High Performance Computing for Remote Sensing

Silviu Panica, Dana Petcu, Daniela Zaharie

West University of Timisoara

Universitatea de Vest din Timişoara

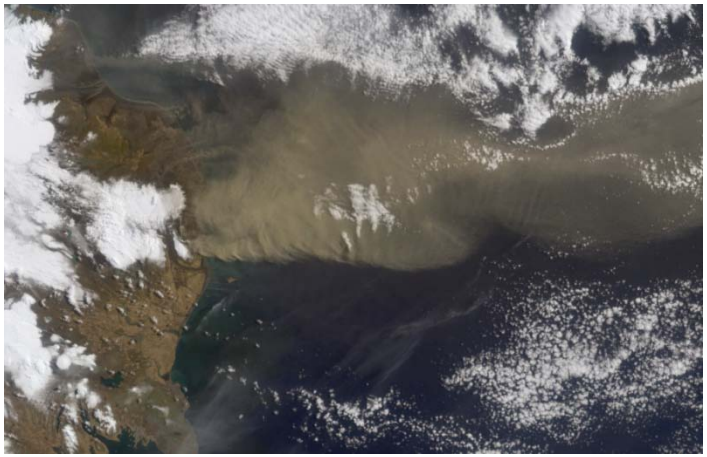
Motivation

Problem

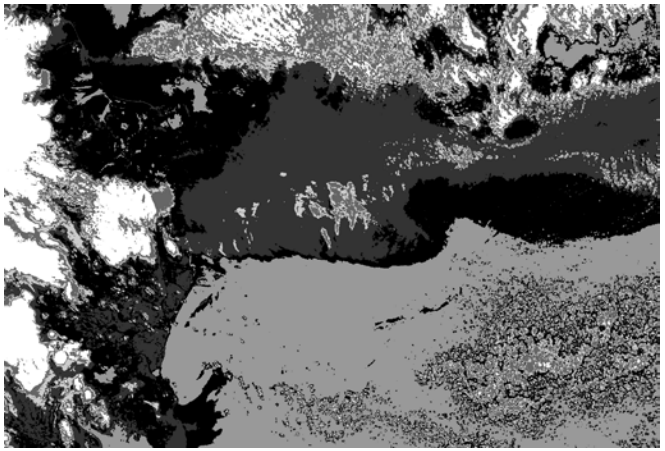
- Unsupervised classification (clustering) = identify regions in the image characterized by similar feature values

Initial context

- ESA PECS Project GiSHEO – On Demand Grid Services for High Education and Training in Earth Observation (2009-2010)
- IBM OCR project - Clustering of Large Satellite Images using HPC (2010)



Original image



Classified image (6 classes)

Current work:
in the framework of

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

Motivation

Challenges in unsupervised classification of satellite images:

- Pixels may contain spectral information corresponding to different ground components
  - Possible solution: assign each pixel to several classes based using membership values (Fuzzy Clustering)
- The image can contain noise because of the limited sensors sensitivity
  - Possible solution: use both spectral and spatial information (spatial variants of Fuzzy Clustering)
- The increase of spatial and spectral resolution of sensors led to large images (high number of pixels and/or high number of spectral bands)
  - Possible solution: spatial (or spectral) domain partitioning and parallel (or distributed) implementation of clustering algorithms

Agenda

Parallel implementations of:

- Fuzzy C means
- Spatial Fuzzy C means

Comparative analysis of the parallel implementations for large images on two platforms (http://hpc.uvt.ro):

- InfraGrid cluster
- BlueGene/P

Lessons learned from porting the parallel implementations to BlueGene/P

Fuzzy C-Means

- Iterative algorithm for fuzzy unsupervised classification (clustering) of images
- **Input data:**
  - Image = $\{x_1, ..., x_n\}$, n = number of pixels

    $x_i = (x_{i1},..., x_{id})$, d = number of spectral bands

    (set of vectors corresponding to all pixels and containing the values corresponding to the spectral bands)
  - Number of classes to be identified (c)
- **Output data:**
  - Membership matrix (of size c x n) =(u_{ij}) , j=1..c, i=1..n

    u_{ij} is a value in [0,1] specifying the degree of membership of pixel i to class j
  - Classes centroids = $\{v_1,...,v_c\}$
  - Classified image = $\{y_1,...,y_n\}$, y_i = value related to the label of the class to which x_i belongs (that corresponding to the largest membership value)

Fuzzy C-Means

FCM Algorithm [Bezdek, 1981]

- Initialization of the membership values (U)
- DO
  - Compute the centroids (V)
  - Estimate the membership values

  WHILE (there are significant changes in the membership values)
- Construct the classified image (each pixel is assigned to the class having the largest membership value)

Centroids computation (global)

$$v_j = \frac{\sum_{i=1}^{n} u_{ij}^m x_i}{\sum_{i=1}^{n} u_{ij}^m}, \quad j = \overline{1,c}$$

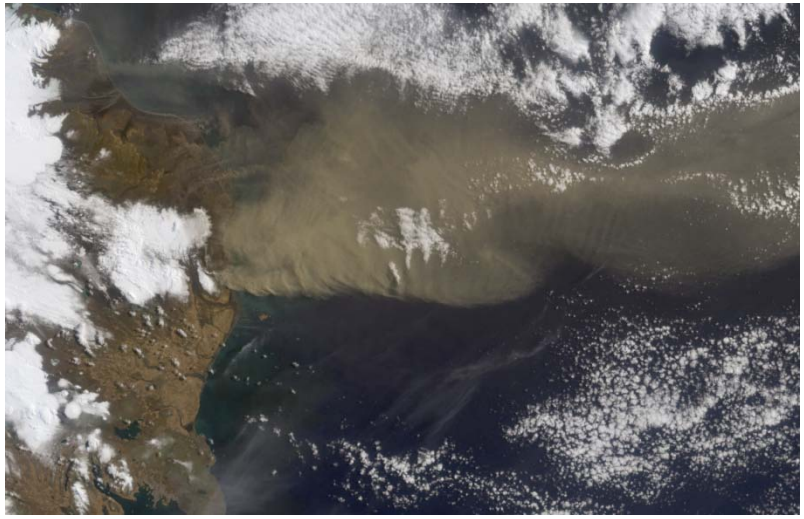
m>1 is a parameter (e.g. m=2)

