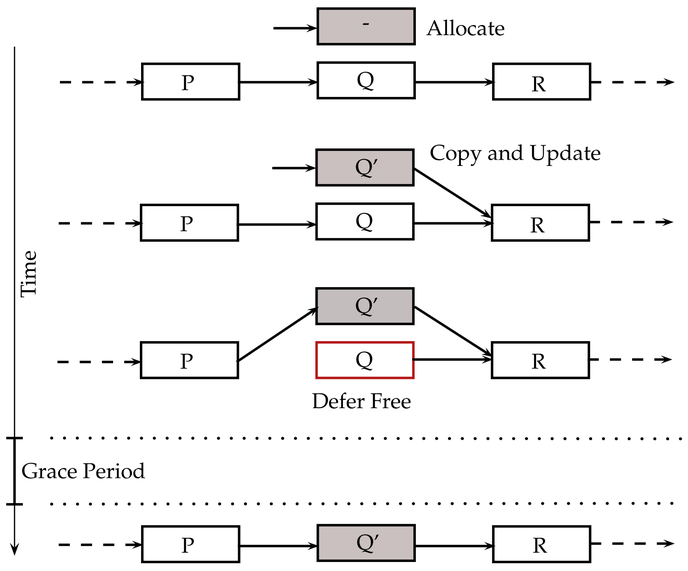
Prudent Memory Reclamation in Procrastination-Based Synchronization

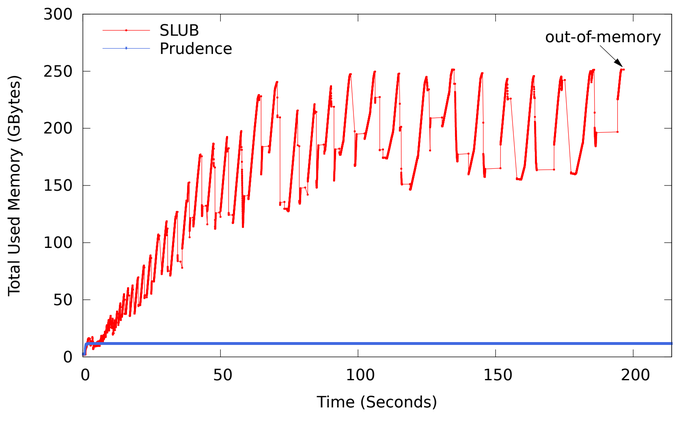
Aravinda Prasad, K Gopinath
{aravinda, gopi}@csa.iisc.ernet.in
Computer Science & Automation (CSA), Indian Institute of Science (IISc), Bangalore

Synchronization via Procrastination

- Readers:
  - Do not synchronize with writers
  - Are wait-free and scale linearly
- Writers:
  - Copy the object and update copied version
  - Wait for pre-existing readers referring the old version to complete



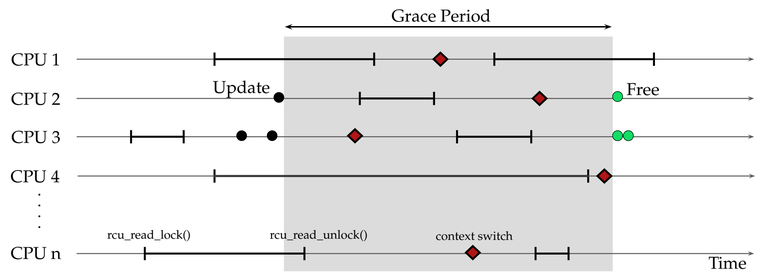
High slab churn rate



Grace Period Computation in the Linux Kernel

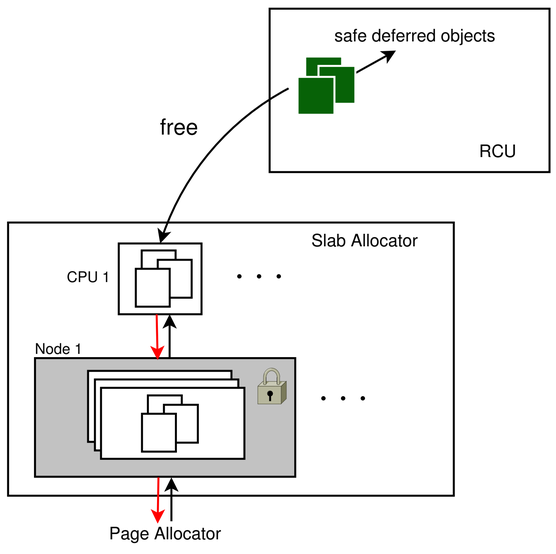
RCU in the Linux kernel restricts readers from:

