IBM Linux Technology Center

Unified Energy Aware Scheduling

Dipankar Sarma <dipankar@in.ibm.com>

Vaidyanathan Srinivasan <svaidy@in.ibm.com>

Trinabh Gupta <trinabh@linux.vnet.ibm.com>

Linux Technology Center, IBM India Systems and Technology Lab

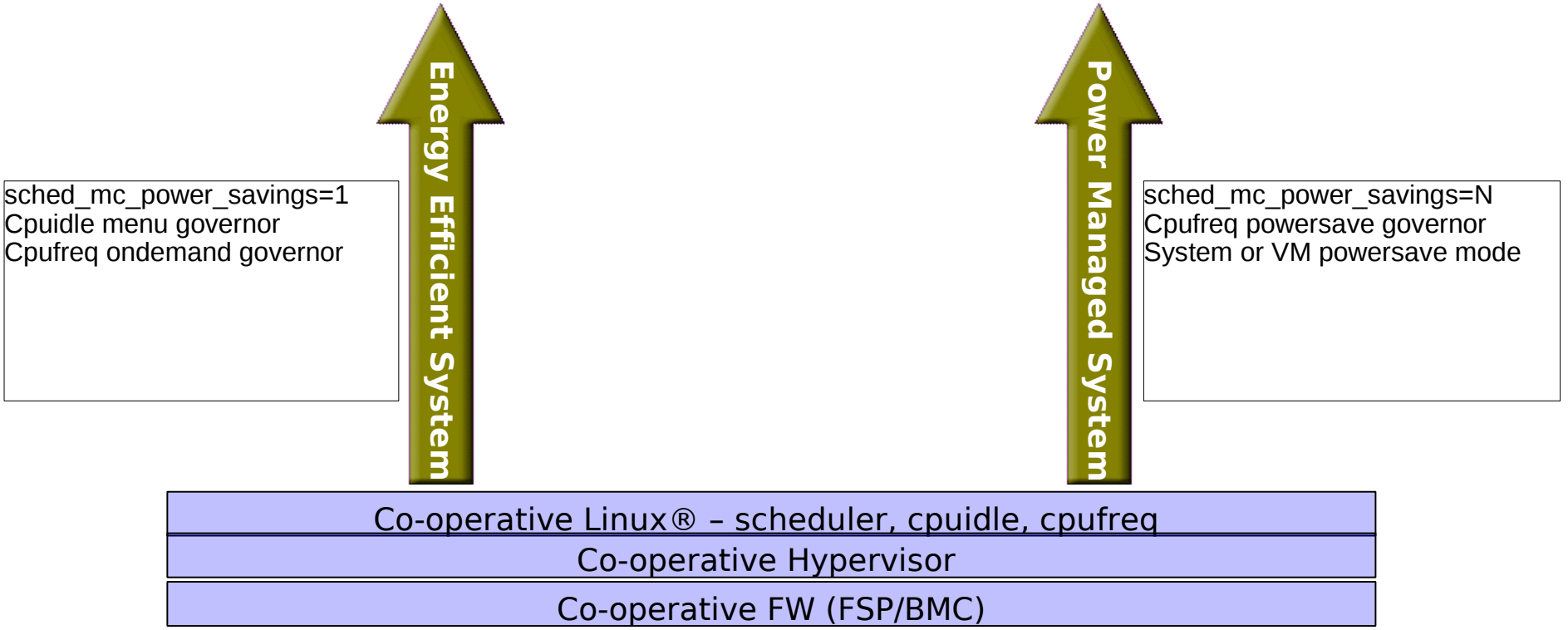
Agenda

- **Server energy and power management situations**
- **Unified scheduling**
- **Low power states - platform issues**
- **Scheduler integration for Idle sleep management**
- **Results**
- **Ongoing and future work**
- **Summary**

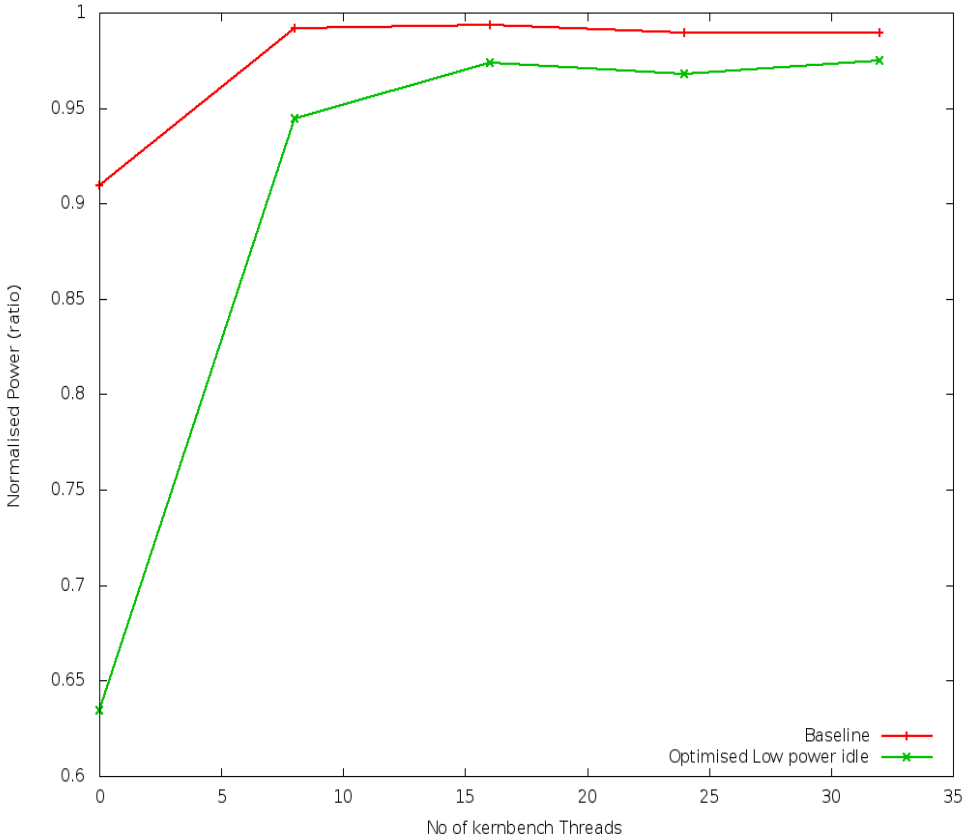
Server energy and power management situations

