Uma métrica para recomendação de federação em rede ponto-a-ponto

Wilton Oliveira Ferreira[1,2,3], Ricardo Batista Rodrigues[2], Rodrigo E. Assad[3], Rafael R. de Souza[2], Legilmo M. F. de Oliveira[1], Josino R. Neto[2,3], Julio C. Damasceno[2]

[1]C.E.S.A.R – Centro de Estudos Avançados de Recife
Rua Bione, 220 – Cais do Apolo – Bairro do Recife, Recife – PE, Brasil

[2]Centro de Informática – Universidade Federal de Pernambuco (UFPE)
Av. Jornalista Anibal Fernandes, s/n, Cidade Universitária, Recife – PE, Brasil

[3]Ustore
Rua do Apolo, 161, Cais do Apolo, Bairro do Recife, Recife – PE, Brazil

{wof,lmfo}@cesar.org.br,{wof,rrs,vcg,jrn,jcd}@cin.ufpe.br

Abstract. *This paper proposes a metric for recommendation engine for creating federations of peers in P2P, systems using as criteria the similarity on the processing capacity of the peers.*

Resumo. *Este artigo propõe uma métrica para mecanismo de recomendação para a criação de federações de pares em sistemas P2P, usando como critério a semelhança na capacidade de processamento dos pares.*

1. Introdução

Redes ponto-a-ponto também conhecidas como P2P (*peer-to-peer*) são redes onde seus participantes são igualmente privilegiados. As ferramentas que utiliza P2P tem suas tarefas ou serviços divididos entre os participantes da rede. Os assuntos sobre P2P sempre foram discutidos nos meios científicos desde o surgimento da *Internet*. Mas só no início da década de 1990 foi que o P2P tornou-se um estilo arquitetônico [Clements et al. 2010]. Algumas ferramentas como o Napster [Parker et al. 2004] [Jiaqing et al. 2006] [Parker et al. 2004] são ferramentais que usam P2P para troca de arquivos entre os participantes e consequentemente popularizou ainda mais este estilo arquitetônico.

Inicialmente as ferramentas desenvolvidas utilizando P2P apresentavam bons desempenho, baixo custo de implementação e velocidade na implantação atendendo as necessidades da comunidade. Porém, com o sucesso destas ferramentas, muitas pessoas começaram a instalar em seus computadores e isto ampliou de forma descontrolada a rede P2P por trás de tais ferramentas.

Com o crescimento descontrolado a qualidade dos serviços começaram a diminuir, a velocidade para encontrar serviços nas redes que antes era um atrativo começou a ser um fator preocupante. Surgiu então necessidade de organizar os *peers* de forma a permitir escalabilidade, velocidade e garantia mínima de disponibilidade. Estas necessidades levaram ao desenvolvimento de pesquisas e experimentos, que observou a necessidade de mecanismos para organizar os *peers* automaticamente [Oliveira et al. 2005] [Ranjan et al. 2008].

A disposição dos *peers* de forma organizada possibilista o direcionamento de requisições de serviços de forma eficiente e consequentemente melhoria no desempenho, e também permite garantia mínima de disponibilidade. Alguns estudos propõem soluções que permitem organizar recursos computacionais em forma de federações [Duarte et al. 2010] [Mancini et al. 2009]. Este artigo apresenta um mecanismo para recomendação de *peer* na criação de federações. Além desta seção introdutória.

2. GroupUsto.re

O GroupUsto.re é uma ferramenta para recomendação na criação de federações de *peers* em sistemas P2P. Esta abordagem realiza agrupamento de *peers* que apresenta o máximo de similaridade entre si. A similaridade entre os *peers*, auxilia para que não haja gargalo nem na troca de mensagens e nem nos processamentos de dados. Este fator auxilia também para que os *peers* de uma determinada federação tenham a mesma capacidade de atender as demandas impostas por outra federação.

Desta forma, todos os *peer* serão submetidos à mesma carga de trabalho, e naturalmente terão tempo de resposta semelhante evitando espera. Além dos fatores relacionados ao desempenho, também são acrescentadas as capacidades de gestão otimizada de um nu´ mero menor de *peer*, e possibilidade de direcionar fluxos de armazenamento de dados para federações especificas.

