ChaosStat: uma plataforma de alto desempenho para detecção de padrões e avaliação de aleatoriedade

Bernardo Maia Coelho, Rodrigo Silva de Almeida, Lucca Rodrigues Cunha, Pedro Calligaris Delbem, César Magno Leite de Oliveira Junior, Alexandre Delbem, Filippo Ghiglieno

***Resumo*— Neste trabalho, apresentamos o ChaosStat, uma plataforma de alto desempenho para detecção de padrões e avaliação de aleatoriedade em geradores de números aleatórios quânticos (QRNGs). Diante do crescimento do volume de dados e da onipresença de sistemas computacionais, integramos testes consagrados na literatura em um pipeline unificado e extensível, sem abrir mão da performance necessária para cargas massivas e cenários em tempo real. ChaosStat adota uma arquitetura modular, permitindo incorporar novos testes e selecionar os mecanismos de otimização adequados a cada hardware, com execução paralela em múltiplos núcleos e aceleração por GPU via CUDA.**

***Palavras-Chave*— aleatoriedade; pseudoaleatoriedade; testes estatísticos; alto desempenho; CUDA; paralelismo.**

***Abstract*— In this work, we present ChaosStat, a high-performance platform for pattern detection and randomness evaluation, focusing on quantum random number generators (QRNGs). In response to growing data volumes and the ubiquity of computational systems, we integrated well-established tests from the scientific literature into a unified, extensible pipeline without sacrificing the performance needed for massive workloads and real-time applications. ChaosStat adopts a modular architecture, enabling the incorporation of new tests and the selection of optimization mechanisms tailored to each hardware configuration, with parallel execution across multiple CPU cores and GPU acceleration using CUDA.**

***Keywords*— randomness; pseudorandomness; statistical tests; high performance; CUDA; parallelism.**

I. Introdução

À medida que sistemas computacionais se tornam alicerces onipresentes da sociedade, a segurança da infraestrutura contemporânea passa a ser indissociável da segurança digital. Um pilar central dessa proteção é a capacidade de geração de números e sequências sem padrões detectáveis [1, 3] — aleatórias ou pseudoaleatórias —, algo essencial não apenas na criptografia, mas também em várias outras áreas da computação, incluindo o aprendizado de máquina e simulações científicas. Dessa forma, a garantia dessa "aleatoriedade" tem se tornado um assunto de extrema relevância para a atualidade.

Bernardo Maia Coelho, ICMC, USP, SC-SP, e-mail: bernardomc@usp.br; Rodrigo Silva de Almeida, Instituto de Ciências Matemáticas e de Computação (ICMC), Universidade de São Paulo (USP), São Carlos(SC)-SP, e-mail: rodrigo_almeida_04@usp.br; Lucca Rodrigues Cunha, Instituto de Física, Universidade de Brasília, Brasília-DF, e-mail: lucca.cunha@aluno.unb.br; Pedro Calligaris Delbem, Instituto de Física de São Carlos, USP, SC-SP, e-mail: pedrodelbem@usp.br; César Magno Leite de Oliveira Junior, Escola Politécnica de São Paulo, USP, São Paulo-SP, e-mail: cesar.magno@usp.br; Alexandre Delbem, ICMC, USP, SC-SP, e-mail: acbd@icmc.usp.br; Filippo Ghiglieno, Departamento de Física, Laboratório de Óptica, Laser e Fotônica, Universidade Federal de São Carlos, SC-SP, e-mail: filippo.ghiglieno@df.ufscar.br.

Diferente dos geradores pseudoaleatórios (PRNGs), os geradores de números aleatórios quânticos (QRNGs) se apoiam em fenômenos quânticos genuinamente imprevisíveis [2, 3] na tentativa de gerar uma aleatoriedade mais pura. No entanto, embora essas abordagens ampliem a entropia de origem física, elas continuam sujeitas a interferências e vieses sutis que podem causar o colapso de estados puros e acarretar a formação de padrões detectáveis. Isso reforça a necessidade de baterias de testes estatísticos consolidadas para garantir a robustez desses geradores [1, 3]. Por essa razão, esse projeto foi desenvolvido para analisar a aleatoriedade de PRNGs - o que só é viável mediante processamento de grande volume de dados, necessário para obter alta confiança na inviolabilidade desses geradores.

