Utilizando Redes de Petri para Modelagem de Desempenho de Middleware Orientado a Mensagem

Roberto D. Arteiro, Fábio N. Souza, Nelson S. Rosa, Paulo R. M. Maciel

Centro de Informática – Universidade Federal de Pernambuco (UFPE)
Caixa Postal 15.064 – 91.501-970 – Recife – PE – Brasil

{rda,fns,nsr,prmm}@cin.ufpe.br

Resumo. *A crescente demanda por integração de aplicações corporativas ressalta a necessidade de uma plataforma tecnológica que garanta confiabilidade e segurança. Nesse contexto, as características dos sistemas de Middleware Orientados a Mensagem (MOMs) favorecem o seu uso como solução de integração no mercado atual. Este artigo apresenta um modelo de desempenho para MOMs desenvolvido em redes de Petri estocásticas que permite a realização de experimentos de simulação visando à identificação da capacidade de entrega de mensagens e do ponto de saturação do sistema. Para validar o modelo proposto, os resultados obtidos através de simulação são comparados com medições caixa-preta, realizadas em ambiente real, utilizando um MOM compatível com o padrão Java Message Service (JMS).*

Abstract. *The increasing demand for enterprise applications integration reveals the necessity of a technological platform that guarantees reliability and security in this process. In this context, the characteristics of the Middleware Oriented Message (MOMs) systems favor its use as an integration solution in the current market. This article presents a performance model for MOMs developed in stochastic Petri nets which allow the execution of simulation experiments intended to identify the capacity of delivery of messages and the point of saturation of the system. To validate the proposed model, the simulation results are compared with black-box measurements in a real environment, using a Java Message Service (JMS) compatible MOM.*

1. Introdução

Atualmente, a maioria das organizações utiliza um número crescente de aplicações para resolver problemas específicos do negócio. Em vários casos, essas aplicações foram criadas em momentos diferentes e utilizam plataformas tecnológicas distintas. Diante deste cenário, o desafio das organizações é integrar as aplicações para atender aos objetivos de negócio que estão em constante evolução. Respondendo a esta necessidade surge o conceito de integração de aplicações, que é o compartilhamento de processos e/ou dados, de forma segura e orquestrada, entre aplicações corporativas [Linthicum 1999].

Nos últimos anos, várias tecnologias surgiram para tratar este problema, entre elas as principais foram: *Middleware* Orientado a Mensagem (MOM), *Message Broker* (MB), *Enterprise Application Integration* (EAI), *Service Oriented Architecture* (SOA),

Web Services (WS) e *Enterprise Service Bus* (ESB). Um ponto em comum entre essas soluções é a utilização de mensagens como um dos elementos básicos da integração.

Dentre essas soluções, ESB é apontado pelo mercado como a principal alternativa [Gartner 2003] [Gartner 2004] [IDC 2003a] [IDC 2003b]. Isto se deve principalmente ao fato desta tecnologia ser uma plataforma de integração baseada em padrões (JMS, JCA, J2CA, SOAP, WSDL, BPEL4WS, XML etc.), e que combina o poder dos MOMs e *web services* com o suporte a roteamento inteligente e a transformação de dados (inerentes aos *Message Brokers)* para conectar de forma segura e coordenada a interação de um número significativo de aplicações corporativas com integridade transacional [Chappell 2004].

Os MOMs compõem o núcleo central da arquitetura dos ESBs, provendo a infra-estrutura básica de uma rede de canais virtuais que o barramento utiliza para rotear mensagens dentro e fora das organizações. A utilização do MOM garante ao barramento a capacidade de desacoplar as aplicações, permitindo que elas conversem entre si, sem necessariamente saberem uma da existência da outra.

Diante desse contexto, onde a responsabilidade final de encaminhamento de todas as mensagens destinadas à integração das aplicações corporativas fica a cargo de um sistema de MOM, surge um desafio maior: como planejar a capacidade dos sistemas de MOM de forma a não comprometer o processo de integração. Segundo [Menascé e Almeida 2002], planejamento de capacidade é o processo de prever e identificar possíveis pontos de saturação no ambiente, e definir um plano de adequação para continuar a prestar o serviço no nível especificado considerando o menor custo possível.

Dessa forma, este artigo tem como objetivo apresentar modelos de desempenho para MOMs, compatíveis com o padrão JMS (*Java Message Service*), de forma a permitir a identificação da (1) capacidade máxima de entrega de mensagens mantendo um nível de serviço acordado, e (2) do ponto de saturação destes sistemas. Estes modelos utilizam o formalismo de redes de Petri estocásticas, permitindo, via processo de simulação, a avaliação de desempenho do MOM sob vários cenários de carga. Com isso, apresenta uma solução para resolver o primeiro passo do processo de planejamento de capacidade.

O restante deste artigo é organizado como segue. A Seção 2 apresenta os conceitos básicos sobre MOM, avaliação de desempenho e redes de Petri. Os modelos desenvolvidos são apresentados na Seção 3, seguido do processo de validação. Na Seção 4 são apresentados resultados obtidos a partir dos modelos. Finalmente, a Seção 5 apresenta as conclusões e trabalhos futuros.

2. Conceitos Básicos

Antes de apresentarmos os modelos em redes de Petri, as próximas subseções introduzem conceitos básicos de MOM, avaliação de desempenho para planejamento de capacidade e redes de Petri estocásticas.

2.1. Middleware Orientado a Mensagem (MOM)

Middleware Orientado a Mensagem [Banavar et al. 1999] é um conceito que envolve a passagem de dados entre aplicações utilizando um canal de comunicação que trafega

unidades autônomas de informação chamadas de "mensagens". Em um ambiente de comunicação baseado em MOM, mensagens são usualmente enviadas e recebidas de forma assíncrona.

Neste cenário, as aplicações são desacopladas, pois transmissores e receptores não precisam conhecer a existência dos outros. Em vez disso, eles enviam e recebem mensagens de/para o MOM, sendo responsabilidade do MOM definir mecanismos que façam com que as mensagens cheguem aos destinatários especificados. De forma geral, em um MOM, uma aplicação utiliza uma API para se comunicar com um agente cliente (messaging client) que é disponibilizado pelo fabricante do MOM.

O MOM é responsável pelo gerenciamento dos pontos de conexão dos múltiplos agentes, e por gerenciar múltiplos canais de comunicação entre os pontos de conexão. Quando uma aplicação está se comunicando desta forma, ela está atuando como um produtor ou um consumidor. Os transmissores (produtores) produzem mensagens e os receptores (consumidores) consomem mensagens. Uma aplicação pode ser ao mesmo tempo um produtor e um consumidor.

O produtor não necessita conhecer quais aplicações estão recebendo uma mensagem, ou em alguns casos nem mesmo quantas aplicações a estão recebendo. Do mesmo modo, o consumidor não necessita conhecer quais aplicações estão enviando os dados, sabe somente que recebe uma mensagem. Produtores e consumidores precisam apenas conhecer qual o formato das mensagens e qual o canal utilizar para enviá-las ou recebê-las. Se houver necessidade de uma resposta, esse fato é informado através da mensagem com a presença de um destino de *ReplyTo* para identificar para que canal deve ser emitido uma resposta.

Com isso pode-se afirmar que produtores e consumidores são fracamente acoplados uma vez que não se comunicam diretamente. Em vez disso, eles são abstratamente conectados um ao outro através de um canal virtual, que pode ser classificado como canal ponto-a-ponto (PTP) ou canal *publish-subscribe* (pub/sub). Esses dois tipos distintos de canais são freqüentemente referenciados como filas e tópicos, respectivamente. No modelo pub/sub, múltiplos consumidores podem registrar interesse em um tópico (assinar o tópico). Um produtor envia uma mensagem no canal publicando nesse tópico, fazendo com que cada assinante do tópico receba uma cópia da mensagem.

