João Pedro Resende Barroso

Interface Gráfica Para Simulação De Sistemas Exascale

São José do Rio Preto

2023

João Pedro Resende Barroso

Interface Gráfica Para Simulação De Sistemas Exascale

Trabalho de Conclusão de Curso apresentado como parte dos requisitos para obtenção do título de Bacharel, junto ao Curso de Graduação em Ciência da Computação, do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista "Júlio de Mesquita Filho", Campus de São José do Rio Preto.

Orientador: Aleardo Manacero Júnior

São José do Rio Preto

2023

B277i

Barroso, João Pedro Resende
Interface gráfica para simulação de sistemas exascale / João Pedro Resende Barroso. -- São José do Rio Preto, 2023
38 p. : il., tabs.

Trabalho de conclusão de curso (Bacharelado - Ciência da Computação) - Universidade Estadual Paulista (Unesp), Instituto de Biociências Letras e Ciências Exatas, São José do Rio Preto
Orientador: Aleardo Manacero

1. Simulação. 2. Sistemas de computação. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Biblioteca do Instituto de Biociências Letras e Ciências Exatas, São José do Rio Preto. Dados fornecidos pelo autor(a).

Essa ficha não pode ser modificada.

João Pedro Resende Barroso

Interface Gráfica Para Simulação De Sistemas Exascale

Trabalho de Conclusão de Curso apresentado como parte dos requisitos para obtenção do título de Bacharel, junto ao Curso de Graduação em Ciência da Computação, do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista "Júlio de Mesquita Filho", Campus de São José do Rio Preto.

Banca Examinadora

Prof. Dr. Aleardo Manacero Júnior
UNESP - São José do Rio Preto
Orientador

Profª Drª. Renata Spolon Lobato
UNESP - São José do Rio Preto

Prof. Dr. Lucas Correia Ribas
UNESP - São José do Rio Preto

São José do Rio Preto
16 de novembro de 2023

Agradecimentos

Primeiramente gostaria de agradecer minha mãe Geane Barroso e meu pai João Barroso pelo eterno apoio ao meu ensino e por todos esses anos aturando minhas loucuras.

Ao meu orientador Prof. Dr. Aleardo Manacero, pela orientação durante todo esse projeto e pelo olhar mais maduro, que me fez pensar nos desafios com mais calma.

Aos meus amigos Ricardo Fares e Matheus Augusto pela companhia durante a faculdade e pela ajuda em diversos projetos.

A Isabela, que durante todo esses anos esteve me apoiando e foi responsável por mudar minha opinião sobre cursar medicina.

Por fim, gostaria de agradecer à FAPESP, que financiou o projeto por meio de bolsas (Processo N° 2022/15806-0) e ajudou na apresentação do projeto em eventos.

"I've seen things you people wouldn't believe...
Attack ships on fire off the shoulder of Orion...
I watched C-beams glitter in the dark near the Tannhäuser Gate.
All those moments will be lost in time, like tears in rain..."
- Roy Batty (Blade Runner)

Resumo

O crescimento de sistemas distribuídos de larga escala tem acelerado exponencialmente nos últimos anos, chegando até o nível de *exascale*. Por causa do custo de aluguel de sistemas desse tamanho, a simulação pode ser uma alternativa barata e eficiente para a avaliação de seu desempenho. Este trabalho tem como objetivo o estudo sobre sistemas *exascale* e o aperfeiçoamento do simulador iSPD (*iconic Simulator of Parallel and Distributed Systems*) de forma que seja possível a modelagem de sistemas de larga escala, assim como a visualização de resultados provenientes das simulações de forma coerente, permitindo assim que qualquer usuário sem conhecimento de programação consiga fazer uma simulação.

Palavras-chave: modelagem, simulação, sistemas de larga escala.

Abstract

The exponential growth of large-scale distributed systems in recent years, reaching the level of exascale, has been remarkable. Due to the high cost of renting such extensive systems, simulation emerges as a cost-effective and efficient alternative for evaluating their performance. This research aims to study exascale systems and enhance the iSPD (iconic Simulator of Parallel and Distributed Systems) simulator, enabling the modeling of large-scale systems and coherent visualization of simulation results. Consequently, this enhancement empowers non-programming users to perform simulations with ease and confidence.

Keywords: modeling, simulation, large-scale systems.

Lista de ilustrações

Lista de abreviaturas e siglas

Flops	floating point operations per second

Sumário

1 Introdução

Na última década houve um crescimento exponencial na capacidade de processamento computacional e no tamanho de sistemas distribuídos. Promovido por avanços tecnológicos acelerados, essas máquinas têm se tornando uma das principais ferramentas na resolução de complexos problemas em diversas áreas do conhecimento. Seu uso na análise de grandes volumes de dados e em simulações complexas mostra que supercomputadores desse tamanho são essenciais para o avanço contínuo da ciência e tecnologia.

Esses sistemas de larga escala, também denominados de supercomputadores, representam o ponto mais alto da evolução da computação. Sendo capazes de realizar trilhões de execuções por segundo, tem sua utilidades na área de meteorologia (WYBORN; EVANS, 2015), saúde (TSAI, 2018), astrologia e entre outras áreas. A demanda por maior poder de processamento impulsiona então um cenário competitivo na construção de supercomputadores cada vez mais rápidos e eficientes.

Esse crescimento acelerado pode ser observado segundo alguns indicadores. Um dos conceitos principais é a velocidade de processamento, sendo medido por FLOPS (*Floating-Point Operations Per Second*), o qual tem tido taxas crescentes nos últimos anos, sendo o marco mais recente o de *exaflops*, ou seja, 10^{18} FLOPS por segundo, abrindo então novas oportunidades para o desenvolvimento tecnológico.

1.1 Motivação

Apesar dos benefícios oriundos do crescimento exacerbado dos sistemas de larga escala, esse fator gera alguns problemas que precisam ser considerados. Um dos maiores deles é o custo de execução, uma vez que o aluguel de um sistema desse tamanho pode chegar na casa de milhões de dólares (PRABHAKARAN; J., 2018).

Tendo essa realidade em consideração, métodos foram desenvolvidos para a análise de desempenho desses sistemas (JAIN, 1991), sendo os mais comuns a modelagem analítica, simulações e medições proporcionais, sendo que o primeiro método tem um alto nível de complexidade, e o último depende de um sistema equivalente ao original, sendo então não recomendável a sua utilização em um sistema *exascale*. Tendo esses fatores em consideração, o método de simulação se torna o mais ideal, garantindo exatidão e baixo custo para a análise de sistemas de larga escala.

O uso de simuladores, entretanto, envolve a difícil tarefa de modelagem, que envolve conhecer a fundo a estrutura e as ferramentas que o simulador fornece, o que pode ser muito complexo quando se trata de um sistema de mais de 500 nós, por exemplo. Com isso

em mente, o simulador iSPD(GARCIA, 2010) foi desenvolvido para fornecer uma interface baseado em ícones para fazer a representação de máquinas e nós, fornecendo ferramentas simples e intuitivas.

Entretanto, o projeto original foi criado visando sistemas de médio porte e com configurações simples, como a topologia estrela e de barramento, sendo necessário novas ferramentas para a modelação de sistemas de larga escala atuais, que utilizam configurações complexas, como a topologia *Dragonfly* e *Fat-tree*.

1.2 Objetivo

O foco desse trabalho é a melhoria do simulador iSPD, visando a sua utilização para modelação de sistemas de larga escala e a visualização correta de resultados provenientes de uma simulação de cargas de trabalhos funcionando em um desses sistemas.

1.3 Organização do Texto

Este trabalho será dividido em 5 capítulos. Após esta introdução, teremos no capítulo 2 uma revisão da literatura, analisando o projeto iSPD, descrevendo suas ferramentas para a modelagem de sistemas distribuídos e mostrando como funciona o sistema de métricas provenientes de simulações.

No capítulo 3 será abordado a metodologia utilizada para o aprimoramento do simulador, apresentando técnicas encontradas em diversos artigos sobre a área de visualização de dados.

Posteriormente, teremos no capítulo 4 os resultados provenientes desse trabalho, mostrando como as técnicas abordadas foram implementadas no projeto.

Por fim teremos no capítulo 5 a conclusão e sugestões para projetos futuros.

2 Revisão da Literatura

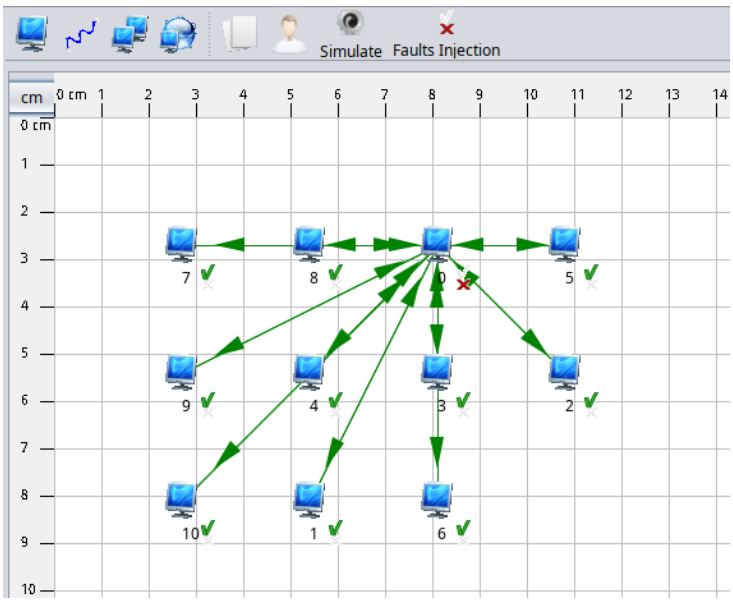
Nesse capítulo teremos inicialmente uma descrição de como o iSPD foi construído, tendo em consideração o contexto no qual foi criado e quais foram seus objetivos, mostrando as duas áreas principais que irão ser aprimoradas. Em seguida, será mostrado quais as mudanças que foram feitas no projeto original para que se consiga alcançar o objetivo final, que será fornecer as ferramentas necessárias para a simulação de sistemas de larga escala.

2.1 Projeto original

Conforme descrito em (GUERRA, 2010), o iSPD foi construído como uma opção a simuladores de sistemas distribuídos da época, como o *SimGrid*(CASANOVA, 2001) e o *GridSim* (BUYYA; MURSHED, 2002), que tinham a capacidade de simular corretamente sistemas distribuídos com eficácia mas demandavam do usuário um alto conhecimento sobre o simulador, além de terem uma interface voltada para especialistas no assunto, sendo necessário ter contato com código fonte.

Nesse contexto, o projeto iSPD foi construído com o foco na facilidade de modelação de um sistema distribuído, por meio da utilização de um interface gráfica e ferramentas para criação de máquinas, enlaces e *clusters*, conforme visualizado na Figura 1.

