Recent Power Management Enhancements in DPDK

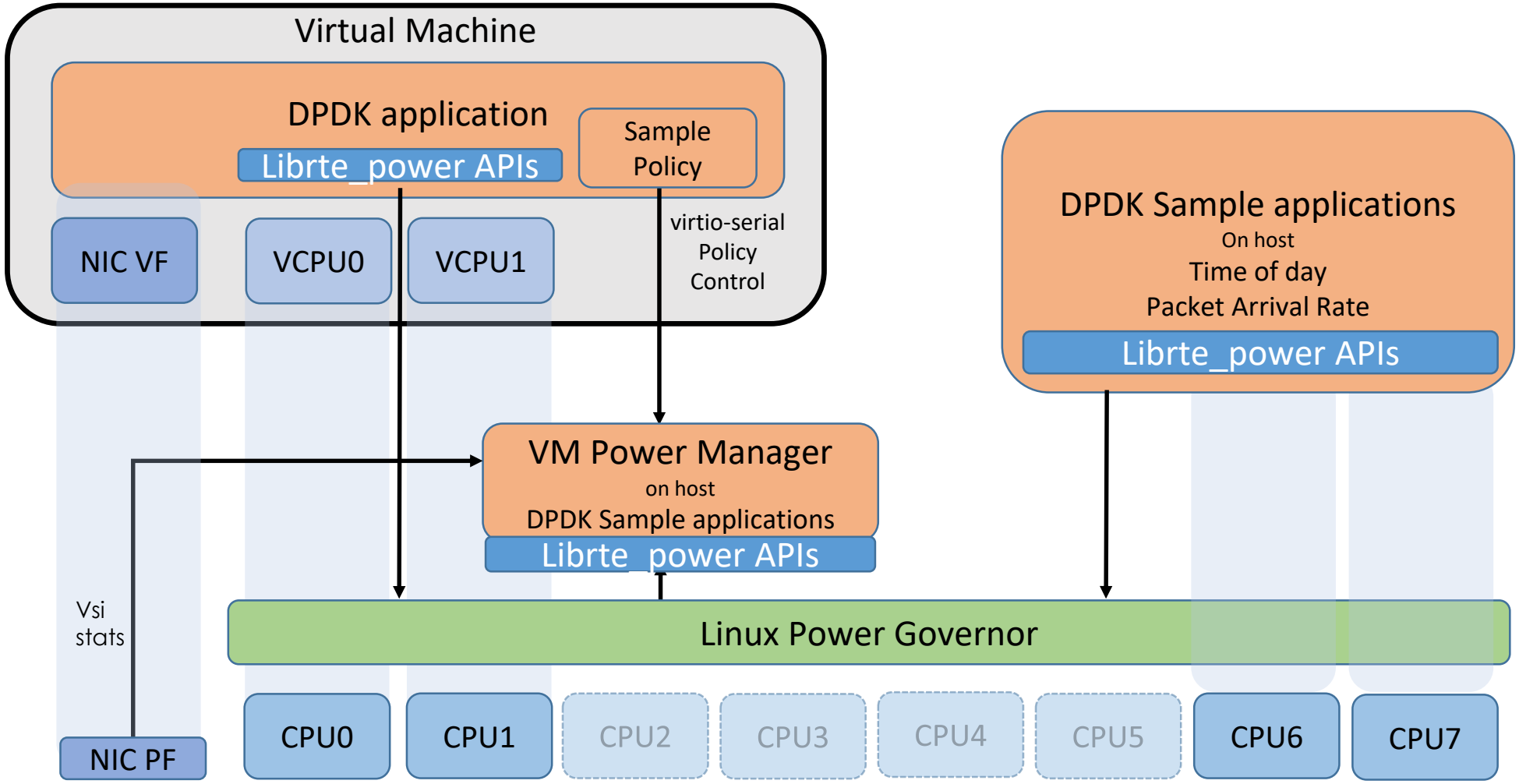
DAVID HUNT, CHRIS MACNAMARA

Updates Since Last Time

- Quick reminder…. feature review

- Updates & Discussions to follow
  - How does an application let something know how busy it is?
    - New telemetry mechanism target polling workloads
  - How do we get more performance for the same power?
    - New hardware mechanisms for giving frequency where it’s needed