No modelo ponto-a-ponto, somente um consumidor pode receber uma mensagem enviada para uma fila. A fila pode até ter mais de um consumidor para fins de balanceamento de carga ou de redundância, garantindo tolerância a falhas no processamento da mensagem. Entretanto, somente um receptor pode consumir cada mensagem individual. Também pode acontecer de não existir nenhum receptor conectado a fila, neste caso a mensagem fica retida no MOM até que algum receptor se habilite a recuperá-la. O mesmo não acontece no modelo pub/sub, no qual as mensagens não marcadas para serem entregues através de um canal confiável podem ser descartadas se não houver nenhum assinante para o tópico. Isto não significa que o assinante precisa estar conectado no momento que a mensagem foi produzida, apenas que ele tenha assinado o tópico.

A entrega confiável de mensagens é um dos serviços disponibilizado pelos MOMs. Para tanto, as mensagens enviadas precisam de uma marcação de persistente indicando este modo de entrega. O serviço pode ser provido para uma mensagem ou para um canal (fila ou tópico). Na implementação desse serviço o MOM utiliza o mecanismo *store-and-forward* para cumprir o contrato com o produtor. O mecanismo de armazenamento (*store*) é utilizado para persistir a mensagem em disco, garantindo que a mesma possa ser recuperada em caso de falhas no MOM ou no consumidor. O mecanismo de encaminhamento (*forward*) é responsável em recuperar as mensagens armazenadas em disco e em seguida fazer o roteamento e a entrega das mesmas ao consumidor.

Atualmente, o padrão estabelecido na área de MOM é o *Java Message Service* (JMS) [Sun 2002]. Ele fornece uma API e um conjunto de regras que especificam a semântica da entrega de mensagens em um ambiente de MOM tanto para o modo de entrega confiável quanto não confiável. A especificação JMS define regras para o comportamento operacional dos modelos de sistemas de mensagem (pub/sub e ponto-a-ponto); um conjunto rico e flexível de definições sobre o formato das mensagens; regras estritas para implementação do mecanismo de *store-and-forward*, garantia de entrega, mecanismos de reconhecimento, controle de transação e recuperação de falhas. JMS também provê um modelo de transação multi-recurso, integrando entidades externas a uma transação. A vantagem da especificação definir regras sobre o comportamento dos MOMs é que as aplicações podem assumir que estão operando com um sistema de mensagem independente de fabricante, garantindo a portabilidade do sistema.

Uma aplicação JMS é composta das seguintes partes: um JMS *Provider* que provê facilidades de controle e administração; JMS *Clients* são programas ou componentes que produzem ou consomem mensagens; *Messages* são objetos que comunicam informações entre JMS *Clients*; *Administered objects* são objetos JMS pré-configurados, criados pelo administrador para uso dos JMS *Clients*; e *Native Clients* são programas que utilizam uma API proprietária de um sistema de MOM específico em vez da API JMS.

2.2. Avaliação de Desempenho para Planejamento de Capacidade

Planejamento de capacidade é o processo de prever quando os níveis futuros de carga saturarão o sistema e determinar o modo mais econômico de adiar a saturação ao máximo [Menascé e Almeida 2002]. Um sistema é dito saturado quando um de seus recursos se aproxima de 100% de utilização, tornando-se um gargalo para o desempenho. O intuito do planejamento de capacidade é garantir que as demandas futuras de carga possam ser atendidas de maneira efetiva considerando-se os acordos de nível de serviço e as restrições tecnológicas e de custo que forem aplicáveis.

Em geral, o planejamento de capacidade compreende um conjunto de etapas que envolvem o conhecimento do ambiente, a caracterização e predição da carga de trabalho, a predição de desempenho do sistema, e a análise de custo.

Durante a etapa de conhecimento do ambiente devem ser obtidas informações acerca do sistema em análise, incluindo uma descrição do hardware e software envolvidos, dos acordos de níveis de serviço aplicáveis e dos padrões do uso atuais.

Normalmente, a coleta dos dados relativos aos padrões de uso é realizada através de técnicas de instrumentação e monitoração do sistema.

Com base nos dados coletados, é realizada uma caracterização da carga de trabalho que envolve a sua decomposição em componentes de acordo com a intensidade de carga representada por cada um e com a demanda destes por recursos. A conclusão desta etapa envolve a predição de cargas futuras, a qual é subsidiada por uma monitoração de longo prazo.

A predição de desempenho pode ser definida como o processo de estimar *métricas* de desempenho para um determinado conjunto de *parâmetros* [Menascé e Almeida 2002]. Tais parâmetros podem ser inerentes ao próprio sistema ou à carga a que este pode ser submetido. Os parâmetros de carga são determinados durante a etapa de caracterização mencionada anteriormente.

O conjunto de métricas selecionadas varia de acordo com a natureza do sistema e com os objetivos da própria predição. No contexto de avaliação de desempenho de um *Middleware* Orientados a Mensagem as seguintes métricas são normalmente consideradas relevantes: Tamanho de fila (número de mensagens que aguardam processamento); Latência (intervalo de tempo decorrido entre o envio de uma mensagem por um produtor e a sua recepção em um consumidor), Vazão do sistema (taxa na qual as mensagens são processadas pelo MOM); e utilização de um recurso (fração do tempo na qual este recurso está sendo utilizado para o processamento de requisições).

Em geral, a etapa de predição de desempenho envolve a construção e uso de modelos que são abstrações de sistemas reais. O nível de detalhe descrito em um modelo depende de sua finalidade. Do ponto de vista de desempenho, dois tipos de modelos podem ser construídos para um sistema: modelos de simulação e modelos analíticos.

Nos modelos de simulação, a descrição do sistema é embutida em um programa de computador que ao executar simula a sua operação. Nestes modelos, o estado do sistema é normalmente representado através de variáveis que assumem valores discretos. Ao longo de um experimento de simulação, as operações do sistema são simuladas repetidas vezes e as métricas configuradas são computadas. Um experimento de simulação deve continuar em execução até que o intervalo de confiança para a média de cada métrica configurada no modelo atinja uma dimensão especificada. As dimensões do modelo e do intervalo de confiança, além da quantidade de métricas coletadas, possuem um grande impacto em termos do tempo necessário para a finalização do experimento.

Nos modelos analíticos, a descrição do sistema é realizada através de um conjunto de formulações matemáticas que podem ser solucionadas para um conjunto de parâmetros de entrada, permitindo a derivação das métricas de desempenho. Para que um modelo analítico possa ser tratável é necessário que suas dimensões sejam pequenas, ou seja, que ele apresente uma visão do sistema em alto nível de abstração. Os modelos analíticos geralmente embutem uma menor quantidade de detalhes que os modelos de simulação. Em contrapartida, os modelos analíticos executam mais rapidamente e, quando solucionados, fornecem resultados exatos, sem a necessidade de especificação

de intervalos de confiança. É importante esclarecer que o termo exatidão se refere ao fato de que os resultados são soluções de sistemas de equações, não indicando que o modelo reflete, necessariamente, o sistema real sendo representado.

Finalmente, para que um modelo possa ser utilizado em experimentos de planejamento de capacidade, ele deve ser validado. Um modelo é considerado válido se as métricas de desempenho obtidas a partir dele se aproximarem suficientemente dos valores correspondentes medidos diretamente do sistema real.

2.3. Redes de Petri Estocásticas

As redes de Petri são uma família de ferramentas gráficas utilizadas para descrição formal de sistemas que possuem características como concorrência, conflito, sincronização e exclusão mútua. Uma rede de Petri é representada através de um multi-grafo bipartite e direcionado que possui dois tipos de nós: lugares e transições (representados por círculos e retângulos, respectivamente). Semanticamente, os lugares representam os recursos, estados locais ou variáveis do sistema, enquanto as transições representam as ações possíveis. Um arco em uma rede de Petri interliga um lugar à uma transição (arco de entrada) ou uma transição a um lugar (arco de saída). Os multi-grafos podem possuir mais de um arco interligando dois nós em uma mesma direção. Estes arcos podem ser representados através de um único arco rotulado com um número natural referenciado como multiplicidade.

