CloudCompiler

Saman Amarasinghe, William S. Moses,
Daniel Donenfeld, Katsumi Okuda

The COMMIT Compiler Group

COM PILERS @ MIT

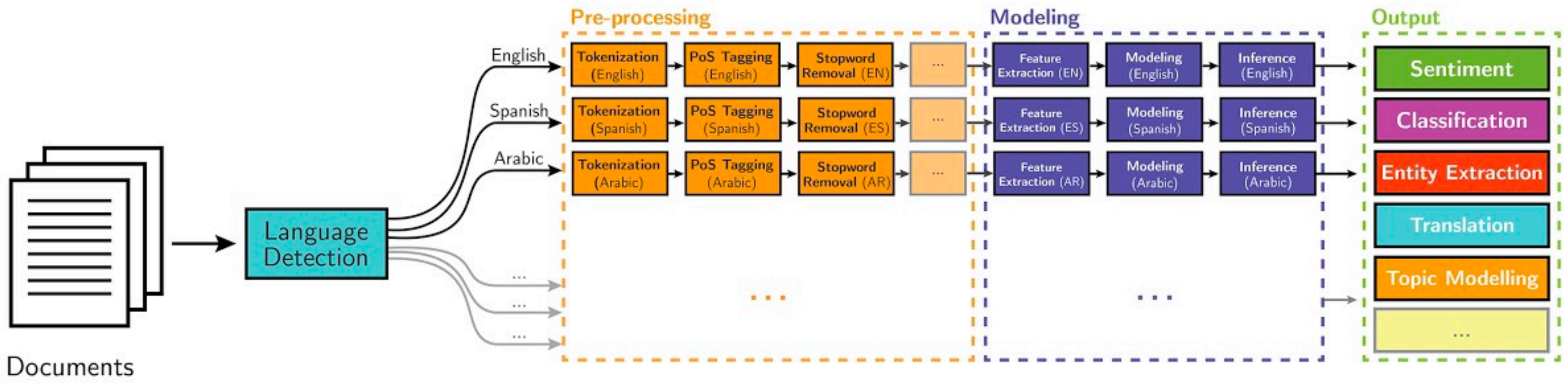
Massachusetts
Institute of
Technology

MIT CSAIL
compute.
collaborate.
create.

ILLINOIS
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

Language Processing Software in the 1990's

Natural Language Processing

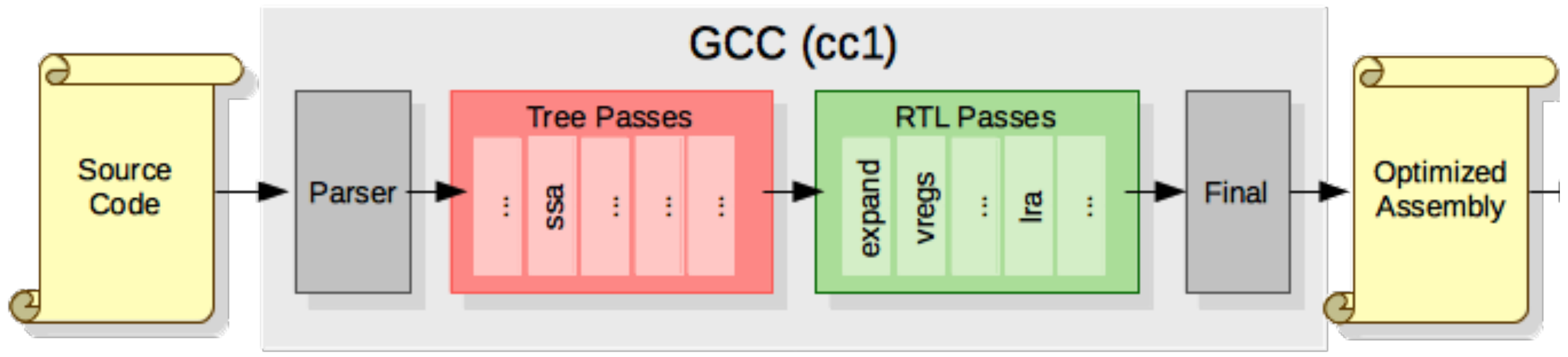


Rule-based Machine Translation (RBMT)

Components

- SL morphological analyser
- SL parser
- Translator
- TL morphological generator
- TL parser
- SL dictionary
- Bilingual dictionary
- TL dictionary

Programming Language Processing



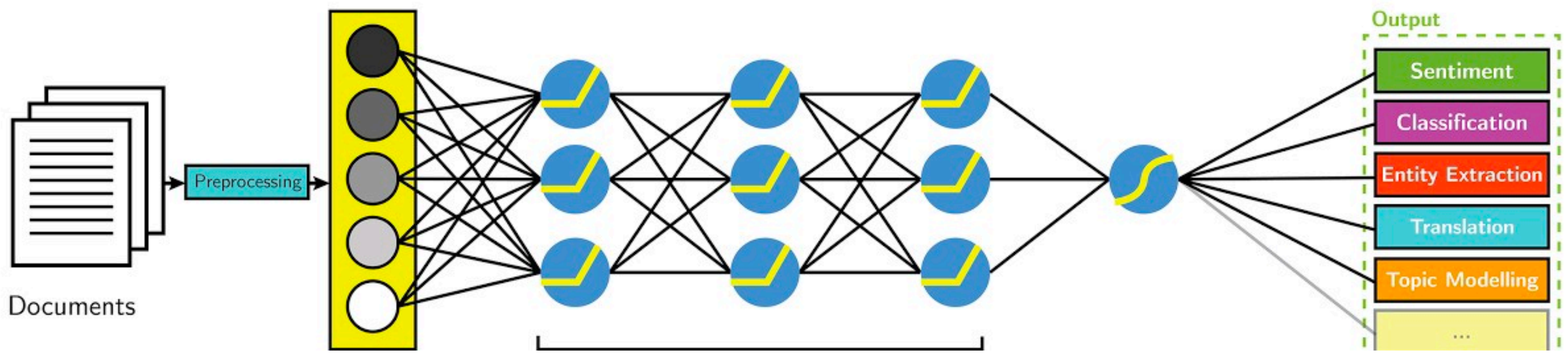
GCC Compiler Flow

Components

- Lexer
- Parser
- Semantic Analyser
- Intermediate Code Generator
- Code optimizer
- Low Level Code Generator

Language Processing Software in the 2020’s

Natural Language Processing



Neural Machine Translation (NMT)

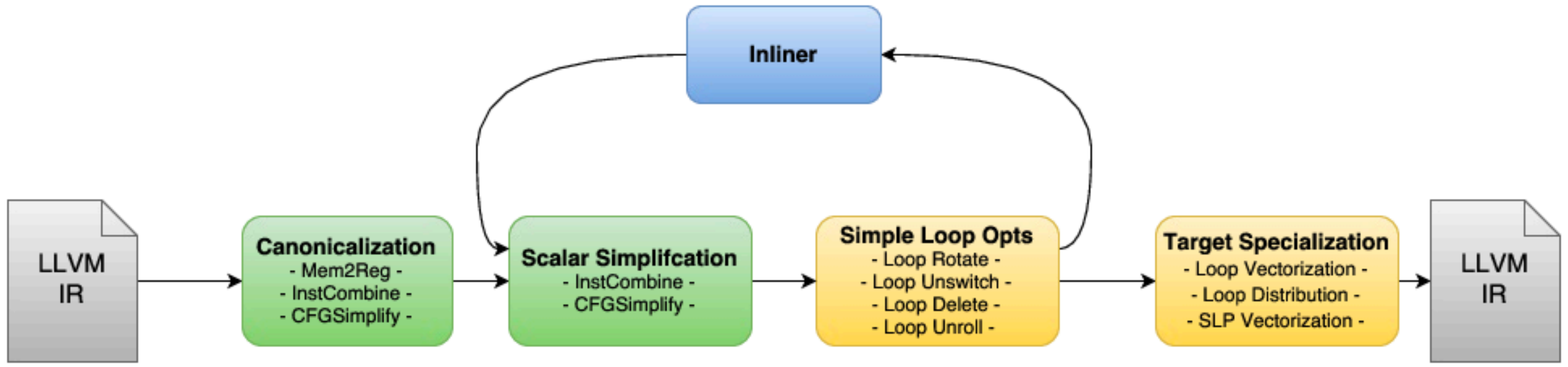
Components

- Sequence to sequence model
  - Encoder
  - Decoder

Sequence to Sequence Learning with Neural Networks
Sutskever, et. al (NIPS 2014)

