Scaling PostgreSQL on SMP Architectures

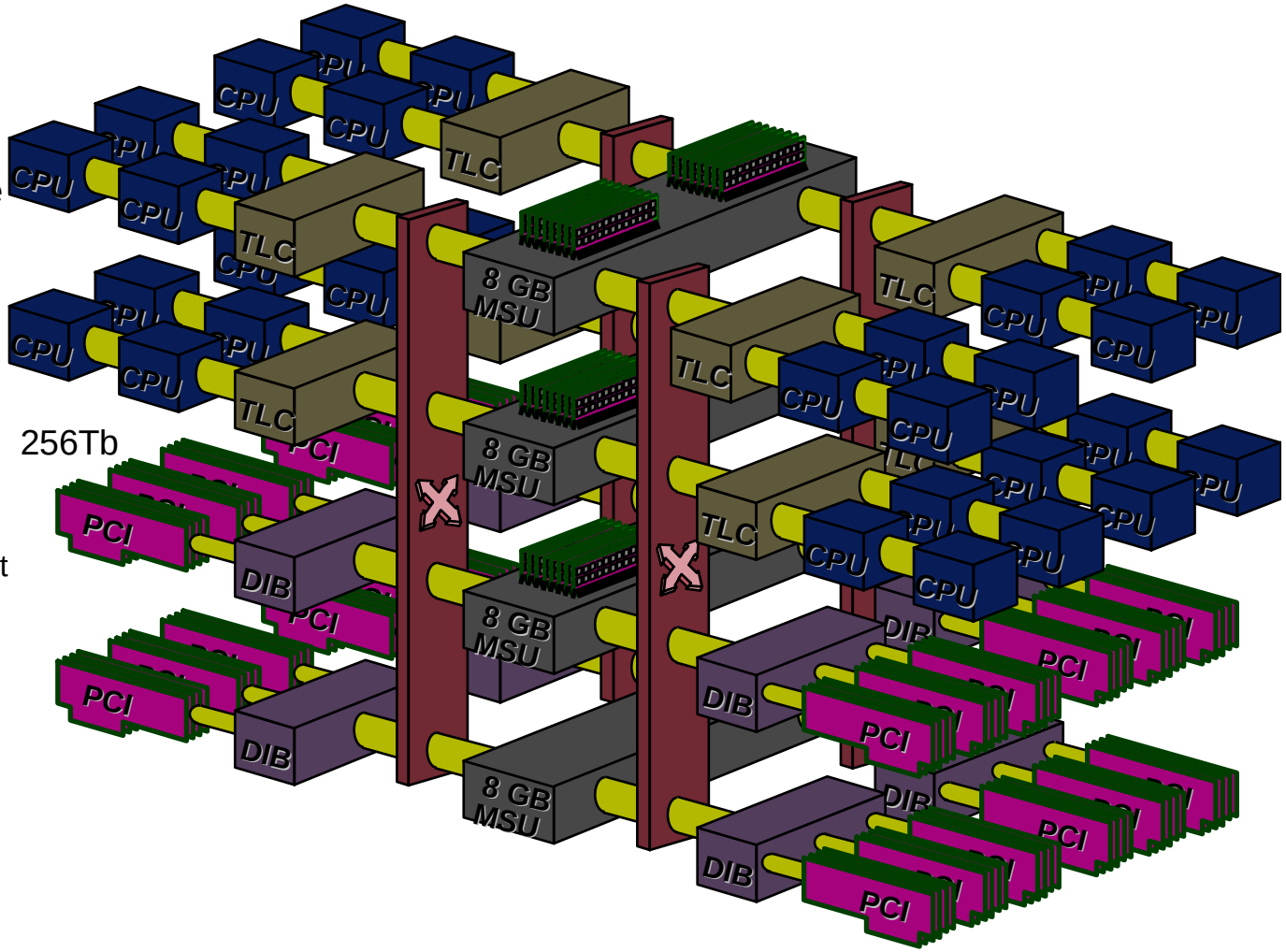
Doug Tolbert, David Strong, Johney Tsai
{doug.tolbert, david.strong, johney.tsai}@unisys.com

Performance vs. Scalability

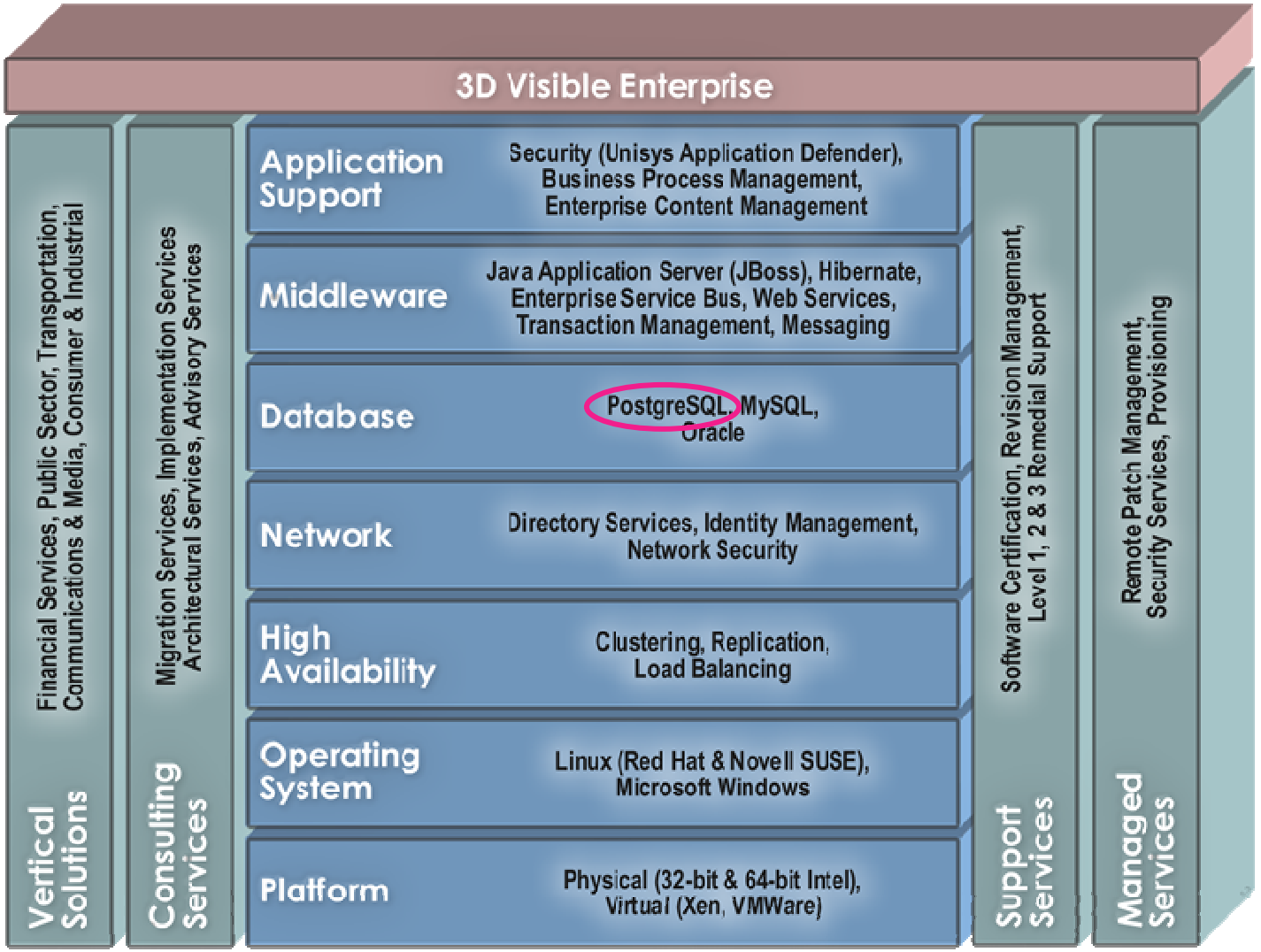
- Performance and Scalability are not the same thing, but are closely related
- Performance
  - Generally based on 1 CPU core, 1 process, 1 thread
- Scalability
  - Generally based on multiple CPU cores, processes and threads
  - Measures throughput as resources are added

