ESTRATÉGIAS DE IDENTIFICAÇÃO PARAMÉTRICA APLICADAS À MODELAGEM DINÂMICA DE UM SERVIDOR *WEB* APACHE

Thiago W. M. Abreu[*] **Walter Barra Jr.**[*]

José A. L. Barreiros[*] **Carlos T. da Costa Junior**[*]

[*]Programa de Pós-Graduação em Engenharia Elétrica (PPGEE)
Universidade Federal do Pará (UFPA) - CEP 66075-900 - Belém, PA

ABSTRACT

Parametric Identification Techniques Applied to Dynamic Modeling of an Apache Webserver

This article presents an experimental study about parametric identification techniques applied to the modeling of an Apache webserver. In order to simulate load variations at the server, an experimental arrangement was developed, which is composed of two personal computers, one used to run the Apache server and the other to generate workload by requesting services to the Apache. Auto-regressive (AR) parametric models were estimated at different operating points and workload conditions. The mean values of the *MaxClients* input (a parameter which is used to set the maximum number of the server's active processes) were used to define the operating points, in order to obtain the Apache server CPU utilization (in %) as output. 600 samples were collected at each operating point every 5 seconds. To proceed with the system identification, half of the data set was used for parameter estimation while the other half was used for model validation, at each operating point. A study of the most adequate system order showed that a 7th order model could be satisfactorily used for *MaxClients* low values operating points. However, the results showed that higher order models are needed for *MaxClients* higher values, due to system inherent non-linearities.

KEYWORDS: System Identification, Parametric Models, Apache Web Server, Dynamic Systems, Computing Systems.

RESUMO

Este artigo apresenta um estudo experimental de técnicas de identificação paramétrica aplicadas à modelagem dinâmica de um servidor *web* Apache. Foi desenvolvido um arranjo experimental para simular variações de carga no servidor. O arranjo é composto por dois computadores PC, sendo um deles utilizado para executar o servidor Apache e o outro utilizado como um gerador de carga, solicitando requisições de serviço ao servidor Apache. Foram estimados modelos paramétricos auto-regressivos (AR) para diferentes pontos de operação e de condição de carga. Cada ponto de operação foi definido em termos dos valores médios para o parâmetro de entrada *MaxClients* (parâmetro utilizado para definir o número máximo de processos ativos) e a saída percentual de consumo de CPU (*Central Processing Unit*) do servidor Apache. Para cada ponto de operação foram coletadas 600 amostras, com um intervalo de amostragem de 5 segundos. Metade do conjunto de amostras coletadas em cada ponto de operação foi utilizada para estimação do modelo, enquanto que a outra metade foi utilizada para validação. Um estudo da ordem mais adequada do modelo mostrou que, para um ponto de operação com valor reduzido de *MaxClients*, um modelo AR de 7^a ordem pode ser satisfatório. Para valores mais elevados de *MaxClients*, os resultados mostraram que são necessários modelos de ordem mais elevada, devido às não-linearidades inerentes ao sistema.

Artigo submetido em 23/06/2010 (Id.: 01162)
Revisado em 30/08/2010, 31/01/2011, 11/05/2011
Aceito sob recomendação do Editor Associado Prof. Luis Antonio Aguirre

PALAVRAS-CHAVE: Identificação de Sistemas, Modelos Paramétricos, Servidor Web Apache, Sistemas Dinâmicos, Sistemas Computacionais.

1 INTRODUÇÃO

Os sistemas servidores *Web* formam a base da maioria dos sistemas de comércio eletrônico e dos sistemas de informações empresariais e governamentais atualmente existentes. O desempenho dinâmico de um servidor *Web* é fortemente dependente da condição de carregamento do sistema. Variações de carga intensas podem degradar substancialmente a qualidade do serviço fornecido aos clientes. A carga em um servidor *Web* pode ser definida, em termos simplificados, como sendo o número de clientes requisitando serviços em um determinado instante de tempo. Tal degradação na qualidade do serviço é percebida pelo usuário através de um aumento considerável no tempo de resposta do sistema às suas requisições.

Para garantir um desempenho satisfatório, um sistema servidor deverá ser dotado de um mecanismo de gerenciamento do consumo instantâneo dos recursos computacionais (percentuais de uso de memória e de uso de CPU (*Central Processing Unit*)), de modo a assegurar certa reserva (margem) de recursos para lidar com eventuais aumentos repentinos de carga. Caso os recursos de memória e CPU não sejam gerenciados de uma forma satisfatória, sob determinadas condições de aumento de carga tais recursos poderão ser exauridos, implicando uma indesejável interrupção do serviço, com prejuízos tanto para os usuários quanto para o provedor do serviço.

Em tais situações extremas, um administrador do sistema necessitaria efetuar novos ajustes nos arquivos de configuração do servidor, de modo a tentar, pelo menos, restabelecer o serviço. Esta tarefa é bastante tediosa e pode se tornar muito repetitiva, justificando um esforço para automatizar o processo de reconfiguração do servidor. Outra solução convencional, que tem sido adotada para lidar com aumentos de carga, é superdimensionar a capacidade do sistema servidor, em termos de memória e de capacidade de processamento da CPU, através da aquisição e instalação de hardware adicional. Uma desvantagem clara dessa solução é que ela tende a encarecer o valor final do serviço fornecido ao usuário. Assim sendo, torna-se altamente desejável o desenvolvimento de novas abordagens para ajuste automático dos parâmetros de operação de um servidor *Web*, baseado no uso de técnicas modernas de identificação de sistemas (Aguirre, 2007; Coelho and Coelho, 2004; Ljung, 1999) e de engenharia de controle (Bazanella and Silva Jr., 2005; Ogata, 2003). Com essa nova abordagem, o objetivo é obter uma utilização otimizada dos recursos computacionais pré-existentes no sistema e manter satisfatória a qualidade do serviço sob diferentes condições de carga do servidor. Nessa abordagem moderna, torna-se necessário primeiramente obter modelos matemáticos que capturem satisfatoriamente a dinâmica do sistema. Tais modelos podem ser obtidos através de testes experimentais de identificação de sistemas e podem ser posteriormente utilizados tanto para fins de predição quanto para o projeto de controladores automáticos.

