UMPa: A Multi-objective, multi-level partitioner for communication minimization

Ümit V. Çatalyürek\textsuperscript{1}, Mehmet Deveci\textsuperscript{1}, Kamer Kaya\textsuperscript{1}, Bora Uçar\textsuperscript{2}

\textsuperscript{1}Department of Biomedical Informatics, The Ohio State University
\textsuperscript{2}LIP, ENS Lyon
Introduction

• Problem: distributing communicating tasks, modeled as a graph, among processing units.
  • Balanced load distribution
  • Good communication pattern

• Objective functions from the literature:
  • Total communication volume
  • Maximum communication volume
  • Maximum send volume
Problem: Input and objective

- **Input**: task graph $G = (V, E)$
  - $V$: vertex set representing a set of tasks
  - $E$: edge set representing task communications
- **Objective**: Find a partition $\prod = \{P_1, P_2, \ldots, P_K\}$ of the tasks s.t.
  $$\max_k \left( \sum_{v \in P_k} c(v) \times f(v) \right)$$
is minimized
- $c(v)$: volume of each transfer sent by $v$.
- $f(v)$: number of parts that requires the data sent by $v$. 
Problem: communication costs

- The objective function is equivalent to minimizing maximum send volume (maxSV).

\[ c(v_1) = 2 \]
\[ c(v_2) = 1 \]
\[ c(v_3) = 3 \]
\[ c(v_4) = 2 \]
\[ c(v_5) = 4 \]

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>P_1</td>
<td>3 + 2</td>
<td>2 + 1</td>
<td>8</td>
</tr>
<tr>
<td>P_2</td>
<td>2 + 1 + 1</td>
<td>3 + 2 + 4</td>
<td>13</td>
</tr>
<tr>
<td>P_3</td>
<td>4</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>Total</td>
<td>13</td>
<td>13</td>
<td>26</td>
</tr>
</tbody>
</table>

- \( \text{maxSRV} \leftarrow \text{max send+receive volume} \)
- \( \text{totV} \leftarrow \text{total communication volume} \)
Hypergraph Model

• **Hypergraph** \( H = (V, N) \)
  - A net is a subset of vertices.
  - Each net \( n \) has cost \( c_H(n) \)

• We model the task graph \( G(V,E) \) as a hypergraph
  - For each task \( s \) in \( G \), let \( v_s \) be the corresponding vertex in \( H \).
  - For each task \( s \) in \( G \), the net set \( N \) contains a net \( n_s \).

  - \( n_s = \{ v_d : ((s,d) \in E) \} \cup \{ v_s \} \)

  - \( c_H(n_s) = c(s) \)
• $\lambda_n$: Connectivity of a net $n$, i.e., the number of parts net $n$ is connected.

\[
tot V = \sum_{n \in N} c_H(n) \times (\lambda_n - 1)
\]

Minimizing the formula, equivalent to minimizing the total communication volume [Ç & Aykanat’99].
Directed Hypergraph Model

- Directed hypergraph:
  - Flow: from the source pin to the other pins.
  - Source of $n_s = v_s$
  - Allows to minimize maxSV and maxSRV (in addition to totV).
- Objective: Partition the vertices into $K$ parts s.t.
  - The load is distributed equally.
  - $W_k < W_{avg}(1 + \varepsilon)$ for $1 \leq k \leq K$
    - maxSV is minimized.
      - $SV(P) = \sum_{v_s \in P} c_H(n_s) \times (\lambda_n - 1)$
      - $\max SV = \max_k (SV(P_k))$
Hypergraph Model: Example

$\lambda_{n_2} = 3, \lambda_{n_1} = \lambda_{n_3} = \lambda_{n_4} = \lambda_{n_5} = 2$
Multi-level Approach

- Three phases:
  - Coarsening: obtain smaller and similar hypergraphs to the original, until either a minimum vertex count is reached or reduction on vertex number is lower than a threshold.
  - Initial Partitioning: find solution for the smallest hypergraph
  - Uncoarsening: Project the initial solution to the coarser graphs and refine it iteratively until a solution for the original hypergraph obtained.
Multi-level Approach

