UNIVERSIDADE ESTADUAL DE SANTA CRUZ
PRO-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO

PROGRAMA DE PÓS-GRADUAÇÃO EM MODELAGEM COMPUTACIONAL
EM CIÊNCIA E TECNOLOGIA

PAULO OLIVEIRA PAIXÃO

PARALELIZAÇÃO E ESTUDO DE DESEMPENHO DE SIMULAÇÕES DOSIMÉTRICAS POR MONTE CARLO UTILIZANDO O CÓDIGO GATE/GEANT4

ILHÉUS-BA
2016

PAULO OLIVEIRA PAIXÃO

PARALELIZAÇÃO E ESTUDO DE DESEMPENHO DE SIMULAÇÕES DOSIMÉTRICAS POR MONTE CARLO UTILIZANDO O CÓDIGO GATE/GEANT4

Dissertação apresentada ao Programa de Pós-Graduação em Modelagem Computacional em Ciência e Tecnologia da Universidade Estadual de Santa Cruz, como parte das exigências para obtenção do título de Mestre em Modelagem Computacional em Ciência e Tecnologia.

Orientador: Prof. Dr. Felix Mas Milian

ILHÉUS-BA
2016

P149 Paixão, Paulo Oliveira.
Paralelização e estudo de desempenho de simulações dosimétricas por Monte Carlo utilizando o código GATE/Geant4 / Paulo Oliveira Paixão. – Ilhéus, BA: UESC, 2016.
85 f. : il. ; anexo.

Orientador: Felix Mas Milian.
Dissertação (Mestrado) – Universidade Estadual de Santa Cruz. Programa de Pós-Graduação em Modelagem Computacional em Ciência e Tecnologia.
Inclui referências e apêndices.

1. Simulação (Computadores). 2. Computação de alto desempenho. 3. Monte Carlo, Método de. 4. Radioterapia. 5. Radiação. I. Título.

CDD 530.15

PAULO OLIVEIRA PAIXÃO

PARALELIZAÇÃO E ESTUDO DE DESEMPENHO DE SIMULAÇÕES DOSIMÉTRICAS POR MONTE CARLO UTILIZANDO O CÓDIGO GATE/GEANT4

Ilhéus-BA, 21 de julho de 2016

Comissão Examinadora

Prof. Dr. Felix Mas Milian
UESC
(Orientador)

Prof. Dr. Esbel Tomás Valero Orellana
UESC

Prof. Dr. Diego Gervasio Frías Suárez
UNEB

Dedico aos meus pais, minha esposa, minha família e todas a pessoas que acreditaram e torceram por mim.

Agradecimentos

- A Deus, eu agradeço pela vida, pela saúde e pelas pessoas existentes na minha vida;
- Aos meus pais, Ângelo O. Paixão (*in memorium*) e Leozina R. de Jesus pela preocupação, apoio, pois me tornaram forte nos momentos de fraqueza, pelos ensinamentos e valores;
- Ao amor de minha vida e minha esposa, Margarete Paixão, eu agradeço pelo amor, pela dedicação, pela compreensão, pelo incentivo, pela solidariedade e sobretudo paciência;
- A toda minha família, fisicamente próximos ou distantes, eu agradeço pelo afeto e pela atenção;
- Ao meu orientador, Prof. Dr. Felix Mas Milian, meus agradecimentos pelo suporte, paciência, confiança, oportunidade e ajuda incondicional;
- A turma de 2014.1 do PPGMC, pelo apoio, suporte, companheirismo, auxílio e compartilhamento de conhecimento. Não sendo injusto com outros, mas em especial: Jorge Farias, Diogo Novais, Tarsila Matos;
- Aos colegas do Instituto Federal de Educação Ciência e Tecnologia - IFBA;
- A todos os funcionários da UESC e do PPGMC, eu agradeço pelo suporte;
- A todos os funcionários do CTR e NBCGIB pelo suporte e cessão das instalações para realização do trabalho
- A todos os professores do PPGMC, eu agradeço pelas informações partilhadas e por todas as contribuições;
- A todos que não foram citados, recebam minhas desculpas e meu muito obrigado.

Paralelização e Estudo de Desempenho de Simulações Dosimétricas por Monte Carlo Utilizando o Código Gate/GEANT4

Resumo

O método de Monte Carlo se tornou um método importante para simular eventos em diferentes áreas como a Física Médica, Ciência das Radiações, Indústrias nucleares e Aeroespacial. Muitos códigos foram desenvolvidos para usar o método de Monte Carlo para simular problemas que podem ser representados por processos estocásticos e, uma delas é o conjunto de ferramentas GATE/Geant4. Simulações complexas com GATE/-Geant4 associado ao Método de Monte Carlo pode durar muito tempo para serem processadas e o uso da computação de alto desempenho pode reduzir o tempo com técnicas de paralelismo que consiste em dividir uma simulação para serem executados por vários processadores. O grupo de estudo de Física Médica da UESC que desenvolve pesquisas e simulações com radioterapia, tem necessidade de reduzir o tempo de processamento mantendo a exatidão dos resultados. Neste trabalho foi realizada a paralelização do GATE/Geant4, estudado o comportamento em *clusters*, analisado o desempenho em simulações com diferentes fantomas, números de histórias e tipos de partículas, utilizando a paralelização com *job spliter*. O GATE não tem suporte nativo para o gerenciador de recursos SLURM para *cluster*. Para realizar as simulações foram implementados *Scripts* de forma a tornar o GATE compatível com esse poderoso gerenciador utilizado em computação de alto desempenho. Vários estudos foram realizados em simulações com fantomas de *box* e fantomas de *voxels*, com dimensões 100 x 100 x 100 cm e resolução de 1 cm e 0,5 cm. As partículas irradiadas pela fonte foram carbono, próton e gama. Simulações foram realizadas com $10^5, 10^6, 10^7 e 10^8$ histórias. A energia de cada partícula foi: 200 MeV para simulações com próton, 18 MeV para o gama e 2500 MeV para íon de carbono. Com os resultados das simulações tabulados, foram gerados gráficos demonstrativos dos parâmetros que afetam as simulações. Desempenho das simulações com o aumento da número de CPUs em cada *cluster*, tempo médio estimado para processamento de uma história, obtenção do percentual aproximado de eficiência das simulações, e analisado o desempenho com base no intervalo de gravação das simulações. Os resultados obtidos permitiram a validação da paralelização do GATE com SLURM e o desempenho satisfatório na paralelização, viabilizando a realização das simulações pelo grupo de Física Médica da UESC.

Palavras-chave: Simulação computacional. Computação de alto desempenho. Método de Monte Carlo. Radioterapia. Gerenciadores de recursos.

Parallelization and Simulation Performance Study Dosing Monte Carlo Using the Gate CATE/Geant4

Abstract

The Monte Carlo method has become an important method to simulate events different areas such as Medical Physics, Science Radiation, Nuclear and Aerospace Industries. Many codes have been developed to use a Monte Carlo method to simulate problems that can be represented by stochastic processes, one of which is the set of gate/Geant4 tools. complex simulations with GATE/Geant4 associated with the Monte Carlo method can last a long time to process and the use of high performance computing can reduce the time with parallelism techniques that is to divide a simulation to run across multiple processors. The study group UESC of Medical Physics that conducts research and simulations with radiotherapy, needs to reduce processing time while maintaining the accuracy of the results. This work was performed parallelization of GATE/Geant4, studied the behavior of clusters, analyzed the performance simulations with different phantoms, stories of numbers and types of particles using parallelization with job spliter. The GATE has no native support for slurm resource manager for the cluster. To perform the simulation scripts have been implemented in order to make the GATE compatible with this powerful manager used in high-performance computing. Several studies were performed simulations with phantoms box and voxel phantoms with dimensions 100 x 100 x 100 cm and a resolution of 1 cm and 0.5 cm. The particles were irradiated by the source carbon, proton and gamma. Simulations were performed with $10^5, 10^6, 10^7$ and 10^8 stories. The energy of each particle was: 200 MeV for simulations with proton 18 MeV for gamma and 2500 MeV for carbon ion. With the tabulated results of simulations graphs showing the parameters were generated that affect the simulations. Performance simulations with increasing numbers of CPUs in each cluster, estimated average time for processing a story, getting the approximate percentage of efficiency of simulations and analyzed the performance based on the recording interval of the simulations. The results allowed validation of parallelization of GATE with slurm and satisfactory performance in parallelization, allowing the realization of simulations by the Medical Physics group UESC.

Keywords: Computational Simulation. High Performace Computing. Monte Carlo Method. Radiotherapy. Resource Managers.

Lista de figuras

Lista de tabelas

Lista de abreviaturas e siglas

CACAU	Centro de Armazenamento de dados e Computação Avançada da UESC
CAD	*Computed Aided Design*
CTR	Centro de Tecnologias das Radiações
CPU	*Central Processor Unit*
CRT	*Cathode Rays Tube*
eV	*eletro volt*
FM	Física Médica
GATE	*Geant4 Application for Tomography Emission*
GB	Giga *Byte*
Geant4	*GEometry ANd Tracking*
GPU	*Graphics Processing Unit*
GUI	*Graphics Inteface Unit*
HPC	*High-performance computing*
INCA	Instituto Nacional do Câncer
ISO	*International Organization for Standardization*
KEK	Sigla japonesa para *(High Energy Accelerator Research Organization)*
LAN	*Local Area Network*
MIMD	*Multiple Instruction stream and Multiple Data stream*
MISD	*Multiple Instruction stream and Single Data stream*
Mhz	*Mega hertz*
MMC	Método de Monte Carlo
MPI	*Message Parsing Interface*
MPP	*Massive Parallel Processors*

NBCGIB Núcleo de Biologia Computacional e Gestão de Informações Biotecnológicas

NUMA *Nonuniform Memory Access*

OpenGL *Open Graphics Language*

OpenMP *Open Multi-Processing*

ParGean4 Parallel Geant4

PENELOPE *PENetration and Energy LOss of Positrons and Electrons*

PET *Positron Emission Tomography*

RAID *Redundant Array of Independent Disks*

RAM *Randomic Access Memory*

SIMD *Single Instruction stream and Mutliple Data stream*

SISD *Single Instruction stream and Single Data stream*

SLURM *Simple Linux Utility for Resource Management*

SMP *Simetric Multiprocessor*

SPECT *Single Photon Emission Computerized Tomography*

TB Tera *Byte*

TeV *Tera eletro volt*

UESC Universidade Estadual de Santa Cruz

UMA *Uniform Memory Access*

DCET Departamento de Ciências Exatas e Tecnológicas

PPGMC Programa de Pós-Graduação em Modelagem Computacional em Ciência e Tecnologia

Sumário

1 Introdução

O método de Monte Carlo se tornou uma importante ferramenta para simular eventos em diferentes áreas como: Física Médica, Ciência das Radiações, Indústria Nuclear e Aeroespacial. É fundamental para simular transporte de partículas e cálculos de dose absorvida, relacionados ao tratamento que usa radiação tanto com fontes externas como com fontes internas. Tem sido usado para avaliação de dose em procedimentos diagnósticos e estudos sobre qualidades de imagens médicas em geral (YORIYAZ, 2009). As simulações de transporte de radiação com o método Monte Carlo processam informações com base em imagens e cálculo com matrizes de grandes dimensões e, em geral, de maneira sequencial.

Projetos de computadores em sua maioria foram desenvolvidos com base na arquitetura *Von Neumann* que são formados por unidades de processamento e memória. A maioria dos programas escritos para essa arquitetura, segue o fluxo sequencial de execução e o tempo total de processamento é a soma dos tempos de cada instrução (STALLINGS, 2010). Alternativamente, novas tecnologias foram desenvolvidas com o objetivo de processar várias tarefas, surgindo assim, o processamento paralelo que consiste em dividir uma tarefa originalmente sequencial em partes e executá-las simultaneamente em unidades de processamento diferentes. São distribuídas através de processos de comunicação entre múltiplos processadores com o objetivo de reduzir o tempo computacional.

O processamento paralelo depende de arquiteturas computacionais multiprocessadas e uma das mais utilizadas são os *clusters*. Consiste em um conjunto formado por *hardware* e *software* unificado, interligado por um sistema de comunicação e transparente ao usuário. Um dos mais utilizados são os *clusters* de alto desempenho que tem como características a capacidade de processar grandes volumes de instruções em tempo reduzido. Apesar dos *clusters* possuírem vários recursos físicos como capacidade de memória e processamento, eles são limitados e devem ser administrados de forma a obter o máximo de desempenho. Essa administração é feita pelos gerenciadores de recursos que tem como finalidade garantir a estabilidade do sistema escalonando os recursos de acordo as prioridades das tarefas e baseada em prioridades. A qualidade de serviço para uma aplicação paralela fornecida por um sistema, não depende apenas dos recursos suficientes para promover o desempenho, mas também de um bom gerenciador (COULORIS; DOLLIMORE; KINDBERG, 2007).

Muitos códigos foram desenvolvidos para usar o método de Monte Carlo na simulação de problemas que podem ser representados por processos estocásticos. Um desses é o conjunto de ferramentas GATE/Geant4. Esse conjunto de ferramentas de-

senvolvido pelo CERN (*European Organization for Nuclear Research*) é mantido de forma colaborativa pela comunidade científica internacional e considerado o maior projeto de *software* fora do mundo corporativo. Esse conjunto dispõe de capacidade para modelar fenômenos dependentes do tempo, como movimentos de detectores ou decaimento de fontes, permitindo assim, simulação de curvas de tempo sob condições realistas (COLLABORATION, 2014; JAN et al., 2014).

As simulações de transporte de radiação realizadas com o método Monte Carlo, utilizando o GATE/Geant4, realizam processamento de informações com base em imagens e cálculo com matrizes de grandes dimensões. Neste sentido, a computação paralela tem sido usada cada vez mais por cientistas, engenheiros e pesquisadores para resolverem problemas complexos com o uso de ferramentas de simulação computacional e muitas delas faz o uso do Método de Monte Carlo. O software GATE/Geant4 tem suporte para processar tarefas paralelas através de técnicas *job splitter*, permitindo a divisão de uma simulação para que seja processada em sistema com vários processadores, GPU's (*Graphics Processing Units*) ou arquiteturas distribuídas com o objetivo de reduzir de tempo execução (COLLABORATION et al., 2016)

Nas simulações envolvendo radioterapia é importante ter uma estimativa do número de histórias e tempo de simulação necessários para se obter uma incerteza inferior a 5% nas doses recebidas por pacientes. O grupo de estudo de Física Médica da Universidade Estadual de Santa Cruz (UESC) que desenvolve pesquisas e simulações com radioterapia, necessita reduzir o tempo de processamento das simulações mantendo a exatidão dos resultados. Para isso, a execução paralela dos códigos GATE/Geant4 deve ser usada como solução para redução do tempo de processamento nos *clusters* da instituição, entretanto, os *clusters* da UESC utilizam como gerenciador de recursos o SLURM *Simple Linux Utility for Resource Management* e o GATE/Geant4 com *job splitter* não tem suporte para o SLURM.

Neste trabalho foi desenvolvido uma solução para executar o GATE/Geant4 nos *clusters* CTR e CACAU da UESC para o Grupo de Física Médica realizar estudos com simulações utilizando partículas gama, próton, carbono, três tipos de geometrias e número de histórias variando de 10^5 a 10^8 nas pesquisas envolvendo radioterapia. O estudo desse trabalho visa responder questionamentos como: Qual a estimativa de histórias necessária para se obter um erro menor que 5% em função da raio da geometria? O tempo de simulação é influenciado pela partícula, mantendo as características das simulações? A geometria interfere no tempo final de simulação? Qual o percentual de eficiência dos *clusters* usados nas simulações? Qual o tempo médio para realizar uma simulação? Qual percentual de redução de tempo de simulação em função do tempo de gravação em disco?

Esses questionamento levam aos objetivos desse trabalho.

1.1 Objetivos geral

Desenvolver solução para paralelização do código GATE/Geant4 com SLURM e estudar o desempenho em arquiteturas computacionais com multiprocessadores ou distribuídos, utilizando diferentes tipos de geometrias, números de histórias e partículas, para redução no tempo de processamento das simulações dosimétricas realizadas pelo grupo de Física Médica da UESC.

1.2 Objetivos específicos

- Implementar uma solução computacional que permita o processamento paralelo de tarefas com *job splitter* no GATE/Geant4 em *clusters* com gerenciador SLURM;
- Estimar o número de histórias necessários para se obter um erro da incerteza da dose depositada inferior a 5% em função do raio da geometria;
- Estudar o comportamento do tempo de execução em função da geometria utilizada, mantidas as mesmas características das simulações;
- Avaliar a influência do tipo de partícula no tempo final de execução, mantidas as características das simulações;
- Calcular o percentual de eficiência das simulações em cada *cluster* e comparar o desempenho das simulações realizadas no CTR e CACAU;
- Quantificar o tempo médio de simulação de uma história em cada *cluster* da UESC;
- Avaliar e quantificar o percentual de redução do tempo de simulação em função do tempo de gravação em disco.

1.3 Estrutura do Trabalho

O restante desse trabalho foi estruturado em quatro capítulos: (ii) Referencial teórico; (iii) Métodos e discussão; (iv) Considerações finais.

No capítulo 2 são abordados temas como: Tipos de radioterapia, radiação ionizante e suas fontes de produção e emissão, método de Monte Carlo, principais características do GATE e Geant4, técnicas de paralelização do Gate/Geant4, computação paralela e distribuída e escalonadores de recursos computacionais.

O capítulo 3 descreve toda a metodologia empregada na pesquisa, descritos os aspectos principiais e procedimentos adotados para realizar as simulações, recursos utilizados, tarefas executadas e configurações de hardware disponíveis. Os *scripts*

utilizados na manipulação dos arquivos e mencionados neste capítulo, foram disponibilizados nos Apêndices. Neste capítulo também são apresentados os resultados obtidos com as simulações realizadas em duas estruturas computacionais de alto desempenho da UESC, o *cluster* existente no Centro de Pesquisas em Ciências e Tecnologias das Radiações (CTR) e o *cluster* CACAU instalado nas dependências do Núcleo de Biologia da UESC.

Por fim no capítulo 4 são apresentadas as principais considerações e conclusões a que se chegou na realização esse trabalho, além das propostas de trabalhos futuros que poderão ser desenvolvidos.

2 Fundamentos Teóricos

Este capítulo é composto pelo referencial teórico base deste trabalho.

2.1 Radioterapia

A radioterapia é o método de tratamento que faz uso de radiação ionizante (partículas como próton, nêutrons e íons) aplicada em áreas do corpo humano demarcadas previamente (Ministério da Saúde (BR), 2015). O objetivo é destruir ou impedir o crescimento de células tumorais através da deposição da maior quantidade de radiação sobre a região a ser tratada, causando o menor dano possível aos órgãos adjacentes (GUIMARÃES, 2015).

Os procedimentos de radioterapia se dividem em duas categorias: a teleterapia e a braquiterapia. Na teleterapia, a fonte localiza-se a uma certa distância do paciente. O alvo interno a ser tratado é irradiado por feixe de radiação externo. Grande parte da radioterapia por feixe externo é realizada com feixes de fótons que podem ser de dois tipos: raios gama ou raios X. O feixe de raios gama são produzidos a partir núcleos radioativos, enquanto o feixe de raios X são produzidos em tubos de raios X ou LINACs (aceleradores lineares) (PODGORSAK et al., 2005). Na braquiterapia, as fontes de radiação são colocados a curta distância diretamente na área onde o feixe será irradiado, (intracavitária ou braquiterapia intersticial), com este modo de tratamento, uma dose elevada pode ser irradiada localmente (PODGORSAK et al., 2005; KHAN; GIBBONS, 2014).

A protonterapia é o tratamento que utiliza prótons para tratar tumores devido ao fato de ser bastante precisa. Em determinadas aplicações, apresenta vantagens significativas devido às características de interação do próton com a matéria. Devido ao comportamento bem definido da partícula o alvo pode ser irradiado com maior precisão. Partículas pesadas como o próton, ao atravessarem os tecidos, desaceleram gradualmente, transferem energia e dispara um pico de dose preciso e localizado, chamado de pico de Bragg. Após o pico, a energia decai a zero resultando em ausência de energia a partir daquele ponto. Esse fenômeno permite que os tecidos adjacentes sejam afetados minimamente tornando o tratamento efetivo (PODGORSAK et al., 2005; CARUSO; CARVALHO; SANTORO, 2005; KHAN; GIBBONS, 2014).

A radioterapia com íons de carbono é outra forma de tratamento através da irradiação de feixes. Diferente do próton, partículas carregadas com os íons de carbono (C-íon), após o pico de Bragg (deposição de energia), a partícula continua depositando energia no meio (SALVAJOLI; SALVAJOLI, 2012).

A figura 1 mostra como cada partícula deposita sua dose sobre o tecido. Radiações ionizantes com fótons depositam a maior parte de sua energia na porção inicial do tecido e diminuindo ao longo da trajetória. Os feixes prótons e íons de carbono depositam radiação reduzida na parte inicial e a dose máxima na região mais profunda (RODER, 2014).

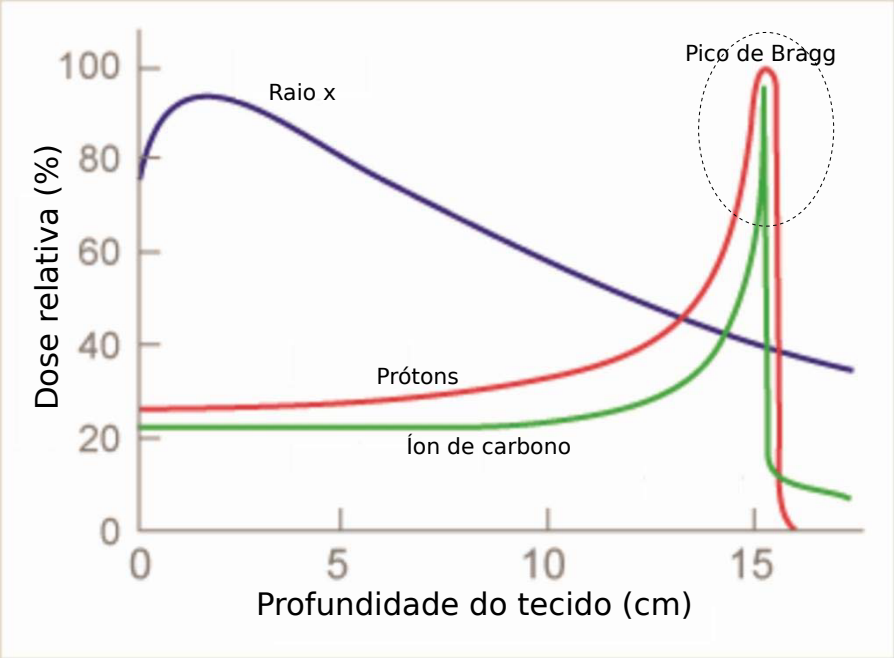


Figura 1 – Representação da deposição de dose com diferentes tipos de radiação. As linhas representam: em azul, o Raio-X; em vermelho, o próton e em verde os íons de carbono. Adaptado de Roder (2014).

2.2 Radiação

A radiação consiste no transporte de energia através de ondas eletromagnéticas ou partículas atômicas. São produzidas por processos de ajustes que ocorrem no núcleo ou nas camadas eletrônicas, ou pela interação de outras radiações ou partículas com o núcleo ou com o átomo (TAUAHATA et al., 2014). Ter conhecimento sobre a estrutura do átomo, elementos das física nuclear, natureza da radiação eletromagnética e produção de raios X é fundamental para entender imagiologia da física médica e proteção contra radiação (AGENCY, 2014a).

A radiação pode ser classificada como eletromagnética, que inclui a luz visível, infravermelho e ultravioleta, raios X, raios gama e radiação de partículas, oriundas de elétrons, pósitrons, prótons e nêutrons (AGENCY, 2014a). Pode ser produzida por meios naturais ou artificiais. A primeira, são produzidas por fontes e reações naturais, o sol e iniciou suas atividades a partir da explosão do *Big Bang*. As fontes artificiais são proveniente de dispositivos de diagnóstico e terapia utilizados na área médica,

como aparelhos de controle, medidores e radiografia usados na indústria e comércio, as instalações do ciclo de combustível nuclear, as máquinas utilizadas na pesquisa científica e sistemas de segurança utilizando raios X (AGENCY, 2014a).

2.2.1 Radiação eletromagnética

Como toda onda eletromagnética, essa radiação pode ser caracterizada quanto a sua amplitude, comprimento de onda, frequência e velocidade.

2.2.2 Classificação

Conforme (AGENCY, 2014b), a ionização pode ser classificada em duas categorias principais, dependendo da capacidade de ionizar a matéria, como ionizante ou não-ionizante.

- Radiação não ionizante - a matéria não pode ser ionizada porque a energia por *quantum*, está abaixo do potencial de ionização. São exemplos de radiação não ionizante, ultravioleta, luz, fótons infravermelho, ondas de rádio e micro-ondas.
- Radiação ionizante - a energia é transferida de uma fonte incidente para a matéria porque a energia do quantum é maior que o potencial de ionização dos átomos. são exemplos de radiação ionizante, raio X, raios gama que pode ser divida em dois tipos, ionização direta e indireta.
  - (i) Ionização direta - No processo de transferência de energia, as radiações que têm carga, como elétrons, partículas alfa e fragmentos de fissão, atuam principalmente por meio de seu campo elétrico e transferem sua energia para muitos átomos simultaneamente.
  - (ii) Ionização indireta - As radiações que não possuem carga, como eletromagnéticas e os nêutrons, interagem individualmente transferindo sua energia para elétrons, que irão provocar novas ionizações.

O potencial de ionização atômica, ou seja, a energia mínima necessária para ionizar um átomo, varia de alguns eletro volts (eV) para elementos alcalinos, a 24,6 eV para o hélio que é no grupo de gases nobres. Para todos os outros, o potencial de ionização está contido nessa faixa (AGENCY, 2014a; AGENCY, 2014b; TAUAHATA et al., 2014).