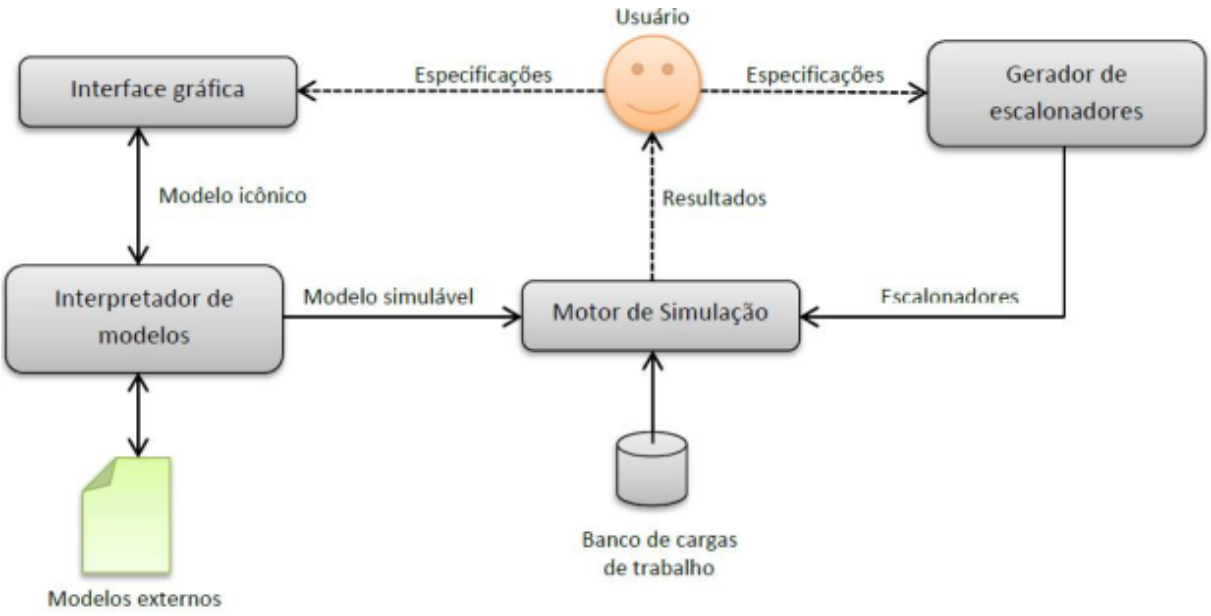
Figura 1 – Exemplo de ferramentas disponíveis para a modelagem de um sistema.



Fonte: Elaborado pelo autor.

Além da modelagem, o iSPD também é responsável por gerar cargas de trabalho para a simulação, além de ser possível criar novos escalonadores para testagem. Por fim, o

Figura 2 – *Workflow* do iSPD



Fonte: Elaborado pelo autor.

motor de simulação executa o modelo criado e gera métricas referentes a cada componente, que irão ser representadas utilizando gráficos e tabelas para mostrar a eficiência da carga de trabalho executada nessa configuração.

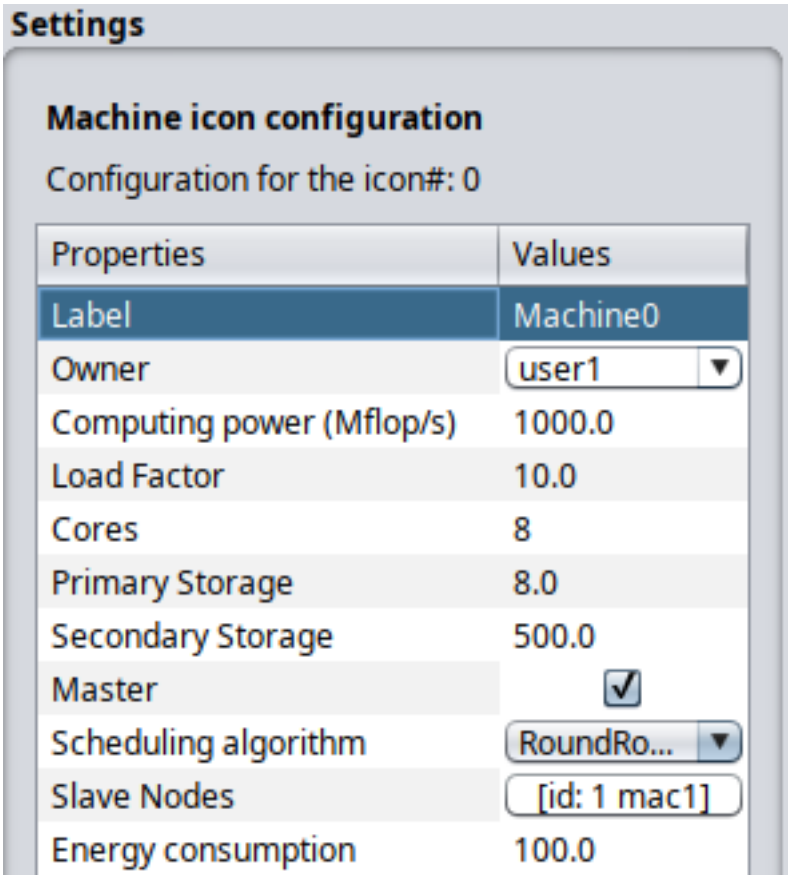
2.2 Atributos

Cada ícone tem atributos referentes ao objeto real a qual representa, tendo como base as características importantes desse componente. A seguir será descrito os atributos de cada componente encontrado no iSPD:

2.2.1 Máquina

- **Label:** Nome da máquina em questão.
- **Owner:** Usuário que está utilizando essa máquina.
- **Computing Power:** Quantos mega *FLOPS* aquela máquina consegue fazer.
- **Load Factor:** Quanto da *CPU* (porcentagem) está em utilização por segundo.
- **Cores:** Quantidade de *Cores* que a máquina tem.
- **Primary Storage:** Quantidade em *gigabytes* de memória RAM que a máquina tem.
- **Secondary Storage:** Quantidade em *gigabytes* de armazenamento secundário que a máquina tem.

Figura 3 – Configuração de uma máquina no iSPD original.



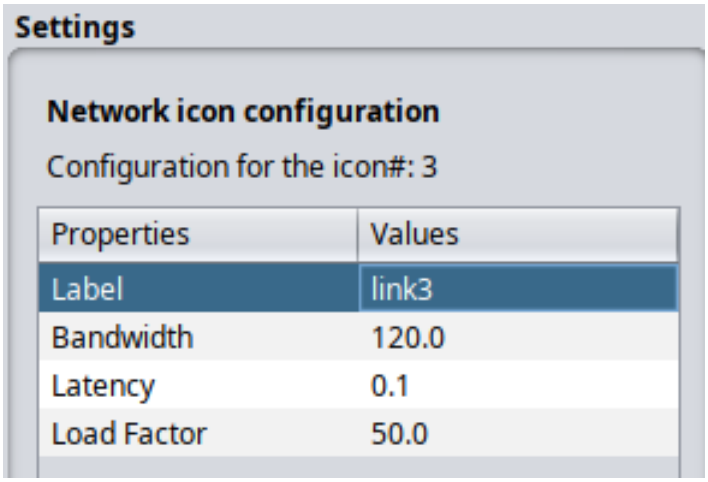
Fonte: Elaborado pelo autor.

- **Master:** Determinação se a máquina é um nó mestre ou escravo.
- **Scheduling Algorithm:** Qual o algoritmo de escalonamento utilizado pelo nó (caso seja um mestre).
- **Slave Nodes:** Quais os escravos dessa máquina (caso seja um mestre).
- **Gasto energético:** Quantidade de potência (em Watts) que a máquina consome.

2.2.2 Enlace

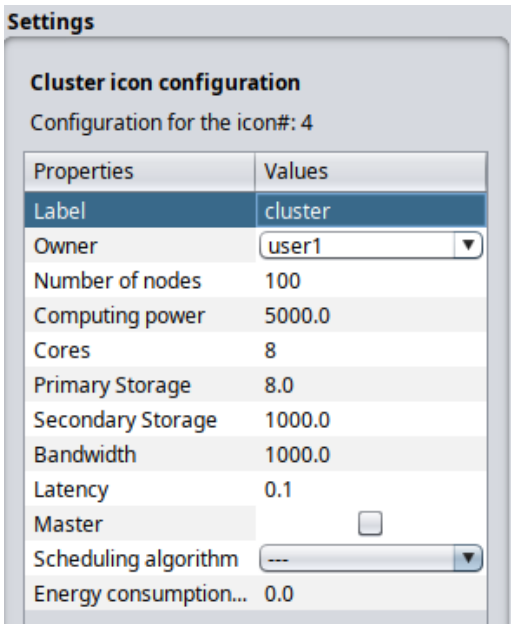
- **Label:** Nome do enlace em questão.
- **Bandwidth:** Velocidade do enlace em questão (em MB/s).
- **Latency:** Quantidade de atraso que o enlace introduz no sistema (em segundos).
- **Load Factor:** Porcentagem de tempo em que o enlace está ocupado.

Figura 4 – Configuração de um enlace no iSPD original.



Fonte: Elaborado pelo autor.

Figura 5 – Configuração de um *cluster* no iSPD original.



Fonte: Elaborado pelo autor.

2.2.3 Cluster

O conceito de *Cluster* no projeto original é descrito por um conjunto homogêneo de N máquinas, quantizada por *Number of Nodes*, como mostrado na Figura 5. Os outros atributos são os mesmos encontrados na página de configuração de uma máquina (Figura 3), exceto os atributo *Bandwidth* e *Latency*, que se referem aos enlaces (Figura 4) conectados a esse *cluster*.

Com esses atributos, é feito a criação do modelo simulável, o qual é utilizado para fazer a simulação de fato, onde será gerado as métricas utilizadas para visualizar a performance do modelo gerado.

2.3 Métricas

Além de atributos relacionados a cada objeto, existe também as métricas relacionadas a como cada componente reagiu a certa carga de trabalho, com o intuito de obter informações importantes para a determinação da eficiência de um determinado sistema distribuído. Em seguida será descrito as principais métricas obtidas e no final como elas são visualizadas no simulador.

2.3.1 Processamento

As métricas relacionadas ao processamento são obtidas a partir da relação entre as máquinas e a carga de trabalho criada pelo usuário.

- **Flops Processados:** Quantos *FLOPS* uma máquina executou, calculado a partir da quantidade de *FLOPS* que cada tarefa executada na máquina custou.
- **Tempo de Processamento:** Por quanto tempo a maquina gastou processando tarefas.

2.3.2 Comunicação

As métricas relacionadas à comunicação são obtidas a partir da relação entre os enlaces e quanta informação passou por eles.

- **Quantidade Transmitida:** Quantos *megabytes* um enlace transmitiu.
- **Tempo de transmissão:** Quanto tempo o enlace gastou transmitindo informação.

