HW/SW Codesign

Exercise 4:
Mapping and Partitioning (2/2)

25. October 2017

Andres Gomez
andres.gomez@tik.ee.ethz.ch
Partitioning

- Partitioning problem is to divide a set of objects into mutually exclusive blocks (see formal definition in lecture slides)

- Several methods – ILP, random, hierarchical clustering, Kernighan-Lin algorithm, simulated annealing, Evolutionary algorithms

- Partitioning is a key step in binding decisions
  - What to run on software (RISC processor) and what to run on hardware (specialized co-processors)?
  - How to bind tasks on a multicore processor?
  - How to implement a given behavior on a FPGA?
Hierarchical clustering

• Define a **closeness function** between every pair of nodes

• Nodes that are close are good candidates for clustering into same partition

• Method:
  – Iteratively, we cluster two **closest** nodes and **appropriately modify** the graph
  – After all steps, we **decide the cut-level** and generate the partition
Hierarchical clustering: lecture example

1. Merge two closest nodes
2. Modify the graph by changing the new weights using arithmetic mean
3. Repeat process till done
Hierarchical clustering: lecture example

- Choose cut line and generate partitioning
- Another art for real problems

step 1:

- \( v_1 \)
- \( v_2 \)
- \( v_3 \)
- \( v_4 \)

step 2:

- \( v_1 \cap v_2 \)
- \( v_3 \cap v_4 \)

step 3:

- \( v_1 \cap v_2 \cap v_3 \cap v_4 \)
Closer look at closeness function

• Question: Given two objects O1 and O2, what should closeness between these two objects denote?

• Hint: We are designing a system to optimize some objective function say $f$

• Answer: It should denote the relative benefit of clustering O1 and O2 in terms of $f$, when compared to not clustering O1 and O2.

• Question: What happens when there are multiple objective functions?

• Answer: Different ways to deal with that. For now, lets say we take a linear combination of the different objective functions.
Ready to answer 4.1.a)

\[
Closeness(o_i, o_j) = \left( \frac{Conn(o_i, o_j)}{TotalConn(o_i, o_j)} \right) + \frac{FU_{cost}(o_i) + FU_{cost}(o_j) - FUGROUP_{cost}(o_i, o_j)}{FUGROUP_{cost}(o_i, o_j)}
\]

- Interpret, physically, this closeness function
Hierarchical clustering and bi-partitioning

- Given dataflow graph, given closeness function
  1. Compute closeness function
  2. Hierarchically cluster to obtain a bi-partition

Partition into two blocks. (Decide cut-level so that you have only two blocks)
4.1)

\[ Closeness(o_i, o_j) = \left( \frac{Conn(o_i, o_j)}{TotalConn(o_i, o_j)} \right) + \left( \frac{FU_{cost}(o_i) + FU_{cost}(o_j) - FUGROUPP_{cost}(o_i, o_j)}{FUGROUPP_{cost}(o_i, o_j)} \right) \]

- Physical interpretation
- Clustering

\[ c(o_1, o_2) = \frac{8}{50} + \frac{2 - 1}{1} = \frac{14}{25} = 1.16 \]
\[ c(o_1, o_3) = \frac{17}{52} + \frac{2 - 1}{1} = \frac{17}{52} = 1.327 \]
\[ c(o_1, o_4) = \frac{9}{52} + \frac{2 - 2}{2} = \frac{9}{52} = 0.173 \]
\[ c(o_2, o_3) = 0 + \frac{2 - 1}{1} = 1 \]
\[ c(o_2, o_4) = \frac{9}{52} + \frac{2 - 2}{2} = \frac{9}{52} = 0.173 \]
\[ c(o_3, o_4) = \frac{9}{54} + \frac{2 - 2}{2} = \frac{1}{6} = 0.167 \]

Clustering 1 and 3

Clustering 13 and 2

Solution : (123)(4)
H/W-S/W partitioning

- What does it mean?
Question 4.2)

- Given
  - A dataflow graph
  - Some delay numbers
  - Some cost functions
  - Some nice algorithm
- To
  - Partition the nodes into either hardware or software
Question 4.2

- Let's read through the long question.

