Hermes: um protótipo de ferramenta de entrega *serverless* com suporte para GPUs

Tiago Martins Nápoli[1], Renato Cordeiro Ferreira[1], e Prof. Dr. Alfredo Goldman[1]

[1] Instituto de Matemática e Estatística - Universidade de São Paulo

Motivação

Nos últimos anos, as GPUs têm ganhado cada vez mais importância na comunidade científica e na indústria. Apesar disso, o fluxo de desenvolvimento de aplicações aceleradas por GPUs mantém diversos inconvenientes – tanto o desenvolvimento local quanto o remoto trazem consigo atritos para compilar e executar código. A alternativa local exige, por exemplo, disponibilidade de uma GPU por programador, além de configuração de drivers e kits de desenvolvimento; a alternativa remota exige acesso via SSH e alterações no código com editores de linha de comando.

Arquitetura *Serverless*

A Arquitetura *Serverless* refere-se a um paradigma relativamente novo de entrega de software, com crescente importância e presença na comunidade de computação e indústria. Ela oferece:

a) *Scaling* automático e elástico – possibilidade de escalar a zero;

b) Custos reduzidos e adequados à demanda;

c) Gerenciamento de infraestrutura abstraído do desenvolvedor.

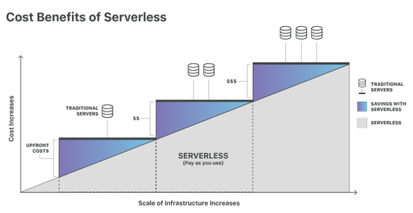

Figura 1: Imagem de *Cloudflare*[2].

Hermes

O protótipo criado, o Hermes, é uma aplicação cliente-servidor que, por meio do uso de uma CLI, oferece ao usuário:

a) Suporte para execução de código criado em C++ ou CUDA;

b) Entrega *serverless* de execuções;

c) Uso opcional de GPUs.

CLI

A CLI para utilização de uma instância Hermes foi disponibilizada por meio do pacote NPM ***@hermes-serverless/cli*** [1].

```
Usage: @hermes-serverless/cli <command> [options]

Commands:

  config [property] [newValue]     Change or print a config

  execution list                   List your executions
  execution inspect <runID>        Get info on an execution

  function build                                        Build a hermes-function locally
  function delete <functionName> <functionVersion>      Delete a function from remote
  function deploy                                       Deploy a function to remote
  function init                                         Init a hermes-function
  function list                                         List your hermes-functions registered on remote
  function run <functionID>                             Start a hermes-function execution

  remote login <username>          Login into a remote Hermes server
  remote logout                    Logout from an remote Hermes server
  remote register                  Register into a remote Hermes server
  remote unregister                Unregister from a remote Hermes server
  remote whoami                    Check the user you're logged on

Options:

  -h, --help  show help information
  -v, --version  show version number
```

Figura 2: Comandos da CLI.

Após a instalação e configuração da CLI, o usuário já poderá criar aplicações que poderão ser registradas e executadas na instância Hermes em uso. A documentação para configurar e instalar a CLI está disponível em [1].

Hermes-Functions

As aplicações a serem registradas e executadas em uma instância Hermes recebem o nome de *hermes-functions*. Uma *hermes-function* é formada por um arquivo *hermes.config.json*, com metadados da função, um *Makefile* e o código da aplicação.

Function	Language	GPU Capable	Watcher Image
char-printer:1.0.0	cpp	true	tiagonapoli/watcher-char-printer:1.0.0
gpu-pi-montecarlo:1.0.0	cuda	true	tiagonapoli/watcher-gpu-pi-montecarlo:1.0.0
pi-montecarlo:1.0.0	cpp	false	tiagonapoli/watcher-pi-montecarlo:1.0.0
pi-montecarlo:1.0.1	cpp	false	tiagonapoli/watcher-pi-montecarlo:1.0.1
reduce-sum:1.0.0	cpp	false	tiagonapoli/watcher-reduce-sum:1.0.0
waiter:1.0.0	cpp	false	tiagonapoli/watcher-waiter:1.0.0

Figura 3: Lista de *hermes-functions* registradas pelo usuário.

A qualquer momento o usuário pode, utilizando a CLI, registrar uma *hermes-function* na instância Hermes em utilização, o que possibilitará a criação de execuções dessa função.

Execução de *hermes-functions*

As *hermes-functions* registradas podem ser executadas com um simples comando da CLI. As execuções ocorrem no servidor, utilizando os recursos e o ambiente de execução dele, assim não haverá nenhuma necessidade de configuração por parte do usuário. As execuções criadas podem ser **síncronas** ou **assíncronas**.

As **execuções síncronas** mantém um canal de comunicação aberto com o servidor, permitindo que o usuário receba a saída da execução em tempo real.

```
tiagonapoli@tiagonapoli-Inspiron-5480:~$ hermes function run --sync tiago/pi-montecarlo:1.0.0
? Input: 100000000 2
Elapsed time: 1283.145825000000 ms
PI: 3.141224800000
```

Figura 4: Execução síncrona da *hermes-function tiago/pi-montecarlo:1.0.0*, que estima π usando o método de Montecarlo, com o número de iterações e *threads* especificados na entrada.

