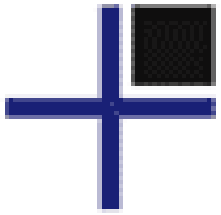
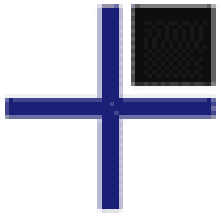


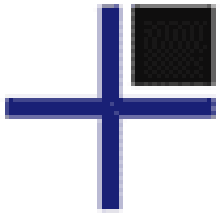
A Batch of Commit Batching

Greg Smith and Peter Geoghegan
2ndQuadrant



Latency components

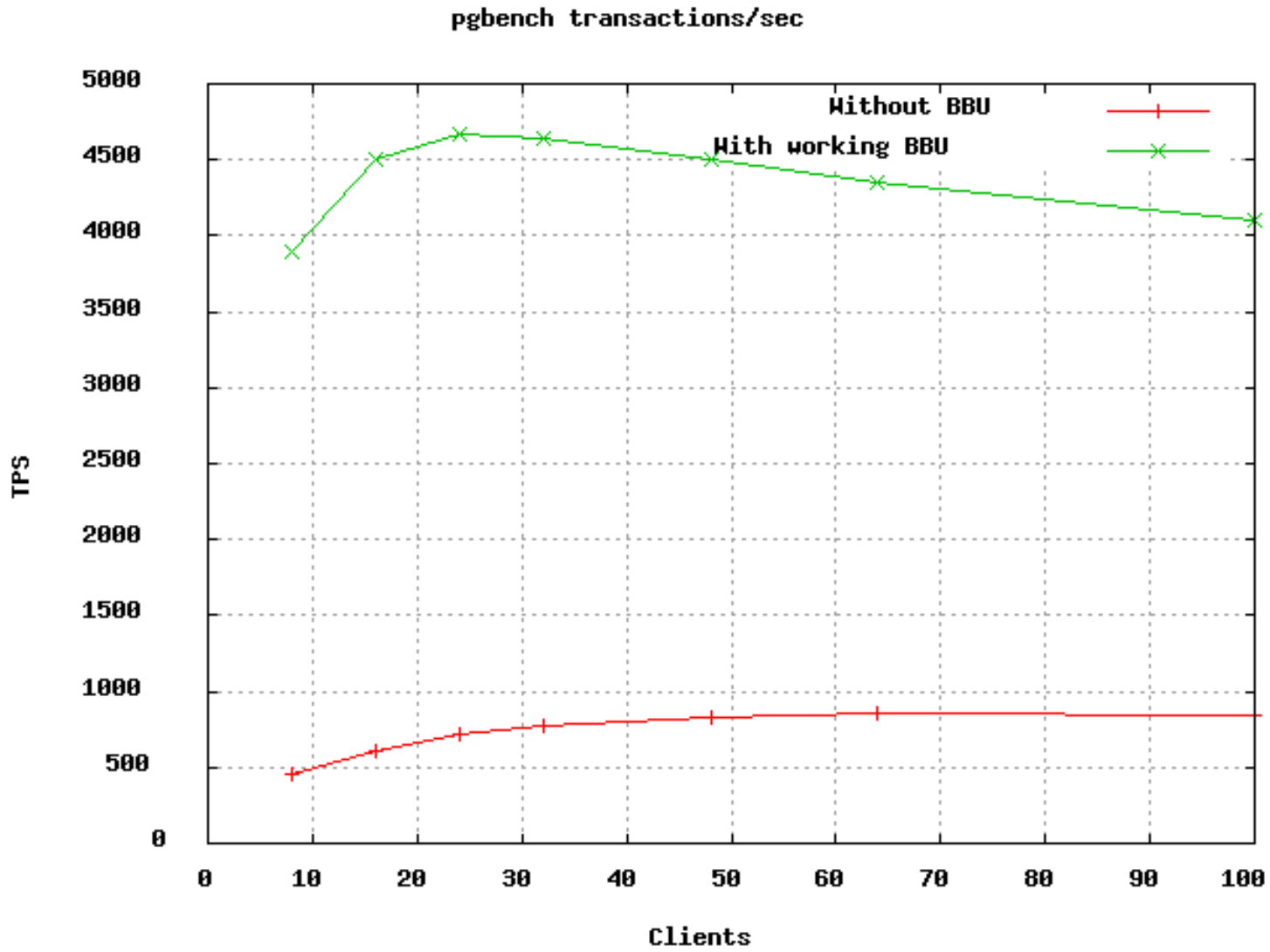
- Local commit
  - Based on hard drive speed
- Network transfer time
- Remote commit
- Parallel throughput increases don't help
  - RAID0
  - Split work among multiple clients

