Injeção de Falhas de Comunicação em Grids com Características de Tolerância a Falhas

Cristina C. Menegotto[1], **Juliano C. Vacaro**[1], **Taisy S. Weber**[1]

[1]Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brazil

{ccmenegotto,jcvacaro,taisy}@inf.ufrgs.br

Abstract. *This paper proposes a methodology for dependability assessment of grids with fault tolerance features using communication fault injection. The methodology was built over an adaption of a performance analysis methodology. As a prove of concept, the methodology is applied to the grid computing platform OurGrid using the injector FIRMI. Results are shown about the dependability of the system.*

Resumo. *Este artigo propõe uma metodologia para avaliação de dependabilidade de grids com características de tolerância a falhas através de injeção de falhas de comunicação. A metodologia foi construída a partir da adaptação de uma metodologia para avaliação de desempenho. Como prova de conceito, a metodologia é aplicada à plataforma para computação em grid OurGrid usando o injetor FIRMI. Resultados são apresentados acerca da dependabilidade do sistema.*

1. Introdução

Falhas são fenômenos comuns em sistemas computacionais. Aplicações distribuídas são suscetíveis a vários tipos de falhas de comunicação, como particionamento de redes, colapso de nodos e atrasos devido à sobrecarga de nodos ou links. Implementar mecanismos de tolerância a falhas não é condição suficiente para um sistema atingir dependabilidade, ou seja, ser capaz de fornecer serviço no qual se pode confiar de modo justificável [Avizienis et al. 2004]. É necessário testar tais mecanismos, de forma que se possa ter confiança de que eles se comportarão como esperado na presença da falhas. Injeção de falhas é a técnica mais adequada para realizar tais testes. Ela atua inserindo falhas artificialmente no sistema sob teste, de forma controlada, visando à obtenção de conhecimento sobre o comportamento do mesmo em presença de falhas [Clark and Pradhan 1995].

Este artigo apresenta uma metodologia para avaliação de dependabilidade utilizando injeção de falhas de comunicação. Tal metodologia é apresentada através de sua aplicação ao OurGrid [Cirne et al. 2006]. A metodologia foi desenvolvida com base nos passos para a condução de um estudo de avaliação de desempenho propostos por Jain [Jain 1991]. Tais passos são os seguintes: estabelecer os objetivos do estudo e definir os limites do sistema, listar serviços do sistema e possíveis resultados, selecionar métricas, listar parâmetros de sistema e de carga de trabalho, selecionar fatores e seus valores, selecionar técnicas de avaliação, selecionar a carga de trabalho, projetar os experimentos, analisar e interpretar os dados e, por fim, apresentar os resultados. Introduzimos um passo de definição da carga de falhas [Madeira and Koopman 2001] a ser injetada na aplicação

sob teste, o qual deve ser cumprido antes do passo de projeto dos experimentos. Também introduzimos a noção de parâmetros de carga de falhas ao passo de listagem de parâmetros de sistema e de carga de trabalho.

OurGrid [Cirne et al. 2006] é uma plataforma livre para computação em grid que visa à execução de aplicações paralelas formadas por tarefas independentes. Neste texto, tais tipos de aplicações serão denominadas trabalhos. Há três componentes principais em sua arquitetura: MyGrid, Peer e User Agent. MyGrid é o componente central do ambiente, responsável pela coordenação e escalonamento de trabalhos. Peers são responsáveis pela organização dos nodos de um dado domínio administrativo, atuando como provedores de nodos disponíveis para MyGrid. User Agents são os componentes que executam as tarefas escalonadas por MyGrid. A plataforma implementa mecanismos de tolerância a falhas como replicação de tarefas e reexecução de tarefas. A versão 3.2.1 foi utilizada para a realização deste trabalho.

Este artigo inova por apresentar a aplicação de uma metodologia completa de injeção de falhas em um sistema desenvolvido por terceiros, sem adaptação que facilitasse os experimentos relatados. Apesar do artigo se restringir ao OurGrid, a metodologia é geral o suficiente para ser aplicada a qualquer sistema distribuído sobre uma rede de comunicação.

Na literatura, há trabalhos que abordam os requisitos de *benchmarks* para avaliação de dependabilidade [Madeira and Koopman 2001] e existem trabalhos sobre *benchmarks* para avaliação de dependabilidade para classes particulares de sistemas como, por exemplo, sistemas operacionais e bancos de dados. Nik Looker e Jie Xu [Looker and Xu 2003] tratam de forma específica a avaliação de dependabilidade de Middleware OGSA através de injeção de falhas de comunicação, porém a abordagem usada não é geral o suficiente para a avaliação de outros sistemas.

Na seção 2, cada passo da metodologia proposta é explicado e exemplificado para o caso particular do OurGrid. Na seção 3, são apresentados os resultados obtidos com os experimentos planejados. A seção 4 apresenta a conclusão do trabalho.

2. Planejamento dos Experimentos

Esta seção apresenta o planejamento dos experimentos realizados no OurGrid, o qual foi feito com base nos passos para avaliação de desempenho apresentados por Jain [Jain 1991]. Cada uma das subseções corresponde à aplicação na avaliação do OurGrid de cada um dos passos propostos por Jain, exceto a subseção 2.8, que estende a metodologia de Jain para injeção de falhas. O passo de listagem de parâmetros de sistema e de carga de trabalho foi estendido com a noção de parâmetros de carga de falhas, que não existia no passo de Jain, e é apresentado na subseção 2.4.