• Most of the available tools adapt multi-level approach with recursive bisection method.
  • A partition is obtained by recursive partitioning into 2 parts.
  • Works fine for total communication.
  • May not be suitable for minimizing maximum send (and/or send+receive) volume.
    • Only the information about 2 parts is available at each step.
    • Send and receive volumes of other parts are unknown.
• K-way multi-level partitioner.
• Uses directed hypergraph model.
  • Communication of the net flows from source to target vertices.
• Minimizes $\text{maxSV}$, while breaking ties by favoring reducing $\text{maxSRV}$, then the total volume.
• Currently, only coarsening phase of UMPa is shared memory parallel.
• Ultimate goal: To parallelize UMPa (MPI+OpenMP).
• Neighbor vertices (u and v) are clustered by using agglomerative matching in coarsening phase.
  • Similarity of u and v

\[
\sum_{n \in \text{nets}(u) \cap \text{nets}(v)} \frac{c_H(n)}{(p(n) - 1)}
\]
• For the initial partitioning, we used PaToH to obtain \( k \) initial parts.
  • Although PaToH is used to minimize total communication volume but not maximum send volume:
    • We do not want a drastic increase in any of the communication metrics. So, minimizing total volume is a good idea both in theory and practice.
    • Using recursive bisection and FM-based improvement are favorable due to the small net sizes and high vertex degrees.
UMPa: Uncoarsening

- Solutions for the coarser hypergraphs are iteratively projected to finer ones and refined.
- Refinement method:
  - Traverses the boundary vertices in random order.
    - Random, since FM/KL based heuristics are expensive especially in K-way.
    - Computes move gains for each visited vertex and select the best move.

```
Data: \( \mathcal{H} = (\mathcal{V}, \mathcal{N}) \), boundary[], part[], SV[], SRV[]
for each unlocked \( u \in \text{boundary} \) do
  \((\text{bestMaxSV}, \text{bestMaxSRV}, \text{bestTotV}) \leftarrow (\text{maxSV}, \text{maxSRV}, \text{totV})\)
  \text{bestPart} \leftarrow \text{part}[u]
  for each part \( p \) other than \( \text{part}[u] \) do
    if \( p \) has enough space for vertex \( u \) then
      \((\text{SV}[], \text{SRV}[], \text{moveV}) \leftarrow \text{calculateComVols}(v, p)\)
      \((\text{moveMaxSV}, \text{moveMaxSRV}) \leftarrow \text{calculateMax}(\text{moveSV}[], \text{moveSRV}[])\)
      \text{MOVESELECT}(\text{moveMaxSV}, \text{moveMaxSRV}, \text{moveV}, p, \text{bestMaxSV}, \text{bestMaxSRV}, \text{bestTotV}, \text{bestPart})\)
  if \text{bestPart} \neq \text{part}[u] then
    move \( u \) to \text{bestPart} and update data structures accordingly
```
Move Selection

- Always move a visited vertex to the part with the maximum reduction on maxSV.
  - Tie-breaking is applied for equal reductions.
  - When there is an equality, the vertex move is favored toward the part that minimizes maxSRV, then totV.
• Initially, maxSV=6, maxSRV=9, totV=12. (v_3, v_4, v_6, v_7)
Experimental Results

• Experiments
  • 123 graphs
  • 10 graph classes
  • For $K = 4, 16, 64, 256$
  • Each instance is partitioned 10 times.

• Compared with PaToH minimizing total volume.
• The power of tie-breaking is also studied.
The geometric mean of the relative results wrt PaToH used to minimize totV.

Tie-breaking is very useful.

As K increases the reduction rate decreases, since the total communication is distributed to more parts.
## Experiment Results: K=16

- **Best and worst improvements for each graph class normalized w.r.t. PaToH.**
- **78% (75%, 67%) improvement on maxSV for ut2010.**