Este trabalho apresenta o ChaosStat, uma plataforma de alto desempenho para avaliar a (pseudo)aleatoriedade a partir de uma bateria unificada de testes estatísticos consagrados na literatura [1]. O sistema foi concebido para processar fluxos e lotes em escala, com execução paralela em CPU e aceleração opcional por GPU, mantendo uma interface de uso simples para integração em pipelines e para uso exploratório. Este artigo oferece uma visão geral da plataforma — arquitetura, escopo funcional e formas de utilização — respeitando as restrições de confidencialidade inerentes aos segredos industriais.

II. Metodologia e Arquitetura do Sistema

O núcleo do ChaosStat é implementado em C/C++, visando controle fino de memória, vetorização e acesso a instruções de baixo nível quando apropriado, além de facilitar o uso de runtimes de paralelização na CPU (OpenMP) e aceleração em GPU (CUDA). A API é exposta nativamente para C/C++ — disponível em *linkagem* estática e dinâmica — e via bindings para Python, visando interoperabilidade com ecossistemas amplamente difundidos na prática. Ademais, a implementação deixa aberta a possibilidade de atender a outras plataformas e linguagens por meio da adição de novos bindings.

O sistema é distribuído primariamente como biblioteca, sobre a qual se apoiam uma CLI e uma interface gráfica leve para uso exploratório. A API de alto nível permite executar, com uma única chamada, uma bateria pré-configurada de testes amplamente utilizados (e.g., NIST SP 800-22) [1]; ao mesmo tempo, a API oferece controle fino sobre a seleção de testes, parâmetros e otimizações via seleção de parâmetros ajustável e disponibilidade de funções intermediárias. Quando o usuário não especifica parâmetros, o sistema adota políticas automáticas, sensíveis ao tamanho do dado, ao backend escolhido

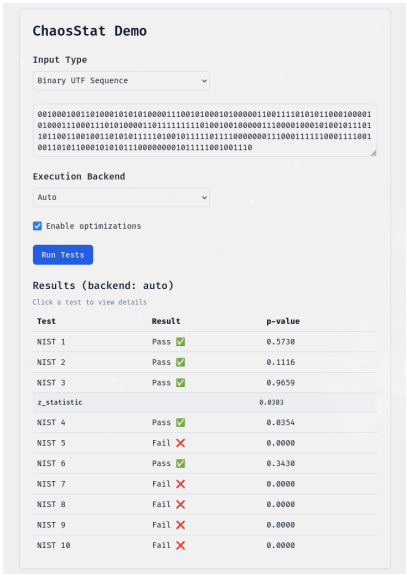

Fig. 1

PRIMEIRA VERSÃO DA APLICAÇÃO WEB GUI DO CHAOSSTAT. AO CLICAR EM UM RESULTADO DE UM TESTE, O USUÁRIO RECEBE MAIS DETALHES.

(CPU/GPU) e às especificidades do hardware e do caso de uso.

Cada teste possui implementações modulares para múltiplos backends. O usuário pode fixar o backend ou delegar a decisão ao sistema - tal como ocorre com os demais parâmetros -, o qual considera dimensões como o tamanho de entrada, o custo de transferências e os sobrecustos de sincronização. As otimizações podem ser ligadas/desligadas de forma granular, automática ou manualmente, favorecendo reprodutibilidade e auditoria.

A maioria dos testes de aleatoriedade segue o padrão: (i) extração de uma ou mais métricas do fluxo de bits; (ii) aplicação de testes estatísticos sobre essas métricas [1, 3]. O ChaosStat materializa métricas compartilháveis e as organiza em um grafo acíclico de dependências, permitindo reuso entre testes e escalonamento paralelo com sincronizações mínimas.

