BSPonP2P: Towards Running Bulk-Synchronous Parallel Applications on P2P Desktop Grids

Alexandre Veith, Gustavo Rostirolla, Vinicius Facco Rodrigues and Cristiano Costa

Contact: veith.alexandre@yahoo.com.br

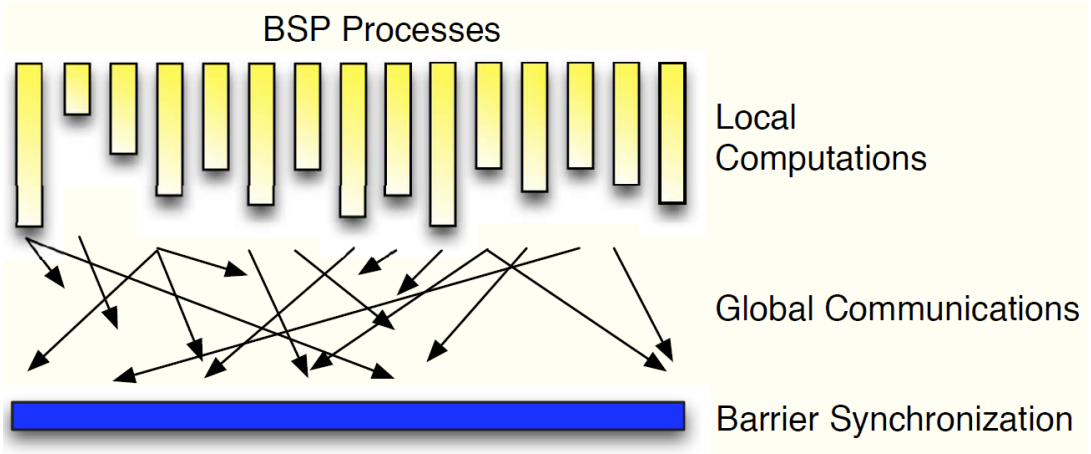
PDPTA'15 - The 21st International Conference on Parallel and Distributed Processing Techniques and Applications, Las Vegas, USA.

Outline

1. Introduction
2. Background
3. BSPonP2P Model
4. Prototype
5. Evaluation Methodology
6. Result Analysis
7. Conclusion

Introduction

- BSP (*Bulk Synchronous Parallel*) programming model
  - Applications (*sorting, broadcast, data mining, computation fluid dynamics, molecular dynamics, minimum spanning tree, LU decomposition, dynamic programming*)
  - Composition of supersteps
  - Programming facilities and idea of execution cost
  - BSP processes can be mapped arbitrarily



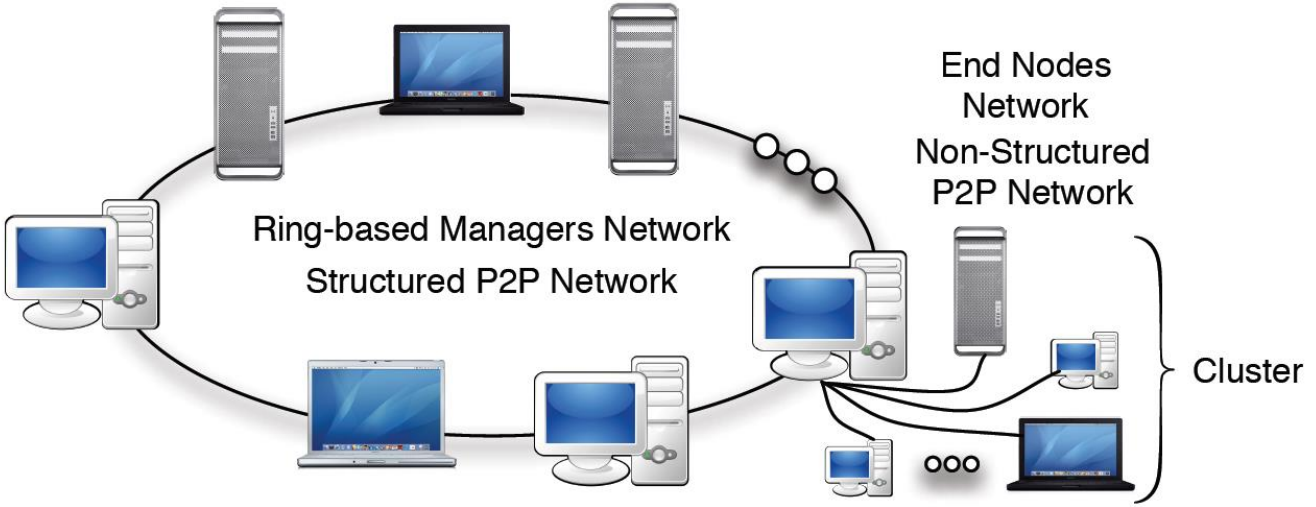
- How can we explore collaborative computing on P2P Desktop Grid (PDG) to run BSP applications efficiently?
  - Proactive processes-resources' mapping adjustment
  - Self-sufficient architecture
  - Automatic load-balancing when launching BSP application

