7
Conclusão

Este trabalho pode ser descrito como um estudo que se inicia e termina com a avaliação do uso de abordagens estatísticas para caracterizar problemas de desempenho em sistemas baseados em *middleware*. Esse estudo começou com a definição de um cenário de referência para os experimentos, o qual estabeleceu a tecnologia de *middleware* a ser utilizada e a aplicação de referência. Em seguida, definiu-se uma arquitetura de gerenciamento do sistema, a qual denominamos SMART. Tendo definido o cenário de referência e a arquitetura de gerenciamento, o problema foi modelado como um caso de classificação. Várias classes de modelos de aprendizado estatístico e supervisionado foram implementadas no SMART e avaliadas sob diferentes critérios. Nessa avaliação, as redes Bayesianas de estruturas simples (NB, TAN e AODE) foram consideradas as mais apropriadas para o problema, uma vez que as mesmas apresentaram a melhor relação entre custo de construção e poder de predição. Adicionalmente, esses algoritmos também apresentam baixa sensibilidade ao número de instâncias em estado de violação e atingem acurácia razoável diante de *datasets* pequenos. A partir das informações e experiência acumuladas dessa avaliação, três problemas emergentes com o uso de técnicas clássicas de aprendizado estatístico foram identificados: uma taxa não negligenciável de alarmes falsos; a dificuldade do uso dos próprios modelos de previsão para classificar os atributos de maior influência para um problema particular; e a dificuldade de um processo de aprendizado *offline* para a construção de modelos de previsão que operam de forma *online*. Tendo como meta a solução do primeiro problema identificado, foi desenvolvido um algoritmo capaz de aumentar a robustez do processo de predição em relação a falhas transientes dos classificadores. Após sua concepção, o algoritmo foi implementado no SMART e avaliado em diferentes experimentos, mostrando resultados satisfatórios. Para o segundo problema, foi proposto o uso de testes estatísticos. Três testes estatísticos foram avaliados com o intuito de construir um mecanismo de diagnóstico de problemas de desempenho. Esses algoritmos foram implementados no SMART e testados também em diferentes *datasets*. Em particular, o teste *z-scores* apresentou bons resultados para o problema pro-

posto. Por fim, o estudo termina avaliando o uso de classificadores capazes de operar em cenários de aprendizado *online* e o desempenho desses para caracterizar problemas de desempenho. Um classificador Naive Bayes incremental foi concebido e implementado no SMART. O classificador *online* foi avaliado em diferentes experimentos e cenários, mostrando bons resultados, especialmente em situações inesperadas, ou seja, situações não percebidas durante a fase de treinamento.

As principais contribuições dessa tese podem ser assim resumidas:

- Estudo e avaliação de algoritmos de aprendizado supervisionado para a construção de modelos de desempenho de aplicações baseadas em *middleware*. Em especial, verificou-se que as redes Bayesianas de estruturas simples, como Naive Bayes, AODE e TAN, podem capturar com acurácia e eficiência o desempenho das aplicações. Apesar dessas características, verificou-se também que os algoritmos de classificação apresentam uma taxa de alarme falso não negligenciável, dificuldade de interpretação dos modelos construídos e impossibilidade de atualização dos modelos.
- Desenvolvimento e avaliação de um algoritmo para aumentar a robustez do processo de predição em relação a falhas transientes dos classificadores. Esse algoritmo aumenta o poder de predição dos classificadores, combinando-os com um teste estatístico para detecção de tendências em séries temporais. Com os datasets utilizados, o algoritmo proposto apresentou uma acurácia superior aos dos classificadores puros.
- Proposta e avaliação de um mecanismo de diagnóstico de problemas de desempenho baseado em testes estatísticos. Foi verificado que o teste *z-scores* pode classificar a influência de cada atributo observado em um problema particular com acurácia e eficiência, obtendo assim, um mecanismo de diagnóstico independente do modelo de previsão.
- Desenvolvimento e avaliação de um classificador Naive Bayes capaz de aprender de forma incremental. Verificou-se que o classificador *online* é capaz de se adaptar a situações inesperadas e, nessas situações, apresenta um desempenho superior aos dos classificadores *offline*. Além disso, para situações esperadas, os classificadores *online* apresentam desempenho semelhante aos classificadores *offline*. Para ambientes dinâmicos, classificadores *online* podem ser facilmente alterados para lidar com o problema de *concept drift*.

