Orientação ao recurso: um modelo de comunicação para a computação em clusters

Albano Alves
ESTiG-IPB
Bragança, Portugal
albano@ipb.pt

António Pina
DI-UMinho
Braga, Portugal
pina@di.uminho.pt

José Exposto
ESTiG-IPB
Bragança, Portugal
exp@ipb.pt

José Rufino
ESTiG-IPB
Bragança, Portugal
rufino@ipb.pt

Resumo

A orientação ao recurso é um novo modelo de comunicação e de computação capaz de tirar partido da comunicação de elevado desempenho, no desenvolvimento de uma classe importante de aplicações paralelas/distribuídas de grande escala. Neste contexto, a biblioteca RoCL, especialmente desenhada para a execução em ambientes cluster, de máquinas SMP, usando múltiplos fios-de-execução, permite a exploração de múltiplas tecnologias de comunicação de elevado desempenho e múltiplos protocolos de comunicação, tais como a Myrinet (GM) e a Gigabit Ethernet (VIA). Os recursos são as abstracções usadas para modelar as entidades aplicacionais que podem ser registadas e localizadas através de um serviço básico de directório, distribuído pelos nodos do cluster.

Palavras chave*: computação em clusters, passagem de mensagens, redes SAN, serviço de directório, localização de recursos, múltiplos fios-de-execução.*

1. Introdução

A passagem de mensagens é uma abordagem amplamente utilizada na programação de aplicações paralelas/distribuídas, tendo em vista a exploração de ambientes multi-máquina e/ou multi-processador. No contexto dos sistemas *cluster*, as bibliotecas e plataformas de programação tradicionais que oferecem passagem de mensagens têm vindo a ser melhoradas, no sentido de contemplar o suporte a novas tecnologias de comunicação/interligação, tais como a Myrinet e a Gigabit Ethernet. Tais tecnologias, vulgarmente designadas por redes SAN, permitem alcançar débitos da ordem dos *gigabits* e garantem tempos de resposta inferiores a uma ou duas dezenas de micro-segundos.

Comunicação inter-máquina [9][5][11]

A exploração das tecnologias actualmente utilizadas na interligação das máquinas de um *cluster* passa, normalmente, pela utilização de bibliotecas de comunicação de nível-utilizador. As primitivas disponibilizadas por estas bibliotecas permitem a emissão e a recepção de mensagens sem recurso a cópias de memória e a minimização dos tempos de comutação de contexto, dado não ser necessária a intervenção do sistema operativo. São bons exemplos desta abordagem, a biblioteca GM e a especificação VIA, através da implementação MVIA, usadas para explorar, respectivamente, as tecnologias Myrinet e Gigabit Ethernet.
Apesar do desempenho notável que estas bibliotecas demonstraram ser possível alcançar, através de testes inter-máquina, a utilização eficiente de tecnologias SAN em aplicações paralelas/distribuídas de grau de complexidade elevado não logrou ainda alcançar o nível desejado. Na verdade, a interface oferecida é de baixo-nível (muito próxima das funcionalidades disponíveis no próprio *hardware* de comunicação) enquanto as aplicações, devido ao seu crescente grau de complexidade e especialização, são desenhadas e desenvolvidas recorrendo, cada vez mais, a abstracções de muito alto-nível.

Ambientes de execução [10]

A computação dita de elevado desempenho tem estado associada, por norma, ao cálculo científico e a aplicações específicas da engenharia. No entanto, com o crescimento da Internet e com o crescente nível de requisitos dos sistemas de informação, sistemas baseados em *clusters* têm vindo a ser propostos no desenho de serviços onde são exigidos elevados níveis de disponibilidade e de capacidade de resposta em situações de pico.
O projecto SIRe[1], no qual estamos envolvidos, é o caso particular de uma aplicação complexa que visa o desenvolvimento de um ambiente escalável para recuperação de informação, incluindo subsistemas de colecta, indexação e pesquisa de informação.
Naquele contexto, os ambientes de programação paralela tradicionais, cujo objectivo último é rentabilizar cada um dos processadores disponíveis num *cluster*, monopolizando de forma estática todo o sistema no sentido de acelerar a execução de um programa em particular, não parecem ser os mais adequados. Do nosso ponto de vista, é de todo conveniente dispor de um ambiente de execução onde