2.1. Estabelecimento dos Objetivos do Estudo e dos Limites do Sistema

O objetivo estabelecido é a avaliação do comportamento do OurGrid em presença de falhas de comunicação (de colapso de nodo, de *colapso de link* e de temporização), testando seus mecanismos de tolerância a falhas. Antes de realizar experimentos de injeção de falhas, é preciso verificar qual é o funcionamento da aplicação em condições normais a fim de se ter um padrão para comparação [Madeira and Koopman 2001]. Assim, também é

objetivo deste estudo a comparação do comportamento do OurGrid em condições normais com o seu comportamento em presença de falhas.

A instalação de OurGrid objeto deste trabalho é local. A tabela 1 apresenta informações sobre os computadores selecionados para os experimentos. Todos eles utilizavam a distribuição Ubuntu 6.10 (Edgy Eft) com kernel Linux 2.6.15-26-386. Optou-se por instalar ambos os componentes MyGrid e Peer na máquina Buick e um componente User Agent em cada uma das demais máquinas da tabela 1.

Tabela 1. Configurações dos computadores disponíveis para experimentos.

nome	modelo	cpu(MHz)	bogomips	memória (KB)
Buick	AMD Athlon(tm) processor	1674.120	3351.10	508196
Jaguar	AMD Athlon(tm) XP 2000+	1674.120	3351.10	508196
Mercedes	AMD Athlon(tm) processor	1674.120	3351.05	508196
Porsche	AMD Athlon(tm) processor	1674.120	3352.97	508260
Bentley	AMD Athlon(tm) XP 2400+	1997.282	3998.54	515940
Maverick	AMD Athlon(tm) XP 2400+	1997.172	3998.41	515940

2.2. Listagem de Serviços do Sistema e Possíveis Resultados

O serviço de escalonamento de tarefas do MyGrid é um serviço essencial oferecido por OurGrid, e inclui mecanismos de replicação de tarefas e reexecução de tarefas em caso de falhas. Ambos os mecanismos são propriedades configuráveis e seus valores padrão são, respectivamente, 1 e 3, ou seja, por padrão, uma tarefa é associada à execução em apenas uma máquina e pode haver até 3 tentativas de execução em caso de falhas. Os possíveis resultados esperados, a priori, do serviço de escalonamento de tarefas incluem saídas desejáveis e não desejáveis. Na ausência de falhas, o desejável é que o serviço possua saídas corretas. Na presença de falhas, o desejável que o serviço se comporte conforme a configuração utilizada para seus mecanismos de replicação e reexecução de tarefas e, quando não puder se recuperar, seja capaz de indicar esta condição ao usuário.

2.3. Seleção de Métricas

Segundo Cirne [Cirne et al. 2006], um dos requisitos para o êxito de OurGrid é que a plataforma seja rápida quanto ao tempo de resposta de um trabalho, ou seja, o tempo que um usuário necessita esperar entre a submissão de um trabalho e a obtenção do resultado da computação. O usuário não tem interesse em medidas em nível de sistema, como *throughput* [Cirne et al. 2006]. Ainda, em uma instalação local de pequeno porte, pode-se esperar que não sejam tão comuns situações de submissão de um trabalho durante execução de outro.

Propomos como métricas relevantes para avaliação de dependabilidade de grids: avaliação da cobertura de falhas (no caso específico, da cobertura de falhas do MyGrid) e razão entre o tempo de resposta de um trabalho em operação normal e tempo de resposta em condições de falhas (quando for possível a recuperação).

2.4. Listagem de Parâmetros de Sistema, de Carga de Trabalho e de Falhas

Um passo importante na avaliação de desempenho ou de dependabilidade é a listagem dos parâmetros que afetam estes atributos. Segundo Jain [Jain 1991], a lista pode ser dividida

em parâmetros de sistema e parâmetros de carga de trabalho. Os parâmetros de sistema identificados para uma instalação local do OurGrid incluem: número de computadores, configuração dos computadores, componentes executados pelos computadores, número de réplicas simultâneas de uma tarefa que MyGrid tentará executar, número de vezes que uma réplica de uma tarefa vai ser criada quando uma máquina que executa o User Agent falha e tipo de escalonador utilizado pelo MyGrid. Os parâmetros de carga de trabalho identificados para uma instalação local do OurGrid incluem: tipo de aplicação a ser executada (*CPU bound* ou *I/O bound*), número de tarefas de um trabalho e tempo médio de execução das tarefas de um trabalho.

Além dos tipos propostos por Jain, parâmetros de carga de falhas devem ser considerados para avaliação de dependabilidade usando injeção de falhas. Tais parâmetros incluem: tipos de falhas a serem injetadas, injeção de falhas em um único nodo ou em mais de um nodo e carga de falhas a ser injetada. A carga de falhas deve ser representativa dos tipos de falhas às quais a aplicação alvo está sujeita em operação normal. Quanto aos tipos de falhas, nos restringimos a falhas de comunicação comuns em sistemas distribuídos como falhas de colapso de nodo, de *colapso de link* e falhas de temporização.

2.5. Seleção de Fatores e seus Valores

Segundo Jain [Jain 1991], a lista de parâmetros obtida anteriormente pode ser dividida em duas partes: parâmetros que serão variados durante a avaliação e que não serão. Os parâmetros variados são chamados *fatores* e seus valores *níveis*. Em geral, a lista de fatores e seus possíveis valores é maior do que os recursos disponíveis permitem variar.