A aplicação de técnicas modernas de identificação e controle na busca da melhoria do desempenho dinâmico de sistemas computacionais tem sido objeto de diversas pesquisas recentes (Abreu et al., 2009; Alvarez et al., 2010; Diao et al., 2002; Gandhi et al., 2002; Hellerstein et al., 2005; Kihl et al., 2008; Kjaer et al., 2007; Kjaer and Robertsson, 2010). O uso de metodologias de ajuste automático de servidores é importante, uma vez que é humanamente impossível aos operadores estarem disponíveis durante 24 horas por dia para acompanhar e atuar em tempo real no reajuste contínuo do desempenho de um sistema servidor. É necessário, portanto, desenvolver ferramentas de *software* que realizem, de uma forma automática, a reconfiguração dos parâmetros de operação dos servidores, sempre que houver alguma mudança detectada nas condições de operação e de carga do servidor.

Embora a engenharia de computação seja uma área de rápida evolução, ainda existem alguns campos que carecem de uma abordagem sistemática para modelagem dinâmica e controle automático. O gerenciamento de uso de recursos de CPU e de memória por parte de um servidor *Web* é certamente um dos campos onde o emprego sistemático de técnicas de identificação e de engenharia de controle automático pode contribuir para aumentar acentuadamente o desempenho dinâmico do processo, eliminando-se o emprego de sintonia manual, como atualmente ainda que é o procedimento adotado por grande parte das organizações e profissionais encarregados de gerenciar tais sistemas.

Vale observar que, diferentemente de alguns sistemas eletromecânicos bem conhecidos dos engenheiros de controle automático, nos quais as leis físicas que regem o comportamento dinâmico são bem conhecidas, o funcionamento interno de um aplicativo servidor Web, do tipo do Apache, é um fenômeno dinâmico bastante complexo e ainda pouco conhecido, sendo, portanto, de difícil modelagem com base apenas em leis fundamentais Assim sendo, para tal tipo de sistema, uma alternativa bem indicada é o emprego sistemático de técnicas de identificação, para a obtenção de modelos paramétricos representativos do comportamento dinâmico do servidor Apache. Os parâmetros de tais modelos podem ser estimados diretamente dos dados coletados durante a operação do sistema. Os modelos paramétricos, assim obtidos, são úteis para auxiliar na previsão do comportamento determinístico do servidor em relação aos consumos de recursos de CPU e de memória. Além disso, os modelos paramétricos

são valiosos na implementação de futuras estratégias de controle automático para regulação do uso de recursos de CPU e de memória pelo servidor Web Apache.

Neste artigo, são apresentados e discutidos os resultados experimentais de um estudo realizado para a obtenção de modelos paramétricos do comportamento dinâmico de um servidor *Web* Apache, em termos de consumo médio de recursos de CPU. Os modelos paramétricos obtidos neste estudo representam um passo inicial no sentido de projetar futuras estratégias para controle digital para regulação automática do consumo de memória e de CPU, de acordo com a condição atual de carga de um servidor. Os modelos foram identificados a partir de medidas coletadas durante a operação do sistema e foram validados com base em análise de auto-correlação do resíduo.

Os autores ressaltam que, embora o emprego de técnicas de identificação de modelos paramétricos não seja um assunto novo, a contribuição principal deste artigo reside na aplicação sistemática e experimental da metodologia em um problema ainda pouco conhecido da comunidade científica, que é a modelagem do comportamento dinâmico de servidores *Web*. Dessa forma, os autores acreditam que a metodologia apresentada possa ser de interesse imediato tanto para leitores da área de engenharia de controle interessados em novas aplicações quanto para profissionais de engenharia de computação buscando metodologias sistemáticas para melhorar o desempenho de sistemas computacionais.

O presente artigo está organizado da seguinte forma. Na seção 2, é feita uma breve apresentação referente à modelagem dinâmica de um Servidor *Web* Apache e de seus respectivos modos de configuração e de ajustes de parâmetros de operação. Na seção 3, discute-se a metodologia de identificação utilizada para obtenção de modelos dinâmicos para o sistema servidor *Web* Apache. Na seção 4, apresenta-se o arranjo experimental utilizado para efetuar a coleta dos dados para o processo de identificação paramétrica de modelos dinâmicos do servidor Apache, discutindo-se os resultados obtidos nos testes. Finalmente, na seção 5 apresentam-se as considerações finais e as conclusões da pesquisa.

2 INTRODUÇÃO AO SERVIDOR *WEB* APACHE

2.1 Características gerais

O servidor *Web* Apache está dentre os mais utilizados servidores do mundo (Apache, 2010; NETCRAFT, 2010). Por ser um sistema de código aberto, foi muito difundido mundialmente devido às possibilidades de alteração e de melhorias em seu funcionamento, realizáveis por programadores iniciantes. Uma grande vantagem do servidor *Web* Apache consiste na existência de diferentes versões do programa, as quais permitem que o servidor funcione em diferentes sistemas operacionais, de forma transparente ao usuário.

Dentre as normas de funcionamento do servidor Apache, é importante analisar-se a forma com que o servidor trata as requisições que pedem atendimento pelo sistema.

O servidor Apache dispõe de processos trabalhadores (ou *threads* trabalhadoras, dependendo da configuração do servidor e do sistema operacional em questão), responsáveis pelo tratamento das requisições de serviços enviadas por clientes.

Estas requisições, antes de serem atendidas, entram em uma fila de requisições, que as organiza de modo que possam ser atendidas por processos que se encontrem em estado ocioso naquele momento (Figura 1).

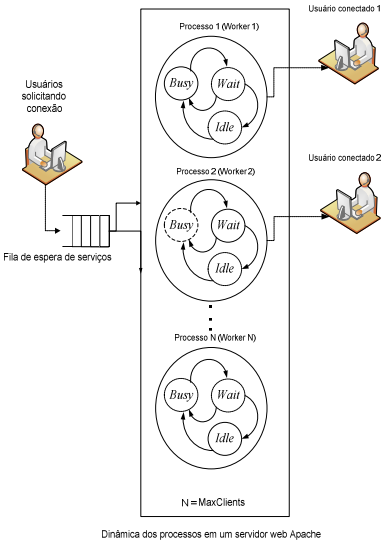


Figura 1: Tratamento de requisições de serviços pelo Servidor Apache. Adaptado de (Hellerstein et al., 2004).

