Caching Explicit Clang Modules with Content-Addressable Storage

Ben Langmuir

LLVM Developers' Meeting 2023 | Apple Inc. | October 11, 2023

Previously...

20 22
LLVM
DEVELOPERS' MEETING
SAN JOSE, CALIFORNIA • HAYES MANSION
Using Content-Addressable Storage in Clang
for Caching Computations and Eliminating Redundancy
Steven Wu / Ben Langmuir

LLVM Dev 2022: Using Content-Addressable Storage in Clang for Caching Computations and Eliminating Redundancy
https://www.youtube.com/watch?v=E9GdNKjGZ7Y

RFC: Add an LLVM CAS library and experiment with fine-grained caching for builds
https://discourse.llvm.org/t/rfc-add-an-llvm-cas-library-and-experiment-with-fine-grained-caching-for-builds/59864

Previously...

2022 LLVM DEVELOPERS' MEETING
SAN JOSE, CALIFORNIA • HAYES MANSION
Using Content-Addressable Storage in Clang for Caching Computations and Eliminating Redundancy
Steven Wu / Ben Langmuir

LLVM Dev 2022: Using Content-Addressable Storage in Clang for Caching Computations and Eliminating Redundancy
https://www.youtube.com/watch?v=E9GdNKjGZ7Y

RFC: Add an LLVM CAS library and experiment with fine-grained caching for builds
https://discourse.llvm.org/t/rfc-add-an-llvm-cas-library-and-experiment-with-fine-grained-caching-for-builds/59864

