



Firebird 3.0

O que podemos esperar

Carlos H. Cantu

Baseado nos slides criados por Dmitry Yemanov



Relembrando



- Idéia original:
FB 3 = Merge do Firebird 2 com o Vulcan
- O que acabou acontecendo:
 - Firebird 2.1, 2.5...
 - Vulcan descartado, usado somente como fonte de “idéias”
 - Não houve merge de código





- **SuperServer** – cache compartilhado, multi-threaded
- **Classic** – múltiplos processos (um por conexão), cache individual
- **SuperClassic** – multi-threaded com cache individual





- **Cache de metadata - Contem informações sobre a estrutura do banco de dados (tabelas, relacionamentos, constraints, etc.) armazenadas na memória.**
- **Cache de páginas – Armazena na memória páginas do BD recentemente acessadas.**



- Objetivos principais
 - Engine multi-threaded 100% compatível com SMP e multi-core, com cache compartilhado.
 - Arquitetura renovada para as futuras versões.
- Objetivos secundários
 - Performance e escalabilidade melhoradas
 - Mais recursos de segurança
 - Melhor otimizador de queries
 - Extensão das habilidades de monitoramento
 - Melhorias na SQL





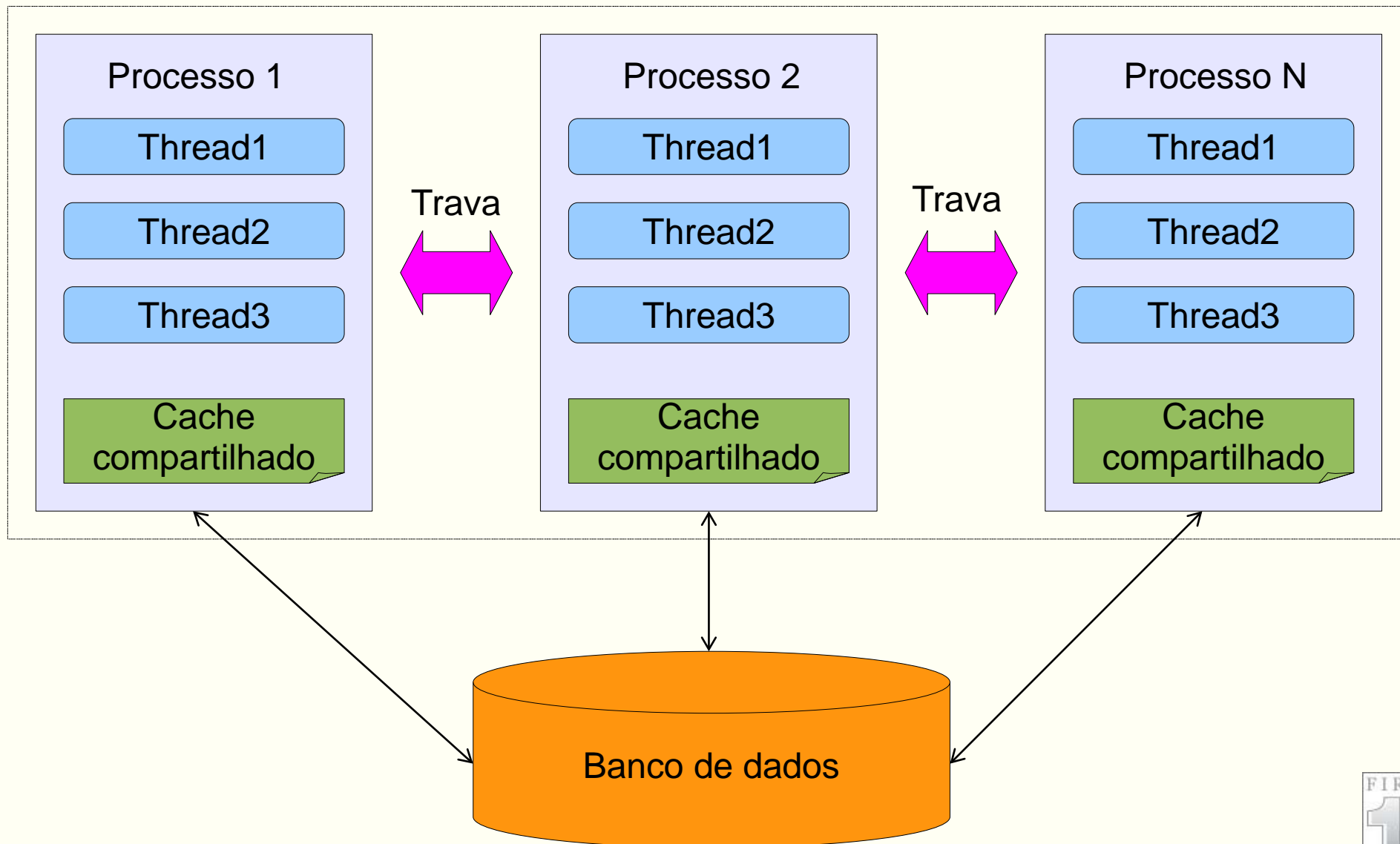
- Com certeza
 - Cache compartilhado com sincronização escalável
 - Suporte para travas entre threads e processos
- Se for possível
 - Plano A: cache de metadados compartilhado, cache de comandos compilados compartilhado
 - Plan B: cache de metadados não compartilhado, mas com redução no uso de memória (liberação das requisições de cache assim que desejável)