Um processo trabalhador ("*workers*") ou *thread* pode assumir três estados de operação: "ocioso" ("*idle*"), quando não está atendendo requisições; "ocupado" ("*busy*"), nos momentos em que está atendendo requisições; e "em espera" ("*wait*"), este último quando um processo trabalhador possui uma conexão estabelecida com um cliente, mas não está processando requisições, e sim apenas está aguardando o momento em que o referido cliente enviará a próxima requisição. Durante o estado "em espera", nenhuma outra requisição de outro cliente poderá ser atendida por este processo, apenas as provenientes do cliente com o qual o processo estiver atualmente conectado.

O servidor Apache é desenvolvido sob a forma de módulos, os quais correspondem a conjuntos de funcionalidades que

operam independentemente (na maioria das vezes). Dentre esses módulos, há alguns específicos para configurar a forma com que o servidor trata as requisições do cliente, variando entre os diferentes sistemas operacionais nos quais o servidor Apache é instalado. Estes módulos são chamados de Módulos de Multi-processamento - MPMs.

2.2 Módulos de Multi-processamento (MPMs)

Módulos de Multi-processamento são módulos selecionáveis pelo administrador do servidor para configurar a forma de funcionamento do Servidor *Web* Apache. Estes módulos pretendem se adequar às necessidades de cada sistema operacional no qual o servidor Apache é instalado, tornando-o mais eficiente, uma vez que o programa pode ser executado em diferentes plataformas e ambientes.

Os MPMs são responsáveis pela ligação com as portas de comunicação do computador, aceitação de requisições e pelo gerenciamento de processos e *threads* para o tratamento destas requisições. Para cada um dos MPMs, existe uma lista de diretivas (parâmetros de configuração) que definem o comportamento do servidor.

Para que se apliquem as técnicas de identificação de sistemas, é necessário primeiramente explicar-se sobre os parâmetros de configuração relativos ao módulo chamado *prefork*. Este módulo é carregado por padrão para as plataformas Linux, que foi utilizada neste estudo, através da distribuição KUBUNTU 7.04. para instalação do servidor Apache.

2.3 Parâmetros de configuração do Servidor *Web* Apache para plataforma Linux

2.3.1 Diretiva *KeepAliveTimeout* (comum a todos os módulos)

O parâmetro *KeepAliveTimeout* define a quantidade de segundos que uma conexão permanecerá ativa, aguardando uma próxima requisição do cliente, ou seja, por quanto tempo estará no estado *wait*. Se dentro deste intervalo de tempo, não houver alguma requisição, então a conexão é encerrada. Este é um recurso útil, pois impede que conexões encerrem de forma brusca, obrigando um usuário a constantemente reiniciar a conexão (o que leva certo tempo) bem como impede que um usuário, que já tenha encerrado seus serviços, permaneça conectado indefinidamente, consumindo recursos de memória e CPU do servidor. Deve-se lembrar que uma *thread* somente pode atender um cliente por vez, enquanto permanecer nos estados *wait* ou *busy*. Uma possível estratégia para melhorar o desempenho do servidor, em termos de uso de CPU, consistiria em ajustar automaticamente o valor corrente do parâmetro *KeepAliveTimeout*. Tal ação de controle permitiria liberar parte considerável de recursos de CPU consumidos por processos em estado de espera. No entanto, deve-se observar se o valor do parâmetro *KeepAliveTimeout* se tornar muito elevado, o sistema inteiro poderá ficar ocioso, subutilizando os recursos de CPU.

2.3.2 Diretiva *MaxClients*

Este parâmetro determina a quantidade de conexões simultâneas que podem ser estabelecidas pelo servidor, em um dado instante. Como no módulo *prefork* cada conexão é tratada exclusivamente por um processo-filho dedicado, o valor corrente de *MaxClients* é usado para se ajustar o número máximo de processos-filho que podem ser criados.

Percebe-se que, se *MaxClients* for ajustado para um valor muito elevado, uma considerável parcela de memória RAM do computador poderá ser consumida, já que cada novo processo-filho demandará o uso exclusivo de uma certa parcela de memória RAM, para manter suas variáveis locais. Além disto, poderá ocorrer uma proliferação de novos processos-filhos disputando o uso de recursos de CPU. Caso esta situação for mantida, a partir de um determinado limiar o desempenho do sistema como um todo seria drasticamente reduzido, afetando adversamente a qualidade do serviço.

Por outro lado, se o valor de *MaxClients* for ajustado para um muito reduzido, o tempo de espera para o atendimento de requisições aumentaria consideravelmente, devido à existência de apenas alguns poucos processos-filhos ativos. Além disso, o tempo médio para o estabelecimento de novas conexões de clientes poderia ficar demasiadamente elevado.

Para avaliar a influência do parâmetro *MaxClients* sobre a utilização dos recursos de CPU do computador, foi efetuado um experimento o qual consistiu em variar gradativamente o valor de MaxClients e registrar o correspondente consumo de recursos de CPU pelo servidor. Conforme mostrado na Figura 2, o valor de *MaxClients* foi variado desde um valor mínimo, igual a *MaxClients*=150, até atingir o valor máximo de *MaxClients*=650. O valor de *MaxClients*=150 corresponde a uma condição operacional na qual o uso de CPU pelo servidor Apache é mínimo (quase 0% de uso de CPU), enquanto que o valor de *MaxClients*=650, por sua vez, corresponde a uma condição operacional na qual o consumo de recursos de CPU, pelo servidor Apache, é máximo (quase 100% dos recursos de CPU são utilizados pelo servidor). Conforme pode ser observado na Figura 2, durante o intervalo de duração do experimento o valor de *MaxClients* foi gradativamente incrementados em passos 50 unidades, a cada intervalo de 480 segundos.

