From Laptop to Lambda

Sadjad Fouladi, Francisco Romero, Dan Iter, Qian Li, Shuvo Chatterjee, Christos Kozyrakis, Matei Zaharia, Keith Winstein

Presented by Val Baturin

September 26, 2019

Outline

Present gg

A framework and a set of command-line tools that helps people execute everyday applications — e.g., software compilation, unit tests, video encoding, or object recognition — using thousands of parallel threads on a cloudfunctions service to achieve near-interactive completion times.

Overview of gg

With gg, applications express a job as a composition of lightweight OS containers that are individually transient (lifetimes of 1–60 seconds) and functional (each container is hermetically sealed and deterministic).

Related Word

Cluster-computation systems

Hadoop, Spark, Dryad, and CIEL

Container orchestrators

Docker Swarm and Kubernetes

outsourcing tools

distcc, icecc, and UCop

Rule-based workflow systems

make, CMake

Cloud-functions tools

ExCamera/mu, PyWren, and Spark-on-Lambda

Design and Implementation

- gg's Intermediate Representation
- Front-ends
- Back-ends

Design and Implementation

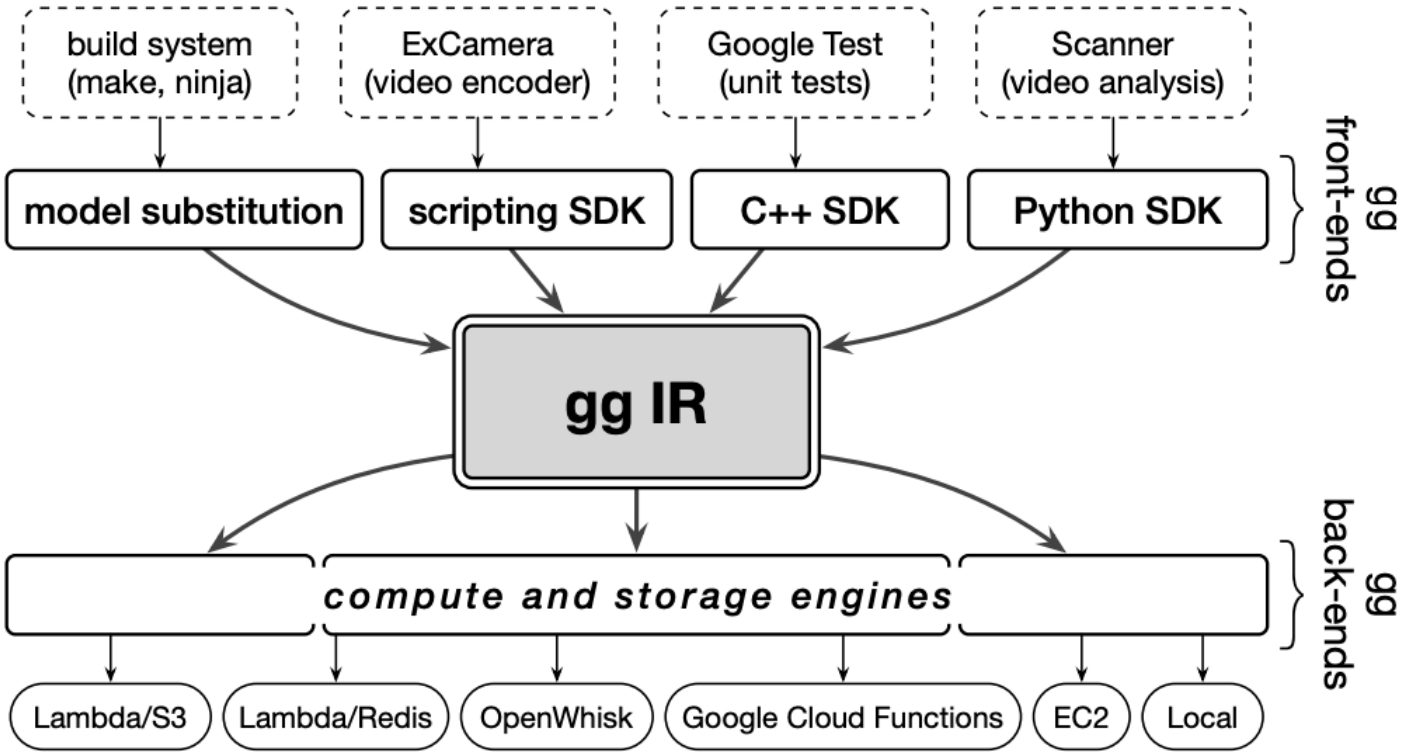

Figure 1: Current implementation

gg's Intermediate Representation

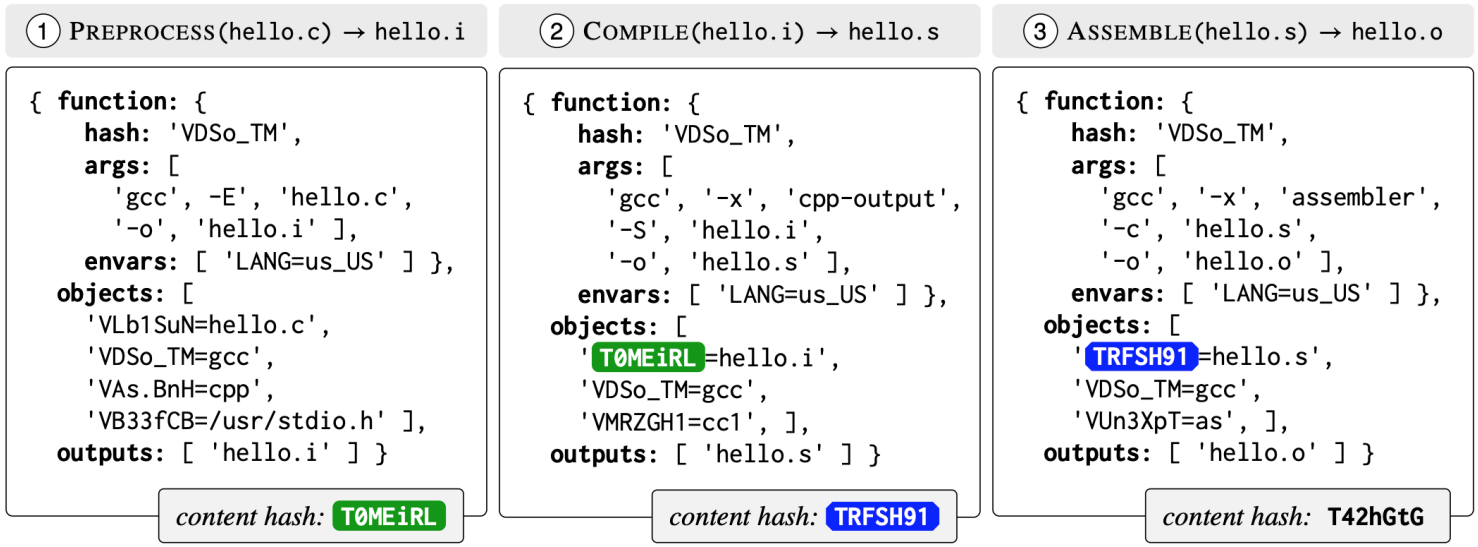

Figure 2: An example of gg IR consisting of three thunks

Back-ends

Storage engine

S3, Redis, and Google Cloud Storage

Execution engine

a local multicore machine, a cluster of remote VMs, AWS Lambda, Google Cloud Functions, and IBM Cloud Functions (OpenWhisk)

