Construção de aplicação: Sistema de arquivos distribuídos utilizando DHT.

Jovani de Souza[1], **Jeferson Augusto Schein**[1]

[1]Ciência Da Computação – Universidade Federal da Fronteira Sul (UFFS)
Chapecó – SC – Brazil

jovanidesouza@hotmail.com, schein.jefer@gmail.com

Abstract. *This paper aims to report the experience of developing an application, used to simulate a distributed file system. To compose the application, it is initially exposed the theoretical reference necessary to clarify the concepts used in the development, and then will describe the implementation processes used. The purpose of this report and application is to finalize the studies on the distributed computing primitives that were studied. In addition, it aims to describe the theoretical and practical stages that involve the project.*

Resumo. *Este artigo objetiva relatar a experiência de desenvolvimento de uma aplicação, utilizada para simular um sistema de arquivos distribuído. Para compor a aplicação, inicialmente é exposto o referencial teórico necessário para esclarecer os conceitos utilizados no seu desenvolvimento e, em seguida será descrito os processos de implementação utilizados. Objetiva-se com este relatório e aplicação, finalizar os estudos sobre as primitivas da computação distribuída que foram estudadas. Ademais, almeja-se descrever as etapas teóricas e práticas que envolvem o projeto.*

1. Introdução

Este relatório objetiva descrever os processos de desenvolvimento de uma aplicação que simula um sistema de arquivos distribuídos. Além isso, este *Software* juntamente com o relatório, também servirá como instrumento avaliativo prático e teórico do componente curricular de 'Computação Distribuída', abordando os principais tópicos relacionados às primitivas da computação distribuída.

A computação distribuída é uma das áreas da computação mais importantes e mais abrangentes estudadas atualmente. Um sistema de arquivos distribuído é importante para que aplicações distribuídas possam compartilhar informações de forma rápida e escalável. Além de evitar a replicações de dados, um sistema distribuído é sempre considerado pela sua tolerância à falhas, visto que ele não tem um único ponto de falha como um sistema de arquivos centralizado.

Para desenvolver esta aplicação, foram estudados diversos conceitos e algoritmos de computação distribuída. Para integrar a aplicação, foram utilizadas as primitivas de comunicação, acordos entre nodos, sincronização e tabela *hash* distribuída, usada para distribuição de carga de dados entre os nodos do sistema.

Conforme a quantidade de dados e arquivos propriamente ditos, vem crescendo, se faz necessário uma forma de gerenciar dinamicamente e de forma eficiente esses dados separados dentro da rede. As soluções estão evoluindo de forma rápida, tanto em qualidade quanto em quantidade de sistemas distribuídos.

Considerando esses motivos a decisão de se fazer um trabalho com uma base de dados abstratos, apenas com identificador foi o escolhido para o desenvolvimento. Podendo mesmo com algo simples implementar processos e métodos para melhorar a distribuição de dados dentro da rede. Com nodos compartilhando memoria onde o controle é descentralizado, ou seja, cada nodo pode fazer todas as funções possíveis e, descentralizando o controle da rede faz com que se tenha uma maior nível de confiabilidade, pois como cada nodo pode fazer o controle dos dados, o sistema consegue assim se adaptara rede.

2. Referencial teórico

Sistemas computacionais centralizados são utilizados em grande quantidade e com muita frequência desde a década de 1970, porém são sistemas que são drasticamente afetados pelo aumento de requisições de usuários. Diante disso, a ideia de sistemas distribuídos se desenvolveu e é de extrema importância para a Ciência da computação como é conhecida atualmente.

2.1. Sistemas distribuídos

Diante da necessidade em evoluir os sistemas computacionais, até então precários, surgem na década de 1980 grandes avanços em sistemas computacionais distribuídos, ou apenas, sistemas distribuídos. A ideia básica de um sistema distribuído é a de um conjunto de nodos (computadores) que são independentes entre si e compartilham recursos para oferecer um determinado serviço que se apresenta para o usuário como centralizado [Tanenbaum and Van Steen 2007].

