Brazilian Journal of Development

Escalonamento de processos não preemptivos: um modelo de simulação

Scheduling of non-preemptive processes: a simulation model

DOI:10.34117/bjdv5n6-215

Recebimento dos originais: 14/04/2019
Aceitação para publicação: 30/05/2019

Jhonatan Thallisson Cabral Néry
Mestrando em Computação Aplicada pela Universidade do Estado de Santa Catarina, UDESC
Instituição: Universidade do Estado de Santa Catarina, UDESC
Endereço: Zona Industrial Norte, Joinville - SC, Brasil
E-mail: jhonatanthallisson@gmail.com

Franciny Medeiros Barreto
Doutoranda em Ciência da Computação pela Universidade Federal de Uberlândia - UFU.
Instituição: Universidade Federal de Goiás – Regional Jataí (UFG – Jataí).
Endereço: BR 364, km 195, nº 3800 – Setor Industrial – Jataí – GO, Brasil
E-mail: francinymedeiros@gmail.com

Joslaine Cristina Jeske de Freitas
Doutora em Ciência da Computação pela Universidade Federal de Uberlândia – UFU.
Instituição: Universidade Federal de Goiás – Regional Jataí (UFG- Jataí)
Endereço: BR 364, km 195, nº 3800 – Setor Industrial – Jataí – GO, Brasil
E-mail: joslaine@gmail.com

RESUMO

A sociedade moderna tem exigido cada vez mais requisitos da tecnologia, sendo alguns deles agilidade, disponibilidade e a possibilidade de tratar diferentes tarefas simultaneamente. Assim, uma das pesquisas para atender o atual cenário tem seu foco no tratamento das aplicações em execução (os chamados processos) que podem estar em um Sistema Operacional em funcionamento, por exemplo. Estes tratamentos visam determinar quando os processos devem ser executados, por quanto tempo, se deve existir um índice de prioridade de execução, etc. Tudo isto é estudado para que não se note atraso ou indisponibilidade dos recursos. Em um computador, o elemento responsável por tratar estes processos é o escalonador. Esta pesquisa tem por objetivo construir um simulador que mostra especificamente o comportamento do escalonador *First-in First-out* em cenários com 1, 2 e 3 processadores com processos de duração aleatória.

Palavras-chave: Simulação, Escalonamento; CPN Tools.

ABSTRACT

Modern society has increasingly demanded technology requirements, some of them agility, availability and the ability to handle different tasks simultaneously. Thus, one of the researches to meet the current scenario has its focus on the treatment of running applications

(so-called processes) that may be in a functioning Operating System, for example. These treatments aim to determine when the processes should be executed, for how long, if there should be apriority index of execution, etc. All this is studied so that there is no delay or unavailability of resources. On a computer, the element responsible for handling these processes is the scheduler. This research aims to construct a simulator that specifically shows the behavior of the First-in First-time scheduler in scenarios with 1, 2 and 3 processors with random duration processes.

Keywords: Simulation; Scheduling; CPN Tools.

1 INTRODUÇÃO

Todo computador possui recursos finitos, então todos estão sujeitos a atrasos e escassez de modo geral. Por este motivo, é importante que se utilizem políticas de alocação bem estruturadas e que não deixem evidente a carência de recursos.

Um Sistema Operacional (SO), segundo [Silberschatz 2009, Deitel et al. 2005], é um programa que controla o *hardware* (parte física) de um computador e é responsável também por fornecer infraestrutura para a execução das aplicações de usuário. Assim, o SO atua como base intermediária entre o usuário e o *hardware* da máquina. Mais especificamente, o SO realiza tarefas complexas e uma delas é o escalonamento de processos, que de acordo com [Maziero 2014] é o mecanismo que aloca recursos de processamento aos programas da máquina, ditando quem usará, em que momento e por quanto tempo.

Atualmente existem várias técnicas de escalonamento diferentes, cada uma delas propondo uma abordagem distinta. Tendo isto em vista, manifesta-se uma pergunta: Quais os possíveis comportamentos de um escalonador e qual sua conduta quando o número de processadores é alterado? Então, este artigo tem como objetivo responder a esta indagação, mostrando através de simulação, como um escalonador *First-Come First-Served* (FCFS), também chamado de First-In First-Out (FIFO) pode se comportar quando aplicado a cenários distintos e como isto pode contribuir para a otimização do uso do processador.

[Medeiros et al. 2014] afirmam que a simulação é uma técnica que já se provou pertinente na resolução de problemas das mais variadas áreas e consiste na criação de modelos abstraídos dos sistemas originais. Estes modelos buscam gerar dados que possibilitem uma análise mais detalhada para se entender o comportamento geral do sistema, pontos fortes e dificuldades, possibilitando assim a elaboração de melhorias.

