数据库智能管家 DBbrain

最佳实践

产品文档

腾讯云

文档目录

最佳实践
如何解决 MySQL 实例 CPU 使用率高问题
如何解决 MySQL 实例锁冲突问题
如何解决 MongoDB 实例 CPU 使用率高问题
如何解决 MongoDB 节点 Oplog 保存时间太短

最佳实践
如何解决 MySQL 实例 CPU 使用率高问题

最近更新时间：2022-07-31 17:12:52

问题描述

MySQL 实例 CPU 利用率过高通常容易导致系统异常，例如响应变慢、无法获取连接、超时等，大量的超时重试往往是性能“雪崩”的罪魁祸首。而 CPU 利用率过高场景，很多时候均是由异常 SQL 所导致，大量锁冲突、锁等待或事务未提交也有可能导致 MySQL 实例 CPU 利用率高。
当数据库执行业务查询、修改语句时，CPU 会先从内存中请求数据块（默认是8kb）：
如果内存中存在对应的数据，CPU 执行计算任务后会将结果返回给用户，可能涉及到排序类高消耗 CPU 的动作。
如果内存中不存在对应的数据，数据库会触发从磁盘获取数据的动作。
这两个数据获取过程分别称为逻辑读和物理读。因此，性能较低的 SQL，在执行时容易让数据库产生大量的逻辑读，从而导致 CPU 利用率过高，也可能让数据库产生大量的物理读，从而导致 IOPS 和 I/O 时延过高。

解决方案

DBbrain 为用户提供三大功能来排查和优化导致 CPU 利用率过高的异常 SQL 语句：
异常诊断：7 * 24小时异常发现诊断，提供实时优化建议。
慢 SQL 分析：针对当前实例出现的慢 SQL 进行分析，并给出慢 SQL 的优化建议。
审计日志分析：利用云数据库审计数据（全量 SQL），多维度深入分析 SQL 语句并给出优化建议。

方式一：使用“异常诊断”功能排查数据库异常情况（推荐）

异常诊断功能提供故障主动定位和优化，不需要任何数据库运维经验，不仅包括 CPU 利用率过高的异常，还几乎涵盖所有 MySQL 实例高频的异常和故障。
操作步骤及示例如下：
1. 登录 DBbrain 控制台，在左侧导航选择**诊断优化**，在上方选择**异常诊断**页。
2. 在左上角选择实例 ID（可输入和搜索），切换至目标实例。
3. 在页面中选择“实时”或“历史”要查询的时间，若该时间段内存在故障，可在右侧的“诊断提示”中查看到概要信息。
4. 单击“实时/历史诊断”栏的**查看详情**或“诊断提示”栏的诊断项可进入诊断详情页。
事件概要：包括诊断项、起止时间、风险等级、持续时长、概要等信息。
现象描述：异常事件（或健康巡检事件）的外在表现现象的快照和性能趋势。
智能分析：分析导致性能异常的根本原因，定位具体操作。
专家建议：提供优化指导建议，包括但不限于 SQL 优化（索引建议、重写建议）、资源配置优化和参数调优。

5. 选择**专家建议**页，即可查看 DBbrain 针对该故障给出的优化建议，本例中是 SQL 语句的优化建议。本例中的 SQL 语句执行时缺少对应的索引，导致该语句执行时要进行全表扫描，单次执行成本高，所以大量并发场景下就很容易导致 CPU 利用率过高，可能会达到100%的状况。

方式二：使用“慢 SQL 分析”功能排查导致 CPU 利用率过高的 SQL

1. 登录 DBbrain 控制台，在左侧导航选择**诊断优化**，在上方选择**慢 SQL 分析**页。
2. 在左上角选择实例 ID（可输入和搜索），切换至目标实例。
3. 在页面中选择要查询的时间，若此实例在该时间段中有慢 SQL，SQL 统计会以柱形图的方式展示慢 SQL 产生的时间点和个数。

