INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
DO SUDESTE DE MINAS GERAIS - CAMPUS RIO POMBA

Miguel Magalhães Lopes

Benchmark de desempenho entre bancos de dados em diferentes arquiteturas

Rio Pomba

20XX

Miguel Magalhães Lopes

Benchmark de desempenho entre bancos de dados em diferentes arquiteturas

Trabalho de Conclusão apresentado ao Campus Rio Pomba, do Instituto Federal de Educação, Ciência e Tecnologia do Sudeste de Minas Gerais, como parte das exigências do curso de Bacharelado em Ciência da Computação para a obtenção do título de Bacharel em Ciência da Computação.

Orientador: Gustavo Henrique da Rocha Reis

Coorientador: CICLANO

Rio Pomba

20XX

Miguel Magalhães Lopes
Benchmark de desempenho entre bancos de dados em diferentes arquiteturas/ Miguel Magalhães Lopes. – Rio Pomba, 20XX-
?? p. : il. (algumas color.) ; 30 cm.

Orientador: Gustavo Henrique da Rocha Reis

Trabalho de Conclusão de Curso – Instituto Federal de Educação, Ciência e Tecnologia do Sudeste de Minas, Campus Rio Pomba, 20XX.

1. 2. I. II. III. IV.

Miguel Magalhães Lopes

Benchmark de desempenho entre bancos de dados em diferentes arquiteturas

Trabalho de Conclusão apresentado ao Campus Rio Pomba, do Instituto Federal de Educação, Ciência e Tecnologia do Sudeste de Minas Gerais, como parte das exigências do curso de Bacharelado em Ciência da Computação para a obtenção do título de Bacharel em Ciência da Computação.

Trabalho aprovado. Rio Pomba, 00 de dezembro de 20XX.

Gustavo Henrique da Rocha Reis,
Orientador, IF Sudeste MG - Rio Pomba

CICLANO, Coorientador, IF Sudeste MG - Rio Pomba

Dr. BELTRANO
IF Sudeste MG - Rio Pomba

Me. XXXXXXXXXXXXX
IF Sudeste MG - Rio Pomba

Rio Pomba
20XX

Este trabalho é dedicado a todos

aqueles que me inspiraram, em especial

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX.

Agradecimentos

Resumo

Palavras-chaves: palavra1. palavra2. palavra3. palavra4.

Abstract

Key-words: word1. word2. word3. word4. word5.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

DACC	Departamento Acadêmico de Ciência da Computação
UFJF	Universidade Federal de Juiz de Fora
arm	ARM, originalmente Acorn RISC Machine, e depois Advanced RISC Machine, é uma família de arquiteturas RISC desenvolvida pela empresa britânica ARM Holdings
IDE	
x64	
x86	
aarch64	
ram	
GPU	
TPU	
CPU	
JVM	JVM (Java Virtual Machine) é uma máquina abstrata. É uma especificação que fornece um ambiente de tempo de execução no qual o bytecode do java pode ser executado. As JVMs estão disponíveis para muitas plataformas de hardware e software (ou seja, a JVM depende da plataforma).
IOT	
SBC	
BIOS	
TTL	

UART

WINE

Sumário

Introdução

Esta pesquisa foi baseada na crescente adoção de processadores arm[3] , estes processadores conseguem entregar uma eficiência energética muito superior a comumente utilizada nos computadores e servidores (arquitetura x86[6]). Esta arquitetura possui uma versão 64 bit, que hoje em dia é praticamente a única variante utilizada a x64[5] . Essa arquitetura é no geral apenas uma variante aditiva da x86[6] na qual são adicionadas varias instruções e vários suportes, o principal deles sendo suporte a comandos 64 bits.O mesmo pode ser dito para a arquitetura aarch64[7] ou arm64 que é uma variante aditiva da arm[3] ,essa não é ,como a x64[5] uma versão única,mas sim uma "denominação"das variantes e evoluções da arquitetura arm[3] com suporte a 64bit,as arquiteturas passaram a ser denominadas assim a partir da armv8,entretanto existem versões do arm[3] ,como o armv7l, que consegue trabalhar com instruções de 64bit,apesar de ser uma arquitetura 32 bits.

