Scale and Scalability
Thoughts on Transactional Storage Systems

Liuba Shrira
Brandeis University

Woman’s Workshop, SOSP 2007
Stuff about me

Brandeis professor, MIT/CSAIL affiliate, more stuff about me:

www.cs.brandeis.edu/~liuba

Brandeis is hiring -

come talk to me!
Transactions: Not Just for Databases Anymore

Hides complexity

– Intermediate states from failures
– Intermediate states from concurrency
Thesis of This Talk

• Transactions no longer scary and complex
• Area is exciting
  – You should work on it too
  – (Bias, what bias?)
DB view of Transactions
Traditional Systems View of Transactions

transactional toenail clipper
Things Change

• Long-lived data services commonplace
  – For example, now in data centers

• Reliability now essential
  – Try explain “best-effort” to Grandma

• Ad-Hoc reliability is no bargain
More Things Change

• Transactional properties “deconstructed” into separable properties
• Efficient specialized techniques for individual properties
  – For example, atomic metadata for standard file systems (Ext3), or transactional ZFS
  – SOSP 07 papers:
    “transactional memory” (no durability),
    “Sinfonia” (in memory),
    “I/O shepherding” (no CC)
Even More Things Change

• Changes in technology have eliminated the need for some of the most complex mechanisms

• This is the focus of my talk.

• First, some definitions …
Transactions

• ACID
  – Atomicity (all-or-nothing)
  – Consistency (invariants observed)
  – Isolation (no visible intermediate states)
  – Durability (committed effects stay that way)

Invented in the 70’s by Jim Gray, Dave Lomet..
Mechanisms

• Atomicity
  – Write-ahead log (disk-based)
  – After crash or abort, roll back to good state

• Isolation (concurrency control)
  – Two-phase locks
  – Optimistic
  – Type-specific mechanisms
More Mechanisms

• Durability with good performance
  – Make log sequential (cheap to commit)
  – Update in-place in the background
    • “no-force”
  – Sometimes need to “steal”
    • Write uncommitted data to disk
    • Complicates recovery
Even More Mechanisms

• Two-Phase Commit
  – Distributed algorithm
  – Ensures all nodes agree to commit transaction
  – Two rounds of messages
    • Coordinator to participants & vice-versa
  – “Window of vulnerability”
    • System hangs if coordinator crashes at wrong moment
Transaction systems were complex

Because -
the name of the game was Performance

Still is -
but the game is changing..
Old School: disk-oriented solutions

STEAL

disk

memory
Today: system fits in main memory

No STEAL
Old-School: Resource Utilization

• **Goal:** better use of single-node resources

• Multiprogramming
  – I/O concurrency
  – If one transaction blocks, run another

• Concurrent B-Trees
  • Avoids blocking
  • Complex type-specific recovery
Today’s Resource Utilization

• No disk stalls (system fits in memory)
• “Snapshot Isolation”
  – Long read transactions don’t conflict
• Can run many transactions to completion
  – No need for multi-threading
  – No concurrent B-Trees
    • No type-specific recovery
    • Much simpler
On-grid Computing and Scalability

• Old School
  – Scale with bigger nodes
  – 70s: SMPs
  – 80s: shared disk architectures

• Today
  – Scale by adding nodes
  – Must partition storage
  – Rebalance on-line
Scalability Today

• Scalability natural for systems that use replication
  – Mechanisms exist for adding, removing nodes & resources

• Larger nodes still useful
  – Use less power
  – Multicore
    + transactional memory? (b.t.w. invented by Herlihy&Moss once Barbara’s students..)

Potentially simpler model if can partition –
interesting source of new problems
High Availability

• Old School
  – Afterthought, “glued on”
  – Off-site backups

• Today
  – Basic requirement
  – Hot-standby part of basic architecture
More Availability

• Built-in replication
  – Eliminates need for disk-based commit log
    • Instead, replicate the log, improve performance!
      (e.g. Harp SOSP 92)
  – Exploit replicas for peering, load-balancing
    • New problems
Fear of Commitment

• Old-School: reject 2-phase commit
  – What if coordinator fails? (All block)
  – Security + autonomy issue

• Today, still heresy for some, but
  – Coordinator & participants live in data center
  – Data center can be held responsible (billable)
    • Replication: coordinator & participants highly available (see Sinfonia)
Large-Scale Systems for the People

In the old days, only privileged researchers could play with scalable systems

• Today, there are playgrounds for all
  – Emulab
  – PlanetLab
  – Google/IBM new Cloud Computing?
New Technology Diet

• Transaction systems are shedding complexity weight
• You can now use transactions in your high-performance large-scale systems
• And evaluate their performance in Internet gyms
Transactional systems today: publishable..

SOSP 07:
“Sinfonia..”
“Commit Barrier scheduling..”
“Transactional memory..”
“I/O Shepherding..”
.. And putting my money where my mouth is...

Transactional cooperative caching in a WAN (OOPSLA 03),

Transactional cooperative caching for disconnected clients (ECOOP 05),

Transactional snapshots (USENIX 06),

Typed transactional leasing (in review)
High-order bit:

• Transactions are useful
  less complex than you may think
• Area is exciting
  – You should pay attention