Administração de Bases de Dados

Engenharia Informática – Universidade do Minho

Trabalho Prático – 2023/2024

Os grupos de trabalho devem ser constituídos por 5 (cinco) elementos, todos inscritos na Unidade Curricular. O resultado do trabalho é um relatório escrito. O relatório deve omitir considerações genéricas sobre as ferramentas utilizadas, focando a apresentação e justificação dos objetivos atingidos. A entrega do relatório é feita na área da Unidade Curricular no *e-Learning*. O relatório deve identificar também o nome e número de todos os elementos do grupo. A data limite é 2024/05/10.

1 Contexto

O trabalho prático consiste na configuração, otimização, e avaliação de um *benchmark* disponível na plataforma *eLearning*, baseado num pequeno excerto do dataset do StackOverflow e contendo operações transacionais e analíticas.

A componente transacional simula operações de *backend* do StackOverflow, considerando os seguintes procedimentos:

- *NewQuestion* (**NQ**) – criar uma nova pergunta;
- *NewAnswer* (**NA**) – criar uma nova resposta;
- *Vote* (**VT**) – votar numa pergunta/resposta;
- *QuestionInfo* (**QI**) – obter a informação geral de uma pergunta;
- *PostPoints* (**PP**) – obter a quantidade de pontos de uma pergunta/resposta;
- *UserProfile* (**UP**) – obter o perfil de um utilizador;
- *Search* (**SR**) – pesquisar perguntas pelo título;
- *LatestByTag* (**LT**) – obter as perguntas mais recentes por *tag*.

As quatro interrogações analíticas simulam operações estatísticas e de *data warehousing* sobre o dataset. Consideram-se que estas são executadas em dois ambientes distintos:

- Diretamente a partir da base de dados (PostgreSQL), sendo que é esperado que o resultado considere informação em tempo real;
- A partir de um *snapshot* da base de dados (Apache Spark), sendo que considera-se que a informação permanece estática durante longos períodos.

Estas interrogações definem os seguintes argumentos:

- **Q1** – o limite inferior de *creationdate* (valor predefinido = 6 meses);
- **Q2** – o limite inferior de *creationdate* (valor predefinido = 5 anos) e o intervalo de cada *bucket* (valor predefinido = 5000);

- **Q3** – o limite inferior de *count(*)* (valor predefinido = 10);
- **Q4** – o tamanho de cada *bucket* (valor predefinido = 1 minuto).

Apesar dos argumentos estarem definidos de forma estática, deve-se considerar que podem ser modificados para valores arbitrários.

Para a otimização da componente analítica a partir da base de dados, deve-se considerar ainda os seguintes pressupostos sobre a atualização dos dados:

- *Answers* - entradas podem ser adicionadas ou removidas; campos *lasteditoruserid*, *lastactivitydate*, *lasteditdate*, e *body* podem ser modificados;
- *Badges* - entradas podem ser adicionadas ou removidas;
- *Comments* - entradas podem ser adicionadas ou removidas; campo *text* pode ser modificado;
- *Questions* - entradas podem ser adicionadas ou removidas; campos *lasteditoruserid*, *acceptedanswerid*, *title*, *closeddate*, *lastactivitydate*, *lasteditdate*, e *body* podem ser modificados;
- *QuestionsLinks* - entradas podem ser adicionadas ou removidas;
- *QuestionsTags* - entradas podem ser adicionadas ou removidas;
- *Tags* - tabela estática;
- *Users* - entradas podem ser adicionadas ou removidas; campos *displayname*, *aboutme*, *websiteurl*, *location*, *lastaccessdate*, e *reputation* podem ser modificados;
- *Votes* - entradas podem ser adicionadas;
- *VotesTypes* - tabela estática.

