

O'REILLY

OSCON™
Open Source Convention



Optimizing APC

Gopal Vijayaraghavan

Zynga India

gopalv@php.net

July 29th, 2011

So, what's this talk about ?



A Retrospective



What Changed?



Virtualization and cpu-0



Cache Surplus



Devil is in the Details



Allocator fixes

- `apc_sma_free()` was taking too much time
- This is now a doubly linked list structure
- This makes `allocate()` slower
- But the `deallocate` is our bottleneck
- Entire user cache locks up during deallocs
- This fix lets us run caches so close to 100%
- Basically, worry about cache flushes no more
- Fragmentation is far less of a headache



Tweak Pool Allocation

- APC-3.1.x branch saw a new pool allocator
- Pools were either small, medium or large
- Pools did away with book-keeping pointers
- No more memory leaks from forgotten free()s
- No locks unless the pool has filled up
- Look back only 8 items, not the whole pool
- Increase pool block size (by size allocated so far)
- This does speed up everything to do with memory



Pluggable Serializers

- Igbinary can be used instead of php serializer
- Igbinary is a far more compact format
- Reduces memory churn for large arrays
- Is faster to unserialize than php serializer
- `apc_register_serializer()` api
 - Write your own serializer in another ext
 - Pure runtime hooks, no need to link to `apc.so`



Read/Write Locks

- RW locks
- Refcount++ does not need a lock
 - Needs atomic inc/dec from gcc to use
- Slows down single-threaded execution
- Speeds up parallel execution
- Pthread lets you prefer writers or readers
 - Uses writer preferred by default
- So it speeds up `apc_store()` as well



Upgrade Locks (FAIL)

- Attempt to create URW locks
- Upgrade() turns a read lock into a write lock
- Works fine
- Makes code slower than mutexes
- During a cache slam, all of them turn into writers
- Was more complicated to debug



Slam Detection for apc_store()

- Throw out duplicate inserts before locks
- What do we throw out?
 - Any parallel write with the same hash
 - One second granularity
- Fixes heavy writes on cache-clear
 - Reduced locking
 - Stabilizes faster
- Reduces fragmentation in memory

