Estimation of MPI Application Performance on Volunteer Environments

Girish Nandagudi, Jaspal Subhlok, Edgar Gabriel
Department of Computer Science, University of Houston
jaspal, egabriel@uh.edu

Judit Gimenez
Barcelona Supercomputing Center
judit@bsc.es

Overview

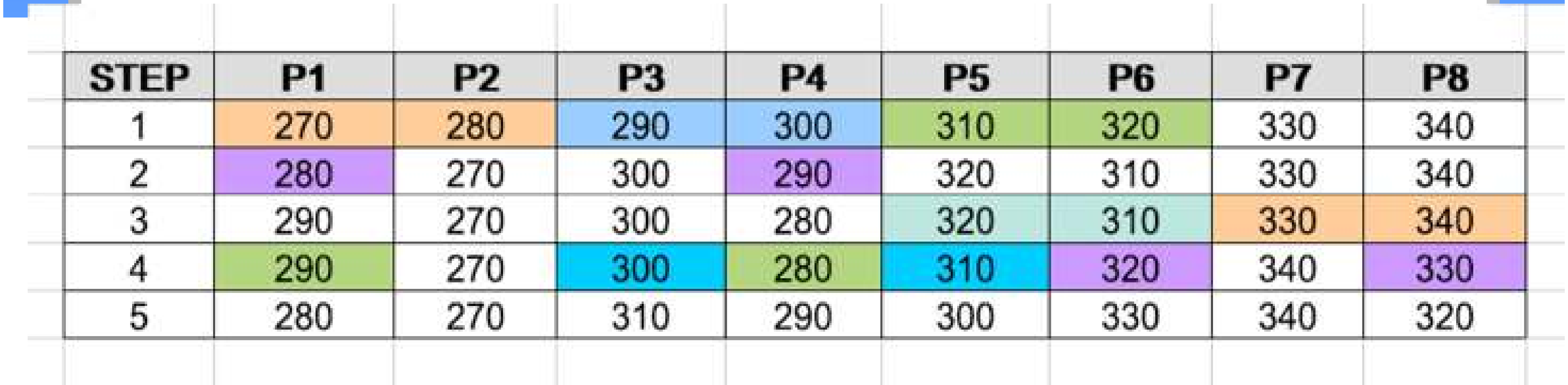
- The Volpex project
- Project Goals
- Simulation Results
- Summary

The Volpex project

- Volpex Project Goals:
  - Efficient execution of communicating parallel programs in ***heterogeneous*** environments with ***high failure rates***

- Motivation
  - High probability for process failures by long running application using thousands of processes
  - Unpredictable behavior of systems in a distributed environment (volunteer computing, grid computing, clouds)

REMD – Temperature swapping between replicas



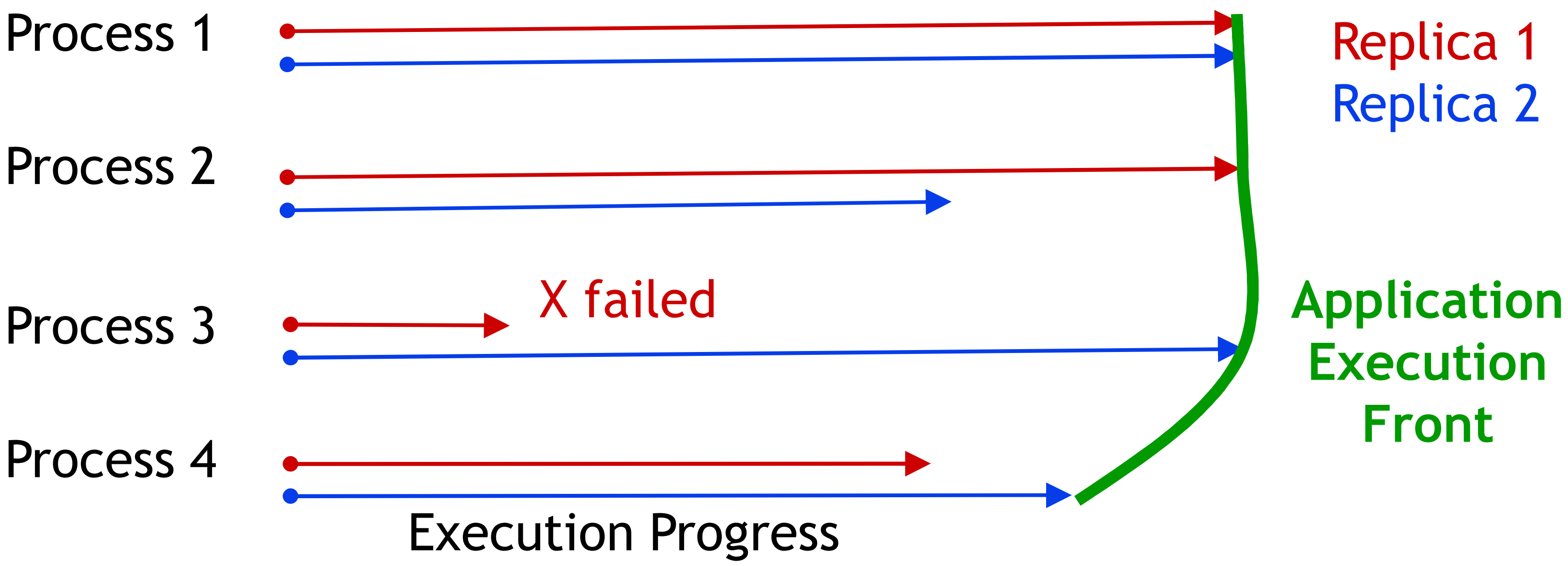
STEP	P1	P2	P3	P4	P5	P6	P7	P8
1	270	280	290	300	310	320	330	340
2	280	270	300	290	320	310	330	340
3	290	270	300	280	320	310	330	340
4	290	270	300	280	310	320	340	330
5	280	270	310	290	300	330	340	320

Processes that have the same background color swap temperatures in that time step

- Collaboration with Prof. Cheung, Department of Physics, UH
- A approach for studying the folding thermodynamics of small to modest size proteins in explicit solvent
- High computational requirements coupled with relatively small (!= 0) communication requirements

The Volpex Approach

Redundancy and/or independent checkpoint/restarts
→ *multiple physical processes per logical process*



Volpex Goals:

- Application progress tied to the fastest process replica(s)
- Seamless progress despite failures
- Minimum overhead of redundancy

Dataspace Programming Model

- Independent processes communicate with one way, PUT/GETs with an abstract dataspace
  - Similar to Linda, Javaspaces, Tspaces etc.