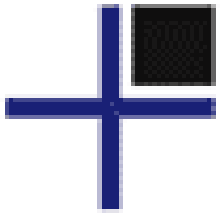


Storage Latency

Type	Latency (ms)	Transactions / Second
7200 RPM	8.3	120
10K RPM	6.0	167
15K RPM	4.0	250
SATA SSD	0.22	4500
Battery-backed Write Cache	0.2	5000
Flash Card	0.1	10000

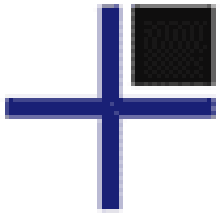
Latency impact on throughput



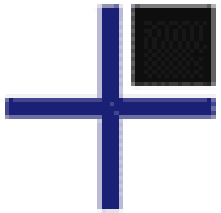


Synchronous Performance

- Full duplex communication
- Reply messages have only write location
  - 64 bytes
- Limited by network plus WAL write time
- Internet is approximately ½ speed of light

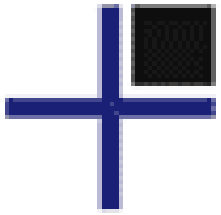


Light is pretty fast, right?



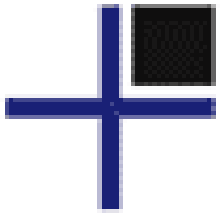
Observed latency

- >10ms common even for close local areas
- 80ms to cross the US or Atlantic Ocean
- 150ms to cross the Pacific



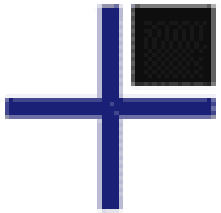
Network Latency

Type	Latency (ms)	Transactions / Second
1Gbps	0.07	14286
100Mbps	0.3	333
Baltimore -> New York City	15	67
Baltimore -> San Francisco	83	12
Baltimore -> Netherlands	100	10



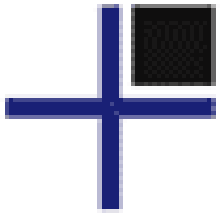
Commit Batching

- Wrap multiple statements into one transaction
  - BEGIN
  - INSERT …
  - INSERT …
  - COMMIT
- One durable write (write plus flush) per transaction commit
- Multiple statements can be grouped into each physical commit
- Similarly, multiple clients worth of commits can be grouped into less physical commits
- Vital to high latency sync rep case



Synchronous Replication

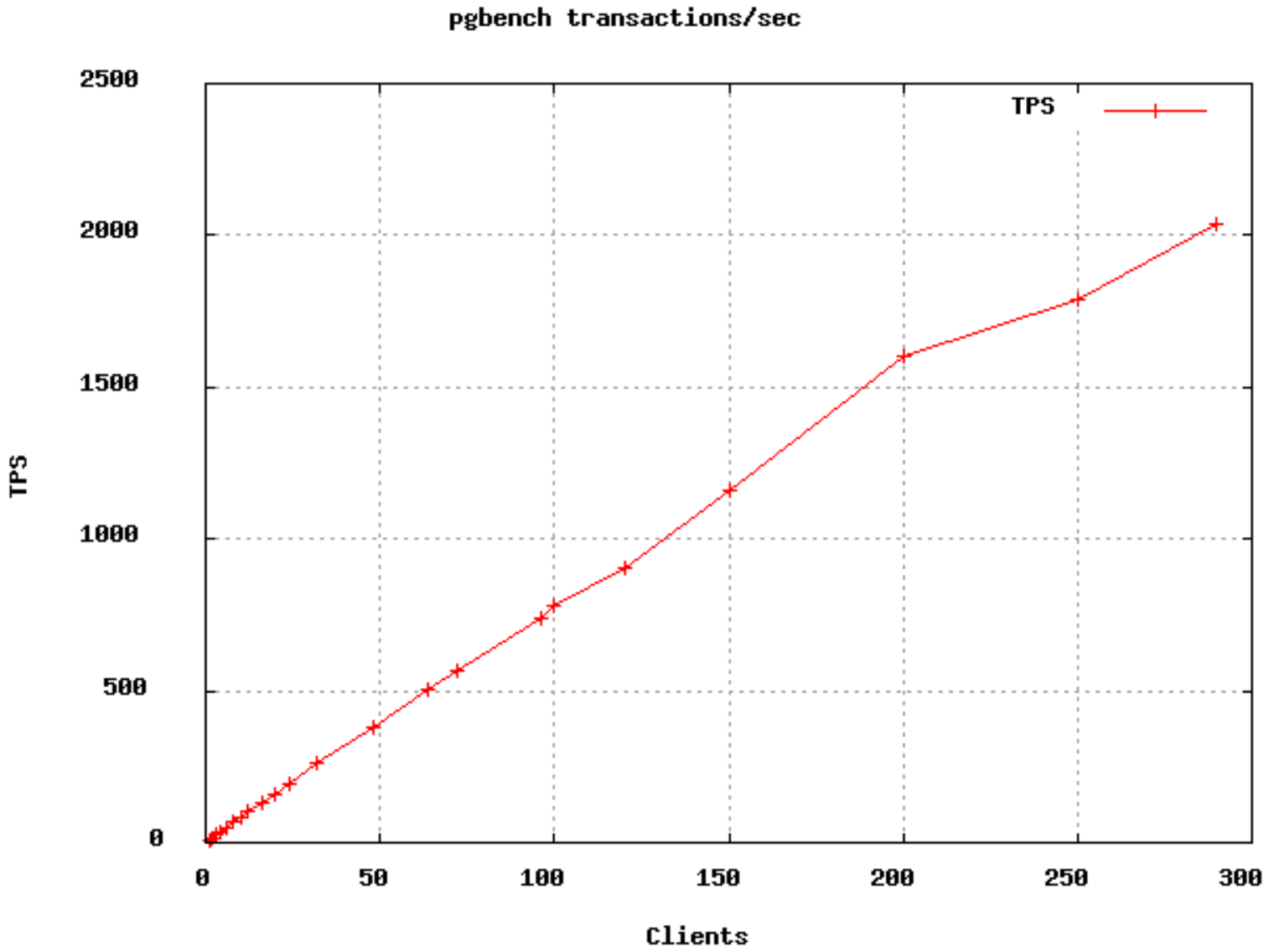
- Zero Data Loss replication
- Efficient – thousands of TPS in tests
- One active synchronous standby
- Transaction controlled

