An Analysis of Partitioning Strategies for the Data Dependency Graphs of k-Means

Francesc Gispert

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

17th September, 2014

Outline

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

RoMoL Project

Outline

Motivation

- Scheduling and task mapping have a huge impact on parallel applications, due to:
  - load balance of cores,
  - communication overhead.
- Hence, a careful placement of tasks on computing nodes may increase concurrency.

Objectives

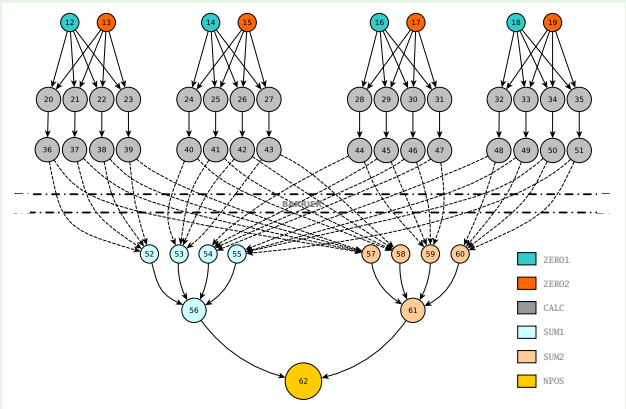
- An execution of a parallel application has an associated data dependency graph.
- This graph can be partitioned in order to facilitate the task mapping.
- Several graph partitioning techniques have been applied to data dependency graphs from k-means.
- The main objective is to compare different approaches and prove the possible improvements.

Outline

How k-Means Works

- k-means aims to partition n points into k clusters such that:
  - each point belongs to the cluster with nearest mean.

Example



Subgraph corresponding to an iteration of k-means.

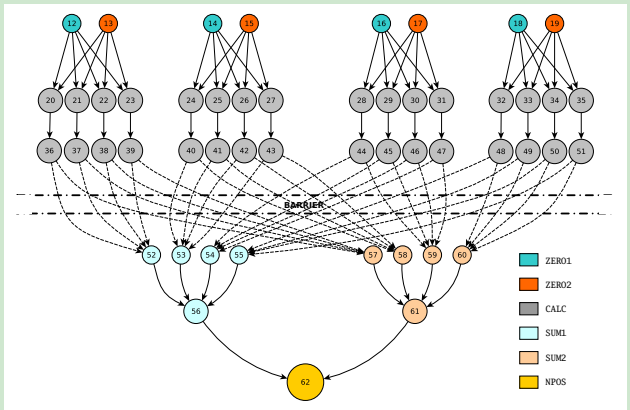
k-means algorithm

- k initial centers.
- Several iterations:
  - the closest center to each point is determined,
  - centers are updated.

How k-Means Works

- k-means aims to partition n points into k clusters such that:
  - each point belongs to the cluster with nearest mean.

Example



Subgraph corresponding to an iteration of k-means.

