Avaliação do Desempenho de Aplicações *CUDA* em um Ambiente com a Tecnologia *Multi-Instance GPU*

Pablo H. S. de Faria[1], **Marcelo Lobosco**[1,2]

[1]Laboratório de Fisiologia Computacional (FISIOCOMP) – Universidade Federal de Juiz de Fora (UFJF) Juiz de Fora – MG – Brazil

[2]Programa de Pós-graduação em Modelagem Computacional (PPGMC)
Universidade Federal de Juiz de Fora (UFJF)
Juiz de Fora – MG – Brazil

Corresponding author: pablohenrique.silva@estudante.ufjf.br

Abstract. *Computer clusters offer computational resources, primarily CPUs and GPUs, to users for various purposes. A key challenge in cluster management is optimizing throughput, especially when applications contend for scarce resources like GPUs. NVIDIA's Multi-Instance GPU (MIG) addresses this issue by virtualizing a single physical GPU into multiple virtual GPUs, enabling simultaneous service to multiple users. This paper employs the NAS parallel benchmark to assess performance differences between running an application on a GPU with and without MIG enabled. Results indicate a performance degradation of up to* 54 *times when computational resources are reduced through virtualization.*

Resumo. *Clusters de computadores fornecem recursos computacionais, principalmente CPUs e GPUs, para seus usuários para diversos fins. Um dos principais desafios no gerenciamento de clusters é otimizar a vazão, especialmente quando aplicativos são executados em recursos disputados com poucas unidades disponíveis, como GPUs. Multi-Instance GPU (MIG), desenvolvido pela NVIDIA, é uma das técnicas utilizadas para abordar esse problema. MIG permite que uma única GPU física seja virtualizada em várias GPUs virtuais, permitindo que ela atenda a vários usuários simultaneamente. Este artigo utiliza o benchmark paralelo NAS para avaliar as diferenças de desempenho entre executar um aplicativo em uma GPU com e sem virtualização habilitada. Os resultados mostram uma degradação no desempenho de até* 54 *vezes com a redução dos recursos computacionais.*

1. Introdução

Um *cluster* é um conjunto de computadores interconectados que oferece de forma eficiente serviço de execução de trabalhos para seus usuários. Ele pode fornecer recursos como *CPUs* e *GPUs* para a execução de aplicações paralelas que exigem alto desempenho. No entanto, alguns recursos, como *GPUs*, podem ser limitados, tornando o uso eficiente do *hardware* um desafio. Por exemplo, um grande número de usuários competindo pelas *GPUs* pode gerar longas filas de espera e comprometer a vazão do *cluster*.

Durante a execução de tarefas em *GPUs*, é possível que nem todo o seu poder computacional seja utilizado, deixando parte do *hardware* ocioso enquanto outros usuários

aguardam sua vez na fila de execução. Essa subutilização pode ocorrer mesmo em tarefas que demandam alto poder computacional, como aplicações *deep learning*, devido aos diferentes gargalos de recursos presentes em suas cargas de trabalho [Hu et al. 2021, Li et al. 2022b]. Nesses casos, uma solução seria permitir o uso simultâneo das *GPUs* por diversos processos dos usuários. No entanto, essa abordagem não garante a ausência de interferências mútuas durante a execução das aplicações. A NVIDIA propôs a tecnologia *Multi-Instance GPU* (*MIG*) [NVIDIA Corporation 2024, Choquette et al. 2021] como solução para virtualizar *GPUs* e atender a mais processos simultâneos, potencialmente reduzindo o tempo de espera total na fila de execução de trabalhos em um *cluster*.

Este trabalho tem como objetivo avaliar o impacto da virtualização de *GPUs* com *MIG* no desempenho de aplicações científicas escritas em *CUDA*. Utilizando uma implementação para *GPUs* [Araujo et al. 2021] do *NAS Parallel Benchmark* (*NPB*) [Bailey et al. 2010], este estudo busca responder à seguinte pergunta: como a virtualização de *GPUs* afeta a execução de códigos científicos em um ambiente de computação de alto desempenho? Embora estudos anteriores tenham investigado o impacto do *MIG* em aplicações de aprendizado de máquina [Li et al. 2022a], a influência dessa tecnologia no desempenho de aplicações científicas *CUDA* ainda não foi amplamente explorada. Este trabalho contribui para preencher essa lacuna na literatura.

