Magazines and Vmem :
Extending the Slab Allocator to Many CPUs and Arbitrary Resources

Jeff Bonwick, Jonathan Adams

USENIX Annual Technical Conference, General Track. 2001

Wonsuk Song

2017.5.1

Intro

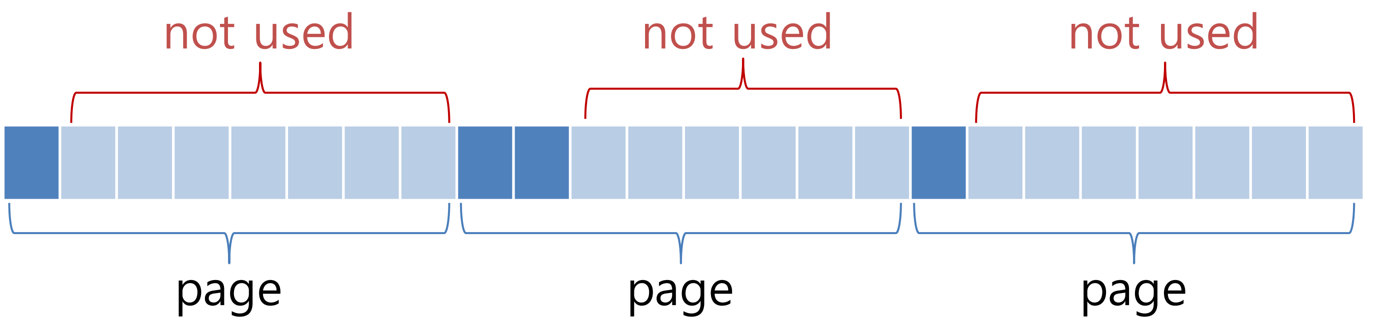
- Jeff Bonwick

- Inventor of slab allocator
- Invented Slab at 1994
- Deployed in Solaris 2.4 (SunOS 5.4)

- Magazine & Vmem
  - To cover limitations of slab allocator
  - 1st : What is slab & it's limitation?
  - 2nd : Magazine
  - 3rd : Vmem

Why Slab Allocator?

- Kernel manages it's memory with pages(4kb)
- What if we kmalloc 64bytes? -> alloc 1 page from buddy allocator?

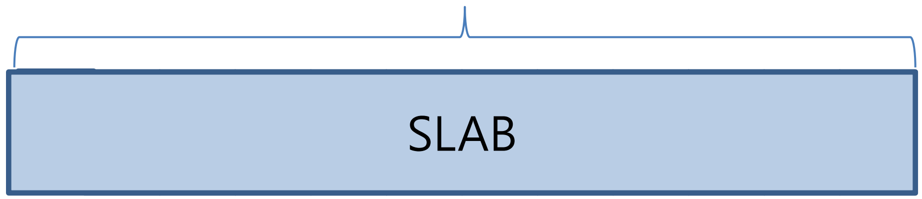


- To avoid this problem, kernel already has a group of objects of similar type

  →**SLAB allocator**

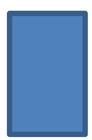
Why Slab Allocator?

Usually 1~2 contiguous page



Objects (32Byte)

kmalloc(19bytes)



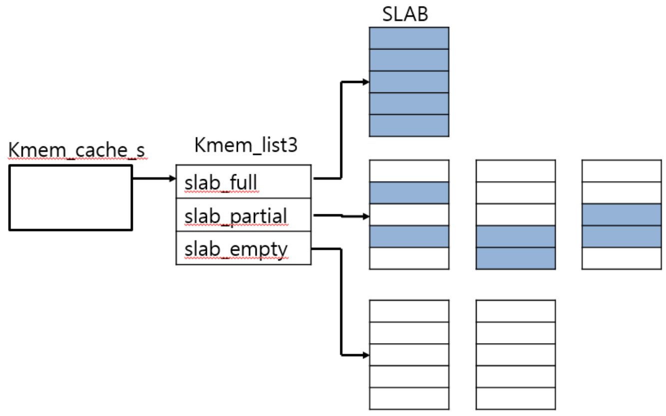
Slab allocates memory size with closest $2^n bytes$

