Um Novo Olhar Sobre o Fortran: da Computação Sequencial à Paralela

Omar Andres Carmona Cortes

Departamento de Computação (DComp)
Instituto Federal de Educação, Ciência e Tecnologia do Maranhão (IFMA)
omar@ifma.edu.br

Resumo: A Linguagem Fortran foi uma das primeiras linguagens de programação tendo sido criada ainda na década de 50. Quase 60 anos depois, a linguagem continua sua evolução e não há indícios que sua história irá se encerrar. Nesse contexto, este artigo tem como objetivo apresentar como a linguagem tem evoluído e como suas características têm permitido que a linguagem continue em uso, iniciando com a computação sequencial, perpassando pela sua adaptação à computação paralela, chegando ao seu auge na programação de muitos núcleos em GPUs. Assim, a ideia do artigo é lançar uma nova perspectiva sobre a linguagem, em especial para o público de ciência da computação, fazendo com que seu real potencial seja visível a toda comunidade.

Palavras-chave: *Fortran; computação paralela; evolução.*

1. Introdução

O título deste artigo pode parecer estranho para muitas pessoas, em especial para aqueles da área de ciência da computação que provavelmente só ouviram falar da linguagem com uma parte da história da computação.

A verdade é que devido à pressão do mercado de trabalho, os cursos de graduação em computação (Ciência da Computação, Engenharia da Computação, Sistemas de Informação e Engenharia de Software), pelo menos no Brasil, tem que dar ênfase às linguagens mais utilizadas no mercado, tais como C, C++, Java, Julia, Python, dentre outras, e seus respectivos ambientes de programação. Não há nada de errado nisso já que o egresso certamente buscará trabalho na área. No entanto, algumas oportunidades podem ser perdidas. Por exemplo, a *Indeed* (https://www.indeed.com/), versão americana, que é uma página de busca por emprego, retorna 104 ofertas de emprego com a busca "*Fortran Programmer*".

O fato é que muito embora pessoas sempre fizessem predições declarando a morte da linguagem Fortran, ela continua sendo utilizada por engenheiros e cientistas [1], podendo ainda ser fonte de trabalho como já apresentado. No contexto científico alguns trabalhos podem ser citados. Por exemplo, Maan e Sing [2] convertem código Fortran para SymPy que executa operações matemáticas simbólicas. Hsu e Asanza [3] investigam como a otimização paralela afeta a portabilidade de programas em Fortran. Barros e Casquillo [4] apresentam como combinar CPLEX e Fortran 90 na resolução de problemas lineares. E, Johnson et al. [5] apresenta uma interface em Fortran 2003 para utilizar bibliotecas numéricas implementadas em C e C++.

Nesse contexto, este artigo investiga o estado da arte da linguagem Fortran de modo a desmistificar sua morte, mostrando que a linguagem ainda vive e com grandes perspectivas de continuar vivendo, especialmente em se tratando de computação paralela.

Assim, este artigo está dividido da seguinte maneira: a Seção 2 mostra a evolução histórica do Fortran e quais características a tornam interessante em cada versão; a Seção 3 apresenta o estado-da-arte em programação sequencial e em programação paralela; a Seção 4 mostra uma discussão sobre como evoluiu o estado da arte ; finalmente, a Seção 5 apresenta as conclusões desta pesquisa.

2. A Linguagem Fortran

Fortran é o acrônimo para *FORmula TRANslator*, sendo que o impulso para o seu nascimento foi o surgimento do IBM 704 em 1954. No mesmo ano foi publicado o relatório intitulado "*The IBM Mathematical FORmula TRANslating System: FORTRAN*", sendo que o primeiro compilador, chamado de Fortran I, foi lançado no ano de 1957 [6].

Seguindo sua evolução, em 1966 surgiu o Fortran IV tornando-se um padrão que ficou conhecido como Fortran 66. Anos depois, em 1978, foi lançada uma revisão conhecida como Fortran 77 tornando-se o padrão conhecido como ANSI Fortran.

