UNIVERSIDADE FEDERAL DO PAMPA

RODRIGO FERREIRA GONÇALVES

IMPLEMENTAÇÃO DE UM *CLUSTER FAILOVER* UTILIZANDO *SOFTWARE* PROPRIETÁRIO

Bagé
2013

RODRIGO FERREIRA GONÇALVES

IMPLEMENTAÇÃO DE UM *CLUSTER FAILOVER* UTILIZANDO *SOFTWARE* PROPRIETÁRIO

Trabalho de Conclusão de Curso apresentado ao Curso de Especialização em Sistemas Distribuídos com Ênfase em Banco de Dados da Universidade Federal do Pampa, como requisito parcial para obtenção do Título de Especialista.

Orientador: Carlos Michel Betemps

Co-orientador: Sandro da Silva Camargo

Bagé
2013

RODRIGO FERREIRA GONÇALVES

IMPLEMENTAÇÃO DE UM *CLUSTER FAILOVER* UTILIZANDO *SOFTWARE* PROPRIETÁRIO

Trabalho de Conclusão de Curso apresentado ao Curso de Especialização em Sistemas Distribuídos com Ênfase em Banco de Dados da Universidade Federal do Pampa, como requisito parcial para obtenção do Título de Especialista.

Trabalho de Conclusão de Curso defendido e aprovado em: 6 de Agosto de 2013.

Banca examinadora:

_____

Prof. Msc. Carlos Michel Betemps
Orientador
UNIPAMPA

_____

Prof. Dr. Leonardo Bidese de Pinho
UNIPAMPA

_____

Prof. Msc. Érico Marcelo Hoff do Amaral
UNIPAMPA

AGRADECIMENTOS

Agradeço a meus pais, que, ao longo desta jornada, pela força que me deram e o convívio familiar que foi muito importante para eu desempenhar meus esforços na especialização e que eu pudesse me preparar para enfrentar uma nova realidade profissional. A todos os professores do Pós – Graduação em Sistemas Distribuídos com Ênfase em Banco de Dados da Universidade Federal do Pampa - UNIPAMPA, por compartilharem seus vastos conhecimentos e em especial o meu orientador e co-orientador, Prof. Carlos Michel Betemps e Prof. Sandro da Silva Camargo, por aceitar, acreditar e orientar este trabalho e também aos colegas de curso pela amizade e companheirismo e principalmente a Deus por ter me dado forças e coragem para realização deste trabalho.

"Seja quem você for, Seja qualquer posição que você tenha na vida, no nível social mais alto ou mais baixo. Tenha sempre como meta muita força, muita determinação e sempre faça tudo com muito amor e com muita fé em Deus, que um dia você chega lá, de alguma maneira você chega lá."

Ayrton Senna

RESUMO

Fornecer serviços sem interrupção é um requisito critico, pois uma falha pode gerar danos financeiros à empresa, bem como a sua reputação. Tendo em vista este cenário, este trabalho teve por objetivo implementar uma solução de Alta Disponibilidade utilizando *software* proprietário. Neste contexto levantou-se a hipótese de que realizada a implementação do *Cluster Failover*, se garantiria a Alta Disponibilidade das informações distribuídas entre servidores e dados. A metodologia utilizada para atingir o objetivo deste trabalho foi abordar revisões bibliográficas sobre o tema e realizar a implementação e experimentação de um *Cluster Failover*. Como resultado constatou-se a possibilidade de realizar ajustes nas configurações do *Cluster*. Concluiu-se que os servidores que participaram da implementação responderam à maioria dos testes aplicados, garantindo assim a Alta Disponibilidade das informações.

Palavras-chave: *Cluster Failover*; Implementação; Alta Disponibilidade.

ABSTRACT

Provide services without interruption are a critical requirement, because a failure can cause financial damage to the company and its reputation. Given this background, this study aimed to implement a High Availability solution using proprietary software. In this context arose the hypothesis that implementing of the Cluster Failover, would ensure the high availability of information distributed across server and data. The methodology used to achieve the objective of this study was to discuss literature reviews on the topic and carry out the implementation and experimentation of a Failover Cluster. As a result it was found the possibility of making adjustments to the settings of the Cluster. It was concluded that the servers that participated in implementing answered to most of the tests, thus ensuring high availability of information.

Keywords: Failover Cluster; Implementation; High Availability.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

BIOS - *Basic Input/Output System* (Sistema Básico de Entrada e Saída)
CPU - *Central Processing Unit* (Unidade Central de Processamento)
CRC - *Cyclic Redundancy Codes* (Redundância Cíclica Códigos)
DBMS - *Database Management System* (Sistema de Gerenciamento de Banco de Dados)
DC - *Data communications* (Comunicações de Dados)
DML - *Data Manipulation Language* (Linguagem de Manipulação de Dados)
DNS - *Domain Name System* (Sistema de Nomes de Domínios)
GPO - *Group Policy Objects* (Objetos de Diretiva de Grupo)
IP - *Internet Protocol* (Protocolo de Internet)
LAN - *Local Area Network* (Rede de Área Local)
OSI - *Open Systems Interconnection* (Interconexão de Sistemas Abertos)
SGBD - *Data Base Management System* (Sistema Gerenciador de Banco de Dados)
PC - *Personal Computer* (Computador Pessoal)
SQL - *Structured Query Language* (Linguagem de Consulta Estruturada)
TCP - *Transmission Control Protocol* (Protocolo de Controle de Transmissão)
TMR - *Triple Modular Redundancy* (Redundância Modular Tripla)
VPN - *Virtual Private Network* (Rede Privativa Virtual)

SUMÁRIO

INTRODUÇÃO

Atualmente, devido às necessidades de atender diferentes requisitos do negócio, muitas organizações estão migrando seus sistemas corporativos do ambiente de rede local para a ambiente Web. Nesta nova realidade, fornecer serviços sem interrupção é um requisito crítico, pois uma falha pode ser significante tanto na geração de impactos financeiros a curto prazo quanto na geração de danos à reputação empresarial a longo prazo (ZHANG, ABDELZAHER, STANKOVIC, 2004).

Uma solução viável para aumentar a disponibilidade dos serviços é a utilização de *clusters*, fazendo com que no momento que um dos nós falhar, o controle seja transferido para outro, fazendo o serviço continuar sendo oferecido mesmo com a falha de um nó.

Os *clusters* surgiram com o objetivo de suprir algumas dificuldades dos sistemas centralizados e também para aumentar o desempenho das aplicações, redes e ferramentas padronizadas para computação distribuída de alto desempenho. Com seu aparecimento foram necessárias algumas mudanças no desenvolvimento das aplicações, que aconteceram devido às tendências das aplicações serem construídas para rodar em ambientes descentralizados. A composição de um *cluster* abrange questões complexas de pesquisa em computação tais como escalabilidade, disponibilidade, tolerância às falhas e ao alto desempenho (ALECRIM, 2013).

A fim de tornar a falha imperceptível ao usuário do *cluster*, a técnica de *failover* permite o redirecionamento das requisições para outro nó. No contexto de banco de dados, um *Cluster Failover* é útil se um nó falha, ou se ocorre qualquer manutenção planejada nos recursos do Sistema de Banco de Dados (ASWINI, MANISANKAR, JAGADEESAN, 2012).

As principais conquistas tecnológicas do século XX se deram no campo da aquisição, processamento e da distribuição de informações. Com o avanço destas tecnologias, o tempo gasto com a coleta, processamento, distribuição e análise de dados tem se tornado invisível para os usuários. As soluções tecnológicas evoluíram, surgindo técnicas de alta disponibilidade e os *clusters* podem oferecer escalabilidade e disponibilidade de recursos, serviços e aplicativos (TANEBAUM, 2003).

1.1 Problemas de Pesquisa

Com o avanço tecnológico e econômico, várias empresas necessitam cada vez mais manipular informações de maneira segura e contínua. Para isso, necessitam de servidores que possam disponibilizar dados armazenados de forma rápida e eficaz com disponibilidade aos usuários. Neste contexto, a implementação do *Cluster Failover* irá garantir Alta Disponibilidade dos recursos e aplicações, deixando estas informações distribuídas entre os servidores?

1.2 Justificativa

A motivação que levou à realização deste trabalho se deve a relevância do tema para as organizações que utilizam *softwares* proprietários para o gerenciamento de dados. Outro fator considerável para a escolha do tema foi que o ambiente corporativo onde foi aplicada esta implementação prefere utilizar a plataforma *Windows* para Servidores e para Banco de Dados o *SQL Server,* pois já utiliza há anos e pensa ser arriscado migrar para uma plataforma de distribuição livre. Uma das vantagens da utilização do *Windows Server 2008* e *SQL Server 2008* é poder realizar uma implementação de Alta Disponibilidade em *hardwares* e *softwares.*

Esta implementação será importante para o pesquisador, pois com o conhecimento adquirido no trabalho haverá uma agregação de valor na vida profissional e acadêmica.

A implementação do *Cluster Failover* irá oferecer várias melhorias em termos de Alta Disponibilidade e Escalabilidade de recursos. Também irá proporcionar mecanismos tolerantes à falhas, com capacidade de permitir acrescentar novos recursos ou substituir os existentes, pois tais mudanças não serão perceptíveis aos usuários.

1.3 Objetivos

1.3.1 **Objetivo Geral**:

- Implementar uma solução de alta disponibilidade utilizando *software* proprietário.

1.3.2 **Objetivos Específicos:**

- Realizar uma pesquisa bibliográfica sobre abordagens e recursos para implementação de alta disponibilidade;
- Fornecer detalhes sobre a metodologia neste trabalho;

- Descrever o ambiente proposto para a realização da implementação;
- Fazer a implementação do *Cluster Failover* nos servidores;
- Aplicar implementações e configurações nos sistemas que farão parte do *cluster*;
- Validar a implementação do *cluster* com ferramentas de analise de desempenho;
- Simular falha de conexão com servidor após a implementação;
- Testar o desempenho dos Sistemas;
- Mostrar os resultados obtidos com os experimentos realizados.

1.4 Metodologia

Foi realizada uma pesquisa bibliográfica no intuito de se obter a fundamentação teórica necessária à compreensão dos aspectos relacionados à implementação.

O procedimento que foi utilizado consiste em um estudo científico de um tema específico, com a finalidade de obter generalizações. A monografia apresentou os fatores que constituíram e influenciaram as implementações aplicadas.

Nesse estudo foi utilizada a técnica bibliográfica ou de fontes secundárias, que visa recolher e levantar informações, a fim de se escolher e estudar uma série de autores que abrangem o tema de interesse para o investigador.

Este trabalho descreveu a implementação de um *Cluster Failover*, o qual foi aplicado da seguinte maneira:

Definição de tema: Onde foi decidido o tema e a forma de implementação, com base em pesquisas bibliográficas.

Fase de análise: Foi realizada a análise de como a implementação do *Cluster Failover* iria distribuir os dados entre os servidores.

Implementações: Nesta fase foram instalados e configurados os recursos nos servidores, bem como as instalações dos *softwares* e configurações dos computadores.

Testes: Neste momento foram realizados os testes para obter os resultados das aplicações, onde foi comentada toda parte de desenvolvimento deste estudo.

1.5 Estrutura do Trabalho

Este trabalho está organizado em sete capítulos, cujos conteúdos são estruturados na tentativa de conduzir de forma clara e concisa diversos assuntos pertinentes ao entendimento da proposta principal aqui exposta, que relacionam conteúdos de significativa importância para a realização da análise dos dados.

Os capítulos dois, três e quatro são compostos pela Revisão Bibliográfica, é onde está todo material pesquisado para o desenvolvimento deste trabalho, também onde serão visualizados temas relativos a Banco de Dados, Replicação de Dados, Confiabilidade, Redundância e Tipos de *Clusters*. O capítulo cinco descreve o processo de desenvolvimento do trabalho, envolvendo as Técnicas de Implementação e a Descrição do *Cluster* Implementado. O capítulo seis mostra os testes realizados com os Experimentos e Dados Coletados. No capítulo sete são apresentadas as conclusões deste trabalho e apontadas propostas para trabalhos futuro.

2 REVISÃO BIBLIOGRÁFICA

Este trabalho esta organizado em sete capítulos, cujos conteúdos são estruturados na tentativa de conduzir de forma clara e concisa diversos assuntos pertinentes ao entendimento da proposta principal aqui exposta e que relacionam conteúdos de significativa importância para a realização da análise dos dados.

2.1 Banco de Dados

De acordo com Melo (2004), Bancos de Dados é uma coleção de dados inter-relacionados logicamente e coerentes com algum significado inerente. Segundo o mesmo autor, um Banco de Dados é projetado, construído e povoado com dados para um propósito específico, possui aplicações pré-concebidas e visa atender certo grupo de usuários. Seu objetivo é fornecer uma representação única, não redundante e estritamente dimensionada dos dados de uma aplicação.

Os Bancos de Dados ditos relacionais organizam seus dados em tabelas, que são estruturadas por linhas, chamadas de tuplas, e colunas, chamadas de campos. Já, o Sistema de Gerenciamento de Banco de Dados (SGBD) – *Database Management System* (DBMS)- é um *software* especial que permite que os dados sejam controlados em um só lugar, tornando-os disponíveis para diversas aplicações.

O SGBD serve como uma interface entre o Banco de Dados comum e os diversos programas de aplicativos. Os SGBD superam muitas das limitações do ambiente de arquivos tradicionais:

- Os dados são independentes dos programas de aplicativos;
- A redundância e a inconsistência de dados são reduzidas;
- A complexidade do Banco de Dados é reduzida pelo gerenciamento consolidado dos dados;
- As informações são mais fáceis de acessar e usar (LAUDON, 2004).

2.2 Arquiteturas de Sistemas de Banco De Dados

A arquitetura de um Sistema de Banco de Dados é dividida em três níveis, conhecidos como nível interno, nível conceitual e nível externo.

De modo geral o nível interno, também conhecido como nível físico, é o mais próximo do meio de armazenamento físico, ou seja, é aquele que se ocupa do modo como os dados são fisicamente armazenados. O nível externo, igualmente conhecido como nível lógico do usuário, é o mais próximo dos usuários, ocupa-se do modo como os dados são vistos por usuários individuais. O nível conceitual, conhecido também como nível lógico comunitário ou nível indireto sem qualificação, é um nível de simulação, ou seja, é um processo de projetar um modelo computacional de um sistema real e conduzir experimentos com este modelo com o propósito de entender seu comportamento e/ou avaliar estratégias para sua operação (DATE, 2004).

2.3 Comunicações de Dados

As requisições aos Bancos de Dados de um usuário final são transmitidas da estação de trabalho do usuário, que pode estar fisicamente afastada do próprio Sistema de Banco de Dados, para alguma aplicação disponível (*on-line)*, embutida ou não, até o SGBD, sob a forma de mensagens de comunicação.

De modo semelhante, as respostas do SGBD e da aplicação para a estação de trabalho do usuário também são transmitidas sob a forma de mensagens. Todas essas transmissões de mensagens têm lugar sob o controle de outro componente de *software*, o gerenciador de comunicações de dados DC (*Data Communications*).

O gerenciador DC não faz parte do SGBD, mas é um sistema autônomo. Porém, como o gerenciador DC e o SGBD são claramente obrigados a trabalharem em harmonia, às vezes, os dois são considerados parceiros de igual nível em um empreendimento cooperativo de nível mais alto, denominado Sistema de Banco de Dados/Comunicações de Dados (Sistema DB/DC). O SGBD toma conta do Banco de Dados e o gerenciador DC manipula todas as mensagens de e para o SGBD ou, mais precisamente, de e para aplicações que utilizam o SGBD (DATE, 2004).

Este capítulo tratou sobre tópicos de Banco de Dados, assunto imprescindível para dar continuidade ao estudo, como fornecer base para a compreensão do funcionamento dos mecanismos de redundância aplicados nesta implementação, assunto que será abordado no próximo capítulo.

3 CONCEITOS ASSOCIADOS À TOLERÂNCIA ÀS FALHAS

O capítulo 3 tem por objetivo apresentar os principais conceitos sobre Redundância, Redundância de Informação, Redundância Temporal, Redundância de *Hardware*, Redundância de *Software*, Redundância de *Software*, Confiabilidade, Replicação de Dados e Tipos de Replicação no *Microsoft SQL Server* 2008.

3.1 Redundância

Pode existir redundância em vários componentes, como fontes de alimentação, interface de rede, processadores, *links* de comunicação e servidores. A redundância é uma prática importante para sistemas de missão crítica, pois se um componente falhar o sistema continua funcionando normalmente.

Um exemplo bem claro são os equipamentos de uma rede de computadores como os *hubs*, *switches* e *routers*, que são atualmente os periféricos construídos com a solução de redundância. A redundância é utilizada para implementar técnicas de tolerância à falha, podendo ser aplicada em diversas formas (WEBER, 2002).

3.2 Redundâncias de Informação

Segundo Weber (2002), este tipo de redundância utiliza uma técnica de codificação para detecção de erro ou mascaramento de falhas. Neste modelo de redundância, os bits ou sinais extras são armazenados ou transmitidos junto ao dado, sem que contenham qualquer informação útil.

Existem alguns exemplos de redundância de informação como os códigos de paridade, onde para cada n bits são armazenados n+1 bits. O bit extra é usado apenas para indicar se a quantidade de bits de dados com valor 1 é par ou ímpar.

Permite também detectar falhas de natureza simples de bit, ou seja, que afeta apenas 1 bit de uma palavra. Outro exemplo bem claro de códigos utilizados para verificar erros são o código de correção de erros (Código de *Haming*) e os códigos cíclicos – *Cyclic Redundancy Codes* (CRC). No modelo Código de *Haming*, múltiplos bits de paridade são adicionados aos dados, cada um associado a um subconjunto de bits.

