# Parsl vs PyCOMPSs: Comparação de Usabilidade e Desempenho em *Workflows* Científicos de HPC

**Reiglan Soares**[1 2], Albert Emidio[1 2], Rafael Terra[1], Kary Ocaña[1],
Carla Osthoff[1], Hiago Rocha[1]

[1]Laboratório Nacional de Computação Científica (LNCC), RJ – Brasil
[2]Faculdade de Educação Tecnológica do Estado do Rio de Janeiro (FAETERJ-RJ), RJ – Brasil

X Escola Regional de Alto Desempenho ERAD-SE 2025

# Agenda

1. **Motivação**
2. **Objetivo**
3. **Metodologia**
4. **Resultados**
5. **Conclusão**

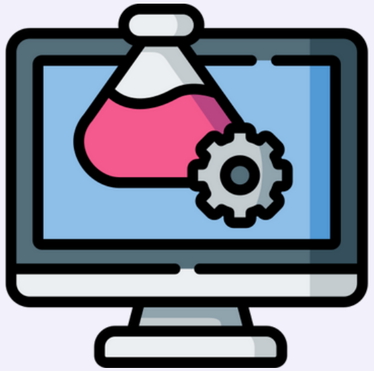
# Motivação

- O *workflow* científico é uma abstração que modela um experimento científico, pois estruturam cada etapa do processo de forma clara e replicável.

- Os sistemas de gerenciamento são ferramentas que controlam e automatizam a execução de *workflows* científicos.

- **Exemplos :**

Bioinformática



Química Computacional

Climatologia

# Motivação

- A motivação deste estudo surgiu pelo fato de **Parsl e PyCOMPSs serem amplamente utilizados no gerenciamento de *workflows* científicos.** Apesar disso, **a literatura carece de comparações diretas entre essas ferramentas quanto à facilidade de uso e desempenho.**

```python
from parsl import python_app, load
from parsl.configs.local_threads import config

load(config)

@python_app
def hello():
    return 'Hello World!'

print(hello().result())
```

Figura 1. Função simples em Parsl com o decorador @python_app.

```python
from pycompss.api.task import task
from pycompss.api.api import compss_wait_on

@task(returns=str)
def hello():
    return "Hello World!"

if __name__ == "__main__":
    result = hello()
    print(compss_wait_on(result))
```

Figura 2. Função simples em PyCOMPSs com decorador @task.

# Motivação

O **Parsl (*Parallel Scripting Library*)** é um modelo orientado a dados, em que as tarefas são executadas automaticamente conforme suas dependências.

- **Utilizando os decoradores:** *@python_app* e *@bash_app*
- **DataFlowKernel :** Responsável pelo controle e agendamento das tarefas.
- **Executores principais:** *ThreadPoolExecutor*, *HighThroughputExecutor* e *WorkQueueExecutor*.
- ***AppFuture*** **e** ***DataFuture*****:** Controle da execução da tarefa e disponibilidade dos arquivos de saída gerados pelas tarefas (dados produzidos).

# Motivação

O **PyCOMPSs** é uma interface Python que implementa o modelo de programação do COMPSs, focado em paralelização de tarefas para ambientes distribuídos.

- **Utilizando os decoradores:** *@task* e *@binary*
- **Runtime:** Controla a execução das tarefas e gerencia dinamicamente o grafo de dependências (DAG).
- **Módulos de execução:** runcompss e enqueue_compss.
- ***Task Descriptor* e *Data Object*:** Controlador responsável pela execução da tarefa e gerenciador dos arquivos de saída gerados pelas tarefas.

# Objetivo

Realizar uma análise comparativa entre os sistemas de gerenciamento **Parsl** e **PyCOMPSs**, avaliando:

- Usabilidade
- Performance

# Metodologia

- ***Workflow* Implementado**

Foi utilizado um *workflow* real de análise evolutiva, cuja principal característica é ser altamente paralelizável, e que é composto por 3 atividades principais:

1. **MAFFT:** Alinhamento múltiplo de sequências.
2. **RAxML:** Inferência de árvores filogenéticas.
3. **CodeML:** Estimação de pressões seletivas.

