Catching up With Herbert

Jamal Hadi Salim
Netconf 2011, Toronto, On, Ca

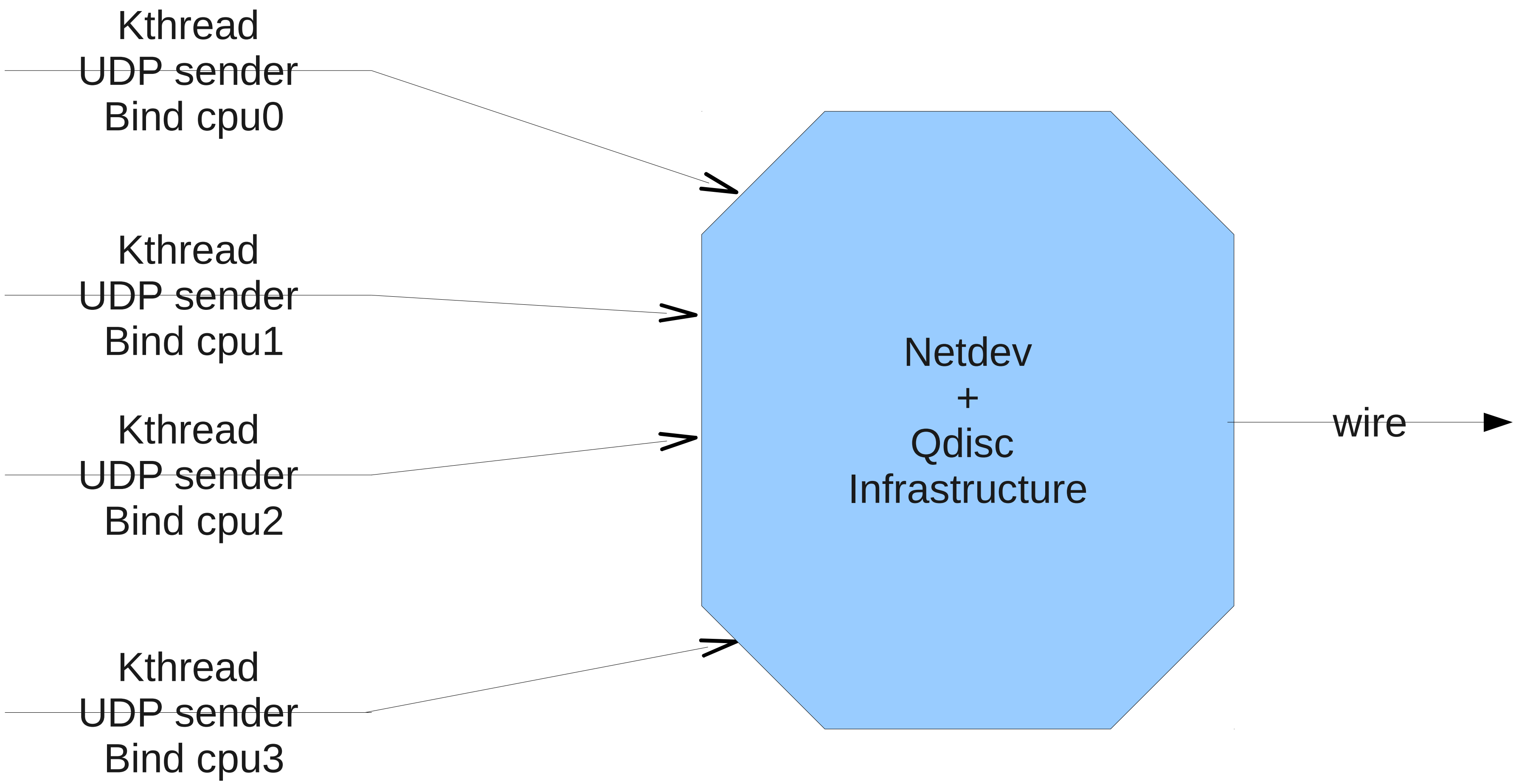
Some History

- Alexey's original scheme with softnet
- Herbert's changes with GRO
- Jamal's decoupling of TX Lock
- Herbert's jiffy/rescheduling changes
- Eric's busylock changes

Challenges Reproducing Theory

- In 2006, did not have pre-requisites
  - Fast enough link to dump packets to
    - I had 2 1xGbps ports
      - 10G is getting commoditized, 40G coming
  - Fast enough and sufficient amount of CPUs
    - I had an “ok” 2 cpu machine
      - 4 to 64 cpus common today