单击柱形图，下方的列表会显示对应的所有慢 SQL 信息（模板聚合之后的 SQL），右方会显示该时间段内 SQL 的耗时分布。

4. 针对 SQL 列表中 SQL 执行的数据进行判断和筛选，下面简单介绍一种判断方式：

1. 先按照平均耗时（或者最大耗时）降序，重点关注耗时处在 top 的 SQL，不推荐使用总耗时，容易受到执行次数多而累加的干扰。
2. 然后关注返回行数和扫描行数的值。

若发现“返回行数”与“扫描行数”值相等的 SQL，大概率是全表查找并返回了。

若发现几行 SQL 都有很多扫描行数，但返回行数都为0或特别小，说明系统产生了大量的逻辑读和物理读。当查找的数据量过大且内存不足时，该请求必然会产生大量的物理 I/O 请求，导致 I/O 资源大量消耗；大量的逻辑读便会占用大量的 CPU 资源，导致 CPU 利用率过高。

3. 单击 SQL 语句，可查看该 SQL 语句的详情、资源消耗以及优化建议。

分析页：可查看完整的 SQL 模板、SQL 样例以及优化建议和说明，您可根据 DBbrain 给出的专家建议优化 SQL，提升 SQL 性能，降低 SQL 执行的耗时。

统计页：可根据统计报表的总锁等待时间占比、总扫描行数占比、总返回行数占比，横向分析该条慢 SQL 产生的具体原因，以及进行对应优化。

耗时分布页：可查看该类型的 SQL（经过聚合后汇总的）运行的时间分布区间，以及来源 IP 的访问占比。

方式三：使用“审计日志分析”功能排查导致 CPU 利用率过高的 SQL

前提条件：实例需要开通数据库审计功能。

1. 登录 DBbrain 控制台，在左侧导航选择**诊断优化**，在上方选择**审计日志分析**页。
2. 在左上角选择实例 ID（可输入和搜索），切换至目标实例。
3. SQL 透视图可选择 QPS 或慢查询次数维度。单击视图右上角的**创建审计任务**，选择任务开始时间和时间间隔，单击**确定**。

任务创建后，任务列表中会显示任务生成，任务完成后，找到目标记录并单击**查看SQL分析**，进入 SQL 分析详情页。

4. 在 SQL 分析页，可选择 SQL Type、Host、User 或 SQL Code 维度的视图，并可选择时间段拉伸视图来查看具体时间点的数据。

下面表格中会展示该时间段内 SQL 的聚合详情以及执行信息。若对图中时间进行部分拉伸选中，表格中的 SQL 数据会随之变化，仅展示图中时间范围内的 SQL 分析结果。

5. SQL 分析详情页下方表格中展示了聚合后的 SQL 执行的信息，包括执行次数、总延迟、最大延迟、最小延迟、总影响行数、最大影响行数、最小影响行数等。可根据各项信息的组合排序，识别出待优化的SQL。
示例：
根据执行次数降序排列，以定位执行次数较大或者执行次数有异常变化的语句，然后分析 SQL 执行次数的合理性并根据 DBbrain 给出的建议优化 SQL 语句。
通过查看执行次数、总延迟、最大延迟，可以判断出执行次数最多的前两个 SQL 语句的平均执行延迟很短，说明这两个 SQL 语句性能未出现异常。
但观察第三条语句就会明显发现其单条执行时间在100s左右，因为执行次数较少，故总延迟未在 top 前2，但这类 SQL 是需要进行优化的，可单击 SQL 查看 DBbrain 给出的专家建议、资源消耗分析曲线以及来源 IP 分析等。
还有一类 SQL 也需要引起重视，可以看到第四条语句的最大延迟和最小延迟相差很大，达到了200s以上，说明整个系统存在波动，需要进一步分析波动是因为网络问题还是数据量变化导致了执行计划改变。
如果 SQL 语句的执行次数和平均耗时相对比较合理，而且执行次数大的 SQL 也是最优的，如果性能达到瓶颈的话，建议 升级实例规格配置 或者使用 读写分离功能 来打散执行次数较大的 SQL 语句，或者采用前置缓存数据库进行优化。

