Data-Driven Scenario-Based Application Mapping For Heterogeneous Many-Core Systems

Jan Spieck, Stefan Wildermann, Tobias Schwarzer, Jürgen Teich, Michael Glaß*

Hardware-Software-Co-Design
Friedrich-Alexander-Universität Erlangen-Nürnberg

*Universität Ulm
Motivation: Input-Dependent Workload

Task-based applications with input-dependent workload distribution, e.g.:

**Stitching**
- $\text{Src}_0$ → $\text{SIFT}_0$ → $\text{Matching}_0$ → $\text{Ransac}_0$ → $\text{Sink}$
- $\text{Src}_1$ → $\text{SIFT}_1$ → $\text{Matching}_1$ → $\text{Ransac}_1$
- $\text{Src}_2$ → $\text{SIFT}_2$

**Ray tracing**
- $\text{Src}$ → $\text{Cell}_0$ → $\text{Sink}$ → $\text{Cell}_8$
Motivation: Data-Driven Mapping I

Task: Mapping application tasks onto a heterogeneous architecture

Problem: Single mappings do not exploit specializations of input data

Example: Processing subsequent data $d_0 = \{100, 200\}$ and $d_1 = \{200, 100\}$ triangles

<table>
<thead>
<tr>
<th>$m_0$</th>
<th>$d$</th>
<th>$C_0$</th>
<th>$C_1$</th>
<th>$m_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$d_0$</td>
<td>50 ms</td>
<td>100 ms</td>
<td>$d_0$</td>
</tr>
<tr>
<td></td>
<td>$d_1$</td>
<td>100 ms</td>
<td>50 ms</td>
<td>$d_1$</td>
</tr>
<tr>
<td>$r_0$</td>
<td>$d$</td>
<td></td>
<td></td>
<td>$r_0$</td>
</tr>
<tr>
<td></td>
<td>$d_0$</td>
<td>150 ms</td>
<td></td>
<td>$d_0$</td>
</tr>
<tr>
<td></td>
<td>$d_1$</td>
<td>150 ms</td>
<td></td>
<td>$d_1$</td>
</tr>
<tr>
<td>$r_1$</td>
<td>$\Sigma$</td>
<td>300 ms</td>
<td></td>
<td>$r_1$</td>
</tr>
</tbody>
</table>
Motivation: Data-Driven Mapping II

Solution: Partitioning the data space $D = \bigcup_k D_k$ into scenarios $D_k$

But how do we determine these scenarios and scenario-optimized mappings?

<table>
<thead>
<tr>
<th>$d$</th>
<th>$C_0$</th>
<th>$C_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d_0$</td>
<td>100 ms</td>
<td>100 ms</td>
</tr>
<tr>
<td>$d_1$</td>
<td>100 ms</td>
<td>100 ms</td>
</tr>
</tbody>
</table>

$D_0 = \{d_1\} \rightarrow m_0$

$D_1 = \{d_0\} \rightarrow m_1$

End to end

\[ \begin{align*}
\sum: & 200 \text{ ms} \\
100 \text{ ms} & \\
100 \text{ ms} & \\
\end{align*} \]
Problem 1: find scenario-optimized mappings \( m \in M \)

Given: scenario distribution \( S = (D_1, \ldots, D_n) \)

\[
\begin{array}{cccccc}
  & p(D_1) & p(D_2) & p(D_3) & \#(R_1) & \#(R_2) \\
m_1 & 10 & 2 & 2 & 4 & 1 \\
m_2 & 8 & 6 & 6 & 3 & 0 \\
m_3 & 4 & 10 & 5 & 4 & 2 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\end{array}
\]

Pareto-optimal mappings

\( M = \{m_1, m_2, m_3, \ldots\} \)
Problem 2: Given a set of optimized mappings $M = \{m_1, \ldots\}$ find a scenario distribution $S \in S^\circ$

Scenarios

$S = \{D_1, D_2, D_3\}$

Scenario $D_i \subseteq D$: All data which performs equally/similarly for different mappings $m_j$
Circular Dependency

Given: scenario distribution $S$
Find: optimized mappings $M$

Given: optimized mappings $M$
Find: optimal scenario distribution $S$
Design-Time Optimization

Solution: Iterative scenario-based design space exploration

Steps:
1. Input Generation
2. Scenario Initialization
Loop:
3. Design space exploration
4. Distillation
5. Scenario Identification
6. Termination

Jan Spieck et al., FAU, HSCD | MCSoC '19
Iterative Optimization Loop I

Input Generation: select representative subset of data
- $D_{train} \subset D$, $D_{test} \subset D$
- $D_{train} \cap D_{test} = \emptyset$

Scenario Initialization:
- Random scenario distribution
- Clustering on default mappings

