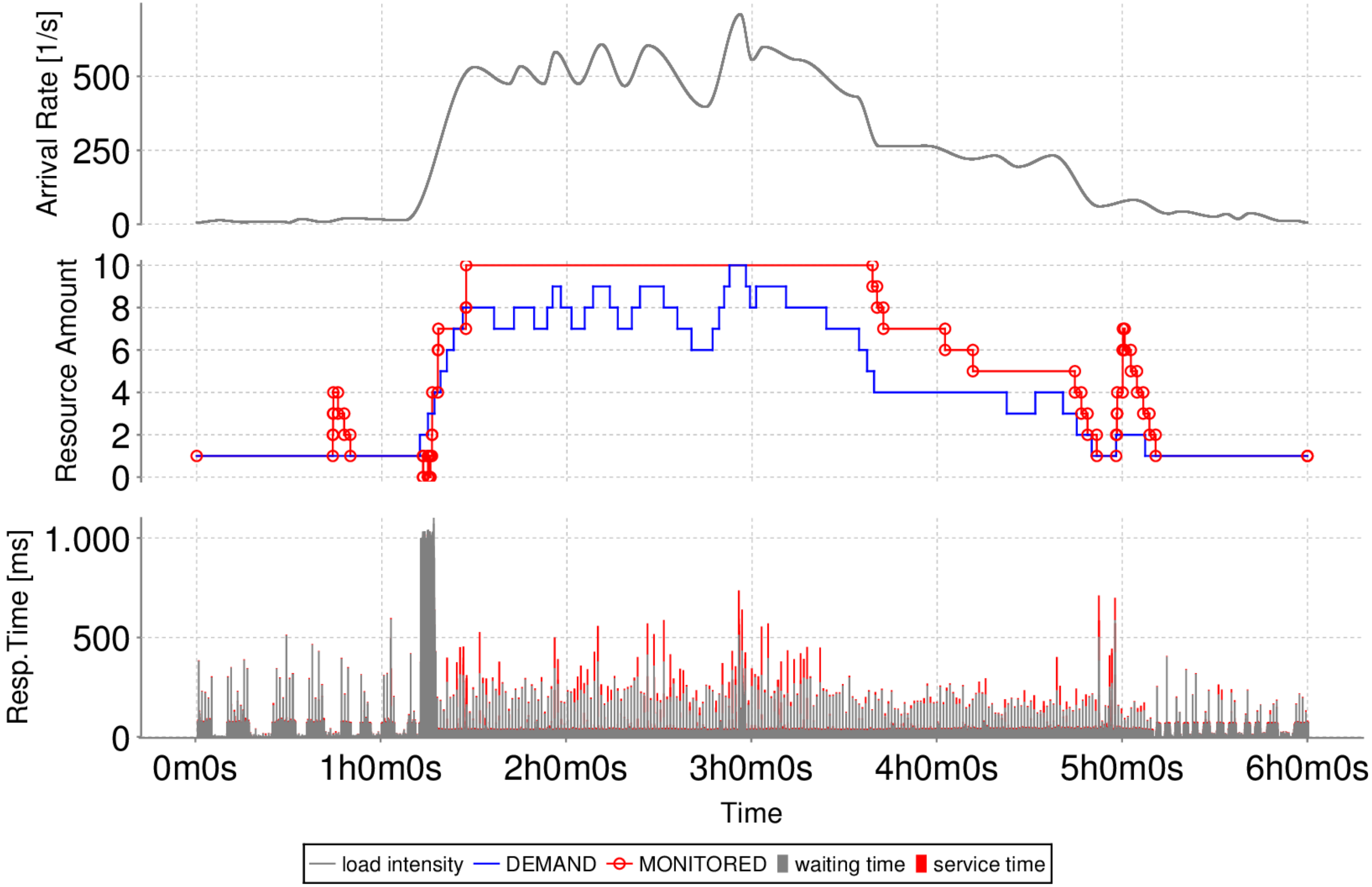
CloudStack (CS) – 2 Core – adjusted



CloudStack Settings
quietTime
120s
condTrueDur
30s
threshUp
65%
threshDown
10%

Configuration	accuarcy$_O$ [res. units]	accuracy$_U$ [res. units]	timeshare$_O$ [%]	timeshare$_U$ [%]	jitter [adap/min.]	elastic speedup	violations [%]
CS – 1Core	2.423	0.067	66.1	4.8	-0.067	**1.046**	7.6
CS – 2Core no adjustment	1.811	0.001	63.8	0.1	-0.033	**1.291**	2.1
CS – 2Core adjusted	2.508	0.061	67.1	4.5	-0.044	**1.025**	8.2

Amazon Web Services (AWS) - m1.small



CloudStack Settings
quietTime
60s
condTrueDur
60s
threshUp
80%
threshDown
50%
instUp/Down
3/1

Configuration	accuarcy$_O$ [res. units]	accuracy$_U$ [res. units]	timeshare$_O$ [%]	timeshare$_U$ [%]	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

Part II: Performance Isolation

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]

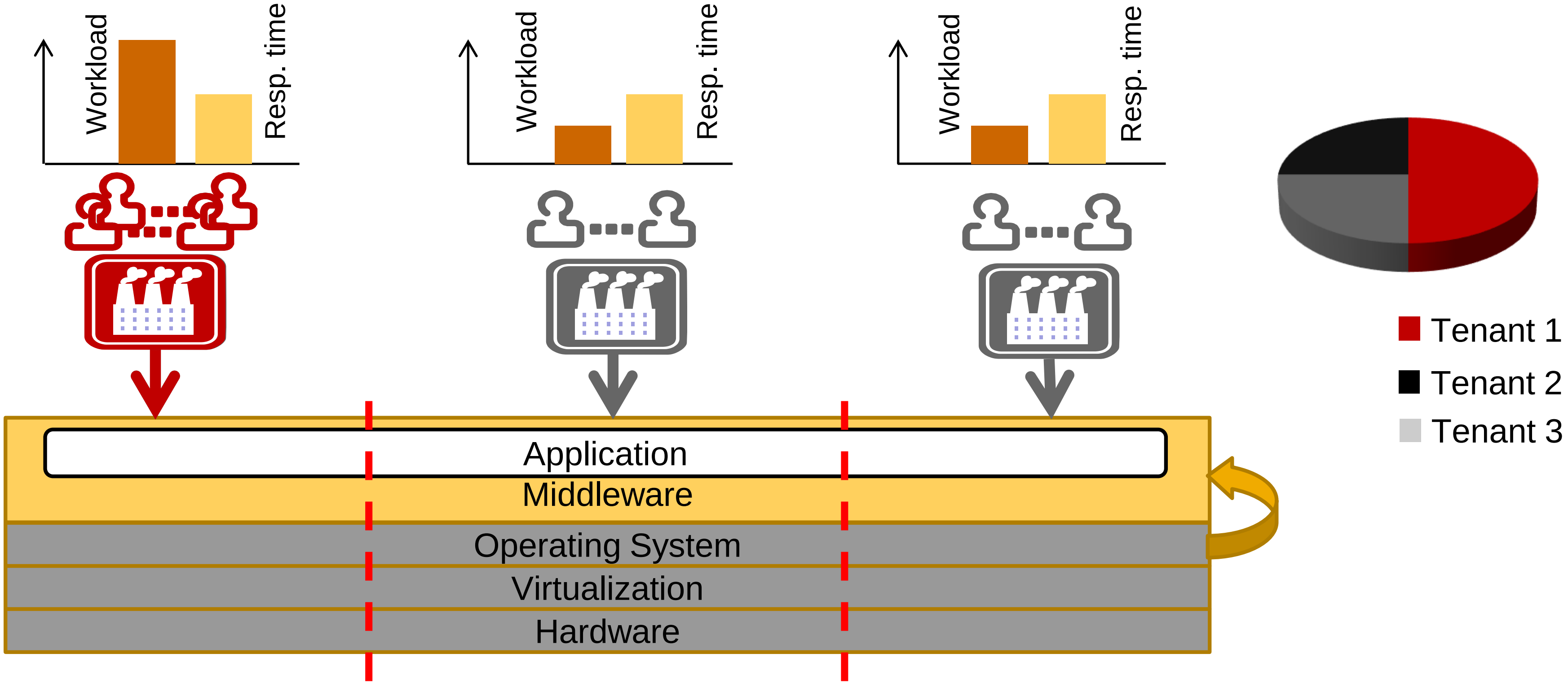
Further references

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



Application

Middleware

Operating System

Virtualization

Hardware

🟥 Tenant 1

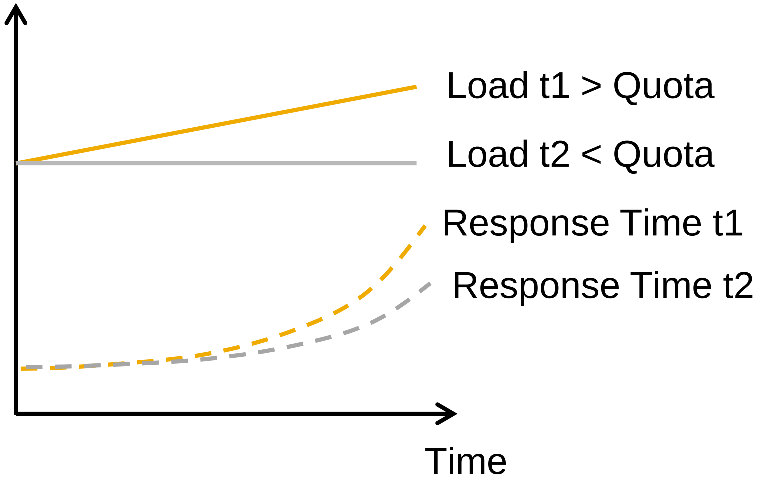
⬛ Tenant 2

⬜ Tenant 3

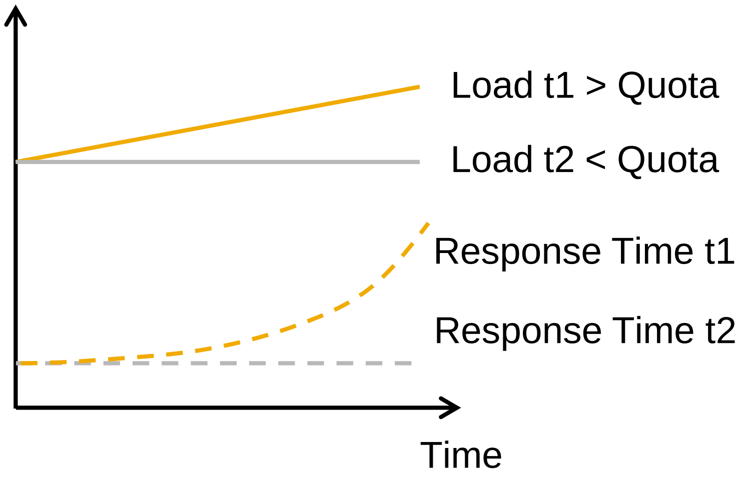
Tenants working within their assigned quota (e.g., # users) should not suffer from tenants exceeding their quotas.

