USFJProf
Open-Source Java Profiling Tool

James Chen & Wei Chen
Advisor: Terence Parr

October 8, 2004

1

Introduction

- What does a profiling tool do?
  - Gives timing information for method execution
  - Tracks object creation
  - Helps programmers to find bottleneck and optimize code
- Why another profiling tool?
  - There isn't a good free and open source java profiling tool available
- USFJPROF is a free open source java profiler

2

Main Features

- USFJPROF tracks:
  - Both inclusive and exclusive timing information for method execution (min/max/average)
  - Object creation and constructor invocations
  - Number of sub-method calls for each method

3

Main Features

- Include/exclude packages or files from tracking
- Allow user to choose methods on which the output could be focused: both inclusive and exclusive time for every calls to these methods will be outputted to files
- can use native system clock on Linux, PC
- Instrumented code can be run as a standalone java application

4

Idea: Source to Source Translation

- instrument source code with profiling method calls; then run the instrumented code to gather profiling information
- Use ANTLR to do the translation

5

Advantages

- Provides very fine level of control and is extremely fast
- Can track object creation
- Very little overhead introduced to target execution

6

Implementation

- Desired application source packages and files are instrumented. The translated files then are placed in temporary directory and compiled.
- The main method is called from within usfjprof, profiling information is recorded and outputted.

7

Application Instrumentation

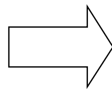
- Added actions in java grammar file for ANTLR
- While parsing java source file, profiling method calls are inserted into the token stream

8

Application Instrumentation

- Sample method instrumentation:

```
public void foo(int arg)
{
    Test t=new Test();
    … // code
}
```



```
public void foo(int arg) {
    try {
        ProfHandler.methodEntry( ...
);
        Test t = (Test)
         ProfHandler.createObject(
              new Test() );
        … // code
    }
    finally {
        ProfHandler.methodExit( ... );
    }
}
```

9

Example Output (A Simple Loop Example with 10,000 iterations)

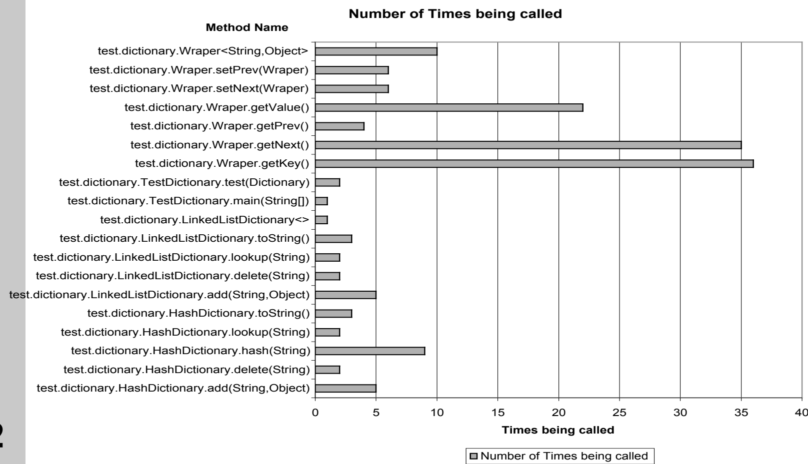
```
## ========================== Thread main ==============================
## ____________________________________________________________________
## Method execution time:
## method name | times called | inclusive: average | min | max | exclusive: average | min | max
## ____________________________________________________________________
test.Loop.loop(int) 1 354.2230 354.2230 354.2230 185.9007 185.9007 185.9007
test.Loop.main(String[]) 1 354.5157 354.5157 354.5157 0.2928 0.2928 0.2928
test.Loop.task() 10000 0.0168 0.0084 12.7994 0.0168 0.0084 12.7994
## ____________________________________________________________________
## Number of methods called:
## method name | (method name | times) *
## ____________________________________________________________________
test.Loop.loop(int) test.Loop.task() 10000
test.Loop.main(String[]) test.Loop.loop(int) 1
test.Loop.task()
## ____________________________________________________________________
## # of constructors called:
## method name | (constructor name | times) *
## ____________________________________________________________________
test.Loop.loop(int)
test.Loop.main(String[])
test.Loop.task()
## ____________________________________________________________________
## Number of instances created:
## class name | (number of instances) *
## ____________________________________________________________________
int[][] 10000
```

10

Output Graph

11

Output Graph



12

Known issues

- Not able to give accurate timing information when multithreading is used
- garbage collection, JIT compilation time are included
- Without using native code to check the microsecond clock, results will appear to be 0.00ms for most small routines
- Not able to profile programs that require standard input

13

The End

- Please visit

 http://cs.usfca.edu/projects/usfjprof/

 More details and downloads are available!

14