[1]Projecto de investigação suportado pela FCT/MCT, Portugal, contracto POSI/CHS/41739/2001, com o nome "SIRe – Scalable Information Retrieval environment".

múltiplos programas, eventualmente de múltiplos utilizadores, cooperam de maneira a alcançar um objectivo comum. Assim, deverá ser tomado em consideração o elevado número de operações de entrada/saída necessárias, sem entrar em linha de conta com o normal suporte à comunicação inter-máquina, e o facto de os requisitos do sistema poderem vir a variar ao longo do tempo, o que pode ter como resultado a execução esporádica de outros módulos aplicacionais e a reconfiguração automática dos recursos envolvidos.

2. Comunicação orientada ao recurso [12]

A nossa abordagem explora o paradigma CoR – Computação orientada ao Recurso – como forma de tornar efectivos, em ambientes *cluster*: *1)* o processo de desenho de plataformas de computação paralelas e distribuídas apropriadas à implementação e execução de aplicações complexas; *2)* o uso eficiente das tecnologias de comunicação de elevado desempenho usadas em redes SAN. Foi neste contexto que desenvolvemos o pCoR, um protótipo para a programação de alto-nível com múltiplos fios de execução, descrito numa publicação anterior, agora enriquecido com uma biblioteca de comunicação de nível intermédio denominada RoCL (*Resource oriented Communication Library*), ambos suportados pelo paradigma da orientação ao recurso.

Conceitos gerais

Em CoR, os recursos são as abstracções de alto-nível usadas para modelar entidades aplicacionais.

Os recursos possuem propriedades descritas através da enumeração dos respectivos atributos. Os atributos são pares $\langle nome, valor \rangle$ em que *nome* é uma *string* e *valor* é uma sequência qualquer de *bytes*.

À biblioteca cabe o papel de definir a interface para a criação e definição das propriedades dos recursos e de estabelecer os mecanismos necessários para o respectivo registo e localização, no contexto global do *cluster*.

Os atributos dos recursos são armazenados num serviço de directório global de baixo-nível – parte integrante do sistema RoCL – no momento do seu registo. A operação de registo envolve ainda a atribuição, ao recurso, de um identificador global, independentemente da máquina a partir de onde o registo do recurso é efectuado.

A comunicação entre recursos faz-se a partir do conhecimento dos respectivos identificadores globais, que podem ser conhecidos à partida ou determinados a partir dos seus atributos, recorrendo a operações de localização no directório. As mensagens usadas na comunicação são mantidas em tampões próprios, especializados, por forma a evitar cópias de memória, quer no envio, quer na recepção.

A interacção com as tecnologias de comunicação disponíveis no *cluster* é efectuada através dos subsistemas de comunicação, baseados em bibliotecas de baixo-nível.

Funcionamento básico

Uma aplicação RoCL começa por instanciar um contexto, usando a primitiva `rocl_init()`, com um porto de comunicação por cada tecnologia de base (GM, VIA, etc.) em utilização. Quaisquer mensagens destinadas a um determinado recurso serão, do ponto de vista do subsistema de comunicação, endereçadas ao contexto a partir de onde o recurso foi criado/registado.

Um determinado recurso é registado, num contexto previamente instanciado, através da primitiva `int rocl_register(rocl_attrl_t *attrs)`, a qual permite especificar um conjunto de atributos e devolve um identificador global.

