

Manual de Redis-IR

Arlino Magalhães¹

¹Centro de Educação Aberta e a Distância

Universidade Federal do Piauí

Teresina – PI – Brasil

{arlino@ufpi.edu.br

Esse documento provê um manual de como manusear o banco de dados Redis Instant Recovery (Redis-IR). Redis-IR é um protótipo que implementa a técnica de recuperação instantânea no banco de dados Redis (versão 5.0.7). Essa técnica permite a aplicações processar transações imediatamente ao reinício do sistema após uma falha de sistema (por exemplo, desligamento de energia). Através da recuperação instantânea, clientes e aplicações não precisam esperar pelo atraso da recuperação após uma falha. A ideia principal da recuperação instantânea é processar transações durante o processo de recuperação, dando a impressão de que a recuperação foi instantânea.

Summary

1	Informações gerais	3
2	Como instalar Redis-IR	4
2.1	Instalando o compilador C++	4
2.2	Instalando Libconfig	4
2.3	Instalando Berkely DB	4
2.4	Instalando Redis-IR	4
3	Usando Redis-IR	5
3.1	Executando o servidor Redis-IR	5
3.2	Executando comandos	5
3.3	”Derrubando” e reiniciando o servidor	5
4	Configurando Redis-IR	6
4.1	Habilitando a recuperação instantânea	6
4.2	Habilitando a recuperação síncrona	6
4.3	Trocando a estrutura de dados do <i>log</i> indexado	6
5	Benchmarking	6

5.1	Instalando o <i>benchmark</i> Memtier	7
5.2	Usando Memtier automaticamente em Redis-IR	7
6	Simulando bancos de dados	8
6.1	Criando uma base de dados usando Memtier	8
6.2	Mudando os nomes dos arquivos de <i>log</i> de Redis-IR	9
6.3	Fazendo o <i>backup</i> do banco de dados	9
7	Simulando cargas de trabalho	9
7.1	Criando cargas de trabalho OLTP usando Memtier	9
8	Simulando de falhas de sistema	9
8.1	Disparando uma falha de sistema	9
8.2	Disparando uma falha durante o processamento normal do SGBD	10
8.3	Disparando falhas sucessivas	10
9	Relatórios da execução de Redis-IR	10
9.1	Gerando relatório de recuperação	10
9.2	Gerando relatório sobre as operações executadas no banco de dados . . .	11
9.3	Gerando relatório sobre a taxa de escrita nos arquivos de <i>log</i>	11
9.4	Gerando relatório sobre uso dos recurso do sistema	11
9.5	Gerando gráficos	12
10	Comece Agora!	12
11	Saiba mais informações sobre a técnica de Recuperação Instantânea	13

1. Informações gerais

A tecnologia de bancos de dados em memória manuseia o banco de dados primário na memória RAM para prover baixa latência e altas taxas de vazão de dados. Entretanto, a memória volátil faz os bancos de dados em memória muito mais sensíveis a falhas de sistemas (falta de energia, por exemplo). Em tais falhas, o conteúdo do banco de dados é perdido e, como resultado, o sistema fica indisponível por um longo tempo até o processo de recuperação do banco de dados ser terminado. Por exemplo, após uma falha, a recuperação padrão do banco de dados Redis (semelhante a recuperação implementada pela maioria dos bancos de dados em memória) deve recuperar o banco de dados completamente na memória antes de executar novas transações.

Assim, novas técnicas de recuperação são necessárias para reparar falhas de bancos de dados o mais rápido possível. Redis-IR é um *fork* de Redis que implementa a técnica de recuperação instantânea. A técnica de recuperação instantânea permite ao banco de dados escalonar transações simultaneamente ao processo de recuperação do banco de dados desde o reinício do sistema. Assim, aplicações e usuários não notam o processo de recuperação, dando a impressão de que o sistema foi recuperado instantaneamente. Uma vez que o sistema começa a escalonar transações o mais rápido possível, ele é capaz de entregar altas taxas de entrada/saída por segundo, ou seja, ele executa uma carga de trabalho mais rápido.

