

Application of Heterogeneous Parallel Computing to EO and Remote Sensing

Antonio Plaza, David Valencia, Javier Plaza & Pablo Martínez
Department of Technology of Computers and Communications
Computer Science Department, University of Extremadura
Contact e-mail: aplaza@unex.es
URL: <http://www.umbc.edu/rssipl/people/aplaza>

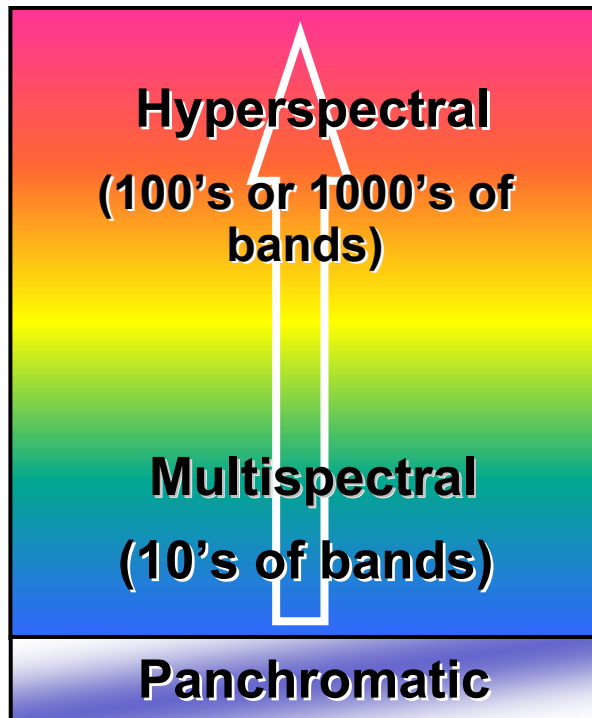
Talk outline

- Introduction to EO & remote sensing
- Detection algorithms
- Classification algorithms
- Heterogeneous implementations
- Use of HeteroMPI
- Conclusions
- Future lines



Levels of information in EO & RS

- Remote sensing technology has evolved from panchromatic and multispectral data, with only a few bands, to hyperspectral imagery with hundreds of bands.
- The evolution in sensor technology has introduced changes in algorithm design:



Quantification: Determines the abundance of materials (e.g. chemical/biological).

Identification: Determines the unique identity of the foregoing generic categories (i.e. material identification).

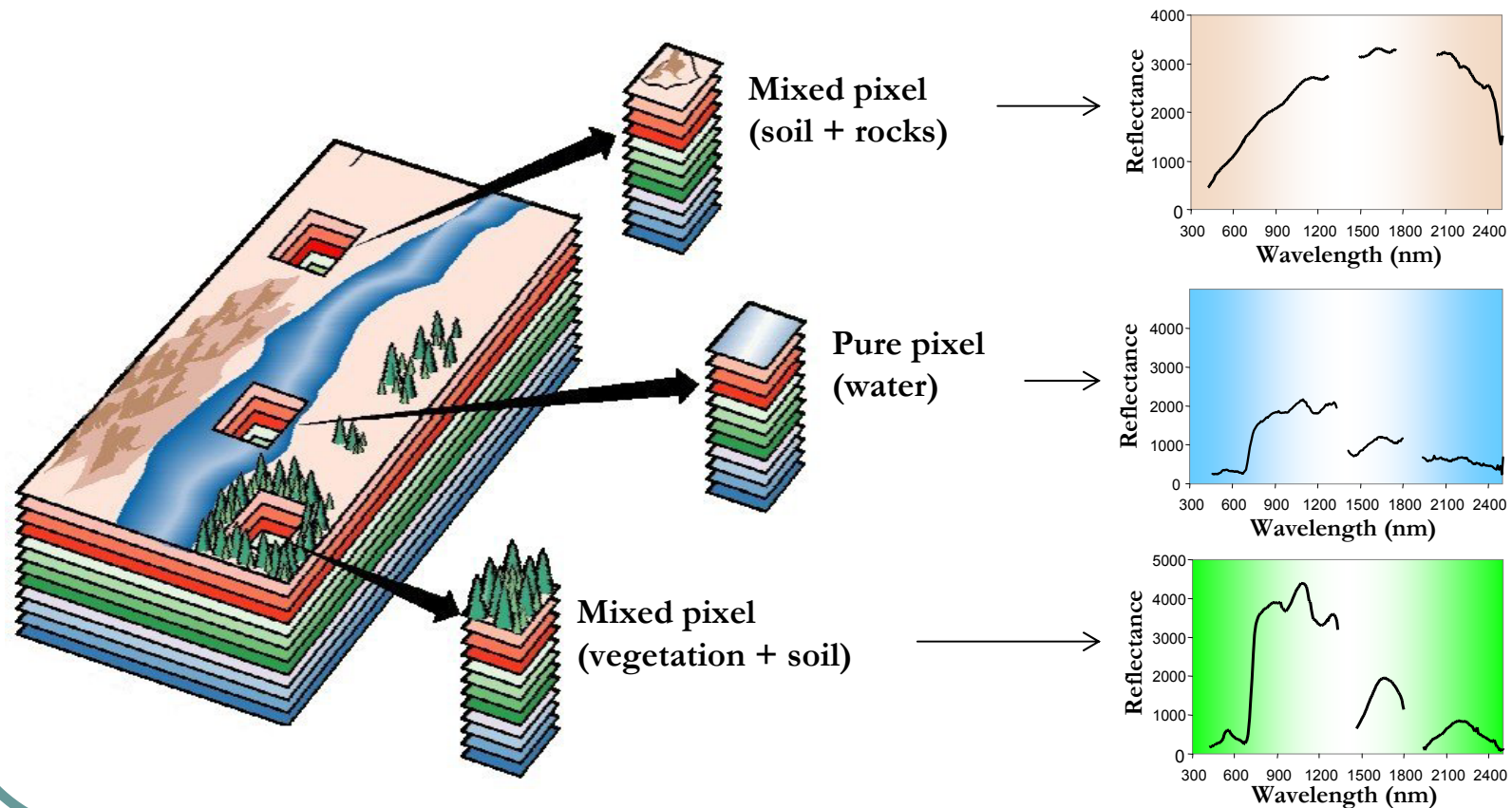
Discrimination: Determines generic categories of the foregoing classes.

Classification: Separates materials into spectrally similar groups.

Detection: Determines the presence of materials, objects, activities, or events.

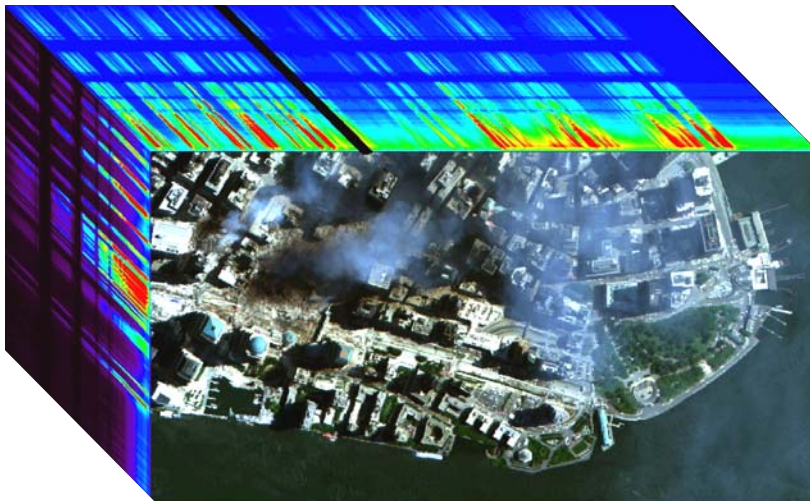
Hyperspectral imaging concept

- One of the most relevant problems is the presence of mixed pixels (in which several substances may be present at sub-pixel levels).



Hyperspectral applications

- Hyperspectral image processing algorithms are very expensive in computational terms.
- High computing performance is essential in many applications (environmental monitoring, fire tracking, chemical and biological detection, target detection in military applications, etc.)



AVIRIS scene over New York WTC



Debris and dust map (USGS)

Why heterogeneous computing?

Problems:

High computational complexities in data processing algorithms.

Large amounts of collected hyperspectral data sets are never used:

Analyses and information mining should be conducted in reasonable processing times.

Results might allow for the extraction of relevant knowledge (e.g. spectral libraries, etc.).

Solutions:

High-performance computers at low cost.

Commodity computers made up of off-the-shelf, low-cost computing components.

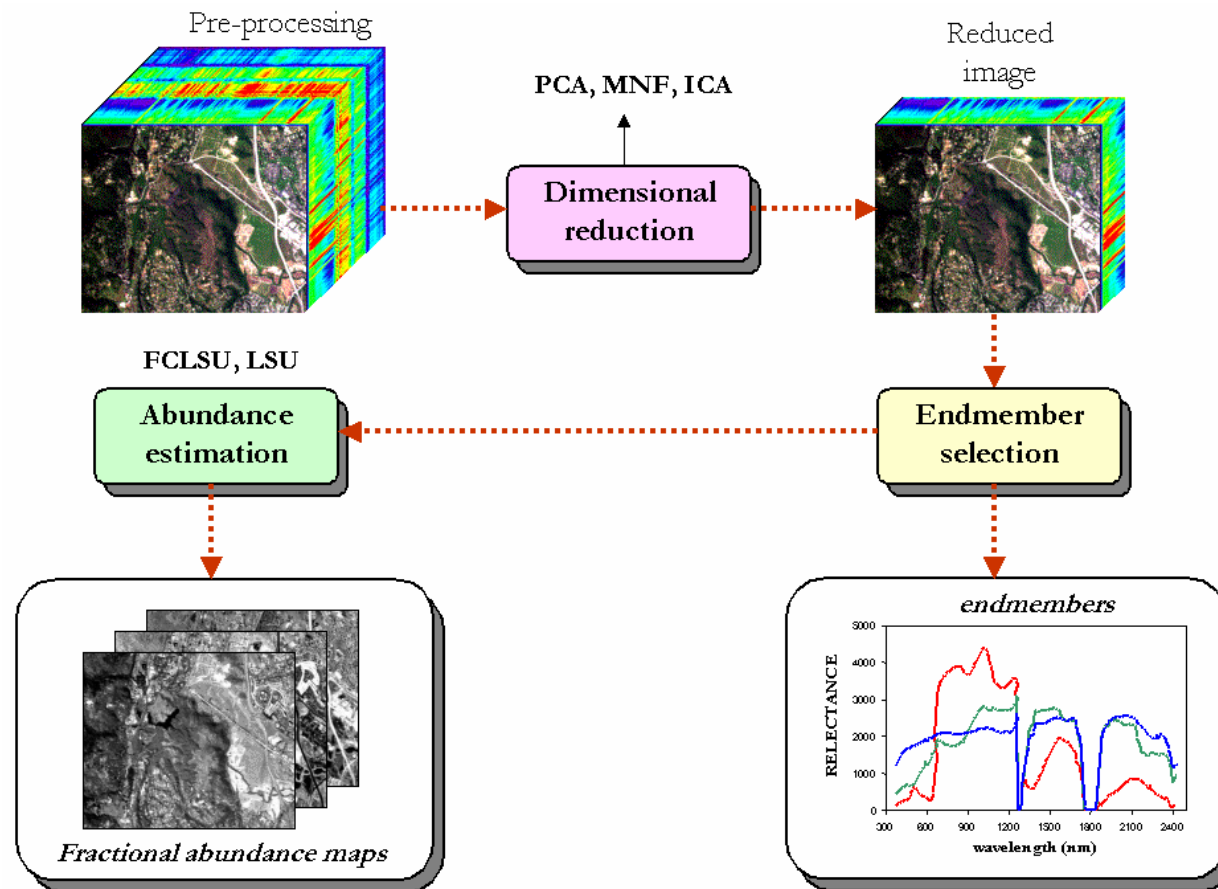
Networks of workstations interconnecting distributed platforms (Grid computing).

