



How to: Porting Advisor for Graviton

Vishal Manan,
Sr. Graviton Specialist SA
AWS

Porting Advisor for Graviton

Porting Advisor for Graviton scans for potentially unsupported or non-portable arm64 code in source code trees

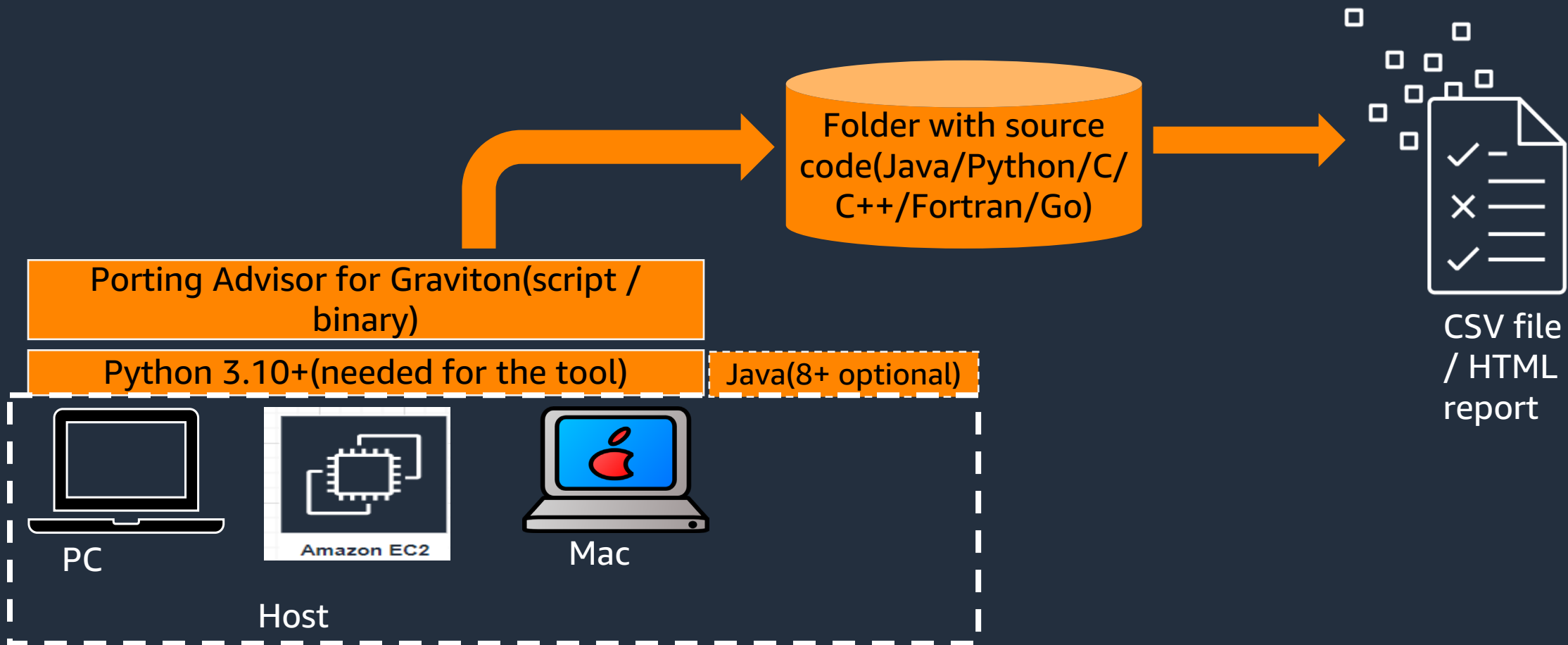
The tool only scans source code files and dependency files such as project/requirements.txt file(s)

Porting Advisor doesn't make any code modifications, API-level recommendations, or send data back to AWS

You can utilize Porting Advisor for Graviton either as a Python script or compile it into a binary and run on x86-64 or arm64 systems

Languages currently supported - C/C++, Fortran, Go 1.11+, Java 8+, Python 3+

High Level Working of Porting Advisor for Graviton



Skillset needed for using the tool

Users though should be versed in the following skills in order to take advantage of the recommendations the tool provides:

- An understanding of the build system – project requirements and dependencies, versioning, etc.
- Basic scripting language skills around Python, PowerShell/Bash
- Understanding hardware (inline assembly for C/C++) or compiler specific (intrinsic for C/C++) constructs when applicable

Kinds of issues detected

1. Inline assembly with no corresponding arm64 inline assembly
2. Assembly source files with no corresponding arm64 assembly source files
3. Missing arm64 architecture detection in autoconf config.guess scripts
4. Linking against libraries that aren't available on the arm64 architecture
5. Use architecture specific intrinsics
6. Preprocessor errors that trigger when compiling on arm64
7. Usages of old Visual C++ runtime (Windows specific)

What the tool doesn't report?

- Compiler specific code guarded by compiler specific pre-defined macros is detected, but not reported by default
- The following cross-compile specific issues are detected, but not reported by default:
 - Architecture detection that depends on the host rather than the target
 - Use of build artifacts in the build process
- The tool doesn't report exhaustively all issues

How are the python package versions tested ?

- PIP installs packages from PPI may require install-time builds
- AWS tracks over 200+ popular packages for compatibility [here](#)
- <https://geoffreyblake.github.io/arm64-python-wheel-tester/>
- Python specific recommendations can be found on [GitHub](#)
- <https://github.com/aws/aws-graviton-getting-started/blob/main/python.md>

Java

- Java is well supported and generally performant out-of-the-box on arm64
- Java JARs can include shared-objects that are architecture specific.
- Some customers haven't obtained the full performance benefit of Graviton until they have upgraded to Java 11 (Java 8 is the minimum version)
- Amazon Corretto JDK is the fastest way to get all of the performance enhancements AWS is making for Java

<https://github.com/aws/aws-graviton-getting-started/blob/main/java.md>

Building and Deploying

Native Compiling vs Emulated Compiling

- Wherever possible use native compiling
 - More performant and less prone to emulation-specific errors
- Docker buildx uses QEMU emulation
 - Will be significantly slower than native builds
 - May be okay for local development work
 - Use a CI/CD pipeline instead!

Demo

[Using Porting Advisor for Graviton | AWS Compute Blog \(amazon.com\)](https://aws.amazon.com/blogs/compute/using-porting-advisor-for-graviton/) - <https://aws.amazon.com/blogs/compute/using-porting-advisor-for-graviton/>

[GitHub - aws/porting-advisor-for-graviton](https://github.com/aws/porting-advisor-for-graviton) - <https://github.com/aws/porting-advisor-for-graviton/>

1. Building the tool

2. Running the tool as a binary and script

- On PC / EC2 Instance

3. Look at few different use cases:

- a) C/C++ Inline assembly files
- b) C/C++ Intrinsics
- c) C/C++ Large Cpp project
- d) Java code with JAR files



Graviton Porting Assistant v1.0.0



Project Information

Project:
inline_assembly

Source root:
..\..\cppCode\inline_assembly

Report Date:
2023-01-19 16:20:03

Results

	File	Line #	Comments
			1 files scanned.
	..\..\cppCode\inline_assembly\mc++.cpp	82	architecture-specific inline assembly



Final words

Porting Advisor for Graviton helps customers quantify the amount of work that is required to port an application

Accelerates your ability to transition to Graviton-based Amazon EC2 instances by reducing the iterative process of identifying and resolving source code and library dependencies

If you run into any issues, then see our CONTRIBUTING link -

<https://github.com/aws/porting-advisor-for-graviton/blob/main/CONTRIBUTING.md#reporting-bugsfeature-requests>

Using Porting Advisor for Graviton | AWS Compute Blog (amazon.com) -

<https://aws.amazon.com/blogs/compute/using-porting-advisor-for-graviton/>

GitHub - aws/porting-advisor-for-graviton - <https://github.com/aws/porting-advisor-for-graviton/>



Helpful Links

[Porting Intel Intrinsics to Arm Neon Intrinsics](#)

[Implementations of SIMD instruction sets for systems which don't natively support them.](#)

Thank you!