2.2.3 Geradores de radiação

Os principais tipos de geradores de radiação ionizante são: tubos de raios X, aceleradores de partícula, irradiadores com radioisótopos e fontes de nêutrons. Estes

radiadores são caracterizados pelo tipo de fonte geradora da radiação. Os aparelhos de raio X e aceleradores utilizam fonte de energia para acelerar as partículas e produzir a radiação; os irradiadores utilizam radioisótopos acoplados a um sistema blindado e guarda da fonte; já as fontes de nêutrons utilizam reação nuclear produzidas por partículas alfa, emitidas por material radioativo (TAUAHATA et al., 2014).

2.2.3.1 Raios X

O raios X foi descoberto pelo o físico alemão Wilhelm Conrad Roentgen em 1895 enquanto estudava os raios catódicos (fluxo de elétrons) em um tubo de descarga de gás. Ele observou que um outro tipo de radiação foi produzido e detectado do lado de fora do tubo. Ela foi capaz de penetrar substâncias opacas, produzir fluorescência, modificar chapa fotográfica e ionizar um gás. A este fenômeno foi dado o nome de raio X que é a denominação dada à radiação eletromagnética de alta energia que tem origem na eletrosfera ou no freamento de partículas carregadas no campo eletromagnético do núcleo atômico ou dos elétrons.

A figura 2 representa o digrama esquemático de uma fonte geradora de raios X. É um tubo constituído por uma ampola de vidro contendo vácuo, em uma das extremidades tem um cátodo (eletrodo negativo) e no outro, um ânodo (eletrodo positivo) no tubo de vidro hermeticamente fechado (KHAN; GIBBONS, 2014).

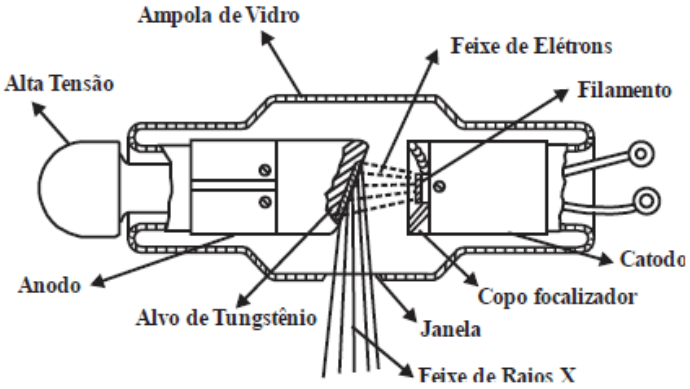


Figura 2 – Diagrama esquemático de um gerador de raios X.

Fonte: Tauahata et al. (2014).

Em um tubo de raios X, o feixe de elétrons é gerado por emissão termoiônica num filamento aquecido normalmente de tungstênio (W), onde ao aplicar alta voltagem entre os terminais, um campo elétrico é obtido e os elétrons partem do cátodo (filamento) polarizado positivamente, na direção do anodo (alvo metálico) polarizado negativamente. Quanto maior a tensão aplicada ao tubo, maior será a energia dos raios X gerados e o poder de penetração. Ao aumentar a corrente, aumenta-se a intensidade do feixe, porém, a emissão de raios X só ocorre quando a alta tensão for aplicada à fonte.

Apesar do mesmo princípio físico, os tubos de raio X sofrem variações no formato, tipo de alvo do anodo, faixa da tensão (kV) e corrente aplicadas além do sistema de refrigeração (TAUAHATA et al., 2014).

2.2.3.2 Aceleradores lineares

Os aceleradores de elétrons são dispositivos mais difundidos principalmente no tratamento de câncer. Permitem a geração de feixes intensos de partículas com energia variável, utilizando processos de aceleração baseados em campos elétricos, magnéticos e ondas eletromagnéticas. O acelerador linear é um dispositivo que usa ondas eletromagnéticas para acelerar partículas carregadas com elétrons de alta energia por um tubo linear (TAUAHATA et al., 2014). A emissão termoiônica nos filamentos aquecidos, geram elétrons que são injetados num tubo e carregados por uma onda portadora estacionária, através das secções da máquina, até atingir a energia desejada. Os feixes de elétrons de alta energia podem ser usados para tratamento de tumores superficiais ou tratamento de tumores mais profundos pela produção de raio X (KHAN; GIBBONS, 2014). Estes aceleradores são utilizados principalmente em hospitais, nas indústrias e nos institutos de pesquisa em todo mundo. A figura 3 representa o diagrama esquemático de um acelerador linear utilizado no tratamento de doenças a base de radiação.

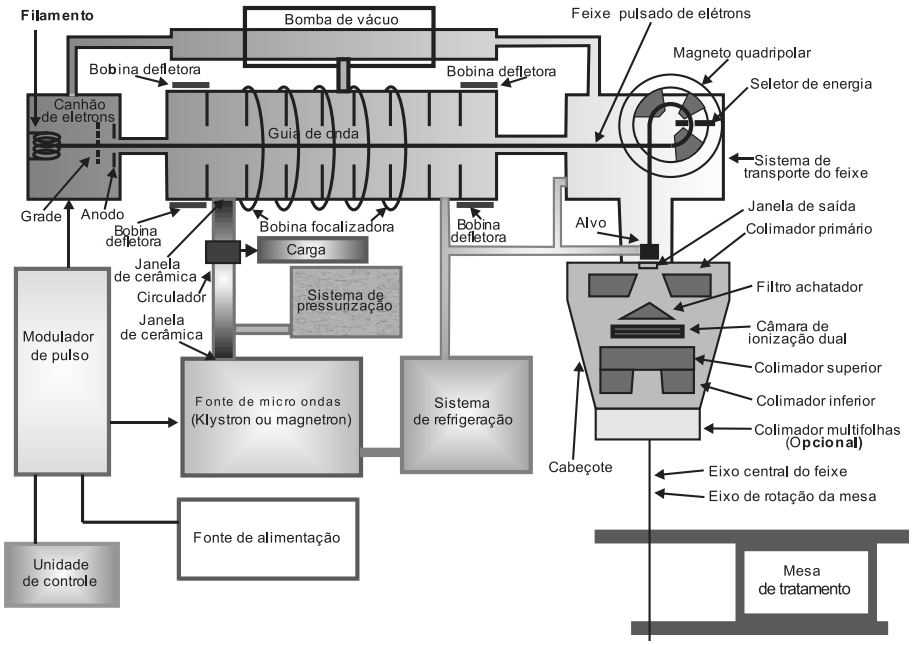


Figura 3 – Diagrama esquemático de um acelerador linear de elétrons.

Fonte: Tauahata et al. (2014).

2.3 Método de Monte Carlo (MMC)

O método de Monte Carlo é uma solução numérica que modela as interações de objetos em seu ambiente, com base no simples relacionamentos objeto-objeto ou objetos-ambiente. Ele representa uma tentativa de modelar a natureza através da uma simulação direta da dinâmica essencial de um sistema.

É considerado um método essencialmente simples na sua abordagem para solucionar um sistema macroscópico e a solução é determinada pela amostragem das interações microscópicas até que o resultado convirja. O MMC pode ser utilizado para simular eventos em vários segmentos como ciência social, crescimento populacional, finanças, genética, ciências das radiações, radioterapia e transporte de partículas. Na medida que as interações microscópicas podem ser modeladas matematicamente, soluções repetitivas podem ser executadas por um computador e apesar desse método preceder a concepção do computador, utilizá-los em conjunto pode determinar a solução de um problema de forma mais rápida. (BIELAJEW, 2001).

O conjunto de ferramentas Geant4 usa uma combinação da composição e rejeição do MMC. Na sequência é descrito o formalismo básico do deste método conforme apresentado por (WRIGTH; COLLABORATION, 2011). Suponha que x esteja contido no intervalo $[x_1,\ x_2]$ da distribuição $f(x)$ e a função da densidade de probabilidade normalizada possa ser escrita como:

$$f(x) = \sum_{i=1}^{n} N_i f_i(x) g_i(x) \tag{1}$$

onde $N_i > 0$, $f_i(x)$ função da densidade normalizada $[x_1,\ x_2]$, e $0 \leq g_i(x) \leq 1$. De acordo com este método, x pode ser amostrado da seguinte forma:

1. selecione um número $i \in \{1, 2, \cdots n\}$ com probabilidade proporcional para N_i
2. selecione um valor x_0 da distribuição $f_i(x)$
3. calcule $g_i(x_0)$ e faça $x = x_0$ com probabilidade $g_i(x_0)$;
4. se x_0 é rejeitado reinicia do passo 1.

Ele mostra que o esquema está correto e o número médio de tentativas para aceitar um valor é $\sum_i N_i$. Na prática, um bom método de amostragem a partir das distribuição $f(x)$ tem as seguintes propriedades:

- todas as sub-distribuições $f_i(x)$ podem ser amostradas facilmente;
- a função rejeição $g_i(x)$ pode ser avaliada facilmente/rapidamente;

- o número médio de tentativas tende a não ser grande.

Assim as diferentes decomposições possíveis de distribuição $f(x)$ não são equivalentes do ponto de vista prático (por exemplo, pode ser muito diferente da velocidade computacional) e pode ser útil para optimizar a decomposição Uma informação de importância prática: se nossa distribuição não é normalizada

$$\int_{x_1}^{x_2} f(x)dx = C > 0$$

, o método pode ser usado da mesma maneira e o número médio de tentativas neste caso é $\sum_i N_i/C$.

O método de Monte Carlo pode ser descrito como um método estatístico, no qual utiliza-se a geração de números aleatórios para realizar uma simulação. Esse método permite implementar processos elementares de forma estocástica e em termos de transporte de radiação, o processo estocástico pode ser visto como uma família de partículas cujas coordenadas individuais mudam de direção aleatoriamente em cada colisão e o comportamento médio dessas partículas é descrito em termos de grandezas macroscópicas, como fluxo ou densidade (CASSOLA, 2007; MARTINS, 2014).

O valor esperado dessas grandezas corresponde à solução determinística da equação de Boltzman (muito utilizada na análise dos fenômenos de transporte) da qual deriva as grandezas como dose ou energia depositada. As simulações estatísticas contrastam com métodos convencionais de discretização, que são tipicamente aplicados em sistemas de equações diferenciais parciais ou ordinárias que descrevem o processo físico. À medida que o número de histórias das partículas simuladas aumenta, melhora-se a qualidade do comportamento médio do sistema, caracterizado pela diminuição das incertezas estatísticas e das grandezas de interesse (YORIYAZ, 2009).

Na simulação do transporte de radiação utilizando o método de Monte Carlo, a história de uma partícula é definida como uma sequência de caminhos que terminam com um evento de interação, nos quais podem ocorrer mudança de direção, perda de energia, bem como a produção de partículas secundárias. Para simular essas histórias é necessário modelos de interações que, geralmente, são baseados em um conjunto de seções de choque diferenciais para os mecanismos de interação relevantes.

As seções de choque determinam as funções densidade de probabilidade das variáveis aleatórias que caracterizam a trajetória, tais como: livre caminho entre os eventos de interação sucessivos; tipo de interação que ocorre; energia perdida e deflexão angular de um evento particular. Uma vez conhecidas essas funções densidades e probabilidade, as histórias podem ser geradas como uso de métodos apropriados de amostragem. Se o número de históricos gerados for grande o suficiente, informações quantitativas do

processo de transporte podem ser obtidas pela simples média das histórias simuladas, reduzindo as incertezas estatísticas e tempo computacional (CORDEIRO, 2013).

A essência do MMC aplicado a transporte de radiação consiste em estimar determinadas quantidades, observando-se o comportamento de um número grande de eventos individuais. Dentre as diversas áreas que utilizam o método, na física médica, ele se tornou ao longo dos anos um código fundamental para cálculos de dose absorvida, relacionados ao tratamento do câncer por radiação, tanto com fontes externas como internas. Além disso, as aplicações do método têm se estendido para a avaliação de dose em procedimentos diagnósticos e estudos sobre qualidades de imagens médicas em geral (YORIYAZ, 2009).

Estimar corretamente as doses de radiação é um dos principais objetivos em tratamento para reduzir ou eliminar um câncer, mas definir corretamente a quantidade e intensidade sobre uma determinada área do corpo humano é necessário um bom planejamento, pois, durante o planejamento realizado pela equipe médica é definido o tipo de radiação a ser utilizada, a intensidade, tempo de exposição e posição de incidência do feixe radiativo.

As simulações computacionais têm sido fundamentais nesse planejamento, evitando que os pacientes sejam expostos a quantidades excessivas de radiação durante o tratamento e minimizando os impactos da radiação nos órgãos sadios adjacentes ao tecido acometido à neoplasia (OBAL, 2011). Os códigos para simulações de partículas ou radiação e interação com a matéria foram desenvolvidos inicialmente para aplicações físicas de alta energia. Com o crescimento das aplicações da radiação ionizante em FM, os códigos de Monte Carlo foram adaptados para aplicações de alta energia capaz de atingir a faixa de *TeV*. Dentre eles o Geant4 é uma das ferramentas mais avançadas para simulação de transporte de partículas. (MALTHEZ, 2011). O Geant4, criado em 1994 a partir do Geant3, foi reescrito em linguagem C++ e utiliza a técnica de programação orientada a objeto. Desenvolvido pelo CERN em conjunto com outros centros de pesquisa e instituições através de um projeto colaborativo (considerado o maior fora do mundo corporativo) e disponibilizado gratuitamente para à comunidade científica internacional (COLLABORATION, 2014).

2.4 Geant4 - *GEometry ANd Tracking*

O Geant4 (GEometry ANd Tracking) está em constante desenvolvimento e consiste em um pacote composto de ferramentas computacionais capaz de simular a passagem de partículas através da matéria. Conforme CERN et al. (2014) estão incluídos nesta ferramenta todos os aspectos que envolve o processo de simulação como:

- geometria do sistema;
- materiais envolvidos;
- partículas fundamentais de interesse;
- geração de eventos primários;
- passagem de partículas através de materiais e campos eletromagnéticos, envolvendo possíveis interações e processos de decaimento;
- processos físicos que regem as interações das partículas;
- respostas dos elementos sensíveis dos detectores;
- armazenamento de eventos e trajetória da partícula;
- visualização do detector e trajetórias das partículas;
- processos físicos que geram interação das partículas;
- captura análise dos dados de simulação em diferentes níveis de precisão e refinamento;
- diferentes níveis de energia baseado em fantomas.

Conforme (CERN et al., 2014), o Geant4 é uma plataforma *OpenSource* mantida sob modelo de colaboração, distribuída gratuitamente pelo CERN (*European Organization for Nuclear Research*). Com essa plataforma é possível a construção de modelos geométricos tridimensionais, possibilitando ao profissional ou pesquisador modelar suas simulações de forma realística. Atende inúmeras áreas como a física nuclear, biologia molecular, medicina nuclear e transportes de partícula em geral, seja um simples objeto ou modelo baseado em fantomas com características mais parecidas com a anatomia do corpo humano. Ele é composto por um grande conjunto de modelos físicos para lidar com interações das partículas com a matéria através de ampla gama de energia.

Os dados e conhecimentos, são produções mundiais de quase tudo que se sabe sobre interações de partículas e agregados ao Geant4 de forma de colaborativa. É um conjunto de ferramentas escrito em C++ que utiliza técnicas avançadas de engenharia de software e paradigma de orientação a objetos tornando o sistema transparente aos usuários. Estes conceitos são importantes para o gerenciamento da complexidade e organização do código comuns a todos os modelos físicos, definição de interface uniforme. Para a construção de novos modelos físicos são necessárias poucas modificações em relação ao original.

2.4.1 A história do Geant4

O surgimento do Geant4 se deu a partir dos estudos realizados em 1993 por dois grupos independentes CERN e KEK. Esses grupos investigavam como técnicas de computação moderna poderiam ser aplicados para melhorar o simulador Geant3 desenvolvido com base no FORTRAN. Em 1994 os dois grupos se uniram para construir um novo programa com novos recursos como orietanção a objetos.

Essa proposta foi submetida ao Comitê de Pesquisa e Desenvolvimento Detector do CERN (*CERN's Detector Research and Development Committee* e a iniciativa recebeu colaboração da comunidade científica internacional de áreas como física de alta energia de países como Japão, Estados Unidos, Canadá, além da comunidade europeia.

O projeto surgiu com o objetivo de atender experiências da física subatômica, mas a ferramenta passou a beneficiar outras áreas como a física nuclear, médica, aceleradores, recebendo então contribuição de um grupo cada vez maior e em 1994 foi lançada a primeira versão batizada de Geant4. Considerando o tamanho de código, natureza do software e número de contribuintes, o projeto Geant4 é considerado o maior projeto fora do mundo corporativo. Cada parte do Geant4 tem um coordenador responsável por gerenciar um grupo de trabalho que envolve atividades como desenvolvimento, testes e garantia da qualidade, gerenciamento de software e documentação (CERN et al., 2014).

Desenvolver aplicações de *softwares* baseado nos conceitos de orientação a objeto é mais intuitivo e tem muitas vantagens associadas. O desenvolvimento em módulos independentes, reduz a possibilidades de erros, melhora a depuração, promove a reusabilidade, manutenibilidade, desenvolvimento ágil e organização. O conceito de orientação a objetos visa construir uma objetos computacionais com base nas entidades reais (BONIFÁCIO, 2011).

2.4.2 Visão geral de funcionalidade do Geant4

A figura 4 representa o diagrama das categorias de classe do Geant4. Essas categorias são definidas no projeto do sistema para representar os relacionamentos e dependências entre as classes. O retângulo representa as classes, as linhas representam os relacionamentos entre as classes e o círculo representa a dependência que uma classe tem da outra.

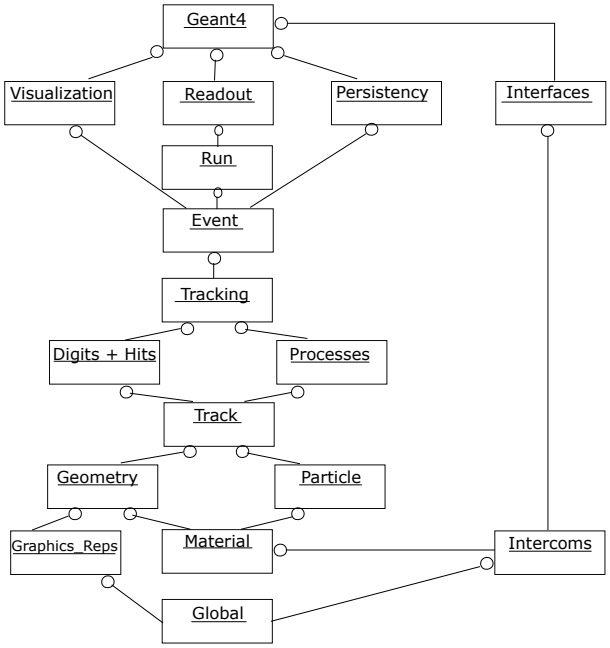


Figura 4 – Diagrama categorias de classes do Geant4.

Fonte: Adaptado de CERN et al. (2014)

Com base na representação gráfica da figura 4, as categorias de classes que possuem um círculo em seu relacionamento é dependente da categoria que está na outra extremidade. Neste caso, categorias da parte inferior do diagrama são utilizados por praticamente todas as categorias mais elevadas que formam a base do projeto desta ferramenta (CERN et al., 2014). A organização do Geant4 da forma como apresentado na figura 4, tem como objetivo facilitar o desenvolvimento de novas aplicações e a manutenção de software. A seguir um breve descrição das principais classes conforme (COLLABORATION, 2013).

- ***Run*** e ***Event*** - Estas são categorias relacionadas com a geração de eventos, interfaces para geradores de eventos, e as partículas secundárias produzidas. Fornece ao Gerenciador de rastreamento as partículas a serem rastreados;

- ***Tracking*** e ***Track*** - São as categorias relacionadas com a propagação de uma partícula através da análise dos fatores que limitam o passo e aplicar os processos físicos relevantes;

- ***Geometry*** e ***Magnetic Field*** - Estas categorias gerenciam as definições geométricas de um detector e cálculo das distância físca para o sólido. O medelador de geometria é baseando na norma ISO para permitir no futuro a integração com sistemas CAD. Uma característica chave da geometria GEANT4 é que as definições de volume é independente da representação do sólido;

- ***Particle Definition*** e ***Matter*** - Define as características das partículas e materiais utilizados pelos processos nas simulações;
- ***Physics*** - Esta categoria gerencia todos os processos físicos participantes nas interações de partículas envolvidas no processo. A interface abstrata de processos físicos permite múltiplas implementações de modelos físicos por interação ou por canal e, esses modelos podem ser selecionados por faixa de energia, tipo de partícula, material. O encapsulamento de dados e polimorfismo implementados possibilitam acesso transparente;
- ***Hits*** e ***Digitization*** - Estas duas categorias gerenciam a criação de respostas positivas e a sua utilização para a fase de digitalização;
- ***Visualization*** - Esta categoria gerencia a visualização de sólidos, trajetórias, e interage com bibliotecas gráficas subjacentes. A maioria das funcionalidades gráficas básicas utilizadas já haviam sido implementadas nas primeiras versões. O projeto orientado a objetos do componente de visualização, permitiu o desenvolvimento de vários projetos pilotos de forma independente; e
- ***Interfaces*** - Esta categoria lida com as interfaces gráficas dos usuários (GUI) e interação com programas externos.

2.5 GATE - (*Geant4 Application for Emission Tomography*

Vários códigos foram desenvolvidos para (PET) e (SPECT) com resultados que não atendem as demandas da comunidade científica, mas outros como EGS4, MCNP, GEANT3, GEANT4, escritos para física de alta energia, incluem modelos físicos bem validados, possuem ferramentas de modelagem geométrica e interfaces de visualização eficientes.

O GATE é uma interface com capacidade para modelagem fenômenos dependentes do tempo, como movimentos de detectores, decaimento de fontes, permitindo assim, simulação de curvas de tempo sob condições realistas. O GATE combina essas vantagens do Geant4 através de centenas de classes C++ implementadas na camada de núcleo (*Core layer*) que em sua forma organizacional está próximo do *kernel*. A figura 5 ilustra a estrutura do Geant4 que, conforme ilustrado, possui três camadas (COLLABORATION, 2014; COLLABORATION et al., 2016) .

- A camada principal refere-se àquela a qual os desenvolvedores tem acesso e podem realizar desenvolvimento de novas funções. Nela define-se as classes e métodos, podem ser definidos: a geometria, fenômenos dependentes de tempo,

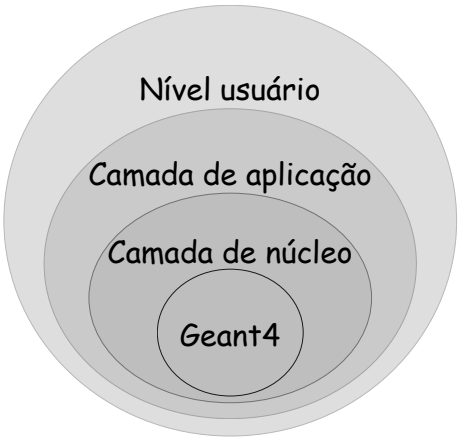


Figura 5 – Estrutura em camadas do GATE.
Fonte: Adaptado de COLLABORATION et al. (2016).

módulo de mimetização da eletrônica do detector, definição da fonte e saída de dados (VIEIRA, 2013);

- Camada de aplicação compostas por classe derivadas das classes-bases para modelar objetos com suas especificidades ou processos; e
- Camada de usuário permite que as classes e métodos das camadas anteriores sejam acessadas através *scripts*.

A camada de aplicação permite a implementação de classes do usuário derivada das classes camada de núcleo e, essas implementações incluem a construção específica de volumes geométricos e operações como rotação sobre esses volumes. Usar o GATE não necessita de programação C++, pois a camada de aplicação já implementa todas suas características. Para que o usuário possa submeter suas simulações, uma linguagem de *scripting* ou linguagem de macro que estende o interpretador de comandos nativos do Geant4 possibilita a realização de simulações por Monte Carlo com configurações realistas (COLLABORATION et al., 2016).

O GATE foi desenvolvido no âmbito colaborativo (*OpenGate Collaboration*) com o objetivo de disponibilizar à comunidade acadêmica um software gratuito, de uso geral, que fosse capaz de realizar simulações baseada na plataforma GEANT4 para tomografia por emissão. A forma de colaboração inclui, melhorar, documentar, e testar o GATE exaustivamente como os principais sistemas de imagens comercialmente disponíveis e, atualmente conta com a participação de 21 laboratórios completamente dedicados ao projeto (WRIGTH; COLLABORATION, 2011; COLLABORATION et al., 2016).

O *GATE/Geant4* possui funcionalidades que permitem a paralelização para reduzir o tempo de execução podem ser feitas de três maneiras com *Multithread*, *MPI* e *Job*

splitter. Neste trabalho não foram utilizadas as técnicas MPI ou *Multithread*, somente o *Job splitter*.

2.5.1 Método de paralelização *MPI-Message Passing Interface*

Conforme Cooperman e Nguyen (2014) à medida que aumenta a quantidade de processadores a velocidade aumenta de forma quase linear (mas não proporcional) em relação ao tempo de execução. O Geant4 possui uma versão paralela que implementa o paralelismo em nível de evento, para realizar simulações em sistemas com processadores remotos através da biblioteca ParGeant4 que implementa as técnicas de MPI. O ParGean4 utiliza mecanismo simples para realizar as chamadas de forma paralela, onde a *thread* principal (*master*) é responsável por enviar aos escravos (*slaves*) as tarefas. Não há necessidade de dividir os eventos em grupos separados, controlar falha momentânea de nós processadores, re-enviar as atividades, pois são realizadas automaticamente.

2.5.2 Método de paralelização *Multithread*

Conforme (COLLABORATION et al., 2014), a partir do Geant4 versão 10.0, o paralelismo a nível de evento foi introduzido. O paralelismo em nível de evento é baseado no modelo mestre-trabalhador (*master-workers*) em que um conjunto de tarefas são geradas e os trabalhadores (*workers*) são responsáveis pela simulação dos eventos, enquanto o controle de fluxo e novas entradas são de responsabilidade do mestre (*master*). As funcionalidades *multi-threading* no Geant4 são implementadas construindo novas classes ou através da modificação das classes existentes.