A Figura 1 mostra dois experimentos de recuperação: recuperação instantânea implementada por Redis-IR (linha azul) e a recuperação padrão implementada pela maioria dos bancos de dados em memória (linha amarela). As linhas dos gráficos representam a média da vazão das transações em pequenos intervalos de tempo. Observando a Figura 1, é possível ver que a abordagem de recuperação padrão possui um tempo de espera depois da falha, enquanto o banco de dados está se recuperando. Por outro lado, a abordagem de recuperação instantânea escalona novas transações imediatamente depois que o sistema reinicia durante o processo de recuperação.

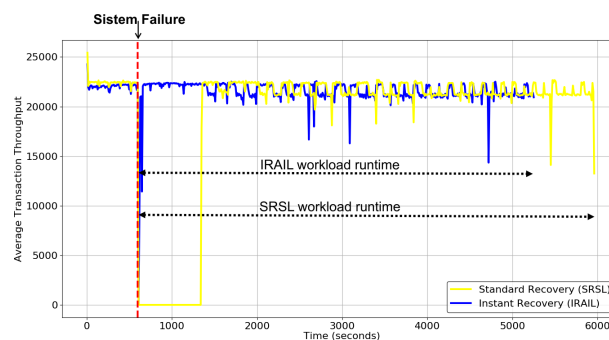


Figure 1. Vazão de transações durante o processo de recuperação: Recuperação Instantânea vs. Recuperação Padrão.

A recuperação padrão precisa percorrer o arquivo de *log* sequencial (Figura 2 (a)) completamente para recuperar o banco de dados. A recuperação instantânea usa um arquivo de *log* indexado (Figura 2 (b)) para recuperar o banco de dados ao invés de um *log* sequencial. O *log* indexado é uma estrutura de índice (por exemplo, árvore B⁺ ou tabela *Hash*) que permite recuperar tuplas na memória incrementalmente ou sob demanda.

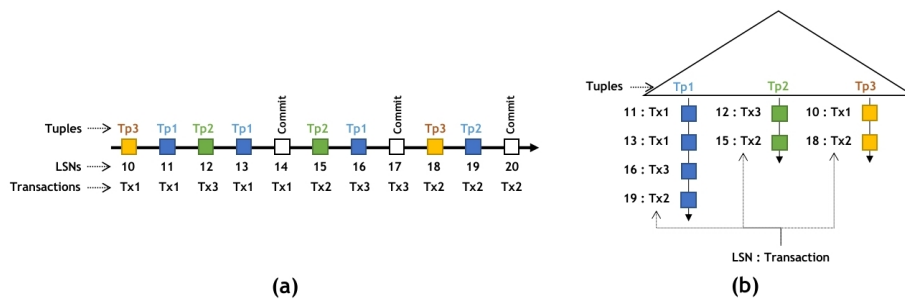


Figure 2. Estruturas de dados dos arquivos de *log* sequencial (a) e indexado (b).

A Seção 11 apresenta alguns artigos que fornecem mais detalhes sobre a abordagem de recuperação instantânea de bancos de dados.

2. Como instalar Redis-IR

Redis-IR pode ser instalado no Ubuntu 20.04 ou versões anteriores. Porém, o protótipo foi testado apenas nas versões 20.04, 18.04 e 16.04.

As seguintes bibliotecas são necessárias para instalar Redis-IR:

