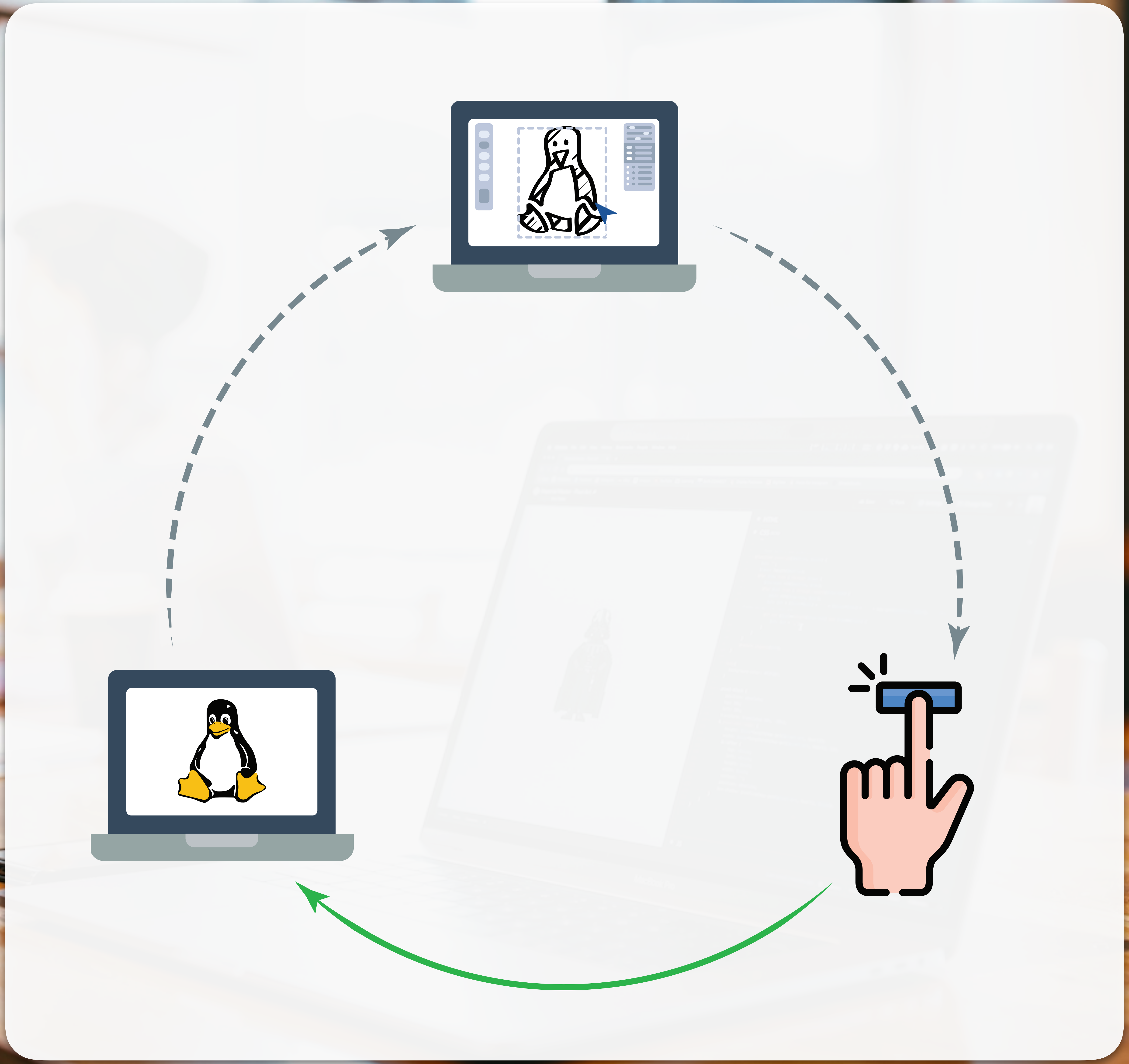
From Laptop to Lambda: Outsourcing Everyday Jobs to Thousands of Transient Functional Containers