Na Figura 3 é mostrada a resposta dinâmica do consumo de recursos de CPU, correspondente ao padrão de variação no valor de *MaxClients* mostrado na Figura 2. Pode-se observar que, à medida que o valor de *MaxClients* aumenta, cresce também o percentual de recursos de CPU utilizado pelo servidor Apache. Também é possível observar que a relação existente entre o consumo de CPU e o valor de *MaxClients* é bastante não-linear, com os efeitos dessa não-linearidade mostrando-se muito mais acentuados quando o sistema opera com um valor elevado de *MaxClients*, tipicamente para valores de *MaxClients* acima de 350, conforme mostrado na Figura 3.

Além disto, observa-se, na Figura 3, que uma mudança no valor de *MaxClients* (ou seja, uma mudança de ponto de operação) dará origem a um período transitório no qual observa-se um comportamento oscilatório no consumo médio de CPU. Após cessado esse período transitório, o valor médio de consumo de CPU atinge um novo valor de regime permanente.

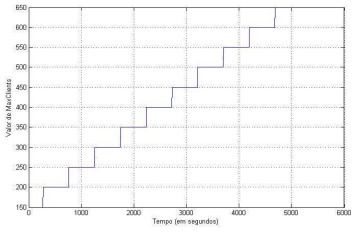


Figura 2: Valores de configuração para o parâmetro *MaxClients*.

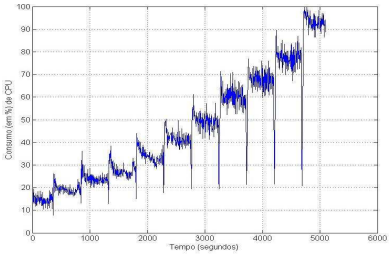


Figura 3: Consumo (em %) da CPU, quando o parâmetro *MaxClients* é variado.

Ao se tentar identificar um modelo para o sistema, percebeu-se que os transitórios obtidos ao se mudar os parâmetros de configuração do servidor e reiniciá-lo influenciavam demasiadamente, de forma negativa, no processo de identificação do sistema. Isto ocorre devido ao fato de estas grandes oscilações conduzirem o sistema a operar na região não-linear. Além disto, a partir de valores de consumo de CPU maiores que 90% e de um número máximo de clientes mais elevado que 600, este comportamento não-linear é ainda mais acentuado. Isto dificulta ainda mais o processo de obtenção de um modelo matemático linearizado para o sistema. Além de que, a partir de 90% de utilização da CPU, o sistema pode não mais responder a todas as requisições.

Como não foi possível variar, para este estudo, os parâmetros de configuração do servidor Apache em tempo de execução, admitiu-se apenas entradas do tipo degrau (independentes umas das outras) e aplicou-se o processo de identificação paramétrica para cada um destes pontos de operação.

2.4 Sinais de sistema no Linux

O sistema operacional LINUX possui ferramentas para o controle de execução dos processos. Isso é feito através de diversos sinais de sistema. Tais sinais correspondem a comandos enviados aos processos, com o objetivo de fazê-los se comportar da forma desejada pelo usuário ou por outro processo que tenha permissão para modificar o estado de outros processos.

Por exemplo, ao se enviar um sinal STOP para um processo, garante-se que ele não será processado pela CPU enquanto este não receber um sinal CONT para liberá-lo. Ao se enviar um sinal KILL, um processo será destruído e seu espaço em memória será liberado.

Neste estudo, foi necessário desenvolver uma macro para efetuar, a cada 5 minutos, a reescrita automática dos arquivos de configuração do servidor, de modo a alterar dinamicamente o valor do parâmetro *MaxClients*, o qual foi utilizado como entrada para a planta (servidor Apache). Entretanto, uma das dificuldades principais, encontrada nos testes realizados, foi que, para a versão do sistema utilizada nos testes, ocorreu uma indesejada re-inicialização automática do sistema Apache a cada vez que os arquivos de configuração do Apache eram dinamicamente alterados pela macro desenvolvida.

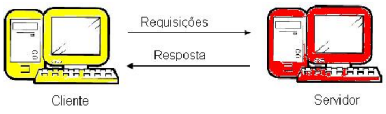
A re-inicialização automática do Apache, a cada alteração dinâmica dos arquivos de configuração, é indesejável porque os usuários que estivessem, naquele momento, recebendo informações do servidor teriam suas conexões subitamente interrompidas. Tal interrupção ocorre porque o processo automático de re-inicialização do Apache destrói todos os processos-filhos em execução naquele momento. Assim sendo, para completar sua tarefa, cada usuário necessitaria re-estabelecer uma nova conexão ao servidor e reiniciar toda transação, o que poderia levar até mesmo a desistência, por parte do usuário, no caso de transação comercial, por exemplo.

A solução adotada, para lidar com o problema da re-inicialização automática do servidor Apache, foi programar uma macro para reler os arquivos de configuração de modo a possibilitar o tratamento das requisições pendentes antes de

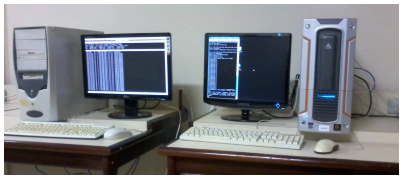
destruir os processos trabalhadores ativos. Para este fim, a macro desenvolvida envia um comando do tipo USR1 para o processo-pai do servidor Apache (o processo- pai é o processo principal do Apache, responsável gerenciar a criação e destruição de processos destinados a lidar com as conexões). O comando USR1 tem a finalidade de fazer com que o processo-pai providencie para que cada processo filho conclua satisfatoriamente o atendimento das conexões ativas antes de destruí-los. Mesmo assim ainda poderá haver uma queda no desempenho do servidor durante o intervalo de tempo necessário para destruição de processos antigos e substituição por novos. Este fenômeno pode ser observado nos transitórios mostrados na Figura 3, após cada mudança de ponto de operação.

2.5 Processo de geração de carga no servidor e leitura de valores

Como forma de se gerar as requisições necessárias para estimular o consumo de CPU por parte da máquina onde o servidor Apache é executado, utilizou-se um computador auxiliar, conectado em rede ao servidor, que simulava dezenas de usuários solicitando serviços. Estes usuários foram criados através do programa computacional *httperf*, uma ferramenta gratuita para testar o desempenho de sistemas computacionais (Bullock, 2007; HPLabs, 2010), conforme ilustrado nas Figuras 4(a) e 4(b).