k-means algorithm

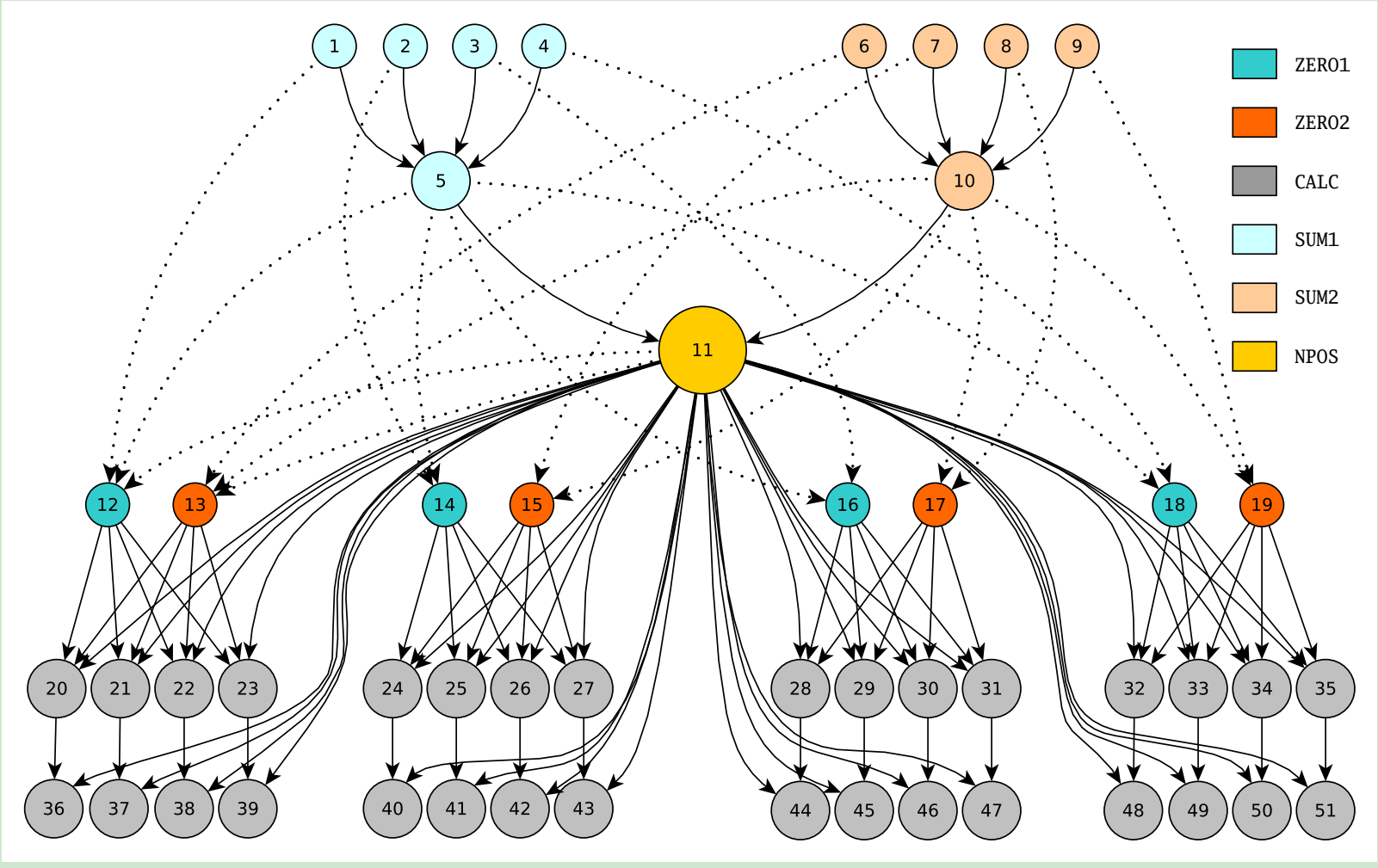
- k initial centers.
- Several iterations:
  - the closest center to each point is determined,
  - centers are updated.

Structure of k-Means

- Barriers divide the graph in layers with the same structure.
- In the OmpSs implementation, there are 4 kinds of tasks:
  1. auxiliary vectors are (re)set to zero;
  2. the closest center to each point is calculated (partial results stored in auxiliary vectors);
  3. partial results are unified using tree sums;
  4. new positions for centers (means) are computed.

Sample Subgraph between Barriers

Example



Subgraph between two consecutive barriers.

