Relatório - MONAN

Avaliação Computacional de Núcleos Dinâmicos

Carlos Renato de Souza (INPE)
Denis Magalhães De Almeida Eiras (INPE)
Eduardo Georges Khamis (INPE)
João Messias Alves da Silva (INPE)
Luiz Flávio Rodrigues (INPE)
Roberto Pinto Souto (LNCC)

30 de Agosto de 2022

Agradecimentos

Agradecemos a DELL Brasil pela permissão de uso das máquinas Minerva e Rattler em seus laboratórios.

Ao apoio imprescindível do Dr. Roberto Pinto Souto do LNCC nos testes com os núcleos dinâmicos.

1. Introdução

Prever a condição climática em escalas de horas, dias e meses à frente é um dos maiores desafios técnico-científicos enfrentados pela humanidade. No entanto, este desafio tem sido vencido por diversos países e tem se demonstrado a importância de se ter informações confiáveis em tempo real das condições atmosféricas e oceânicas presentes e futuras. A importância se qualifica principalmente pela preservação da vida de pessoas e mitigação de prejuízos socioeconômicos advindos do uso destas informações. O Brasil tem um relevante histórico de desenvolvimento e aplicação de modelagem para a previsão de tempo e clima.

Porém, os esforços estão fragmentados e dispersos em vários grupos e modelos que não interagem entre si. Como resultado, as previsões hoje produzidas no país não utilizam em sua plenitude o atual conhecimento científico considerado como estado-da-arte em modelagem numérica, métodos e dados e, assim, há um grande espaço para melhorar a competitividade em relação aos produtos de previsão gerados por outras instituições internacionais. Nesse contexto, há um potencial enorme do país se colocar em um patamar superior com a focalização e unificação dos esforços e expertises através da adoção de um sistema unificado de modelagem do Sistema Terrestre que seja comunitário e que atenda todas as escalas (temporais e espaciais) de fenômenos de relevância para a sociedade brasileira.

Para tanto foi desenvolvido um projeto que tem por objetivo principal desenvolver tal sistema de modelagem ancorado em um robusto sistema de assimilação de dados e aperfeiçoado com técnicas de inteligência artificial. Com tal sistema, o Brasil alcançará o estado-da-arte em previsão da atmosfera e oceanos para o benefício da sociedade brasileira e da América do Sul em geral.

Esse projeto ancorado no Termo de Abertura de Projeto, SEI 01340.005344/2021-50, tem como objeto o Desenvolvimento comunitário de um modelo numérico do Sistema Terrestre adaptado para as condições tropicais e subtropicais da América do Sul e de suas aplicações para previsão de tempo, clima e ambiente em escalas espaço-temporais relevantes para a sociedade brasileira.

No desenvolvimento do projeto alinham-se diversos órgãos da ciência, da educação e da meteorologia brasileira formando um esforço comunitário capitaneado pelo INPE. Esse esforço é gerenciado por um comitê científico que tem como função precípua planejar e gerenciar o desenvolvimento do modelo comunitário.

O modelo numérico oriundo do projeto foi denominado **MONAN,** acrônimo de **Model for Ocean-laNd-Atmosphere predictioN** e tem em sua primeira fase a avaliação para escolha de um Núcleo Dinâmico (ND) onde serão acoplados os códigos computacionais contendo as físicas adequadas às características específicas do modelo.

A avaliação dos NDs ficou a cargo do Grupo de Computação Científica (GCC), parte da Divisão de Modelagem Numérica (DIMNT) da Coordenação de Ciências da Terra (CGCT) que é uma das

coordenações do Instituto Nacional de Pesquisas Espaciais (INPE). A tarefa tem o auxílio substancial do Laboratório Nacional de Computação Científica (LNCC).

Os NDs avaliados e objetos deste relatório foram escolhidos por estarem no estado da arte em termos computacionais e já terem passado por processo de avaliação em passado recente pela NOAA - National Oceanic and Atmospheric Administration, órgão de pesquisa e operacional norte americano e por alguns deles estarem operacionais em diversos centros. Os testes de escolha realizados pela NOAA se deram entre os anos de 2015 e 2017 e culminaram na escolha de um ND que foi posteriormente usado no modelo global da instituição[1].

Há que se ressaltar que os NDs continuaram a receber atualizações após os testes realizados pela NOAA e portanto estão modernizados e com novas capacidades exibindo assim novas performances computacionais e capazes de utilizar arquiteturas computacionais diversas. Optou-se então para que fossem realizados outros testes com essas novas versões disponibilizadas de forma a se escolher aquele que melhor se adapta às necessidades do modelo MONAN. Tal processo é mostrado nos próximos capítulos deste relatório.

2. Metodologia

Modelos numéricos são softwares complexos. Softwares tratam de duas classes de requisitos:

a) Requisitos funcionais
b) Requisitos Não Funcionais

Requisitos funcionais são aqueles diretamente ligados aos interesses dos usuários na funcionalidade do software, isto é, a produção de resultados adequados que atendem minimamente a função para qual o software foi especificado. No caso de modelos de previsão de tempo, caso específico do software em avaliação, trata-se do acerto das previsões quando comparados com dados observados. Esse documento não trata desses requisitos e avalia o segundo grupo de requisitos, os requisitos não funcionais.

Requisitos não funcionais são totalmente ligados à qualidade do software. O desenvolvimento de um software tem ciclos contínuos de requisitos-desenvolvimento-testes-operação que permitem que a vida útil do software seja longa e possa sobreviver por décadas. O ciclo de vida do software se perpetua por gerações de desenvolvedores, os quais devem garantir a qualidade para que novas gerações possam dar continuidade ao desenvolvimento. Para isso torna-se fundamental o uso de técnicas e processos que facilitem o atendimento aos requisitos não funcionais.

A escolha do Core Dinâmico do MONAN levou em consideração alguns aspectos de Qualidade de Software (QS). Segundo a norma ISO/IEC 9126, revisada pela norma ISO/IEC 25010:2011, quando a

1 https://www.weather.gov/media/sti/nggps/Phase%202%20Dycore%20Evaluation%20Briefing%2022%20June%202016%20UMAC%20Final%20for%20posting%2006272016.pdf

QS se refere ao produto, estas fornecem uma estrutura para especificar características de QS e realizar comparações entre produtos de software. Outras características e métricas de QS encontradas em livros de engenharia de software também foram introduzidas. Estas normas, características e métricas serão utilizadas como base para a construção dos critérios de escolha do core Dinâmico. As Características de QS avaliadas estão detalhadas no Apêndice A.

Dentro do contexto de qualidade interna, onde o software é avaliado para ser reutilizado e modificado (ciclo de vida contínua), a definição das Características e Subcaracterísticas devem ser levantadas em função da área de aplicação do produto de software. Esta definição deve ser feita antes do início do desenvolvimento do mesmo. Produtos de maior porte devem ser subdivididos em módulos e cada um destes deve ter seus próprios conjuntos de Características e Subcaracterísticas. Portanto, um Core Dinâmico de qualidade deve ser utilizado considerando alguns aspectos principais da QS que são inerentes ao Produto.

Nas seções 3, 4 e 5 são apresentados, respectivamente, os resultados das Características de Manutenibilidade, Portabilidade e Eficiência, que estão relacionadas com a avaliação computacional dos NDs.

2.1. Modelos e Núcleos Dinâmicos

2.1.1 MPAS

O *Model for Prediction Across Scales* (MPAS)[2] é um projeto de desenvolvimento de um modelo acoplado que abrange vários modelos componentes do sistema terrestre, como por exemplo, modelos atmosféricos e oceânicos. Desenvolvido pelo grupo de modelagem climática do *Los Alamos National Laboratory* (COSIM) e pelo *National Center for Atmospheric Research* (NCAR) com o objetivo de criar um modelo acoplado capaz de realizar desde previsões de tempo regionais até simulações climáticas do sistema terrestre de alto nível.

O modelo usa a grade de Voronoi, formalmente chamada de *Spherical Centriodal Voronoi Tesselations* (SCVT), que é uma malha hexagonal semelhante a um favo de mel que pode ser estendido em algumas regiões e comprimido para maior resolução em outras, como mostra a Figura 1. Isso permite que os pesquisadores executem um modelo global em escalas de resolução de tempestades (alta-resolução) onde ele mais importa e resolução mais baixa em todos os demais lugares para conservar recursos computacionais. O NCAR lidera o desenvolvimento do componente atmosférico do MPAS enquanto que outros parceiros da comunidade de desenvolvimento do MPAS trabalham na parte do oceano, gelo terrestre e componentes de gelo marinho.

[2] mpas-dev.github.io

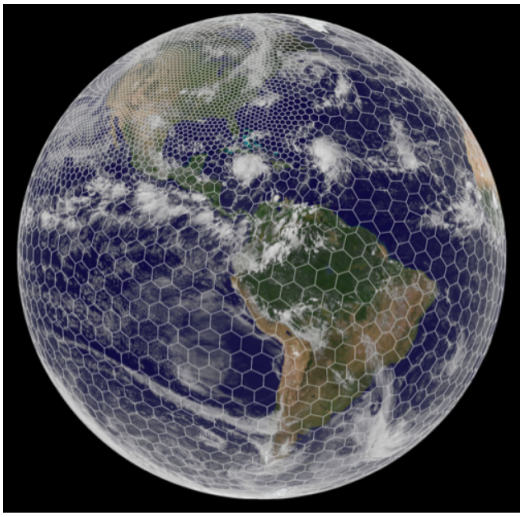

Figura 1: Ilustração da grade não estruturada de Voronoi do MPAS.

Códigos e artefatos

O pacote completo utilizado para a Previsão Global de Tempo, em sua resolução de 15km foi obtido à partir de:

- https://github.com/MPAS-Dev/MPAS-Model.git
- https://www2.mmm.ucar.edu/projects/mpas/benchmark/v6.x/MPAS-A_benchmark_15km_L56.tar.gz
- https://mpas-dev.github.io/

No primeiro link acima encontra-se o repositório oficial do modelo MPAS, nele podem ser baixados todas as versões dos códigos do modelo, para esta avaliação foi utilizada a versão 6.3.
No segundo link contém um pacote de *benchmark,* onde pode-se encontrar todos os arquivos e dados necessários para que uma rodada exemplo possa ser executada. Para esta avaliação executou-se o modelo MPAS na resolução de 15km com 56 níveis verticais.
No terceiro link é o local onde pode ser encontrado todo o material referencial do modelo MPAS, documentação detalhada, manual do usuário, publicações importantes, e muito mais.

Configurações utilizadas nos experimentos

Os parâmetros de configuração do modelo MPAS são definidos no arquivo *namelist.atmosphere.* Segue abaixo a lista dos parâmetros contidos no *namelist.atmosphere:*

```
&nhyd_model
   config_time_integration_order = 2
   config_dt = 90.0
   config_start_time = '2010-10-23_00:00:00'
   config_run_duration = '0_12:00:00'
```

```
   config_split_dynamics_transport = true
   config_number_of_sub_steps = 2
   config_dynamics_split_steps = 3
   config_h_mom_eddy_visc2 = 0.0
   config_h_mom_eddy_visc4 = 0.0
   config_v_mom_eddy_visc2 = 0.0
   config_h_theta_eddy_visc2 = 0.0
   config_h_theta_eddy_visc4 = 0.0
   config_v_theta_eddy_visc2 = 0.0
   config_horiz_mixing = '2d_smagorinsky'
   config_len_disp = 15000.0
   config_visc4_2dsmag = 0.05
   config_w_adv_order = 3
   config_theta_adv_order = 3
   config_scalar_adv_order = 3
   config_u_vadv_order = 3
   config_w_vadv_order = 3
   config_theta_vadv_order = 3
   config_scalar_vadv_order = 3
   config_scalar_advection = true
   config_positive_definite = false
   config_monotonic = true
   config_coef_3rd_order = 0.25
   config_epssm = 0.1
   config_smdiv = 0.1
/
&damping
   config_zd = 22000.0
   config_xnutr = 0.2
/
&io
   config_pio_num_iotasks = 0
   config_pio_stride = 1
/
&decomposition
   config_block_decomp_file_prefix = 'x1.2621442.graph.info.part.'
/
&restart
   config_do_restart = false
/
&printout
   config_print_global_minmax_vel = true
   config_print_detailed_minmax_vel = false
/
&IAU
   config_IAU_option = 'off'
   config_IAU_window_length_s = 21600.
/
&physics
   config_sst_update = false
   config_sstdiurn_update = false
   config_deepsoiltemp_update = false
   config_radtlw_interval = '00:15:00'
   config_radtsw_interval = '00:15:00'
   config_bucket_update = 'none'
```

```
   config_physics_suite = 'mesoscale_reference'
/
&soundings
   config_sounding_interval = 'none'
/
```

Nos parâmetros listados acima, destacam-se as linhas em azul que definem a data e hora de início da simulação (**config_start_time**) e o tempo de duração das simulações (**config_run_duration**), que para esta avaliação foi de 12 horas de duração.

