APLICAÇÃO DE TOLERÂNCIA A FALHAS EM UM SISTEMA CLIENTE-SERVIDOR NA VARIAÇÃO DE MÚLTIPLOS SERVIDORES UTILIZANDO PYRO

Ezequiel Ramos, Juliani Schlickmann Damasceno
Universidade do Estado de Santa Catarina
Departamento de Ciência da Computação
200, Rua Paulo Malschitzki, 89219-710 Joinville, Brasil
E-mail: {ezequielmr94, juliani.schlickmann}@gmail.com

Abstract *- The necessity of decentralized systems is a reality nowadays, where there is the desire to the widest public possible. Distributed systems have the purpose to render the service offered* ***available, integrated*** *and* ***reliable.*** *These pillars are essential for exist a fault tolerant system. Lots of services such as electronic mail and sharing multimedia data (streaming) must have been developed thinking how it is must behave in fault situations. Applications that must be executed in real time have to possess more and more this feature of revert critical situations. We are motivated with the purpose of developing a fault-tolerant distributed system, this paper intend to show the process done in the development of a client-server system in the variation of multiple servers. The services offered will be the echo operations and the possibility of retrieving the list of messages already sent by the user. Throughout the study will have raised the concepts and the metrics used for the success of the development of the fault tolerant system.*

Resumo - A necessidade de sistemas descentralizados é uma realidade nos dias atuais, onde há o anseio de abranger o maior público possível. Sistema distribuídos possuem como propósito tornar o serviço oferecido **disponível**, **íntegro** e **confiável**. Esses pilares são essenciais para existir um sistema tolerante a falhas. Muitos serviços como correio eletrônico e compartilhamento de dados multimídia (*streaming*) devem ser desenvolvidos pensando em como devem agir em caso de falha. Aplicações que devem ser executadas em tempo real devem possuir ainda mais essa característica de reverter situações críticas. Motivados com o propósito de desenvolvimento de um sistema distribuído tolerante a falha, o presente artigo visa demonstrar o processo realizado no desenvolvimento de um sistema cliente-servidor (*client-server*) na variação de múltiplos servidores. Os serviços oferecidos serão as operações de *echo* e a possibilidade de recuperação da listagem de mensagens já enviadas pelo usuário. Ao longo do estudo serão levantados os conceitos e as métricas utilizadas para o sucesso do desenvolvimento do sistema tolerante a falhas.

I. INTRODUÇÃO

O cenário tecnológico e econômico atual faz com que cada vez sejam mais

comuns o desenvolvimento de sistemas distribuídos. Sistemas distribuído podem ser definidos como componentes localizados em computadores que estão interligados por uma rede comunicando-se e controlando suas ações através de mensagens. A possibilidade de compartilhar recursos e possuir um sistema escalável são fatores que influenciam o desenvolvimento de aplicações distribuídas [1]. Contudo, deve se atentar às características dos sistemas distribuídos que podem se tornar empecilhos quando se está desenvolvendo aplicações. O fato de não haver um controle centralizado dos componentes e suas atividades geram como reflexos a concorrência de componentes, a inexistência de um relógio global que sincronize as ações dos recursos, assim como a possibilidade de falha de componentes isolados [1]. Essas características levam a necessidade de pesquisas a fim de estudar meios de construção de aplicações confiáveis.

Ao planejar e desenvolver um sistema distribuído características como, flexibilidade, transparência, escalabilidade, desempenho e confiabilidade devem ser almejadas. Essas propriedades são benéficas aos proprietários e aos usuários do sistema. No que tange à confiabilidade, quanto mais confiável é o sistema, menos perdas irão existir e um serviço de melhor qualidade irá ser ofertado [4]. Ao referir-se a aplicações confiáveis, pode-se entender como um ambiente de sistema distribuído tolerante a falhas. Atualmente há uma obrigatoriedade de que os serviços sejam ofertados sem interrupções, ou seja, mesmo em presença de falhas parciais no sistema, essas não devem ser refletidas durante a utilização por parte do usuário [2]. As falhas devem ser contornadas utilizando algum meio de controle, seja em *hardware*, utilizando replicação de componente, como por exemplo, o uso de múltiplos servidores com uma função redundante. No exemplo dado, perante a falha do servidor principal da aplicação qualquer outro deve estar apto para assumir a função, não permitindo assim que o serviço oferecido fique indisponível.

