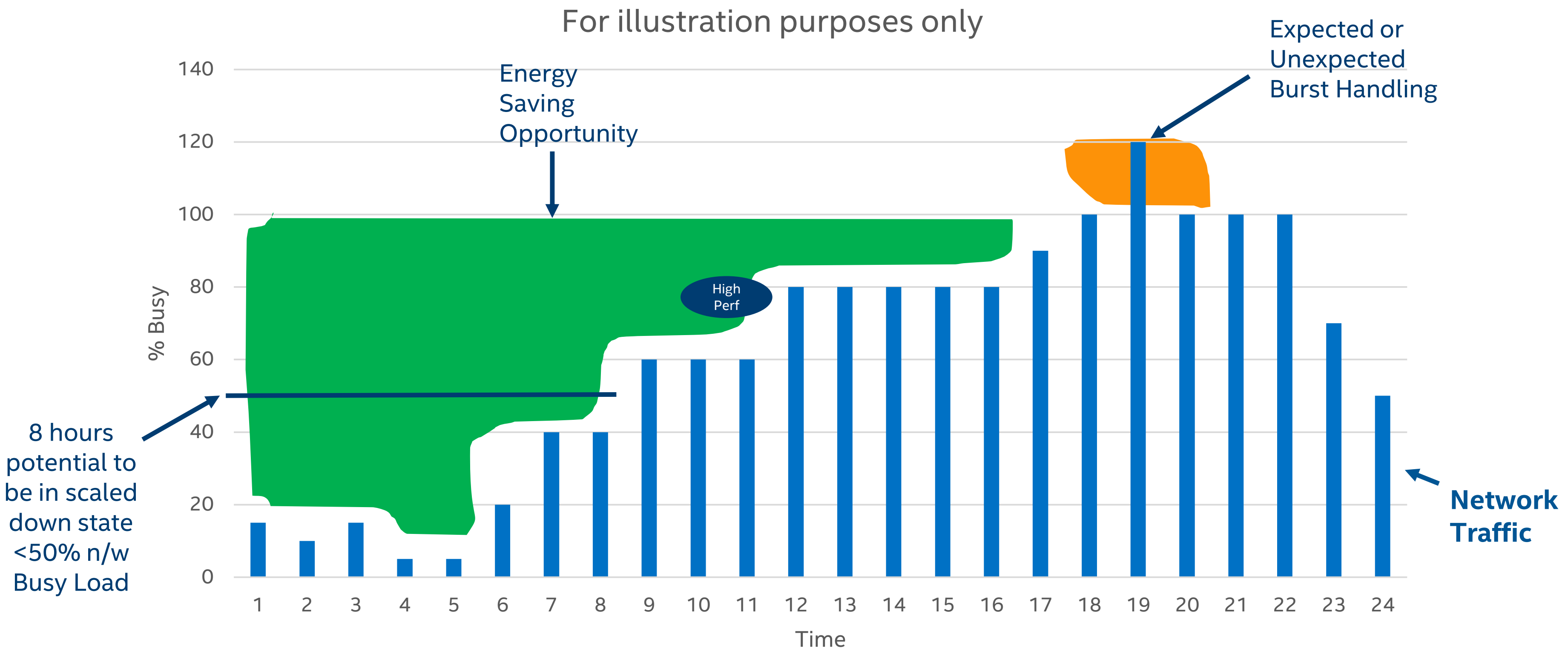
PACKET PROCESSING PERFORMANCE & POWER EFFICIENCY

