GPU Performance Auto-tuning using Machine Learning

Tianyi David Han and Tarek S. Abdelrahman

**Summary:** We explore the use of machine learning techniques to automatically select the best optimization configurations for computational photography applications.

**Learning Formulation**

Why Auto-tuning?

- **GPU Program**
  - Launch config.
  - Loop order

**GPU Resources**

- Loop unrolling
- Shared memory

Optimized GPU Program

**Motivation**

- Computational photography applications
- **GPU Program**
  - Optimized GPU Program

**Why Machine Learning?**

<table>
<thead>
<tr>
<th>Performance</th>
<th>Speed</th>
<th>Effort</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analytical Modeling</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Empirical Search</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Machine Learning</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Analytical Modeling**
- **Empirical Search**
- **Machine Learning**

**Tuning Problem**

- **CUDA**
- **GPU Platform**
  - NVIDIA Tesla M2090, CUDA 5.5

**Training Kernels**

- Synthetically generated
- Random # of memory reads in the inner loop body
- Random computation length
- Random instruction dependencies

**Kernel Features**

- Compile-time / run-time
- "Raw": memory access index expression
- Derived: # of transactions induced by memory access

**Learning Algorithm**

- Support Vector Machine (SVM)
- Decision Tree
- Random Forest

**Optimization Configurations**

- 7 Parameters
- **Loop order**
- **Input caching**
- **Inner loop unrolling**
- **Launch configuration**

**Total # of valid optimization configurations = 38K**

**Learning Formulation**

**Predict the best-performing combination and configuration of optimizations for a kernel, by automatically building models based on how other kernels perform under various optimization configurations**

**Training Kernels**

- For each row
  - Unroll UF
  - For each col:
    - ... = input_img[row][col];
    - // computation
    - ... = input_img[col][row];
    - // computation
    - ...
  - output_img[row][col] = ...;

**Learning Algorithm**

- Support Vector Machine (SVM)
- Decision Tree
- Random Forest

**Experiment Setup**

- Kernels:
  - 120 synthetic kernels
  - 4 memory reads in inner loop body
- **GPU Platform**
  - NVIDIA Tesla M2090, CUDA 5.5
- **Training Set**
  - 25% of the kernels (30 kernels)
- **Test Set**
  - The remaining 90 kernels
- **Kernel Features**
  - ‘Raw’ features
  - Coefficients of each memory access index expression
  - Length of interleaved computation
  - Register usage
- **Learning Algorithm**
  - Random Forest

**Results**

- Speedup of Predicted Opt. Config. (w.r.t. Oracle)

<table>
<thead>
<tr>
<th>Model</th>
<th>H1</th>
<th>H2</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speedup</td>
<td>0.50</td>
<td>0.55</td>
<td>0.60</td>
<td>0.65</td>
<td>0.70</td>
<td>0.75</td>
</tr>
</tbody>
</table>

**What’s Next**

- Study other optimization spaces and application domains
- Further explore the design space of learning-based auto-tuning