WaveScript Benchmarks Perfromance Report

January 2, 2008

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: 2976

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 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 "Scheme -O2" "Scheme -O3" "XStream -j 1 --at_once" "XStream DepthFirst -j 1 --at_once" "CoreFit DF -j 1" "CoreFitDF 1Thread -j 1" "MLton -O2" "MLton -O3"
readfile_bigwins 2484 1236 280.0 8.0 12.0 8.0 536.0 444.0
printing_lists 12320 1780 696.0 656.0 660.0 608.0 812.0 1060.0
conv_SigsegArr 27682 64 1012.0 968.0 868.0 864.0 8.0 4.0
fft 4636 2624 360.0 336.0 312.0 360.0 36.0 36.0

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" "CoreFitDF 1Thread -j 1" "MLton -O2" "MLton -O3"
fannkuch2 15889 12581 752.0 748.0 728.0 724.0 980.0 720.0

Application Benchmarks:

Benchmark "Scheme -O2" "Scheme -O3" "XStream -j 1 --at_once" "XStream DepthFirst -j 1 --at_once" "CoreFit DF -j 1" "CoreFitDF 1Thread -j 1" "MLton -O2" "MLton -O3"

## Running orig marmot phase 1
run_first_phase 29234 11061 4011.9999999999995 1768.0 1796.0 1548.0 420.0 344.0
## Running marmot2
test_marmot2 13509 7664 728.0 712.0 768.0 772.0 492.0 484.0
## Running marmot3
test_heatmap 12844 7692 3144.0 3188.0 3068.0 2840.0 2332.0 2400.0

B Appendix: Additional system information

Top results before running benchmarks:
top - 08:37:39 up 156 days, 19:10, 8 users, load average: 3.08, 2.84, 2.95
Tasks: 178 total, 3 running, 168 sleeping, 6 stopped, 1 zombie
Cpu(s): 14.0%us, 1.7%sy, 2.0%ni, 80.8%id, 1.5%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 2076424k total, 1459060k used, 617364k free, 52380k buffers
Swap: 1951856k total, 47316k used, 1904540k free, 899240k cached

PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
16909 newton 22 0 9532 8176 556 R 100 0.4 40:43.15 larceny.bin
24178 newton 25 0 50700 46m 2120 R 98 2.3 0:02.07 mzscheme
1 root 15 0 2092 88 60 S 0.0 2:07.90 init
2 root RT 0 0 0 0 S 0.0 0:05.04 migration/0
3 root  34   19   0   0   0   S   0   0.0  0:40.10  ksoftirqd/0
4 root  RT   0   0   0   0   S   0   0.0  0:05.96  migration/1
5 root  34   19   0   0   0   S   0   1:59.59  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  0:00.12  events/1
8 root  14  -5   0   0   0   S   0   0.0  0:00.00  khelper
9 root  16  -5   0   0   0   S   0   0.0  0:00.02  kthread
13 root 10  -5   0   0   0   S   0   0.0  0:00.94  kblockd/0
14 root 10  -5   0   0   0   S   0   0.0  0:01.44  kblockd/1

Top results after running benchmarks:

```
top - 09:21:36 up 156 days, 19:54, 8 users, load average: 3.01, 3.06, 3.07
Tasks: 183 total, 3 running, 173 sleeping, 6 stopped, 1 zombie
Cpu(s): 14.0%us, 1.7%sy, 2.0%ni, 80.8%id, 1.5%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 2076424k total, 1406748k used, 669676k free, 11628k buffers
Swap: 1951856k total, 47316k used, 1904540k free, 936704k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM TIME+ COMMAND
16909 newton  25   0  9532 8176 556 R  99   0.4 71:23.20 larceny.bin
  939 newton  25   0  42032 39m  768 R  99  1.9  0:02.39 scheme
    1 root    RT   0   0   0   0   S  99  0.0  2:07.90 init
    2 root  34   19   0   0   0   S  99  0.0  0:05.04 migration/0
    3 root  34   19   0   0   0   S  99  0.0  0:00.40 ksoftirqd/0
    4 root  RT   0   0   0   0   S  99  0.0  0:05.97 migration/1
    5 root  34   19   0   0   0   S  99  0.0  1:59.59 ksoftirqd/1
    6 root  10  -5   0   0   0   S  99  0.0  0:00.16 events/0
    7 root  10  -5   0   0   0   S  99  0.0  0:00.12 events/1
    8 root  14  -5   0   0   0   S  99  0.0  0:00.00 khelper
    9 root  16  -5   0   0   0   S  99  0.0  0:00.02 kthread
   13 root 10  -5   0   0   0   S  99  0.0  0:00.94 kblockd/0
   14 root 10  -5   0   0   0   S  99  0.0  0:01.44 kblockd/1
```