Avaliação Experimental do Índice de Carga Usado no Kernel Linux

Rivalino Matias Jr, Leonardo Alt, Otávio Augusto

Faculdade de Computação – Universidade Federal de Uberlândia (UFU) –
38400-902 – Uberlândia – MG – Brasil

rivalino@facom.ufu.br, {leonardoalt,octopos}@comp.ufu.br

Abstract. *In this paper we experimentally evaluate the quality of loadavg index available at Linux kernel. The research hypothesis is that loadavg is not adequate to characterize scenarios with considerable kernel-level activity. For specific classes of Linux-based systems (e.g., firewalls), the system load in kernel level may be significantly higher than in user level. The experimental results indicate that loadavg is not able to appropriately represent scenarios of high kernel-level workload, what confirms the evaluated hypothesis.*

Resumo. *Neste artigo avaliamos experimentalmente a qualidade do loadavg implementado no kernel Linux. A hipótese de pesquisa é de que esse índice de carga não é adequado para caracterizar cenários com significante atividade em nível de kernel. Para alguns tipos de sistemas baseados em Linux (ex. firewals), a carga em nível de kernel pode ser significantemente maior do que em espaço de usuário. Os resultados experimentais mostraram que o loadavg não é capaz de representar apropriadamente cenários com carga elevada em nível de kernel, confirmando a hipótese de pesquisa avaliada.*

1. INTRODUÇÃO

Índices de carga são informações de estado mantidas pelo *kernel* de um sistema operacional. Esses índices indicam a carga corrente do sistema e são muito usados para decidir sobre a capacidade e disponibilidade de um sistema em receber novas tarefas, ou mesmo da necessidade de remanejar tarefas de um sistema para outro. Atualmente, uma das aplicações práticas dos índices de carga é o balanceamento de carga em sistemas distribuídos. A literatura cita suas aplicações no escalonamento de tarefas em *clusters* de computadores [Sit *et al*. 2004] e *Grids* Computacionais [Payli *et al*. 2004].

De forma geral, o conceito de carga (*load*) diz respeito ao uso dos recursos de processamento de um sistema. Na literatura, existem diversos trabalhos que investigam diferentes abordagens para quantificar a carga de um sistema. Dentre eles, destaca-se o trabalho de Ferrari e Zhou (1987) que realiza uma comparação entre diversas abordagens para cálculo do índice de carga em sistemas Unix. São comparados índices baseados apenas na utilização de CPU com índices baseados no tamanho de filas de tarefas aguardando por CPU, I/O de disco, e recursos de memória. Os autores concluem que índices baseados no tamanho de filas são melhores do que apenas na utilização de recursos. Índices baseados apenas no uso de recursos não conseguem quantificar o tamanho da contenção que ocorre em casos de alta carga de trabalho, onde

o recurso (ex. CPU) está com sua utilização próxima ou igual a 100%. Nesses casos, o tamanho da contenção (número de processos aguardando pelo recurso) não é considerado, mas apenas a ocupação do recurso. Por exemplo, dois sistemas idênticos apresentam 100% de utilização de CPU. Um índice baseado na utilização da CPU indicaria que os dois sistemas têm o mesmo nível de carga. Já um índice baseado no número de tarefas aguardando na fila da CPU permitiria concluir que uma nova tarefa deve ser alocada no sistema com menor fila de espera. Ferrari e Zhou também observaram que uma melhor caracterização da carga é obtida com valores médios do tamanho de filas, para períodos curtos (ex. 10 segundos), em alternativa ao uso de valores obtidos de forma instantânea.

Atualmente, o índice de carga usado em sistemas Linux (e também Unix) é o *loadavg* (*load average*) [Gunther 2004], o qual segue a abordagem de filas de recursos descrita anteriormente [Ferrari e Zhou 1987]. Por ser uma métrica de carga tradicional, o *loadavg* é usado por diversas aplicações legadas. Além disso, vários trabalhos acadêmicos (ex. [Ferrari e Zhou 1986], [Zhou 1988], [Chen *et al*. 2006], [Broberg *et al*. 2006]) utilizam os valores do *loadavg*. Nesse contexto, esse estudo avalia a qualidade das caracterizações fornecidas pelo *loadavg* no *kernel* Linux. Durante o levantamento bibliográfico, verificou-se que a maioria dos trabalhos relacionados se concentra em cenários de carga de trabalho predominantemente exercida no espaço do usuário. Portanto, esse estudo considera tanto cargas de trabalho em nível de usuário quanto em nível de *kernel*, com foco para o segundo tipo, já que não foram encontrados trabalhos voltados para esse tipo de caracterização. Cenários com carga elevada em nível de *kernel* são freqüentes em sistemas especializados, tais como sistemas embarcados para aplicações de rede (ex. roteadores), os quais passam a maior parte do tempo executando atividades em nível de *kernel*.

