OpenVMS Technical Journal V6

Porting RPG: Moving a Compiler to the HP Integrity Server

Porting RPG: Moving a Compiler to the HP Integrity Server

Bruce Claremont, Software Migration and OpenVMS Consultant

Overview

This article covers MSI's experience in porting its Migration RPG compiler to the HP Integrity server, based on the industry-standard Intel® Itanium® 2 processor. The article covers the following subjects:

- Our software development methodology

- Our porting plan

- The porting process

- Product Q/A testing after the port was completed

- Summary and recommendations to others planning to port applications

Introduction

MSI is a small, privately held corporation that specializes in software migration and HP OpenVMS services. We also have a unique niche in the industry in that we provide the only RPG compiler available under OpenVMS, *Migration RPG*. This article covers our port of the compiler to OpenVMS running on the HP Integrity server line.

Software Development Methodology

This section briefly describes MSI's software development methodology. It's relevant to the porting process because our methodology greatly simplifies software porting.

MSI's software is developed using structured programming techniques. We stick to the rule of one entry and exit point for each routine, the only exception being routines that handle fatal errors. We strive to make routines as small and concise as possible and to reuse routines whenever possible.

Our code is well documented. In the case of Migration RPG, most of the documentation is embedded within the source code, with a summary document that describes the overall structure of the compiler and defines key internal layouts. Our standard for source code documentation is to provide enough descriptive text to allow a programmer unfamiliar with the programming language in use to understand the purpose and structure of each module.

Migration RPG is primarily written in HP Macro-32. It also contains a few modules developed in HP Fortran and HP C. The entire product comprises 120 source files containing about 320,000 lines of code.

I became lead software engineer on the product 20 years ago when its native environment was VAX/VMS. At that time, the product was both poorly documented and structured. It took about seven years to standardize and document the code while simultaneously adding new features and improving overall product quality. When MSI acquired the product in 1996, it was pretty much in full compliance with the standards described above. Because I am the founder of MSI, these standards continue to be enforced.

It is important to note that as Migration RPG was standardized and enhanced, a quality assurance suite of test programs was developed to test all aspects and features of the product suite. The basic rules of test suite development were:

- Every product feature has one or more test programs.

- Every bug fix generates one or more test programs.

- Every new feature generates one or more test programs.

The test suite is maintained under DEC Test Manager (DTM). It is highly structured and fully automated. It is critical to maintaining the quality of Migration RPG. I cannot overemphasize how important having the test suite was to a quick and successful port.

Porting Plan

MSI had two advantages as it prepared to port Migration RPG to the HP Integrity server:

- We specialize in software migrations so we have a great deal of experience in porting software applications.

- We had successfully ported Migration RPG from the VAX processor to the Alpha processor back in 1996.

We use the following steps to plan and implement a software port:

1. Inventory the application and remove any modules that are no longer in use. It makes no sense to port unused modules.

2. Develop test scripts, data, reports, and an acceptance test plan. Our existing DTM test suite took care of this requirement.

3. Develop a schedule and identify resources for the port. Because we were stepping into somewhat uncharted waters by being an early Integrity server adopter, setting a schedule was difficult. Our past software porting experience, specific experience porting Migration RPG to Alpha, and research into the tools available under Intel Itanium-based OpenVMS led us to estimate a 2- 3 week porting effort.

4. Train the personnel. In this instance, being an OpenVMS shop was advantageous. OpenVMS is OpenVMS, regardless of the underlying platform. Likewise, Fortran, C and DTM changed very little in moving to the Integrity server. Much to our surprise and pleasure, Macro-32 was ported to the Integrity server as well. Thus, personnel training requirements were minimal.

5. Walk before you run. When ever possible, we always port a small, self-contained application before moving on to a large one. This allows us to test our porting plan, the porting tools, and the target environment without a large investment or risk. We chose our EBCDIC to ASCII conversion product, CVTFILE, to conduct the initial test port.

