Firebird Scalability

Dmitry Yemanov

The Firebird Project
http://www.firebirdsql.org

Outline

- **Classic Server vs Super Server**
  - comparison, limits, solutions
- **Memory usage**
  - page and sorting buffers
- **CPU and concurrency**
  - multi-CPU and multi-core, TPC-C figures
- **I/O specifics**
  - possible improvements
- **Scaling up vs scaling out**
  - clustering possibilities

Architectural Differences

- **Classic Server**
  - Process per connection, single-threaded
  - Native scheduling
  - Private caches
  - Shared database file usage
  - Shared lock table
- **Super Server**
  - Single process, multi-threaded
  - Cooperative scheduling
  - Shared caches
  - Exclusive database file usage
  - In-process lock table

Classic Server: Limiting Factors

- **Page cache**
  - Balance between available RAM and load
  - Frequent modifications = extra page writes
- **Lock manager bottlenecks**
  - Single mutex, LockAcquireSpins on SMP
  - Lock access time, LockHashSlots
  - LockMemSize: 100 * <connections> * <cache pages>
- **Other limits**
  - Signal delivery (POSIX)
  - Desktop heap size (Windows)

Super Server: Limiting Factors

- **Page cache**
  - Total amount, 32-bit is not enough
  - Many pages (old FB versions)
- **Cooperative thread scheduling**
  - One request active at a time
  - Rescheduling points
- **Other limits**
  - Socket pool limits (1024 on Windows)
  - Stack size (~1000 WNET/XNET connections)

