

Re: tipc_init(), WARNING: at arch/x86/mm/highmem_32.c:52, [2.6.24-rc4-git5: Reported regressions from 2.6.23]

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2007-12/msg04901.html>

- *From:* Eric Dumazet <dada1@xxxxxxxxxxxxxx>
 - *Date:* Fri, 14 Dec 2007 08:16:34 +0100
-

Christoph Lameter a écrit :

On Sat, 8 Dec 2007, Ingo Molnar wrote:

Good. Although we should perhaps look at that reported performance problem with SLUB. It looks like SLUB will do a memclear() for the area twice (first for the whole page, then for the thing it allocated) for the slow case. Maybe that exacerbates the problem.

i dont think the SLUB problem could be explained purely via a double memset(). [which ought to be extremely fast anyway] We are talking about a 10 times slowdown on a 64-way box of a workload that is fairly common-sense. (tasks sending messages to each other via bog standard means)

while i dont want to jump to conclusions without looking at some profiles, i think the SLUB performance regression is indicative of the following fallacy: "SLAB can be done significantly simpler while keeping the same performance".

Well this is double crap. First of all SLUB does not do memclear twice. There is no reason to assume that SLUB has the problem just because SLOB hat that. A "fix" for that nonexistent problem went into Linus tree. WTH is going on?

SLUB was done because of a series of problem with the basic concepts of SLAB that treaten it usability in the future.

I couldnt point to any particular aspect of SLAB that i could characterise as "needless bloat".

I agree, SLABs architecture is pretty tight and I was one of those who helped it along to be that way.

However, SLAB is just fundamentally wrong for today's machine. The key problem today is cacheline fetch latency and that problem will increase significantly in the future. Sure under some circumstances that exploit the fact that SLAB sometimes gets its guesses on the CPU cache right SLAB can still win but the more processors and nodes we get the more it will become difficult to keep SLAB around and the more it will become difficult to establish what cachelines are in the CPU cache.

I think we should we make SLAB the default for v2.6.24 ...

If you guarantee that all the regression of SLAB vs. SLUB are addressed then that's fine but AFAICT that is not possible.

Here is a list of some of the benefits of SLUB just in case we forgot:

- SLUB is performance wise much faster than SLAB. This can be more than a factor of 10 (case of concurrent allocations / frees on multiple processors). See <http://lkml.org/lkml/2007/10/27/245>
- Single threaded allocation speed is up to double that of SLAB
- Remote freeing of objects in a NUMA system is typically 30% faster.
- Debugging on SLAB is difficult. Requires recompile of the kernel and the resulting output is difficult to interpret. SLUB can apply debugging options to a subset of the slab caches in order to allow the system to work with maximum speed. This is necessary to detect difficult to reproduce race conditions.
- SLAB can capture huge amounts of memory in its queues. The problem gets worse the more processors and NUMA nodes are in the system. The amount of memory limits the number of per CPU objects one can configure.
- SLAB requires a pass through all slab caches every 2 seconds to expire objects. This is a problem both for realtime and MPI jobs that cannot take such a processor outage.
- SLAB does not have a sophisticated slabinfo tool to report the state of slab objects on the system. Can provide details of object use.
- SLAB requires the update of two words for freeing and allocation. SLUB can do that by updating a single word which allows to avoid enabling and disabling interrupts if the processor supports an atomic instruction for that purpose. This is important for realtime kernels where special measures may have to be implemented if one wants to disable interrupts.

Re: tipc_init(), WARNING: at arch/x86/mm/highmem_32.c:52, [2.6.24-rc4-git5: Reported regressions from 2.6.23]

- SLAB requires memory to be set aside for queues (processors times number of slabs times queue size). SLUB requires none of that.
- SLUB merges slab caches with similar characteristics to reduce the memory footprint even further.
- SLAB performs object level NUMA management which creates a complex allocator complexity. SLUB manages NUMA on the level of slab pages reducing object management overhead.
- SLUB allows remote node defragmentation to avoid the buildup of large partial lists on a single node.
- SLUB can actively reduce the fragmentation of slabs through slab cache specific callbacks (not merged yet)
- SLUB has resiliency features that allow it to isolate a problem object and continue after diagnostics have been performed.
- SLUB creates rarely used DMA caches on demand instead of creating them all on bootup (SLAB).

Yes, SLUB should be the way to go, but some issues are not yet solved.

I had to switch back to SLAB on a production NUMA server, with 2 nodes and 8GB ram. Using a lot of sockets, so a large part of memory was used by kernel.

SLUB kernel was hitting OOM after 2 or 3 days of uptime.
SLAB kernel never hit this.

Unfortunately I don't have a test machine to reproduce the setup.

Maybe the problem is not related to SLUB at all, but an underlying VM/NUMA bug.

The /proc/buddyinfo showed that :

Node 0 contained two zones (DMA and DMA32) total 4 GB
Node 1 contained one zone (Normal) total 4 GB

So Node 0 contained no (Normal) zone

part of /proc/meminfo

Slab: 3338512 kB
SReclaimable: 789716 kB
SUnreclaim: 2548796 kB

I remember network interrupts were taken by CPU 1, so most allocations were done by CPU 1 (node 1), and many freeing were done on CPU 0

Re: tipc_init(), WARNING: at arch/x86/mm/highmem_32.c:52, [2.6.24-rc4-git5: Reported regressions from 2.6.23]

Re: tipc_init(), WARNING: at arch/x86/mm/highmem_32.c:52, [2.6.24-rc4-git5: Reported regressions from 2.6.23]

Hope this helps

--

To unsubscribe from this list: send the line "unsubscribe linux-kernel" in the body of a message to majordomo@xxxxxxxxxxxxxxxxx

More majordomo info at <http://vger.kernel.org/majordomo-info.html>

Please read the FAQ at <http://www.tux.org/lkml/>

Re: tipc_init(), WARNING: at arch/x86/mm/highmem_32.c:52, [2.6.24-rc4-git5: Reported regressions from 2.6.23]