Um Estudo Exploratório das Ferramentas de Código Aberto para a Replicação de Dados no PostgreSQL

Danilo S. de Carlo, Darlan F. S. Andrade, Rafael Liberato, André L. Schwerz

[1]Universidade Tecnológica Federal do Paraná (UTFPR)
Campus Campo Mourão - Departamento de Computação (DACOM)

{daniloc, darlanandrade}@alunos.utfpr.edu.br,

{liberato, andreluis}@utfpr.edu.br

Abstract. *Data distribution is indispensable to keep applications available and accessible when dealing with a high volume of requests. To meet this demand, PostgreSQL provides a wide variety of tools for data replication. However, understanding its characteristics and limitations is complex and time consuming. In this paper, we present an exploratory study about the features and limitations of the main open source tools for data replication in PostgreSQL, to reduce the complexity and time spent by database administrators in choosing the tool that meets their replication requirements.*

Resumo. *A distribuição de dados é indispensável para manter aplicações disponíveis e acessíveis ao lidar com um grade volume de requisições. Para atender essa demanda, o PostgreSQL fornece uma vasta variedade de ferramentas para replicação de dados. Entretanto, entender suas características e limitações é algo complexo e demanda tempo. Neste artigo, apresenta-se um estudo exploratório sobre os recursos e limitações das principais ferramentas de código aberto para a replicação de dados no PostgreSQL, buscando reduzir a complexidade e o tempo gasto por administradores de bases de dados na escolha da ferramenta que melhor preencha seus requisitos de replicação.*

1. Introdução

Um Banco de Dados Distribuídos (BDD) é uma coleção de vários bancos de dados interligados logicamente por meio de uma rede de computadores [Gupta et al. 2011]. Neste contexto, servidores de bancos de dados podem trabalhar juntos para que os dados de uma aplicação estejam sempre disponíveis e acessíveis mesmo em caso de falhas ou de um alto número de acessos simultâneos. Desta forma, em caso de falha em um dos servidores da rede, outros servidores podem suprir a demanda para que os dados mantenham-se disponíveis. Além disso, um número alto e distribuído de servidores e o balanceamento de cargas podem atender uma alta demanda de acessos simultâneos que a aplicação pode requerer. A distribuição dos dados entre os vários nós de um BDD é feita por meio da replicação ou fragmentação de dados [Heisler 2008].

Os Sistemas de Gerenciamento de Banco de Dados (SGBDs) relacionais mais conhecidos como o MariaDB [MariaDB 2019], SQL Server [Server 2019], PostgreSQL [Group 2019a], entre outros, possuem soluções para distribuir dados entre os servidores da rede. Bancos de dados NoSQL também possuem soluções para replicação

[Tauro et al. 2013], entretanto somente soluções relacionais são abordadas neste artigo. MariaDB provê alguns métodos de replicação como a replicação primário-réplica, replicação anel, replicação estrela e replicação de múltiplas fontes [MariaDB 2019]. SQL Server provê replicação transacional, replicação de mesclagem e replicação instantânea (*snapshot*) [Server 2019]. Em especial, foco deste estudo, o PostgreSQL possui diversas ferramentas e extensões para realizar o gerenciamento de BDD. Estas ferramentas combinam métodos de replicação e características de gerenciamento de dados distribuídos para atender demandas específicas.

Embora, o grande número de ferramentas do PostgreSQL disponíveis seja um fator positivo, compreender suas características e limitações não é uma tarefa trivial. A ampla variedade de alternativas para replicação, demanda conhecimento do projetista na escolha da solução mais satisfatória para o seu projeto. Por exemplo, em certos cenários pode ser mais importante manter os dados sempre consistentes no sistema do que prover uma alta disponibilidade ou vice-versa. Há meios de replicação que podem colocar a consistência dos dados em risco se não forem corretamente planejados.

Para auxiliar o projetista nesta escolha, apresentamos um estudo exploratório sobre os recursos, características e limitações das principais ferramentas de código aberto para a replicação de dados no PostgreSQL. O objetivo do estudo é levantar informações sobre as ferramentas para identificar quais são as mais eficazes em diferentes situações. Com isso, espera-se reduzir a complexidade e o tempo gasto por administradores de bases de dados na pesquisa e escolha da ferramenta que melhor preencha seus requisitos de replicação.

