Comparison of the Execution Times of Ada, C and Java

Marcus Weiskirchner
EADS Deutschland GmbH
Military Aircraft
81663 Muenchen, Germany
marcus.weiskirchner@m.eads.net

September 25, 2003

Abstract

This paper presents a comparison of the execution times in Ada, C and Java. Two Benchmarks were used for this comparison, Whetstone [CW76] and Dhrystone [Wei88]. These benchmarks were compiled with Ada-Multi [GHS03] and Ada-GNAT [GNA01], with two Java Real-Time Virtual Machines Perc VM [New02] and Jamaica VM [SW01], with the gcc C-compiler as a reference program and interpreted with the Sun VM [LY99]. Furthermore, different operating system have been used, like VxWorks as a real-time operating system, Windows, Linux and Unix Sun Solaris.

1 Introduction

This report should give a comparison of the execution times in Ada, C and Java, using two Benchmarks, Whetstone and Dhrystone. Section 2 gives a short description of both benchmarks, describing what tests are made and how the specific measurements were made. Section 3 tries to interpret the benchmarks execution times on a VxWorks system with all the existing compilers for VxWorks in Java, Ada and C. Some differences in the compiled programs are also mentioned. Section 4 lists all used operating systems and how the execution times of the benchmarks and the compilers change, depending on the underlying operation system. Not every compiler was available for every operating system. Section 5 includes some further information, like a comparison of the filesizes of different compiled programs, ahead-of-time (AOT) versus just-in-time (JIT), the reproducibility of the benchmarks and some unexpected results of the test. All diagrams of the benchmark's

execution times are included to appendix A to D. Appendix E lists the used compiler options.

2 Used Benchmarks

Two benchmarks have been used for measuring the execution time. Dhrystone [Wei88] as a fixed point benchmark and Whetstone [CW76] as a floating point benchmark. It was important to have some benchmarks available in Ada as well as in Java. The benchmarks in C are a kind of reference.

2.1 Whetstone

There are two versions of the Whetstone benchmark [CW76], which had been tested. One with a 64-bit calculation and the other with 32-bit calculation. All test results are included to the appendix. Most tests are faster in the 32-bit calculation, that's why the comparison of the compilers isn't different from the 64-bit test.

The benchmark has 8 different single tests. These tests are executed in a loop with varied loop cycles. The following table gives a short description of the single tests and the weight of each test:

- Array elements (floating point), 12
- Array as parameter (floating point), 14
- Conditional jumps (if then else), 345
- Integer arithmetic (fixed point), 210
- Trigonometric functions (sin, cos, etc.), 32
- Procedure calls (floating point), 899
- Array references (assigns), 616
- Standard functions (exp, sqrt, etc.), 93

The whole test was executed with an iteration of 100 in each single test and a loop cycle of 100 over all single tests. This full test was run ten times, incrementing the loop cycle by ten (means 100 cycles first time, 110 second, 120 third, etc.). The result of the test is the average execution time for one single cycle and the average over all ten tests.

2.2 Dhrystone

The Dhrystone benchmarks [Wei88] is used to model what was viewed as a ”typical” application mix of mathematical and other operations. Integer performance predominated, with little or no floating-point calculations, and

applications could be contained inside small memory subsystems. Most operations include invoking other functions and procedures, doing some calculations and returning the result to the invoking procedure. In Ada and C pointers are used to return results. Because Java doesn't use pointers that way an object was created to return the result to the invoking methods. The first version of the Java program was creating a new object every time the method was called. Thus the comparison of the languages wasn't 100% fair and that's why Java took a lot more time for executing this benchmark. Therefore in the modified version (or better: the more compareable version) of the benchmark, a global object (reference) was used to return the result to the invoking method. Now the test isn't measuring the time it takes to create an object, but the time for executing the methods. Because of this small conditioning the Java benchmark is up to four times faster.

One run of the benchmark includes 100.000 iterations of the main loop. The result of the benchmark is the average time needed for one run.

3 Interpretation of the Test Results

The more interesting testing was made on the real-time operating system VxWorks. Thus these test results are taken for the further discussion. Otherwise the results of the other operating systems and the different hardware is similar to the VxWorks system. All other test results are included in appendix A to D.

3.1 Whetstone on VxWorks

Figure 1 shows the result of the Whetstone benchmark on a VxWorks system, version 5.4 on a VME board with a PPC 400 MHz and a 750 kernel. To have a fair comparison, some details have to be considered. Including or excluding **runtime-checks** has a big influence in optimising speed. C does not have any checks during runtime. Java is always performing runtime-checks and in Ada runtime-checks can be switched on or off by a compiler switch.

In figure 1 both Ada compiler use runtime-checks. Therefore we have a more or less direct comparison of the execution time of the Whetstone benchmark in Ada and in Java. All programs are optimized for speed, or better to say, the used compiler options are optimized for speed. The result of the benchmark was a bit suprising. I did not expect a faster execution time in Java than in Ada, compiled with Ada-Multi. But at a second try I used the Ada-GNAT compiler, which seems to compile the program better, means the GNAT compiled Whetstone runs faster than the Java program, compiled with the Jamaica VM.

The last bar in the diagram shows the execution time of the C-compiled program. This is the fastest version. On the other hand there must be noted, that the C-version doesn't make any runtime-checks. Therefore there is an-

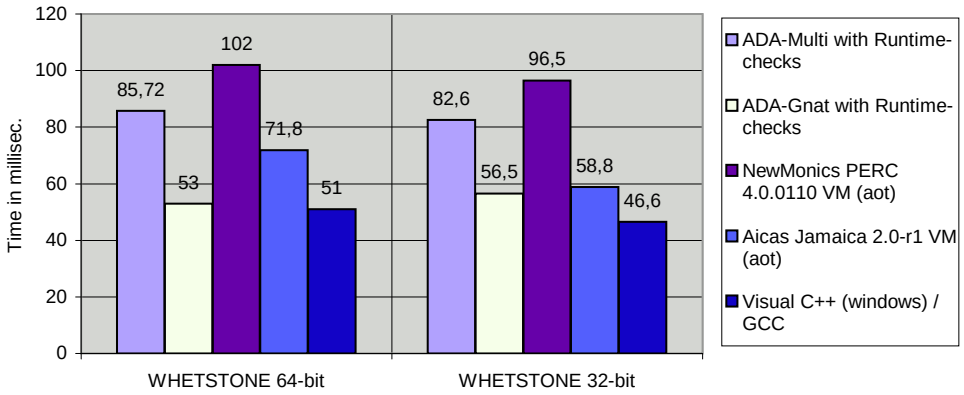

Figure 1: Whetstone 64/32-bit with Runtime Checks

other diagram comparing the Ada compiled programs with the C-version. Here the timings of the Ada compiled programs are closer to the C-version (see fig. 5).