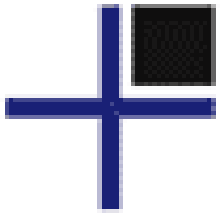


WAL Latency + Sync Rep Test

- Master in Baltimore
  - BBWC to limit its overhead
- Standby at Casa 400, Amsterdam
- Commit rate measured with INSERT statements
- Measured ping time >=100ms
- Typical sync commit time >=112ms
- Theoretical single client max = 10 TPS
- Measured single client rate = 7 to 8 TPS
- How does it scale?

Sync rep group commit, 112ms

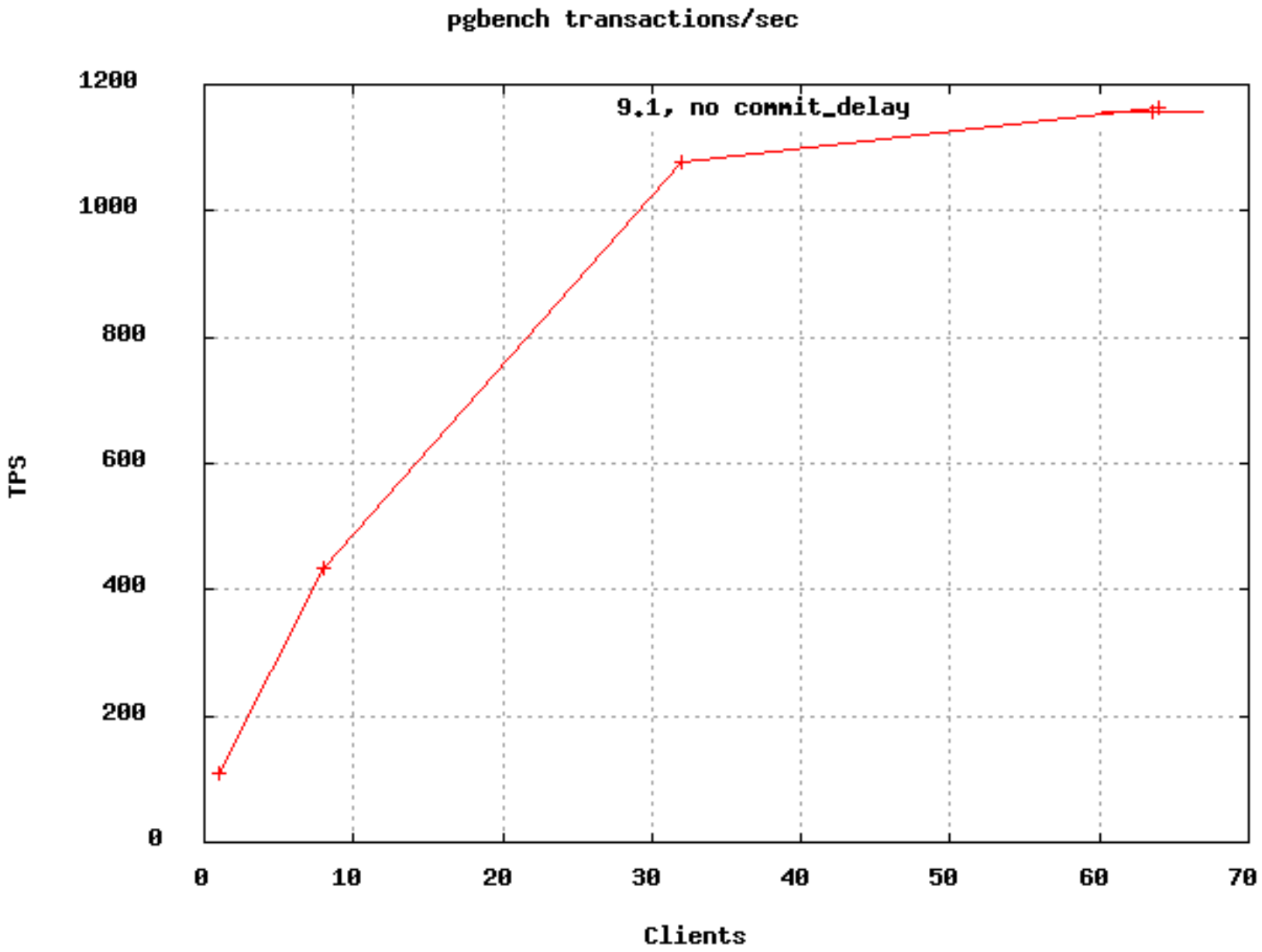




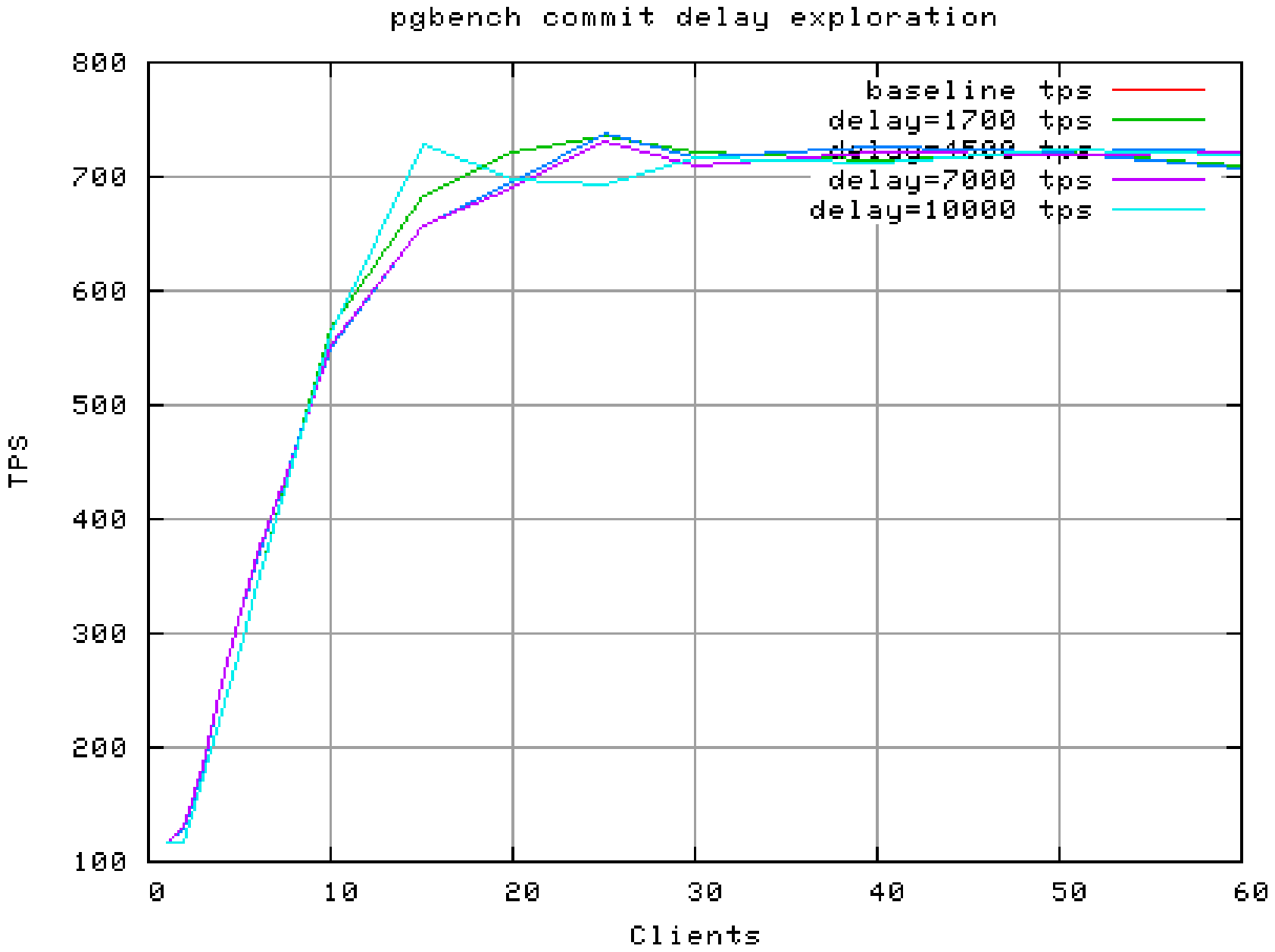
Local group commit in 9.1

- Sync group commit is almost linear with client count
- Hundreds of commits in each commit disk flush
- Local commit rates should do the same
- ...but they didn't. Why?