1. Compilador C++
2. Libconfig (libconfig-dev)
3. Berkely DB (libdb-dev)

2.1. Instalando o compilador C++

```
1 sudo apt install build-essential
```

2.2. Instalando Libconfig

```
1 sudo apt-get update -y
2 sudo apt-get install -y libconfig-dev
```

2.3. Instalando Berkely DB

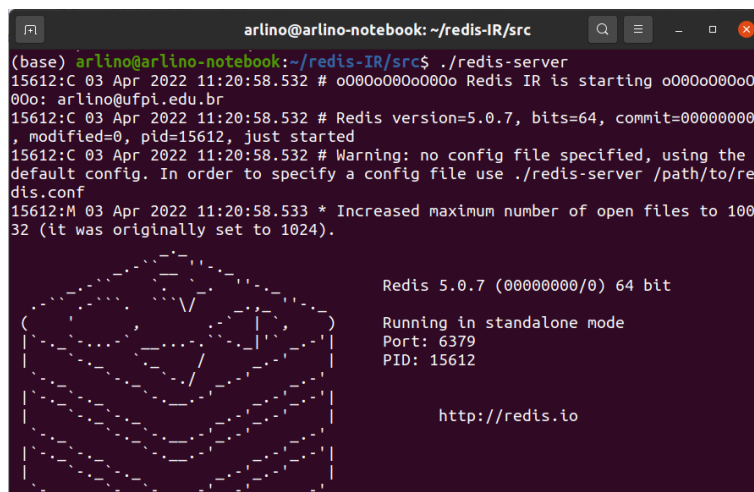
```
1 sudo apt-get update
2 sudo apt-get install libdb-dev
```

2.4. Instalando Redis-IR

Faça o *download* de Redis-IR no seguinte *link*: <https://drive.google.com/drive/folders/1LTbtY36O0kWIpXZBM-hc1BPvIjICuy2F?usp=sharing>

No mesmo diretório onde Redis-IR foi salvo, execute os seguintes comandos:

```
1 tar -xf redis-IR.tar.xz
2 cd redis-IR
3 make
```

A terminal window titled 'arlino@arlino-notebook: ~/redis-IR/src' showing the execution of './redis-server'. The output includes timestamps, version information (Redis 5.0.7), and a warning about the default configuration. A large Redis logo is displayed on the left, and server details (Port: 6379, PID: 15612) and the URL 'http://redis.io' are shown on the right.

```
(base) arlino@arlino-notebook:~/redis-IR/src$ ./redis-server
15612:C 03 Apr 2022 11:20:58.532 # 000000000000 Redis IR is starting 000000000
000: arlino@ufpi.edu.br
15612:C 03 Apr 2022 11:20:58.532 # Redis version=5.0.7, bits=64, commit=00000000
, modified=0, pid=15612, just started
15612:C 03 Apr 2022 11:20:58.532 # Warning: no config file specified, using the
default config. In order to specify a config file use ./redis-server /path/to/re
dis.conf
15612:M 03 Apr 2022 11:20:58.533 * Increased maximum number of open files to 100
32 (it was originally set to 1024).
```

Figure 3. Redis-IR executando.

3. Usando Redis-IR

3.1. Executando o servidor Redis-IR

Entre no diretório "redis-IR/src" e execute o comando abaixo:

```
1 ./redis-server
```

A Figura 3 mostra o servidor do sistema de banco de dados Redis-IR executando.

3.2. Executando comandos

Redis possui uma aplicação, chamada de **redis-cli** (*Redis command line interface*), que permite interagir com o sistema de bancos de dados. A aplicação *redis-cli* é usado para enviar comandos e ler respostas do servidor Redis. Para usar *redis-cli*, entre no diretório "redis-IR/src" e execute o comando abaixo:

```
1 ./redis-cli
```

O comando **SET** insere uma chave/valor no banco de dados. O comando abaixo insere uma chave *key1* com um valor *value1*. Execute o comando em *redis-cli*.

```
1 set key1 value1
```

O comando **GET** recupera um valor do banco de dados usando uma chave. O comando abaixo obtém o a valor *value1* através da chave *key1*. Execute o comando em *redis-cli*.

```
1 get key1
```

3.3. "Derrubando" e reiniciando o servidor

Tente "derrubar" o sistema de banco de dados e executá-lo novamente para observar a recuperação instantânea. Use o comando abaixo na aplicação *redis-cli* para "derrubar" o servidor de bancos de dados.

```
1 shutdown
```

Grandes arquivos de *log* são necessários para observar melhor o processo de recuperação instantânea. Assim, é necessário colocar o sistema em produção para gerar grandes arquivos de *log*. A Seção 6.1 mostra como gerar um banco de dados usando uma ferramenta de *benchmark*.

4. Configurando Redis-IR

As configurações de execução de Redis-IR são armazenadas no arquivo ***redis.ir.conf***, localizado no diretório principal de Redis-IR.

