Adaptive Applications for IBM Power using PowerVP

Robert Schmid

Abstract—**One of the main advantages of system virtualization are the reduced costs for acquiring and maintaining server hardware. Because a typical workload on average only consumes a fragment of the available resources, the hardware utilization can be improved by consolidating multiple workloads onto one virtualization host.**

During peak usage times however, the host machine must still be able to provide all guests with sufficient resources. The PowerVM hypervisor available on IBM Power Systems provides a flexible model to dynamically assign additional resources to virtual machines when needed. In order for the applications to efficiently use the dynamically assigned resources during load peaks, it is necessary for them to be aware of the specific topology of the processor cores and memory that are currently available. In addition, detailed processor metrics can help applications to detect inefficient behavior at runtime and to react accordingly. During this seminar, a prototype of such a mechanism was developed and evaluated by using metrics provided by IBM PowerVP. Several performance counters are recorded into a SQL database, a custom metric is derived from this data and a warning is presented to the user when a critical situation occurs.

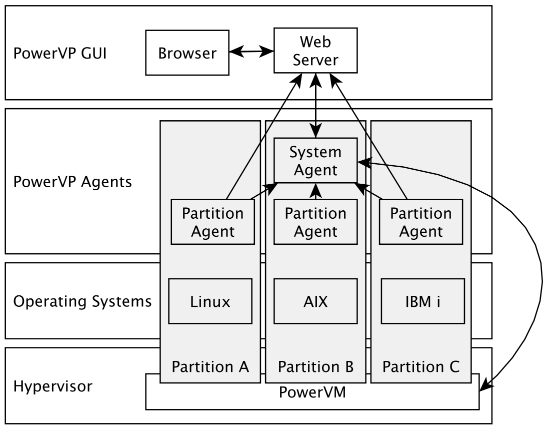


Fig. 1. Overview about PowerVP components.

I. Introduction

An important goal when operating virtualized systems is to increase the average utilization on each host machine while still providing enough reserved capacity to accomodate peak load situations. The IBM Power architecture is well suited for this purpose: So-called *Scale-Up* systems, that combine multiple CPU sockets with memory attached to each of them, are especially common when a high density of virtual machines per host is required. The theoretical benefit of running many workloads on a single high-end machine is the lower expected variance in overall utilization, which allows for a higher average utilization and a smaller share of reserved capacity without risking to violate service-level agreements (SLAs) [1]. IBM PowerVM is the built-in hypervisor that is used to run and manage virtual machines (*partitions*) on these systems.

A *dedicated* partition in PowerVM has a fixed size in terms of CPU cores and memory. In this case the hypervisor reserves a fixed set of physical resources that will always belong to the partition. Alternatively, a set of physical resources can be assigned to a *Shared Processor Pool*. *Shared* partitions that belong to a Shared Processor Pool are provided with a minimum amount of CPU time slices depending on their *entitlement*, and may receive additional resources on demand if they run in *uncapped* mode. If this model is compatible with the application requirements in terms of technical support as well as availability, it helps to further increase the application density on the physical system, thereby decreasing costs for hardware and operation.

Seminar Resource Management On Power
Winter Term 2016/2017
Topic selection and advice: Felix Eberhardt and Prof. Dr. Andreas Polze

Because of the architecture of Scale-Up systems, direct access to certain memory regions is not possible from every CPU core. Instead, it has to communicate with a remote core that the respective memory is directly attached to, which leads to increased latency and reduced data throughput (non-uniform memory access, *NUMA* architecture).

During high-load situations, the hypervisor may be forced to re-distribute time slices of physical cores among the partitions in a processor pool. This in turn can lead to a situation where a guest's thread that is working with a large volume of memory (i.e. performing a table scan in a database) faces a significant impact on performance due to the NUMA architecture of the physical system. Although the hardware helps to mitigate these performance issues by providing memory pre-fetchers, caches and memory migration technologies, the application itself may still want to react by re-distributing or deferring work items.

In order to detect these situations, it is possible to use data provided by the built-in performance measurement units of the POWER processors. One suitable metric is the amount of CPU instructions in a given time interval that were stalled due to remote memory accesses. IBM PowerVP (*IBM PowerVP Virtualization Performance*) provides system administrators with a graphical tool to monitor these metrics for partitions on a PowerVM host. By recording and analyzing the performance counters, custom metrics can be derived and used to create adaptive applications.

