Problem: select_idle_cpu() not scalable

- SIS_PROP not work as expected
  - Idle core scan not throttled at all
  - Idle CPU scan not throttled very well

- Select_idle_cpu() scale poorly, searching up to max number of LLC CPUs frequently

- Hardware: 4 nodes, 96 cores, 192 CPUs (24core/48HT in one LLC domain)
Proposal: Idle CPU Mask

- Track Idle CPUs per LLC domain
  - Bit Set every Idle entry
  - Bit Clear every scheduler tick if not idle (update ratelimited)

- Task wakeup path very sensitive to change
  - Scan efficiency improved but performance not universal win

Kernel: V5.14 V.S. Idle_CPU_mask V10
Cluster Topology Level

• Hardware Topology
  • ARM64 Kunpeng920 32/24 cores share LLC, each 4 cores of them share L3 tag/internal bus
  • X86 Jacobsville has 24 cores sharing LLC, but each 4 cores sharing L2

• Needs
  • Add scheduler level for cluster to support load balance between clusters to decrease resource contention and increase memory bandwidth
    ✓ SPECrate mcf has up to +25.1% on Jacobsville; + 13.574% on Kunpeng920
    ✓ stream has up to +19.85% on Kunpeng920
    ✓ Patch V1 sent after several RFCs, expecting review: https://lore.kernel.org/lkml/20210820013008.12881-1-21cnbao@gmail.com
  • Scan cluster before scanning LLC in wake_affine to leverage the lower communication latency within cluster
    ✓ much more tricky; RFC sent but formal patch not yet. Latest version: https://op-lists.linaro.org/pipermail/linaro-open-discussions/2021-June/000219.html
Scanning cluster first

- [https://op-lists.linaro.org/pipermail/linaro-open-discussions/2021-June/000221.html](https://op-lists.linaro.org/pipermail/linaro-open-discussions/2021-June/000221.html)

Prototype:

```c
static int select_idle_sibling(struct task_struct *p, int prev, int target) {
    ...
    if (sched_cluster_active()) {
        i = select_idle_cluster(p, cluster_sd, has_idle_core, target);
        if (((unsigned) i < nr_cpumask_bits)
            return i;
        +
        */
        * if prev and target are not in same LLC, give other cpus who have
        +
        +
        +
        +
        +
        */
        +
        if (cpus_share_cache(prev, target) & has_idle_core)  
            return target;
    ...
    i = select_idle_cpu(p, sd, has_idle_core, target);
    ...
}
```

For unpinned apps -> much more tricky

- ![Prototype](https://example.com/prototype.png)

- If scanning cluster has scanned 4 CPUs and spent some time, how to adjust `select_idle_cpu()` for scanning avg time and SIS accordingly?
- Seeing idle CPU even system is busy; seeing -2% performance on busy mysql; removing this "return" and always doing further scan can give positive performance on mysql
Prefer idle(r) cores to cache affinity

At task wakeup, Current scheduler

- Chooses CPU based on load of previous + waking CPU.
- Find an idle core or idle CPU (within chosen CPU LLC).
- On systems with lesser cores per LLC:
  - Maybe no idle cores in chosen LLC, idle cores in other LLC.
- Chosen LLC may have lower idle CPUs compared other LLC.
- Doesn't consider different cache latencies between LLCs within the socket.
  - Nearby LLC idler than the chosen LLC.

Proposed Solution : Idler LLC approach

- Maintain a list of idle cores per LLC.
- If waker and previous CPUs are from a different LLCs.
  - Choose a LLC which has idle core.
  - If no idle cores select a CPU whose LLC has more idle CPUs.
  - Else fallback to existing approach.

Fallback LLC Approach: (Archs that support different Cache latencies)

- Select an idle core within the parent sched-domain on the chosen LLC.
- If no idle cores in parent sched-domain, select a CPU whose LLC has more idle CPUs.
Searching idle cpu/core

- Looking for an idle CPU takes time
  - It impacts local running task
  - Delays task wake up

- Limit the time spent for searching an idle CPU
  - Don’t waste time searching a nonexistent idle cpu
  - At some point it’s better to simply wake up locally and let LB migrate task

- Using local avg idle is often misleading
  - Do not reflect other CPUs state but only reflect local cpu state

- Using local cpu and task load/utilization
  - Long running task vs missing short idle cpu
  - Short running task vs a lightly loaded local cpu