Chapter 1

Benchmarking with SMark

It is a common understanding that when we write application we can gain around 30% to 50% once we profiled it. Pharo offers several tools to profile execution and they are presented in the Deep into Pharo. Now Smark is another interesting approach to build benchmarks. SMark is benchmarking framework for Smalltalk developed By Stefan Marr. It inspired by unit testing in SUnit and following this idea, a benchmark is implemented by adding a method name benchMyBenchmark to a subclass of SBenchmarkSuite. This enables performance regression tracking in the same way as unit-testing allows one to track functional regressions.

1.1 Getting Started

The code is originally based on PBenchmark the benchmark framework used for the PinocchioVM and RoarBenchmark a framework used for performance regression testing of the RoarVM.

1.2 SMarkSuite

A benchmark suite is a set of benchmarks. Such suite knows what exactly needs to be executed. However, it does not really know how to execute it. It knows how to set up and tear down the environment for the benchmarks, but does not have the knowledge of how many iterations need to be done and how to evaluate any results that might be produced. The abstract class representing suites is SMarkSuite. Let us start to see how to execute existing bench suites.

Runnning a bench suite using run and run:

SMarkLoops is a subclass of SMarkSuite. It defines one suite of benchmarks for evaluating loops performance. This suite is simple since it does not define resources or environments set up and tear down behavior. However as the following execution shows it, the suite is composed of multiple benchmarks.

```
SMarkLoops run: 10

Report for: SMarkLoops
Benchmark ClassVarBinding
ClassVarBinding total: iterations=10 runtime: 0.30ms +/−0.54

Benchmark FloatLoop
FloatLoop total: iterations=10 runtime: 1.40ms +/−0.72

Benchmark Send
Send total: iterations=10 runtime: 0.00ms +/−0.00

Benchmark InstVarAccess
InstVarAccess total: iterations=10 runtime: 0.20ms +/−0.36

Benchmark SendWithManyArguments
SendWithManyArguments total: iterations=10 runtime: 0.20ms +/−0.36

Benchmark IntLoop
IntLoop total: iterations=10 runtime: 0.00ms +/−0.00

Benchmark ArrayAccess
ArrayAccess total: iterations=10 runtime: 1.10ms +/−0.69
```

run executes the benchmarks once while run: anInteger performs a given number of iterations.

Defining a benchmark

Defining a benchmark is simple, you just need to define methods with a selector starting with the suffix 'bench'. For example SMarkLoops defines the method benchArrayAccess as follows:

```
SMarkLoops>>benchArrayAccess
   | i obj array |

   array := Array new: 1.
   obj   := Object new.

   i := self problemSize.
   [i > 0] whileTrue: [
```

```
    array at: 1.
    array at: 1 put: obj.

    i := i − 1.
].
```

1.3 Profiling a specific benchmark

Using the message profile: aSelector you can invoke a time profiler on the given benchmark.

Book Todo: add screenshot

You can also specify the number of iteration using the message profile: aSelector iterations: anInteger.

1.4 How to select a benchmark

1.5 What is a Runner

1.6 Conclusion

This is just the start of the chapter.