Um modelo em redes de Petri possui, além dos elementos descritos acima, *tokens* que são marcas indistinguíveis entre si. Os *tokens* são denotados por pontos sólidos que residem nos lugares e trafegam pela rede através dos arcos. A marcação de um modelo em um dado instante, denotada por *M(p)*, é uma função que associa a cada lugar um número natural, representando a quantidade de *tokens* residindo naquele lugar. Cada marcação possível corresponde a um estado no qual o modelo pode se encontrar.

A execução de um modelo provoca alterações em sua marcação e é ocasionada pelo disparo das transições que são os componentes ativos. Para que uma transição possa disparar, é necessário que cada lugar de entrada desta transição possua pelo menos um número de *tokens* igual à multiplicidade do arco que o conecta à transição. Neste cenário diz-se que a transição está habilitada. O disparo de uma transição promove uma alteração na marcação corrente de um modelo removendo de cada lugar de entrada, o número de *tokens* indicado pela multiplicidade do arco que o conecta a transição, e criando, em cada lugar de saída, o número de *tokens* indicado pela multiplicidade do arco de saída que o liga à transição disparada. O disparo é uma operação atômica, de forma que os *tokens* são removidos dos lugares de entrada e criados nos lugares de saída em uma operação indivisível.

Diferentes modelos de redes de Petri propõem a incorporação do conceito de tempo através da associação de transições a atrasos determinísticos ou estocásticos, sendo os últimos bastante utilizados no contexto de avaliação de desempenho. Dentre os modelos estocásticos as redes *Generalised and Stochastic Petri Net* (GSPN) [Marsan et al. 1984] possuem um papel de destaque. Nestas redes, as transições podem ser de dois tipos: *imediatas*, que possuem tempo de disparo igual a zero e descrevem ações irrelevantes do ponto de vista de desempenho, e *exponenciais* que possuem tempos de

disparo representados através de variáveis aleatórias com distribuição exponencial. As transições imediatas possuem prioridade de disparo com relação às temporizadas.

Balbo [Balbo 2001] define GSPN como uma tupla (P,T,I,O,H,G,M0,W,Π), onde: P é o conjunto de lugares; T é o conjunto de transições; I, O e H são relações representando os arcos e descrevendo pré-condições, pós-condições e condições de inibição, respectivamente; G é uma função de habilitação que, dada uma transição imediata e uma marcação, determina se a transição pode ou não ser disparada; M0 representa a marcação inicial atribuindo a cada lugar um número natural; W é uma função que associa a cada transição um número real não-negativo representando o tempo médio de disparo, caso a transição seja exponencial, ou o peso, caso a transição seja imediata (o peso é um fator probabilístico considerado na escolha estocástica da próxima transição a ser disparada quando mais de uma transição imediata está habilitada); e Π associa a cada transição imediata um número natural que representa o seu nível de prioridade.

Cada transição exponencial tem uma semântica de disparo que pode ser *single-server*, *infinite-server* ou *multiple-server* (*k-server*). A semântica *single-server* simula a execução da transição em um único servidor, de forma que disparos simultâneos desta transição não são possíveis quando ela possuir grau de habilitação maior do que um. A semântica *infinite-server* simula a execução da transição quando se tem um número infinito de servidores. Desta forma, a execução de uma transição com grau de habilitação *n* ocorre sob a forma de *n* execuções em paralelo. Por fim, a semântica *multiple-server* simula a execução da transição quando se tem um número finito *k* de servidores disponíveis. Com esta semântica a execução de uma transição com grau de habilitação *n* ocorre através de seqüências de *k* execuções simultâneas.

2.4. Refinamento de GSPNs

Em geral, uma boa prática para representar variáveis aleatórias com distribuições empíricas é utilizando-se uma composição de exponenciais de forma que alguns momentos da distribuição empírica correspondam aos momentos desta composição. Tais composições são conhecidas como *Phase Type Distributions* (PHD) [Neuts 1975]. Algoritmos que fazem mapeamentos entre distribuições empíricas e PHD são conhecidos como algoritmos de *moment matching* (casamento de momentos).

Um algoritmo de casamento de momentos interessante foi proposto por [Watson e Desrochers 1991]. Este algoritmo tira proveito do fato de que distribuições de Erlang usualmente têm média maior que o desvio padrão, enquanto que distribuições hiperexponenciais geralmente possuem média menor que o desvio. Desta forma, o algoritmo propõe que uma atividade com duração determinada empiricamente seja modelada através de subredes que representem distribuições de Erlang ou hiperexponenciais, chamadas *s-transitions*. De acordo com o coeficiente de variação (razão entre desvio e média amostrais) associado à duração de uma atividade, uma implementação de *s-transition* (Erlangiana ou hiperexponencial) pode ser selecionada. Para cada implementação há parâmetros que podem ser configurados de forma que o primeiro e o segundo momentos associados à duração de uma atividade empírica casem com o primeiro e segundo momentos da *s-transition* como um todo.

A implementação da *s-transition* hiperexponencial possui três parâmetros: r1 e r2 que representam os pesos das transições imediatas utilizadas e λ que representa a taxa da componente exponencial da implementação. Para aproximar uma distribuição empírica com duração média μ e desvio padrão σ, onde $\mu<\sigma$, estes parâmetros devem ser configurados para $r1 = 2\mu^2/(\mu^2+\sigma^2)$, $r2 = 1 - r1$, e $\lambda = 2\mu/(\mu^2+\sigma^2)$, fazendo com que o primeiro e segundo momentos da *s-transition* casem com os momentos correspondentes medidos.

A implementação da *s-transition* Erlangiana possui três parâmetros: λ_1 e λ_2 que representam as taxas das transições exponenciais utilizadas na implementação da *s-transition*; e γ que é um inteiro que dá a multiplicidade do arco de entrada da transição imediata e representa o número de fases da distribuição de Erlang. Para aproximar uma distribuição empírica com duração média μ e desvio padrão σ, onde $\mu>\sigma$, estes parâmetros devem ser configurados de forma que: $(\mu/\sigma)^2-1 \leq \gamma < (\mu/\sigma)^2$, $\lambda_1=1/\mu_1$ e $\lambda_2=1/\mu_2$, onde:

$$\mu_1 = \frac{\mu \mp \sqrt{\gamma(\gamma+1)\sigma^2 - \gamma\mu^2}}{\gamma+1} \qquad \mu_2 = \frac{\gamma\mu \pm \sqrt{\gamma(\gamma+1)\sigma^2 - \gamma\mu^2}}{\gamma+1}.$$

3. Modelos de Desempenho para MOMs

Esta seção define e valida um modelo de desempenho para sistemas de MOMs compatível com o padrão JMS. A abordagem utilizada foi baseada em [Menascé et al. 2004].

O primeiro passo para a definição do modelo é o estabelecimento dos objetivos. Neste contexto, pode-se afirmar que a finalidade principal deste modelo é a identificação da capacidade máxima de entrega de mensagens de uma plataforma de MOM sem o comprometimento do nível de serviço especificado. Um segundo objetivo é a identificação do ponto de saturação do MOM. Estes objetivos são passos imprescindíveis para a realização do planejamento de capacidade.

Para atingir os objetivos, as métricas observadas incluem: tamanho e utilização da fila, vazão do sistema, e latência. Por questões práticas, não foram incluídas neste momento métricas que representam a execução incorreta do serviço, nem a não execução do serviço. Desta forma definimos métricas relativas a tempo, taxa e recurso.

O segundo passo é o entendimento do sistema a ser modelado e avaliado. Para facilitar esse entendimento é descrito um cenário típico da utilização de MOM para integração de aplicações. O cenário é caracterizado por um modelo de sistema ponto-a-ponto, com um número de produtores (*JMS Producer*) crescente, que estão todos conectados com uma única fila (*JMS Destination*). Na outra ponta do canal, existe apenas um único consumidor (*JMS Consumer*) recuperando e processando estas mensagens.