`PUT (tag, data)`	place data in dataspace indexed with tag
`READ (tag, data)`	return data matching the tag
`GET (tag, data)`	return and remove data matching tag

- Volpex approach implies redundant execution of processes

  ➔ a logical PUT/GET may be executed many times

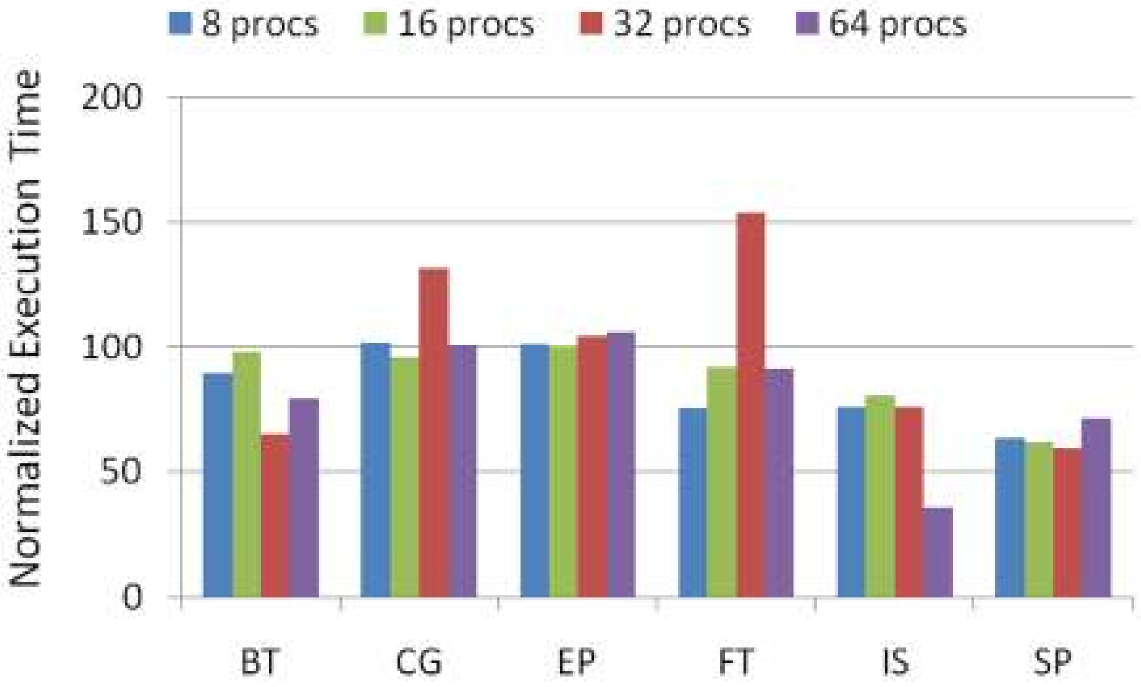
  ➔ a late replica may PUT a value that is out of date

VolpexMPI

- MPI library for execution of parallel application on volatile nodes
- Key features:
  - controlled redundancy: each MPI process can have multiple replicas
  - Receiver based direct communication between processes
  - Distributed sender logging
- Prototype implementation supports ~40 MPI functions
  - point-to-point operations (blocking and non-blocking)
  - collective operations
  - communicator management

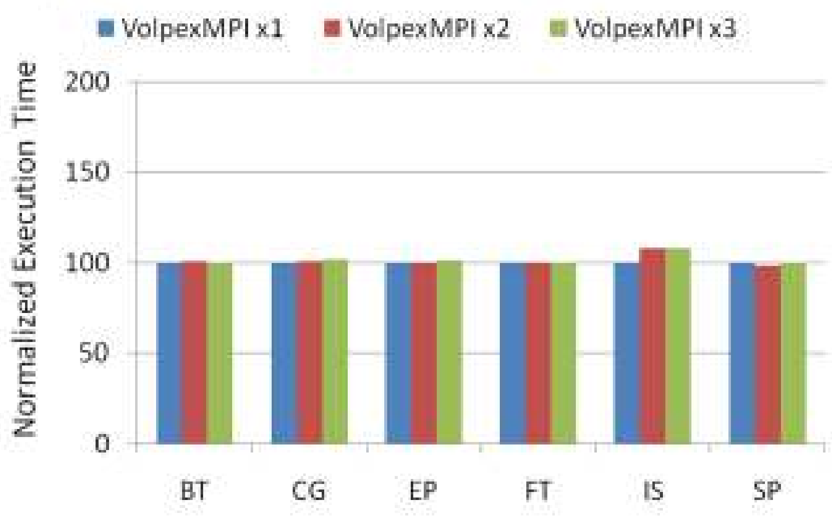
NAS Parallel Benchmarks

- Normalized execution times of VolpexMPI on a dedicated cluster over Gigabit Ethernet
- Open MPI v1.4.1 reference times are 100

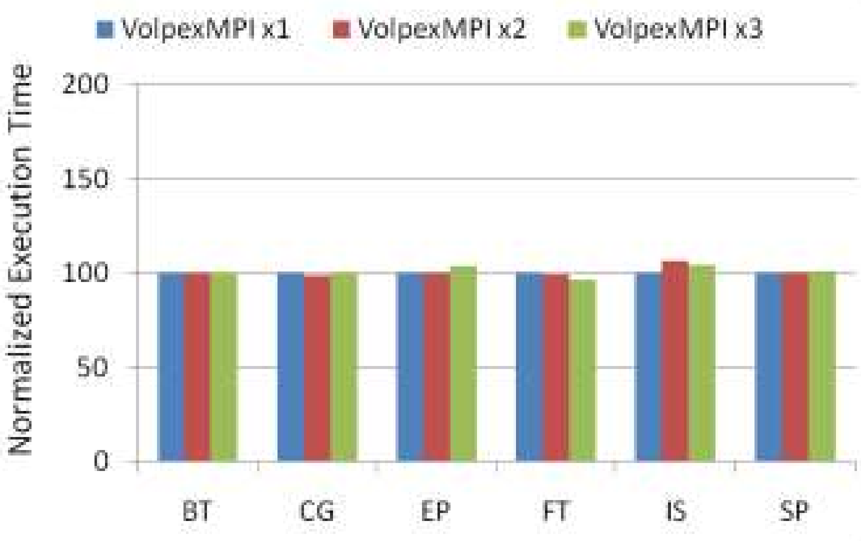


Influence of redundancy level

- Performance impact of executing one (x1), two (x2) and three (x3) copies of each process
- Normalized to the single redundancy VolpexMPI execution times



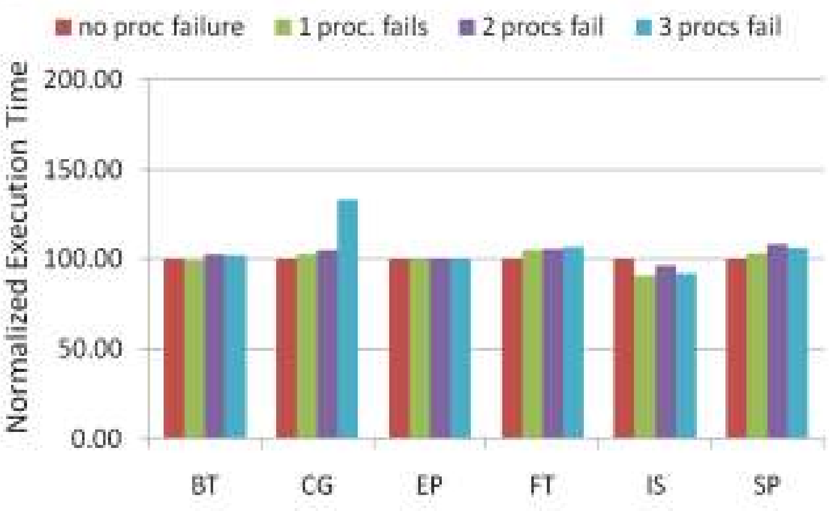
8 processes



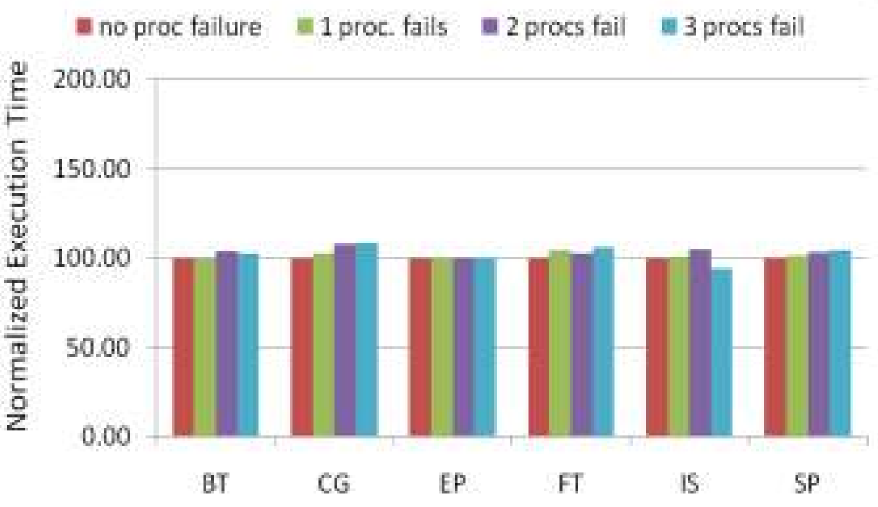
16 processes

Influence of process failures

- Double redundancy
- Failing processes from both teams
- Normalized to the double redundancy execution times