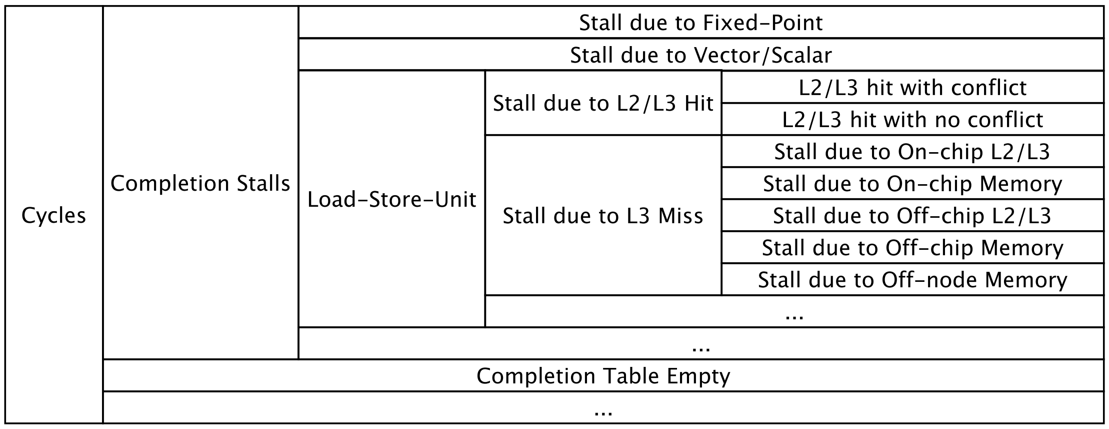

Cycles
Completion Stalls
Stall due to Fixed-Point
Stall due to Vector/Scalar
Load-Store-Unit
Stall due to L2/L3 Hit
L2/L3 hit with conflict
L2/L3 hit with no conflict
Stall due to L3 Miss
Stall due to On-chip L2/L3
Stall due to On-chip Memory
Stall due to Off-chip L2/L3
Stall due to Off-chip Memory
Stall due to Off-node Memory
...
...
Completion Table Empty
...

Fig. 2. Excerpt of the POWER8 CPI stack [2].

II. IBM PowerVP

PowerVP is a performance analysis tool provided by IBM which can be used to monitor resource utilization and performance indicators on PowerVM-based systems. It provides machine-wide statistics as well as detailed per-partition metrics that can be used to diagnose performance issues of the running workloads. Furthermore, administrators can learn when to use the *Dynamic Platform Optimizer* (DPO, [3]) to resolve inefficient resource assignments. All metrics can be recorded and played back for later analysis e.g. during load peaks or performance tests. Throughout this seminar, PowerVP version 1.1.3 Service Pack 6 was used, running on an Ubuntu 16.04 (Linux 4.4) partition. [4]

A. PowerVP Architecture

A PowerVP installation consists of one or more system-level agents, one or more partition-level agents and a Java-based web server for hosting the user interface (see figure 1). In earlier versions, a desktop client was provided instead of the browser-based UI.

All agents can be installed on IBM i, AIX, Linux and VIO operating systems. In order to improve reliability, system-level agents can be replicated on multiple partitions. To obtain system-wide statistics, these partitions need a special permission set in the PowerVM Hardware Management Console (HMC). System agents provide information about the overall system topology (nodes, sockets and cores), utilization data of individual cores and interconnects as well as an overall *CPI* (cycles per instruction) counter.

Partition-level agents are needed to collect more detailed information about the performance characteristics of a particular partition. Each partition-level agent connects to a system-level agent if it doesn't act as a system-level agent itself. Besides information about the utilization of individual hard disks and network adapters, a detailed breakdown of the average CPI number is provided which corresponds to the *Power CPI Stack* (see figure 2). A low CPI number close to one is equivalent to an efficient program execution that makes good use of the prefetching and pipelining capabilities of the processor. PowerVP allows to determine the functional unit or type of memory stall that accounts for high CPI numbers.

The user interacts with a PowerVP installation by connecting to a system-level agent through the PowerVP GUI. Afterwards, the partition-level agents are enumerated and a separate connection is established to each of them. Specifically, the PowerVP GUI server creates TCP connections to the agents and communicates with them using a proprietary binary format. The obtained information is forwarded to the user's browser using a WebSocket connection.