Existing DPDK Power Capabilities



Many use cases, support for direct control, virtualized architecture

Usage Options

- Bare Metal – librte_power APIs
  - Turn up/down frequency to save energy, performance via
- Virtualized Applications
  - App in VM sends change frequency command to host (vm_power_manager)
  - App in VM sends policy (time of day) to host (vm_power_manager)
- Virtualized Environment with no Application Instrumentation
  - Using CPU counters, can tell idle or busy, not how busy…

Docs on the way :
https://dpdk-power-docs.readthedocs.io/en/latest/sample_app_ug/vm_power_management.html

Existing DPDK Power Features

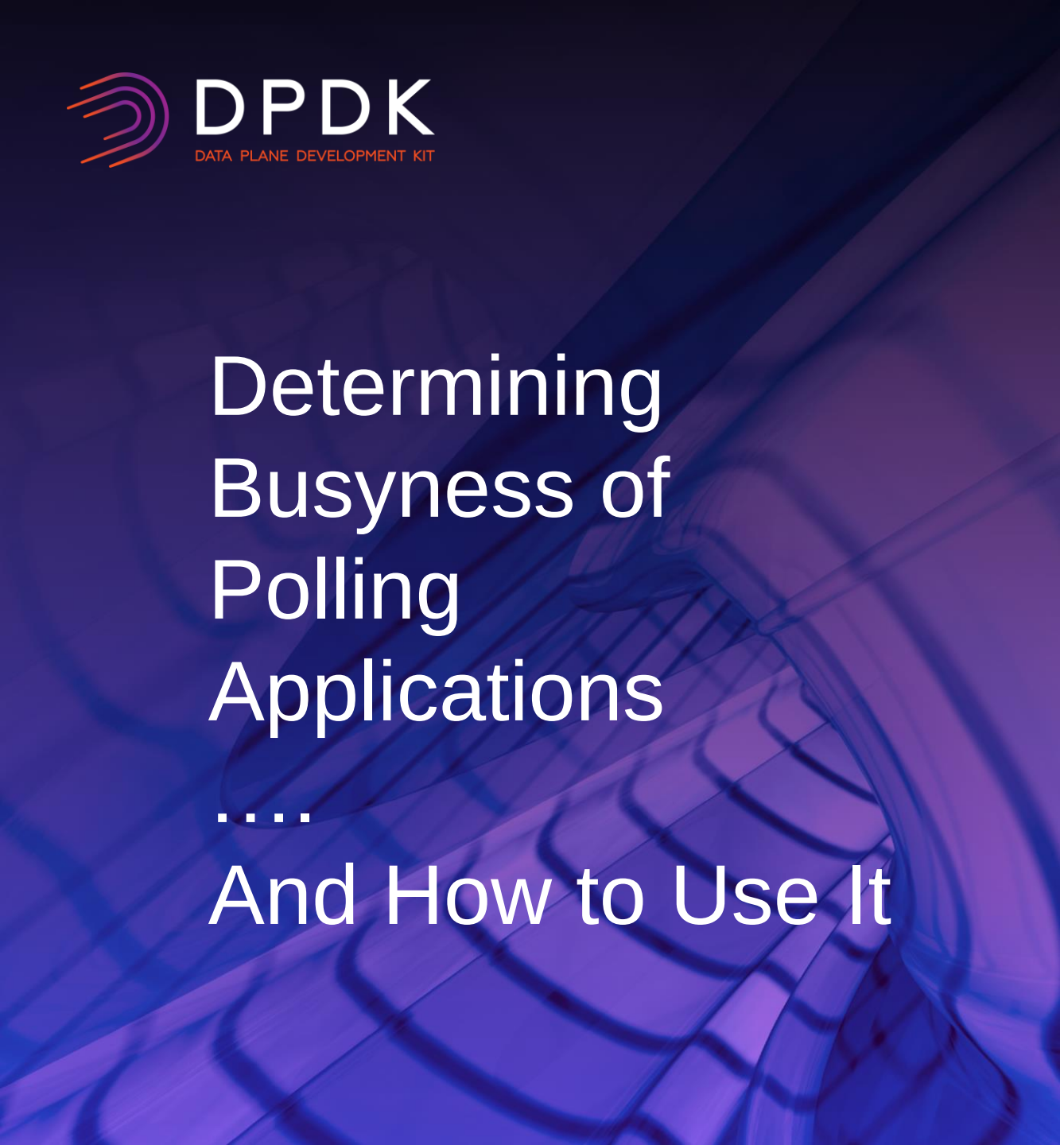
Challenge / Problem	DPDK Solution / Status
L3fwd power using C states (updates coming)	Sample app
Traffic always running, always on cores. Save power when low traffic, boost when busy.	Added core Frequency State APIs including Turbo Boost
Virtualized Software Architecture: High latency of direct requests to change frequency	Move to policy based control: Time of day / Packet Arrival Rate
App Agnostic mechanism to detect when DPDK is 100% polling and no packets or work	Sample code: Branch prediction ratio used as trigger to detect idle -> modify power
Pin DPDK threads/lcores to high priority cores	Pinning relevant workloads to Turbo Cores
Power Policies for Containers	FIFO interface to Power Manager that accepts policies via JSON

