IBM XL C/C++ and Fortran Compiler for AIX

Enhanced Performance with latest XLC/C++ compiler version 12.1 and IBM XL Fortran Compiler version 14.1 for AIX

By: Anh Tuyen Tran

Level: Introductory

June 2012

Contents

Before you start

About this series

Walk through this scenario and others online as part of the IBM XL C/C++ and Fortran Compiler for AIX.

About this Tutorial

This Demo demonstrates the performance gain achievable when using the latest version of the compiler, as a result of the optimizations incorporated in the latest version of IBM compilers. This demo is applicable to both C/C++ and Fortran compilers.

Since performance depends on multiple factors such as computer architecture, CPU power, available memory and most importantly the characteristics of each individual program, users might experience some fluctuation in performance results. However, compiling the same program with the latest version of IBM compiler can generally bring about an improvement in performance.

Objectives

- Use the latest IBM compiler to compile a program and measure its running time
- Compare the running time of the executables compiled with different versions of the compiler.

- Total time: 15 minutes

Prerequisites

- Basic Unix skills
- Basic C and/or Fortran language knowledge

System Requirements

http://www.ibm.com/software/awdtools/xlcpp/aix/sysreq/
http://www.ibm.com/software/awdtools/fortran/xlfortran/aix/sysreq/

Glossary

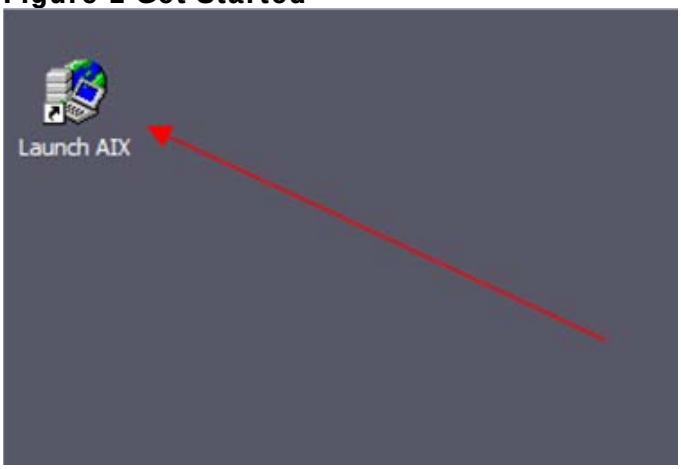
IBM XL C/C++ Compiler: IBM® XL C and C++ compilers offer advanced compiler and optimization technologies and are built on a common code base for easier porting of your applications between platforms. They comply with the latest C/C++ international standards and industry specifications and support a large array of common language features.

IBM XL Fortran Compiler: The IBM® XL Fortran compiler offers advanced compiler and optimization technologies and is built on a common code base for easier porting of your applications between platforms. It complies with the latest Fortran international standards and industry specifications and supports a large array of common language features.

Getting Started

Start the Terminal Emulator to AIX System

Figure 1 Get Started



Double click the “Launch AIX” icon on the desktop (See Figure 1) to start the terminal connection to AIX system.

Get started with compiler performance

Successful login will result with the user presented with a menu of demos hosted on the server. Type 11 and press Enter to select the “Latest compiler provide better performance” demo.

Figure 2 Demo Prepared

```
################################################################################

        Thank you for waiting, setup is now ready

        IBM XL C/C++ version 12.1 compiler is installed at /usr/vac
        IBM XL Fortran version 14.1 compiler is installed at /usr


        Old version of compiler is installed at following location

        IBM XL C/C++ version 11.1 compiler is installed at
        /opt/IBM/old.c/111/usr/vac
        IBM XL Fortran version 13.1 compiler is installed at
        /opt/IBM/old.fortran/131/usr

        Note: Starting another command window will redo the demo setup
              of your environment. This will result in loss of any
              work done in your home directory.

################################################################################
```

In the terminal window you will see important information and the directory path to the compiler install directory (See Figure 2 Demo Prepared).

Note: Starting another command window will start the demonstration setup of your environment. This will result in loss of any work done in your home directory. This will impact any progress you have made on demo steps going forward.

This demo does not require more than one terminal window. However, if you prefer more than one terminal window then you may open them before going forward.

The terminal window is now ready for commands. Your home directory contains necessary source code to perform the tutorial. Type the ls command to see the directory content (See Figure 3 Contents).

Command:

```
ls
```

Figure 3 Contents

Notice two (2) directories in your home directory.

