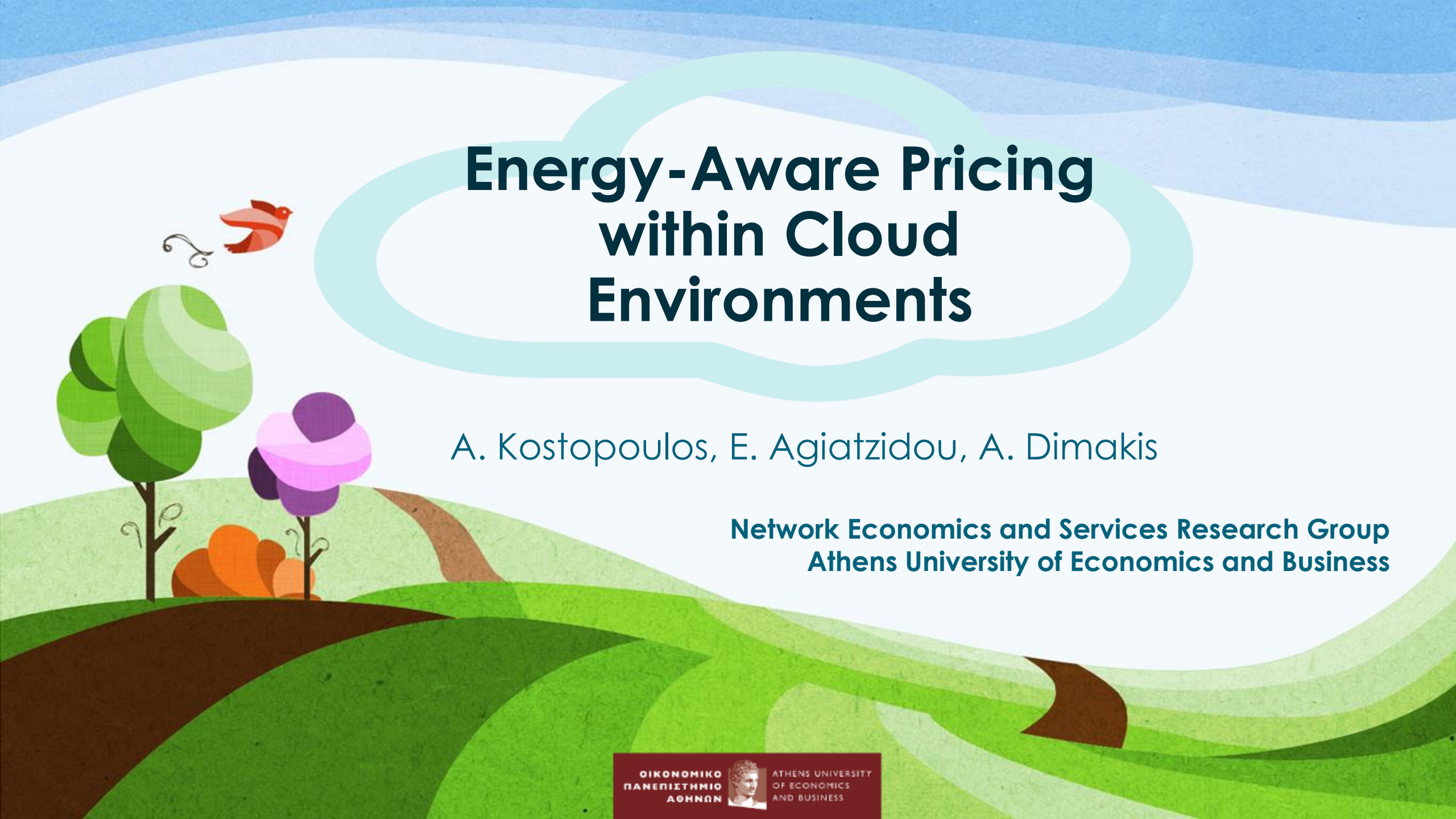
Energy-Aware Pricing within Cloud Environments



A. Kostopoulos, E. Agiatzidou, A. Dimakis

Network Economics and Services Research Group
Athens University of Economics and Business

ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

Outline

- Motivation
- ASCETiC Approach
- Overall Architecture
- Pricing Modeller Components
- Adopted pricing schemes in the market
- Proposed energy-aware pricing schemes
- Service plans
- Conclusions & Future work

Motivation

- *How could energy efficiency be supported across all cloud layers?*
- *How software systems actually use cloud resources for optimization purposes?*
- *How energy-awareness could be incorporated within pricing schemes to provide incentives?*

