PER-MARE: Adaptive Deployment of MapReduce over Pervasive Grids

L. A. Steffenel, O. Flauzac, A. Schwertner Charão,
P. Barcelos, B. Stein, S. Nesmachnow,
M. Kirsch Pinheiro and D. Diaz

3PGCIC 2013 Conference
28-30 October 2013 - Compiègne, France

Outline

Motivation

The PER-MARE initiative

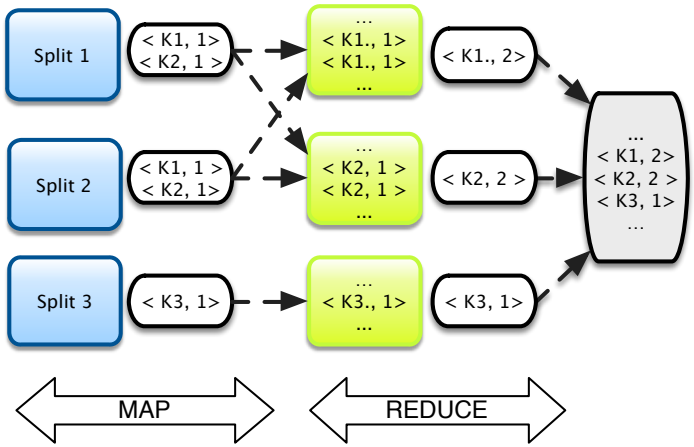
Preliminary results

Discussions & future works

Conclusions

MapReduce

- MapReduce as a computing paradigm
  - Proposed in 2004 by Google
  - Best known by **Hadoop**, an Apache project since 2005
- MapReduce basics
  - Map-Reduce algorithm = job
  - Operates with key-value pairs: (k,V)
  - MR Job defined by 2 functions
    - `map: (k1;v1)→{(k2;v2)}`
    - `reduce: (k2;{v2})→{(k3;v3)}`



Pervasive Grids vs Cloud

- Cloud computing and MapReduce
  - ✓ Elasticity of resources
  - ✗ Sensible data cannot be deployed on the cloud
- Why not use the enterprises own resources to provide computing power?
- Pervasive Grids
  - Computing environment built over available resources
  - "Costless" solution
  - Challenges
    - High volatility of resources
    - High heterogeneity

MapReduce Today

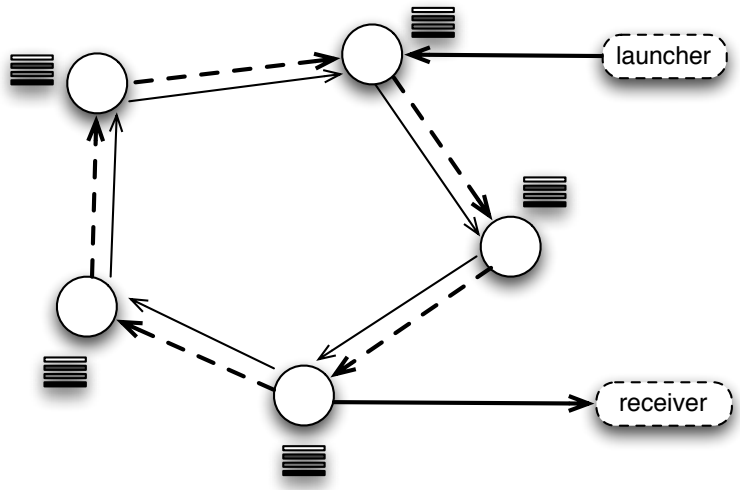
- Hadoop is not tailored for heterogeneous and volatile environments
  - Extensive configuration parameters
  - Limited to disconnections – no join
- A few works tried to port MapReduce to dynamic environments but lack some essential points
  - API compatibility
  - Adaptive scheduling
  - High volatility support

The PER-MARE Vision

- The PER-MARE project aims to adapt MapReduce to pervasive grids under a two-fold approach
  - Improving Hadoop
    - Context-aware task scheduling
    - Automatic tuning of nodes
    - Improved fault tolerance
  - Developing a MapReduce P2P framework
    - Compatible with Hadoop API
    - Easy to deploy in pervasive environments

