

Parsl vs PyCOMPSS: Comparação de Usabilidade e Desempenho em Workflows Científicos de HPC

Reiglan Soares^{1,2}, Albert Emidio^{1,2}, Rafael Terra¹, Kary Ocaña¹,
Carla Osthoff¹, Hiago Rocha¹

¹Laboratório Nacional de Computação Científica (LNCC)

²Faculdade de Educação Tecnológica do Rio de Janeiro (FAETERJ)
Petrópolis – RJ, Brasil

{reiglan, albert, rafaelst, karyann, osthoff, mayk}@lncc.br

Abstract. This work presents a comparative study between the Scientific Workflow Management Systems (SGWfCs) Parsl and PyCOMPSS, assessing both the user perspective in the implementation of a bioinformatics scientific workflow and the performance aspects. The results obtained, together with the practical usage experience, indicate that although both systems exhibit similar performance – with Parsl being, on average, only 6 seconds faster – Parsl stands out in terms of usability and ease of integration for the end user.

Resumo. Este trabalho apresenta um estudo comparativo entre os Sistemas de Gerenciamento de Workflows Científicos (SGWfCs) Parsl e PyCOMPSS, avaliando tanto a perspectiva do usuário na implementação de um workflow científico de bioinformática quanto em termos de desempenho. Os resultados obtidos, aliados à experiência prática de uso, indicam que, embora ambos apresentem desempenho semelhante – com o Parsl sendo, em média, apenas 6 segundos mais rápido –, o Parsl se destaca em termos de usabilidade e facilidade de integração para o usuário final.

1. Introdução

Workflows científicos são fundamentais para a organização e execução de experimentos computacionais, pois estruturam cada etapa do processo analítico de forma clara e replicável, garantindo padronização, reproduzibilidade e eficiência. Quando suportados por Sistemas de Gerenciamento de Workflows Científicos (SGWfC), esses workflows não apenas automatizam as etapas envolvidas, mas também otimizam o uso dos recursos computacionais, reduzindo o tempo de processamento e minimizando falhas manuais [1].

No contexto de pesquisas científicas, como na bioinformática, em que a complexidade e o volume de dados exigem soluções computacionais capazes de executar experimentos de forma eficiente em ambientes de Computação de Alto Desempenho (HPC), os SGWfC tornam-se extremamente úteis. Ao fornecerem uma interface de programação mais acessível e abstraírem os detalhes do *hardware* subjacente, os SGWfC simplificam significativamente o trabalho dos pesquisadores, permitindo foco na lógica do experimento em vez da infraestrutura computacional.

Existem diversos SGWfC disponíveis, com foco em diferentes áreas da indústria e da pesquisa, como (e.g., bioinformática, modelagem computacional e análise de dados)

[2]. No contexto de HPC, os mais amplamente utilizados e com suporte contínuo de suas equipes de desenvolvimento são o *Parallel Scripting Library* (Parsl)¹, desenvolvido pela Universidade de Chicago em colaboração com o *Argonne National Laboratory*, e o *Python COMPSS Superscalar* (PyCOMPSS)², do *Barcelona Supercomputing Center* (BSC). Embora Parsl e PyCOMPSS sejam amplamente utilizados e bem documentados individualmente, a literatura ainda carece de estudos que realizem uma comparação direta entre essas ferramentas [1], [2], [3], especialmente no que diz respeito tanto à facilidade de uso quanto ao desempenho.

Com base no exposto, o presente trabalho apresenta uma análise comparativa entre Parsl e PyCOMPSS, considerando tanto a perspectiva do usuário na implementação de um *workflow* de bioinformática quanto o desempenho na execução do *workflow* em ambientes de HPC. Nossos resultados mostraram que, embora ambos apresentem desempenho semelhante, o Parsl se destacou pela maior facilidade de uso e instalação e pela maior simplicidade na modelagem do *workflow*.

