Turbo-Sched

A scheduler for sustaining Turbo Frequencies for longer duration

Parth Shah
<parth@linux.ibm.com>
IBM India Pvt Ltd

Outline

- Turbo Frequencies
  - Problem with Power budget
  - How to sustain?
- CFS design
  - Problem with task spreading
- TurboSched Algorithm
- Results
- Future plans

Turbo Frequencies

- Many SMP servers have support for P-states in the range of Turbo Frequencies
- These P-states burn more power
- Power budget is fixed for a chip
- Systems may be **throttled to a lower frequency** if power consumption exceeds this power budget
- This makes difficult to sustain Turbo Frequencies for longer duration

How to sustain?

CPU-IDLE

- CPU-Idle allows system to goto Idle/C-states in absence of any workload
- Deeper the C-states level, more the power savings
- Advantageous for sustaining Turbo frequencies
- Keeping fewer CPUs idle saves power which can be **channeled to busier CPUs**
- *TurboSched Policy*: keep maximum possible CPUs idle

CFS: Why cannot sustain Turbo?

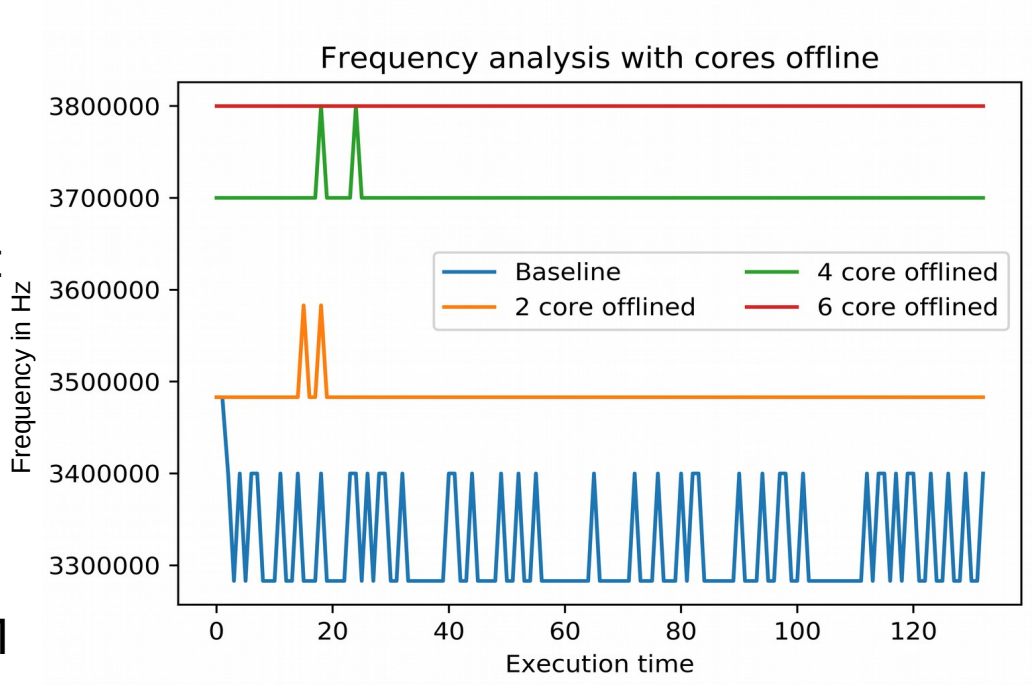
Task Spreading

- Base of CFS for SMP systems is to spread tasks as much as possible to effectively utilize system resources
- Pros:
  - Better task wake-up/turn around time
  - Better cache resource utilization
  - Better memory throughput
- Cons:
  - Wake up task on idle CPU if any available

