Analyzing Performance and Efficiency of HPC Applications in the Cloud

Vinicius Facco Rodrigues, Gustavo Rostirolla, Rodrigo da Rosa Righi, Cristiano André da Costa
Applied Computing Graduate Program, Universidade do Vale do Rio dos Sinos (Unisinos)

Presenter: Gustavo Rostirolla
Author Contact: viniciusfacco@live.com

WSPPD 2015 - XIII Workshop de Processamento Paralelo e Distribuído

August 21st, 2015

Summary

> Introduction

> Performance and Efficiency Models

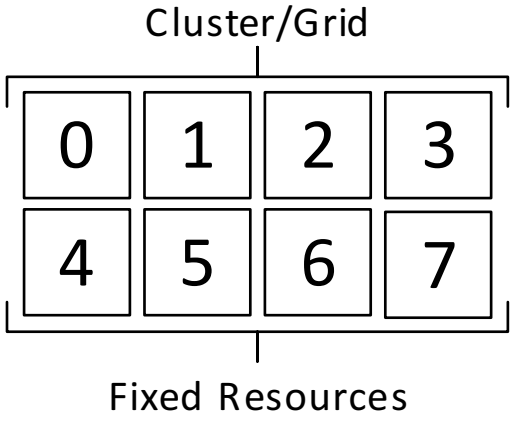
> Evaluation Methodology

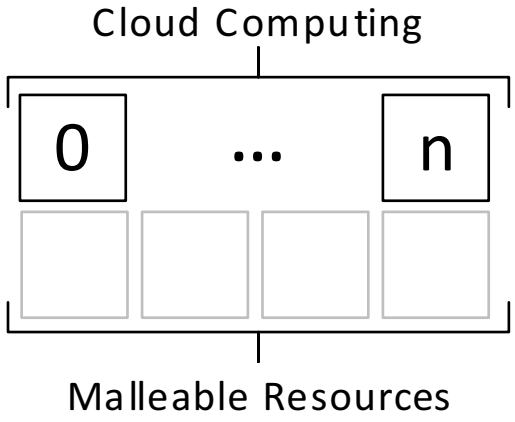
> Results

> Conclusion

Introduction

- Traditionally, high-performance computing (HPC) applications are executed on clusters or even in grid architectures;
- These environments have a fixed number of resources;
- Recently, cloud computing appears as possibility to execute HPC applications;
- One of the main cloud characteristics is the possibility to increase or decrease the number of resources on-the-fly;





Introduction

- Traditionally, the **speedup** and **efficiency** metrics are used to measure **performance** and **efficiency** of parallel applications in scenarios where the **number of processes is the same** in the entire application execution time;
- In **elastic** environments, the **number of processes can change** at any moment and the speedup metric may not be suitable;
- Related work presents a **gap** concerning defining **decision functions** for elasticity viability in the combination of **performance** and **efficiency**;
- We propose a **redefinition** of the so-called speedup and parallel efficiency metrics for elastic HPC environments: **ES (Elastic Speedup)** and **EE (Elastic Efficiency)**.

Performance and Efficiency Models

Speedup: $S(p) = \frac{t(1)}{t(p)}$

- $t(1)$: time to execute the application with $\mathbf{1}$ process (sequential time);
- $t(p)$: time to execute the application with $\boldsymbol{p}$ processes (distributed time).

Efficiency: $E(p) = \frac{S(p)}{p}$

- $S(p)$: speedup of the execution with $\boldsymbol{p}$ processes;
- p : number of processes used in the execution.

Elastic Speedup (ES)

$$ES(n, l, u) = \frac{tne(l)}{te(n, l, u)}$$

$$where\ l \leq n \leq u$$

- n: **initial** number of processes when the application is started;
- l : the **lower** amount of processes that the environment can achieve while executing the application (defined by cloud SLA);
- u: the **upper** amount of processes that the environment can achieve while executing the application (defined by cloud SLA);
- $tne(l)$: time to execute the application with $\boldsymbol{l}$ processes without elasticity. In this execution the amount of resources is not changed in the execution time;
- $te(n, l, u)$: time to execute the application in the **cloud with elastic environment** where an SLA define the lower(l) and upper(u) amounts of resources.

