

## Um modelo de armazenamento de metadados tolerante a falhas para o DDGfs

Alexandro S. Soares, Thiago Emmanuel Pereira,  
Jonhunny W. Silva, Francisco Brasileiro  
Universidade Federal de Campina Grande  
Laboratório de Sistemas Distribuídos  
Campina Grande, PB, Brasil  
{alexandro,thiagopdc,jonhunny}@lsd.ufcg.edu.br, fubica@dsc.ufcg.edu.br

### Resumo

*O Desktop Data Grid file system (DDGfs) é um sistema de arquivos distribuído desenvolvido para atender a requisitos de escalabilidade e manutenibilidade não oferecidos por sistemas de arquivos distribuídos amplamente utilizados na prática, como NFS e Coda. No DDGfs, dados e metadados são armazenados em componentes separados. Os dados são armazenados em servidores de dados, enquanto os metadados são mantidos em um único servidor de metadados. Essa arquitetura facilita o projeto do sistema, mas torna o servidor de metadados um ponto único de falha. Apesar de ser considerado um componente confiável, falhas são inevitáveis. Se esse componente falhar, os metadados podem ser perdidos e todos os dados armazenados se tornam inacessíveis. Este artigo apresenta um modelo de armazenamento de metadados que permite que o servidor de metadados seja restaurado após uma falha catastrófica que corrompa seu estado.*

### 1. Introdução

O Desktop Data Grid file system (DDGfs)<sup>1</sup> é um sistema de arquivos distribuído desenvolvido para atender a requisitos de escalabilidade e manutenibilidade não oferecidos por sistemas de arquivos distribuídos amplamente utilizados na prática, como NFS [7] e Coda [10]. No DDGfs, dados e metadados são armazenados em componentes separados. Os dados são armazenados em servidores de dados, enquanto os metadados são mantidos em um único servidor de metadados.

Os servidores de dados residem em estações de trabalho da rede local da corporação. Garantias de disponibilidade dos dados são providas através de um modelo de replicação

<sup>1</sup>Mais informações sobre o DDGfs disponíveis em <http://www.lsd.ufcg.edu.br/ddg>

passiva não-bloqueante que mantém cópias dos dados em servidores de dados distintos, assegurando um nível desejado de redundância.

O servidor de metadados é um componente especial do sistema. Em favor da simplicidade de projeto, assumimos que este resida em uma máquina especialmente reservada a este propósito, sujeita a condições especiais de administração que implementem restrições de acesso, mecanismos de segurança, entre outras medidas, para garantir que o servidor de metadados seja um componente confiável e disponível. Apesar dessas considerações, falhas de software e de hardware são inevitáveis. Uma dessas falhas pode comprometer permanentemente o servidor de metadados e fazer com que as referências para os dados armazenados no sistema sejam perdidas, tornando os dados inacessíveis.

A solução para este problema é conferir redundância aos metadados, tornando o modelo de armazenamento de metadados tolerante a falhas. Neste artigo, propomos uma abordagem de armazenamento de metadados para o DDGfs que torna possível a recuperação do servidor de metadados a partir da replicação dos metadados do servidor central nos servidores de dados. Antes de entrar no mérito da recuperação propriamente dita, apresentaremos na Seção 2 uma visão geral da arquitetura do DDGfs. Em seguida, dedicamos a Seção 3 à apresentação dos metadados e a uma discussão de como estes são armazenados e manipulados no contexto do DDGfs. Na Seção 4, apresentaremos o modelo de recuperação de metadados proposto para o DDGfs. A Seção 5 é reservada à avaliação do modelo proposto e à apresentação dos resultados das simulações realizadas. Por fim, apresentaremos considerações finais e perspectivas para trabalhos futuros na Seção 6.

### 2. Arquitetura do DDGfs

Um instalação do DDGfs é composta por um servidor de metadados e um conjunto de servidores de dados que são acessados por clientes, como ilustrado na Figura 1. O

DDGfs apresenta uma arquitetura híbrida que reúne características de sistemas P2P e cliente/servidor e cujo objetivo é manter dados e metadados armazenados em componentes separados. Diversos servidores de dados armazenam os blocos de dados de maneira colaborativa e distribuída, enquanto os metadados são mantidos em um único servidor central, conhecido por todos os servidores de dados.

