Benchmarking CMS applications
[Using CMSSW to benchmark machines]

Gabriele Benelli
CERN

HEPiX Spring 2008 May 8th 2008
HEPiX cpu performance

- Test the validity of the industry-standard benchmarks (SPEC CPU) when compared with HEP experiments code
- Provide some recommendation to guide institutional purchases
- Use CMSSW applications to benchmark a number of machines with different architecture
Benchmarked machines

- Used 10 machines with different architectures, frequencies, memory:
  - 7 machines at CERN from the lxbench cluster
    - Cluster TWiki: https://twiki.cern.ch/twiki/bin/view/FIOgroup/TsiLxbench
  - 1 machine at DESY Zeuten (hpbl1)
  - 2 machines at INFN Padua (lxcmssrv7, lxcmssrv8)

<table>
<thead>
<tr>
<th></th>
<th>lxbench01</th>
<th>lxbench02</th>
<th>lxbench03</th>
<th>lxbench04</th>
<th>lxbench05</th>
<th>lxbench06</th>
<th>lxbench07</th>
<th>lxcmssrv07</th>
<th>lxcmssrv08</th>
<th>hpbl1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of cores</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>Frequency (GHz)</td>
<td>2.8</td>
<td>2.8</td>
<td>2.2</td>
<td>2.66</td>
<td>3.0</td>
<td>2.6</td>
<td>2.33</td>
<td>2.33</td>
<td>2.1</td>
<td>2.83</td>
</tr>
<tr>
<td>Cache (could be L2/L3) (MB)</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>8</td>
<td>12</td>
<td>2</td>
</tr>
<tr>
<td>Memory (GB)</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>Vendor</td>
<td>Intel</td>
<td>Intel</td>
<td>AMD</td>
<td>Intel</td>
<td>Intel</td>
<td>AMD</td>
<td>Intel</td>
<td>Intel</td>
<td>AMD</td>
<td>Intel</td>
</tr>
</tbody>
</table>

Gabriele Benelli, CERN
CMSSW benchmarking

- All tests run with CMSSW_2_0_0_pre5 release
- Used 7 different physics processes ("candles"):
  1. HiggsZZ4LM190
  2. MinBias
  3. QCD_80_120
  4. SingleElectronE1000
  5. SingleMuMinusPt10
  6. SinglePiMinusE1000
  7. TTbar
CMSSW Benchmarking

- Run 100 events per candle
- Run GEN+SIM, DIGI, RECO steps separately
- Run the 7 candles sequentially on each core
- Four tests ran up to now:
  - Loading all cores simultaneously
  - Loading 1, 3 (only for 4 cores machines) and 5 cores (only for 8 cores machines) with our application while running a cpu-intensive, cache-contained benchmarking tool (cmsScimark2) on the other cores
CMSSW Benchmarking

- The result of the benchmarking is seconds/event averaged on the 99 events (skipping the first one to avoid biases due to initialization)

- The results are reported in 3 formats:
  - seconds/events per core
  - events/seconds per core
  - events/seconds per machine

- Link: https://hepix.caspur.it/processors/dokuwiki/doku.php?id=benchmarks:cms
Comparing GEN+SIM

All cores loaded

AMD

Machine Maximum Performance

Events/second

SinglePiMinusE1000
QCD_80_120
TTbar

kbench01  kbencn02  kbenc03  kbenc04  kbenc05  kbenc06  kbenc07  xmserv07  xmserv08  npol1
Machine Maximum Performance Normalized by Frequency

- SinglePiMinusE1000
- QCD_80_120
- Ttbar

All cores loaded

Gabriele Benelli, CERN
GEN+SIM normalized by frequency and # cores

All cores loaded
Comparing RECO

Machine Maximum Performance

Events/second

AMD

All cores loaded

Gabriele Benelli, CERN
RECO normalized by frequency

Machine Maximum Performance Normalized by Frequency

- SinglePiMinusE1000
- QCD_80_120
- TTbar

All cores loaded

Gabriele Benelli, CERN
RECO normalized by frequency and # cores

Machine Maximum Performance Normalized by Frequency, Number Of Cores

- SinglePiMinusE1000
- QCD_80_120
- TTbar

All cores loaded
RECO vs SPECint2006

Machine Maximum Performance Normalized by SPECint2006

- SinglePiMinusE1000
- QCD_80_120
- TTbar

![Bar chart showing performance normalized by SPECint2006]
SIM vs SPECfp2006

Machine Maximum Performance Normalized by SPECfp2006
Excellent scaling to the last core (cmsScimark2)
Scaling with multicores

