WaveScript Benchmarks Performance Report

August 30, 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: 3550

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 c2boehmsglst c2 c2seglst c2def c2defseglst
just_timer 2516.000 2512.000 2516.000 2516.000 2512.000 5016.000 5008.000
readfile_bigwins 4200.000 456.000 1044.000 740.000 3964.000 284.000 1020.000
printing_lists 2844.000 896.000 896.000 868.000 860.000 808.000 792.000
conv_SigsegArr 2256.000 408.000 7280.000 816.000 5472.000 76.000 6560.000
fft 100.000 936.000 908.000 884.000 988.000 856.000 836.000

Language Shootout:

## User time for each language-shootout benchmark/backend
Benchmark c2
fannkuch2 4448.000

Application Benchmarks:

Benchmark mltonO3 c2boehm c2boehmsglst c2 c2seglst c2def c2defseglst
## Running orig marmot phase i
## Running marmot2

### Running marmot3

### Running marmot multinode offline

---

### B Appendix: Additional system information

Top results before running benchmarks:

```
run_first_phase 7184.000 11889.000 4088.000 7340.000 5616.000 7116.000 4084.000
```

```
## Running marmot2
test_marmot2 2216.000 5240.000 5252.000 4648.000 4664.000 4588.000 4576.000
## Running marmot3
test_heatmap 7776.000 3248.000 3248.000 2564.000 2544.000 3236.000 3232.000
## Running marmot multinode offline
run_3phases 9725.000 5984.000 4816.000 5604.000 5312.000 5616.000 4856.000
```

```
B Appendix: Additional system information

Top results before running benchmarks:

top - 06:22:31 up 39 days, 14:56, 5 users, load average: 0.97, 1.00, 0.76
Tasks: 157 total, 1 running, 156 sleeping, 0 stopped, 0 zombie
Cpu(s): 24.8%us, 4.4%sy, 1.0%ni, 68.8%id, 0.1%wa, 0.4%hi, 0.5%si, 0.0%st
Mem: 2073956k total, 1031980k used, 1041976k free, 55596k buffers
Swap: 14996668k total, 34748k used, 14961920k free, 688288k 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 %CPU</th>
<th>%MEM</th>
<th>TIME+</th>
<th>COMMAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>19782</td>
<td>newton</td>
<td>21</td>
<td>0</td>
<td>2364</td>
<td>1080</td>
<td>784</td>
<td>R 0.1</td>
<td>0:00.01</td>
<td>top</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</td>
<td>S 0.6</td>
<td>6:47.94</td>
<td>emacs21-x</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>root</td>
<td>22</td>
<td>0</td>
<td>2948</td>
<td>1856</td>
<td>532</td>
<td>S 0.1</td>
<td>0:05.21</td>
<td>init</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>root</td>
<td>11</td>
<td>-5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S 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</td>
<td>S 0.0</td>
<td>0:00.38</td>
<td>migration/0</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>root</td>
<td>RT</td>
<td>19</td>
<td>0</td>
<td>0</td>
<td>S 0</td>
<td>0.00</td>
<td>0:00.70</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>S 0</td>
<td>0.00</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>S 0</td>
<td>0.00</td>
<td>0:00.31</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>S 0</td>
<td>0.00</td>
<td>0:01.17</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>S 0</td>
<td>0.00</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</td>
<td>S 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</td>
<td>S 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</td>
<td>S 0.0</td>
<td>0:00.02</td>
<td>khelper</td>
<td></td>
</tr>
</tbody>
</table>

Top results after running benchmarks:

```
run_first_phase 7184.000 11889.000 4088.000 7340.000 5616.000 7116.000 4084.000
```

```
## Running marmot2
test_marmot2 2216.000 5240.000 5252.000 4648.000 4664.000 4588.000 4576.000
## Running marmot3
test_heatmap 7776.000 3248.000 3248.000 2564.000 2544.000 3236.000 3232.000
## Running marmot multinode offline
run_3phases 9725.000 5984.000 4816.000 5604.000 5312.000 5616.000 4856.000
```

```
B Appendix: Additional system information

Top results after running benchmarks:

top - 06:40:33 up 39 days, 15:14, 5 users, load average: 1.00, 1.00, 0.91
Tasks: 157 total, 1 running, 156 sleeping, 0 stopped, 0 zombie
Cpu(s): 24.8%us, 4.4%sy, 1.0%ni, 68.8%id, 0.1%wa, 0.4%hi, 0.5%si, 0.0%st
Mem: 2073956k total, 1019316k used, 1054640k free, 58484k buffers
Swap: 14996668k total, 34748k used, 14961920k free, 670000k 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 %CPU</th>
<th>%MEM</th>
<th>TIME+</th>
<th>COMMAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>root</td>
<td>20</td>
<td>0</td>
<td>2948</td>
<td>1856</td>
<td>532</td>
<td>S 0.1</td>
<td>0:05.23</td>
<td>init</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>root</td>
<td>11</td>
<td>-5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S 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</td>
<td>S 0.0</td>
<td>0:00.38</td>
<td>migration/0</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>root</td>
<td>RT</td>
<td>19</td>
<td>0</td>
<td>0</td>
<td>S 0</td>
<td>0.00</td>
<td>0:00.70</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>S 0</td>
<td>0.00</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>S 0</td>
<td>0.00</td>
<td>0:00.31</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>S 0</td>
<td>0.00</td>
<td>0:01.17</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>S 0</td>
<td>0.00</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</td>
<td>S 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</td>
<td>S 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</td>
<td>S 0.0</td>
<td>0:00.02</td>
<td>khelper</td>
<td></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 0.0</td>
<td>0:00.46</td>
<td>kblockd/0</td>
<td></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 0.0</td>
<td>0:00.00</td>
<td>kblockd/1</td>
<td></td>
</tr>
</tbody>
</table>

---

6