Jorge @jvican

- Devtools for ~2.5 years at Scala Center
  - I co-maintain Scala's incremental compiler (Zinc)
  - I work on build tools and build servers
  - `scalac`, compiler plugins and infrastructure
- **Obsession**: developer productivity

Martin @mnduhem

- Software engineer at Scala Center since March 2017
  - I work on compilers and build tools.
  - I don't like it when tools get in my way.
  - Worked on sbt, Zinc, Scala Native and now Dotty.
- **Obsession**: Build tool agnosticity

*«In the common use case when someone is **working on a customer plugin** in one build and wanting to **iterate on core tooling** the turnarounds are **seconds instead of 10+ minutes**.»*

-- Happy Bloop user

Good tools should be sufficiently
flexible to embrace your workflow;
not force you to adopt their vision.

How Bloop makes you more productive

Productivity: any task you do is bound by the speed of your thinking process, not the tools you use.

Goal

Minimize the amount of time it passes between a change in the code and an execution result.


pants
sbt
bazel
gradle
maven
[adultswim.com]


Bloop

zinc++

`zinc++`

A gap to fill in

- `scalac` gives you only cold compilation
- `scalac` doesn't give you incrementality
- `scalac` cannot know in which context is used
- Build tools are too heavyweight
- Build tools are difficult to integrate with

zinc++

- It **complements** your go-to build tool
- It enables you to write custom extensions for your team
  - Developer workflows are orthogonal to build tools
  - Developer workflows depend on the culture of the company

What can we do?

Centralization

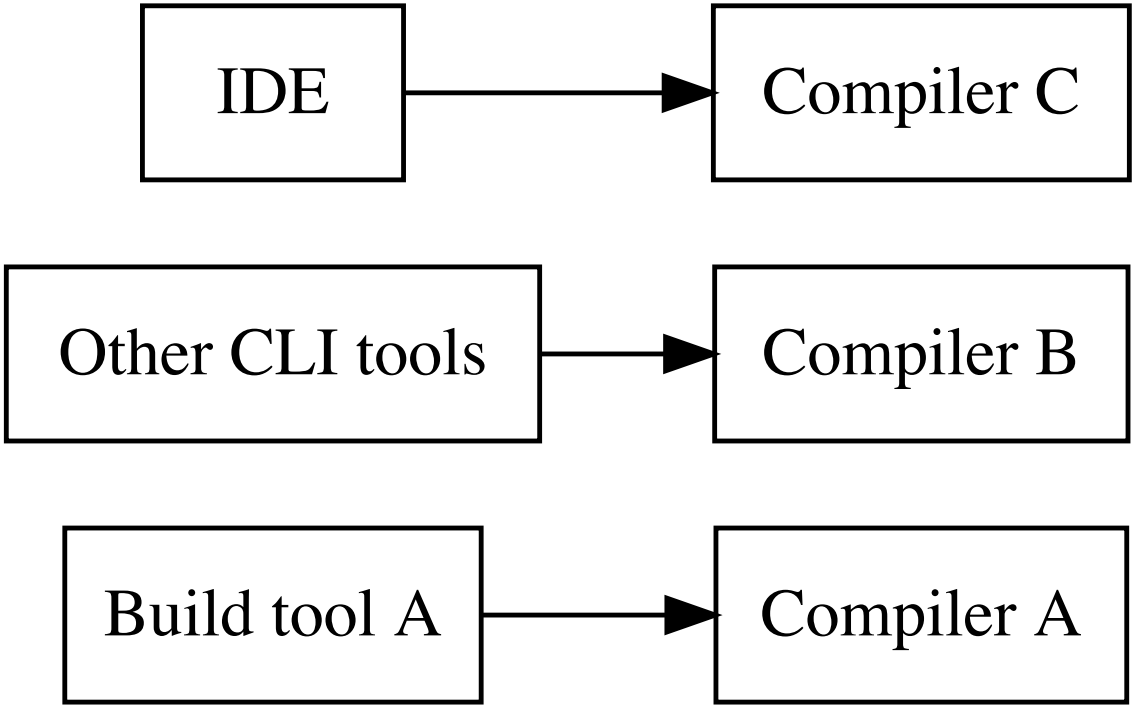
"One server to compile them all"

- Changes to the codebase have higher impact
- Easier to maintain, optimize and test
- No more fragmentation in build tools
  - e.g. Gradle is still using Zinc 0.13.x!

