



Super Server (SS) **x** **Classic Server (CS)** **x** **Super Classic (v2.5) (SC)**

Alexandre Benson Smith

6º Firebird Developers Day





Desenvolvimento de software
ODIN - Sistema de Gestão Empresarial

www.thorsoftware.com.br
alexandre@thorsoftware.com.br

ServerPTBR
Firebird'it with us

Hosting especializado em Firebird

www.serverptbr.com
alexandre@serverptbr.com



Comunidade sem vínculo com qualquer
empresa, voltada ao suporte e divulgação
do Firebird em países da língua portuguesa

www.comunidade-firebird.org
alexandre.smith@comunidade-firebird.org





Se temos um Super, qual a razão de usar um “não super” ???

“Super” <> “Melhor”

“Super” = “Supervisor”

“Super” <> “Superman”





Diferença das arquiteturas:

SS – Um processo Listener, uma thread por conexão, cache compartilhado entre as diversas conexões, sem isolamento entre as threads, mais leve, menos robusto, atualmente não funciona bem em SMP

CS – Um processo Listener (xinetd), um processo por conexão, cache independente entre as diversas conexões, isolamento entre os processos, mais pesado, mais robusto, funciona bem em SMP

SC – Um processo Listener, uma thread por conexão, cache independente entre as diversas conexões, misto entre SS e CS, mais leve que CS, menos robusto que o CS, funciona bem em SMP





SuperServer:

- Bem leve
- Bom para máquinas com poucos recursos
- Bom para máquinas Mono-processadas (Não faz uso adequado de outros processadores)
- Compartilhamento do cache entre as conexões
- Mais suscetível a “queda geral” pois é um único processo
- Queries muito pesadas interferem em todas as conexões
- Requer pouco hardware mesmo que para muitas conexões
- Limite de 1024 conexões em Windows (TCP/IP)





ClassicServer:

- Mais pesado
- Ideal para máquinas Multi-processadas (mas também tem vantagens em mono-processadas)
- Cache individual para cada conexão
- Super robusto, se uma conexão cair, não derruba as demais
- Queries pesadas não interferem de forma significativa nas demais
- Muitas conexões precisam de hardware adequado
- Sincronização entre os processos
- TempCacheLimit (Default 8MB por Conexão !)





SuperClassic:

- Mais leve que o CS
- Ideal para máquinas Multi-processadas (mas também tem vantagens em mono-processadas)
- Cache individual para cada conexão
- Não é tão robusto como o CS, por ser um processo único como o SS
- Queries pesadas não interferem de forma significativa nas demais
- Não requer muito hardware para muitas conexões





SuperServer:

- Tendência a ter um cache grande definido
- Configurar para que use apenas uma CPU





ClassicServer:

- Tendência a ter um cache pequeno (default 75 páginas) o cache efetivo será feito pelo sistema de arquivos do SO
- Problemas com Semaphores
- Problemas com quantidade de conexões (xinetd)
- Bom ter bastante memória RAM
- Monitoramento da Lock Table





SuperClassic:

- Tendência a ter um cache pequeno (default 75 páginas) o cache efetivo será feito pelo sistema de arquivos do SO
- Bom ter bastante Memória RAM





SuperServer (v2.5):

- Balanceamento das conexões a cada banco de dados entre processadores
- Muitas conexões a um único banco de dados continuam fixas a um único processador





Embedded:

- Até a versão 2.1 é baseado na arquitetura SS em windows, para sistemas POSIX é CS
- Na versão 2.5 será baseado na arquitetura SC tanto para Windows quanto para POSIX

Quais as mudanças de SS -> SC ?

- Acesso simultâneo por diversos aplicativos ao mesmo banco
- Unificação de arquitetura nas diferentes plataformas





Garbage Collection:

- SS
 - Background
 - Cooperative
 - Combined (default)
- CS
 - Cooperative





Como serão as arquiteturas no futuro ?

- Realidade é SMP, SS sem suporte a SMP não tem futuro
- SC trará o acesso a SMP necessário com a “leveza” do SS, mas, terá que ter cache compartilhado implementado
- CS continuará a existir, mesmo sendo mais pesado que o SC, a solidez oferecida compensa o custo em termos de recursos utilizados
- Embedded, existirá e será baseado em SC

Provavelmente teremos então:

- ClassicServer
- SuperClassic
- Embedded (que não deixa de ser SC)





Lock Tables, Lock Manager, FB_Lock_Print

- <http://ibdeveloper.com/issues/issue-2-oct-17-2005/locking-firebird-and-the-lock-table> (**Leitura Obrigatória !**)
- Page Locks, são transientes (adquiridos e liberados ao longo da transação)
- Não pode-se pegar um lock em uma nova página até que libere o lock na página anterior (evitar dead-lock)
- Lock-Table usa uma área compartilhada de memória, no CS, cada processo mapeia a Lock-Table
- Todos os bancos no servidor compartilham a mesma Lock-Table
- Lock-Manager é parte integrante do Engine, quando um processo CS adquire ou libera um lock, o gerenciamento de lock cria um mutex na área desejada e muda seu estado





Lock Tables, Lock Manager, FB_Lock_Print

- Quando um lock incompatível com um já existente no objeto é solicitado pode-se retornar um erro imediatamente ou entrar numa fila de espera
- Lock Table pode ser usado para notificações, como por exemplo a criação de um novo índice
- FB_Lock_Print, utilitário que ajuda a identificar pontos a serem ajustados para melhoria de performance
- Lock Table é usado muito mais intensamente em CS do que em SS, então precisa de uma atenção especial





FB_Lock_Print

Mutex wait: 10.3%

Hash slots: 101, Hash lengths (min/avg/max): 3/ 15/ 30

- Números Mágicos

min 5, avg 10, or max 30

Se for maior do que isso, aumentar o tamanho de hash slots (LockHashSlots no Firebird.conf). Usar um número primo menor que 2048.

Se aumentar o LockHashSlots, aumente também o tamanho da tabela de locks (LockMemSize)

Version: 114, Active owner: 0, Length: 262144, Used: 85740

- Se o espaço usado (Used) estiver próximo do tamanho máximo (Length) aumentar o LockMemSize





Obrigado !