Slab Allocator

- Object Cache
  - Maintains distinct freelists of the most commonly requested buffer size
- Slab Allocator
  - Preserve the invariant portion of an object's initial state(*constructed state*) between use
  - Not have to be destroyed and recreated every time, when the object is used

Slab Allocator

- 94’ Sun Microsystems Solaris implemented



Slab: One or more pages of virtually contiguous memory

Object: Prepared space for frequently objects (*ex. Struct task, file, buffer_head, inode etc.*)

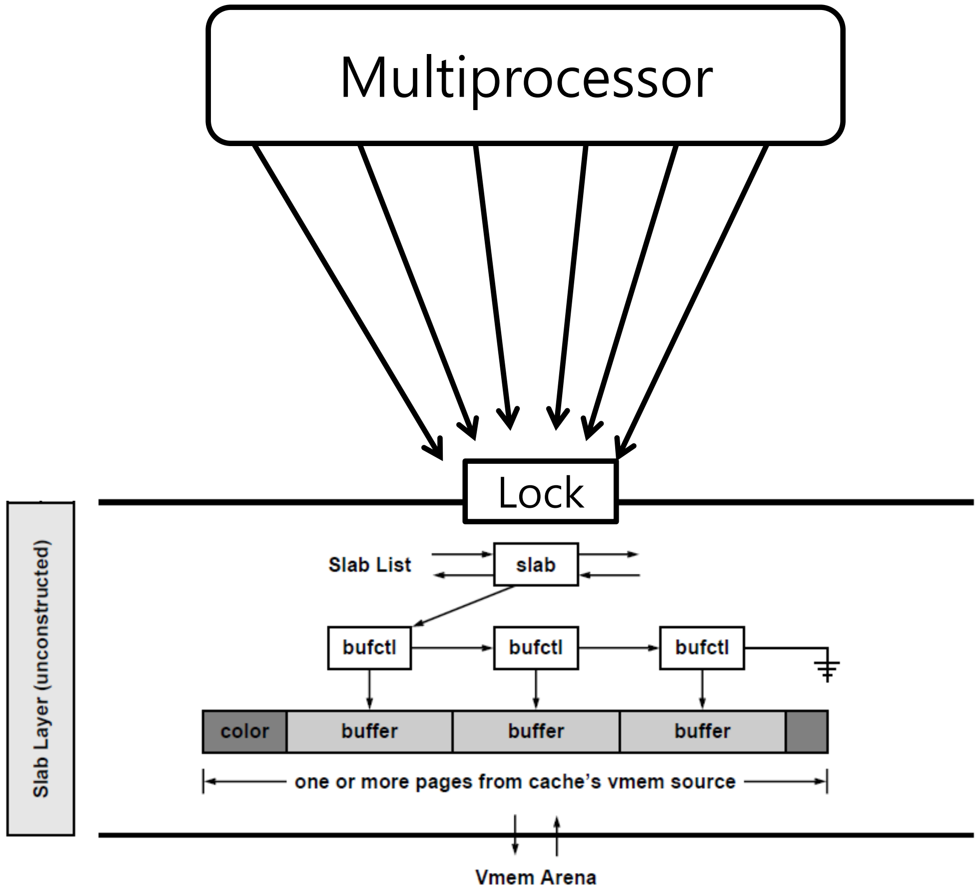
<Slab structure of linux>

Frequently used data structures tend to be allocated and freed often, so cache them

cost of constructing an object >> cost of allocating memory for it

→ free without destroying and reinitializing

Slab Allocator(limitations)



Slab Allocator(limitations)

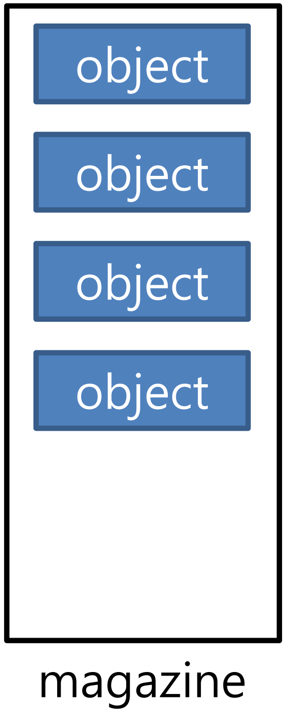
- Lacks multiprocessor scalability
  - Lock(to protect the cache's slab list) makes allocations serialize

