REVISTA Mundi
Engenharia, Tecnologia e Gestão
INSTITUTO FEDERAL Paraná

APRIMORAMENTO DO CÓDIGO CYRANO DE SIMULAÇÃO DE AQUECIMENTO DE PLASMA POR ONDAS RF UTILIZANDO TÉCNICAS DE PROCESSAMENTO PARALELO

IMPROVEMENT OF THE CYRANO CODE FOR PLASMA HEATING SIMULATION BY RF WAVES USING PARALLEL PROCESSING TECHNIQUES

Anusio Menezes Correia[1]

Ernesto Augusto Lerche [2]

Dirk Van Eester [3]

Gesil Sampaio Amarante Segundo [4]

Esbel Tomas Valero Orellana [5]

Resumo: A reação de fusão termonuclear controlada se apresenta como uma fonte potencialmente relevante alternativa de energia, ainda em desenvolvimento. O estudo de reatores de fusão utiliza intensivamente modelos computacionais e, entre os códigos utilizados para esta finalidade destaca-se o CYRANO, que simula o aquecimento do plasma por ondas de RF e tem como limitação o elevado consumo de recursos computacionais. Neste trabalho é proposta a utilização de técnicas de processamento paralelo para diminuir o seu tempo de processamento. Com esta finalidade apresentamos modelo utilizado e uma análise do desempenho da implementação original, o que permitiu destacar as rotinas que concentram a maior parte do tempo de processamento. A pesar de ter uma importância maior em execuções menores, a função OUTPOW perde em relevância para a rotina GENERA em problemas de grande porte. Desta forma, numa primeira abordagem, foi feita uma substituição das funções mais utilizadas nesta rotina, por versões otimizadas e paralelizadas com OpenMP. O algoritmo da OUTPOW foi reescrito e o laço principal paralelizado por meio de diretivas de OpenMP. Os resultados obtidos mostram uma melhora significativa no desempenho geral. A continuidade deste trabalho deve visar problemas que demandam maior consumo de memória, sobretudo envolvendo a sub-rotina GENERA3.

Palavras-chave: CYRANO, Plasma, Tokamak, OpenMP, OpenBLAS.

Abstract: The reaction of thermonuclear controlled fusion is presented as a high potential alternative source of energy, but still in development. The study of fusion reactors use intensively computational models and among the codes used for that goal CYRANO stands out.

[1] Bacharel, Universidade Estadual de Santa Cruz (UESC), amcorreia@uesc.br.
[2] Dr, Laboratory for Plasma Physics, Ecole Royale Militaire, E.Lerche@rma.ac.be.
[3] Dr, Laboratory for Plasma Physics, Ecole Royale Militaire, d.van.eester@fz-juelich.de.
[4] Dr, Universidade Estadual de Santa Cruz (UESC), gsamarante@uesc.br.
[5] Dr, Universidade Estadual de Santa Cruz (UESC), evalero@uesc.br.

CYRANO Code simulates the plasma heating by RF waves but has as a limitation the high consumption of computational resources. In this paper we propose the use of parallel processing techniques to decrease its processing time. We present the model used and a performance analysis of the original implementation, which allowed to highlight the routines that concentrate the most of the processing time. Even though having the major importance in smaller executions, the OUTPOW function loses in relevance for the GENERA3 routine in larger problems. That way, as a first approach, we made a replacement of most used functions in that subroutine by optimized and parallelized versions with OpenMP. The algorithm of OUTPOW was then rewritten and the main loop parallelized using OpenMP directives. The obtained results showed a meaningful improvement in the overall performance. The continuity of this work should focus on problems that require bigger memory consumption, principally involving the subroutine GENERA3.

Keywords: CYRANO, Plasma, Tokamak, OpenMP, OpenBLAS.

1 INTRODUÇÃO

Nas últimas décadas temos assistido a aumento exponencial da demanda por energia (U.S. ENERGY INFORMATION ADMINISTRATION, 2016). As principais fontes de energia utilizadas na atualidade são, também, geradoras de poluição ambiental das mais diversas formas. Fontes de energia alternativas e limpas, são ainda caras e ineficientes e muitas pesquisas são desenvolvidas na atualidade para seu melhoramento. Pesquisadores pelo mundo trabalham no desenvolvimento de formas eficientes de gerar energia, sem agredir o ambiente e que sejam economicamente viáveis.

