

Construção Otimizada de Grafos na biblioteca NetFog

Optimized Graph Construction in NetFog library

Tiago Fogolin Ragassi – ticofogolin@gmail.com
Rafael Gutierrez Castanha – rafaelcastanha@unimar.br

INTRODUÇÃO

Grafos de larga escala sustentam aplicações que crescem em volume e criticidade, de motores de recomendação a grafos de conhecimento usados por sistemas de IA, passando por análise de redes sociais e bioinformática. Nesses contextos, o tempo de construção do grafo compete com o de consulta e treinamento, tornando a etapa de ingestão um alvo relevante de otimização.

A NetFog é uma biblioteca de grafos para Python, escrita em Rust via PyO3, que permite a escolha da estrutura de dados mais adequada a cada caso de uso. Este trabalho foca na otimização da etapa de construção do grafo, buscando aproximar ou superar o tempo de bibliotecas consolidadas da área, como *igraph* e *Networkx*.

O objetivo geral é avaliar o quanto camadas de abstração, como rótulos textuais e extração genérica via Foreign Function Interface (FFI), custam em desempenho frente a essas bibliotecas. Os objetivos específicos são: (i) escrever o algoritmo da Compressed Sparse Row (CSR) em complexidade $O(V+E)$; (ii) avaliar os trade-offs de otimizar a fronteira de Foreign Function Interface (FFI) entre Python e Rust; e (iii) quantificar o custo de uma API baseada em rótulos textuais (labels) frente a uma de inteiros puros.

DESENVOLVIMENTO

A CSR foi escolhida para este trabalho por oferecer o melhor desempenho de travessia do motor, já que as arestas de cada vértice ficam ordenadas e contíguas na memória (Beamer; Asanović; Patterson, 2015), o que também explica seu custo de construção mais alto, pois exige conhecer a posição final de cada aresta antes de escrevê-la. Por isso, o recorte deste estudo é a construção estática, não incremental. O algoritmo foi escrito usando contagem de grau seguida por soma de prefixos de GVEL (Sahu; Kothapalli, 2024): o grau final de cada vértice soma arestas existentes e novas; a soma de prefixos define a posição final de cada aresta, permitindo alocação única e exata via cursor por vértice.

Na fronteira de FFI, a extração automática do PyO3 usa protocolo genérico com verificação de tipo por elemento. Na geração em lote, isso foi substituído por acessos por referência emprestada, sem cópia, ao custo de menos verificação e risco de comportamento inesperado.

O benchmark usou grafo direcionado de 2.000.000 de vértices e 30.000.000 de arestas, com 5 execuções por variante, isoladas em processos próprios para evitar estado compartilhado do alocador. Foram comparadas as ingestões da NetFog por rótulos e por inteiros puros, cada uma utilizando a PyO3 sem alterações e com a FFI modificada.

A ingestão por inteiros com a NetFog levou 4,06s em média, 3,07x mais rápida que a melhor biblioteca comparada, *igraph* (12,44s em média). A por rótulos levou 16,21s, 1,30x mais lenta que essa mesma referência. A diferença entre as variantes da NetFog quantifica em ~4,00x o custo da camada de rótulos. A FFI modificada superou o PyO3 puro em ambos os casos (1,70x rótulos; 1,35x ids), confirmando o custo mensurável da fronteira de FFI.

CONCLUSÃO

Os resultados atendem aos objetivos propostos: a CSR foi escrita em O(V+E), e a ingestão por id puro superou a melhor biblioteca comparada, enquanto a construção com rótulos foi mais lenta, como esperado. Em ambos os casos, a FFI modificada superou a extração automática do PyO3. Esse padrão sugere que, quanto mais próxima do dado bruto a interface, melhor o desempenho de construção: cada camada de abstração adicionada (rótulos textuais, verificação automática, extração genérica) tem custo mensurável. Como limitação, a redução de verificações que sustenta parte desse ganho expõe os caminhos otimizados a comportamento inesperado com entradas malformadas, risco a avaliar antes de uma versão final.

REFERÊNCIAS

BEAMER, S.; ASANOVIĆ, K.; PATTERSON, D. The GAP benchmark suite. arXiv, 2015. Disponível em: <https://arxiv.org/abs/1508.03619>. Acesso em: 11 setembro 2026.

SAHU, S.; KOTHAPALLI, K. GVEL: fast graph loading in edgelist and compressed sparse row (CSR) formats. In: EURO-PAR 2024: PARALLEL PROCESSING WORKSHOPS. Cham: Springer, 2024. Disponível em: https://doi.org/10.1007/978-3-031-90200-0_22.