Elastic Efficiency (EE)

$$EE(n,l,u) = \frac{ES(n,l,u) \times n}{Resource(n,l,u)} \qquad Resource(n,l,u) = \sum_{i=l}^{u} (i \times \frac{pte(i)}{te(n,l,u)})$$

- Function $EE(n,l,u)$ computes elastic efficiency;
- Its parameters, except $pte(i)$, are the same as the preceding function ES:
  - n: **initial** number of processes when the application is started;
  - l : the **lower** amount of processes that the environment can achieve while executing the application (defined by cloud SLA);
  - u: the **upper** amount of processes that the environment can achieve while executing the application (defined by cloud SLA);
  - $te(n,l,u)$: time to execute the application in the **cloud with elastic environment** where an SLA define the lower(l) and upper(u) amounts of resources.

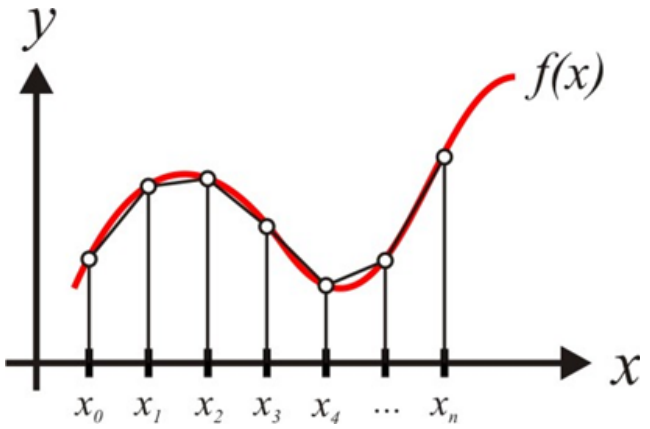
Elastic Efficiency (EE)

$$EE(n,l,u) = \frac{ES(n,l,u) \times n}{Resource(n,l,u)} \qquad Resource(n,l,u) = \sum_{i=l}^{u} (i \times \frac{pte(i)}{te(n,l,u)})$$

- Unlike equation $E(p)$, the resources here are malleable;
- To define a value that represents the **amount of resources used** in the execution, we defined the equation $Resources(n,l,u)$, where $pte(i)$ is the application's **partial** time when running over i resources;
- $Resources(n,l,u)$ captures the **amount of time** that the application was executed using $\boldsymbol{i}$ **resources**, and divide this time by the total time of execution. This value is results in a **weight** that is multiplied by i. The final result is the sum of the each possible i (from l to u) multiplied by its weight;
- EE presents parameter $\boldsymbol{n}$ in the **numerator**, which is multiplying the elastic speedup, because $ES(n,n,n)$ (**without elasticity**) is always equal to **1** and $Resources(n,n,n)$ equal to $\boldsymbol{n}$, so n in the numerator **returns** an elastic efficiency of **100%**.

Evaluation Methodology

- The application used in the tests computes the numerical integration of a function *f(x)* in a closed interval *[a; b]*;

