

Sand Luz Corrêa

**Detecção estatística de anomalias de
desempenho em sistemas baseados em
middleware**

Tese de Doutorado

Tese apresentada ao Programa de Pós-graduação em Informática
do Departamento de Informática da PUC-Rio como requisito
parcial para obtenção do título de Doutor em Informática

Orientador: Prof. Renato Fontoura de Gusmão Cerqueira

Rio de Janeiro
abril de 2011

Sand Luz Corrêa

**Detecção estatística de anomalias de
desempenho em sistemas baseados em
middleware**

Tese apresentada ao Programa de Pós-graduação em Informática do Departamento de Informática do Centro Técnico Científico da PUC-Rio como requisito parcial para obtenção do título de Doutor em Informática. Aprovada pela Comissão Examinadora abaixo assinada.

Prof. Renato Fontoura de Gusmão Cerqueira

Orientador

Departamento de Informática — PUC-Rio

Profa. Noemi de la Rocque Rodriguez

Departamento de Informática — PUC-Rio

Prof. Ruy Luiz Milidiú

Departamento de Informática — PUC-Rio

Prof. Raimundo José de Araújo Macêdo

Departamento de Ciência da Computação — UFBA

Prof. Rodrigo Fernandes de Mello

Instituto de Ciências Matemáticas e de Computação — USP

Prof. José Eugenio Leal

Coordenador Setorial do Centro Técnico Científico — PUC-Rio

Rio de Janeiro, 1 de abril de 2011

Todos os direitos reservados. É proibida a reprodução total ou parcial do trabalho sem autorização da universidade, do autor e do orientador.

Sand Luz Corrêa

Mestre em Ciência da Computação pela Universidade Estadual de Campinas. Bacharel em Ciência da Computação pela Universidade Federal de Goiás.

Ficha Catalográfica

Corrêa, Sand

Detecção estatística de anomalias de desempenho em sistemas baseados em middleware / Sand Luz Corrêa; orientador: Renato Fontoura de Gusmão Cerqueira. — Rio de Janeiro : PUC–Rio, Departamento de Informática, 2011.

v., 146 f: il. ; 29,7 cm

1. Tese (doutorado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática.

Inclui referências bibliográficas.

1. Informática – Tese. 2. Computação Autônômica. 3. Previsão de Problemas de Desempenho. 4. Diagnóstico de Problemas de Desempenho. 5. Técnicas de Aprendizado Estatístico. 6. Testes Estatísticos. 7. Sistemas baseados em *Middleware*. I. Cerqueira, Renato. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

Agradecimentos

Ao meu pai e minhas irmãs, pelo amor e apoio durante toda a vida.

Ao meu companheiro, Kleber, pelo amor, carinho e pelas inúmeras discussões e críticas construtivas sobre este trabalho.

Ao Prof. Renato Cerqueira, por sua orientação e amizade.

Ao CNPq e ao Tecgraf, pelo suporte financeiro, sem o qual este trabalho não poderia ter sido realizado.