A disponibilidade é um atributo da confiabilidade e para muitos sistemas é considerado um aspecto crítico. Pois em alguns cenários se um serviço parar haverá uma grande perda, seja economicamente ou fisicamente falando [4]. Devido essa criticidade é essencial que os componentes do sistema executem suas atividades corretamente. A fim de aprofundar o conhecimento e por em prática os conceitos a respeito de disponibilidade e integridade, o presente artigo tem o objetivo de relatar o processo realizado no desenvolvimento de um sistema de *echo* utilizando a variação de múltiplos servidores. A aplicação desenvolvida usufrui do conceito de Objetos Remotos do Python (*Python Remote Objects* - *PYRO*) que possui propósito semelhante ao da funcionalidade de Invocação de Métodos Remotos (*Remote Methods Invocation* - *RMI*) do JAVA. Na sessão de desenvolvimento do sistema no presente estudo, são discriminadas as tecnologias utilizadas bem como a motivação de usufruí-las.

II. TOLERÂNCIA A FALHAS

Para conseguir implementar um sistema tolerante a falhas, deve-se compreender o que é uma falha. Uma falha é a causa primária da ocorrência de um erro. A

falha ocorre no universo físico, exemplos são a queda de energia e queima de componentes. A Figura 1 faz a representação das possíveis consequências de uma falha.

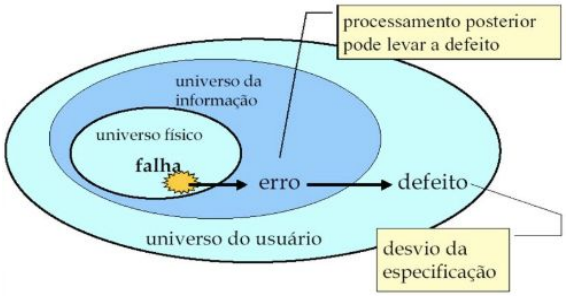

Figura 1 - Falha, Erro, Defeito

Um sistema tolerante a falhas não permite que sejam gerados erros ou defeitos. Para contornar faltas são diversos os obstáculos existentes que deverão ser ultrapassados. Dentre eles pode-se listar:

- Saber como evitar, detectar e contornar falhas no projeto;
- Saber como gerenciar a alta complexidade dos sistemas;
- Saber como conciliar alta confiabilidade e alta disponibilidade com as crescentes demandas por alto desempenho;
- Entre outros;

A habilidade de tolerar falhas leva o alcance de um certo nível de qualidade pelo sistema distribuído (dependabilidade). As técnicas utilizadas para garantir o correto funcionamento do sistema perante a falhas baseiam-se em replicação de componentes ou execução de algoritmos especiais [6].

Muitos problemas de aplicações distribuídas tolerantes a falhas não são corrigidas efetivamente, contudo, são invisíveis aos olhos do usuário. O uso de redundância permite que o sistema continue sua execução mascarando assim os problemas ocorridos. Sistemas tolerantes a falhas são comumente vistos em áreas como:

- Medicina;
- Transporte aéreo;
- Telefonia;
- Transmissão de dados multimídia;
- Economia (bancos, bolsa de valores)

O desenvolvimento de sistemas para essas áreas devem ser realizados prevendo as possíveis falhas que possam vir a ocorrer, pois, se a aplicação não conseguir contornar o problema irão ocorrer perdas graves.

III. TRABALHOS CORRELATOS

A necessidade de desenvolvimento de sistemas confiáveis instigam pesquisadores a buscarem entender meios de alcançarem melhores resultados no que diz respeito a tolerância de falhas. Na presente seção são mencionados alguns estudos onde os autores visam a tolerância de falhas utilizando a replicação de servidores.