Uma alternativa promissora é o uso de reações de fusão para gerar energia. A reação de fusão ocorre quando átomos leves se fundem formando átomos mais pesados, liberando grande quantidade de energia no processo. Para que este tipo de reação aconteça átomos de núcleos leves, como hidrogênio e hélio, podem ser aquecidos a aproximadamente 150.000.000 Kelvin, em confinamento magnético numa câmara com formato de toroide. Este aquecimento pode ser feito da por ondas eletromagnéticas emitidas por uma antena ICRH (*Ion Cyclotron Radio Frequency Heating*), como pretende o projeto de um reator de fusão conhecido como *Tokamak* (LAWSON, 1957).

*Tokamak*s são protótipo de reatores de fusão nuclear, pensados com o propósito de gerar energia a partir da fusão de átomos, mais precisamente por isótopos de hidrogênio como deutério e trítio. O *Tokamak* teve sua viabilidade técnica e cientifica comprovada através de experiências realizadas em *Tokamaks* de grande porte, como indica o estudo realizado por (SMITH; COWLEY, 2010).

Atualmente, o principal projeto que visa demonstrar a capacidade de geração de energia através de *Tokamak*s de forma sustentável é o ITER (*International Thermonuclear Experimental Reactor*) (CLERY, 2015). O alto custo envolvido no desenvolvimento e operação destes dispositivos, unido à sua complexidade técnica de operação fazem dos códigos de modelagem computacional uma importante ferramenta de tomada de decisão.

Um dos vários tipos de código utilizados na modelagem física de aspectos importantes da operação de um *Tokamak*, particularmente o aquecimento dos plasmas (gases ionizados) por meio de ondas de radiofreqüência (RF) inclui os chamados códigos de onda completa (*full wave codes*) (BILATO et al., 2015). Estas ferramentas simulam a propagação de ondas de RF que estão dentro do espectro de frequência íon-ciclotrônicas e a absorção de sua energia por plasmas de isótopos de hidrogênio confinados por um intenso campo magnético *Tokamak*.

Utilizados recentemente em *benchmaks* do atual projeto ITER (BILATO et al., 2015), quatro códigos de onda completa se destacam. São eles o TORIC, EVE, LION e o CYRANO, estes códigos fazem parte do *EUROfusion Code Development for Integrated Modelling project* (WPCD), e utilizam métodos diferentes em suas modelagens. Neste artigo abordamos as características do CYRANO.

Desenvolvido por Philippe Lamalle na tese (LAMALLE, 1994), o código CYRANO resolve o sistema de equações conhecidas como equações de Maxwell–Vlasov, isto em uma configuração toroidal de plasma que possui seção transversal poloidal na faixa de frequência ciclotrônica iônica utilizando a formulação fraca de Galerkin. Esta formulação é explicada com mais detalhas em (LAMALLE, 1997). De forma simples a formulação pode ser representada através da equação:

$$W(\mathbf{F},\mathbf{E}) + iR(\mathbf{F},\mathbf{E}) = A(\mathbf{F},\mathbf{j}_S) \tag{1}$$

onde:

- ***W*** da Equação (1) e (2) representa a contribuição do plasma e a contribuição parcial de cada partícula;

$$W(\mathbf{F},\mathbf{E}) = \sum_{\beta}\left(\frac{q_\beta}{2}\int_V dr^3 \int dv^3 f_\beta \mathbf{F}^* \cdot v\right) \tag{2}$$

- ***R*** da Equação (1) e (3) representa a contribuição do operador de onda no vácuo;

$$R(\mathbf{F},\mathbf{E}) = \frac{1}{2}\int_V \left[\frac{1}{\omega\mu_0}\nabla\times\mathbf{F}^*\cdot\nabla\times\mathbf{E} - \omega\varepsilon_0\mathbf{F}^*\cdot\mathbf{E}\right]dr^3 \tag{3}$$

- ***A*** da Equação (1) e (4) representa a contribuição das fontes do sistema (Antena RF);

$$A(\mathbf{F}, \mathbf{j}_S) = -\frac{1}{2}\int_V \mathbf{F}^* \cdot \mathbf{j}_S dr^3 \qquad (4)$$

Escrito na linguagem FORTRAN77, o CYRANO é um código complexo que possui muitas rotinas e variáveis compartilhadas. Dentre as funções utilizadas cabe destacar algumas, como a ZAXPY e a ZGEMM, que fazem parte da biblioteca *Basic Linear Algebra Subprograms* (BLAS). O CYRANO inclui os códigos-fonte destas rotinas, disponíveis no endereço eletrônico http://www.netlib.org/blas/, e podem ser compiladas junto com as outras funções do sistema.

