The 6th International Symposium on Intelligent Distributed Computing - IDC 2012
September 24-26, 2012, Calabria, Italy

Efficient Group Communication for Large-Scale Parallel Clustering

David Pettinger and Giuseppe Di Fatta

G.DiFatta@reading.ac.uk

University of Reading

Tuesday, 25 September 2012

Outline

- k-means for cluster analysis
  - The ‘brute force’ algorithm
  - Computational efficiency: ”how fast (slow) is the k-means method”?
- Parallel formulation for distributed memory systems
  - Parallel ‘brute force’ k-means
  - Parallel KD-tree k-means
  - Communication efficiency: how scalable are these algorithms?
- Global communication and synchronisation requirement
  - Contribution: first work to challenge this assumption in k-means
    - optimisation technique for the communication cost
    - novel communication operation
- Results and conclusions

k-means Clustering

- The basic k-means algorithm (MacQueen 1967, Loyd 1982)
  - Partitional Clustering approach
    - Clusters are disjoint subsets of the input data

- Each cluster is associated with a centroid (center of mass)
- Each point is assigned to the cluster with the closest centroid
  - ‘Closeness’ is measured by a distance function, e.g. euclidean distance, cosine similarity, correlation, etc.
- Number of clusters, K, must be specified.
  - Initial centroids are often chosen randomly.
  - Given the initial centroids, the algorithm is deterministic.
- Greedy approach: k-means is guaranteed to converge
  - local minima vs global optimum

The ‘brute force’ Algorithm

University of Reading

```
Repeat until convergence                      → # iterations: I
    For each input pattern x                  → # patterns: n
        For each centroid $c_i$               → # clusters: K
            compute the distance $d_E(x, c_i)$ → # attributes: d
        Find closest centroid $c^*$           → K
        Add x to cluster $c^*$                → d
    For each cluster                          → K
        recompute centroid                    → d
```

- At each iteration the amount of computation is constant.
- # operations at each iteration: $\underbrace{3nKd}_{\text{distance calculations}} + \underbrace{nK}_{\text{cluster assignment}} + \underbrace{nd + Kd}_{\text{centroid recalculations}}$
- Complexity: $O(I \cdot n \cdot K \cdot d)$

Optimisation Techniques for k-means

Main techniques:

1. Hierarchical Data Partitioning
   - data space is partitioned, statistics on the aggregated data points are computed in a pre-processing step, e.g. space partitioning trees
     - KD-Trees (Alsabti 1998, Pelleg 1999, Kanungo 2002)
     - BSP-Trees (Di Fatta 2010)
2. The Triangular Inequality Principle
   - to avoid distance computations
   - e.g. spheres of guaranteed assignment (Judd 1998)
3. Distance Bounds
   - computing the distance between two data points can be performed in O(d), where d is the space dimensionality. While lower and upper bounds of distances can be computed in constant time. Tests on exact distances can be replaced by equivalent tests on the corresponding bounds.

Approaches may use a combination of the above.