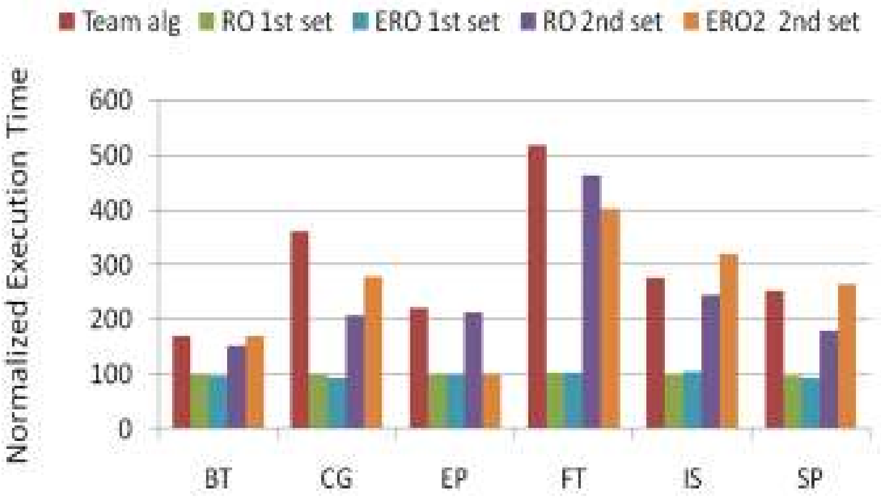
8 processes



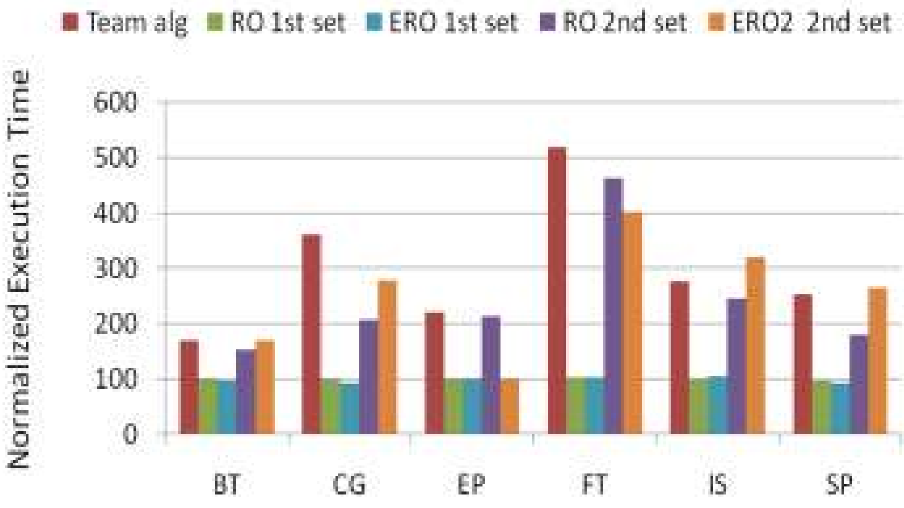
16 processes

Target Selection in Heterogeneous Settings

- Double redundancy tests on a heterogeneous configuration
  - fast nodes: Gigabit Ethernet, 2.2 GHz
  - slow nodes: Fast Ethernet, 1.0 GHz
- Initially, both teams contain processes on fast and slow nodes
- Each MPI rank has one fast and one slow process
- Normalized towards double redundancy numbers on GE