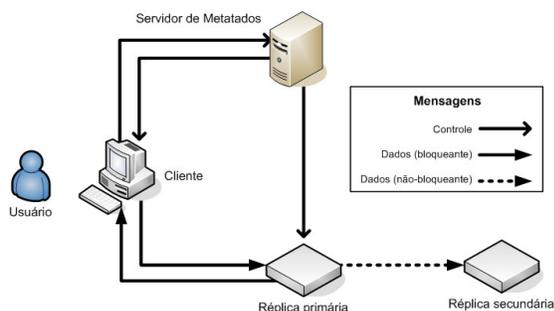


Figura 1. Componentes do DDGfs

O servidor de metadados é um componente confiável e reside em uma máquina sujeita a condições especiais de administração e monitoramento. Além do armazenamento de metadados, o servidor central é responsável pelo serviço de descoberta de servidores de dados e pela coordenação do mecanismo de replicação de dados.

Os servidores de dados são componentes simples que disponibilizam primitivas básicas de escrita e leitura. Ao contrário do servidor central, os servidores de dados não residem em máquinas sujeitas a condições especiais de administração ou monitoramento. As máquinas usadas pelos servidores de dados são estações de trabalho que fazem parte da rede corporativa. As primitivas oferecidas pelos servidores de dados são utilizadas pelos clientes em operações de escrita/leitura de dados; ou por outros servidores de dados, quando precisam atualizar o conteúdo de cópias de um bloco de dados, por exemplo.

Os processos dos usuários têm acesso aos arquivos através de um cliente do DDGfs. Clientes não mantêm informação sobre alocação de dados, isto é, não sabem, a priori, em quais servidores os blocos de dados dos arquivos que desejam acessar estão armazenados. Para obter a alocação dos dados, os clientes usam o serviço de descoberta de servidores de dados, provido pelo servidor de metadados. Após conseguir as referências para os dados, os clientes contatam os servidores de dados para realizar operações de escrita/leitura. Transferências de dados são realizadas apenas entre os servidores de dados e entre servidores de dados e clientes. O servidor de metadados não participa, em hipótese alguma, do processo de transferência de dados. Clientes e servidores de dados podem coexistir

numa mesma máquina. O algoritmo de alocação de dados do DDGfs explora essa possibilidade para fins de melhoria de desempenho.

O modelo de replicação adotado pelo DDGfs mantém cópias dos dados dos clientes em outros servidores de dados. Os dados originais são denominados primários, enquanto as cópias são denominadas de dados secundários ou réplicas. Operações de escrita de dados são realizadas imediatamente apenas nos servidores que mantêm os dados primários. As réplicas são atualizadas posteriormente, em algum instante, sob a coordenação do servidor de metadados. A atualização, isto é, a propagação das alterações nos dados primários para as réplicas, é realizada no modo não-bloqueante. Dessa forma, o cliente não precisa aguardar a conclusão da atualização. O cliente indica o nível desejado de redundância e o servidor de metadados se encarrega de coordenar o processo de propagação de forma a manter o nível de redundância.

### 3. Metadados

O termo metadado é empregado de formas distintas pelas diferentes comunidades que o utilizam. Como parte de uma definição mais ampla [6], metadados são constituídos de informação estruturada e são empregados com o objetivo de descrever, explicar, localizar ou, de alguma forma, facilitar a recuperação, o uso ou o gerenciamento de um ou mais recursos de informação.

No contexto do DDGfs, metadados armazenam informações sobre a localização e a estruturação dos dados armazenados. Mais especificamente, classificamos os metadados do DDGfs em três categorias: nós-i, atributos estendidos e estruturas de dados do sistema.

Os nós-i mantêm informações referentes ao conteúdo dos blocos de dados armazenados pelo sistema que incluem: tamanho dos blocos de dados, proprietário, data de criação, data da última modificação, diretório pai, entre outros.

Os atributos estendidos permitem que informações adicionais definidas pelo usuário sejam agregadas aos dados. O DDGfs faz uso dos atributos estendidos para armazenar informações importantes para o mecanismo de replicação de dados. Uma vez que as alterações dos dados primários não são propagadas imediatamente para as réplicas, faz-se necessário introduzir o conceito de versão para manter o controle de atualização. Logo, um dado é dito consistente se todas as suas cópias possuem o mesmo número de versão dos dados primários a que se referem. O nível de redundância, isto é, o número de cópias que devem ser mantidas para um determinado bloco de dados também deve ser armazenado. O intervalo de tempo requerido para que alterações efetuadas em um bloco de dados comecem a ser propagadas para as réplicas também precisa ser mantido para cada bloco. Esse intervalo é denominado “tempo de

coerência”. Além do nível de redundância e do tempo de coerência, um descritor de tipos deve ser associado a cada um dos blocos de dados armazenados. Esse descritor torna possível a distinção entre dados primários e secundários.

