Despachador de Simulações Tolerante a Falhas

Matheus M. Sarmento, Lucas R. Costa, Kaio A. Silva, André C. Drummond

[1]Departamento de Ciência de Computação – Universidade de Brasília(UnB)
Brasília, DF, 70910-900 – Brasil

matheus.m.sarmento@gmail.com, lucasrc.rodri@gmail.com,
kaio.silva@ifro.edu.br, andred@unb.br

Abstract. *Scientific applications often require a lot of computational resources to achieve their results. Understanding and proving scientific results implies the performance of appropriate statistical analyzes. For this, researchers commonly need to perform several simulations by correctly analyzing the differences obtained in each replication. In some scenarios, this makes the conduct of research an impractical process. The automation of this process implies saving the time spent both in configuration and in possible human errors in data analysis. As a way to overcome such challenges, this work proposes an automation system for the dispatch of simulations with statistical analysis of the results. In addition, the proposed system provides distributed computing resources and assists the organization of tasks in a distributed system, with load balancing and fault tolerance. Examples of running the dispatcher are presented through an intuitive interface for system management and the display and export of results.*

Resumo. *Aplicações científicas podem exigir bastante recurso computacional para que os resultados sejam alcançados. Compreender e comprovar resultados científicos implicam na realização de análises estatísticas apropriadas. Para isso, pesquisadores normalmente necessitam realizar diversas simulações, analisando corretamente as diferenças obtidas em cada replicação. Em alguns cenários, esta situação torna a realização da pesquisa um processo oneroso. A automação deste processo implica na economia do tempo gasto tanto em configuração, quanto em possíveis erros humanos na análise dos dados. Como forma de superar tais desafios, este trabalho propõe um sistema de automação para o despacho de simulações com análises estatísticas dos dados. Além disso, o sistema proposto disponibiliza os recursos computacionais de forma distribuída e auxilia a organização de tarefas em um sistema distribuído, com o balanceamento de carga e com tolerância a falhas. São apresentados exemplos de execução do despachador através de uma interface intuitiva para gerenciamento do sistema e a exibição e exportação dos resultados.*

1. Introdução

Muitas pesquisas acadêmicas são fundamentadas sob o aspecto da simulação, por permitir a repetição dos experimentos a um baixo custo, sem englobar a totalidade e a complexidade de um sistema real. Entretanto, realizar uma análise estatística apropriada através da simulação implica fundamentalmente na representatividade e na quantidade das amostras consideradas. Para alcançar tais resultados, pesquisadores frequentemente necessitam realizar diversas simulações analisando corretamente as diferenças obtidas em cada

replicação. Todavia, em alguns cenários, esta situação torna a realização de uma pesquisa científica um processo oneroso. A configuração e a utilização de simuladores normalmente é um procedimento delicado e vulnerável a erros, além de poder exigir bastante recurso computacional para a obtenção de resultados corretos. Automatizar este processo implica na economia do tempo gasto tanto em configuração, quanto em possíveis erros humanos na análise dos dados.

Tendo em vista estes desafios, este trabalho propõe um sistema de automação para o despacho de simulações com análises estatísticas dos resultados. O sistema proposto utiliza o simulador utilizado pelo pesquisador como uma caixa preta, disponibilizando os recursos computacionais de modo distribuído, auxiliando na configuração e organização das tarefas do simulador através de um balanceamento de carga entre os recursos computacionais disponíveis. O sistema ainda possui um modelo com tolerância a falhas entre os equipamentos utilizados pela simulação. As funcionalidades projetadas são apresentadas por meio de uma *interface* intuitiva para o gerenciamento, exibição e a exportação dos resultados das simulações, permitindo uma execução fácil e ágil da pesquisa.

2. Sistema de despachamento

Baseado no paradigma mestre-escravo, o sistema consiste em dois elementos principais: o **despachador** e as máquinas **trabalhadoras**. Os usuários interagem via *interface WEB*, enviando comandos para o sistema despachador. O despachador gerencia as tarefas para as máquinas trabalhadoras e recupera seus resultados, persistindo-os em uma base de dados. A Figura 1 apresenta uma visão geral da arquitetura do sistema de despachamento.

