随着越来越多的用户将生产应用迁移到云平台，一些传统IT的运维功能也相应的需要改变，例如监控，备份等等。我们希望通过这一系列的文章来协助用户更好的理解在Azure云平台上实现资源监控的方法。

在今后的系列文章中，我们会详细介绍详细的Azure平台的一些监控服务。由于很多用户以虚拟机方式迁移之前本地数据中心的生产环境，我们就从虚拟机的监控作为切入点。今天的内容就从最基本的了解虚拟机的性能指标开始。

我们知道Azure的宿主机是基于Hyper-V平台，从平台层面，无论运行的是Windows还是Linux的虚拟机，Hyper-V平台都可以针对虚拟机这个对象来提供一定的性能指标。具体的技术实现细节可以参考<https://blogs.technet.microsoft.com/virtualization/2012/08/16/introduction-to-resource-metering/>。对于磁盘和网络的指标很容易理解。而CPU的计算相对复杂，建议可以仔细阅读其中关于CPU资源的详细解释。

|  |  |
| --- | --- |
| 计数器 | 说明 |
| Disk Read Bytes | 上一个采样周期内的磁盘读取数据量 |
| Disk Read Operations/Sec | 虚拟机的各个磁盘上每秒读操作次数的总和 |
| Disk Write Bytes | 上一个采样周期内的磁盘写入数据量 |
| Disk Write Operations/Sec | 虚拟机的各个磁盘上每秒写操作次数的总和 |
| Network In | 虚拟机所有网卡上的进站数据总量 |
| Network Out | 虚拟机所有网卡上的出站数据总量 |
| Percentage CPU | 虚拟机的CPU资源的总体运行繁忙程度 |

除了由平台层面提供的性能指标，虚拟机可以通过内部运行的应用拓展来提供更细节的性能指标。对于Windows和Linux虚拟机的性能指标，在这里列出的是本人对这些指标的理解，在不同的操作系统上可能有细微的差别。

Windows虚拟机：

1. 内存相关

|  |  |
| --- | --- |
| 计数器 | 说明 |
| \Memory\% Committed Bytes in Use | 这两个计数器都是关于Committed Bytes。在Windows系统的内存管理中，内存使用遵循Reserve和Commit的方式。Committed Byes可以认为是系统确认使用的内存。而系统可以使用的内存是有限的，其上限为内存 + Paging File。当% Committed Bytes in Use接近90%，我们可以认为当前虚拟内存的使用已经接近极限，需要特别留意。 |
| \Memory\% Committed Bytes |
| \Memory\Available Bytes | 在系统中现在可以用于直接满足内存申请的内存数量。这个数值包括了内存中的Standby内存页列表，Free内存页列表和全零内存页列表。通常情况，如果此计数器低于内存总数的10%，需要引起注意。但是对于某些特定的生产压力，如SQL，Exchange和IIS等，这些应用会从操作系统尽可能多的申请内存来自主管理。因此，仅仅以这一个指标不足以说明是否存在内存不足的问题。通常需要总和考虑Page/Sec这个计数器 |
| \Memory\Cache Faults/Sec | Cache Faults是Paging Faults其中的一种，通常由于系统尝试访问一个打开文件的某些段数据时，该段数据不在内存中而产生的。注意Cache Faults包含Hard Fault和Soft Fault，只有Hard Fault的类型才会真正出发磁盘文件读写。一般这个计数器被用作内存分析的辅助判断。 |
| \Memory\Page Faults/Sec | 这几个计数器是被用作是否存在内存不足情况的最主要计数器。其中Paging Faults/Sec指的是系统中产生的内存页交换请求。注意这个请求包含Hard Fault和Soft Fault。Soft Fault指的是该请求可以不通过从磁盘上读写文件就可以满足，而hard fault指的是必须经过物理磁盘读写才可以解决。很显然，hard fault更影响系统的性能。因此，我们用Page/Sec来标注所有的Hard Fault。当Hard Fault引起的磁盘IO超过系统IO总量的70%时，并参照可用内存的数量，我们可以判断是否存在内存不足的问题。 |
| \Memory\Page Reads/Sec |
| \Memory\Page/Sec |
| \Memory\Pool Nonpaged Bytes | Nonpaged Pool和Paged Pool是操作系统在系统内核定义的两种内存资源。其中NonPaged Pool是指这块资源必须存储在物理内存中，而Paged Pool可以被写入页面交换文件。这两种资源在操作系统内部是有限的。一旦耗尽会导致系统失去相应。在64位系统中，由于地址空间的扩展和内存增大，资源耗尽的问题相对比较少见。监控这两种资源可以判断是否存在特定的资源泄露问题。 |
| \Memory\Pool Paged Bytes |
| \Process(\_Total)\Working Set | Working Set是Windows平台一个常用术语，指的是某个进程在物理内存中使用的内存总量。单个进程的Working Set包含可共享部分（例如DLL文件的代码段）和私有部分（数据段）。其中可以跟踪私有部分的Working Set数值来判断是否内存使用量过高或是否存在内存泄露的问题 |
| \Process(\_Total)\Working Set Private |