Outline

1. Introduction
2. Background
3. BSPonP2P Model
4. Prototype
5. Evaluation Methodology
6. Result Analysis
7. Conclusion

Background

- Ring-Based Manager Network (Structured P2P Network)
  - Chord uses a DHT and a Finger table to provide message exchange and routing in an efficient, scalable and secure way
  - Provide performance for large scale deployments
- End Nodes Network (Non-Structured P2P Network) – Cluster
  - Offers better flexibility and dynamism with heterogeneous and unstable resources



Background

- Strategies to turn viable the matching involving collaborative infrastructure and the BSP programming model
  - Checkpointing brings reliability and performance saving to the model: when someone leaves the system in a superstep then a checkpointing is used to restart the application in the last saved point
  - Reescheduling, in its turn, aims to covering dynamism, since both nodes and networks can become overloaded at application runtime; so, process can be on-the-fly migrated to novel locations to improve application performance

Background

- PM receive the inputs of a process i and a cluster j
- Comp, Comm and Mem denote the computation, communication and memory metrics
- The cluster j that has the larger PM value is the most profitable target to receive the process I

$$PM(i,j) = Comp(i,j) + Comm(i,j) - Mem(i,j)$$

Background

- T(i) and Set(j) are inherited from MigBSP, and denote the computational time of process I in the last superstep and the relative performance of the cluster j, repectively
- BSPonP2P adds $\bar{x}_{Resource}$ and $\bar{x}_{User}$

$$Comp(i,j) = \left(\frac{\overline{X}_{Resource(j)} + \overline{X}_{User(j)}}{2}\right).T(i).Set(j)$$

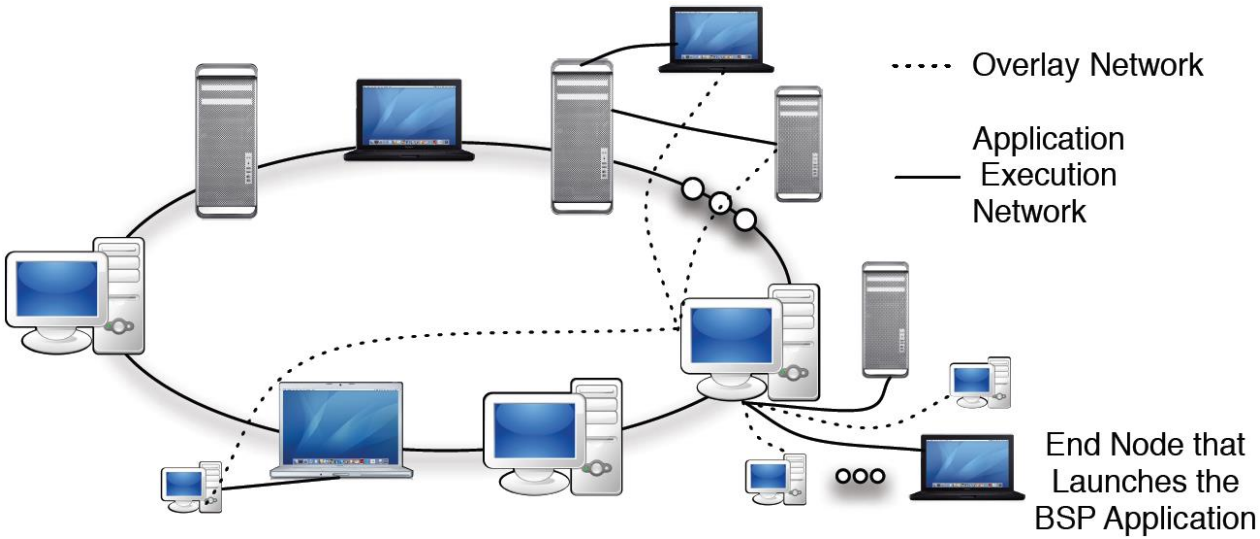
Outline

1. Introduction
2. Background
3. BSPonP2P Model
4. Prototype
5. Evaluation Methodology
6. Result Analysis
7. Conclusion

BSPonP2P Model

- The communication level is divided into two levels, depending on the node role's:
  - First level comprises communication among Managers
  - Second level represents an interaction between a Manager and End Node

