May 2023

RESOURCE-CENTRIC AND CONTEXT AWARE ENHANCED METRIC COLLECTION FOR ENRICHED APPLICATION PERFORMANCE VISIBILITY

Nagendra Kumar Nainar

Dave Zacks

Akram Sheriff

Rajesh I. V

RESOURCE-CENTRIC AND CONTEXT AWARE ENHANCED METRIC COLLECTION FOR ENRICHED APPLICATION PERFORMANCE VISIBILITY

AUTHORS:
Nagendra Kumar Nainar
Dave Zacks
Akram Sheriff
Rajesh I V

ABSTRACT

In many cloud computing environments, cloud-native applications can be decoupled from the underlying hardware that operates such applications. Presented herein are techniques through which capabilities and metrics collected from a virtual/physical compute resource can be used to identify the presence of different accelerators and their capabilities in real time. The metrics that can be collected from within an application can be extended to include execution context awareness, which may be used to suggest a resource profile for the application. Further, holistic visualization of various metrics can be provided in some instances based on the accelerator from where a service is being executed.

DETAILED DESCRIPTION

With the evolution of cloud computing, more and more apps are becoming cloud-native that offers a high level of flexibility. However, such decoupling of the software from the hardware raises performance challenges that are often addressed by different types of accelerators. Figure 1, below, illustrates an example cloud-native environment in which certain application functionality can be offloaded to an accelerator.

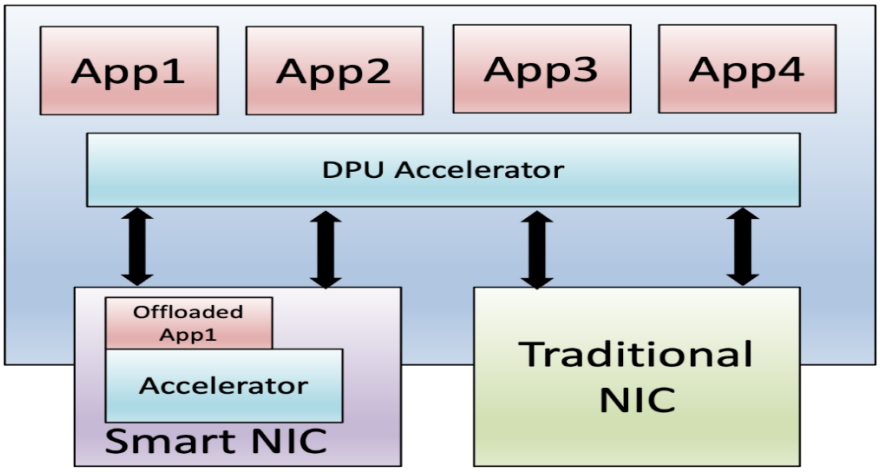

Figure 1: Example Cloud-Native Application Offloading

Any cloud-native application may be hosted on a server with a data processing unit (DPU) accelerator that is connected with a smart network interface card (sNIC) capable of offloading services for acceleration. In such a case, depending on the various factors such as incoming interface, type of the user, current load, etc., the request may be processed/executed by an accelerator in the sNIC, by a DPU, or within the software itself.

Any application agent or the instrumentation code injected into an application may collect metrics without context awareness of where a service is offloaded/executed, which may not offer a holistic view of the actual performance outcome of the microservice. Consider an illustrative example of such a limitation.

For example, consider that service1, while hosted/executed by a kernel component, may be capable of handling 500 concurrent connections (or service request) as one metric. Whereas the same service, while offloaded to a DPU, may have 1000 concurrent connections as a metric and, while offloaded to a sNIC, may have 800 concurrent connections. Thus, the location of offloading can result in different metrics. Yet, current application performance monitoring (APM) tools are often not capable of collecting metrics with the granular level of insights that may be gleaned from identifying different service offloading locations.

For example, in many current implementations, offloading application functionality to an accelerator, such as a DPU, sNIC, etc. is more akin to creating relevant state entries in the hardware, which may not include observability functionality. Thus, current implementations often lack the ability to not just collect but also differentiate an accelerator

from where the metrics are collected for a holistic visibility without compromising with the granularity. Service offload metrics may be collected, but such metrics lack the ability to collect actual metrics directly from the accelerators at which applications are offloaded.

In order to address such issues, this proposal provides techniques through which capabilities and metrics collected from a virtual/physical compute resource can be used to identify the presence of different accelerators and their capabilities in real time. Stated differently, techniques herein provide for the ability to collect service offload metrics from the accelerators directly and to differentiate such metrics. In some instances, an application performance management agent can be extended/leveraged to collect such metrics. The metrics collected from within an application can be extended to include execution context awareness (e.g., DPU vs sNIC vs kernel vs normal NIC vs Tensor Processing Unit (TPU), etc.), which may be used to suggest a resource profile for the application. Further, holistic visualization of various metrics can be provided in some instances based on the accelerator from where a service is being executed. A resource profile can be suggested for an application based on performance determined from the collected capabilities/metrics.

Figure 2, below, illustrates an example architecture involving an application performance management platform and various agents through which various capabilities and metrics can be gathered for various applications.

