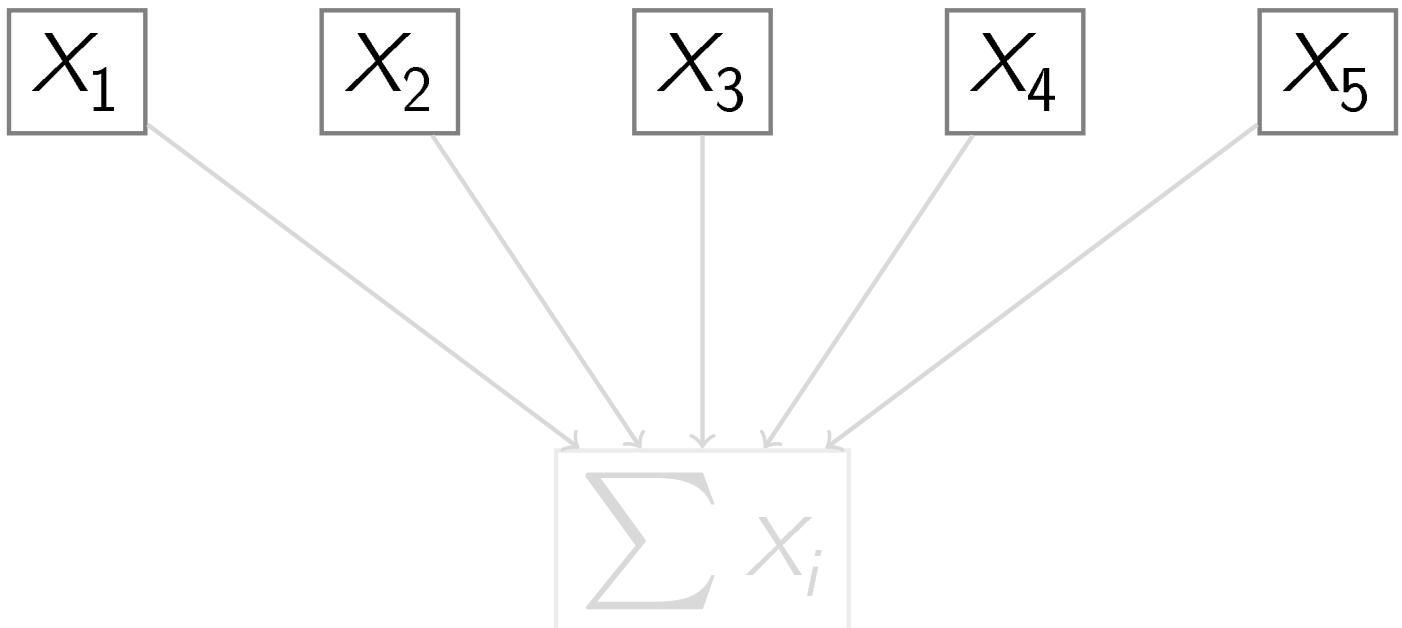
Non-Clairvoyant Reduction Algorithms for Heterogeneous Platforms

Anne Benoit, *Louis-Claude Canon* and Loris Marchal

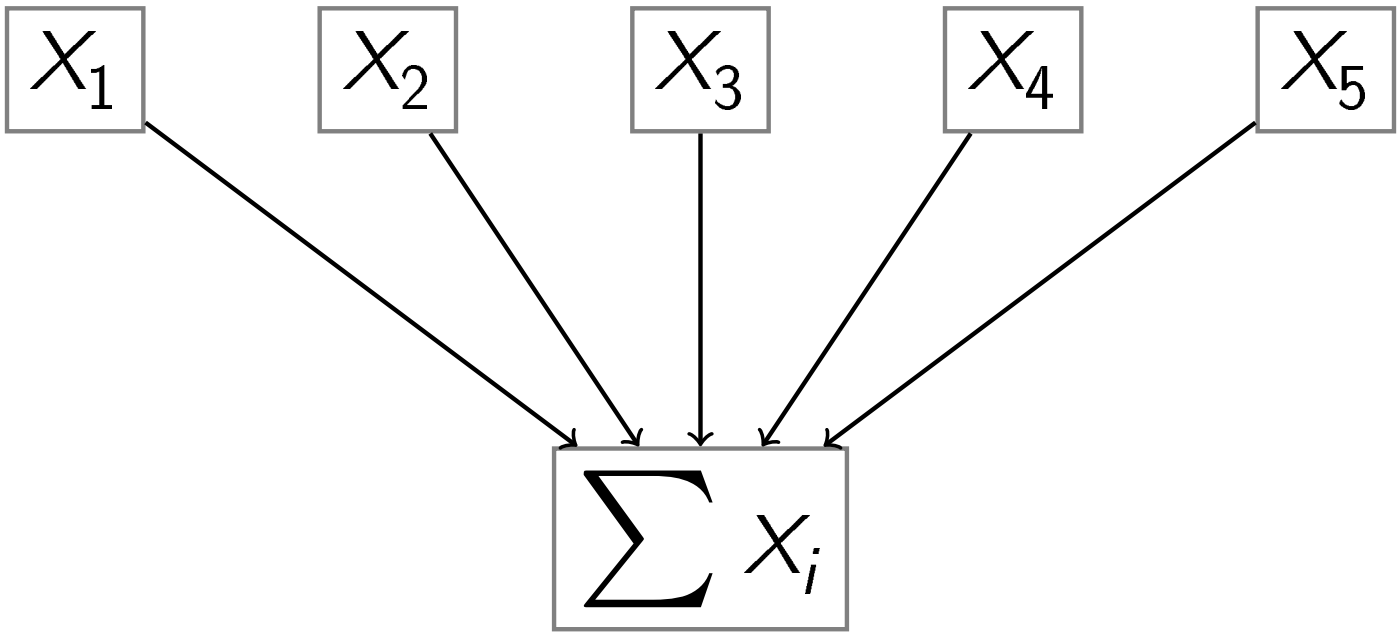
ENS Lyon, Université de Franche-Comté
Roma (LIP), DISC (FEMTO-ST)

