Efficient Clustering of Geo-Spatial Timeseries using Parallel Dataflow Programs
GIS Day 2012 – Geoforschungszentrum Potsdam

Matthias J. Sax
DBIS-Group, HU Berlin
mjsax@informatik.hu-berlin.de

Mike Sips
GFZ Potsdam
sips@gfz-potsdam.de

November 14th 2012
Agenda

Introduction

Approach: Pyramid Matching Kernel

Parallel Execution

Conclusion
Problem Statement

- given is a data set with about 90,000 spatial patterns (more than 600GB)
- **goal**: finding prominent spatial patterns
Problem Statement

- given is a data set wit about 90,000 spatial patterns (more than 600GB)
- **goal**: finding prominent spatial patterns
  - cluster the input
  - **SSE-distance no applicable**
Analysis of Flood Simulation Data

Timeseries:
- each value is a grid with about $1000 \times 800$ points
- 2-dimensional matrix which models a dedicated area
- each grid-point data value represents water level

![Graph showing flood simulation data](image-url)
Analysis of Flood Simulation Data

Timeseries:

- each value is a grid with about $1000 \times 800$ points
  - 2-dimensional matrix which models a dedicated area
  - each grid-point data value represents water level

- using SSE-Algorithm for clustering takes too long
  - $800,000$ “comparisons” per grid-pair
- use different distance measure to reduce complexity
Approach: Pyramid Matching Kernel

Idea

- transform grids into compact representation
- define distance measure using this representation
Approach: Pyramid Matching Kernel

Idea

- transform grids into compact representation
- define distance measure using this representation

Approach:

Compact Representation:

- feature extraction: divide value range in $\#f$ partition of equal size
- usage of spatial pyramid (count number of features)

Distance measure:

- compare number of features for each grid-pair
Approach: Pyramid Matching Kernel (cont.)

Compact Representation:
- feature extraction: divide value range in $\#f$ partition of equal size
- usage of spatial pyramid (count number of features)

Distance measure:
- compare number of features for each grid-pair
Approach: Pyramid Matching Kernel (cont.)

Compact Representation:
- feature extraction: divide value range in \( f \) partition of equal size
- usage of spatial pyramid (count number of features)

Distance measure:
- compare number of features for each grid-pair

Level 0

\[
\begin{array}{c}
\bullet \\
\bullet \\
\bullet \\
\bullet \\
\end{array}
\]

0: 4
Approach: Pyramid Matching Kernel (cont.)

Approach:

Compact Representation:
- feature extraction: divide value range in $\#f$ partition of equal size
- usage of spatial pyramid (count number of features)

Distance measure:
- compare number of features for each grid-pair
Approach: Pyramid Matching Kernel (cont.)

Approach:

Compact Representation:
- feature extraction: divide value range in $\#f$ partition of equal size
- usage of spatial pyramid (count number of features)

Distance measure:
- compare number of features for each grid-pair

Level 2

0: 4
1: 1,1,0,2
2: 1,0,0,0
    0,0,1,0
    0,0,1,0
    0,0,0,1
Approach: Pyramid Matching Kernel (cont.)

Approach:

Compact Representation:
- feature extraction: divide value range in \( \# f \) partition of equal size
- usage of spatial pyramid (count number of features)

Distance measure:
- compare number of features for each grid-pair
Clustering with Stratosphere

- joined research project at TU Berlin, HU Berlin, and HPI Potsdam
- parallel data processing system for clusters
- executes a dataflow program
  - five different types of nodes in the program
  - user-defined functions
  - imperative Java code
- automatic parallelization of dataflow program
Parallel Computation of Pyramid Matching Kernel

<table>
<thead>
<tr>
<th></th>
<th>1.2</th>
<th>8.9</th>
<th>9.1</th>
<th>4.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.9</td>
<td>7.3</td>
<td>1.9</td>
<td>2.8</td>
<td></td>
</tr>
<tr>
<td>7.1</td>
<td>5.2</td>
<td>4.0</td>
<td>0.3</td>
<td></td>
</tr>
<tr>
<td>3.9</td>
<td>8.1</td>
<td>6.2</td>
<td>2.4</td>
<td></td>
</tr>
</tbody>
</table>
### Parallel Computation of Pyramid Matching Kernel

#### 3 features:

- **0.0 to 3.9**
- **4.0 to 6.9**
- **7.0 to 9.9**

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.2</td>
<td>8.9</td>
<td>9.1</td>
<td>4.0</td>
</tr>
<tr>
<td>6.9</td>
<td>7.3</td>
<td>1.9</td>
<td>2.8</td>
</tr>
<tr>
<td>7.1</td>
<td>5.2</td>
<td>4.0</td>
<td>0.3</td>
</tr>
<tr>
<td>3.9</td>
<td>8.1</td>
<td>6.2</td>
<td>2.4</td>
</tr>
</tbody>
</table>
Parallel Computation of Pyramid Matching Kernel