Membership values estimation (local)

$$u_{ij} = \frac{1}{\left\|x_i - v_j\right\|^{2/(m-1)} \sum_{k=1}^{c} 1/\left\|x_i - v_k\right\|^{2/(m-1)}}$$

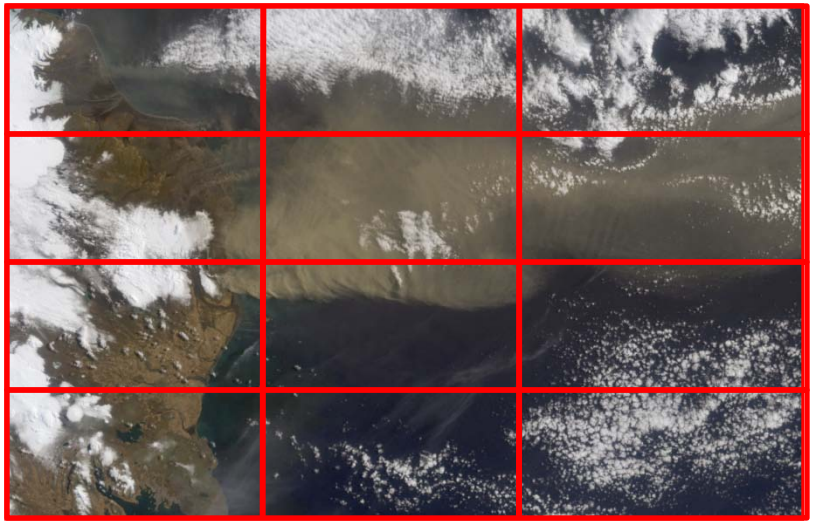
$$i = \overline{1,n}, \; j = \overline{1,c}$$

Fuzzy C-Means



- **Costs (sequential implementation)**
  - Space: O(max{n*c,c*d,n*d})
  - Time: O(n*c*d*iterations)
- **Typical values:**
  - Image size: $n > 10^6$
  - Number of classes: c in {2,..,50}
  - Number of spectral bands: d in {3,..,250}
  - Number of iterations: around several hundreds
- **Main impact on the cost:** image size (number of pixels * number of spectral bands)

Fuzzy C-Means



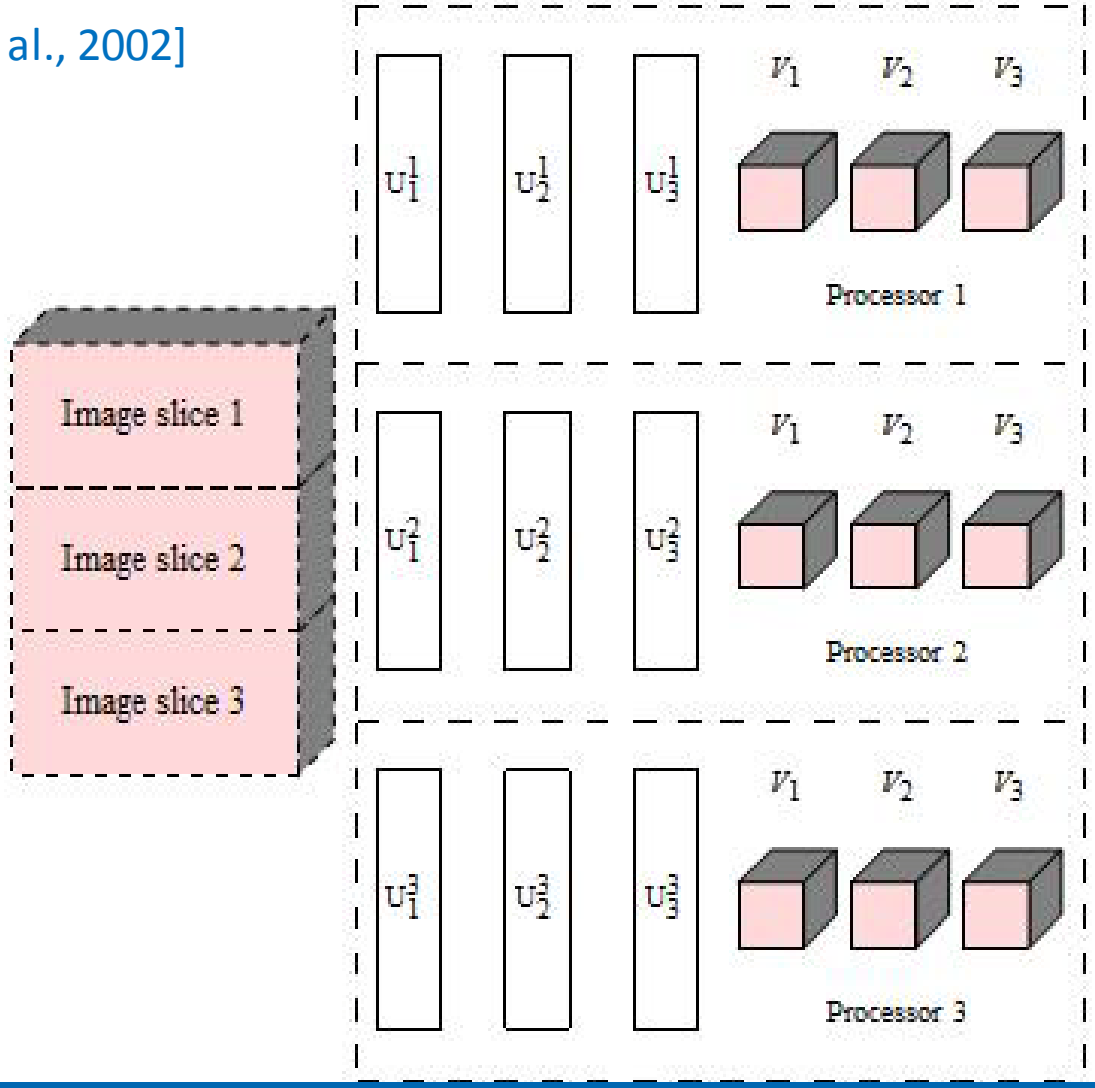
- **Costs (sequential implementation)**
  - Space: O(max{n*c,c*d,n*d})
  - Time: O(n*c*d*iter)
- **Typical values:**
  - Image size: $n > 10^6$
  - Number of classes: c in {2,..,50}
  - Number of spectral bands: d in {3,…,50}
  - Number of iterations: around several hundreds
- **Main impact on the cost:** image size (number of pixels * number of spectral bands)
- **Parallelization idea:** split the image in smaller slices and estimate the membership values in parallel for each slice