C	This directory contains C Source code for this demo
Fortran	This directory contains Fortran Source code for this demo

Steps:

The purpose of this demo is to demonstrate the performance gains achieved using the latest version of the compilers, as compared to an older version of compilers on the same hardware. The executable produced by the latest version of XL C/C++ compiler V12.1 and XL Fortran compiler V14.1 can yield a performance gain. This is performance difference is more visible for programs that handle large amounts of data.

This demo has two sections for C/C++ code example and Fortran code example. You may choose to conduct either or both of them.

Part 1: C/C++ Source code steps to demonstrate new compiler performance:

C/C++ program given in this demo multiplies 2 matrices with large dimensions. Please note the program was not designed to perform optimal matrix multiplication; it was meant to make use of very large loops in a program. Due to the size of data, the number of registers available on the CPU becomes insufficient to hold all program data effectively. With the improvements incorporated in XL C/C++ V12.1 compiler the executable performs better than executable produced by XL C/C++ V11.1 compiler.

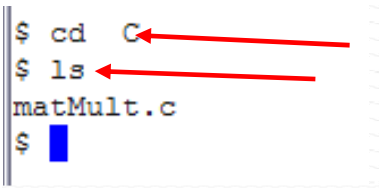
For comparison purpose, you will compile the program with the same compiler options, at the same optimization level, on the same machine.

1. Change directory to 'C' and list its contents

   Command:

```
cd C
ls
```

Figure 4 Change directory and list



Notice one (1) file in C directory.

matMult.c	C source code that does matrix multiplication.

2. Compile the source code using version 11.1 of XL C/C++ compiler

   Command:

```
/opt/IBM/old.c/111/usr/vac/bin/xlc -O5 -q64 -qstrict -qaltivec  \
   -qsimd=auto -qhot=simd -qarch=pwr7 -qtune=pwr7 -o v11MatMult \
   ./matMult.c
```

Figure 5 Compile using version V11.1 of XL C/C++ compiler

```
$ /opt/IBM/old.c/111/usr/vac/bin/xlc -O5 -q64 -qstrict -qaltivec
  -qsimd=auto -qhot=vector -qarch=pwr7 -qtune=pwr7 -o v11MatMult
  ./matMult.c
$
```

The compiler options used:

O5: The highest level of optimization. This demo can be compiled at any optimization level, even with no optimization at all.

q64: This option selects the 64-bit compiler mode. Specifying -q64 (or -q32) together with **-qarch** and **-qtune** compiler options, will optimize the output of the compiler to the architecture on which that output will be used.

qstrict: This option ensures that optimizations done at optimization levels –O2 and higher will not alter the semantics of a program.

qaltivec: This option instructs the compiler to support vector data types and operators.

qsimd=auto: This option enables automatic generation of vector instructions for processors that support them. It replaces the **-qenablevmx** option, which has been deprecated. With this option, the compiler converts certain operations that are performed in a loop on successive elements of an array into vector instructions. These instructions calculate several results at one time, which is faster than calculating each result sequentially.

qhot=vector: The **-qhot** compiler option is a powerful alternative to hand tuning that provides opportunities to optimize loops and array language. This compiler option will always attempt to optimize loops, regardless of the suboptions you specify.

qarch=pwr7 qtune=pwr7: Specifically targets power 7 CPU architecture for the compilation, which can provide additional performance improvements.

3. List the compiled file

   Command:
   ```
   ls
   ```

Figure 6 List file compiled with V11.1 compiler

```
$ ls
matMult.c    v11MatMult
$
```

4. Run the `v11MatMult` program and note time it takes for this program to run

   Command:
   ```
   time ./v11MatMult
   ```

Figure 7 Run the executable produced by V11.1 compiler

```
$ time ./v11MatMult
Elapsed Time Before I/O: 3489.49 ms.

real    0m5.90s
user    0m3.52s
sys     0m0.00s
$
```

Notice the time it takes for this program to run.

5. Compile the source code using version 12.1 of XL C/C++ compiler

   Command:
   ```
   xlc -O5 -q64 -qstrict -qaltivec -qsimd=auto -qhot=simd \
       -qarch=pwr7 -qtune=pwr7 -o v12MatMult ./matMult.c
   ```

   Note: C/C++ Compiler version 12.1 is setup in the default location and is available in your path.