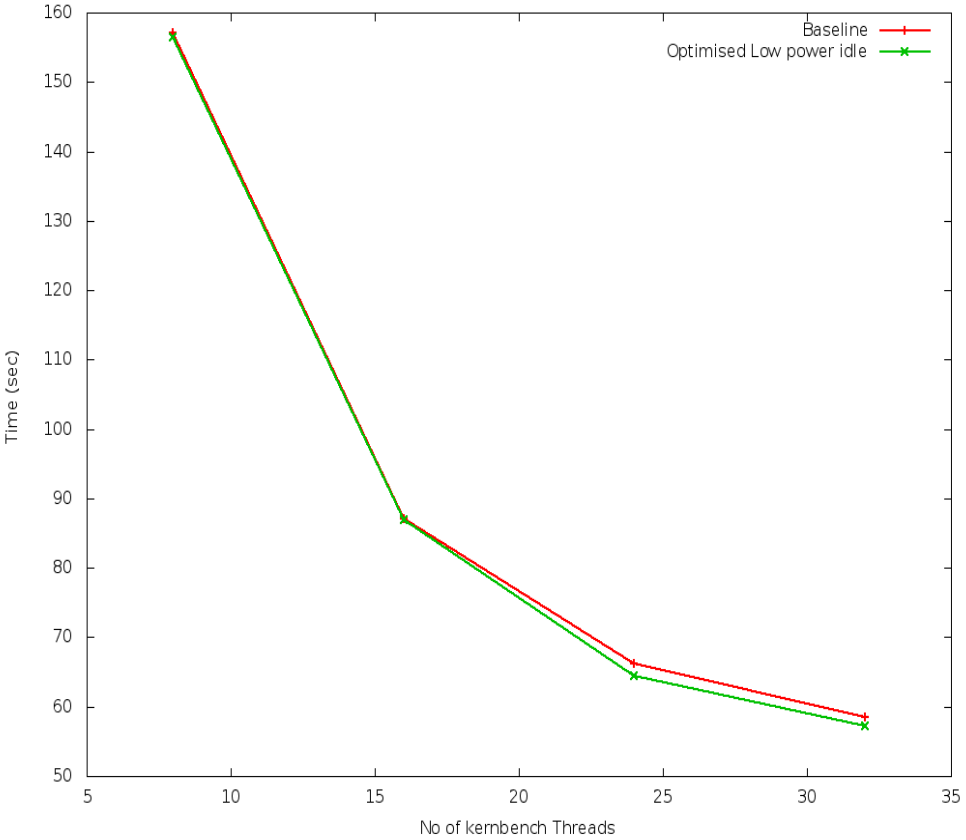
Specpower, perf/watt metric, EPA

Power constrained branch or datacenter



Power cap vs OS driven powersave mode





- System wide power cap implementations have limited co-operation with OS
- OS based powersave mode can achieve “soft” power limits with more efficiency
- Can be useful in power constrained datacenter situations

Unified scheduling

Motivation

- Doesn't distinguish between idle CPUs
  - Bigger impact on archs like powerpc
- Idle state transitions don't take topology into account
  - Cpuidle uses its own heuristics based on residency
  - Topology based biasing of low power states
- Idle loadbalancer selection doesn't take idle cpu power states into account
  - Semi-idle cpu for loadbalancer optimization helps (in mainline)
  - Very low power idle cores in a package can be selected for idle loadbalancer
- Some idle cpus may be better off in biased towards status quo
  - Bias towards avoiding wake up
  - Delay injection on long sleep
- Interrupt target cpus can be excluded from some idle optimizations
- Frequency and utilization together is an indication of capacity, may help in better decisions

Low power states – platform issues

Power state	Latency
C1	cycles
C1E	Microseconds
C6	100s of microseconds

- Imact on response time
- No special handling required
- Core level optimization is relevant
- Package level optimization is not relevant
- System level optimization is relevant

