Transparent Hugepages on Steroids

Nitin Gupta
Principal Software Engineer, Oracle
Transparent Hugepages (THP)

- Replaces candidate default-sized pages with hugepages
- No application changes necessary
- Helps reduce TLB misses and is likely to improve application performance
## Hugepage support by architectures

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Default Page Size</th>
<th>Hugepages</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>PTE</td>
</tr>
<tr>
<td>ARM64</td>
<td>4K</td>
<td>64K</td>
</tr>
<tr>
<td></td>
<td>16K</td>
<td>2M</td>
</tr>
<tr>
<td></td>
<td>64K</td>
<td>2M</td>
</tr>
<tr>
<td>SPARC</td>
<td>8K</td>
<td>64K</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x86</td>
<td>4K</td>
<td></td>
</tr>
</tbody>
</table>
THP: keep it disabled?

```
echo never > /sys/kernel/mm/transparent_hugepage/enabled
```

- **Disabling THP for TokuDB**
  - Storage engine for MySQL, MariaDB
- **Disable THP on MongoDB**
- **Disable THP for Couchbase Server**
- **Redis**
- Numerous other DB vendors
THP: keep it disabled?

- Memory bloat (MySQL, MariaDB)
  - MADV_DONTNEED hint ineffective within a hugepage backed region
  - Need proactive breaking of hugepage back into smaller pages based on these hints.
- Allocation delays (Couchbase)
  - THP default mode stalls until hugepage is available
  - Change default to defer / madvise / defer+madvise?
Limitations of THP in Linux

- THP considers only one hugepage size when replacing small pages
  - The "default" hugepage size: 2M on x86, 8M on SPARC, ..
- Other available hugepage sizes are ignored
- THP chosen default hugepage size might not be available for allocation
- THP ineffective for memory regions smaller than the default hugepage size
- Only anonymous (heap) regions are considered
  - Text (executable) regions would benefit from hugepages too
THP-next: Use larger pages

V_o + 512M

65536 TLB entries

V_o
(aligned at 256M boundary)

THP

8K
8K
...
8K

256M
256M

THP-next

8M
8M
...
8M

256M

64 TLB entries

2 TLB entries

(aligned at 256M boundary)
THP-next: Fallback to smaller pages

$V_o + 512M$

65536 TLB entries

$V_o$
(aligned at 256M boundary)

THP

63 + 1024 = 1087 TLB entries

Fallback to default page size (8K)

1 + 31 + 128 = 160 TLB entries

Fallback to next smaller hugepage (64K)

1 + 32 = 33 TLB entries

Fallback to next smaller hugepage (8M)

THP

256M

256M

8K

8K

8K

8K

8M

8M

8M

8M

8K

8K

8K

8K

8K

8K
THP-next: Greedy splitting

89608 TLB entries

$V_o + 700M + 64K$

256M boundary

(aligned at 64K boundary)

512 + 87 + 8 = 607 TLB entries

8M boundary

THP ignored

64 + 23 + 2 + 1 = 90 TLB entries

4M = 64 x 64K

184M = 23 x 8M

512M = 2 x 256M

64K = 1 x 64K
THP-next: small regions and text segments

- THP currently does not consider text regions
- Many text regions often smaller than default hugepage size
  - ~1MB for executable libc region
    - r-xp 00000000 08:51 1182914 /lib/x86_64-linux-gnu/libc-2.23.so
- THP-next aims to add support for text regions too
  - Exploits smaller hugepages like 64K
Summary

THP-next

- Uses all supported hugepage sizes
- Fallback to smaller hugepage sizes if required
- Picks most appropriate page size based on region alignment
- Support for text/executable memory regions
- Tracks actual hugepage usage
Milestones

- Recently available in mainline Linux
  - sparc64: Multi-page size support (make code generic, 256M support)
  - sparc64: Add 64K page size support
  - sparc64: Add support for 2G hugepages
  - sparc64: Add 16GB hugepage support

- Under internal review
  - Make THP code generic to support all page sizes
  - Greedy region splitting
  - Fallback to smaller hugepages
Thank you

natin.m.gupta@oracle.com
nitingupta910
nitingupta910