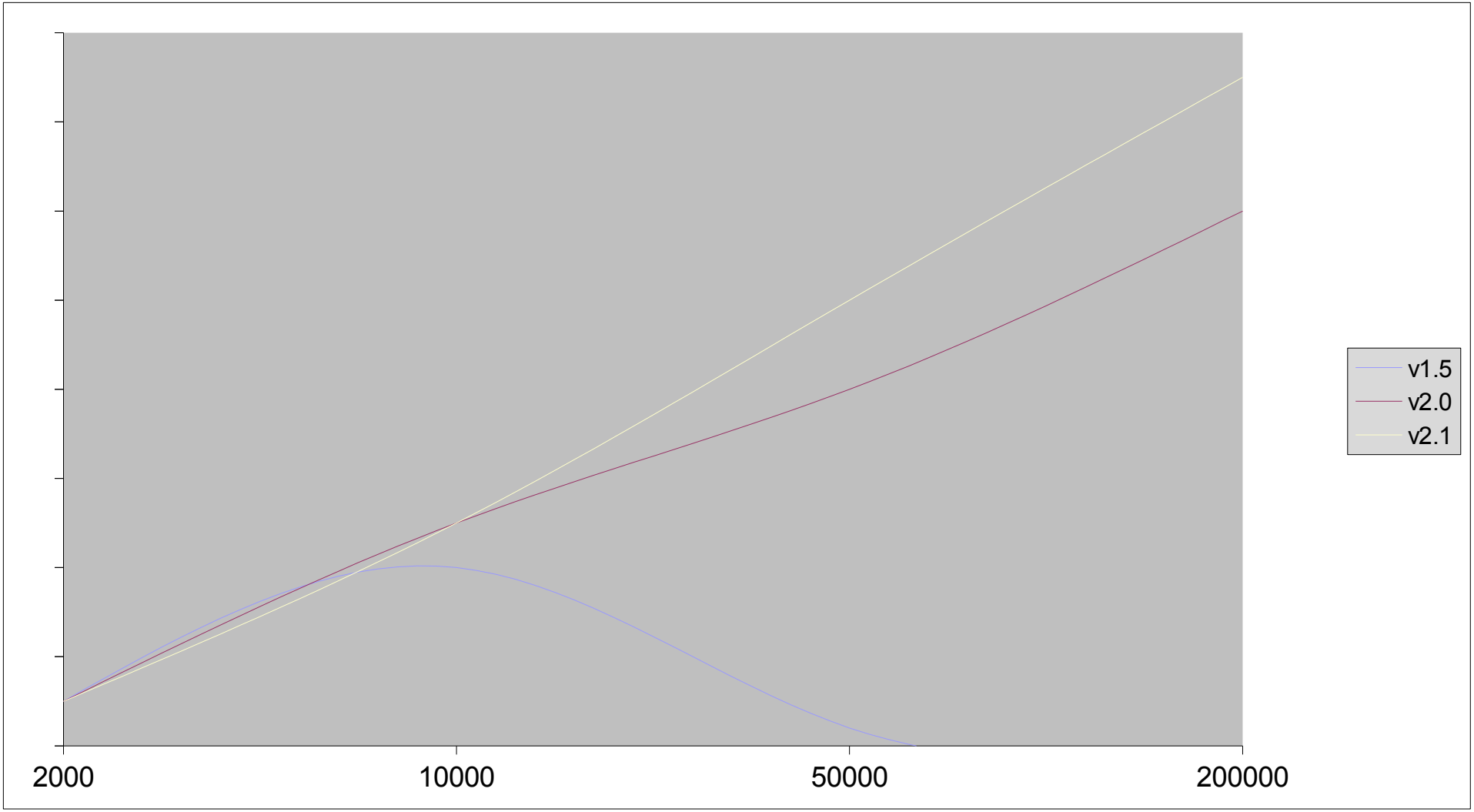
A Mixture of Both

- **Key ideas behind**
  - Single process, multi-threaded
  - Thread pooling
  - Private caches per connection
  - Communication via the lock manager
  - In-process lock table
- **Implementations**
  - Vulcan “No-Shared-Cache” mode
  - RedSoft “SuperClassic” architecture
  - Firebird “Transition” branch

