MRF Satellite Image Classification on GPU

-

UCAA 2012

-

Pedro Valero-Lara
Unidad de Simulación, CIEMAT
Facultad de Informática, UCM

Outline

- Introduction
- Image Classification
- Classification using Markov Random Fields
- Use GPUs for Parallel Computing
- Why GPU?
- Parallel Implementation
- Performance Analysis
- Conclusions and Future Work

Introduction

- Reduce the execution time for satellite image classification
- Software Platform
  - Orfeo ToolBox
    http://www.orfeo-toolbox.org/otb/
  - OpenSource (CeCILL)
  - CMRF
- Hardware Platform
  - GPU
  - GTX 285
  - CUDA

Image Classification

Is the problem of classifying pixels into several homogeneous regions

- Features vector (v1 , v2 , . . . , vn) describes the image in terms of several n attributes (classes)
- Consists in organizing data into categories (classes)
- This process is an important and fundamental problem in image pattern analysis applied to a large number of fields (satellite image)

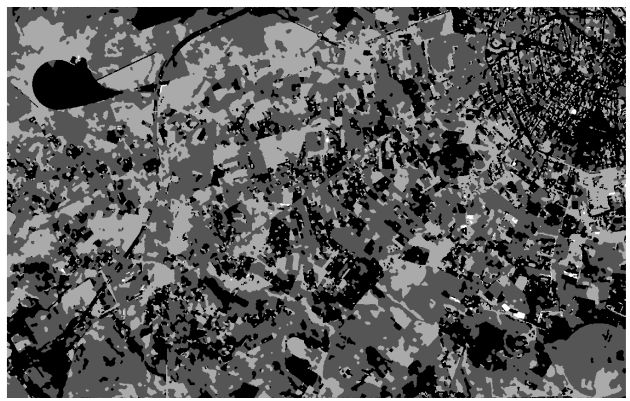
Classification using Markov Random Fields (CMRF)

Refine an initial classification by introducing the spatial coherence

Two images as input

1. Satellite image
2. Classified image (mean,max-min,k-means,...)

The output is the second image refined



Steps of CMRF

The images are divided in windows

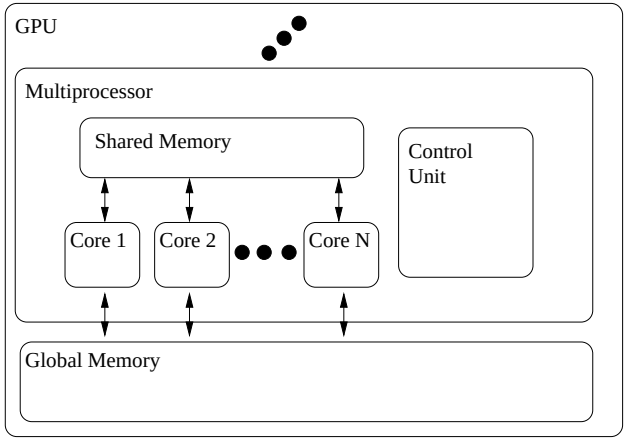
All steps are carried out on all windows

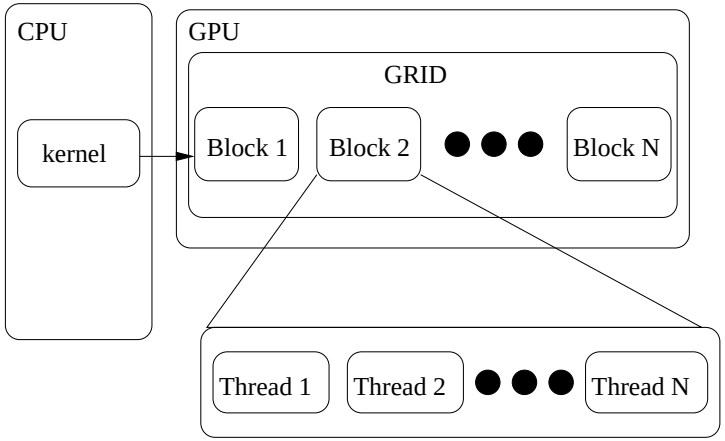
1. Influence of the classes on central pixel (satellite image)
2. Influence of the classes of all the pixels around central one (classified image)
3. Mahalanobis distance of the central pixel
4. New class for the central pixel?
5. Continue or stop?

