Quantifying Memory Overheads of OS in Big Data Stacks

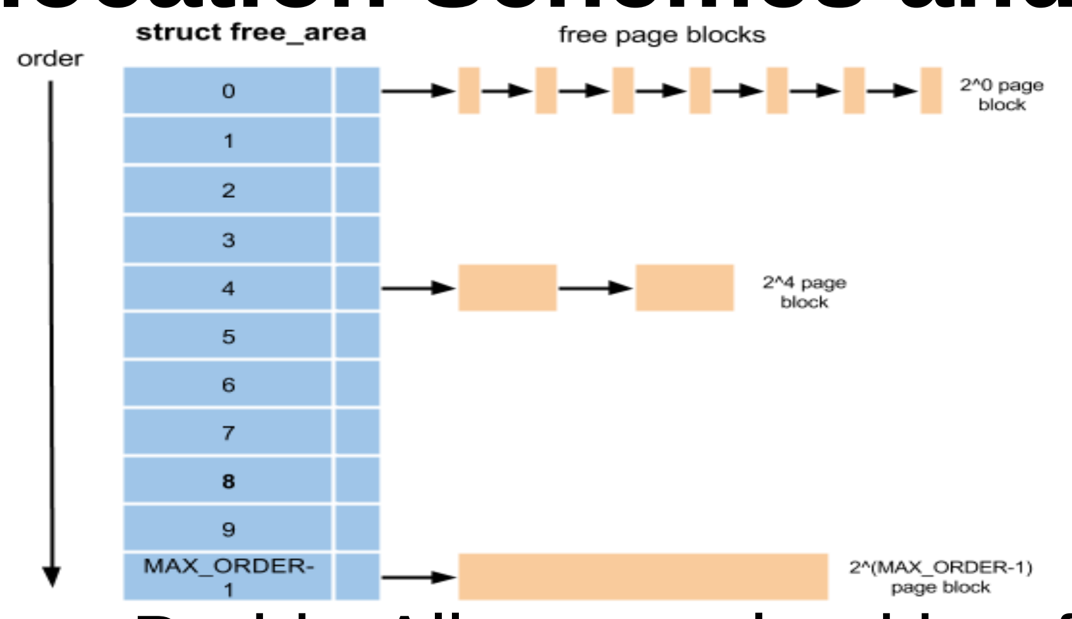
CS744: Big Data Systems – Group 18

Pradeep Kashyap, Karan Talreja, Anshu Raina

Motivation

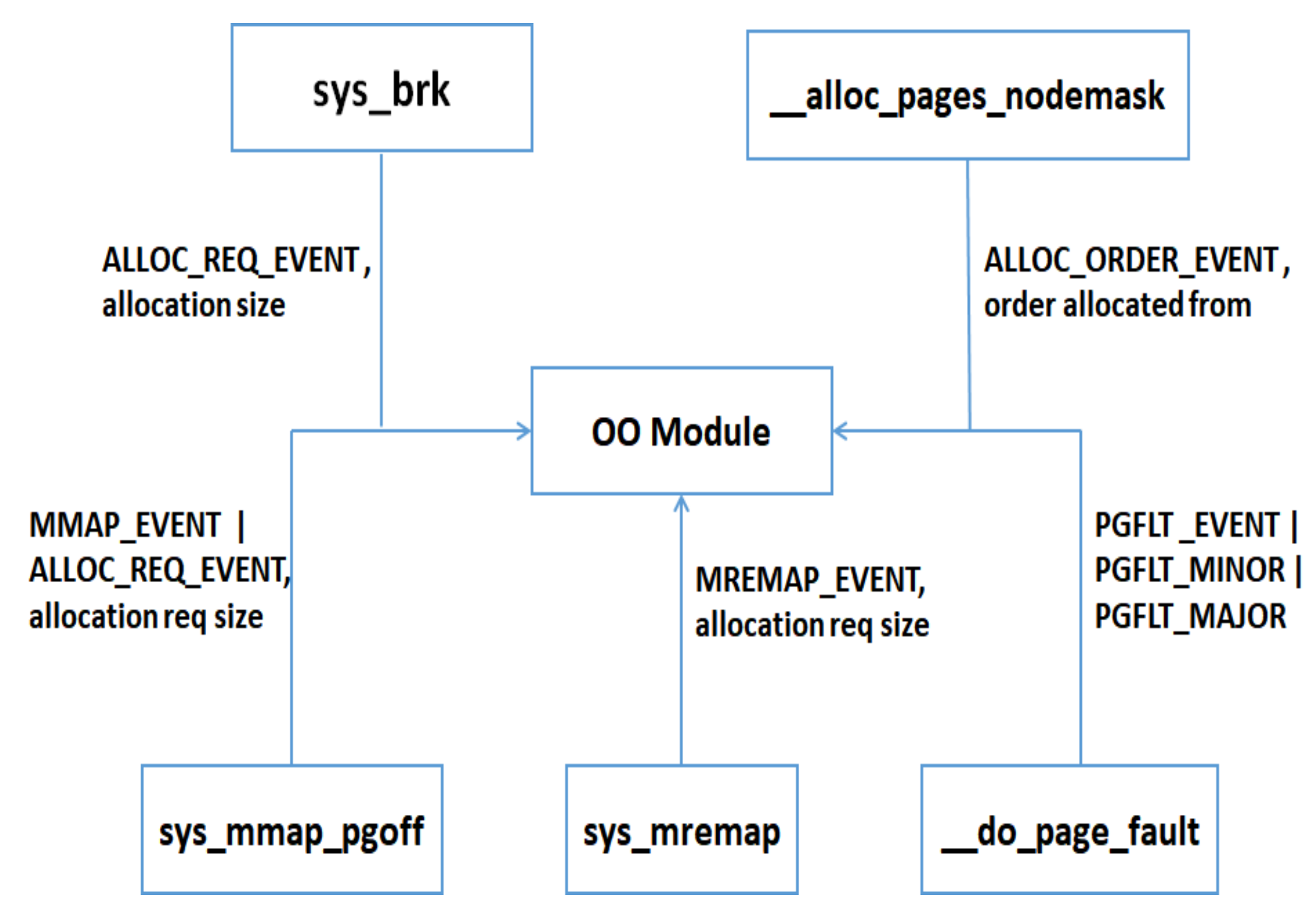
Demand Paging (DP) in general is good for a generic OS because of poorly written applications / frameworks. DP can have an adverse effect on applications which use their memory efficiently. Applications accessing most of their allocated memory would be preempted from processor (due to page faults) in a non-deterministic way based on application's memory access patterns. In our experiments, we are analyzing the overheads of DP on big data frameworks like Map-Reduce, Spark, and Flink to figure out whether it makes sense to always use DP for these frameworks.

Memory Allocation Schemes and Overheads



- Linux uses Binary Buddy Allocator algorithm for page allocation which divides memory into large blocks of pages.
- In DP, when the application tries to access the memory location which was requested before, OS takes a minor page fault.
- Eager paging allocates the memory right away which gets rid of minor page faults at the cost of memory overhead.