Os parâmetros sobre os quais se espera grande impacto no desempenho devem ser selecionados preferencialmente como fatores [Jain 1991]. Para este artigo, optou-se por utilizar, primordialmente, como fatores: quantidade de tarefas, quantidade de máquinas que executam User Agents (GuMs), número de réplicas e tipo de falha a ser injetada. A tabela 2 apresenta tais fatores associados aos níveis escolhidos. Os níveis escolhidos para o número de GuMs foram limitados pelas máquinas disponíveis.

Tabela 2. Fatores e níveis para a condução do estudo de caso.

Número de GuMs	3, 4 ou 5
Número de tarefas	3, 4 ou 5
Número de réplicas	1 ou 2
Falha a ser injetada	nenhuma, colapso de nodo, *colapso de link* ou temporização

Optou-se por utilizar um tipo fixo de tarefa e compor trabalhos utilizando um número variável de tais tarefas. O escalonador Workqueue with Replication (WQR) [Cirne et al. 2006] também foi fixado como o único a ser utilizado nos experimentos.

2.6. Seleção de Técnicas de Avaliação

Existem diversas técnicas para avaliação de dependabilidade de sistemas, tais como injeção de falhas, modelagem analítica e avaliação empírica durante a vida de um sistema [Clark and Pradhan 1995]. As técnicas de injeção de falhas ainda podem ser divididas entre injeção de falhas baseada em simulação, a qual é útil durante a fase de projeto de

um sistema, e injeção de falhas baseada em protótipo, útil quando já existe um protótipo do sistema a ser testado [Hsueh et al. 1997].

Este trabalho visa à injeção de falhas de comunicação em um sistema pronto. Tal decisão implica na necessidade de seleção de um injetor de falhas adequado para a aplicação a ser testada. O injetor selecionado foi FIRMI [Vacaro and Weber 2006b], um injetor de falhas de comunicação para aplicações distribuídas baseadas em Java RMI, adequado aos propósitos de testar o OurGrid 3.2.1, que é construído sobre Java RMI. Em um experimento anterior [Vacaro and Weber 2006a], FIRMI havia sido aplicado brevemente sobre OurGrid a fim de mostrar as funcionalidades do injetor.

2.7. Seleção de Carga de Trabalho

A carga de trabalho é uma lista de requisições de serviços para o sistema sob teste e é essencial que ela seja representativa da utilização do sistema em condições reais [Jain 1991].

Visando alcançar um tempo de execução relativamente alto e confiabilidade na correta implementação, selecionamos um trabalho que realiza multiplicação de matrizes quadradas dentre as aplicações disponíveis na página do Projeto OurGrid. Cada uma das tarefas deste trabalho é composta pelas subtarefas inicial, remota e final. A subtarefa inicial é responsável pela transferência do código da aplicação para uma máquina do grid, a subtarefa remota é responsável pela execução do código e, por fim, a subtarefa final é responsável pela transferência do resultado para a máquina com MyGrid. A execução começa pela geração de duas matrizes quadradas, cujo tamanho é passado como parâmetro à aplicação. Os elementos gerados para tais matrizes são números randômicos variando entre 1 e 5. Após a geração das duas matrizes, elas são multiplicadas e o resultado da computação e as matrizes geradas são retornados.

Decidimos por um tamanho *n* de matriz que resultasse em tempo de execução relativamente alto. A tabela 3 apresenta os tempos médios, em segundos, de execução em cada uma das máquinas com User Agents (GuMs), fornecendo-se o valor 700 para *n*. A primeira linha mostra os nomes das GuMs. A segunda mostra o tempo médio para transferência do arquivo com o código executável entre a máquina com MyGrid e uma GuM. A terceira apresenta o tempo médio de execução da computação propriamente dita. A quarta mostra o tempo médio para a transferência dos resultados para a máquina com MyGrid. Por fim, a última coluna contém os tempos médios de execução das subtarefas inicial, remota e final em relação a todas as máquinas. A subtarefa final necessita de um tempo médio para concretização maior do que a tarefa inicial, neste caso, pois consiste da transferência de um volume de dados superior ao que a tarefa inicial realiza.

Tabela 3. Tempos médios da tarefa para composição de carga de trabalho.

GuM	Jaguar	Mercedes	Porsche	Bentley	Maverick	**Média (s)**
inicial (s)	0,2184	0,2665	0,3169	0,3607	0,2012	0,2727
remota (s)	627,5845	654,2845	627,3204	388,9807	390,3018	537,6944
final (s)	1,9647	2,6254	1,9226	4,2445	4,1116	2,9737

2.8. Seleção de Carga de Falhas

Para realizar experimentos de injeção de falhas é preciso definir a carga de falhas a ser injetada na aplicação alvo. Nas metodologias convencionais de avaliação de desempenho,

como a proposta por Jain, esse passo não existe. Para o nosso propósito, foram definidos três módulos de carga de falhas: um para emulação de uma falha de colapso, outro para falhas de temporização e o último para *colapso de link*.

