WaveScript Benchmarks Performance Report

September 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: 3575

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 c2boehmseglist c2 c2seglist c2def c2defseglist
just_timer 2532.000 2512.000 2512.000 2516.000 2512.000 5024.000 5032.000
readfile_bigwins 3800.000 444.000 1100.000 760.000 3864.000 240.000 960.000
printing_lists 2808.000 876.000 892.000 864.000 896.000 816.000 844.000
conv_SigsegArr 2248.000 400.000 7348.000 816.000 5452.000 36.000 6660.000
fft 144.000 920.000 880.000 960.000 932.000 864.000 880.000

Language Shootout:

## Real or User time for each benchmark/backend
## LD_PRELOAD:
## NOSUDO:
## NICE:
Benchmark c2
fannkuch2 4500.000
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 7188.000 11809.000 4276.000 8677.000 5720.000 7264.000 3808.000

## Running marmot2
test_marmot2 2344.000 5232.000 5252.000 4688.000 4712.000 4560.000 4584.000

## Running marmot3
test_heatmap 7756.000 3240.000 3204.000 2520.000 2536.000 3244.000 3256.000

## Running marmot multinode offline
run_3phases 9493.000 5928.000 4848.000 5632.000 5284.000 5488.000 4768.000

### Appendix: Additional system information

Top results before running benchmarks:

top - 06:21:36 up 69 days, 14:55, 3 users, load average: 0.99, 1.00, 0.75
Tasks: 167 total, 1 running, 166 sleeping, 0 stopped, 0 zombie
Cpu(s): 31.8%us, 3.8%sy, 1.1%ni, 62.6%id, 0.2%wa, 0.3%hi, 0.3%si, 0.0%st
Mem: 2073956k total, 1497844k used, 576112k free, 157032k buffers
Swap: 14996668k total, 34744k used, 14961924k free, 902180k 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+ COMMAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>32643</td>
<td>newton</td>
<td>15</td>
<td>0</td>
<td>15836</td>
<td>11a</td>
<td>4200</td>
<td>S 2</td>
<td>0.6</td>
<td>172:54.59 emacs21-x</td>
</tr>
<tr>
<td>1</td>
<td>root</td>
<td>18</td>
<td>0</td>
<td>2948</td>
<td>1856</td>
<td>532</td>
<td>S 0.1</td>
<td>0:06:93 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 kthreadd</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>root</td>
<td>34</td>
<td>19</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S 0.0</td>
<td>0:04:78 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 0.0</td>
<td>0:14:61 ksoftirqd/0</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>root</td>
<td>34</td>
<td>19</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S 0.0</td>
<td>0:00:00 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 0.0</td>
<td>0:03:94 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</td>
<td>S 0.0</td>
<td>0:03:85 ksoftirqd/1</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>root</td>
<td>34</td>
<td>19</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S 0.0</td>
<td>0:00:00 watchdog/1</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>root</td>
<td>34</td>
<td>19</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S 0.0</td>
<td>0:00:00 events/0</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>root</td>
<td>34</td>
<td>19</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S 0.0</td>
<td>0:00:00 events/1</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>root</td>
<td>34</td>
<td>19</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S 0.0</td>
<td>0:00:00 events/1</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>root</td>
<td>34</td>
<td>19</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S 0.0</td>
<td>0:01:12 kblockd/0</td>
<td></td>
</tr>
</tbody>
</table>

Top results after running benchmarks:

top - 06:39:56 up 69 days, 15:14, 3 users, load average: 0.92, 0.99, 0.90
Tasks: 167 total, 1 running, 166 sleeping, 0 stopped, 0 zombie
Cpu(s): 31.8%us, 3.8%sy, 1.1%ni, 62.6%id, 0.2%wa, 0.3%hi, 0.3%si, 0.0%st
Mem: 2073956k total, 1051124k used, 1022832k free, 159976k buffers
Swap: 14996668k total, 34744k used, 14961924k free, 499444k 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+ COMMAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>30311</td>
<td>newton</td>
<td>21</td>
<td>0</td>
<td>2364</td>
<td>1084</td>
<td>784</td>
<td>R 2</td>
<td>0.1</td>
<td>0:00:01 top</td>
</tr>
<tr>
<td>1</td>
<td>root</td>
<td>18</td>
<td>0</td>
<td>2948</td>
<td>1856</td>
<td>532</td>
<td>S 0.1</td>
<td>0:06:95 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 kthreadd</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>root</td>
<td>34</td>
<td>19</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S 0.0</td>
<td>0:04:78 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 0.0</td>
<td>0:14:61 ksoftirqd/0</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>root</td>
<td>34</td>
<td>19</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S 0.0</td>
<td>0:00:00 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 0.0</td>
<td>0:03:94 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</td>
<td>S 0.0</td>
<td>0:03:86 ksoftirqd/1</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>root</td>
<td>34</td>
<td>19</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S 0.0</td>
<td>0:00:00 watchdog/1</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>root</td>
<td>34</td>
<td>19</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S 0.0</td>
<td>0:00:00 events/0</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>root</td>
<td>34</td>
<td>19</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S 0.0</td>
<td>0:00:04 events/1</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>root</td>
<td>34</td>
<td>19</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S 0.0</td>
<td>0:00:02 khelper</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>root</td>
<td>34</td>
<td>19</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S 0.0</td>
<td>0:01:12 kblockd/0</td>
<td></td>
</tr>
</tbody>
</table>