Computational Challenges in Processing Hyperspectral Images

Adina Toma, Silviu Panica, Daniela Zaharie, Dana Petcu

Department of Computer Science
West University of Timisoara, Romania
Motivation

Current status in remote sensing:
• Large hyperspectral images (containing hundreds of spectral bands) are available
• Need of tools to efficiently process large amount of data

Main challenges in processing large hyperspectral data:
• The image could be too large to be efficiently read and store on one computing node
• The image analysis could involve massive computations

Thus, an efficient exploitation of HPC particularities is required ...
Aim of this work

To address the computational challenges of parallel implementations of two clustering algorithms for hyperspectral images:

- a spatial variant of Fuzzy C-means (SFCM)
  - iterative process requiring collective computations and frequent transfer of data between computing nodes
- a morphological automated endmember extraction algorithm (AMEE)
  - requires the extraction of global endmembers from the local ones generated at each computing node and the computation of a similarity measure between a large number of endmembers

In the context of using

- BlueGene/P supercomputer
- 1024 quad-core, 850Mhz PowerPC 450d each with 1GB RAM
I/O challenges

- When reading and storing large amount of data on BlueGene/P one should take BG/P I/O System particularities
I/O challenges

Possible approaches:

a) All-read image distribution

b) One-to-All image distribution

c) One-to-All using shared memory
I/O challenges

All-read image distribution

• Simplest but less efficient

• Memory usage:
  – each processor has to load the entire image into memory and extract only one slice of it

• Network bandwidth usage:
  – each processor the entire image into memory -> the network bandwidth becomes soon inefficient
I/O challenges

One-to-all image distribution

• Minimizes the storage-memory data load

• Main particularities:
  – a subset of computing nodes are selected to be data distribution processors
  – the image slices are distributed over the computing nodes via the MPI communication subsystem
  – the number of data distribution nodes should be carefully selected in order to not induce high communication overhead

Data distribution models

- Torus network – 5.6GB/s per link
  - Compute node → Compute node → Compute node → Compute node

- Collective network – 1.7GB/s per link
  - I/O Node → I/O Node
  - external storage

- Torus network – 5.6GB/s per link
  - Compute node → Compute node → Compute node → Compute node

- Collective network – 1.7GB/s per link
  - I/O Node → I/O Node
  - external storage

- SHARED MEMORY
  - Torus network – 5.6GB/s per link
  - Compute node → Compute node → Compute node → Compute node

- Collective network – 1.7GB/s per link
  - I/O Node → external storage

RO-LCG, 27 October 2012
I/O challenges

One-to-all using shared memory

• Optimized version of one to all image distribution model

• Main particularities:
  – usage of global arrays toolkit for exposing a shared image such that the computing nodes can copy their corresponding slices
  – one processor read the image and populate a data structure in a shared memory
Computational Challenges

• **Case study** for parallel unsupervised classification (clustering) based on two iterative algorithms: SFCM and AMEE

• **Input data:**
  - Image = \{x_1, \ldots, x_n\}, n = number of pixels, \(x_i = (x_{i1}, \ldots, x_{id})\), d = number of spectral bands
    (set of vectors corresponding to all pixels and containing the values corresponding to the spectral bands)
  - Number of classes/endmembers to be identified (c)

• **Output data:**
  - **SFCM:** membership matrix (of size c x n): \(u_{ij}\) is a value in [0,1] specifying the degree of membership of pixel \(i\) to class \(j\); classes centroids = \(\{v_1, \ldots, v_c\}\)
  - **AMEE:** set of endmembers (of size c x n)
  - Classified image = \{y_1, \ldots, y_n\}, \(y_i\) = value related to the label of the class to which \(x_i\) belongs

RO-LCG, 27 October 2012
Parallelization Idea

- **Split the image in slices**
  - horizontal/vertical slices
  - rectangular slices
- **Process each slice and compute local info**
  - SFCM: local membership values
  - AMEE: local endmembers
- **Collect/transfer local information via**
  - collective operations (MPI AllReduce)
  - point to point transfer (MPI Send/Recv)
Spatial Fuzzy C-Means

**SFCM Algorithm** [Chuang, 2006]

- 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

**Centroids computation (global)**

\[
\mathbf{v}_j = \frac{\sum_{i=1}^{n} u_{ij}^m x_i}{\sum_{i=1}^{n} u_{ij}^m}, \quad j = 1, c
\]