A interação do usuário com o sistema se dá através do máster que é responsável por criar e controlar as tarefas. Ao final da execução da tarefa os resultados de cada uma são combinados pelo máster para formar o resultado final. Para manter a compatibilidade entre sistemas e integração com *frameworks* avançados de paralelização, o Geant4 faz uso do padrão POSIX (*Portable Operating System Interface*), um padrão mundialmente aceito.

2.5.3 Método de paralelização *Job Splitter*

Simulações GATE podem ser executadas em um computador pessoal, *cluster* ou grades de computadores. Para reduzir o tempo das simulações e quando há disponibilidade de *clusters*, as simulações GATE podem ser executadas em modo *cluster*. Este modo consiste na divisão (*splitting*) de uma simulação (*jobs*) usando o *job splitter* e essas são processadas em várias CPUs e os resultados individuais são combinados em um único arquivo através *merge splitter* e, as entradas do *job splitter* são arquivos ou comandos no terminal. O gerador de números aleatórios produz uma semente para

cada *job* gerado e cada um produz um arquivo de saída que poderá ser combinado em um único arquivo, caso a saída seja do tipo ROOT (LJUNBERG; STRAND; KING, 2013). Para gerenciar o pseudo-paralelismo, três plataformas de controle de recursos computacionais são suportadas: openMosix[1], openPBS[2], Condor[3] conforme Jan et al. (2014). O GATE/Geant4, até a versão utilizada neste trabalho, não possuía suporte para o SlURM[4] e, para que as simulações fossem executadas foi necessário implementar *scripts* que permitisse a realizações das simulações deste trabalho.

2.6 Computação Paralela e Distribuída

A maioria dos projetos de computadores modernos foram desenvolvidos com base nos conceitos desenvolvidos por Jhon Von Neumann. A arquitetura *Von Neumann* é formada basicamente por unidade de processamento e memória, apresentando três conceitos (STALLINGS, 2010):

- Dados e instruções são armazenados em uma única memória de leitura e escrita;
- Conteúdo de memória endereçável por local, sem levar em consideração os tipo de dados independente do conteúdo; e
- Instruções são executadas sequencialmente, ou seja, uma instrução após a outra.

A computação paralela, por outro lado, consiste na divisão de uma grande tarefa em tarefas menores para que sejam processadas simultaneamente. São distribuídas através de processos de comunicação entre múltiplos processadores, com o objetivo de resolver um problema em tempo computacional reduzido. A troca de informações entre os processos é necessária para que as informações sejam sincronizados em tempo de execução. O simples fato de realizar a divisão de uma tarefa não garante proporcionalidade quanto ao tempo de execução, pois, devem ser levados em consideração o código a ser paralelizado, tempo de comunicação e sincronismo entre os processos (ELLER JUNIOR, 2013).

A computação paralela tem sido usada cada vez mais por cientistas, engenheiros e pesquisadores para resolverem problemas complexos que normalmente seria desencorajado caso fosse utilizado o processamento convencional (PINTO, 2011). Dividida em três partes principais: a arquitetura de computadores (*hardware*), modelo de programação (*software*) e implementação de aplicações com processamento para-

[1]https://sourceforge.net/projects/openmosix/
[2]http://www.pbsworks.com/
[3]https://research.cs.wisc.edu/htcondor/
[4]http://slurm.schedmd.com/

lelo. A computação paralela pode ser representada por sistemas distribuídos, grades computacionais e *clusters*.

2.6.1 Arquitetura de Computadores Paralelos

A categorização de uma taxonomia para computadores paralelos, já foi proposta por muitos pesquisadores, produzindo resultados mistos e a mais usada ainda é a Taxonomia Flynn (TANENBAUM, 2007). Em seu artigo publicado, Flynn (1972) propôs a categorização baseada em dois conceitos - fluxos de instruções e fluxo de dados, neste sentido, tem-se as instruções executadas em paralelo e o conjunto de dados sob os quais as instruções são submetidas e são divididos em quatro combinações. A figura 6 ilustra organização das arquiteturas definidas por Flynn e a organização dos processadores é distribuída em quatro grupos: SISD (*Single Instruction stream and Single Data stream*), SIMD (*Single Instruction stream and Mutliple Data stream*), MISD (*Multiple Instruction stream and Single Data stream*) e MIMD (*Multiple Instruction stream and Multiple Data stream*).

Figura 6 – Taxonomia de arquitetura para computadores paralelos segundo Flynn (1972).

Fonte: Adaptado de Stallings (2010).

A seguir uma breve descrição quanto às instruções e os dados.

- SISD - caracterizada pelo processamento serial onde a unidade de processamento executa uma única instrução em um conjunto de dados armazenado em uma única memória, chamada de execução serial ou sequencial. São exemplos dessa arquitetura as máquinas baseadas na arquitetura *Von Neuman*, que fazem processamento escalar.
- SIMD - Uma unidade de processamento pode executar uma mesma instrução e opera em múltiplos conjuntos de dados. Os elementos de processamento possui uma memória associada, neste sentido, cada instrução pode ser executada por

processadores diferentes em um conjunto diferente de dados. Os processadores de vetores e matrizes encaixam nesta categoria.

- MISD - Múltiplas unidades de processamento executam diferentes instruções simultaneamente em um mesmo conjunto de dados. São de propósito especiais e úteis se sobre a mesma entrada forem realizadas várias operações diferentes. Essa estrutura não é implementada comercialmente, mas uma das possíveis aplicações são a decodificação de criptografia.

- MIMD - Múltiplas unidades de processamento executam diferentes instruções sobre diferentes conjuntos de dados. Cada processador age de forma independente e paralela, gerando múltiplos fluxos de dados. Nessa categoria estão presentes, *cluster*, SMP (multiprocessador simétrico) e sistemas NUMA (acesso à memória não uniforme) e computadores pessoais com processadores de vários núcleos. Sistemas paralelos podem ser classificados quanto à organização de memória. A arquitetura MIMD conforme Barney (2016), pode subdividir-se em três categorias: memória compartilhada, memória distribuída e sistema híbridos descritas na sequência.

  (i) Memória compartilhada: Sistemas computacionais com memória compartilhada é um sistema multiprocessador simétrico que compartilha uma área de memória comum e acessa através de uma endereço global. Múltiplos processadores podem operar de forma independente, mas compartilham os mesmos recursos de memória, onde as modificações realizadas pelos processadores tornam-se visíveis a todos. É uma arquitetura fortemente acoplada, através de um sistema local de conexão onde o acesso à memória pode ser realizado de forma diferente e pode ser classificadas como UMA e NUMA.

  a)UMA - Memória de acesso uniforme: Sistema mais comumente usado em multiprocessamento simétrico (SMP). O tempo de acesso à memória é aproximadamente igual para todos os processadores e sempre que uma alteração é feita na memória, imediatamente é atualizada para todos os processadores (ELLER JUNIOR, 2013). A figura 7 ilustra a interconexão dos componentes do sistema.

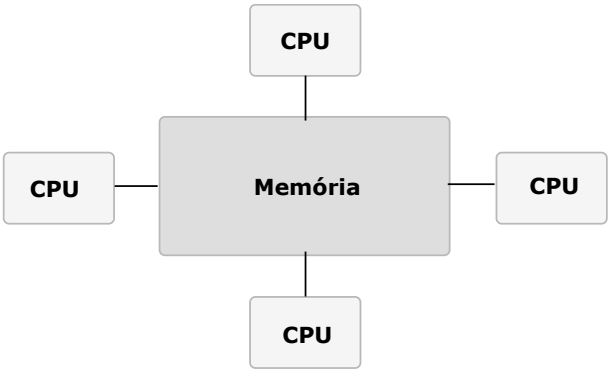


Figura 7 – Arquiteturas de memória de computador paralelas (UMA).

Fonte: Adaptado de Barney (2016).

b) NUMA - Memória de acesso não uniforme: Este sistema consiste pela interconexão de SMPs, onde um pode acessar a memória do outro. Ao contrário do item anterior, o tempo de acesso à memória pode ser diferente para todos os processadores além de ser mais lento devido ao barramento de interconexão (BARNEY, 2016).

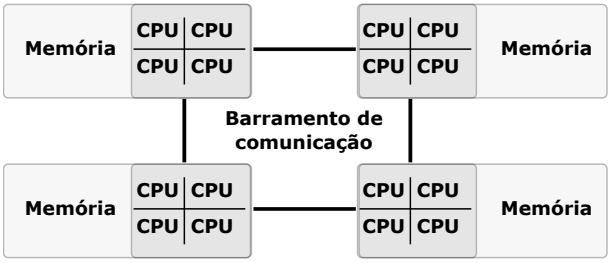


Figura 8 – Arquiteturas de memória de computador paralelas (NUMA).

Fonte: Adaptado de Barney (2016).

Dentre as vantagens de memória compartilhada pode-se enumerar: endereço global de memória que provê uma programação amigável com relação ao uso da memória e rapidez no compartilhamento, dados a proximidade da memória e processador. Por outro lado, a falta de escalabilidade entre memória e CPU caracterizado pela sobrecarga de comunicação é responsabilidade do programador responsável pela construção de sincronização para garantir o acesso à memória global (PINTO, 2011).

(ii) Memória distribuída: Arquitetura composta por unidades de processamento interligadas por uma infraestrutura de comunicação. Cada unidade de processamento possui sua área de memória local e seus endereços não são mapeados por outro processador, pois, o conceito de endereço global não existe (figura 9). As alterações locais de memória não tem efeitos sobre a memória de outros processadores. A figura 9 ilustra o diagrama dessa arquitetura.

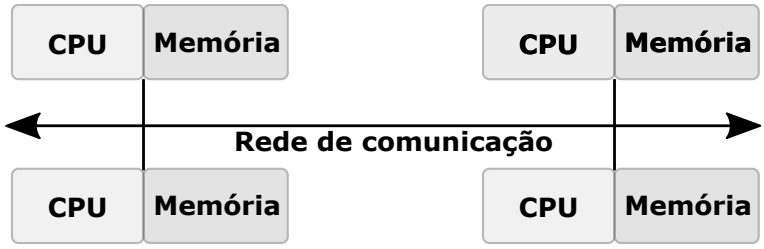


Figura 9 – Arquiteturas de memória distribuída.

Fonte: Adaptado de Barney (2016).

Se houver necessidade de acesso a informações em outras regiões de memória que não seja a local, essa tarefa fica a cargo do programador bem como a sincronização de tarefas através do barramento de comunicação.

Como vantagens do sistema de memória distribuída pode-se enumerar: a escalabilidade - a quantidade memória aumenta com a adição de novas unidades de processamento, uma vez que de cada unidade possui sua própria memória; inexistência de sobrecarga de comunicação e baixo custo comparado a outras arquiteturas. As desvantagens são: dificuldade de mapear estrutura de dados existentes; tempos de acesso à memória não uniforme e responsabilidade do programador por problemas associados com a comunicação de dados entre os processadores.

(iii) Sistemas híbridos: Nestes sistemas ambas as formas de organização de memória dos itens anteriores (UMA e NUMA), adotada segundo Barney (2016) pelos maiores e mais velozes computadores existente atualmente no mundo. Cada equipamento presente na rede possui uma ou mais unidades de processamento que compartilham os mesmos endereços de memória. A memória compartilhada é utilizada na unidade de processamento, enquanto a memória distribuída é compartilhada através do barramento de rede responsável pela interconexão das estações como ilustrado na figura 10 que pode ser composto por CPUs e GPUs (ELLER JUNIOR, 2013).

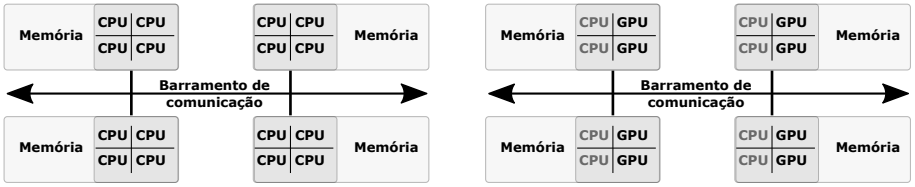


Figura 10 – Modelo híbrido de memória distribuída e compartilhada.

Fonte: Adaptado de Barney (2016).

Como vantagens tem-se a escalabilidade como mais importante, além das vantagens comuns apresentadas em sistema de memória compartilhada e distribuída. A desvantagem está relacionada a aumento da complexidade para o programador.

Os avanços tecnológicos e a redução de preços dos equipamentos de hardware tem permitido a construção de computadores capazes de processar trilhões de operações de ponto flutuante por segundo a um custo cada vez menor. Para a ciência que utiliza cada vez mais recursos computacionais para cálculos e simulação isso é essencial, pois, tarefas que normalmente são processadas em vários dias e requisitam grandes quantidades de memória utilizando sistemas não paralelos, podem ser executado em horas, caso se utilize desses recursos. Um exemplo dessa arquitetura são os *clusters* discutido na subsessão 2.6.4 e HPC (*High-performance computing*) na sessão 2.6.4.3.

2.6.2 Sistemas distribuídos

A evolução tecnológica ocorrida nos últimos 50 anos aconteceu de tal forma que jamais se registrou tal evolução em outras áreas. Essa evolução passa por dois momentos importantes: o desenvolvimento de processadores com capacidade para processar bilhões de operações por segundo; a invenção e modernização das redes de computadores que atualmente pode transmitir bilhões de informações por segundo. Essas tecnologias permitem montar sistemas compostos por vários computadores conectados por uma rede de alta velocidade como sistema distribuídos. *Um sistema distribuído é uma coleção de computadores independentes que se apresenta a seus usuários como um sistema único e coerente* segundo (TANENBAUM; STEEN, 2007).

Em sistemas distribuídos, dois aspectos importantes estão envolvidos. Em relação ao *hardware* os computadores devem ter autonomia, podendo ser homogêneos ou heterogêneos, multiprocessados ou não. O software tem a função de transparecer um sistema único para o usuário.

Para que computadores e redes heterogêneas sejam vistos como um sistema único, os sistemas distribuídos costumam ser organizados por uma camada de *software* situada entre a camada de nível mais alto e o sistema operacional, esse *software* é denominado de *middleware* vide figura 11 (TANENBAUM; STEEN, 2007).

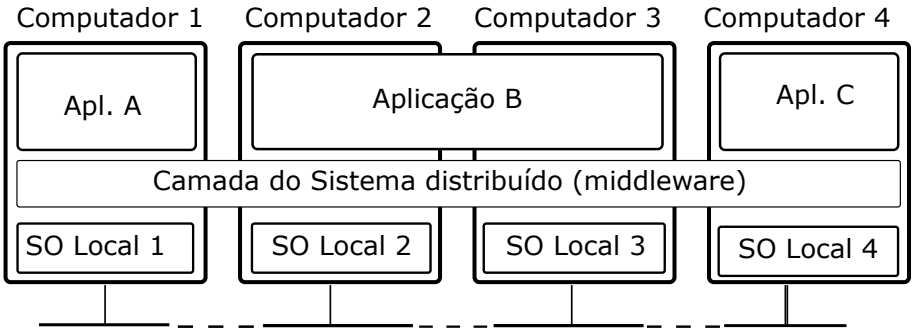


Figura 11 – Sistema distribuído organizado como *middleware*. A camada de *middleware* se estender por várias máquinas e oferece a mesma interface a cada aplicação.

Fonte: Adaptado de Tanenbaum e Steen (2007).

Sistemas providos com essa camada intermediária (*middleware*) atende à definição de sistema distribuídos na definição anterior. O objetivo de um sistemas distribuído é permitir aos usuários e aplicações que tenham acesso a recursos remotos e compartilhados de maneira controlada e eficiente (TANENBAUM; STEEN, 2007).

Construir sistemas distribuídos gera desafios conforme (COULORIS; DOLLIMORE; KINDBERG, 2007):

- Heterogeneidade - construídos a partir de redes, *hardware*, *software*, sistemas operacionais, e linguagens de programação diferentes;
- Sistemas abertos - os sistemas devem ser extensíveis e capaz de integrar componentes desenvolvidos por outros programadores;
- Segurança - proporcionar proteção adequada para os recursos compartilhados e informações sigilosas;
- Escalabilidade - custo constante por usuário adicionado em termos de recursos acrescentados, algoritmos usados em compartilhamento de dados devem evitar gargalos de desempenho e os dados devem ser organizados para obter melhores tempos de acesso;
- Tratamentos de falhas - componentes dependem de outros que podem falhar e devem ser projetados para tratar as falhas adequadamente;
- Concorrência - cada recurso deve ser projetado para manter a consistência nos estados de seus dados;
- Transparência - tornar certos aspectos de distribuição transparentes para o programador de aplicativos.

Nas seções a seguir serão apresentados dois tipos de sistema distribuídos um, os *grids*, chamados de grades computacionais, o outro, os*clusters*, baseados em agregados de computadores.

2.6.3 Computação em grade (*grid computing*)

Um *Grid* consiste em um sistema computacional de alto desempenho, heterogêneo, que provê compartilhamento de recursos distribuídos de organizações geograficamente distribuídas (DANTAS, 2005). Os sistemas de computação em grade como uma das principais características, apresentam alto grau de heterogeneidade, pois, nenhuma premissa é adotada para o *hardware*, sistemas operacionais, redes, políticas de segurança, domínio administrativo, etc. Recursos de diferentes organizações são reunidos para permitir a colaboração de um grupo de pessoas ou instituição e realizada sobre a forma de uma organização virtual. Entre os recursos típicos fornecidos, estão estruturas de supercomputadores, facilidade de armazenamento e banco de dados (TANENBAUM; STEEN, 2007). A figura 12 apresenta um esquema da arquitetura de uma sistema em grade proposta por Foster, Kesselman e Tuecke (2001 apud TANENBAUM; STEEN, 2007, p. 11).

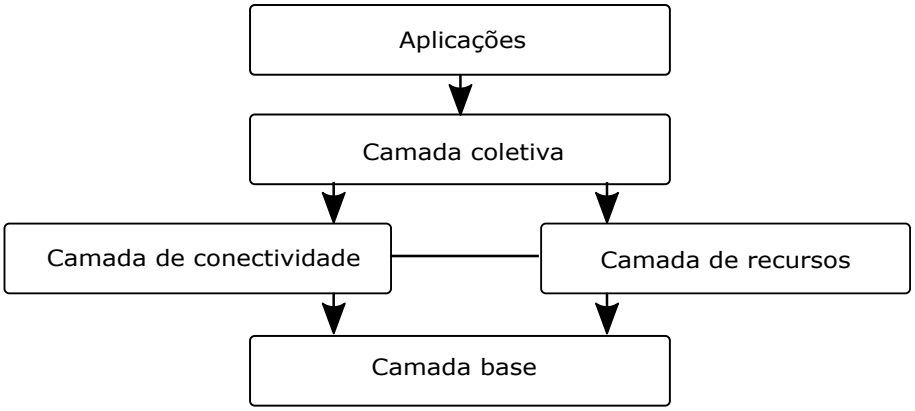


Figura 12 – Arquitetura em camadas para sistema computação em grade.

Fonte: Adaptado de Tanenbaum e Steen (2007).

A arquitetura consiste de quatro comadas. Apresentando as camadas da inferior para a superior, a *camada base* provê interfaces para recursos locais e são projetadas para permitir o compartilhamento de recursos dentro da organização virtual. A *Camada de conectividade* consiste na utilização de protocolos de comunicação para suportar transações que abranjam a utilização de múltiplos recursos, como transferência de dados entre recursos, autenticação de usuários. A *Camada de recursos* é responsável pelo gerenciamento de um único recurso, o controle de acesso e depende da autenticação realizada na camada de conectividade. A *Camada de aplicação* consiste nas aplicações disponíveis dentro de uma organização virtual que fazem uso do ambiente de computação

em grade. As camadas coletiva, conectividade e recursos formam o principal agente de sistemas em grade, a camada de *middleware*, pois, essas camadas dão acesso aos principais recursos que estão distribuídos por vários pontos distantes geograficamente (DANTAS, 2005; TANENBAUM; STEEN, 2007).

2.6.4 *Cluster*

Um *cluster* é definido como um conjunto formado por *hardware* e *software* local interligado por um sistema de comunicação com o objetivo de solucionar os problemas de uma organização, sendo um recurso computacional unificado e transparente ao usuário. É um sistema totalmente dedicado e sua construção tem o objetivo de redução tempo na realização de tarefas através de processamento paralelo. Surgiu como uma alternativa de baixo custo se comparado aos supercomputadores e capaz de atender demandas principalmente na área acadêmica. O desempenho dos *clusters* atuais, além da melhoria do hardware, o aumento da capacidade de transmissão das conexões de redes locais tem sido fundamental (DANTAS, 2005; BARNEY, 2016).

Um exemplo bastante conhecido de *clsuter* é o modelo *Beowulf* baseado em Linux e seu esquema de configuração geral pode ser visto na figura 13 é composto por um conjunto de nós e acessados através de um único nó mestre. O nó mestre é responsável pela alocação de nós, manutenção da fila de *jobs*, em proporcionar um sistema transparente para os usuários, além de executar o *middleware*, responsável pela execução de programas e gerenciamento. Uma parte desse *middleware* é formada pelas bibliotecas para execução de programas paralelos.

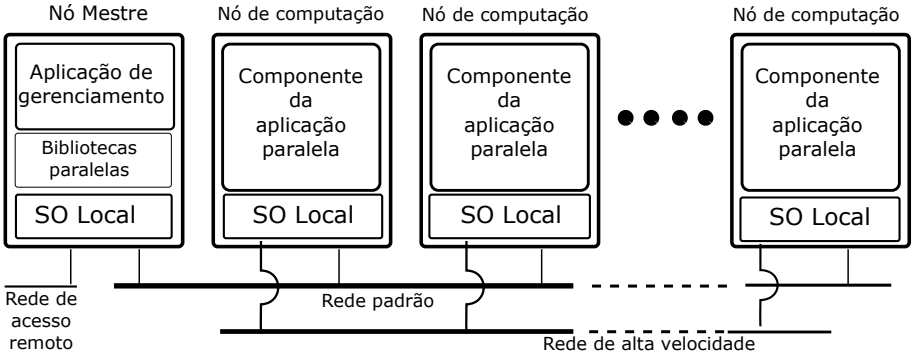


Figura 13 – Exemplo de um sistema de computação de *cluster* Beowulf.

Fonte: Adaptado de Tanenbaum e Steen (2007).

Pode ser formado por computadores pessoais ou infraestrutura apropriada e ter autonomia, pois, o sistema deve ser capaz de funcionar de forma independente e cada dispositivo que compõe o *cluster* é chamado de nó. É formado por um conjunto de nós de computação controlados, acessados e gerenciados através de uma entidade central

(*master*) fazendo com que os nós (*slaves*) trabalhem juntos como única entidade física (TANENBAUM; STEEN, 2007).

Em um projeto que envolve a implantação ou construção, deve-se levar em conta os benefícios e objetivos fornecidos por ele. Segundo Stallings (2010) quatro são os principais objetivos dessa arquitetura:

- Escalabilidade absoluta: pode-se construir *clusters* grandes capazes de processar um volume de informações superior à capacidade de uma computador de grande porte que trabalha sozinho.
- Escalabilidade incremental: um *cluster* pode ser configurado de forma que outros equipamentos possam ser agregados à estrutura em pequenos incrementos. Dessa forma, um sistema modesto pode ser expandido sem a necessidade de substituição de um sistema pequeno por um maior. A escalabilidade é um fator diferencial, segundo Dantas (2005), pois a configuração pode crescer à medida que mais recursos forem disponibilizados, sendo possível aumentar o desempenho através de configurações com boa relação custo-benefício.
- Alta disponibilidade: cada equipamento que compõe um *cluster* é independente e, em caso de falha não há interrupção do funcionamento do outros equipamentos que compõem a estrutura. A tolerância a falhas é tratada de forma automática por software na maioria dos equipamentos.
- Preço/Desempenho: através da ideia de blocos de construção é possível montar um *cluster* com poder computacional igual ou superior de máquinas de grande porte por um custo menor.

Apesar da classificação de Flynn para arquitetura de computadores é comum efetuar a classificação de *clusters* através da observação de alguns aspectos conforme Dantas (2005). Esses aspectos incluem pontos relevantes como: o limite geográfico, utilização do nós, tipo de tipologia e hardware, aplicações e tipo de nós.

- Limite geográfico: O uso de *clusters* tem crescido nas organizações e diferentemente da computação em grade ou distribuída, a maioria ocupam espaços em salas específicas ou laboratórios, ou seja, locais com limitação geográfica. Destaca-se pela simples agregação de computadores comuns ou dispositivos especiais e, em geral, os usuários tem boa noção da importância de configuração e políticas de segurança para prover alto desempenho das aplicações. Essas políticas devem ser bem definidas e existirem independentemente do espaço físico.
- Utilização do nós: para utilização dos nós deve ser estabelecida a participação dedicada ou não dedicada dos nós que compõem o conjunto. Estão relacionadas a

utilização dos nós as políticas de gerenciamento, segurança, alta disponibilidade, escalonamento de processos, balanceamento de carga, *middleware*. Pode ser estabelecida pelo tipo de participação dos nós que compõe o *cluster*, sendo esta dedicada ou não dedicada. Nos *clusters* definidos como não dedicados, são aproveitados os períodos em que os processadores encontram-se ociosos para executar aplicações. Em geral são computadores interligados por meio de uma rede local ou não e que dispõem de aplicativos e softwares para atender demandas de usuários. Nos *clusters* dedicados, em geral, compartilham um único meio de comunicação (LAN) através de um *switch* e executam aplicações submetidas ao *cluster* (DE FARIA, 2015).

- Tipos de hardware: as características aqui citadas são relacionadas com o *hardware* empregado. A composição desses agregados podem ser com estações de trabalho heterogêneas distribuídas em uma rede local, estações para *cluster* que constituem máquinas homogêneas dedicas para aplicações específicas e máquinas com arquitetura SMP.