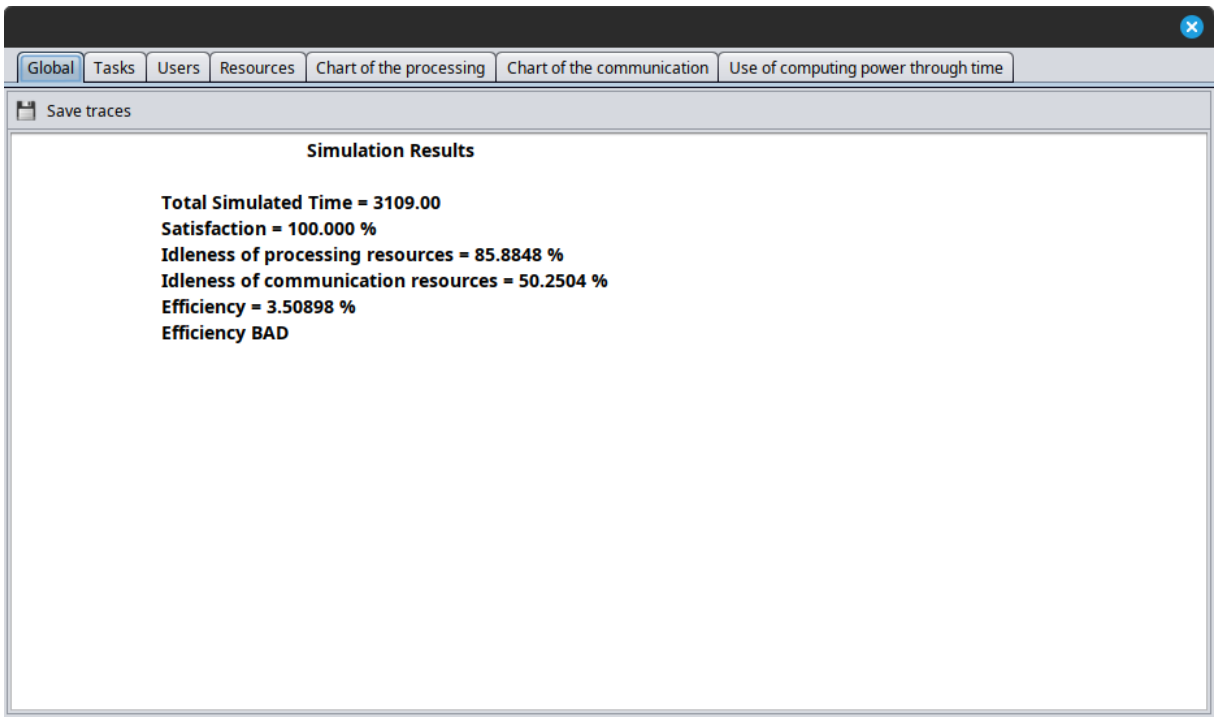
2.3.3 Globais

As métricas globais descrevem as estatísticas globais de uma simulação.

- **Quantidade Transmitida:** Quantos *megabytes* um enlace transmitiu a um nó.
- **Tempo de transmissão:** Quanto tempo o enlace gastou transmitindo informação.

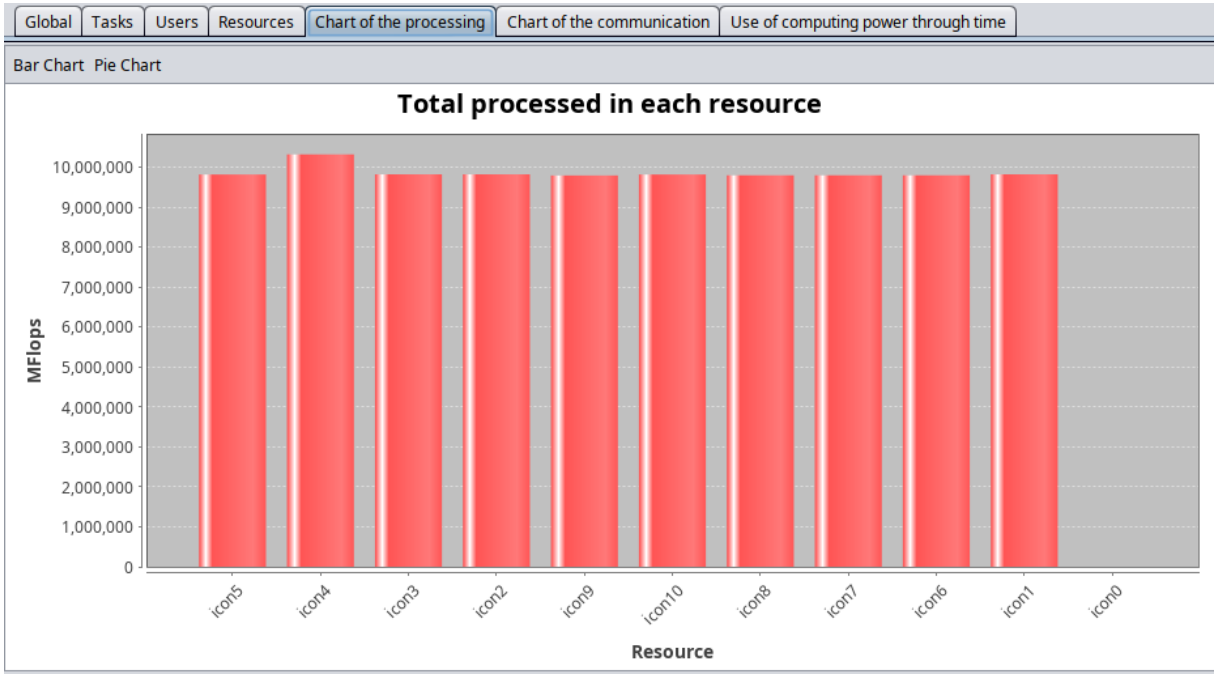
Essas métricas então são coletadas e representadas na interface gráfica, conforme visualizado nas Figuras 6, 8 e 7. Com essas informações, portanto, é possível fazer a análise sobre a performance de um sistema distribuído executando uma carga de trabalho específica.

Figura 6 – Informações gerais sobre a simulação.



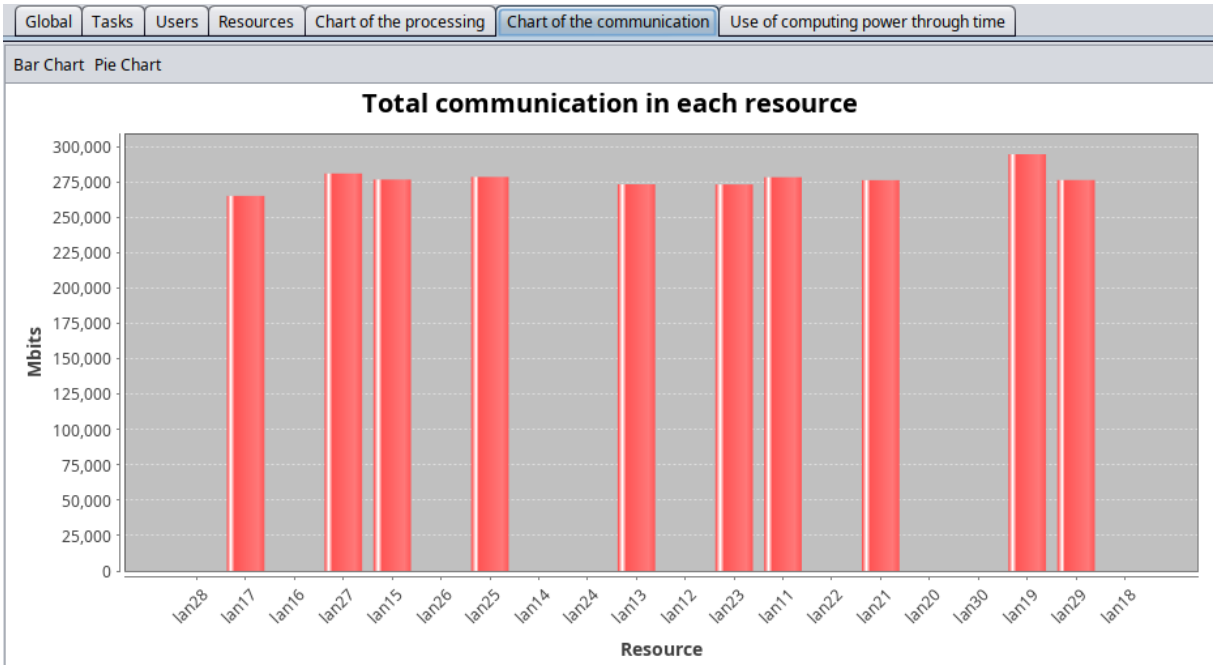
Fonte: Elaborado pelo autor.

Figura 7 – Gráfico representando o processamento que cada nó efetuou durante a simulação.



Fonte: Elaborado pelo autor.

Figura 8 – Gráfico representando a transmissão de cada enlace durante a simulação.



Fonte: Elaborado pelo autor.

2.4 Problemas Encontrados

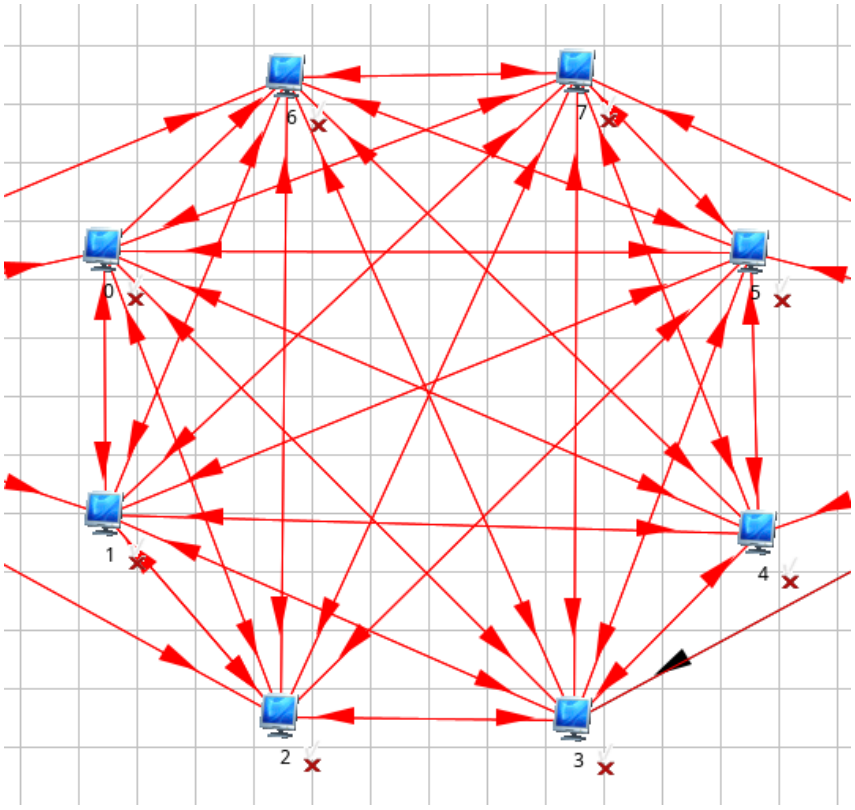
Por causa do seu foco na modelagem de sistemas pequenos e simples, o iSPD contêm somente as ferramentas necessárias para esse fim, isso quer dizer que qualquer configuração mais complexa teria que ser feita manualmente, tornando inviável a utilização do simulador nessas ocasiões.