Steps of CMRF

1. for i = 0, 1, . . . , NumClasses
   ClassInf[i] = $\sqrt{(ValuePixelCentral - Classes[i])^2}$
2. for i = 0, 1, . . . , WindowSize
   LabInf[$LabelPixel_i$] += Weight[i]
3. for i = 0, 1, . . . , NumClasses{
   DisMa[i] = LabInf[i] - ClassInf[i]}
4. Max(DisMa) = NewLabel
   if (LabelPixelCentral $\neq$ NewLabel)
   LabelPixelCentral = NewLabel
   Condition[WindowNumber] = 1
   else
   Condition[WindowNumber] = 0
5. if (NumIter $<$ MaxIter AND Changes) stop
   else continue

GPU for Parallel Computing





New architecture (many-core)

- Multiprocessor, global and shared memory

CUDA

- Kernel, grid, blocks and warp of threads

GPU for Parallel Computing

Memory Management

- Overhead in the transfers between memories
- Global Memory
  - High latency
  - An impressive impact on performance
  - Similar or equal pattern of memory accesses
  - Coalescing accesses
  - Overlap the blocks threads execution with memory accesses (higher number of blocks than multiprocessor)
- Shared Memory
  - 100 $\times$ faster
  - Load data from global memory
  - Several thread access to same data
  - Data are used many times by one thread

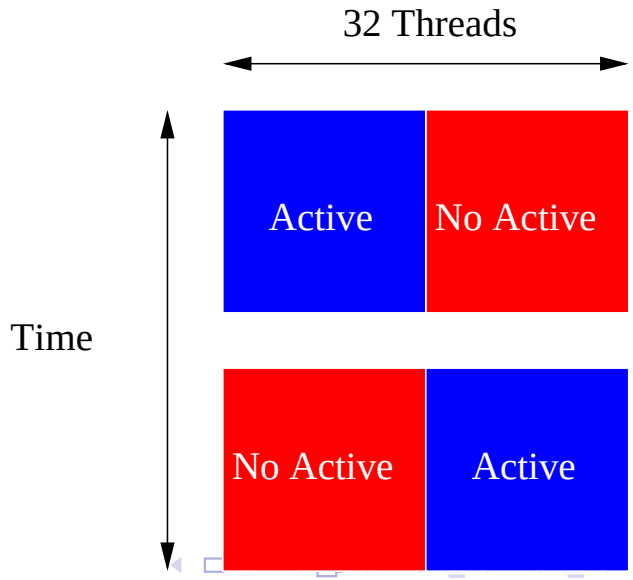
GPU for Parallel Computing

Divergence

- Threads of the same warp share the same path
- Threads of a warp diverge via a data-dependent conditional branch
- Threads which satisfy the condition are executed, the rest are stopped
- A high level of divergence decreases the benefit of using GPU computing

Example Warp Code

```
if thread.id < 16
        2+2
else
        3+3
endif
```



Why GPU?

GPU

- High ratio performance/cost
- Energy-efficient design
- Free software support (CUDA,OpenCL)

Classification using Markov Random Fields (CMRF)

- High data-parallelism
  - Independence between windows
- Memory management very suitable
  - Efficient use of Global and Shared Memory
- Low requirements for memory transfers

Parallel Implementation

The order of the steps has to be respected (one kernel per step)

Pseudocode of GPU-CMRF algorithm

MRFImageFilterKernel(ISatellite,ILabel,Classes,ConVar,Window)

```
Transfer CPU → GPU ISatellite
Transfer CPU → GPU ILabel
Step 1 (GPU)
While (iter < MaxIter and Changes)
    Step 2 (GPU)
    Step 3 (GPU)
    Step 4 (GPU)
    Transfer GPU → CPU ConVar
    Step 5 (CPU)
EndWhile
Transfer GPU → CPU ILabel
```

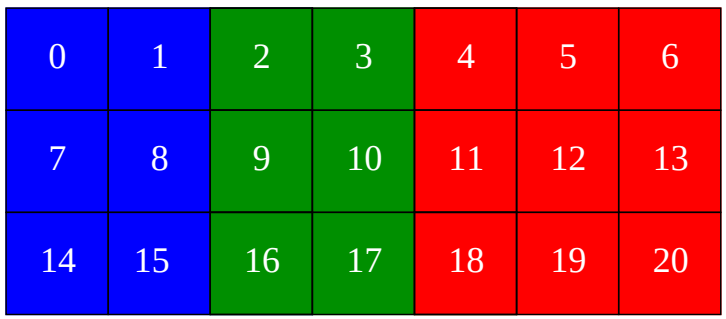
Parallel Implementation

Global Memory

Accesses to images

Example: 3 windows with a size 3×3

- Threads share the same pattern for the accesses to memory
- Accesses to the same memory segment