Parallel k-means

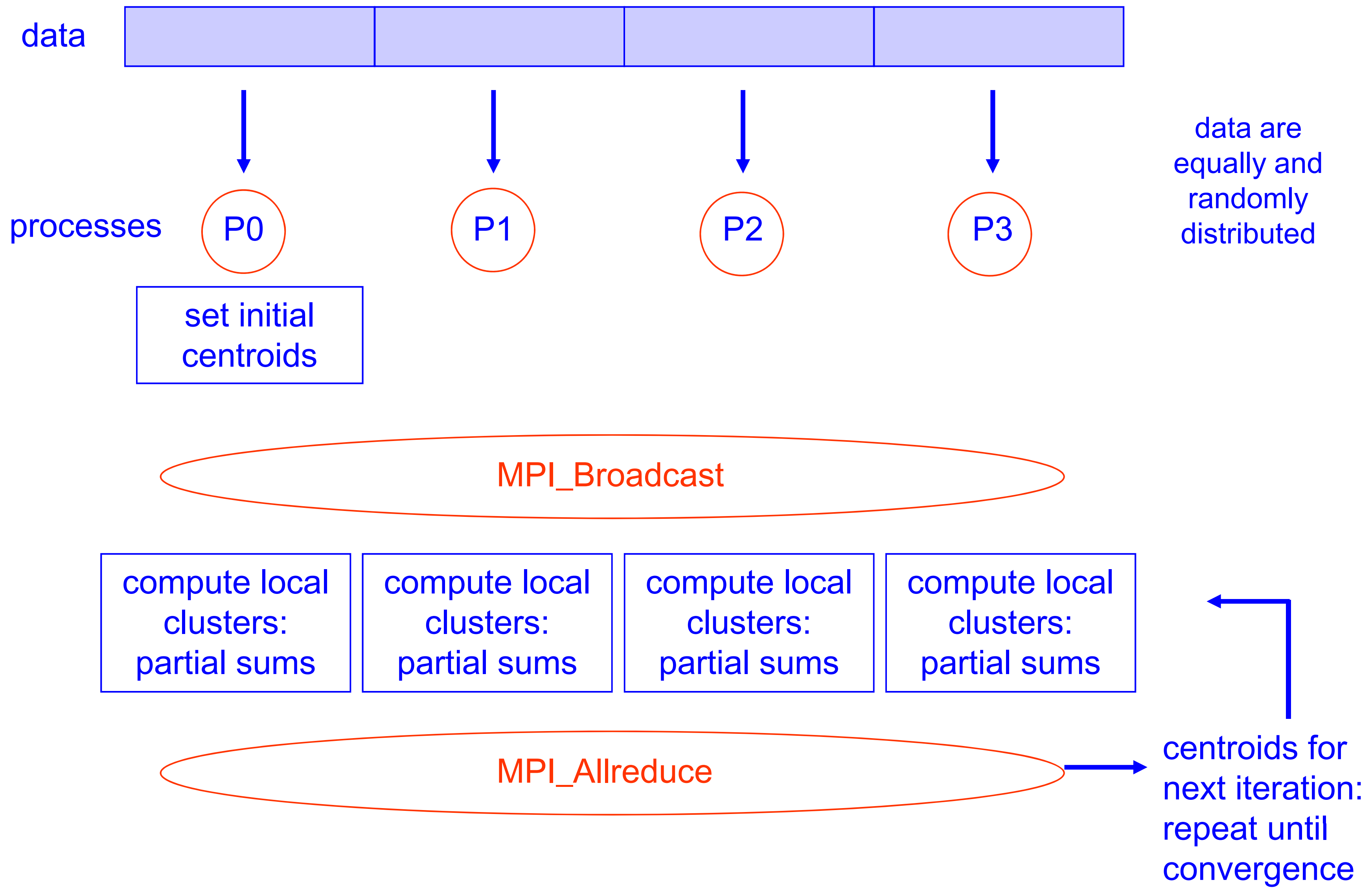
- Single Program Multiple Data (SPMD) approach
- Shared-nothing machine with p processing elements

➢ Parallel ‘brute force’ k-means (Dhillon 2000)
  - patterns equally distributed among processors (random partitioning)
