WaveScript Benchmarks Performance Report

September 16, 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: 3570

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 mltonO3 c2boehm c2boehmseglst c2 c2seglist c2def c2defseglst
just_timer 2572.000 2516.000 2516.000 2532.000 2540.000 5056.000 5040.000
readfile_bigwins 3800.000 536.000 1144.000 776.000 3932.000 264.000 1000.000
printing_lists 2708.000 896.000 936.000 824.000 844.000 812.000 824.000
conv_SigsegArr 2360.000 360.000 7288.000 796.000 5548.000 44.000 6672.000
fft 108.000 892.000 892.000 952.000 920.000 872.000 896.000

Language Shootout:

## Real or User time for each benchmark/backend
## LD_PRELOAD:
## NOSUDO:
## NICE:
Benchmark c2
fannkuch2 4452.000
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 7152.000 11833.000 4192.000 7392.000 5676.000 7392.000 3856.000

## Running marmot2
test_marmot2 2316.000 5236.000 5224.000 4716.000 4680.000 4556.000 4540.000

## Running marmot3
test_heatmap 7796.000 3236.000 3228.000 2544.000 2532.000 3224.000 3232.000

## Running marmot multinode offline
run_3phases 9489.000 5880.000 4860.000 5608.000 5340.000 5576.000 4812.000

B Appendix: Additional system information

Top results before running benchmarks:

top - 11:40:35 up 56 days, 20:14, 2 users, load average: 1.13, 1.28, 1.18
Tasks: 162 total, 2 running, 160 sleeping, 0 stopped, 0 zombie
Cpu(s): 29.7%us, 4.0%sy, 0.7%ni, 64.8%id, 0.1%wa, 0.3%hi, 0.4%si, 0.0%st
Mem: 2073956k total, 1667384k used, 406572k free, 106084k buffers
Swap: 14996668k total, 34744k used, 14961924k free, 1142252k 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>9140</td>
<td>newton</td>
<td>25</td>
<td>0</td>
<td>53576</td>
<td>47m</td>
<td>2148 R</td>
<td>99</td>
<td>2.3</td>
<td>0:02.68</td>
<td>mzscheme</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 S</td>
<td>0</td>
<td>0.1</td>
<td>0:05.82</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</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</td>
<td>0.0</td>
<td>0:00.46</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>0:13.07</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>0:00.38</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>0:03.42</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>18</td>
<td>-5</td>
<td>0</td>
<td>0</td>
<td>0 S</td>
<td>0</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</td>
<td>0.0</td>
<td>0:00.66</td>
<td>kblockd/0</td>
<td></td>
</tr>
</tbody>
</table>

Top results after running benchmarks:

top - 11:58:54 up 56 days, 20:33, 2 users, load average: 1.03, 1.06, 1.08
Tasks: 159 total, 1 running, 158 sleeping, 0 stopped, 0 zombie
Cpu(s): 29.7%us, 4.0%sy, 0.7%ni, 64.8%id, 0.1%wa, 0.3%hi, 0.4%si, 0.0%st
Mem: 2073956k total, 885360k used, 1187596k free, 63928k buffers
Swap: 14996668k total, 34744k used, 14961924k free, 483084k 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 S</td>
<td>0</td>
<td>0.1</td>
<td>0:05.83</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</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</td>
<td>0.0</td>
<td>0:00.46</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>0:13.07</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>0:00.38</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>0:03.42</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>18</td>
<td>-5</td>
<td>0</td>
<td>0</td>
<td>0 S</td>
<td>0</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</td>
<td>0.0</td>
<td>0:00.66</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 S</td>
<td>0</td>
<td>0.0</td>
<td>0:00.00</td>
<td>kblockd/1</td>
<td></td>
</tr>
</tbody>
</table>