Super Server in Firebird 3

Sponsors !


24-25 OCTOBER
PRAGUE
Firebird Conference 2014
Firebird Conference 2014

Introduction into Firebird3

- Firebird3 files
  - Root directory
    - network server (firebird.exe)
    - client library (fbclient.dll)
    - utilities (gbak, gfix, isql, etc)
    - configuration files (*.conf)
    - security database (security3.fdb)
    - support libraries (icu*.dll, ib_util.dll, MSVC CRT dll's)
    - support files (firebird.msg, icu*.ddat, license.txt)

Introduction into Firebird3

- Firebird3 files
  - ~~*bin sub-directory*~~
    - yes, no more bin sub-directory on Windows
  - intl sub-directory
    - internalization support library (fbintl.dll, fbintl.conf)
  - plugins sub-directory
    - engine itself (engine12.dll)
    - trace plugin (fbtrace.dll)
    - auth plugins (srp.dll, legacy_auth.dll, legacy_usermanager.dll)
    - UDR engine plugin
  - udf sub-directory
    - standard UDF libraries

Introduction into Firebird3

- Unified network listener
  - single network listener firebird.exe
    - no more fbserver.exe\fb_inet_server.exe
  - launch new process for incoming connection
    - process mode: by default
  - launch new thread for incoming connection
    - threading mode: add switch -m

Introduction into Firebird3

- Unified client library and unified engine library
  - No more fbclient.dll\fbembed.dll
    - single client: fbclient.dll
      - used for remote and embedded access
    - single engine: engine12.dll
      - used as Super Server and Classic Server
  - Super\Classic mode of engine defined by two new firebird.conf settings:
    - SharedDatabase
    - SharedCache

Introduction into Firebird3

- Classic Server
  - network server in process mode (firebird -a)
  - SharedDatabase = true
  - SharedCache = false
- Super Server
  - network server in threading mode (firebird -a -m)
  - SharedDatabase = false
  - SharedCache = true
- Super Classic (obsolete?)
  - network server in threading mode (firebird -a -m)
  - SharedDatabase = true
  - SharedCache = false

Introduction into Firebird3

- Embedded
  - need both fbclient.dll and plugins\engine12.dll
- Embedded in Classic mode
  - SharedDatabase = true
  - SharedCache = false
- Embedded in Super mode
  - SharedDatabase = false
  - SharedCache = true

Super Server

- Main difference from Classic
  - All attachments are handled by the single process
  - Shared common data
    - metadata cache
    - page cache
    - transaction inventory cache
  - Hand-made scheduler: cooperative multitasking
    - single scheduler, single active thread in process
    - threads voluntary switched control to each other
  - Metadata still synchronized by LM
  - Page cache not used LM
    - individual “latches” for page buffers

Super Server before Firebird3

- Pluses
  - shared caches - used much less memory than lot of Classic processes
  - single active thread - almost no need to synchronize concurrent access to in-memory structures
- Minuses
  - single active thread - no way to use more than one CPU\core
    - SMP is not possible
  - cooperative multitasking is very far from fair scheduling
    - “heavy” query in one attachment could get more CPU and slowdown queries in another attachments

Super Server in Firebird3

- On the way to the true Super Server
  - shared page cache
  - shared TIP cache
  - metadata cache is not shared (private per attachment)
  - no more custom scheduler, all threads works in parallel
    - SMP is supported now
  - new synchronization schema

Page Cache

- Page buffers
  - used to cache contents of database file pages
    - descriptor of page buffer, data buffer itself
- Control structures
  - Hash table
    - used to fast search for page buffer by given page number
  - LRU queue
    - defines page buffer eviction policy
  - Precedence graph
    - used to implement “careful write” policy
  - Dirty pages queue
    - what to write on commit

Page Cache

- Synchronization
  - individual latch for every page buffer
    - latch is a lightweight RW-lock
    - used always (despite of SharedCache setting)
  - individual Lock Manager's lock for every page buffer
    - used in SharedCache = false (Classic) mode only
  - separate RW-lock for every control structure
    - hash table
    - LRU queue
    - precedence graph
    - dirty queue

Page Cache

- Fine-grained synchronization is not free !
- Every page access consists of following steps
  - fetch page buffer
    - read contents from disk, if necessary
  - access (use) page buffer
    - read record data, for example
  - release page buffer

