Breaking down the BitBake build on the process level

Amir Kirsh

Yocto Project Summit, 2022-05

https://bit.ly/YPS-2022_IB_bitbake

About me

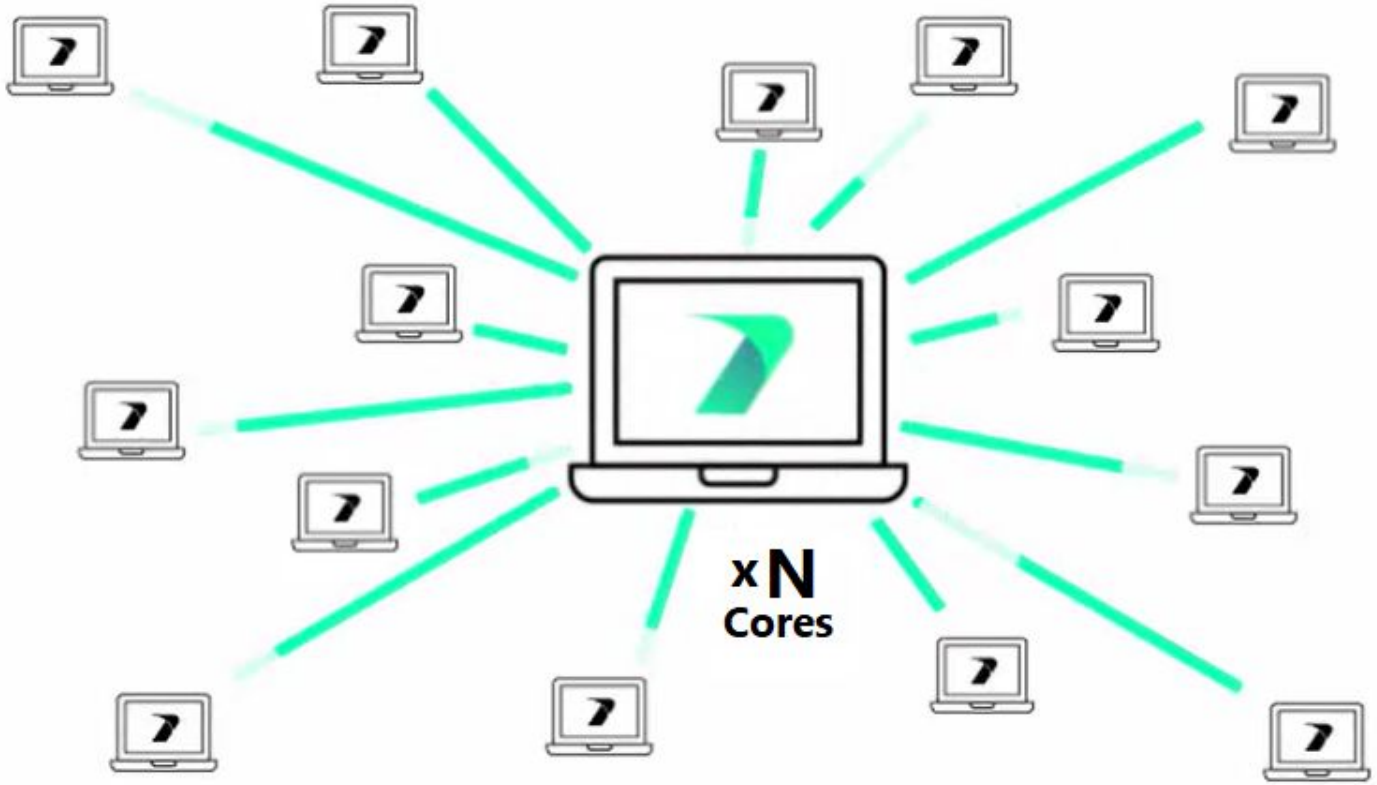
Developer Advocate at

INCREDIBUILD

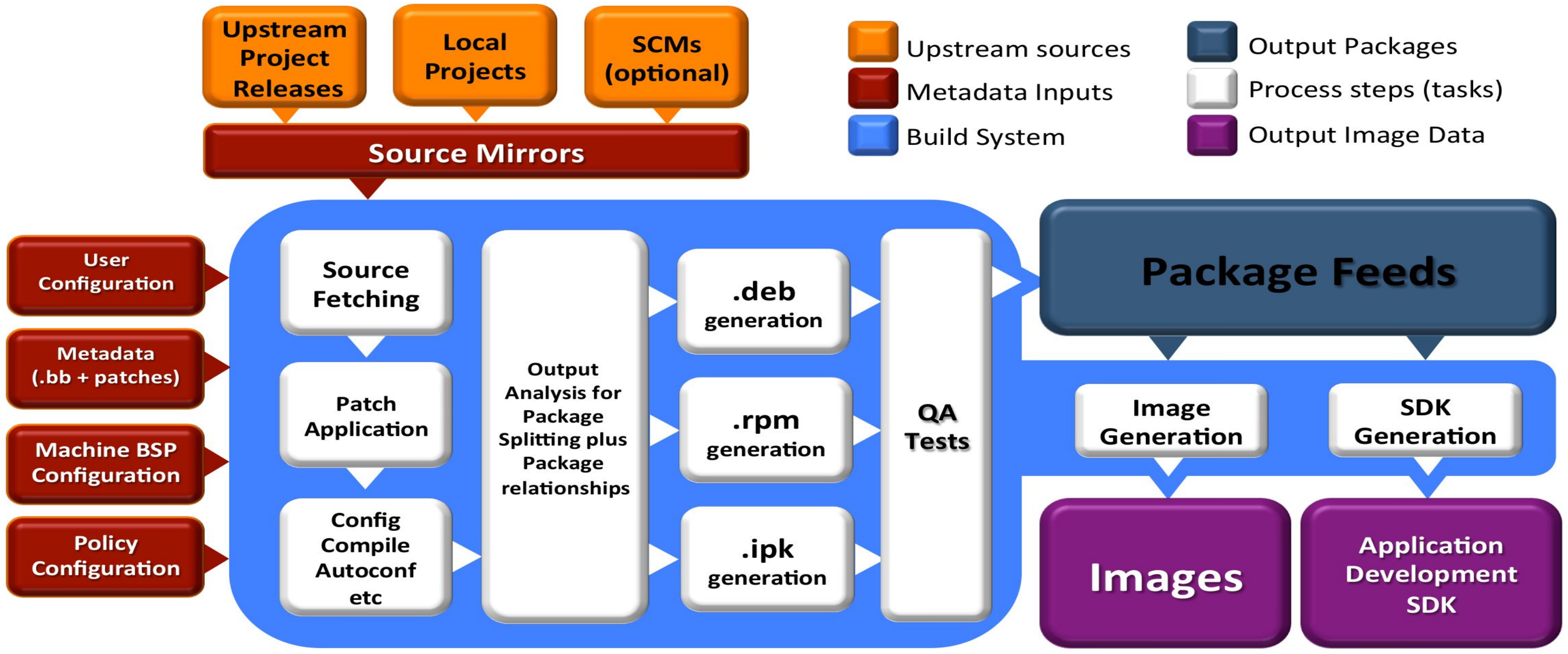
Lecturer
Academic College of Tel-Aviv-Yaffo
Tel-Aviv University

Member of the Israeli ISO C++ NB

Co-Organizer of the **CoreCpp** conference and meetup group


xN
Cores

Build System Workflow



Slide shamelessly stolen from: http://bit.ly/YPS202105Intro

A FILM FROM DANIELS
EVERYTHING
EVERYWHERE
ALL AT ONCE
天馬行空
ORANGE FIZZ

First steps for a more efficient bitbake build

First steps for a more efficient bitbake build (1)

```
$ bitbake -k
```

```
-k, --continue    Continue as much as possible after an error. While the target
                  that failed and anything depending on it cannot be built,
                  as much as possible will be built before stopping.
```

https://www.yoctoproject.org/docs/latest/bitbake-user-manual/bitbake-user-manual.html#usage-and-syntax

First steps for a more efficient bitbake build (2)

`sstate cache`

https://wiki.yoctoproject.org/wiki/Enable_sstate_cache

First steps for a more efficient bitbake build (2)

`sstate cache`

https://wiki.yoctoproject.org/wiki/Enable_sstate_cache

I will talk more about caching in my talk tomorrow:
"Augmenting sstate-cache with ccache"

First steps for a more efficient bitbake build (3)

Parallel build

local.conf in build/conf directory:

PARALLEL_MAKE

PARALLEL_MAKEINST

https://www.yoctoproject.org/docs/latest/ref-manual/ref-manual.html#var-PARALLEL_MAKE