Para carregar os dados iniciais ou parâmetros de entrada do CYRANO, o mesmo utiliza o padrão NAMELIST (padrão da linguagem de programação FORTRAN77, de entrada e saída com formatação livre para um conjunto ou grupo de variáveis). Dos valores carregados para uma simulação pelo CYRANO cabe destacar os parâmetros “iele”, do conjunto “NAMSUB”, que define a quantidade de elementos radiais a serem calculados, “modva1” e “modva2”' do conjunto “NAMANT”, que define os modos poloidais considerados pela simulação e “klim”' do conjunto “NAMSYS”, que representa o número de acoplamentos poloidais considerados. A complexidade da simulação e, portanto o consumo de recursos pelo programa está diretamente ligada a estes parâmetros.

A realização de estudos mais complexos com a utilização do CYRANO esbarra em limitações impostas pelo alto consumo de memória e de tempo de processamento. Neste artigo, apresentamos os resultados obtidos com a utilização de técnicas de processamento paralelo em arquitetura de memória distribuída, com o objetivo de reduzir o tempo de execução das simulações.

Mantido pelo grupo KHRONOS, OpenMP (*Open Multi-Processing*) é uma API (*application programming interface*) que suporta processamento de memória compartilhada, através de diretivas de pré processamento simples, permitindo que sem muita alteração no código, seja possível aplicar áreas que serão executadas em múltiplas *threads* simultaneamente a uma grande

variedades linguagens dentre as quais estão C/C++, FORTRAN ou mesmo JAVA.

Nas seções a seguir, apresentamos brevemente a infra-estrutura computacional utilizada para este estudo. Posteriormente, mostramos a metodologia empregada para determinar os principais gargalos no código visando implementar uma paralelização incremental do mesmo. Apresentamos também a estratégia proposta para introduzir paralelismo via utilização de diretivas de OpenMP. Finalmente, os resultados obtidos são discutidos e apresentados as conclusões e recomendações para trabalhos futuros.

2 PLATAFORMA COMPUTACIONAL UTILIZADA

Para realizar os estudos de desempenho do código e os desenvolvimentos posteriores, foi utilizado um conjunto de ferramentas disponíveis sob diversas licenças de software livre. Para compilar o código foi utilizado o Fortran GFORTRAN (GNU versão 5.4.0). Visando realizar uma análise mais detalhada do código, foi utilizada a ferramenta GPROF (versão 2.20.51.0.2-5.34.el6) em conjunto com a ferramenta GPROF2dot, que é um script em Python que permite converter a saída do GPROF em um gráfico do tipo “dot”, disponível em https://github.com/jrfonseca/GPROF2dot.

Os testes foram realizados no CACAU (Centro de Armazenamento de Dados e computação avançada da UESC - http://nbcgib.uesc.br/cacau), cujo hardware está organizado em filas de execução gerenciadas com SLURM (versão 2.5.0-Bull.6.2). As características dos nós de computo utilizados são descritos na Tabela 1.

Para obter melhoras no desempenho do CYRANO foram utilizadas também implementações mais eficientes da BLAS, em conjunto com o código original. Foi comparado o desempenho da aplicação com a BLAS, a ATLAS e OpenBLAS. A seguir é apresentada a estratégia utilizada na análise e paralelização do código.

3 METODOLOGIA

Tabela 1 – Descrição do hardware do CACAU.