- Slab allocator can't manage resources other than kernel memory

Need to allow all CPUs to allocate in parallel (per-CPU caching)
→ give each CPU an M-element cache of objects (magazine)

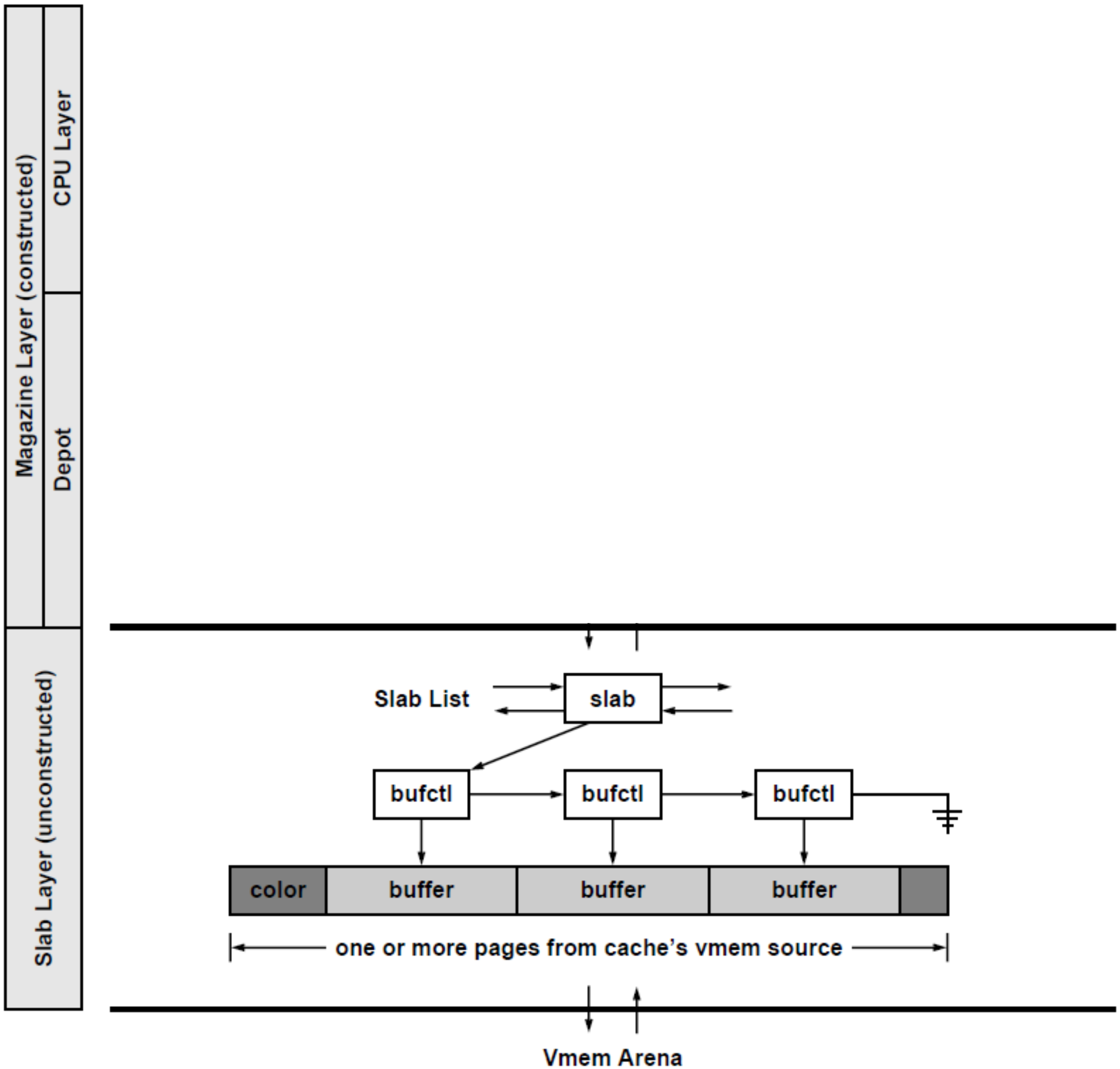
Magazine

- Part of automatic gun that contains bullets
- Each CPUs own it's own magazine



Magazine

- M-element array of pointers to objects(work like stack)
  - Pop to allocate top element
    - obj = magazine[--rounds]; //round : valid pointers
  - Push to free an object
    - Magazine[round++] = obj;