2.1.2 GEF

Global Eta Framework (GEF) é um modelo atmosférico global desenvolvido em coordenadas lineares (esfera cubada) e capaz de funcionar em grades esféricas retangulares arbitrárias, usando *stepwise* ("Eta") para a representação do terreno.[3] Isso permite que o modelo possa rodar em várias grades esféricas retangulares. O GEF usa a globalização do modelo regional Eta[4] que tem como uma das grandes vantagens a formulação stepwise do terreno que reduz o erro do gradiente de pressão do sistema sigma na presença de montanhas íngremes. Essa capacidade é particularmente útil para a topografia típica da América do Sul com a cordilheira de montanhas dos Andes.

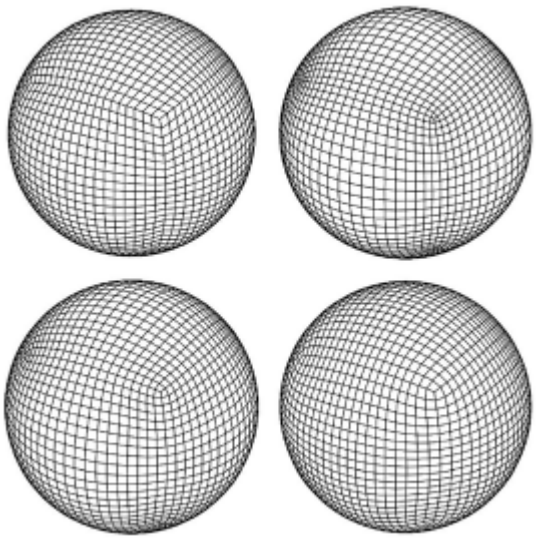

Figura 2. Esferas cubadas - Grade GEF. (a) Painel superior esquerdo: gnomonic (Sadourny, 1972); (b) Painel superior direito: conformal (Rancic ˇ et al., ´ 1996); (c) Painel inferior esquerdo: smooth (SM) (Purser and Rancic ˇ, 1998); ´ (d) Painel inferior direito: UJ (Rancic ˇ et al., 2017).

[3] Dragan Latinovic´ 1 , Sin Chan Chou1 , and Miodrag Rancic ˇ c´ 2. Seasonal range test run with Global Eta Framework. Adv. Sci. Res., 14, 247–251, 2017 https://doi.org/10.5194/asr-14-247-2017

[4] Mesinger, F., Chou, S. C., Gomes, J. L., Jovic, D., Bastos, P., Bustamante, J. F., Lazic, L., Lyra, A. A., Morelli, S., Ristic, I., and Veljovic, K.: An upgraded version of the Eta model, Meteorol. Atmos. Phys., 116, 63–79, https://doi.org/10.1007/s00703-012- 0182-z, 2012.

Códigos e artefatos

O modelo GEF foi adquirido diretamente do site de ftp do CPTEC no endereço abaixo definido: http://ftp1.cptec.inpe.br/pesquisa/grpeta/tempo/GEF_Shapiro_filter/GEF_v1.0.0_Shapiro_Filter.tar.gz

Configurações utilizadas nos experimentos

O modelo GEF não foi executado mas foi adquirido com as seguintes configurações:

Variável	Descrição	Valor
IM0	Dimensão de cada face	401
LM	Número de níveis verticais	50
NMIN	Número de faces da esfera	6
NSUB	Número de subdivisões por face	10
LSM	Número de níveis na saída	31
DT	Delta T	40
OUTPUTNUM	Número de outputs na saída	120
NDAYS	Dias de integração	30
SSTC	Fonte do SST (1=NCEP)	1
IDATIN	Data de inicialização	02/09/2019
Resolução	Resolução do modelo	25 km

Tabela 1 - Dados do modelo GEF

2.1.3 Shield (FV3)

O SHiELD [5] é um protótipo de modelo de atmosfera do Unified Forecast System (UFS) com capacidade de um sistema de previsão unificado em uma variedade de escalas de tempo e espaço, projetado para uma ampla gama de aplicações. Seu ND de esfera cúbica de volume finito (*Finite-Volume Cubed-Sphere Dynamical Core* - FV3) é importante especialmente sua dinâmica não hidrostática flexível, recursos de resolução variável e física integrada e sua estrutura do sistema de modelagem flexível (FMS). À esquerda da Figura 3 observa-se suas diversas formas de aplicação.

O FV3 origina-se do FV-Core (*Finite-Volume dynamical core*) desenvolvido na década de 90 pelo GSFC (Goddard Space Flight Center) da NASA. O FV3 é, atualmente, o ND utilizado no modelo acoplado

[5] https://www.gfdl.noaa.gov/shield/

GFS (Global Forecast System) do GFDL (Geophysical Fluid Dynamics Laboratory) capaz de suportar simulações atmosféricas hidrostática e não-hidrostática.[6]

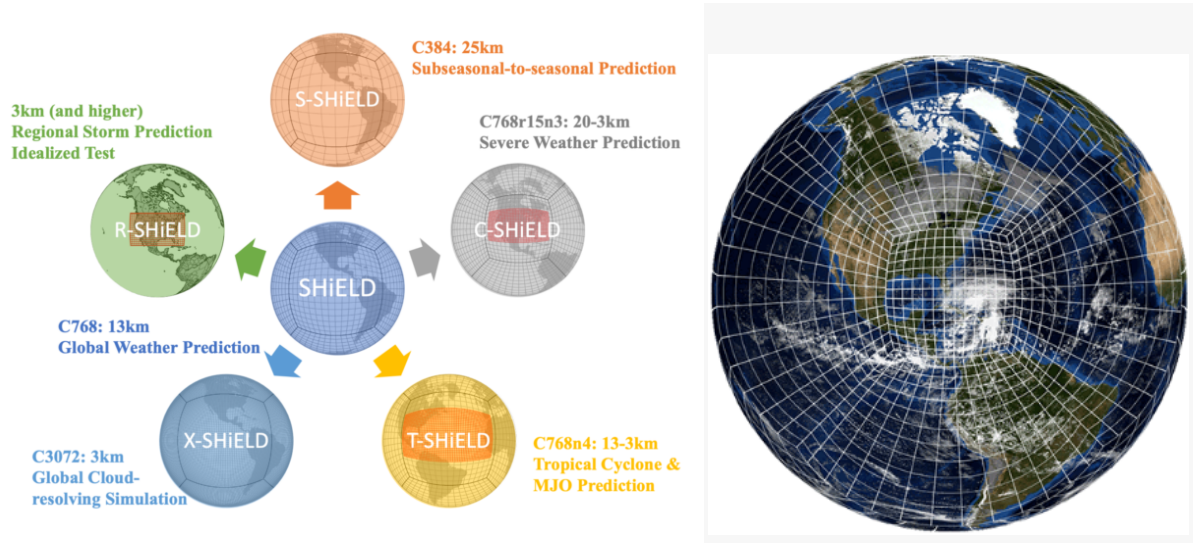

Figura 3: Aplicações do modelo Shield (esquerda).Ilustração da grade não estruturada do FV3 (direita).

Como a Figura 3 apresenta, à direita, o FV3 utiliza uma grade cúbica esférica flexível, que permite variar a resolução espacial em locais específicos de interesse do globo terrestre, enquanto que uma resolução mais grossa pode ser utilizada concorrentemente no restante do globo.

Códigos e artefatos

O pacote completo utilizado para a Previsão Global de Tempo, em sua resolução de 13km ((C768) foi obtido à partir de:

- https://zenodo.org/record/5090895/files/shield_container.tar
- https://zenodo.org/record/5090124/files/GFS_fix.zip
- https://zenodo.org/record/5090124/files/global_scalability.zip

O primeiro link contém um pacote com todo o código do Shield e FV3. O segundo link contém os arquivos fixos de condição de contorno do GFS utilizado em diversos experimentos. O terceiro link contém dados do experimento Shield para previsão de tempo Global.

Configurações utilizadas nos experimentos

Os principais parâmetros, fixos ou variáveis, usados no experimentos deste relatório, são configurados no arquivo input.nml [7] e estão descritos na tabela abaixo.

[6] gfdl.noaa.gov/fv3/
[7] https://www.gfdl.noaa.gov/wp-content/uploads/2017/09/fv3_namelist_Feb2017.pdf

Parâmetro	Descrição	Valor
layout	Layout de processadores em cada bloco. O número de processadores atribuídos a um domínio deve ser igual a layout(1)*layout(2)*tiles	Variado entre (3,4), (4,6), (6,8),(8,12),(12,16) e (16,24) para atingir o número de processadores desejado nos experimentos
npx	Número de cantos da grade na direção x em um bloco do domínio	769
npy	Número de cantos da grade na direção y em um bloco do domínio. Este valor deve ser idêntico ao npx em uma malha de esfera cúbica	769
npz	Número de níveis verticais. Cada escolha de npz vem com um conjunto predefinido de níveis de pressão sigma híbridos e topo do modelo	63
ntiles	Número de blocos no domínio. Para a esfera cubada, deve ser 6, um bloco para cada face da esfera cubada; para a maioria dos outros domínios (incluindo grades aninhadas), normalmente deve ser definido como 1	6
hours	Tempo total de integração. Nos experimentos, o tempo total de integração foi extrapolado para 24h	12

Tabela 2 - Configurações utilizadas nos experimentos

2.2 Recursos Computacionais

Os recursos computacionais disponíveis para a realização do estudo de Portabilidade e Performance, Escalabilidade e Eficiência dos modelos (PEE), dos modelos MPAS e SHiELD (FV3) foram:

- **Cluster Egeon (CPTEC)**
    - 33 nós
    - 2 CPU AMD EPYC 7H12 64-Core (128 núcleos), por nó
        - 503 GBytes de RAM (total)
- **Cluster Minerva (Dell)**

    - 64 nós
    - 2 CPU AMD EPYC 7713 64-Core (128 núcleos), por nó
        - 251 GBytes de RAM (total)
- **Cluster Rattler (Dell)** - nós computacionais com
    - 7 nós com GPU
    - 2 CPU AMD EPYC 7543 32-Core (64 núcleos), por nó
        - 503 GBytes de RAM (total)
    - 4 GPU NVIDIA A100-SXM4, por nó
        - 40 GBytes de RAM (cada GPU)

2.3. Avaliação da Qualidade de Software

A QS dos NDs foram analisadas em três etapas:

- Na primeira etapa, o código foi analisado quanto a Característica de Manutenibilidade, ou seja, foram analisados a facilidade de manutenção e os possíveis impactos (seção 2.3.1)

- A segunda etapa consistiu em avaliar a portabilidade dos modelos, onde os modelos foram devidamente instalados nos ambientes computacionais, juntamente com todas as bibliotecas, pacotes e dados necessários para sua compilação e execução (seção 2.3.2).

- Na terceira etapa da avaliação foram feitos testes de PEE onde os modelos foram executados utilizando paralelismo computacional, variando-se a quantidade de cores computacionais. O tempo de execução de cada rodada foi medido e analisado (seção 2.3.3).

2.3.1 Manutenibilidade

A Característica de Manutenibilidade diz respeito à facilidade de modificação e impactos sobre modificações. Esta seção exibe os resultados da avaliação da característica de Manutenibilidade dos NDs , cuja metodologia está descrita no APÊNDICE A.

Manutenibilidade: A capacidade (ou facilidade) do produto de software ser modificado, incluindo tanto as melhorias ou extensões de funcionalidade quanto às correções de defeitos, falhas ou erros. Suas Subcaracterísticas são:

- Analisabilidade: identifica a facilidade em se diagnosticar eventuais problemas e identificar as causas das deficiências ou falhas.
- Modificabilidade: caracteriza a facilidade com que o comportamento do software pode ser modificado.
- Reusabilidade: identifica a capacidade de se reutilizar o software por completo ou partes dele.

- Estabilidade: avalia a capacidade do software de evitar efeitos colaterais decorrentes de modificações introduzidas.
- Testabilidade: representa a capacidade de se testar o sistema modificado, tanto quanto as novas funcionalidades quanto as não afetadas diretamente pela modificação.

2.3.2 Portabilidade

A portabilidade deve ser avaliada em referência a 5 requisitos básicos:

- Arquitetura de máquinas
    - O modelo deve ser capaz de compilar no maior número de arquiteturas de máquinas distintas. Não deve apresentar disfunções ou perda de funcionalidade.
    - O modelo deve apresentar bons resultados nas diferentes arquiteturas, seja na funcionalidade quanto em uma performance mínima.
- Processadores
    - O modelo deve mostrar comportamento estável em diferentes tipos de processadores e não apresentar performance inesperada para esses processadores.
- Pacotes de software necessários para seu uso
    - O modelo deve ser capaz de funcionar com o mínimo de bibliotecas de software distintas sem que o resultado seja comprometido.
- Bibliotecas distintas MPI, OpenMP e OpenACC
    - O modelo deve ser capaz de responder bem, funcionar sem erros, ao uso de diferentes implementações do MPI, e de paralelismo com diretivas OpenMP e OpenACC.
- Compiladores
    - O modelo deve ser capaz de ser compilado por pelo menos 3 compiladores distintos: GNU, INTEL e PGI/NVIDIA

Para cada um dos modelos, as bibliotecas necessárias, ou seja, as suas dependências, são:

- **MPAS**
    - **NetCDF-C/Fortran:** NetCDF (network Common Data Form) é um conjunto de bibliotecas de software e de formatação de dados que independe de arquitetura de máquina e que permite a criação, acesso e compartilhamento de dados científicos
    - **Parallel-NetCDF:** PnetCDF (Parallel netCDF) é uma biblioteca de I/O paralelo de alta performance para acesso a dados em formato compatível com o NetCDF, especificamente os formatos CDF-1, 2, e 5.
    - **HDF5:** HDF5 é um modelo de dados, biblioteca e formato para arquivamento e gerência de armazenamento. Suporta uma grande variedade de tipos de dados

(datatypes) e foi projetado para I/O flexível e eficiente quando está trabalhando com grande volume de dados.
    - **ParallelIO:** Parallel IO libraries (PIO) é uma biblioteca de I/O de alto-nível para aplicações C e Fortran que precisam de NetCDF para um número elevado de processadores em sistemas de processamento de alto desempenho.

