ILP formulation **July 2015**

FlexPipe — ILP compiler for P4

Lisa Yan (yanlisa@stanford.edu)

Problem overview

The compiler problem is to optimally fit a P4 program into the FlexPipe switch.

- **Program**: Please refer to the RMT document for more information about the P4 program.
- **Switch**: The FlexPipe switch is a series of five pipeline stages, each with a different type of memory. The first stage is a ”Mapper” stage that can support exact match only. There are two FFU stages which are ternary memory types (can support wildcard, prefix, and exact match), a BST (Binary Search Tree) stage for prefix matches, and a large hash table stage supporting exact match only. One FFU stage and the BST stage are executed in parallel. Finally, the notion of multiple packing factors does not exist in FlexPipe; blocks can only be strung together in the minimum amount to match on wide words. For example, a 140 bit word requires four 40-bit memory blocks to be strung together. However, FlexPipe has the notion of table sharing within each memory block. Unlike RMT, which assigns an entire block to one table, multiple tables can occur within a block, provided the tables are not matched simultaneously. For example, IPv6 and IPv4 forwarding tables could share blocks, if packets are exclusively IPv4, IPv6, or neither. We define the number of tables sharing a given block as the *order* of a table; an order of three implies three tables share a block, and so on. More information is shown in Figure 8.
- **Action memory and Crossbars**: FlexPipe does not have these features.

Notation

- Logical tables: $l \in \{1, \cdots, N\}$, where N is the number of tables in the P4 program
- Table dimension: (f_l, e_l, a_l), differing a bit from the paper. f_l is the width of match entry, e_l is the number of match entries, and a_l is the width of action entries.
- Packing factor: In FlexPipe, words cannot be packed together to create entries of larger widths; the packing factor for all tables is 1, and thus the packing units for each table l are consisted of the same number of blocks, which is the minimum blocks to string together to fit a single word of width f_l in a memory type m. The preprocessor function $f_U(l, m)$ returns the number of blocks of memory type m that entries from

table l would occupy. For example, if $f_l = 140$ bits and memory type m has blocks of width 40 bits, $f_U(l, m) = 4$.

- Memory types: $m \in \{1, \cdots m_{max}\}$ for $m_{max} = 4$ memory types, where Mapper, FFU, BST, and Hash Table are indexed 1, 2, 3, 4 respectively.
- Block index: $b \in \{1, \cdots, \widetilde{b}_m\}$, where there are $\widetilde{b}_m$ blocks of memory type m across all stages. Thus, suppose stage 1 and 2 both have b_m blocks of memory type m. Then the first block in stage 1 would be indexed by $b = 1$, whereas the first block in stage 2 would be indexed by $b = b_m + 1$. If no other stages have blocks of memory type m, $b = \widetilde{b}_m$ is the last block of memory type m in stage 2. The set of blocks of type m that are contained in stage s can be found via the preprocessor by calling $f_b(s)$.
- Stages: $s \in \{1, \cdots, M\}$, where there are $M = 5$ stages in the RMT pipeline.
- Block dimension: (b_m, w_m, d_m), where for a stage s and memory m, b_m is the number of blocks, w_m is the width of these blocks, and d_m is the depth (number of entries) each block holds.
- Order: $\theta \in \{1, \cdots, \theta_{max}\}$ refers to the θ-th table assigned in a block, independent of table name. Order allows us to find that table l is the first table in one block, but could be the second table in a different block. Refer to Figure 8 in the paper for details.
- Binary and aggregation variables: $\hat{U}_{l,s,m}$, $\widetilde{U}_{l,s}$, and $\hat{\widetilde{U}}_{l,s}$ for example. $U_{l,s,m}$ is the number of blocks assigned to table l in stage s, memory m, and $\hat{U}_{l,s,m}$ designates a boolean of whether or not there are any blocks assigned in memory m; $\hat{U}_{l,s,m} = \mathbb{1}\{U_{l,s,m}\}$. $\widetilde{U}_{l,s}$ is the total blocks assigned for table l to any memory in stage s, and $\hat{\widetilde{U}}_{l,s}$ is a boolean of whether any blocks are assigned to any memory; $\hat{\widetilde{U}}_{l,s} = \mathbb{1}\{\widetilde{U}_{l,s}\} = \mathbb{1}\{m \in \{1, 2\} : \hat{U}_{l,s,m}\}$.
- Product variables: $P^{U,S}_{l,m}$ indicates the product of U and S variables, both of which have reference subscripts l and m.
- Lower bound variables: $R^{\geq n}$ is an indicator on a variable R. If $R \neq 0$, then it *must* satisfy the lower bound $R \geq n$. $R^{\geq n}$ indicates whether or not the lower bound is satisfied; if $R^{\geq n} = 0$, then R necessarily is zero, and if $R^{\geq n} = 1$, then $R \geq n$. Constraints link these two variables.
- Dot notation: In the paper, we have start row variables as $\hat{\cdot}$, but this overlaps with our binary operator. So we use the $\dot{cdot}$ notation to represent start rows instead.