Applications:

Data mining and information extraction from large data repositories.

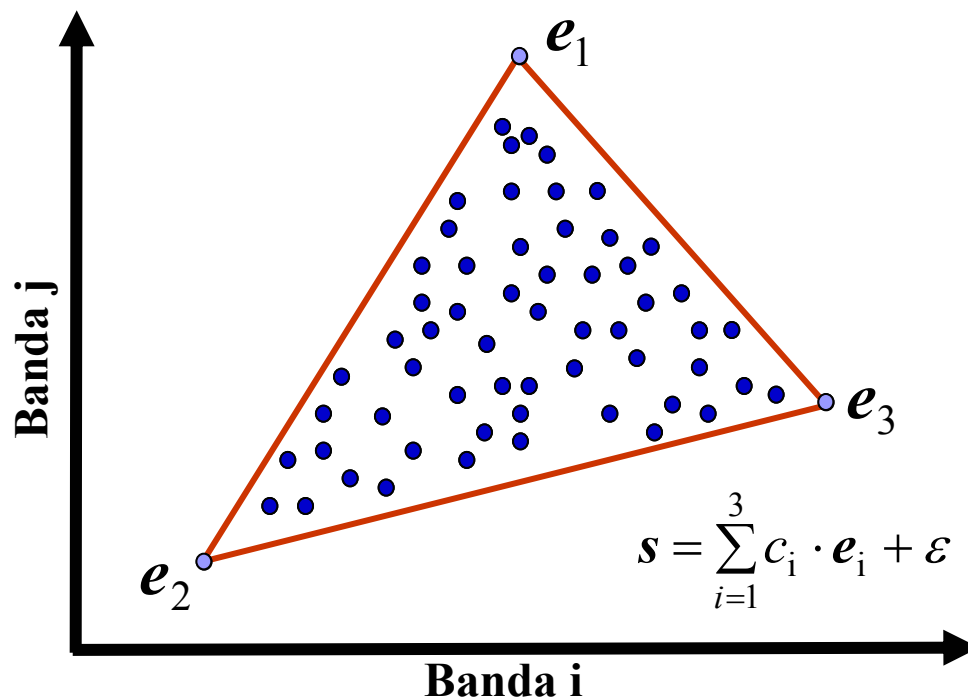
Classic analysis methodology

- The standard analysis methodology relies on the following steps:



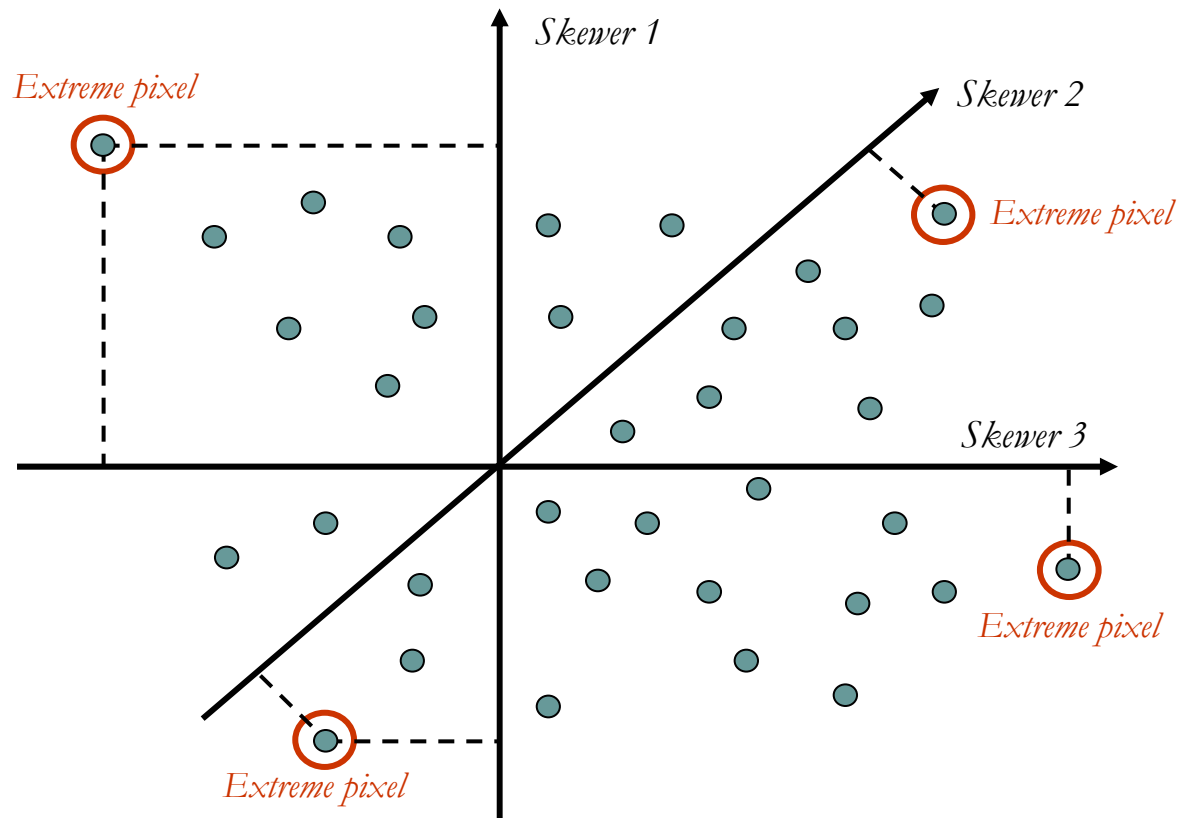
Detection algorithms

- One of the most robust sub-pixel analysis techniques consists of extracting extreme “pure” pixels (endmembers) and then model mixed pixels as combinations of pure spectral signatures:



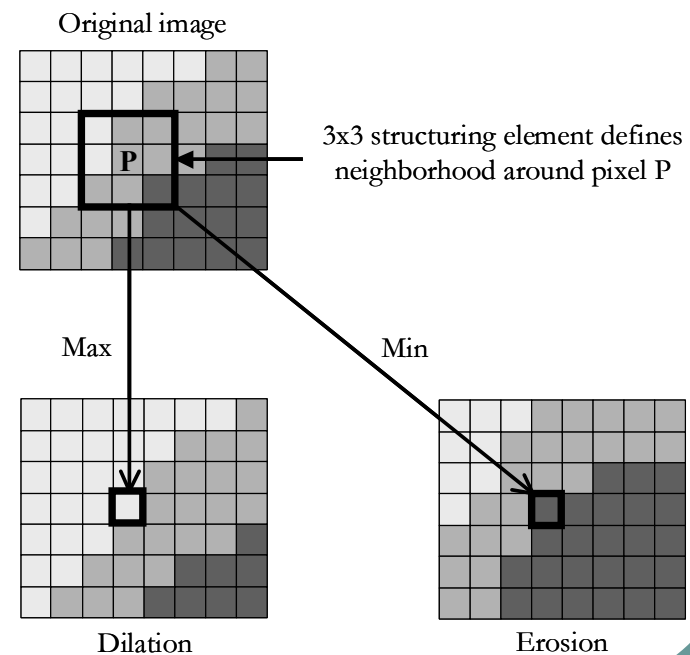
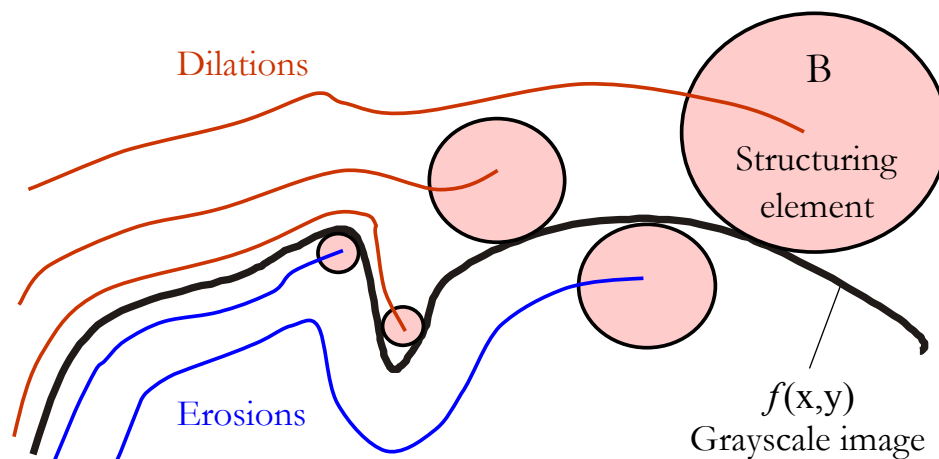
Pixel purity index (PPI)