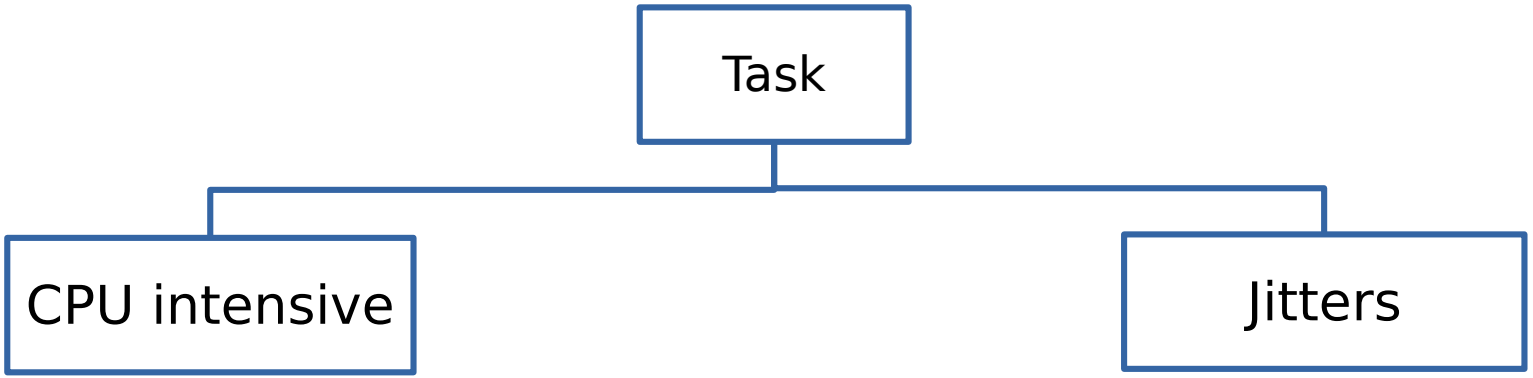
Adapting CFS for Turbo

Task Packing

- Keep fewer CPUs idle
- This allows us to over-clock few cores thereby remaining within power budget
- Experimentation
  - Hot-Plugging out few cores have shown to sustain turbo frequencies
  - Results shows Max frequency available at the given time on a IBM POWER9 system



Task classification



- Usually CPU bound important tasks
- Memory/cache hungry tasks
- Performance oriented
- ***CFS is the best algorithm***

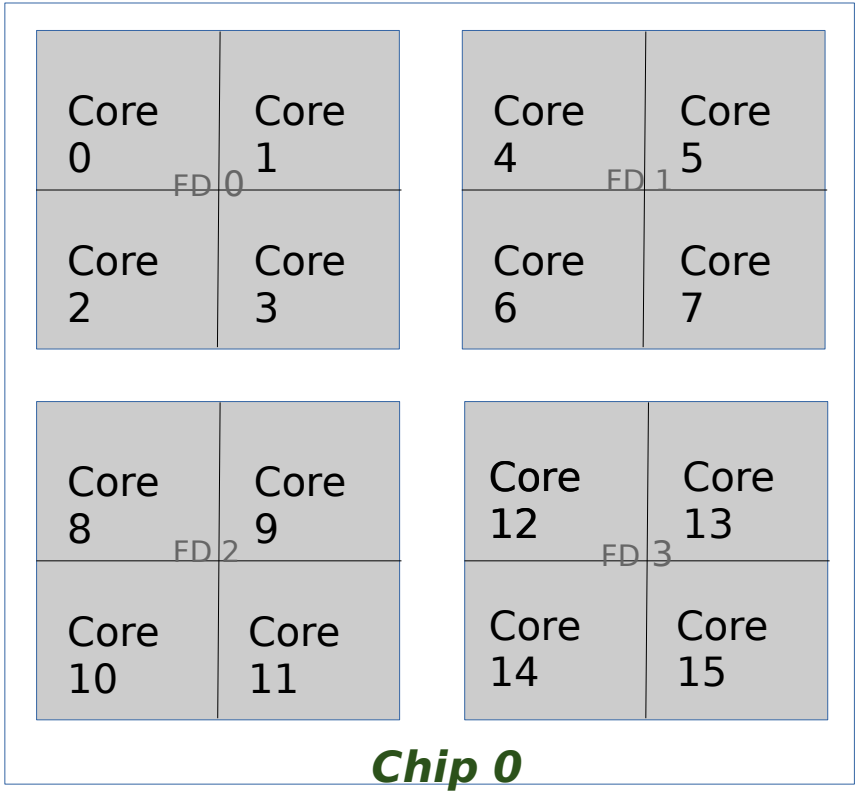
- Usually system daemons/house keeping task
- Low utilization and/or bursty workload
- ***Sleeps before load balance gets invoked***
- ***Can be packed to Intensive CPUs***
- ***CFS is not a good choice***