Variables

- Block assignment variables.
  Count:
  - $U_{s,l,m}$ blocks, the number of blocks in stage s dedicated to table l.
  - $W_{s,l,m}$ words, the maximum number of match entries of table l in stage s based on the assignment $U_{s,l,m}$.
  - $\widetilde{B}_{s,m}$ total blocks assigned to any table in stage s, memory type m.
  - Binary block variable per logical table: $\hat{\widetilde{U}}_{s,l}$ whether any blocks are assigned to table l in any memory type in stage s.
  - Binary block variable per memory: $\hat{\widetilde{B}}_{s,m}$ whether any blocks of any table l are assigned to memory type m in stage s.
- Row assignment variables. Refer to the Notation section for an explanation of block indexing.
  Count:
  - $R_{l,m,b}$ rows, the number of rows in block b (memory type m) of words of table l *that start* in this block. For example, suppose 120-bit wide table l has 30 words that start in block 1 of type m (40 bits wide) and continue across blocks 2 and 3. Then $R_{l,m,1} = 30, R_{l,m,2} = R_{l,m,3} = 0$ if blocks 2 and 3 are empty otherwise. Note that this notation is $r_{l,m,b}$ in the paper, but we switch to uppercase for variables where possible.
  - $\dot{R}_{l,m,b}$-th row , the "start row", or the index of the first of the $R_{l,m,b}$ rows of table l in block b of memory type m. So all the words of table l that start in block b are grouped together, and the first row of b that contains such a word is indexed by $\dot{R}_{b,l,m}$. This notation is $\hat{r}$ in the paper, which overloads the binary $\hat{\cdot}$ representation we use for this documentation.
  - $r_{l,m,b}$ rows, the number of rows in block b (memory type m) that contain *any part* of a word of table l. Continuing our first example, if 120-bit wide table l has 30 words that start in block 1 of type m (40 bits wide) and continue across blocks 2 and 3, then $r_{l,m,1} = r_{l,m,2} = r_{l,m,3} = 30$ if all blocks are empty otherwise. Note that $\forall b, r_{l,m,b} \geq R_{l,m,b}$. In our model, we focus primarily on finding values for $R_{l,m,b}$, of which $r_{l,m,b}$ is a function, and is necessary for certain computations.
  - $\dot{r}_{l,m,b}$-th row, the smallest row index in block b that contains any part of a word from table l. Note that $\forall b, \dot{r}_{l,m,b} \leq \dot{R}_{l,m,b}$.
  - $R^{\geq n}_{l,m,b}$ Indicator if lower bound n on the number of rows in a particular block is active. See Notation section for details.
  - Binary and product variables of the above.