O restante deste trabalho está organizado da seguinte forma. A Seção 2 apresenta os métodos utilizados na pesquisa. A Seção 3 apresenta os resultados obtidos, incluindo a análise do desempenho das aplicações *CUDA* em *GPUs* com e sem *MIG*. Por fim, a Seção 4 apresenta as conclusões do trabalho e sugere direções para futuras pesquisas.

2. Métodos

O *Multi-Instance GPU* (*MIG*) [NVIDIA Corporation 2024] é uma tecnologia criada pela NVIDIA que permite virtualizar uma *GPU* em múltiplas instâncias de menor capacidade computacional. Essa virtualização, introduzida na *GPU* NVIDIA `A100` [Choquette et al. 2021] e disponível em outras *GPUs* para servidores a partir da arquitetura Ampere [NVIDIA Corporation 2024], permite que administradores dividam suas *GPUs* para atender melhor aos usuários, reduzindo o tempo de espera e garantindo o isolamento entre as tarefas [Choquette et al. 2021]. Essa tecnologia é especialmente útil em situações onde há subutilização dos recursos da *GPU* por parte de um grande número de tarefas. Esta seção apresentará as configurações do *MIG* e os *benchmarks* utilizados neste trabalho.

2.1. Configuração do *MIG*

A *GPU* `A100` permite criar até 7 instâncias de *GPU*, cada uma com um perfil distinto de recursos, com número de *SMs* (*streaming multiprocessors*) e quantidade de memória, dentre outras características de *hardware* [Choquette et al. 2021, NVIDIA Corporation 2024]. A terminologia NVIDIA estabelece os conceitos de fatias de: a) memória (*GPU Memory Slice*), b) multiprocessador de *streaming* (*GPU Streaming Multiprocessor - SM - Slice*), e c) *GPU* (*GPU Slice*). Uma fatia de *GPU* combina uma única fatia de memória com uma única fatia de *SM*. Outro conceito é o de instância de *GPU* (*GI*), que pode ser entendido como uma combinação de fatias de *GPU* e de outros *hardwares* (*DMAs*, *NVDECs*, *etc*.). Aplicações em uma *GI* sempre compartilham

a sua fatia de memória e outros *hardwares* associados, mas suas fatias de *SM* podem ser subdivididas em instâncias de computação (*CI*). Seguindo essa terminologia, os perfis são identificados por nomes compostos pela *GI*, pela *CI* e pelo tamanho de sua fatia de memória [NVIDIA Corporation 2024]. A Tabela 1 exemplifica os perfis disponíveis na *GPU* `A100` [Choquette et al. 2021], e suas respectivas características de *hardware*. Nesta tabela, o nome `3g.40gb` indica um dispositivo *MIG* cujo tamanho da *GI* e da *CI* é de três fatias de computação e a fatia de memória é de 40GB. Nesse exemplo, a *CI* e a *GI* possuem tamanhos iguais, logo, o nome assume apenas um elemento para indicar os dois [NVIDIA Corporation 2024]. Em outras palavras, caso apenas uma *CI* seja criada para o perfil utilizado, de modo que ela utilize todos os *SMs* da *GI*, o nome dela ficará implícito no nome do perfil [NVIDIA Corporation 2024]. Por outro lado, caso as *GI* sejam divididas em grupos menores, a *CI* será explicitada no nome do *MIG*. Por exemplo, o perfil `3g.20gb` pode ser dividido em `2c.3g.20gb` e `1c.3g.20gb`, em que `2c` e `1c` representam as *CIs* após a divisão e `3g` permanece referenciando a *GI* [NVIDIA Corporation 2024].