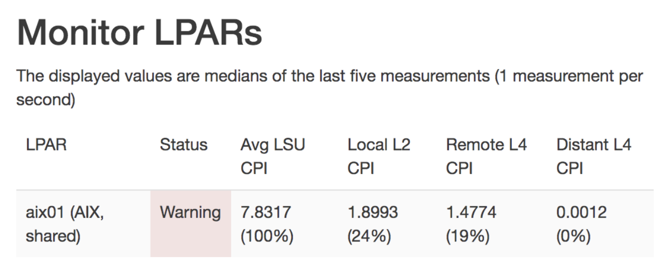
B. Data sources of PowerVP

On Linux, the PowerVP agent consists of a kernel-mode driver and a user-mode daemon. Because the driver's source code is distributed with PowerVP and by default emits detailed debug messages, the interaction of the two components can be easily observed.

The daemon mostly handles communication with other agents and the GUI server. Whenever system-wide metrics are requested, it interacts with the driver through the */dev/powervp* device. The following hypercalls belong to the *24x7* and *GPCI* interfaces and are used by the driver to obtain the requested data from PowerVM [5]:

- H_24x7Catalog (0xF078)
  Provides meta data and descriptions of available 24x7 performance counters.
- H_24x7Data (0xF07C)
  Returns values of the requested 24x7 performance counter. Most counters are not specific to a particular partition. Counter values are easily accessible through the sysfs on recent Linux kernels.
- H_GetPerformanceCounterInfo (0xF080)
  This legacy interface was already available in pre-POWER8 architectures and is partially specified in the *Linux on Power Architecture Platform Reference* [6]. However, the performance counters used by PowerVP are not documented.

To query partition-level performance counters, the PowerVP daemon executes Linux' *perf* tool, without the need to perform



Monitor LPARs

The displayed values are medians of the last five measurements (1 measurement per second)

LPAR	Status	Avg LSU CPI	Local L2 CPI	Remote L4 CPI	Distant L4 CPI
aix01 (AIX, shared)	Warning	7.8317 (100%)	1.8993 (24%)	1.4774 (19%)	0.0012 (0%)

Fig. 3. Screen shot of the application prototype.

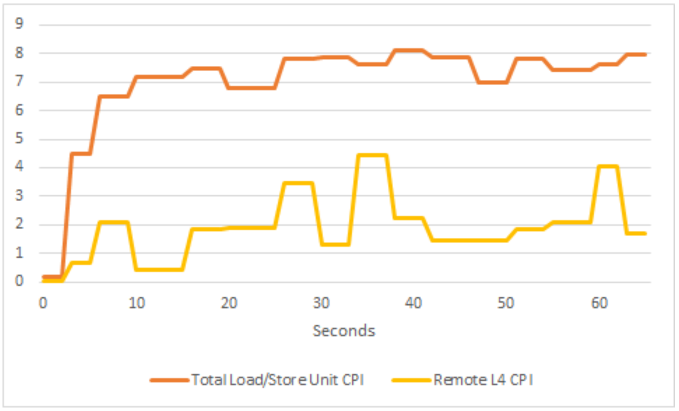


Fig. 4. Graph of CPI metrics when running the pointer chasing benchmark.

hypercalls through the driver module. Because each instance of perf can only measure five performance counters at a time, multiple instances are started in parallel. As a result, the CPI breakdown numbers shown in PowerVP sometimes don't add up to the total CPI number because of slight offsets during measurement.

In conclusion, the performance counters collected on Linux partitions provide a good overview about the general characteristics of a partition. New 24x7 performance counters are easily accessible even without using the PowerVP driver. This will further improve as development of POWER-specific performance monitoring capabilities in Linux continues.

III. Adaptive Application Prototype

The goal of the prototype built during this seminar was to evaluate how POWER8 performance counters can be systematically collected and used to trigger reactive application behavior. It is comprised of several components; the source code is available on GitHub [7].

To be able to flexibly query and analyze the performance counter data, SAP HANA was used as a relational data store and application server. In addition, the prototype contains a component that receives performance counters from PowerVP and forwards the data into the database. A web site with server-side logic is used to query the database and to display a warning whenever a given threshold is exceeded for a monitored partition. Finally, a suitable benchmark was chosen to simulate a load situation.

PowerVP was chosen as the data provider for performance counters because it provides sufficiently precise data for the purpose of this prototype and offers a simple way to access system and partition counters: A client for the PowerVP GUI WebSocket communication protocol can be easily implemented by sending and parsing JSON-based messages. Specifically, a Python script sends data requests to the GUI server every second and stores the incoming data in the HANA database. The database contains three tables to store overall system metrics, individual utilization numbers for each CPU core as well as partition-specific CPI counters.