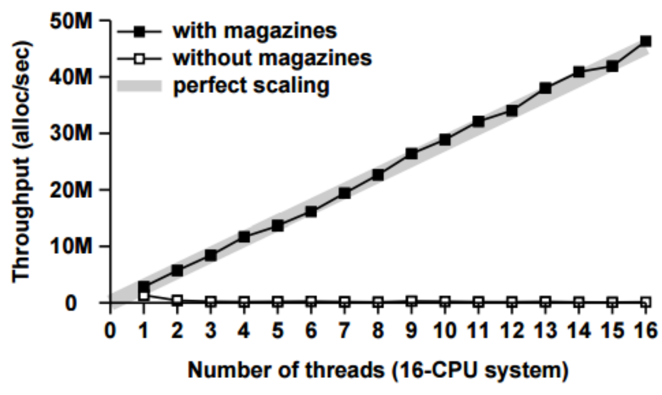
To prevent frequent trade, there is previous magazine per-CPU cache

Magazine (cont.)

- Magazine Size (M)
  - Observe CPU layer's miss rate as low as by increasing M (Initial value)
  - Observe the contention rate on the depot lock (Increment a contention count)
  - If contention rate exceeds fixed threshold, increase the magazine size
- Depot Size
  - If depot's full magazine list varies between 37~47 over a given period, then working set is 10 magazines (Remainder are eligible for reclaiming)

Magazine Performance

- Microbenchmark Performance
  - Measured latency as the average time per iteration of tight alloc/free loop
  - Measured scability by running multiple instances of the latency test on 333MHz 16-CPU Starfire
  - Result
    - Single-CPU performance improved (743ns → 356ns)
    - Linear performance improvement as the thread increases
    - Without magazine throughput decreases with additional threads (because of lock contention)

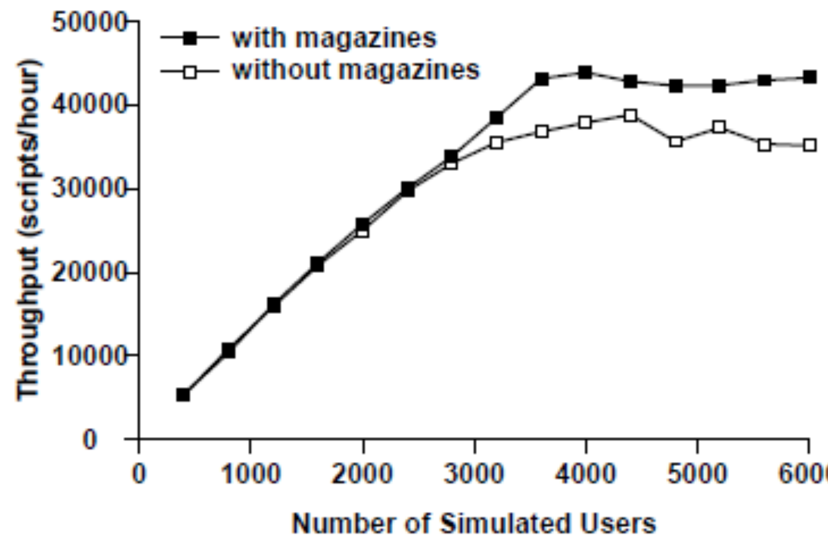


<Allocation Scalability>

Magazine Performance (cont.)