O estudo "A Distributed Fault-Tolerant Design for Multiple-Server VOD Systems" dos autores Ing-Jye Shyu e Shiuh-Pyng Shieh [7] desenvolvem dois métodos de recuperação de reprodução para sistemas distribuídos de vídeo. Os métodos são aplicados a para sistemas cliente-servidor na variação de múltiplos servidores e visam minimizar as perdas existentes em aplicações de Vídeo sob Demanda (VOD).

No estudo "Application-Transparent Fault Tolerance in Distributed Systems" [3] o autor realiza um caso de estudo considerando o problema de prover um serviço confiável apesar da existência de falhas, para isso utiliza o método de servidores replicados. O objetivo

do estudo é criar uma camada para detectar e controlar a ocorrência da falha, para que a aplicação não sinta reflexos do problema ocorrido.

Os autores do estudo "Using replication for increased availability of a distributed telecommunication management system" discorreram sobre o uso de replicação para prover a escalabilidade de tolerância a falhas em aplicações distribuídas. A contribuição do estudo é a introdução a um *framework* de tolerância a falhas fundamentado em orientação a objetos.

Durante a busca por trabalhos correlatos notou-se que os autores buscam diversos meios para prover uma aplicação tolerante a falhas. Alguns implementaram a comunicação em grupo do JAVA RMI para prover tolerância a falhas, utilizando a técnica de replicação ativa de componentes. Outros desenvolveram algoritmos a fim de reduzir as perdas existentes em determinados tipos de sistemas, como a VOD. As medidas criadas juntamente com a redundância de servidores proveram resultados positivos para os autores.

IV. DESENVOLVIMENTO DE UM SISTEMA DE ECHO TOLERANTE A FALHAS NA VARIAÇÃO MÚLTIPLOS SERVIDORES

Planejar o desenvolvimento de um projeto é uma tarefa complexa quando se tem um mercado tecnológico tão amplo como o atual. Quais seriam os critérios corretos para definir qual é a melhor tecnologia a ser aplicada? Devido a inexistência de um melhor caminho, optamos pelo o desenvolvimento do projeto de tolerância a falhas utilizando as tecnologias que nos proporcionam maior segurança.

Os requisitos obrigatórios impostos para o desenvolvimento foram a utilização de Invocação de Métodos Remotos (RMI), com isso abstrair a responsabilidade (de criação e finalização de *sockets*) das aplicações cliente e servidor. Outro requisito mínimo é a existência de tolerância a falhas utilizando a replicação do serviço servidor, fazendo assim com que uma possível queda do servidor não impacte na utilização do sistema. A aplicação deverá prover as funcionalidades de envio de mensagens a serem espelhadas pela aplicação servidora e possibilitar que o usuário consiga recuperar a lista de mensagens já espelhadas.

A Tabela 1 a seguir mostra quais tecnologias foram empregadas para o desenvolvimento do presente projeto.

Tecnologia	Descrição
Python	Linguagem de programação, dando suporte para as versões 3 e inferiores.
PYRO4	Possibilita a utilização de métodos remotos, semelhante ao JAVA RMI.

Tabela 1 Principais tecnologias

A motivação para utilizar a linguagem de programação *Python* provém dos fatores de ser uma tecnologia em ascensão dentro do território nacional. Grandes organizações já aplicam *Python* no desenvolvimento de seus produtos, além de possuir uma grande comunidade nacional e internacional que está disposta a ajudar na correção de problemas ou mesmo esclarecer dúvidas. A manutenção da

tecnologia é realizada pela comunidade. O *Python* permite que sejam desenvolvidos diversos tipos de aplicações como:

- Construção de sistemas Web com Django, Flask, Pyramid, etc;
- Análise de dados, Inteligência Artificial, Machine Learning e etc com Numpy, Pandas, Matplotlib, etc;
- Construção de aplicativos com Kivy e Pybee;
- Construção de sistemas desktop com Tkinter, WxPython, etc;

Devido a isso e ser uma tecnologia simples de aprender, desde sintaxe a conceitos um pouco mais complexos, o Python foi escolhido para ser empregado no trabalho final da disciplina de Sistemas Distribuídos.