Attention is all you need
Vaswani, et. al (NIPS 2017)

Programming Language Processing



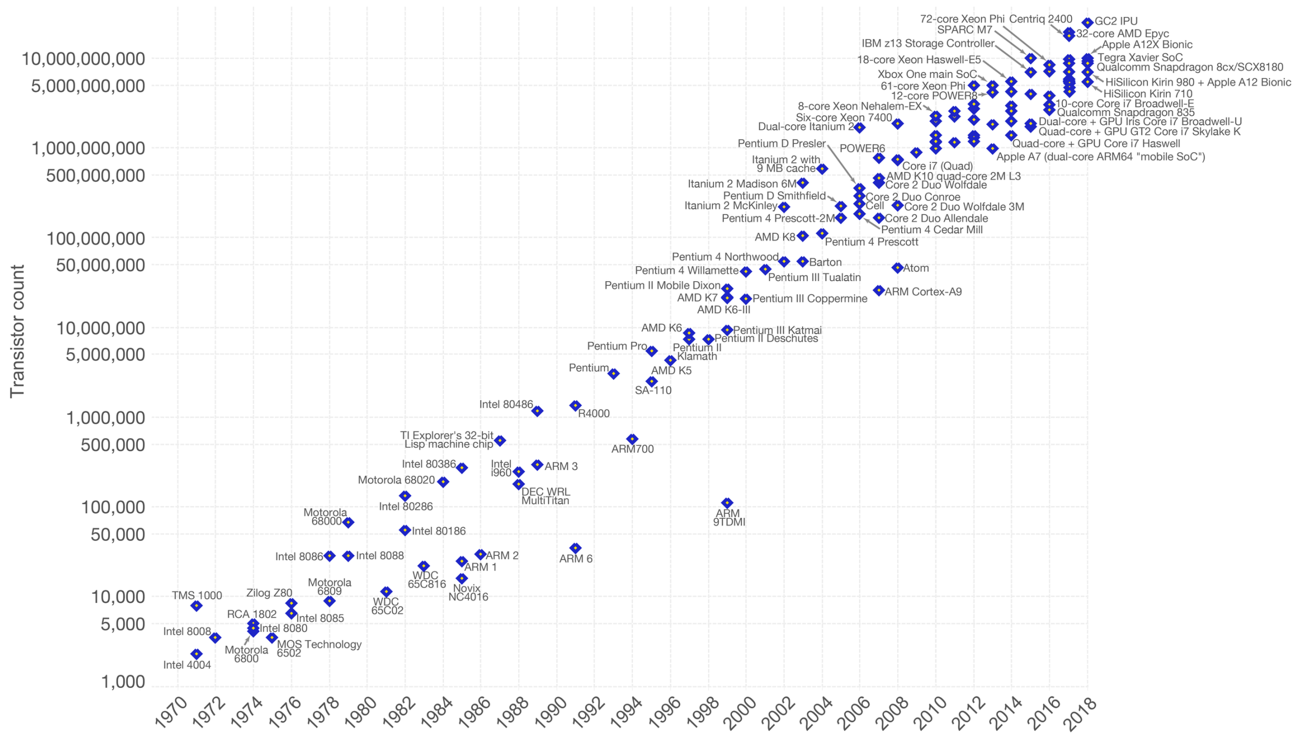
LLVM Compiler Flow

Components

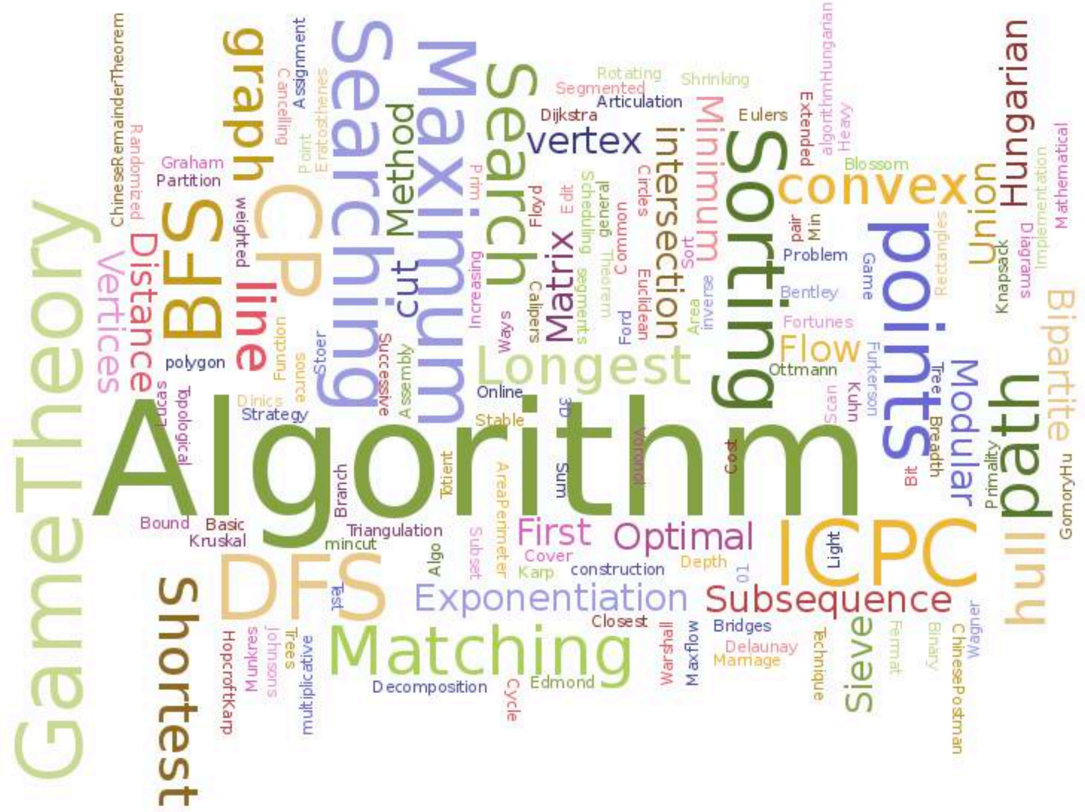
- Lexer
- Parser
- Semantic Analyser
- Intermediate Code Generator
- Code optimizer
- Low Level Code Generator

What Changed in 30 years?