- With System-Level Benchmarks
  - Faster with magazines
  - Performance improved a lot in allocator-intensive workloads(network I/O)

- System-Level Benchmark
  - SPECweb99 (8-CPU E6000)
  - TPC-C (8-CPU E6000)
  - **Kenbus (24-CPU E6000)**



Vmem

- Slab allocator relies on two lower-level system
  - Virtual address allocator : provide kernel virtual addresses
  - VM routines : back addresses with physical pages & V to P translation
  - Linux uses *rmalloc() & rmfree()*
  - *Problem : linear time performance depending on fragmentation*

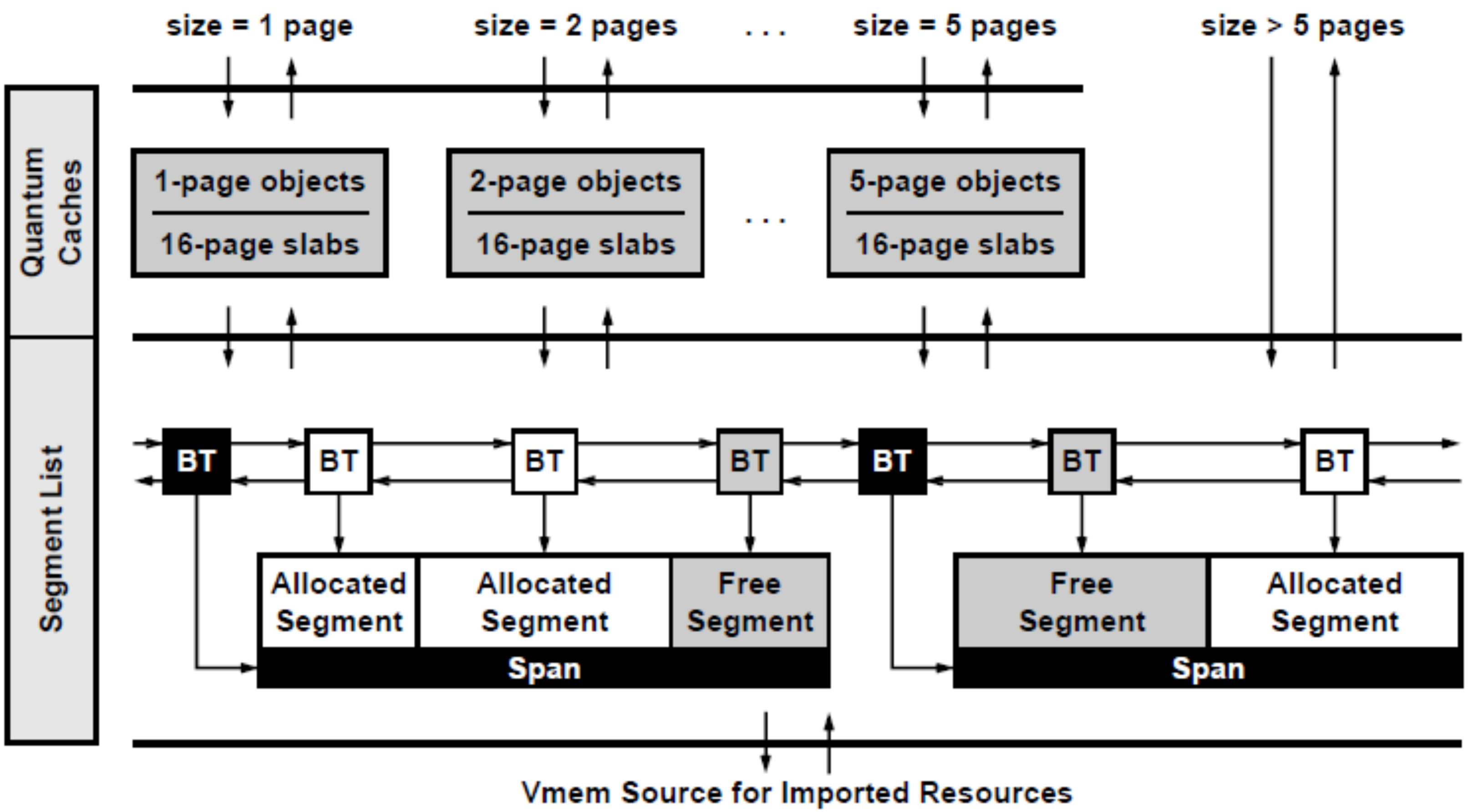
- Vmem : general purpose resource allocator
  - Guarantee constant-time performance → O(1)
  - Interface that can express the most common resource allocation problems
  - Linear scalability to any number of CPUs
  - Low fragmentation