    * $\hat{R}_{l,m,b}$ Indicator for whether there are any rows containing words of table l that start in block b.
    * $\hat{r}_{l,m,b}$ Indicator for whether there are any rows that contain parts of words of table l in block b.
    * $P^{\dot{R},\hat{R}}_{l,m,b}$ Product of start row $\dot{R}_{l,m,b}$ and $\hat{R}_{l,m,b}$, whether there are any rows containing words of table l that start in block b of memory type m.

- Order assignment variables.
  Count:
  - $\rho_{m,b,\theta}$ rows, the number of rows in block b (memory type m) of words of the θ-th table *that start* in this block.
  - $\dot{\rho}_{m,b,\theta}$-th row , the "start row", or the index of the first of the $R_{l,m,b}$ rows, of the θ-th table in block b of memory type m.
  - $L_{\theta,l,m,b}$ Indicator if the θ-th table in block b of memory type m corresponds to logical table l. We call this the *mapping* variable from order θ to table l.
  - $P^{r,L}_{\theta,l,m,b}$ Product of $r_{l,m,b}$, the number of rows that contain any part of words of table l, and $L_{\theta,l,m,b}$, the mapping of the θ-th table in block b to table l.
  - $P^{\dot{r},L}_{\theta,l,m,b}$ Product of $\dot{R}_{l,m,b}$, the earliest row of any part of words of table l, and $L_{\theta,l,m,b}$, the mapping of the θ-th table in block b to table l.
  - $\hat{\rho}_{m,b,\theta}$, indicator for whether there are any rows of the θ-th table in block b, memory type m. This variable indicates if there is a θ-th table at all in this block.
- Dependency variables. Count: $2(NM) + 2(NM)^2 = O(N^2M^2)$
  - $\hat{S}_{s,l}$, indicator if stage s is the first stage that has match memory of l, TCAM or SRAM; "start stage."
  - $\hat{E}_{s,l}$, indicator if stage s is the last stage that has match memory of l, TCAM or SRAM; "end stage".
  - $P^{\hat{S},\hat{\hat{U}}}_{s,l}$ Product of $\hat{S}_{s,l}$, whether stage s the start stage of and $\hat{\hat{U}}_{s,l}$ is whether there are any table l match memory blocks in stage s.
  - $P^{\hat{E},\hat{\hat{U}}}_{s,l}$ Product of $\hat{E}_{s,l}$, whether stage s the end stage of and $\hat{\hat{U}}_{s,l}$ is whether there are any table l match memory blocks in stage s.

Constraints

Basic constraints

These are constraints that relate the memory assignment variables to each other.

- Word layout constraints. The number of blocks of table l in stage s, memory type m, is the number of blocks that have any part of a word of table l assigned to it. On the other hand, the number of words of table l in stage s, memory type m, must be the number of unique words, so this is a count of the number of words that start in each block. Recall the preprocessor function $f_b(s)$ returns the set of blocks of type m that are contained in stage s.
  Count: $2m_{max}MN = O(m_{max}MN)$

$$U_{s,l,m} == \sum_{b \in f_b(s)} \hat{r}_{l,m,b}$$

$$W_{s,l,m} == \sum_{b \in f_b(s)} R_{l,m,b}$$

- Upper bound on $R_{l,m,b}$, the number of rows of table l's words that start in block b. If $R^{\geq n}_{l,m,b} = 0$, then necessarily the number of rows must be zero. Otherwise the bound is simply d_m, the row dimension of a block of memory type m.

$$R_{l,m,b} \leq d_m \cdot R^{\geq n}_{l,m,b}$$

- Lower bound on $R_{l,m,b}$. If $R^{\geq n}_{l,m,b} = 1$, then necessarily the number of rows must be more than n.

$$R_{l,m,b} \geq n - 2n(1 - R^{\geq}_{l,m,b})$$

- Total blocks constraint. The number of blocks assigned in a stage is summed over all logical tables.

