Accelerating Medoids-based Clustering with the Intel Many Integrated Core Architecture

Timofey Rechkalov, Mikhail Zymbler
South Ural State University, Russia

The reported study was supported by "Participant of Youth Scientific Innovational Competition" UMNIK program.
Intel Xeon Phi
Partitioning Clustering
PAM properties

• **PAM algorithm** (*Partitioning Around Medoids*) – partitioning clustering algorithm which selects cluster centers among clustered objects

• Such objects called *medoids*

• Iteration time complexity is $O(k(n-k)^2)$, where
  • $n$ is the number of clustered objects
  • $k$ is the number of clusters
Objective function

- Objective function

\[ E = \sum_{j=1}^{n} \min_{1 \leq i \leq k} \rho(c_i, o_j) \]

where \( c_i \) is the medoid, \( o_j \) is the clustered object, \( \rho \) is the distance metric.
PAM pseudocode

Input: Set of objects $O$, number of clusters $k$
Output: Set of clusters $C$

1. Initialize $C$; // BUILD phase
2. repeat // SWAP phase
3. Find best swapping estimation $T_{min}$;
4. Swap $c_{min}$ and $o_{min}$, determined by $T_{min}$;
5. until $T_{min} < 0$;
Calculating distance matrix

![Diagram of a distance matrix]

<table>
<thead>
<tr>
<th></th>
<th>$o_1$</th>
<th>$o_2$</th>
<th>$o_3$</th>
<th>...</th>
<th>$o_n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$o_1$</td>
<td>$\rho(o_1, o_1)$</td>
<td>$\rho(o_1, o_2)$</td>
<td>$\rho(o_1, o_3)$</td>
<td>...</td>
<td>$\rho(o_1, o_n)$</td>
</tr>
<tr>
<td>$o_2$</td>
<td>$\rho(o_2, o_1)$</td>
<td>$\rho(o_2, o_2)$</td>
<td>$\rho(o_2, o_3)$</td>
<td>...</td>
<td>$\rho(o_2, o_n)$</td>
</tr>
<tr>
<td>$o_3$</td>
<td>$\rho(o_3, o_1)$</td>
<td>$\rho(o_3, o_2)$</td>
<td>$\rho(o_3, o_3)$</td>
<td>...</td>
<td>$\rho(o_3, o_n)$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$o_n$</td>
<td>$\rho(o_n, o_1)$</td>
<td>$\rho(o_n, o_2)$</td>
<td>$\rho(o_n, o_3)$</td>
<td>...</td>
<td>$\rho(o_n, o_n)$</td>
</tr>
</tbody>
</table>
BUILD phase

$k=3$

Time complexity $O(kn^2)$
SWAP phase

\[ E = 1,014 \quad \rightarrow \quad E = 0,865 \]

Time complexity \( O(k(n - k)^2) \) per iteration
Used Optimizations

• Parallelizing with OpenMP
• Loops with arithmetic operations were reorganized for vectorized execution
  – Data consists of 32 element blocks
• Tiling for better locality and cache performance
PAM implementation

Input: Set of objects $O$, number of clusters $k$
Output: Set of clusters $C$

1. $M \leftarrow \text{PrepareDistanceMatrix}(O)$;
2. $C \leftarrow \text{BuildMedoids}(M)$; // BUILD phase
3. repeat // SWAP phase
   4. $T_{\text{min}} \leftarrow \text{FindBestSwap}(M, C)$;
   5. Swap $c_{\text{min}}$ and $o_{\text{min}}$, determined by $T_{\text{min}}$;
4. until $T_{\text{min}} < 0$;
Experimental evaluation

• Hardware
  – Intel Xeon Phi 60 cores
  – Intel Xeon 12 cores

• Parameters
  – Data type: float
  – Intel Xeon Phi mode: offload

• Purpose
  – Compare work time of PAM algorithm on CPU and Intel Xeon Phi
## Dataset properties

<table>
<thead>
<tr>
<th>Dataset</th>
<th>p</th>
<th>k</th>
<th>( n \times 2^{10} )</th>
<th>\text{min}</th>
<th>\text{max}</th>
</tr>
</thead>
<tbody>
<tr>
<td>FCS Human</td>
<td>423</td>
<td>10</td>
<td></td>
<td>2</td>
<td>18</td>
</tr>
<tr>
<td>Corel Image Histogram</td>
<td>32</td>
<td>10</td>
<td></td>
<td>5</td>
<td>35</td>
</tr>
</tbody>
</table>

- \( p \) – size of real-valued tuple which describes clustering object
- \( k \) – the number of clusters
- \( n \) – the number of clustering objects
FCS Human evaluation
Corel Image Histogram evaluation

![Graph showing execution time vs number of objects for different configurations.](image_url)
Conclusion

• The paper has described a parallel version of Partitioning Around Medoids clustering algorithm for the Intel Xeon Phi many-core coprocessor
  – OpenMP
  – Vectorization
  – Tiling
• Experimental results show effectiveness of suggested approach
• Experiments show that PAM performance depends on clustered data nature