Implementations of Timing Wheels

Julian Squires

AdGear Technologies, Inc.

March 16, 2017

Disclaimers

- Performance claims without benchmarks are damned lies;
- I don't understand all these systems;
- I'm handwaving on many complex issues, especially concurrency and multiprocessor issues;
- 20 minutes is not enough time;
- Time is the enemy.

Varghese and Lauck. "Hashed and hierarchical timing wheels: Data structures for the efficient implementation of a timer facility." ACM SIGOPS Operating Systems Review 21.5 (1987): 25–38.

Varghese and Lauck. "Hashed and hierarchical timing wheels: efficient data structures for implementing a timer facility." IEEE/ACM transactions on networking 5.6 (1997): 824–834.

Varghese. Network Algorithmics: An Interdisciplinary Approach To Designing Fast Networked Devices. Morgan Kaufmann, 2004.

Recent activity

- LWN on Gleixner's timing wheels patch: https://lwn.net/Articles/646950/
- Adrian Colyer's Morning Paper blog: https://blog.acolyer.org/2015/11/23/hashed-and-hierarchical-timing-wheels/
- Juho Snellman on Ratas: https://www.snellman.net/blog/archive/2016-07-27-ratas-hierarchical-timer-wheel/

Timing facility users

Optimistic

- simulations
- scheduling
- flow control
- circuit breakers
- watchdogs

Pessimistic

- requests awaiting responses
- TCP retransmit timers
- I/O timeouts

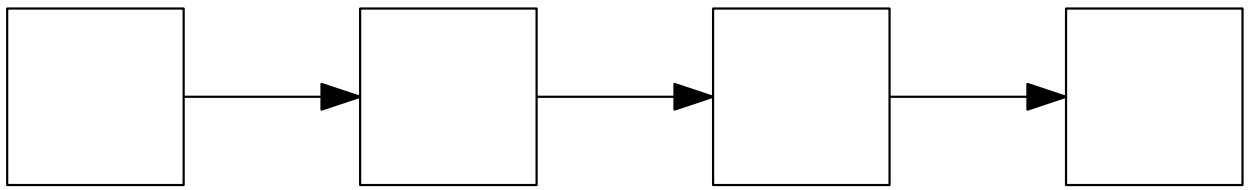
Guarantees of a timer system

A timer scheduled to fire after t ticks will have its action executed, some time after t ticks (if your clock is monotonic and doesn't jump forward or skew forward and . . .)

Design considerations

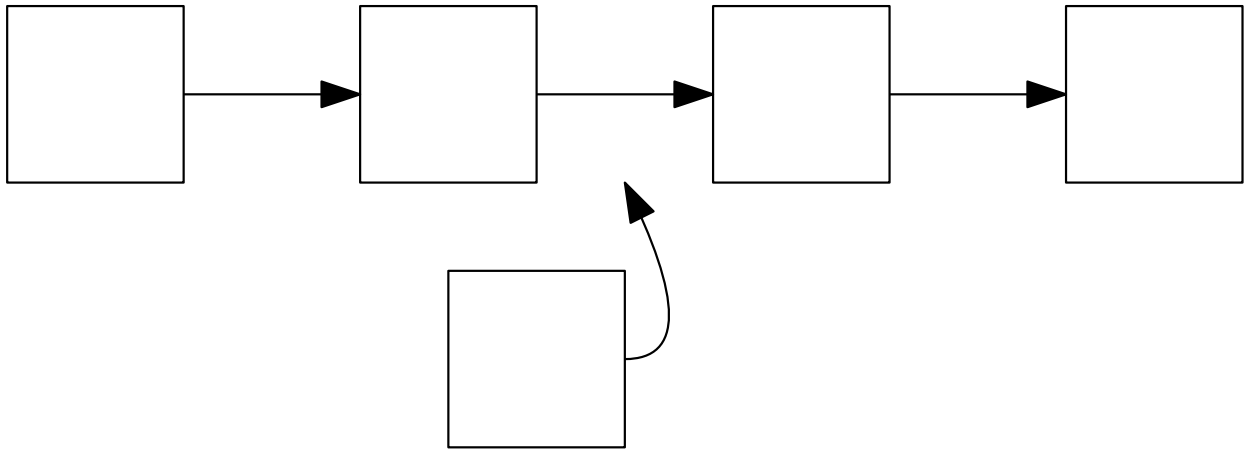
- optimistic versus pessimistic
- accuracy versus performance
- ranged scheduling versus exact
- periodic versus one-shot
- throughput versus latency
- bounded work per timer interrupt
  - timer stampede