<table>
<thead>
<tr>
<th>Graph</th>
<th>$maxSV$</th>
<th>$maxSRV$</th>
<th>$\text{totV}$</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>coPapersDBLP</td>
<td>0.862</td>
<td>0.845</td>
<td>1.252</td>
<td>1.591</td>
</tr>
<tr>
<td>as-22july06</td>
<td>0.760</td>
<td>0.787</td>
<td>1.016</td>
<td>4.286</td>
</tr>
<tr>
<td>road_central</td>
<td>0.558</td>
<td>0.577</td>
<td>0.716</td>
<td>0.247</td>
</tr>
<tr>
<td>smallworld</td>
<td>0.907</td>
<td>0.909</td>
<td>0.928</td>
<td>6.236</td>
</tr>
<tr>
<td>delaunay_n14</td>
<td>0.966</td>
<td>1.004</td>
<td>1.019</td>
<td>4.632</td>
</tr>
<tr>
<td>delaunay_n17</td>
<td>0.917</td>
<td>0.928</td>
<td>1.033</td>
<td>2.330</td>
</tr>
<tr>
<td>hugetrace-00010</td>
<td>0.980</td>
<td>0.981</td>
<td>1.107</td>
<td>0.462</td>
</tr>
<tr>
<td>hugetric-00020</td>
<td>0.964</td>
<td>0.964</td>
<td>1.075</td>
<td>0.484</td>
</tr>
<tr>
<td>venturiLevel3</td>
<td>0.924</td>
<td>0.925</td>
<td>1.072</td>
<td>0.584</td>
</tr>
<tr>
<td>adaptive</td>
<td>0.944</td>
<td>0.945</td>
<td>1.062</td>
<td>0.543</td>
</tr>
<tr>
<td>rgg_n_2.15_s0</td>
<td>0.815</td>
<td>0.867</td>
<td>0.982</td>
<td>3.029</td>
</tr>
<tr>
<td>rgg_n_2.21_s0</td>
<td>0.919</td>
<td>0.949</td>
<td>1.030</td>
<td>0.440</td>
</tr>
<tr>
<td>tn2010</td>
<td>0.838</td>
<td>1.062</td>
<td>4.214</td>
<td>1.222</td>
</tr>
<tr>
<td>ut2010</td>
<td>0.219</td>
<td>0.253</td>
<td>0.328</td>
<td>1.907</td>
</tr>
<tr>
<td>af_shell9</td>
<td>0.987</td>
<td>0.986</td>
<td>1.065</td>
<td>0.583</td>
</tr>
<tr>
<td>audikw1</td>
<td>0.787</td>
<td>0.826</td>
<td>1.094</td>
<td>0.479</td>
</tr>
<tr>
<td>asia.osm</td>
<td>0.476</td>
<td>0.496</td>
<td>0.790</td>
<td>0.190</td>
</tr>
<tr>
<td>belgium.osm</td>
<td>0.855</td>
<td>0.865</td>
<td>0.990</td>
<td>0.408</td>
</tr>
<tr>
<td>memplus</td>
<td>0.696</td>
<td>0.522</td>
<td>1.267</td>
<td>3.130</td>
</tr>
<tr>
<td>t60k</td>
<td>0.958</td>
<td>0.958</td>
<td>1.055</td>
<td>3.414</td>
</tr>
</tbody>
</table>
### Experiment Results: K=256

- 50% improvement on maxSV and maxSRV for memplus although total volume increases by 26%.

