1
Introdução

É cada vez maior o controle de atividades cotidianas pelos sistemas computacionais (Northrop et al., 2006). Aplicações recentes lidam frequentemente com distribuição, heterogeneidade, mobilidade, embarcação em dispositivos diversos e alta disponibilidade. Nesse contexto, os sistemas computacionais são cada mais exigidos em termos de funcionalidades complexas e garantias de qualidade de serviço. Uma consequência direta desse fato é que o processo de desenvolvimento de software tem se tornado mais difícil e dispendioso. Para reduzir o custo do desenvolvimento de aplicações, a indústria de software tem feito uso extensivo de tecnologias de *middleware* como CORBA (OMG, 2011), J2EE (Oracle, 2011) e .NET (Microsoft, 2011).

Tecnologias de *middleware* são usadas com o intuito de prover reusabilidade de serviços entre diferentes aplicações. A abordagem de desenvolvimento de sistemas usando *middleware* consiste em separar claramente a lógica do negócio, específica de cada aplicação, dos serviços comuns a diversos sistemas (Campbell et al., 1999). Esses serviços são comumente denominados de infraestrutura e incluem, dentre outras, funções de conectividade, segurança e persistência.

Em geral, a separação das funcionalidades de uma aplicação em lógica do negócio e serviços de infraestrutura permite que esses últimos possam ser desconsiderados durante a construção da aplicação. Desenvolvedores podem, então, concentrar-se no núcleo do negócio e ignorar detalhes de infraestrutura. Os serviços de infraestrutura, por sua vez, são interligados à lógica da aplicação apenas em tempo de execução.

Enquanto tal abordagem reduz a complexidade do desenvolvimento de aplicações, especialmente as distribuídas, o comportamento completo do sistema emerge apenas em tempo de execução (Diaconescu, 2006). Tradicionalmente, tecnologias de *middleware* não fornecem apoio para gerenciar comportamentos emergentes, ou seja, comportamentos inesperados que surgem da composição de comportamentos bem conhecidos (Mogul, 2006). Como consequência, prever ou mesmo fazer suposições sobre características relacionadas à qualidade de serviço de aplicações baseadas em *middleware* pode ser uma

tarefa não trivial.

Para ilustrar esse fato, considere, por exemplo, características de desempenho. Segundo Cutlip (Cutlip, 2005), a indústria de TI considera esse requisito uma das propriedades mais importantes relacionadas à qualidade de serviço, referindo-se ao ato da entrega de produtos de valor em tempo e custo hábeis. Diversos trabalhos reconhecem que o uso de tecnologias de *middleware* torna ainda mais difícil o problema da caracterização do desempenho de aplicações (Williams e Smith, 1998; Gorton e Liu, 2002; Aguilera et al., 2003; Diaconescu et al., 2004; Chen et al., 2005). Em geral, esses trabalhos apresentam dois fatores como os principais motivadores de tal dificuldade: *i*) o forte acoplamento entre a lógica de negócio e os serviços de infraestrutura, o que dificulta a mensuração individual do desempenho do módulo de negócio; e *ii*) a dificuldade de aferir a sobrecarga imposta pela infraestrutura, uma vez que essa sobrecarga depende da implementação provida por cada tecnologia de *middleware*. Portanto, em aplicações baseadas em *middleware*, o desempenho da aplicação é determinado não só pelo comportamento dos módulos de negócio, mas também pelo comportamento dos serviços de infraestrutura em uma implementação particular.

Apesar dessas dificuldades, a garantia de desempenho é uma questão crucial para muitas aplicações. Portanto, muitas abordagens têm sido propostas com o intuito de ajudar a compreender melhor o desempenho de aplicações baseadas em *middleware*. Dentre essas abordagens, incluem-se métodos tradicionais de testes e criação de protótipos, bem como expressões assertivas no código da aplicação (Deelman et al., 1998; Kvasnicka et al., 2001; Wilmarth et al., 2005). Embora importantes, esses procedimentos são estáticos e definidos em tempo de projeto e, portanto, são incapazes de prover garantias de desempenho para aplicações que apresentam comportamentos que emergem em tempo de execução.