https://www.yoctoproject.org/docs/latest/ref-manual/ref-manual.html#var-PARALLEL_MAKEINST

First steps for a more efficient bitbake build (3)

Parallel build

local.conf in build/conf directory:

PARALLEL_MAKE

PARALLEL_MAKEINST

By default, the OpenEmbedded build system automatically sets this variable to be equal to the number of cores the build system uses.

https://www.yoctoproject.org/docs/latest/ref-manual/ref-manual.html#var-PARALLEL_MAKE
https://www.yoctoproject.org/docs/latest/ref-manual/ref-manual.html#var-PARALLEL_MAKEINST

First steps for a more efficient bitbake build (3)

`Parallel build`

`local.conf in build/conf directory:`

PARALLEL_MAKE

PARALLEL_MAKEINST

By default, the OpenEmbedded build system automatically sets this variable to be equal to the number of cores the build system uses.

https://www.yoctoproject.org/docs/latest/ref-manual/ref-manual.html#var-PARALLEL_MAKE
https://www.yoctoproject.org/docs/latest/ref-manual/ref-manual.html#var-PARALLEL_MAKEINST

Setting different PARELLEL_MAKE value at the recipe level (+ reasons for doing that):
https://stackoverflow.com/questions/70597349/yocto-build-maxium-parallel-make-bb-number-threads

First steps for a more efficient bitbake build (4)

Additional advice by Yocto manual:

https://www.yoctoproject.org/docs/latest/dev-manual/dev-manual.html#speeding-up-a-build

Throwing in additional cores

- More powerful HW
- Distributing to additional machines (on-prem / cloud)

Throwing in additional cores

- More powerful HW
- Distributing to additional machines (on-prem / cloud)

But would it help?

