WaveScript Benchmarks Performance Report

September 3, 2008

Machine information:
Linux chastity 2.6.22-14-generic #1 SMP Tue Feb 12 07:42:25 UTC 2008 i686 GNU/Linux

WaveScript SVN:
Revision: 3554

WaveScope Engine SVN:
(omitted for now)

1 Microbenchmarks

This section reports various microbenchmarks that stress the implementation of particular language constructs or data types.
Per-stream-element overheads

One thing that you can see, is that currently (2007.10) the C++/XStream engine has a high per-tuple (that is, per-element) on the communication channels relative to the ML backend. The just_timer test stresses this, doing nothing but passing a large number of unit tuples.

Focusing on scheduling overheads a bit more, we turn to the following data passing microbenchmarks. These do nothing but generate a stream of numbers, and then add up windows of those numbers. We vary the window size in the following graphs. The numbers are passed either one at a time (“raw”), or in bulk using arrays or lists.

Notes:
• FFT results for Scheme above depend on whether or not it is configured to use FFTW, or a native Scheme fourier transform.

2 Language Shootout Benchmarks

This is where I will accumulate some of the small benchmarks from the language shootout. Here are some per-benchmark comments:

• **fannkuch** - “pancake flipping”. This is a translation of the gcc version of the benchmark. Tests indexed access to a small array.

3 Application Benchmarks

This section includes performance results on larger programs, namely, our current applications. Presently (2007.10) the largest of these by far is the marmot application.

3.1 Marmot Application

We start off by looking at the original, hand-optimized marmot application that we deployed.
4 Data Representation Profiling

This is stale data for now... having sneaky problems with the datarep Makefile that are hosing regression tests. [2007.11.07]

This section includes an analysis of the efficiency of different data representations under different backends. This should theoretically be run on different hardware platforms as well (such as the ARM-based ensboxes).

4.1 Arrays of Arrays

Arrays of arrays are notable because they cannot generally be flattened (the inner arrays will always be pointers). In the future we may look at tentative flattening based on profiling data. But first, here are the times for repeatedly allocating an array of arrays, and for repeatedly folding the values in an array of arrays.

Next we look at allocating arrays of tuples and vice versa. We look at both square sizes and at highly skewed dimensions. This is limited by not being able to make tuples very large.
Then we do examine folding over arrays of tuples and tuples of arrays.

A Appendix: Raw numbers for above graphs

Microbenchmarks

## User time for each benchmark/backend

Benchmark mltonO3 c2boehm c2boehmseglist c2 c2seglist c2def c2defseglist
just_timer 2520.000 2536.000 2536.000 2508.000 2560.000 5044.000 5036.000
readfile_bigwins 3772.000 512.000 1148.000 764.000 3988.000 240.000 1024.000
printing_lists 2636.000 896.000 908.000 852.000 868.000 800.000 852.000
conv_SigsegArr 2308.000 396.000 7320.000 820.000 5588.000 60.000 6712.000
fft 112.000 1032.000 952.000 936.000 868.000 800.000 852.000

Language Shootout:

## User time for each language-shootout benchmark/backend

Benchmark c2
fannkuch2 4472.000

Application Benchmarks:

Benchmark mltonO3 c2boehm c2boehmseglist c2 c2seglist c2def c2defseglist
## Running orig marmot phase 1
run_first_phase 7324.000 12093.000 4136.000 7520.000 5532.000 7392.000 4024.000
## Running marmot2
test_marmot2 2204.000 5236.000 5288.000 4676.000 4672.000 4548.000 4572.000
## Running marmot3
test_heatmap 7776.000 3260.000 3252.000 2532.000 2548.000 3236.000 3268.000
## Running marmot multinode offline
run_3phases 9741.000 6004.000 4828.000 5604.000 5268.000 5608.000 4864.000

B Appendix: Additional system information

Top results before running benchmarks:

top - 09:25:30 up 43 days, 17:59, 6 users, load average: 4.29, 3.10, 2.72
Tasks: 180 total, 2 running, 178 sleeping, 0 stopped, 0 zombie
Cpu(s): 26.2%us, 4.3%sy, 0.9%ni, 67.6%id, 0.1%wa, 0.4%hi, 0.5%si, 0.0%st
Mem: 2073956k total, 1450724k used, 623232k free, 156340k buffers
Swap: 14996668k total, 34744k used, 14961924k free, 869000k cached