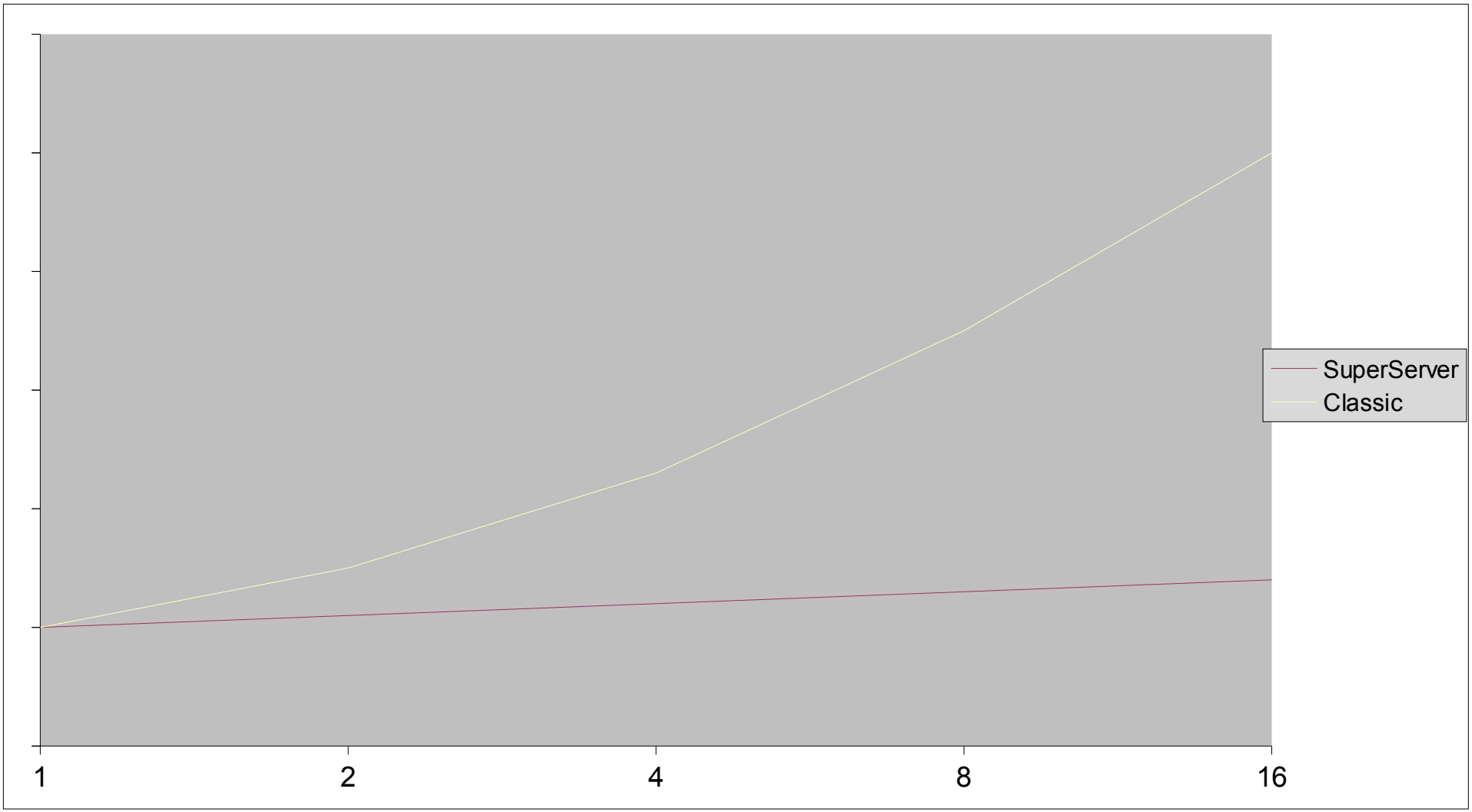
Memory Usage

- **Page cache**
  - Few pages = many I/O operations
  - Many pages = possible problems in CS
  - Remember about the metadata in CS
  - Efficient implementation in SS
- **Temporary space**
  - The bigger the better
  - Should be used with care in CS
- **Other**
  - Lock and event tables (shared memory)