2. Fundamentação Teórica

Embora existam diversos SGWfCs voltados a diferentes áreas [1], [2], [3], os mais representativos em HPC são o Parsl e o PyCOMPSS. Esta seção descreve suas principais características e funcionalidades.

Parsl. O Parsl é uma biblioteca Python de programação paralela que utiliza decoradores para executar funções e softwares externos, como *scripts* Python e Bash. Baseado em programação orientada a dados, uma tarefa só é executada quando todas as entradas estão disponíveis, permitindo gerenciamento dinâmico e uso eficiente de recursos em máquinas locais, clusters HPC ou nuvens. Suporta diversos mecanismos de execução (*executors*), como *ThreadPool*, HTEX, Slurm e Condor. Seu objetivo é simplificar a escrita de *workflows* científicos complexos, oferecendo abstração de paralelismo e monitoramento de tarefas e recursos via *DataFlowKernel*.

PyCOMPSS. O PyCOMPSS é um *framework* de programação paralela que facilita a execução de aplicações Python em ambientes distribuídos e de alto desempenho. Baseado na infraestrutura COMPSS, permite escrever *workflows* científicos como programas sequenciais, enquanto o paralelismo é inferido automaticamente pelo sistema de *runtime*. Utiliza um modelo orientado a tarefas, identificando dependências de dados entre funções anotadas com `@task` e organizando-as em um grafo de dependência para execução assíncrona e paralela.

3. Metodologia

Workflow implementado. Foi implementado um workflow de bioinformática para alinhamento múltiplo de sequências, ilustrado na Fig. 1. O workflow inicia com a leitura dos arquivos multi-FASTA, cada um processado independentemente em paralelo. Para cada arquivo, realiza-se o alinhamento múltiplo com MAFFT, cuja saída alimenta duas tarefas: (i) reconstrução da árvore filogenética pelo RAxML e (ii) conversão do alinhamento

¹Parsl Documentation, disponível em <https://parsl.readthedocs.io/en/latest/>, acesso em 26 set. 2025.

²PyCOMPSS Documentation, disponível em <https://pypi.org/project/pycompss/>, acesso em 26 set. 2025.

para PHYLIP. As saídas dessas tarefas são então usadas em seis execuções do CodeML, aplicando os modelos de substituição M0, M1, M2, M3, M7 e M8, com os resultados organizados em diretórios específicos para análise posterior.

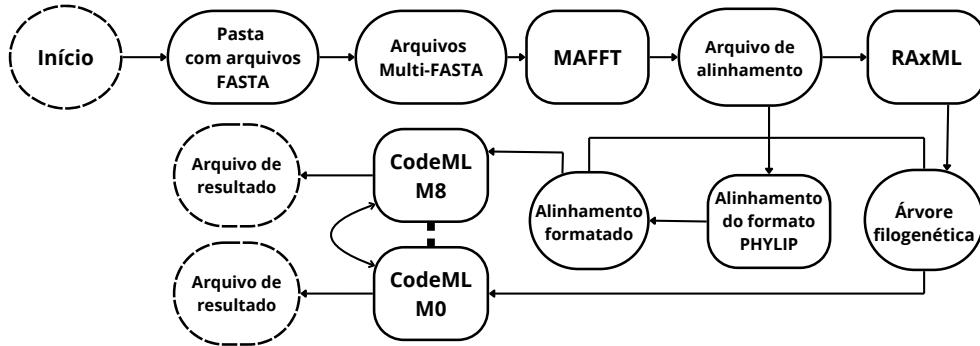


Figura 1. Workflow para alinhamento múltiplo de sequências

Ferramentas utilizadas. A implementação do *workflow* considerou os seguintes softwares, bibliotecas e ferramentas: Python 3.9.13, Parsl 2023.2.13, MAFFT v7.453, PAML v4.10.7 (CodeML), RAxML v 8.2.12 e PyCOMPSs 3.3.3.