August 26, 2013

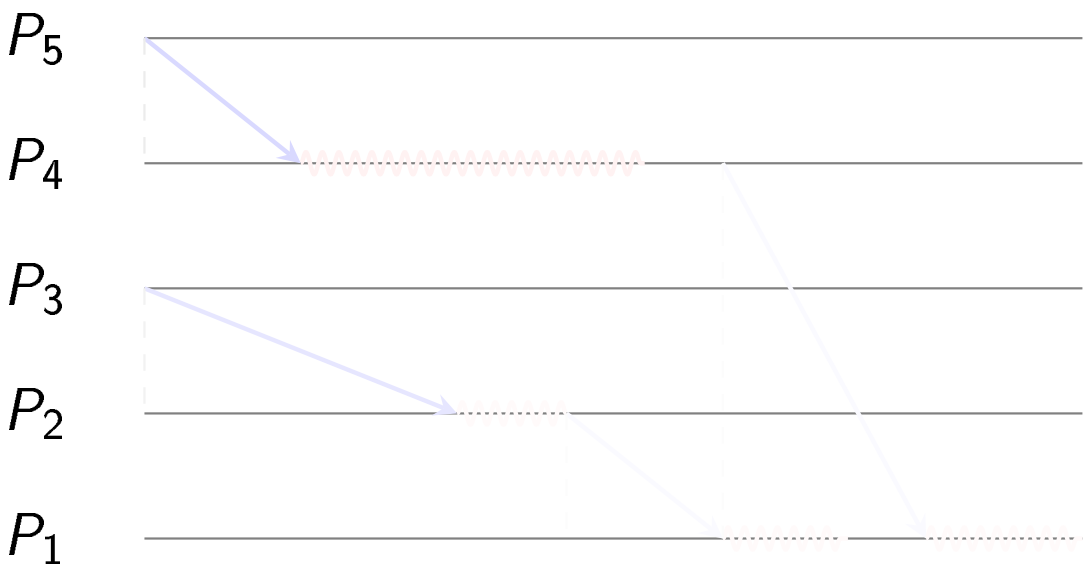
Reduction Example



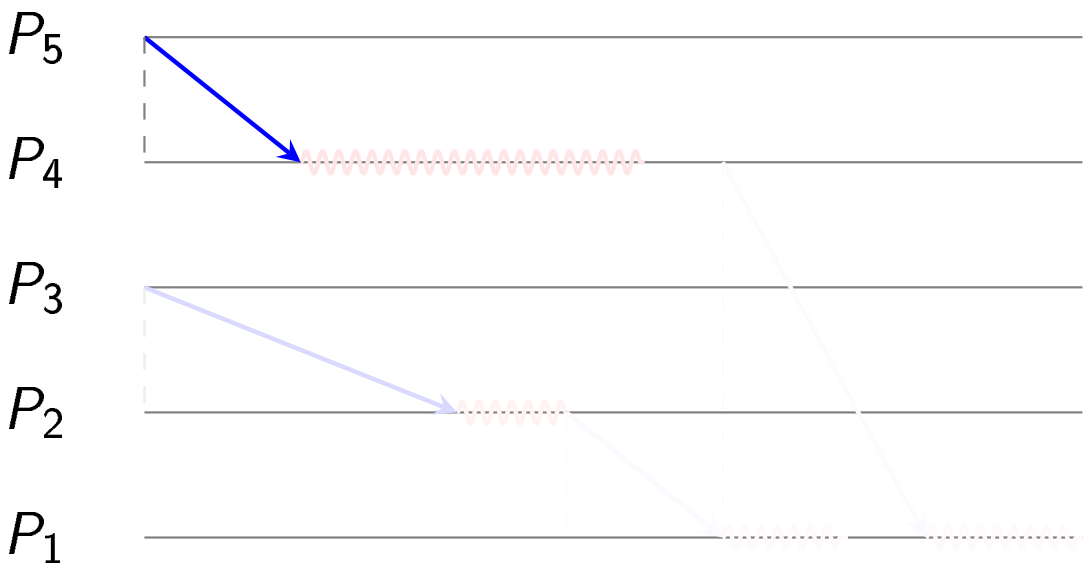
Reduction Example



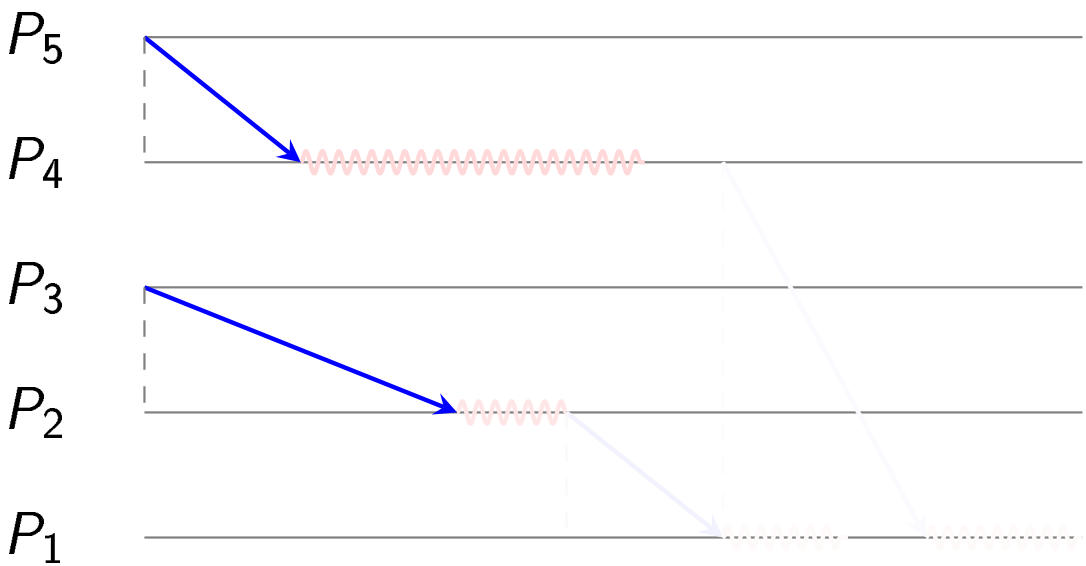
Execution Example



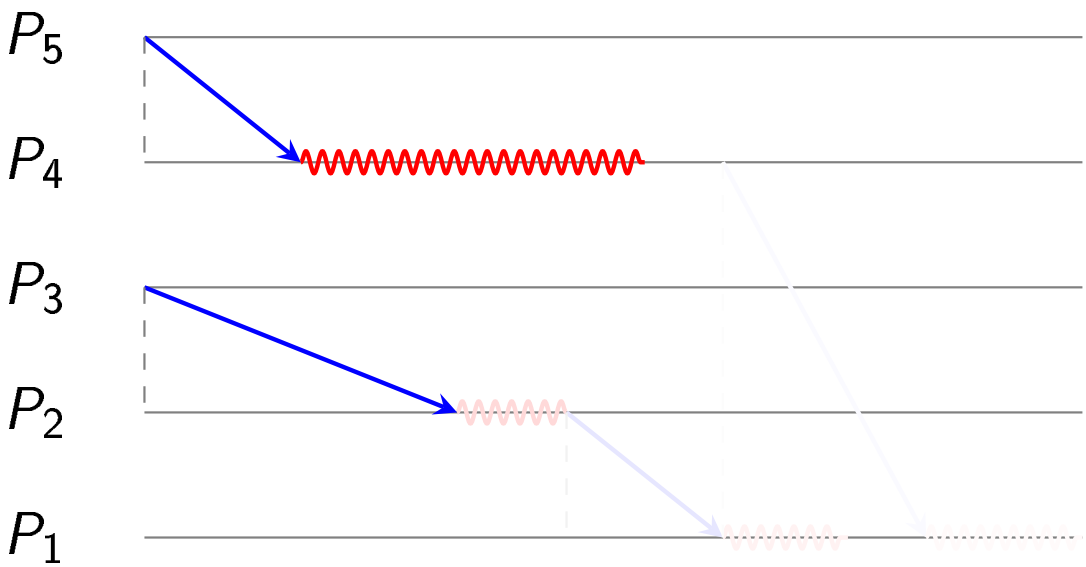
Execution Example



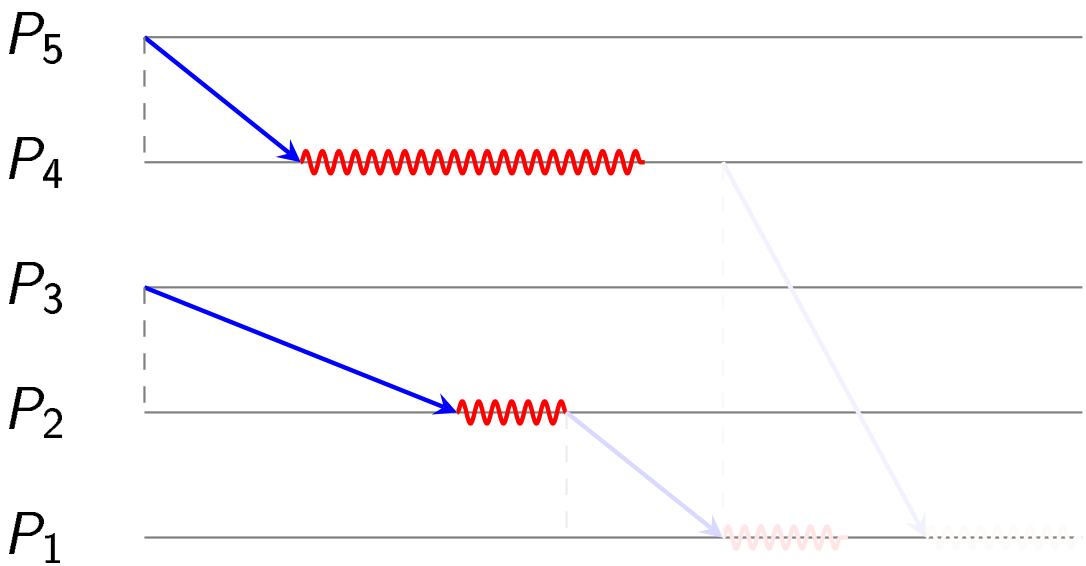
Execution Example



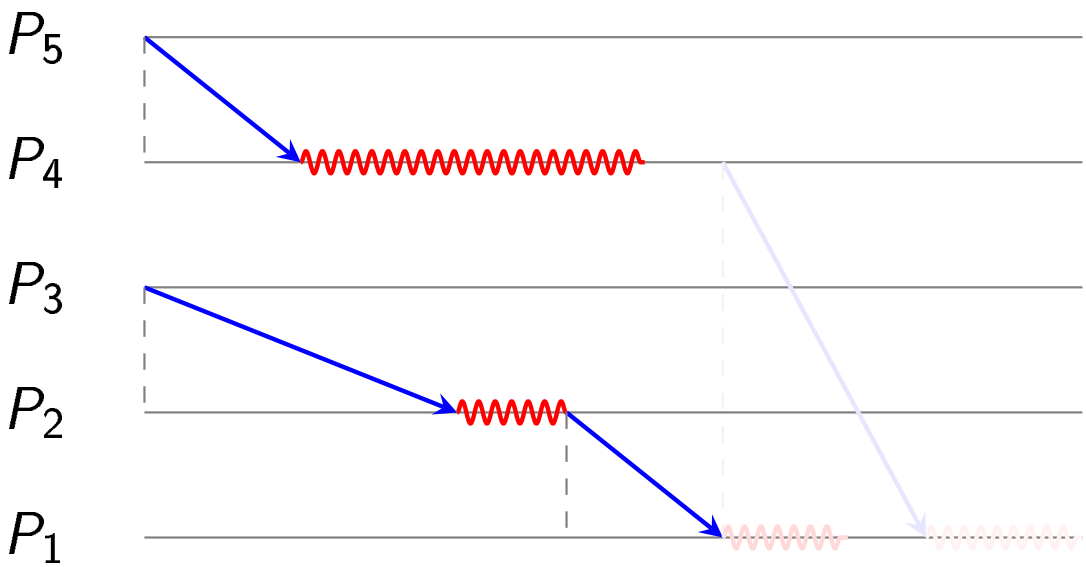
Execution Example



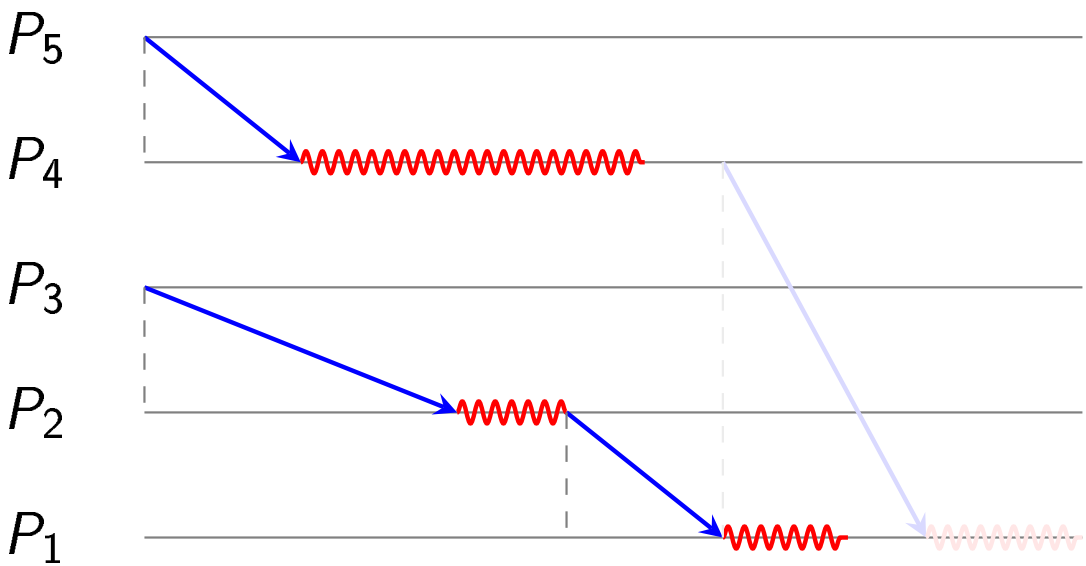
Execution Example



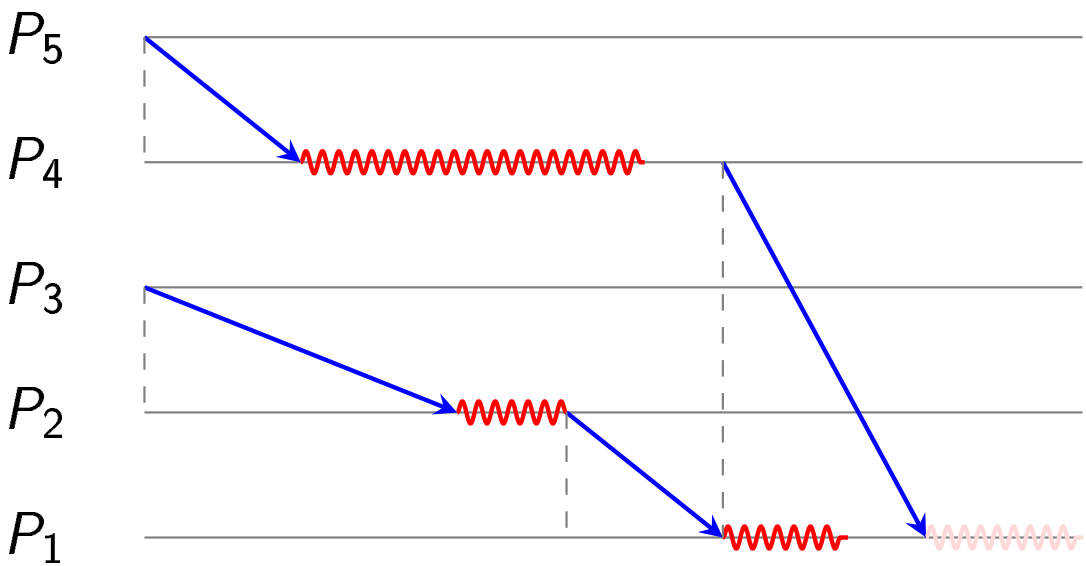
Execution Example



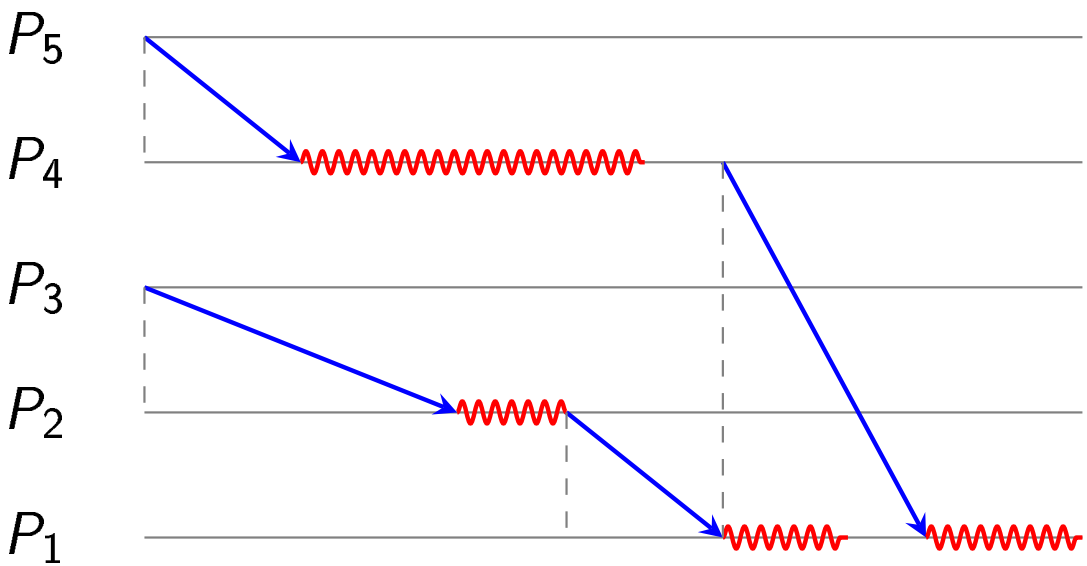
Execution Example



Execution Example



Execution Example



Motivation

What are the performance of existing algorithms when transfers and computations may overlap and when costs are unkown?

Outline

Outline

Objective

Reduce efficiently n elements that are available simultaneously (each on a distinct processor).

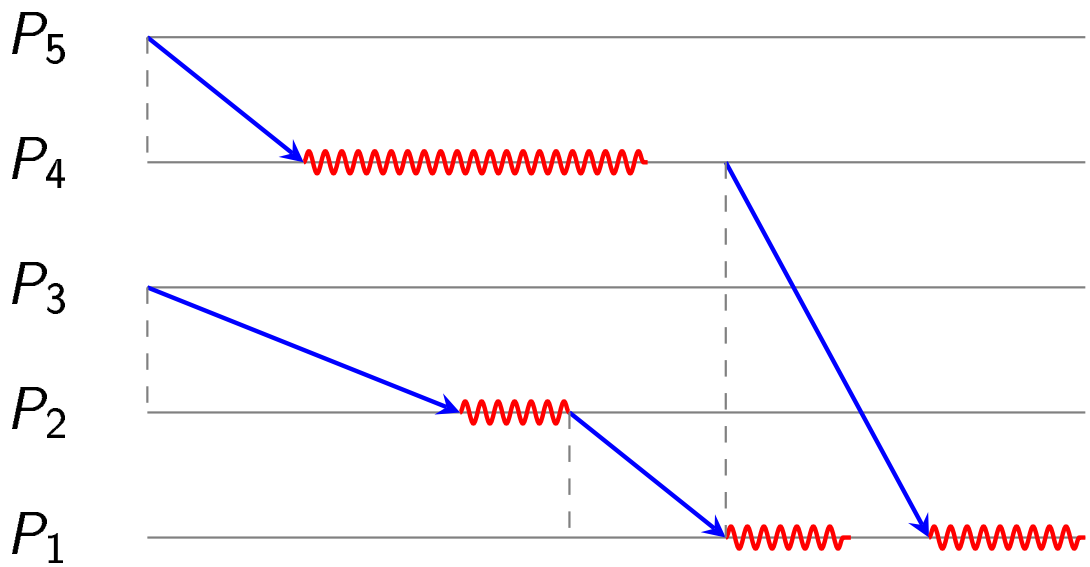
Execution Model

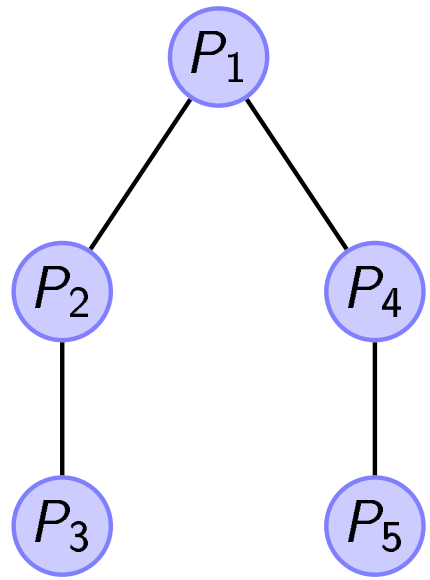