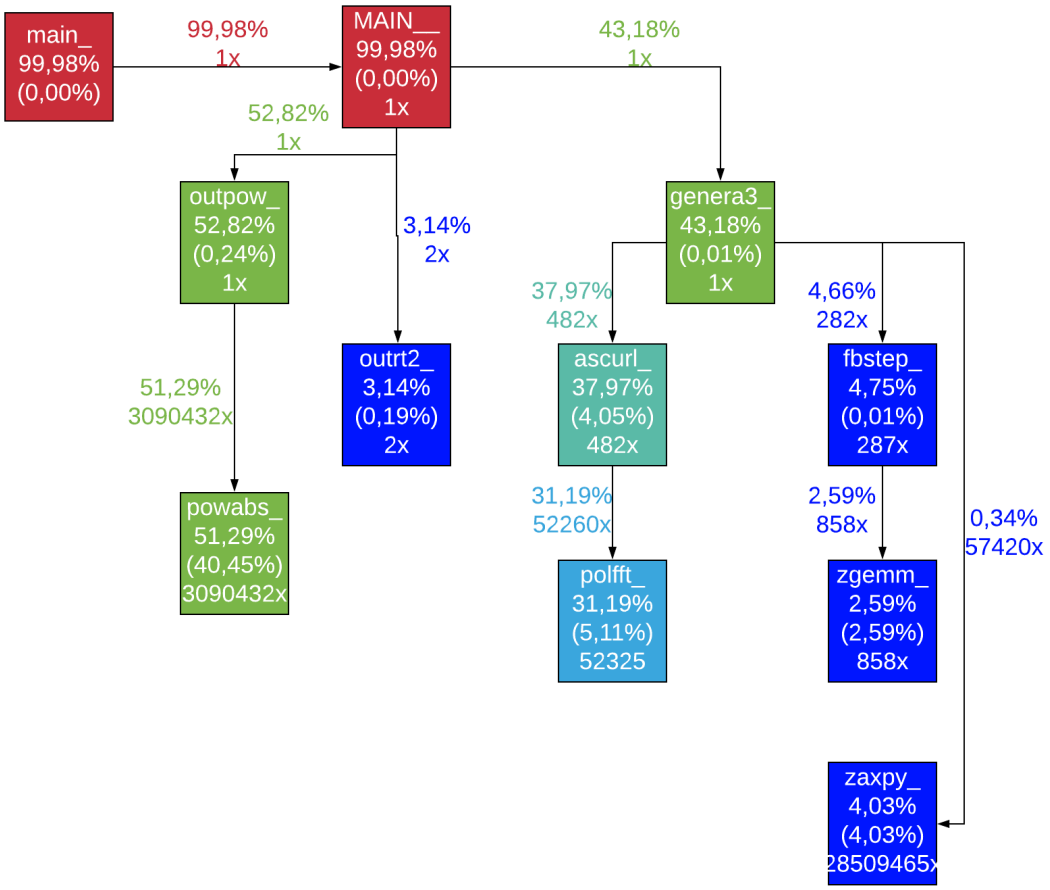
	“Fila gpu”
Processador	2x Intel(R) Xeon(R) CPU E5-2440 @ 2.40GHz
Memória RAM	6x 8GB DIMM DDR3 Synchronous 1600 MHz (48 GB)
SO	Linux Server release 6.3
Versão do Kernel	2.6.32-279.el6.x86_64

Fonte: http://nbcgib.uesc.br/cacau.

Com base nas características do modelo, definimos um conjunto de casos de estudo para analisar o desempenho da aplicação na medida em que aumentamos o tamanho da simulação computacional. Desta forma, foram definidos dez casos de testes para avaliar o desempenho do código, cinco com 280 elementos e cinco com 480 elementos radiais, cada um com 32, 64, 128, 256 e 512 modos poloidais considerados. Os modos poloidais variam de -16 a 16, -32 a 32, -64 a 64, -128 a 128 e -256 a 256 acoplamentos poloidais. Os diferentes casos foram executados nas duas filas do CACAU com ajuda da ferramenta GPROF para *profiling* de código, visando determinar as rotinas com maior custo computacional. Para processar os resultados do GPROF foi utilizado o script GPROF2dot, que apresenta o resultado do perfil em um gráfico. Os resultados obtidos com esta ferramenta mostram que as rotinas GENERA3 e OUTPOW ocupam a maior parte do tempo de processamento. Nas Figuras 1 e 2 são apresentados os casos para 32 e 512 modos poloidais, ambas com 280 elementos radiais. Desta forma foi possível determinar que para simulações de pequeno porte, Figura 1, a função OUTPOW tem grande importância no tempo total. Entretanto esta relevância decai na medida em que se trabalha com simulações maiores, Figura 2, e nestes casos GENERA3 passa a representar uma parcela maior do tempo total.

Analisando a Figura 1, é possível concluir que as chamadas a POWABS representam quase 100% do tempo de OUTPOW, que é uma rotina de pós-processamento, encarregada de gerar vários arquivos de saída do CYRANO. Da análise da Figura 2 também foi possível extrair o grande número de chamadas que se faz, dentro de GENERA3, ás funções ZGEMM e ZAXPY. No código, GENERA3 é a função responsável por montar e resolver o método Galerkin.