<table>
<thead>
<tr>
<th>1.2</th>
<th>8.9</th>
<th>9.1</th>
<th>4.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.9</td>
<td>7.3</td>
<td>1.9</td>
<td>2.8</td>
</tr>
<tr>
<td>7.1</td>
<td>5.2</td>
<td>4.0</td>
<td>0.3</td>
</tr>
<tr>
<td>3.9</td>
<td>8.1</td>
<td>6.2</td>
<td>2.4</td>
</tr>
</tbody>
</table>

3 features:

- Blue: 0.0 to 3.9
- Green: 4.0 to 6.9
- Red: 7.0 to 9.9

Features: 1.2, 8.9, 9.1, 4.0, 6.9, 7.3, 1.9, 2.8, 7.1, 5.2, 0.3, 3.9, 8.1, 6.2, 2.4
Parallel Computation of Pyramid Matching Kernel

3 features:
- Blue square: 0.0 to 3.9
- Green circle: 4.0 to 6.9
- Red triangle: 7.0 to 9.9

Table:

<table>
<thead>
<tr>
<th>1.2</th>
<th>8.9</th>
<th>9.1</th>
<th>4.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.9</td>
<td>7.3</td>
<td>1.9</td>
<td>2.8</td>
</tr>
<tr>
<td>7.1</td>
<td>5.2</td>
<td>4.0</td>
<td>0.3</td>
</tr>
<tr>
<td>3.9</td>
<td>8.1</td>
<td>6.2</td>
<td>2.4</td>
</tr>
</tbody>
</table>

Diagram:
- Arrows pointing down indicating the features:
  - 1.2, 8.9, 9.1, 4.0
  - 6.9, 7.3, 1.9, 2.8
  - 7.1, 5.2, 4.0, 0.3
  - 3.9, 8.1, 6.2, 2.4
Parallel Computation of Pyramid Matching Kernel II

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(0/0)</td>
<td>(1/0)</td>
<td>(2/0)</td>
<td>(3/0)</td>
</tr>
<tr>
<td>(0/1)</td>
<td>(1/1)</td>
<td>(2/1)</td>
<td>(3/1)</td>
</tr>
<tr>
<td>(0/2)</td>
<td>(1/2)</td>
<td>(2/2)</td>
<td>(3/2)</td>
</tr>
<tr>
<td>(0/3)</td>
<td>(1/3)</td>
<td>(2/3)</td>
<td>(3/3)</td>
</tr>
</tbody>
</table>

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(0/0)</td>
<td>(0/1)</td>
</tr>
<tr>
<td>(0/1)</td>
<td>(1/1)</td>
</tr>
</tbody>
</table>

<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(0/0)</td>
</tr>
</tbody>
</table>
Parallel Computation of Pyramid Matching Kernel II