O exemplo na Figura 9 apresenta uma subárea de um sistema com topologia *Dragonfly*, apesar de ser possível fazer essa configuração de máquinas no projeto original, a criação do sistema inteiro seria um trabalho difícil, pois teria que configurar cada máquina e verificar a conexão de todos os enlaces, portanto pode-se dizer que a modelagem não é facilmente escalável.

Além disso, na parte de visualização de resultados, como mostrado na Figura 10, é nítido a dificuldade em representar uma grande quantidade de informação de um vez, o que é inevitável em um sistema de larga escala, onde há mais de mil máquinas e enlaces conectados entre si.

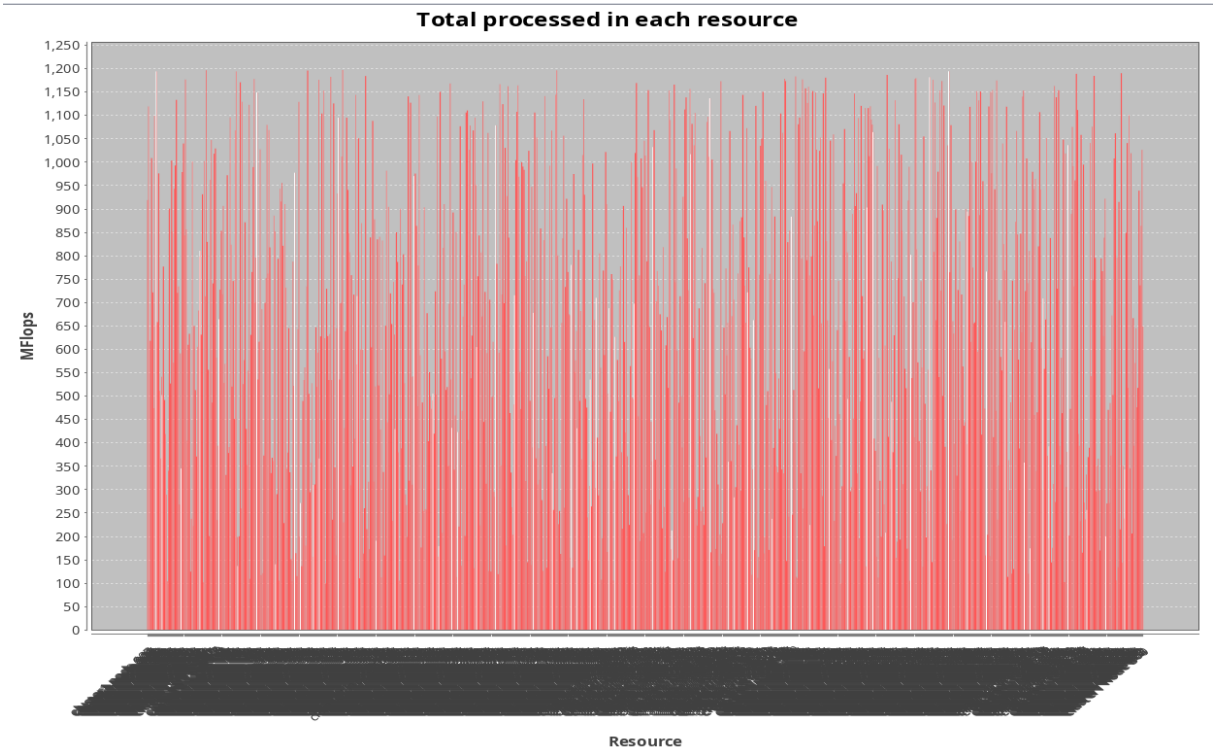
A causa dessa dificuldade é por causa da escolha de técnicas de representação pensadas somente na análise individual de máquinas, o que é comum em um sistema distribuído de pequeno porte. Porém, em um sistema com tantos nós, é preferível ter conhecimento de como um grupo de máquinas funciona, em vez de componentes isolados.

Figura 9 – Modelagem de uma subárea com 8 máquinas.



Fonte: Elaborado pelo autor.

Figura 10 – Gráfico de processamento de 1000 máquinas após uma simulação.



Fonte: Elaborado pelo autor.

3 Metodologia

Este capítulo refere-se ao trabalho efetuado para alcançar os objetivos propostos pela pesquisa. O processo foi separado em duas etapas, sendo elas a reimplementação do simulador *iSPD* e a implementação de novos métodos referentes à modelação de sistemas e a representação de resultados provenientes de simulações.

3.1 Reimplementação

Por causa de inúmeros trabalhos feito em cima do *iSPD* durante os anos, como (PAULO, 2017), (SILVA, 2012) e (TAMASHIRO, 2020), o projeto se tornou desnecessariamente complexo. Com esse fator em consideração, houve a decisão de fazer a reengenharia do código fonte, efetuando as seguintes alterações:

- Alteração da linguagem utilizada de *Java* para *C++* e *Python*.
- Utilização do *Framework Qt*
- Divisão de interface gráfica e motor de simulação

A mudança de linguagem de programação se deu por causa da necessidade de maior performance no manuseio da maior gama de dados e por fornecer bibliotecas de interfaces gráficas mais maduras, por meio do *Framework Qt*(BLANCHETTE; SUMMERFIELD, 2008), que tem foco no desenvolvimento *Desktop*, plataforma original do *iSPD*; além disso, na área de representação de métricas, foi determinado a utilização de bibliotecas em *Python*, como a *matplotlib* e *pandas*, pois elas fornecem ferramentas úteis que seriam de difícil implementação em *C++*.

Por fim, a interface gráfica é um componente independente do motor de simulação para tornar a modelação e visualização agnóstica da simulação em si, portanto é possível utilizar a parte gráfica com um motor de simulação que o usuário queira implementar, por exemplo.

3.2 Métodos Implementados

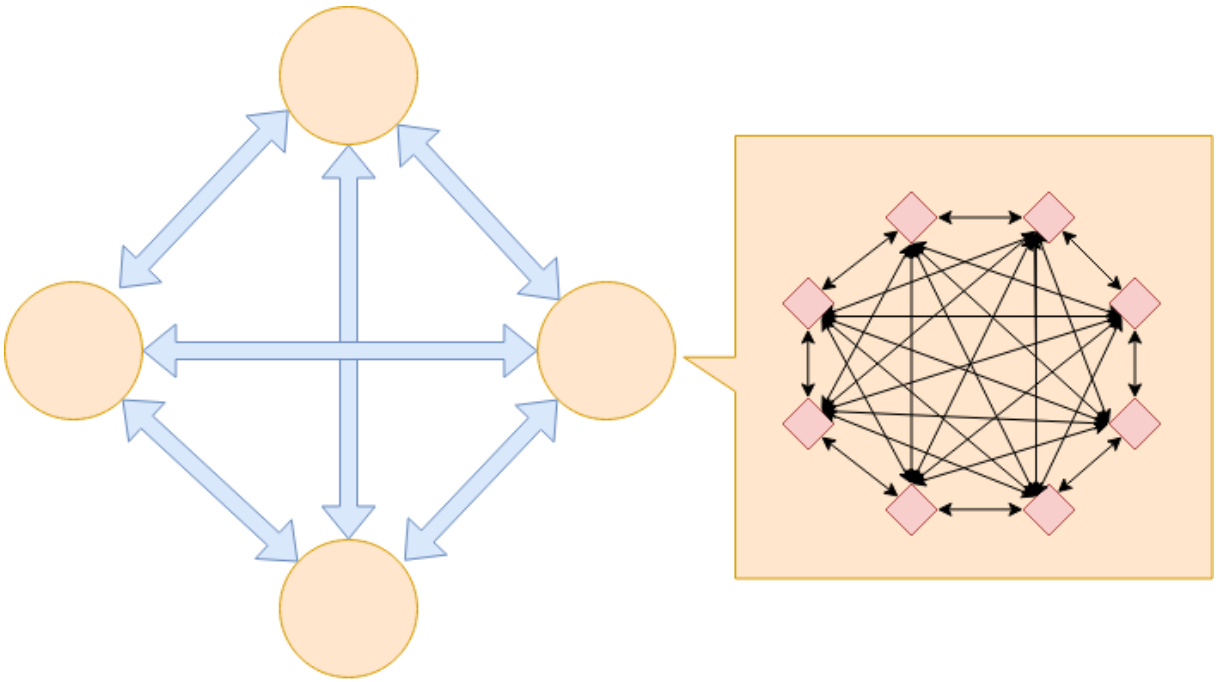
Para resolver os problemas encontrados no projeto original, foi necessário encontrar soluções em outras áreas de pesquisa, como na área de *Big Data* e na área da Medicina no contexto da pandemia, onde a criação de novas técnicas foi de extrema importância para a análise do impacto da doença pelo mundo.

3.2.1 Modelagem

O principal método a ser utilizado para o aperfeiçoamento da modelagem foi descrito em (KREKHOV et al., 2015) e se caracteriza pela criação de módulos para a concatenação de configurações heterogêneas de componentes. No contexto deste projeto, cada módulo descreve uma configuração compacta de máquinas, sendo necessário então criar somente módulos básicos para a modelação de um sistema maior e mais complexo.

Exemplo desse método em questão pode ser observada na Figura 11, na qual há 4 módulos interligados, cada um com sistema de máquinas interligadas. Com isso, pode-se observar que topologias mais complexas como *Dragonfly*, que é composta por vários conjuntos de máquinas, pode ser modelada facilmente utilizando esse método, sendo necessário somente criar um dos conjuntos, o qual pode ser replicado quantas vezes for necessário.

Figura 11 – 4 módulos conectados entre si, cada um com a mesma configuração.



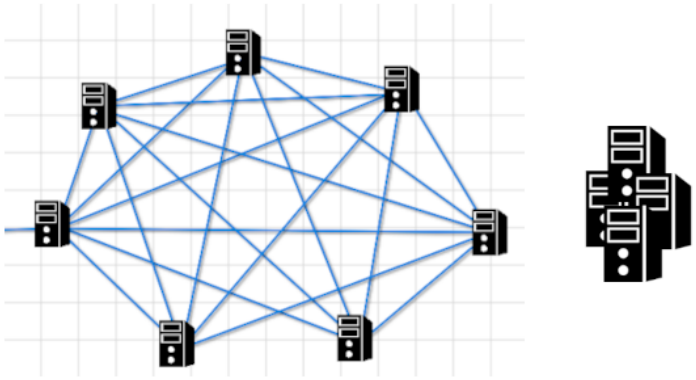
Fonte: Elaborado pelo autor.