1 – End Node submits the BSP demand to it Manager
2 – Manager choose the target cluster for each process (first-level)
3 – Cluster Manager define the End Node under its responsability to run a process (second-level)
4 – After selecting on End Node per process, an Execution Network is composed
5 – Rescheduling according with the PM (Potential of Migration) evaluation
6 – At rescheduling the checkpointing and migrating are executed



Outline

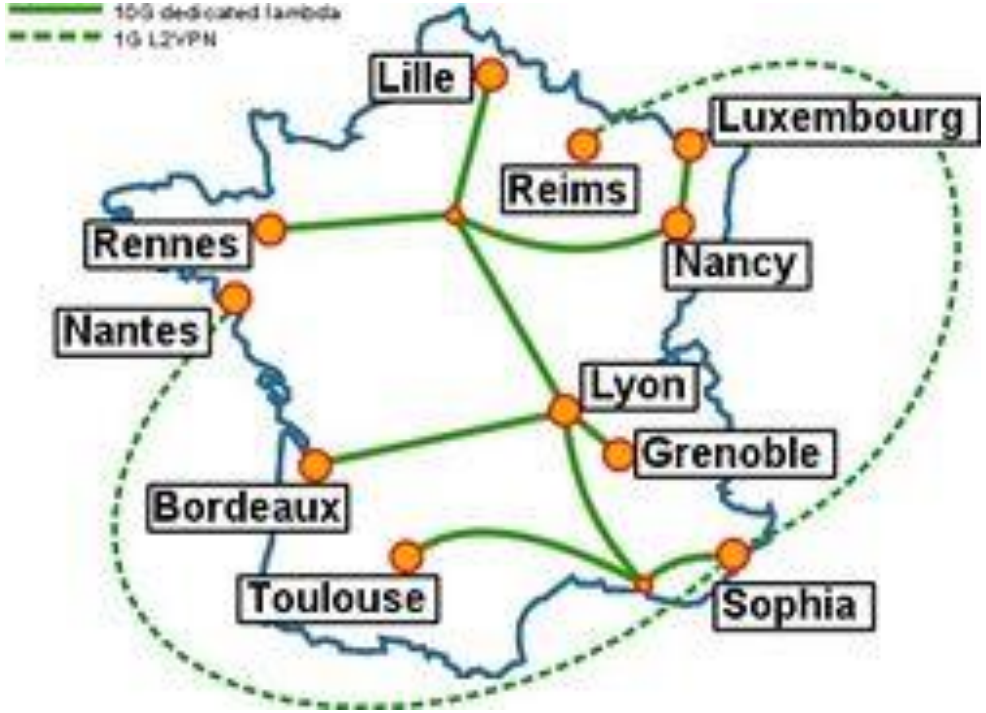
1. Introduction
2. Background
3. BSPonP2P Model
4. Prototype
5. Evaluation Methodology
6. Result Analysis
7. Conclusion

Prototype

- Linux
- SimGrid Simulation
  - MSG module
- Grid'5000 platform
- Computation Pattern (*Pcomp*) varies depending on the configuration
  - Scenario ii and iii

Prototype

- Total 150 nodes (9 clusters)
  - chimint and chicon located in Lille
  - paradente from Rennes
  - graphene from Nancy
  - gdx from Orsay
  - capricorne from Lyon
  - Adonis from Grenoble
  - borderplage from Bordeaux
  - pastel from Toulouse
  - suno from Sophia