Page Cache

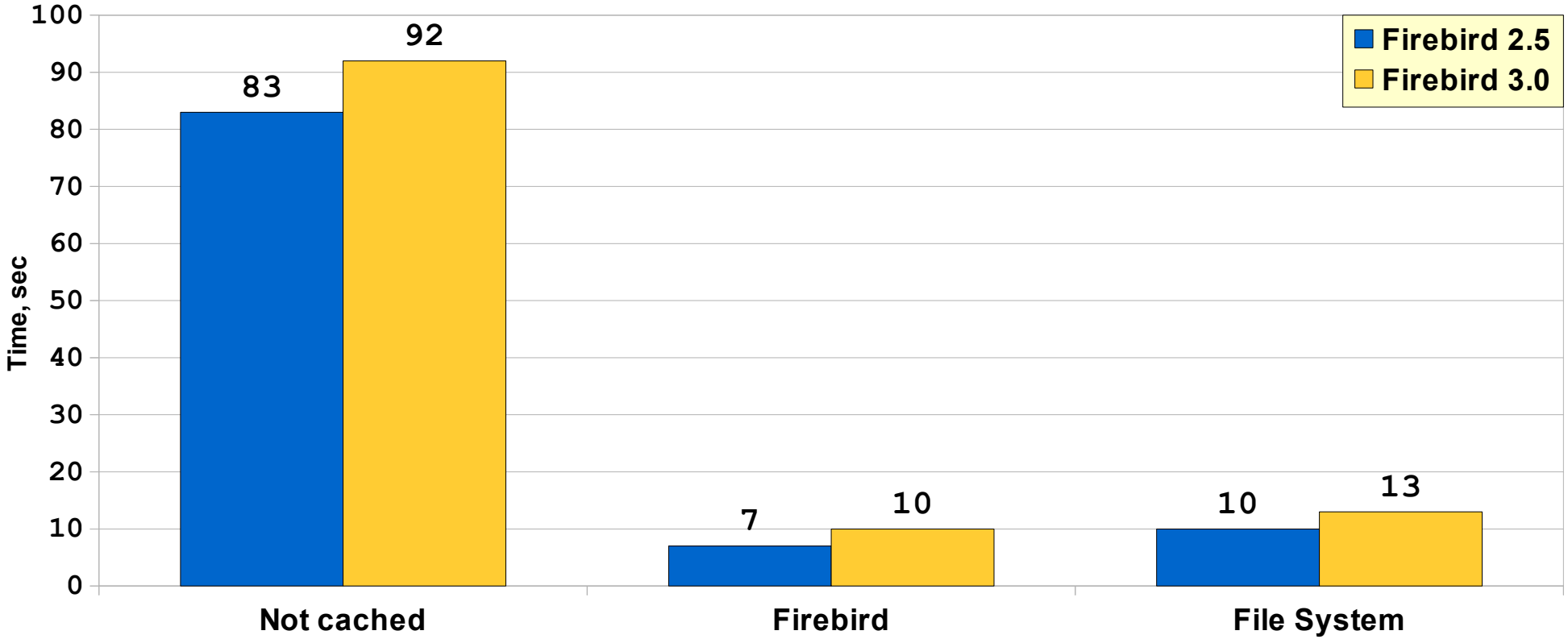
- Cost of page access in Super Server, before v3
  - fetch page
    - find buffer in hash table
    - if buffer is not used - increment use_count
    - else - voluntary switch to another thread and wait while current buffer owner granted us access
  - release page
    - decrement use_count
    - grant access to waiting tread(s)
- Minimal cost is two changes of use_count
    - very, very cheap

Page Cache

- Cost of page access in Super Server, in v3
  - fetch page
    - acquire lock for hash table
    - find buffer in hash table
    - release lock for hash table
    - acquire latch for page buffer
      - interlocked increment of latch state,
      - or wait while current latch owner granted us access
  - release page
    - release latch for page buffer
      - interlocked decrement of latch state
    - grant access to waiting tread(s)
- Minimal cost is four interlocked operations

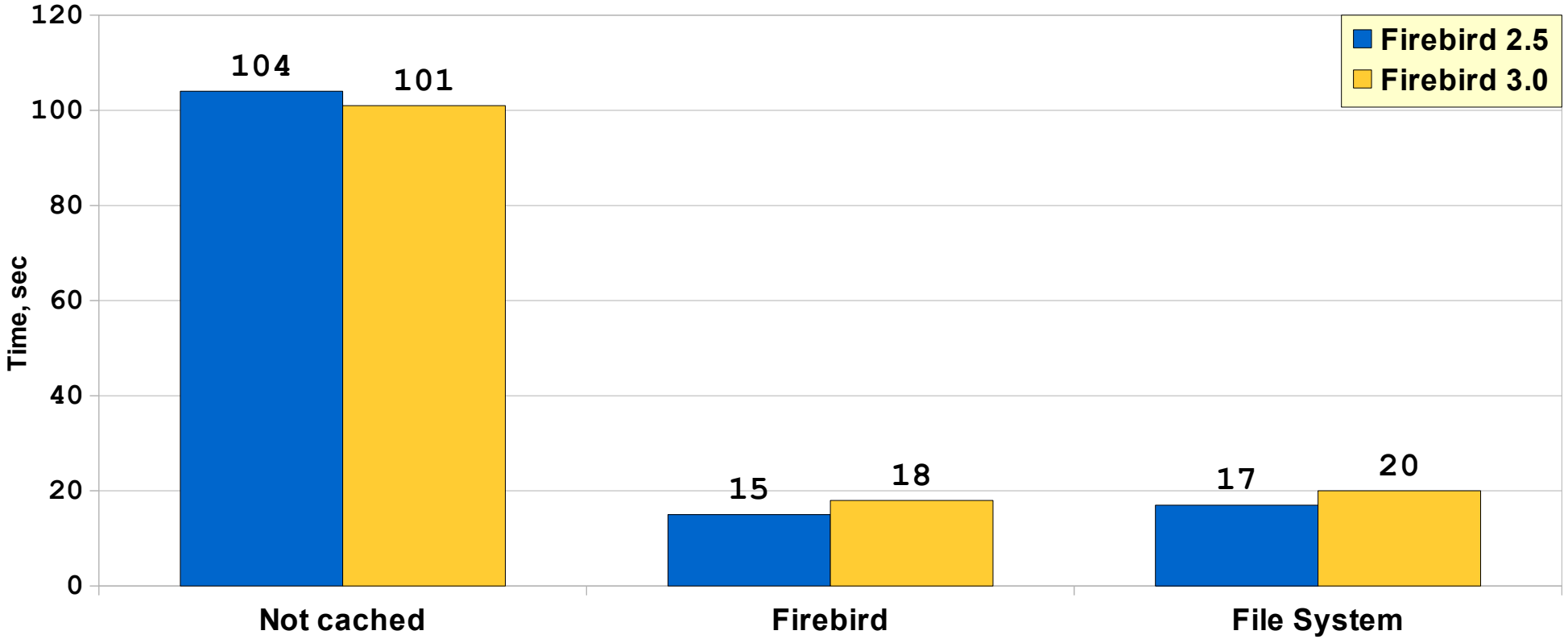
Benchmark, single thread read

```
SELECT COUNT(*) FROM STOCK

10'000'000 rows
Memory buffers = 500'000
Reads from disk to cache = 418'604    ' not cached
Reads from disk to cache = 0          ' cached by Firebird
Reads from disk to cache = 418'822    ' cached by File System
Fetches from cache = 20'837'432
```



