Platform Metrics Collection Guide for Anomaly Detection

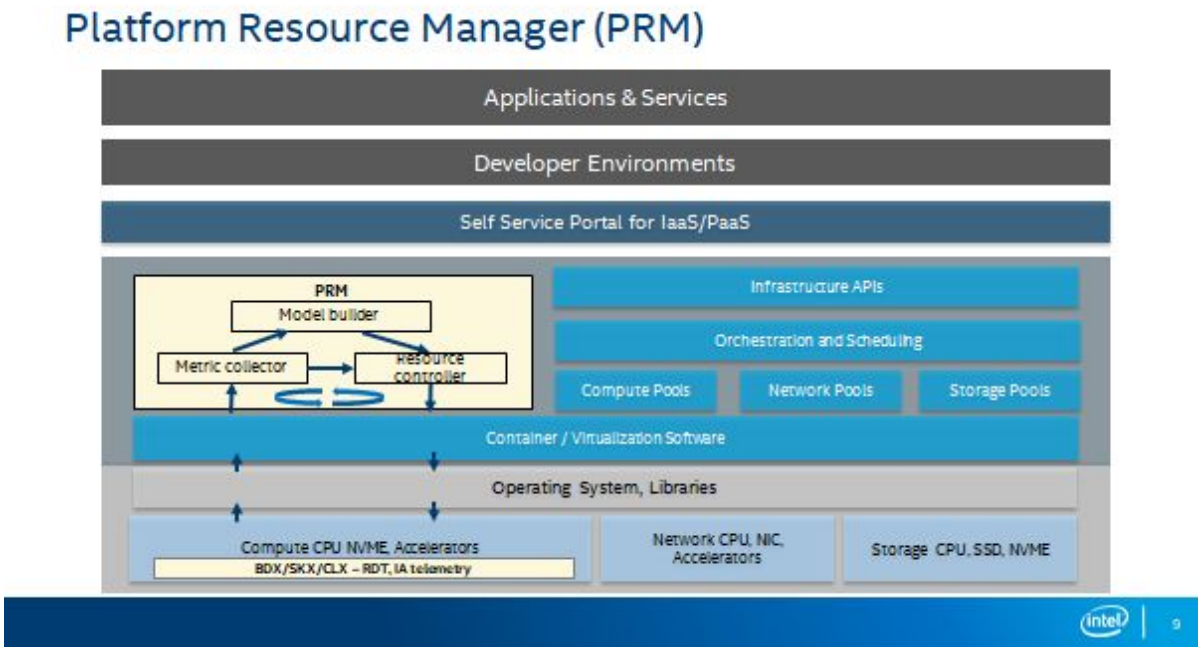
(Workload fingerprint analysis for collocated cloud workloads)