O foco da pesquisa foi feito em cima do uso de servidores arm[3] , que é baixo, apesar de totalmente possível e existente no mundo corporativo.Existem alguns servidores comerciais que utilizam a arquitetura arm[3] para seu funcionamento.Desta forma foi feita uma comparação na utilização desses processadores para a simulação de uma aplicação de banco de dados.Essa aplicação simula a utilização de forma realística de uma base de dados de uma locadora.

O cenário foi escolhido a partir de um esquema de banco de dados genérico da internet e foram utilizados os bancos de dados PostgreSQL e MariaDB,visto que são os bancos de dados de propósito geral mais utilizados atualmente.Foi preferido o MariaDB sobre o MySQL visto que não existe uma versão dele para a arquitetura arm[3] e ambos são basicamente o mesmo sistema.

foi criado um programa para a geração de dados realísticos baseados na biblioteca faker implementada na linguagem Python.Estes dados são gerados para cada pais especifico em idiomas e caracteres compatíveis com a região escolhida.Desta forma é plausível que estes dados,como nome,telefone,endereço e até mesmo usuário e senha sejam possíveis de existirem.Foi escolhido essa forma de inserção pois um preenchimento de dados totalmente randômicos e de tamanho fixo podem apresentar discrepâncias com o desempenho num

ambiente real de uso,afetando tanto o tempo,quanto carga dos processadores de forma negativa.Os dados utilizados em cada teste são exatamente os mesmos,com a diferença apenas da quantidade,simbolizando o uso em um ambiente real,sendo assim o benchmark se torna mais aplicável a realidade.

1 Fundamentação Teórica

1.1 arquiteturas

Por definição arquitetura de computador é um conjunto de circuitos eletrônicos padronizado associado a um conjunto de instruções de forma a simplificar a programação deles para que funcionem comandos diferentes do binário para a programação de um eletrônico,os compiladores utilizam esses conjuntos de instrução para que seja convertido o código de uma linguagem de programação para um binário de um programa, a arquitetura também define/limita varias propriedades do hardware,como quantidade máxima de ram[8] ,de disco,suporte ou não de saída de video,capacidades de rede e vários outros,mesmo que algumas dessas limitações possam ser contornadas utilizando variações da arquitetura chamadas microarquiteturas.

Uma micro arquitetura é quando é adicionado tanto um circuito eletrônico novo ao circuito original da arquitetura quanto apenas uma simplificação ou reorganização dos comandos originais de uma arquitetura,entretanto,numa micro arquitetura essas modificações são muito pequenas de forma a serem mais similares a arquitetura original do que uma nova arquitetura,dessa forma as micro arquiteturas podem ser consideradas updates de uma arquitetura,e quando são acumulados muitos desses updates,pode ser que seja gerada uma nova arquitetura,como foi o caso da arquitetura x86[6] para a arquitetura x64[5] ,onde a x64[5] foi um upgrade grande o bastante da x86[6] a ponto de ser considerado uma nova arquitetura,onde a principal e mais visível diferença entre esses dois é a mudança de 32bit(na x86[6]) para 64bit(no x64[5]).

As arquiteturas também podem ser definidas para coisas fora de CPU[11] ,elas podem estar definidas em GPU[9] , TPU[10] e vários outros módulos de hardware de um computador,inclusive existindo arquiteturas especiais que são aplicadas a nível de software,que não são necessariamente arquiteturas de computador,mas sim um tipo diferente de arquitetura,onde existem maquinas abstratas que simplificam a programação de uma linguagem para que ela funcione de forma mais compatível com varias maquinas de ar-

quiteturas de hardware diferentes,onde você faz otimizações de código na parte do código e da maquina virtual,como o caso da JVM[12] do java,em que a maquina virtual de cada arquitetura pode sofrer otimizações e isso faz com que ela funcione de forma melhor dependendo da maquina virtual em uma arquitetura e pior em outra,mas sem alterar o código,e ao mesmo tempo,outra maquina virtual pode funcionar pior na primeira arquitetura e melhor na segunda.

