OPERATING SYSTEMS SHOULD PROVIDE TRANSACTIONS

Donald E. Porter and Emmett Witchel

The University of Texas at Austin
Example: browser plug-in upgrade

write new plug-in binary
start browser, old config, old plug-in arguments
**corrupt data files**
exec post-install script (updates browser config)

- API can’t ensure consistent updates to OS resources
- Concurrency and crashes cause subtle inconsistencies
System Transactions

- Express consistency requirements to OS
- Transaction wraps group of system calls
  - Results isolated until commit
  - Interfering operations automatically serialized
- Long-overdue OS feature
  - Natural abstraction
  - Solves important problems
  - Practical implementation
Transactional Software Install

sys_xbegin();
apt-get upgrade
sys_xend();

- A failed install is automatically rolled back
  - Concurrent operations are not
- System crash: reboot to entire upgrade or none
- Concurrent apps see consistent state
System Transactions

- Operating systems should provide them
- Operating systems can provide them
The POSIX API is broken

- System resources have long-standing race conditions
  - Time-of-check-to-time-of-use (TOCTTOU)
  - Temporary file creation
  - Signal handling
- Correct, concurrent apps need system-level isolation
- Multi-core chips raise importance of concurrency
System-level races

if(access("foo")) {
    fd = open("foo");
    ...
}

foo == secret
Complex work-arounds

- **TOCTTOU**: users write their own directory traversal
  - `openat()`, `fstatat()`, etc.
  - User re-implements filename translation
- Race between `open/fcntl`
  - Add `CLOSE_ON_EXEC` flags to 15 system calls
- Temporary file creation libraries
  - `mkstemp`, `tmpfile`, etc.
Work-arounds don’t work

- Complex APIs do not yield secure programs
- Experts can’t even agree
  - `mkstemp` man page:
    "Don’t use this function, use `tmpfile(3)` instead."
  - [www.securecoding.cert.org - VOID FI039-C](https://www.securecoding.cert.org):
    "It is thus recommended that...`mkstemp()` be used [instead of `tmpfile()`]"
- Transactions can fix the problem
sys_xbegin();
if(access("foo")) {
  fd = open("foo");
  read(fd,...);
  ...
}

sys_xend();
Transactions solve important problems

- **Applications**
  - Replace databases for simple synchronization
  - Support system calls in transactional memory apps
  - Tolerate faults in untrusted software modules
  - Atomically update file contents and access control list

- **Easier to write OS extensions**
  - System Tx + Journal = Tx Filesystem
Hasn’t this already been done?

donporter@wesley:~$ man transaction
No manual entry for transaction
Related Systems

- Similar interface, different implementation
  - QuickSilver [SOSP ‘91], TABS [SOSP ‘85]
    - Weaker guarantees
  - TxF, Valor [FAST ‘09]
    - Only file system transactions

- Different interface, similar implementation
  - Speculator [SOSP ’05, OSDI ’06]

- Terms “transaction” and “OS” appear in paper title
  - TxLinux [SOSP ’07, ASPLOS ‘09]
Can OSes provide transactions?

- **TxOS**: Extends Linux 2.6.22 to support transactions
  - Runs on commodity hardware
- **Rest of talk:**
  - Approach
  - Validation
Version Management

- How to keep old and new data?
  - Need old data to roll back

- TxOS approach:
  - Transactions operate on private copies of data
  - Replace old data structures at commit

- Example: kernel data structures
sys_xbegin();
if(access("foo")){
    fd = open("foo");
    write(fd, "Hi");
}
sys_xend();
Object versioning in TxOS

- Deadlock-free
  - Transactions do not hold kernel locks across syscalls
  - Follows existing locking discipline
- Previous work used 2-phase locking, undo log
  - Prone to deadlock
- Efficient – a pointer swap per committed object
  - Copy-on-write optimizations
Serializing Tx with No-Tx

- Important property for intuitive semantics
  - Supports incremental adoption
- Serialize TOCTTOU attacker
  - Attacker will not use transactions
- Hard to support in software systems
  - Not provided by historical OSes, many STMs
Validation

- Is implementation tractable?
- Is performance acceptable?
Transactions:
- Add 8,600 LOC to Linux
- Minor modifications to 14,000 LOC

Simple API, not a simple implementation
- Hard to write concurrent programs
- Developers need good abstractions

Transactions are worth the effort
Acceptable Performance

- 40% overhead for dpkg install

Speedup compared to unmodified Linux
OSes can support transactions

- Tractable Implementation
- Acceptable Performance
OSes should provide transactions

- Solve long-standing problems
  - Replace ad hoc solutions
- Broad range of applications
- Acceptable cost

http://www.cs.utexas.edu/~porterde/txos
porterde@cs.utexas.edu