

This work is licensed under the Creative Commons Attribution-No Derivative Works 3.0 License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nd/3.0/> or send a letter to Creative Commons, 171 Second Street, Suite 300, San Francisco, California, 94105, USA.

In short:

- You are free to copy, distribute, and transmit the work**
- You must attribute the work to the author, but not in a way that suggests he endorses you or your use of the work**
- You may not alter, transform, or build upon this work.**
- To redistribute, you must make it clear to others how this work is licensed.**



NEW!

Now with more
happy people!

Merging Processor Sets and Pools

Colin Gordon

Brown University

Solaris Kernel Intern

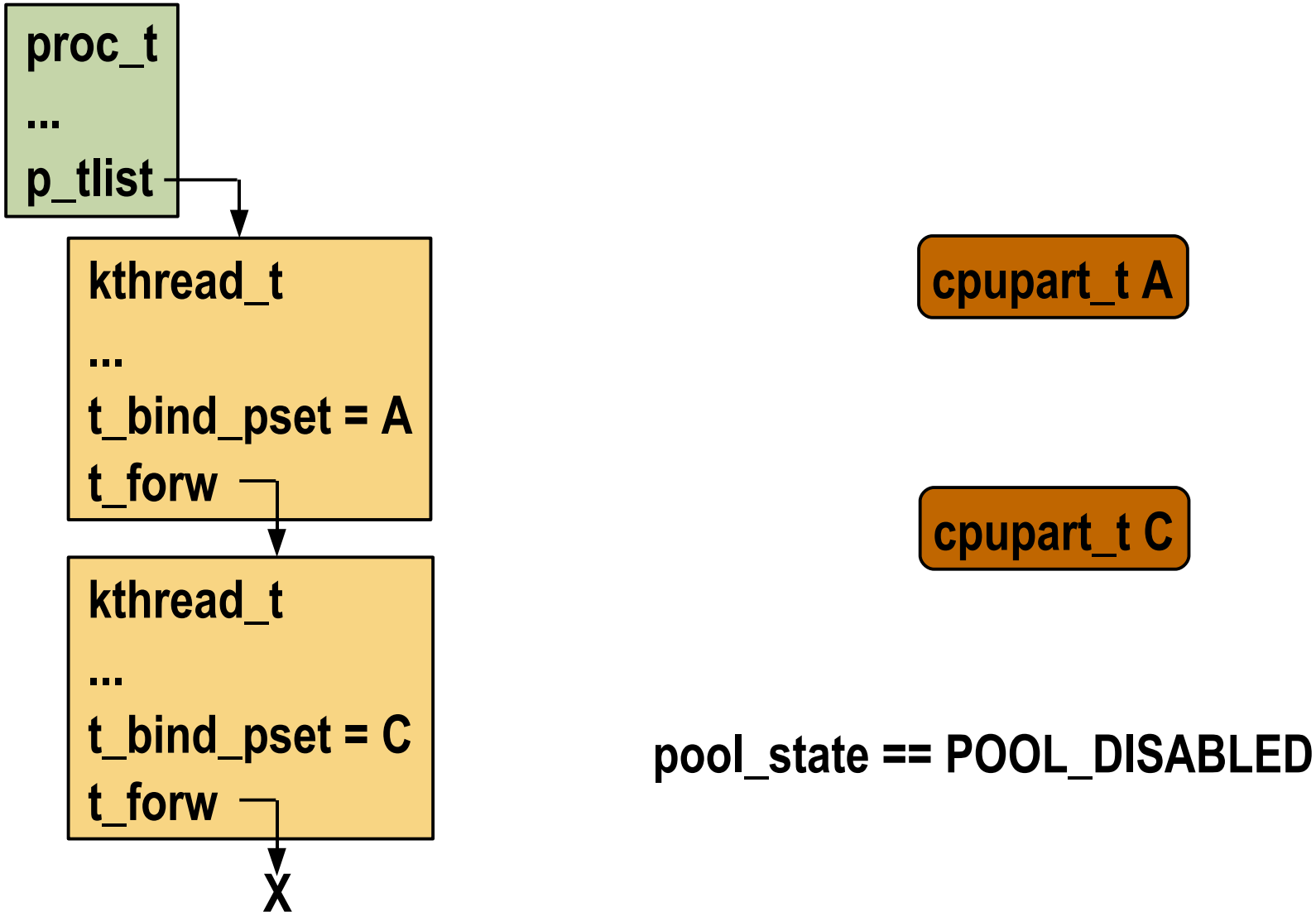
Summer 2007



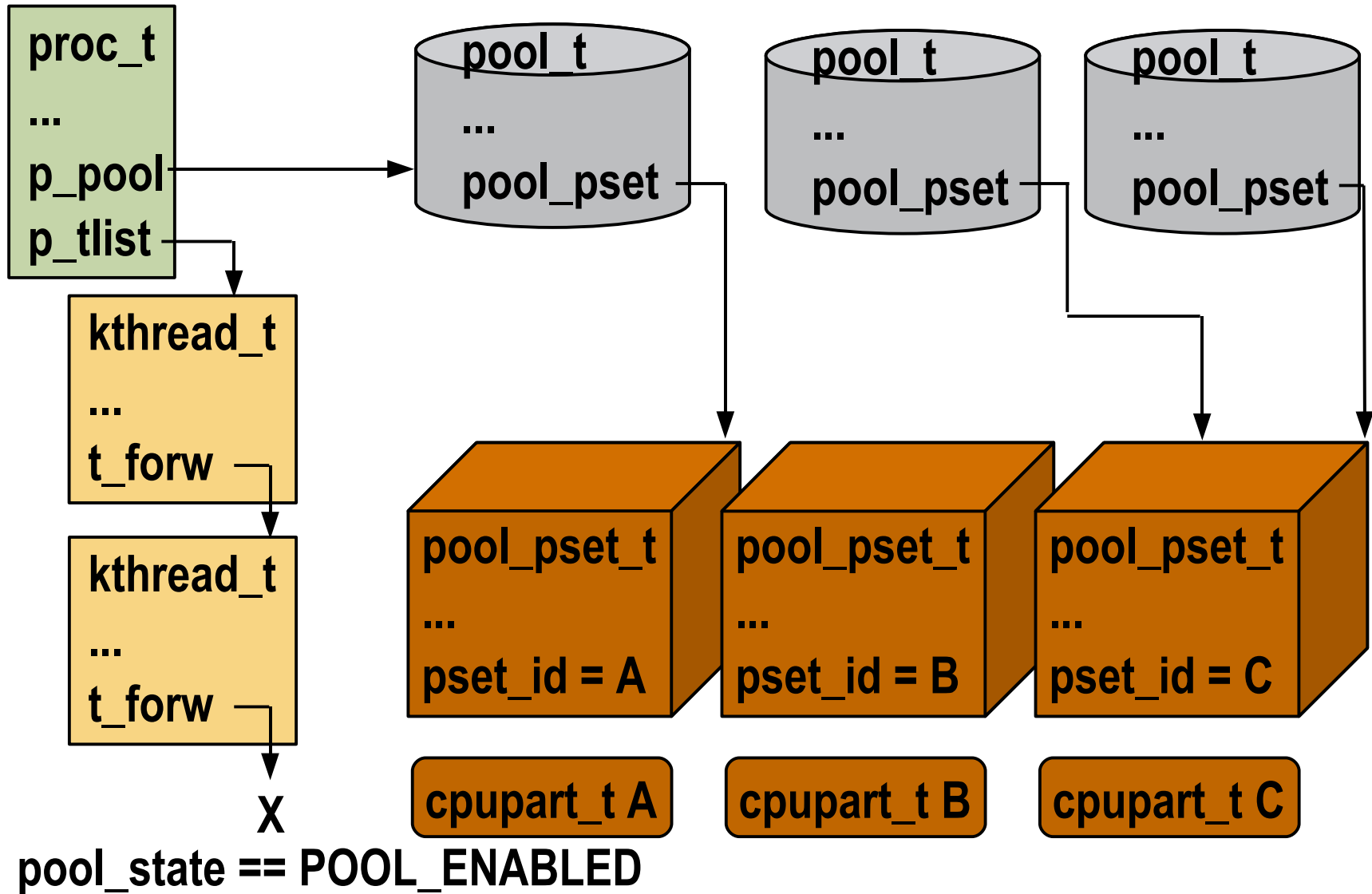
What are these things?