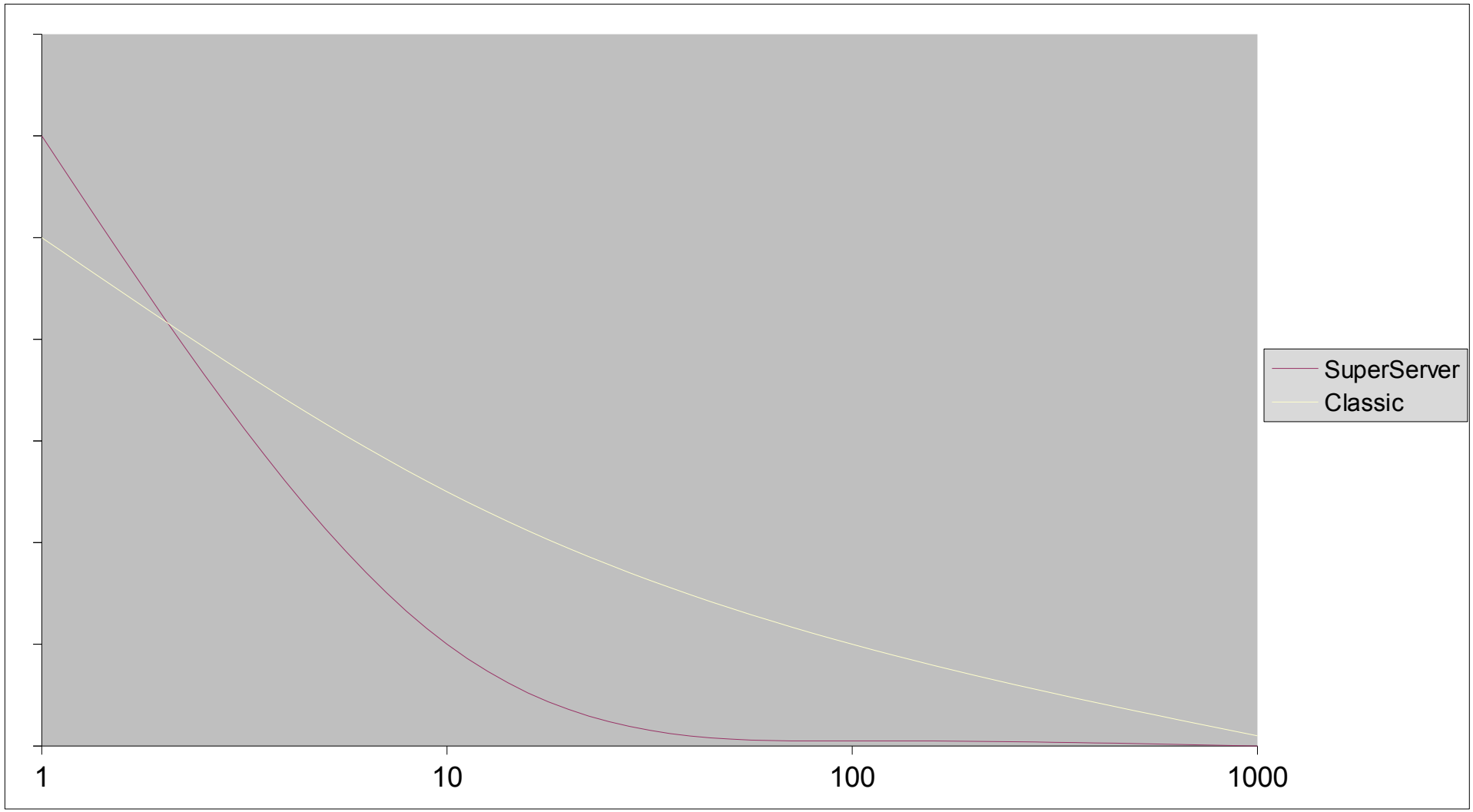
Page Cache Performance for Writers



CPU Scalability



Concurrency: TPC-C benchmark



I/O Specifics

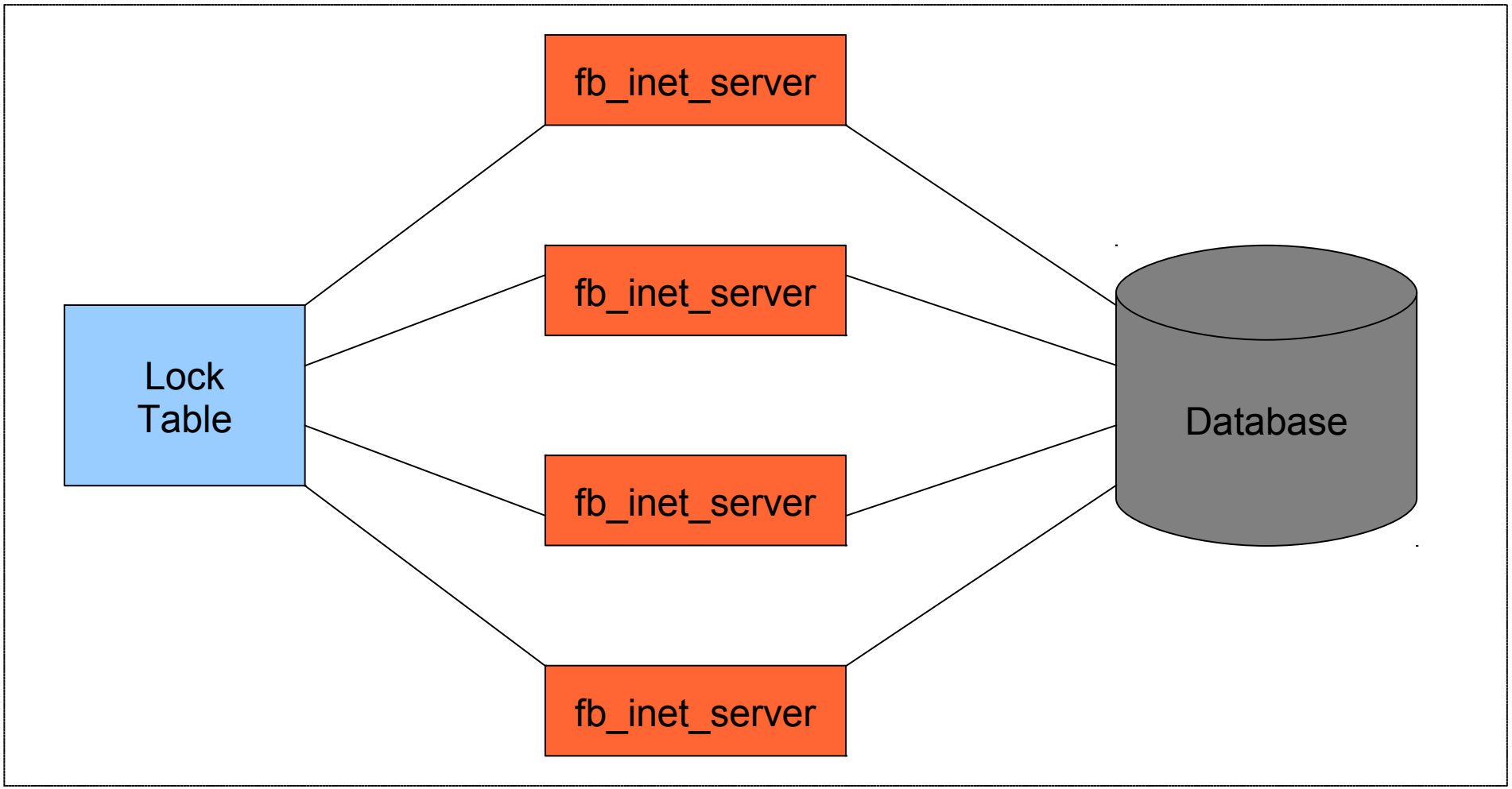
- **Tablespaces**
  - Data separated from indexes and blobs
  - Separated user data (per schema)
- **Partitioning**
  - Active vs backup, by warehouse, etc
- **Underlying improvements**
  - Asynchronous I/O
  - Multi-page reads
  - Cache prefetch

