WaveScript Benchmarks Performance Report

August 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: 3548

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.

![Graph](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

## User time for each benchmark/backend
Benchmark mltonO3 c2boehm c2boehmseglist c2 c2seglist c2def c2defseglist
   just_timer 2512.000 2556.000 2528.000 2500.000 2544.000 4976.000 5048.000
   readme_bigwins 3916.000 452.000 1076.000 740.000 3856.000 276.000 1040.000
   printing_lists 2664.000 904.000 888.000 868.000 828.000 800.000 808.000
   conv_SigsegArr 2292.000 376.000 7384.000 796.000 5484.000 40.000 6640.000
   fft 100.000 920.000 952.000 944.000 972.000 864.000 872.000

Language Shootout:

## User time for each language-shootout benchmark/backend
Benchmark c2
   fannkuch2 4460.000

Application Benchmarks:

Benchmark mltonO3 c2boehm c2boehmseglist c2 c2seglist c2def c2defseglist
   # Running orig marmot phase 1
# Running marmot2

test_marmot2 2236.000 5324.000 5316.000 4636.000 4652.000 4576.000 4520.000

# Running marmot3
test_heatmap 7760.000 3224.000 3268.000 2560.000 2556.000 3224.000 3272.000

# Running marmot multinode offline
run_3phases 9925.000 6260.000 4872.000 5648.000 5376.000 5596.000 4864.000

## B Appendix: Additional system information

Top results before running benchmarks:

top - 16:04:09 up 39 days, 38 min, 6 users, load average: 2.18, 2.32, 2.40
Tasks: 177 total, 3 running, 174 sleeping, 0 stopped, 0 zombie
Cpu(s): 25.0%us, 4.4%sy, 1.0%ni, 68.5%id, 0.1%wa, 0.4%hi, 0.5%si, 0.0%st
Mem: 2073956k total, 1084020k used, 989936k free, 54236k buffers
Swap: 14996668k total, 34752k used, 14961916k free, 641388k 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>22102</td>
<td>newton</td>
<td>25</td>
<td>0</td>
<td>2948</td>
<td>1856</td>
<td>532</td>
<td>S</td>
<td>0.1</td>
<td>0:05.14 init</td>
</tr>
<tr>
<td>25064</td>
<td>newton</td>
<td>15</td>
<td>0</td>
<td>4048</td>
<td>1328</td>
<td>404</td>
<td>S</td>
<td>0.0</td>
<td>763:02.48 unison-2.18</td>
</tr>
</tbody>
</table>

Top results after running benchmarks:

top - 16:26:20 up 39 days, 1:00, 6 users, load average: 2.44, 2.90, 2.62
Tasks: 178 total, 2 running, 176 sleeping, 0 stopped, 0 zombie
Cpu(s): 25.1%us, 4.4%sy, 1.0%ni, 68.5%id, 0.1%wa, 0.4%hi, 0.5%si, 0.0%st
Mem: 2073956k total, 833164k used, 1240792k free, 11736k buffers
Swap: 14996668k total, 34752k used, 14961912k free, 394708k 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>4537</td>
<td>newton</td>
<td>25</td>
<td>0</td>
<td>65468</td>
<td>60m</td>
<td>2064</td>
<td>R</td>
<td>99</td>
<td>0:09.99 mzschem</td>
</tr>
<tr>
<td>32643</td>
<td>newton</td>
<td>15</td>
<td>0</td>
<td>15836</td>
<td>11a</td>
<td>4200</td>
<td>S</td>
<td>3.0</td>
<td>18:40 emacs21-x</td>
</tr>
</tbody>
</table>

Top results after running benchmarks:

top - 16:26:20 up 39 days, 1:00, 6 users, load average: 2.44, 2.90, 2.62
Tasks: 178 total, 2 running, 176 sleeping, 0 stopped, 0 zombie
Cpu(s): 25.1%us, 4.4%sy, 1.0%ni, 68.5%id, 0.1%wa, 0.4%hi, 0.5%si, 0.0%st
Mem: 2073956k total, 833164k used, 1240792k free, 11736k buffers
Swap: 14996668k total, 34752k used, 14961912k free, 394708k 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>4537</td>
<td>newton</td>
<td>25</td>
<td>0</td>
<td>65468</td>
<td>60m</td>
<td>2064</td>
<td>R</td>
<td>99</td>
<td>0:09.99 mzschem</td>
</tr>
<tr>
<td>32643</td>
<td>newton</td>
<td>15</td>
<td>0</td>
<td>15836</td>
<td>11a</td>
<td>4200</td>
<td>S</td>
<td>3.0</td>
<td>18:40 emacs21-x</td>
</tr>
</tbody>
</table>

Top results after running benchmarks:

top - 16:26:20 up 39 days, 1:00, 6 users, load average: 2.44, 2.90, 2.62
Tasks: 178 total, 2 running, 176 sleeping, 0 stopped, 0 zombie
Cpu(s): 25.1%us, 4.4%sy, 1.0%ni, 68.5%id, 0.1%wa, 0.4%hi, 0.5%si, 0.0%st
Mem: 2073956k total, 833164k used, 1240792k free, 11736k buffers
Swap: 14996668k total, 34752k used, 14961912k free, 394708k 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>4537</td>
<td>newton</td>
<td>25</td>
<td>0</td>
<td>65468</td>
<td>60m</td>
<td>2064</td>
<td>R</td>
<td>99</td>
<td>0:09.99 mzschem</td>
</tr>
<tr>
<td>32643</td>
<td>newton</td>
<td>15</td>
<td>0</td>
<td>15836</td>
<td>11a</td>
<td>4200</td>
<td>S</td>
<td>3.0</td>
<td>18:40 emacs21-x</td>
</tr>
</tbody>
</table>

Top results after running benchmarks:

top - 16:26:20 up 39 days, 1:00, 6 users, load average: 2.44, 2.90, 2.62
Tasks: 178 total, 2 running, 176 sleeping, 0 stopped, 0 zombie
Cpu(s): 25.1%us, 4.4%sy, 1.0%ni, 68.5%id, 0.1%wa, 0.4%hi, 0.5%si, 0.0%st
Mem: 2073956k total, 833164k used, 1240792k free, 11736k buffers
Swap: 14996668k total, 34752k used, 14961912k free, 394708k 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>4537</td>
<td>newton</td>
<td>25</td>
<td>0</td>
<td>65468</td>
<td>60m</td>
<td>2064</td>
<td>R</td>
<td>99</td>
<td>0:09.99 mzschem</td>
</tr>
<tr>
<td>32643</td>
<td>newton</td>
<td>15</td>
<td>0</td>
<td>15836</td>
<td>11a</td>
<td>4200</td>
<td>S</td>
<td>3.0</td>
<td>18:40 emacs21-x</td>
</tr>
</tbody>
</table>