- Processor grouping
- Process binding
- Persistence & policy differences

Old Processor Sets



Old Processor Pools



Motivation

- Processor sets and processor pools do the same thing
- Mutually exclusive? Why disable functionality?
- Zones' dedicated-cpu feature should be usable by default
- Part of the point of the pools framework is consolidating resource management
- Already a common scheduler abstraction

Goals

- Pools and psets both enabled at boot, always-on
- Psets API works as expected in isolation
- Pools API works as expected in isolation
- Both APIs observe both groups of entities
 - > poolstat
- Psets API can't alter pools or processes bound to pools
- Do what makes sense

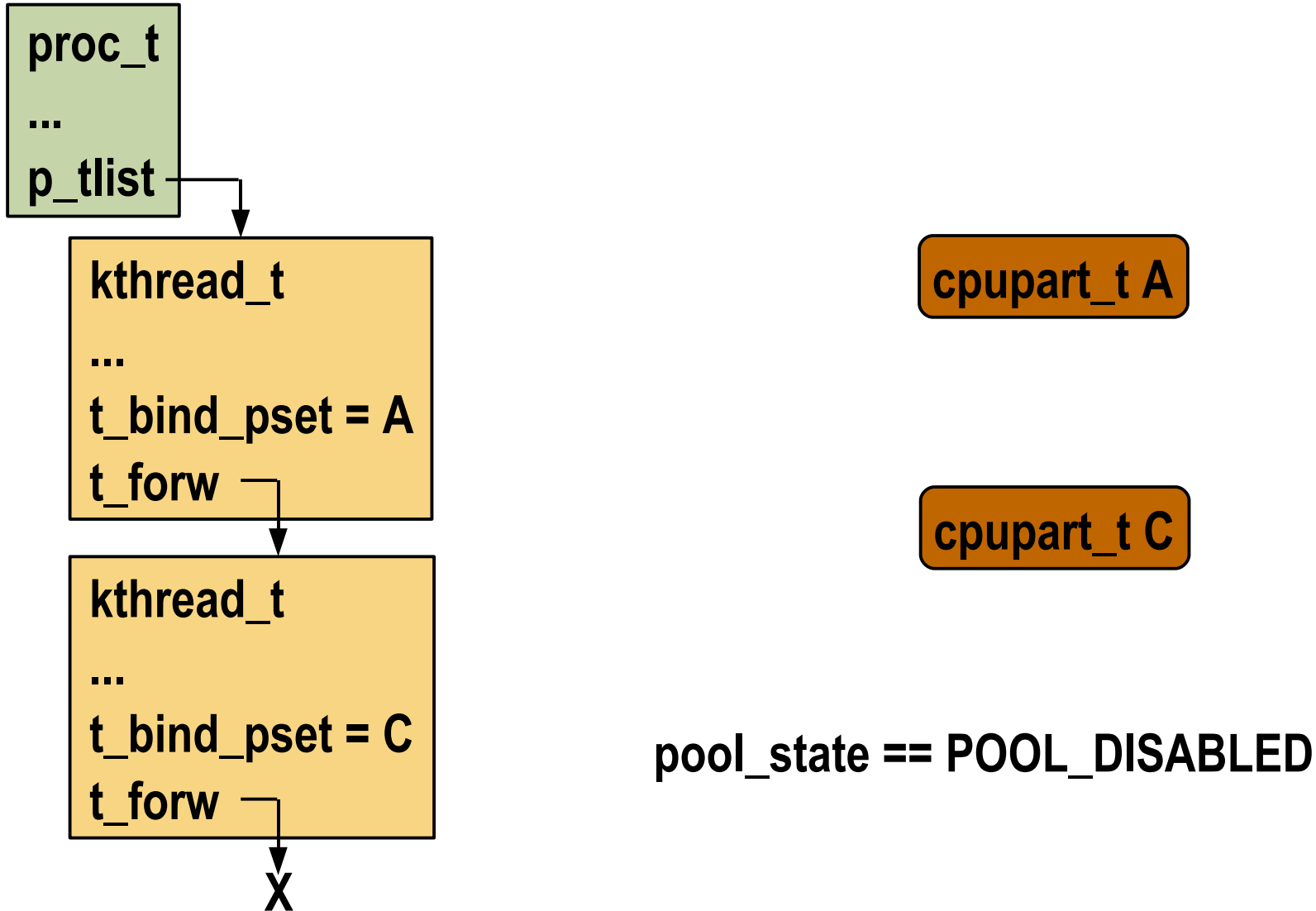
Strategy

- Psets as temporary pools
- Merge binding paths
- Extend pools API for individual threads
- Switch binding and reference counts from per-process to per-thread
 - > Psets **only** (vs. msets, etc., semantic & technical reasons)
- Enable / setup pools at boot
- Never disable either API
 - > pooladm -d / -e: reset properties