- Aplicações alvo: aplicações diversas tem requisitos diferentes. As aplicações podem necessitar de alto poder de processamento e neste caso os requisitos são que os *clusters* sejam dotados de inúmeros processadores, grande quantidade de memória e espaço em disco. Outros tipos de aplicações não podem sofrer interrupções e requerem alta disponibilidade da infraestrutura. Importante ressaltar que existem aplicações que demandam além do alto poder de processamento, também alta disponibilidade.

- Tipos de nós: essa caracterização refere-se quanto a similaridade de *software* e *hardware* que compõe o *cluster*. Os agregados homogêneos são formados por computadores de arquitetura semelhantes como PCs e SMPs e deve ser a opção de projetos quando é possível estabelecer parâmetro de similaridade. Já os agregados de computadores heterogêneos são formados por computadores de arquiteturas diferentes, seja em *software* ou *hardware*. Este tipo de agregado é característico de projetos que crescem de forma modular e o desafio é garantir que a heterogeneidade seja transparente e o funcionamento correto. Esse tipo de projeto pode causar erros em aplicações onde há necessidade de precisão em cálculos com ponto flutuante. Como o objetivo é a distribuição igualitária de tarefas em função da quantidade de nós, quando existe capacidade de processamento diferente é preciso analisar e utilizar um algoritmo de balanceamento para distribuir a carga proporcionalmente (ELLER JUNIOR, 2013).

Os *clusters* podem ser caracterizados quanto modo de trabalho em: Alta Disponibilidade (*High Availability*), Balanceamento de Carga (*Load Balancing*) e Alto Desempenho (*High Performance Computer*) (ELLER JUNIOR, 2013).

2.6.4.1 *Cluster* de alta disponibilidade

São *clusters* que tem por objetivo prover serviço por período de tempo longo independentemente das condições de uso ou falhas. Trata-se de um serviço redundante onde um nó pode realizar tarefas de outros nós. Em seu projeto deve ser previstas possíveis vulnerabilidades ou falhas. Este tipo de equipamento na maioria das vezes está relacionado com serviços *WEB* como banco de dados, servidores de hospedagem emails, DNS, etc (ELLER JUNIOR, 2013).

2.6.4.2 *Cluster* de balanceamento de carga

Apesar de ser considerado um *cluster* de alta disponibilidade, este tipo de *cluster* tem por característica aumentar a capacidade de processamento de tarefas de seus nós e redução da latência através da distribuição de processos de forma equilibrada e evitando a ociosidade dos processadores. Algoritmos especializados são utilizadas para que o balanceamento possa ocorrer com efetividade. Dois tipos de balanceamento podem ser usados, o estático e o dinâmico. O balanceamento estático, que consiste em distribuir uma quantidade igual de problemas para os processadores, apesar de fácil implementação, não garante equilíbrio. Não há garantia de tempos de processamento igual em problemas semelhantes. O balanceamento de carga dinâmico por sua vez fornece resultados melhores, pois um processo fica responsável pelo controle e distribuição de tarefas à medida que os processadores finalizam, porém é mais difícil a sua implementação (PINTO, 2011).

2.6.4.3 *Cluster* de alto desempenho (HPC)

É um tipo de agregado computacional que tem como finalidade processar milhares de operações e problemas de alta complexidade através do processamento de tarefas simultâneas (DANTAS, 2005). O poder computacional desses equipamentos, além de otimização na resolução de problemas complexos, tem custo pequeno se comparado a supercomputadores (ELLER JUNIOR, 2013). O processamento de alto desempenho durante muito tempo foi baseado em computadores paralelos específicos com arquitetura massivamente paralelas (MPP) e os de memória compartilhada (SMP), mas devido ao custo dessas máquinas e a oferta de computadores pessoais existentes, nas instituições, a política de agregar máquinas tem sido cada vez mais utilizadas. Esse paradigma tem sido usado cada vez mais pelas organizações devido ao excelente custo benefício e a possibilidade de dispor de maiores e melhores recursos computacionais (BRASIL, 2006).

2.6.5 Métricas de desempenho em Computação Paralela

Otimizar recursos computacionais em busca de performance, são características da utilização de computação paralela e distribuída. Em busca dessa otimização os *clusters* tem sido usados em escala cada vez maior. A finalidade dos *clusters* de alto desempenho é o processamento de milhares de operações em tempo reduzido, métricas devem ser usadas para medir a performance do equipamento (BARNEY, 2016). Existem duas classes distintas de métricas de desempenho (STALLINGS, 2010). O desempenho de processadores e desempenho das aplicações paralelas:

- Desempenho de processadores - avaliam a performance do processador baseado na velocidade e o números de operações que este processador é capaz de realizar em um espaço de tempo.
- Desempenho das aplicações paralelas - avaliam a performance de uma aplicação paralela comparando o tempo gasto por uma único processador e o tempo gasto por múltiplos processadores. Este é um dos focos desse trabalho e as métricas são apresentadas na sequência.

Ao submeter um problema num sistema paralelo, pretende-se verificar o ganho de desempenho que se tem sobre a versão serial. Duas métricas serão abordadas neste trabalho, o *speedup* (S) e a eficiência (E).

O *speedup* é o ganho obtido de velocidade e processamento quando uma aplicação é executada com n processadores. Quanto maior o valor dessa métrica, mais rápido é a parte paralela do código. Essa métrica, em termos gerais, pode ser dada pela razão entre o tempo serial e o tempo paralelo como mostrado na equação:

$$S_n = \frac{T_s}{T_n}, \tag{2}$$

onde T_s representa o tempo serial e T_n o tempo paralelo. Por maior que seja a quantidade de processadores disponíveis, o *speedup* é limitado e decorre da parcela serial existente no código. Segundo (STALLINGS, 2010), *a lei de Amdahl foi proposta inicialmente por Gene Amdahl (1967) e lida com o potencial speedup de um programa usando múltiplos processadores em comparação com um único processador*. A Lei de Amdahl define o *speedup* teórico máximo em função da porção de código que pode ser paralelizada e mostra que o *speedup* não é proporcional ao aumento de processadores. Se p é a porção paralelizável de um código, o aumento do número de processadores influenciará apenas na parte paralelizável conforme a equação

$$S = \frac{1}{1-p}, \tag{3}$$

onde, p representa a porção paralelizável do código. Ao introduzir o número de processadores que executam a fração paralelo, a relação pode ser modelada conforme a equação:

$$S = \frac{1}{(\frac{p}{n}) + s} \tag{4}$$

onde p representa a fração paralela, n número de processadores e s a fração serial. Logo, evidencia que há limites para escalabilidade de paralelismo. O aumento da quantidade de processadores, transforma a equação 4 na equação 5, pois quando n torna grande, a razão p/n tende a zero. Logo,

$$S = \lim_{n\to\infty} \frac{1}{(\frac{p}{n}) + s} = \frac{1}{s}. \tag{5}$$

Ao final, com base nas equações resume-se que há limites na quantidade de processadores necessários para produzir o *speedup* teórico máximo (BARNEY, 2016).

A eficiência (E) é a razão entre o *speedup* S e o número de processadores n utilizados na execução paralela. Esta métrica indica o percentual de quanto do paralelismo foi explorado no algoritmo. Quanto maior a porção paralelizável, maior é a eficiência da arquitetura na execução do programa.

$$E = \frac{S}{n} \tag{6}$$

A eficiência mostra se os ganhos obtidos com a adição de n processadores são relevantes, a ponto de determinar a quantidade máxima que produz melhores resultados.

A "lei de Amdahl" presume a porção do código que será paralelizado, entretanto, no trabalho atual com a técnica de *job splitter*, utilizar-se-a o cálculo do *speedup* com base somente nos tempos obtidos com a execução serial e paralelo. Neste trabalho, como não é possível determinar a porção paralela, o *speedup* foi calculado com base na equação 2.

2.7 Gerenciamento de Recursos Computacionais

A qualidade de serviço fornecida por um sistema para uma aplicação não depende apenas dos recursos suficientes para promover o desempenho. Precisam estar disponíveis para a aplicação quando forem necessário e, para haver disponibilidade os recursos devem ser escalonados. Tarefas devem ter os recursos atribuídos de acordo com sua prioridades e um escalonador de recursos determina a prioridade de tarefas baseada em critérios, como por exemplo, o tempo de resposta. (COULORIS; DOLLIMORE; KINDBERG, 2007).

O escalonador, de forma geral, deve garantir que as tarefas executem utilizando o máximo de recursos disponíveis, e termine de executar no menor tempo, respeitando as restrições de tempo ou políticas aplicadas às tarefas (TEODORO, 2013).

Gerenciadores de recursos tem a finalidade de garantir a estabilidade dos sistema, definindo qual tarefa será executado primeiro com base na prioridade ou requisição de recursos. Em sistemas paralelos os gerenciadores são desenvolvidos para atender a várias solicitações em arquiteturas com diferentes propósitos. A função *job splitter* do GATE/Geant4 tem suporte para o gerenciador PBS *PBS-Portable Batch System* dentre outros (COLLABORATION, 2014).

2.7.1 *Simple Linux Utility for Resource Management* (Slurm)

O SLURM é um gerenciador de *clusters* altamente escalável, tem suporte para tolerância a falhas e é um sistema de planejamento de tarefas para grandes estruturas computacionais destinadas ao processamento paralelo. Gerencia a utilização global de recursos através de uma fila onde ficam os trabalhos pendentes. É capaz de administrar nós computacionais de forma exclusiva ou não, distribuir tarefas para um conjunto de nós e fazer o monitoramento da execução à conclusão. Iniciou como gerenciador de recursos de *software* livre, sendo desenvolvido por várias empresas em um processo colaborativo e atualmente é um gerenciador presente nos principais supercomputadores do mundo (SHEDMD, 2016; JONES, 2014).

2.7.1.1 Arquitetura SLURM

A arquitetura de gerenciamento de *clusters* implementada pelo SLURM (figura 14) possui um par de controladores redundantes responsáveis pelo gerenciamento do *cluster* e implementados através de um *daemon* chamado *slurmctld*. Este *daemon* consiste em três susbsistemas com funções específicas. O gerenciador de nó monitora o estado e configurações de cada nó no *cluster*. O gerenciador de partição agrupa nós em conjunto e controla o acesso a cada partição e aloca nós com base nas propriedades de cada partição. O gerenciador de *jobs* aceita as requisições e coloca na fila ordenada por prioridades. Seu papel é monitorar os recursos e mapear tarefas recebidas para recursos subjacentes de computação e distribuir as tarefas entre os nós (JONES, 2014).

Cada nó que faz parte da infraestrutura implementa um *daemon slurmd* que gerencia cada nó em que é executado, monitora tarefas do nó e o estado atual de cada máquina, aceita trabalhos do controlador, mapeia os trabalhos para tarefas em cada núcleo dentro do nó, execução remota, controle dos *jobs* e serviço de controle de fluxo de informações. Existem outros *daemons* dentro da arquitetura para controle de funções para autenticação, banco de dados e configurações (BULL CEDOC, 2010).

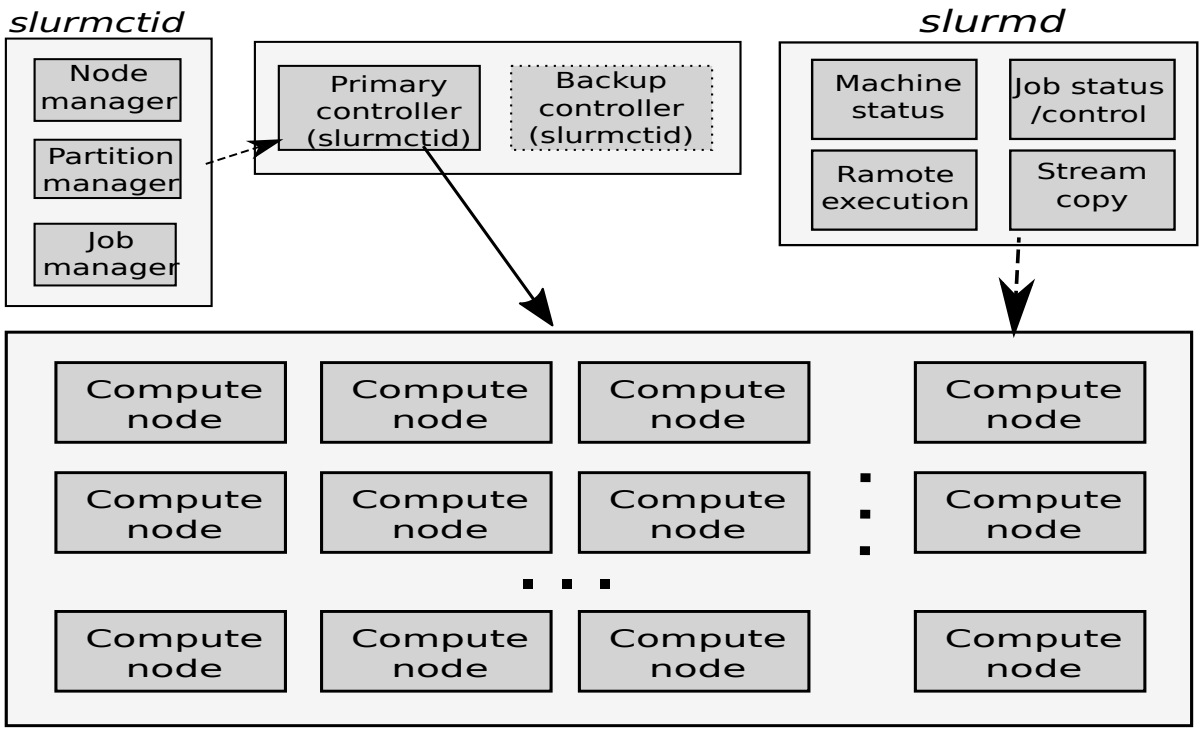


Figura 14 – Visualização de alto nível da arquitetura do SLURM.

Fonte: Adaptado de Jones (2014)

2.7.1.2 Particionamento SLURM

Um conjunto de nós pode ser alocado em um grupo lógico chamado partição, que normalmente é uma fila de tarefas. As partições são configuradas de forma que controlam o tempo de acesso, restrições para usuários, limite de tempo, tamanho do *job*. Dentro de uma partição é possível mapear um conjunto de nós para um usuário por um período de tempo para realizar uma tarefa. O particionamento de recurso do SLURM, como ilustrado na (figura 15) pode alocar nós próximos para assegurar baixa latência de comunicação entre os nós de cooperação. Uma vez que uma tarefa é atribuída a um conjunto de nós o usuário é capaz de iniciar uma tarefa paralela (JONES, 2014; BULL CEDOC, 2010).

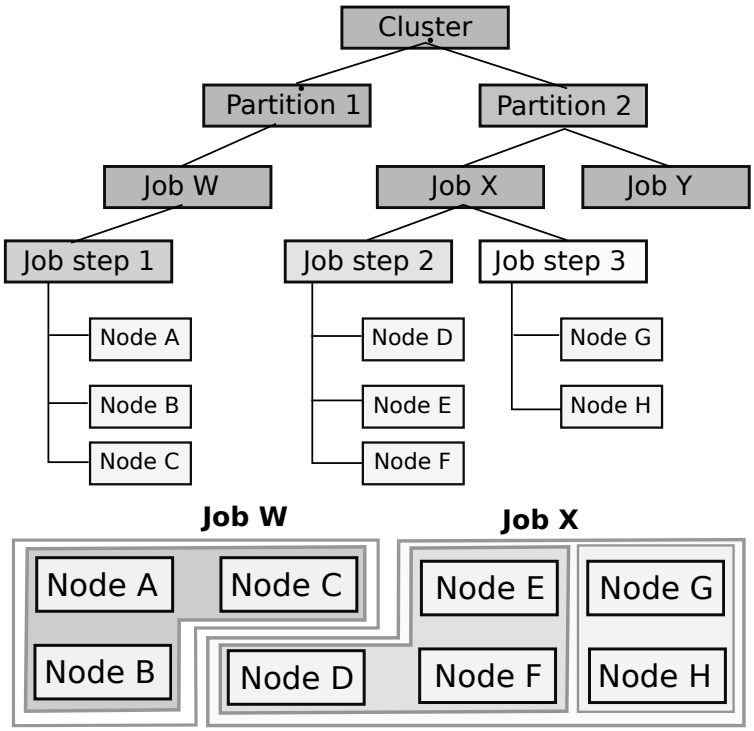


Figura 15 – Particionamento de recursos no SLURM.

Fonte: Adaptado de Jones (2014)

2.7.1.3 Compatibilidade do SLURM com outros gerenciadores

Cada gerenciador de recursos apresenta funcionalidades e recursos específicos, entretanto, possui funcionalidades comuns entre eles. São variáveis de ambiente e opções de especificação de trabalho utilizado pelos principais sistemas de gerenciamento de carga de trabalho: PBS / Torque, Slurm, LSF, SGE e LoadLeveler. Cada um destes gerenciadores de carga de trabalho tem características únicas e as funcionalidades mais comuns estão disponíveis em todos estes ambientes, como mostra a tabela do anexo A.

3 Métodos e Discussão dos Resultados

Neste capítulo são descritos os aspectos principiais e procedimentos adotados para realizar as simulações, recursos utilizados, tarefas executadas e configurações de *hardware* disponíveis. Está dividido em três partes: Instalação, alidação e paralelização do GATE/Geant4 nos *clusters*; Estudo de desempenho com (1 CPU) e Análise de desempenho da paralelização.

Os *scripts* utilizados na manipulação dos arquivos e mencionados neste capítulo foram incluídos nos Apêndices permitindo a sua utilização em outros trabalhos.

3.1 Características de *hardware* e *software* dos *clusters*

Tarefas ou processos computacionais podem responder de forma diferente em função das características de *hardware* e *software* utilizados. Nesta seção as principais características dos equipamentos utilizados neste trabalho são apresentadas para o *cluster* CTR e CACAU.

Tabela 1 – Principais características de hardware dos *clusters* CTR e CACAU

Características CTR	Características CACAU
Cluster com 8 nós, composto por um processador Intel com quatro núcleos e *clock* de 2.8 GHz; Memória RAM de 16 GB para máster e 8 GB para cada *slave*; Capacidade de armazenamento de 1 TB no máster e 360 GB nos *slaves*, com sistema RAID ; *Interface* de rede *Ethernet* 1 Gb/s	Cluster com 20 nós composto por dois processadores *quad-core Xeon* E5430 de 2.66 GHz; Memória RAM de 16 GB para máster e 16 GB para cada *slave*; Capacidade de armazenamento de 2 TB no máster e 360 GB nos *slaves* com sistema RAID; *Interface InfiniBand* e rede *Ethernet* 1 Gb/s.

Após consulta bibliográfica, foram instalados a lista de *softwares* da Tabela 2 nos *clusters* CTR e CACAU.

Tabela 2 – Principais características de software dos *clusters* CTR e CACAU

Características CTR	Características CACAU
O Sistema operacional CENTOS Linux release 7.2; Plataforma GATE versão 7.1; Plataforma GATE versão 7.1; Compilador Gcc versão 4.9; Biblioteca CLHEP versão 2.2.0.4; Plataforma ROOT versão 5.34; Gerenciador SLURM *Workload Manager* versão 15.08; Pacotes e bibliotecas requeridos para a distribuição do sistemas operacional e versões de *softwares*.	Sistema operacional Red Hat Enterprise Linux Server release 6.3 (Santiago); Plataforma GATE versão 7.1; Plataforma Geant4 versão 4.10.1; Compilador Gcc versão 5.0; Biblioteca CLHEP versão 2.2.0.4; Plataforma ROOT versão 5.34; Gerenciador SLURM *Workload Manager* versão 15.08; Pacotes e bibliotecas requeridos para a distribuição do sistemas operacional e versões de *softwares*.

3.2 Instalação da plataforma GATE/Geant4

A primeira etapa deste trabalho foi realizada no CTR com a preparação da infraestrutura computacional do *cluster* para instalação e configuração do GATE/Geant4. Após a conclusão da instalações das ferramentas, bibliotecas e aplicativos requeridos para as atividades, iniciou-se a consulta bibliográfica sobre o conjunto de ferramentas GATE/Geant4. A fonte de estudo principal foi a documentação do GATE (JAN et al., 2014; COLLABORATION et al., 2016) e Geant4 (COLLABORATION, 2013; COLLABORATION, 2014).

Um procedimento recomendado após a instalação da plataforma GATE/Geant4 é o teste de *Benchmarking* para validar a instalação. O teste de *Benchmarking* inclui a simulação de três macros (gama, carbono e próton) e a geração do gráfico da Figura 16.

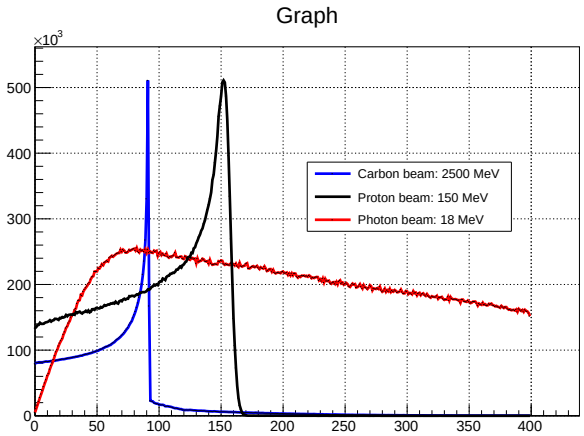


Figura 16 – *Benchmarking* de validação da instalação GATE/Geant4. Representação gráfica das partículas gama, próton e íon de carbono.

A validação foi feita com base nas instruções e arquivos de simulação que acompanham a instalação do GATE, presentes no diretório:
caminho_Gate_7.1/benchmarch/benchRT/output/.

3.3 Paralelização nos *Clusters*

Esta subseção descreve procedimentos de instalação do GATE e ajustes para execução de tarefas em paralelo no CTR e CACAU.

3.3.1 Paralelização do GATE com *job splitter*

A técnica *job splitter* consiste em dividir um arquivo de simulações em partes e distribuir as tarefas entre os processadores. Para isso é necessário compilar o *gjs* (*Gate job splitter*) e o *gjm* (*Gate job merger*), necessários para dividir as macros e combinar os arquivos *.root com resultados, respectivamente.

Para submeter *jobs* ou processos paralelos é necessário um escalonador de recurso que mantém o controle dos processos em execução. O GATE/Geant4 atualmente tem suporte para as plataformas openMosix, openPBS, Condor e Xgrid (COLLABORATION et al., 2016), entretanto, os *clusters* existentes na UESC tem como gerenciador de recursos o SLURM. Foi feito estudo de possibilidade de uso do SLURM e com base nas características comuns entre a plataforma openPBS (Anexo A) foi implementada uma aplicação em linguagem *Bash Script* para criar *scripts* que possibilitassem a submissão de *jobs* com SLURM (Apêndice A) (RODRIGUES, 2011; SCHEDMD, 2013).

Desenvolver uma solução para submissão de tarefas GATE/Geant4 em *clusters* com gerenciador SLURM é muito importante para o grupo de pesquisa em Física Médica da UESC, pois, viabiliza a continuidade das pesquisas em Radioterapia e a realização de simulações em tempo reduzido através da técnica *job splitter*.

3.4 Métodos e procedimentos para realização das simulações

Foram realizadas simulações seriais para estudar e avaliar a incerteza da dose em função do número de histórias. Para esta etapa foi utilizado um fantoma de *Box* representado por um cubo geométrico de 100 cm (100x100x100) definido nas funções do GATE. Cada simulação foi realizada com a partículas gama, próton e carbono. Para cada partícula, o número de histórias variou de 10^5 até 10^8. Para estimar a dose, o fantoma foi dividido em *voxels* de 1 cm e uma fonte de energia isotrópica foi posicionada instalada no centro do fantoma. A energia definida para cada partícula foi de: 200 MeV para

o próton, 18 MeV para o gama e 2500 MeV para íon de carbono. A análise foi feita quando as simulações realizaram $10^5, 10^6, 10^7$ e 10^8 histórias.

3.4.1 Definição de parâmetros na macro de simulação

As simulações com o GATE podem ser feitas pelo usuário diretamente no terminal em caso de sistema operacional Linux. Essa forma é desaconselhada a depender do tamanho e características simulações. Outra alternativa é a criação de arquivos ou *scripts* com todas as definições e parâmetros que envolve cada simulação. Neste trabalho foram criados *scripts* (macros), sendo suas variações feitas com modificação de parâmetros específicas de acordo o tipo de simulação a ser realizada. As macros com os parâmetros de simulação devem ser criados em arquivos com a extensão *.*mac* para que o GATE processe corretamente. Para elaborar uma simulação alguns parâmetros importantes devem ser definidos.

Para criar a macro de simulação vários passos devem ser definidos:

- Visualização da simulação - O GATE permite que uma simulação seja visualizada em tempo real. Durante a simulação pode ser observado a posição da fonte, ângulos de visualização, pontos específicos da através do deslocamento da imagem, trajetória da partícula estudada e objeto simulador. Um dos visualizadores mais utilizados no GATE é o OpenGL[1] (*Open Graphics Language*). A ativação da visualização no GATE é feita através do comando:

  /vis/open OGLSX

  Ao ativar visualização o ambiente simulado é exibido em três dimensões. Durante a preparação da simulação, a visualização é importante porque possibilita ajustes, entretanto, utiliza muito recurso de processamento tornando as simulações mais demoradas, ou seja, a visualização interfere no tempo total de execução. Caso a visualização não seja acompanhada durante o processamento é recomendável desativar esta opção.

- Verbose - Durante a simulação várias informações podem ser emitidas e exibidas no terminal. Este parâmetro produz muitas informações aumentando espaço de armazenamento devido ao fato do gerenciador de recursos registrar o comportamento de cada tarefa executada além de aumentar o tempo de processamento. Esta opção pode ser desabilitada.

- Mundo (espaço de simulação) - A definição do espaço onde ocorre a simulação é chamado de *world*, ambiente tridimensional e que abriga tudo que está relacionado à simulação. Neste espaço da simulação, deve ser definido o material do qual é

[1] https://www.opengl.org/

composto (em geral o ar), o local onde está o banco de dados com as características de todos os materiais envolvidos.