- More Computing Power
  - Faster CPUs with multicores, GPUs & accelerators
  - More memory and storage
  - Cloud computing



- Better/Faster Algorithms
  - Integer Liner Programming
  - SMT solvers
  - Theorem provers
  - Deep Neural Networks



- More Data
  - Larger, better curated, and globally available data sets

```
=e.fn.button;e.fn.button=function(n){return this.each(function(){var r
data("button",i=new t(this,s)),n=="toggle"?i.toggle():n&&i.setState(n)
button.Constructor=t,e.fn.button.noConflict=function(){return e.fn.but
[data-toggle^=button]",function(t){var n=e(t.target);n.hasClass("btn")
!function(e){"use strict";var t=function(t,n){this.$element=e(t),this.
options=n,this.options.pause=="hover"&&this.$element.on("mouseenter",e
his))};t.prototype={cycle:function(t){return t||(this.paused=!1),this.
interval&&!this.paused&&(this.interval=setInterval(e.proxy(this.next,t
return this.$active=this.$element.find(".item.active"),this.$items=thi
$active)},to:function(t){var n=this.getActiveIndex(),r=this;if(t>this.
$element.one("slid",function(){r.to(t)}):n==t?this.pause().cycle():thi
t){return t||(this.paused=!0),this.$element.find(".next, .prev").lengt
support.transition.end),this.cycle(!0)),clearInterval(this.interval),t
eturn;return this.slide("next")},prev:function(){if(this.sliding)retur
=this.$element.find(".item.active"),i=n||r[t](),s=this.interval,o=t=="
his.sliding=!0,s&&this.pause(),i=i.length?i:this.$element.find(".item"
f(i.hasClass("active"))return;this.$indicators.length&&(this.$indicato
"slid",function(){var t=e(a.$indicators.children()[a.getActiveIndex()]
$element.hasClass("slide")){this.$element.trigger(f);if(f.isDefaultPre
i.addClass(o),this.$element.one(e.support.transition.end,function(){i.
removeClass(["active",o].join(" ")),a.sliding=!1,setTimeout(function()
```

Bringing the Compiler Technology to the 21st Century

- Use more compute power
  - Why not use parallelism, GPUs and the cloud?
- Use better algorithms
  - Complexity of compiler optimizations is due to search
    - Can we search better, faster, simpler?
- Use data better
  - From using data for testing and intuition to learning from data
    - From running SPEC benchmarks to Github mining

The Structure of a Modern Compiler

Build with ancient technology

- A command line tool
- Running on the developer's workstation (or a local cluster)
- With a single CPU thread
- Sequential execution of passes
  - Prog → AST →IR_1 →…→ IR_n → Assembly

Impact

- Compile time still matters
  - No expensive analyses
  - Limited to no global optimizations
- Memory footprint still matters
  - Highly optimized data structures
  - Limited to no global optimizations
- No path to learn and improve

CaaS: Compilation as a Service

- Access to unlimited processing power
- Access to accelerators
- Access to unlimited memory and storage
- Use of modern system building methods and frameworks
- Ability to learn from everyone and improve over time

- Build LLVM in 90 seconds (vs 10 minutes)
  - Using llama -- A CLI for outsourcing computation to AWS Lambda
  - Many related works of General Offloading
    Eg: “From Laptop to Lambda..” USENIX 2019

Analysis & Transformations with Serverless

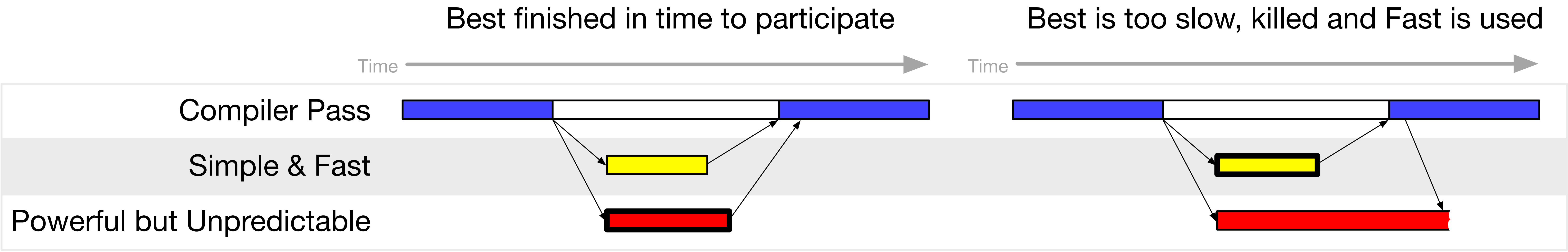
- Most of the compiler is parallel and stateless
  - Passes → Files → Functions → Basic Blocks → Statements
- Fits well to the serverless computing paradigm
- Scale-out for to match any program size
  - Size of functions and basic blocks are normally constant
  - Constant compile time for any size program!

Interprocedural Analysis with Distributed Graph Processing

- Compilers rarely/never do global analysis on real applications
  - Eg: Interprocedural type specialization, constant prop., inlining etc.
  - Too slow or too much memory consumption
  - Many papers written, never used in practice :(
- On the cloud, fits nicely to distributed graph processing
  - Many frameworks available, scales well, may even use GPUs

Expensive and Unpredictable Analysis using Redundancy Techniques used in Latency Reduction

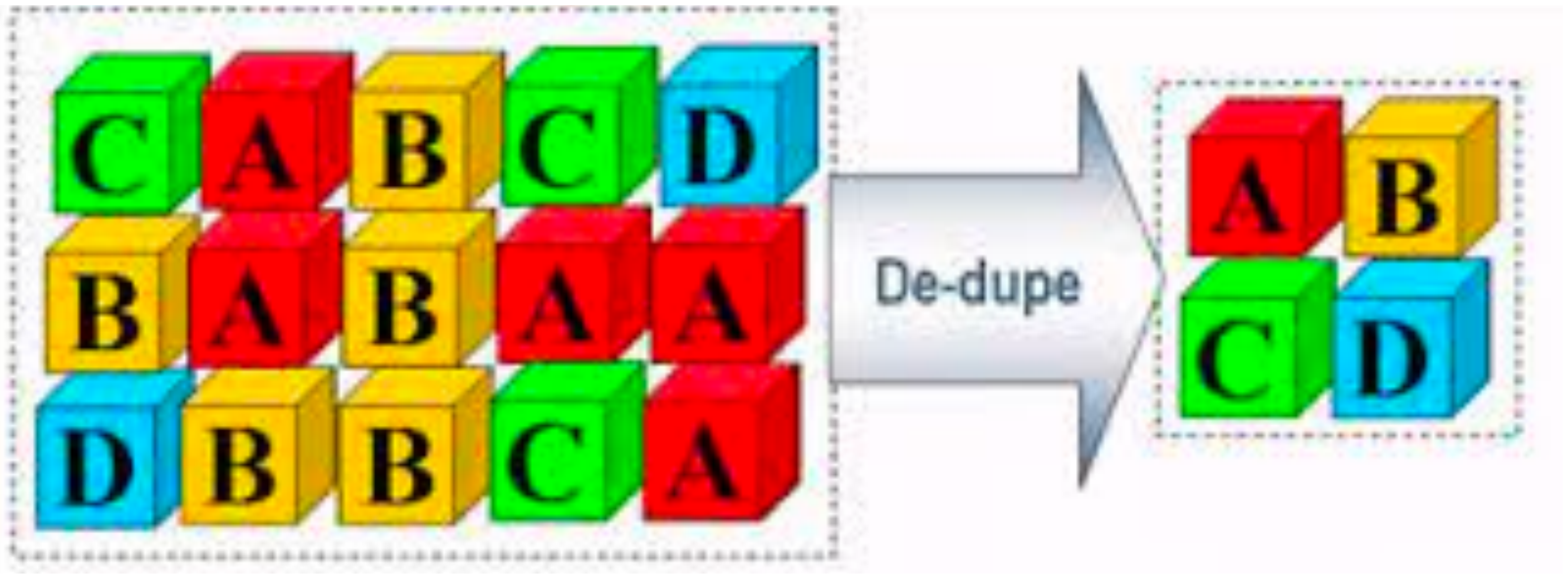
- Production compilers don't use expensive analyses or analyses with unpredictable runtimes
  - Ex: Polyhedral analysis, program synthesis etc.
  - Many papers written, never used in practice :(
- Many modern systems use redundancy to hide tail latency
- Compilers can use redundancy to incorporate powerful but unpredictable analyses