Librte_power APIs and Sample Apps

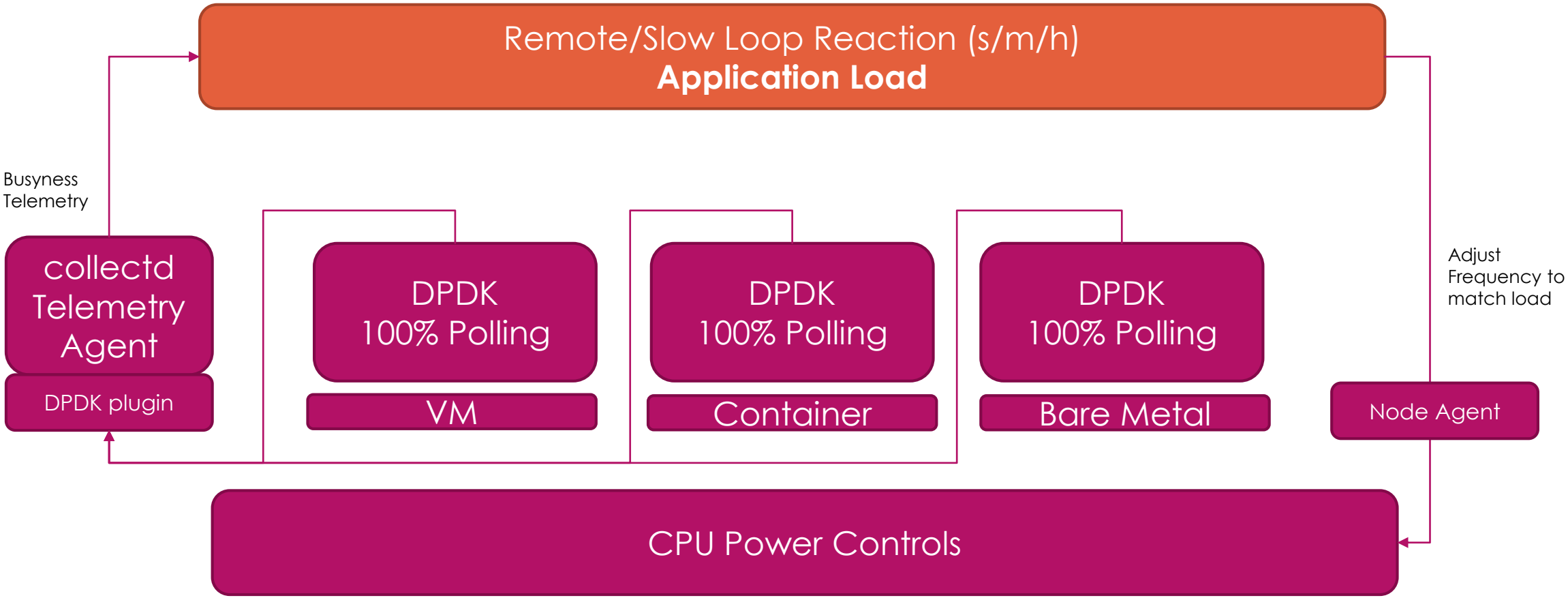
New DPDK Features Since Last Time!