Os códigos cíclicos são códigos aplicados a blocos de dados, sequência de bits associados a polinômios. Utilizando um polinômio gerador de grau x, para calcular os bits extras, é possível detectar sequências de bits errados com comprimento menor que x.

3.3 Redundância Temporal

Este tipo de redundância utiliza o tempo para implementar uma dada função destinada a detectar ou tolerar falhas, onde esta função pode ser implementada sob a forma de repetição de cálculos, comparação de resultados, repetição de envio de mensagem ou comparação de valores.

Em sistemas onde o tempo não é crítico ou o processador tem tempo de ociosidade, pode ser usado para detectar falhas transitórias de *hardware*.

Existem duas aplicações usuais de redundância temporal:

1ª Aplicação: Detecção de falhas transitórias: Repete o processamento. Com a existência de resultados distintos e diferentes é uma indicação muito forte de uma falha transitória.

2ª Aplicação: Detecção de falha permanente: Nesse modelo repete-se a computação com as informações codificadas e antes da comparação com o resultado anterior, decodifica as informações (WEBER, 1990, 2002).

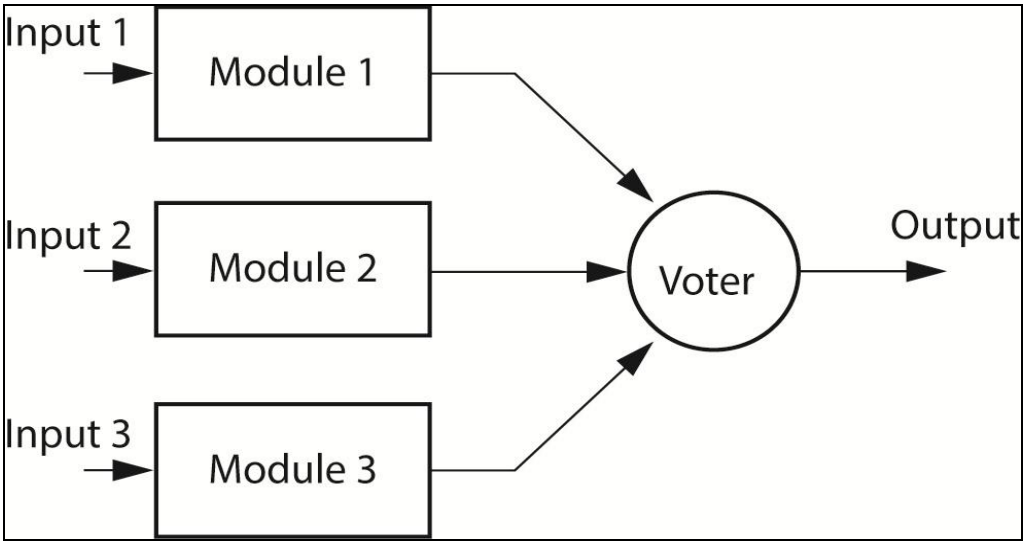
3.4 Redundância De *Hardware*

A redundância de *hardware* está baseada na replicação de componentes físicos. Classificada em:

- **Redundância de *Hardware* Passiva**

Neste tipo, componentes redundantes são utilizados para mascarar falhas, ou seja, corrige erros sem implicar ações do sistema. Todos os elementos efetuam as mesmas atividades e o resultado é determinado por votação, que é a *Triple Modular Redundancy* (TMR).

Figura 1 - *Triple Modular Redundancy* (TMR).



Fonte: (http://hpc.ee.kmutnb.ac.th/~vara/perf/node132.html).

De acordo com Shoman (2002), o sistema indicado na figura 1 é constituído por três módulos: módulo 1, módulo 2 e módulo 3, todos com a mesma saída. Os resultados dos três módulos são comparados pelo elemento votador (*Voter),* que verifica e dá o parecer da saída do sistema. Se todos os três módulos estão operando corretamente, todos os resultados são iguais, assim, a saída do sistema está correta. Porém, uma observação importante é quando a saída não é única, ou seja, quando existe mais de um *output*, porque isto aumenta a fragilidade do *voter* e diminui a confiabilidade da operação.

- **Redundância de *Hardware* Ativa**

Esta redundância possui diversos elementos de processamento, mas apenas um funciona por vez, os outros são suplementos. Se uma falha é encontrada em uma unidade de operação, então esta é retirada e substituída por uma suplente através de um circuito de "*switching*" [interligação de dois ou mais circuitos].

É utilizada em aplicações que suporta permanecer em um estado errôneo durante um determinado período de tempo, onde este tempo é suficiente para a detecção do erro e recuperação para um estado livre de falha. Este tipo de redundância requer a detecção de erros, diagnóstico e reconfiguração.

3.5 Redundância de *Software*

Conforme Weber (2002), uma das formas que garante a redundância de *software*, sem a ocorrência de erros, é a utilização de independentes versões de *software* [como é óbvio tornando o custo mais elevado], não se usando a hipótese de que grupos independentes de programação irão incorporar aos mesmos (modo comum) erros de *software*, degradando a quantidade de redundância fornecida.

Existem outras maneiras de aplicar a redundância de *software* para detectar falhas e mascaramento de falhas que não utilizam cópia de *software* igual, que são a Diversidade e os Blocos de Recuperação.

A Diversidade é uma técnica de redundância empregada para conseguir uma programação diversitária (“n” versões), ou seja, várias alternativas de implementação são desenvolvidas. Portanto, na fase de testes, erros eventuais podem ser localizados e corrigidos. Sendo assim, pode-se optar pela alternativa em que se detectou a menor ocorrência de erros.

A técnica de Blocos de Recuperação usa teste de aceitação, ou seja, programas são executados e testados um a um até que o primeiro passe no teste de aceitação. Dessa forma, programas secundários só serão necessários na detecção de um erro no programa primário.

3.6 Confiabilidade

Segundo Vollertt (1996), a confiabilidade é uma função que representa a probabilidade de um sistema estar operante ou não, implica em algumas condições que são fundamentais:

- **Estado Operacional no Início do Período**

É uma premissa básica, pois não é possível falar de confiabilidade em sistemas que já partem operando com defeito.

- **Especificação**

Para saber se um sistema é confiável ou não é necessário que se tenha uma especificação de como ele deve funcionar. Podemos tomar uma ponte como exemplo, ela deve apresentar uma especificação de quantas toneladas de peso suporta, a fim de evitar que um caminhão com uma carga muito pesada tente fazer a travessia e a ponte venha abaixo.

- **Condições Definidas**

As condições como temperatura do ambiente ou nível de umidade do ar devem ser conhecidas, a fim de que o sistema funcione de maneira adequada.

- **Período de Funcionamento**

O tempo em que o sistema estará no ar e funcional deve ser conhecido. No caso da ponte, supondo que ela seja elevada e possa ser usada somente das 6h às 18h, deverá ser especificado então que no período das 18h até às 6h ela estará indisponível, não por ter ocorrido algum problema, mas por questões relacionadas à especificação.

3.7 Replicação de Dados

Segundo Freitas (2003), a Replicação de Dados é utilizada para aumentar a disponibilidade dos dados. Existem casos onde será necessário replicar todos os Bancos de Dados dentro da transação, criando um Banco de Dados Distribuído totalmente replicado, gerando assim características como Disponibilidade, Aumento do Paralelismo e Aumento do *Overhead* de Atualização.

- **Disponibilidade**: Se um nó (ponto de acesso da rede) que contém uma relação "x" falhar, consequentemente essa relação poderá ser encontrado em outro nó. Assim, o Sistema poderá continuar a processar consultas envolvendo a relação "x", apesar da falha de um nó.
- **Aumento do Paralelismo**: Vários nós podem processar consultas paralelamente em "x". Quanto mais réplicas de "x" haver, maior é a chance de que os dados necessários sejam encontrados no nó onde a transação está sendo executada. Portanto, a replicação de dados minimiza o movimento de dados entre os nós.
- **Aumento do *Overhead* de Atualização**: O sistema deve assegurar que todas as réplicas de uma relação "x" estejam consistentes, caso contrário processamentos errados podem acontecer. Assim, toda vez que "x" é atualizada, a atualização deve ser propagada a todos os nós que contém replicas. O resultado é um *overhead* aumentado.

3.8 Replicação de Banco De Dados

Conforme Silberschatz e Karth (1991), O objetivo de um mecanismo de replicação de dados é permitir a manutenção de várias cópias idênticas de um mesmo dado em vários sistemas gerenciadores de Banco de Dados. A Replicação de Banco de Dados pode se dar de duas formas:

- **Replicação Síncrona**: Todas as cópias ou replicações de dados serão realizadas no instante da sincronização e consistência. Se alguma cópia do banco é alterada, essa

alteração será imediatamente aplicada a todos os outros bancos dentro da transação. A Replicação Síncrona é apropriada em aplicações comerciais onde a consistência exata das informações é de extrema importância.

- **Replicação Assíncrona:** Armazena e faz a replicação. Em um primeiro momento, a cópia de dados fica fora de sincronia entre os Bancos de Dados, se algum dado é alterado a modificação será propagada e aplicada para outro Banco de Dados. Num segundo momento, dentro de uma transação separada (que poderá ocorrer em segundos, minutos, horas ou até dias depois, ficando as cópias temporariamente fora de sincronia), a sincronização ocorrerá e os dados irão para todos os locais especificados.

Ambos os tipos de replicações podem ser utilizados. A replicação síncrona será utilizada quando cada transação for dada como concluída, ou seja, quando todos os nós confirmarem que a transação local foi bem sucedida. A replicação assíncrona será utilizada quando um nó executar a transação e enviar a confirmação ao solicitante para depois encaminhar a transação aos demais nós.

3.9 Tipos de Replicação no *Microsoft Sql Server* 2008

Segundo Lopes (2013), quando um servidor de Banco de Dados é habilitado para Replicação, ele solicitará a criação de um novo Banco de Dados (*Database)* de Sistema, que funcionará como um repositório de informações sobre as publicações criadas. Esse *Database* é chamado de Distribuidor.

A *Microsoft SQL Server* 2008 suporta as seguintes replicações: *Replication Snapshot*, *Transactional Replication* e *Merge Replication*.

- **Replicação *Snapshot***: Com esse tipo de replicação todo conjunto de dados é replicado a cada intervalo de tempo programado. Os servidores subscritores [servidores que irão replicar e fazer as alterações para os outros servidores que participam da replicação] são atualizados com uma cópia exata dos dados publicados pelo *Publisher*, não sendo considerada a distinção entre os dados que foram alterados ou não.

Neste tipo de replicação os servidores não precisam ficar monitorando as atualizações e não controlam as alterações, permanecendo uma grande quantidade de dados a ser transmitidos entre os servidores, assim congestionando as redes.

- ***Transactional Replication***: Nesse tipo de replicação somente as alterações efetuadas no publicador são replicadas para os subscritores. Qualquer alteração realizada em um ou mais artigos de uma publicação é imediatamente capturada pelo log [Registro] de transações, e replicadas para o distribuidor, sendo esta atualização quase que instantânea.

A principal vantagem é a diminuição no tempo de latência de atualização dos servidores, exigindo para isso servidores com mais recurso de memória para processamento de agente de controle interno das atualizações.

- ***Merge Replication***: Sua principal característica é que as alterações podem ser feitas em qualquer uma das réplicas dos dados. As alterações feitas em uma das réplicas serão repassadas para as demais, convergindo apenas para uma base de dados.

O presente capítulo tratou sobre formas de Redundância e suas características, que são necessárias para a compreensão de como poderá ser realizada a implementação de *clusters*, conteúdo abordado no capítulo 4.

4 CARACTERIZAÇÃO DOS *CLUSTERS*

Este capítulo tem por objetivo apresentar os principais conceitos sobre *Clusters* e as suas classificações.

4.1 *Clusters*

Cluster é um sistema que compreende dois ou mais computadores ou subsistemas (denominados nós), os quais trabalham em conjunto para executar alguma aplicação ou realizar tarefas específicas, de tal maneira que os usuários finais devem possuir a noção de um sistema único e homogêneo. A comunicação é feita através de *polling*, que são pedidos contínuos de informações, permitindo que um servidor saiba que existem outros nós do *cluster* em funcionamento (ROCHA, 2004).

Segundo Backer (2000) *apud* Rocha (2004), os *clusters* possuem alguns critérios para que sua estrutura tenha uma segurança, um padrão definido e uma finalidade. Para isso são feitos algumas exigências para o seu desenvolvimento e configuração, mantendo um só padrão entre os nós que compõem o *cluster*, o Sistema Operacional e a configuração de *hardware*.

- **Finalidade de Aplicação**: *Clusters* são usados para o processamento de alto desempenho ou alta disponibilidade das aplicações;
- **Utilização dos Nós**: Os *clusters* são dedicados quando o seu processamento está sendo usado apenas para executar alguns tipos de aplicações específicas. Já os não dedicados são usados quando as estações são aproveitadas para o uso individual e as aplicações paralelas só são processadas quando o sistema não está sendo usado pelos usuários;
- ***Hardware* dos Nós**: Um *hardware* dos nós pode ter sua estrutura formada por vários computadores (onde cada nó pode ser um computador pessoal), ou seja, várias estações de trabalho**;**
- **Sistema Operacional do Nó:** Existem diversos sistemas operacionais que podem ser usados para criação de um *cluster,* dentre eles os mais utilizados são o *Linux* e o *Windows Server*. O *Linux* possui vantagens quanto à estabilidade, flexibilidade e robustez, além do baixo custo. Já a administração e o gerenciamento do *Windows* são bastante intuitivos, além dos processos de instalação e configuração de sistema ser automatizado.

- **Configuração dos Nós**: Em *clusters* homogêneos, todos os nós do sistema possuem um perfil de arquiteturas parecidas e o Sistema Operacional é homogêneo. Os *clusters* heterogêneos utilizam computadores que possuem as arquiteturas diferentes e o Sistema Operacional também é diferente.

A utilização de *clusters* pode invocar características como Transparência, Escalabilidade, Confiabilidade, Gerenciamento e Manutenção.

- **Transparência**

O *cluster* deve apresentar-se como um sistema único, isto é, a integração do processamento distribuído deve ser clara e eficiente, independente de qual sistema está operando, para prover um determinado serviço (Alta Performance e/ou Alta Disponibilidade).

- **Escalabilidade**

A prática mais relevante dentro da escalabilidade é a adição e a remoção de nós dentro do *cluster* de forma transparente. Deve existir a possibilidade de acrescentar componentes no *cluster* sem interromper a disponibilidade dos serviços ativos, como exemplo: acrescentar nós, periféricos e efetuar interconexões de rede.

- **Confiabilidade**

É uma análise sobre as falhas que podem acontecer durante o seu ciclo de vida, não sendo medido somente por cálculo de falha, e sim pelo histórico das falhas durante o seu tempo de funcionamento.

Alguns fatores podem interferir na variação da confiabilidade de um equipamento como: temperatura ambiente, umidade e manutenções anteriores. O *cluster* deve ter capacidade de detectar falha interna ao grupo, assim como de tomar atitudes para que estas não comprometam os serviços oferecidos.

- **Gerenciamento e Manutenção**

Devem existir mecanismos que permitam um fácil gerenciamento, devido à complexidade de configuração e manutenção de *clusters*.

4.2 Classificação Geral de *Clusters*

Segundo Pitanga (2003), existem alguns tipos de *clusters* que são bastantes conhecidos, dentre eles estão o *Cluster Failover*, *Cluster* de Alta Disponibilidade, *Cluster* de Balanceamento de Carga e *Clusters* Homogêneos e Heterogêneos.

- ***Cluster Failover***

De acordo com a *Technet*, *Microsoft* (2013), o processo de *failover* é o modelo de recuperação em *cluster*. Este processo pode ser entendido como a troca, através de transações, para um nó alternativo ou um nó de *backup*, devido a uma situação simples de falha em um dos nós do *cluster*.

O processo de *failover* deve ser totalmente transparente e automático, sem a intervenção de um administrador de sistema computacional e principalmente sem a necessidade de reconexão manual do cliente da aplicação crítica.

Para algumas aplicações, não críticas, que podem suportar um tempo maior até a recuperação da falha, pode-se utilizar *failover* manual. Além do tempo entre a falha e a sua detecção, existe também o tempo entre a detecção e o restabelecimento da disponibilidade da aplicação e dos recursos envolvidos no sistema computacional.

Grandes Bancos de Dados podem exigir um considerável período de tempo até que ajustem suas tabelas. Durante este tempo a aplicação ainda não estará disponível.

Para realizar o *failover* de uma aplicação crítica é necessário que os nós do *cluster* envolvidos na utilização deste mecanismo possuam recursos equivalentes. Um recurso pode ser uma interface de comunicação de rede ou um disco, onde os dados existentes no disco são elementos necessários à prestação de um determinado serviço para aplicação crítica em questão.

Dependendo da natureza da aplicação e do sistema de computação, executar um *failover* significa interromper as transações em andamento perdendo-as, sendo necessário reiniciá-las após o *failover*. Em outros casos, significa apenas um retardo até que a aplicação e os serviços disponíveis pela aplicação estejam novamente disponíveis.