Este artigo está dividido como se segue. As principais características sobre replicação e bancos de dados distribuídos são descritas na Seção 2. A lista de ferramentas que apoiam a replicação no PostgreSQL é apresentada na Seção 3. Os principais resultados desta pesquisa estão na Seção 4 e os trabalhos relacionados estão na Seção 5. Por fim, as considerações finais são descritas na Seção 6.

2. Conceitos de replicação de banco de dados

A dificuldade fundamental em BDD é a sincronização dos dados entre os diferentes servidores. Qualquer escrita em um servidor precisa ser propagada para todos os outros servidores, para que futuras requisições de leituras nestes servidores retornem resultados consistentes. Normalmente, os servidores que recebem as alterações e replicam os dados são chamados de primários e os servidores que recebem esses dados são chamados de réplicas.

Uma replicação assíncrona é utilizada quando a propagação de operações de escrita ocorre em momentos oportunos (em um certo intervalo de tempo) para os outros servidores na rede [Heisler 2008]. Suas principais desvantagens estão na detecção tardia de possíveis erros e conflitos, e no tempo em que os servidores permanecem desatualizados até que as alterações sejam transmitidas. Por outro lado, na replicação síncrona, a propagação das operações de escrita para todos os servidores ocorre assim que elas chegam, e nenhuma transação é considerada finalizada até que todos os servidores tenham realizado aquela transação [Heisler 2008]. Suas principais desvantagens são uma queda significativa no desempenho e, na indisposição de algum servidor, podem ocorrer possíveis travamentos ou cancelamentos de operações de escrita. Nenhuma das soluções

elimina o impacto do problema de sincronização para todos os cenários. Desta forma, há diversas características e formas de replicação que devem ser observadas pelas ferramentas que a apoiam. As principais são descritas a seguir.

Modelo de replicação: os modelos básicos de replicação são o **primário-réplica** e o **multi-primário** [Mazilu et al. 2010]. Ambos os modelos podem ser síncronos ou assíncronos. No modelo primário-réplica, todas as modificações de escrita são realizadas no servidor primário e distribuídas para as réplicas, utilizando algum dos métodos de replicação. Tanto na solução assíncrona quanto na solução síncrona, os conflitos de sincronização não são comuns. No modelo multi-primário, cada servidor trabalha de forma independente, recebendo instruções de leitura e de escrita. Na solução assíncrona, o envio das modificações é realizado para todos os servidores periodicamente [Group 2019a]. Geralmente, o método assíncrono resulta em bastantes conflitos, que podem ser resolvidos manualmente ou por meio de regras de resolução. Por outro lado, no método síncrono, podem haver bloqueios prolongados e queda de desempenho. Há, ainda, outros modelos de replicação, como a replicação em anel ou a replicação em estrela, porém são variações derivadas da combinação dos modelos primário-réplica e multi-primário.

Balanceamento de carga: o balanceamento de carga pode ser divido em **balanceamento de leitura** e **balanceamento de escrita**. O balanceamento de leitura é o mais simples e pode ser executado desde que haja algum tipo de replicação. As requisições de leitura enviadas ao BDD são divididas entre os vários servidores da rede de forma que nenhum fique sobrecarregado. O balanceamento de escrita é mais complexo e dependente da forma que a replicação foi aplicada. BDD que possuem apenas um servidor primário são inaptos a executar o balanceamento de escrita, visto que é restrito ao servidor primário a execução das instruções de alteração. Sistemas que permitem a escrita em mais de um servidor podem executar o balanceamento de escrita, porém deve-se ficar atento aos possíveis conflitos e ao desempenho das sincronização dessas modificações.

Tolerância a falhas (*failover*): é inevitável que em um BDD servidores venham eventualmente a ser desligados da rede, seja devido a falhas ou a manutenções preventivas. Quando isso ocorrer, o sistema dever ser capaz de automaticamente redirecionar o fluxo dos dados ou promover servidores para manter o BDD ativo e disponível. Ferramentas de replicação devem automatizar o tratamento de falhas reduzindo intervenções humanas.

Replicação Parcial: replicar todos os dados de um BD nem sempre é algo desejável. A possibilidade de replicar tabelas específicas ou, até mesmo, partes de tabelas é um recurso útil em cenários de distribuição de dados.

Envio do *Write-Ahead Log (WAL)*: esse método de replicação, também conhecido por replicação física, consiste na replicação dos registros do WAL (*write-ahead log*), encaminhado do servidor primário para as diferentes réplicas [Group 2019a]. Pode ser realizado de forma síncrona ou assíncrona, não permitindo replicações parciais.