- **SHiELD (FV3)** - as mesmas bibliotecas requeridas pelo MPAS **(exceto ParallelIO)** mais as seguintes:
    - **Bacio:** Biblioteca BACIO realiza I/O binário para modelos do NCEP e processa registros de dados (data records) endereçados por bytes e formatado com N dimensões e faz a transformação entre dados/arquivos little-endian e big-endian.
    - **ESMF:** Earth System Modeling Framework (ESMF) é um software de infra-estrutura de alta performance e flexível para a construção de modelos acoplados de tempo, clima e aplicações de ciências da Terra. O ESMF define uma arquitetura para compor sistemas de modelagem complexos e acoplados e inclui estruturas de dados e utilidades para o desenvolvimento de modelos individuais.
    - **NEMSIO:** NOAA Environmental Modeling System I/O (NEMSIO). É uma biblioteca de funções básicas e fornece conjunto de dados para leitura e escrita para todas as aplicações NEMS(NOAA Environmental Modeling System).
    - **spLIB:** Biblioteca com diversas funções em subprogramas Fortran para transformada espectral. É parte das bibliotecas do projeto NCEPLIBS.
    - **w3nco:** Biblioteca com rotinas Fortran90 para codificar/decodificar formato GRIB edição 1 com as mudanças da NCO (NCEP Central Operations).

Para instalação dos softwares que são pré-requisito dos modelos, foi empregado o gerenciador de pacotes Spack (https://spack.io/). Dentre os pacotes disponíveis no Spack, incluem-se os modelos MPAS (**mpas-model**) e UFS (**ufs-weather-model**). Assim como o modelo SHiELD, o modelo UFS também utiliza o FV3 no núcleo da dinâmica, e possui praticamente os mesmos softwares pré-requisito.

Ao proceder a instalação de um pacote, tais como o **mpas-model** e o **ufs-weather-model**, consequentemente, o Spack providencia também a instalação de todas as suas dependências, ou seja, os respectivos softwares e bibliotecas pré-requisito. Assim então foi feito em todos os ambientes computacionais.

O Spack provê suporte para uso de diferentes compiladores para instalação dos pacotes. Portanto, para todas as bibliotecas, com o Spack foram também geradas instalações para cada um dos três compiladores utilizados nos testes de portabilidade: **GNU, INTEL e PGI/NVIDIA**.

Métrica	**Descrição**	**Impactos Positivos**	**Impactos Negativos**	**Ferramentas**	**Unidade de medida**

Arquitetura Dependência de máquina	Verifica diferentes arquiteturas que o modelo roda. Deve ser capaz de rodar pelo menos em 1 delas: Cluster Massivo Paralelo, GPU	- Adaptabilidade - Capacidade para ser Instalado		Rodada com avaliação de saídas. Avalia-se confiabilidade e portabilidade	Pontos por status funcional: +1.0
Processadores Capacidade de funcionar em diferentes processadores	Verifica se o core dinâmico é funcional e estável em processadores X86 (AMD ou INTEL)	- Adaptabilidade - Capacidade para ser Instalado		Rodada com avaliação de saídas. Avalia-se confiabilidade e portabilidade	Pontos por status funcional: +1.0
Pacotes Verifica a necessidade de uso de pacotes de software extra para uso com o modelo	Deve usar o mínimo de pacotes e bibliotecas adicionais para funcionar		-coexistência -capacidade de substituir	Verificação da exigência	Pontos negativos por pacote extra: -0.2
Bibliotecas de comunicação Verifica a capacidade do software funcionar com diversos tipos de biblioteca de comunicação	Deve ser capaz de funcionar com implementação de MPI, OpenMP e OpenACC	-coexistência -capacidade de substituir		Rodada Teste de funcionalidade	Pontos por biblioteca +1.0
Compiladores Verifica a capacidade de compilar em diversos compiladores	Avalia se modelo compila com compilador GNU, NVIDIA e INTEL	- Adaptabilidade - Capacidade para ser Instalado		Compilação e Rodada Teste de funcionalidade	Pontos por compilador +1.0

Tabela 3 - Critérios de pontuação de acordo com as métricas de avaliação de portabilidade

2.3.3 Performance, Escalabilidade e Eficiência (PEE)

Para avaliar os NDs utilizou-se os ambientes computacionais Minerva e Rattler. Outros ambientes computacionais foram considerados e testados. Entretanto foram descartados para finalidade das avaliações dos NDs. A razão do descarte é que os demais ambientes testados tem arquitetura e sistemas semelhantes aos dois sistemas já mencionados, Minerva e Rattler.

Cada ND foi submetido a cinco execuções de 12 horas de previsão cada, a média das cinco execuções é a medida do tempo final considerada para o cálculo das métricas de performance "*Speedup*" e "Eficiência". O *speedup* (Sp) (ou aceleração) é a relação entre o tempo gasto para executar uma tarefa com um único processo e o tempo gasto com "p" processos. Quanto mais próximo de "p" for o

speedup, melhor, pois se aproxima do *speedup* linear (p), considerado ideal. Neste caso, considerou-se como sequencial as execuções utilizando 128 cores equivalente à 1 nó do ambiente computacional. O *speedup* é definido como:

$$S_p = \frac{T_1}{T_p} \quad (1)$$

Onde:

S_p : *speedup* utilizando "p" processos
T_1 : tempo sequencial
T_p : tempo utilizando "p" processos

Outra métrica utilizada para avaliar a performance dos NDs foi a "Eficiência" (Ep), que é o *speedup* normalizado pelo número de processos, ou seja, quanto mais próximo de 1 (ou 100%) melhor, à semelhança do *speedup* linear se aproximando de "p". A eficiência é definida por:

$$E_p = \frac{S_p}{p} \quad (2)$$

Onde:

E_p : eficiência utilizando "p" processos
S_p : *speedup* utilizando "p" processos
p : número de processos utilizados.

Utilizou-se o gerenciador de pacotes Spack[8] especialmente desenvolvido para facilitar a instalação de softwares científicos em ambientes de supercomputadores. Os códigos do modelo MPAS, por exemplo, foram obtidos e instalados, primeiramente na sua versão 7.[redacted], através do Spack. A compilação do sistema SHiELD foi facilitada pela utilização do spack, pois sua base de bibliotecas pôde ser gerada previamente com a compilação do UFS usando o Spack.

3. Resultados

3.1 Manutenibilidade

Foram avaliados os modelos FV3, GEF e MPAS. Os códigos dinâmicos avaliados estão disponíveis em https://github.com/monanadmin/monan/tree/main/codigos_originais . Nesta

[8] spack.io/

primeira avaliação automatizada, seguindo os procedimentos documentados, o modelo GEF obteve um melhor desempenho em algumas métricas como documentação total, documentação de rotinas, tamanho médio dos módulos, *Fan-In*, *Fan-out* e aninhamento média de rotinas, como pode ser visto na Tabela 4.

	Avaliação			Pontuação		
Métricas de Manutenibilidade	**FV3**	**GEF**	**MPAS**	**FV3**	**GEF**	**MPAS**
Comprimento de código	42.014,00	107.221,00	264.016,00	3	1	1
Complexidade Ciclomática Média (McCabe)	16,82	18,08	12,62	1	1	3
Documentação total	0,24	**2,09**	0,25	1	3	1
Documentação de rotinas	0,36	**0,77**	0,55	1	3	1
Documentação de arquivos	0,40	**0,39**	0,32	3	1	1
Documentação de estruturas de controle	0,02	**0,31**	0,11	1	3	1
Tamanho médio das rotinas	75,95	158,76	97,85	3	1	1
Tamanho médio dos módulos	1.623,31	**41,26**	916,98	1	3	1
Tamanho médio do nome das variáveis	5,24	4,09	9,45	1	1	3
Razão de *only* em *uses*	92,77	3,56	47,22	3	1	1
Razão de "*goto*" e "*continue*" por laço	2,50	13,76	0,00	1	1	3
Razão de "*exit*" e "*cycle*" por laço	0,63	0,25	2,17	1	1	3
Razão do uso de "*implicit*"	4,43	**8,19**	0,32	1	3	1
Total de "*equivalence*" ou "*common*"	0,00	44,00	0,00	3	1	3
Profundidade média de laços	7,81	13,56	13,26	3	1	1
Aninhamento médio de laços	0,30	0,38	0,19	1	1	3
Fan-in (mesma rotina sendo chamada)	4,31	**86,00**	4,66	1	3	1
Fan-out (rotinas chamadas por rotina)	8,35	**3,00**	16,50	1	3	1
Aninhamento médio de rotinas	1,89	**1,21**	1,93	1	3	1
TOTAIS				**31**	**35**	**31**

Tabela 4: Análise das métricas e pontuação dos modelos FV3, GEF e MPAS.

No entanto, ao aplicar uma análise qualitativa do código, verificou-se alguns pontos, destacados em negrito na Tabela 4, que foram beneficiados erroneamente, devido a:

- A documentação gerada automaticamente para todo o código não é informativa, isto é, foram gerados os cabeçalhos automáticos mas não foram inseridas informações explicativas sobre o conteúdo dos códigos. Com isso, houve benefício nos itens relacionados à documentação.
- Do total de módulos, 75 deles não contêm rotinas, os quais colaboraram para gerar uma maior pontuação nos itens “Tamanho médio dos módulos” e “Razão do uso de *‘implicit’* ”.
- Rotinas criadas dentro de arquivos sem módulos, o que é desencorajado pelos padrões de codificação, geraram imprevistos de cálculo na ferramenta FortranMakeUtils, aumentando significativamente as métricas *Fan-in* e *Fan-out* e diminuindo a métrica “Aninhamento médio de rotinas”

Levando em consideração os três pontos citados acima, os critérios destacados em negrito na Tabela 4 foram removidos desta avaliação com os três modelos. Seguindo os critérios de

pontuação definidos no DTN4 (Documento Técnico Normativo 4), foram obtidos os resultados compilados na Tabela 5. Os melhores resultados das métricas receberam três pontos, e os piores um ponto. Esta pontuação foi associada às colunas com os nomes dos modelos e às colunas com subcaracterísticas associadas às métricas pontuadas.

Métricas de Manutenibilidade	**FV3**	**GEF**	**MPAS**	**FV3**	**GEF**	**MPAS**
Comprimento de código	42.014	107.221	264.016	3	1	1
Complexidade Ciclomática Média (McCabe)	16,82	18,08	12,62	1	1	3
Tamanho médio das rotinas	75,95	158,76	97,85	3	1	1
Tamanho médio do nome das variáveis	5,24	4,09	9,45	1	1	3
Razão de only em uses	92,77	3,56	47,22	3	1	1
Razão de "*goto*" e "*continue*" por laço	2,50	13,76	0,00	1	1	3
Razão de "*exit*" e "*cycle*" por laço	0,63	0,25	2,17	1	1	3
Total de "*equivalence*" ou "*common*"	0,00	44,00	0,00	3	1	3
Profundidade média de laços	7,81	13,56	13,26	3	1	1
Aninhamento médio de laços	0,30	0,38	0,19	1	1	3
TOTAIS				**20**	**10**	**22**

Analis.	**Modific.**	**Reusab.**	**Estab.**	**Testab.**	**Analis.**	**Modific.**	**Reusab.**	**Estab.**	**Testab.**	**Analis.**	**Modific.**	**Reusab.**	**Estab.**	**Testab.**
FV3					**GEF**					**MPAS**				
3	3	3	3	3	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	3	3	3	3	3
3	3	3	3	3	1	1	1	1	1	1	1	1	1	1
1	1			1	1	1			1	3	3			3
3	3		3		1	1		1		1	1		1	
1	1		1		1	1		1		3	3		3	
1	1		1		1	1		1		3	3		3	
	3		3			1		1			3		3	
3	3	3	3	3	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	3	3	3	3	3
17	**20**	**11**	**19**	**12**	**9**	**10**	**5**	**9**	**6**	**19**	**22**	**9**	**19**	**12**

Tabela 5: Análise das métricas e pontuação dos modelos FV3, GEF e MPAS sem os critérios relativos a documentação, tamanho médio dos módulos, “implicit”, “Fan-in” e “Fan-out”.

As maiores pontuações, totalizadas na última linha, representam as características que foram mais consideradas. A Figura 4 ilustra graficamente os resultados dos totais apresentados na Tabela 5.