Excellent scaling to the last core (cmsScimark2)
Conclusions

- Used CMSSW to benchmark 10 machines
- Observed a different behavior in AMD vs. Intel machines for complex vs. simple events at the RECO step
- Compared CMSSW applications with SPEC benchmarks: differences due to architecture/type of event are larger than the differences between different SPEC benchmarks
- The CMSSW application scales nicely with the current multicore architectures
- Work is ongoing (data analysis, more tests, developing a benchmarking suite to distribute)
All Cores DIGI

Machine Maximum Performance

- SinglePiMinusE1000
- QCD_80_120
- TTbar
All Cores SIM freq core

Machine Maximum Performance Normalized by Frequency, Number Of Cores

- SinglePiMinusE1000
- QCD_80_120
- TTbar

Events/(second)/(frequency)/(cores)

kbench01, kbench02, kbench03, kbench04, kbench05, kbench06, kbench07, kmessrv07, kmessrv08, kphil
CMSSW vs SPEC
Data Playback

- Emulation of High Level Trigger mode of operation
- Several processes (EP's) per node analyzing data as they were provided by the DAQ
- “Building” of the events performed in the same CPU
- Shared Memory used to exchange event data
CMS HLT validation farm

✓ 1 rack (20 PC’s) of the CMS DAQ farm at LHC P5:
  ▪ Dual dual-core
  ▪ CPU: Intel Xeon 5130 @2.00 GHz, 4 MB L2 cache
  ▪ 8 GB memory
✓ Up to 4 EventProcessor’s (cmsRun equivalent) per node.
  1000 events per node

Marco Zanetti CERN PH
Results: HiLTon

- Number of events processed per second per node scaled nicely with number of EP’s
- Hilton machines have 4 cores, but even with 5 processes per node (4 EP + 1 BU) the EP’s take ~100% of the CPU

Marco Zanetti CERN PH
Results: Bench07

- Scaling valid up to 8 EP's on lxbench07 (dual quad-core)
- Better performances due to higher CPU frequency
Scaling with frequency

✓ Nice scaling with CPU frequency (1 EP per node)
✓ (not the same configuration w.r.t what shown before, i.e. not same timing)
### TTbar numbers

#### TTbar **per core** (when running on all cores at once)

<table>
<thead>
<tr>
<th></th>
<th>lxbench01</th>
<th>lxbench02</th>
<th>lxbench03</th>
<th>lxbench04</th>
<th>lxbench05</th>
<th>lxbench06</th>
<th>lxbench07</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 cores</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s/evt</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GEN+SIM</td>
<td>210.207</td>
<td>208.060</td>
<td>134.939</td>
<td>109.988</td>
<td>98.194</td>
<td>112.939</td>
<td>124.910</td>
</tr>
<tr>
<td>DIGI</td>
<td>3.361</td>
<td>3.688</td>
<td>2.408</td>
<td>1.869</td>
<td>1.674</td>
<td>2.047</td>
<td>2.112</td>
</tr>
<tr>
<td>RECO</td>
<td>17.987</td>
<td>17.739</td>
<td>18.482</td>
<td>9.610</td>
<td>8.570</td>
<td>15.782</td>
<td>10.906</td>
</tr>
<tr>
<td>TOTAL(SUM)</td>
<td>231.555</td>
<td>229.487</td>
<td>155.829</td>
<td>121.467</td>
<td>108.438</td>
<td>130.768</td>
<td>137.928</td>
</tr>
</tbody>
</table>

#### TTbar **per core** (when running on all cores at once)

<table>
<thead>
<tr>
<th></th>
<th>lxbench01</th>
<th>lxbench02</th>
<th>lxbench03</th>
<th>lxbench04</th>
<th>lxbench05</th>
<th>lxbench06</th>
<th>lxbench07</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 cores</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>evts/s</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GEN+SIM</td>
<td>0.004757</td>
<td>0.004806</td>
<td>0.007411</td>
<td>0.009092</td>
<td>0.010184</td>
<td>0.008854</td>
<td>0.008006</td>
</tr>
<tr>
<td>DIGI</td>
<td>0.297530</td>
<td>0.271150</td>
<td>0.415282</td>
<td>0.535045</td>
<td>0.597372</td>
<td>0.488520</td>
<td>0.473485</td>
</tr>
<tr>
<td>RECO</td>
<td>0.055596</td>
<td>0.056373</td>
<td>0.054107</td>
<td>0.104058</td>
<td>0.116686</td>
<td>0.063363</td>
<td>0.091693</td>
</tr>
<tr>
<td>TOTAL(SUM)</td>
<td>0.004319</td>
<td>0.004358</td>
<td>0.006417</td>
<td>0.008233</td>
<td>0.009222</td>
<td>0.007647</td>
<td>0.007250</td>
</tr>
</tbody>
</table>