Replicação lógica: este método de replicação envia as alterações em um alto nível (*database/object level*) sendo mais seletivo que o WAL. Ele replica uma base de dados por vez, enviando apenas alterações confirmadas dos registros. Este método permite a replicação parcial da base de dados, e os dados podem fluir em múltiplas direções [Severalnines 2018].

Replicação baseada em *triggers*: neste método de replicação as modificações disparam gatilhos (*triggers*) que as transmitem assincronamente para os servidores réplicas [Group 2019a]. Além disso, como a atualização das réplicas é feita de forma assíncrona, um tratamento de falhas é necessário para não ocorrer perda de dados.

Fragmentação de dados: este método de replicação particiona a tabela em vários fragmentos (vertical ou horizontal), chamados *sets*, e os distribui entre os servidores. Cada servidor pode modificar apenas o seu *set* da tabela [Group 2019a]. É um método que divide os dados em quantidades menores, requer menor processamento das consultas e armazena tamanhos menores de índices.

Execução de instruções paralelas em múltiplos servidores (MPP): muitas das soluções existentes permitem a vários servidores tratar várias instruções SQL, mas esta permite vários servidores tratarem uma única instrução SQL simultaneamente, para a completar de forma rápida [Group 2019a]. É comumente utilizado com a fragmentação de dados, em que cada servidor executa a instrução para sua porção de dados, e retorna os resultados para um servidor central que combina os resultados e, então, retorna ao usuário [Group 2019a]. Este método diminui consideravelmente o tempo de resposta para quantidades massivas de dados.

Dado uma visão geral sobre os principais conceitos sobre replicação de dados, as diversas ferramentas de código aberto que apoiam a replicação no PostgreSQL serão abordadas a seguir.

3. Ferramentas de replicação do PostgreSQL

As diversas ferramentas de código aberto para replicação de dados no PostgreSQL foram selecionadas por meio da documentação do [Group 2019a] e de artigos científicos que utilizam dessas ferramentas para experimentos entre outras finalidades. Suas principais características estão na Tabela 1.

Ferramenta	Maturidade	Licença	Versões suportadas do PostgreSQL	Versão atual
PgCluster	Descontinuado	BSD	8.0	1.3
Mammoth	Descontinuado	BSD	8.3	1.8
Bi-Directional Replication (BDR)	Estável	BSD	9.4	1.0.65
pglogical	Estável	PostgreSQL	9.4 a 11	2.2.1
rubyrep	Inativo	MIT	Não encontrado	2.0.1
Pgpool-II	Estável	BSD	7.3 a 11	4.0.1
Slony-I	Estável	BSD	8.3 a 10	2.2.7
Bucardo	Estável	BSD	8.1 a 11	5.5.0
Postgres-XL	Estável	PostgreSQL	9.5 a 10	10R1
Citus	Estável	AGPL	9.5 a 10	7.5

Tabela 1. Ferramentas de replicação do PostgreSQL com suas características

No desenvolvimento deste trabalho, consideramos ferramentas que apoiam as versões ativas do PostgreSQL, 9.3 a 11, estão estáveis e não estão descontinuadas ou

inativas (sem atualização no último ano). As ferramentas PgCluster [PgFoundry 2009] e Mammoth [Bishop 2010] estão descontinuadas, portanto não serão abordadas neste trabalho. A ferramenta Bi-Directional Replication [2ndQuadrant 2019b] possui uma versão 3.0 proprietária que tem como dependência a extensão pglogical [2ndQuadrant 2019a], adicionando recursos a essa ferramenta para poder realizar replicações multi-primário e algumas outras operações. Embora sua versão 1.0.65 seja de código aberto, ela funciona apenas em uma versão modificada do PostgreSQL 9.4, e por isso não será avaliada neste artigo. A ferramenta pglogical [2ndQuadrant 2019a] possui diversas restrições em sua replicação, principalmente em redes de com muitas máquinas onde o fato da configuração das relações de *provider-subscriber* só pode ser realizada uma máquina de cada vez. Nesse cenário, também só pode haver um índice, ou uma restrição (*constraint*), ou uma chave primária. Dado estas restrições, o pglogical não será avaliado neste artigo. A ferramenta rubyrep [Lehmann 2017] está inativa, com atualizações esporádicas, a última datando da metade de 2017, e também não será considerada para as comparações. Por fim, a ferramenta Pgpool-II também não será avaliada, pois sua documentação recomenda outros meios para realizar a replicação, como a replicação nativa do PostgreSQL ou a ferramenta Slony-I, sendo a replicação nativa do próprio Pgpool-II a menos recomendada [PgFoundry 2019].