- The PPI is one of the most popular endmember detection algorithms (available in Kodak's Research Systems ENVI software):



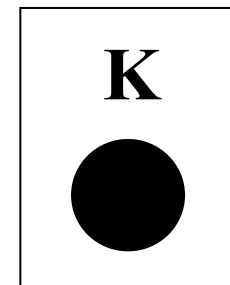
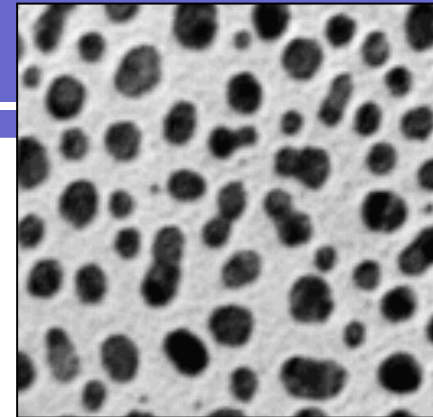
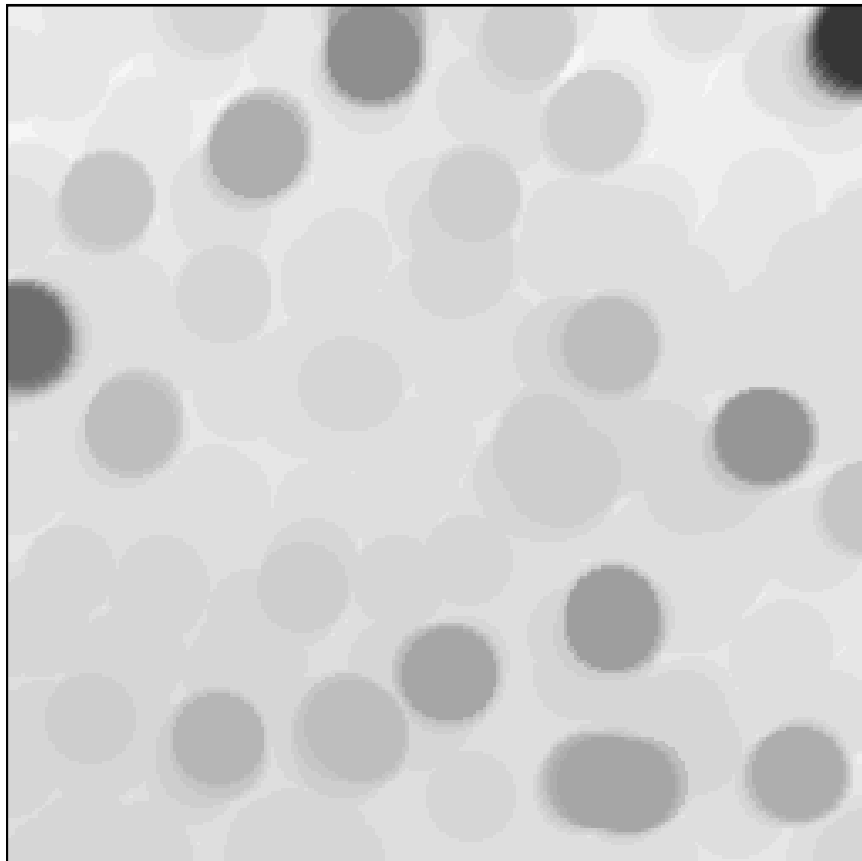
Morphological classification

- Mathematical morphology is a very well-consolidated technique in the spatial domain that can be extended to the spectral domain.
- It relies on a (partial) ordering relationship between the pixels of the image, and the application of a so-called structuring element:



Morphological filtering

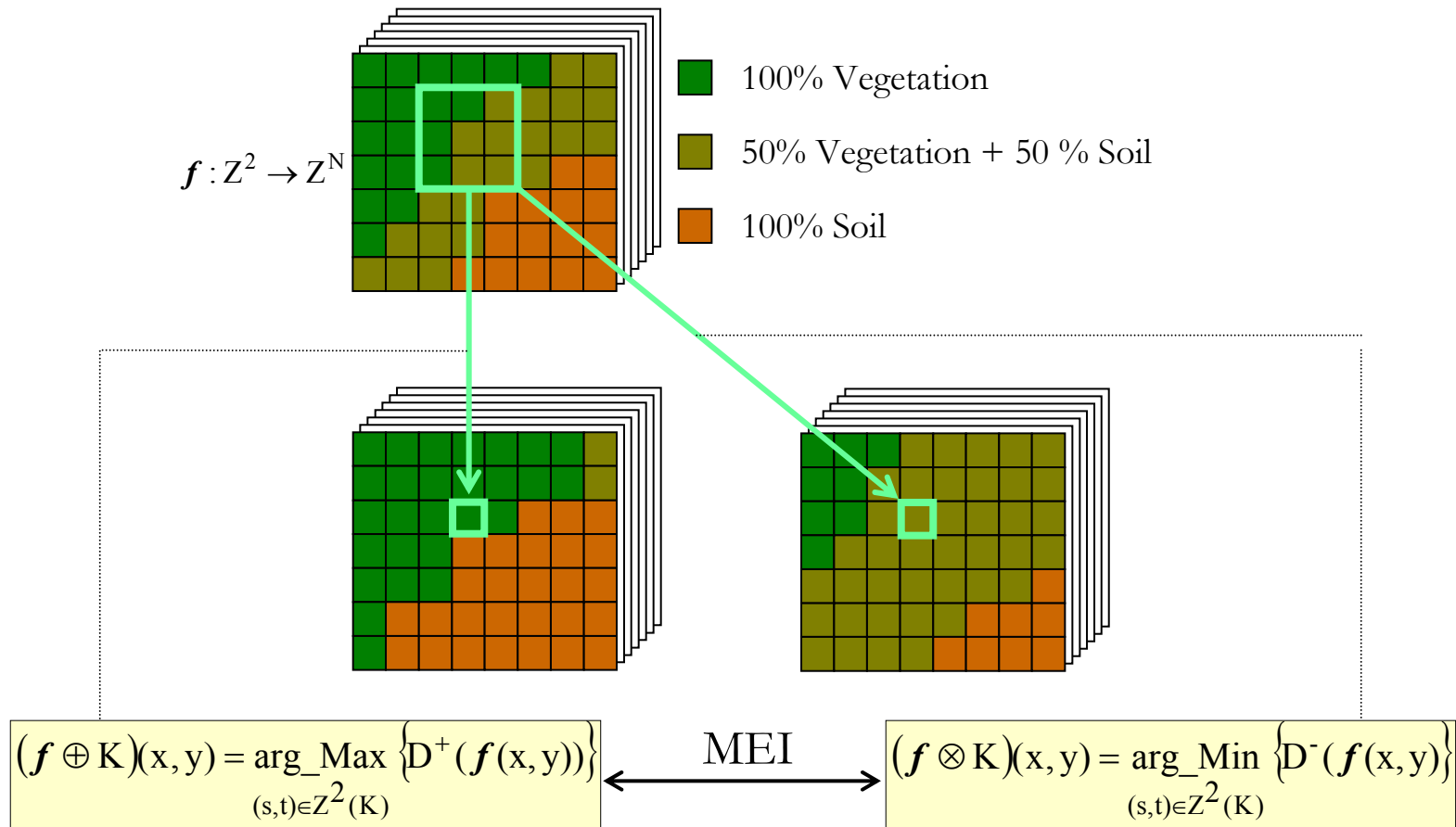
Morphological opening (erosion + dilation)



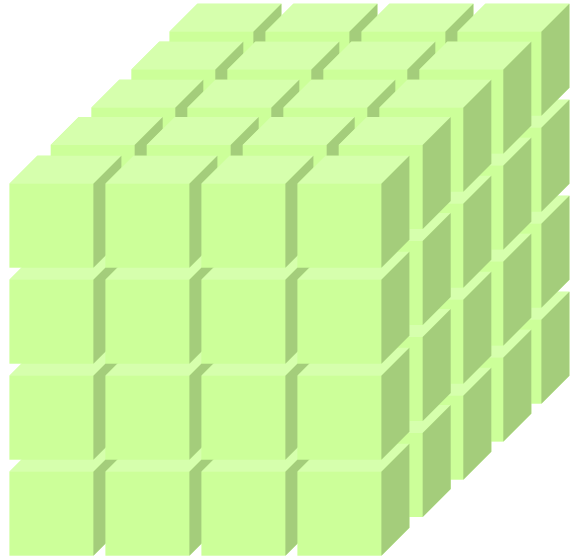
*Structuring
element*

Extended math morphology

- Extended mathematical morphology allows for spatial/spectral integration:



Data partitioning strategies

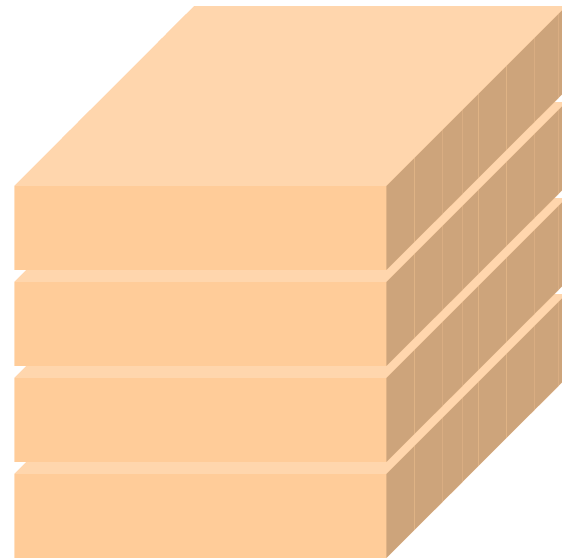


Spectral-domain partitioning:

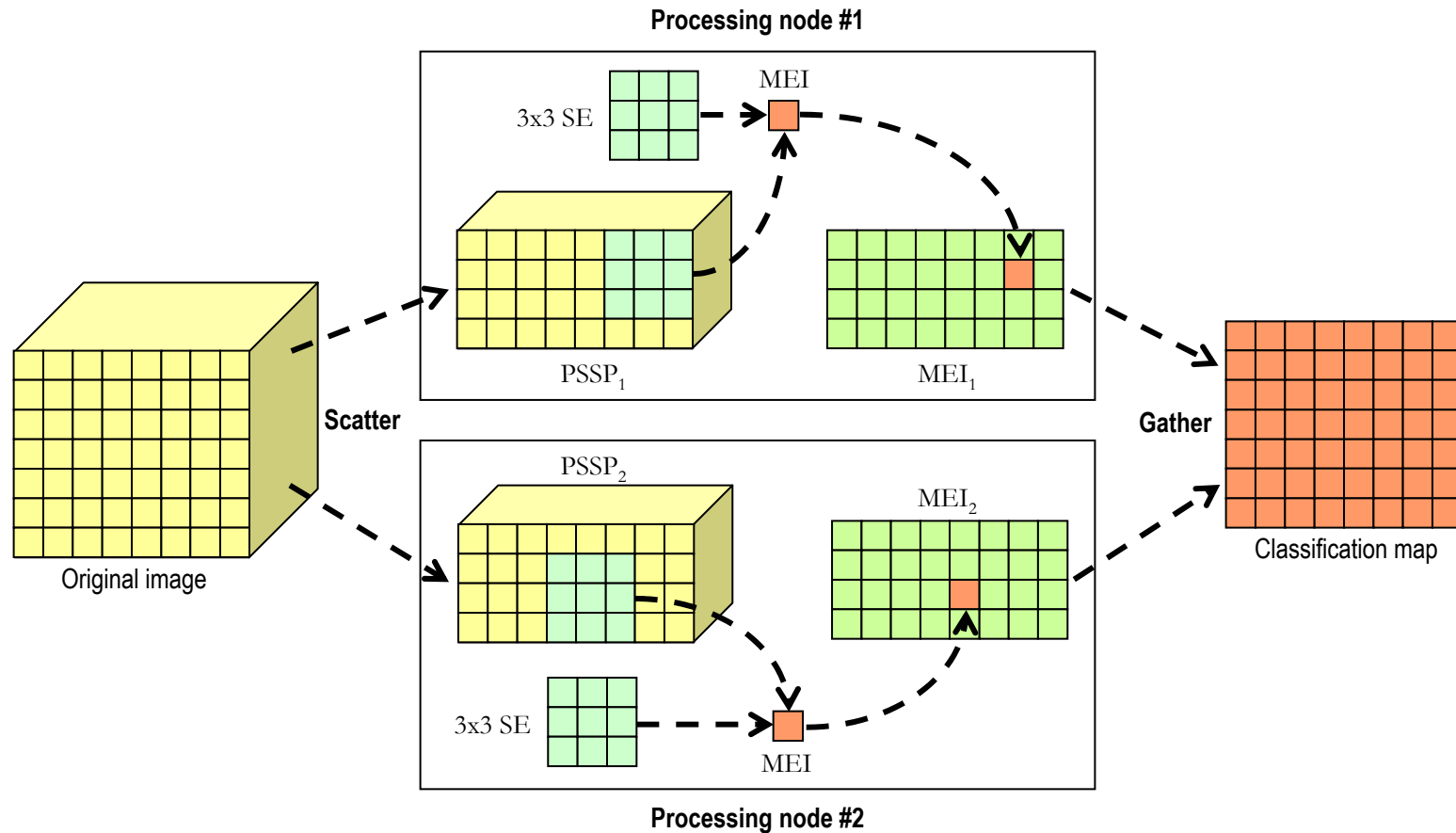
A single pixel vector (spectral signature) may be stored in different processing units and communications would be required for individual pixel-based calculations such as those in the PPI algorithm.

Spatial-domain partitioning:

A pixel vector (spectral signature) is always stored in the same processing unit. As a result, the entire spectral signature of each hyperspectral image pixel is never partitioned, thus reducing the cost of inter-processor communications.



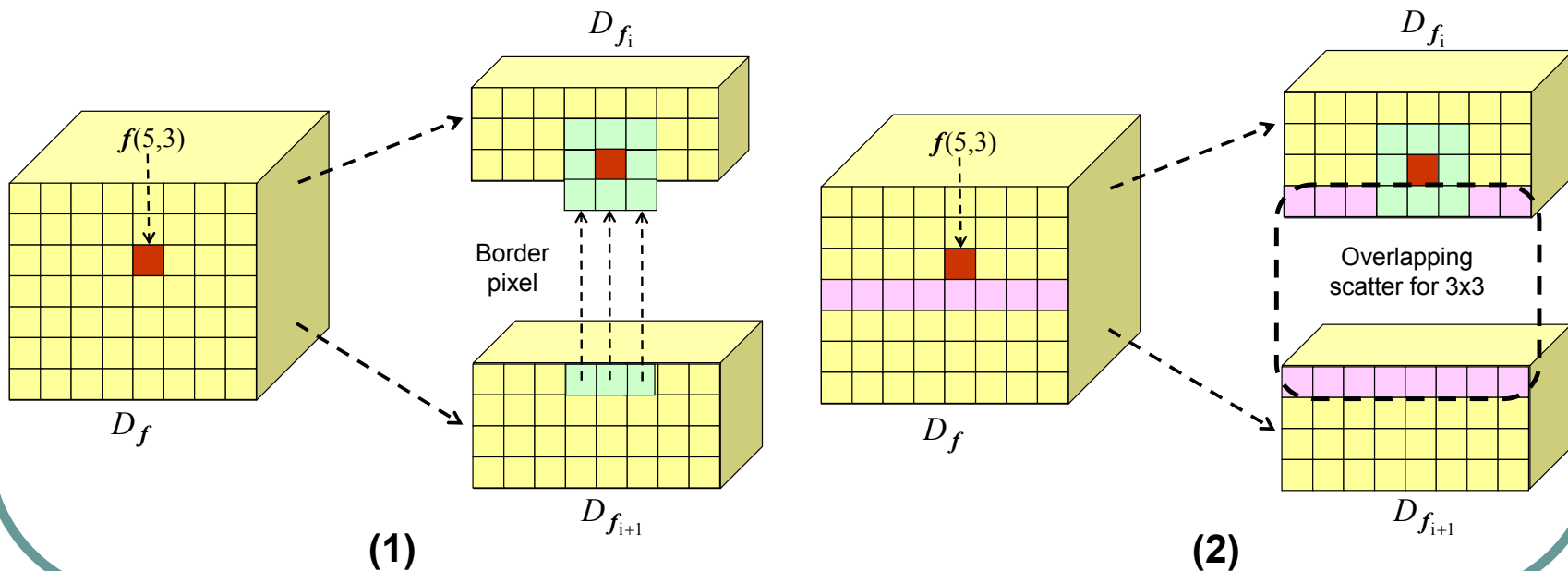
Spatial-domain partitioning



Parallel implementation of MM

Handling communications:

- (1) Need for communication when the structuring element is centered around a border pixel of a local partition.
- (2) Overlapping scatter allows one to reduce the cost introduced by communications for small structuring element sizes (the proposed classification algorithm is based on a constant, 3x3 structuring element)



Definition of benchmark function

Definition of a performance model for the morphological processing algorithm (mpC):

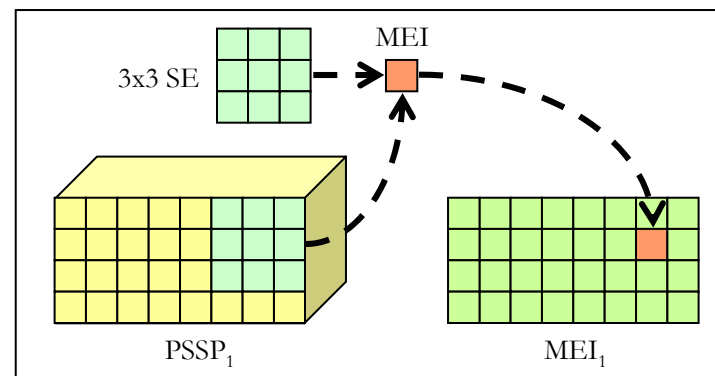
```
algorithm MM_perf (int m, int n, int se_size, int iter, int p, int q, int partition_size[p*q]) {  
  coord I = p, J = q;  
  node { I >= 0 && J >= 0: benchmark * ((partition_size[I*q+J]*iter); };  
  parent[0,0];  
}
```

- Parameter **m** specifies the number samples of the data cube.
- Parameter **n** specifies the number of lines.
- Parameters **se_size** and **iter** respectively denote the size of the SE and the number of iterations executed by the algorithm.
- Parameters **p** and **q** indicate the dimensions of the computational grid (in columns and rows, respectively), which are used to map the spatial coordinates of the individual processors within the processor grid layout.
- Finally, parameter **partition_size** is an array that indicates the size of the local PSSPs (calculated automatically using the relative estimated computing power of the heterogeneous processors using the **benchmark** function).

Heterogeneous implementation

The **benchmark** function should be representative of the application and computationally light.

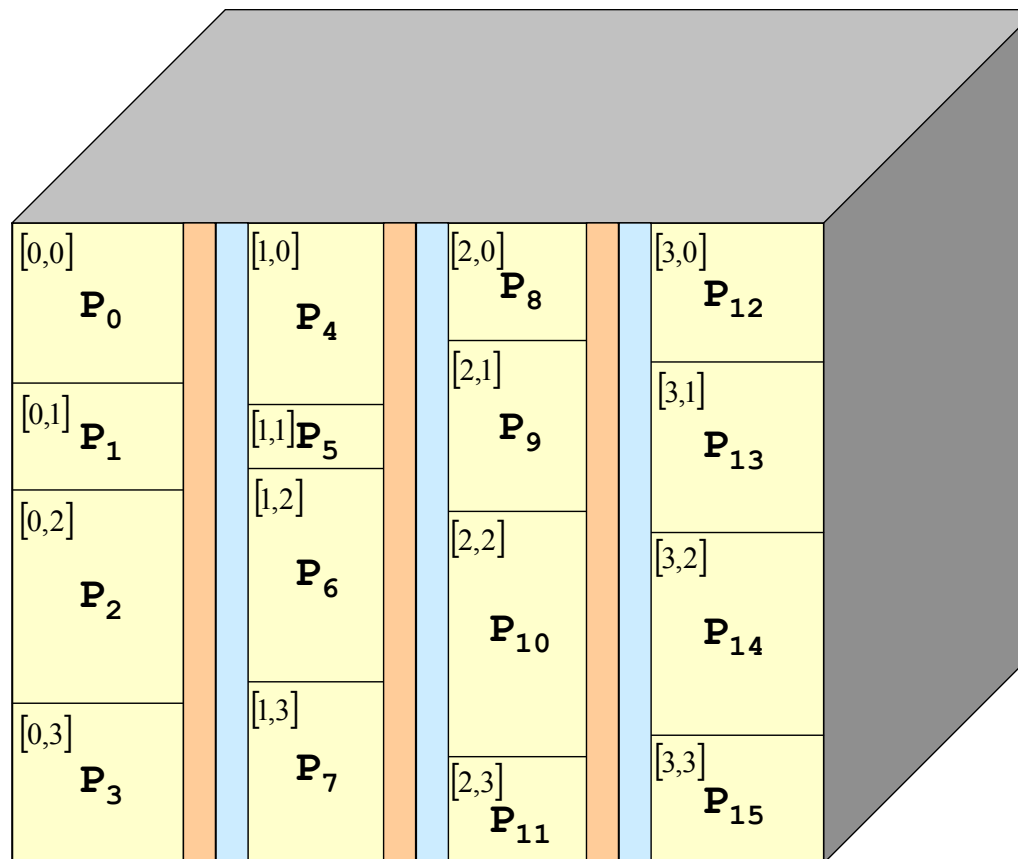
We have adopted as benchmark function the computation of the MEI index for a 3x3 SE:



- 1) The morphological algorithm is based on repeatedly computing this function.
- 2) It prevents the inclusion into the performance model of optimization aspects, such as the possible presence in cache memory of pixels belonging to a certain SE neighborhood.
- 3) We assume for the computation of the benchmark function that the amount of data allocated to a single processor in the cluster is a full AVIRIS hyperspectral cube with 614x512 pixels (we assume an *unfavorable* scenario in which each processor is probably forced to make use of reallocation/paging mechanisms due to cache misses.)

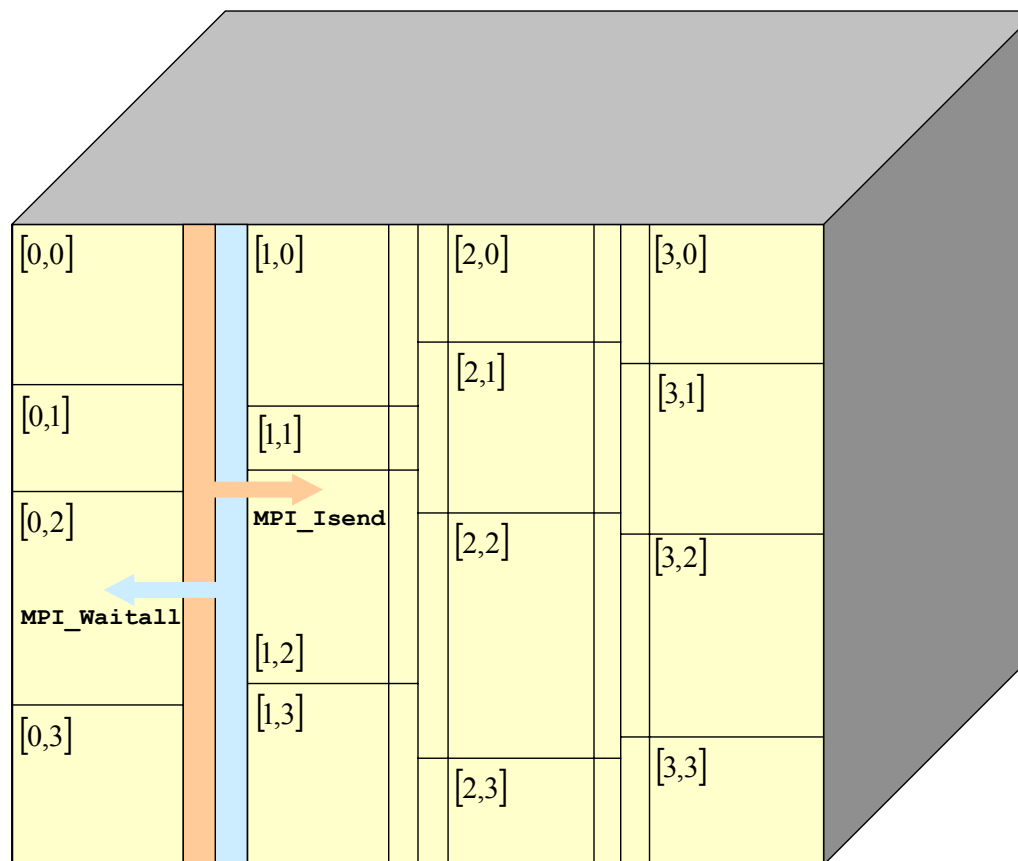
Communication framework (I)

Assignment of data partitions to a set of heterogeneous processors arranged in a 4x4 virtual processor grid:



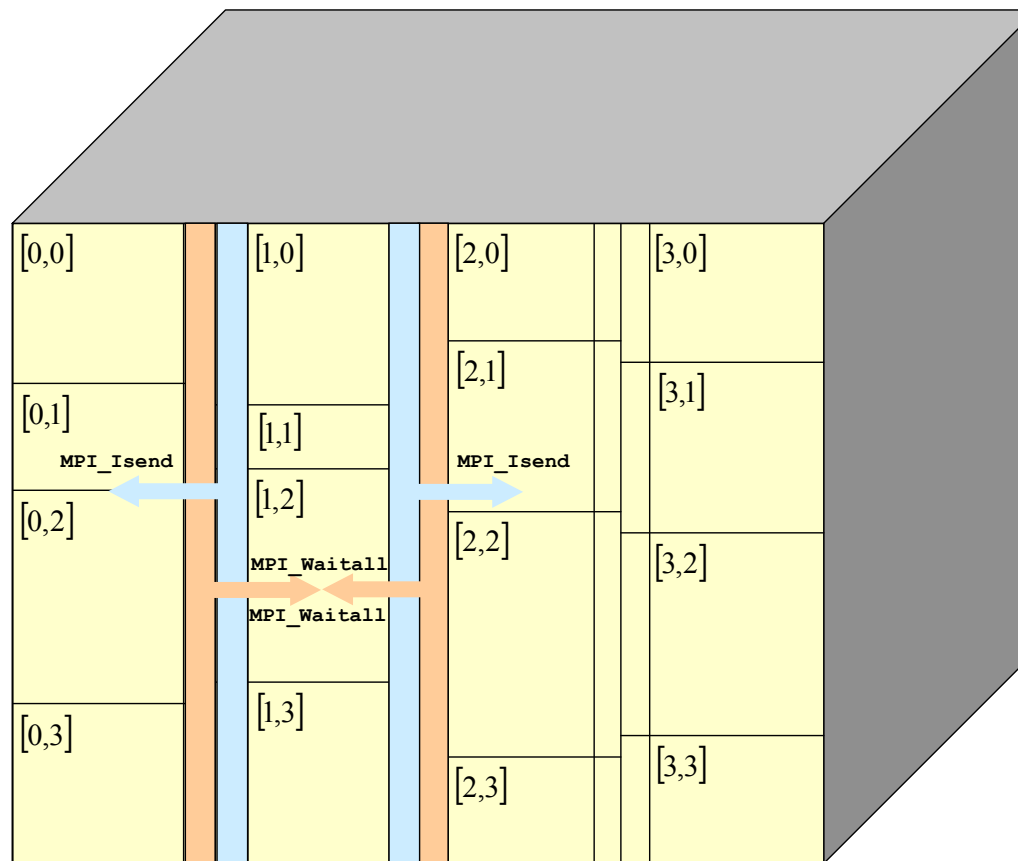
Communication framework (II)

Processors in leftmost column (column 0) first send their overlap borders to processors in column 1 and then wait for the overlap borders of processors in that column:



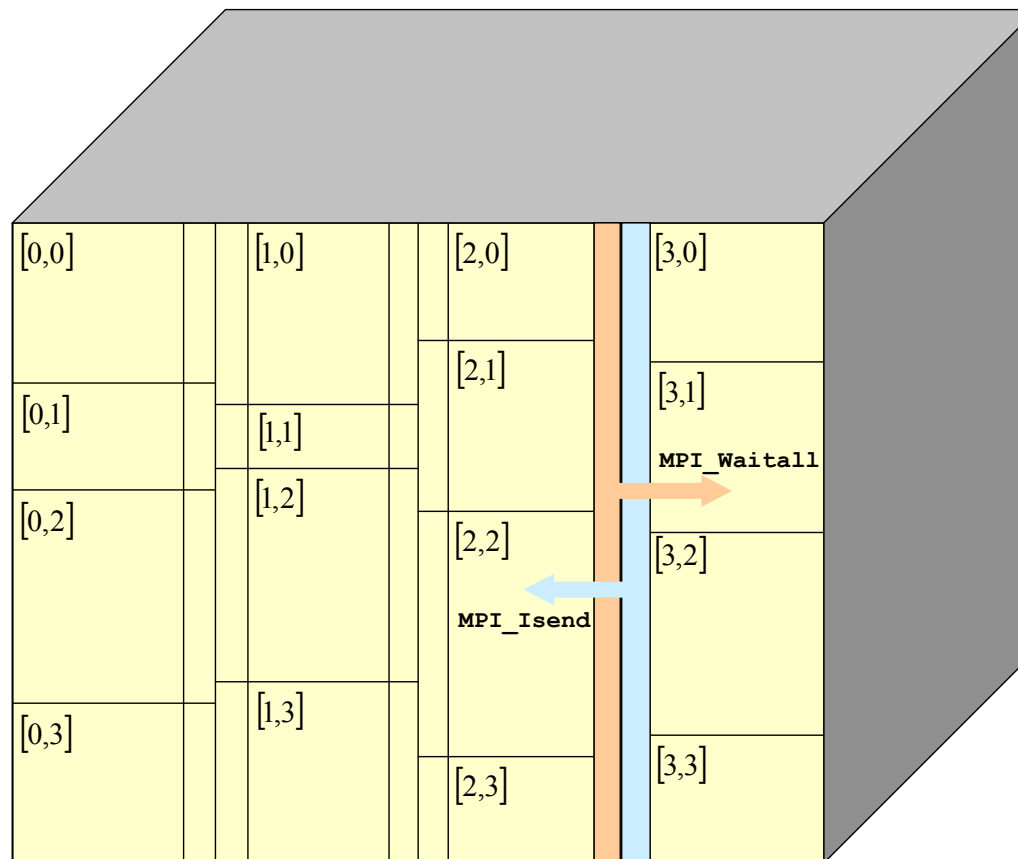
Communication framework (III)

Processors in middle column (column 1) first send their overlap borders to processors in columns 0 and 2 and then wait for the overlap borders of processors in those columns:



Communication framework (IV)

Processors in rightmost column (column 3) first wait for the overlap borders of processors in column 2 and then send their overlap borders to processors in that column:



HeteroMPI implementation (I)

```
main(int argc, char *argv[]){
  HeteroMPI_Init(&argc,&argv);
  if (HeteroMPI_Is_member(HMPI_COMM_WORLD_GROUP)){
    HeteroMPI_Recon(benchmark, dims, 15, &output);
  }
  HeteroMPI_Group_create(&gid, &MPC_NetType_MM_perf, modelp, num_param);
  if (HeteroMPI_Is_free()){
    HeteroMPI_Group_create(&gid, &MPC_NetType_MM_rend, NULL, 0);
  }
  if (HeteroMPI_Is_free()){
    HeteroMPI_Finalize(0);
  }
  //Cont'd in next slide
}
```

- Runtime system initialized using **HeteroMPI_Init**.
- Operation **HeteroMPI_Recon** estimates the performances of processors using benchmark.
- A group of processes is created using **HeteroMPI_Group_create** (the members of the group execute the parallel algorithm).