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

**Membership values estimation (local)**

\[
w_{ij} = \frac{1}{\sum_{k=1}^{c} 1/\|x_i - \mathbf{v}_j\|^{2/(m-1)}}
\]

\[
u_{ij} = \frac{w_{ij}^p h_{ij}^q}{\sum_{k \in N(i)} w_{ik}^p h_{ik}^q}, \quad h_{ij} = \sum_{k \in N(i)} w_{kj}
\]

\(i = 1, n, j = 1, c\)
Parallel SFCM

• Split the image in slices: $S_1, \ldots, S_p$

• Split the computation:

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

• Processor $k$ computes:

  • The corresponding membership values (requires transfer of border values between processors dealing with neighboring slices)
  • 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

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

Algorithm 1 AMEE Parallel

\begin{algorithm}
\begin{algorithmic}
\State \textbf{Scatter} \( N \) \textbf{partial data structures} \( \{PSSP\}_{n=1}^{N} \) of \( F \)
\State \( i = 1 \)
\State \( MEI(x, y) = 0, \forall (x, y) \in PSSP_n \)
\While{\( i < I_{\text{max}} \)}
  \State Move kernel through each pixel
  \State Compute minimum and maximum
  \State Update MEI with SAM between minimum and maximum
  \State \( i = i + 1 \)
  \If{\( i = I_{\text{max}} \)}
    \State break
  \Else
    \State Replace \( PSSP_n \) with its dilation
  \EndIf
\EndWhile
\State \textbf{Select} \( P \) endmembers \textbf{with highest MEI scores}
\State \textbf{Master} gathers all endmembers \textbf{and forms a unique set of} \( P \) \textbf{endmembers by computing all possible pairs}
\end{algorithmic}
\end{algorithm}

\textbf{Notations:}

- PSSP = image slice
- MEI = matrix of eccentricity indices
- kernel = structuring element for the morphological operation
- SAM = spectral angle measure
- \( P \) = number of endmembers
Experiments: Implementation

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: 53Gbps bandwith (5μs response time for MPI communication)

Parallel implementation:

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

Communication between processors:
- MPI_COMM_WORLD
- MPI_AllReduce
- MPI_Send, MPI_Recv (SFCM)

Particularities:
- IBM XL Compiler
- MPICH BlueGene/P version
- optimization flags: “-O3 -qhot -qipa=level=2 -qarch=450d”
SFCM: influence of partitioning

Test image: AVIRIS Low Altitude (224 bands, 614x1097 pixels)

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

Remark: almost square like partitioning leads to a significantly better speedup than vertical (or horizontal) image partitioning
AMEE: optimization

Optimization elements:

• exploit the structure of spectral angle metric to optimize the paired distances between local endmembers
• avoid a global computation by a particular procedure to merge local sets of endmembers
• control the synchronization among processes in the context of using collective communications (MPIBarrier)

\[
s(e_i) = \sum_{j=1}^{P \cdot E} \left( \left( \sum_{k=1}^{B} e_i^k \cdot e_j^k \right) l(\|e_i\| \cdot \|e_j\|) \right)
\]

\[
s(e_i) = \frac{1}{\|e_i\|} \sum_{k=1}^{B} e_i^k \sum_{j=1}^{P \cdot E} e_j^k
\]

Test image: AVIRIS Cuprite (224 bands, 614x2206 pixels)
Comparative results

**Test image:** AVIRIS Cuprite (224 bands, 614x2206 pixels)

**Remarks:**
- computational costs of SFCM higher than for AMEE
- better efficiency for parallel SFCM than for parallel AMEE
Comparative results

Test image: AVIRIS Cuprite (224 bands, 614x2206 pixels)
Left: original image
Middle & right: results after 50 iterations

Clustering quality index:
(V$_K$ = Kwon index)
- smaller values mean better clustering
- SFCM leads to a better clustering

(a) AMEE, $V_K = 51.9$
(b) SFCM, $V_K = 13.04$
Conclusions

Efficient parallel implementations hyperspectral images processing algorithms requires:

• exploitation of I/O system particularities
• careful division of the image in order to minimize the point to point communication costs
• optimized usage of collective operations

Choosing an appropriate classification algorithm usually leads to:

• trade-off between costs and the classification quality