O objetivo de um sistema distribuído é o de escalar requisições de usuários (que podem ser pessoas ou programas), de forma que todos sejam atendidos sem a centralização de recursos, como processamento e armazenamento. Modelos Cliente-Servidor são inapropriados para diversos cenários computacionais, pois acabam perdendo desempenho quando o número de clientes aumenta. A solução de um sistema distribuído é considerar cada nodo independente e autossuficiente, além disso expande suas capacidades de armazenamento e processamento ao compartilhar recursos com outros nodos.

Outro adicional de auto valor para sistemas distribuídos é a tolerância à falhas, em um sistema centralizado existe também um único ponto de falha, que pode comprometer totalmente uma aplicação. Já em um sistema distribuído, o sistema continua operando mesmo que um ou mais nodos sejam prejudicados, é possível ainda que novos nodos sejam utilizados para substituir nodos defeituosos e mantendo assim o funcionamento do sistema sem interoperabilidade.

No entanto o uso de sistemas distribuídos precisa seguir modelos próprios de desenvolvimento e tecnologias que permitam alta escalabilidade e baixa latência para os nodos e, também que permitam que o sistema seja seguro e transparente para todos os seus usuários. Dentre os conceitos mais importantes para sistemas distribuídos estão a comunicação orientada a mensagem e à fluxo, identificadores de endereços, sincronização com exclusão mútua e relógios lógicos, acordos e eleição de líder, além de tolerância a falhas, consistência e replicação.

2.2. Comunicação

A comunicação é umas das principais primitivas em um sistema distribuído, ela é essencial para que os nodos possam compartilhar recursos e também possam obter informações sobre disponibilidade e *status* de outros nodos. Ela pode ser desenvolvida de diversas formas e aplicada em diversos protocolos de rede. A comunicação pode ser orientada a fluxo, utilizada por *streaming* de vídeo por exemplo, e comumente orientada a mensagens, onde a troca de mensagens entre os nodos é que determina o funcionamento do sistema como um todo [Tanenbaum and Van Steen 2007].

Em um sistema distribuído a comunicação é também chamada de comunicação entre processos, pois é possível que processos em um mesmo nodo precisem realizar troca de mensagens entre si, além de se comunicar com outros processos em outros nodos. A comunicação tem alta relevância em sistemas distribuídos devido a ausência de uma memória compartilhada nesse tipo de sistema, portanto toda ação ou requisição feita em memória local em um sistema centralizado, é feito por meio de troca de mensagens entre processos distribuídos.

2.3. Acordos e eleição de líder

A eleição de líder faz parte da primitiva de acordos em sistemas distribuídos, um acordo pode ser visto como um objetivo da comunicação, pode ser usado como forma de realizar uma tarefa em comum e também como acordo de liderança. Um acordo de liderança ou eleição é o procedimento adotado pelos nodos de um sistema distribuído em um determinado momento, para escolher um dos nodos para realizar uma tarefa que possua restrições. Um exemplo de uso de eleição é onde é preciso escolher um nodo para funcionar como um provedor de informação centralizado temporariamente.

Um acordo, é considerado muito útil em diversas aplicações distribuídas e é propagado pela primitiva da comunicação. Um acordo também é usado para determinar um padrão a ser seguido pelos nodos, como modelo de sincronização ou padrão de dados utilizados. Também é possível utilizar acordos para realizar procedimentos de segurança como autenticação e validação de informações.

2.4. Tabela Hash distribuída (DHT)

Uma tabela *Hash* é uma estrutura de dados que utiliza associação de pares chave-valor, para mapear informações. Também é conhecida como tabela de dispersão, que tem como objetivo facilitar a busca de um dado, reduzindo o custo computacional da operação de busca. Usando como parâmetro, alguns valores presentes no arquivo é possível criar um código único e, aplicando alguns cálculos matemáticos esse código é conhecido como *hash*. Esse código *hash* consiste em um id único para cada registro ou arquivo, assim como também pode ser usado para agrupar registros.

A tabela *hash* distribuída funciona da mesma forma, porém é acessada e modificada por vários nodos distribuídos. Outra característica importante é que a tabela *hash* distribuída não é alocada de forma centralizada, ou seja, cada nodo é responsável por alocar uma parte da tabela e, assim utilizar da comunicação caso precise acessar um registro que esteja em outra parte da tabela.