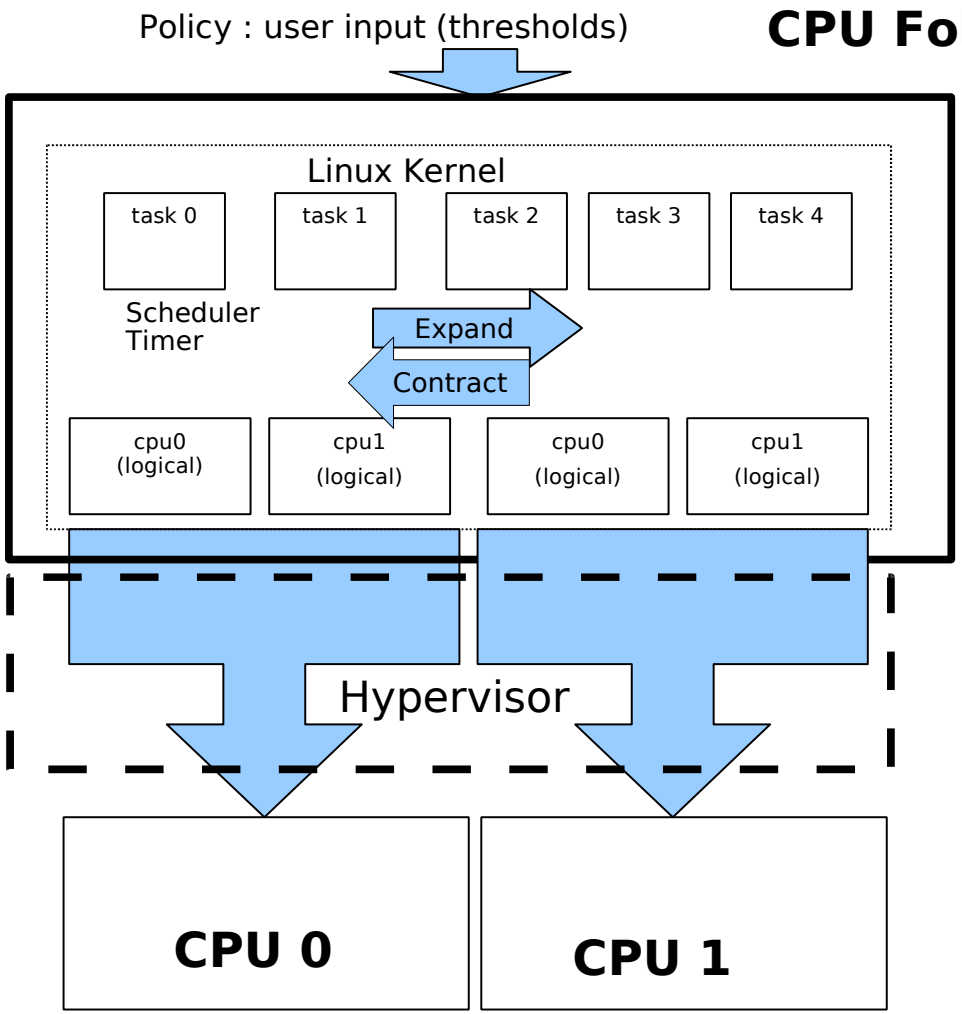
Intel™ Nehalem Family

Power state	Latency
Snooze	cycles
H_CEDE(0)	Range in us
H_CEDE(2)	Range of latencies

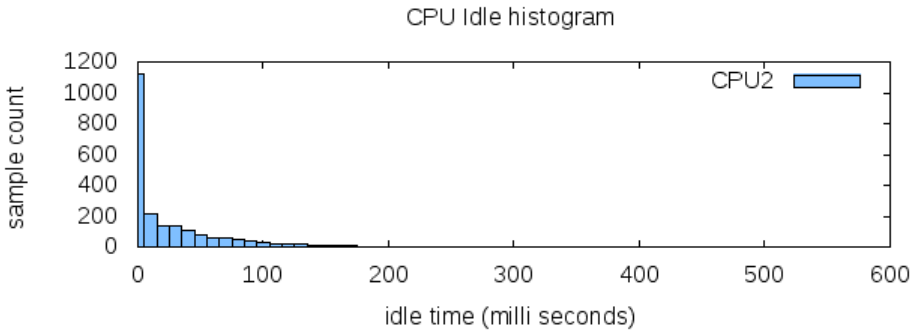
- Imact on response time
- Some states may need precaution for idle
- Use require significant care
- Core level optimization is relevant
- Package level optimization is relevant
- System level optimization is relevant
- Significant savings if used aggressively

IBM POWER® Family

Coarse low power state through hotplug

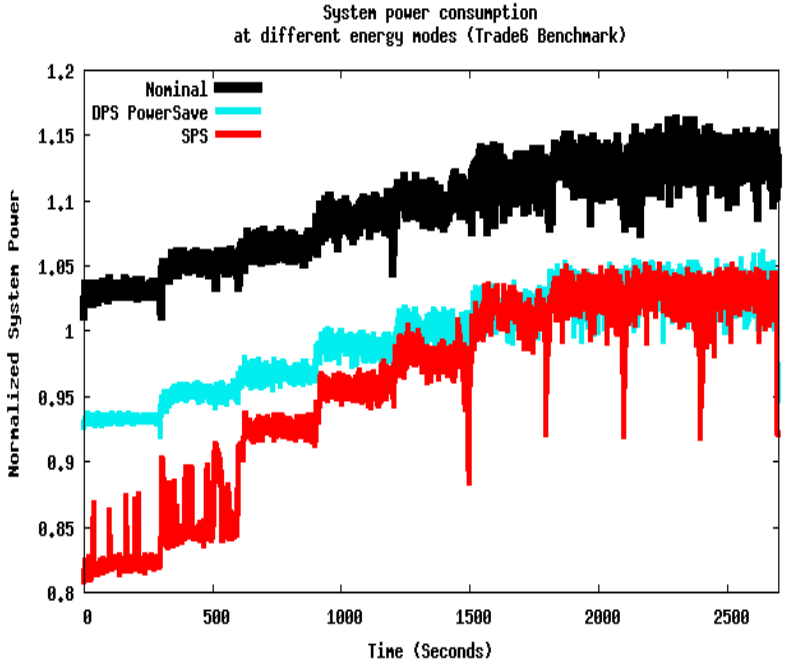


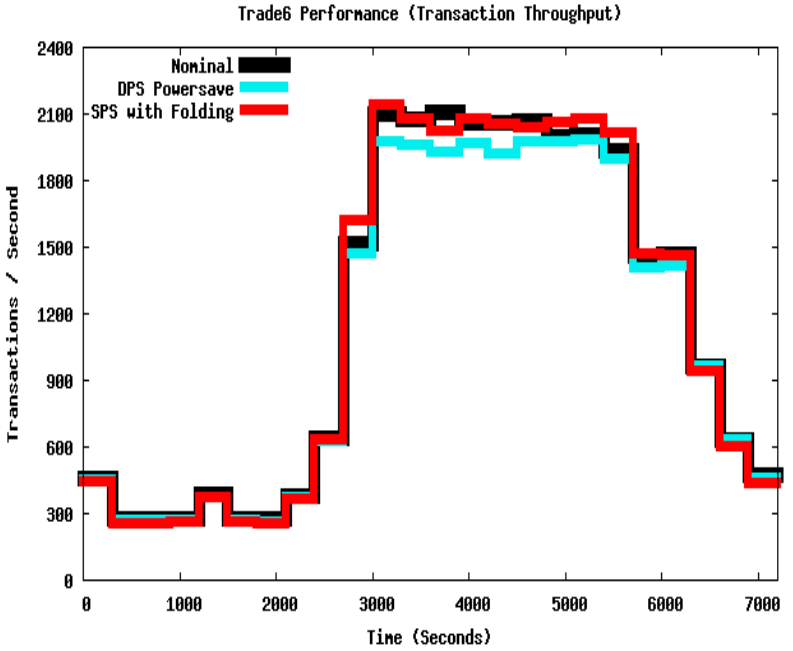
- Driven by large latency very low power state
- Guest OS decides on latency tolerance
- Expands or contracts vcpu set
- Coarse grain reaction to workload
- Uses the cpu online/offline framework
- May miss shorter (ms) idle opportunities
- Independent of cpuidle heuristics
- Can be made topology aware