As arquiteturas não são limitadas apenas a esses previamente citados,as arquiteturas podem estar presentes todos os tipos de circuitos integrados,como os processadores de roteadores e aparelhos IOT[13] como lampadas e tomadas inteligentes.isso quer dizer que uma arquitetura não necessariamente é algo que precise de um hardware potente ou que só funcione ou exista em computadores,as arquiteturas são a forma como os algorítimos são interpretados no hardware,o que quer dizer que dês de que exista um software e um hardware que se comuniquem existe uma arquitetura e provavelmente houve uma conversão da linguagem de programação para a linguagem de maquina dessa arquitetura deste dispositivo

as arquiteturas de computador são definidas para hardwares específicos,mas o softwares não necessariamente precisam de ser funcionais apenas em uma única arquitetura,por mais que ela seja diferente da arquitetura de outro computador,isso por que os conjuntos de instruções podem ser diferentes mas suas funcionalidades gerais podem ser iguais,de forma que mesmo que uma arquitetura seja totalmente diferente de outra,os softwares de uma podem funcionar na outra,por mais que sejam necessárias algumas adaptações,algumas dessas adaptações podem ser tão grandes que as vezes é muito complexo essa adaptação de código,para esses casos,ou mesmo para se testar o código de uma arquitetura em outra,para se executar esses códigos sem ser necessária essa adaptação são usados programas chamados de emuladores ou simuladores estes programas funcionam como uma camada de compatibilidade entre a arquitetura real da maquina que está rodando e a arquitetura na qual o programa foi pensado em funcionar,entretanto esse processo pode acarretar em uma perda considerável de desempenho,podendo resultar em casos onde maquinas com 516 gigaflops sejam necessárias para se emular maquinas com 230 gigaflops,como no caso de um emulador do sistema de videogame playstation 3

utilizando o emulador rpcs3,e mesmo com essa ineficiência,esse emulador não tem 100% de compatibilidade com os softwares existentes na plataforma,de forma que nem todos os softwares dessa plataforma funcionam exatamente como deveriam,ou mesmo funcionam.por mais que ambas as maquinas rodem o mesmo kernel de sistema,o kernel Linux no caso, ainda assim a perda de desempenho é muito grande pois apesar de em teoria serem o mesmo sistema operacional a diferença de arquiteturas possui um peso muito maior do que o que o sistema operacional utilizado,por mais que pareça que não é o caso.

Esse é um exemplo de como mesmo com tudo para ser um cenário igual de utilização ou mesmo um cenário melhor ao se trocar uma arquitetura de um computador inúmeras adaptações devem ser feitas ou,como no caso do macos,criadas camadas de compatibilidade.Após o lançamento dos macbooks de 2020 com processador M1 que funcionam com a arquitetura aarch64[7] a apple lançou uma camada de compatibilidade dos softwares com arquitetura x64[5] para aarch64[7] chamado de rosetta2 esse software funciona parcialmente como um emulador,exceto que ele faz as adaptações num nível mais próximo do da máquina real e do sistema operacional nativo da máquina,resultando num desempenho muito superior a qualquer emulador existente,o rosetta2 funciona de forma análoga ao projeto WINE[18] do Linux que reinterpreta os programas windows para funcionarem no Linux,você tem uma pequena perda de desempenho por esse processo de reinterpretação em alguns casos,mas em outros essa perda é bem mais visível

As arquiteturas de computador podem variar em diversos fatores de uma para outra de forma que existam varias funcionalidades que não foram pensadas para uma arquitetura que existem em outras.existem também propósitos diferentes para diferentes arquiteturas,como o caso dos processadores arm[3] que foram pensados para entregar uma grande eficiência energética,enquanto os processadores x86[6] foram pensados para apresentarem grande poder de processamento

