Deep Reinforcement Learning no Problema de Escalonamento de *Jobs* em Computação em Grids

Lucas Casagrande, Maurício Pillon

[1]Programa de Pós-Graduação em Computação Aplicada (PPGCA)
Universidade do Estado de Santa Catarina – UDESC – Joinville – SC – Brasil

lucas.casagrande@edu.udesc.br, mauricio.pillon@udesc.br

Resumo. *Algoritmos de escalonamento desempenham um papel chave na otimização dos recursos de uma infraestrutura em grids. No presente trabalho, três métodos de Deep Reinforcement Learning são treinados e comparados com a heurística Easy Backfilling via simulação. É possível concluir que métodos de DRL são capazes de aprender políticas de escalonamento que se adaptam a carga de trabalho, atingindo uma redução significativa no slowdown.*

1. Introdução

Através do compartilhamento de recursos entre organizações geograficamente distribuídas, computação em grids consegue prover uma massiva quantidade de poder computacional sob demanda. Neste contexto, políticas de escalonamento buscam otimizar o processo de mapeamento entre as demandas de cada usuário e os recursos disponíveis na grid de modo eficiente e justo. Este processo de otimização configura um problema da classe NP-Difícil sendo, de um modo geral, resolvido por métodos heurísticos [Poquet 2017].

Métodos heurísticos funcionam bem no espaço de solução em que foram projetados. Contudo, em um ambiente em grids, requisições chegam em rajadas, indicando que o sistema passa de períodos ociosos para períodos com sobrecarga de jobs na fila [Di et al. 2012]. Tal comportamento dinâmico impacta negativamente o desempenho do escalonador e adequar uma heurística geral que se adapte em tais condições é um processo complexo. Neste contexto, algoritmos de escalonamento adaptativos podem ser uma possível alternativa.

Métodos de *Reinforcement Learning* (RL) apresentam uma abordagem computacional, formulada como um *Markov Decision Process* (MDP), onde um agente consegue aprender uma política através da interação com um ambiente. Contudo, métodos comuns, como Q-learning e SARSA, possuem limitações e se tornam impraticáveis em problemas com muitas dimensões [Orhean et al. 2018]. Em contrapartida, métodos de *Deep Reinforcement Learning* (DRL) atacam este problema com a utilização de *Deep Learning* (DL) durante a aproximação das recompensas e estimação de uma política [Mnih et al. 2015].

Deste modo, o presente trabalho realiza uma análise do comportamento de três métodos de DRL (PPO, A2C e ACER) no problema de escalonamento de *jobs* em computação em *grids*. Os agentes são treinados em um ambiente simulado e comparados com a heurística *Easy Backfilling* (EASY), amplamente utilizada em ambientes em produção, através do simulador Batsim. Durante as próximas seções, são discutidos o modelo do sistema, contendo a formulação do problema e da plataforma, e os resultados obtidos com a experimentação. Na última seção encontram-se as considerações finais.

2. Modelo do Sistema

No decorrer desta seção, são apresentados os aspectos do modelo, levado em consideração durante a experimentação, assim como a formulação do problema para aplicação de métodos baseados em RL.

2.1. Plataforma

Uma plataforma em *grid* é composta por recursos computacionais organizados em locais. Cada local contém um ou mais *clusters* com um número arbitrário de máquinas. Cada máquina pode conter um ou mais processadores, cada qual com um número variável de núcleos. No presente trabalho, a comunicação entre os *clusters* e locais é desconsiderada e cada *job* pode requisitar um número arbitrário de núcleos, os quais daqui em diante serão chamados apenas de recursos.

Jobs chegam de modo *online* e são caracterizados de acordo com seu tempo de chegada, a quantidade de recursos requisitada e uma estimação do tempo de reserva dos recursos. Ambas as quantidade de recursos e de tempo de reserva são definidas pelo usuário e desconhecidas até o momento de sua submissão. Por simplicidade, consideramos que o tempo de execução dos *jobs* é conhecido e coincide com o tempo de reserva definido pelo usuário. Também é considerado que os recursos são homogêneos, com a mesma capacidade computacional, e preemptividade não é permitida. Portanto os recursos são liberados somente quando o *job* termina a sua execução.

