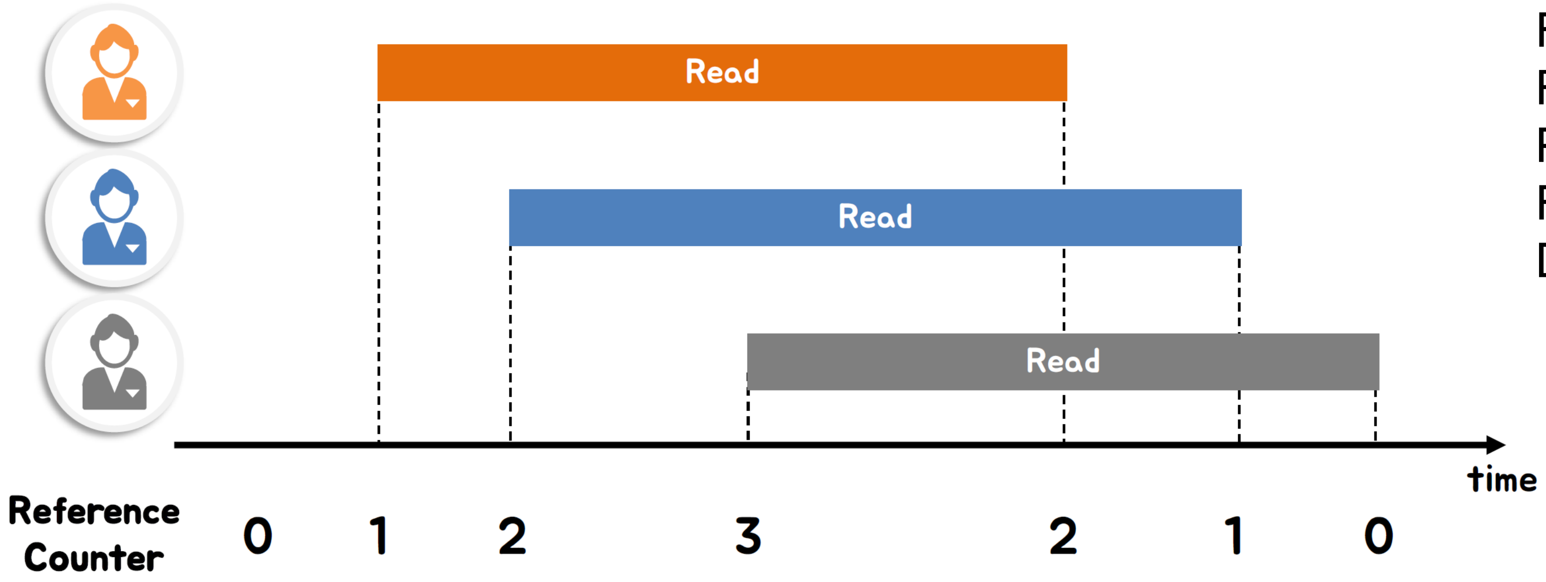
LODIC : Logical Distributed Counting for Scalable File Access

Jeoungahn Park , Taeho Hwang, Jongmoo Choi, Changwoo Min, Youjip Won

USENIX ATC 21

Reference Counter

- Reference counter
  - The number of access for a given object

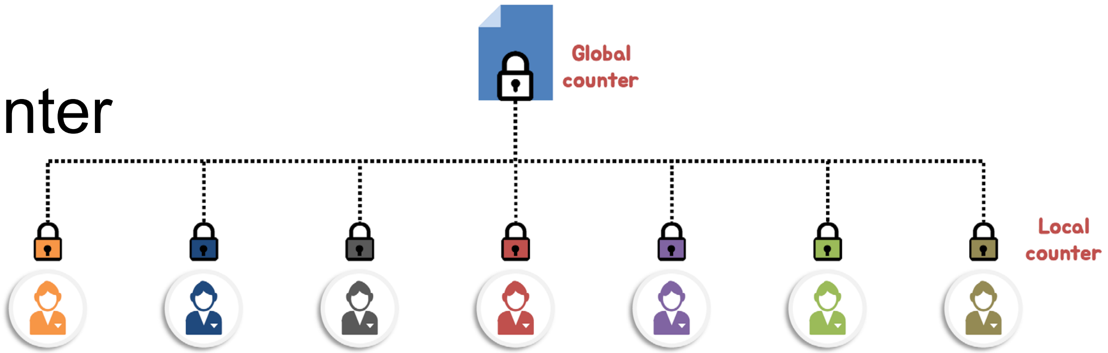


File descriptor table: files_struct
Read write semaphore: rw_semaphore
Physical page frame: page
File object: file
Directory entry: dentry

  - Usage: Avoid the operation on the device file be referenced to the released file, reclaim space