8 processes



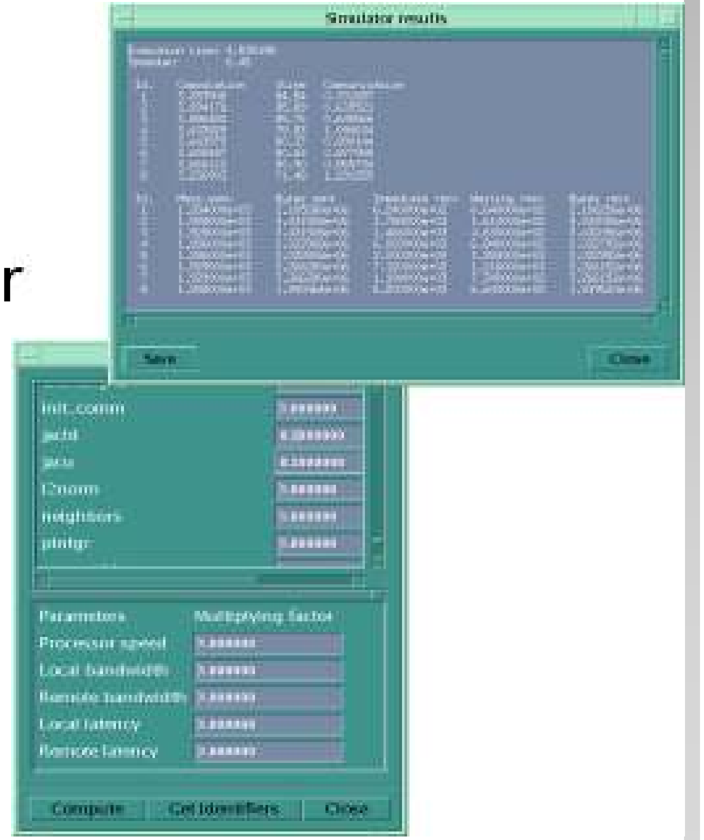
16 processes

Goal if this study

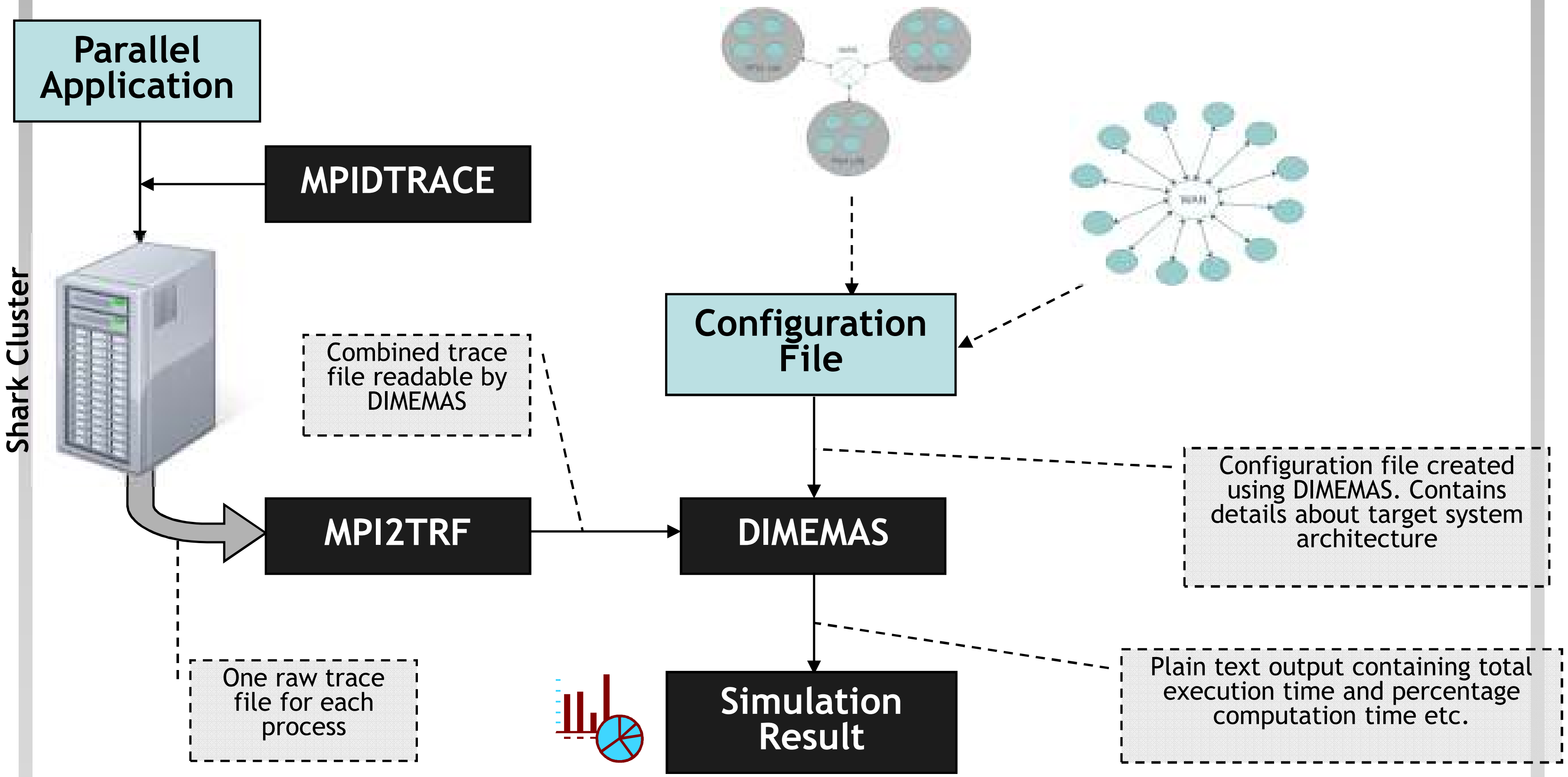
- To simulate the performance of parallel applications on desktop grids
  - To estimate the effects of bandwidth, latency on the performance
  - To estimate the effects of occurrence of failure and overheads of fault tolerance mechanisms on the performance
- To estimate the usage potential of desktop grids to run parallel applications
- Focusing on heterogeneity on the networking level

Dimemas

- Application performance analysis and prediction tool for message passing programs
- Developed and distributed by Barcelona Supercomputing Center (BSC)



Simulation Procedure



Parallel Application

MPIDTRACE

Shark Cluster

Combined trace file readable by DIMEMAS

Configuration File

Configuration file created using DIMEMAS. Contains details about target system architecture

MPI2TRF

DIMEMAS

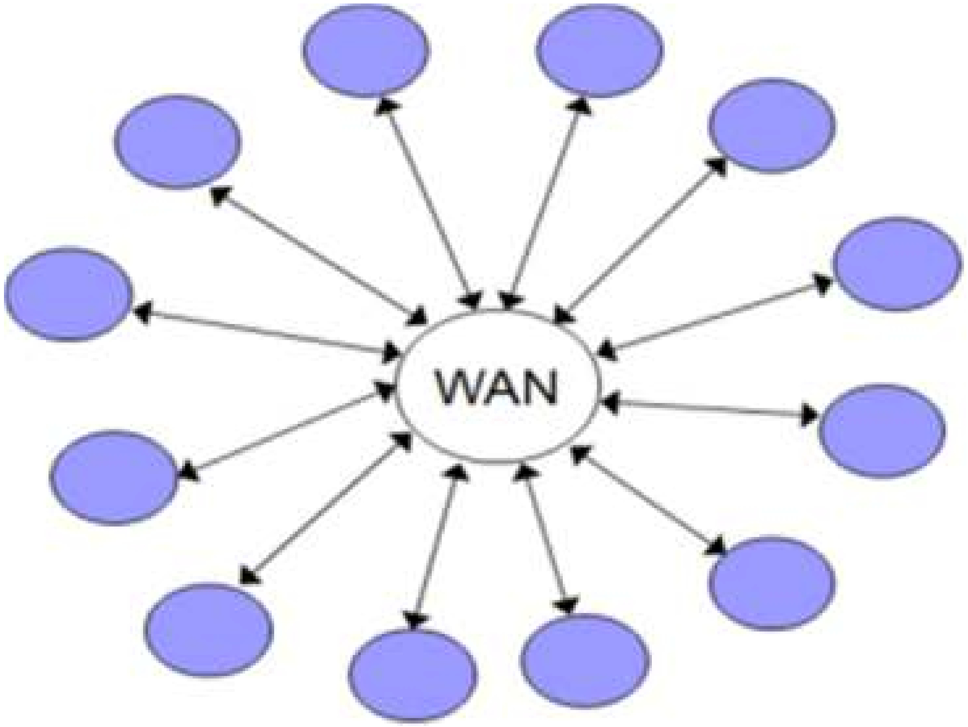
One raw trace file for each process

Simulation Result

Plain text output containing total execution time and percentage computation time etc.

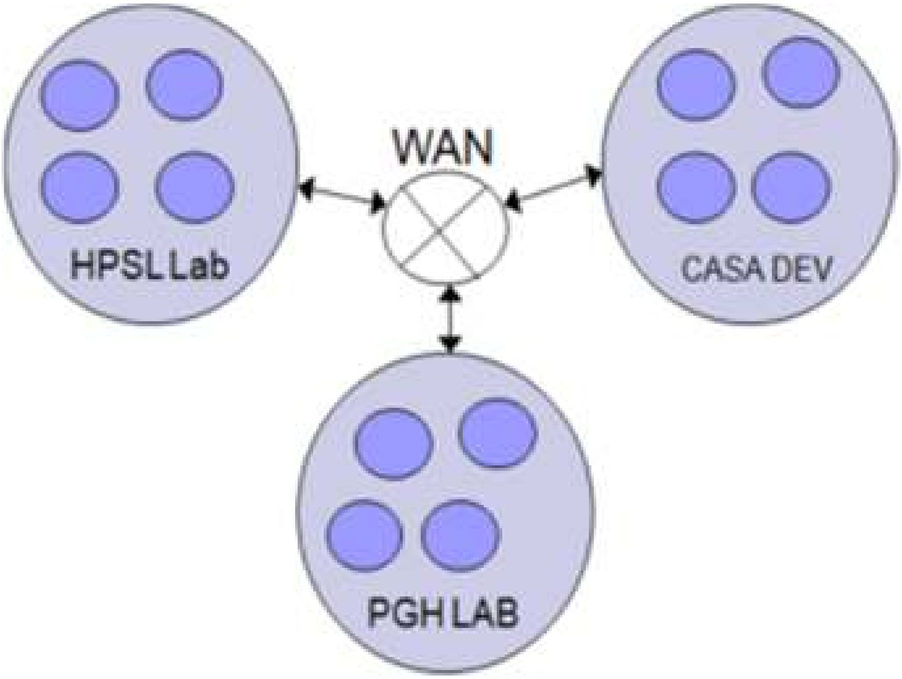
Network Configurations (I)