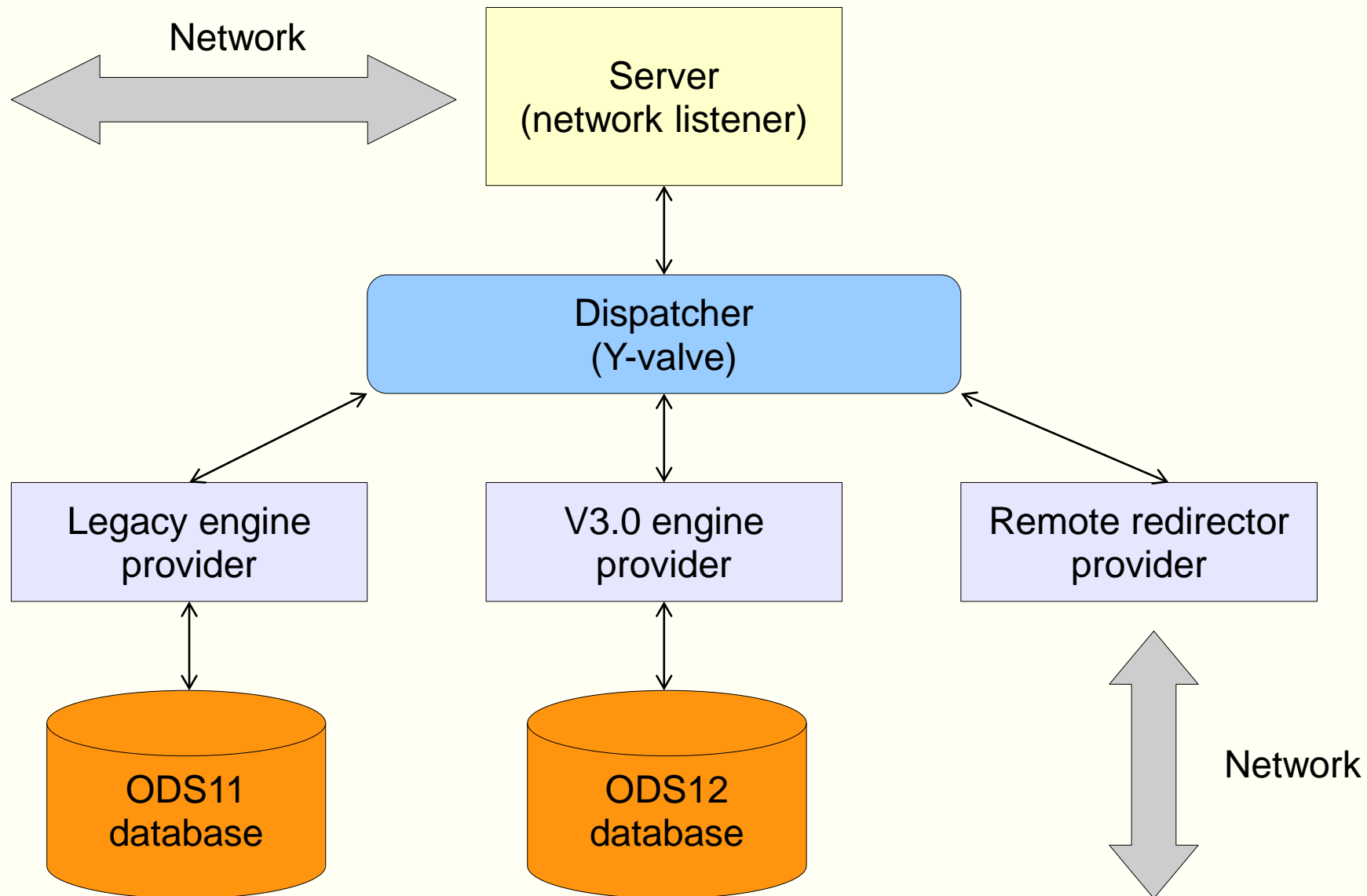




- Unificada, mas flexível
 - Biblioteca da engine suportando uma única ODS + executável do servidor (network listener) lendo a biblioteca da engine
 - Acesso exclusivo ou compartilhado aos bancos de dados, configurado por banco através do firebird.conf
 - Modelos de execução disponíveis para acesso compartilhado e configurado por servidor através de parâmetros de linha de comando:
 - Baseado em processos (Classic)
 - Baseado em threads (SuperClassic)
 - Acesso pelo embedded pode também ser compartilhado ou exclusivo









- Estrutura física
 - Flag para marcar páginas de dados livres de lixo (melhora a performance do sweep)
 - Inventário de páginas ↔ SCN relationships (melhoria na velocidade do backup incremental)
 - Melhoria na compressão RLE, passando de blocos de 128 bytes para 32K (menos espaço ocupado por campos grandes que não estejam totalmente utilizados)
- Estrutura lógica
 - Novas tabelas de sistema e monitoramento
 - Metadata armazenada em UTF8 ao invés de UNICODE_FSS





- No nível de rede
 - Plugins de autenticação (API pública, plugin customizável)
 - Criptografia dos dados trafegados
- No nível do banco de dados
 - Autenticação por BD (usuários armazenados no BD)
 - **Criptografia de páginas**