Distributed Reference Counter

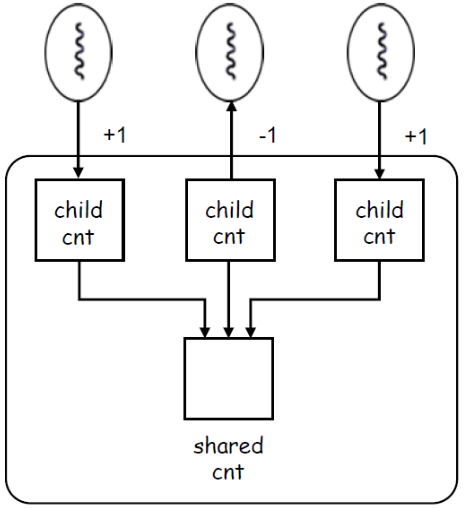
- Working mode: per-core counter
  - Allocate local counter for each core
  - Update operation: update the local counter
  - Counter query scan all local counters



- Memory pressure
  - Memory overhead increase in proportion to number of CPUs and objects
- Query latency
  - For reclaim the object, checking all local counter increase query latency
  - Overhead of obtaining the global state of the counter

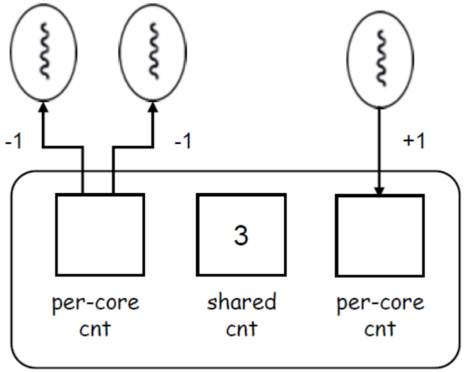
Existing Works on Per-Core Counter

SNZI
[PODC'07]



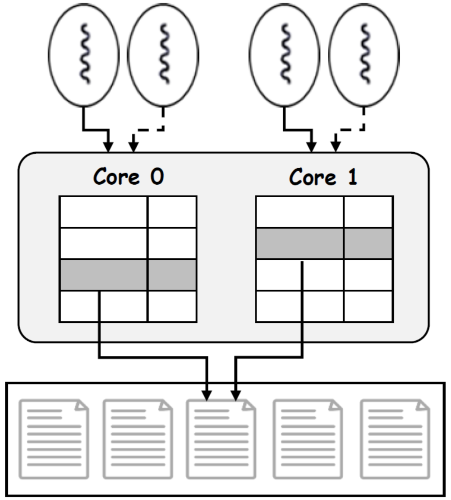
Memory overhead

Sloppy Counter
[OSDI'10]



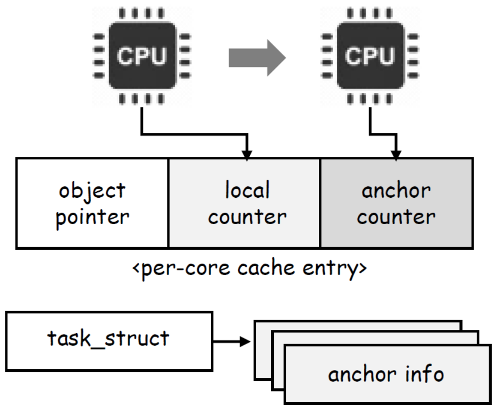
Memory overhead
Query overhead

RefCache
[EuroSys'13]



Overhead of handling hash collision
Memory overhead for counter cache

PayGo
[FAST'19]



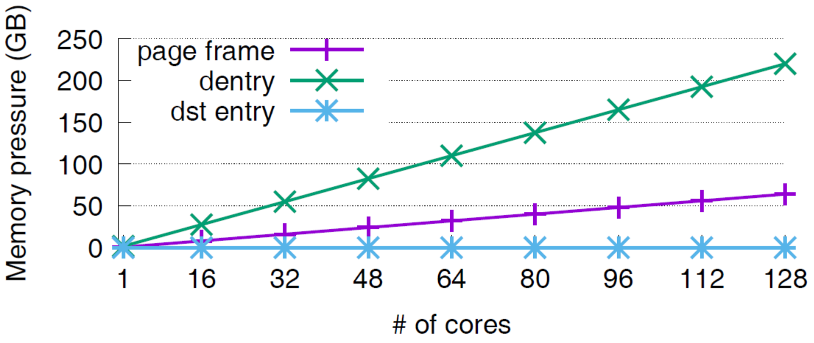
Query overhead
Memory overhead for counter cache

