WEKAREL
A Parallel WEKA Implementation
Scicomp Team

- Myrzabek Murataliev
- Serdar Dalgıç
- Haşim Timurtaş
- Barış Yüksel

Assistant: Çelebi Kocair

Sponsor: METU Computer Eng. Department
OUTLINE

• Problem Definition
• Project Aim
• Architecture
• Tools
• What We Have Done So Far
OUTLINE (cont..)

• Main Problems Encountered
• Project Status
• Test Results
• What's Next – Conclusion
Problem Definition

- **What is Data Mining?**

  Huge amount of data
  
  Mining
  
  Valuable Information and Knowledge

Data mining—searching for knowledge (interesting patterns) in your data.
Problem Definition

• **What are we doing?**
  – Combining Data Mining and Parallelization Concepts
    • WEKA runs on single processors.
Problem Definition

• What are we doing?
  – **WEKAREL**: Parallel processing implementation with *PJ (Parallel Java Library)*
Project Aim

• To build up a general parallelization solution to WEKA tool with notable improvements on running time

• This solution should not be *algorithm – based*
Architecture

• **Modules:**
  - ParallelInstances *Class*
  - ParallelUtils *Class*
  - ParallelMatrix *Class*
  - ParallelKMeans *Class* (as a *proof of concept* for parallelization)
TOOLS

eclipse + Subversion = Subclipse

PJ Parallel Java Library + Scientific Linux
What We Have Done So Far

- **End of First Semestre**:
  - Parallelization with *Mpijava* – 1st Sem. Demo
  - Problems of *Mpijava*
    - Not stable for communication
    - Giving Unnatural results

- **February**:
  - Switched to PJ
  - Design has been reviewed
What We Have Done So Far  

**March :**
- **1st - 3rd** Weeks: Try to parallelize **Instances** class.
  - Class for handling an ordered set of weighted instances (data).
- Changes in Weka.core Package
  - Improvements not noticeable, sometimes it takes longer time. (Then, we decided to enlarge our inputs.)

**April :**
- **1st - 3rd** Weeks: Try to parallelize **Utils** class.
  - Commonly Used Functions like mean, sort, sum, variance ..
- Some parts are canceled due to unexpected results.
What We Have Done So Far (cont.)

• April (cont.):
  – For mobility, tried to run PJ on our multicore laptops.
    • Bug-like problems occurred (still no feedback from maintainer)
    • Considered as a waste of time, switched back to NAR.
  – 4^{th} Week April: Started to work on parallelizing Matrix class.
    • Dealing with LU Decomposition and Cholesky Decomposition.
      \(O(n^2), O(n^3)\) algorithmic complexities
    • Weka code of *Decompositions written in a way not suitable for parallelization (unable to distribute data in an efficient way)
What We Have Done So Far (cont.)

• *May:*
  – *2*\(^{nd}\) - *3*\(^{rd}\) *Weeks*: Try to parallelize `KMeans` class.
    • A chance to see our improvements vastly, succeeded.
    • *Why KMeans?* Used in other classes (i.e. E.M.) too.
    • Reconsidering future plans
    • Making the package ready for a release
Main Problems Encountered

The #1 programmer excuse for legitimately slacking off:

"My code's compiling."

Hey! Get back to work!

Compiling!

Oh. Carry on.
Main Problems Encountered

• Some Code Snippets with \(O(n)\) complexity, needs huge amount of data.
• It takes a long time to test the changes.
• Eclipse and JVM's memory limits for huge amount of data.
• Hard to find huge amount of data; when found, not so easy to move them into NAR. Usually created by hand.
• Some parts of Weka code (i.e. Decomposition in Matrix Class) are not written
Project Status

• ParallelInstances Class
  – Design and Coding = 100 %, Testing = 100 %

• ParallelUtils Class
  – Design and Coding = 95 %, Testing = 90 %

• ParallelMatrix Class
  – Design and Coding = 80 %, Testing = 80 %

• ParallelKMeans Class
  – Design and Coding = 100 %, Testing = 80 %
# Test Results

<table>
<thead>
<tr>
<th>K – means with 2.5K instance and 2K attribute</th>
<th>Sequential Version</th>
<th>60 s.</th>
<th>Parallel Version (8-cores)</th>
<th>45 s.</th>
</tr>
</thead>
<tbody>
<tr>
<td>K – means with 10K instance and 4K attribute</td>
<td>450 s.</td>
<td></td>
<td>260 s.</td>
<td></td>
</tr>
<tr>
<td>Sorting 1 M data</td>
<td>370 ms</td>
<td></td>
<td>260 ms</td>
<td></td>
</tr>
<tr>
<td>Sorting 2 M data</td>
<td>820 ms</td>
<td></td>
<td>410 ms</td>
<td></td>
</tr>
</tbody>
</table>
What's Next

• Emphasizing and documenting Testing process
• API Documentation (similar to/integrated into) WEKA's
• Trying Naive-Bayesian *proof-of-concept* parallelization as an additional example.
• Making the project open-source
Thanks for Listening...

A.Q.?