Outline

1. Introduction
2. Background
3. BSPonP2P Model
4. Prototype
5. Evaluation Methodology
6. Result Analysis
7. Conclusion

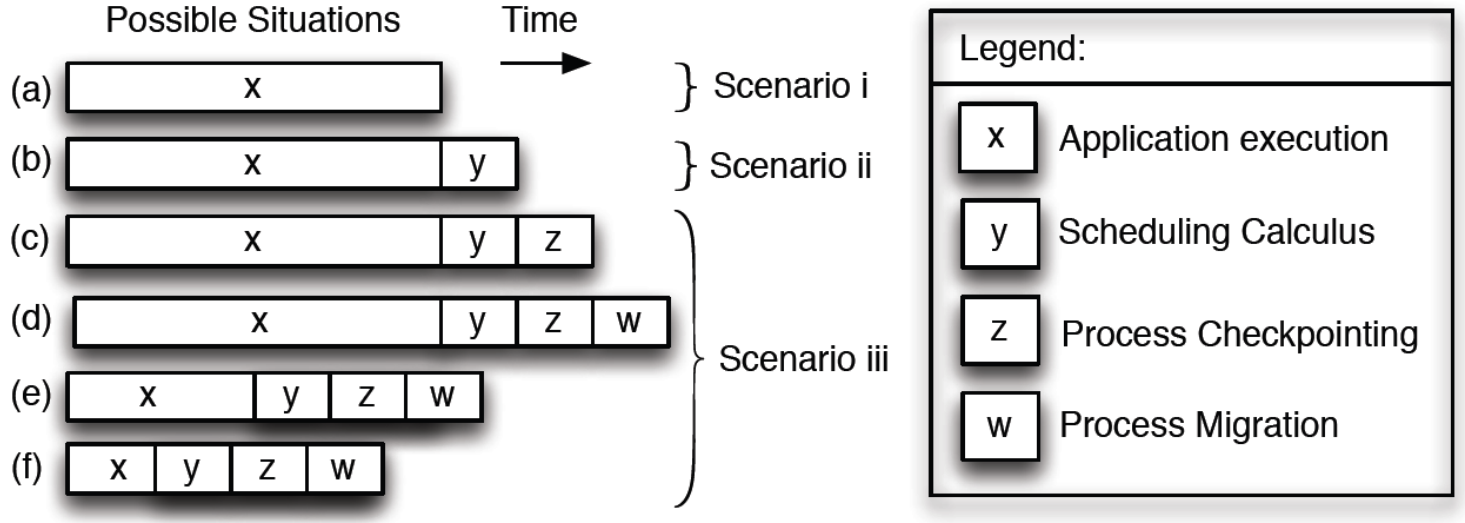
Evaluation Methodology

- BSPonP2P's differential approach is highlighted by the adoption of process migration and checkpointing
- The Figure illustrates different scenarios after running a BSP application using BSPonP2P

Scenario i: represents the simple execution, disabling any service or scheduling functionality

Scenario ii: adds the scheduling calculus in the first and second levels of CON

Scenario iii: this scenario enables process chekpointing and reescheduling



Situation f is the best execution, because beside have all services running the time is smaller then the situation a. Although situation e has a larger time when compared to situation a, it was computed using the checkpointing strategy

Evaluation Methodology

- Scenarios of tests
  - Scenario I
    - Application
  - Scenario II
    - Application + Model – Migration – Checkpointing
  - Scenario III
    - Application + Model + Migration + Checkpointing
  - Scenario IV
    - Checkpointing usage to recovery the system

Tests conducted in each scenario suffered the 3 parameters' variation :

1 – Alpha (4, 8 and 16)

2 – Supersteps (10, 50, 100, 500, 1000 and 2000)

3 – Process (11, 26, 51 and 89)