Pe-core Distributed Counter

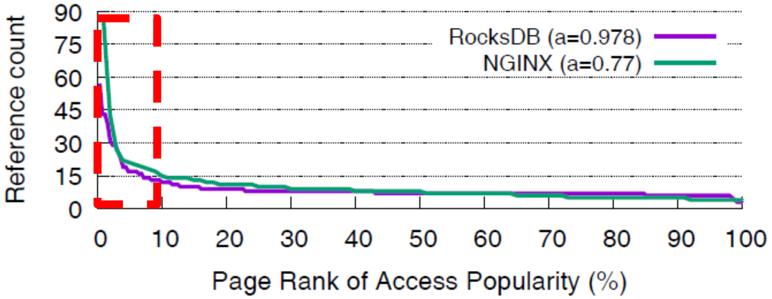
Hash Table Based Counter

Memory and Performance Issues

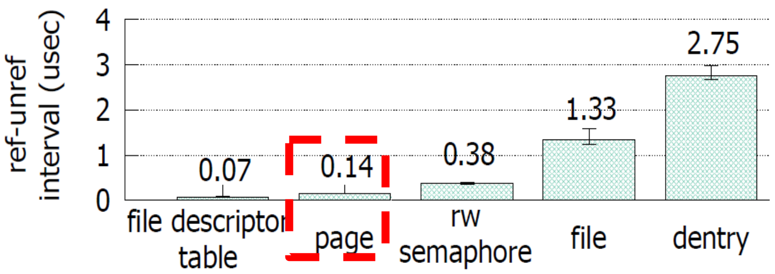
- The growth of computer resources
  - The number of cores is rapidly increasing
  - Main memory is getting larger and larger
  - Manycore scalability becomes a serious issue in the modern OS design
- Characteristics of kernel objects