A localização de recursos e a correspondente recuperação de atributos é efectuada através da primitiva `rocl_query(int id, rocl_attrl_t *attrs)`. Note-se que, numa dada pesquisa, ou é fornecido o identificador global do recurso (parâmetro `id`) ou são fornecidos alguns atributos conhecidos desse recurso (parâmetro `attrs`).

A lista de atributos usada como parâmetro na primitiva `rocl_query` poderá incluir atributos para selecção do recurso – atributos com o nome e o valor especificados – e atributos para recuperação de informação – atributos com o nome especificado e o valor desconhecido.

Qualquer recurso pode ser eliminado através da primitiva `rocl_delete(int id)`.

Passagem de mensagens: O envio e a recepção de mensagens fazem-se através da primitiva `rocl_send(int oid, int did, int tag, void *ptr, int len)` e da primitiva `rocl_recv(int did, int oid, int tag, void **ptr, int *aoid, int *atag, int *alen, int timeout)`. No envio de uma mensagem é indicado o recurso na origem da mensagem, para além do destinatário, enquanto que, na recepção, para além do recurso na origem, é indicado o recurso onde irá ser recebida a mensagem.

A troca de mensagens com *zero cópias* de memória é conseguida através das primitivas `void * rocl_bfget(int len)`, `rocl_bfret(void *ptr)`, `rocl_bftoret(void *ptr)` e `rocl_bfstat(void *ptr)`, as quais se destinam, respectivamente, a requerer um tampão de memória para posterior envio, a devolver um tampão à biblioteca RoCL, a indicar que um dado tampão deve ser automaticamente devolvido à biblioteca RoCL após a conclusão do envio associado e, finalmente, a averiguar o estado da operação de envio associada a um determinado tampão.

3. Serviço de directório [2][7]

Em contraste com as tradicionais abordagens à comunicação por passagem de mensagens, em que máquinas, processadores e processos têm um papel pre-

ponderante no desenho de aplicações, no RocL, o recurso assume-se como a entidade básica de modelação.

Em conformidade, a arquitectura adoptada para o desenvolvimento da biblioteca usa um servidor de directório por máquina do *cluster*, para armazenar os atributos dos recursos criados/registados localmente.

Cada servidor de directório obtém no momento do arranque um intervalo de identificadores globais únicos, por forma a permitir a atribuição de identificadores a recursos, sem negociação com outros servidores. Isto significa que as operações de registo são tratadas sem comunicação inter-máquina.

O serviço de directório do RoCL foi desenhado de forma a potenciar o registo e a localização de recursos genéricos, com tempos de resposta mínimos, por forma a suportar o nível de abstracção desejado sem comprometer o desempenho do sistema.

Localização de Recursos

A localização/pesquisa de recursos é realizada em dois tempos. Em primeiro lugar é pesquisada a base de dados do servidor de directório local à maquina onde é invocada a primitiva `rocl_query` – pesquisa local. Em segundo lugar, e caso a resposta não possa ser fornecida com base na informação armazenada localmente, são contactados os outros servidores de directório dispersos pelo *cluster* – pesquisa global.

Nas pesquisas globais (ver figura 1), a interrogação de servidores de directório remotos é efectuada através de um mecanismo de difusão: o pedido de localização/pesquisa recebido pelo servidor local é difundido pelos restantes servidores e apenas aqueles que possuirem informação relativa ao recurso alvo procederão à devolução de uma resposta.

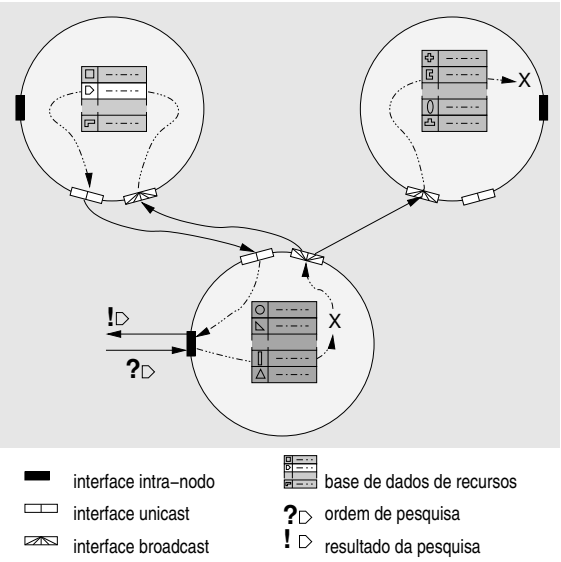

Figura 1. Mecanismo de pesquisa global.