(a)



(b)

Figura 4: (a) Modelo da rede para geração de cargas. (b) Arranjo experimental para geração de cargas.

Foram utilizadas duas máquinas na rede em questão, conforme a Figura 4(b) 4b. As especificações dos computadores podem ser observadas na tabela 1.

O *httperf* permite que se configure o número de usuários que podem ser simulados. Para isto, pode-se criar sessões a uma dada taxa, as quais simulam estes usuários. Na verdade, cada sessão corresponde a um cliente que deseja se conectar ao servidor. Se houver possibilidade deste cliente se conectar, então uma conexão persistente é estabelecida, permitindo ao usuário fazer requisições de serviço.

Tabela 1: Características dos computadores Cliente e Servidor.

Características	CLIENTE	SERVIDOR
Processador	Intel Dual-core 1.66 GHz	Intel Pentium 4 2.6 GHz
Memória RAM	1 GB	512 MB
Sistema Operacional	Kubuntu, versão 7.04	Kubuntu, versão 7.04
Conexão Ethernet	10/100 Mbps	10/100 Mbps

As requisições são estruturadas na forma de *bursts*. Um *burst* consiste num conjunto de requisições que são enviadas seqüencialmente pelo cliente. O *burst* simula, na verdade, o ato de se "navegar" por uma página da *web*, pois, ao se clicar sobre um link, o usuário não manda apenas uma requisição de página, mas requisições de imagens, quadros, textos, outros links, arquivos e todos os objetos necessários para exibir corretamente uma página *web*.

Um usuário permanece conectado enquanto estiver interagindo com o servidor ou enquanto seu período de ociosidade não ultrapassar o parâmetro *KeepAliveTimeout*. Dessa forma, pode haver sucessivos *bursts* por um mesmo usuário que são espaçados entre si pelo *think-time*, um intervalo de tempo entre as chamadas seqüenciais. Desta forma, adotou-se os valores exibidos na tabela 2 para o processo de geração de sinais.

Tabela 2: Configurações e condições de operação do gerador de carga.

N^o total de sessões	10000000
Think-time	11
Requisições / sessão	6
Comprimento do *burst*	3
Sessões / segundo	100

A quantidade total de sessões criadas é grande o suficiente para permitir que o teste permaneça em execução por várias horas. O período corresponde à taxa com a qual as sessões são criadas. O valor 100 corresponde a 100 sessões a cada segundo.

É extremamente importante ressaltar que os valores utilizados para a configuração do *httperf* foram escolhidos arbitrariamente, de forma a se obter consumo máximo e mínimo de CPU no servidor, através da manipulação única do parâmetro

MaxClients. Assim, não houve estudos aprofundados sobre modelos de geração de cargas para servidores, teoria de filas ou processos estocásticos voltados para esta ferramenta.

Foi adotada uma mesma condição de operação para o gerador de carga em todos os testes realizados, conforme a tabela2.

Para a leitura dos valores dos sinais de entrada e saída, além do processo de envio do sinal USR1 periodicamente ao servidor Apache, foi escrito um programa em linguagem de programação C capaz de realizar tais tarefas. Este programa coletou os dados de utilização de CPU a cada 5 segundos, durante 50 minutos, para cada ponto de operação estudado. A leitura destes valores foi feita com este intervalo para que o programa desenvolvido não influenciasse nas próprias medições de CPU, o que aconteceria se este intervalo fosse definido em milissegundos.

3 IDENTIFICAÇÃO PARAMÉTRICA DE SISTEMAS

Conforme já comentado na subseção 2.1 deste artigo, a dinâmica de um servidor Apache é complexa e de difícil modelagem através de primeiros princípios, pois se trata de um sistema a eventos discretos, com características bastante variantes no tempo. Dessa forma, neste trabalho utilizou-se a metodologia baseada em estimação paramétrica de modelos dinâmicos representando a relação entrada-saída para um servidor Apache. Foram utilizados modelos paramétricos do tipo auto-regressivos (AR). Estes são modelos lineares cujos parâmetros podem ser estimados a partir de conjuntos de dados de entrada e saída do sistema, com base nas técnicas de mínimos quadrados, tendo a seguinte estrutura:

$$y(k) = -\hat{a}_1 y(k-1) - \hat{a}_2 y(k-2) - \ldots - \hat{a}_n y(k-n) + \xi(k) \quad (1)$$

em que $y(k-i)$, $i = 0, 1, 2, \ldots, n$ representa os valores presentes e passados da saída da planta e os parâmetros $\hat{a}_i$, $i = 1, 2, \ldots, n$, são os parâmetros a serem estimados para o modelo AR da planta, utilizando-se a técnica de mínimos quadrados não-recursivo (Aguirre, 2007; Coelho and Coelho, 2004; Ljung, 1999). Na equação (1), $\xi(k)$ é uma seqüência aleatória, do tipo ruído branco, modelando o erro de estimação entre a saída da planta e a saída do modelo AR.

O modelo paramétrico da equação (1) pode ser colocado sob a seguinte forma:

$$y(k) = \psi^T(k-1)\hat{\theta} + \xi(k) \quad (2)$$

onde $\psi^T(k-1) = [-y(k-1) \ \ldots \ -y(k-n)]$ é o vetor regressores e $\hat{\theta} = [\hat{a}_1 \ \ldots \ \hat{a}_n]^T$ é o vetor de parâmetros a ser estimado.