/gate/geometry/setMaterialDatabase caminho_arquivo/GateMaterials.db

O material tem suas características definidas, como por exemplo aos elementos químicos de sua composição, densidade, número de massa, número atômico, estado, etc. Os materiais cujas características não estão presentes no GATE, podem ser definidas pelo usuário. As partículas utilizadas nas simulações (gama, carbono e próton) estão definidas nos materiais do GATE.

- Definição da geometria do fantoma - O fantoma é o objeto em que as partículas proveniente da fonte promove a interação com o meio. Nele deve ser definido a forma geométrica, dimensões e material. O fantoma pode ser um modelo existente no GATE, uma imagem produzida pelo usuário ou adquirida através de tomografia computadorizada. Os fantomas deste trabalho foram definidos do GATE conforme o Apêndice B.

- Processos físicos - Uma vez definido o volume é necessário definir o processos físicos envolvidos na simulação devendo ser feito para cada partícula.

- Cortes (*cuts*)- Nesta etapa são definidos alguns parâmetros como valores mínimos de energia, distância máxima que o fóton ou partícula deixa de ser rastreado depositando a energia restante no ponto.

- Fonte de energia - No GATE a fonte é representada por um volume em que as partículas (íons, prótons, pósitrons, gama) são emitidos. Nesta etapa pode ser definida características como energia da fonte, direção da emissão, distribuição da energia e a atividade.

- Definição dos atores (*actors*) - A definição dos atores permite a geração de informações relevantes para as análises dos resultados das simulações. Nesta etapa define-se: o tempo em que os resultados são gravados nos arquivos, a distribuição da dose de radiação, energia depositada, resolução utilizada no fantoma, incertezas da dose e energia, tempo de atualização e armazenamento de dados estatísticas. O formato de saída dos dados é configurado em modo de compatibilidade com as ferramentas que serão usadas nas análises.

- Configuração de medições - Nesta etapa é possível determinar o tempo para realizar uma simulação, número mínimo de partículas, a aquisição do número da semente e principalmente o números de histórias, ou seja, a quantidade de interações da partícula.

3.5 Análise do erro em função do número de histórias e o raio do fantoma

Nesta etapa deseja-se obter a incerteza da dose em função do raio e números de histórias. Nessas simulações devem observados comportamento do erro da dose em função do número de histórias. A partir dessa análise será possível com base nas dimensões do fantoma, estimar o número de histórias necessários para se obter um erro menor que 5%. As simulações foram realizadas com fantoma de *Box*, partículas gama, carbono e próton. As simulações foram submetidas com 10^8 histórias, sendo os dados coletados a cada hora, nas primeiras 24 horas, quatro horas nas vinte e quatro horas seguintes e, finalmente, a cada doze horas até o final das simulações. Os valores analisados para à partir do centro do fantoma em 5 cm, 10 cm e 25 cm para cada das simulações.

Os arquivos de saída com os resultados no formato .mhd/raw, foram analisados com o *software* VV[2]. As incertezas foram verificadas a partir de resultados obtidos em imagens.

3.5.1 Resultados e discussão

As figuras 17, 18 e 19 representam as intensidades das incertezas para gama, carbono e próton, respectivamente. Esse procedimento permitiu saber o número de histórias necessário para validar uma simulação em doses baixas, baseado na dimensão do fantoma e problema analisado.

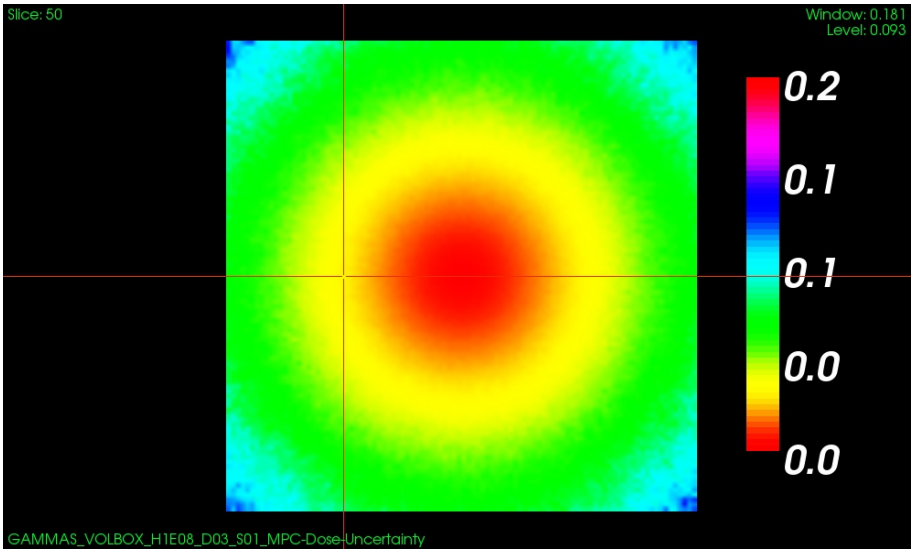


Figura 17 – Representação da intensidade da incerteza da dose na simulação com partícula gama e fantoma de *box* e 10^8 histórias.

[2]https://www.creatis.insa-lyon.fr/rio/vv

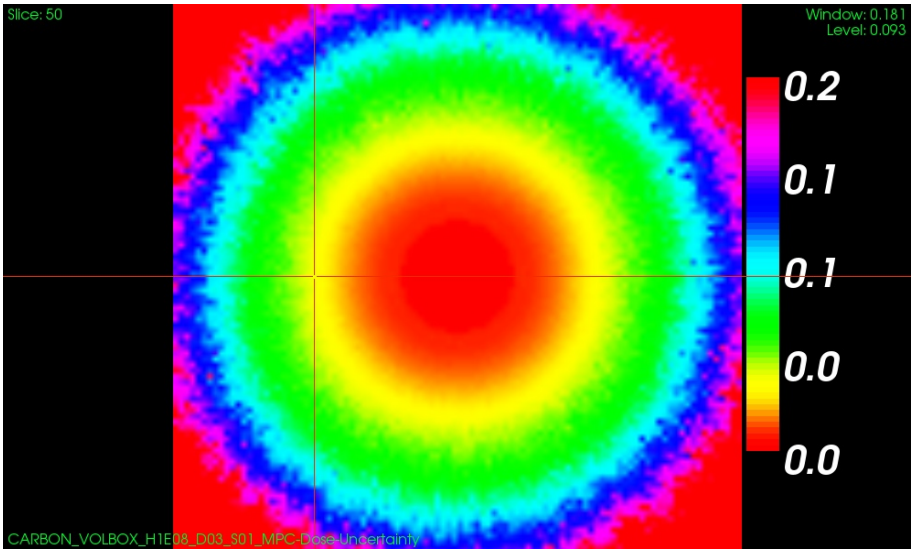


Figura 18 – Representação da intensidade da incerteza da dose na simulação com carbono e fantoma de *box* e 10^8 histórias.

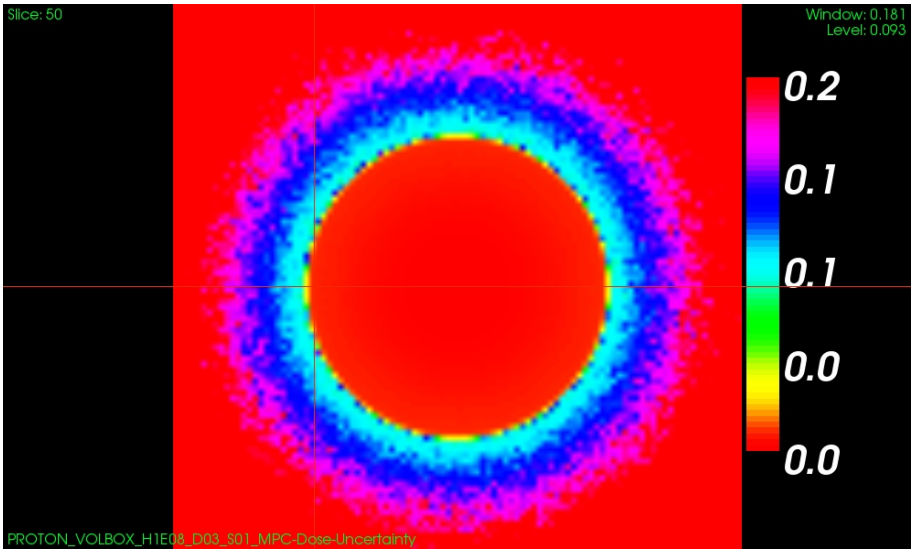


Figura 19 – Representação da intensidade da incerteza da dose na simulação com próton e fantoma de *box* e 10^8 histórias.

Nas figuras acima, a incerteza é menor que 5% a aproximadamente $25cm$ da origem no ponto de intersecção dos eixos. As tabelas 3, 4 e 5 com o resultados obtidos para o fantoma de *box*.

3.5.1.1 Análise do erro em função do raio e número de história para gama

Os resultados obtidos com a partícula gama (Tabela 3), mostram que para um erro menor que 5%, o número de histórias necessário é 10^7 histórias para um raio de até 10 cm e 10^8 histórias para raio 25 cm.

Tabela 3 – Tabela de incerteza da dose fantoma *box* com gama

Número de Histórias	5 cm	10 cm	25 cm
10^5	19,0%	29,0%	57,0%
10^6	6,5%	10,3%	25,6%
10^7	1,3%	2,4%	7,2%
10^8	0,6%	1,1%	2,9%

3.5.1.2 Análise do erro em função do raio e número de história para próton

Os resultados para a partícula próton (Tabela 4), em um raio de até 10 cm são necessários 10^6 histórias para atingir um erro menor que 5% e 10^7 histórias para raio de até 25 cm.

Tabela 4 – Tabela de incerteza da dose fantoma *box* com próton

Número de Histórias	5 cm	10 cm	25 cm
10^5	5,5%	11,5%	32,0%
10^6	2,4%	4,2%	11,3%
10^7	0,7%	1,4%	3,1%
10^8	0,2%	0,4%	1,0%

3.5.1.3 Análise do erro em função do raio e número de história para carbono

Para erro menor que 5% com o carbono, são necessários 10^6 histórias para raio de 5 cm e 10^7 histórias para raio de 10 cm. Para raio de 25 cm foi necessário número inferior a 10^8 histórias. Devido ao erro analisado ser inferior a 5%, a simulação foi interrompida.

Tabela 5 – Tabela de incerteza da dose fantoma *box* com carbono

Número de Histórias	5 cm	10 cm	25 cm
10^5	6,0%	11,2%	58,0%
10^6	1,5%	7,6%	25,0%
10^7	0,5%	2,6%	9,5%
10^8	0,2%	0,9%	3,6%

Comparando aos dados das tabelas 3, 4 e 5), os resultados do *cluster* CTR e CACAU foram similares, mesmo com variações, para os mesmos números de histórias e raio, o erro foi inferior a 5%.

3.6 Estudo de desempenho (1 CPU) - Complexidade da Geometria

Nesta etapa foi realizado o estudo sobre o efeito que a geometria tem sobre o tempo de cálculo. Ela pode ser modelada através de funções matemáticas ou figuras geométricas com fantoma de *voxels*.

Deseja-se estudar o desempenho do GATE com o aumento da complexidade da geometria, o que indica a possibilidade de aumento no tempo. Deseja-se obter a quantificação do tempo e de maneira específica para cada modelo geométrico em cada plataforma. A fonte de energia localizada no centro do cubo de água, é isotrópica, depositando energia do centro para as bordas do fantoma como ilustrado na Figura 20.

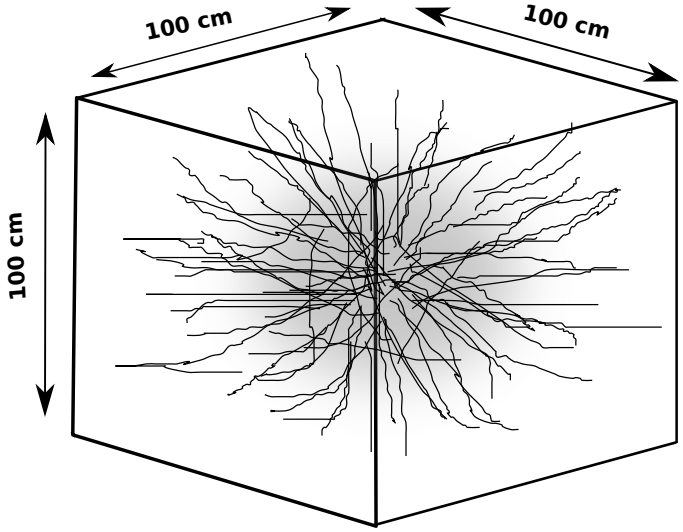


Figura 20 – Ilustração da fonte de energia isotrópica localizada no centro do fantoma.

Nesta etapa, foram definidos três tipos de fantomas: um cubo geométrico de *Box* e dois cubos geométricos de *voxels* com resolução 0,5 e 1,0 cm para cada elemento.

O fantoma de *Box* foi definido através no GATE. Possui dimensão de 100 cm como mostra a Figura 21, água como material e suas definições estão presentes no Apêndice B.

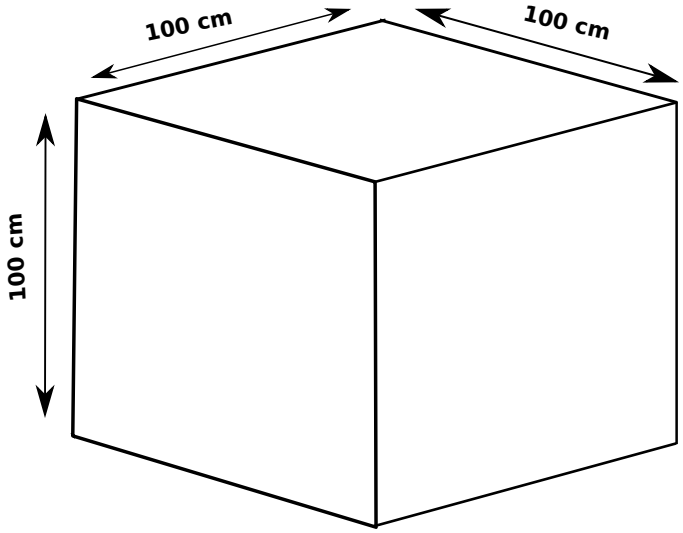


Figura 21 – Geometria do fantoma de *Box*.

O segundo fantoma, foi definido por uma imagem 3D de *voxels* em formato .mhd/raw com dimensão de 100 cm, resolução de 1 cm e total de 100x100x100 elementos como ilustrado na figura 22. Os parâmetros da macro estão contidos no Apêndice B.

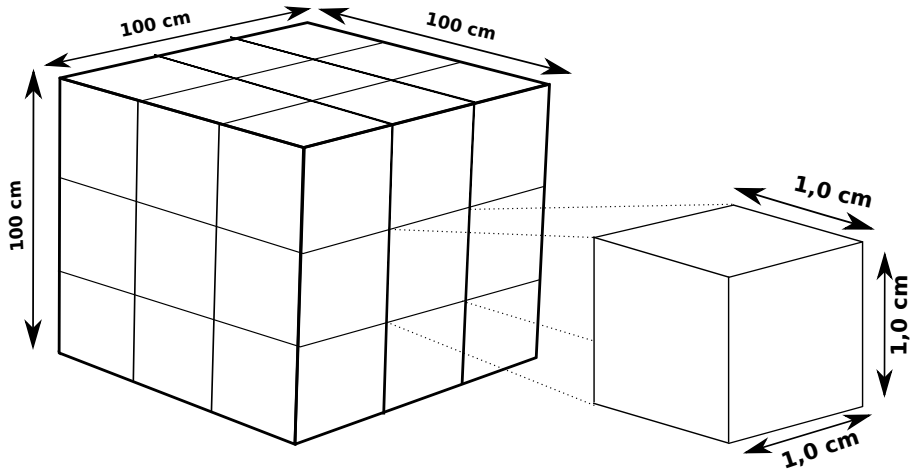


Figura 22 – Geometria do fantoma de *Voxel* de 100x100x100 elementos.

O terceiro fantoma, também definido por uma imagem 3D de *voxels* em formato .mhd/raw com dimensão de 100 cm, resolução de 0,5 cm e total de 200x200x200 elementos como ilustrado na figura 23. Os parâmetros da macro estão contidos no Apêndice B.

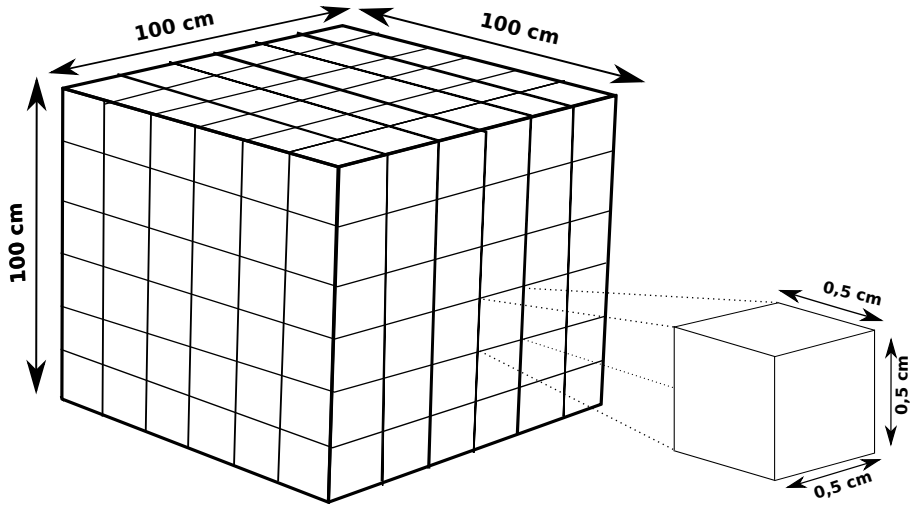
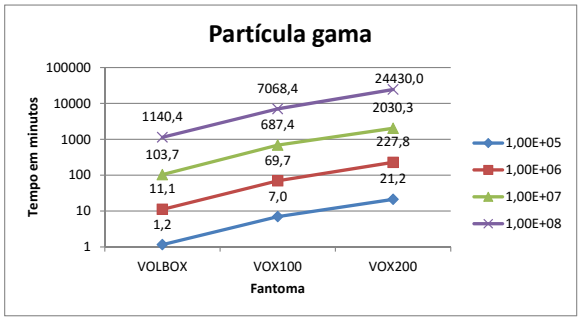


Figura 23 – Geometria do fantoma de *Voxel* de 200x200x200 elementos.

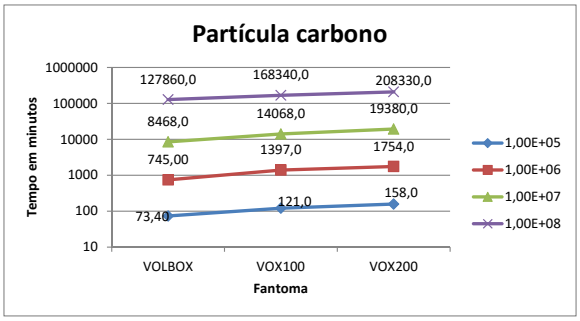
3.6.1 Resultados de discussão

3.6.1.1 Análise do tempo de simulação com base no modelo do fantoma *cluster* CTR

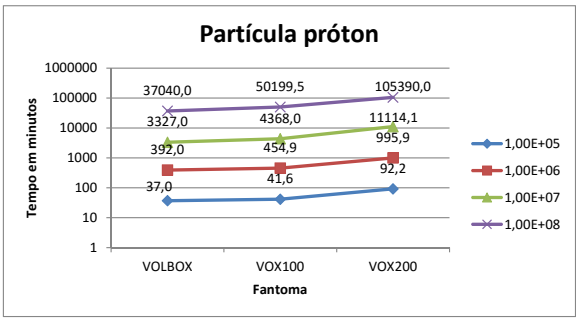
As figuras 24 (a), (b) e (c), representam os gráficos para as simulações realizadas com as partículas gama, carbono e próton respectivamente. Os gráficos representam o comportamento do tempo em relação à mudança no modelo do fantoma nas simulações. Para simulações com a partícula gama considerando o mesmo número de histórias, a mudança geometria o tempo de processamento aumentou, mesmo comportamento foi observado na para o carbono no gráfico (b) e próton mostradas no gráfico (c).



(a) Gráfico partícula gama



(b) Gráfico partícula carbono

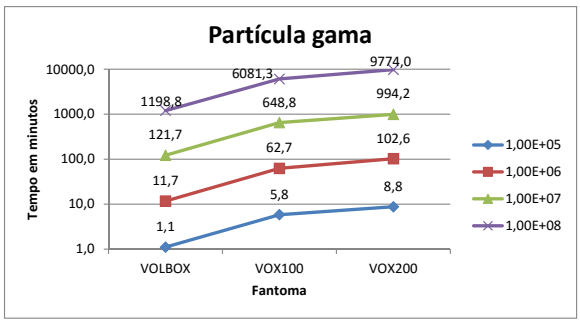


(c) Gráfico partícula próton

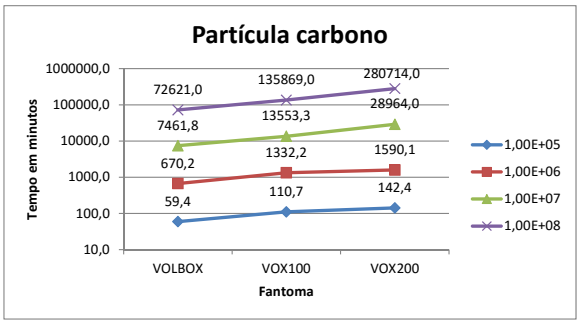
Figura 24 – Gráficos demonstrativos do tempo para as partículas gama, carbono e próton em razão da variação do fantoma no *cluster* CTR.

3.6.1.2 Análise do tempo de simulação com base no modelo do fantoma *cluster* CACAU

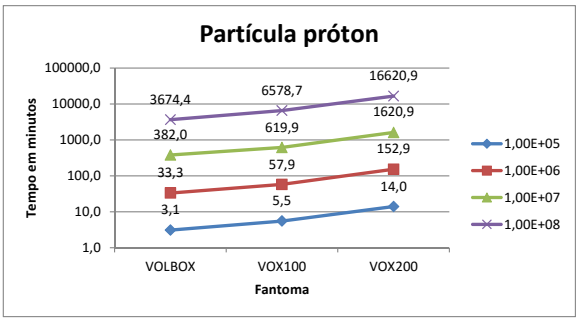
Os gráficos (a), (b) e (c) da figura 25 ilustram o comportamento de uma simulação com modelo de fantoma diferente e referem respectivamente às partículas gama, carbono e próton. Esses tempos foram obtidos em simulação realizada com um único processador, sendo expressos em minutos. Os resultados apresentados nos gráficos que nas simulações realizadas no CACAU, para a mesma partícula e mesmo números de histórias, a geometria do fantoma afetam o desempenho das simulações. Conforme os gráficos, as simulações que utilizaram gama foi processada em tempos menores, seguida por próton e carbono. As simulações do carbono com 10^8 histórias e fantomas de *voxel* 100 e 200 foram obtidos a partir linearização do tempo de processamento com base no tempo médio para realizar uma história.



(a) Gráfico partícula gama



(b) Gráfico partícula carbono



(c) Gráfico partícula próton

Figura 25 – Gráficos demonstrativos do tempo para as partículas gama, carbono e próton em razão da variação do fantoma no *cluster* CACAU.

3.7 Análise do tempo de simulação em função do número de histórias

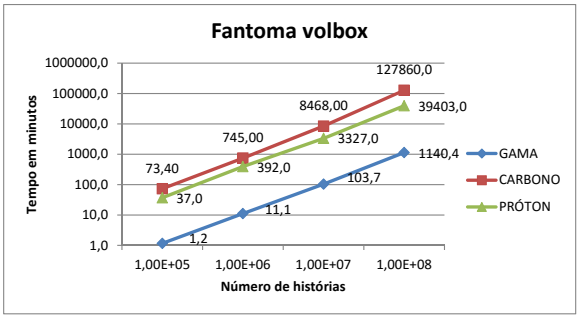
Deseja-se verificar a estabilidade na operação do *cluster* com o aumento do número de histórias. Para analisar o comportamento das simulações em função do número de histórias, cada simulação foi executada com $10^5, 10^6, 10^7$ e 10^8 histórias. A fonte fornece energia de acordo a partícula, sendo 200 MeV para o próton, 18 MeV para o gama e 2500 MeV para íon de carbono. Cada simulação foi processada com o tipo de partícula, número de histórias e fantoma, tanto para CTR quanto para o CACAU. Espera-se que nas estruturas as simulações mantenham estabilidade e comportamento linear na relação tempo x histórias.

3.7.1 Resultados e discussão

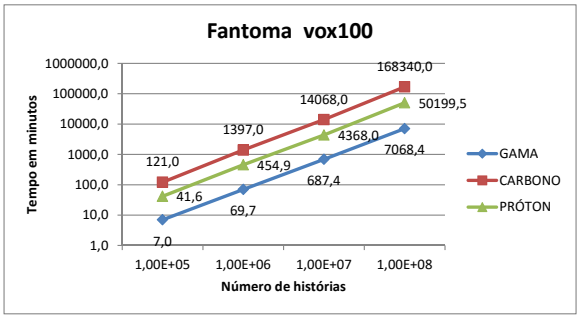
3.7.1.1 Análise do tempo de simulação em função do número de histórias no CTR

A figura 26 apresenta os gráficos contendo a variação de tempo para as partículas gama, próton e carbono com base no número de histórias. O tempo está em minutos e a escala logarítmica é de base 10. Nos respectivos gráficos, nota-se que para os três fantomas (*Box, voxel de 100 e 200* elementos) os resultados apontam um variação significativa de tempo em razão do aumento no número de histórias para cada uma das três partículas. No gráfico da figura 26 (c) apesar das linhas dos gráficos estarem próximas devido a escala logarítmica, a diferença é significativa. Sendo assim, o aumento do número de histórias aumento proporcionalmente o tempo de simulação independentemente do tipo do fantoma ou partícula.

