Promoting Huge pages
Linux Plumbers 2018 Vancouver

Mike Kravetz

Christoph Lameter <cl@linux.com> @qant

State of Huge pages

- Huegtlbfs
  - Original/oldest method
  - System config and application changes
  - Good for ‘single purpose’ use cases
- Transparent Huge Pages (THP)
  - Enabled by default on most distros
  - No sys config or application changes required
    - Although, desirable for optimal usage
- DAX (Persistent Memory)
  - Uses 2M and 1G mappings by default

hugetlbfs

Preallocation at boot or early sys init time

Memory ONLY available for hugetlbfs

Applications must change (open/mmap)

Not all file operations (such as write)

Multiple huge page sizes (arch dependent)

Dynamic allocation ‘possible’ but troublesome

- SIGBUS catcher

Transparent Huge Pages

System wide setting: always, madvise, never

Now for shm/tmpfs as well as Anon memory

No application changes ‘required’

posix_memalign() and madvise()

Single huge page size (PMD_SIZE)

Problems with Huge pages

Page cache use?

Larger base pages instead?

Configuring system for the needs of different apps may require reboot

Work in progress

More support for file based mappings

Transparent Hugepage support for Pagecache

Larger base pages instead

Ability to reserve pages of arbitrary order.

Further out work

Huge page based VM under a hypervisor (stripped down Linux kernel?)

Automatic huge page configuration via Ansible and various diagnostics

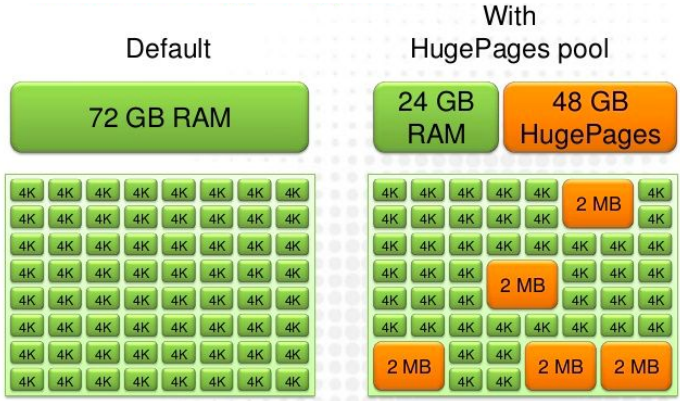
Pressure Hardware vendors to do the right thing and increase base page size. Non x86 platforms usually do not have this issue.

Deal with fragmentation through increasing the mobility of kernel object and increase of mobility of 4k pages (Babka).

Special issue with dentries and inode mobility. Work on slabs focused on xarray support.

Huge Memory

- Typical memory is handled in chunks of base page size (Intel 4k, IBM PowerX 64K, ARM 64K)
- Systems support larger memory chunks of memory called Huge pages (Intel 2M)
- Must be pre configured on boot in order to guarantee that they are available
- Required often for I/O bottlenecks on Intel.
- 4TB requires 1 billion descriptors with 4K pages. Most of this is needed to compensate for architectural problems on Intel. Intel processors have difficulties using modern SSDs and high speed devices without this.
- Large contiguous segments (I/O performance)
- Fragmentation issues
- Uses files on a special file system that must be explicitly requested by mmap operations from special files.



THE LINUX FOUNDATION

Transparent Huge Pages

- Allows the use of huge pages “transparently” without changes to the applications code.
- Improves performance (less TLB faults, more contiguous memory, less management by the OS)
- But less efficient than manual Huge pages
- Defragmentation requires to create more THPs
- madvise() to control THP allocations
- A potential to waste memory
- Scans memory to find contiguous pages that can be coalesced to huge pages
- ***Not supported (yet) for files on disks***