

sparsemask

How to get around the topology nuances?

K Prateek Nayak <kprateek.nayak@amd.com>



Acknowledgement

This work is derived from former efforts to implement sparsemask by:

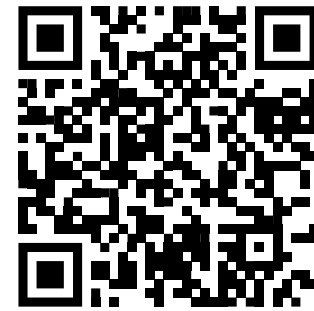
- **Steve Sistare** <steven.sistare@oracle.com>
- **Peter Zijlstra** <peterz@infradead.org>
- **Chen Yu** <yu.c.chen@intel.com>
- **Tim Chen** <tim.c.chen@linux.intel.com>



[Steve's original
implementation](#)



[Peter's version
for x86](#)



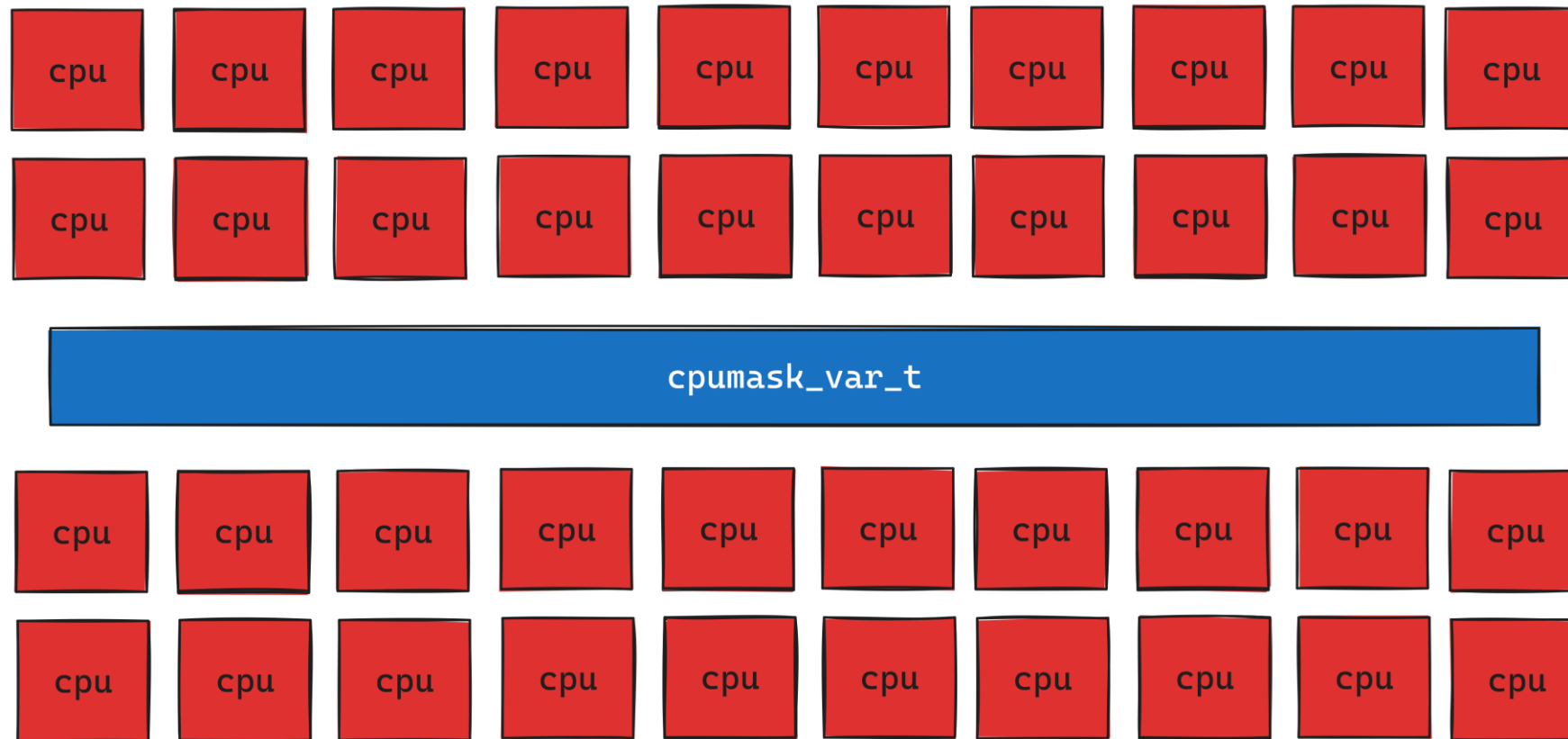
[Extending Peter's
implementation for
all architectures](#)