Frequency domain (FD)

- Many SMP systems also support *SMT or Hyper-threading modes*
- A core can have multiple threads(called CPUs) depending on SMT mode
- Frequency controlling is **per core** in such systems
- Other way around, providing *more frequency to a CPU in a core leads to more frequency for all the threads* in the core
- But, *IDLE domain is per core*
- So, **Task packing needs to be done across cores**
- TurboSched Policy: Keep maximum possible **Cores** Idle

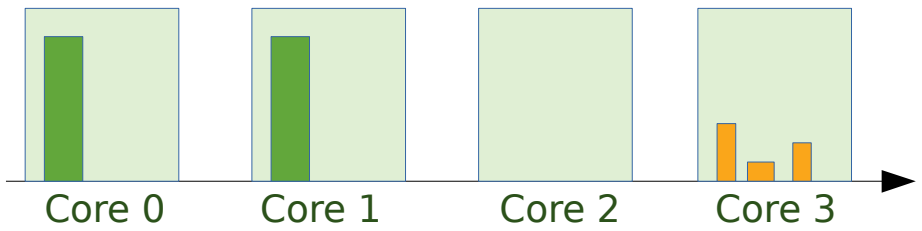
Experimental Setup

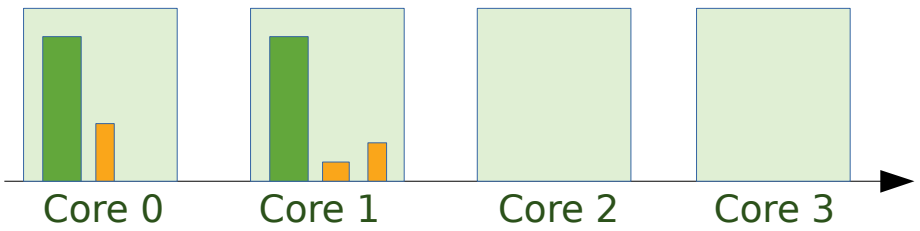
- IBM Power 9 system:
  - Arch: POWERPC
  - Cores: 16
  - SMT: 4 per core
  - Rated frequencies: 2.1 – 3.2 GHz
  - Turbo freq: 3.2 - 3.8 GHz (18%)
  - FD: Set of 4 Cores
- Workload:
  - Intensive: Integer ops, MIPS scales with freq
  - Jitter: Timed Int ops, frequency invariant
  - Sibling thread regression= ~4%



Ways to pack tasks

- **Isolate**
  - ➔ Create a dedicated core to pin all jitters there
  - ➔ Intensive tasks uses CFS spreading policy
- **Side by side packing**
  - ➔ Keep jitters on near by threads where Intensive tasks are running
  - ➔ Use CFS when no such tasks are found