3.2 Dhrystone on VxWorks

Figure 2 shows the result of the Dhrystone benchmark on a VxWorks system, version 5.4 on a VME board with a PPC 400 MHz and a 750 kernel. As mentioned above in the Whetstone benchmark, including or excluding **runtime-checks** has a big influence in optimising the speed.

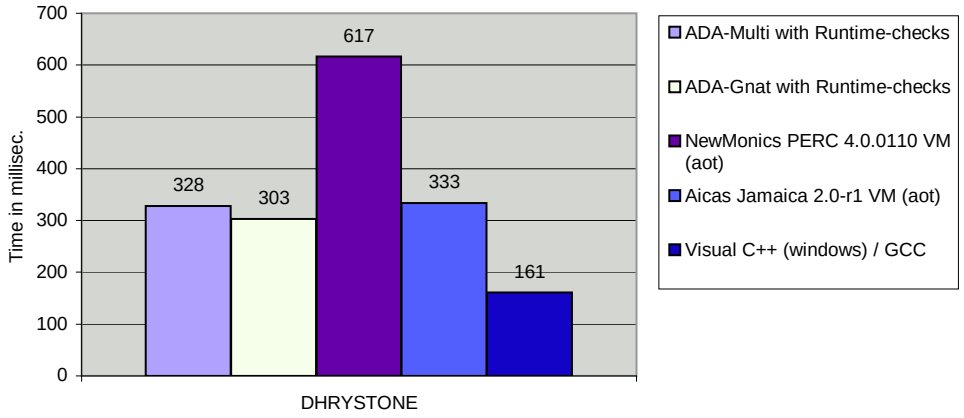

Figure 2: Dhrystone with Runtime Checks

In figure 2 again two Ada compilers (Ada-Multi and Ada-GNAT) and

two Java Real-Time Virtual Machines are compared. This time the Ada programs are executed faster or equal than the Java programs, depending on the used compiler. Again, all benchmarks without runtime-checks make the Ada programs almost two times faster.

4 Differences in Operating Systems

4.1 Windows

In figure 3 both Ada programs, the Perc VM, the Sun VM and the C program are compared. Again to be correct the C program is a bit out of the comparison, making no runtime-checks. But also the Sun VM is out of place, because it can not guarantee any time-constraints (no real-time garbage collection). These results should just be a reference to the results.

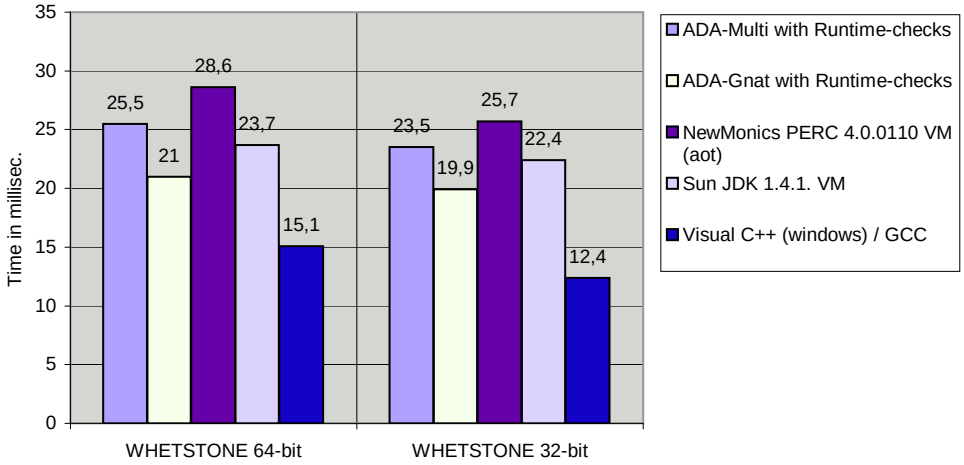

Figure 3: Whetstone 64/32-bit with Runtime Checks

But still, if you compare the Ada programs with the Perc VM, the Java program is almost as fast as the Ada programs. The difference between Ada to C gets smaller without runtime-checks (see Fig. 13, p. 13)

The Dhrystone benchmark has different results than the Whetstone. Here the Ada-Multi program needs half the time as Ada GNAT. The reference programs with C and Sun VM have there own competition and are a lot faster than the others. The interpreted (JIT) Java program needs almost the same time as the C program. The RT Perc VM executes the benchmark a little bit slower. It also seems, that the JIT compilation works perfect for this example. The full Dhrystone benchmarks was run 10 times (started be the main-method), the first run was the slowest, and all other runs were faster.

An explanation for this effect can be, that reused parts of the program don't have to be recompiled and therefore are executed faster than the first run.

4.2 Sun Solaris

Appendix C.1, p. 17, show the results of Ada programs with/without runtime-checks and of the JIT interpretation of the Sun VM. As a conclusion it can be said, except of Ada-Multi, all other execution times have more or less the same execution time (between 32 ms and 42 ms).

The Dhrystone benchmark has a similar view. Just the C program needs half the execution time as the GNAT (without runtime-checks).

By the look at the diagrams always have in mind that there are a lot of differences in the programs itself. Here the Ada programs with and without the runtime-checks are in the same diagram, as well as the C program (no runtime-checks) and the Java program, which is a JIT compilation, means just-in-time compilation with no timing guarantees.

4.3 Linux

The appendix C.2, p. 18 shows the diagrams of Whetstone and Dhrystone on a Linux operating system. Four different programs compiled with Perc, Jamaica, Sun VM and C are compared. The two RT Virtual Machines have almost the same execution time. The C Programm is the fastest, but has still a similar execution time. The Dhrystone shows a very clear picture, RT Virtual Machines take a lot more time to execute the benchmark as Sun VM and the C program, which are two respectively almost five times faster.

5 Additional Information

5.1 Filesize

Sometimes the file size can also be an interesting issue. Therefore appendix D.1, p. 19 shows a bar diagram of the different languages and the file sizes. For VxWorks Ada must include the Ada-Runtime Object adalib.o. Java programs have a similar problem and have to include the Java Virtual Machine. But as it looks in the size comparison, Perc VM includes a lot more libraries than Jamaica or doesn't make a perfect optimization of code. But no other language can beat the C compilation, which size is just 8 kb.

On a Windows System the file sizes are similar and again the Perc VM needs a lot of disc space. There is no Version of Jamaica available for Windows.

5.2 AOT versus JIT

The Perc VM includes ready compiled Virtual Machines for every platform. Thus it is possible to let run Java byte code directly with this JIT VM. To optimize the bytecode before starting the program, an optimization tool is included to get an ”accelerated” bytecode. The figure D.2 compares the execution time of a AOT compiled program and the JIT version of the benchmarks on different platforms. With the two benchmarks the AOT version and the JIT version of the program have almost the same execution time.