Em 1990, o Fortran 77 foi anexado ao Fortran 90 tornando-se um subconjunto deste [7]. Algumas características consideradas obsoletas foram retiradas da linguagem e foram inseridas características que antes só poderiam ser utilizadas em linguagens como o C, como por exemplo: (i) operações diretas com matrizes; (ii) aumentou-se a facilidade de se trabalhar com a computação numérica; (iii) possibilitou-se a definição de novos tipos pelos usuários; (iv) foram adicionadas novas estruturas de controle de fluxo, possibilitando a programação mais estruturada; (v) foram adicionadas facilidades na definição de módulos e procedimentos; e, (vi) incluiu-se a recursão, a alocação dinâmica e o uso de ponteiros.

Interessantemente, a partir do Fortran 90 também foi possível emular os principais conceitos de orientação a objeto encontrados em C++ tais como classes, herança, encapsulamento e sobrecarga [8][9]. Essas características foram incorporadas efetivamente no Fortran 2003 [10], sendo que também foi incorporada a linguagem a interoperabilidade com a linguagem C [6]. Na versão 2008 foi dada uma perspectiva mais de computação paralela à linguagem introduzindo características de programação paralela diretamente na linguagem, ou seja, sem a necessidade de extensões.

Até o presente momento o padrão é o Fortran 2018 cujas melhorias foram o aumento da interoperabilidade com a linguagem C e adição de mais características de paralelismo. Espera-se que após 2020 seja publicada a revisão do padrão Fortran 2018. O novo padrão ainda chamado de 202x está previsto para ser lançado em 2021 [11].

A seguir apresenta-se o estado da arte da linguagem Fortran indicando como a linguagem evoluiu naturalmente da versão sequencial para a versão paralela.

3. Evolução do Fortran: da programação sequencial à paralela

Como já mencionado, a partir do Fortran 90, características antes restritas a programadores de Linguagem C ficaram disponíveis para programadores Fortran. Com esse padrão deu-se a possibilidade de se trabalhar com matrizes dinâmicas. Além disso, a produtividade também foi aumentada introduzindo operações matriciais do tipo C=*A*+*B*, na qual *A*, *B* e *C* são matrizes. Em outras palavras, em linguagens como C ou mesmo Fortran 77, por exemplo, esse tipo de operação tinha que ser feita através de laços embutidos, ou seja, iterando-se dentro das matrizes executando a operação elemento por elemento. A partir do Fortran 90 a operação ocorre em apenas uma linha.

A partir do padrão 2003 incorporou-se a programação orientada a objetos em Fortran. Na verdade, introduziu-se a extensão de tipos definidos, *type extension*, que acabou por permitir a utilização de uma das características mais importantes da programação orientada a objetos, a herança. No mesmo padrão também se introduziram a utilização de polimorfismo, interfaces e encapsulamento, dando a linguagem a possibilidade de programação orientada a objetos e seu principal benefício, o reuso.

No padrão de 2008 foi adicionado a linguagem o que se chama de *coarray*, que é uma estrutura de dados que pode ser compartilhada entre diferentes *images*, sendo *images* cópias idênticas do executável. Essa característica permite que a linguagem se aproveite da computação paralela através de uma técnica de paralelismo chamada SPMD, do inglês *Single Program Multiple Data*. Em outras palavras, o objetivo era tornar a programação paralela de fácil implementação, fazendo com que várias *images* pudessem trabalham em diferentes dados ou partes de um mesmo *coarray*, fornecendo ao programador a possibilidade de trabalhar com programação de memória compartilhada. Além disso, foi introduzida também a estrutura *DO...CONCURRENT* que especifica laços sem interdependência, que é particularmente interessante para compiladores de paralelização automática.

O Fortran 2018 é uma extensão menor do Fortran 2008 [12]. Como já mencionado, suas principais melhorias se limitam a estender o trabalho com *coarrays* e tornar a linguagem mais interoperável com a linguagem C. A principal modificação foi a adição de *Teams*. A ideia principal de *Teams* é que conjuntos separados de *images* possam ser executadas de forma independente [12], fazendo possível programar no modelo de programação paralela chamado de *Multiple Program Multiple Data* (MPMD).

A seguir serão detalhadas as extensões paralelas do Fortran desde as primeiras extensões paralelas para linguagens sequenciais até a utilização de GPUs.

