Executando na Nuvem um Detector de Padrões de Acesso de E/S *

Cristiano A. Künas[1], Edson L. Padoin[2], Jean L. Bez[3], Alexandre S. Carissimi[1], Philippe O. A. Navaux[1]

[1]Instituto de Informática – UFRGS, [2]DCEEng – UNIJUI, [3]LBNL

{cakunas, asc, navaux}@inf.ufrgs.br, padoin@unijui.edu.br, jlbez@lbl.gov

Resumo. *As operações de E/S são um gargalo para várias aplicações, portanto, otimizar o desempenho dessas operações é de suma importância. Detectar corretamente os padrões de acesso, em tempo de execução, torna-se essencial para sistemas que buscam autoajustar seus parâmetros. Este artigo aborda uma técnica de aprendizado de máquina para detectar padrões de acesso de E/S e propõe transferir a carga de trabalho de treinamento local para a nuvem usando um acelerador TPU.*

1. Introdução e motivação

No contexto de HPC, os aplicativos emitem sua operação de entrada e saída (E/S) para um Sistema de Arquivos Paralelos (PFS) compartilhado, onde máquinas dedicadas atuam como servidores de dados [Ross et al. 2000]. Essas operações de E/S geralmente são consideradas um gargalo para um conjunto crescente de aplicativos. Além disso, vários fatores podem afetar seu desempenho, como a implantação e configuração da infraestrutura de armazenamento compartilhado, topologia de rede para acessar servidores PFS, o conjunto de bibliotecas usadas e seu padrão de acesso. Este último fator é alvo de muitas técnicas de otimização, pois define como as aplicações emitem suas requisições de E/S, ou seja, a operação (escreve ou lê), a espacialidade (contígua, 1D ou aleatória) e o tamanho das requisições [Boito et al. 2018]. Técnicas de otimização comumente fornecem melhorias para configurações de sistema e padrões de acesso específicos, mas não para todos eles. Na prática, o obstáculo que surge é que a carga de trabalho continua mudando conforme novas aplicações iniciam e terminam.

Neste trabalho, propomos eliminar a carga de treinamento do ambiente HPC através do uso da Nuvem. Desta forma, evitamos sobrecarregar ainda mais o sistema, que já é muito concorrido pelas aplicações que estão executando. Adaptamos o modelo de rede neural implementado por Bez et al. [Bez et al. 2019], a fim de possibilitar o treinamento em dispositivos *Tensor Processing Unit* (TPU) em Nuvem no *Google Cloud Platform*.

2. Implementação e resultados

O modelo consiste de três camadas, uma camada de entrada com 5 recursos (número de manipuladores de arquivo, tamanho médio, mínimo e máximo da solicitação e distância média de deslocamento), uma camada oculta com o mesmo número de neurônios e uma camada de saída com três unidades, uma para cada classe. As duas primeiras camadas usam uma função de ativação de Unidade Linear Retificada (ReLU) com função de inicialização normal do kernel. A camada de saída usa softmax para ajustar as saídas de cada unidade no intervalo [0, 1] e para

*O presente trabalho foi apoiado pela Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001, pelo edital CNPq/MCTI/FNDCT - Universal 18/2021 sob número 406182/2021-3, pela Petrobras sob número 2020/00182-5 e pelos projetos CIARS RITEs/FAPERGS e CI-IA FAPESP-MCTIC-CGI-BR.

garantir que a soma total das saídas seja igual a um. Para a fase de treinamento na Nuvem utilizando dispositivos TPU, algumas modificações se fizeram necessárias. I) Configuração da conexão com o dispositivo; II) Alteração do tamanho do lote para obtermos um lote global que será distribuído entre os núcleos da TPU, *i.e.*, se o tamanho do lote é 32, o lote global será 256 $(8 \times 32 = 256)$. Definimos uma estratégia e dentro do escopo desta estratégia, criamos e compilamos o modelo.

A Figura 1 apresenta os tempos de treinamento de 10 execuções para cada arquitetura. O tempo médio de treinamento no V100 foi $\approx 1,74\times$ mais rápido do que no P100, e o tempo médio de treinamento do TPUv2 foi $\approx 1,41\times$ mais rápido do que no P100. No entanto, usando o V100, observamos um aumento de velocidade de $\approx 1,24\times$ em comparação com o TPUv2. Por outro lado, o tempo médio de treinamento no TPUv3 foi $\approx 1,68\times$ mais rápido que no P100 e $\approx 1,20\times$ mais rápido que no TPUv2. Porém, o aumento de velocidade observado no V100 sobre o TPUv2 não é observado em comparação com o TPUv3. O V100 foi apenas $\approx 1,03\times$ mais rápido que o TPUv3. A primeira hipótese para explicar esse resultado é que o contexto da aplicação pode influenciar negativamente o desempenho por se tratar de um modelo relativamente pequeno e com poucos parâmetros. O segundo ponto é que as TPUs podem funcionar melhor quando o tamanho do lote é maior, aproveitando melhor a memória da TPU.

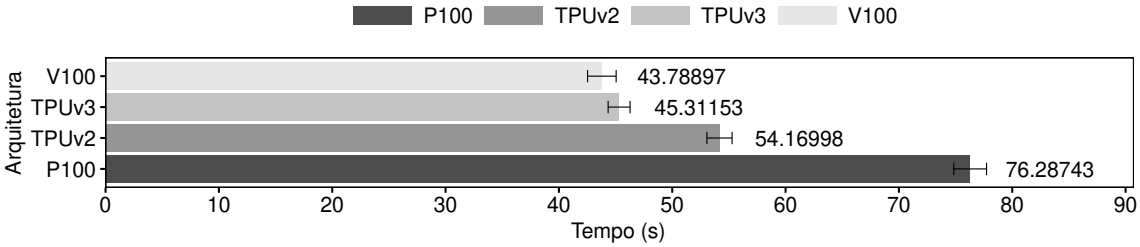

Figura 1. Tempos de treinamento da Rede Neural conforme a arquitetura utilizada.

3. Conclusões e trabalhos futuros

Concluímos que o treinamento em dispositivos TPU na Nuvem mostra-se uma solução viável, eliminando a carga de treinamento de sistema e mantendo a precisão do modelo. A fase de treinamento pode ser feita em segundo plano e pode levar mais tempo para ser concluída. Além disso, será necessário apenas atualizar o modelo, o que não deve acontecer com a frequência das previsões em tempo de execução. Isso evita que o sistema pare para uma nova fase de treinamento ou atualização. Trabalhos futuros estenderão a avaliação de desempenho para diferentes TPUs na Nuvem, variando a versão e quantidade total de núcleos, como TPUv3-32 e TPUv4.

Referências

Bez, J. L., Boito, F. Z., Nou, R., Miranda, A., Cortes, T., and Navaux, P. O. A. (2019). Detecting I/O access patterns of HPC workloads at runtime. In *2019 31st International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*, pages 80–87.

Boito, F. Z., Inacio, E. C., Bez, J. L., Navaux, P. O., Dantas, M. A., and Denneulin, Y. (2018). A checkpoint of research on parallel I/O for high-performance computing. *ACM Computing Surveys (CSUR)*, 51(2):1–35.

Ross, R. B., Thakur, R., et al. (2000). PVFS: A parallel file system for linux clusters. In *Proceedings of the 4th annual Linux showcase and conference*, pages 391–430.