Há também um número considerável de trabalhos que exploram instrumentação passiva para desenvolver modelos que capturam a dinâmica das aplicações em tempo de execução. Nesse contexto, muito trabalho foi feito em modelagem analítica de desempenho de sistemas, onde *modelos de fila* (Chen et al., 2005; Urgaonkar et al., 2005; Zhang et al., 2008), *teoria de controle* (Kusic e Kandasamy, 2007) e *redes de Petri* (Dingle et al., 2002) têm sido aplicados para caracterizar o comportamento de sistemas de grande escala[1], muitos desses construídos a partir de tecnologias de *middleware*. No entanto, modelos analíticos podem ser de difícil derivação para alguns domínios. Além disso, o processo de modelagem analítica pode requerer um esforço considerável de co-

[1]Do termo inglês *enterprise systems*.

nhecimento *a priori*, tornando os modelos sujeitos a erros ou suposições irreais.

Por outro lado, a recente explosão de ferramentas de coleta de dados e monitoramento, tais como Astrolabe (Renesse et al., 2003), Tivoli (IBM, 2010), HP Openview (HP, 2010) e Ganglia (Massie, 2010), propiciou uma grande quantidade de dados de medição associados a diversos componentes dos sistemas. Em particular, esse dados incluem métricas coletadas no nível da aplicação, como tempo de resposta de serviços ou transações, vazão, etc., e no nível do sistema, como utilização de CPU, memória, disco e rede. A complexidade dos sistemas, a qual torna inviável o uso de abordagens estáticas para a construção de modelos de comportamento, e a disponibilidade de dados de instrumentação incentivaram o uso do aprendizado de máquina para induzir automaticamente o modelo de desempenho das aplicações. Diversas técnicas têm sido propostas com esse intuito, incluindo métodos da teoria do aprendizado estatístico (Cohen et al., 2004; Powers et al., 2005; Barham et al., 2004), redes neurais artificiais (Senger et al., 2006; Senger et al., 2004; Sarioglu et al., 2006), computação bio-inspirada (Chakravarti et al., 2005; Miorandi et al., 2008) e aprendizado por reforço (Wu et al., 2011; Tesauro et al., 2006). Ao contrário de modelos analíticos, abordagens baseadas em aprendizado de máquina assumem pouco ou nenhum conhecimento *a priori* do domínio da aplicação, sendo, portanto, genéricas. Além disso, muitas dessas abordagens possuem um grande potencial de adaptabilidade, podendo ajustar-se a mudanças nos sistemas e no ambiente de execução.

De fato, é amplamente reconhecido que a complexidade dos sistemas atuais supera a habilidade humana em diagnosticar e responder prontamente e corretamente os problemas de desempenho, configuração, segurança ou proteção que ocorrem. Portanto, há uma grande necessidade de transferir essas atividades para os sistemas computacionais e minimizar a intervenção humana na execução dessas tarefas. Tais sistemas, capazes de gerenciar a si próprio, são denominados sistemas autonômicos (Kephart e Chess, 2003). Técnicas baseadas em aprendizado de máquina desempenham um papel fundamental na implementação desses sistemas.

Apesar de todo o esforço empreendido recentemente na aplicação de técnicas de aprendizado de máquina para o gerenciamento de sistemas e a contextualização de problemas, o uso dessas técnicas, nesse domínio, encontra-se ainda em estágio inicial. Há, portanto, um espaço considerável para contribuições. A grande maioria dos trabalhos que abordam esse tema o faz exclusivamente sob a perspectiva da previsão das necessidades de recursos de uma aplicação (Powers et al., 2005; Zhang e Bivens, 2007; Andrzejak e Silva, 2008). Poucos trabalhos têm concentrado esforços em outras perspectivas também

importantes para a caracterização de problemas de desempenho, como, por exemplo, o diagnóstico das causas de um problema. Há também uma grande necessidade de estudos comparativos, em que a adequação de diferentes métodos de aprendizado de máquina possa ser avaliada especificamente no contexto de gerenciamento autonômico de sistemas (Zhang e Bivens, 2007). Finalmente, devido à pouca maturidade das soluções de autogerenciamento, é importante ressaltar a relevância de trabalhos experimentais que possam contribuir com a validação dessas ideias. Esses são os problemas tratados nesta tese.