Resumo

Corrêa, Sand; Cerqueira, Renato. **Detecção estatística de anomalias de desempenho em sistemas baseados em middleware**. Rio de Janeiro, 2011. 146p. Tese de Doutorado — Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Tecnologias de *middleware* têm sido amplamente adotadas pela indústria de software para reduzir o custo do desenvolvimento de sistemas computacionais. No entanto, é difícil estimar o desempenho de aplicações baseadas em *middleware* devido a fatores como a especificidade de implementação das plataformas de *middleware* e a multiplicidade de serviços e configurações providas para diferentes cenários de implantação. Assim, o gerenciamento do desempenho de aplicações distribuídas baseadas em *middleware* pode ser uma tarefa não trivial. Computação autônoma é um novo paradigma para construir sistemas autogerenciáveis, isto é, que procuram operar com o mínimo de intervenção humana. Este trabalho investiga o uso de abordagens estatísticas para construir mecanismos autônomos de controle do desempenho de aplicações baseadas em *middleware*. Particularmente, o tema é investigado sob três perspectivas. A primeira é pertinente à previsão de problemas de desempenho. O uso de técnicas de classificação é proposto para derivar modelos de desempenho que auxiliem o gerenciamento autônomo das aplicações. Nesse sentido, diferentes classes de modelos em aprendizado estatístico são avaliadas, tanto em cenários de aprendizado *offline* quanto *online*. A segunda perspectiva refere-se à redução da emissão de alarmes falsos, visando a construção de mecanismos robustos a falhas transientes dos classificadores. Este trabalho propõe um algoritmo que aumenta o poder de predição das técnicas de aprendizado estatístico, combinando-as com testes estatísticos para a detecção de tendência. Por fim, a terceira perspectiva é pertinente ao diagnóstico das causas de um problema de desempenho. Para isso, propõe-se também o uso de testes estatísticos. Os resultados apresentados nesta tese demonstram que abordagens estatísticas podem contribuir para a construção de ferramentas eficazes e eficientes para a caracterização do desempenho de aplicações baseadas em *middleware*. Portanto, essas abordagens podem contribuir de forma decisiva para diferentes perspectivas do problema.

Palavras-chave

Computação Autônoma; Previsão de Problemas de Desempenho; Diagnóstico de Problemas de Desempenho; Técnicas de Aprendizado Estatístico; Testes Estatísticos; Sistemas baseados em *Middleware*;

Abstract

Corrêa, Sand; Cerqueira, Renato (Advisor). **Statistical detection of performance anomalies in middleware-based systems.**

Rio de Janeiro, 2011. 146p. DsC Thesis — Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Middleware technologies have been widely adopted by the software industry to reduce the cost of developing computer systems. Nonetheless, predicting the performance of middleware-based applications is difficult due to specific implementation details of a middleware platform and a multitude of settings and services provided by middleware for different deployment scenarios. Thus, the performance management of middleware-based applications can be a non-trivial task. Autonomic computing is a new paradigm for building self-managed systems, *i.e.*, systems that seek to operate with minimal human intervention. This work investigates the use of statistical approaches to building autonomic management solutions to control the performance of middleware-based applications. Particularly, we investigate this issue from three perspectives. The first is related to the prediction of performance problems. We propose the use of classification techniques to derive performance models to assist the autonomic management of distributed applications. In this sense, different classes of models in statistical learning are assessed in both offline and online learning scenarios. The second perspective refers to the reduction of false alarms, seeking the development of reliable mechanisms that are resilient to transient failures of the classifiers. This work proposes an algorithm to augment the predictive power of statistical learning techniques by combining them with statistical tests for trend detection. Finally, the third perspective is related to diagnosing the root causes of a performance problem. For this context, we also propose the use of statistical tests. The results presented in this thesis show that statistical approaches can contribute to the development of tools that are both effective, as well as efficient in characterizing the performance of middleware-based applications. Therefore, these approaches can contribute decisively to different perspectives of the problem.

Keywords

Autonomic Computing; Performance Problem Forecasting; Performance Problem Diagnosis; Statistical Learning Techniques; Statistical Testing; Middleware-based Systems;