Unisys Cellular Multi-Processor Architecture

- Up to 32 CPU sockets
  - In 4- or 8-CPU pods
  - Dual-core supported
    Multi-core eventually
  - Dynamically partitionable
- 32-bit & 64-bit technologies
- Multilayered caches
- Large cross-bar memories
  - IA32: 64Gb (with PAE)
  - EM64T, Opteron, Athlon: 256Tb
  - Itanium: 1Pb
  - Theoretical: Multi-Pb
    Processor family dependent
- Up to 96 PCI Slots
  - Lower on newer models
- Linux: SLES, RedHat
- Windows: Data Center



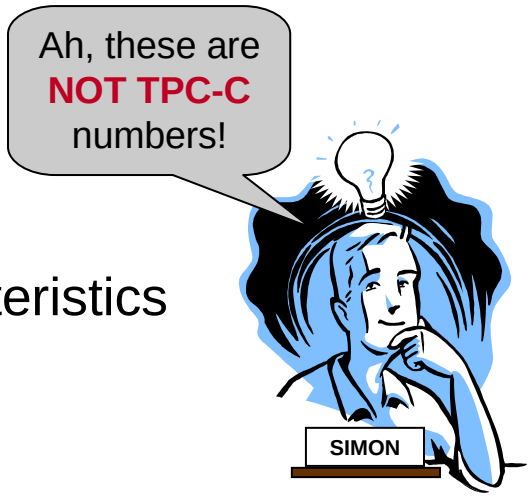
Unisys Open And Secure Integrated Stack



TPC-C Benchmark

- Transaction Processing Performance Council (TPC)
  - Defines suite of benchmarks
  - Sole certifier of vendor-submitted benchmark results
  - Specs, results, more info: http://www.tpc.org
- TPC-C benchmark is a complete order-entry environment
  - Industry standard measure of DBMS OLTP performance across vendors
  - Reports transactions per minute (tpmC) and cost metrics
  - Mix of 5 transactions
    - New Order (selects, updates, inserts)
    - Payment (selects, updates, inserts)
    - Order Status (selects)
    - Stock Level (selects)
    - Delivery (selects, updates, deletes), interactive & deferred variants
- Data volumes, number of clients scaled by vendor's tpmC goal
  - See spec for details
- Current (April 2007) result ranges
  - Best performance: 4,092,799 tpmC @ $2.93/tpmC
  - Lowest performance: 9,347 tpmC
  - Best price/performance: 82,774 tpmC @ $0.94/tpmC

Test Application



- DBMS stressor application with well-known characteristics
- TPCC4J = “TPC-C For Java”
  - Unisys-developed from TPC-C version 5.3 (April 2004)
  - Runs on 1.4.2 JVM or higher
  - Small application footprint (~72K)
  - Very few garbage collection cycles
  - Client overhead low (7-9% for ~1000 users)
- Deviates from TPC-C in important ways
  - Non-audited components and results
  - Data entry screens not populated, only SQL statements run
  - User think time eliminated
  - Reports TPS, not tpmC
    - No, you can’t multiply by 60 to get tpmC!
  - No checkpoints during benchmark runs
  - Not run for a minimum of 8 hours
  - Data volumes not scaled

Test Configuration

Client Driver: Unisys ES3040

4 x 3.0Ghz Intel Xeon MP CPUs (IA32)
Hyper threading enabled
4Gb RAM
1 x 36Gb boot driver (10K rpm)
1 Gbit NIC (copper)

OS: SLES 9 SP3

PostgreSQL: Unisys ES7000/540

32 x 3.0Ghz Intel Xeon MP CPUs (IA32)
Hyper threading disabled
16Gb RAM
1 x 36Gb boot driver (10K rpm)
1 Gbit NIC (copper)

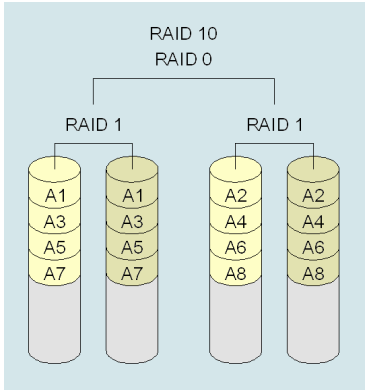
OS: SLES 9 SP3

Storage: EMC CX-200 External Disk

8 x 36Gb drives (15K rpm)
RAID 10 configured (striped mirrors)

Small data volume: 50 TPC-C warehouses, ~5.4Gb (4.7Gb data + 700Mb indexes)