A JavaScript-based HANA XSJS web-app hosted on the SAP HANA application server acts as a consumer of the data. By defining a suitable metric and threshold, a real application could automatically detect and react to load situations that e.g. cause high CPI numbers in its respective partition. Alternatively, the application could notify the user and offer an option to manually change its behavior. This is simulated in the prototype by showing a red placeholder button combined with a warning, whenever the average Remote L4 stall CPI number exceeds one. To filter outlier values, the median value of the last five measurements is computed. Updated values are retrieved every second by issuing a SQL query against the database.

For testing purposes, a suitable benchmark is needed to artificially raise the Remote L4 stall CPI number by incurring cache misses for memory accesses on a remote NUMA node. The first benchmark considered is the STREAM benchmark published by John D. McCalpin [8]. It measures the memory bandwidth and was configured to allocate large areas of memory that likely span multiple memory nodes and subsequently access these areas from multiple parallel threads. However, running the benchmark turned out to have no effect on the desired CPI metric, probably because of intelligent caching and pre-fetching mechanisms provided by the POWER8 hardware.

Instead, a custom *pointer chasing* benchmark had the desired effect on the performance counter:

1) Allocate an array of elements, where each element contains a pointer to another element and a padding. The array is big enough to span multiple NUMA nodes when the OS allocates the memory.
2) Initialize the elements' pointers, so that each pointer points to another random element and each element is pointed to exactly once (forming a circular graph)
3) Execute the benchmark: Using an OpenMP parallel for-loop let each thread 'chase' the pointers through the array by iteratively accessing and de-referencing pointers in array elements

By randomly accessing different memory regions, it is difficult for the pre-fetchers to predict future memory accesses. This results in the desired stalls when data has to be loaded from remote memory into remote L4 caches.

IV. Future Work

To further explore the potential of live performance counter measurements, a more realistic benchmark and use case should be evaluated: One possibile experiment could be to observe the throughput of a database engine answering a set of pre-defined queries, thereby accessing memory on different NUMA nodes. At the same time, another busy partition in same processor pool causes the database partition to migrate threads across physical cores, in turn decreasing the database's throughput. If the experiment proves that this situation can be detected by observing CPI metrics, a follow-up question could be if the database is able to make use of this information to improve the query throughput.

Capturing longer periods of resource utilization data can also be utilized to derive profiles for applications and partitions, that make predictions about resource usage more accurate. Partitions with complementary profiles can be discovered, which in turn helps to optimize the shared processor pool configuration: Variables such as the number of partitions in the same pool, partitions running in capped or uncapped mode and the number of reserved processing units that are needed to achieve the given availability targets are difficult to determine otherwise. A suitable configuration can again increase utilization and reduces costs for operating the hardware.

V. Conclusion

PowerVP metrics are useful for detecting inefficient resource utilization situations in a partition. However, actually making the right decisions based on these performance counters can be difficult: Because PowerVP does not monitor single applications but entire partitions, it can be hard to draw the right conclusions from the input data. Since the measurements only occur once per second, but the execution times of an application's work items are often on a smaller scale, it may be difficult to establish a feedback loop between application behavior and observable changes in performance counters. Further research is necessary in this direction.

The improved accessibility of more detailed POWER8 performance counters on Linux will simplify the development of adaptive applications. Because the operating system interfaces are at least partially documented, it may not be necessary to build upon PowerVP; directly reading performance counters from the operating system provides a better flexibility to fulfill the specific application requirements.

Having a flexible way to capture and analyze large amounts of performance counter data in a relational database will become even more useful for deriving application usage profiles to guide partition sizing and placement.

References

[1] G. Andrews, *Top Reasons Why You Should Run Linux Workloads on Power*, 2013.

[2] S. K. Sadasivam, "Performance evaluation methodology to the open-power user community to evaluate the performance using the advanced instrumentation capabilities available in the power 8 microprocessor," vol. Open Power Summit, 2015.

[3] IBM, "Dynamic platform optimizer," 2017. [Online]. Available: https://www.ibm.com/support/knowledgecenter/POWER8/p8hat/p8hat_dpoovw.htm

[4] L. Rosca and S. Stefanov, *IBM PowerVP: Introduction and Technical Overview*. IBM, 2015.

[5] C. Schafer, "Add support for power hypervisor supplied performance counters," 2014. [Online]. Available: https://lwn.net/Articles/586454/

[6] IBM, *Linux on Power Architecture Platform Reference*, 2016.

[7] R. Schmid, "Ibm powervp monitoring framework," 2017. [Online]. Available: https://github.com/rs22/powervp-monitoring-framework

[8] J. D. McCalpin, "Stream benchmark," 2017. [Online]. Available: https://www.cs.virginia.edu/stream/