Desktops over Internet (DOI)

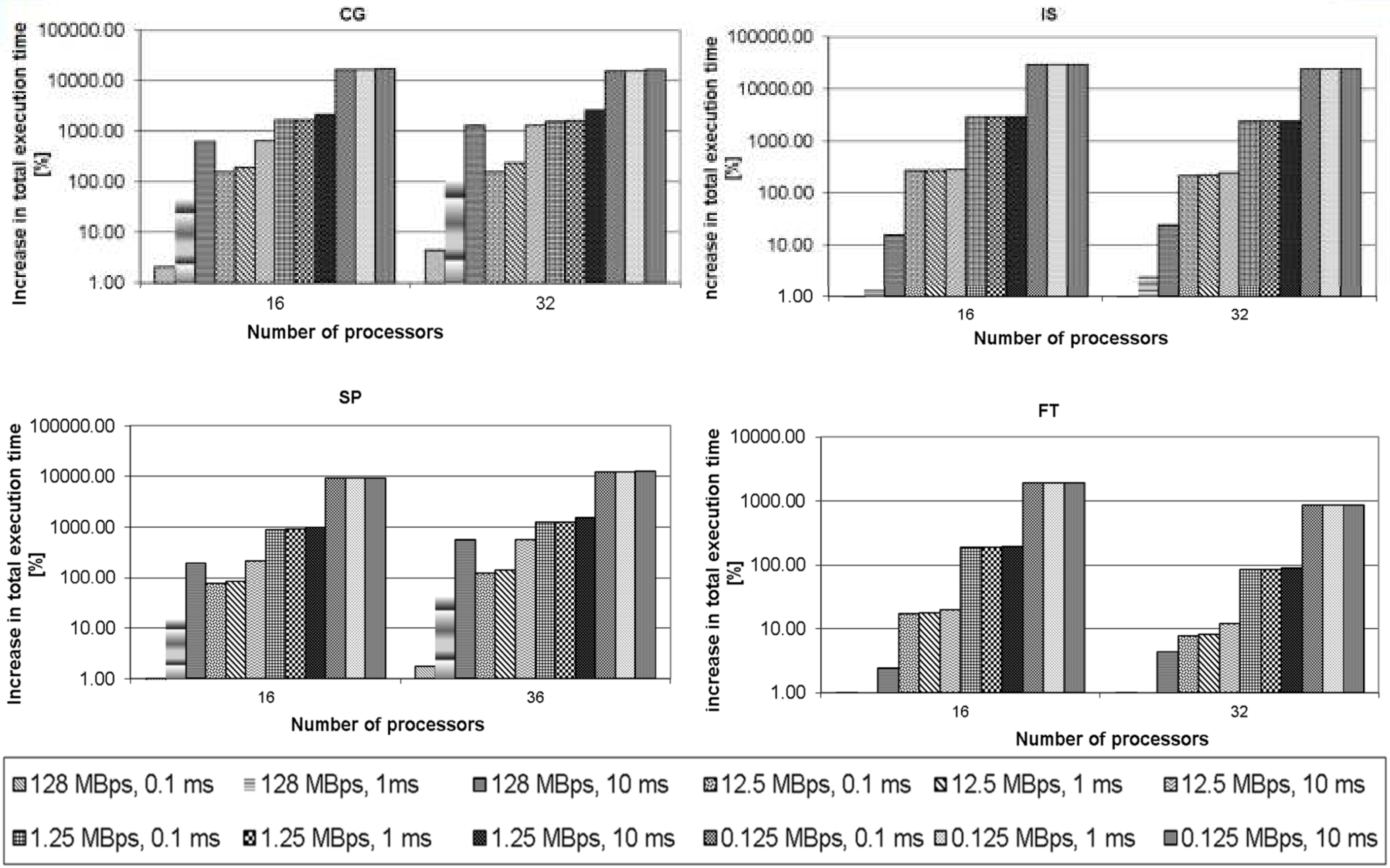


Network Configurations (II)

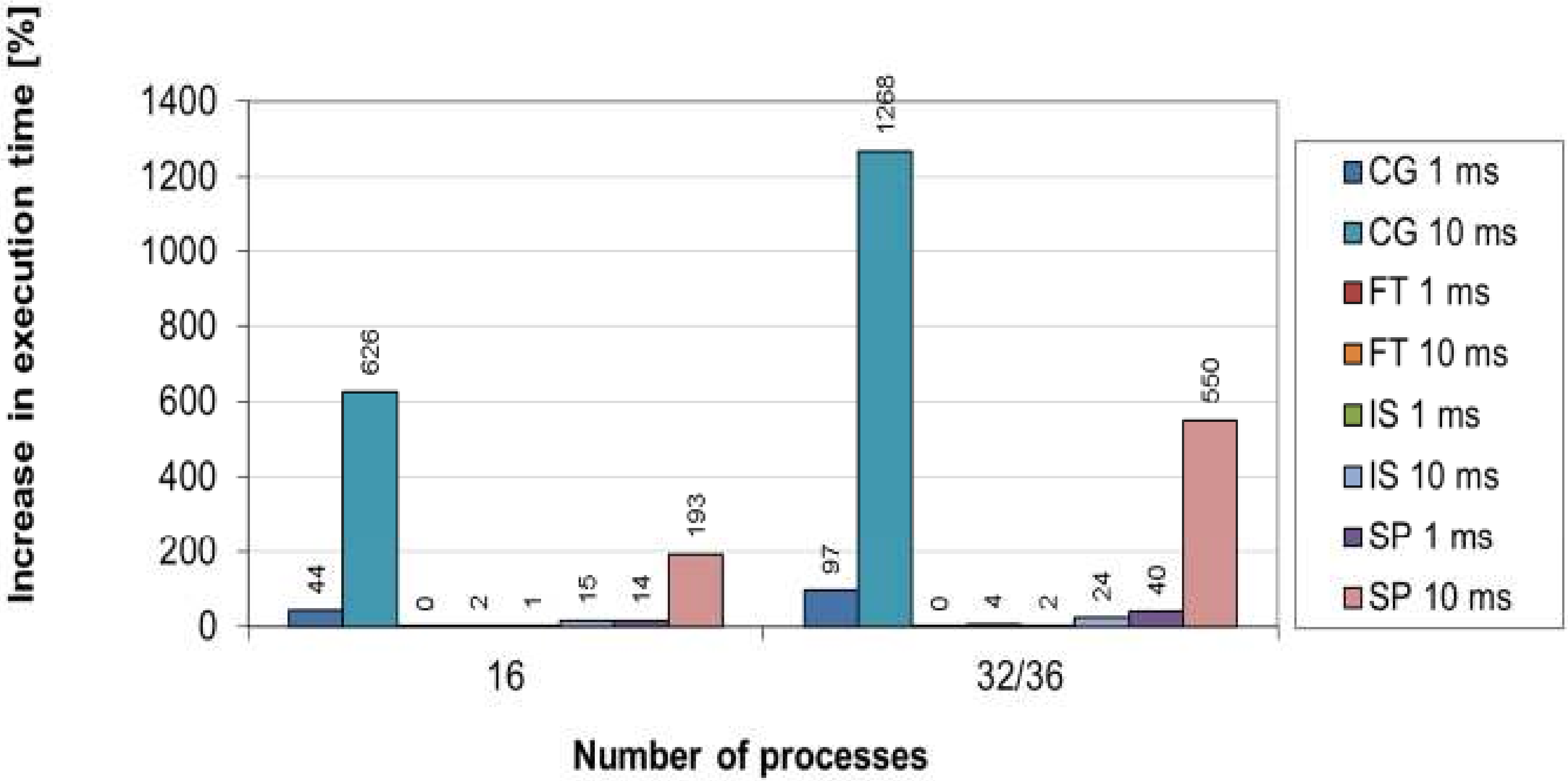
Collection of Networked Computers (CNC)