1. 处理器相关

|  |  |
| --- | --- |
| 计数器 | 说明 |
| \Processor Information(\_Total)\Processor Performance | Processor Frequency反映了CPU的运行频率而Processor Performance反映了CPU的运行效能，比如在CPU主频的多大范围内运行。在物理系统上，由于CPU可能存在一些操作系统之外的功能来提高频率，这个数据有可能超过100%。而在虚拟机环境中，正常数据应在100%以下。通常我们使用Processor Performance来判断CPU的负载效能。 |
| \Processor Information(\_Total)\Processor Frequency |
| \Processor Information(\_Total)\Parking Status | Parking一般用于物理系统上有效安排系统的使用的物理内核，这样可以在负载较低时关闭一定的CPU处理能力而节省能源。在虚拟机的运行环境中处理了解CPU的负载状态外，没有特别的意义。 |
| \Processor Information(\_Total)\% Interrupt Time | 系统使用的CPU时间片中，用于中断处理程序（ISR）的CPU时间。一般这个计数器的数值很低，在5%以下。如果数值较高，很有可能是硬件出现问题导致中断异常。 |
| \Processor Information(\_Total)\% Processor Time | Windows操作系统中，由于将代码运行模式划分为内核态（kernel mode）和用户态（User Mode），因此代码的运行时间也就相应被划分为% Privileged Time和% User Time。而两者的总和为% Processor Time。一般来讲，桌面应用程序和系统服务的CPU异常，反映在User Time上，而硬件，驱动程序和内核异常反映在% Privileged Time上 |
| \Processor Information(\_Total)\% Privileged Time |
| \Processor Information(\_Total)\% User Time |

1. 系统资源，进程相关

|  |  |
| --- | --- |
| 计数器 | 说明 |
| \Process(\_Total)\% Processor Time | 操作系统会以每秒100次的频率产生内部中断，中断处理程序会去检查当时CPU上运行的各个线程，从而以次数来推断该线程/进程占用的时间片，继而计算出全部进程的cpu时间占用，即便单个进程的CPU统计可能有些的偏差，总计的数值应该精确的反应了CPU的负载压力。 |
| \Process(\_Total)\Handle Count | 进程的句柄数一半代表了进程访问的系统对象的数目。通过判断句柄数过高，或者有异常增长状况，可以判断是否存在资源使用异常，或是泄露问题。 |
| \Process(\_Total)\Page Faults/sec | 此计数器同Memory/Page Faults/sec意义相同，只是将各个进程引起的Page Faults累加得到 |
| Process(\_Total)\Private Bytes | 所有进程的私有内存空间（可以是在物理内存中，或者是在内存交换文件中的空间）总和。一般使用这个计数器来跟踪私有内存空间的变化趋势，从而判断是否有内存泄露的问题 |
| \Process(\_Total)\Thread Count | 所有进程中的线程数目总和。在Windows系统中，线程是真正执行代码的单元。线程数目可以反应出当前系统中运行的代码单元的多少。线程数目的异常变化，一定程度上反应了系统的负载变化。 |
| \System\Processes | 当前操作系统中运行的进程和线程总数。 |
| \System\Threads |
| \Thread(\_Total)\Context Switches/sec | Context Switch指的是在CPU上运行的线程被其他线程替代。在Windows系统中，Context Switch是一个正常线程处理操作。这个数据的高低并不代表系统是否异常。系统管理员也无法对这个数据进行调整。通常我们可以根据长期观察到的单个系统上的Context Switch数值作为此系统的一个基础数值。只有出现极度异常的量级改变时，才需要引起注意。而这类问题也多发于物理设备异常。 |