Em resumo, esse estudo demonstra que abordagens estatísticas podem contribuir de forma decisiva tanto para a previsão de problemas de desempenho

de aplicações baseadas em *middleware* quanto para a determinação das causas desses problemas. Em relação à previsão de problemas de desempenho, conclui-se que técnicas de aprendizado estatístico podem capturar com acurácia o relacionamento, muitas vezes não trivial, que envolve um ambiente de execução e os componentes de aplicação. No entanto, as características do ambiente de execução influenciam a acurácia dos modelos. De maneira geral, observa-se que, em ambientes de execução cujas características se alteram com pouca frequência, um algoritmo robusto a falhas transientes dos classificadores produz previsões mais acuradas que um classificador puro. No entanto, o mesmo algoritmo implantado em um ambiente com características dinâmicas, apresenta um desempenho inferior, visto que o algoritmo é mais conservador em relação a mudanças, as quais são interpretadas inicialmente como alarmes falsos. Por outro lado, em condições de dinamismo, um classificador *online* obtém o melhor desempenho por ser capaz de incorporar novas situações e aprender com elas. Em relação ao diagnóstico de problemas de desempenho, conclui-se que testes estatísticos podem caracterizar com acurácia o potencial de tendência de uma métrica. Esse fato foi observado especialmente nos testes que estimam as diferenças entre duas populações. Observamos que testes estatísticos são simples, não demandam treinamento prévio e não exigem nenhuma parametrização. Conclui-se, portanto, que abordagens estatísticas podem ser usadas de forma eficaz e eficiente em diferentes perspectivas da caracterização de um problema de desempenho.

Além das contribuições principais descritas acima, este trabalho apresentou outras secundárias. A seguir, essas contribuições são comentadas brevemente.

– Uma arquitetura de gerenciamento desenvolvida para apoiar o estudo proposto. 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. Por outro lado, a implementação do SMART só seria possível se existisse uma infraestrutura de monitoramento para o SCS. Essa necessidade gerou um trabalho de mestrado sobre infraestruturas de monitoramento para sistemas baseados em componentes de software (Andrea, 2009). De maneira geral, as principais contribuições da infraestrutura de monitoramento do SCS foram publicadas em (Fonseca et al., 2008) e do SMART podem ser encontradas em (Correa e Cerqueira, 2010). O código fonte do SMART e todos os *datasets* usados neste trabalho podem ser obtidos em http://www.inf.ufg.br/∼sand/.

– Um estudo sobre computação autonômica em sistemas distribuídos e um levantamento das necessidades de infraestrutura para sua realização. Os resultados desse estudo foram publicados em (Correa e Cerqueira, 2009).

– Implementação dos algoritmos Pid Naive Bayes e CDHPid Naive Bayes na ferramenta Weka. Essa implementação está disponível em http://www.inf.ufg.br/∼sand/.

– O desenvolvimento de uma ferramenta de injeção de carga. Essa ferramenta foi usada para gerar os vários *datasets* usados nesta tese. Para carga de CPU, essa ferramenta usa o ReservationSuite (Reis, 2010). A carga de disco e rede é feita usando o Fio (*Flexible I/O Tester*) (Fio, 2011). A necessidade de um gerador de carga para os testes deste trabalho gerou um trabalho de conclusão de curso de graduação (Pedras, 2009).

Apesar das contribuições listadas acima, o estudo proposto neste trabalho apresenta limitações, como discutido nos capítulos anteriores e resumidas a seguir:

– O comportamento homogêneo de aplicações que usam o modelo de programação MapReduce. Esse comportamento dificulta a generalização das conclusões para outros tipos de aplicações que têm uma natureza transacional como, por exemplo, aplicações Web.

– A arquitetura completamente descentralizada do SMART que leva à impossibilidade dessa arquitetura correlacionar eventos distribuídos.

– A necessidade de escolher parâmetros adequados para o algoritmo de alerta, a fim de evitar que o algoritmo se torne reativo. Essa parametrização depende do *dataset* usado.

– O mecanismo de diagnóstico se baseia em estatística Gaussiana, podendo se tornar menos aplicável em dados não paramétricos.

Por fim, sendo este um trabalho experimental, torna-se essencial discutir a generalização dos resultados. Em princípio, é razoável afirmar que os resultados obtidos neste trabalho podem ser estendidos para sistemas distribuídos em geral. Essa afirmação é pertinente visto que o trabalho realizado depende muito pouco de características específicas de sistemas de componentes. Por outro lado, torna-se difícil generalizar os resultados deste trabalho para aplicações com comportamentos ou perfis de utilização de recursos muito diferentes do perfil de aplicações do tipo MapReduce. Algumas decisões importantes neste trabalho foram tomadas com base no perfil de aplicações desse tipo. Um exemplo consiste na própria frequência com que os dados do sistema são coletados

e analisados. Para aplicações com perfil transacional, acreditamos que uma janela de monitoramento de 15 segundos seja demasiado. Também não é trivial a generalização dos resultados deste trabalho para diferentes implementações dos algoritmos de classificação investigados. Uma preocupação e cuidado constante neste trabalho foi o uso de bibliotecas e implementações amplamente utilizadas na comunidade de aprendizado de máquina. No entanto, acreditamos que a generalização em termos de implementação dos algoritmos exige um estudo formal que analise os limites dos algoritmos em função de perfis e características próprias das aplicações. Esse formalismo não foi abordado neste trabalho.