Best finished in time to participate
Best is too slow, killed and Fast is used
Time
Time
Compiler Pass
Simple & Fast
Powerful but Unpredictable

Ansel et. al "SiblingRivalry: Online Autotuning Through Local Competitions." [CASES'12]

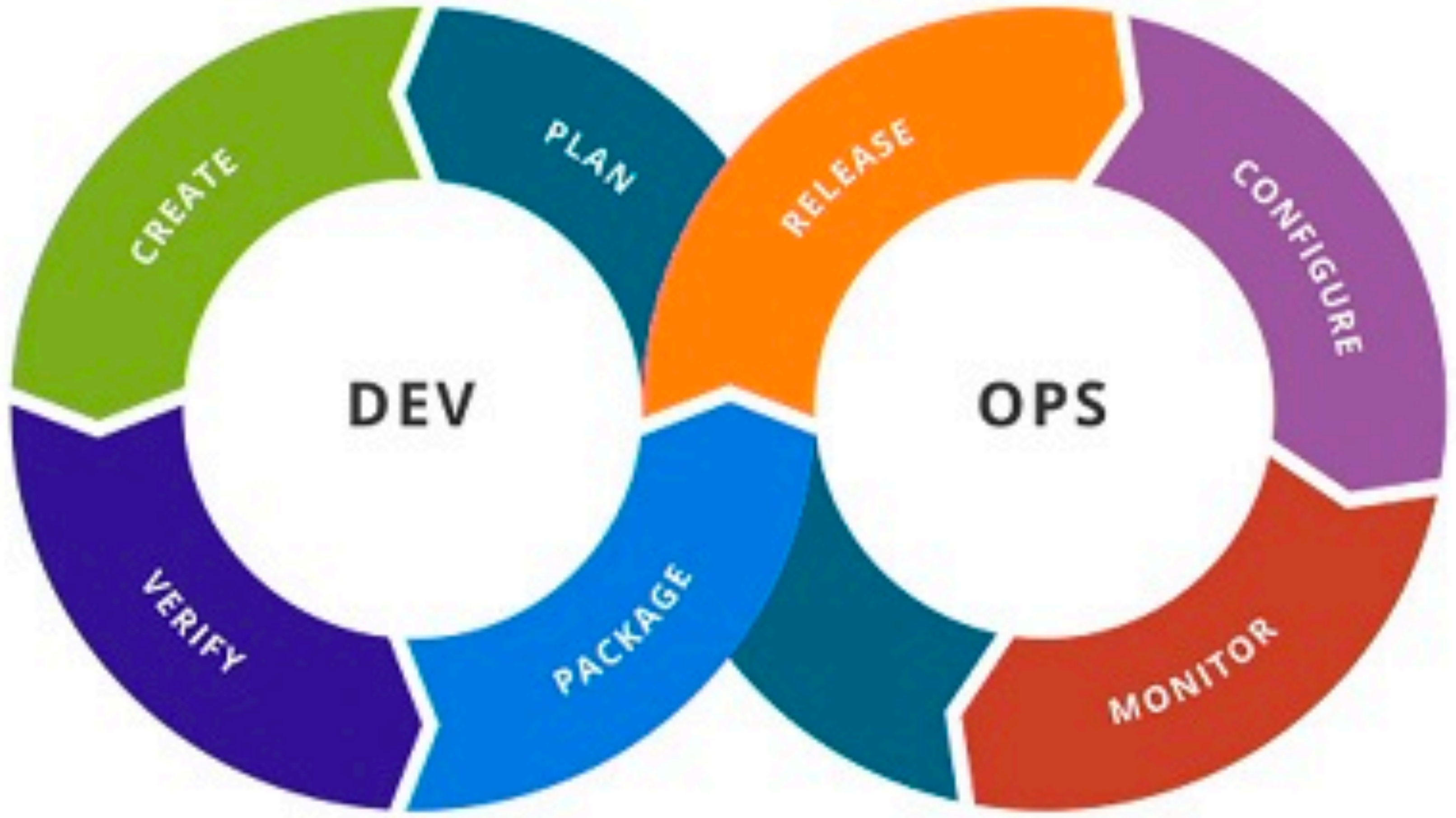
Overall Cost Reduction with Deduplication

- Reuse of compiled files is nothing new
  - Makefiles only compile changed files and their dependencies
- If most programmers use a single CaaS system for compiling
  - Each run is a small modification to a one seen before
  - Most probably exactly the same program as seen before
- Memoization can drastically reduce the cost of compilation
  - As done by many SaaS systems for storage



Centrally Collected Data for Continuous Improvement

- CaaS will see many programs
  - Usage is clear
  - Failures are obvious
- Can use the usage information for continuous improvement