Como um dos objetivos do modelo é determinar a capacidade máxima de entrega do sistema de mensagem, o tempo de processamento da mensagem foi desprezado, fazendo com que o consumidor ao receber a mensagem rapidamente possa estar pronto para receber outra. Obviamente, o tempo de processamento da mensagem afetaria a avaliação do MOM, incluindo no processo uma variável (a aplicação) que não está sob análise.

O próximo passo está relacionado à caracterização da carga a ser submetida ao sistema para que o objetivo seja atingido. Neste caso, o teste de esforço [Menascé e Almeida 2002] foi a opção escolhida, devido ao fato de que este tipo de teste identifica o comportamento de um sistema em situações extremas, determinando assim sua capacidade máxima, ou ainda o seu comportamento depois de atingido o ponto de saturação.

Na carga submetida ao modelo, cada produtor é representado por um modelo de carga que atende a uma distribuição de Poisson com tempo médio entre a produção de mensagens de 0,1s (que representa uma taxa de 10 mensagens por segundo ou "mps"). Cada mensagem gerada possui 1024 bytes de conteúdo útil (*payload*).

O passo seguinte é a identificação dos parâmetros e a seleção dos fatores. Os parâmetros de sistema considerados foram: QoS, tipo de reconhecimento, número de produtores, número de destinatários, número de consumidores, tempo médio de entrega e o tempo médio de recepção. Destes parâmetros, os dois últimos representam variáveis aleatórias (VAs), pois não são determinados pelo administrador do MOM, e sim variam de acordo com cada implementação do *middleware* e do ambiente onde ele está instalado.

No cenário deste artigo, o QoS escolhido foi "*No Persistent No Transaction*" (NPNT), que significa um modelo de entrega sem persistência e sem transação. O QoS foi estabelecido para todas as mensagens postadas na fila (JMS *Destination*). A escolha deste QoS foi motivada apenas por questões práticas. O tipo de reconhecimento de mensagens (*ACK*) estabelecido foi o que a especificação JMS chama de *AUTO ACK*, de forma que cada mensagem recebida pelo consumidor é gerado automaticamente um *ACK* para o *Provider*. Os parâmetros número de destinatários (igual a um) e número de consumidores (também igual a um) já foram descritos e justificados anteriormente na descrição do cenário.

Os dois parâmetros que são VAs serão definidos mais adiante, na Seção 3.2. O motivo para essa definição tardia é que estes parâmetros representam o desempenho de um ambiente de MOM específico, e o objetivo agora é desenvolver um modelo que abstraia os ambientes e possam ser facilmente instanciados posteriormente. Por último, o parâmetro número de produtores foi escolhido para ser o fator da avaliação (será variado), pois para determinar capacidade de um sistema precisa-se variar a carga.

Os próximos passos são descritos nas subseções seguintes, sendo eles: definição de um modelo de referência, parametrização do modelo (instanciação para o ambiente real) e validação do modelo. Com isso, os objetivos dessa seção são atendidos: "definir e validar um modelo de desempenho para sistemas de MOM compatível com o padrão JMS". Posteriormente, na seção de resultados, este modelo é utilizado para cumprir os objetivos do artigo: "identificar a capacidade máxima de entrega de mensagens de uma plataforma de MOM sem o comprometimento do nível de serviço especificado", e "identificar o ponto de saturação do MOM".

3.1. Modelo de Referência

O processo de construção de um modelo de desempenho se inicia com a elaboração de um modelo de referência. A principal característica do modelo de referência é que ele pode ser utilizado em qualquer ambiente (instalação) de MOM, desde que este ambiente

possua um cenário similar ao proposto pelo modelo. Isto significa que para qualquer ambiente onde existam muitos produtores, um consumidor, o sistema seja ponto-a-ponto, o QoS seja NPNT e o tipo de reconhecimento seja *AUTO ACK*, este modelo poderá ser utilizado. Este cenário é bem típico em processos de integração de aplicações.

A construção do modelo de referência é realizada tendo como base a especificação JMS, uma vez que o modelo representará o comportamento deste tipo de MOM. Inicialmente, foram identificados três componentes importantes que precisam estar representados no modelo.

O primeiro é o *JMS Provider*, que representa o próprio MOM em si. O *provider* contém a fila e o seu gerenciador, sendo este último responsável por ler, escrever e remover itens da fila, mantendo a sincronização no acesso ao recurso compartilhado através de um *lock*. O segundo é o *JMS Producers*, que representa os produtores. E por último, o *JMS Consumer*, representando os consumidores.

A API JMS define duas operações básicas na interação com o *provider* que foram representadas no modelo (ver Figura 1) pelas transições: *JMS_send* e *JMS_receive*. Pode-se verificar analisando o modelo que estas transições estão sincronizadas por um *Lock* dentro do *provider*, garantindo a sincronização no acesso a fila. Estas transições foram definidas como imediatas, pois o tempo associado a cada uma dessa atividade está representado dentro do *provider*.

Desta forma, iniciando a descrição do modelo pelo *JMS Provider*, os seguintes lugares foram modelados: *Queue* representa a fila; *Lock* representa o mecanismo de sincronização para acesso a fila; *Sending* representa as mensagens que foram produzidas, enviadas para o MOM, mas ainda estão sendo gravadas na fila. Ou seja, o produtor ainda não foi liberado da chamada *JMS_send*; *Receiving* representa as mensagens que estão sendo enviadas para o consumidor, após este ter realizado a chamada *JMS_receive*; e *AckReceived* representa o recebimento do ACK, que é gerado automaticamente pelo consumidor após a mensagem ter sido recebida;

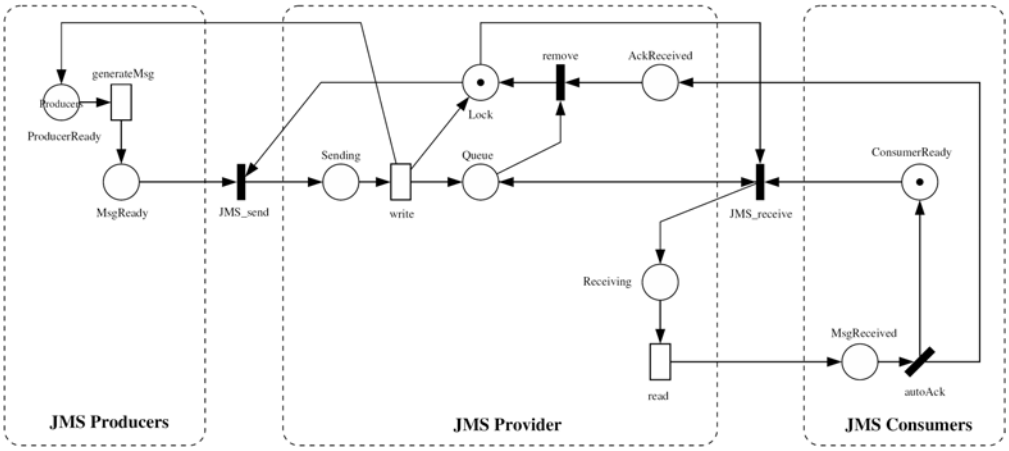

Figura 1. Modelo de Referência GSPN do MOM.