The coordinator

The main entry-point for executing a thunk

Applications

- Software Compilation
- Unit Testing
- Video Encoding
- Recursive Fibonacci

Application example

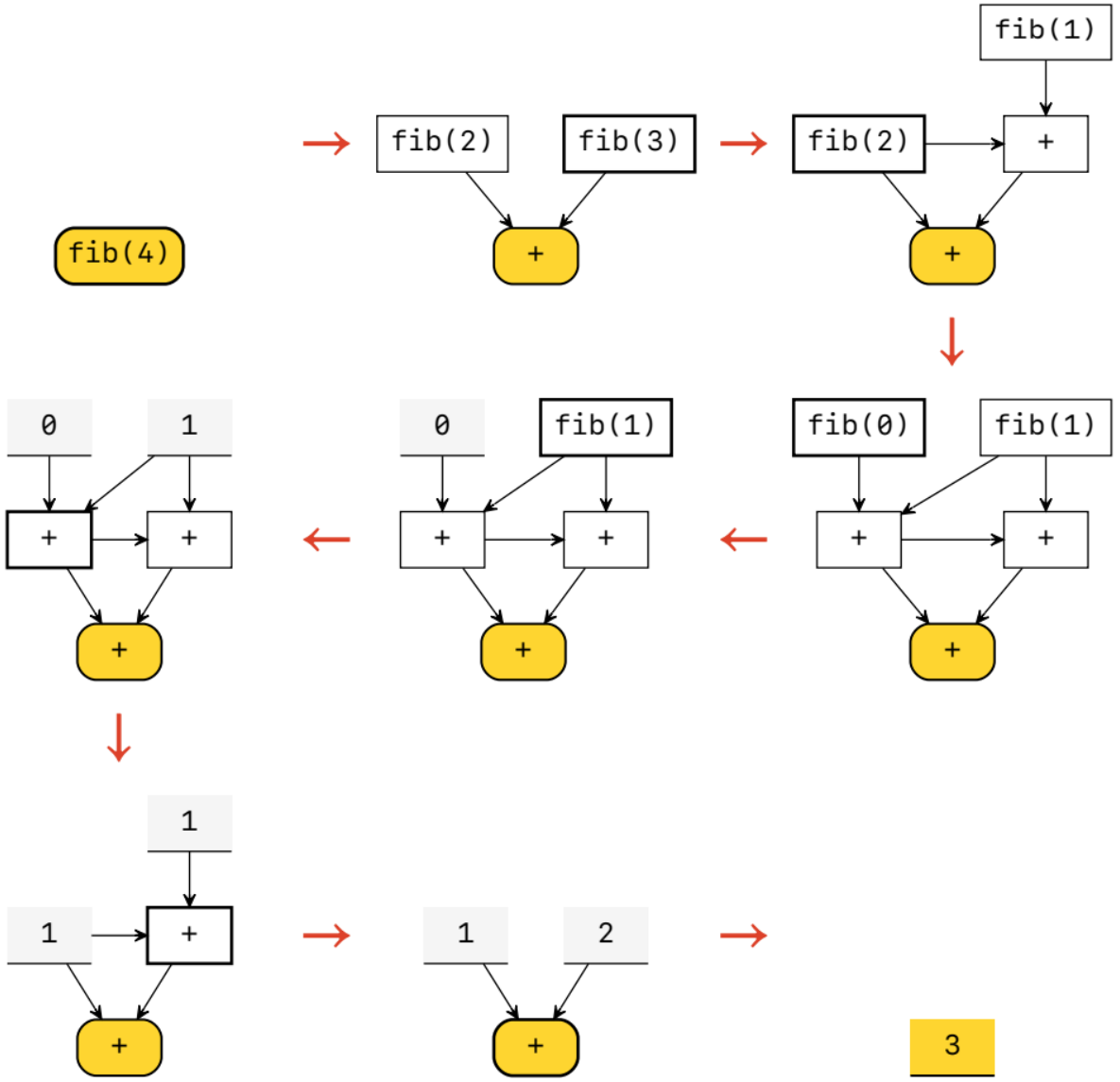

Figure 3: Evolution of the IR for a recursive Fibonacci application

Startup Overhead

1K trivial containers running "sleep 2"		
AWS Lambda	gg-λ	06s ± 01s
	PyWren	46s ± 08s
	Spark-on-Lambda	54s ± 21s
Google Kubernetes Engine	Kubernetes	03m 08s ± 03s

Figure 4: Comparison of completion time for running 1,000 sleep(2) tasks using four different systems. gg's lightweight design and implementation has less overhead than other systems

Real World Performance Example

		Local (make)		Distributed (icecc)		**Distributed (gg)**	
	Estimated SLoC	1 core	48 cores	48 cores	384 cores	384 cores	AWS Lambda
FFmpeg	1,200,000	06m 19s	20s	01m 03s	39s	40s	44s ± 04s
GIMP	800,000	06m 48s	49s	02m 35s	02m 38s	01m 26s	01m 38s ± 03s
Inkscape	600,000	32m 34s	01m 40s	06m 51s	06m 57s	01m 20s	01m 27s ± 07s
Chromium	24,000,000	15h 58m 20s	38m 11s	46m 01s	42m 18s	40m 57s	18m 55s ± 10s

Figure 5: Software compilation example

Limitations and Discussion

- Direct communication between workers
- Limited to CPU programs
- A gg DSL to program for the IR
- Why cloud functions?