Idle histogram at 60% load level

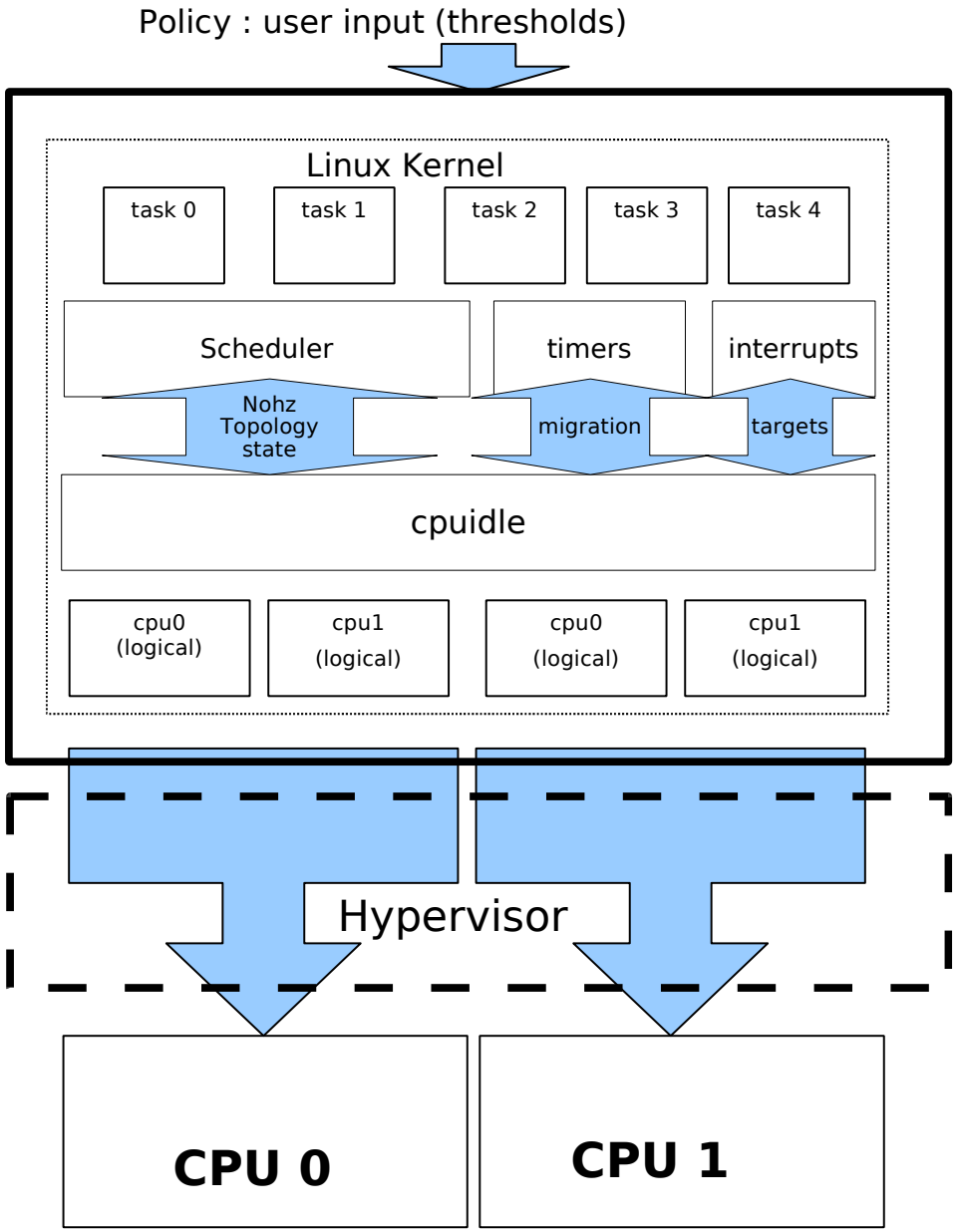
CPU folding results





- Ideal for powersave mode
- Performance loss is minimal for some transactional workloads and utilization patterns
- Works reasonably well at low utilization, but has some impact at high utilization levels

Unified scheduling architecture



- Bi-directional co-ordination between scheduler, cpuidle, timers and interrupts
- Cpuidle use nohz and topology awareness from scheduler
- Scheduler uses power state indication from cpuidle to avoid wakeup from low power
- Timer migration to migrate away from cpus with low power bias
- Interrupt frequency as input for selection of low power state

cpuidle with scheduler collaboration

Motivation

- Driven by large latency very low power state
- Policy driven bias towards letting full cores and packages remain idle
- Cpuidle itself doesn't have the global picture
- Cpuidle doesn't have control over scheduling on idle cpus