(i) holding reference to an object outside the read-side critical section
(ii) relinquishing the CPU inside a read-side critical section



Bursty freeing of memory

- Object cache flushing due to overflow
- Slab cache shrinking

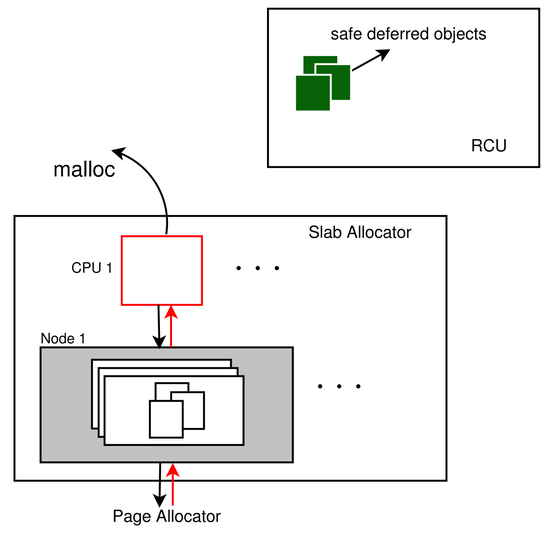


Bursty freeing of memory

Extended object lifetimes

Deferred objects are not freed as soon as they are safe:

- Triggering of interrupts, Preemption
- Throttling the processing of deferred objects to avoid jitters



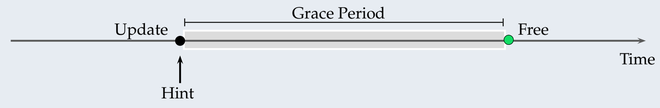
Extended Object Lifetime

Synchronization via Procrastination from Memory Allocator's Point of View

- Frequent allocation and freeing of objects
- Allocation is spread over an interval of time. Freeing occurs in bursts
- Reclamation of safe deferred objects is controlled by synchronization mechanism

Hints about the future

Deferred frees provide "precise hints" about the memory regions that are about to be freed



The Prudence Dynamic Memory Allocator

The basic design principle is to have deferred objects visible and processed in the memory allocator

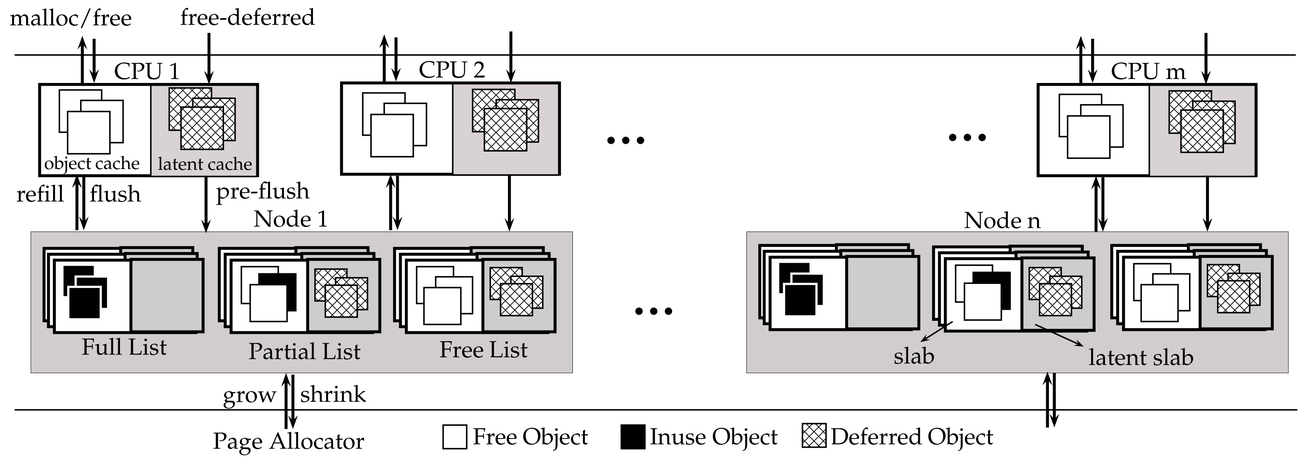
Requirements

- Interface to defer the freeing of an object
- Identify safe time to reclaim deferred objects

