2nd DataStorm Big Data Summer School

Big Data Infrastructures

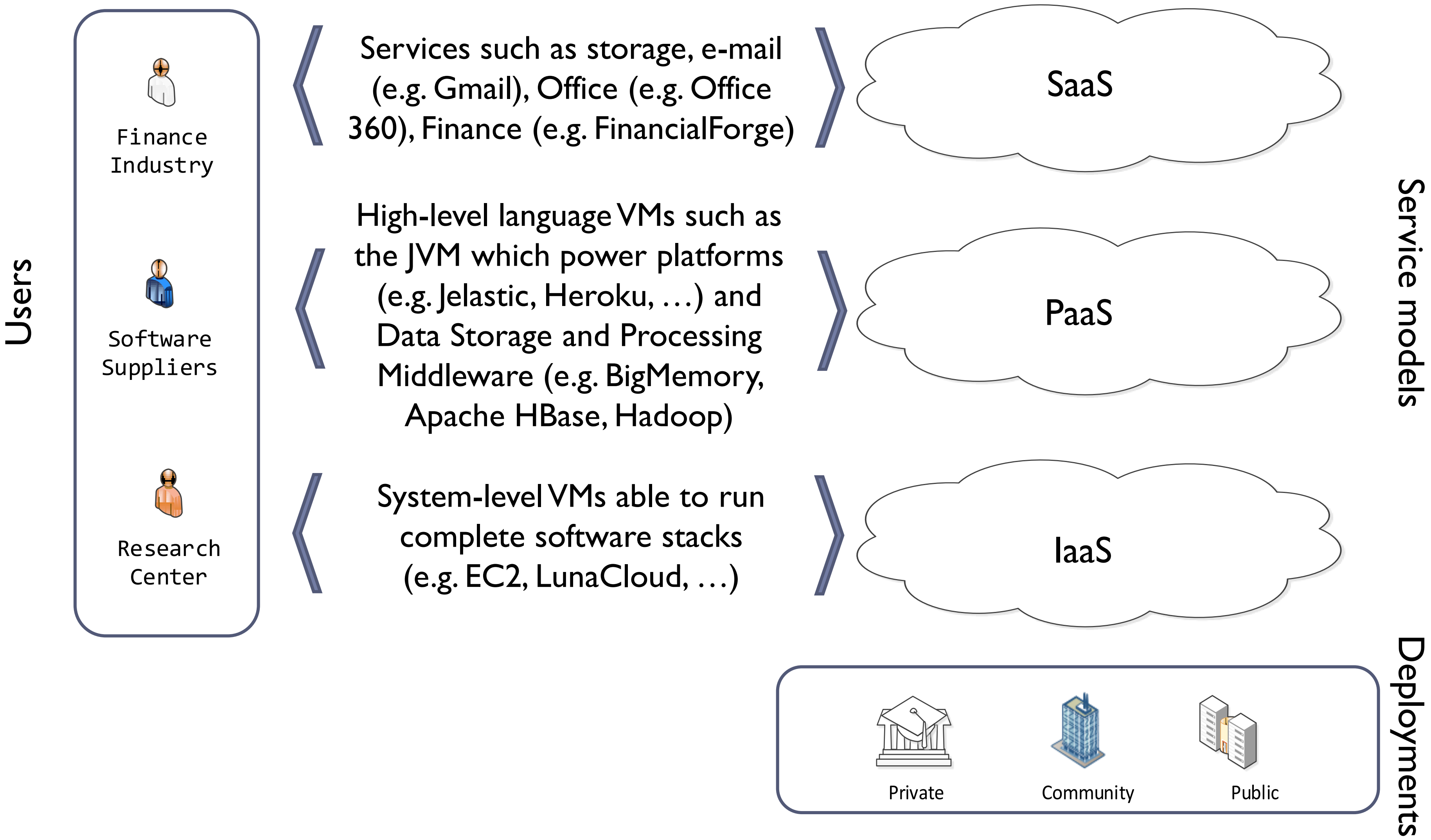
Economics of Resources, Energy, Data, and Applications

Luís Veiga

Distributed Systems Group
INESC-ID Lisboa
Instituto Superior Técnico
Universidade de Lisboa

July 2015

A *day* in the Clouds



Main challenges

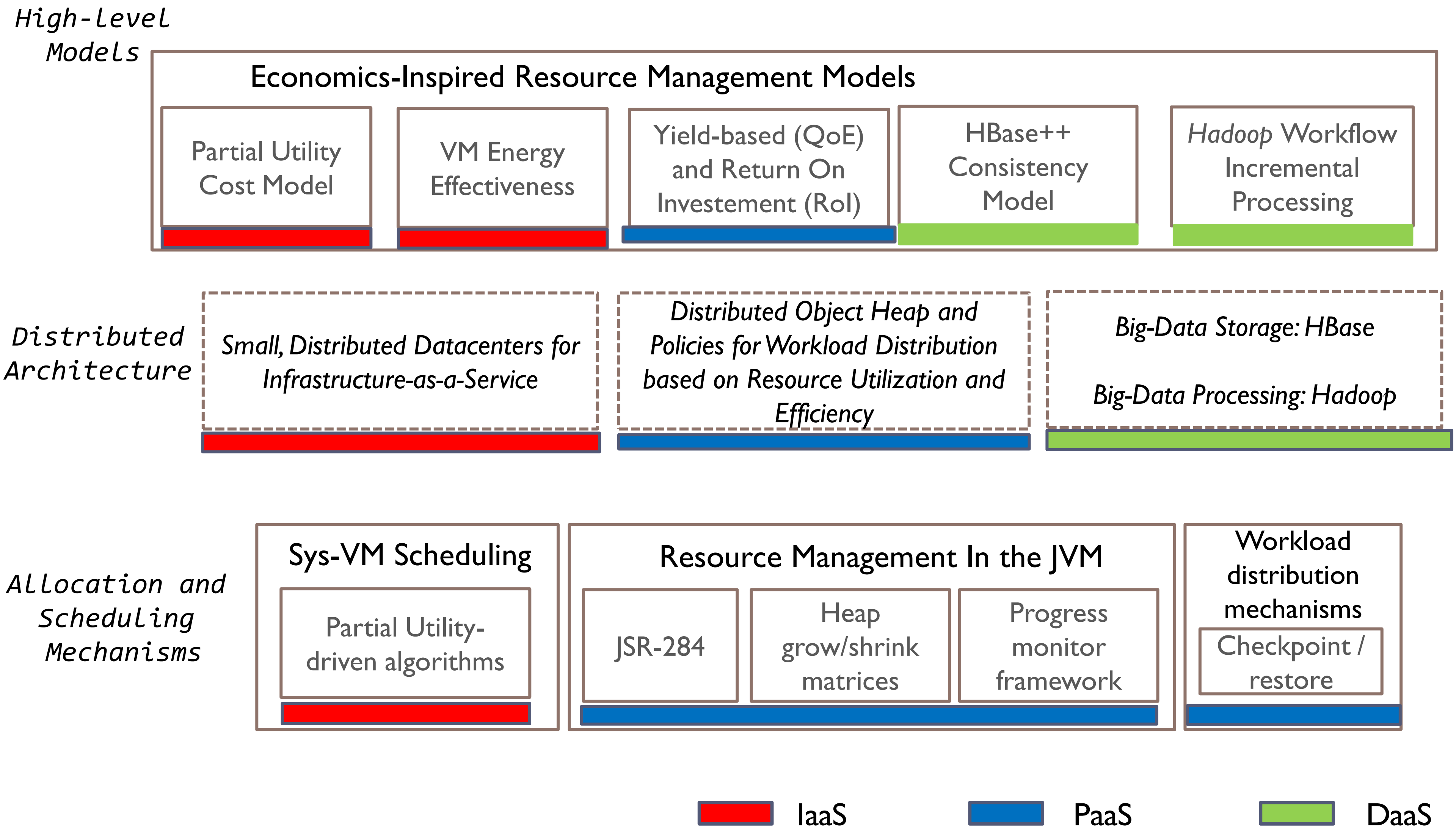
- **In general…**
  - Providers aim to maximize clients' satisfaction while minimizing operational cost
  - But, some defend the infant cloud market is an *oligopoly* and not fully passing the benefits to the client

- **IaaS (Infrastructure-as-a-Service)**
  - In public, but mostly in community an private clouds, all-or-nothing resource allocation is not flexible enough
  - multi-level SLA agreement can foster competition and enlarge the market
  - Energy and environmental footprint become prime concerns

Main challenges

- **PaaS (Platform-as-a-Service)**
  - Big-Data and e-Science applications increasingly depend on language runtimes (e.g. Java VM, .NET CLR, Scala,…)
  - Resource allocation should be tailored to the applications, taking into account the effective progress of the workload
- **DaaS (Data-as-a-Service)**
  - Big-Data and e-Science applications more and more centered on:
    - Storage: no/new-SQL distributed cloud data storage (Big-Data)
      - HBase, Cassandra, Dynamo, …
    - Processing: using (typically also Java) frameworks
      - Hadoop / Map-Reduce, Workflows of MR jobs, Pig Latin, …
  - Underlying infrastructure and resource management should fit data and application behavior *effectively* and *efficiently*

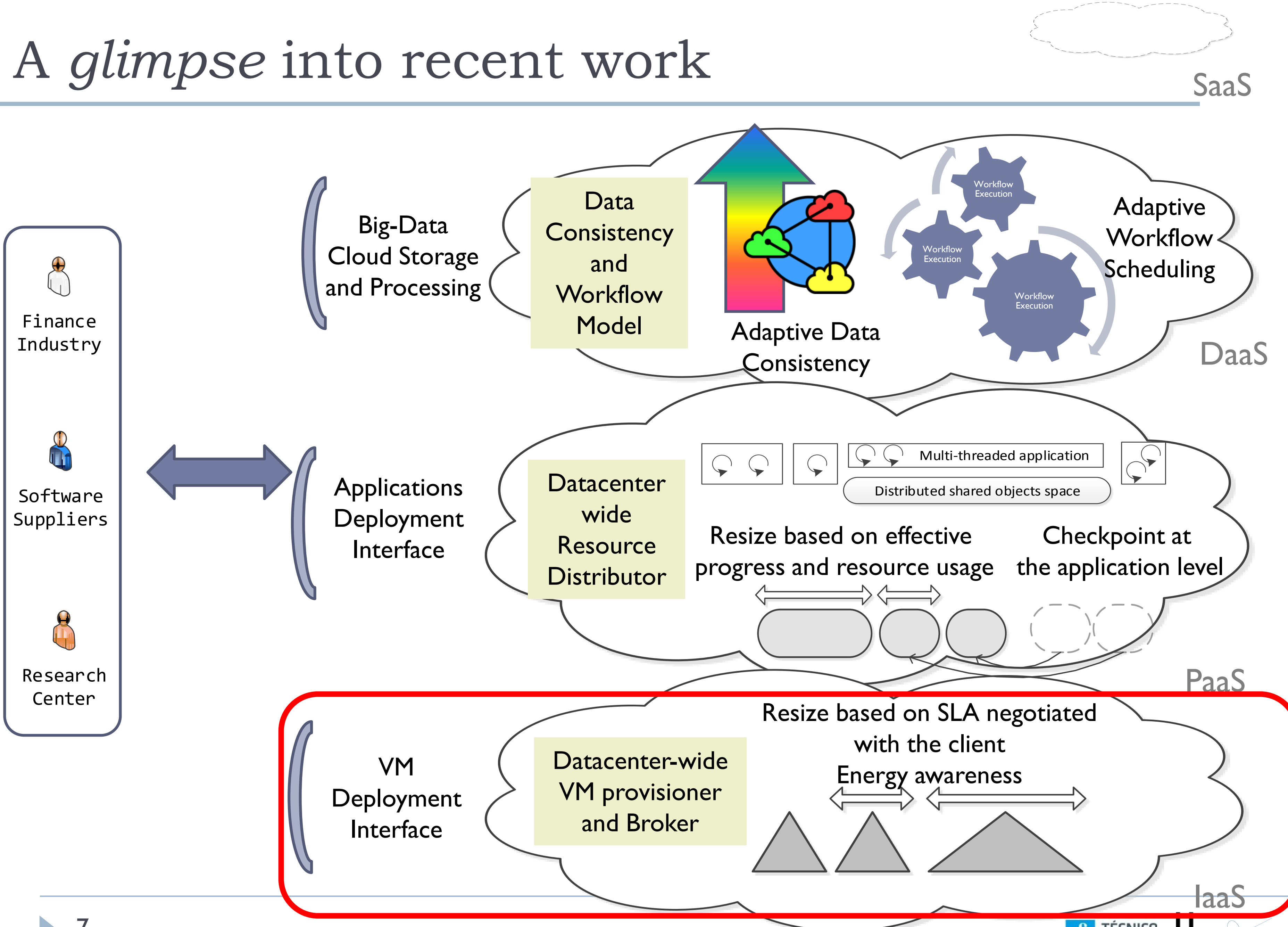
Layered view of the researched topics



Talk Outline

- Introduction
- IaaS
  - Models
  - Mechanisms
  - Evaluation
- Energy and Community Clouds
  - Models, Mechanisms, Evaluation
- PaaS
  - Models, Mechanisms, Evaluation
- DaaS
  - Models, Mechanisms, Evaluation

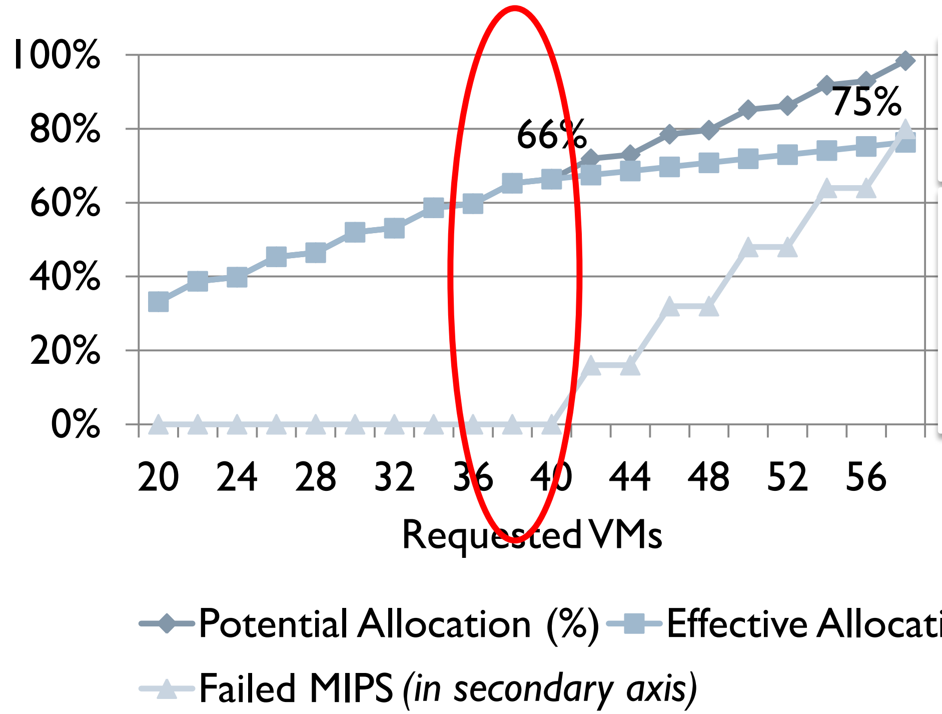
A *glimpse* into recent work



Life in a small (classic) datacenter

H_{type}	A	B
Cores	2	2
Hz	1860	2660
Mem (Gb)	4	4
# Hosts	10	10

VM_{type}	$x10^3$ MIPS
Small	0.5
Medium	1
Regula	2
Extra	2.5



25% of resources are idle (wasted)

31% of VM requests are rejected

- In summary, clients are not satisfied but datacenters are not fully utilized
  - Idle machines consume ~70% of peak power

Research at the IaaS level - overview

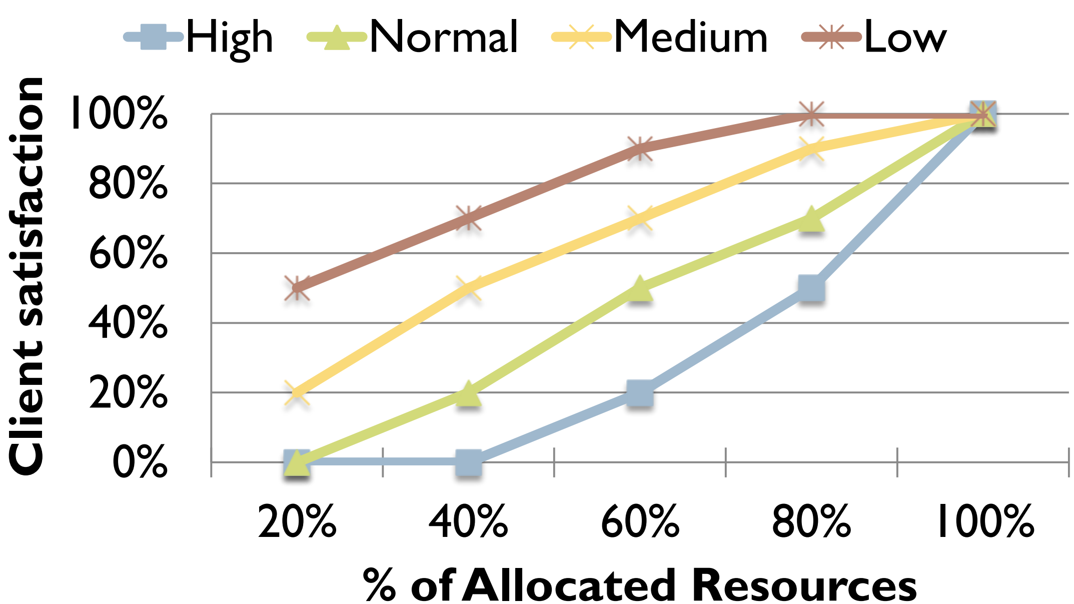
- An **architectural extension** to the current relation between cloud users and providers, particularly useful for private and community cloud deployments;

- A **cost model** which takes into account the clients' partial utility of having their VMs depreciated when in overcommit;

- **Strategies** to determine, in a overcommitted scenario, the best distribution of workloads (from different classes of users) among VMs with different execution capacities, aiming to maximize the utility of the allocation.

