Unchain Your Toolchains with CROPS (CROssPlatformS)

Todor Minchev
todor.minchev@linux.intel.com
Intel Open Source Technology Center

Embedded Linux Conference
San Diego, 5 April 2016
Agenda

• Traditional Cross-Platform Development Workflow
• CROPS Definition & Value
• Native Tools vs CROPS Containers
• Technical Overview
• Current Status
• Future Plans & Challenges
• Demo
• Q & A
Traditional cross-platform development workflow

 GCC
GDB  The GNU Project Debugger
Traditional cross-platform development workflow on multiple host platforms

- Linux
- Windows
- Mac OS X

GCC

GDB
The GNU Project Debugger

GDB
The GNU Project Debugger

GDB
The GNU Project Debugger
Scalability of the traditional cross-platform development model

<table>
<thead>
<tr>
<th>DEVELOPMENT HOST (N)</th>
<th>SDK (N x M)</th>
<th>TARGET PLATFORMS (M)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>X86</td>
<td>X86</td>
</tr>
<tr>
<td></td>
<td>ARM</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MIPS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>OTHERS</td>
<td></td>
</tr>
<tr>
<td>Windows</td>
<td>X86</td>
<td>ARM</td>
</tr>
<tr>
<td></td>
<td>ARM</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MIPS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>OTHERS</td>
<td>MIPS</td>
</tr>
<tr>
<td>Mac OS X</td>
<td>X86</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ARM</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MIPS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>OTHERS</td>
<td></td>
</tr>
</tbody>
</table>
AND THIS IS HOW WE BUILD CROSS-PLATFORM TOOLCHAINS FOR WINDOWS, LINUX AND MAC OS X DEVELOPERS
What is CROPS?

CROPS is an open source, cross-platform development framework that leverages Docker containers to provide an easily managed, extensible environment which allows developers to build binaries for a variety of architectures and use native Linux tools on Windows, Mac OS X and Linux hosts.
What Value does CROPS provide?

CROPS provides the following capabilities:

• a solution to allow cross building for different targets from Windows, Mac, & Linux hosts
• the ability to leverage Linux based tools in addition to the cross compiler e.g. bitbake, image creator, kernel menuconfig, perf, oprofile
• a path to embrace the cloud as part of the solution
• an alternative to a full Linux VM
• easy toolchain distribution and updates
• a clean, reproducible state for development and testing
Native Solution

Port the Linux toolchains for all the desired architectures to each of the desired hosts

<table>
<thead>
<tr>
<th>PROS</th>
<th>CONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>No added complexity from containers</td>
<td>Need to qualify N toolchains on M hosts</td>
</tr>
<tr>
<td>Based on well known technologies</td>
<td>Windows represents several hosts (7,8,8.1,10, future…)</td>
</tr>
<tr>
<td>Meets user expectation for a toolchain app</td>
<td>Toolchain updates are likely monolithic, making rollbacks harder</td>
</tr>
<tr>
<td></td>
<td>Easily polluted by host environment</td>
</tr>
<tr>
<td></td>
<td>Remote builds infeasible</td>
</tr>
</tbody>
</table>
Containers Solution

Leverage containers to solve problem needs while relying on the host for the graphical IDE

<table>
<thead>
<tr>
<th>PROS</th>
<th>CONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Need to qualify N toolchains on only 1 container</td>
<td>Containers add additional level of complexity (largely hidden)</td>
</tr>
<tr>
<td>Easy toolchain distribution, updates, and rollbacks</td>
<td>Based on forward-looking, state of the art technologies</td>
</tr>
<tr>
<td>Isolated from host environment</td>
<td></td>
</tr>
<tr>
<td>Supports remote builds</td>
<td></td>
</tr>
<tr>
<td>Active community support enabling containers on different hosts</td>
<td></td>
</tr>
</tbody>
</table>

The primary advantage of containers is that we concentrate on the API and what is inside the container while others manage the issue of making it work on diverse platforms.
## Scalability of the CROPS development model

<table>
<thead>
<tr>
<th>DEVELOPMENT HOST (N)</th>
<th>CROPS SDK(M)</th>
<th>TARGET PLATFORMS (M)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows</td>
<td>X86</td>
<td>X86</td>
</tr>
<tr>
<td></td>
<td>ARM</td>
<td>ARM</td>
</tr>
<tr>
<td></td>
<td>MIPS</td>
<td>MIPS</td>
</tr>
<tr>
<td>Mac OS X</td>
<td>OTHERS</td>
<td>OTHERS</td>
</tr>
</tbody>
</table>
What the framework allows

**Development Host**
- Windows, Linux, and Mac OS C Native Eclipse Plugin
- Windows Visual Studio Plugin
- XCode Plugin

**API**
- Request toolchain
- Build project

**REGISTER**

**Toolchains**
- X86 Cross Compiler Container
- Yocto Project Extensible SDK Container
- Zephyr SDK Container
- ARM Cross Compiler Container

**Source code and meta data**

**Shared Host Folder**

**Cloud Share**

...... OR ......
How CROPS works

Development Host
(Windows, Linux, Mac OS X)

- Host Build Manager
- IDE (Eclipse, VisualStudio, XCode)

Build Device Container

- Build Device Agent
- Toolchain
- Source Code

Dispatcher

SERVICE
REQUEST
REGISTER
OUTPUT
COMPILER

TARGET
MinnowBoard, Edison, Galileo, etc

Shared Host Folder

...... OR ......

