

## Re: [00/17] Large Blocksize Support V3

---

*Source:* <http://linux.derkeiler.com/Mailing-Lists/Kernel/2007-04/msg09713.html>

---

- *From:* Nick Piggin <[nickpiggin@xxxxxxxxxxxxx](mailto:nickpiggin@xxxxxxxxxxxxx)>
  - *Date:* Fri, 27 Apr 2007 01:08:17 +1000
- 

Andy Whitcroft wrote:

Nick Piggin wrote:

I don't understand what you mean at all. A block has always been a contiguous area of disk.

Lets take Nick's definition of block being a disk based unit for the moment. That does not change the key contention here, that even with hardware specifically designed to handle 4k pages that hardware handles larger contiguous areas more efficiently. David Chinner gives us figures showing major overall throughput improvements from (I assume) shorter scatter gather lists and better tag utilisation. I am loath to say we can just blame the hardware vendors for poor design.

So their controllers get double the throughput when going from 512K (128x4K pages) to 2MB (128x16K pages) requests. Do you really think it is to do with command processing overhead?

Actually, I don't know why people are so excited about being able to use higher order allocations (I would rather be more excited about never having to use them). But for those few places that really need it, I'd rather see them use a virtually mapped kernel with proper defragmentation rather than putting hacks all through the core code.

Virtually mapping the kernel was considered pretty seriously around the time SPARSEMEM was being developed. However, that leads to a non-constant relation for converting kernel virtual addresses to physical ones which leads to significant complexity, not to mention

Re: [00/17] Large Blocksize Support V3

runtime overhead.

Yeah, a page table walk (or better, a TLB hit). And yeah it will cost a bit of performance, it always does.

As a solution to the problem of supplying large pages from the allocator it seems somewhat unsatisfactory. If no significant other changes are made in support of large allocations, the process of defragmenting becomes very expensive. Requiring a stop\_machine style hiatus while the physical copy and replace occurs for any kernel backed memory.

That would be a stupid thing to do though. All you need to do (after you keep DMA away) is to unmap the pages.

To put it a different way, even with such a full defragmentation scheme available some sort of avoidance scheme would be highly desirable to avoid using the very expensive defragmentation underlying it.

Maybe. That doesn't change the fact that avoidance isn't a complete solution by itself.

Is that a big problem? Really? You use 16K pages on your IPF systems, don't you?

To my knowledge, moving to a higher base page size has its advantages in TLB reach, but brings with it some pretty serious downsides. Especially in caching small files. Internal fragmentation in the page cache significantly affecting system performance. So much so that development is ongoing to see if supporting sub-base-page objects in the buffer cache could be beneficial.

I think 16K would be pretty reasonable (ia64 tends to use it). I guess powerpc went to 64k either because that's what databases want or because their TLB refills are too slow, so the internal fragmentation bites them a lot harder.

But that was more of a side comment, because I still think io controllers should be easily capable of operation on 4K pages. Graphics cards are, aren't they?

Re: [00/17] Large Blocksize Support V3

---

SUSE Labs, Novell Inc.

-

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

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

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