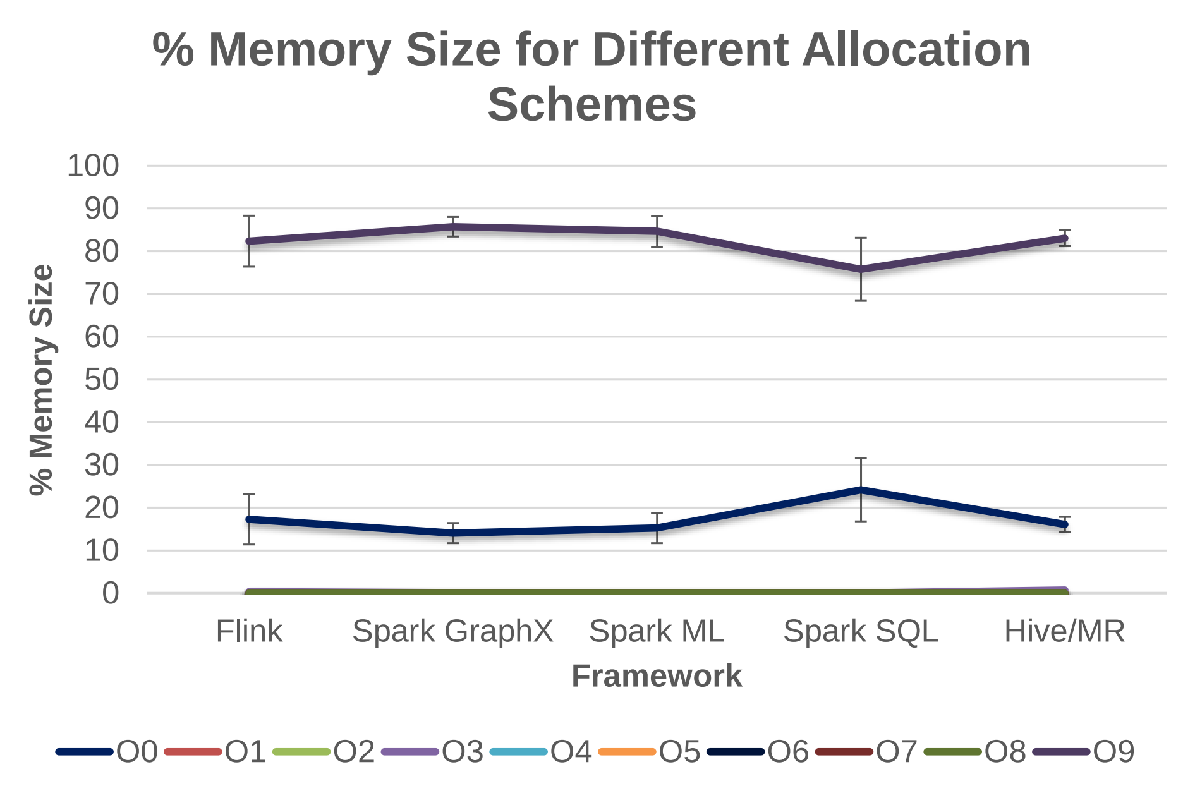
Design/Contributions

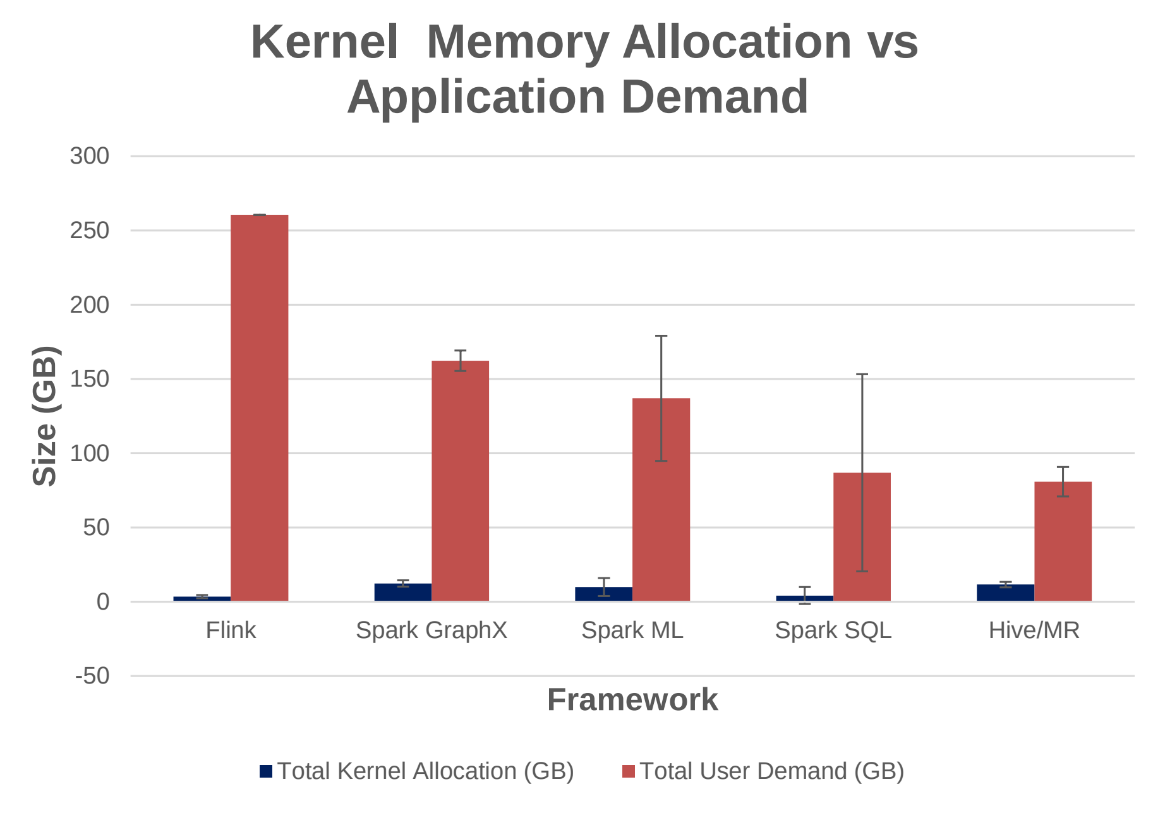

sys_brk
__alloc_pages_nodemask
ALLOC_REQ_EVENT, allocation size
ALLOC_ORDER_EVENT, order allocated from
O0 Module
MMAP_EVENT | ALLOC_REQ_EVENT, allocation req size
MREMAP_EVENT, allocation req size
PGFLT_EVENT | PGFLT_MINOR | PGFLT_MAJOR
sys_mmap_pgoff
sys_mremap
__do_page_fault