A replicação nativa do PostgreSQL e as ferramentas: Slony-I, Bucardo, Postgres-XL e Citus serão abordadas neste artigo e estão explanadas a seguir.

3.1. Replicação nativa do PostgreSQL

A replicação nativa do PostgreSQL [Group 2019a] utiliza o método de replicação WAL (Write-Ahead-Logging), em que é realizado a transmissão e cópia do log do banco de dados para os outros servidores, de modo que os dados entre eles fiquem consistentes. Inicialmente assíncrona em sua primeira implementação na versão 9.0 do PostgreSQL, ela pode ser configurada para funcionar de maneira síncrona a partir da versão 9.1. A partir da versão 10 do PostgreSQL replicação lógica passou a ser apoiada nativamente. A realização de cascateamento e de promoção de nós são possíveis, assim permitindo tolerância a falhas com uma configuração manual, envolvendo reiniciamento do nó primário problemático para resolução de problemas e reinserção.

3.2. Slony-I

O Slony-I [Group 2019b] é uma ferramenta de replicação assíncrona que utiliza a replicação primário-réplicas baseada em *triggers*. A ferramenta suporta replicação parcial sendo possível replicar apenas algumas tabelas do BD. Também é possível que diferentes nós repliquem diferentes tabelas. Cascateamento, promoção de nós e tolerância a falhas de forma automatizada também são suportados pela ferramenta. A ferramenta funciona com réplicas em diferentes versões do PostgreSQL e em diferentes sistemas operacionais. O Slony possui configuração complexa, sendo necessário adicionar, classificar e gerenciar todos os nós por meio de *scripts bash*.

3.3. Bucardo

Uma ferramenta de replicação assíncrona, multi-primário e primário-réplicas baseada em *triggers*. Aceita qualquer número de fontes (*primary*) e alvos (*replica*), replicação parcial e por demanda [Bucardo 2019]. Essa ferramenta permite lidar com a falha de nós da rede

de duas maneiras. No caso de falha em uma replica, realiza-se o redirecionamento do fluxo para outro nó. No caso de falhas de um servidor primário, há uma pequena sequência de passos a serem seguidos, sendo necessário uma resolução de conflito personalizada para casos de replicação multi-primário. Tolerância a falhas não é um objetivo primário do Bucardo [Bucardo 2019], de forma que nenhum dos tratamentos citados acima funcionam nativamente nem são configurados de forma automática.

3.4. Postgres-XL

A ferramenta Postgres-XL [Postgres-XL 2019] realiza a replicação de forma síncrona e balanceamento de leitura e escrita, utilizando MPP de forma transparente. O Postgres-XL possui dois componentes que lidam com os dados: o *Coordinator* e o *Datanode*. O *Coordinator* é uma interface que recebe as declarações em SQL, as analisa e planeja, para então determinar quais *Datanodes* serão envolvidos na transação, enviando uma plano serializado para cada componente envolvido.

3.5. Citus Community

A ferramenta Citus [Citus Data 2019] realiza replicação síncrona e assíncrona, apresentando também um recurso de tabelas distribuídas e MPP. Ela possui um nó coordenador que serve os nós *workers*. O coordenador possui apenas os metadados dos *workers*, que por sua vez armazenam as tabelas de dados.

A forma como a ferramenta Citus lida com o balanceamento de carga e com a fragmentação de dados facilita a inclusão de novos nós na rede. Quanto ao tratamento de falhas, existem algumas opções. Para tratar falhas em *workers*, Citus recomenda: (i) para sistemas OLTP, com grandes cargas de trabalho, baseia-se em habilitar a replicação nativa do PostgreSQL para substituir temporariamente o nó; e (ii) para sistemas com cargas de trabalho de anexação (leituras), baseia-se em habilitar a replicação de fragmentos do Citus no nó, garantindo os seus dados sejam replicados para outros nós. No caso do nó coordenador, que são comparativamente menores e menos manipuladas, a replicação nativa do PostgreSQL também pode ser utilizado, tornando simples e rápida a tarefa de substituição do nó em caso de falhas ou manutenções programadas. Servidores em *standby* são gerados utilizando WAL, de maneira que a alta disponibilidade é garantida.

