ORQUESTRAÇÃO AUTOMATIZADA DE INFRAESTRUTURAS VIRTUAIS: TEORIA E PRÁTICA

Claudinei Cabral Júnior, Guilherme Piêgas Koslovski

INTRODUÇÃO

Ambientes modernos de *data centers* de Computação de Alto Desempenho (HPC) executam *workflows* científicos compostos por tarefas interdependentes, tradicionalmente organizados como grafos acíclicos direcionados (DAGs). O escalonamento inicial dessas tarefas define sua alocação nos recursos disponíveis. Entretanto, essa decisão tende a permanecer fixa durante toda a execução, mesmo diante de mudanças na carga de trabalho, contenção de recursos e congestionamento de rede, o que compromete progressivamente o desempenho do *workflow*. Este trabalho parte da premissa de que a orquestração de *workflows* em HPC deve ser tratada como um processo contínuo de tomada de decisão, e não como uma decisão única tomada no momento da submissão.

Nesse contexto, propõe-se o DARTS (*Dragonfly-Aware Reinforcement Learning Task Scheduler*), um orquestrador autônomo baseado em Aprendizado por Reforço *Actor-Critic* (ACRL) (KOSLOVSKI; PEREIRA; ALBUQUERQUE, 2024) que integra escalonamento com consciência topologíca de rede com reconfiguração dinâmica de tarefas em tempo de execução, alinhado ao paradigma de gerenciamento *zero touch* (GALLEGO-MADRID et al., 2022; LIYANAGE et al., 2022). O objetivo geral consiste em investigar se a aprendizagem conjunta de políticas de escalonamento e reconfiguração, orientada pela topologia DragonFly, é capaz de reduzir o tempo de resposta dos *workflows* sem comprometer a eficiência de comunicação da infraestrutura.

DESENVOLVIMENTO

A metodologia foi validada com dados reais do supercomputador Marconi100 (CINECA), coletados pela infraestrutura ExaMon (BORGHESI et al., 2023), combinando métricas do gerenciador de filas SLURM com a topologia física DragonFly, organizada em três grupos (DGs) interconectados, totalizando 49 *racks* e 980 nós. A partir da *job table*, foram extraídos atributos essenciais de cada tarefa (tempo de submissão, tempo de execução declarado e número de nós requisitados) e, dada a ausência de dependências explícitas nos registros bem-sucedidos, uma cadeia de precedência foi reconstruída com base na ordenação temporal de tarefas pertencentes ao mesmo lote de submissão.
O agente ACRL adota uma arquitetura *dual-headed*, na qual um *encoder* de *racks* compartilhado alimenta duas políticas independentes, escalonamento e reconfiguração, cada uma associada a um crítico próprio, evitando que objetivos de recompensa incompatíveis degradem a estimativa de valor. A recompensa de escalonamento combina um bônus de localidade sensível à carga do grupo DragonFly de destino com uma penalidade proporcional ao congestionamento do *rack*, enquanto a recompensa de reconfiguração é definida pela variação de congestionamento entre *racks* de origem e destino, comprimida por transformação *symlog* para estabilizar o treinamento.
A decisão de reconfigurar segue um mecanismo probabilístico inspirado nos algoritmos de controle de congestionamento RED/ECN (FLOYD; JACOBSON, 1993), disparando avaliações de reconfiguração de forma gradual conforme a utilização dos *racks* se aproxima da saturação. Foram avaliadas três variantes de agressividade: ACRL1 (seletiva), ACRL2 (ampla e pouco frequente) e ACRL3 (agressiva). As variações foram comparadas a escalonadores de referência (FIFO, EASY *Backfilling* e HEFT) (TOPCUOGLU; HARIRI; WU, 2002) e a uma versão do agente sem reconfiguração. A avaliação experimental foi conduzida em simulador de eventos discretos sob dois regimes de execução, *early finish* e *full runtime*, utilizando cinco métricas: *bounded slowdown* (bsld),

overhead de comunicação, penalidade de congestionamento, dispersão topológica entre grupos e número de reconfigurações executadas.

RESULTADOS

No regime *early finish*, em que os *jobs* podem liberar recursos antes do prazo declarado, o escalonamento topologicamente consciente já supera os escalonadores heurísticos mesmo sem reconfiguração (bsld 3,32, contra 3,38 do HEFT e 5,61 do EASY). A ativação da reconfiguração traz ganhos marginais nesse regime: a variante conservadora ACRL1 obtém o melhor resultado (bsld 3,27), enquanto a variante agressiva ACRL3 degrada o desempenho (bsld 6,27), pois o elevado volume de reconfigurações (3.430) não é acompanhado de ganho proporcional.

O cenário se inverte no regime *full runtime*, em que cada *job* ocupa recursos até o limite declarado. Os algoritmos heurísticos e o agente sem reconfiguração convergem para um patamar semelhante (bsld entre 9,43 e 9,71), pois a única alavanca disponível é a escolha do *rack* no momento da submissão. Ao habilitar a reconfiguração, o desempenho melhora expressivamente: ACRL3 torna-se a melhor configuração (bsld 5,76), seguida de ACRL1 (5,99), representando redução de aproximadamente 40% em relação aos escalonadores de referência. Esse resultado evidencia que, sob esse regime, a reconfiguração constitui elemento central da metodologia, e não um refinamento opcional.

A análise de dispersão topológica confirma que os ganhos de desempenho não comprometem a eficiência de comunicação: em todos os cenários avaliados, a proporção de *jobs* distribuídos entre múltiplos grupos DragonFly manteve-se baixa (entre 0,4% e 0,9%). A Figura 1 ilustra a arquitetura *dual-headed* proposta, e a Tabela 1 resume o bsld médio por cenário nos dois regimes avaliados.

