WaveScript Benchmarks Performance Report

August 29, 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: 3547

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 2532.000 2516.000 2512.000 2512.000 2512.000 5024.000 5028.000
readfile_bigwins 3800.000 504.000 1048.000 752.000 3944.000 316.000 992.000
printing_lists 2672.000 900.000 944.000 876.000 848.000 816.000 812.000
conv_SigsegArr 2276.000 348.000 7440.000 840.000 5544.000 44.000 6748.000
fft 132.000 900.000 972.000 884.000 936.000 816.000 812.000

Language Shootout:

## User time for each language-shootout benchmark/backend
Benchmark c2
fannkuch2 4480.000

Application Benchmarks:

Benchmark mltonO3 c2boehm c2boehmseglist c2 c2seglist c2def c2defseglist
## Running orig marmot phase 1
# Running marmot2

test_marmot2 2224.000 5368.000 5304.000 4676.000 4644.000 4584.000 4560.000

# Running marmot3

test_heatmap 7836.000 3216.000 3312.000 2544.000 2552.000 3212.000 3176.000

# Running marmot multinode offline
	n_run_3phases 9549.000 6080.000 4872.000 5668.000 5316.000 5760.000 4844.000

## Appendix: Additional system information

### Top results before running benchmarks:

```
run First phase 7724.000 12601.000 4196.000 7376.000 5548.000 8369.000 3864.000
```

```
## Running marmot2

test marmot2 2224.000 5368.000 5304.000 4676.000 4644.000 4584.000 4560.000

## Running marmot3

test heatmap 7836.000 3216.000 3312.000 2544.000 2552.000 3212.000 3176.000

## Running marmot multinode offline
	run 3phases 9549.000 6080.000 4872.000 5668.000 5316.000 5760.000 4844.000
```

### B Appendix: Additional system information

#### Top results before running benchmarks:

```
top - 15:52:23 up 39 days, 26 min, 6 users, load average: 2.23, 2.58, 2.50
Tasks: 176 total, 2 running, 174 sleeping, 0 stopped, 0 zombie
Cpu(s): 25.0%us, 4.4%sy, 1.0%ni, 68.5%id, 0.1%wa, 0.4%hi, 0.5%si, 0.0%st
Mem: 2073956k total, 964120k used, 1109836k free, 50440k buffers
Swap: 14996668k total, 34752k used, 14961916k free, 515796k 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>9689</td>
<td>newton</td>
<td>25</td>
<td>0</td>
<td>27140</td>
<td>22m</td>
<td>3364 R</td>
<td>99</td>
<td>1.1</td>
<td>0:00.76</td>
<td>cc1</td>
<td></td>
</tr>
<tr>
<td>25064</td>
<td>newton</td>
<td>15</td>
<td>0</td>
<td>48940</td>
<td>40m</td>
<td>4048 S</td>
<td>14</td>
<td>2.0</td>
<td>761:41.07</td>
<td>unison-2.18</td>
<td></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>2</td>
<td>0.0</td>
<td>340:36.43</td>
<td>afs_rxlistener</td>
<td></td>
</tr>
<tr>
<td>32643</td>
<td>newton</td>
<td>15</td>
<td>0</td>
<td>15836</td>
<td>11m</td>
<td>4200 S</td>
<td>2</td>
<td>0.6</td>
<td>3:06:79</td>
<td>emacs21-x</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>root</td>
<td>18</td>
<td>0</td>
<td>2948</td>
<td>1856</td>
<td>532 S</td>
<td>0</td>
<td>0.1</td>
<td>0:05.13</td>
<td>init</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>root</td>
<td>13</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>0.0</td>
<td>0:00.00</td>
<td>kthread</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>root</td>
<td>10</td>
<td>-5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>0.0</td>
<td>0:00.37</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</td>
<td>S</td>
<td>0.0</td>
<td>0:00.70</td>
<td>ksoftirqd/0</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>root</td>
<td>10</td>
<td>-5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</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>34</td>
<td>19</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>0.0</td>
<td>0:01.16</td>
<td>ksoftirqd/1</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>root</td>
<td>10</td>
<td>-5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>0.0</td>
<td>0:00.00</td>
<td>watchdog/1</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>root</td>
<td>10</td>
<td>-5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>0.0</td>
<td>0:00.03</td>
<td>events/0</td>
<td></td>
</tr>
</tbody>
</table>

#### Top results after running benchmarks:

```
top - 16:12:53 up 39 days, 47 min, 6 users, load average: 2.28, 2.26, 2.34
Tasks: 178 total, 2 running, 176 sleeping, 0 stopped, 0 zombie
Cpu(s): 25.1%us, 4.4%sy, 1.0%ni, 68.5%id, 0.1%wa, 0.4%hi, 0.5%si, 0.0%st
Mem: 2073956k total, 730120k used, 1343836k free, 4316k buffers
Swap: 14996668k total, 34752k used, 14961912k free, 334780k 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>30021</td>
<td>newton</td>
<td>25</td>
<td>0</td>
<td>42864</td>
<td>40m</td>
<td>468 R</td>
<td>101</td>
<td>2.0</td>
<td>0:00.88</td>
<td>ikarus</td>
<td></td>
</tr>
<tr>
<td>25064</td>
<td>newton</td>
<td>15</td>
<td>0</td>
<td>48940</td>
<td>40m</td>
<td>4048 S</td>
<td>14</td>
<td>2.0</td>
<td>764:02.90</td>
<td>unison-2.18</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>root</td>
<td>18</td>
<td>0</td>
<td>2948</td>
<td>1856</td>
<td>532 S</td>
<td>0</td>
<td>0.1</td>
<td>0:05.16</td>
<td>init</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>root</td>
<td>13</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>0.0</td>
<td>0:00.00</td>
<td>kthread</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>root</td>
<td>10</td>
<td>-5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>0.0</td>
<td>0:00.37</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</td>
<td>S</td>
<td>0.0</td>
<td>0:00.70</td>
<td>ksoftirqd/0</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>root</td>
<td>10</td>
<td>-5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</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>34</td>
<td>19</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>0.0</td>
<td>0:01.17</td>
<td>ksoftirqd/1</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>root</td>
<td>10</td>
<td>-5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>0.0</td>
<td>0:00.03</td>
<td>events/0</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>root</td>
<td>10</td>
<td>-5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>0.0</td>
<td>0:00.04</td>
<td>events/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</td>
<td>S</td>
<td>0.0</td>
<td>0:00.02</td>
<td>khelper</td>
<td></td>
</tr>
</tbody>
</table>