A versão avaliada implementa os dez primeiros testes da NIST SP 800-22 [1]. Os testes 1–5 contam com versões paralelas em CPU e versões aceleradas em GPU; os testes 6–10 contam, nesta etapa, com versões seriais otimizadas. Testamos seu funcionamento com sequências geradas pelas plataformas ID Quantique Quantis QRNG e IBM Quantum Experience [3].

III. RESULTADOS

Observou-se que implementações seriais otimizadas superam alternativas paralelas para sequências pequenas, dado o sobrecusto de threads e sincronizações. Em tamanhos médios a grandes, o backend CPU paralelo (multi-core com OpenMP) prevalece; o backend GPU mostra vantagem apenas em escalas suficientemente grandes, quando o paralelismo maciço compensa transferências e barreiras.

Nessa versão, todos os testes 1–10 da NIST SP 800-22 foram validados estatisticamente contra casos de teste conhecidos gerados sinteticamente ou advindos de QRNGs [3]; e foram comparados com implementações de referência [1].

TABELA I

SPEEDUP OBSERVADO NO TESTE NIST 05

Tamanho	backend		
	CPU Serial	CPU Paralela	GPU (CUDA)
8 kB	2.34×	3.76×	0.25×
128 kB	2.56×	4.24×	3.21×
2 mB	2.44×	4.01×	18.65×
32 mB	2.45×	4.06×	22.64×
1 gB	1.73×	2.86×	12.11×

Speedups de cada backend otimizado quando comparado com a implementação baseline. 1× implica que não houve melhora. Testes realizados em uma máquina Linux, Ubuntu 22.04, processador i5-11400H e GPU RTX 3050.

IV. CONCLUSÕES E TRABALHO FUTURO

Apresentamos o ChaosStat, uma plataforma modular e de alto desempenho para avaliação de (pseudo)aleatoriedade que integra testes consagrados em um pipeline unificado, com suporte a CPU e GPU. Os resultados indicam a importância de selecionar o backend em função do regime de tamanho: serial para entradas pequenas, CPU paralela para faixas intermediárias e GPU apenas quando o volume amortiza custos de transferência.

Como próximos passos, planejamos: (i) ampliar a aceleração dos testes 6–15 e integrar outras baterias de teste; (ii) incorporar streaming em tempo real com janelas deslizantes; (iii) enriquecer os diagnósticos na aplicação GUI, preservando a simplicidade da API.

AGRADECIMENTOS

Os autores agradecem ao Centro de Inteligência Artificial (C4AI-USP), pelo apoio da Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP, processo nº 2019/07665-4), da IBM Corporation, do Centro de Ciências Matemáticas Aplicadas à Indústria (CeMEAI, FAPESP, processo nº 2013/07375-0) e do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq).

REFERÊNCIAS

[1] A. Rukhin *et al.*, *NIST SP 800-22 Rev. 1a: A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*. National Institute of Standards and Technology, abr. 2010. Disponível em: https://doi.org/10.6028/NIST.SP.800-22r1a. Acesso em: 20 ago. 2025.

[2] M. Herrero-Collantes e J. C. Garcia-Escartin, "Quantum random number generators," *Rev. Mod. Phys.*, v. 89, art. 015004, 2017. DOI: 10.1103/RevModPhys.89.015004. Disponível em: https://link.aps.org/doi/10.1103/RevModPhys.89.015004. Acesso em: 20 ago. 2025.

[3] F. Ghiglieno, L. Roncaratti, L. R. Cunha, P. C. Delbem, R. S. de Almeida, A. M. Saraiva, e A. Delbem, "Geradores de números aleatórios: da pseudoaleatoriedade à verdadeira aleatoriedade na era da segunda revolução quântica," *Revista Brasileira de Ensino de Física*, v. 47, e20240469, 2025. DOI: 10.1590/1806-9126-RBEF-2024-0469. Disponível em: https://doi.org/10.1590/1806-9126-RBEF-2024-0469. Acesso em: 20 ago. 2025.