    - work load is identical for each processor

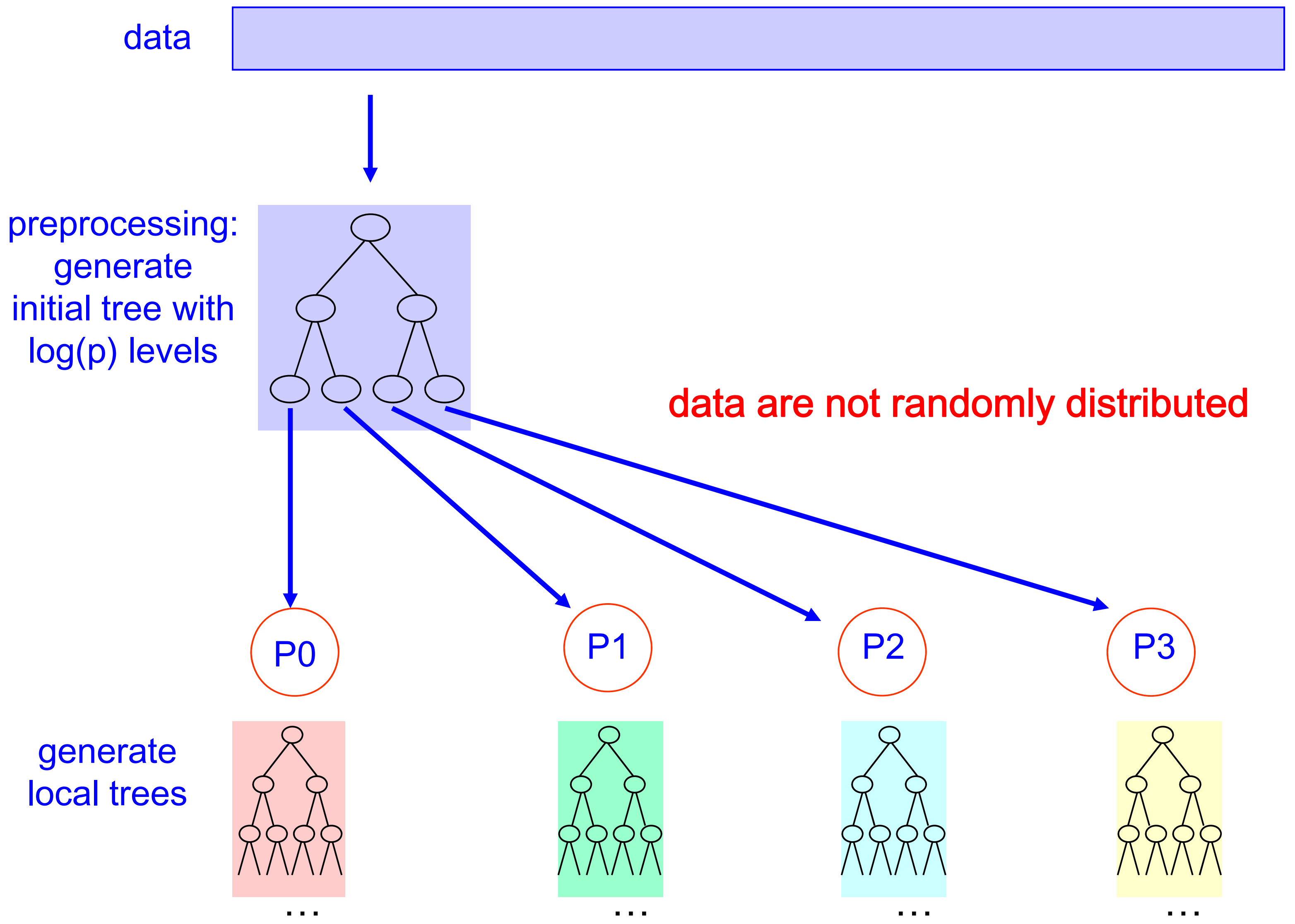
➢ Parallel KD-tree k-means (Di Fatta 2010)
  - computationally more efficient
  - load imbalance of static partitioning limits its scalability
    - Dynamic Load Balance (DLB)