Para a construção do modelo e simulação do escalonador será utilizada a ferramenta CPN Tools, que se mostra adequada para a criação e simulação de modelos computacionais. Como passos para chegar ao objetivo proposto serão cumpridas as seguintes etapas:

- Analisar o escalonador não-preemptivo FIFO.

Brazilian Journal of Development

- Construir a representação do algoritmo em rede de Petri.
- Construir o modelo no CPN Tools baseado na rede de Petri.
- Criar cenários e simular o funcionamento do escalonador FIFO com um, dois e três processadores.
- Analisar os dados colhidos.

2 REFERENCIAL TEÓRICO

Nesta seção é apresentada a fundamentação teórica necessária para compreensão deste trabalho. A subseção 2.1 apresenta os conceitos teóricos relacionados aos Sistemas Operacionais. A subseção 2.2 mostra o conceito de Redes de Petri e por fim a subseção 2.3 apresenta a ferramenta CPN.

2.1 SISTEMAS OPERACIONAIS

Um SO é formado por um programa, ou um conjunto de programas e, de acordo com [Tanenbaum 2003], tem basicamente duas funções:

- Criar uma interface, ou seja, intermediar as interações entre aplicações de usuário e o *hardware*
- Ser um gerenciador de recursos, um administrador dos recursos de *hardware* e *software*, coordenando como e quando os recursos são utilizados.

[Deitel et al. 2005, Silberschatz 2009] definem SO como sendo um *software* que possibilita a interação entre as aplicações e o *hardware* de um computador, além disto fornece serviços que permitem que cada aplicação seja executada com segurança e efetividade.

[Maziero 2014] afirma ainda que SOs são *softwares* grandes e por isto precisam ser criados com certos requisitos, como eficiência e agilidade na gerência das atividades computacionais, escalabilidade para prover facilidade de evolução e segurança para que as atividades computacionais sejam protegidas dos demais usuários da máquina e de agentes externos.

Os computadores atuais vêm com uma série de componentes, como por exemplo um ou mais processadores, memórias, dispositivos de E/S, etc. Os programas que utilizam estes componentes precisam saber interagir, ou seja, precisam de um tradutor que consiga gerar comunicação entre as aplicações e os dispositivos físicos [Machado e Maia 2013].

[Machado e Maia 2013]ainda reiteram que os SOs se fazem necessários, pois em um cenário onde as aplicações não têm este intermediário, elas precisam compreender todas as particularidades do *hardware* utilizado, pois terão também o papel de se comunicar diretamente com estes dispositivos a fim de conseguir por fim executar suas tarefas principais, as tarefas para as quais foram desenvolvidas. Neste cenário, se terá um gasto maior com programação, pois haverá maior complexidade na criação das aplicações de usuário, maior complexidade de manutenção e os *softwares* não terão portabilidade, pois para cada *hardware* diferente, será necessário criar uma nova aplicação feita sob medida. Aplicações projetadas sobre um SO são mais racionais, pois detêm o foco em sua tarefa central (alto nível de abstração), normalmente são portáveis e exigem um custo menor em sua criação e manutenção.

2.2 REDES DE PETRI

As redes de Petri (ou RdP) surgiram em 1962, por iniciativa de Carl Adam Petri. Desde então este modelo tem se popularizado cada vez mais. Um grande autor que se aprofundou e disseminou este modelo formal foi [Murata 1989]. Sua obra retrata as redes de Petri como sendo uma forma eficiente de simulação de sistemas. É importante salientar que quando é citada a palavra sistema neste contexto, não significa especificamente sistemas computacionais, mas qualquer espécie de sistema organizacional, como, por exemplo, logística de entrega de mercadorias, tráfego de automóveis, atendimento em supermercados, processos industriais, etc.

Redes de Petri são consideradas uma ferramenta gráfica e matemática de representação formal que permite modelagem, análise e controle de sistemas a eventos discretos que comportam atividades paralelas, concorrentes e assíncronas [Murata 1989]. Uma rede de Petri pode ser vista como um grafo bipartido e direcionado com dois tipos de nós chamados de lugares e transições. Os nós são conectados por meio de arcos direcionados. Conexões entre dois nós do mesmo tipo não são permitidas.

Formalmente, as redes de Petri são definidas da seguinte forma

[Cardoso e Valette 1997] e [Murata 1989]:

Uma rede de Petri é uma quádrupla $R = <P, T, Pre, Pos>$ onde:

- P é um conjunto finito de lugares de dimensão n,
- T é um conjunto finito de transições de dimensão m,

Brazilian Journal of Development

- *Pre: P x T→**N*** é a aplicação de entrada (lugares precedentes ou incidência anterior), com ***N*** sendo o conjunto dos números naturais,

*Pos: P x T →**N*** é a aplicação de saída (lugares seguintes ou incidência posterior).