1.1 Objetivos, Abordagem e Contribuições

Neste trabalho, é apresentado um estudo experimental do uso de abordagens estatísticas para caracterizar problemas de desempenho em aplicações baseadas em *middleware*. Particularmente, o estudo é estruturado de forma a responder quatro questões principais:

- $\mathbf{Q_1}$: Considerando um conjunto corrente de métricas observadas, quais métodos da teoria do aprendizado estatístico[2] são mais apropriadas para prever ou estimar quando um sistema, ou seja, uma aplicação baseada em *middleware*, não conseguirá atingir seu objetivo num futuro próximo?
- $\mathbf{Q_2}$: Preditores induzidos a partir de métodos de aprendizado estatístico são passíveis de emitir alarmes falsos. Como tornar o processo de previsão mais robusto a falhas transientes dos preditores?
- $\mathbf{Q_3}$: Quando um preditor estima um problema de desempenho, como determinar as causas do problema?
- $\mathbf{Q_4}$: Considerando o monitoramento das aplicações um processo constante, os dados de instrumentação constituem um fluxo contínuo e sequencial. Dados os métodos de aprendizado estatístico que melhor respondem a questão $\mathbf{Q_1}$, quão apropriadas são as versões *online* desses métodos para caracterizar problemas de desempenho nesse novo contexto?

As questões de $\mathbf{Q_1}$ a $\mathbf{Q_3}$ permitem analisar o problema em termos de três perspectivas que são relevantes para qualquer ferramenta autonômica de gerenciamento de desempenho. A primeira é a perspectiva pertinente à previsão de problemas de desempenho, ou seja, aquela que permite antecipar que um problema de desempenho ocorrerá em um futuro próximo. Prever

[2]Na teoria do aprendizado estatístico, um problema de aprendizado é formulado como um problema de estimação de funções (Vapnik, 1998).

períodos de degradação de desempenho é essencial para a alocação racional dos recursos computacionais e um melhor escalonamento das tarefas a serem executadas. A segunda perspectiva refere-se à redução da emissão de alarmes falsos, ou seja, a detecção inoportuna de problemas de desempenho em períodos de baixa utilização dos recursos computacionais. A redução de detecções inoportunas possibilita maior robustez e confiabilidade ao processo de previsão. Um mecanismo de previsão robusto é essencial para uma ferramenta de gerenciamento, pois ele aumenta a confiança depositada pelo administrador do sistema na ferramenta utilizada. Finalmente, a terceira perspectiva é pertinente à identificação das causas de um problema de desempenho. A determinação da natureza de um problema de desempenho é importante para definir a ação a ser tomada como reparação ao problema identificado.

Uma questão que surge com o uso de técnicas de aprendizado estatístico para o gerenciamento de aplicações baseadas em *middleware* é o dinamismo dos ambientes em que essas aplicações executam. Esse dinamismo exige que os sistemas e, consequentemente, os modelos que descrevem seu comportamento se adaptem continuamente a novas situações. Nessas condições, aplicações baseadas em *middleware* possuem várias características para as quais um processo de adaptação de modelos em tempo real pode ser extremamente benéfico. No entanto, a grande maioria dos métodos de aprendizado estatístico são implementados para cenários de aprendizado *batch* ou *offline*, onde todos os dados de treinamento são passados para o algoritmo de aprendizado ao mesmo tempo e o modelo induzido não é mais revisado com novos dados. Além disso, para induzir um modelo, alguns métodos podem exigir um grande consumo de recursos computacionais e um tempo de processamento elevado, podendo inviabilizar o uso desses métodos. A despeito dessas dificuldades, a questão $\mathbf{Q_1}$ é formulada, neste trabalho, com o intuito de aferir a adequação de diferentes classes de algoritmos de aprendizado estatístico clássicos, ou seja, que operam em ambientes de aprendizado *offline*, na estimação de problemas de desempenho em aplicações baseadas em *middleware*. Duas razões justificam essa investigação: os algoritmos de aprendizado *offline* ainda são os mais usados e citados na literatura, sendo, portanto, importantes para fins de comparabilidade; e a necessidade de medir os limites das técnicas *offline* para o problema proposto.

Por outro lado, é importante caracterizar também a adequação e aplicabilidade de algoritmos de aprendizado estatístico que operam em cenários de aprendizado *online*. Dawid (Dawid, 1984) define um cenário de aprendizado *online* como um processo em que os dados de treinamento chegam ao sistema de aprendizado de forma sequencial e não ao mesmo tempo. Nessas condições,

o modelo atual é usado primeiramente para aferir conhecimento, sendo atualizado com os novos dados posteriormente. Neste trabalho, a questão $\mathbf{Q_4}$ é formulada para avaliar o comportamento de uma versão *online* de um algoritmo clássico de aprendizado estatístico.