- Both require global communication and synchronisation

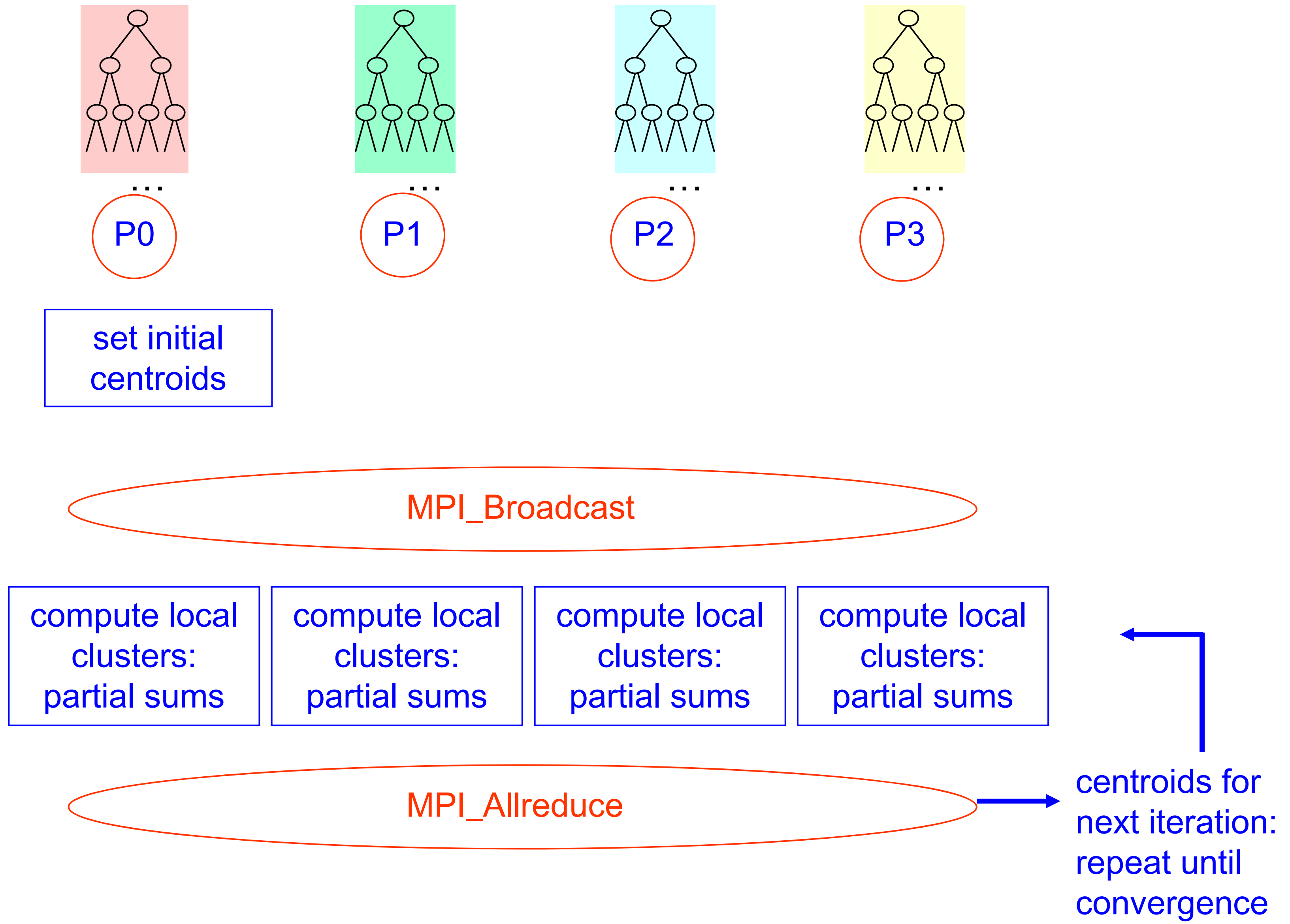
Parallel 'brute force' k-means


data
processes
P0
P1
P2
P3
set initial centroids
data are equally and randomly distributed
MPI_Broadcast
compute local clusters: partial sums
compute local clusters: partial sums
compute local clusters: partial sums
compute local clusters: partial sums
MPI_Allreduce
centroids for next iteration: repeat until convergence

Parallel KD-tree k-means


data
preprocessing: generate initial tree with log(p) levels
data are not randomly distributed
P0
P1
P2
P3
generate local trees
...

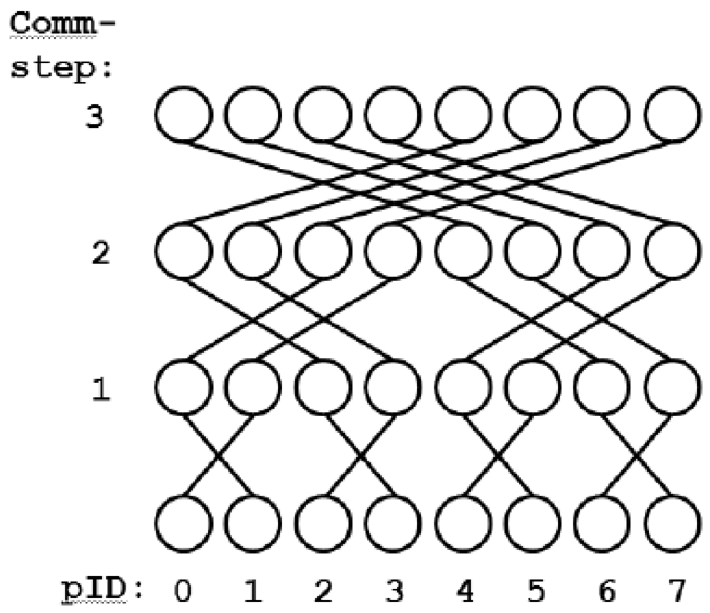
Parallel KD-tree k-means



...
P0
P1
P2
P3
set initial centroids
MPI_Broadcast
compute local clusters: partial sums
compute local clusters: partial sums
compute local clusters: partial sums
compute local clusters: partial sums
MPI_Allreduce
centroids for next iteration: repeat until convergence

Communication Requirements

- Parallel k-means implementations adopt a global communication approach: broadcast and allreduce.
  - *Allreduce* is used to compute the updated centroids globally at the end of each iteration.
  - Recursive doubling algorithm



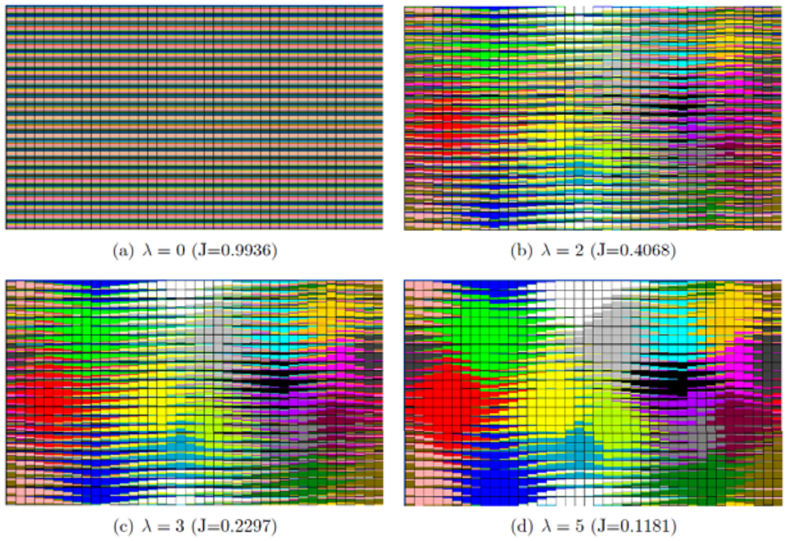
Recursive doubling for all-reduce

Can we remove the global communication requirement?

