ESTUDO E IMPLEMENTAÇÃO DE UMA METODOLOGIA DE ESCRITA BINÁRIA BASEADA EM UM ACOPLAMENTO ENTRE HDF5 E XDMF

Marcelo Maia Ribeiro Damasceno, MFLab, marcelomrd@gmail.com
João Marcelo Vedovoto, MFLab, jmvedovoto@mecanica.ufu.br
Aristeu da Silveira Neto, MFLab, aristeus@ufu.br

Resumo. *O presente trabalho tem como objetivo a busca por uma alternativa para o método de escrita em códigos computacionais de alto desempenho. Para tal fim, uma metodologia baseada em um acoplamento entre HDF5 e XDMF foi estudada e implementada em um código baseado em volumes finitos. Resultados significativos foram obtidos através da realização de tal tarefa, fazendo desta metodologia uma interessante alternativa para a geração de arquivos de pós processamento.*

Palavras chave*: hdf5, xdmf, escrita binária.*

1. INTRODUÇÃO

O estudo do comportamento de escoamentos com relevância prática para a engenharia, na grande maioria dos casos, relaciona-se com o estudo da turbulência. A caracterização desta, por sua vez, demanda a resolução de uma grande faixa de escalas de tempo e de comprimento, situação que necessita de uma significativa quantidade de recursos computacionais, quando utiliza-se CFD. A resolução de um problema com esta metodologia abrange três etapas básicas: pré-processamento (geração de geometria, malha e organização das informações do problema), processamento (solução numérica das equações que caracterizam o mesmo) e o pós-processamento (representação dos resultados obtidos em forma gráfica).

O aumento do poder de processamento computacional acarretou na possibilidade de experimentação numérica de uma maior faixa de escoamentos reais. Uma consequência imediata de tal fato é a grande quantidade de dados gerados, além do aumento da demanda de armazenamento dos mesmos. Neste âmbito, duas ferramentas são estudadas neste trabalho: HDF5 e XDMF. A primeira pode ser definida como um modelo para manipulação e armazenamento de dados "pesados" (valores numéricos armazenados das variáveis), ao passo que a segunda é aplicada ao armazenamento de dados "leves" (descrição dos dados "pesados").

2. FUNDAMENTOS

Nas seções que se seguem, uma breve descrição das principais características das duas metodologias estudadas será apresentada, baseada em informações disponibilizadas pelos grupos The HDF group e XDMFWeb.

2.1. HDF5

O HDF5 pode ser descrito como um algoritmo capaz de gerenciar e de armazenar dados. Tal algoritmo é composto por modelos de dados e de armazenamento abstratos (MDA e MAA, respectivamente), além de bibliotecas para a implementação dos mesmos e para o mapeamento do armazenamento, para diferentes mecanismos. Esta biblioteca também dispõe de um eficiente algoritmo para a troca de dados entre as representações armazenadas.

O MDA diz respeito a um modelo conceitual de dados, tipo e organização dos mesmos, além de ser independente do meio de armazenamento ou ambiente de programação utilizado. O MAA, por sua vez, é uma representação padronizada para os objetos do MDA e este é definido pelas especificações de formato de arquivos do HDF5.

Na seção seguinte, um maior detalhamento do modelo de dados abstrato é apresentado. O MAA, entretanto, não será discutido com mais intensidade, já que este não é o intuito do presente trabalho.

2.1.1. MDA

O MDA é responsável pela definição de conceitos para a definição e descrição dados complexos armazenados em arquivos. Vários tipos de dados podem ser mapeados em objetos do MDA e, portanto, armazenados e recuperados com a utilização de HDF5. Entretanto, para tal utilização, os dados a serem utilizados deverão ser convertidos para os conceitos desta metodologia, que são os seguintes:

- Arquivo: é um recipiente para um conjunto organizado de objetos (grupos, conjuntos de dados, etc.). Cada arquivo HDF5 possui, pelo menos, um objeto, o grupo raiz.
- Grupo: um coleção de objetos (incluindo grupos).
- Conjuntos de dados: um arranjo multidimensional de elementos de dados com atributos e outros metadados.

- Espaço de dados: descrição das dimensões de um arranjo multidimensional.
- Tipo de dados: descrição de uma classe específica de elementos de dados, incluindo, por exemplo, o layout de armazenamento da mesma como um padrão de bits.
- Atributos: um valor de dados nomeado, associado com um grupo, conjunto de dados ou tipo de dados nomeado.
- Lista de propriedades: uma coleção de parâmetros para o controle das opções nas bibliotecas.

Torna-se interessante, portanto, o conhecimento da estrutura de um arquivo dotado destes conceitos. Tal explicitação é realizada a seguir.