Critical to the success of our port was the HP implementation of the following products on the HP Integrity server:

- OpenVMS operating system

- Macro-32 compiler

- DTM

C and Fortran are not listed as critical because the modules they support could have been re-written, if necessary. Thus, failure of the port of those components would have been an inconvenience, not a show stopper.

The Porting Process

I initiated the porting process by attending an HP/Intel Porting Forum. This gave me access to Integrity server equipment and HP engineers. It also had me working among my peers, which proved to be a rewarding experience in its own right.

My first step was to recreate our development environment on the Integrity server. I had brought along all of the setup procedures used on our Alpha development system to accomplish this. Configuring the Integrity server was no more difficult that configuring a new AlphaServer.

CVTFILE Test Case

I began with the test port of the CVTFILE utility. The utility is a straight-forward program written in Macro-32 that can convert data between ASCII and EBCDIC representation. It contained no compiler directives. It compiled and linked cleanly using its build procedure and passed the conversion tests associated with the product. It required no code modification to port successfully. From start to finish, the CVTFILE port was completed within an hour.

Porting RPG: Moving a Compiler to the HP Integrity Server - Bruce Claremont
Encouraged by the ease of the CVTFILE port and the apparent maturity of the late beta releases of
OpenVMS and its layered products, I was ready to tackle porting Migration RPG.

Migration RPG Port

Before discussing the Integrity server port of Migration RPG, a little background on the VAX to Alpha
port is necessary. This information is especially pertinent if you are planning to port directly from a
VAX system to the Integrity server.

Prior to being ported to the Integrity server, Migration RPG ran on both VAX and Alpha processors
and was maintained via a common code base. In porting from the VAX to the Alpha, the only
changes we needed to make to the Macro source code involved formatting of the external routine
calls of modules in the shareable runtime image. We used compiler directives to accomplish this, as
illustrated in Figures 1 and 2.

Figure 1: Original VAX Call

```
.ENTRY S3X_ACCEPT,  ^M<R2,R3,R4,R5,R6,R7>
```

Figure 2: VAX/Alpha Common Code Call

```
.IF DEFINED VAX
        .ENTRY S3X_ACCEPT,  ^M<R2,R3,R4,R5,R6,R7>
.ENDC

.IF DEFINED ALPHA
S3X$ACCEPT::     .CALL_ENTRY -
                    MAX_ARGS=4, -
                    HOME_ARGS=TRUE, -
                    PRESERVE=<R2,R3,R4,R5,R6,R7>
.ENDC
```

Because of the changes in external call structures and link vectors, the VAX to Alpha port necessitated
creation of an Alpha-specific linker option file for the shareable image. This is one of only two places
where our VAX and Alpha code sets are not identical. The other is the compile and link qualifiers
used on the VAX and Alpha systems.

Obviously, being able to maintain a common code set across multiple platforms is desirable, so we
hoped to accomplish this with the port to the Integrity server. Much to my relief, I found the Macro-32
compiler for the Integrity server to be fully compatible with the Alpha version of our Macro-32 code.
Other than modifying the compiler directives to accommodate the Intel Itanium architecture, as shown
in Figure 3, *no code changes were needed* to achieve clean compiles on all of our Macro modules.
Likewise, our C modules only needed compiler directive modifications. Our Fortran modules
compiled without modification.

Figure 3: VAX/Alpha/Integrity Common Code Call

```
.IF DEFINED VAX
        .ENTRY S3X_ACCEPT,  ^M<R2,R3,R4,R5,R6,R7>
.ENDC

.IF NDF VAX
S3X$ACCEPT::     .CALL_ENTRY -
                    MAX_ARGS=4, -
                    HOME_ARGS=TRUE, -
                    PRESERVE=<R2,R3,R4,R5,R6,R7>
.ENDC
```

Because our code is standardized, modifying the compiler directives was a matter of running a
couple of global find and replace operations via TPU. Likewise, our build procedures were equally
easy to update. The Macro-32 compiler and linker on the Integrity server accept the same qualifiers
as the Alpha. Modification to our build procedures involved changing a few IF statements to
recognize the Intel Itanium architecture and default to the Alpha compile and link directives. Once

the compiler directives and build procedures were updated, Migration RPG compiled and linked successfully.