5.3 Reproducibility

During the benchmarks it was conspicuous, that the execution times of some single test were not exactly the same time as in the test before. As a first assumption I disabled the instruction and data cache to get rid of cache effects. To be sure that the telnet daemon is not working during the Benchmark, I used the serial connection for communication. All these test were made on the VxWorks system to have no operating system side effects. But the runtime effects were still the same.

The conclusion of these tests are: Ada and C always produce the same execution times on every single test, only a tolerance of one millisecond on some tests. Jamaica produced a less reproduceable benchmark view. There were much more differences in the single tests, but still less than 0,5% tolerance overall. The Perc VM has differences in the single tests up to 1,5% tolerance overall.

Some speculations of Aicas, why there is a small tolerance:

- The synchronisation thread is on. Should be off by default in VxWorks. Manually set off achieved no difference.
- The benchmark itself runs within different threads with same priority. Different priorities of threads could help and should be mapped to different VxWorks priorities. This is the default for VxWorks.

5.4 Curiosities

The Whetstone benchmark compiled with the **Ada-GNAT** for VxWorks executes the 64-bit program faster than the 32-bit version, what is an unexpected result.

Comparing the Dhrystone benchmark compiled with **Ada-GNAT** and **Ada-Multi**, the Ada-Multi version is two times faster, but only on the Windows operating system.

6 Conclusion and Future Work

There are a lot of dependencies influencing the execution times of the benchmarks. The benchmarks itself have totally different results in execution times. Not every compiler is available for every operating system. Every compiled program has a different execution time, depending on the used compiler. The language is not the main reason, it is more the used compiler. For the Whetstone benchmark it definitely depends on the used compiler, most of the platforms provide a similar result. The Dhrystone benchmark has different execution times on different platforms. The first version of Dhrystone was probably translated to Java by a automatic tool. Therefore on every method invocation a new object was created, unlike the original C or Ada code. This artificial object creation was removed from the Java code for the comparison of the three languages.

The runtime-checks are also a big issue, Java always uses runtime checks, C never and Ada by compiler switch. Therefore two kinds of diagrams are appended. Java is always compared with the programs using runtime checks (except C). Some tests with the Sun VM were also made, although this is a non real-time VM having no RT garbage collection.

Figure 1 gives probably the best comparison of this paper, comparing two Ada compiled programs, two Java compiled programs (with real-time Java VMs) and a C compiled program. All tests were made with the RT operating system VxWorks on a Power PC. The benchmark itself makes a lot of calculations with floating points, no pointers are used.

As a conclusion, the difference in execution time between realtime Java and Ada is less than expected. Differences between the Ada runtime environments seem to have a more significant impact on execution speed than choosing between Ada and Java as long as runtime-checks are used. As demonstrated by the machine-translated Java benchmark program, it is important to avoid object creation in methods that are executed frequently inside loops.

A more detailed test on tasking should be made in future. Hartstone would be a good standardized benchmark for testing task switches and task behavior, but is presently not publicly available for Java. A short benchmark sending and receiving messages between real-time tasks using a message queue can also help to have a better view of task behavior and timing constraints of Java and Ada.