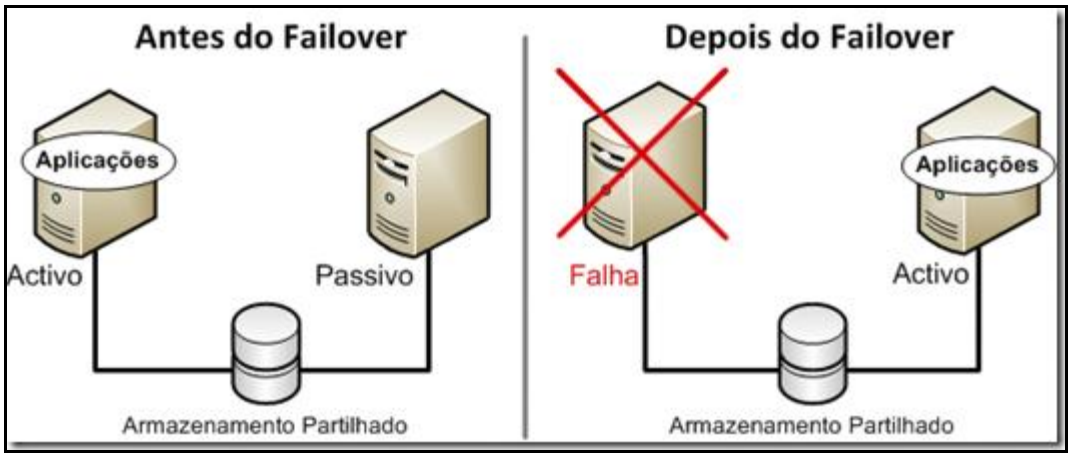
Conforme a *Technet*, *Microsoft* (2013), o processo reverso do *failover* é determinado *failback* e consiste basicamente em mover de volta a aplicação crítica e os clientes desta, para o nó/servidor original totalmente livre dos defeitos, erros ou falhas. Da mesma forma que o *failover*, o *failback* deve ser transparente e automático para os clientes.

O *failover* pode ser usado para outro importante propósito, a manutenção do nó/servidor original da aplicação crítica. Ações simples e rotineiras, como atualização do Sistema Operacional ou outros *softwares*, e principalmente adição ou manutenção de *hardwares* no sistema computacional são executadas sem a necessidade de parada de fornecimento dos serviços e recursos da aplicação crítica.

Em alguns casos, em função da possível nova interrupção na prestação dos serviços envolvidos no ambiente computacional, o processo *failback* pode ser não atraente.

Dependendo da aplicação utilizada, o método de *failover* pode ser eficiente ou não. Após resolver o problema da falha, o servidor disponibilizará o serviço e então se tem a opção de realizar o processo de *failback* (processo inverso).

Figura 2 - Antes e Depois do *Failover*.



Fonte: (http://redes-e-servidores.blogspot.com.br/2011/09/failover-clustering-iv.html).

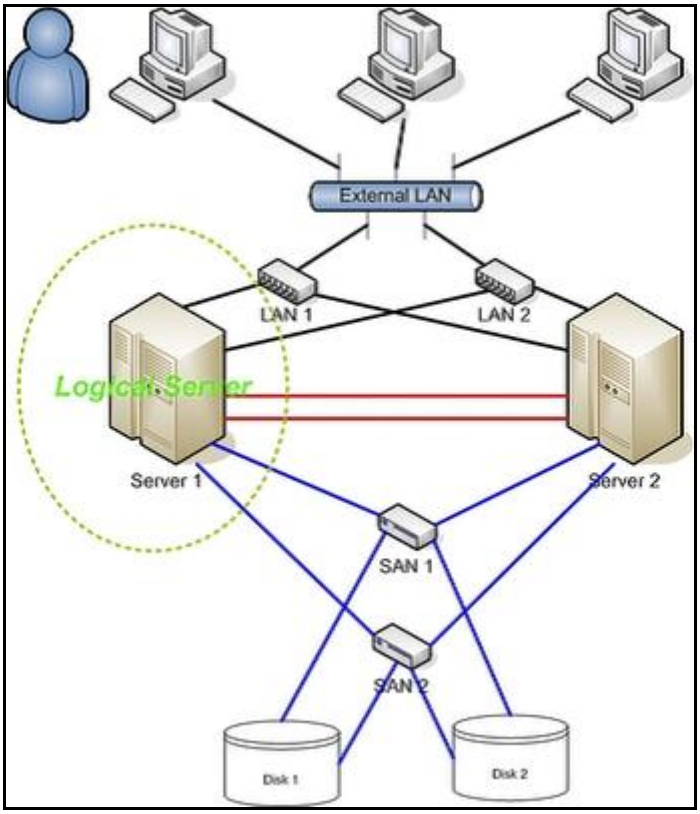
- ***Cluster* de Alta Disponibilidade**

Utilizados em tarefas que precisam da disponibilidade total do sistema, onde a falha seja insignificante, pois qualquer falha pode causar danos sérios ao usuário final.

Segundo Simões (2003), o *Cluster* de Alta Disponibilidade inclui duas ou mais máquinas que estão configuradas para que, se uma máquina (ou aplicação) apresentar problemas, a segunda máquina assume a carga de trabalho de ambas as máquinas.

Cada uma dessas máquinas é chamada de nó de alta disponibilidade do *cluster*. O *Cluster* de Alta Disponibilidade é tipicamente usado em um ambiente que deverá obrigatoriamente estar disponível, por exemplo, um sistema bancário para onde os clientes devem ligar continuamente. A figura 3 ilustra um *Cluster* de Alta Disponibilidade.

Figura 3 - *Cluster* de Alta Disponibilidade.



Fonte: (http://en.wikipedia.org/wiki/File:2nodeHAcluster.png).

Para criar um ambiente de Alta Disponibilidade é importante considerar itens como as configurações necessárias para obter Alta Disponibilidade, a Redundância de *Hardware* e nós principais e secundários, que seriam como espelhos do nó principal.

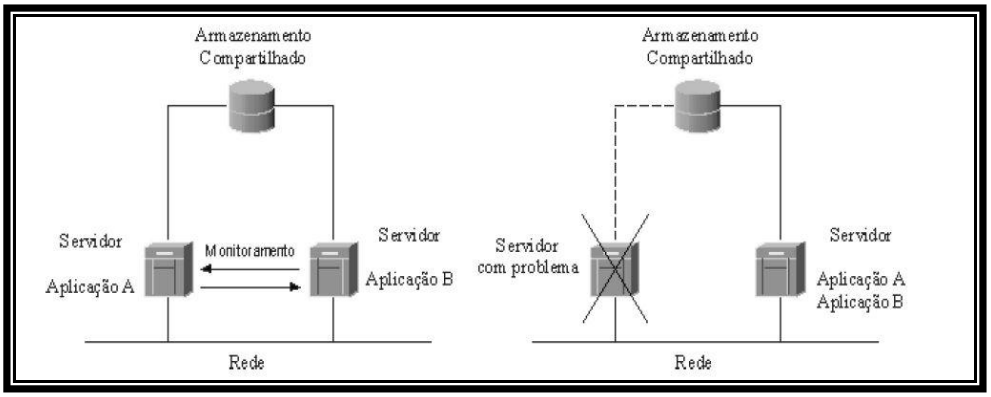
Os nós secundários (ou os chamados nós espelhos) devem possuir estado consistente com o nó principal (ou que é espelhado), tratando-se de dados e/ou recursos envolvidos no serviço que é oferecido (TORRES, 2003).

Para construir um *cluster*, não se deve somente pensar em um monte de máquinas conectadas entre si, e sim em sua estrutura interna, no papel que essas máquinas deverão desempenhar.

Os *clusters* de Alta Disponibilidade dividem-se em dois tipos: simétricos e assimétricos. Nos *clusters* simétricos, como ilustra a figura 4, os nós funcionam como se fossem uma máquina individual. Cada servidor é configurado para executar uma aplicação, não ocorrendo de um

servidor estar em *standby* [Em espera]. Acontecendo uma falha, o nó em operação assumirá a função do nó que falhou (FERREIRA e SANTOS, 2005, p.152).

Figura 4 - Tipo de *Cluster* Simétrico.

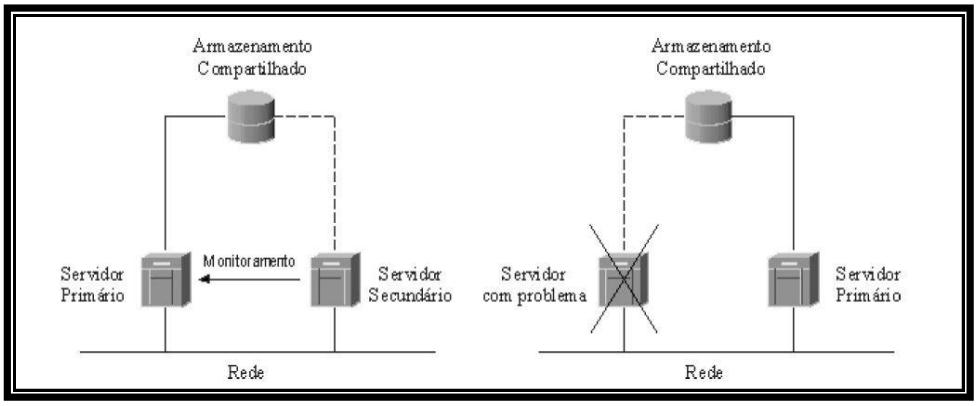


Fonte: (SIMÕES e TORRES, 2003).

Existem desvantagens no uso de *cluster* simétrico, como a dificuldade de gerenciamento e segurança do *cluster*. A distribuição dos trabalhos pode ser um problema, tornando mais difícil encontrar um desempenho ideal.

A arquitetura do *cluster* assimétrico, mostrado na figura 5, é mais comum, as aplicações são executadas no servidor principal, existindo um servidor redundante, caso haja falha, iniciando o processo de *failover*.

Figura 5 - Tipo de *Cluster* Assimétrico.



Fonte: (SIMÕES e TORRES, 2003).

A principal desvantagem dessa arquitetura vem das limitações estabelecidas na performance do nó principal. O servidor principal deve ter alto desempenho, pois suas limitações vão surgindo com o aumento do *cluster*. A maneira de se resolver esse problema é adicionar servidores dentro do *cluster*.

- ***Cluster* de Balanceamento de Carga**

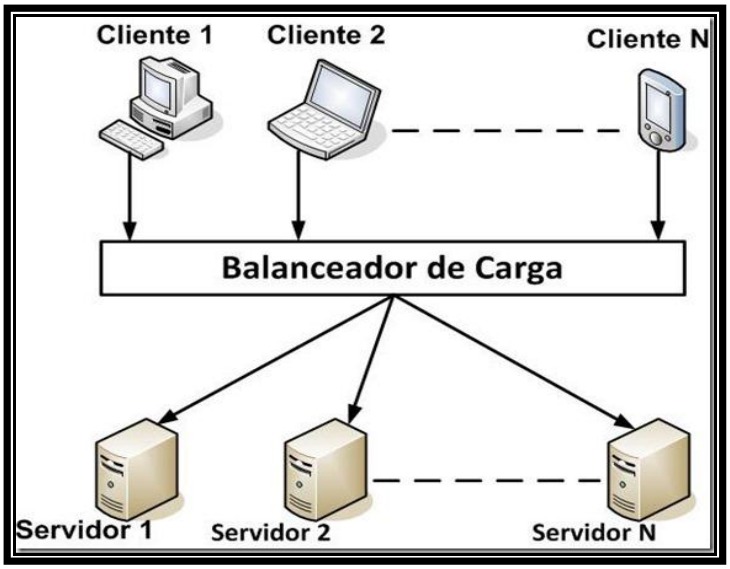
Conforme Oliveira (2005), este tipo de *cluster* tem a finalidade de distribuir atividades entre os nós. Através de um algoritmo de balanceamento de carga as tarefas são distribuídas entre os nós do sistema, fazendo com que ocorra uma considerável melhoria no desempenho.

É utilizado quando um serviço é requisitado por muitos usuários e existem diversas máquinas responsáveis por responder pelo mesmo serviço.

Para garantir que todas as máquinas recebam uma carga de trabalho equivalente, *daemons* (programas de monitoramento) estão ativos em cada servidor, monitorando a utilização da CPU e outros recursos. Com o resultado da monitoração, os servidores são capazes de realizar uma redistribuição da carga de tarefas, balanceando a utilização de todos os servidores.

Se um nó falhar, as requisições são redistribuídas entre os nós disponíveis. Um *cluster* com Balanceamento de Carga possui uma Alta Escalabilidade e pode ser facilmente expandido com a inclusão de novos nós. A Figura 6 mostra o *Cluster* de Balanceamento de Carga.

Figura 6 - *Cluster* de Balanceamento de Carga.



Fonte: (http://redes-e-servidores.blogspot.com.br/2011/03/balanceamento-de-carga-i.html).

- ***Clusters* Homogêneos e Heterogêneos**

Os *clusters* homogêneos possuem os nós compostos com a mesma arquitetura e mesmo Sistema Operacional logo entendem as mesmas instruções sem a necessidade de transformação de dados, a fim de possibilitar o processamento dos mesmos em diferentes processadores. Estão se tornando um padrão na área de *clusters* de alto desempenho por serem mais simples de operar e por não apresentarem problemas ligados à conversão de dados entre diferentes Sistemas Operacionais e arquiteturas (BACKER, 2000).

Os *clusters* heterogêneos são aqueles onde os nós que formam possuem processadores diferentes e, possivelmente, diferentes Sistemas Operacionais. Exigem a conversão de dados para que uma instrução possa processar em diferentes processadores.

De acordo com Backer (2000), quando são adicionados novos nós ou antigos nós são trocados ou reparados, o *cluster* pode tornar-se moderadamente heterogêneo. Um exemplo é quando componentes como ano de fabricação diferente, tais como a CPU, são acrescentados.

Quando um *cluster* é constituído por mistura de diferentes componentes de *hardware* e *software*, em longo prazo poderá ser potencialmente heterogêneo.

O referencial teórico abordado até o momento foi necessário para a compreensão de como poderá ser implementado uma solução de alta disponibilidade, utilizando *softwares* proprietários, conforme será abordado no próximo capítulo.

5 DESENVOLVIMENTO DA IMPLEMENTAÇÃO DO *CLUSTER FAILOVER*

Com base em conteúdos bibliográficos e experimentos já realizados com a utilização de máquinas virtuais, foi possível verificar excelentes resultados com a implementação do *Cluster Failover*, podendo estes oferecer melhorias para infraestruturas que utilizam servidores centralizados. Este capítulo tem por objetivo realizar a implementação detalhada sobre uma solução de Alta Disponibilidade aplicada em um ambiente corporativo.

5.1 Descrição da Implementação do *Cluster*

Este trabalho descreveu passo-a-passo as configurações e instalações das funções e dos recursos utilizados pelos servidores durante a implementação do *Cluster Failover*. Quando um dos servidores ficar indisponível, esta implementação reduzirá o tempo de indisponibilidade dos *hardwares* e *softwares*.

O estudo proposto não foi uma simulação, mas sim uma implementação em um ambiente real com um plano de contingência, para melhoria de desempenho das execuções das atividades de um sistema crítico.

A finalidade da implementação foi manter disponíveis para os usuários os recursos instalados nos servidores junto com suas aplicações, mesmo quando ocorrer algum tipo de interrupção dos sistemas por algum evento inesperado, como desastres naturais, falhas de *hardware* ou *software*.

De acordo com os objetivos propostos neste trabalho para a implementação do *cluster*, o propósito foi informar ao usuário de que não existe um servidor apenas, mas na verdade são dois servidores trabalhando como se fossem um só, assim aumentando o desempenho, a escalabilidade e a alta disponibilidade.

Inicialmente foram necessários mapear os componentes e os recursos que seriam utilizados na implementação, bem como analisar o domínio da disponibilidade, a fim de saber quais os dados e serviços que deveriam ficar disponíveis pela aplicação. Em seguida, colocou-se redundância suficiente nestes componentes para que situações de falha pudessem ser contornadas.

A estratégia para aplicar este procedimento foi o processo de *failover* dos recursos afetados, isto é, houve uma reconfiguração dinâmica no *cluster*, a fim de que ele continuasse provendo

seus serviços básicos e disponibilizando os dados das aplicações.

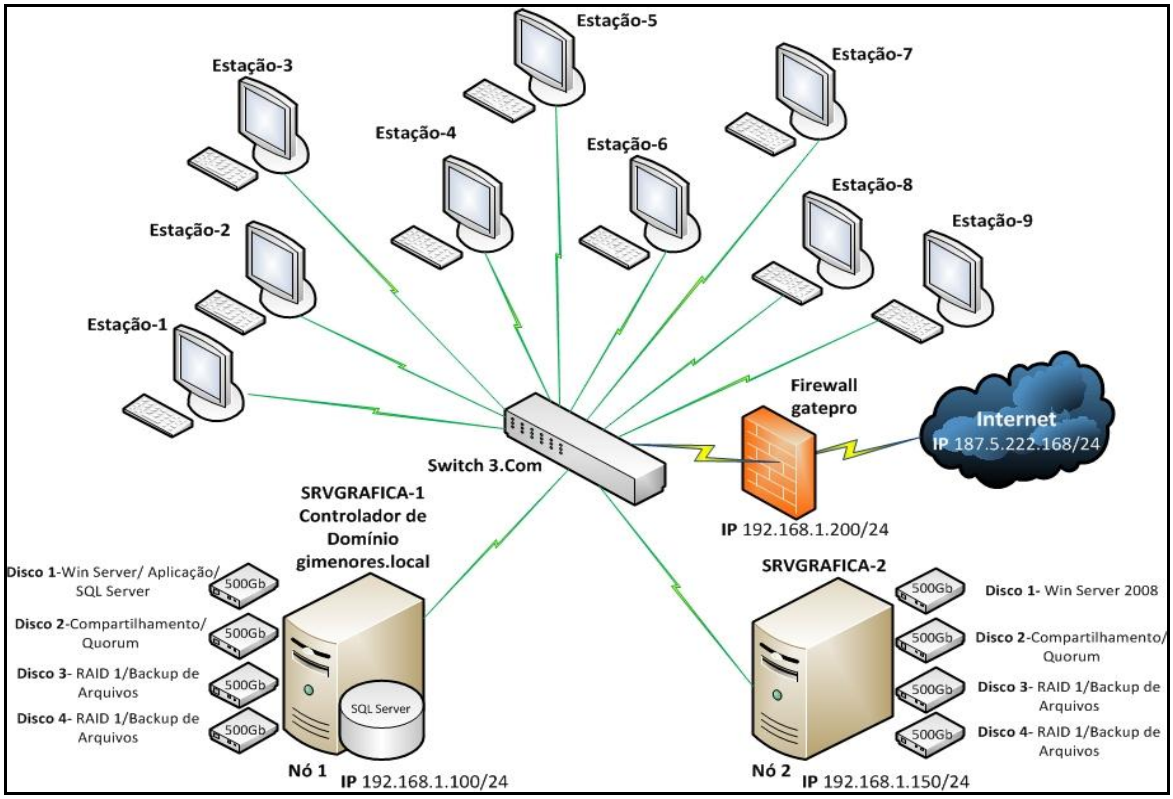
5.2 Descrição do Ambiente Físico

O ambiente onde estão armazenados os servidores mede aproximadamente 6m², com ambiente climatizado. Os servidores são acessados em um *rack* que está ligado em uma rede elétrica estabilizada com a utilização de *nobreaks* com potência de 7.500W e 3.500W com entrada de 220 V e saída de 110 V. cuja função é garantir o mínimo de indisponibilidade caso aconteça uma interrupção do fornecimento de energia.