A difusão de pedidos faz-se de duas formas: através de *broadcast UDP*, usando a tecnologia Ethernet, ou através de *spanning trees*, usando a tecnologia Myrinet ou o protocolo VIA.

Relativamente à primeira opção, é de referir que, na generalidade dos *clusters*, é usada a tecnologia FastEhternet como meio de interligação secundário, de baixo custo, essencialmente para suporte às operações de instalação, manutenção e gestão. A utilização deste meio de interligação para o funcionamento do serviço de directório permite, por um lado, explorar capacidades nativas de difusão e, por outro, libertar os meios de interligação primários (Myrinet, por exemplo) para as funções directamente relacionadas com a passagem de mensagens.

Quanto à segunda opção, é necessário ter em consideração que, em certas aplicações, a utilização do serviço de directório poderá ser mais intensa, exigindo uma capacidade de comunicação superior à conseguida com *broadcast UDP*. Dado que, nem o hardware Myrinet suporta difusão, nem o protocolo VIA explora essa funcionalidade em relação à tecnologia Gigabit Ethernet, o RoCL utiliza *spanning trees*.

Por forma a manter actualizada, em cada servidor de directório, a lista de todos os servidores presentes no *cluster*, os servidores anunciam-se, periodicamente, através de *broadcast UDP*.

Devido ao método flexível de localização/pesquisa de recursos que o RoCL oferece, uma única invocação da primitiva `rocl_query` poderá devolver múltiplos resultados. De facto, os critérios de selecção/pesquisa indicados na primitiva poderão ter como objectivo a obtenção de todos os recursos que cumprem um determinado requisito. Por isso, o serviço de directório implementa também um protocolo de compilação/fusão de respostas devolvidas por um ou mais servidores.

4. Modelo de Comunicação [4][3][1][8][6]

A interface oferecida pelo RoCL uniformiza a exploração de múltiplos protocolos de baixo-nível, nomeadamente GM e VIA. Apesar das diferenças entre estes dois protocolos (VIA é orientado à conexão e GM não), a arquitectura da biblioteca RoCL permite tirar o máximo partido de ambas as tecnologias.

Na verdade, o RoCL foi desenhado para possibilitar a inclusão de outros subsistemas de comunicação. O próprio UDP, por exemplo, foi entretanto facilmente incluído.

Animação de recursos

A biblioteca RoCL combina múltiplas tecnologias de comunicação para garantir um melhor desempenho e permite a operação em *clusters* com nodos não homogéneos. No envio de uma única mensagem, são gerados vários fragmentos, os quais são enviados usando múltiplas interfaces de comunicação – exploração das múltiplas interfaces, da mesma tecnologia ou não, presentes numa máquina.

Quando os nodos de um *cluster* não partilham todos pelo menos uma determinada tecnologia de comunicação, são usados os nodos multi-conectados como encaminhadores de mensagens – conectividade entre nodos com tecnologias distintas.

Uma vez que as entidades comunicantes suportadas pelo RoCL não são processos nem processadores, torna-se necessário ter em consideração a forma como são animadas essas entidades – os recursos. O sistema de comunicação do RoCL assume que os recursos são animados por fios-de-execução POSIX, obrigando a cuidados específicos devido ao paralelismo associado, principalmente no caso de máquinas SMP.

