Avaliação de Desempenho Sobre Agregações de *Links* em Modo *Round Robin*

Kassiano J. Matteussi[1], **Bruno G. Xavier**[1], **Adalto S. Sparremberger**[1]
César De Rose [1], **Tiago C. Ferreto**[1]

[1] Programa de Pós Graduação em Ciência da Computação - PPGCC
Faculdade de informática - FACIN
Pontifícia Universidade Católica do Rio Grande do Sul - PUCRS
Caixa Postal 1429 – 90619-900 – Porto Alegre – RS – Brazil

{kassiano.matteussi,bruno.xavier,adalto.sparremberger}@acad.pucrs.br

{cesar.derose,tiago.ferreto}@pucrs.br

Abstract. *This paper aim to evaluate the throughput losses over link aggregations in a round robin distribution way, by building a network emulation based on containers with CoreEmu framework. The Round Robin method intent to be the most flexible algorithm among the aggregation types, although issues related to network latencies may happen, caused by the TCP/IP congestion control. The work concludes, that kernel tuning is necessary according to the bandwidth and aggregation method used by collecting metrics whithin the emulated environment.*

Resumo. *O presente artigo contém uma avaliação quanto aos níveis de perda de vazão sobre a variação de interfaces agregadas em modo Round Robin, fazendo uso de uma simulação utilizando o framework CoreEmu. O algoritmo Round Robin apresenta o maior nível de aceitabilidade dentre os métodos de* teaming. *Porém, este tende a apresentar latências de tráfego, causadas pelo controle de congestionamento TCP/IP. O trabalho conclui que, ajustes de kernel são necessários de acordo com a agregação e largura de banda utilizada em um cenário real.*

1. Introdução

A busca por grande capacidade de largura de banda até pouco tempo atrás limitavam-se ao uso de internet discada e placas de rede 10/100 Mbit's. Em seguida os serviços de ADSL estavam presente com *links* dedicados, porém extremamente custosos e limitados. Dessa forma, a procura por soluções visando suprir tal demanda fez com que fabricantes de *switches*, em meados de 1990, incluíssem em seus dispositivos a capacidade de agregação. Entretanto, cada fabricante acabava por desenvolver sua própria tecnologia, gerando assim, problemas de compatibilidade. Observando essas dificuldades o grupo *IEEE 802.3ad* desenvolveu em 1997 o *Link Agreggation Control Protocol* (LACP) [Ko et al. 2011].

O LACP lida com dois problemas em conexões *Ethernet*: limitações com largura de banda e incapacidade de lidar com falhas. Atualmente, é muito comum que a capacidade dos *links* cresça em ordem de grandeza: 10 Mbit's, 100 Mbit's, 1 Gbit's, 10 Gbit's. Todavia, esta tecnologia demanda alto custo. Esse fator faz com que as empresas utilizem

a técnica [Deek et al. 2011] chamada *Channel Bonding*. Este método combina dois ou mais *links* físicos em um único *link* lógico de maior velocidade, provendo disponibilidade e redundância.

O objetivo central deste artigo é avaliar o desempenho de agregações de *link* utilizando o algoritmo *Round Robin* [Fürnkranz 2002]. Os testes foram executados em uma topologia virtualizada por meio de *containers* e análise sendo realizada sobre interfaces virtuais.

2. Agregação de *Links*

A agregação de *links* [Asante 2008] é um termo utilizado em redes de computadores para descrever métodos de combinação (agregações), em que múltiplas conexões de redes são agrupadas, a fim de prover balanceamento de carga e/ou tolerância a falhas. Estas agregações são também chamadas de *Bonding*, *NIC teaming* ou *Port Trunking*. Os agrupamentos são classificados segundo seu comportamento. Em [Davis et al. 2007] são enumerados 6 modos de agregação aplicados em sistemas operacionais e ativos de rede. Para a realização deste trabalho foi escolhido o modo *Round Robin*. Este modo apresenta a implementação mais simplificada entre os modos, onde a distribuição dos *frames* ocorre de maneira igual e sequencial para todas as interfaces escravas.