Subgraph Partitioning Techniques

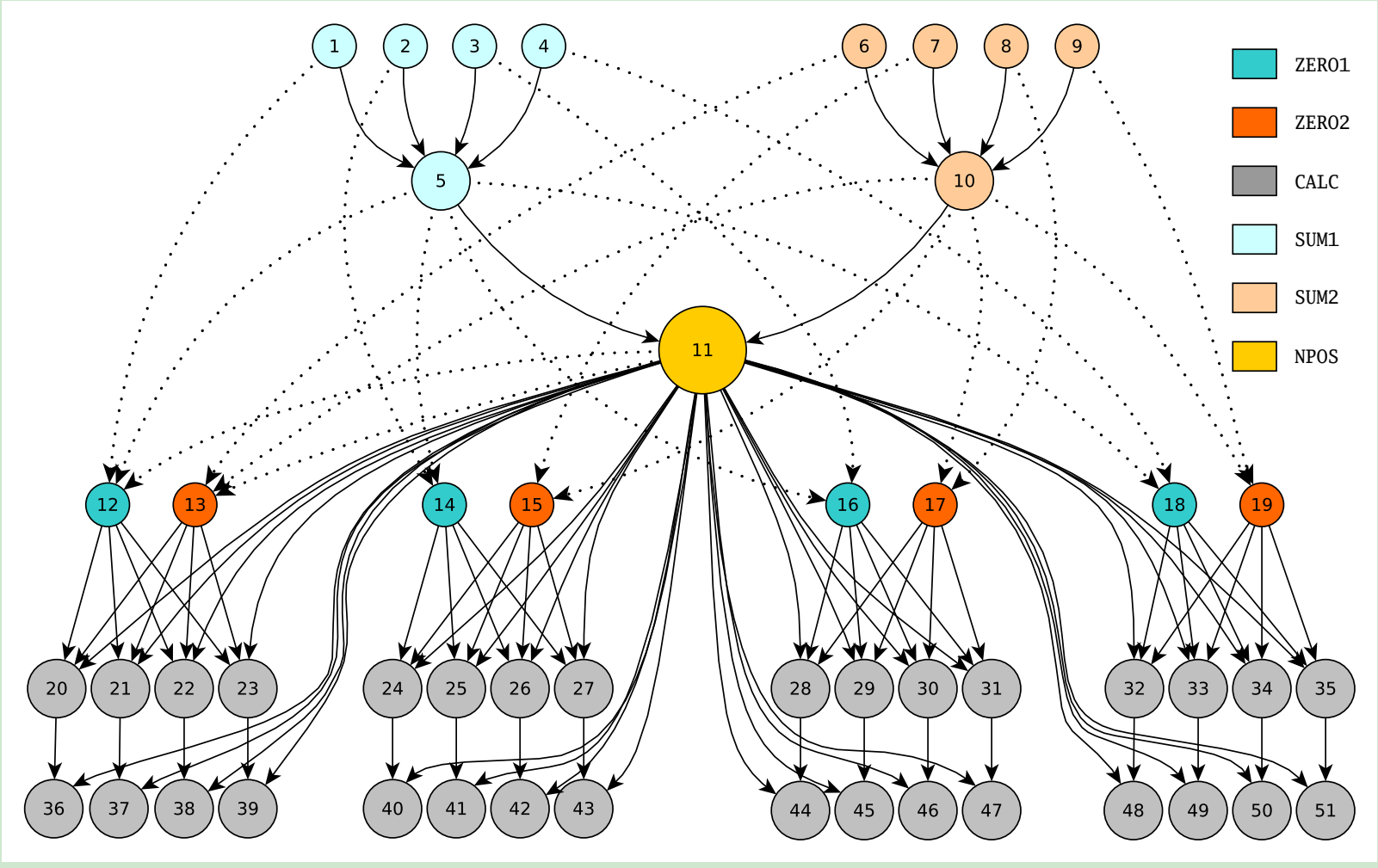
1. **METIS Partitioning (MP)** makes use of the METIS library.
   - Balanced partitions and small edge-cut.
2. **Greedy Partitioning (GP)** exploits the regularity of the studied graphs.
   - Balanced partitions and small (close to optimal) edge-cut.
3. **Random Partitioning (RP)** assigns each node to a part randomly.
   - Balanced partitions but edge-cut is not considered.
4. **Ordered Partitioning (OP)** orders the nodes and assigns them to the parts in a round-robin fashion.
   - Balanced partitions but edge-cut is not considered.