A figura 1 apresenta o módulo de carga de falhas desenvolvido para a injeção de uma falha de colapso de nodo em Jaguar usando o injetor FIRMI. A classe `JaguarFaultloadColapso` define a emulação do colapso do nodo Jaguar para o nodo Buick. Quando a terceira requisição RMI recebida pelo nodo Jaguar é interceptada pelo injetor, a falha é ativada. Ativar a falha após a terceira requisição RMI garante que o nodo Jaguar já tenha sido detectado pelo componente Peer antes de sofrer o colapso.

```
1  public class JaguarFaultloadColapso implements Faultload {
2    private int times;
3    public JaguarFaultloadColapso() throws Exception {
4      times = 0;
5      CrashFault c = new CrashFault("buick".split());
6    }
7    public void update(RMIRequest req) throws Exception {
8      times += 1;
9      if (times == 3)
10       Manager.getInstance().getRMIRequestFilter().add(c);
11   }
12 }
```

Figura 1. Módulo de carga de falhas para emulação de falhas de colapso de nodo.

A figura 2 apresenta o `Faultload` desenvolvido para a emulação de falhas de *colapso de link* entre os nodos Jaguar e Buick. A classe `JaguarFaultloadLinkCrash` emula colapsos de link entre os nodos. As falhas são injetadas segundo um padrão de repetição por tempo. O tempo entre ativação de falhas (tbf) é obtido segundo uma distribuição de probabilidade uniforme com valor mínimo de 30 segundos e valor máximo de 60 segundos. O tempo para a duração de cada ativação da falha (ttr) é definido a partir de uma distribuição de probabilidade uniforme com valor mínimo de 20 segundos e valor máximo de 30 segundos.

```
1  public class JaguarFaultloadLinkCrash implements Faultload {
2    public JaguarFaultloadLinkCrash() throws Exception {
3
4      RNG tbf = ProbabilityFactory.getInstance().createRNG(ProbabilityFactory.UNIFORM, "30
         60".split(), 0);
5
6      RNG ttr = ProbabilityFactory.getInstance().createRNG(ProbabilityFactory.UNIFORM, "20
         30".split(), 0);
7
8      LinkCrashFault lc = new LinkCrashFault("buick".split(), new TimeRepetitionPattern(
         tbf, ttr));
9
10     Manager.getInstance().getRMIRequestFilter().add(lc);
11   }
12   public void update(RMIRequest req) throws Exception {
13   }
14 }
```

Figura 2. Módulo de carga de falhas para emulação de falhas de colapso de link.

A figura 3 apresenta o módulo de carga de falhas desenvolvido para a emulação de falhas de temporização entre os nodos Jaguar e Buick. A classe

`JaguarFaultloadTemp` introduz atrasos na comunicação entre os nodos. Cada uma das requisições RMI é atrasada, segundo uma distribuição de probabilidade uniforme, por entre 0,25 segundos e 1 segundo.

```
1 public class JaguarFaultloadTemp implements Faultload {
2   public JaguarFaultloadTemp() throws Exception {
3     RNG delay = ProbabilityFactory.getInstance().createRNG(ProbabilityFactory.UNIFORM, "
          0.25 1".split(), 0);
4     TimingFault tf = new TimingFault(delay, "buick".split());
5     Manager.getInstance().getRMIRequestFilter().add(tf);
6   }
7   public void update(RMIRequest req) throws Exception {
8   }
9 }
```

Figura 3. Módulo de carga de falhas para emulação de falhas de temporização.

2.9. Projeto dos Experimentos

Um experimento de injeção de falhas corresponde a uma rodada de testes utilizando um injetor, uma carga de trabalho e uma carga de falhas [Vacaro and Weber 2006a]. Para a decisão sobre quais experimentos devem ser realizados, visando a obtenção do máximo de informação com o mínimo de esforço, é importante a elaboração de uma lista de *fatores* e seus *níveis* [Jain 1991].

Além do próprio tempo consumido para a execução de um trabalho, características intrínsecas do OurGrid e de sua integração com o injetor de falhas tornam a execução repetitiva de testes de injeção de falhas um processo demasiadamente demorado. O componente Peer pode levar até 10 minutos para detectar se as GuMs que deve monitorar estão disponíveis. Após a coleta de um elemento de uma amostra de um experimento, a execução do injetor deve ser encerrada desligando o componente User Agent e, ao iniciar a próxima coleta, é preciso ligar novamente os componentes e esperar alguns minutos para que o Peer detecte os agentes. Além disso, alguns tipos de falhas, como as de temporização, aumentam significativamente o tempo de resposta. Assim, foi necessário definir um valor viável para o número de elementos de cada amostra e optou-se por 10.

Com base na lista de fatores e níveis selecionados, na necessidade do conhecimento do comportamento de uma aplicação em condições normais antes de realizar injeção de falhas e nas limitações de tempo, foram definidas as configurações dos experimentos prioritários, as quais foram sintetizadas na tabela 4. Para cada coluna, foram realizados experimentos sem injeção de falhas, injeção de falhas de colapso de nodo, de *colapso de link* e de falhas de temporização. A primeira linha indica o número de GuMs de cada experimento; a segunda indica o número de tarefas que compõe o trabalho usado como carga de trabalho; a terceira indica o número de réplicas que o MyGrid pode criar.

Para cada número de GuMs especificado na tabela 4, foram fixadas quais máquinas seriam usadas em todos os experimentos. No caso de 3 máquinas, foram fixadas Jaguar, Porsche e Bentley. No caso de 4 máquinas, Maverick foi fixada adicionalmente. No caso de 5 máquinas, Mercedes foi fixada adicionalmente às 4 máquinas anteriores. Assim, por exemplo, a primeira configuração especificada na tabela 4 refere-se a um experimento com 4 máquinas (Jaguar, Porsche, Bentley e Maverick), uma carga de trabalho composta por 4 tarefas e com o MyGrid configurado para criar uma réplica.