➢ **Incorporation of an approach that combines energy-awareness related to cloud environments.**

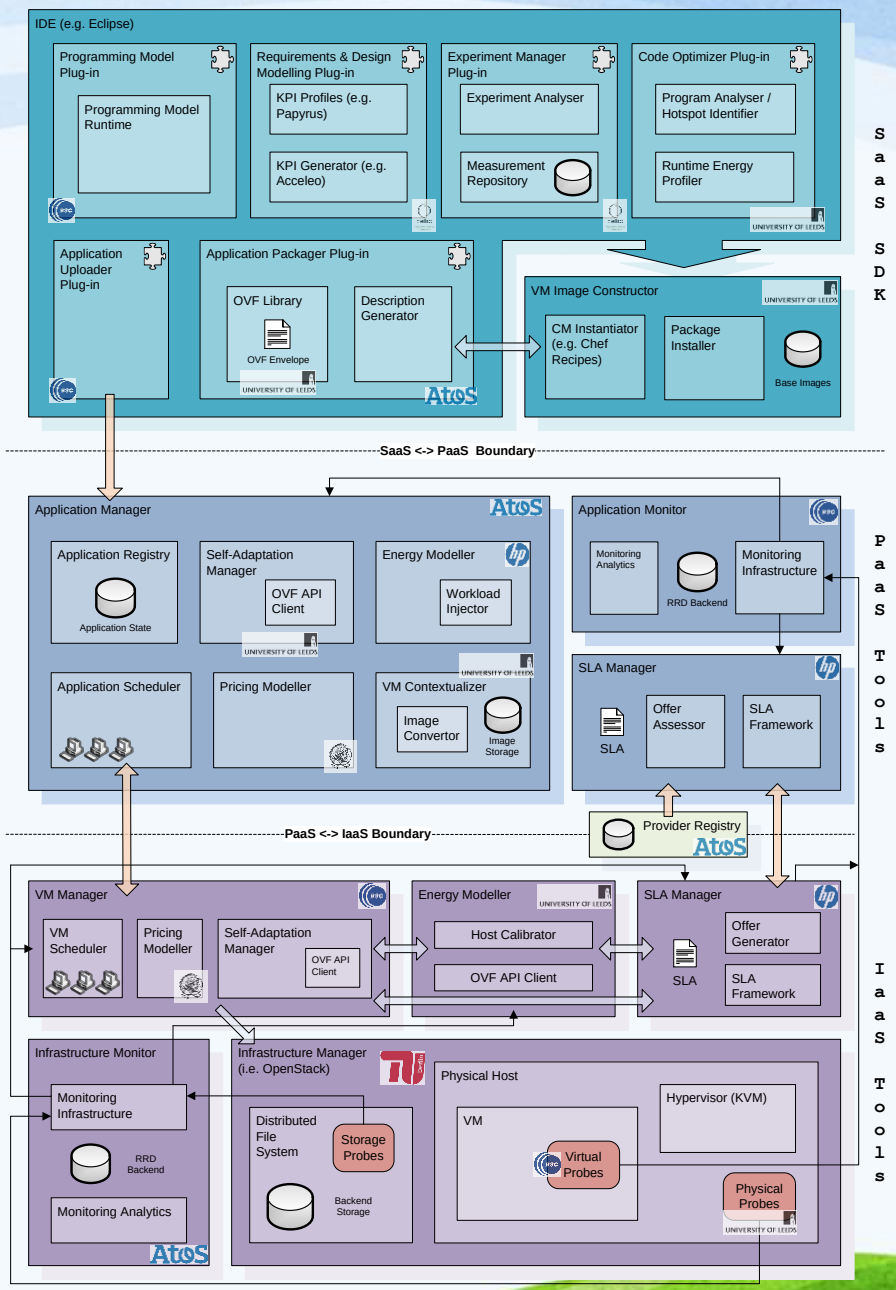
ASCETiC Approach

- Adapting Service lifeCycle towards EfficienT Clouds (**ASCETiC**)
  - ✓ provide novel methods and tools to support software developers
  - ✓ aiming to *optimize energy efficiency resulting from designing, developing, deploying and running software* at the different cloud layers,
  - ✓ while maintaining other quality aspects of software to meet the agreed levels.

- **Identification of the missing functionalities to support energy efficiency across all cloud layers**, and
- **Definition and integration of explicit measures of energy requirements into the design and development process for software** which can be executed on a cloud platform.

