Packing, spreading and scheduling latency

20th september, 2013

V0.1
Content

- Spreading for performance
  - An example with cyclic test
  - Latency results
- Why such difference?
  - Topology of a system
  - Idle state and wake up latency
  - Back to our latency results
- How to select the right CPU?
- Is it enough?
- Packing tasks for saving power
  - Packing tasks for saving energy
  - Power CPU topology
- Create a list of packing CPUs
  - Example of packing policy
  - Updating the list
- Is it enough?
Spreading for performance

- Spreading tasks in the system
  - Default policy of the scheduler
  - Minimize resources sharing

- Often the best policy
  - Long running tasks
  - Memory / CPU intensive workloads

- But some use cases don't follow the rule
  - Shared resources are not the critical path
  - Light workload
An example with cyclictest

- Use various intervals in the range [1ms:2ms]
- Use both tasks placement policy
  - Spread tasks on CPUs (default behavior)
  - Pack tasks on 1 CPUs
- Enable/Disable C-state
- 10 runs per configuration
  - Get min/max/avg and stdev of the average latency of each run
Latency results

![Graph showing latency results with categories: spread, packed, shallow c-state.](image)
Why such difference ?
Topology of a system

- Have a look at the topology of a typical system
Idle state and wake up latency

- Generally, we can powergate/power down at all level:
  - Each core can be power gated independently
  - The cluster/package with/without the associated PLLs, power domains and regulators.
  - Nearly the complete system when all masters are off

- Wakeup latency increases with powered-down area
  - PLLs state
  - regulators state
  - peripherals state
Back to our latency results

- **Idle statistics for cyclictest -q -t 3 -i 1800-d 100 -e 1000000 -D 5**

<table>
<thead>
<tr>
<th></th>
<th>% cpu0</th>
<th>avg idle cpu0</th>
<th># cpu0</th>
<th>% cpu1</th>
<th>avg idle cpu1</th>
<th># cpu1</th>
<th>% cpu2</th>
<th>avg idle cpu2</th>
<th># cpu2</th>
</tr>
</thead>
<tbody>
<tr>
<td>C0</td>
<td>55</td>
<td>0.53</td>
<td>5315</td>
<td>0</td>
<td>0.4</td>
<td>73</td>
<td>0</td>
<td>0.4</td>
<td>32</td>
</tr>
<tr>
<td>C1</td>
<td>40</td>
<td>1.12</td>
<td>1787</td>
<td>98</td>
<td>2.02</td>
<td>2427</td>
<td>98</td>
<td>1.92</td>
<td>2564</td>
</tr>
</tbody>
</table>

- **Spread**

- **Packed**

<table>
<thead>
<tr>
<th></th>
<th>% cpu0</th>
<th>avg idle cpu0</th>
<th># cpu0</th>
<th>% cpu1</th>
<th>avg idle cpu1</th>
<th># cpu1</th>
<th>% cpu2</th>
<th>avg idle cpu2</th>
<th># cpu2</th>
</tr>
</thead>
<tbody>
<tr>
<td>C0</td>
<td>30</td>
<td>0,36</td>
<td>4125</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>C1</td>
<td>62</td>
<td>1,57</td>
<td>1974</td>
<td>100</td>
<td>0</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
How to select the right CPU?

- Take wake up latency of core into account
  - Select the idle CPU with shortest latency in the LLC
  - Compare runnable_avg of the task with the cost of wake up latency

- weighted_cpuload
  - Used to compare the load of CPUs
  - All idle CPUs have a null load

- Modify the weighted_cpuload of Idle CPUs
  - No more null but reflect the effort to wake it up

- Don't choose the 1st idle CPU
  - Use the weighted_cpuload for selecting an idle CPU
Is it enough?
Packing tasks for saving power

- Scheduler knows when CPUs share
  - Core capacities
  - Resources like the cache

- But it doesn't know their power dependency

- Packing makes sense only if there is a gain in
  - Latency as seen previously
  - Power consumption by increasing powered down area
Packing tasks for saving energy

- Optimizing the power down area
- With minimal increase of the running time
Power CPU topology

- Add a new flag in sched_domain: `SD_SHARE_POWERDOMAIN`
  - Domain member shares their power down capabilities

- Let architecture describes their topology
  - New function `arch_sd_local_flags (cpu, flag)`
  - Return per-cpu power dependency in a domain
  - Used during the init of sched_domain's levels

- Use DT to describe power dependency
  - Add a new property for CPU topology description
    - power-gate= <0/1>
Create a list of packing CPUs

- Pack tasks only if you can power down/gate the forced idle CPUs
  - Use all cores that share their power state

- Use group of CPUs with lowest capacity first
  - Assuming they are the most power efficient

- Use 1st CPU in the mask
  - Default policy in the scheduler
Example of packing policy

- Full sharing of power state (default configuration)
Example of packing policy

- Can power gate cluster independently

CPU0-1,4-5 ➔ CPU2-3,6-7
Example of packing policy

- Each core can be power gated independently

CPU0,4 → CPU1,5 → CPU2,6 → CPU3,7
Updating the list

- Periodically evaluate system activity
  - Use runnable_avg_sum/period of CPUs
  - Use CPU's capacity (cpu_power)
  - Sync activity monitoring with load balance

- Deduct how many CPUs needed
  - Use CPUs' capacity (cpu_power)

- Then ensure that a target CPUs is in this list
  - check CPU selection at wake up
  - Define a buddy CPU that will handle the activity of non packing CPU
  - New task can use a CPU out of the list
Is it enough?
Questions ?