Population: Memory pressure increased with core numbers

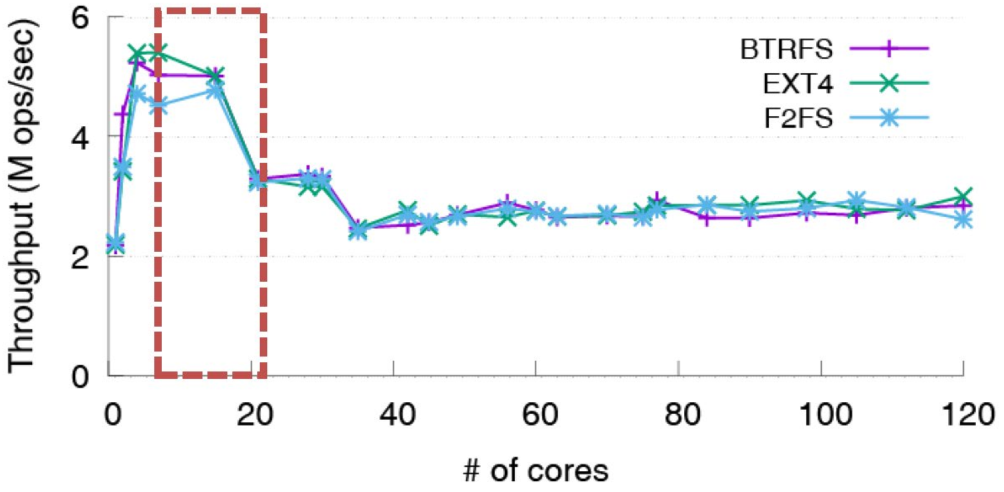


Popularity: Highly skewed popularity



Access Brevity: Very short access duration

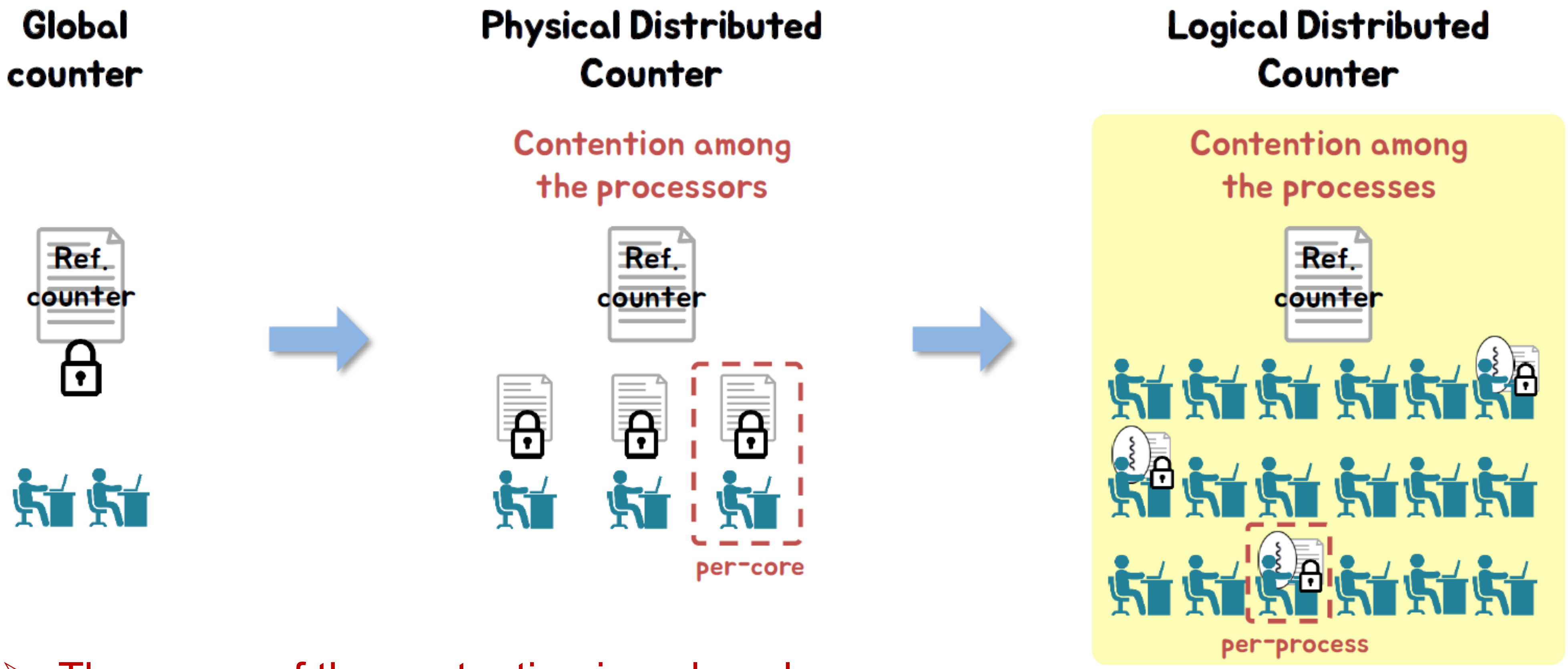
Cacheline Contention Example



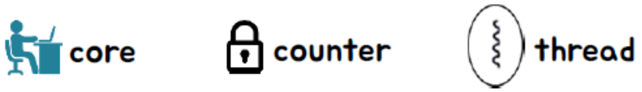
Performance collapse due to cacheline contention

- Contention in update reference counter is driven by contention among the processes, rather than processors
- **Use pre-process counter rather than per-core counter**

Compare Different Counters

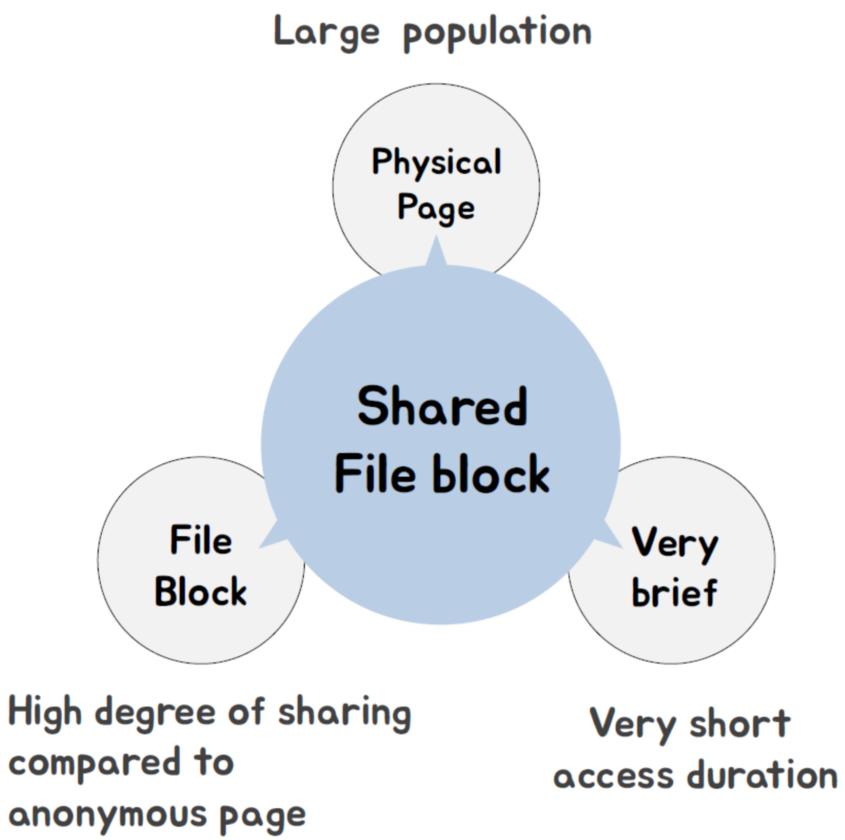


➢ The scope of the contention is reduced



LODIC: Logical Distributed Counter

- LODIC
  - Counter contention is caused by the contention among the processes
  - Distributed counter with local counters are defined in per-process basis
- Used characteristics
  - Popularity : Define the counter with respect to the degree of sharing
  - Access brevity : Not consider the reference split