2.3 CPN TOOLS

As redes de Petri coloridas foram idealizadas por Kurt Jensen, pesquisador da Universidade de Aarhus, na Dinamarca. Seu artigo, datado do ano de 1981, intitulado *Coloured Petri Nets and the Invariant Method* foi um trabalho promissor, que induziu uma série de estudos sobre coloração das fichas.

Há várias razões para o uso de Redes de Petri Coloridas e CPN Tools [van der Aalst et al. 2013]:

- O uso de redes de Petri fornece uma análise para correção do modelo por meio de, por exemplo, uma análise do *state space*. É importante salientar que a análise de estado é realizada a partir do modelo de alto nível através da replicação de cenário e não do estudo do modelo de rede de Petri ordinário subjacente. Isso é então um tipo de análise de propriedade de um modelo de alto nível que é o equivalente a um programa produzido numa linguagem de programação de alto nível.
- É muito fácil fazer pequenas adaptações dos modelos e depois comparar os resultados, dadas várias propostas de modelagem.
- A integração de processos e de dados complexos é essencial para modelagem de processos complexos. CPN Tools oferece a possibilidade de integrá-los em um único modelo.
- CPN Tools contém um ambiente de simulação, em que se pode seguir passo-a-passo uma sequência de disparo. Simulação do tipo Monte Carlo (com replicação de um mesmo cenário aleatório e o cálculo de estimativas de valores médios que pertencem a intervalos de confianças calculados automaticamente pelo *software*.

A ideia do CPN é unir a capacidade de representar a sincronização e a competição das redes de Petri com o poder expressivo das linguagens de programação com seus tipos de dados [Jensen e Kristensen 2009]. CPN é uma linguagem para a modelagem, simulação e validação de sistemas em que a concorrência, a comunicação e a sincronização

desempenham um papel importante. Então, CPN é uma linguagem de modelagem de eventos discretos que combina redes de Petri com a linguagem de programação funcional Standard ML [Milner et al. 1997].

3 IMPLEMENTAÇÃO

Esta seção apresenta como o modelo é construído. Inicialmente, defini-se o modelo (subseção 3.1). Na subseção 3.2 a metodologia é apresentada. O monitoramento do modelo é mostrado na subseção 3.3 e por fim, na subseção 3.4 a execução do modelo é realizada.

3.1 DEFINIÇÃO DO MODELO

[Maziero 2014]define o escalonador de processos como sendo o nível mais baixo, complexo e o processo com maior prioridade em um SO. [Tanenbaum 2003]cita diferentes tipos de SO multiprogramados possíveis. Dentre estes, a presente pesquisa buscará mostrar o escalonador FIFO, que se encaixa no grupo dos sistemas do tipo Lote não preemptivos.

Para criar um modelo de simulação será necessário encontrar um cenário de escalonamento, com o objetivo de retirar alguns dados fundamentais, como tempo de chegada de processos e tempo de execução de cada um deles.[Stallings 2012]descreve um cenário de escalonamento de cinco processos que, como ilustrado na Tabela 1 é composto pelos dados de hora de chegada (tempo de chegada de cada processo) e tempo de serviço (tempo de execução), que são exatamente os dados necessários para dar andamento as simulações.[Stallings 2012]especifica que neste cenário os processos exigem uso alternativo de processador e E/S de forma repetitiva.

Processo	Hora de chegada	Tempo de Serviço
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2

Tabela 1. Exemplo de escalonamento de processo, [Stallings 2012]

Ainda usando este cenário, é possível identificar o tempo médio de chegada e o tempo médio de execução dos processos. Usando a Tabela 1 como base, tem-se o seguinte:

- Média chegada = (0+2+4+6+8)/5 = 4
- Média execução = (3+6+4+5+2)/5 = 4

Baseado no cenário proposto por [Stallings 2012] e no algoritmo apresentado por [Maziero 2014]este trabalho propõe construir o algoritmo de escalonamento FIFO utilizando CPN Tools e simulá-lo em diferentes cenários. Esta proposta difere-se do apresentado por[Stallings 2012]pois deseja-se mostrar uma situação mais próxima da realidade. Para tanto, funções exponenciais serão utilizadas para gerar valores aleatórios das datas de chegada e tempo de serviço.

A Figura 1 mostra a rede de Petri gerada baseada no funcionamento do escalonador FIFO.

