Paper title: Practical, transparent operating system support for super pages
Paper authors: Juan Navarro, Sitaram Iyer, Peter Druschel, Alan Cox
Your name: Parth Kansara (115135130)

What problem does the paper address? How does it relate to and improve upon previous work in its domain?
Large memory pages, called superpages, enhance the TLB coverage and boost performance for many applications but effective implementation of these superpages entails an overhead cost for the operating system. Further, very large superpages may inflate the application footprints, and are hence replaced by a mixture of page sizes which may lead to memory fragmentation. Operating systems are thus required to administer the fragmentation without any degradation of system performance.

Talluri and Hill employed a scheme based on the reservation of a region at page-fault time, which is promoted once the number of frames in use reaches a promotion-threshold. **IRIX** and **HP-UX** used a scheme where several contiguous frames are allocated and immediately promoted to a superpage. This scheme is not transparent as it needs to determine the optimum superpage size for the best performance.

Romer et al. utilized an online cost-benefit analysis, based on the number of TLB misses, to assess the situations when the benefit of superpages exceeds the overhead of relocation-based superpage promotion. This technique has a consistently lower performance than reservation-based approach in the absence of memory contention.

Talluri and Hill suggested the use of partial-subblock TLBs, which allow for missing base pages. However, it might not be possible to extend the partial-subblock TLBs to multiple superpage sizes. **Fang et al.** leverage additional address translation for eliminating the contiguity requirement of the superpages.

What are the key contributions of the paper?
The paper proposes a transparent system for managing the superpages, which can optimize the various tradeoffs of allocating superpages. This system is based on the reservation-based approach and uses multiple superpage sizes. When a page fault occurs, a region equal to the preferred superpage size is reserved and this set of frames is added to the reservation list. For fixed-size memory objects, the preferred superpage size is the size of the largest, aligned superpage that has the faulting page, given that it does not overlap with any pages, and it does not reach beyond the end of the object. For dynamically sized memory objects, the preferred superpage size is determined in a similar way, except the condition of reaching beyond the end of the object is dropped.

Whenever an allocation is requested, the system preempts existing unused reservations instead of denying the allocation. To control fragmentation, the buddy allocator performs coalescing of existing memory regions whenever possible. Only regions that are fully populated by the application are promoted to superpages of increasing sizes. For demotion of a superpage, a recursive approach is used when the page daemon resets the reference bit of a superpage's base page. If all the base pages of a demoted superpage are referenced, incremental repromotions are allowed. Clean superpages are demoted whenever they are being written to, and then repromoted if all the base pages are dirty to reduce I/O of writing partially dirty superpages to memory.

Reservation lists are leveraged to track the reserved page frames that are not fully populated. It maintains a reservation list for every page size supported by the hardware, excluding the largest superpage size. Each list contains reservations sorted by the time of their most recent page frame allocation so that preemption is efficient. Within each memory object, the allocated base pages are tracked using population maps. This population map helps to lookup reserved frames, avoid overlaps, decide promotions and assist in the preemptions.

Briefly describe how the paper's experimental methodology supports the paper's conclusions.
The paper implements the transparent system on FreeBSD-4.3 kernel, and is able to use superpage support to eliminate 100% of the TLB misses. It is also evident that allowing the system to dynamically choose the best page size provides higher performance. After a state of fragmentation, the proposed system is able to recover and serve almost 40 out of the 60 requested superpages, while the traditional system was unable to serve any superpage. During concurrent execution, the traditional system served 3.3% of the superpage requests while the proposed system was able to serve 29.9% of the requests. This concludes that the proposed system is able to effectively manage and leverage the superpages.

Write down one question you may want to ask the authors.
What could be the foreseeable performance downgrades of the demotion and incremental repromotion of the superpages?