2.1.2. Estrutura de um arquivo HDF5

A Fig. (1), disposta a seguir, denota uma representação esquemática da organização de um arquivo HDF5, contendo grande parte dos conceitos que foram mencionados anteriormente. Na figura mencionada, pode-se observar a existência de quatro grupos (/, group1, group2 e group2), três conjuntos de dados (dset1, dset2 e dset2) e as conexões entre eles, caracterizados, respectivamente, por círculos, quadrados e setas.

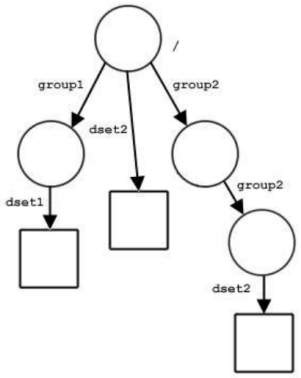

Figura 1. Representação esquemática da organização de um arquivo HDF5.

Tal como foi mencionado anteriormente, o grupo raiz sempre recebe o nome "/". Os outros objetos criados são inseridos com a respectiva conexão entre o grupo anterior. Pelo exemplo apresentado, percebe-se que um objeto pode ser alvo de mais de uma conexão. Por este motivo, os nomes nestas devem ser exclusivos dentro de cada grupo, o que não é mandatório caso os objetos estejam em grupos diferentes. Finalmente, o arquivo é acessado de forma similar à utilização de sistemas de arquivo Unix. Como exemplo, o acesso ao conjunto de dados dset2 é realizado da seguinte forma: /group2/group2/dset2. A "tradução do arquivo" binário, gerado pelo HDF5, será realizada pelo XDMF. Tal ferramenta é explicitada na seção seguinte.

2.2. XDMF

O modelo de dados em XDMF, armazenado em XML, provê a tradução do que representam os dados "pesados". Neste modelo, os dados são visualizados como uma hierarquia de domínios, que devem conter um mínimo de uma malha. A malha, por sua vez, denota a representação básica dos valores geométricos e computados/medidos. É considerada como sendo um grupo de elementos com topologia estruturada ou não estruturada e de seus valores associados.

Esta ferramenta utiliza o XML como um HTML (*HyperText Markup Language*, linguagem de marcação de hipertexto) personalizado, sensível a caracteres maiúsculos e minúsculos e composto de três componentes principais: elementos, entidades e informações de processamento.

Analogamente ao que foi apresentado com respeito ao HDF5, uma breve explicitação da estrutura de um arquivo XDMF é realizada a seguir.

2.2.1. Estrutura de um arquivo XDMF

Um arquivo XDMF é composto por um elemento Xdmf, que contêm um ou mais elementos *Domain* (referente ao domínio computacional). Os últimos, por sua vez, podem possuir um ou mais elementos *Grid* (denotando a malha utilizada), relacionados à porção do modelo de dados desta ferramenta. Este elemento é um "recipiente" para as informações relativas aos pontos 2D e 3D, conectividades estruturadas ou não estruturadas e valores atribuídos. O elemento *Grid* possui uma atribuição *GridType*, que deve ser do tipo:

- *Uniform* - uma simples malha homogênea.

- *Collection* - um conjunto de malhas uniformes, todas com os mesmos *Attributes.*
- *Tree* - um grupo hierárquico.
- *SubSet* - uma parte de outro *Grid.*

O *Grid* é composto por elementos do tipo *Topology*, *Geometry* e zero ou mais *Attributes*, que serão definidos de acordo com o problema a ser estudado:

- *Topology*: especifica a conectividade da malha. Os seguintes tipos de topologias de células são definidos:
  - Linear: *Polyvertex* (um grupo de pontos não conectados), *Polyline* (um grupo de segmentos de reta), *Polygon*, *Triangle*, *Quadrilateral*, *Tetrahedron*, *Pyramid*, *Wedge* e *Hexahedron*.
  - Quadrático: *Edge_3* (linha quadrática com três nós), *Tri_6*, *Quad_8*, *Tet_10*, *Pyramid_13*, *Wedge_15* e *Hex_20*.
  - Arbitrário: *Mixed* (uma mistura de células não estruturadas).
  - Estruturado: *2DSMesh* (curvilínea), *2DRectMesh* (eixos perpendiculares), *2DCoRectMesh* (eixos perpendiculares e espaçamento constante), *3DSMesh*, *3DRectMesh* e *3DCoRectMesh*.
- *Geometry*: descreve os valores XYZ da malha. A importância deste elemento está relacionada com a organização dos pontos. Possíveis parâmetros são: *XYZ* (locais entrelaçados), *XY* (Z é igualado a zero), *X_Y_Z* (X,Y e Z são conjuntos separados), *VXVYVZ* (três vetores, um para cada eixo) e *ORIGIN_DXDYDZ* (seis valores: Ox,Oy,Oz + Dx,Dy,Dz).
- *Attributes*: define valores associados com a malha. Os tipos de valores suportadossão: *Scalar*, *Vector*, *Tensor* (nove valores esperados), *Tensor6* (um tensor simétrico) e *Matrix* (uma matriz arbitrária NxM). Estes valores podem ser centrados em: *Node*, *Edge*, *Face*, *Cell*, e *Grid*.