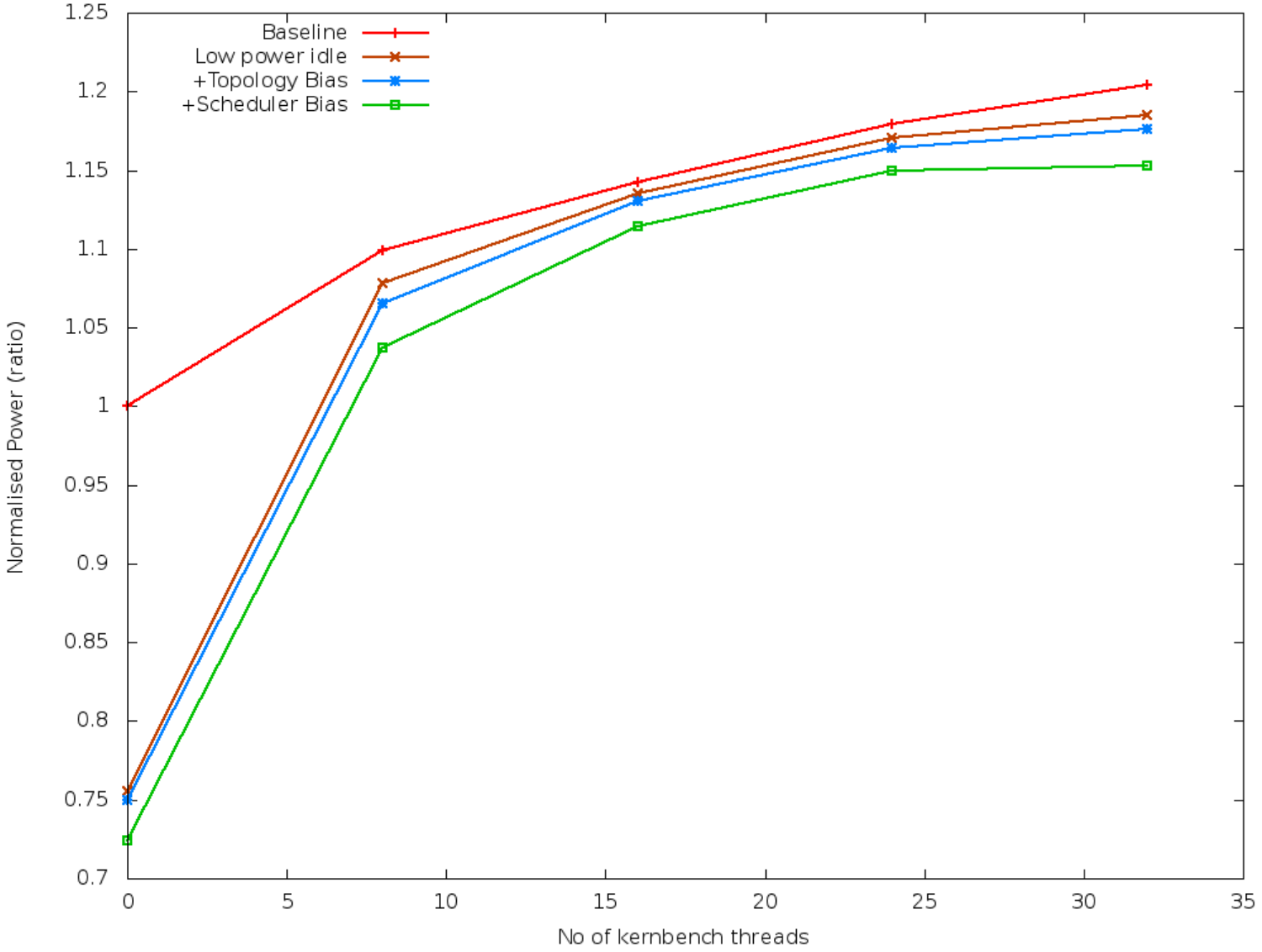
Cpuidle heuristics

- Use event prediction using tick_nohz_get_sleep_length()
- May need a ->check in cpuidle_state for additional checks
- Maintain percentage of threads for cores in lowest power state and in package
- **Toplogy bias -** Choose the lowest power state based on policy and threshold of % of threads and % of cores in lowest power state