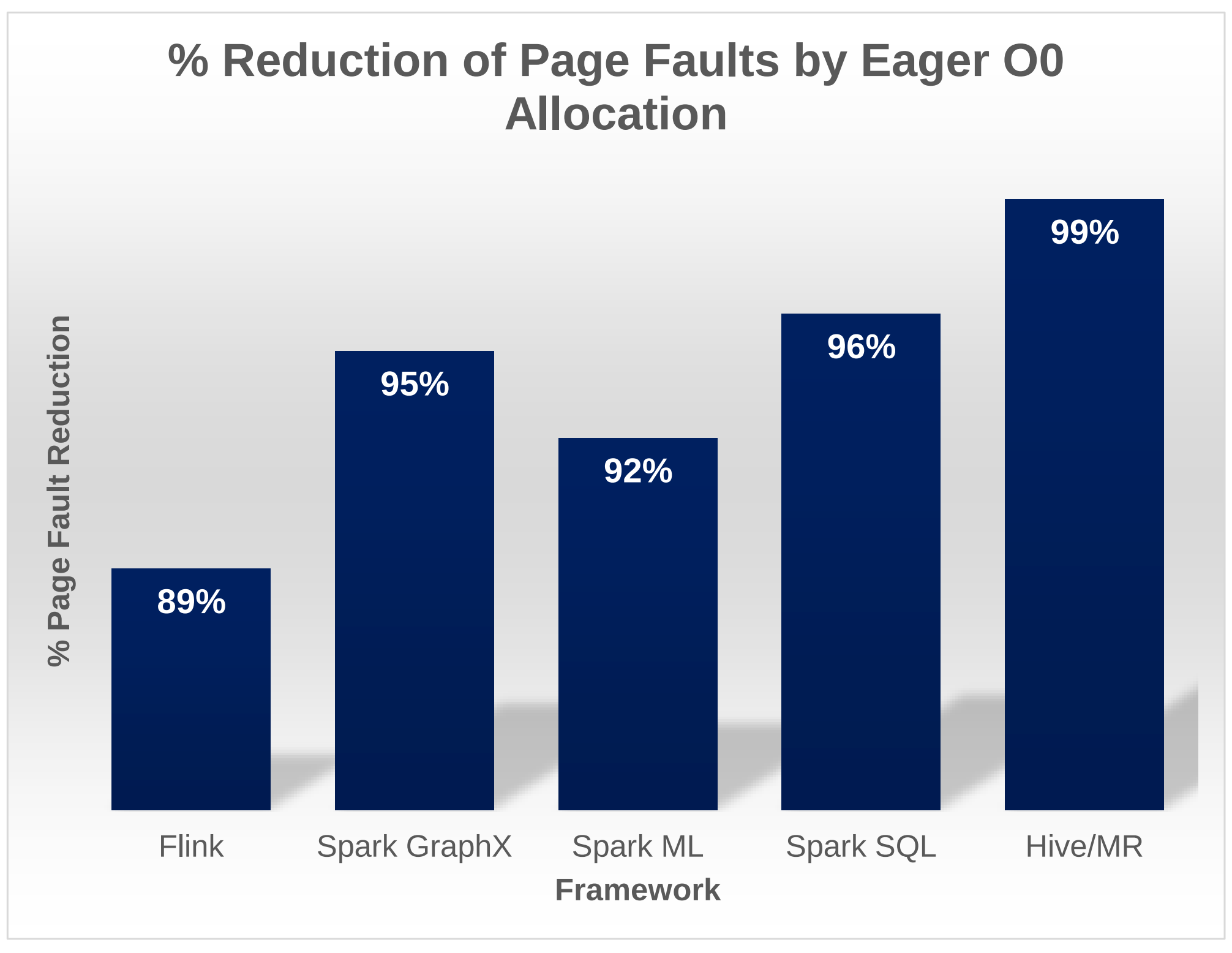
Event based mechanism to collect statistics