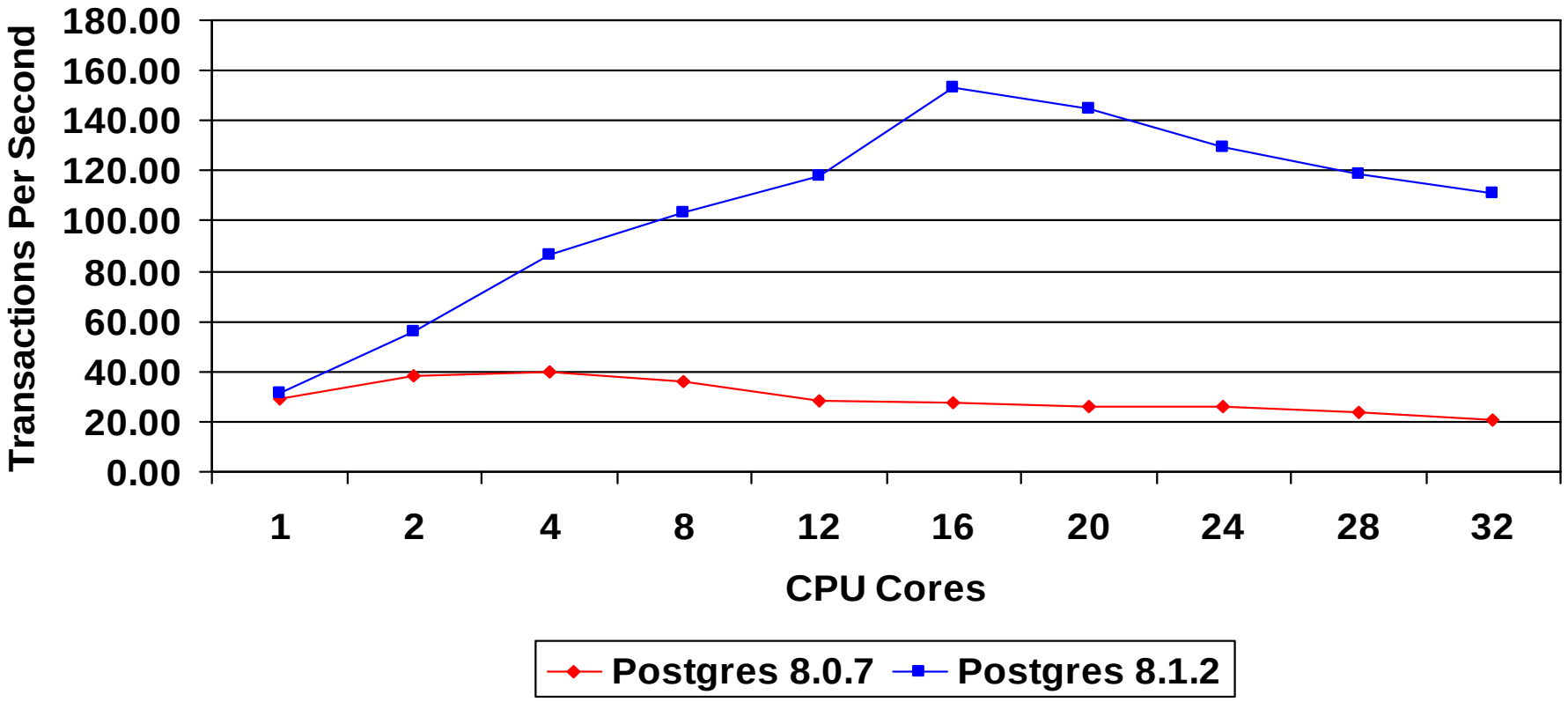
No I/O issues in benchmarking



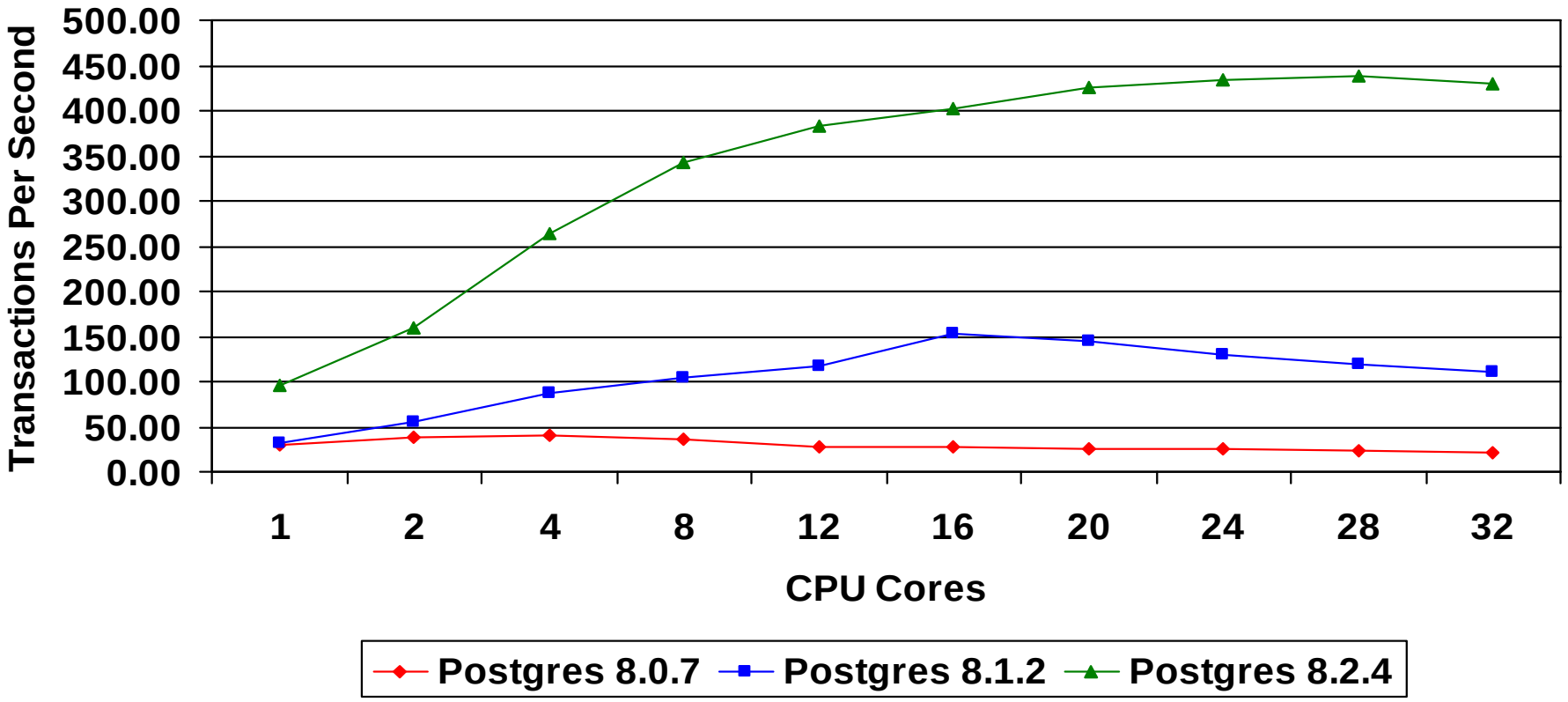
Scalability Considerations

- All components must scale
  - Hardware
  - OS
  - System Applications (database etc.)
  - User Applications
- Every workload will be different

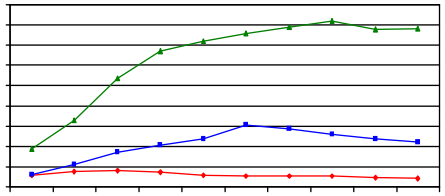
Scaling – Results (2006)



Scaling – Results (2007)