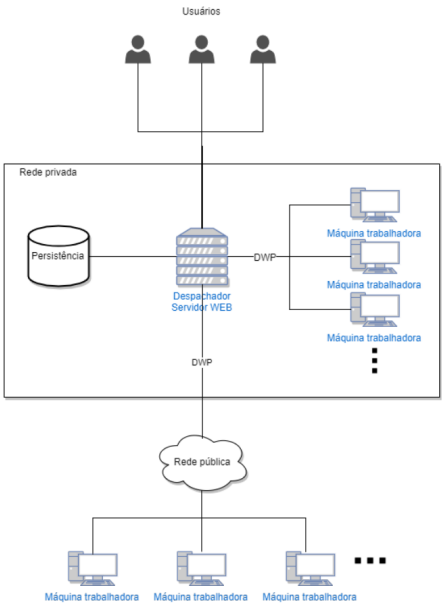

Figura 1. Visão geral da arquitetura do sistema.

2.1. Despachador

O despachador é um serviço mestre encarregado de organizar e expedir tarefas de um simulador para máquinas trabalhadoras. É um serviço leve e de fácil implantação, implementado com tecnologias recentes, sendo elas: NodeJS [NodeJS 2018], interpretador

JavaScript de alta performance para servidores e MongoDB [MongoDB 2018], aplicação que realiza a persistência dos dados de maneira altamente escalável e ágil. O serviço atende aos requisitos de alta escalabilidade, alta disponibilidade e tolerancia a falhas adversas em um sistema distribuído.

Juntamente ao despachador, uma *interface WEB* também é provida, facilitando a interação do usuário com o sistema (Figura 2). A *interface* encarrega-se em receber as entradas das simulações do usuário, exportar resultados, e também exibe informações como estado dos grupos de simulações, gráficos gerados em tempo real, *logs* do sistema, máquinas trabalhadoras ativas, configurações do despachador, entre outros. A *interface*, dessa forma, otimiza o tempo gasto com a configuração do cenário de teste, onde tipicamente seria despendido um tempo considerável em tarefas que não agregariam qualquer tipo de valor ao resultado final da pesquisa realizada.

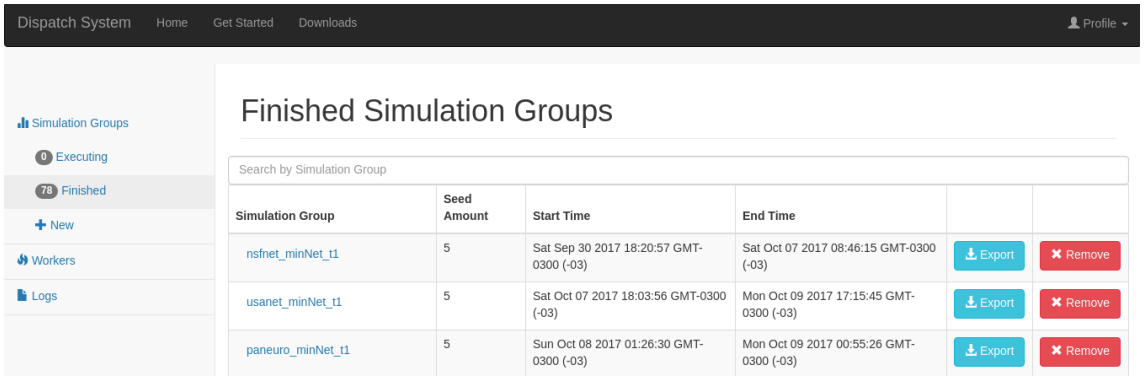

Figura 2. Interface WEB do despachador.

Um importante aspecto do sistema despachador é que o simulador utilizado é visto como uma caixa preta, ou seja, não é do escopo do sistema despachador o conhecimento de qualquer forma de implementação do simulador, tendo interesse apenas nas suas diretivas de saída que serão melhor explicadas na Seção 2.1.5. Dessa maneira, deixa-se flexível possíveis alterações no simulador, possibilitando até mesmo a alteração da linguagem de programação com o qual foi programado.