2 Objetivos

A configuração de referência é uma máquina virtual na *Google Cloud* do tipo N2, 8 vCPUs, 16 GB RAM, disco Persistent SSD 500 GB[1], e sistema operativo Linux Ubuntu 22.04 LTS x86/64. A carga transacional deve considerar 16 clientes, enquanto que a carga analítica deve considerar apenas 1. Usando esta configuração, devem:

1. Otimizar o desempenho da carga transacional;
2. Otimizar o desempenho das interrogações analíticas no PostgreSQL;
3. Adaptar e otimizar as interrogações analíticas no Spark.

A realização destes objetivos deve considerar **redundância** (e.g., índices, materialização), **paralelismo**, **parâmetros de configuração**, e até **código SQL/Java**. Para a componente transacional, devem primeiro otimizar através da redundância e/ou código e só depois considerarem a otimização através dos parâmetros de configuração, de modo a obter diferenças mais significativas neste último ponto.

[1] Ao escolher o disco deverá ser necessário selecionar "Request quota adjustment" na altura da criação da máquina.

3 Código e Dados

O código que implementa as operações descritas está disponível na plataforma *e-Learning*. São disponibilizadas duas versões dos dados:

- Uma versão reduzida dos dados para agilizar testes preliminares com a componente analítica em Spark está disponível em `https://storage.googleapis.com/abd24/data-lite.zip`. Esta versão não deve ser utilizada para obter resultados finais.
- A versão completa dos dados a utilizar para obter os resultados finais a incluir no relatório está disponível em `https://storage.googleapis.com/abd24/data.zip`.

O dataset original, que não é necessário para a resolução deste trabalho, e algumas informações complementares sobre o seu conteúdo e estrutura está disponível em `https://meta.stackexchange.com/questions/2677`.

4 Notas

- Como medidas de desempenho da carga transacional deve considerar-se principalmente o **debito máximo** atingível. Nas operações analíticas deve considerar-se o **tempo de resposta** médio de várias execuções.
- Poderão modificar as interrogações SQL e o código Java. Devem explicar no relatório em que medida essas alterações preservam o funcionamento da aplicação original.
- Para o PostgreSQL, devem considerar que cada *workload* corre separadamente, não sendo necessário executar a carga transacional e analítica de forma concorrente. Contudo, devem verificar que impacto a criação de redundância num *workload* tem no outro.
- Para a leitura dos dados CSV no Spark funcionar de forma esperada, devem fornecer os argumentos `header=True`, `inferSchema=True`, `multiLine=True`, e `escape='\"'` à função `spark.read.csv`.
- Em cada um dos objetivos, poderão não ter tempo para considerar todas as otimizações possíveis nas suas várias combinações. Devem focar-se nas que consideram mais prometedoras e que mais vos interessam, justificando no relatório essas opções.
- No caso de não obterem melhorias de desempenho, devem explicar porque é que a configuração de referência já era ótima. Por outro lado, a simples apresentação de uma melhoria de desempenho, não justificada, não é muito interessante.
- A utilização de ferramentas de monitorização e diagnóstico do PostgreSQL, do Spark, e do sistema operativo (e.g., pgbadger, psutil, iostat) valoriza o trabalho.
- Devem considerar uma instalação nativa para o PostgreSQL e uma instalação virtualizada (Docker) para o cluster Spark.
- A automatização da instalação e execução do *benchmark* permitirá obter resultados em maior quantidade e uma análise mais profunda, valorizando o trabalho.
- Devem também procurar estratégias para poupar recursos na *Google Cloud*, por exemplo, armazenando os dados em *Cloud Storage* e reutilizando-os[2], em vez de re-executar o `load.py`, e/ou explorando o uso de *Machine Images*. Considerem as opções mais baratas na *Google Cloud* (i.e., regiões nos EUA, instâncias *spot*, ...) e de **não deixar máquinas virtuais ativas a consumir recursos desnecessariamente!**

[2]Para PostgreSQL, ver `pg_dump` (`https://www.postgresql.org/docs/16/app-pgdump.html`) e `pg_restore` (`https://www.postgresql.org/docs/16/app-pgrestore.html`).