Extensões Paralelas

A partir da década de 1980 houve o crescimento exponencial da informática, o que levou a quebra do paradigma de von *Neumman* fazendo surgir às primeiras arquiteturas paralelas. Assim, a década seguinte, de 1990, foi bastante promissora para a linguagem Fortran, sendo que nessa década surgiram as primeiras extensões paralelas de linguagens sequenciais: o *Parallel Virtual Machie* (PVM) [13] e o *Message Passing Interface* (MPI) [14]. Ambas as extensões permitiram ao programador Fortran a programação paralela através de um paradigma chamado de programação por passagem de mensagem, sendo adequado para arquiteturas com memória distribuída. A grande vantagem advinda do uso das extensões paralelas de linguagens sequenciais é a curva de aprendizado, já que o programador não precisa aprender uma linguagem do zero e sim estender seu conhecimento.

Em 1993 surge o High Performance Fortran (HPF) [15] cuja meta era fornecer a possibilidade de trabalhar com paralelismo de dados, provendo como principal vantagem a identificação e paralelização de dados de forma automática pelo compilador através da instrução *forall*, além de proporcionar o controle da distribuição dos dados pelo programador. O objetivo do HPF era um modelo de programação comum que pudesse ser executado eficientemente em todos os tipos de arquiteturas paralelas [16].

Em seguida lançou-se o HPF+ [17] em 1998, que permitia um maior controle sobre a distribuição dos dados, podendo inclusive especificar a carga de trabalho em laços irregulares. Além disso, o HPF+ permitia a utilização de parâmetros para especificar redundância de dados, facilitando o trabalho do compilador. Ainda na década de 90, por volta de 1995, surgiu o Vienna Fortran [18] cujo objetivo é estender o Fortran para que programadores pudessem escrever código para arquiteturas com memória distribuída. De fato, o compilador Vienna Fortran é um sistema *source-to-source* que transforma programas em Fortran 95/HPF+ para Fortran 90/MPI [19].

Mais ou menos ao mesmo tempo que se começaram a popularizar os processadores com múltiplos núcleos no início dos anos 2000, surgiu também o OpenMP [20], que é uma API para programação paralela com memória compartilhada. Coincidentemente ou não, as arquiteturas com múltiplos núcleo são arquiteturas com memória compartilhada.

Já na década de 2010 surge o CUDA Fortran [21], que acaba por ganhar mais popularidade dado as *General Purpose Graphic Processor Unit* (GPGPU), que são placas gráficas que podem ser utilizadas para computação de propósito geral. CUDA Fortran é

essencialmente um Fortran 90 com algumas extensões que permite ao programador aproveitar o poder das GPUs em suas aplicações [22]. Finalmente, surge o OpenACC [23] que, de forma similar ao OpenMP, é uma biblioteca de diretivas que permite a paralelização automática de código em Fortran em GPUs.

4. Discussão

Até o Fortran 77, as operações com vetores e matrizes eram feitas de forma similar a outras linguagens, ou seja, dentro de laços. A partir do Fortran 90 as operações passaram a ser diretas, inclusive incluindo a capacidade de se trabalhar com subconjuntos dentro de uma mesma matriz ou vetor. Essa característica foi bastante visionária e viria a aparecer em linguagens como R, Julia e Python (usando *NumPy*), que atualmente são as principais linguagens *open source* para ciência dos dados e aprendizagem de máquina. Obviamente essas linguagens são mais produtivas que Fortran. Porém, devido à grande quantidade de código disponível para supercomputadores ainda estar em Fortran e de ter usuários para esses códigos, a possibilidade dessas novas linguagens substituírem o Fortran é pequena em curto e médio prazo.

Com o surgimento das arquiteturas paralelas, como já foi mencionado, Fortran foi uma das primeiras linguagens a receber extensões paralelas de linguagens sequenciais: PVM e MPI. Dessas duas extensões o MPI foi a que permaneceu.

De fato, a década de 90 foi rica em soluções para computação paralela, sendo uma delas o HPF, que apesar de inicialmente promissor se mostrou ineficiente. O problema com o HPF era que estruturas de dados irregulares podiam gerar uma sobrecarga na comunicação entre processos ou gerar um desbalanceamento de carga que inevitavelmente leva a um desempenho pobre da aplicação paralela [24]. Para solucionar esse problema foi criado o HPF+ e em seguida o Vienna Fortran para flexibilizar a programação paralela dada a quantidade de arquiteturas paralelas que vinham surgindo na época.