Optou-se por utilizar o injetor de falhas sempre no mesmo nodo em todos os testes. A máquina Jaguar foi selecionada para tal função e FIRMI foi integrado ao OurGrid através da alteração do *script* de inicialização do componente User Agent com a inclusão das opções necessárias ao injetor.

Tabela 4. Configurações de experimentos.

GuMs	4	4	3	4	5	4
Tarefas	4	4	4	5	4	3
Réplicas	1	2	2	2	2	2

A execução propriamente dita de um experimento com uma determinada configuração exige que um número considerável de passos seja seguido. Para este trabalho, a automatização de experimentos mostrou-se essencial e *Shell Scripts* foram desenvolvidos para suprir essa necessidade.

2.10. Análise e Interpretação dos Dados

Os resultados de medidas e simulações são quantidades randômicas, pois o resultado pode ser diferente a cada vez que o experimento é repetido [Jain 1991]. É necessário levar em consideração a variabilidade dos resultados.

Interpretar os resultados da análise é uma parte essencial de um estudo [Jain 1991]. Jain atenta para a necessidade de compreensão de que a análise somente produz resultados e não conclusões. Os resultados provêem uma base sobre a qual é possível tirar conclusões e, quando diferentes analistas recebem o mesmo conjunto de resultados, as conclusões obtidas por cada um podem ser diferentes.

2.11. Apresentação dos Resultados

O passo final de um estudo de avaliação de desempenho ou dependabilidade é comunicar os resultados obtidos. É importante que os resultados sejam apresentados de uma maneira que seja facilmente compreensível pelos interessados [Jain 1991]. Assim, optamos por, na medida do possível, apresentar os resultados de forma gráfica.

Freqüentemente, ao chegar a este passo, o conhecimento adquirido pelo estudo pode requerer que o analista volte e reconsidere algumas decisões feitas nos passos anteriores [Jain 1991]. Por exemplo, o analista pode querer redefinir os limites do sistema ou incluir outros fatores e métricas que não foram considerados anteriormente.

3. Resultados dos Experimentos

Esta seção apresenta os resultados da condução dos experimentos da tabela 4. Os tempos de resposta foram computados sem considerar a intrusividade do injetor, que é desconhecida, mas pode-se estimar que seu valor é baixo devido às características do injetor.

3.1. Comportamentos em Condições Normais

A verificação do comportamento do OurGrid em condições normais é apresentada nesta seção e será usada na seção 3.5 para a comparação com o comportamento do sistema em situações de falhas. O comportamento em condições normais foi avaliado com base na

coleta e análise de amostras correspondentes às configurações de experimentos especificadas na tabela 4.

Quando um trabalho é submetido ao MyGrid, ele determina os recursos que necessita para a sua execução e envia uma requisição ao componente Peer. O Peer associa recursos à requisição e retorna o resultado ao MyGrid. O escalonador WQR envia randomicamente cada tarefa para uma GuM. Quando a execução de uma tarefa termina, se há ainda alguma tarefa a ser submetida, ela é enviada para esta máquina. Quando não há mais tarefas a serem submetidas, uma tarefa em execução, escolhida randomicamente, é replicada.

A tabela 5 apresenta os tempos médios de execução e desvios padrão para cada um dos experimentos de verificação de operação normal do OurGrid determinados na seção 2.9. Os desvios padrão de todas as amostras são da ordem de poucos segundos. Os tempos médios de resposta obtidos dos experimentos em que o número de máquinas é menor do que o número de tarefas se mostraram superiores aos tempos médios de resposta dos experimentos em que o número de máquinas é maior ou igual ao número de tarefas.

Tabela 5. Experimentos sem falhas – médias e desvios padrão.

Experimento	tempo médio (s)	desvio padrão (s)
4 máquinas, 4 tarefas, 1 réplica	640,4	6,0773
4 máquinas, 4 tarefas, 2 réplicas	645,8	5,8651
3 máquinas, 4 tarefas, 2 réplicas	786,4	1,3499
4 máquinas, 5 tarefas, 2 réplicas	789,6	1,9550
5 máquinas, 4 tarefas, 2 réplicas	640,1	6,1001
4 máquinas, 3 tarefas, 2 réplicas	633,3	14,2052

3.2. Comportamentos com Colapso de Nodo

O comportamento do OurGrid diante do colapso de um nodo foi avaliado com base na coleta e análise de amostras correspondentes às configurações de experimentos especificadas na tabela 4. Conforme abordado na seção 2.8, a carga de falhas selecionada para a condução dos experimentos de colapso de nodo é aquela especificada na figura 1. O OurGrid foi capaz de concluir a execução dos jobs em todos os testes executados, a menos do primeiro experimento, conforme será explicado a seguir.

O primeiro experimento de injeção de falhas de colapso de nodo realizado foi o correspondente à seguinte configuração: 4 máquinas, 4 tarefas e 1 réplica. Em todos os elementos da amostra, a emulação do colapso de Jaguar impossibilitou a conclusão do trabalho. Uma exceção foi registrada no arquivo de *log* do MyGrid, indicando a detecção da queda do nodo, segundo média de 934,6 segundos após o início da execução da tarefa, com desvio padrão de 3,0151 segundos.