$$\widetilde{B}_{s,m} == \sum_{l} U_{s,l,m}$$

- Memory type constraint: Each logical table has a *match type* that restricts the table's memory to a memory type. For example, a prefix IPv4 must be assigned to FFU (ternary, wildcard memory), whereas an exact-match MAC table can be assigned to any memory type. The preprocessor returns, $f_{\mathbb{1}}(l, m)$, an indicator if table l can be assigned to memory m.
  Count: $m_{max}N = O(m_{max}N)$

$$\sum_{s} U_{s,l,m} \leq (b_m M) \cdot f_{\mathbb{1}}((l, m)$$

Order constraints

Inside each block, there are interleavings of parts of words from different tables; for example, the first order could be the first 40 bits of a word from table 1, the second order could be the last 40 bits of a different table 2, and the third order could be the middle 40 bits of a third table 3. There must be constraints to ensure that different orders do not overlap.

The boolean function $f_o(m, b_1, l, b_2)$ returns true if a word that starts in b_1 would continue into b_2. This happens if the width of the word spans all blocks between b_1 and b_2. For example, given 40 bit-wide blocks of memory type m and four blocks $1, \ldots, 5$ and a 160 bit-wide table l whose start row is in block 1, $f_o(m, 1, l, b)$ will return true for blocks $b = 1, \ldots, 4$ and false for block 5. Of course, $r_o(m, 5, l, 1)$ will return false, since block 5 occurs after block 1 in the same stage.

- Ensure that each table is only in a block once. So different orders cannot correspond to the same table.
  Count:
$$\sum_{b' \in f_b(s)} \hat{R}_{l,m,b'} \cdot f_o(m, b', l, b) \leq 1$$

- Connects $\dot{R}_{l,m,b}$ to $\dot{r}_{l,m,b}$. Identify the earliest row of any start row assignment from earlier blocks in the same stage that overlaps into block b.
$$\dot{r}_{l,m,b} == \sum_{b' \in f_b(s)} P^{\dot{R},\hat{R}}_{l,m,b} * f_o(m, b', l, b)$$

- Connects $R_{l,m,b}$ to $r_{l,m,b}$. Sum up any row assignments of table l from earlier blocks in the same stage that overlap into block b.
$$r_{l,m,b} == \sum_{b' \in f_b(s)} R_{l,m,b} * f_o(m, b', l, b)$$

- Constrain the order to log mapping to be one-to-one.
  Count:
$$\sum_{\theta} L_{\theta,l,m,b} == \hat{R}_{l,m,b}$$
$$\sum_{l} L_{\theta,l,m,b} \leq 1$$

- Constrain each order to have the same start row and number of rows as the corresponding logical table.
$$\dot{\rho}_{m,b,\theta} == \sum_{l} P^{\dot{r},L}_{\theta,l,m,b}$$
$$\rho_{m,b,\theta} == \sum_{l} P^{r,L}_{\theta,l,m,b}$$

- Constrain inter-order; the rows of one order should not overlap those of the following order.
$$\theta \in \{1, \ldots, \theta_{max} - 1\} :$$
$$(\dot{\rho}_{m,b,\theta} + \rho_{m,b,\theta}) \leq \dot{\rho}_{m,b,\theta+1} + (2 - \hat{\rho}_{m,b,\theta} - \hat{\rho}_{m,b,\theta+1}) \cdot (2d_m)$$

- Constrain grouping of empty tables. If there are θ_0 orders of non-empty tables, then all higher orders are all empty tables; that is, the number of rows $\rho_{m,b,\theta} = 0$ for $\theta \in \{\theta_0 + 1, \ldots, \theta_{max}\}$.
  Count:

$$\theta \in \{1, \ldots, \theta_{max} - 1\} :$$

$$\hat{\rho}_{m,b,\theta} \geq \hat{\rho}_{m,b,\theta+1}$$

- Constrain the maximum tables per block to correspond with the maximum order, θ_{max}.

$$\sum_l \hat{r}_{l,m,b} \leq \theta_{max}$$

Product and Binary constraints

It takes a lot to get product and binary constraints working. They are all listed in this section. Note $\epsilon = 1e - 4$.