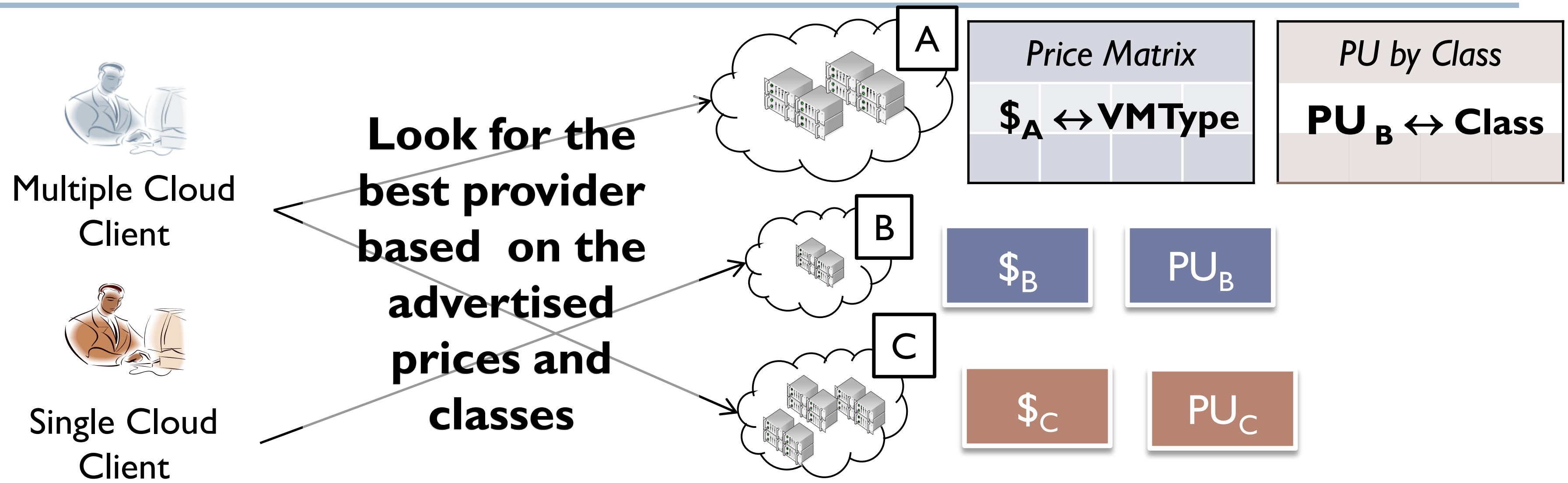
Exploring the remainder "25%"

- Base scenario: A new VM is requested but no space is available without some kind of degradation – results in a *VM rejection*
- Our proposal: Use the user's *partial utility specification*, to explore a degradation factor for each allocated VM



- Provider wants to *maximize VM allocations* while *maximizing clients' satisfaction*

A new cost model



$$VMCost(vm) = Pr(VMType(vm)) * (1 - Df(vm)) * Pu(Df(vm))$$

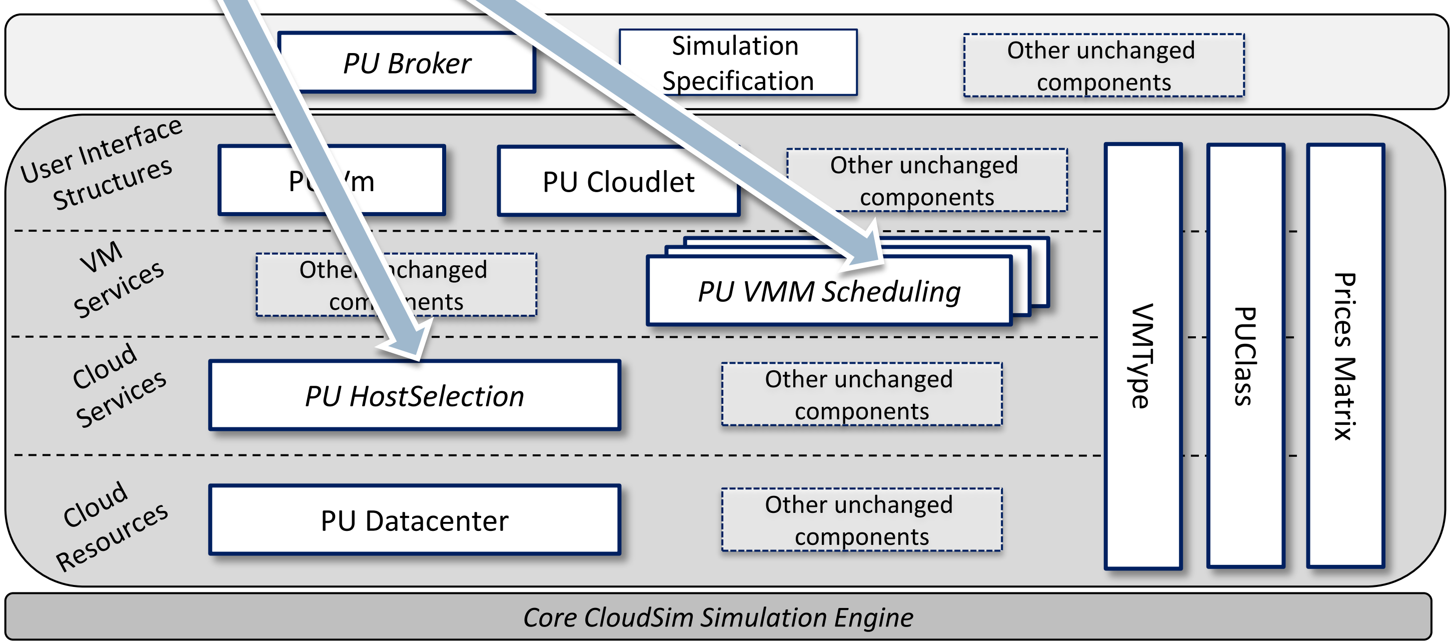
- Price of *vm* based on computational capacity
- VMs are sorted by computational power

- Depreciation factor of *vm*
- *Df(vm)=0* if provider can assign maximum resources

- Partial-utility of client based on the depreciation
- It varies based on the client class

IaaS Scheduling Algorithms

- Resources of requested VMs are changed according to multi-level partial-utility negotiation between the client and the provider
- Heuristics used by the provider
  - Sort hosts by computational power and increasingly take from allocated VMs
  - Asymptotic cost bellow quadratic: *O(nr_hots · nr_vms · lg(nr_vms))*
- Extension to CloudSim – highly cited/used cloud simulation framework

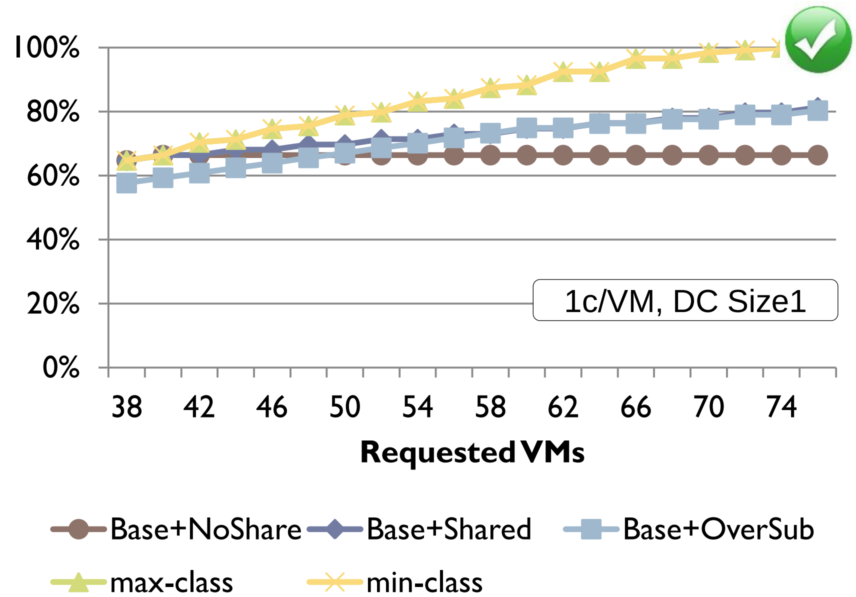


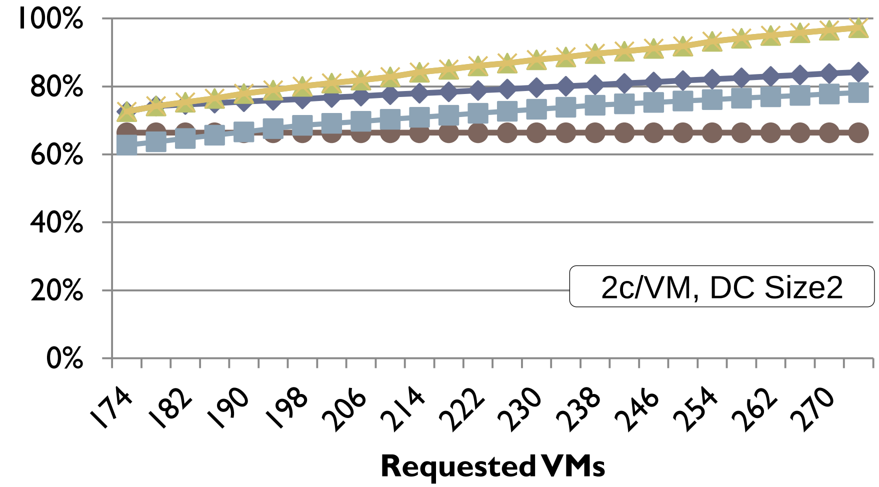
[1] JS, LV @ Transactions on Cloud Computing, IEEE, 2014
[2] JS, LV @ IEEE, CloudCom, 2013, Best Paper runner-up

Evaluation

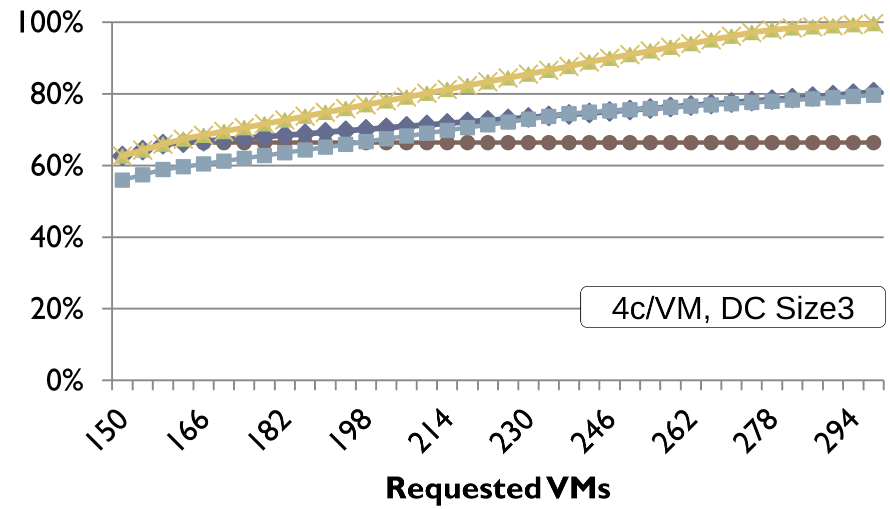
- Questions regarding this cost model and algorithms
  - Q1: Resource usage increases? (provider interest)
  - Q2: Revenue increases? (provider interest)
  - Q3: Impact on the workload execution time (client interest)
  - Transversal: How does this approach scale?
    - DC_1 (2 Cores) DC_2 (4 Cores) and DC_3 (4 Cores+HT)
    - VMs requesting 2 Cores and 4 Cores
- Evaluated with traces from VMs running in PlanetLab collected in the context of the CoMon project
  - A trace from a PlanetLab VMs is assigned to each VM in the simulation

Q1: Resource Usage

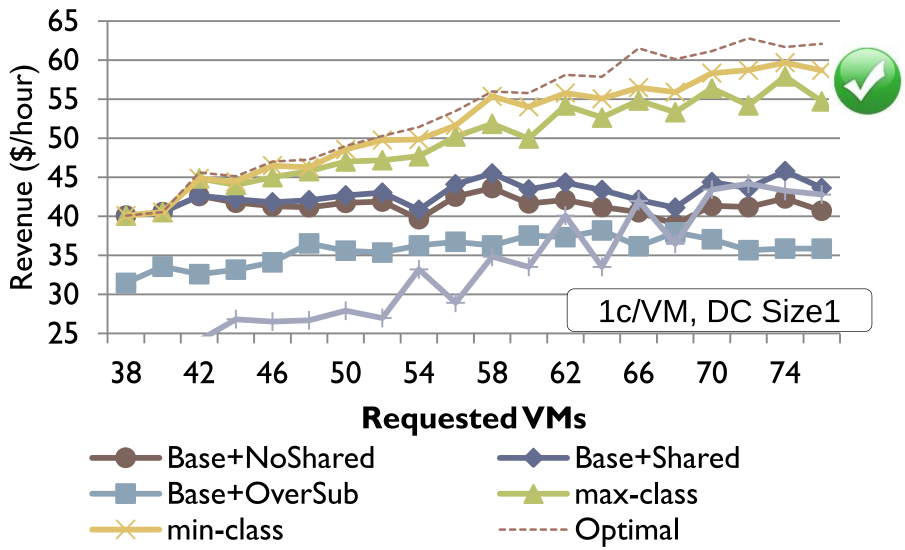


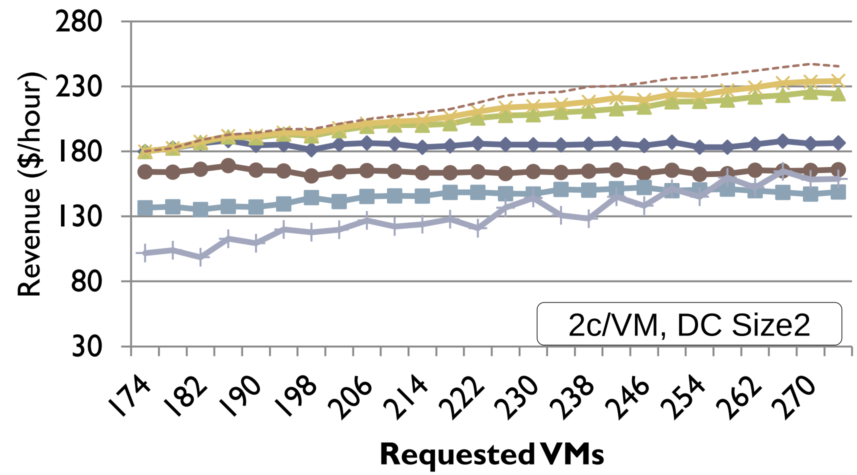


- Utility-driven approaches
  - achieves better resource utilization, while allocating all VMs
  - reach the peak in a similar fashion, across all sizes of datacenters.

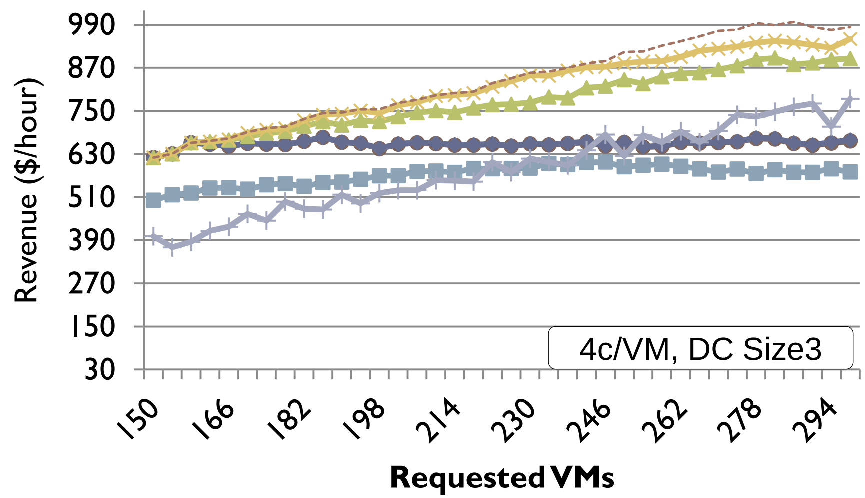


Q2: Revenue

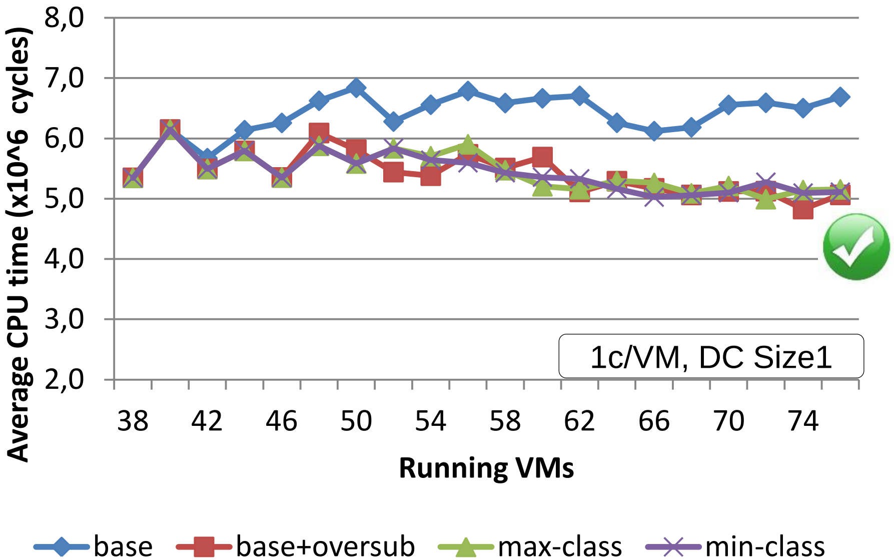


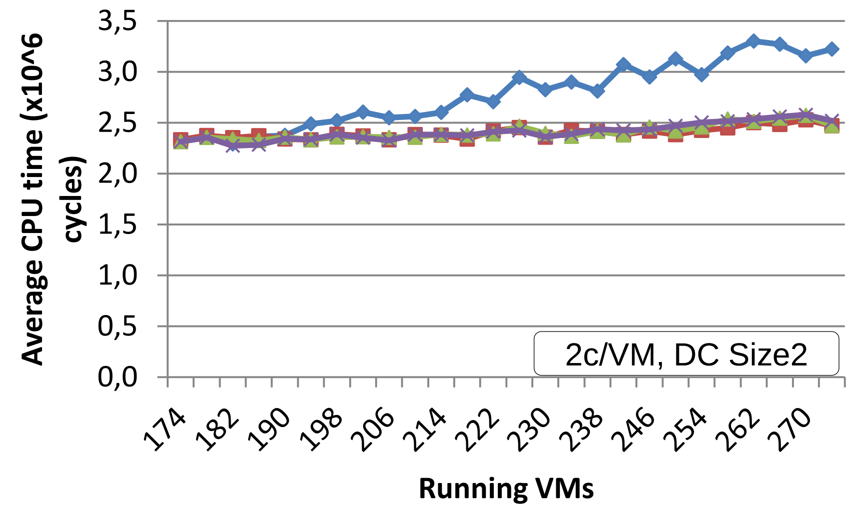


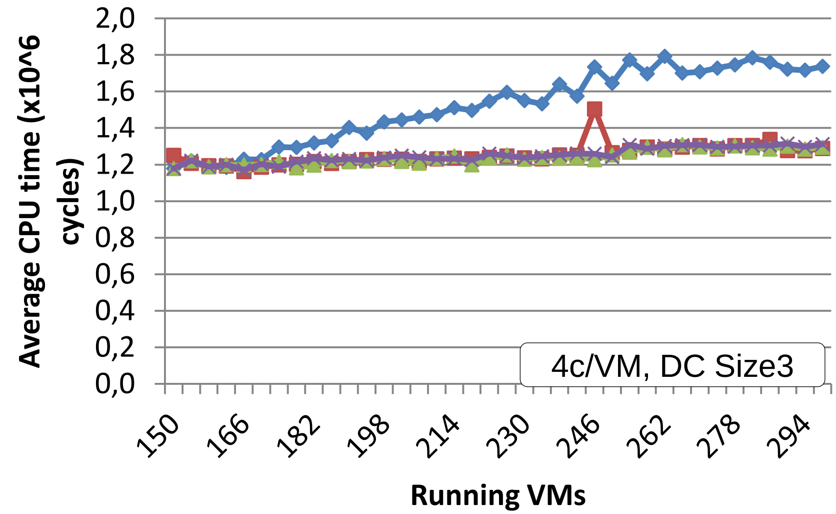
- Revenue increases with more VMs allocated
- What would be rejected VMs are accepted with a partial utility-driven allocation



Q3: Impact in workloads’ execution time





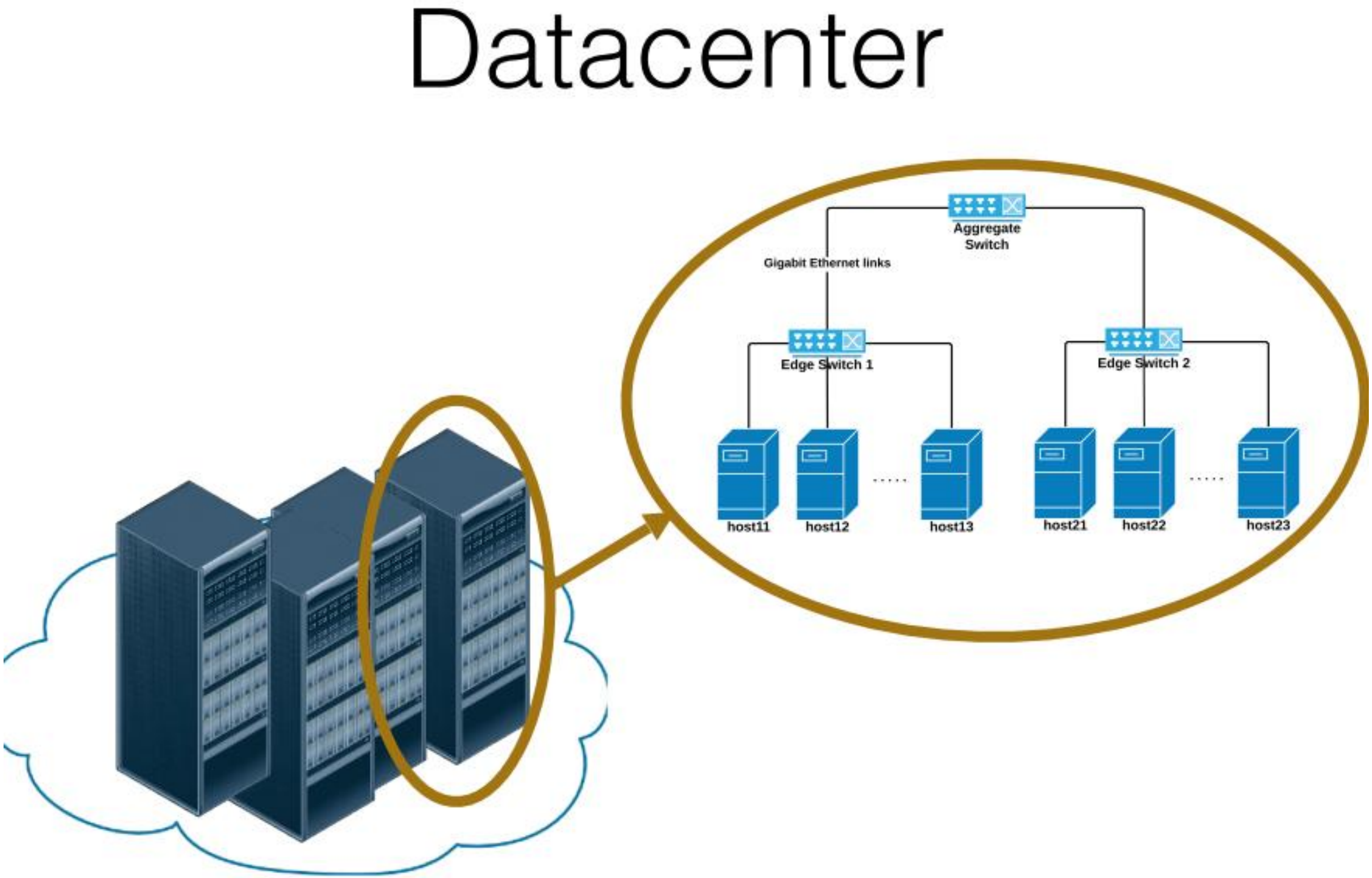


- *With more VMs allocated*, even if with less allocated resources than the ones requested, as it is the case, *average execution time is below the execution times achieved with the base strategies.*