<table>
<thead>
<tr>
<th>PID</th>
<th>USER</th>
<th>PR</th>
<th>NI</th>
<th>VIRT</th>
<th>RES</th>
<th>SHR</th>
<th>S</th>
<th>%CPU</th>
<th>%MEM</th>
<th>TIME+</th>
<th>COMMAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>18725</td>
<td>newton</td>
<td>25</td>
<td>0</td>
<td>34208</td>
<td>31m</td>
<td>492</td>
<td>R</td>
<td>99</td>
<td>1.6</td>
<td>157:30.47</td>
<td>ikarus</td>
</tr>
<tr>
<td>30078</td>
<td>newton</td>
<td>15</td>
<td>0</td>
<td>28092</td>
<td>19m</td>
<td>4048</td>
<td>S</td>
<td>16</td>
<td>1.0</td>
<td>147:24.10</td>
<td>unison-2.18</td>
</tr>
<tr>
<td>28787</td>
<td>newton</td>
<td>18</td>
<td>0</td>
<td>1700</td>
<td>452</td>
<td>372</td>
<td>D</td>
<td>6</td>
<td>0.0</td>
<td>0:02.65</td>
<td>rm</td>
</tr>
<tr>
<td>6353</td>
<td>root</td>
<td>15</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>2</td>
<td>0.0</td>
<td>360:24.15</td>
<td>afs_rxlistener</td>
</tr>
<tr>
<td>32643</td>
<td>newton</td>
<td>15</td>
<td>0</td>
<td>15836</td>
<td>11m</td>
<td>4200</td>
<td>S</td>
<td>2</td>
<td>0.6</td>
<td>29:56.18</td>
<td>emacs21-x</td>
</tr>
</tbody>
</table>

Top results after running benchmarks:

top - 09:45:00 up 43 days, 18:19, 6 users, load average: 1.20, 1.42, 1.80
Tasks: 173 total, 1 running, 172 sleeping, 0 stopped, 0 zombie
Cpu(s): 26.2%us, 4.3%sy, 0.9%ni, 67.6%id, 0.1%wa, 0.4%hi, 0.5%si, 0.0%st
Mem: 2073956k total, 1065012k used, 1008944k free, 33872k buffers
Swap: 14996668k total, 14961924k used, 38732k free, 696080k cached

<table>
<thead>
<tr>
<th>PID</th>
<th>USER</th>
<th>PR</th>
<th>NI</th>
<th>VIRT</th>
<th>RES</th>
<th>SHR</th>
<th>S</th>
<th>%CPU</th>
<th>%MEM</th>
<th>TIME+</th>
<th>COMMAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>30078</td>
<td>newton</td>
<td>15</td>
<td>0</td>
<td>29556</td>
<td>21m</td>
<td>4048</td>
<td>S</td>
<td>12</td>
<td>1.0</td>
<td>149:27.86</td>
<td>unison-2.18</td>
</tr>
<tr>
<td>1 root</td>
<td>19</td>
<td>0</td>
<td>2948</td>
<td>1856</td>
<td>532</td>
<td>S</td>
<td>0</td>
<td>0.1</td>
<td>0:05.37</td>
<td>init</td>
<td></td>
</tr>
<tr>
<td>2 root</td>
<td>11</td>
<td>-5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>0</td>
<td>0.0</td>
<td>0:00.00</td>
<td>kthread</td>
<td></td>
</tr>
<tr>
<td>3 root</td>
<td>RT</td>
<td>-5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>0</td>
<td>0.0</td>
<td>0:00.40</td>
<td>migration/0</td>
<td></td>
</tr>
<tr>
<td>4 root</td>
<td>34</td>
<td>19</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>0</td>
<td>0.0</td>
<td>0:00.70</td>
<td>ksoftirqd/0</td>
<td></td>
</tr>
<tr>
<td>5 root</td>
<td>RT</td>
<td>-5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>0</td>
<td>0.0</td>
<td>0:00.00</td>
<td>watchdog/0</td>
<td></td>
</tr>
<tr>
<td>6 root</td>
<td>RT</td>
<td>-5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>0</td>
<td>0.0</td>
<td>0:00.33</td>
<td>migration/1</td>
<td></td>
</tr>
<tr>
<td>7 root</td>
<td>34</td>
<td>19</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>0</td>
<td>0.0</td>
<td>0:01.18</td>
<td>ksoftirqd/1</td>
<td></td>
</tr>
<tr>
<td>8 root</td>
<td>RT</td>
<td>-5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>0</td>
<td>0.0</td>
<td>0:00.00</td>
<td>watchdog/1</td>
<td></td>
</tr>
<tr>
<td>9 root</td>
<td>10</td>
<td>-5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>0</td>
<td>0.0</td>
<td>0:00.03</td>
<td>events/0</td>
<td></td>
</tr>
<tr>
<td>10 root</td>
<td>10</td>
<td>-5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>0</td>
<td>0.0</td>
<td>0:00.04</td>
<td>events/1</td>
<td></td>
</tr>
<tr>
<td>11 root</td>
<td>10</td>
<td>-5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>0</td>
<td>0.0</td>
<td>0:00.02</td>
<td>khelper</td>
<td></td>
</tr>
<tr>
<td>31 root</td>
<td>10</td>
<td>-5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>0</td>
<td>0.0</td>
<td>0:00.52</td>
<td>kblockd/0</td>
<td></td>
</tr>
</tbody>
</table>