Benchmark, single thread read

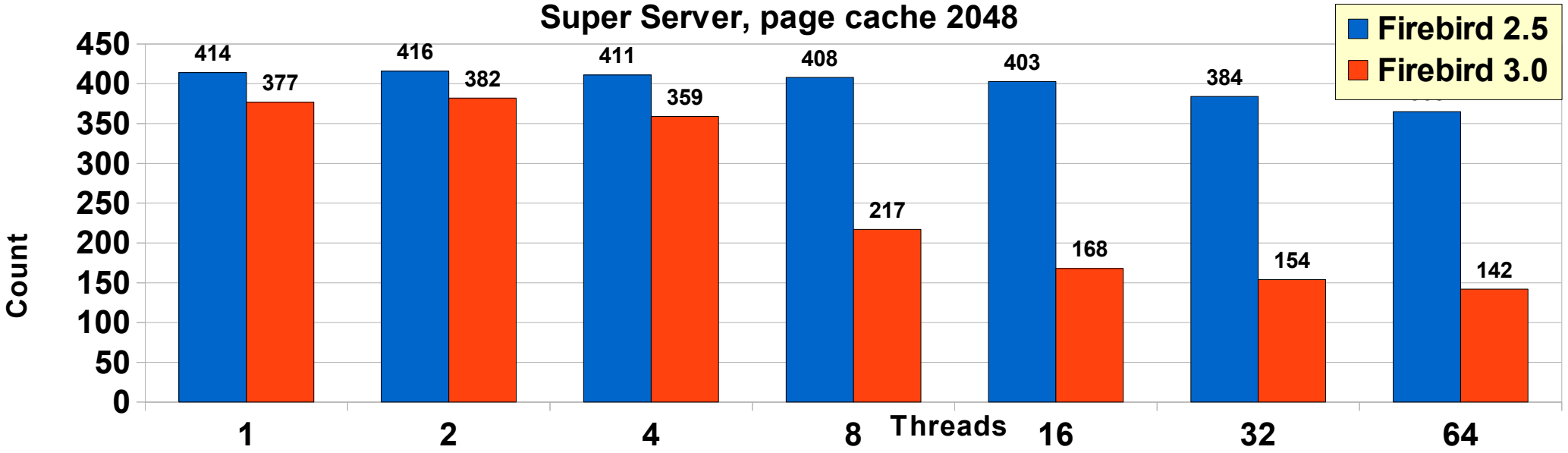
```
SELECT COUNT(DISTINCT S_W_ID), COUNT(DISTINCT S_I_ID) FROM STOCK

10'000'000 rows
Memory buffers = 500'000
Reads from disk to cache = 418'604    ' not cached
Reads from disk to cache = 0          ' cached by Firebird
Reads from disk to cache = 418'822    ' cached by File System
Fetches from cache = 20'837'432
```