O ambiente corporativo onde foi implementado o *Cluster Failover* conta com um parque de trinta computadores estações *desktop*.

Estes computadores estão interligados por uma rede *Gigabit Ethernet*, que oferece alta performance e baixa latência com uma melhor escalabilidade através do *switch* gerenciável de marca 3Com modelo 2952 com 48 portas, que utiliza cabeamento de rede modelo Categoria 6. Este *switch* faz a ligação e distribuição da *Internet* para os componentes ativos que compõem a estrutura da rede. De acordo com a figura 7.

Figura 7 - Parque de Computadores do Ambiente Corporativo



Fonte: Próprio Autor.

5.3 Descrição dos Servidores

Foram utilizados dois computadores de marca DELL com as seguintes configurações: Processadores *Intel Xeon* CPU 2.40GHz, 8Gb de memoria RAM, 4 (Quatro) HD de 500Gb de espaço de armazenamento.

Os servidores utilizados possuem uma redundância com fontes de alimentação *hot-plug* (tecnologia que permite conectar dispositivos com o computador ligado sem precisar reiniciar o mesmo).

A tabela 1, abaixo, mostra os *hardwares* redundantes e alguns *softwares* utilizados pelos servidores.

Tabela 1 - Configurações de *Hardwares* e *Softwares* Redundantes dos servidores.

SERVIDORES	**SRVGRAFICA-1**	**SRVGRAFICA-2**
SISTEMA OPERACIONAL	*Windows Server 2008 Enterprise R2 SP1*	*Windows Server 2008 Enterprise R2 SP1*
BANCO DE DADOS	*SQL Server 2008 Enterprise R2*	
SUPORTE A RAID	Controladora SAS 5/I integrada	Controladora SAS 5/I integrada
REDE	2x Broadcom NetXtreme 5721, *Single Port*, *Gigabit Ethernet*, PCI-E	2x Broadcom NetXtreme 5721, *Single Port*, *Gigabit Ethernet*, PCI-E
DESEMPENHO DOS DISCOS	7200RPM	7200RPM
FONTE DE ENERGIA	2x670W *Hot Plug*, fontes redundantes	2x670W *Hot Plug*, fontes redundantes

Fonte: Próprio autor.

O servidor SRVGRAFICA-1 é o controlador de domínio (gimenores.local) e faz o gerenciamento de todos usuários do AD (*Active Directory*), o qual é uma implementação de serviço de diretório que armazena informações sobre objetos em rede de computadores e disponibiliza essas informações a usuários e administradores de rede. Também são aplicadas

algumas regras de restrições aos usuários através da utilização de objetos de políticas de grupo (GPO).

Este servidor faz o gerenciamento e armazenamento dos dados e possui uma aplicação corporativa. Tal aplicação possui as funções de gerenciar informações de produção e operações financeiras, as quais estão instaladas no servidor SRVGRAFICA-1, que possui um servidor de Banco de Dados *SQL Server* 2008 com versão *Enterprise*.

Existe o segundo servidor, SRVGRAFICA-2, que possui as mesmas configurações de *hardware* e *software*, porém não existe nenhum Banco de Dados ou aplicação instalada, portanto não está realizando gerenciamento ou armazenamento dos dados do ambiente corporativo.

A função deste servidor é apenas gerenciar uma máquina industrial através de um aplicativo específico da própria máquina.

O objetivo foi colocá-lo, como domínio filho, no mesmo domínio do servidor SRVGRAFICA-1. Este procedimento foi realizado da seguinte forma:

- Configurações avançadas do sistema;
- Propriedades do sistema;
- Nome do Computador;
- Alterar nome do computador ou alterar seu domínio.

A implementação do *Cluster Failover* do Sistema Operacional e Banco de Dados aplica as regras de replicações das informações contidas no servidor SRVGRAFICA-1. Assim garante-se a disponibilidade dos dados e aplicativos, assegurando uma redundância de *hardwares* e *softwares* quando houver falhas. Mas para que esta implementação ocorresse com sucesso, foi necessário instalar os mesmos aplicativos no servidor SRVGRAFICA-2.

5.4 Descrição da Rede Local

A largura de banda de *Internet* utilizada pela rede local é de 2 Mbps de velocidade, fornecida por meio de uma conexão ADSL interligada pelo modem de marca *D-Link* modelo 500B.

A estrutura de rede é baseada em TCP/IP, pois a mesma utiliza o protocolo padrão para redes IPv4, o que facilita a implantação do *cluster*.

Para gerenciar e aplicar algumas regras na estrutura de rede o ambiente corporativo conta com *firewall Gatepro*, que utiliza o Sistema Operacional *Linux Red Hat* de versão 5.1, cuja

função é desempenhar o papel de um *gateway*, o qual faz a segurança de conteúdos acessados pelos usuários, realizando o roteamento, serviços de VPN e controle de banda, também fazendo a prevenção contra ameaças de vírus.

Este equipamento possui duas interfaces de rede, uma para acesso externo e outra para a rede interna. O mesmo tem dois endereços IPs diferentes, um para a *Internet* e outro para a rede interna.

Os intervalos destes endereços IPs são adicionados nas interfaces de rede externa e local da seguinte forma:

- A interface de rede externa utiliza o IP 187.5.222.168/24.
- A interface de rede interna utiliza o IP 192.168.1.200/24.

Os endereços IPs dos servidores SRVGRAFICA-1 e SRVGRAFICA-2 serão visualizados na tabela 2, abaixo.

Tabela 2 - Configurações de endereços de rede dos servidores.

SERVIDORES	INTERFACE-1	INTERFACE-2
SRVGRAFICA-1	192.168.1.100/24	192.168.1.110/24
SRVGRAFICA-2	192.168.1.150/24	192.168.1.155/24

Fonte: Próprio autor.

Para implementar o *Cluster Failover* foi necessário configurar as duas interfaces de rede de cada um dos servidores. Os endereçamentos citados na tabela acima foram configurados antes do *cluster* ser construído, pois o endereço IP da segunda interface do servidor SRVGRAFICA-1 é o IP 192.168.1.110/24 denominado *host*/domínio, no qual já deve estar definido com o nome de domínio, recursos e funções do *failover*.

5.5 Estrutura da Base de Dados

O Banco de Dados utilizado no servidor SRVGRAFICA-1 é o *SQL Server* 2008, que está instalado em uma arquitetura centralizada, composta por uma coleção de 250 tabelas e contendo 2.893.285 registros, bem como outros objetos definidos, sendo: exibições, índices, procedimentos armazenados, funções definidas pelo usuário e gatilhos para dar suporte à execução de atividades com os dados.

Os dados armazenados no Banco de Dados são relacionados a um determinado assunto ou processo, tais como: informações de estoque para depósito de produção, operações financeiras, informações cadastrais de clientes, fornecedores e a produção de ordens de serviços.

Para manter a segurança e a integridade das informações é realizado diariamente o *backup* (Cópia de segurança) deste Banco de Dados. O principal objetivo deste processo é prover um ambiente adequado e eficiente para recuperar, armazenar e garantir a alta disponibilidade das informações armazenadas através da distribuição dos dados entre os servidores.

Propõem-se instalar e configurar o *failover* no *Windows* e *SQL Server* no SRVGRAFICA-1 e SRVGRAFICA2, pois esta implementação eliminará o problema do ponto único de falha, característica inerente aos Bancos de Dados com estrutura centralizada.

5.6 Funções e Recursos Instalados nos Servidores

O servidor SRVGRAFICA-1 foi utilizado como nó 1, que teve a função de controlador de domínio. Ficou responsável por replicar as informações do Banco de Dados e do Sistema Operacional para o servidor SRVGRAFICA-2, que fez o papel de nó 2 e ficou com a cópia de todas as informações inseridas, alteradas e excluídas do Banco de Dados do Servidor 1.

O tempo de replicação das informações entre os servidores foi configurado de forma instantânea através do assistente de replicação, este método foi adotado por que caso ocorresse uma falha de um dos servidores, o nó ativo assumiria o controle e continuaria disponível aos usuários sem que estes percebessem a ocorrência de alguma falha.

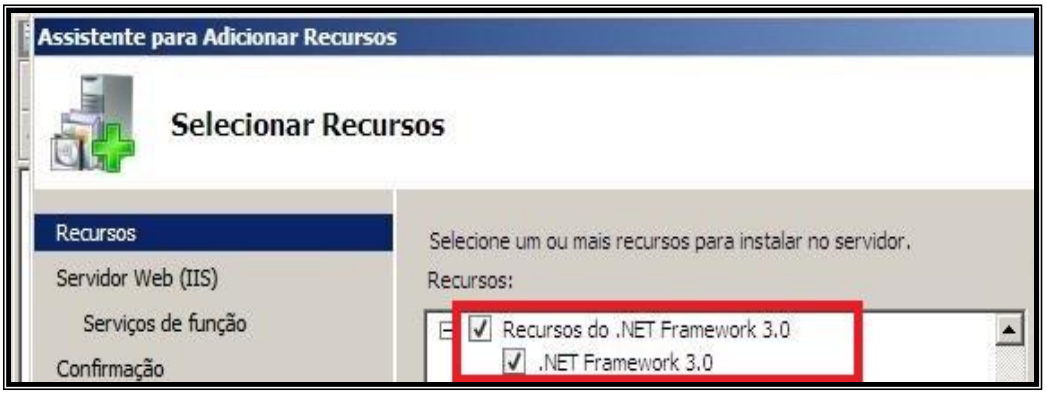
Quando ocorrer a indisponibilidade de um dos servidores por algum motivo relacionado ao Banco de Dados, as informações deste podem ficar desatualizadas devido ao sincronismo dos servidores. A replicação aplicada de forma assíncrona garantirá que o servidor que ficou indisponível (*off-line*) por algum evento de falha, irá receber as informações atualizadas após a sincronização com o servidor ativo. Isso ocorrerá após o problema ser solucionado, ficando novamente disponível (*on-line*).

Antes de começar a realizar a instalação dos recursos foi necessário desabilitar o *firewall* do *Windows* de cada um dos servidores através do painel de controle, pois o mesmo, habilitado causou problemas durante as instalações de alguns recursos nativos dos servidores.

O processo de instalação do *Cluster Failover* nos servidores foi feito através do gerenciador de servidores do *Windows Server*. Após ter acessado o gerenciador foi selecionado o .NET

Framework para a instalação do *Cluster Failover*, que é um recurso que possui as funções necessárias para funcionamento dos programas sob qualquer Sistema Operacional da plataforma *Windows*, conforme mostra a figura 8.

Figura 8 - Instalação do .NET *Framework*.

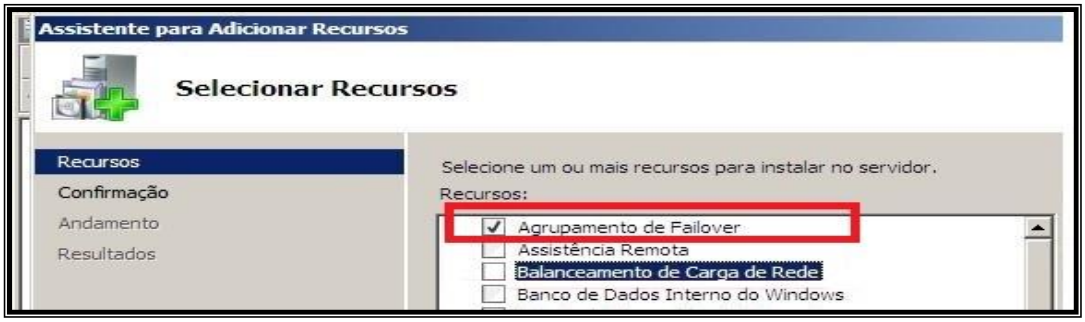


Fonte: Próprio Autor.

5.7 Implementação do *Cluster Failover* nos Servidores

O Agrupamento de *Failover* é usado para serviços de arquivos, impressões e Bancos de Dados, pois este grupo ficará com cópia das informações que participam do *cluster* nos servidores, quando houver falha de um nó por algum evento inesperado o servidor ativo irá fazer uma requisição ao grupo e começará a fornecer a disponibilidade das aplicações. A figura 9 mostra o período em que foi realizada a instalação do Agrupamento do *Failover*, cuja função foi permitir que vários servidores funcionassem em conjunto para fornecer disponibilidade dos recursos e aplicações.

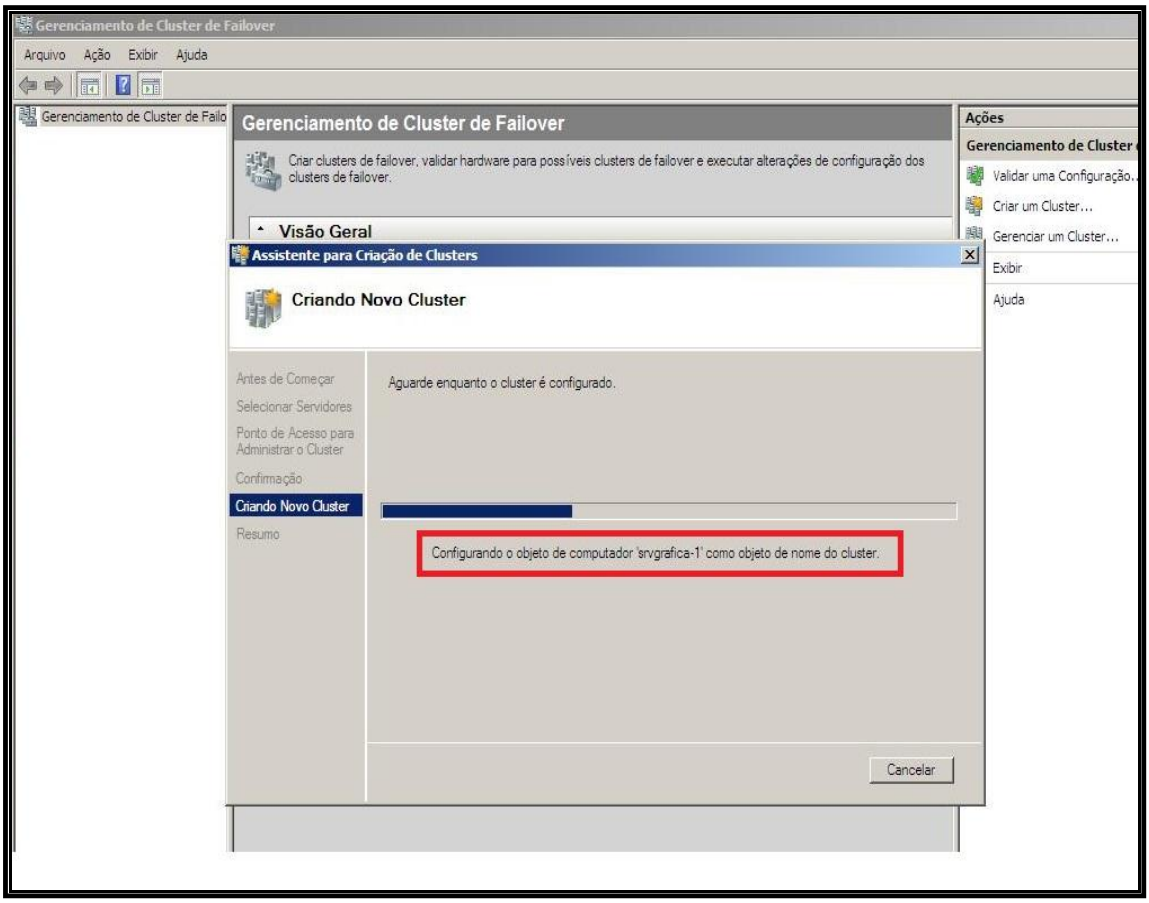
Figura 9 - Instalação do Agrupamento de *Failover*.



Fonte: Próprio Autor.

Esta instalação do Agrupamento de *Failover* é muito importante para dar continuidade às futuras instalações, pois é necessário para criar e configurar o *cluster*. Na figura 10 é mostrada a formação do novo *cluster* do SRVGRAFICA-1.