O restante do artigo está organizado como descrito a seguir. A Seção 2 descreve o modelo de índice de carga do *loadavg* e sua implementação no *kernel* Linux. Na Seção 3 são apresentados os detalhes envolvidos na etapa de experimentação, enfatizando a metodologia e os planos experimentais adotados. A Seção 4 apresenta uma análise dos principais resultados obtidos no estudo experimental. Finalmente, na Seção 5 são apresentadas as conclusões desse trabalho.

2. ÍNDICE DE CARGA NO LINUX

Como discutido na Seção 1, o *loadavg* é o índice de carga padrão do *kernel* Linux [Gunther 2004]. De fato, o *loadavg* foi um dos conceitos de instrumentação implementados no Multics [Saltzer e Gintell 1970], sistema operacional predecessor da família de sistemas operacionais Unix. Sistemas *Unix-like* implementam essa métrica e a disponibilizam por meio de programas utilitários (ex. uptime) e, no caso do Linux, também via *procfs* (/proc/loadavg). O *loadavg* é composto por três valores que caracterizam a carga do sistema em intervalos de 1, 5 e 15 minutos. Esses valores iniciam em zero, variando de acordo com a carga do sistema. Considerando a carga do último minuto, *loadavg_1*, um valor de 0.5 indica que o sistema tem uma carga que, em média, ocupa 50% da sua capacidade de processamento. Por conseguinte, um *valor* de 1.5 indica uma carga adicional de 50% além da capacidade de processamento do sistema.

No Linux, o *loadavg* é implementado como um vetor de três elementos do tipo *unsigned long*. Cada posição do vetor armazena um número, em notação de ponto fixo, representando um valor real, e correspondendo a um dos três intervalos descritos anteriormente. Como exemplo, a função que calcula o *loadavg_1* segue o modelo apresentado na Equação 1 [Gunther 2004].

$$load(t) = load(t-1)e^{-5/60} + n(1-e^{-5/60}), \quad (1)$$

onde *n* é o número de processos ativos, ou seja, na fila de execução (TASK_RUNNING) ou bloqueados em eventos "não interruptíveis" (TASK_UNINTERRUPTIBLE) [Bovet e Cesati 2005]. No Linux o valor de *n* é amostrado a cada 5 segundos. Portanto, conclui-se que (1) é uma média móvel exponencial onde a carga corrente, load(*t*), é igual ao último valor calculado de *loadavg_1* multiplicado por uma constante de suavização exponencial, e-5/60, e somado ao número de processos ativos ponderado pela constante de suavização apropriada ao período de 1 minuto. O mesmo algoritmo é usado para o cálculo de *loadavg_5* e *loadavg_15*, mudando-se apenas as constantes de suavização para e-5/300 e e-5/900, respectivamente. Maiores detalhes sobre essas constantes estão disponíveis em [Gunther 2004]. O propósito desse modelo de média móvel não é apenas suavizar exponencialmente os extremos, mas também reduzir a importância dos valores menos recentes em relação aos novos valores de carga, visto que eles têm maior influência no estado corrente do sistema. Como pode ser observado em (1), o modelo de carga do *loadavg* considera apenas a quantidade (*n*) de atividades que são executadas em contexto de processos. O número de atividades aguardando para executar tarefas do *kernel*, principalmente em *bottom halves* [Love 2003] e usualmente no contexto de interrupções, não são consideradas. No *kernel* Linux, diversos mecanismos são usados para executar tarefas *bottom halves*, a saber: *work queues*, *tasklets*, *softirqs*, e *timers* [Wilcox 2003]. Esses mecanismos são usados principalmente por controladores de dispositivos, onde alguns desses mecanismos (ex. *timers* e *work queues*) possuem filas de execução próprias e que não são consideradas no cálculo do *loadavg*. Como citado anteriormente, cargas de trabalho consideráveis, em nível de *kernel*, estão presentes em sistemas especializados, tais como *bridges*, filtros de pacotes, controladores de banda, dentre outros. Adicionalmente, sistemas com funcionalidades como NFS e RAID, implementadas no *kernel*, também apresentam elevadas cargas em nível de *kernel*. Com base nessa análise teórica do *loadavg*, a etapa experimental desse estudo investiga tanto cargas de trabalho em nível de usuário quanto de *kernel*.