A Benchmarks on VxWorks Systems

A.1 VxWorks 5.5 with a PPC603 350MHz

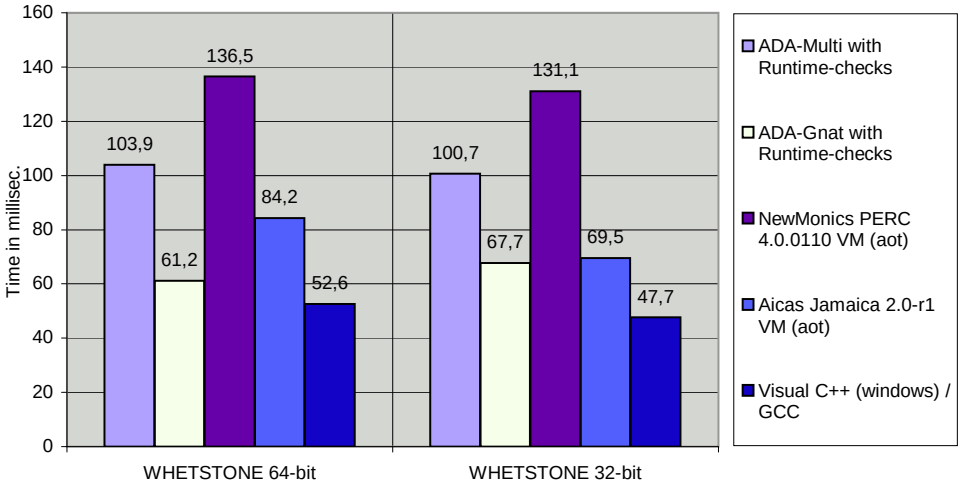

Figure 4: Whetstone 64/32-bit with Runtime Checks

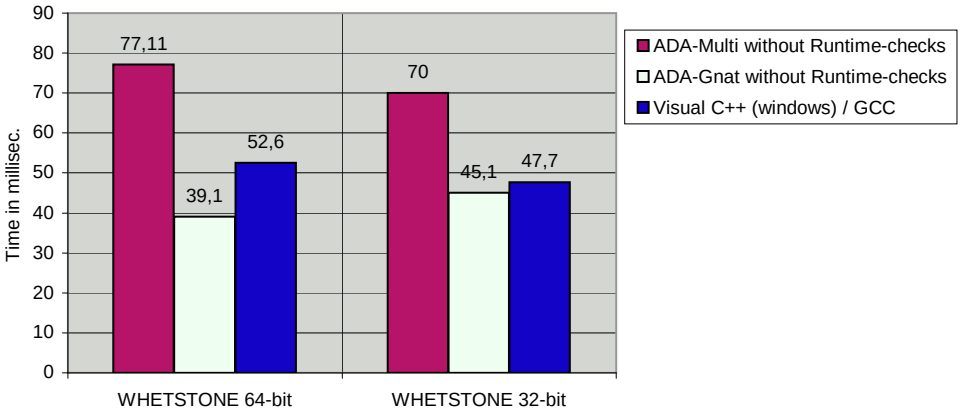

Figure 5: Whetstone 64/32-bit without Runtime Checks

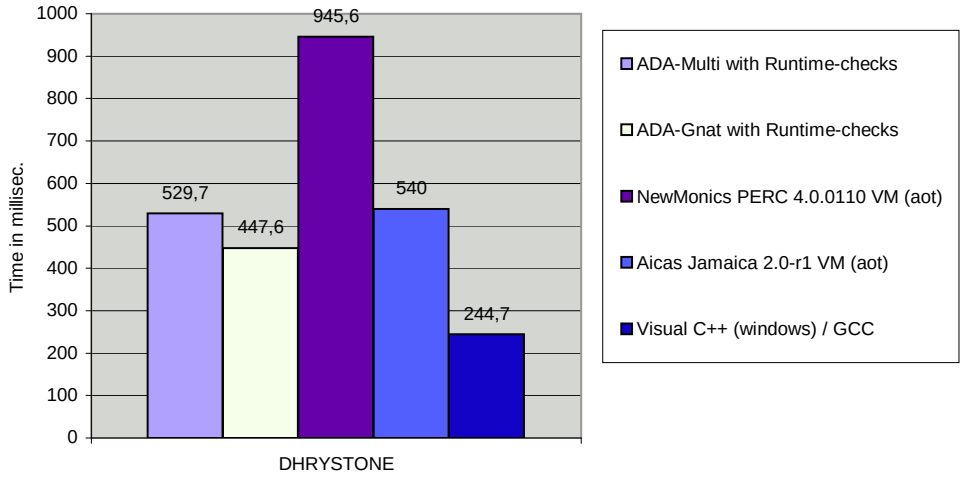

Figure 6: Dhrystone with Runtime Checks

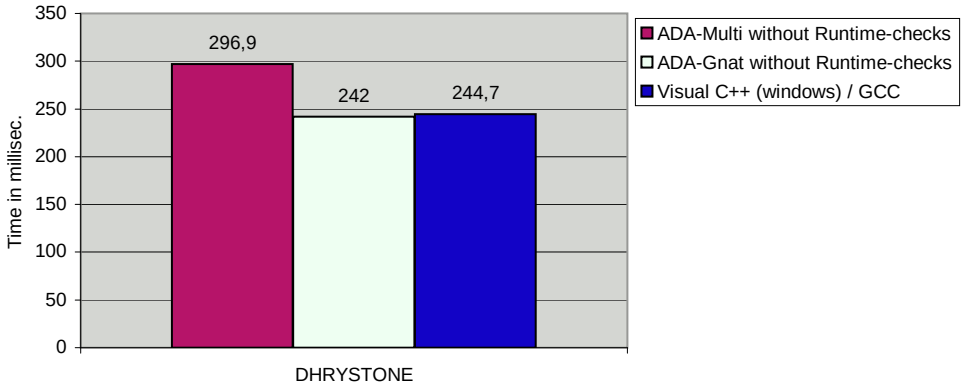

Figure 7: Dhrystone without Runtime Checks

A.2 VxWorks 5.4 with a PPC750 400MHz

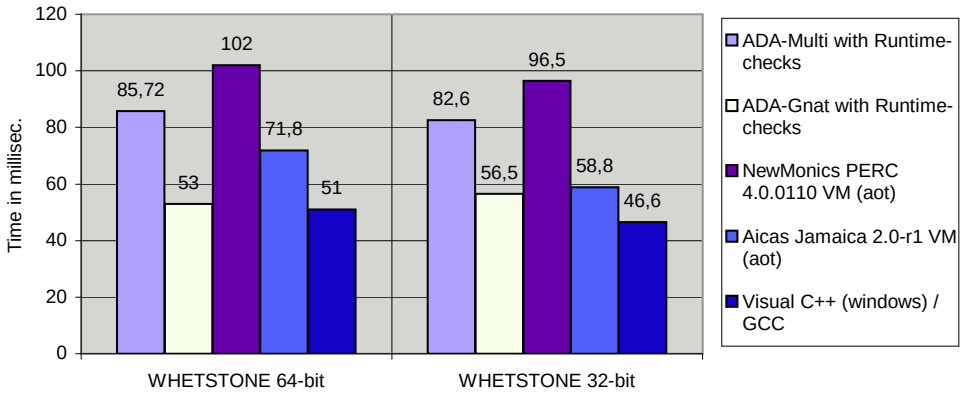

Figure 8: Whetstone 64/32-bit with Runtime Checks

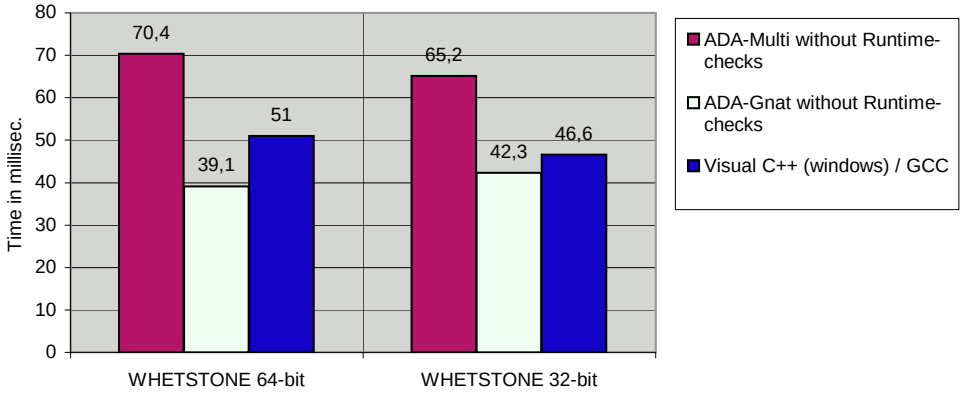

Figure 9: Whetstone 64/32-bit without Runtime Checks

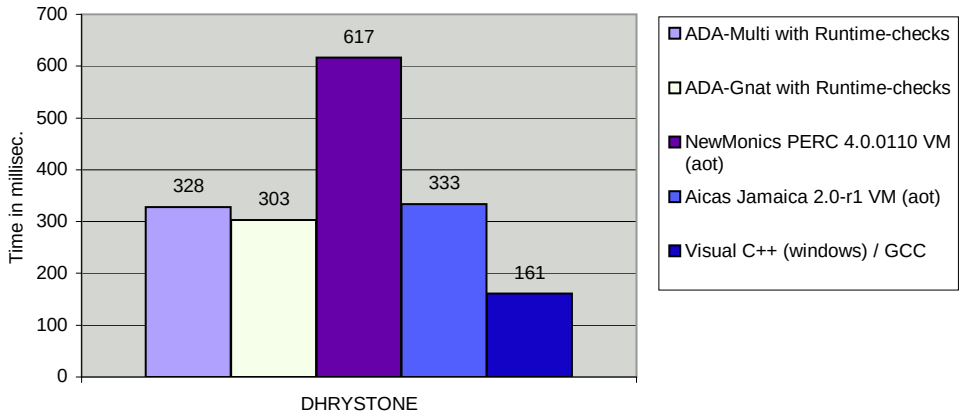

Figure 10: Dhrystone with Runtime Checks

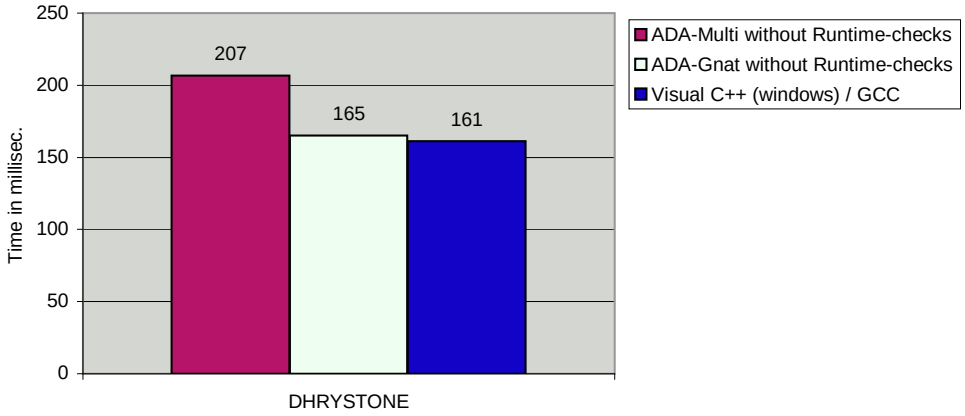

Figure 11: Dhrystone without Runtime Checks

B Benchmarks on Windows Systems