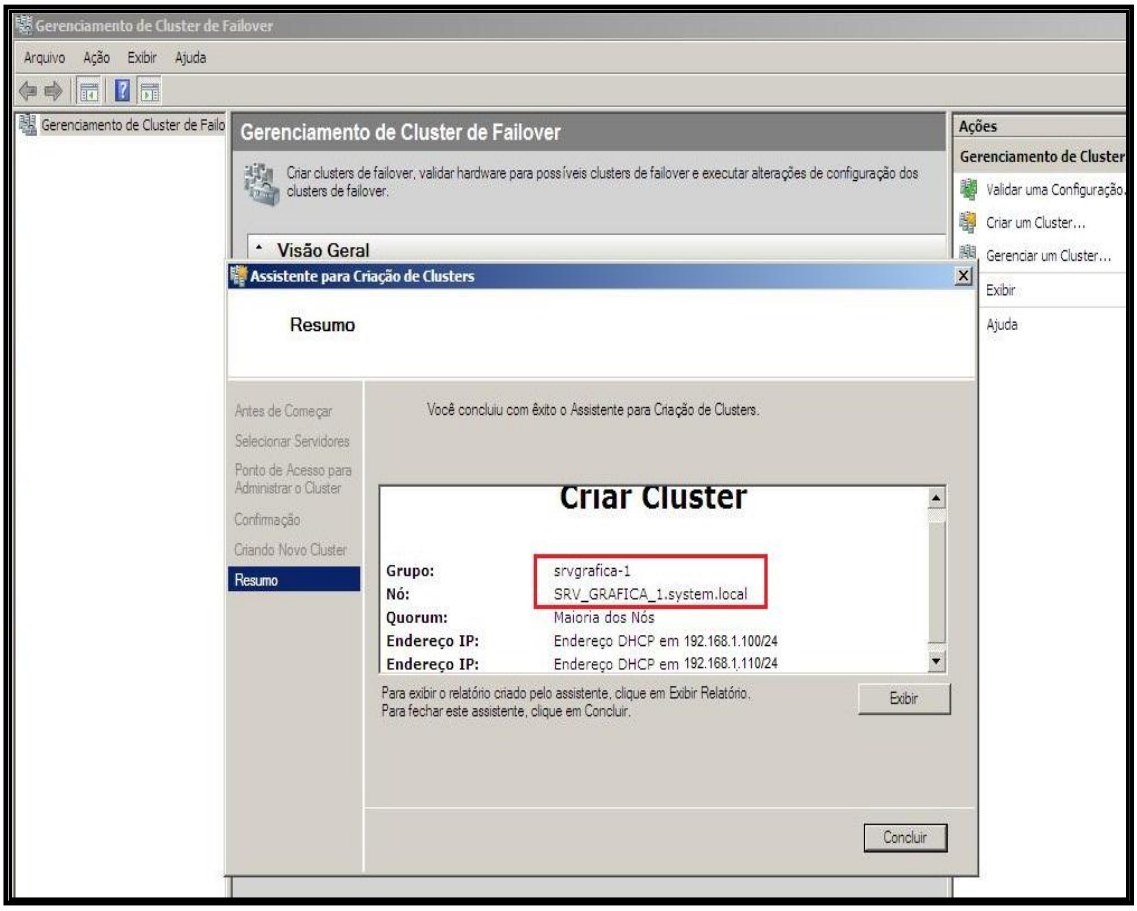
Figura 10 - Formação do novo *Cluster*.



Fonte: Próprio Autor.

Na figura 11 é visualizada a etapa de formação do *cluster* já concluída, onde é possível identificar através do gerenciador todas as configurações do servidor SRVGRAFICA-1.

Figura 11 - Formação do *Cluster* Concluída.



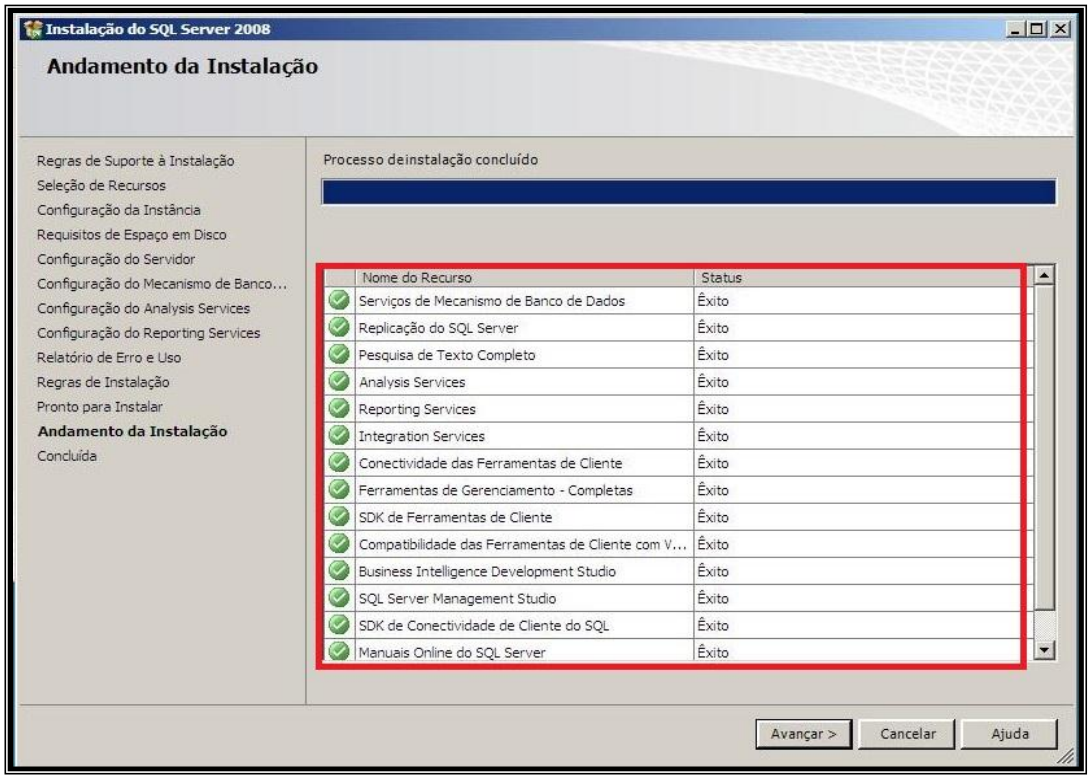
Fonte: Próprio Autor.

5.8 Implementação do *Cluster Failover* no *SQL Server*

Para manter as mesmas configurações de Banco de Dados do servidor SRVGRAFICA-1 foi necessário instalar o *SQL Server* 2008 *Enterprise* no servidor SRVGRAFICA-2, pois o mesmo não possuía nenhum tipo de Banco de Dados instalado.

A figura 12 mostra a instalação do *SQL Server* no servidor SRVGRAFICA-2 concluída, expondo várias etapas que foram processadas com êxito.

Figura 12 - Instalação do *SQL Server* Concluída.



Fonte: Próprio Autor.

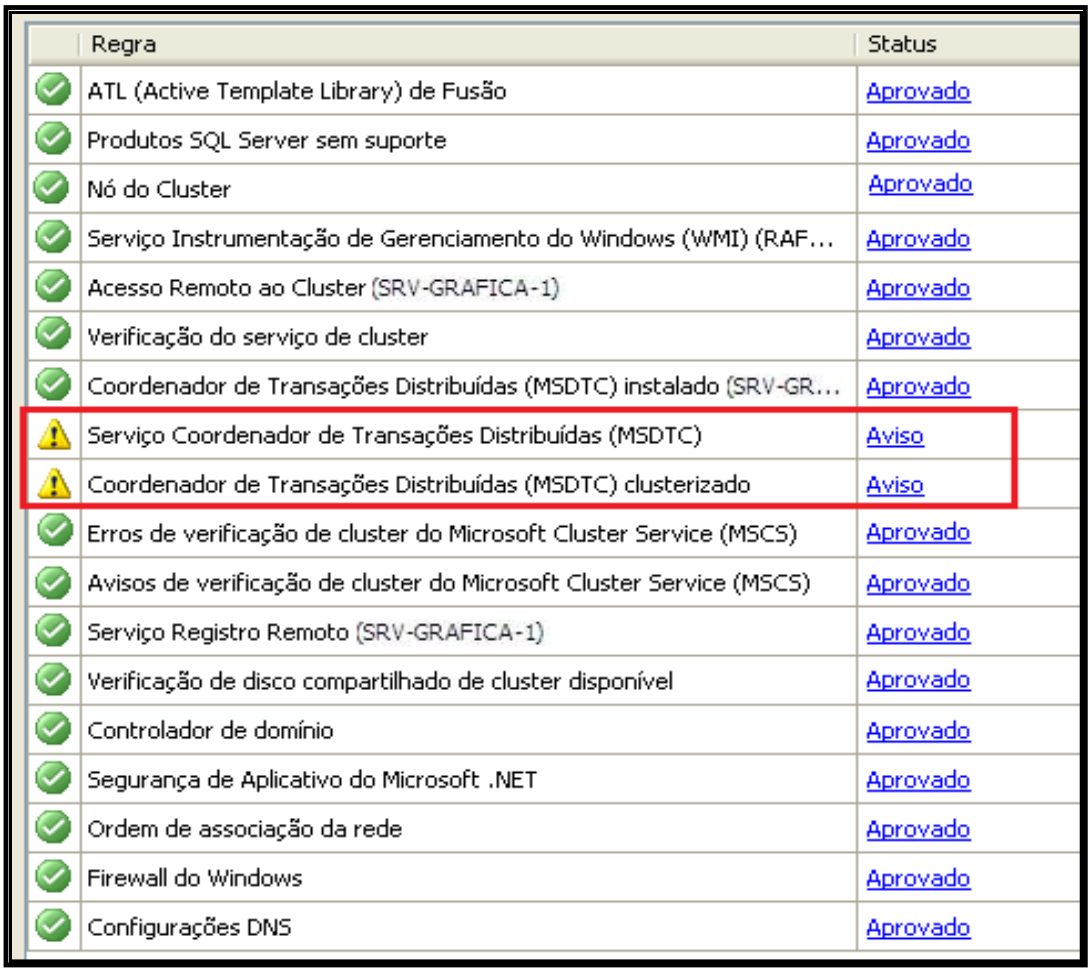
Após concluir a etapa anterior, foi realizada a instalação do *Cluster Failover* do *SQL Server* de uma forma idêntica nos servidores SRVGRAFICA-1 e SRVGRAFICA-2, conforme destacado na figura 13.

Figura 13 - Instalação de *Cluster Failover* do *SQL Server*.

Fonte: Próprio Autor.

Durante a nova instalação do *cluster* do *SQL Server* nos servidores, ocorreu um problema na análise dos processos executados, desta forma impedindo a continuação da implementação, conforme mostra a figura 14.

Figura 14 - Problemas durante a instalação do *Cluster Failover* do *SQL Server*.



Regra	Status
ATL (Active Template Library) de Fusão	Aprovado
Produtos SQL Server sem suporte	Aprovado
Nó do Cluster	Aprovado
Serviço Instrumentação de Gerenciamento do Windows (WMI) (RAF...	Aprovado
Acesso Remoto ao Cluster (SRV-GRAFICA-1)	Aprovado
Verificação do serviço de cluster	Aprovado
Coordenador de Transações Distribuídas (MSDTC) instalado (SRV-GR...	Aprovado
Serviço Coordenador de Transações Distribuídas (MSDTC)	Aviso
Coordenador de Transações Distribuídas (MSDTC) clusterizado	Aviso
Erros de verificação de cluster do Microsoft Cluster Service (MSCS)	Aprovado
Avisos de verificação de cluster do Microsoft Cluster Service (MSCS)	Aprovado
Serviço Registro Remoto (SRV-GRAFICA-1)	Aprovado
Verificação de disco compartilhado de cluster disponível	Aprovado
Controlador de domínio	Aprovado
Segurança de Aplicativo do Microsoft .NET	Aprovado
Ordem de associação da rede	Aprovado
Firewall do Windows	Aprovado
Configurações DNS	Aprovado

Fonte: Próprio Autor.

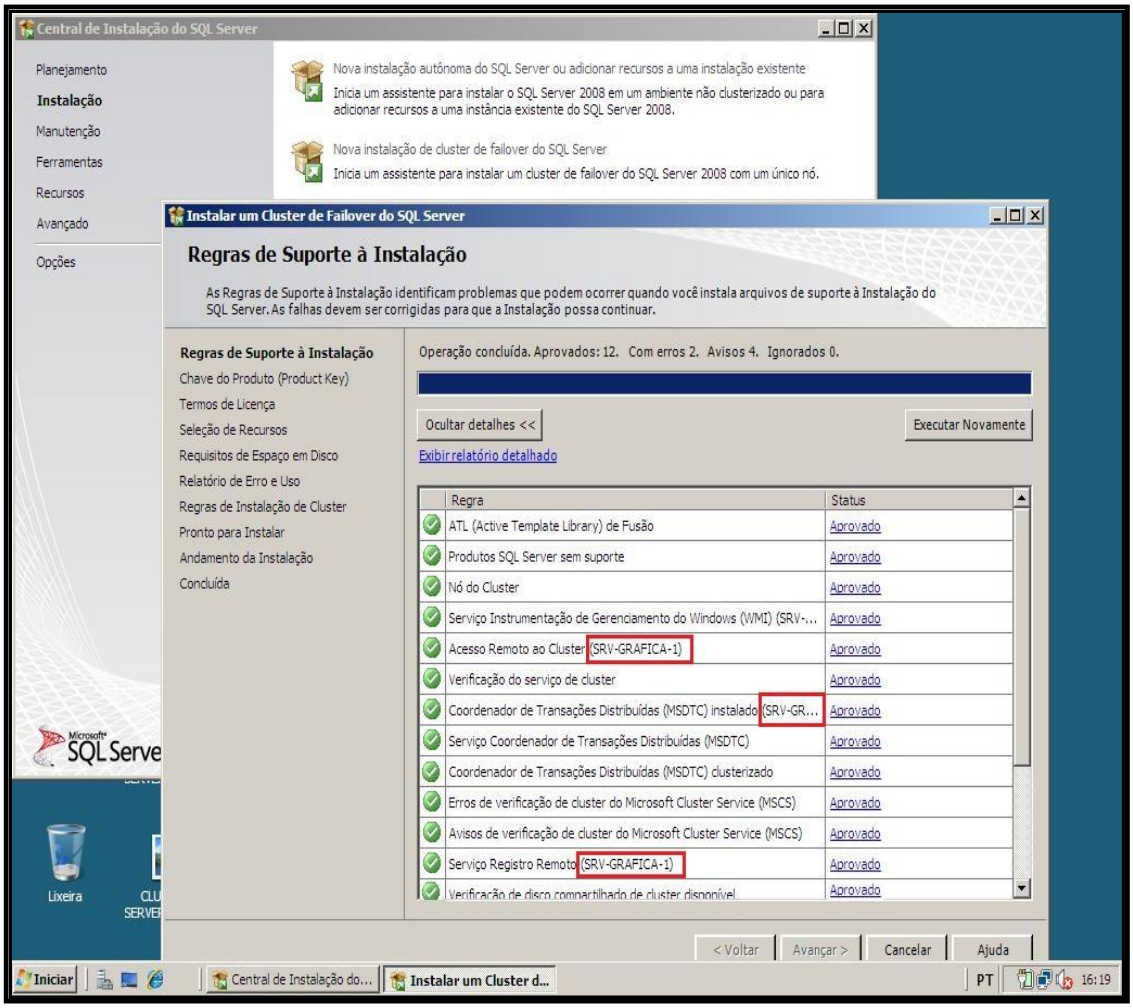
Este problema pode ter sido causado por motivos como falhas na placa iSCSI, falha em um nó da rede ou no Sistema Operacional. Este problema foi identificado através dos *logs* [Registros] de eventos de sistema dos servidores.

Neste caso este problema ocorreu porque o MSDTC (Coordenador de Transações Distribuídas da *Microsoft* do *SQL Server*) não estava em execução. Para ativar este processo novamente foi necessário realizar a instalação da atualização de um pacote chamado *hotfix*, que

tem a função de realizar vários reparos em *softwares* com o sistema operando, ou seja, estando com o sistema ligado.

Após esta correção foi reinicializado o servidor, executado novamente a instalação do *cluster do SQL Server*, onde foram analisadas e processadas as instalações de cada *cluster* e ao mesmo tempo realizadas as configurações, conforme mostra a figura 15.

Figura 15 - Análise dos Processos do *Cluster*.



Fonte: Próprio Autor.

5.9 Configurando a Replicação do Banco de Dados

Após a realização de todos os processos de instalações e configurações do Banco de Dados e ter criado o *cluster* do *SQL Server*, foi definido o tipo de replicação a ser adotada para estes Bancos de Dados.

Para este caso foi optada a Replicação Assíncrona, que por função fará a replicação das 250 tabelas do Banco de Dados, e qualquer informação alterada nas tabelas serão propagadas e aplicadas para o outro servidor de Banco de Dados, mesmo se este ficar temporariamente fora de sincronismo. Mas quando a sincronização ocorrer com o servidor que ficou indisponível (*off-line)*, este enviará os dados para todos os locais especificados.

Depois de definir o tipo de Replicação que fora aplicada foi necessário configurar as publicações e assinaturas locais dos servidores de Banco de Dados SRVGRAFICA-1 e SRVGRAFICA-2.