O uso de tabela *hash* distribuída em sistemas distribuídos é de grande utilidade, pois o desempenho ganho com o uso da estrutura de *hash*, faz com o atraso de resposta

inevitável nesse tipo de sistema seja diminuído. Uma tabela instanciada por dois nodos por exemplo, dividindo igualmente a alocação de memória entre eles, ao acessar um dado, o nodo só vai utilizar da comunicação com o outro nodo caso precise um dado que não esteja em sua *hash* local, diminuindo assim o tempo de resposta da consulta e ainda mantendo as características de sistema distribuído.

3. Especificação do problema distribuído

No problema explorado em questão, a principal característica que ser leva em consideração, é que o sistema necessita ser distribuído e dinâmico. Não necessariamente os *nodo* que compartilham espaços de memoria necessariamente serão instanciados em ordem, assim como também o possível tratamento de eventuais problemas de conexão entre os nodos.

3.1. Sistema de arquivos distribuídos

Um sistema de arquivos é uma abstração que permite alteração, remoção, inserção e consulta de informações em arquivos de sistema computacional. Um sistema de arquivos distribuídos possui o objetivo de compartilhar recursos de vários computadores para ampliar a capacidade de armazenamento e diminuir a latência das operações entre vários nodos, além de incluir a tolerância a falhas nessas operações.

Diante disso, a proposta do projeto prático é desenvolver uma aplicação que simule um sistema de arquivos distribuídos, que seja escalável e que permita todas as operações sobre arquivos que foram citadas. Para isso, as primitivas utilizadas são, primeiramente a comunicação e também acordos, que serão utilizados para a construção da base da aplicação. Além disso, o compartilhamento de informações do sistema será feito utilizando o conceito de tabela *hash* distribuída, visando facilitar o acesso e a persistência das informações pelos diversos nodos.

4. Implementação

Como já apresentado anteriormente, existem varias maneiras de se implementar um sistema de arquivos distribuído. Foi seguindo um *Back-bone* com os conceitos básicos, e algumas alternativas entre as primitivas escolhidas para a implementação que a implementação ocorreu. A implementação do sistema foi realizada na linguagem *Python 3* e com isso foi possível utilizar facilmente alguns módulos de sistema, sincronização e comunicação, como *time*, *os* e *sockets*.

Cada nodo possui um identificador (id), informado quando o nodo é executado. Com base no id é criada uma porta de rede especifica, no caso dessa implementação a porta sempre será, 5000 + id, considerando o valor máximo do id sendo 10, para testes, porém para aumentar o valor, se faz apenas a modificação da quantidade na função *alive()*, sendo assim, variável para *n* nodos.

Quando um nodo é instanciado o mesmo envia uma mensagem para todas as portas menos para si mesmo. Fazendo esse *Broadcast* entre a porta 5001 a 5010, fazendo isso é possível verificar os vizinhos presentes nessas portas, cada nodo já instanciado ira retornar uma mensagem com seu próprio id, e guardar o id do novo nodo em seu próprio vetor de vizinhos. O nodo instanciado consegue gerar um vetor de vizinhos com os identificadores recebidos pelas mensagens. Depois de um novo vizinho se conectar todos os nodos iram

ordenar seus vetores de vizinhos para que todos tenham uma sequência igual, para assim poder compartilhar memória.

Após iniciar o nodo, ele passa a fazer parte da vizinhança, porém para o mesmo compartilhar memória é necessário fazer a requisição de compartilhamento.Quando é solicitado o compartilhamento de memória, a partir da posição do nodo no vetor de de vizinhos, cada nodo irá receber um id interno, ou seja, um identificador dentro do compartilhamento de memoria. Com base no id interno que serão executados todos os controles feitos pela *hash*.

A ordem dos identificadores informados pelo usuário ira impactar diretamente nos identificadores de memória interna, pois o id interno é gerado a partir da posição dento do vetor de vizinhos e o vetor de vizinhos sempre é ordenado. Após a devida execução de um número de instancias do programa, os nodos conseguem reconhecer a vizinhança atual, ou seja, ver quais são os outros nodos ativos no momento. No entanto para fazer parte do sistema distribuído, pelo menos um desses nodos precisa requisitar o compartilhamento de memória, essa requisição pode ser vista como um acordo entre os nodos, já que após a requisição, todos irão compartilhar um vetor de índices de memória.