As estruturas de dados do sistema mantêm informações referentes ao conjunto de servidores de dados, à estrutura de diretórios e aos grupos de replicação. A estrutura de diretórios consiste em uma árvore contendo os diretórios e arquivos dos usuários. Todos os usuários têm a mesma visão da árvore de diretórios. Os grupos de replicação consistem em tuplas (nó-i, servidor), que permitem o mapeamento dos nós-i para os servidores de dados.

O espaço necessário para o armazenamento de metadados é consideravelmente pequeno. Um estudo sobre o conteúdo de sistemas de arquivos [3] mostra que um usuário típico armazena cerca de 6.000 arquivos. Considerando que 100 bytes sejam suficientes para armazenar os metadados, são necessários apenas 600KB de metadados por usuário. Por esse motivo, e pelo bem do desempenho, o servidor de metadados mantém os metadados na memória principal e, periodicamente, sincroniza as alterações realizadas nos metadados com um arquivo mantido em disco para assegurar a persistência dos metadados.

### 3.1. Operações envolvendo metadados

Os metadados estão intimamente ligados com os dados aos quais se referem. Quaisquer alterações que os dados sofram refletem alterações nos metadados. Operações de escrita/leitura de dados alteram metadados que armazenam informações do tamanho, datas de acesso e de modificação de um determinado arquivo, por exemplo. A recíproca, no entanto, nem sempre é verdadeira, isto é, há operações que modificam os metadados, mas que não ocasionam alterações nos dados. A chamada de sistema *utimes()* do UNIX é um exemplo de operação que apresenta esse tipo de comportamento. Sua funcionalidade consiste em atualizar as datas de último acesso e de última modificação de um arquivo para a data corrente, sem realizar modificações nos dados.

O DDGfs oferece tanto operações que modificam dados e metadados, quanto as que incidem apenas nos metadados. Na primeira categoria, enquadram-se operações de leitura/escrita de dados (*write()*, *read()*) e operações de criação, remoção de arquivos e truncamento de dados (*create()*, *remove()* e *truncate()*); e operações para criação e remoção de diretórios e de links (*mkdir()*, *rmdir()*, *link()* e *symlink()*). Na segunda categoria, operações para definir e recuperar valores para atributos comuns e estendidos (*getattr()*, *setattr()*, *gettextattr()* e *setextattr()*).

As operações que atuam unicamente sobre os metadados fazem com que haja troca de mensagens apenas entre o cliente e o servidor de metadados. Essas operações são

realizadas no modo bloqueante, isto é, o cliente aguarda até que as alterações nos metadados tenham sido, de fato, registradas pelo servidor de metadados. Usar o modo bloqueante nessa ocasião não produz impacto sobre o desempenho do sistema, uma vez que essas operações são realizadas em um intervalo curto de duração, devido ao fato de os metadados serem mantidos em memória.

Por outro lado, nas operações que atuam sobre os dados, é indispensável a participação dos servidores de dados na troca de mensagens. Os metadados dos dados primários são modificados imediatamente, enquanto os metadados das réplicas são atualizados pelo mecanismo de replicação, no momento em que as alterações estão sendo propagadas para as réplicas.

## 4. Um modelo de armazenamento de metadados tolerante a falhas

A arquitetura do DDGfs mantém dados e metadados armazenados em componentes separados e implementa um modelo de replicação não-bloqueante que mantém um nível apropriado de redundância dos dados, sem comprometer o desempenho do sistema. Os metadados, entretanto, são armazenados em um único componente, o servidor de metadados, que está sujeito a falhas. Como não há redundância sobre os metadados, uma falha pode comprometer permanentemente o servidor de metadados, fazendo com que as referências para os blocos de dados sejam perdidas e, conseqüentemente, os dados armazenados no sistema se tornem inacessíveis. Torna-se evidente, portanto, a necessidade de conceber um modelo de armazenamento de metadados tolerante a falhas que assegure que os metadados possam ser recuperados após a ocorrência de falhas que comprometam o servidor central.