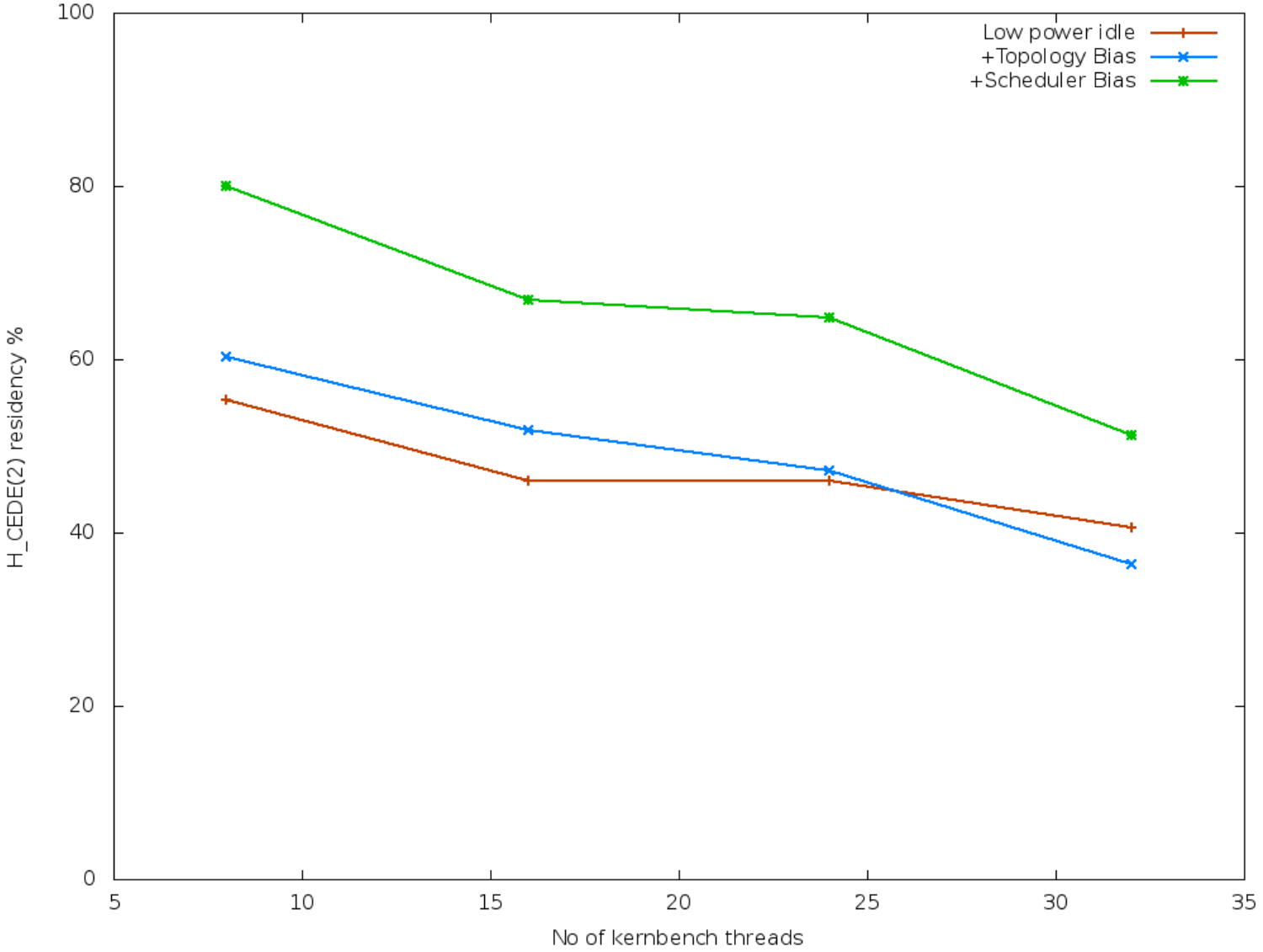
Scheduler heuristics

- Cpuidle API called to detect idle cpu to avoid wake up
- **Scheduler bias -** Bias *try_to_wake_up()* to avoid cpu in lowest power state

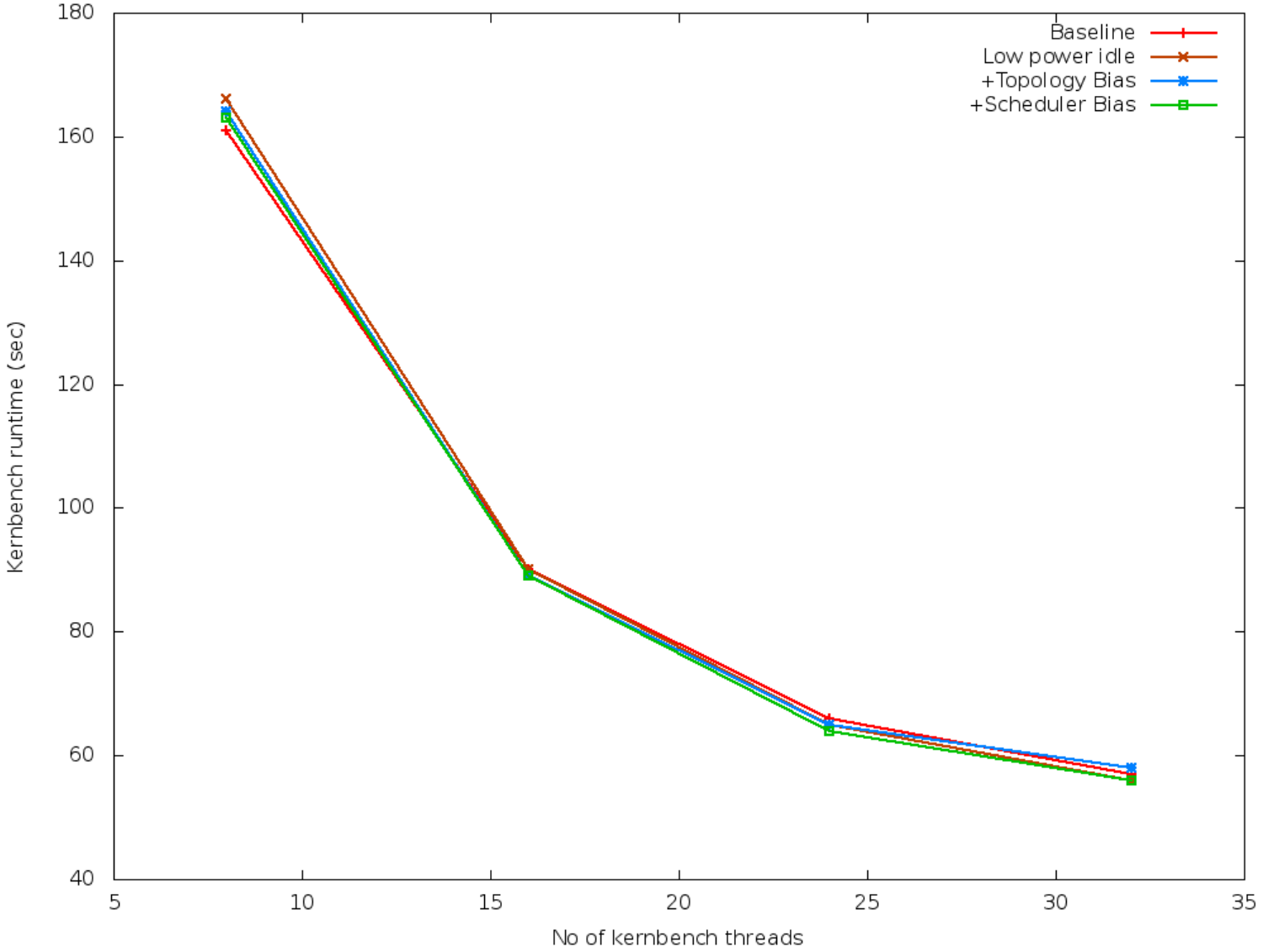
Cpuidle with scheduler collaboration - powerpc



