

Artigo

[Alexey Maslov](#) · Fev. 1, 2021 12min de leitura

## Processamento Paralelo de Dados Multimodelos no InterSystems IRIS e Caché

Como todos nós sabemos, o InterSystems IRIS possui uma ampla gama de ferramentas para melhorar a escalabilidade dos sistemas de aplicação. Em particular, muito foi feito para facilitar o processamento paralelo de dados, incluindo o uso de paralelismo no processamento de consultas SQL e o recurso mais chamativo do IRIS: o sharding. No entanto, muitos desenvolvimentos maduros que começaram no Caché e foram transportados para o IRIS usam ativamente os recursos de multimodelos deste DBMS, que são entendidos como permitindo a coexistência de diferentes modelos de dados dentro de um único banco de dados. Por exemplo, o banco de dados [HIS qMS](#) contém modelos de dados semânticos relacionais (registros médicos eletrônicos), relacionais tradicionais (interação com PACS) e hierárquicos (dados de laboratório e integração com outros sistemas). A maioria dos modelos listados é implementada usando a ferramenta qWORD do [SP.ARM](#) (um mini-DBMS que é baseado no acesso direto a globais). Portanto, infelizmente, não é possível usar os novos recursos de processamento de consulta paralela para escalonamento, uma vez que essas consultas não usam o acesso IRIS SQL.

Enquanto isso, conforme o tamanho do banco de dados cresce, a maioria dos problemas inerentes a grandes bancos de dados relacionais tornam-se adequados para os não relacionais. Portanto, esse é o principal motivo pelo qual estamos interessados no processamento paralelo de dados como uma das ferramentas que podem ser usadas para escalonamento.

Neste artigo, gostaria de discutir os aspectos do processamento paralelo de dados com os quais tenho lidado ao longo dos anos ao resolver tarefas que raramente são mencionadas em discussões sobre Big Data. Vou me concentrar na transformação tecnológica de bancos de dados, ou melhor, em tecnologias de transformação de bancos de dados.

Não é segredo que o modelo de dados, a arquitetura de armazenamento e a plataforma de software e hardware geralmente são escolhidos nos estágios iniciais do desenvolvimento do sistema, geralmente quando o projeto ainda está longe de ser concluído. No entanto, algum tempo passará, e é bastante comum quando vários anos após a implantação do sistema, os dados precisam ser migrados por um motivo ou outro. Aqui estão apenas algumas das tarefas comumente encontradas (todos os exemplos são tirados da vida real):

1. Uma empresa está planejando se internacionalizar e seu banco de dados com codificação de 8 bits deve ser convertido para Unicode.
2. Um servidor desatualizado está sendo substituído por um novo, mas é impossível transferir perfeitamente os registros (journals) entre os servidores (usando o espelhamento ou os recursos do sistema Shadowing IRIS) devido a restrições de licenciamento ou falta de recursos para atender às necessidades existentes, como, por exemplo, quando você está tentando resolver uma tarefa (1).
3. Você descobre que precisa alterar a distribuição dos globais entre os bancos de dados, por exemplo, movendo um grande global com imagens para um banco de dados separado.

Você deve estar se perguntando o que há de tão difícil nesses cenários. Tudo o que você precisa fazer é parar o sistema antigo, exportar os dados e importá-los para o novo sistema. Mas se você estiver lidando com um banco de dados com várias centenas de gigabytes (ou mesmo vários terabytes) de tamanho e seu sistema estiver funcionando no modo 24x7, você não conseguirá resolver nenhuma das tarefas mencionadas usando as ferramentas IRIS padrão.