Skewed Cluster Data Distributions

- In intrinsically distributed systems cluster data are not uniformly distributed over the nodes.
- In parallel systems, we can induce skewed data partitions by means of KD-trees or BSP-trees.

- The contribution of every process may not be required in every cluster.
- In other words, in each cluster the contribution of a subset of processes may be required.

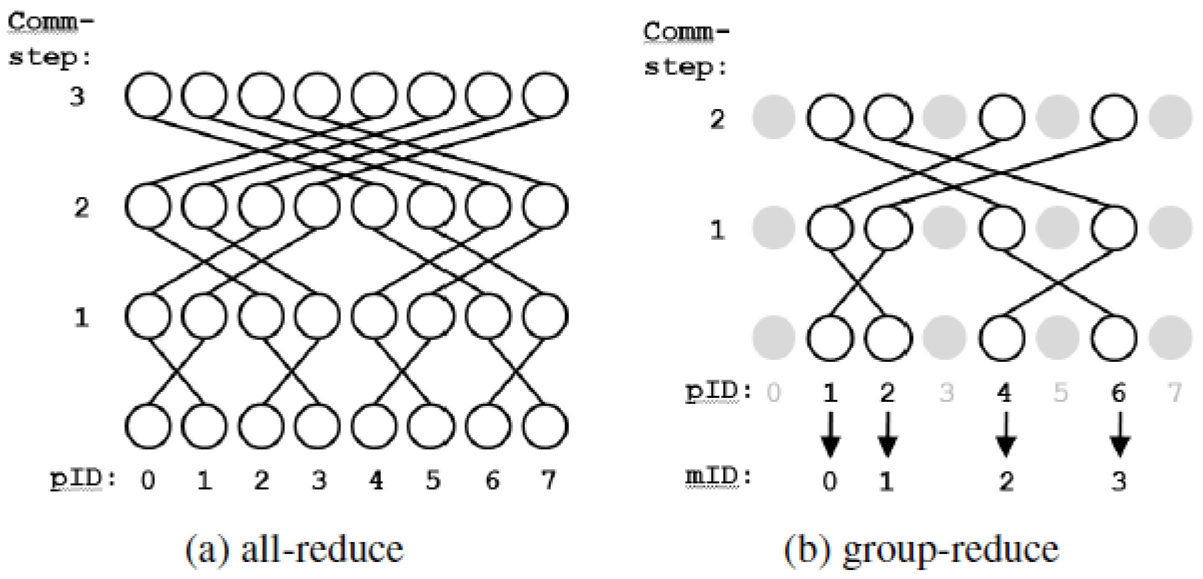


Four examples of cluster distributions

New Reduction Operation

- Group-reduce is an efficient reduction operation, which is performed on dynamic groups of processes with adaptive membership.
- Number of communication steps:
  - $\lceil log2(p') \rceil$, *where $p' = max_k(|P_k|)$ and P_k is the subset of processes* contributing to cluster *k.*