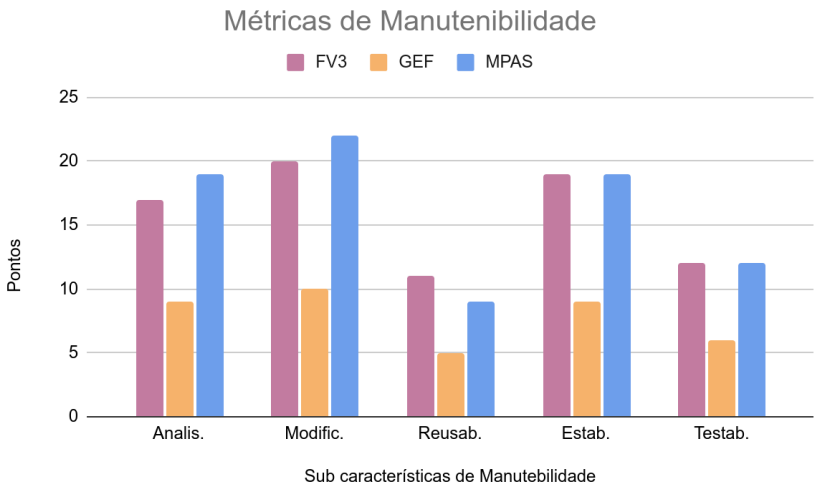

Figura 4 - Métricas das Subcaracterísticas da Manutenibilidade com os modelos FV3, GEF e MPAS.

Ao analisar a pontuação do modelo GEF, utilizando todas as métricas possíveis de comparação para os três modelos, verificou-se que todas suas pontuações em métricas obtiveram valor mínimo (1 ponto). Por esse motivo, o modelo foi considerado insatisfatório com relação ao desempenho em Manutenibilidade e uma nova avaliação foi feita somente com os modelos FV3 e MPAS, incluindo-se todas as métricas possíveis, de onde se obtiveram os resultados apresentados na Tabela 6.

	Pontuação		Analis.	Modific.	Reusab.	Estab.	Testab.	Analis.	Modific.	Reusab.	Estab.	Testab.
Métricas de Manutenibilidade	**FV3**	**MPAS**	**FV3**					**MPAS**				
Comprimento de código	3	1	3	3	3	3	3	1	1	1	1	1
Complexidade Ciclomática Média (McCabe)	1	3	1	1	1	1	1	3	3	3	3	3
Documentação total	1	3	1	1				3	3			
Documentação de arquivos	3	1	3	3				1	1			
Documentação de estruturas de controle	1	3	1	1				3	3			
Tamanho médio das rotinas	3	1	3	3	3	3	3	1	1	1	1	1
Tamanho médio dos módulos	1	3	1	1	1	1	1	3	3	3	3	3
Tamanho médio do nome das variáveis	1	3	1	1			1	3	3			3
Razão de *only* em *uses*	3	1	3	3		3		1	1		1	
Razão de "*goto*" e "*continue*" por laço	1	3	1	1		1		3	3		3	
Razão de "*exit*" e "*cycle*" por laço	1	3	1	1		1		3	3		3	
Razão do uso de "*implicit*"	3	1	3	3		3		3	3		3	
Total de "*equivalence*" ou "*common*"	3	3		3		3			3		3	
Profundidade média de laços	3	1	3	3	3	3	3	1	1	1	1	1
Aninhamento médio de laços	1	3	1	1	1	1	1	3	3	3	3	3
Fan-in (mesma rotina chamada)	1	3	1	1	1	3	1	3	3	3	1	3

Fan-out (rotinas chamadas por rotina)	3	1		3	3	3	3		1	1	1	1
Aninhamento médio de rotinas	3	1		3	3	3	3		1	1	1	1
TOTAIS	**37**	**41**	**28**	**37**	**20**	**32**	**21**	**38**	**43**	**20**	**28**	**23**

Tabela 6: Análise do conjunto total de métricas e pontuação dos modelos considerados satisfatórios

Ao incluir critérios removidos na avaliação com o GEF, a Analisabilidade passa a ser mais importante que a Estabilidade, como se pode observar nos totais da Tabela 4. O modelo FV3 totalizou 28 + 37 + 20 + 32 + 21 = 138 pontos e o MPAS 152 pontos na somatória geral da Manutenibilidade.

A Figura 5 ilustra graficamente os totais da Tabela 6, onde se verifica uma superioridade do modelo MPAS nas Características de Analisabilidade, Modificabilidade e Testabilidade, enquanto o FV3 supera o MPAS na característica de Estabilidade. Os modelos empataram em termos de Reusabilidade.

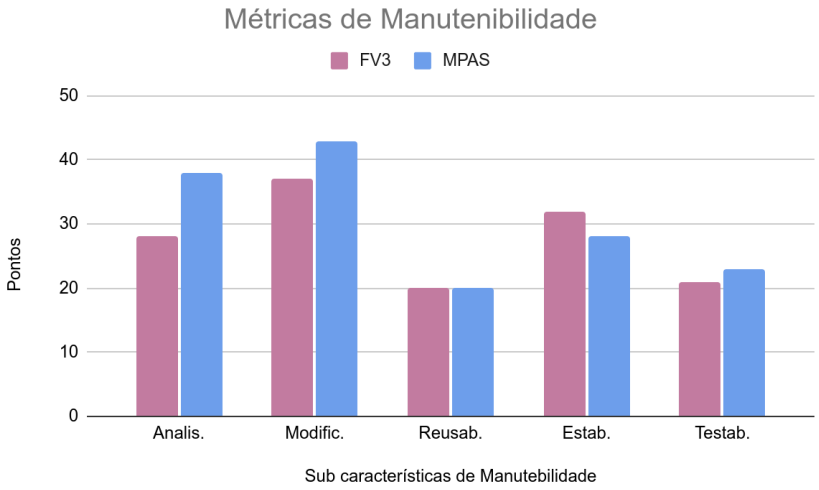

Figura 5 - Métricas das Subcaracterísticas da Manutenibilidade dos modelos considerados satisfatórios.

3.2 Portabilidade

Nesta seção é avaliada a portabilidade dos códigos referentes aos códigos do núcleo da dinâmica do MPAS e do FV3 (SHiELD/UFS), nos ambientes computacionais disponíveis.

3.2.1 MPAS

A portabilidade do MPAS foi avaliada para todos os requisitos básicos descritos na seção 2.3.2, seguindo critérios definidos na Tabela 3. Na Tabela 7 está a pontuação obtida para cada uma das

métricas dos requisitos para o MPAS, onde são indicadas as tarefas (*issues*) no projeto do repositório GitHub do MONAN[9], que validam os modelos para as métricas **Arquitetura** e **Compiladores**.

Métrica	MPAS	Pontos
Arquitetura Dependência de máquina	- Funciona na Egeon e na Minerva: issues #44 e #63 - Funciona em GPU no cluster Rattler: issue #53	2.0
Processadores Capacidade de funcionar em diferentes processadores	- Funciona em X86 (AMD ou INTEL)	1.0
Pacotes Verifica a necessidade de uso de pacotes de software extra para uso com o modelo	- HDF5 - netCDF - Parallel-NetCDF - ParallelIO	-0.8
Bibliotecas de comunicação Verifica a capacidade do software funcionar com diversos tipos de biblioteca de comunicação	- MPI: OpenMPI e Intel-MPI - OpenMP - OpenACC	3.0
Compiladores Verifica a capacidade de compilar em diversos compiladores	- GNU: gfortran, gcc, g++: issue #56 - INTEL: ifort, icc, icpc: issue #61 - PGI/NVIDIA: pgf90, pgcc, pg++: issue #45	3.0
Pontuação Geral		8.2

Tabela 7 - Pontuação do modelo MPAS de acordo com as métricas de avaliação de portabilidade

3.2.2 FV3

A portabilidade do FV3 foi avaliada para todos os requisitos básicos descritos na seção 2.3.2, seguindo critérios de pontuação definidos na Tabela 3. Na Tabela 8 está a pontuação obtida para cada uma das métricas dos requisitos para o FV3, onde são indicadas as tarefas (*issues*) no projeto do repositório GitHub do MONAN[10], que validam os modelos para as métricas **Arquitetura** e **Compiladores**.

9

10 MONAN - Model for Ocean-laNd-Atmosphere PredictioN. Repositório Git. Disponível em: <https://github.com/monanadmin/monan>. Acesso em: 25 de agosto de 2022

Para os testes de portabilidade foi utilizado o modelo UFS, com resolução C96 (~100km), tendo a finalidade de verificar se o código do FV3 compilava e executava corretamente nas arquiteturas disponíveis. O pacote utilizado para a previsão global com 24h de integração nesta resolução foi obtido à partir de https://ftp.emc.ncep.noaa.gov/EIB/UFS/simple-test-case.tar.gz, onde a configuração dos parâmetros é dada a seguir:

- layout = 1,1
- io_layout = 1,1
- npx = 97
- npy = 97
- ntiles = 6
- npz = 64

Métrica	**FV3**	**Pontos**
Arquitetura Dependência de máquina	- Funciona na Egeon e na Minerva: issues #38 e #60 - Não funciona em GPU no cluster Rattler	1.0
Processadores Capacidade de funcionar em diferentes processadores	- Funciona em X86 (AMD ou INTEL)	1.0
Pacotes Verifica a necessidade de uso de pacotes de software extra para uso com o modelo	- HDF5 - netCDF - Parallel-NetCDF - Bacio - ESMF - NEMSIO - spLIB - w3nco	-1.6
Bibliotecas de comunicação Verifica a capacidade do software funcionar com diversos tipos de biblioteca de comunicação	- MPI - OpenMP - Não dá suporte a OpenACC	2.0
Compiladores Verifica a capacidade de compilar em diversos compiladores	- GNU: gfortran, gcc, g++: issue #37 - INTEL: ifort, icc, icpc: issue #41 - PGI/NVIDIA: pgf90, pgcc, pg++: não validado (#42)	2.0
Pontuação Geral		4.4

Tabela 8 - Pontuação do FV3 de acordo com as métricas de avaliação de portabilidade

3.3 PEE (Performance, Escalabilidade e Eficiência)

Nesta seção são mostrados e avaliados os resultados de desempenho paralelo do núcleo da dinâmica dos modelos MPAS e SHiELD/FV3. Os testes de desempenho foram conduzidos sobre configurações específicas de rodada, detalhadas a seguir para cada um dos modelos.

3.3.1 MPAS

Para a avaliação de desempenho do núcleo dinâmico do modelo MPAS, foi escolhida uma das parametrizações do *benchmark* mantido pelos desenvolvedores do modelo, disponibilizado no endereço https://www2.mmm.ucar.edu/projects/mpas/benchmark/. São para três versões do modelo (v5.2, v6.x, v7.0). Contém os dados de condição inicial para diferentes resoluções de malha, onde a mais grossa é de 120km e a mais fina de 10km. Os arquivos de *namelist* com a parametrização espacial e física de execução, são também disponibilizados. Para os testes de desempenho mostrados neste relatório, foi escolhido o *benchmark* relativo a malha com resolução espacial de 15 km. A motivação para escolha de malhas mais grossas se deve ao fato da quantidade de memória disponível nos nós computacionais das máquinas usadas no teste.

O tempo de execução para 12 horas de integração do modelo é mostrado na Tabela 9, em rodada realizada com 2 nós do cluster Minerva, usando 64 núcleos por nó. É também mostrado o tempo de execução em diferentes sub-rotinas do MPAS, sendo que na cor azul estão destacadas aquelas relativas ao **núcleo da dinâmica**. O tempo total obtido foi de 5930.33s, e o tempo somado das sub-rotinas do núcleo da dinâmica foi de 3019.63s. Extrapolando-se para 24h de integração, este tempo de processamento é o dobro, ou seja, 6039.26s.

timer_name	total
1 total time	5920.33105
2 initialize	32.38702
2 time integration	5886.15186
3 physics driver	2033.52197
4 calc_cldfraction	0.58155
4 RRTMG_sw	756.18988
4 RRTMG_lw	429.19873
4 Monin-Obukhov	4.56684
4 Noah	12.75321
4 YSU	168.60727
4 GWDO_YSU	36.13308
4 New_Tiedtke	512.27197
3 atm_rk_integration_setup	46.28673
3 atm_compute_moist_coefficients	23.35006
3 physics_get_tend	344.41339
3 atm_compute_vert_imp_coefs	31.42596
3 atm_compute_dyn_tend	934.06744
3 small_step_prep	111.46519
3 atm_advance_acoustic_step	311.87317
3 atm_divergence_damping_3d	81.90127

3	atm_recover_large_step_variables	348.64746
3	atm_compute_solve_diagnostics	493.36813
3	atm_rk_dynamics_substep_finish	102.01469
3	atm_advance_scalars	196.52936
3	atm_advance_scalars_mono	420.60504
3	microphysics	438.77209
4	WSM6	389.99307

Tabela 9 - Tempos de execução retornados pela saída do modelo MPAS, para rodada com 12 horas de integração do benchmark de 15km. Destacados em azul as sub-rotinas relativas ao núcleo da dinâmica.