### Abordagens básicas para paralelização de tarefas

## Paralelização "Vertical"

Suponha que você possa dividir uma tarefa em várias tarefas de componentes. Se você tiver sorte, descobrirá que pode resolver algumas delas em paralelo. Por exemplo,

- Preparar dados para um relatório (cálculos, agregação de dados...)
- Aplicar regras de estilo
- Imprimir relatórios

todas podem ser executadas ao mesmo tempo para vários relatórios: um relatório ainda está em fase de preparação, outro já está sendo impresso ao mesmo tempo, etc. Essa abordagem não é nova. É utilizada desde o advento do processamento de dados em lote, ou seja, há 60 anos. No entanto, embora não seja um conceito novo, ainda é bastante útil. Contudo, você só perceberá um efeito de aceleração perceptível quando todas as subtarefas tiverem um tempo de execução comparável, e nem sempre é esse o caso.

## Paralelização "Horizontal"

Quando a ordem das operações para resolver uma tarefa consiste em iterações que podem ser executadas em uma ordem arbitrária, elas podem ser executadas ao mesmo tempo. Por exemplo:

- Pesquisa contextual no global:
  - Você pode dividir o global em subglobais (ordenar pelo primeiro índice).
  - Pesquisar separadamente em cada um deles.
  - Reunir os resultados da pesquisa.
- Transferir o global para outro servidor por meio de um soquete ou ECP:
  - Dividir o global em partes.
  - Passar cada um deles separadamente.

Recursos comuns dessas tarefas:

- Processamento idêntico em subtarefas (até compartilhar os mesmos parâmetros).
- A correção do resultado final não depende da ordem de execução dessas subtarefas.
- Há uma conexão fraca entre as subtarefas e a tarefa "pai" apenas no nível de relatório de resultados, onde qualquer pós-processamento necessário não é uma operação que consome muitos recursos.

Esses exemplos simples sugerem que o paralelismo horizontal é natural para tarefas de transformação de dados e, de fato, é. A seguir, enfocaremos principalmente nesse tipo de processamento paralelo.

## Paralelização "Horizontal"

### Uma das abordagens: MapReduce

[MapReduce](#) é um modelo de computação distribuída que foi introduzido pelo [Google](#). Ele também é usado para executar essas operações, e para processar grandes quantidades de informações ao mesmo tempo. Implementações populares de software livre são construídas em uma combinação do [Apache Hadoop](#) and [Mahout](#).

Etapas básicas do modelo: Map, distribuição de tarefas entre manipuladores, o processamento atual e Reduce combinar os resultados do processamento.

Para o leitor interessado que queira saber mais, posso recomendar a série de artigos de Timur Safin sobre a abordagem dele para a criação da ferramenta MapReduce em IRIS/Caché, que começa com [Caché MapReduce - uma introdução ao BigData e ao conceito MapReduce \(Parte 1\)](#).

Observe que, devido à "capacidade inata" do IRIS de gravar dados no banco de dados rapidamente, a etapa de Reduce, via de regra, acaba sendo trivial, como na [versão distribuída do WordCount](#). Ao lidar com tarefas de transformação de banco de dados, isso pode ser completamente desnecessário. Por exemplo, se você usou manipuladores paralelos para mover um grande global para um banco de dados separado, então, não precisamos de mais nada.

## Quantos servidores?

Os criadores de modelos de computação paralela, como MapReduce, geralmente o estendem a vários servidores, os chamados nós de processamento de dados, mas em tarefas de transformação de banco de dados, um desses nós geralmente é suficiente. O fato é que não faz sentido conectar vários nós de processamento (por exemplo, via Enterprise Cache Protocol (ECP)), uma vez que a carga da CPU necessária para a transformação dos dados é relativamente pequena, o que não pode ser dito sobre a quantidade de dados envolvidos no processamento. Nesse caso, os dados iniciais são usados uma vez, o que significa que você não deve esperar nenhum ganho de desempenho do cache distribuído.