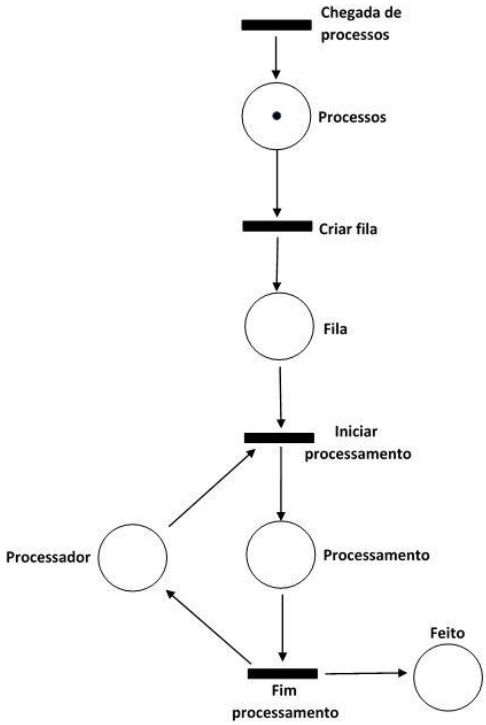

Figura 1: Rede de Petri do escalonador FIFO.

3.2 METODOLOGIA DE DESENVOLVIMENTO

A seguir serão mostrados os passos da construção do modelo em CPN Tools. As funções utilizadas e definidas como **iat** e **Mtime** são funções prontas da ferramenta e segundo [CPNTools 2018] são definidas pelo seguinte:

- **fun iat(ET) = round (exponential(1.0/ET))**: Os valores aleatórios gerados durante a simulação são feitos através desta função, onde 1.0 é a unidade de tempo e o parâmetro ET da função é igual ao valor médio da distribuição exponencial, neste caso 4.

- **fun Mtime() = IntInf.toInt(time())**: Esta função foi declarada de tal forma que retornará o tempo atual no modelo como um número inteiro.

A Figura 2 mostra a chegada de processos no lugar de mesmo nome, onde o número *1* significa a chegada de um processo por vez, tendo cada um deles um identificador crescente (1,2,3...) chegando em um tempo aleatório. A variável *JobID* é uma variável temporizada responsável por armazenar o número identificador de cada processo e o momento em que cada um chega. A variável *j* recebe o valor de *JobID* e o leva até a transição *criar fila*, que envia cada processo ao próximo lugar. Já o arco *j+1@+iat(4.0)* soma o *j* atual com o processo que entrará posteriormente, sendo que *iat* é a função exponencial, o que proporcionará a chegada de processos em intervalos de tempo aleatórios dentro da média de 4 unidades de tempo.

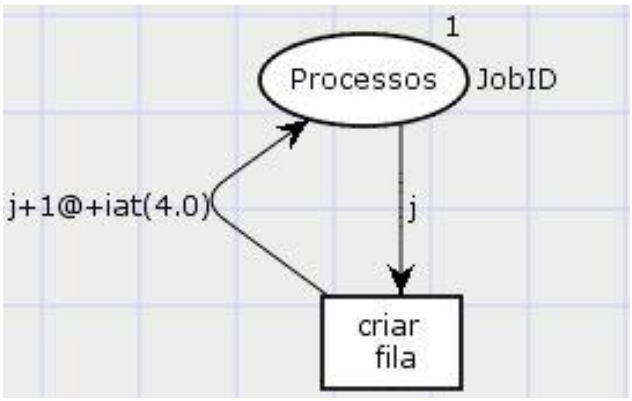

Figura 2: Chegada de processos.

A Figura 3 descreve a chegada de processos na fila de processos prontos *Fila*, onde a variável do tipo lista *l* é concatenada a lista com o número do identificador do processo e o tempo que o processo chegou na fila, ou seja é concatenada com [(j, Mtime())], sendo **Mtime()** uma função criada para calcular o tempo exato que o processo chega ao lugar *Fila*. O lugar *Fila* guarda todos os processos em uma lista em ordem de chegada. A lista *l* é devolvida ao lugar *criar fila* sempre que uma tupla de processo é adicionada na lista, de modo a formar a fila de processos.

Brazilian Journal of Development

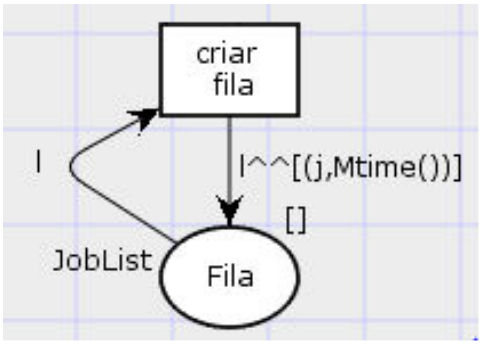

Figura 3: Chegada de processos a fila.