Subgraph Partitioning Techniques

1. **METIS Partitioning (MP)** makes use of the METIS library.
   - Balanced partitions and small edge-cut.
2. **Greedy Partitioning (GP)** exploits the regularity of the studied graphs.
   - Balanced partitions and small (close to optimal) edge-cut.
3. **Random Partitioning (RP)** assigns each node to a part randomly.
   - Balanced partitions but edge-cut is not considered.
4. **Ordered Partitioning (OP)** orders the nodes and assigns them to the parts in a round-robin fashion.
   - Balanced partitions but edge-cut is not considered.

Sample Subgraph between Barriers

Example



Subgraph between two consecutive barriers.

Subgraph Partitioning Techniques

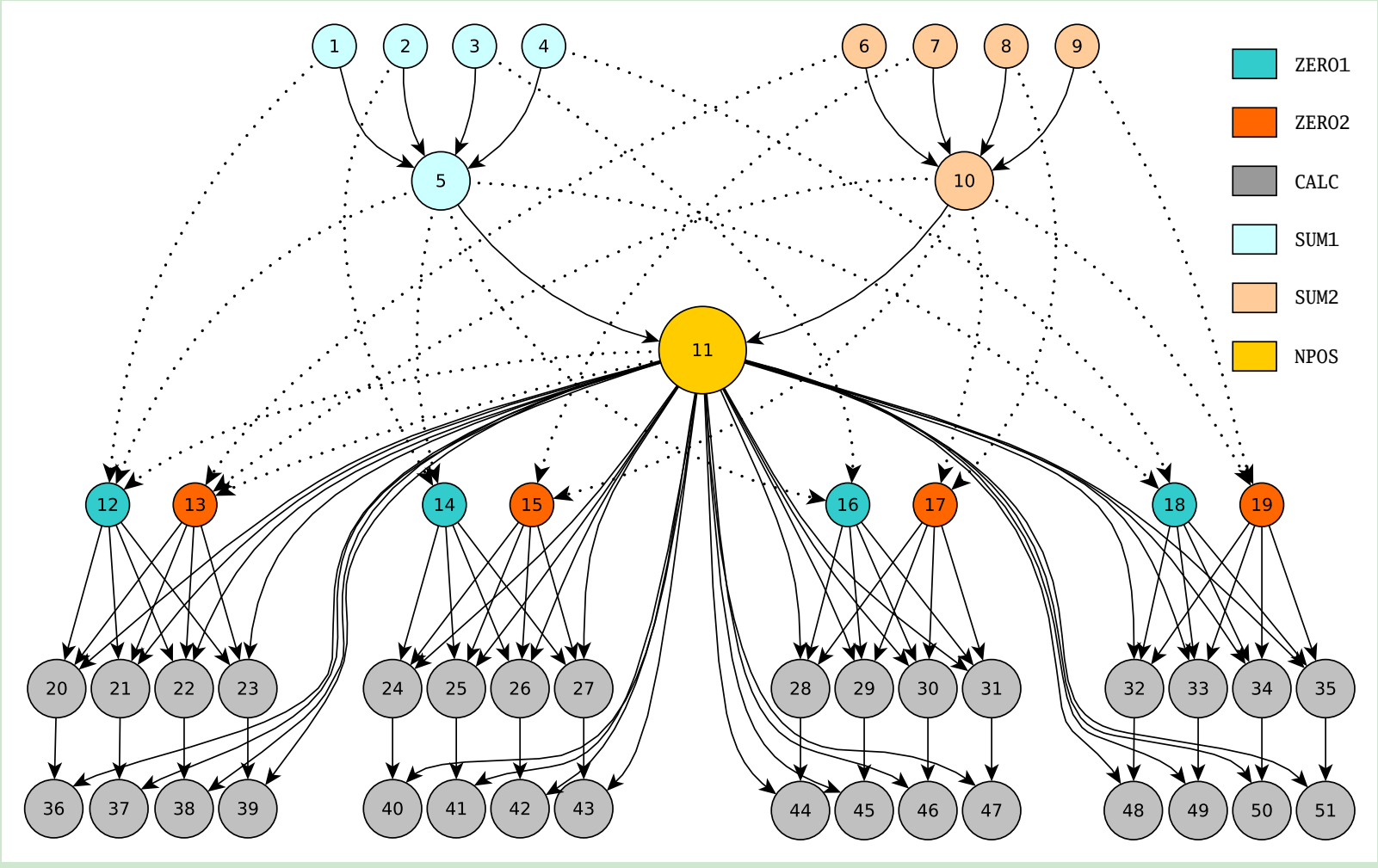
1. **METIS Partitioning (MP)** makes use of the METIS library.
   - Balanced partitions and small edge-cut.
2. **Greedy Partitioning (GP)** exploits the regularity of the studied graphs.
   - Balanced partitions and small (close to optimal) edge-cut.
3. **Random Partitioning (RP)** assigns each node to a part randomly.
   - Balanced partitions but edge-cut is not considered.
4. **Ordered Partitioning (OP)** orders the nodes and assigns them to the parts in a round-robin fashion.
   - Balanced partitions but edge-cut is not considered.