Throwing in additional cores

- More powerful HW
- Distributing to additional machines (on-prem / cloud)

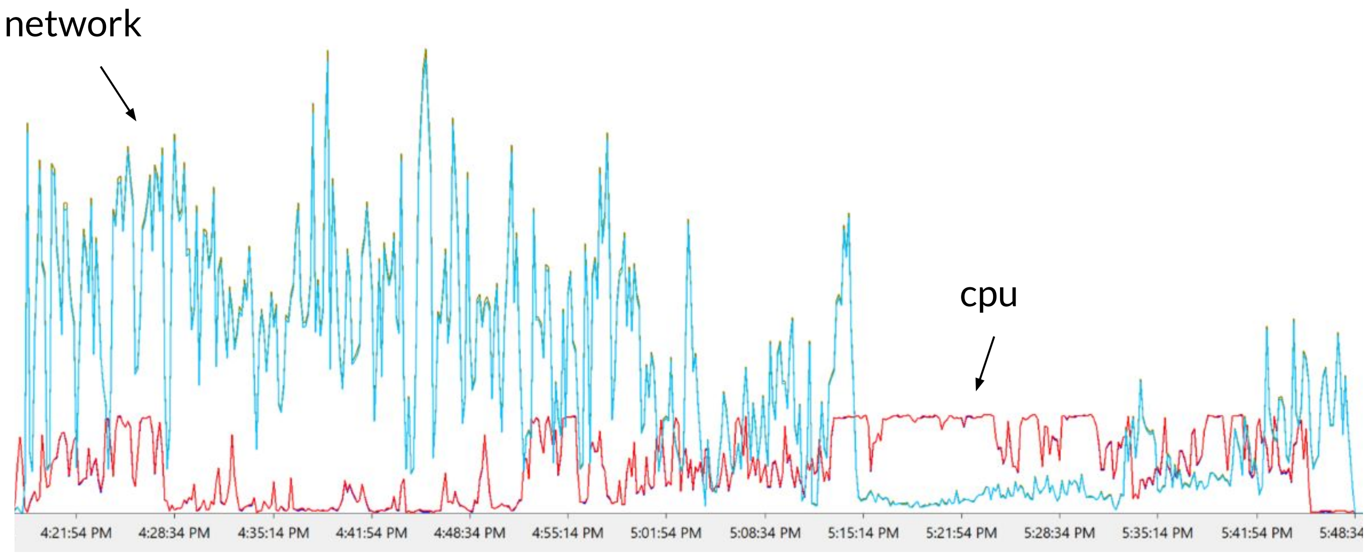
But would it help?

Where is the bottleneck?

Common HW resources a build consumes

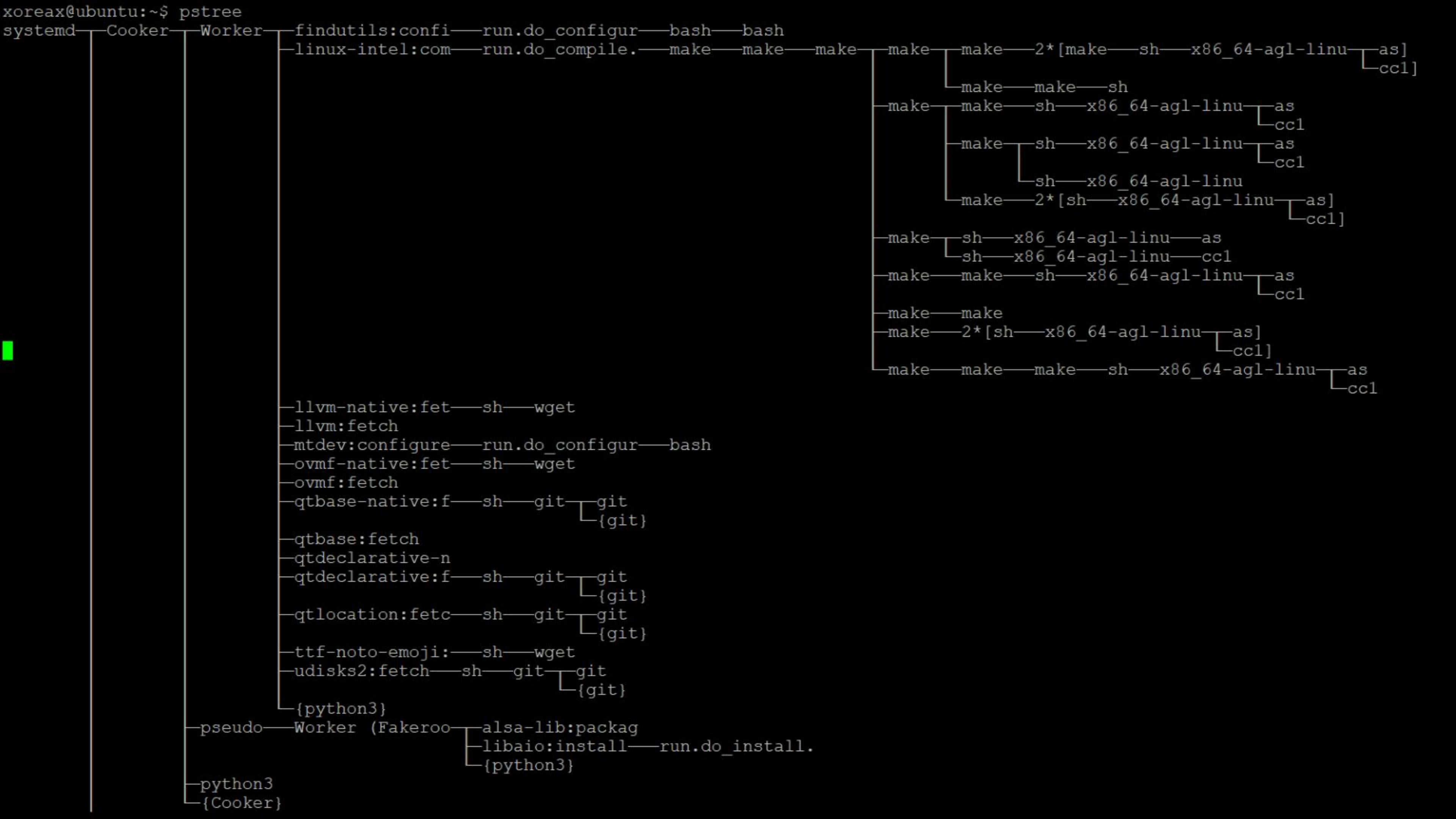
- **CPU**
  - Multicore: if tasks exceed core number they will get queued
  - There are tasks that can utilize only a single CPU
- **Memory**
  - Swapping to virtual memory is possible but costly
- **IO**
  - Slow disks can cause waits until data is fetched or till a dependency file finished writing
- **Network**
  - Yocto has no separate “download” phase. Slow network would affect download speed and build time