A Figura 4 demonstra o estágio onde os processos são entregues a transição *iniciar processamento*, o arco *(j,t) :: l* retira o primeiro processo da lista *l* sendo que *j* é o número do processo e *t* recebe Mtime(), que é o tempo de chegada na fila. A lista *l* é devolvida ao lugar *Fila* sempre que um processo é retirado da lista, de modo a atualizar a lista de processos.

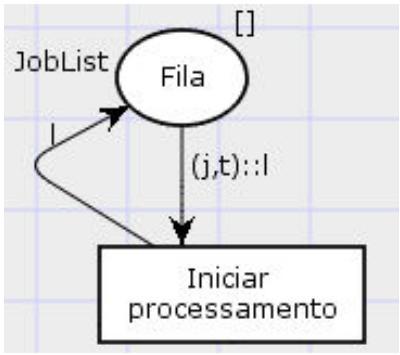

Figura 4: Saída de processos da fila.

A partir deste ponto, para que um processo possa ser enviado da transição *iniciar processamento* para o lugar *Processamento* é necessário que se tenha certeza que o processador estará livre para receber um novo processo. Para atender a esta necessidade, a transição *iniciar processamento* foi criada de modo à somente enviar novos processos ao lugar *Processamento* se receber antes uma ficha do lugar *Processador* avisando que o mesmo está livre e apto a receber um novo processo.

Brazilian Journal of Development

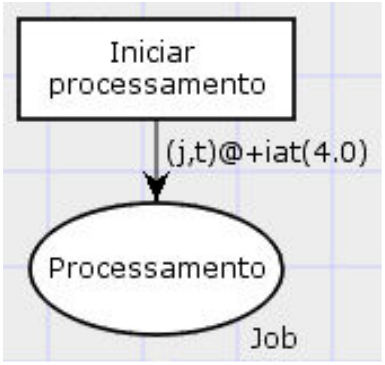

Figura 5: Execução de processos.

A Figura 5 ilustra o tempo de processamento definido pelo arco *(j,t)@+iat(4.0)* para cada processo, onde cada um deles recebe um tempo de processamento aleatório de acordo com a média de tempo de execução informada na função exponencial **iat**.

O fim do processamento se dará logo que o processo chegar a transição *Fim Processamento*, representado na Figura 6. A variável *Job* armazena e repassa para o arco *(j,t)* os dados de identificador de processo *j*, tempo de chegada na fila e tempo de término da execução do processo, sendo que as duas informações estão contidas agora na variável *t*.

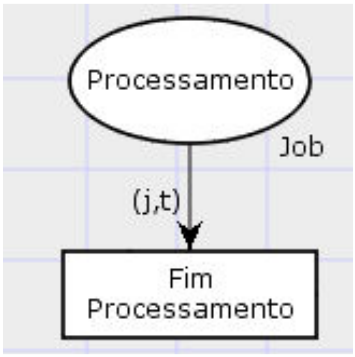

Figura 6: Fim do processamento.

A finalização da execução de cada processo será quando a transição *Fim Processamento* enviar os processos para o lugar *Feito* e devolver a ficha para o recurso *Processador*. Isto é mostrado na Figura 7.

Brazilian Journal of Development

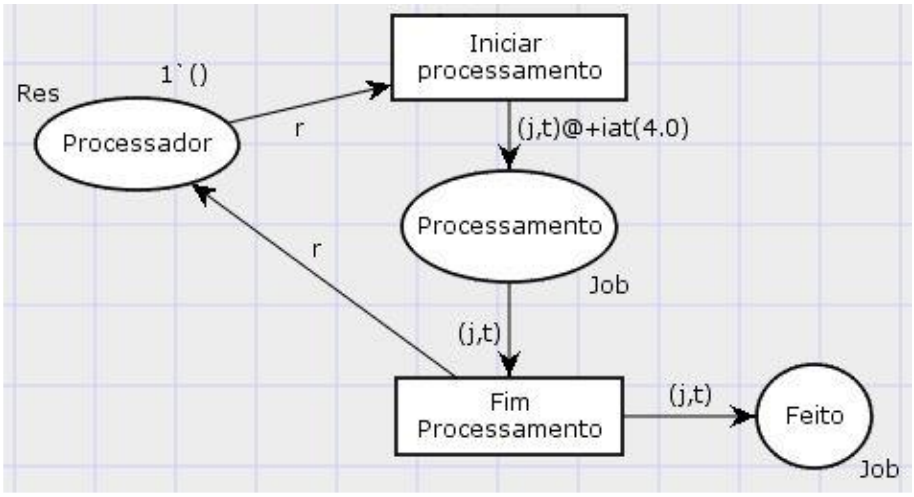

Figura 7: Fim do escalonador.