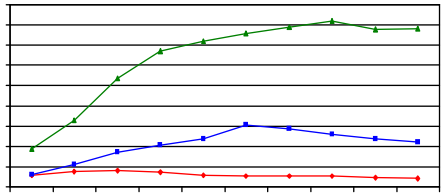


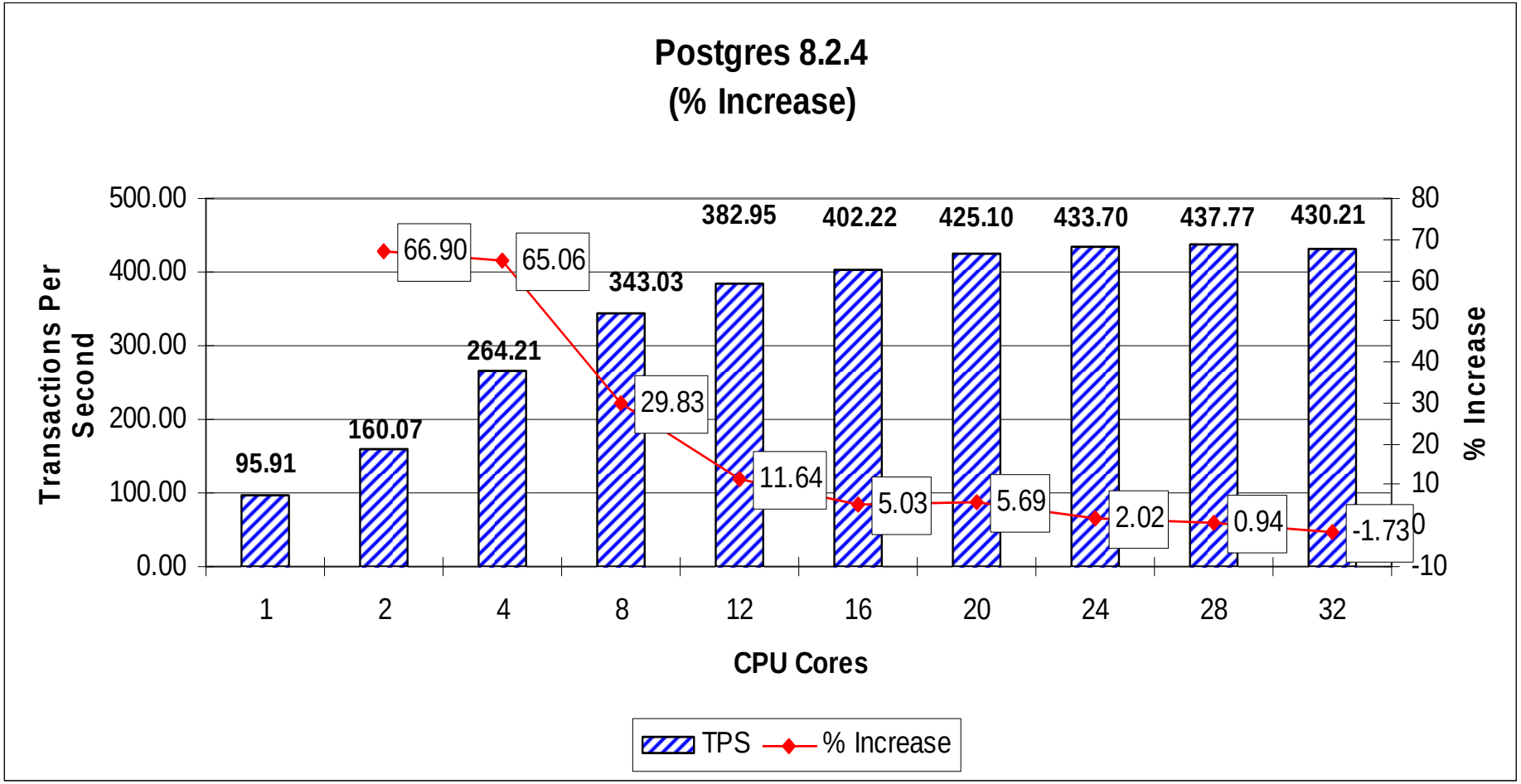
Postgres 8.2.4 Results



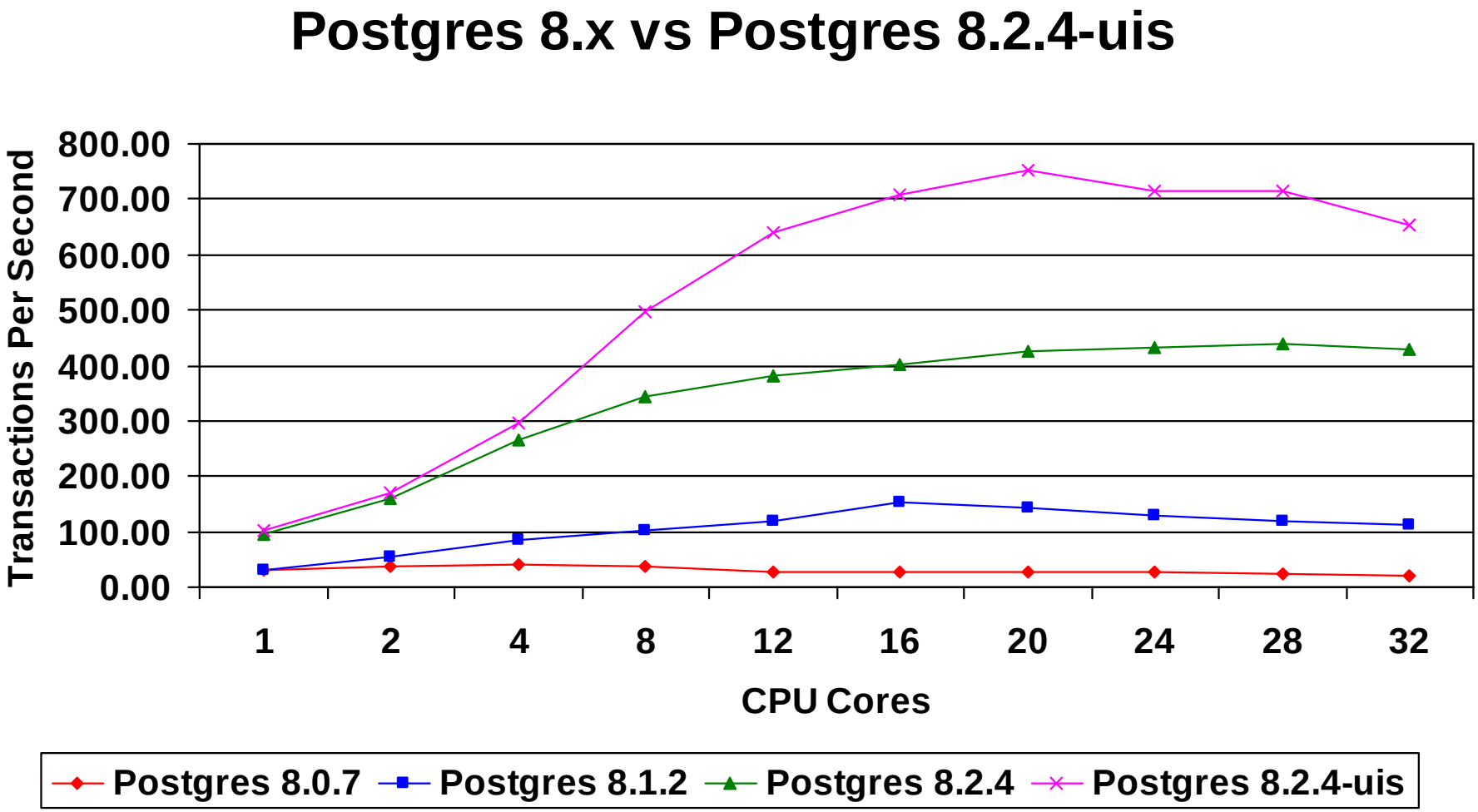
- Performance
  - Based on 1 CPU core
  - 8.2.4 exhibits a significant improvement of ~300% over 8.1.2
- Scalability
  - 8.2.4 uses incremental resources more effectively than 8.1.2
  - At 16 CPU cores, 8.2.4 is ~2.6 times better than 8.1.2
  - 8.2.4 throughput still increasing at 28 CPU cores
  - Small margin of increase after 8 CPU cores
- Fair to conclude that 8.2.4 scales to 28 CPU cores
  - When compared to 8.1.2 (more on this later)
  - At least on this application and platform combination
  - YMMV