1. 磁盘相关

|  |  |
| --- | --- |
| 计数器 | 说明 |
| \PhysicalDisk(\_Total)\Disk Read Bytes/sec | 所有磁盘上的每秒读或写的数据量 |
| \PhysicalDisk(\_Total)\Disk Write Bytes/sec |

1. 网络相关

|  |  |
| --- | --- |
| 计数器 | 说明 |
| \TCPv4\Connection Failure | 连接失败的数量。连接失败指的是连接的状态从SYN-SENT或是SYN-RCVD直接被置为CLOSED，或者是从SYN-RCVD状态置为LISTEN。 |
| \TCPv4\Connection Established | 当前系统中TCP连接的状态是ESTABLISHED 或CLOSE-WAIT的数目。 |
| \TCPv4\Connection Reset | 连接被重置的数量。重置指的是TCP连接的状态从ESTABLISHED或是CLOSE-WAIT的直接被置为CLOSED |
| \TCPv4\Segments Received/sec | 当前建立的连接中，每秒接收的数据段，包括错误的数据段。 |
| \TCPv4\Segments Restransmitted/sec | 每秒中重传的数据段数目。重传的数据段指的是数据段中包括1个以上的字节数是以前传送过的数据。 |
| \TCPv4\Segments Sent/sec | 当前建立的连接中，每秒发送的数据段。但如果一个数据段中只包含之前的重传数据，则不被计入。 |

此外，Azure平台还收集了一些Windows系统中应用相关的计数器，如.Net，由于我们主要讨论的是虚拟机层面的监控，在此就不再具体解释。如果需要，可以才参考相应的技术文档，如<https://docs.microsoft.com/en-us/dotnet/framework/debug-trace-profile/performance-counters>

Linux虚拟机：

同Windows虚拟机不同的是，Windows虚拟机的某些性能指标指定了\_Total，也就是说系统已经统计了同一个指标在多个实例上的数据，比如多CPU系统，多个磁盘系统。在Linux虚拟机中，我们可以指定是针对某一个特定实例还是总计的数值。具体设置请参照<https://docs.microsoft.com/en-us/azure/virtual-machines/linux/diagnostic-extension>

1. 内存相关：

|  |  |
| --- | --- |
| 计数器 | 说明 |
| Mem. Percent available | 当前系统的可用内存比例 |
| Mem. Used by cache | 系统内存中被磁盘缓存使用的数量 |
| Memory available | 当前系统可以使用的内存。可用内存指的是系统在收到内存请求时，可以不需要进行Paging操作，而直接可以使用的内存。他包括Free的内存和一部分可以被重新使用的Cache内存。 |
| Memory percentage | 已用内存的比例 |
| Memory used | 已用内存的数量 |
| Page reads | 每秒中从后端存储中（swap file, program file, mapped file）读取\写入\总计的内存页的个数。这个数据对应于Windows平台的Pages/Sec |
| Page writes |
| Pages |
| Swap available | 这些数据分别对应了swap文件的可用空间大小，可用空间比例，已经使用的比例和已经使用的大小 |
| Swap percent available |
| Swap percent used |
| Swap used |

注：对于可用内存的理解，我参照的文档是根据<https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=34e431b0ae398fc54ea69ff85ec700722c9da773>。