A função custo para modelos AR pode ser expressa da seguinte forma:

$$J_{MQ}(\hat{\theta}) = \sum_{k=1}^{N} \xi(k|k-1, \hat{\theta})^2 = \xi^T \xi = ||\xi||^2 \quad (3)$$

sendo que $\xi(k|k-1, \hat{\theta})$ é o erro de predição (ou resíduo) cometido no instante k, ao se fazer a predição baseada na informação coletada até o instante $k-1$, usando-se o valor do vetor de parâmetros estimado $\hat{\theta}$. Dessa forma, o estimador de mínimos quadrados pode ser representado por (Aguirre, 2007; Coelho and Coelho, 2004; Ljung, 1999):

$$\hat{\theta}_{MQ} = [\psi^T \psi]^{-1} \psi^T y \quad (4)$$

Após a estimação do modelo paramétrico, testes de auto-correlação do resíduo $\xi(k)$. podem ser utilizados. De acordo com (Aguirre, 2007, capítulo 12, página 445), "a motivação de se verificar quão aleatórios são os resíduos pode ser entendida lembrando-se que os resíduos são a parte dos dados que o modelo não conseguiu explicar". Dessa forma, intuitivamente, um bom modelo, que capture satisfatoriamente boa parte da informação determinística presente nos dados, deveria apresentar como resíduo uma seqüência aleatória que se aproximasse das características de um ruído branco. Uma forma de se avaliar isso é através do cálculo de uma estimativa da função de auto-correlação do resíduo, $\hat{r}_{\xi\xi}(k)$. Para casos onde existe ergocidade, essa estimativa pode ser calculada utilizando-se a seguinte fórmula:

$$\hat{r}_{\xi\xi}(k) = \frac{1}{2N+1} \sum_{i=-N}^{N} \xi(i)\xi(i+k) \quad (5)$$

onde N é o número de amostras utilizadas na estimação do modelo (Aguirre, 2007). A estimativa é tanto melhor quanto maior for o número de amostras N.

4 PROCEDIMENTO EXPERIMENTAL

O modelo dinâmico do servidor Apache pode ser representado de acordo com a Figura 5

Percebe-se,na Figura 5, que além das entradas u_1=*MaxClients* e u_2=*KeepAliveTimeout*, o nível de carga atuando no servidor também pode ser tratada como uma entrada para a planta. Contudo, optou-se, neste trabalho, por

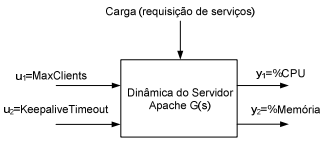


Figura 5: Modelo MIMO do servidor *web* Apache.

considerar a carga como uma perturbação não-mensurável atuando no sistema. A razão para isto é que normalmente o projetista não tem como atuar diretamente sobre seus valores. As variações de carga podem ser de forma aleatória e imprevisível, e estão sujeitas a condições externas ao sistema.

No presente trabalho, optou-se por trabalhar com uma representação SISO, do tipo mostrado na Figura 5, como uma etapa inicial visando o futuro desenvolvimento de um sistema de controle automático para o servidor Apache. Considerou-se, então, o parâmetro *MaxClients* como a entrada e o percentual de utilização de CPU como saída do sistema. Escolheu-se esse parâmetro de configuração para esta etapa inical devido ao seu forte acoplamento com o percentual de utilização de CPU, como foi mostrado em (Diao et al., 2002).

Para realizar a identificação do servidor com os dados coletados, utilizou-se a ferramenta computacional MATLAB® versão 7.0 através de seu *System Identification Toolbox* - IDENT, que consiste em um programa que acompanha o MATLAB, cuja função é auxiliar na identificação de sistemas, baseado nos sinais de entrada e saída oferecidos pelo usuário (neste caso, *MaxClients* e consumo de CPU, respectivamente).

Para o estudo desenvolvido aqui, realizou-se um processo de identificação para as saídas associadas a quatro valores distintos de *MaxClients*: 150, 300, 450, 600. Para os resultados obtidos, fez-se as devidas análises com o intuito de se descobrir se o processo de identificação utilizado foi satisfatório. Foram coletadas 600 amostras do sinal de saída da planta. Este conjunto de dados foi separado em dois sub-conjuntos, cada um com 300 amostras. O sub-conjunto composto pelas 300 primeiras amostras foi utilizado no processo de estimação dos parâmetros do modelo, enquanto que a o sub-conjunto composto pelas 300 amostras restantes foi utilizado no processo de validação do modelo estimado. A predição é de um passo à frente e foi feita com o sub-conjunto de dados de saída que foi utilizado para validação do modelo estimado.

primeiras amostras foi utilizado no processo de estimação dos parâmetros do modelo, enquanto que a o sub-conjunto composto pelas 300 amostras restantes foi utilizado no processo de validação do modelo estimado. A predição é de um passo à frente e foi feita com o sub-conjunto de dados de saída que foi utilizado para validação do modelo estimado.

4.1 Identificação 1: *MaxClients* = 150, *KeepAliveTimeout* = 5s

Para este primeiro caso, considerou-se a entrada *MaxClients* como sendo constante durante todo o experimento. A saída do processo corresponde ao consumo de CPU. O valor de *MaxClients*, para o ponto de operação deste teste, foi ajustado para *MaxClients*=150. Na Figura 6 mostra-se a saída real do sistema. Nota-se que o sinal de saída apresenta alguma não-estacionariedade. Possivelmente, isto se deveu tanto ao fato de que a dinâmica do servidor Apache ser altamente não-linear, conforme já mostrado (ver Figura 3). No entanto, observa-se que o valor médio do sinal varia lentamente no intervalo de tempo do ensaio, 3000 segundos. Dessa forma, é razoável supor que um modelo AR a parâmetros fixos possa apresentar desempenho razoável para pequenas variações de carga. Para grandes variações de carga, seria necessário desenvolver um modelo não-linear, possivelmente uma rede de modelos locais com um mecanismo de chaveamento entre modelos previamente identificados para diferentes pontos de operação, utilizando metodologias baseada em multi-modelos (Murray-Smith and Johansen, 1997; Barra Jr et al., 2005). Este aspecto não-linear da modelagem do sistema Apache não foram abordados no presente artigo.

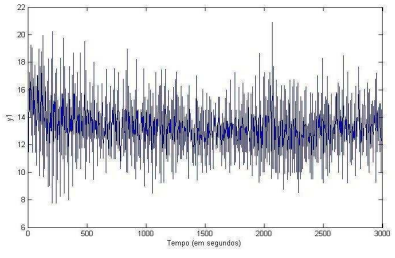


Figura 6: Saída real do sistema para *MaxClients* igual a 150.

Foram testados modelos candidatos com estrutura AR, com ordem variando de 1 a 10. Com base na análise da função custo para modelos AR, com diferentes parâmetros, foi possível a identificação da melhor estrutura para o modelo, conforme mostrado na Figura 7.

