WaveScript Benchmarks Perfomance Report

August 13, 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: 3480

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

```bash
## User time for each benchmark/backend
Benchmark mltonO3 c2boehm c2boehmseglist c2 c2seglist c2def c2defseglist
just_timer 2528.000 2520.000 2516.000 2524.000 2516.000 5040.000 5036.000
readfile_bigwins 3636.000 500.000 1092.000 768.000 3928.000 288.000 996.000
printing_lists 2132.000 900.000 904.000 844.000 896.000 808.000 804.000
conv_SigsegArr 2328.000 388.000 7328.000 788.000 6420.000 36.000 6376.000
fft 112.000 960.000 912.000 860.000 868.000 940.000 852.000
```

Language Shootout:

```bash
## User time for each language-shootout benchmark/backend
Benchmark c2
fannkuch2 4524.000
```

Application Benchmarks:

```bash
Benchmark mltonO3 c2boehm c2boehmseglist c2 c2seglist c2def c2defseglist
## Running orig marmot phase 1
```
run_first_phase 1504.000 2464.000 852.000 1544.000 1144.000 1752.000 772.000

## Running marmot2
test_marmot2 2308.000 5292.000 5264.000 4648.000 4712.000 4576.000 4556.000

## Running marmot3
test_heatmap 7836.000 3276.000 3292.000 2624.000 2620.000 3280.000 3292.000

## Running marmot multinode offline
run_3phases 9365.000 6220.000 4936.000 5668.000 5360.000 5644.000 4836.000

B Appendix: Additional system information

Top results before running benchmarks:

top - 17:22:10 up 23 days, 1:56, 5 users, load average: 2.22, 1.95, 1.26
Tasks: 163 total, 2 running, 161 sleeping, 0 stopped, 0 zombie
Cpu(s): 34.7%us, 4.4%sy, 1.7%ni, 58.2%id, 0.1%wa, 0.4%hi, 0.4%si, 0.0%st
Mem: 20739568k total, 1356968k used, 716988k free, 31676k buffers
Swap: 14996668k total, 34752k used, 14961916k free, 960836k 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>10742</td>
<td>newton</td>
<td>21</td>
<td>0</td>
<td>66848</td>
<td>15m</td>
<td>4396 R</td>
<td>101</td>
<td>0.8</td>
<td>0:00.64</td>
<td>memcheck</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>root</td>
<td>16</td>
<td>0</td>
<td>2948</td>
<td>1856</td>
<td>532 S</td>
<td>0</td>
<td>0.1</td>
<td>0:04.35</td>
<td>init</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>root</td>
<td>10</td>
<td>-5</td>
<td>0</td>
<td>0</td>
<td>0 S</td>
<td>0</td>
<td>0.0</td>
<td>0:00.00</td>
<td>kthread</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>root</td>
<td>RT</td>
<td>-5</td>
<td>0</td>
<td>0</td>
<td>0 S</td>
<td>0</td>
<td>0.0</td>
<td>0:00.20</td>
<td>migration/0</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>root</td>
<td>34</td>
<td>19</td>
<td>0</td>
<td>0</td>
<td>0 S</td>
<td>0</td>
<td>0.0</td>
<td>0:00.52</td>
<td>ksoftirqd/0</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>root</td>
<td>RT</td>
<td>-5</td>
<td>0</td>
<td>0</td>
<td>0 S</td>
<td>0</td>
<td>0.0</td>
<td>0:00.00</td>
<td>watchdog/0</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>root</td>
<td>RT</td>
<td>-5</td>
<td>0</td>
<td>0</td>
<td>0 S</td>
<td>0</td>
<td>0.0</td>
<td>0:00.16</td>
<td>migration/1</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>root</td>
<td>34</td>
<td>19</td>
<td>0</td>
<td>0</td>
<td>0 S</td>
<td>0</td>
<td>0.0</td>
<td>0:00.94</td>
<td>ksoftirqd/1</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>root</td>
<td>RT</td>
<td>-5</td>
<td>0</td>
<td>0</td>
<td>0 S</td>
<td>0</td>
<td>0.0</td>
<td>0:00.00</td>
<td>watchdog/1</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>root</td>
<td>10</td>
<td>-5</td>
<td>0</td>
<td>0</td>
<td>0 S</td>
<td>0</td>
<td>0.0</td>
<td>0:00.03</td>
<td>events/0</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>root</td>
<td>10</td>
<td>-5</td>
<td>0</td>
<td>0</td>
<td>0 S</td>
<td>0</td>
<td>0.0</td>
<td>0:00.04</td>
<td>events/1</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>root</td>
<td>10</td>
<td>-5</td>
<td>0</td>
<td>0</td>
<td>0 S</td>
<td>0</td>
<td>0.0</td>
<td>0:00.01</td>
<td>khelper</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>root</td>
<td>14</td>
<td>-5</td>
<td>0</td>
<td>0</td>
<td>0 S</td>
<td>0</td>
<td>0.0</td>
<td>0:00.24</td>
<td>kblockd/0</td>
<td></td>
</tr>
</tbody>
</table>