Parallel Fuzzy C-Means

Parallelization idea [Kwon et al., 2002]

Each processor:

- Deals with an image slice
- computes the membership values (U) corresponding to the image slice (local computation)
- computes the complete set of centroids (V) (global computation)



Parallel Fuzzy C-Means

- **Split the image in slices:** $S_1,\ldots, S_P$
- **Split the computation:**

$$v_j = \frac{\sum_{i \in S_1} u_{ij}^m x_i + \ldots + \sum_{i \in S_P} u_{ij}^m x_i}{\sum_{i \in S_1} u_{ij}^m + \ldots + \sum_{i \in S_P} u_{ij}^m}$$

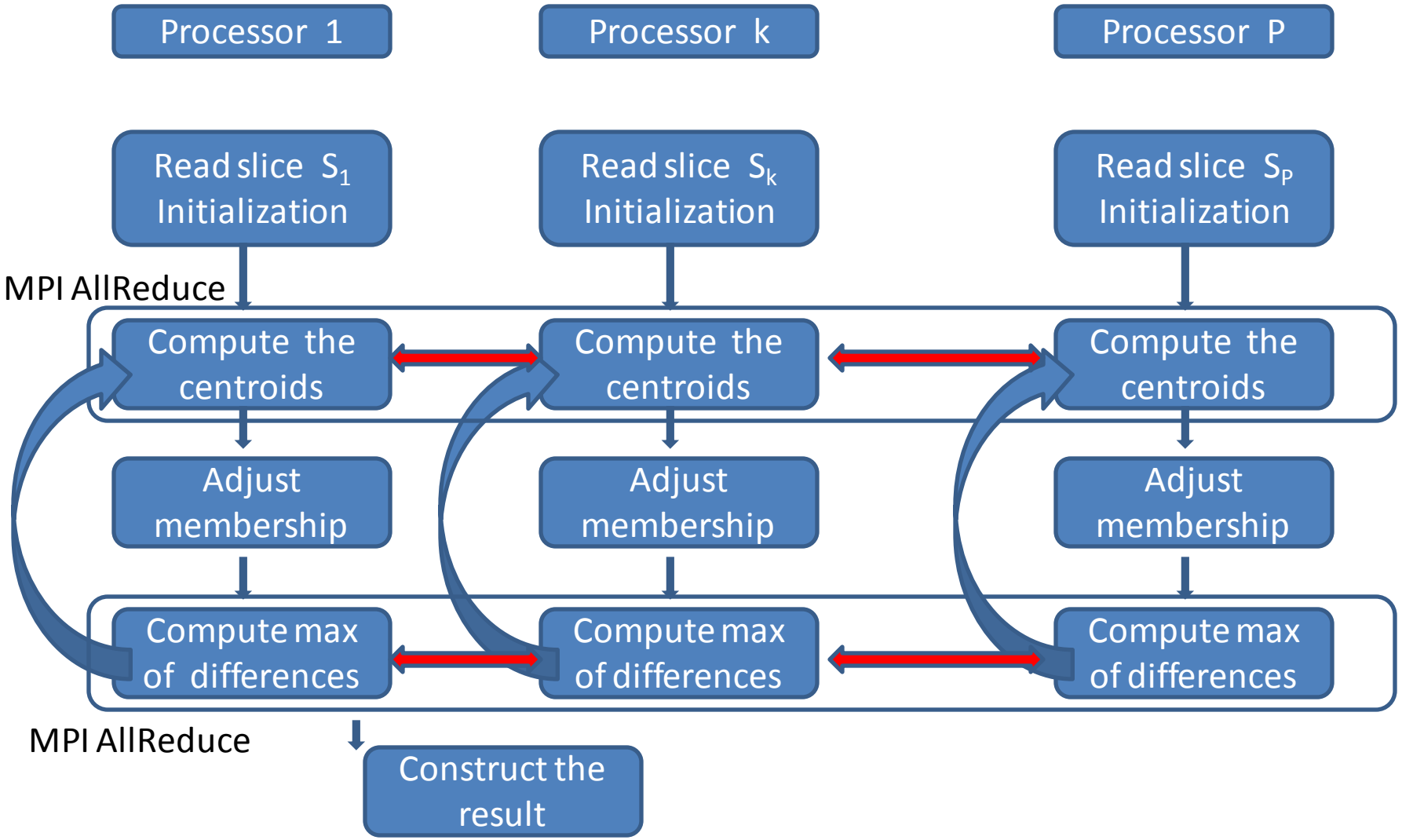
- **Processor k computes:**
  - The corresponding membership values
  - The partial sums involved in the centroids computation

$$\sum_{i \in S_k} u_{ij}^m x_i \quad \text{and} \quad \sum_{i \in S_k} u_{ij}^m$$