Subgraph Partitioning Techniques

1. **METIS Partitioning (MP)** makes use of the METIS library.
   - Balanced partitions and small edge-cut.
2. **Greedy Partitioning (GP)** exploits the regularity of the studied graphs.
   - Balanced partitions and small (close to optimal) edge-cut.
3. **Random Partitioning (RP)** assigns each node to a part randomly.
   - Balanced partitions but edge-cut is not considered.
4. **Ordered Partitioning (OP)** orders the nodes and assigns them to the parts in a round-robin fashion.
   - Balanced partitions but edge-cut is not considered.

Sample Subgraph between Barriers

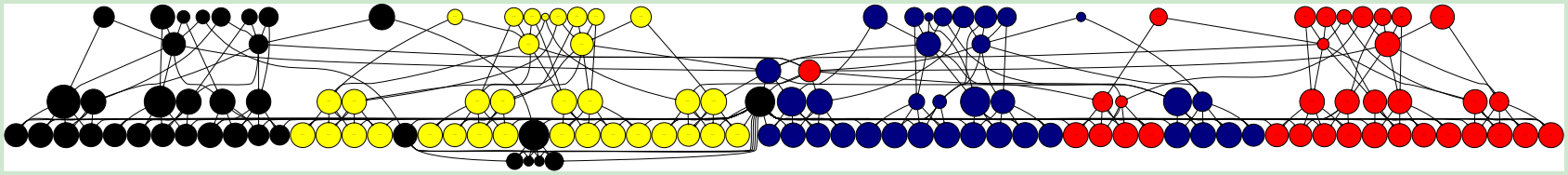
Example



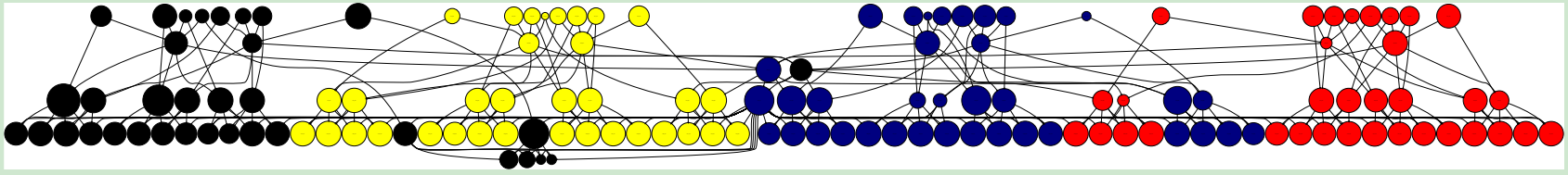
Subgraph between two consecutive barriers.