Top results after running benchmarks:

top - 17:40:16 up 23 days, 2:14, 5 users, load average: 2.18, 2.24, 1.90
Tasks: 169 total, 2 running, 167 sleeping, 0 stopped, 0 zombie
Cpu(s): 34.8%us, 4.4%sy, 1.7%ni, 58.1%id, 0.1%wa, 0.4%hi, 0.4%si, 0.0%st
Mem: 20739568k total, 1560408k used, 513548k free, 3776k buffers
Swap: 14996668k total, 34752k used, 14961916k free, 1136576k 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>27505</td>
<td>newton</td>
<td>25</td>
<td>0</td>
<td>29324</td>
<td>24m</td>
<td>2080 S</td>
<td>61</td>
<td>1.2</td>
<td>0:00.31</td>
<td>mzscheme</td>
<td></td>
</tr>
<tr>
<td>27443</td>
<td>newton</td>
<td>22</td>
<td>0</td>
<td>43368</td>
<td>40m</td>
<td>504 R</td>
<td>36</td>
<td>2.0</td>
<td>0:01.00</td>
<td>ikarus</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>root</td>
<td>16</td>
<td>0</td>
<td>2948</td>
<td>1856</td>
<td>532 S</td>
<td>0</td>
<td>0.1</td>
<td>0:04.36</td>
<td>init</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>root</td>
<td>10</td>
<td>-5</td>
<td>0</td>
<td>0</td>
<td>0 S</td>
<td>0</td>
<td>0.0</td>
<td>0:00.00</td>
<td>kthread</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>root</td>
<td>RT</td>
<td>-5</td>
<td>0</td>
<td>0</td>
<td>0 S</td>
<td>0</td>
<td>0.0</td>
<td>0:00.21</td>
<td>migration/0</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>root</td>
<td>34</td>
<td>19</td>
<td>0</td>
<td>0</td>
<td>0 S</td>
<td>0</td>
<td>0.0</td>
<td>0:00.52</td>
<td>ksoftirqd/0</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>root</td>
<td>RT</td>
<td>-5</td>
<td>0</td>
<td>0</td>
<td>0 S</td>
<td>0</td>
<td>0.0</td>
<td>0:00.00</td>
<td>watchdog/0</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>root</td>
<td>RT</td>
<td>-5</td>
<td>0</td>
<td>0</td>
<td>0 S</td>
<td>0</td>
<td>0.0</td>
<td>0:00.16</td>
<td>migration/1</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>root</td>
<td>34</td>
<td>19</td>
<td>0</td>
<td>0</td>
<td>0 S</td>
<td>0</td>
<td>0.0</td>
<td>0:00.94</td>
<td>ksoftirqd/1</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>root</td>
<td>RT</td>
<td>-5</td>
<td>0</td>
<td>0</td>
<td>0 S</td>
<td>0</td>
<td>0.0</td>
<td>0:00.00</td>
<td>watchdog/1</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>root</td>
<td>10</td>
<td>-5</td>
<td>0</td>
<td>0</td>
<td>0 S</td>
<td>0</td>
<td>0.0</td>
<td>0:00.03</td>
<td>events/0</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>root</td>
<td>10</td>
<td>-5</td>
<td>0</td>
<td>0</td>
<td>0 S</td>
<td>0</td>
<td>0.0</td>
<td>0:00.04</td>
<td>events/1</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>root</td>
<td>10</td>
<td>-5</td>
<td>0</td>
<td>0</td>
<td>0 S</td>
<td>0</td>
<td>0.0</td>
<td>0:00.01</td>
<td>khelper</td>
<td></td>
</tr>
</tbody>
</table>

6