Possobiles Comparasons

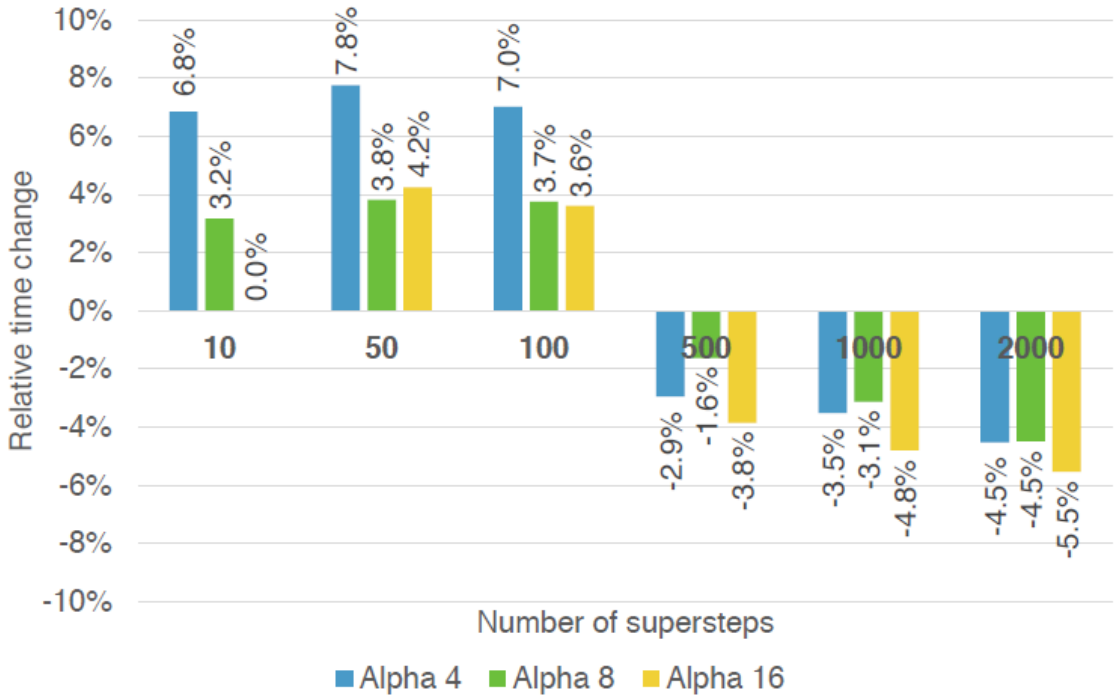
- Between scenarios i and ii
  - Observe the model's intrusiveness on application execution
- Between scenarios i and iii
  - Analyze the performance gain/loss with processes migrations
- Between scenarios ii and iii
  - Observe changes occurred on application execution taking into account performed migrations

Outline

1. Introduction
2. Background
3. BSPonP2P Model
4. Prototype
5. Evaluation Methodology
6. Result Analysis
7. Conclusion

Result Analysis

Relative time variation of scenario iii when compared to scenario i varying the number of supersteps with 26 processes

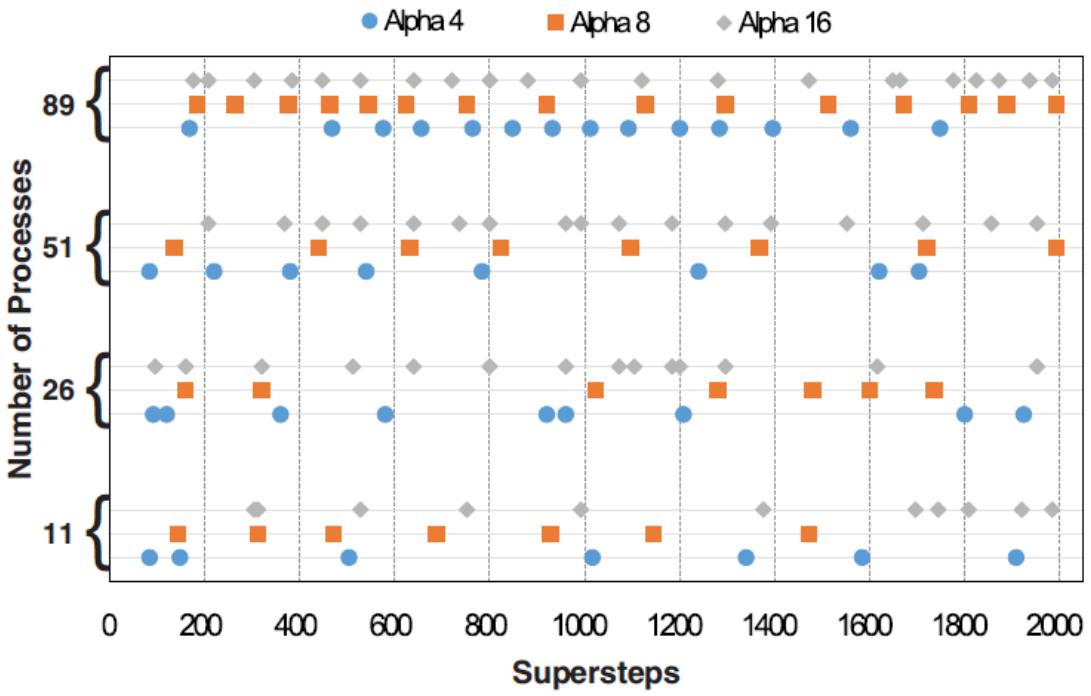


With the number of supersteps above 500 there is a decrease in the execution time, varying:

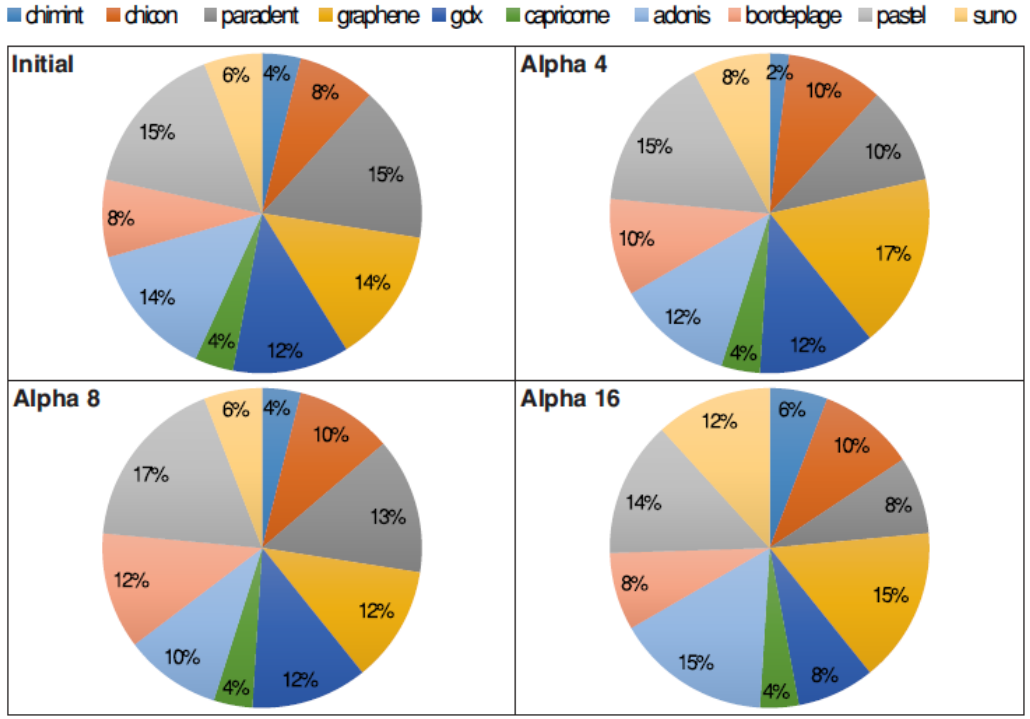
- -2.9% and -4.5% when alpha is equal to 4
- -1.6% and -4.5% when alpha is equal to 8
- -3.8% to -5.5% when alpha is 16

BSPonP2P Application: Results

Migrations distribution along the application execution varying the alpha value and number of processes