Overall Architecture

- **SaaS**: facilitates the modelling, design and construction of a cloud application.
- **PaaS**: provides middleware functionality for a cloud application and facilitates the deployment and operation of the application as a whole.
- **IaaS**: the admission, allocation and management of virtual resources are performed.



Pricing Modeller Components

- Existence of a component dedicated to support the financial operations of the provider:
  - *cost*
  - *charging*
- *PaaS Pricing Modeler*:
  - estimate the price per hour and charges of an application before deployment,
  - calculate its charges after its operation.
- *Iaas Pricing Modeler:*
  - estimate the price per hour and the charges of a VM before deployment
  - calculate its charges after its operation

Pricing Modeller Components (cont.)

- Enable the use of appropriate cost and price functions for the provider of the corresponding cloud layer.
- *Cost function:* the costs that a provider faces during its operation in order to offer his services.
- *Pricing scheme:* dictates the way of making revenues from the customers.
- IaaS-based energy-aware pricing schemes
- A PaaS provider
  - neither pays the electricity bill, nor owns any physical machines
  - broker providing cloud services from different IaaS providers
  - playing also the role of an IaaS / bundling

Adopted pricing schemes in the market

- **Pay-as-you-go:** the customer pays only for the resources he uses
- **Periodic payment:** monthly, semester, yearly subscriptions, etc., for the use of specific resources
- **On-demand instances**: reservation of the resources before their use for a specific amount of time. The customers can make a low, one-time payment for each instance they reserve to receive a significant discount.
- **Spot instances:** customer buys the unused Amazon EC2 capacity and runs it until the price of the instances bought becomes higher than his bid. Prices change periodically based on supply / demand.

Adopted pricing schemes in the market (cont.)

Provider	Pricing Scheme	Provided cloud services
Amazon	On-Demand Instances, Reserved Instances, Spot Instances	Standard, Second Generation, Micro, High Memory, High CPU, Cluster Compute, High Memory Cluster, High I/O, High Storage, and according to the operating system
RackSpace	Pay-as-you-go	Cloud Servers: Size, Disk, vCPUs, Public / Internal Network, operating system
GoGrid	hourly, monthly, semiannual, and annual cloud	RAM, Cores, Storage
Microsoft	Pay-as-you-go, semester, year	CPU, RAM
Terremark	Pay-as-you-go	Memory, VPU
AT&T	Pay-as-you-go	Capacity, memory and system storage
Google	Pay-as-you-go	Virtual Cores, Memory, Local disk
OpScource	Pay-as-you-go, monthly	Size, Disk, vCPUs, Public/Internal Network, operating system
SoftLayer	Pay-as-you-go, monthly	Core, RAM, storage, operating system
HP	Pay-as-you-go	Core, RAM, storage, operating system,
Engine Yard	Pay-as-you-go	Core, RAM, storage
Acquia	Pay-as-you-go	Core, RAM, storage

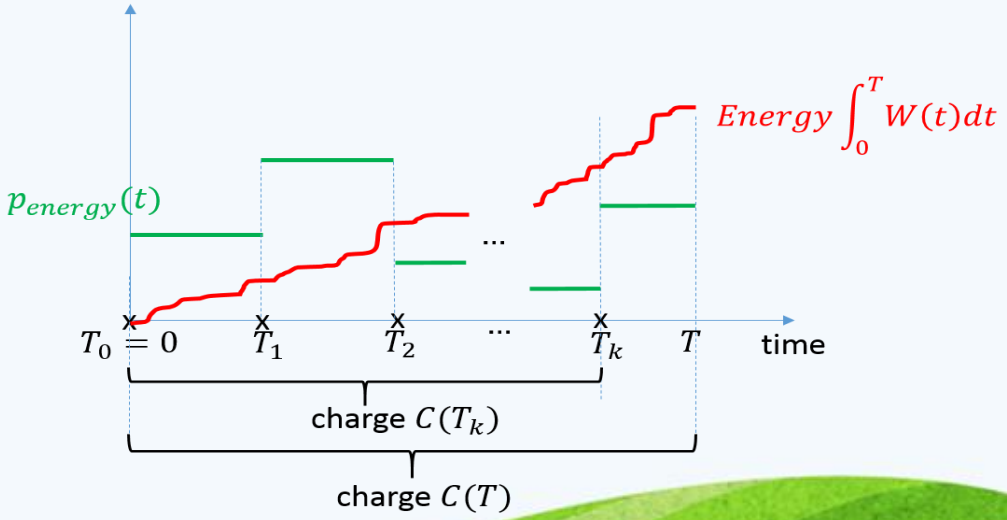
Two-Part Tariff

The price p of a VM (starting at time 0 and up to time T):

$$p = \frac{1}{T}\int_0^T p_{static}(VM, t)dt + \frac{1}{T}\int_0^T p_{energy}(t)W(t)dt$$

where,

- VM: a parameter identifying the characteristics of the VM
- $p_{static}(VM, t)$: the static price of VM at time t
- $p_{energy}(t)$: the energy price at time t
- $W(t)$: the power usage of the VM at time t



Two-Part Tariff with Energy Saving Discounts

The actual energy of an application is not known by the developers at the time the SLA is established.