Sumário

1	Introdução	13
1.1	Objetivos, Abordagem e Contribuições	16
1.2	Estrutura da Tese	21
2	Fundamentos e Trabalhos Relacionados	22
2.1	Gerenciamento de Desempenho	22
2.2	Sistemas Baseados em Componentes de Software	24
2.3	Computação Autônoma	27
2.4	Trabalhos Relacionados	29
2.4.1	Modelos para Previsão de Problemas de Desempenho	29
2.4.2	Modelos baseados em Aprendizado <i>Online</i>	32
2.4.3	Modelos para Diagnóstico de Problemas de Desempenho	33
2.5	Considerações Finais	35
3	Cenário de Referência	36
3.1	SCS	36
3.2	O Modelo de Programação MapReduce	39
3.2.1	Um <i>framework</i> MapReduce usando um Sistema de Componentes	40
3.3	SMART	45
3.3.1	Infraestrutura de Monitoramento	45
3.3.2	Máquina de Inferência	48
3.4	Considerações Finais	52
4	Estimação de Problemas de Desempenho	54
4.1	Conceitos Básicos	55
4.1.1	Fundamentos de Probabilidade	55
4.1.2	Classificação e Aprendizado Supervisionado	56
4.2	Classes de Modelos em Aprendizado Supervisionado	57
4.2.1	Redes Bayesianas	57
4.2.2	Árvore de Decisão	60
4.2.3	<i>Support Vector Machine</i>	62
4.3	Discretização	64
4.4	Métricas de Acurácia	65
4.5	Estimação de Problemas de Desempenho	67
4.6	Experimentos	68
4.6.1	<i>Log</i> da Aplicação MapReduce	70
4.6.2	Avaliação do Nível de Discretização	71
4.6.3	Avaliação do Desempenho dos Algoritmos	72
4.6.4	Avaliação do Impacto da Definição do SLO	75
4.6.5	Avaliação do Impacto do Tamanho do <i>Dataset</i>	76
4.6.6	Avaliação do Desempenho do SMART	76
4.7	Considerações Finais	78
5	Robustez e Diagnóstico de Problemas de Desempenho	81
5.1	Alarmes Falsos em Métodos de Classificação	81

5.1.1	Algoritmo Robusto de Alerta de Problema de Desempenho	83
5.1.2	Experimentos	85
5.2	Diagnóstico de Problemas de Desempenho	91
5.2.1	Testes Estatísticos Avaliados	92
5.2.2	Experimentos	93
5.3	Considerações Finais	97
6	Aprendizado <i>Online</i>	99
6.1	Motivação para o uso de aprendizado <i>online</i>	99
6.2	Aprendizado incremental	101
6.2.1	PID - <i>Partition Incremental Discretization</i>	103
6.2.2	PID Naive Bayes	106
6.2.3	Experimentos	107
6.3	Gerenciamento de <i>concept drift</i>	113
6.3.1	CDHPid Naive Bayes	114
6.3.2	Gráfico de controle	115
6.3.3	Experimentos	117
6.4	Considerações Finais	125
7	Conclusão	127
8	Referências Bibliográficas	134

Lista de figuras

2.1	Ciclo de gerenciamento em arquiteturas autonômicas. Baseada em (Parashar e Hariri, 2007)	28
3.1	Um componente SCS típico	37
3.2	Componentes da infraestrutura de execução do SCS	38
3.3	Instanciação de um componente SCS	39
3.4	Fluxo de controle de um programa MapReduce	41
3.5	Componentes que formam o framework MapReduce	43
3.6	A arquitetura do SMART.	46
3.7	Distribuição do tempo de resposta para as operações map e reduce	49
4.1	Redes Bayesianas com diferentes estruturas.	59
4.2	Estrutura de uma árvore de decisão.	61
4.3	Um classificador SVM.	63
4.4	Avaliação do comportamento dos algoritmos de aprendizado supervisionado em relação ao número de intervalos na discretização.	72
4.5	Comparação dos algoritmos de aprendizado supervisionado em relação ao poder de estimação e tempo requerido para treinamento.	73
4.6	Comparação dos algoritmos de aprendizado em relação à sensibilidade ao SLO.	76
4.7	Comparação dos algoritmos de aprendizado supervisionado em relação ao número de instâncias no <i>dataset</i> de treinamento.	77
4.8	Desempenho da aplicação MapReduce usando uma política de escalonamento baseada em CPU e na probabilidade de violação.	78
5.1	Comportamento da probabilidade de violação emitida por um classificador em um sistema submetido a uma carga crescente.	82
5.2	Comportamento do algoritmo de alerta de problemas de desempenho quando o sistema é submetido a um padrão de carga senoidal.	87
5.3	Comportamento do algoritmo de alerta de problemas de desempenho quando o sistema é submetido a um padrão de carga do tipo escada.	88
5.4	Desempenho da aplicação MapReduce usando o algoritmo de alerta de problemas de desempenho.	91
5.5	Tempo de resposta requerido por cada teste estatístico para diagnosticar um problema.	97
6.1	Um sistema de aprendizado supervisionado incremental.	102
6.2	Classes usadas na implementação do classificador Pid Naive Bayes.	107
6.3	Comportamento do algoritmo Pid Naive Bayes em relação ao número de intervalos da discretização e comparação desse comportamento com o comportamento dos algoritmos <i>offline</i> .	108
6.4	Comportamento do algoritmo Pid Naive Bayes ao longo do processo de aprendizado.	109