Experiment Setup



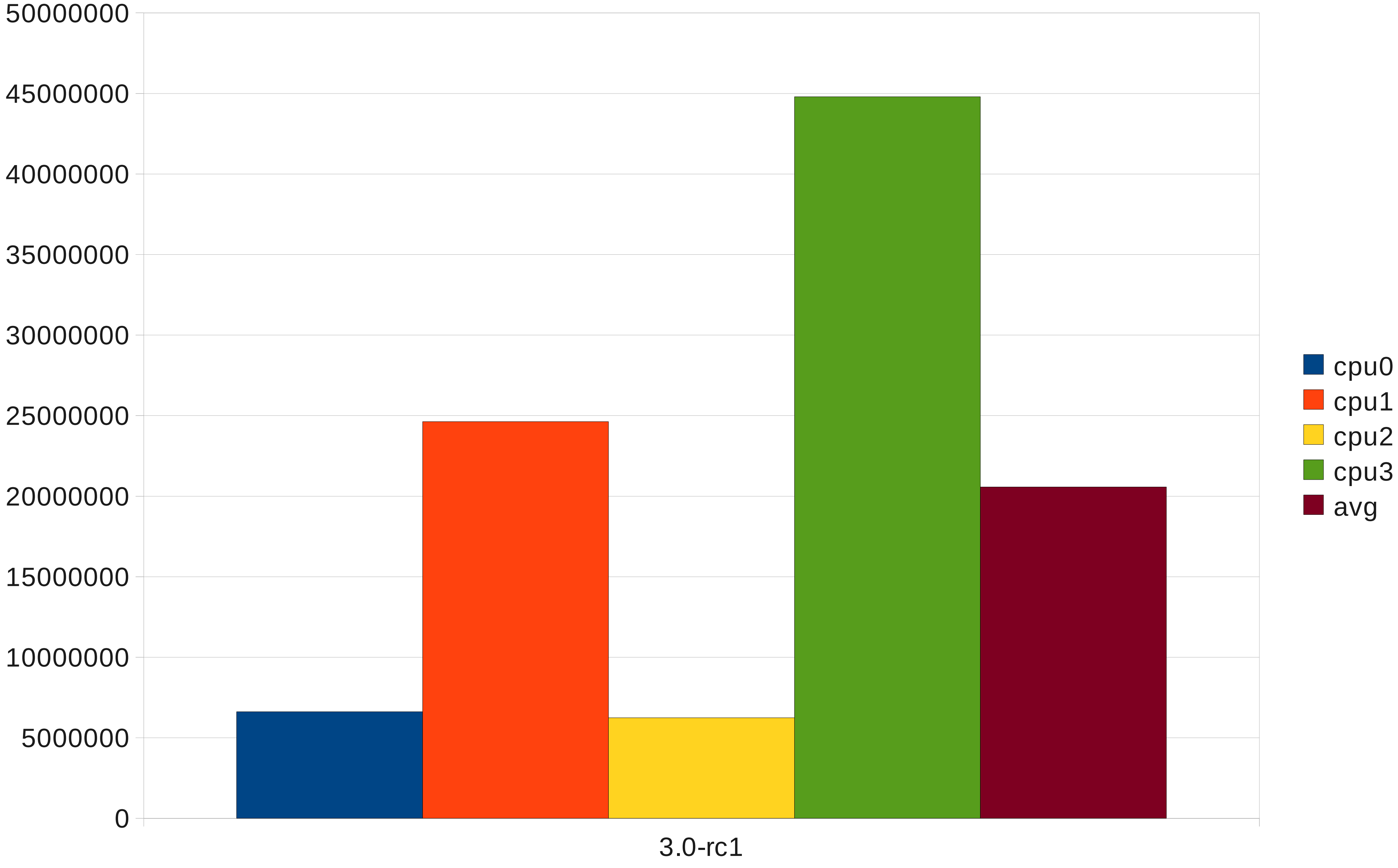
Experiment Setup

- A 4-cpu Intel *i5* Machine *2.27 Ghz*
- Dummy device
  - Infinite bandwidth
- Generate UDP traffic as fast as possible from each CPU, concurrently for 30s or more
  - Designed to overwhelm the qdisc enqueue/dequeue subsytem
- Collect how long each CPU sits in the dequeue region

Experiment Calibration

- One thread
  - 1.24 Mpps, no drops
- Two Threads
  - 2.03 Mpps, 2% drops
- Three Threads
  - 1.59 Mpps, 15% drop
- Four Threads
  - 1.32 Mpps, 40% drop
  - Lets go for this