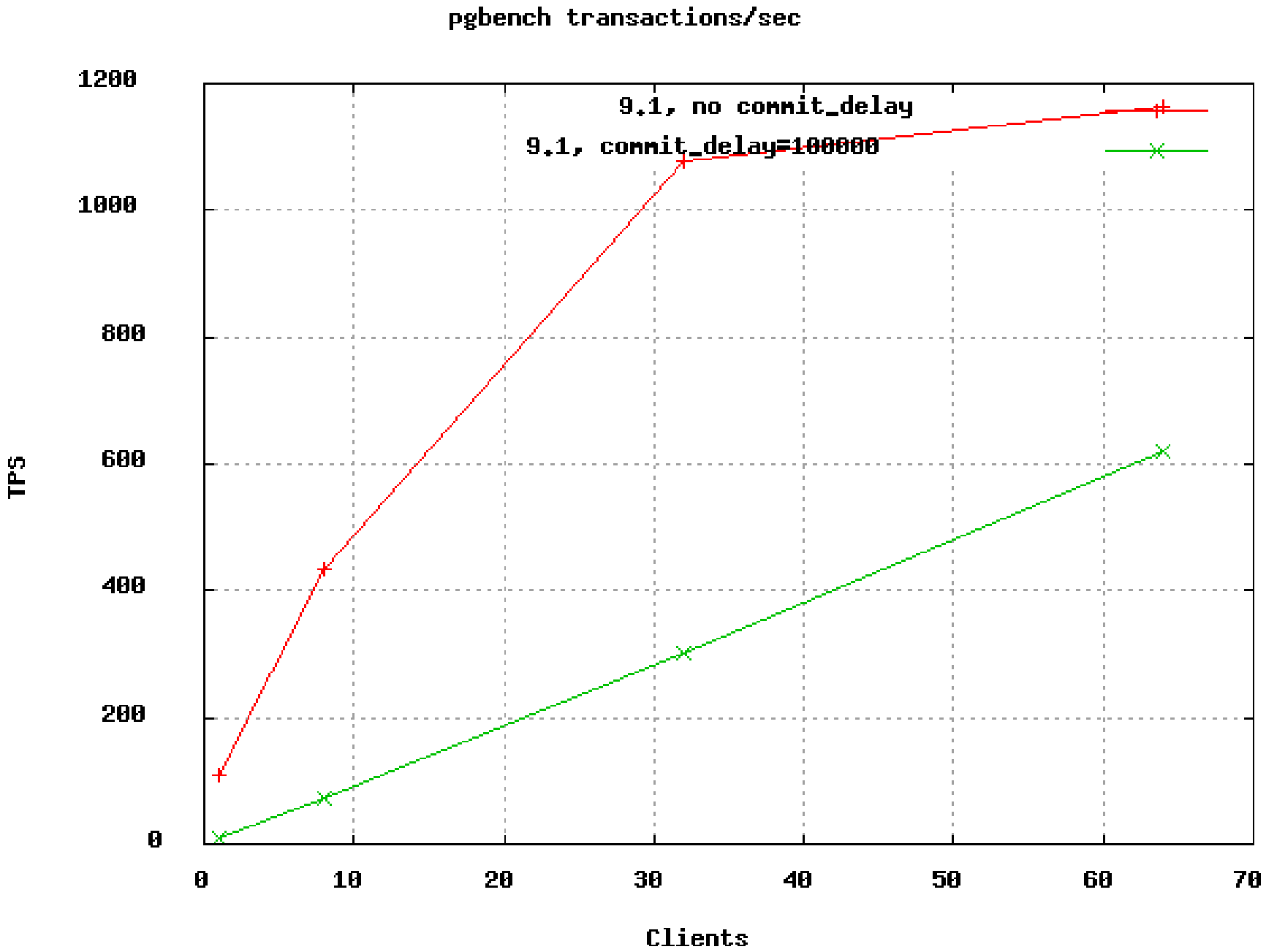
9.1 INSERT scaling, 8ms

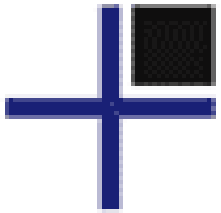


Commit Delay in 8.3



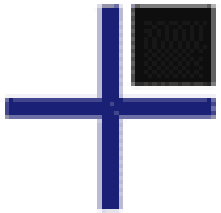
Counter Tuned INSERT scaling





Existing local group commit

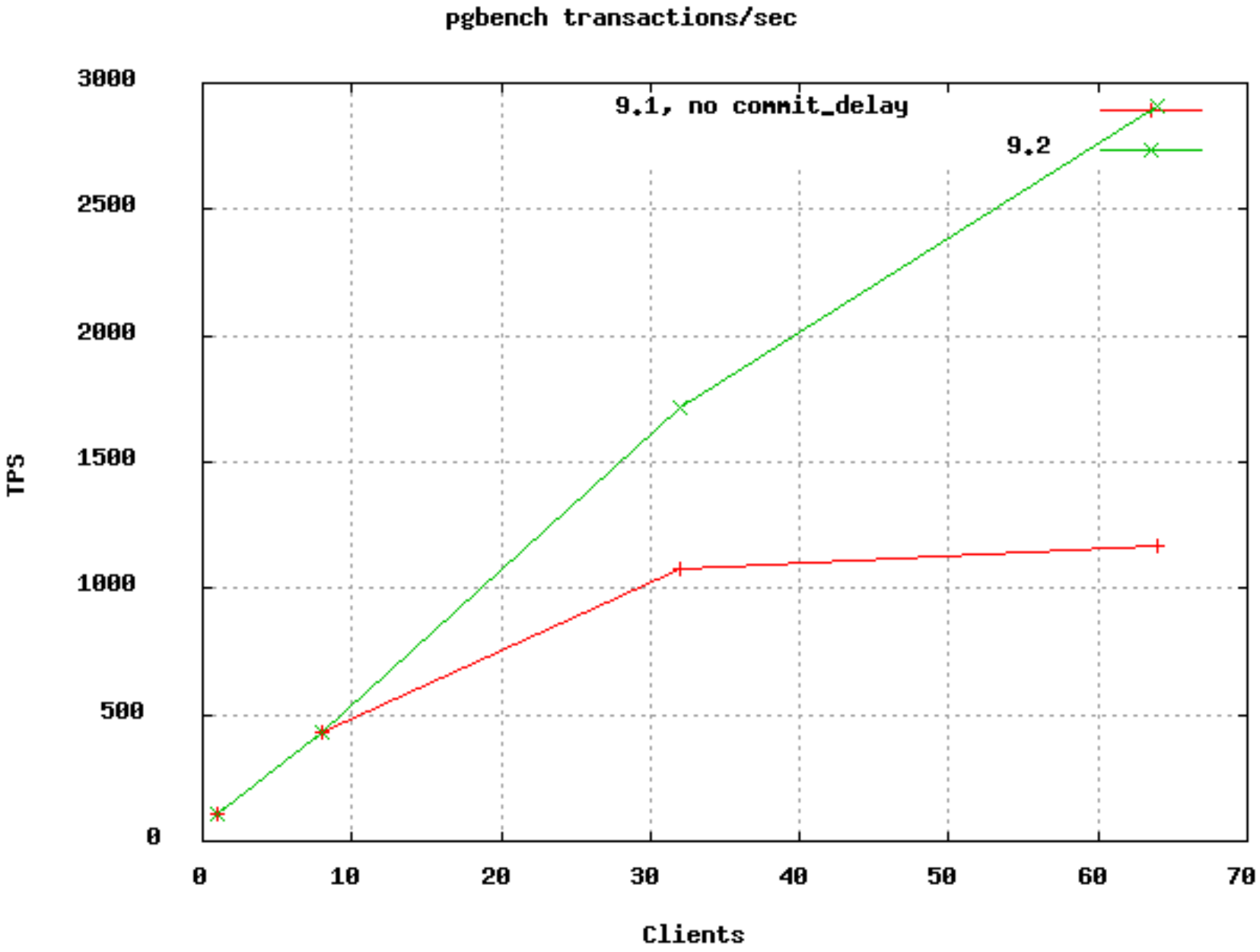
- Write-ahead Log disk flush (fsync) allows commit batching
  - XLogFlush checks if others have pushed forward the known flushed point of LogwrtResult.Flush (a XLogRecPtr) with a “fast path” check
- Only 1 client can flush data to disk at any time; only one person can call XLogWrite.
  - Synchronized with WALWriteLock
  - XLogWrite function updates shared memory
  - Exclusive lock means only one client can hold it at a time
  - Measured as a heavy bottleneck
- Optimal behavior for high concurrency has few physical commits
  - 100 clients? Odds of commit are 1%
- Start by refactoring the sync rep code for local commits