O Python pode ser considerada uma tecnologia grande, pois sua comunidade sempre faz melhorias para aumentar sua robustez e novas tecnologias para elevar seu potencial. Com isso em 1998 foi desenvolvido o suporte para utilização de Métodos Remotos do Python (PYRO). A versão 4.60 do PYRO foi aplicada ao projeto para que este seja utilizado por pessoas com Python 3 ou inferiores instalado na máquina. Algumas das características que nos fez utilizar o PYRO foram:

- Ser extremamente portável;
- Trabalhar entre arquiteturas e sistemas operacionais diferentes;
- Pode ser utilizado com IPv4, IPv6 e sockets de domínio Unix;
- As invocações são realizadas em uma via para ganho de desempenho;
- Entre outros aspectos;

Além disso, pode-se utilizar o PYRO para distribuir e integrar vários tipos de recursos ou responsabilidades computacionais, como recursos de *hardware*, ou mesmo recursos de informação, como níveis de privilégio [5]. Com isso, optamos pelo aprendizado dessa ferramenta para utilização no trabalho. Com auxílio da documentação conseguimos utilizar a invocação de métodos remotos, bem como o tratamento de possíveis falhas.

A. INSTALAÇÃO PYTHON/PYRO

O *Python* é padrão em distribuições Linux e MacOS. Em ambientes Windows deve-se realizar a devida instalação seguindo as orientações do *site* oficial do *Python* (https://www.python.org/). Como o propósito do presente estudo não é aprofundar-se nos quesitos de instalação das tecnologias será apenas norteado o caminho para realização das mesmas.

O *Python Remote Object* (PYRO) não está nativamente incorporado como biblioteca da linguagem de programação. Contudo, sua instalação é simples, pode ser utilizado em distribuições Linux, MacOS ou Windows. Para realizar a instalação do PYRO basta acessar o site do PYRO 4.60 (https://pythonhosted.org/Pyro4/index.html)

B. ASPECTOS PYRO NO SISTEMA

Assim como o JAVA RMI, o PYRO possui um Servidor de Nomes (*Name Server*) que é uma ferramenta para auxiliar a manter os objetos remotos na rede. O Servidor de Nomes permite que seja atribuído um nome lógico para o objeto PYRO ao invés de sempre informar o exato código ou nome do objeto e sua respectiva localização [5].

Para auxílio nesse quesito de localização dos objetos remotos, foi utilizado o Servidor de Nome do PYRO para que a aplicação servidora realizasse o registro do Objeto Remoto (*Remote Object*) contendo as funcionalidades da aplicação. O Servidor de Nomes deve ser iniciado previamente à tentativa de inicialização da aplicação servidora, para isso, no **Terminal** (Linux/MacOS) ou **Prompt Comando** (Windows) deve ser executado o seguinte comando: ***pyro4-ns***. Após instanciado o Servidor de Nomes, poderá ser instanciada quantas vezes haja necessidade a aplicação servidora. A cada instância do serviço servidor deve ser informado o nome a ser utilizado pelo mesmo, essa informação é utilizada para o controle de tolerância a falha da aplicação.

Após ter o Servidor de Nomes instanciado e aplicação servidora iniciada pode ser executada a aplicação cliente que irá usufruir do objeto remoto cadastrado no serviço de nomes. A partir desse momento a aplicação passa ter acesso aos métodos de serviço de *echo* bem como o de recuperação das mensagens enviadas. Para alcançar o objetivo de tolerância a falha os servidores replicados mantém-se atualizados conforme última interação realizada entre o cliente e o servidor principal da aplicação distribuída. Um esboço genérico do funcionamento da aplicação pode ser vista na Figura 2.

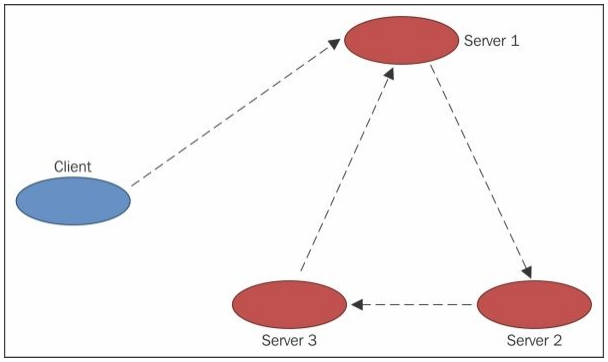

Figura 2 - Modelo utilização

Em presença de falha no servidor principal, o servidor secundário continua a execução da aplicação sem que a falha seja percebida pelo usuário final. Com isso temos sempre um objeto intacto na rede até que a aplicação seja finalizada.