Alternative: *pay a lump sum and then apply a discount based on the actual power consumption*

$$p = \frac{1}{T}\int_0^T p_{static}(VM,t)dt + \min\left\{\frac{1}{T}\int_0^T p_{energy}(t)W(t)dt - \frac{1}{T}\int_0^T p_{energy}(t)W_{nominal}dt\,, 0\right\}$$

where,

- $W_{nominal}$: the power consumption already accounted for in the static price.

❖ Any average power consumption above $W_{nominal}$ does not increase price.

Linearly Increasing Energy-Based Pricing

A traditional pricing scheme is to provide a lower price per energy unit (e.g., day / night). Incentives to shift their energy demand to less bursty periods.

Price/energy unit based on the total consumed energy (linear increasing).

$$p = \frac{1}{T}\int_0^T p_{static}(VM,t)dt \; + \min\left\{\frac{1}{T}\int_0^T c\,W^2(t)dt\,, \frac{1}{T}\int_0^T p_{energy\_upper}(t)W(t)dt\right\}$$

where

- $c\,W(t) = p_{energy}$
- c is a constant parameter set by the IaaS provider, showing how aggressively p_{energy} increases w.r.t. the total energy consumption.
- $c\,W(t) \leq p_{energy_{upper}}$ is an upper bound preventing IaaS provider to charge arbitrarily high prices

95th Percentile Rule

Similar pricing scheme already adopted by higher-tier ISPs.

Energy consumption within the infrastructure of an IaaS provider is measured or sampled and recorded (e.g., log file).

At the end of each billing cycle, the energy consumption samples are sorted from highest to lowest, and the top 5% of data is thrown away.

95th%: the next highest measurement. The customer will be billed based on that.

$$p = \frac{1}{T}\int_0^T p_{static}(VM, t)dt + \frac{1}{T}\int_0^T p_{energy}(t)l^*(t)dt$$

$l^*(t)$: the 95th% measurement of the energy consumed at time t.

l^*: $max\{l|P(W > l) \geq 0.05\}$

Service Plans

- To form the basis of business level contracts between cloud layers by specifying the responsibilities between the participating layers.
- Objective: Make **upper** layers (**PaaS** | IaaS or **SaaS** | PaaS) employ energy-aware policies
- Lower layer offers one of several **plans** to the upper layer
- **Plan =** pricing scheme + SLA + adaptation semantics

Service Plans (cont.)

- **Adoption:** A successful adoption strategy should include plans similar to the ones offered in legacy architectures in order to not exclude customers (i.e., layers, applications) which
  - do not want to change method they are charged,
  - not capable of exploiting the additional features of an energy-aware architecture.
- Adoption degrees:
  i. legacy customers
  ii. non-legacy customers which prefer legacy payments
  iii. non-legacy customers

Service Plans (cont.)

HiPerf: legacy customers with high performance SLA-related terms. Adaptation by lower layer.

Budget: legacy customers with no performance guarantees specified in the SLA. Adaptation by lower layer.

Vegan: non-legacy customers with legacy payment plan. Strict limits on maximum power usage. Adaptation by lower layer. Customer is free to adapt to make best use of system resources without exceeding the power.

Green: same as Vegan, but the power limits are higher at a higher price.

Energy-aware: non-legacy customers with energy aware payments. High / low performance, as the performance/cost tradeoff is determined by the customer's adapting actions.

Conclusions and Future Work

- Cloud providers mainly charge for their resources on the basis of fixed rates per unit of time, without taking explicitly into account the energy usage.
- Four novel energy-aware pricing schemes proposed to enhance IaaS providers choosing their optimal pricing strategy.
- A set of envisaged service plans presented intending to facilitate the gradual adoption of the ASCETiC architecture.
- *Economic Implications of Energy-Aware Pricing in Cloud*
  - *Competition among different providers*
  - *Evaluation of the different pricing schemes*
  - *Selecting the appropriate scheme based on the type of the application*

Thank you!