As interfaces são agrupadas em uma única entidade lógica, que por sua vez possui o endereçamento *MAC* e *IP* para a comunicação. Normalmente, o endereço *MAC* obtido corresponde à primeira interface da sequência agregada. O modo *Round Robin* apresenta o maior nível de compatibilidade dentre os métodos de *teaming*, visto que não depende de protocolos específicos em configurações de *switch* ou servidores. Todavia, por se tratar de um envio com pouco controle, onde os pacotes são apenas divididos e distribuídos nas interfaces, tende a apresentar perdas no recebimento de tráfego, causado pela retransmissão de pacotes por parte do controle de congestionamento *TCP/IP* [Wu et al. 2010]. Este comportamento ocorre quando os pacotes chegam fora de ordem no destino e são descartados pelo *kernel*. É devido a esse motivo que a agregação de interfaces neste cenário não fornece um crescimento totalmente linear. É importante ressaltar que o funcionamento da agregação *Round Robin* depende de um *port trunking* no lado do ativo que irá receber e endereçar o tráfego, por exemplo, um *Switch*.

3. Simulação

Com o objetivo de avaliar o percentual de perda de velocidade sobre a capacidade total das interfaces agregadas, este trabalho promoveu uma simulação utilizando a ferramenta CoreEmu[Ahrenholz et al. 2008] com a finalidade de reproduzir cenários de agregação em modo *Round Robin*. Para a configuração do ambiente, esforços sobre métodos de limitação de velocidade e construção de estruturas de agregação sobre *containers* foram necessários. Embora o experimento tenha sido executado em um ambiente virtual, a reprodução do cenário atendeu a todos os aspectos que fazem parte de uma agregação física em um ambiente real. Os testes foram aplicados sobre uma topologia simplificada entre dois *hosts* interligados por um *switch*, conforme apresentada na Figura 1. Com base nessa topologia foi desenvolvido um *script* para que, de forma dinâmica, alocasse a quantidade de interfaces necessárias para os testes e as unisse em duas estruturas de *bonding*.

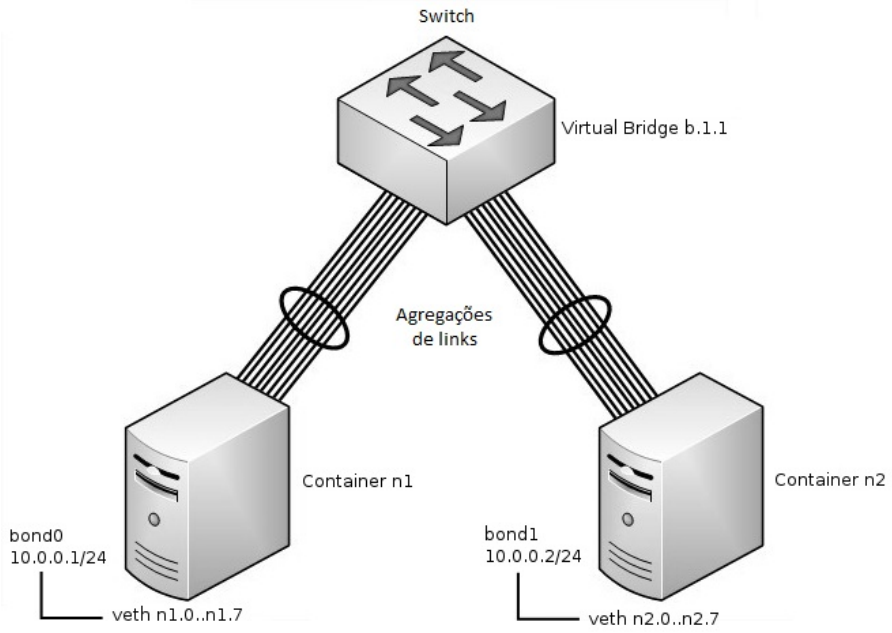

Figura 1. Topologia

3.1. Criação da interfaces virtuais

Canais *veth* foram configurados no *host* para cada interface criada. Interfaces *veth*, ou *virtual ethernet*, representam *fifos* inseridos no contexto dos *containers*, em que uma ponta é atrelada ao *host*, e outra ao nodo virtual. Desta forma, é construído um *streaming* de tráfego entre o *host* e cada *container*.

3.2. Construção dos *bondings*

Para o *trunking*, duas interfaces, *bond0* e *bond1* foram configuradas no *host*. O *driver* de *bonding* foi descrito para atender as duas interfaces em modo balance-rr (*round robin*) com os *aliases* correspondentes aos nomes definidos. Ambas foram alocadas nos *namespaces* dos *containers* junto às *interfaces* virtuais criadas anteriormente. Tendo em vista que uma agregação de *links* representa uma entidade lógica, não existe a necessidade de uma *ethernet* virtual para este recurso, apenas as interfaces virtuais devem estar pré configuradas junto ao *host*. Já dentro do nodo virtual, cada interface *trunk* recebeu um endereço IP para a comunicação entre os *containers*. Com as interfaces virtuais e os *bondings* configurados, o próximo passo foi a alocação dos *links* em suas respectivas agregações por meio da ferramenta *ifenslave*. Já com a intenção de simular um *switch*, cada interface foi inserida no contexto de um bridge (b.1.1). Deste modo, toda a comunicação entre *containers* utiliza o switch virtual configurado dentro do *host* como meio de transporte. Uma visão geral do *setup* pode ser verificada na Figura 1.