4.2 HW/SW partitioning

Fig. 2 shows the pseudo code of a greedy algorithm for HW/SW partitioning. The algorithm starts with a partition where all objects are realized in hardware. Then, objects are migrated to software as long as the performance requirement is satisfied (function SatisfiesPerformance) and the cost of the new partitioning is lower (function f). If an object is migrated, the algorithm also tries to migrate all successor nodes (function Successors).

• Apply the algorithm to the sequence graph shown in Fig. 3. The function SatisfiesPerformance(P) should return TRUE if P satisfies the latency bound \( L < 7 \). It determines the latency of a partitioning, you have to construct a valid schedule. The execution times of start- and end nodes of the sequence graph are 0, all other node execution times are given in Fig. 3, split into HW (\( d_{HW} \)) and SW (\( d_{SW} \)). For a communication between HW and SW, a delay of 0.5 per edge has to be accounted for. This communication delay always adds to the latency, since it requires the processor to be idle during this time. For HW nodes there are no resource constraints, i.e., all ready nodes can be executed in parallel. The SW nodes have to share one processor. The function f determines the cost. For a SW node the cost is 0, and for a HW node the cost is 1.

\[
P = \{\{\}, O\}; /* all in HW */
\]

PROCEDURE PARTITIONING
REPEAT
\[
P_{old} = P;
\]
FOR ALL \( o_i \in HW \)
\[
\text{AttemptMove}(P, o_i);
\]
ENDFOR
UNTIL \( (P == P_{old}) \)
END PROCEDURE

PROCEDURE AttemptMove(P, o_x)
IF SatisfiesPerformance(Move(P, o_x)) AND (f(Move(P, o_x))) < f(P)
\[
P = \text{Move}(P, o_x);
\]
FOR ALL \( o_y \in \text{Successors}(o_x) \)
\[
\text{AttemptMove}(P, o_y);
\]
ENDFOR
ENDPROCEDURE

Figure 2: Pseudo code for a greedy HW/SW partitioner.
4.2)

- One possible solution: $SW = (\text{op1}, \text{op3}, \text{op4}, \text{op5})$
  $HW = (\text{op2}, \text{op6}, \text{op7}, \text{op8}, \text{op9}, \text{op1}, \text{op11})$

<table>
<thead>
<tr>
<th>Latency</th>
<th>Cost</th>
<th>Tasks in SW</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>L0 $= 1+1+1+1 = 4$</td>
<td>C0 $= 11$</td>
<td>-</td>
<td>move op1 to SW</td>
</tr>
<tr>
<td>L1 $= 2+0.5+1+1+1 = 5.5$</td>
<td>C1 $= 10$</td>
<td>op1</td>
<td>move op3 to SW</td>
</tr>
<tr>
<td>L2 $= 2+0.5+2+0.5+1+1 = 7$</td>
<td>C2 $= 9$</td>
<td>op1,op3</td>
<td>move op4 to SW</td>
</tr>
<tr>
<td>L3 $= 2+0.5+2+1+0.5+1 = 7$</td>
<td>C3 $= 8$</td>
<td>op1,op3,op4</td>
<td>move op5 to SW</td>
</tr>
<tr>
<td>L4 $= 2+0.5+2+1+0.5+1 = 7$</td>
<td>C4 $= 7$</td>
<td>op1,op3,op4,op5</td>
<td>try to move op2 to SW</td>
</tr>
<tr>
<td>L5 $= 2+2+2+1+0.5+1 = 8.5$</td>
<td>C5 $= 6$</td>
<td>op1,op3,op4,op5,op2</td>
<td>Keep old partitions. No other element can be moved to software without violating constraint.</td>
</tr>
</tbody>
</table>