Sample Partitions of a Subgraph

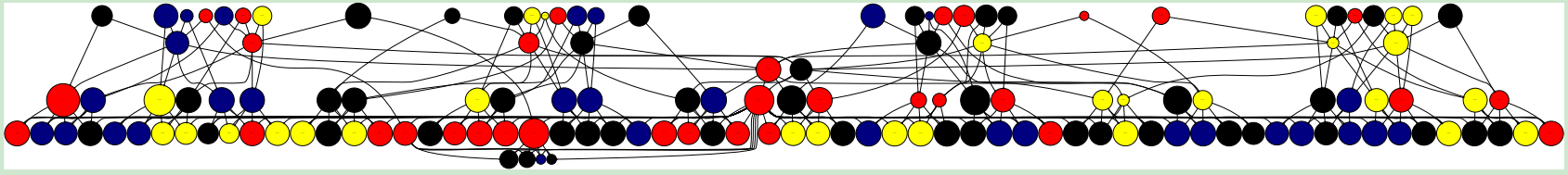
Example



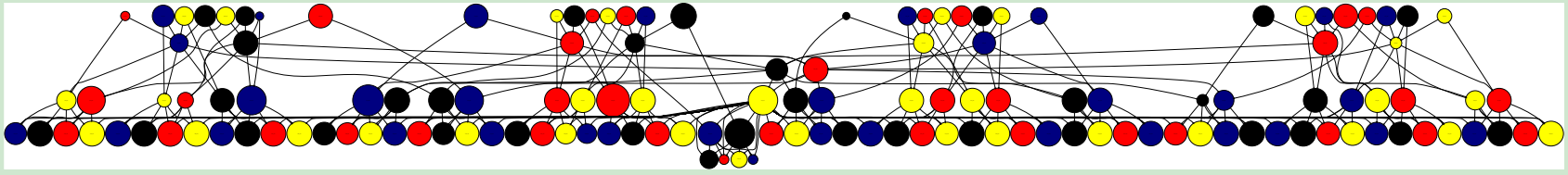
MP.



GP.



RP.



OP.

Matching Schemes

1. **Propagating Partitions (PP)** partitions only one subgraph and replicates the partition to all other subgraphs.
2. **Best Permutation Matching (BPM)** partitions every subgraph independently and finds the permutation of parts which minimizes the edge-cut.
3. **Worst Permutation Matching (WPM)** partitions every subgraph independently and finds the permutation of parts which maximizes the edge-cut.
   - Only for comparison purposes.

Matching Schemes

1. **Propagating Partitions (PP)** partitions only one subgraph and replicates the partition to all other subgraphs.
2. **Best Permutation Matching (BPM)** partitions every subgraph independently and finds the permutation of parts which minimizes the edge-cut.
3. **Worst Permutation Matching (WPM)** partitions every subgraph independently and finds the permutation of parts which maximizes the edge-cut.
   - Only for comparison purposes.