Definition of Performance Isolation

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



Non-Isolated System

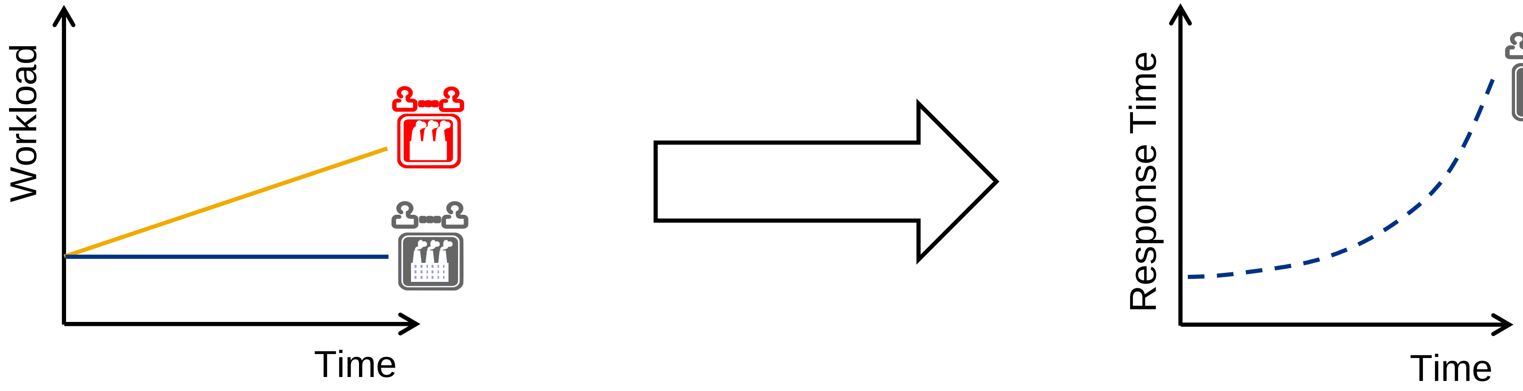


Isolated System

Performance Isolation Metrics

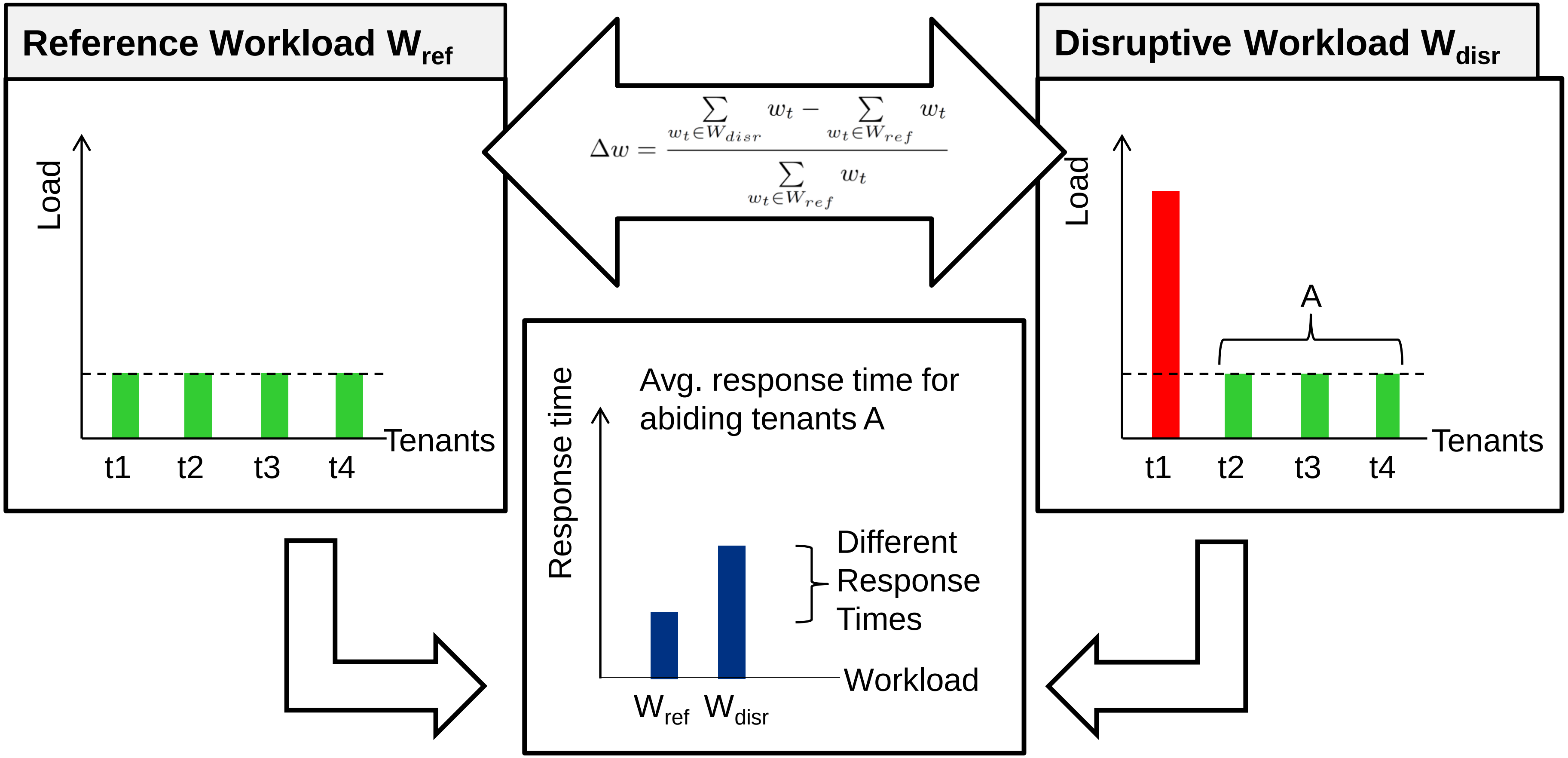
D is a set of **disruptive tenants** exceeding their quotas.

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



Reference Workload W_{ref}

Disruptive Workload W_{disr}

$$\Delta w = \frac{\sum\limits_{w_t \in W_{disr}} w_t - \sum\limits_{w_t \in W_{ref}} w_t}{\sum\limits_{w_t \in W_{ref}} w_t}$$

Avg. response time for abiding tenants A

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

Example Metric

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

Difference in response time

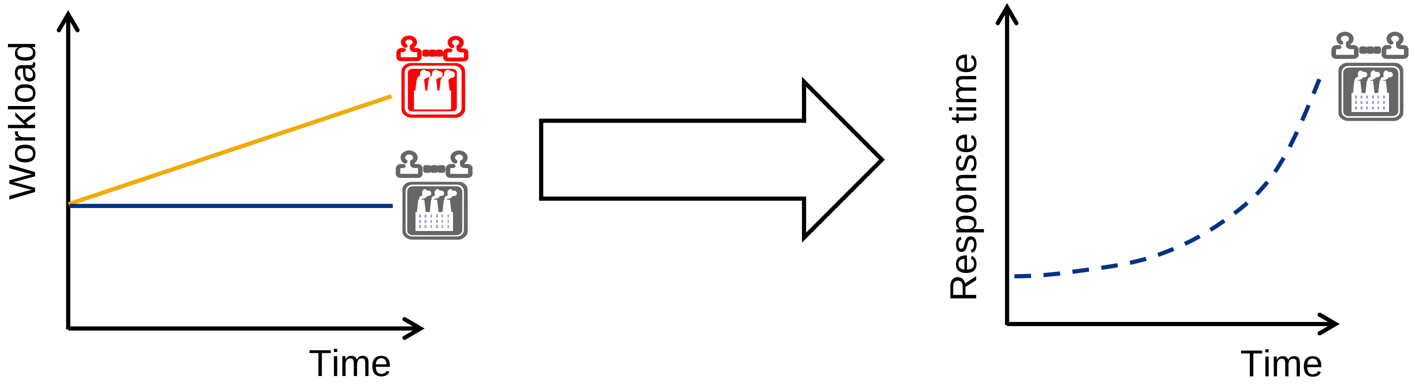
Difference in workload

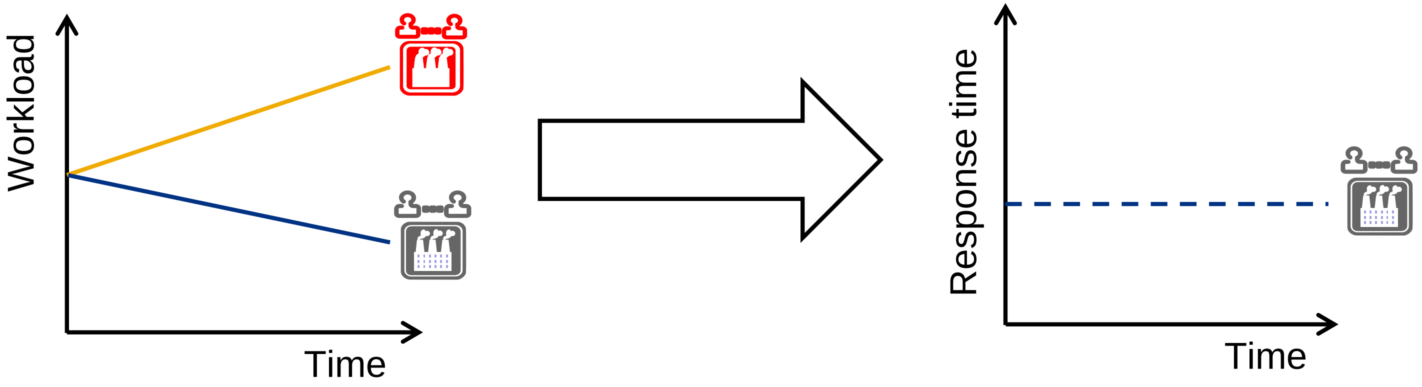
Perfectly Isolated = 0

Non-Isolated = ?