Por fim, um elemento do tipo *Time*, também definido como um filho do elemento *Grid*, é responsável pelas informações temporais para a malha. O tipo de elemento temporal é definido pelo atributo *TimeType*. Parâmetros válidos para tal especificação são: *Single* (um único valor temporal para toda a malha), *HyperSlab* (início, passo e índices de contador), *List* (uma lista temporal discreta) e *Range* (valores mínimo e máximo).

A partir do estudo destas duas ferramentas, tornou-se possível a implementação das mesmas no código computacional FLUIDS3D, desenvolvido por Vedovoto et al. (2011). Os resultados obtidos com a realização de tal tarefa são apresentados a seguir.

3. RESULTADOS E DISCUSSÕES

A Fig. (2), disposta a seguir, apresenta uma implementação simplificada da subrotina criada no código computacional mencionado.

```
        CALL h5open_f(error)
          CALL h5fcreate_f(filename, H5F_ACC_TRUNC_F, file_id, error)
!-------------------------------------------------------------------------------
!  Dataspace/dataset z
!-------------------------------------------------------------------------------
          CALL h5screate_simple_f(rank2, dimsz, z_space_id, error)
          CALL h5dcreate_f(file_id, dsetname_z, H5T_NATIVE_DOUBLE, z_space_id, &
                           z_id, error)
          CALL h5dwrite_f(z_id, H5T_NATIVE_DOUBLE, z(1:ztot), data_dims2, error)
          CALL h5dclose_f(z_id, error)
          CALL h5sclose_f(z_space_id, error)
!-------------------------------------------------------------------------------
!  Dataspace/dataset y
!-------------------------------------------------------------------------------
          CALL h5screate_simple_f(rank2, dimsy, y_space_id, error)
          CALL h5dcreate_f(file_id, dsetname_y, H5T_NATIVE_DOUBLE, y_space_id, &
                           y_id, error)
          CALL h5dwrite_f(y_id, H5T_NATIVE_DOUBLE, y(1:ytot), data_dims2, error)
          CALL h5dclose_f(y_id, error)
          CALL h5sclose_f(y_space_id, error)
!-------------------------------------------------------------------------------
!  Dataspace/dataset x
!-------------------------------------------------------------------------------
          CALL h5screate_simple_f(rank2, dimsx, x_space_id, error)
          CALL h5dcreate_f(file_id, dsetname_x, H5T_NATIVE_DOUBLE, x_space_id, &
                           x_id, error)
          CALL h5dwrite_f(x_id, H5T_NATIVE_DOUBLE, x(1:xtot), data_dims2, error)
          CALL h5dclose_f(x_id, error)
          CALL h5sclose_f(x_space_id, error)
!-------------------------------------------------------------------------------
!  Dataspace/dataset u
!-------------------------------------------------------------------------------
          CALL h5screate_simple_f(rank, dims, u_space_id, error)
          CALL h5dcreate_f(file_id, dsetname_u, H5T_NATIVE_DOUBLE, u_space_id, &
                           u_id, error)
          CALL h5dwrite_f(u_id, H5T_NATIVE_DOUBLE, un(1:xtot,1:ytot,1:ztot), data_dims, error)
          CALL h5dclose_f(u_id, error)
          CALL h5sclose_f(u_space_id, error)
!-------------------------------------------------------------------------------
          CALL h5fclose_f(file_id, error)
          CALL h5close_f(error)
```