A Figura 8 mostra o modelo de simulação completo do escalonador FIFO, criado na ferramenta CPN Tools.

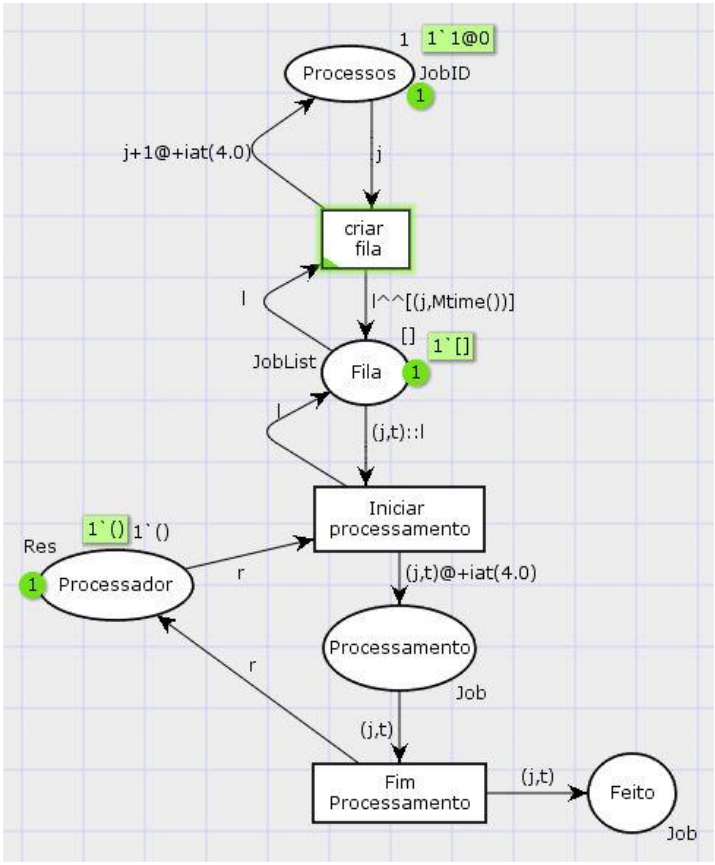

Figura 8: Modelo de simulação completo do escalonador FIFO.

Brazilian Journal of Development

Para a simulação do funcionamento do escalonador FIFO com 2 e 3 processadores o lugar *Processador* receberá o valor 2() e 3() respectivamente, demonstrando que o mesmo trabalhará com mais de um processo por vez.

3.3. MONITORAMENTO

O CPN Tools possui uma ferramenta eficiente para a observação do comportamento de modelos simulados, os chamados monitores. Através de monitores é possível obter informações claras e precisas que não ficam aparentes apenas com a visualização dos modelos. Com isto em vista foram criados os seguintes monitores:

- **Disponibilidade do Processador**: É um monitor do tipo *Marking size* que identifica quando o processador está livre.
- **Tamanho da Fila**: Monitor do tipo *List length data collection*, que verifica o tamanho da fila de processos.
- **Tempo de Chegada**: Monitor tipo *Data collection*, responsável por identificar o tempo em que cada processo chega ao escalonador.
- **Tempo de Espera**: Monitor tipo *Data collection*, responsável por identificar o tempo em que cada processo permanece aguardando processamento. Este monitor captura o tempo de cada processo na transição *Iniciar processamento* e subtrai o tempo de chegada do processo, expondo assim o tempo de espera.
- **Tempo de Processamento**: Monitor também do tipo *Data collection*, responsável por monitorar o tempo de cada processo desde a sua chegada até o final do processamento. Isto é dado pelo tempo capturado na transição *Fim Processamento* menos o tempo capturado na transição *Criar fila*.

3.4 EXECUÇÃO DO MODELO

As simulações simples podem ser executadas aplicando as ferramentas da paleta *Simulation* do CPN Tools. Mas é importante se certificar que os valores extraídos das simulações são confiáveis e que não se tratam de dados anômalos que invalidarão as estatísticas. Com o intuito de conseguir esta segurança as simulações devem ser replicadas inúmeras vezes. A função **CPN'Replications.nreplications** pode ser usada para executar automaticamente um determinado número de replicações.

Brazilian Journal of Development

4 ANÁLISE DOS RESULTADOS

A partir das simulações geradas pela ferramenta CPN Tools é possível identificar vários aspectos da execução e suas nuances quando alterado o número de processadores.

