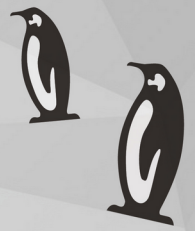
Limitations of Tuning glibc malloc

By Patrick McGehearty

LINUX
PLUMBERS
CONFERENCE > September 20-24, 2021

Malloc usage

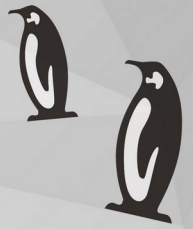


malloc/free is widely used directly in C and by runtime Systems in many other languages

glibc's malloc should give competitive performance.

- Reduces desire to use alternatives such as jemalloc or TCMalloc.
- Easier to support standard malloc than many alternatives.
- Knowledge of tunable benefits can help

Optimizing malloc



Different usage patterns will favor different strategies.

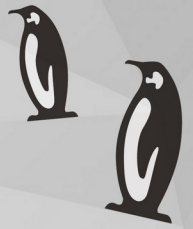
Some applications malloc/free many small chunks.

Some applications allocate large chunks for long durations.

sbrk and mmap have different benefits and limitations.

No single best solution.

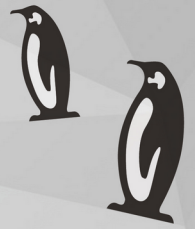
sbrk() vs mmap()



sbrk() obtains additional memory by expanding application data segment. Storage is only returned to the system when a contiguous segment of sufficient size is available at the end of the data segment. This approach can lead to memory fragmentation but typically will mean less system overhead from allocation of pages.

mmap() obtains sufficient additional pages independently mapped for the current allocation request. When the corresponding free() occurs all the pages are immediately returned to the operating system. The advantage is much less wasted memory but the disadvantage is much higher system overhead as the pages are zero'ed between each free() and malloc()

Measuring malloc performance

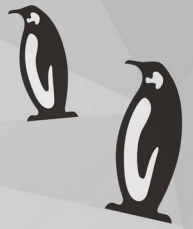


Current presentation will use:
- Single measurement tool to show potential benefits of tuning

Remember: your application will be different.

libMicro benchmark set includes tests for malloc performance.
- The tests have many options.
- Tests used here will allocate 16 identical blocks of memory and then free them.
- The block size will be varied from 16 Kbytes up to 32 Mbytes.

malloc Tunables



MALLOC_MMAP_MAX_ (set to 0 to disable mmap)
default= 64K active mmap allocations

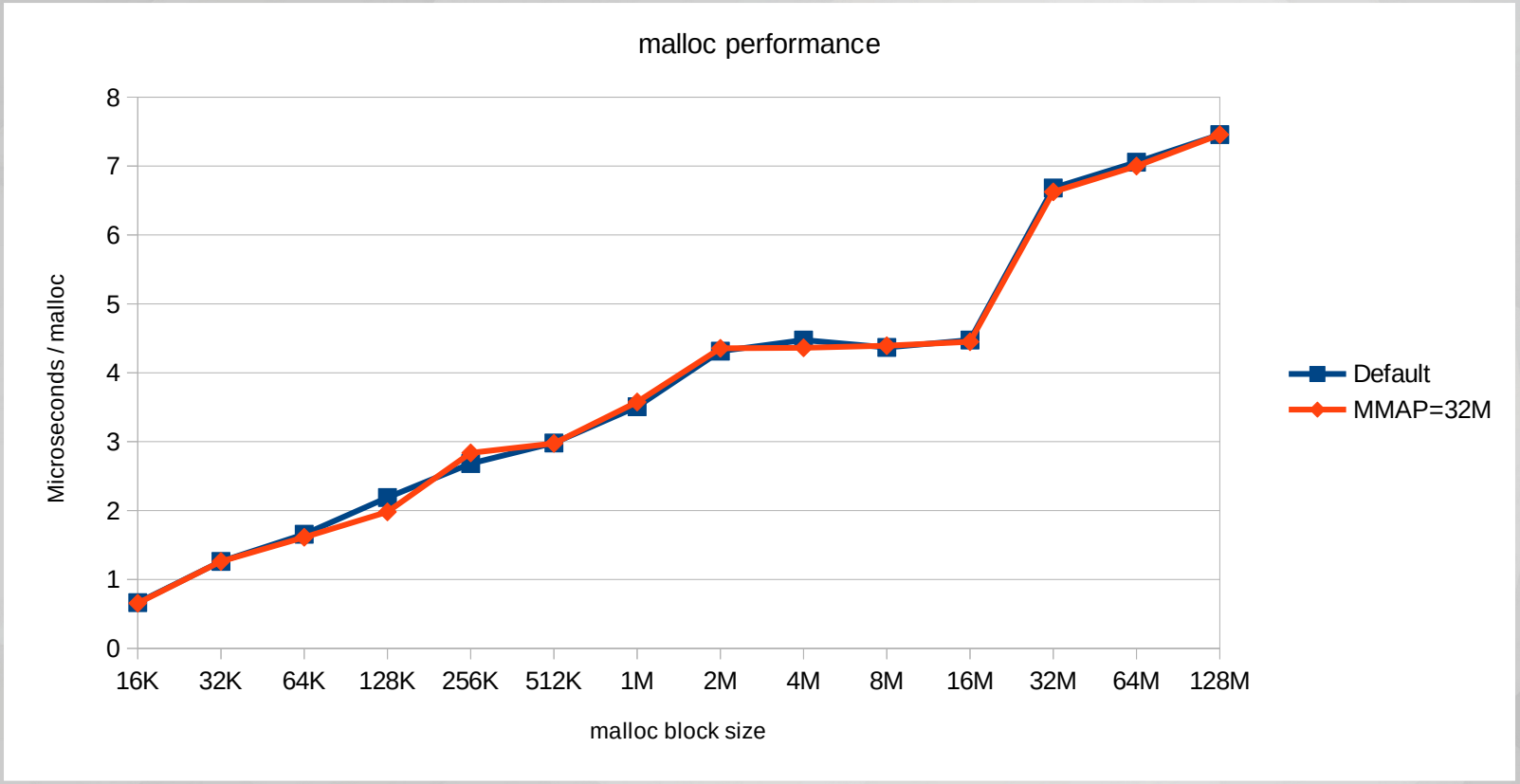
MALLOC_MMAP_THRESHOLD_ (sets limit for mmap)
minimum 128K, maximum 32MB (dynamically adjusts)