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

Stanford University

https://snr.stanford.edu/gg

MacBook Pro



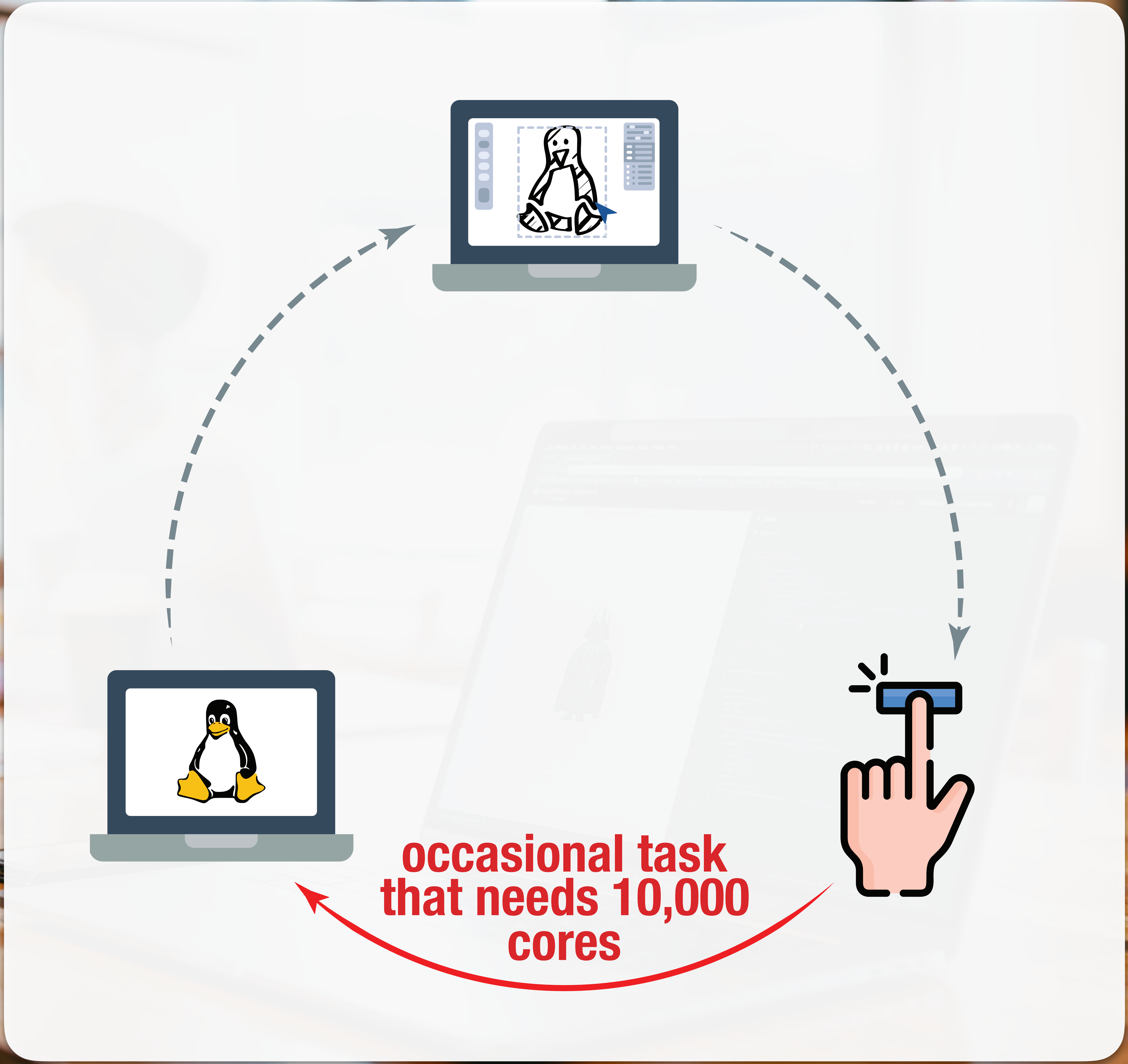
xkcd.com/303

Compiling **Google Chrome** takes **16 hours.**

Running unit tests for **LibVPX** takes **1.5 hours.**

Encoding a **15-minute 4K video** takes **7.5 hours.**

Rendering **a single frame** of **Monsters University** takes **29 hours.**



occasional task that needs 10,000 cores

Many others share this dream

- **Outsourcing computation:**
  distcc ('04), icecc ('11), UCop (ATC'10)

- **Cluster-computing frameworks:**
  Hadoop ('06), Dryad (EuroSys'07), CIEL (NSDI'11), Spark (NSDI'12)

- **Burst-parallel cloud functions:**
  ExCamera (NSDI'17), PyWren (SoCC'17), Sprocket (SoCC'18), Serverless MapReduce

But this dream is not a reality yet…

Limited speed-ups*, high costs, limited applicability, etc.