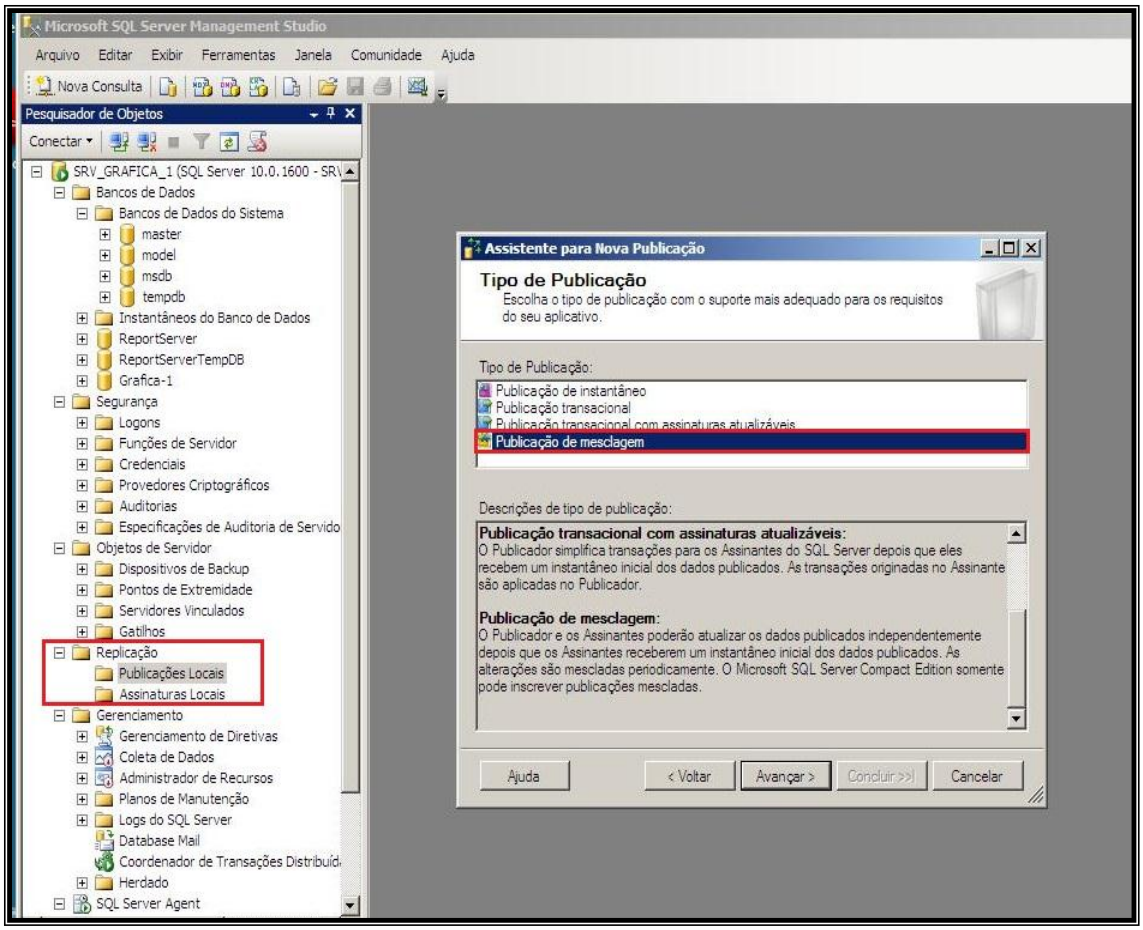
Quando o *SQL Server* foi instalado no servidor SRVGRAFICA-2, foram configurados os metadados que compõem as suas estruturas lógicas e de controle. Dentre esses metadados, as estruturas mais importantes são: *Master, Model, Tempdb e Msdb*. Estes metadados fazem parte do Sistema de Gerenciamento de Banco de Dados.

O Distribuidor foi criado no servidor de distribuição quando foi configurado o processo de Replicação, sendo utilizado para o armazenamento dos históricos das transações a serem replicadas.

Também foi necessário definir os Distribuidores e os Publicadores das transações nos Bancos de Dados. Neste caso, definiu-se que os dois servidores teriam a função de Distribuidor/Publicador. Um servidor Distribuidor/Publicador tem por função a distribuição e armazenagem dos metadados e dados de histórico para todos os tipos de replicação.

Já os servidores que tem apenas a função de publicadores incluem tarefas que permitem aos usuários adicionar conteúdo a um servidor de relatório. As figuras 16, 17, 18 e 19, abaixo, mostram as principais configurações realizadas.

Figura 16 - Instalação da Publicação de Mesclagem.



Fonte: Próprio Autor.

Neste tipo de publicação de mesclagem, a qual foi aplicada aos servidores, o Distribuidor/Publicador SRVGRAFICA-1, assim como o servidor SRVGRAFICA-2, atualizaram os dados publicados independentemente depois que receberam um instantâneo inicial dos dados publicados.

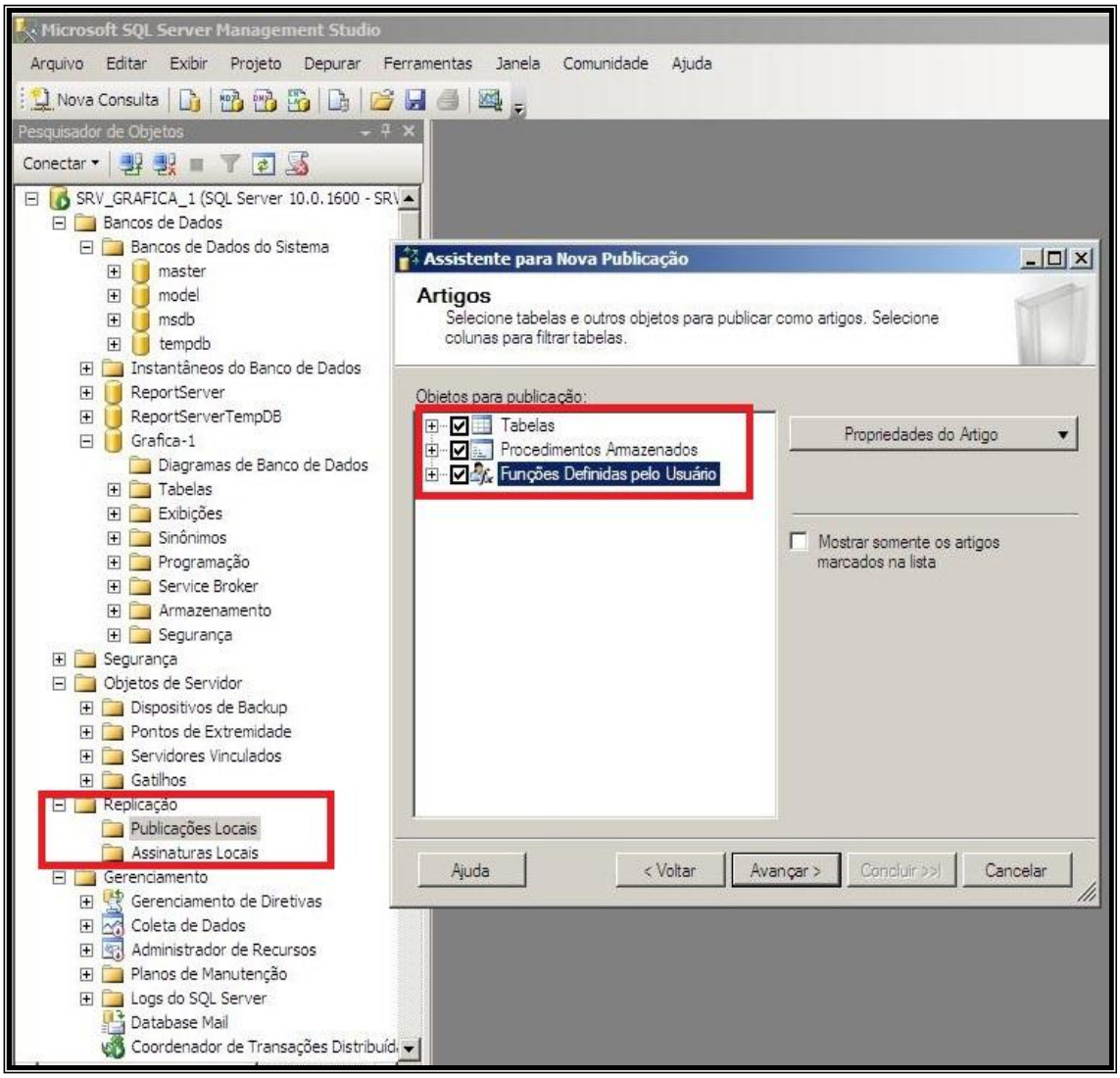
As alterações serão mescladas periodicamente conforme a disponibilidade de cada servidor. Este procedimento pode ser alterado ou definido na própria instalação da publicação. Neste caso, foram selecionadas todas as tabelas de cada Banco de Dados para que sejam replicadas de um servidor para o outro.

Durante a escolha dos itens que serão publicados e replicados pode-se especificar quais os objetos de esquema que devem ser copiados para os servidores, como integridade referencial

declarada. Podem ser incluídas as restrições de chave primária, restrições de referência, restrições exclusivas, índices, gatilhos DML de usuário, propriedades estendidas e agrupamentos.

As propriedades estendidas são replicadas apenas na sincronização inicial entre os servidores. A figura 17 mostra o processo de Publicação.

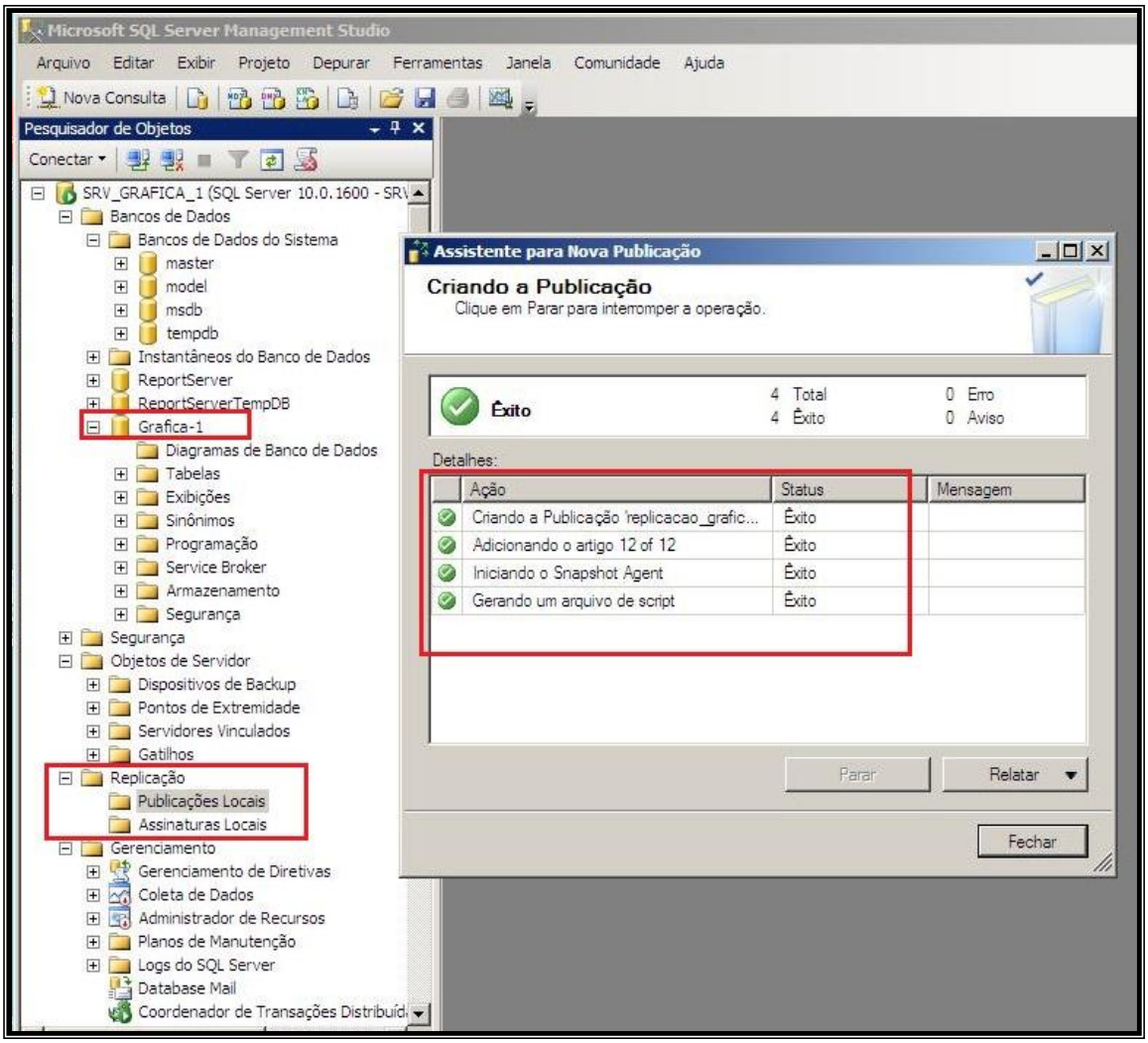
Figura 17 - Publicação das Tabelas do Banco de Dados.



Fonte: Próprio Autor.

Após definir os dados que serão publicados e replicados de um servidor para o outro, o servidor Distribuidor/Publicador faz uma análise dos itens selecionados, se não houver nenhum problema a publicação é concluída, conforme exemplifica a figura 18.

Figura 18 - Publicação dos Dados Concluída.



Fonte: Próprio Autor.

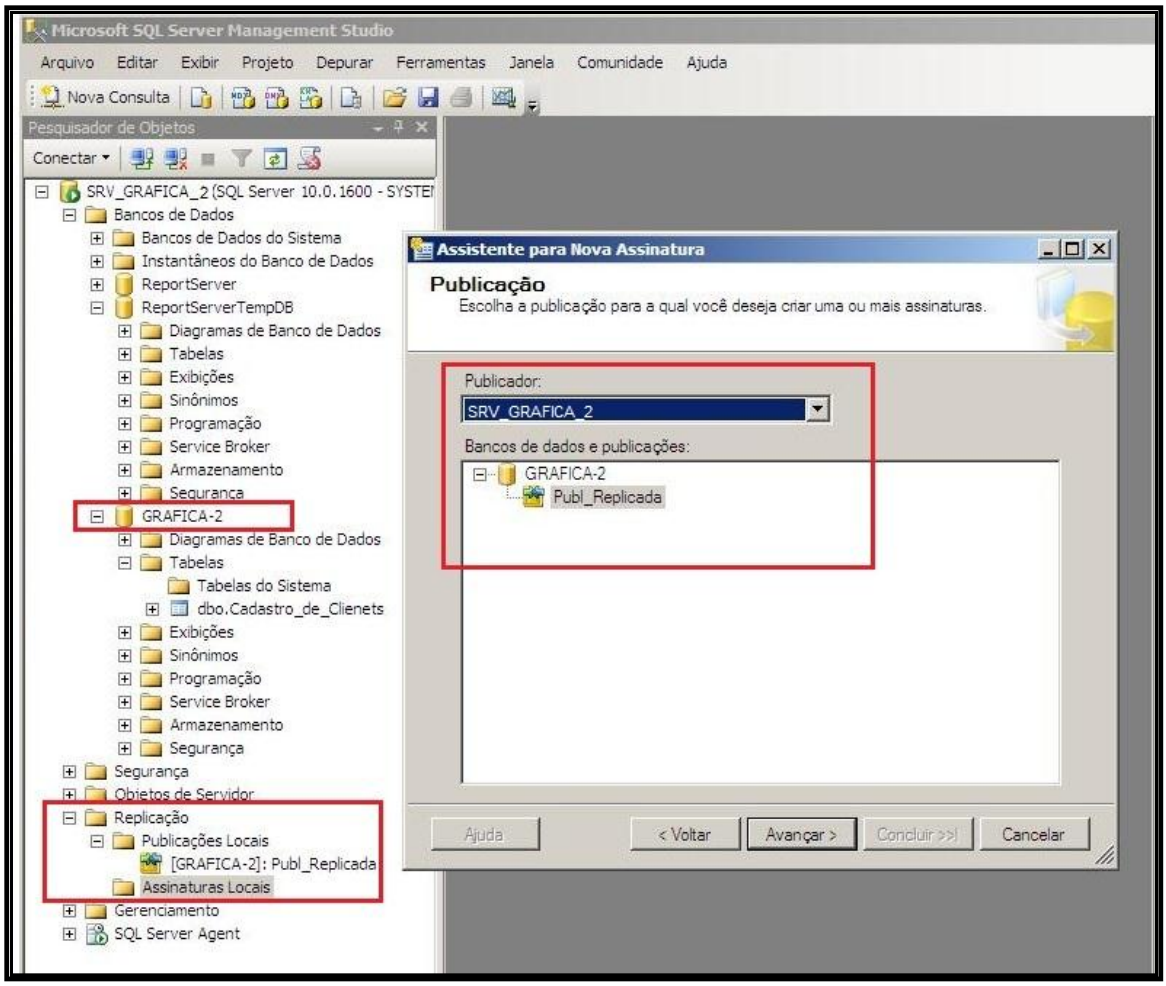
A assinatura criada no servidor SRVGRAFICA-2 é uma solicitação para obter uma cópia dos dados e objetos do Banco de Dados em uma publicação que foi realizada no servidor Distribuidor/Publicador SRVGRAFICA-1. Situação mostrada na figura 19.

Essa assinatura definiu qual publicação foi utilizada, onde e quando foi recebida, conforme os itens selecionados anteriormente na publicação.

Para planejar essa assinatura foi considerado onde o processamento de agente deverá acontecer. Para isso, o tipo de assinatura que controlou o local de execução do agente foi

executado nos dois servidores, SRVGRAFICA-1 e SRVGRAFICA-2. O Agente de Mesclagem ou de Distribuição foi executado no próprio Distribuidor. Conforme figura 19.

Figura 19 - Assinatura do Servidor SRVGRAFICA-2.



Fonte: Próprio Autor.

Este capítulo descreveu passo-a-passo as instalações e configurações para a implementação do *Cluster Failover*. Ações estas necessárias para fazer os experimentos e coletar os dados analisados, a fim de responder ao problema de pesquisa e objetivos propostos neste trabalho.