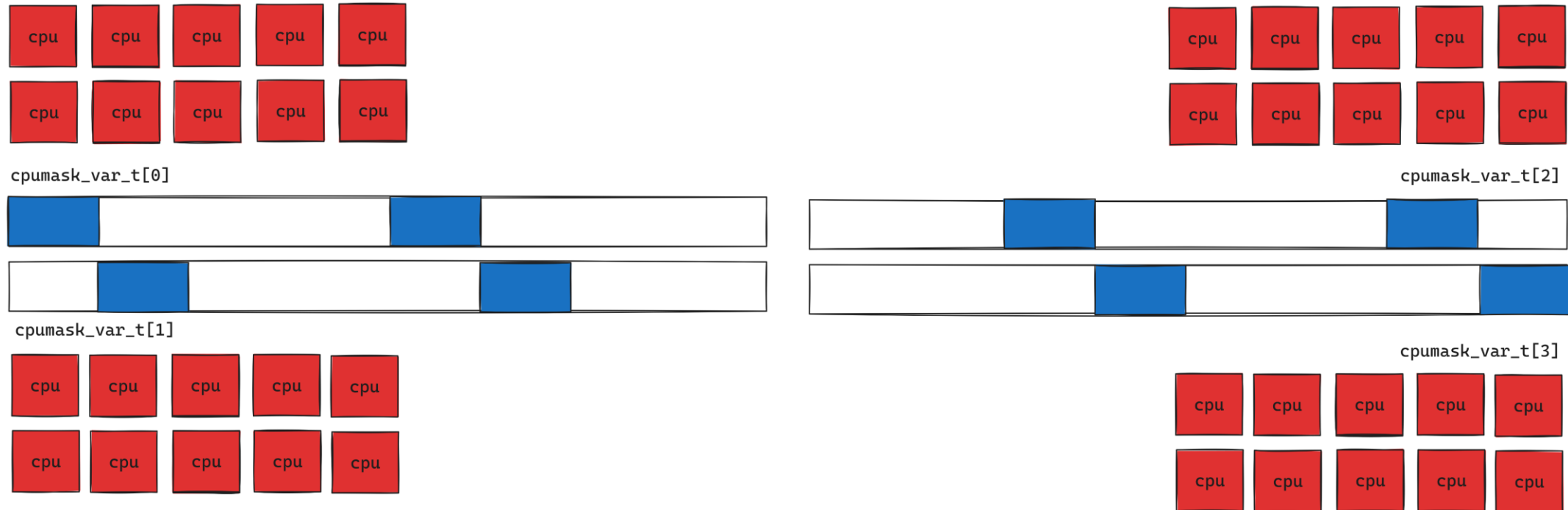
LINUX
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT 5-7, 2026