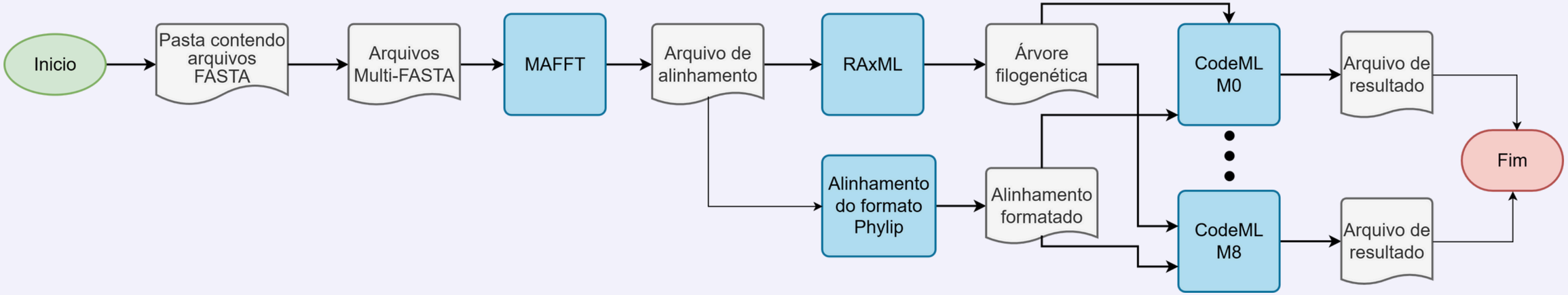



Figura 3. Adaptado de Terra et al., 2025.

# Metodologia

- **Dados de entrada:** Foram utilizados genomas de vírus da dengue dos sorotipos DENV-1 a DENV-4, obtidos no banco de dados biológicos BV-BRC.
  O conjunto total consistiu em 40 arquivos de dados reais no formato FASTA.

- **Comparação de desempenho:** Para a comparação de desempenho, foi utilizado o tempo total de execução como parâmetro de avaliação, variando o número de *threads* de 1 a 48 para analisar como o aumento da capacidade de processamento influencia na redução do tempo de execução de cada gerenciador.

- **Ambiente computacional:** 1 nó com 2× Intel Xeon Gold 6252 (48 núcleos) e 384 GB de RAM.

- **Ferramentas utilizadas:** Python 3.9.13, Parsl 2023.2.13, PyCOMPSs 3.3.3, MAFFT v7.453 ,RAxML v8.2.12, PAML CodeML) v4.10.7

# Resultados

## Considerações sobre as implementações

### Instalação

- O Parsl, totalmente em Python, possui instalação simples e imediata.
- O PyCOMPSs requer múltiplas dependências (Python, C++, Java), tornando o processo um pouco mais complexo.

### Desenvolvimento

- O Parsl gerencia automaticamente dependências e diretórios temporários.
- O PyCOMPSs exige anotações explícitas e sincronização via código , para realizar as mesmas configurações.

### Abstração do paralelismo

- Ambos abstraem o paralelismo e otimizam tarefas de forma bem eficientes.

## Consideração final

Em questão de implementação, o **Parsl** se destacou mais devido à praticidade de instalação e desenvolvimento mais simples de tarefas.

# Resultados

## Comparação de desempenho

- O Parsl apresentou uma ligeira vantagem de 6 segundos na média geral de todas as execuções, devido à sua inicialização mais rápida em comparação ao PyCOMPSs. Essa diferença, entretanto, é pequena e não representa perda nenhuma de desempenho.