Throwing in additional cores



Analyzing things on the process level

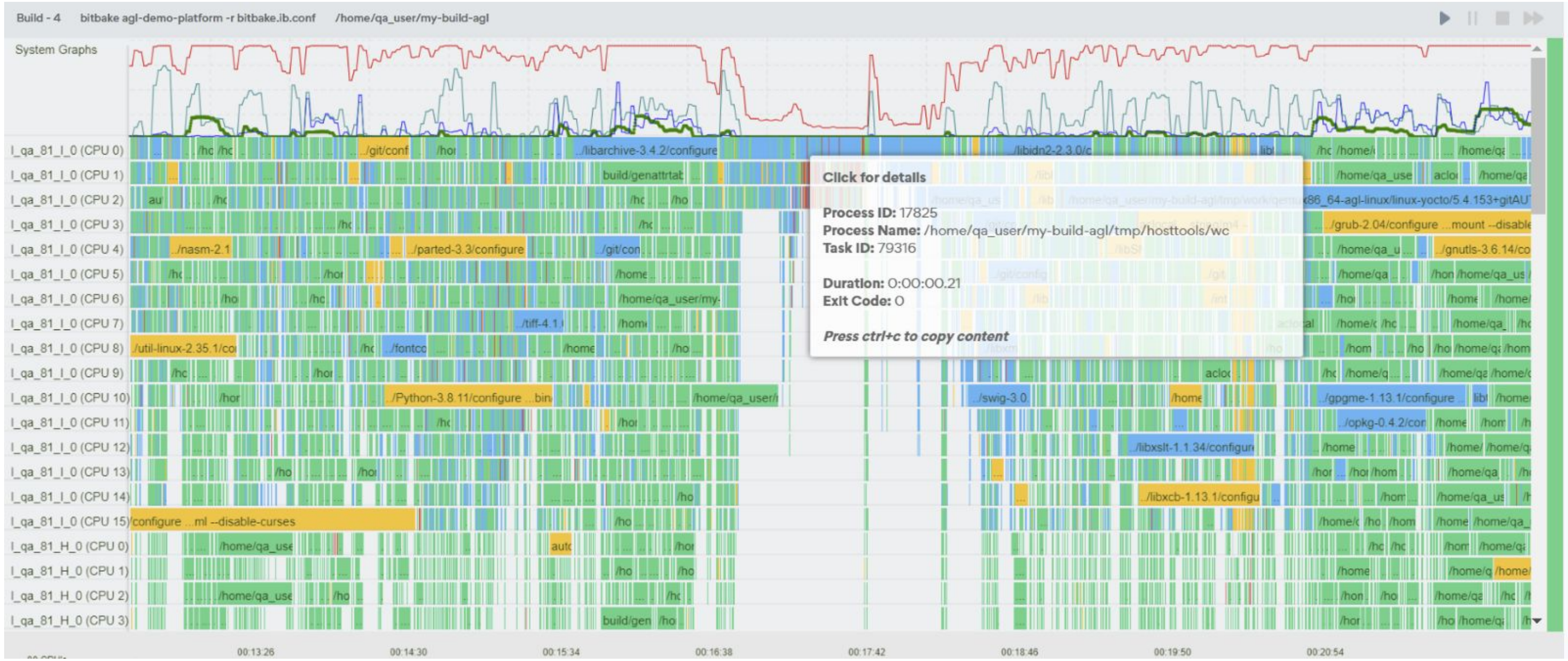
- Which tools are common across many recipes?
- Which tools take up most of the time?
- What are the bottlenecks on the process level?



```
xoreax@ubuntu:~$ pstree
systemd─┬─Cooker─┬─Worker─┬─findutils:confi───run.do_configur───bash───bash
        │        │        ├─linux-intel:com───run.do_compile.───make───make───make─┬─make─┬─make───2*[make───sh───x86_64-agl-linu─┬─as]
        │        │        │                                                        │      │                                       └─cc1]
        │        │        │                                                        │      └─make───make───sh
        │        │        │                                                        ├─make─┬─make───sh───x86_64-agl-linu─┬─as
        │        │        │                                                        │      │                             └─cc1
        │        │        │                                                        │      ├─make─┬─sh───x86_64-agl-linu─┬─as
        │        │        │                                                        │      │      │                      └─cc1
        │        │        │                                                        │      │      └─sh───x86_64-agl-linu
        │        │        │                                                        │      └─make───2*[sh───x86_64-agl-linu─┬─as]
        │        │        │                                                        │                                       └─cc1]
        │        │        │                                                        ├─make─┬─sh───x86_64-agl-linu───as
        │        │        │                                                        │      └─sh───x86_64-agl-linu───cc1
        │        │        │                                                        ├─make───make───sh───x86_64-agl-linu─┬─as
        │        │        │                                                        │                                    └─cc1
        │        │        │                                                        ├─make───make
        │        │        │                                                        ├─make───2*[sh───x86_64-agl-linu─┬─as]
        │        │        │                                                        │                                └─cc1]
        │        │        │                                                        └─make───make───make───sh───x86_64-agl-linu─┬─as
        │        │        │                                                                                                    └─cc1
        │        │        ├─llvm-native:fet───sh───wget
        │        │        ├─llvm:fetch
        │        │        ├─mtdev:configure───run.do_configur───bash
        │        │        ├─ovmf-native:fet───sh───wget
        │        │        ├─ovmf:fetch
        │        │        ├─qtbase-native:f───sh───git─┬─git
        │        │        │                            └─{git}
        │        │        ├─qtbase:fetch
        │        │        ├─qtdeclarative-n
        │        │        ├─qtdeclarative:f───sh───git─┬─git
        │        │        │                            └─{git}
        │        │        ├─qtlocation:fetc───sh───git─┬─git
        │        │        │                            └─{git}
        │        │        ├─ttf-noto-emoji:───sh───wget
        │        │        ├─udisks2:fetch───sh───git─┬─git
        │        │        │                          └─{git}
        │        │        └─{python3}
        │        ├─pseudo───Worker (Fakeroo─┬─alsa-lib:packag
        │        │                          ├─libaio:install───run.do_install.
        │        │                          └─{python3}
        │        ├─python3
        │        └─{Cooker}
        │
```