A Figura 6 mostra os tempos de processamento paralelo. A figura apresenta os resultados obtidos em duas simulações distintas. Uma rodada submetida usando 128 núcleos (*cores*) por nó e outra utilizando 64 núcleos. Observa-se que as rodadas que usam 64 núcleos por nó tem melhor desempenho (*performance*). Não foram verificados os motivos que levam a essa melhora de desempenho, contudo é esperado que tal variação seja resultado do uso de menos memória em cada nó. A investigação para esse comportamento será realizada em experimentos futuros.

Nas curvas de speedup (ganho de desempenho), teórico x medido, é observado que o modelo apresentou um comportamento super linear. Esse comportamento é determinado quando $S_p > p$ (Veja equação 1). Comportamentos de super-escalabilidade (super linear) são sintomas patológicos. Em geral estão associados à topologia de memória, formas de acesso à memória (cache, page fault, etc), uso concomitante de OpenMP e MPI ou outras causas. A investigação para esse comportamento deverá ser realizada em experimentos futuros.

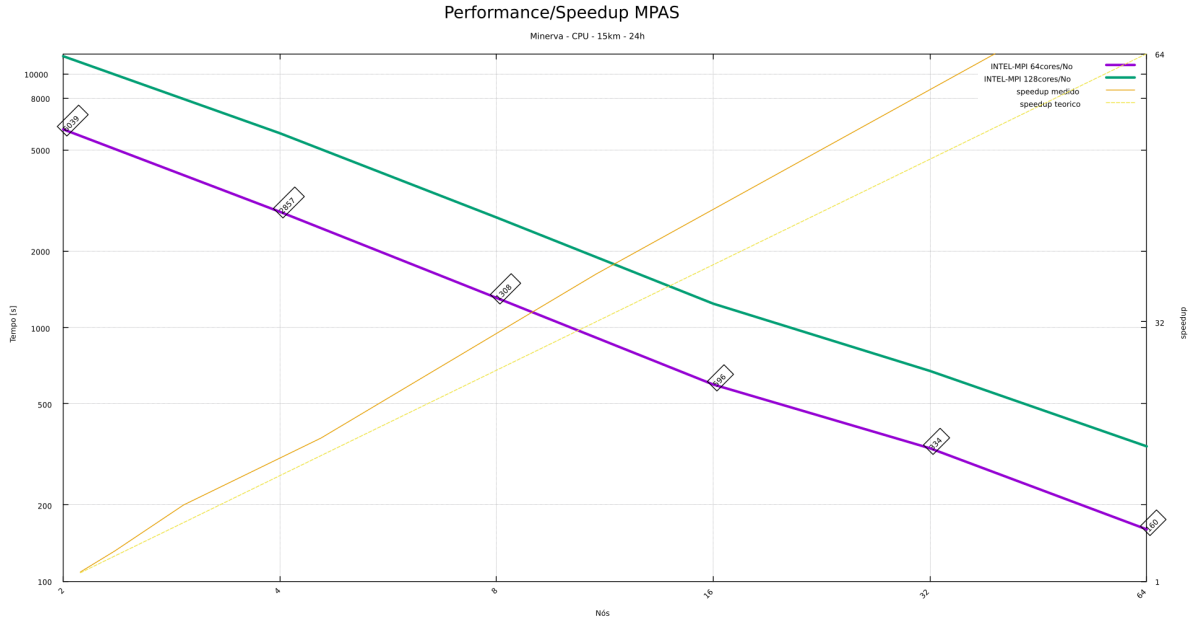

Figura 6 - Tempo de processamento paralelo (MPI) e speedup (Ganho de desempenho) do núcleo da dinâmica do MPAS no cluster Minerva

Na figura 7 vemos a eficiência do paralelismo (2). Devido ao comportamento super linear a eficiência medida é maior que a eficiência máxima esperada, isto é, maior que 100%. Isso fere a lei de Amdahl

que trata da relação entre a fração paralelizável e a fração escalar de um código. Contudo os casos de super-escalabilidade levam a esse comportamento e devem ser investigados.

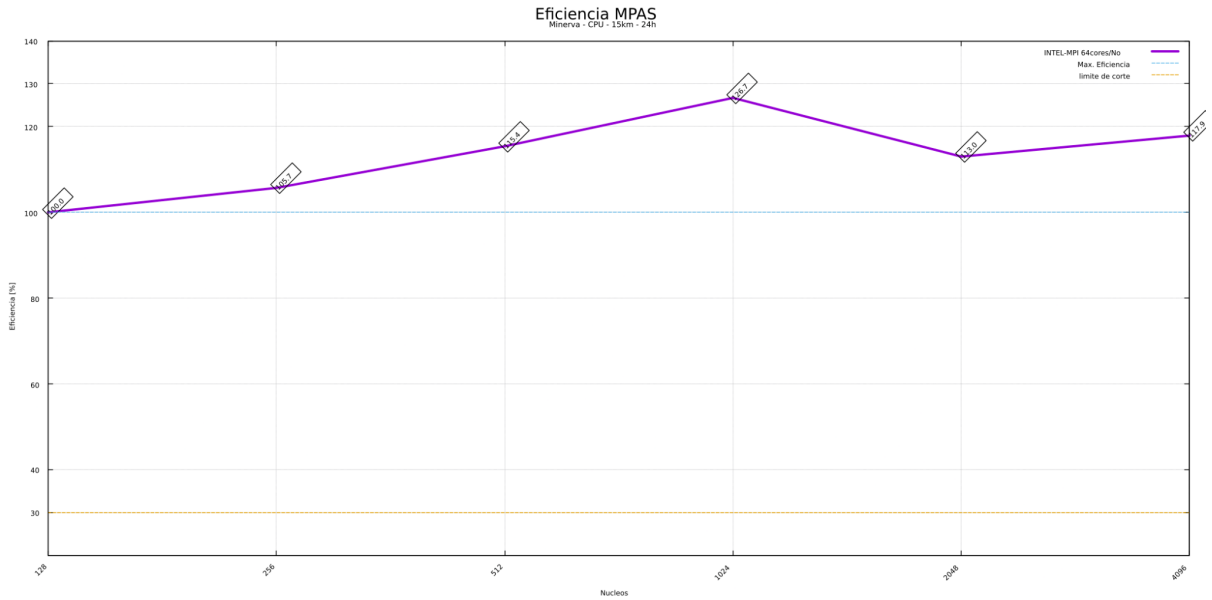

Figura 7 - Eficiência do processamento paralelo (MPI) do núcleo da dinâmica do MPAS no cluster Minerva

O modelo MPAS foi submetido para rodar na máquina Rattler. O modelo está preparado para rodar com GPU utilizando diretivas *OpenACC - Padrão de Programação para Computação Paralela* [11]. Essa máquina dispõe apenas de 7 nós (448 núcleos) na configuração adequada para a rodada do modelo. Portanto o experimento foi realizado até esse limite. A figura 8 mostra a performance e speedup do núcleo dinâmico do modelo nesse sistema. Observa-se um comportamento excelente de performance e speedup.

[11] https://www.openacc.org/

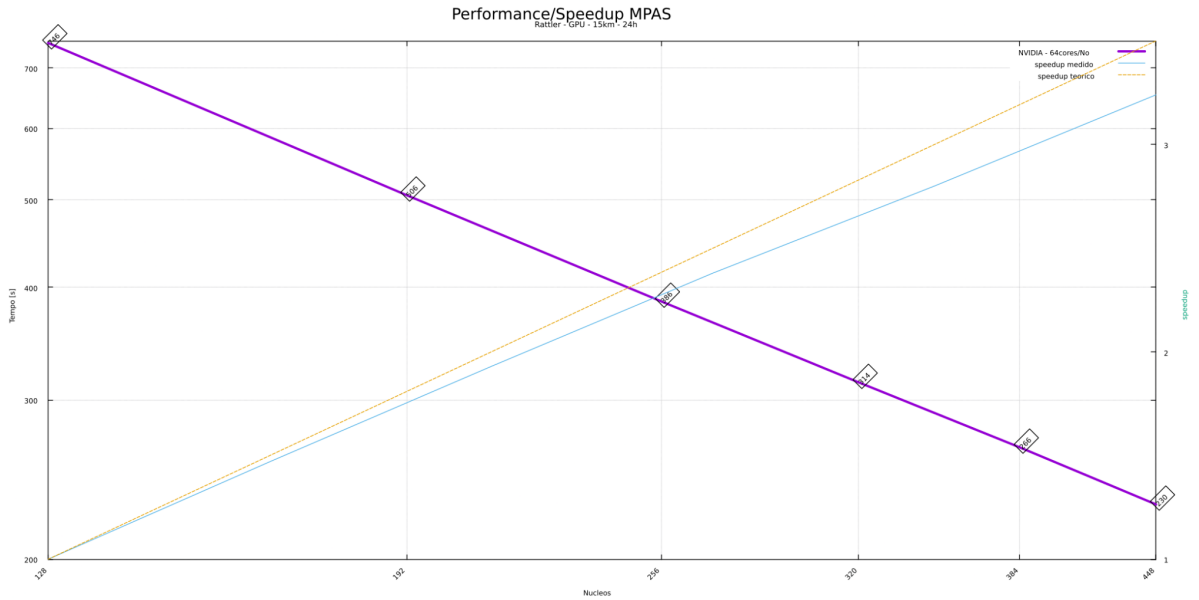

Figura 8 - Tempo de processamento paralelo (MPI) e Speedup do núcleo da dinâmica do MPAS no cluster Rattler usando 4 GPUs NVIDIA A100 por nó.

Na figura 9 é apresentado o resultado da eficiência do modelo MPAS no sistema Rattler. Foi determinada uma equação para representar a tendência da eficiência nos 7 nós disponíveis e aplicada a mesma equação para determinar a eficiência para número de núcleos acima de 448. Essa simulação será útil na comparação dos núcleos dinâmicos em diferentes sistemas computacionais. A aplicação da equação e os resultados são teóricos.

Em geral, o comportamento real dos modelos em sistemas paralelos não acompanham curvas com estimação baseadas em equações de tendência. Mas por falta de sistemas maiores isso se tornou necessário. Aplicando a equação observa-se que a eficiência cai abaixo de 30% para 4096 núcleos computacionais. Esse limite de 30% foi adotado como limite mínimo aceitável.

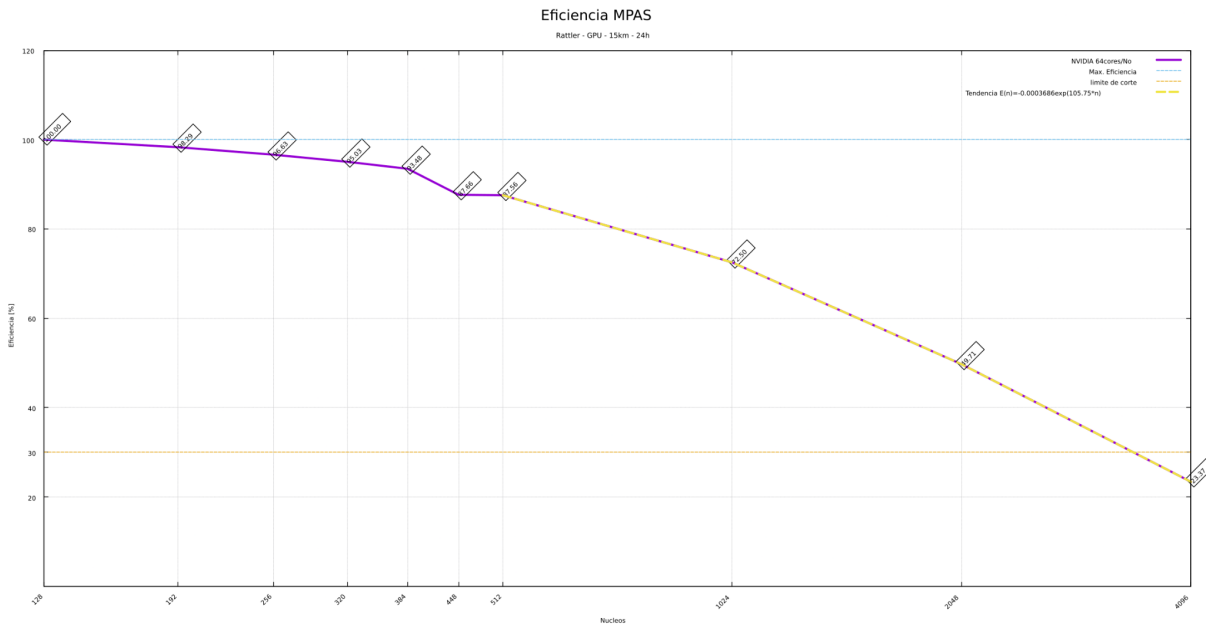

Figura 9 - Eficiência do processamento paralelo (MPI) do núcleo da dinâmica do MPAS no cluster Rattler usando 4 GPUs NVIDIA A100 por nó.

3.3.2 FV3

Esta seção exibe os resultados de Performance, Escalabilidade e Eficiência do FV3 através da execução do modelo FV3-Shield no Cluster Minerva, onde foi possível avaliar os requisitos de até 64 nós de CPUs.