Os problemas criados pelas versões de HPF e Viena Fortran acabaram por ficar para trás quando surgiu o OpenMP, sendo que seu ponto forte é ser baseado em diretivas de compilação (*#pragma*), ou seja, em instruções especiais que indicam ao compilador quais regiões do programa devem ser paralelizadas pelo compilador. Assim, o programador também não tem que aprender uma nova linguagem e sim onde colocar as diretivas e algumas funções novas, diminuindo consideravelmente a curva de aprendizagem.

Ao mesmo tempo que soluções paralelas iam sendo desenvolvidas, a própria linguagem ia tendo sua capacidade de executar código paralelo sendo melhorada a cada versão. No Fortran 2018, por exemplo, *images* podem se comunicar em uma única direção, isto é, se uma chamada e feita da *image* A para a *image* B, não é necessário ter uma chamada correspondente em B para A como deve ser feito em MPI, o que simplifica consideravelmente a programação. Assim, possivelmente, devido às melhorias e facilidades de programação paralela nativa do Fortran aliado à facilidade disponibilizada pelo OpenMP na programação de memória compartilhada, levaram ao abandono das outras iniciativas como o HPF.

Por outro lado, o MPI permaneceu no mercado. Acredita-se que o MPI persistiu devido ao controle do paralelismo que a biblioteca fornece ao programador em sistemas de memória distribuída com passagem de mensagem. Inclusive o *gcc* [25], que é o compilador mais conhecido no mundo *Unix-alike*, tem suporte tanto para linguagem Fortran quanto para OpenMP sem a necessidade de instalação de bibliotecas adicionais. A partir da versão 8, o *gcc* possui inclusive suporte ao OpenACC para paralelização automática em GPU.

Para finalizar tem-se o CUDA Fortran, cuja capacidade de paralelismo ainda tem muito a ser explorado por dois motivos. O primeiro pela popularidade das GPGPUs que hoje estão disponíveis até em dispositivos móveis como celulares e o segundo pela construção dos supercomputadores baseados em GPUs. A Tabela 1 apresenta os seis supercomputadores baseados em GPU e sua posição no Top 500 [26] dos supercomputadores em junho de 2020.

Tabela 1 – Supercomputadores baseados em GPU.

Nome	Posição	Núcleos CUDA
Summit	2º	2.414.592
Sierra	3º	1.572.489
HPC5	6º	669.769
Selene	7º	272.800
Marconi-100	9º	347.776
Piz Daint	10º	387.872

Como se pode observar, dentre os 10 computadores mais rápidos do mundo, seis são baseados em GPU. Além dessa evolução do Fortran, a quantidade de código e bibliotecas científicas disponíveis na linguagem ainda a fazem atrativa tanto na computação científica quanto na engenharia e na supercomputação. Em outras palavras, ainda existe um poder computacional significativo a ser explorado, seja pela capacidade de computação paralela nativa da linguagem, seja pelo OpenMP, pelo MPI, pelo CUDA Fortran ou pelo OpenACC.

5. Conclusões

Este artigo apresentou uma evolução histórica do estado da arte da linguagem Fortran. Mostraram-se as características que foram evoluídas de modo a passar naturalmente da computação sequencial para computação paralela, sendo que o auge de novas tecnologias para a linguagem Fortran foi a década de 90, na qual surgiram tecnologias como o MPI, HPF e Vienna Fortran.

Atualmente a computação paralela em Fortran está focada na utilização de cinco tecnologias: computação paralela nativa do Fortran 2018, OpenMP, MPI, OpenACC e CUDA Fortran. Dentre essas destaca-se a possibilidade de exploração de OpenACC e CUDA Fortran já que, dentre as arquiteturas paralelas mais rápidas do mundo, seis são baseadas em GPU. Em outras palavras, Fortran ainda tem um longo caminho pela frente.

Bibliografia

[1] Kolakowski, N. (2019) Five Programming Languages that Refuse to Die, Dice, https://insights.dice.com/2019/10/03/5-programming-languages-refuse-die/, Acesso em 16/08/202.

[2] Maan, N.; Singh, P. (2020) Parsing C and Fortran code to SymPy Expressions, 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence), Noida, India, 2020, pp. 592-597, DOI: 10.1109/Confluence47617.2020.9057820.