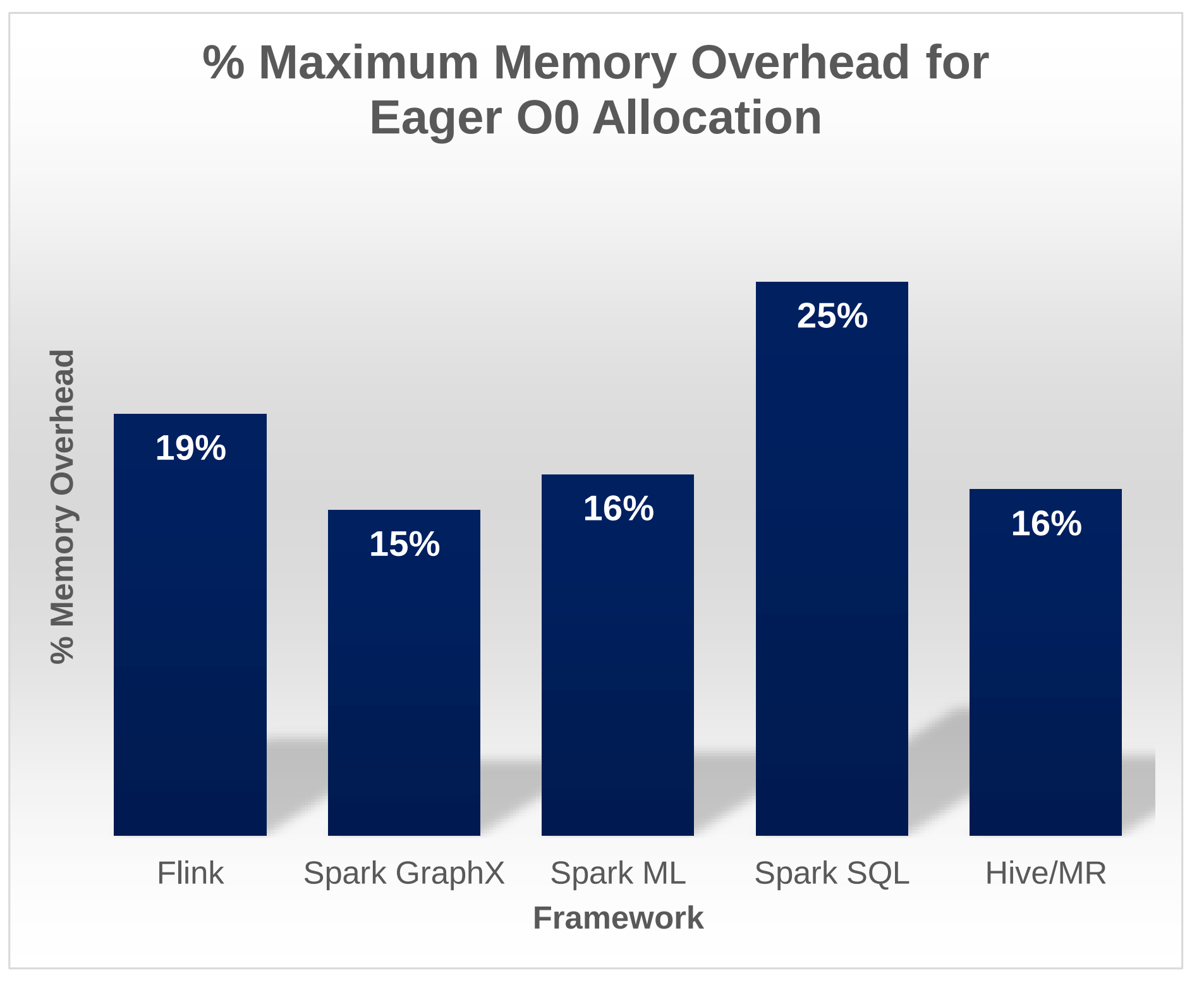
- The major parameters that we have captured to analyze the overheads are time spent in kernel, number of page fault exceptions, and size and number of memory allocations.
- We use an event based scheme to capture statistics. When an event like page fault or memory allocation occurs, our module is notified with an event ID and certain values. Based on these event IDs, we store statistics for applications.
- We developed a general kernel framework for analyzing memory allocation profile for arbitrary big data workloads.
- The framework tracks an application process hierarchy side stepping the issues seen using tools such as ftrace.

Evaluation









- The amount of memory allocated by kernel for these applications is highest using O9 scheme followed by O0. Note that O0 allocates pages of 4KB and O9 allocates pages of 2 MB.
- The dominance of O9 scheme shows that the kernel prefers larger sized pages for these applications.
- Kernel allocates a small fraction of requested pages depending on the access pattern of the application.
- The gap between accessed memory and requested memory is larger for streaming frameworks as compared to batch analytics frameworks.
- We propose eager page allocation for O0 scheme since it reduces the number of page faults dramatically without incurring significant memory overhead.
- Results show that we can get up to 99% page fault reduction with at most 25% memory overhead by doing eager O0 allocation.

Conclusion/Future Work

- Demand Paging is not always useful. While OS does a good job of allocating memory to applications, it may not be very good for big data frameworks in terms of performance penalty incurred for page faults.
- Eager paging can remove minor page faults but with memory overhead. Since O0 has largest number of page faults with least page size, eager O0 allocation gives huge benefits in terms of reduction of page faults with very less memory overhead.
- In future, we hope to analyze other overheads pertaining to file systems, networks etc. for big data applications.

References

[1] A. Baumann, P. Barham, P.-E. Dagand, T. Harris, R. Isaacs, S. Peter, T. Roscoe, A. Schüpbach, and A. Singhania. The multikernel: a new OS architecture for scalable multicore systems. In SOSP, 2009.
2] J. Giceva, G. Zellweger, G. Alonso, and T. Rosco. Customized OS support for data-processing. In DaMoN, 2016.