6.5	Desempenho da aplicação MapReduce usando o algoritmo Pid Naive Bayes.	111
6.6	Desempenho da aplicação MapReduce usando o algoritmo Pid Naive Bayes, num cenário diferente dos percebidos no treinamento.	112
6.7	Comportamento do algoritmo CDHPid Naive Bayes.	114
6.8	Um gráfico de controle típico.	116
6.9	Comportamento do algoritmo CDHPid Naive Bayes durante treinamento usando o <i>dataset</i> STAGGER.	120
6.10	Desempenho dos classificadores Pid Naive Bayes e CDHPid Naive Bayes no <i>dataset</i> STAGGER.	121
6.11	Desempenho dos classificadores Pid Naive Bayes e CDHPid Naive Bayes no <i>dataset MapReduce CDH</i> .	122
6.12	Teste de estacionariedade para o <i>dataset MapReduce CDH</i>	124
6.13	Desempenho da aplicação MapReduce usando o algoritmo CDHPid Naive Bayes.	125

Lista de tabelas

3.1	Métricas coletadas e usadas no processo de análise	48
4.1	Funções <i>kernel</i> não lineares comumente usadas.	64
4.2	Resultados possíveis para uma classificação binária.	66
4.3	Configuração dos parâmetros usados nos algoritmos de aprendizado supervisionado.	69
4.4	Comparação detalhada do desempenho dos algoritmos de aprendizado supervisionado.	74
4.5	Acurácia e tempo de processamento do classificador TAN em cada <i>dataset</i> gerado a partir de uma definição de SLO.	75
4.6	Cenário de carga para o experimento que testa a eficácia do SMART.	77
5.1	Configuração dos parâmetros usados no algoritmo de alerta de problemas de desempenho quando o sistema é submetido a um padrão de carga senoidal.	86
5.2	Configuração dos parâmetros usados no algoritmo de alerta de problemas de desempenho para o teste de acurácia.	89
5.3	Teste de desempenho para o algoritmo de alerta com diferentes classificadores.	89
5.4	Teste de desempenho para os classificadores.	90
5.5	<i>Datasets</i> para avaliação dos mecanismos de diagnóstico.	94
5.6	Métricas selecionadas pelos testes estatísticos	96
6.1	Comparação do desempenho do algoritmo Pid Naive Bayes em relação ao desempenho dos algoritmos <i>offline</i> e o algoritmo Naive Bayes Updateable.	110
6.2	Configuração usada no experimento para avaliar o desempenho da aplicação MapReduce usando o algoritmo Pid Naive Bayes, considerando o cenário de carga senoidal.	112
6.3	Acurácia de TAN e PID Naive Bayes durante o experimento	113
6.4	Testes de estacionariedade.	123

Testemunhar com ares de quem possui a verdade é um rotundo desacerto... Pensar certo supõe a disponibilidade à revisão dos achados, reconhece não apenas a possibilidade de mudar de opção, de apreciação, mas o direito de fazê-lo. Mas como não há pensar certo à margem de princípios éticos, se mudar é uma possibilidade e um direito, cabe a quem muda - exige o pensar certo - que assuma a mudança operada.

Paulo Freire