As **execuções assíncronas** fecham o canal de comunicação com o servidor e fornecem ao usuário um ID, que pode ser usado para checar o status da execução utilizando o comando de inspeção `hermes execution inspect`.

```
tiagonapoli@tiagonapoli-Inspiron-5480:~$ hermes function run --async tiago/waiter:1.0.0
? Input: 120 2
{ startTime: '2019-11-26T22:11:06.951Z', runID: '8' }
tiagonapoli@tiagonapoli-Inspiron-5480:~$ hermes execution inspect 8
{
  status: 'running',
  startTime: '2019-11-26T22:11:06.951Z',
  runningTime: '00:00:13.031',
  out: 'Hello 0\nHello 2\nHello 4\nHello 6\nHello 8\nHello 10\nHello 12\n',
  err: ''
}
```

Figura 5: Execução assíncrona da *hermes-function tiago/waiter:1.0.0*, que recebe como entrada dois números *t* e *i* e roda por *t* segundos, imprimindo uma mensagem a cada *i* segundos.

O histórico das execuções criadas pelo usuário é armazenado e pode ser acessado utilizando-se o comando `hermes execution list`. Além disso, as execuções já terminadas também podem ser inspecionadas, o que possibilita o resgate de suas saídas.

```
tiagonapoli@tiagonapoli-Inspiron-5480:~$ hermes execution list -p
```

RunID	Status	Start	End	Elapsed	Function
1	success	11/26 20:01:44.655	11/26 20:01:44.677	00:00:00.22	tiago/char-printer:1.0.0
2	success	11/26 20:03:08.991	11/26 20:03:09.024	00:00:00.33	tiago/pi-montecarlo:1.0.0
3	success	11/26 20:03:45.896	11/26 20:03:45.939	00:00:00.43	tiago/pi-montecarlo:1.0.0
4	success	11/26 20:05:14.981	11/26 20:05:15.139	00:00:00.158	tiago/pi-montecarlo:1.0.0
5	success	11/26 20:06:47.121	11/26 20:06:47.278	00:00:00.157	tiago/pi-montecarlo:1.0.0
6	success	11/26 20:07:36.678	11/26 20:08:36.707	00:01:00.29	tiago/waiter:1.0.0
7	success	11/26 20:10:32.051	11/26 20:11:32.090	00:01:00.39	tiago/waiter:1.0.0
9	success	11/26 20:12:54.176	11/26 20:12:55.487	00:00:01.311	tiago/pi-montecarlo:1.0.0
8	success	11/26 20:11:06.951	11/26 20:13:07.006	00:02:00.55	tiago/waiter:1.0.0
10	success	11/26 20:15:06.944	11/26 20:15:07.743	00:00:00.799	tiago/pi-montecarlo:1.0.0
11	success	11/26 20:15:36.200	11/26 20:15:36.969	00:00:00.769	tiago/pi-montecarlo:1.0.0

Figura 6: Lista de execuções criadas por um usuário.

Arquitetura

Uma instância Hermes é formada pelos seguintes serviços:

- ***function-orchestrator***: Responsável pelo recebimento de requisições feitas à instância Hermes e integração entre os serviços para respondê-las.
- ***function-registry***: Responsável pela persistência (*db*) e disponibilização (*api*) de dados.
- ***function-lifecycle-broker***: Responsável pela mensageria de eventos implementada no sistema.
- ***function-watcher***: Responsável por todas as tarefas relacionadas à execução dos binários de *hermes-functions*.

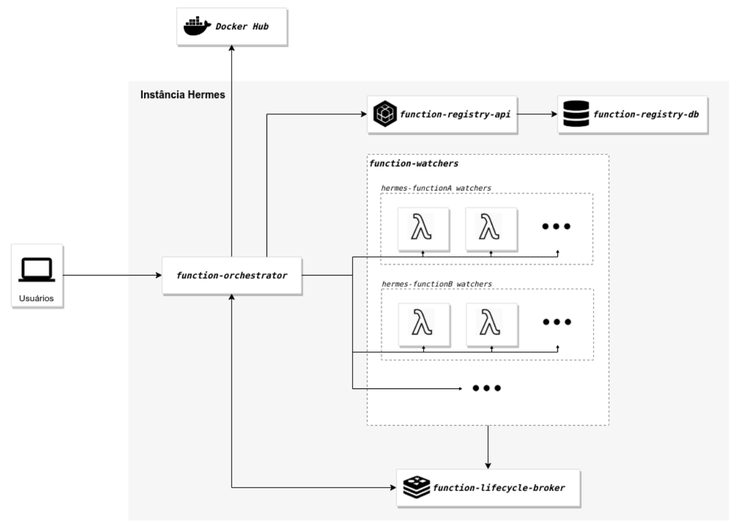

Diagrama 1: Arquitetura de uma instância Hermes.

Trabalhos Futuros

O projeto ainda possui diversos problemas a serem resolvidos e melhorias arquiteturais em aberto. Sendo o código *open-source*, as contribuições são muito bem-vindas – todo o projeto Hermes está disponível em [3]. Alguns desses desafios e melhorias são relativos à orquestração dos serviços, coleta e centralização de *logs*, usabilidade da CLI e, principalmente, aos problemas de segurança que surgem quando se executa código não-confiável em uma infraestrutura.

Fonte *imgflip.com/i/3hoakn*

Referências

[1] *@hermes-serverless/cli npm package*. URL: https://www.npmjs.com/package/@hermes-serverless/cli.

[2] *Cloudflare Learning - Serverless*. URL: https://www.cloudflare.com/learning/serverless/what-is-serverless/.

[3] *Hermes Serverless Project*. URL: https://github.com/hermes-serverless/.