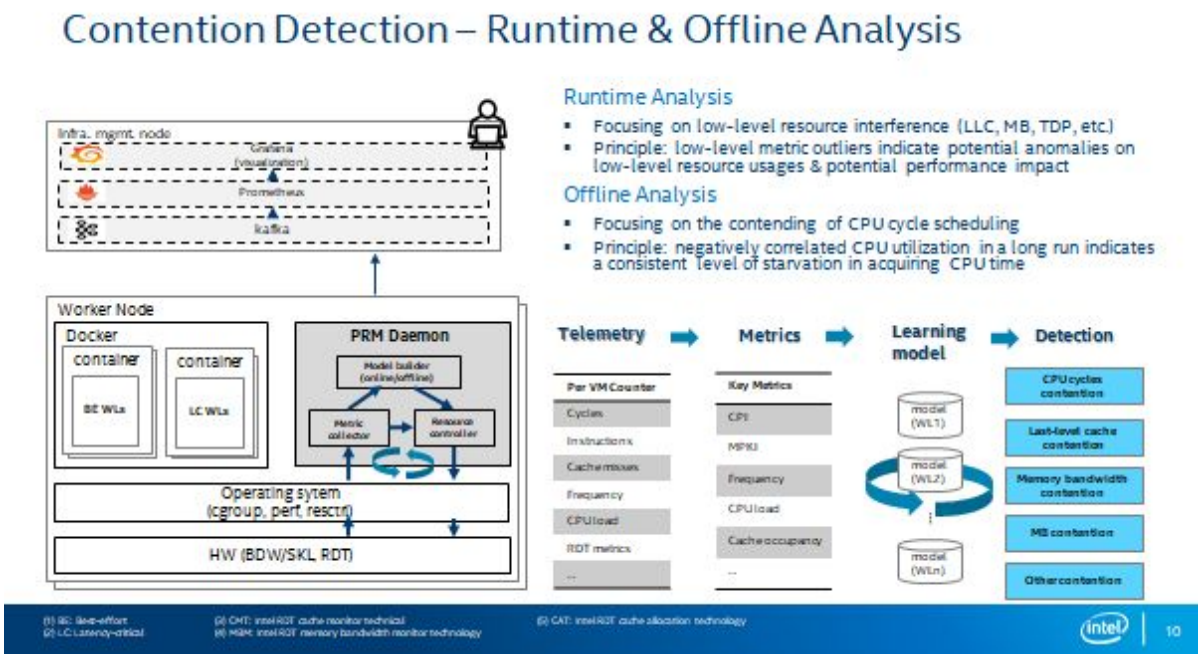
Intel® Platform Resource Manager

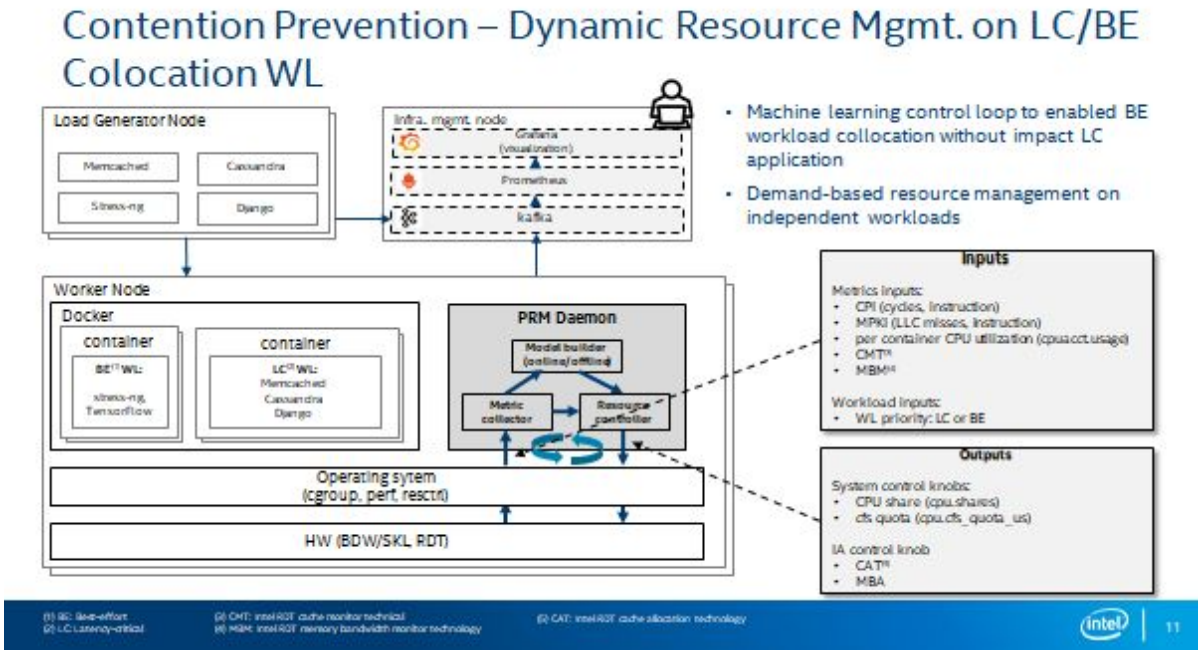
Intel® Platform Resource Manager (Intel® PRM) is a suite of software packages to help you to co-locate best-efforts jobs with latency-critical jobs on a node and in a cluster.