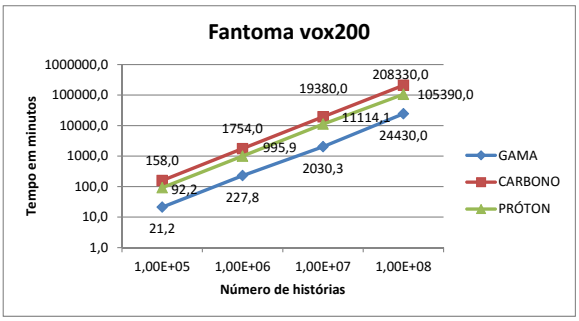
No CTR as simulações com fantoma de *voxel* 200 e carbono como partícula e 10^8 histórias são processadas no maior tempo, equanto as simulações com gama tem menor tempo de simulação. Foi o cenário esperado, com base na validação da instalação.



(a) Gráfico fantoma de box



(b) Gráfico fantoma de *voxel 100*



(c) Gráfico fantoma de *voxel 200*

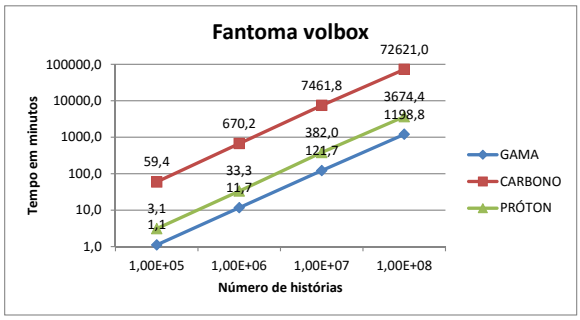
Figura 26 – Gráficos demonstrativos de tempo para fantomas *Box voxel* 100 e *voxel* 200 em função do aumento no número de histórias no CTR.

Observa-se nos gráficos que as partículas próton, gama e carbono mantiveram o comportamento de linearidade nas simulações com diferentes fantomas e números de histórias. Isso sugere a estabilidade do sistema.

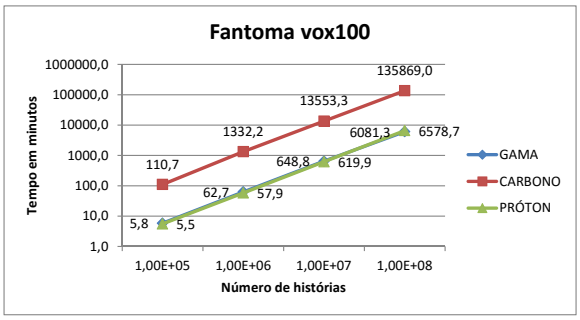
3.7.1.2 Análise do tempo de simulação em função do número de histórias no CACAU

A figura 27 através dos gráficos (a), (b) e (c) ilustra o comportamento das simulações em razão do aumento no número de histórias. Como nos resultados apresentados anteriormente, o tempo está expresso em minutos. Cada gráfico mostra os resultados para os fantomas *box*, *voxel* 100 e 200 elementos respectivamente.

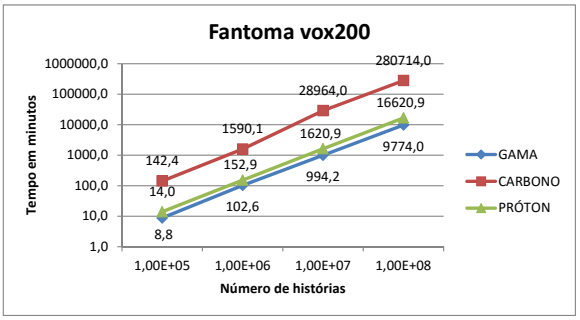
Nota-se pelos gráficos que o aumento no número de histórias tem relação direta com o tempo de processamento em todas as partículas para cada fantoma. Conforme as linhas dos gráficos (a), (b) e (c), os tempos obtidos com as simulações processadas com a carbono foram muito superiores aos das outras partículas, atingindo tempo superior a dez vezes para processar a mesma simulação. O desempenho apresentado no CACAU envolvendo gama e próton foram semelhantes, chegando a atingir valores próximos conforme o gráfico (b). A diferença apresentada entre carbono e as outras partículas pode estar relacionada ao fato de ter sido realizada em um período de demanda alta no CACAU (Dez/2015 a Jan/2016).



(a) Gráfico fantoma de box



(b) Gráfico fantoma de *voxel 100*



(c) Gráfico fantoma de *voxel 200*

Figura 27 – Gráficos demonstrativos de tempo em função do aumento do número de histórias no CACAU.

Observa-se também nos gráficos obtidos para o CACAU que as partículas próton, gama e carbono mantiveram o comportamento de linearidade nas simulações com diferentes fantomas e números de histórias. Isso sugere a estabilidade do sistema nas simulações.

3.8 Análise de desempenho da paralelização

Deseja-se analisar o tempo de simulação com aumento no número de CPUs no CTR e no CACAU. Cada simulação executada com uma CPU foi processada com quatro e oito CPUs no CTR. O mesmo procedimento foi repetido no CACAU com oito e dezesseis CPUs.

A escolha da quantidade de tarefas paralelas foi definido em função do número de núcleos processadores existentes em cada nó. No CTR cada nó possui um processador e quatro núcleos, enquanto o CACAU possui dois processadores com quatro núcleos. O

objetivo foi utilizar no máximo dois nós nas simulações paralelas e virtude demanda existente.

Deseja-se verificar o desempenho das simulações em função do aumento do número de CPUs. Para isso, a simulação deve ser divida na quantidade correspondente ao número de CPUs através do GJS (Gate Job Splitter) da plataforma OpenPBS. Para submissão das simulações, foi necessário a implementação dos *scripts* SLURM, gerados com código implementado no apêndice C para CTR com quatro e oito CPUs. Este mesmo procedimento foi repetido para o CACAU com oito e dezesseis CPUs.

As estatísticas da simulação são armazenadas em arquivo formato texto e devido a quantidade de arquivos gerados foi usado o código do Apêndice D para extrair e calcular o tempo médio de processamento de cada simulação.

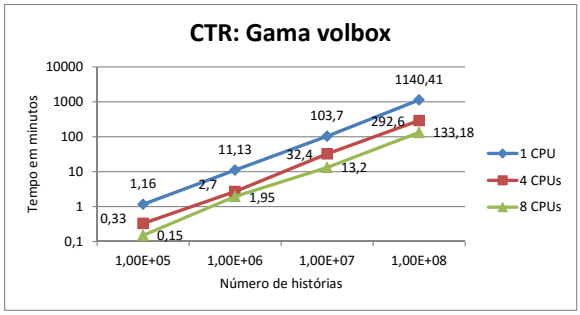
3.8.1 Resultados e discussão

Alguns parâmetros foram citados nas subseções anteriores que influenciam no tempo de processamento de uma simulação dosimétrica. Esta subseção mostra o desempenho da paralelização com o aumento no número de CPUs.

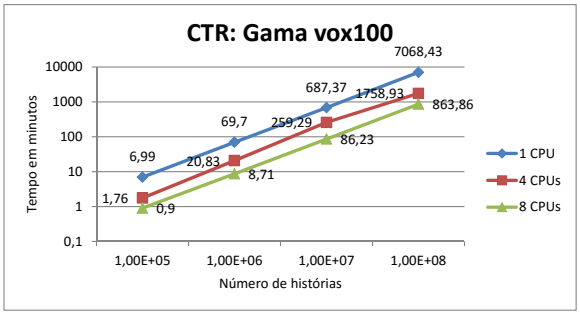
3.8.1.1 Análise de desempenho da paralelização no CTR

Os gráficos apresentados nas figuras 28, 29 e 30, apresentam o desempenho observado nas simulações realizadas no CTR. Os resultados obtidos conforme apresentados nos gráficos, mostram redução significativa de tempo com o aumento de processadores.

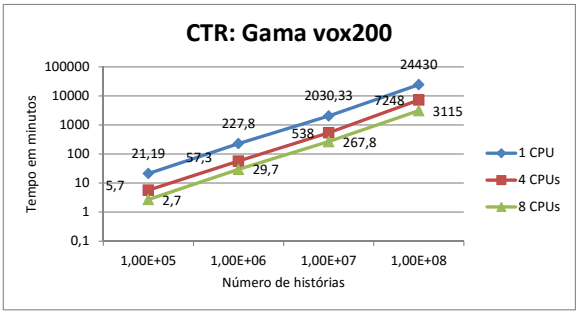
Nos gráficos (a), (b), e (c) da figura 28, para gama, os tempos de simulação foram reduzidos para todos os números de histórias e modelo de geometria utilizada. A figura 28 (c) o desempenho de processamento de oito CPUs em relação quatro, foi menor que quatro em relação a um CPU. Ainda sim, o tempo de simulação foi reduzido com o aumento do número de CPUs.



(a) Desempenho fantoma de box



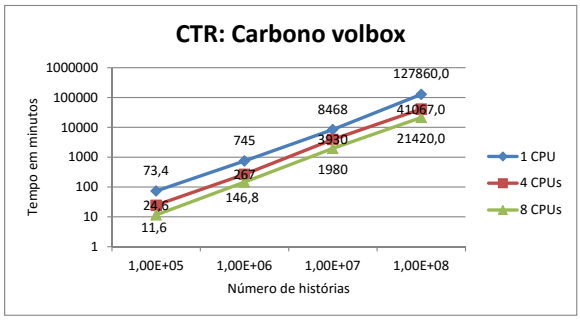
(b) Desempenho fantoma de *voxel 100*



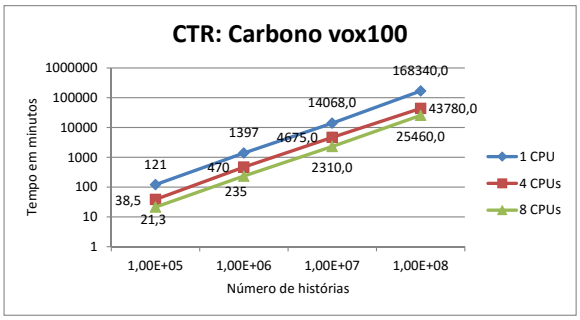
(c) Desempenho fantoma de *voxel 200*

Figura 28 – Gráficos demonstrativos de desempenho da partícula gama com fantoma de box, *voxel* de 100 e 200 elementos com o aumento do número de CPUs no CTR.

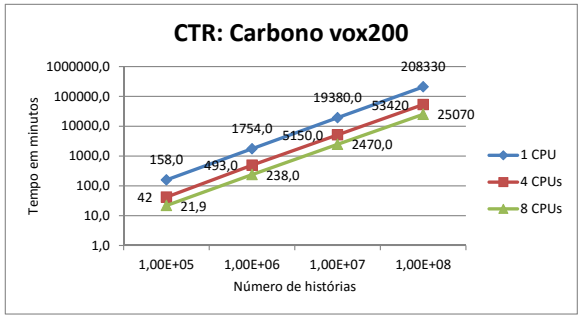
Simulações com carbono foram realizadas em tempo superior às outras partículas. Neste sentido, com mostra a figura 29 (a), apresentou desempenho significativo, tendo em vista que o tempo de simulação foi alto para $10^7, 10^8$ histórias. Os resultados obtidos demonstram que o tempo foi reduzido com o aumento do número de CPUs.



(a) Desempenho fantoma de box



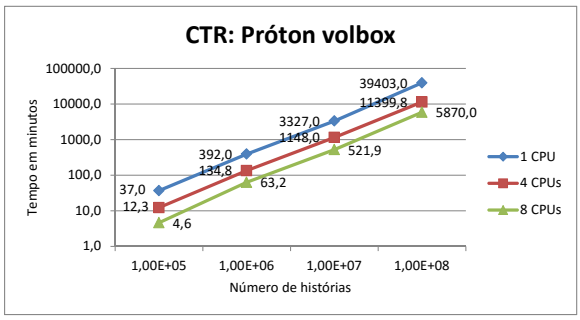
(b) Desempenho fantoma de *voxel 100*



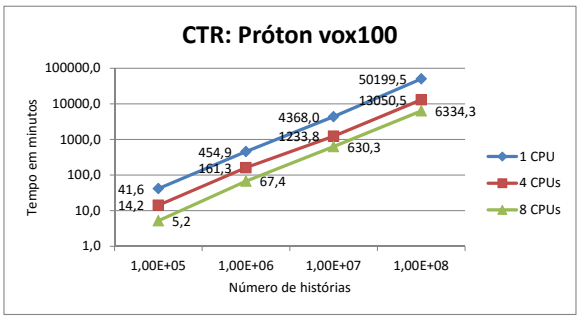
(c) Desempenho fantoma de *voxel 200*

Figura 29 – Gráficos demonstrativos de desempenho da partícula carbono com fantoma de box, *voxel* de 100 e 200 elementos em função do número de CPUs no CTR.

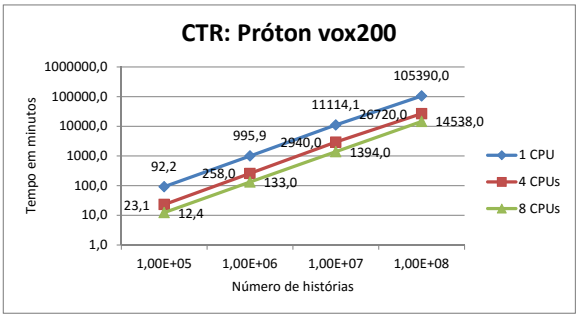
O desempenho apresentado para o próton com fantoma de *box* nas figuras 30 (a) e 30 (b), manteve-se regular. As simulações com próton, *voxel* de 200 elementos e número de histórias de 10^8 da figura 30 (c) apresentou um desempenho compatível se comparado ao outros resultados tanto para quatro quanto para oito CPUs.



(a) Desempenho fantoma de box



(b) Desempenho fantoma de *voxel 100*



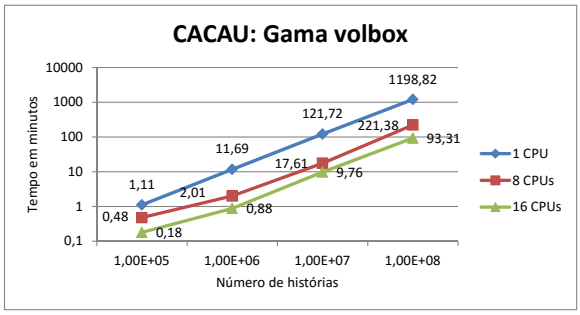
(c) Desempenho fantoma de *voxel 200*

Figura 30 – Gráficos demonstrativos de desempenho da partícula próton com fantoma de box, *voxel* de 100 e 200 elementos com o aumento do número de CPUs no CTR.

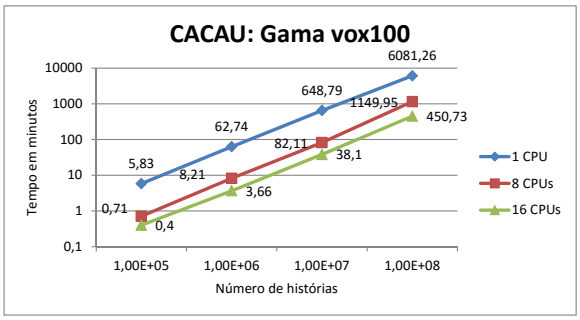
3.8.1.2 Análise de desempenho da paralelização no CACAU

Os gráficos das figuras 31, 32 e 33 mostram as desempenho obtido com a divisão das tarefas entre vários processadores (oito e dezesseis).

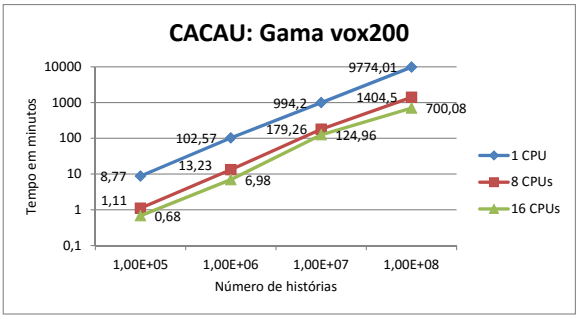
Nos gráficos (a), (b), e (c) da figura 31, para gama, o aumento no número de processadores reduziu significativamente os tempos de simulação, garantindo a redução para todos os números de histórias e modelo de geometria utilizado. Estas simulações tiveram o menor tempo de processamento. O desempenho apresentado entre oito e dezesseis CPUs não foi proporcional se comparados os resultados obtidos com oito em relação a um CPU. Neste caso, o desempenho tende a ser menos significativo devido ao pouco tempo de simulação.



(a) fantoma de box



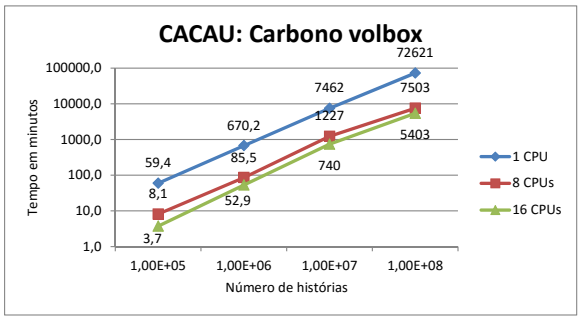
(b) fantoma de *voxel 100*



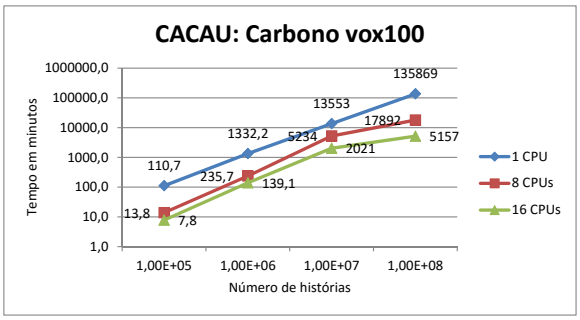
(c) fantoma de *voxel 200*

Figura 31 – Gráfico demonstrativo de desempenho da partícula gama com fantoma de box, *voxel* de 100 e 200 elementos com o aumento do número de CPUs no CACAU.

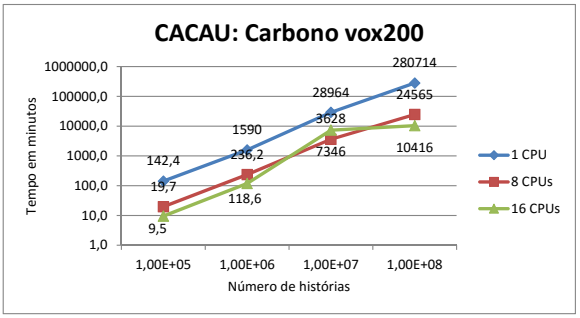
As simulações realizadas com carbono foram executadas em tempo superior comparado ao gama e próton. Na figura 32 (c) a simulação com 10^7 histórias com dezesseis CPUs foi superior ao realizado por oito CPUs. Essa situação atípica está relacionada ao grande número de solicitações do CACAU no período, como mostrado na seção 3.11.



(a) fantoma de box



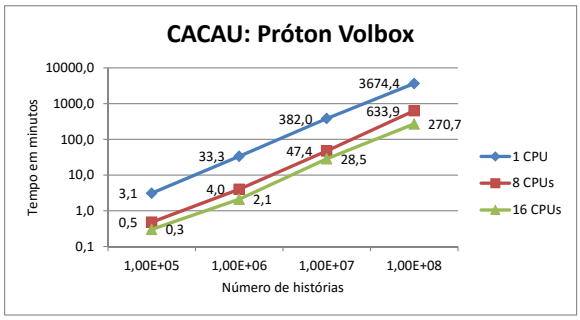
(b) fantoma de *voxel 100*



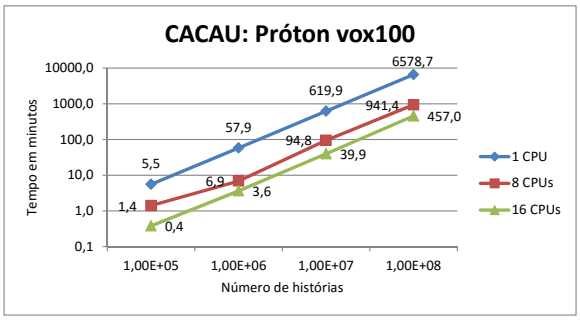
(c) fantoma de *voxel 200*

Figura 32 – Gráfico demonstrativo de desempenho da partícula carbono com fantoma de box, *voxel* de 100 e 200 elementos com o aumento do número de CPUs no CACAU.

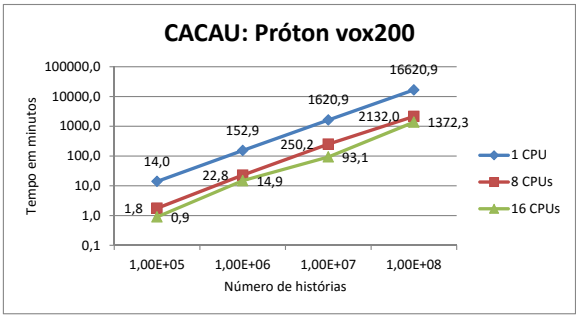
Os gráficos (a), (b) e (c) da figura 33 apresentam os resultados obtidos no CACAU para o próton. Para todos os valores de histórias e modelos de geometria o desempenho menteve-se estável com o aumento de processadores, exceto na figura 32 (c) com 10^7 histórias.



(a) fantoma de box



(b) fantoma de *voxel 100*



(c) fantoma de *voxel 200*

Figura 33 – Gráfico demonstrativo de desempenho da partícula próton com fantoma de box, *voxel* de 100 e 200 elementos com o aumento do número de CPUs no CACAU.

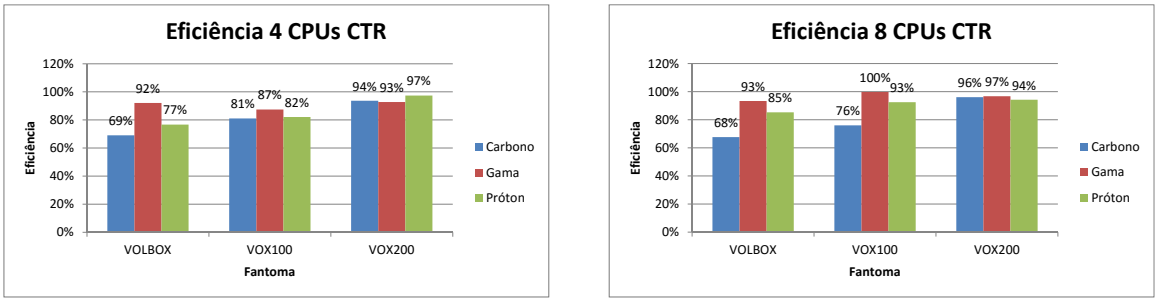
3.9 Análise da eficiência para avaliar o desempenho dos *clusters*

A partir dos tempos obtidos, deseja-se analisar o quanto as simulações paralelas foram mais eficientes em comparação com as realizadas com um único CPU, bem como obter o percentual de eficiência de maneira específica para cada plataforma. Para isso foi necessário calcular o *speedup* e eficiência das simulações paralelas no CTR e no CACAU, permitindo comparar o desempenho entre as duas estruturas computacionais. O cáculo é feito com base na média da eficiência obtida para cada valor de histórias, considerando o mesmo fantoma e mesma partícula. Essa comparação permite sinalizar o desempenho e a estabilidade através da eficiência média obtida em função do número de CPUs. Obtidos os tempos médios gastos, os dados foram tabulados e armazenados em uma planilha eletrônica para serem analisados quanto ao *speedup*, a eficiência.

3.9.1 Resultados e discussão

A eficiência é umas das principais métricas utilizadas na computação paralela para avaliar o desempenho de uma infraestrutura computacional. Nesta seção os gráficos (a) e (b) das Figuras 34 e 35 trazem o valor médio da eficiência no processamento das tarefas paralelas calculado com base na eficiência obtida para cada número de histórias. Sendo assim, permite-se avaliar o desempenho médio para uma partícula utilizando determinado fantoma variando a quantidade de histórias processadas na simulação.

No gráficos 34 (a) e 34 (b), a eficiência registrada do carbono (quatro e oito nós) apresentou a maior variação. Os resultados para quatro nós, em geral a média foi abaixo de 100% melhorando a eficiência quando o número de nós processadores foi aumentado. Os gráficos mostram que nas simulações realizadas com gama o desempenho apresentou maior estabilidade, assim com as simulações realizadas com fantoma de *voxel* de 200 elementos. Os resultados mostram que o *cluster* CTR apresenta estabilidade nas simulações.

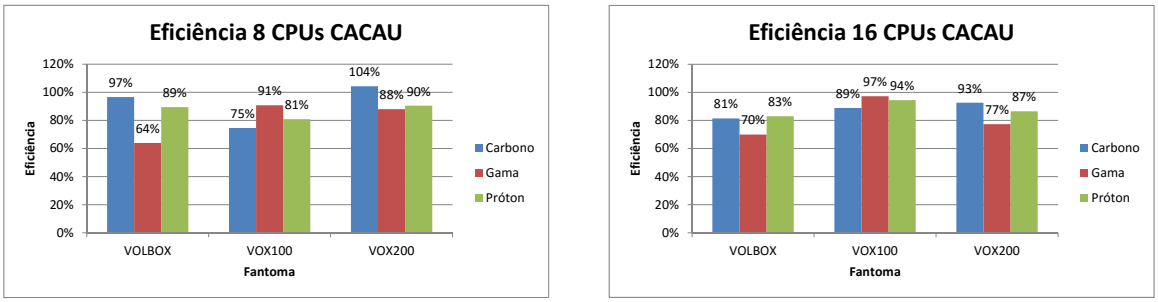


(a) Eficiência 4 CPUs

(b) Eficiência 8 CPUs

Figura 34 – Média da eficiência calculada pelo número de histórias das simulações processadas no *cluster* CTR.

