Design-space Optimization of Streaming Applications

Shobana Padmanabhan
Professors Yixin Chen, Roger D. Chamberlain
Washington University

Funded under NSF grants CNS-0905368, CNS-0931693

http://arl.wustl.edu/~sp3/

SAAHPC 2010
July 13, 2010
Problem overview

How to find efficiently knob settings (configuration) that satisfy goals and constraints optimally?

Conflicting goals

Knobs (parameters) may not be independent
Our approach

• Formulate as optimization problem

• Domain-specific branch & bound
  – Exploit structure of application’s pipelining

• Queueing network performance models
Design-space optimization

1. Problem formulation

2. Domain-specific branch & bound
   1. Ordering of branching variables
   2. Number of branches evaluated

3. Empirical results
Design-space optimization

1. Problem formulation

2. Domain-specific branch & bound
   1. Ordering of branching variables
   2. Number of branches evaluated

3. Empirical results
Sort application, knobs

High throughput, low latency

• Number of sort “blocks”
• Mapping choices
  – Number of resources
  – Type of resources
• Algorithmic choices
• Communication architecture
• Message size
• Input job arrival rate
Design-space exploration is hard

- Knobs may not be independent

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Knob 1</th>
<th>...</th>
<th>Knob ( n )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>...</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>...</td>
<td>1</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( 10^{21} ) possible configurations</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Design-space exploration is hard

- Knobs may not be independent
- Goals tend to conflict

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Knob 1</th>
<th>...</th>
<th>Knob_n</th>
<th>Throughput</th>
<th>Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>...</td>
<td>0</td>
<td>1.1 MB/sec</td>
<td>.5 sec</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>...</td>
<td>1</td>
<td>1.2 MB/sec</td>
<td>.9 sec</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>10^{21} possible configurations</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Search takes months, even a few Masters degrees!

*We want to make the search automatic, efficient*
Queueing network model

- Divide
- Sort
- Combine

Queueing station (QS)

\( \lambda_{in} \), \( \lambda_0 \), \( \mu_0 \), \( \lambda_1 \), \( \mu_1 \), \( \lambda_2 \), \( \mu_2 \), \( \lambda_3 \), \( \mu_3 \), \( \lambda_4 \), \( \mu_4 \)

\( \lambda \): mean (job) arrival rate
\( \mu \): mean service rate
Mean service, job arrival rates

- $\lambda$: mean (job) arrival rate
- $\mu$: mean service rate

$\mu_0 = (C_1 \cdot cpu_0 + C_2 \cdot fpga_0) \cdot numRes_0 \cdot \frac{1}{2^{NUM\_ELEM}}$

- $\mu$ is a function of knobs
- $\lambda$ related by system of linear equations
  
  Each $\lambda = \lambda_{\text{in}}$ (input job arrival rate)

- Each $\lambda <$ corresponding $\mu$
Cost functions

- Throughput = Input job arrival rate ($\lambda_{in}$)
  - Only throughput: bottleneck in pipeline = min $\mu$

- Latency = $\sum_{j=0}^{4} \frac{1}{\mu_j - \lambda_j}$

Assuming classic M/M/1 queueing network
Knobs may change queueing network

Queueing network₁

Queueing network₂

Divide → Sort → Combine

Divide → Sort → Combine

Divide → Sort → Combine

Divide → Sort → Combine

Divide → Links → Div

Divide → Links → Sorts

Divide → Links → Sorts

Divide → Links → Sorts
Knobs may change queueing network

Queueing network_1

Queueing network_2

Divide
Sort
Combine

Div
Links .5
Sorts
Comb

Shobana Padmanabhan
Design-space optimization
Complication from multiple queueing networks

Queueing network$_1$

Queueing network$_2$

Queueing network$_3$

\[ \mu_0 = (C_1 \cdot cpu_0 + C_2 \cdot fpga_0) \cdot nRes_0 \cdot \frac{1}{2^{NUM\_ELEM}} \]

\[ \mu_0 = QN_1 \cdot (\text{same as before}) + QN_2 \cdot (\cdots) + \cdots \]

Binary variables
Provided by application developers

Queueing network\textsubscript{1}

Queueing network\textsubscript{2}

Queueing network\textsubscript{3}

Account for all queueing networks

- Knobs, ranges
- Expressions for mean service rates
- Expressions for mean job arrival rates
- Other requirements
- Cost functions
Provided by application developers

Account for all queueing networks
- Knobs, ranges
- Expressions for mean service rates
- Expressions for mean job arrival rates
- Other requirements
- Cost functions

Queueing network_1
Queueing network_2
Queueing network_3
We formulate optimization problem

Account for all queueing networks

- Knobs, ranges
- Expressions for mean service rates
- Expressions for mean job arrival rates
- Other requirements
- Cost functions

Minimize
\[ z = W_1 \cdot \left( \frac{1}{\text{Throughput}} \right) + W_2 \cdot (\text{Latency}) \]