6 EXPERIMENTOS E DADOS COLETADOS

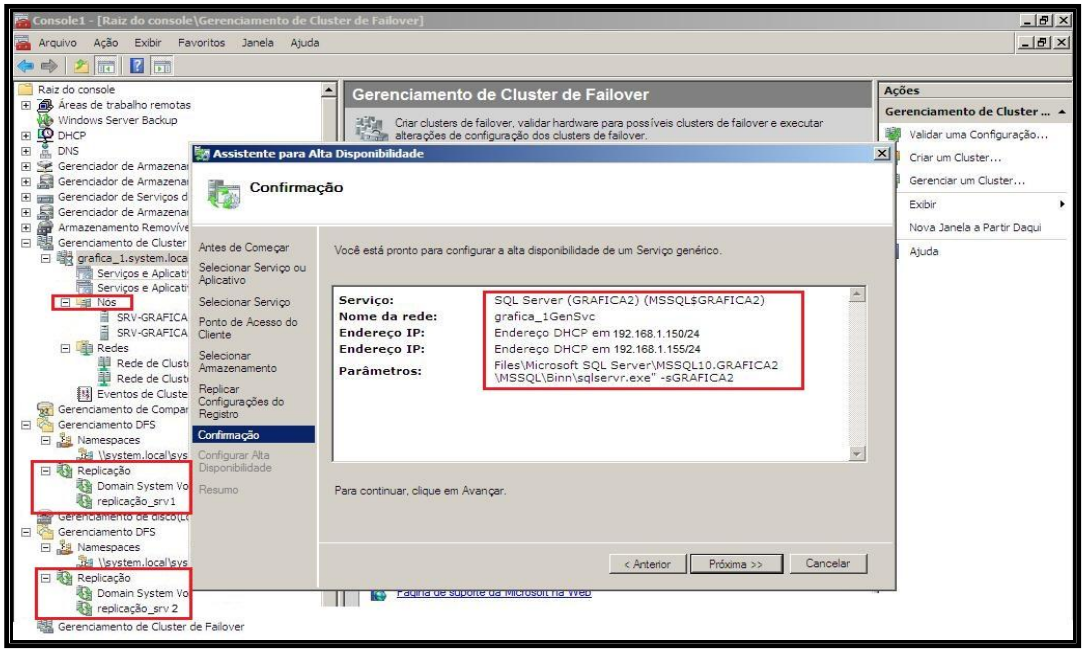
Dentro de todo o referencial teórico estudado não foi identificada uma regra ou pré-requisito que abordasse um padrão ou sequência de realização de testes com *Cluster Failover*. Portanto, para coletar os resultados obtidos com as implementações não foi adotado neste trabalho nenhum tipo de regra ou sequência de testes específicos a serem executados.

Para analisar o desempenho da implementação foram utilizadas ferramentas nativas que estavam pré-instaladas nos servidores e *softwares* disponibilizados pelo fabricante.

A primeira ferramenta utilizada para a realização de testes do *cluster* foi a console do *Windows MMC* (*Microsoft Management Console*), que é o gerenciador de Recursos de Servidores de Arquivos. Esta ferramenta é um recurso que possui vários serviços e que tem função de gerenciar os aplicativos instalados no sistema operacional.

A figura 20 identifica todas as implementações realizadas em cada servidor, neste caso SRVGRAFICA-1 e o SRVGRAFICA-2 também as configurações do *cluster* do Sistema Operacional, as configurações do *cluster* do *SQL Server* e as replicações definidas. Na mesma oportunidade, foi executado outro recurso, o Assistente de Alta Disponibilidade do *cluster*, onde foi possível visualizar as aplicações.

Figura 20 - Implementações e Confirmação da Alta Disponibilidade do *Cluster*.

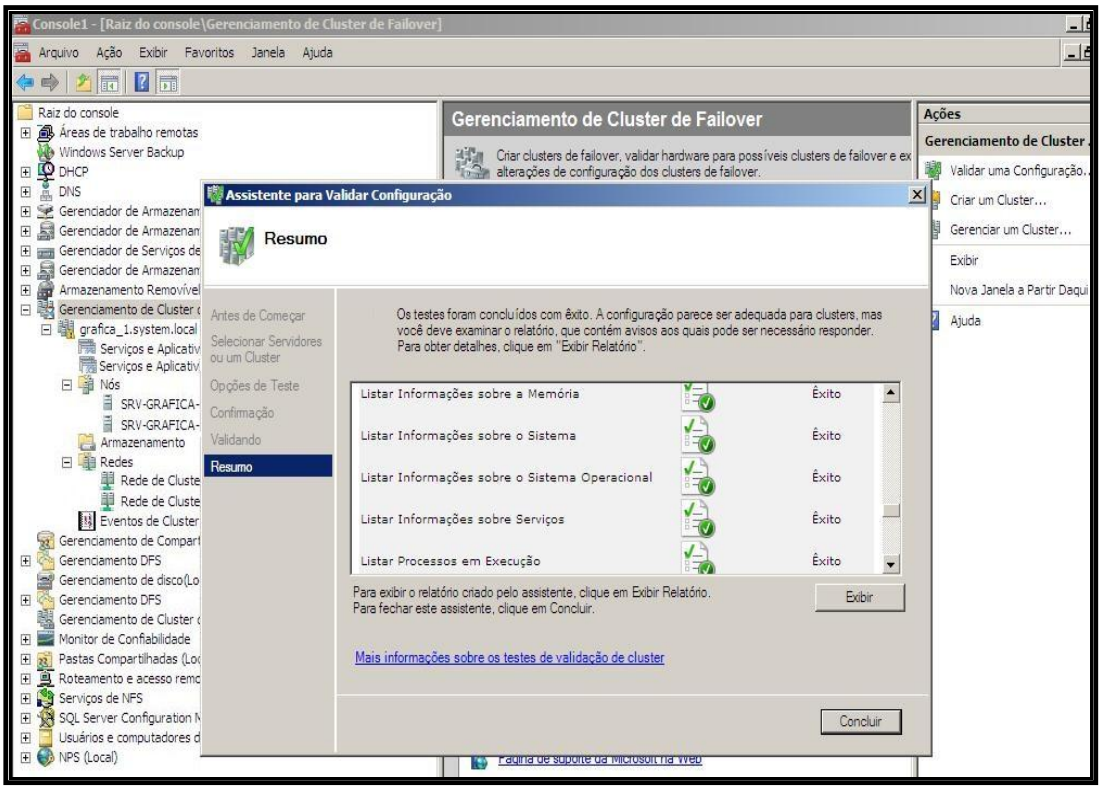


Fonte: Próprio Autor.

Após este procedimento, outro teste muito importante foi realizado. Foi executada a ferramenta de validação dos *clusters* através do gerenciador de *Cluster Failover*, que tem função de analisar todas as configurações de *hardwares* e *softwares* implementados em cada servidor.

Assim, foi possível identificar todos os processos executados e analisados pelo Assistente de Validação do *cluster*, onde não teve nenhuma restrição quanto às implementações configuradas anteriormente (Figura 21).

Figura 21 - Validação do *Cluster*.



Fonte: Próprio Autor.

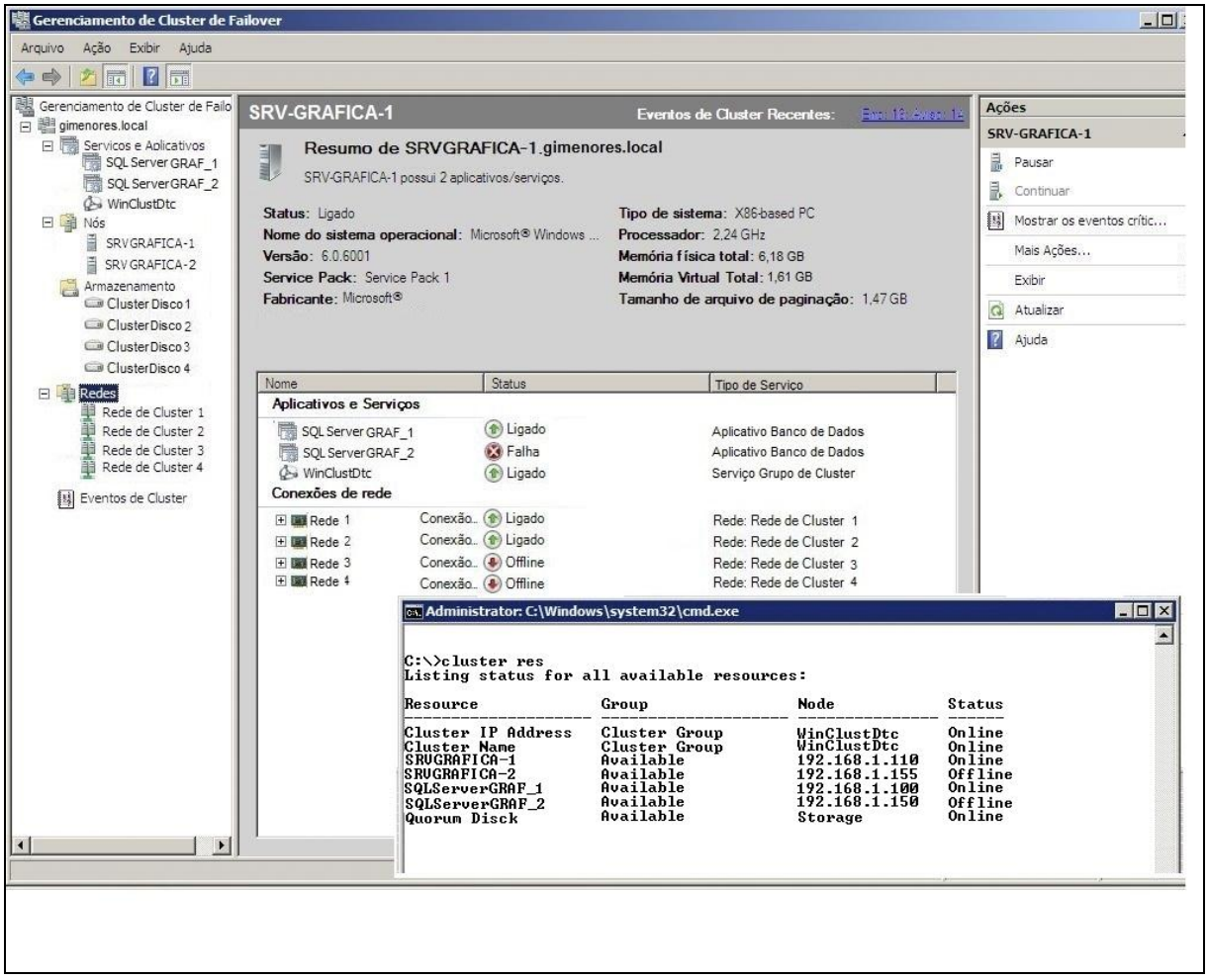
Depois de ter concluído os testes anteriores, foi realizada a desconexão de um dos servidores através do desligamento do cabo de rede da interface. Tal processo poderia ter sido realizado também através do desligamento da energia ou através da simulação de falha do Banco de Dados.

A desconexão do servidor foi executada no momento que acontecia uma atualização de um dos Bancos de Dados. Este procedimento foi realizado propositalmente para verificar e testar o *Cluster Failover* e também para verificar como iria se comportar a replicação após a falha.

O servidor que estava passando pela atualização no Banco de Dados era o SRVGRAFICA-2. Como este havia perdido a conexão anteriormente, o servidor SRVGRAFICA-1 assumiu o controle automaticamente de todos os serviços que estavam sendo fornecidos pelo servidor 2, porém a atualização do Banco de Dados não foi replicada devido à falha de rede acontecer no mesmo momento em que era realizada a atualização. Apesar disso, todas as aplicações ficaram disponíveis aos usuários sem que eles percebessem a falha de um dos servidores.

A figura 22 mostra através da ferramenta de gerenciamento do *cluster failover* do servidor SRVGRAFICA-1 que o servidor SRVGRAFICA-2 está indisponível (*off-line)* em sua interface de rede e com falha de replicação no Banco de Dados.

Figura 22 - Falha do Servidor SRVGRAFICA-2.

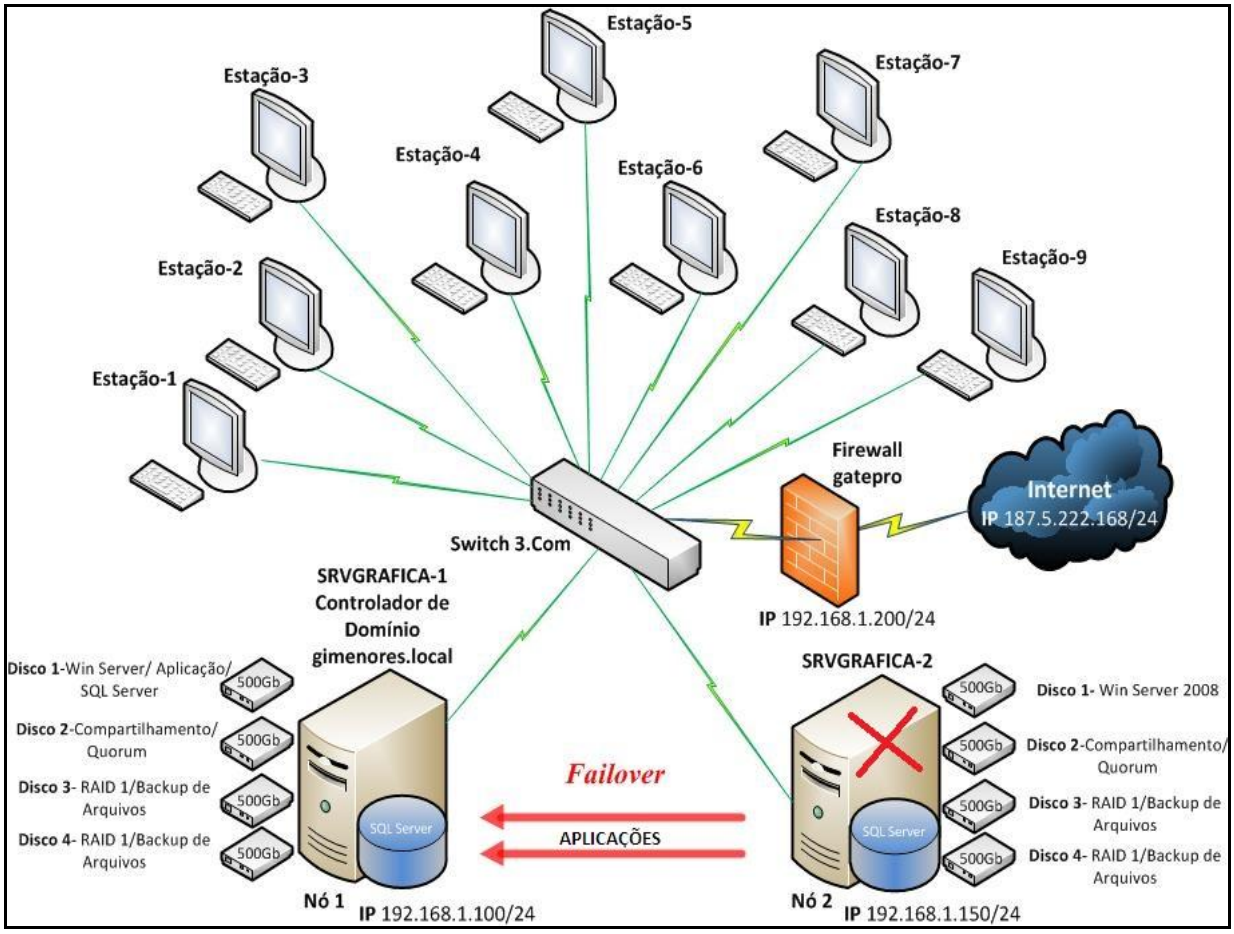


Fonte: Próprio Autor.

O fato do procedimento da replicação não acontecer no momento da falha não interferiu no funcionamento do servidor SRVGRAFICA-1, pois o mesmo já estava com as informações necessárias do Banco de Dados para disponibilização aos usuários.

O intervalo de desconexão do servidor SRVGRAFICA-2 foi de cinco minutos, tempo suficiente para analisar o comportamento do *cluster* e seus resultados. A figura 23 mostra como funciona o procedimento de *failover*.

Figura 23 - Momento em que acontece o *failover*.



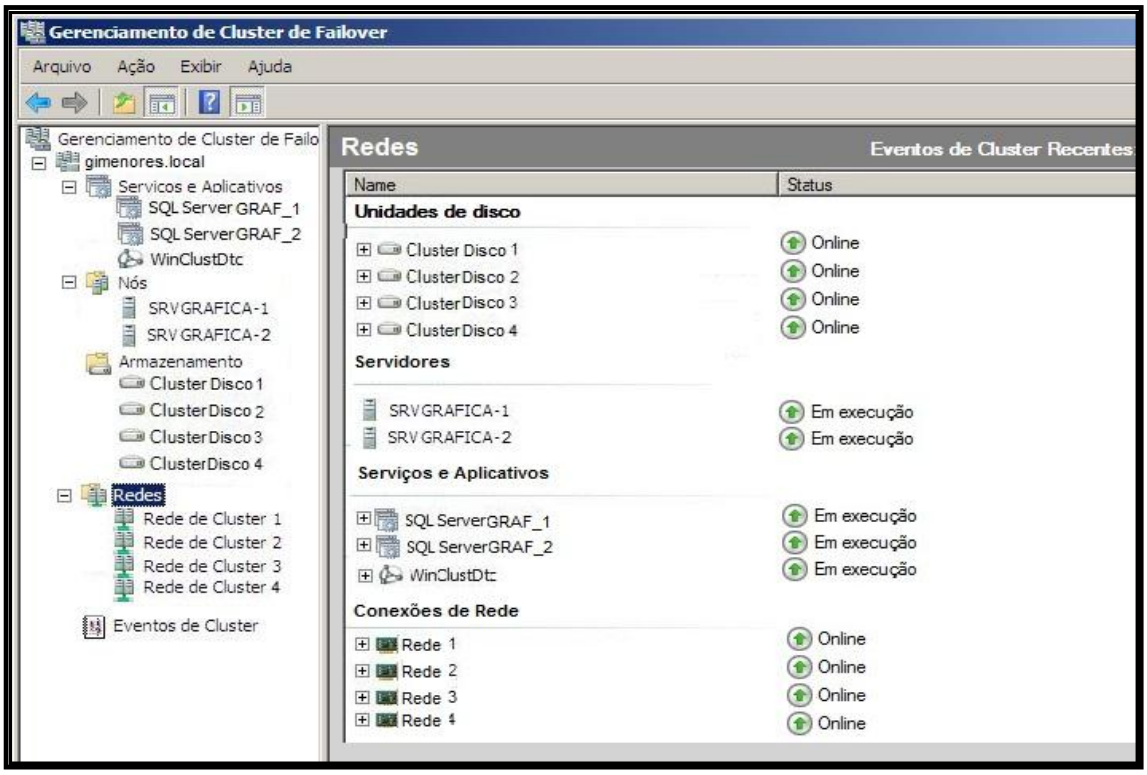
Fonte: Próprio Autor.