Matching Schemes

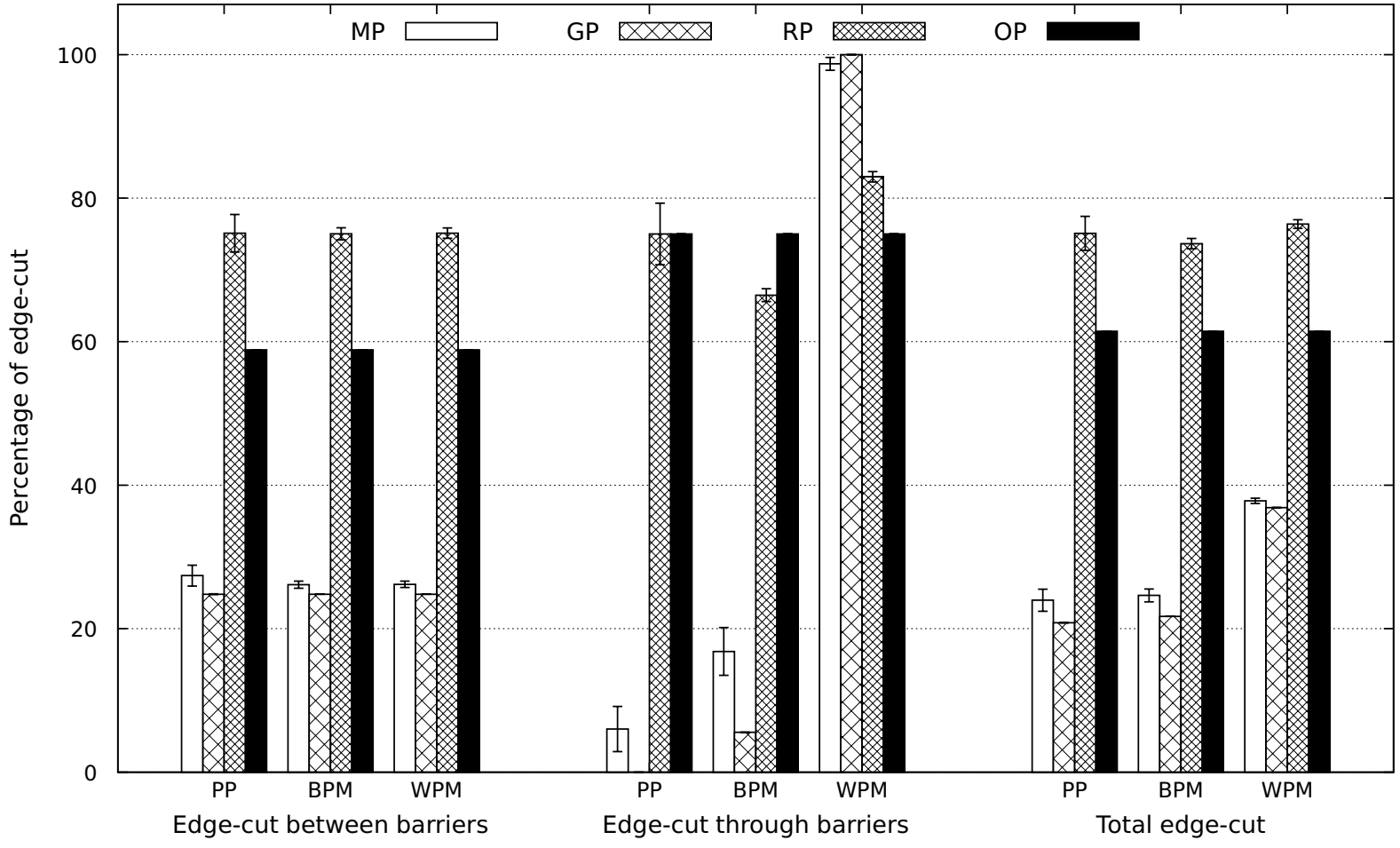
1. **Propagating Partitions (PP)** partitions only one subgraph and replicates the partition to all other subgraphs.
2. **Best Permutation Matching (BPM)** partitions every subgraph independently and finds the permutation of parts which minimizes the edge-cut.
3. **Worst Permutation Matching (WPM)** partitions every subgraph independently and finds the permutation of parts which maximizes the edge-cut.
   - Only for comparison purposes.