Challenge / Problem	DPDK Solution / Status
Exposing how busy an application is to external systems, no standard method	Introduce new telemetry for busyness metrics, patch to DPDK, collectd
Some workloads need more performance to help balance a multi-core workload	Add support for Intel® Speed Select Technology – Base Frequency – pinning to high priority cores
Enhanced security around Container communication to Power Manager	Updates completed

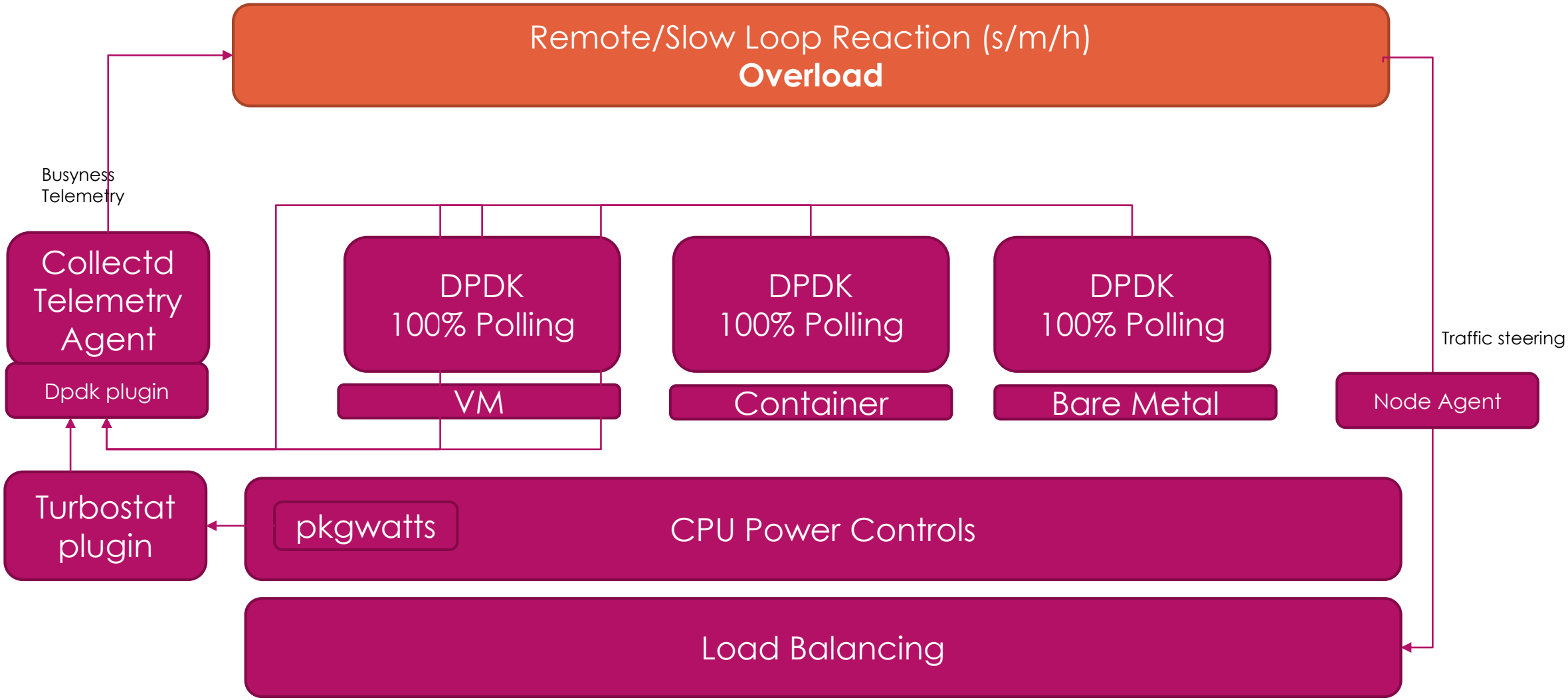
New triggers and capabilities enabling new use cases