Tal opção impõe a utilização de mecanismos para multiplexagem de canais de comunicação que, no nosso caso, se baseiam na definição de filas de envio e recepção e na utilização de um ou mais fios-de-execução para sondar o meio de comunicação (ou tratar as interrupções associadas) e manipular as filas.

A fila de envio é utilizada quando existe contenção no acesso ao meio de comunicação, enquanto que a fila de recepção possibilita a entrega de mensagens destinadas a recursos que ainda não se encontram na fase de recepção.

Estes mecanismos são também usados para o suporte à comunicação assíncrona oferecida pelo RoCL.

Gestão de tampões

A utilização de protocolos/bibliotecas de comunicação de baixo-nível que garantam uma eficiente exploração das tecnologias de interligação existentes nos *cluster* actuais (redes SAN) é um dos objectivos da biblioteca de nível-intermédio que nos propusemos desenvolver. Estas bibliotecas permitem contornar as pesadas comutações para o contexto do sistema operativo e oferecem a possibilidade de envio e recepção de mensagens sem cópias intermédias de memória.

A primeira vantagem é facilmente transposta para as camadas de nível superior (bibliotecas de nível-intermédio ou aplicações que usem directamente os mecanismos de comunicação baixo-nível). No entanto, para integrar as potencialidades da comunicação *zero cópias*, as camadas de nível superior têm que definir uma interface apropriada, compatível com as primitivas de baixo-nível.

De acordo com esta abordagem, o RoCL recorre ao conceito de tampão específico para o armazenamento de uma mensagem e integra um sistema de gestão de tampões. Assim, o modelo de comunicação oferecido impõe a obtenção de tampões apropriados antes que qualquer envio possa ter lugar e utiliza tampões pré-alocados para alojar as mensagens recebidas. O sistema de gestão de tampões visa, essencialmente, minimizar os tempos associados à criação de tampões e ao respectivo registo, uma vez que a comunicação sem cópias obriga ao pré-registo de zonas de memória.

Operações de escrita remota

Em geral, a passagem de mensagens envolve dois intervenientes: o emissor e o receptor. Ao receptor cabe a tarefa de, por cada envio, desencadear a execução de uma operação para consumo da mensagem. Em abordagens do género da adoptada nas mensagens activas, não há a execução explícita de uma operação de recepção, no destino, mas uma entidade de sistema encarrega-se do processamento associado à entrega da mensagem.

Actualmente, algumas tecnologias de comunicação possibilitam operações de leitura e escrita remota – operações de comunicação em que o emissor é o único interveniente – vulgarmente denominadas de operações RDMA (*Remote* DMA).

No caso do RoCL, a tradicional passagem de mensagens obriga, no destino, à sinalização do fio-de-execução de sistema o qual, por sua vez, irá proceder à sinalização do fio-de-execução que anima o recurso de destino da mensagem (ver figura 2).

Com a integração de operações RDMA, o sistema RoCL passa a dispor de primitivas de comunicação com tempos de resposta muito próximos do *hardware* de interligação.

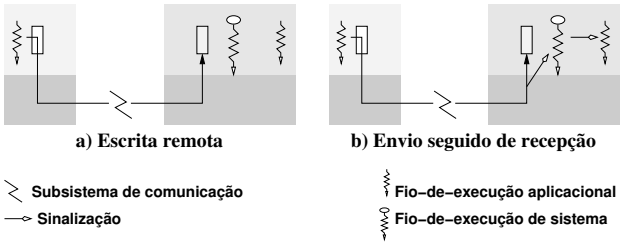

Figura 2. Escrita remota *vs* envio/recepção.

As operações RDMA no RoCL usam os conceitos recurso e tampão de memória, sem introduzir novas abstracções. As primitivas usadas para o efeito são: `rocl_bfshare(int ogid, void *ptr)` e `rocl_put(int ogid, int dgid, void *ptr, int loffset, int roffset, int length)`.