Jan Spieck et al., FAU, HSCD | MCSoC '19
Iterative Optimization Loop II

Design space exploration (DSE):

\[
\begin{pmatrix}
  p(D_1, m) \\
  \vdots \\
  p(D_n, m) \\
  |R_1(m)| \\
  \vdots \\
  |R_u(m)|
\end{pmatrix}
\]

- minimize

- Using evolutionary algorithms
Iterative Optimization Loop III

Distillation: Reduce resulting set $M'$ to a (smaller) set $M \subseteq M'$

- Improves identification step
- Option 1: clustering over mappings and sampling
- Option 2: based on a weighted sum over $p(D_k, m)$

Example: $p(D_k, m) = (\text{latency, energy})$

$w_p = \text{latency} + 0.5 \cdot \text{energy}$  \quad |M| = 2

<table>
<thead>
<tr>
<th>$M'$</th>
<th>latency</th>
<th>energy</th>
<th>$w_p$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m_1$</td>
<td>10 ms</td>
<td>20 mJ</td>
<td>20</td>
</tr>
<tr>
<td>$m_2$</td>
<td>20 ms</td>
<td>10 mJ</td>
<td>25</td>
</tr>
<tr>
<td>$m_3$</td>
<td>30 ms</td>
<td>5 mJ</td>
<td>32.5</td>
</tr>
</tbody>
</table>
Scenario Identification: $S = \arg \min_{(D_1, ..., D_n) \in S^o} \sum_{k=1}^n dist(D_k, M)$

Option 1: Clustering (e.g., K-Means)
- Performance per mapping $m_i \in M$
- $v(d) = [p(d, m_1) ... p(d, m_l)]^T$
- $dist(D_k, M) = \sum_{d \in D_k} ||v(d) - \mu_k||^2$

Option 2: Performance Optimization
- $dist(D_k, M) = \min_{m_k \in M_i} \{\sum_{d \in D_k} p(d, m_k)\}$
- Best suited for low-dimensional $p(d, m)$
Evaluation Setup

- Applications (Data):
  - Ray tracing (virtual 3D-scenes)
  - Stitching (partial images of panoramas)
- Architecture: heterogeneous 3x3 NoC mesh
- Data: split into training and test set
- Each test data is executed in the best-suited scenario $D_i \in S$
- Goal: minimal latency for processing the total scenario distribution
Latency for different optimization approaches (test set with bigger scenes)

- **Training Set**
  - Single Optimized Mapping: 44.52 s
  - State-Of-The-Art Sequential: 39.08 s
  - Proposed: 34.82 s
  - Improvement: 12.2% for State-Of-The-Art Sequential, 21.8% for Proposed

- **Test Set**
  - Single Optimized Mapping: 61.71 s
  - State-Of-The-Art Sequential: 56.04 s
  - Proposed: 52.17 s
  - Improvement: 9.2% for State-Of-The-Art Sequential, 15.5% for Proposed

Jan Spieck et al., FAU, HSCD | MCSoC '19
Eval.: Stitching Latency

Latency for different optimization approaches (test set with bigger images)

![Bar chart showing latency comparisons for training and test sets. The x-axis represents latency in seconds, and the y-axis shows different optimization methods. The training set latencies are: Single Optimized Mapping at 3.22, State-of-the-Art Sequential at 3.12, and Proposed at 2.89 seconds. The test set latencies are: Single Optimized Mapping at 11.31, State-of-the-Art Sequential at 11.11, and Proposed at 10.27 seconds. The differences in percentages are: 3.1% for training and 1.8% for testing.]
Latencies for different resource availability (considered during DSE by $R_i(m)$):

Ray tracing

Stitching
Conclusion

• Mapping optimization for applications with input-dependent task workload onto heterogeneous architectures

• Scenario-based design space exploration
  1. Input Generation
  2. Scenario Initialization
  3. Design space exploration
  4. Distillation
  5. Scenario Identification
  6. Termination

• Significant speedup compared to a single optimized mapping for the average-case (15% ray tracing, 10% stitching (test set))
Thanks for listening!

This work was supported by the German Research Foundation (DFG) as part of the Transregional Collaborative Research Centre „Invasive Computing“ (SFB/TR 89)

Are there any questions?
Run-Time Manager

At run time: Optimize latency of data mappings
Given: Sequence of data with unknown scenario affiliation

Data Sequence
\[ d_0, d_1, d_2, d_3, d_4, d_5, ... \]

Execution Properties \( e(d_i, c_i) \)

Mapping \( c_{i+1} \in M \)

Application

Run-Time Manager

Design-Time Knowledge

\[ Scenarios \ S = \{s_0, s_1, s_2, s_{avg}\}, Mappings \ M = \{m(s) \mid s \in S\} \]