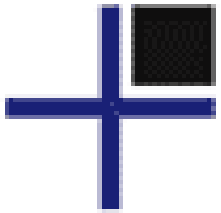


Improved 9.2 group commit

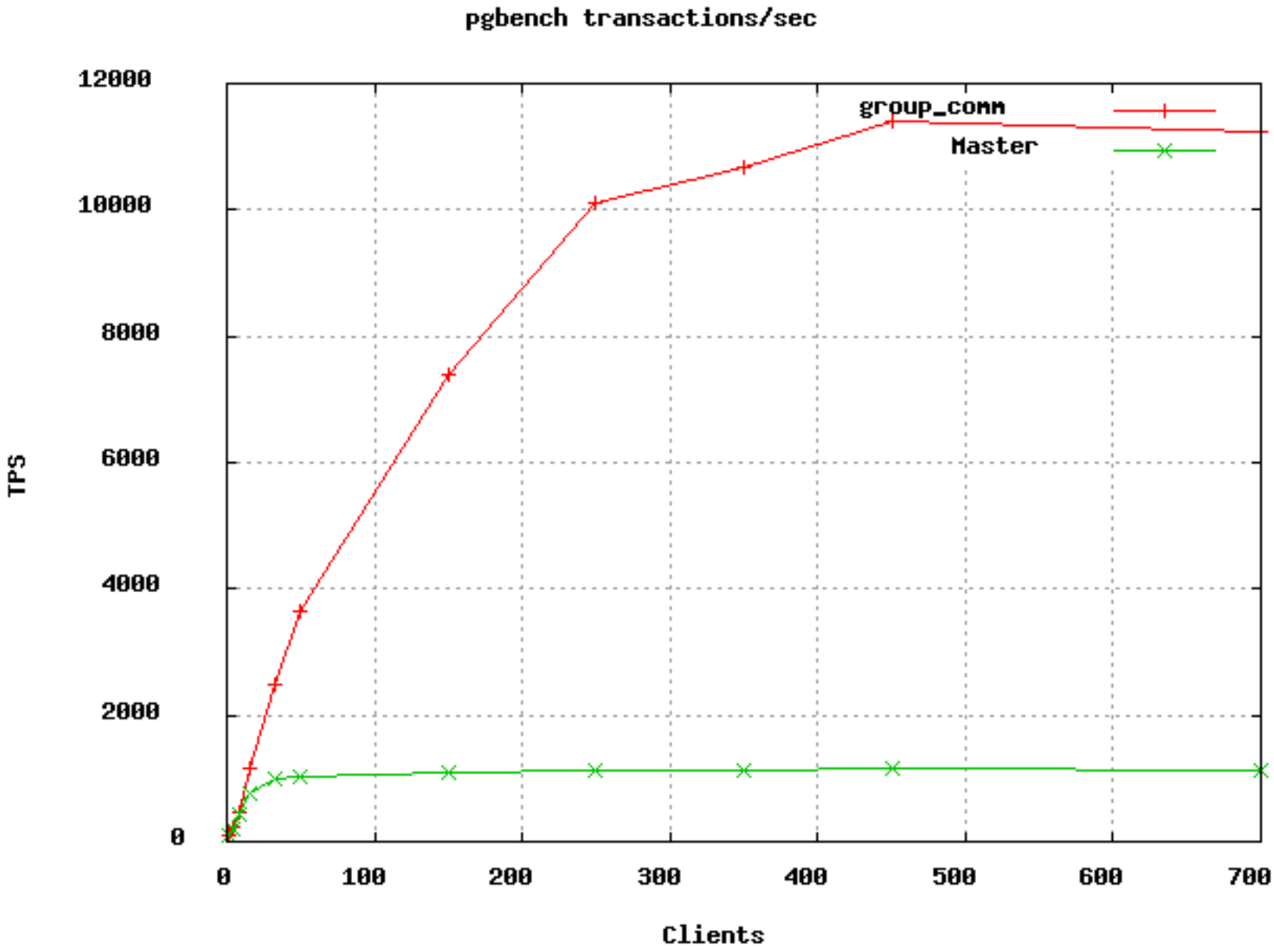
- Assume most clients will fail to commit
- Try to obtain a Lightweight Lock (LWLock)
- Clients wait to become the commit leader
  - Only the leaders ever get the lock
  - Most clients never acquire the WAL write lock
  - They only wait for its release, then see the leader took care of them
- Efficient loop trying to become the new leader
  - Assume the current leader will do the work
  - Most time is spent sleeping, not actually acquiring the lock
- Optimized locking wait
  - No delay for 1 or 2 client cases
  - Hundreds of database commits per physical commit possible

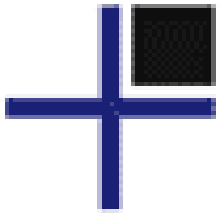
9.2 INSERT scaling





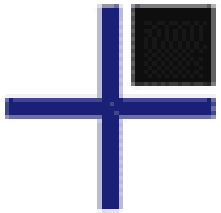
9.2 INSERT scaling





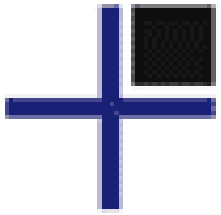
User Selectable Durability

- Set via synchronous_commit
- Two existing modes control master fsync
- Three new modes control sync rep
- World-first from PostgreSQL and 2ndQuadrant
  - Users can control the durability of each transaction
  - All durability levels can co-exist in one application



