Introduction to Superpages
Background

• Virtual Memory systems divide memory into pages
• Pages are allotted by the operating system for each process
• Every memory access that is paged will go through a virtual to physical address translation
• For quicker translation, page tables are cached in TLBs.
• TLBs cannot be too big as they are in the critical path so multi-level TLBs are used
Pros and Cons of Page Sizes

- Smaller pages reduce internal fragmentation but occupy more memory (more entries) compared to a larger one.
- When swapping from memory, access times dominate so larger pages are better.
  - Recent trends: 100 times improvement in transfer rate
  - 3 times improvement in access times [1]
- 128 entry TLB for a 4KB page size = $\frac{1}{2}$ MB
  So code reach is very little for modern day applications
- Larger pages also cause coherency issues
Intel Support for Multiple Size Pages

- Hierarchical page tables
- Each level is 4KB in size
  - 32 bit, 1024 entries (4B per entry)
  - 64 bit, 512 entries (8B per entry)
- 64 bit architecture -- 4 levels
  - Nomenclature: Page map level4, page directory pointers, page directories and page tables
  - Supported sizes 4KB, 2MB and 1GB
Intel Support for Multiple Size Pages

Paging Structure for 4KB Pages [2]
Intel Support for Multiple Size Pages

Paging Structure for 2MB Pages [2]
Intel Support for Multiple Size Pages

Paging Structure for 1GB Pages [2]
Intel Support for Multiple Size Pages

Overall Diagram [3]
Implementation of Superpages

• Allocation [6]
  – Superpage is allocated and loaded at the first page fault. All subpages are loaded and TLB entry is made
  – Size of superpages is either supplied by the user or detected by the operating system based on the size of the request
  – Further TLB misses are avoided once pages are loaded
  – Better to transfer large page from disk than many shorter ones
Implementation of Superpages

• Reservation [6]
  - Reservation for superpage is made but not loaded into memory
  - Loading of pages is performed at the base page size one at a time
  - All entries of the superpage are loaded when a particular threshold of loading base pages is met.
  - Advantage of ensuring that superpage will be used effectively before loading it
  - Also reduces start-up time of a process
Implementation of Superpages

- Relocation [6]
  - Base pages are loaded by default but memory usage is monitored
  - Decision to promote to superpage taken by operating system
  - Contiguous locations are found and pages are copied or brought in from secondary storage
Linux Support for Multipage Sizes

- Hugetlbfs [5] library supports hugepages (Linux 2.6.16 onwards)
  - Function calls for using of huge pages
    - Application developer driven
  - Linker scripts for dynamically linked programs
    - Requires program to be relinked but no source code modification is necessary
    - Compiler directives for linking are available
- Environment variables needs to be set for enabling hugepages, enabling sharing of huge pages, etc
- If the hardware supports multiple page sizes, heap page size can be chosen as per program needs
- Information on current allocation can be found at /proc/meminfo
## Case Study: Big Memory Servers

### TLB miss rates for server benchmarks for different page sizes [4]

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Base Pages (4KB)</th>
<th>Large Pages (2MB)</th>
<th>Huge Pages (1GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>graph500</td>
<td>51.1</td>
<td>9.9</td>
<td>1.5</td>
</tr>
<tr>
<td>memcached</td>
<td>10.3</td>
<td>6.4</td>
<td>4.1</td>
</tr>
<tr>
<td>MySQL</td>
<td>6.0</td>
<td>4.9</td>
<td>4.3</td>
</tr>
<tr>
<td>NPB:BT</td>
<td>5.1</td>
<td>1.2</td>
<td>0.06</td>
</tr>
<tr>
<td>NPB:CG</td>
<td>30.2</td>
<td>1.4</td>
<td>7.1</td>
</tr>
<tr>
<td>GUPS</td>
<td>83.1</td>
<td>53.2</td>
<td>18.3</td>
</tr>
</tbody>
</table>

TLB miss rates for server benchmarks for different page sizes [4]
Case Study: Big Memory Servers

Memory allocation with time [4]

<table>
<thead>
<tr>
<th>Memory allocated in GB</th>
<th>Percentage of allocated memory with read-write permission</th>
</tr>
</thead>
<tbody>
<tr>
<td>graph500</td>
<td>99.96%</td>
</tr>
<tr>
<td>memcached</td>
<td>99.38%</td>
</tr>
<tr>
<td>MySQL</td>
<td>99.94%</td>
</tr>
<tr>
<td>NPB:BT</td>
<td>99.97%</td>
</tr>
<tr>
<td>NPB:CG</td>
<td>99.97%</td>
</tr>
<tr>
<td>GUPS</td>
<td>99.98%</td>
</tr>
</tbody>
</table>

Percentages of pages with read-write permissions [4]
Big Memory Servers

• Inference
  – Dynamic memory allocation is predictable
  – Accesses permissions for pages is predictable (RW, R, W, etc)
  – Swapping is almost non-existent due to careful optimizations and sufficient memory
  – Fragmentation problem managed internally
  – Scaling not possible with pages

• Conclusion
  – Makes a case for segmentation rather than pages
  – Fixed size of memory allocation not suitable for these applications
References