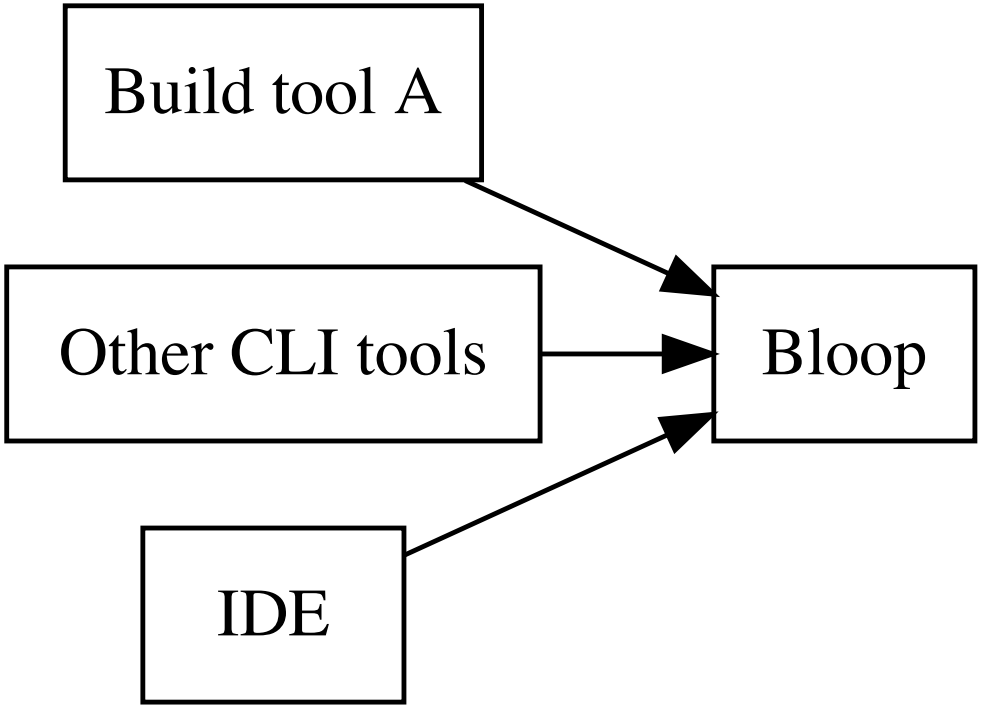
Compilation server

The new kid in the block

Old architecture



New architecture



Properties

Compiler agnostic

- Scala 2 support.
- Scala 3 support.

Properties

Platform agnostic

- JVM
- JS
- Native

Properties

Build-tool agnostic

- sbt
- Maven
- Gradle
- etc.

Properties

Better optimization opportunities

- Knowledge about build graph enables
  - In-memory storage over writing to disk/SSD
  - Safer cache invadalidation
    - Classpath
    - Source hashes
  - Better task parallelization

Hot compilers

What is that?

Hot compilers

- Performance diff against cold compilers is critical.
- Every developer tools has its own compiler instance.
- We kill hot compilers more often than we think.
  - `sbt> reload`
  - Killing accidentally the build tool.
  - Every time we use sbt in different builds concurrently.