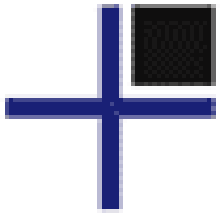
Five durability levels

Sync Standby?	Sync Commit?	Master fsync	Send	Standby fsync	Reply
off	off	off	off	off	off
off	on	on	off	off	off
on	off	off	off	off	off
on	local	on	off	off	on
on	on	on	on	on	on



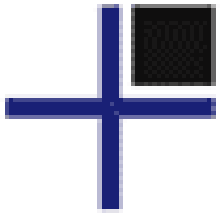
Futures

- Remove commit_delay
- Improve scheduling of commit flush fsync calls



Conclusions

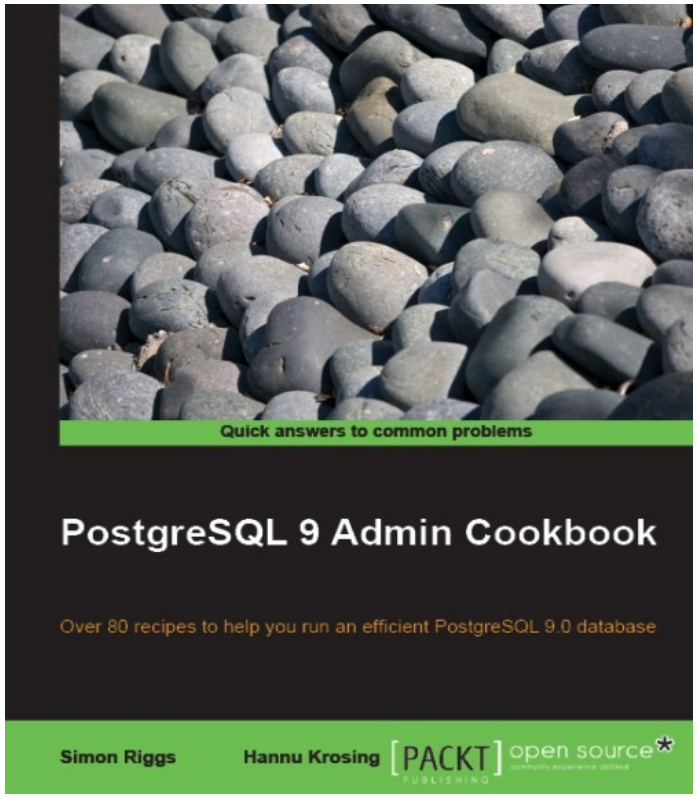
- Battery-backed cache vital for local systems
- WAN overhead dominated by light speed
- Group commit really helps with multiple clients
  - 9.2? Big improvement in sync rep *and* local cases
- Need to only flush what's necessary
- Applications need to be aware of durability

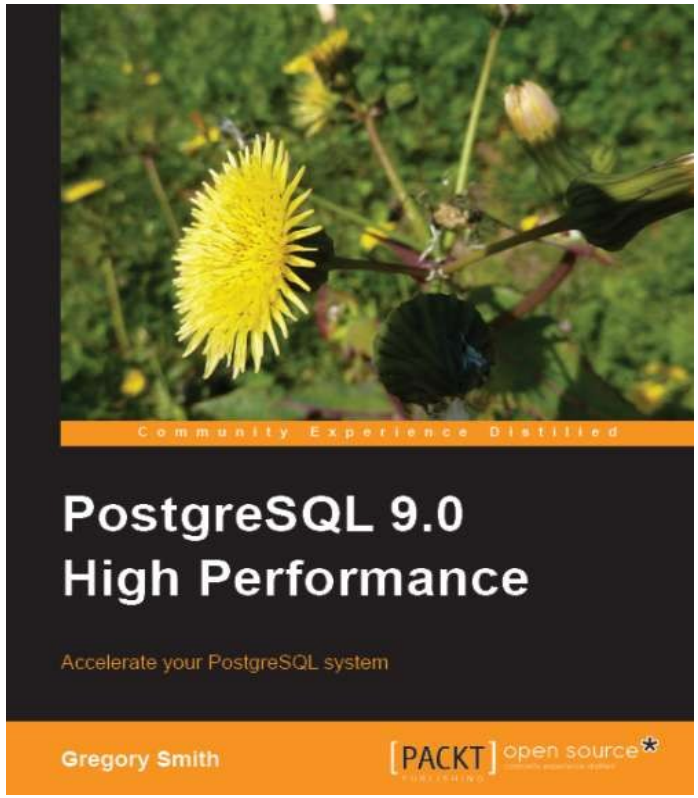


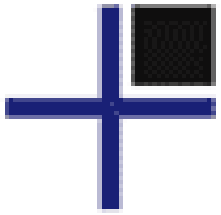
Questions?

PostgreSQL Books

http://www.2ndQuadrant.com/books/







2ndQuadrant Training

- Available now in US, UK, Italy, Germany
- Includes hands-on workshops with instructor interaction
- Database administration
- Performance
- Replication and Recovery
- Real-world experience from production DBAs