Perspectivas para Trabalhos Futuros

Durante a elaboração desta tese foi possível identificar alguns temas de interesse para investigação futura. Algumas investigações consistem em pequenas atividades que podem completar este trabalho, como descrito a seguir.

- Na metodologia empregada para gerar os *datasets* deste trabalho, optou-se pela remoção dos *outlier* do conjunto de dados. Recentemente, no entanto, testes preliminares sugeriram que os classificadores se comportam melhor, isto é, são mais acurados, quando treinados em *datasets* em que há a presença de ruído. Há, portanto, a necessidade de se realizar mais testes a fim de determinar a melhor abordagem para este problema.
- Uma alternativa ao algoritmo de alerta proposto neste trabalho consiste em tornar o tempo uma dimensão do problema. Dessa forma, o tempo passa a ser um atributo do *dataset* a ser analisado pelo classificador. Nesse sentido, novos testes se tornam necessários para avaliar o desempenho dos classificadores nesse novo modelo.
- O SMART se baseia na medida da probabilidade de um nó violar um SLO definido pelo usuário. Neste trabalho, essa informação foi utilizada para realizar melhores escolhas no escalonamento de um conjunto de tarefas de um *framework* MapReduce. Como o interesse era explorar a correlação existente entre componentes de software de uma aplicação e sua infraestrutura e ambiente de execução, foi desenvolvido um *framework* MapReduce usando o modelo de componentes do SCS. No entanto, seria de grande importância acoplar o SMART à implementação MapReduce oferecida pelo Hadoop (ApacheSoftwareFoundation, 2010). Isso permitiria verificar se as observações deste trabalho são válidas também para

outras implementação de MapReduce, além de permitir comparar resultados de desempenho.

Outras investigações demandam mais esforço mas são desdobramentos naturais deste trabalho. A seguir, discutimos algumas propostas.

- Uma linha de investigação que deve ser melhor explorada é o tratamento de *concept drift*. Acreditamos que esse não é um fenômeno raro em computação distribuída. Nesse sentido, seria de grande importância testar o algoritmo CDHPid Naive Bayes em cenários em que há maior chance de mudança de conceito. Alguns exemplos incluem: ambientes de grades computacionais em que uma máquina com características de hardware diferentes pode vir a integra a grade ou ambientes onde múltiplas aplicações executam concorrentemente, quer sejam múltiplas instâncias de aplicações MapReduce em estágios de processamento distintos, quer sejam aplicações compostas por processos com perfis de uso de recurso e de interação distintos.
- Uma outra linha de estudos relaciona-se com o enriquecimento dos modelos de predição com informações sobre a arquitetura da aplicação. Embora a adição desse tipo de informação possa dificultar a possibilidade de generalização da solução, os modelos construídos tenderiam a ser mais acurados. Além disso, a adição de tais informações ao modelo poderia levar à necessidade de correlacionar eventos distribuídos, levando a uma outra linha de trabalho futuro.
- Ainda na linha de trabalhos futuros, seria de grande importância ligar o trabalho realizado com o *loop* autonômico. Ou seja, o SMART poderia não só antecipar problemas de desempenho, mas também atuar no sistema antes que eles ocorressem. Nessa linha, podemos realizar experimentos que envolvam reconfigurações do sistema, seja através de mudanças na configuração da aplicação, através de reconexões entre componentes ou na infraestrurura de execução, seja através da migração de máquinas virtuais.
- Finalmente, uma conclusão deste trabalho é que as características do ambiente de execução influenciam a acurácia dos modelos de previsão e, portanto, algumas técnicas são mais apropriadas que outras, dependendo do contexto. Esse fato serve de motivação para investigar se a combinação de múltiplas técnicas ou máquinas de inferência simultaneamente pode levar a um melhor diagnóstico e gerenciamento do sistema.

Por fim, apesar dos resultados já obtidos na área, acreditamos que ainda existem várias linhas de investigação a serem exploradas no contexto de técnicas estatísticas como uma ferramenta para tornar a visão de Computação Autonômica uma realidade.