3.3. Simulação de velocidade

Tendo em vista que a simulação foi construída sobre *containers*, mecanismos de limitação de *link* foram aplicados sobre as interfaces virtuais. Para tanto, o subsistema *Traffic Control* do *Kernel* [Gerofi and Ishikawa 2012], por meio do binário **tc**, foi utilizado com o intuito de restringir a velocidade de cada link à 100 Mbits. Esta velocidade foi escolhida

com base nas limitações de *hardware* no ambiente de teste (1Gbit). Ainda, como controle de fluxo de *frames* é processado por *threads* do sistema, quanto maior a exigência de *throughput*, maior será o consumo de CPU e alocação de memória nos *buffers* de *socket*.

4. Resultados

O experimento foi reproduzido em computadores com arquiteturas distintas, onde os resultados inerentes as coletas foram confrontados para a avaliação final. As especificações dos ambientes são apresentadas na Tabela 1.

Tabela 1. Configuração das máquinas

	Máquina 1	Máquina 2
Processador	i5 1,76 GHz	i5 2.3 GHz
Memória	8	6
Memória da Máquina Virtual	6	4
Cores da Máquina Virtual	4	2

A coleta das métricas foi possível por meio da ferramenta *iperf*, respeitando o tamanho padrão de *MTU*(1500) [Inc. 2008], foram transmitidos utilizando o protocolo TCP durante 240 segundos. As médias foram capturadas para 8 agregações, incrementadas uma a uma, a fim de extrair uma linha base de desempenho. Os valores iniciais foram obtidos sem nenhuma mudança de configuração relativa ao sistema operacional. A Figura 2 apresenta o percentual de perda de *throughput* para a soma de banda, referente à agregação coletada. Para duas interfaces, por exemplo, considera-se um total de 200Mbits, 300Mbits e assim por diante.

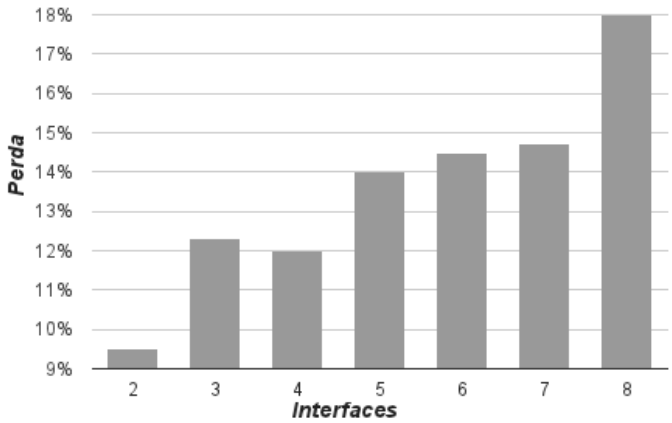

Figura 2. Perda de Desempenho por Agregações

Percebe-se repentina queda de desempenho à partir do incremento da quinta interface, o que corresponde à aproximadamente 400Mbits de *throughtput*. Isso ocorre em consequência dos contadores TCP [Ciliendo and Kunimasa 2007], onde são verificados gargalos relacionados à:

- *Socket Buffers* (rmem_size, wmen_size) - *Buffers* são utilizados pelo kernel para envio e recebimento de fluxo de dados. O ajuste destes parâmetros favorece o

desempenho em ambientes, onde o paralelismo é alto com um envio constante, evitando que o controle de fluxo atue sobre a chegada de tráfego e reduza o *throughput*. Os valores contidos nas variáveis *tcp* são valores referência: mínimo, inicial e final para as configurações *rmem e wmen*.

- *Transmit Queue Length* (txqueuelen) - Implementado via *hardware* ou *software*, este componente representa o tamanho da fila utilizada pela interface para o envio de pacotes, e pode virar um gargalo em ambientes onde o CPU é mais eficiente do que a velocidade de rede.

A Tabela 2 mostra os valores padrão e os ajustes feitos nos atributos mencionados.