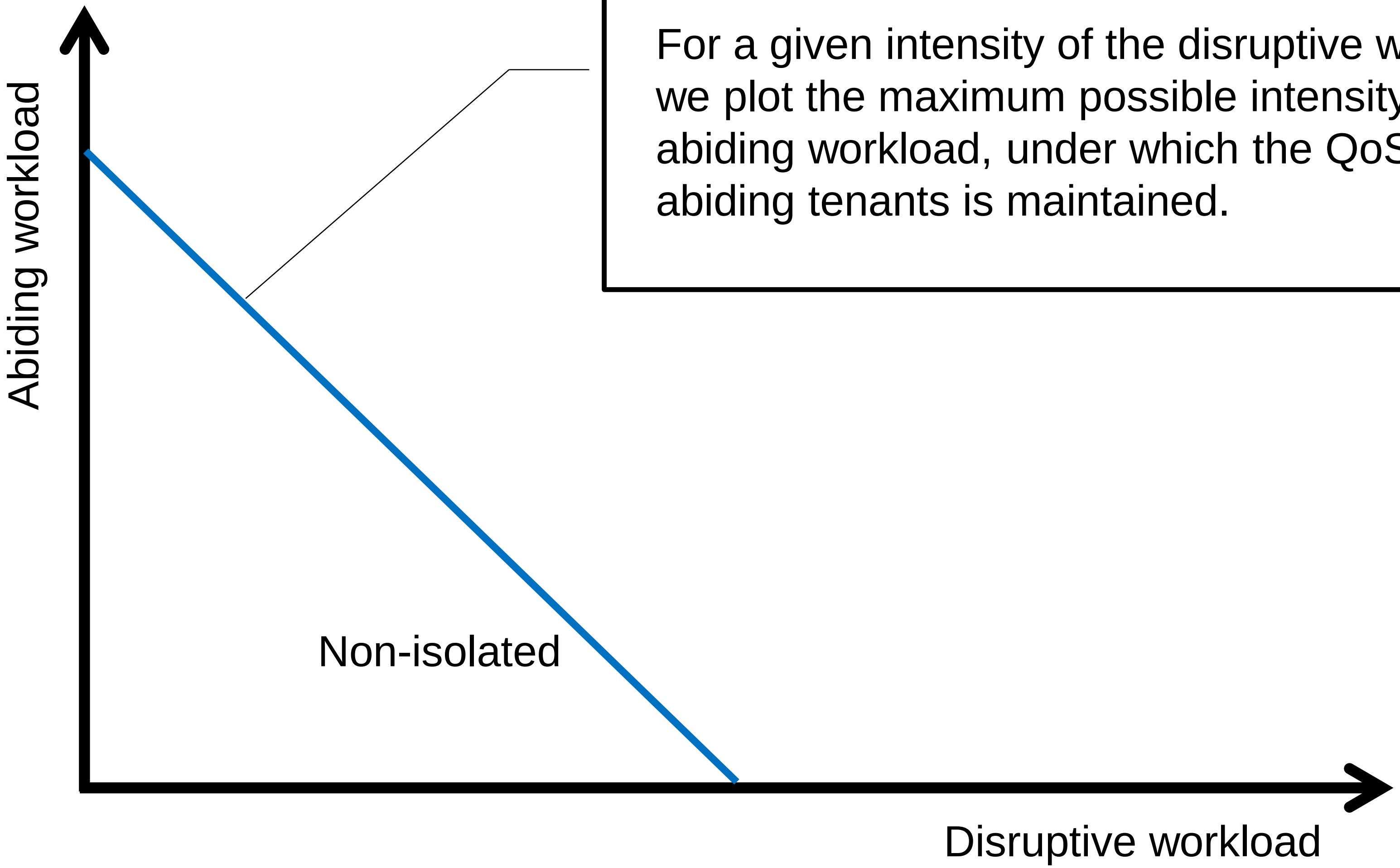
Answers: How strong is a tenant’s influence on the others?

Metrics Based on Workload Ratio





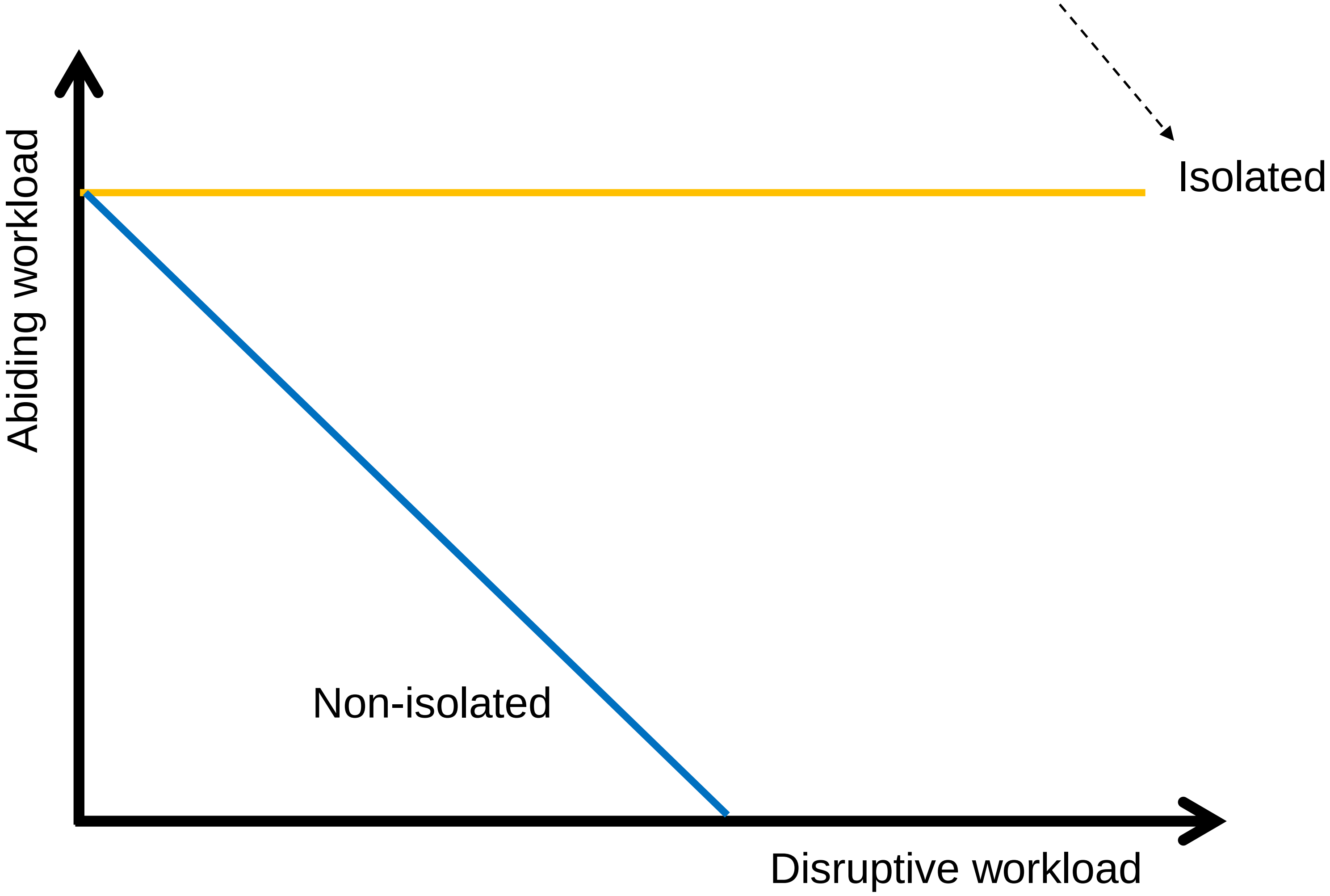
Metrics Based on Workload Ratio



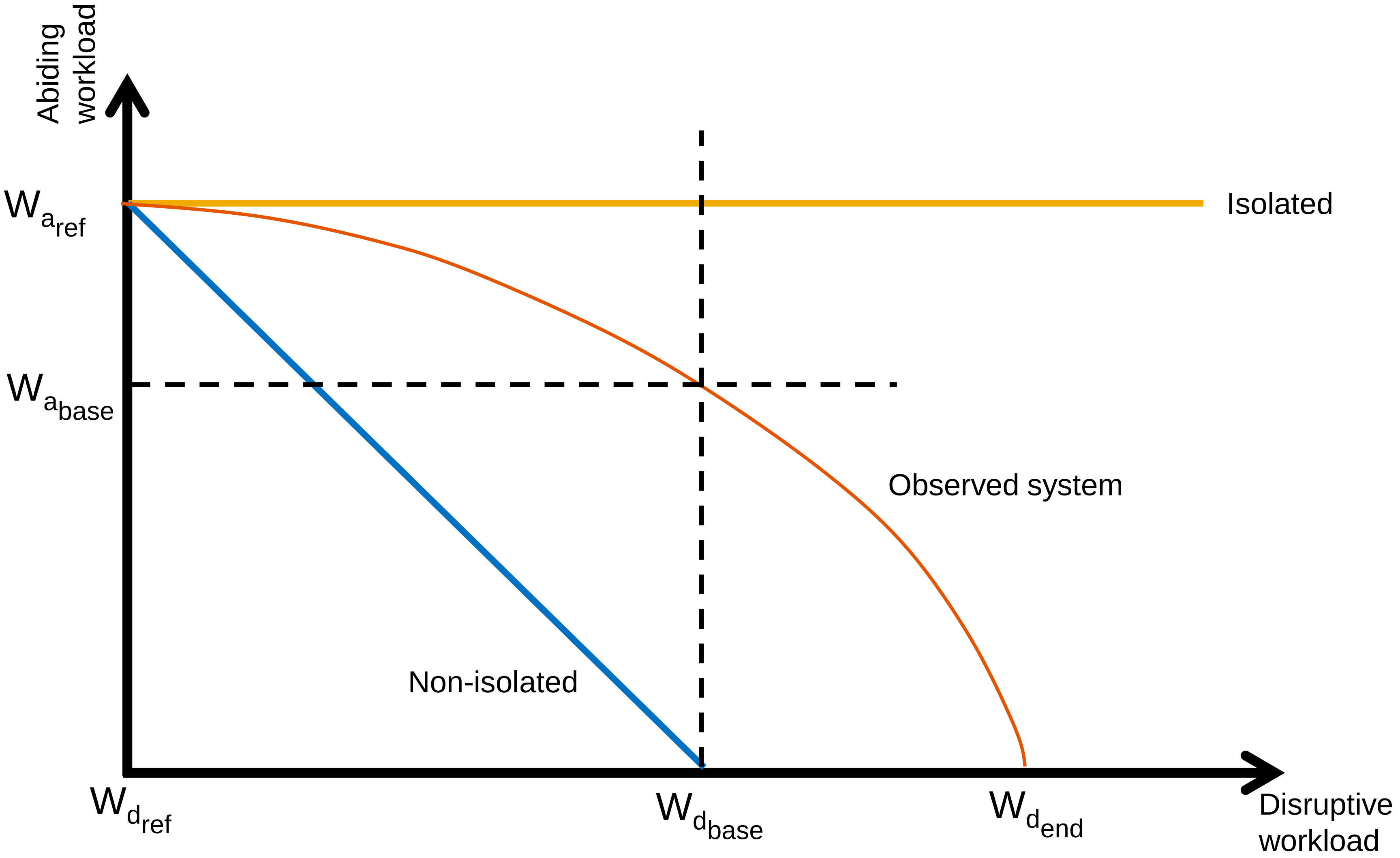
For a given intensity of the disruptive workload, we plot the maximum possible intensity of the abiding workload, under which the QoS of the abiding tenants is maintained.