4. Resultados

Existem diversos conceitos e características ligados a replicação de banco de dados, resultando em vários métodos de replicação. As ferramentas de replicação de banco de dados analisadas implementam um ou mais métodos de replicação aliados a características e/ou facilidades provenientes da própria ferramenta. Um resumo da comparação entre as características e métodos de replicação está exposto na Tabela 2.

Modelo de replicação: a ferramenta Bucardo apoia primário-réplica e multi-primário. A replicação nativa do PostgreSQL e a ferramenta Slony-I apoiam apenas replicação primário-réplica e as ferramentas Postgres-XL e Citus apoiam apenas multi-primário.

Balanceamento de Carga: uma vez que para realizar o balanceamento de leitura basta haver dados replicados, todas as ferramentas analisadas permitem esse tipo de balanceamento. A ferramenta Citus, em especial, pode apresentar melhores resultados nesta tarefa caso utilize a fragmentação de dados aliada ao processamento paralelo de instruções

Características	Ferramentas				
	Replicação nativa do PostgreSQL	Slony-I	Bucardo	Postgres-XL	Citus
Modelo de sincronismo	Síncrono e Assíncrono	Assíncrono	Assíncrono	Síncrono e Assíncrono	Síncrono
Modelo de replicação	Primário-Réplica	Primário-Réplica	Primário-Réplica e Multi-Primário	Multi-Primário	Multi-Primário
Balanceamento de carga	Apenas Leitura	Leitura e Escrita*	Leitura e Escrita	Leitura e Escrita	Leitura e Escrita
Tolerância a falhas automatizada		✓			✓
Replicação Parcial		✓	✓		✓
WAL	✓				
Replicação Lógica	✓				
Replicação por meio de *triggers*		✓	✓		
Fragmentação de dados					✓
MPP				✓	✓

Tabela 2. Características e métodos de replicação ofertadas pelas ferramentas analisadas.

SQL. O balanceamento de escrita, no entanto, é mais complexo e pode facilmente gerar conflitos e bloqueios em excesso. A única ferramenta analisada que não permite balanceamento de escrita é a replicação nativa do PostgreSQL. A ferramenta Slony-I permite que haja mais de um servidor primário na rede, sendo que cada um deles deve ser responsável por tabelas diferentes. Este recurso permite realizar balanceamento de escrita até certo nível, ao deixar servidores diferentes responsáveis pela escrita de tabelas diferentes. Não é possível, no entanto, realizar o balanceamento entre instruções de escrita na mesma tabela. As demais ferramentas viabilizam o balanceamento de escrita na mesma tabela, porém pode haver vários travamentos que podem causar lentidão. As ferramentas Postgres-XL e Citus, por utilizarem o processamento paralelo de instruções SQL, podem ter o melhor desempenho neste balanceamento.

Tolerância a falhas automatizada: a rápida recuperação de uma falha é uma característica importante para manter a disponibilidade da aplicação. As ferramentas Slony-I e Citus permitem que tais recuperações sejam feitas de forma automatizada, desde que configurado anteriormente. No Slony-I deve-se informar qual nó será promovido para primário em caso de falhas. A Citus replica os fragmentos de um nó para uma quantidade previamente estipulada de outros nós na mesma rede. Quando um nó falha, a ferramenta automaticamente redireciona o fluxo daquele nó para outro nó que possua os dados solicitados. As demais ferramentas apresentam recursos primitivos para tratamento de falhas, que envolvem intervenção manual e retrabalho.

Replicação parcial: dentre as ferramentas analisadas, Slony-I, Bucardo e Citus permitem a replicação parcial de dados. As ferramentas Slony-I e Bucardo permitem que apenas tabelas específicas sejam replicadas para a rede de banco de dados distribuídos. A ferramenta Citus permite fragmentar os dados de forma que é possível replicar até mesmo apenas determinadas linhas ou colunas de uma tabela.

WAL: este método de replicação implementado pela ferramenta nativa do PostgreSQL

executa a replicação de forma assíncrona ou síncrona. As demais ferramentas analisadas não apresentam este tipo de replicação implementado.

Replicação lógica: a partir de sua versão 10, o PostgreSQL disponibiliza nativamente a replicação lógica, porém este recurso ainda não está apto a lidar com a replicação de grandes quantidades de dados [Severalnines 2018]. A ferramenta pglogical implementa este método; entretanto, não foi considerada para esta avaliação devido aos motivos citados anteriormente.

