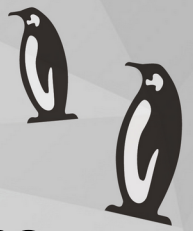
Optimize for Multi-core or Single core?

By Patrick McGehearty

LINUX
PLUMBERS
CONFERENCE > September 20-24, 2021

Modern Cache Sharing



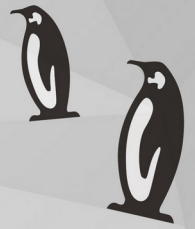
Modern chips with multiple cores often share cache across cores. Examples:

Intel – Ice Lake Xeon Platinum 8380 (40 cores, 80 threads)
60MB L3 Cache shared by all threads (0.75MB/thread)

AMD – EPYC 7702P (64 cores, 128 threads)
256MB L3 cache per chip, 16MB/CCX or 2MB per thread

It is common for server class chips with many processors to share L3 cache across some or all processors.

X86 non-temporal stores

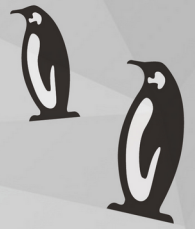


“Non-temporal” store: Hint to HW – data to be stored is not expected to be read again soon. Also called “streaming” stores. HW may choose to bypass cache when a series of stores would fill a cache line.

Benefits: Avoids pushing more useful data out of cache.
May reduce memory traffic (cache line need not be fetched).

Drawbacks: May reduce opportunities for cache reuse when
Very large caches are available.

Performance Issues



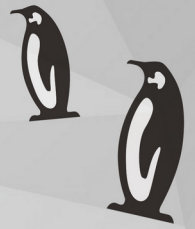
A optimization patch to glibc memcpy was tuned to provide maximum improvement for single thread performance on an eight core system by increasing the threshold for non-temporal stores.

It was later observed that the new threshold prevented the use of non-temporal stores in the highly parallel Stream benchmark, causing a 20% performance regression when run on a 128 core system.

In other studies, large copies clearing shared cache can cost 20% performance in pointer-chasing applications (i.e. linked-list management)

Reduction is performance is most likely when system is at highest load.

Decision: per core sharing



Old threshold: 6 * 1 core's share of L3 cache
(based on early 8 cores/chip assumptions)

Changed threshold: 3/4 total L3 cache
(based on tuning for single core)

New threshold: 3/4 * 1 core's share of L3 cache
(based on tuning for all cores active)

Current glibc Philosophy:
Activity on one thread should not cause significant negative effects on other threads.