WaveScript Benchmarks Performance Report

October 7, 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: 3589

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:
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

```shell
## Real or User time for each benchmark/backend
## LD_PRELOAD:
## NOSUDO:
## NICE:
Benchmark mlton03 c2boehm c2boehmseglist c2 c2seglist c2def c2defseglist
just_timer 2516.000 2508.000 2552.000 2540.000 2512.000 5044.000 5044.000
readfile_bigwins 3824.000 452.000 1080.000 732.000 3896.000 272.000 968.000
printing_lists 2348.000 884.000 900.000 892.000 852.000 864.000 816.000
conv_SigssegArr 2360.000 388.000 7244.000 824.000 5628.000 44.000 6676.000
fft 140.000 896.000 856.000 876.000 876.000 892.000 840.000
```

Language Shootout:

```shell
## Real or User time for each benchmark/backend
## LD_PRELOAD:
## NOSUDO:
## NICE:
Benchmark c2
fannkuch2 4376.000
```
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 7124.000 11837.000 4136.000 7340.000 5688.000 7164.000 3808.000

### Running marmot2

test_marmot2 2332.000 5256.000 5240.000 4624.000 4708.000 4604.000 4592.000

### Running marmot3

test_heatmap 7776.000 3236.000 3216.000 2552.000 2524.000 3224.000 3236.000

### Running marmot multinode offline

rn_3phases 9477.000 5900.000 4844.000 5580.000 5316.000 5568.000 4800.000

### B Appendix: Additional system information

Top results before running benchmarks:

top - 10:01:32 up 77 days, 18:35, 7 users, load average: 0.97, 1.04, 0.82
Tasks: 178 total, 1 running, 177 sleeping, 0 stopped, 0 zombie
Cpu(s): 32.1%us, 3.5%sy, 1.0%ni, 62.7%id, 0.2%wa, 0.2%hi, 0.3%si, 0.0%st
Mem: 2073956k total, 1615792k used, 458164k free, 128792k buffers
Swap: 14996668k total, 34744k used, 14961924k free, 1082644k cached

| PID | USER  | PR | NI | VIRT | RES | SHR | S %CPU %MEM TIME+ COMMAND         |
|-----|-------|----|----|------|-----|-----|-----|-------------|----------------------------------|
| 1   | root  | 20 | 0  | 2948 | 1856| 532 | S   | 0.1         | 0:07.09 init init                 |
| 2   | root  | 10 | -5 | 0    | 0   | 0   | S   | 0.0         | 0:00.00 kthreadd kthreadd        |
| 3   | root  | RT | -5 | 0    | 0   | 0   | S   | 0.0         | 0:04.82 migration/0 migration/0  |
| 4   | root  | 34 | 19 | 0    | 0   | 0   | S   | 0.0         | 0:14.62 ksoftirqd/0 ksoftirqd/0  |
| 5   | root  | RT | -5 | 0    | 0   | 0   | S   | 0.0         | 0:00.00 watchdog/0 watchdog/0    |
| 6   | root  | RT | -5 | 0    | 0   | 0   | S   | 0.0         | 0:03.96 migration/1 migration/1  |
| 7   | root  | 34 | 19 | 0    | 0   | 0   | S   | 0.0         | 0:03.88 ksoftirqd/1 ksoftirqd/1  |
| 8   | root  | RT | -5 | 0    | 0   | 0   | S   | 0.0         | 0:00.00 watchdog/1 watchdog/1    |
| 9   | root  | 10 | -5 | 0    | 0   | 0   | S   | 0.0         | 0:00.30 events/0 events/0        |
| 10  | root  | 10 | -5 | 0    | 0   | 0   | S   | 0.0         | 0:00.04 events/1 events/1        |
| 11  | root  | 10 | -5 | 0    | 0   | 0   | S   | 0.0         | 0:00.02 khelper khelper          |
| 31  | root  | 10 | -5 | 0    | 0   | 0   | S   | 0.0         | 0:01.26 kblockd/0 kblockd/0      |
| 32  | root  | 10 | -5 | 0    | 0   | 0   | S   | 0.0         | 0:00.03 kblockd/1 kblockd/1      |

Top results after running benchmarks:

top - 10:20:02 up 77 days, 18:54, 7 users, load average: 0.92, 1.04, 1.00
Tasks: 178 total, 1 running, 177 sleeping, 0 stopped, 0 zombie
Cpu(s): 32.1%us, 3.5%sy, 1.0%ni, 62.7%id, 0.2%wa, 0.2%hi, 0.3%si, 0.0%st
Mem: 2073956k total, 1052116k used, 1021840k free, 131032k buffers
Swap: 14996668k total, 34744k used, 14961924k free, 561312k cached

| PID | USER  | PR | NI | VIRT | RES | SHR | S %CPU %MEM TIME+ COMMAND         |
|-----|-------|----|----|------|-----|-----|-----|-------------|---------------------------------|
| 18726 | newton | 21 | 0  | 2948 | 1856| 532 | R   | 2.1          | 0:00.01 top top                 |
| 1   | root  | 18 | 0  | 2948 | 1856| 532 | S   | 0.1         | 0:07.09 init kthreadd           |
| 2   | root  | 10 | -5 | 0    | 0   | 0   | S   | 0.0         | 0:00.00 kthreadd kthreadd      |
| 3   | root  | RT | -5 | 0    | 0   | 0   | S   | 0.0         | 0:04.82 migration/0 migration/0 |
| 4   | root  | 34 | 19 | 0    | 0   | 0   | S   | 0.0         | 0:14.62 ksoftirqd/0 ksoftirqd/0|
| 5   | root  | RT | -5 | 0    | 0   | 0   | S   | 0.0         | 0:00.00 watchdog/0 watchdog/0  |
| 6   | root  | RT | -5 | 0    | 0   | 0   | S   | 0.0         | 0:03.96 migration/1 migration/1 |
| 7   | root  | 34 | 19 | 0    | 0   | 0   | S   | 0.0         | 0:03.88 ksoftirqd/1 ksoftirqd/1|
| 8   | root  | RT | -5 | 0    | 0   | 0   | S   | 0.0         | 0:00.00 watchdog/1 watchdog/1  |
| 9   | root  | 10 | -5 | 0    | 0   | 0   | S   | 0.0         | 0:00.42 events/0 events/0      |
| 10  | root  | 10 | -5 | 0    | 0   | 0   | S   | 0.0         | 0:00.04 events/1 events/1      |
| 11  | root  | 10 | -5 | 0    | 0   | 0   | S   | 0.0         | 0:00.02 khelper khelper        |
| 31  | root  | 10 | -5 | 0    | 0   | 0   | S   | 0.0         | 0:01.26 kblockd/0 kblockd/0    |