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
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
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.
An example thunk: Preprocessing a C source file.

```json
{
  function: {
    hash: 'VDSo_TM',
    args: ['gcc', '-E', 'hello.c', '-o', 'hello.i'],
    envvars: ['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.
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.
An example thunk: Preprocessing a C source file.
② Roundtrips to the cloud hurt performance

Containers can reference each other’s outputs

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

content hash: T0MEiRL

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

content hash: TRFS91
Roundtrips to the cloud hurt performance

**Representation of gg IR for compiling GNU Hello**

```
dirname.c  string.h  closeout.c  stdio.h  hello.c
  ↓     ↓     ↓     ↓     ↓
dirname.i closeout.i hello.i
  ↓     ↓     ↓
dirname.s closeout.s hello.s
  ↓     ↓
dirname.o closeout.o
  ↓
libc libhello.a hello.o
  ↓
hello
```

1. `hello.o` generated from `hello.s` and `string.o`.
2. `libhello.o` generated from `dirname.o`, `libhello.a`, and `closeout.o`.
3. `hello` linked to create `hello (stripped)`.
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
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:

<table>
<thead>
<tr>
<th>System</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>PyWren</td>
<td>46s</td>
</tr>
<tr>
<td>Spark-on-Lambda</td>
<td>54s</td>
</tr>
<tr>
<td>Google Kubernetes Engine</td>
<td>03m 08s</td>
</tr>
<tr>
<td>gg on AWS Lambda</td>
<td>06s</td>
</tr>
</tbody>
</table>
Cloud functions are promising, but hard to use well

**Getting data to the cloud**

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

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>awscli</td>
<td>11s</td>
<td></td>
</tr>
<tr>
<td>gg</td>
<td>0.4s</td>
<td>28x faster</td>
</tr>
</tbody>
</table>

27x faster
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.
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.

<table>
<thead>
<tr>
<th>Compiling Inkscape</th>
<th>Time</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>icecc to a warm cluster of 48 cores</td>
<td>~7 min</td>
<td>$2.3/hr</td>
</tr>
<tr>
<td>icecc to a warm cluster of 384 cores</td>
<td>~7 min</td>
<td>$18.40/hr</td>
</tr>
<tr>
<td>gg to AWS Lambda</td>
<td>~1.5 min</td>
<td>50¢/run</td>
</tr>
</tbody>
</table>
THE #1 PROGRAMMER EXCUSE FOR LEGITIMATELY SLACKING OFF:

"MY CODE'S COMPILING."

HEY! GET BACK TO WORK!

COMPILED!

USING GG!
Running unit tests for LibVPX takes 1.5 hours and 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](https://snr.stanford.edu/gg)

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