Escalonando tarefas comunicantes em topologias DragonFly com Aprendizado por Reforço

Claudinei Cabral Junior[1], **Guilherme Piêgas Koslovski**[1]

[1]Departamento de Ciência da Computação
Universidade do Estado de Santa Catarina (UDESC) – Joinville, SC – Brasil

claudinei.cj@edu.udesc.br, guilherme.koslovski@udesc.br

Resumo. *O escalonamento eficiente de workflows em data centers de computação de alto desempenho exige decisões que considerem a topologia da rede de interconexão. Este trabalho apresenta um escalonador baseado em Aprendizado por Reforço com arquitetura Actor-Critic que incorpora consciência topológica da rede DragonFly. Resultados demonstram que a incorporação de informações topológicas permite ao agente aprender políticas que favorecem a localidade das tarefas, reduzindo o custo de comunicação.*

1. Introdução

Data Centers (DCs) de *High Performance Computing (HPC)* representam infraestruturas críticas para o avanço científico e tecnológico, projetadas para executar aplicações de larga escala que demandam capacidade de processamento elevada, largura de banda e armazenamento massivo. Contudo, o crescimento em escala impõe desafios ao gerenciamento eficiente de recursos. Entre eles, o escalonamento de tarefas comunicantes permanece como um problema central, uma vez que a qualidade das decisões de alocação é diretamente perceptível por usuários e administradores. Em particular, a topologia de interconexão exerce papel determinante no desempenho de *workflows* em HPC. A topologia *DragonFly*, adotada em supercomputadores modernos, organiza nós em grupos interconectados por enlaces locais e globais, permitindo comunicação com baixo diâmetro de rede. No entanto, os enlaces globais constituem recursos compartilhados de capacidade limitada: quando tarefas são alocadas em grupos distintos, o tráfego inter-grupo pode se tornar um gargalo significativo. Apesar dessa relevância, muitos escalonadores utilizam políticas simples, que não consideram a estrutura topológica nas decisões.

Nesse contexto, a introdução de técnicas de *Machine Learning (ML)* no gerenciamento de DCs tem avançado significativamente, tanto para refinamento de políticas existentes quanto para criação de novas abordagens direcionadas a cenários específicos. Entre elas, políticas baseadas em Reinforcement Learning (RL) têm sido aplicadas ao escalonamento adaptativo [Mao et al. 2019, Koslovski et al. 2024], porém normalmente modelam a infraestrutura como homogênea, sem incorporar a topologia de rede ao estado do agente. Para avançar nessa direção, este trabalho apresenta um escalonador baseado na arquitetura *Actor-Critic Reinforcement Learning (ACRL)* que incorpora consciência topológica da rede *DragonFly* às decisões de alocação. O cenário de avaliação utiliza dados reais de execução do supercomputador Marconi100 da CINECA.

2. Topologia DragonFly

A topologia DragonFly organiza roteadores em grupos interconectados que operam como unidades estruturais de maior alcance lógico. Com essa estrutura, pacotes ro-

teados atravessam no máximo um enlace global, reduzindo o custo da rede em 20% frente à *Flattened Butterfly* e 52% frente à *folded-Clos* em configurações acima de 16 mil nós [Kim et al. 2008]. A arquitetura combina enlaces locais intra-grupo e enlaces globais inter-grupo, garantindo comunicação com baixo número de saltos. Apesar do baixo diâmetro, os enlaces globais são recursos compartilhados de capacidade limitada, tornando-se gargalos sob tráfego intenso entre grupos [Bhatele et al. 2016].

Nesse contexto, a política de escalonamento torna-se fator crítico. Zhang et al. [Zhang et al. 2018] propuseram a política *Level-Spread*, que distribui tarefas dentro do menor nível hierárquico possível, reduzindo em média 16% a sobrecarga de comunicação. Kang et al. [Kang et al. 2024] combinaram roteamento baseado em aprendizado por reforço com posicionamento flexível de tarefas, demonstrando que a abordagem conjunta é mais eficaz do que otimizar roteamento e escalonamento isoladamente. Essa relação direta entre decisões e congestionamento nos enlaces globais motiva a proposta deste trabalho: integrar a consciência topológica da DragonFly ao escalonamento por meio de um agente ACRL, permitindo que as decisões de alocação considerem automaticamente o impacto sobre o congestionamento da rede.