Vmem

- Resource → set of integers

- *ARENA : simple set of integers*
    - *virtual address*
    - *Device number*
    - *PIDs*
    - *etc*

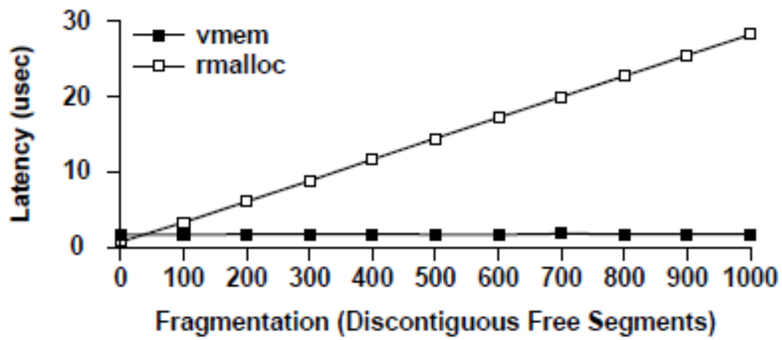
- Vmem is composed of three basic things
  - Create & Destroy arenas
  - Alloc & free resources
  - Import new resources(from another arena) dynamically



1 page quantum and 5 page qcache_max

Vmem Performance

- Microbenchmark Performance
  - Vmem vs rmalloc
    - Zero fragmentation : Vmem(1560ns), rmalloc(715ns)
    - Vmem is faster than rmalloc even at low fragmentation.



- System-Level Benchmark
  - LADDIS : 50% improvement in peak throughput
  - Web Service : reduce system time from 60% to 10%
  - I/O Bandwidth : performance improved by several orders of magnitude

Core Slab Allocator Enhancement

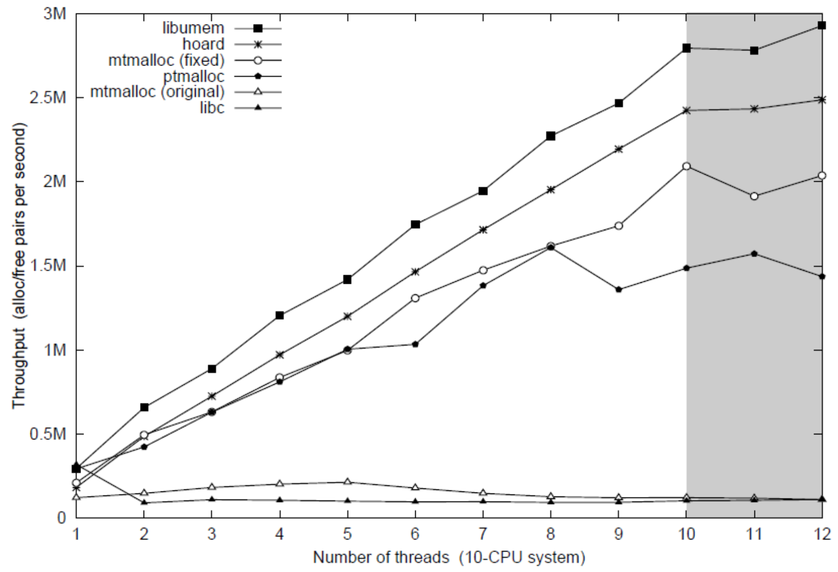
- To support Magazine and Vmem layer, some enhancements were developed to the slab allocator

- Object caching for any resource
  - Provide object caching for any arena
  - Able Vmem's high-performance quantum caching

- Reclaim Callbacks

User-Level Memory Allocation : Libumem Library

- To transplant the Magazine, Slab, and Vmem to user-level
- Libumem is a library that provides all the same service.



<Compare of libumem's performance with other user level allocator>

Summary

- Limitation of Slab
  - Lacks multiprocessor scalability
  - Slab allocator can’t manage resources other than kernel memory
- Magazine
  - Provides efficient object caching with very low latency and linear scaling
- Vmem
  - Guarantee constant-time performance regardless of allocation size or arena fragmentation