1. 处理器相关

|  |  |
| --- | --- |
| 计数器 | 说明 |
| CPU DPC time | 在Linux中很少提到DPC Time, DPC是在windows平台中常用的一个术语，是中断处理程序将一些可以不在中断处理进程中的任务，以DPC的方式执行。在Linux中，按我的理解，这个数据应该是SoftIRQ的时间。应该是在Interrupt的一部分。 |
| CPU IO wait Time | 这个时间很容易被理解为CPU等待同步IO完成的时间。实际上，IO Wait Time是，CPU Idle Time的一个子集。他指的是当某个CPU处于空闲状态时，至少有一个任务在等待他的磁盘IO完成，进而可以在此CPU上继续运行。因此，较低的IO wait Time并不能代表理想的磁盘性能，我们必须结合CPU的数量，使用率，磁盘的读写性能，读写模式来具体分析。 |
| CPU idle time | CPU 在运行空闲进程的时间。 |
| CPU Interrupt time | CPU运行在Interrupt模式的时间，包括硬件中断和软件中断。 |
| CPU nice time | CPU上运行niced进程所用到的时间。niced进程指的是进程的nice级别（Priority）被改变的进程。 |
| CPU percentage | CPU上运行非Idle线程的时间比例 |
| CPU privileged time | 在非Idle时间内，用于运行Kernel模式进程的时间比例 |
| CPU user time | CPU上运行非niced进程所用到的时间。 |

1. 磁盘相关

|  |  |
| --- | --- |
| 计数器 | 说明 |
| Disk queue length | 磁盘队列长度。对于Azure标准存储账户中的磁盘，我们可以认为是Spindle是1的标准IDE磁盘。由于队列长度会影响磁盘IO完成的时间，如果Disk Queue Length持续高于1，说明总是有磁盘IO在等待处理。需要结合Disk Transfer Time来判断当前的IO压力是否超过磁盘能力。 |
| Disk read guest OS | 每秒磁盘中数据读取的总量 |
| Disk read time | 完成每次读IO操作时需要的平均时间 |
| Disk reads | 每秒磁盘上的进行读操作的次数 |
| Disk total bytes | 每秒磁盘中数据传输的总量，包括读取和写入的数据总量。 |
| Disk transfer time | 完成每次IO操作是需要的平均时间，一般来说，对于生产系统，理想的IO操作需要在10ms以内完成。超过10ms，需要对具体的IO进行分析并引起注意。如果IO完成时间总是在30ms甚至以上，磁盘的效能应该不能满足当前的IO需求。 |
| Disk transfers | 每秒磁盘上的进行读或写操作的次数，也就是经常提到的IOPS |
| Disk write guest OS | 每秒磁盘中数据写入的总量 |
| Disk write time | 完成每次写IO操作时需要的平均时间 |
| Disk writes | 每秒磁盘上的进行写操作的次数 |

1. 网络相关

|  |  |
| --- | --- |
| 计数器 | 说明 |
| Network collisions | 从系统启动开始发生的网络冲突数量。如果持续出现冲突值表示网络基础架构存在性能瓶颈，而并非服务器。在Azure平台上，不应该看到网络冲突的发生 |
| Network in guest OS | 从系统启动开始计算的网络出口流量 |
| Network out guest OS | 从系统启动开始计算的网络入口流量 |
| Network total bytes | 从系统启动开始计算的网络传输数据总量 |
| Packets received | 从系统启动开始计算的接收到的数据包个数 |
| Packets received errors | 从系统启动开始计算的接收端数据包错误的个数 |
| Packets sent | 从系统启动开始计算的发送的数据包个数 |
| Packets sent errors | 从系统启动开始计算的发送端数据包错误的个数 |

以上是我们在Azure的管理门户中常用的性能技术器，在了解了这些计数器内容的基础上，我们可以设定警报，或是自动缩放功能。在下一篇文章中我们会具体了解这些数据是如何被收集并存放的。