Minimizing Delay in Shared Pipelines

Ori Rottenstreich (Technion, Israel)

Joint work with
Isaac Keslassy (Technion, Israel)
Yoram Revah, Aviran Kadosh (Marvell Israel)
Pipeline Sharing for H.264 video-compression standard

9 profiles with 21 features are organized in 9 pipelines in a 9x9 multicore scalability problem:
more profiles, more features
more cores
Pipeline Sharing for H.264 video-compression standard

9 profiles with 21 features are organized in 9 pipelines in a 9x9 multicore.

**Pipeline Sharing:** The 9 profiles are serviced by one of two possible pipelines in a smaller 5x5 multicore.
Another Pipeline Sharing Example

- Regular architecture, without pipeline sharing:
  - 3 pipelines for the $k=3$ (uniformly distributed) packet types with $N=8$ cores
  - Average delay of $T=(3+2+3)/3 \approx 2.67$ time slots

- With pipeline sharing:
  - 2 pipelines for the $k=3$ packet types with only $N=6$ cores
  - Average delay of $T=(4+2+4)/3 \approx 3.33$ time slots
  - Tradeoff: Less cores, larger average delay
Model and Problem Definition

- **Traffic:**
  - There are \(k\) packet types with known probabilities, each requires to perform tasks among \(\{1, \ldots, r\}\) possible tasks.
  - Example: Type 1 w.p. \(p_1=0.6\) requires tasks \(S_1=\{1,2,3\}\) and Type 2 w.p. \(p_2=0.4\) requires tasks \(S_2=\{1,4\}\).

- **Pipeline sharing:**
  - A limited number of \(N\) homogeneous cores is given. Each core can serve any task.
  - The cores should be divided into pipelines, each serves one or more packet types.
  - The delay of a packet equals the length of its pipeline.

- **Optimization Problem:**
  - Divide the \(N\) cores into pipelines, such that the average delay is minimized.
  - For a given \(N\), we denote by \(T_{OPT}(N)\) the minimal possible average delay.
Outline

- Introduction and Problem Definition
- General Properties
  - Optimal Algorithm for a Special Case
  - Greedy Algorithm
  - Experimental Results
- Summary
**General Properties**

- **Property:**
  
  (i) At least $N = |\bigcup_{i=1}^{k} S_i|$ cores are required
  
  (ii) For all $N \geq 0$, the optimal average delay $T_{OPT}(N)$ satisfies
  \[
  T_{OPT}(N) \geq \sum_{i=1}^{k} \left( p_i \cdot |S_i| \right)
  \]
  
  (iii) $T_{OPT}(N) = \sum_{i=1}^{k} \left( p_i \cdot |S_i| \right)$ for $N \geq \sum_{i=1}^{k} |S_i|$
• **Property:** Let \( k \) be the number of packet types. Then,
  
  (i) Given an unlimited number of cores, the number of solutions with \( d \) pipelines \( Q_1, \cdots, Q_d \) is given by \( S(k, d) \), where

  \[
  S(k, d) = \frac{1}{d!} \cdot \sum_{j=1}^{d} \left( (-1)^{d-j} \cdot \binom{d}{j} \cdot j^k \right),
  \]

  is the **Stirling number of the second kind** of \( k, d \)

  (ii) The total number of solutions is \( G(k) = \sum_{d=1}^{k} S(k, d) \)

• **Example:** Consider \( k=3 \) packet types. There is a single \( S(3, 1) = 1 \) solution with 1 pipeline:

  There are \( S(3, 2) = 3 \) solutions with 2 pipelines:

  There is \( S(3, 3) = 1 \) solution with 3 pipelines:

  The total number of solutions is \( G(3) = 1 + 3 + 1 = 5 \)
Separability

- **Property:** Assume that packet types $S_1, \cdots, S_k$ can be partitioned into two disjoint sets, i.e. they can be ordered s.t.

  \[(\exists m \in [1, k]) \left( \bigcup_{i=1}^{m} S_i \right) \cap \left( \bigcup_{i=(m+1)}^{k} S_i \right) = \emptyset.\]

  Then, $(\exists N_0 \in [0, N])$ s.t. an optimal solution given $N$ cores can be obtained as the union of the two sets of pipelines in the optimal solutions for packet types $[1, m]$ with $N_0$ cores and for packet types $[(m+1), k]$ with $(N-N_0)$ cores.