Figura 1 – Gráfico gerado a partir dos resultados do GPROF para o código CYRANO, com ênfase nas rotinas importantes, utilizando um caso com 32 modos poloidais e 280 elementos. Em destaque as funções que ocupam maior porcentagem do tempo de execução, a função OUTPOW com 52,3% e GENERA3 com 42,1%. No casso de OUTPOW a maior parte do processamento se concentra POWABS com 51,29%.

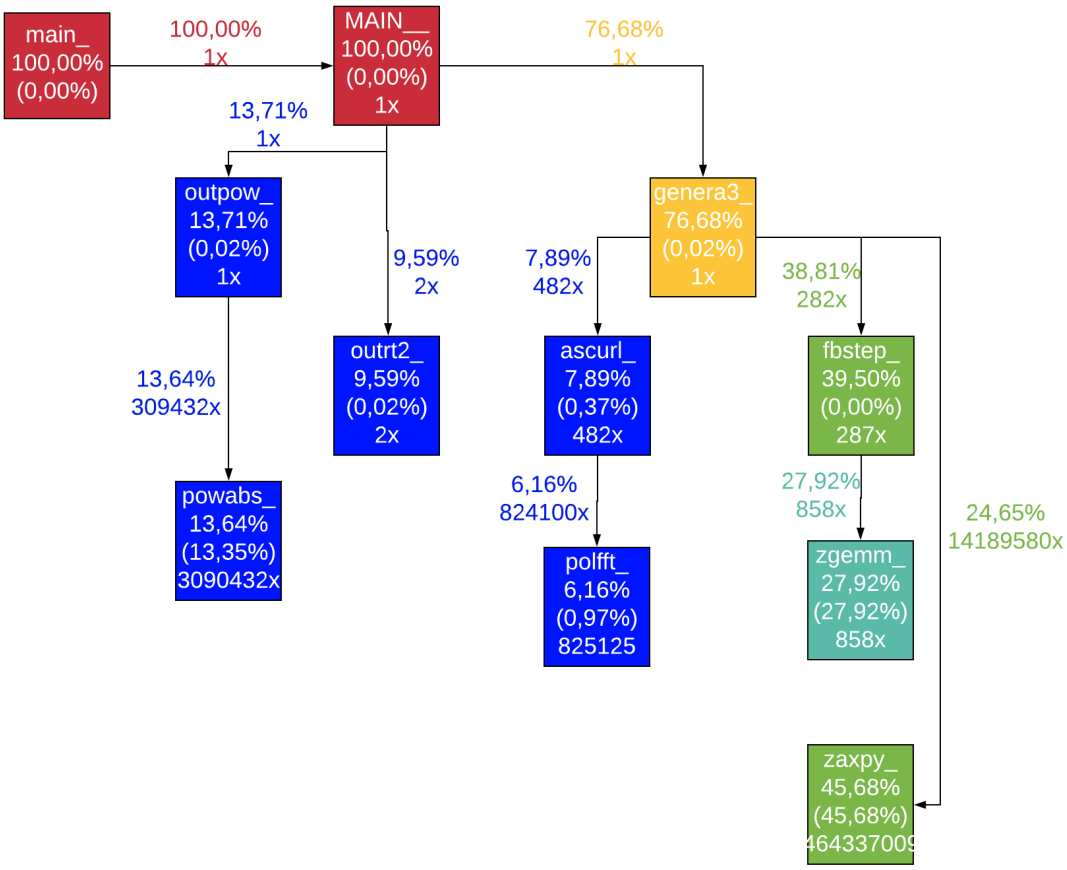


Os resultados obtidos com GPROF foram validados introduzindo rotinas para quantificar o tempo total e o tempo gasto pelas duas rotinas principais. Os resultados obtidos para os casos que utilizam 280 e 480 elementos radiais, variando a quantidade de modos poloidais, são apresentados na Figura 3. Pode se observar que na medida em que aumenta a quantidade de modos poloidais ocorre uma inversão no consumo de tempo entre a GENERA3 e OUTPOW. A partir destas análises iniciais foi definida a seguinte estratégia para diminuir o tempo de processamento:

- Substituir as funções ZGEMM e ZAXPY disponíveis no código por versões otimizadas implementadas em bibliotecas como a ATLAS e a OpenBLAS. No caso da OpenBLAS obtemos por uma versão paralelizada para arquiteturas de memória compartilhada;

- Reestruturar o código da OUTPOW, especialmente a POWABS, otimizando o uso das variáveis locais visando facilitar a introdução de diretivas de compilação de OpenMP;

Figura 2 – Gráfico gerado a partir dos resultados do GPROF para o código CYRANO, com ênfase nas rotinas importantes, utilizando um caso com 512 modos poloidais e 280 elementos. Em destaque as funções que ocupam maior porcentagem do tempo de execução, a função GENERA3 com 76,68% e OUTPOW com 13,71%. No casso de GENERA3 se destacam as funções ZAXPY com 45,68% e ZGEMM com 27,92%.