From start to finish, achieving clean compiles and links of Migration RPG on the Integrity servers was accomplished in under 4 hours. This includes the time spent modifying source code and procedures to work correctly under the new architecture.

Q/A Testing

Testing began with a few manual tests conducted during the Porting Forum. Initial testing quickly revealed a problem with floating point numbers. Had I reviewed *Porting Applications from HP OpenVMS Alpha to HP OpenVMS Industry Standard 64 for Integrity Servers* porting guide more carefully, this would not have been an issue. The C compiler on Alpha systems defaults to G_FLOAT representation. C on Integrity server systems defaults to IEEE_FLOAT. The addition of the /G_FLOAT qualifier to our C compile command line resolved the issue.

After returning to the home office and taking delivery of our own Integrity server, serious testing using the DTM test suite was initiated. Initial testing revealed a couple of minor coding errors in our own code. These problems had never been detected by the VAX and Alpha-based test suites, but showed up when run on the Integrity server. Hence, the porting process served to improve our product.

Several tests failed during the initial pass because they report back file sizes in blocks. The same file on VAX, Alpha, and Integrity server will vary in size due to the difference in underlying architecture, so the DTM tests were catching the differences between files sizing on the Alpha and Integrity server system. Updating the DTM benchmark files on the Integrity server eliminated this issue.

Having addressed the minor issues, we ran the test suite again. Out of several hundred tests, we had only two failures. The errors proved to be a fault in the Macro-32 compiler. We conducted our port under a field test version of OpenVMS, so this was not unexpected. HP has addressed the issue and our Q/A test suite now runs cleanly.

During the testing phase, we also encountered problems with the symbolic debugger. The problems were encountered during the porting forum and workarounds were quickly provided by HP engineering. The problems have been addressed in the production release of OpenVMS Version 8.2.

During the Q/A test phase, we had no problems with the DTM software. This was a tremendous relief and had a significant impact on the speed of our port.

Summary

Our entire port of Migration RPG was completed with about 60 man-hours of effort. Preparation and planning consumed another 20 man-hours. Credit for the ease and speed of our port is twofold:

1. HP did their job well. The smoothness of the port speaks well to the care HP has taken in moving OpenVMS and its layered products to the Integrity server.

2. MSI did its job well. Being specialists in software migrations, we have developed our products with an eye towards portability and ease of maintenance. While Migration RPG is a sophisticated, complex application, its individual modules are very structured and as simplistic as possible. Our code is highly standardized, making changes that impact several modules easy to accomplish. Our build procedures are automated, allowing ease of execution and quick resolution of compile and link problems. Our test suite is large, comprehensive, and automated, permitting thorough and efficient testing. Finally, we documented our code well, greatly simplifying code review and error resolution.

Recommendations

The steps needed to prepare for a port are pretty simple:

1. **Prepare:** Clean up source code, resolve known problems, and eliminate obsolete modules. In short, clean house.

2. **Plan:** Schedule the process; allocate human, software, and hardware resources; define and prepare test material; and test it. Be realistic with the schedule and be prepared to be flexible should your port have difficulties.

3. **Port:** Conduct the port. Review HP porting guides and modify code as required. Update build procedures and source code to accommodate the new architecture. Compile and fix problems. Link and fix problems. Conduct initial tests.

4. **Test:** Execute the acceptance test plan. Fix problems and run the tests again. Repeat until all problems are resolved.

5. **Celebrate:** Once you've ported and successfully tested your code, throw a party. You'll deserve it.

Depending upon the state of your code, documentation, and application knowledge, the porting process can run the gauntlet from simple to arduous. Clean code, careful planning, and quality testing will go a long way towards making the process simple and successful. Good luck!

For more Information

More information about MSI products and services can be found at http://www.MigrationSpecialties.com.