  - The local maximal difference between the membership values at two consecutive iterations

$$\max\{|u_{ij}(\text{iter}+1) - u_{ij}(\text{iter})|;\ i \in S_k,\ j = \overline{1,c}\}$$

Parallel Fuzzy C-Means



Parallel Fuzzy C-Means

Communication costs (at each iteration):

FCM computation	MPIAllReduce (calls/iteration)	MPIAllReduce (data size)	MPIAllReduce (operation)
$\sum_i u_{ij} x_i,\ j = \overline{1,c}$	1	c*d (classes*bands)	Sum
$\sum_i u_{ij},\ j = \overline{1,c}$	1	c (classes)	Sum
$\max_{i,j} \lvert u_{ij}(k+1) - u_{ij}(k) \rvert$	1	1	Maximum

SFCM – Spatial FCM

Aim: reduce the number of spurious blobs caused by noisy pixels

Main idea: adjust the membership values using averages over a neighborhood [Chuang et al, 2006]

$$u'_{ij} = \frac{u_{ij}^{p} h_{ij}^{q}}{\sum_{k} u_{ik}^{p} h_{ik}^{q}}, \quad h_{ij} = \sum_{k \in N(i)} u_{kj}$$

Additional costs:

Space: O(n*c)

Time: $O(n*c*NS^2)$ – spatial info comp.

O(n*c) – membership values adjust.

NS = neighborhood size (e.g. 5, 7, 9 …)

SFCM Algorithm

- Initialization of the membership values
- DO
  - Compute the centroids
  - Compute the spatial information (h_{ij})
  - Estimate the membership values (u_{ij})
  - Adjust the membership values (u'_{ij})

  WHILE (there are significant changes in the membership values)
- Construct the classification

Parallel SFCM

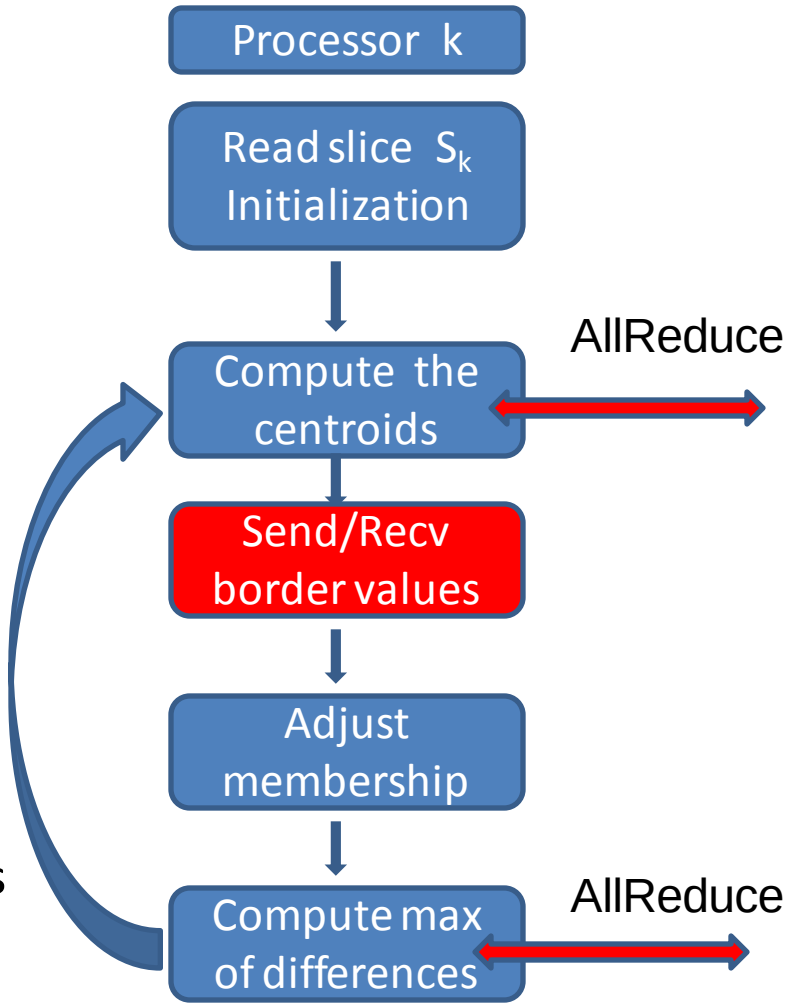
General structure: same idea as in parallel FCM

Processor k computes:

- The corresponding membership values
- The partial sums involved in the centroids computation
- The local maximal difference between the membership values at two consecutive iterations

Particularity:

- The computation of spatial information for pixels on the border of the image slice needs the communication of some membership values between processors



Parallel SFCM

Costs of Send/Receive operations:

- It depends on the image partitioning model

Hypotheses: h = image height, w = image width, NS = neighborhood size

- Vertical/horizontal partitioning: processor p communicates with (p-1) and (p+1)

  size of data: c*w*NS/2 (or c*h*NS/2)