* *“Scalability! But at what COST?,” F. McSherry, M. Isard, and D. G. Murray, HotOS XV*

Enter gg

- **gg** is **a framework and a toolkit** that makes it practical to outsource **everyday applications** using thousands of parallel threads on cloud-functions services.

- We ported several latency-sensitive applications to run on gg:
  - **software compilation**
  - **unit testing**
  - **video encoding**
  - **object recognition**

Challenges of outsourcing applications to the cloud

① **Software dependencies must be managed**

② Roundtrips to the cloud hurt performance

③ Cloud functions are promising, but hard to use well

Challenge ①
Software dependencies must be managed

- With data flow frameworks like Spark, Hadoop and Dryad the **software dependencies** remain unmanaged.
- Need a warm cluster with everything (e.g., FFmpeg, ImageMagick, NumPy, TensorFlow, SQLite, etc.) preloaded.
- Not amenable to occasional one-off tasks.

A 10,000-core cluster on EC2 costs ~$400/hour

① Software dependencies must be managed

Thunk abstraction

- A **thunk** is a lightweight container.
  - It identifies an **executable**, along with its **arguments**, **environment** and **input data**.
- Data is named by the hash of its content.

```
{ function: {
    hash: ’VDSo_TM',
    args: [ ’gcc’, ’-E’, ’hello.c’, ’-o’, ’hello.i' ],
    envars: [ ’LANG=us_US’ ] },

  objects: [
    ’VDSo_TM=gcc’,
    ’VLb1SuN=hello.c’,
    ’VOHGODN=/usr/include/stdlib.h’,
    ’VB33fCB=/usr/include/stdio.h’ ],

  outputs: [ ’hello.i’ ]
}
```

An example thunk: Preprocessing a C source file.

```
{ function: {
    hash: ’VDSo_TM',
    args: [ ’gcc’, ’–E’, ’hello.c’, ’–o’, ’hello.i' ],
    envars: [ ’LANG=us_US’ ] },

  objects: [
    ’VDSo_TM=gcc’,
    ’VLb1SuN=hello.c’,
    ’VOHGODN=/usr/include/stdlib.h’,
    ’VB33fCB=/usr/include/stdio.h’ ],

  outputs: [ ’hello.i’ ]
}
```

An example thunk: Preprocessing a C source file.

```
{ function: {
    hash: ’VDSo_TM',
    args: [ ’gcc’, ’-E’, ’hello.c’, ’-o’, ’hello.i' ],
    envars: [ ’LANG=us_US’ ] },

  objects: [
    ’VDSo_TM=gcc’,
    ’VLb1SuN=hello.c’,
    ’VOHGODN=/usr/include/stdlib.h’,
    ’VB33fCB=/usr/include/stdio.h’ ],

  outputs: [ ’hello.i’ ]
}
```

An example thunk: Preprocessing a C source file.

Challenges of outsourcing applications to the cloud

① Software dependencies must be managed
→ lightweight containers (thunks)

② **Roundtrips to the cloud hurt performance**

③ Cloud functions are promising, but hard to use well

Challenge ②
Roundtrips to the cloud hurt performance

- Current application-specific outsourcing tools perform best over **fast networks:**
  - distcc
  - icecc (Icecream)
  - Utility Coprocessor (UCop) (ATC'10)
- **The laptop is in the driver's seat.**