Mixed-Integer Nonlinear problem (MINLP)

These nonlinear functions are not convex or continuous

*State-of-the-art solvers fail*
1. Problem formulation

2. Domain-specific branch & bound
   1. Ordering of branching variables
   2. Number of branches evaluated

3. Empirical results
Standard branch & bound (B&B)

Efficiency depends on

1. Ordering of branching variables
2. Number of branches evaluated

We exploit structure of application’s pipelining
Identify decomposability

Jordan block form

We identify a queueing station as a block
Complicating variables prevent decomposability

Jordan block form

We identify a queueing station as a block
Complicating, non-complicating categories

Knobs

topology  Multi QS  Single QS

Knobs = topology U M-QS U S-QS
Branch on the most complicating variables first

Knobs = topology U M-QS U S-QS

1. Topological variables
2. Multi queueing station variables
3. Single queueing station variables
Design-space optimization

1. Problem formulation

2. Domain-specific branch & bound
   1. Ordering of branching variables
   2. Number of branches evaluated

3. Empirical results
Reduce number of branches

Not able to “bound” → can not prune branches

We discover convexity property:

*If objective function of a BV is convex, solve for the variable analytically*

Minimize  \[ z = W_1 \cdot \left( \frac{1}{\text{Throughput}} \right) + W_2 \cdot (\text{Latency}) \]

\[ \lambda_{\text{in}} \]

\[ \sum_{j=0}^{4N} \frac{1}{\mu_j - \lambda_j} \]
Reduce number of branches

Not able to “bound” → can not prune branches

We discover convexity property:

*If objective function of a BV is convex, solve for the variable analytically*

\[
\min z = W_1 \cdot \left( \frac{1}{\text{Throughput}} \right) + W_2 \cdot (\text{Latency})
\]

\[z(\lambda_{in}) \text{ is convex, } \lambda_{in} \in \mathbb{R}\]

We can solve for \( \lambda_{in} \) analytically 😊
1. **Problem formulation**

2. **Domain-specific branch & bound**
   1. Ordering of branching variables
   2. Number of branches evaluated

3. **Empirical results**
Empirical results (objective function value in ms)

Original problem: solver fails (SF)

Num sort blocks ($2^N$):

- 2
- 8
- 1024

Mapping: $m_0$, $m_3$

$\lambda_{in}$

Objective function values:

<table>
<thead>
<tr>
<th>Num sort blocks</th>
<th>305.28</th>
<th>$\cdots$</th>
<th>SF</th>
<th>$\cdots$</th>
<th>SF</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m_0$</td>
<td>305.39</td>
<td>$\cdots$</td>
<td>299.16</td>
<td>$\cdots$</td>
<td>299.14</td>
</tr>
<tr>
<td>$m_3$</td>
<td>305.13</td>
<td>SF</td>
<td>SF</td>
<td>SF</td>
<td>SF</td>
</tr>
</tbody>
</table>

Shobana Padmanabhan

Design-space optimization
Number of branches evaluated reduces

Original problem: solver fails (SF)

Num sort blocks ($2^N$):

Mapping: $m_0$, $m_3$

$\lambda_{in}$

B&B: $|\text{map}| \times |\lambda_{in}|$: $4 \times 20 = 80$ branches

Ours: $|\text{map}|$: $= 4$ branches 😊
Continue branching for global optimum

Original problem: solver fails (SF)

Num sort blocks ($2^N$):

- 2
- 8
- 1024

Mapping: $m_0$, $m_3$

- $\lambda_{in}$
  - 305.39
  - 305.13
  - 299.16
  - 299.14
  - 299.21

- SF

Shobana Padmanabhan  Design-space optimization
Number of branches evaluated reduces further

Original problem: solver fails (SF)

Num sort blocks \((2^N)\):

- 2
- 8
- 1024

Mapping:

- \(m_0\)
- \(m_3\)

\(\lambda_{in}\)

Mapping:

- \(m_0\)
- \(m_3\)

Mapping:

- \(m_0\)
- \(m_3\)

Mapping:

- \(m_0\)
- \(m_3\)

versus

Our B&B 😊

Standard B&B
Number of branches evaluated reduces further

Original problem: solver fails (SF)

Num sort blocks ($2^N$):

- 2
- 8
- 1024

Mapping: $m_0$, $m_3$

Mapping:

Original problem: solver fails (SF)

Mapping: $m_0$, $m_3$

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping:

Mapping: 512 branches

Mapping: 20 branches 😊
Conclusion

• **We exploit structural info to help solvers**
  – Solve problem not solved by state-of-the-art solvers
  – For a different problem, improve cost function value from 10s to 2ms

• **Easy adaptation for changing requirements**
  – Just change weights in objective function

• **Ongoing research**
  – Latency, throughput benefit from higher service rate but power does not – how to handle it?
  – Apply to “filtering” streaming applications such as bio-sequence searching