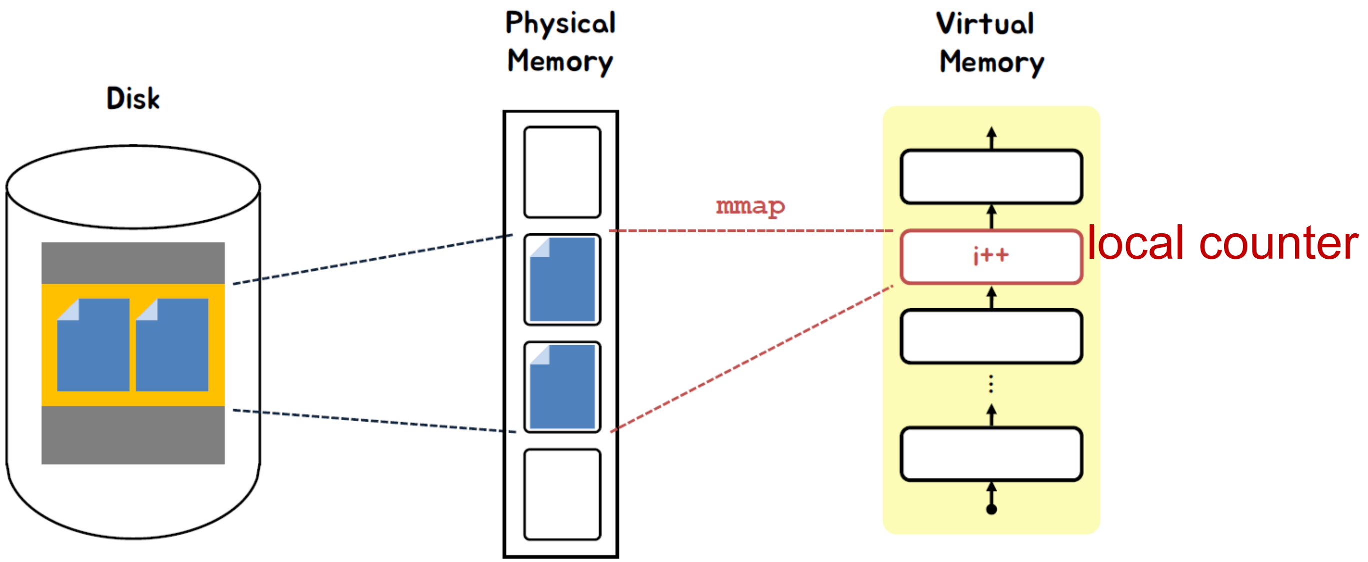


Target and Key Techniques

- Target for LODIC: Higher scalability; Lower query latency; Lower memory overhead
- The number of counters are proportional to the degree of sharing
- Three key techniques:
  - **File mapping**: map file block to process address space
  - **Reverse mapping**: between process address space and file's address space
  - **Counter embedding**: use unused bits in page table entry

File Mapping

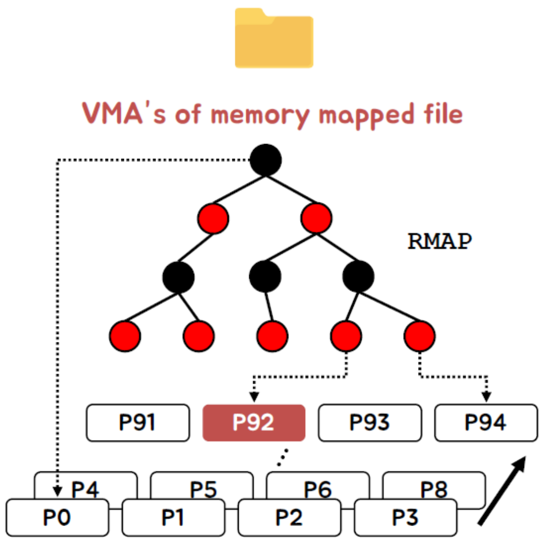
- *Mmaping* file region to the process virtual address space
- Allocates the local counters for the virtual pages in the mapped file region.
- Selective distributed reference counting for hot file blocks



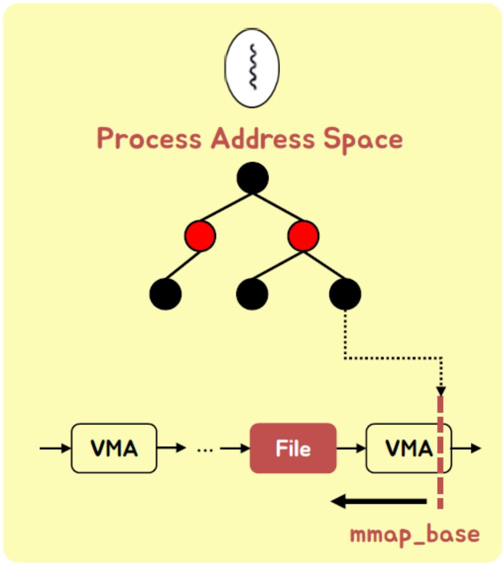
Reverse Mapping

- Legacy reverse mapping which is used in rmap() can degrade the page reclamation performance by 3 times.
- The virtual segment allocation algorithm of Linux tends to place the file mapped segments at the high-end of the process virtual address space.