- Aggregation block binary per memory type. Indicator if any blocks of type m in stage s are assigned to any table. b_m is the number of blocks of type m in stage s.
  Count: $2m_maxM = O(m_maxM)$

$$\hat{\widetilde{B}}_{s,m} \cdot \epsilon \leq \widetilde{B}_{s,m} \text{ (LB)}, \qquad \hat{\widetilde{B}}_{s,m} \cdot b_m \geq \widetilde{B}_{s,m} \text{ (UB)}$$

- Aggregation block binary per logical table. Indicator if any blocks of any type m in stage s are assigned to table l
  Count: $2NM = O(NM)$

$$\hat{\widetilde{U}}_{s,l} \leq \sum_m U_{s,l,m} \text{ (LB)}, \qquad (\sum_m b_m)\hat{\widetilde{U}}_{s,l} \geq \sum_m U_{s,l,m} \text{ (UB)}$$

- Number of rows binary of words of table l that start in block b.

$$\hat{R}_{l,m,b} \cdot \epsilon \leq R_{l,m,b} \text{ (LB)}, \qquad \hat{R}_{l,m,b} \cdot d_m \geq R_{l,m,b} \text{ (UB)}$$

- Number of rows binary of any part of words of table l in block b.

$$\hat{r}_{l,m,b} \cdot \epsilon \leq r_{l,m,b} \text{ (LB)}, \qquad \hat{r}_{l,m,b} \cdot d_m \geq r_{l,m,b} \text{ (UB)}$$

- Product of start row and number of rows, both of which correspond to words of table l that start in block b.

$$P^{\dot{R},\hat{R}}_{l,m,b} \leq \dot{R}_{l,m,b}, \qquad P^{\dot{R},\hat{R}}_{l,m,b} \leq d_m \cdot \hat{R}_{l,m,b}$$

$$P^{\dot{R},\hat{R}}_{l,m,b} \geq 0,, \qquad P^{\dot{R},\hat{R}}_{l,m,b} \geq \dot{R}_{l,m,b} - d_m(1 - \hat{R}_{l,m,b})$$

- Product of number of rows of any part of a word from table l to the mapper variable order θ to table l
$$P^{r,L}_{\theta,l,m,b} \leq r_{l,m,b}, \qquad P^{r,L}_{\theta,l,m,b} \leq d_m \cdot L_{\theta,l,m,b}$$
$$P^{r,L}_{\theta,l,m,b} \geq 0, , \qquad P^{r,L}_{\theta,l,m,b} \geq r_{l,m,b} - d_m(1 - L_{\theta,l,m,b})$$
- Product of start row of any part of a word from table l to the mapper variable order θ to table l.
$$P^{\dot{r},L}_{\theta,l,m,b} \leq \dot{r}_{l,m,b}, \qquad P^{\dot{r},L}_{\theta,l,m,b} \leq d_m \cdot L_{\theta,l,m,b}$$
$$P^{\dot{r},L}_{\theta,l,m,b} \geq 0, , \qquad P^{\dot{r},L}_{\theta,l,m,b} \geq \dot{r}_{l,m,b} - d_m(1 - L_{\theta,l,m,b})$$
- Number of rows binary of any parts of the θ-th table in block b.
$$\hat{\rho}_{m,b,\theta} \cdot \epsilon \leq \rho_{l,m,b} \text{ (LB)}, \qquad \hat{\rho}_{l,m,b} \cdot d_m \geq \rho_{l,m,b} \text{ (UB)}$$
- Start and end stage binary per logical table (for dependency constraints).
Count: $6NM = O(NM)$
$$P^{\hat{S},\hat{\hat{U}}}_{s,l} \leq \hat{S}_{s,l} \qquad P^{\hat{E},\hat{\hat{U}}}_{s,l} \leq \hat{E}_{s,l}$$
$$P^{\hat{S},\hat{\hat{U}}}_{s,l} \leq \hat{\hat{U}}_{s,l} \qquad P^{\hat{E},\hat{\hat{U}}}_{s,l} \leq \hat{\hat{U}}_{s,l}$$
$$P^{\hat{S},\hat{\hat{U}}}_{s,l} \geq \hat{S}_{s,l} + \hat{\hat{U}}_{s,l} - 1 \qquad P^{\hat{E},\hat{\hat{U}}}_{s,l} \geq \hat{E}_{s,l} + \hat{\hat{U}}_{s,l} - 1$$