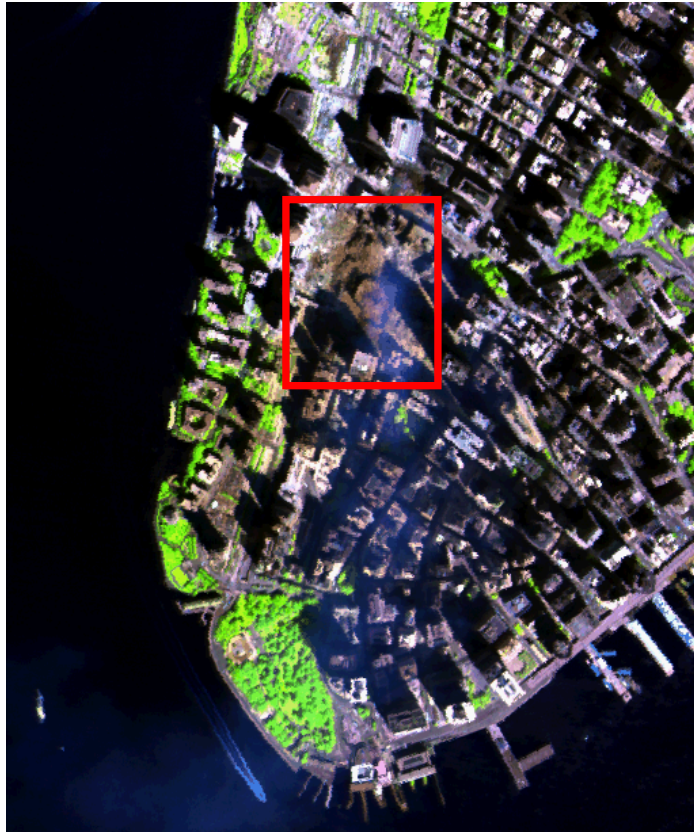
HeteroMPI implementation (II)

```
if (HeteroMPI_Is_member(&gid)){
    Communicator = *(MPI_Comm *)HeteroMPI_Get_comm(&gid);
    if (&Communicator == NULL){
        HeteroMPI_Finalize(0);}
    if (HeteroMPI_Group_coordof(&gid,&dim,&coord) == HMPI_SUCCESS){
        HeteroMPI_Group_performances(&gid, speeds);
        Read_image(name,image,lin,col,bands,data_type,init);
        for (i=imax; i>1; i=i--){
            AMC_algorithm(image,lin,col,bands,sizeofB,res);
            //Communication framework through MPI_Isend and MPI_Waitall
        }
        if (HeteroMPI_Is_member(&gid)){
            free(image);}
        HeteroMPI_Group_free(&gid);
        HeteroMPI_Finalize(0);
    }
```

- HeteroMPI and MPI interconnected by **HeteroMPI_Get_comm**, which returns an MPI communicator with communication group of MPI processes defined by **gid**.
- Communicator used to call standard MPI comm routines as **MPI_Isend** and **MPI_Waitall**.
- Group freed with **HeteroMPI_Group_free**; runtime system finalized with **HeteroMPI_Finalize**.

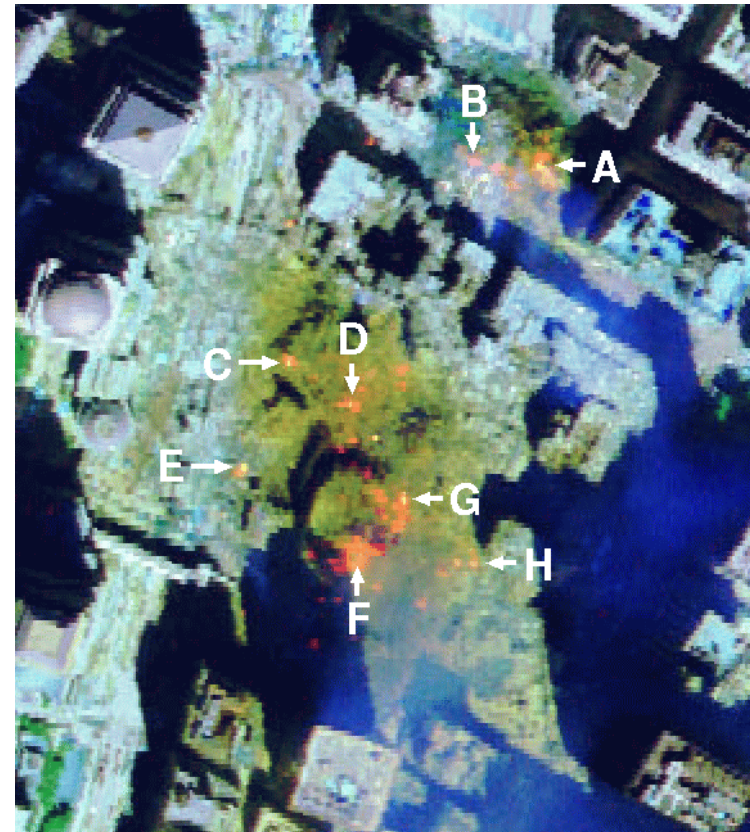
Experimental data (I)

Data set owned by NASA/Jet Propulsion Lab



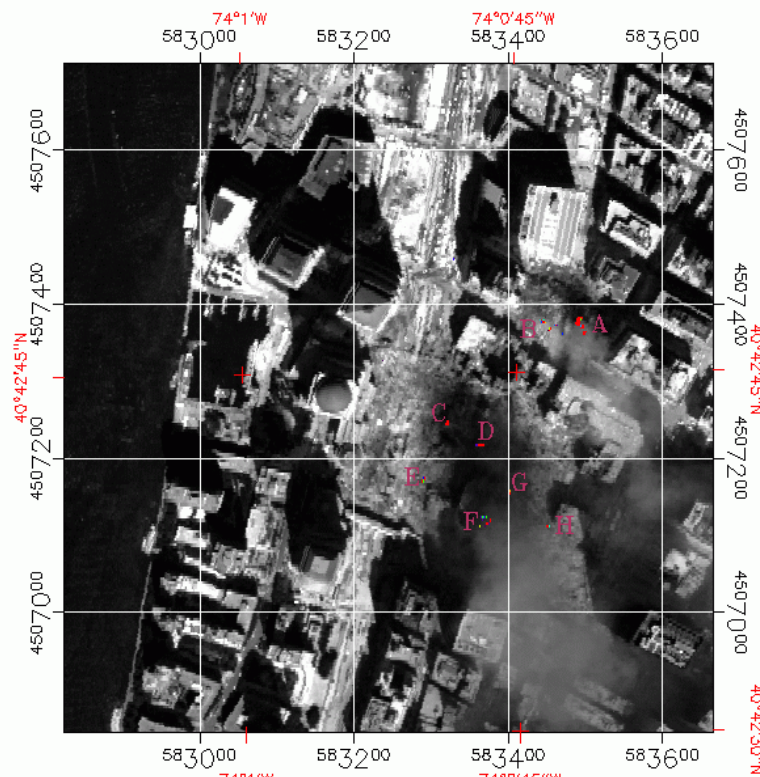
AVIRIS data over lower Manhattan (09/15/01)

Data set owned by U.S. Geological Survey




Spatial location of thermal hot spots in WTC area

Experimental data (II)



AVIRIS Hot Spot Mapping 09/16/2001
 Color Coded Hot Spots on B/W Image Background
 UTM projection, zone 18
 NAD-83/WGS-84 datum, 1.7 meter pixels


 lower temp. scale higher

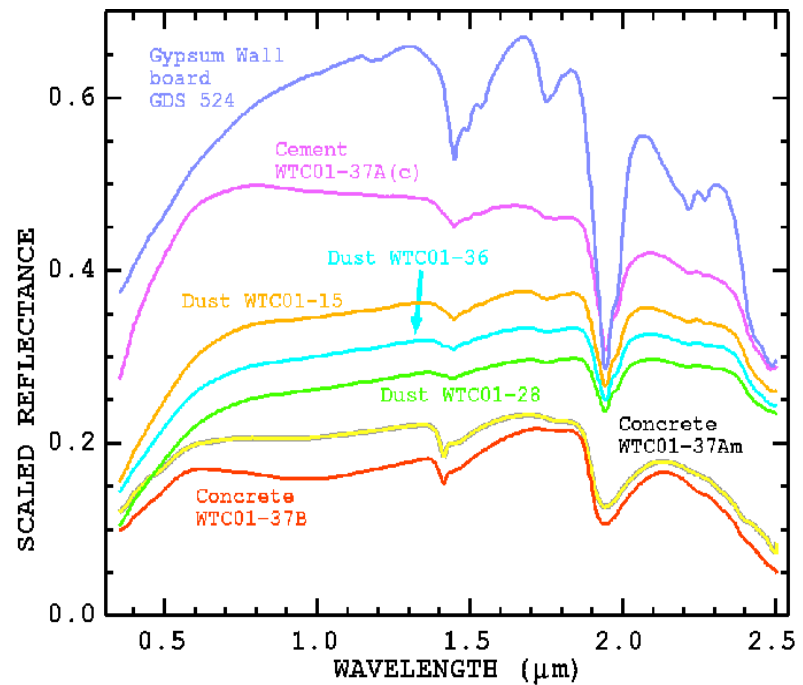
Spot	Kelvin	% FOV	Area (m ²)
A	1000	15	0.56
B	830	2	0.08
C	900	20	0.8
D	790	20	0.8
E	710	10	0.4
F	700	10	0.4
G	1020	1	0.04
H	820	2	0.08

Properties of the eight main thermal hot spots in the WTC area, including temperature, percentage of field of view occupied and approximate size (in square meters)

Experimental data (III)



Dust/debris map generated by USGS



Detection/classification accuracy

Hot Spot	Temp (K)	Size M ²	PPI (4012)	ATGP (1263)	UFCLS (916)
A	1000	0.56	0.001	0.002	0.123
B	830	0.08	0.001	0.001	0.005
C	900	0.8	0.003	0.005	0.012
D	790	0.8	0.002	0.003	0.002
E	710	0.4	0.005	0.008	0.026
F	700	0.4	0.001	0.001	0.169
G	1020	0.04	0.000	0.000	0.000
H	820	0.08	0.000	0.000	0.008

SAD-based spectral similarity scores between target pixels detected by heterogeneous algorithms and the known ground targets. Single-processor times are given in the parentheses.