Outline

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

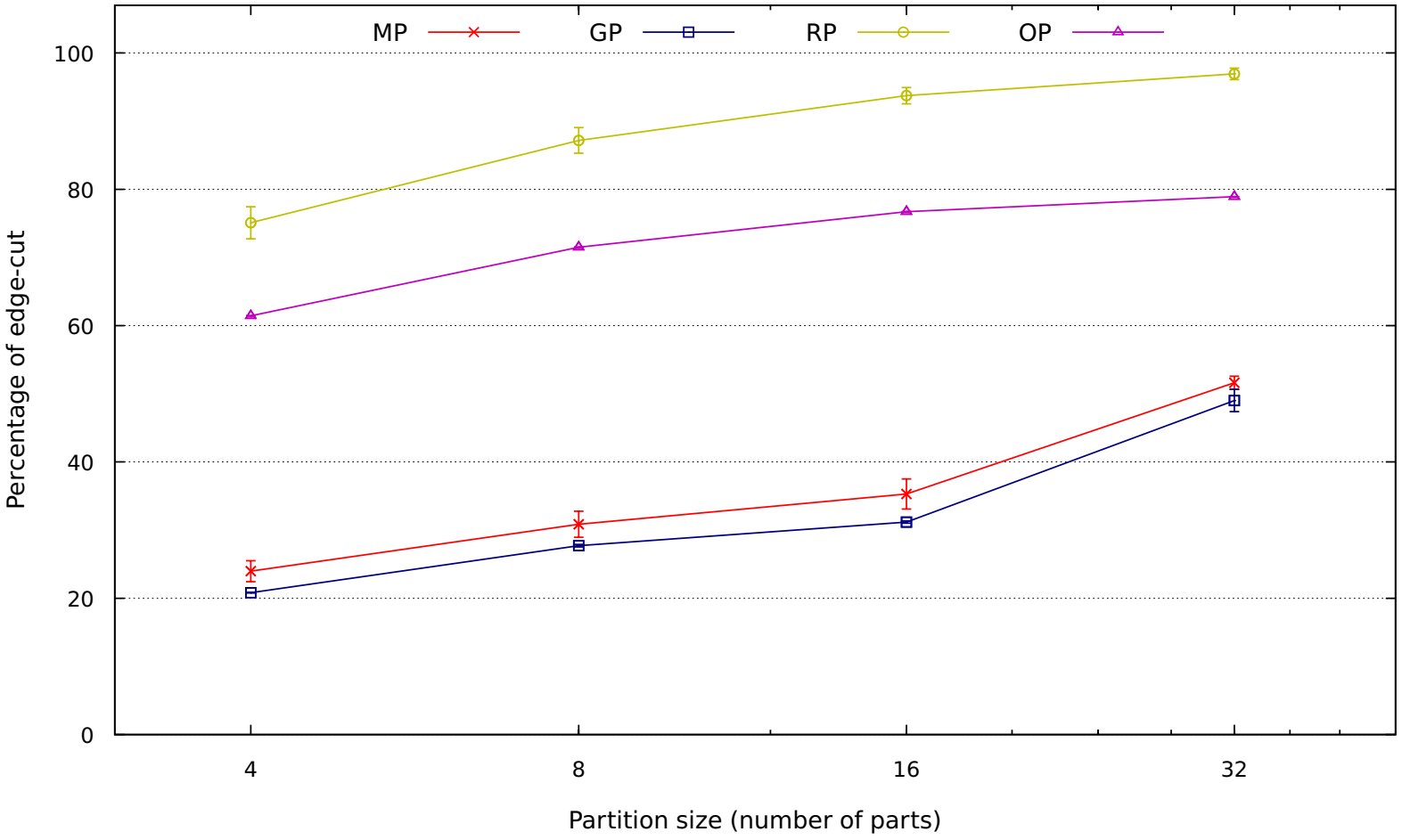
RoMoL Project

Comparison of Partitioning Strategies



Edge-cut of partitions in 4 parts of the graph corresponding to an execution of k-means with 65536 points in $\mathbb{R}^4$, 8 clusters and 64 SPUs.

How Partitioning Techniques Scale



Edge-cut of partitions of the graph corresponding to an execution of k-means with 65536 points in $\mathbb{R}^4$, 8 clusters and 64 SPUs.

Outline

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

RoMoL Project

Summary

- Communication between computing nodes can be effectively reduced with some preprocessing.
  - METIS produces very good partitions for k-means executions.
- The same strategy is likely to achieve comparable results on many other parallel applications.

Thank you for your attention.

Any questions?