3. ESTUDO EXPERIMENTAL

Nessa etapa avaliou-se a qualidade do índice *loadavg* no *kernel* Linux. A metodologia adotada baseou-se na realização de experimentos controlados, utilizando 10 replicações para cada tratamento [Montgomery 2005]. Dessa forma, buscou-se reduzir a influência dos efeitos experimentais presentes em cada execução. Cada replicação de um tratamento executa por um período de 20 minutos. O plano experimental contou com tratamentos que testa o *loadavg* em diferentes cenários de carga, tanto em nível de usuário quanto de *kernel*. Para o nível de usuário, implementou-se um programa de multiplicação de matrizes. Para o nível de *kernel* foram usadas *works* executando uma espera-ocupada de 10ms e, posteriormente, se auto re-escalonando (*schedule_work*()). Além disso, foram avaliadas cargas de trabalho constante e variável, sendo essa última implementada como um incremento do número de processos/*works*. Todos os testes foram realizados com um número reduzido de processos padrões do sistema (*runlevel* 3). O ambiente de teste foi um computador com processador Intel Core 2 Quad de 2.66 GHz e 2 Gb RAM, usando apenas um núcleo habilitado, sendo os demais desativados [Raj 2010]. A versão de *kernel* usada foi a 2.6.33.3, obtida em *www.kernel.org* e compilada com as configurações padrões. A seguir uma descrição dos tratamentos.

No primeiro tratamento (T1), o *loadavg* foi monitorado com o sistema em repouso (sem carga de trabalho). Os próximos 9 tratamentos adotaram cargas constantes, onde T2 até T5

utilizam cargas em nível de usuário: T2 (1 processo), T3(5), T4(10), T5(15). Já T6 até T9 adotam cargas em nível de *kernel*: T6(1 *work*), T7(5), T8(10), T9(15). Os demais tratamentos implementaram carga variável, onde T10 utilizou uma carga em nível de usuário iniciando em 1 processo e incrementando 5 processos a cada 5 minutos. Em T11, repetiu-se o mesmo procedimento de T10, contudo para cargas em nível de *kernel* (*works*). A próxima seção apresenta os resultados obtidos.

4. ANÁLISE DOS RESULTADOS

Para cada tratamento calculou-se a média aritmética das 10 replicações. Também, calculou-se o desvio padrão da amostra (médias das 10 replicações) para se observar a variabilidade dos dados. Devido à limitação de espaço, serão apresentados apenas os resultados do *loadavg_1*. Os demais valores (5 e 15 min.) corroboram o padrão apresentado pelo *loadavg_1*. O valor médio para o *loadavg_1* com o sistema sem carga foi de 0,1057. Idealmente, esse valor deveria ser zero, contudo os diversos processos (ex. *agetty*) e *threads* (ex. *ata/0*) do sistema, presentes em qualquer cenário, influenciam nesse valor. As Tabelas 1 e 2 apresentam os resultados dos tratamentos T2 a T9.

Tabela 1. Workload Constante em user level

Trat.	Média do loadavg_1	D.P.
T2	1,0697	0,0161
T3	5,1478	0,0899
T4	10,4412	0,1149
T5	15,7251	0,2137

Tabela 2. Workload Constante em kernel level

Trat.	Média do loadavg_1	D.P.
T6	4,5333	0,1169
T7	4,3739	0,0919
T8	4,1501	0,0656
T9	3,8855	0,0489