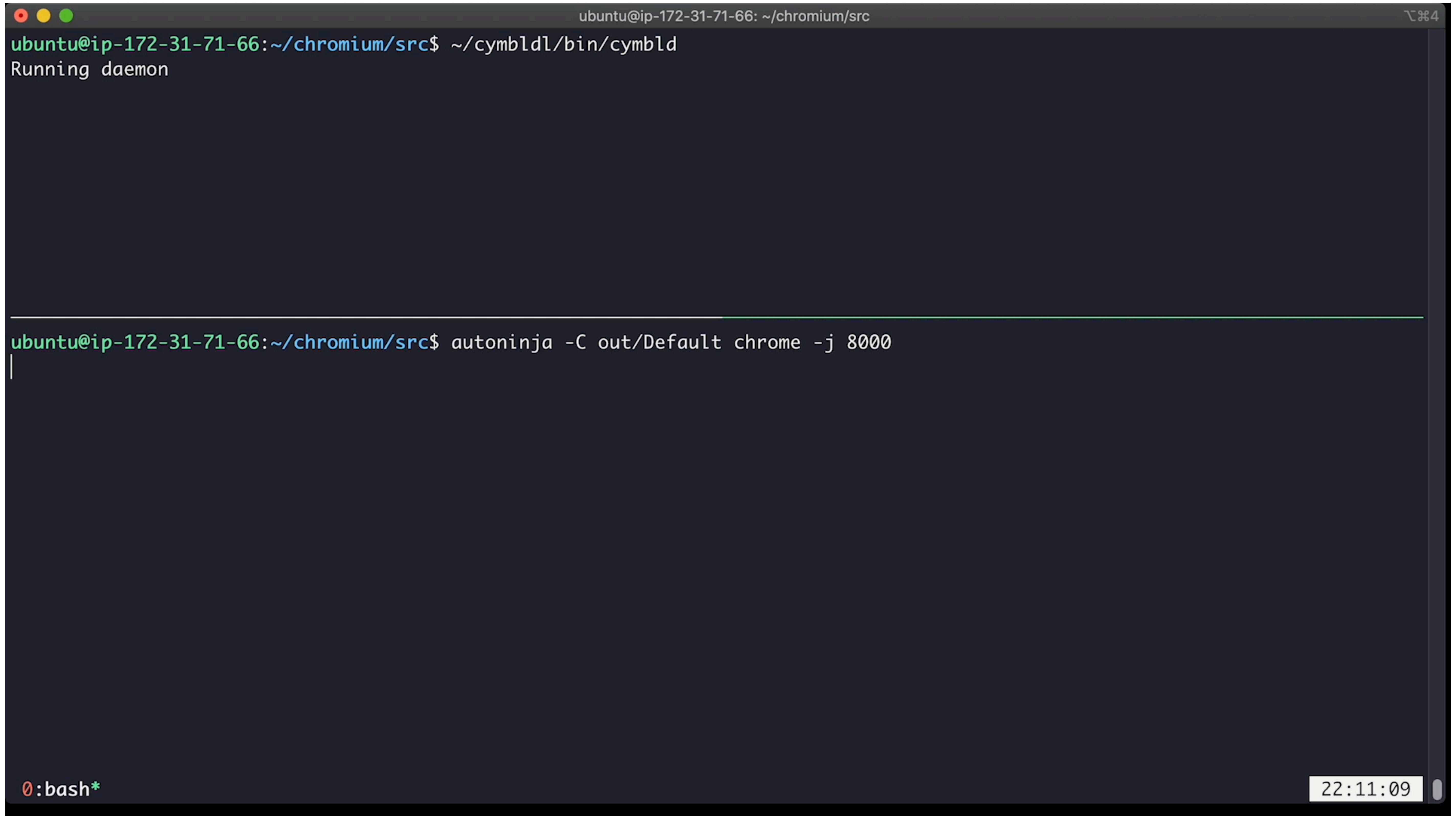


Existing Cloud Compilation Infrastructure

	Compatibility	Parallelism	Caching	Extensible
Bazel	✗ Must use build system	✗ Requires user cluster*	? Per-codebase	✗ Only Bazel Tasks
DistCC	? Models compile command	✗ Requires user cluster	✗ Limited or none	✗ Wrapper for cc
Goma	? Models compile command	✗ Requires user cluster	? Per-codebase	✗ Wrapper for cc
gg	✗ Models all build commands	✓ On-demand compute	? Per-invocation	✗ Wrapper for cc

cymbl in action

- Hackable drop in replacement for existing compilers:
  - Start the daemon, set desired parallelism and let it run!



```
ubuntu@ip-172-31-71-66:~/chromium/src$ ~/cymbldl/bin/cymbld
Running daemon
```

```
ubuntu@ip-172-31-71-66:~/chromium/src$ autoninja -C out/Default chrome -j 8000
```

cymbl

- Integrate remote execution into the compiler itself
  - Use in any existing build system & “model” will always be perfect
  - Compiler-level information of source code => better task normalization and more effective cache
  - Merged remote execution and compilation => reduced latency and total build time
  - Hackable! Re-use (or augment) any compilation phase
- Cloud functions provide parallelism without user-level infrastructure