(a)

```
<?xml version="1.0" ?>
<!DOCTYPE Xdmf SYSTEM "Xdmf.dtd" []>
<Xdmf Version="2.0">
 <Domain>
   <Grid Name="mesh" GridType="Collection" CollectionType="Spatial">
    <Time Type="Single" Value="   0.12727272727736499E-03"/>
    <Grid Name="mesh0" GridType="Uniform">
     <Topology TopologyType="3DRectMesh" NumberOfElements="  11  11  46"/>
     <Geometry GeometryType="VXVYVZ">
       <DataItem Dimensions="  46" NumberType="Float" Precision="8" Format="HDF">
        ./hdf5/output_000_0000.h5:/X
       </DataItem>
       <DataItem Dimensions="  11" NumberType="Float" Precision="8" Format="HDF">
        ./hdf5/output_000_0000.h5:/Y
       </DataItem>
       <DataItem Dimensions="  11" NumberType="Float" Precision="8" Format="HDF">
        ./hdf5/output_000_0000.h5:/Z
       </DataItem>
     </Geometry>
     <Attribute Name="U" AttributeType="Scalar" Center="Node">
       <DataItem Dimensions="  11  11  46" NumberType="Float" Precision="8" Format="HDF">
        ./hdf5/output_000_0000.h5:/U
       </DataItem>
     </Attribute>
    </Grid>
    <Grid Name="mesh1" GridType="Uniform">
     <Topology TopologyType="3DRectMesh" NumberOfElements="  11  11  46"/>
     <Geometry GeometryType="VXVYVZ">
       <DataItem Dimensions="  46" NumberType="Float" Precision="8" Format="HDF">
        ./hdf5/output_001_0000.h5:/X
       </DataItem>
       <DataItem Dimensions="  11" NumberType="Float" Precision="8" Format="HDF">
        ./hdf5/output_001_0000.h5:/Y
       </DataItem>
       <DataItem Dimensions="  11" NumberType="Float" Precision="8" Format="HDF">
        ./hdf5/output_001_0000.h5:/Z
       </DataItem>
     </Geometry>
     <Attribute Name="U" AttributeType="Scalar" Center="Node">
       <DataItem Dimensions="  11  11  46" NumberType="Float" Precision="8" Format="HDF">
        ./hdf5/output_001_0000.h5:/U
       </DataItem>
     </Attribute>
    </Grid>
   </Grid>
 </Domain>
</Xdmf>
```

(b)

Figura 2. Implementação simplificada de uma sub-rotina para escrita binária: (a) HDF5 (b) XDMF.

Tal exemplo caracteriza a criação, em HDF5, de três vetores de coordenadas (x, y e z) e uma matriz para a velocidade na abcissa (u). Por este motivo, os parâmetros *rank2*, *dimsx*, *dimsy e dimsz* são unidimensionais, ao passo que *rank* e *dims* são tridimensionais. O XDMF, por sua vez, descreve a malha utilizada como sendo um conjunto de duas malhas uniformes (*mesh0* e *mesh1*). Tal descrição é realizada com a definição (*CollectionType="Spatial"*).

Percebe-se, também, que a topologia da malha foi descrita como tridimensional e com eixos perpendiculares (*TopologyType="3DRectMesh"*). Por fim, a organização dos pontos da malha foram descritos como três vetores para cada eixo (*GeometryType="VXVYVZ"*) e os valores das variáveis, propriamente ditos, são obtidos de arquivos HDF5, através da definição do diretório no qual os mesmos se encontram como, por exemplo, *./hdf5/output_000_0000.h5:/X*. Neste caso, busca-se a variável *X* no arquivo *output_000_0000.h5*.

Após a realização de tais implementações, uma comparação entre as metodologias de escrita utilizada anteriormente (ASCII) e aplicada no presente trabalho (HDF5/XDMF) foi realizada. Os resultados obtidos são apresentados na Tab. (1), diposta a seguir.

Tabela 1. Comparativo entre metodologias de escrita para códigos computacionais de alto desempenho.

Formato	Tempo de escrita (s)	Tamanho de arquivos (gb)
ASCII	291,4	8,1
HDF5/XDMF	0,048	1,9
Redução	99%	75%

Observa-se que o método de escrita apresentado no presente trabalho foi capaz de reduzir sensivelmente tanto o tempo necessário para a escrita das variáveis, quanto o tamanho dos arquivos gerados mostrando-se, assim, como uma interessante alternativa para códigos computacionais de alto desempenho.

4. CONCLUSÕES

Pôde-se concluir que o tempo de gravação, bem como o tamanho dos arquivos gerados com a utilização da metodologia estudada, foram significativamente menores quando comparados com o método de escrita utilizado anteriormente. Como consequência de tal trabalho, obteve-se uma sensível redução nos tempos de simulação e de pós processamento em simulações numéricas.

5. REFERÊNCIAS

The HDF group. Disponível em: <http://www.hdfgroup.org/HDF5>. Acesso em 21 fev. 2013.

Vedovoto, J.M., Silveira-Neto, A.d., Mura, A., Silva, L.F.F.d., 2011, ”Application of the Method of Manufactured Solutions to the Verification of a Pressure-Based Finite-Volume Numerical Scheme”, Computers & Fluids, Vol.51, pp. 85-99.

XDMFWeb. Disponível em: <http://www.xdmf.org/index.php/Main_Page>. Acesso em 21 fev. 2013.

6. RESPONSABILIDADE PELAS INFORMAÇÕES

Os autores são os únicos responsáveis pelas informações incluídas neste trabalho.

7. AGRADECIMENTOS

Os autores gostariam de agradecer à CAPES, PETROBRÁS, CNPq, FAPEMIG e à Faculdade de Engenharia Mecânica da Universidade Federal de Uberlândia por todo o apoio material e financeiro concedido.