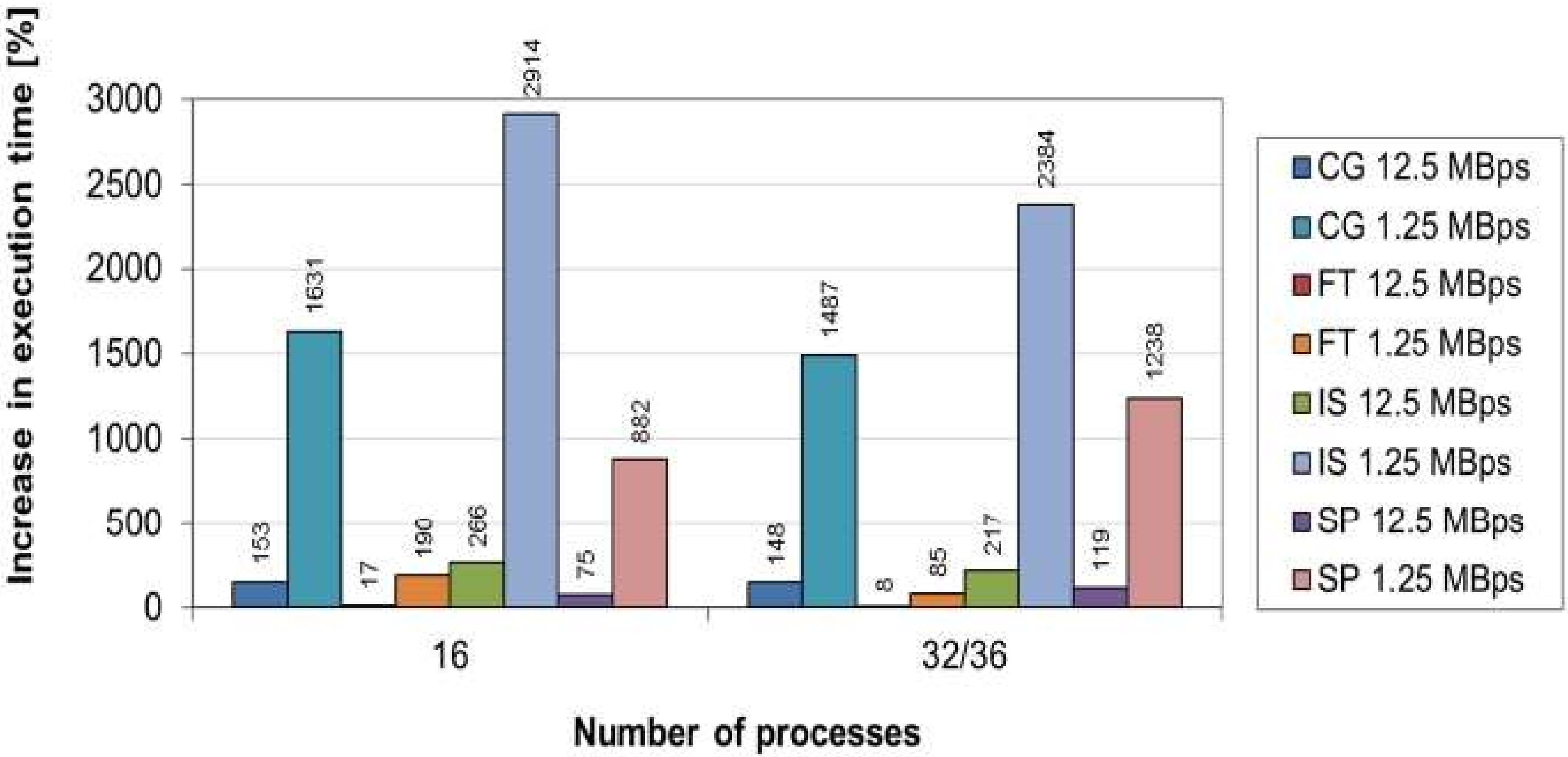
DOI


CG
IS
SP
FT
Increase in total execution time [%]
100000.00
10000.00
1000.00
100.00
10.00
1.00
16
32
36
Number of processors
▧128 MBps, 0.1 ms
≡128 MBps, 1ms
▩128 MBps, 10 ms
▨12.5 MBps, 0.1 ms
▧12.5 MBps, 1 ms
▨12.5 MBps, 10 ms
▦1.25 MBps, 0.1 ms
☒1.25 MBps, 1 ms
■1.25 MBps, 10 ms
▩0.125 MBps, 0.1 ms
□0.125 MBps, 1 ms
▩0.125 MBps, 10 ms