Para responder as questões acima, este trabalho adotou a seguinte abordagem. Uma arquitetura de gerenciamento de problemas de desempenho, a qual denominou-se SMART (Correa e Cerqueira, 2010), foi implementada. No SMART, desempenho é definido em termos de objetivos de nível de serviço[3] ou SLO. Sturm et al. (Sturm et al., 2000) definem SLO como a medida de desempenho acordada entre os clientes e o provedor de um serviço. Exemplos de SLO incluem limiares definidos sobre medidas como disponibilidade, frequência, vazão e tempo de resposta. Particularmente, no SMART, o SLO é definido como um limiar sobre o tempo de resposta de um serviço. Uma vez definido esse limiar, a caracterização de um problema de desempenho é feita sob três perspectivas: previsão, robustez e diagnóstico. Na perspectiva de previsão, o problema de estimação de degradação de desempenho é formulado como um problema de classificação. Em Estatística, classificação consiste no problema de identificar a subpopulação a qual pertence uma nova observação, tendo como base um conjunto de treinamento contendo observações cujas subpopulações são conhecidas (Gnanadesikan, 1977). No problema tratado neste trabalho, uma observação é um conjunto de métricas de desempenho e os valores observados para essas métricas em um momento particular. Portanto, partindo de um *log* de observações do sistema em operação, em que cada observação é acompanhada de uma indicação positiva ou negativa de degradação de desempenho, um modelo é induzido a fim de estimar se uma observação particular correlaciona-se ou não com um problema de desempenho. Para aumentar a acurácia do modelo, a saída desse classificador é acoplada a um teste estatístico[4] para detecção de tendência. Dessa forma, a saída final do processo de estimação é feita não com base em um resultado imediato do classificador, mas numa análise de tendência em uma série de saídas computadas ao longo do tempo. Obtém-se, assim, um mecanismo de previsão mais robusto a falhas transientes do classificador. Por outro lado, testes estatísticos também são a abordagem utilizada para diagnosticar a causa de um problema de desempenho. Nesse contexto, dois tipos de testes estatísticos são avaliados: teste para detecção de tendência em séries temporais e teste de independência. O primeiro tipo de teste é aplicado sobre os valores observados para uma métrica de desempenho particular, com o intuito de descobrir padrões de crescimento.

[3]Do termo inglês *Service Level Objectives*

[4]Em geral, testes estatísticos estimam as diferenças entre duas populações (NIST/SEMATECH, 2011).

Considerando que uma métrica de desempenho geralmente quantifica a intensidade de utilização de um recurso específico, a observação de um padrão de valores crescentes para essa métrica é um forte indicativo de grande utilização de recurso e, portanto, degradação de desempenho. O segundo tipo de teste é aplicado para aferir o quanto duas métricas são dependentes entre si. Tomando a indicação de degradação de desempenho como uma das métricas do modelo, é possível, através de testes desse tipo, mensurar o nível de dependência entre essa métrica e qualquer outra do modelo.

Para realizar a coleta das métricas usadas na análise e modelagem de problemas de desempenho, um cenário de referência foi definido. Nesse cenário, uma aplicação MapReduce (Dean e Ghemawat, 2004) é executada no topo do SCS (Tecgraf/PUC-Rio, 2010), um *middleware* baseado em componentes. Sistemas baseados em componentes de software permitem uma enormidade de combinações de componentes tanto de aplicação quanto de infraestrutura. Tipicamente, o conjunto completo dessas combinações não pode ser previsto antecipadamente. Por esse motivo, problemas de gerenciamento de desempenho em sistemas baseados em componentes são, muitas vezes, maiores que em outros tipos de *middleware*. Adicionalmente, aplicações MapReduce são inerentemente distribuídas e tendem a ser de longa duração. Essas características tornam o problema de gerenciamento de desempenho mais complexo, pois o ambiente de execução e a necessidade de recursos da aplicação variam ao longo do tempo. Dessa forma, é razoável considerar que o cenário proposto irá lidar com diferentes condições de carga, ambiente de execução e configuração que impactam diretamente no desempenho das aplicações. No entanto, é importante destacar que o estudo realizado neste trabalho depende muito pouco de características específicas de tecnologias de componentes. Portanto, os resultados obtidos neste trabalho podem ser generalizados para outras tecnologias de *middleware*. O estudo experimental realizado sobre o cenário de referência permitiu as seguintes contribuições como parte deste trabalho:

- Uma avaliação sistemática de diversos algoritmos de aprendizado estatístico aplicados na previsão de problemas de desempenho em sistemas baseados em *middleware*. Nessa avaliação, são estabelecidas as habilidades das principais classes de classificadores ao prover um mecanismo *online* para capturar o comportamento do desempenho de sistemas baseados em *middleware*. Tais habilidades incluem acurácia, tempo requerido para treinamento, tempo requerido para classificação, sensibilidade ao tamanho da amostra de treinamento, dentre outras.
- O projeto e validação de um mecanismo de análise de desempenho mais robusto a alarmes falsos. Este trabalho apresenta um algoritmo

de alerta para problemas de desempenho que aumenta o poder de predição das técnicas de aprendizado estatístico, combinando-as com testes estatísticos para detecção de tendências em séries temporais. A combinação dessas duas técnicas para aumentar a robustez do processo de previsão em relação a falhas transientes dos classificadores é uma proposta inovadora, não tendo sido encontrado nenhum trabalho na literatura que propusesse algo semelhante.

– Uma avaliação da aplicabilidade de testes estatísticos na identificação das causas de um problema de desempenho. Testes estatísticos são simples e podem ser computados de forma eficiente, além de proverem um mecanismo de diagnóstico independente do algoritmo de aprendizado utilizado no processo de previsão. Isso é especialmente importante pois muitas técnicas de aprendizado não são de fácil interpretação, o que dificulta o processo de diagnóstico. Não foi encontrado na literatura nenhum outro trabalho que fez uso de testes estatísticos para diagnosticar problemas de desempenho.

– A implementação de um algoritmo de aprendizado estatístico capaz de aprender de forma *online* e a validação desse algoritmo no contexto de previsão de problemas de desempenho de aplicações baseadas em *middleware*. Essa validação permite estabelecer as principais diferenças de uma abordagem *online* em relação às abordagens tradicionais no contexto do problema proposto.

Como uma contribuição relacionada, vários experimentos foram descritos, implementados e testados com o intuito de apoiar o estudo. Os experimentos mostram claramente o impacto que os modelos de desempenho, induzidos pelas abordagens estatísticas empregadas, têm sobre características como tempo de resposta dos serviços e utilização de recursos. Os resultados comprovam que abordagens estatísticas podem contribuir de forma decisiva para um controle maior do desempenho de aplicações baseadas em *middleware*. Outra contribuição deste trabalho é a arquitetura de gerenciamento desenvolvida para apoiar o estudo. Através do SMART, sistemas baseados em componentes podem prever momentos de degradação de desempenho, mesmo diante de flutuações ou variações no ambiente de execução. Também é possível diagnosticar e classificar o grau de influência de diversas métricas sobre o problema de desempenho observado.

1.2 Estrutura da Tese

Este documento está organizado da seguinte forma.

– Capítulo 2: é apresentada uma introdução aos sistemas baseados em componentes e à computação autonômica. Também é apresentada uma visão geral sobre a área de gerenciamento de desempenho em sistemas distribuídos. As principais direções tomadas pela área nos últimos anos são descritas e analisadas.

– Capítulo 3: é estabelecido um cenário de referência para o estudo. Esse cenário consiste em um sistema de componentes particular e uma aplicação construída sobre o mesmo. O capítulo apresenta também o projeto de uma arquitetura de gerenciamento de sistemas.

– Capítulo 4: descreve a solução proposta neste trabalho para construir modelos de desempenho de aplicações baseadas em *middleware*. O foco é um estudo sistemático das principais classes de algoritmos de aprendizado estatístico a fim de estabelecer suas habilidades em prover um mecanismo para capturar o comportamento do desempenho de aplicações desse tipo.

– Capítulo 5: descreve o uso de testes estatísticos na solução de dois problemas. O primeiro é a construção de um mecanismo de análise de desempenho robusto a falhas transientes dos preditores. O segundo problema é a construção de um mecanismo de diagnóstico de problemas de desempenho em sistemas baseados em *middleware*.

– Capítulo 6: são apresentadas motivações para o uso de técnicas de aprendizado *online* para prever problemas de desempenho em aplicações baseadas em *middleware*. A versão *online* de um preditor baseado em aprendizado estatístico é implementada e avaliada com o intuito de estabelecer as vantagens e desvantagens de técnicas de aprendizado *online* para o problema proposto. Também é avaliada uma versão desse algoritmo para situações de dinamismo, onde os conceitos aprendidos sofrem mudanças ao longo do tempo.

– Capítulo 7: são apresentadas as conclusões deste trabalho juntamente com uma revisão das contribuições desta pesquisa e a indicação de trabalhos futuros.