Dust/debris USGS Ground-truth class	MM (2334)	PCT (1884)
Concrete (37A)	93.56	95.67
Concrete (37B)	90.23	93.28
Cement	81.64	89.43
Dust (15)	79.23	88.65
Dust (28)	76.67	92.05
Dust (36)	85.02	91.23
Gypsum wall board	82.99	96.89
Overall	80.45	93.96

Classification accuracies (percentage) obtained by ANN-based heterogeneous algorithms for the dust/debris ground classes available from USGS. Sequential times are given in the parentheses.

Heterogeneous clusters (I)

Heterogeneous cluster (**HCL-1**) at University College Dublin

Proc. Number	Name (processors)	Architecture description	CPU (MHz)	Memory (MB)	Cache (KB)	Relative speed
0-7	Pg1cluster (2 processors)	Linux 2.4.18-10smp Intel(R) XEON(TM)	1977	1024	512	70
8-14	Csultra (1 processor)	SunOS 5.8 sun4u sparc SUNW, Ultra-5_10	440	512	2048	30

Heterogeneous clusters (II)

Heterogeneous cluster (**HCL-2**) at University College Dublin

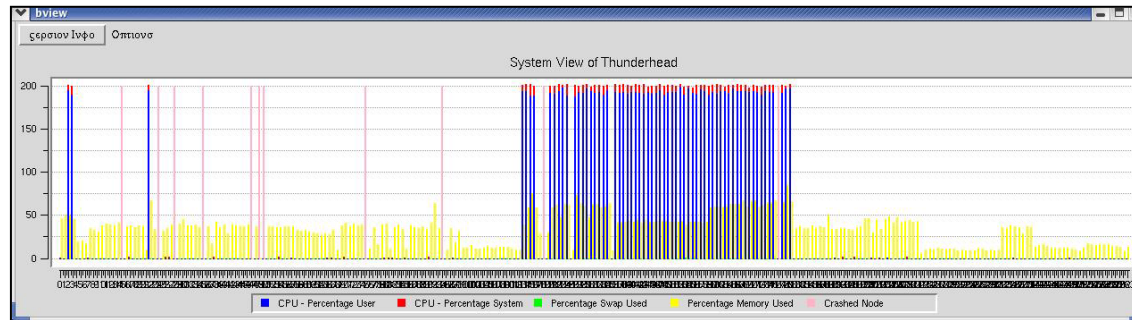
Proc #	Model and description	Processor type	Operating system	CPU (Ghz)	Mem. (MB)	Cache (KB)	HDD 1	HDD 2	Rel. speed
0,1	Dell Poweredge SC1425	Intel Xeon	Fedora Core 4	3.6	256	2048	240 GB SCSI	80 GB SCSI	7.93
2-7	Dell Poweredge 750	Intel Xeon	Fedora Core 4	3.4	1024	1024	80 GB SATA	N/A	7.20
8	IBM E-server 326	Opteron	Debian	1.8	1024	1024	80 GB SATA	N/A	2.75
9	IBM E-server 326	Opteron	Fedora Core 4	1.8	1024	1024	80 GB SATA	N/A	2.75
10	IBM X-Series 306	Pentium 4	Debian	3.2	512	1024	80 GB SATA	N/A	6.13
11	HP Proliant DL 320 G3	Pentium 4	Fedora Core 4	3.4	512	1024	80 GB SATA	N/A	6.93
12	HP Proliant DL 320 G3	Celeron	Fedora Core 4	2.9	1024	256	80 GB SATA	N/A	3.40
13	HP Proliant DL 140 G2	Intel Xeon	Debian	3.4	1024	1024	80 GB SATA	N/A	7.73
14	HP Proliant DL 140 G2	Intel Xeon	Debian	2.8	1024	1024	80 GB SATA	N/A	3.26
15	HP Proliant DL 140 G2	Intel Xeon	Debian	3.6	1024	2048	80 GB SATA	N/A	8.60

Beowulf commodity cluster

Thunderhead (NASA/GSFC)

<http://thunderhead.gsfc.nasa.gov>

Aggregate Specification	
Number of nodes	256
Total processors	512
Total memory (Gb)	256
Total disk (GB)	20480
Interconnect 1	Myrinet 2000
Interconnect 2	Gigabit Ethernet
Total peak performance (Gflops)	2457.6
Node Specification	
Motherboard	Tyan Thunder 2720
Number of processors	Dual Intel 4 Xeon 2.4Ghz
Memory (Gb)	1
Local disk (Gb)	80
Interconnect 1	Myrinet 2000
Interconnect 2	Gigabit 10/100/1000
Peak performance (Gflops)	9.6



Experimental results on HCL-1 (I)

Execution times of the HeteroMPI based algorithm on HCL-1 (different numbers of iterations):

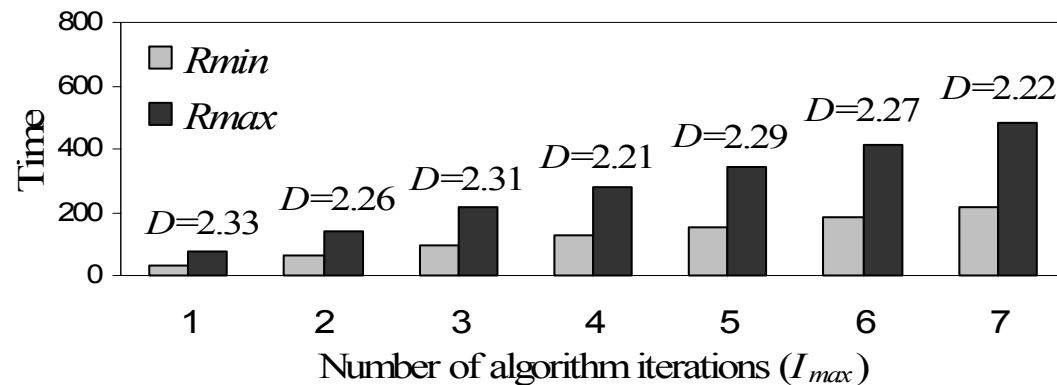
Processor:	1	2	3	4	5	6	7
0	46.86	91.25	140.69	186.46	226.06	285.51	337.49
1	47.05	90.74	141.49	183.66	228.06	288.77	328.88
2	47.32	92.15	138.23	187.38	227.75	287.96	325.31
3	47.09	92.96	134.46	180.55	226.68	274.10	317.73
4	50.01	95.57	149.55	199.20	237.06	300.94	340.53
5	50.59	94.95	148.70	197.76	235.17	309.22	345.14
6	48.32	99.48	139.15	188.48	246.55	291.75	329.67
7	48.26	91.82	143.86	191.09	246.61	294.96	333.94
8	48.90	101.28	141.44	188.25	250.61	290.83	322.06
9	50.48	98.63	152.04	200.33	238.35	304.19	358.36
10	51.07	98.48	154.39	197.50	238.12	308.83	358.06
11	46.43	92.69	139.80	180.44	227.03	274.77	321.50
12	47.12	93.24	141.40	183.85	229.87	282.43	328.16
13	46.54	92.35	137.60	184.44	231.65	288.52	315.20
14	46.85	94.47	137.70	186.32	235.26	288.67	326.25

Experimental results on HCL-1 (II)

Load-balancing rates using the proposed benchmark function:

Iterations:	1	2	3	4	5	6	7
<i>Rmin</i>	46.43	90.74	134.46	180.44	226.06	309.22	358.36
<i>Rmax</i>	51.07	101.28	154.39	200.33	250.61	274.10	315.20
<i>D</i> (imbalance)	1.09	1.11	1.14	1.11	1.10	1.12	1.13

Load-balancing rates using a benchmark function without memory considerations:



Experimental results on HCL-2 (I)

Execution times of the HeteroMPI based algorithm on HCL-2 (different numbers of bands):