Scaling Up: Searching for the Best Setup

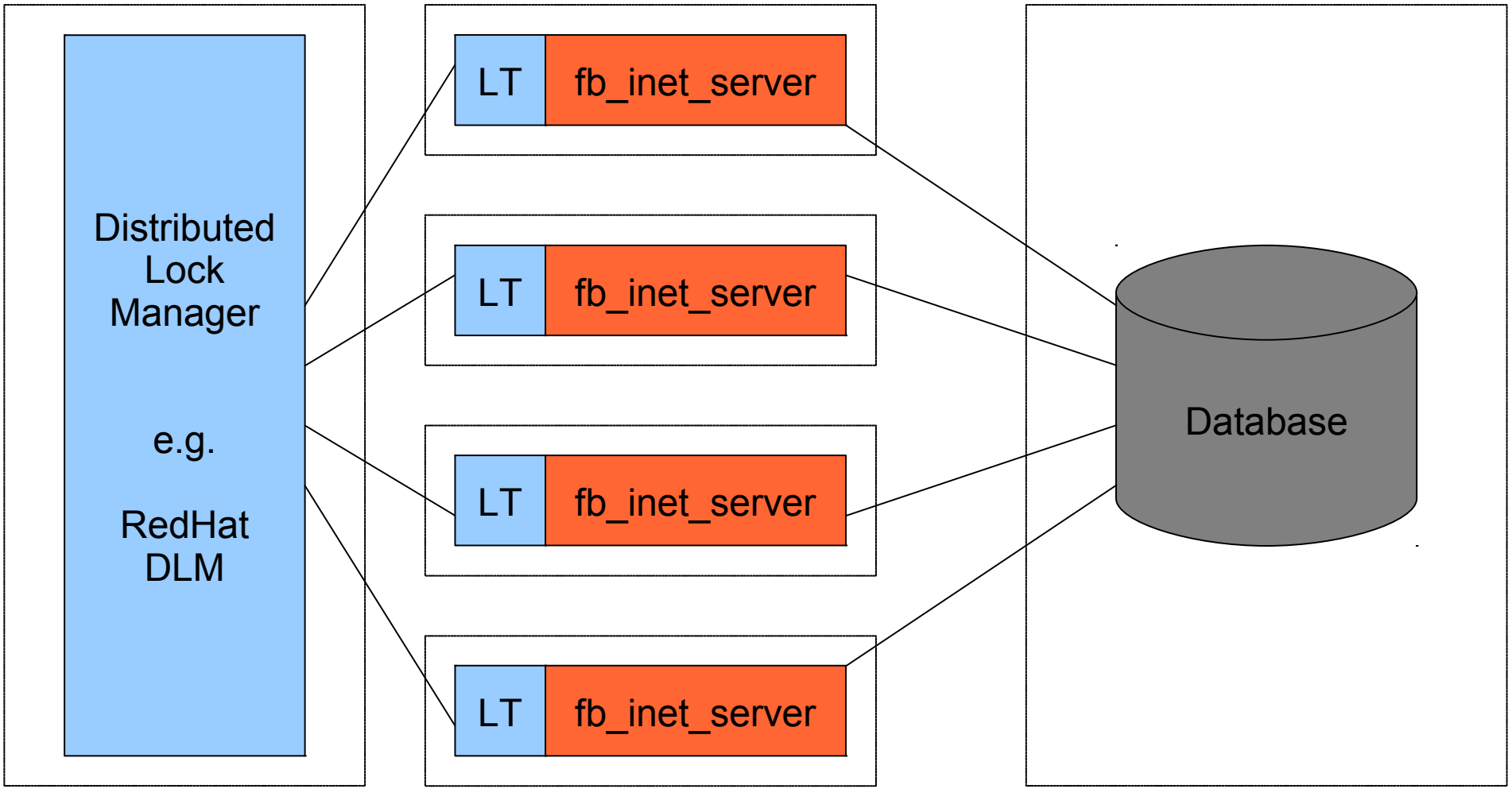
- **Firebird 2.0 Classic on Linux**

- Use a 8K or 16K page size
- Find (experimentally) an optimal page cache size
- Calculate and set up LockMemSize accordingly
- Play with LockAcquireSpins and test the performance
- Consider FW=OFF with MaxUnflushedWrites = 1
- Set up SortMemUpperLimit, if necessary
- Monitor the lock table and increase LockHashSlots

Scaling Out: Firebird Cluster



Scaling Out: Firebird Cluster



Questions?