Design

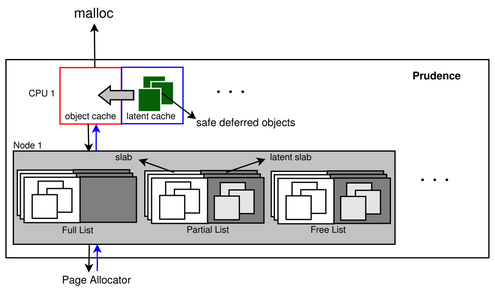
- Export a new API to defer free an object
- Integrate synchronization mechanism with Prudence to identify the safe time to reclaim deferred objects

An individual slab cache in Prudence



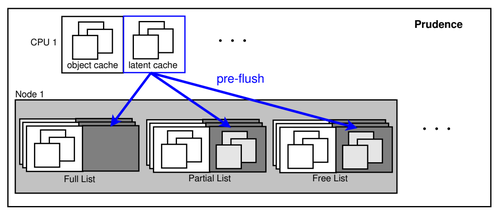
Optimization during object cache refill

Safe deferred objects in latent cache are merged with the object cache



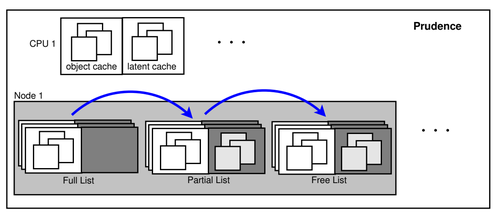
Latent cache pre-flush

Prudence initiates latent cache pre-flush if it foresees an object cache flush

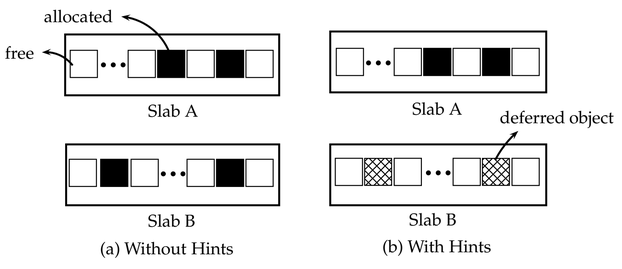


Slab pre-movement

Prudence moves a slab between full, partial and free lists if it foresees such a movement in the future

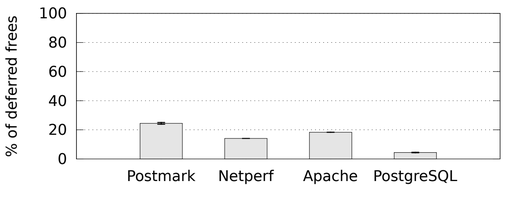


Reducing total fragmentation with hints

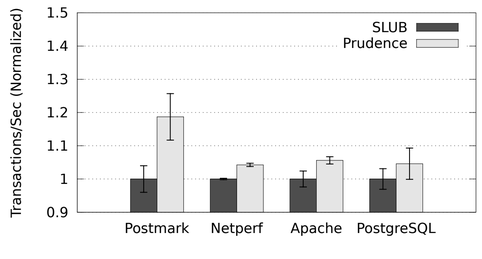


Results - Application/Synthetic Benchmarks (Intel Xeon processor, 64 CPUs (4 sockets, 8 cores/socket, 2-way HT), 252 GB physical memory, Linux 3.17 kernel)

Throughput

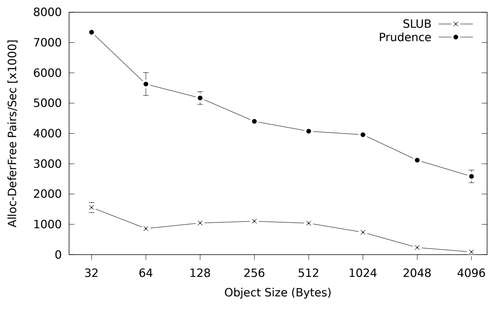


Percentage of deferred frees out of total memory free operations



Improvement in overall performance

Results - Micro Benchmark



Summary

- Synchronization via procrastination has direct impact on the performance of memory allocators
  - *Performance impact can be avoided by having deferred objects visible to memory allocators*
- Deferred frees provide hints about the memory regions that are about to be freed
  - *Optimizations based on hints can be exploited to improve the performance of memory allocators*

Reference: "Prudent Memory Reclamation in Procrastination-Based Synchronization", Aravinda Prasad, K Gopinath, ASPLOS 2016