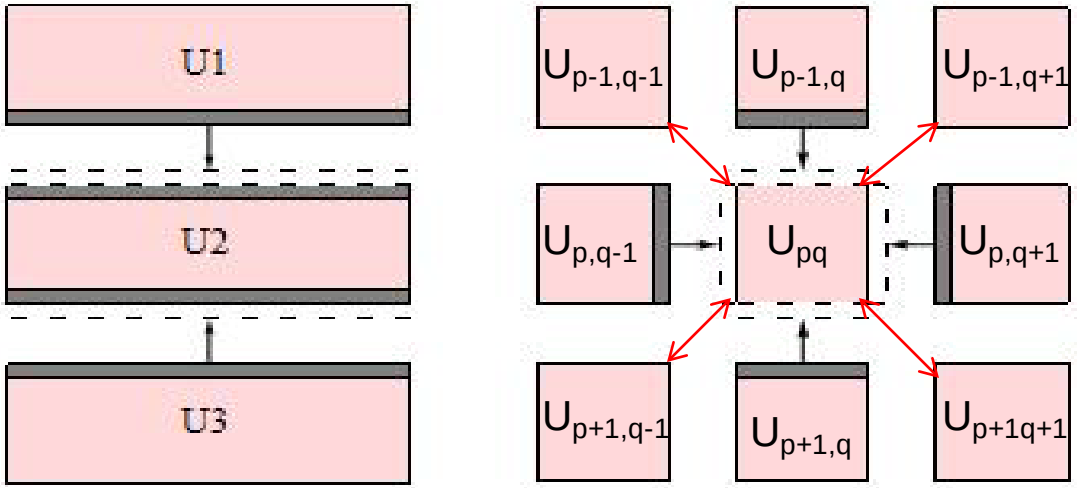
- Grid like partitioning: processor (p,q) communicates with 4 (8) neigbours

  size of data $c*(h/K_h+w/K_w)*NS/2$ $(+ (NS/2)^2)$

Remark:

$K_h*K_w = P$ (number of proc)

If P is large, the data size to be sent can be significantly smaller in the case of the grid like splitting



Parallel SFCM

Communication costs (at each iteration):

SFCM computation	MPI AllReduce (calls/ iteration)	MPIAll Reduce (data size/ call)	MPIAll Reduce (op.)	MPI Send/ Recv (data size/ iteration)
$\sum_i u'_{ij}\, x_i,\ j = \overline{1,c}$	1	c*d	Sum	-
$\sum_i u'_{ij},\ j = \overline{1,c}$	1	c	Sum	-
$\max_{i,j} \lvert u'_{ij}(k+1) - u'_{ij}(k) \rvert$	1	1	Max	-
$\sum_{k \in N(i)} u_{kj}$	-	-	-	$c*(h/K_h+w/K_w)*NS/2$

Environments

InfraGrid Cluster :

- Nodes: 16 nodes, 8 cores / node;
- CPU: Intel Xeon 2.0Ghz CPUs, 4 cores per CPU (64 bits mode);
- RAM: 1.25GB / core;
- High-speed interconnect: 40Gbps 4xQDR Infiniband (2.5 µs response time on MPI communication);

BlueGene/ P

• Nodes: 32 nodes x 32 compute cards x 1CPU

• CPU: 850Mhz PowerPC 450d, 4 cores per CPU (32 bits mode);
• RAM: 1GB / core;

• High-speed interconnect: 3D Torus 40Gbps bandwith (3µs response time on MPI communication)
• Collective interconnect: 40Gbps bandwith (5µs response time for MPI communication)

Test images

Landsat image: image of the Danube region

Source: Romanian Catalogue

Image size:

Width = 8786 pixels

Height = 7856 pixels

No. spectral bands = 4

Files total size = 270 Mb

Multispectral image: AVIRIS low altitude, high spatial resolution

Source: NASA Catalog (http://aviris.jpl.nasa.gov/html/aviris.freedata.html)

Image size:

Width = 614 pixels

Height = 1087 pixels

No. spectral bands = 224

Files total size = 294 Mb

Implementation

Parallel implementation:

- C, MPI (MPICH-2), libtiff (3.9.1)

- Communication between processors:
  - MPI_COMM_WORLD
  - Computation of the sums involved in the centroids formula (MPI_Allreduce, MPI_SUM)
  - Computation of the maximum used in the stopping condition (MPI_Allreduce, MPI_MAX)
  - Membership values transfer in SFCM (MPI_Send, MPI_Recv)

Implementation

Porting the code to BlueGene/P

Steps:

- acquiring the correct libraries
- building the libraries on BG/P
- compiling
- running the code

Particularities:

- IBM XL Compiler
- MPICH BlueGene/P version

Initial problems with the BG/P implementation

- loss of efficiency in the case of a large number of processors

Critical optimization aspects

- floating point usage
- vector routines optimization
- high order transformation module
- inter- procedure analysis

Optimization flags:

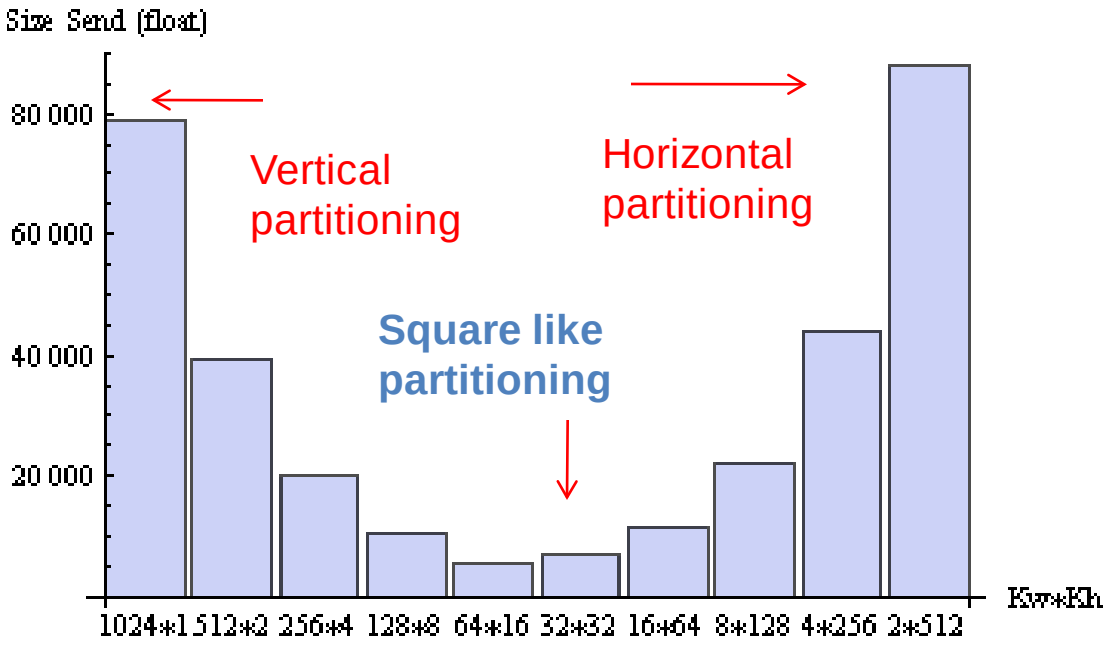
"-O3 -qhot -qipa=level=2 -qarch=450d"

