WaveScript Benchmarks Performance Report

October 17, 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: 3598

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

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>mltonO3</th>
<th>c2boehm</th>
<th>c2boehmseglist</th>
<th>c2</th>
<th>c2seglist</th>
<th>c2def</th>
<th>c2defseglist</th>
</tr>
</thead>
<tbody>
<tr>
<td>just_timer</td>
<td>2516.0</td>
<td>2520.0</td>
<td>2540.0</td>
<td>2512.0</td>
<td>2500.0</td>
<td>5036.0</td>
<td>5036.0</td>
</tr>
<tr>
<td>readfile_bigwins</td>
<td>3812.0</td>
<td>436.0</td>
<td>1140.0</td>
<td>1332.0</td>
<td>3748.0</td>
<td>276.0</td>
<td>984.0</td>
</tr>
<tr>
<td>printing_lists</td>
<td>2644.0</td>
<td>896.0</td>
<td>904.0</td>
<td>888.0</td>
<td>840.0</td>
<td>820.0</td>
<td>816.0</td>
</tr>
<tr>
<td>conv_SigsegArr</td>
<td>2256.0</td>
<td>408.0</td>
<td>7576.0</td>
<td>868.0</td>
<td>5652.0</td>
<td>40.0</td>
<td>6420.0</td>
</tr>
<tr>
<td>fft</td>
<td>108.0</td>
<td>940.0</td>
<td>944.0</td>
<td>946.0</td>
<td>944.0</td>
<td>960.0</td>
<td>876.0</td>
</tr>
</tbody>
</table>

Language Shootout:

## Real or User time for each benchmark/backend

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>c2</th>
</tr>
</thead>
<tbody>
<tr>
<td>fannkuch2</td>
<td>4332.0</td>
</tr>
</tbody>
</table>
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 7328.000 12285.000 4140.000 7528.000 5704.000 7416.000 3780.000
## Running marmot2
test_marmot2 2316.000 5224.000 5220.000 4628.000 4652.000 4536.000 4580.000
## Running marmot3
test_heatmap 7744.000 3220.000 3240.000 2532.000 2540.000 3248.000 3248.000
## Running marmot3 multinode offline
run_3phases 9497.000 5892.000 5220.000 5636.000 5272.000 5480.000 4852.000

B Appendix: Additional system information

Top results before running benchmarks:
top - 11:59:43 up 87 days, 20:33, 5 users, load average: 2.18, 2.21, 1.79
Tasks: 186 total, 2 running, 184 sleeping, 0 stopped, 0 zombie
Cpu(s): 28.5%us, 3.2%sy, 0.9%ni, 66.8%id, 0.2%wa, 0.2%hi, 0.3%si, 0.0%st
Mem: 2073956k total, 891164k used, 1182792k free, 82840k buffers
Swap: 14996668k total, 34752k used, 14961916k free, 494400k cached

### COMMAND
PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
27826 newton 25 0 2252 804 548 R 99 0.0 0:02.05 query.exe
1 root 18 0 2948 1856 532 S 0 0.1 0:07.21 init
2 root 13 -5 0 0 0 S 0 0.0 0:00.00 kthreadd
3 root RT -5 0 0 0 S 0 0.0 0:04.85 migration/0
4 root 34 19 0 0 0 S 0 0.0 0:14.69 ksoftirqd/0
5 root RT -5 0 0 0 S 0 0.0 0:00.00 watchdog/0
6 root RT -5 0 0 0 S 0 0.0 0:04.01 migration/1
7 root 34 19 0 0 0 S 0 0.0 0:03.94 ksoftirqd/1
8 root RT -5 0 0 0 S 0 0.0 0:00.00 watchdog/1
9 root 10 -5 0 0 0 S 0 0.0 0:01.78 events/0
10 root 10 -5 0 0 0 S 0 0.0 0:00.04 events/1
11 root 14 -5 0 0 0 S 0 0.0 0:00.02 khelper
31 root 10 -5 0 0 0 S 0 0.0 0:01.46 kblockd/0

Top results after running benchmarks:
top - 12:19:01 up 87 days, 20:53, 5 users, load average: 2.01, 2.08, 2.00
Tasks: 180 total, 1 running, 179 sleeping, 0 stopped, 0 zombie
Cpu(s): 28.5%us, 3.2%sy, 0.9%ni, 66.8%id, 0.2%wa, 0.2%hi, 0.3%si, 0.0%st
Mem: 2073956k total, 859948k used, 1214008k free, 86956k buffers
Swap: 14996668k total, 34752k used, 14961916k free, 481076k cached

### COMMAND
PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
1 root 18 0 2948 1856 532 S 0 0.1 0:07.22 init
2 root 13 -5 0 0 0 S 0 0.0 0:00.00 kthreadd
3 root RT -5 0 0 0 S 0 0.0 0:04.85 migration/0
4 root 34 19 0 0 0 S 0 0.0 0:14.69 ksoftirqd/0
5 root RT -5 0 0 0 S 0 0.0 0:00.00 watchdog/0
6 root RT -5 0 0 0 S 0 0.0 0:04.01 migration/1
7 root 34 19 0 0 0 S 0 0.0 0:03.94 ksoftirqd/1
8 root RT -5 0 0 0 S 0 0.0 0:00.00 watchdog/1
9 root 10 -5 0 0 0 S 0 0.0 0:01.78 events/0
10 root 10 -5 0 0 0 S 0 0.0 0:00.04 events/1
11 root 14 -5 0 0 0 S 0 0.0 0:00.02 khelper
31 root 10 -5 0 0 0 S 0 0.0 0:01.46 kblockd/0
32 root 10 -5 0 0 0 S 0 0.0 0:00.03 kblockd/1