File-Space based reverse mapping

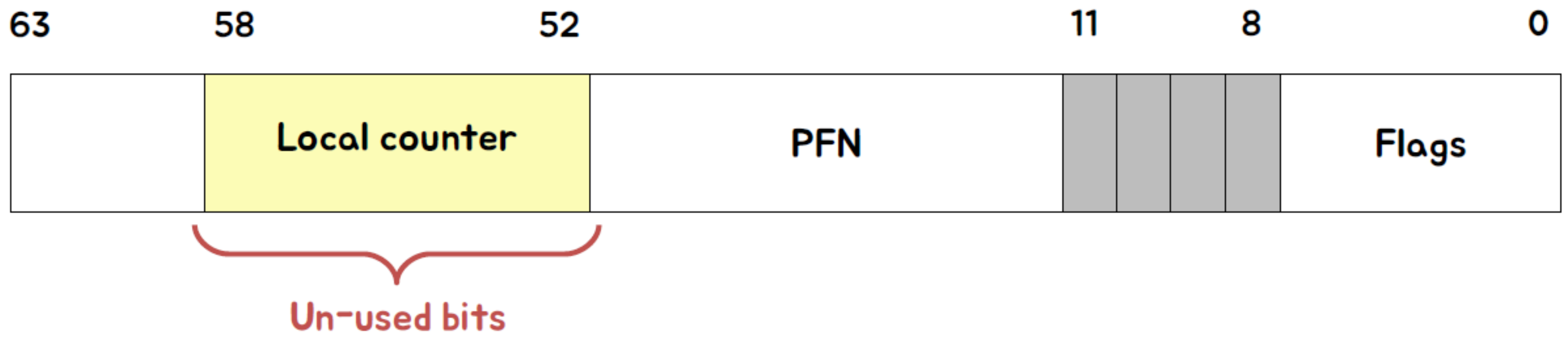


Process-Space based reverse mapping

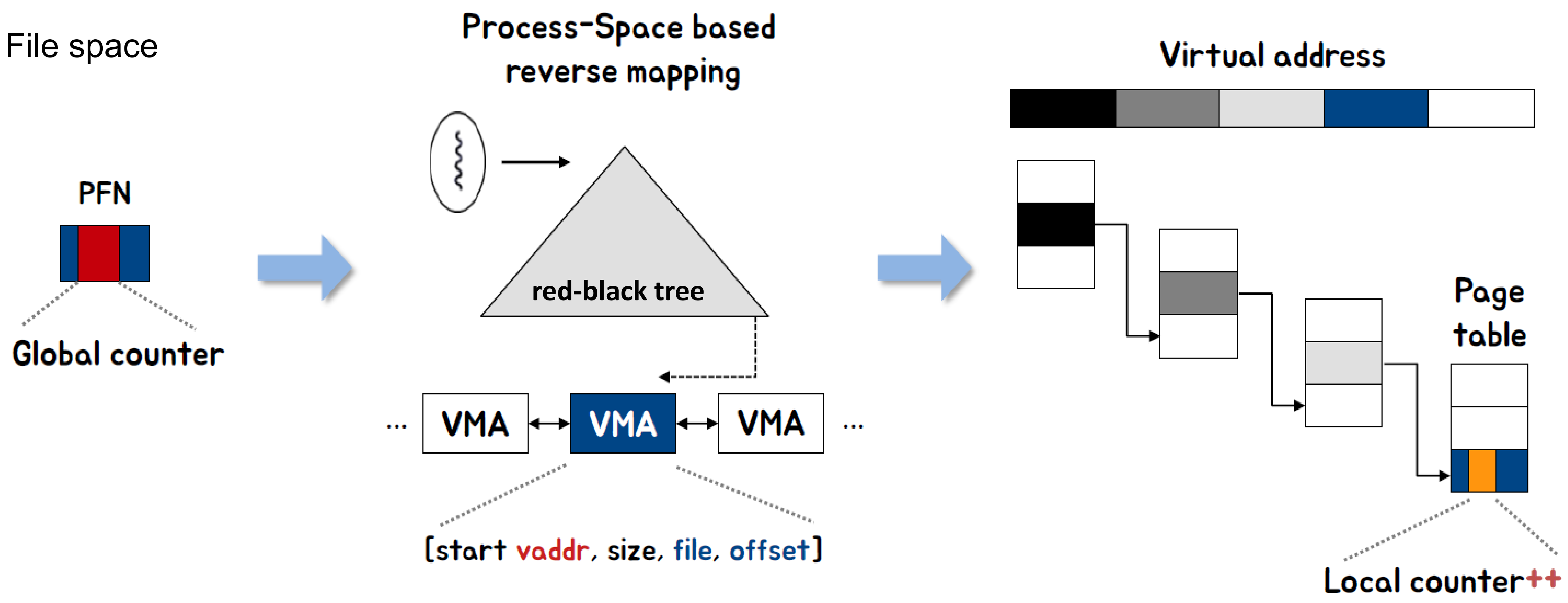


Counter Embedding

- ➢ The unused (ignored) bits in the page table entry (PTE).
- ➢ Local counter overflows infrequently (Global counter exist)
- ➢ Update with atomic CAS instruction