Para isso, a interface WEB provê um mecanismo genérico de aceitação de tarefas a serem executadas. Primeiramente, deve-se disponibilizar um executável do simulador (arquivos interpretados ou compilados) onde o sistema fará o reconhecimento do formato do arquivo e saberá como executá-lo.

O usuário também deve informar um modelo dos argumentos que resultam na menor tarefa de seu simulador. Nesse ponto, a interface disponibiliza algumas diretivas interpretativas onde o usuário as posiciona em argumentos variáveis do comando. Essas diretivas geram dinamicamente novos componentes na interface de usuário que aceitam uma faixa de valores, que serão percorridos, e cada valor será substituído em sua devida posição no modelo de linha de comando. Dessa forma, em suma, são geradas múltiplas linhas de comando com argumentos distintos. Para cada uma das diretivas interpretativas, o usuário poderá decidir a ordem de precedência com a qual deverão ser interpretadas na hora da geração dos comandos. As diretivas interpretativas são as seguintes: (*i*) **Numéricos**; (*ii*) **Cadeias de caracteres**; e (*iii*) **Arquivos**.

2.1.1. Numéricos

As diretivas interpretativas numéricas são utilizadas ao substituir o argumento pelos caracteres **%d**. Possuem dois tipos de notação: notação de laço e notação de valores rígidos.

A notação de laço é definida da seguinte forma (valores divididos por ponto e vírgula):

valor-inicial;valor-final;incremento

onde **valor-inicial** define um valor numérico o qual o laço deverá iniciar, e incrementa-se do valor numérico **incremento** até atingir **valor-final**.

A notação de valores rígidos aceita a seguinte expressão regular (valores separados por vírgulas):

valor1,valor2,valor3,...

utilizando esse critério de notação serão utilizados somente os valores denotados na hora da substituição.

2.1.2. Cadeias de caracteres (*Strings*)

As diretivas interpretativas de cadeias de caracteres são utilizadas ao substituir o argumento pelos caracteres **%s**. A notação utilizada é a de valores separados por vírgulas, onde cada cadeia de caracteres é colocada entre aspas duplas:

“valor1”,“valor2”,“valor3”, ...

2.1.3. Arquivos

As diretivas interpretativas de arquivos são utilizadas ao substituir o argumento pelos caracteres **%f**. Esse tipo de diretiva disponibiliza uma entrada para múltiplos arquivos, onde os argumentos serão substituídos por seus respectivos nomes.

2.1.4. Exemplo de uso

Como forma ilustrativa da utilização das diretivas, suponha que um simulador em java (exemplo.jar) tenha os seguintes argumentos: **–arg0 [numérico]**, **–arg1 [*strings*]**, **–arg2 [arquivos]**, **–verbose**. O modelo de argumentos na linha de comando proposto seria:

–arg0 %n –arg1 %s –arg2 %f –verbose

Se, por exemplo, o usuário optar pelos seguintes valores para os argumentos: **arg0: 1,2** com precedência 1, **arg1: “teste1”,“teste2”** com precedência 2, e dois arquivos **arq1.txt**, **arq2.txt** com precedência 3, o resultado final iria gerar os seguintes comandos que serão enviados para as máquinas trabalhadoras com seus respectivos executáveis e arquivos:

java -jar exemplo.jar –arg0 1 –arg1 teste1 –arg2 arq1.txt –verbose
java -jar exemplo.jar –arg0 1 –arg1 teste1 –arg2 arq2.txt –verbose
java -jar exemplo.jar –arg0 1 –arg1 teste2 –arg2 arq1.txt –verbose
java -jar exemplo.jar –arg0 1 –arg1 teste2 –arg2 arq2.txt –verbose
java -jar exemplo.jar –arg0 2 –arg1 teste1 –arg2 arq1.txt –verbose
java -jar exemplo.jar –arg0 2 –arg1 teste1 –arg2 arq2.txt –verbose
java -jar exemplo.jar –arg0 2 –arg1 teste2 –arg2 arq1.txt –verbose
java -jar exemplo.jar –arg0 2 –arg1 teste2 –arg2 arq2.txt –verbose