Observa-se na Figura 7 que o melhor modelo é o de 7^a ordem. Esta análise é feita observando-se o eixo Y do gráfico que mostra o erro quadrático obtido com a identificação (função custo). Quanto maior o valor da função custo, pior é o modelo candidato. Dessa forma, o modelo estimado foi o seguinte:

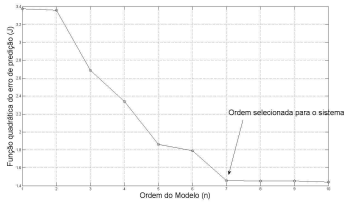


Figura 7: Função quadrática do erro de predição (*MaxClients* = 150).

$$\begin{aligned}\hat{y}(k) \quad = \quad & -0,3009y(k-1)+0.1351y(k-2) \\ & -0,02197y(k-3)-0,05187y(k-4) \\ & +0,2138y(k-5)-0,03788y(k-6) \\ & +0,4073y(k-7)\end{aligned} \tag{6}$$

Onde $\hat{y}(k)$ é a estimativa do desvio da saída da planta em torno de seu valor médio no ponto de operação onde foi feita a estimação.

Para se validar o modelo de 7^a ordem estimado, foram utilizados testes baseados nas estimativas das funções de auto-correlação das amostras da saída da planta, Figura 8, e das amostras do resíduo, Figura 9. Observando-se a Figura 8, nota-se que existe uma correlação estatisticamente significativa entre as amostras da saída da planta, para diversos valores de atraso de amostragem. Isso significa que um comportamento dinâmico determinístico está presente nos dados de saída da planta e que este pode ser capturado por um modelo AR de ordem adequada.

Na Figura 9, por sua vez, observa-se que a estimativa da função de auto-correlação entre as amostras do resíduo apresentou valores confinados dentro de um nível de confiança de 99,7%, para praticamente todos os atrasos de amostragem, com exceção do atraso 0, como era de se esperar. Isso demonstra o bom desempenho do modelo estimado, o qual foi capaz de capturar boa parte do comportamento determinístico presente nas amostras de saída da planta.

Na Figura 10, mostra-se a comparação entre a saída real e a saída estimada pelo modelo de 7^a ordem. Observa-se que o modelo identificado é capaz de acompanhar com eficiência as variações observadas no sistema real.

4.2 Identificação em outros pontos de operação.

Testes adicionais, semelhantes aos descritos na sub-seção 4.1, foram também realizados para outros pontos de operação. Para esses testes, o valor do parâmetro *MaxClients* foi

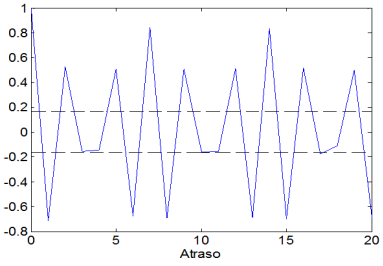


Figura 8: Estimativa da função de auto-correlação da saída (para *MaxClients*=150), intervalo de confiança de 99,7%.

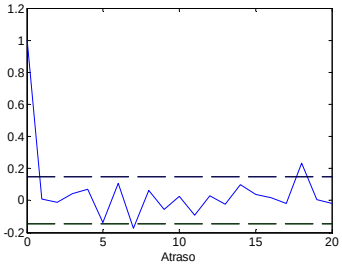


Figura 9: Função de auto-correlação do resíduo (*MaxClients*=150), intervalo de confiança de 99,7%.

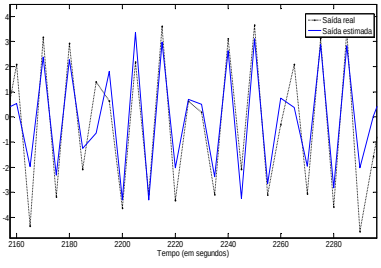


Figura 10: Aproximação das saídas real (–) e estimada (-) (*MaxClients* = 150).

ajustado, em cada teste, para os valores *MaxClients* = 300, 450 e 600, respectivamente. Para todos os testes apresentados neste artigo, o valor do parâmetro *KeepAliveTimeout* foi mantido fixo no valor *KeepAliveTimeout* = 5s.

Observando-se os resultados apresentados nas Figuras 11, 12 e 13, para a função quadrática do erro de predição, nota-se que modelos de 7^a ordem já não são mais satisfatórios em pontos de operação caracterizados por um valor elevado do parâmetro *MaxClients*. Observa-se que, à medida que se aumenta o valor de *MaxClients*, a ordem do modelo AR também deverá aumentada, sendo necessário o uso de modelos mais complexos (com um número maior de parâmetros a estimar). Esses resultados confirmam as características altamente não-lineares da dinâmica do servidor Apache.

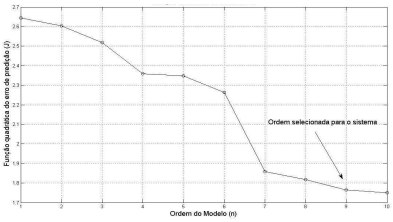


Figura 11: Função quadrática do erro de predição (*MaxClients* = 300).

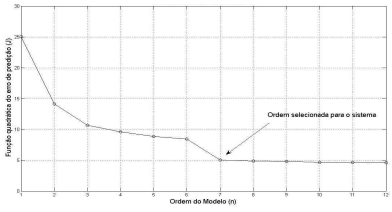


Figura 12: Função quadrática do erro de predição (*MaxClients* = 450).

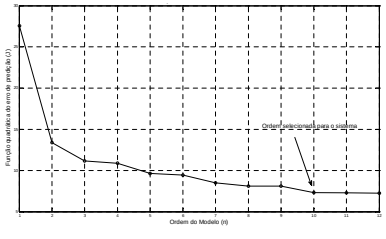


Figura 13: Função quadrática do erro de predição (*MaxClients* = 600).