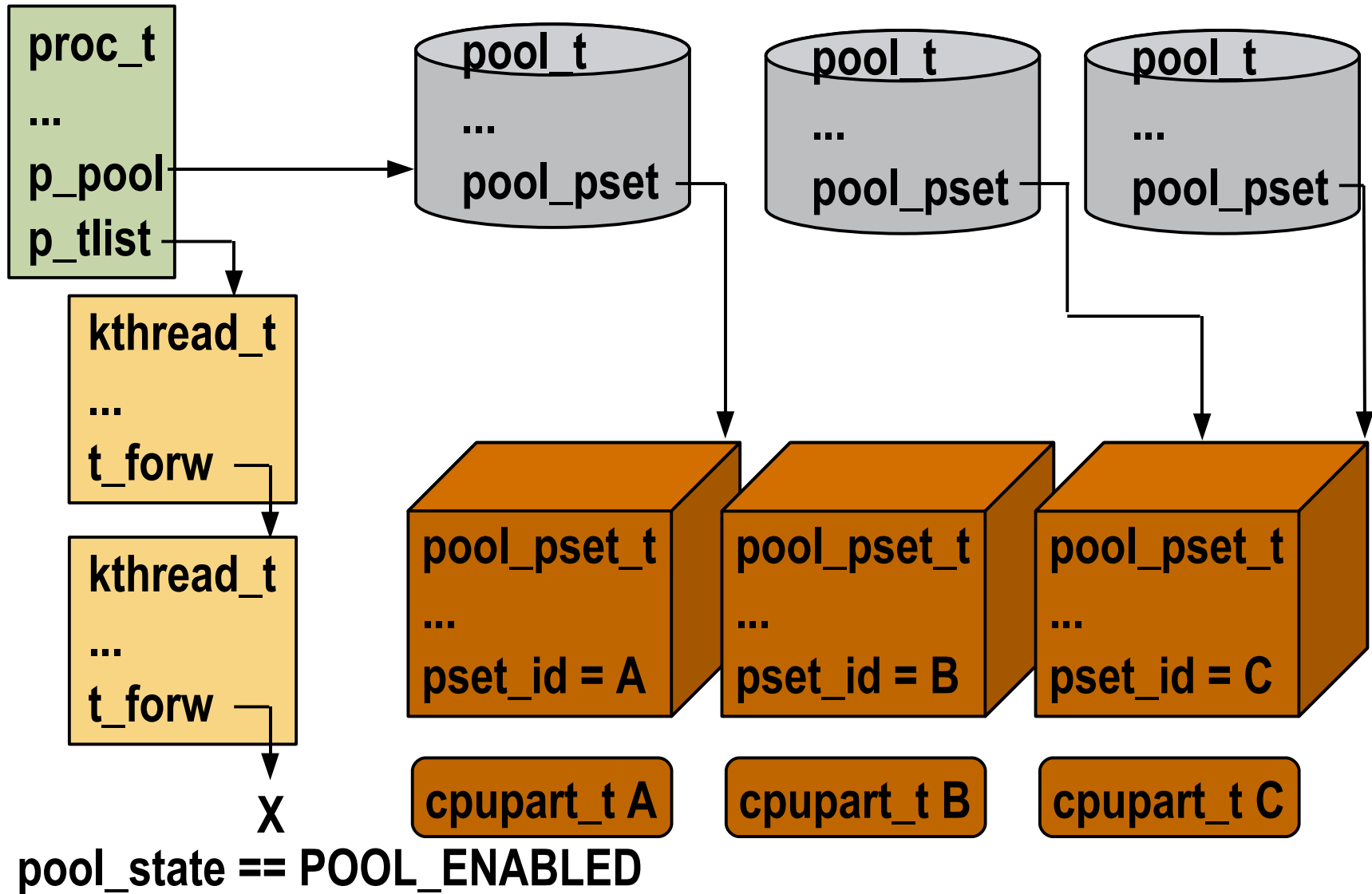
Challenges

- Psets and pools have different semantics
 - > Argument checking
- Random API differences
 - > Previous binding
- Psets granularity: threads. Pools granularity: processes.
 - > Changes to `kthread_t`, `pool_count` refcount
 - > Process synchronization
- Choosing correct behavior for mixed API usage
- General synchronization issues
- libpool assumptions

