WaveScript Benchmarks Performance Report

October 22, 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: 3603

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

```
## Real or User time for each benchmark/backend
## LD_PRELOAD:
## NOSUDO:
## NICE:
Benchmark mlton03 c2boehm c2boehmseglist c2 c2seglist c2def c2defseglist  
just_timer 2556.000 2516.000 2572.000 2528.000 2508.000 5056.000 5032.000  
readfile_bigwins 3760.000 1112.000 1272.000 3768.000 272.000 1028.000  
printing_lists 2676.000 492.000 1112.000 1272.000 3768.000 272.000 1028.000  
conv_SigsegArr 2312.000 832.000 836.000 984.000 5696.000 40.000 6552.000  
fft 124.000 904.000 832.000 836.000 984.000 876.000 852.000  
```

Language Shootout:

```
## Real or User time for each benchmark/backend
## LD_PRELOAD:
## NOSUDO:
## NICE:
Benchmark c2  
fannkuch2 4312.000  
```
Application Benchmarks:

## Real or User time for each benchmark/backend
## LD_PRELOAD:
## NOSUDO:
## NICE:

Benchmark mltonO3 c2boehm c2boehmsegslist c2 c2seglist c2def c2defseglist
## Running orig marmot phase 1
run_first_phase 7092.000 11945.000 4156.000 7424.000 5588.000 7160.000 3784.000
## Running marmot2
test_marmot2 2332.000 5232.000 5224.000 4640.000 4700.000 4580.000 4560.000
## Running marmot3
test_heatmap 7736.000 3220.000 3256.000 2524.000 2564.000 3208.000 3248.000
## Running marmot multinode offline
run_3phases 9465.000 5908.000 5924.000 5620.000 5284.000 5500.000 4828.000

B Appendix: Additional system information

Top results before running benchmarks:

```
top - 11:56:02 up 92 days, 20:30, 3 users, load average: 1.00, 1.04, 0.80
Tasks: 175 total, 1 running, 174 sleeping, 0 stopped, 0 zombie
Cpu(s): 27.1%us, 3.0%sy, 0.8%ni, 68.5%id, 0.2%wa, 0.2%hi, 0.2%si, 0.0%st
Mem: 2073956k total, 1625964k used, 447992k free, 121548k buffers
Swap: 14996668k total, 347992k free, 14648776k used, 1112320k 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>1</td>
<td>root</td>
<td>18</td>
<td>0</td>
<td>2948</td>
<td>1856</td>
<td>532</td>
<td>S</td>
<td>0.1</td>
<td>0.00</td>
<td>0:07.42</td>
<td>init</td>
</tr>
<tr>
<td>2</td>
<td>root</td>
<td>13</td>
<td>-5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>0.0</td>
<td>0.00</td>
<td>0:00</td>
<td>kthread</td>
</tr>
<tr>
<td>3</td>
<td>root</td>
<td>RT</td>
<td>-5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>0.0</td>
<td>0.04</td>
<td>0:04</td>
<td>migration/0</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>0.14</td>
<td>0:14</td>
<td>ksoftirq/0</td>
</tr>
<tr>
<td>5</td>
<td>root</td>
<td>RT</td>
<td>-5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>0.0</td>
<td>0.00</td>
<td>0:00</td>
<td>watchdog/0</td>
</tr>
<tr>
<td>6</td>
<td>root</td>
<td>RT</td>
<td>-5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>0.0</td>
<td>0.04</td>
<td>0:04</td>
<td>migration/1</td>
</tr>
<tr>
<td>7</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>0.03</td>
<td>0:03</td>
<td>ksoftirq/1</td>
</tr>
<tr>
<td>8</td>
<td>root</td>
<td>RT</td>
<td>-5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>0.0</td>
<td>0.00</td>
<td>0:00</td>
<td>watchdog/1</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>0.01</td>
<td>0:01</td>
<td>events/0</td>
</tr>
<tr>
<td>10</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</td>
<td>0:00</td>
<td>events/1</td>
</tr>
<tr>
<td>11</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</td>
<td>0:00</td>
<td>khelper</td>
</tr>
<tr>
<td>31</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>0.01</td>
<td>0:01</td>
<td>kblockd/0</td>
</tr>
<tr>
<td>32</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</td>
<td>0:00</td>
<td>kblockd/1</td>
</tr>
</tbody>
</table>

Top results after running benchmarks:

```
top - 12:14:56 up 92 days, 20:49, 3 users, load average: 1.02, 1.05, 0.95
Tasks: 176 total, 1 running, 175 sleeping, 0 stopped, 0 zombie
Cpu(s): 27.1%us, 3.0%sy, 0.8%ni, 68.5%id, 0.2%wa, 0.2%hi, 0.2%si, 0.0%st
Mem: 2073956k total, 1052964k used, 921040k free, 80168k buffers
Swap: 14996668k total, 347992k free, 14648776k used, 594528k 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>1</td>
<td>root</td>
<td>18</td>
<td>0</td>
<td>2948</td>
<td>1856</td>
<td>532</td>
<td>S</td>
<td>0.1</td>
<td>0.00</td>
<td>0:07.44</td>
<td>init</td>
</tr>
<tr>
<td>2</td>
<td>root</td>
<td>13</td>
<td>-5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>0.0</td>
<td>0.00</td>
<td>0:00</td>
<td>kthread</td>
</tr>
<tr>
<td>3</td>
<td>root</td>
<td>RT</td>
<td>-5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>0.0</td>
<td>0.04</td>
<td>0:04</td>
<td>migration/0</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>0.14</td>
<td>0:14</td>
<td>ksoftirq/0</td>
</tr>
<tr>
<td>5</td>
<td>root</td>
<td>RT</td>
<td>-5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>0.0</td>
<td>0.00</td>
<td>0:00</td>
<td>watchdog/0</td>
</tr>
<tr>
<td>6</td>
<td>root</td>
<td>RT</td>
<td>-5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>0.0</td>
<td>0.04</td>
<td>0:04</td>
<td>migration/1</td>
</tr>
<tr>
<td>7</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>0.03</td>
<td>0:03</td>
<td>ksoftirq/1</td>
</tr>
<tr>
<td>8</td>
<td>root</td>
<td>RT</td>
<td>-5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>0.0</td>
<td>0.00</td>
<td>0:00</td>
<td>watchdog/1</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>0.01</td>
<td>0:01</td>
<td>events/0</td>
</tr>
<tr>
<td>10</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</td>
<td>0:00</td>
<td>events/1</td>
</tr>
<tr>
<td>11</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</td>
<td>0:00</td>
<td>khelper</td>
</tr>
<tr>
<td>31</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>0.01</td>
<td>0:01</td>
<td>kblockd/0</td>
</tr>
<tr>
<td>32</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</td>
<td>0:00</td>
<td>kblockd/1</td>
</tr>
</tbody>
</table>