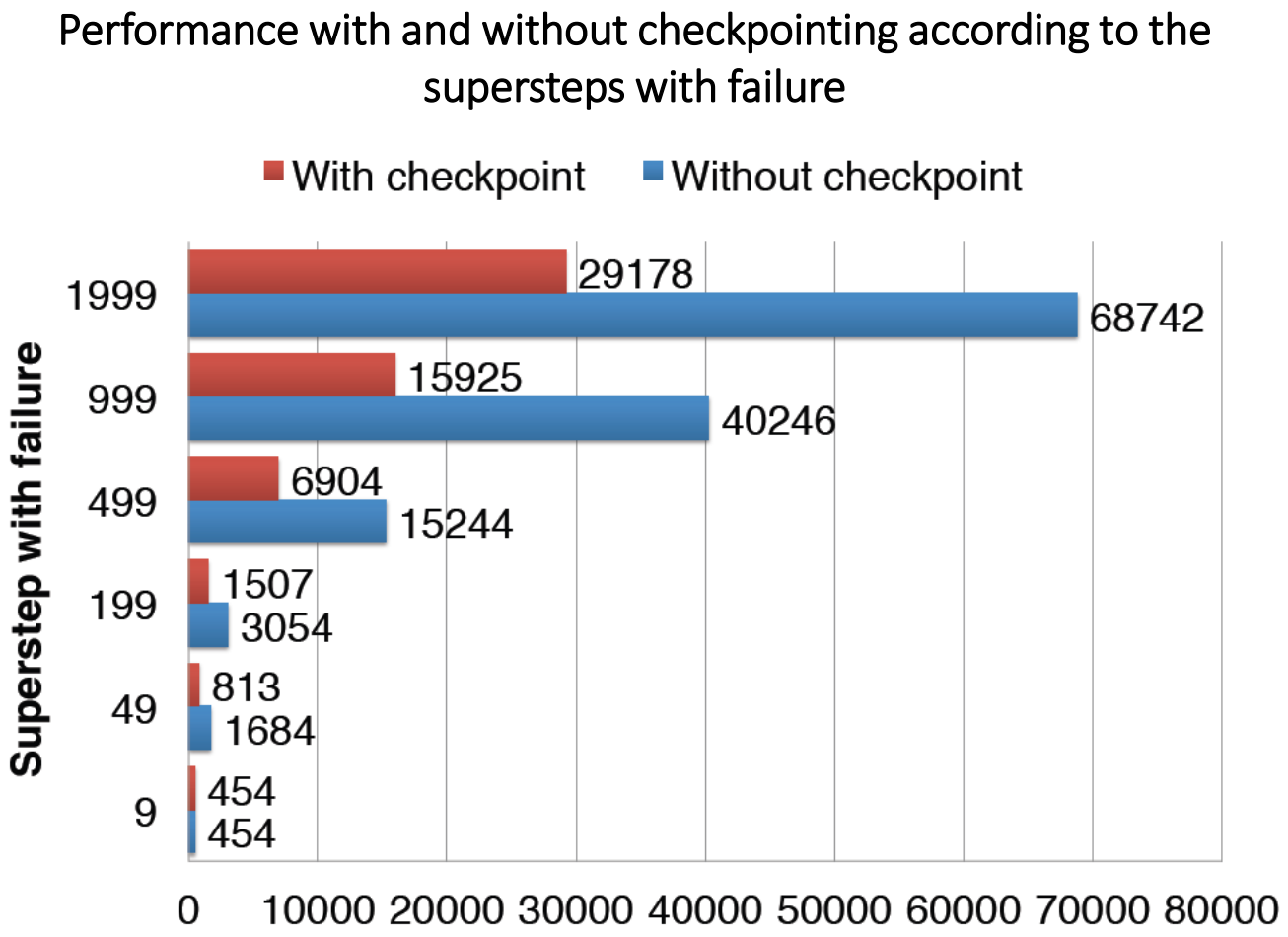


Distribution of 51 processes among the clusters. The first graph indicates the initial distribution and the others the final distribution according to the alpha values



Despite of better computational resources of cluster Graphene (144 CPUs Xeon X3440, 16 GB memory and Infiniband-20G) when compared to Chicon (52 CPUs Opteron 285, 4 GB memory and Myri-10G) for instance no migration pattern to this cluster can be detected

BSPonP2P Application: Results



Superstep 1999 and the last checkpoint in the superstep 1016, an economy of more than 57% in time could be obtained

Outline

1. Introduction
2. Background
3. BSPonP2P Model
4. Prototype
5. Evaluation Methodology
6. Result Analysis
7. Conclusion

Conclusion

- BSPonP2P
  - Sliding interval for processes rescheduling calls
  - Computation and Communication Patterns
  - Multiple metrics: Computation, Communication and Memory
- The option to migrate a percentage of processes was pertinent, since we can relocate all processes from a slower cluster to a faster one
- The application behavior implies that the processors may present variations on their load during the supersteps, changing their viability to receive processes
- Checkpointing gives a fault control, because the application must not be restarted from the scratch when any fault occurs (either when a node crashes or when an user sudden leaves the collaborative infrastructure)

BSPonP2P: Towards Running Bulk-Synchronous Parallel Applications on P2P Desktop Grids

Alexandre Veith, Gustavo Rostirolla, Vinicius Facco Rodrigues and Cristiano Costa

Contact: veith.alexandre@yahoo.com.br

PDPTA'15 - The 21st International Conference on Parallel and Distributed Processing Techniques and Applications, Las Vegas, USA.