2.1.5. Diretiva de saída

A diretiva de saída do executável (simulador) deve ser processada à saída de fluxo padrão (*standard output stream*) em um formato JSON conforme a Figura 3[1].


```
{
    "Load": 60,
    "BBR": 133.71058620912592,
    "Seed": 2,
    "Called Blocked by Cos": 68.70385798268221,
    "BP-2": 84.01288924501667,
    "BP-1": 105.14770303338827,
    "BP-0": 101.46514596118337,
    "LPS": 11086
}
```


Figura 3. Exemplo de dados exportados pelo simulador após a execução de uma simulação.

2.2. Trabalhador

O trabalhador é um serviço encarregado de executar as tarefas expedidas pela máquina despachadora. As máquinas trabalhadoras atuam como escravas no paradigma mestre/escravo e possuem a tarefa de executar simulações designadas pelo despachador.

Para se conectar ao despachador, existem dois mecanismos implementados: (*i*) Descoberta dinâmica; e (*ii*) Conexão direta. A descoberta dinâmica consiste em situações em que o pesquisador deseja montar um sistema de despacho em uma rede local. Para isso, as máquinas trabalhadoras são capazes de reconhecer e conectar-se automaticamente ao despachador, eliminando qualquer configuração manual da aplicação. A máquina trabalhadora submete uma mensagem simples sobre o protocolo UDP, fazendo um broadcast na porta 16180 para todos os dispositivos da rede local, tendo como objetivo notificar o despachador. Enquanto uma mensagem de resposta do despachador não é recebida, periodicamente a cada um segundo, a máquina repete o procedimento. Uma vez que a mensagem é recebida, obtém-se o endereço do despachador na rede local e então estabelece-se uma conexão *TCP*.

No caso da conexão direta, é necessário que o usuário defina no arquivo de configuração o endereço do despachador alvo. A aplicação realizará a leitura do arquivo e tentará realizar diretamente uma conexão *TCP* com o endereço configurado. Caso o endereço do despachador esteja configurado, o sistema realiza a tentativa de conexão direta. Em caso de falha, é feita a alternância para o procedimento de descoberta automática.

[1]Os elementos apresentados na Figura 3 são um exemplo de saída. Pode-se criar ou remover elementos de acordo com o objetivo do simulador.

2.3. Protocolo DWP

Para realizar a comunicação entre o despachador e as máquinas trabalhadoras foi desenvolvido um protocolo da camada de aplicação denominado *Dispatcher-Worker Protocol* (DWP). O protocolo *DWP* cria um *socket TCP* para o estabelecimento de chamadas entre as máquinas trabalhadoras e a máquina de despacho.

Configurado periodicamente por um usuário administrador do sistema, o despachador requisita a disponibilidade de recursos das máquinas que estão conectadas à rede. Esses recursos são tipicamente os utilizados pelo simulador para processamento de dados da memória e do processamento da *CPU*. Isso se faz necessário para determinar a disponibilidade de cada máquina ativa à aceitação de uma nova execução de simulação, com base em outros parâmetros configuráveis do despachador que delimitam o quanto de recursos da máquina devem remanescer disponíveis.

Além disso, existe outro parâmetro configurável que determina o período de expedição de simulações em lotes. O despachador regularmente distribui simulações pendentes para máquinas que estejam à disposição de seus recursos. O mecanismo de distribuição das simulações é baseado no algoritmo de escalonamento por prioridades, as classificações variam de mais prioritária para menos prioritária, são elas: **urgente**, **alta**, **normal**, **baixa** e **mínima**. Simulações com mesma prioridade são executadas seguindo o princípio *first come, first served*.

Uma vez que o processamento é realizado pelas máquinas trabalhadoras e são devidamente retornados ao despachador, os resultados são organizados e persistidos na base de dados, em caso de sucesso. Em caso de erro, procedimentos de tolerância a falhas são realizados. Dessa forma, garante-se que apenas resultados corretos serão armazenados e apresentados ao usuário no seu término.