Metrics Based on Workload Ratio

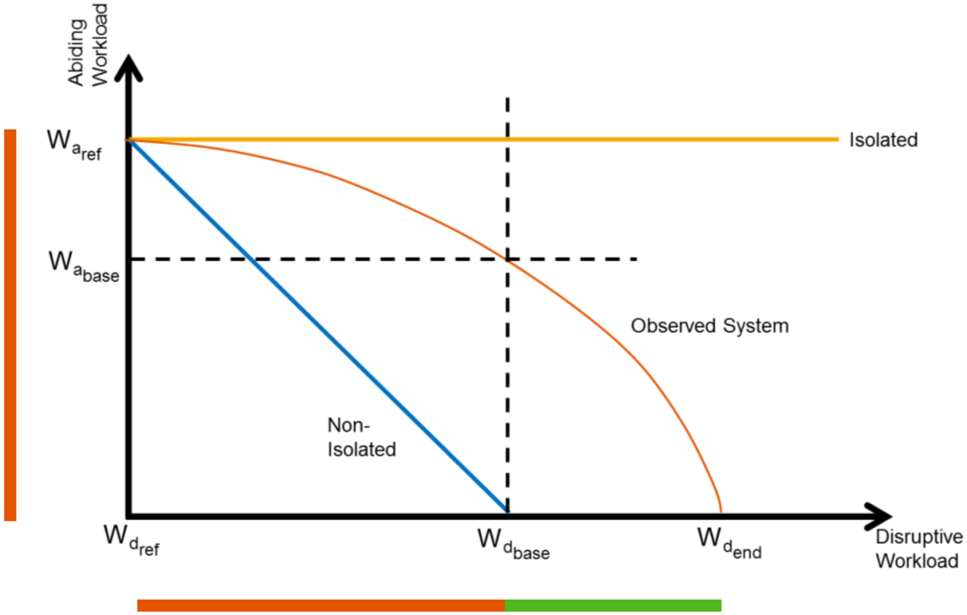
We can maintain the QoS for the abiding tenant without decreasing his workload.



Metrics Based on Workload Ratio



Example Metric: I_{end}



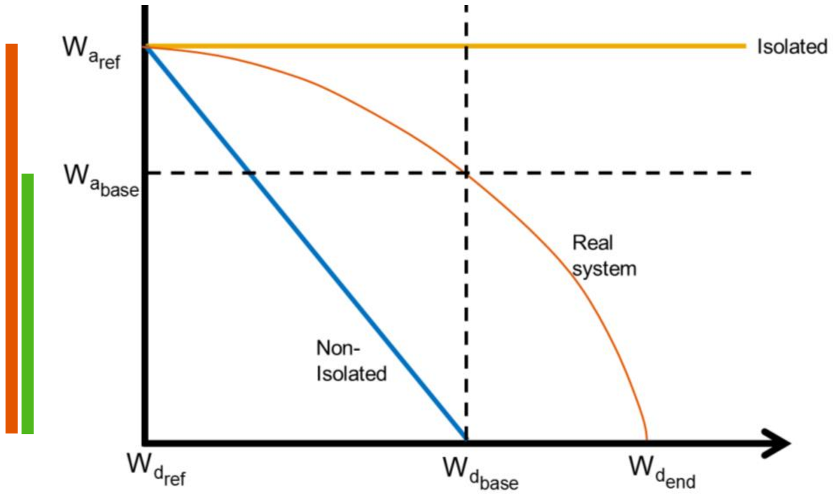
$$I_{end} = \frac{\boxed{W_{d_{end}} - W_{d_{base}}}}{\boxed{W_{a_{ref}}}}$$

Perfectly Isolated = ?

Non-Isolated = 0

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

Example Metric: I_{base}



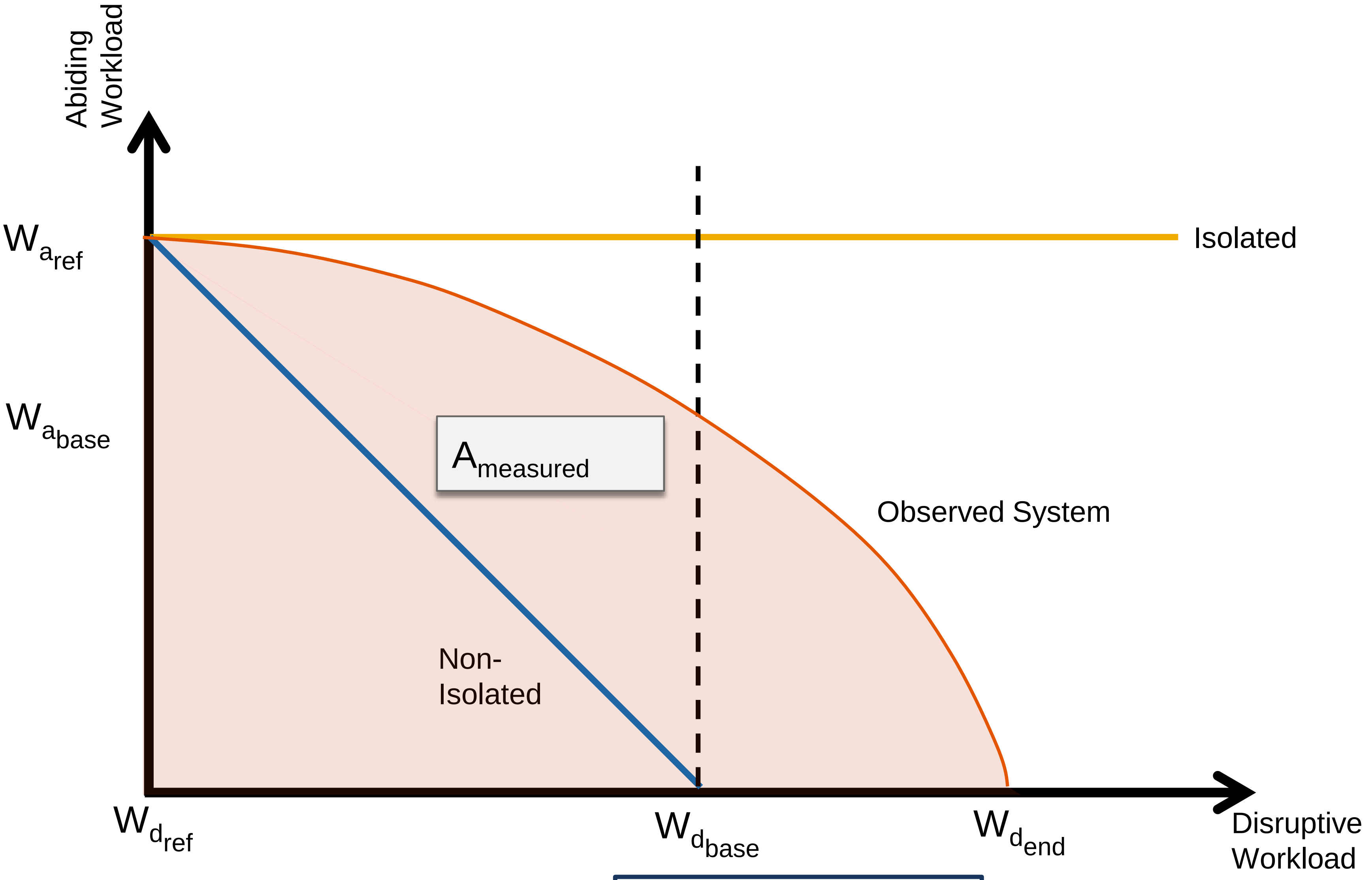
$$I_{base} = \frac{\boxed{W_{a_{base}}}}{\boxed{W_{a_{ref}}}}$$