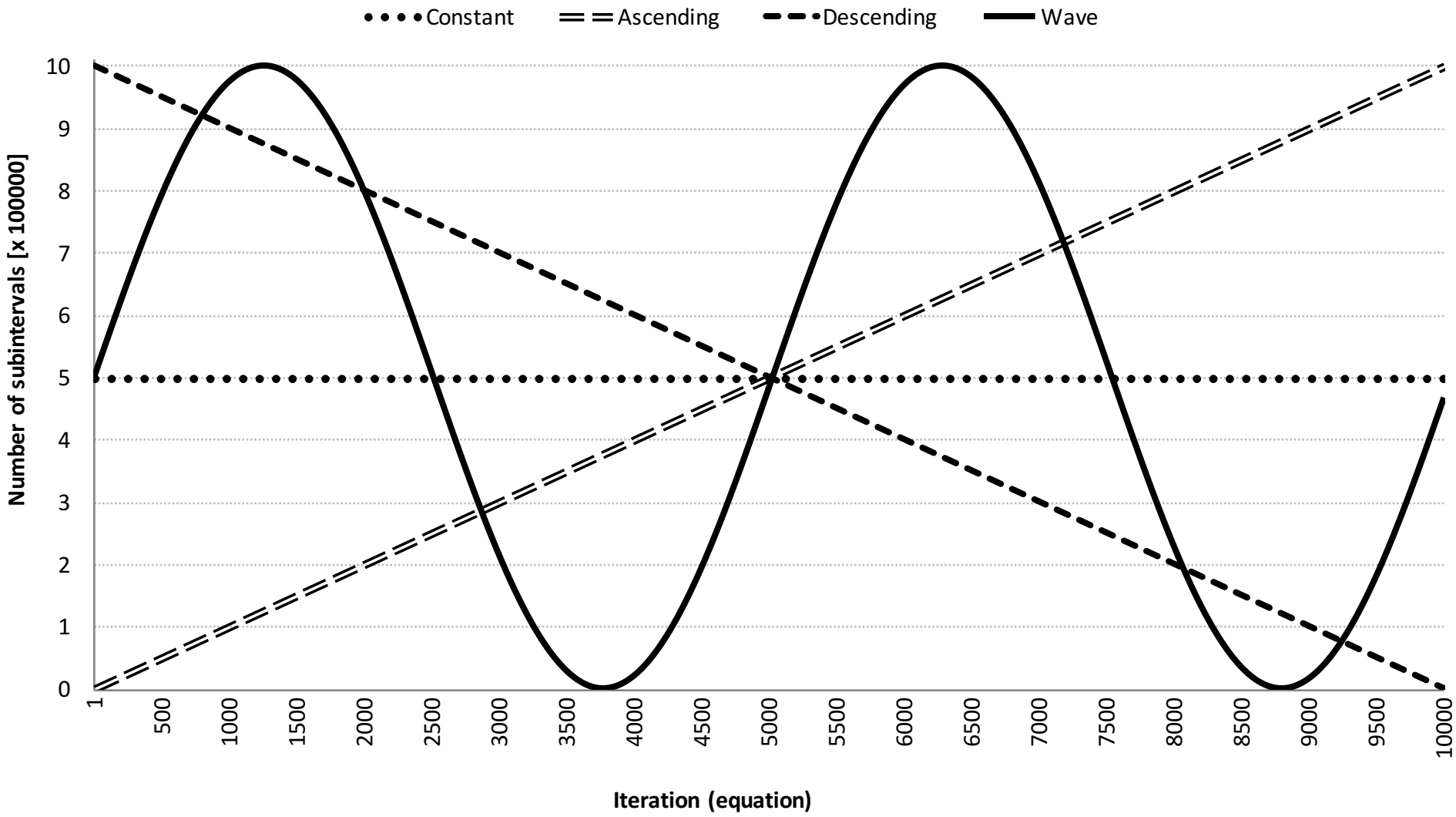


- We used the Composite Trapezoidal rule from a Newton-Cotes postulation to implement a loop-based Java application using Sockets.

- The application receive a set of equations and in each iteration distributes one equation with a different amount of subintervals between x_0 and x_n between the processes;

- To evaluate different application behavior, four patterns were modeled: Ascending, Constant, Descending and Wave.