#### TTbar **per machine** (when running on all cores at once)

<table>
<thead>
<tr>
<th></th>
<th>lxbench01</th>
<th>lxbench02</th>
<th>lxbench03</th>
<th>lxbench04</th>
<th>lxbench05</th>
<th>lxbench06</th>
<th>lxbench07</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 cores</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>evts/s</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GEN+SIM</td>
<td>0.009514</td>
<td>0.009612</td>
<td>0.029644</td>
<td>0.036358</td>
<td>0.040736</td>
<td>0.035416</td>
<td>0.064048</td>
</tr>
<tr>
<td>DIGI</td>
<td>0.595060</td>
<td>0.542300</td>
<td>1.661128</td>
<td>2.140180</td>
<td>2.389488</td>
<td>1.954080</td>
<td>3.787880</td>
</tr>
<tr>
<td>RECO</td>
<td>0.111192</td>
<td>0.112746</td>
<td>0.216428</td>
<td>0.416232</td>
<td>0.466744</td>
<td>0.253452</td>
<td>0.733544</td>
</tr>
<tr>
<td>TOTAL(SUM)</td>
<td>0.008638</td>
<td>0.008716</td>
<td>0.025668</td>
<td>0.032932</td>
<td>0.036888</td>
<td>0.030588</td>
<td>0.058000</td>
</tr>
</tbody>
</table>
### TTbar numbers

**TTbar **"per core"** (when running on 1 core only, cpu1, while running cmsScimark on the other cores)**

<table>
<thead>
<tr>
<th></th>
<th>lxbench01</th>
<th>lxbench02</th>
<th>lxbench03</th>
<th>lxbench04</th>
<th>lxbench05</th>
<th>lxbench06</th>
<th>lxbench07</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>2 cores</strong></td>
<td>s/evt</td>
<td>s/evt</td>
<td>s/evt</td>
<td>s/evt</td>
<td>s/evt</td>
<td>s/evt</td>
<td>s/evt</td>
</tr>
<tr>
<td>GEN+SIM</td>
<td>209.565</td>
<td>210.290</td>
<td>134.687</td>
<td>110.259</td>
<td>98.406</td>
<td>112.645</td>
<td>122.560</td>
</tr>
<tr>
<td>DIGI</td>
<td>3.335</td>
<td>3.673</td>
<td>2.468</td>
<td>1.798</td>
<td>1.622</td>
<td>2.098</td>
<td>2.021</td>
</tr>
<tr>
<td>RECO</td>
<td>17.782</td>
<td>17.653</td>
<td>18.477</td>
<td>9.488</td>
<td>8.460</td>
<td>15.766</td>
<td>10.794</td>
</tr>
<tr>
<td><strong>TOTAL(SUM)</strong></td>
<td>230.682</td>
<td>231.616</td>
<td>155.632</td>
<td>121.545</td>
<td>108.488</td>
<td>130.509</td>
<td>135.375</td>
</tr>
</tbody>
</table>

Events/Second per core figure for candle TTbar:

<table>
<thead>
<tr>
<th></th>
<th>s/evt</th>
<th>s/evt</th>
<th>s/evt</th>
<th>s/evt</th>
<th>s/evt</th>
<th>s/evt</th>
<th>s/evt</th>
</tr>
</thead>
<tbody>
<tr>
<td>GEN+SIM</td>
<td>0.004772</td>
<td>0.004755</td>
<td>0.007425</td>
<td>0.009070</td>
<td>0.010162</td>
<td>0.008877</td>
<td>0.008159</td>
</tr>
<tr>
<td>DIGI</td>
<td>0.299850</td>
<td>0.272257</td>
<td>0.405186</td>
<td>0.556174</td>
<td>0.616523</td>
<td>0.476644</td>
<td>0.494805</td>
</tr>
<tr>
<td>RECO</td>
<td>0.056237</td>
<td>0.056648</td>
<td>0.054121</td>
<td>0.105396</td>
<td>0.118203</td>
<td>0.063428</td>
<td>0.092644</td>
</tr>
<tr>
<td><strong>TOTAL(SUM)</strong></td>
<td>0.004335</td>
<td>0.004317</td>
<td>0.006425</td>
<td>0.008227</td>
<td>0.009218</td>
<td>0.007662</td>
<td>0.007387</td>
</tr>
</tbody>
</table>