What Does it Take to Run LLVM Buildbots?

David Spickett, Staff Software Engineer, Arm
Who is Linaro?

- Linaro works to ensure open source projects are best on Arm.
- I am assigned from Arm to Linaro’s Toolchain Working Group (TCWG).
- TCWG works on LLVM, GCC and QEMU.

We care about the quality of LLVM.
Bots All The Way Down...

- **LLVM pre-commit checks**
- **Libcxx pre-commit checks**

**Pre-commit**

- **LLVM Buildbots**
- **Software builds (Chrome, ClangBuiltLinux)**

**Post commit**

- **Downstream toolchains (Arm Compiler for Embedded)**
- **Language projects (Zig, Rust)**
- **Out of Tree LLVM Projects (CIRCT, Polygeist)**
- **Green Dragon**
What is a Buildbot?

- Post commit verification of changes.
- Build anything from all to just one project.
- Emails when your commit was in a failed build.

Buildbot failure in LLVM Buildbot on lldebug-aarch64-ubuntu

@llvm.buildmaster@lab.llvm.org
Fri, 23 Sept, 13:48 (6 days ago)

The Buildbot has detected a failed build on builder lldebug-aarch64-ubuntu while building lldebug.
LLVM Buildbots

Run by Galina Kistanova.

Buildmaster → Builder → Worker
Buildmaster → Builder → Worker
Buildmaster → Builder → Worker

Build configs e.g. “clang-armv7-vfpv3-full-2stage”

Community provided machines.

~162 Builders for LLVM
LLVM Commits: The Numbers
(from January 1st 2021 to January 1st 2022)

32810 commits*
~90 commits a day
~4 commits an hour

● Buildbots batch commits.
● Many are rarely idle.

* includes 1617 reverts and relands
## Linaro and Buildbots

- First buildbot added 2013
- 29 currently

<table>
<thead>
<tr>
<th>Buildbot</th>
<th>Status 1</th>
<th>Status 2</th>
<th>Status 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>clang-armv7-2stage</td>
<td>8724</td>
<td></td>
<td></td>
</tr>
<tr>
<td>clang-arm64-windows-msvc</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>clang-armv7-global-isel</td>
<td>8595</td>
<td></td>
<td></td>
</tr>
<tr>
<td>clang-armv7-int</td>
<td>13008</td>
<td>13007</td>
<td></td>
</tr>
<tr>
<td>clang-armv7-quick</td>
<td>19961</td>
<td>19960</td>
<td>19959</td>
</tr>
<tr>
<td>clang-armv7-vfpv3-2stage</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>clang-armv7-vfpv3-full-2stage</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>clang-armv8-ld-2stage</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>clang-native-arm-int-perf</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>llb-arm-ubuntu</td>
<td>27166</td>
<td>27167</td>
<td>27166</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Buildbot</th>
<th>Status 1</th>
<th>Status 2</th>
<th>Status 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>clang-aarch64-full-2stage</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>clang-aarch64-global-isel</td>
<td>7290</td>
<td></td>
<td></td>
</tr>
<tr>
<td>clang-aarch64-lld-2stage</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>clang-aarch64-quick</td>
<td>19411</td>
<td>19410</td>
<td></td>
</tr>
<tr>
<td>clang-aarch64-sve-vla</td>
<td>2735</td>
<td></td>
<td></td>
</tr>
<tr>
<td>clang-aarch64-sve-vla-2stage</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>clang-aarch64-sve-vls</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>clang-aarch64-sve-vls-2stage</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>flang-aarch64-debug</td>
<td>7047</td>
<td>7046</td>
<td></td>
</tr>
<tr>
<td>flang-aarch64-dylib</td>
<td>8696</td>
<td>8695</td>
<td>8694</td>
</tr>
<tr>
<td>flang-aarch64-latest-clang</td>
<td>7817</td>
<td>7816</td>
<td></td>
</tr>
<tr>
<td>flang-aarch64-latest-gcc</td>
<td>10629</td>
<td>10628</td>
<td>10627</td>
</tr>
<tr>
<td>flang-aarch64-out-of-tree</td>
<td>17493</td>
<td>17492</td>
<td></td>
</tr>
<tr>
<td>flang-aarch64-rei-assert</td>
<td>8717</td>
<td>8710</td>
<td>8709</td>
</tr>
<tr>
<td>flang-aarch64-release</td>
<td>8665</td>
<td>8664</td>
<td>8663</td>
</tr>
<tr>
<td>flang-aarch64-sharedlibs</td>
<td>8631</td>
<td>8630</td>
<td></td>
</tr>
</tbody>
</table>
Why So Many?

Architectures
- Armv7
- AArch32
- AArch64

Vector Extensions
- VFPv3
- NEON
- SVE

Platforms
- AArch64 Linux
- Arm Linux
- Windows on Arm

Compilers
- Clang
- GCC

Instruction Selection
- SelectionDAG
- Globallsel

Projects
- LLVM
- Clang
- LLDB
- Flang

Build Type
- 1 stage
- 2 stage
- Debug
- Shared libraries

Many testing dimensions.
Where Does All This Run?

- 2 Ampere Mt. Jade servers
- 2 Surface Pro X laptops
- Several Nvidia Jetson TK1s
- Fujitsu FX700 (for scalable vectors)

- ~400 cores
- ~800GB of RAM

- >1 worker per machine where possible.
Resource Allocation is Difficult
Resource Allocation

● You need to do these things, in parallel:
  ○ Run the bot
  ○ Triage build issues
  ○ Work on fixes

● Fixed resources instead of dynamic allocation.
  ○ Swings in allocation cause inconsistent tests.
  ○ “flaky buildbot”

● But - watch out for excessive idle time.
Graphs!