如何解决 MySQL 实例锁冲突问题

最近更新时间：2022-07-31 17:10:56

问题描述

锁冲突问题一直是 MySQL 数据库业务经常遇到的问题，锁冲突问题在不同场景下表现也大不相同，某些场景下可能直接导致业务瘫痪，而某些场景下业务侧可能很难感知，出现数据错乱等情况。
锁冲突场景的多样性、复杂性以及隐秘性，决定了要解决这一类问题对数据库管理员的技术和能力要求极高，同时处理锁冲突故障的时效性也会大打折扣。
数据库智能管家 DBbrain 提供的异常诊断功能，包含了数十个锁相关诊断项，涵盖了死锁、等待行锁、出现表锁、只读锁、DDL/select/DML等待锁、SQL 等待 MDL 锁等多个锁冲突场景，能够高效、便捷、专业化地帮助用户解决锁冲突的问题。
本文以常见的“等待行锁”和“死锁”为案例，介绍如何使用 DBbrain 解决锁冲突问题。

解决方案

场景一：“等待行锁”场景

步骤一：查看“等待行锁”事件

方式1：

1. 登录 DBbrain 控制台，在左侧导航选择**监控告警>异常告警**页。
2. 选择需要查看的时间范围，在“诊断项”列，选择“等待行锁”进行过滤。
3. 列表会展示实例出现“等待行锁”的事件列表，单击“操作”列的**详情**可进入事件详情页。

方式2：

1. 登录 DBbrain 控制台，在左侧导航选择诊断优化，然后选择异常诊断页。
2. 在上方选择对应实例，然后选择实时（默认动态更新）或历史（可自定义）时间段。
3. 查看“诊断提示”栏中是否存在“等待行锁”的事件，单击事件可进入事件详情页。

步骤二：解决“等待行锁”事件

1. 进入事件详情页后，在“现象描述”页，了解出现“等待行锁”事件时，数据库中语句的运行情况。
2. 切换至“智能分析”页，查看出现“等待行锁”事件的原因。
等待行锁事务中，清晰显示被阻塞的事务语句情况，例如下图中的 delete 语句，是处于 LOCK WAIT 状态。
根据持有锁事务结果，快速定位目前持有行锁的事务当前状态以及 ID。
性能监控曲线中，直观展示出等待行锁次数的趋势。
3. 切换至“专家建议”页，根据提示解决锁冲突问题。例如，通过使用 kill 命令，中止会话3965158来释放锁，解决本案例中的锁冲突问题。

场景二：“死锁”场景

注意：

由于 MySQL 具有“死锁”监测和自动事务回滚的能力，故大多数“死锁”场景可自愈，业务无感知，但由于业务逻辑不够健壮或极端情况下也会导致系统雪崩，以及数据不一致等后果，故对于出现死锁的情况也需足够重视。

步骤一：查看“死锁”事件

方式1：

1. 登录 DBbrain 控制台，在左侧导航选择**监控告警>异常告警**页。
2. 选择需要查看的时间范围，在“诊断项”列，选择“死锁”进行过滤。
3. 列表会展示实例出现“死锁”的事件列表，单击“操作”列的**详情**可进入事件详情页。

方式2：

1. 登录 DBbrain 控制台，在左侧导航选择诊断优化，然后选择异常诊断页。
2. 在上方选择对应实例，然后选择实时（默认动态更新）或历史（可自定义）时间段。
3. 查看“诊断提示”栏中是否存在“死锁”的事件，单击事件可进入事件详情页。