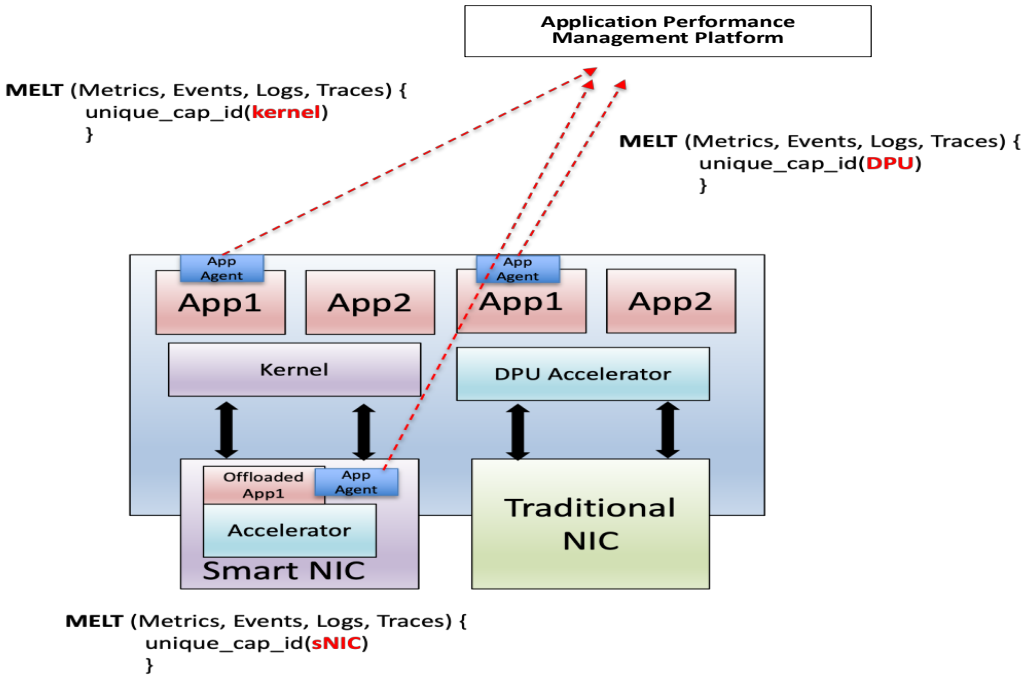

Figure 2: Example Metric Gathering Architecture

For the architecture of Figure 2, unique IDs can be assigned and associated to different metrics collected from different execution points based on the capabilities of the virtual/physical resource where an application is hosted. For example, as illustrated in Figure 2, each application executed by the kernel and each offloaded version of the application being executed on the DPU and the sNIC will each have a unique ID associated therewith. Agents for the applications can collect metrics (e.g., concurrent calls, latency, type, etc.) and push the metrics to the application management platform by associating the unique ID to metrics, events, logs, and traces (MELT) data. An example MELT JavaScript Object Notation (JSON) container may be formatted as:

```
"events": [
 {"Service_Offload": Yes
  "name": "",
  Unique_ID: "xyz",
  "message": "OK",
  "timestamp": "2023-1-22 16:04:01.209512872 +0000 UTC"
 }
]
}
```

Upon gathering metrics for applications, the metrics can be visualized utilizing a visualization tool, as shown below in Figure 3.

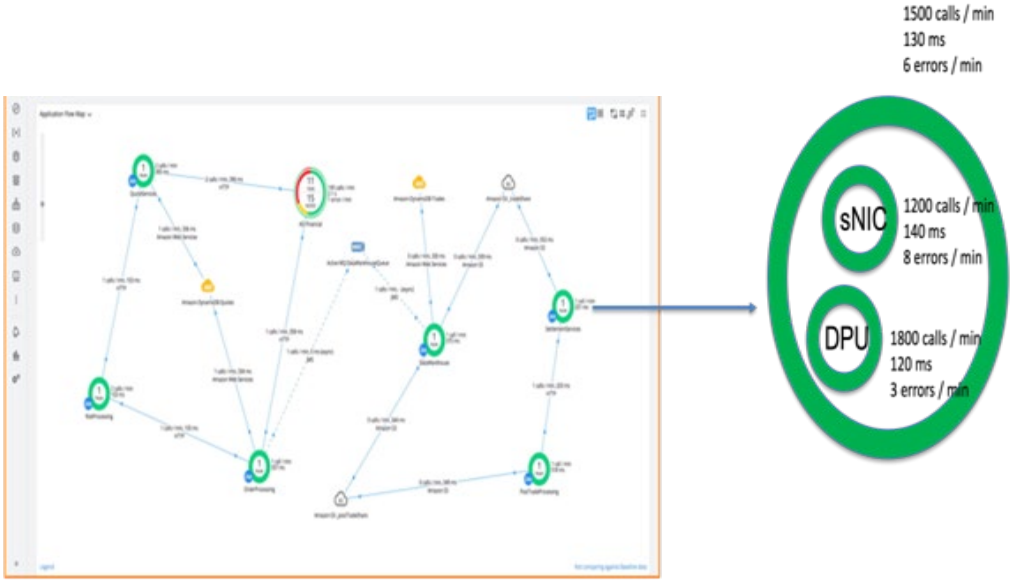

Figure 3: Example Metric Visualization

As shown in Figure 3, a flow map that can be generated by the application performance management platform can be augmented to highlight application metrics on a per-resource basis (e.g., sNIC vs DPU vs Kernel execution versus TPU or Normal NIC) along with a cumulative derived measurement.

The baselined cumulative measurement for an application can used as a reference point for the overall performance, while metrics from an offloaded version of the application can be used as reference point to detect any deviation on a given accelerator/resource in real time. In one instance, such measurements may facilitate anomaly detection for an application.

Accordingly, techniques presented herein may provide for the ability to indicate an offload/accelerator capability of a host and an application in order to indicate or otherwise differentiate whether a service is offloaded versus regular (in the metrics) along with providing additional details, such as an offload location identifier, which can be used to differentiate the location from which different application metrics are collected.