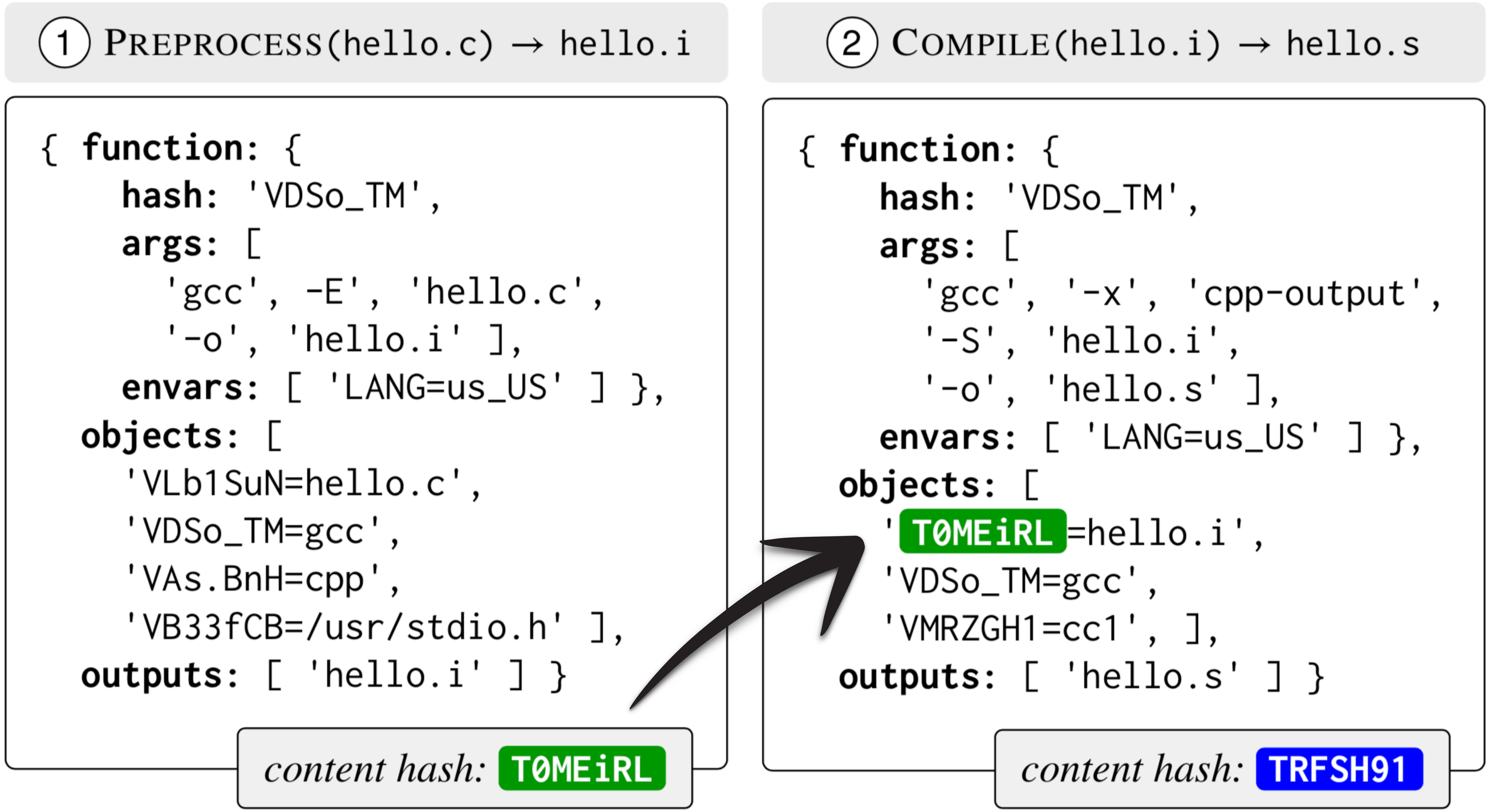
```
{ function: {
    hash: ’VDSo_TM',
    args: [ ’gcc’, ’-E’, ’hello.c’, ’-o’, ’hello.i' ],
    envars: [ ’LANG=us_US’ ] },

  objects: [
    ’VDSo_TM=gcc’,
    ’VLb1SuN=hello.c’,
    ’VOHGODN=/usr/include/stdlib.h’,
    ’VB33fCB=/usr/include/stdio.h’ ],

  outputs: [ ’hello.i’ ]
}
```

content hash: **T0MEiRL**

An example thunk: Preprocessing a C source file.