2.4. Tolerância a falhas

O sistema garante a consistência da simulação através da atribuição de estados à cada simulação, os estados são: **pendente de execução**, **em execução**, **finalizada** ou **inválida**. Uma vez que determinada simulação está sendo executada, associa-se à ela o identificador da máquina trabalhadora que a executa. Caso a mesma máquina deixe a rede de trabalhadores, o despachador desassocia todas suas simulações, e retorna os respectivos estados das simulações para pendência de execução. Na ocorrência de erros durante a execução de uma simulação, o despachador incrementa um contador de sanidade da simulação e atribui a ela o estado de pendência, para que seja executada em uma nova rodada de expedição em lotes. Caso o contador de sanidade atinja o valor cinco, a simulação é sinalizada com o estado de "Inválida", registrando-se um *log* de erro para o usuário responsável. Dessa forma, erros transientes são tolerados e erros contínuos são descartados. Somente com o resultado efetivo, a simulação é sinalizada como finalizada.

3. Demonstração para o salão de ferramentas

Para demonstrar a utilização do sistema proposto é preterivelmente necessário uma máquina *Intel Core 2Quad* de 2.66 *GHz* com 4 *GB* de *RAM* com distribuição *Debian* 8 de preferência. Nesta máquina serão feitas a execução do simulador ONS [Costa et al. 2016] como exemplo para utilização do despacho de trabalhos. Esta mesma máquina se comportará como mestre (*Despachador*) e escravo (*Trabalhador*). Possivelmente, pode-se

adicionar novas máquinas para atribuição do papel de trabalhador, basta que as máquinas estejam na mesma rede do *Despachador* e não haja bloqueios na porta UDP 16180.

Quanto aos detalhes de instalação do sistema despachante, orientações serão passadas sobre o seu funcionamento. Algumas configurações precisam ser ajustadas em cada máquina, despachador e trabalhadores, como a instalação do nodeJS, mongoDB e configuração dos arquivos no sistema. Outro ponto importante é o acompanhamento dos grupos de simulação e análise estatística da simulação em tempo real bem como a caracterização de métricas dinâmicas durante o experimento.

O projeto possui código aberto e está disponível nos endereços eletrônicos:

Despachador: `https://github.com/comnetunb/web_dispatcher`
Trabalhador: `https://github.com/comnetunb/worker`
Protocolo DWP: `https://github.com/comnetunb/protocol`

Os projetos estão separados em repositórios diferentes para evidenciar a separação lógica das aplicações e permitir mais facilmente uma eventual mudança de linguagem de programação, se necessário, desde que o protocolo de comunicação seja respeitado. Além disso, apresenta-se um vídeo explicativo sobre o sistema de despacho proposto no endereço eletrônico: `https://youtu.be/xrKwYhuAhdA`.

4. Exemplo de execução e resultados

Com o objetivo de validar a ferramenta proposta foram realizados testes de verificação para três grupos de simulação do simulador ONS [Costa et al. 2016]. Cada grupo contava com cerca de dez simulações, 5 sementes, e 10 pontos de carga cada, gerando um total de mais de 3000 instâncias de simulação que consumiam em média 30 minutos de processamento (utilizando em média 30% de processamento e 40% de memória por processo) em uma máquina trabalhadora.

Caso o pesquisador fosse executar as simulações citadas em apenas uma máquina, esta levaria cerca de 63 dias para a obtenção dos resultados. Se o pesquisador tivesse a disponibilidade de realizar a simulação em 10 máquinas, estas levaria cerca de 6 dias para terminar a execução, sem levar em conta o tempo desperdiçado para a configuração e a manutenção de possíveis erros na simulação.

Nos testes realizados com o sistema de despacho proposto, foram utilizadas aproximadamente 100 máquinas trabalhadoras. Nas máquinas utilizadas havia uma constante intermitência de conectividade (provocada propositadamente). Todas as 3000 instâncias foram terminadas em pouco mais de 8 horas de atividade do sistema onde não se fez necessária qualquer intervenção manual do usuário para auxiliar com a continuidade dos processos. Uma combinação dos resultados pode ser vista na Figura 4.