Hot vs Cold

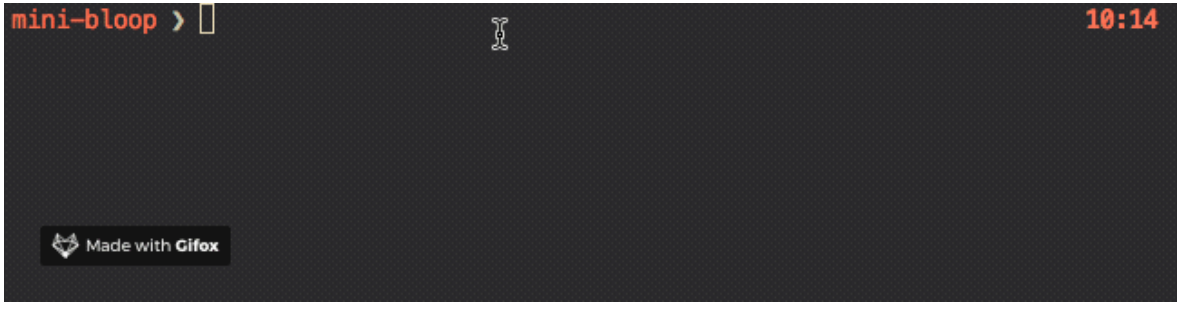
Benchmarks in grafana.

- Cold compilation can be up to 18x slower!
- Hot'ting the JVM is expensive, it takes:
  - CPU cycles, RAM, power (battery)...

Using **Bloop**

A command-line tool

- Ensures snappy developer workflow
- Use the shell you know and love.
- Benefit from awesome `tab`-completion



Bloop configuration file

```
{
"project" : {
  "name" : "mini-bloop",
  "directory" : "/Users/martin/Desktop/mini-bloop",
  "sources" : [ "/Users/martin/Desktop/mini-bloop/src/main/scala" ],
  "dependencies" : [  ],
  "classpath" : [ "/Users/martin/.ivy2/cache/org.scala-lang/scala-libra
  "classpathOptions" : { (...) },
  "out" : "/Users/martin/Desktop/mini-bloop/.bloop/mini-bloop",
  "classesDir" : "/Users/martin/Desktop/mini-bloop/.bloop/mini-bloop/sc
  "scala" : {
    "organization" : "org.scala-lang",
    "name" : "scala-compiler",
    "version" : "2.12.3",
    "options" : [ ],
    "jars" : [ (...) ]
  }, (...)
```

Bloop configuration file

- Transparent, well-specified configuration file
- Query build information from the project files.
- Write easy and performant scripts.
- Specification of our configuration format

A two-step process

- Bloop needs configuration files to understand your build
- The configuration files are generated by your build tool
- Let's see how to get ready using Bloop

```
diff --git a/project/plugins.sbt b/project/plugins.sbt
index 278929bd0..6665ae7c9 100644
--- a/project/plugins.sbt
+++ b/project/plugins.sbt
@ -7,3 +7,4 @ addSbtPlugin("ch.epfl.scala" % "sbt-release-early" % "2.1
+addSbtPlugin("ch.epfl.scala" % "sbt-bloop" % "1.0.0-M10")
```

```
$ sbt bloopInstall
```

Using Bloop

`bloop compile`

- Compiles your project (and its dependencies) in parallel.
- `--watch` will re-compile when source files change.
- `--reporter` lets you define the error reporter to use

```
mini-bloop ❯ bloop compile mini-bloop --reporter bloop
Compiling 1 Scala source to /Users/martin/Desktop/mini-bloop/.bloop/mini-bloop/scala-2.12/classes ...
[E] [E1] src/main/scala/bar/Main.scala
[E]      not found: value printkn
[E]      L5:     printkn("Goodbye!")
[E]             ^
[E] src/main/scala/bar/Main.scala: L5 [E1]
[E] 'mini-bloop' failed to compile.
mini-bloop ❯ bloop compile mini-bloop --reporter scalac
Compiling 1 Scala source to /Users/martin/Desktop/mini-bloop/.bloop/mini-bloop/scala-2.12/classes ...
[E] /Users/martin/Desktop/mini-bloop/src/main/scala/bar/Main.scala:5:5: not found: value printkn
[E]     printkn("Goodbye!")
[E]     ^
[E] one error found
[E] 'mini-bloop' failed to compile.
```

- Use `--help` to see all the options.