Cloud Share
Current Status – Project Home (https://git.yoctoproject.org/cgit/cgit.cgi/crops/)

<table>
<thead>
<tr>
<th>Branch</th>
<th>Commit message</th>
</tr>
</thead>
<tbody>
<tr>
<td>master</td>
<td>installers: add gdb wrapper script to Zephyr installer</td>
</tr>
<tr>
<td>10 min.</td>
<td>installers: add gdb wrapper script</td>
</tr>
<tr>
<td>10 min.</td>
<td>README: Update README</td>
</tr>
<tr>
<td>10 min.</td>
<td>installer: moved platform check to top of script so we only do it once</td>
</tr>
<tr>
<td>10 min.</td>
<td>installer: added a check for docker installation</td>
</tr>
<tr>
<td>10 min.</td>
<td>installer: install ced for mac</td>
</tr>
<tr>
<td>10 min.</td>
<td>installer: pass ced ip address to ced on Windows and Mac</td>
</tr>
<tr>
<td>2016-03-01</td>
<td>installer: add a universal ced installer for Zephyr builds</td>
</tr>
<tr>
<td>2016-02-26</td>
<td>travis: force docker version to be 1.10</td>
</tr>
<tr>
<td>2016-02-26</td>
<td>travis: add docker version so we know what we are running on</td>
</tr>
</tbody>
</table>

Clone
    git@git.yoctoproject.org:crops
    http://git.yoctoproject.org/git/crops

generated by cgit v0.16.2 at 2016-03-18 16:59:34 (GMT)
Current Status - GitHub mirror with wiki (https://github.com/todorez/crops)

1. Install Docker
   - Windows users can download the standard Docker Toolbox installer package for Windows.
   - These are the choices you need to have selected when you do the Docker Toolbox installation on Windows.

   ![Setup - Docker Toolbox](image)

   - Mac users can download the standard Docker Toolbox installer package for Mac OS X.

2. Run Zephyr Installer
   - Now you need to run the Zephyr installer from the Docker Quickstart Terminal. From within the Docker Quickstart Terminal do:
Current Status – DockerHub (https://hub.docker.com/r/crops/)
Current Status - CLI

```
THE CROPS ENVIRONMENT HAS BEEN SET UP

Initialize Zephyr environment for CLI use
Example:
/c/Users/tminchev.GER/.crops/ceed/ceed -i 192.168.99.100 -d crops-zephyr-0-7-2-src -g "git clone --branch v1.0.0 /zephyr-src /crops/zephyr-project/"

You can now build Zephyr applications from the CLI
Example:
/c/Users/tminchev.GER/.crops/make.zephyr BOARD=arduino_101 -C /crops/zephyr-project/samples/nanokernel/apps/hello_world/
```

```
Current Status – Eclipse IDE plug-in
Future Plans

- **RESTful API**
  - Current framework uses Internet sockets
  - Firewalls allow only well known ports through (e.g. 80, 443)

- **Remote Toolchain/Projects Support**
  - Host toolchain containers remotely
  - Store project workspaces remotely
  - Share toolchains

- **Toolchain descriptors**
  - Describe toolchain capabilities
  - Supported architectures
  - Default compiler flags
Future Plans

• Dynamic Eclipse IDE UI
  • Provide different UI perspectives based on toolchain capabilities
  • RTOS builds vs userspace application builds
Challenges

• **Remote Projects Support**
  • File synchronization
  • Depends on Internet connectivity
  • Binary File Diffs

• **Debugging from Eclipse on Windows**
  • Pseudo terminals on Windows
DEMO
Q & A
CROPS Team

Todor Minchev
todor.minchev@linux.intel.com

Brian Avery
brian.avery@intel.com

Tim Orling
timothy.t.orling@intel.com
Legal Disclaimer

INFORMATION IN THIS DOCUMENT IS PROVIDED “AS IS”. NO LICENSE, EXPRESS OR IMPLIED, BY
ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS
DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS
OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING LIABILITY OR
WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR
INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

• Other names and brands may be claimed as the property of others.

Copyright © 2016 Intel Corporation.