MALLOC_TRIM_THRESHOLD_ (set to -1 to disable trimming)
default = 128K
malloc returns unused sbrk()/heap memory when end of heap on free list is greater than the trim threshold

Default values shown are for 64 bit environments.

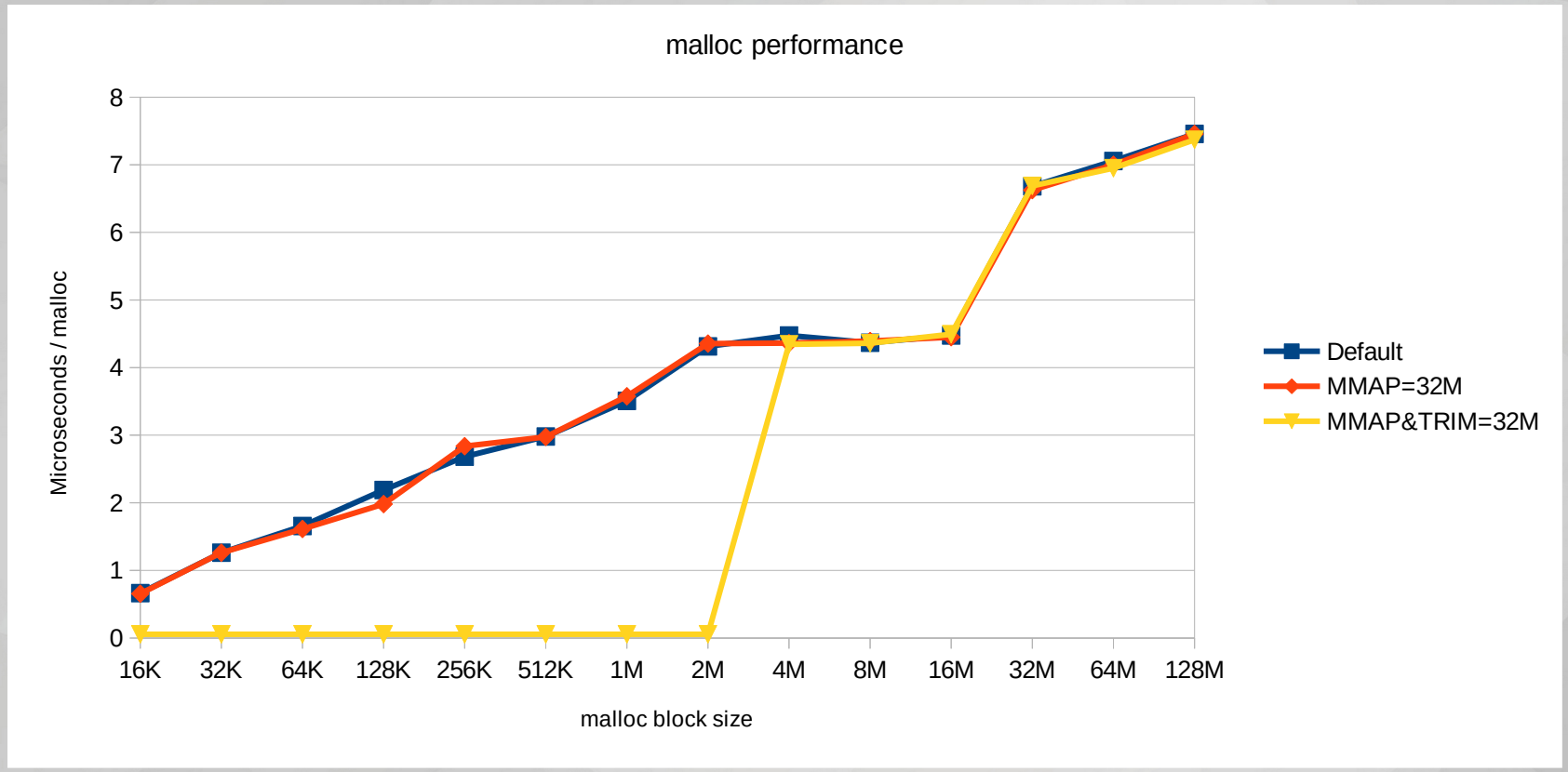
In later slides, we'll use short names for environment variables:
MMAP == MALLOC_MMAP_THRESHOLD_
TRIM == MALLOC_TRIM_THRESHOLD_