<table>
<thead>
<tr>
<th></th>
<th>(0/0)</th>
<th>(0/1)</th>
<th>(0/2)</th>
<th>(0/3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0/0)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(0/1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(0/2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(0/3)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>(0/0)</th>
<th>(0/1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0/0)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(0/1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(1/1)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```
(0/0)             (0/1)
(0/1)             (1/1)
(0/0)             (0/1) 
(0/1)             (1/1)
```

(0/0)
Parallel Computation of Pyramid Matching Kernel II

<table>
<thead>
<tr>
<th>(0/0)</th>
<th>(0/1)</th>
<th>(0/2)</th>
<th>(0/3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1/0)</td>
<td>(1/1)</td>
<td>(1/2)</td>
<td>(1/3)</td>
</tr>
<tr>
<td>(2/0)</td>
<td>(2/1)</td>
<td>(2/2)</td>
<td>(2/3)</td>
</tr>
<tr>
<td>(3/0)</td>
<td>(3/1)</td>
<td>(3/2)</td>
<td>(3/3)</td>
</tr>
</tbody>
</table>

Diagram:

- (0/0) connected to (0/1) and (1/1)
- (0/1) connected to (1/1)
### Parallel Computation of Pyramid Matching Kernel II

#### Table 1: Pyramid Levels

<table>
<thead>
<tr>
<th>Level</th>
<th>Elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>0/0</td>
<td>(0/0)</td>
</tr>
<tr>
<td>1/0</td>
<td>(0/1)</td>
</tr>
<tr>
<td>2/0</td>
<td>(0/2)</td>
</tr>
<tr>
<td>3/0</td>
<td>(0/3)</td>
</tr>
<tr>
<td>0/1</td>
<td>(1/0)</td>
</tr>
<tr>
<td>1/1</td>
<td>(1/1)</td>
</tr>
<tr>
<td>2/1</td>
<td>(1/2)</td>
</tr>
<tr>
<td>3/1</td>
<td>(1/3)</td>
</tr>
<tr>
<td>0/2</td>
<td>(2/0)</td>
</tr>
<tr>
<td>1/2</td>
<td>(2/1)</td>
</tr>
<tr>
<td>2/2</td>
<td>(2/2)</td>
</tr>
<tr>
<td>3/2</td>
<td>(2/3)</td>
</tr>
<tr>
<td>0/3</td>
<td>(3/0)</td>
</tr>
<tr>
<td>1/3</td>
<td>(3/1)</td>
</tr>
<tr>
<td>2/3</td>
<td>(3/2)</td>
</tr>
<tr>
<td>3/3</td>
<td>(3/3)</td>
</tr>
</tbody>
</table>
Parallel Computation of Pyramid Matching Kernel II

<table>
<thead>
<tr>
<th></th>
<th>(0/0)</th>
<th>(0/1)</th>
<th>(0/2)</th>
<th>(0/3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0/0)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(1/0)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(2/0)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(3/0)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>(0/0)</th>
<th>(0/1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0/0)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(0/1)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The table and diagram illustrate the parallel computation of the Pyramid Matching Kernel II, showing how different levels of the pyramid are computed in parallel, with arrows indicating the progression from lower to higher levels.
Parallel Computation of Pyramid Matching Kernel II

<table>
<thead>
<tr>
<th></th>
<th>(0/0)</th>
<th>(0/1)</th>
<th>(0/2)</th>
<th>(0/3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1/0)</td>
<td>(1/1)</td>
<td>(1/2)</td>
<td>(1/3)</td>
<td></td>
</tr>
<tr>
<td>(2/0)</td>
<td>(2/1)</td>
<td>(2/2)</td>
<td>(2/3)</td>
<td></td>
</tr>
<tr>
<td>(3/0)</td>
<td>(3/1)</td>
<td>(3/2)</td>
<td>(3/3)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>(0/0)</th>
<th>(0/1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0/1)</td>
<td>(1/1)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>(0/0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0/0)</td>
<td></td>
</tr>
</tbody>
</table>

1 2

1 2

6
### Parallel Computation of Pyramid Matching Kernel II

<table>
<thead>
<tr>
<th></th>
<th>(0/0)</th>
<th>(0/1)</th>
<th>(0/2)</th>
<th>(0/3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0/0)</td>
<td>(0/0)</td>
<td>(0/1)</td>
<td>(0/2)</td>
<td>(0/3)</td>
</tr>
<tr>
<td>(1/0)</td>
<td>(1/0)</td>
<td>(1/1)</td>
<td>(1/2)</td>
<td>(1/3)</td>
</tr>
<tr>
<td>(2/0)</td>
<td>(2/0)</td>
<td>(2/1)</td>
<td>(2/2)</td>
<td>(2/3)</td>
</tr>
<tr>
<td>(3/0)</td>
<td>(3/0)</td>
<td>(3/1)</td>
<td>(3/2)</td>
<td>(3/3)</td>
</tr>
</tbody>
</table>

### Diagram Explanation:
- The diagram illustrates the parallel computation of the pyramid matching kernel.
- Each row represents a level in the pyramid, with the rows indicating the matching levels.