(very significant gain in the processing time)

Image partitioning:

- influence on the load balancing and communication costs

Influence of the partitioning



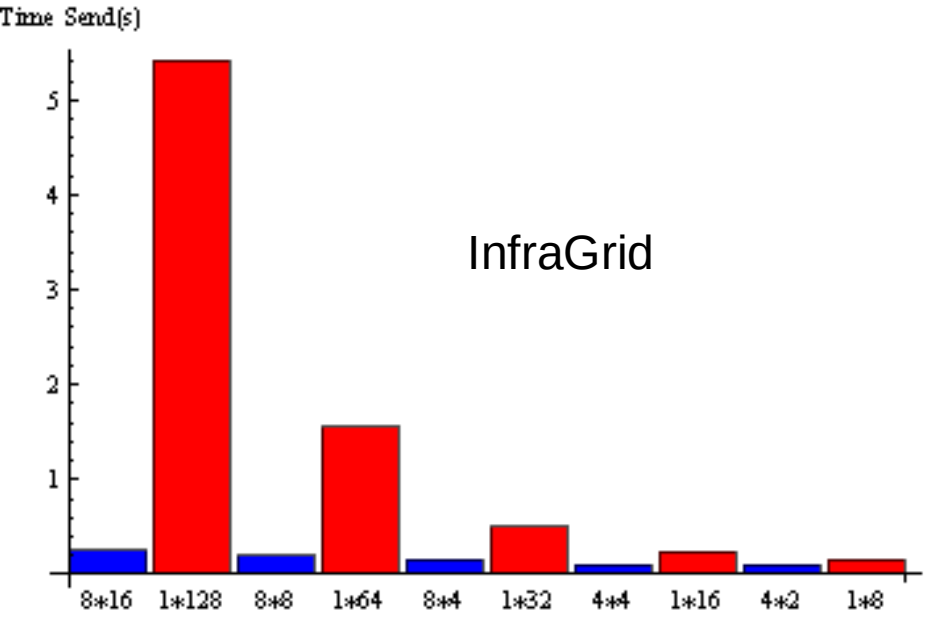
Test image: Landsat (8786x 7856 pixels)

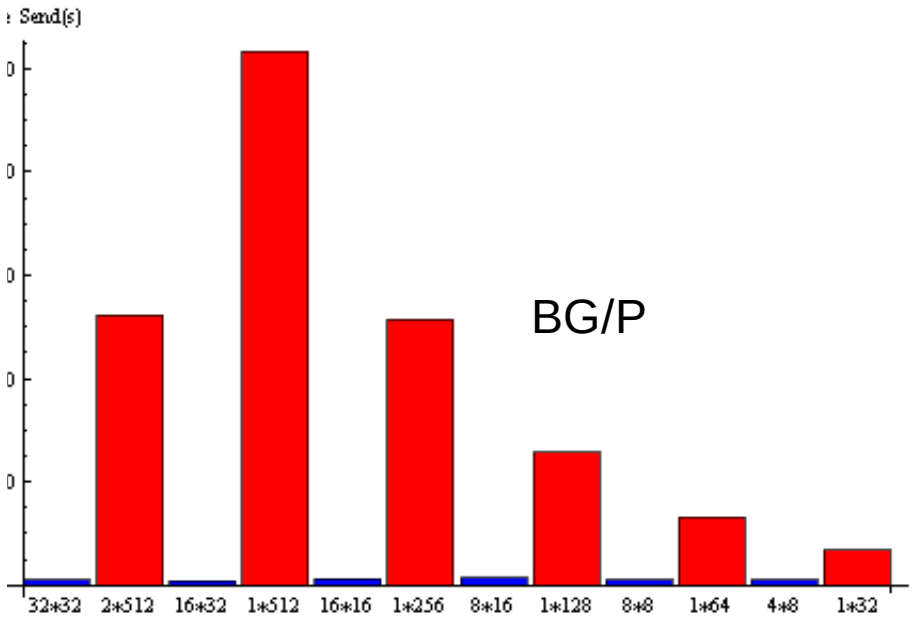
Algorithm: SFCM

Remark:

(almost) square like partitioning leads to a significantly smaller number of data transferred between processors than vertical (or horizontal) image partitioning