Incredibuild View

🟩 distributable
🟦 not distributable (local only)
🟨 task which printed to stderr


Build - 4   bitbake agl-demo-platform -r bitbake.ib.conf   /home/qa_user/my-build-agl

Click for details

Process ID: 17825
Process Name: /home/qa_user/my-build-agl/tmp/hosttools/wc
Task ID: 79316

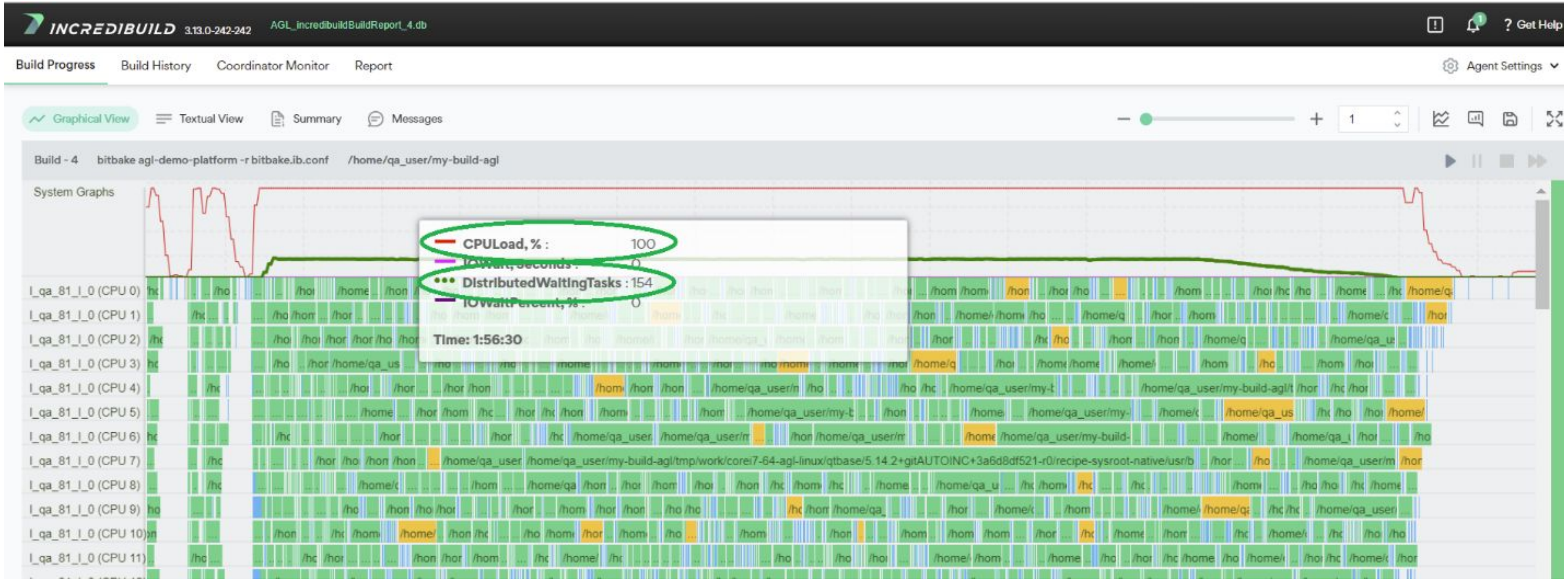
Duration: 0:00:00.21
Exit Code: 0

Press ctrl+c to copy content

https://bit.ly/YPS-2022_IB_bitbake

Incredibuild View

🟩	distributable
🟦	not distributable (local only)
🟨	task which printed to stderr



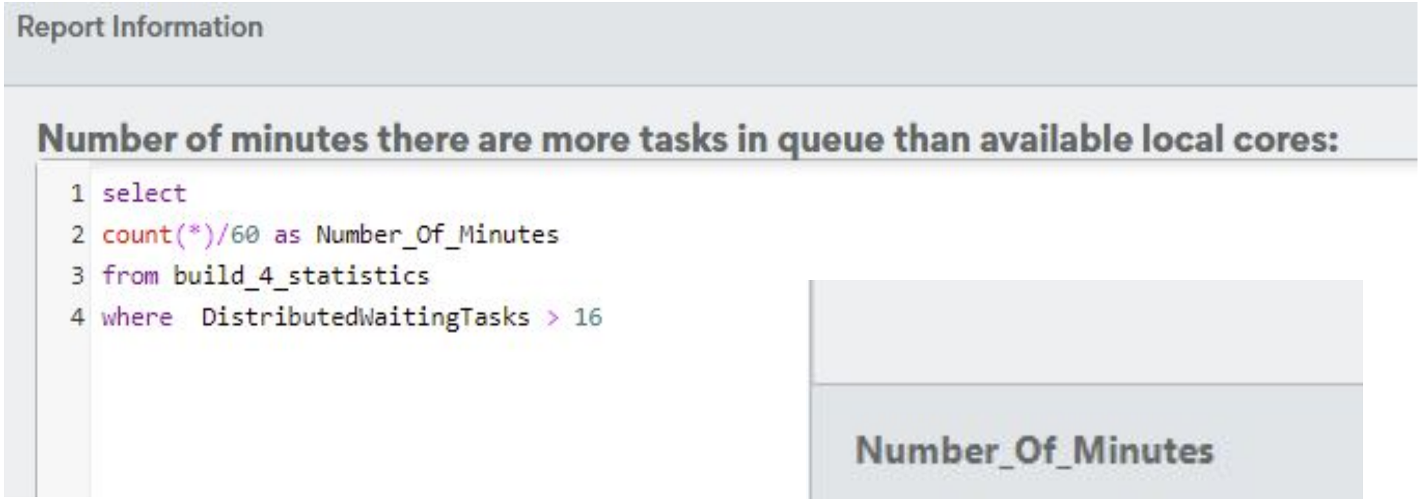
Build - 4 bitbake agl-demo-platform -r bitbake.ib.conf /home/qa_user/my-build-agl

CPULoad, % : 100
IOWait, seconds : 0
DistributedWaitingTasks : 154
IOWaitPercent, % : 0
Time: 1:56:30

/home/qa_user/my-build-agl/tmp/work/corei7-64-agl-linux/qtbase/5.14.2+gitAUTOINC+3a6d8df521-r0/recipe-sysroot-native/usr/b