Unordered lists



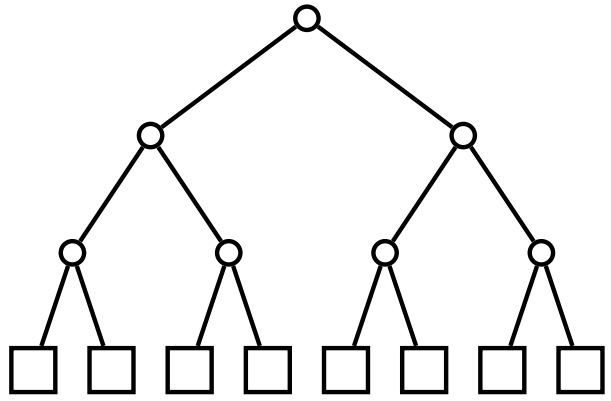
- Vixie cron

Ordered lists



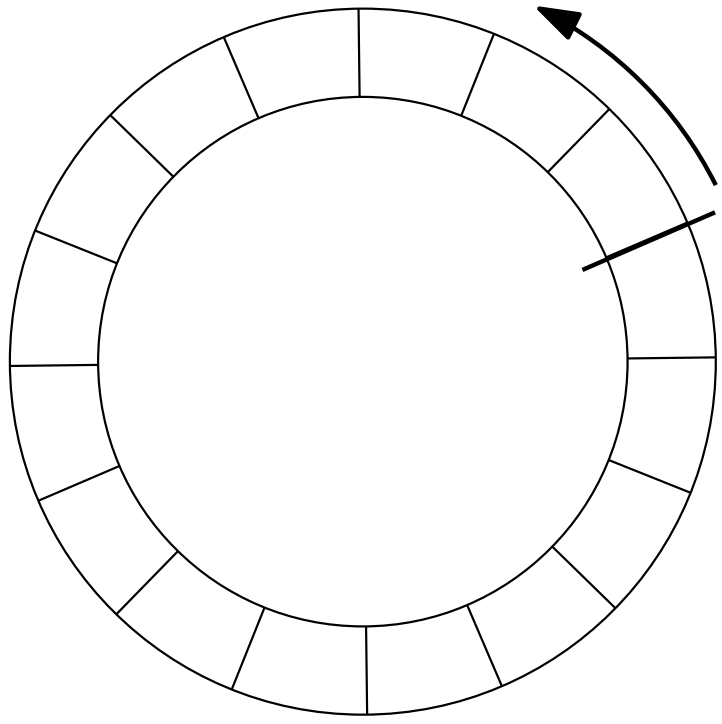
- Zephyr (http://zephyrproject.org/)
  - `kernel/include/timeout_q.h`, `kernel/timer.c`
- XNU (Darwin/macOS) (http://opensource.apple.com/)
  - `osfmk/kern/call_entry.h`, `osfmk/kern/thread_call.c`, `osfmk/kern/timer_call.c`
- pre-1997 Linux, *BSD