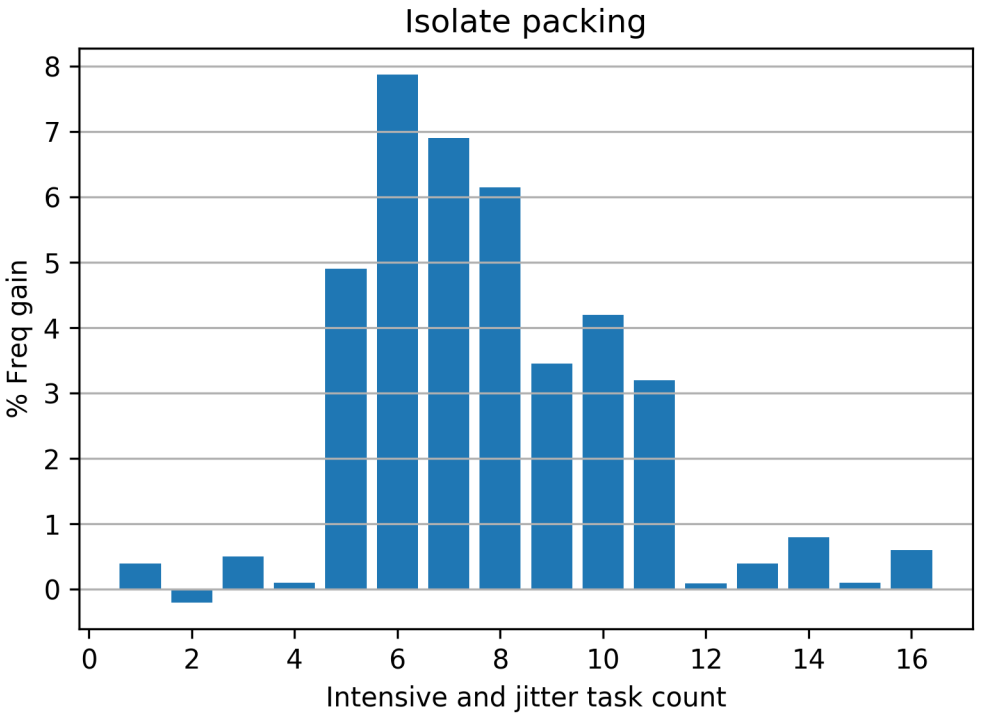


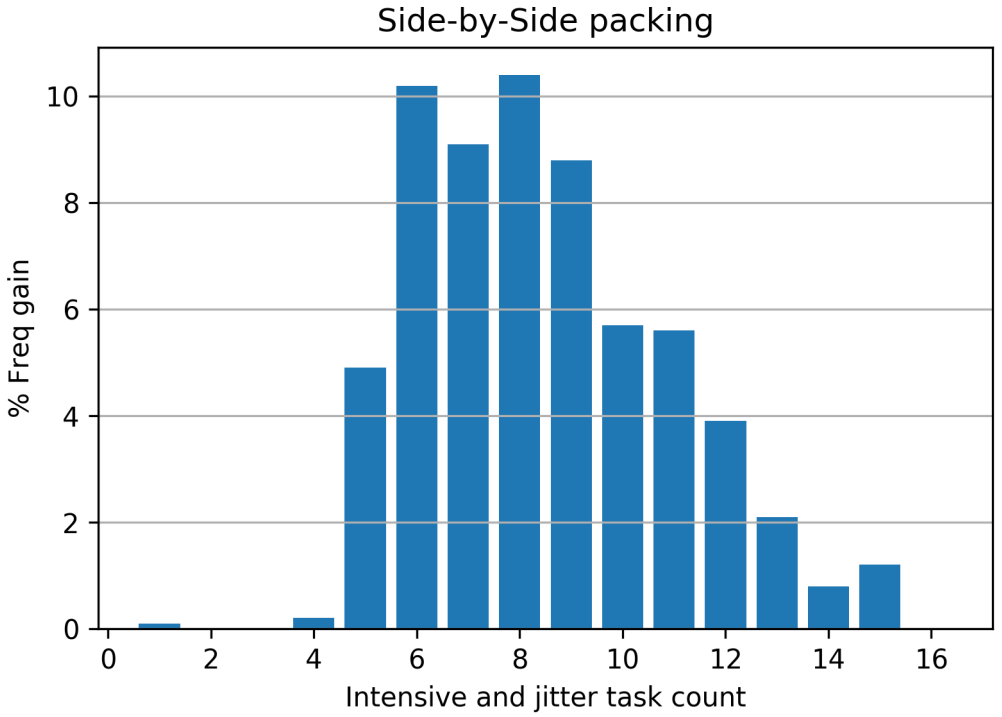
Intensive task utilization

Jitter task utilization

Isolation vs Side by side packing

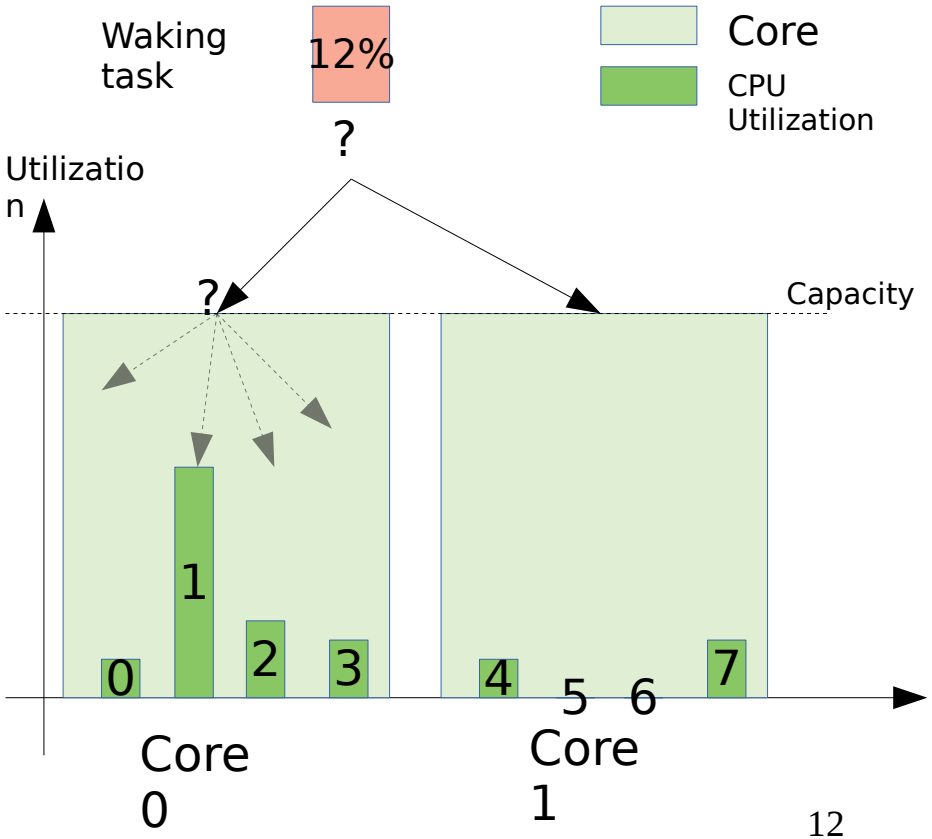
- Frequency advantage w.r.t. CFS





Task wake-up logic

- Given waking task utilization, where it should be placed?
- Given scenario:
  - CFS may pick CPU 5/6
- TurboSched:
  - Optimize for ***Performance/Watts***
  - ***Policy:*** Find **first-fit core** and **least util CPU** to wake a task
  - Iterative scan on DIE/LLC domain
  - Don’t select cores with util <12.5%



TurboSched Tipping Point

- What is ***core occupation capacity?***
- On SMT systems, the capacity of core scales linearly w.r.t. the online threads(SMT Mode).
- So,

  Core capacity= (1 +SMT-mode/8) * capacity_of(any CPU)