B.1 Windows XP with AMD Athlon 2000+

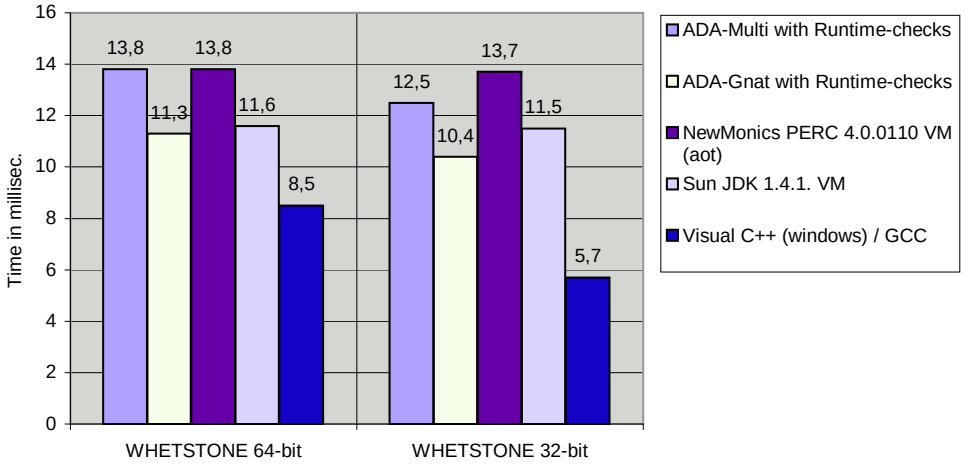

Figure 12: Whetstone 64/32-bit with Runtime Checks

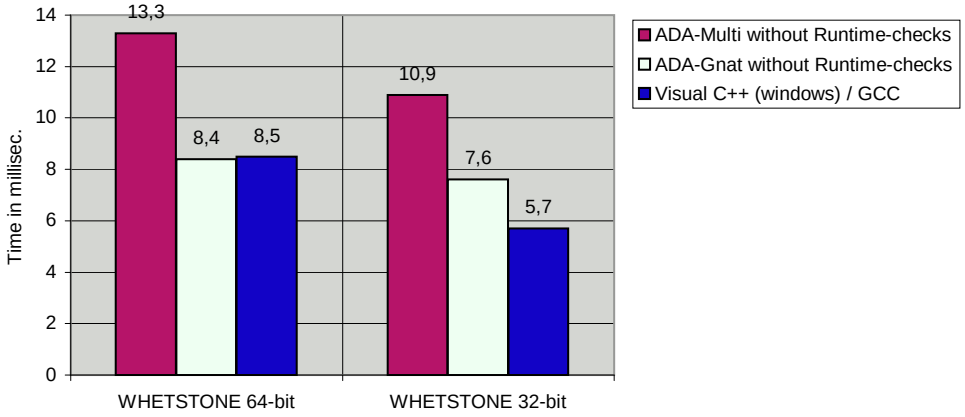

Figure 13: Whetstone 64/32-bit without Runtime Checks

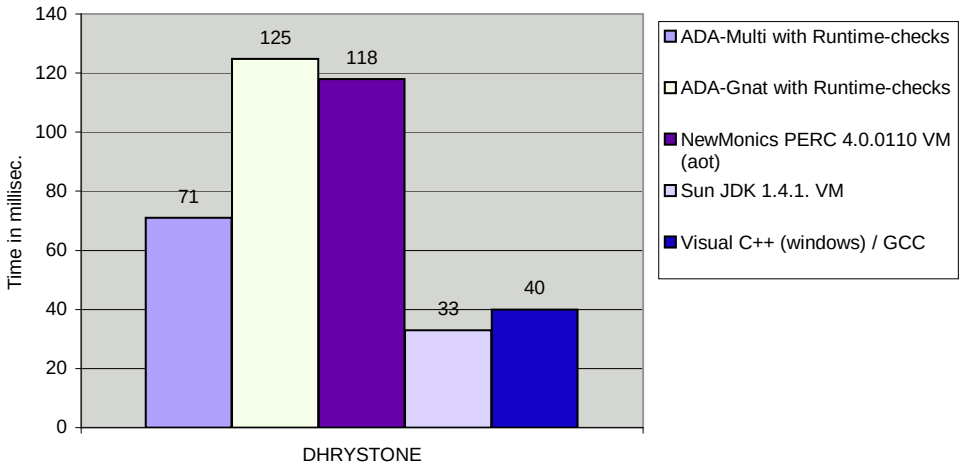

Figure 14: Dhrystone with Runtime Checks

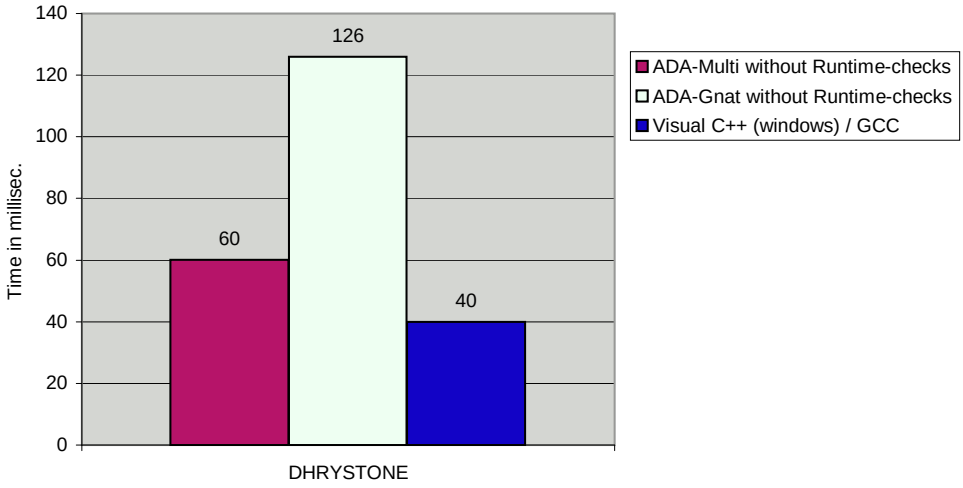

Figure 15: Dhrystone without Runtime Checks

B.2 Windows 2000 with Intel Pentium III 933MHz

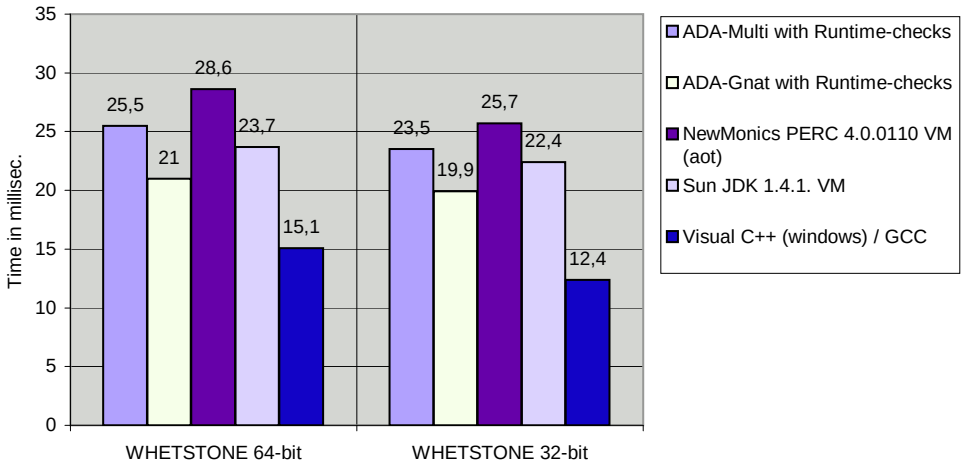

Figure 16: Whetstone 64/32-bit with Runtime Checks

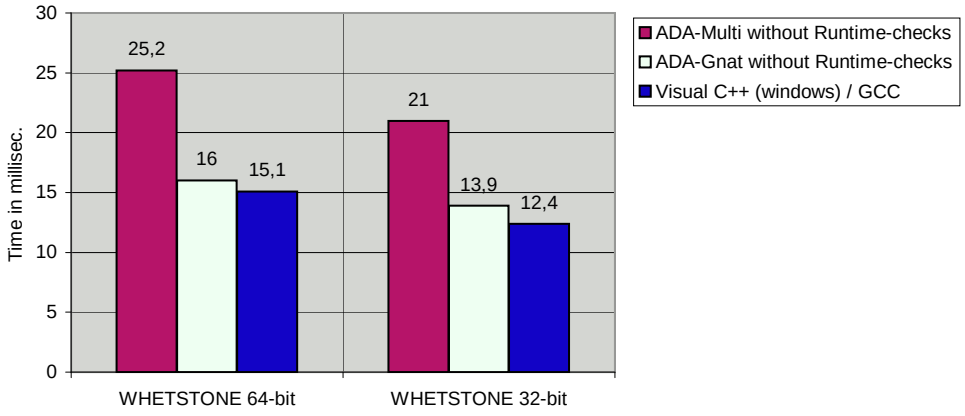

Figure 17: Whetstone 64/32-bit without Runtime Checks

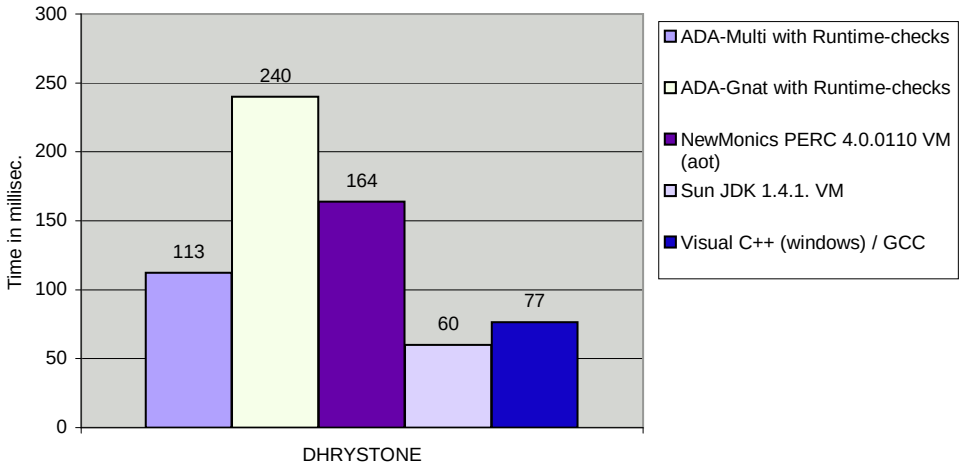

Figure 18: Dhrystone with Runtime Checks

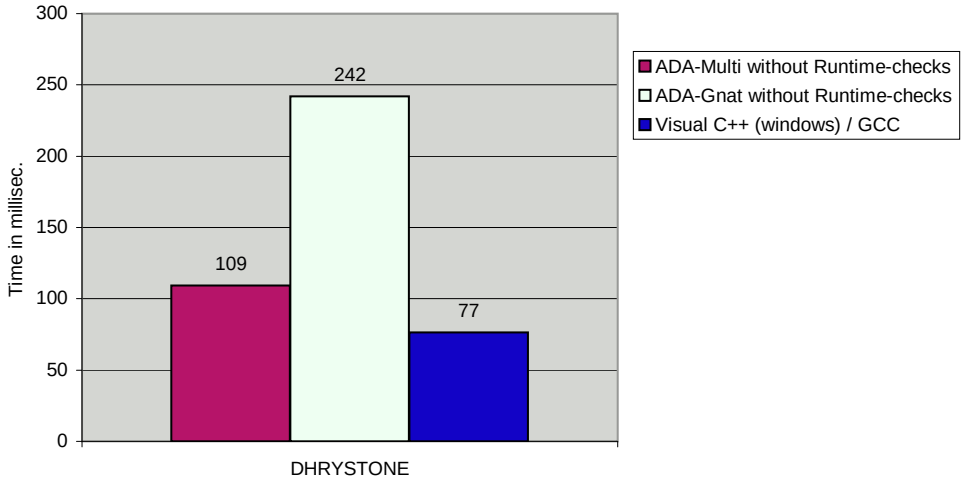

Figure 19: Dhrystone without Runtime Checks

C Benchmarks on Unix and Linux System

C.1 Sun Solaris with SPARC Processor