Ma Liang
Senior Software Engineer INTEL
07, Sep, 2019

Mapping Power Usage to Network Traffic

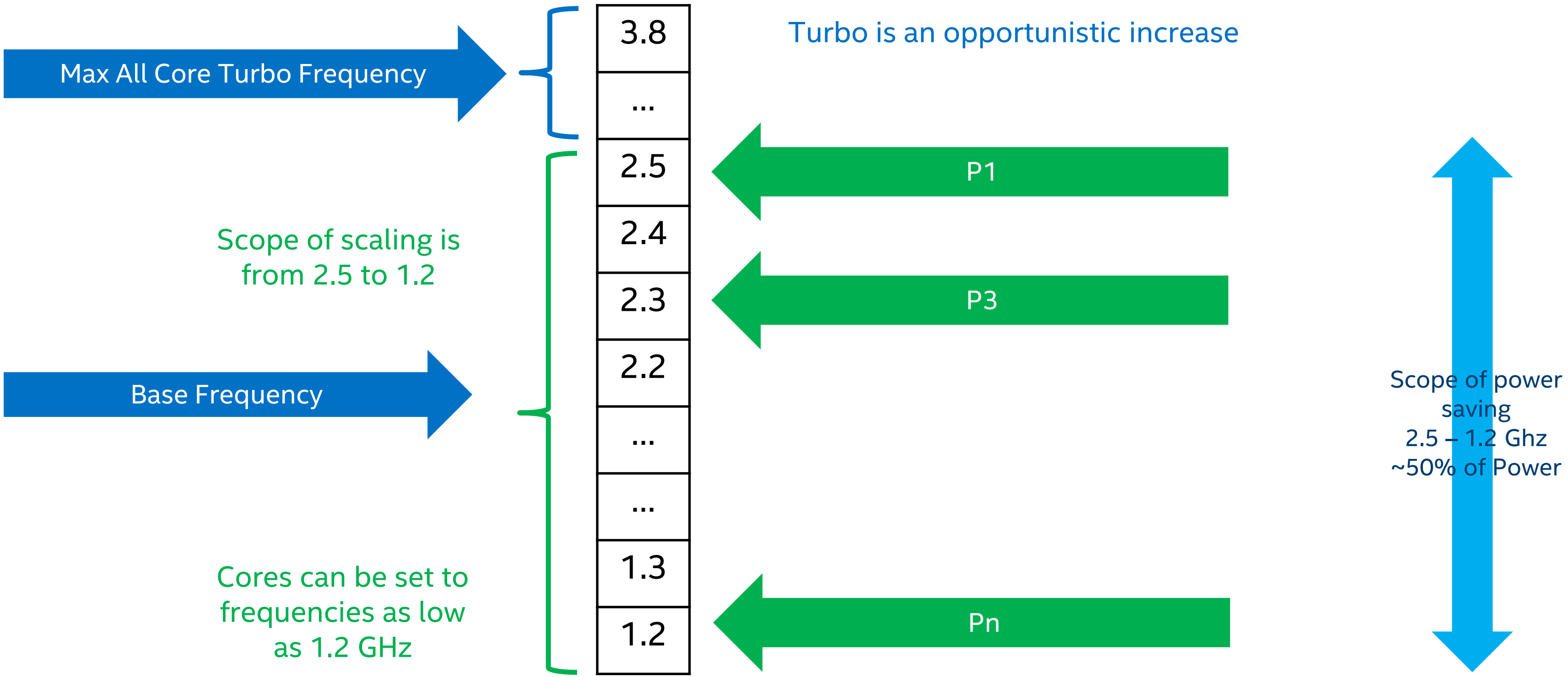


Summary Table of Power Management Features

Feature	Summary
P-state	Changing frequency, works per core. Execution continues.
C-state	Turning off execution of cores and instructions. Fast Exit(C1) and Longer Exit (C6), power saving versus exit latency.
Turbo Boost	Allows for exceeding base frequencies, opportunistic frequency increase. Can be controlled per core.
Uncore Frequency Scaling (UFC)	Interconnect and L3 shared cache frequency scaling for energy efficiency
Hardware P-State	Intel® Speed Select Technology (Hardware P-state, HWP) is a capability for cooperative hardware + software performance control
Intel® SST-Base Frequency	Intel® Speed Select Technology (Base Frequency) is a capability which allow application to choose High Priority or Standard Priority cores