Existem alguns computadores com processador arm[3] que não são SBC[14] isso faz com que eles possuam varias possibilidades de upgrade,que não são possíveis nos computadores SBC[14] ,sendo assim existem algumas pequenas variações no funcionamento dos computadores mesmo dentro de uma mesma arquitetura que tenderia a seguir padrões

mais uniformes,entretanto uma peculiaridade tende a ser comum nos processadores arm[3] ,eles costumam apresentar uma GPU[9] integrada e algumas outras unidades de processamento especializadas que os processadores x86[6] costumam ter que ser adicionadas com chips externos,uma dessas unidades é o TPU[10] que ficou mais conhecida com o lançamento do windows 11 que o exigia na sua instalação por propósitos de segurança,o principal propósito da arquitetura arm[3] entretanto não é se diferenciar tanto da arquitetura x86[6] ,mas sim tornar computadores mais energeticamente eficientes,tanto que um computador doméstico comum utiliza de 200 a 300w por hora enquanto um computador raspberry pi 4,que é o computador arm[3] mais potente atualmente da marca mais popular,consome coisa de 15w hora,que é uma diferença absurda,principalmente se levar em conta que ambos tem a capacidade de utilizar os mesmos programas de trabalho,se considerarmos sistema operacional Linux e programas open source,tanto editores de texto quanto navegadores de internet quanto IDE[4] s de programação estão disponíveis para ambos e para um uso comum funcionam tão bem quanto num cenário real.

Como os processadores arm[3] começaram a ficar mais comuns visto que algumas fabricantes como a apple agora fabricam computadores baseados em arm[3] e a gigabyte agora possui versões de servidor com essa arquitetura de processador,isso faz com que seja cada vez mais fácil de se utilizar essa arquitetura já que se existem mais consumidores também vão existir mais programas feitos para rodar nessa arquitetura,e visto o quanto um computador com processador arm[3] economiza energia para entregar o mesmo poder de processamento de um outro com processador x86[6] , essa diferença pode ser muito benéfica para os vários tipos de empresa que utilizam servidores,já que isso pode significar um impacto considerável no consumo energético da empresa dependendo do quanto ele é utilizado a nível de processamento.

1.2 bancos de dados

banco de dados é um metodo de armazenamento de dados de forma estruturada para que sejam mais faceis de serem associados e de serem filtradas,elas também podem ser armazenadas de forma a economizar espaço de armazenamento dependendo da forma

como foi otimizado o banco de dados,os bancos de dados ainda possuem a capacidade de realizar algumas redundancias de segurança para o armazenamento de dados,de forma que a validação de um dado inserido possa ser feito a nivel de armazenamento de dados e não a nivel de programação,o que pode simplificar o desenvolvimento de um programa esses motivos são os principais de por que foram escolhidos os bancos de dados como alvo do benchmark realizado para essa comparação de arquiteturas.

os bancos de dados são onde a maioria dos dados de um sistema são salvos,esses dados são as informações necessárias para o funcionamento dos mais diversos tipos de programas tanto para ambientes comerciais quanto entreterimento ou mesmo comunicação,todos esses ramos de software utilizam bancos de dados de algum tipo para se salvar os dados e acelerar o acesso dos mesmos,isso quer dizer que os bancos de dados são extremamente importantes para qualquer tipo de aplicação.isso por si só já é motivo o bastante para se utilizar esse tipo de programa como base para os testes de desempenho entre as arquiteturas do ponto de vista de um servidor,enquanto as mais diversas aplicações podem rodar de um lado do servidor uma coisa é constante , as aplicações que rodam seguindo o modelo cliente-servidor utilizam algum tipo de banco de dados.
os tipos de bancos de dados analizados são os bancos de dados sql que são os mais genéricos,de forma que podem ser utilizados no máximo de aplicações diferentes possiveis,isso faz com que os bancos de dado sql sejam os melhores para serem simulados,um dos que foi analizado para ser testado foi o mongodb e o oracle,mas o mongodb não possui propósito geral e o da oracle não existe uma versão para arm[3] até o momento que o projeto foi pensado.