4.1. Habilitando a recuperação instantânea

A recuperação instantânea pode ser habilitada setando o valor do campo ***instant_recovery_state*** para **"ON"**. O valor padrão desse campo é **"ON"**, ou seja, Redis-IR executa a recuperação instantânea por padrão.

```
1 instant_recovery_state = "ON" ;
```

Se o valor do campo *instant_recovery_state* for **"OFF"**, o sistema executará a recuperação padrão de Redis, ou seja, ele executará uma recuperação através de *log* sequencial. Nesse caso, transações podem ser executadas apenas depois do processo de recuperação ser completado.

4.2. Habilitando a recuperação síncrona

A técnica de recuperação instantânea usa um *log* indexado. Por questões de performance, as inserções no *log* indexado são assíncronas ao *commit* da transação, ou seja, uma transação não precisa esperar por inserções no *log* indexado para poder *"comitar"*. Porém, a indexação pode ser síncrona, ou seja, cada atualização de transação deve esperar as inserções no *log* para confirmar suas escritas. Para habilitar a indexação síncrona, aplique o valor do campo *instant_recovery_synchronous* para **"ON"**. O valor padrão desse campo é **"OFF"**.

```
1 instant_recovery_synchronous = "ON";
```

4.3. Trocando a estrutura de dados do *log* indexado

Redis-IR usa uma árvore B⁺ como arquivo de *log* indexado por padrão. Entretanto, uma tabela *hash* pode ser utilizada também. O campo ***indexedlog_structure*** no arquivo de configuração de Redis-IR permite a mudança da estrutura de dados do *log* indexado. O valor padrão desse campo é **"BTREE"**.

```
1 indexedlog_structure = "BTREE"; //BTREE | HASH.
```

5. Benchmarking

Redis-IR pode usar o *benchmark* Memtier para simular cargas de trabalho. Memtier é uma ferramenta de *benchmark* desenvolvida pela Redis Labs para o banco de dados Redis. Essa ferramenta possui uma interface de linha de comando que provê um conjunto de customizações para gerar vários padrões de cargas de trabalho. Redis e Memtier são programas diferentes e devem ser instalados e executados separadamente. Porém, Redis-IR foi implementado para usar Memtier automaticamente.

5.1. Instalando o *benchmark* Memtier

A bibliotecas abaixo são necessárias para instalar Memtier:

1. libevent 2.0.10 ou versão mais nova
2. libpcre 8.x
3. OpenSSL

Nas distribuições Ubuntu/Debian, instale os pré-requisitos de instalação de Memtier usando o comando abaixo:

```
1 sudo apt-get install build-essential autoconf automake libpcre3-dev  
libevent-dev pkg-config zlib1g-dev libssl-dev
```

Embora Memtier e Redis sejam programas diferentes, Memtier foi colocado dentro do diretório de Redis-IR para que esse primeiro pudesse ser executado juntamente com esse último. Assim, para instalar Memtier, entre no diretório "*redis-IR/src/memtier_benchmark*" e execute os comandos abaixo:

```
1 autoreconf -ivf  
2 ./configure  
3 make  
4 sudo make install
```

Se alguns pacotes não forem encontrados durante a instalação (libssl e zlib1g, por exemplo), tente instalar novamente os pré-requisitos de instalação de Memtier ou instale os pacotes que faltaram separadamente. Para mais informações sobre Memtier, acesse o *git* da ferramenta: https://github.com/RedisLabs/memtier_benchmark.

5.2. Usando Memtier automaticamente em Redis-IR

Memtier pode ser executado automaticamente quando Redis-IR for iniciado através de configurações no arquivo *redis.ir.conf*. Assim, antes de executar Redis-IR, o valor do campo *memtier_benchmark_state* deve ser setado para "ON". O valor padrão desse campo é "OFF".

```
1 memtier_benchmark_state = "ON";
```

Uma carga de trabalho produzida por Memtier pode ser configurada através de alguns parâmetros no campo *memtier_benchmark_parameters*. Abaixo estão dois exemplos de parâmetros de carga de trabalho:

Exemplo 1: cria uma carga de trabalho que gera um banco de dados contendo 5.000 tuplas.

```
1 memtier_benchmark_parameters = " --hide-histogram -n 5000 --key-  
prefix='redisIR-' --key-minimum=1 --key-maximum=5000 --command='set  
__key__ __data__' --command-ratio=5000 --command-key-pattern=S"
```