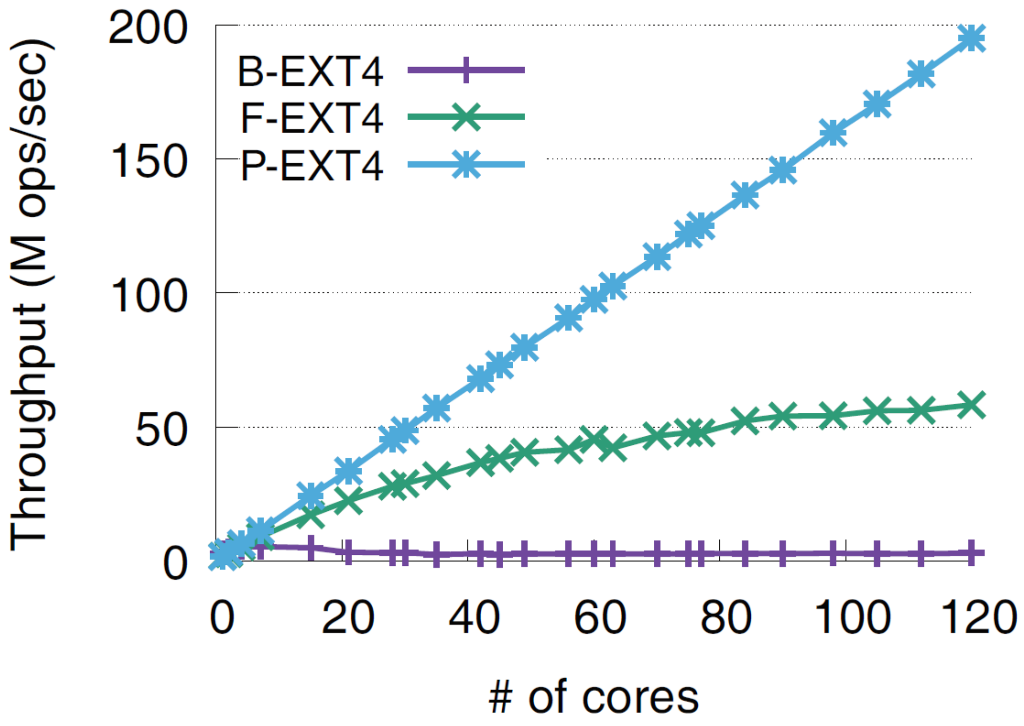
Integration



Evaluation

➢ Throughput on shared file block read

- 120 cores (15 cores/CPU, 8 socket, Intel Xeon E7 8870), 780 GB DRAM, Linux 4. 11. 6
- DRBH Workload on FxMark

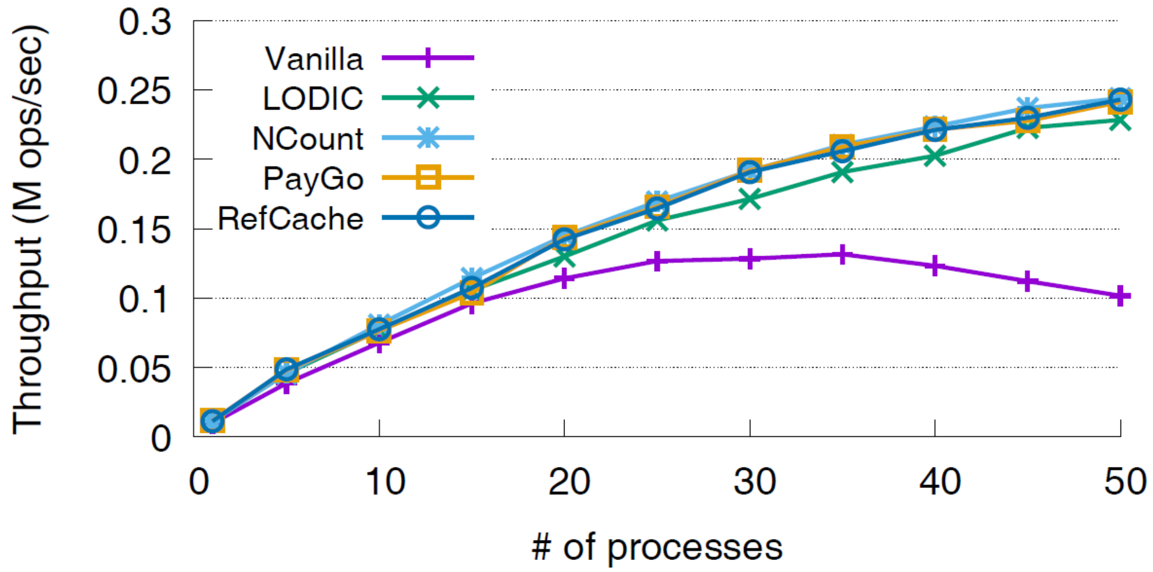


B : Baseline vanilla Linux
F : File-based reverse mapping
P : Process-based reverse mapping