$log_2(p)$



$log_2(p')$ with $p' \leq p$

Recursive doubling for all-reduce and group-reduce

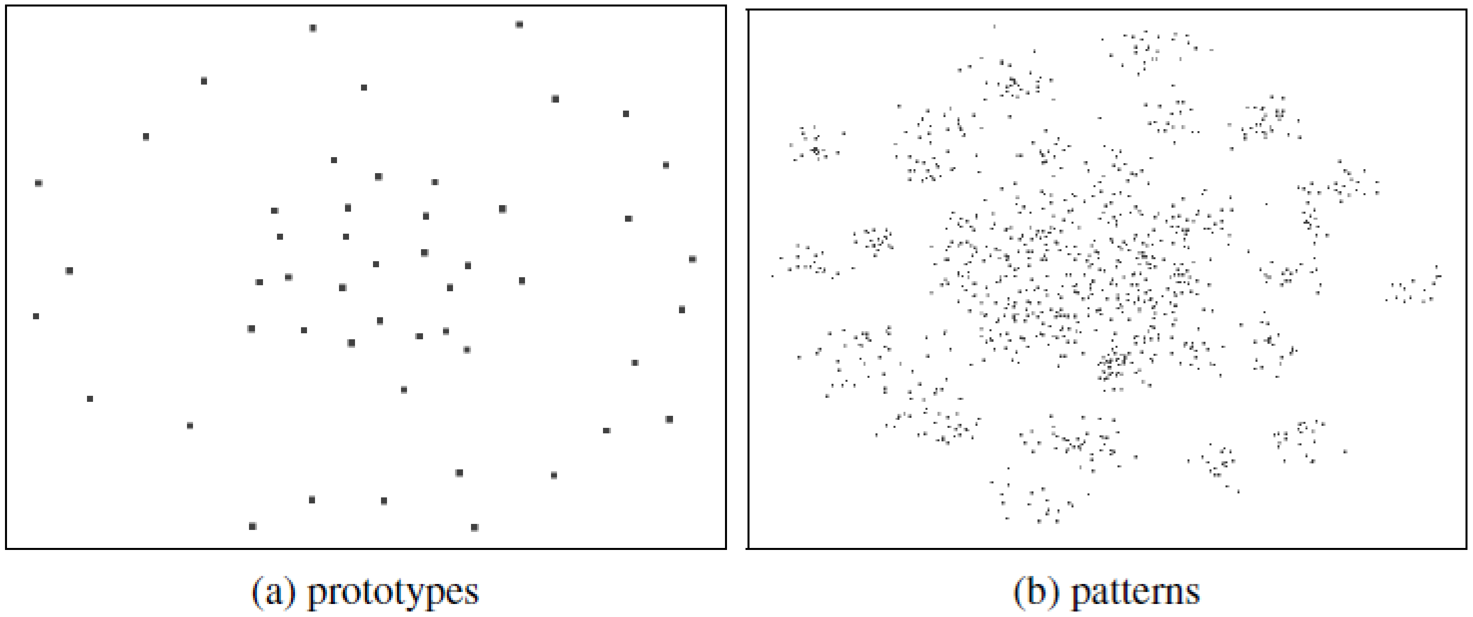
Exact and Approximate Algorithms

- Exact algorithm:
  - Based on parallel KD-tree (or BSP-tree) k-means with group-reduce.
  - Exact: processes contribute with all their local information.
  - The results are deterministic and identical to the sequential k-means algorithm run on the aggregated data.
    - same clustering solution in the same number of iteration
    - less overall computation, less communication

- Approximate algorithm:
  - Processes contribute with local data if sufficiently relevant in the cluster:
    - the number of local data point assigned to the cluster is more than a minimum percentage (threshold) of the number of global data point in the cluster.
  - Provides even further optimisation in terms of both computation and communication.
  - The results are not deterministic, nor identical to the sequential k-means algorithm run on the aggregated data.

Artificial Datasets

- N=200000 patterns in d=20-dimensional space, K = 50
- Skewed data distribution: mixed Gaussian distributions
  - 25 prototypes uniformly in the whole domain and 25 prototypes restricted to a subdomain. Patterns generated using a multivariate Gaussian distribution with a random standard deviation in the range [0.0,0.1].
- MDS projection in 2D:



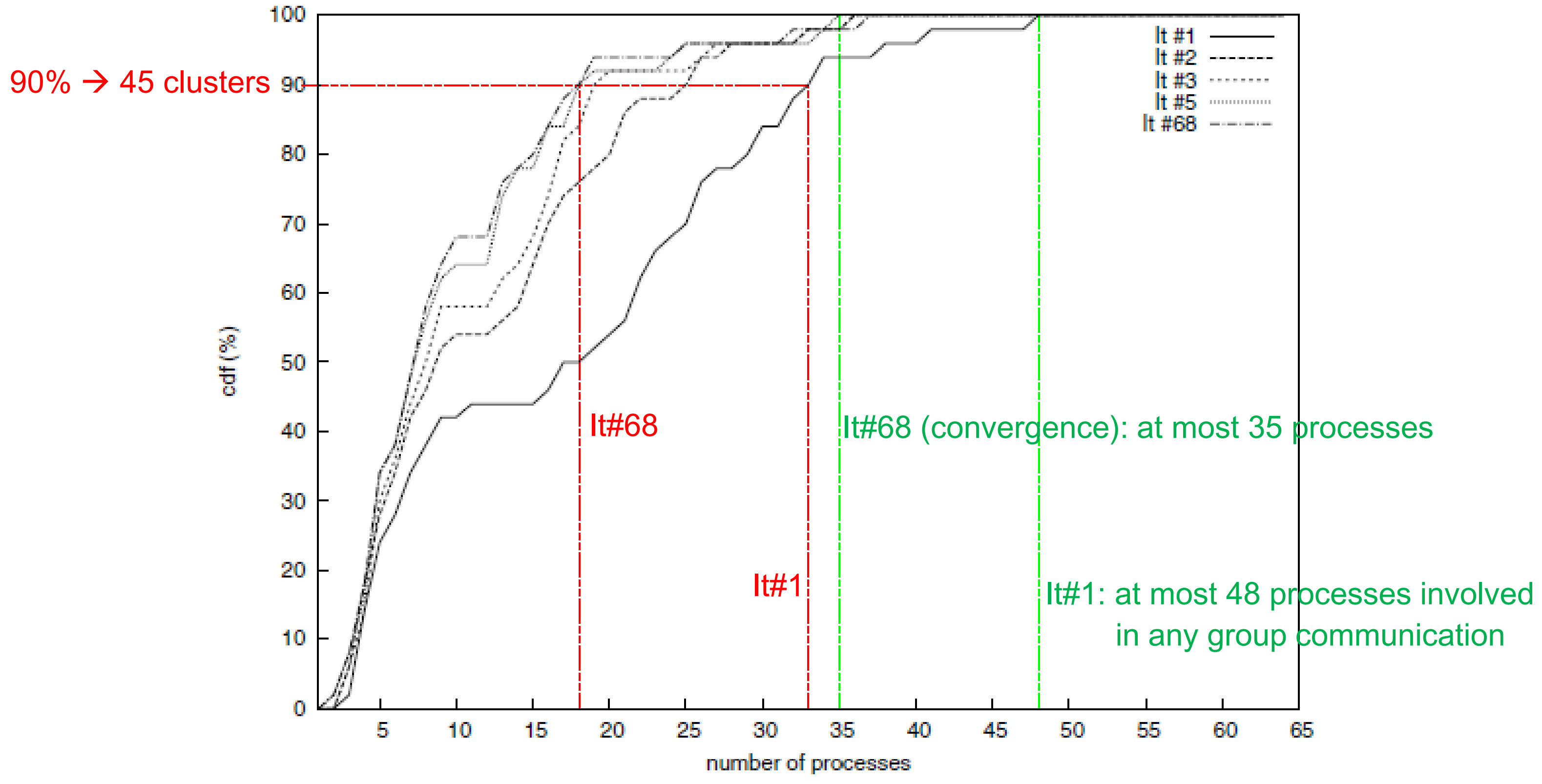
(a) prototypes (b) patterns

2D representation of the multi-dimensional prototypes (a) and patterns (b)

Results – exact algorithm

- Executions with 64 processes
  - IBM Bladecenter cluster (2.5GHz dual-core PowerPC 970MP) connected via a Myrinet network, running Linux (2.6.16.60-0.42.5-ppc64) and J2RE 1.5.0 (IBM J9 2.3)
  - N=200000 patterns in d=20 dimensional space with K=50 and mixed Gaussian distributions

Empirical cumulative distribution function Prob[p≤x] : 100% → 50 clusters

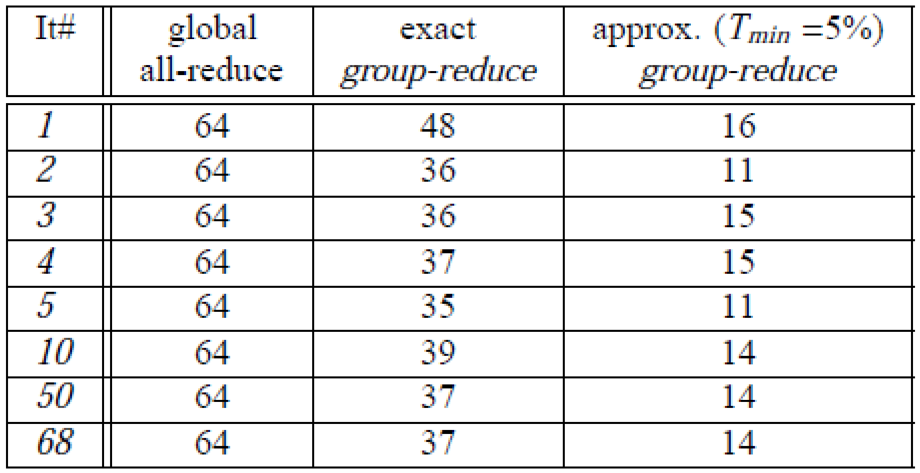


Processes in the *group-reduce* operations: exact reduction at different iteration cycles

Results – approximate algorithm

- Max number of processes in any cluster group for the exact and approximate (5%) approaches.