New: Clang Modules Support

Quick Introduction

Content-addressable storage and compilation caching

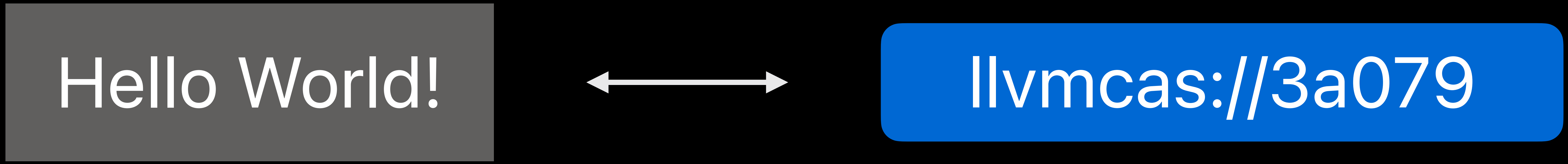
CAS Object Store

CAS object address = hash of contents

CAS Object Store

CAS object address = hash of contents

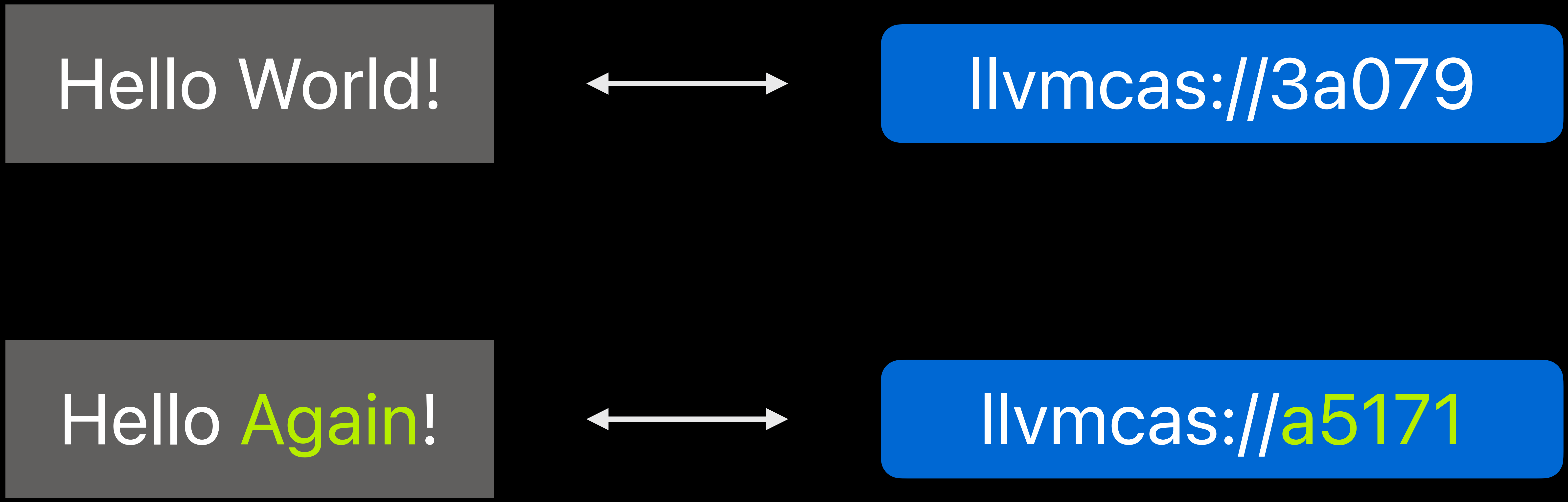
1:1 mapping



CAS Object Store

CAS object address = hash of contents

1:1 mapping

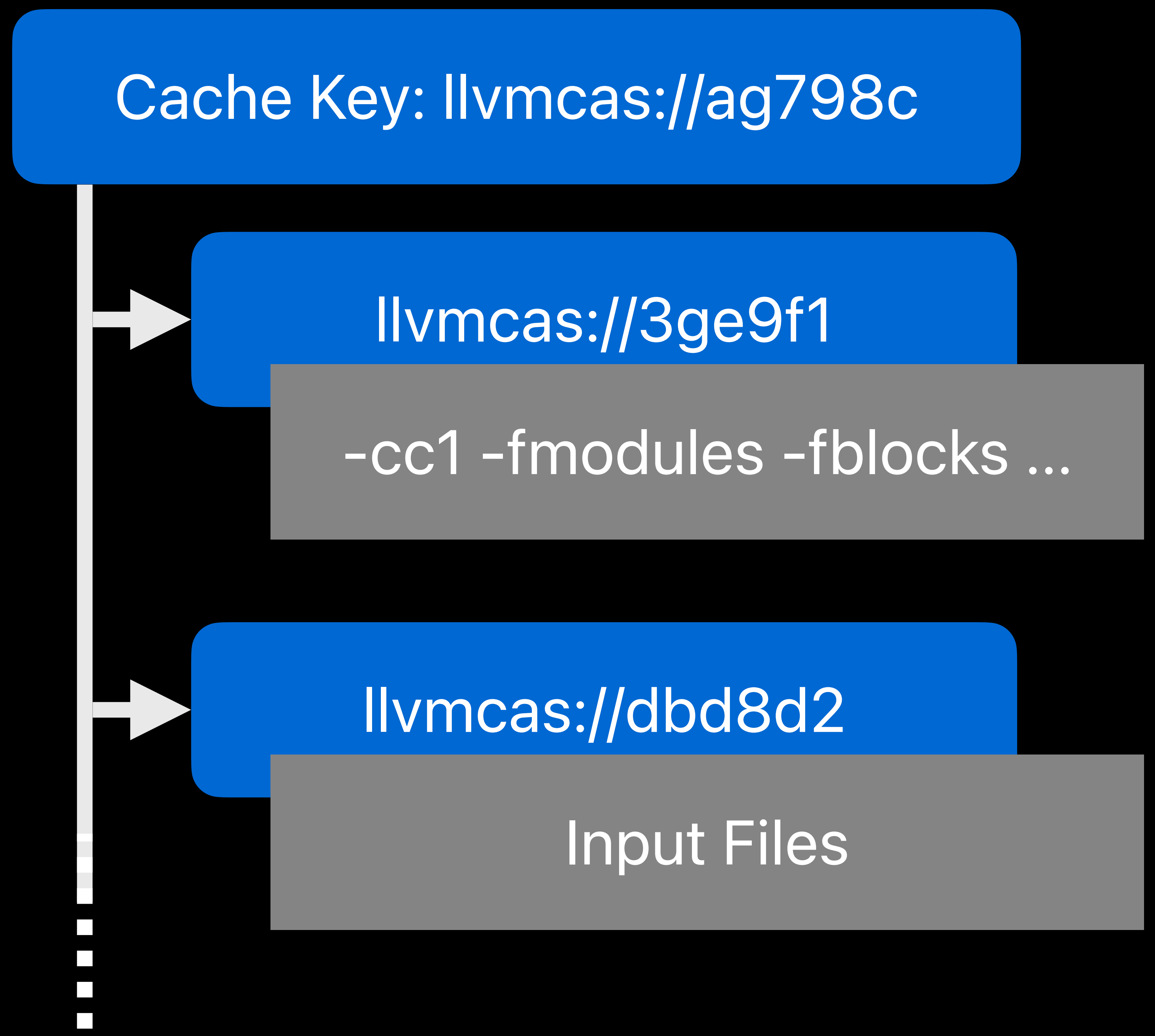


Caching Compilation

Map compiler inputs to outputs

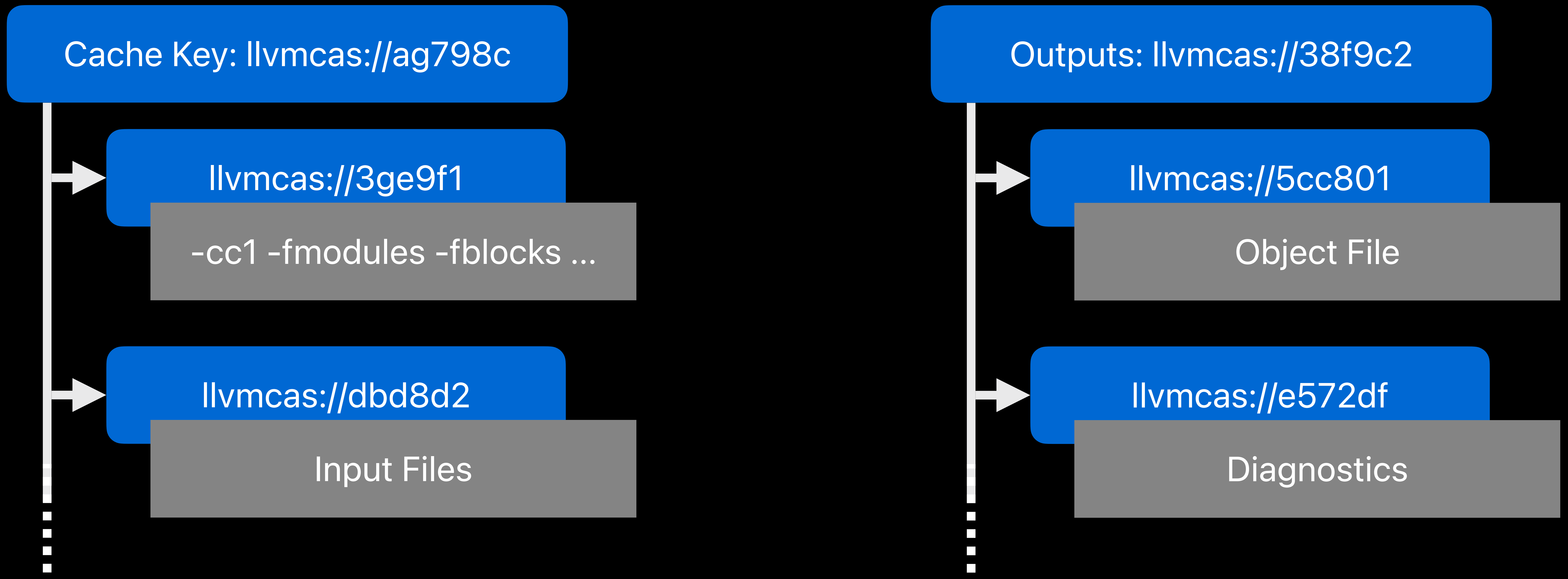
Caching Compilation

Map compiler inputs to outputs



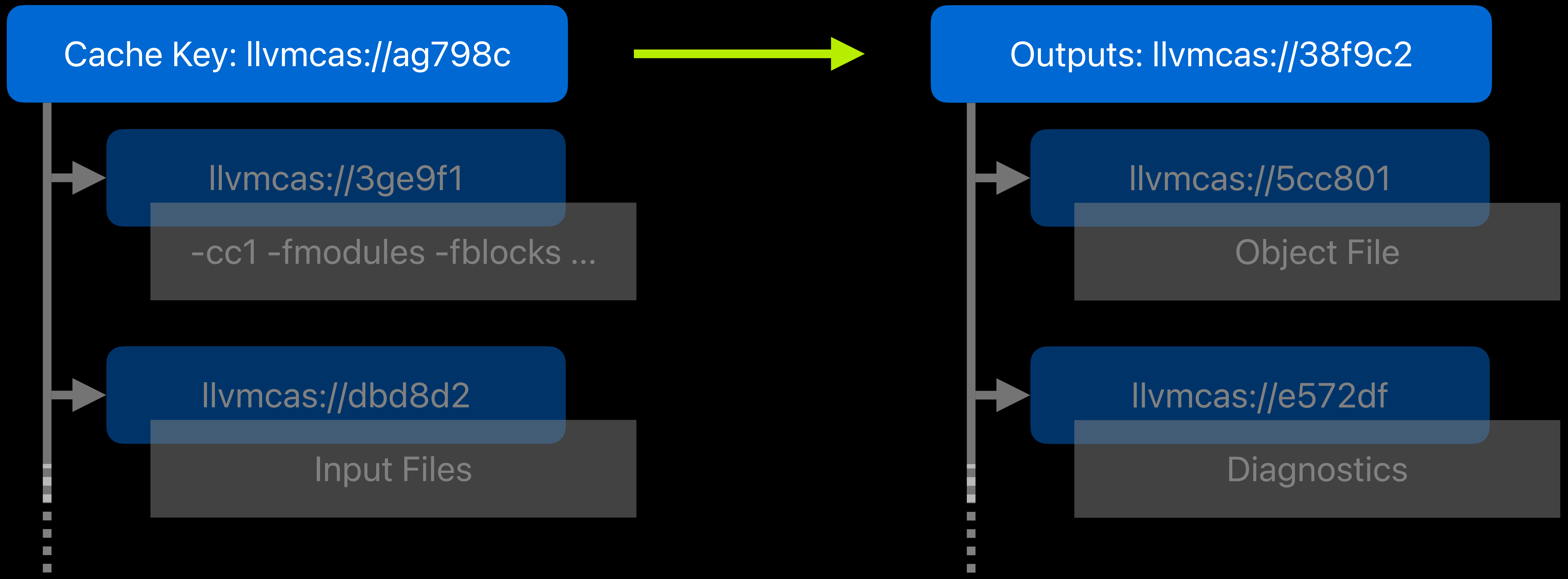
Caching Compilation

Map compiler inputs to outputs