CONSIDERAÇÕES FINAIS

Os resultados obtidos corroboram a hipótese de que a orquestração de *workflows* em ambientes HPC pode ser tratada como um processo contínuo de decisão, em vez de uma alocação estática definida na submissão. O DARTS demonstrou que a aprendizagem conjunta de escalonamento topologicamente consciente e reconfiguração dinâmica reduz o tempo de resposta médio dos *workflows*, especialmente no regime *full runtime*, sem sacrificar a eficiência de comunicação da infraestrutura. Constatou-se ainda que a agressividade ideal da política de reconfiguração depende diretamente da janela de oportunidade disponível em cada regime de execução, não sendo uma propriedade fixa da variante empregada.

Como limitação, a proposta foi validada exclusivamente sobre a topologia DragonFly do Marconi100, o que restringe a generalização imediata dos resultados. Como trabalhos futuros, pretende-se avaliar a metodologia em outras topologias de *data center* e investigar políticas de reconfiguração adaptativas, capazes de ajustar automaticamente sua agressividade conforme o regime de execução observado.

Palavras-chave: aprendizado por reforço; HPC; reconfiguração de *workflows*; escalonamento; DragonFly; ACRL.

ANEXOS

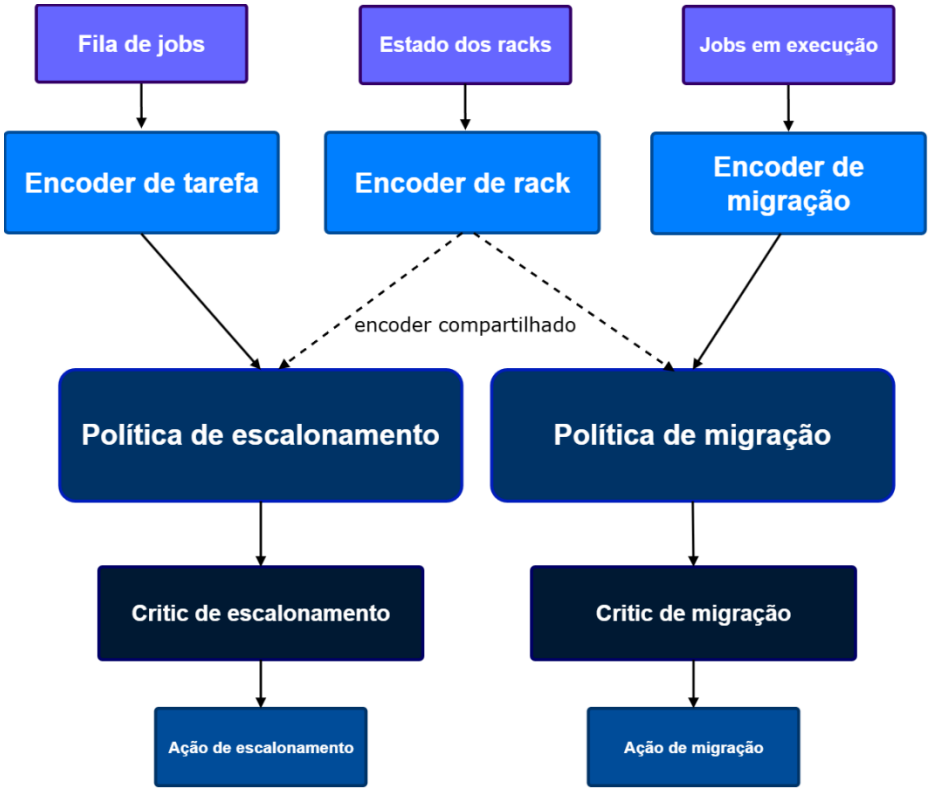


Figura 1. *Arquitetura dual-headed do agente ACRL do DARTS.*

Cenário	Early finish	Full runtime
FIFO	---	9,43
EASY Backfilling	5,61	---
HEFT	3,38	9,44
DARTS sem reconfiguração	3,32	9,71
DARTS (ACRL1)	3,27	5,99
DARTS (ACRL2)	3,28	9,52
DARTS (ACRL3)	6,27	5,76

Tabela 1. *BSLD médio por cenário nos dois regimes de execução avaliados*

REFERÊNCIAS BIBLIOGRÁFICAS

BORGHESI, Andrea et al. M100 ExaData: a data collection campaign on the CINECA's Marconi100 Tier-0 supercomputer. **Scientific Data**, v. 10, n. 1, p. 288, 2023.

FLOYD, Sally; JACOBSON, Van. Random early detection gateways for congestion avoidance. **IEEE/ACM Transactions on Networking**, v. 1, n. 4, p. 397-413, 1993.

GALLEGO-MADRID, Jorge et al. Machine learning-based zero-touch network and service management: a survey. **Digital Communications and Networks**, v. 8, n. 2, p. 105-123, 2022.

KOSLOVSKI, Guilherme Piêgas; PEREIRA, Kleiton; ALBUQUERQUE, Paulo Roberto. DAG-based workflows scheduling using Actor–Critic Deep Reinforcement Learning. **Future Generation Computer Systems**, v. 150, p. 354-363, 2024.

LIYANAGE, Madhusanka et al. A survey on Zero touch network and Service Management (ZSM) for 5G and beyond networks. **Journal of Network and Computer Applications**, v. 203, p. 103362, 2022.

TOPCUOGLU, Haluk; HARIRI, Salim; WU, Min-You. Performance-effective and low-complexity task scheduling for heterogeneous computing. **IEEE Transactions on Parallel and Distributed Systems**, v. 13, n. 3, p. 260-274, 2002.