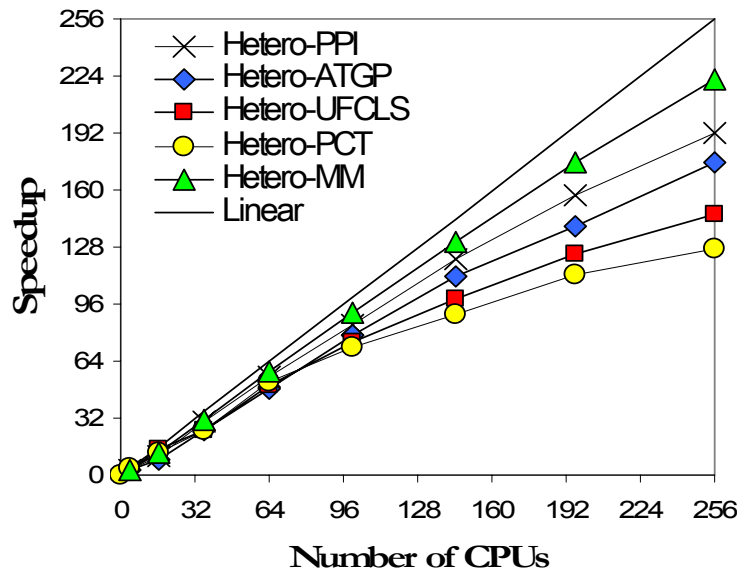
Processor Number:	Number of spectral bands in the Indian Pines AVIRIS scene:								
	20	40	60	80	100	120	140	160	180
0	3.41	5.03	6.93	9.69	11.49	14.02	16.73	19.09	21.45
1	3.46	5.32	7.05	9.16	13.33	14.02	16.74	19.09	21.46
2	3.25	5.03	6.92	9.12	13.30	13.98	16.67	19.03	21.39
3	3.27	5.10	7.17	9.34	13.34	14.03	16.74	19.11	21.47
4	3.45	5.27	7.14	9.67	13.29	13.98	16.69	19.04	21.41
5	3.45	5.31	7.16	9.69	13.32	14.01	16.72	19.10	21.46
6	3.44	5.28	7.15	9.67	13.31	13.99	16.70	19.05	21.41
7	3.46	5.32	7.17	9.70	13.34	14.03	16.74	19.11	21.47
8	3.26	5.02	6.91	9.14	13.32	13.99	16.72	19.08	21.42
9	3.24	5.01	6.91	9.12	13.29	13.97	16.67	19.02	21.39
10	3.26	5.04	6.93	9.13	13.31	14.00	16.70	19.07	21.44
11	3.24	4.98	6.90	9.10	13.28	13.95	16.65	19.00	21.37
12	2.08	2.48	2.81	3.24	3.67	12.72	16.71	19.06	21.40
13	2.09	2.64	2.99	3.35	3.67	12.71	16.68	19.04	21.42
14	2.10	2.47	2.82	3.26	3.68	12.72	16.70	19.07	21.44
15	2.09	2.47	2.81	3.24	3.66	12.71	16.68	19.05	21.40

Experimental results on HCL-2 (II)

Execution times of the HeteroMPI based algorithm on HCL-2 (different numbers of iterations):

Processor:	1	2	3	4	5	6	7
0	21.45	42.53	63.98	84.84	107.74	130.67	145.63
1	21.46	42.59	63.88	84.80	107.68	130.71	145.64
2	21.39	42.58	63.83	84.99	107.67	130.65	145.65
3	21.47	42.56	63.77	84.74	106.42	130.72	145.64
4	21.41	42.56	62.86	84.80	107.68	130.72	145.56
5	21.46	42.49	63.84	84.85	107.74	130.59	145.58
6	21.41	42.61	63.81	84.77	107.73	130.66	144.39
7	21.47	42.60	63.97	84.95	107.74	130.67	145.56
8	21.42	42.54	63.81	83.88	107.67	130.65	145.60
9	21.39	42.52	63.82	84.79	107.70	128.88	145.52
10	21.44	42.60	63.80	84.78	107.69	130.71	145.63
11	21.37	42.53	63.84	84.84	107.71	130.64	145.61
12	21.40	42.61	63.80	84.77	107.66	130.64	145.59
13	21.42	42.52	63.88	84.77	107.69	130.63	145.59
14	21.44	42.59	63.83	84.78	107.63	130.66	145.58
15	21.40	42.59	63.88	84.95	107.73	130.70	145.58

Results on NASA's Thunderhead



- A performance drop is observed for all algorithms when the number of processors is very large.
- This is because the partition sizes decrease significantly, which result in a significant increase in the execution times of the single **MPI_Gather** operation used to develop our MPI-based parallel code for the homogeneous platform.
- A possible solution is to replace the single **MPI_Gather** gathering medium-sized messages by an equivalent sequence of MPI_Gather operations, each gathering messages with a size that fits a range of smaller messages.

	1	4	16	36	64	100	144	196	256
Hetero-PPI	4012	1638	388	135	72	48	33	26	21
Hetero-ATGP	1263	493	141	49	26	16	11	9	7
Hetero-UFCLS	916	286	63	36	18	12	9	7	6
Hetero-PCT	1884	460	154	73	36	26	21	17	15
Hetero-MM	2334	741	191	74	40	26	18	13	11

Processing times (in seconds) of heterogeneous algorithms on NASA's Thunderhead system

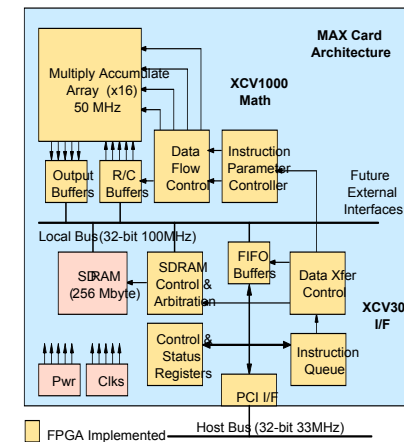
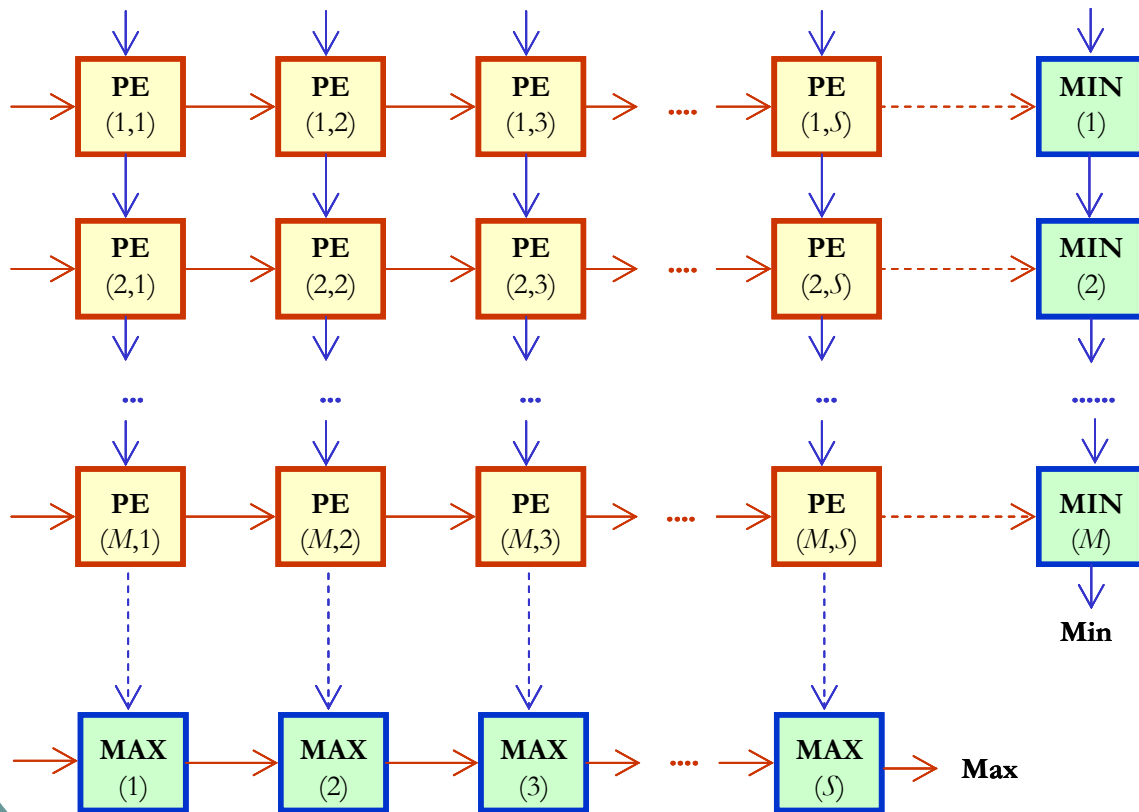
Summary and observations

- Despite the enormous computational demands and potential societal impact, the remote sensing community has not yet developed standardized parallel algorithms for low-cost computing architectures.
- Heterogeneous computing offers an excellent alternative to expensive dedicated computers in data mining and information extraction apps.
- Distributed nature of such networks fits the properties of remote sensing processing environments, with many (different but related) institutions collecting high-dimensional data (Grid computing).
- Evaluation strategy based on comparing efficiency of heterogeneous algorithms on heterogeneous NOWs with the efficiency achieved by homogeneous versions on equally powerful homogeneous NOWs.
- Experimental results reveal that heterogeneous computing may introduce relevant changes in current parallel remote sensing systems.
- Further research is required to incorporate a model of collective communications into our considered application.



Future research lines

- Real-time onboard processing using low-weight hardware components such as FPGAs, GPUs and heterogeneous networks of such devices.



References and support

European support:

Hyperspectral Imaging Network (HYPER-I-NET), *Marie Curie Research Training Network* (2007-2011). Budget: 2.8 MEuro (15 European partners, coordinated by A. Plaza). <http://www.hyperinet.eu>.

Hyperspectral processing in commodity clusters:

A. Plaza, D. Valencia, J. Plaza and P. Martinez, "Commodity cluster-based parallel processing of hyperspectral imagery," *Journal of Parallel and Distributed Computing*, vol. 66, no. 3, pp. 345-358, 2006.

Hyperspectral processing in heterogeneous networks:

A. Plaza, J. Plaza and D. Valencia, "Impact of platform heterogeneity on the design of parallel algorithms for morphological processing of high-dimensional image data," *Journal of Supercomputing*, vol. 40, no. 1, pp. 81-107, 2007.

Hyperspectral processing in FPGAs:

A. Plaza and C.-I Chang, "Clusters versus FPGAs for parallel processing of hyperspectral imagery," *International Journal of High Performance Computing Applications*, accepted for publication.

Hyperspectral processing in GPUs:

J. Setoain, M. Prieto, C. Tenllado, A. Plaza and F. Tirado, "Parallel morphological endmember extraction using commodity graphics hardware," *IEEE Geoscience and Remote Sensing Letters*, vol. 4, no. 3, 2007.

Upcoming publications:

A. Plaza and C.-I Chang, Eds., *High-Performance Computing in Remote Sensing*, CRC Press, 2007.

A. Plaza and C.-I Chang, Eds., Special issue on "High-Performance Computing for Hyperspectral Imaging," *International Journal of High Performance Computing Applications*, to appear in 2008.