Os parâmetros acima geram 5.000 comandos por cliente. Cinquenta clientes são manuseados por cada *thread* por padrão. Quatro *threads* são manuseadas por padrão. O prefixo do nome da chave é definido como 'redisIR-'. O sufixo da chave está numa faixa de valores entre 1 e 5000. Um exemplo de chave: *redisIR-1999*. As operações são comandos *SET*. O parâmetro "*__key__*" gera valores de chaves seguindo os padrões definidos de prefixo e sufixo das chaves. O parâmetro "*__data__*" gera valores de dados seguindo a opção padrão de objetos de Memtier. A geração de chaves segue um padrão sequencial (S).

Exemplo 2: simula uma carga de trabalho OLTP para o banco de dados gerando no Exemplo 1 (acima).

```
1 memtier_benchmark_parameters = " --hide-histogram -n 5000 --ratio 5:5  
  --randomize --key-prefix='redisIR-' --key-minimum=1 --key-maximum  
  =1000"
```

Esses parâmetros geram 5.000 comandos randômicos por cliente em uma faixa de 5:5 entre comandos *SET* e *GET*. O prefixo do nome da chave é mudado para *'redisIR-'*. O sufixo do nome da chave está na faixa entre 1 e 1.000.

Existem outros exemplos de parâmetros de carga de trabalho no arquivo *redis-ir.conf*. Para ver todas as opções de configuração de Memtier, execute o comando abaixo no diretório raiz da ferramenta.

```
1 ./memtier_benchmark --help
```

6. Simulando bancos de dados

Redis-IR implementa persistência apenas através de dois arquivos de *log*. Redis-IR cria um arquivo de *log* sequencial, chamado de *sequentialLog.aof* por padrão, e de um arquivo de *log* indexado, chamado de *indexedLog.db* por padrão. Esses dois arquivos são gerados juntos durante o processamento normal das transações e são armazenados no diretório *redis-IR/src/logs*.

6.1. Criando uma base de dados usando Memtier

Memtier pode ser usado para simular cargas de trabalho para criar uma base de dados. Abaixo está um exemplo de parâmetros para uma carga de trabalho que cria 5.000 tuplas de banco de dados e 20.000.000 de registros de *log*. Depois do parâmetro da carga de trabalho ser setado, o sistema de banco de dados deve ser iniciado (veja a Seção 3.1). Além disso, Memtier deve ser habilitado para executar automaticamente junto com o sistema (veja a Seção 5.2).

```
1 memtier_benchmark_parameters = " --hide-histogram -n 5000 --key-  
  prefix='redisIR-' --key-minimum=1 --key-maximum=5000 --command='set  
  __key__ __data__' --command-ratio=5000 --command-key-pattern=S"
```

A carga de trabalho acima deve ser executada algumas vezes para gerar arquivos de *log* grandes o suficiente para melhor observar o processo de recuperação instantânea. Uma mesma carga de trabalho pode ser executada um número dado de vezes através do campo *memtier_benchmark_workload_run_times*. O valor padrão para esse campo é 1.

```
1 memtier_benchmark_workload_run_times = 10
```

O *log* indexado tende a crescer muito com o tempo. Assim, é recomendado que o *checkpoint* do *log* indexado seja habilitado para acelerar a recuperação. O *checkpoint* pode ser ativado através do parâmetro de configuração abaixo:

```
1 checkpoint_state = "ON"
```


6.2. Mudando os nomes dos arquivos de *log* de Redis-IR

Opcionalmente, os nomes dos arquivos de *log* podem ser renomeados.

Para mudar o nome do *log* sequencial, utilize o campo *aof_filename*.

```
1 aof_filename = "logs/sequentialLog.aof";
```

Para mudar o nome do *log* indexado, utilize o campo *indexedlog_filename*.

```
1 indexedlog_filename = "logs/indexedLog.db";
```

6.3. Fazendo o *backup* do banco de dados

É possível fazer o *backup* do banco de dados armazenando os arquivos de *log* sequencial e indexado em outro local ou dispositivo. Além disso, existe um arquivo, chamado de "*finalLogSeek.dat*", responsável por sincronizar as inserções de registros do *log* sequencial para o *log* indexado. Assim, esse arquivo deve ser armazenado juntamente com os outros.