P-States Overview

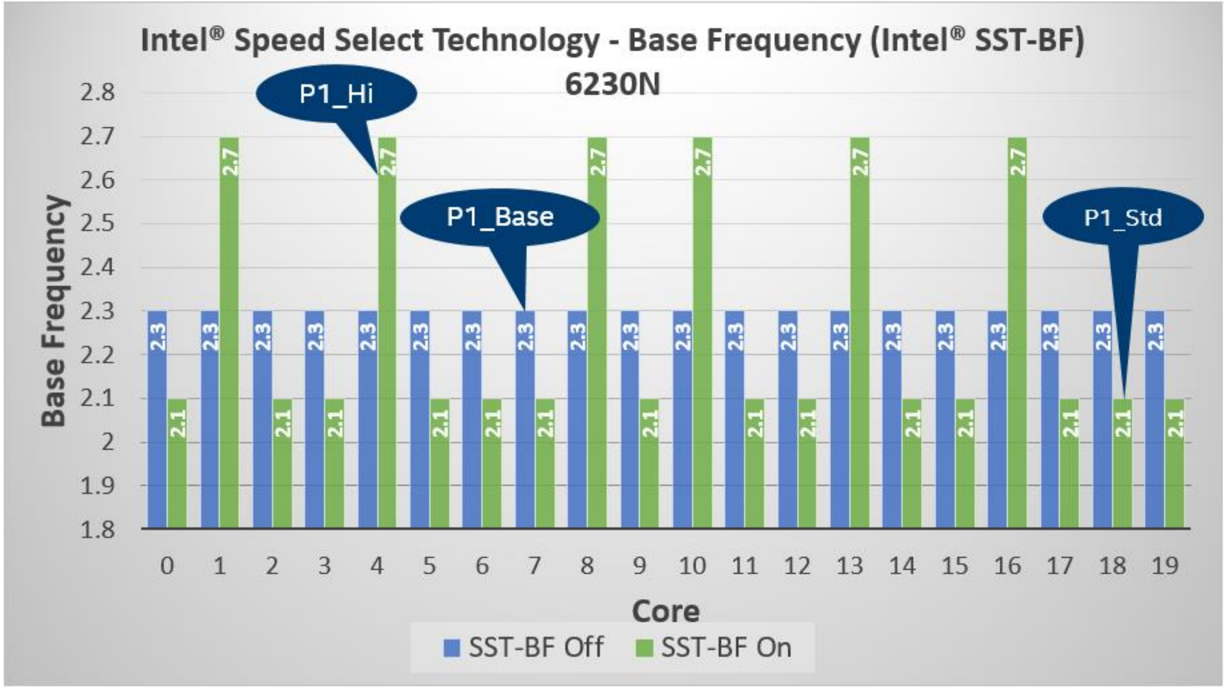
Frequency range in GHz of Scalable 8180



Per Core P State Control at Runtime via SW

Intel® SST-Base Frequency

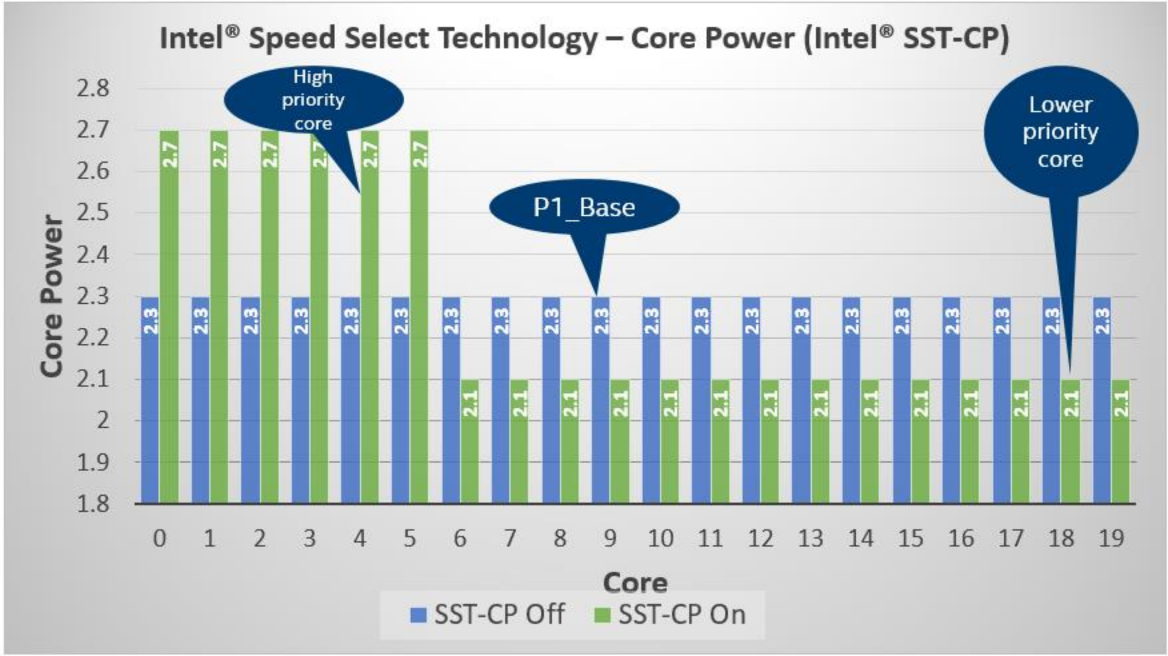
- Trade off base frequency between higher and lower priority cores
- Improve overall performance by giving critical cores higher frequency
- Intel® Speed Select Technology - Base Frequency (Intel® SST-BF) is enabled in BIOS at boot time; activated/deactivated at run time



All SKUs, frequencies, and performance estimates are PRELIMINARY and can change without notice.