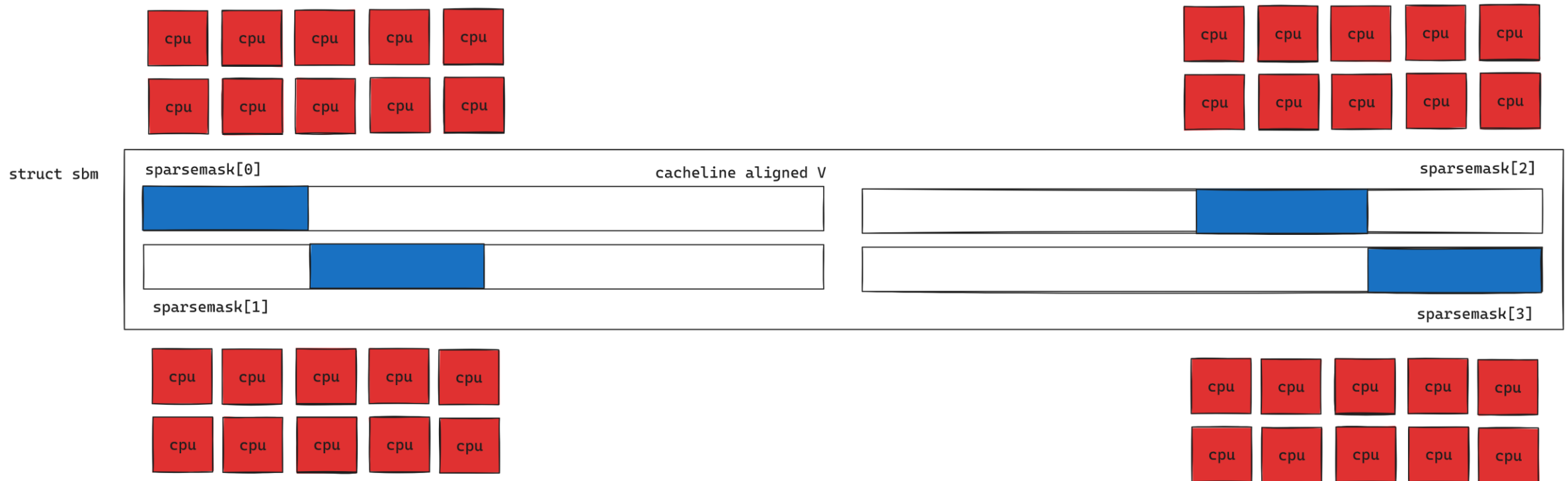
Challenge: Global cpumask scalability



Solution(1): Maintain one cpumask per LLC / NUMA domain



Solution(2): sparsemask (or sparsebitmap)



sbm: Is that a sbitmap I spot?

```
struct sbitmap_word {  
    unsigned long word;  
    unsigned long cleared ____cacheline_aligned;  
    raw_spinlock_t swap_lock;  
} ____cacheline_aligned_in_smp;
```

```
struct sbitmap {  
    unsigned int depth;  
    unsigned int shift;  
    unsigned int map_nr;  
    bool round_robin;  
    struct sbitmap_word *map;  
    unsigned int __percpu *alloc_hint;  
};
```

Reduced to

```
struct sbm_leaf {  
    unsigned long bitmap;  
} ____cacheline_aligned;  
  
struct sbm {  
    struct sbm_leaf leaf[SBM_SIZE];  
};
```



sbm: Determining SBM_SIZE at boot

- **CONFIG_NUMA**

- arm64
- loongarch
- mips
- powerpc
- riscv
- s390
- sh
- sparc
- x86

- **ARCH_SUPPORTS_SCHED_MC**

- arm
- arm64
- loongarch
- parisc
- powerpc
- riscv
- s390
- sparc
- x86

- **CONFIG_HOTPLUG_CPU**

- arm
- arm64
- csky
- loongarch
- mips
- parisc
- powerpc
- riscv
- s390
- sh
- sparc
- x86
- xtensa



Topology: PowerPC pSeries

Snippet from `pseries_add_processor()` from [arch/powerpc/platforms/pseries/hotplug-cpu.c](https://elixir.bootlin.com/linux/v6.12.10/source/arch/powerpc/platforms/pseries/hotplug-cpu.c):

```
rc = find_cpu_id_range(nthreads, node, &cpu_mask);
if (rc && nr_node_ids > 1) {
    /* Try again, considering the free CPU ids from the other node. */
    node = NUMA_NO_NODE;
    rc = find_cpu_id_range(nthreads, NUMA_NO_NODE, &cpu_mask);
}
...
set_hard_smp_processor_id(cpu, be32_to_cpu(*intserv++));

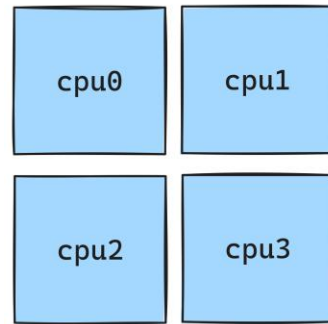
...

/* If node is set to NUMA_NO_NODE, CPU ids have be reused from
 * another node, remove them from its mask.
 */
if (node == NUMA_NO_NODE) {
    cpu = cpumask_first(cpu_mask);
    pr_warn("Reusing free CPU ids %d-%d from another node\n", cpu, cpu + nthreads - 1);
    ...
}
```