Action Cache

Append-only key value store



Clang Compilation Caching

From Headers to Modules

Idea

Isolate compilation from mutable filesystem

Store all inputs in CAS

Plan

1. clang-scan-deps discovers inputs; ingest into CAS
2. Produce a -cc1 command that only accesses the CAS
3. Capture outputs in CAS
4. Cache results

clang-scan-deps Discovers Inputs

```c
#include <a.h>
void some_declaration(void);
#include <c.h>
#include <module.h>
```

clang-scan-deps Discovers Inputs

```
#include <a.h>
void some_declaration(void);
#include <c.h>
#include <module.h>
```

```
clang-scan-deps \
    -compilation-database compile_commands.json \
    -format experimental-include-tree-full
```

clang-scan-deps Discovers Inputs

```c
#include <a.h>
void some_declaration(void);
#include <c.h>
#include <module.h>
```

Dependency Scan Output

```json
"modules": [
  {
    "command-line": [
      "-cc1",
      ...
    ]
  }
],
"translation-units": [
  {
    "command-line": [
      "-cc1",
      ...     ]
  }
```

CAS Inputs

Dependency Scan Output

```
#include <a.h>
void some_declaration(void);
#include <c.h>
#include <module.h>
```

```
"modules": [ ... ],

"translation-units": [
  {
    "command-line": [
      "-cc1",

      "-fcas-include-tree",
      "llvmcas://062e95...",

      ...

    ]

  }
```