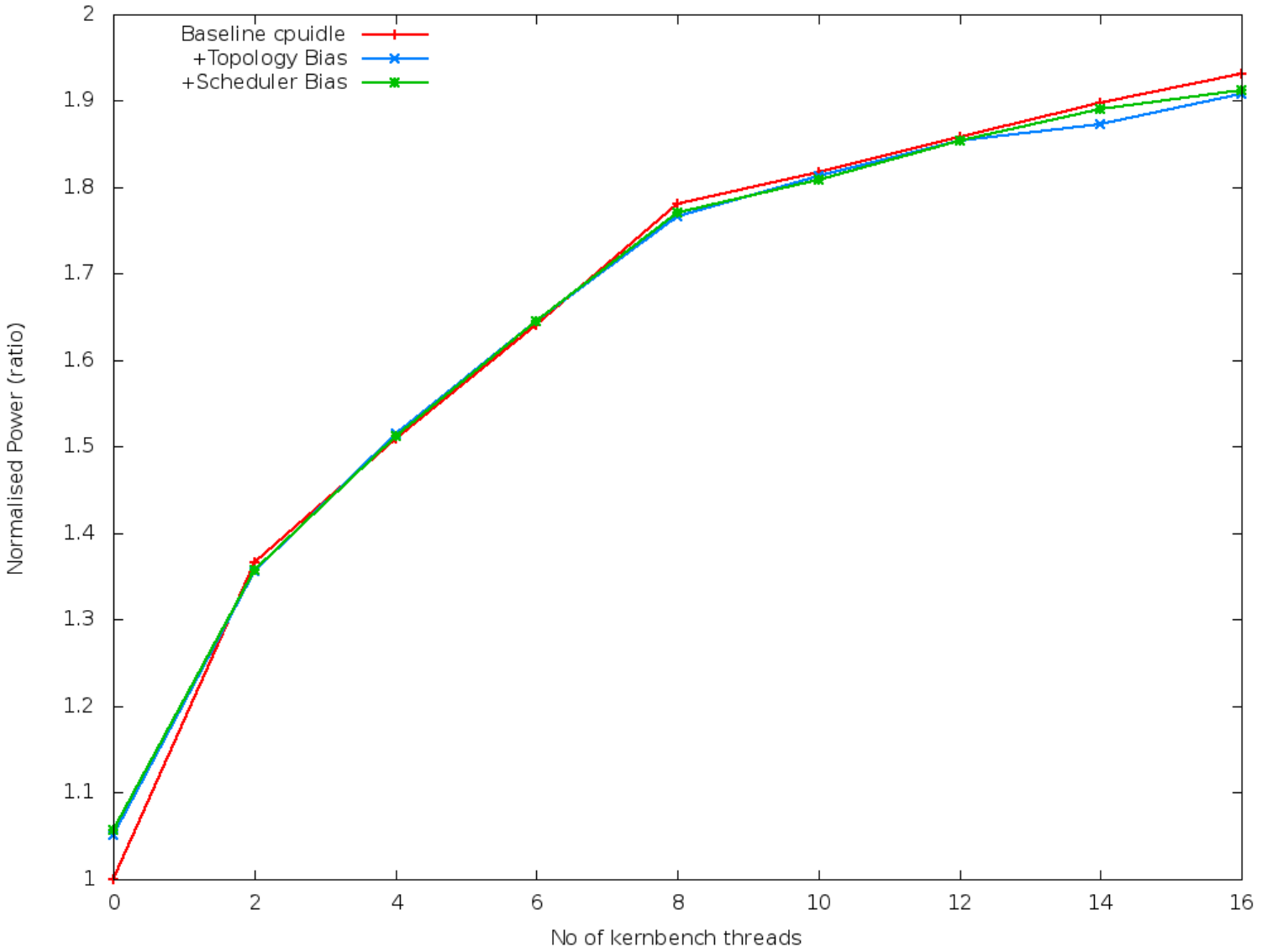
Cpuidle with scheduler collaboration - powerpc



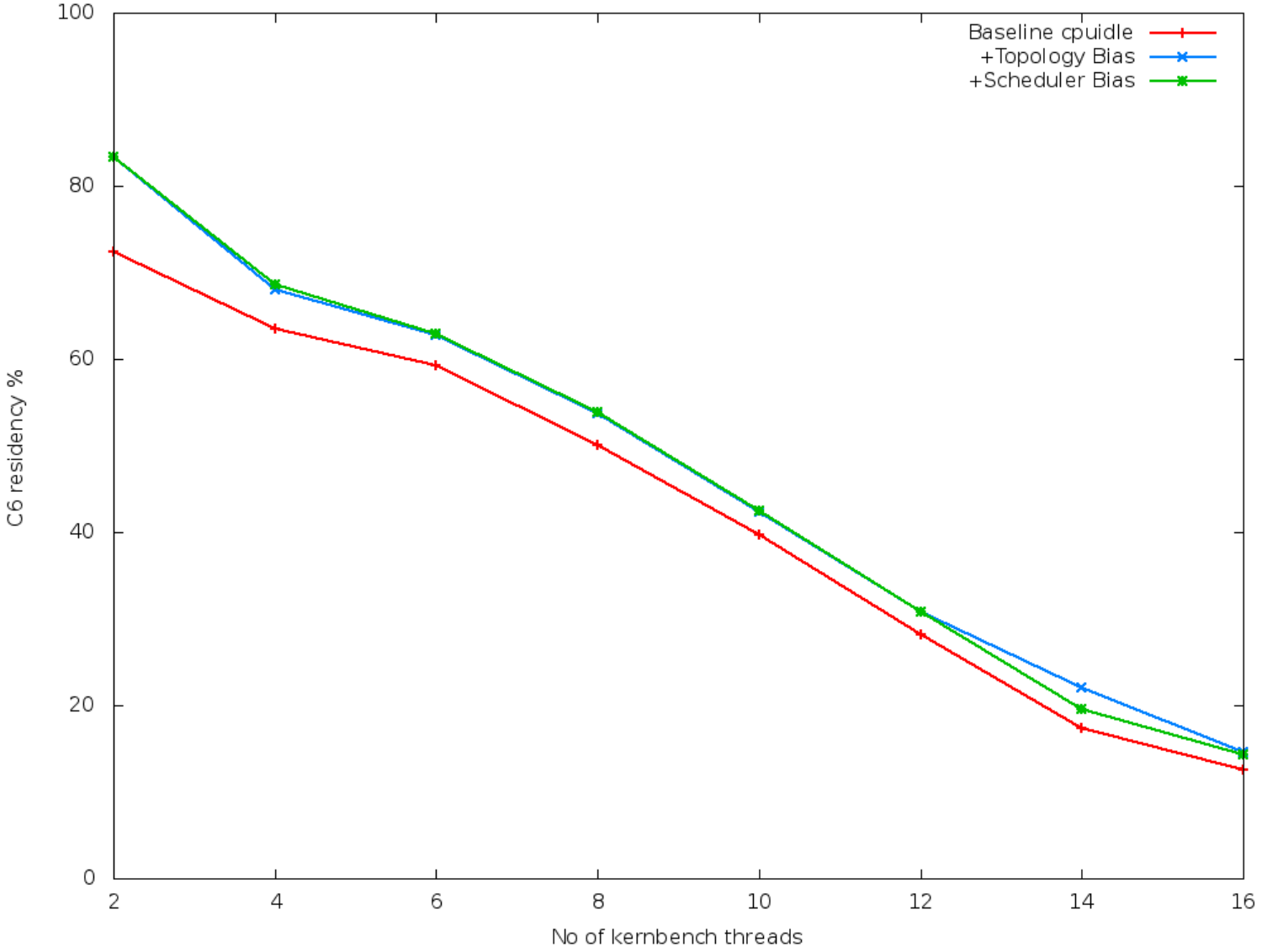
Cpuidle with scheduler collaboration - powerpc