2.2. Formulação do Problema

O problema é formulado como sendo um MDP de horizonte finito formado pela tupla $\{S, A, T, R\}$, onde S é o conjunto de estados, A determina o conjunto de ações, T determina a dinâmica de transição entre os estados, considerada como sendo desconhecida, e R é a função de recompensa que representa o custo da ação tomada:

Conjunto de Estados: representa o estado dos recursos da grid e da fila de *jobs* em espera. Cada estado é definido como uma imagem representando um diagrama de Gantt, onde: a altura representa unidades de tempo; e a largura representa a quantidade de recursos na grid e os requisitados por cada *job* na fila. *Jobs* que não podem ser totalmente representados no estado, devido a limitação de tamanho, são encapsulados em um contador que indica a quantidade total de *jobs* em espera.

Conjunto de Ações: representa a escolha de um *job* na fila para ocupar os recursos da grid. É definida pela quantidade de espaços de *jobs* que estão representados no estado. Em cada situação, o agente pode tanto escolher uma ação vazia $\emptyset$, que avança o tempo em uma unidade, ou selecionar um *job* para ser escalonado nos primeiros recursos disponíveis.

Função de Recompensa: O objetivo do agente é aprender uma política que maximize a sua recompensa esperada através da sucessiva tomada de decisões. Neste trabalho, é utilizado a minimização do *slowdown* como objetivo, sendo a função de recompensa formulada como a média negativa do *slowdown*. Está função pode ser interpretada como sendo uma penalização proporcional ao *slowdown*.

3. Experimentação

3.1. Configuração

Na realização dos experimentos, a plataforma utilizada no Batsim é formada por 10 recursos homogêneos. A carga de trabalho é composta por *jobs* paralelos, cujo seu perfil de execução é dividido igualitariamente entre os recursos requisitados. Para limitar o tamanho da imagem, somente os 10 primeiros *jobs* da fila são apresentados. Na medida que o escalonador consome estes *jobs*, os demais são retirados da fila de acordo com sua ordem de chegada.

A carga de trabalho é sintética e composta por 80% de *jobs* pequenos. Os tempos de execução dos *jobs* pequenos são gerados seguindo uma distribuição uniforme que varia entre $[1, 3]$ unidades de tempo enquanto demais *jobs* variam entre $[10, 15]$. A quantidade de recursos de cada *job* é determinada seguindo uma distribuição uniforme dentro do intervalo definido em dois grupos, $[1, 2]$ e $[5, 10]$, que são escolhidos aleatoriamente. O intervalo de chegada entre os *jobs* é definido seguindo um processo de Bernoulli de modo que a carga nos recursos varie entre 10% e 190% referente a sua capacidade em um período de tempo fixo.

Todos os métodos de DRL utilizam a mesma configuração de parâmetros quando aplicável. Tanto a política como a função valor dos métodos são definidas como sendo uma *Feedforward Neural Network* (FNN) contendo uma camada oculta com 20 unidades e função de ativação ReLU. Os parâmetros entre as redes não são compartilhados e um coeficiente de entropia de 0.01 é utilizado para incentivar a exploração. A taxa de aprendizado (α) é mantida constante em 0.001 enquanto a quantidade de *actors* é definida em 12. Não é aplicado desconto (γ) nas recompensas e a mesma quantidade de interações no treinamento é utilizada em cada um dos métodos. A quantidade limite de interações varia entre [.5M, 9M] proporcionalmente ao tamanho da carga de trabalho utilizada.

3.2. Resultados Obtidos

Na Figura 1 é demonstrado o desempenho dos métodos de DRL avaliados em comparação ao EASY em todas as cargas de trabalho utilizadas. É possível perceber que os métodos de DRL conseguem aprender uma política de escalonamento que se comporta igual ou melhor na maioria dos cenários. Comparando o EASY com o PPO, houve uma redução de 71,51% no *slowdown*, levando em consideração uma carga de 190% nos recursos. Já em relação ao A2C, houve uma redução de 54,86% enquanto com o ACER foi obtido uma redução de 58,82% nesta mesma carga de trabalho.