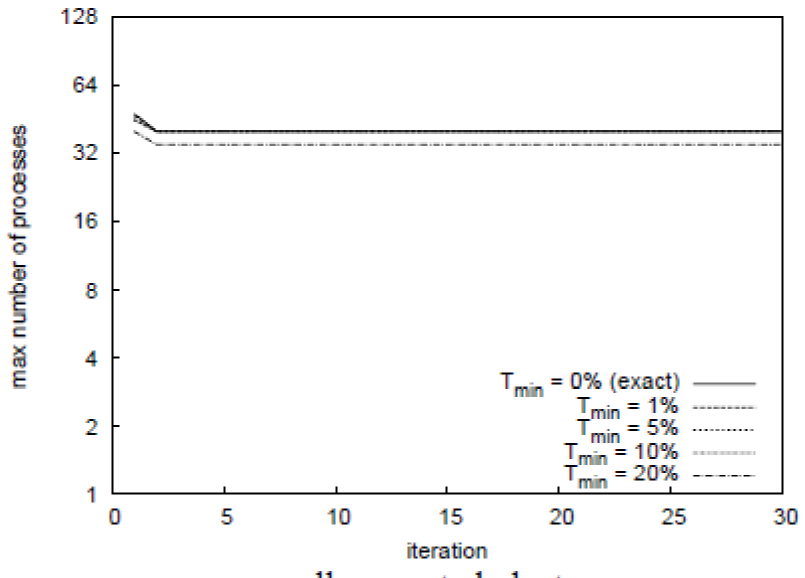
Maximum number of processes involved in reduction operations



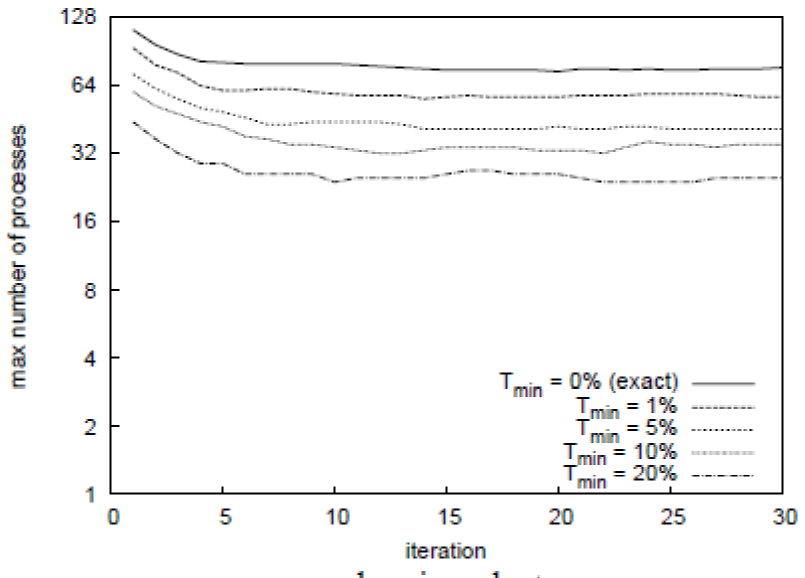
It#	global all-reduce	exact *group-reduce*	approx. (T_{min} =5%) *group-reduce*
1	64	48	16
2	64	36	11
3	64	36	15
4	64	37	15
5	64	35	11
10	64	39	14
50	64	37	14
68	64	37	14

Simulations (additional results)

- When does the approximate approach lead to an improvement?
- PeerSim simulations with 1024 nodes.
  - charts: max procs vs it#
- data similarity matters!



well-separated clusters



overlapping clusters

Maximum number of processes in the *group-reduce* operations (1024 nodes): exact and approximate methods

Conclusions

- k-means method for cluster analysis
  - ~50 years from the first publication, a popular and influential data mining algorithm
  - sequential algorithm: ‘brute force’ approach and several optimisation techniques

- Parallel k-means
  - Computation
    - The ‘brute force’ algorithm enjoys perfect load balancing for homogenous systems
    - A more efficient and scalable variant is based on partitioning trees with dynamic load balancing
  - Communication
    - All-reduce requires global synchronisation and communication.
    - The proposed group-reduce does not.
      - equivalent to the original k-means algorithm and fully deterministic
      - approximate variant (not equivalent, not deterministic)
      - better scalability w.r.t. #procs can be achieved: for extreme-scale systems

- Future work: simulations on larger systems and asynchronous approach
- Alternative paradigm: Epidemic k-means Clustering (DiFatta 2011)

G.DiFatta@reading.ac.uk