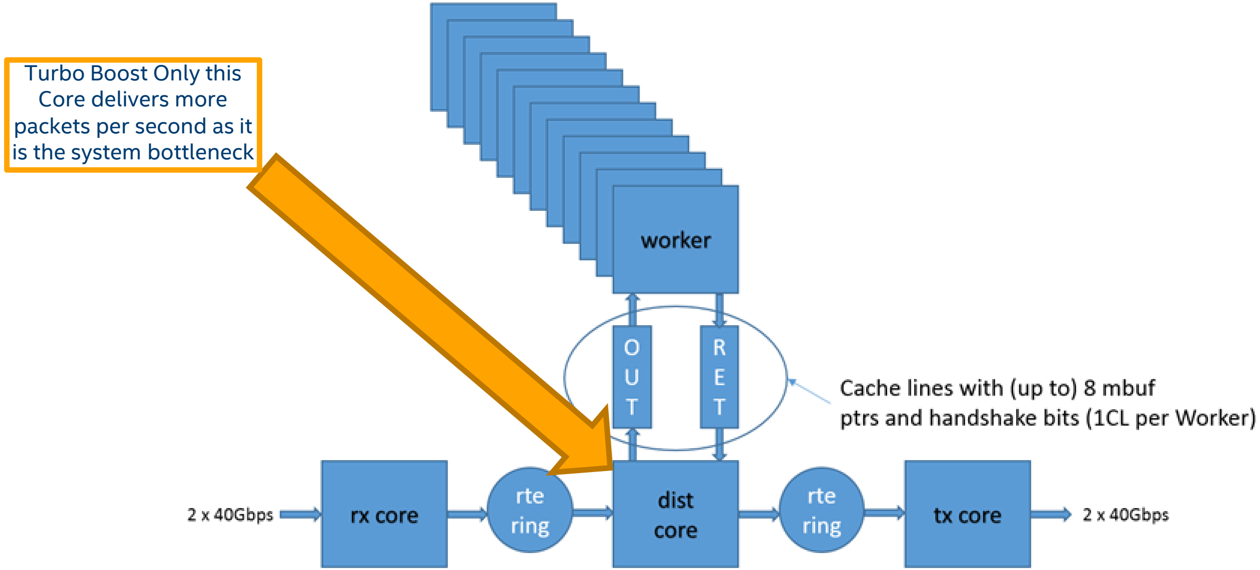
Intel® SST-Core Power

- TDP is Fixed
- Distribute surplus power to cores based on SW assigned weights/priorities
- Improve performance by directing frequency to high priority cores
- Best Effort



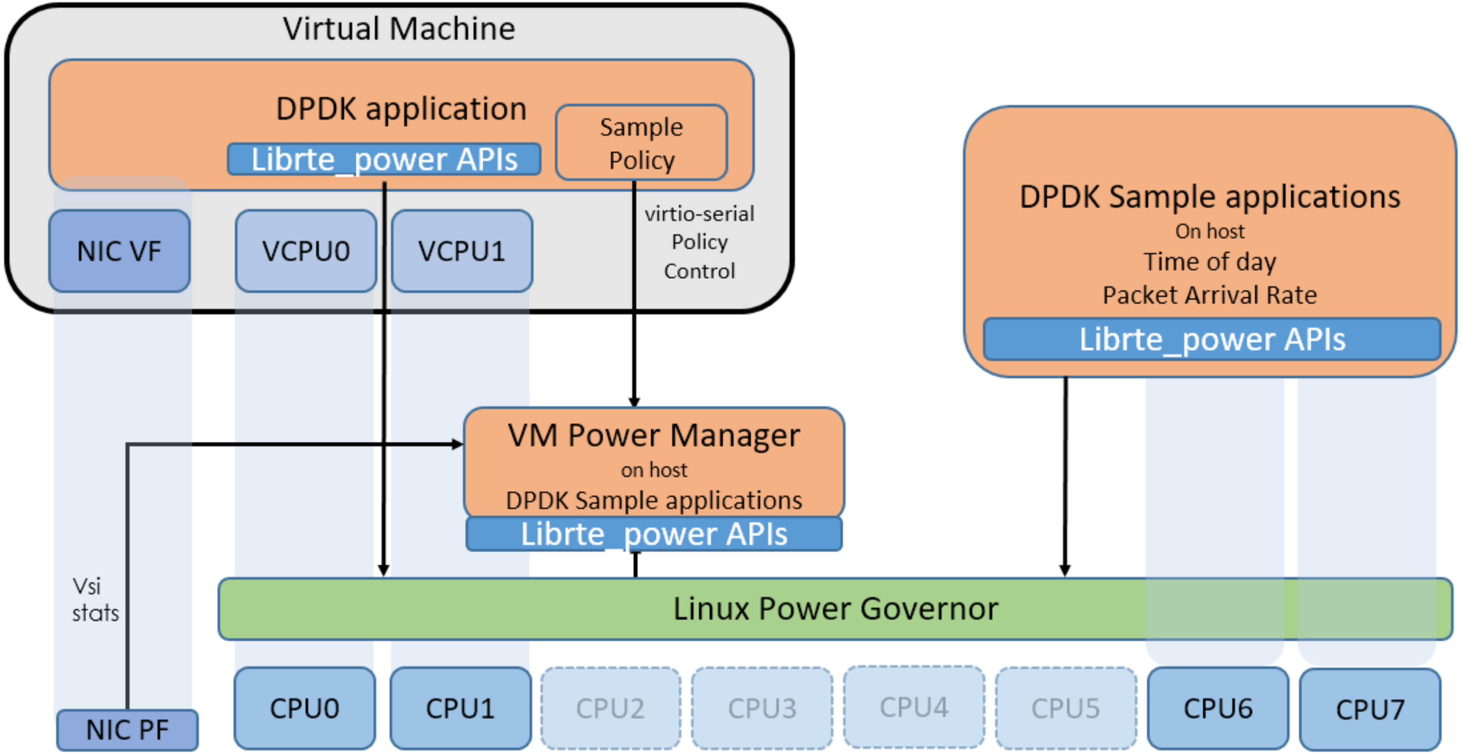
All SKUs, frequencies, and performance estimates are PRELIMINARY and can change without notice.