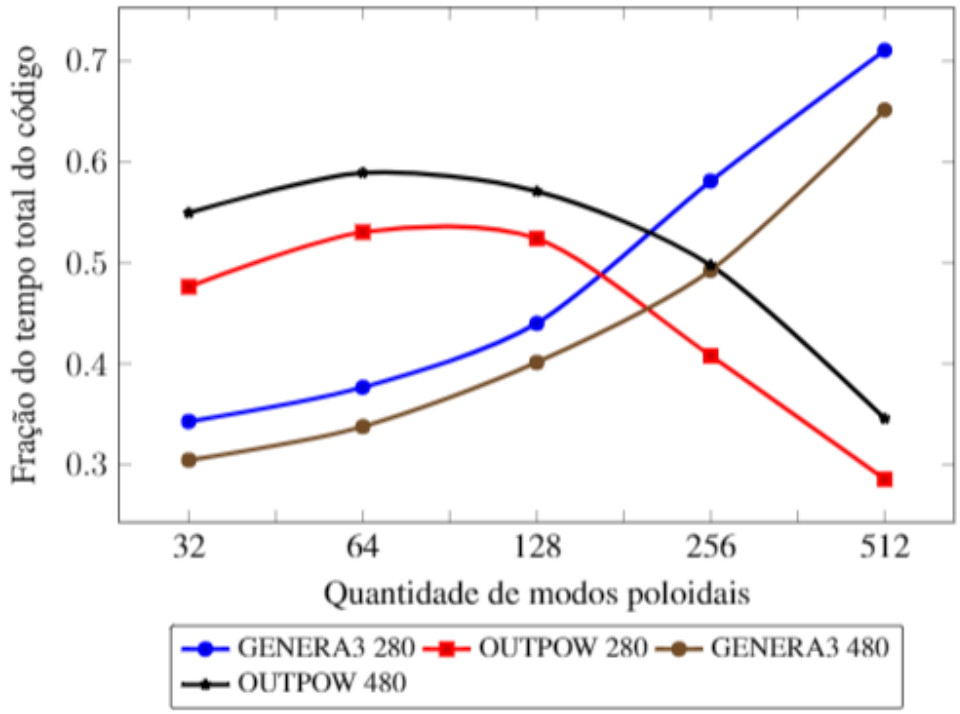


4 RESULTADOS E DISCURSSÕES

Implementadas as modificações no código, foram realizados os testes com duas versões do mesmo e os diferentes casos de estudo definidos anteriormente. Na primeira versão foi utilizada uma implementação otimizada da biblioteca BLAS, a ATLAS. A segunda versão foi compilada com OpenBLAS uma versão otimizada e paralelizada com OpenMP. Em ambos os casos foi utilizada a versão de OUTPOW melhorada e paralelizada com a introdução de

diretivas de compilação OpenMP no laço principal. Todos os testes foram comparados, em termos de desempenho e resultado da simulação, com o código original do CYRANO.

Figura 3 – Relação do consumo do tempo das funções GENERA3 e OUTPOW em relação ao tempo total, com 280 e 480 elementos. Código original.



Feitos os testes com os diferentes casos, pode ser analisando o tempo total e o gasto pelas funções GENERA3 e OUTPOW. A Tabela 2 mostra os tempos e os resultados de *speedup* obtidos nos casos de 280 elementos.

No caso da função OUTPOW, considerando que foram utilizados 12 *threads*, os resultados se mantiveram constantes, em acima de 12. Este comportamento supra linear, pois o *speedup* ultrapassou a quantidade de *threads*, se deu pela otimização feita na rotina. As modificações, necessárias na paralelização do código, se deram no código da função POWABS retirado um conjunto de operações, que se repetiam desnecessariamente, para fora do laço e assim, otimizando o uso das variáveis locais. Obteve-se desta forma, uma versão mais eficiente de OUTPOW, com desempenho melhor mesmo sem ativar as diretivas OpenMP.