Uma vez criada uma instância, esta recebe um identificador único (*UUID*). Para a utilização desta instância, o usuário deve definir a variável de ambiente `CUDA_VISIBLE_DEVICES` em conjunto com o *UUID* da instância a ser utilizada pela aplicação. Caso essa variável não seja explicitada, o ambiente *runtime* buscará a primeira instância disponível e a utilizará [NVIDIA Corporation 2024].

Tabela 1. Perfis de *MIG* disponíveis na *GPU* NVIDIA `A100` 80GB.

Nome do Perfil	Memória	Fração de *SMs*	*Cache*	Instâncias Disponíveis
`1g.10gb`	10GB	1/7	1/8	7
`1g.10gb+me`	10GB	1/7	1/8	1
`1g.20gb`	20GB	1/7	1/8	4
`2g.20gb`	20GB	2/7	2/8	3
`3g.40gb`	40GB	3/7	4/8	2
`4g.40gb`	40GB	4/7	4/8	1
`7g.80gb`	80GB	7/7	8/8	1

Atualmente, apenas o administrador pode criar e destruir *GIs*. Algumas pesquisas propõem o emprego de escalonadores capazes de automatizar este processo [Oberholzer, Pascal 2021].

2.2. *Benchmarks*

Para avaliar o desempenho da tecnologia *MIG*, utilizamos o conjunto de *benchmarks NAS Parallel* (*NPB*) [Bailey et al. 2010]. O *NPB* é composto por cinco núcleos (CG, EP, FT, IS, MG), que representam problemas computacionais comuns em diversas áreas da ciência e engenharia, e 3 pseudoaplicações (BT, LU e SP), projetadas para medir o desempenho de diferentes aspectos de um sistema computacional. Uma vez que a versão original do *NPB* não foi desenvolvida para *GPUs*, utilizamos uma adaptação para *CUDA*[1] [Araujo et al. 2021]. Essa adaptação incorpora diversas técnicas de otimização

[1]Disponível em https://github.com/GMAP/NPB-GPU, *commit* f65e28b

para *GPUs* que visam maximizar o uso de seus recursos e, consequentemente, o desempenho das aplicações [Araujo et al. 2021].

O *Conjugate Gradient* (CG) calcula uma estimativa do menor autovalor de uma matriz esparsa. É caracterizado por computações irregulares e multiplicação de matrizes não estruturadas. *Embarrassingly Parallel* (EP) gera um conjunto de números aleatórios e calcula suas derivadas Gaussianas. Na versão em *CUDA*, esse conjunto de números é dividido em subconjuntos e cada um é associado a um bloco de *threads*, em que cada *thread* vai computar apenas um dos números [Araujo et al. 2021]. O *benchmark Fourier Transform* (FT) resolve uma equação diferencial 3D usando Transformada Rápida de Fourier. O *Integer Sort* (IS) ordena números inteiros, sendo útil para avaliar tanto a velocidade de computação de inteiros quanto o desempenho da comunicação [Bailey et al. 2010]. *Multi-Grid* (MG) é um método para resolver equações diferenciais usando uma hierarquia de discretizações. Na versão *CUDA*, esse *benchmark* possui baixa dependência de dados e possibilidade de agrupar o acesso aos dados na memória [Araujo et al. 2021]. O mesmo problema de CFD (*Computational Fluid Dynamics*) serve de base para as três pseudoaplicações. Todas resolvem um sistema sintético de equações diferenciais parciais não-lineares usando abordagens distintas. *Block Tri-diagonal Solver* (BT) resolve o sistema usando blocos tridiagonais. *Lower-Upper Gauss-Seidel* (LU) resolve o sistema usando a fatoração LU. Por fim, *Scalar Pentadiagonal* (SP) resolve o sistema usando escalares pentadiagonais.

O *NPB* oferece diferentes tamanhos de problemas (`S`, `W`, `A`, `B`, `C`, `D`, `E`, e `F`) para simular diversas cargas de trabalho. O tamanho `S` é o que lida com a menor instância de entrada e `F` o maior. Entretanto, é importante salientar que a progressão do tamanho da entrada entre as classes não é linear: `S` é considerado pequeno e feito para testes rápidos; `W` para cargas de trabalho para sistemas com pouca memória; `A`, `B` e `C` são problemas de teste padrão, sendo que uma classe é 4 vezes maior que a outra, e `D`, `E` e `F` são problemas de teste grandes e cada uma é 16 vezes maior que a anterior.