Além dos lugares, estão representadas no *provider* as suas principais atividades, que são modeladas pelas seguintes transições (imediatas ou temporizadas):

- *write*: representando a operação de escrita na fila. Após a execução desta atividade o produtor é liberado da chamada *JMS_send.* Esta transição é temporizada e o seu tempo corresponde ao tempo médio de entrega (parâmetro representando por uma VA). A semântica de disparo desta transição é *single-server*, devido ao fato que ela ocorre dentro do *provider* com uma única CPU;

- *read*: representando a operação de leitura da fila. Esta transição temporizada corresponde ao tempo médio de recepção (outro parâmetro representado por uma VA). Como se pode verificar no modelo, esta transição não libera o *Lock*, pois a liberação somente ocorrerá após a remoção da mensagem da fila. A semântica de disparo desta transição também é *single-server*;

- *remove*: representando a operação de remoção da mensagem da fila. Esta atividade foi considerada imediata, pois o tempo dela está embutido no tempo médio de recepção. Tal fato decorre da utilização de medições caixa-preta, inviabilizando a determinação deste tempo de forma isolada.

Dando seqüência ao detalhamento do modelo, o próximo componente descrito é o *JMS Producers*, que representa os produtores. Este componente foi modelado utilizando dois lugares e uma transição, como segue:

- *ProducersReady*: este lugar representa a quantidade de produtores que estão prontos para produzir mensagens. Ele é iniciado com um número de *tokens* igual ao parâmetro interno *Producers*. Este parâmetro será variado durante a realização dos experimentos;

- *generateMsg*: representando a atividade de gerar mensagens. Esta transição temporizada define a carga (mensagens por segundo, mps) que cada produtor está tentando gerar para o MOM. Neste ponto é que deve-se representar o modelo de carga adequado. No cenário utilizado neste trabalho, a carga foi definida seguindo uma distribuição de Poisson, com taxa de 10mps. É importante ressaltar que esta transição segue a semântica de disparo *infinite-server*, pois está representando a execução concorrente dos vários produtores;

- *MsgReady*: este lugar representa as mensagens geradas e que estão prontas para serem enviadas para o MOM.

O último componente a ser descrito é o *JMS Consumers*, que é modelado por dois lugares e uma transição:

- *MsgReceived*: este lugar representa as mensagens recebidas pelo consumidor, resultado da chamada *JMS_receive*;

- *autoACK*: esta transição representa a geração automática de um ACK para cada mensagem recebida. A recepção deste ACK pelo *provider* irá determinar a remoção da mensagem da fila. Esta transição foi modelada como imediata, pois o tempo desta atividade está embutido no tempo médio de recepção, pois a chamada *JMS_receive* só é liberada quando o ACK foi gerado;

- *ConsumersReady*: este lugar representa os consumidores que estão prontos para realizar um *JMS_receive*. Como o tempo de processamento de mensagens foi desconsiderado, os consumidores estão prontos para recuperar uma nova

mensagem tão logo realizem o reconhecimento da anterior. A presença de um único *token* representa a existência de um único consumidor.

Para finalizar a descrição do modelo GSPN, os atributos das transições imediatas estão definidos na Tabela 1.

Tabela 1. Atributos das transições imediatas do modelo de referência.

Transição	Peso	Prioridade	Função de Habilitação (guarda)
JMS_send	1	1	-
JMS_receive	1	1	-
remove	1	1	-
autoAck	1	1	-

É importante ressaltar que, no modelo proposto, o lugar *Queue* ficou ilimitado, podendo receber uma quantidade indefinida de *tokens*. Este fato decorre diretamente de um dos objetivos do artigo que é encontrar a capacidade máxima de entrega, levando a uma variação no tamanho da fila que não pode ser determinada previamente. A ausência da propriedade de limitação leva a um espaço de estados infinito, impedindo que o modelo possa ser solucionado analiticamente. Sendo assim, os experimentos realizados são baseados na técnica de simulação, em particular em simulações estacionárias, pois o que se pretende avaliar são métricas em estado estacionário.

Outro ponto a enfatizar é que o modelo de referência foi desenvolvido tendo como orientação a construção de um modelo bastante sintético, conseguindo com isso a avaliação caixa-preta da plataforma de MOM. Esta abordagem, além de levar a uma utilização mais racionalizada dos recursos computacionais, permite que o modelo represente qualquer implementação de MOM sem se considerar os detalhes específicos de implementação de cada produto. O único requisito é que o MOM seja compatível com o padrão JMS, pois foi a partir da especificação que o modelo foi elaborado.

3.2. Modelo Refinado

Esta seção apresenta a derivação de um modelo de desempenho específico (chamado modelo refinado) a partir do modelo de referência desenvolvido na seção anterior. No modelo de referência, as atividades relevantes do ponto de vista de desempenho são representadas através de transições temporizadas, as quais possuem um tempo de disparo definido por meio de uma variável aleatória. Em particular, duas atividades realizadas junto ao MOM foram consideradas relevantes, a leitura (*read*) e a escrita (*write*) de mensagens.

A transformação do modelo de referência no modelo refinado ocorre em uma etapa referenciada como parametrização. Nesta etapa, realizam-se experimentos de medição visando determinar, empiricamente, distribuições de probabilidades que aproximem satisfatoriamente o comportamento das variáveis aleatórias correspondentes aos tempos de disparo das transições temporizadas representadas no modelo.

O algoritmo apresentado na Seção 2.4 indica que uma representação adequada das atividades modeladas inicialmente pelas transições *read* e *write* envolve a substituição destas transições por sub-redes erlangianas ou hiperexponenciais (de acordo com o tempo médio e o desvio padrão medidos para cada uma destas atividades em um ambiente alvo). A medição de tais tempos foi realizada considerando-se o MOM como

uma caixa-preta. Esta característica simplifica a parametrização e torna o modelo de referência genérico e utilizável em qualquer ambiente de MOM compatível com o padrão JMS.

Na realização dos experimentos de medição foi utilizado um cliente desenvolvido em Java que acessa o MOM através da API JMS. Uma discussão mais detalhada deste cliente está fora do escopo deste artigo. Cada experimento de medição realizado teve uma duração de 20 minutos, sendo os dois primeiros considerados como tempo de *rampup* e o último como *rampdown*. Desta forma, o tempo útil considerado em cada experimento foi 17 minutos. A Tabela 2 apresenta média, desvio padrão e coeficiente de variação dos tempos medidos para cada atividade (*read* e *write*). Estes dados foram obtidos a partir do tratamento estatístico dos valores coletados durante as medições.

Tabela 2. Média (μ), desvio padrão (σ) e coeficiente de variação das VAs (transições temporizadas).

Transição	μ (ms)	σ (ms)	Coeficiente de variação
write	2,2031	4,9844	2,2624
read	1,2832	0,4107	0,3201

A partir dos coeficientes de variação, pode-se determinar a natureza das sub-redes a serem utilizadas na construção do modelo refinado. A Tabela 3 apresenta o valor do coeficiente de variação e a natureza das transições *read* e *write* (tipo da *s-transition*).

Por fim, a Figura 2 apresenta o modelo refinado obtido pela substituição das transições *read* e *write* do modelo de referência por sub-redes *s-transitions* Erlangiana e hiperexponencial, respectivamente.