CAS Inputs

Dependency Scan Output

```
#include <a.h>
void some_declaration(void);
#include <c.h>
#include <module.h>
```

```
"modules": [ ... ],

"translation-units": [
  {
    "command-line": [
      "-cc1",

      "-fcas-include-tree",
      "llvmcas://062e95...",

      ...

    ]

  }
```

```
clang-cas-test -print-include-tree llvmcas://062e95...
```

Main File

```
#include <a.h>
void some_declaration(void);
#include <c.h>
#include <module.h>
```

Dependency Scan Output

Include Tree

```
main.c llvmcas://dbd8d2...
```

Main File

```
#include <a.h>
void some_declaration(void);
#include <c.h>
#include <module.h>
```

Dependency Scan Output
Include Tree

```
main.c llvmcas://dbd8d2...
```

```
#include <a.h>
void some_declaration(void);
#include <c.h>
#include <module.h>
```

Headers

Dependency Scan Output
Include Tree

```c
#include <a.h>
void some_declaration(void);
#include <c.h>
#include <module.h>
```

```
main.c llvmcas://dbd8d2...
2:1 include/a.h llvmcas://44e78e...
```

Headers

```
#include <a.h>
void some_declaration(void);
#include <c   >
#include <   le.h>
```

```
#include <b.h>
```

Dependency Scan Output
Include Tree

```
main.c llvmcas://dbd8d2...
2:1 include/a.h llvmcas://44e78e...
```

```
#include <b.h>
```

Headers

```
#include <a.h>
void some_declaration(void);
#include <c  >
#include <   le.h>
```

```
#include <b.h>
```

Dependency Scan Output
Include Tree

```
main.c llvmcas://dbd8d2...
2:1 include/a.h llvmcas://44e78e...
  2:1 include/b.h llvmcas://e572df...
```

Headers

```
#include <a.h>
void some_declaration(void);
#include <c.h>
#include <module.h>
```

Dependency Scan Output
Include Tree

```
main.c llvmcas://dbd8d2...
2:1 include/a.h llvmcas://44e78e...
  2:1 include/b.h llvmcas://e572df...
4:1 include/c.h llvmcas://e572df...
```

Module Imports

```c
#include <a.h>
void some_declaration(void);
#include <c.h>
#include <module.h>
```

Dependency Scan Output
Include Tree

```
main.c llvmcas://dbd8d2...
2:1 include/a.h llvmcas://44e78e...
  2:1 include/b.h llvmcas://e572df...
4:1 include/c.h llvmcas://e572df...
5:1 (Module) MyModule
```

Compiling Modules

Building Module MyModule

```
module MyModule {
  header "module.h"
  link "ModLib"
  module Sub {
    header "sub.h"
    export *
  }
}
```

Building Module MyModule

```
module MyModule {
  header "module.h"
  link "ModLib"
  module Sub {
    header "sub.h"
    export *
  }
}
```

Dependency Scan Output

```
"modules": [
  {
    "command-line": [
      "-cc1",
      "-fcas-include-tree",
      "llvmcas://901282...",
      ...
    ]
  }
],
"translation-units": [
  {
    "command-line": [
      "-cc1",
```

Building Module MyModule

```
module MyModule {
  header "module.h"
  link "ModLib"
  module Sub {
    header "sub.h"
    export *
  }
}
```

Dependency Scan Output
Include Tree

```
<module-includes> llvmcas://c2c4bd...
2:1 include/module.h llvmcas://e572df...
3:1 include/sub.h llvmcas://e572df...
```

Building Module MyModule

```
module MyModule {
  header "module.h"
  link "ModLib"
  module Sub {
    header "sub.h"
    export *
  }
}
```

Dependency Scan Output
Include Tree

```
<module-includes> llvmcas://c2c4bd...
2:1 include/module.h llvmcas://e572df...
3:1 include/sub.h llvmcas://e572df...
```

Header ➡ Submodule

```
module MyModule {
  header "module.h"
  link "ModLib"
  module Sub {
    header "sub.h"
    export *
  }
}
```

Dependency Scan Output
Include Tree

```
<module-includes> llvmcas://c2c4bd...
2:1 include/module.h llvmcas://e572df...
  Submodule: MyModule
3:1 include/sub.h llvmcas://e572df...
  Submodule: MyModule.Sub
```