Após este tempo de desconexão do servidor, foi plugado novamente o cabo de rede em sua interface, onde foi possível analisar através da ferramenta de gerenciamento de *cluster* do servidor que todos os processos do SRVGRAFICA-2 haviam se restabelecido.

A atualização que estava acontecendo no momento da desconexão do servidor

SRVGRAFICA-2 foi transmitida após o mesmo ficar sincronizado de forma ativa (*on-line)* com o servidor SRVGRAFICA-1, conforme a figura 24.

Figura 24 - Todos os Serviços do *Cluster on-line* após os testes.



Fonte: Próprio Autor.

Após estabelecer o sincronismo e as atualizações dos servidores, foi realizado o *download* de três pacotes: *SqlCmdLnUtils_x86.msi, SharedManagementObjects_x86.msi, PowerShellTools_x86.msi*[1].

Estes pacotes são ferramentas que usam *scripts* que auxiliam na técnica de *Benchmark*[2] (ato de executar um programa de computador, um conjunto de programas ou outras operações, a fim de avaliar o desempenho de um processo), que instalados em conjunto irão avaliar parcialmente desempenho dos Bancos de Dados.

[1] _____. Disponível em: < http://www.microsoft.com/en-us/download/details.aspx?id=30440>. Acesso em 15 de julho de 2013.

[2] _____. Disponível em:< http://pt.wikipedia.org/wiki/Benchmark_(computa%C3%A7%C3%)>. Acesso em 15 de julho de 2013.

Estes pacotes foram muito importantes para o entendimento de como os Bancos de Dados responderiam sob as variações de condições executadas.

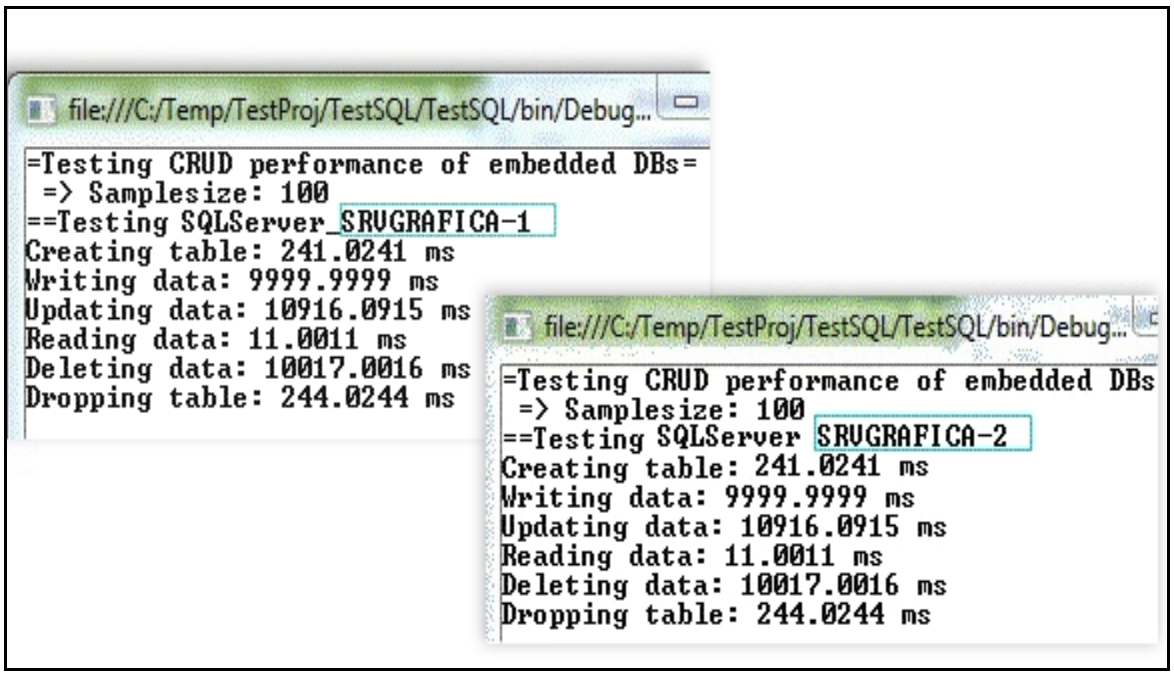
Com o uso destes pacotes pode-se criar cenários que avaliaram o tratamento de *deadlock* [bloqueio], desempenho dos aplicativos, diferentes métodos de carregar dados, características da taxa de transação, quantos usuários são adicionados e ainda o efeito na aplicação.

Os pacotes foram instalados e executados nos Banco de Dados SRVGRAFICA-1 e SRVGRAFICA-2, a fim de avaliar o desempenho de tempo de resposta de cada um destes servidores.

A figura abaixo (figura 25) mostra o tempo de cada processo executado em milissegundos. Estes pacotes usam *script*s que executam os comandos dentro de cada Banco de Dados: *Creating Table, Writing Data, Updating Data, Reading Data, Deleting Data e Dropping Table.*

A função de cada processo executado é criação de tabela, escrita dos dados, atualização, leitura, exclusão dos dados e exclusão de tabelas. Estes comandos mostraram o tempo de resposta de cada operação, desta forma podendo ser analisado o desempenho obtido por cada Banco de Dados.

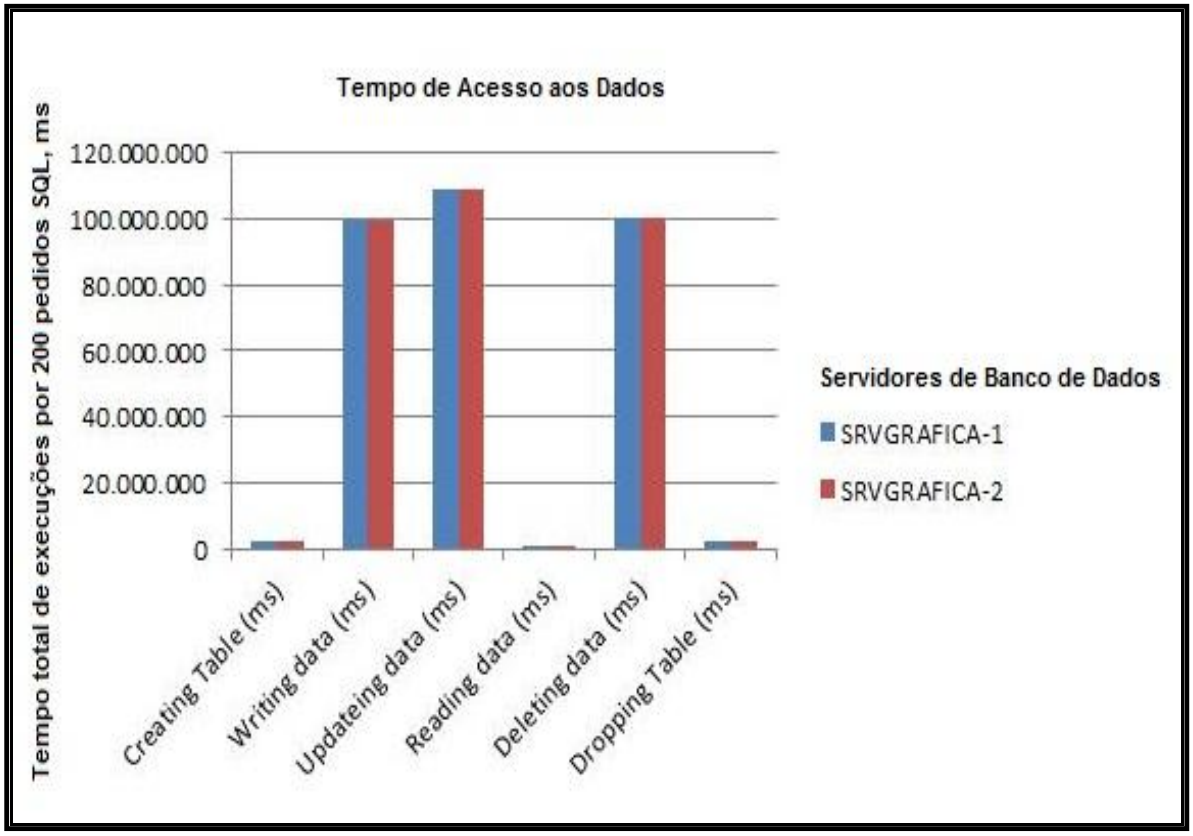
Figura 25 - Tempo de resposta de cada Servidor.



Fonte: Próprio Autor.

Para analisar o resultado da avaliação de desempenho dos processos executados em cada Banco de Dados, é gerado um gráfico com os dados obtidos pelos pacotes, os quais identificam que não existem diferenças entre os processos. Isso quer dizer que todas as operações executadas na figura acima (criar, escrever, atualizar, ler e excluir) estão no mesmo intervalo de tempo, possuindo o mesmo nível de desempenho de tempo de resposta, conforme mostra a figura 26.

Figura 26 - Análise de processos executados nos servidores.



Fonte: Próprio Autor.

O procedimento descrito acima mostrou que o *Cluster Failover* no *SQL Server* apresentou resultados esperados em termos de desempenho de tempo de resposta dos servidores.

Durante a obtenção dos resultados, conforme os testes apresentados anteriormente, foram constatadas as possibilidades de realização de ajustes nas configurações do *cluster*, como controlar as configurações, identificar os erros, as falhas ou os possíveis problemas antes que eles causem um período de inatividade.

7 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

Esta implementação foi realizada utilizando a plataforma *Windows Server* 2008 e *SQL Server 2008*, porém pode ser utilizada com a plataforma *Linux* e distribuições de Banco de Dados livres. A implementação do *Cluster Failover* em *softwares* proprietários pode possuir um custo elevado para as organizações, porém contar com um tipo de aplicação deste porte traz uma maior disponibilidade das aplicações e tornam os dados mais seguros.

Durante o desenvolvimento deste trabalho mostrou-se passos para a instalação e configuração de um *Cluster Failover*. Para avaliar os testes foram utilizadas ferramentas do próprio Sistema Operacional e outras que foram instaladas, conforme o capítulo anterior.

As ferramentas de verificação/Validação do *Cluster Failover* avaliaram os seguintes itens: Armazenamento: onde foi feita à validação dos discos do *cluster;* Configuração do Sistema: foi realizada a checagem dos *drives* do sistema, configurações do *Active Directory* e atualizações de *softwares;* Inventário: onde foram listadas as informações sobre *drives* do sistema, informações da BIOS e processos em execução; Rede: onde foi validada a comunicação de rede, nós do *cluster,* endereços IPs e configurações do *firewall* do *Windows*. Após os testes anteriores, foram utilizadas as ferramentas: *SqlCmdLnUtils_x86.msi; SharedManagementObjects_x86.msi e PowerShellTools_x86.msi*, instaladas para avaliar o desempenho do *Cluster Failover do SQL Server*. Executaram comandos de criação de tabelas, escrita dos dados, atualização, leitura, exclusão dos dados e exclusão de tabelas.

Em seguida, foi gerado um gráfico com os dados obtidos para comparar o desempenho dos servidores. Constatou-se que os dois servidores de Banco de Dados possuem o mesmo desempenho. Analisou-se que foram atingidos os objetivos propostos pelo trabalho, porém o objetivo de testar o desempenho dos sistemas foi atingido parcialmente, isso porque existem mais testes a ser executados como teste de carga e teste de *stress*. Os servidores responderam aos testes aplicados, garantindo assim a disponibilidade dos dados entre os servidores e suas aplicações.

Como trabalhos futuros sugere-se um estudo mais aprofundando sobre o tema, expandindo assim a possibilidade para a execução de mais trabalhos científicos que necessitam de um ambiente desse porte para gerar dados significativos, como por exemplo, a utilização de mais testes para avaliar o desempenho dos sistemas. Pode-se utilizar a arquitetura desenvolvida neste estudo para incitar novas configurações para a utilização do *Cluster Failover*.

REFERÊNCIAS

ALECRIM, Emerson. ***Clusters*: Conceitos e Características**. Disponível em: <http://www.infowester.com/cluster.php>. Acesso em: 23 Jun. 2013.

ASWINI, A.A.; MANISANKAR, V.R.; JAGADEESAN, J. ***An Efficient Approach For Avoiding Down Time of Production Data Base Server In Active/Passive Windows Failover Clustering Environment.*** International Journal of Innovative Technology and Exploring Engineering (IJITEE), V.1, N.3, 2012 .

BACKER, M. ***Cluster Computing White Paper***, Lançamento Final, Versão 2.0, Universidade de Portsmouth, Reino Unido, 2000.

DATE, C. J. **Arquitetura a Sistemas de Bancos de Dados**. p8. ed. Rio de Janeiro: Campus, 2004.

FERREIRA, Filipa Silva; SANTOS, Nélia C. Gaspar. **Alta Disponibilidade dividem-se em dois tipos: Simétricos e Assimétricos**, Instituto Leira, Politécnico de Leira, 2005. 152p.

FREITAS, F. (2003) e BRAGA (2001). **Replicação de Dados** . In: Renata Vieira; Fernando Osório. (Org.). Anais do XXIII Congresso da Sociedade Brasileira de Computação. Volume 8: Jornada de Mini-Cursos em Inteligência Artificial. Campinas: SBC, 2003,v. 8, p. 1-52.

LAUDON, K.C. LAUDON, J.P., 2004, **Sistemas de Informação Gerenciais**. 5. ed. São Paulo, Prentice-Hall Inc.

LOPES, Léo. **Replicação de Dados utilizando *SQL Server* 2008**. Disponível em: < http://www.blogdati.com.br/index.php/2011/07/replicacao-de-dados-utilizando-sql-server-2008/ >. Acesso em: 20 Abr. 2013.

MELO, 1997, **Banco de Dados**. r. 2. ed. Rio de Janeiro, Livraria e Editora Infobook S. A, 2004.

OLIVEIRA, Leonardo R. **Análise de Tolerância a Falhas e Balanceamento de Carga**, Bacharel em Informática, Trabalho de Conclusão de Curso – TCC, Salvador, UCSal, 2005.

OLIVEIRA, Daniel Cândido. **Desenvolvimento de Cluster de Alto Desempenho,** Dissertação (Conclusão da Disciplina – Pratica de Sistema de Informação I), Palmas, ULBRA, 2004.

PEREIRA FILHO, Nélio Alves. **Serviços de Pertinência para Clusters de Alta Disponibilidade,** Dissertação de Mestrado, São Paulo, 2004.

PITANGA, Marcos J. **Computação em *Cluster*.** 1. Ed. Brasport, 2003.

ROCHA, Lidiane Lins. ***Cluster*** – Visão Geral, Artigo, Brasília: UCB, 2004.

SILBERSCHATZ, A.; KORTH, H.F. ***Database System Concepts***. 2. ed. Singapore: Ed. McGraw Hill, 1991.

SIMÕES, Eduardo Issi; TORRES, Guilherme Mundim. ***Clusters* de Alta Disponibilidade em Servidores**. Projeto Final, Goiânia, UFG, 2003.

SHOMAN, M.L. ***Probabilistc models for Software Realiabibity Prediction***. Ed. Academic, 2002.

TECHNET, Microsoft**. *Failover and Failback*.** Disponível em: <http://technet.microsoft.com/en-us/library/cc757139.aspx>. Acesso em: 20 Abril, 2013.

TECHNET, Microsoft**. *Cluster Failover*.** Disponível em: <http://technet.microsoft.com/pt-br/library/cc731844%28v=WS.10%29.aspx>. Acesso em: 27 Abril, 2013.

WEBER, Taisy Silva. **Tolerância a Falhas: Conceitos e Exemplos**. Programa de Pós-Graduação, UFRGS, 2002.

WEBER, T.;Jansch-Pôrto, I.;Weber, **Fundamentos de tolerância a falhas**.Vitória: SBC/UFES, 1990. (apostila preparada para o IX JAI - Jornada de Atualização em Informática, no X Congresso da Sociedade Brasileira de Computação.

TANENBAUM, Andrew S. ***Computer Networks.*** 4 ed. Rio de Janeiro: Editora Campus, 2003.

VOLLERTT, João Rosaldo**. Confiabilidade e Falhas de Campo: Um Estudo de Caso para Melhoria da Confiabilidade de um Produto e do Reparo, Através de um Procedimento Sistemático de Coleta de Dados**. Monografia de Pós-Graduação, Santa Catarina, UFSC, 1996.

ZHANG, R., ABDELZAHER, T. F., and STANKOVIC, J. A. 2004. ***Efficient TCP connection failover in web server clusters***. In Proceedings of the IEEE InfoCom Conference. Vol. 2, 1219--1228.