Increase MMAP=32M



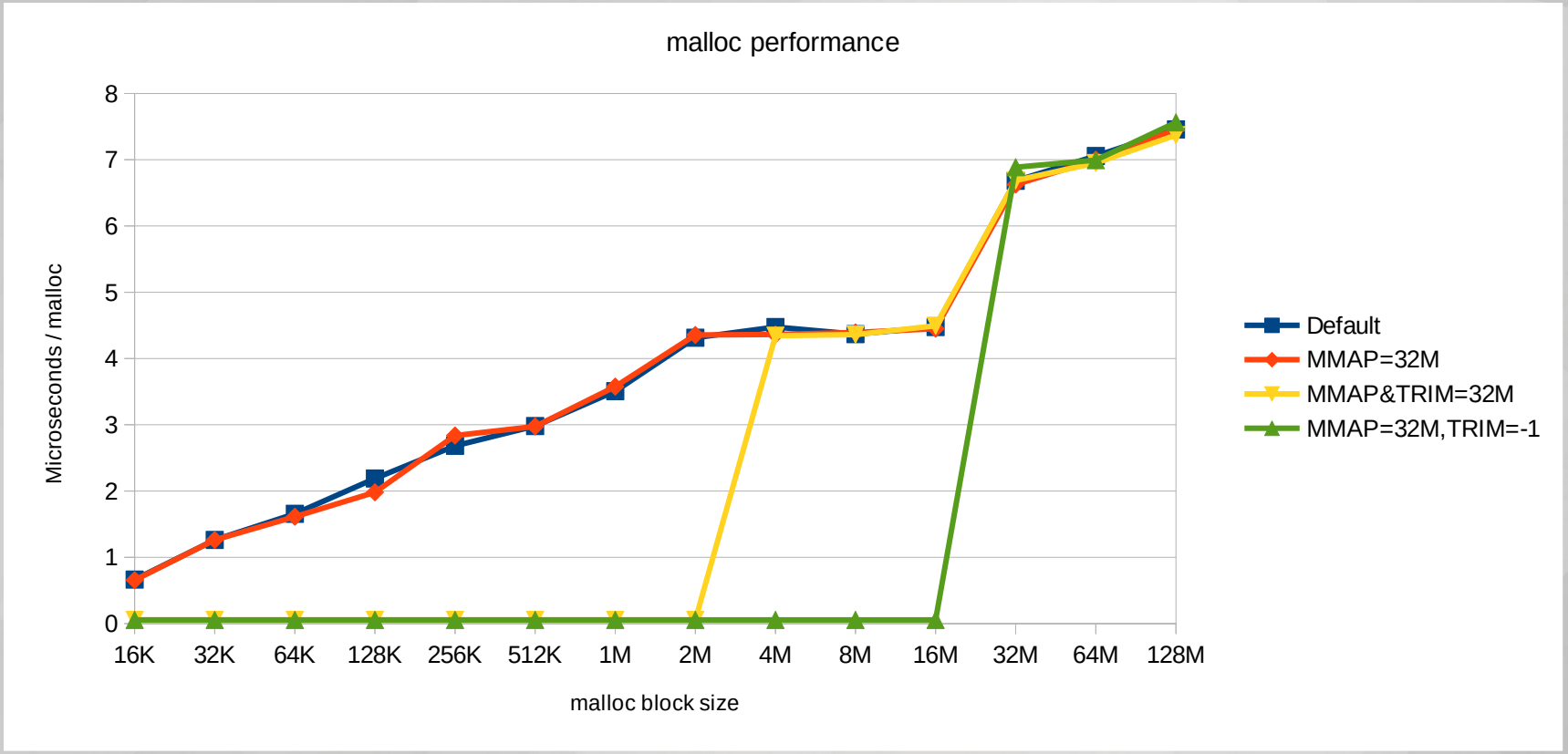
Lower lines are faster.