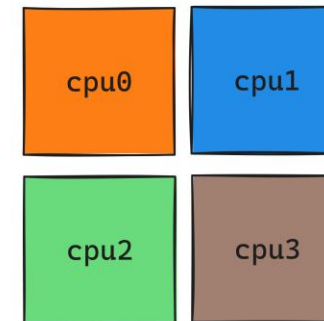
Topology: PowerPC pSeries

 NUMA Nodes



SBM_SIZE = 1
CPUs per sbm leaf = 4

Few hotplugs later



SBM_SIZE = 4
CPUs per sbm leaf = 1



**LINUX
PLUMBERS CONFERENCE**

PRAGUE, CZECHIA / OCT 5-7, 2026

Image created using <https://excalidraw.com/>

Topology: GENERIC_ARCH_TOPOLOGY

`cpu_coregroup_mask()` from [drivers/base/arch_topology.c](#):

```
const struct cpumask *cpu_coregroup_mask(int cpu)
{
    const cpumask_t *core_mask = cpumask_of_node(cpu_to_node(cpu));

    /* Find the smaller of NUMA, core or LLC siblings */
    if (cpumask_subset(&cpu_topology[cpu].core_sibling, core_mask)) {
        /* not numa in package, lets use the package siblings */
        core_mask = &cpu_topology[cpu].core_sibling;
    }

    if (last_level_cache_is_valid(cpu)) {
        if (cpumask_subset(&cpu_topology[cpu].llc_sibling, core_mask))
            core_mask = &cpu_topology[cpu].llc_sibling;
    }

    if (IS_ENABLED(CONFIG_SCHED_CLUSTER) &&
        cpumask_subset(core_mask, &cpu_topology[cpu].cluster_sibling))
        core_mask = &cpu_topology[cpu].cluster_sibling;

    return core_mask;
}
```



Topology: Disabled APIC (loongarch, x86)

Snippet from `acpi_numa_x2apic_affinity_init()` from [arch/loongarch/kernel/acpi.c](https://arch.loongarch/kernel/acpi.c):
(Same is also true for x86)

```
void __init
acpi_numa_x2apic_affinity_init(struct acpi_srat_x2apic_cpu_affinity *pa)
{
    int pxm, node;

    if (srat_disabled())
        return;

    if (pa->header.length < sizeof(struct acpi_srat_x2apic_cpu_affinity)) {
        bad_srat();
        return;
    }

    if ((pa->flags & ACPI_SRAT_CPU_ENABLED) == 0)
        return;

    ...

    set_cpuid_to_node(pa->apic_id, node);

    ...
}
```



Topology: Fallback for disabled CPUs

Potential fallback mechanism (used for PowerPC):

```
/* Assume disabled CPUs can form their own singleton node. */  
sbm_num_instances += disabled_cpus;
```

```
/* Assume all disabled CPUs can show up on instance with max CPUs. */  
sbm_max_cpus_per_instance += disabled_cpus;
```



Topology: x86: Guests + CPU Hotplug

New CPUs can be hot-plugged on demand in guests as long as they are set in `cpus_possible_mask`.

On QEMU + KVM, the hotplug of vCPU follows:

- MADT parses the disabled CPU and does `set_cpu_possible(cpu, true)`.
- Guest gets an ACPI notification during hotplug.
- ACPI properties are evaluated again and CPU is onlined.

For the guest OS:

- Correct APICIDs are present since the guest OS boot for non-present CPUs too.
- Topology information derived from them is consistent with the view post hotplug.

Any concerns with other VMMs?

Any concerns for other architecture that may support hotplug?



sbm for wakeup: Data formats

- Using first 8-bytes of the sbm leaf
 - Atomic updates to set and clear bits
 - Single u64 read can be used to traverse 64 CPUs
- Spreading the CPUs at byte boundary
 - No atomic operations – single byte store to set / clear CPU.
 - Optimizations from hardware for non-atomic stores on the same cacheline.
 - Single u64 read only traverses 8 CPUs.
 - Requires a gather step to construct the full mask.