[3] Hsu, A.; Asanza, D. N. ; Schoonover, J. A.; Jibben, Z.; Carlson, N. N.; Robey, R. (2018) Performance Portability Challenges for Fortran Applications, IEEE/ACM International Workshop on Performance, Portability and Productivity in HPC (P3HPC), Dallas, TX, USA, 2018, pp. 47-58, DOI: 10.1109/P3HPC.2018.00008.

[4] Barros, M.; Casquilho, M. (2019) Linear Programming with CPLEX: An Illustrative Application Over the Internet CPLEX in Fortran 90, 14th Iberian Conference on Information Systems and Technologies (CISTI), Coimbra, Portugal, pp. 1-6, DOI: 10.23919/CISTI.2019.8760632.

[5] Johnson, S. R.; Prokopenko, A.; Evans, K. J. (2020) Automated Fortran–C++ Bindings for Large-Scale Scientific Applications, *Computing in Science & Engineering*, vol. 22, no. 5, pp. 84-94, DOI: 10.1109/MCSE.2019.2924204.

[6] Sebesta, R. W. (2018) Conceitos de Linguagens de Programação, 11ª edição, Porto Alegre: Bookman.

[7] Reid, J. (1992) The advantages of Fortran 90, *Computing*, 48:219-238.

[8] Decyk, V., K.; Norton, C., D.; Szymansky. B., K. (1999) How to Express C++ Concepts in Fortran 90, Jet Propulsion Laboratory – California Institute of Tecnology, Passadena: California, Rensisselear Polytechnic Institute, Troy: N.Y.

[9] Decyk, V. K.; Norton, C. D..; Szymansky. B., K. (1999) Introduction to Object-Oriented Concepts Using Fortran, Jet Propulsion Laboratory – California Institute of Tecnology, Passadena: California AND Departament of Computer Science – Rensisselear Polytechnic Institute, Troy: N.Y.

[10] Metcalf, M.; Reid, J.; Cohen, M. (2004) Fortran 95/2003 Explained, 3e. Oxford University Press, Oxford, England.

[11] ISO, JTC1/SC22/WG5, Fortran 202x, https://wg5-fortran.org/f202x.html, Acesso em 16/08/2020.

[12] Metcalf, M.; Reid, J. and Cohen, M. (2018) Modern Fortran Explained, Oxford Press: UK.

[13] Index for PVM3 Library, http://www.netlib.org/pvm3/, Acesso em 16/08/2020

[14] Open MPI: Open Source High Performance ComputingHPF, https://www.open-mpi.org/, Acesso em 15/08/2020

[15] High Performance Fortran, http://hpff.rice.edu/index.htm, Acesso em 10/08/2020.

[16] Kennedy, K.; Koelbel, C.; Zima, H. (2007) The rise and fall of High Performance Fortran: an historical object lesson. In Proceedings of the third ACM SIGPLAN conference on History of programming languages (HOPL III). Association for Computing Machinery, New York, NY, USA, DOI: 10.1145/1238844.1238851.

[17] Optimizing HPF for Advanced Applications, http://www.par.univie.ac.at/project/hpf+/, Acesso em 01/08/2020.

[18] VFC, http://www.par.univie.ac.at/project/hpf+/vfcs.html, Acesso em 10/08/2020

[19] Benker, S. (1999) The Vienna Fortran Compiler, Scientific Programming, IOS Press, pp. 1058-9244.

[20] OpenMP: Enabling HPC since 1997, https://www.openmp.org/, Acesso em 10/08/2020

[21] Cuda Fortran, https://developer.nvidia.com/cuda-fortran, Acesso em 16/08/2020

[22] Ruetsch, G.; Fatica, M. (2014) CUDA Fortran for Scientistis and Engineers: Best Practices for Efficient CUDA Fortran Programming, Morgan Kaufman: San Francisco-USA.

[23] OpenACC: more science less programming, https://www.openacc.org/, Acesso 05/07/2020

[24] Ryne, R. D.; Habib, S. (1996) Beam Dynamics Calculation and Particle Tracking Using Massively Parallel Processors, Particle Accelerators, vol. 55, pp. 365-374.

[25] GCC the GNU Compiler, https://gcc.gnu.org/, Acesso em 01/08/2020

[26] Top 500, Lista Junho de 2020, https://www.top500.org/lists/top500/2020/06/, Acesso em 16/08/2020.