Builder Time Usage

<table>
<thead>
<tr>
<th></th>
<th>clang-quick</th>
<th>flang-release</th>
<th>clang-2stage</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Idle</strong></td>
<td>40.00%</td>
<td>60.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td><strong>Building</strong></td>
<td>60.00%</td>
<td>40.00%</td>
<td>100.00%</td>
</tr>
</tbody>
</table>
We heard you like timeouts so we timed out checking that you were able to check for a timeout.
LLVM Committer vs Bot Maintainer

- They see 1 build and only if it fails.
- You see every build.
Maintainer vs Committer Perspective

Committers are:
  ● Seeing only 1 build out of 100s.
  ● Unaware that you existed until now.
  ● Unfamiliar with your target.
  ● Less incentivised than you to fix the issue.
  ● Unable to access your hardware.

These are not good or bad in themselves.
The Maintainer Approach

- Remember that failure emails come out of the blue.
- Inform without making assumptions.
- Proactively notify committers.
- 1 flaky build == a flaky bot
- Be ready to work with the committer.
- Be ready for you to do the fix instead.
Monitoring

- Add yourself to email notifications.
- Use the builder page.
- Build a status page using the API [0]
- Rotate monitoring duty across your team/community.
  - It's not all downside, I promise.

Triage

- Embrace the power of knowing nothing
  - Find the change first
  - “why” comes later

- Know your categories.
  - Are all <architecture> bots broken?
  - Use the “console” view.

- Find the common changes.

- Bisect all the things!
Reverts

“Remember, it is normal and healthy to have patches reverted.”

This policy is great but not fully embraced.

- Live by it and set the example.
- Repeat it at every opportunity.
- Make reverts less surprising.

https://llvm.org/docs/DeveloperPolicy.html#patch-reversion-policy
The Bad, The Good And The Future
Buildbot: The Bad

● Only one repository in the change list.
  ○ llvm-project + llvm-test-suite, you only see llvm-project changes.

● Config changes need a buildmaster restart.
  ○ Requesting one is easy (thanks to Galina) but there is still a delay.

● Every builder builds every commit - even if it’s known incorrect.
  ○ Bad for low availability workers.
Buildbot: The Good

- The patience of the LLVM community.
- Bisecting a monorepo is 1000x easier than llvm + clang.
- The web interface is clean and functional.
  - Easy to go from change list to github to Phabricator review.
Buildbot: The Future

Short term - put the basic builds in pre-commit

- Catch the obvious issues early.
- Phabricator is doing some of this, with difficulty.
Buildbot: The Future

Long term - move **all** bots to pre-commit.

- No surprises for comitters.
- Rust’s “main is always green”. [0]
- Libcxx is a success story. [1]

The big question - what is the cost multiplier?

How many more builds in pre vs. post commit?

[0] https://github.com/rust-lang/homu
[1] https://www.youtube.com/watch?v=B7gB6van7Bw
Thank you

Extra thanks to:
Linaro TCWG Team
Galina Kistanova
Everyone clicking “mark as read” on our buildbot emails :)}
Backup: Cost in Engineering Time

- 4 team members on a weekly rota.
- ~1 day of the week spent on triage.
- Spikes to multiple days for big changes (e.g. opaque pointers).
Backup: Cost estimate

To run:
- 2 1-stage AArch64 SVE bots
- 2 2-stage AArch64 SVE bots
- Some details removed to fit on slide.
- On AWS Gravtion 3.
- Other clouds are available.

<table>
<thead>
<tr>
<th>Cores per bot</th>
<th>8</th>
<th>16</th>
<th>32</th>
<th>64</th>
</tr>
</thead>
<tbody>
<tr>
<td>Worst case build time 1 stage</td>
<td>01:42:58</td>
<td>00:55:19</td>
<td>00:33:15</td>
<td>00:25:14</td>
</tr>
<tr>
<td>time saved</td>
<td>0</td>
<td>00:47:39</td>
<td>00:22:04</td>
<td>00:08:01</td>
</tr>
<tr>
<td>% time saved</td>
<td>0.00%</td>
<td>46.28%</td>
<td>39.89%</td>
<td>24.11%</td>
</tr>
<tr>
<td>Worst case build time 2 stage</td>
<td>03:05:31</td>
<td>01:38:56</td>
<td>00:57:09</td>
<td>00:39:32</td>
</tr>
<tr>
<td>time saved</td>
<td>0</td>
<td>01:26:35</td>
<td>00:41:47</td>
<td>00:17:37</td>
</tr>
<tr>
<td>% time saved</td>
<td>0.00%</td>
<td>46.67%</td>
<td>42.23%</td>
<td>30.83%</td>
</tr>
</tbody>
</table>

Cost per year:
- $7,923
- $14,842
- $30,515
- $60,485
Backup: What Do Bot Names Mean?

- Names are arbitrary but there are some patterns.
  
  **clang-arm64-windows-msvc**
  
  Building clang, on AArch64 Windows on Arm, using clang-cl.
  
  (yes, the msvc is confusing)

- **lldb-aarch64-ubuntu**
  
  Building lldb on AArch64 Ubuntu using clang.
  
  (you'd think it would have “-clang” on the end)

- For the full configuration:
  
  
  - Check the cmake stage logs from the buildbot web UI.
Backup: More Future

- Can we learn from the Linux Kernel?
  - Many configs, some more popular than others.

- Github bot to explain revert and reland process.
  - Prevent surprises and a feeling of being singled out.
  - Extend to the whole “lifecycle” of a change?

- Trigger bots from a pre-commit pull request
  - Reviewers can catch failures in review.
  - Test without hardware access.

- Automatic bisect on failure
  - Several non-buildbot systems do this.
  - Send to bot owner first in case of bad results.