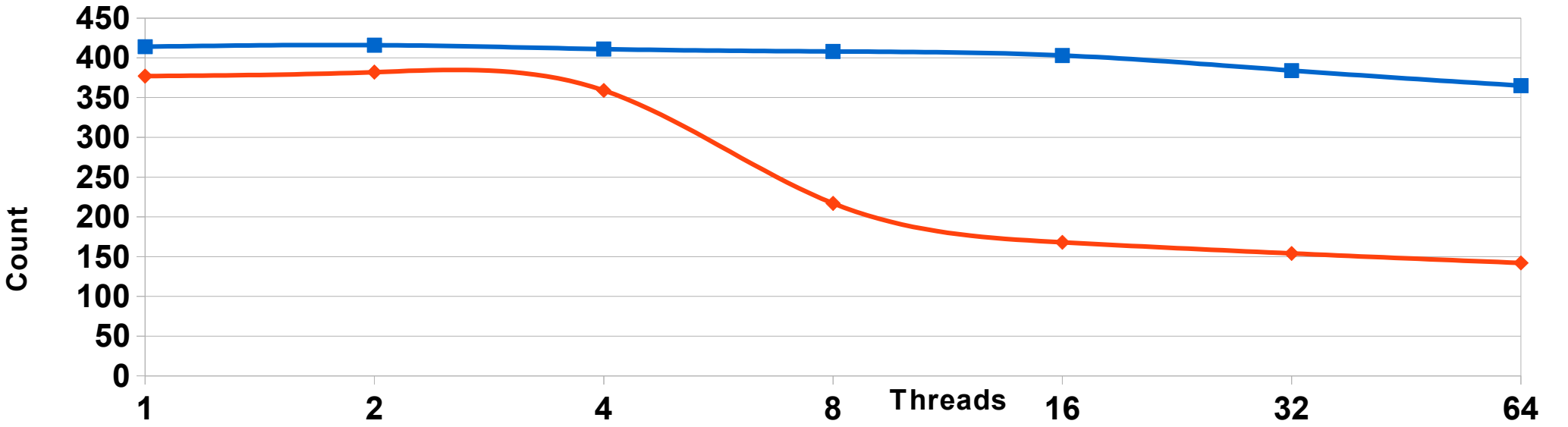


Benchmark, multithreaded read

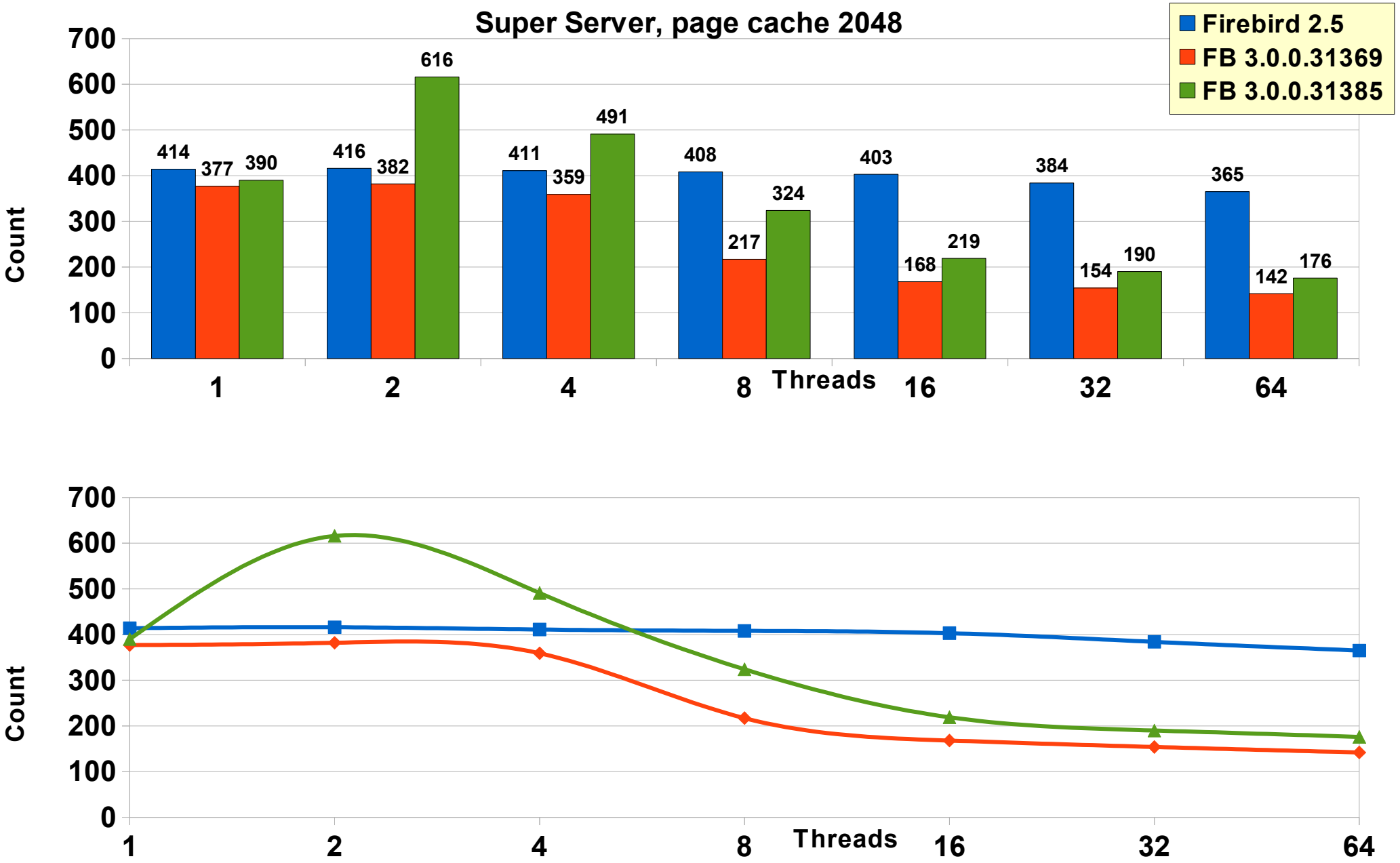
- Multithreaded read only benchmark
  - Table STOCK have 10'000'000 records
    - 100 warehouses, 100'000 items in each
  - Each reader reads random items in own warehouse
  - We don't test network subsystem
    - client application executes procedure, which selects a number of random items (100 by default)
  - We don't test IO subsystem
    - before each test series we ensure whole table STOCK is fully cached by filesysem

