Melhorando a Experiência do usuário com Paralelismo e Cache

Não otimizado vs Otimizado

Sem paralelismo e cache

Cada operação espera a outra terminar, criando filas e travamentos.
Sem cache, dados são buscados do servidor todas as vezes, mantendo o tempo de resposta alto.
E pode causar sobrecarga na api (servidor).

Com paralelismo e cache

Paralelismo: Várias operações acontecem ao mesmo tempo, sem bloqueios.
Cache: Os dados são retornados instantaneamente, reduzindo requisições e acelerando o PDV.
Aliviando o servidor de requisições desnecessárias.

O Paralelismo no PDV

Múltiplas Requisições

O PDV React pode "disparar" várias tarefas de rede ao mesmo tempo: buscar dados do produto e sem esperar a resposta de cada uma.

UI Sempre Responsiva

Graças as requisições “disparadas” por de baixo dos panos e ao "Concurrent Mode" do React 18, operações pesadas (como filtrar uma lista longa) podem ser pausadas para processar o input do usuário (como digitar em um campo de busca), dando a sensação de fluidez total.

Graças a estrutura fibers do React 18 que consegue priorizar ou bloquear renderizações específicas, e a hooks (useTransition, useDeferredValue) para controle de forma explicita.

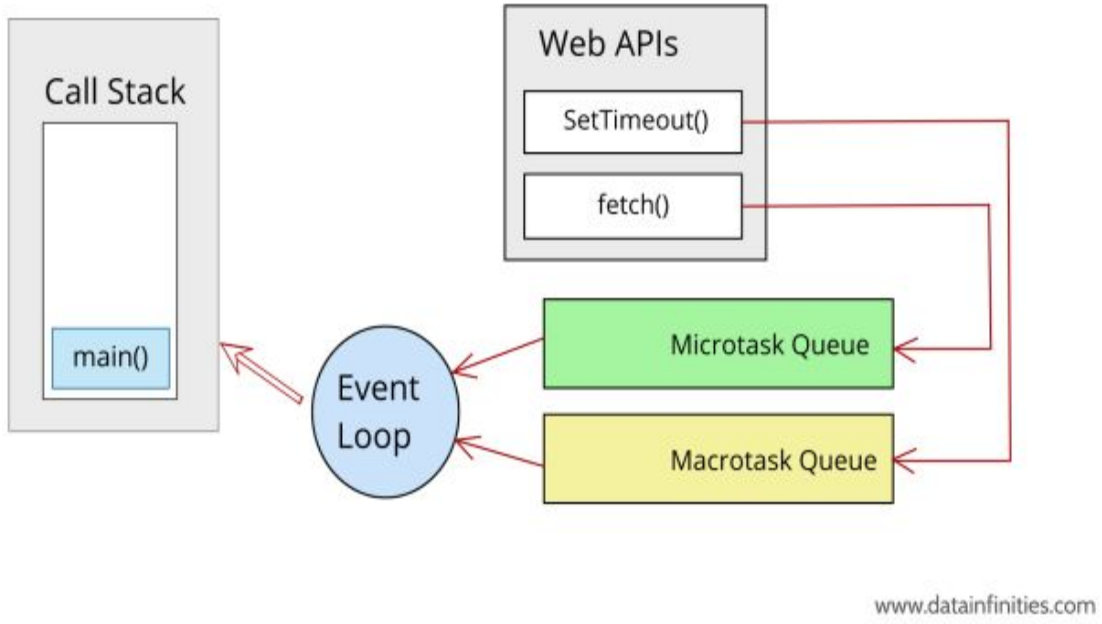
O Fluxo: Lendo vários itens em sequência

Fluxo React (Assíncrono)

A UI permanece responsiva. O operador não "sente" a espera.

1. Lê o código de barra e busca o produto
2. Operador já pode ler o próximo item
3. Mesmo com o anterior sem resolver
4. Tudo está sendo processado em paralelo
5. Conforme são resolvidas, entram na lista
6. O item entra no cache.
7. Uma vez no cache, os itens vão vir do cache.

Visualizando a Fluidez (Javascript)



A Fluidez do Assíncrono

Funcionamento interno do javascript ajuda o algoritmo de promises do projeto.

Já que as promises se resolvem na microtask queue que tem alta prioridade e outros callbacks do webapis ficam pra depois.

O que o faz contribuir seus 2 centavos na otimização.

Otimização de Cache: O Projeto Atual

Cache LRU (Least Recently Used)

Se o item já está no Cache LRU (ou foi pré-carregado), a busca é **instantânea**. Sem espera, sem travamento. Estrutura de dado escolhida foi **doubly linked** list com **hashmap** que permite operações eficientes, O(1) no big O.

Oportunidade 1: Stale Time (SWR)

Usar o dado do cache (instantâneo) e, “ao mesmo tempo”, buscar uma atualização em segundo plano (`stale-while-revalidate`). UX rápida, dados sempre frescos.

Oportunidade 2: Pré-Carga "Hot"

Carregar os 100 produtos mais vendidos no cache LRU assim que o PDV abre. A primeira venda de Coca-Cola do dia será **instantânea**.

Impactos do bloqueio

O Custo do Síncrono

Quando o PDV precisa processar um pagamento ou consultar um item, a tela inteira "congela".

Impacto na UX: O operador para. O cliente na fila fica impaciente. A percepção é de lentidão, mesmo que a operação dure apenas 2 segundos.

Conclusão: A Escolha do Negócio

Com as otimizações o **PDV** ganha em **experiência do usuário (UX), agilidade e integração**, graças ao uso inteligente de concorrência (paralelismo assíncrono) e cache.