6
Conclusões

Neste trabalho, apresentamos o GridFS como um servidor de arquivos distribuído e sobreposto a diversos sistemas de arquivos locais. Sua interface é baseada em CORBA e uma implementação em Java está disponível. As diversas funcionalidades foram apresentadas e, apesar de termos um sistema distribuído totalmente implementado em Java, os resultados experimentais apresentaram um bom desempenho do sistema quando comparado com outras técnicas e sistemas existentes.

O GridFS foi inspirado em características apresentadas por diversos sistemas como o Globus, o Condor, o Avaki Data Grid e em sistemas de arquivos distribuídos, como o NFS e o AFS. Os sistemas como Globus e o Condor, por atacarem o problema específico do processamento de grandes volumes de dados, definem mecanismos de transferência de arquivos entre as diversas máquinas gerenciadas [1, 3, 18, 19, 21, 4]. O Avaki Data Grid [22] atende boa parte das características desejadas, mas não possui a opção para o armazenamento de metadados e fornecimento de informações auxiliares para os escalonadores de processos. Algumas características do NFS [8] e AFS [9] também foram adotadas por serem soluções tradicionais e bem sedimentadas para o acesso remoto aos arquivos na rede. Vale salientar que, além de agregar os diversos benefícios dos sistemas apresentados, o GridFS pode ser utilizado como um módulo independente, eventualmente eliminando a necessidade da instalação e configuração de infra-estruturas mais complexas, como o Globus. Isso é especialmente útil no momento em que a aplicação precisa apenas de uma solução para o compartilhamento de arquivos, e não de toda uma infra-estrutura para grades.

O GridFS foi motivado pelas necessidades existentes no CSBase, servindo como um mecanismo para o compartilhamento de arquivos em um ambiente para computação em grade. A integração do GridFS no CSBase garante a aplicabilidade deste trabalho e também serviu como guia para a definição das

diversas características presentes no sistema. Apesar de ter sido desenvolvido especialmente para o CSBase, ele pode ser usado isoladamente, fazendo com que diversas aplicações possam se beneficiar da interface remota para manipulação e acesso aos arquivos, ou do uso das operações de cópia entre os diversos servidores, bem como das interfaces de acesso via servidores FTP ou pela interface com o FUSE no sistema operacional Linux.

Alguns aspectos ainda precisam de uma maior atenção no GridFS, principalmente na questão de políticas para o controle de acesso. Atualmente, a acessibilidade ao GridFS deve ser restrita pelo uso de um mecanismo externo que permita o acesso apenas para as máquinas previamente autorizadas. No CSBase, os serviços que intermediam o acesso ao GridFS possuem controle de acesso e estão restritos apenas aos usuários apropriados. Apesar de ser um item fundamental para um servidor de arquivos, o controle de acesso não era necessário inicialmente para o CSBase e deixamos para adicionar essa funcionalidade ao GridFS posteriormente.

Por fim, o GridFS aplica diversos conceitos importantes como um sistema distribuído e pode ser eficientemente usado com uma alternativa para as soluções de compartilhamento de arquivos existentes. Na próxima seção, apresentamos algumas idéias que podem servir para dar continuidade ao trabalho e encerramos este documento.

6.1
Trabalhos Futuros

A interface de navegação do sistema é baseada em funções que retornam o estado atual do sistema de arquivos. Caso o sistema seja usado em uma aplicação onde a árvore de diretórios deva ser atualizada sem a necessidade de uma operação de recarga explícita, a aplicação terá que realizar um *polling* para que seja possível manter a consistência entre a árvore apresentada e a árvore corrente. Deve ser possível que um mecanismo de notificação seja acoplado no sistema, permitindo que *callbacks* sejam chamadas sempre que alguma alteração em determinado ramo da árvore ocorrer. Dado que as alterações nos sistemas de arquivos podem ocorrer externamente ao GridFS, um mecanismo de monitoração, como o FAM[1], pode estar presente para auxiliar nessa tarefa ao nível dos sistemas de arquivos locais. Esse mecanismo

[1] http://oss.sgi.com/projects/fam/

de notificação e monitoramento também pode ajudar para a implementação de algumas políticas de *caching* e replicação de dados usando o GridFS.