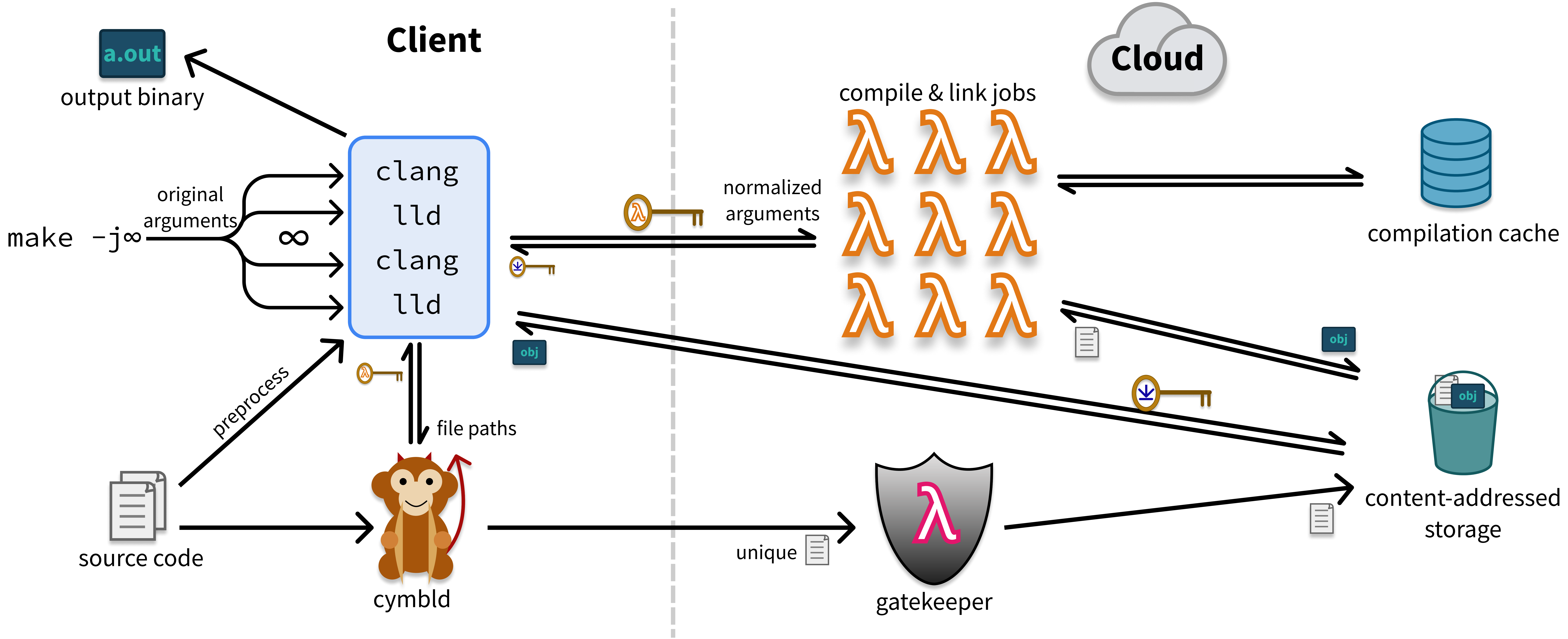
cymbl Smoke Test

21 Hour Google Chrome Build

↓

4.5 minutes with cymbl

cymbl Workflow

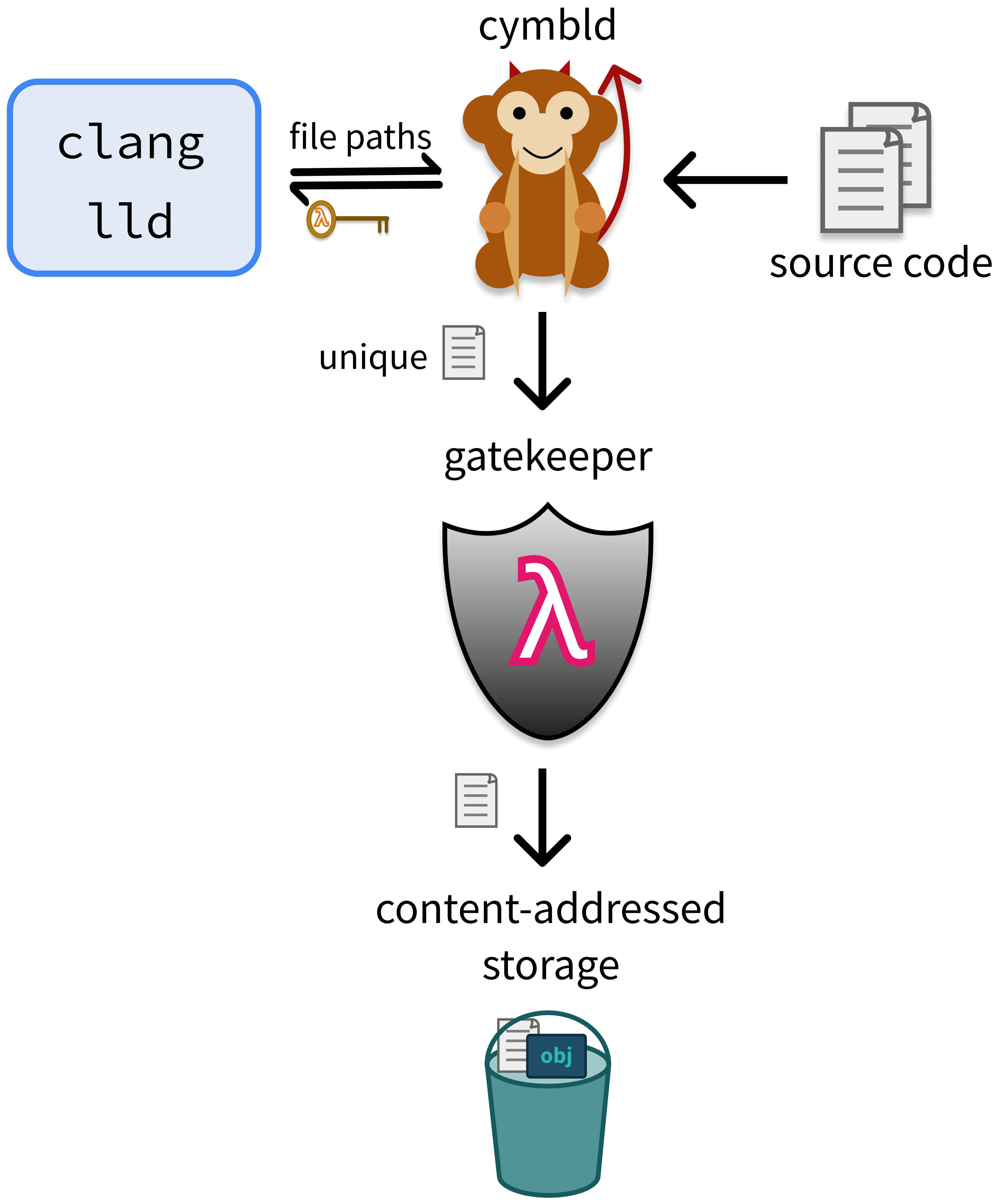


cymbl Daemon

- Use a shared daemon process (cymbld) to avoid duplicate uploads across compilation jobs and manage authentication

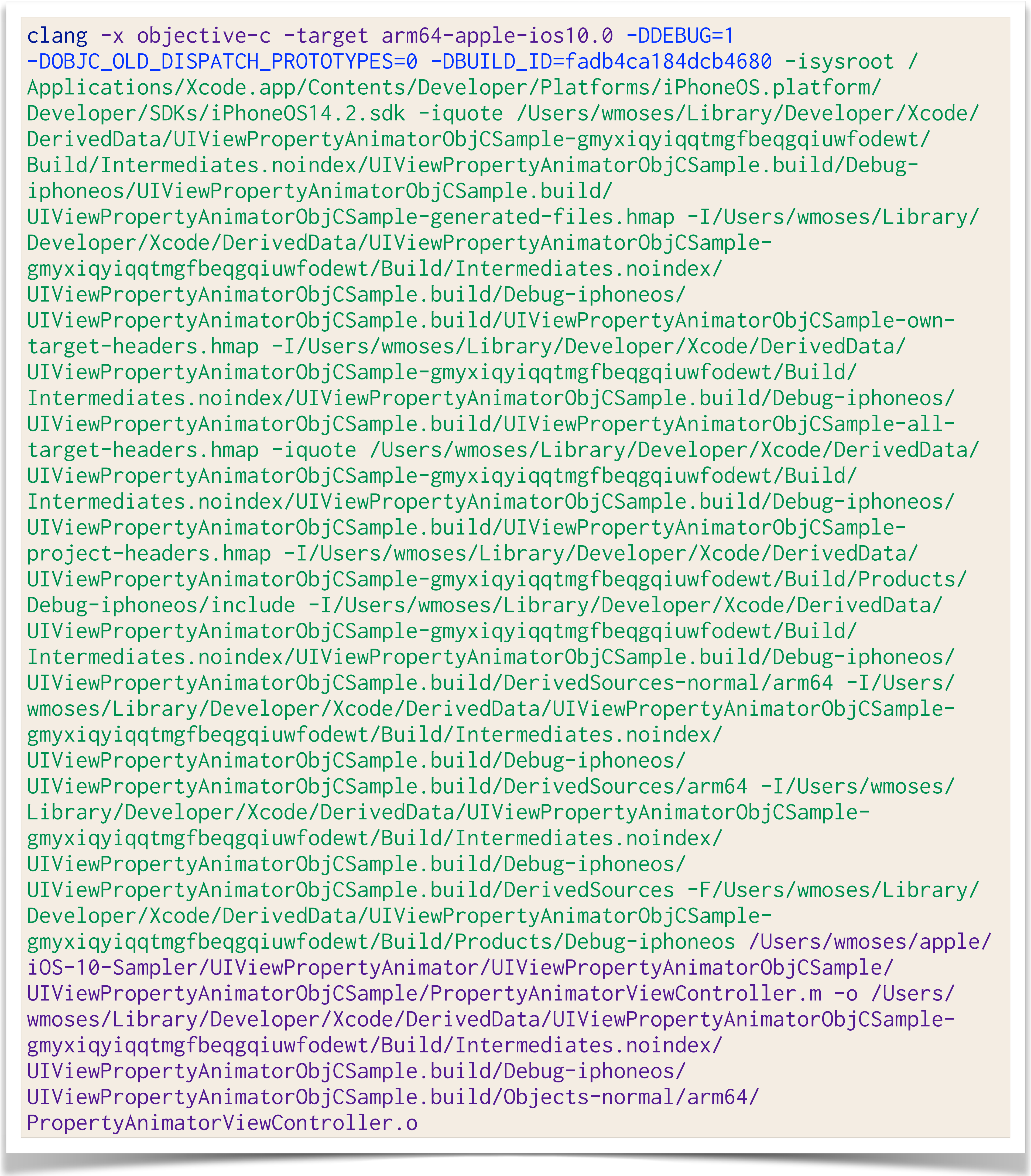
- clang and lld processes send dependency file paths to daemon through IPC.