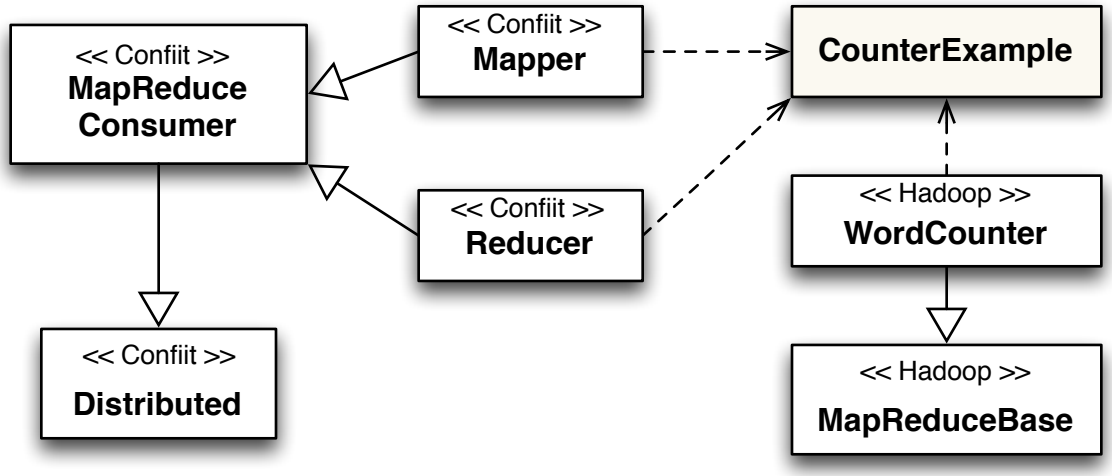
Some Results

- Porting MapReduce to a P2P environment
  - Use of CONFIIT P2P middleware
    - Support to nodes joining/leaving the network
    - "Bag of tasks" scheduling with speculative execution
    - Full replication of results to ensure high availability



Porting MapReduce to CONFIIT

- Two dependent jobs mimicking Hadoop behavior:
  - MAP – as many tasks as input files
  - REDUCE – as many tasks as computing cores
  - Intermediate data exchange (shuffle/sort)
    - Provided by the replication mechanism of CONFIIT
- Basic API compatibility