PRM is an open source project published in github:
https://github.com/intel/platform-resource-manager

PRM Overview







Platform Metrics Collection

Collect per container (or VM) level platform metrics.

1. Instructions
    - **Platform metrics:** instructions
    - **Description:** CPU instructions retired
    - **Metrics type:** Standard Per Hardware Event
    - **Platform Requirement:** IA platform
    - **Recommended sampling period:** 20s (need record timestamp)
    - **Collection level:** per container, (or per VM)
    - **How to collect:** PerSysCall
    - **Reference implementation:** data collection for container level data collection. Reference code implementation is available in PRM open source project
        - pgos/main.go
        - pgos/perf.c
2. Cycles
    - **Platform metrics:** Cycles
    - **Description:** Unhalted CPU cycles
    - **Metrics type:** Standard Per Hardware Event
    - **Platform Requirement:** IA
    - **Recommended sampling period:** 20s (need record timestamp)
    - **Collection level:** per container, (or per VM)
    - **How to collect:** PerSysCall
    - **Reference implementation:** data collection for container level data collection. Reference code implementation is available in PRM open source project
        - pgos/main.go
        - pgos/perf.c
3. Last-level Cache Misses(LLC Misses)
    - **Platform metrics:** LLC Misses
    - **Description:** Last level cache misses
    - **Metrics type:** Standard Per Hardware Event
    - **Platform Requirement:** IA
    - **Recommended sampling period:** 20s (need record timestamp)
    - **Collection level:** per container, (or per VM)
    - **How to collect:** PerSysCall
    - **Reference implementation:** data collection for container level data collection. Reference code implementation is available in PRM open source project

        - pgos/main.go
        - pgos/perf.c
4. Stalls Memory Load
    - **Platform metrics:** Stalls memory load
    - **Description:** Stalls while memory subsystem has an outstanding load
    - **Metrics type:** Perf raw event
    - **Platform Requirement:** IA
    - **Recommended sampling period:** 20s (need record timestamp)
    - **Collection level:** per container, (or per VM)
    - **How to collect:** PerSysCall
    - **Reference implementation:** data collection for container level data collection. Reference code implementation is available in PRM open source project
        - pgos/main.go
        - pgos/perf.c
5. Last-level Cache Occupancy (LLC Occupancy)
    - **Platform metrics:** LLC Occupancy
    - **Description:** Last level cache occupancy
    - **Metrics type:** resctrl, kernel 4.14+ /sys/fs/restctrl
    - **Platform Requirement:** BDX (E5 v4), SKX (E5 v5)
    - **Recommended sampling period:** 20s (need record timestamp)
    - **Collection level:** per container, (or per VM)
    - **How to collect:** libpqos from https://github.com/intel/intel-cmt-cat
    - **Reference implementation:** data collection for container level data collection. Reference code implementation is available in PRM open source project
        - pgos/main.go
        - pgos/perf.c
6. MB Local
    - **Platform metrics:** MB Local
    - **Description:** Memory bandwidth local
    - **Metrics type:** resctrl, kernel 4.14+ /sys/fs/restctrl
    - **Platform Requirement:** BDX (E5 v4), SKX (E5 v5)
    - **Recommended sampling period:** 20s (need record timestamp)
    - **Collection level:** per container, (or per VM)
    - **How to collect:** libpqos from https://github.com/intel/intel-cmt-cat

- **Reference implementation:** data collection for container level data collection. Reference code implementation is available in PRM open source project
  - pgos/main.go
  - pgos/perf.c

7. MB Remote
   - **Platform metrics:** MB Remote
   - **Description:** Memory bandwidth remote
   - **Metrics type:** resctrl, kernel 4.14+ /sys/fs/restctrl
   - **Platform Requirement:** BDX (E5 v4), SKX (E5 v5)
   - **Recommended sampling period:** 20s (need record timestamp)
   - **Collection level:** per container, (or per VM)
   - **How to collect:** libpqos from https://github.com/intel/intel-cmt-cat
   - **Reference implementation:** data collection for container level data collection. Reference code implementation is available in PRM open source project
     - pgos/main.go
     - pgos/perf.c
8. CPU usages
   - **Platform metrics:** CPU usages
   - **Description:** CPU utilization (similar to %CPU in Linux Top, one full logical CPU is 100)
   - **Metrics type:** cpuacct.usage in container cgroup
   - **Platform Requirement:** BDX (E5 v4), SKX (E5 v5)
   - **Recommended sampling period:** 20s (need record timestamp)
   - **Collection level:** per container, (or per VM)
   - **How to collect:** cgroup /sys/fs/cgroup interface
   - **Reference implementation:** data collection for container level data collection. Reference code implementation is available in PRM open source project
     - pgos/main.go
     - pgos/perf.c