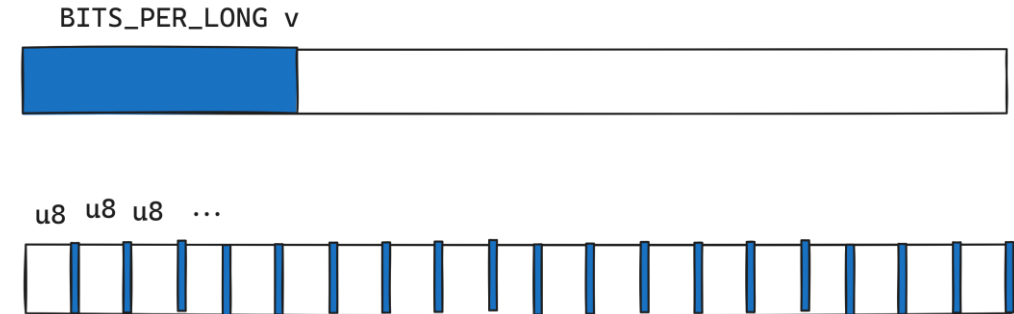


Image created using <https://excalidraw.com/>

Variant	%cycles vs global mask
Per-NUMA mask	32.9209% (3x faster)
Per-LLC mask	1.2977% (77x faster)
Per-LLC mask (u8 operations)	0.4930% (202x faster)

Table: Worst case latency analysis for bitmask operations

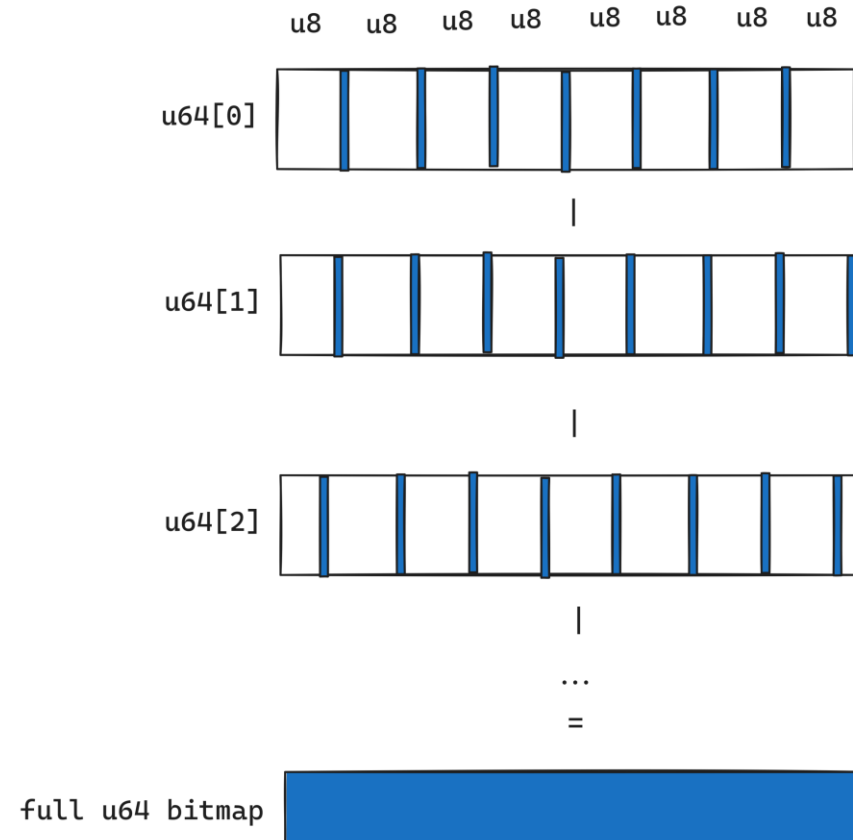
Source: <https://lore.kernel.org/lkml/e093d930-79df-4285-a492-cc6d40b3cd51@amd.com/>



LINUX
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT 5-7, 2026

sbm for wakeup: Constructing the mask



Will this be a problem for Big Endian systems?



LINUX
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT 5-7, 2026

Image created using <https://excalidraw.com/>



LINUX PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT 5-7, 2026

Děkuju (Thank you)

To everyone hacking, reviewing, testing, and maintaining the Linux kernel

Special thanks to the Maintainers, Reviewers, and Contributors of the Linux scheduler, and everyone here for putting up with my crazy ideas (^_^)