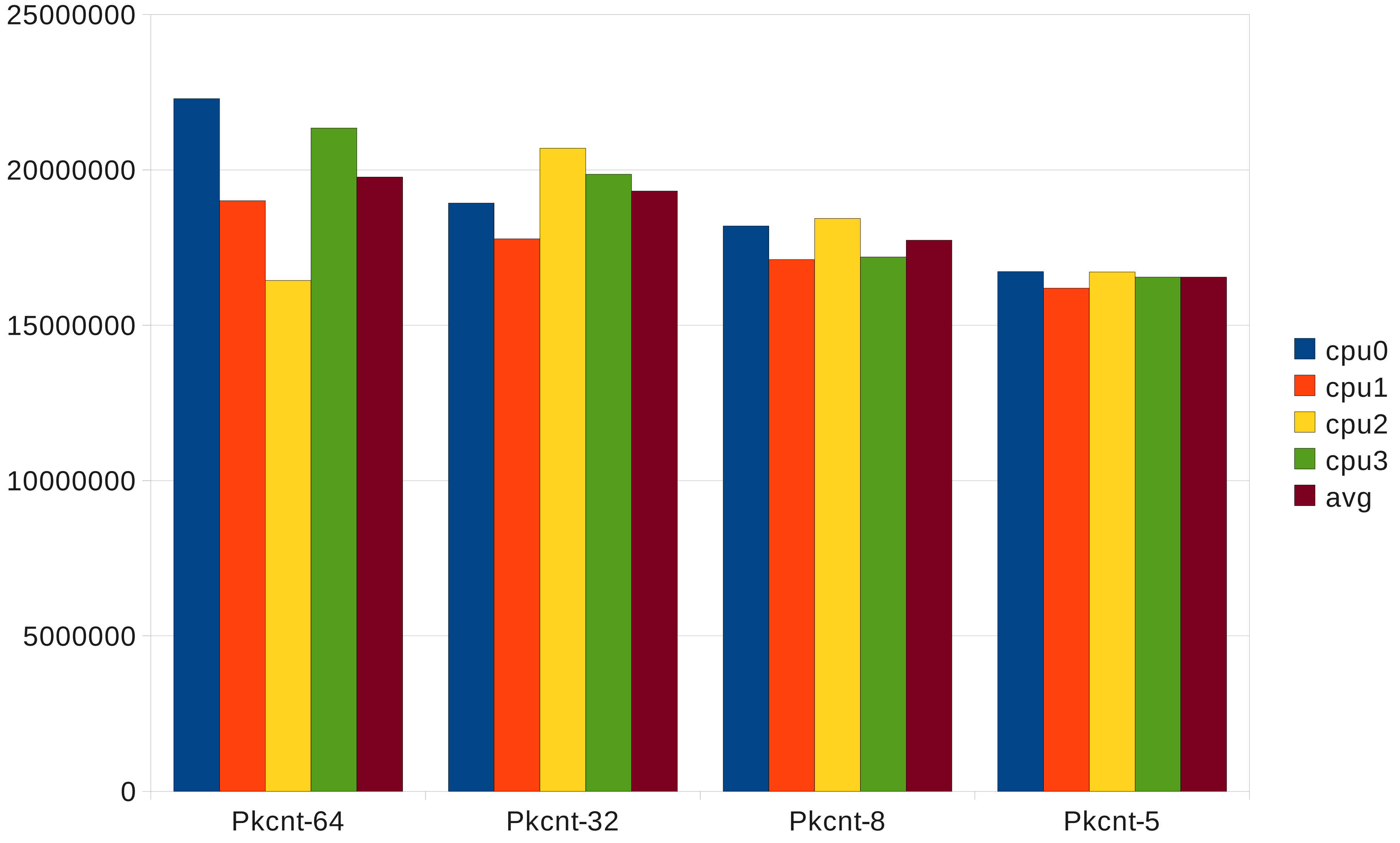
Kernel 3.0-rc1 Dequeue Distribution



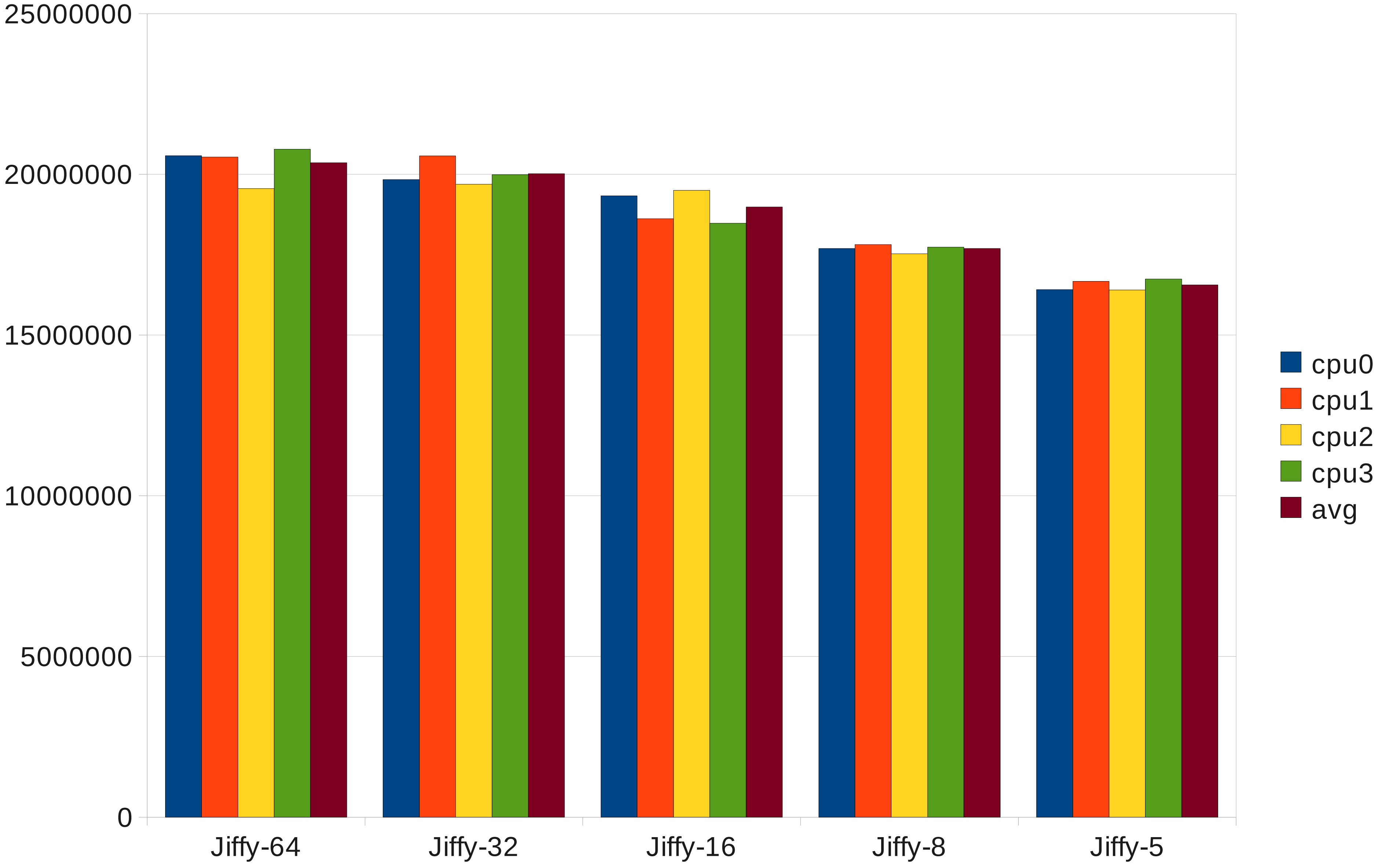
Observations on *3.0-rc1*

- Jiffy is dependent on Hz and clock sources
- Yielding is a factor of how many processes asking for the cpu
- Introduce a packet quota
  - Equivalent to the NAPI poll weight
  - Less subjective to system load

Dequeue Distribution Packet Quota



Add Packet Quota To Existing Scheme



Packet Quota Observations

- Worked better when I had all 3 variables together
  - Better distribution across variety of weights
- A packet quota of N+1 to 2N seemed the most effective
  - However, even at large quotas, there was a huge fairness improvement over status quo

Conclusion

- Things have improved greatly since the first GRO patches
  - Batching no longer buys much
  - Small change to improve fairness needed

Discussions

The dumb Drop at Qdisc

- Old problem
  - ENOBUFS return code to sendmsg/to
  - We yield and get another ENOBUFS
    - We see worst case between 40-60% drops depending on processor capacity
- Possible solution
  - The qdisc code already knows when space becomes available
  - The caller could register for async notification when space becomes available
    - Playing around with a couple possible approaches

Revisiting Busylock

- An improvement, but locks are bad for you
  - *The Cache-pingpong Express Train*
  - Recent studies have shown cache hits could be nastier than local memory trips
  - Own analysis looking at various cache coherency approaches
    - cache traffic increases exponentially with number of contending cpus
    - Memory trips increase only linearly