Talk Outline

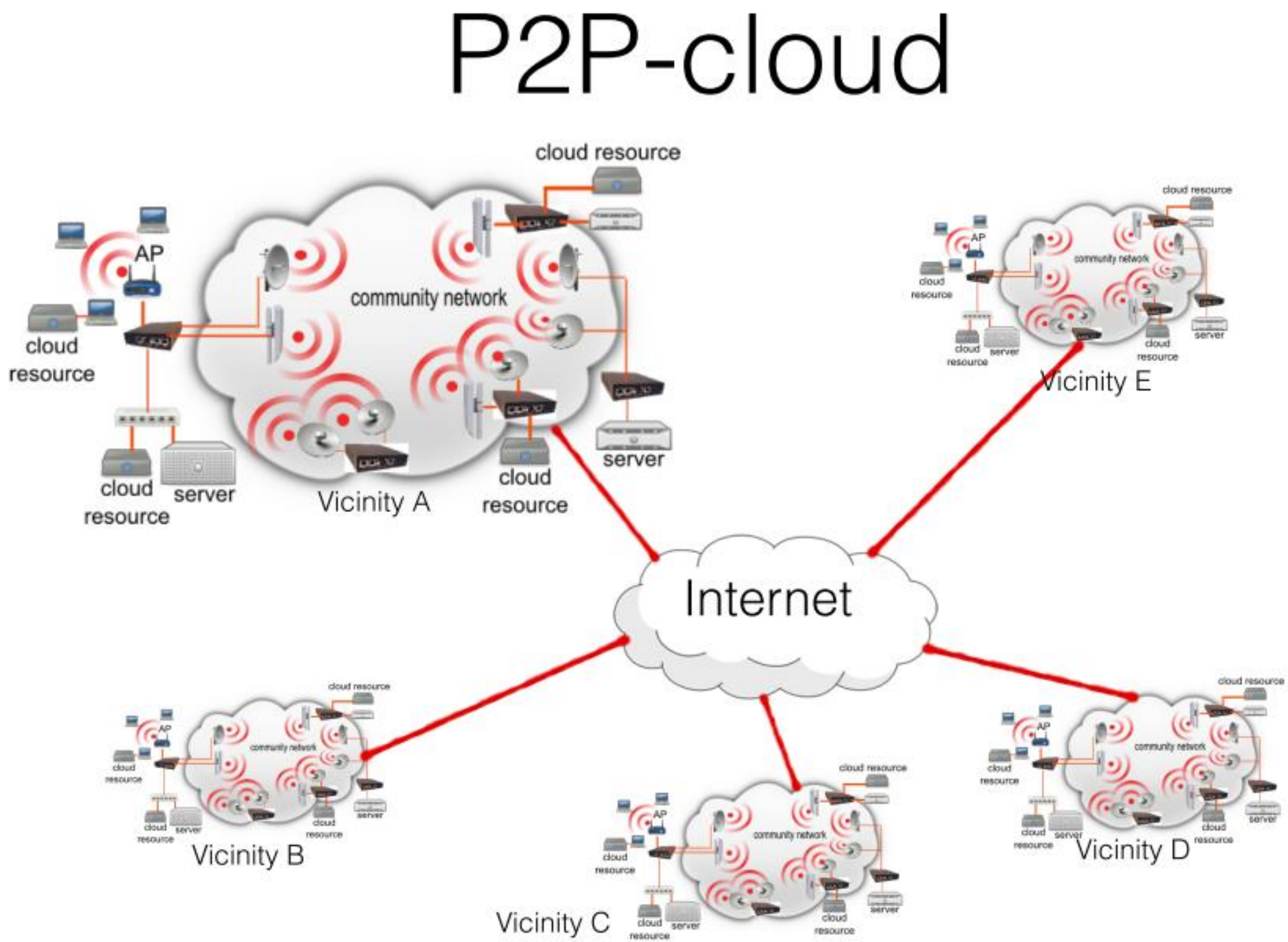
- Introduction
- IaaS
  - Models, Mechanisms, Evaluation
- Energy and Community Clouds
  - Models
  - Mechanisms
  - Evaluation
- PaaS
  - Models, Mechanisms, Evaluation
- DaaS
  - Models, Mechanisms, Evaluation

Life in the **Corporate Clouds**

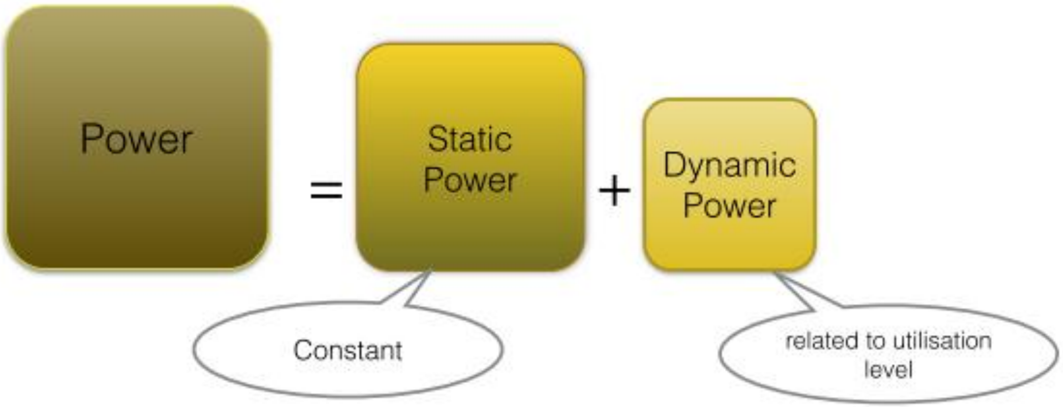


[3] LS, FF, LV @ IEEE CloudCom 2014 Best Paper Candidate

Life in Peer-to-Peer **Community Clouds**



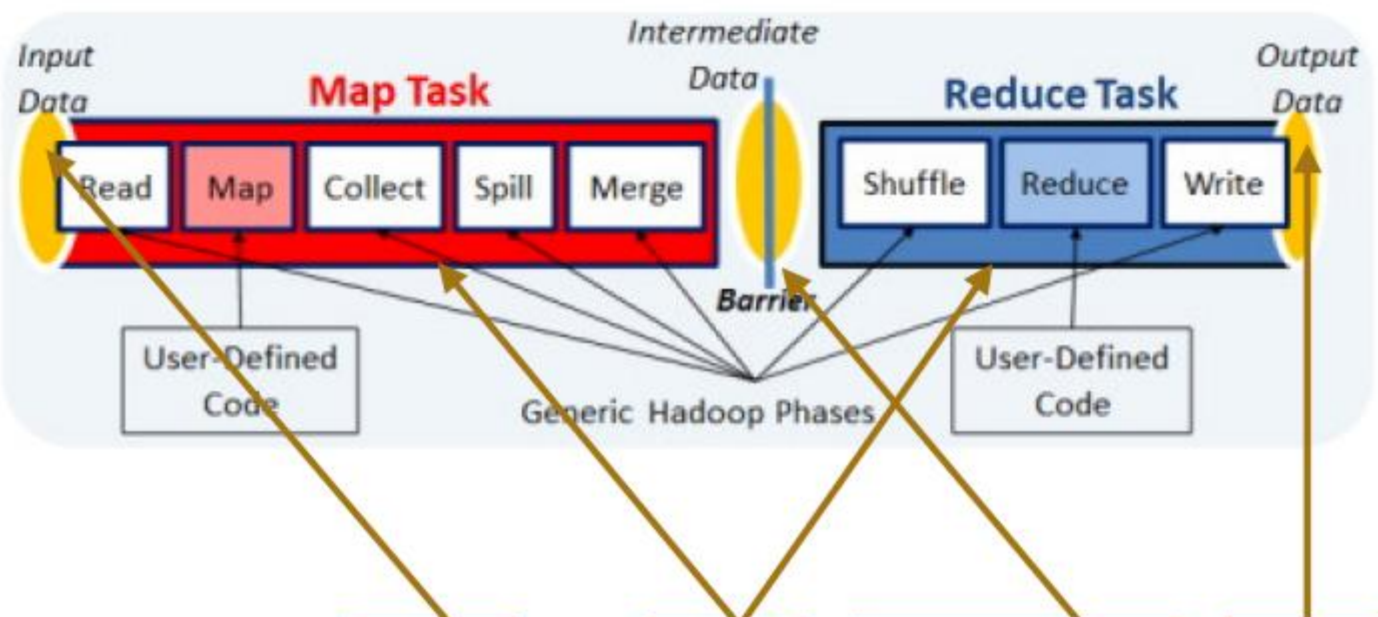
General Power Model



Linear Power Model: $$P_{Linear} = P_{static} + (P_{\max} - P_{static}) \times U$$

Real world workloads resource consumption

MapReduce case study



$$N^{MR}_{msg} = MTU_p \times \frac{Size_{input} + n_{hosts} \times Size_{VM} + Size_{intermediate\_output} + Size_{output}}{MTU_p - Overhead_p}$$

Model real world workloads energy usage

MapReduce Energy Model

intra-datacenter

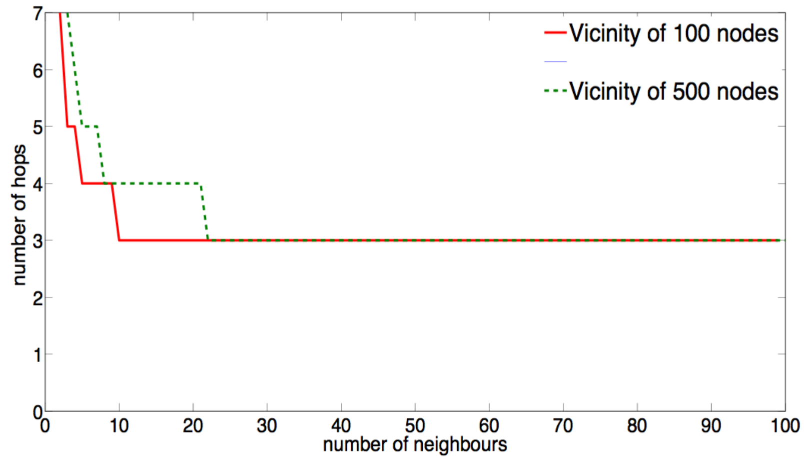
$$E^{MR}_{intra\_DC} = P^{Com}_{intra\_DC}(Size_{msg}) \times t_{send} \times N_{msgs} + \sum_{i=1}^{phases} (E[task]_i \times \sum_{nt} P_{host})$$

Communication — Process

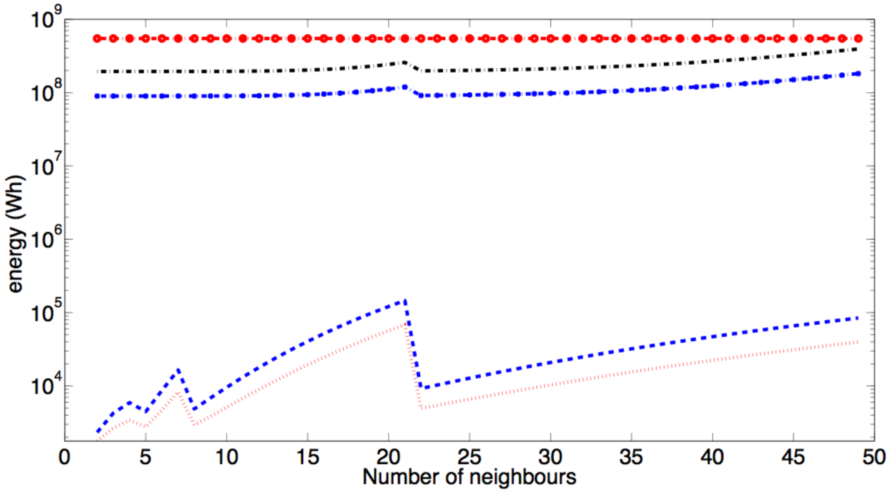
$$E^{MR}_{intra\_P2P} = P^{Com}_{WN}(Size_{msg}) \times t_{send} \times N_{msgs} + \sum_{i=1}^{phases} (E[task]_i \times \sum_{nt} P_{host})$$