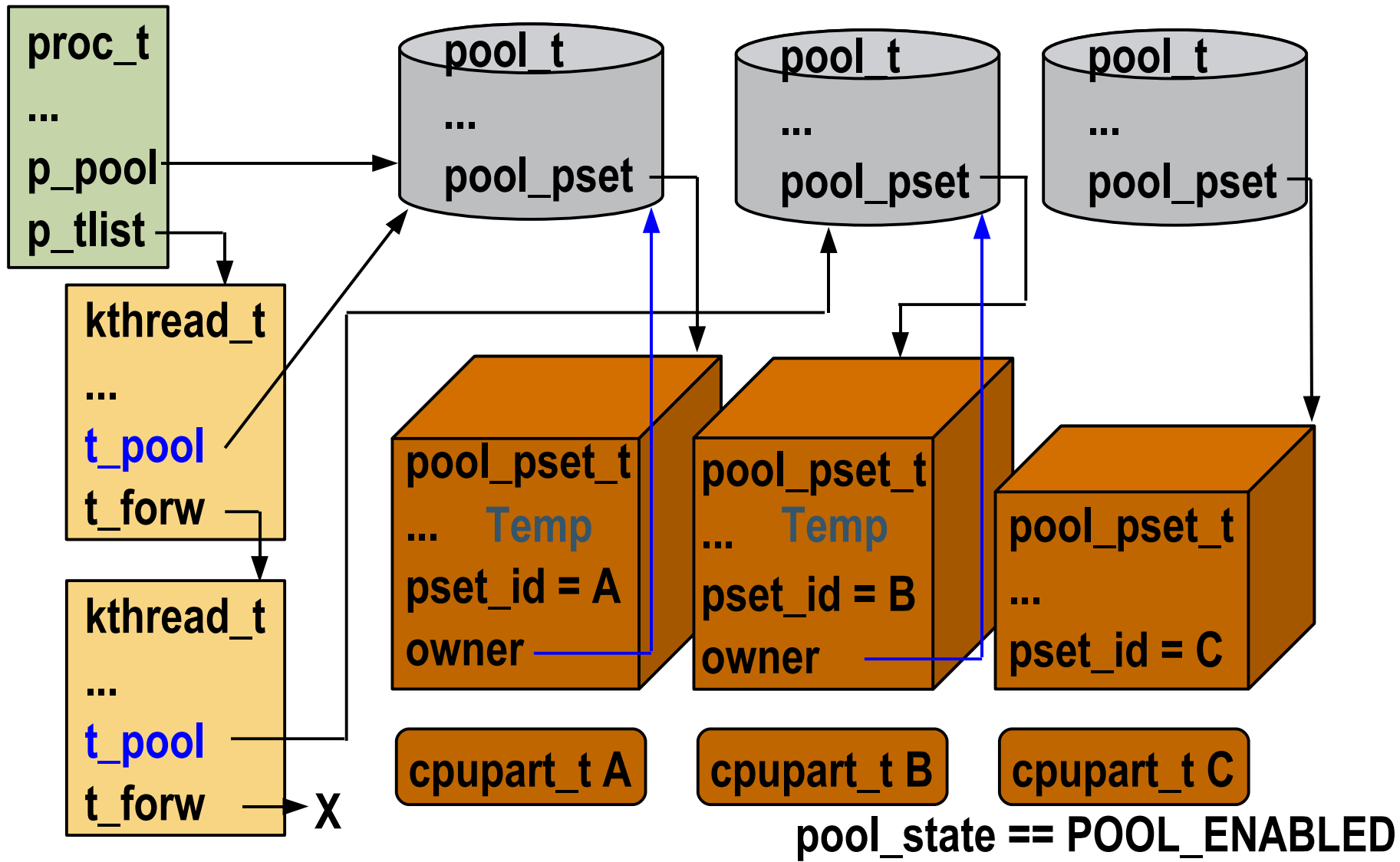
Old Processor Sets



Old Processor Pools



Processor Sets As Pools



Results

- Fully functional, stable implementation
- With pools test suite properly modified for the couple places where semantics changed, pass/fail results match the gate
- Survives arbitrarily many pset stress test runs
- Needs some artifact cleanup from the process-at-a-time binding and disabling, could use some code review
- Workspace: /net/coupe/builds/cg209009/cpupools/
- Webrev:
<file:///net/coupe/builds/cg209009/cpupools/webrev/>
- Updated pools tests: /net/coupe/builds/cg209009/pooltest

What I Learned

- Pools framework – duplication between kernel and libpool
- Boot code
- Process and threading model
- Syscall layer
- (Sort of) How to read x86 assembly, via mdb
- DTrace
- STC/STF Testing Framework
- If you spend enough time debugging certain types of things, you start recognizing addresses; the address of pidlock on my test machine is 0xfec8cd10

Internship Experience

- Cool people, interesting problems
- Some of the most interesting projects are too large for a summer (live migration of zones?)
- I sometimes wanted a couple projects I passed (coredumps earlier in boot, mdb generating process core from a kernel dump)
- Many thanks, especially to Steve Lawrence for project guidance
- <http://blogs.sun.com/csg/>

And now that I've said goodbye...

- I'm here for another two weeks.