Existem diversas abordagens para conferir redundância aos metadados e, com isso, assegurar que possam ser recuperados após a ocorrência de falhas. Uma delas é o espelhamento (*mirroring*) do servidor de metadados. Essa técnica é simples e consiste em manter uma cópia de segurança do servidor de metadados em uma máquina distinta. Essa técnica é empregada pelo HDFS [1] e pelo GFS [4] para manter uma cópia de segurança do arquivo de metadados armazenado no servidor de metadados. A desvantagem dessa solução está no custo associado em manter dois servidores, ao invés de um. Uma vez que as mesmas considerações especiais de administração e monitoramento devem ser aplicadas ao servidor redundante, o custo de manter o sistema é duplicado, deteriorando a manutenibilidade do sistema.

Uma solução alternativa que oferece um menor custo de manutenção é armazenar réplicas dos metadados nos servidores de dados. Dessa forma, após uma falha que comprometa permanentemente os metadados, o servidor central pode ser restaurado a partir da recuperação das réplicas dos

metadados. Como essa abordagem não compromete a manutenibilidade do sistema, optamos por adotá-la como base para um modelo de armazenamento de metadados tolerante a falhas, como descrito a seguir.

#### 4.1. O modelo

O modelo mantém cópias dos metadados nos servidores de dados, o que torna possível recuperá-los quando ocorrem falhas no servidor central. As réplicas dos metadados são armazenadas pelos mesmos servidores que mantêm os dados aos quais fazem referência. O modelo deve assegurar que é possível recuperar os três tipos de metadados armazenados: nós-i, atributos estendidos e estruturas de dados.

Os nós-i e os atributos estendidos são mantidos nos servidores de dados em uma estrutura de dados idêntica à usada pelo servidor central. Dessa forma, os metadados são mantidos em memória e, periodicamente, sincronizados com um arquivo persistido em disco. Os atributos estendidos não sofrem mudanças com muita frequência e são alterados apenas pela chamada *setextattr()* e recuperadas a partir da chamada *getextattr()*. A propagação das alterações sofridas pelos metadados é realizada pelo mecanismo de replicação de dados. Dessa forma, o modelo de propagação se mantém, fazendo com que alterações sejam realizadas de maneira imediata e em modo bloqueante apenas nos metadados primários e que sejam posteriormente propagadas, em modo não-bloqueante, para as réplicas.

Quanto às estruturas de dados mantidas pelo servidor central, apenas a estrutura de diretórios precisa ser replicada nos servidores de dados. Isto porque as outras duas, grupos de replicação e grupo de servidores de dados, podem ser facilmente recuperadas. Os grupos de replicação podem ser inferidos, uma vez que os metadados são mantidos nos mesmos servidores que os dados correspondentes. O grupo de servidores de dados é normalmente do conhecimento do administrador da rede corporativa. A estrutura de diretórios, no entanto precisa ser persistida. Apesar de ser vista de uma única forma por todos os servidores de dados, estrutura de árvore de diretórios não pode ser replicada por inteiro em todos servidores de dados. A propagação das alterações incidentes nos metadados ocasionariam inconsistências, caso fossem realizadas no modo não-bloqueante; ou teriam forte impacto no desempenho, se fossem realizadas no modo bloqueante. Ao invés de ser replicada por inteiro, cada caminho na estrutura de diretórios é mapeado na forma de uma cadeia de caracteres que é armazenada como atributo estendido do bloco de dados correspondente.

#### 4.2. O processo de recuperação

O processo de recuperação é coordenado por um módulo de recuperação implementado no servidor de metadados.

O servidor de metadados possui dois modos de operação: normal e recuperação. No segundo modo, o módulo de recuperação é ativado.

No modo normal de operação, ao iniciar, o servidor de metadados carrega os metadados a partir do disco. Na ocasião de falha que danifique o arquivo de persistência dos metadados, o servidor precisa ser iniciado no modo de recuperação. Nesse modo, ao invés de carregar as informações de metadados a partir do disco, o servidor de metadados recebe, como parâmetro de entrada, a lista de endereços de servidores de dados. O servidor então inicia o processo de recuperação.