- The reduction operation is associative.
- Each processor can perform a reduction and any pair of processors can communicate.
- Any processor is involved in at most one transfer at any time.
- Computation and transfer can overlap.
- Costs are heterogeneous and the system is non-clairvoyant ($d_{i,j}$ for the communication cost between processor i and j and c_i for the computation cost of processor i).

Scheduling

- Each processor reduces all its received elements with its own pairwise. Then, it sends the result to another processor.
- Static or dynamic plan.
- The communication structure can be represented by a *spanning tree*.

Spanning Tree Example





Related Work

MPI Context

Rabenseifner et al. Propose the butterfly algorithm for array reductions (Computational Science, 2004, Lecture Notes in Computer Science, 2004 and IJHPCA, 2005).

Kielmann et al. Strategy for hierarchical platforms (MPIDC, 1999).

Bosilca et al. Empirical and analytical comparison of several existing algorithms (Cluster Computing, 2007).

MapReduce Context

Agarwal et al. Implementation of AllReduce using a spanning tree (submitted).

Sandia National Laboratories MapReduce–MPI implements MapReduce on top of MPI.

Hoefler, Dongarra et al. Highlight the benefits of MPI-3 features for MapReduce applications (EuroPVM/MPI 2009).

Algorithms Overview

Binomial-stat and **Fibonacci-stat** strategies[1]

Static approaches, optimal with specific computation and communication costs.

Dynamic strategies

Tree-dyn Any communication starts as soon as possible.

Non-Commut-Tree-dyn Adaptation for non-commutative operations.

[1] Canon, 2013

Binomial-stat

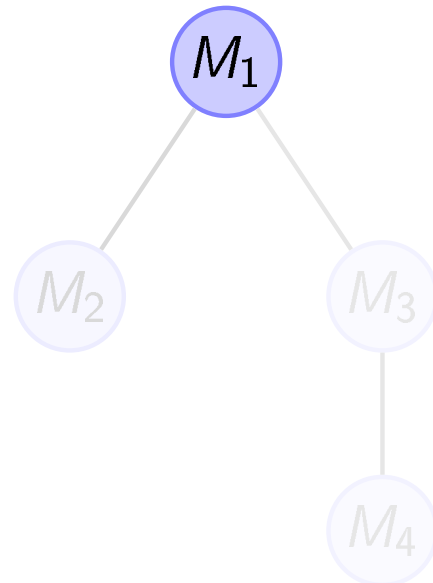
Definition

- A binomial tree of order 0 is a single node.
- A binomial tree of order k consists of a binomial tree of order $k-1$ whose sink is the parent of the sink of another binomial tree of order $k-1$.