intra-vicinity P2P-cloud

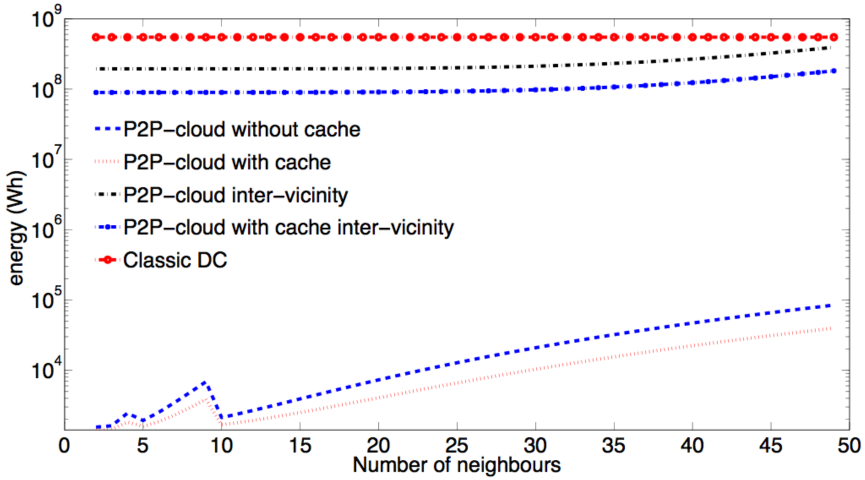
Evaluation: Vicinity Density effect



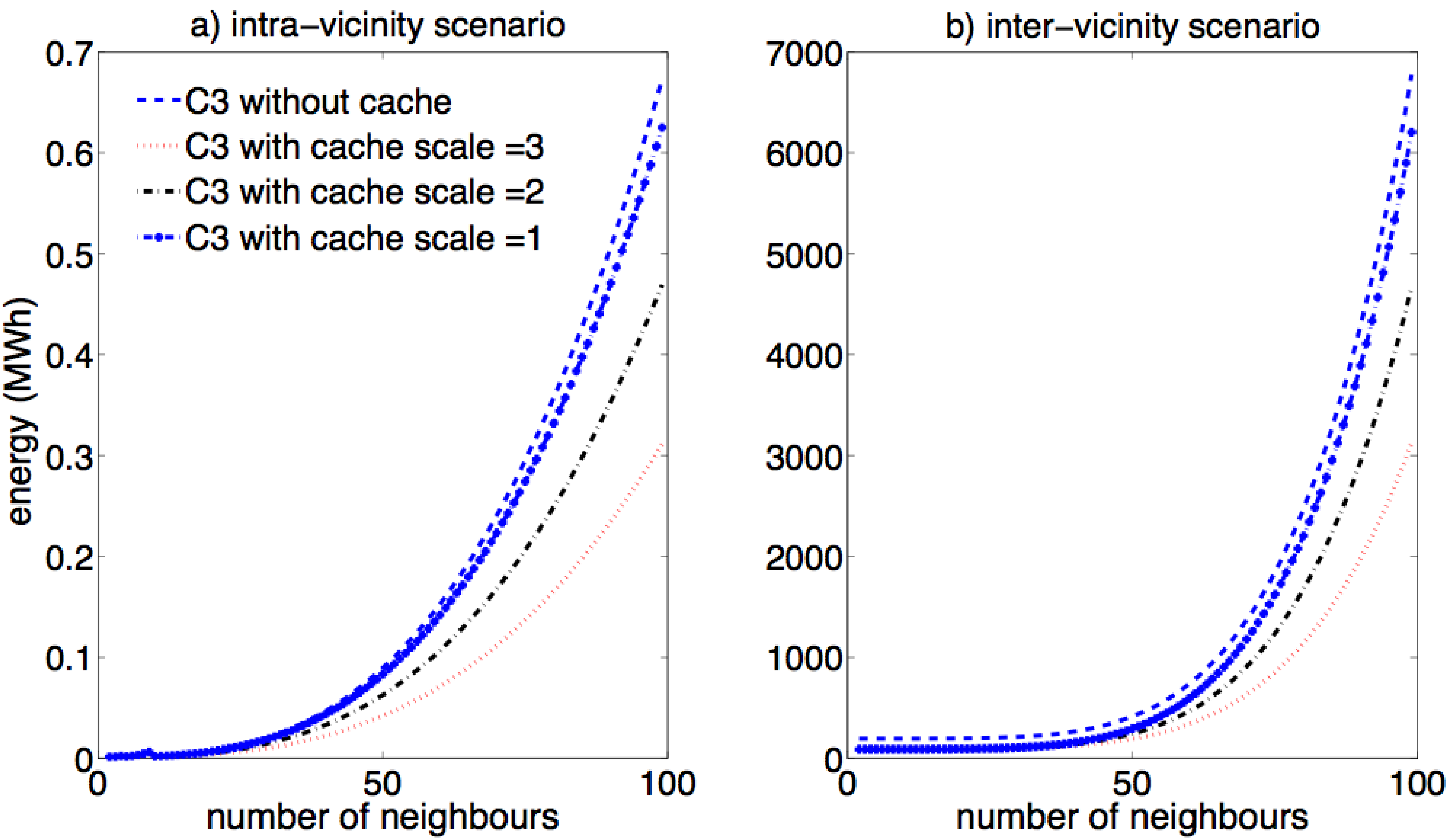
Vicinity of 500 nodes



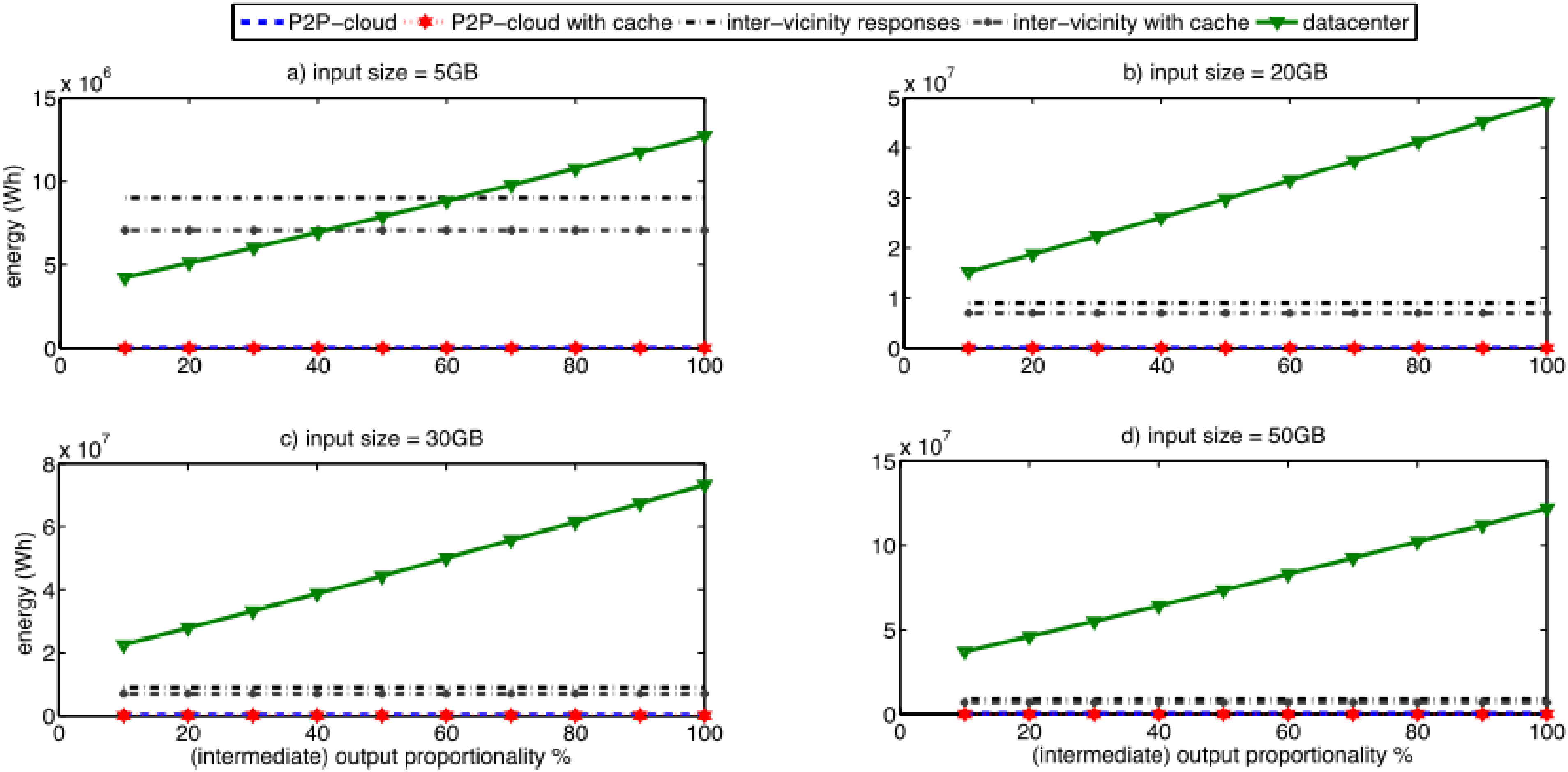
Vicinity of 100 nodes



Impact of cache scale



Input-(Intermediate) Output proportionality



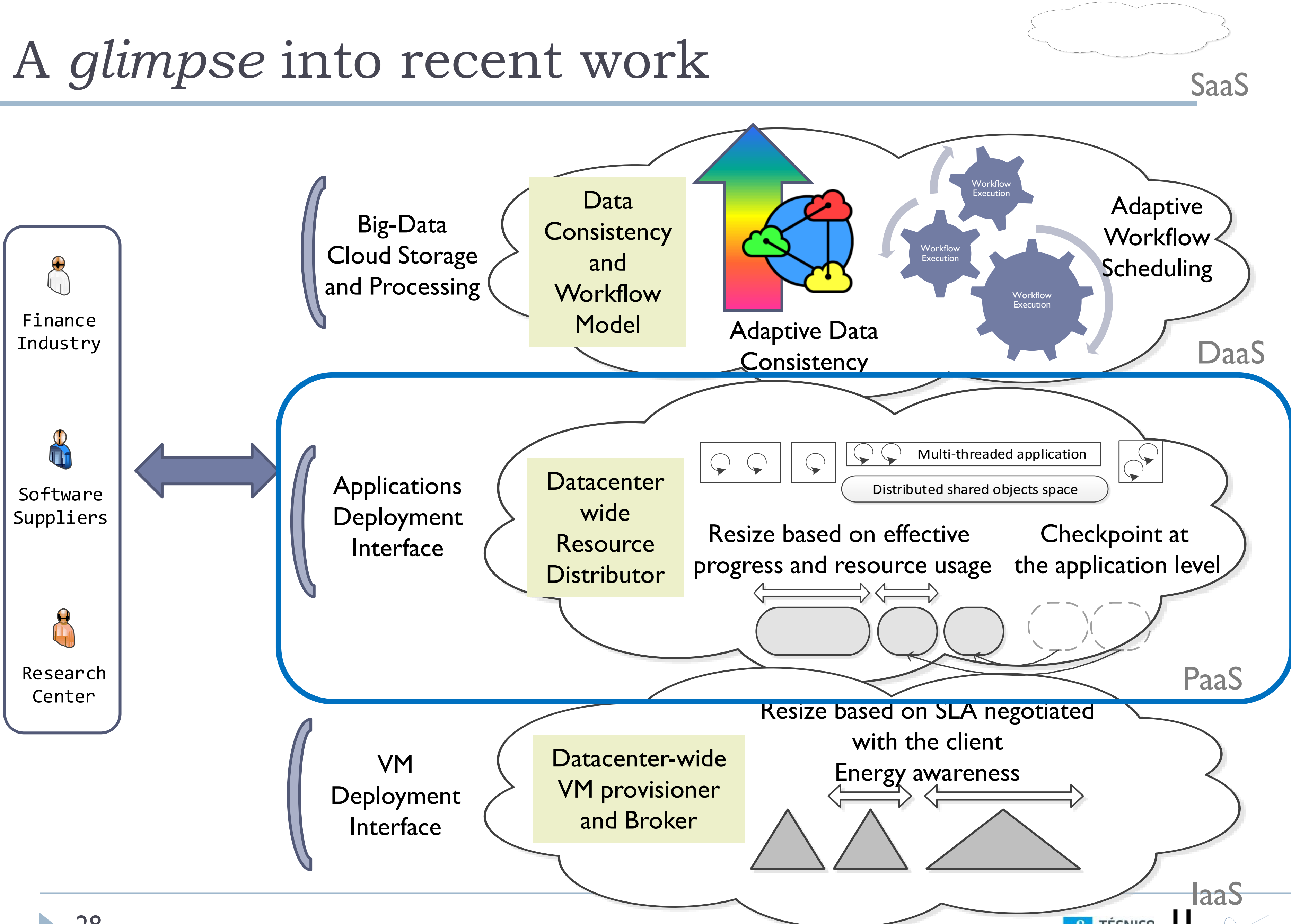
Energy Take Aways

- P2P-cloud can provide an ecosystem for energy efficient decentralised clouds
- Intra-vicinity supply is the most energy efficient.
- Trade-off between energy efficiency and resource availability- Cache mechanism
- However:
  - Not easy to support large VM
  - Performance degradation
- Looking forward:
  - There is room to improve the energy efficiency of P2P-cloud
  - A decision support system for energy aware resource provisioning

Talk Outline

- Introduction
- IaaS
  - Models, Mechanisms, Evaluation
- Energy and Community Clouds
  - Models, Mechanisms, Evaluation
- PaaS
  - Models
  - Mechanisms
  - Evaluation
- DaaS
  - Models, Mechanisms, Evaluation

A *glimpse* into recent work



PaaS-level motivation and goals

- How to influence an application behavior, ***effectively*** (wide range and impact), ***efficiently*** (low overhead) and ***flexibly*** (with no or little intrusive coding)?
- *Line of work*: Extend managed runtimes (e.g., Java VMs such as Jikes RVM and OpenJDK) to operate efficiently in multi-tenancy scenarios such as those of cloud computing infrastructures
  - ***Accurately monitor resource usage***
  - ***Monitor application progress***
  - ***Resource management***
  - ***Elasticity and horizontal scaling***

Economic *yield*

- ***yield*** is a return/reward from applying a given allocation strategy (S) to some resource (r)

$$Yield_r(S_a, S_b) = \frac{\text{Savings}_r(S_a, S_b)}{\text{Degradation}(S_a, S_b)}$$

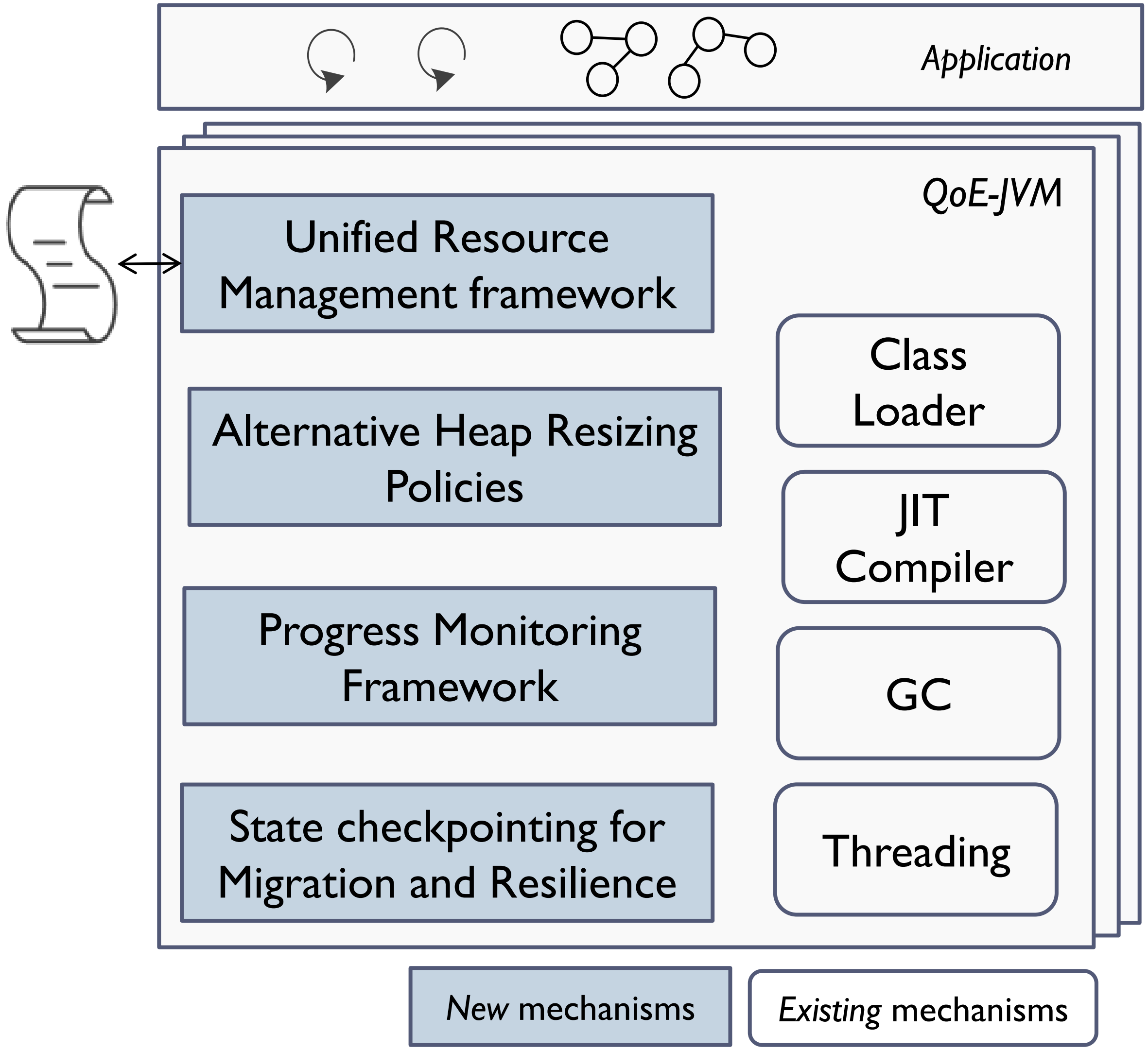
$$\text{Savings}_r(S_a, S_b) = \frac{U_r(S_a) - U_r(S_b)}{U_r(S_a)}$$

- ***Savings*** represents how much of a given resource *(r) is saved* when two management strategies are compared.
- It relates the usage (*U*) of a resource with the old and the new configuration

$$\text{Degradation}(S_a, S_b) = \frac{P(S_b) - P(S_a)}{P(S_a)}$$

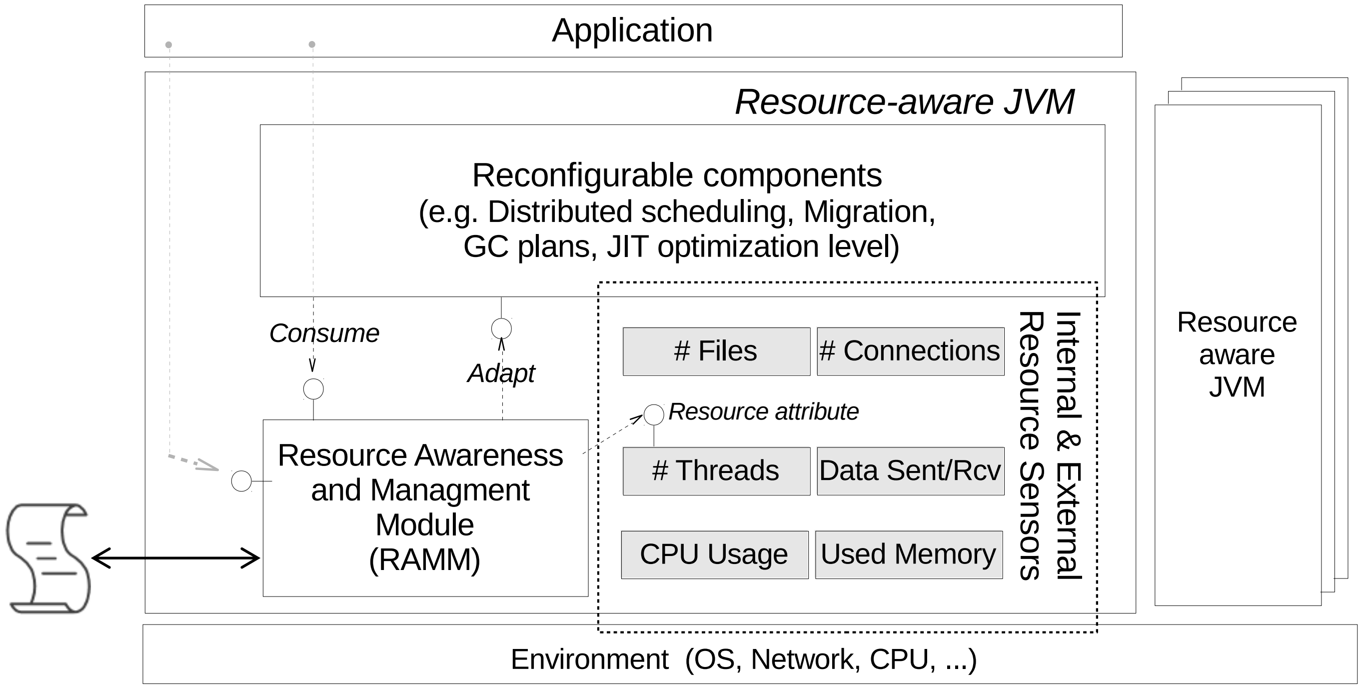
- ***Degradation*** represents the impact of the savings, given a specific performance or progress metric (e.g. execution time).
- It relates the progress (*P*) made with the old and the new configuration

PaaS Mechanisms



- Mechanisms incorporated in Jikes RVM, «winner of the *ACM SIGPLAN Software award*, cited for its "high quality and modular design"» in http://en.wikipedia.org/wiki/Jikes_RVM
- Progress monitor supported on Java instrumentation agent infrastructure

Unified Resource Management Framework

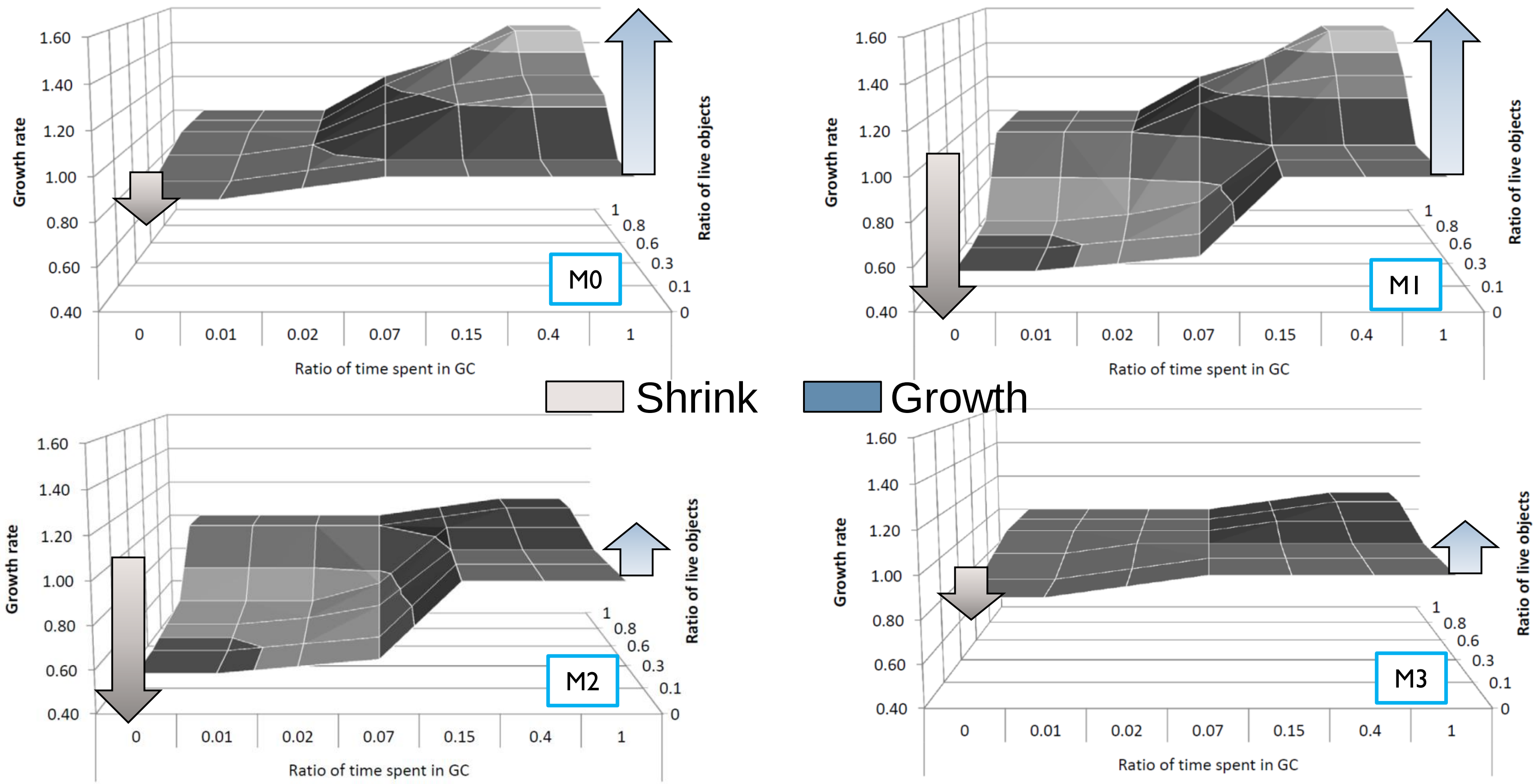


- Extension of Jikes RVM, and the GNU classpath, with JSR 284 – The resource management API
- Monitoring and enforcement points include
  - Memory allocation (heap growth rate), CPU usage, Thread creation

[4] JS, JL, LV @ CoopIS 2011, LNCS
[5] JS, LV @ DOA-SVI 2012, LNCS

GC Heap Policies: Base and alternatives

- Garbage Collection Economics in Jikes RVM
  - *heap growth rate driven by wasted CPU on GC*



[6] JS, LV @ CSSE, CRL Publishing, 2013
[5] JS, LV @ DOA-SVI 2012, LNCS

Progress Monitoting framework

```
@Retention(RetentionPolicy.RUNTIME)
@Target({ElementType.METHOD,
         ElementType.FIELD,
         ElementType.PARAMETER})
public @interface Progress {
  double relevance() default 1.0;
}
```

(a)
Definition

Call rates
updater

update

```
public class AClass {
  @Progress(relevance=0.
  public void m1() { ... }

  @Progress(relevance=0.2)
  public void m2() { ... }
}
```

(b)
Usage

update

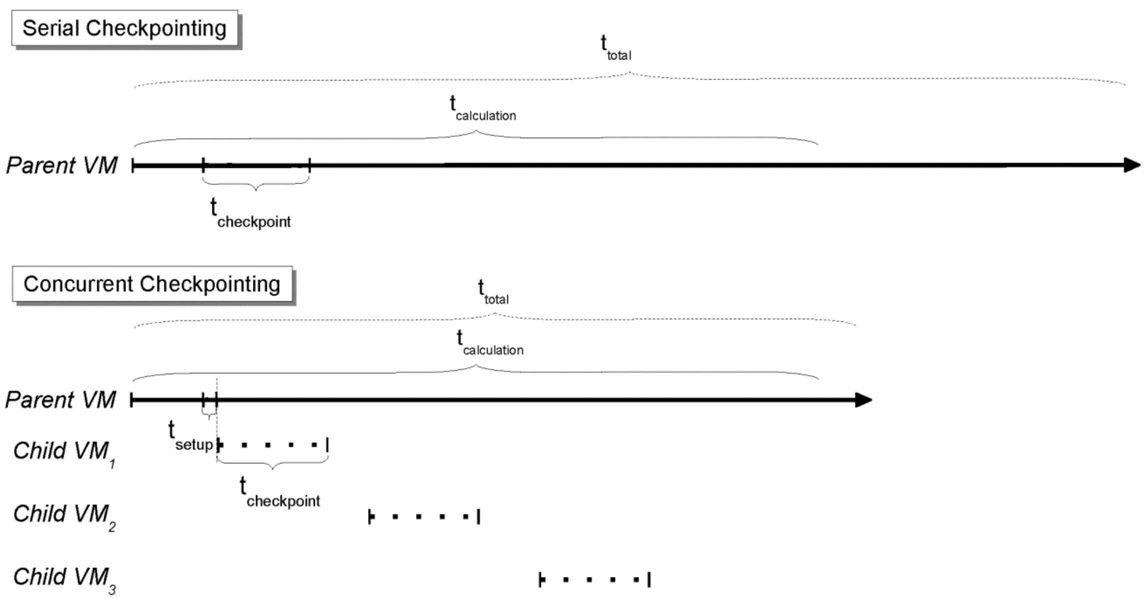
Method	MethodStats
m1	*Counters, Call rates*
m2	*Counters, Call rates*
...	