Perfectly Isolation = 1

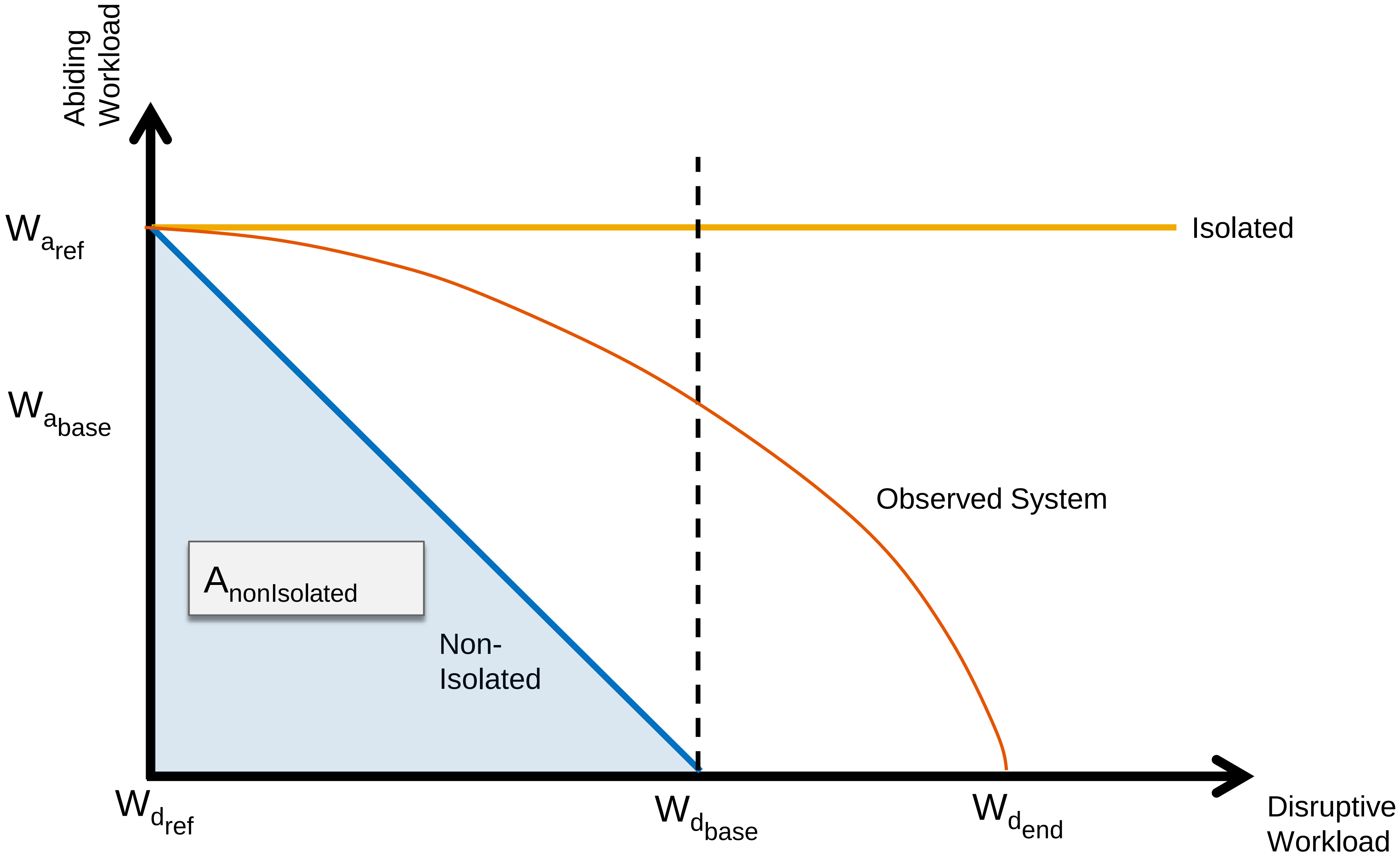
Non-Isolated = 0

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

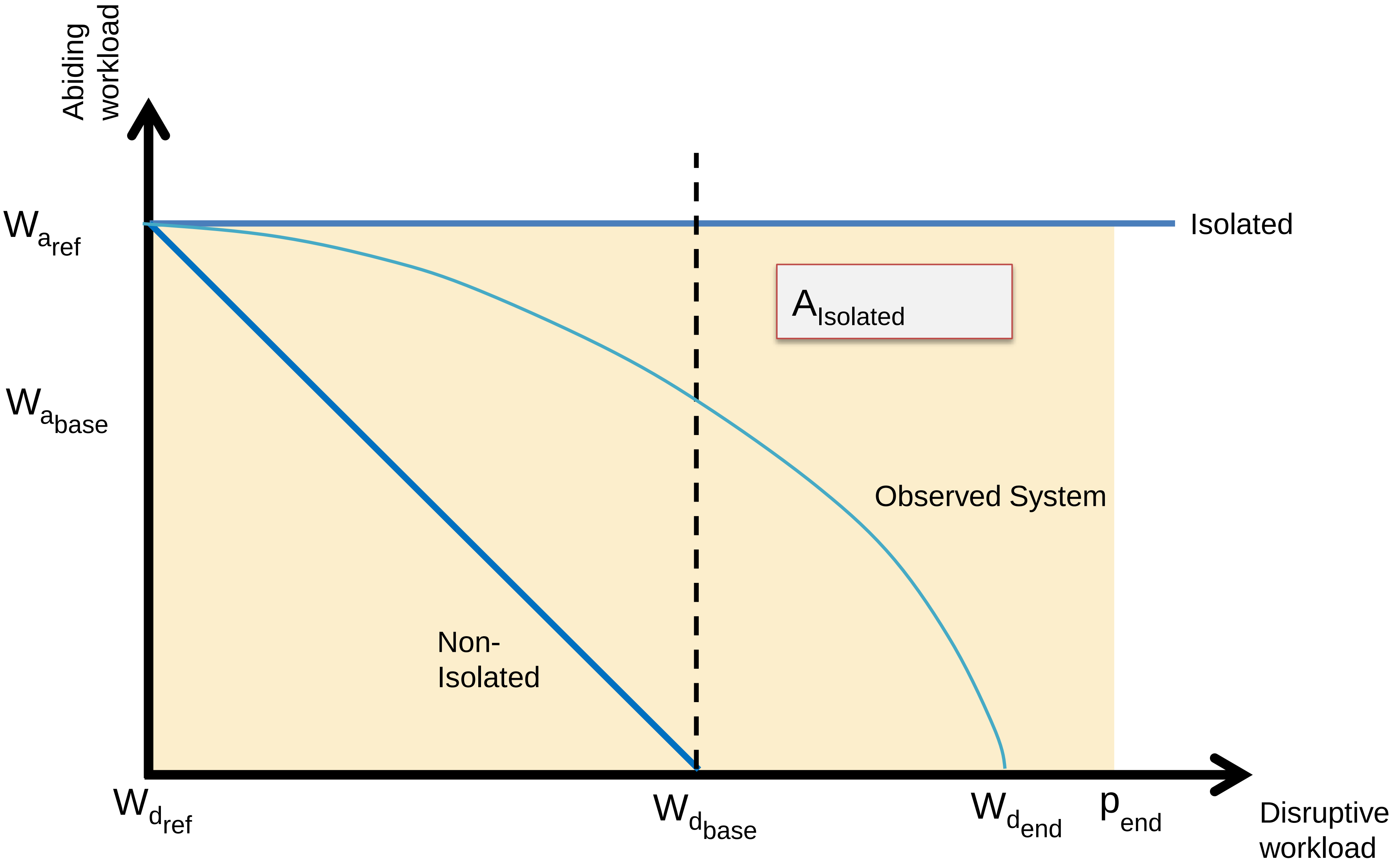
Metrics Based on Workload Ratio Integrals



Metrics Based on Workload Ratio Integrals

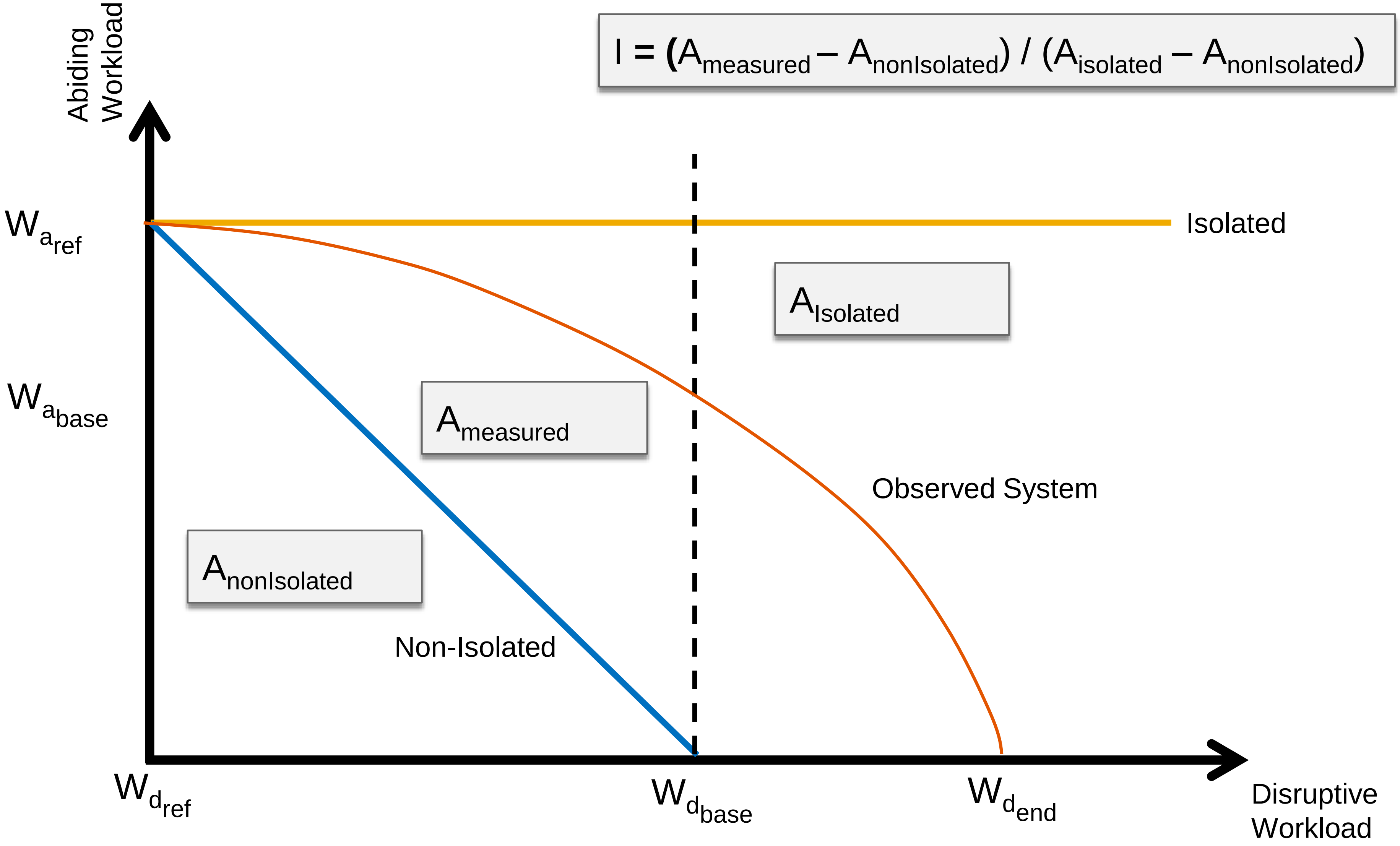


Metrics Based on Workload Ratio Integrals

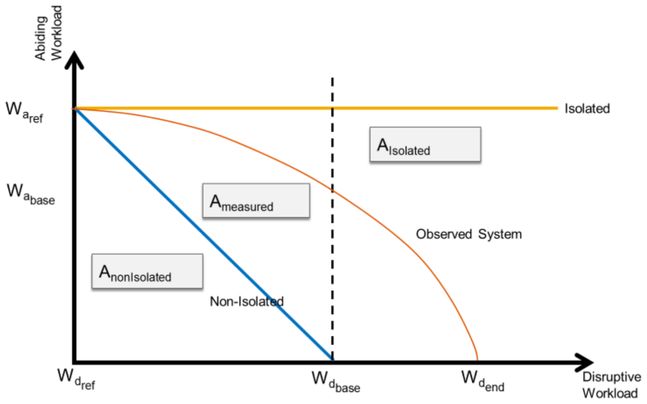


Metrics Based on Workload Ratio Integrals

$$I = (A_{measured} - A_{nonIsolated}) / (A_{isolated} - A_{nonIsolated})$$



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



$$I_{intBase} = \frac{\left(\int_{W_{d_{ref}}}^{W_{d_{base}}} f_m(W_d)\,dW_d\right) - W_{a_{ref}}^2/2}{W_{a_{ref}}^2/2}$$

Areas within W_{dref} and W_{dbase}

$$I_{intFree} = \frac{\left(\int_{W_{d_{ref}}}^{p_{end}} f_m(W_d)\,dW_d\right) - W_{a_{ref}}^2/2}{W_{a_{ref}} \cdot (p_{end} - W_{d_{ref}}) - W_{a_{ref}}^2/2}$$

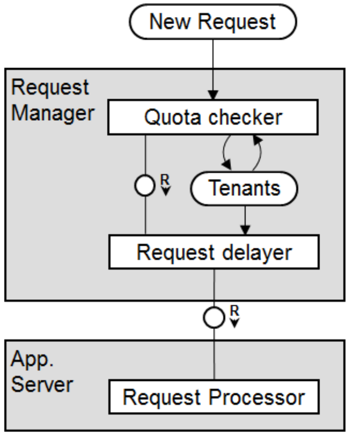
Areas within W_{dref} and predefined bound.

