RPM: Um Sistema Para Posicionar Réplicas

André Zampieri[1,2], Taisy Silva Weber[1], Marcia Pasin[1]
[1]Programa de Pós-Graduação em Computação, UFRGS
[2]Universidade de Caxias do Sul
{azampier@ucs.tche.br; taisy@inf.ufrgs.br,pasin@inf.ufrgs.br}

RESUMO

Replicação de objetos é usada para garantir uma maior disponibilidade de recursos em um sistema distribuído. Porém, surgem problemas como o controle da consistência, o número de réplicas necessárias e onde estas réplicas devem estar posicionadas. A consistência é garantida por um protocolo de consistência de réplicas. O problema da determinação do número de réplicas e onde as mesmas devem estar posicionadas é resolvido, atualmente, de forma empírica pelos projetistas ou administradores dos sistemas. Neste artigo são abordados aspectos a serem considerados nas decisões de posicionamento automático de réplicas e é proposto um módulo gerente, o RPM, *Replica Placement Manager*, o qual determina onde as réplicas devem ser posicionadas, através de informações obtidas pelo monitoramento dinâmico do sistema.

ABSTRACT

Distributed systems use object replication to assure availability. However, new problems arise as how to control the replica consistency, how to determine the needed number of replicas and where the replicas should be placed. A replica consistency protocol guarantees the replica consistency. The replication level and replica placement, nowadays, are an empirical designer decision. Here we show aspects that should be considered in replica placement. We also introduce a manager module, the RPM – *Replica Placement Manager* – that decides where these replicas should be placed, using data obtained from the dynamic monitoring of the system.

1. Introdução

Em sistemas distribuídos, a replicação de objetos pode aumentar a disponibilidade dos recursos do sistema além de, sob certas condições, contribuir para aumentar o desempenho. Porém, com a introdução da replicação novos problemas surgem, como a necessidade de um protocolo para o controle de consistência das réplicas. Exemplos desses protocolos são o protocolo de primário-backup [BUD93] e o protocolo de réplicas ativas [SCH93], responsáveis por manter as cópias em um estado mutuamente consistente. Adicionalmente, o projetista ou o administrador deve lidar com as seguintes questões: Qual é o nível de replicação necessário (número de cópias que devem ser criadas) e onde estas réplicas devem estar posicionadas no sistema?

Estas questões não são simples de serem respondidas. Na inicialização, o administrador poderia determinar o número de réplicas e o posicionamento destas. Esta abordagem exige o conhecimento das características dos componentes do sistema (nodos, links de comunicação, interdependência dos objetos, dependência de falhas, ...) sendo portanto inadequada. O administrador pode não ter a base técnica necessária para determinar o nível de replicação, uma vez que a disponibilidade não varia proporcionalmente ao número de réplicas (aumentar o número de réplicas não garante sempre uma maior disponibilidade). Além disso, para o posicionamento, devem ser considerados a confiabilidade, o desempenho e as interdependências de falhas nos nodos nos quais as réplicas devem ser posicionadas. A

medida que o sistema progride ocorrem mudanças na topologia da rede ou na carga do sistema. Isto exige um monitoramento constante pelo administrador, a fim de que, quando estas situações surgirem, este tome providências para garantir o progresso do sistema.

O ganho na disponibilidade, confiabilidade e desempenho, que são atingidos pela replicação de objetos, é uma função complexa, dependente de vários fatores, tais como: número e posicionamento das réplicas; natureza das transações realizadas sobre os objetos; a carga média dos nodos onde as réplicas estão posicionadas; os protocolos de consistência e recuperação para manter as réplicas; as interdependências dos objetos e as características de disponibilidade e desempenho das máquinas e da rede que compõem o sistema.

Esses fatores justificam a necessidade de um mecanismo que, através do monitoramento constante do sistema, consiga determinar qual é o melhor nodo para receber uma réplica, tirando essa responsabilidade do projetista ou administrador do sistema.