A impossibilidade de recuperação, nestas condições, era previsível, pois havia somente 4 máquinas disponíveis e MyGrid estava configurado para a execução de 1 réplica. Porém, um novo teste foi realizado: em alguns elementos da amostra, adicionou-se um segundo trabalho a MyGrid, após o componente ter detectado a falha. Em todos os casos, com esta inclusão, MyGrid solicitou ao componente Peer um número de máquinas cor-

respondente ao número de tarefas do trabalho em questão e uma delas foi utilizada para o reescalonamento da tarefa inacabada no primeiro trabalho devido ao colapso emulado.

O segundo experimento de injeção de falhas de colapso de nodo foi o correspondente à seguinte configuração: 4 máquinas, 4 tarefas e 2 réplicas. Em todos os elementos da amostra, a estratégia de replicação possibilitou a conclusão da tarefa que, primeiramente, tinha sido atribuída ao nodo Jaguar. O tempo médio de resposta do trabalho para este experimento foi de 790,9 segundos, sendo o desvio padrão de 1,1972 segundos. Em 9 entre as 10 amostras, MyGrid não teve tempo para detectar a falha emulada em Jaguar antes do final da execução do trabalho. As máquinas de configurações mais privilegiadas, como Maverick e Bentley, terminavam a execução das primeiras tarefas as quais eram atribuídas e então uma delas passava a executar uma réplica da tarefa que, supostamente, ainda estava sendo executada em Jaguar. Assim, a replicação da tarefa em questão em uma de tais máquinas possibilitava a conclusão do trabalho em um tempo inferior à média de 934,6 segundos para a detecção de colapso determinada no primeiro experimento.

A tabela 6 apresenta os tempos médios de resposta e desvios padrão para cada um dos experimentos com injeção de falhas de colapso no nodo Jaguar determinados na seção 2.9, a menos do experimento com 1 réplica.

Tabela 6. Experimentos com colapso de nodo – médias e desvios padrão.

Experimento	tempo médio (s)	desvio padrão (s)
4 máquinas, 4 tarefas, 2 réplicas	790,9	1,1972
3 máquinas, 4 tarefas, 2 réplicas	1177,4	1,4298
4 máquinas, 5 tarefas, 2 réplicas	944,6	193,9789
5 máquinas, 4 tarefas, 2 réplicas	683,2	75,7757
4 máquinas, 3 tarefas, 2 réplicas	633,3	119,6518

3.3. Comportamentos com Colapso de Link

O comportamento do OurGrid na presença de falhas de *colapso de link* foi avaliado com base na coleta e análise de amostras correspondentes às configurações de experimentos especificadas na tabela 4. Conforme abordado na seção 2.8, a carga de falhas selecionada para a condução dos experimentos de *colapso de link* é aquela especificada na figura 2.

O primeiro experimento de injeção de falhas de *colapso de link* conduzido foi o correspondente à seguinte configuração: 4 máquinas, 4 tarefas e 1 réplica. Em 6 dos 10 elementos da amostra, as falhas injetadas provocaram erro e exceções foram capturadas por OurGrid que conseguiu se recuperar. Apesar do número de réplicas estar limitado em 1, a propriedade de reexecução de tarefas foi mantida com seu valor padrão 3, em todos os experimentos. Assim, quando erros ocorreram, MyGrid procurou escalonar novamente a tarefa para a máquina Jaguar. Um fato interessante observado em um dos testes foi MyGrid ter escalonado 4 vezes para a máquina Jaguar a mesma tarefa, até conseguir concluir sua execução, ultrapassando o valor estabelecido para a propriedade. Assim, o sistema não se comportou conforme o esperado.

A tabela 7 apresenta os tempos médios de execução e desvios padrão para cada um dos experimentos com injeção de falhas de *colapso de link* determinados na seção 2.9. O OurGrid foi capaz de concluir a execução dos *jobs* em todos os testes executados.

Tabela 7. Experimentos com colapso de link – médias e desvios padrão.

Experimento	tempo médio (s)	desvio padrão (s)
4 máquinas, 4 tarefas, 1 réplicas	847,5	180,8819
4 máquinas, 4 tarefas, 2 réplicas	778,2	46,1298
3 máquinas, 4 tarefas, 2 réplicas	957,8	140,6997
4 máquinas, 5 tarefas, 2 réplicas	790,5	1,6499
5 máquinas, 4 tarefas, 2 réplicas	764,2	57,8442
4 máquinas, 3 tarefas, 2 réplicas	623,6	25,0519

3.4. Comportamentos com Falhas de Temporização

O comportamento do OurGrid na presença de falhas de temporização foi avaliado com base na coleta e análise de amostras correspondentes às configurações de experimentos especificadas na tabela 4. Conforme abordado na seção 2.8, a carga de falhas selecionada para a condução dos experimentos de temporização é aquela especificada na figura 3. Os testes realizados com falhas de temporização não provocaram exceções no OurGrid, mas proporcionaram alta queda de desempenho no tempo de resposta de trabalhos.

A tabela 8 apresenta os tempos médios de execução e desvios padrão para cada um dos experimentos com injeção de falhas de temporização determinados na seção 2.9. Os desvios padrão das amostras em que o número de máquinas é igual ao número de tarefas e das amostras em que o número de máquinas é menor do que o número de tarefas são da ordem de poucos segundos. Porém, os desvios padrão das amostras em que o número de máquinas é maior do que o número de tarefas são bastante significativos no sentido de que serão claramente perceptíveis pelo usuário. A explicação para tal magnitude dos desvios padrão é simples e deve-se ao fato de que, sendo o número de GuMs disponíveis para o MyGrid superior ao número de tarefa, a máquina Jaguar, selecionada para a emulação das falhas, nem sempre é escalonada para a execução de réplicas primárias.