As recomendações são feitas a partir de análise no perfil e avaliação da capacidade de cada *peer*. A métrica utilizada para determinar o perfil e a capacidade individual do *peer* utiliza valores obtidos através de agentes. Cada agente presente na arquitetura ilustrada na Figura 1, atua na coleta de diferentes informações. Todas as informações coletadas pelos agentes são processadas e para cada grupo de informação é atribuído uma determinada função. Como por exemplo: todos os *peers* que estiver em um mesmo segui- mento de rede farão parte de uma mesma federação, exceto quando houver limites de *peers* por federação. E para estes casos, após formadas todas federações possíveis, serão indicadas federações com o resto dos *peers* que sobrarem de um mesmo seguimento de rede juntamente com outros que estiverem mais próximos possível. Esta proximidade será determinada pelo agente *HopAgent* apresentado na Seção 2.1.1.

2.1. Controlador de Agentes

As federações para recomendação são criadas por meio da união de *peers* em um único grupo. Os critérios para esta união é a similaridade entre os *peers*, que são mensuradas através de informações coletadas a partir de agentes. O módulo controlador de agentes tem como objetivo gerenciar os agentes descrito na Seção 2.1.1.

As funcionalidades executadas por este módulo são: enviar parâmetros de execução para os agentes; verificar se todos os agentes estão em execução; receber os dados enviados por cada um dos agentes; invocar o serviço de padronização de dados para padronizar as informações coletadas pelos agentes e entregar os dados já padronizados para o gerenciador de federação.

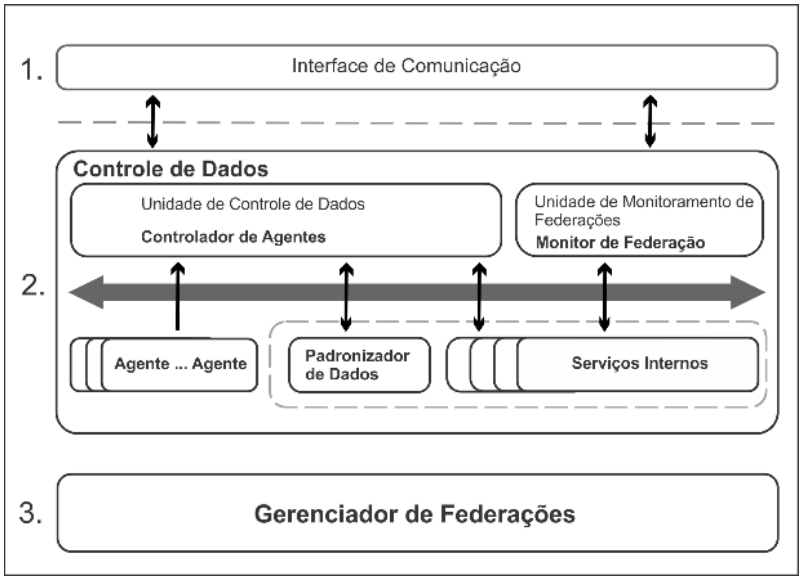

Figure 1. Arquitetura GroupUsto.re

2.1.1. Agentes

No contexto deste trabalho, os agentes são rotinas de software autônomas que tem finalidades especificas. Como já mencionado, os agentes são responsáveis por coletas as informações em cada *peer*, e transmitir tais informações para o controlador de agentes. A seguir são apresentados os agentes que compõe esta ferramenta:

- ***LatenceAgent*** – tem como objetivo verificar a velocidade que o *peer* transferi dados pela rede;
- ***MemoryAgent*** – tem como objetivo verificar a capacidade de memória RAM (*Random Access Memory*) disponibilizada para uso da aplicação P2P;
- ***ProcessorAgent*** – tem como objetivo extrair as informações referentes ao número de núcleo de processador disponibilizado para o uso da aplicação P2P;
- ***StorageAgent*** – tem por objetivo coletar informações referente a capacidade de armazenamento de dados do *peer*, este agente também observa a quantidade de dados já armazenado;
- ***HopAgent*** – tem o objetivo de medir a quantidade de saltos por roteadores que uma mensagem deve percorrer para sair de um *peer* e alcançar os outros *peers*;
- ***AvailabilityAgent*** – tem o objetivo de mensurar quanto tempo o *peer* tem disponível para o sistema;
- ***ReliabilityAgent*** – tem como objetivo medir o quanto o peer é confiável com relação a todas as características e serviços por ele disponibilizados;

Todo o processo de coleta das informações por parte dos agentes pode ser parametrizado, isto permite que ao processar tais informações possam ser aplicadas ponderações. As ponderações dos valores coletados pelos agentes são realizadas através da configuração individual. Este procedimento não é padrão da ferramenta, porém, permite que valores como os coletados pelos agentes *AvailabilityAgent* e *ReliabilityAgent*, tenham maior expressividade no momento das recomendações.

