WaveScript Benchmarks Performance Report

November 13, 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: 2905

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" "MLton -O2" "MLton -O3"

readfile_bigwins 2424 1180 292.0 8.0 8.0 496.0 420.0
edge_stress 7760 3620 10545.0 1168.0 776.0 52.0 52.0
printing_lists 8908 6624 1476.0 1388.0 1456.0 1696.0 1716.0
conv_SigsegArr 26949 12 2060.0 1940.0 1952.0 0.0 4.0
fft 4836 2748 404.0 384.0 412.0 1008.0 1036.0

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 24010 16629 752.0 756.0 740.0 972.0 716.0

Marmot Application:

## Running original marmot app.

B Appendix: Additional system information

Top results before running benchmarks:

```
top - 21:15:26 up 107 days, 7:48, 14 users, load average: 1.71, 1.27, 0.89
Tasks: 480 total, 1 running, 476 sleeping, 3 stopped, 0 zombie
Cpu(s): 13.9%us, 1.6%sy, 3.0%ni, 79.8%id, 1.7%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 2076424k total, 787040k used, 1289384k free, 21756k buffers
Swap: 1951856k total, 111620k used, 1840236k free, 234472k cached

PID USER PR NI VIRT RES SHR %CPU %MEM TIME+ COMMAND
23707 newton 16 0 9160 7752 1108 S 45.0 4 5:47.73 unison-2.28
23356 newton 20 0 2632 1296 792 R 2 0.1 0:00.03 top
1 root 15 0 2072 100 72 S 0 0.0 2:00.24 init
2 root RT 0 0 0 0 S 0 0.0 0:02.87 migration/0
3 root 34 19 0 0 0 S 0 0.0 0:38.06 ksoftirqd/0
4 root RT 0 0 0 0 S 0 0.0 0:03.50 migration/1
5 root 34 19 0 0 0 S 0 0.0 0:153.48 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
```
Top results after running benchmarks:

```
top - 21:15:27 up 107 days, 7:48, 14 users, load average: 1.71, 1.27, 0.89
Tasks: 480 total, 1 running, 476 sleeping, 3 stopped, 0 zombie
Cpu(s): 13.9%us, 1.6%sy, 3.0%ni, 79.8%id, 1.7%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 2076424k total, 787164k used, 1289260k free, 21804k buffers
Swap: 1951856k total, 111620k used, 1840236k free, 234424k cached
```

```
PID USER  PR NI VIRT  RES  SHR S %CPU %MEM TIME+ COMMAND
2386 newton 18  0 2628 1296  72  R  2  0.1 0:00.02 top
  1 root  20  0 2072  72  32  S  0  0.0 0:00.24 init
  2 root  40  0  0   0   0 S  0  0.0 0:02.87 migration/0
  3 root  34 19  0   0   0 S  0  0.0 0:38.06 ksoftirqd/0
  4 root  40  0  0   0   0 S  0  0.0 0:03.50 migration/1
  5 root  34 19  0   0   0 S  0  0.0 1:53.48 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
  9 root  12  5  0   0   0 S  0  0.0 0:00.01 kthread
 13 root  10  5  0   0   0 S  0  0.0 0:00.67 kblockd/0
 14 root  14  5  0   0   0 S  0  0.0 0:01.09 kblockd/1
 15 root  15  5  0   0   0 S  0  0.0 0:00.00 kacpid
```