Os gráficos mostram que a eficiência média no CACAU oscilou pouco apesar de quase todos os resultados estarem abaixo de 100%. Os 64% de eficiência média do *Volbox* com gama para oito nós, foi o mais baixo registrado para o CACAU.



(a) Eficiência 8 CPUs

(b) Eficiência 16 CPUs

Figura 35 – Média da eficiência calculada pelo número de histórias das simulações processadas no *cluster* CACAU.

Nessas circunstâncias, os resultados obtidos com oito nós no CTR apresentaram-se mais eficiente que quatro, o mesmo aconteceu no CACAU para dezesseis em relação a oito nós. Ao comparar os gráficos das Figuras 34 e 35, em geral as simulações realizadas no CTR sofreu maior variação quando a simulação envolveu o carbono. Nas simulações envolvendo gama os tempos de processamento tem variações menores no CTR.

3.10 Análise de desempenho do tempo de simulação por história

Deseja-se determinar o tempo médio para processar uma história por CPU com relação a um fantoma. Inicialmente foi estimado o tempo para um, quatro, oito e dezesseis por CPUs através da fórmula:

$$Tph = \frac{T_{sim}}{N_{hist}} \times N_{cpus} \ , \tag{7}$$

onde, Tph é o tempo por CPU, T_{sim} é o tempo para realizar a simulação, N_{hist} o número de histórias da simulação e N_{cpus} o número de CPUs envolvidos no processamento. Considerando o mesmo fantoma e partícula, espera-se que o número de histórias processadas por segundo seja aproximadamente constante. O cálculo do tempo estimado é dado pela média dos tempos. Calculando-se a média dos Tph obtidos é possível estimar o tempo médio gasto para realizar uma simulação completa tendo em vista o número de histórias diferentes para mesmo fantoma e partícula.

3.10.1 Resutados e discussão

Com base nessas informações foi possível estimar o tempo aproximado para realizar uma simulação com as mesmas características das que foram feitas neste trabalho. Assim o tempo final estimado para finalizar uma simulação pode ser calculado com base nos dados das tabelas 6 e 7 e número de histórias.

3.10.1.1 Estimativa de tempo para processamento por história no CTR

Nesta seção são apresentadas as estimativas de tempo para realização de uma história no *cluster* CTR. Após as análises das tabelas de 10 a 18, os resultados obtidos foram racionalizados com base na fórmula 7 e obtendo os tempos médios constantes na tabela 6 como tempo de referência para realização de uma simulação.

Tabela 6 – Tempo estimado para simular uma história no *cluster* CTR

Fantoma	Tempo carbono (min)	Tempo gama (min)	Tempo próton (min)
VolBox	1,2 x 10^{-3}	1,2 x 10^{-5}	4,5 x 10^{-4}
Vox100	1,6 x 10^{-3}	7,0 x 10^{-5}	5,0 x 10^{-4}
Vox200	1,9 x 10^{-3}	2,2 x 10^{-4}	1,1 x 10^{-3}

Conforme os resultados apresentados, o tempo para realização de uma simulação pode ser estimado com base no tipo de partícula, modelo do fantoma e número de histórias.

3.10.1.2 Estimativa de tempo para processamento por história no CACAU

Os resultados obtidos com as simulações no CACAU passaram pelo mesma análise com base nos resultados existentes nas tabelas 19 a 27. Apesar do tempo de processamento se apresentar superior aos constantes na tabela 6, a estrutura CACAU se mostrou mais estável em situações normais.

Tabela 7 – Tempo estimado para simular uma história no *cluster* CACAU

Fantoma	Tempo carbono (min)	Tempo gama (min)	Tempo próton (min)
VolBox	1,0 x 10^{-3}	1,5 x 10^{-5}	4,5 x 10^{-5}
Vox100	1,5 x 10^{-3}	6,5 x 10^{-5}	7,5 x 10^{-5}
Vox200	2,0 x 10^{-3}	1,1 x 10^{-4}	1,9 x 10^{-4}

Nota-se pela análise realizada que o tempo gasto por uma simulação no *cluster* CACAU em geral é menor que as simulações realizadas no CTR e isso se deve em parte pela infraestrutura computacional ser superior. Entretato, em se tratando da partícula

carbono, está relacionado ao fato do CACAU possuir demandas de outros usuários e tarefas que consomem muitos recursos computacionais como escrita, armazenamento, uso de memória e processamento no período em que foram simulados.

Os resultados comprovam que aumentando a complexidade do fantoma, o tempo de simulação também aumenta, comprovado tanto nos valores da tabela 6 como os da tabela 7. Quanto às partículas, nas simulações envolvendo a partícula carbono, os tempos obtidos no CACAU foram superiores aos obtidos no CTR. No período em que foram realizadas (dezembro de 2015 e janeiro de 2016), a demanda no CACAU era alta, devido às solicitações de vários usuários.

3.11 Análise de desempenho da simulação em função do tempo de escrita

Vários processos tentando escrever dados em disco, gera concorrência e o tempo total de execução pode sofre alterações. Essas alterações pode ocorrer devido a frequência em que dados são gravados em disco e quantidade de processos sendo executados simultaneamente.

Deseja-se analisar o tempo de processamento em função do intervalo de gravação em disco e número de tarefas no mesmo nó.

Para realizar essa análise, todas as simulações forma executadas em um único nó do *cluster* CACAU. Assegurou-se que nenhuma tarefa de outros usuários foi processada durante o período. A simulação serial foi realizada com a partícula carbono, fantoma de *Voxel* 200x200x200 e 10^8 histórias. Cada simulação realizada nesta etapa foi executada com intervalo de gravação em disco de 1, 10 e 100 minutos. Após o processamento serial, realizou-se o processamento paralelo com uma, duas, quatro e oito simulações simultâneas no mesmo nó e tempo de gravação de 1 e 100 minutos.

3.11.1 Resultados e discussão

A Tabela 8 mostra os tempos obtidos para simulações sem paralelismo. Aumentar o tempo de gravação em disco representa um ganho significativo no tempo final da simulação. O tempo de processamento reduziu em 17,6% em quando gravado a cada 10 minutos e 18,4% quando gravado a cada 100 minutos. A diferença entre o tempo de gravação a cada 10 e 100 minutos é inferior a 1%, neste sentido, conclui-se que, o tempo de gravação a cada 10 minutos para as características do *hardware* utilizado e tipo de simulação descrita acima é mais indicado. Neste sentido, tempo de gravação representa um percentual considerável no tempo de simulação.

Tabela 8 – Tempo registrados para cada simulação serial em função do tempo de gravação

Intervalo de gravação	Tempo de simulação	Percentual de redução
1 minuto	1587	—-
10 minutos	1308	17,6%
100 minutos	1295	18,4%

Na análise para estudar o desempenho quanto ao número de simulações executadas no mesmo nó, a Tabela 9 mostra o desempenho das simulações considerando o tempo de gravação e o número de simulações executadas simultaneamente no nó do *cluster*. Para um nó contendo oito núcleos á partir de quatro simulações o tempo começa aumentar, considerando a gravação a cada minuto. Gravando a cada 100 minutos o tempo foi praticamente o igual tanto para uma quanto oito simulações. Comparando o tempo de simulação gravando a cada 100 minutos em face à gravação a cada 1 minuto, ao utilizar oito simulações no mesmo nó, o tempo reduziu em aproximadamente 25%.

Tabela 9 – Tempo registrados para cada simulação em função do tempo de gravação e número de CPUs utilizados

Números de simulações por nó	Intervalo de gravação (1 min)	Intervalo de gravação (100 min)	Percentual de redução
1	182	160	12,1%
2	185	160	13,5%
4	194	161	17,0%
8	213	161	24,4%

Conclui-se que a quantidade de tarefas simultâneas no mesmo nó, diminui o desempenho das simulações caso o tempo de escrita seja realizado em intervalos curtos de tempo.

4 Considerações finais

4.1 Conclusões

O gerenciador de recursos computacionais Slurm, está presente nos maiores computadores de alto desempenho do mundo. A UESC através do núcleo de tecnologia da informação, faz o uso desse poderoso gerenciador. Dois gerenciadores de recursos computacionais não deve ser utilizado concomitantemente em infraestrutura tipo *cluster*, pois, geram concorrência pelos recursos disponíveis, inviabilizando a sua eficiência. O GATE/Geant4 não traz suporte para o Slurm, neste sentido, através da metodologia descrita foi possível criação de códigos para viabilizar a execução do GATE em computadores da UESC usando o *Job Splitter*. Através desse trabalho os *clusters* da UESC foram confiugurados para realizar simulações com Gate e gerenciados pelo SLURM.

As simulações foram analisadas em função do erro e do raio do fantoma, para três partículas com fantoma de *box* em um computador *desktop* para definir o número histórias necessários para realizar uma simulação. Esperava-se que os mesmos resultados no CTR e CACAU o que foi confirmado expectativas iniciais após as simulações.

Após vários testes de desempenho e análises foi constatado que a complexidade do modelo geométrico afeta no tempo final de simulação. Dentre geometrias apresentadas na seção 3.6, os gráficos mostram que a ordem de aumento de tempo de simulação segue esta ordem: *box*, *voxel* de 100 e *voxel* 200 elementos.

Analisando pelo tipo de partícula nas dependências CTR, a ordem em que afetam o tempo final da simulação é: gama, próton e o carbono, observado igualmente no CACAU.

Em todos os testes, alteração do tipo de fantoma, tipo de partícula, e número de histórias, salvo casos pontuais, o aumento no número de processadores reduziu o tempo de simulação nos dois equipamentos usados para simular.

A análise da eficiência registra o percentual de quanto uma tarefa processada por meios paralelos foi melhor que uma realizada sem paralelismo. Nos resultados obtidos, as simulações realizadas o CTR apresentou desempenho semelhante ao CACAU e quantificação da eficiência.

Estimar o tempo médio para processar uma história permite calcular o tempo aproximado necessário para concluir uma simulação. Ao final foi calculado o tempo médio para realizar cada tipo de simulação para cada tipo de fantoma, partícula e número de histórias.

Aumentar o intervalo de gravação das simulações representa ganhos significativos. Aumentar de um para dez minutos a redução de tempo foi superior a 17%. A quantidade de tarefas paralelas executadas no mesmo nó afeta diretamente o tempo de simulação. Nas simulações realizadas com oito CPUs e intervalo de gravação de cem minutos comparado ao intervalo de um minuto, o desempenho foi superior a 24%.

Ao final deste trabalho os objetivos foram atingidos conforme propostos. O tempo de processamento gasto para realizar uma simulação está diretamente relacionado com o tipo da partícula, o fantoma e o número de histórias envolvidos no processamento. Sem instalar novos gerenciadores de recursos, os *scripts* feitos permitiram a instalação, execução e validação do GATE na UESC permitindo ao grupo de estudo de Física Médica realizar os pesquisas com transporte de partículas com um software público, poderoso e eficiente. O planejamento das simulações com base nos estudos realizados neste trabalho pode tornar a obtenção dos resultados em tempo reduzido.

4.2 Limitações do trabalho

Durante a realização deste trabalho, foram encontradas algumas limitações que ao final enriqueceu-o. Em um primeiro momento, ao instalar da versão Geant 4.9 no *cluster* CTR, percebeu-se nos teste iniciais que nas simulações envolvendo a partícula carbono o tempo de simulação era muito baixo e os resultados inconsistentes. Sendo assim, optou-se pela instalação da versão 4.10 do Geant4.

Ao realizar as simulações com próton com fantoma de *voxel* 200 no *cluster* CTR, o tempo de simulação foi alto. ao verificar as definições da macro de simulação, identificou-se que a visualização da simulação estava ativada. Novas simulações foram realizadas com a opção desativada.

Conforme mostrado na seção 3.11 a quantidade de simulações sendo executadas em um mesmo nó, aumenta o tempo de simulação se o intervalo de gravação em disco for reduzido. Nas simulações realizadas no CACAU envolvendo as partículas carbono, com dezesseis CPUs o desempenho foi baixo se comparado com oito CPUs, pois, no período em que as tarefas foram submetidas, a demanda por recursos foi alta. Em alguns momentos a espera foi de mais de 24 horas na fila de execução.

4.3 Trabalhos Futuros

Como sugestão de trabalhos futuros:

- A lei de Amdahl's (BARNEY, 2016) sugere a existência de limite de escalabilidade de paralelismo, onde o acréscimo de processadores a partir de um valor não reduz o tempo de processamento. O *cluster* do CTR possui 32 cores e o CACAU possui 160 cores. Como trabalho futuro, realizar simulações com várias comparações de desempenho com o máximo de processadores possíveis a fim de registrar o número máximo que produz resultados.
- Desenvolver uma interface WEB que permita usuários externos submeter simulações dosimétricas na UESC. A interface deve permitir o envio de relatórios com os resultados diretamente ao usuário após a finalizar o processamento da simulações.
- Realizar estudo comparativo de testes de desempenho com simulações dosimétricas (dose, energia depositada, incerteza da dose) utilizando técnicas de processamento paralelo com CPU e GPU.
- Realizar estudo comparativo de desempenho de simulações com técnicas de processamento paralelo *Job Spliter*, MPI e *Multithread* associado em conjunto de arquitetura compostas por CPU, GPU e mista.

Referências

AGENCY, I. A. E. **Diagnostics Radiology Physics : A Handbook for Teachers and Students**. Vienna: IAEA - Internacional Atomic Energy Agency, 2014.

AGENCY, I. A. E. **Nuclear Medicine Physics : A Handbook for Teachers and Students**. Vienna: IAEA - Internacional Atomic Energy Agency, 2014.

BARNEY, B. **Introduction to Parallel Computing**. 2016. Acesso em 22/03/2016. Disponível em: <https://computing.llnl.gov/tutorials/parallel_comp/#Flynn>.

BIELAJEW, A. F. **Fundamentals of the Monte Carlo method for neutral and charged particle transport**. Michigan: [s.n.], 2001. 1–338 p. Disponível em: <http://www-personal.umich.edu/~bielajew/MCBook/book.pdf>.

BONIFÁCIO, D. A. B. **Modelagem de sistema de detecção para momografia por emissão de pósitrons utilizando detectores cintiladores monolíticos - Instituto de Física**. 176 p. Dissertação Mestrado — Universidade de São Paulo, 2011.

BRASIL, S. **Guia de Estruturação e Administração do Ambiente de Cluster e Grid**. 1. ed. Brasília: [s.n.], 2006. 456 p. Disponível em: <http://guialivre.governoeletronico.gov.br/guiaonline/guiacluster/>.

BULL CEDOC, B. **Slurm Users Guide: extreme computing**. 2.2. ed. France 357, 2010.

CARUSO, F.; CARVALHO, B.; SANTORO, A. **A FÍSICA DE ALTAS ENERGIAS E A TERAPIA DE CÂNCER E PERSPECTIVAS.** 2005. Acesso em 18/06/2016. Disponível em: <http://slurm.schedmd.com/quickstart.html>.

CASSOLA, V. F. **Acoplamento dos Fantomas Tomográficos FAX06 e MAX06 ao Código Monte Carlo GEANT4**. 93 p. Dissertação de Mestrado — Universidade Federal de Pernambuco, 2007.

CERN et al. **Introduction to Geant4**. Genebra: CERN, 2014. Disponível em: <http://geant4.web.cern.ch/geant4/UserDocumentation/UsersGuides/IntroductionToGeant4/fo/BookIntroToGeant4.pdf>.

COLLABORATION, G. Geant4 User's Guide for Toolkit Developers. n. December, 2013. Disponível em: <http://geant4.web.cern.ch/geant4/UserDocumentation/UsersGuides/ForToolkitDeveloper/fo/BookForToolDev.pdf>.

COLLABORATION, G. **Geant4 User ' s Guide for Toolkit Developers**. [s.n.], 2014. Disponível em: <http://www.opengatecollaboration.org/sites/opengatecollaboration.org/files/GATE-UsersGuideV7.1.pdf>.

COLLABORATION, G. et al. **Geant4 User's Guide for Toolkit Developers**. Genebra: CERN, 2014. Disponível em: <http://geant4.web.cern.ch/geant4/UserDocumentation/UsersGuides/ForToolkitDeveloper/fo/BookForToolDev.pdf>.

COLLABORATION, O.; JAN, S.; SANTIN, G.; STRUL, D.; STAELENS, S.; ASSIé, K.; AUTRET, D.; AVNER, S.; BARBIER, R.; BARDIèS, M.; BLOOMFIELD, P. M.; BRASSE, D.; BRETON, V.; BRUYNDONCKX, P.; BUVAT, I.; CHATZIIOANNOU, A. F.; CHOI, Y.; CHUNG, Y. H.; COMTAT, C.; DONNARIEIX, D.; FERRER, L.; GLICK, S. J.; GROISELLE, C. J.; GUEZ, D.; HONORE, P.-F.; KERHOAS-CAVATA, S.; KIROV, A. S.; KOHLI, V.; KOOLE, M.; KRIEGUER, M.; LAAN, D. J. van der; LAMARE, F.; LARGERON, G.; LARTIZIEN, C.; LAZARO, D.; MAAS, M. C.; MAIGNE, L.; MAYET, F.; MELOT, F.; MERHEB, C.; PENNACCHIO, E.; PEREZ, J.; PIETRZYK, U.; RANNOU, F. R.; REY, M.; SCHAART, D. R.; SCHMIDTLEIN, C. R.; SIMON, L.; SONG, T. Y.; VIEIRA, J.-M.; VISVIKIS, D.; WALLE, R. V. de; WIEERS, E.; MOREL, C. **Users Guide V7.2:Introduction - GATE collaborative documentation wiki**. 2016. Accesso em 09 Março de 2016. Disponível em: <http://wiki.opengatecollaboration.org/index.php/Users_Guide_V7.2:Introduction>.

COOPERMAN, G.; NGUYEN, V. H. **Parallel Geant4 (ParGeant4)**. 2014. Acesso: março de 2016. Disponível em: <http://geant4.web.cern.ch/geant4/G4UsersDocuments/UsersGuides/ForApplicationDeveloper/html/Examples/parallel.html>.

CORDEIRO, T. d. P. V. **Coeficientes de Conversão para Dose Efetiva e Equivalente de Dose Ambiente para Feixes de Raios X Utilizados em Radioterapia**. 91 p. Tese (Doutorado) — Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2013.

COULORIS, G.; DOLLIMORE, J.; KINDBERG, T. **Sistemas Distribuídos**: conceitos e projetos. 4ª. ed. Porto Alegre: Bookman, 2007. 792 p.

DANTAS, M. **Computação Distribuída de Alto Desempenho**: Redes, clusters e grids computacionais. Florianópolis: Axcel Books, 2005.

DE FARIA, I. **Uma Abordagem de Seleção de Recursos Consciente de Consumo de energia baseada em Topologia de Rede, Tamanho de Arquivos e Potência de Equipamentos**. 102 p. Dissertação (Mestrado) — Universidade Federal de Santa Catarina, Florianópolis, 2015.

ELLER JUNIOR, E. **Estudo de Tecnologias para Computação Paralela e Distribuída**: Implementação de um cluster beowulf. Dissertação (Mestrado) — Universidade Federal Fluminense, Volta Redonda, 2013.

FLYNN, M. J. Some computer organizations and their effectiveness. **IEEE Transactions on Computers**, C-21, n. 9, p. 948–960, Sept 1972. ISSN 0018-9340.

FOSTER, I.; KESSELMAN, C.; TUECKE, S. The anatomy of the grid: Enabling scalables virtual organizations. **International journal of high performance computing applications**, Sage Publications, v. 15, p. 200–222, 2001.

GUIMARÃES, N. A. **Avaliação Dosimétrica para o Tratamento de Câncer de Mama Masculina**. 110 p. — Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2015.

JAN, S.; SANTIN, G.; STRUL, D.; STAELENS, S.; ASSIé, K.; AUTRET, D.; AVNER, S.; BARBIER, R.; BARDIèS, M.; BLOOMFIELD, P. M.; BRASSE, D.; BRETON, V.; BRUYNDONCKX, P.; BUVAT, I.; CHATZIIOANNOU, A. F.; CHOI, Y.; CHUNG, Y. H.; COMTAT, C.; DONNARIEIX, D.; FERRER, L.; GLICK, S. J.; GROISELLE, C. J.; GUEZ, D.; HONORE, P.-F.; KERHOAS-CAVATA, S.; KIROV, A. S.; KOHLI, V.; KOOLE, M.; KRIEGUER,

M.; LAAN, D. J. van der; LAMARE, F.; LARGERON, G.; LARTIZIEN, C.; LAZARO, D.; MAAS, M. C.; MAIGNE, L.; MAYET, F.; MELOT, F.; MERHEB, C.; PENNACCHIO, E.; PEREZ, J.; PIETRZYK, U.; RANNOU, F. R.; REY, M.; SCHAART, D. R.; SCHMIDTLEIN, C. R.; SIMON, L.; SONG, T. Y.; VIEIRA, J.-M.; VISVIKIS, D.; WALLE, R. V. de; WIEERS, E.; MOREL, C. **Users Guide V7 . 1 : Introduction From GATE collaborative documentation wiki**. 2014. Disponível em: <http://www.opengatecollaboration.org/sites/opengatecollaboration.org/files/GATE-UsersGuideV7.1.pdf>.

JONES, M. T. **Otimizando o Gerenciamento de Recurso em Supercomputadores com o SLURM**. 2014. Acesso em 27/03/2016. Disponível em: <http://www.ibm.com/developerworks/br/library/l-slurm-utility/>.

KHAN, F. M.; GIBBONS, J. P. **Khan's the Physics of Radiation Therapy**. 5ª. ed. Philadelphia: Lippincott Williams & Wilkins, 2014.

LJUNBERG, M.; STRAND, S.; KING, M. A. **Monte Carlo Calculations in Nuclear Medicine**: Applications in diagnostic imaging. 2ª. ed. New York: CRC Press, 2013. ISBN 9781439841099.

MALTHEZ, A. L. M. C. **Aplicabilidade e Validação do Geant4 para Fótons e Elétrons em Radioterapia**. 81 p. Tese (Dissertação) — Universidade Estadual de Campinas, Campinas, SP, 2011.

MARTINS, M. C. **Simulações por Monte Carlo de Tratamentos de Braquiterapia Utilizando Simuladores Antropomórficos em Voxels**. 125 p. Tese (Doutorado) — Universidade Federal do Rio de Janeiro, 2014.

Ministério da Saúde (BR). **Manual de Bases Técnicas da Oncologia–SIA/SUS-Sistema de Informações Ambulatoriais. Brasília: MS**. 21ª. ed. Brasília, 2015.

OBAL, T. M. **Uma Abordagem Multi Objetivo ao Problema da Intensidade de Dose em Planejamentos do Tratamento de Câncer por Radioterapia**. 93 p. Tese (Doutorado) — Universidade Federal do Paraná, 2011.

PINTO, R. J. **Aplicação de Processamento Paralelo ao Problema de Planemamento da Oparação de Sistemas Hidrotérmicos Baseado em *Cluster de Computadores***. Tese (Doutorado) — Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2011.

PODGORSAK, E. B. et al. Radiation oncology physics:a handbook for teachers and students. **a handbook for teachers and students/EB Podgorsak.–Vienna: International Atomic Energy Agency**, IAEA - Internacional Atomic Energy Agency, Vienna, v. 657, 2005.

RODER, A. F. V. **Estudo da interação de prtótons com alvos não homogêneos, aplicado a tomografia com feixes de prótons**. 79 p. Dissertação (Mestrado) — Universidade Estadual Paulista, Instituto de Biociências de Botucatu, Botucatu, 2014.

RODRIGUES, D. **SLURM-BULL-veredas Documentation**. UFMG: CENEPAD, 2011. Acesso em 28/08/2015. Disponível em: <http://www.cenapad.ufmg.br/index.php/documentosemanuais?download=4:slurm-bull-veredas>.

SALVAJOLI, J. V.; SALVAJOLI, B. P. O papel da radioterapia no tratamento do câncer–avanços e desafios. **Rev Onco [Internet]**, v. 13, p. 32–6, 2012.

SCHEDMD, S. L. U. f. R. **Rosetta Stone of Workload Managers**. 2013. Acesso em 14/08/2015. Disponível em: <http://slurm.schedmd.com/rosetta.html>.

SHEDMD. **Quick Start User Guide**. 2016. Acesso em 28/03/2016. Disponível em: <http://slurm.schedmd.com/quickstart.html>.

STALLINGS, W. **Arquitetura e Organização de Computadores**. 8ª. ed. São Paulo: Pearson Prentice Hall, 2010.

TANENBAUM, A. S. **Organização e Arquitetura de Computadores**. 5ª. ed. São Paulo: Pearson Prentice Hall, 2007.

TANENBAUM, A. S.; STEEN, M. V. **Sistemas Distribuídos**: princípios e paradigmas. 2ª. ed. São Paulo: Pearson Prentice Hall, 2007.

TAUAHATA, L.; SALATI, I.; PRINZIO, R. D.; PRINZIO, A. R. D. et al. **Radioproteção E Dosimetria : Fundamentos**. 10ª. ed. RIO DE JANEIRO: IRD/CNEN, 2014. 345 p. ISBN 9788567870021.

TEODORO, S. **Algoritmos de escalolnamento para grades comptuacionais voltados à eficiÊncias energética**. 117 p. Dissertação (Mestrado) — Puntifícia Universidade Católica do Rio Grande do Sul, Porto Alegre, 2013.

VIEIRA, I. F. **Desenvolvimento de um software para modelagem de tomógrafos por emissão de pósitrons**. 104 p. Dissertação Mestrado — Universidade Federal de Pernambuco, 2013.

WRIGTH, D.; COLLABORATION, G. Physics Reference Manual. **Geant4 - Physics Reference Manual**, v. 0, n. December, p. 1–558, 2011. Disponível em: <http://geant4.web.cern.ch/geant4/UserDocumentation/UsersGuides/PhysicsReferenceManual/fo/PhysicsReferenceManual.pdf>.

YORIYAZ, H. Método de Monte Carlo : princípios e aplicações em Física Médica Monte Carlo Method : principles and applications in Medical Physics. **Rev. Bras. Física Médica**, v. 3, n. 1, p. 141–149, 2009.