Using Bloop

`bloop test`

- Test a project and its dependencies
- `--isolated` to skip testing the dependencies
- Use `--only` to filter the tests to run

```
bloop test frontend --only bloop.engine.* --watch
```

Using Bloop

`bloop run`

- Runs a `main` method in your project
- Use `--main` to specify the main (or let Bloop decide)
- Use `--watch` to re-run when source files are modified

```
bloop run neural-network -- -train img/**/*.jpg -out $HOME/data
```

Using Bloop

`bloop console`

- Starts a new scala REPL with your project on the classpath
- Equivalent to `sbt console`

Using Bloop

- All commands support `tab`-completion.
- You have access to all the features of your usual shell.
- Get help with `--help`

Integrations

Paramount for adoption.

«We created bloop to primarily support all the existing build tools that companies and Scala open-source developers use today.»

-- Us

- Improving only sbt is **not** a solution.
- Improving only new build tools is **not** a solution.

Build Server Protocol (**BSP**)

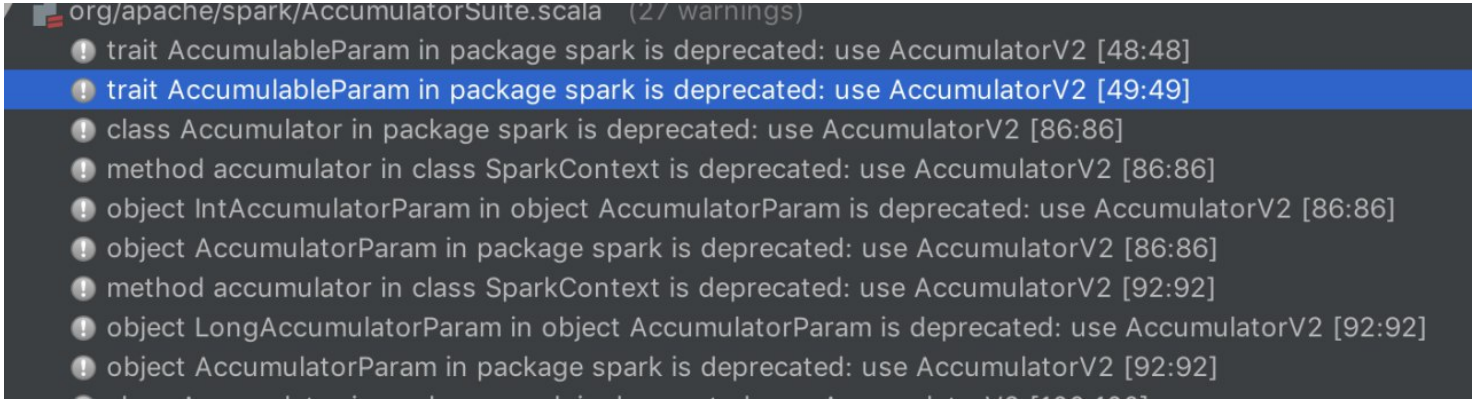
- Bloop is a BSP server.
- An LSP-like protocol to communicate with build tools.
- Clients are language servers and editors.
- More at "Build Server Protocol and new IDEAs"
  - Presented by Jorge and Justin at **Scalasphere 2018**