3. Resultados e Discussão

Esta seção apresentará os resultados dos experimentos computacionais conduzidos em um ambiente composto por dois computadores idênticos. Cada computador é equipado com dois processadores AMD EPYC 7713, totalizando 128 núcleos físicos. Cada núcleo possui 64 KB de *cache* de dados L1, 64 KB de *cache* de instruções L1, 512 KB de *cache* unificada L2 e compartilha 32 MB de *cache* L3 com outros 7 núcleos. Cada computador possui 528 GB de memória RAM e duas *GPUs* NVIDIA `A100` de 80 GB. Cada *GPU* possui 108 *SMs*, com 64 *CUDA cores* por *SM*, totalizando assim 6.912 *CUDA cores*. O Sistema Operacional utilizado foi o GNU/Linux, *kernel* `4.18.0-513.9.1.el8_9`. O compilador NVCC (versão 12.3) foi utilizado com a *flag* de otimização `-O3`. Foram realizadas 10 execuções de cada aplicação, e calculados os tempos médios de execução e desvios.

3.1. Configuração dos Experimentos

Dos perfis disponíveis na *GPU* `A100` (veja a Tabela 1), utilizamos neste trabalho os seguintes: `1g.10gb`, `1g.20gb`, `2g.20gb`, `3g.40gb` e `4g.40gb`. O perfil `1g.10gb+me` foi excluído da análise, uma vez que as aplicações avaliadas não utilizam recursos específicos para processamento de mídia. O perfil `7g.80gb` também foi

descartado, pois suas características de *hardware* são equivalentes às de uma *GPU* sem *MIG*. Uma das máquinas de teste foi configurada com múltiplas instâncias de *GPU*, conforme os perfis listados anteriormente, enquanto a outra máquina manteve suas *GPUs* intactas, sem a criação de instâncias. Dessa forma, foi possível avaliar o desempenho das aplicações em cenários com sem o uso da tecnologia *MIG*. O *MIG* foi criado antes da execução das aplicações, não afetando assim seus tempos de execução. Deve-se salientar que nenhuma subdivisão adicional das *CIs* foi realizada em nenhum dos perfis utilizados.

Neste trabalho, o tamanho escolhido para a execução do *NPB* foi o `C`, conforme apresentado na Tabela 2. A escolha deste tamanho é justificada pela quantidade de memória demandada pelas aplicações e disponível na *GPU*, sendo o tamanho `C` o maior tamanho suportado para a execução no dispositivo. O número de *threads* usado na execução dos *benchmarks* foi mantido como o definido como padrão pela versão *NAS CUDA* utilizada em nosso estudo [Araujo et al. 2021].

Tabela 2. *Benchmarks* usados e o tamanho de seus problemas na classe `C`.

Nome	Tamanho do Problema	Nome	Tamanho do Problema
BT	$162 \times 162 \times 162$	IS	134.217.728
CG	150.000	LU	$162 \times 162 \times 162$
EP	8.589.934.592	MG	$512 \times 512 \times 512$
FT	$512 \times 512 \times 512$	SP	$162 \times 162 \times 162$

Para a coleta das informações apresentadas neste artigo, foram utilizadas as saídas padrão dos *benchmarks*, que apresentam os tempos de execução, e, para observar as demais métricas de desempenho, utilizamos o `NVIDIA Nsight Compute`. Os experimentos foram realizados de duas formas, ambas utilizando um sistema de submissão de tarefas para garantir que não ocorressem interferências de processos de outros usuários durante a execução dos testes.