Apêndices

APÊNDICE A – Código para geração dos *scripts slurm*

```
#!/bin/bash
#Script para gerar arquivos .srm para o slurm
#Universidade Estadual de Santa Cruz
#PPGMC - Programa de Pós Graduação em Modelagem Computacional
#Mestrado em Modelagem Computacional
#Mestrando Paulo Oliveria Paixão
#paulpaixao@gmail.com

#=====================================================================
=======================
for arq in $(ls -d *S08_MPC) #Lista os diretórios
do
  cd $arq  #Entra no diretório
for ((i=1;i<=8;i++))do
  touch ${arq}${i}.srm; ##Cria o arquivo .srm
  echo "#!/bin/bash" >> ${arq}${i}.srm;
        ##Seleciona caracteres em posições específicas a partir do
        nome do diretrório
  echo "#SBATCH -J
  ${arq:0:1}${arq:7:1}${arq:10:1}_${arq:17:2}_${arq:25:2}" >>
  ${arq}${i}.srm;
  echo "#SBATCH --partition long" >> ${arq}${i}.srm;
  echo "#SBATCH --nodes 1" >> ${arq}${i}.srm;
  echo "#SBATCH --ntasks 1" >> ${arq}${i}.srm;
  echo "#SBATCH --cpus-per-task 1" >> ${arq}${i}.srm;
  echo 'export=$PATH:/usr/local/root/bin' >> ${arq}${i}.srm;
  echo "srun Gate ${arq}${i}.mac" >> ${arq}${i}.srm;
done
  cd .. #Sai do diretório
done
#=====================================================================
=======================

## Gerador de 16 arquivos .srm
#=====================================================================
=======================
for arq in $(ls -d *S16_MPC) #Lista os diretórios
do
        cd $arq  #Entra no diretório
for ((i=1;i<=16;i++))do
        touch ${arq}${i}.srm;   ##Cria o arquivo .srm
        echo "#!/bin/bash" >> ${arq}${i}.srm;
        ##Seleciona caracteres em posições específicas a partir do
        nome do diretrório
        echo "#SBATCH -J
        ${arq:0:1}${arq:7:1}${arq:10:1}_${arq:17:2}_${arq:25:2}" >>
        ${arq}${i}.srm;
        echo "#SBATCH --partition long" >> ${arq}${i}.srm;
        echo "#SBATCH --nodes 1" >> ${arq}${i}.srm;
        echo "#SBATCH --ntasks 1" >> ${arq}${i}.srm;
        echo "#SBATCH --cpus-per-task 1" >> ${arq}${i}.srm;
        echo 'export=$PATH:/usr/local/root/bin' >> ${arq}${i}.srm;
        echo "srun Gate ${arq}${i}.mac" >> ${arq}${i}.srm;
done
        cd .. #Sai do diretório
done
#=====================================================================
=======================
```

APÊNDICE B – Definição dos fantomas utilizados nas simulações

Aqui estão definidas apenas geometria de cada fantoma.

```
#=====================================================
# Definição do WORLD para os três Fantomas
#=====================================================
/gate/world/geometry/setXLength 101. cm
/gate/world/geometry/setYLength 101. cm
/gate/world/geometry/setZLength 101. cm
/gate/world/setMaterial Air

#=====================================================
# Definição do Fantoma de BOX
#=====================================================
/gate/world/daughters/name phantomBox
/gate/world/daughters/insert box
/gate/phantomBox/geometry/setXLength 100 cm
/gate/phantomBox/geometry/setYLength 100 cm
/gate/phantomBox/geometry/setZLength 100 cm
/gate/phantomBox/setMaterial Water
/gate/phantomBox/vis/setVisible 1
/gate/phantomBox/vis/setColor blue
#=====================================================

#=====================================================
# Definição do Fantoma de VOXEL 100x100x100
# Cada elemento possui resolução de 1,0 cm
#=====================================================
/gate/world/daughters/name voxel_100
/gate/world/daughters/insert ImageRegularParametrisedVolume
/gate/voxel_100/geometry/setRangeToMaterialFile data/range_atten.dat
/gate/voxel_100/geometry/setImage data/water_100x100x100.mhd
/gate/voxel_100/placement/setTranslation 0. 0. 0. mm
/gate/voxel_100/placement/setRotationAxis 1 0 0
/gate/voxel_100/placement/setRotationAngle 0 deg
```

```
#=================================================
# Definição do Fantoma de VOXEL 200x200x200
# Cada elemento possui resolução de 0,5 cm
#=================================================
/gate/world/daughters/name voxel_200
/gate/world/daughters/insert ImageRegularParametrisedVolume
/gate/voxel_200/geometry/setRangeToMaterialFile data/range_atten.dat
/gate/voxel_200/geometry/setImage data/water_200x200x200.mhd
/gate/voxel_200/placement/setTranslation 0. 0. 0. mm
/gate/voxel_200/placement/setRotationAxis 1 0 0
/gate/voxel_200/placement/setRotationAngle 0 deg
```

APÊNDICE C – Script para extração de tempos nos arquivos das simulações paralelas de oito e dezesseis *jobs*

```
#!/bin/bash
#============================Cluster CTR=================================
#Script para ler tempos paralelos e calcular a média
#para cada tipo de simulação 04 jobs
#Universidade Estadual de Santa Cruz
#PPGMC - Programa de Pós Graduação em Modelagem Computacional
#Mestrado em Modelagem Computacional
#Mestrando Paulo Oliveira Paixão
#paulpaixao@gmail.com
#========================================================================
#Recomenda-se executar este arquivo com /bin/bash
#========================================================================
function ler_tempo04(){
soma=0
media=0
horas=0
quant_arquivo=0
minutos=0
tempo_04="TIME_CARBON_GAMMA_PROTON_04"
    #echo "Lendo tempos de simulação do arquivo..."
    cat /dev/null > ${tempo_04}.txt
    for diretorio in $(ls -d *S04_MPC)
    do #Lista os diretórios
  soma=0
        if [ -e ${diretorio}-tempo.txt ]
        then
        cat /dev/null > ${diretorio}-tempo.txt
        else
        touch ${diretorio}-tempo.txt
        fi
        if [ ! -z $diretorio ]
        then
        cd $diretorio
        for nome_arquivo in $(ls ${diretorio}*.txt)
           do #Lista quantidade de arquivos
        # tempo de simulação
              grep "ElapsedTime " $nome_arquivo | cut -d" " -f14 >>
              ../${diretorio}-tempo.txt
           done
        cd ..
        else
        echo "Diretório vazio"
        fi
        for valor in $(cat ${diretorio}-tempo.txt)
        do
           soma=$(echo "$soma+$valor" | bc)
        done
        quant_arquivo=$(cat ${diretorio}-tempo.txt | wc -l) #Conta
        quantidade de arquivos
        if [ $quant_arquivo -gt 0 ]
        then
           media=$(echo "$soma/$quant_arquivo" | bc)
           minutos=$(echo "scale=2;$media/60" | bc)
           horas=$(echo "scale=4;$media/3600" | bc)
           echo "Media = "$media" Tempo = "$minutos  >>
           ${diretorio}-tempo.txt
     echo "Tempo simulação " $diretorio  >> ${tempo_04}.txt
     echo "Média hor = "$horas >> ${tempo_04}.txt
     echo "Média min = "$minutos >> ${tempo_04}.txt
     echo "Média seg = "$media >> ${tempo_04}.txt
        fi
    done
}
```

```
#=========================================================================
function ler_tempo08(){
soma=0
media=0
horas=0
quant_arquivo=0
minutos=0
tempo_08="TIME_CARBON_GAMMA_PROTON_08"
    #echo "Lendo tempos de simulação do arquivo..."
    cat /dev/null > ${tempo_08}.txt
    for diretorio in $(ls -d *S08_MPC) #lista diretórios
    do #Lista os diretórios
  soma=0
  minutos=0
  if [ -e ${diretorio}-tempo.txt ] #verifica se arquivos existem
  then
  cat /dev/null > ${diretorio}-tempo.txt #limpa conteúdo do arquivo
  else
  touch ${diretorio}-tempo.txt #cria diretório
  fi
  if [ ! -z $diretorio ] #verifica se o diretório existe
  then
  cd $diretorio
  for nome_arquivo in $(ls ${diretorio}*.txt)
     do #Lista quantidade de arquivos
        # tempo de simulação
              grep "ElapsedTime " $nome_arquivo | cut -d" " -f14 >>
              ../${diretorio}-tempo.txt
            done
  cd ..
  else
  echo "Diretório vazio"
  fi
  for valor in $(cat ${diretorio}-tempo.txt)
  do
            soma=$(echo "$soma+$valor" | bc) # faz chamada à calculadora
            bash
  done
  quant_arquivo=$(cat ${diretorio}-tempo.txt | wc -l) #Conta quantidade
  de arquivos
  if [ $quant_arquivo -gt 0 ] #quantidade de arquivos diferente de zero
  then
     media=$(echo "$soma/$quant_arquivo" | bc)
     minutos=$(echo "scale=2;$media/60" | bc)
     horas=$(echo "scale=4;$media/3600" | bc)
           echo "Media = "$media "Tempo = "$minutos  >>
           ${diretorio}-tempo.txt
     echo "Tempo simulação " $diretorio  >> ${tempo_08}.txt
     echo "Média hor = "$horas >> ${tempo_08}.txt
     echo "Média min = "$minutos >> ${tempo_08}.txt
     echo "Média seg = "$media >> ${tempo_08}.txt
  fi
    done
}
#=========================================================================
ler_tempo04;
ler_tempo08;
```

APÊNDICE D – Tabelas de resultados obtidos nas simulações do CTR

Tabela 10 – Tempos registrados para partícula carbono e fantoma BOX no *Cluster* CTR

Histórias	1 CPU (min)	4 CPUs (min)	8 CPUs (min)	Speedup 4 CPUs	Eficiência 4 CPUs	Speedup 8 CPUs	Eficiência 8 CPUs
10^5	73,4	24,6	11,6	3,0	75%	6,3	79%
10^6	745	267	146,8	2,8	70%	5,1	63%
10^7	8468	3930	1980	2,2	54%	4,3	53%
10^8	127860,0	41067,0	21420,0	3,1	78%	6,0	75%

Tabela 11 – Tempos registrados para partícula gama e fantoma de BOX no *Cluster* CTR

Histórias	1 CPU (min)	4 CPUs (min)	8 CPUs (min)	Speedup 4 CPUs	Eficiência 4 CPUs	Speedup 8 CPUs	Eficiência 8 CPUs
10^5	1,16	0,33	0,15	3,5	88%	7,7	97%
10^6	11,13	2,7	1,95	4,1	103%	5,7	71%
10^7	103,7	32,4	13,2	3,2	80%	7,9	98%
10^8	1140	292,6	133,2	3,9	97%	8,6	107%

Tabela 12 – Tempos registrados para partícula próton e fantoma de BOX no *Cluster* CTR

Histórias	1 CPU (min)	4 CPUs (min)	8 CPUs (min)	Speedup 4 CPUs	Eficiência 4 CPUs	Speedup 8 CPUs	Eficiência 8 CPUs
10^5	37,0	12,3	4,6	3,0	75%	8,0	100%
10^6	392,0	134,8	63,2	2,9	73%	6,2	78%
10^7	3327,0	1148,0	521,9	2,9	72%	6,4	80%
10^8	39403,0	11399,8	5870,0	3,5	86%	6,7	84%

Tabela 13 – Tempos registrados para partícula carbono e fantoma VOXEL 100 no *Cluster* CTR

Histórias	1 CPU (min)	4 CPUs (min)	8 CPUs (min)	Speedup 4 CPUs	Eficiência 4 CPUs	Speedup 8 CPUs	Eficiência 8 CPUs
10^5	121	38,5	21,3	3,1	79%	5,7	71%
10^6	1397	470	235	3,0	74%	5,9	74%
10^7	14068,0	4675,0	2310,0	3,0	75%	6,1	76%
10^8	168340,0	43780,0	25460,0	3,8	96%	6,6	83%

Tabela 14 – Tempos registrados para partícula gama e fantoma de VOXEL 100 no *Cluster* CTR

Histórias	1 CPU (min)	4 CPUs (min)	8 CPUs (min)	Speedup 4 CPUs	Eficiência 4 CPUs	Speedup 8 CPUs	Eficiência 8 CPUs
10^5	6,99	1,76	0,9	4,0	99%	7,8	97%
10^6	69,7	20,83	8,71	3,3	84%	8,0	100%
10^7	687,4	259,3	86,23	2,7	66%	8,0	100%
10^8	7068	1759	863,9	4,0	100%	8,2	102%

Tabela 15 – Tempos registrados para partícula próton e fantoma de VOXEL 100 no *Cluster* CTR

Histórias	1 CPU (min)	4 CPUs (min)	8 CPUs (min)	Speedup 4 CPUs	Eficiência 4 CPUs	Speedup 8 CPUs	Eficiência 8 CPUs
10^5	41,6	14,2	5,2	2,9	73%	8,0	100%
10^6	454,9	161,3	67,4	2,8	70%	6,7	84%
10^7	4368,0	1233,8	630,3	3,5	89%	6,9	87%
10^8	50199,5	13050,5	6334,3	3,8	96%	7,9	99%

Tabela 16 – Tempos registrados para partícula carbono e fantoma VOXEL 200 no *Cluster* CTR

Histórias	1 CPU (min)	4 CPUs (min)	8 CPUs (min)	Speedup 4 CPUs	Eficiência 4 CPUs	Speedup 8 CPUs	Eficiência 8 CPUs
10^5	158,0	42	21,9	3,8	94%	7,2	90%
10^6	1754,0	493,0	238,0	3,6	89%	7,4	92%
10^7	19380,0	5150,0	2470,0	3,8	94%	7,8	98%
10^8	208330	53420	25070	3,9	97%	8,3	104%

Tabela 17 – Tempos registrados para partícula gama e fantoma de VOXEL 200 no *Cluster* CTR

Histórias	1 CPU (min)	4 CPUs (min)	8 CPUs (min)	Speedup 4 CPUs	Eficiência 4 CPUs	Speedup 8 CPUs	Eficiência 8 CPUs
10^5	21,19	5,7	2,7	3,7	93%	7,8	98%
10^6	227,8	57,3	29,7	4,0	99%	7,7	96%
10^7	2030	538	267,8	3,8	94%	7,6	95%
10^8	24430	7248	3115	3,4	84%	7,8	98%

Tabela 18 – Tempos registrados para partícula próton e fantoma de VOXEL 200 no *Cluster* CTR

Histórias	1 CPU (min)	4 CPUs (min)	8 CPUs (min)	Speedup 4 CPUs	Eficiência 4 CPUs	Speedup 8 CPUs	Eficiência 8 CPUs
10^5	92,2	23,1	12,4	4,0	100%	7,4	93%
10^6	995,9	258,0	133,0	3,9	97%	7,5	94%
10^7	11114,1	2940,0	1394,0	3,8	95%	8,0	100%
10^8	105390,0	26720,0	14538,0	3,9	99%	7,2	91%

APÊNDICE E – Tabelas de resultados obtidos nas simulações do CACAU

Tabela 19 – Tempos registrados para partícula carbono e fantoma de BOX no *Cluster* CACAU

Histórias	1 CPU (min)	8 CPUs (min)	16 CPUs (min)	Speedup 8 CPUs	Eficiência 8 CPUs	Speedup 16 CPUs	Eficiência 16 CPUs
10^5	59	8,1	3,7	7,3	91%	15,9	100%
10^6	670	85	53	7,8	98%	12,7	79%
10^7	7462	1227	740	6,1	76%	10,1	63%
10^8	72621	7502	5402	9,7	121%	13,4	84%

Tabela 20 – Tempos registrados para partícula gama e fantoma de BOX no *Cluster* CACAU

Histórias	1 CPU (min)	8 CPUs (min)	16 CPUs (min)	Speedup 8 CPUs	Eficiência 8 CPUs	Speedup 16 CPUs	Eficiência 16 CPUs
10^5	1,1	0,5	0,2	2,3	29%	6,2	39%
10^6	11,7	2	0,9	5,8	73%	13,3	83%
10^7	122	18	9,8	6,9	86%	12,5	78%
10^8	1198	221	93	5,4	68%	12,8	80%

Tabela 21 – Tempos registrados para partícula próton e fantoma de BOX no *Cluster* CACAU

Histórias	1 CPU (min)	8 CPUs (min)	16 CPUs (min)	Speedup 8 CPUs	Eficiência 8 CPUs	Speedup 16 CPUs	Eficiência 16 CPUs
10^5	3,1	0,5	0,3	6,5	81%	10,3	65%
10^6	33	4	2,1	8,3	104%	15,8	99%
10^7	382	47,4	28,5	8,1	101%	13,4	84%
10^8	3674	634	270	5,8	72%	13,6	85%

Tabela 22 – Tempos registrados para partícula carbono e fantoma VOXEL 100 no *Cluster* CACAU

.

Histórias	1 CPU (min)	8 CPUs (min)	16 CPUs (min)	Speedup 8 CPUs	Eficiência 8 CPUs	Speedup 16 CPUs	Eficiência 16 CPUs
10^5	110	13,8	7,7	8,0	100%	14,3	89%
10^6	1332	236	139	5,7	71%	9,6	60%
10^7	13553	5233	2021	2,6	32%	6,7	42%
10^8	135869	17892	5157	7,6	95%	26,3	165%

Tabela 23 – Tempos registrados para partícula gama e fantoma de VOXEL 100 no *Cluster* CACAU

Histórias	1 CPU (min)	8 CPUs (min)	16 CPUs (min)	Speedup 8 CPUs	Eficiência 8 CPUs	Speedup 16 CPUs	Eficiência 16 CPUs
10^5	5,8	0,7	0,4	8,2	103%	14,6	91%
10^6	62,7	8,2	3,7	7,6	96%	17,1	107%
10^7	648	82	38	7,9	99%	17,0	106%
10^8	6081	1149	450	5,3	66%	13,5	84%

Tabela 24 – Tempos registrados para partícula próton e fantoma de *VOXEL* 100 no *Cluster* CACAU

Histórias	1 CPU (min)	8 CPUs (min)	16 CPUs (min)	Speedup 8 CPUs	Eficiência 8 CPUs	Speedup 16 CPUs	Eficiência 16 CPUs
10^5	5,5	1,4	0,38	3,9	49%	14,6	91%
10^6	58	7	3,6	8,4	105%	16,0	100%
10^7	620	95	40	6,5	82%	15,6	97%
10^8	6579	941	457	7,0	87%	14,4	90%

Tabela 25 – Tempos registrados para partícula carbono e fantoma VOXEL no *Cluster* CACAU

Histórias	1 CPU (min)	8 CPUs (min)	16 CPUs (min)	Speedup 8 CPUs	Eficiência 8 CPUs	Speedup 16 CPUs	Eficiência 16 CPUs
10^5	142	19,7	9,5	7,2	90%	15,0	94%
10^6	1590	236	118	6,7	84%	13,4	84%
10^7	28964	3628	7345	8,0	100%	3,9	25%
10^8	280714	24565	10416	11,4	143%	27,0	168%

Tabela 26 – Tempos registrados para partícula gama e fantoma de VOXEL 200 no *Cluster* CACAU

Histórias	1 CPU (min)	8 CPUs (min)	16 CPUs (min)	Speedup 8 CPUs	Eficiência 8 CPUs	Speedup 16 CPUs	Eficiência 16 CPUs
10^5	8,8	1,1	0,7	7,9	99%	12,9	81%
10^6	102	13,2	7	7,8	97%	14,7	92%
10^7	994	179	125	5,5	69%	8,0	50%
10^8	9774	1404	700	7,0	87%	14,0	87%

Tabela 27 – Tempos registrados para partícula próton e fantoma de VOXEL 200 no *Cluster* CACAU

Histórias	1 CPU (min)	8 CPUs (min)	16 CPUs (min)	Speedup 8 CPUs	Eficiência 8 CPUs	Speedup 16 CPUs	Eficiência 16 CPUs
10^5	14	1,8	0,9	8,0	100%	15,6	97%
10^6	153	22,8	14,9	6,7	84%	10,3	64%
10^7	1621	250	93	6,5	81%	17,4	109%
10^8	16621	2132	1372	7,8	97%	12,1	76%

Anexos

ANEXO A – Tabela de equivalência de comandos PBS/Torque para Slurm

						28-Apr-2013
User Commands	**PBS/Torque**	**Slurm**	**LSF**	**SGE**	**LoadLeveler**	
Job submission	qsub [script_file]	sbatch [script_file]	bsub [script_file]	qsub [script_file]	llsubmit [script_file]	
Job deletion	qdel [job_id]	scancel [job_id]	bkill [job_id]	qdel [job_id]	llcancel [job_id]	
Job status (by job)	qstat [job_id]	squeue [job_id]	bjobs [job_id]	qstat -u * [-j job_id]	llq -u [username]	
Job status (by user)	qstat -u [user_name]	squeue -u [user_name]	bjobs -u [user_name]	qstat [-u user_name]	llq -u [user_name]	
Job hold	qhold [job_id]	scontrol hold [job_id]	bstop [job_id]	qhold [job_id]	llhold -r [job_id]	
Job release	qrls [job_id]	scontrol release [job_id]	bresume [job_id]	qrls [job_id]	llhold -r [job_id]	
Queue list	qstat -Q	squeue	bqueues	qconf -sql	llclass	
Node list	pbsnodes -l	sinfo -N OR scontrol show nodes	bhosts	qhost	llstatus -L machine	
Cluster status	qstat -a	sinfo	bqueues	qhost -q	llstatus -L cluster	
GUI	xpbsmon	sview	xlsf OR xlsbatch	qmon	xload	
Environment	**PBS/Torque**	**Slurm**	**LSF**	**SGE**	**LoadLeveler**	
Job ID	$PBS_JOBID	$SLURM_JOBID	$LSB_JOBID	$JOB_ID	$LOAD_STEP_ID	
Submit Directory	$PBS_O_WORKDIR	$SLURM_SUBMIT_DIR	$LSB_SUBCWD	$SGE_O_WORKDIR	$LOADL_STEP_INITDIR	
Submit Host	$PBS_O_HOST	$SLURM_SUBMIT_HOST	$LSB_SUB_HOST	$SGE_O_HOST		
Node List	$PBS_NODEFILE	$SLURM_JOB_NODELIST	$LSB_HOSTS/LSB_MCPU_HOST	$PE_HOSTFILE	$LOADL_PROCESSOR_LIST	
Job Array Index	$PBS_ARRAYID	$SLURM_ARRAY_TASK_ID	$LSB_JOBINDEX	$SGE_TASK_ID		
Job Specification	**PBS/Torque**	**Slurm**	**LSF**	**SGE**	**LoadLeveler**	
Script directive	#PBS	#SBATCH	#BSUB	#$	#@	
Queue	-q [queue]	-p [queue]	-q [queue]	-q [queue]	class=[queue]	
Node Count	-l nodes=[count]	-N [min[-max]]	-n [count]	N/A	node=[count]	
CPU Count	-l ppn=[count] OR -l mppwidth=[PE_count]	-n [count]	-n [count]	-pe [PE] [count]		
Wall Clock Limit	-l walltime=[hh:mm:ss]	-t [min] OR -t [days-hh:mm:ss]	-W [hh:mm:ss]	-l h_rt=[seconds]	wall_clock_limit=[hh:mm:ss]	
Standard Output File	-o [file_name]	-o [file_name]	-o [file_name]	-o [file_name]	output=[file_name]	
Standard Error File	-e [file_name]	e [file_name]	-e [file_name]	-e [file_name]	error=[File_name]	
Combine stdout/err	-j oe (both to stdout) OR -j eo (both to stderr)	(use -o without -e)		-j yes		
Copy Environment	-V	--export=[ALL | NONE | variables]	-V	-V	environment=COPY_ALL	
Event Notification	-m abe	--mail-type=[events]	-B or -N	-m abe	notification=start|error|complete|never|always	
Email Address	-M [address]	--mail-user=[address]	-u [address]	-M [address]	notify_user=[address]	
Job Name	-N [name]	--job-name=[name]	-J [name]	-N [name]	job_name=[name]	
Job Restart	-r [y|n]	--requeue OR --no-requeue (NOTE: configurable default)	-r	-r [yes|no]	restart=[yes|no]	
Working Directory	N/A	--workdir=[dir_name]	(submission directory)	-wd [directory]	initialdir=[directory]	
Resource Sharing	-l naccesspolicy=singlejob	--exclusive OR--shared	-x	-l exclusive	node_usage=not_shared	
Memory Size	-l mem=[MB]	--mem=[mem][M|G|T] OR --mem-per-cpu=[mem][M|G|T]	-M [MB]	-l mem_free=[memory][K|M|G]	requirements=(Memory >= [MB])	
Account to charge	-W group_list=[account]	--account=[account]	-P [account]	-A [account]		
Tasks Per Node	-l mppnppn [PEs_per_node]	--tasks-per-node=[count]		(Fixed allocation_rule in PE)	tasks_per_node=[count]	
CPUs Per Task		--cpus-per-task=[count]				
Job Dependency	-d [job_id]	--depend=[state:job_id]	-w [done | exit | finish]	-hold_jid [job_id | job_name]		
Job Project		--wckey=[name]	-P [name]	-P [name]		
Job host preference		--nodelist=[nodes] AND/OR --exclude=[nodes]	-m [nodes]	-q [queue]@[node] OR -q [queue]@@[hostgroup]		
Quality Of Service	-l qos=[name]	--qos=[name]				
Job Arrays	-t [array_spec]	--array=[array_spec] (Slurm version 2.6+)	J "name[array_spec]"	-t [array_spec]		
Generic Resources	-l other=[resource_spec]	--gres=[resource_spec]		-l [resource]=[value]		
Licenses		--licenses=[license_spec]	-R "rusage[license_spec]"	-l [license]=[count]		
Begin Time	-A "YYYY-MM-DD HH:MM:SS"	--begin=YYYY-MM-DD[THH:MM[:SS]]	-b[[year:][month:]daty:]hour:minute	-a [YYMMDDhhmm]		