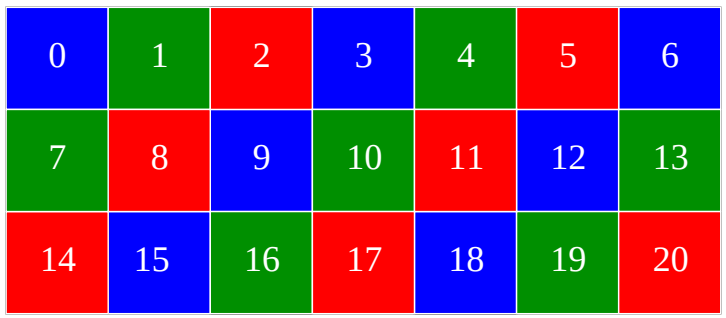


Parallel Implementation

Global Memory

Accesses to auxiliar variables (ClassInf, LabInf, MahDis)

- Threads access to contiguous spaces of memory (coalescing accesses)



Parallel Implementation

Shared Memory

- Weight and Classes are common in all windows
- All threads have to access to the same memory positions

Divergence

- Only the 4^{th} step present cases of divergences
- The rest of steps all threads have the same path

Performance Analysis

2 different images

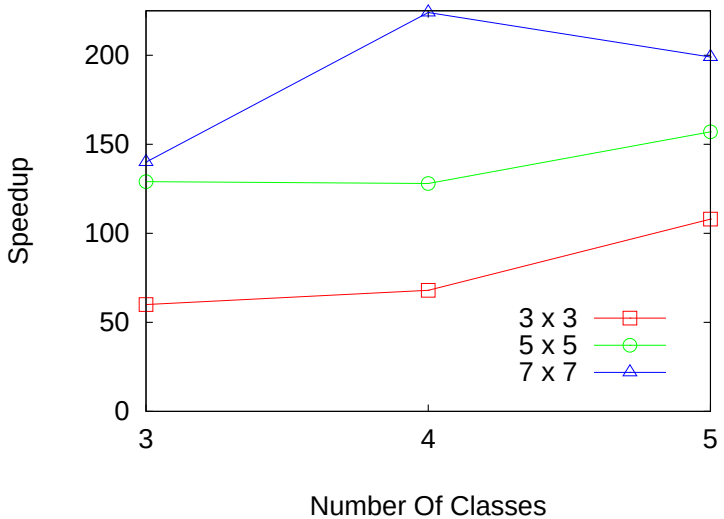
- Resolution of satellite images: 5.6 meters
- Image Size: 1657×849 pixels
- Spatial Reference: 0629_2-4 and 0790_4-1
- Number of classes: 3,4 and 5

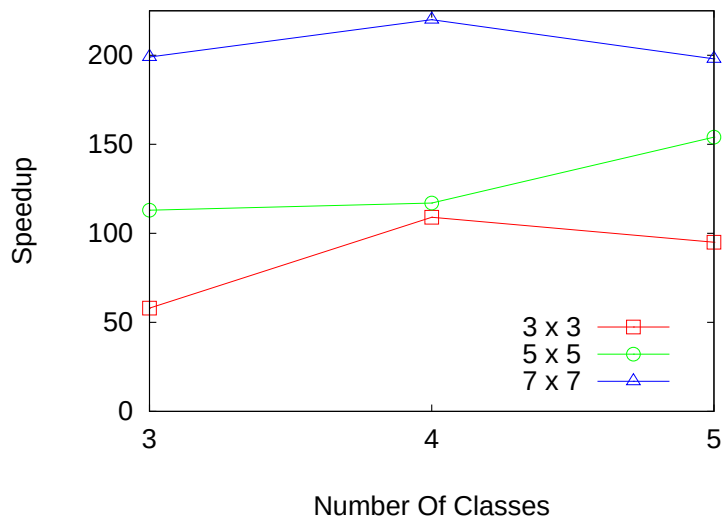
Platform

- CPU: Intel Core 2 Quad at 2.66GHz and 4GB
- GPU: GTX 285 with 240 cores and 1 GB
- CUDA version 4.0 (nvcc)