Postgres 8.2.4 Results

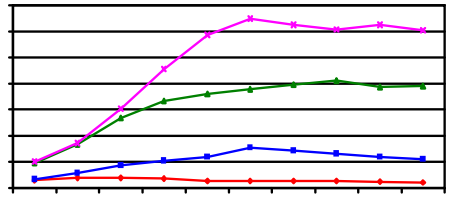




Scaling – Results (2007)

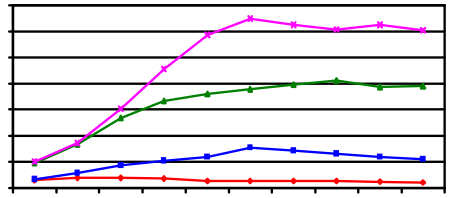


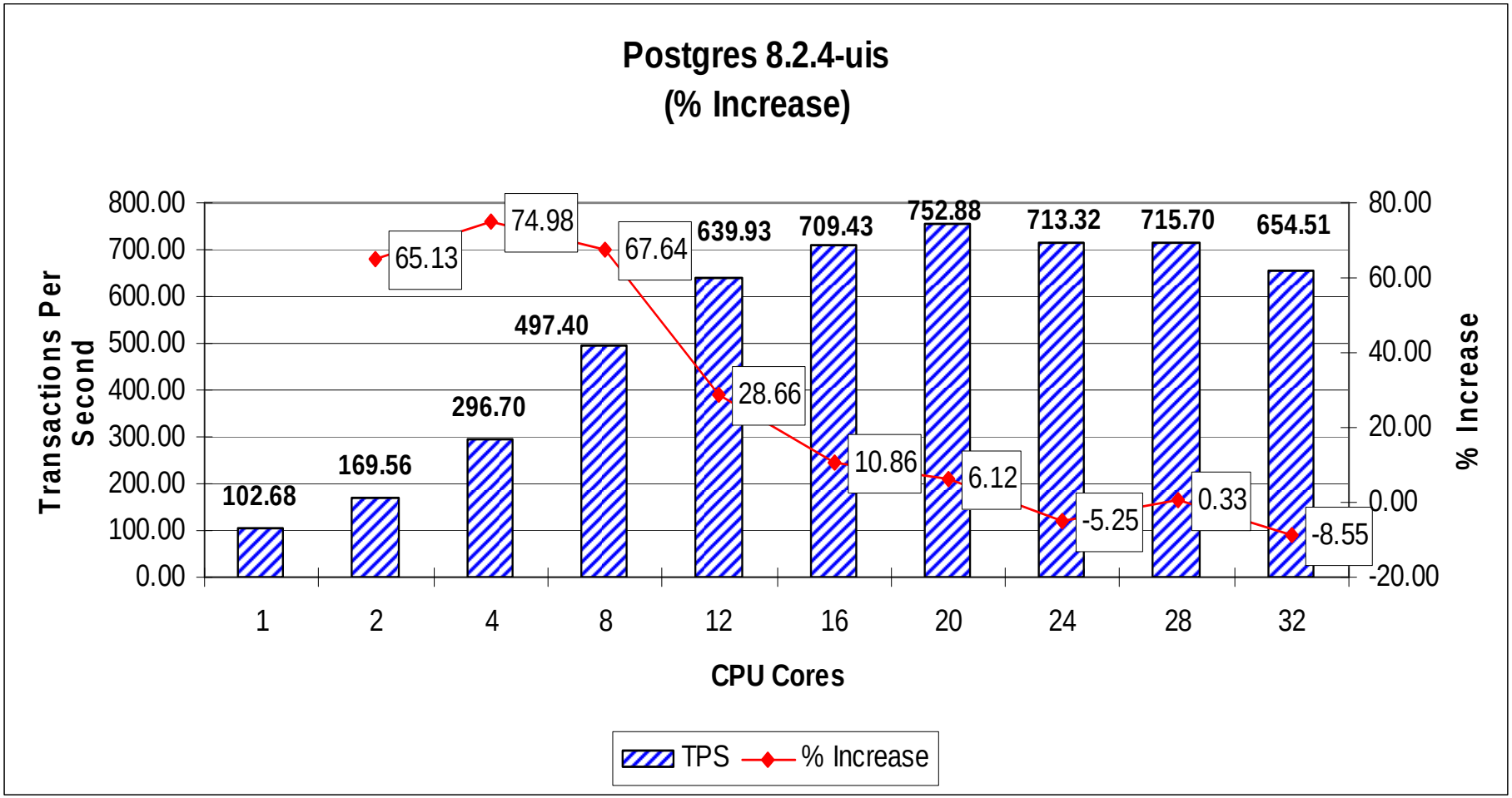
Postgres 8.2.4-uis Results



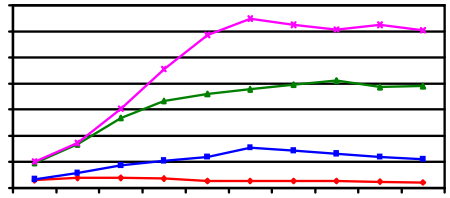
- Base Postgres 8.2.4 with UIS modifications
- Performance
  - Based on 1 CPU core
  - Postgres 8.2.4-uis shows improvement of ~7% over 8.2.4-base
- Scalability
  - Fair to conclude UIS results scale to 20 CPU cores (more on this later)
  - 8.2.4-uis uses incremental resources more effectively than 8.2.4-base
  - Improvement of ~45% at 8 CPU cores (1 cell)
  - Improvement of ~77% at 20 CPU cores (8.2.4-uis peak)
  - Improvement of ~63% at 28 CPU cores (8.2.4-base peak)