DPDK
DATA PLANE DEVELOPMENT KIT
Determining Busyness of Polling Applications
....
And How to Use It

1. Busy Indication Used to Save Energy



2. Busy Indication to Detect Overload



Pushed Patches To Support This (telemetry)

- Released as part of DPDK 19.08
- In addition to per-port metrics, telemetry lib now has global metrics per app (busyness, number of polls, etc.).
- L3fwd-power implements example algorithm to demonstrate populating the new metrics.
- dpdk-telemetry.py now shows the new global stats
- New plugin for “collectd” being upstreamed to be able to view DPDK telemetry.

@development
Application implements algorithm for it’s own ‘busyness’

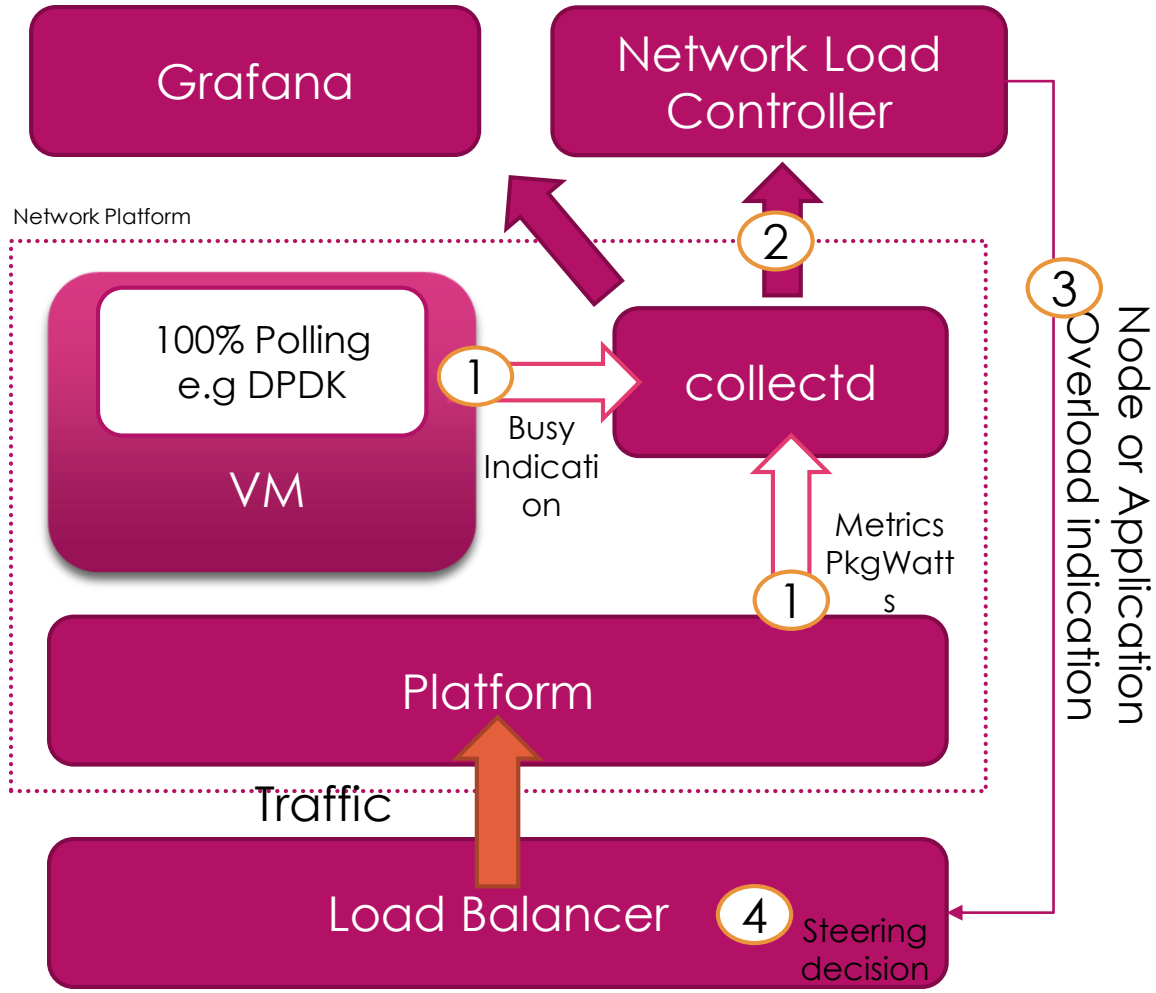
@Init
Application calculates what constitutes 100% busy