https://bit.ly/YPS-2022_IB_bitbake

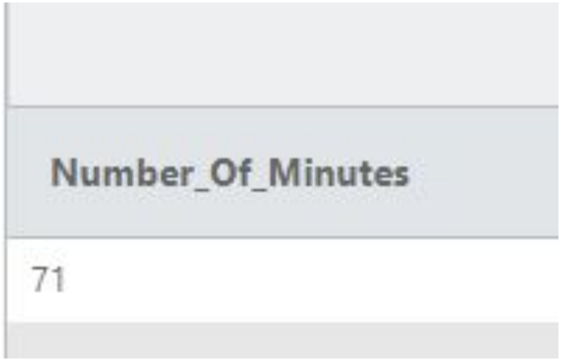
Data Analysis - CPU Wait



Report Information

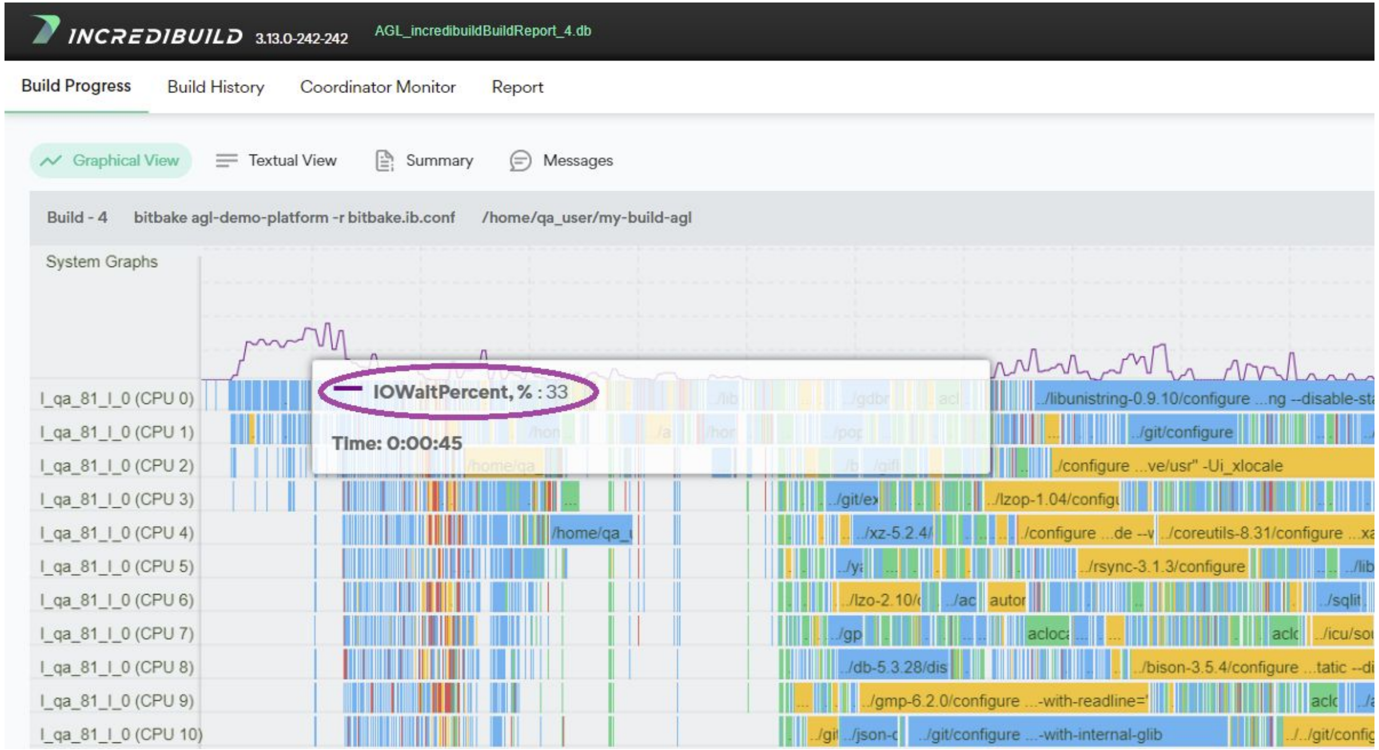
Number of minutes there are more tasks in queue than available local cores:

```
select
count(*)/60 as Number_Of_Minutes
from build_4_statistics