Perfectly Isolated = 1

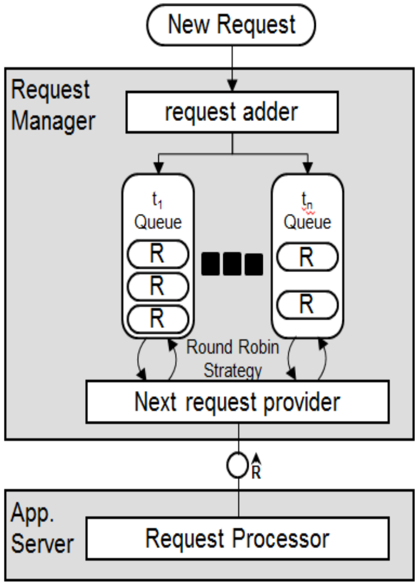
Non-Isolated = 0

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

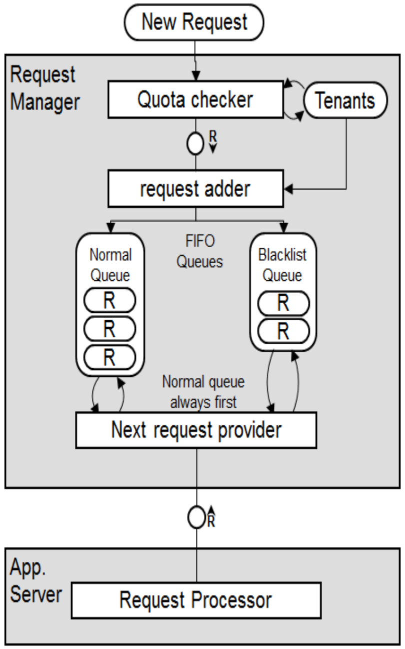
Case Study



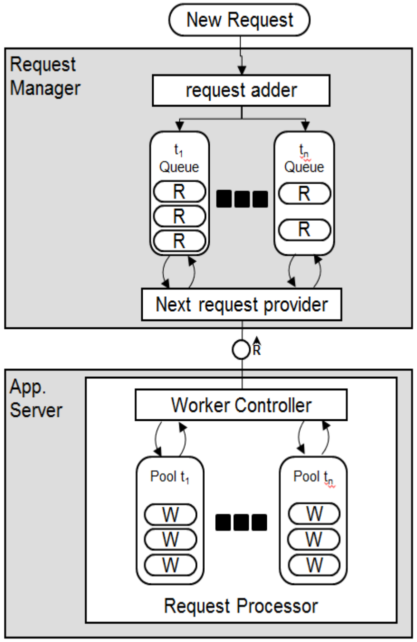
Add Delay



Round Robin



Blacklist



Separate Thread Pools

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]

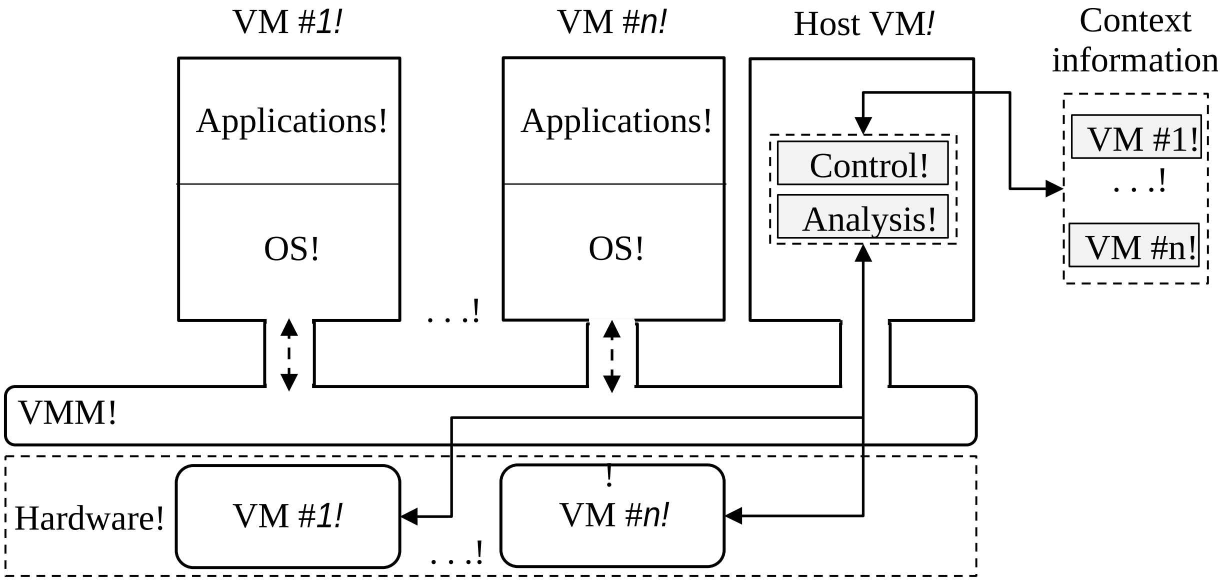
Further references

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]

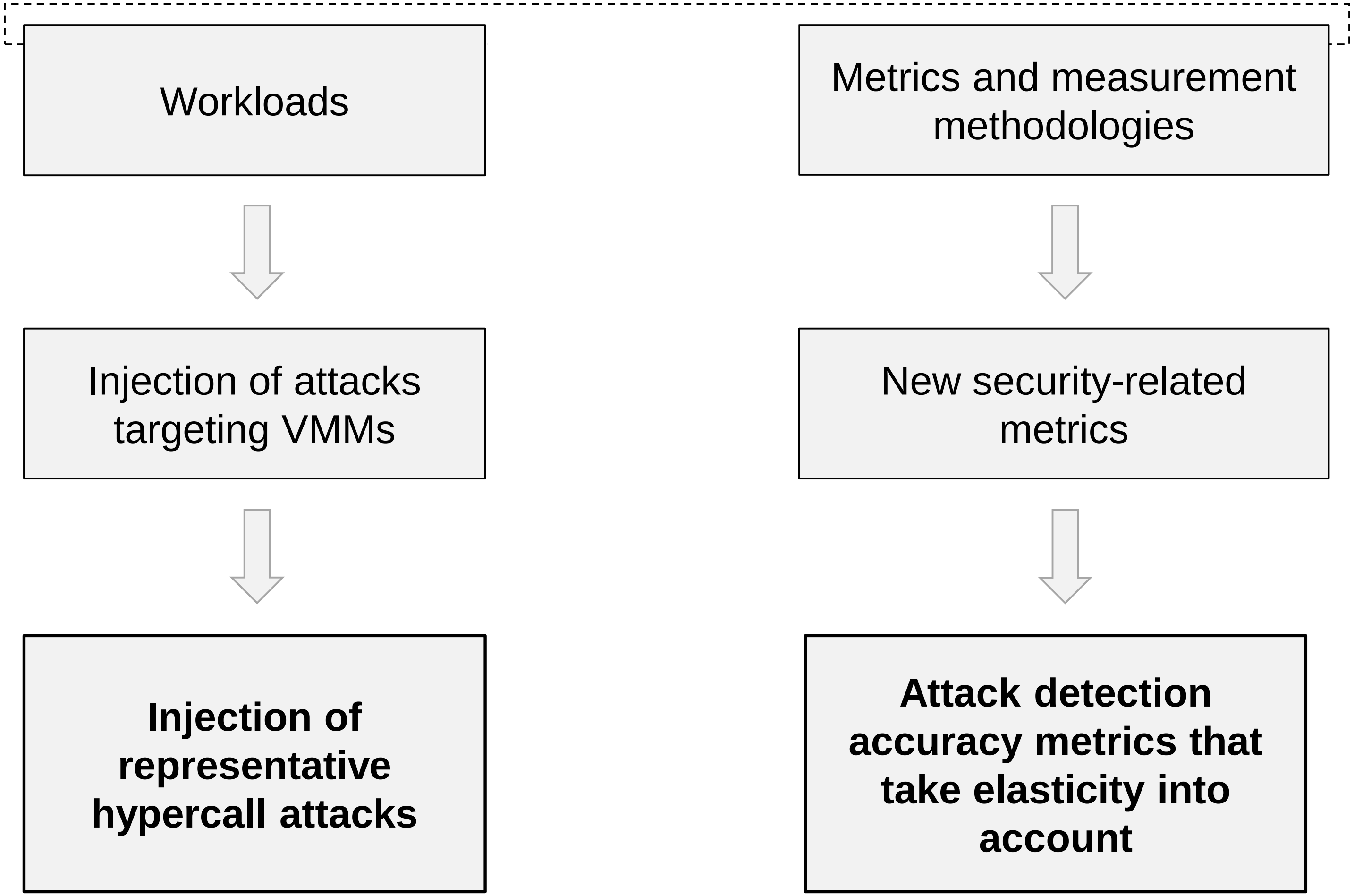
IDS Evaluation

- 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



Workloads

Injection of attacks targeting VMMs

Injection of representative hypercall attacks

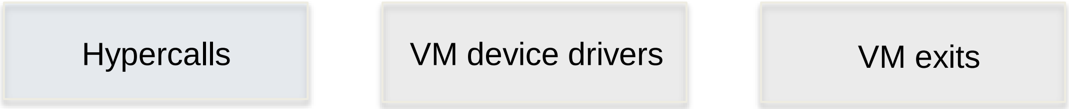
Metrics and measurement methodologies

New security-related metrics

Attack detection accuracy metrics that take elasticity into account

Malicious Workloads: Generating Attacks

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



Hypercalls

VM device drivers

VM exits

- 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)