② Roundtrips to the cloud hurt performance

Containers can reference each other’s outputs



① PREPROCESS(hello.c) → hello.i

```
{ function: {
    hash: 'VDSo_TM',
    args: [
      'gcc', -E', 'hello.c',
      '-o', 'hello.i' ],
    envars: [ 'LANG=us_US' ] },
  objects: [
    'VLb1SuN=hello.c',
    'VDSo_TM=gcc',
    'VAs.BnH=cpp',
    'VB33fCB=/usr/stdio.h' ],
  outputs: [ 'hello.i' ] }
```

content hash: T0MEiRL

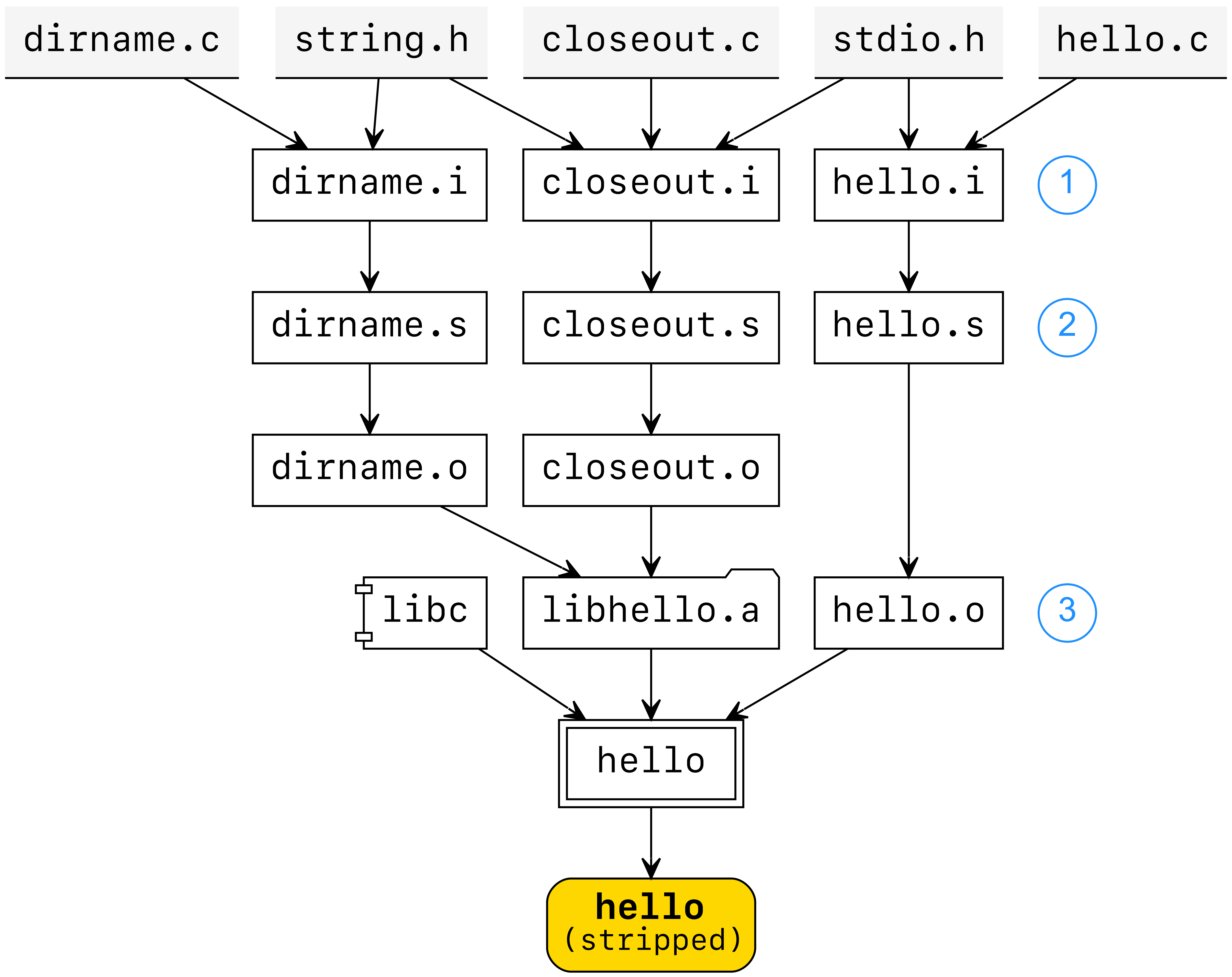
② COMPILE(hello.i) → hello.s

```
{ function: {
    hash: 'VDSo_TM',
    args: [
      'gcc', '-x', 'cpp-output',
      '-S', 'hello.i',
      '-o', 'hello.s' ],
    envars: [ 'LANG=us_US' ] },
  objects: [
    'T0MEiRL=hello.i',
    'VDSo_TM=gcc',
    'VMRZGH1=cc1', ],
  outputs: [ 'hello.s' ] }
```