3. Desenvolvimento

3.1. Cenário e *dataset*

O cenário é baseado no supercomputador Marconi100 da CINECA, cujos dados foram coletados pela infraestrutura ExaMon [Borghesi et al. 2023]. A topologia DragonFly possui 980 nós em 49 racks (20 nós/rack). Do dataset SLURM foram extraídos: `job_id`, `submit_time`, `start_time`, `end_time`, `nodes` e `array_job_id`. Este último permitiu reconstruir dependências entre tarefas de um mesmo grupo por ordenação temporal, formando cadeias de precedência que representam *workflows.*

3.2. Modelagem topológica e agente ACRL

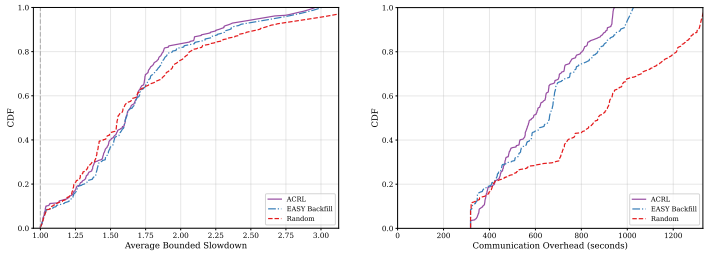
A topologia foi modelada em NetworkX reproduzindo a hierarquia DragonFly com conectividade completa intra-grupo e enlaces globais entre grupos, garantindo diâmetro três. O agente utiliza arquitetura *Actor-Critic* e RL com TensorFlow. O espaço de estados codifica, para cada *rack*: congestionamento interno, congestionamento externo (proporção de nós com comunicação *inter-rack*) e capacidade livre. As tarefas na fila são representadas por requisitos de nós, tempo de execução e tempo de espera. O *Actor* define a política sobre os *racks* viáveis, enquanto o *Critic* estima o valor do estado. A função de recompensa combina dois objetivos: minimizar o *bounded slowdown (bsld)* das tarefas completadas e maximizar a localidade topológica das alocações. Tarefas finalizadas com menor tempo de espera geram bônus de conclusão mais alto, enquanto alocações que concentram tarefas comunicantes em *racks* próximos recebem bônus de localidade proporcional à redução do custo de comunicação *inter-rack*. Esse duplo incentivo direciona o agente a buscar simultaneamente eficiência no escalonamento e proximidade topológica.

4. Simulações e resultados

Os *workflows* foram reconstruídos do *dataset* via composição temporal de dependências, e o treinamento foi realizado ao longo de múltiplos episódios correspondentes às janelas de submissão. A metodologia de avaliação adotou duas cargas de trabalho distintas. Para o

Tabela 1. Comparação entre as políticas de escalonamento.

Política	*Bounded slowdown*	*Sobrecarga*
ACRL	2,959	943,7
EASY Backfilling	3,005	1026,7
Random	3,405	1443,3



(a) *Bounded slowdown*.

(b) Sobrecarga de comunicação (segundos).

Figura 1. Distribuições acumuladas de bsld e sobrecarga

treinamento, foi introduzida uma carga com predominância de tarefas pesadas em número de nós requisitados, que demandam alocação em múltiplos *racks* simultaneamente. Essa escolha visa expor o agente a cenários nos quais a decisão de localidade é crítica: tarefas que ocupam poucos nós tendem a caber em um único *rack* independentemente da política, enquanto tarefas grandes forçam o agente a aprender a distribuir nós de forma a minimizar o tráfego *inter-rack*. Para a avaliação, foi utilizada uma carga de trabalho extraída do *dataset* do Marconi100, preservando suas características originais, a fim de verificar a capacidade de generalização da política aprendida para perfis de submissão distintos daqueles utilizados no treinamento.