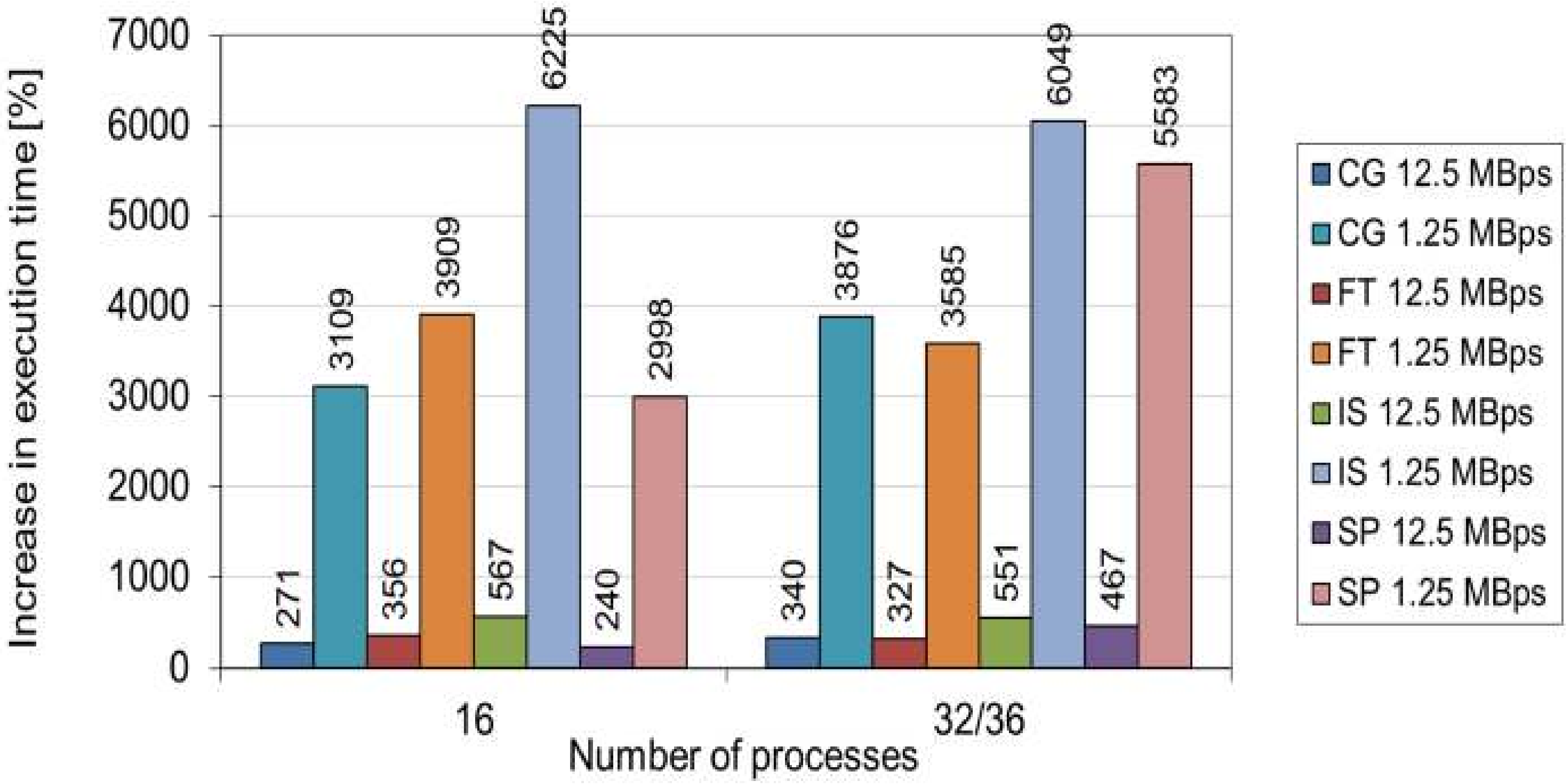
DOI with fixed Bandwidth



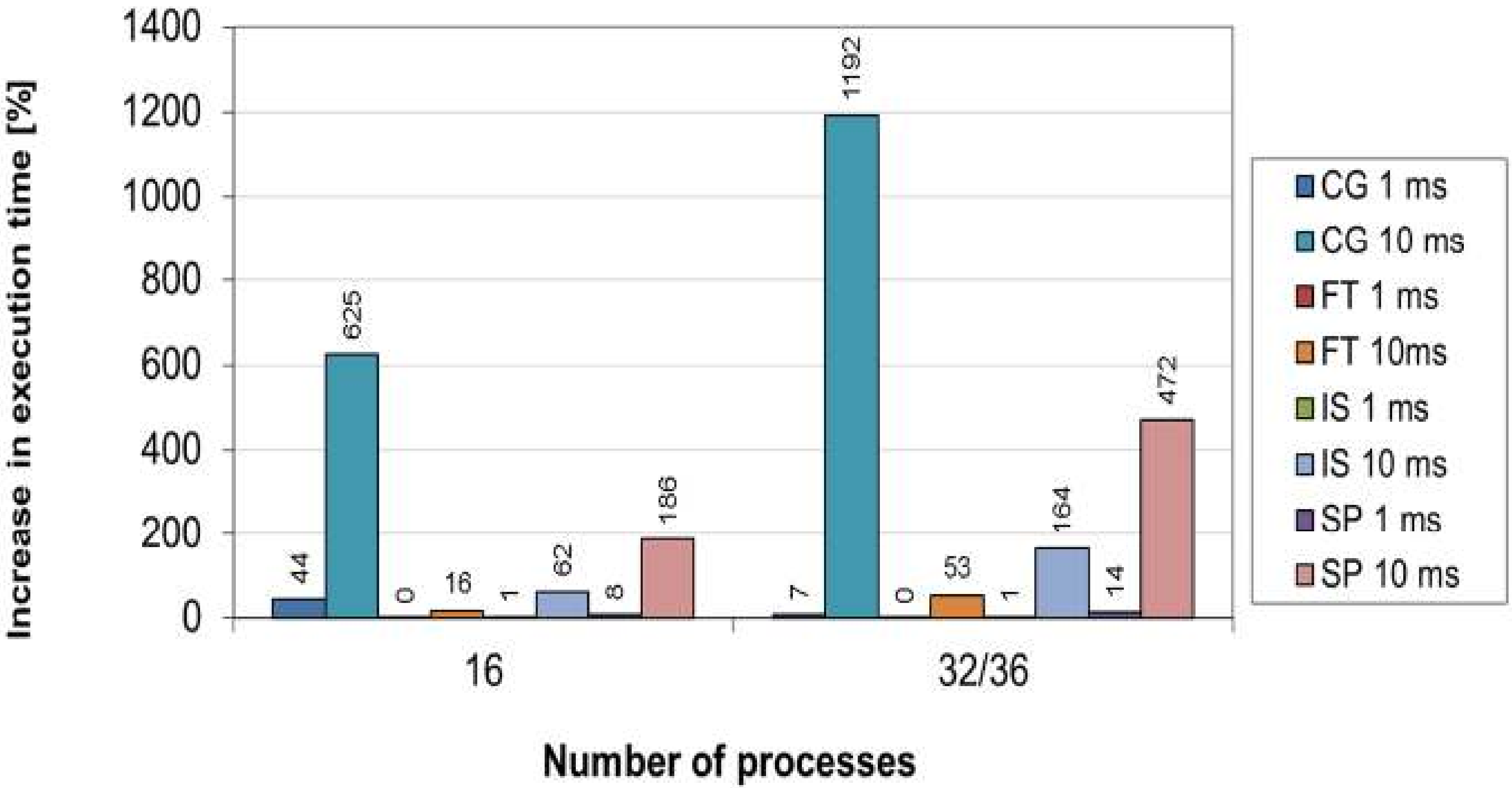
DOI with fixed latency



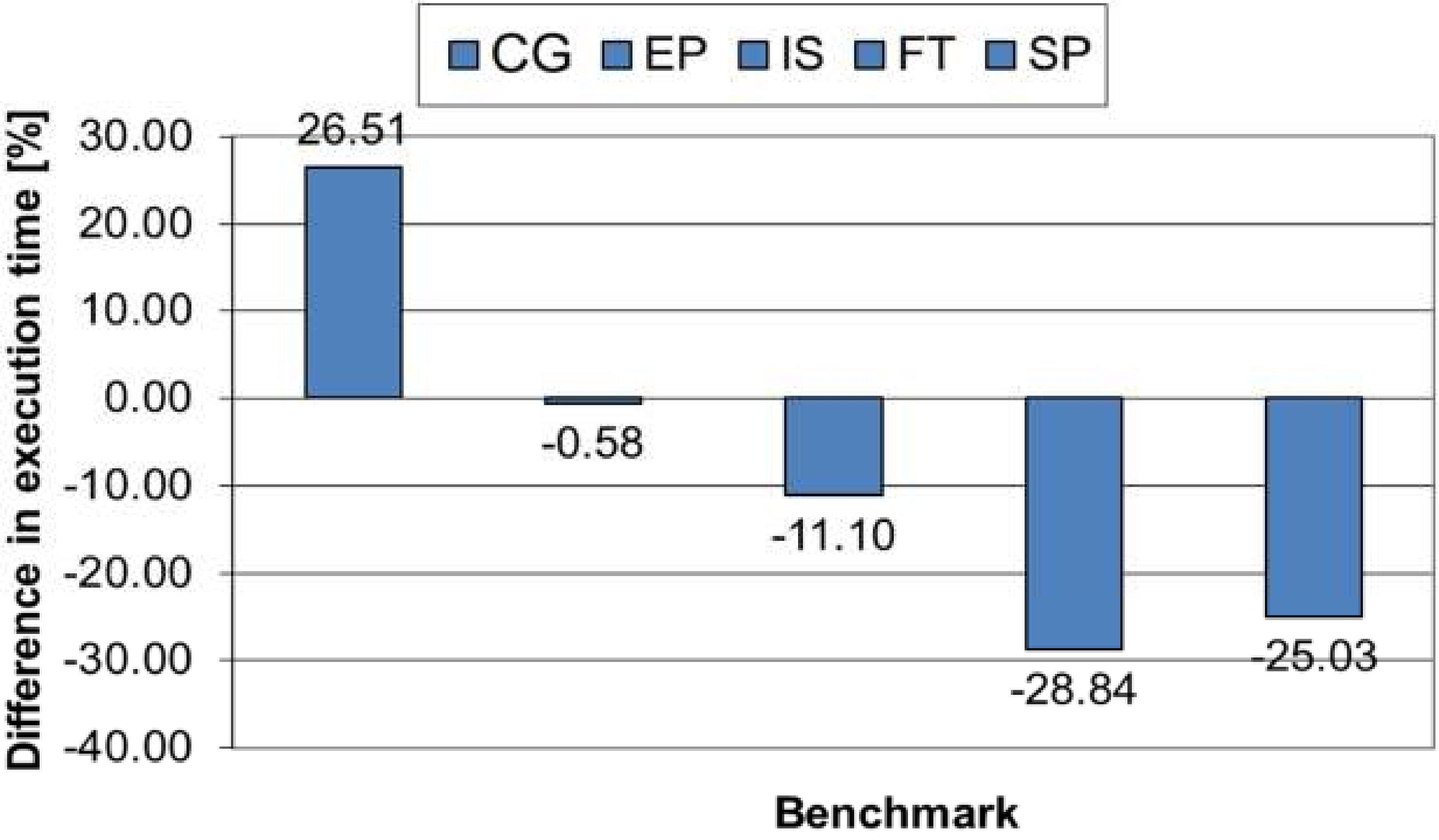
CNC with fixed latency



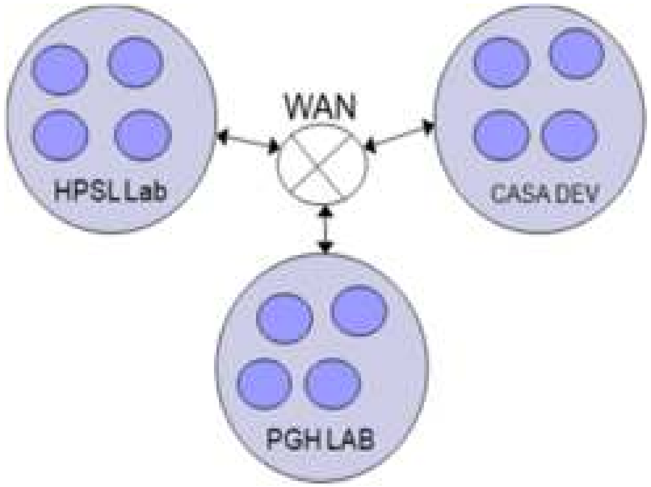
CNC with fixed bandwidth



Validation

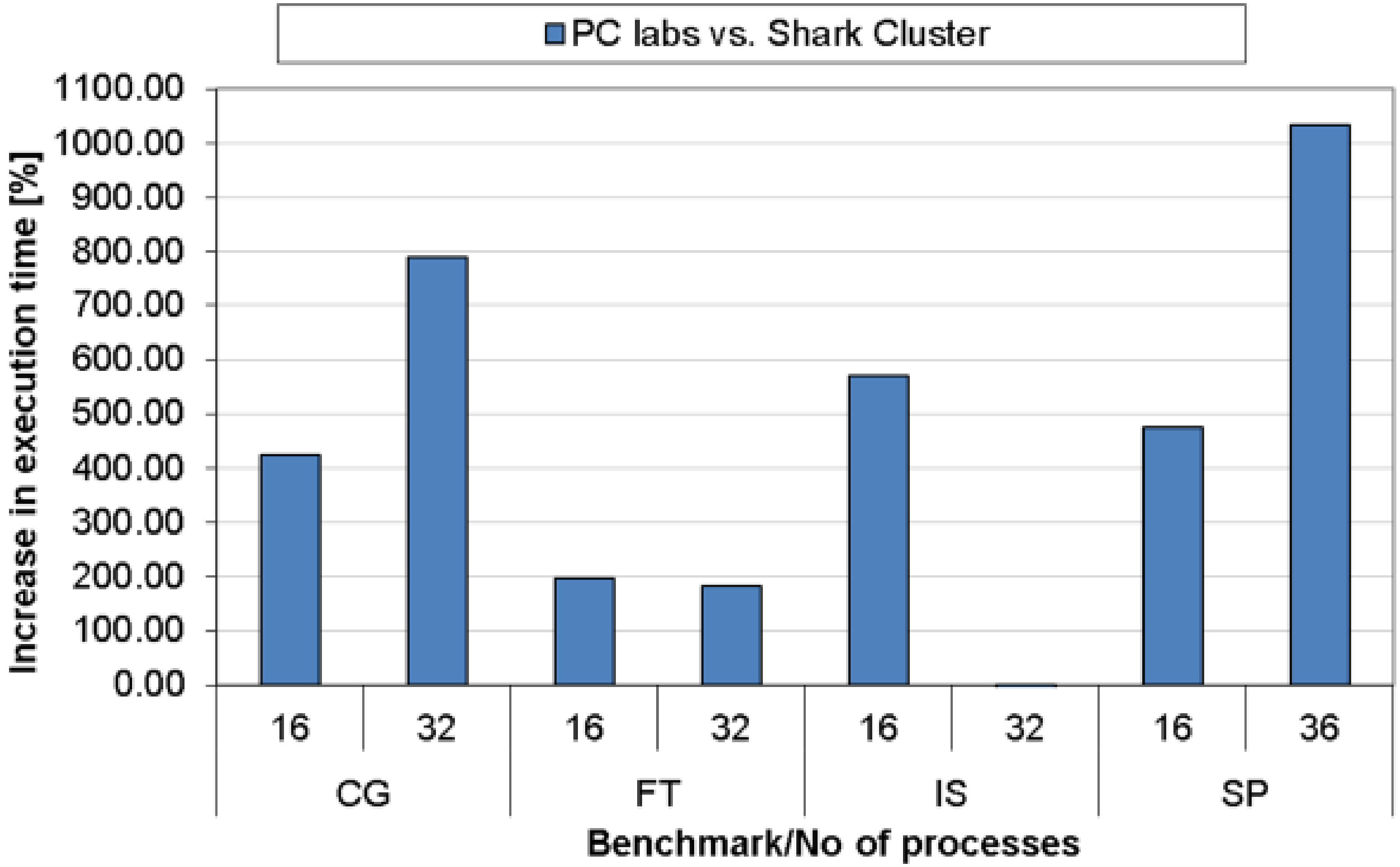

■ CG ■ EP ■ IS ■ FT ■ SP
Difference in execution time [%]
30.00
20.00
10.00
0.00
-10.00
-20.00
-30.00
-40.00
26.51
-0.58
-11.10
-28.84
-25.03
Benchmark

PC Labs vs. Shark Cluster



- Lab internal network parameters:
  - Latency: 0.4 ms, bandwidth: 12 MBps
- External parameters:
  - Latency: 0.8 ms, bandwidth: 7 MBps

PC Labs vs. Shark Cluster



Summary

- Execution time estimation gives application developers an idea for performance
  - University/multi-lab style settings with reasonable overhead
  - Home systems settings with high overhead
- Higher sensitivity to bandwidth vs. latency for the applications analyzed