Tabela 2 – Tempo em segundos, gasto pelo código nas rotinas OUTPOW, GENERA3 e o tempo total, medido no cacau, fila gpu, caso com 280 elementos radiais.

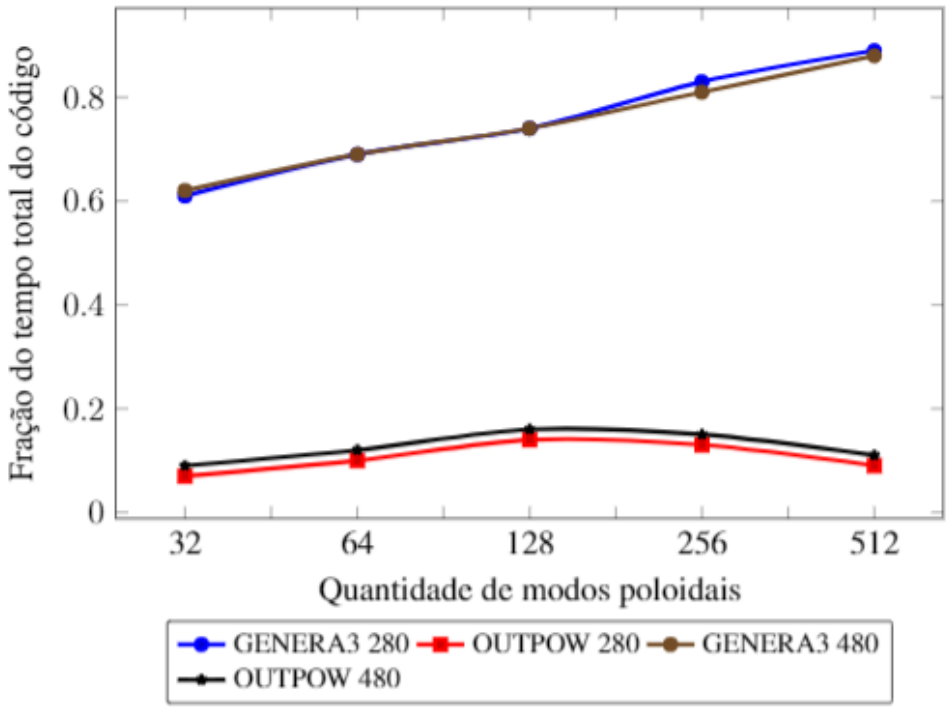
Casos em modos	Original	OpenBLAS	Atlas	*Speedup* OpenBLAS	*Speedup* Atlas
GENERA3					
32	189,44	190,89	186,68	0,99	1,01
64	455,25	367,19	411,02	1,24	1,11
128	1.705,80	856,19	1.590,59	1,99	1,07
256	10.518,28	3.786,95	8.068,04	2,78	1,30
512	69.938,47	21.470,46	51.242,83	3,26	1,36
OUTPOW					
32	263,27	21,76	21,95	12,10	12,00
64	641,19	51,62	51,34	12,42	12,49
128	2.031,61	158,95	159,64	12,78	12,73
256	7.382,96	580,04	579,86	12,73	12,73
512	28.098,32	2.210,14	2.210,64	12,71	12,71
TOTAL					
32	552,76	312,55	310,35	1,77	1,78
64	1.208,78	530,91	575,43	2,28	2,10
128	3.876,20	1150,37	1.889,50	3,37	2,05
256	18.102,27	4.558,44	8.847,58	3,97	2,05
512	98.418,68	24.049,36	53.839,58	4,09	1,83

Já a função GENERA3 pode-se observar um ganho de desempenho com *speedup* de até 3,26, no caso de 512 modos poloidais para a versão baseada em OpenBLAS. A utilização da versão sequencial otimizada da biblioteca ATLAS não trouxe diferença significativa no desempenho, mas mostra o potencial de se trabalhar na melhora das funções da biblioteca BLAS.

Em relação ao tempo total de execução pode-se observar que acompanha, principalmente, os ganhos obtidos pela função GENERA3. Conseguindo resultados que chegam a *speedup* de 4,09 para caso de 512 modos poloidais a implementação baseada em OpenBLAS tem um desempenho superior em todos os casos.

