WaveScript Benchmarks Performance Report

November 10, 2007

Machine information:
Linux faith 2.6.18-4-k7 #1 SMP Wed May 9 23:42:01 UTC 2007 i686 GNU/Linux

WaveScript SVN:
Revision: 2890

WaveScope Engine SVN:
Revision: 1495

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.

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 "Scheme -O2" "Scheme -O3" "XStream -j 1 --at_once" "XStream DepthFirst -j 1 --at_once" "CoreFit DF -j 1" "MLton -O2" "MLton -O3"
readfile_bigwins 2280 1128 300.0 20.0 12.0 456.0 428.0
edge_stress 7124 3628 9757.0 1284.0 568.0 48.0 48.0
printing_lists 8572 6412 1496.0 1380.0 1488.0 1716.0 1680.0
conv_SigsegArr 25509 12 2156.0 1852.0 1904.0 11409.0 8.0
fft 4268 2736 380.0 348.0 364.0 952.0 1024.0

Language Shootout:

## User time for each language-shootout benchmark/backend
Benchmark "Scheme -O2" "Scheme -O3" "XStream -j 1 --at_once" "XStream DepthFirst -j 1 --at_once" "CoreFit DF -j 1" "MLton -O2" "MLton -O3"
fannkuch2 20654 15593 736.0 736.0 724.0 968.0 712.0

Marmot Application:

## Running original marmot app.

B Appendix: Additional system information

Top results before running benchmarks:

top - 06:56:00 up 153 days, 17:28, 10 users, load average: 1.04, 1.01, 1.00
Tasks: 453 total, 1 running, 452 sleeping, 0 stopped, 0 zombie
Cpu(s): 14.2%us, 1.7%sy, 3.1%ni, 79.4%id, 1.7%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 2076424k total, 451476k used, 1624948k free, 29568k buffers
Swap: 1951856k total, 192856k used, 1759000k free, 221736k cached

PID USER PR NI VIRT RES SHR %CPU %MEM TIME+ COMMAND
10249 newton 19 0 2624 1272 720 4 0.1 0:00.02 top
 1 root 15 0 2072 100 72 S 0 0.0 2:00.18 init
 2 root RT 0 0 0 0 S 0 0.0 0:02.76 migration/0
 3 root 34 19 0 0 0 S 0 0.0 0:37.21 ksoftirqd/0
 4 root RT 0 0 0 0 S 0 0.0 0:03.33 migration/1
 5 root 39 19 0 0 0 S 0 0.0 1:47.24 ksoftirqd/1
 6 root 10 -5 0 0 0 S 0 0.0 0:00.16 events/0
 7 root 10 -5 0 0 0 S 0 0.0 0:00.11 events/1
 8 root 11 -5 0 0 0 S 0 0.0 0:00.00 khelper
Top results after running benchmarks:

```
top - 06:56:01 up 103 days, 17:28, 10 users, load average: 1.04, 1.01, 1.00
Tasks: 453 total, 1 running, 452 sleeping, 0 stopped, 0 zombie
Cpu(s): 14.2%us, 1.7%sy, 3.1%ni, 79.4%id, 1.7%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 2076424k total, 451608k used, 1624816k free, 29580k buffers
Swap: 1951856k total, 192856k used, 1759000k free, 221724k 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>15</td>
<td>0</td>
<td>2072</td>
<td>100</td>
<td>72</td>
<td>S</td>
<td>0</td>
<td>0.0</td>
<td>2:00.18</td>
<td>init</td>
</tr>
<tr>
<td>2</td>
<td>root</td>
<td>RT</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>0</td>
<td>0.0</td>
<td>0:02.76</td>
<td>migration/0</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</td>
<td>0</td>
<td>0.0</td>
<td>0:37.21</td>
<td>ksoftirqd/0</td>
</tr>
<tr>
<td>4</td>
<td>root</td>
<td>RT</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>0</td>
<td>0.0</td>
<td>0:03.33</td>
<td>migration/1</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</td>
<td>0</td>
<td>0.0</td>
<td>1:47.24</td>
<td>ksoftirqd/1</td>
</tr>
<tr>
<td>6</td>
<td>root</td>
<td>10</td>
<td>-5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>0</td>
<td>0.0</td>
<td>0:00.16</td>
<td>events/0</td>
</tr>
<tr>
<td>7</td>
<td>root</td>
<td>10</td>
<td>-5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>0</td>
<td>0.0</td>
<td>0:00.11</td>
<td>events/1</td>
</tr>
<tr>
<td>8</td>
<td>root</td>
<td>11</td>
<td>-5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>0</td>
<td>0.0</td>
<td>0:00.00</td>
<td>khelper</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</td>
<td>0.0</td>
<td>0:00.01</td>
<td>kthread</td>
</tr>
<tr>
<td>13</td>
<td>root</td>
<td>10</td>
<td>-5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>0</td>
<td>0.0</td>
<td>0:00.67</td>
<td>kblockd/0</td>
</tr>
<tr>
<td>14</td>
<td>root</td>
<td>10</td>
<td>-5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>0</td>
<td>0.0</td>
<td>0:01.07</td>
<td>kblockd/1</td>
</tr>
<tr>
<td>15</td>
<td>root</td>
<td>15</td>
<td>-5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>0</td>
<td>0.0</td>
<td>0:00.00</td>
<td>kacpid</td>
</tr>
<tr>
<td>82</td>
<td>root</td>
<td>12</td>
<td>-5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>0</td>
<td>0.0</td>
<td>0:00.00</td>
<td>kseriod</td>
</tr>
</tbody>
</table>
```