Other Module Map Semantics

```
module MyModule {
  header "module.h"
  link "ModLib"
  module Sub {
    header "sub.h"
    export *
  }
}
```

Dependency Scan Output
Include Tree

```
<module-includes> llvmcas://c2c4bd...
2:1 include/module.h llvmcas://e572df...
  Submodule: MyModule
3:1 include/sub.h llvmcas://e572df...
  Submodule: MyModule.Sub
Module Map:
MyModule
  link ModLib
  Sub
    export *
```

Module Include Tree

Compilation does not

- Parse .modulemap files
- Search for headers or modules
- Use -ivfsoverlay or headermaps

This is all handled once during dependency scan

Caching Module Build

```
Scan dependencies of main.c

Precompile Clang module MyModule

    remark: compile job cache hit for 'llvmcas://5ae4ab' =>
        'llvmcas://a07e21' [-Rcompile-job-cache-hit]
```

Importing Modules

Import PCM by CAS ID?

Dependency Scan Output
Include Tree

```
#include <a.h>
void some_declaration(void);
#include <c.h>
#include <module.h>
```

```
main.c llvmcas://dbd8d2...
2:1 include/a.h llvmcas://44e78e...
  2:1 include/b.h llvmcas://e572df...
4:1 include/c.h llvmcas://e572df...
5:1 (Module) MyModule
```

Import PCM by CAS ID?

```
#include <a.h>
void some_declaration(void);
#include <c.h>
#include <module.h>
```

Dependency Scan Output
Include Tree

```
main.c llvmcas://dbd8d2...
2:1 include/a.h llvmcas://44e78e...
  2:1 include/b.h llvmcas://e572df...
4:1 include/c.h llvmcas://e572df...
5:1 (Module) MyModule llvmcas://??????
```

```
<BLOCKINFO_BLOCK/>
<UNHASHED_CONTROL_BLOCK NumWords=90 BlockCodeSize=5>
  <AST_BLOCK_HASH abbrevid=4/> blob data = unprintable, 20 bytes.
  <SIGNATURE abbrevid=5/> blob data = unprintable, 20 bytes.

...
```

Problem

Module is not built at scan time ➡ do not know CAS ID

Would require scan to run after building module in dependency order

Solution: Use Cache Key for Module

```
#include <a.h>
void some_declaration(void);
#include <c.h>
#include <module.h>
```

Dependency Scan Output

```
"modules": [ ... ],

"translation-units": [
  {
    "command-line": [
      "-cc1",

      "-fcas-include-tree",
      "llvmcas://062e95...",

      "-fmodule-file-cache-key",
      "MyModule.pcm",
      "llvmcas://5ae4ab...",

      ...
```

Solution: Use ActionCache Key for Module

```
#include <a.h>
void some_declaration(void);
#include <c.h>
#include <module.h>
```

Dependency Scan Output

```
"modules": [ ... ],

        "-fmodule-fil    ache-key",
        "MyModule.pcm"
        "llvmcas://5ae4ab...",

        ...
```

Cache Key:

- `-cc1 -x c -fmodules ...`
- Include tree: `llvmcas://cfa9bb...`

Cache Keys

Compilation is pure computation

Key computed during dependency scan

Dependency scan in parallel

Use ActionCache to lookup MyModule.pcm contents during build

Compilation Caching with Modules

Compiling Modules ✅

Importing Modules ✅

Compilation Caching with Modules

```
Scan dependencies of main.c

Precompile Clang module MyModule

    remark: compile job cache hit for 'llvmcas://5ae4ab' =>
        'llvmcas://a07e21' [-Rcompile-job-cache-hit]

Compile main.c

    remark: compile job cache hit for 'llvmcas://7bd4c8' =>
        'llvmcas://12b876' [-Rcompile-job-cache-hit]
```

Conclusions

- Extended compilation caching to support Clang modules
- Tomorrow 4:45 pm
  - Optimizing Debug Info for Caching in llvm-cas
- Initial CAS patches in review
  - CAS: https://github.com/llvm/llvm-project/pull/68448
  - VirtualOutputBackend: https://github.com/llvm/llvm-project/pull/68447