where  DistributedWaitingTasks > 16
```



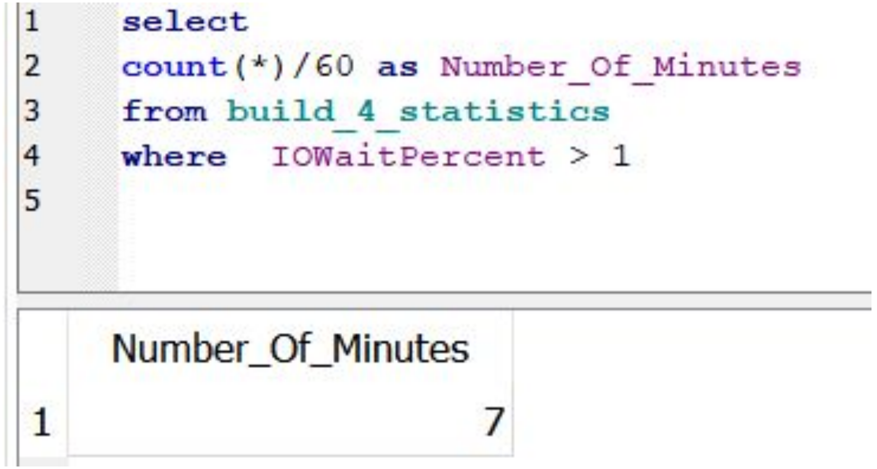
Number_Of_Minutes
71

Data Analysis - IO Wait



https://bit.ly/YPS-2022_IB_bitbake

Data Analysis - IO Wait

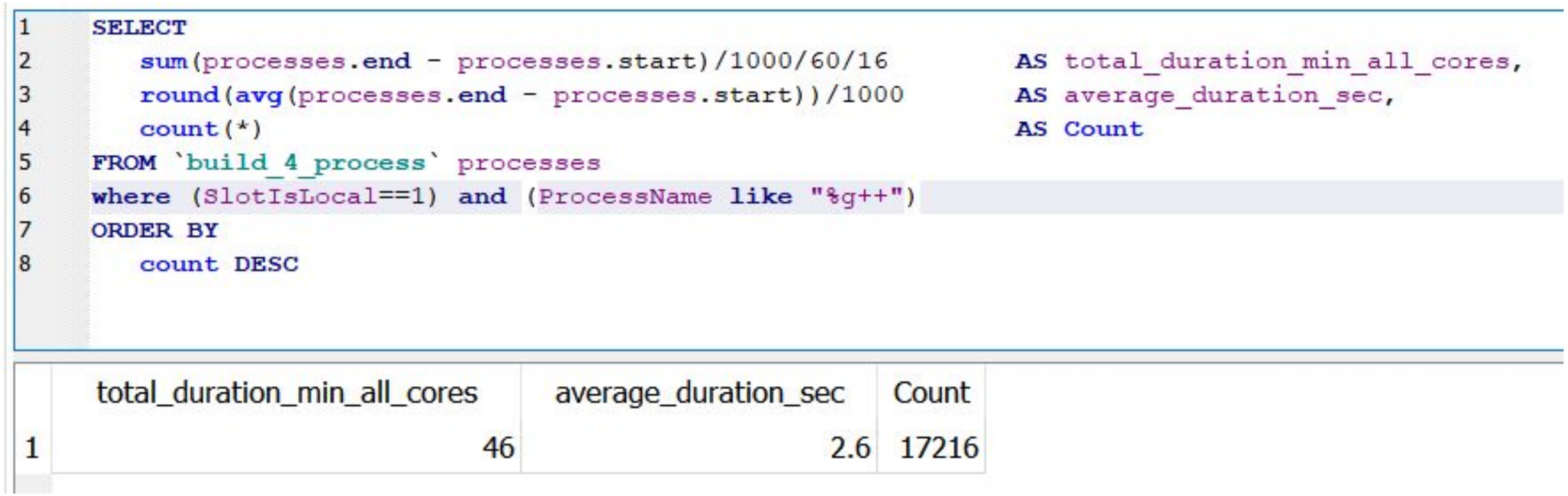


```
select
count(*)/60 as Number_Of_Minutes
from build_4_statistics
where  IOWaitPercent > 1