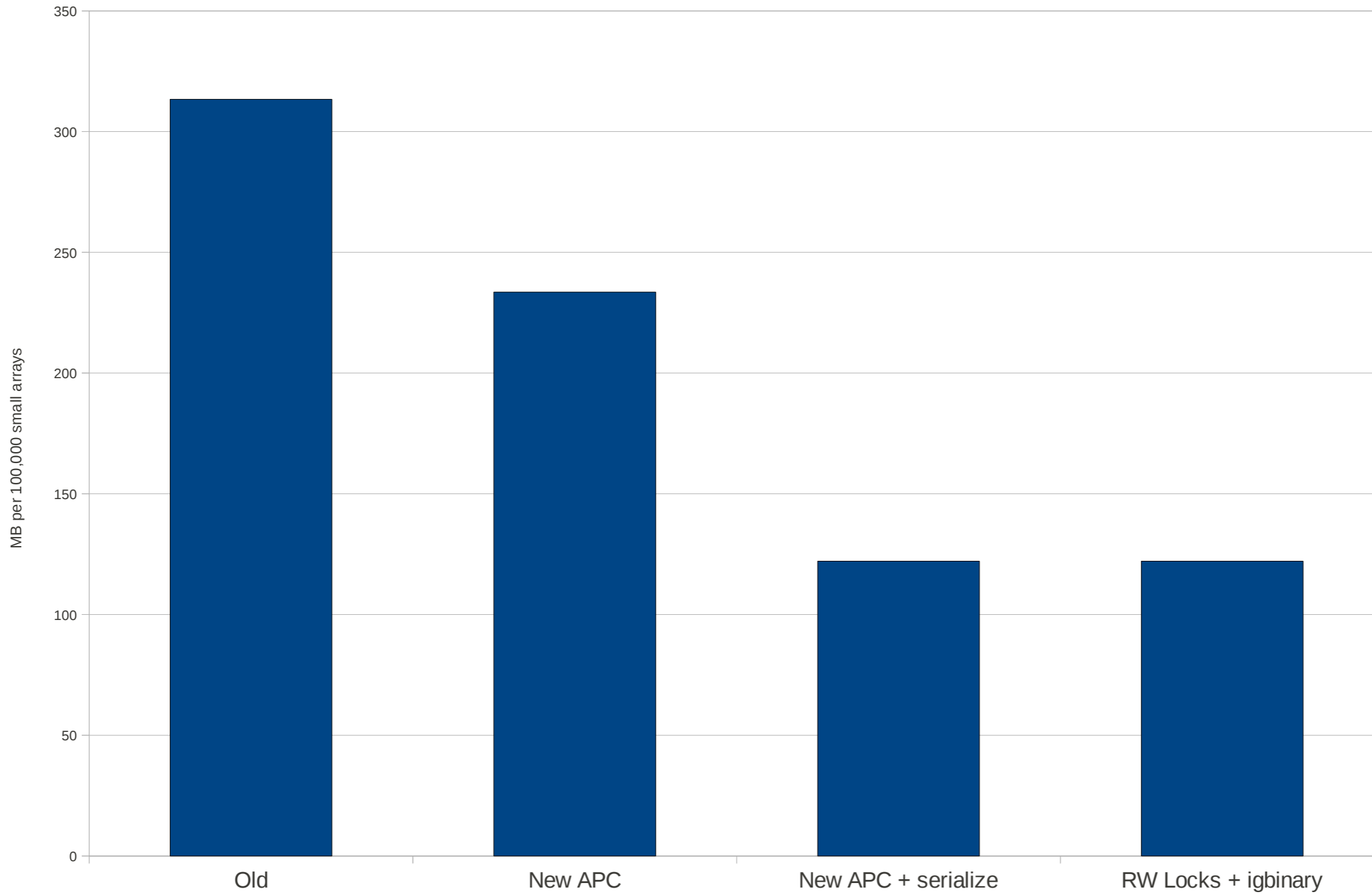


Benchmarks

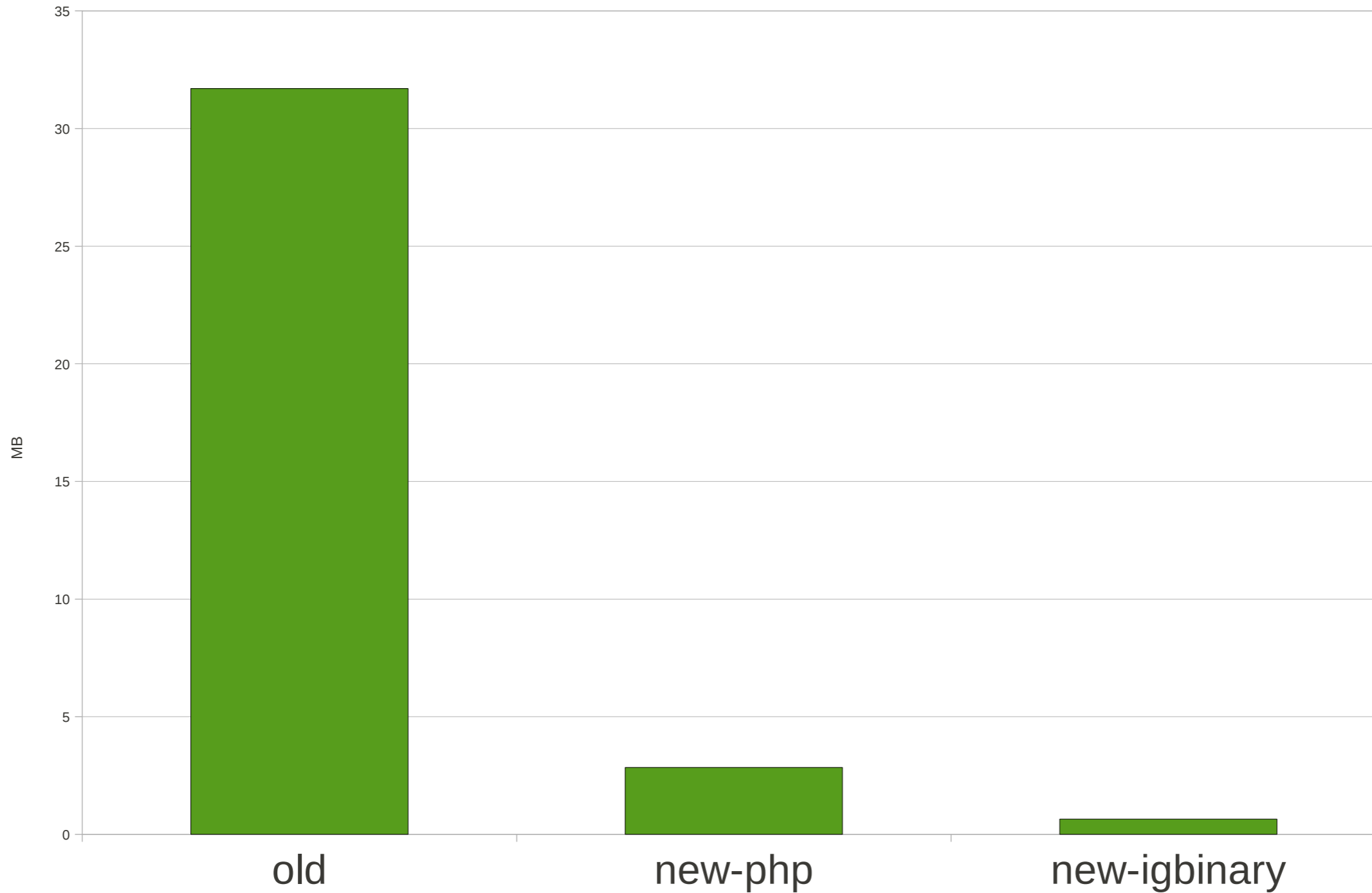
- What do we expect to change
 - Straight line speed
 - read/write
 - Concurrency
 - read/write
 - Memory efficiency
- <http://notmysock.org/code/apc-torture.tgz>