V. CONSIDERAÇÕES FINAIS

O projeto proposto como trabalho de conclusão da disciplina de Sistemas Distribuídos fez com que a teoria aprendida no decorrer das aulas fosse empregada na prática. O que nos fez planejar e desenvolver uma aplicação buscando a dependabilidade do sistema proposto. Além disso, aprender uma nova tecnologia fez com que nós como acadêmicos pudéssemos expandir nosso conhecimento técnico. A utilização do PYRO como meio para atingir resultados semelhantes ao JAVA RMI, nos fez verificar os conceitos referentes a invocação de métodos remotos o que era um dos propósitos do desenvolvimento do trabalho.

É visível a ascensão dos estudos referentes a confiabilidade de software. Prover um serviço de qualidade com o propósito de crescer e ser bem visto perante ao mercado de sistemas é o objetivo da maior parte das organizações. Em sistemas distribuídos uma

das características primordiais é o funcionamento da aplicação do seu início a fim, mesmo na ocorrência de falhas. Devido a isso um sistema que tolere falhas de uma maneira transparente tende a fornecer um bom serviço.

VI. REFERÊNCIAS

[1] COULOURIS, George; DOLLIMORE, Jean; KINDBERG, Tim. **Distributed Systems**: Concepts and Design. 4th. ed. [S.l.]: Addison Wesley Publishers Limited, 2005. 944 p.

[2] LUNG, Lau Cheuk. **Experiências com Tolerância a Faltas no CORBA e Extensões ao FT-CORBA para Sistemas Distribuídos de Larga Escala.** 2001. 240 p. Tese de Doutorado (Programa de Pós-Graduação em Engenharia Elétrica)- Universidade Federal de Santa Catarina, Florianópolis, 2001. Disponível em: <https://repositorio.ufsc.br/handle/123456789/81470>. Acesso em: 23 maio 2018.

[3] BECKER, Thomas. **Application-Transparent Fault Tolerance in Distributed Systems**. 1994. 10 p. Article (Computer Science Department)- University of Kaiserslautern, Germany, 1994. Disponível em: <https://ieeexplore.ieee.org/document/289937/>. Acesso em: 25 maio 2018.

[4] DAI, Y. S. et al. A study of service reliability and availability for distributed systems. **Reliability Engineering and System Safety**, Singapore, n. 79, p. 103-112, set. 2002. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0951832002002004>. Acesso em: 23 maio 2018.

[5] DE JONG, Irmen. **PYRO - Python Remote Objects - 4.60**. Disponível em: <https://pythonhosted.org/Pyro4/#>. Acesso em: 24 maio 2018.

[6] WEBER, Taisy Silva. **Tolerância a falhas:**: conceitos e exemplos. 24 p. Artigo (Programa de Pós-Graduação em Computação)- Instituto de Informática, Universidade Federal do Rio Grande do Sul, [S.l.], Disponível em: <http://www.inf.ufrgs.br/~taisy/disciplinas/textos/ConceitosDependabilidade.PDF>. Acesso em: 24 maio 2018.

[7] SHYU, Ing-jye; SHIEH, Shiuh-pyng. A Distributed Fault-Tolerant Design for Multiple-Server VOD Systems. **Multimedia Tools and Applications**, The Netherlands, n. 8, p. 219-247, jan. 1999. Disponível em: <https://link.springer.com/article/10.1023/A:1009685918587>. Acesso em: 24 maio 2018.

[8] OVE, K. M. Goschka; SMEIKAL, R. **Using replication for increased availability of a distributed telecommunication management system**. 2004. Disponível em: <https://ieeexplore.ieee.org/document/1385832/>. Acesso em: 26 maio 2018.