```

	Number_Of_Minutes
1	7

Data Analysis - Compiler Cache Potential



```
SELECT
   sum(processes.end - processes.start)/1000/60/16       AS total_duration_min_all_cores,
   round(avg(processes.end - processes.start))/1000      AS average_duration_sec,
   count(*)                                              AS Count
FROM `build_4_process` processes
where (SlotIsLocal==1) and (ProcessName like "%g++")
ORDER BY
   count DESC
```

	total_duration_min_all_cores	average_duration_sec	Count
1	46	2.6	17216

Top Time Takers Tasks:

- do_compile
- do_configure
- do_package (rpmbuild)

Parallelization - Special Challenges (1)

Agl 12.1 uses 321 different compilers

- Most of them are real files on disk (i.e. not a symlink)
- We see a few different gcc compilers and g++ compilers

Parallelization - Special Challenges (2)

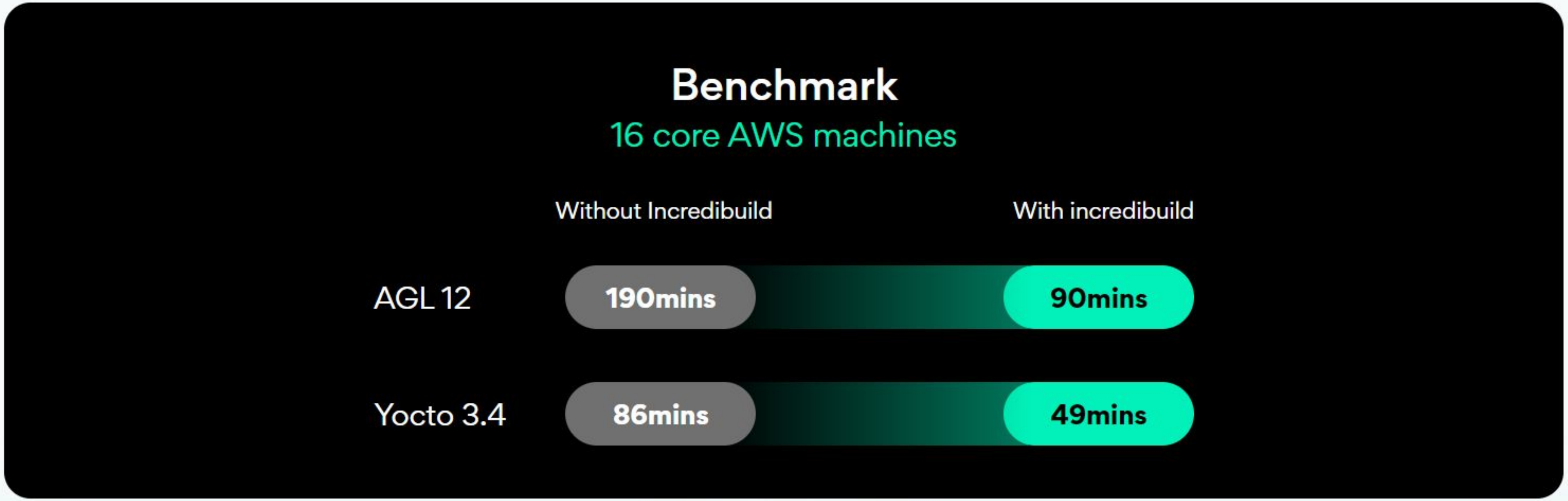
Virtualizing the entire filesystem on the task level without overwhelming the network

- No need to maintain a homogenous environment on the other machines in the grid (no compilers needed on helper, no source – everything synced on demand)
- Task runs in total isolation from helper with full emulation of the filesystem of the initiator

Parallelization - Special Challenges (3)

Permission and user management required to make sure images created with IB are identical to stock Yocto builds, especially supporting pseudo and different forms of chroot

Parallelization - Benchmark Results



To Summarize

- We love Yocto and bitbake

To Summarize

- We love Yocto and bitbake
- The bitbake build can be parallelized to additional machines to gain faster build time

To Summarize

- We love Yocto and bitbake
- The bitbake build can be parallelized to additional machines to gain faster build time
- I will discuss caching in my talk tomorrow

INCREDIBUILD

+

yocto PROJECT

https://www.incredibuild.com/blog/announcing-incredibuild-support-for-yocto
https://www.incredibuild.com/lp/yocto

Everything Everywhere All At Once

Breaking down the BitBake build on the process level

https://bit.ly/YPS-2022_IB_bitbake

amir.kirsh@incredibuild.com