content hash: TRFSH91

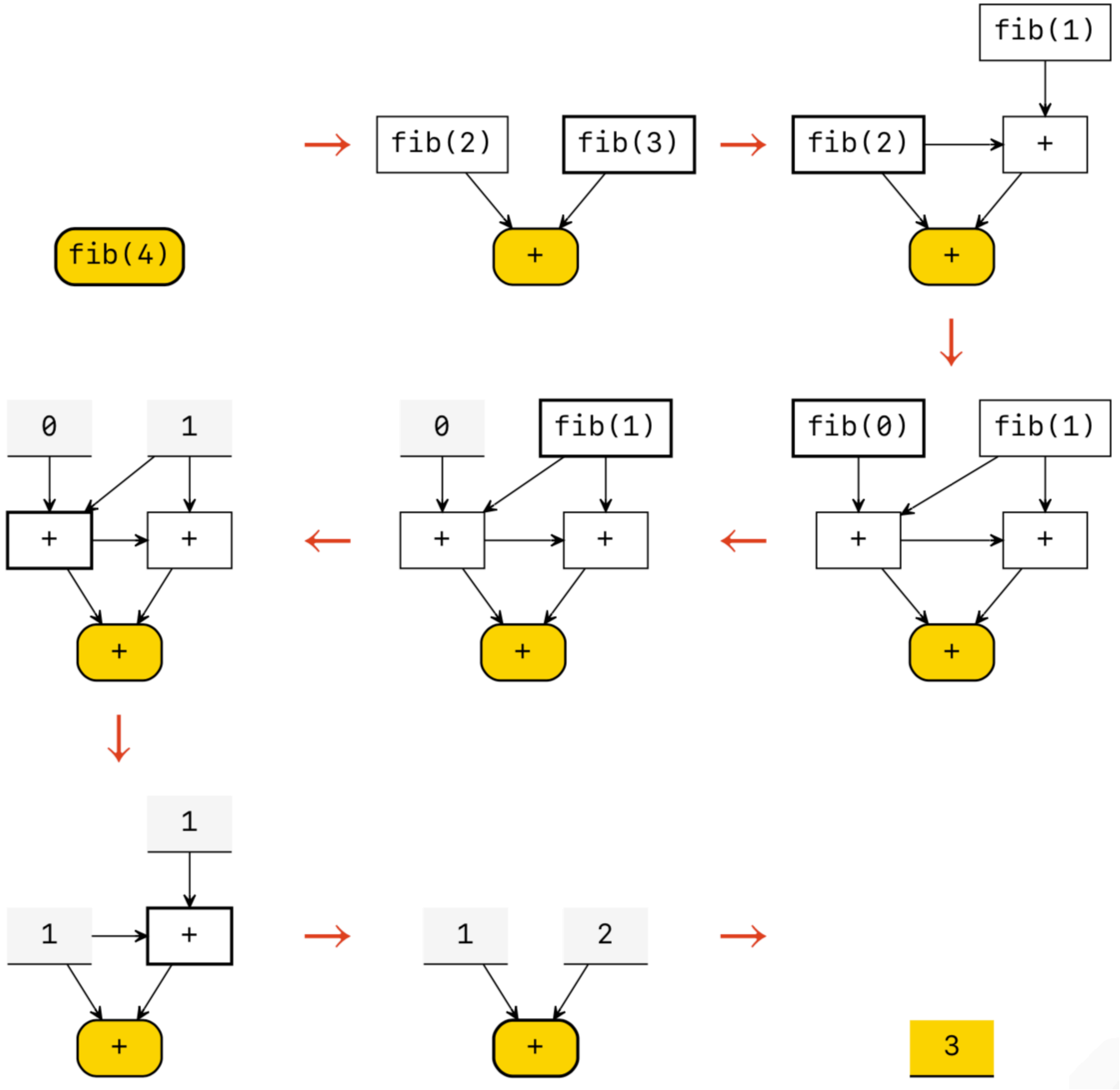
② Roundtrips to the cloud hurt performance

