Evaluation and Exploration of Next Generation Systems for Applicability and Performance

Volodymyr Kindratenko
Guochun Shi
Plan of work for Q4

- Develop a stand-alone C test-bed of the image extraction component of doc2learn
  - integrate the developed image probability density function computation algorithm (both the CPU and GPU implementations)
  - investigate how to extend the CPU implementation of the histogram computation to the multi-core architecture of modern CPUs
  - conduct a study how the stand-alone implementation compares to the original doc2learn Java-based implementation
  - use the stand-alone framework to analyze power consumption of the CPU and GPU implementations
    - Integrate new power monitoring hardware
- Investigate other image comparison algorithms and their suitability for GPU acceleration
- Investigate pros and cons of extending Versus framework to use GPU-based image processing algorithms
Integrate the developed image probability density function computation algorithm (both the CPU and GPU implementations)

Based on xpdf-3.02, with the following modifications

- Replaced method ImageOutputDev::drawImage with the code for computing image histogram
  - Computes histogram of an image and stores it in a file
- Added new method ImageStream::getLine to extract one image row directly into user-supplied buffer
  - Eliminates a memcpy

No GPU code has been integrated yet
Stand-alone C test-bed of the image extraction component of doc2learn

• *Investigate how to extend the CPU implementation of the histogram computation to the multi-core architecture of modern CPUs*

• Not done yet
Stand-alone C test-bed of the image extraction component of doc2learn

- Conduct a study how the stand-alone implementation compares to the original doc2learn Java-based implementation

- Work in progress
Effects of image size

- 50x50
- /tmp
Effects of image size

- 100x100
- /tmp
Effects of image size

- 150x150
- /tmp
Effects of image size

• 200x200
• /tmp
Effects of image size

- 500x500
- /tmp
Effects of image size

- 1Kx1K
- /tmp
Effects of image size

- 2Kx2K
- /tmp
Effects of file system type

- nfs
- 200x200
Effects of file system type

- /tmp
- 200x200
Effects of file system type

- ramdisk
- 200x200
Comparison with doc2learn
Comparison with doc2learn
Observations/Conclusions

• Stand-alone C-based implementation is substantially faster than the Java-based framework
  – For small images, the entire application runs faster than Java-based image analysis part of doc2learn (not even including file I/O)
  – For larger images, Java-based image analysis code is still substantially slower

• Reading images from disk takes an order of magnitude more time than to compute histograms
  – Overall application speedup of no more than 10% can be achieved by speeding up the image processing time
  – Does a 10% speedup really matter?

• Suggestions for improving doc2learn performance without GPUs:
  – Save all histograms for a given PDF file into just one output file
    • Otherwise saving computed histograms to disk takes more time than to compute them
    • fopen/fclose are very expensive
  – Use ramdisk to temporary store PDF files while processing them
    • Eliminates OS jitter due to disk access
Stand-alone C test-bed of the image extraction component of doc2learn

• *Use the stand-alone framework to analyze power consumption of the CPU and GPU implementations*
  – *Integrate new power monitoring hardware*

• Based on the new power monitoring hardware developed at ISL by Craig Steffen
  – *We wrote data acquisition and analysis scripts necessary to collect and visualize power levels*
Power consumption measurements

- /tmp
- 50-1K
- idle
  - \(~247\) watt
- computing
  - \(~265\) watt
- \(\Delta \sim 18\) watt
Power consumption measurements

- /tmp
- 50-2K
- “idle”
  - ~418 watt
- computing
  - ~432 watt
- $\Delta \sim 14$ watt
Doc2learn power efficiency analysis

- 2Kx2K images, 250 image count
Doc2learn power efficiency analysis

<table>
<thead>
<tr>
<th>Image size/ count</th>
<th>Image analysis only</th>
<th>Image extraction and analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1000x1000/250</td>
<td>2000x2000/250</td>
</tr>
<tr>
<td>t (sec)</td>
<td>2.749</td>
<td>10.322</td>
</tr>
<tr>
<td>t_c (sec)</td>
<td>1.131</td>
<td>4.575</td>
</tr>
<tr>
<td>t_g (sec)</td>
<td>0.950</td>
<td>3.763</td>
</tr>
<tr>
<td>p (watt)</td>
<td>260</td>
<td>260</td>
</tr>
<tr>
<td>p_c (watt)</td>
<td>260</td>
<td>260</td>
</tr>
<tr>
<td>p_g (watt)</td>
<td>325</td>
<td>325</td>
</tr>
<tr>
<td>s_c=t_c/t</td>
<td>2.43</td>
<td>2.26</td>
</tr>
<tr>
<td>s_g=t_g/t</td>
<td>2.89</td>
<td>2.74</td>
</tr>
<tr>
<td>e_c=p/p_c*s_c</td>
<td>2.43</td>
<td>2.26</td>
</tr>
<tr>
<td>e_g=p/p_g*s_g</td>
<td>2.31</td>
<td>2.19</td>
</tr>
</tbody>
</table>

If we **do not** take into account image extraction time (which is huge compared to the image processing time), GPU-based implementation is more power-efficient.

If we **do** take into account the image extraction time, GPU-based implementation becomes power-inefficient!
Other topics

• Investigate other image comparison algorithms and their suitability for GPU acceleration
  – Work in progress

• Investigate pros and cons of extending Versus framework to use GPU-based image processing algorithms
  – Work in progress
Future work

• Finish image analysis work
  – Perform final set of measurements
  – Write report

• New directions (after phone calls with Mark Conrad and Richard Lopez)
  – Data compression on GPUs
  – Computing file checksums on GPUs
  – Investigating applicability of database appliances for iRODS
    • XtremeData's dbX Data Warehousing Appliance