Postgres 8.2.4-uis Results



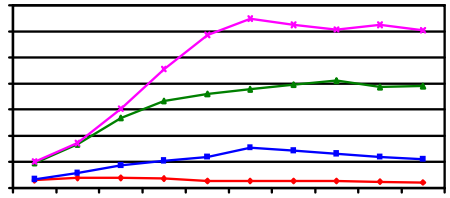


Postgres 8.2.4-uis Modifications



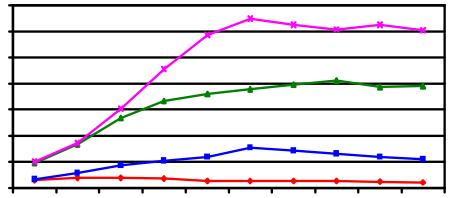
- Additional cache modifications
- Locking modifications
- Affinity management

Postgres 8.2.4-uis Cache



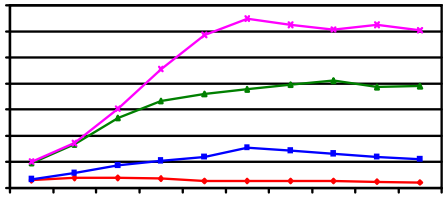
- Alignment to 64 bytes
  - Prevents false sharing
  - Architecture dependant
- Cache prefetch
  - __builtin_prefetch used for LWLockAcquire () as lock not expected to be in cache
  - Need to investigate use of prefetch in hash_search_with_hash_value ()
  - Careful usage and placement required

Postgres 8.2.4-uis Locking



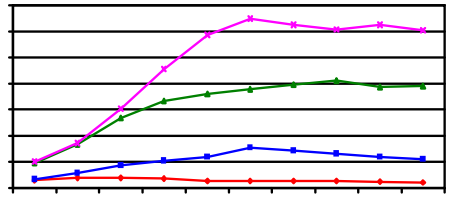
- Expand BufMapping Partition
  - Using 2048 partitions to cut down contention
- Add loop to tas ()
  - Loop added to tas () to prevent early entry to s_lock ()
- Change WALInsertLock access
  - Using SpinLockAcquire () as WALInsertLock locked most of time
  - Considering a queue mechanism for WALInsertLock

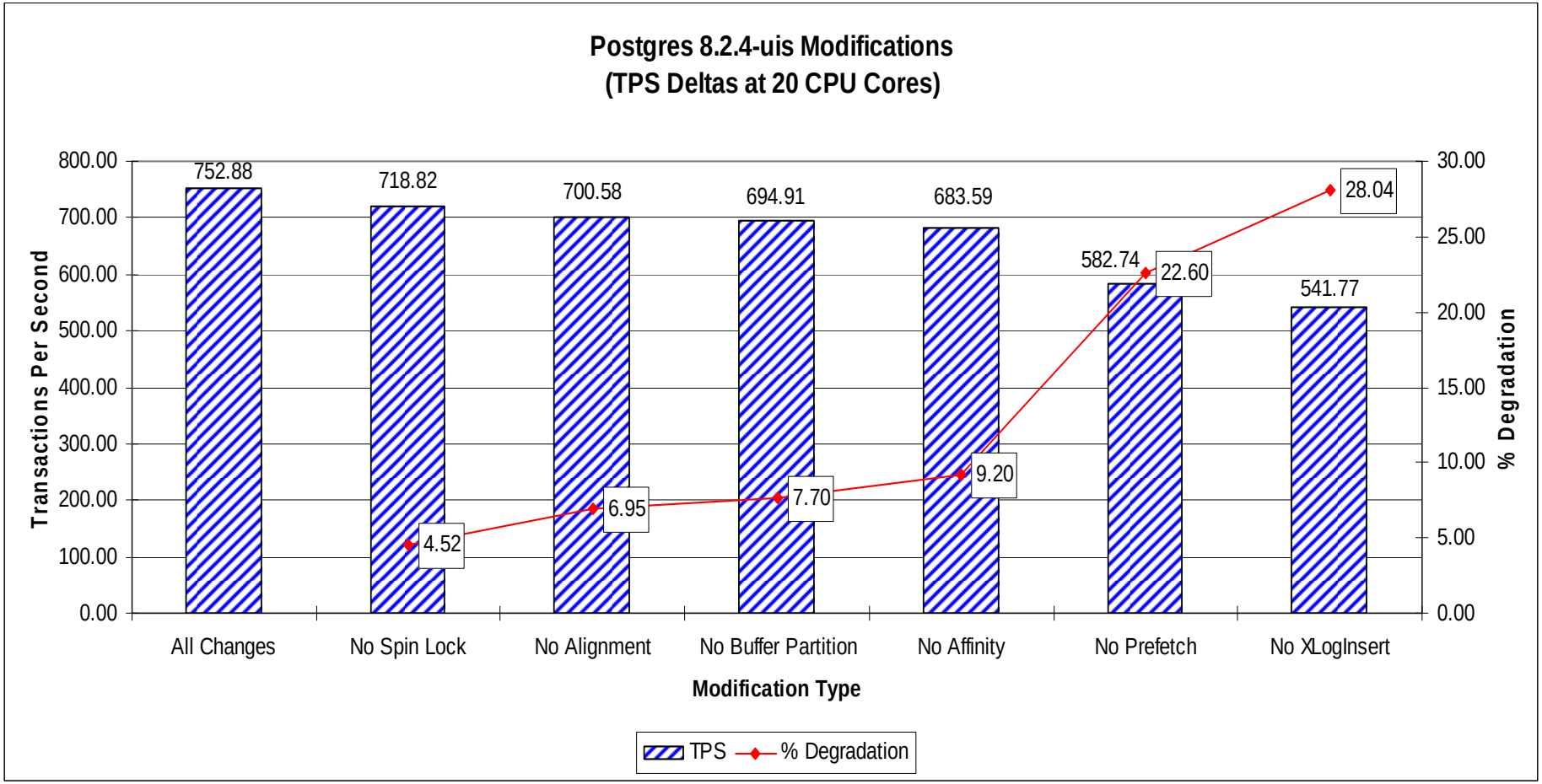
Postgres 8.2.4-uis Affinity



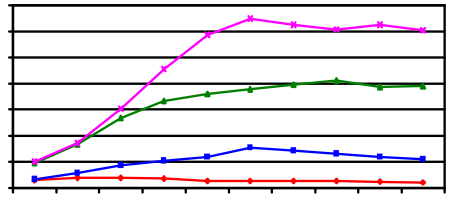
- Affinitize backend to range of CPU cores
  - BackendStartup () and StartChildProcess () modified
  - Allocate on cell by cell basis
  - Lowers cache invalidations for local data
  - Can’t be too restrictive
  - Can’t be too open
  - NUMA concerns