1.3 software de benchmark

o benchmark foi feito com 2 softwares principais um software de terceiros chamado dbbench que é um programa dedicado para o benchmark de bancos de dados e o segundo foi criado para inserir de forma padronizada os dados para o dbbench realizar os testes. o dbbench é um software para benchmark e teste de estresse de bancos de dados variados que utilizam arquivos de configuração para fazer variados tipos de testes.o software desenvolvido gera esses arquivos de configuração,sendo que dentro dos arquivos é possivel inserir as queries que serão testadas e o software criado faz isso,de forma a manter os mes-

mos dados pesquisados mas alterar a quantidade de consultas recorrentes ou de operações por segundo,o software criado gera a quantidade desejada de operações

1.4 biblioteca faker

a biblioteca faker é uma biblioteca conhecida para a geração de mockdata,que são dados gerados randomicamente para se testar a capacidade de um algoritimo lidar com dados,essa biblioteca comumente é usada na etapa de testes de um algoritimo,seja para testes de segurança,para proteger de bots de criação de contas ou algo do tipo,quanto para testes de estresse, quanto para qualquer tipo de teste que dependa de dados "realisticos". essa biblioteca foi selecionada pois possui facil utilização,documentação abrangente e de facil compreenção e suporte para multiplos idiomas.apesar da bibliotea apenas suportar todos os seus tipos de geração mais completos na localização dos EUA , dentre esses está um tipo que corresponde a todo um perfil de usuario,contendo de endereço até a senha e podendo ser uma senha fraca ou forte.

para propósitos de teste foram gerados apenas dados na localização do Brasil apenas por conveniencia,e também pois varios dos dados utilizados são os mais simples ,de modo que não são necessários os tipos mais complexos que existem apenas para certos idiomas.

2 desenvolvimento

o desenvolvimento do software foi feito utilizando varios metodos de analize de log para a agilização do debug,alem disso possibilitou que os dados gerados pudessem ser mais facilmente conferidos durante o desenvolvimento fazendo com que tivesse,apesar de uma geração randômica,uma filtragem humana dos tipos de dados gerados.
durante esse capítulo serao descritos os procedimentos mais importantes das etapas de desenvolvimento e funcionamento do software,de forma que fique o mais simples de entender possivel.

2.1 geração de dados

os dados gerados para os testes foram gerados em 2 etapas:

- valores brutos em sqlite
- valores trabalhados em csv

essas etapas funcionam da seguinte forma:
de inicio os dados são gerados em sqlite pelo fato de , caso algum problema ocorra e a maquina que estava gerando os dados seja abruptamente desligada,os dados ja gerados ja terão sido salvos no banco de dados sqlite,fazendo com que nenhum progresso tenha sido perdido,já que a etapa de geração de dados é a etapa mais demorada entre todas as etapas do algoritimo de criação dos dados de testes.
o codigo consegue gerar os dados de 3 formas,apenas a ultima é realmente utilizada. as outras duas foram feitas para propósito de testes

- gerar uma quantidade x de dados de uma tabela expecifica
- gerar uma quantidade x de cada tipo de dado para cada tabela do bd final
- gerar uma quantidade aleatória de cada tipo de dado para cada tabela do bd final ate atingir o total de operações informadas

todos esses tipos de geração sao feitos pela mesma função,e sai alterados pelos parâmetros passados para ela,de forma que ela prioriza os parametros referentes ao 3 tipo de geração.
fora os parametros relacionados aos tipos de geracao informados existem os parametros relacionados a seção 1.4 e uma lista que define quais tipos de operação,descritos em seção 2.2 ,que defines,caso nao seja vazia,quais os tipos de dados que serao gerados,isso foi util pois foi utilizado o tipo de criacao de dados de inserção na primeira etapa,antes de gerar oa dados dos outros tipos , juntamente com mais dados de inserção.
devido a limitações ,intencionais,o valor total de operações inserido na a cada operação deve considerar a quantidade da operação anterior,pois o algoritimo não irá verificar nada alem da quantidade total de elementos cadas trados no sqlite,ou seja,no caso dos testes feitos,foram gerados 50.000 dados de inserção,e posteriormente foram pedidos 5.000.000 dados randomicos,o que resultou em 50.000 dados de inserção seguidos de 4.950.000 dados randomicos gerados.