Até o momento de elaboração destes resultados, não foi possível concluir a compilação do Shield com GPU, que não está preparado para o compilador da NVidia. Futuros trabalhos poderão apresentar estes resultados.

As Figuras 10 e 11 apresentam a performance, *Speedup* e Eficiência do Shield em duas linhas de execução: Utilizando 96 e 72 processadores por core. Devido a grade cúbica do modelo, o número de núcleos para a rodada do Shield deve ser compatível com sua configuração de LayoutX, LayoutY e ntiles=6. Isso força a escolha de um número de núcleos múltiplos destas configurações para cada rodada. Por esse motivo, foi estressado o máximo de processadores por nó (96) e também foram feitos testes com 72 processadores, número mais próximo para comparação com o número de processadores utilizado nos testes do MPAS (64).

Os resultados dos tempos do FV3 foram obtidos a partir do relatório de execução do Shield com 12h de integração. Abaixo, observa-se o tempo total e o tempo específico do ND destacado em negrito, onde foram utilizados 3072 processadores distribuídos em 32 nós (96 cores por nó):

Tabulating mpp_clock statistics across 3072 PEs...

	hits	tmin	tmax	tavg	tstd	tfrac	grain	pemin	pemax
Total runtime	1	316.532990	316.537048	316.534790	0.000586	1.000	0	0	3071
Initialization	1	46.709026	47.009949	46.959171	0.110229	0.148	0	0	3071
FV dy-core	576	95.576279	137.230118	**111.872803**	10.009806	0.353	11	0	3071
FV subgrid_z	288	0.122203	0.368742	0.215803	0.055221	0.001	11	0	3071
FV Diag	288	1.965000	2.972553	2.283237	0.156009	0.007	11	0	3071
GFS Step Setup	288	66.412865	100.922356	92.205933	9.730636	0.291	1	0	3071
GFS Radiation	288	2.741683	15.935287	8.426137	2.717996	0.027	1	0	3071
GFS Physics	576	15.045197	20.578415	16.700846	0.763495	0.053	1	0	3071
Dynamics get state	288	4.008958	4.928978	4.202710	0.081553	0.013	1	0	3071
Dynamics update state	288	9.644940	51.693634	26.468935	10.011416	0.084	1	0	3071
FV3 Dycore	576	99.676285	141.782654	116.474121	10.018317	0.368	1	0	3071
Main loop	1	238.051605	238.054733	238.052979	0.000446	0.752	0	0	3071
Termination	1	12.268960	12.269122	12.269039	0.000025	0.039	0	0	3071
MPP_STACK high water mark=	0								

Tabela 10 - Especificação dos tempos da rodada FV3

No caso, o FV3 levou o tempo de 111.87s no total para a integração de 12h. Extrapolando-se para 24h, o tempo total foi de 223,74s, que pode ser observado na Figura 10. No gráfico os valores foram arredondados.

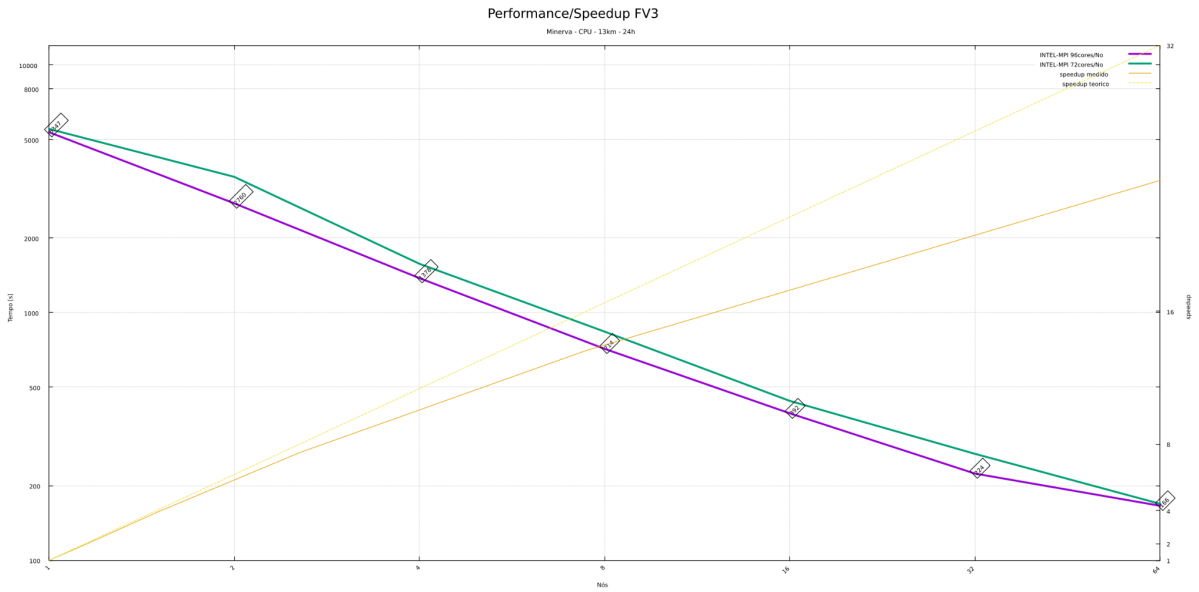

Figura 10 - Desempenho e speedup do core dinâmico do modelo FV3 na Minerva

A eficiência do modelo usando 96 núcleos é mostrada na figura 11. Observa-se que o ND apresenta boa eficiência até o limite de núcleos testados, estando sempre acima do limite de 30%.

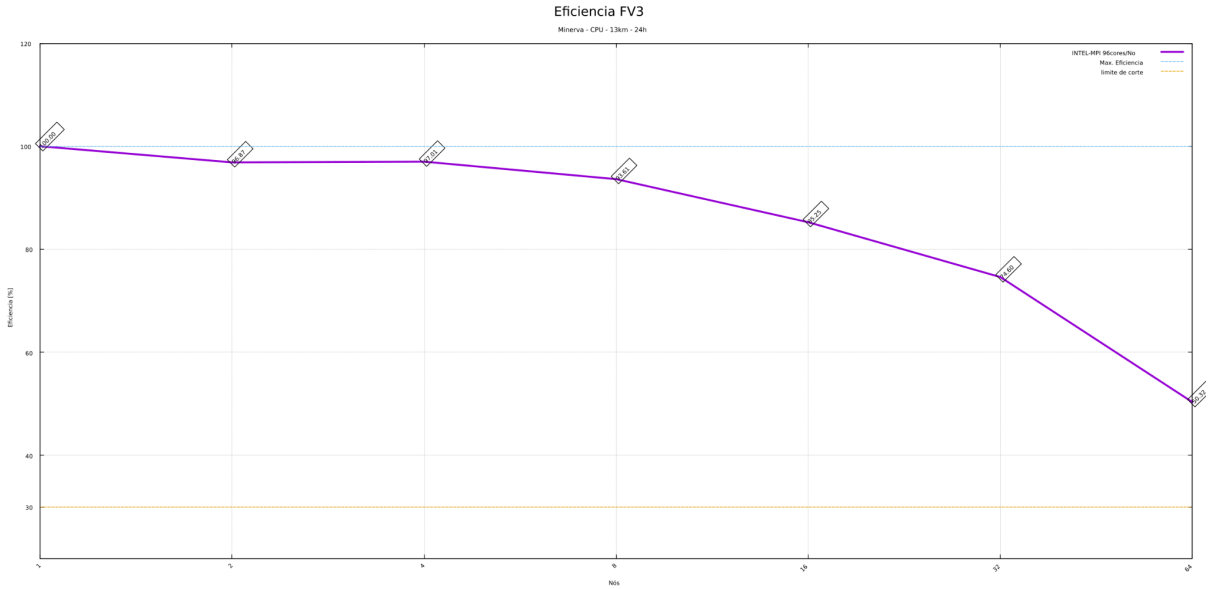

Figura 11 - Eficiência do core dinâmico do modelo FV3 na Minerva

4. Conclusão e Discussões

4.1 Manutenibilidade

Ao considerar a avaliação do conjunto total de métricas para os modelos FV3, GEF e MPAS, os resultados indicaram o modelo MPAS com o maior somatório de pontuações de Subcaracterísticas de Manutenibilidade. Tendo em vista que o modelo GEF foi beneficiado em alguns critérios indevidamente, os cálculos foram refeitos desconsiderando estas métricas, e este foi considerado insatisfatório, por não ter nenhuma pontuação máxima (3 pontos) em todas as sub características avaliadas. Em seguida, o GEF foi removido da análise e foram incluídas as métricas desconsideradas. Nesta segunda avaliação, o modelo FV3 totalizou 138 pontos e o MPAS 152. Os resultados indicaram que o MPAS obteve o melhor desempenho na maioria das características em ambas as situações.

Ao comparar os modelos FV3 e MPAS, considerando todas as métricas possíveis, o FV3 recebeu pontuação superior em Estabilidade, enquanto que o MAPS superou o FV3 em Analisabilidade, Modificabilidade e Testabilidade. Em termos de Reusabilidade a performance foi idêntica. Reforça-se que a Estabilidade, no contexto de Manutenibilidade, avalia a capacidade do software de evitar efeitos colaterais decorrentes de modificações introduzidas. **Não se refere à estabilidade de execução** dos modelos.

Destaca-se que, em termos da Manutenibilidade, os resultados aqui encontrados não são suficientes para eliminar o modelo FV3 da lista de modelos elegíveis para compor o ND do modelo MONAN. Com relação ao GEF, devido às pontuações consideradas insatisfatórias (1 ponto em todas Subcaracterísticas), considera-se que este modelo não é elegível para compor o ND do modelo MONAN se comparado com as duas outras opções.

4.2 Portabilidade

Os testes de portabilidade foram realizados apenas nos modelos FV3 e MPAS. Os dois modelos foram testados nas máquinas disponíveis, com os compiladores e bibliotecas de comunicação possíveis e, em geral, apresentaram bom comportamento. Não foi possível obter o modelo FV3 com portabilidade para uso em GPUs. Os critérios de pontuação foram aplicados garantindo que a instalação e compilação sejam feitas de forma mais simples e que exijam pouco dos usuários.

O resultado da pontuação aplicada mostra que o modelo MPAS teve nota 8.4 e o modelo FV3 teve nota 4.4.

O FV3 tem 4 pontos desfavoráveis quando comparado com o MPAS:

a) Falta de diretivas OpenACC (Não portado);
b) a não possibilidade de execução usando GPU (reflexo do item anterior);
c) exigir a instalação de uma grande quantidade de bibliotecas adicionais;
d) não ter sido possível executá-lo com compilação NVIDIA/Portland.

4.3 Performance, Escalabilidade e Eficiência

Os dois núcleos dinâmicos testados tem bom comportamento quando avaliados nos quesitos de desempenho, escalabilidade e eficiência. Contudo a análise não pode ser completa por alguns

motivos. Entre eles houve a indisponibilidade de máquinas com GPU nos mesmos patamares que a máquina com CPU. Os testes foram então realizados com os hardwares disponíveis.

Comparação por número de nós computacionais

A comparação de performance por número de nós permite que se entenda o comportamento e desempenho do modelo quando no uso de sistemas, ainda que os Jobs não ocupem um nó completo. Essa comparação é mais realista pois infere diretamente um custo financeiro que é dependente do total de nós de um sistema.

O gráfico apresentado nas figuras 12 e 13 mostram o comportamento comparado entre os modelos. Conforme discutido e apresentado na figura 8, o modelo MPAS em GPU foi rodado com no máximo 7 nós. Para estimar o tempo do modelo para mais nós utilizou-se o speedup ideal e aplicou-se a tendência de eficiência mostrada na figura 9.

Além disso foi necessário inferir um valor do tempo para o MPAS fazendo-se uma aproximação pois o FV3 foi rodado com resolução de 13 km e o MPAS com resolução de 15 km. Ademais há uma diferença de número de níveis. O FV3 usou 63 níveis e o MPAS 56. Para obter o tempo estimado da rodada do MPAS com resoluções compatíveis ao FV3 usou-se obter uma razão $P_{fv3/mpas}$ entre os dois modelos. A razão foi obtida da seguinte forma:

$$A_t = 4\pi R_t^2$$

onde:
A_t - Área da superfície da Terra
R_t - Raio da Terra (6378 km)

$$A_{fv3} = r_{fv3}^2$$
$$A_{mpas} = r_{mpas}^2$$

onde:
A_{fv3} - Área de um pixel do modelo FV3
A_{mpas} - Área de um pixel do modelo MPAS
r_{fv3} - Resolução do FV3
r_{mpas} - Resolução do MPAS

$$P_{fv3/mpas} = \frac{(A_t/A_{fv3})}{(A_t/A_{mpas})}\frac{L_{fv3}}{L_{mpas}}$$

onde:
L_{fv3} - Número de Níveis do FV3

L_{mpas} - Número de Níveis do MPAS

Não foram consideradas as características da grade de Voronoi do MPAS ou especificidades da esfera cubada do FV3. Tratou-se cada pixel como um ponto quadrado de lado igual a resolução de cada um.