2. Posicionamento de Réplicas no RMS

Quanto maior o número de réplicas, mais tempo será gasto para manter a consistência, comprometendo o desempenho do sistema. Portanto, não adianta criar réplicas indiscriminadamente e jogá-las aleatoriamente nos nodos do sistema. Réplicas mal posicionadas podem não trazer benefícios para a disponibilidade e nem para o desempenho.

Entretanto, muito pouca pesquisa tem sido realizada sobre esse tema. Alguns sistemas, como o Hector [RUS98], preocupam-se com a migração de tarefas, mas não lidam com replicação. O MARS [KOP90] posiciona objetos em nodos que possuem confiabilidade consistentes com o objetivo da aplicação, porém opera com posicionamento estático. O RMS [LIT94] – *Replica Management System* - trata detalhadamente o problema de posicionamento, permitindo ao programador especificar a qualidade do serviço requerida em termos de disponibilidade e desempenho. Ele trabalha com níveis de replicação e posicionamento em tempo de execução, considerando as falhas dos nodos e as dependências de falhas entre os nodos quando as réplicas são posicionadas.

2.1 Aspectos Relevantes Para o Posicionamento de Réplicas

Os aspectos descritos a seguir são considerados relevantes para o posicionamento de réplicas pelo sistema RMS. Estes aspectos podem servir de base à construção de outras estratégias de posicionamento, como a que propomos.

Confiabilidade dos Componentes

O termo componente refere-se aos componentes de hardware e software do sistema distribuído, como por exemplo nodos e links de comunicação. É necessário considerar as interações entre os componentes do sistema distribuído para determinar o grau de confiabilidade de um componente específico.

Por exemplo, um nodo que raramente falha pode estar conectado a uma rede que perde mensagens freqüentemente, passando a visão de não confiável ao usuário.

Confiabilidade e Disponibilidade do Nodo

As duas medidas mais comuns são: MTTF (tempo médio para falhas) e MTTR (tempo médio para recuperação). O ideal é que os nodos possuam um alto MTTF - falhem raramente - e um baixo MTTR - recuperem-se rápido. Para se obter estas medidas é necessário monitorar continuamente os nodos do sistema por um longo período de tempo.

Confiabilidade dos Links de Comunicação

Para monitorar a confiabilidade dos *links* deve ser determinada a probabilidade de perda de mensagens e de particionamento da rede. A perda de mensagens ocorre por problemas no meio de comunicação e, mais freqüentemente, pelo *overflow* do *buffer* do nodo receptor.

Dependência de Falhas

O objetivo dessa indicação é avaliar como uma falha pode levar a outras, para isso deve-se levar em consideração as dependências entre os componentes. Por exemplo: fontes de energia compartilhadas, sub-rede comum, dependência do mesmo disco da rede. As interdependências são propriedades estáticas da topologia da rede ou de nodos individuais, mas fatores dinâmicos podem também afetar a confiabilidade.

Interdependência dos Objetos

Sempre que um objeto invoca um método de outro objeto ele é dependente daquele objeto. O grau de dependência do objeto inclui a freqüência de invocação dos métodos e as propriedades dos objetos sendo invocados. As dependências entre objetos mudam mais rapidamente que as interdependências entre os nodos e podem variar de uma execução da aplicação para outra. Esta qualidade dinâmica implica que esse fator deve ser continuamente monitorado. O desempenho e a disponibilidade serão aumentados se posicionarmos juntos os objetos que são dependentes. Se os objetos podem migrar de um nodo para outro, então os dados associados pela dependência podem indicar outros objetos que também devem ser migrados. Esse fator pode indicar também que a migração não é possível (o objeto a ser migrado pode ser criticamente dependente de outro objeto que deve ser fixo).

Desempenho dos Nodos

O desempenho de um nodo é uma função complexa da carga do nodo, isto é, o número de processos ativos sendo executados nele e da sua configuração, como por exemplo, a velocidade do processador e da memória disponível. Deve-se considerar o sistema distribuído como um todo, ao invés de um único nodo, para poder proporcionar um melhor desempenho e disponibilidade. Por exemplo, posicionar um objeto em um nodo levemente carregado, porém conectado com o resto da rede por um link de comunicação lento, pode, se o objeto necessitar de muitas interações remotas, reduzir o desempenho do objeto .