Performance Gain Potential



Many use cases: boosting 1-2 cores will benefit workloads

Existing DPDK Power Capabilities



DPDK API Reference

rte_power.h File Reference

```
#include <rte_common.h>
#include <rte_byteorder.h>
#include <rte_log.h>
#include <rte_string_fns.h>
```

Go to the source code of this file.

Data Structures

struct	rte_power_core_capabilities

Typedefs

typedef uint32_t(*	rte_power_freqs_t)(unsigned int lcore_id, uint32_t *freqs, uint32_t num)
typedef uint32_t(*	rte_power_get_freq_t)(unsigned int lcore_id)
typedef int(*	rte_power_set_freq_t)(unsigned int lcore_id, uint32_t index)
typedef int(*	rte_power_freq_change_t)(unsigned int lcore_id)
typedef int(*	rte_power_get_capabilities_t)(unsigned int lcore_id, struct rte_power_core_capabilities *caps)

Functions

int	rte_power_set_env (enum power_management_env env)
void	rte_power_unset_env (void)
enum power_management_env	rte_power_get_env (void)
int	rte_power_init (unsigned int lcore_id)
int	rte_power_exit (unsigned int lcore_id)

Variables

rte_power_freq_change_t	rte_power_freq_up
rte_power_freq_change_t	rte_power_freq_down
rte_power_freq_change_t	rte_power_freq_max
rte_power_freq_change_t	rte_power_freq_min
rte_power_freq_change_t	rte_power_turbo_status
rte_power_freq_change_t	rte_power_freq_enable_turbo
rte_power_freq_change_t	rte_power_freq_disable_turbo

DETERMINING AND PREDICTING LOAD

Meeting the needs of an on demand network

Scale always on DPDK performance with the network demand

Common Challenges

- Always On
  - Adjust PMD cores frequency to adjust to packet demand
  - Potential to save power drawn per core using frequency scaling
    - ++ from sleeping

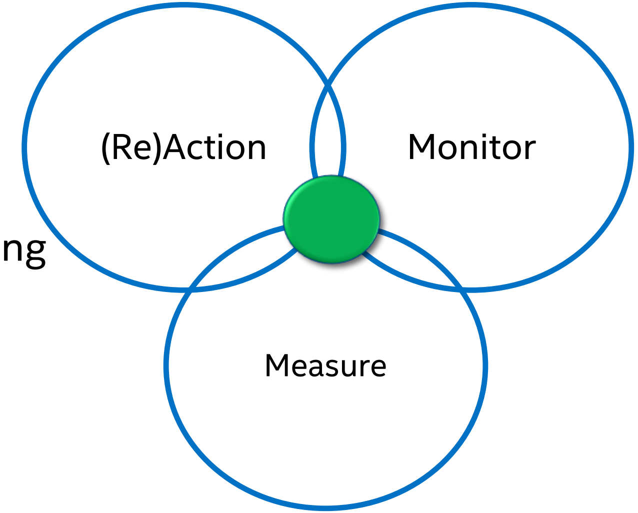
Speed of Re(Action)

- Challenge: Fast Scale Up to react to increases in n/w traffic
- Time = queueing/buffering