Postgres 8.2.4-uis TPS Deltas





Postgres 8.2.4-uis TPS Deltas

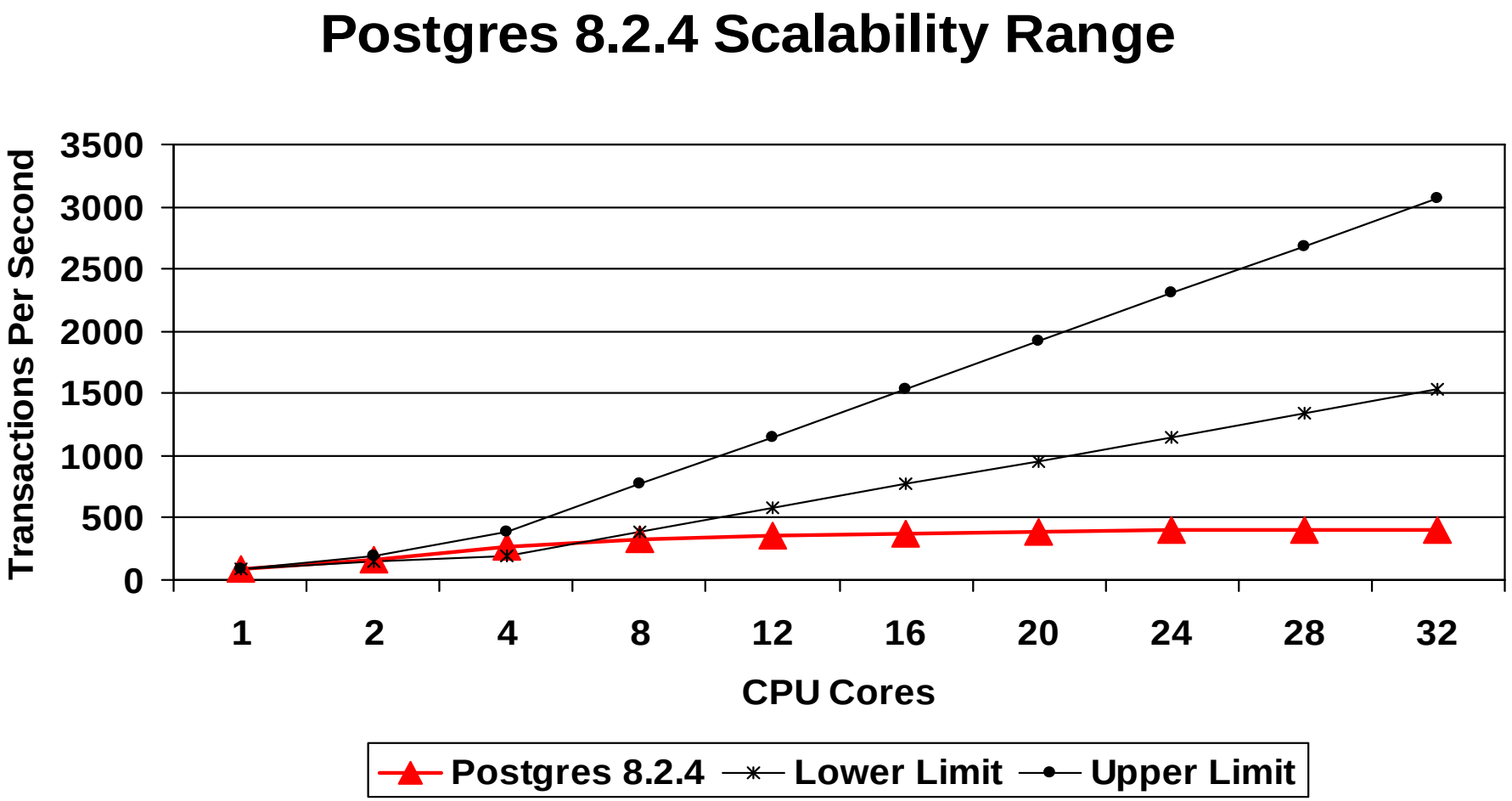


- Largest degradation
  - XlogInsert Lock (~28%)
  - prefetch (~22%)
- Smallest degradation
  - Spin lock (~4%)
- All modifications work in concert