system call

User-mode applications → OS

hypercall

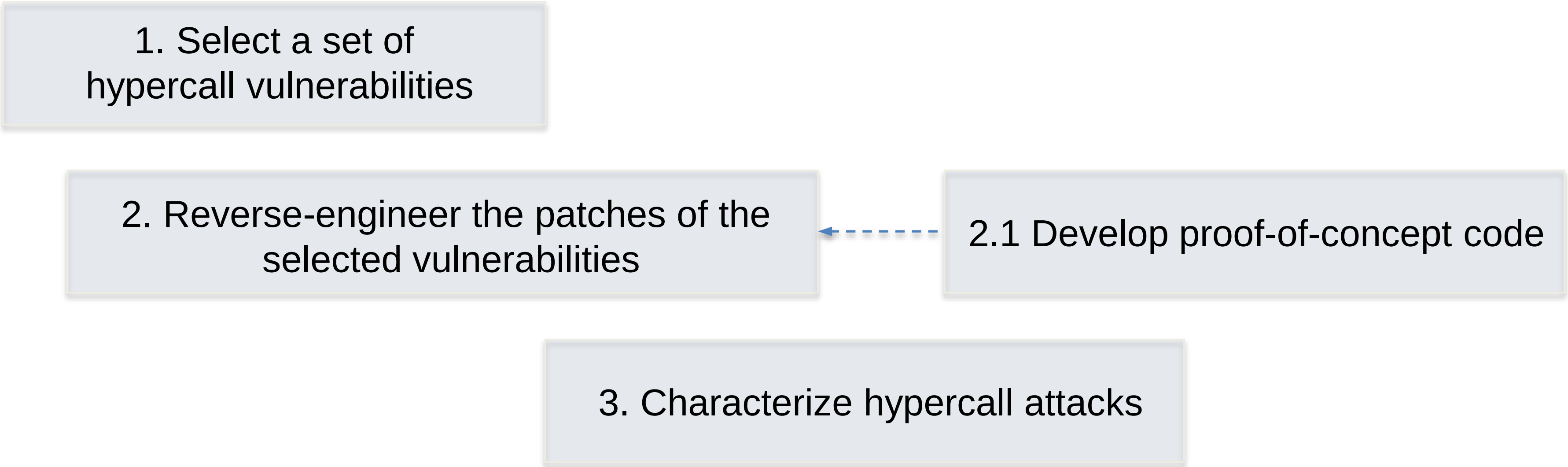
Guest VM’s OS → VMM

  - 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

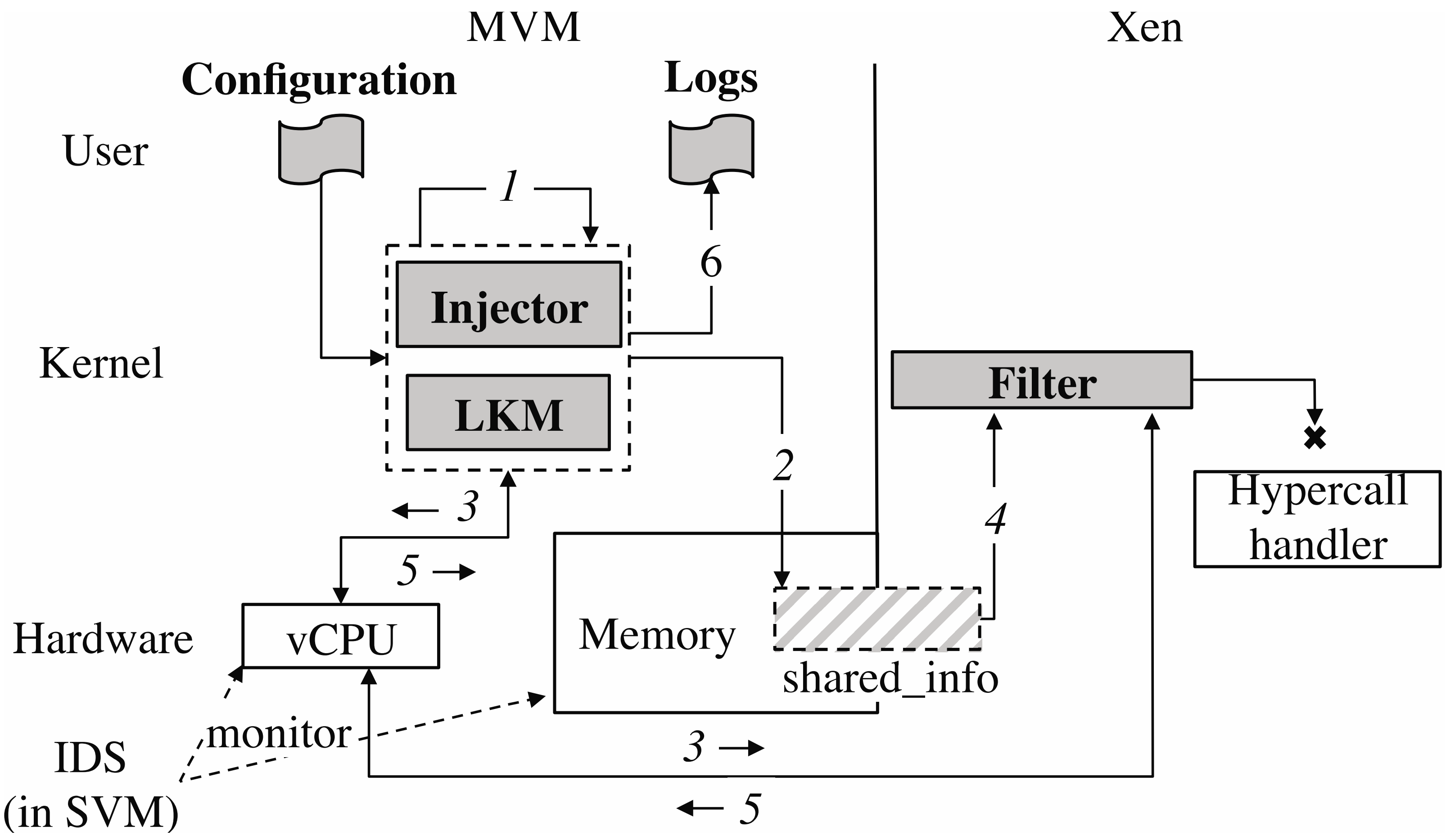
1. Invoking hypercalls from irregular call sites

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
CVE-2013-1964	**grant_table_op**	**Xen 4.1.x – 4.1.5**
CVE-2012-4539	**grant_table_op**	**Xen 4.1.x – 4.1.4**
CVE-2012-5525	**mmuext_op**	**Xen 4.2.x**
CVE-2012-5515	**memory_op**	**Xen 3.4.x – 4.1.4**
CVE-2012-3494	**set_debugreg**	**< Xen 4.1.4 (4.1 ser.), Xen 4.2.0 (4.2 ser.)**
CVE-2012-3496	**memory_op**	**Xen 3.9.x – 4.1.4**
CVE-2012-5514	**memory_op**	**Xen 3.4.x – 4.1.4**
CVE-2012-3495	**physdev_op**	**Xen 4.1.x**
CVE-2013-0154	mmuext_op	Xen 4.2.x
CVE-2012-5513	**memory_op**	**Xen 4.1.x**
CVE-2013-4553	domctl	> Xen 3.4.x
CVE-2013-0151	hvm_op	Xen 4.2.x
CVE-2013-4494	**grant_table_op**	**All versions of Xen up to the current date**
CVE-2012-5510	**grant_table_op**	**< Xen 4.1.4 (4.1 ser.), Xen 4.2.0 (4.2 ser.)**
CVE-2013-3898	unknown	Windows 8 / Windows Server 2012

Observations

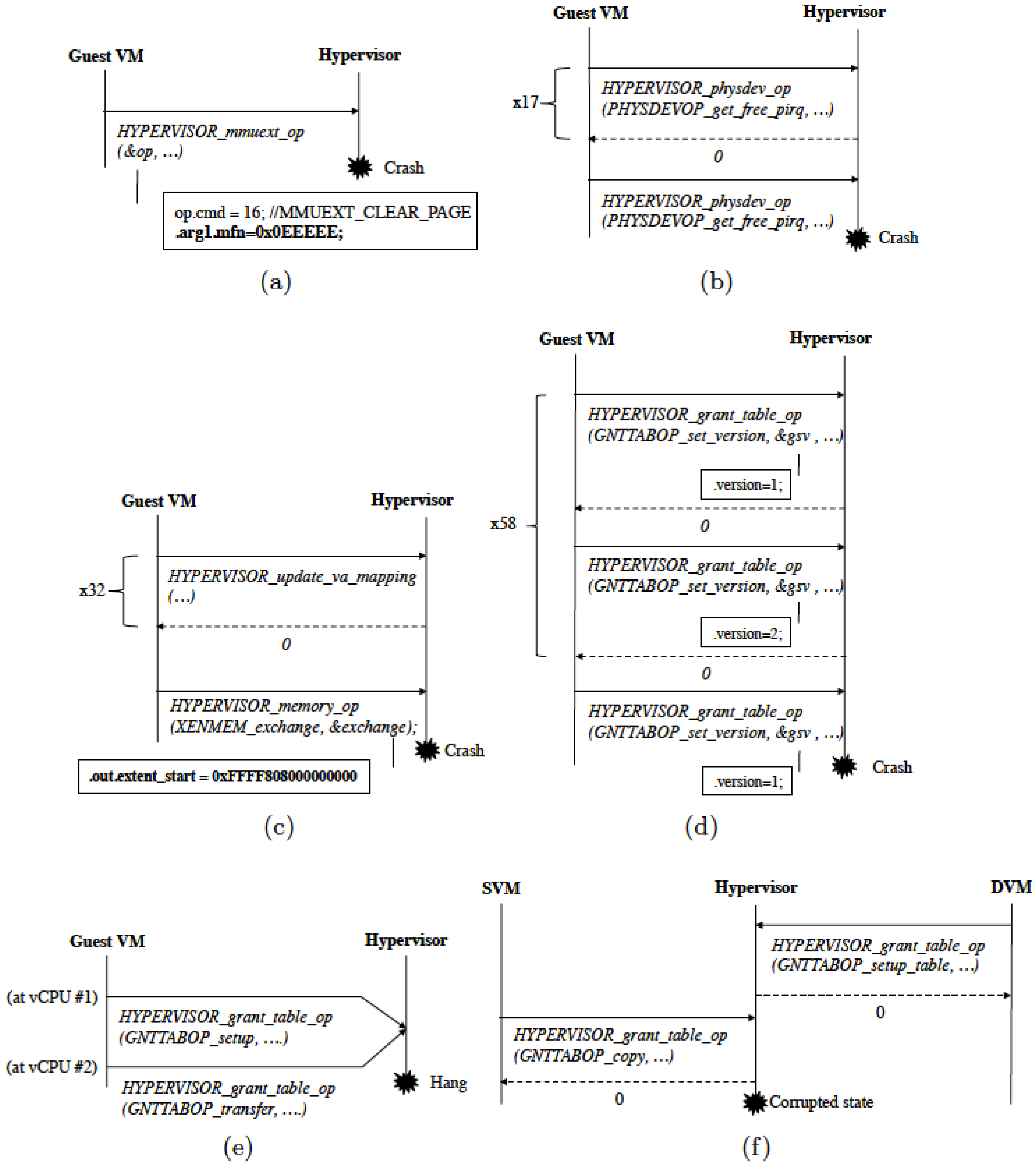
- Errors causing hypercall vulnerabilities
  - Implementation errors (missing value validation, incorrect value validation, and incorrect implementation of inverse procedures)
  - Hypervisor design errors

- 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)

Field study on hypercall vulnerabilities: Observations (cont.)