Build Server Protocol

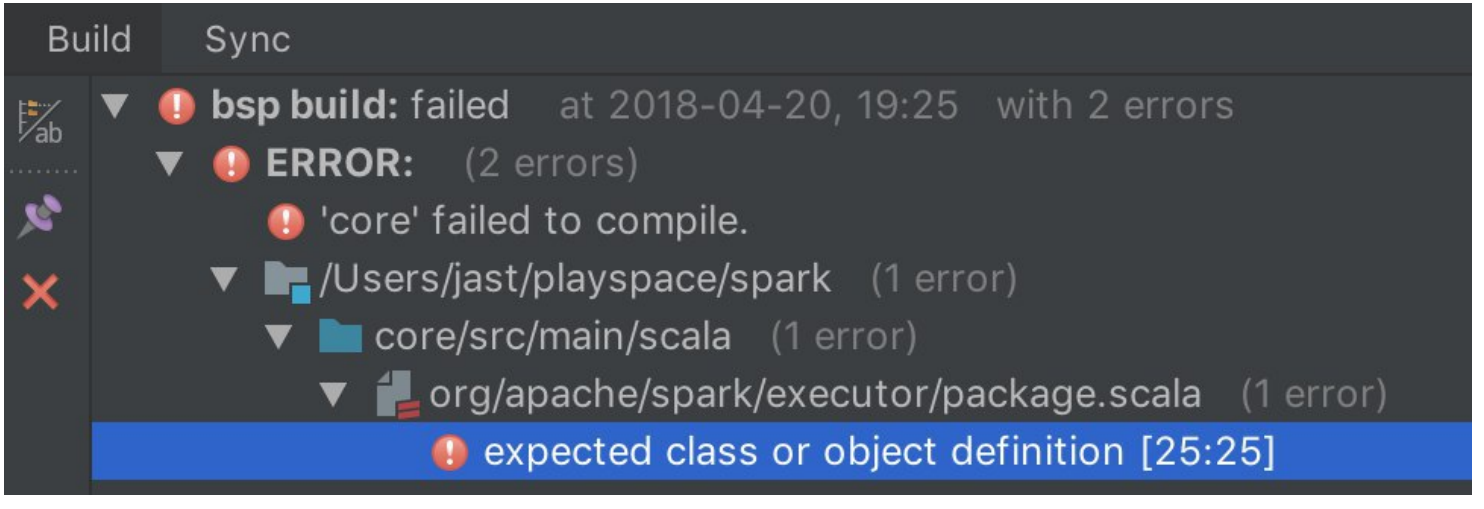
Use cases in Bloop

- Efficient import project from IDEs and editors
- Better integration with bloop
  - sbt, maven, bazel, et cetera

Build Server Protocol



Build Server Protocol



Compilation **benchmarks**

`sbt/sbt`

25% faster

- The Scala ecosystem's build tool.
- Small to medium project
- **800** source files, **50,000** lines of code, **20** modules
- Compiles in **38.05** s with sbt, or **28.47** s with Bloop

`guardian/frontend`

28% faster

- The source for theguardian.com.
- Medium-sized project.
- **1,800** source files, **180,000** lines of code, **18** modules.
- Compiles in **88.62** s with sbt, or **63.42** s.