2.2 tipos de dados gerados

para os testes foram selecionadas apenas algumas das possiveis operações de um banco de dados,sendo a maioria delas as operações mais utilizadas de um banco de dados:

- inserção de um novo dado
- leitura completa de todos os dados de uma tabela
- busca de elementos filtrados em determinada tabela
- busca de apenas alguns dados de elementos filtrados em determinada tabela
- edição de elementos
- deleção de elementos filtrados

foram selecionados essas operações devido ao quanto eles são utilizados,exceto pela leitura completa de uma tabela e pela busca filtrada com apenas a seleção de algumas colunas essas operações constituem um crud basico de um bd sql.
esses tipos de operação antes de serem geradas passam por varios processos de tratamento

de erro para se certificar que não houve nenhuma dependencia que não foi gerada,como gerar uma cidade sem existir um pais cadastrado,esse foi um dos motivos de ter sido utilizado um arquivo sqlite ao inves de outro metodo de armazenamento,poder executar essa consulta de dependencia de forma rpida e apenas retornar indices validos para associações de tabela.

2.3 alimentação do algoritimo

o algorítimo funciona de forma que qualquer banco de dados que seja informado num arquivo json seguindo determinado padrão pode ter seus dados gerados,fazendo com que o algoritimo de geração de dados possa ser usado para qualquer banco de dados,o padrão é composto por uma tag para cada bd,sendo o nome do bd,e dentro dela uma estrutura de json com uma tag sendo o nome da coluna e seu conteudo um array de string contendo primeiro o tipo de dado,seguindo os padrões do algoritimo,e os outros valores sendo dados adicionais para a geração,sendo que a maioria das colunas do banco de dados inserido apenas possui um elemento

dentro do algoritimo existem varios tipos de dados aceitaveis,tais como id,nome,associação e timestamp,cada um deles possui um tipo bem definido pelo seu nome,mas o unico que vale a pena citar seu funcionamento é o associação,como descrito em seção 2.5 existe uma tabela do sqlite contendo as quantidades de elementos associados a cada tabela do banco de dados final,o tipo de associação vai pegar esse valor e usar uma função de seleção randômica para que seja escolhido um elemento de id existente no intervalo descrito,caso não exista nenhum elemento dessa tabela,por um tratamento de erro é gerado um elemento para ela,para assim poder fazer a associação funcionar no novo elemento associado que foi criado.

os unicos dados que não são passados para o algoritimo funcionar são o pais que deve ser gerado os dados,de acordo com seção 1.4, e a quantidade de dados que devem ser geradas,ambos sendo necessários de serem inseridos na chamada da função. os outros dados relevantes referentes ao funcionamento da chamada da função estão em seção 2.1

2.4 ambiente de desenvolvimento

o ambiente utilizado foi dividido em 2 partes,programação local e remota,na programação remota as execuções eram feitas num servidor baseado em raspberry pi,e na programação local foram feitas na propria maquina local,sendo que a programação remota se provou bem util quando houve a necessidade de troca de computador ou sistema operacional,a programação remota ainda facilitou a implementação de um servidor unificado de analize de log

foram utilizados visual studio code e dbeaver para o desenvolvimento da aplicação principal,ainda foi utilizado uma implementação da suite ELK(elasticsearch ,logstash e kibana) de analize de log,os dois primeiros foram escolhidos dentre outros motivos por serem open-source e estarem disponiveis tanto para linux quanto windows,visto que ambos sistemas operacionais foram utilizados para o desenvolvimento,de acordo com a necessidade no momento.

para o controle de verão foi utilizado o git,deixando registrado todo o histórico de alterações do programa,o que se mostrou bem util para o rastreio de erros ocorridos durante o longo tempo de desenvolvimento do codigo.

para os testes iniciais do codigo foram utilizados containers docker não limitados durante a etapa de implmentação dos scripts do dbbench,que demandam a existencia dos servidores dos bancos de dados rodando,ao contrário do restante do desenvolvimento da aplicação,que não interagiu diretamente com os bancos de dados.

a geração do bd inicial foi feito completamente em um raspberry pi 4 com um pequeno overclock,essa geração foi feita durante alguns dias,visto que o raspberry pi ,de acordo com testes de velocidade feitos seção 3.1,esse bd constitui-se de todas as operações feitas durante o teste de estresse

2.5 sqlite

o banco de dados do sqlite foi projetado para ser totalmente maleavel e modular,de forma que não teriam que ser geradas varias tabelas para os varios tipos de dados do benchmark.Foi pensado no seguinte metodo para se facilitar o desenvolvimento,uma tabela de indices de total de elementos de cada tabela,sendo cada elemento composto apenas pelo nome da tabela e pelo total de elementos cadastrados,a outra tabela é um pouco mais

complexa.