A primeira primitiva permite associar a um recurso um determinado tampão, tornando-o acessível remotamente, isto é, permitindo a escrita remota. A segunda possibilita a movimentação de um fragmento de um dado tampão local para uma determinada zona do tampão remoto associado ao recurso indicado como destino.

A nossa abordagem à escrita remota tem um nível de abstracção superior ao oferecido pelos subsistemas de comunicação GM e VIA. Com efeito, os modelos de comunicação daqueles subsistemas impõe o uso das convencionais primitivas de envio e recepção de mensagens para obter os endereços de memória necessários nas operações RDMA. No nosso modelo de comunicação, o directório e a informação de controlo trocada entre contextos, de forma transparente às aplicações, permite esconder a complexidade associada à operação de escrita remota.

A operação de leitura remota (`rocl_get`) não foi, ainda, considerada, por não existir actualmente suporte nativo, ao nível das bibliotecas GM e MVIA.

5. Discussão

No contexto da computação e comunicação em sistemas baseados em *clusters* a nossa contribuição reside, essencialmente, na exploração do conceito de recurso e no papel atribuído aos fios-de-execução.

Fios-de-execução

No RoCL, tanto o mecanismo de interacção com os subsistemas de comunicação, como a animação dos recursos se baseiam em fios-de-execução POSIX, oferecidos pelo sistema operativo.
Os fios-de-execução são usados, de uma forma natural, para explorar o paralelismo de aplicações, particularmente no caso de máquinas SMP, muito utilizadas, hoje em dia, na construção de *clusters*.
Habitualmente, os sistemas de passagem de mensagens convencionais não integram fios-de-execução ou então incluem sistemas de fios-de-execução proprietários, com funcionalidades limitadas, por forma a evitar tempos de comutação e sincronização, supostamente muito elevados e, portanto, não compatíveis com os tempos de resposta do *hardware* de comunicações.
No contexto da investigação que prosseguimos, orientada à utilização de sistemas não proprietários, o aparecimento da biblioteca NPTL (actualmente já incluída no Linux RedHat 9.0), veio mostrar que a utilização dos fios-de-execução não compromete, necessariamente, o desempenho do sistema de passagem de mensagens.

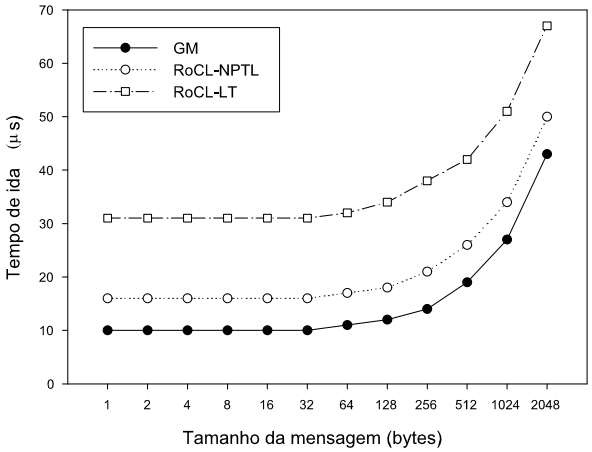

Figura 3. Desempenho das primitvas RocL.

A figura 3 compara os tempos de resposta associados à troca de mensagens de variados tamanhos obtidos para a tecnologia Myrinet. A curva *GM* mostra o desempenho que é possível alcançar com a biblioteca de baixo-nível GM ocupando um processador, a tempo inteiro, para efectuar a sondagem do meio de comunicação. As curvas *RoCL-LT* e *RoCL-NPTL* mostram o desempenho do sistema RoCL com a utilização, respectivamente, da clássica biblioteca LinuxThreads e da nova biblioteca NPTL.
É de notar o aumento de desempenho que se deve, essencialmente, à diminuição dos tempos de comutação e de sincronização entre fios-de-execução. Note-se que, o adicional de 5μs presente na curva *RoCL-NPTL* é o preço a pagar pela flexibilidade oferecida pelo RoCL, isto é, a orientação ao recurso e o modelo de comunicação totalmente assíncrono.