- Annotations are used at load time to insert measurement code (by an instrumentation agent)
- Measurements: overall call rate, window call rate (last *n* ms.)

[7] JS, LV @ ACM SAC 2013

Checkpointing for application-level migration

- Serial checkpoint needs to:
  - 1. Stop all running threads, 2. Build method descriptors, 3. Save execution state (i.e. stack frames), 4. Save graph of reachable objects
- Concurrent checkpoint makes the two final steps in parallel with the application
- Relies on on-stack-replacement, serialization and fork technologies
- Limitations
  - JNI code that touches heap managed objects

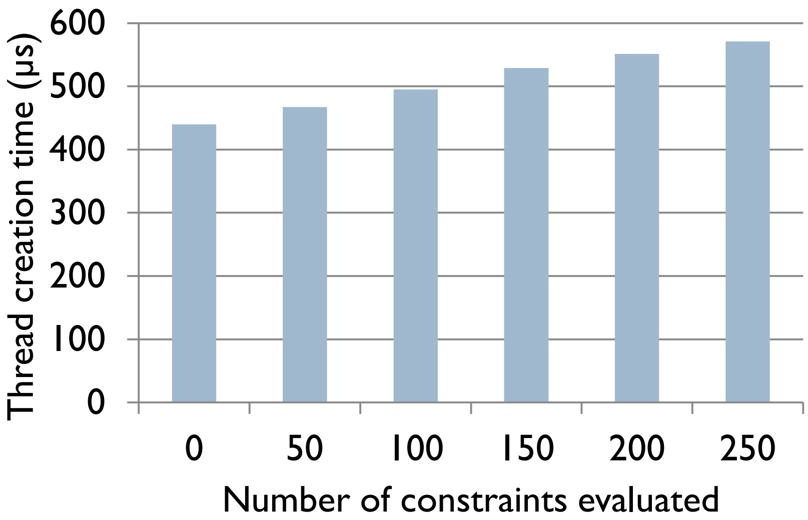


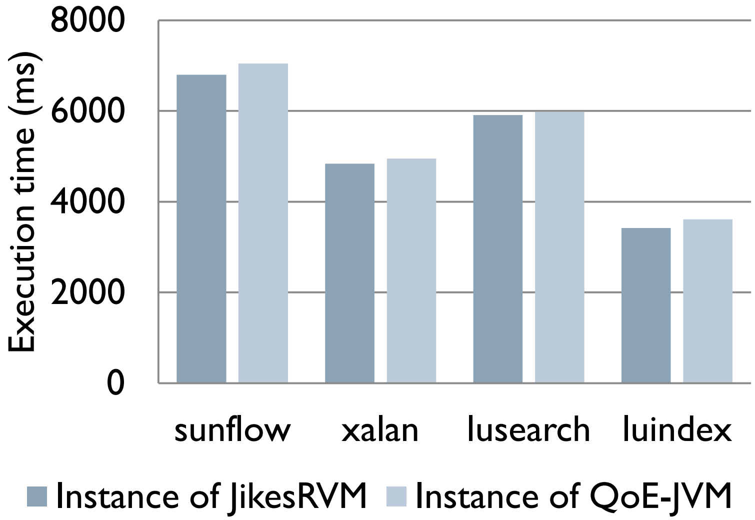
[8] JS, TG, LV @ CCPE, Willey , 2012

Evaluation

- Questions regarding these extensions
  - Q1: How costly is to account *resource usage* and *execution progress*?
  - Q2: What are the benefits of applying application-tailored policies (e.g. heap policies)?
  - Q3: Which are the costs and benefits of concurrent checkpoint?
- Evaluated with Dacapo benchmarks
  - Each benchmark explores a different aspect of a Java VM, as shown with a principal components analysis using metrics that architecture, code, and memory behavior

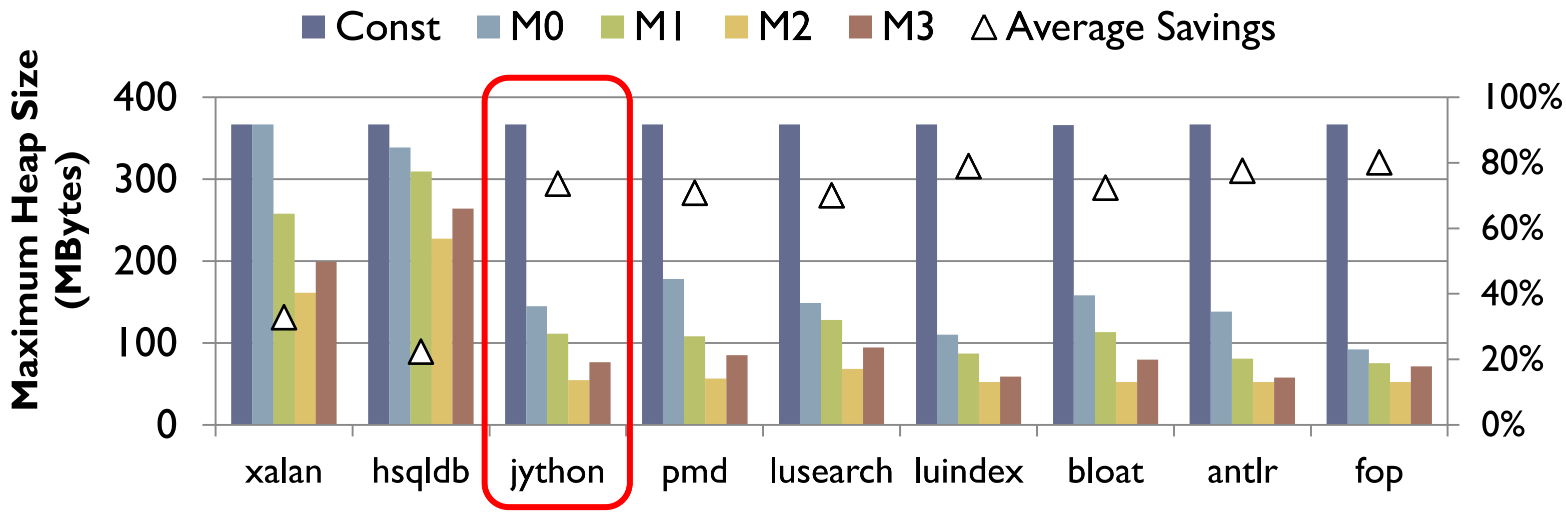
Q1: Accounting *resource usage* and *execution progress*?





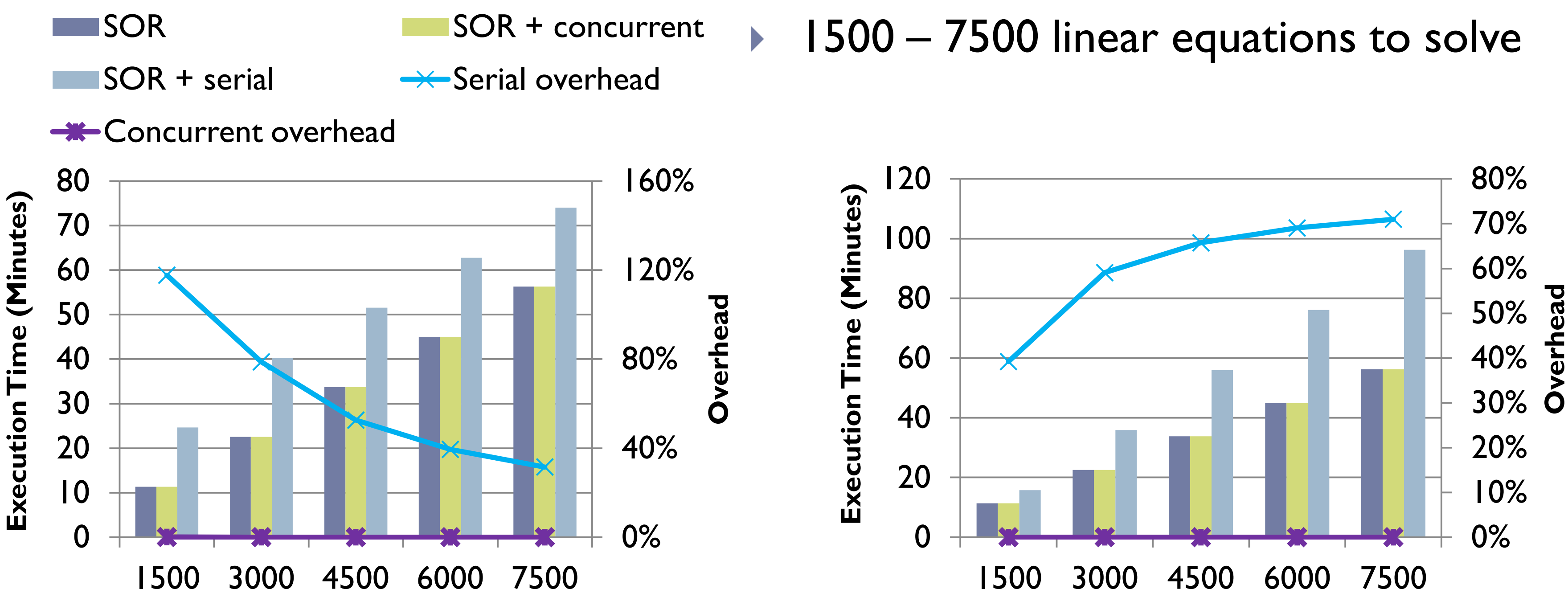
- **Policy evaluation overheads** (for *resource domain* thread creation):
  - +6% to the baseline using a (complex) policy with 50 constraints
  - +3% (average) overhead in real multi-threaded applications
  - The accounting of other resources (mem, cpu) also shows very small overhead
- **Progress monitoring related overheads** (using complete version of *Sunflow*)
  - At load time: +105 ms
  - At run time: +0.5%

Q2: *Yield* applied to heap management



		xalan	hsqldb	jython	pmd	lusearch	luindex	bloat	antlr	fop
Degradation in Execution time	M0	-1.5%	-1.2%	17.6%	8.6%	20.3%	9.1%	14.5%	3.9%	-2.1%
	M1	7.4%	-3.5%	2.5%	7.7%	26.1%	14.7%	12.4%	24.8%	-3.2%
	M2	11.2%	50.9%	80.5%	43.7%	225.5%	26.9%	17.7%	31.0%	18.7%
	M3	7.6%	17.0%	13.1%	23.2%	66.2%	25.0%	-4.9%	39.4%	10.8%
Yield	M0	0.0	**-6.6**	3.4	6.0	**2.9**	**7.7**	3.9	**15.8**	**-36.3**
	M1	4.0	-4.5	**28.4**	**9.2**	2.5	5.2	5.6	3.1	-24.7
	M2	5.0	0.7	1.1	1.9	0.4	3.2	4.8	2.8	4.6
	M3	**6.0**	1.7	6.1	3.3	1.1	3.4	**-15.8**	2.1	7.5

Q3: Checkpointing mechanisms

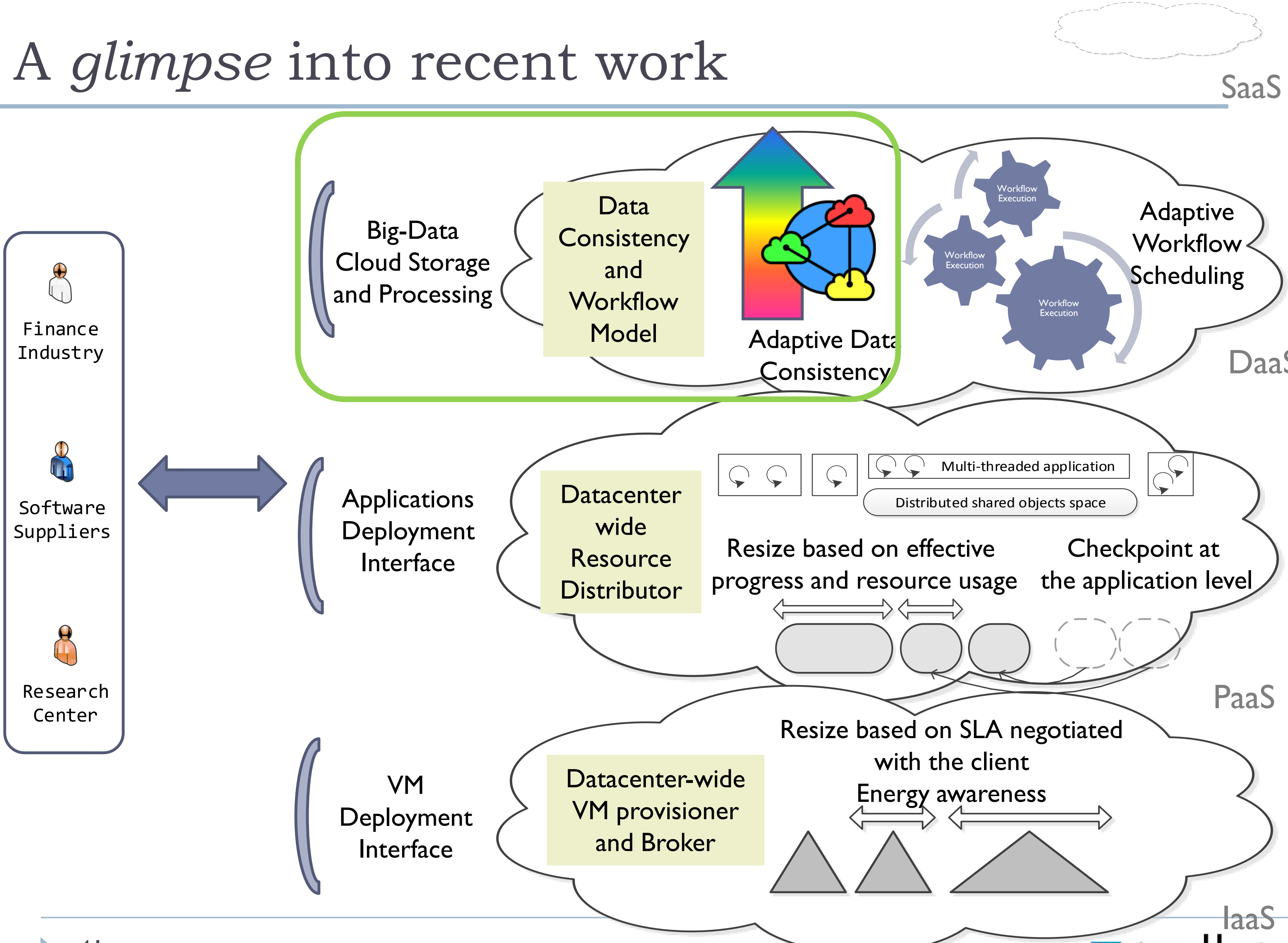


- 1500 – 7500 linear equations to solve

- Checkpoint at 20%, 40%, 60% and 80% of progress
  - Serial overhead is amortized
- Checkpoint at approximately every 5 minutes
  - Serial overhead increasingly stretches
- The overhead of concurrent checkpoint is negligible - less than 1% in all configurations

Talk Outline

A *glimpse* into recent work



Big-Data Storage Background

- Most systems provide either *strong* or *eventual* consistency, thereby allowing some level of stale data
  - which is the same for every application/subsets of data (eg, HBase, Amazon S3).
- Higher availability/performance is typically favored over strong consistency
- Lack of widespread support of differentiated consistency levels
- Coarse-grained approaches defined globally per-application, or explicitly per individual operation/function
  - not per subsets of data, within same application, with several classes of consistency

DaaS - Storage Motivation

- Current trend is distributed/micro DCs over single mega DCs
  - require more and more and selective synchronization
- Dependency on critical data stored in DCs across the world
  - requires high-availability
- Classical consistency models necessarily degrade performance
- An important class of applications can:
  - tolerate and benefit from relaxed consistency
  - bounding inconsistent access in application-specific manner
- Data semantics is not regarded to guide consistency enforcement

Goals

- Tackle well-studied tradeoff consistency vs. availability
  - explore the spectrum between pessimistic and eventual consistency
- Provide a novel consistency model with Quality-of-Service
  - tailored to geo-replication
  - *Quality-of-Service for Consistency of Data*
- Offer replication framework (VFC3)
  - enforcing QoS-consistency model on
  - distributed noSQL databases (HBase)
  - partially replicated caches of tables

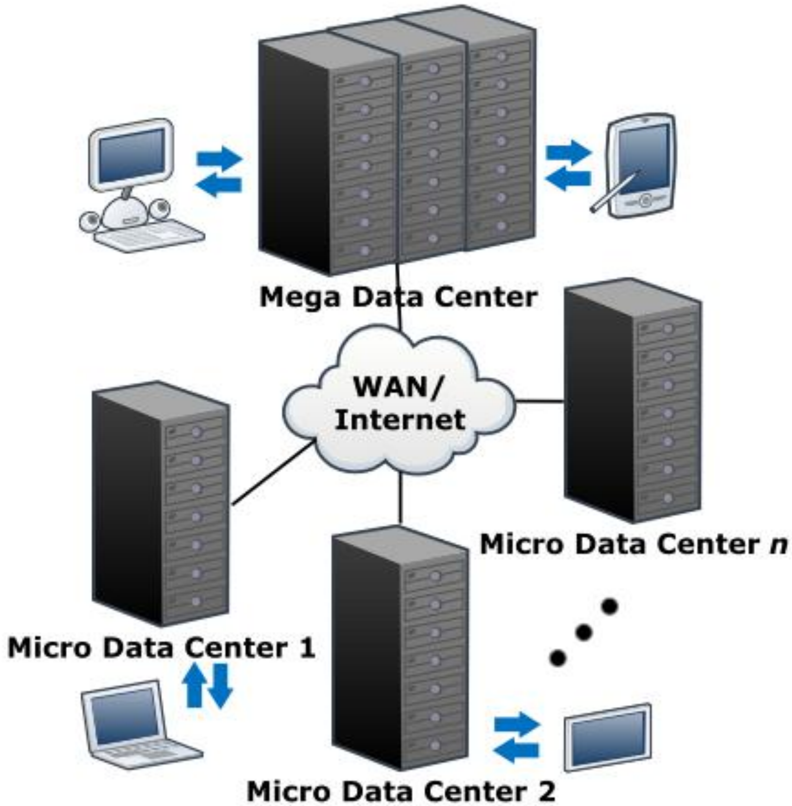
[9] SE, LV @ EuroPar 2012

[10] AGR, SE, LV @ IEEE CloudCom 2013

Data Storage in Cloud: HBase

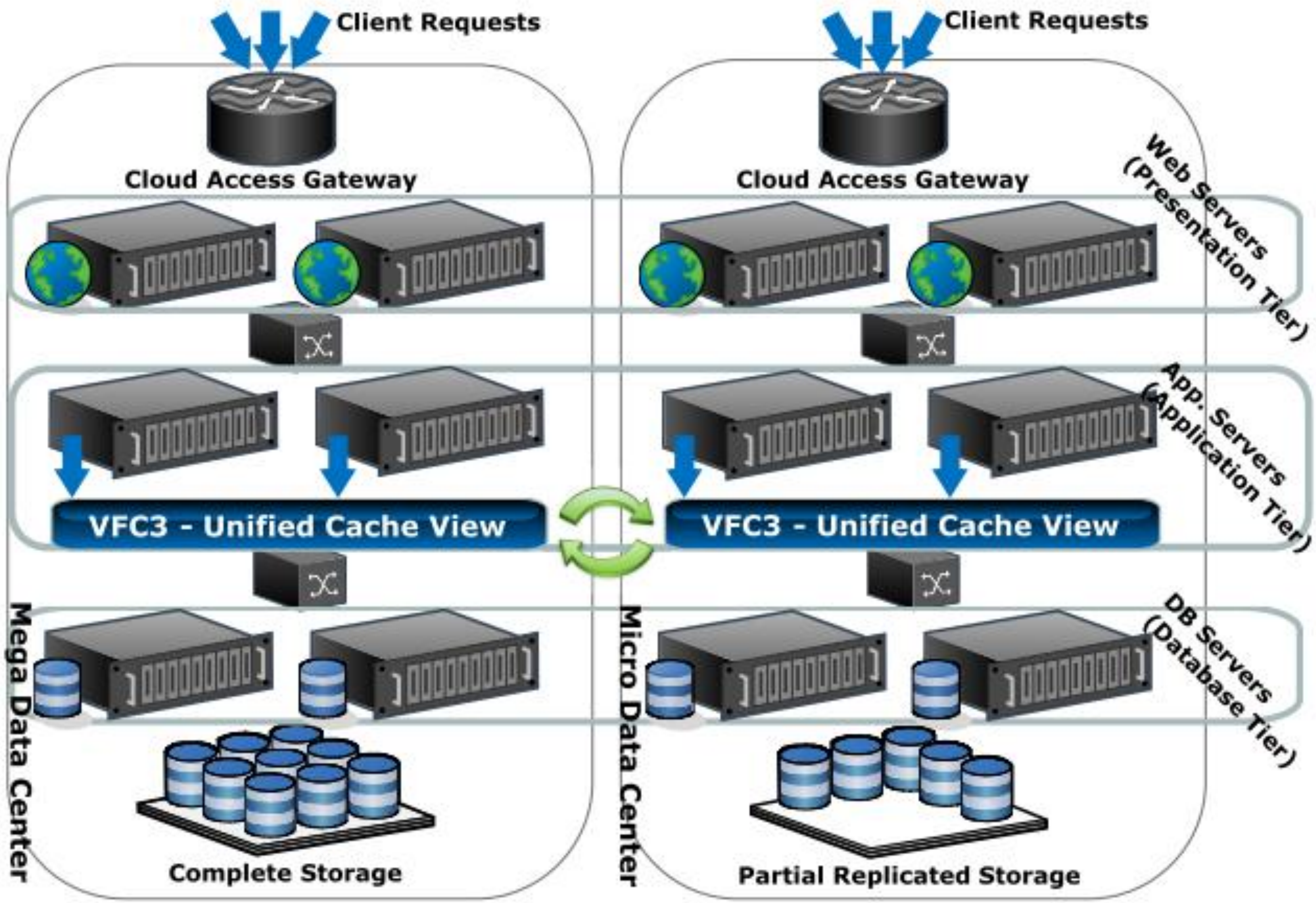
- Target is cloud-like distributed tabular data store, HBase:
  - Sparse, multi-dimensional sorted map, indexed by row, column (includes family and qualifier), and timestamp;
  - Maps to an uninterpreted array of bytes
  - Put/Get interface
  - Scale into the petabytes with read/write speed remaining constant
- Data is semantically divided through object containers
  - (table/row/column)