Performance Analysis

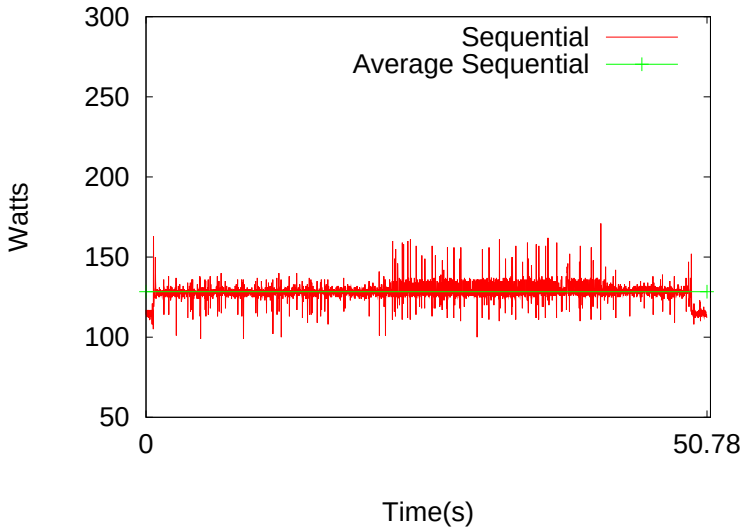
Speedup

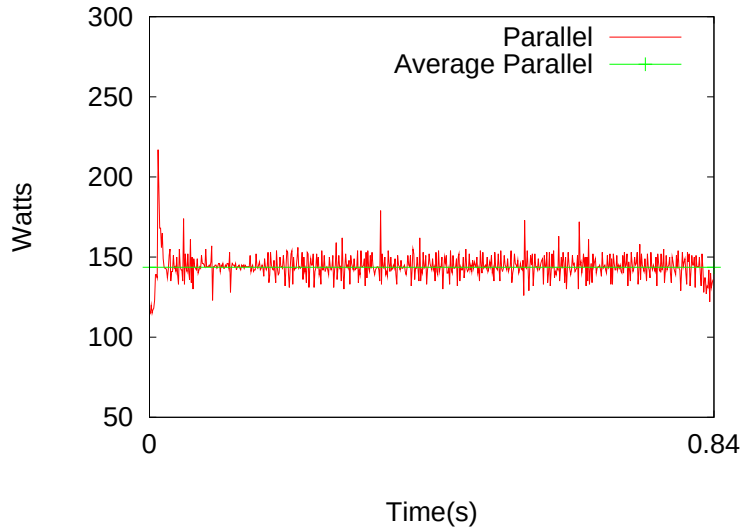


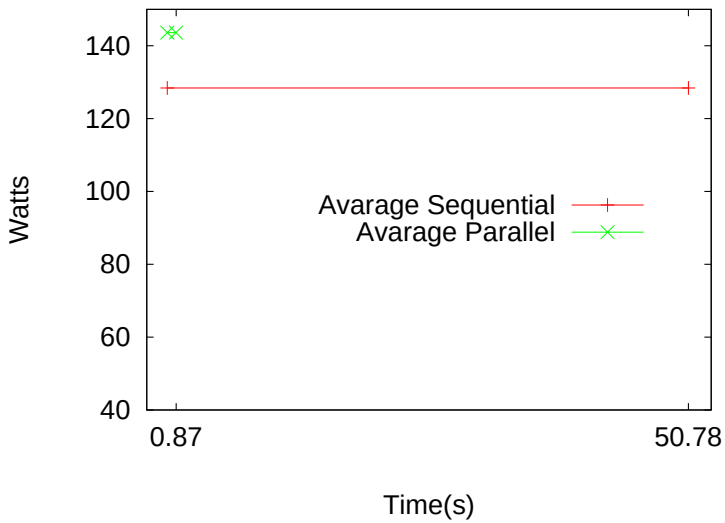


Performance Analysis

Power Consumption







Conclusion and Future Works

Conclusions

- It is achieved high reduction in terms of:
  - Execution time is achieved
  - Power consumption
- Efficient use of GPU to resolve CMRF:
  - Global and Shared memory
  - Low Divergence cases
- GPU-based hardware accelerators are a very suitable platform to tackle this problem

Future Works

- Continue implementing other algorithm
- Check new parallel platforms, Multi-GPU, Heterogeneous Cluster, Heterogeneous architectures on chip, ...

THANKS FOR YOUR ATTENTION!!!

MRF Satellite Image Classification on GPU

-

UCAA 2012

-

Pedro Valero-Lara
Unidad de Simulación, CIEMAT
Facultad de Informática, UCM