Dados de entradas do workflow. Para os experimentos foram utilizados 40 arquivos no formato FASTA, correspondentes a genomas originais do vírus da dengue (DENV-1 a DENV-4) obtidos no banco BV-BRC. As sequências foram reduzidas em relação ao tamanho original e organizadas em dez genes: C, prM, E, NS1, NS2A, NS2B, NS3, NS4A, NS4B e NS5, constituindo o conjunto de entrada do *workflow*.

Ambiente computacional. Os experimentos foram realizados um nó com dois processadores Intel Xeon Cascade Lake Gold 6252 (24 núcleos cada, totalizando 48 núcleos) e 384 GB de RAM.

4. Resultados

Esta seção apresenta os resultados sob duas perspectivas: (*i*) a partir das observações relativas ao uso dos SGWfCs e (*ii*) considerando o desempenho durante a execução dos *workflows*.

Considerações sobre as implementações. Durante a implementação do *workflow* em ambos os SGWfCs, foram observados fatores que influenciam a curva de aprendizado, o desenvolvimento e, consequentemente, a usabilidade dos sistemas. Para isso, consideramos os seguintes aspectos³:

Instalação. O Parsl, sendo totalmente em Python, instala-se facilmente e está pronto para uso ao ser importado. Já o PyCOMPSs requer Python, C++, Java e pacotes adicionais, tornando sua instalação mais complexa e sujeita a falhas de integração.

Desenvolvimento. O Parsl se destaca por sua abordagem declarativa e gerenciamento automático de dependências. Ele identifica tarefas subsequentes a partir dos arquivos de saída, gerencia automaticamente *sandboxes* e diretórios temporários, e organiza

³É importante ressaltar que essas observações referem-se a um usuário implementando um *workflow* científico específico, logo, essas conclusões podem variar conforme o usuário ou o tipo de *workflow*.

a execução do pipeline com base nas dependências, bastando fornecer os caminhos de entrada e saída. No PyCOMPSs, as tarefas devem ser anotadas com entradas e saídas, deixando explícito seu consumo e produção. Entretanto, o *sandbox* é parcialmente transparente: o usuário precisa gerenciar diretórios e sincronização manualmente. Assim, embora ofereça maior controle, exige mais conhecimento na implementação.

Abstração do paralelismo. Ambos os SGWfCs abstraem o paralelismo e otimizam a execução de tarefas. O Parsl gerencia dependências e execução automaticamente, enquanto o PyCOMPSs oferece maior controle, exigindo que o usuário sincronize tarefas e gerencie diretórios manualmente.

Em resumo, o Parsl se destacou pela simplicidade e rapidez de instalação, sem depender de múltiplas ferramentas externas. Na modelagem de *workflows*, ele gerencia automaticamente dependências e *sandbox*, dispensando a configuração manual de sincronização e diretórios de arquivos.

Comparação de desempenho. A seguir, comparamos o desempenho dos SGWfCs durante a execução do *workflow* implementado. Para isso, utilizamos os 40 arquivos de entrada, variando o número de *threads* do nó entre 1, 12, 24 e 48. Para cada variação, foram realizadas 3 repetições. A Fig. 2 apresenta o tempo total de execução (em minutos, eixo y) dos dois SGWfCs em função do número de *threads* (eixo x). No geral, observa-se desempenho similar entre ambos, com diferença média de apenas 6 segundos nas quatro configurações avaliadas, conferindo uma ligeira vantagem ao Parsl.



Figura 2. Comparativo de desempenho

5. Conclusão

Este trabalho comparou Parsl e PyCOMPSs em termos de implementação de *workflows* de bioinformática e desempenho. Ambos apresentaram desempenho semelhante, com Parsl apenas 6 segundos mais rápido em média, destacando-se pela maior usabilidade e facilidade de integração. Como trabalhos futuros, pretende-se expandir a análise para outros domínios científicos.

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.