文章
Hao Ma · 一月 10, 2021 阅读大约需 7 分钟

完整性检查_加速还是减速

虽然 Caché 和 InterSystems IRIS
数据库的完整性完全不会受到系统故障的影响，但物理存储设备故障确实会损坏其存储的数据。 因此，许多站点选择运行定期数据库完整性检查，尤其要与备份配合，以验证在发生灾难时是否可以依赖给定的备份。
系统管理员在应对涉及存储损坏的灾难时，也可能强烈需要完整性检查。 完整性检查必须读取所检查的 global 的每个块（如果尚未在缓冲区中），并且按照 global 结构指示的顺序读取。
这会花费大量时间，**但完整性检查能够以存储子系统可以承受的最快速度进行读取**。
在某些情况下，需要以这种方式运行以尽快获得结果。
在其他情况下，完整性检查需要更加保守，以避免消耗过多的存储子系统带宽。

行动计划

以下概述适合大多数情况。 本文其余部分中的详细讨论提供了采取其中任一行动或得出其他行动方案所需的信息。

1. 如果使用 Linux 并且完整性检查很慢，请参阅下面有关启用异步 I/O 的信息。
2. 如果完整性检查必须尽快完成，则在隔离的环境中运行；或者如果迫切需要结果，则使用多进程完整性检查来并行检查多个 global 或数据库。 进程数乘以每个进程将执行的并发异步读取数（默认为 8，如果使用 Linux 并且禁用异步 I/O 则为 1）是实时并发读取数的限制。
   假定平均数是限制数量的一半，然后与存储子系统的能力进行比较。 例如，存储由 20 个驱动器条带化，每个进程的默认并发读取数为 8，则可能需要 5 个或更多进程才能利用存储子系统的全部能力 (5*8/2=20)。
3. 在平衡完整性检查速度与对生产的影响时，首先调整多进程完整性检查的进程数，然后如果需要的话，查看可调参数 SetAsyncReadBuffers。 对于长期解决方案（以及为消除误报），请参见下面的隔离完整性检查。
4. 如果已经被限制为一个进程（例如有一个极大的 global 或存在其他外部约束），并且完整性检查的速度需要上下调整，则查看下面的可调参数 SetAsyncReadBuffers。

多进程完整性检查

让完整性检查更快完成（以更高的速度使用系统资源）的一般解决方案是将工作分给多个并行进程。
一些完整性检查用户界面和 API 会这样做，而其他一些则使用单个进程。 对进程的分配按 global 进行，因此对单个 global 的检查始终由一个进程执行（Caché 2018.1 之前的版本按数据库而不是按 global 分配工作）。

多进程完整性检查的主要 API 是 **CheckLIst^Integrity**（有关详细信息，请参阅文档）。 它将结果收集在一个临时的 global 中，通过 Display^Integrity 来显示。 以下是使用 5 个进程检查 3 个数据库的示例。
这里如省略数据库列表参数，将检查所有数据库。

```
set dblist=$listbuild("/data/db1/","/data/db2/","/data/db3/")
set sc=$$CheckList^Integrity(,dblist,,,5)
do Display^Integrity()
kill ^IRIS.TempIntegrityOutput(+$job)

/* Note: evaluating 'sc' above isn't needed just to display the results, but...
   $system.Status.IsOK(sc) - ran successfully and found no errors
   $system.Status.GetErrorCodes(sc)=$$$ERRORCODE($$$IntegrityCheckErrors) // 267
                           - ran successfully, but found errors.
   Else - a problem may have prevented some portion from running, 'sc' may have
```

```
        multiple error codes, one of which may be $$$IntegrityCheckErrors. */
```

像这样使用 CheckLIst^Integrity 是实现我们感兴趣的控制水平的最直接方法。
管理门户接口和完整性检查任务（内置但未安排）使用多个进程，但可能无法为我们的用途提供足够的控制。*

其他完整性检查接口，尤其是终端用户接口 ^INTEGRIT 或 ^Integrity 以及
Silent^Integrity，在单个进程中执行完整性检查。
因此，这些接口不能以最快的速度完成检查，并且它们使用的资源也较少。
但一个优点是，它们的结果是可见的，可以记录到文件或输出到终端，因为每个 global
都会被检查，而且顺序明确。

异步 I/O

完整性检查进程会排查 global 的每个指针块，一次检查一个，根据它指向的数据块的内容来进行验证。
数据块以异步 I/O
的方式读取，以确保每时每刻都有一定数量的读取请求供存储子系统处理，并且每次读取完成后都进行验证。