Binomial-stat

Definition

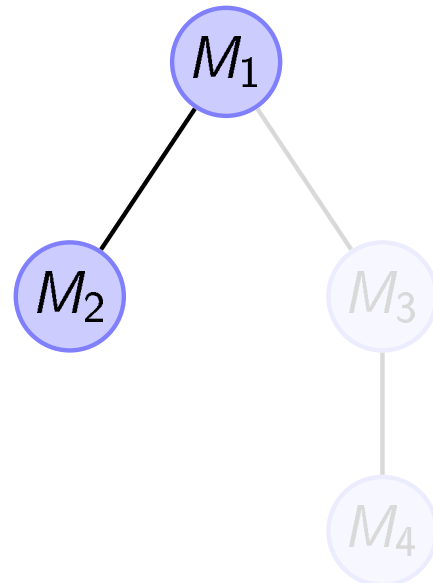
- A binomial tree of order 0 is a single node.
- A binomial tree of order k consists of a binomial tree of order $k-1$ whose sink is the parent of the sink of another binomial tree of order $k-1$.



Binomial-stat

Definition

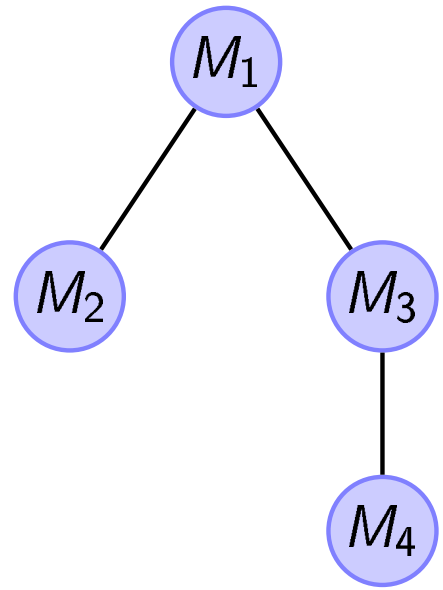
- A binomial tree of order 0 is a single node.
- A binomial tree of order k consists of a binomial tree of order $k-1$ whose sink is the parent of the sink of another binomial tree of order $k-1$.



Binomial-stat

Definition

- A binomial tree of order 0 is a single node.
- A binomial tree of order k consists of a binomial tree of order $k-1$ whose sink is the parent of the sink of another binomial tree of order $k-1$.

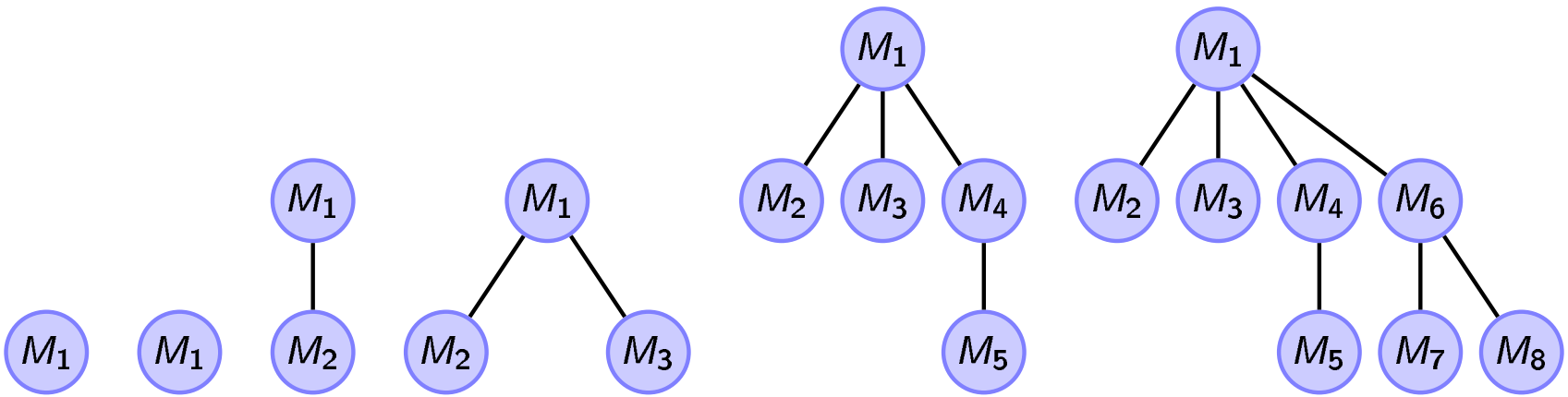


Fibonacci-stat

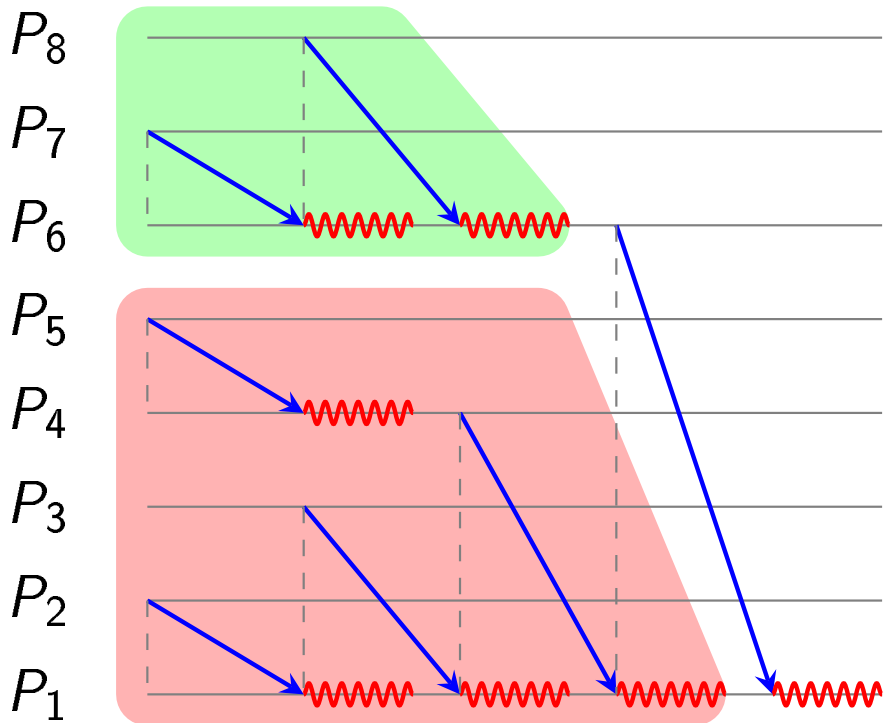
Definition

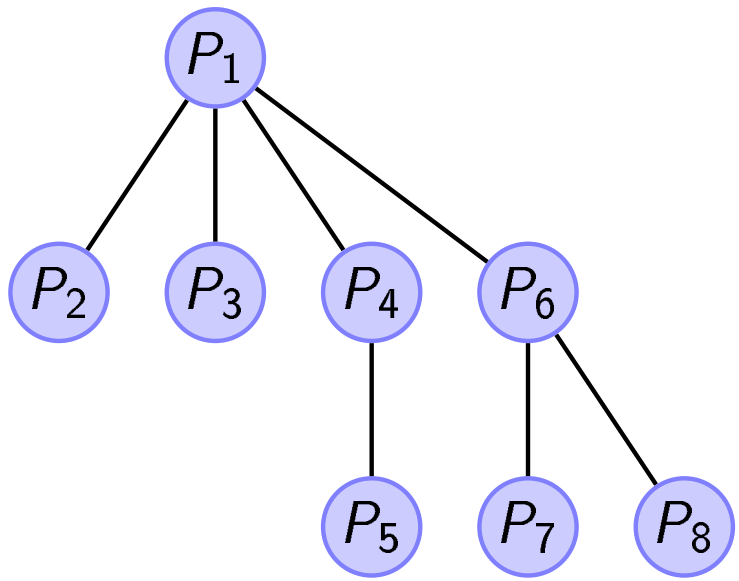
- A k-ary Fibonacci tree of order 0 or 1 is a single node.
- A k-ary Fibonacci tree of order k consists of a k-ary Fibonacci tree of order $k-2$ whose sink is the parent of the sink of another k-ary Fibonacci tree of order $k-1$.