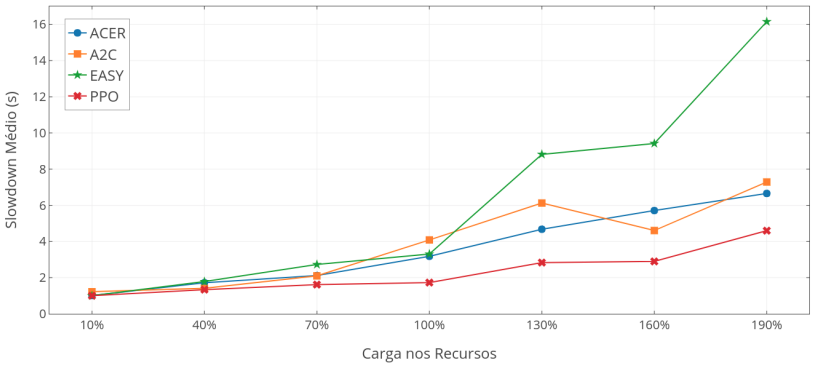

Figura 1. Desempenho dos algoritmos sob diferentes cargas de trabalho

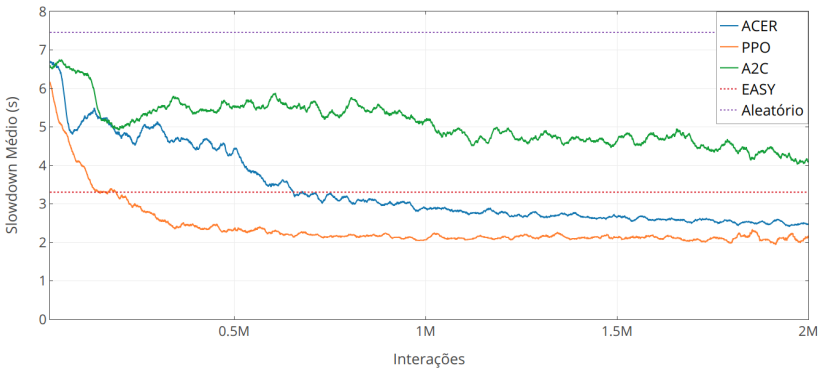

Figura 2. Curva de aprendizado dos métodos utilizados

Na Figura 2 é possível visualizar a curva de aprendizado dos métodos com uma carga de 100% nos recursos. No início da interação, os métodos possuem comportamento semelhante a uma política aleatória. Contudo, na medida com que a quantidade de interações aumenta, a política é otimizada até que o algoritmo converge. Com exceção do A2C, ambos os métodos PPO e ACER convergiram, respectivamente, entre 1 e 2 milhões de interações. Não obstante, o PPO foi capaz de ultrapassar o desempenho do EASY nas primeiras 250 mil interações com o ambiente.

4. Considerações Finais

No presente trabalho, três métodos de DRL são avaliados e comparados com a heurística EASY no problema de escalonamento de *jobs* em computação em *grids*. Através da interação com um ambiente, os métodos foram capazes de aprender uma política de escalonamento que atinge um melhor compromisso entre o momento de escalonar um *job* e o momento de forçar a sua espera na maioria dos cenários. Como trabalhos futuros, pretende-se explorar diferentes formulações de MDP para o problema em questão, abordando novas representações de estados e funções de recompensa.

Agradecimentos

O presente trabalho foi conduzido no laboratório LabP2D com apoio da FAPESC, UDESC e da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

Referências

Di, S., Kondo, D., and Cirne, W. (2012). Characterization and comparison of cloud versus grid workloads. In *2012 IEEE International Conference on Cluster Computing*, pages 230–238. IEEE.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529.

Orhean, A. I., Pop, F., and Raicu, I. (2018). New scheduling approach using reinforcement learning for heterogeneous distributed systems. *Journal of Parallel and Distributed Computing*, 117:292 – 302.

Poquet, M. (2017). *Simulation approach for resource management*. Theses, Université Grenoble Alpes.