akka/akka

19% faster

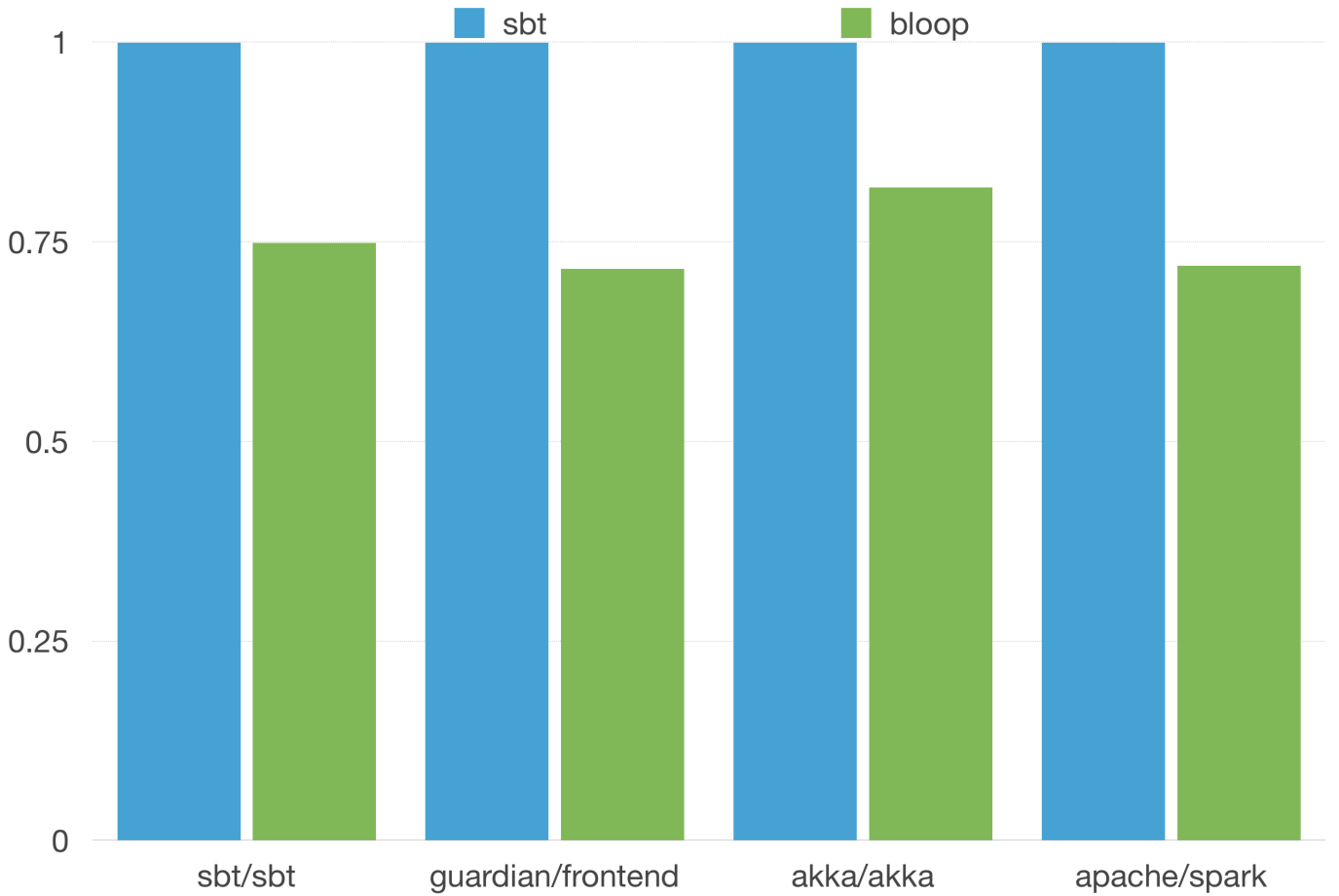
- Toolkit for building concurrent, distributed applications.
- Large project.
- **1,800** source files, **330,000** lines of code, **36** modules.
- Compiles in **263.7** s with sbt, or **215.7** s.

apache/spark

28% faster

- Cluster computing system for Big Data.
- Large project.
- **1,800** source files, **400,000** lines of code, **24** modules.
- Compiles in **238.24** s with sbt, or **171.48** s