Examples of k-ary Fibonacci Tree



Execution Example with **Fibonacci-stat**





Dynamic Approaches

Tree-dyn

Any processor becoming idle (initially or after a reduction) sends its element to any other idle processor. If there is none, it waits.

Non-Commut-Tree-dyn

In addition to being idle, the targeted processor with **Tree-dyn** must be a "neighbor" (such that this processor can then perform a reduction with the received element).

Outline

Summary

	$c = 0$	any c	$c = d$
Binomial-stat	Δ	$\Delta + 1$	$(\Delta + 1)(1 + \frac{1}{\log_2 n}) \log_2 \varphi$
Tree-dyn	Δ	$\Delta + 1$	$(\Delta + 1)(1 + \frac{1}{\log_2 n}) \log_2 \varphi$
Fibonacci-stat	$\frac{\Delta}{\log_2 \varphi} + \frac{\Delta}{\lceil \log_2 n \rceil}$	$\frac{\Delta}{\log_2 \varphi} + \frac{2\Delta}{\lceil \log_2 n \rceil}$	Δ

with $\Delta = \frac{\max_{i,j} d_{i,j}}{\min_{i,j} d_{i,j}}$, $\log_2 \varphi \approx 0.69$ and $\frac{1}{\log_2 \varphi} \approx 1.44$.

Perspectives

- Improve or simplify the ratios.
- Analyse **Non-Commut-Tree-dyn**.

Outline

Tree-dyn Analysis

Method

- Model the algorithm as a memory-less process.
- Apply the first-step analysis.

Results with no computation

Average completion time: $\frac{1}{\lambda_d}\left(2H\left(\frac{n}{2}-1\right)+\frac{2}{n}\right)$ where $H(n)$ is the nth harmonic number and λ_d is the rate of the communication cost.

Variance of the completion time: $\frac{1}{\lambda_d^2}\left(2\sum_{i=1}^{\frac{n}{2}-1}\frac{1}{i^2}+\frac{4}{n^2}\right)$.

Drawback

Only **Tree-dyn** could be completely analysed.

Tree-dyn Analysis

Method

- Model the algorithm as a memory-less process.
- Apply the first-step analysis.

Results with no computation

Average completion time: $\frac{1}{\lambda_d}\left(2H\left(\frac{n}{2}-1\right)+\frac{2}{n}\right)$ where $H(n)$ is the nth harmonic number and λ_d is the rate of the communication cost.

Variance of the completion time: $\frac{1}{\lambda_d^2}\left(2\sum_{i=1}^{\frac{n}{2}-1}\frac{1}{i^2}+\frac{4}{n^2}\right)$.

Drawback

Only **Tree-dyn** could be completely analysed.

Tree-dyn Analysis

Method

- Model the algorithm as a memory-less process.
- Apply the first-step analysis.

Results with no computation

Average completion time: $\frac{1}{\lambda_d}\left(2H\left(\frac{n}{2}-1\right)+\frac{2}{n}\right)$ where $H(n)$ is the nth harmonic number and λ_d is the rate of the communication cost.
Variance of the completion time: $\frac{1}{\lambda_d^2}\left(2\sum_{i=1}^{\frac{n}{2}-1}\frac{1}{i^2}+\frac{4}{n^2}\right)$.

Drawback

Only **Tree-dyn** could be completely analysed.

Outline

Cost Model

- Use of mixture of gamma distributions (with 6 parameters) for modelling runtimes[2].
- Trade-off between the precision of the model and the number of parameters (risk of overfitting and intractability).
- Gamma distribution is one of the most relevant with 2 parameters (positive and adjustable mean and variance).

[2] Lublin and Feitelson, 2003

MC Simulation

- Communication (resp., computation) costs follow a gamma distribution with mean μ_d (resp., μ_c) and standard deviation σ_d (resp., σ_c).
- One Monte Carlo (MC) simulation consists in instantiating all the costs for a given number of elements n and to run an reduction algorithm with these costs.

XP Design

Summary:

- 5 parameters: n, μ_d, μ_c, σ_d and σ_c.
- 4 methods: **Binomial-stat**, **Fibonacci-stat**, **Tree-dyn** and **Non-Commut-Tree-dyn**.

Three plans:

1. Test all methods with no computation cost and different Coefficients of Variation (CV) for the communication ($n = 64$, $\mu_d = 1$, 1 000 000 MC simulations).
2. Test the improvement of **Tree-dyn** over **Fibonacci-stat** with varying overlapping (i.e., $\frac{c}{d}$) and different CV ($\frac{\sigma_c}{\mu_c} = \frac{\sigma_d}{\mu_d}$).
3. Determine the best method among those that support a non-commutative operation (same settings as previously).

XP Design

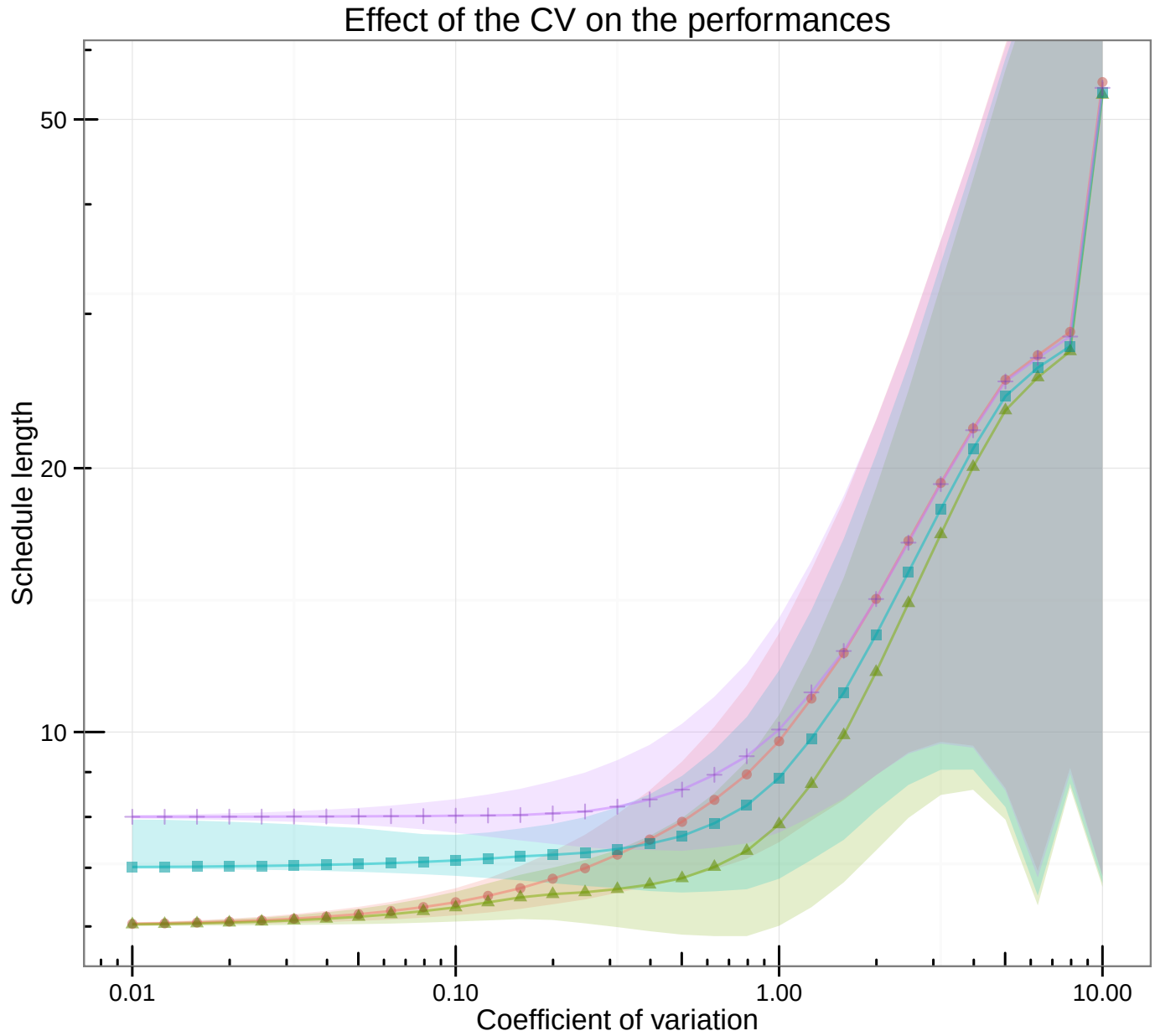
Summary:

- 5 parameters: n, μ_d, μ_c, σ_d and σ_c.
- 4 methods: **Binomial-stat**, **Fibonacci-stat**, **Tree-dyn** and **Non-Commut-Tree-dyn**.

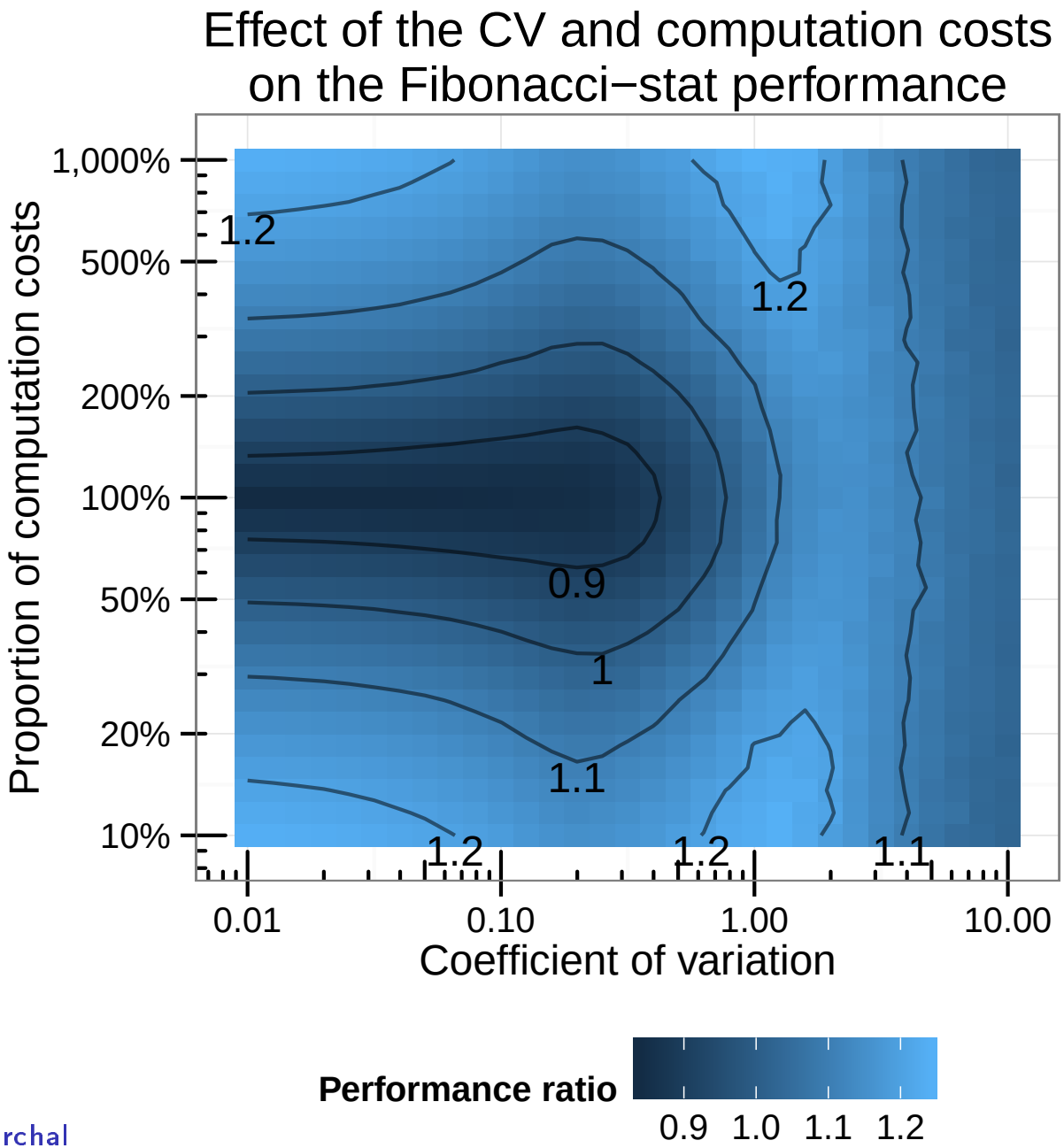
Three plans:

1. Test all methods with no computation cost and different Coefficients of Variation (CV) for the communication ($n = 64$, $\mu_d = 1$, 1 000 000 MC simulations).
2. Test the improvement of **Tree-dyn** over **Fibonacci-stat** with varying overlapping (i.e., $\frac{c}{d}$) and different CV ($\frac{\sigma_c}{\mu_c} = \frac{\sigma_d}{\mu_d}$).
3. Determine the best method among those that support a non-commutative operation (same settings as previously).