Além do desenvolvimento de conjuntos heterogêneos, foi também estudado o desenvolvimento de simples ferramentas para desenvolver conjuntos homogêneos configuráveis(GUOZHENG; YULIANG; HUIXIAN, 2011), ou seja, diferentemente da implementação feita pelo iSPD, onde um *cluster* é somente um grupo de máquinas conectadas a outro componente, as novas ferramentas serão representativos de diferentes topologias, como na Figura 12, a qual representa 6 máquinas conectadas em série utilizando um único ícone, ou como na Figura 13, que representa um grupo de 7 máquinas iguais interlaçadas.

Figura 12 – Grupo homogêneo conectado em série.

Fonte: Elaborado pelo autor.

Figura 13 – Grupo homogêneo interligado.



Fonte: Elaborado pelo autor.

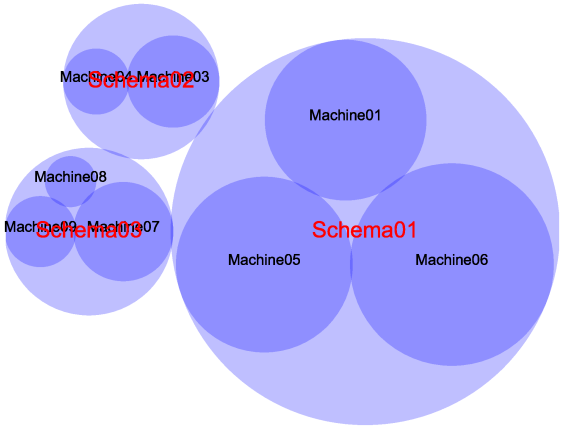
3.2.2 Visualização de resultados

Na área de representação de resultados, a técnica de *Circle Packing* (ALI et al., 2016) é o método mais apropriado para a representação de métricas por razão da sua sinergia com o método de modelação por módulos. Nessa técnica, cada módulo terá um círculo representativo da métrica sendo analisado, quanto maior o círculo maior será o valor da métrica daquele conjunto. Com isso, é possível analisar e comparar a performance de diferentes conjuntos naturalmente.

Na Figura 14 podemos ver um exemplo do uso de *Circle Packing*, no caso, temos um sistema com 3 módulos, denominados *Schemes*, sendo cada um deles um conjunto de máquinas. Como pode ser observado, é fácil observar a diferença de performance entre os diferentes módulos; mesmo que os módulos *Scheme01* e *Scheme02* tenham a mesma quantidade de máquinas, o *Scheme01* apresenta melhor performance, o que pode ser um indicativo que o escalonador está favorecendo um grupo em vez de outro, por exemplo.

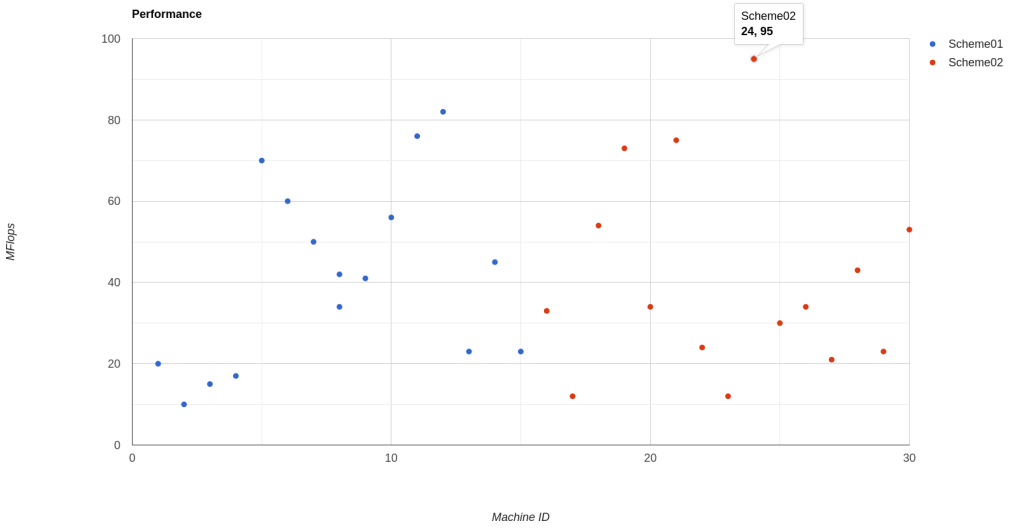
Contudo, para casos onde se deseja maior granularidade, o método abordado não é adequado, sendo ideal utilizar uma técnica onde seja possível analisar o modelo no nível de componentes unitários, como máquina e enlace. Para isso, foi implementado o *Scatter Plot*, que poderá mostrar a métrica de cada componente isoladamente, mas com capacidade de visualização melhor que o gráfico de barra, anteriormente utilizado. Como mostrado na Figura 15, o gráfico além de mostrar a diferença de performance entre máquinas, consegue relacionar cada máquina com seu módulo.

Figura 14 – Exemplo do uso de *Circle Packing* para comparar a performance de diferentes módulos



Fonte: Elaborado pelo autor.

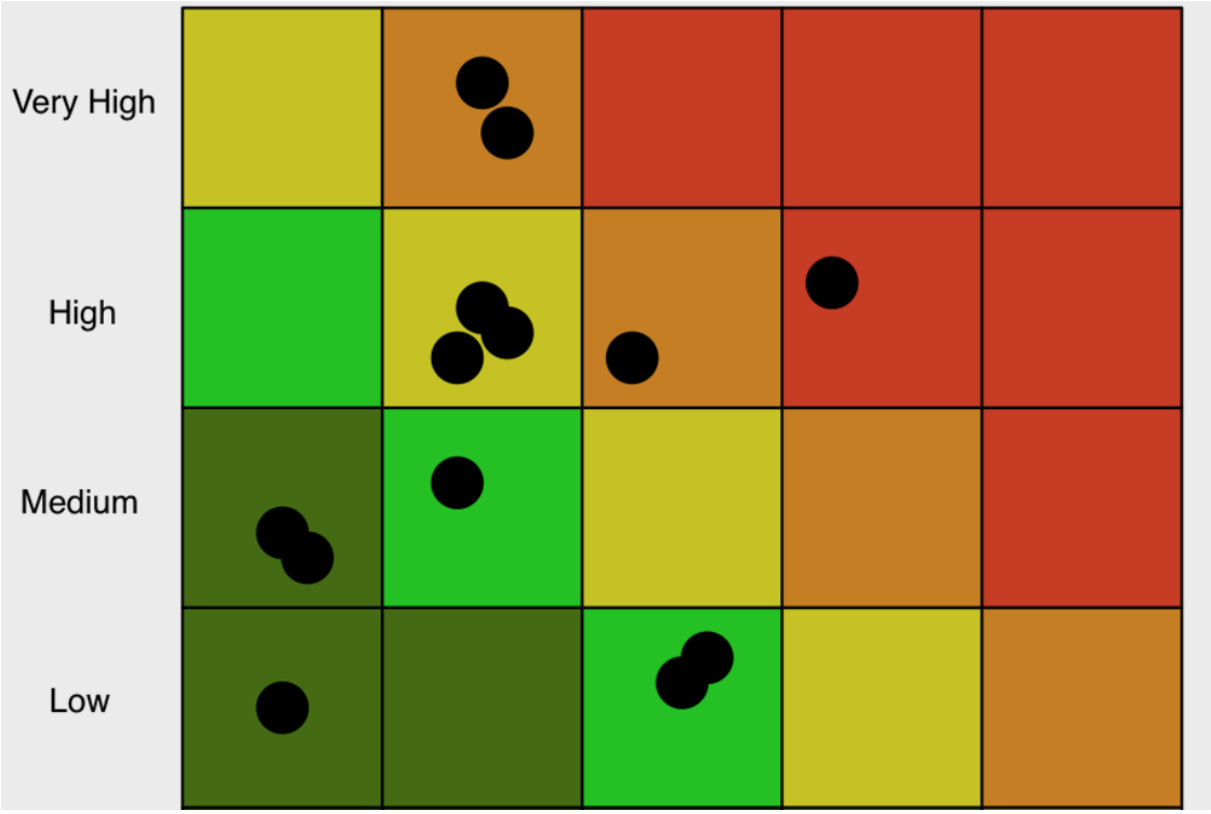
Figura 15 – Exemplo de um *Scatter Plot* representando a performance de máquinas.



Fonte: Elaborado pelo autor.

Para finalizar, nos casos onde há uma comparação entre métricas, como na representação de eficiência energética das máquinas, o método do *Heatmap*(SHNEIDERMAN, 2008). No exemplo da Figura 29, podemos ver que no eixo y temos a quantidade de *MFlops* que uma máquina efetuou, enquanto que no eixo X temos o gasto de energia da máquina, sendo as máquinas representadas por pontos. No exemplo, quanto mais próximo da cor vermelha for a região, pior será a eficiência energética, por contra partida, quanto mais verde, mais eficiente será a máquina.

Figura 16 – *Heatmap* para representar a relação entre gasto energético e *MFlops*.



Fonte: (MICROSOFT, s.d.)

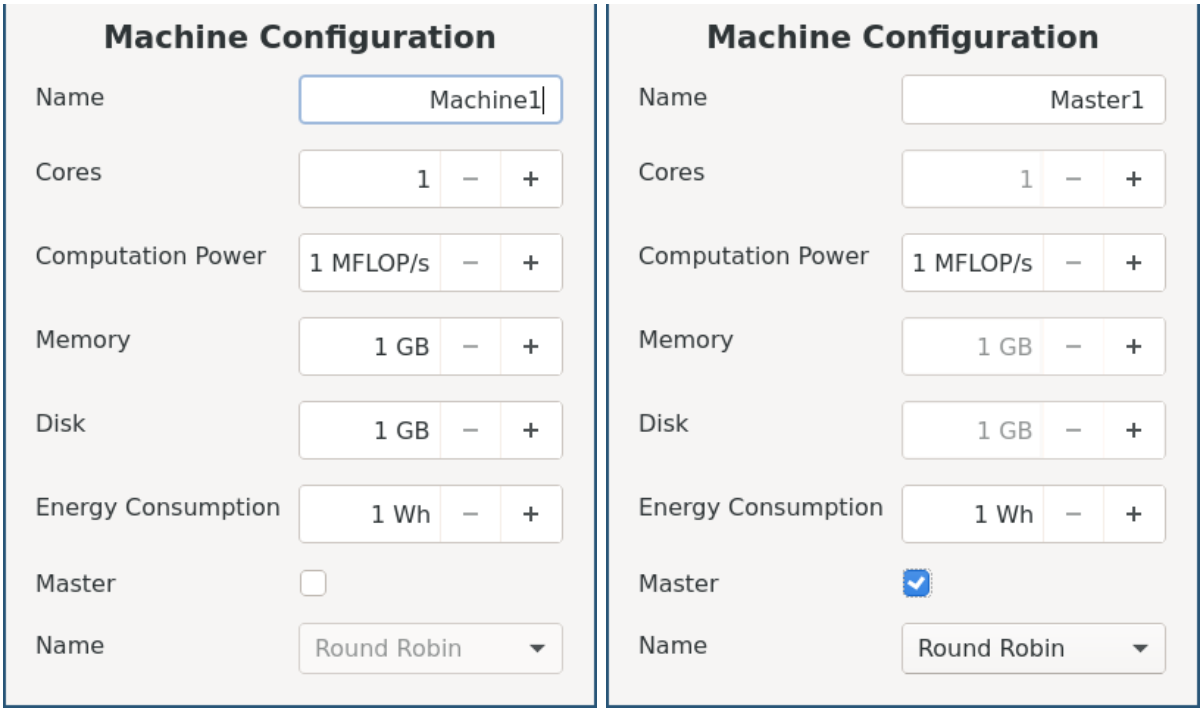
4 Resultados

Nesse capítulo será descrito os resultados das implementações, mostrando as mudanças ocorridas durante o processo de reengenharia do simulador, a capacidade dos métodos anteriormente abordados e a utilização deles em um caso concreto.