在 Linux 上，异步 I/O 只有与直接 I/O 结合时才有效，而 InterSystems IRIS 2020.3 之前的版本默认不启用直接 I/O。 这解释了大量 Linux 上完整性检查时间过长的情况。 幸运的是，可以在 Cache 2018.1、IRIS 2019.1 及以后的版本上启用直接 I/O，方法是在 .cpf 文件的 [config] 部分中设置 **wduseasyncio=1**，然后重新启动。
通常建议设置此参数，以在繁忙系统上实现 I/O 可伸缩性，并且自 Caché 2015.2 起，在非 Linux
平台上默认设置此参数。 在启用之前，确保已经为数据库缓存（global 缓冲区）配置了足够的内存，因为启用直接 I/O 后，数据库将不再被 Linux（冗余）缓存。 未启用时，完整性检查执行的读取会同步完成，不能有效利用存储。

在所有平台上，完整性检查进程一次执行的读取数默认设置为 8。
如果必须更改单个完整性检查进程从磁盘读取的速率，可以调整此参数 –
向上调会使单个进程更快完成，向下调则使用更少的存储带宽。 请记住：

- 此参数应用于每个完整性检查进程。 当使用多个进程时，进程数会使实时读取数增加。更改并行完整性检查进程数会产生较大影响，因此这通常是先做的事情。
  每个进程还受到计算时间的限制（除其他限制外），因此增加此参数的值所获得的收益也有限。
- 这只在存储子系统处理并发读取的能力范围内有效。 如果数据库存储在单个本地驱动器上，再高的数值也没有用处，而在几十个驱动器上条带化的存储阵列可以并发处理几十个读取。

要从 %SYS 命名空间调整此参数，则 **do SetAsyncReadBuffers^Integrity(**value**)**。 要查看当前值，则 **write $$GetAsyncReadBuffers^Integrity()**。 更改在检查下一个 global 时生效。
目前，该设置在系统重启后不会保持，虽然可以将其添加到 SYSTEM^%ZSTART 中。

有一个相似的参数用于在磁盘上的块是连续（或接近连续）分布时控制每次读取的最大大小。
此参数很少需要用到，尽管具有高存储延迟的系统或具有较大块大小的数据库可能会从微调中受益。 该值的单位为 64KB，因此值 1 表示 64KB，4 表示 256KB 等等。0（默认值）表示让系统选择，当前选择 1 (64KB)。 此参数的 ^Integrity 函数（类似于上面提及的函数）为 **SetAsyncReadBufferSize** 和 **GetAsyncReadBufferSize**。

隔离完整性检查

许多站点直接在生产系统上运行定期完整性检查。 这当然是最简单的配置，但并不理想。 除了完整性检查对存储带宽的影响，并发数据库更新活动有时还可能导致误报错误（尽管检查算法内置了缓解措施）。
因此，在生产系统上运行的完整性检查所报告的错误，需要由管理员进行评估和/或重新检查。

很多时候，存在更好的选择。 可以将存储快照或备份映像挂载到另一台主机上，在那里由隔离的 Caché 或 IRIS 实例运行完整性检查。
这样不仅可以防止任何误报，而且如果存储也与生产隔离，运行完整性检查可以充分利用存储带宽并更快完成。
这种方法非常适合使用完整性检查来验证备份的模型；经过验证的备份可以有效验证截至生成备份前的生产情况。
还可以通过云和虚拟化平台更容易地从快照建立可用的隔离环境。

* 管理门户接口、*完整性检查*任务和 SYS.Database 的 IntegrityCheck 方法会选择相当多的进程（等于 CPU 内核数），在很多情况下缺少所需的控制。
管理门户和任务还会对任何报告错误的 global 执行完整的重新检查，以识别可能因并发更新而出现的误报。
除了完整性检查算法内置的误报缓解措施，也可能进行这种重新检查；在某些情况下，由于会花费额外的时间（重新检查在单个进程中运行，并检查整个 global），可能并不需要重新检查。 此行为将来可能会更改。

#性能 #系统管理 #Caché #InterSystems IRIS

源 URL: https://cn.community.intersystems.com/post/%E5%AE%8C%E6%95%B4%E6%80%A7%E6%A3%80%E6%9F%A5-%E5%8A%A0%E9%80%9F%E8%BF%98%E6%98%AF%E5%87%8F%E9%80%9F