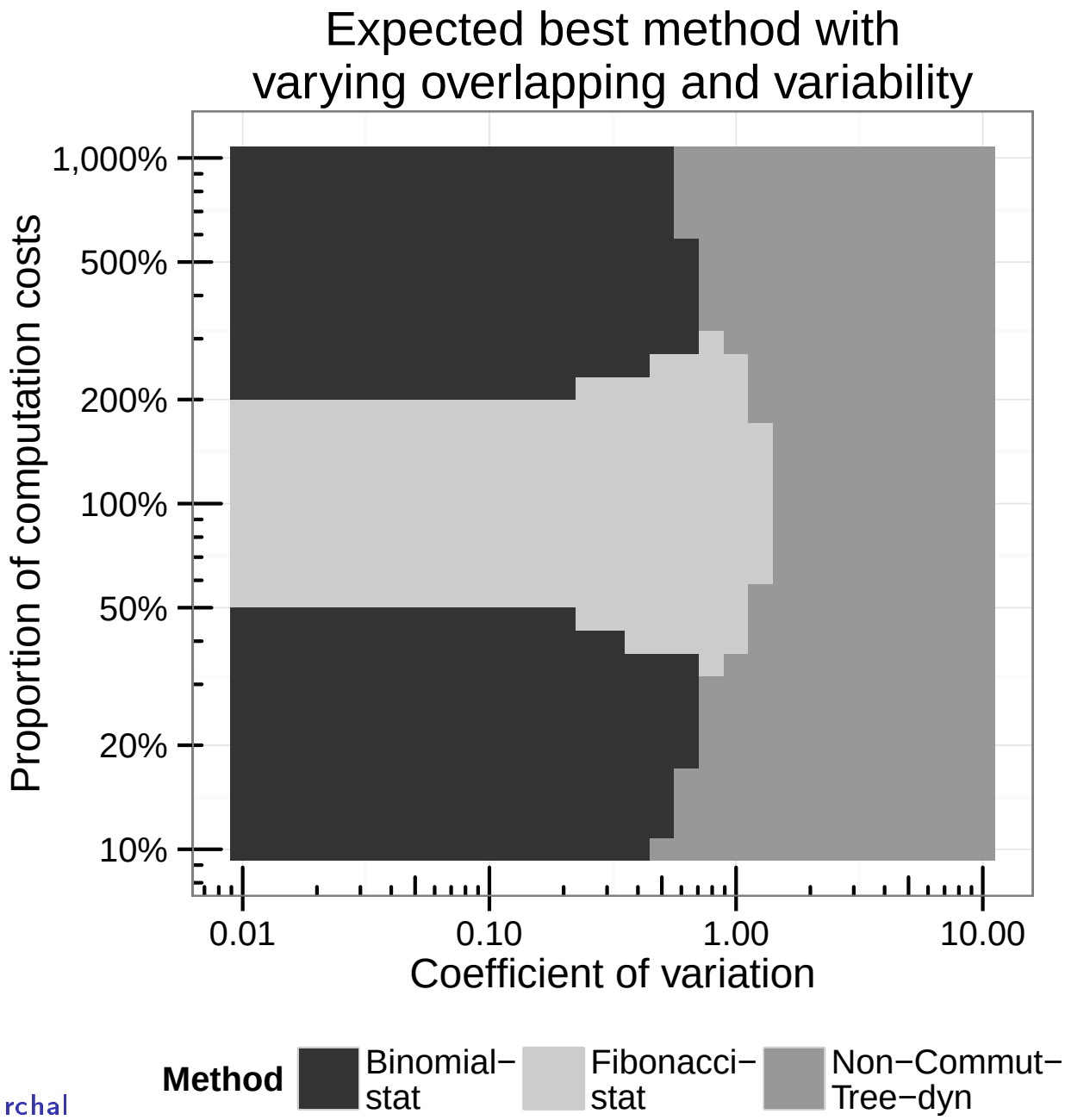
Cost Dispersion Effect



Non-Negligible Computation

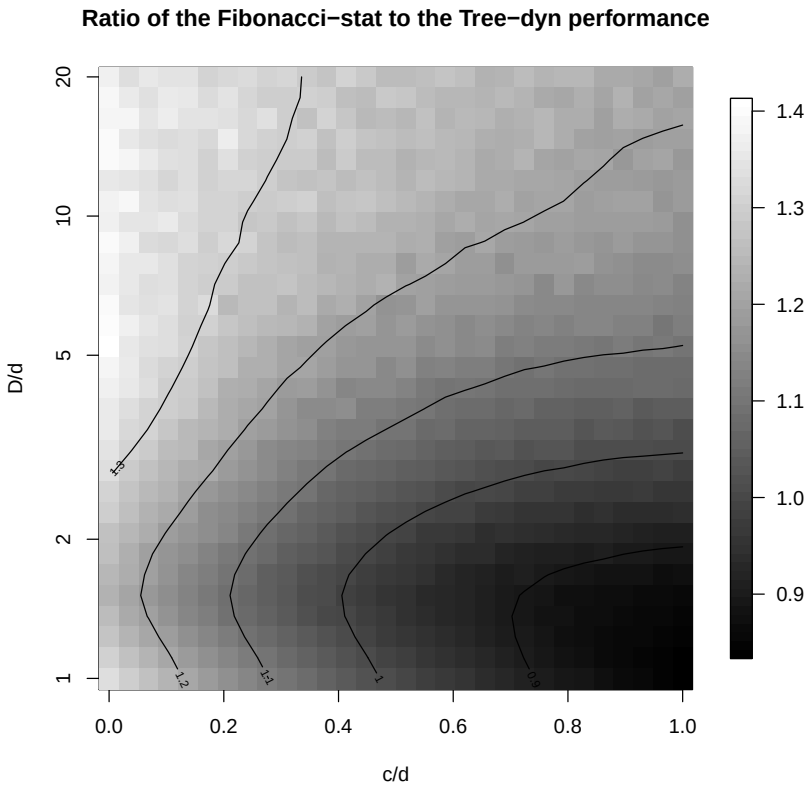


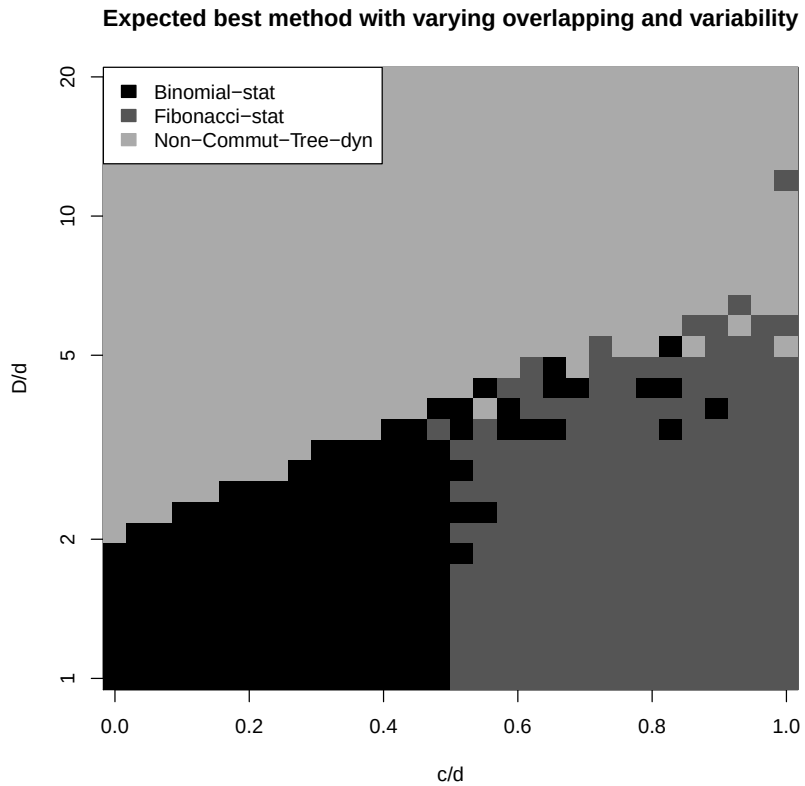
Non-Commutative Operation



Analysis Robustness

Consistent conclusions with a Bernoulli distribution.





Outline

Conclusion and Future Directions

Conclusion

- Reduction of several distributed elements (MPI, MapReduce).
- Analysis of four strategies (two static and two dynamic).
- Three different analysis: worst-case, Markov chain and probabilistic.

Future Directions

- Model extensions (arrival dates, network topologies, array reductions).
- Introduce some limited clairvoyance (such as uncertain predictions).

Conclusion and Future Directions

Conclusion

- Reduction of several distributed elements (MPI, MapReduce).
- Analysis of four strategies (two static and two dynamic).
- Three different analysis: worst-case, Markov chain and probabilistic.

Future Directions

- Model extensions (arrival dates, network topologies, array reductions).
- Introduce some limited clairvoyance (such as uncertain predictions).