Approximately linear growth with the number of cores

Evaluation

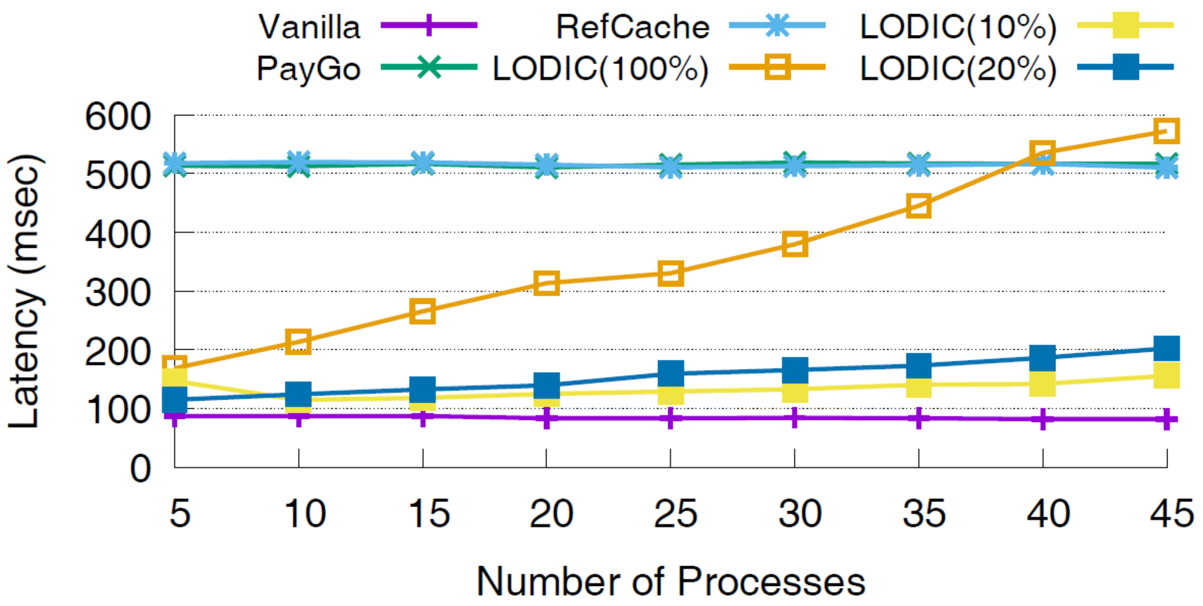
➢ Web server throughput
- 50 client processes, 50 server processes
- NGINX : Reverse proxy server that handles client request
- wrk benchmark : Make the client process to read request for the same file



Up to 2.5X performance improvement at 50 cores

Evaluation

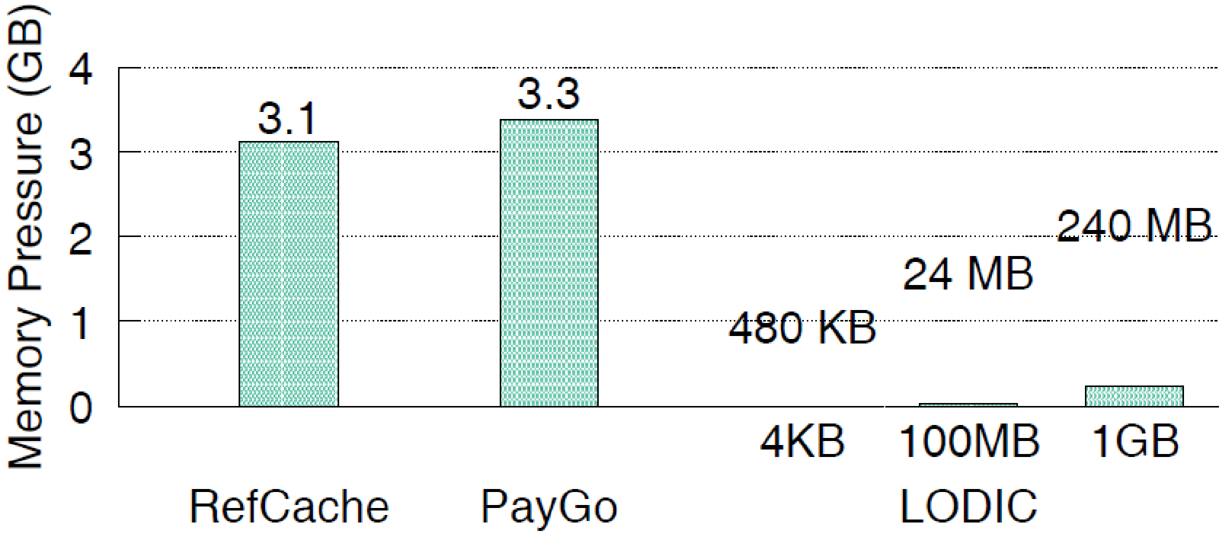
- Counter query latency: reclaiming all page frames
  - *fadvise ()* : System call to reclaim the page
  - File size : 1GB
  - LODIC(x%) : x% of file blocks are mapped



Close to the performance of the Linux kernel using only global counters

Evaluation

- Counter query latency: reclaiming all page frames
  - 120 cores machine, the degree of sharing in LODIC is 120-core
  - 4KB, 100MB, 1GB file size



At least 13X lower memory pressure

Conclusion

- Take process centric view in designing the distributed counting scheme
  - Counter contention is caused by the contention among the processes not by the contention on the processors
  - Number of local counters : With respect to the actual degree of sharing
  - Memory pressure : Almost none
- Striking the balance among the three factors of the reference counter:
  - Memory pressure; Counter query latency; Counter update performance.