O monitor **Disponibilidade do Processador** mostrou que o número máximo de processos que passaram pelo processador (independentemente da quantidade de processadores) este número permaneceu em 6668 processos. Também foi possível identificar que a disponibilidade máxima de processadores, ou seja, o número máximo de processadores livres simultaneamente em algum instante da execução do modelo, é proporcional a quantidade de processadores do mesmo. Outro dado exposto é a média de disponibilidade, que correlaciona o aumento de processadores ao aumento da ociosidade dos mesmos, mostrando que, em média, para um modelo de 1 processador 1,73% de seu poder de processamento fica ocioso, já no modelo com 2 processadores este valor é de 50,16% e com 3 processadores este dado de ociosidade fica mais alarmante ainda, mostrando 66,91% de desuso do poder de processamento dos 3 juntos. Estes dados podem acionar um alerta para os utilizadores de uma arquitetura como esta e provocar estratégias de melhor utilização do poder de processamento.

Para o monitor **Tamanho da Fila** foi possível observar que a média de elementos na fila diminui substancialmente com o aumento do número de processadores, sendo que com 1 processador a média de elementos é de 44,90684, quando aplicados 2 processadores ao modelo este número cai 99,25%, indo para 0,338066 elementos e caindo mais ainda com o uso de 3 processadores, alcançando 0,046035, o que é 86,38% menor que com o modelo de 2 processadores e 99,90% menor que o tamanho da fila quando se tem apenas 1 processador no modelo.

O monitor **Tempo de Chegada** apresenta várias informações a respeito dos tempos de chegada dos processos ao escalonador FIFO. A média de tempo de chegada é bastante próxima em todos os modelos, pois na geração dos tempos aleatórios de chegada foi empregada uma função exponencial de média 4 que gera valores aleatórios em torno desta média e se usando o intervalo de confiança de 95% os valores ficam praticamente iguais.

O monitor **Tempo de Espera** mostra algumas respostas sobre quanto tempo os processos tiveram que esperar na fila por sua execução. É possível distinguir que o tempo de espera é expressivamente menor quando o número de processadores é aumentado. A média de tempo de espera mostra que no cenário de um escalonador FIFO, onde não existe preempção, ou prioridades, os processos precisam esperar em ordem de chegada na fila de

processos prontos e no modelo com 1 processador esta espera penaliza as aplicações, fazendo com que aguardem por muito tempo pelos resultados do processamento, sendo em média 178,585786 unidades de tempo o que cada processo aguarda na fila. O modelo com 2 processadores já mostra uma redução deste tempo em 99,24% e com 3 processadores a redução é de 86,40% em comparação ao modelo com apenas 2 e de 99,90% em relação ao modelo com 1 processador. Usando o intervalo de confiança de 95% é possível afirmar que estes valores podem variar em até 20,499204 unidades de tempo. Esta informação é valiosa para a definição de eficiência do escalonador FIFO em comparação com outros escalonadores e com isto auxilia na eleição do melhor escalonador a se implantar em um SO.

Por fim o monitor **Tempo de Processamento** que visa expressar o tempo total que cada processo gastou no escalonador (tempo de espera + tempo de processamento), consegue abstrair dados bastante pertinentes, já que o tempo total de processamento é fator determinante para avaliar uma arquitetura de escalonamento. Nestas estatísticas é possível ver que quanto menos processadores, mais sobrecarregado o sistema fica, gerando grandes filas e com isto uma espera proporcional, o que faz com que o tempo total de processamento seja maior.

O monitor **Tempo de Processamento** mostra os seguintes valores: O tempo médio total de processamento com 1 processador é de 182,566413 unidades de tempo, com 2 esta média fica em 5,33624 unidades e se colocando 3 processadores para trabalhar em paralelo este número decresce para 4,150723 unidades de tempo. Sabendo estas informações é possível alegar que o cenário menos atrativo é o de 1 processador, o de 2 e 3 processadores quase se equiparam em eficiência, à vista que o modelo de 2 processadores é 97,08% mais rápido que o modelo com 1 e o modelo de 3 processadores é pouco diferente, ficando com 97,73% de rapidez a mais que o de 1 processador apenas.

Sabendo da diferença próxima de desempenho entre 2 e 3 processadores é possível concluir que com o aumento da quantidade de processadores, esta diferença de performance tenderá a diminuir e a ser cada vez menos significativa dentro de um escalonador FIFO. Com isto, indivíduos e instituições podem usar estes dados para comparação com outros escalonadores e também para decidir a respeito do melhor custo/benefício ante a questão quantidade de processadores *versus* ganho de desempenho.

5 CONSIDERAÇÕES FINAIS

As simulações mostraram que a fila de processos prontos diminui expressivamente quando o número de processadores é aumentado e por consequência o tempo de espera dos processos também regride, o que impacta diretamente no tempo total de processamento.