Challenge: Fast Monitor & Reaction Time

- Closer to hardware gives faster reaction time

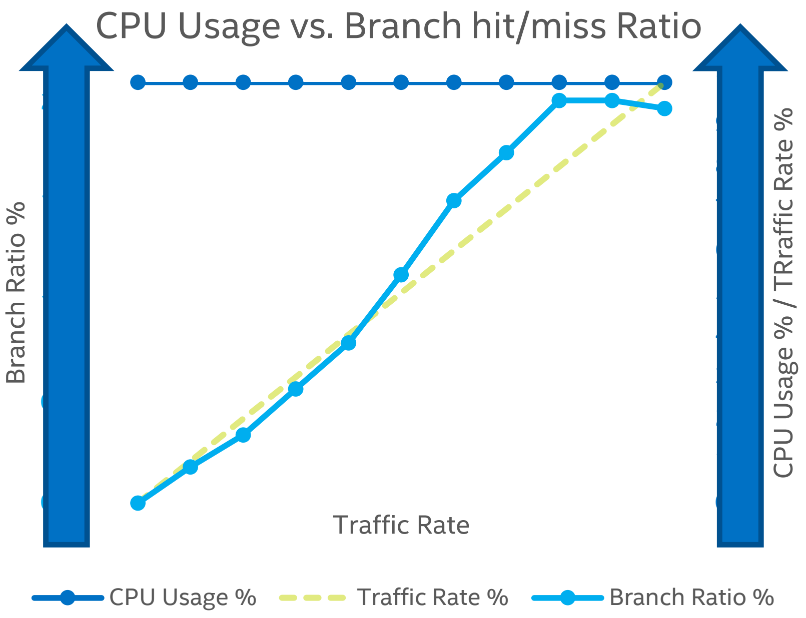
Move to Policy based control



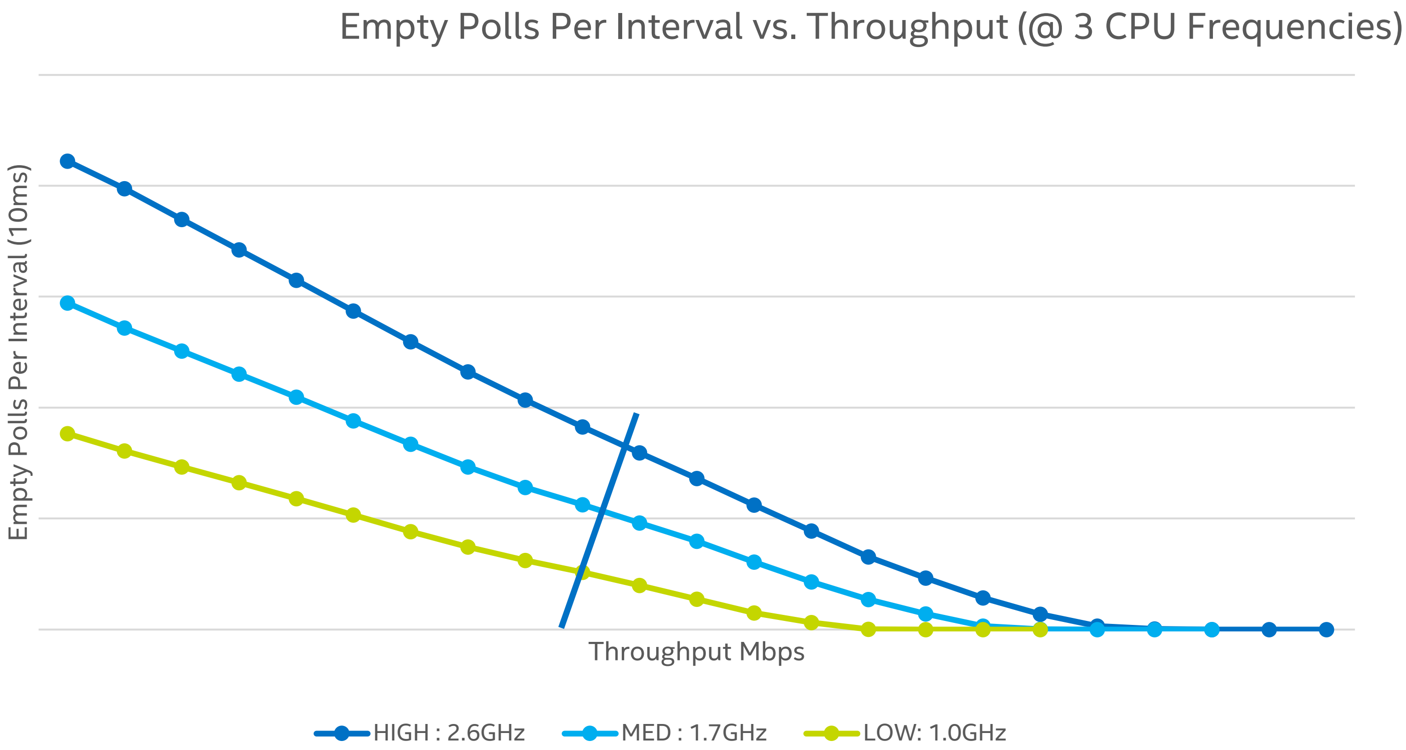
Apply Power Where and When it's needed

Poll Loop Work Rate Detection (PMD Load%)