Tabela 8. Experimentos com temporização – médias e desvios padrão.

Experimento	tempo médio (s)	desvio padrão (s)
4 máquinas, 4 tarefas, 1 réplicas	1360	9,9554
4 máquinas, 4 tarefas, 2 réplicas	1370	4,3716
3 máquinas, 4 tarefas, 2 réplicas	1374,4	3,8644
4 máquinas, 5 tarefas, 2 réplicas	1360,5	8,2898
5 máquinas, 4 tarefas, 2 réplicas	1150,6	340,8470
4 máquinas, 3 tarefas, 2 réplicas	898,2	390,8048

3.5. Comparações entre Condições Normais e de Falhas

Esta seção visa comparar os tempos de resposta obtidos nos experimentos realizados em condições normais com os tempos de resposta obtidos em cada uma das condições de falha exploradas.

A figura 4 relaciona o tempo médio de resposta de trabalhos com a quantidade de máquinas executando User Agents, para experimentos em condições normais (tabela 5), com falha de colapso de nodo (tabela 6), com falha de *colapso de link* (tabela 7) e com

falha de temporização (tabela 8). O número de tarefas foi fixado em 4 e o número de réplicas foi fixado em 2.

Com 3 máquinas, observa-se que o tempo médio de resposta com a emulação de uma falha de colapso é consideravelmente superior ao tempo médio de resposta em condições normais. Conforme a tabela 6, o tempo médio de resposta na presença da falha é de 1177,4 segundos, sendo o desvio padrão de apenas 1,4298 segundos. Ainda, o tempo médio de resposta em condições normais para a mesma configuração é de 786,4 segundos, sendo o desvio padrão de 1,3499 segundos. Como os desvios padrão são baixos, a simples comparação entre as médias indica que há uma queda significativa de desempenho na presença de falhas de colapso para essa configuração. Ainda para 3 máquinas, a inspeção visual da figura 4 indica que o tempo médio de resposta das outras condições de falha também é consideravelmente superior ao tempo médio de resposta em condições normais.

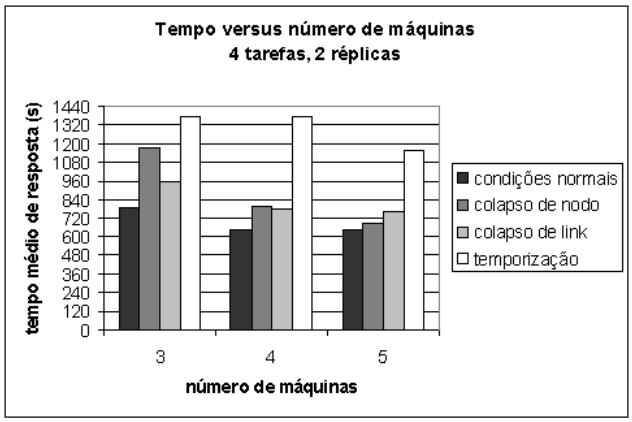

Figura 4. Tempo médio de resposta de trabalhos segundo quantidade de GuMs.

A figura 5 é um gráfico que relaciona o tempo médio de resposta de trabalhos com a quantidade de tarefas utilizadas na composição da carga de trabalho, para experimentos em condições normais, com falha de colapso de nodo, com falha de *colapso de link* e com falha de temporização. O número de máquinas foi fixado em 4 e o número de réplicas foi fixado em 2.

Observa-se, na figura 5 que o aumento no número de tarefas aumenta o tempo médio de resposta tanto para condições normais como para condições de falhas. Também nota-se que, no caso de falhas de *colapso de link*, a maior queda de desempenho ocorre quando o número de tarefas é igual ao número de máquinas.

A figura 6 é um gráfico que relaciona o tempo médio de resposta de trabalhos com a quantidade de réplicas, para experimentos em condições normais, com falha de *colapso de link* e com falha de temporização. O número de máquinas foi fixado em 4 e o número de tarefas foi fixado em 4.

A simples inspeção visual do gráfico indica uma queda de desempenho significativa em presença de falhas de *colapso de link* e de temporização, tanto para 1 réplica como para 2 réplicas. Porém, no caso das falhas de *colapso de link*, os desvios padrão das amostras são altos, indicando que se pode esperar tempos de resposta bastante variáveis para essas configurações. A explicação para tal é relacionada aos instantes em que os colapsos acontecem, os quais são variáveis.

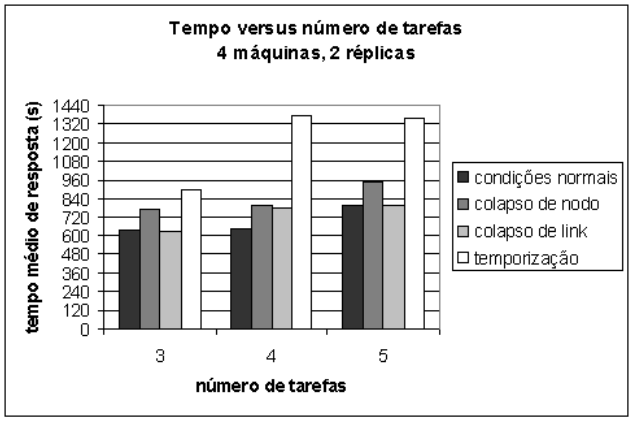

Figura 5. Tempo médio de resposta de trabalhos segundo quantidade de tarefas.