Geo-Distributed Cloud Scenario



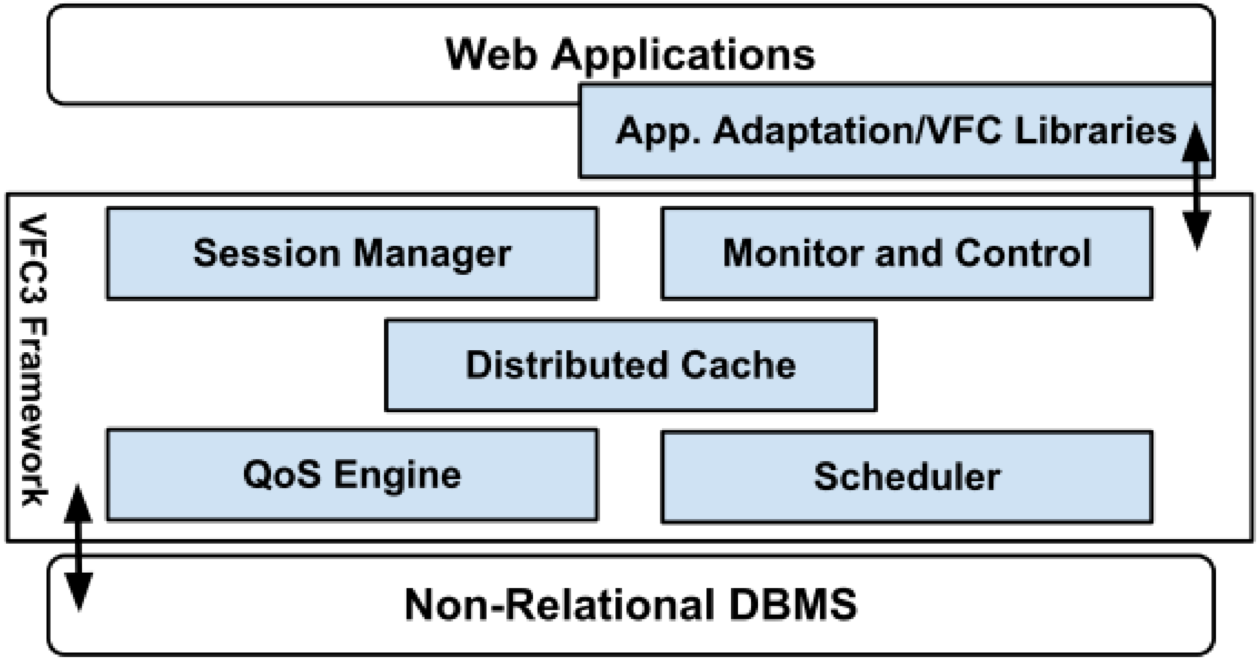
- Scenario: Each micro-DC has different requirements over different subsets of data (multi-homing)
- Requiring selective replication with prioritization

Data-Center Detailed View



- Mediates application read/write access to the data store.

VFC3 Middleware Architecture



VFC3: In-memory Storage and Caching

- In-memory DB cache at the application level for:
  - i) Keep tracking of items waiting for being fully replicated
  - ii) Temporarily store frequently accessed values (or within the same locality group)
- Provide transactional consistency with the underlying DB
  - Data is automatically maintained/invalidated
  - Relieving developers from a very error prone task
- Distributed while still giving a logical view of a single cache
  - Horizontal partitioning
  - Hash of row keys determines the location of data
- Eviction policies (default: LRU) and size configurable

QoS-Consistency Model

- Object containers are associated with divergence bounds
- A divergence bound, κ, defines 3 orthogonal dimensions:

Time (θ) Maximum time a replica can be without being refreshed with its latest value (eg, seconds)

Sequence (σ) Maximum number of updates without refreshing replicas

Value (ν) Relative difference between replica contents or against a constant (eg, top value)

- Example: Level of pollution in a city.
  - 100 sensors gauging the concentration of Pollution (eg, CO_2)
  - Possible bound: $\kappa = [3600\theta, 10\sigma, 15\nu]$
  - Small changes do not impact the overall level of pollution

QoS: Enforcement of Divergence Bounds

- Upon write operation:
  - Identify affected object containers
  - For each container identified:
    - Increment vector sequence
    - Calculate vector value by dividing the average of the variation of all changed data by that same average but when last replication occurred
  - Check if any divergence bound is crossed
    - If so, values of that container are replicated
- Scheduler uses checks every sec if there is any object for being sync with a timestamp about to expire

QoS: Enforcement of Divergence Bounds

Concurrent Updates

- Last-writer-wins rule to make DCs converge on the same values
- Rotating leases for stronger agreement

Dynamic Adjustment of Consistency Guarantees

- QoS constraints can be specified as intervals (eg, $\kappa = [120 - 300\theta, 10 - 30\sigma, 5 - 15\nu]$)
- Many updates on different nodes over the same replica cause conflicting accesses => strengthen consistency
- Few updates concentrated in few nodes (less conflicts) => weaken consistency
- Many reads on data that is mainly written in other nodes => strengthen consistency

Evaluation Scenario

- HBase standalone instances at two fairly distant locations
- Evaluate gains of VFC3 framework
  - enforcing QoS-Consistency
  - against the regular HBase replication (pessimistic)

- Yahoo's YCSB Benchmark provides workloads for a key set of scenarios

Evaluation: Latency and throughput (Updates)

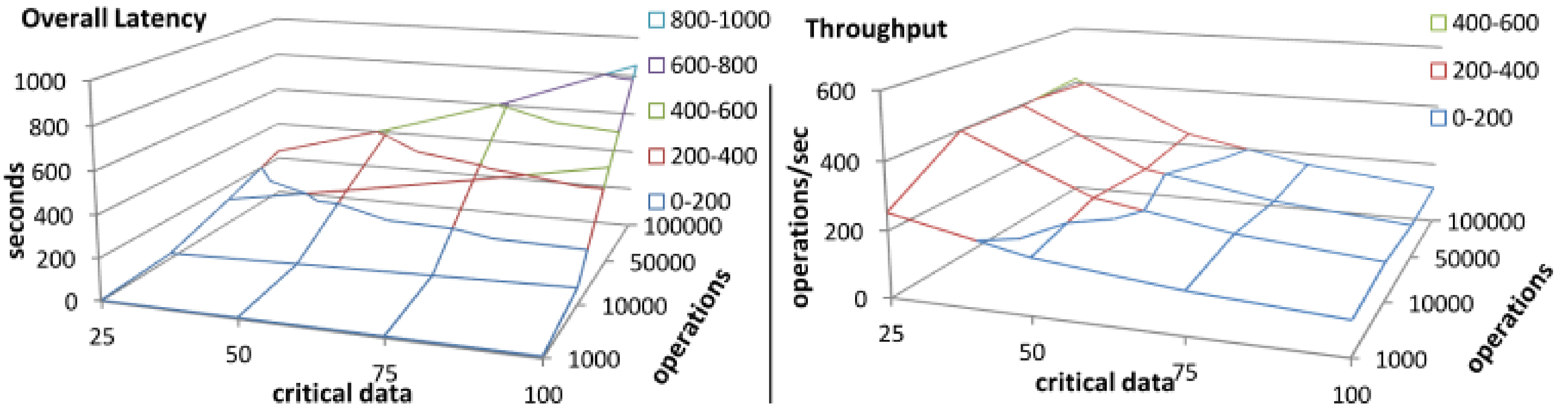


Figure : 95/5% updates/reads

- Latency gains almost linear (straight lines)
  - Latency of single ops was nearly constant
  - Especially for write ops
- Gains obtained every time percentage of critical data decreased
- Average throughput about same for each class of critical data
  - Sustained, irrespective of the number of operations

Evaluation: Latency and throughput (Reads)

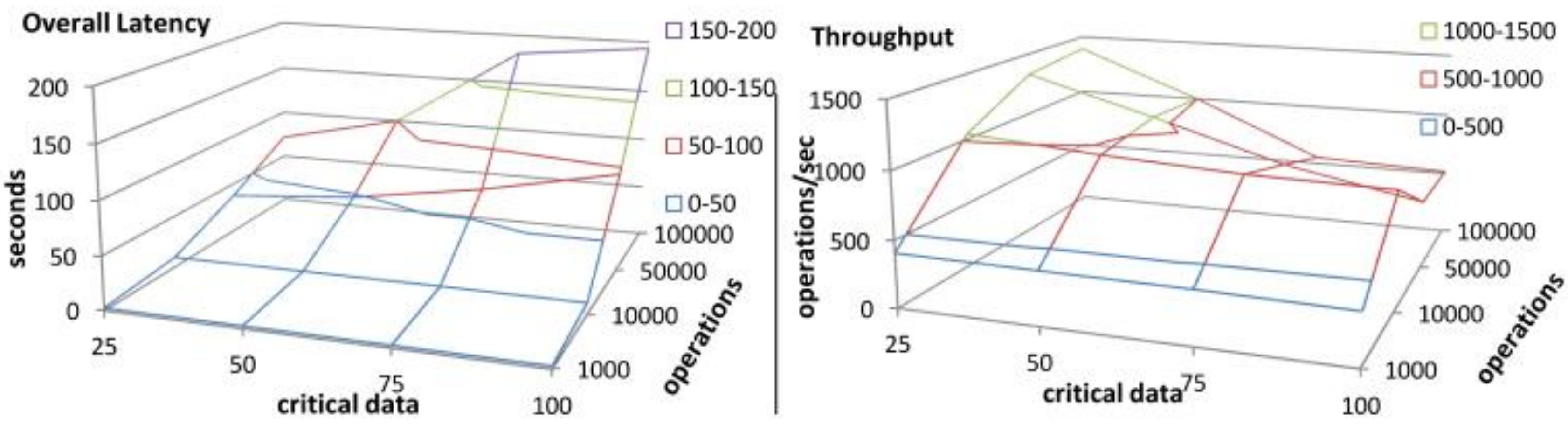


Figure : 5/95% updates/reads

- The level of critical data is almost irrelevant in heavy reads' workloads
- The gains here were mostly supported by our cache
- Throughput gains are only significant for 50K and 100K operations

Evaluation: Latency and throughput (Mix)

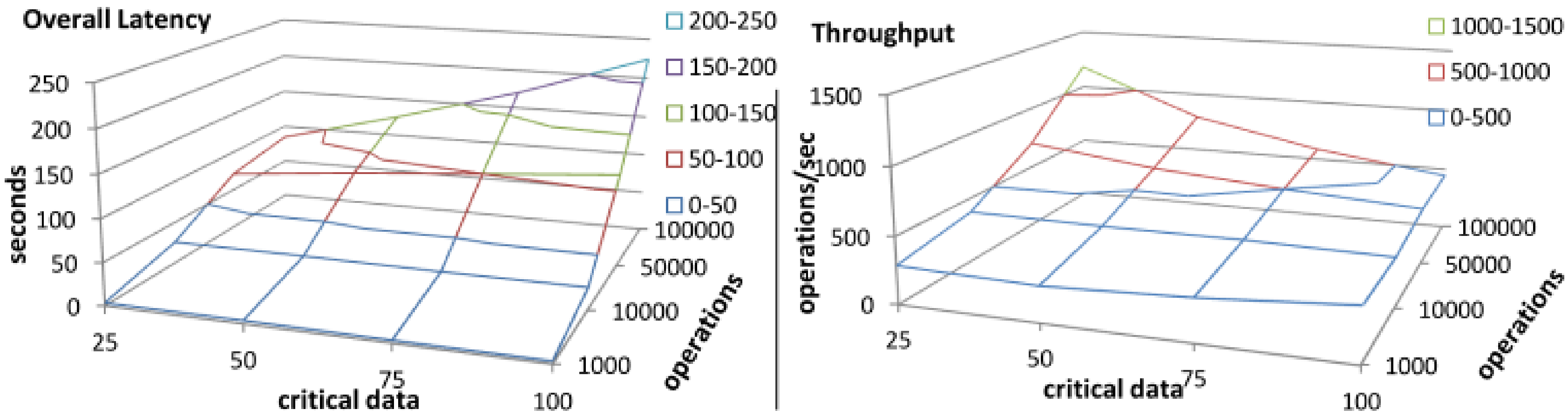
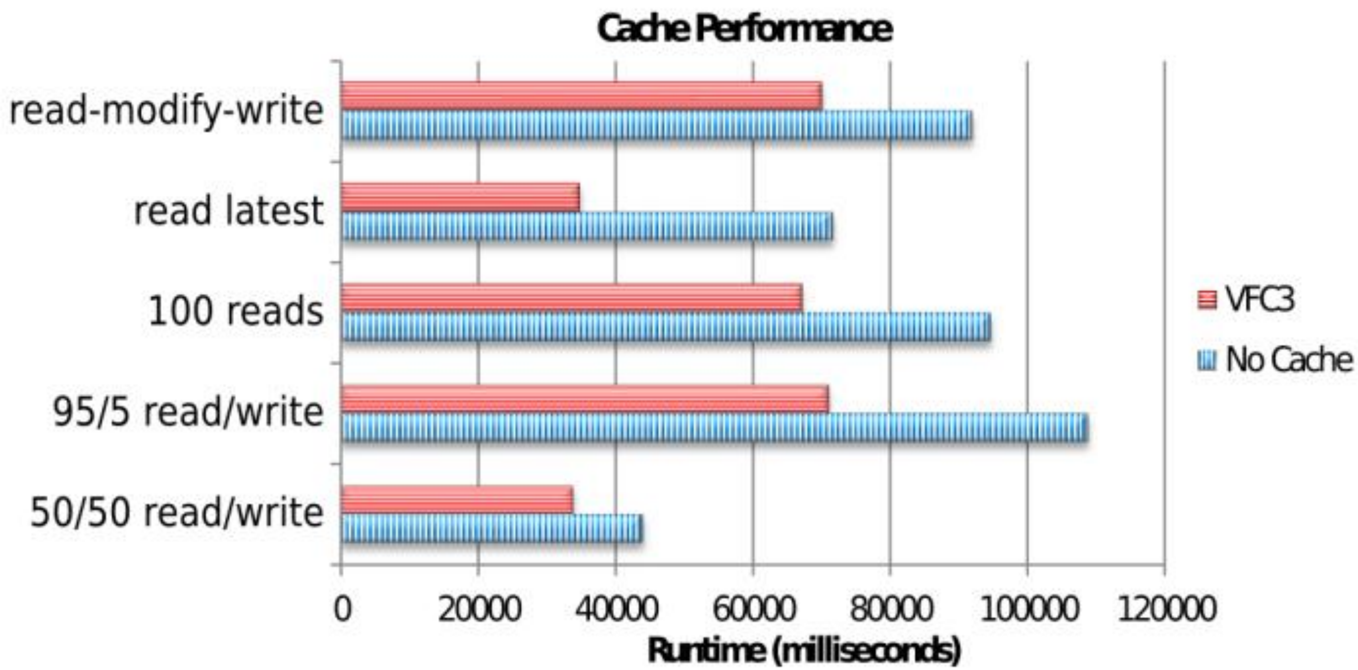


Figure : 50/50% updates/reads

- Latency gains almost linear as the number of ops and the amount of critical data increases
- Throughput gains are more accentuated for higher number of ops

Evaluation - Cache Performance

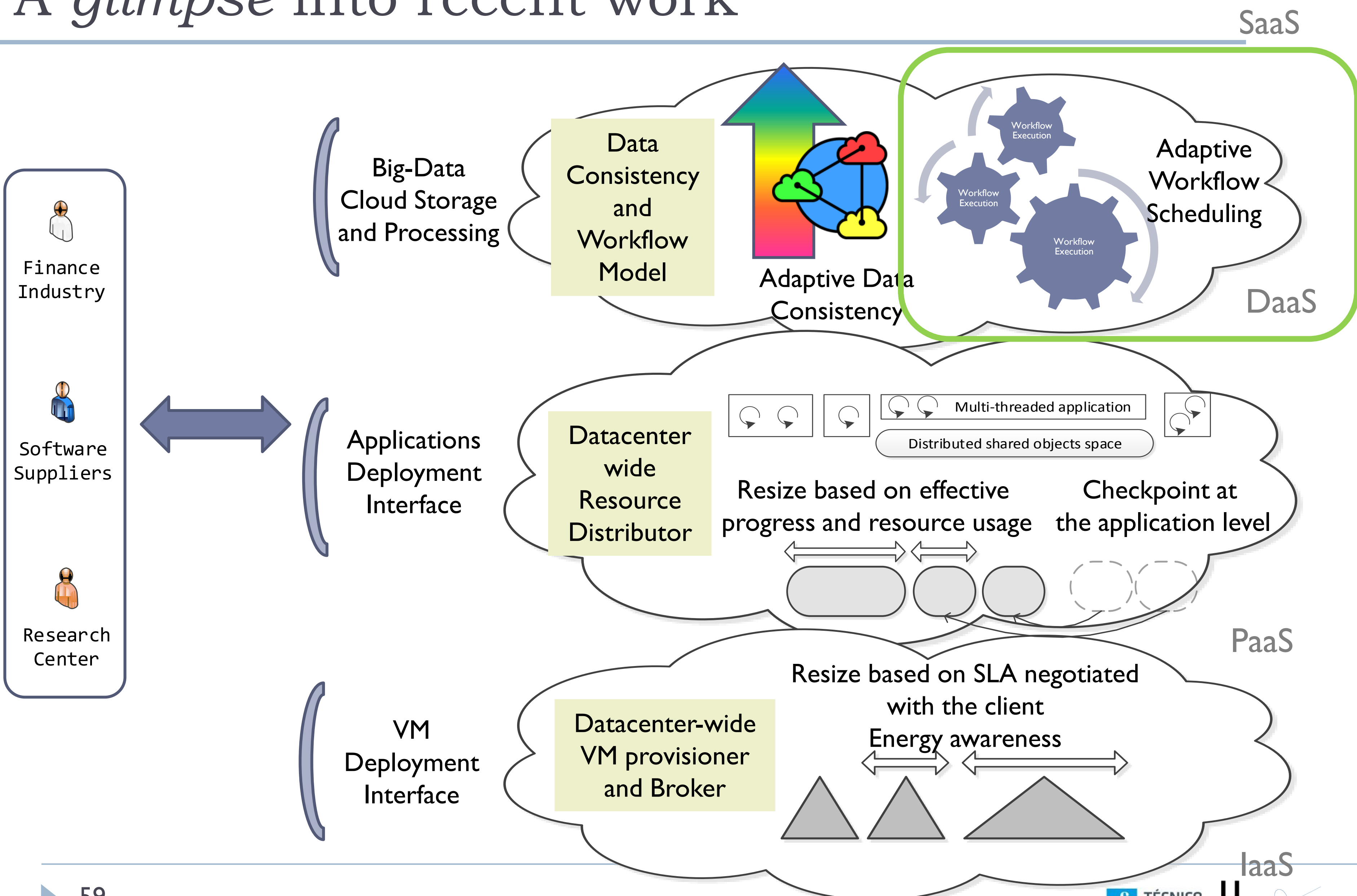


- The average hit rate was 51%
- Workload "read-latest" obtained 77% (LRU eviction policy)
- So, it can definitely improve performance for a set of typical scenarios and save expensive trips to the database