Replicação baseado em *triggers*: as ferramentas Slony-I e Bucardo são as únicas a implementar este método de replicação. É um método assíncrono e pode mostrar dados infiéis ao acessar os nós réplicas caso haja alterações que ainda não foram aplicadas a eles. Ambas as ferramentas utilizam tabelas auxiliares para manter o controle das replicações.

Fragmentação de dados: a ferramenta Citus implementa este modelo de replicação de fragmentação de dados, em que divide-se linhas e/ou tabelas entre os nós da malha de replicação. Por ser uma replicação focada em desempenho, recomenda-se um servidor com uma cópia de todos os dados, sem fragmentação, seja mantido para casos de falha. Manter cópias dos fragmentos de um nó da rede em outros nós da rede também é uma opção. A ferramentas Citus é a única a implementar este método de replicação.

MPP: este modelo de replicação faz com que a malha de replicação se torne um *cluster*, paralelizando a execução e concentrando os resultados em um servidor central. Isso resulta em um aumento de desempenho, principalmente para dados massivos. Seu desempenho é maximizado quando combinado com a fragmentação de dados. A ferramenta Postgres-XL implementa este modelo de replicação, tal como a ferramenta Citus.

5. Trabalhos Relacionados

A replicação de dados é um tema abrangente e amplamente discutido na área de Banco de Dados. Um levantamento de ferramentas comerciais e replicação é apresentado em [Moiz et al. 2011], porém o PostgreSQL não foi o foco principal. Detalhes sobre o processo da replicação seus conceitos e contextos são apresentados em [Wiesmann et al. 2000]. Entretanto, não faz comparativo entre os métodos de replicação existentes. Há um interessante estudo comparativo das ferramentas Slony-I e pgpool-II apresentado em [Partio 2007]. Neste trabalho, realiza-se uma breve descrição uma avaliação comparativa de desempenho, com ênfase no balanceamento de leitura. Em [Mauchle 2008], realiza-se um breve comparativo entre a replicação de dados no MySQL e no PostgreSQL, citando algumas ferramentas de replicação, como Slony-I e Bucardo.

6. Considerações Finais

O PostgreSQL possui diversas ferramentas que possibilitam replicação dos dados; entretanto, determinar quais delas é mais adequada para cada situação pode ser algo complexo. Neste trabalho, realizou-se uma estudo exploratório das características relevantes das principais de ferramentas de replicação do PostgreSQL.

Dentre todas soluções analisadas, a ferramenta nativa do PostgreSQL é a que tem menos recursos. Embora possibilite tanto a replicação síncrona quanto a assíncrona, ela não permite balanceamento de escrita nem replicação parcial. É uma solução de replicação para casos simples que não lidem com grandes volumes de dados, nos quais a

tolerância a falhas de forma automática não se faz necessária. O Slony-I é uma solução que abrange dos casos mais simples (backup), aos mais complexos (balanceamento de carga de uma grande aplicação). Trabalha apenas de forma assíncrona e possui alguns recursos como a tolerância a falhas automatizada e a replicação parcial das bases de dados. Um diferencial interessante do Slony-I é que ele permite que haja mais de um nó primário no BDD, desde que cada nó primário seja responsável pela replicação de uma tabela diferente. Essa característica possibilita o balanceamento de escrita até certo nível, mesmo estando em um modelo primário-réplica.

Bucardo é uma ferramenta semelhante ao Slony-I, e suas diferenças moram em dois recursos: (i) o Bucardo realiza replicações multi-primário, o que não é possível no Slony-I; e (ii) a tolerância a falhas no Bucardo só existe de forma totalmente manual, diferentemente do Slony-I que possui uma forma automatizada. A ferramenta Bucardo permite ainda o balanceamento de escrita na mesma tabela, porém há falta de suporte a tratamento de falhas.

As duas ferramentas restantes, Postgres-XL e Citus, são recomendadas para bases com um grande volume de dados e vários servidores na rede de replicação. Ambas ferramentas transformam a rede de replicação em um *cluster*, aumentando o desempenho no processamento das consultas. A ferramenta Postgres-XL permite o balanceamento de carga e realiza sua replicação síncrona de forma paralela em todo o BDD. Ela não possui, entretanto, formas para replicação parcial ou tolerância a falhas automatizada. A ferramenta Citus também possibilita balanceamento de carga e realiza sua replicação de forma síncrona e paralela. A diferença entre estas ferramentas é que a Citus possui tolerância a falhas de forma automatizada e também possibilita a replicação parcial por meio da fragmentação da dados.