4.1 Reestruturação da interface gráfica

Com a finalidade de melhorar a usabilidade do programa, as telas de configuração foram alteradas para conter informações mais relevantes para o usuário. Um exemplo disso pode ser visto na Figura 17, que em contraste à janela original (Figura 3), a nova possui uma interface simples e com maior interatividade, sendo possível somente configurar atributos relacionados ao fato de que a máquina é um mestre ou não.

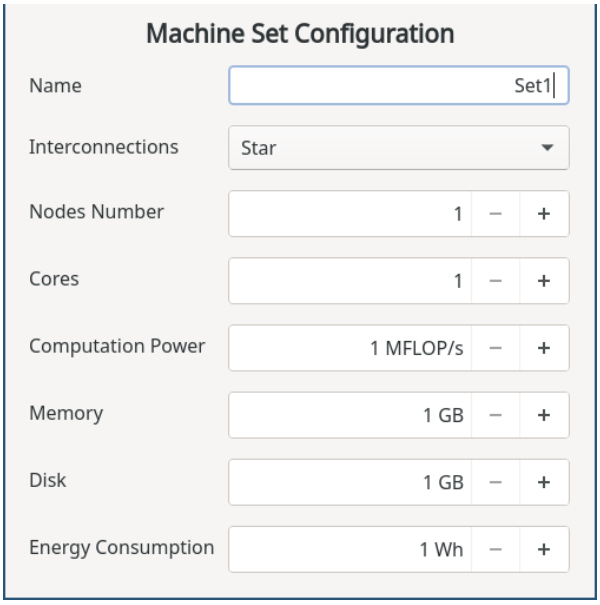
Figura 17 – Janela de configuração de nós e mestres.



Fonte: Elaborado pelo autor.

Para a configuração de um conjunto homogêneo de máquinas, foi criado a janela de configuração referente a Figura 18. Nessa janela, é possível atribuir quantas máquinas compõe esse conjunto, assim como o tipo de topologia que as maquinas irão seguir, sendo elas a topologia Dragonfly, em Série e Estrela.

Figura 18 – Janela de configuração de um conjunto homogêneo de máquinas.

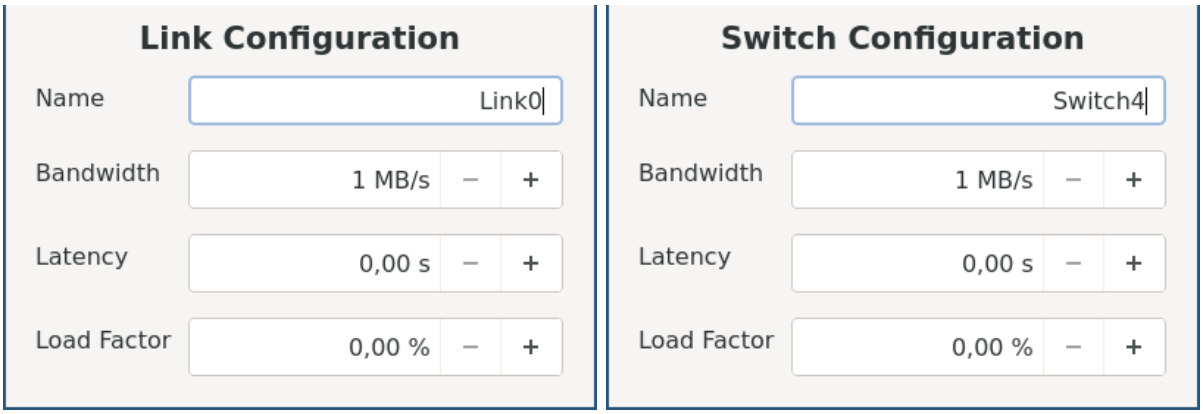


Fonte: Elaborado pelo autor.

Para auxiliar na criação de conexões mais complexas, o ícone de comutador foi criado. Com esse componente é possível modelar topologias como a de *Barramento*, *Fat-Tree* e *Dragonfly*; além disso, esse componente pode ser utilizado para facilitar na modelagem de sistemas com muitas interconexões entre módulos.

Por meio da Figura 19 podemos observar a janela de configuração de um comutador, a qual é idêntica a de um *Link*, sendo possível configurar em ambos os casos banda do componente, a latência em segundos que o componente demora para transportar um pacote, e por fim a porcentagem do tempo que esse componente ficará em utilização.

Figura 19 – Janelas de configuração de *links* e *switchs*



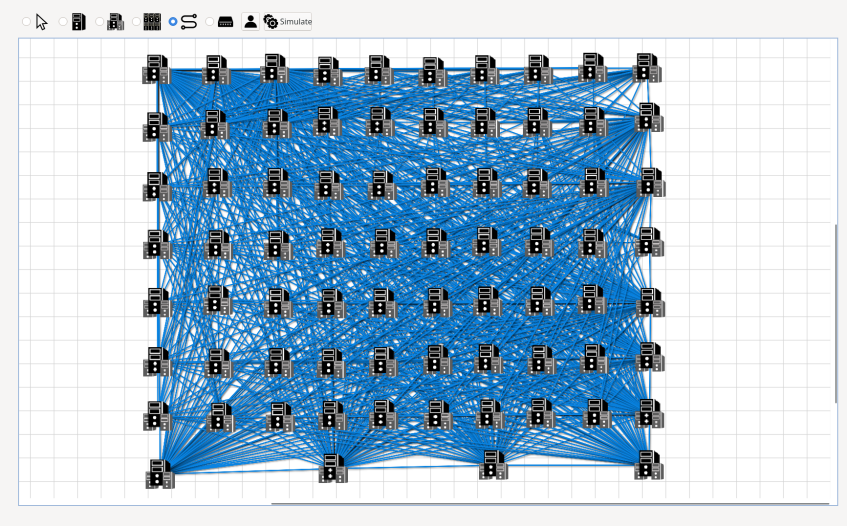
Fonte: Elaborado pelo autor.

4.2 Modelagem

Para a demonstração do método de módulos, foi efetuado a modelação de um sistema distribuído equivalente ao supercomputador *Frontier*, visando sua alta complexidade e tamanho. O sistema é composto de 74 *Racks*, cada um contendo 64 nós, totalizando 4736 máquinas.

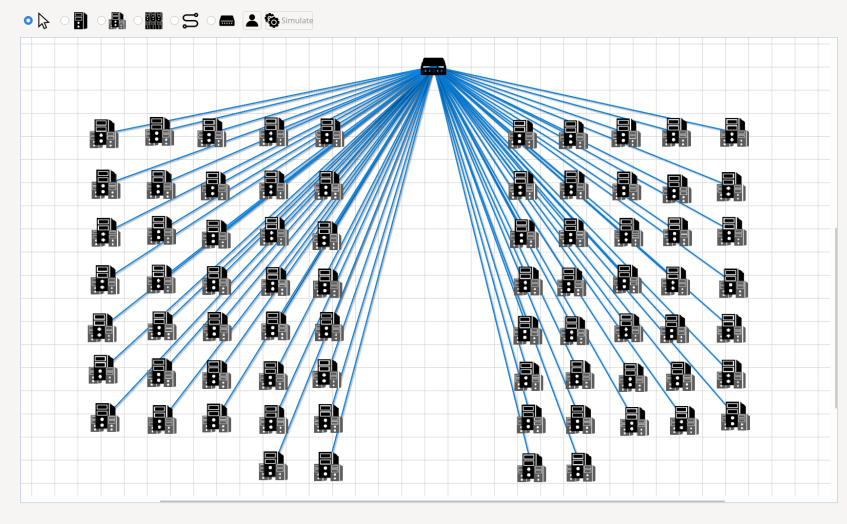
Apesar da capacidade de modelar sistemas de alta complexidade, como na Figura 20, pode-se modelar um sistema mais simples mas equivalente ao real, como visualizado na Figura 21, onde há a utilização de um comutador com *bandwidth* igual à soma dos *links* que irá substituir.

Figura 20 – *Racks* conectados de forma complexa e confusa, existindo 2701 enlaces.



Fonte: Elaborado pelo autor.

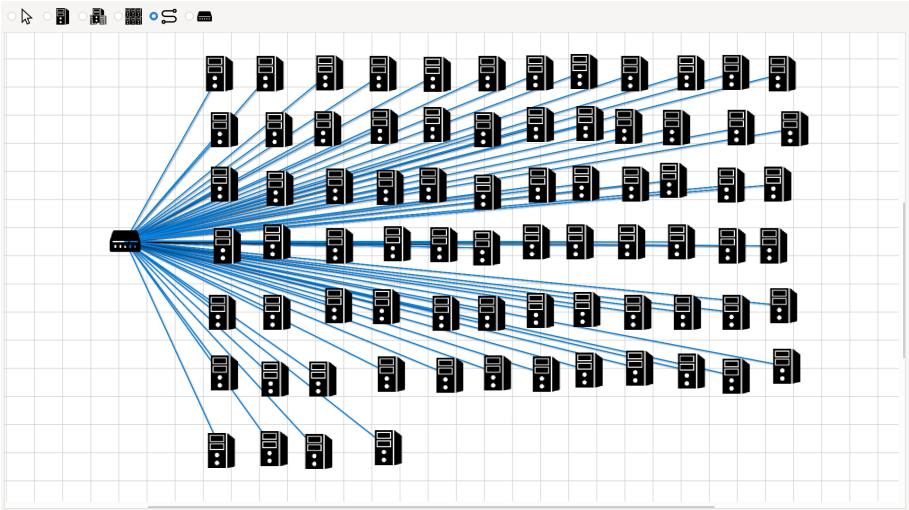
Figura 21 – *Racks* conectados de forma simples, utilizando métodos alternativos.



Fonte: Elaborado pelo autor.