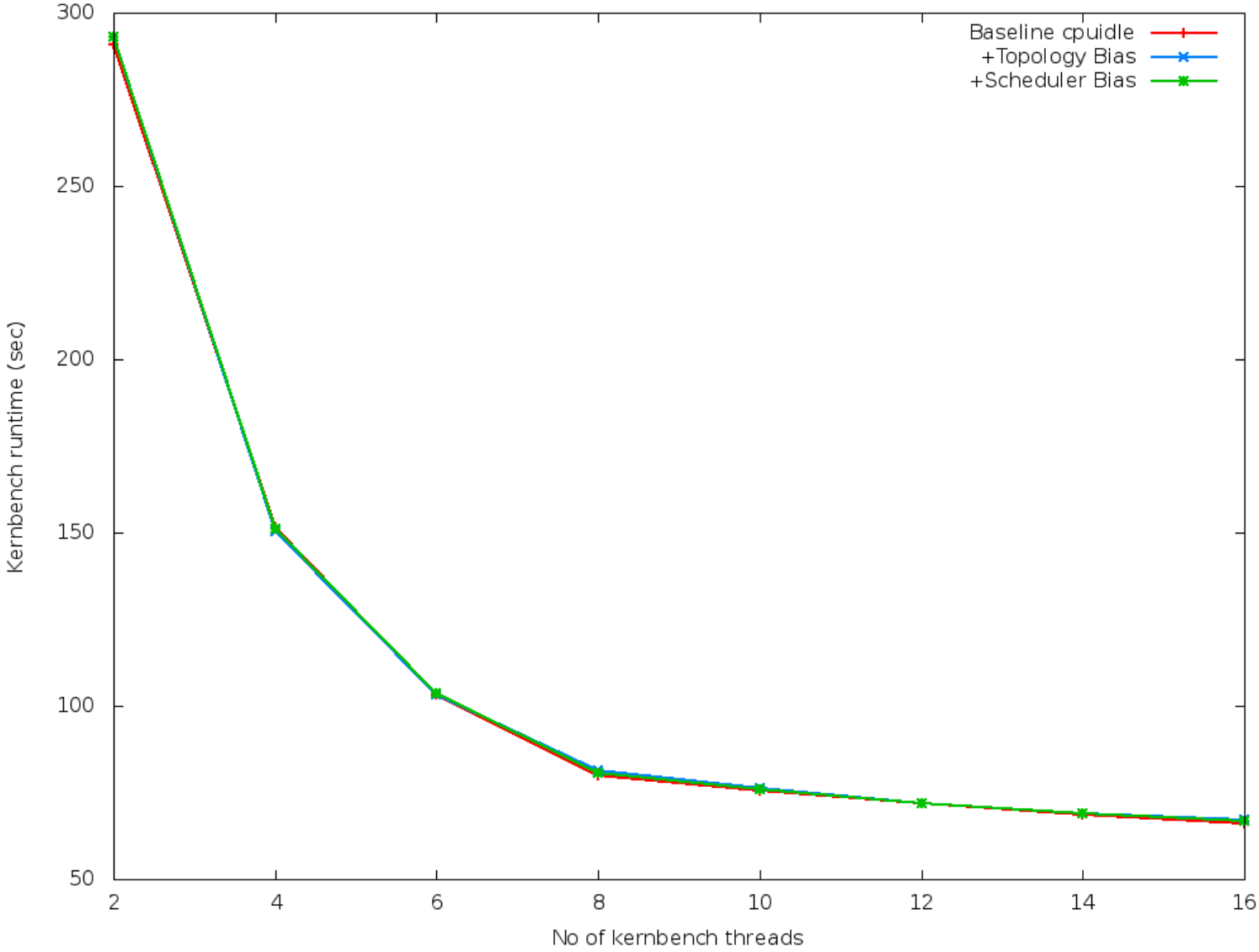
Cpuidle with scheduler collaboration - x86



Cpuidle with scheduler collaboration - x86



Cpuidle with scheduler collaboration - x86



Ongoing and future work

- Additional experiments
  - Bias idle load balancer away from deep sleep CPUs – hooks in select_nohz_load_balancer()
- Current work needs validation on as many architectures as possible
  - X86, powerpc experiments with various thresholds
  - More analysis on x86
- Our current code is researchy – with HW specific hacks
  - Continue that way until we can clearly show benefits across workloads and architectures
- Need validation across a wide range of workloads
- Cpufreq integration experiment is in the works
- Once all the data points are known, we need to come up with a list of intersection points with the scheduler and try to build consensus on corresponding interfaces

Summary

- With very low power states and architecture quirks, it has become necessary to look at more co-operative idle management among various subsystems
- Potential use has been identified in multiple architectures
- Will contribute, but will benefit greatly from reviews and inputs
- Collaboration between various architecture developers would be helpful to work out common interfaces needed

Legal Statements

- Copyright International Business Machines Corporation 2010.
- Permission to redistribute in accordance with Linux Plumbers Conference 2010 submission guidelines is granted; all other rights reserved.
- This work represents the view of the authors and does not necessarily represent the view of IBM.
- IBM, IBM logo, ibm.com are trademarks of International Business Machines Corporation in the United States, other countries, or both.
- Intel is a trademark or registered trademark of Intel Corporation or its subsidiaries in the United States and other countries.
- Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.
- Other company, product, and service names may be trademarks or service marks of others.
- References in this publication to IBM products or services do not imply that IBM intends to make them available in all countries in which IBM operate
- INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you. This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.