@run
Application populates relevant metrics

@run
Third party app pulls metrics (collectd, dpdk_telemetry.py)

Allowing applications publish how busy they are

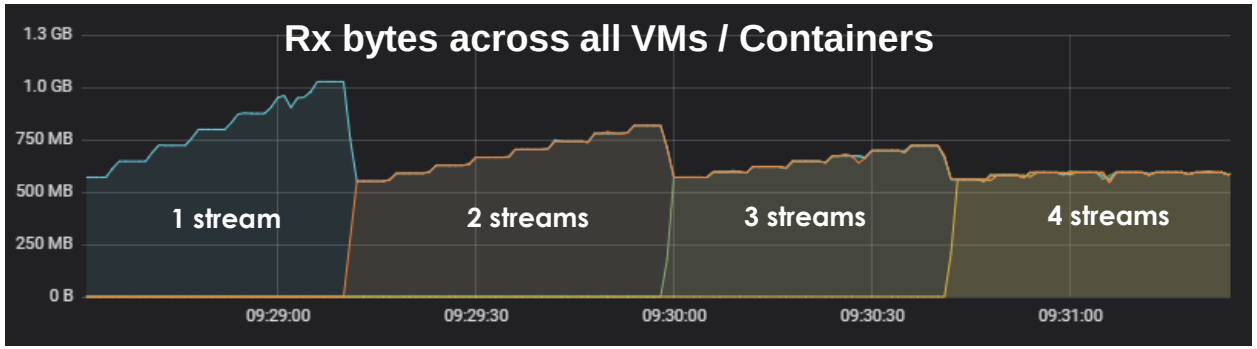
- Patch to telemetry library to allow for global stats, not just port specific
- Patch to l3fwd-power to add telemetry mode to publish busyness
- New plugin for collectd to read this telemetry
- White paper in progress to demonstrate use case for putting it all together



Use telemetry to make informed decisions

Use case for busyness telemetry

- Alert generated when busyness goes over 80%
- Network Load Controller splits traffic on each alert
- Each traffic stream goes to different destination
- Load balancing takes effect



Orchestration now DPDK Busyness visibility


DPDK
DATA PLANE DEVELOPMENT KIT
Maximising
Performance
within the same
Power Envelope