- a tabela de operações a ser executado é constituido de uma coluna inteira para o tipo de operação que será realizada,de acordo com seção 2.2
- uma coluna é um string contendo o nome do banco de dados que será executada a operação
- uma coluna inteira para ,se for necessário, conter o id no bd do elemento trabalhado na operação,no caso de uma inserção é o id do novo elemento por exemplo
- uma coluna text,nessa coluna serão inseridos valores adicionais necessários para a execução da operação,como os parametros de quais colunas devem ser atualizadas em um update,aqui os dados inseridos são salvos em formato json para facilitar o trabalho com a linguagem python,visto que existe uma conversão direta de string json para o tipo dictionary do python
- seguindo a mesma idéia da coluna anterior ,essa coluna também é uma coluna text onde são inseridos dados em formato json,esses dados são os dados obrigatórios de qualquer operação

dessa forma independente se é apenas uma operação de listagem completa,que só necessita de ter preenchida a coluna com o nome do bd e a coluna com o tipo de operação ou se for uma operação de update onde todos os campos podem estar preenchidos, o banco de dados sqlite consegue lidar de forma rapida e segura com qualquer uma das operações e dados gerados pelo algoritimo

3 testes

os testes foram realizados utilizando um orangepi pc + e um computador de mesa . o orange pi é um SBC ARM baseado no processador allwinner h3 com 3 usb 2.0 , 1 gb de memória ram ddr3 , uma porta de rede 10/100 e wifi bgn,essa é relativamente antiga e seu processador é um 4-core de 1.3ghz no clock máximo,é um processador relativamente bom mas não é bom o bastante para substituir um computador atual,mas consegue funcionar de forma satisfatória como servidor doméstico ,visto que seu processador é bom o bastante para operações simplificadas e poucos acessos,mas quando se tratam de muitos acessos ele pode não ser potente o bastante para aguentar. o computador de mesa é um dual core intel com 4 gb de memória ram ddr3, 4 usb 2.0 porta de rede gigabit e algumas portas sata2,entretanto essas portas sta não serão usadas já que o propósito é manter as 2 maquinas o mais próximas em relação a hardwaree possivel outros metodos que serão usados para manter as maquinas mais similares serão limitar o clock de ambos para 1.2ghz,que é o clock mais estável para o orange pi ,a memória usada será limitada a 750 mb para o stack do docker e tanto o sistema operacional quanto os dados do docker serão salvos em cartões de memória microsd classe 10 com limitação de acesso de 10 MB/s de escrita e leitura , no orangepi será usado o leitor da própria placa para conectar o microsd e no pc será usado um adaptador usb. alem disso serão usadas versões do sistema debian,no orangepi o armbian e no pc o prórpio debian padrão,ambos na versão buster

3.1 testes de tempo

foi feito um teste aditivo que testará 30 ciclos com adição de 100 em 100 elementos e cada ciclo,foram feitos um teste de 4 sub ciclos internos para cada um dos 30 ciclos,isso para ver o quanto de tempo é gasto em média para casos com muitos ou poucos dados gerados,sendo assim podendo gerar um valor de base de valor minimo gasto obrigatóriamente para cada interação.

os testes dessa vez foram modificados de forma a salvar os dados em um arquivo para consulta futura,que estão em anexo,os testes dessa vez foram realizados em um pc arm64 e um amd64,os dados resultaram em valores de tempo consistente em relação a diferença

de frequencias das maquinas,sendo assim se existe um valor de perda entre as arquiteturas para esse teste é um valor irrisório,sendo que o pc amd64 apresentou testes cerca de 2 vezes mais rapidos e seu clock é aproximadamente o dobro do arm64,esses valores foram dados em relação ao tempo gasto por elemento em cada interação

4 Trabalhos Relacionados

5 Metodologia

6 Resultados

6.1 Resultados do Método

7 Conclusão

Anexos