Set MMAP and TRIM to 32M



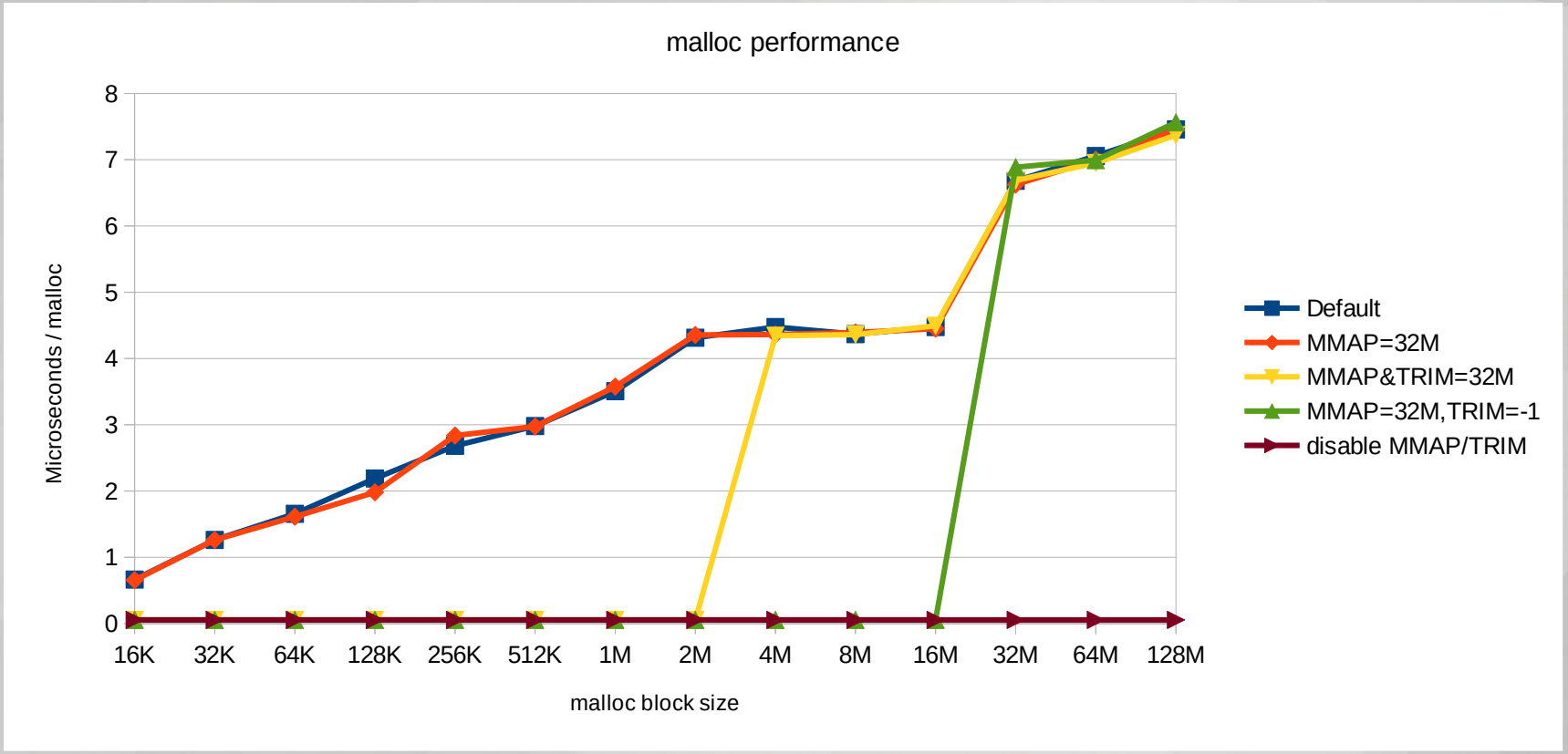
Lower lines are faster.

MMAP=32M, Disable TRIM



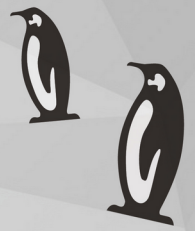
Lower lines are faster.

Disable MMAP and TRIM



Lower lines are faster.

Review Findings/Discussion



Appropriate setting of MALLOC_MMAP_MAX_,
MALLOC_MMAP_THRESHOLD_,
MALLOC_TRIM_THRESHOLD_
to match application needs can substantially improve malloc performance.

Current limit of maximum allowed MALLOC_MMAP_THRESHOLD_
was set in 2006. Much too small for some applications today.

Investigating revised maximum with minimal effects on apps using defaults.