Podemos visualizar na Figura 22 a configuração de um *Rack* formado por 64 máquinas, sendo construída dentro de um módulo, o qual é replicado 74 vezes, não sendo necessário configurar cada conjunto. Entretanto, a replicação pode se tornar um fardo quando há uma má configuração na máquina a ser copiada, o que pode tornar a modelagem mais difícil.

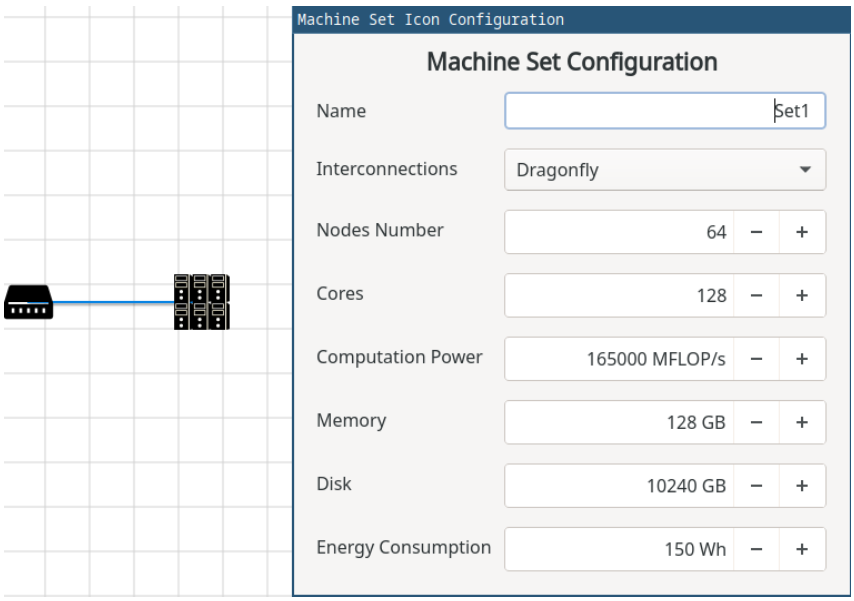
Figura 22 – Modelagem de um *Rack*, tendo 64 máquinas conectadas em um comutador de saída.



Fonte: Elaborado pelo autor.

Para auxiliar nesses casos, pode-se utilizar o método de sistemas homogêneos, que como demonstrado na Figura 23, resume a configuração de 64 máquinas em uma janela, fazendo com que possíveis mudanças sejam viabilizadas em tempo hábil.

Figura 23 – Modelagem de um *Rack* utilizando o método de conjuntos homogêneos.



Fonte: Elaborado pelo autor.

4.3 Representação de métricas

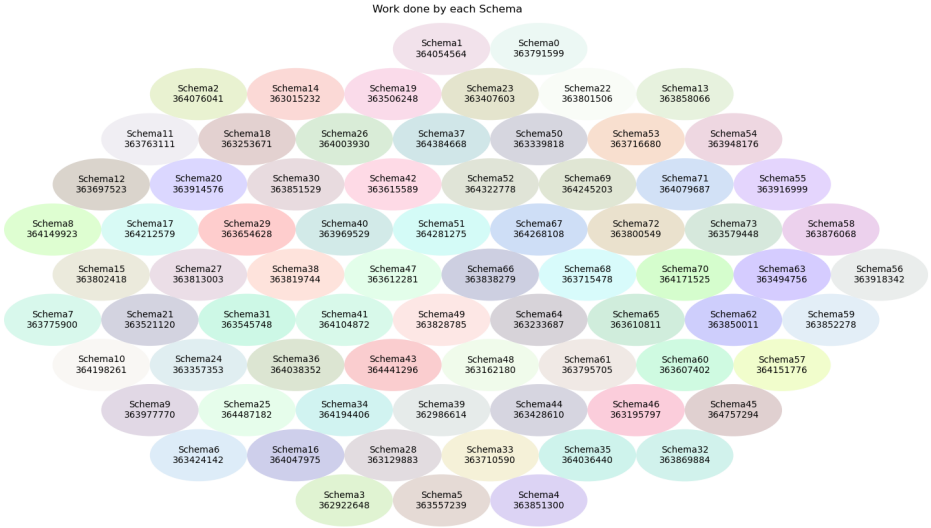
Para mostrar a utilização do método de *Circle Packing*, foi utilizado o mesmo sistema abordado na seção anterior, levando em conta uma carga de trabalho aleatória. Como pode ser visto na Figura 24, onde é analisado a quantidade de *MFLOPS* efetuado por cada módulo, é possível visualizar todo o sistema de forma simplificada, tendo uma percepção inicial de qual módulo obteve uma performance melhor ou pior.

Com o intuito de se obter uma análise melhor entre os diferentes módulos, podemos ver na Figura 25 um gráfico de dispersão referente ao mesmo sistema. Neste caso, o intuito do gráfico é analisar o valor real de cada métrica, sendo possível visualizá-lo a partir do canto inferior direito, onde há o valor de *MFLOPS* e *Id* de um módulo. Além disso, por meio da utilização de cores, torna-se possível analisar a diferença entre componentes de diversos módulos, o que é visível na Figura 26, onde é possível utilizar cores para a identificação de quais máquinas pertencem a quais módulos.

Para auxiliar na exploração dos dados, ações como zoom e a movimentação interativa pelo gráfico foram implementados, conforme visualizado na Figura 27 e 28. Com essas implementações, a visualização de um número alto de nós se torna elementar.

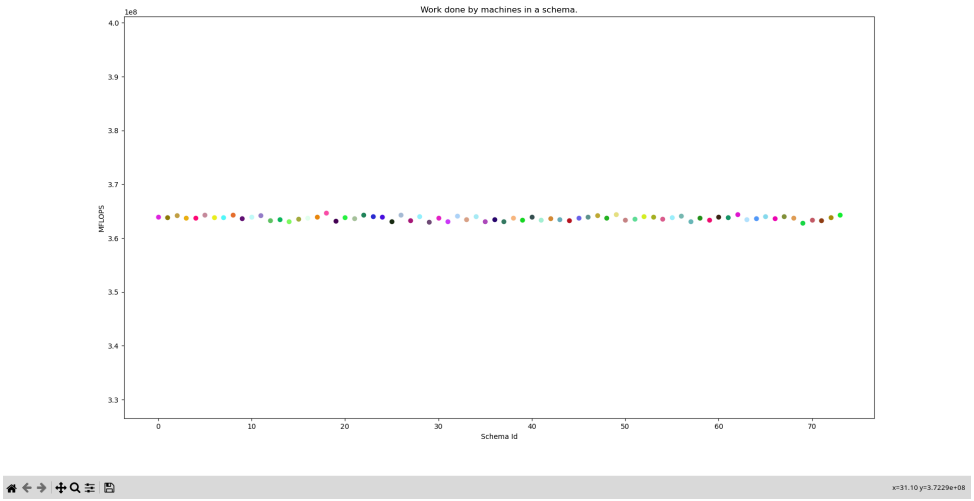
Por meio da utilização de um gráfico de calor é possível obter uma comparação de eficiência energética entre os diferentes módulos. Como exemplificado pela figuras 29 e 30, quanto maior o trabalho efetuado pelo módulo e menor o gasto energético, maior a eficiência dele, o que é indicado pela cor esverdeada; no caso contrário, o módulo é considerado mais ineficiente do que os outros, sendo representado então pela cor vermelha.

Figura 24 – Exemplo do uso do método de *Circle Packing* para a comparação de performance entre diferentes módulos.



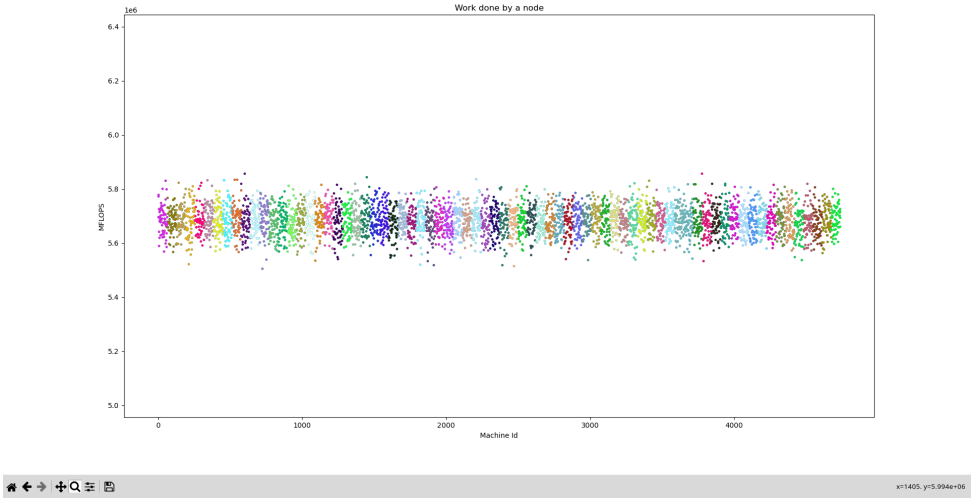
Fonte: Elaborado pelo autor.

Figura 25 – *Scatter Plot* representando o trabalho efetuado por cada módulo.



Fonte: Elaborado pelo autor.

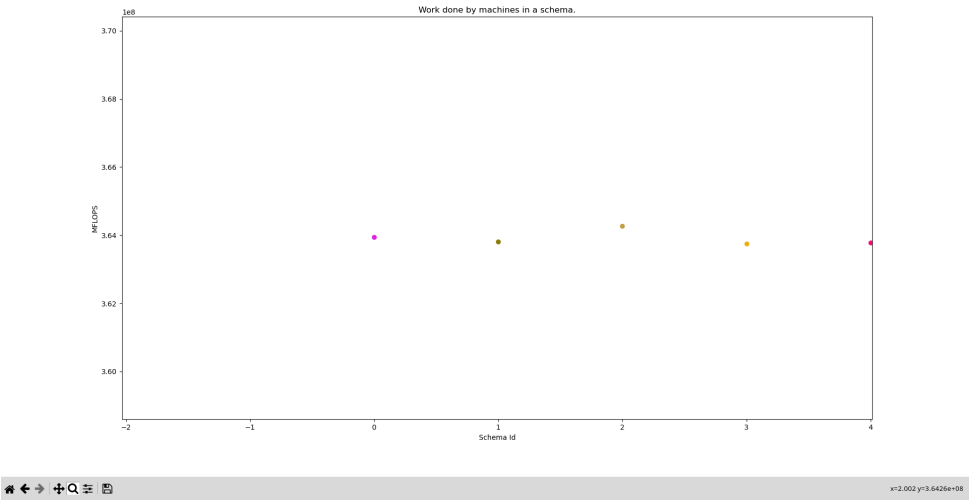
Figura 26 – *Scatter Plot* representando o trabalho efetuado por cada máquina.



Fonte: Elaborado pelo autor.