7. Simulando cargas de trabalho

Memtier pode simular padrões de carga de trabalho. Por exemplo, é possível acessar apenas um subconjunto do banco de dados através de múltiplos clientes para simular uma carga de trabalho OLTP.

7.1. Criando cargas de trabalho OLTP usando Memtier

Os parâmetros abaixo geram uma carga de trabalho que acessa 20% do banco de dados criado no exemplo da Seção 6.1.

```
1 memtier_benchmark_parameters = " --hide-histogram -n 5000 --ratio 5:5  
  --randomize --key-prefix='redisIR-' --key-minimum=1 --key-maximum  
  =1000"
```

8. Simulando de falhas de sistema

Redis-IR tem algumas configurações para simular falhas de sistema. Uma falha de sistema é simulada através de um reinício de sistema de banco de dados. Simulações de falhas podem ser disparadas com ou sem a execução de uma carga de trabalho.

8.1. Disparando uma falha de sistema

É possível simular falhas de sistema através do campo *number_restarts_after_time*. Esse campo permite programar uma quantidade dada de simulações de falhas. Se o valor do campo for 0 (zero), nenhuma falha é realizada. O valor padrão desse campo é 0 (zero).

```
1 number_restarts_after_time = 4
```

O campo *restart_after_time* representa o tempo (em segundos) para simular uma falha depois do início do sistema. Se o valor do campo é 0 (zero), nenhuma falha é realizada. O valor padrão desse campo é 600 segundos.

```
1 restart_after_time = 1200;
```

8.2. Disparando uma falha durante o processamento normal do SGBD

É também possível simular uma falha durante o processamento normal do banco de dados. Nesse caso, primeiro, o banco de dados é carregado inteiramente na memória e só depois o sistema é disponibilizado para novas transações. Apenas após isso, uma falha é realizada depois de um tempo dado. Para isso, o campo *preload_database_and_restart* deve ser setado com o tempo de execução da falha (em segundos). Se o valor do campo for 0 (zero), nada é realizado, ou seja, o sistema não será pré carregado e nem reiniciado.

```
1 preload_database_and_restart = 300
```

8.3. Disparando falhas sucessivas

Falhas sucessivas podem ser simuladas depois do sistema de banco de dados ser carregado na memória usando o campo *number_restarts_after_preloading*. Esse campo é o número de falhas de sistema a serem simuladas. O valor padrão desse campo é 1. Se esse campo tem o valor 0 (zero), nenhuma falha é executada, ou seja, o sistema é só pré carregado. O tempo das falhas sucessivas é setado através do campo *restart_time_after_first_restart*. O tempo da primeira falha é setado no campo *preload_database_and_restart*. O valor padrão do campo *restart_time_after_first_restart* é o mesmo valor do campo *preload_database_and_restart*.

```
1 number_restarts_after_preloading = 5
2 restart_time_after_first_restart = 6000
```

Opcionalmente, falhas sucessivas pode ser realizadas utilizando o campo *number_restarts_after_time*, descrito na Seção 8.1. Contudo, o sistema não será pré carregado na memória e o tempo de realização da falhas sucessivas serão necessariamente iguais ao da primeira falha.

9. Relatórios da execução de Redis-IR

Redis-IR é capaz de produzir relatórios simples com informações sobre a recuperação do sistema, tais como: tempo de recuperação, número de operações executadas, operações realizadas, vazão e latência de transações, uso de recursos do sistema, entre outros. Adicionalmente, informações de cada operação cliente executada no banco de dados podem ser armazenadas em um arquivo CSV. A produção relatórios deve ser habilitada no arquivo *redis_ir.conf* antes do sistema começar a executar.