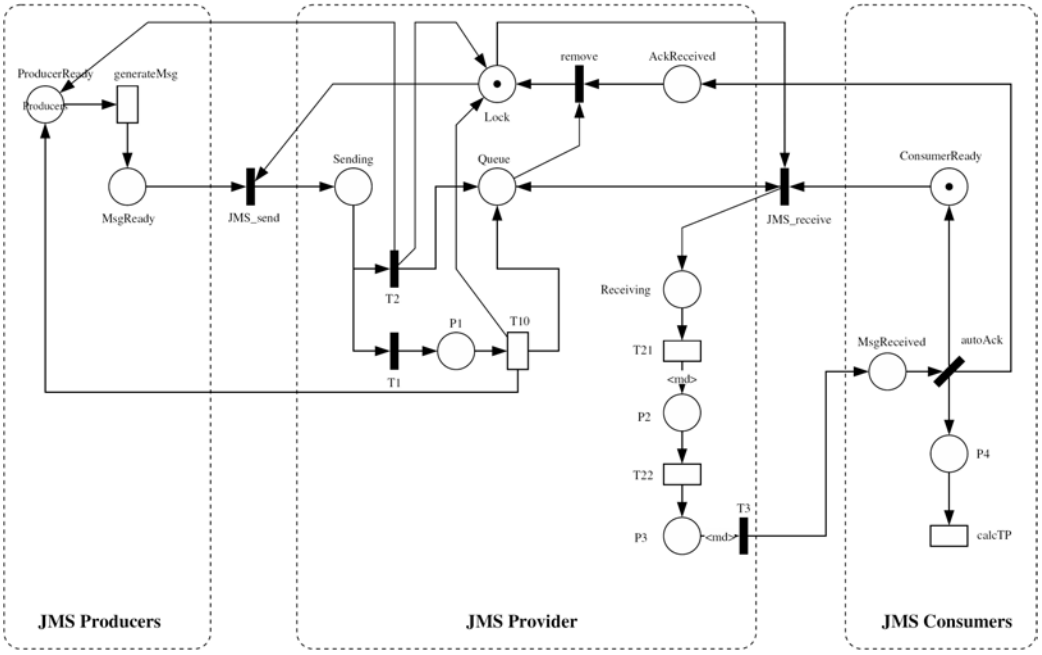

Figura 2. Modelo Refinado GSPN do MOM.

Os valores calculados para os parâmetros das sub-redes utilizadas no modelo refinado são: *write* (**r1** = 0,32687758, **r2** = 0,67312242 e **λ** = 148,370398) e *read* (**γ** =9, **λ1** = 0,000068017**, λ2** = 0,000135017).

Tabela 3. Coeficiente de variação das atividades temporizadas e identificação do tipo de s-transition adequado.

Transição	Coeficiente de variação	Tipo de *s-transition*
write	2,2624	Hiperexponencial
read	0,3201	Erlangiana

O próximo passo é a validação do modelo refinado. É importante ressaltar que todo o processo de validação é realizado comparando-se os resultados obtidos via simulação com os resultados obtidos a partir de medição realizada em ambiente real. Esta medição utiliza ferramentas específicas de geração artificial de carga [Apache 2005], e também acontece de forma não intrusiva (caixa-preta).

O tratamento estatístico dos dados obtidos da medição considerou a ocorrência de *outliers*[1] e utilizou o algoritmo de *Box Plot* [Turkey 1977] para encontrar os valores médios e desvio padrão do tamanho da fila. O processo de validação avaliou o tamanho da fila, pois o comportamento desta métrica é vital para a análise do comportamento do sistema como um todo. Logo, a hipótese considerada é que se o modelo refinado e a medição obtiverem resultados aproximados para o comportamento de crescimento da fila o modelo pode ser considerado validado.

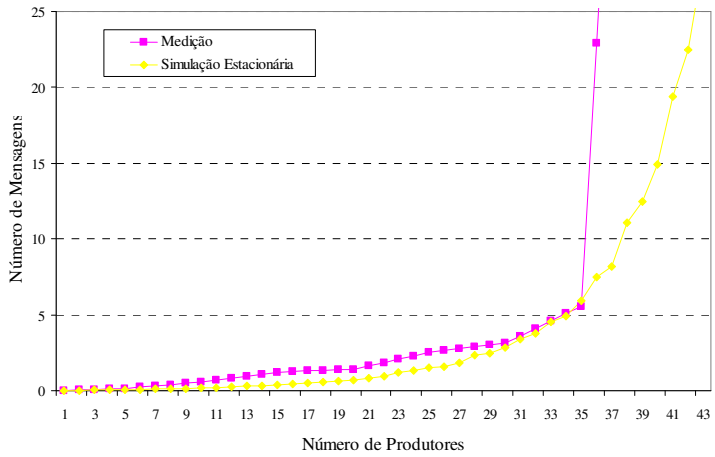

Figura 3. Validação do modelo refinado (métrica: tamanho da fila).

O gráfico apresentado na Figura 3 demonstra que os valores obtidos a partir da medição se aproximam significativamente daqueles obtidos via simulação estacionária do modelo refinado. Distorções mais relevantes somente são observadas com mais de 35 produtores, ponto a partir do qual o sistema começa a entrar em um estado de desequilíbrio, acumulando mensagens na fila. Uma possível explicação para esta distorção é a utilização de equipamentos voltados para o uso pessoal (PCs) nos ambientes nos quais foram realizados os experimentos de medição. Tais equipamentos

[1] Valores que apresentam distorções em relação ao comportamento geral, também chamados de pontos fora da curva.

normalmente apresentam uma grande degradação no desempenho quando operam próximo do seu ponto de saturação, diferentemente do poder de escalabilidade dos servidores. Vale salientar que a tendência de crescimento exponencial da fila se refletiu tanto nos experimentos de medição quanto nos de simulação.

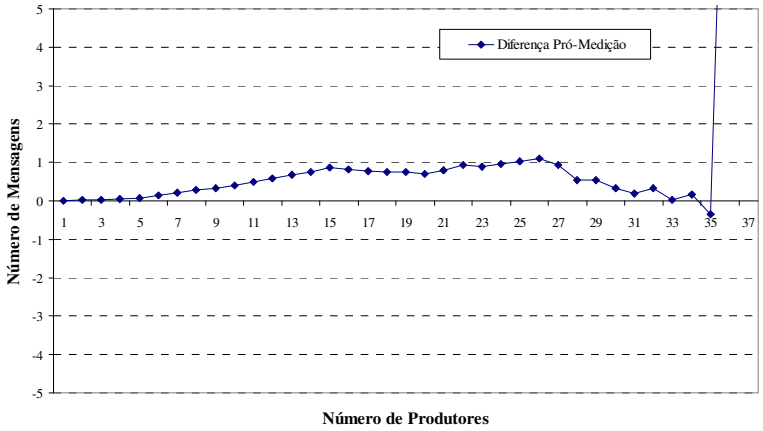

Figura 4. Análise do erro em número de mensagem (métrica: tamanho da fila).

Para reforçar a análise comparativa entre medição e simulação, o gráfico apresentado na Figura 4 demonstra o erro entre as duas técnicas. Pode-se notar que o erro até 35 produtores ficou inferior a uma mensagem, reforçando a aproximação das curvas mostras no gráfico anterior.

4. Resultados

Terminada a validação, o modelo encontra-se pronto para a realização de simulações com objetivo de avaliar as métricas selecionadas (tamanho e utilização da fila, vazão do sistema e latência) e determinar a capacidade máxima do sistema mantendo-se o nível de serviço estabelecido. É importante lembrar que esta etapa é o primeiro passo para o planejamento de capacidade. Outro ponto a ressaltar é que todas as avaliações mostradas nesta seção são resultados de experimentos de simulação estacionária com nível de confiança de 95% e erro máximo de 10% realizados utilizando a ferramenta TimeNET.

Para determinar a capacidade máxima mantendo o acordo de nível de serviço estabelecido (latência máxima de 10ms), precisa-se avaliar a latência do sistema à medida que se varia a carga (número de produtores). A Figura 5 mostra o comportamento da latência e da utilização da fila quando se varia a quantidade de produtores. Analisando o gráfico, pode-se perceber que a carga máxima para manter a latência inferior a 10ms, é de 29 produtores. Como esperado, o nível de utilização da fila cresce a com o aumento do número de produtores, sendo este aumento um fator determinante para o aumento da latência imposta pelo sistema para entrega das mensagens.