O servidor de metadados contata cada um dos servidores de dados presentes na lista e requisita as listas de nós-i e de atributos estendidos. As listas originais de nós-i e de atributos estendidos são obtidas através de uma simples operação de junção das listas fornecidas pelos servidores de dados. Os grupos de replicação são inferidos a partir da localização das réplicas nos metadados. A estrutura de diretórios é remontada sob demanda, a partir das cadeias de caracteres armazenadas nas listas de atributos estendidos recuperadas.

O processo apresentado considera que todos os servidores de dados estão disponíveis desde o instante em que o processo de recuperação é iniciado até a recuperação total. No entanto, é possível que a falha que tenha comprometido o servidor de metadados tenha comprometido também os servidores de dados. Por exemplo, em consequência dos efeitos colaterais de uma queda de energia. Dessa forma, o processo de recuperação deve contemplar cenários nos quais nem todos os servidores de dados estejam disponíveis. Servidores de dados indisponíveis são detectados durante a fase de contatação. Servidores presentes na lista de servidores fornecida que não respondam à requisição feita pelo servidor central serão considerados indisponíveis.

A recuperação total dos metadados não depende, necessariamente, da participação de todos os servidores de dados, mas do subconjunto destes que mantêm os dados primários. Servidores de dados que mantenham apenas réplicas secundárias são dispensáveis, se todos os que mantêm dados primários já estiverem disponíveis. Entretanto, se, para um dado bloco de dados, apenas servidores secundários que mantêm a réplica deste estiverem disponíveis, não se pode aplicar a mesma semântica de recuperação, uma vez que não há como ter certeza da consistência daquela réplica em relação ao dado primário correspondente. Os dados que dispõem apenas de réplicas são denominados indisponíveis. A lista de dados indisponíveis também pode ser utilizada pelo administrador para identificar servidores de dados indisponíveis.

Se for possível tornar disponíveis os servidores de dados ausentes, os metadados são recuperados pelo servidor central tão logo os servidores possam ser contactados. Caso contrário, isto é, se os servidores de dados foram permanen-

temente comprometidos, as réplicas podem ser, a critério do administrador, promovidas a dados primários.

O DDGfs não dispõe de um serviço de verificação de integridade para os dados armazenados. Portanto, o processo de recuperação não verifica a integridade do dados no momento da recuperação. Por outro lado, a integridade do arquivo que mantém os metadados persistidos em disco no servidor central é verificada. Dessa forma, caso esse arquivo seja corrompido, o servidor é iniciado no modo recuperação.

## 5. Avaliação do modelo

A tolerância a falhas do servidor de metadados provida pelo modelo descrito é implementada a partir da replicação passiva dos metadados nos servidores de dados que compõem o DDGfs. Essa abordagem aumenta as chances de recuperação dos metadados após a ocorrência de uma falha que comprometa permanentemente o servidor de metadados. A recuperabilidade do modelo, isto é, a capacidade de recuperação dos metadados a partir dos servidores de dados, precisa ser avaliada.

O espaço de estados do sistema na ocasião de falha é muito grande, pois precisa contemplar o número e as combinações de servidores de dados que podem falhar em conjunto com o servidor central, o nível de redundância e o tempo de coerência dos dados e metadados. Para realizar a avaliação em tempo hábil e varrer uma parcela significativa do espaço de estados, a recuperabilidade foi avaliada a partir de simulações. As subseções seguintes descrevem a metodologia empregada e os resultados obtidos com a simulação do modelo.

### 5.1. Metodologia

O ambiente de simulação é constituído de uma rede local composta por 100 estações de trabalho. Em cada estação de trabalho residem um servidor de dados e um cliente do serviço de armazenamento.

Cada um dos servidores de dados armazena exatamente 6000 arquivos, que é o número médio de arquivos por usuário reportado no estudo de Doceur [3]. Os tamanhos dos arquivos seguem uma distribuição Pareto ( $shape=1.05$ ,  $scale=3800$ ), como sugerido por Crovella [2]. A taxa de transferência de dados do disco foi arbitrada em 57 MB/s. A rede conecta as estações de trabalho a uma banda passante de 100Mb/s. Transferências de dados concorrentes ocasionam a divisão igualitária da banda disponível.