9.1. Gerando relatório de recuperação

O relatório de recuperação mostra informações sobre o processo de recuperação, tais como: tempo de recuperação, tempo de execução da carga de trabalho, número de tuplas recuperadas incrementalmente e sob demanda, entre outras. O relatório de recuperação é habilitado setando o valor do campo *generate_recovery_report* para "ON". Assim, um arquivo texto é armazenado no diretório *redis-IR/src/recovery_report*, por padrão.

```
1 generate_recovery_report = "ON"
```

Adicionalmente, o nome do arquivo do relatório pode ser modificado através do campo *recovery_report_filename*.

```
1 recovery_report_filename = "recovery_report/recovery_report.txt";
```

9.2. Gerando relatório sobre as operações executadas no banco de dados

É possível gerar um arquivo CSV contendo informações sobre as operações de banco de dados executadas. O arquivo CSV contém os seguintes campos: chave da tupla atualizada, comando da operação, tempo de início da operação, tempo de fim da operação e tipo da operação. A primeira linha do arquivo contém o tempo de início do banco de dados. Além disso, o arquivo pode conter informações de execução de cargas de trabalho do *benchmark Memtier*, *checkpoints* realizados, reinício do sistema, tempo de recuperação e falhas de sistema simuladas.

O arquivo CSV de informações de comandos executados pode ser gerado setando o valor do campo *generate_executed_commands_csv* para "ON". Assim, um arquivo CSV é armazenado no diretório *redis-IR/src/datasets*, por padrão.

```
1 generate_executed_commands_csv = "ON"
```

Adicionalmente, o nome do arquivo CSV pode ser modificado através do campo *executed_commands_csv_filename*.

```
1 executed_commands_csv_filename = "datasets/dataset.csv";
```

É importante ressaltar que a cada reinício do sistema os arquivos de relatório são sobrescritos. Assim, é necessário aplicar o valor do campo *overwrite_report_files* para "OFF" se experimentos com falhas de sistema são realizados.

```
1 overwrite_report_files = "OFF"
```

9.3. Gerando relatório sobre a taxa de escrita nos arquivos de log

A escrita no *log* sequencial é potencialmente mais rápida do que no *log* indexado. Porém, Redis-IR implementa um mecanismo de *buffer* no *log* indexado para mitigar o problema de taxa de escrita. É possível gerar um gráfico para medir e comparar a taxa de escrita nos arquivos de *log* sequencial e indexado. Para isso, um arquivo CSV contendo informações de escrita no *log* indexado pode ser gerado setando *generate_indexing_report_csv* para "ON".

```
1 generate_indexing_report_csv = "ON"
```

Adicionalmente, o nome do arquivo CSV pode ser modificado através do campo *indexing_report_csv_filename*.

```
1 executed_commands_csv_filename = "indexing_report/indexing.csv";
```

9.4. Gerando relatório sobre uso dos recursos do sistema

É possível gerar informações sobre o uso de CPU (em porcentagem) e de RAM (em *kilobytes*). Para isso, o campo *system_usage_state* deve ser setado para "ON".

```
1 generate_system_usage = "ON"
```

Adicionalmente, o nome do arquivo CSV pode ser modificado através do campo *system_usage_csv_filename*.

```
1 system_usage_csv_filename = "system_usage/system_usage.csv";
```

9.5. Gerando gráficos

Existem alguns *scripts* em python no diretório **redis-ir/src/graphics** para gerar gráficos através de arquivos CSV. Por exemplo, o *script* chamado de **throughput** produz um gráfico da taxa de vazão das transações através do arquivo CSV gerado no exemplo da Seção 9.2. A Figura 1 mostra a taxa de vazão de transações durante dois experimentos de recuperação.

Outros gráficos sobre outros aspectos da estratégia de recuperação de Redis-IR podem ser produzidos, tais como: latência, taxa de escrita nos *logs* e uso de recursos do sistema. A Figura 4 mostra alguns gráficos que podem ser gerados usando os *scripts* disponíveis em Redis-IR.