A experiência mostra que geralmente é conveniente usar dois servidores cujas funções são assimétricas. Para simplificar um pouco o conceito:

- O banco de dados de origem é montado em um servidor (DB de origem).
- O banco de dados convertido é montado no segundo servidor (DB de destino).
- O processamento de dados paralelos horizontais é configurado apenas em um desses servidores. Os processos operacionais neste servidor são também os processos mestres.
- Os processos em execução no segundo servidor, são, por sua vez, os processos escravos. Quando você usa o ECP, esses são os processos do sistema DBMS (ECPSrvR, ECPSrvW e ECPWork) e, ao usar um mecanismo de transferência de dados orientado por soquete, esses são os processos filhos das conexões TCP.

Podemos dizer que esta abordagem de distribuição de tarefas combina paralelismo horizontal (que é usado para distribuir a carga dentro do servidor mestre) com paralelismo vertical (que é usado para distribuir "responsabilidades" entre os servidores mestre e escravo).

## Tarefas e ferramentas

Vamos considerar a tarefa mais geral de transformar um banco de dados: transferir todos ou parte dos dados do banco de dados de origem para o banco de dados de destino, enquanto possivelmente executa algum tipo de recodificação de globais (isso pode ser uma mudança de codificação, mudança de agrupamento, etc.). Nesse caso, os bancos de dados antigos e novos são locais em servidores de banco de dados diferentes. Vamos listar as subtarefas a serem resolvidas pelo arquiteto e desenvolvedor:

1. Distribuição de funções entre servidores.
2. Escolha do mecanismo de transmissão de dados.
3. Escolha da estratégia de transferência de globais.
4. Escolha da ferramenta para distribuição de tarefas entre vários processos.

Vamos 'dar uma olhada' neles.

## Distribuição de funções entre servidores

Como você já está familiarizado, mesmo que o IRIS esteja sendo instalado com suporte a Unicode, ele também pode montar bancos de dados de 8 bits (locais e remotos). No entanto, o oposto não é verdadeiro: a versão de 8 bits do IRIS não funcionará com um banco de dados Unicode e haverá erros inevitáveis de <WIDE CHAR> se você tentar fazer isso. Isso deve ser levado em consideração ao decidir qual dos servidores - de origem ou de destino - será o mestre se a codificação de caracteres for alterada durante a transformação dos dados. Contudo, é impossível aqui decidir sobre uma solução final sem considerar a próxima tarefa, que é a:

## Escolha do mecanismo de transmissão de dados

Você pode escolher uma das seguintes opções aqui:

1. Se as versões de licença e DBMS em ambos os servidores permitirem o uso do ECP, considere o ECP como um transporte.
2. Caso contrário, a solução mais simples é lidar com os dois bancos de dados (de origem e de destino) localmente no sistema de destino. Para fazer isso, o arquivo de banco de dados de origem deve ser copiado para o servidor apropriado por meio de qualquer transporte de arquivo disponível, o que, é claro, levará mais tempo (para copiar o arquivo de banco de dados na rede) e espaço (para armazenar uma cópia do arquivo do banco de dados).
3. Para evitar perder tempo com a cópia do arquivo (pelo menos), você pode implementar seu mecanismo de troca de dados entre os processos do servidor por meio de um soquete TCP. Essa abordagem pode ser útil se:
  - O ECP não pode ser usado por algum motivo, por exemplo, devido à incompatibilidade das versões do DBMS servindo os bancos de dados de origem e destino (por exemplo, o DBMS de origem é de uma versão muito legada).
  - Ou: É impossível impedir que os usuários trabalhem no sistema de origem e, portanto, a modificação de dados no banco de dados de origem que ocorre no processo de transferência deve ser refletida no banco de dados de destino.