Na primeira forma, cada núcleo e pseudoaplicação que compõe o *benchmark* foi submetido para execução de forma isolada, garantindo assim que um não interferisse no outro. Em outras palavras, mesmo que uma *GPU* possuísse múltiplas instâncias disponíveis, apenas uma era utilizada por vez. Para isso, utilizamos um *script* que gerava um roteiro de execução para cada instância do *MIG* a ser usada e para cada núcleo/pseudoaplicação, cobrindo todas as combinações. Dentro de cada roteiro de execução, foi passado o valor do *UUID* da instância a ser utilizada pela variável de ambiente `CUDA_VISIBLE_DEVICES`. Referenciaremos esta primeira forma de execução como isolada. Na segunda forma, testamos a possível interferência que a execução em um dispositivo *MIG* poderia gerar em outro. Para isso, foram feitos testes em que várias cópias do mesmo núcleo/pseudoaplicação eram executadas simultaneamente em instâncias diferentes de uma mesma *GPU*. Nesse experimento usamos uma *GPU* com seis instâncias, sendo elas: uma `2g.20gb`, uma `1g.20gb` e quatro `1g.10gb`. Assim, seis cópias do mesmo núcleo/pseudoaplicação eram executadas simultaneamente, uma em cada instância. Referenciaremos esta segunda forma de execução como conjunta.

3.2. Análise dos Resultados

A Tabela 3 apresenta os tempos médios de execução, em segundos, para cada *benchmark* do *NPB*, considerando a execução isolada na *GPU*, com e sem o uso de *MIG*. O tempo

médio de execução é apresentado como primeiro valor de cada coluna. Nestas execuções, o maior desvio padrão registrado foi de 0,9%. O segundo valor de cada coluna apresenta o desempenho relativo da execução com *MIG* em relação à execução sem *MIG*, *i.e.*, quantas vezes mais lenta foi a execução com o uso do *MIG*. Os resultados evidenciam que a divisão de recursos da *GPU* através da tecnologia *MIG* resulta em um aumento significativo nos tempos de execução de todos os *benchmarks*.

A análise dos resultados revela uma correlação entre a redução na quantidade de *SMs* e de memória disponível por instância e o aumento no tempo de execução. Em diversos casos, a degradação de desempenho observada é proporcional à redução no número de *SMs* disponíveis. Por exemplo, a *GPU* sem *MIG* tem $7/4 = 1.75$ mais *SMs* do que o perfil `4g.40gb`, e o desempenho foi pior em uma taxa próxima a esse valor para EP, FT, MG e SP. A literatura reporta resultados similares, com redução de desempenho maior sendo observada em perfis com menor quantidade de *SMs* [Li et al. 2022a]. Além da quantidade de *SMs*, a quantidade de memória disponível por instância também influencia significativamente o desempenho. A comparação entre os perfis `1g.10gb` e `1g.20gb` demonstra que uma maior quantidade de memória resulta em uma menor degradação de desempenho, mesmo com o mesmo número de *SMs*. A memória pode ainda ajudar a explicar outros resultados relacionados a variação considerável na magnitude da degradação observada nos diferentes núcleos e pseudoaplicações. Por exemplo, IS apresenta uma degradação muito maior em relação aos demais *benchmarks* que compõem o *NPB*, degradando-se mais rapidamente com a redução na memória disponível. Notamos efeitos semelhantes em outros *benchmarks*, como em BT: observa-se que a diferença entre o tempo médio dos perfis `1g.10gb` e `1g.20gb`, que possuem a mesma quantidade de *SMs* mas distintas quantidades de memória, foi de cerca de $7,7$ segundos, porém, a diferença entre `3g.40gb` e `4g.40gb`, que possuem distintas quantidades de *SMs* mas a mesma quantidade de memória, foi de apenas $0,98$ segundos.

Tabela 3. Tempos médios de execução, em segundos, para o *NPB* na *GPU* `A100` 80GB considerando a forma isolada de execução. São reportados os tempos sem o uso de *MIG* (*S/MIG*) e com o uso de diferentes instâncias. Também é apresentado, como segundo valor em cada coluna, quantas vezes mais lenta é a execução com *MIG* em relação à execução sem *MIG*.