- **Proof Outline:** Any pipeline in an optimal solution cannot serve tasks from both sets $\left( \bigcup_{i=1}^{m} S_i \right)$, $\left( \bigcup_{i=(m+1)}^{k} S_i \right)$. Otherwise, it could be partitioned into two smaller pipelines to reduce the average delay.
Outline

- Introduction and Problem Definition
- General Properties
- Optimal Algorithm for a Special Case
  - Greedy Algorithm
  - Experimental Results
- Summary
Simple Case of the Required Tasks: \( S_i = [1, X_i] \)

- We suggest an **optimal algorithm** for a special case of the sets of required tasks: 
  \( S_i = [1, X_i] \) for \( X_i \in [1, r] \)

- **Example:** \( k=3, ([1, p_1), ([1, 3], p_2), ([1, 5], p_3) \)

- The condition is equivalent to the requirement \( S_1 \subseteq S_2 \subseteq \cdots \subseteq S_k \)
- Assume that \( S_1 \cdots S_k \) are ordered s.t. \( X_i \leq X_j \) for \( i < j \)

- Let \( Q_1, \cdots, Q_d \) be the sets of tasks served by the pipelines in an optimal solution

- **Proposition:** The pipelines in an optimal solution \( Q_1, \cdots, Q_d \) satisfy 
  \( (\forall j \in [1, d]) \ Q_j \in \{ S_1, \cdots, S_k \} \), i.e. the pipelines in the solution are among the pipelines in the input
• **Proposition:** Assume that $Q_1, \cdots, Q_d$ are ordered such that $Q_1 \subseteq Q_2 \subseteq \cdots \subseteq Q_{d-1} \subseteq Q_d$. Then,

(i) The packet types are served by an increasing order of pipelines. In particular, the packet types served by each pipeline form a subset of consecutive packet types from the input.

(ii) The last packet type is served by the last pipeline $Q_d$.

\[ S_1, S_2, S_3 \quad S_4, S_5 \quad S_6, \ldots \quad \ldots \quad \ldots, S_k \]

\[ \downarrow \quad \downarrow \quad \downarrow \quad \quad \downarrow \quad \quad \downarrow \quad \downarrow \]

\[ Q_1 \quad Q_2 \quad Q_3 \quad \ldots \quad \ldots \quad Q_d \]
Simple Case of the Required Tasks: $S_i = [1, X_i]$

- Let $T^i(n)$ (for $i \in [0, k]$, $n \in [0, N]$) be the minimal possible average delay obtained in the service of the first $i$ types with at most $n$ cores. Let $Q^i(n)$ be the corresponding set of pipelines.

- **Proposition:**
  (i) For $i \geq 1$, $T^i(n) = \min_{j \in [1, i]} \left( T^{i-j}(n - |S_i|) + \left( \sum_{m=(i-(j-1))}^{i} p_m \right) \cdot |S_i| \right)$
  (ii) For $j$ that minimizes (i), $Q^i(n) = Q^{i-j}(n - |S_i|) \cup \{S_i\}$

- **Algorithm:**
  - In step $i$ (for $i \in [1, k]$), calculate $T^i(n)$ for $n \in [0, N]$
  - Return: $T^k(N)$ (optimal delay), $Q^k(N)$ (set of pipelines)
  - Complexity: $O(k \cdot N \cdot k) = O(k^2 \cdot N)$
Outline

- Introduction and Problem Definition
- General Properties
- Optimal Algorithm for a Special Case
- Greedy Algorithm
- Experimental Results
- Summary
• **Idea:** Start with a pipeline for each type, merge pairs of pipelines with common cores

![Pipeline Diagram](image)

• **Intuition:** For each pair with common cores, we prefer to merge
  - More common cores
  - Less non-common cores
  - Low probability to be served by a pipeline in the pair

• **Marginal cost** of a possible merging operation
  - $x$ – expected increment in the average delay
  - $y$ – decreased number of cores
  - Marginal cost of $R = \frac{x}{y}$

• **Algorithm:** Until the required number of cores is obtained, merge pairs of pipelines with minimal marginal cost $R = \frac{x}{y}$

• Not necessarily optimal but very efficient on synthetic and real-life applications
Experimental Results

- **Synthetic Simulation Parameters:**
  - We consider $k=8$ packet types
  - Tasks selected among $r=10$ possible tasks $\{1,\ldots,r=10\}$
  - Each type requires a specific task w.p. 0.5 without any dependency between different types or tasks
  - Two options for the packet types probabilities:
    - **fixed prob.** – all types w.p. $1/k = 0.125$
    - **variable prob.** – geometrically decreasing probabilities of $2^{-1}$, $2^{-2}$, $2^{-3}$, $2^{-4}$, $2^{-5}$, $2^{-6}$, $2^{-7}$ and $2^{-7}$
  - Results are based on the average of $10^3$ iterations
Effectiveness of the number of cores on the average delay in time slots. $k=8$ packet types with $r=10$ possible tasks, each required w.p. 0.5 by each type. $10^3$ iterations.