Figure 8 Compile using new version (V12.1) of XL C/C++ compiler

```
$ /usr/vac/bin/xlc -O5 -q64 -qstrict -qaltivec -qsimd=auto
  -qhot=vector -qarch=pwr7 -qtune=pwr7 -o v12MatMult ./matMult.c
$
```

6. List the compiled file

   Command:
   ```
   ls
   ```

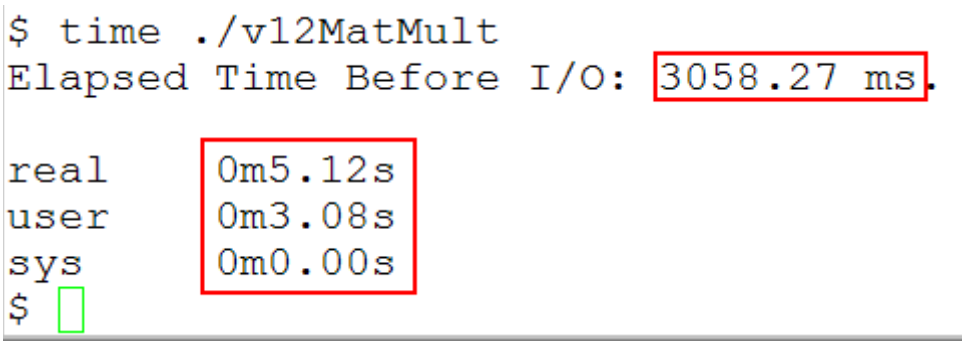
Figure 9 List file compiled with V12.1 XL C/C++ compiler

```
$ ls
matMult.c    v11MatMult  v12MatMult
$
```

7. Run the `v12MatMult` program and note time it takes for this program to run

   Command:
   ```
   time ./v12MatMult
   ```

Figure 10 Run the executable produced by V12.1 compiler



Note the drop in execution time for the executable compiled using XL C/C++ compiler version 12.1 compared to executable compiled with XL C/C++ compiler version 11.1 with same optimization options.

8. Change to your home directory

   Command:
   ```
   cd ..
   ```

Figure 11 Change directory back

This concludes the demo for C/C++ source code part of “Enhanced Performance with latest XLC/C++ compiler version 12.1 and IBM XL Fortran Compiler version 14.1 for AIX”.

Part 2: Fortran Source code steps to demonstrate new compiler performance:

The Fortran program used in this demo multiplies 2 matrices with large dimensions. Please note the program was not designed to perform optimal matrix multiplication; it was meant to make use of very large loops in a program. With the improvements incorporated in XL Fortran V14.1 compiler the executable performs significantly better than executable produced by XL Fortran V13.1 compiler.

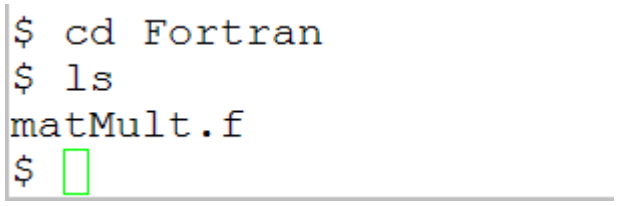
For comparison purpose, you will compile the program with the same compiler options, at the same optimization level, on the sandbox same machine.

1. Change directory to 'Fortran' and list its contents

   Command:

```
cd Fortran
ls
```

Figure 12 Change directory and list



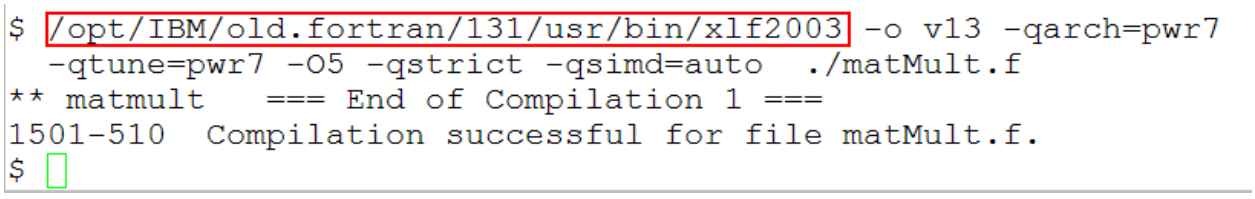
Notice one (1) file in C directory.

2. Compile the source code using version 13.1 of XL Fortran compiler

   Command:

```
/opt/IBM/old.fortran/131/usr/bin/xlf2003 -o v13 -qarch=pwr7 \
          -qtune=pwr7 -O5 -qstrict -qsimd=auto  ./matMult.f
```

Figure 13 Compile using version V13.1 of XL Fortran compiler



The compiler options used:

qarch=pwr7 qtune=pwr7: Specifically targets power 7 CPU architecture for the compilation, which can provide additional performance improvements.

O5: The highest level of optimization. This demo can be compiled at any optimization level, even with no optimization at all.