Minhas prioridades ao escolher uma abordagem são bastante evidentes: se o ECP está disponível e o banco de dados de origem permanece estático enquanto é transferido – 1, se o ECP não está disponível, mas o banco de dados ainda está estático – 2, se o banco de dados de origem é modificado – 3. Se combinarmos essas considerações com a escolha do servidor mestre, podemos produzir a seguinte matriz de possibilidade:

O banco de dados de origem está estático durante a transmissão?	O protocolo ECP está disponível?	Localização do banco de dados de origem	Sistema mestre
Sim	Sim	Remoto, no sistema de destino	Destino
Sim	Não	Local (cópia) do sistema de destino	Destino
Não	Não importa, pois usaremos nosso mecanismo para transferir dados por soquetes TCP.	Local (original) no sistema de origem	Origem

## Escolha da estratégia de transferência de globais

À primeira vista, pode parecer que você pode simplesmente passar os globais um a um lendo o Diretório Global. No entanto, os tamanhos dos globais no mesmo banco de dados podem variar muito: Recentemente, encontrei uma situação em que os globais em um banco de dados de produção variavam entre 1 MB e 600 GB. Vamos imaginar que temos os processos de trabalho nWorkers à nossa disposição, e há pelo menos um ^Big global para o qual é verdade:

$\text{Size}(\text{^Big}) > (\text{Summary Size of All ^Globals}) / n\text{Workers}$

Então, não importa o quão bem-sucedida a tarefa de transferência dos globais restantes seja distribuída entre os processos de trabalho, a tarefa que acaba sendo atribuída à transferência do ^Big global permanecerá ocupada pelo restante do tempo alocado e provavelmente só terminará a tarefa muito depois dos outros processos terminarem de processar o restante dos globais. Você pode melhorar a situação pré-ordenando os globais por tamanho e iniciando o processamento com os maiores primeiro, mas nos casos em que o tamanho de ^Big se desvia significativamente do valor médio para todos os globais (que é um caso típico do banco de dados MIS qMS):

$\text{Size}(\text{^Big}) \gg (\text{Summary Size of All ^Globals}) / n\text{Workers}$

Esta estratégia não o ajudará muito, pois leva inevitavelmente a um atraso de muitas horas. Consequentemente, você não pode evitar a divisão de grandes globais em partes para permitir seu processamento usando vários processos paralelos. Desejo enfatizar que esta tarefa (número 3 na minha lista) acabou sendo a mais difícil entre as outras que estão sendo discutidas aqui, e levou a maior parte do meu tempo (ao invés da CPU!) para resolvê-la.

## Escolha da ferramenta para distribuição de tarefas entre vários processos

A maneira como interagimos com o mecanismo de processamento paralelo pode ser descrita da seguinte maneira:

- Criamos um pool de processos de trabalho em segundo plano.
- Uma fila é criada para este pool.
- O processo iniciador (vamos chamá-lo de gerente local), ter um plano que foi preparado com antecedência na etapa 3, e colocar as unidades de trabalho na fila. Como regra, a unidade de trabalho compreende o nome e os argumentos reais de um determinado método de classe.
- Os processos de trabalho recuperam unidades de trabalho da fila e executam o processamento, que se resume a chamar um método de classe com os argumentos reais que são passados para as unidades de trabalho.
- Após receber a confirmação de todos os processos de trabalho de que o processamento de todas as unidades de trabalho enfileiradas foi concluído, o gerente local libera os processos de trabalho e finaliza o processamento, se necessário.

Felizmente, o IRIS fornece um excelente mecanismo de processamento paralelo que se encaixa perfeitamente neste esquema, que é implementado na classe %SYSTEM.WorkMgr. Vamos usá-lo em um exemplo de execução que exploraremos em uma série planejada de artigos.

No próximo artigo, pretendo me concentrar em esclarecer a solução para a tarefa número 3 com mais detalhes.

No terceiro artigo, que aparecerá se você mostrar algum interesse na minha escrita, falarei sobre as nuances da resolução da tarefa número 4, incluindo, em particular, sobre as limitações de %SYSTEM.WorkMgr e as formas de superá-las.

[#Big Data](#) [#DevOps](#) [#Caché](#) [#InterSystems IRIS](#)

---

URL de origem: <https://pt.community.intersystems.com/post/processamento-paralelo-de-dados-multimodelos-no-intersystems-iris-e-cach%C3%A9>