- cymbld hashes, dedups, and batches before querying the server for cache misses

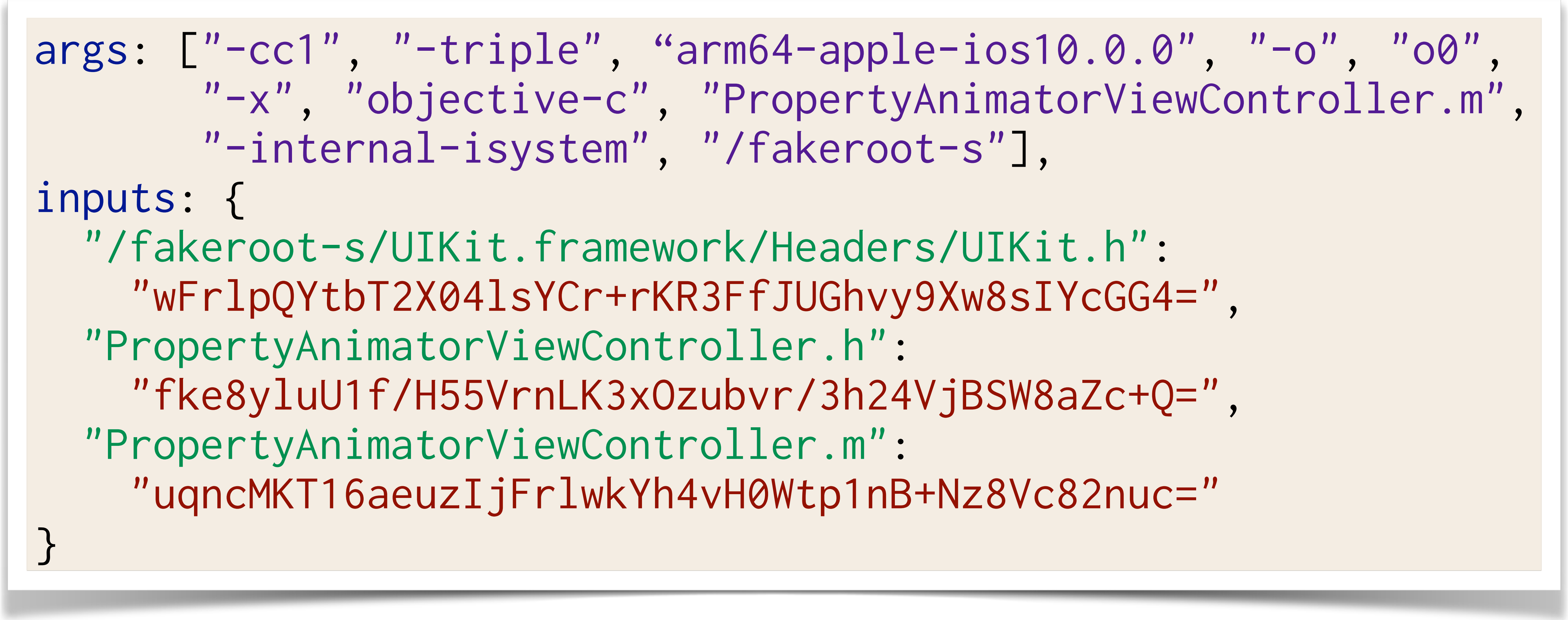


cymbl Normalization

- Identify required arguments & inputs (**purple**)
- Remove unused defines (**blue**)
- Normalize include paths (**green**)
- Provide map of exactly what files are used with their corresponding hash in content-addressable storage (**red**)



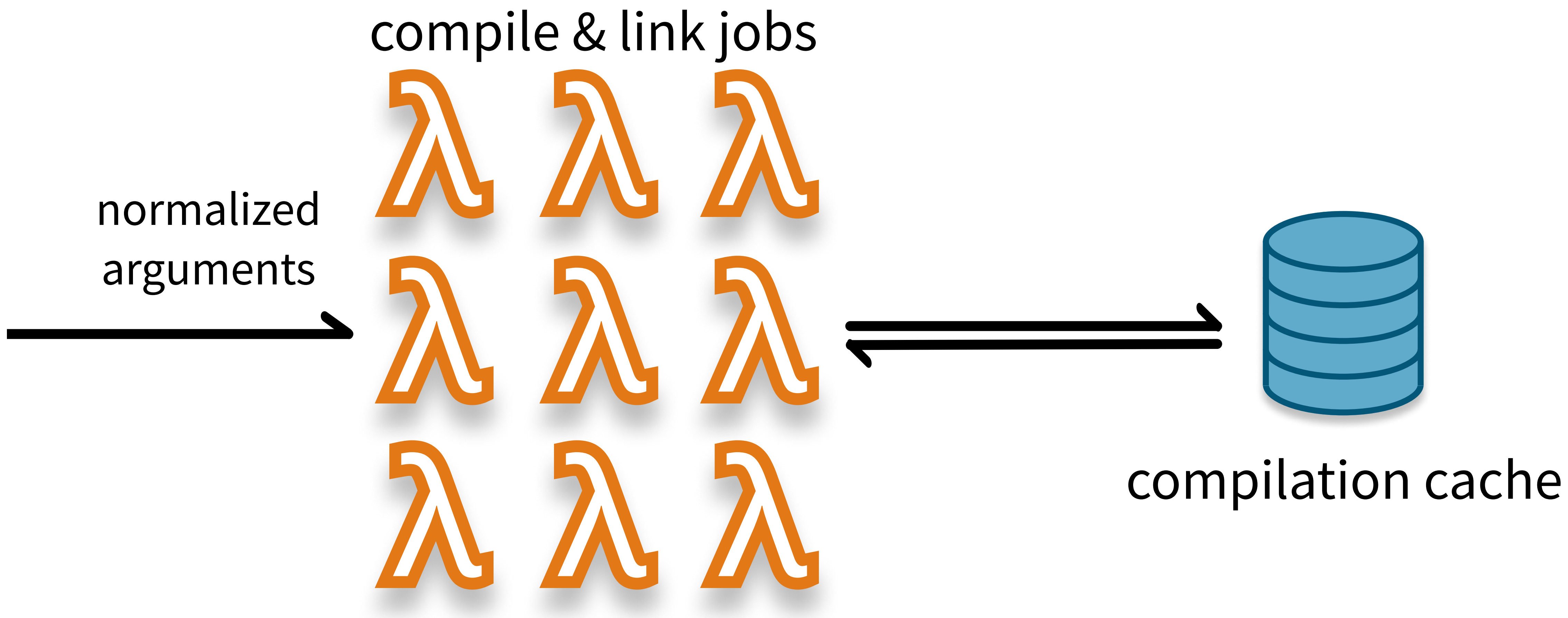
```
clang -x objective-c -target arm64-apple-ios10.0 -DDEBUG=1
-DOBJC_OLD_DISPATCH_PROTOTYPES=0 -DBUILD_ID=fadb4ca184dcb4680 -isysroot /
Applications/Xcode.app/Contents/Developer/Platforms/iPhoneOS.platform/
Developer/SDKs/iPhoneOS14.2.sdk -iquote /Users/wmoses/Library/Developer/Xcode/
DerivedData/UIViewPropertyAnimatorObjCSample-gmyxiqyiqqtmgfbeqgqiuwfodewt/
Build/Intermediates.noindex/UIViewPropertyAnimatorObjCSample.build/Debug-
iphoneos/UIViewPropertyAnimatorObjCSample.build/
UIViewPropertyAnimatorObjCSample-generated-files.hmap -I/Users/wmoses/Library/
Developer/Xcode/DerivedData/UIViewPropertyAnimatorObjCSample-
gmyxiqyiqqtmgfbeqgqiuwfodewt/Build/Intermediates.noindex/
UIViewPropertyAnimatorObjCSample.build/Debug-iphoneos/
UIViewPropertyAnimatorObjCSample.build/UIViewPropertyAnimatorObjCSample-own-
target-headers.hmap -I/Users/wmoses/Library/Developer/Xcode/DerivedData/
UIViewPropertyAnimatorObjCSample-gmyxiqyiqqtmgfbeqgqiuwfodewt/Build/
Intermediates.noindex/UIViewPropertyAnimatorObjCSample.build/Debug-iphoneos/
UIViewPropertyAnimatorObjCSample.build/UIViewPropertyAnimatorObjCSample-all-
target-headers.hmap -iquote /Users/wmoses/Library/Developer/Xcode/DerivedData/
UIViewPropertyAnimatorObjCSample-gmyxiqyiqqtmgfbeqgqiuwfodewt/Build/
Intermediates.noindex/UIViewPropertyAnimatorObjCSample.build/Debug-iphoneos/
UIViewPropertyAnimatorObjCSample.build/UIViewPropertyAnimatorObjCSample-
project-headers.hmap -I/Users/wmoses/Library/Developer/Xcode/DerivedData/
UIViewPropertyAnimatorObjCSample-gmyxiqyiqqtmgfbeqgqiuwfodewt/Build/Products/
Debug-iphoneos/include -I/Users/wmoses/Library/Developer/Xcode/DerivedData/
UIViewPropertyAnimatorObjCSample-gmyxiqyiqqtmgfbeqgqiuwfodewt/Build/
Intermediates.noindex/UIViewPropertyAnimatorObjCSample.build/Debug-iphoneos/
UIViewPropertyAnimatorObjCSample.build/DerivedSources-normal/arm64 -I/Users/
wmoses/Library/Developer/Xcode/DerivedData/UIViewPropertyAnimatorObjCSample-
gmyxiqyiqqtmgfbeqgqiuwfodewt/Build/Intermediates.noindex/
UIViewPropertyAnimatorObjCSample.build/Debug-iphoneos/
UIViewPropertyAnimatorObjCSample.build/DerivedSources/arm64 -I/Users/wmoses/
Library/Developer/Xcode/DerivedData/UIViewPropertyAnimatorObjCSample-
gmyxiqyiqqtmgfbeqgqiuwfodewt/Build/Intermediates.noindex/
UIViewPropertyAnimatorObjCSample.build/Debug-iphoneos/
UIViewPropertyAnimatorObjCSample.build/DerivedSources -F/Users/wmoses/Library/
Developer/Xcode/DerivedData/UIViewPropertyAnimatorObjCSample-
gmyxiqyiqqtmgfbeqgqiuwfodewt/Build/Products/Debug-iphoneos /Users/wmoses/apple/
iOS-10-Sampler/UIViewPropertyAnimator/UIViewPropertyAnimatorObjCSample/
UIViewPropertyAnimatorObjCSample/PropertyAnimatorViewController.m -o /Users/
wmoses/Library/Developer/Xcode/DerivedData/UIViewPropertyAnimatorObjCSample-
gmyxiqyiqqtmgfbeqgqiuwfodewt/Build/Intermediates.noindex/
UIViewPropertyAnimatorObjCSample.build/Debug-iphoneos/
UIViewPropertyAnimatorObjCSample.build/Objects-normal/arm64/
PropertyAnimatorViewController.o
```



```
args: ["-cc1", "-triple", “arm64-apple-ios10.0.0", "-o", "o0",
       "-x", "objective-c", "PropertyAnimatorViewController.m",
       "-internal-isystem", "/fakeroot-s"],
inputs: {
  "/fakeroot-s/UIKit.framework/Headers/UIKit.h":
    "wFrlpQYtbT2X04lsYCr+rKR3FfJUGhvy9Xw8sIYcGG4=",
  "PropertyAnimatorViewController.h":
    "fke8yluU1f/H55VrnLK3xOzubvr/3h24VjBSW8aZc+Q=",
  "PropertyAnimatorViewController.m":
    "uqncMKT16aeuzIjFrlwkYh4vH0Wtp1nB+Nz8Vc82nuc="
}
```