- Hypercall attacks
  - Effects: crash, hang, corrupt state, information leakage
  - Very effective hypervisor DoS attacks – critical: downtime minute of the virtualized cloud infrastructure of Amazon costs $66,240
  - An effective mechanism for intruding hypervisors, however, as part of a multi-step attack
    - Hypercall attack -> paving the way for further malicious activities

- Hypercall attack models
  - execution of a single hypercall with:
    - **regular** parameter value(s) (i.e., regular hypercall), or
    - parameter value(s) **specifically crafted** for triggering a given vulnerability, which includes values inside and outside valid value domains, or
  - execution of a series of **regular** hypercalls in a given order, including:
    - repetitive execution of a single hypercall, or
    - repetitive execution of multiple hypercalls.
  - where an execution of (a) regular hypercall(s) is performed in a way such that:
    - the targeted hypervisor cannot properly handle by design, or
    - an erroneous program code is reached.


Guest VM
Hypervisor
HYPERVISOR_mmuext_op
(&op, …)
Crash
op.cmd = 16; //MMUEXT_CLEAR_PAGE
.arg1.mfn=0x0EEEEE;
(a)
Guest VM
Hypervisor
x17
HYPERVISOR_physdev_op
(PHYSDEVOP_get_free_pirq, …)
0
HYPERVISOR_physdev_op
(PHYSDEVOP_get_free_pirq, …)
Crash
(b)
Guest VM
Hypervisor
x32
HYPERVISOR_update_va_mapping
(…)
0
HYPERVISOR_memory_op
(XENMEM_exchange, &exchange);
Crash
.out.extent_start = 0xFFFF808000000000
(c)
Guest VM
Hypervisor
x58
HYPERVISOR_grant_table_op
(GNTTABOP_set_version, &gsv , …)
.version=1;
0
HYPERVISOR_grant_table_op
(GNTTABOP_set_version, &gsv , …)
.version=2;
0
HYPERVISOR_grant_table_op
(GNTTABOP_set_version, &gsv , …)
Crash
.version=1;
(d)
Guest VM
Hypervisor
(at vCPU #1)
HYPERVISOR_grant_table_op
(GNTTABOP_setup, ….)
(at vCPU #2)
HYPERVISOR_grant_table_op
(GNTTABOP_transfer, ….)
Hang
(e)
SVM
Hypervisor
DVM
HYPERVISOR_grant_table_op
(GNTTABOP_setup_table, …)
0
HYPERVISOR_grant_table_op
(GNTTABOP_copy, …)
0
Corrupted state
(f)

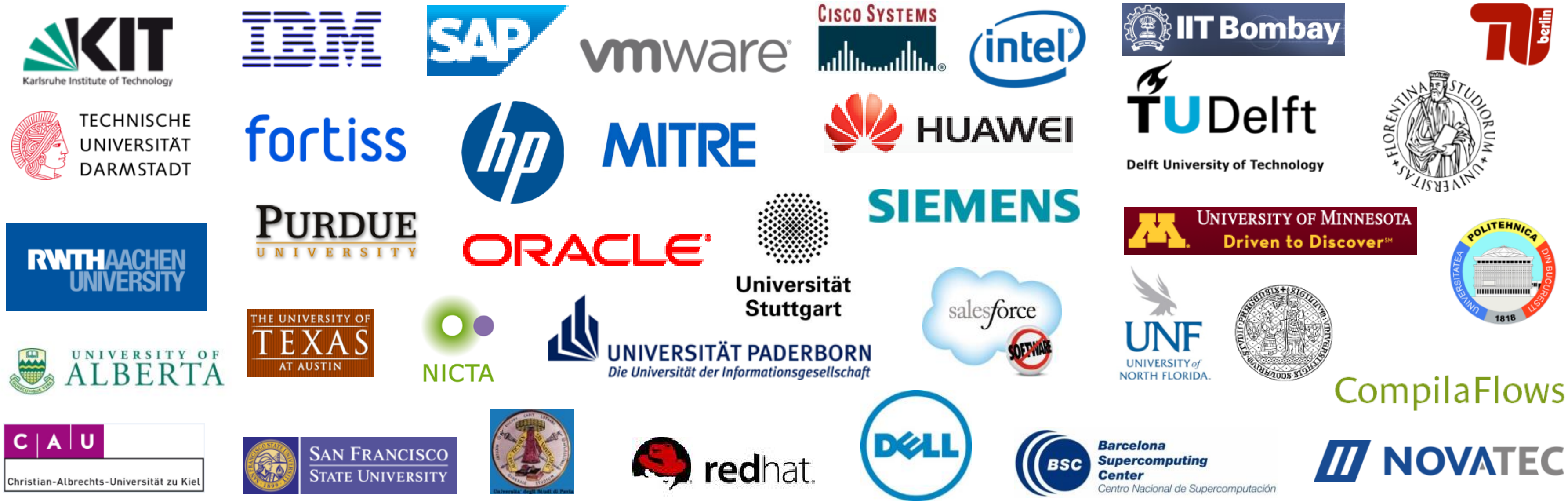
Standard-Performance-Evaluation-Corporation

SPEC Research Group

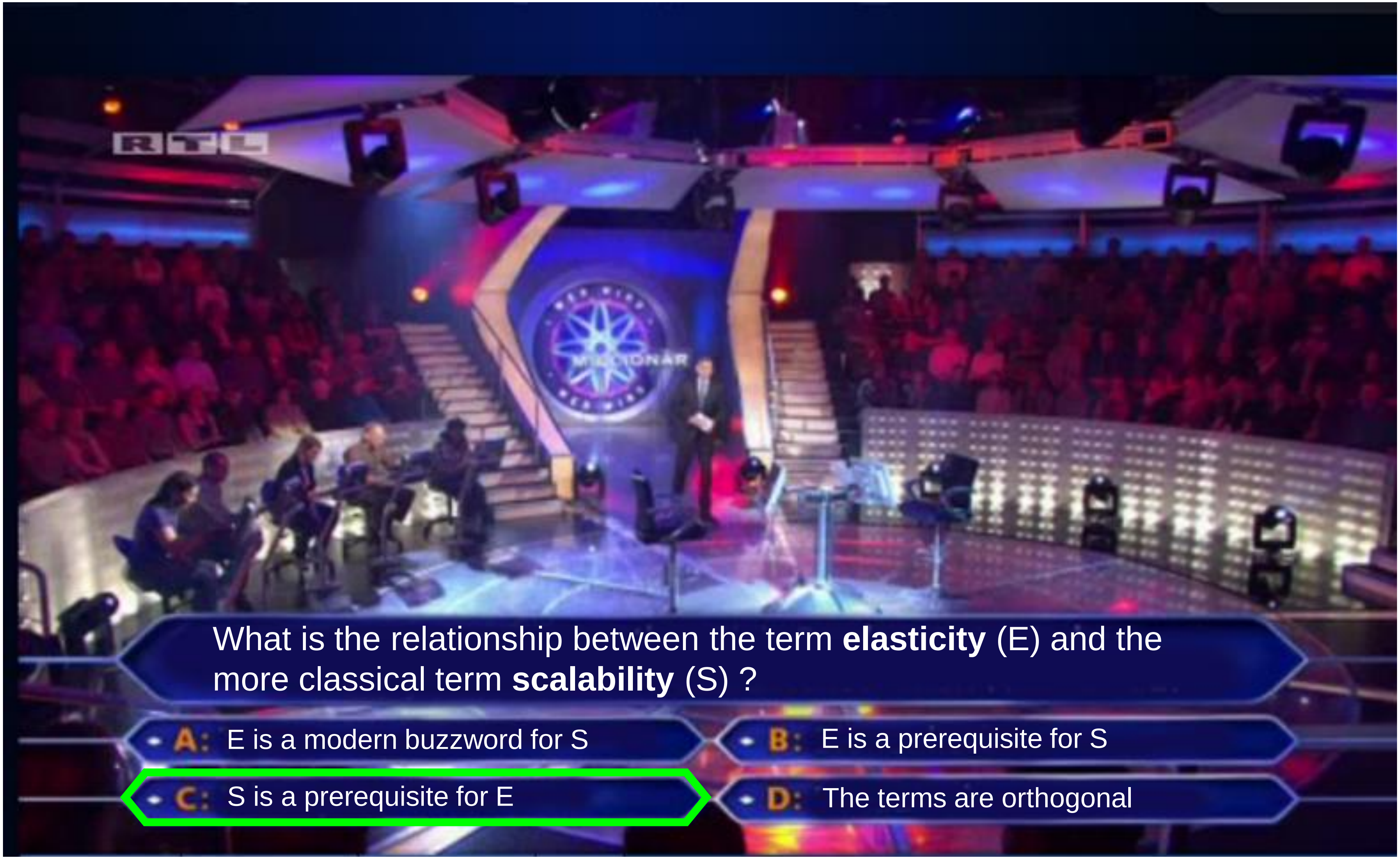
- **Open-Systems-Group (OSG)**
  - Processor and computer architectures
  - Virtualization platforms
  - Java (JVM, Java EE)
  - Message-based systems
  - Storage systems (SFS)
  - Web-, email- and file server
  - SIP server (VoIP)
  - Cloud computing
- **High-Performance-Group (HPG)**
  - Symmetric multiprocessor systems
  - Workstation clusters
  - Parallel and distributed systems
  - Vector (parallel) supercomputers
- **“Graphics and Workstation Performance Group” (GWPG)**
  - CAD/CAM, visualization
  - OpenGL

SPEC Research Group (RG)

- Founded in March 2011
  - Transfer of knowledge btw. academia and industry
- Activities
  - Methods and techniques for experimental system analysis
  - Standard metrics and measurement methodologies
  - Benchmarking and certification
  - Evaluation of academic research results
- Member organizations (Feb 2014)


KIT Karlsruhe Institute of Technology
IBM
SAP
vmware
Cisco Systems
intel
IIT Bombay
TU berlin
Technische Universität Darmstadt
fortiss
hp
MITRE
HUAWEI
TU Delft Delft University of Technology
Florentina Studiorum Universitas
RWTH Aachen University
Purdue University
ORACLE
Universität Stuttgart
SIEMENS
University of Minnesota Driven to Discover
Politehnica Universitatea din Bucuresti 1818
University of Alberta
The University of Texas at Austin
NICTA
Universität Paderborn Die Universität der Informationsgesellschaft
salesforce
UNF University of North Florida
CompilaFlows
CAU Christian-Albrechts-Universität zu Kiel
San Francisco State University
Università degli Studi di Pavia
redhat
DELL
BSC Barcelona Supercomputing Center Centro Nacional de Supercomputación
NOVATEC

Elasticity vs. Scalability



What is the relationship between the term **elasticity** (E) and the more classical term **scalability** (S) ?

A: E is a modern buzzword for S

B: E is a prerequisite for S

[✓ C: S is a prerequisite for E]

D: The terms are orthogonal

Thank You!

skounev@acm.org

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