Platform Metrics Sample

Per server platform metrics data sample

time	Container name (or container ID)	instructions	cycles	LLC misses	cpu usages	LLC occupancy (KB)	MB local (MB)	MB remote (MB)	stalls memory load
1546409448	memcache_workload_1	132213585	107263102	1581	0.366631318	0	0	0	19843761
1546409448	cassandra_workload	42708877	109007525	29544	0.439034745	1496	0.215945513	0	35639799
1546409468	memcache_workload_1	132887360	111667688	506	0.373262533	0	0	0	21467449
1546409468	cassandra_workload	42660489	109619992	19671	0.430393757	616	0.101362179	0	35702083
1546409488	memcache_workload_1	133915699	112247124	158466	0.369151724	88	0.735977564	0	21890032
1546409488	cassandra_workload	24675096578	31035554633	15758276	77.67602618	52888	197.0304487	0	10626335944
1546409508	memcache_workload_1	60431780403	70897296563	18066269	163.8985873	2992	96.06490385	0	22494300730
1546409508	cassandra_workload	297828000000	373491000000	182296766	955.0617583	48664	2418.695513	0.0132212	132808000000
1546409528	memcache_workload_1	64743808764	85413565352	19794795	199.5140946	3256	83.49158654	0	28351243114
1546409528	cassandra_workload	300510000000	382217000000	189595182	975.6061917	48136	2519.017628	0.0044071	137833000000
1546409548	memcache_workload_1	64789108022	85414061490	19929785	199.4831878	1584	80.50360577	0	28342538490
1546409548	cassandra_workload	315443000000	392463000000	221314651	998.1253688	51480	2599.151042	0.0088141	141543000000
1546409568	memcache_workload_1	64897533449	85436718966	20252248	199.5600436	1496	90.80288462	0	28329982333
1546409568	cassandra_workload	339613000000	410020000000	278182255	1041.134183	50336	3174.116987	0.0132212	145983000000
1546409588	memcache_workload_1	65213440239	85315867103	18243372	199.3454832	2200	77.35697115	0	28258909408
1546409588	cassandra_workload	247707000000	299750000000	165183631	764.7531161	50600	2307.307292	0.0176282	105537000000
1546409608	memcache_workload_1	65221920722	85234898247	19099109	199.224656	5280	90.70592949	0	28091716318

Server workload configuration

In order to analysis per workload characteristic, a workload configuration is require for each of servers where the platform metrics are collected. You can can provide a workload configuration file in json or other format.

Workload configuration file should include below information:

1. **Container name or container ID:** container name or container ID should be correlated with the “container name” or “container ID” field in platform metrics data.
2. **CPUS:** Assigned CPU count of one container (or VM):
3. **Workload type:** Best-Efforts, Latency-Critical

Workload Configuration Sample

The following is a sample workload configuration file for one server.

```
{
    "cassandra_workload": {
        "cpus": 10,
        "type": "latency_critical"
    },
    "django_workload": {
        "cpus": 8,
        "type": "latency_critical"
    },
    "memcache_workload_1": {
        "cpus": 2,
        "type": "latency_critical"
    },
    "memcache_workload_2": {
        "cpus": 2,
        "type": "latency_critical"
    },
    "memcache_workload_3": {
        "cpus": 2,
```

```
        "type": "latency_critical"
    },
    "stress-ng": {
        "cpus": 2,
        "type": "best_efforts"
    },
    "tensorflow_training": {
        "cpus": 1,
        "type": "best_efforts"
    }
}
```