É possível compreender também que o escalonador FIFO a partir de 2 processadores passa a ter seu tempo total de processamento praticamente igual ao tempo de utilização do processador, pois a partir desta quantidade de processadores o tempo de espera diminui drasticamente e por isto o tempo total quase não é impactado. Isto pode ser visto porque a média dada a função exponencial para criação dos tempos de execução foi 4 e quanto mais processadores adicionados, mais o tempo total se aproxima deste tempo de execução no processador.

Ciente da informação de que mais processadores melhoram o desempenho de um sistema de escalonamento FIFO, mas que em determinado ponto este aumento de *hardware* não gera mais ganho significativo de performance, expondo também todos os aspectos de comportamento deste escalonador e ainda gerando um método para que a eficiência desta arquitetura seja atestada, o indivíduo, ou organização conseguirá construir ambientes otimizados para manipulação de processo. Conseguirá também avaliar se investimentos em processamento são necessários e se houver um limite de orçamento, é possível identificar os parâmetros de comparação e meios para simular outros escalonadores e outras quantidades de processadores.

Sobre a ferramenta CPN Tools, pode-se afirmar que esta se mostrou maleável e prática, conseguindo implementar elementos teoricamente complexos de forma simples e isto foi decisivo para a realização deste trabalho.

Quanto a visão do poder computacional das simulações para explicar e auxiliar na análise de situações e arquiteturas, é correto dizer que este método é eficaz quando se consegue gerar um modelo assertivo. Assim sendo, as simulações conseguem fornecer informações precisas e relevantes sobre situações hipotéticas ou reais, dando plenas condições para a tomada de decisão.

6 TRABALHOS FUTUROS

A contribuição deste trabalho é a modelagem, construção e simulação de um escalonador do tipo FIFO a fim de mostrar o seu comportamento em diversos cenários. Deste modo é possível visualizar o tratamento de processos feito por ele, permitindo a

análise de situações que podem ocorrer quando processos são escalonados e quando são escalonados com números diferentes de processadores. Com os dados obtidos pela simulação, é possível aplicar este conhecimento em outras situações onde o escalonamento é necessário, por exemplo, em agendamento de tarefas em um Sistema Gerenciador de Workflow.

Algumas sugestões de trabalhos futuros vem de uma das vertentes que não foi abordada aqui, que é a comparação entre o escalonador FIFO e outros escalonadores. Onde a ferramenta CPN Tools poderia facilmente gerar relatórios e provocar discussões construtivas a respeito do melhor escalonador para determinados cenários.

Trabalhos posteriores podem também explorar simulações com um número maior de processadores, podendo talvez levar a novas descobertas sobre escalonamentos.

Pode-se também usar a metodologia exposta aqui para criar novas abordagens que tratem da eficiência do uso de processadores e da relação custo x benefício que estas novas abordagens podem oferecer.

REFERÊNCIAS

Cardoso, J. e Valette, R. (1997). **Redes de Petri**. Série Didática. Editora da UFSC, Florianópolis, SC.

CPNTools(2018). http://cpntools.org/2018/01/16/documentation-2

Deitel, H., Deitel, P., e Choffnes, D. (2005). **Sistemas Operacionais**. PRENTICE HALL BRASIL.

Jensen, K. e Kristensen, L. M. (2009). **Coloured Petri Nets**. Springer, Germany.

Machado, F. B. e Maia, L. P. (2013.). **Arquitetura de Sistemas Operacionais**:. LTC, Rio de Janeiro-RJ:, 5.ed. edition.

Maziero, C. A. (2014). **Sistemas Operacionais: Conceitos e Mecanismos**. Livro aberto.

Medeiros, L., Moser, A., e Santos, N. (2014). **A Simulação Computacional como Técnica de Pesquisa na Administração**. Revista Intersaberes — vol.9, n. especial.

Milner, R., Tofte, M., e Macqueen, D. (1997). **The Definition of Standard ML**. MITPress, Cambridge, MA, USA.

Murata, T. (1989). **Petri nets: Properties, Analysis and Applications**. Proceedings of the IEEE, 77(4):541–580.

Silberschatz, A. (2009). **Operating System Concepts**. John Wiley & Sons Software, 8thedition.

Stallings, W. (2012). **Operating Systems: Internals and Design Principles**. Prentice Hall.

Tanenbaum, A. (2003). **Sistemas Operacionais Modernos**. Prentice-Hall do Brasil.

van der Aalst, W. M. P., Stahl, C., e Westergaard, M. (2013). **Strategies for Modeling Complex Processes using Colored Petri Nets**. In Jensen, K., van der Aalst, W. M. P.,Balbo, G., Koutny, M., and Wolf, K., editors, Transactions on Petri Nets and OtherModels of Concurrency VII, pages 6–55, Berlin, Heidelberg. Springer Berlin Heidelberg.