Recursos

Habitualmente, os sistemas de passagem de mensagens oferecem aos programadores modelos de comunicação onde as entidades envolvidas correspondem às máquinas ou processadores disponíveis num *cluster*. Em alguns casos são oferecidos portos de comunicação como abstracções para multiplexagem do meio de comunicação, ao nível do processo ou da máquina. Não existem, no entanto, mecanismos adequados e suficientemente flexíveis, ao dispor dos programadores, para a associação entre entidades computacionais e entidades comunicantes. A maior dificuldade surge no momento da distribuição ou redistribuição das entidades computacionais pelos nodos de um *cluster*.
Por outro lado, a existência de plataformas de programação paralela/distribuída de mais alto nível, que incluem sistemas sofisticados para interacção com o *hardware* disponível, não vem facilitar a tarefa do programador no desenho de soluções que sigam novos paradigmas. Com efeito, estas plataformas encerram, habitualmente, modelos de programação próprios e a implementação de componentes que não sigam esses modelos acabam por resultar numa deficiente exploração do *hardware* disponível.
O RoCL, a meio caminho entre as duas abordagens acima, ao oferecer o recurso como entidade básica para a modelação de aplicações, contorna a complexidade e limitações dos subsistemas de comunicação de baixo-nível, sem impor, ao programador, um modelo de programação rígido. A abstracção recurso garante um nível de abstração suficientemente elevado para o desenvolvimento conveniente de aplicações e plataformas de algum nível de complexidade sem que fique comprometida a exploração eficiente dos sistemas físicos e lógicos subjacentes.

Referências

[1] A. Alves, A. Pina, J. Exposto e J. Rufino. ToCL: a thread oriented communication library to interface VIA and GM. a ser apresentado em ICCS '03, 2003.

[2] M. Beck, J. J. Dongarra, G. E. Fagg, G. A. Geist, P. Gray, J. Kohl, M. Migliardi, K. Moore, T. Moore, P. Papadopoulous, S. L. Scott e V. Sunderam. HARNESS: A next generation distributed virtual machine. *Future Generation Computer Systems*, 15(5–6):571–582, 1999.

[3] R. Bhoedjang, T. Rühl, R. Hofman, K. Langendoen e H. Bal. Panda: A Portable Platform to Support Par. Programming Languages. Em *SEDMS IV*, 1993.

[4] L. Bougé, J.-F. Méhaut e R. Namyst. Madeleine: An Efficient and Portable Communication Interface for RPC-Based Multithreaded Environments. Em *PACT '98*, 1998.

[5] Compaq Computer Corp., Intel Corporation & Microsoft Corporation. Virtual Interface Architecture Specification, 1997.

[6] R. Espenica e P. Medeiros. Porting PVM to the VIA architecture using a fast communication library. Em *PVM/MPI '02*, 2002.

[7] S. Fitzgerald, I. Foster, C. Kesselman, G. von Laszewski, W. Smith e S. Tuecke. A Directory Service for Configuring High-Performance Distributed Computations. Em *HPDC '97*, 1997.

[8] J.-S. Kim, K. Kim e S.-I. Jung. SOVIA: A User-level Sockets Layer Over Virtual Interface Architecture. Em *CLUSTER '01*, 2001.

[9] Myricom. *The GM Message Passing System*, 2000.

[10] R. Namyst e J. Méhaut. PM^2: Parallel Multithreaded Machine. A computing environment for distributed architectures. Em *ParCo'95*, 1995.

[11] National Energy Research Scientific Computing Center. M-VIA: A High Performance Modular VIA for Linux. http://www.nersc.gov/research/FTG/via/index, 2002.

[12] A. Pina, V. Oliveira, C. Moreira e A. Alves. pCoR - a Prototype for Resource Oriented Computing. Em *HPC '02*, 2002.