Benchmark, multithreaded read

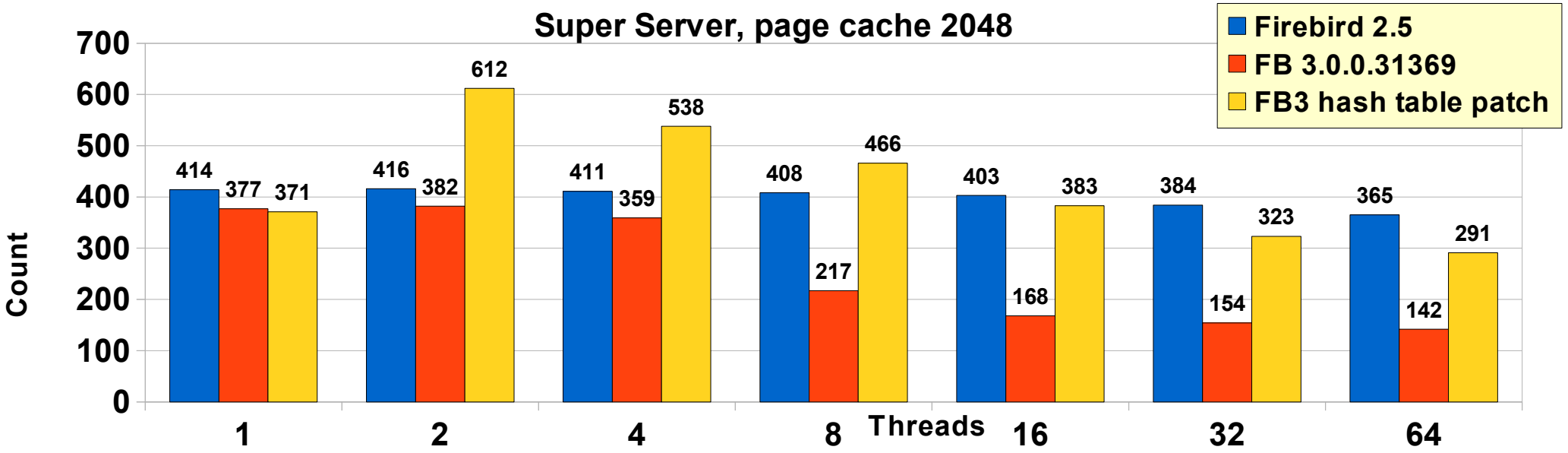


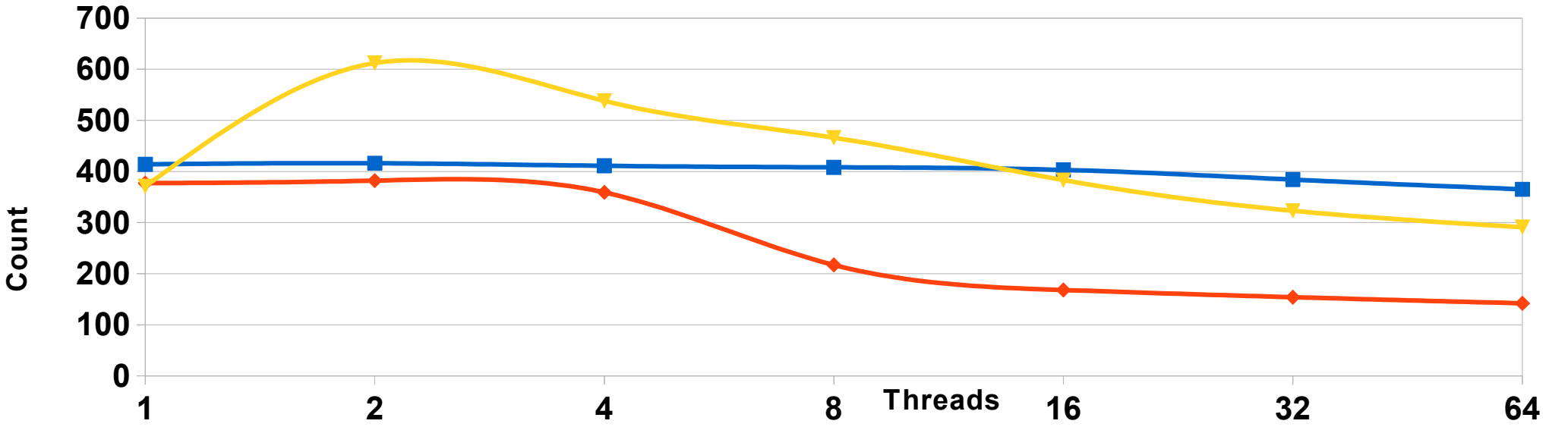


Benchmark, multithreaded read