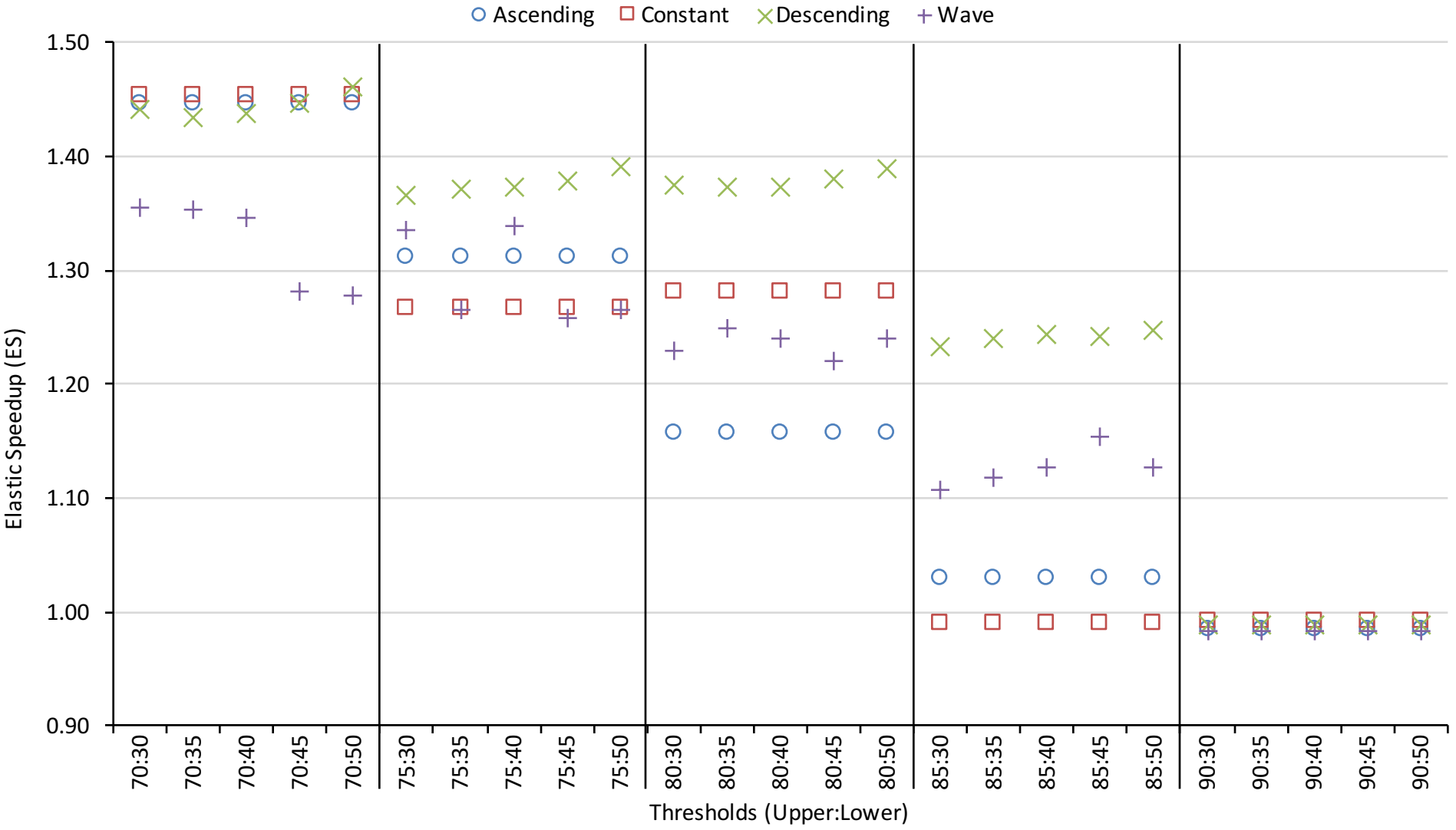
Evaluation Methodology



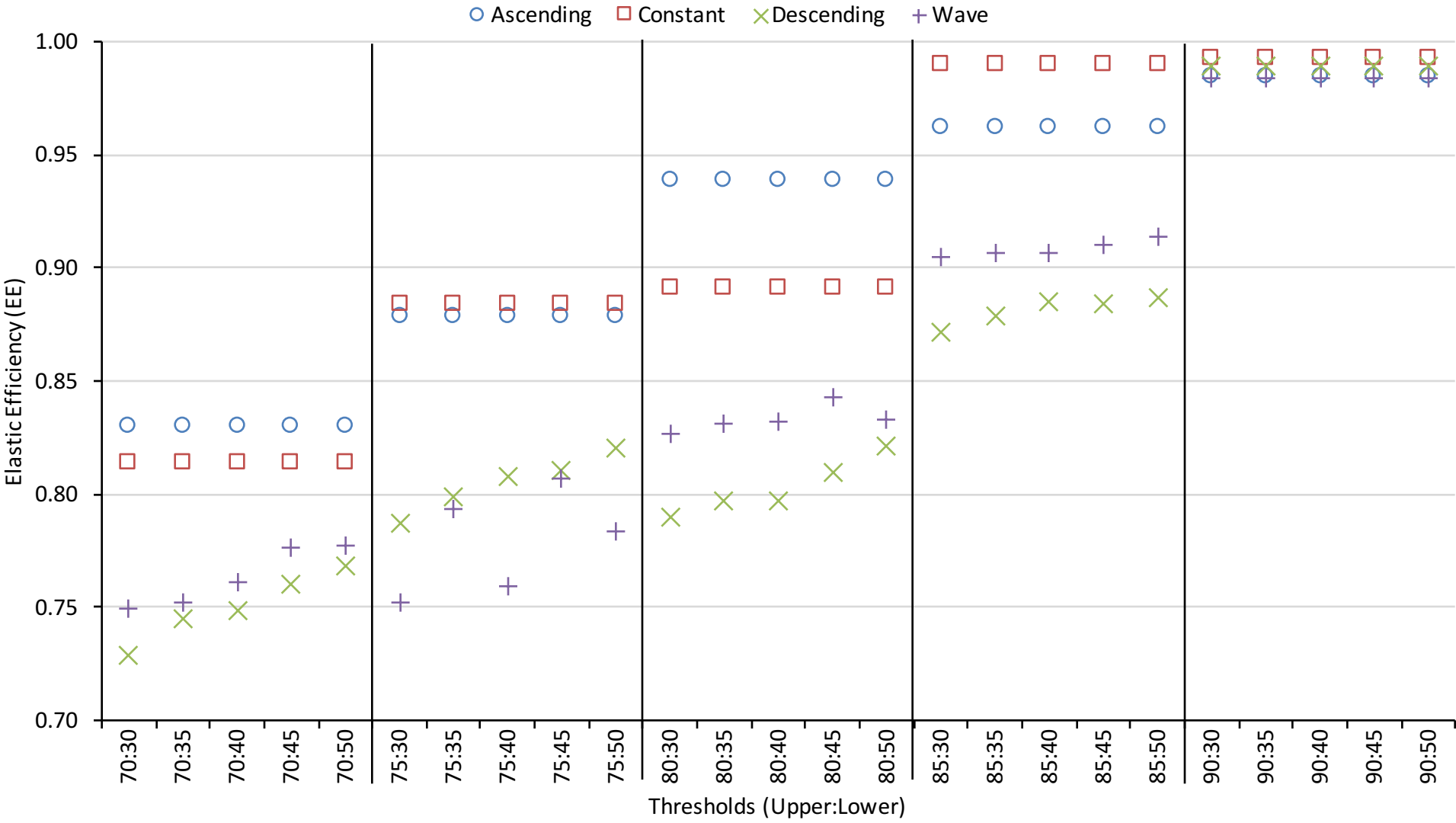
Evaluation Methodology

- The tests were executed in a private cloud environment using OpenNebula;
- We used AutoElastic to provide automatic elasticity for HPC applications. The elasticity model consists in horizontal elasticity, by virtual machine replication, based on thresholds;
- Used thresholds:
  - Lower: 30, 35, 40, 45 and 50;
  - Upper: 70, 75, 80, 85 and 90.
- Minimum of resources(*l*): 4 virtual machines;
- Maximum of resources(*u*): 10 virtual machines.

Results: Elastic Speedup (ES)



Results: Elastic Efficiency (EE)



Conclusions

- Considering the initial number of VMs and the lower and upper bounds, the work contribution explores the traditional speedup and efficiency for parallel systems, now considered in elastic infrastructures;
- Although ES provides possible gains with on-the-fly resource reorganization, EE indicates the effectiveness of resource use;
- Using a automatic elasticity based in thresholds, the variation of the upper threshold causes more significant variations on the Elastic Speedup and Elastic Efficiency;
- Different thresholds result in different resource usage impacting the application performance.

Analyzing Performance and Efficiency of HPC Applications in the Cloud

Vinicius Facco Rodrigues, Gustavo Rostirolla, Rodrigo da Rosa Righi, Cristiano André da Costa
Applied Computing Graduate Program, Universidade do Vale do Rio dos Sinos (Unisinos)

Presenter: Gustavo Rostirolla
Author Contact: viniciusfacco@live.com

WSPPD 2015 - XIII Workshop de Processamento Paralelo e Distribuído

August 21st, 2015