3. Avaliação dos Resultados

Os resultados obtidos nestes experimentos são ilustrados na Figura 2. Como pode ser observado, houve redução da dissimilaridade em todos os casos do experimento. Em alguns casos esta redução de dissimilaridade foi maior, já em outros casos a ela foi pouco notável.

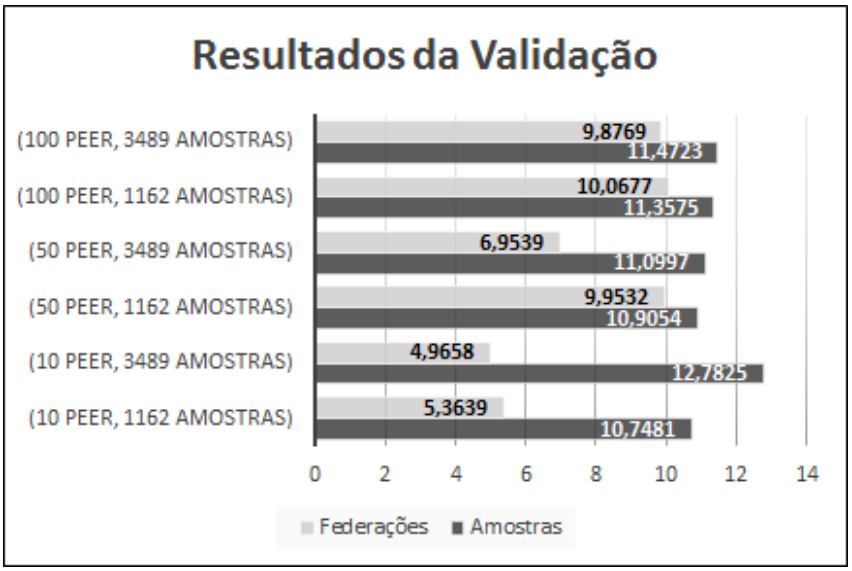

Figure 2. Resultados e Validações

4. Conclusão

Como foi observado na Figura 2, os experimentos apresentaram-se satisfatórios. Com isso conclui-se que o GroupUsto.re sugere organização de *peers* em federações de forma eficiente. E ainda observando o gráfico é possível notar que quanto maior o número de *peers* conectados, mais eficiente torna-se o GroupUsto.re.

Desta forma conclui-se que esta ferramenta atende satisfatoriamente demandas de escalabilidades do sistema.

Referências

Clements, P., Bachmann, F., Bass, L., Garlan, D., Ivers, J., Little, R., Merson, P., Nord, R., and Stafford, J. (2010). Documenting Software Architectures: Views and Beyond (2nd Edition). Addison-Wesley Professional, 2 edition.

Duarte, M., Assad, R., Ferraz, F., Ferreira, L., and de Lemos Meira, S. (2010). An availability algorithm for backup systems using secure p2p platform. In Software Engineer- ing Advances (ICSEA), 2010 Fifth International Conference on, pages 477 – 481.

Jiaqing, L., Shijie, Z., Chunjiang, W., Yiyi, D., and Xiaoqian, Y. (2006). Adaptive flooding routing algorithm in unstructured p2p. In Communications, Circuits and Systems Proceedings, 2006 International Conference on, volume 3, pages 1557 –1561.

Mancini, E., Rak, M., and Villano, U. (2009). Perfcloud: Grid services for performance-oriented development of cloud computing applications. In Enabling Technologies:

Infrastructures for Collaborative Enterprises, 2009. WETICE '09. 18*th* IEEE International Workshops on, pages 201 –206.

Oliveira, M., Garcia, I., and Nunes, A. (2005). Resource discovery em uma arquitetura p2p aplicado à distribuição e compartilhamento de componentes de software. In I Workshop de Redes Peer-to-Peer - WP2P 2005, pages 73–84.

Parker, D., Collins, S., and Cleary, D. (2004). Building near real-time p-2-p applications with jxta. In Cluster Computing and the Grid, 2004. CCGrid 2004. IEEE International Symposium on, pages 338 – 345.

Ranjan, R., Harwood, A., and Buyya, R. (2008). A case for cooperative and incentive-based federation of distributed clusters. Future Generation Computer Systems,24(4):280 – 295.