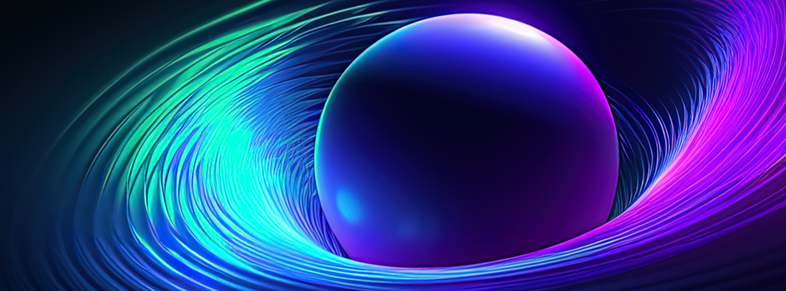
The Challenges of Scalability in Linux Kernel

Muchun Song <songmuchun@bytedance.com>
Qi Zheng <zhengqi.arch@bytedance.com>

Linux Kernel Engineer from ByteDance
Maintainer of HugeTLB and memory cgroup in Linux kernel



2048 GB

Size of memory

384

Number of CPUs

Page Structure

One-to-one mapping between PFN and page structure

The size of page structure is 64 bytes on 64-bit system

3.2%

Double the overhead of page structure initialization

Overhead of virtualization

16 GB

Overhead of page structures on 1TB system

HVO

An approach of minimizing

overhead of struct page for HugeTLB and DAX pages

Folio

Folio is a page structure that is guaranteed not to be a tail page.

More scalable in support of different page size

More clear to take folios as arguments

Saves both time and memory

Reduce bugs

2‰

Overhead of PTE page table

madvise

Free PTE page tables when necessary

Free PTE page tables when necessary

1. Introduce a refcount into page table descriptor (page structure)

2. The walker of page table should grab a reference to PTE table before accessing and release the reference after completion

3. The PTE page table will be freed once its refcount drops to zero

Reclaim PTE page tables when needed

Reclaim PTE page tables when needed

1. Introduce a refcount into page table descriptor (page structure)

2. The walker of page table should grab a reference to PTE table before accessing and release the reference after completion

3. The PTE page table will be freed on the routine of memory reclaiming or syscall of madvise

Interaction between multiprocessors

Software mechanism of inter-CPU synchronization

TLB flushing via IPI

Any solution?

Lock Contention

Even if the lock itself is not contended, the constant cache-line bouncing hurts performance.

Lock Contention

Reduce frequency to acquiring a lock

Release lock as soon as possible

Split the global lock

Lockless

LRU lock

slub node lock

mmap lock

zone lock

shrinker lock

mmap lock

Significant impact is the performance in the routine of page fault

mmap lock

mmap lock

mmap lock

mmap lock

LRU lock

slub node lock

mmap lock

zone lock

shrinker lock

LRU lock

Slow down the performance of memory allocation

LRU lock

per-memcg LRU lock

folio?

LRU lock

slub node lock

mmap lock

zone lock

shrinker lock

zone lock

Slow down the performance of memory allocation from buddy

zone lock

Automatically tune the cache size of PCP

LRU lock

slub node lock

mmap lock

zone lock

shrinker lock

shrinker lock

Slow down the performance of memory reclaiming

shrinker lock

Lockless based on refcount and RCU

LRU lock

slub node lock

mmap lock

zone lock

shrinker lock

slub node lock

Slow down the performance of memory allocation from slub

Any solution?

Thanks