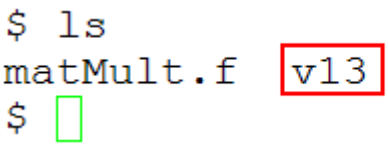
qstrict: This option ensures that optimizations done at optimization levels –O2 and higher will not alter the semantics of a program.

qsimd=auto: This option enables automatic generation of vector instructions for processors that support them. It replaces the **-qenablevmx** option, which has been deprecated. With this option, the compiler converts certain operations that are performed in a loop on successive elements of an array into vector instructions. These instructions calculate several results at one time, which is faster than calculating each result sequentially.

3. List the compiled file

   Command:
   ```
   ls
   ```

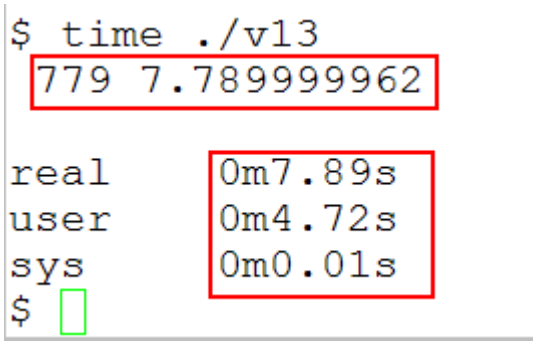
Figure 14 List file compiled with V13.1 compiler



4. Run the v13 program and note time it takes for this program to run

   Command:
   ```
   time ./v13
   ```

Figure 15 Run the executable produced by V13.1 XL Fortran compiler



Note the time it takes for this program to run.

5. Compile the source code using version 14.1 of XL Fortran compiler

   Command:
   ```
   xlf2003 -o v14 -qarch=pwr7 -qtune=pwr7 -O5 -qstrict -qsimd=auto \
           ./matMult.f
   ```

   Note: Fortran Compiler version 14.1 is setup in the default location and is available in your path.

Figure 16 Compile using version V14.1 of XL Fortran compiler

```
$ xlf2003 -o v14 -qarch=pwr7 -qtune=pwr7 -O5 -qstrict -qsimd=auto
  ./matMult.f
** matmult    === End of Compilation 1 ===
1501-510  Compilation successful for file matMult.f.
$
```

6. List the compiled file

   Command:
   ```
   ls
   ```

Figure 17 List file compiled with V14.1 compiler

```
$ ls
matMult.f   v13         v14
$
```

7. Run the v14 program and note time it takes for this program to run

   Command:
   ```
   time ./v14
   ```

Figure 18 Run the executable produced by V14.1 XL Fortran compiler

```
$ time ./v14
 716 7.159999847

real    0m7.24s
user    0m4.33s
sys     0m0.00s
$
```

Notice the drop in execution time for the executable compiled using XL Fortran compiler version 14.1 compared to executable compiled with XL Fortran compiler version 13.1 with same optimization options.

8. Change to your home directory

   Command:
   ```
   cd ..
   ```

Figure 19 Change Dir 2

This concludes the demo for Fortran source code part of “Enhanced Performance with latest XLC/C++ compiler version 12.1 and IBM XL Fortran Compiler version 14.1 for AIX”.

What you have learned

In this exercise you learnt how to:

- Use IBM XL C/C++ and Fortran compiler for AIX to compile source code.
- Use optimization flags to optimize the program

Conclusion

The tutorial presented here demonstrates how the latest XL C/C++ v12.1 and XL Fortran v14.1 compilers can provide performance improvements over previous versions of the compilers. It should be noted that the test cases presented are cases where improvements in the latest compiler are especially apparent, and user programs may see more modest improvements.

Trademarks

Resources

Optimization:

Optimization and Programming Guide - XL C/C++ for AIX, V12.1
http://pic.dhe.ibm.com/infocenter/comphelp/v121v141/nav/2_4

Optimization and Programming Guide - XL Fortran for AIX, V14.1
http://pic.dhe.ibm.com/infocenter/comphelp/v121v141/nav/3_4

Papers

"Optimizing C Code at Optimization Level 2"
http://www.ibm.com/support/docview.wss?uid=swg27022103

"Code Optimization with the IBM XL Compilers"
http://www.ibm.com/support/docview.wss?uid=swg27005174

"How to improve the performance of programs calling mathematical functions – Taking advantage of IBM XL C/C++ or XL Fortran compiler auto-vectorization"
http://www.ibm.com/developerworks/rational/library/10/improveperformanceprogramsmathfunctions/index.html

Community Cafe

http://www.ibm.com/software/rational/cafe/community/ccpp