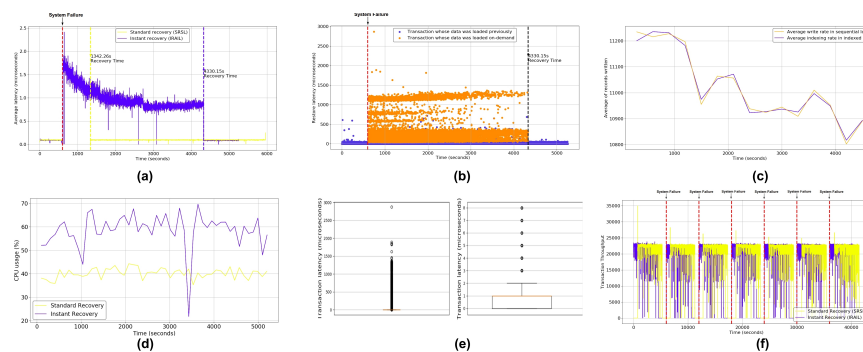


Figure 4. Gráficos Redis-IR: (a) média da latência, (b) latência, (c) taxa de escrita nos logs, (d) uso de CPU, (e) dispersão de latência e (f) vazão de transações em falhas sucessivas.

10. Comece Agora!

Essa seção ordenar os passos básicos para realizar um experimento de recuperação usando um *backup* de banco de dados. Os passo abaixo mostram como instalar Redis-IR, gerar um banco de dados e, finalmente, executar um experimento.

1. Instale Redis-IR seguindo as instruções da Seção 2.
2. Instale o *benchmark* Memtier seguindo as instruções da Seção 5.1.
3. Primeiro, antes de começar um experimento, um banco de dados deve ser simulado. Assim, configure Memtier para executar automaticamente com Redis-IR seguindo as instruções da Seção 5.2. Em seguida, gere o banco de dados (Seção 6.1) e faça o seu *backup* (Seção 6.3). Depois do banco de dados gerado, o sistema deve ser "derrubado" (Seção 3.3). Esse passo pode ser executado apenas uma vez, pois o mesmo *backup* pode ser utilizado em vários experimentos.
4. Configure uma carga de trabalho a ser executada durante o experimento. A Seção 7.1 mostra um exemplo de carga de trabalho OLTP.
5. Configure uma falha de sistema a ser simulada durante o experimento seguindo as instruções da Seção 8. Esse passo é opcional.
6. Habilite a criação de relatórios seguindo as instruções da Seção 9. Esse passo é opcional.
7. Execute o servidor do banco de dados seguindo as instruções da Seção 3.1.
8. **Isso é tudo, pessoal!** Agora você pode observar o sistema executando.

9. Adicionalmente, depois de executar uma carga de trabalho, ou seja, após a execução de um experimento, gráficos pode ser gerados (Seção 9.5) se o passo 6 tiver sido realizado.

Você pode também executar um experimento usando a recuperação padrão de Redis (veja a Seção 4.1) para compará-lo com um experimento de recuperação instantânea.

11. Saiba mais informações sobre a técnica de Recuperação Instantânea

Você pode encontrar mais detalhes sobre a técnica de recuperação instantânea implementada em Redis-IR nos artigos abaixo. Além disso, os artigos possuem exemplos de experimentos de recuperação usando Redis-IR.

Arlino Magalhães. 2021. **Main Memory Databases Instant Recovery**. In Proceedings of the VLDB 2021 PhD Workshop co-located with the 47th International Conference on Very Large Databases (VLDB 2021), Copenhagen, Denmark, August 16, 2021 (CEUR Workshop Proceedings), Philip A. Bernstein and Tilmann Rabl (Eds.), Vol. 2971. CEUR-WS.org.
<http://ceur-ws.org/Vol-2971/paper10.pdf>

Arlino Magalhães, Angelo Brayner, José Maria Monteiro, and Gustavo Moraes. 2021. **Indexed Log File: Towards Main Memory Database Instant Recovery**. In Proceedings of the 24th International Conference on Extending Database Technology, EDBT 2021, Nicosia, Cyprus, March 23 - 26, 2021, Yannis Velegrakis, Demetris Zeinalipour-Yazti, Panos K. Chrysanthis, and Francesco Guerra (Eds.). OpenProceedings.org, 355–360.
<https://doi.org/10.5441/002/edbt.2021.34>

Arlino Magalhães, José Maria Monteiro, and Angelo Brayner. 2021. **Main Memory Database Recovery: A Survey**. ACM Comput. Surv. 54, 2 (2021), 46:1–46:36.
<https://doi.org/10.1145/3442197>