5 CONCLUSÕES

O estudo relatado neste artigo apresentou uma metodologia baseada em identificação paramétrica para estimar modelos dinâmicos do comportamento de um servidor Apache. Percebeu-se que a alteração das características de funcionamento do servidor modifica de forma acentuada o comportamento global do mesmo. Mostrou-se também que o comportamento dinâmico do servidor pode ser satisfatoriamente representado por um modelo paramétrico auto-regressivo quando a planta estiver operando em torno de um ponto de operação definido pelos valores médios de *MaxClients* e de percentual de consumo de CPU.

Através de um arranjo experimental desenvolvido para simular múltiplos usuários solicitando acesso ao servidor e também da análise da carga medida de CPU do sistema ante tais solicitações, foi possível desenvolver-se um estudo de identificação de um sistema computacional (sistema real) de forma satisfatória. Isto permitiria, futuramente, realizar testes de controladores digitais, como forma de se garantir que um sistema computacional opere dentro de uma faixa desejada e configurada pelo usuário.

Nos testes apresentados neste artigo, não foi possível variar dinamicamente o valor da entrada *MaxClients*, conforme era desejado. Dessa forma, com a medida dinâmica apenas do sinal de saída, somente modelos com estrutura AR puderam ser estimados. Essa limitação ocorreu devido ao fato de que a versão do Apache com a qual os autores trabalharam efetua uma re-inicialização automática do servidor a cada vez que os arquivos de configuração são alterados dinamicamente pela macro de controle, interrompendo indevidamente todas as conexões ativas. Os autores continuam investigando este ponto na busca de uma solução que permita alterar dinamicamente o parâmetro *MaxClients* sem re-inicializar o sistema. Quando isto tiver sido feito, será possível projetar uma lei de controle digital, programado na forma de uma macro no servidor Apache, para regular dinamicamente o valor de consumo percentual de CPU, atuando-se no valor do parâmetro *KeepAliveTimeout*. Assim que novos progressos forem obtidos, os autores divulgarão os resultados de testes de controle em futuros artigos.

Como outra contribuição, este estudo mostrou que os métodos de identificação de sistemas podem ser muito úteis como ferramentas de estudo da dinâmica de sistemas computacionais, já que a modelagem desses sistemas através de primeiros princípios é de grande dificuldade devido ao desconhecimento de diversos parâmetros internos do sistema e também devido a sua característica altamente não-linear. Com isto, pretende-se estimular o estudo da melhoria de desempenho de outros sistemas computacionais, evitando-se que tal melhoria seja realizada apenas através de superdimensionamento de hardware (atualmente, a forma mais utilizada para o aumento de desempenho de sistemas computacionais).

REFERÊNCIAS

Abreu, T., Barra Jr, W., Barreiros, J. and Costa Jr, C. (2009). Técnica de identificação paramétrica aplicada ao servidor web apache, *VI Seminário Nacional de Controle e Automação Industrial, Elétrica e de Telecomunicações*, Salvador, Brasil, pp. 88–93.

Aguirre, L. (2007). *Introdução à Identificação de Sistemas: Técnicas Lineares e Não-Lineares Aplicadas a Sistemas Reais*, 3 edn, Editora UFMG, Belo Horizonte.

Alvarez, T., Herrero, D., Francisco, J. and Madrigal, D. (2010). How can apache help to teach and learn automatic control?, *IEEE Education Engineering (EDUCON), 2010*, pp. 1539 –1546.

Apache (2010). Apache software foundation - the apache http server project, Disponível em http://www.apache.org. Acessado em 30/03/2010.

Barra Jr, W., Barreiros, J., Costa Junior, C. and Ferreira, A. (2005). Controle fuzzy aplicado à melhoria da estabilidade dinâmica de sistemas elétricos de potência, *Controle & Automação* **16**(2): 173–186.

Bazanella, A. and Silva Jr., J. (2005). *Sistemas de Controle: princípios e métodos de projeto*, Editora UFRGS.

Bullock, T. (2007). httperf - web workload generator - quick start guide, Disponível em http://httperf.comlore.com/. Acessado em 01/07/2008.

Coelho, A. and Coelho, L. (2004). *Identificação de sistemas dinâmicos lineares*, 1 edn, Editora da UFSC.

Diao, Y., Gandhi, N., Hellerstein, J. L., Parekh, S. and Tilbury, D. M. (2002). Using mimo feedback control to enforce policies for interrelated metrics with application to the apache web server., *Proceedings of the Network Operations and Management Symposium 2002. NOMS'02*, pp. 219–234.

Gandhi, N., Tilbury, D., Diao, Y., Hellerstein, J. and Parekh, S. (2002). Mimo control of an apache web server: modeling and controller design, *Proceedings of the American Control Conference, 2002.*, Vol. 6, pp. 4922–4927.

Hellerstein, J., Diao, Y., Parekh, S. and Tilbury, D. (2004). *Feedback Control of Computing Systems*, Wiley Interscience, New York.

Hellerstein, J., Diao, Y., Parekh, S. and Tilbury, D. (2005). Control engineering for computing systems - industry experience and research challenges, *IEEE Control Systems* **25**(6): 56–68.

HPLabs (2010). Welcome to the httperfhomepage, Disponível em http://www.hpl.hp.com/research/linux/httperf/. Acessado em 30/03/2010.

Kihl, M., Robertsson, A., Andersson, M. and Wittenmark, B. (2008). Control-theoretic analysis of admission control mechanisms for web server systems, *World Wide Web* **11**(1): 93–116.

Kjaer, M., Kihl, M. and Robertsson, A. (2007). Response-time control of a single server queue, *46th IEEE Conference on Decision and Control, 2007*, pp. 3812–3817.

Kjaer, M. and Robertsson, A. (2010). Analysis of buffer delay in web-server control, *American Control Conference (ACC), 2010*, pp. 1047–1052.

Ljung, L. (1999). *Systems Identification - Theory for the user*, Prentice Hall, Upper Saddle River, NJ.

Murray-Smith, R. and Johansen, T. (1997). *Multiple Models Approaches to Modeling and Control*, Taylor and Francis, London.

NETCRAFT (2010). March 2010 archives, Disponível em http://news.netcraft.com/archives/2010/03/index.html. Acessado em 30/03/2010.

Ogata, K. (2003). *Engenharia de Controle Moderno*, Prentice Hall, Sao Paulo.