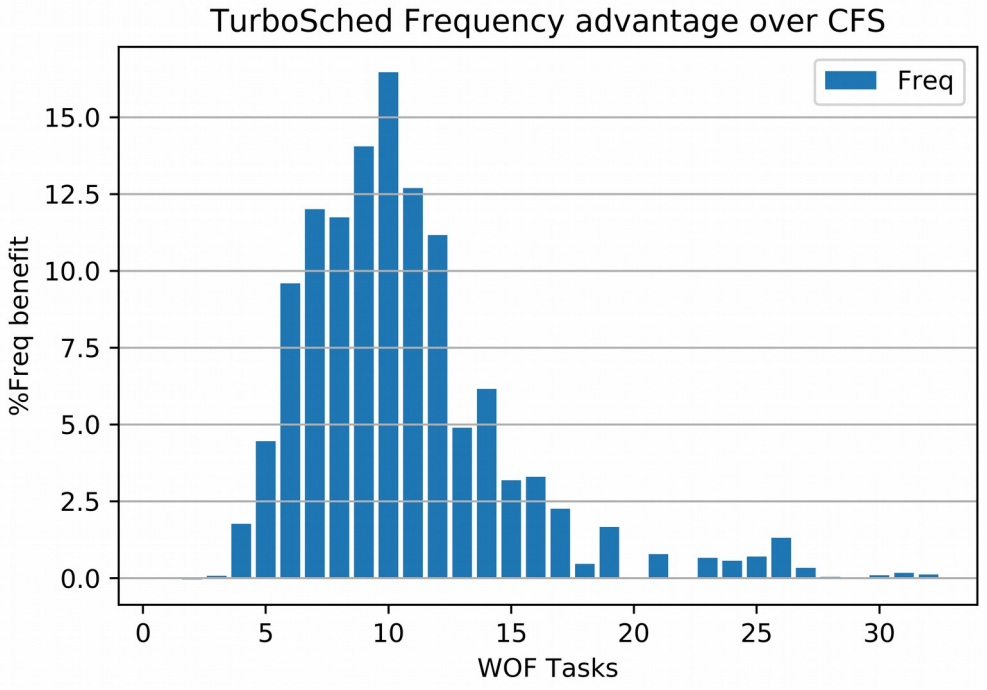
- Such that for SMT-4,

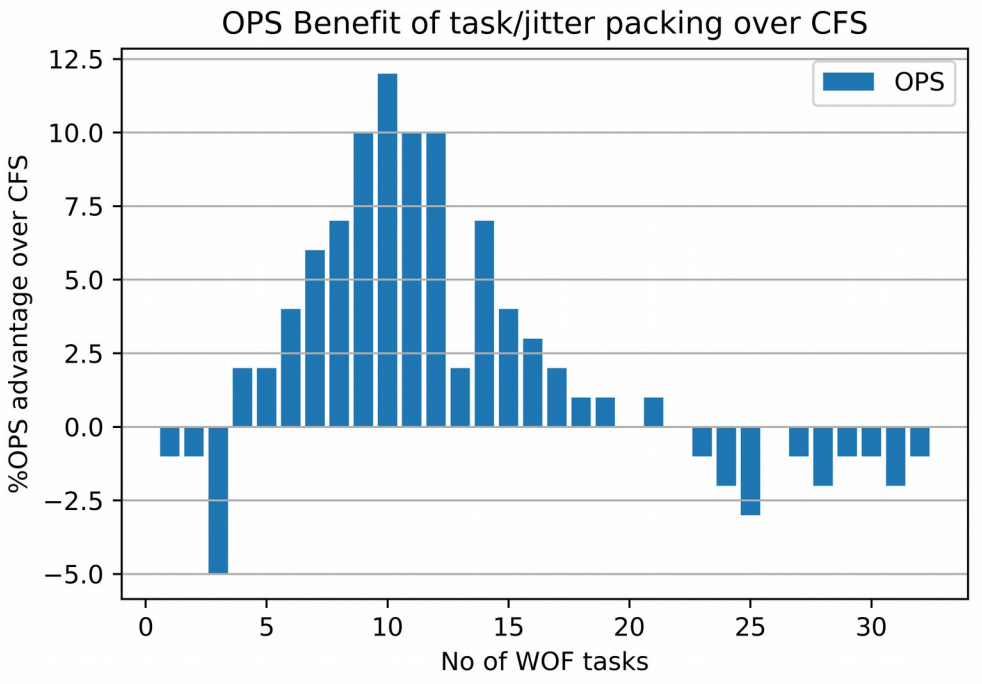
  Core cap = 1.5 * capacity_of(CPU 0)

Results

Results shows a benefit of upto

- 16% frequency benefit
- 12% in workload performance





Future

Use UCLAMP to classify tasks

- **Challenge**: Deals with CPUFREQ, while TurboSched deals with CPUIDLE

Adopt EAS model to control task placement for SMT systems as well.

- **Challenge**: Requested freq and obtained freq can be different in Turbo Range
- EM complexity is higher for >8 CPUs
- Frequency domain can be different making difficult to request frequency per thread/core basis

Contribute at TurboSched RFC: https://lkml.org/lkml/2019/5/16/824

Legal Statement

This work represents the view of the authors and does not necessarily represent the view of the employers (IBM Corporation).

IBM and IBM (Logo) are trademarks or registered trademarks of International Business Machines in United States and/or other countries.

Linux is a registered trademark of Linus Torvalds.

Other company, product and service names may be trademarks or service marks of others.

Thank You

Questions/Ideas?