Talk Outline

- Introduction
- Adaptability in virtual machines
- PaaS
  - Models, Mechanisms, Evaluation
- IaaS
  - Models, Mechanisms, Evaluation
- Energy and Community Clouds
  - Models, Mechanisms, Evaluation
- DaaS – Data Processing
  - Models
  - Mechanisms
  - Evaluation

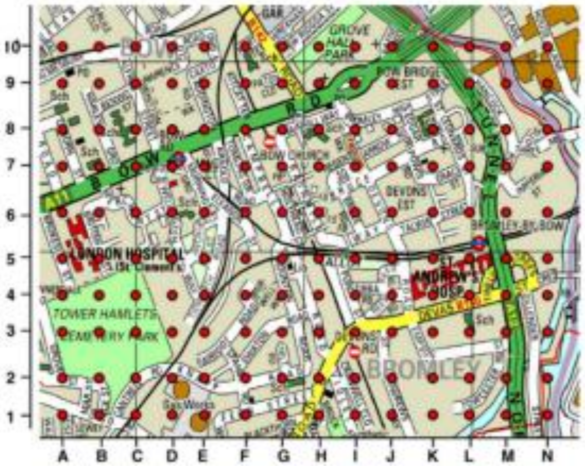
A *glimpse* into recent work

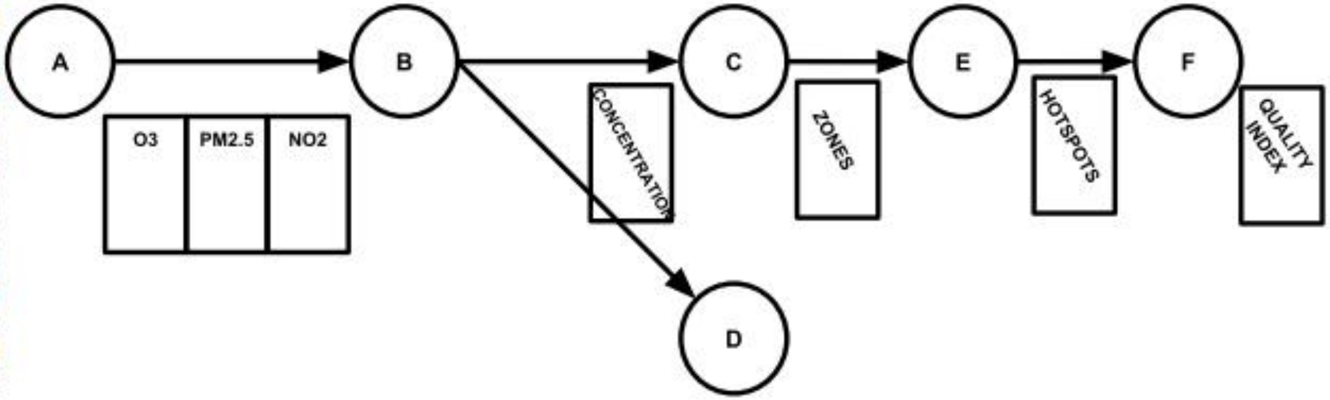


Big-Data Processing Motivation

- Growing need to analyze, process and store data from application with real-time demands
- Traditional WMS enforce strict temporal synchronization
- In continuous processing, resources are wasted
  - small impact that input data might have on the final output
- No data- and application-aware reasoning is performed
  - evaluate impact of new executions on the final dataflow output
- Fail to deliver high resource efficiency for long-lasting applications while keeping costs low

Use Case - Assessing Air Quality and Pollution





- AQHI – Air Quality Health Index (Canada)
- Executed every fixed interval with a wave of data fed from sensors
- Most times, sequential waves would not change the workflow result
  - pollution stable during most part of the day and night,
  - changing more significantly at rush hours
- Resources are wasted with unnecessary workflow executions

Data-Aware Data Processing Model

- Novel dataflow model with *Quality-of-Data (QoD)*
  - for continuous and incremental processing
- WMS framework to enforce the QoD model
- Trade-off results accuracy with resource savings
- Achieve:
  - resource efficiency
  - controlled performance
  - task prioritization

[11] SE, JS, LV @ Springer JISA 2013
[12] SE, LV @ Oxford Press Computer 2014

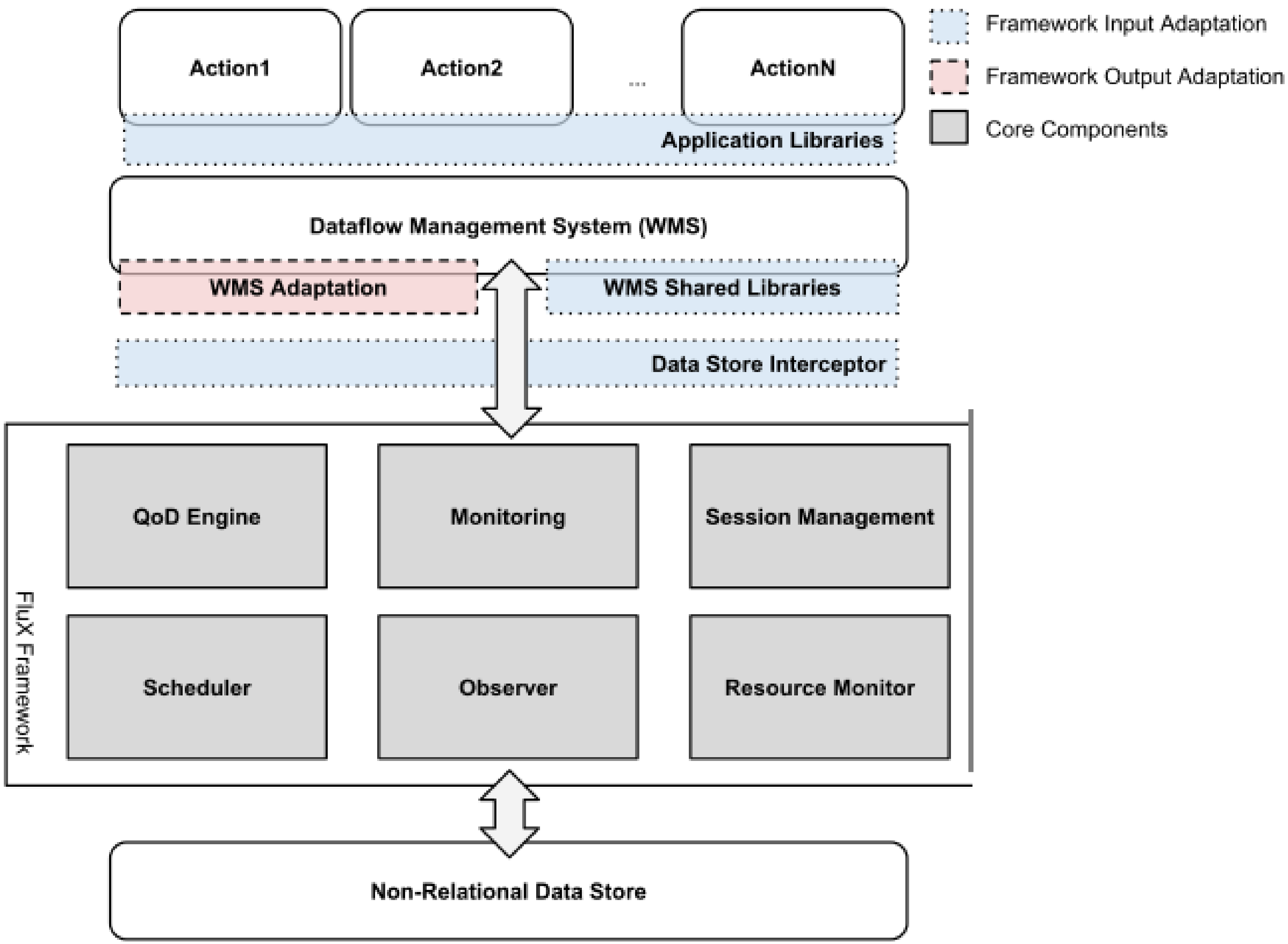
Model: *Quality-of-Data*

- Targeting data-intensive dataflows
- Processing steps communicate data via noSQL database
- Data container can be a column or set of columns
- Steps are triggered when predecessor steps:
  - make sufficient impactful changes in containers,
  - according to QoD bound k

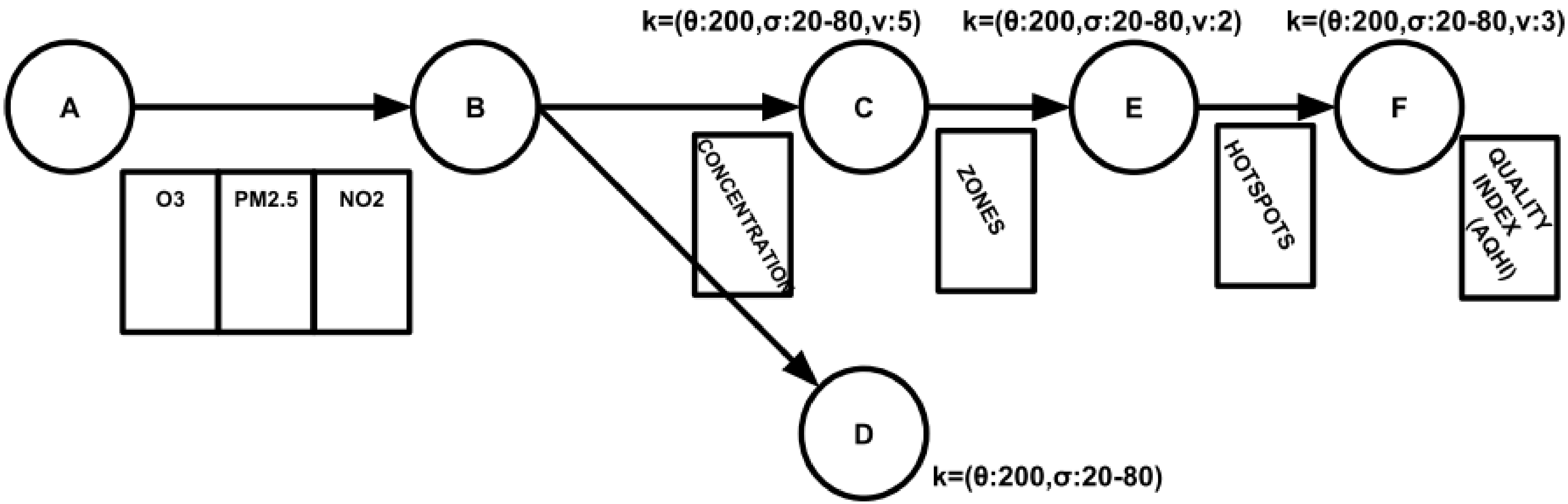
$k = (\theta, \sigma, \nu)$

Time (θ)	Maximum time a task can be on hold (eg, in seconds)
Sequence (σ)	Maximum number of updates
Value (ν)	Maximum relative divergence (eg, in percentage)

FluX: Hadoop Workflow Framework

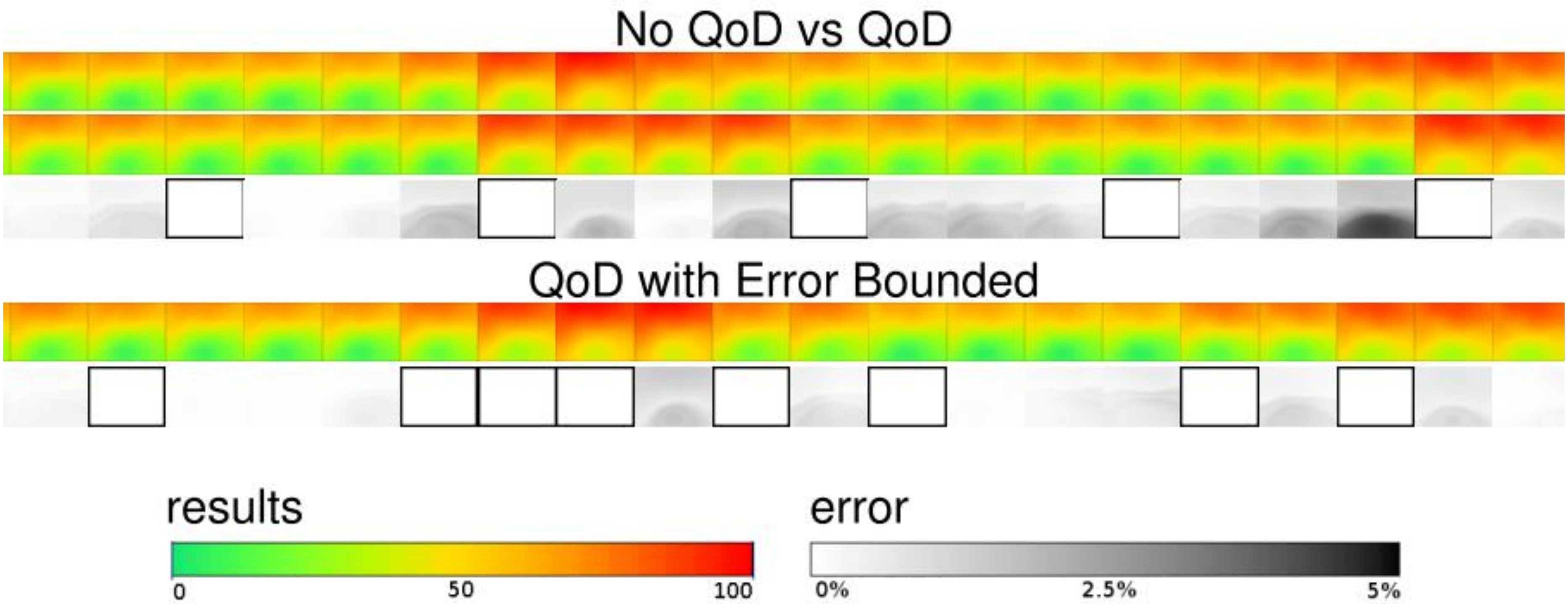


FluX Evaluation Scenario

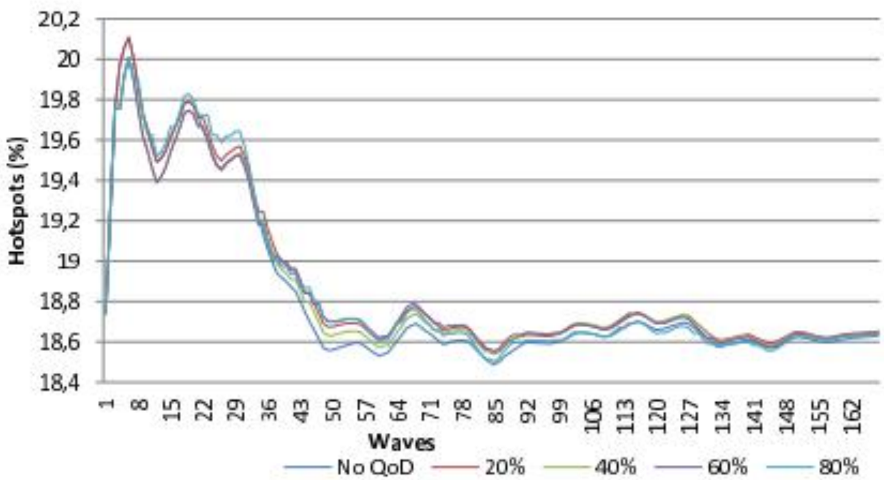


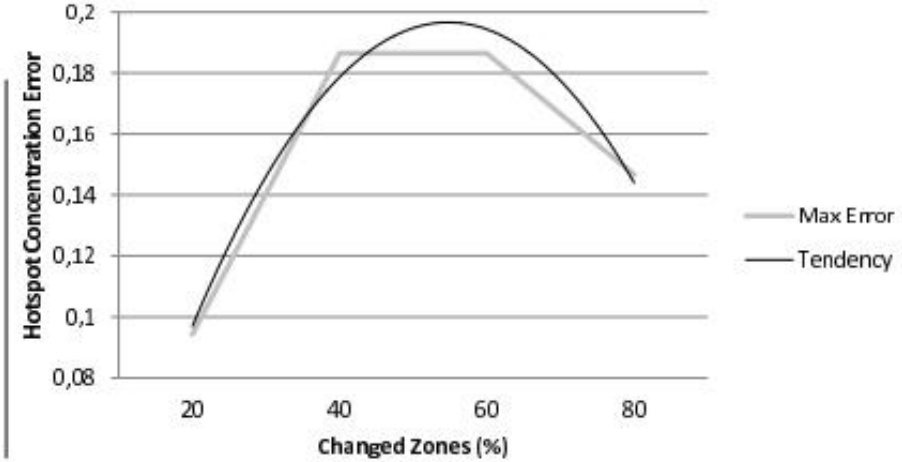
- Matrix with 2500 (50 × 50) to 40000 (200 × 200) detectors
- Each hour sensor data is injected in the dataflow (a wave)
- 168 waves (24 hours per 7 days)