Representation of gg IR for compiling GNU Hello



② Roundtrips to the cloud hurt performance

gg IR can handle dynamic dependency graphs



Challenges of outsourcing applications to the cloud

① Software dependencies must be managed
→ lightweight containers (thunks)

② Roundtrips to the cloud hurt performance
→ linked containers (gg IR)

③ **Cloud functions are promising, but hard to use well**

Challenge ③

Cloud functions are promising, but hard to use well

- **The dream:** renting a supercomputer by the second.
- Warm clusters are expensive, cold clusters are slow to start

10,000 workers for 10 seconds on AWS Lambda costs ~$5

- ExCamera, PyWren, Sprocket, Serverless MapReduce, Spark-on-Lambda
- Using cloud functions is challenging!

③ Cloud functions are promising, but hard to use well

Faster startup

- 1,000 workers running `sleep(2)` on AWS Lambda:

PyWren	46s
Spark-on-Lambda	54s
Google Kubernetes Engine	03m 08s
gg on AWS Lambda	06s

Getting data to the cloud

- Uploading 1,000 files each sized 50 KB to S3:

awscli	11s	
gg	0.4s	*28x faster*

③ Cloud functions are promising, but hard to use well

Many applications require inter-function communication

- Was commonly believed that direct communication is forbidden by design.

  e.g., "Two Lambda functions can only communicate through an autoscaling intermediary service; […] a storage system like S3."
  ~Serverless Computing: One Step Forward, Two Steps Back

- Current systems use indirect techniques such as using shared storage, e.g., S3.

③ Cloud functions are promising, but hard to use well

Many applications require inter-function communication

- Using off-the-shelf **NAT-traversal** techniques, the Lambdas can talk to each other at speeds up to **600 Mbps**.

Challenges of outsourcing applications to the cloud

① Software dependencies must be managed
→ lightweight containers (thunks)

② Roundtrips to the cloud hurt performance
→ linked containers (gg IR)

③ Cloud functions are promising, but hard to use well
→ grind, grind, grind! 🔥

Applications: **Software Compilation**

- Build systems are often large and complicated; very difficult to manually rewrite them in `gg` IR.
- We need a system that works with existing build systems, like `make`, CMake, Ninja, etc.

Model substitution:
A technique to extract gg IR from existing applications

- **Idea:** run the original build system, but replace every stage with a ‘model’ program that produces a thunk, instead of the actual output.

Showtime!

Let’s see gg in action.

Applications: **Software Compilation**

- **gg on AWS Lambda** is 2–5× faster than **`icecc`** outsourcing to a 384-core cluster.

Compiling Inkscape		
icecc to a warm cluster of 48 cores	~7 min	$2.3/hr
icecc to a warm cluster of 384 cores	~7 min	$18.40/hr
gg to AWS Lambda	~1.5 min	50¢/run

Compiling **Google Chrome** takes ~~16 hours~~ **18 minutes.**

Running unit tests for **LibVPX** takes ~~1.5 hours~~ **4 minutes.**

Encoding a **15-minute 4K video** takes ~~7.5 hours~~ **2.5 minutes.**

Rendering **a single frame** of **Monsters University** takes **(?)**

Takeaways

- **gg is a framework and a toolkit** that makes it practical to outsource **everyday applications** using thousands of parallel threads on cloud-functions services.

- We ported several latency-sensitive applications to run on `gg`: **software compilation**, **unit testing**, **video encoding**, and **object recognition**.

- For example, `gg` can speed up compilation by 2–5× compared to a conventional tool (`icecc`), without requiring a warm cluster.

- gg is open-source software: https://snr.stanford.edu/gg

Thank you: NSF, DARPA, Google, Dropbox, VMware, Huawei, Facebook, Stanford Platform Lab, and lambda calculus.