Nome	S/ *MIG*	`4g.40gb`	`3g.40gb`	`2g.20gb`	`1g.20gb`	`1g.10gb`
BT	5,54	8,02 / 1,45	9,00 / 1,62	15,60 / 2,82	23,20 / 4,19	30,89 / 5,58
CG	1,03	1,63 / 1,58	1,97 / 1,91	3,16 / 3,07	5,47 / 5,31	6,27 / 6,09
EP	0,13	0,22 / 1,69	0,26 / 2,00	0,40 / 3,08	0,68 / 5,23	0,75 / 5,77
FT	2,40	4,39 / 1,83	4,61 / 1,92	8,60 / 3,58	10,28 / 4,28	16,95 / 7,06
IS	0,03	0,17 / 5,67	0,17 / 5,67	0,66 / 22,00	0,65 / 21,67	1,64 / 54,67
LU	3,89	5,42 / 1,39	6,53 / 1,68	12,21 / 3,14	16,97 / 4,36	27,54 / 7,08
MG	0,26	0,44 / 1,69	0,54 / 2,08	0,91 / 3,50	1,48 / 5,69	2,25 / 8,65
SP	3,13	5,44 / 1,74	5,82 / 1,86	13,16 / 4,20	14,45 / 4,62	25,56 / 8,17

Para avaliar a possível interferência entre dispositivos durante a execução em múltiplas instâncias de *GPU*, foram realizados experimentos de execução conjunta. Os resultados obtidos indicam que não houve diferenças estatisticamente significativas nos tempos médios de execução em comparação com os resultados apresentados na Tabela 3. Considerando que as diferenças observadas nos tempos de execução estavam dentro do

intervalo de confiança dos resultados individuais, optou-se por não apresentar os dados completos da execução conjunta neste trabalho. Essa observação corrobora os resultados de estudos anteriores [NVIDIA Corporation 2024, Choquette et al. 2021], que demonstram a eficiência do isolamento de recursos computacionais entre as diferentes instâncias de *GPU* proporcionado pela tecnologia *MIG*.

Para aprofundar a análise dos resultados dos *benchmarks*, foram coletadas métricas de desempenho utilizando o `NVIDIA Nsight Compute`. Para as execuções sem *MIG*, foram obtidas métricas (Tabela 4) que indicam a porcentagem de tempo em que os núcleos da *GPU* estão realizando cálculos (vazão da computação), a porcentagem de tempo em que a memória está sendo acessada (vazão da memória), dados sobre as taxas de acerto e utilização das *caches* L1 e L2, bem como o percentual de tempo em que a *DRAM* está em uso (vazão da *DRAM*). Os percentuais de tempo em que a memória e os canais (*pipes*) de memória estão ocupados também foram coletados. Restrições da ferramenta `Nsight` não permitem a coleta de métricas de desempenho em *GPUs* com múltiplas fatias *MIG*. Assim não foi possível obter os mesmos dados para as execuções com *MIG*, limitando a comparação do desempenho das duas configurações de modo direto.

Tabela 4. Métricas de desempenho para execução do *NPB* na *GPU* `A100`, sem *MIG*. Todos os valores são percentuais.

Métrica	BT	CG	EP	FT	IS	LU	MG	SP
Vazão da Computação	19,0	4,5	62,5	5,3	28,1	7,8	31,0	23,8
Vazão da Memória	80,8	33,0	46,3	67,2	50,6	34,2	66,8	58,7
Acerto na L1	55,7	0,0	54,6	44,4	96,9	79,9	12,2	0,0
Vazão da *Cache* L1	2,8	44,3	33,4	84,5	39,9	34,2	62,3	54,4
Acerto na L2	68,0	81,6	74,5	100	99,3	99,9	77,6	99,8
Vazão da *Cache* L2	83,2	46,1	70,7	65,6	50,6	34,2	75,4	66,0
Vazão da *DRAM*	80,8	10,3	46,1	67,2	4,8	0,0	53,8	58,7
Memória Ocupada	42,6	20,6	33,1	33,3	44,5	13,5	66,8	31,7
Pipes de Mem. Ocup.	18,8	4,1	4,6	5,3	2,6	2,5	27,8	5,2