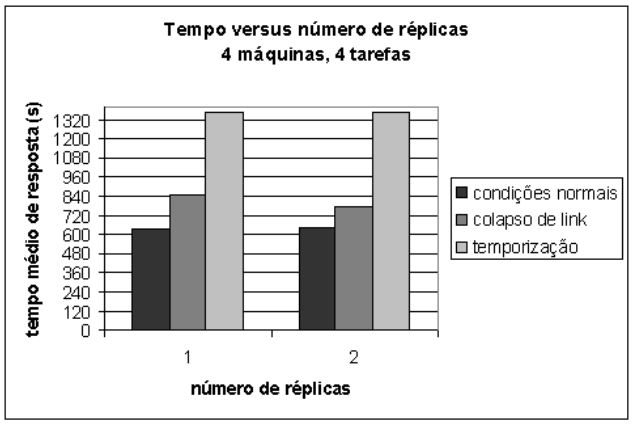

Figura 6. Tempo médio de resposta de trabalhos segundo número de réplicas.

3.6. Análise dos Resultados dos Experimentos

Os experimentos de injeção de falhas realizados mostraram a robustez do sistema alvo quanto a falhas de colapso de nodo quando o número de réplicas é superior a 1 e também quanto a falhas de *colapso de link*. A cobertura do mecanismo de detecção de erros foi de 100% em todos os casos de teste de falhas de colapso com número de réplicas superior a 1, bem como em todos os casos de teste de falhas de *colapso de link*. Ainda, a conclusão das tarefas, nestes testes, foi garantida pela estratégia de tolerância a falhas utilizada. Os experimentos com falhas de temporização, mostraram que tais falhas não podem ser detectadas pelo sistema. A conclusão das tarefas foi atingida em todos os testes relacionados a este tipo de falha, com queda de desempenho elevada.

4. Conclusão

Injetores de falhas de comunicação possibilitam a realização de testes que seriam inviáveis com outros métodos, são mais práticos e menos prejudiciais do que a inserção manual de falhas e possibilitam o teste de sistemas distribuídos de larga escala.

A avaliação de dependabilidade através de injeção de falhas deve ser conduzida de forma sistemática e este artigo propôs uma metodologia para tal baseada nos passos para avaliação de desempenho propostos por Jain [Jain 1991]. Da experiência com o estudo,

nota-se que a escolha de fatores e níveis é uma tarefa complexa e que pode ser mais interessante investir na investigação de diferentes fatores e níveis do que em amostras com grande quantidade de elementos quando o foco é a avaliação de comportamentos em presença de falhas. A automatização de testes também mostrou-se essencial para que um número adequado de testes pudesse ser realizado.

O trabalho também mostrou que adaptar uma metodologia de avaliação de desempenho para experimentos de injeção de falhas é vantajoso, pois permite aproveitar a vasta experiência da área de avaliação de desempenho para a avaliação de dependabilidade em *grids* com características de tolerância a falhas. Muita pesquisa ainda é necessária para o desenvolvimento de *benchmarks* para avaliação de dependabilidade de grids e subsistemas de grids [Dikaiakos 2007].

Referências

Avizienis, A., Laprie, J.-C., Randell, B., and Landwehr, C. (2004). Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing*, 1(1):11–33.

Cirne, W., Brasileiro, F., Andrade, N., Costa, L., Andrade, A., Novaes, R., and Mowbray, M. (2006). Labs of the world, unite!!! *Journal of Grid Computing*, 4(3):225–246.

Clark, J. A. and Pradhan, D. K. (1995). Fault injection: A method for validating computer-system dependability. *Computer*, 28(6):47–56.

Dikaiakos, M. D. (2007). Grid benchmarking: vision, challenges, and current status: Research articles. *Concurrency and Computation: Practice and Experience*, 19(1):89–105.

Hsueh, M.-C., Tsai, T. K., and Iyer, R. K. (1997). Fault injection techniques and tools. *Computer*, 30(4):75–82.

Jain, R. (1991). *The Art of Computer Systems Performance Analysis: techniques for experimental design, measurement, simulation and modelling*. John Wiley & Sons, Inc., New York, USA, 1st edition.

Looker, N. and Xu, J. (2003). Assessing the dependability of ogsa middleware by fault injection. In *Proceedings of the 22nd International Symposium on Reliable Distributed Systems*, pages 293–302.

Madeira, H. and Koopman, P. (2001). Dependability benchmarking: making choices in an n-dimensional problem space. In *Workshop on Evaluating and Architecting System Dependability (EASY)*.

Vacaro, J. C. and Weber, T. S. (2006a). Injeção de falhas na fase de teste de aplicações distribuídas. In *Anais do SIMPÓSIO BRASILEIRO DE ENGENHARIA DE SOFTWARE, SBES, 20.*, volume 1, pages 161–176, Florianópolis. Porto Alegre: SBC.

Vacaro, J. C. and Weber, T. S. (2006b). Um injetor de falhas para a avaliação de aplicações distribuídas baseadas em rmi. In *Anais do Workshop de testes e tolerância a falhas, WTF, 7.*, volume 1, pages 159–170, Curitiba. Porto Alegre: SBC.