Os resultados comparam três políticas: o agente ACRL treinado, a heurística *EASY Backfilling*, e uma alocação aleatória (*Random*) como linha de base. A Tabela 1 apresenta os valores médios obtidos. Ao analisar o bsld (Figura 1(a)), calculado com constante de limitação $\tau = 10$s, o ACRL apresenta uma progressão favorável ao longo de toda a distribuição. Aproximadamente 10% da distribuição mantêm bsld próximo a 1,0, correspondente à fase inicial do escalonamento. A curva apresenta crescimento acentuado entre bsld 1,3 e 1,7, atingindo 60% da distribuição em bsld 1,70. Aos bsld 2,0 a Cumulative Distribution Function (CDF) alcança 83%, e os 17% restantes se distribuem entre 2,0 e 2,96. Em contraste, na política *EASY Backfilling* observa-se aproximadamente 10% da distribuição concentrando-se em bsld próximo a 1,0, refletindo a fase inicial com ausência ou baixa formação de fila. A curva cresce de forma contínua entre bsld 1,3 e 1,8, alcançando cerca de 60% da distribuição em torno de bsld 1,75. Ao atingir bsld 2,0, a CDF aproxima-se de 80%, enquanto os 20% restantes distribuem-se entre 2,0 e aproximadamente 3,0, durante condições de maior saturação do sistema. A Figura 1(b) apresenta a CDF da sobrecarga de comunicação. O ACRL alcançou sobrecarga média de 943,7s contra 1026,7s do *EASY Backfilling*, uma redução de aproximadamente 8%. A diferença é mais expressiva na cauda da distribuição, na qual tarefas com comunicação intensiva se beneficiam mais da localidade topológica.

Os resultados demonstram que a consciência topológica melhora ambas as

métricas simultaneamente. Ao concentrar tarefas comunicantes em *racks* do mesmo grupo DragonFly, o ACRL reduz a sobrecarga de comunicação das tarefas, que por sua vez diminui o tempo efetivo de execução. Tarefas que terminam mais rápido liberam recursos mais cedo, beneficiando as subsequentes na fila e reduzindo o bsld global. Esse efeito cascata explica por que a melhoria no custo de comunicação de 8% se traduz também em ganho no bsld $1,5\%$, mesmo sem otimizar diretamente o tempo de espera.

5. Conclusão

Este trabalho apresentou um escalonador ACRL com consciência topológica DragonFly. A avaliação com dados reais do Marconi100 demonstrou resultados promissores, evidenciando que a consciência topológica beneficia simultaneamente a qualidade do escalonamento e a eficiência da rede. Como trabalhos futuros, propõe-se a incorporação de mecanismos de migração dinâmica de tarefas na topologia, permitindo reconfiguração em tempo de execução para mitigar congestionamentos e reduzir interferência *inter-rack*.

Agradecimentos: Este trabalho recebeu apoio do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), Fundação de Amparo à Pesquisa e Inovação (FAPESC), desenvolvido no LabP2D.

Referências

Bhatele, A., Jain, N., Livnat, Y., Pascucci, V., and Bremer, P.-T. (2016). Analyzing inter-job contention in dragonfly networks. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, pages 296–307. IEEE.

Borghesi, A., Di Santi, C., Molan, M., Ardebili, M. S., Mauri, A., Guarrasi, M., Galetti, D., Cestari, M., Barchi, F., Benini, L., Beneventi, F., and Bartolini, A. (2023). M100 exadata: a data collection campaign on the cineca's marconi100 tier-0 supercomputer. *Scientific Data*, 10(1):288.

Kang, Y., Wang, X., McGlohon, N., Mubarak, M., Chunduri, S., and Lan, Z. (2024). Preventing workload interference with intelligent routing and flexible job placement strategy on dragonfly system. *ACM Transactions on Modeling and Computer Simulation*, 34(2):1–26.

Kim, J., Dally, W. J., Scott, S., and Abts, D. (2008). Technology-driven, highly-scalable dragonfly topology. In *Proceedings of the 35th Annual International Symposium on Computer Architecture (ISCA)*, pages 77–88. IEEE.

Koslovski, G. P., Pereira, K., and Albuquerque, P. R. (2024). Dag-based workflows scheduling using actor–critic deep reinforcement learning. *Future Generation Computer Systems*, 150:354–363.

Mao, H., Schwarzkopf, M., Venkatakrishnan, S. B., Meng, Z., and Alizadeh, M. (2019). Learning scheduling algorithms for data processing clusters. In *Proceedings of the ACM Special Interest Group on Data Communication*, SIGCOMM '19, page 270–288, New York, NY, USA. Association for Computing Machinery.

Zhang, Y., Furlani, T., Jones, M. D., White, J. P., and DeLeon, R. L. (2018). A new job allocation policy for dragonfly networks. In *Proceedings of the IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 254–263. IEEE.