A análise das métricas de desempenho permite compreender melhor os resultados apresentados na Tabela 3. Por exemplo, o *benchmark* BT, que apresenta um aumento significativo no tempo de execução com a redução dos recursos disponíveis, apresenta alta utilização de memória e *cache* L2. Essa alta dependência de memória explica a sensibilidade do *benchmark* à redução da largura de banda e do tamanho da *cache*. O *benchmark* CG, por sua vez, apresenta baixa vazão de computação e alta dependência do *cache* L2. Consequentemente, a redução do número de *SMs* tem um impacto menor no desempenho do que a redução da memória. O *benchmark* EP, com alta vazão de computação e menor dependência de memória, é menos afetado pela redução dos recursos. No caso de IS, o enorme aumento no tempo de execução deve-se a alta taxa de acerto nas *caches*, especialmente a L1. Com *MIG*, a redução do tamanho das *caches* e largura de banda de memória impacta severamente o desempenho. Os demais *benchmarks* (FT, LU, MG e SP) apresentam comportamentos semelhantes, com a degradação do desempenho sendo mais pronunciada para aqueles que apresentam maior dependência de memória e *cache*.

4. Conclusão e trabalhos futuros

Este trabalho investigou o impacto da virtualização de *GPUs* com a tecnologia MIG na execução de códigos científicos em um ambiente de alto desempenho, utilizando para este propósito o *benchmark NPB*. Os resultados mostram que a redução de recursos computacionais e de memória nas instâncias virtuais causa degradação de desempenho, especialmente em *benchmarks* que demandam alta utilização de memória e *cache*. No entanto, instâncias com mais recursos podem equilibrar desempenho e suporte a múltiplos usuários simultâneos. A análise também confirmou a eficácia do isolamento de recursos, crucial para o uso simultâneo por vários usuários. Como trabalhos futuros, propomos avaliar o desempenho de outras aplicações que exploram características específicas das *GPUs*, como processamento de mídia, além de desenvolver um mecanismo para auxiliar na escolha da instância mais adequada. Por fim, pretendemos quantificar o impacto da virtualização na vazão de um sistema multiusuário.

Agradecimentos

Este trabalho de pesquisa foi parcialmente financiado pela UFJF, CNPq (308745/2021-3), FAPEMIG (APQ-02513-22) e FINEP (SOS Equipamentos 2021 AV02 0062/22).

Referências

Araujo, G., Griebler, D., Rockenbach, D. A., Danelutto, M., and Fernandes, L. G. (2021). NAS parallel benchmarks with CUDA and beyond. *Software: Practice and Experience*, 53(1):53–80.

Bailey, D., Barszcz, E., Barton, J., Browning, D., Carter, R., Dagum, L., Fatoohi, R., Fineberg, S., Frederickson, P., Lasinski, T., et al. (2010). The NAS parallel benchmarks,". Technical report, Technical Report RNR-94-007, NASA Ames Research Center,(March 1994).

Choquette, J., Gandhi, W., Giroux, O., Stam, N., and Krashinsky, R. (2021). NVIDIA A100 tensor core GPU: Performance and innovation. *IEEE Micro*, 41(2):29–35.

Hu, Q., Sun, P., Yan, S., Wen, Y., and Zhang, T. (2021). Characterization and prediction of deep learning workloads in large-scale GPU datacenters. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '21. ACM.

Li, B., Gadepally, V., Samsi, S., and Tiwari, D. (2022a). Characterizing multi-instance gpu for machine learning workloads. In *2022 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 724–731.

Li, B., Patel, T., Samsi, S., Gadepally, V., and Tiwari, D. (2022b). MISO: exploiting multi-instance GPU capability on multi-tenant GPU clusters. In *Proceedings of the 13th Symposium on Cloud Computing*, SoCC '22. ACM.

NVIDIA Corporation (2024). *NVIDIA multi-instance GPU user guide*. NVIDIA Corporation.

Oberholzer, Pascal (2021). Scheduling for MIG-capable GPUs: Accelerator-aware operating system scheduling. Master's thesis, ETH Zurich.