An Overview of the Hash Equivalence & PR Services

Paul Barker, SanCloud Ltd

Yocto Project Summit, 2021.11

About me

- Involved in Yocto Project since 2013
- Work across the whole embedded stack
- Principal Engineer @ SanCloud Ltd
- https://sancloud.co.uk/

SanCloud

Contact details

- Email: paul@pbarker.dev
- Web: https://pbarker.dev/
- Twitter: @pbarker_dev

Acknowledgements

- Several improvements described here were implemented while I worked at Konsulko Group

- This work was funded by the Automotive Grade Linux (AGL) project

Acknowledgements

- Joshua Watt
- Scott Murray

About this talk

- PR Service
- Hash Equivalence Service
- Recent developments
- Future work

PR Service (prserv)

PR Service: The problem

- To support on device updates (rpm/deb/ipk), package revisions need to increment each time a package is rebuilt
- Manually modifying PR for every change is inefficient and prone to error

PR Service: The solution

- Maintain a database of input hashes for each recipe
- Each input hash is mapped to a PR value
- When a new input hash is added, assign a PR value one higher than the maximum PR value previously used

Using the PR Service (auto start)

- In local.conf.sample.extended:

```
# The network based PR service host and port
# Uncomment the following lines to enable PRservice.
# Set PRSERV_HOST to 'localhost:0' to automatically
# start local PRService.
# Set to other values to use remote PRService.
#PRSERV_HOST = "localhost:0"
```

Using the PR Service (manual start)

- **Start and stop standalone PR service**
  - `bitbake-prserv --host $IP --port $PORT --start`
  - `bitbake-prserv --host $IP --port $PORT --stop`
- **Set PRSERV_HOST in your local or distro config**
  - `PRSERV_HOST = "$IP:$PORT"`

Exporting PR Service data

- Run `bitbake-prserv-tool export $FILE`
- Exports data to a .inc file

Importing PR Service data

- Run `bitbake-prserv-tool import $FILE`
- Restores the data in the given .inc file

Hash Equivalence Service (hashserv)

Hash Equivalence: The problem

- Tasks are always re-executed when their dependencies have changed
- Some dependency changes have no impact and result in unnecessary rebuilds
- Manually tracking this with SIGGEN_EXCLUDE_SAFE_RECIPE_DEPS and/or SIGGEN_EXCLUDERECIPES_ABISAFE is inefficient and prone to error

Hash Equivalence: The solution

- Maintain a database of input and output hashes for each sstate task
- If a new input hash results in an output hash that has already been seen for this task, mark the corresponding input hashes as equivalent
- Re-compute input hashes for dependent tasks using the matched input hash

Hash Equivalence: More information

- See the presentation “Reproducible Builds and Hash Equivalence in the Yocto Project” by Joshua Watt

  Embedded Linux Conference 2020

  https://www.youtube.com/watch?v=zXEdqGS62Wc

  https://elinux.org/images/3/37/Hash_Equivalence_and_Reproducible_Builds.pdf

Using Hashserv (auto start)

```
#
# Hash Equivalence
#
# Enable support for automatically running a local hash equivalence server and
# instruct bitbake to use a hash equivalence aware signature generator. Hash
# equivalence improves reuse of sstate by detecting when a given sstate
# artifact can be reused as equivalent, even if the current task hash doesn't
# match the one that generated the artifact.
#
# A shared hash equivalent server can be set with "<HOSTNAME>:<PORT>" format
#
#BB_HASHSERVE = "auto"
#BB_SIGNATURE_HANDLER = "OEEquivHash"
```

Using Hashserv (manual start)

- **Start standalone Hash Equivalence service**
  - `bitbake-hashserv`
    - `-b 'unix://./socket/path.sock'`
    - `-b 'host:port'`
- **Set BB_HASHSERV in your local or distro config**
  - `BB_HASHSERV = "unix://./socket/path.sock"`
  - `BB_HASHSERV = "host:port"`

A brief aside: Publishing sstate

- **Sharing a hash equivalence database only makes sense when also sharing sstate**

- **sstate can be served via a local directory, NFS share or HTTP(S)**
  - Local directory or NFS share can be read/write
  - HTTP(S) is always read-only

Recent Improvements

Recent Improvements: Under the hood

- Replaced old xmlrpc in prserv with modern json rpc
- Switched hashserv & prserv to async code
- Improved testing
- Improved sstate re-use in hashserv

Recent Improvements: PR Service read-only mode

- Pass -r argument when manually starting PR service
- Prevents database modification, new PR values are not inserted into the database
- Useful when you want to share PR values from an autobuilder

Recent Improvements: Hashserv read-only mode

- Pass -r argument when manually starting hashserv
- Operations which would modify the database are rejected
- Useful when sharing hash equivalence data & sstate from an autobuilder

Recent Improvements: Hashserv upstream

- **Pass `-u "host:port"` or `-u "unix://./socket/path.sock"` arguments when manually starting hashserv**
- **Allows connection of a local read-write hashserv instance to an upstream read-only hashserv instance**
- **Still allows hash equivalences to be detected in local sstate**
  - Connecting only to the read-only instance would not allow local hash equivalences to be added to a database

Future Work

Future Work: Sharing PR server effectively

- **Upstream connection is currently only implemented for hashserv**
- **You can point directly at a read-only prserv instance but then PR values may not be correctly incremented for local changes**
- **Builds from different users or machines may end up with the same PR values even if the input hashes are different**
  - May cause issues when working in a team

Future Work: Hashserv import & export

- Data export and import are currently only implemented for prserv
- Adding these features for hashserv would make it easier to backup & restore or migrate data

Future work: Better introspection tools

- **Command line tools to query prserv & hashserv databases**
  - E.g. check highest PR value allocated for a recipe
- **Collect stats in PR service**
  - Hashserv already does this

Future work: Better test coverage

- **Stress testing**
  - Hashserv has a command for this, would be good to add to prserv as well
  - Could also add testing over a network
- **Better testing of service start & stop**
- **prserv unit tests in bitbake**

Summary

- PR service supports on-device updates by incrementing package revisions each time a recipe is built
- Hash Equivalence service improves sstate reuse by allowing insignificant dependency changes to be ignored
- Both services have recent improvements and opportunities for future work

yocto PROJECT | THE LINUX FOUNDATION