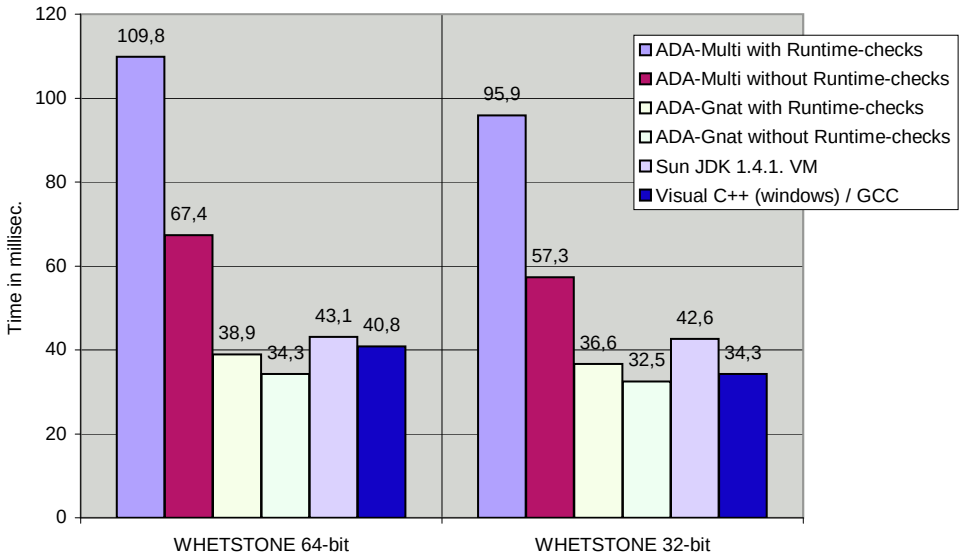

Figure 20: Whetstone 64/32-bit

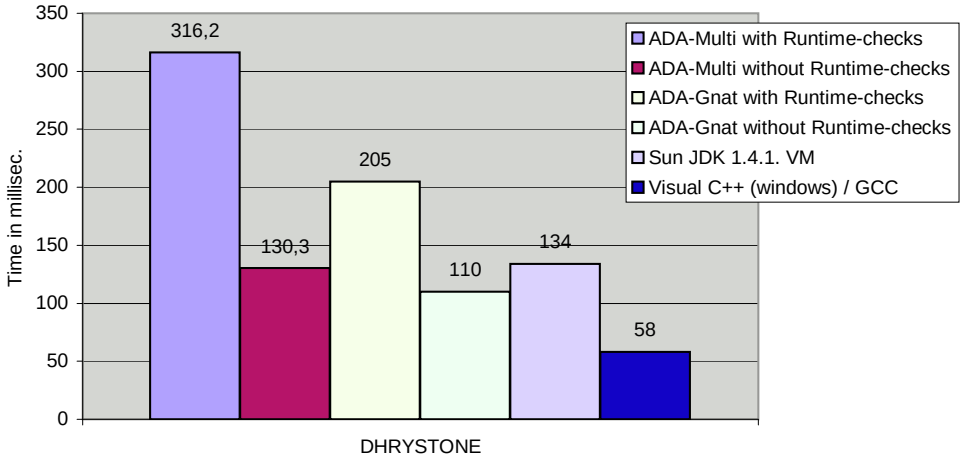

Figure 21: Dhrystone

C.2 Linux SuSe 8.0 with AMD Athlon 2000+

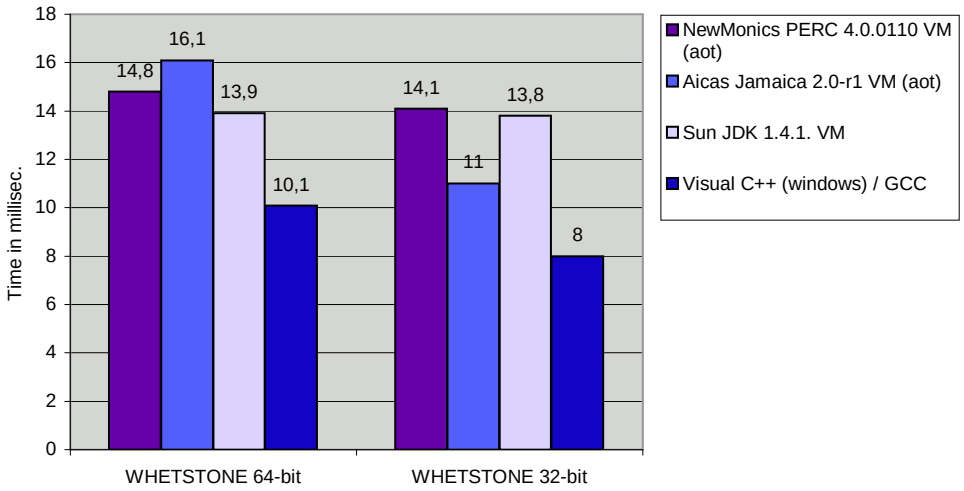

Figure 22: Whetstone 64/32-bit

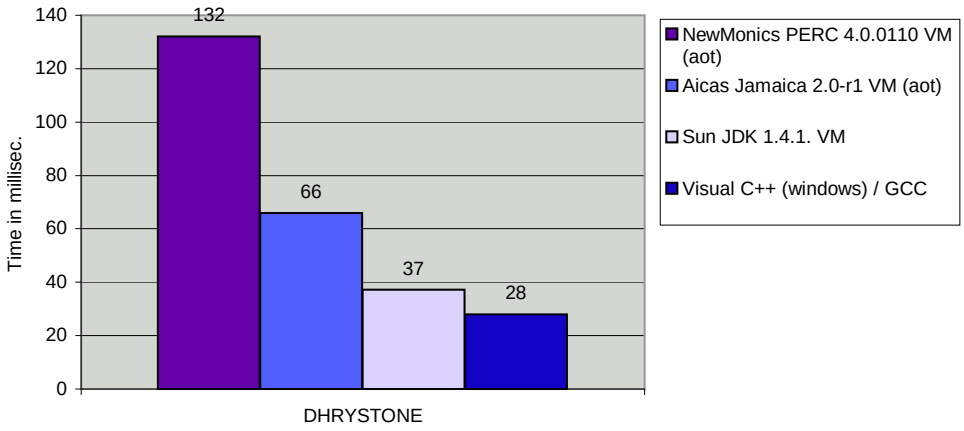

Figure 23: Dhrystone

D Additional Information

D.1 Comparison of the File Size

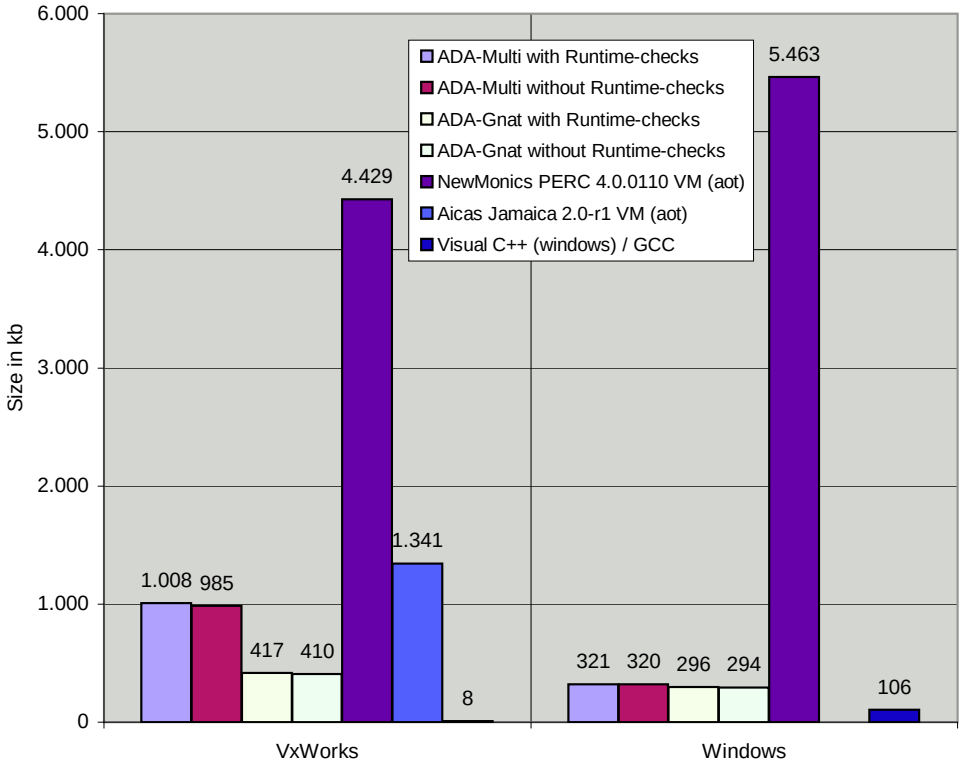

Figure 24: Different Filesizes

D.2 Comparison PERC AOT vs. JIT

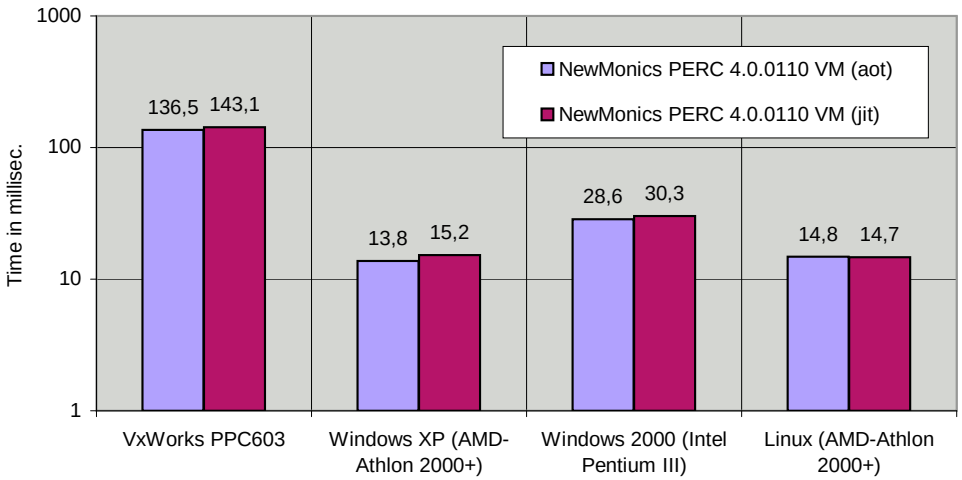

Figure 25: AOT versus JIT

E Compiler Options

E.1 NewMonics Perc VM

```
romize
  -r debug
  -target PPC -os VxWorks
  -pni-classes pni_classes
  -aot
  -hardware-floating-point
  -lazy-resolve
  whetstone/whet -o image.o
ldppc -o pvm.o -r hooks.o main.o pvmStart.o image.o
```

E.2 Aicas Jamaica VM

```
jamaica
  -compile
  -smart
  -timeSlice 0