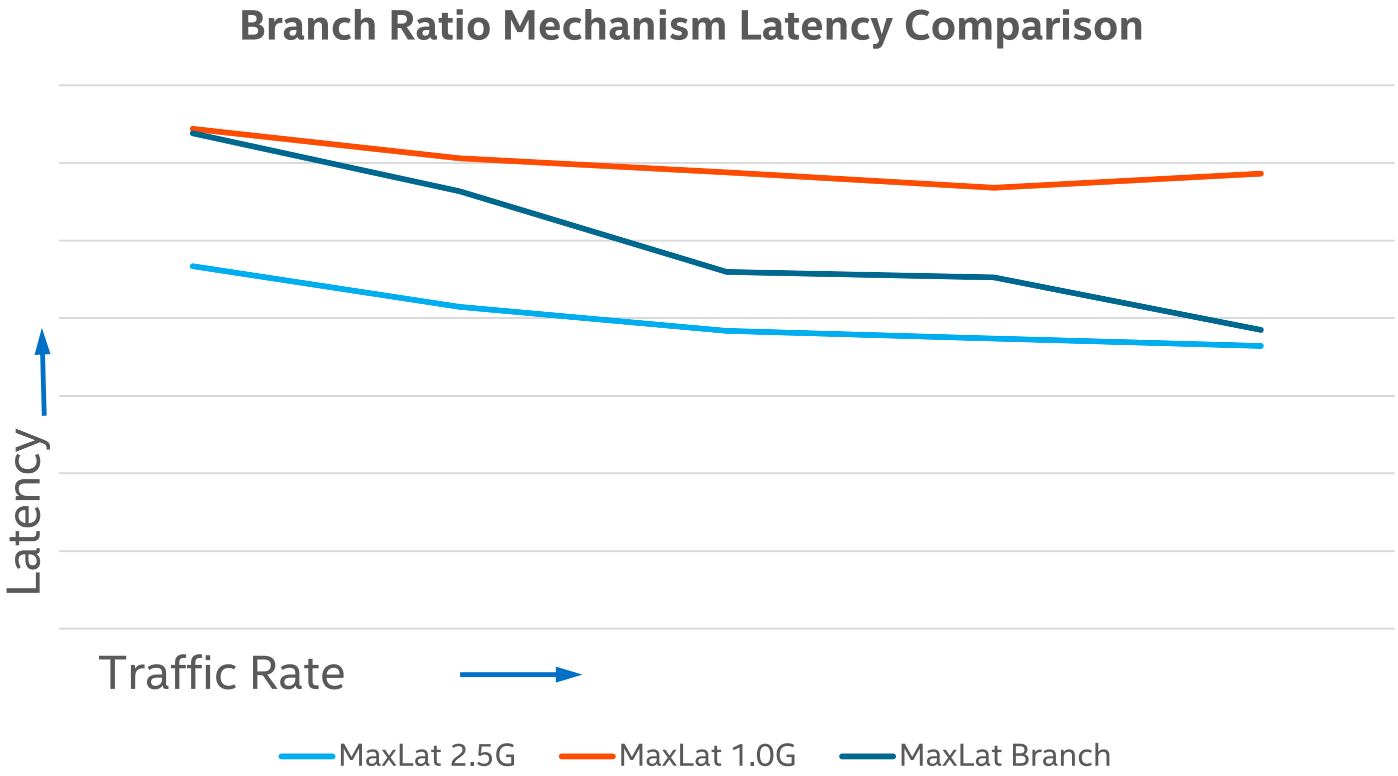
- CPU Load is always 100% for DPDK PMD Poll Loops
- Actual workload may be zero (processing zero packets)
- Use PMU counters to calculate the actual work done
- Use the ratio between Branch Hits and Branch Misses
- Ratio is low when tight code loop (empty polling), and significantly is higher when processing packets (due to larger code path)
- Almost linear with traffic rate