Performance Prioritization Opportunity

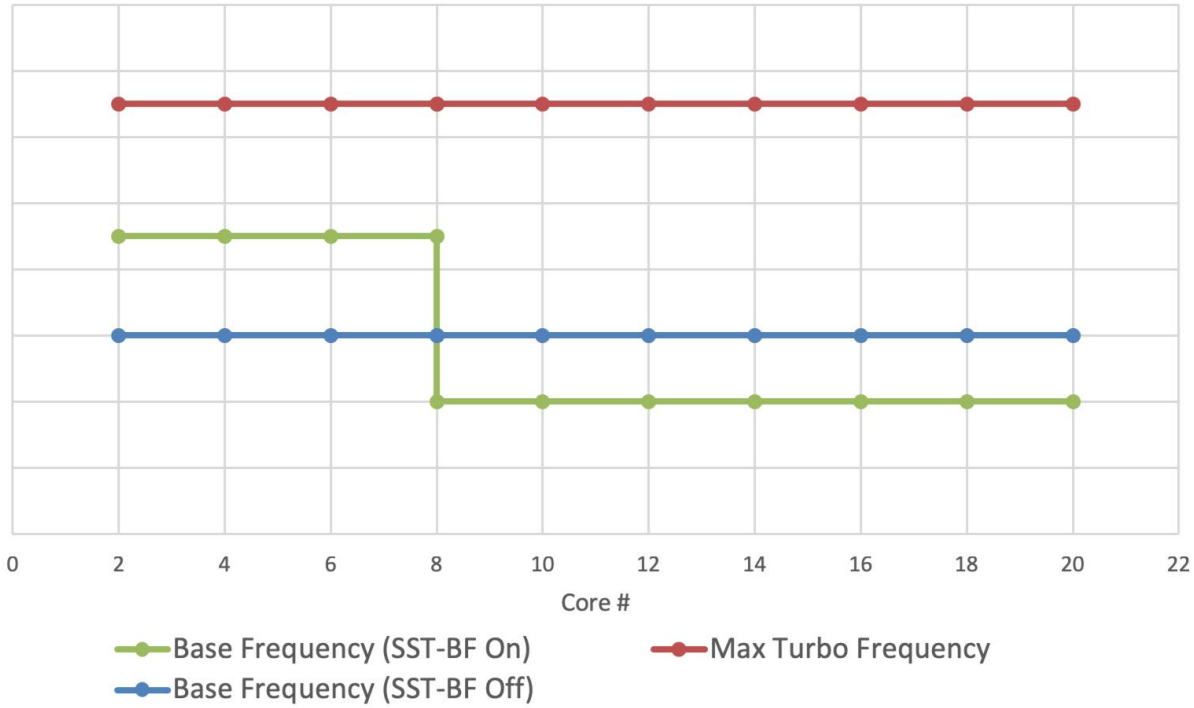
DPDK Based Networking Workloads

- Unbalanced workloads – NFV Data Plane or Control Plane, OVS
- Pipeline software architectures
- Frequency bound workloads
- Priority threads for run to completion
- Packet distribution / workload distribution in SW
- PMD consolidation

Prioritization?

Re-balancing power & frequency to enhance DPDK performance

- A CPU mode allows SW to configure an asymmetric core frequency.
- The placement of key workloads on higher frequency enabled cores can result in an overall system workload increase as compared to deploying the CPU with symmetric core frequencies.
- 6-8 High Priority cores, depending on CPU.
- Intel® SST-BF allows this configuration



Unlock performance bottlenecks

Application Interface

- Additional bit returned by rte_power_get_capabilities indicating High Priority cores:

```
struct rte_power_core_capabilities {
        RTE_STD_C11
        union {
                uint64_t capabilities;
                RTE_STD_C11
                struct {
                        uint64_t turbo:1;
                        uint64_t priority:1;
                };
        };
};

typedef int (*rte_power_get_capabilities_t)(unsigned int lcore_id,
                struct rte_power_core_capabilities *caps);

extern rte_power_get_capabilities_t rte_power_get_capabilities;
```

@init
Application queries capabilities of each available core

@Init
Application pins critical workloads to high priority cores

@run
Application runs as normal

Implemented in l3-fwd-power sample

No DPDK Application or Workload Mods

- Just launch…using EAL launch to map lcores to physical cores

- It was hidden (to us anyway)
- https://doc.dpdk.org/guides/linux_gsg/linux_eal_parameters.html

- `--lcores <core map>`

  Map lcore set to physical cpu set

  The argument format is:

  ```
  <lcores[@cpus]>[<,lcores[@cpus]>...]
  ```

  Lcore and CPU lists are grouped by `(` and `)` Within the group. The `-` character is used as a range separator and `,` is used as a single number separator. The grouping `()` can be omitted for single element group. The `@` can be omitted if cpus and lcores have the same value.

Thank You

Chris MacNamara (chris.macnamara@intel.com)
David Hunt (david.hunt@intel.com)

Acknowledgements

Lee Daly, Reshma Pattan, Anatoly Burakov, Liang Ma

+ “Collectd” team https://github.com/collectd