Stony Brook University

COMPAS

One build system to rule them all: the return of the meson

Sergey Madaminov (Stony Brook University)

William Tu (VMWare, Inc.)

December 7, 2021

Introduction

- Open vSwitch supports multiple operating systems
  - Linux
  - FreeBSD
  - Windows
  - and more
- Build process is mostly straightforward but not quite
  - autotools suite has no native support for Windows
  - MSYS2+MinGW for Windows
  - Top-level Makefile has 7343 LoC
  - All Makefiles comprise 8263 LoC

Agenda

- Introduction
- Motivation
- New build system
- Current progress
- Future work
- Conclusion

Why OvS Needs New Build System?

- Original motivation
  - Attempts to port OvS-DPDK to Windows
- Why care about Windows?
  - Widely used OS in the enterprise world
  - DPDK may help with the lack of Win/Linux kernel knowledge
  - Found autotools to be a limiting factor
- Thus, there are two main reasons
  - Better experience within *nix family
  - Better experience outside *nix family

Why OvS Needs New Build System?

- What is wrong with autotools?
  - Steep learning curve => harder to contribute
  - Humongous Makefiles => harder to debug
- What is wrong with MSYS2+MinGW?
  - Indirection layer => more sources of errors
  - Additional dependency => hampers adoption
  - Separate code to maintain (cccl wrapper)
  - Slow build process

Testbench

- Intel Core i7-8665U @ 1.90GHz
  - Single-socket, 4 physical cores with Hyper-Threading
- 16 GB RAM
- OS: Windows 11
- WSL2 with RHEL8
- MSYS2: MSYS_NT-10.0-22000 3.2.0-340.x86_64
- Developer Command Prompt for VS 2019
- Compiler: Clang 12.0.1, Meson: 0.59, Ninja: 1.10
- OVS: 91e1ff5dde396fbcc8623ac0726066e970e6de15

RHEL8 WSL With Autotools

Command	Time reported by /usr/bin/time
./boot.sh	8 seconds
./configure CC=clang CFLAGS="-O2"	23 seconds
make	3 minutes 10 seconds
make –j2	2 minutes
make –j4	1 minute 25 seconds
make –j8	1 minute 11 seconds

RHEL8 WSL With Meson

- Ninja performs parallel build by default

Command	Time reported by /usr/bin/time
meson build (think ./configure)	7 seconds
ninja –C build	58 seconds

MSYS2+MinGW With Autotools

Command	Time reported by /usr/bin/time
./boot.sh	35 seconds
./configure with Windows-specific flags	2 minutes 13 seconds
make –j4	2 hours 42 minutes 28 seconds

Not yet clear what causes such long compilation

MSYS2+MinGW With Meson

- Ninja performs parallel build by default

Command	Time reported by /usr/bin/time
meson build (think ./configure)	4 seconds
ninja –C build	1 minute 42 seconds

Windows Native Build With Meson

- Developer Command Prompt for VS 2019
- Ninja performs parallel build by default

Command	Time reported by ptime
meson build (think ./configure)	3.9 seconds
ninja –C build	1 minute 23 seconds

No reason to keep using MSYS!

Agenda

- Introduction
- Motivation
- New build system
- Current progress
- Future work
- Conclusion

How To Choose Your Build System

- Identify what is that we want from the build system
  - Native support for both Linux and Windows
  - Easy to read and write build files
  - Interaction with external projects and libraries
  - Efficient
- What others have done?
  - DPDK transitioned to the meson build system

"Those who cannot remember the past are condemned to repeat it,"
Spanish philosopher George Santayana

Picking New Build System

Build system	Linux	Windows	Ease of use	External projects	Efficient
Make	Yes	Yes	No	Yes	No
autotools	Yes	No	No	Yes	No
CMake	Yes	Yes	Yes	Yes	Yes
VS solution	No	Yes	Yes	Yes	Yes
SCons	Yes	Yes	Yes	Yes	No
Bazel	Yes	No	Yes	No	Yes
Meson	Yes	Yes	Yes	Yes	Yes

Breaking a tie: use the same build system as DPDK

Meson Build System

- The only prerequisite is Python3.6+
- Declarative build system
- Build files are easy to read, write, and understand
- Simple dependency handling
- It is fast!
- Gaining adoption
  - DPDK, Qemu, libvirt, Rizin, and many more!

Meson Build Files

- Write down your intentions in *meson.build* files
- Familiar tree-like nested hierarchy
  - Start with top-level meson.build file
  - Consecutively call meson.build files from subfolders
- Same (almost) commands for Linux and Windows

```
$ meson build
$ ninja -C build
```

What Is Done And What Is Left

- Can build OvS on both Linux and Windows
- bash scripts are re-written in Python

- Port remaining checks and features
- Fix hundreds of compilation warnings
  - requires changes to both Linux and Windows code
- Tests
  - OvS uses *autotest* framework, which is for Linux
  - Exploring *avocado* as a potential replacement

Concluding Remarks

- Autotools limits OvS expansion to Windows
- Meson eliminates the root cause for that issue
- Furthermore, it is fast(er)!
- RFC was published on the mailing list

- Contributions are welcome!
  - Try it out, comment, submit patches, and open issues
  - https://github.com/smadaminov/ovs-dpdk-meson-issues