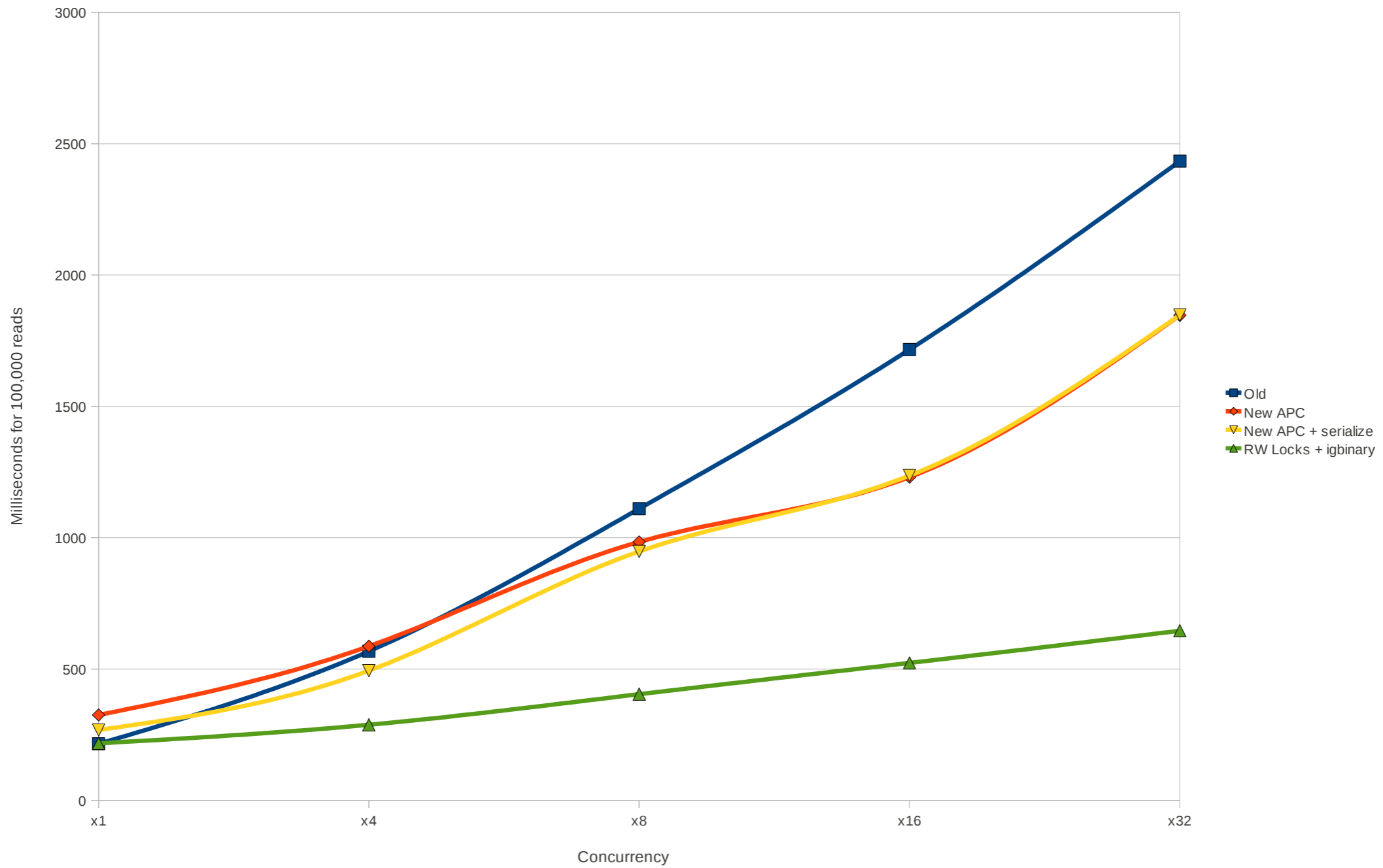
APC memory sizes



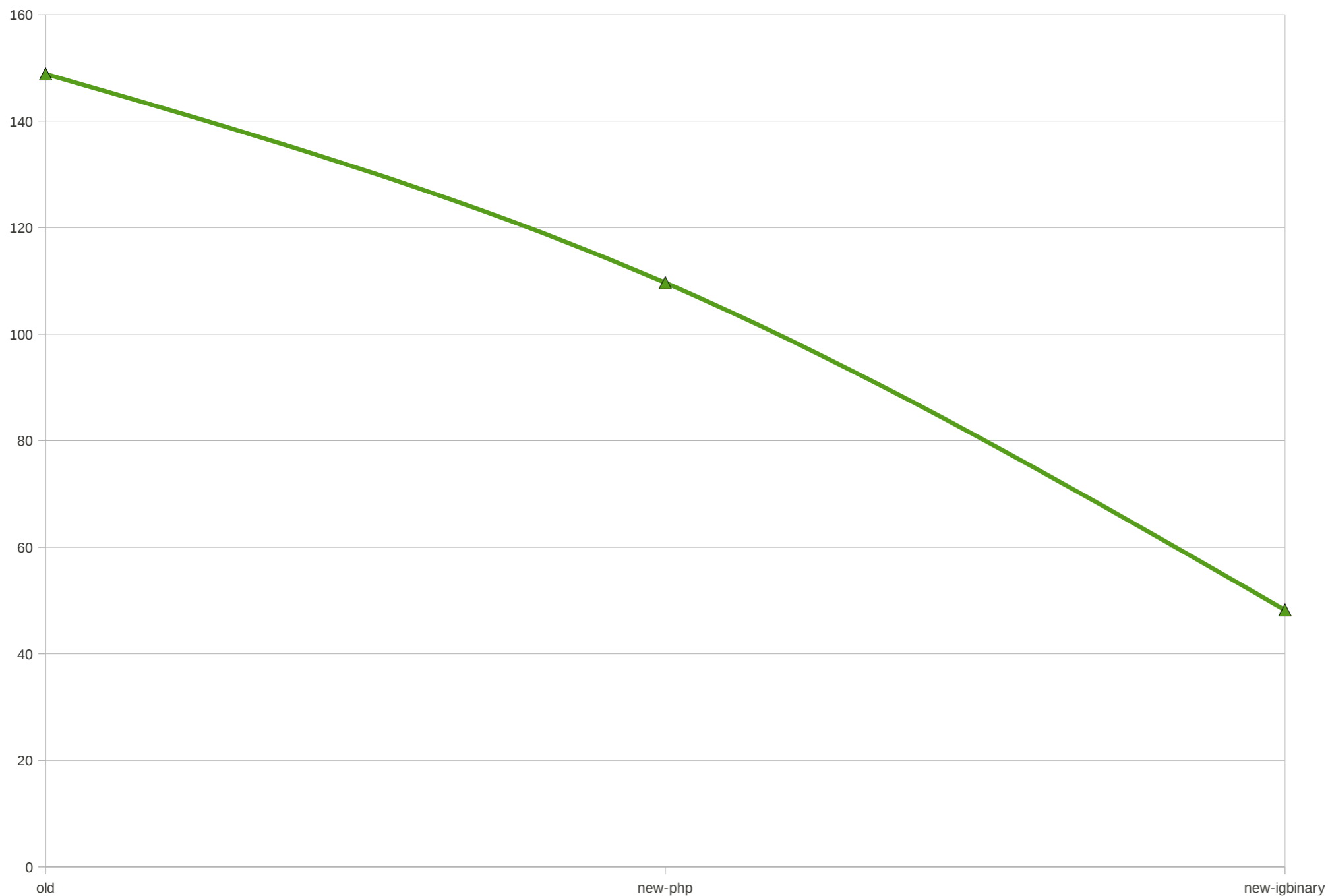
Size per Item



Read Concurrency (test3)



Time to Read



pecl/hidef hits stable

- APC isn't the only horse in this stable
- For read only data – hidef is preferred
- Does not hit the php memory_limit as hard
- Zero locking
- Hidef's worst-case performance matches APC's best-case performance
- Will not have data corruption due to a crashing process



Any questions ?

- Resources
- <http://pecl.php.net/package/APC>
- <http://pecl.php.net/package/hidef>
- And thank you for listening !



Thanks to Zynga

- For supporting my work on PHP & APC
- Giving me time off to come talk here
- And Zynga's hiring php programmers!
- <http://www.zynga.com/jobs>