Apesar de termos implementado um mecanismo de suporte a metadados, um mecanismo de indexação e busca pode ser implantado no sistema para explorar essa funcionalidade, permitindo que arquivos sejam localizados com base nos metadados associados. A utilização de *transdutores* pode ser realizada para a extração de metadados referentes aos arquivos [29] e a criação de diretórios virtuais com os resultados das consultas pode se caracterizar como uma boa forma de organização dos arquivos com base em diferentes visões sobre os dados [30, 31].

Dada a limitação dos ambientes disponíveis durante o desenvolvimento desta dissertação, novos testes de desempenho podem ser executados em *clusters* e redes com características diferentes das apresentadas neste trabalho. Especialmente no que se refere a um maior número de máquinas em uma rede de longa distância, pela utilização de discos com uma maior taxa de leitura/escrita, ou mesmo pela instalação de softwares para a criação de discos virtuais. Um maior número de máquinas pode ser utilizado visando verificar o comportamento da rede para os diversos métodos de cópia antes que o limite de velocidade do disco seja atingido, ou, considerando o limite de velocidade dos discos, discos virtuais em memória podem ser utilizados para fazer com que a leitura e escrita de informações ocorra com altas taxas de transferência. Esses discos podem ser criados pelo uso de técnicas como o ramdisk[2], ramfs, ou tmpfs.

O sistema poderia realizar um cálculo automático para o tamanho do *buffer* TCP de acordo com as características da rede em uso. Essa técnica é conhecida por *autobuf*[3] e atualmente o GridFS permite apenas que o usuário especifique manualmente o valor do *buffer* usado. O sistema pode ser melhorado para realizar algumas medições, potencialmente usando pequenos arquivos de teste, e assim calcular dinamicamente o tamanho da janela TCP. A principal dificuldade enfrentada na implementação atual se refere a impossibilidade de se alterar o tamanho da janela depois que o servidor de soquetes está ouvindo uma determinada porta. Assim, servidores de soquetes precisariam ser criados especificamente para cada variação no tamanho da janela.

Com o objetivo de aumentar a escalabilidade do servidor para as

[2]http://www.vanemery.com/Linux/Ramdisk/ramdisk.html
[3]http://dast.nlanr.net/projects/Autobuf/

operações de acesso remoto, e evitar a necessidade do mecanismo de *leasing*, podemos fazer com que o *offset* da operação de leitura ou escrita seja controlado pelo cliente e enviado como parâmetro para cada chamada remota. Caso a queda de desempenho pela abertura repetitiva do arquivo seja elevada, um mecanismo de *caching* pode ser utilizado para manter alguns descritores de arquivos abertos, aumentando o desempenho do sistema e mantendo a alocação de recursos em um nível configurável.

Considerando a aplicação do GridFS no CSBase, o serviço de projetos, por oferecer uma interface RMI que possui uma grande interseção com as funcionalidades do GridFS, pode ser refatorado visando simplificar sua implementação. A utilização do GridFS no serviço de projetos também faria com que a área de projetos fosse capaz de utilizar uma federação de servidores para armazenar os dados dos projetos. Além disso, um escalonador de tarefas no CSBase pode utilizar as estimativas de desempenho de cópia para a avaliação sobre qual máquina de execução deve ser responsável pela execução de um determinado algoritmo.

A utilização da Interface de Invocação Dinâmica de CORBA [24] pode ser utilizada para realizar chamadas assíncronas (*deferred mode*) sobre as operações dos canais de leitura e escrita. Especialmente para o caso de cópias de arquivos, esse procedimento tem o objetivo de aumentar o desempenho dessas operações, dado que o mecanismo implementado atualmente se baseia em chamadas síncronas, que aguardam a confirmação da operação remota. Isso faz com que a origem e o destino dos arquivos não trabalhem em paralelo, dado que enquanto o servidor lê ou escreve os dados, o cliente fica bloqueado. Da mesma forma, enquanto o cliente trata os dados obtidos do servidor, ou prepara um conjunto de informações para ser enviado, o servidor permanece ocioso. O uso de chamadas assíncronas tem uma tendência a aumentar a utilização de ambas as máquinas e, consequentemente, um aumento de desempenho deve ser obtido.

Por fim, o GridFS também pode ser aplicado a outros cenários de computação distribuída como, por exemplo, em aplicações para computação móvel e ubíqua [32, 33, 34]. Aplicações para esses cenários tipicamente precisam compartilhar arquivos e integrar diversas plataformas heterogêneas.