Experimental Results

Less cores $\Rightarrow$ larger delay

A lower bound: Average number of tasks per type $10 \cdot 0.5 = 5$
Summary of the synthetic simulations. $k=8$ packet types with $r=10$ possible tasks, each required w.p. 0.5 by each type. $10^3$ iterations.

Experimental Results

<table>
<thead>
<tr>
<th></th>
<th>Fixed Priority</th>
<th>Greedy</th>
<th>Optimality</th>
<th>Optimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>(i) Average delay for $N \in [10,60]$</td>
<td>6.20</td>
<td>6.00</td>
<td>5.80</td>
<td>5.73</td>
</tr>
<tr>
<td>(ii) Delay for $N = 16$</td>
<td>8.47</td>
<td>8.11</td>
<td>7.35</td>
<td>7.12</td>
</tr>
</tbody>
</table>

A difference of less than 2%

16 cores instead of 60, delay of 8.47 / 8.11 time slots instead of ~5

- Summary of the synthetic simulations. $k=8$ packet types with $r=10$ possible tasks, each required w.p. 0.5 by each type. $10^3$ iterations.
### Packet-processing Application

<table>
<thead>
<tr>
<th></th>
<th>packet type</th>
<th>tasks</th>
<th>Prob.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>L2 unicast packet</td>
<td>$S_1 = {1, 2, 4, 10}$</td>
<td>0.25</td>
</tr>
<tr>
<td>2</td>
<td>L2 unicast packet with security control</td>
<td>$S_2 = {1, 2, 3, 4, 10, 11}$</td>
<td>0.15</td>
</tr>
<tr>
<td>3</td>
<td>L3 multicast packet</td>
<td>$S_3 = {1, 2, 5, 6, 8, 9, 10}$</td>
<td>0.2</td>
</tr>
<tr>
<td>4</td>
<td>MPLS packet</td>
<td>$S_4 = {1, 2, 3, 7, 8, 10, 11}$</td>
<td>0.3</td>
</tr>
<tr>
<td>5</td>
<td>ACL Packet</td>
<td>$S_5 = {1, 2, 3, 10}$</td>
<td>0.1</td>
</tr>
</tbody>
</table>

- **Possible Tasks:** (1) Parsing, (2) Ingress interface attributes, (3) Ingress ACL, (4) L2 bridging, (5) L3 routing, (6) L3 replication, (7) MPLS switching, (8) header modification, (9) L2 replication, (10) Egress interface resolution, (11) Egress ACL
• $k=5$ packet types w.p. (0.25, 0.15, 0.2, 0.3, 0.1) and $r=11$ possible tasks. The total number of solutions is $G(5) = 52$. The greedy algorithm starts with 28 cores, then reduces it to 24, 20, 15 and finally 11. It obtains the optimal delay in 16 out of 18 values of $N$. 

The greedy algorithm obtains the optimal delay in 16 out of 18 values of $N$. 

The figure shows the delay for different numbers of cores. The black bars represent the optimal solution, and the gray bars represent the greedy algorithm. The y-axis represents the delay, and the x-axis represents the number of cores. The total number of tasks is 20.
H.264 video-compression standard

9 profiles with 21 features are organized in 9 pipelines in a 9x9 multicore.

*Pipeline Sharing:* The 9 profiles are serviced by one of two possible pipelines in a smaller 5x5 multicore.
• $k=9$ popular (uniformly-distributed) profiles (types) of the H.264 standard with $r=21$ supported features (tasks). Results are based on the greedy algorithm.
Concluding Remarks

• New approach of sharing pipelines to reduce the number of required cores

• Analysis of the optimal average delay

• Optimal solution for \( S_i = [1, X_i] \)

• Greedy algorithm for the general case
Thank You