Na Tabela 1, o valor do *loadavg_1* acompanha linearmente o incremento da carga de trabalho no nível de usuário. Já o incremento da carga no nível de *kernel* não foi caracterizado adequadamente, pois o *loadavg_1* não acompanha o crescimento da carga de trabalho (ver Tabela 2). Por exemplo, T9 adota 15 *works* e T6 apenas uma. O valor de 4,5 para T6 é explicado pelo acúmulo de processos do sistema (em nível de usuário) aguardando para executar. Esses processos devem aguardar a execução da *work* criada em T6, pois a *thread* de *kernel*, *events/0,* quem executa o código da *work*, é enfileirada na mesma fila de processos. Diferentemente das *works*, as *tasklets*, *softirqs* e *timers* não são servidas por uma *thread* de *kernel* e, portanto, são totalmente transparentes para o índice de carga *loadavg*, reduzindo ainda mais a sua capacidade de caracterização de carga em nível de *kernel*. Observa-se nas Tabelas 1 e 2 que os valores de desvio padrão (D.P.) indicam uma baixa variabilidade entre as replicações.

Outra avaliação do *loadavg* foi quanto à sua capacidade de capturar cargas incrementais. As Figuras 1 e 2 apresentam os resultados para os tratamentos T10 e T11. Verifica-se que para cargas em nível de usuário, o *loadavg* acompanhou o crescimento dinâmico da carga, ao contrário do cenário para cargas em nível de kernel, onde o valor do *loadavg* não acompanhou o crescimento a partir dos primeiros cinco minutos.

Como osbervado, o *loadavg* mostrou-se inadequado para cenários de carga em nível de *kernel*. A fim de identificar os subsistemas que mais agendam atividades de computação em nível de *kernel*, realizou-se uma pesquisa no código fonte do *kernel* usado nos experimentos, a fim de localizar chamadas de criação de *work queues*, *tasklets*, *softirqs*, e *timers*. Verificou-se que 96,49% de todas as chamadas concentramse em três subsistemas: *device drivers* (76,9%),

network (16,23%), *filesystem* (3,37%). Nesses três subsistemas, fica claro que o mecanismo mais usado são as *work queues* (46,31%), seguido de *timers*(37,66%) e *tasklets*(16,03%), como apresentado na Figura3.

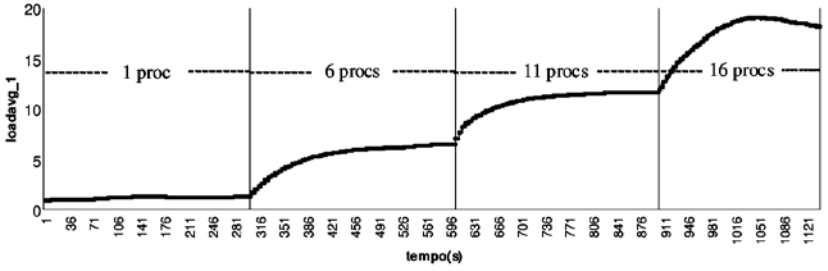

Figura 1. Carga incremental em nível de usuário (*user level*)

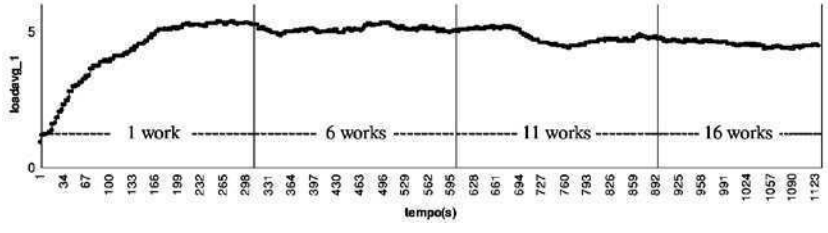

Figura 2. Carga incremental em nível de kernel (*kernel level*)

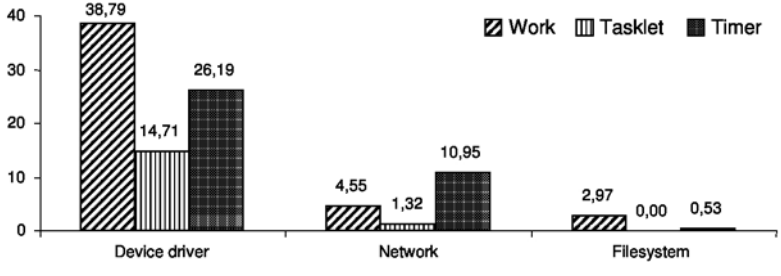

Figura 3. Distribuição dos mecanismos de execução em nível de kernel

No total de 1571 chamadas, apenas 4 são relativas a *softirqs*, e nenhuma localizada nos três subsistemas citados anteriormente. Esse resultado confirma a tendência em se usar *work queues* para agendamento de tarefas *bottom halves*. Apesar disso, salienta-se que *tasklets* são construídas sobre a infra-estrutura de *softirqs*.