<table>
<thead>
<tr>
<th>Graph</th>
<th>maxSV</th>
<th>maxSRV</th>
<th>totV</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>coPapersCiteseeer</td>
<td>0.694</td>
<td>0.693</td>
<td>1.005</td>
<td>2.937</td>
</tr>
<tr>
<td>coPapersDBLP</td>
<td>0.730</td>
<td>0.690</td>
<td>0.972</td>
<td>7.216</td>
</tr>
<tr>
<td>as-22july06</td>
<td>0.397</td>
<td>0.637</td>
<td>1.181</td>
<td>12.495</td>
</tr>
<tr>
<td>smallworld</td>
<td>0.839</td>
<td>0.843</td>
<td>0.899</td>
<td>7.965</td>
</tr>
<tr>
<td>delaunay_n20</td>
<td>0.927</td>
<td>0.943</td>
<td>1.024</td>
<td>3.829</td>
</tr>
<tr>
<td>delaunay_n21</td>
<td>0.948</td>
<td>0.963</td>
<td>1.033</td>
<td>3.066</td>
</tr>
<tr>
<td>hugetrace-00000</td>
<td>1.020</td>
<td>1.021</td>
<td>1.075</td>
<td>2.228</td>
</tr>
<tr>
<td>hugetric-00010</td>
<td>0.950</td>
<td>0.951</td>
<td>1.065</td>
<td>1.814</td>
</tr>
<tr>
<td>adaptive</td>
<td>0.976</td>
<td>0.978</td>
<td>1.063</td>
<td>2.322</td>
</tr>
<tr>
<td>venturiLevel3</td>
<td>0.993</td>
<td>0.996</td>
<td>1.057</td>
<td>2.642</td>
</tr>
<tr>
<td>rgg_n_2.22_s0</td>
<td>0.906</td>
<td>0.941</td>
<td>1.010</td>
<td>1.647</td>
</tr>
<tr>
<td>rgg_n_2.23_s0</td>
<td>0.862</td>
<td>0.891</td>
<td>1.009</td>
<td>1.286</td>
</tr>
<tr>
<td>ri2010</td>
<td>0.866</td>
<td>0.965</td>
<td>0.994</td>
<td>12.028</td>
</tr>
<tr>
<td>tx2010</td>
<td>0.586</td>
<td>0.816</td>
<td>0.951</td>
<td>1.836</td>
</tr>
<tr>
<td>af_shell10</td>
<td>0.986</td>
<td>0.987</td>
<td>1.056</td>
<td>1.758</td>
</tr>
<tr>
<td>audikw1</td>
<td>0.895</td>
<td>0.917</td>
<td>1.018</td>
<td>2.551</td>
</tr>
<tr>
<td>asia.osm</td>
<td>0.917</td>
<td>0.925</td>
<td>0.989</td>
<td>0.260</td>
</tr>
<tr>
<td>great-britain.osm</td>
<td>0.788</td>
<td>0.804</td>
<td>0.997</td>
<td>0.505</td>
</tr>
<tr>
<td>finan512</td>
<td>0.965</td>
<td>1.040</td>
<td>1.022</td>
<td>10.073</td>
</tr>
<tr>
<td>memplus</td>
<td>0.509</td>
<td>0.541</td>
<td>1.264</td>
<td>16.837</td>
</tr>
</tbody>
</table>
- K-way partitioners are costly due to the complexity of the refinement heuristic for maxSV.
- UMPa gets slower when the number of parts is large.
• Proposed a directed hypergraph model to minimize maxSV, maxSRV and totV.
• We developed a multi-level, K-way partitioner, UMPa.
• Employed a tie-breaking scheme to handle multiple communication metrics.
• Currently, UMPa is parallel (shared memory) at coarsening phase.
• Parallelizing & speeding up UMPa and the proposed refinement approach.
Thanks

• For more information visit
  • http://bmi.osu.edu/hpc

• Research at the HPC Lab is funded by
for each $n \in \text{nets}[u]$ do
  if $s(n) = u$ then
    sendGain[part[u]] $\leftarrow$ sendGain[part[u]] $+$ $(\lambda_n - 1)c[n]$  
    if $\Lambda(n, \text{part}[u]) > 1$ then
      receiveGain $\leftarrow$ receiveGain $-$ $c[n]$  
      uToPartU $\leftarrow$ uToPartU $+$ $c[n]$ 
  else if $\Lambda(n, \text{part}[u]) = 1$ then
    sendGain[part[s(n)]] $\leftarrow$ sendGain[part[s(n)]] $+$ $c[n]$  
    receiveGain $\leftarrow$ receiveGain $+$ $c[n]$
for each part $p$ other than part[$u$] do
  if $p$ has enough space for vertex $u$ then
    $receiveLoss \leftarrow 0$
    $sendLoss[] \leftarrow 0$
    $sendLoss[p] \leftarrow sendGain[part[u]] + uToPartU$
  for each $n \in nets[u]$ do
    if $s(n) = u$ then
      if $\Lambda(n, p) > 0$ then
        $sendLoss[p] \leftarrow sendLoss[p] - c[n]$
        $receiveLoss \leftarrow receiveLoss - c[n]$
      else if $\Lambda(n, p) = 0$ then
        $sendLoss[part[s(n)]] \leftarrow sendLoss[part[s(n)]] + c[n]$
        $receiveLoss \leftarrow receiveLoss + c[n]$
    $(moveSV, moveSRV) \leftarrow (-\infty, -\infty)$
  for each part $q$ do
    $\Delta_S \leftarrow sendLoss[q] - sendGain[q]$
    $\Delta_R \leftarrow 0$
    if $q = part[u]$ then
      $\Delta_R \leftarrow receiveGain$
    else if $q = p$ then
      $\Delta_R \leftarrow receiveLoss$
    $moveSV \leftarrow \max(moveSV, SV[q] + \Delta_S)$
    $moveSRV \leftarrow \max(moveSRV, SV[q] + \Delta_S + RV[q] + \Delta_R)$
  $moveV \leftarrow totV + receiveLoss - receiveGain$
  $MoveSelect(moveSV, moveSRV, moveV, p,$
  bestMaxSV, bestMaxSRV, bestTotV, bestPart)$