cymbl Compilation & Caching

- Before compiling, check if it has previously been compiled (perhaps by another user)



- When downloading final results, re-localize file paths and debug information

- Produces same result as local compilation, now taking advantage of parallelism and redundancy

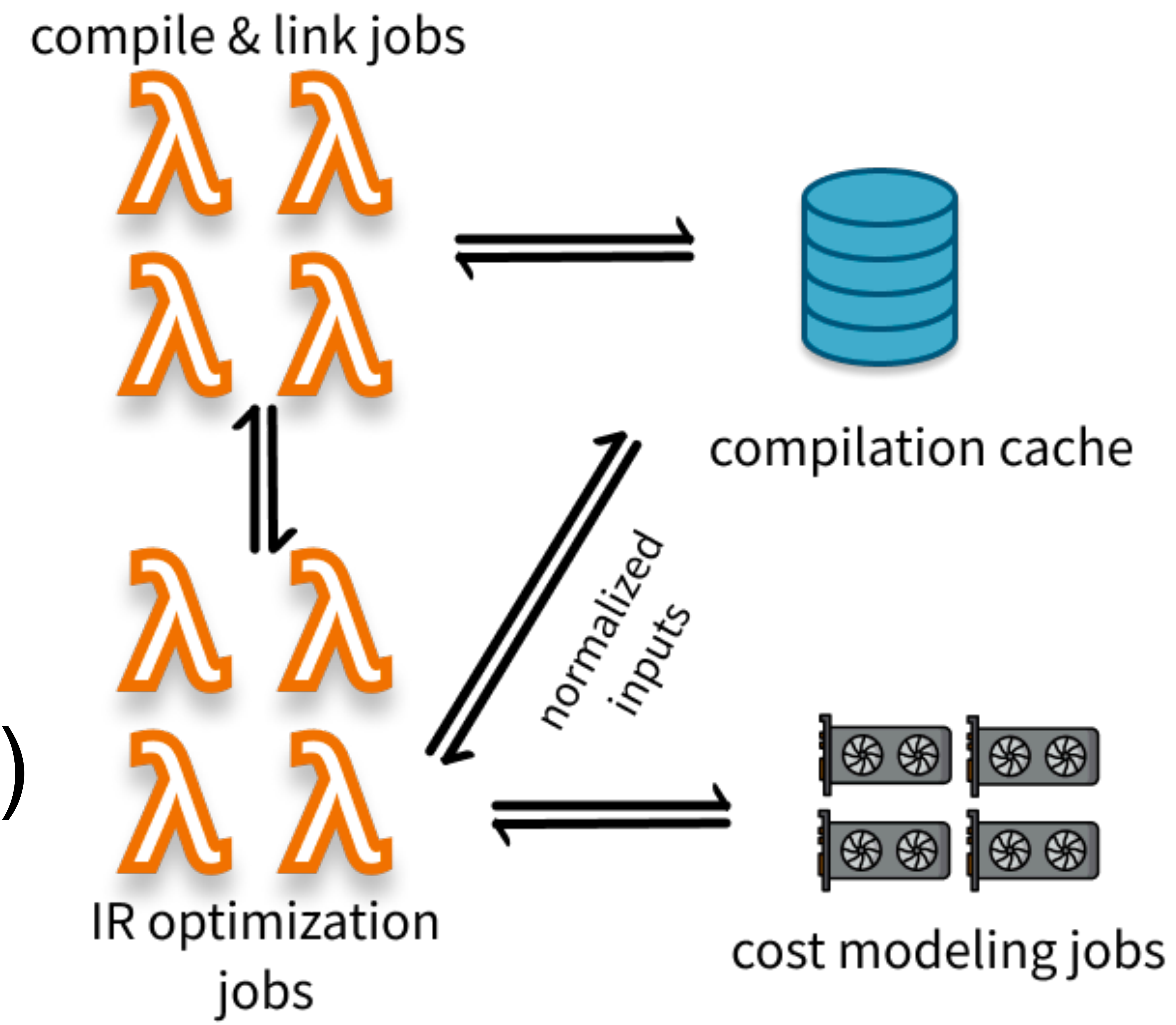
cymbl Performance

	1-Core	96-Core	Cymbl	Cached Cymbl	gg*
FFmpeg	9.43	0.48	0.53	0.04	0.73*
InkScape	39.96	1.06	1.12	0.25	1.45*
Clang	183.55	4.32	2.42	0.36	
Chrome	1302.65	25.71	6.99	4.42	18.92*

*gg results taken from paper, due to inability to reproduce results

cymbl Advanced Compilation*

- Leverage parallelism and execution environment of the cloud to extend the capabilities of compilers!
- Simultaneously run multiple optimization pipelines
- Use cost modeling (or real machines) to predict the runtime of programs
- Leverage profiling information across all users to improve models



* Currently in progress

The Future of Compilation is Cloud-Based

- Embarrassingly parallel structure makes compilation an excellent candidate for speedup with cloud resources
- Direct integration of the compiler and cloud infrastructure provides:
    - Easy-to-use in existing workflows
    - Reduced maintenance and engineering effort
    - Extensibility for novel capabilities