Como trabalho futuro, pretende-se realizar testes de desempenho com as ferramentas Slony-I e Citus por meio de *benchmarks* com o objetivo de averiguar as possibilidades de utilização dessas ferramentas em produção.

Referências

2ndQuadrant (2019a). pglogical — 2ndquadrant. Disponível em: https://www.2ndquadrant.com/en/resources/pglogical. Acesso em 17/01/2019.

2ndQuadrant (2019b). Postgres-bdr documentation. Disponível em: http://bdr-project.org/docs/stable/index.html. Acesso em 17/01/2019.

Bishop, S. (2010). Mammoth replicator. Disponível em: https://launchpad.net/mammoth-replicator. Acesso em 10/12/2018.

Bucardo (2019). Bucardo asynchronous postgresql replication system. Disponível em: https://bucardo.org/Bucardo. Acesso em 15/01/2019.

Citus Data, I. (2019). Citus documentation. Disponível em: https://docs.citusdata.com/en/v8.1/index.html. Acesso em 19/01/2019.

Group, P. G. D. (2019a). Postgresql. Disponível em: http://www.postgresql.org. Acesso em 16/03/2019.

Group, S. D. (2019b). Slony-i enterprise-level replication system. Disponível em: http://slony.info/. Acesso em 15/01/2019.

Gupta, S., Saroha, K., and Bhawna (2011). Fundamental research of distributed database. IJCSMS - International Journal of Computer Science and Management Studies, Vol. 11, Issue 02, Aug 2011 - Disponível em: https://pdfs.semanticscholar.org/f935/1f0cf3c4307dd76c85d6815e2a1b8095324b.pdf. Acesso em 16/03/2019.

Heisler, D. A. (2008). Estudo de algoritmos e técnicas de teplicação de banco de dados em software livre. Disponível em: https://www.univates.br/bdu/bitstream/10737/563/1/2008DanielAfonsoHeisler.pdf. Acesso em 06/02/2019.

Lehmann, A. (2017). rubyrep: Home. Disponível em: http://www.rubyrep.org. Acesso em 10/12/2018.

MariaDB (2019). Replication overview. Disponível em: https://mariadb.com/kb/en/library/replication-overview/. Acesso em 09/02/2019.

Mauchle, F. (2008). Database replication with mysql and postgresql. Disponível em: https://wiki.hsr.ch/Datenbanken/files/Mauchle_Replication_MySQL_Postgres_Paper.pdf. Acesso em 08/02/2019.

Mazilu, M. C. et al. (2010). Database replication. *Database Systems Journal*, 1(2):33–38.

Moiz, S. A., P., S., G., V., and Pal, S. N. (2011). Article: Database replication: A survey of open source and commercial tools. *International Journal of Computer Applications*, 13(6):1–8.

Partio, M. (2007). Evaluation of postgresql replication and load balancing implementations. Unpublished.

PgFoundry (2009). Pgfoundry: Pgcluster. Disponível em: http://pgfoundry.org/projects/pgcluster. Acesso em 10/12/2018.

PgFoundry (2019). Pgpool wiki. Disponível em: http://www.pgpool.net/mediawiki/index.php/Main_Page. Acesso em 11/01/2019.

Postgres-XL (2019). Open sourcescalable sql database cluster. Disponível em: https://www.postgres-xl.org. Acesso em 15/01/2019.

Server, S. (2019). Tipos de replicação. Disponível em: https://docs.microsoft.com/pt-br/sql/relational-databases/replication/types-of-replication. Acesso em 09/02/2019.

Severalnines (2018). An overview of logical replication in postgresql. Disponível em: https://severalnines.com/blog/overview-logical-replication-postgresql. Acesso em 08/02/2019.

Tauro, C. J., Patil, B. R., and Prashanth, K. (2013). A comparative analysis of different nosql databases on data model, query model and replication model. In *Proceedings of the International Conference on ERCICA*.

Wiesmann, M., Pedone, F., Schiper, A., Kemme, B., and Alonso, G. (2000). Understanding replication in databases and distributed systems. In *Proceedings 20th IEEE International Conference on Distributed Computing Systems*, pages 464–474.