  -target vxworks-powerpc
  -heapSize 20m -stackSize 150k -numThreads 15 -numDynamicTypes 0
  whetstone.whet
```

E.3 Ada-Multi

In the GUI some options are made different from the default:

- File Options: Optimization: Optimize for Speed
- Language Options: Ada: Runtime checks on/off

E.4 Ada-GNAT

```
powerpc-wrs-vxworks-gnatmake
  -O3
  -gnatn
  -v
  -mlongcall
```

and `-gnatp` without runtime-checks

E.5 C / Visual C++

```
powerpc-wrs-vxworks-gcc
  -O2 -nostdinc -nostdlib -ffreestanding
  -Wall -U_BIG_ENDIAN -DCPU=PPC603 -mlongcall
  -o c/whetstone.o -c c/whetstone.c -DNDEBUG
powerpc-wrs-vxworks-ld
  -r -nostdlib -u main -o c/whetstone c/whetstone.o
```

Or for the Linux system:

```
gcc
  -Wall -o c/whetstone c/whetstone.c
  -I/usr/include -L/usr/lib -lm -O2
```

And for the Windows System with Visual C++ the ”Optimization for Speed” was choosen.

References

[CW76] H. J. Curnow and B. A. Wichmann. Whetstone benchmark: A synthetic benchmark. *The Computer Journal, Volume 19, No.1 (Feb. 1976)*, 1976.

[GHS03] Green Hills Software Inc. GHS. Adamulti - ada 95 integrated development environment. *http://www.ghs.com*, 2003.

[GNA01] Ada Core Technologies GNAT. Gnat reference manual. *http://gcc.gnu.org*, 2001.

[LY99] Tim Lindholm and Frank Yellin. *The Java Virtual Machine Specification*. Addisson Wesley, second edition, 1999. http://java.sun.com.

[New02] Inc. NewMonics. Perc virtual machine 4.0 – user manual. *http://www.newmonics.com*, 2002.

[SW01] Fridtjof Siebert and Andy Walter. Jamaica virtual machine – user documentation. *http://www.aicas.com*, 2001.

[Wei88] Dr. Reinhold P. Weicker. Dhrystone benchmark: Rationale for version 2 and measurement rules. *SIGPLAN Notices 23,8 (Aug. 1988), [49-62]*, 1988.

Contents