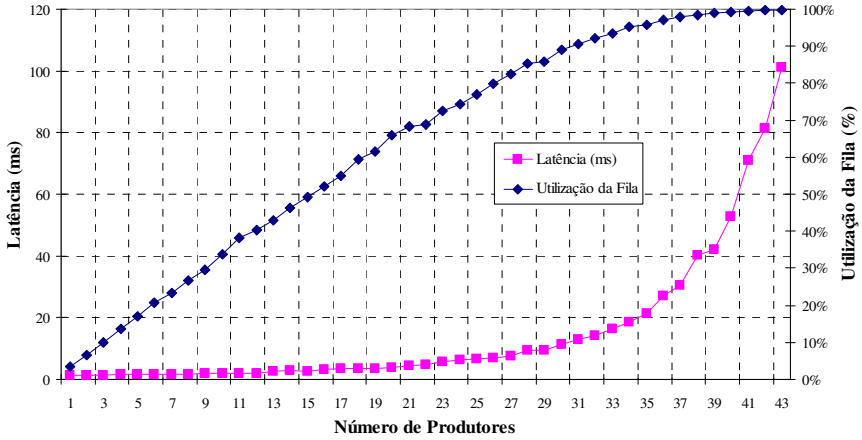

Figura 5. Comparativo entre a Latência e a Utilização da Fila.

A Figura 6 apresenta o comportamento do tamanho da fila e da vazão do sistema com variações na carga. Pode-se observar que a partir de 29 produtores, aumentos sucessivos de carga provocam pequenas alterações na vazão, mas alterações crescentes no tamanho da fila. Para esta carga, a capacidade entrega de mensagens é de aproximadamente 262mps e o tamanho médio esperado para a fila é de 2,47 mensagens.

Para finalizar, falta determinar o ponto de saturação do sistema. Para tanto, precisa ser identificada a saturação na utilização de algum recurso do MOM. O recurso a ser investigado é a fila, pois a competição pelo acesso exclusivo a ela será determinante para o comportamento de desempenho do sistema de mensagem. Quando o sistema fica saturado, tende ao desequilíbrio, fazendo com que o tamanho da fila cresça indefinidamente (logicamente, considerando a manutenção de uma carga constante). Este crescimento continua até exaurir os recursos do ambiente (memória, por exemplo).

Para analisar a saturação, foi gerado o gráfico mostrado na Figura 7, que apresenta a utilização da fila e vazão do sistema à medida que o número de produtores aumenta. Com base neste gráfico, pode-se observar que a partir de um nível de utilização da fila em torno de 97%, a vazão não apresenta variação significativa com aumentos sucessivos da carga, estabilizando-se próximo de seu valor máximo.

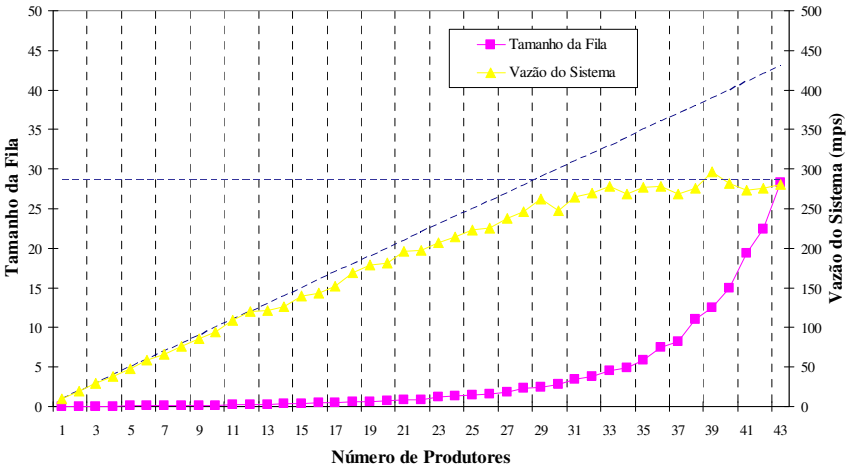

Figura 6. Comparativo entre a Vazão do Sistema e o Tamanho da Fila.

Por esse gráfico, pode-se constatar que a partir de 36 produtores a utilização da fila passa de 97% estando o ponto de saturação bem próximo. Este dado é reforçado pelo gráfico mostrado na Figura 3, que mostra a medição do sistema, onde observa-se um crescimento demasiado do tamanho da fila a partir de 35 produtores. A Figura 7 mostra, ainda, que na saturação a vazão do sistema é de cerca de 275mps.

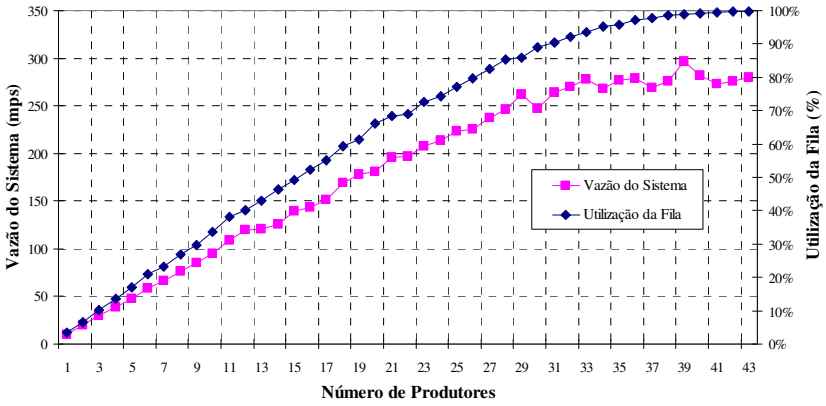

Figura 7. Comparativo entre a Vazão do Sistema e a Utilização da Fila.

Uma análise dos gráficos apresentados mostra que próximo ao ponto de saturação a expectativa é que a latência esteja em torno de 27ms e que o tamanho médio da fila seja superior a oito mensagens.

5. Trabalhos Relacionados

Existem na literatura várias iniciativas no sentido de avaliar desempenho de sistemas de MOM. Esses trabalhos utilizam técnicas de avaliação de desempenho de sistemas baseadas em medições ou na construção de modelos (resolução analítica ou simulação). Em [Tran et al. 2002], os autores apresentam uma avaliação de desempenho de MOM utilizando medição, com o foco no IBM MQSeries v5.2 em um ambiente de teste. Além disso, eles avaliam o impacto de alguns fatores, assim como o tamanho de *buffers* e o mecanismo de persistência, em uma variedade de cenários, para avaliar a vazão máxima sustentável (MST) alcançada pelos sistemas.

Alguns relatórios técnicos produzidos pelo mercado trazem também uma contribuição para a avaliação de desempenho de MOMs. Em [Sonic 2003], a Sonic apresenta uma comparação de MOMs JMS quanto à capacidade de entrega de mensagens. Neste trabalho, os autores realizam medições em ambiente de teste, variando o mecanismo de persistência e a carga submetida ao sistema. Como continuidade dessa iniciativa a Sonic propôs em [Sonic 2004] a definição de um processo sistemático e estruturado de *benchmark* para MOMs. Este trabalho apresenta cenários típicos de utilização de MOMs, além de uma abordagem objetiva e direta para condução do processo, descrevendo os principais fatores a serem analisados. O CSRIO apresentou um relatório independente [CSRIO 2001] avaliando três dos principais competidores no mercado de MOMs: Microsoft Message Queue, IBM MQSeries e TIBCO Rendezvous. Este relatório realiza medições dos sistemas submetendo-os a vários cenários. O objetivo principal é identificar os pontos fortes e fracos de cada

plataforma fazendo com que a escolha do sistema mais adequado dependa da determinação de quais destes pontos são mais relevantes em cada situação específica.

Em [Liu et al. 2003], os autores propõem um modelo para avaliação de desempenho de um Middleware para Integração de Processos de Negócio. Este modelo é baseado em *Layered Queuing Networks* (LQN) e os resultados são validados com medições obtidas utilizando o IBM *CrossWorlds InterChange Server*. [Fernandes et al. 2004] apresenta uma avaliação do IBM WebSphere MQ v5.3 realizada através de modelos desenvolvidos em redes de Petri estocásticas. Os resultados foram bastante satisfatórios fazendo com que o modelo obtivesse um bom nível de precisão nos resultados. [Liu e Gorton 2005] apresenta uma abordagem para predizer, durante a fase de projeto, o desempenho de aplicações baseada em componentes que efetuam comunicação síncrona e assíncrona. Para isso, os autores propõem um modelo analítico que captura o comportamento de desempenho das aplicações J2EE que utilizam a API JMS, predizendo o desempenho de três atributos de QoS diferentes no MOM. [Menascé 2005] propõe um *framework* quantitativo para comparar soluções baseadas em MOM e RPC. O objetivo do autor, através do modelo matemático, é realizar uma avaliação comparativa da arquitetura de comunicação (síncrona ou assíncrona) das aplicações.