Obteve-se o valor 1,5 para essa razão. Os resultados de tempo obtidos nos experimentos com o modelo MPAS foram multiplicados por esse valor.

Há que se ressaltar que essa extrapolação pode não representar muito bem a realidade envolvendo GPUs pois, em geral, o aumento de resolução de modelos aplicadas a GPU costuma ter como resultado uma eficiência melhor. Quanto mais cálculos efetuados melhor a resposta das GPUs.

Observando os resultados apresentados nas figuras 12 e 13 verifica-se que há um desempenho melhor para o modelo FV3 usando CPU. O FV3 é aproximadamente 44% mais rápido que o modelo MPAS quando rodado com 64 nós.

Quando comparamos o modelo FV3 rodando em CPU com o modelo MPAS rodando em GPU observa-se que o modelo MPAS é 11% mais rápido para 64 nós. Para 32 nós, até onde o MPAS apresentou eficiência aceitável (estimada) a razão entre o MPAS e o FV3 é de 59%.

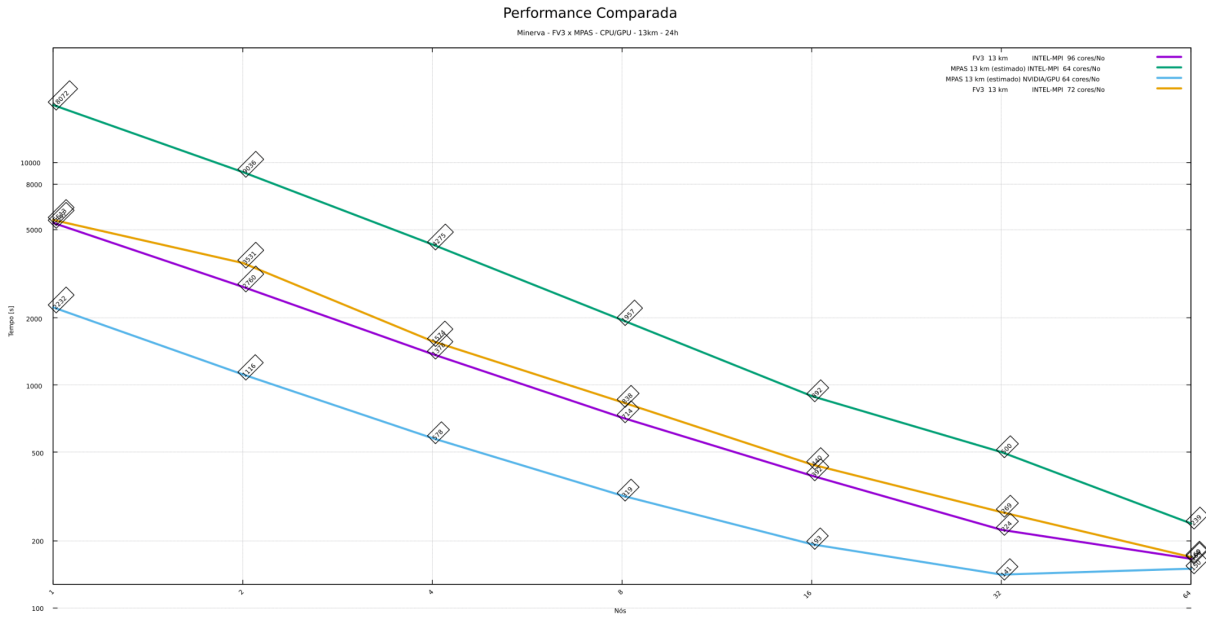

Figura 12 - Comparação da performance entre os modelos

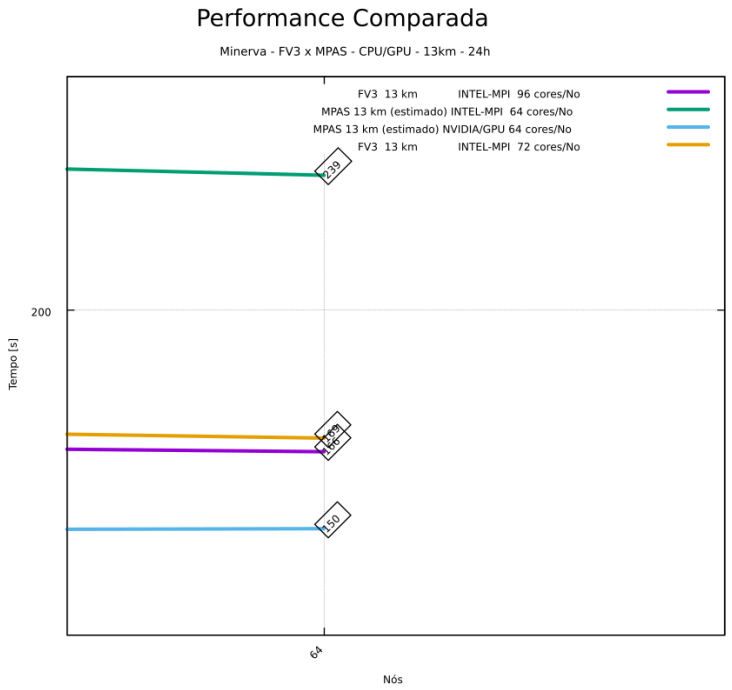

Figura 13 - Comparação da performance entre os modelos - detalhes para 64 nós

Discussão e Análise

Os resultados dos requisitos não funcionais obtidos nos experimentos nos permitem avaliar que o núcleo dinâmico do modelo MPAS apresenta vantagem na manutenibilidade e na portabilidade. O MPAS vence com 152 pontos contra 132 do FV3 em manutenibilidade e 8,4 a 4,4 pontos em portabilidade.

A manutenibilidade é fundamental para garantir um longo ciclo de vida de software e garante mais facilidade em sua melhoria, manutenção e correção de bugs. A maturidade é diretamente dependente desses requisitos não funcionais. Com a evolução constante das arquiteturas e compiladores é fundamental que os softwares apresentem menos impedimentos e exigências de trabalhos extras para migrá-los quando da aquisição de novos sistemas computacionais ou do lançamento de novas versões de compiladores e bibliotecas. Dessa forma a vantagem apresentada pelo núcleo dinâmico do modelo MPAS é significativa e deve ser considerada na escolha do futuro núcleo dinâmico do modelo MONAN.

A análise do desempenho dos dois núcleos dinâmicos mostra que o FV3 tem nítida vantagem em sistemas computacionais baseados em CPU. Já o núcleo dinâmico do modelo MPAS é superior que o FV3 quando comparamos rodadas em GPU e rodadas do FV3 em CPU. Infelizmente não tivemos acesso ao código do núcleo dinâmico do FV3 portado para GPU que poderia mostrar uma comparação direta sob a mesma arquitetura.

O documento **“FV3 on GPUs**” de Lucas Harris, Rusty Benson e Oli Fuhrer[12] mostra que existem esforços de migração do FV3 para GPU. Há um alerta na apresentação citada onde verifica-se que o modelo necessita de algum trabalho para que possa usar adequadamente arquiteturas GPUs. Apesar dos esforços envidados junto a empresa NVIDIA para tentar receber esse código, não pudemos ser atendidos. Já o MPAS foi totalmente portado e tem excelente comportamento nessa arquitetura.

Não cabe aqui a escolha do melhor núcleo dinâmico a ser adotado pelo futuro modelo MONAN. Mas esse trabalho apresenta subsídios e aponta variáveis importantes para essa escolha.

Trabalhos futuros

Alguns pontos envolvendo os modelos ainda cabem ser elucidados. Um deles é tentar determinar a razão que leva a super linearidade observada com o modelo MPAS sendo executado em máquinas baseadas em arquitetura X86 (CPU). Outra futura investigação é tentar determinar o comportamento do modelo MPAS usando máquinas maiores baseadas em GPU, com mais memória e com as arquiteturas atualizadas.

Por fim é fundamental que os dois núcleos dinâmicos sejam executados de forma pré-operacional, de forma contínua, tanto em arquitetura X86 (CPU) quanto em máquinas com GPU. O INPE dispõe de máquinas X86 (CPU) que permitem a rodada dos dois modelos para exames comportamentais e de estabilidade. Não existe ainda uma máquina que permita examinar continuamente o comportamento em máquinas com GPU. Ainda que uma máquina pequena que permita executar o modelo com resolução grossa.

As máquinas usadas na avaliação dos núcleos dinâmicos foram gentilmente cedidas pela empresa DELL que não tem responsabilidade de mantê-las disponíveis e portanto cabe a aquisição de máquina para substituí-la.

[12] https://ufscommunity.org/wp-content/uploads/2021/04/Harris-UFS-Webinar-25-mar-21.pdf

APÊNDICE A - CARACTERÍSTICAS DA QUALIDADE DE SOFTWARE

As características de qualidade selecionadas para a avaliação do Core Dinâmico foram baseadas nas normas ISO e estão descritas abaixo.

Funcionalidade: A capacidade de um software prover funcionalidades que satisfaçam o usuário em suas necessidades declaradas e implícitas, dentro de um determinado contexto de uso. Suas Subcaracterísticas são:

- Adequação: Capacidade do produto de software de prover um conjunto apropriado de funções para tarefas e objetivos do usuário especificados.
- Acurácia: Capacidade do produto de software de prover, com o grau de precisão necessário, resultados ou efeitos corretos ou conforme acordados.
- Interoperabilidade: Capacidade do produto de software de interagir com um ou mais sistemas especificados.

Usabilidade: A capacidade do produto de software de ser compreendido, aprendido, operado e atraente ao usuário, quando usado sob condições especificadas. Note que este conceito é bastante abrangente e se aplica mesmo a programas que não possuem uma interface para o usuário final. Por exemplo, um programa batch executado por uma ferramenta de programação de processos também pode ser avaliado quanto a sua usabilidade, no que diz respeito a ser facilmente compreendido, aprendido, etc. Suas Subcaracterísticas são:

- Inteligibilidade: Capacidade do produto de software de possibilitar ao usuário compreender se o software é apropriado e como ele pode ser usado para tarefas e condições de uso específicas.
- Apreensibilidade: Capacidade do produto de software de possibilitar ao usuário aprender sua aplicação.
- Operacionalidade: Capacidade do produto de software de possibilitar ao usuário operá-lo e controlá-lo.
- Proteção frente a erros de usuários: como produto consegue prevenir erros dos usuários;

Eficiência: O tempo de execução e os recursos envolvidos são compatíveis com o nível de desempenho do software. Suas Subcaracterísticas são:

- Comportamento em Relação ao Tempo: avalia se os tempos de resposta (ou de processamento) estão dentro das especificações;
- Utilização de Recursos: mede tanto os recursos consumidos quanto a capacidade do sistema em utilizar os recursos disponíveis; exemplo: processador e memória.

Manutenibilidade: A capacidade (ou facilidade) do produto de software ser modificado, incluindo tanto as melhorias ou extensões de funcionalidade quanto às correções de defeitos, falhas ou erros. Suas Subcaracterísticas são:

- Analisabilidade: identifica a facilidade em se diagnosticar eventuais problemas e identificar as causas das deficiências ou falhas.
- Modificabilidade: caracteriza a facilidade com que o comportamento do software pode ser modificado.
- Reusabilidade: identifica a capacidade de se reutilizar o software por completo ou partes dele.
- Estabilidade: avalia a capacidade do software de evitar efeitos colaterais decorrentes de modificações introduzidas.
- Testabilidade: representa a capacidade de se testar o sistema modificado, tanto quanto as novas funcionalidades quanto as não afetadas diretamente pela modificação.

Portabilidade: A capacidade do sistema ser transferido de um ambiente para outro. Como "ambiente", devemos considerar todos os fatores de adaptação, tais como diferentes condições de infraestrutura (sistemas operacionais, versões de bancos de dados, etc.), diferentes tipos e recursos de hardware (tal como aproveitar um número maior de processadores ou memória). Além destes, fatores como idioma ou a facilidade para se criar ambientes de testes devem ser considerados como Características de portabilidade. Suas Subcaracterísticas são:

- Adaptabilidade: representando a capacidade do software se adaptar a diferentes ambientes sem a necessidade de ações adicionais configurações).
- Capacidade para ser Instalado: identifica a facilidade com que pode se instalar o sistema em um novo ambiente.
- Coexistência: mede o quão facilmente um software convive com outros instalados no mesmo ambiente.
- Capacidade para Substituir: representa a capacidade que o sistema tem de substituir outro sistema especificado, em um contexto de uso e ambiente específicos. Este atributo interage tanto com adaptabilidade quanto com a capacidade para ser instalado.

APÊNDICE B – CRITÉRIOS DE PONTUAÇÃO DE SUBCARACTERÍSTICAS E CARACTERÍSTICAS

As Características de Qualidade descritas no item 2 devem ser avaliadas no início do projeto para ajudar a escolher o ND.

Os critérios de pontuação definidos pelas normas ISO são:

- Três pontos para as subcategorias que atendem satisfatoriamente os requisitos de qualidade (excelente, bom e razoável).
- Um ponto para as subcategorias com pontuação Insatisfatória.