Binary heaps

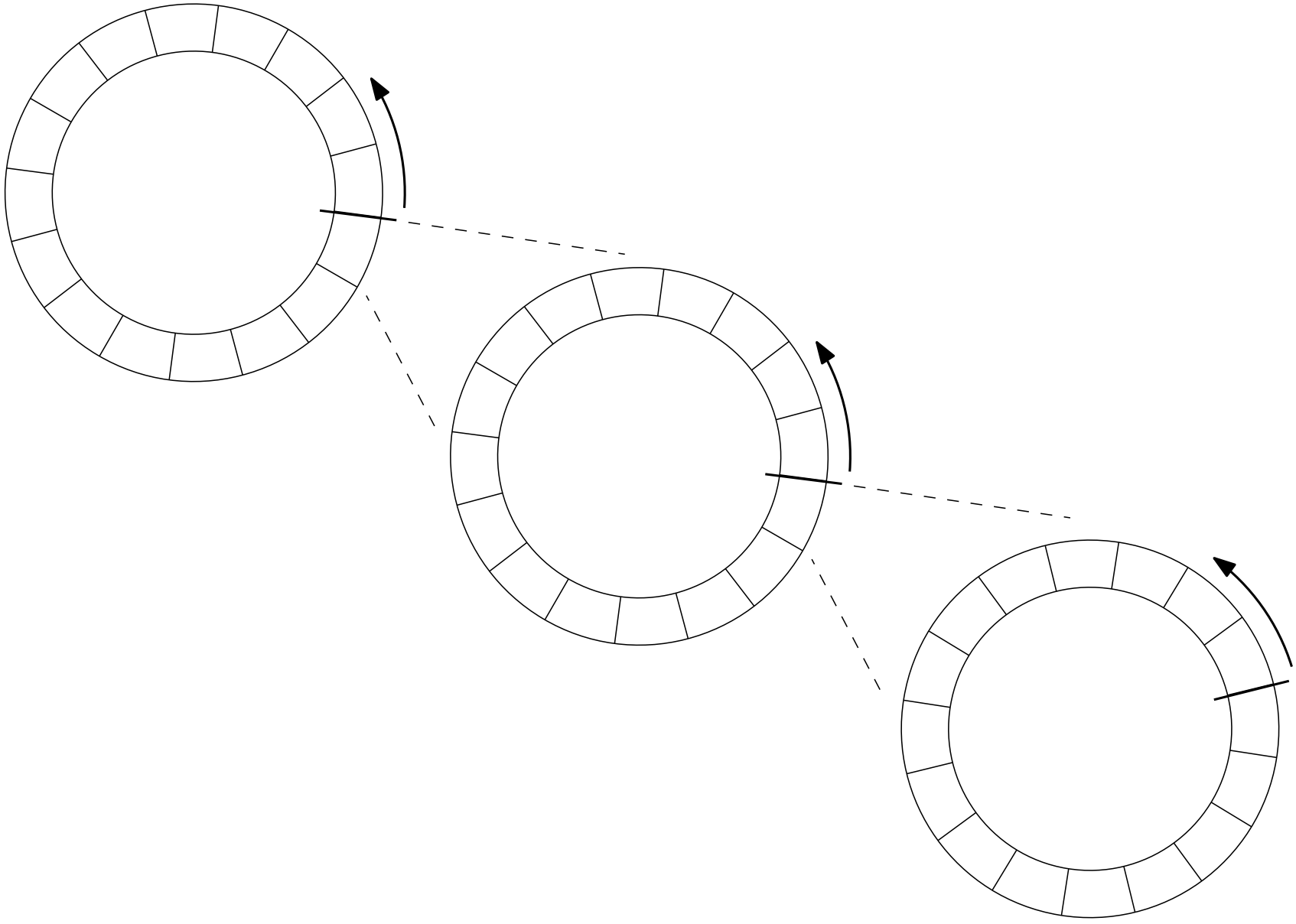


- libev (also a 4-heap!)
- libuv
  - `src/heap_inl.h, src/unix/timer.c`
- OpenJDK DelayQueue
  - `src/java.base/share/classes /java/util/concurrent/DelayQueue.java`
- Illumos
  - `usr/src/uts/common/os/cyclic.c`
  - `usr/src/uts/common/os/callout.c`

Timing wheels



Hierarchical timing wheels



Hierarchical timing wheels

- Linux
  - `linux/kernel/time/timer.c`
- Ratas (https://github.com/jsnell/ratas)
- Kafka
  - `core/src/main/scala/kafka/utils/timer/TimingWheel.scala`
- timeout (https://github.com/wahern/timeout)

Hashed timing wheels

- *BSD
  - `sys/kern/kern_timeout.c`
- Erlang (port and proc timers)
  - `erts/emulator/beam/time.c`
- Netty
  - `common/src/main/java/io/netty/util/HashedWheelTimer.java`

Costello and Varghese. "Redesigning the BSD callout and timer facilities." (1995).

Italiano and Motin. "Calloutng: a new infrastructure for timer facilities in the FreeBSD kernel." (2013).

Other techniques

- calendar queues
- skip lists
  - DPDK (http://dpdk.org/) (`lib/librte_timer/rte_timer.c`)
- red-black trees
  - Linux (`kernel/time/hrtimer.c`)
  - Erlang (`erts/emulator/beam/erl_hl_timer.c`)
- hash table
  - node.js (`lib/timers.js`)
- softheaps?

$O(lgN)$ versus $O(1)$

Jason Evans says:

> *In essence, my initial failure was to disregard the difference between a O(1) algorithm and a O(lg n) algorithm. Intuitively, I think of logarithmic-time algorithms as fast, but constant factors and large n can conspire to make logarthmic time not nearly good enough.*

http://t-t-travails.blogspot.ca/2008/07/overzealous-use-of-my-red-black-tree.html

Julian Squires
<julian@cipht.net>