5. CONCLUSÃO

A utilização de índices de carga em áreas como balanceamento e escalonamento de tarefas é de grande importância no contexto de sistemas computacionais distribuídos. Estudos preliminares comprovam que índices baseados não apenas no uso da CPU, mas principalmente no tamanho de filas de recursos (ex. CPU e I/O), possuem melhor desempenho para a caracterização de carga em sistemas computacionais. Nesse trabalho, verificou-se experimentalmente que o modelo implementado pelo índice de carga *loadavg*, extensivamente usado em sistemas operacionais da família Unix e Linux, não é capaz de capturar, de forma adequada, cenários de carga em nível de *kernel*. Já para cenários de cargas predominantemente em nível do usuário, esse índice mostrou-se satisfatório. Verificou-se também que, atualmente, tem-se uma maior utilização de *work queues* como mecanismo de agendamento de tarefas em nível de *kernel* no Linux. Essas evidências motivam aprofundar a investigação de modelos de índices com capacidade de caracterização acurada para cargas em ambos cenários, com ênfase para o nível de *kernel*, já que não foram encontrados estudos específicos para esse problema.

Desse modo, encontra-se em andamento um trabalho que estende o presente estudo, visando implementar um índice de carga específico para o nível de *kernel*, o qual tem sido chamado de *kloadavg*. Esse novo índice pode ser usado em conjunto com o *loadavg* padrão. Alternativamente, pode-se incorporar o cálculo do *kloadavg* no próprio modelo do *loadavg*. Ambas alternativas estão sendo implementadas para avaliação.

REFERÊNCIAS

Bovet, D. P., Cesati, M. (2005) Understanding the Linux Kernel, 3rd ed., O'Reilly Media.

Broberg , J., Tari, Z., Zeephongsekul, P. (2006) "Task assignment with work-conserving migration", Parallel Computing archive, vol. 32, ed. 11-12, pp. 808-830.

Chen, X., Quan, Q., Jia, Y., Cai, K. (2006) "A Threshold Autoregressive Model for Software Aging", IEEE Int'l Symp. on Service-Oriented System Engineering, China.

Ferrari, D., Zhou, S. (1986) "A load index for dynamic load balancing", Proc. of 1986 ACM Fall joint computer conference, Dallas/TX, USA.

Ferrari, D., Zhou, S. (1987) "An empirical investigation of load indices for load balancing applications", In Proc. of Int'l Symp. on Computer Performance Modeling, Measurement and Evaluation, The Netherlands.

Gunther, N. J. (2004) "Linux Load Average Revealed", CMG 2004, Las Vegas, USA.

Love, R. (2003) Linux Kernel Development, 2nd ed., Sams.

Montgomery, D.C. (2005) Design and Analysis of Experiments, 6 ed., J. Wiley & Sons.

Payli, R. U., Yilmaz, E., Ecer, A., Akay, H. U., Chien, S. (2004) "DLB-A Dynamic Load Balancing Tool for Grid Computing", Proc. of Parallel CFD Conf., Can. Island.

Raj, A., "CPU hotplug Support in Linux Kernel". Disponível em: www.kernel.org/doc/Documentation/cpu-hotplug.txt. Acessado em 24/10/2010.

Saltzer, J. H., Gintell, J. W. (1970) "The Instrumentation of Multics", Communications of the ACM, vol. 13, nr. 8, USA.

Sit, H. Y. *et al*. (2004) "An Adaptive Clustering Approach to Dynamic Load Balancing", Int'l Symp. on Parallel Architectures, Algorithms and Networks, China.

Wilcox, M. (2003) "I'll Do It Later: Softirqs, Tasklets, Bottom Halves, Task Queues, Work Queues and Timers", Proc. of Linux Conference Australia, Perth.

Zhou, S. (1988) "A Trace-Driven Simulation Study of Dynamic Load Balancing", IEEE Transactions on Software Engineering, pp. 1327-1341.