- The arrows between the levels represent the matching operations.
- The numbers along the arrows indicate the matching values.

### Example:
- For level (0/0), the matching values are 1, 0, 0, 0.
- For level (1/0), the matching values are 0, 0, 1, 1.
- For level (2/0), the matching values are 0, 0, 0, 1.
- For level (3/0), the matching values are 1, 0, 0, 1.
- The final matching value for the entire computation is 6.
Parallel Computation of Pyramid Matching Kernel III
Parallel Computation of Pyramid Matching Kernel III

6,1,2,1,2,1,0,0,0,0,1,1,0,0,0,1,1,0,0,1
5,1,1,1,2,0,0,0,1,1,0,0,0,0,1,1,0,0,0,1,0
5,2,1,2,0,0,1,1,0,0,1,0,0,1,0,0,0,0,1,0,0
Parallel Computation of Pyramid Matching Kernel III

G1, G2, G3, G4, ...

G1, G1, G1

6,1,2,1,2,1,0,0,0,0,1,1,0,0,1,1,0,0,1,0,0,1
5,1,1,1,2,0,0,0,1,1,0,0,0,0,1,1,0,0,0,1,0
5,2,1,2,0,0,1,1,0,0,1,0,0,0,1,0,0,0,0,1,0,0
Parallel Computation of Pyramid Matching Kernel III

G2, G3, G4, ...

G1

G1

G1

6,1,2,1,2,1,0,0,0,0,1,1,0,0,1,1,0,0,1,1,0,0,1

5,1,1,1,2,0,0,0,1,1,0,0,0,0,1,1,0,0,0,1,0

5,2,1,2,0,0,1,1,0,0,1,0,0,1,0,0,0,0,1,0,0,1,0,0
Parallel Computation of Pyramid Matching Kernel III

G1

6,1,2,1,2,1,0,0,0,0,0,1,1,0,0,1,1,0,0,1

5,1,1,1,2,0,0,0,1,1,0,0,0,0,1,1,0,0,0,1,0

5,2,1,2,0,0,1,1,0,0,1,0,0,1,0,0,0,0,1,0,0

G2, G3, G4, ...

G1

G2

G1

G2
Parallel Computation of Pyramid Matching Kernel III

G1, G2, G3, G4, ...

6, 1, 2, 1, 2, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1
5, 1, 1, 1, 2, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0
5, 2, 1, 2, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0
Parallel Computation of Pyramid Matching Kernel III

G1

G2, G3, G4, ...

G1

G1

6,1,2,1,2,1,0,0,0,0,0,1,1,0,0,0,1,1,0,0,1

5,1,1,2,0,0,0,1,1,0,0,0,1,1,0,0,0,1,0,0

5,2,1,2,0,0,1,1,0,0,1,0,0,0,0,1,0,0,0,0

G1

G1

G2

G2

= 0.3
Parallel Computation of Pyramid Matching Kernel III

6,1,2,1,2,1,0,0,0,0,0,1,1,0,0,0,1,1,0,0,1
5,1,1,1,2,0,0,0,1,1,0,0,0,0,1,1,0,0,0,1,0
5,2,1,2,0,0,1,1,0,0,1,0,0,0,0,1,0,0,0,1,0,0
Evaluation (32 Machines)

Input of 876 grids with 192x94 cells.

Baseline:
- SSE-Algorithm
- shown as dotted line
Conclusion

- Clustering with Pyramid Matching Kernel yield prominent spatial patterns.
- Pyramid Matching Kernel can reduce amount of computation reasonably.
- Pyramid Matching Kernel is parallelize-able.
- Future work: evaluation with 90,000 grids.
Conclusion

- Clustering with Pyramid Matching Kernel yield prominent spatial patterns.
- Pyramid Matching Kernel can reduce amount of computation reasonably.
- Pyramid Matching Kernel is parallelize-able.
- Future work: evaluation with 90,000 grids.

Questions?

Thanks!
Parallel Dataflow Program

READ
Reading lines of files

MAP
Split the records
Determine the feature
Calculate pyramid vectors

REDUCE
Count the features per pyramid vector

REDUCE
Create the signatures

MATCH
Compare featurewise

REDUCE
Summing up feature per compare relation

MAP
Normalize

WRITE
Writing the result to files
PACT Operators: Map and Reduce

- Map(f, { } )
  { f( ), f( ), f( ), f( ), f( ), f( ), f( ), f( ), f( ), f( ) }

- Reduce(g, { } )
  { g( ), g( ), g( ) }

<table>
<thead>
<tr>
<th>record consisting of key and remaining attributes (same color, indicates equal keys)</th>
</tr>
</thead>
<tbody>
<tr>
<td>f</td>
</tr>
<tr>
<td>g</td>
</tr>
<tr>
<td>Map/Reduce</td>
</tr>
</tbody>
</table>
PACT Operators: Match

- **record consisting of key and remaining attributes** (same color, indicates equal keys)
- **f** user defined function taking two single records as input
- **Match** system provided second order function

\[
\text{Match}(f, \{ \text{record consisting of key and remaining attributes, same color, indicates equal keys} \}, \{ \text{system provided second order function} \})
\]
PACT Operators: Cross

- record consisting of key and remaining attributes (same color, indicates equal keys)
- \( f \) user defined function taking two single records as input
- Cross system provided second order function

\[
\text{Cross}(f, \{ \text{red, blue, green} \}, \{ \text{red, green, orange, red} \})
\]
PACT Operators: CoGroup

- record consisting of key and remaining attributes (same color, indicates equal keys)
- $f$: user defined function taking two sets of records as input
- CoGroup: system provided second order function

CoGroup($f$, {red, blue, green, red}, {red, blue, green, red, orange, red})