Os trabalhos que utilizam medições ressaltam a dificuldade imposta pelo método de avaliação para preparar o ambiente de teste e tratar os resultados, além, obviamente, do custo de se manter réplicas da infra-estrutura de produção quando se pretende realizar testes de *stress* para identificar pontos de saturação do sistema [Menascé e Almeida 2002]. Os trabalhos que apresentam modelos analíticos ou de simulações utilizam, em sua grande maioria, ou modelos analíticos puramente matemáticos, ou modelos que representam um sistema de MOM específico, ou focam na avaliação de uma métrica apenas, ou ainda utilizam um processo de parametrização (calibragem) do modelo a partir de técnicas invasivas (instrumentação de código fonte).

Este trabalho possui alguns diferenciais que ressaltam a sua relevância. O primeiro é o fato de serem utilizados modelos no processo de avaliação de desempenho, permitindo a realização de simulações, que tornam a tarefa muito menos custosa. O segundo ponto é a utilização do formalismo das redes de Petri estocásticas, que permite a captura de aspectos dinâmicos (comportamentais) do sistema através de uma abordagem visual, permitindo a avaliação de várias métricas. Outro ponto importante é o fato do processo de parametrização dos modelos ter sido realizado através de medições caixa-preta (não intrusivas), o que permite a reutilização deste modelo em outras implementações de MOMs. Por último, e não menos importante, é o fato de serem utilizadas no modelo as primitivas básicas da API JMS, o que o torna capaz de representar praticamente qualquer sistema de MOM atualmente no mercado.

6. Conclusão

Este artigo investigou a utilização de modelos de desempenho para MOMs, desenvolvidos em redes de Petri estocásticas, no auxílio ao planejamento de capacidade deste tipo de *middleware*. Para isso, são utilizados modelos que identificam a capacidade máxima de entrega de mensagens mantendo um nível de serviço acordado, e o ponto de saturação do sistema de mensagem.

Os resultados obtidos da simulação do modelo GSPN se aproximaram de forma significativa dos resultados obtidos através da medição, validando ambos, a abordagem e o modelo. Como o processo de medição possui um alto custo associado, utilizar modelos para avaliar objetivos de desempenho é uma importante contribuição desse artigo. Outro importante aspecto é o desenvolvimento de um modelo de referência, o qual pode ser parametrizado para qualquer ambiente de MOM, tornando o modelo reusável.

Apesar da relevância dos resultados obtidos, o modelo proposto possui algumas limitações. Inicialmente, ele representa apenas um MOM ponto-a-ponto, com um modelo específico de QoS (NPNT). Outra limitação é que o modelo não consegue ainda representar o comportamento do sistema pós-saturação, gerando algumas distorções. Por último, o modelo representa apenas um destinatário (fila).

Alguns trabalhos futuros são identificados. Primeiramente, propõe-se a extensão do modelo apresentado para suportar cenários *publisher/subscribe.* A relevância deste novo modelo é permitir a representação mais completa de barramentos de serviço (ESB). Outra importante extensão é a introdução do suporte aos serviços de persistência e transação. Finalmente, outra contribuição futura é a automação da geração de modelos refinados a partir do modelo de referência, representando um ganho de escala no processo de avaliação.

7. Referências

Apache (2005) “Apache JMeter version 2.1.1”, http://jakarta.apache.org/jmeter.

Balbo, G. (2001) “Introduction to Stochastic Petri Nets”, In Lectures on Formal Methods and Performance Analysis, Vol. 2090 of LNCS, pp. 88-155.

Banavar, G., Chandra, T., Strom, R., Sturman, D. (1999) “A Case for Message-Oriented Middleware”. In Proceedings of the 13th International Symposium on Distributed System, LNCS 1693, p. 1-18.

Chappell, D. (2004) “Enterprise Service Bus”. ISBN 0-596-00675-6, O'Reilly.

CSRIO (2001) “Performance Evaluation of Message-Oriented Middleware Technology”, version 1.01, CSIRO Middleware Technology Evaluation Series.

Fernandes, S., Silva, W., Silva, M, Rosa, N., Maciel, P., Sadok, D. (2004) “Performance Analysis of Message-Oriented Middleware Using Stochastic Petri Nets”, In: 22nd Brazilian Symposium on Computer Networks (SBRC 2004), Anais do 22º Simpósio Brasileiro de Redes de Computadores.

Gartner Inc. (2003) “Predicts 2003: Enterprise Service Buses Emerge (DF-18-7304)”. http://www.gartner.com.

Gartner Inc. (2004) “Predicts 2004: Enterprise Service Buses Are Taking Off”. http://www.gartner.com.

IDC (2003a) “The Enterprise Service Bus: Disruptive Technology for Software Infrastructure Solutions” (Document #29132). http://www.idc.com.

IDC (2003b) “Integration Standards Trends in Program Development: It All Depends on What the Meaning of Open Is” (Document #30365). http://www.idc.com/.

Jain, R. (1991) "The Art of Computer Systems Performance Evaluation", Wiley Computer Publishing.

Linthicum, D. (1999) "Enterprise Application Integration", ISBN 0-201-61583-5, Addison Wesley.

Liu, T., Behroozi, A., Kumaran S. (2003) "A Performance Model for a Bussiness Process Integration Middleware", Proceedings of the IEEE International Conference on E-Commerce (CEC'03).

Liu, Y., Gorton, I (2005) "Performance Prediction of J2EE Applications using Messaging Protocols", LNCS Component-Based Software Engineering (CBSE 2005), ISBN 978-3-540-25877-3, Springer Berlin / Heidelberg.

Marsan, M., Balbo, G. and Conte, G. (1984) "A class of generalized stochastic Petri nets for the performance evaluation of multiprocessor systems", ACM Transactions on Computer Systems, Vol.2, No.2, p.93-122.

Menascé, D. (2005) "MOM vs. RPC: Communication Models for Distributed Applications". In IEEE Internet Computing, IEEE Computer Society.

Menascé, D., Almeida, V. (2002) "Planejamento de capacidade para serviços na Web: métrica, modelos e métodos". ISBN 85-352-1102-0, Campus.

Menascé, D., Almeida, V., Dowdy, L. (2004) "Performance by Design: Computer Capacity Planning by Example". ISBN 0-13-090673-5, Prentice Hall PTR.

Neuts ,M. F. (1975) "Probability distributions of phase type." In: Liber Amicorum Professor Emeritus H. Florin, University of Louvain, Belgium, p. 173-206.

Sonic Software Corporation (2003) "JMS Performance Comparison: Publish/Subscribe Messaging - SonicMQ® vs. TIBCO Enterprise™ for JMS", http://www.sonicsoftware.com.

Sonic Software Corporation (2004) "Benchmarking e-Business Messaging Providers", http://www.sonicsoftware.com.

Sun Microsystems, Inc (2002) "Java Message Service Specification - version 1.1". http://java.sun.com/products/jms/docs.html.

Tran, P., Greenfield, P., Gorton, I. (2002) "Proceedings of the 22nd International Conference on Distributed Computing Systems Workshops (ICDCSW'02)". IEEE.

Tukey, J. (1977) "Exploratory Data Analysis". Addison-Wesley, Reading, MA.

Watson III, J. and Desrochers, A. (1991) "Applying Generalized Stochastic Petri Nets to Manufacturing Systems Containing Nonexponential Transition Functions", IEEE Transactions on Systems, MAN, and Cybernetics, Vol.21, No.5.