Empty Poll Number Driven Model



Latency Comparison

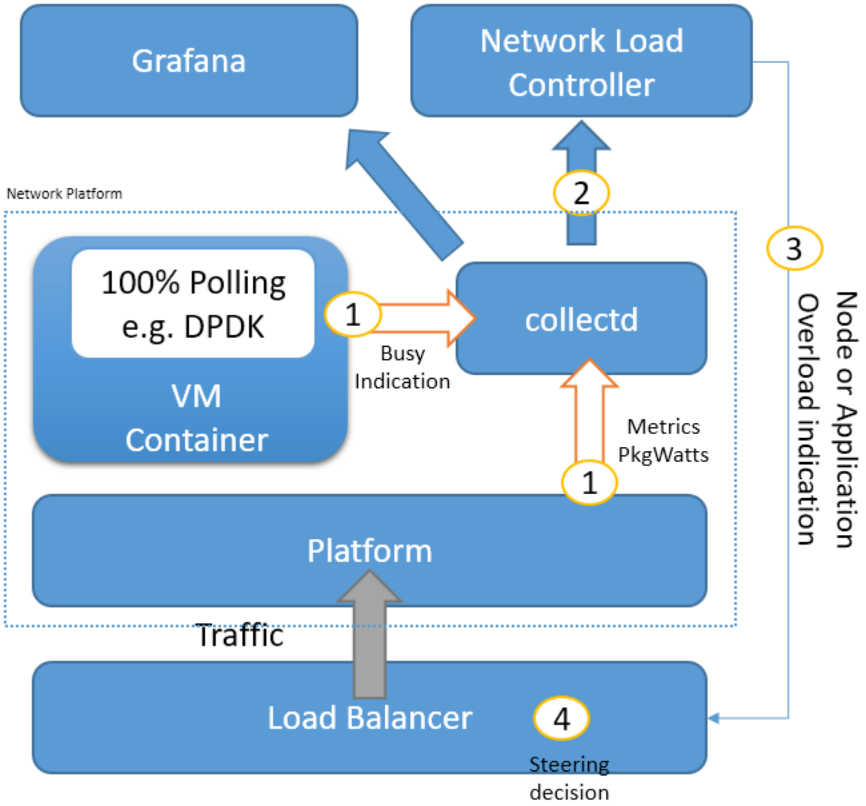


ECO-SYSTEM ENABLEMENT

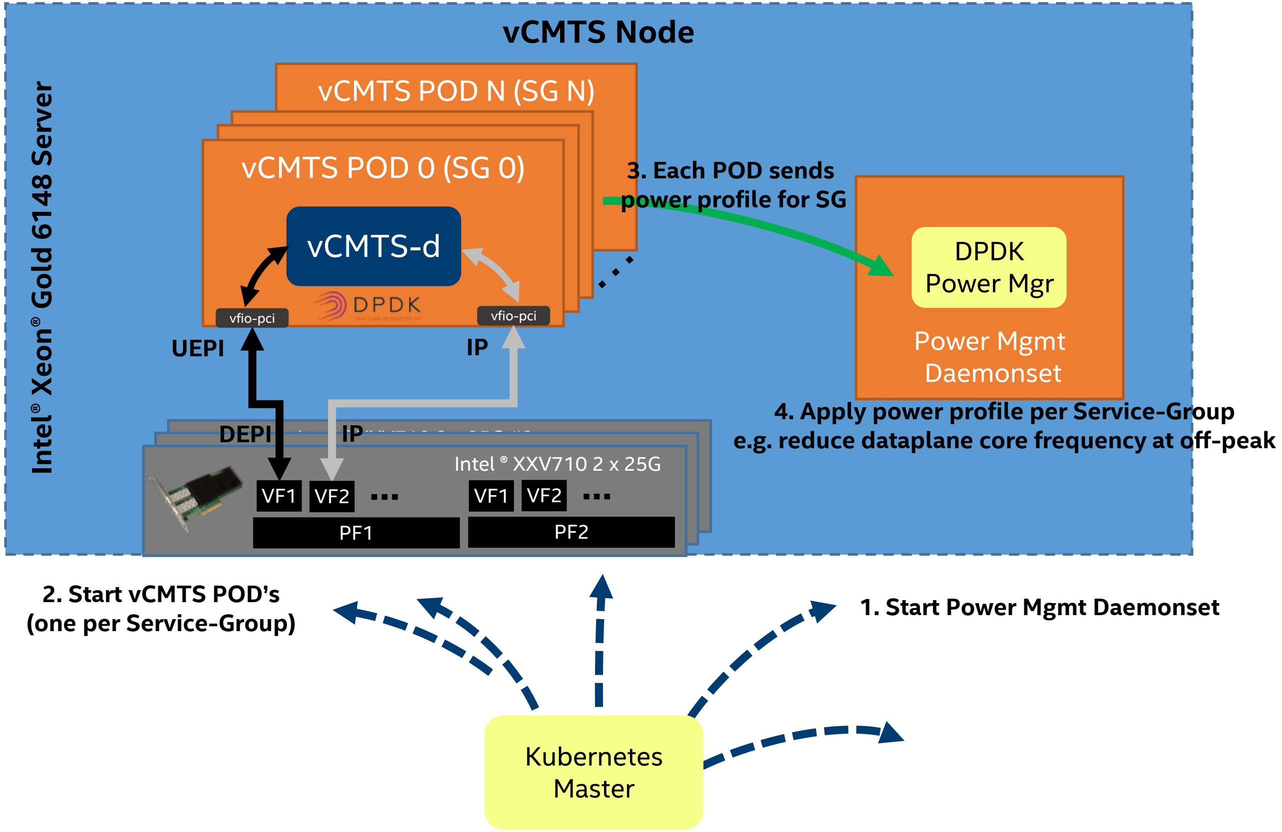
Telemetry Integration

- Metric **Busy Indication**
- Metric **PkgWatts**
- collectd (next) with new dpdk_plugin, updated platform power metrics
- DPDK 19.08 with new telemetry mode sample

Metric #1 Busy Indication	Metric #2 PkgWatts	Action
No	No	Steady State
Yes	No	Backoff
No	Yes	Backoff



Kubernetes Integration Example



Software Reference

- DPDK APIs available
  - Application APIs to support in band and out of band use cases
  - http://dpdk.org/doc/api/rte__power_8h.html
- Sample applications
  - L3fwd-power
- Presentations
  - https://dpdksummit.com/Archive/pdf/2017Userspace/DPDK-Userspace2017-Day2-8-Power.pdf
  - https://www.dpdk.org/wp-content/uploads/sites/35/2018/10/pm-01-DPDK_Summit18_PowerManagement.pdf
- Power Tools Repo Info
  - https://github.com/intel/commspowermanagement

intel®