Finalmente apresentamos o gráfico que mostra a fração do tempo total que representa o tempo das funções GENERA3 e OUTPOW na Figura 4. Desta vez, com a OpenBLAS e a OUTPOW modificada com OpenMP, observa-se uma relevância ainda maior no tempo gasto com GENERA3, chegando a representar quase 90% do tempo, total no caso mais complexo. A função OUTPOW consome uma fração quase constante do tempo total, independentemente do tamanho do problema.

Figura 4 – Gráfico do consumo do tempo pelas funções GENERA3 e OUTPOW em relação ao tempo total, com 280 e 480 elementos. Código com OUTPOW OpenMP e OpenBLAS.



5 CONCLUSÂO

Neste artigo, apresentamos uma versão do código cientifico CYRANO, apresentando redução do tempo de execução, em comparação com a versão serial. Para isto, utilizamos técnicas de paralelismo com OpenMP além de versões melhoradas da biblioteca BLAS como a ATLAS e a OpenBLAS.

Embora OUTPOW tenha obtido uma redução significativa no tempo de execução, o melhor *speedup* obtido ficou próximo ao obtido na sub-rotina GENERA3, o que era esperado, já que esta última é a sub-rotina que utiliza mais tempo de computo. Apenas utilizando OpenMP e a OpenBLAS com OpenMP, reduzimos o tempo de 98.418 segundos para 24.049 segundos, ou seja, de aproximadamente um dia para aproximadamente seis horas, o que representa um ganho de tempo relevante. Mostrando não somente que OpenMP é uma ferramenta de paralelização adequada a este código, e exemplifica o potencial de melhoria trazida pela atualização da biblioteca BLAS. Os resultados obtidos mostram que ainda existe espaço para paralelização,

utilizando memória distribuída, para garantir uma redução ainda maior do tempo e possibilitar a execução de modelos mais complexos que requeiram maior quantidade de memória.

REFERÊNCIAS

BILATO, R.; BERTELLI, N.; BRAMBILLA, M.; DUMONT, R.; JAEGER, E. F.; JOHNSON, T.; LERCHE, E.; SAUTER, O.; VAN EESTER, D.; VILLARD, L. Status of the benchmark activity of ICRF full-wave codes within EUROfusion WPCD and beyond. In: AIP Conference Proceedings, 633053, **Anais**...2015.

CLERY, D. FUSION ENERGY. More delays for ITER fusion project. **Science (New York, N.Y.)**, v. 350, n. 6264, p. 1011, 27 nov. 2015. Disponível em: <http://www.ncbi.nlm.nih.gov/pubmed/26612925>. Acesso em: 1 mar. 2018.

LAMALLE, P. **Nonlocal theoretical generalization and tridimensional numerical study ofthe coupling of an ICRH antenna to a tokamak plasma**. 1994. Faculté des Sciences de l'Universite de Mons, 1994.

LAMALLE, P. U. On the radiofrequency response of tokamak plasmas. **Plasma Physics and Controlled Fusion**, v. 39, n. 9, p. 1409–1460, 1 set. 1997. Disponível em: <http://stacks.iop.org/0741-3335/39/i=9/a=011?key=crossref.bbfb7855bcbc79088a91a337b63753eb>. Acesso em: 1 mar. 2018.

LAWSON, J. D. Power from Nuclear Fusion. **Nature 1957 180:4590**, 19 out. 1957.

SMITH, C. L.; COWLEY, S. The path to fusion power. **Philosophical transactions. Series A, Mathematical, physical, and engineering sciences**, v. 368, n. 1914, p. 1091–108, 13 mar. 2010. Disponível em: <http://www.ncbi.nlm.nih.gov/pubmed/20123748>. Acesso em: 1 mar. 2018.

U.S. ENERGY INFORMATION ADMINISTRATION. **International Energy Outlook 2016**. [s.l: s.n.]v. 0484(2016)

AGRADECIMENTOS

Pesquisa desenvolvida com o auxílio do Centro de Armazenamento de dados e Computação Avançada da UESC - CACAU, implantado com recursos FINEP/MCT. A pesquisa contou também com o apoio da FAPESB, da UESC e do NBCGIB.

Edição especial - XX ENMC (Encontro Nacional de Modelagem Computacional) e VIII ECTM (Encontro de Ciência e Tecnologia dos Materiais), realizado entre 16 e 19 de outubro de 2017 na cidade de Nova Friburgo – RJ.

Editor – Mateus das Neves Gomes