Neste contexto as dependências entre os objetos também devem ser consideradas. Por exemplo, se um objeto faz uso excessivo dos recursos de outros objetos que devem residir em um nodo pesado, pode fazer mais sentido posicionar o objeto naquele nodo, desde que o *overhead* gerado com esta realocação seja inferior ao *overhead* proporcionado pela comunicação remota.

Os padrões de utilização de recursos dos objetos a serem posicionados e os recursos providos pelos nodos nos quais eles podem ser posicionados necessitam ser levados em conta. É necessário um mapeamento dinâmico da utilização dos recursos pelos objetos e a correspondente disponibilidade de recursos nos nodos que compõem o sistema distribuído.

Desempenho do Link de Comunicação

O desempenho do *link* é expresso em termos de quão rápido ele entrega uma mensagem e a sua largura de banda. O meio de comunicação deve ser levado em consideração de acordo com o tipo de aplicação que será executada. Por exemplo, para aplicações multimídia ou de tempo real, as tecnologias de FDDI e ATM oferecem vantagens significativas.

Para obter eficiência e tolerância a falhas, o RMS está disponível em vários nodos. Para os nodos onde o RMS reside adquirirem os valores descritos acima, é assumida uma distribuição "preguiçosa" [LIT94], isto é, a cada 24 hs, cada nodo do sistema divulga por broadcast, para cada um dos outros nodos do sistema, os valores dos seus atributos, tais como valores de confiabilidade, carga média, etc. Dessa maneira cada nodo possui as informações de todos os outros nodos do sistema.

2.2 Estratégia de posicionamento no RMS

O RMS [LIT94] é composto por vários módulos que interagem entre si, cada um com funções específicas. Estes módulos obtém, do sistema distribuído, os valores relacionados aos aspectos descritos acima. O módulo específico que tem por função determinar o nível de replicação e o posicionamento é chamado de PPM – *Placement Policy Module* (figura 1).

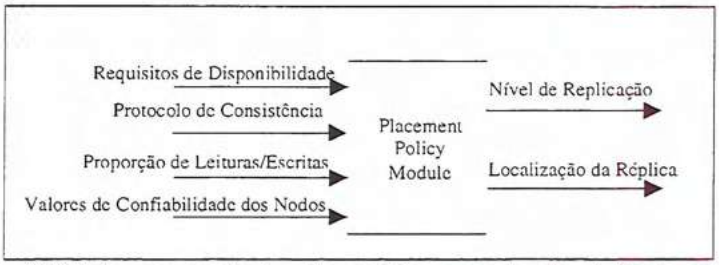

Fig. 1 –Módulo responsável pelo posicionamento de réplicas no RMS.

O PPM determina o nível de replicação e o posicionamento das réplicas através das seguintes informações obtidas do sistema:

- Requisitos de disponibilidade: são determinadas pelo projetista. Por exemplo, "deve falhar menos do que uma vez por mês".
- Valores de confiabilidade dos nodos: são determinados pelo MTTF e MTTR dos nodos, obtidos através de monitoramento constante dos componentes do sistema. Essas informações provém de um módulo chamado *Monitor Daemon*.
- Protocolo de consistência usado: alguns protocolos se comportam de uma forma para operações de leitura e de forma diferente para escrita. Assim, é necessário levar em conta o número de leituras e escritas do sistema.
- Razão (proporção) de leituras e escritas: em sistemas cuja característica é possuir mais leituras do que escritas, o aumento do nível de replicação pode gerar um aumento do desempenho do sistema. Em outros, cuja característica seja, ao contrário, possuir mais escritas do que leituras, o aumento do número de réplicas fará com que o protocolo tenha de "trabalhar mais", aumentando o *overhead* do sistema e degradando o desempenho.

