A Case Study in Optimizing GNU Radio’s ATSC Flowgraph

Presented by Greg Scallon and Kirby Cartwright

GNU Radio Conference 2017
Thursday, September 14th 10am
ATSC FLOWGRAPH LOADING

Architecture: Intel i7 multiple core processor Linux OS with Thread per Block Scheduler
L2 SIBLING COMMUNICATION

Core 2

Floating Point Engine
GPU A
GPU B
L2 Cache 256 KB, 10~

Floating Point Engine
GPU A
GPU B
L2 Cache 256 KB, 10~

L3 Cache 1.5 MB/core, 40 – 75 ~

RAM xx GB, 300+~

Core 1

Floating Point Engine
GPU A
GPU B
L2 Cache 256 KB, 10~

Floating Point Engine
GPU A
GPU B
L2 Cache 256 KB, 10~

File Source
L2 Input Buffer
Rx Filter
L2 Output Buffer
Receiver FPLL

RAM

Core 2

DC Blocker
L3 Output Buffer
Larger buffer sizes increase procedure efficiency

- Economy of scale amortizes overhead

Buffers too large overflow L2 cache memory

- Increases access overhead

Experiments with different buffer sizes shows effect:

<table>
<thead>
<tr>
<th>Default Buffer size</th>
<th>Throughput Impact</th>
</tr>
</thead>
<tbody>
<tr>
<td>64 KB</td>
<td>0 reference</td>
</tr>
<tr>
<td>56 KB</td>
<td>-11.1%</td>
</tr>
<tr>
<td>48 KB</td>
<td>3.8%</td>
</tr>
<tr>
<td>40 KB</td>
<td>3.5%</td>
</tr>
<tr>
<td>32 KB</td>
<td>2.9%</td>
</tr>
</tbody>
</table>

Exploiting L2 allocation improves performance by more than 21%

- Baseline tuning doesn’t change any ATSC application code or structures
**UPGRADE to NEWER LINUX OS RELEASE**

Throughput differences between Linux versions

<table>
<thead>
<tr>
<th>Ubuntu OS release version:</th>
<th>4.4.0</th>
<th>4.11.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline configuration</td>
<td>reference</td>
<td>0%</td>
</tr>
<tr>
<td>Allocate for L2 communication</td>
<td>23.2%</td>
<td>25.4%</td>
</tr>
</tbody>
</table>

| Net improvement            | 23.2% | 11.3% |

Latest Linux offsets 46% of the L2 allocation savings
COMPARING INTEL and AMD CHIPS

New AMD Ryzen 7 processors rival Intel i7 chips

<table>
<thead>
<tr>
<th>Characteristics</th>
<th>Intel i7 6950X</th>
<th>AMD Ryzen 7 1700</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default Clock speed (GHz.)</td>
<td>3.0</td>
<td>3.0</td>
</tr>
<tr>
<td>L1 Cache (instruction, data) KB per thread</td>
<td>32, 32</td>
<td>64, 32</td>
</tr>
<tr>
<td>L2 Cache KB per core</td>
<td>256</td>
<td>512</td>
</tr>
<tr>
<td>Number of cores each with Hyper-Thread</td>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td>L3 Cache MB shared across all cores</td>
<td>25</td>
<td>16</td>
</tr>
<tr>
<td>Max DDR4 RAM GB shared by all cores</td>
<td>128</td>
<td>64</td>
</tr>
<tr>
<td>Memory Channels</td>
<td>4</td>
<td>2</td>
</tr>
</tbody>
</table>

Larger L1 & L2 Cache promises more performance

- For executing the ATSC using L2 cache communication
INTRODUCING: KIRBY CARTWRIGHT

Using the AMD Chip for ATSC

Experience getting the ATSC application to run on a Ryzen 7 processor (and the Intel i7)

Problems and solutions
ISSUES REHOSTING to a RYZEN CHIP

AMD Ryzen chips are New

- Lack full support for both processor and different motherboards (Gigabyte ga-ab350 gaming)

Finally found a source patch for Ubuntu 4.11.2 allowing Linux to boot on the Ryzen chip

- Not sure if core allocation and threading is completely supported
- Kudos to Canonical and Ubuntu support and the Internet community for the support!

As of 5/23/17, there is no direct VOLKS library support for Ryzen

- Some overlap with Intel CPUs but not all specialized floating point instructions are supported - see me at the booth for details
ISSUES REHOSTING to a RYZEN CHIP

• On the other hand - there are great Linux run-time analysis & performance tools; e.g., top and htop. They apply directly to the AMD and Intel i7 processors, cores, and threads.

• On the third hand there are no runtime cache analysis and monitoring tools. Does anybody in the radio audience know of any?

• By the way - If you have tweaked a GRC application to run optimally on one machine - That’s all you have done. You will need different tweaking on a different machine.
RUNNING GNU ATSC on a RYZEN CHIP

The 14 ATSC blocks now load and run on the Ryzen CPU
- Once the Ubuntu kernel booted ...

Default performance is 4% slower on the AMD Ryzen
- As compared to the particular Intel configuration options previously identified

The larger L2 Cache promises more performance
- For executing the ATSC flow graph using L2 cache for inter-block communications
RYZEN BUFFER SIZE TUNING

- Experimentally tuning the ATSC buffer sizes for the ATSC Flowgraph running on the Ryzen 1700 chip

<table>
<thead>
<tr>
<th>Buffer Size (KB)</th>
<th>Performance Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>56K</td>
<td>372s</td>
</tr>
<tr>
<td>64K</td>
<td>361s</td>
</tr>
<tr>
<td>128K</td>
<td>363s</td>
</tr>
<tr>
<td>152K</td>
<td>355s</td>
</tr>
<tr>
<td>256K</td>
<td>358s</td>
</tr>
</tbody>
</table>
CHANGES THAT IMPROVED PERFORMANCE

Allocate critical blocks to fixed hyper-threads
  • Increases throughput of the critical Rx Filter block
  • Allocate L2 Cache Communications for faster access
  • Allow Turbo Boost to focus on critical Rx Filter block

Experimentally tune default buffer size allocation
  • Find the buffer size which gives the best performance

Upgrade to the latest Linux OS version release
  • Apply the latest OS system optimizations & improvements
Questions?

Contact: greg@simpleXecutive.com