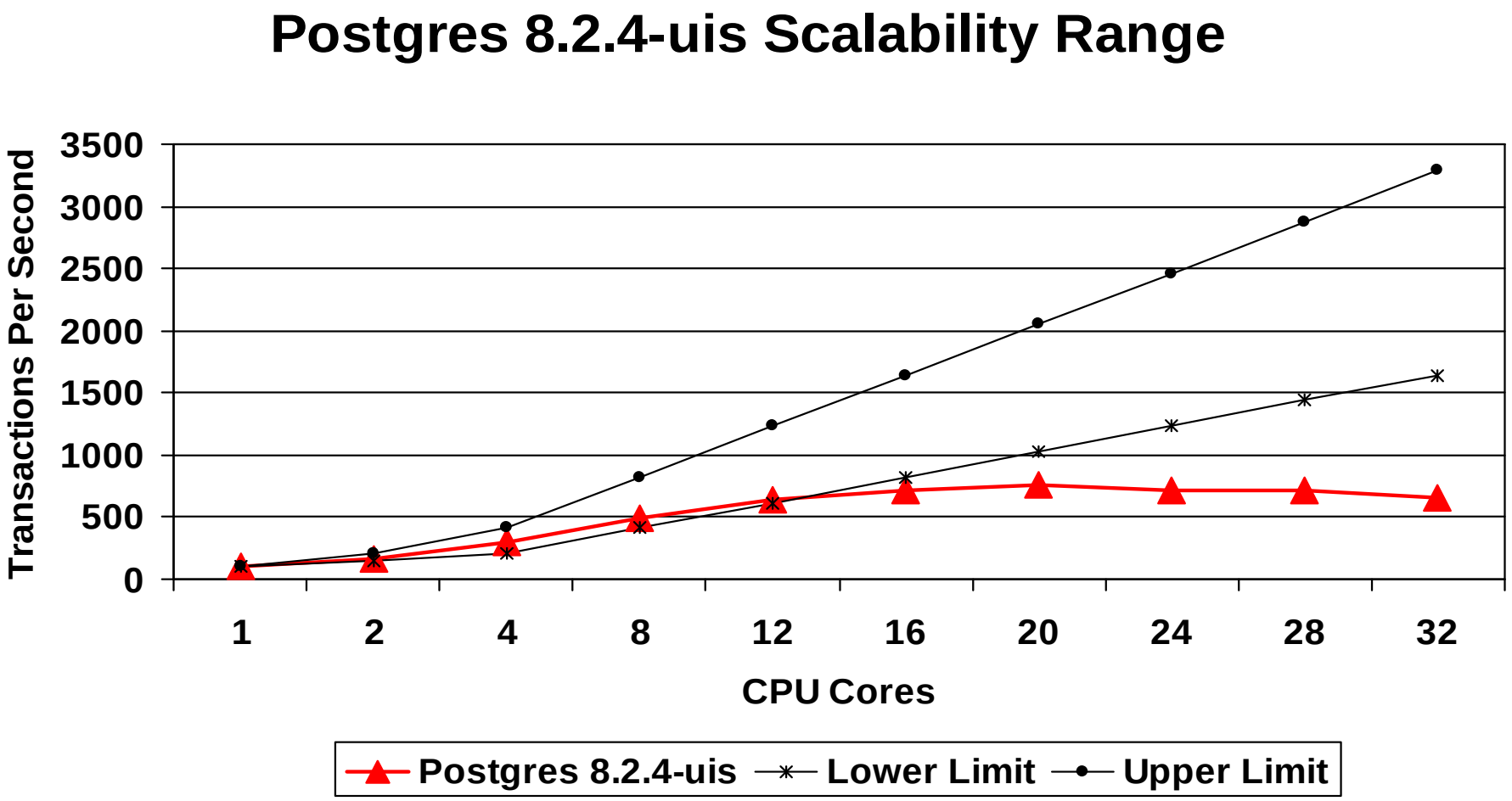
Scalability Range

- All applications have a scalability range
  - Range 0 is no scaling
  - Range 0.1 to 0.4 shows some scaling
  - Range 0.5 to 0.9 is preferable
  - Range 1 is perfect scaling
  - Range 1 is difficult to achieve
- Following graphs lower range based on 0.5 scaling
- Following graphs upper range based on 1 scaling
- Actual scalability range values depend on single thread performance

Postgres 8.2.4

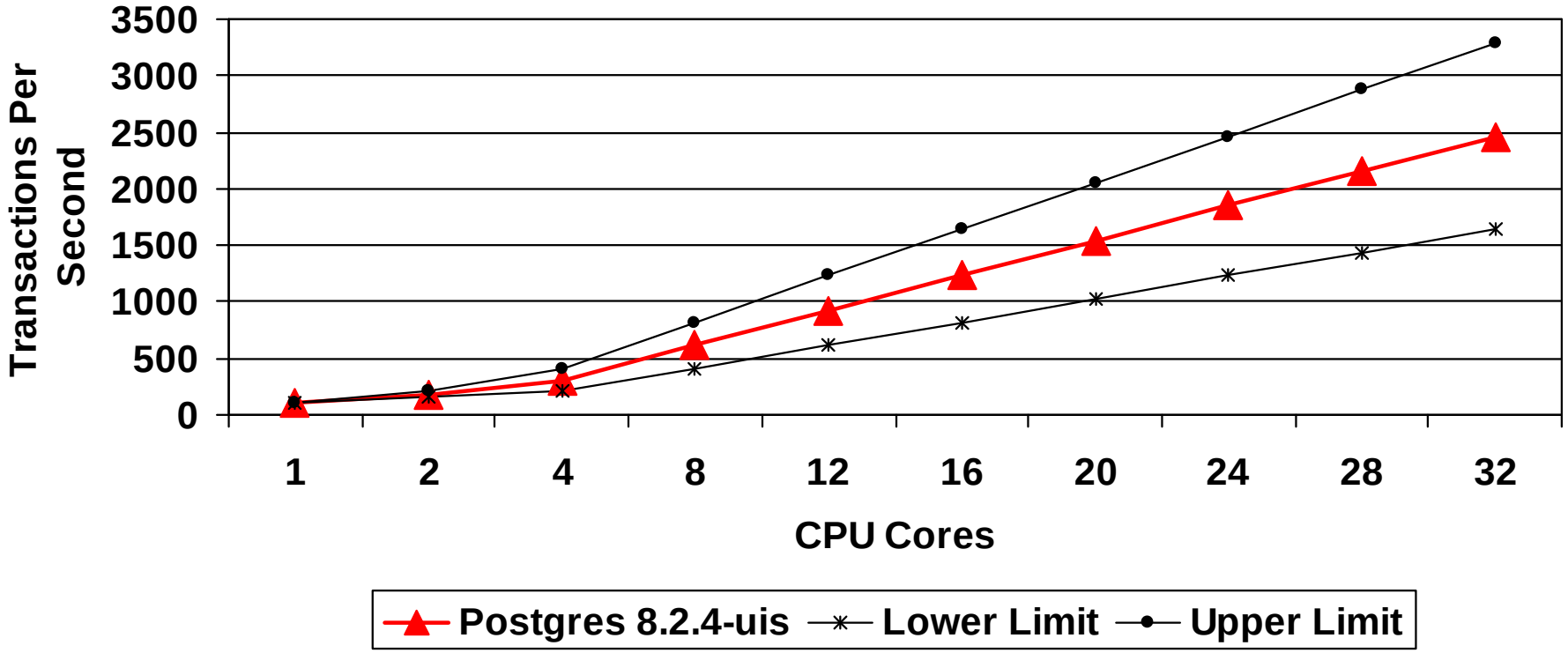


Postgres 8.2.4-uis



Postgres 8.2.4-uis

Postgres 8.2.4-uis Desirable Scalability Range

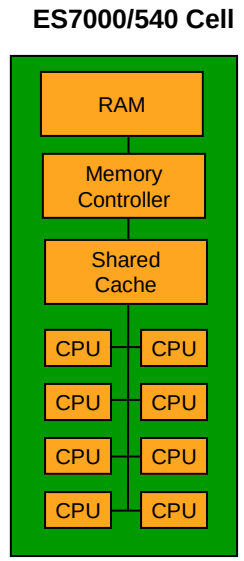


Scalability Range

- Postgres 8.2.4-base scales better than previous versions
  - Scales well to 8 CPU cores, based on the range
- Postgres 8.2.4-uis scales better than base
  - Scales well to 12 CPU cores, based on the range
- Time to start the monitoring process again!

Hardware Architecture

- Cache invalidations
  - Handled quickly when on-cell
  - Slower across cells
- Cross-cell cache invalidations increase quickly when data structures are poorly aligned with cache line size
- Every large-scale computing platform, regardless of vendor, will have some sort of hardware related behavior that needs to be considered in performance and scalability work



Software Architecture

- Degradation beyond 20 CPU cores (UIS modifications)
  - Continued decline implies systemic origin
  - Characteristic of software algorithms or data structures reaching inherent scalability limits
- Possibilities include
  - The Usual Suspects: lock protocol, resource mgmt, etc.

Conclusions

- Scalability dramatically improved in 8.2 release
  - Lock partitions significant
- Depending on observation
  - Scales to 20 CPU cores (UIS mods/Postgres 8.2.4-base)
  - Scales to 12 CPU cores (UIS mods/scalability range)
- Impressive single CPU core performance improvement
  - Assisted by 8.2 performance enhancements
- Hardware architecture can affect scalability
  - Important consideration for high-capacity, high-throughput applications
  - Systems of all sizes moving to multi-core processor chips
- Substantial scalability and performance challenges remain
  - Algorithm scalability limits exist (especially >= 16 CPU cores)