Influence of the partitioning

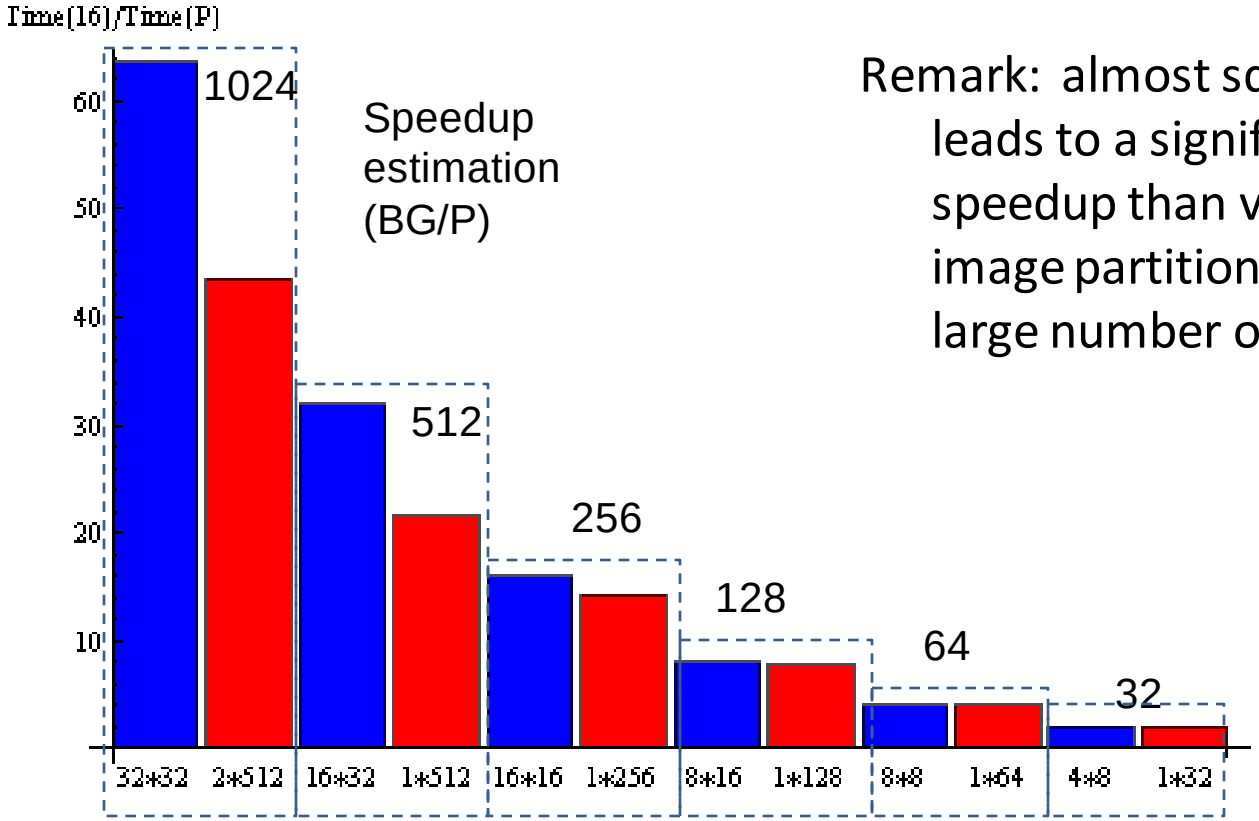




Remarks:

- square like partitioning leads to a significantly smaller time used by send operations (especially in the case of a large number of processors)
- the time of reduce operations is significantly smaller for a large number of processors

Influence of the partitioning

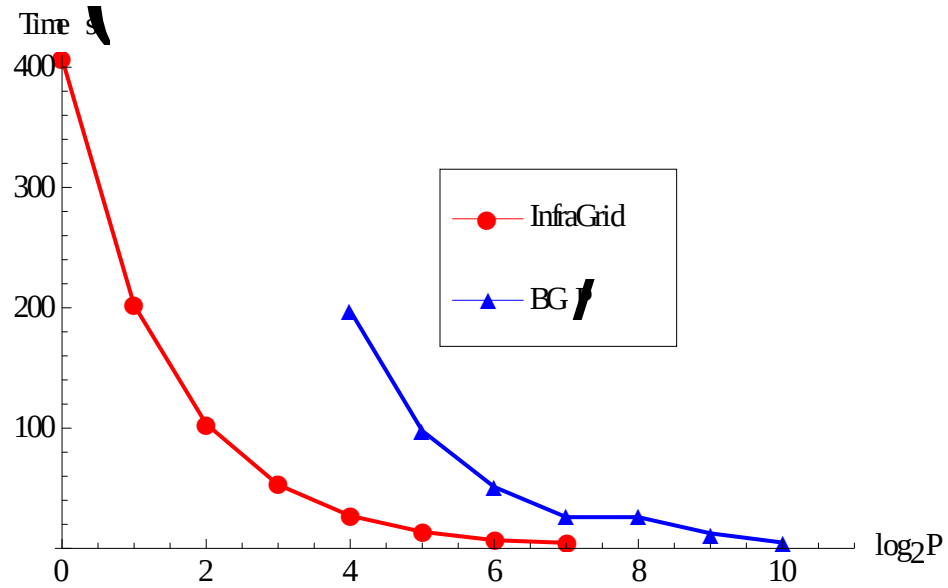


Remark: almost square like partitioning leads to a significantly better speedup than vertical (or horizontal) image partitioning in the case of a large number of processors