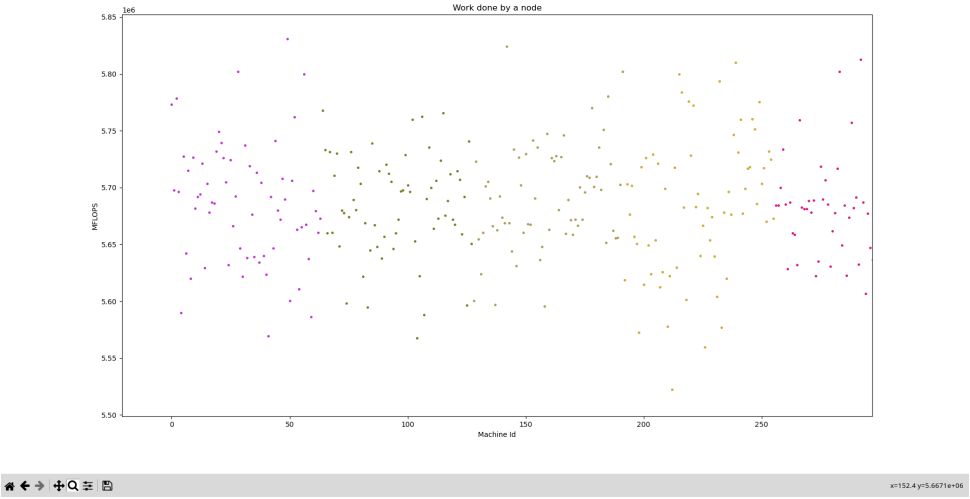
Apesar da implementação dos novos métodos, aspectos do projeto original, como a geração de uma tabela geral, foram mantidas nesta nova implementação, pois ainda há a necessidade de que valores absolutos sejam visualizados pelo usuário. Como mostrado pela Figura 31, com a nova reestruturação do código é possível obter dados de módulos inteiros, além de componentes básicos, como *links* e máquinas e comutadores.

Figura 27 – *Zoom* sendo utilizado para uma melhor análise entre diferentes módulos.



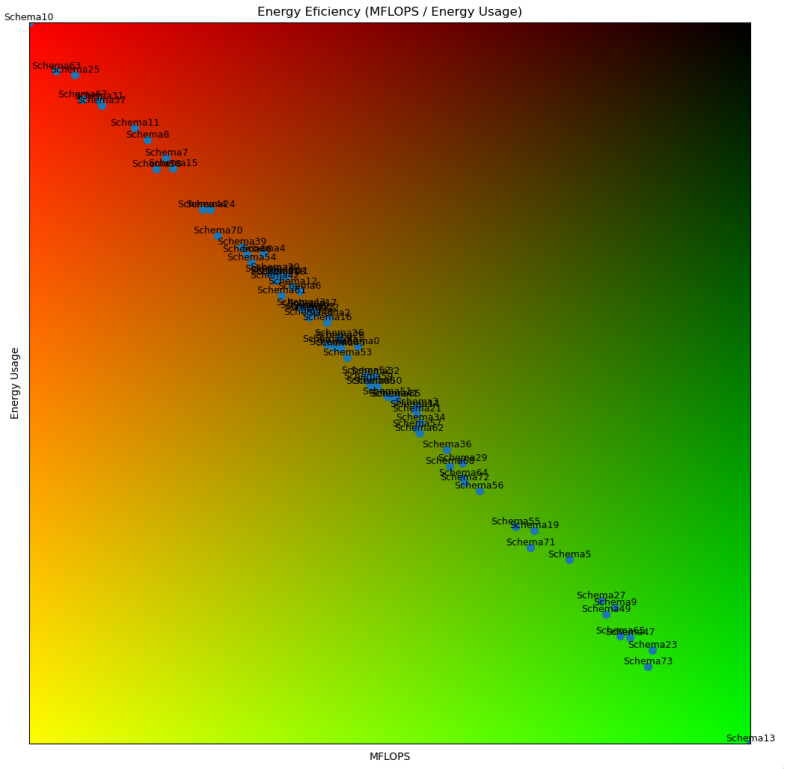
Fonte: Elaborado pelo autor.

Figura 28 – *Zoom* sendo utilizado para uma melhor análise entre diferentes máquinas.



Fonte: Elaborado pelo autor.

Figura 29 – Comparação de eficiência energética entre os diferentes módulos.



Fonte: Elaborado pelo autor.

Figura 30 – *Zoom* sendo utilizado para uma melhor análise da eficiência energética de diferentes módulos.

Fonte: Elaborado pelo autor.

Figura 31 – Utilização de tabelas para a visualização de valores absolutos.

	Name	MFLOPS	Energy Usage	Second at Usage
1	Schema0	363340371	330309	34
2	Schema1	363476276	330433	36
3	Schema2	363122627	330113	37
4	Schema3	364043624	330951	38
5	Schema4	364087814	330992	39
6	Schema5	364266172	331156	40
7	Schema6	363462155	330426	41
8	Schema7	363484787	330447	42
9	Schema8	363453824	330420	43
10	Schema9	363224334	330212	43
11	Schema10	363722622	330666	45

Fonte: Elaborado pelo autor.

5 Conclusão

5.1 Conclusões

Tendo em vista o que foi explicado nos primeiros capítulos, podemos entender a relevância da simulação de sistemas de larga escala. A ferramenta *iSPD* era capaz de modelar sistemas de baixa escala, sendo então o propósito deste projeto a adaptação do simulador com o intuito de torná-lo capaz de trabalhar com maiores projetos.

Para a adequação dos novos métodos, uma reestruturação do projeto inteiro foi necessário. Com isso, um trabalho de reengenharia foi feito, igualando o novo projeto ao anterior.

O projeto teve sucesso em cumprir os objetivos propostos. O simulador está agora em um estado de funcionalidade superior, apresentando interfaces adequadas para o uso de usuários de todo nível. Esta versão atende a alunos e professores que a irão utilizar, sendo capaz também de ter utilidade profissional.

Foi fundamental a execução deste projeto pois promove uma nova gama de possibilidades de uso para o projeto *iSPD*, o adequando para as tecnologias atuais. Esta nova versão teve o seu desenvolvimento focado em tornar o simulador funcional e intuitivo, visando assim facilitar sua utilização por seus usuários.

5.2 Trabalhos futuros

Para um maior aperfeiçoamento da ferramenta, um trabalho futuro pode ser o desenvolvimento na adição de *GPUs* na configuração de máquinas, tendo em vista a alta utilização desses componentes nos sistemas de larga escala.

O *iSPD* possuiu um base de código reestruturada, com um nível de abstração alto, permitindo a modificação de partes fundamentais sem demasiado trabalho. Além disso, o trabalho feito na modularizadão de seu código permite que novos módulos sejam construídos facilmente.

Referências

ALI, Syed Mohd et al. Big data visualization: Tools and challenges. In: 2016 2nd International Conference on Contemporary Computing and Informatics (IC3I). [S.l.: s.n.], 2016. P. 656–660. DOI: 10.1109/IC3I.2016.7918044.

BLANCHETTE, J.; SUMMERFIELD, M. **C++ GUI Programming with Qt4**. [S.l.]: Pearson Education, 2008. ISBN 9780132703000.

BUYYA, Rajkumar; MURSHED, Manzur. GridSim: A toolkit for the modeling and simulation of Distributed Resource Management and Scheduling for Grid Computing. **Concurrency and Computation: Practice and Experience**, v. 14, 13–15, p. 1175–1220, 2002. DOI: 10.1002/cpe.710.

CASANOVA, H. Simgrid: a toolkit for the simulation of application scheduling. In: PROCEEDINGS First IEEE/ACM International Symposium on Cluster Computing and the Grid. [S.l.: s.n.], 2001. P. 430–437. DOI: 10.1109/CCGRID.2001.923223.

GARCIA, Marco Antonio Barros Alves. **Motor de simulação e geração de métricas para avaliação de desempenho para o simulador de grades computacionais iSPD**. por. São José do Rio Preto: [s.n.], 2010.

GUERRA, Aldo Ianelo. **Plataforma de simulação de grades computacionais : interface icônica e interpretador de modelos externos**. por. São José do Rio Preto: [s.n.], 2010.

GUOZHENG, Yang; YULIANG, Lu; HUIXIAN, Chen. A New Network Topology Visualization Algorithm. In: 2011 First International Conference on Instrumentation, Measurement, Computer, Communication and Control. [S.l.: s.n.], 2011. P. 369–372. DOI: 10.1109/IMCCC.2011.99.

JAIN, R. **The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling**. [S.l.]: Wiley, 1991. ISBN 9780471503361.

KREKHOV, Andrey et al. Towards in Situ Visualization of Extreme-Scale, Agent-Based, Worldwide Disease-Spreading Simulations. In: SIGGRAPH Asia 2015 Visualization in High Performance Computing. Kobe, Japan: Association for Computing Machinery, 2015. (SA '15). ISBN 9781450339292. DOI: 10.1145/2818517.2818543.

MICROSOFT. **Usage of Heatmap**. [S.l.: s.n.]. https://community.fabric.microsoft.com/t5/Desktop/Matrix-heatmap-fixed-background-color/td-p/2740970.

PAULO, Guilherme Pasqualato de. **Simulação de falhas iSPD**. por. São José do Rio Preto: [s.n.], 2017.

PRABHAKARAN, Akhila; J., Lakshmi. Cost-Benefit Analysis of Public Clouds for Offloading In-House HPC Jobs. In: 2018 IEEE 11th International Conference on Cloud Computing (CLOUD). [S.l.: s.n.], 2018. P. 57–64. DOI: 10.1109/CLOUD.2018.00015.

SHNEIDERMAN, Ben. Extreme Visualization: Squeezing a Billion Records into a Million Pixels. In: PROCEEDINGS of the 2008 ACM SIGMOD International Conference on Management of Data. Vancouver, Canada: Association for Computing Machinery, 2008. (SIGMOD '08), p. 3–12. ISBN 9781605581026. DOI: 10.1145/1376616.1376618.

SILVA, Diogo Tavares da. **Implementação de um banco de Traces de Cargas de Trabalho para o iSPD**. por. São José do Rio Preto: [s.n.], 2012.

TAMASHIRO, Camila Baleiro Okado. **Modelagem de falhas em nuvem**. por. São José do Rio Preto: Dissertação (mestrado) - Universidade Estadual Paulista "Júlio de Mesquita Filho", Instituto de Biociências Letras e Ciências Exatas, São José do Rio Preto, 2020.

TSAI, Yin-Te. An Overview of Machine Learning and HPC in Open Sources for Bioinformatics. In: 2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM). [S.l.: s.n.], 2018. P. 1338–1342. DOI: 10.1109/BIBM.2018.8621078.

WYBORN, Lesley; EVANS, Benjamin J. K. Integrating 'Big' geoscience data into the petascale national environmental research interoperability platform (NERDIP): Successes and unforeseen challenges. In: 2015 IEEE International Conference on Big Data (Big Data). [S.l.: s.n.], 2015. P. 2005–2009. DOI: 10.1109/BigData.2015.7363981.