Embora alguns estudos tenham sido feitos sobre caracterização do uso de sistemas de arquivos [8], [3], nenhum destes fornece os dados colhidos. Por outro lado, alguns trabalhos que tornaram seus dados públicos têm menor relevância, devido ao curto período de monitoramento

ou ao reduzido número de componentes monitorados. Devido a ausência de rastros de utilização real de sistemas, decidimos gerar uma carga sintética do acesso aos dados. A carga de trabalho submetida ao sistema é constituída de operações de escrita sequenciais realizadas a uma taxa constante de 1 operação por segundo, durante o período de 30 dias. O intervalo de tempo entre operações sucessivas segue uma distribuição exponencial. O cliente, os servidores de dados e o arquivo envolvidos na operação de escrita são escolhidos ao acaso, respeitando uma distribuição uniforme. Ao término de cada uma das operações, a atualização das réplicas é agendada de acordo com o tempo de coerência definido. O processo de atualização das réplicas transfere o conteúdo inteiro dos dados para as réplicas.

A recuperabilidade do sistema foi mensurada em termos do número de arquivos consistentes no sistema após a falha. Durante o processo de recuperação, um arquivo é dito consistente se o servidor de dados que armazena os seus dados primários está disponível, ou se, ao menos, uma de suas réplicas estiver atualizada e armazenada em um servidor de dados disponível.

Os cenários submetidos à simulação levaram em consideração a variação de três grandezas: o nível de redundância, o tempo de coerência e o número de servidores de dados comprometidos permanentemente pela falha. O nível de redundância indica a quantidade necessária de réplicas de um arquivo mantida pelo sistema. O tempo de coerência indica o intervalo de tempo entre a conclusão de uma operação de escrita em um dado primário e a propagação das alterações realizadas para as réplicas. Nas simulações, os valores atribuídos às variáveis foram: 2, 3, 4 e 5 réplicas, para o nível de redundância; 30, 60, 120, 300 e 600 segundos, para o tempo de coerência; e 5 grupos, cada um contendo 100 combinações de máquinas comprometidas pela falha. O primeiro grupo possui 100 combinações nas quais apenas 1 máquina sofre falha permanente. O segundo considera que 2 máquinas falham e os demais contemplam combinações onde 3, 4 e 5 máquinas falham. Todas as combinações foram selecionadas ao acaso respeitando a uma distribuição uniforme. A simulação totalizou 100 cenários, levando em consideração os grupos ao invés das combinações. Cada um dos cenários foi submetido a 720 ocorrências de falhas em instantes de tempo distintos, uniformemente espaçados.

### 5.2. Resultados

Os resultados das simulações são mostrados na Figura 2, que contém 5 gráficos, um para cada grupo de combinações de máquinas comprometidas pela falha. A figura emprega uma escala logarítmica para facilitar a visualização das curvas.

No eixo das abscissas, estão dispostos os níveis de re-

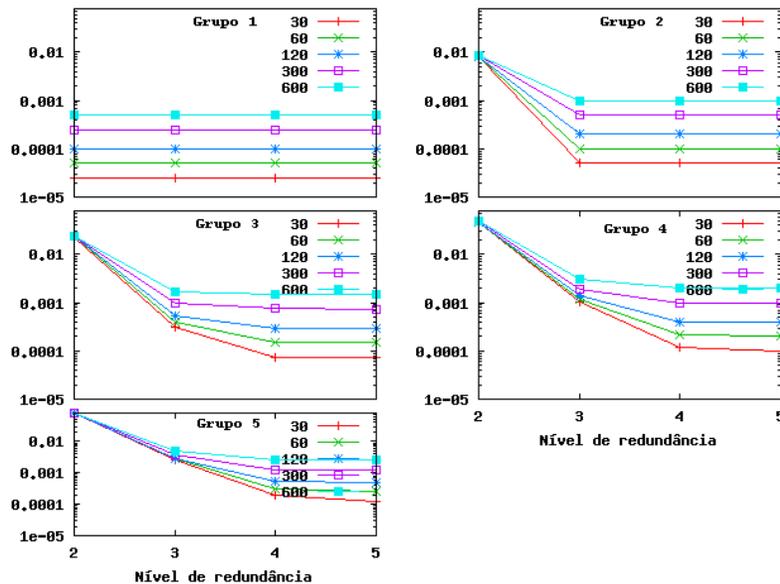


Figura 2. Resultados das simulações para os 5 grupos de máquinas comprometidas pela falha.

dundância. As ordenadas, representam o percentual de arquivos perdidos devido à falha do servidor de metadados e dos servidores de dados de cada um dos grupos. Cada curva corresponde a uma configuração diferente de tempo de coerência.