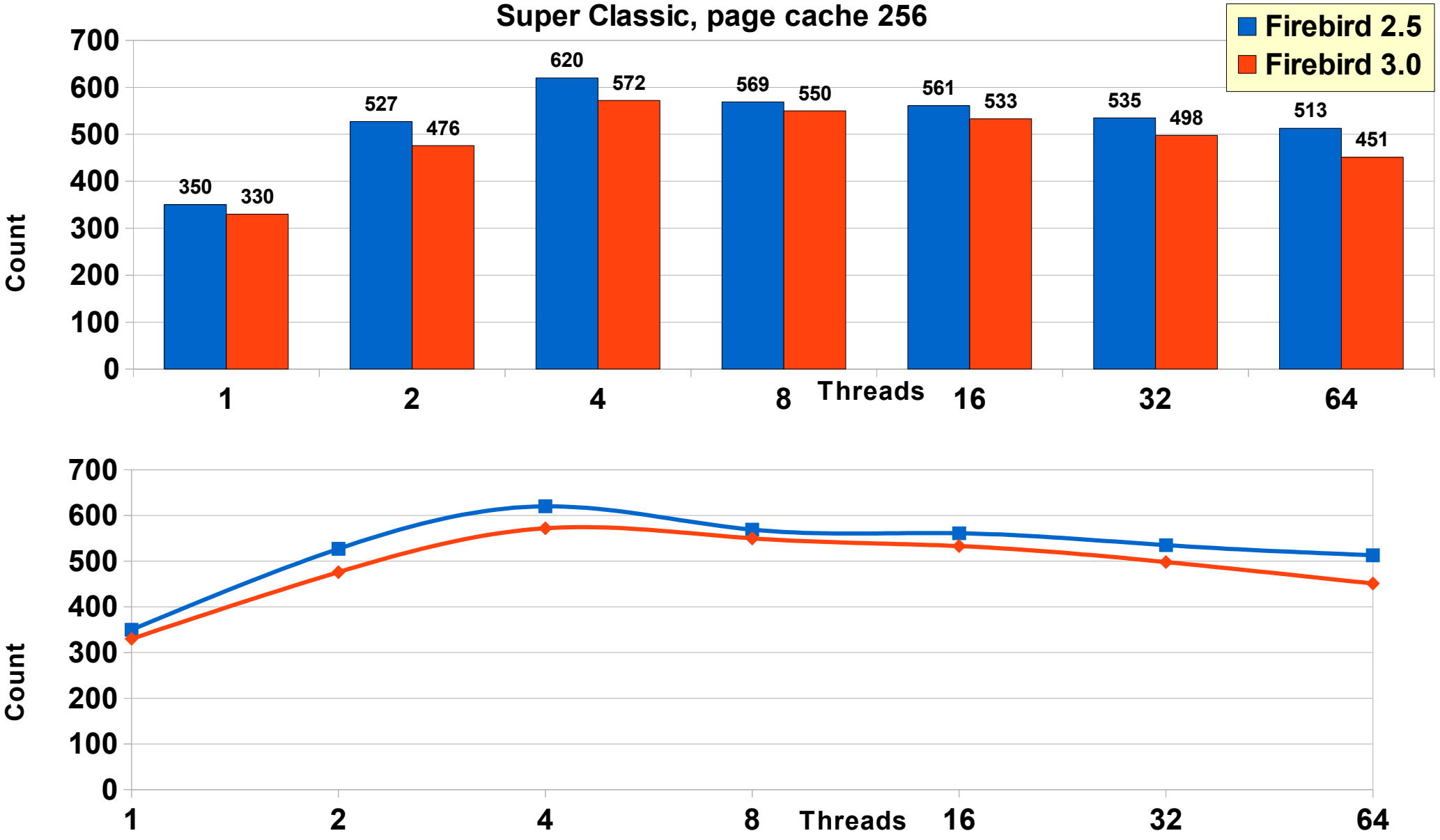


Benchmark, multithreaded read

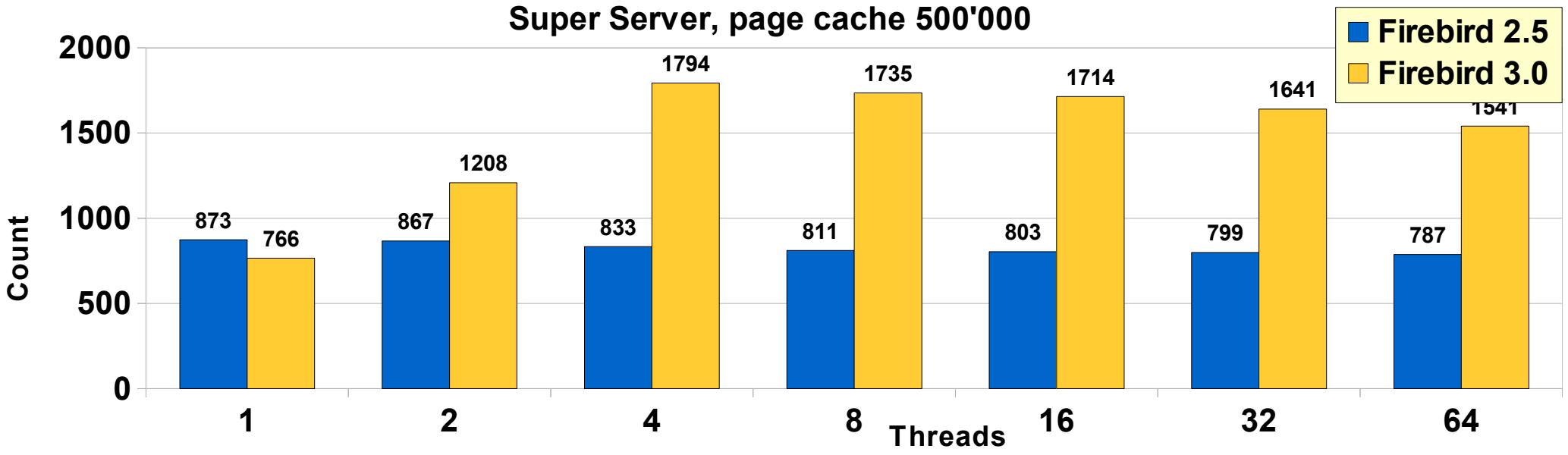


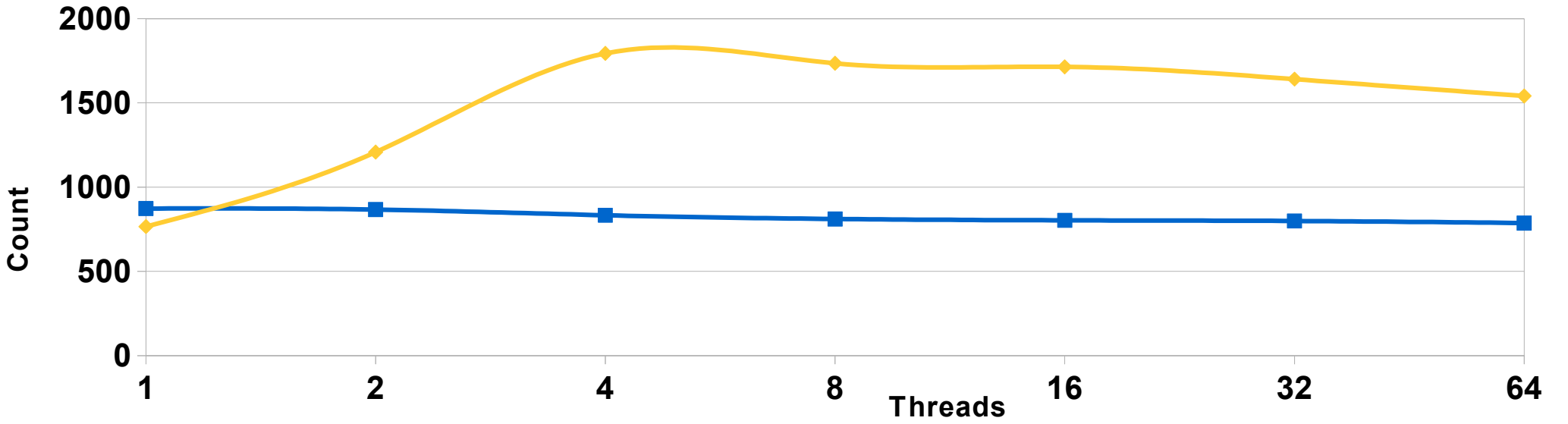