Com estas informações, o PPM toma decisões sobre o nível de replicação desejado e sobre o melhor posicionamento de uma réplica. Ele gera uma lista com os nodos disponíveis do sistema, que irão atingir o nível desejado de disponibilidade e desempenho.

3. RPM – Replica Placement Manager

O sistema RPM, atualmente em fase inicial de desenvolvimento, se propõe a resolver o problema do posicionamento de réplicas considerando a carga atual, a carga média e a confiabilidade dos nodos. O nível de replicação não é tratado no RPM; este fator é determinado por outro módulo do sistema (como no Ape [PAS00] atualmente sendo desenvolvido na UFRGS) ou até mesmo pela aplicação.

O sistema conta com agentes coletores de informações em cada nodo do sistema, que podem se comunicar com o nodo que possui o RPM através de um *middleware* de comunicação de grupos, como por exemplo o sistema Horus [BIR96] ou o sistema Ensemble [HAY98]. Um sistema de comunicação de grupos provê um suporte adequado para troca confiável de mensagens entre os nodos e suporte aos protocolos de consistência de réplicas, garantindo atomicidade e ordenação de mensagens mesmo na ocorrência de falhas de nodos e links.

A grande diferença entre o sistema RMS e o sistema RPM é que este considera como fator primordial para determinar o melhor nodo para receber uma réplica a carga atual dos nodos. Ele obtém estas informações de forma dinâmica, enquanto que o sistema descrito anteriormente obtém estas informações em longos intervalos de tempo. Além disso o uso de agentes coletores de informações nos nodos e um *middleware* de comunicação de grupos para facilitar a troca de mensagens entre os nodos e o sistema, o diferenciam do sistema anterior.

Como o RPM é dinâmico, interessa a ele valores dinâmicos e simples, de fácil aquisição. Dessa forma, valores de significado após um longo período de observação (como o MTTF e MTTR) e estáticos (como configuração) são acrescidos de métricas obtidas pela verificação de nodos falhos e pela carga momentânea e história de carga dos nodos.

O RPM, possui um módulo responsável pela determinação da localização de uma réplica (fig. 2). Este, quando solicitado localizar uma réplica, deve inquirir os nodos do sistema (comunicando-se com os agentes de monitoramento de carga nos nodos) para saber quais as suas respectivas cargas de processamento.

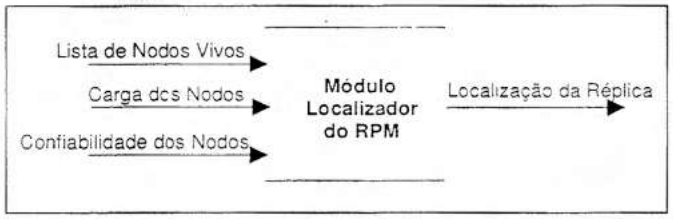

Fig. 2 – Módulo de posicionamento de réplicas do RPM.

A informação de carga é obtida pelos agentes em cada nodo a partir de variáveis de estado do *middleware* (componentes de grupos ativos no nodo, quantidade de mensagens esperando para ser entregues, tempo médio de mensagem na fila...), ou, se não disponíveis, do próprio sistema operacional (número de processos na fila de espera e na fila de prontos e número de arquivos abertos, por exemplo). Com base nestas informações, e também com valores estáticos de confiabilidade e de configuração dos nodos, ele determina, dentre a lista de nodos vivos, qual é o melhor para receber a réplica (a lista de nodos vivos é mantida por agentes detectores de falhas).

Essa busca de informações pode ser lenta se a rede for muito grande. Para solucionar este problema pode-se pensar em heurísticas baseadas em nodos candidatos. O RPM mantém uma lista de nodos candidatos a receber uma réplica baseado nas cargas e posições anteriores, e também nas remoções e migrações promovidas pelo sistema. Só os nodos candidatos são verificados a cada rodada. A informação sobre o melhor candidato no momento, segundo a heurística usada, é retornada para o solicitador da informação, que se encarregará de criar a réplica neste nodo.

