Uma Avaliação do Desempenho do Sistema DSM Nautilus

Mario Donato Marino*e Geraldo Lino de Campos

Departamento de Engenharia de Computação - Escola Politécnica da Universidade de São Paulo
e-mail: {mario, geraldo}@regulus.pcs.usp.br

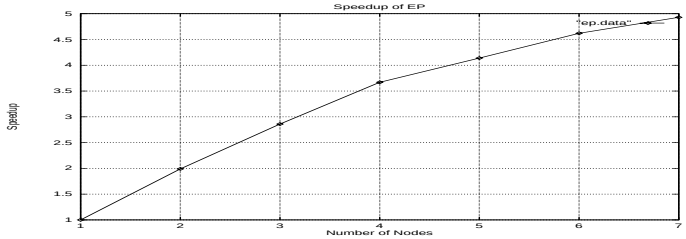
Através de sistemas DSM (Distributed Shared Memory), o uso de redes de computadores, como supercomputadores para a resolução de problemas científicos, tem sido bastante difundido. Os sistemas DSM permitem ao usuário enxergar uma rede de computadores como um sistema com memória compartilhada única e vários processadores[1]. Esta visão é vantajosa pois permite a utilização direta de programas desenvolvidos para máquinas paralelas sem a necessidade de portá-los para o modelo de passagem de mensagens, o que, dependendo do caso, pode ser muito custoso.

Os sistemas DSM mantêm a coerência através de dois possíveis esquemas: atualização e invalidação. As mensagens de atualização carregam as informações das diferenças sofridas pelas variáveis compartilhadas dentro de uma seção crítica. As mensagens de invalidação carregam informações de quais páginas devem se tornar inválidas (não mais coerentes). Os sistemas DSM podem replicar ou migrar dados. O princípio da localidade, aplicado através da replicação ou migração pode ser aplicado, dependendo do aplicativo que está sendo executado.

Para minimizar o número de mensagens, principal fator determinante do desempenho, que trafegam na rede os sistemas DSM mais recentes como o ThreadMarks[2], o CVM[3] e o JIAJIA[4], utilizam modelos de consistência fracos que mantêm a sincronização somente nos pontos de sincronização. Dentre os modelos de consistência fracos, pode-se exemplificar os seguintes: de liberação, de liberação preguiçosa, de entrada, de escopo.

O sistema DSM *Nautilus*, seguindo a corrente internacional: *(i)*replica dados para obter maior localidade; porém com a replicação desperdiça-se muita memória compartilhada; *(ii)*utiliza *threads* para minimizar o chaveamento de contexto; *(iii)*utiliza a consistência de liberação; *(iv)*para manter a consistência de liberação utiliza os mecanismos de atualização e invalidação; *(v)*distingue entre mensagens de sincronização e consistência; *(vi)*utiliza protocolos TCP/IP e UDP (em desenvolvimento); *(vii)* não utiliza *broadcast* nem *multicast* para a implementação de sincronização.

Para averiguar o desempenho do Nautilus, foi executado o programa EP (NAS *benchmark*)em até 7 nós. O desempenho do EP está mostrado na figura abaixo.



Observando a figura acima, percebe-se que com o aumento do número de nós, o desempenho do EP aumenta com o aumento do número de nós até o momento que ocorre a saturação devido ao maior número de nós existentes, que acabam saturando a rede.

References

[1] Stum M., Zhou S. , *Algorithms Implementing Distributed Shared Memory*, University of Toronto, IEEE Computer v.23, n.5, pp.54-64, May 1990.

[2] Keleher P., *Lazy Release Consistency for Distributed Shared Memory*, PHD Thesis¸University of Rice, Texas, Houston, January, 1995.

[3] Keleher P., *The Relative Importance of Concurrent Writers and Weak COnsistency Models*, in Proceedings of 16th International Conference on Distributed Computing Systems (ICDCS-16), pp. 91-98, May 1996.

[4] Hu W., Shi W., Tang Z., *JIAJIA: An SVM System Based on a new Cache Coherence Protocol*, technical report no.980001, Center of High Performance Computing, Institute of Computing Technology, Chinese Academy of Sciences, January, 1998.

*Suporte financeiro da agência CNPq-Brasil, processo no. 142753/97-1