Nessa abordagem foi decidido que cada nodo teria 5 posições de memoria, ou seja, o total de memoria será 5, multiplicado pelo número de vizinhos compartilhando memória. Com isso, o compartilhamento e a distribuição de dados é feita baseado no número de nodos que compartilham memória no momento, isso faz com o sistema seja escalável em tempo real, e que suporte uma boa distribuição de carga entre os nodos.

Caso o nodo não esteja compartilhando memória o mesmo não poderá inserir registros, consultar dados, nem mesmo remover algum registro. Ao longo do tempo de execução do sistema, novos nodos podem requisitar a participação no compartilhamento de memória. Com isso, o sistema todo se adapta ao número de nodos que vão se acoplando e, assim permitem uma boa escalabilidade horizontal para a aplicação. Os dados guardados elo sistema são, para fins explicativos, nome e RG de pessoas hipotéticas, e com a informação sobre o RG são calculados os valores *hash* da memória.

O processo de inserção na memória segue uma abordagem onde, tenta gravar em uma posição calculada pela tabela *hash* e caso essa posição esteja ocupada, a inserção é então feita na primeira posição vazia à direita da posição calculada inicialmente. A inserção também trata os casos onde a memória local do nodo está cheia, sendo assim, quando isso ocorrer o nodo responsável irá enviar o dado para o nodo mais a direita, fazendo com que esse insira se possível na sua memória interna. Caso o nodo responsável pelo dado esteja cheio e seja o mais a direita, então ele envia os dados para o primeiro nodo conhecido na memória.

Outra operação realizada pelo sistema é a consulta de informações. Informando o RG cadastrado em um dos nodos, o sistema busca inicialmente na sua memória local e caso não encontre inicia então a busca nos nodos vizinhos, informando a esses uma mensagem com o RG e espera então por uma resposta. O nodo que encontrar o RG informado, retorna uma mensagem de resposta para o nodo requisitante e assim a consulta é exibida na tela, como se o sistema fosse de um nodo só.

A terceira operação feita pelo sistema, é a remoção, funcionando de forma similar à consulta, a remoção baseada no RG tenta buscar e remover o registro localmente, caso

não exista esse registro, o nodo informa essa requisição aos demais nodos. Assim que o registro for encontrado, o nodo responsável remove o registro e informa o nodo que requisitou que a remoção foi bem sucedida. Em casos de RG inexistentes, tanto a consulta como a remoção não retornam nenhum valor ao nodo requisitante.

5. Conclusão

Com o trabalho realizado, foi possível complementar os conteúdos teóricos do componente curricular de Computação Distribuída. O projeto permitiu a aplicação prática dos conceitos estudados sobre sistemas distribuídos e consequentemente ampliou os conhecimentos teóricos e práticos dos autores sobre o tema. Como embasado pela teoria, os sistemas computacionais atuai precisam ser preparados para escalar, horizontalmente e verticalmente, para suprir o aumento no número de usuários que inevitavelmente acontecem.

A evolução das redes, dos *hardwares* e das tecnologias de desenvolvimento, permitem que novas abordagens sejam criadas para gerenciar e otimizar os sistemas computacionais. Os sistemas distribuídos são uma das melhores alternativas quando se precisa de escalabilidade, integridade e tolerância a falhas. Por esse motivo, aumenta ainda mais a importância desse tipo de estudo prático teórico, visando assim a otimização do processamento e do armazenamento de dados.

Conclui-se de forma positiva o projeto, com esta aplicação foi possível chegar a bons resultados referentes aos conceitos básicos de sistemas distribuídos. Ademais, com o desenvolvimento da aplicação, foi possível vivenciar os problemas enfrentados ao desenvolver aplicações distribuídas, bem como entender as particularidades de uma aplicações desse tipo. Sobretudo, tais fatores contribuem para o estudo de sistemas computacionais distribuídos e com certeza no desenvolvimento pessoal e profissional dos acadêmicos.

Referências

Tanenbaum, A. and Van Steen, M. (2007). *Sistemas distribuídos: princípios e paradgmas*. Pearson Educación, 2 edition.