Foram realizadas validações parciais dos resultados através de comparações com a execução do simulador ONS de forma manual. Fatores como tolerância a falhas são parâmetros difíceis de se mensurar. Para tal, foram avaliados os registros do sistema durante a execução do grupo de simulações no cenário supracitado. Durante toda sua execução, mais de 100 desconexões e reconexões de máquinas foram evidenciadas. No entanto, sem qualquer tipo de intervenção manual, o sistema conseguiu manter o estado

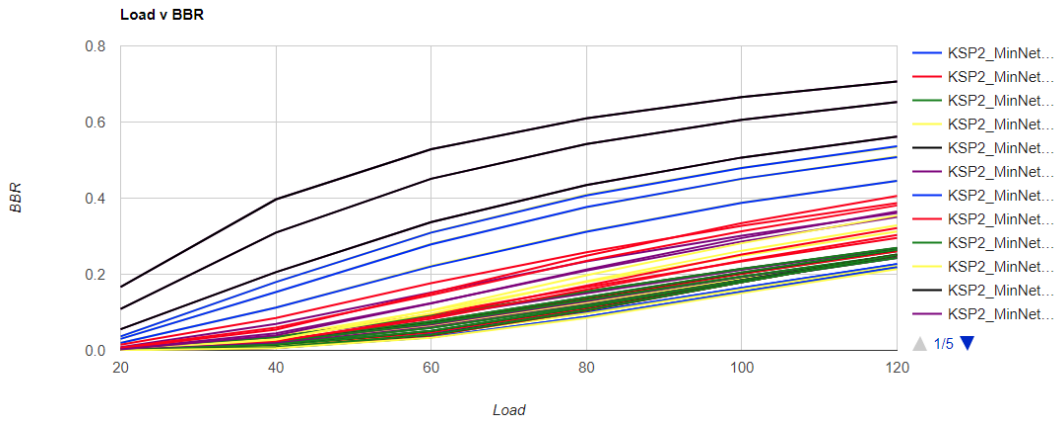

Figura 4. Exemplo de gráfico gerado pelo sistema de despacho.

de segurança das tarefas, garantindo a corretude total dos resultados obtidos. Além desse fator, o despachador conseguiu completar toda a execução em um tempo relativamente próximo ao esperado em um cenário ideal sem falhas, onde cada máquina executaria dois processos por vez durante o tempo de 30 minutos, resultando em 7,5 horas de processamento contínuo das máquinas.

5. Considerações finais

O uso de simuladores têm sido uma ferramenta indispensável para o estudo e modelagem de novas tecnologias de carácter científico. Compreender e comprovar resultados de simulação implica a realização de análises estatísticas bem definidas e a analise de diversos dados obtidos em cada replicação. A configuração e a organização das simulações é um processo oneroso suscetível a erros, o que torna a realização da pesquisa um processo dispendioso. Com o objetivo de sanar estes problemas, este trabalho propõe um sistema de automação para o despacho de simulações de forma distribuída com o propósito de auxiliar a configuração e a organização de simulações de cunho científico. O sistema proposto ainda é capaz de realizar o balanceamento de carga e prover um sistema de tolerância a falhas.

Os resultados experimentais da proposta permitiram a otimização do tempo gasto no processo de execução das simulações decorrentes do retrabalho que frequentemente ocorria por conta de falhas adversas a erros de execução e configuração do ambiente. Além disso, o sistema possibilita a organização dos resultados gerados em base de dados central facilitando a manuseio e a gestão dos resultados das simulações. Foram apresentados exemplos de execução do despachador através de uma interface intuitiva que permite a otimização do tempo despendido pelos pesquisadores em suas pesquisas acadêmicas.

Referências

Costa, L. R., de Sousa, L. S., Rodopoulos, F., Silva, K., Júnior, P. J. S., and Drummond, A. C. (2016). ONS: Simulador de Eventos Discretos para Redes Ópticas WDM/EON. In *SBRC 2016 - Salão de Ferramentas*, Salvador, Bahia, Brazil.

MongoDB (2018). Mongodb, Inc. https://www.mongodb.com/.

NodeJS (2018). Node.js Foundation. https://nodejs.org/en/.