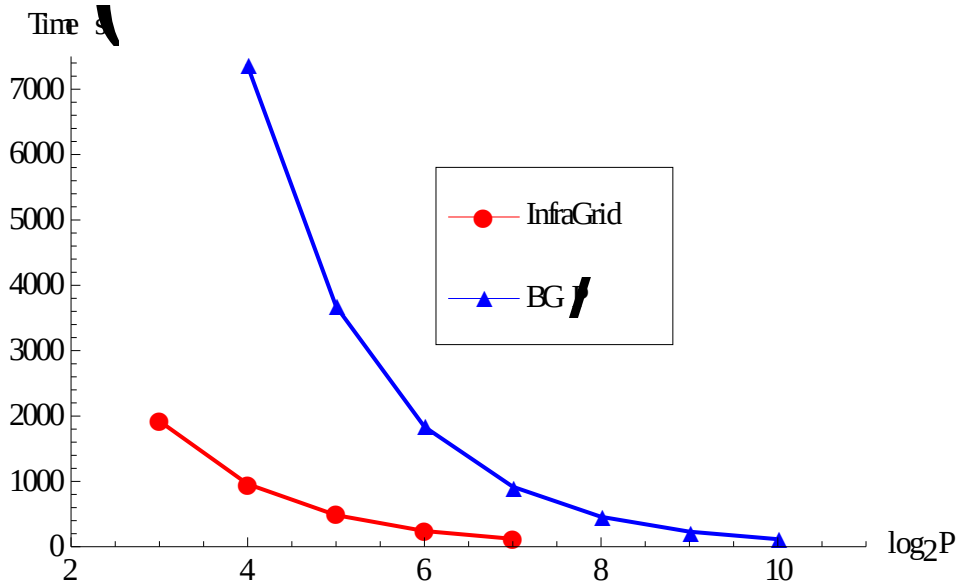
Test image: Landsat Algorithm:SFCM

Parameters: 100 iterations, 5 classes, neighborhood size=5

InfraGrid vs. BG/P



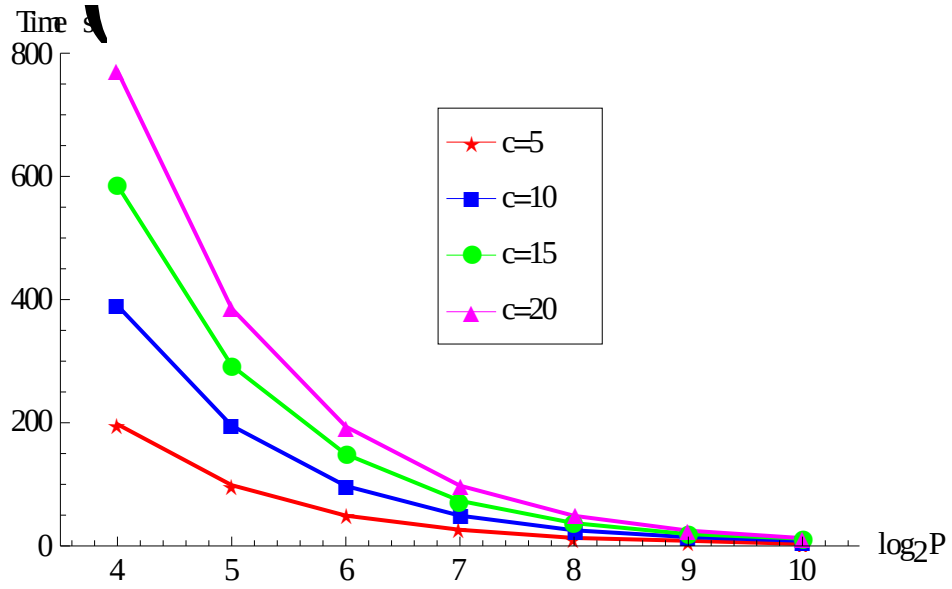
AVIRIS image (224 spectral bands)
614x1087 pixels)



Landsat image (4 spectral bands,
8786x7856 pixels)

Remark: the difference in running times mainly caused by the differences in the performance of the processors corresponding to the two infrastructures

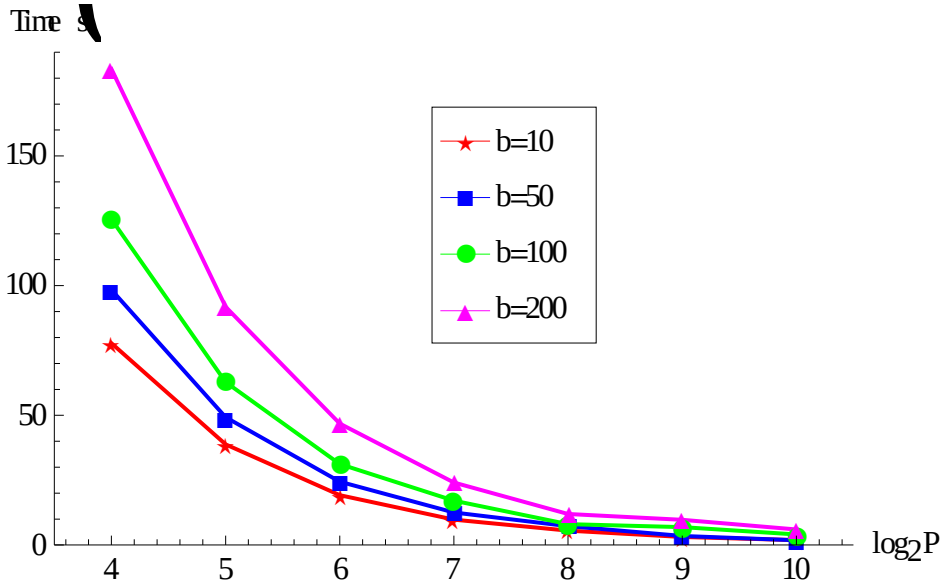
Scalability



Scalability with respect the number of classes

Algorithm: SFCM Image: Landsat

BG/P



Scalability with respect the number of spectral bands

Algorithm: SFCM Image: AVIRIS

Lessons learned

- Particular attention should be paid to image partitioning as it can have a significant influence on the efficiency of the parallel implementation
- Careful choice of optimization options is critical
- The platform (e.g. Infragrid vs BG/P) should be chosen according to the particularities of the problem/image
- Better speedup on BlueGene/P for:
  - large images with a smaller number of spectral bands

  than
  - smaller images but with a large number of spectral bands

Ongoing work

- Developing an application support for datasets slicing (to support larger images) using two approaches:
  - Physical dataset partitioning with slice pre-allocation for reading;
  - Based on the RDMA protocol for implementing virtual datasets partitioning.

- Porting a parallel algorithm for endmember extraction in order to deal with the spectral mixture problem in the case of hyperspectral data (under testing)