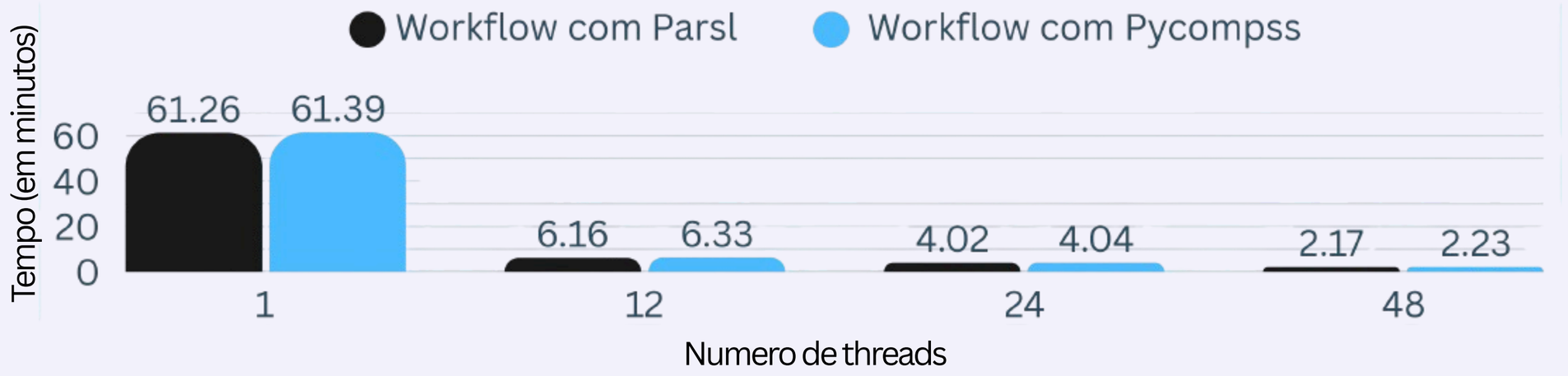



Figura 4. Comparativo de desempenho.

# Conclusão

- Parsl e PyCOMPSs apresentaram **desempenho semelhante**, com o Parsl sendo, em média, 6 segundos mais rápido nesse caso de estudo realizado e tambem se destacou em relação a usabilidade e facilidade de integração.

- Na visão do usuário que implementou e testou o *workflow*, o Parsl é mais indicado para este estudo específico, pois os dados utilizados não eram extremamente volumosos e a ferramenta demanda menos conhecimento técnico na implementação, facilitando o aprendizado.

## Trabalhos Futuros

- Análisar outros tipos de dados  principalmente com grande volume de dados,  para uma nova avaliação entre os sistemas de gerenciamnerto.

- Aprimorar a metodologia de avaliação, observando outros pontos e métricas para uma avaliação e perfilagem mais ampla  entre as duas ferramentas.

# Referências

**[1]** H. A. Saeed, S. T. F. Al-Janabi, E. T. Yassen e O. A. Aldhaibani, “Survey on Secure Scientific Workflow Scheduling in Cloud Environments,” Future Internet, v. 17, n. 2, p. 51, 2025.

**[2]** F. Suter et al., “A terminology for scientific workflow systems,” Future Generation Computer Systems, v. 174, p. 107 974, 2026.

**[3]** R. Terra et al., “HighSPA: A Scalable and Reproducible Parsl Framework for Molecular Evolutionary Analyses on HPC Systems,” em Proceedings of the Latin American High Performance Computing Conference (CARLA), 2025.

**[4]**Y. Babuji et al., “Parsl: Pervasive Parallel Programming in Python,” em Proceedings of the 28th International Symposium on High-Performance Parallel and Distributed Computing (HPDC), 2019, pp. 25–36.

**[5]**Tejedor, E., Becerra, Y., & et al. (2017). PyCOMPSs: Parallel computational workflows in Python. International Journal of High Performance Computing Applications, 31(1), 66-77.

# Obrigado!

LNCC
Laboratório Nacional de Computação Científica

FAETERJ Petrópolis

MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÃO

GOVERNO FEDERAL
BRASIL
UNIÃO E RECONSTRUÇÃO

CNPq
Conselho Nacional de Desenvolvimento Científico e Tecnológico

SINAPAD

CENAPAD RJ
Centro Nacional de Processamento de Alto Desempenho

X Escola Regional de Alto Desempenho ERAD-SE 2025