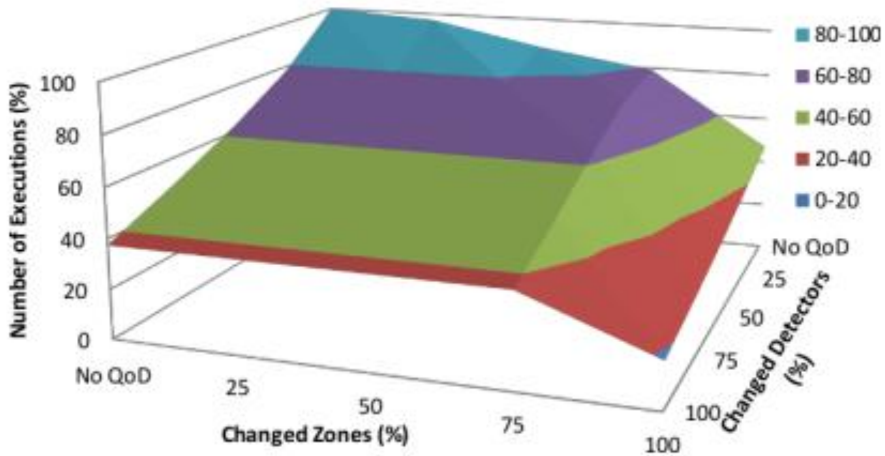
Evaluation: Pollution Maps Step



Evaluation: Hotspots Step

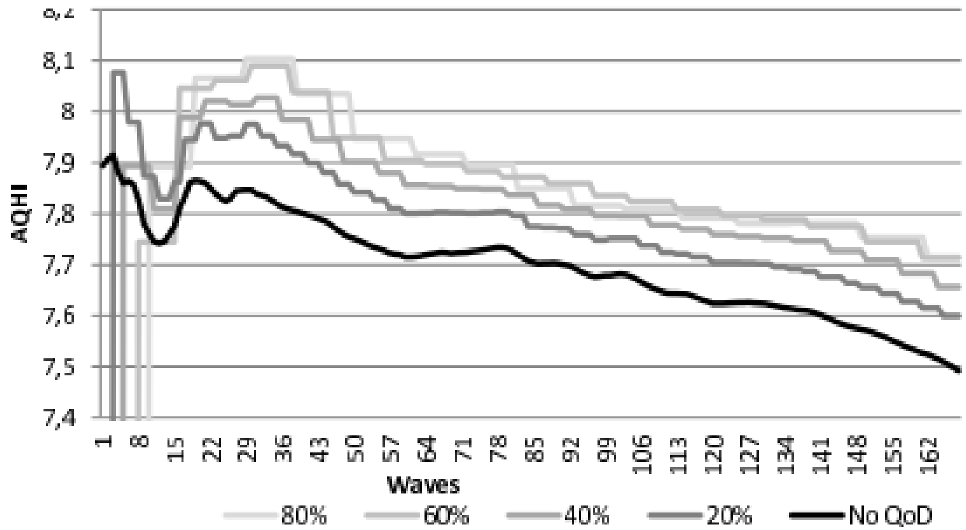


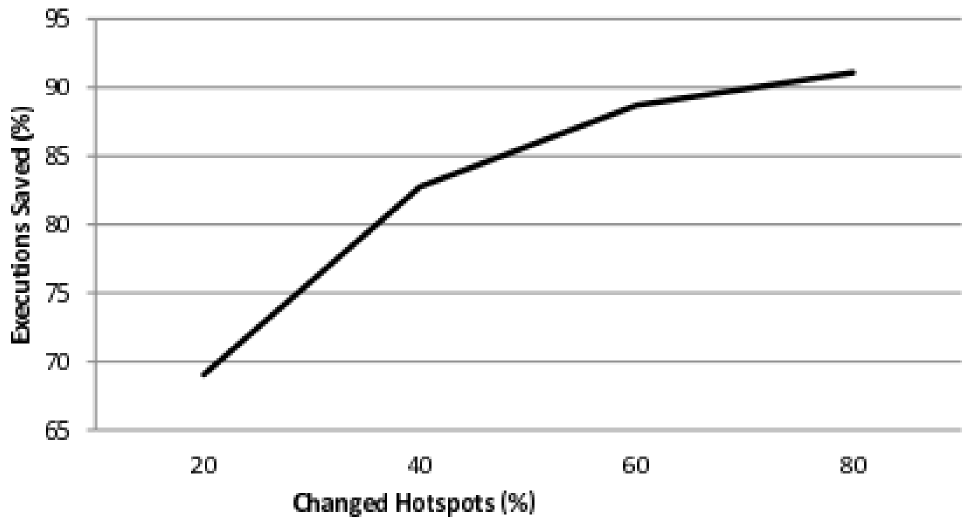




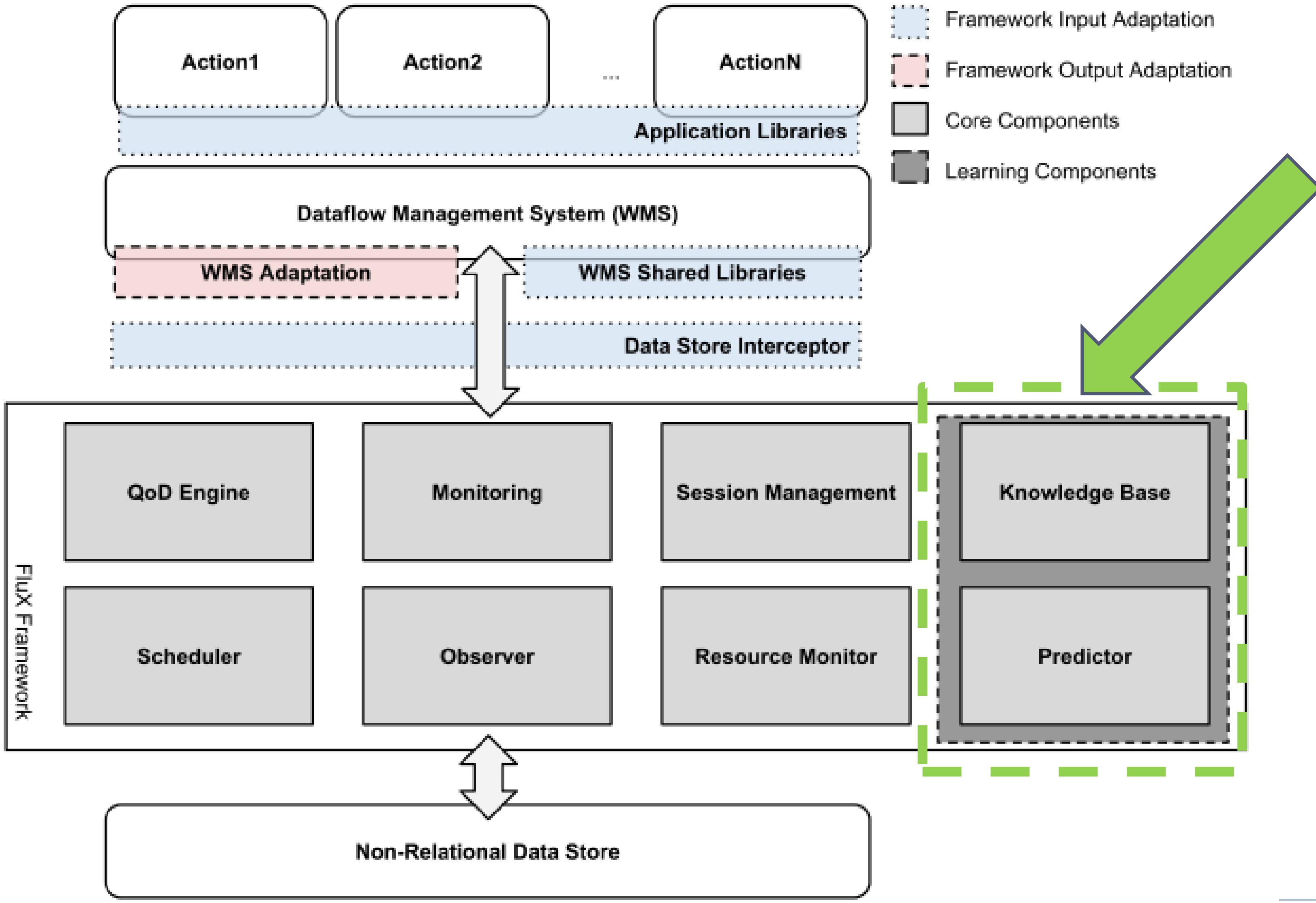
TÉCNICO LISBOA

Evaluation: Air Quality (AQHI) Step





Fluxy: **Learning** DataFlow **Patterns**



Fluxy: **Learning DataFlow Patterns**

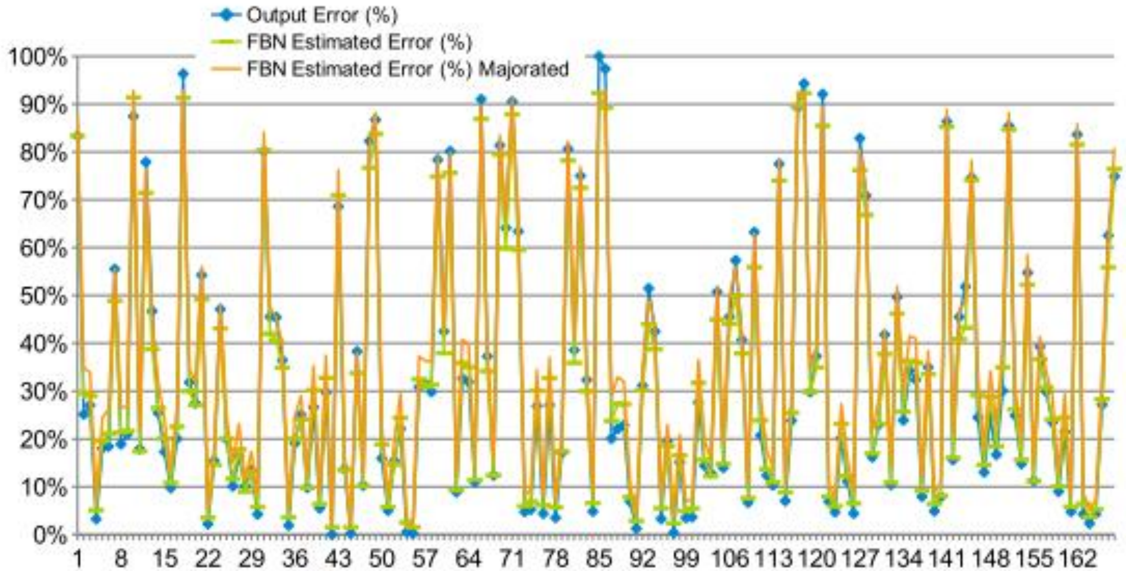
- **Error Assessment and Bounding**
- Postponing the execution of processing steps **introduces divergence** (error) in the values, as opposed to the synchronous model
- Error $\varepsilon = \dfrac{\sum_{i=1}^{m} |x_i - x'_i| \times m}{\sum_{i=1}^{n} x'_i \times n}$
- Bound error with Machine Learning to **guarantee correctness** of the dataflow with minimum accuracy

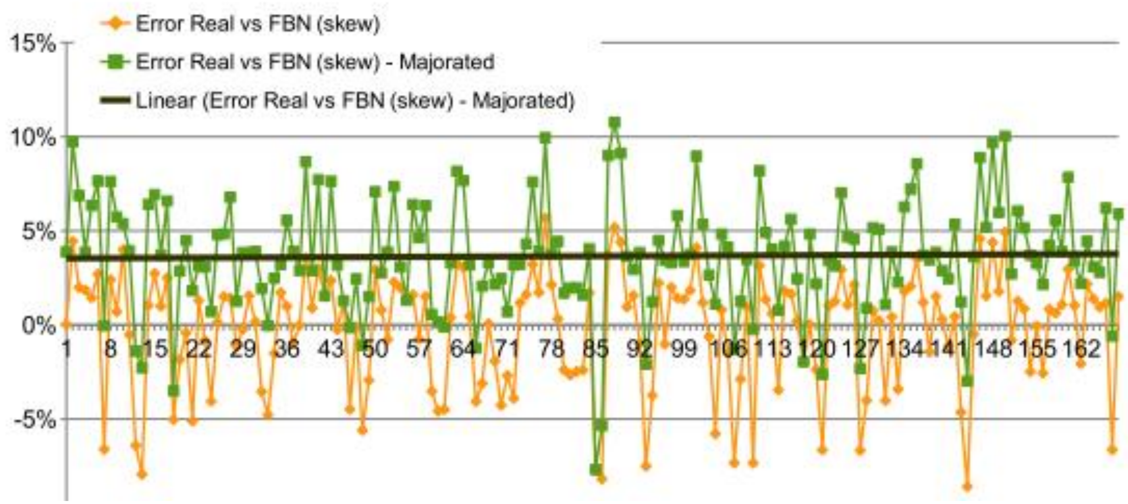
Fluxy: **Learning** DataFlow **Patterns**

- Learn dataflow patterns: correlation between input variation (ι) and corresponding output error (ϵ)
  - $\iota = \sum_{i=1}^{m} |x_i - x'_i|$
- Fuzzy Boolean Nets
  - competitive performance
  - cope very well with sparse and imbalanced datasets
- Define maximum tolerated error (max_ϵ)

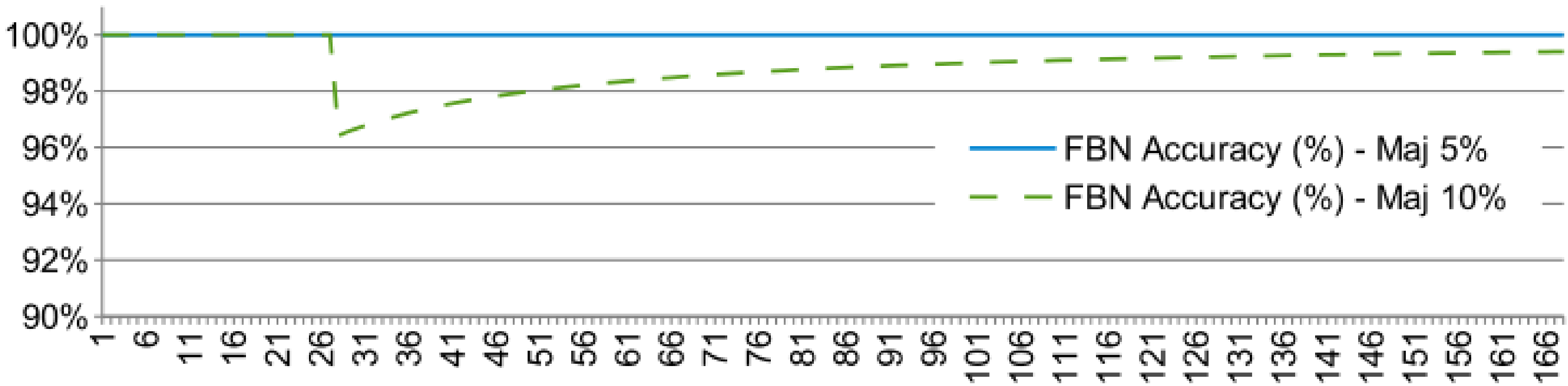
[13] SE, JPC, JS, LV @ Elsevier JPDC 2015

Evaluation: Output Error and Estimation



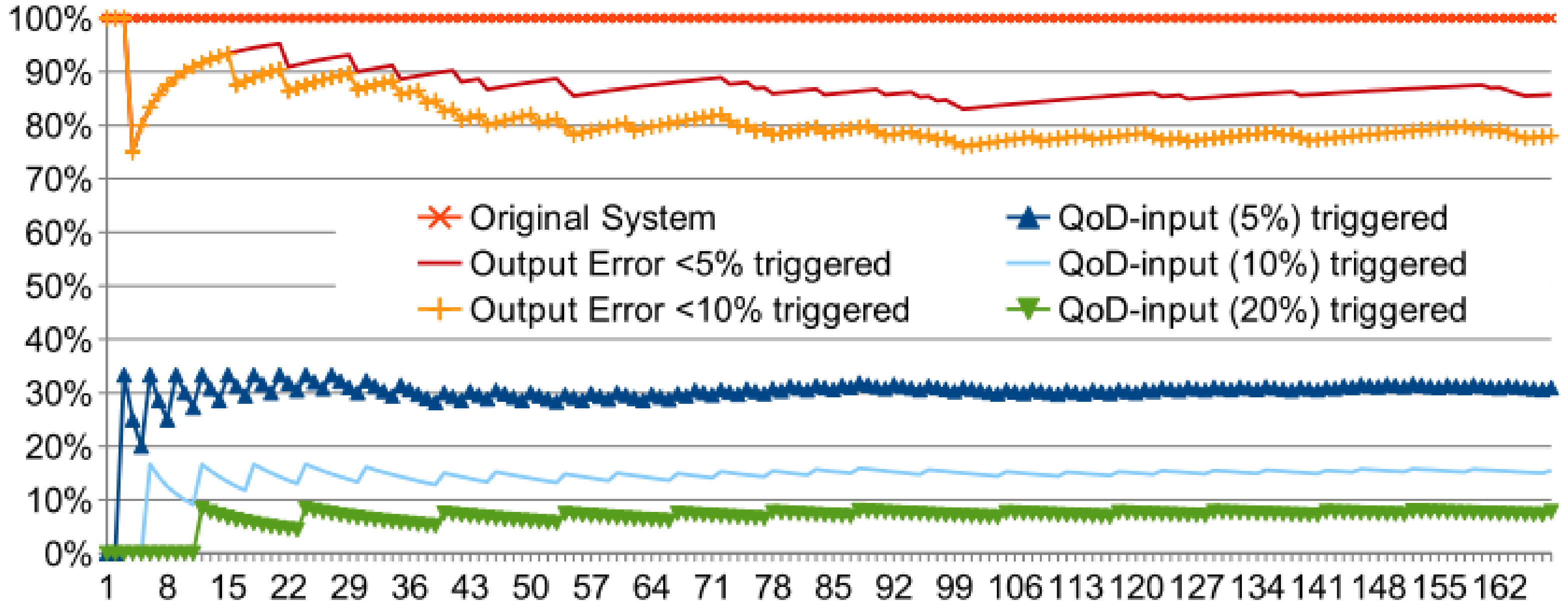


Evaluation: Prediction Accuracy



- Fully correct when $max_{\epsilon} = 5\%$
- Above 96% when $max_{\epsilon} = 10\%$
- Good confidence for decision makers (above 95%)

Evaluation: Resource Usage



- Without error bounding, resource savings are huge
- Ensuring error bounding of 5/10%, we can save up to 18/25% executions
- Significant in a shared computing environment with multiple dataflows

Take Away on Big-Data Processing

- Novel workflow model for continuous data-intensive processing with QoD
- Evaluation based on a realistic data-intensive dataflow
- With Fuzzy Boolean Nets we are able to bound output error and keep computations "correct"
- High resource efficiency and controlled performance
- Future Work:
  - DSL to capture the impact of updates in data containers
  - Develop cloud service with pricing model for different SLAs (QoD)

Future Work

- Stream and Graph processing
  - Adapt consistency and workflow processing models to streams
  - Online/Incremental Graph Processing
  - Provide Timeliness, Accuracy and Resource guarantees

- Data processing at the edge of the networks
  - Scale out distributed/micro-DCs towards Internet-of-Things (IoT)

- Networking issues
  - Big-Data workloads aware SDN (software-defined networking)
  - CPU/VM and memory resources guarantees are not enough to ensure performance → network must be 1st class resource

2nd DataStorm Big Data Summer School

Big Data Infrastructures

Economics of Resources, Energy, Data, and Applications

** Thank you for your attention **
Questions ?

Acknowledgements: PhD and MSc students
Sérgio Esteves, Leila Sharifi, and José Simão (graduated)

References

[1] José Simão and Luís Veiga, **Partial Utility-driven Scheduling for Flexible SLA and Pricing Arbitration in Cloud**, article in *IEEE Transactions on Cloud Computing*, Nov. 2014, IEEE, **[DOI Article link][bibTex]**

[2] José Simão and Luís Veiga, **Flexible SLAs in the Cloud with Partial Utility-driven Scheduling (Best-Paper Award Runner-up)**, presented at *IEEE 5th International Conference on Cloud Computing Technology and Science (CloudCom 2013)*, Dec. 2013, IEEE, **[bibTex]**

[3] Leila Sharifi and Navaneeth Rameshan and Felix Freitag and Luís Veiga, **Energy Efficiency Dilemma: P2P-cloud vs. mega-datacenter (Best-Paper Candidate)**, presented at *IEEE 6th International Conference on Cloud Computing Technology and Science (CloudCom 2014)*, Dec. 2014, IEEE, **[bibTex]**

[4] José Simão and João Nuno Pessanha Alcoforado Sampaio de Lemos and Luís Veiga, **A2-VM: A Cooperative Java VM with Support for Resource-Awareness and Cluster-Wide Thread Scheduling**, presented at *19th International Conference on COOPERATIVE INFORMATION SYSTEMS (CoopIS 2011)*, Sep. 2011, LNCS, Springer, **[bibTex]**

[5] José Simão and Luís Veiga, **QoE-JVM: An Adaptive and Resource-Aware Java Runtime for Cloud Computing**, presented at *2nd International Symposium on Secure Virtual Infrastructures (DOA-SVI 2012), OTM Conferences 2012*, Sep. 2012, Springer, LNCS, **[bibTex]**

[6] José Simão and Luís Veiga, **Adaptability Driven by Quality Of Execution in High Level Virtual Machines for Shared Environments**, article in *International Journal of Computer Systems Science and Engineering* pp. 59-72,Nov. 2013, CRL, **[Article][bibTex]**

[7] José Simão and Luís Veiga, **A Progress and Profile-driven Cloud-VM for Improved Resource-Efficiency and Fairness in e-Science Environments**, presented at *28th ACM Symposium On Applied Computing (SAC 2013)*, Mar. 2013, ACM, **[bibTex]**

[8] José Simão and Tiago Garrochinho and Luís Veiga, **A Checkpointing-enabled and Resource-Aware Java VM for Efficient and Robust e-Science Applications in Grid Environments**, article in *Concurrency and Computation: Practice and Experience* pp. 1421-1442, Sep. 2012, Wiley, **[DOI Article link][bibTex]**

[9] Sérgio Esteves and João Nuno de Oliveira e Silva and Luís Veiga, **Quality-of-Service for Consistency of Data Geo-Replication in Cloud Computing**, presented at *International European Conference on Parallel and Distributed Computing (Euro-Par 2012)*, Aug. 2012, Springer, LNCS, **[bibTex]**

[10] Álvaro Garcia Recuero and Sérgio Esteves and Luís Veiga, **Quality-of-Data for Consistency Levels in Geo-replicated Cloud Data Stores**, presented at *IEEE CloudCom 2013*, Dec. 2013, IEEE, **[bibTex]**

[11] Sérgio Esteves and João Nuno de Oliveira e Silva and Luís Veiga, **FluX: a quality-driven dataflow model for data intensive computing**, article in *Journal of Internet Services and Applications (JISA)* pp. 1-23, Apr. 2013,Springer, **[Article][DOI Article link][bibTex]**

[12] Sérgio Esteves and Luís Veiga, **WaaS: Workflow-as-a-Service for the Cloud with Scheduling of Continuous and Data-intensive Workflows**, article in *Computer Journal, ISSN 0010-4620*, Dec. 2014, Oxford University Press,**[DOI Article link][Supplementary Material][bibTex]**

[13] Sérgio Esteves and João Nuno de Oliveira e Silva and Joao P. Carvalho and Luís Veiga, **Incremental Dataflow Execution, Resource Efficiency and Probabilistic Guarantees with Fuzzy Boolean Nets**, article in *Journal of Parallel and Distributed Computing (JPDC)*, to appear on 2014, Elsevier, **[bibTex]**