Common constraints

These are the constraints in Section 4.1 of the attached paper, which are the constraints required of any program assignment to any match-action switch.

- Capacity constraint.
  - Capacity constraint by stage. If wide tables assigned to block b would overlap past the last block of the stage b_{max}, the tables cannot be assigned to block b in the first place. Recall the preprocessor function $f_U(l,m)$ returns the number of blocks of memory type m that entries from table l would occupy, and $\max f_b(s)$ returns the last block index in this stage.
  Count:
  $$\text{If } b + f_U(l,m) \geq \max f_b(s) :$$
  $$R_{l,m,b} == 0$$
  - Capacity constraint by row. The row assignment to each order in a block must not exceed the row dimension of the block; this means that the index of the last row of each assignment should not be larger than the last possible row index d_m.
  $$\dot{\rho}_{m,b,]theta} + \rho_{m,b,\theta} \leq d_m + (1 - \hat{\rho}_{m,b,\theta}) \cdot (d_m)$$

- Assignment constraint. e_l is the number of entries of table l.
  Count: $N = O(N)$
  $$\sum_m \sum_s W_{s,l,m} \geq e_l.$$
- Dependency constraint. This involves assigning all of the dependency variables as well, where ϵ is some small non-zero margin.
  Count: $2N + 2NM + 2N + d_{tot} = O(NM + d_{tot})$, where d_{tot} is the total number of match, action, successor, and reverse match dependencies.
  - Unique start and end stage constraints. There is a unique start and end stage for each logical table l. These constraints allow us to get the start and end stage for table l, $\sum_s(\hat{S}_{s,l} \cdot s)$ and $\sum_s(\hat{E}_{s,l} \cdot s)$, respectively.
    $$\sum_s \hat{S}_{s,l} == 1$$
    $$\sum_s \hat{E}_{s,l} == 1$$
  - Start and end stage constraint. If a stage has blocks, the start stage is at least as small, and the end stage is at least as big. M is an upper bound.
    $$\sum_{s'} (\hat{S}_{s',l} \cdot s') \leq s + (1 - \hat{\tilde{U}}_{s,l})M$$
    $$\sum_{s'} (\hat{E}_{s',l} \cdot s') \geq s + (1 - \hat{\tilde{U}}_{s,l})M$$
  - Non-zero start and stages. Both the start and end stages must have blocks.
    $$\sum_s P_{s,l}^{\hat{S},\hat{\tilde{U}}} \geq 1$$
    $$\sum_s P_{s,l}^{\hat{E},\hat{\tilde{U}}} \geq 1$$
  - Match dependencies. Suppose table l_2 has a match dependency on l_1, so l_2 must execute after l_1.
    $$\sum_s (\hat{S}_{s,l_2} \cdot s) \geq \sum_s (\hat{E}_{s,l_1} \cdot s) + \epsilon$$
  - All other dependencies. Suppose table l_2 has a successor or reverse dependency on l_1.
    $$\sum_s (\hat{S}_{s,l_2} \cdot s) \geq \sum_s (\hat{E}_{s,l_1} \cdot s)$$

Objectives

We do not have an objective function to optimize over for this problem; we are just focused on any feasible solution. So we can write the objective function as:

$$\min 0$$