A análise desses resultados permitiu a confirmação de comportamentos esperados: o percentual de arquivos perdidos é diretamente proporcional ao intervalo de tempo de coerência e ao número de máquinas comprometidas pela falha; e inversamente proporcional ao nível de redundância. As justificativas para esses comportamentos são de fácil compreensão.

Adotar um tempo de coerência pequeno faz com que as atualizações sejam realizadas mais frequentemente, reduzindo o intervalo de tempo no qual as réplicas permanecem inconsistentes e aumentando as chances de recuperação do arquivo. Entretanto, quanto menor o tempo de coerência, maior é o consumo de banda da rede.

O aumento do nível de redundância favorece a recuperabilidade. Os resultados indicam, no entanto, que essa contribuição não é significativa a partir do limiar de 3 réplicas. Uma vez que a adoção de um nível de redundância maior não traz benefícios significativos, o espaço de armazenamento que seria destinado a réplicas adicionais pode ser utilizado para o armazenamento de outros dados primários, favorecendo a capacidade de armazenamento do sistema.

## 6. Considerações finais e perspectivas de trabalho futuro

Neste trabalho, apresentamos um modelo de armazenamento de metadados tolerante a falhas para o DDGfs baseado na replicação passiva dos metadados. De acordo com esse modelo, réplicas dos metadados armazenados no servidor central são mantidas pelos servidores de dados. Essa abordagem aumenta significativamente as chances de recuperação dos metadados após uma falha permanente do servidor de metadados. Os resultados obtidos nas simulações indicam que é possível recuperar efetivamente os metadados a partir do modelo descrito, justificando a viabilidade de se empregar a solução proposta para atribuir redundância aos metadados.

As simplificações impostas ao modelo podem ser exploradas como fonte de trabalhos futuros. Resultados mais precisos podem ser obtidos utilizando rastros reais de utilização do sistema de arquivo, por exemplo. Os rastros sintéticos utilizados neste trabalho consideram que o padrão de realização de operações de escritas é uniformemente distribuído. No entanto, estudos [9], [11] indicam a existência de uma correlação entre arquivos baseada nos seus padrões de acesso e sugerem que o acesso aos arquivos não é realizado segundo uma distribuição uniforme.

## Referências

- [1] D. Borthakur. The Hadoop Distributed File System: Architecture and Design. *Document on Hadoop Wiki*, 2008.
- [2] M. E. Crovella. Heavy-tailed probability distributions in the world wide web. 1998.
- [3] J. R. Douceur and W. J. Bolosky. A large-scale study of file-system contents. *SIGMETRICS Perform. Eval. Rev.*, 27(1):59–70, 1999.
- [4] S. Ghemawat, H. Gobioff, and S.-T. Leung. The google file system. In *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 29–43, New York, NY, USA, 2003. ACM Press.
- [5] J. Howard, M. Kazar, S. Menees, D. Nichols, M. Satyanarayanan, R. Sidebotham, and M. West. Scale and performance in a distributed file system. *ACM Transactions on Computer Systems (TOCS)*, 6(1):51–81, 1988.
- [6] N. I. S. Organization. Understanding metadata. NISO Press, 2004.
- [7] B. Pawlowski, C. Juszczak, P. Staubach, C. Smith, D. Lebel, and D. Hitz. NFS version 3 design and implementation. In *Proceedings of the Summer USENIX Conference*, pages 137–152, 1994.
- [8] D. Roselli, J. R. Lorch, and T. E. Anderson. A comparison of file system workloads. In *In Proceedings of the 2000 USENIX Annual Technical Conference*, pages 41–54, 2000.
- [9] S. Santosh. Factoring file access patterns and user behavior into caching design for distributed file system. Master's thesis, Wayne State University, USA, 2007.
- [10] M. Satyanarayanan, J. Kistler, P. Kumar, M. Okasaki, E. Siegel, and D. Steere. Coda: a highly available file system for a distributed workstation environment. *IEEE Transactions on Computers*, 39(4):447–459, 1990.
- [11] C. Tait and D. Duchamp. Detection and exploitation of file working sets. *Proceedings of the 11th International Conference on Distributed Computing Systems (ICDCS91)*, May 1991.