Tabela 2. Ajustes de Parâmetros do Kernel

Propriedade	Valor Original	Valor Ajustado
net.core.rmem_max	8388608	16777216
net.core.wmem_max	8388608	16777216
net.ipv4.tcp_rmem	4096 87380 8388608	4096 87380 16777216
net.ipv4.tcp_wmem	4096 87380 8388608	4096 65535 16777216
ifconfig bond0 txqueuelen	1000 à 20000	2000
ifconfig bond1 txqueuelen	1000 à 20000	2000

Na Figura 3, é apresentada a comparação das métricas coletadas antes e após os ajustes.

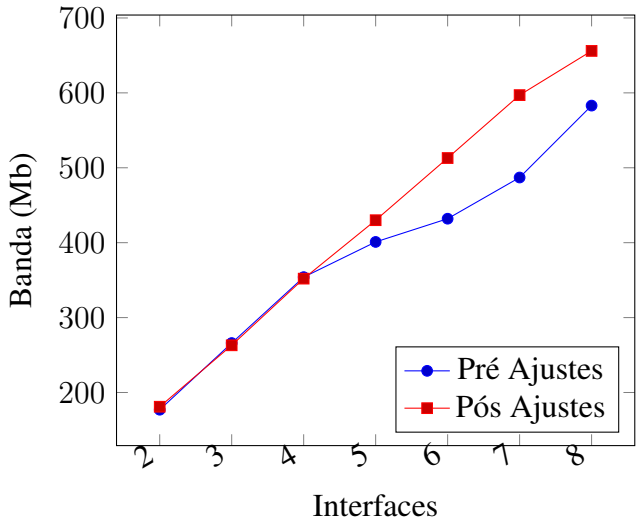

Figura 3. Gráfico de Desempenho

Após ajustes de *Kernel*, houve uma melhoria significativa no desempenho para 6, 7 e 8 *links*, com aumento aproximado de 25% sobre o *throughput* anterior.

5. Conclusão e Trabalhos Futuros

Esse trabalho apresentou uma análise de desempenho sobre agregações *round robin*, utilizando um método de simulação baseado em *containers*. Apesar das limitações de um ambiente virtual, foi possível reproduzir com relativa precisão um cenário real. Com base

nos resultados coletados, observou-se um desvio acentuado a partir de agregações entre mais de dois *links*. Para diminuir os gargalos encontrados foram feitos ajustes diretamente no *kernel* do sistema operacional. Tendo em vista que os experimentos utilizaram links limitados à 100Mbits, tais atributos de ajustes irão variar dependendo da largura de banda da agregação. Como futuros experimentos, sugere-se a avaliação de ambientes com interfaces de maior grandeza, como 1 e 10 Gbits, bem como testes envolvendo modos diferentes de *bonding*, tais como: *LACP, adaptive load balance e transmit load balance*.

Referências

Ahrenholz, J., Danilov, C., Henderson, T., and Kim, J. (2008). *CORE: A real-time network emulator*.

Asante (2008). *Link Aggregation and its Applications*. "Asante".

Ciliendo, E. and Kunimasa, T. (2007). *Linux Performance and Tuning Guidelines*. "IBM".

Davis, T., Tarreau, W., Gavrilov, C., Tindel, C. N., Girouard, J., and Vosburgh, J. (2007). Linux ethernet bonding driver howto. *available as download (bonding. txt) from the Linux Channel Bonding project Web site http://sourceforge. net/projects/bonding/(November 2007)*.

Deek, L., Garcia-Villegas, E., Belding, E., Lee, S.-J., and Almeroth, K. (2011). The impact of channel bonding on 802.11n network management. In *Proceedings of the Seventh COnference on Emerging Networking EXperiments and Technologies*, CoNEXT '11, pages 11:1–11:12, New York, NY, USA. ACM.

Fürnkranz, J. (2002). Round robin classification. *J. Mach. Learn. Res.*, 2:721–747.

Gerofi, B. and Ishikawa, Y. (2012). Enhancing tcp throughput of highly available virtual machines via speculative communication. *SIGPLAN Not.*, 47(7):87–96.

Inc., C. (2008). Solução de problemas de fragmentação de ip, mtu, mss e pmtud com gre e ipsec.

Ko, J., Eriksson, J., Tsiftes, N., Dawson-Haggerty, S., Vasseur, J.-P., Durvy, M., Terzis, A., Dunkels, A., and Culler, D. (2011). Industry: Beyond interoperability: Pushing the performance of sensor network ip stacks. In *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*, SenSys '11, pages 1–11, New York, NY, USA. ACM.

Wu, H., Feng, Z., Guo, C., and Zhang, Y. (2010). Ictcp: Incast congestion control for tcp in data center networks. In *Proceedings of the 6th International COnference*, CoNEXT '10, pages 13:1–13:12, New York, NY, USA. ACM.