WaveScript Benchmarks Performance Report

September 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: 3580

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.

![fannkuch2](image)

## 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 back-ends. 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
## Real or User time for each benchmark/backend
## LD_PRELOAD:
## NOSUDO:
## NICE:
Benchmark mltonO3 c2boehm c2boehmseglist c2 c2seglist c2def c2defseglist
just_timer 2504.000 2516.000 2520.000 2512.000 2524.000 5052.000 5044.000
readfile_bigwins 3764.000 464.000 1152.000 816.000 3908.000 292.000 1028.000
printing_lists 2680.000 916.000 920.000 876.000 844.000 816.000 820.000
conv_SigsegArr 2280.000 396.000 7468.000 816.000 5648.000 60.000 7412.000
fft 108.000 916.000 952.000 908.000 944.000 840.000 896.000
```

Language Shootout:

```bash
## Real or User time for each benchmark/backend
## LD_PRELOAD:
## NOSUDO:
## NICE:
Benchmark c2
fannkuch2 4448.000
```
Application Benchmarks:

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

Benchmark mltonO3 c2boehm c2boehmseglist c2 c2seglist c2def c2defseglist
## Running orig marmot phase 1
run_first_phase 7416.000 12217.000 4388.000 7796.000 5728.000 8425.000 3936.000
## Running marmot2
test_marmot2 2324.000 5248.000 5248.000 4708.000 4684.000 4560.000 4560.000
## Running marmot3
test_heatmap 7780.000 3216.000 3204.000 2528.000 2512.000 3228.000 3240.000
## Running marmot multinode offline
run_3phases 9505.000 6104.000 4940.000 5664.000 5332.000 5748.000 4792.000

B Appendix: Additional system information

Top results before running benchmarks:

top - 06:22:32 up 70 days, 14:56, 2 users, load average: 2.00, 2.00, 1.77
Tasks: 171 total, 2 running, 169 sleeping, 0 stopped, 0 zombie
Cpu(s): 32.0%us, 3.8%sy, 1.1%ni, 62.4%id, 0.2%wa, 0.3%hi, 0.3%si, 0.0%st
Mem: 2073956k total, 1566556k used, 507400k free, 138944k buffers
Swap: 14996668k total, 34748k used, 14961920k free, 934352k 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>%CPU</th>
<th>%MEM</th>
<th>TIME+</th>
<th>COMMAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>30398</td>
<td>newton</td>
<td>25</td>
<td>0</td>
<td>34212</td>
<td>31m</td>
<td>492 R</td>
<td>99</td>
<td>1.6</td>
<td>1420:58</td>
<td>ikarus</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>0.1</td>
<td>0:06.96</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 S</td>
<td>0.0</td>
<td>0:00:00</td>
<td>kthreadd</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>0.0</td>
<td>0:04:78</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>0.0</td>
<td>0:14:61</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>0.0</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>0.0</td>
<td>0:03:94</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>0.0</td>
<td>0:03:86</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>0.0</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>0.0</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>0.0</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>0.0</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 S</td>
<td>0.0</td>
<td>0:01:13</td>
<td>kblockd/0</td>
<td></td>
</tr>
</tbody>
</table>

Top results after running benchmarks:

top - 06:41:39 up 70 days, 15:15, 2 users, load average: 2.18, 2.11, 1.98
Tasks: 171 total, 3 running, 168 sleeping, 0 stopped, 0 zombie
Cpu(s): 32.0%us, 3.8%sy, 1.1%ni, 62.4%id, 0.2%wa, 0.3%hi, 0.3%si, 0.0%st
Mem: 2073956k total, 1042928k used, 1031028k free, 94952k buffers
Swap: 14996668k total, 34748k used, 14961920k free, 479552k 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>%CPU</th>
<th>%MEM</th>
<th>TIME+</th>
<th>COMMAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>30398</td>
<td>newton</td>
<td>25</td>
<td>0</td>
<td>34212</td>
<td>31m</td>
<td>492 R</td>
<td>101</td>
<td>1.6</td>
<td>1439:59</td>
<td>ikarus</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>0.1</td>
<td>0:06:96</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 S</td>
<td>0.0</td>
<td>0:00:00</td>
<td>kthreadd</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>0.0</td>
<td>0:04:79</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 R</td>
<td>0.0</td>
<td>0:14:61</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>0.0</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>0.0</td>
<td>0:03:94</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>0.0</td>
<td>0:03:86</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>0.0</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>0.0</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>0.0</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>0.0</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 S</td>
<td>0.0</td>
<td>0:01:13</td>
<td>kblockd/0</td>
<td></td>
</tr>
</tbody>
</table>