Esses critérios de pontuação serão utilizados como base para a avaliação que pode ser feita por métricas, da seguinte forma: Para cada métrica, os NDs que apresentarem:

- a melhor pontuação da métrica: recebem três pontos para cada subcategoria de qualidade relacionada à métrica (Ex. Tabela C.2).
- As piores pontuações da métrica: recebem um ponto para cada subcategoria de qualidade relacionada à métrica.

Em caso de empate dos melhores, os empatados recebem três pontos. Analogamente, os piores empatados recebem um ponto.

Ao final da pontuação de métricas, para cada subcategoria, serão somados os pontos. Um maior somatório de pontos de subcategorias, entre os NDs avaliados, indica uma melhor pontuação para uma Categoria, que recebe três pontos. Os menores somatórios, um ponto.

APÊNDICE C – AVALIAÇÃO DE CARACTERÍSTICAS DA MANUTEBILIDADE

C.1 Introdução

Métricas de Software podem ser utilizadas para a avaliação de algumas Características de Qualidade. Existem métricas para avaliação de código estruturado e de código Orientado a Objeto. Serão utilizadas somente as métricas de código estruturado, por se tratar da técnica de codificação comum entre os códigos analisados.

Diversos tipos de métricas foram criadas para avaliar a qualidade dos softwares. Algumas delas medem a complexidade do software e características da linguagem de programação, relacionados à Manutenibilidade.

C.1.1 Métrica de Complexidade Ciclomática de McCabe

McCabe desenvolveu uma métrica que permite aos desenvolvedores identificar módulos difíceis de testar ou manter. Como resultado, ele desenvolveu uma métrica de software que iguala a complexidade ao número de decisões em um programa. Os desenvolvedores podem usar essa medida para determinar quais módulos de um programa são excessivamente complexos e precisam ser recodificados. A métrica também pode ser usada para definir o número mínimo de casos de teste necessários para testar adequadamente os caminhos do programa.

Os valores de referência são exibidos na Tabela C.1, abaixo:

Tabela C.1 - Valores de referência da Complexidade Ciclomática de McCabe

Complexidade	Avaliação
1-10	Método simples. Baixo risco
11-20	Método razoavelmente complexo. Moderado risco
21-50	Método muito complexo. Elevado risco
51-N	Método de altíssimo risco e bastante instável

C.1.2 Métricas de Software da RADC

A metodologia RADC consiste em elementos métricos, métricas, critérios e fatores. As pontuações dos elementos de métrica são combinadas para formar uma pontuação de métrica e as pontuações de métrica são combinadas para formar uma pontuação de critério. As pontuações de critérios são então combinadas de várias maneiras para produzir várias pontuações de fatores de qualidade. Elementos métricos são as medidas quantitativas de objetos de software. Métricas são os detalhes orientados ao software das características do software. Critérios são as características orientadas ao software que contribuem para vários tipos de qualidade.

C.2 Metodologia

A metodologia consistiu em utilizar métricas de software, baseadas na metodologia RADC, na qual todas Subcaracterísticas relacionadas à métrica podem ser pontuadas negativamente ou positivamente. Essa relação está descrita na seção C.2.1. O cálculo das métricas é avaliado sobre os códigos de versões dos modelos que se deseja avaliar.

C.2.1 Métricas

As métricas abaixo foram selecionadas para avaliar as Subcaracterísticas de Manutenibilidade. O método de pontuação de métricas está descrito no item 6 do documento original - Avaliação da Qualidade Total, e também no Apêndice B deste, onde aqui se destaca os critérios de pontuação. A Tabela C.2 exibe todas as métricas e suas descrições utilizadas na avaliação (colunas “Métrica” e “Descrição”), as subcaracterísticas que são impactadas positivamente e negativamente, com o aumento do valor da métrica (colunas “Impactos Positivos” e “Impactos Negativos”). A unidade de medida da métrica está disposta na última coluna “Unidade de medida”. As ferramentas utilizadas para o cálculo de cada métrica estão dispostas na coluna Ferramentas).

Tabela C.2 - Métricas utilizadas na avaliação da Manutebilidade

Métrica	Descrição	Impactos Positivos	Impactos Negativos	Ferramentas	Unidade de medida
Comprimento de código	Essa é uma medida do tamanho de um programa. Geralmente, quanto maior o tamanho do código de um componente, mais complexo e sujeito a erros o componente é. O comprimento de código tem mostrado	-	Analisabilidade, Modificabilidade, Estabilidade, Testabilidade	FortranAnalyser	linhas de código

	ser uma das métricas mais confiáveis para prever a propensão a erros em componentes				
Complexidade Ciclomática Média (McCabe)	Permite aos desenvolvedores identificar módulos difíceis de testar ou manter. Calcula o número de decisões em um programa, que pode ser usado para determinar quais módulos de um programa são excessivamente complexos e precisam ser recodificados.	-	Analisabilidade, Modificabilidade, Reusabilidade, Estabilidade, Testabilidade	FortranAnalyser	Ver tabela C.1
Documentação total	Mede a porcentagem de código comentado (todo o código) (vazios e "!" sem continuação estão excluídos).	Analisabilidade, Modificabilidade	-	Check.py	razão
Documentação de rotinas	Mede a porcentagem rotinas comentadas. Facilita o entendimento do código e registra alterações históricas.	Analisabilidade, Modificabilidade, Reusabilidade, Testabilidade	-	FortranAnalyser	razão
Documentação de arquivos	Mede a porcentagem arquivos com comentários no início. Facilita o entendimento do código e registra alterações históricas.	Analisabilidade, Modificabilidade	-	FortranAnalyser	razão
Documentação de estruturas de controle	Mede a porcentagem de comentários em estuturas de controle. Facilita o entendimento do código.	Analisabilidade, Modificabilidade	-	FortranAnalyser	razão
Tamanho médio das rotinas	Tamanho médio em linhas de rotinas, que afeta a compreensibilidade e Manutebilidade	-	Analisabilidade, Modificabilidade, Reusabilidade, Estabilidade, Testabilidade	Check.py	linhas de código

Tamanho médio dos módulos	Dado importante. Quando comparado com o número médio das rotinas, se menor, vai indicar que as rotinas não estão encapsuladas em módulos.	-	Analisabilidade, Modificabilidade, Reusabilidade, Estabilidade, Testabilidade	Check.py	linhas de código
Tamanho médio do nome das variáveis	Considera-se rotinas e funções	Analisabilidade, Modificabilidade, Testabilidade	-	Check.py	quantidade de caracteres
Razão de only em uses	A falta do only é problemática, pois propaga todas as variáveis para a estrutura e comumente é fonte de bugs.	Analisabilidade, Modificabilidade, Estabilidade	-	Check.py	razão
Razão de "goto" e "continue" por laço	O uso de gotos e continues é desconselhado, pois torna o código incompreensível e de difícil manutenção. Idealmente deve ser zero.	-	Analisabilidade, Modificabilidade, Estabilidade.	Check.py	razao
Razão de "exit" e "cycle" por laço	O baixo uso de exit e cycle indica estruturas de laços mal formadas e podem apontar para códigos "macarrônicos"	Analisabilidade, Modificabilidade, Estabilidade	-	Check.py	razão
Razão do uso de "implicit	Todas as rotinas e módulos deveriam ter "implicit" pois impedem erros por variáveis não declaradas e bugs potenciais. Idealmente deve ser 100%	Analisabilidade, Modificabilidade, Estabilidade	-	Check.py	razão
Total de "equivalence" ou "common"	Equivalence ou common não são recomendados. Indicam estruturas de codificação antigas e riscos de bugs não mapeados. Essas duas	-	Modificabilidade, Estabilidade	Check.py	soma de linhas com palavra chave

	keywords devem ser proibidas pois levam a erros graves quando partes do código são modificadas e vão afetar outras.				
Profundidade média de laços	Mede a média de linhas em laços. Laços muito grandes devem ser evitados pois são de baixa compreensão	-	Analisabilidade, Modificabilidade, Reusabilidade, Estabilidade, Testabilidade	Check.py	número de linhas
Aninhamento médio de laços	Número de laços aninhados médio. Grandes aninhamentos indicam complexidade.	-	Analisabilidade, Modificabilidade, Reusabilidade, Estabilidade, Testabilidade	Check.py	número de laços aninhados médio
Fan-in	Média de chamadas por subrotina. Número de vezes que a mesma subrotina é chamada. Números altos indicam que erros podem ser corrigidos em área comum facilitando a manutenção, mas que pode afetar a estabilidade	Analisabilidade, Modificabilidade, Reusabilidade, Testabilidade	Estabilidade	FortranMakeUtils	número de chamadas
Fan-out	Número de rotinas que são chamadas por cada rotina (digamos, X). Um valor alto para fan-out sugere que a complexidade geral do X pode ser alta, devido a complexidade da lógica de controle necessária para coordenar as rotinas chamadas		Modificabilidade, Reusabilidade, Estabilidade, Testabilidade	FortranMakeUtils	número de chamadas
Aninhamento médio de rotinas	Um número alto pode indicar que a complexidade da rotina X pode ser alta, devido a complexidade da lógica de controle necessária para coordenar as rotinas chamadas		Modificabilidade, Reusabilidade, Estabilidade, Testabilidade	FortranMakeUtils	número de rotinas aninhadas

C.3 Ferramentas para avaliação da Manutebilidade

As ferramentas necessárias para a avaliação estão versionadas em https://github.com/monanadmin/monan/tree/main/tools/qas_eval . Algumas ferramentas foram adaptadas, como o FortranAnalyser, para possibilitar o uso sem interface gráfica. Outras utiliza-se somente o código binário, como no caso do fortran-src.

Uma imagem do container singularity (qas_eval.sif) foi criada para armazenar e executar as ferramentas binárias (fortran-src e FortranAnalyser.jar) sobre uma distribuição Linux compatível com as ferramentas. As ferramentas binárias estão disponíveis na pasta /home/qas_files/tools/ dentro da imagem.

Ferramentas com código fonte no repositório:

- check.py (https://github.com/monanadmin/monan/tree/main/tools/qas_eval). Programa desenvolvido para ler estatísticas geradas pelo programa (binário) fortran-src. Também contabiliza outras estatísticas através da análise direta do código, gerando um relatório sintetizado de métricas.
- FortranAnalyser (http://fortrananalyser.ephyslab.uvigo.es/). Ferramenta escrita em java, modificada para ser executada em linha de comando. Gera um relatório em PDF contendo estatísticas e escores, que são gerados por arquivo e em forma sintética para todo o conjunto de arquivos.
- FortranMakeUtils (https://github.com/deniseiras/fortranMakeUtils). Ferramenta desenvolvida inicialmente para gerar a hierarquia de dependência de módulos e gerar os makefiles automaticamente. As dependências geradas entre chamadas de rotinas foi utilizada no check.py para calcular métricas como Fan-In e Fan-Out.

Ferramentas com código binário:

- fortran-src (https://github.com/camfort/fortran-src). Ferramenta escrita na linguagem Haskell que gera diversas estatísticas do código.
- FortranAnalyser.jar. Binário gerado pelo código fonte. Para gerar o binário a partir do código fonte, baixe ou instale a ferramenta maven, entre na pasta principal do FortranAnalyser (onde se encontra o pom.xml) e execute "mvn clean install".

C.4 Geração das métricas

1. Use o módulo singularity no servidor Rattler (a) ou instale em outro local desejado (b). OBS: A versão do singularity disponível para Ubuntu 18.04 (2.6.1-dist - via apt install) não funciona com o pull do passo 3. A instalação manual de versões mais recentes, via passos de https://sylabs.io/guides/3.0/user-guide/installation.html , requer pacotes que não estão disponíveis na versão 18.04 do Ubuntu. Recomenda-se alguma das formas abaixo:
    - a) module load singularity # máquina Rattler
    - b) sudo apt install -y singularity-container
2. Crie uma pasta de nome qas_eval e entre nela

    - mkdir qas_eval;ND qas_eval
3. Baixe a imagem do singularity na pasta criada, com uma das opções abaixo. Use a opção b) caso não consiga usar o pull (ex. Ubuntu 18.04) em a):
    - a) singularity pull qas_eval.sif library://denis.eiras/monan/qas_eval:dev
    - b) Clique no link de download disponível na página https://cloud.sylabs.io/library/denis.eiras/monan/qas_eval e baixe a imagem para a pasta atual (qas_eval) usando o nome qas_eval.sif
4. Baixe o código inteiro do monan (será usada a subpasta tools):
    - git clone https://github.com/monanadmin/monan.git
5. Execute o script principal. Todas as subpastas dos NDs dos códigos em monan/codigos_originais, serão avaliadas:
    - monan/tools/qas_eval/run_eval.sh

O último passo executará as ferramentas para os códigos de NDs e gerará os seguintes relatórios:

- QualityReport_[MODELO].pdf : Relatório gerado pela ferramenta FortranAnalyser. As estatísticas sintetizadas estão no final do arquivo
- Check_Report_[MODEL].txt : Relatório gerado pela ferramenta Check.py.

Os relatórios informam os valores das métricas obtidas por cada ND, que serão usados para pontuar as Características de Qualidade mapeadas para cada métrica (Tabela C.2).