Benchmark, multithreaded read

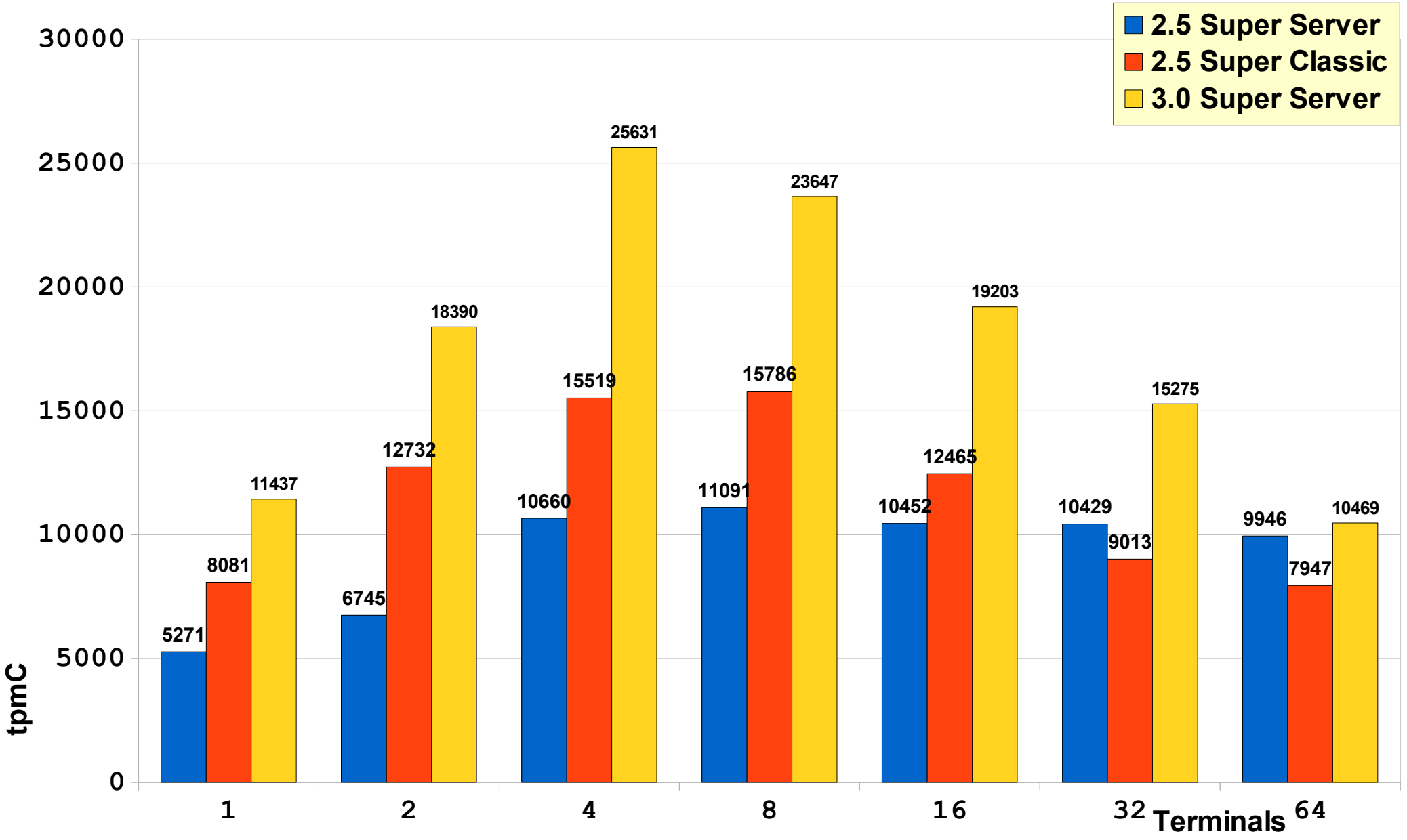


Benchmark, multithreaded read

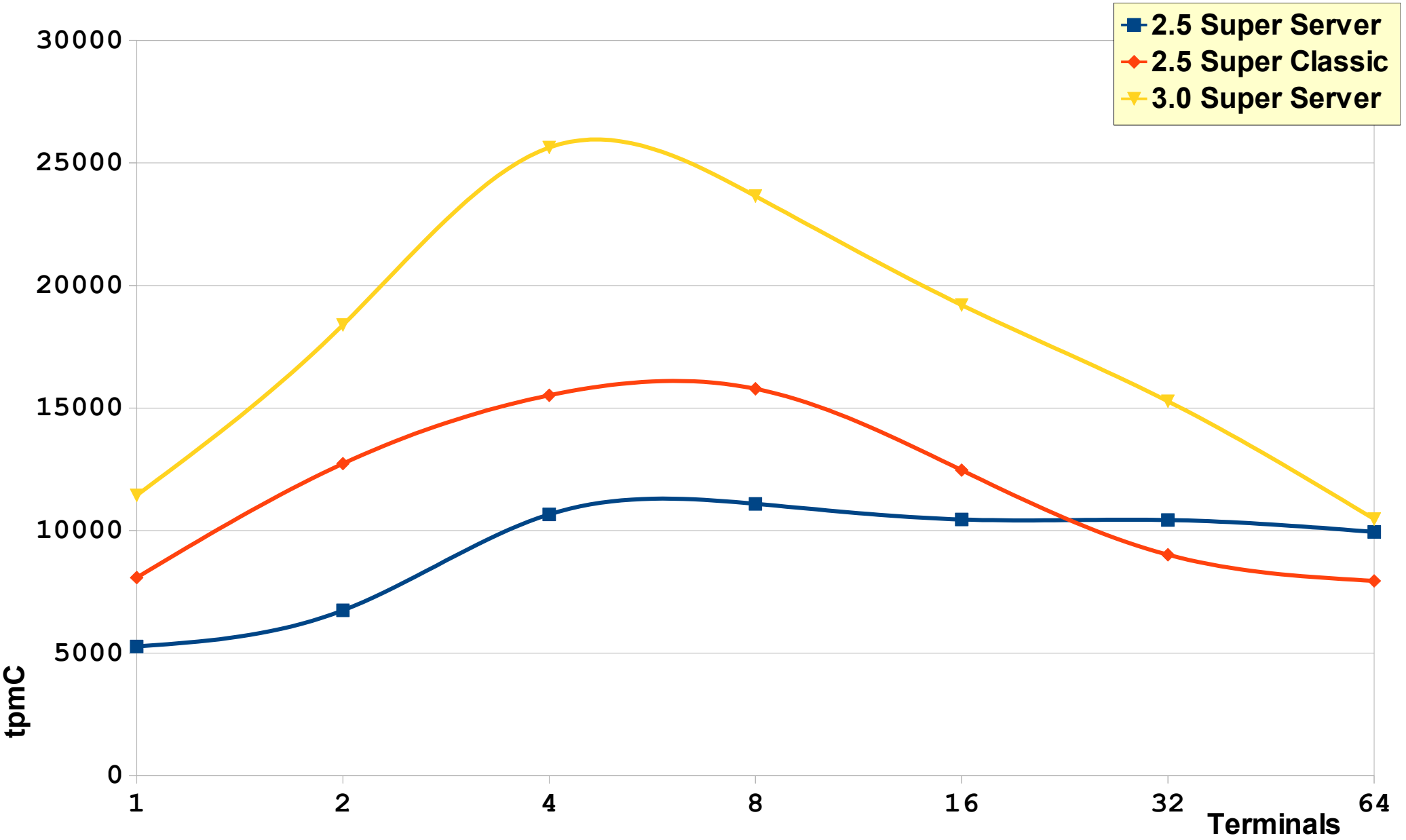




Benchmark, TPCC



Benchmark, TPCC



THANK YOU FOR ATTENTION

Questions ?

Firebird official web site

Firebird tracker

hvlad@users.sf.net