 - GRANT ROLE TO ROLE
 - Roles implícitos (ex: grupos de usuários)
 - Permissões para operações de DDL





- Geral
 - Baseada em “custo”, sempre que possível
 - “Joins” mais eficazes em streams contendo agregações, unions e procedures
 - Hash joins, outer merge joins, cached invariant subqueries, etc
 - Exibição avançada dos planos (muito mais detalhada)
- Estatísticas
 - Informações a nível de tabelas e índices, histogramas com distribuição dos valores dos campos
 - Refresh total ou parcial, automático



Exibição de PLANos avançada



SELECT statement

- > First [10]

- > Sort [SUM desc, O_ORDERDATE asc]

- > Aggregate

- > Sort [L_ORDERKEY, O_ORDERDATE, O_SHIPPRIORITY]

- > Inner Loop Join

- > Filter

- > Table [ORDERS] Access By ID

- > Bitmap

- > Index [ORDERS_ORDERDATE] Range Scan

- > Filter

- > Table [CUSTOMER] Access By ID

- > Bitmap

- > Index [CUSTOMER_PK] Unique Scan

- > Filter

- > Table [LINEITEM] Access By ID

- > Bitmap

- > Index [LINEITEM_PK] Unique Scan





- Já implementados

- Procedures, funções e triggers externos
- Funções escalares em PSQL
- PSQL packages (similar ao Oracle)
- Triggers DDL
- Identity columns
- Funções de janela (Window functions)
- Cursores bi-direcionais no PSQL
- Exceções parametrizadas definidas pelo usuário





Duvidas?