Benchmarks


sbt
bloop
1
0.75
0.5
0.25
0
sbt/sbt
guardian/frontend
akka/akka
apache/spark

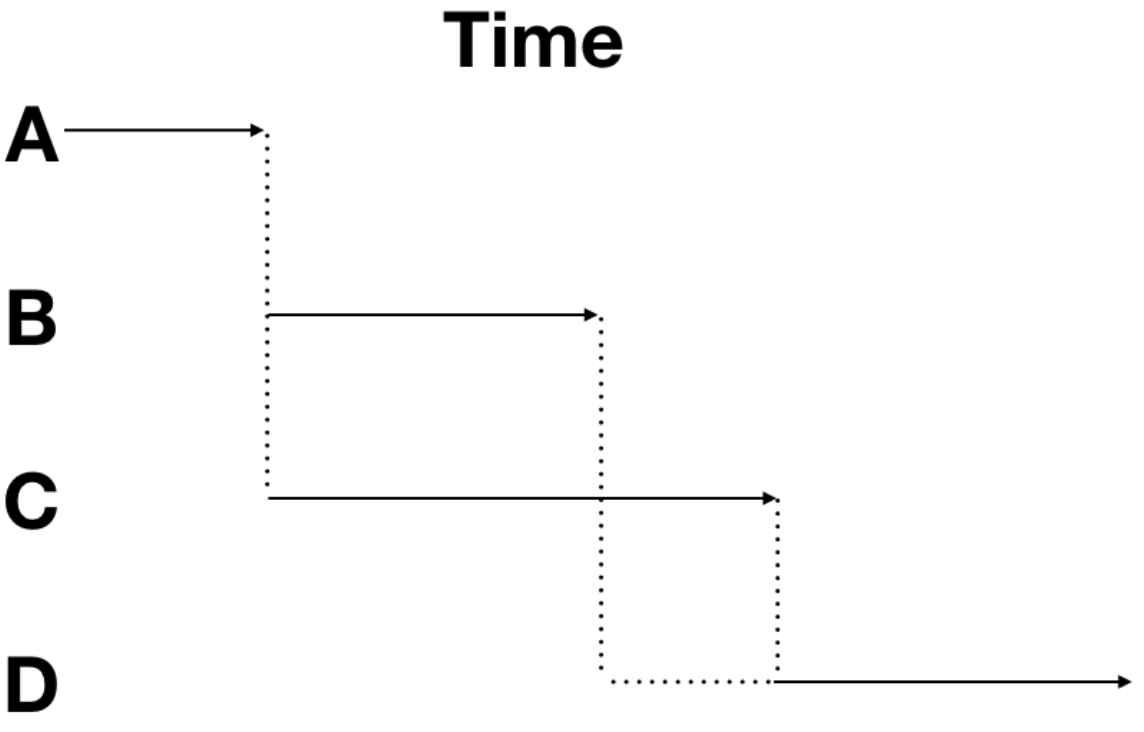
On the roadmap

Remote compilation

- Popular use case for CI servers
- Keep cloud instances alive for compilation jobs
- Requires architectural changes:
  - Independence of compile processes
  - Deduplication of compile requests

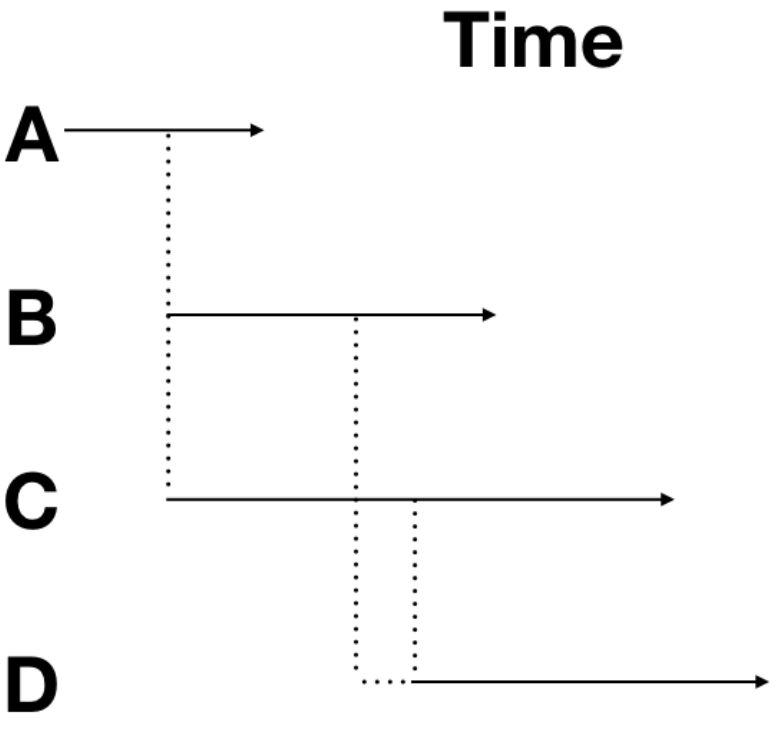
On the roadmap

Traditional compilation


Time
A
B
C
D

On the roadmap

Pipelined compilation


Time
A
B
C
D

On the roadmap

Pipelined compilation

- Pitched in by Rory Graves at Scalasphere 2017
- High impact for any build graph!
- Restricted gains when Java compilation is required
- Implemented in Zinc and Bloop (for now)

Thanks!

Questions?

MacOS

```
$ brew install scalacenter/bloop/bloop
```

Linux / Windows

```
$ curl -L https://git.io/vpdVs | python
```