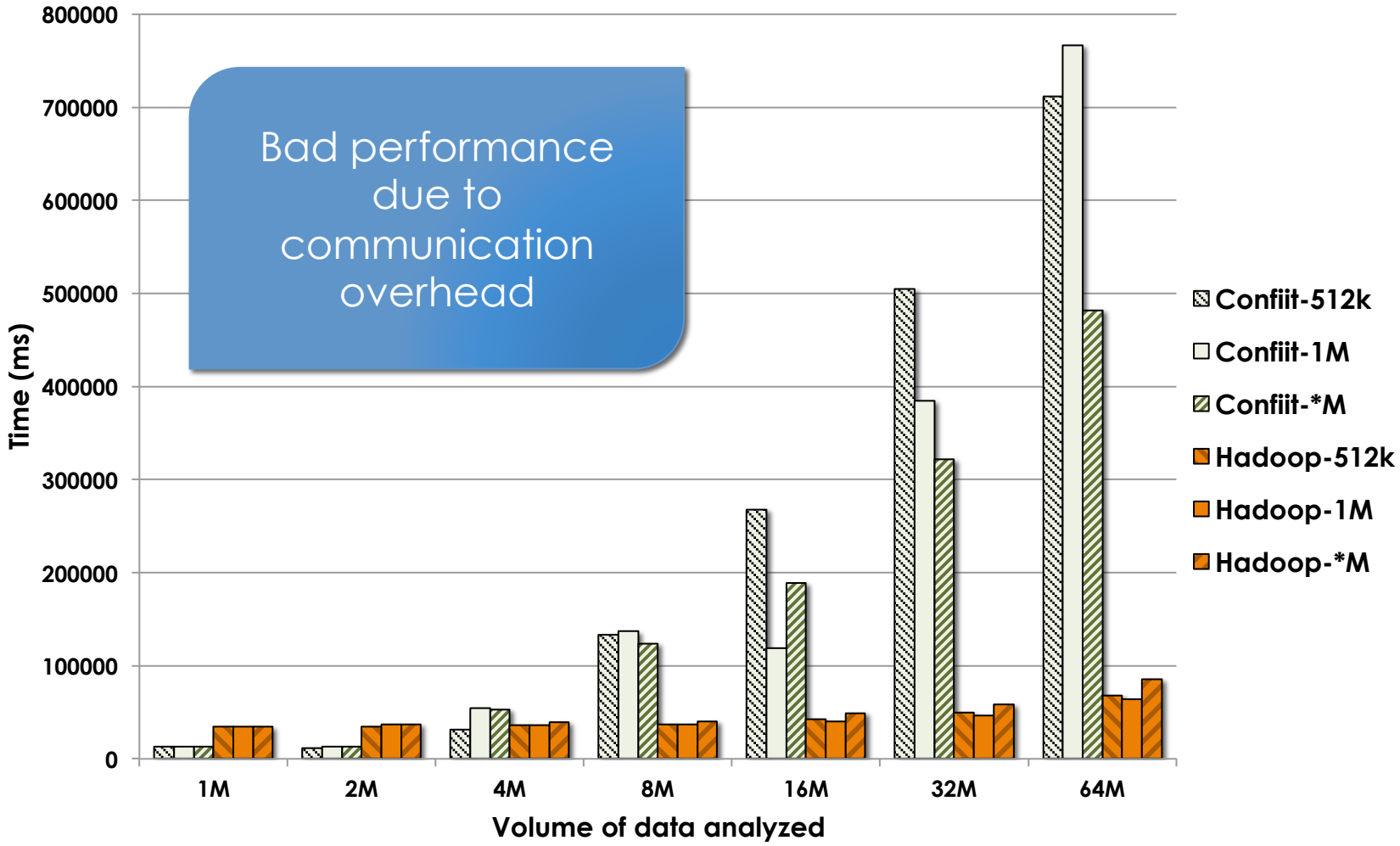


Basic experiment

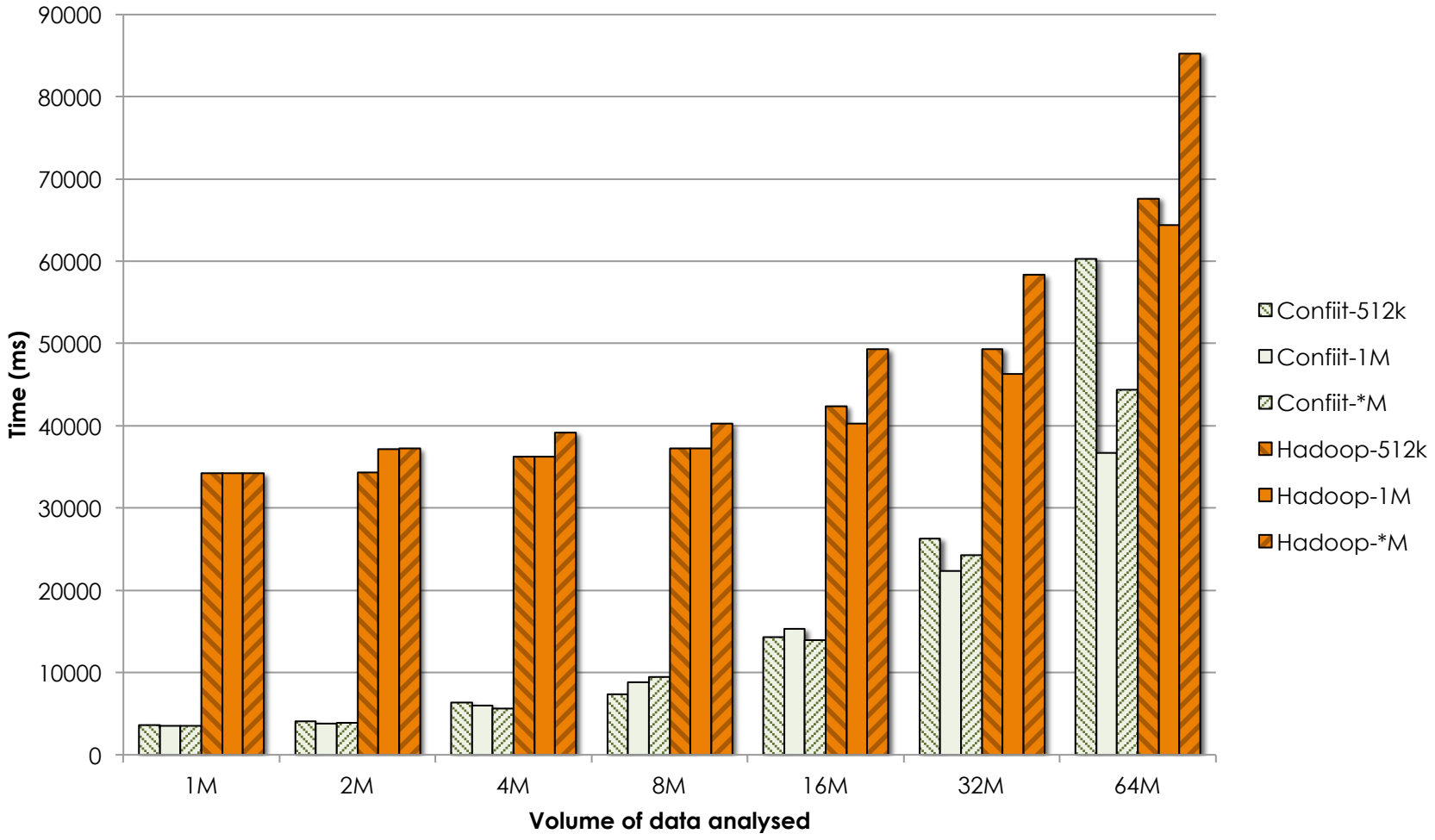
- MapReduce "WordCount"
  - Up to 16 nodes from Grid'5000 testbed
  - Gutenberg Project Science Fiction Bookshelf CD
- Comparison of Hadoop and CONFIIT-MR
  - Different data sizes
  - Different data blocks
    - 512kB, 1MB, full-file

Performance of MR over CONFIIT

(July 2013)



Performance of MR over CONFIIT (September 2013)



Discussions & Future Works

- P2P implementation
  - Explore different replication strategies to improve performance
- Context-awareness
  - Development of a context module
    - Automatic tuning of Hadoop nodes
    - Context-aware task scheduling for Hadoop and P2P
    - Improve data locality for the P2P implementation
- Fault tolerance on Hadoop
  - Hadoop 1.x and 2.x daemon architecture
    - Reinsertion of nodes

Conclusions

- MapReduce is one of the leading paradigms for BigData but is also a good paradigm for generic distributed computations (HPC, etc.)
- The PER-MARE initiative
  - Porting Hadoop to dynamic environments
    - Context-awareness is a key element
  - Implementing MapReduce on a P2P platform
    - Initial experiments offer encouraging results