O progresso do sistema não pode ser interrompido por falhas no RPM, portanto este deve possuir características de tolerância a falhas. Pode-se criá-lo como um objeto que é replicado em vários nodos para garantir a disponibilidade utilizando-se um dos protocolos mencionados anteriormente para garantir a consistência das réplicas. Se o protocolo utilizado for o de réplicas ativas, todas as réplicas recebem a requisição do serviço, computam o melhor

posicionamento e devem chegar ao mesmo resultado (dessa maneira, a estratégia de posicionamento deve ser determinística). Se o protocolo utilizado for o de primário-backup, apenas a réplica primária do RPM calcula o posicionamento. As demais réplicas mantém o estado do primário e podem substituí-lo em caso de falhas. Como o RPM é um objeto do sistema, ele próprio pode estar sujeito a remoção, migração, criação de novas réplicas e reposicionamento.

4. Conclusão

A disponibilidade e o desempenho de um objeto replicado são significativamente afetados pelo posicionamento, número de réplicas e protocolo de consistência. As decisões sobre o posicionamento das réplicas devem se basear nas características do sistema que está sendo desenvolvido. Isso exige o conhecimento de vários aspectos diferentes, muitos deles mudando continuamente durante o progresso do sistema.

A implementação de um módulo gerente que monitora constantemente a disponibilidade das máquinas e as cargas dos processadores, a medida em que o sistema progride, surge como uma solução para o problema do posicionamento das réplicas.

Neste artigo foi discutido o sistema RMS e os aspectos por ele considerados para determinar o posicionamento de uma réplica. Ele se preocupa com valores de confiabilidade dos nodos, resultados de um monitoramento constante dos nodos, porém não se preocupa com a carga dinâmica dos nodos do sistema. É proposta a criação do RPM (inspirado no RMS), módulo capaz de determinar o posicionamento de uma réplica através da obtenção de valores simples e dinâmicos, de aquisição mais fácil do que os valores considerados no RMS. O RPM está em fase inicial de implementação e deverá ser incorporado ao Ape, sistema de geração, migração e controle de réplicas atualmente em desenvolvimento na UFRGS.

5. Referências Bibliográficas

[BIR96] Birman, Kenneth P.; et alii. **HORUS: A Flexible Group Communication System**. Communications of the ACM. Vol. 39, n° 4, pág. 76-83, April, 1996.

[BUD93] Budhiraja, Navin; et alii. **The Primary-Backup Approach** em Mullender, Sape. Distributed Systems. 2ª ed. , New York: ACM Press, 1993. Pg. 199-215

[GUE97] Guerraoui, R.; Schiper, André **Software-Based Replication for Fault Tolerance**. IEEE Computer, April, 1997, pág. 68-74

[HAY98] Hayden, Mark G. **The Ensemble System**. Cornell University, January, 1998

[LIT94] Little, M.C.; McCue, D.L. **The Replica Management System: a Scheme for Flexible and Dynamic Replication**, Proc. Of the 2nd Workshop on Configurable Distributed Systems, University of Newcastle upon Tyne, March, 1994

[PAS00] Pasin, Marcia, et alii. **Reconfiguring Replicated Objects to Provide Fault Tolerance and Load Balancing in a Distributed System**. Universidade Federal do Rio Grande do Sul, Maio, 2000.

[RUS98] Russ, Samuel H.; et alii. **The Hector Distributed Run-Time Environment**. IEEE Transactions on Parallel and Distributed Systems, Vol. 9, n° 11, Nov., 1998

[SCH93] Schneider, Fred B. **Replication Management Using the State-Machine Approach** em Mullender, Sape. Distributed Systems. 2ª ed. , New York: ACM Press, 1993. Pg. 169-197

[KOP90] Kopetz, H. et alii. **Tolerant Transient Faults in MARS**. Proceedings of the 20th International Symposium on Fault-Tolerant Computing, 1990, pp. 466-473.