步骤二：解决“死锁”事件

进入事件详情页后，在“现场描述”页，通过分析如下信息优化对应语句，来解决“死锁”事件。

发生时间以及来源 IP，便于溯源。

发生死锁的两条 SQL 语句，例如下图的两条 INSERT INTO 语句。

根据 Status 字段状态，判断哪条语句回滚（Rollback），哪条语句正常执行（Normal）。

（可选）根据 LockRequest、LockHold 信息分析具体锁的持有和类型。

如何解决 MongoDB 实例 CPU 使用率高问题

最近更新时间：2022-08-13 20:18:46

问题描述

在日常运维中，MongoDB 数据库，如果 CPU 利用率过高，通常容易导致系统异常。例如读写速率变慢、链接耗光、超时增加等，大量的访问超时还会触发客户端反复重连认证，最终可能会引起数据库“雪崩”。
生产中的 MongoDB 数据库，出现 CPU 利用率过高场景是很常见的，这种问题一般由 SQL 异常、流量过高、产生内存排序、语句无索引、或索引使用不当等情况导致。
当数据库执行查询、修改等操作时，CPU 会先从存储引擎 cache 中请求数据：
如果引擎 cache 中存在对应的数据，CPU 执行计算任务后会将结果返回给用户，与此同时，还可能涉及到排序类高消耗 CPU 的动作。
如果内存中不存在对应的数据，还会触发从磁盘获取数据的动作。
这两个数据获取过程分别称为逻辑读和物理读。因此，性能较低的 SQL，在执行时容易让数据库产生大量的逻辑读，从而导致 CPU 利用率过高，也可能让数据库产生大量的物理读，从而导致 IOPS 和 I/O 时延过高。

解决方案

使用数据库智能管家 DBbrain，从异常诊断，可定位 CPU 过高的异常问题，可确定问题发生时间，确定造成问题的具体 SQL，还能给您对应的处理措施；在解决问题的同事，结合慢 SQL 优化功能 ，通过语句优化分析建议，可帮您精准分析语句情况，避免类似问题再次产生。
异常诊断：7 * 24小时异常发现诊断，提供实时优化建议。
慢 SQL 分析：针对当前实例出现的慢 SQL 进行分析，并给出慢 SQL 的优化建议。
实时会话：查看当前生产库中正在执行中的操作，可针对异常操作做处理。

方式一：使用“异常诊断”功能排查数据库异常情况（推荐）

异常诊断功能提供故障主动定位和优化，不需要任何数据库运维经验，不仅能发现 CPU 利用率过高的异常情况，经过多年 MongoDB 运维专家经验沉淀，结合机器学习、大数据与智能分析算法的运用，快速复制资深数据库专家经验，赋能您的数据库，智能运维 MongoDB 数据库，几乎可以实时发现 MongoDB 生产数据库中所有的异常与故障问题。
操作步骤及示例如下：
1. 登录 DBbrain 控制台，在左侧导航选择**诊断优化**，在上方选择**异常诊断**页。
2. 在左上角选择实例 ID（可输入和搜索），切换至目标实例。
3. 在页面中选择**实时**或**历史**要查询的时间，若该时间段内存在故障，可在右侧的**诊断提示**中查看到概要信息。
4. 单击**实时/历史诊断**栏的**查看详情**或**诊断提示**栏的诊断项可进入诊断详情页。

事件概要：包括诊断项、起止时间、风险等级、持续时长、概要等信息。
现象描述：异常事件（或健康巡检事件）的外在表现现象的快照和性能趋势。
智能分析：分析导致性能异常的根本原因，定位具体操作。
优化建议：提供优化指导建议。
5. 选择**优化建议**页，即可查看 DBbrain 针对该故障给出的优化建议。

方式二：使用“慢 SQL 分析”功能排查导致 CPU 利用率过高的库表信息

1. 登录 DBbrain 控制台，在左侧导航选择**诊断优化**，在上方选择**慢 SQL 分析**页。
2. 在左上角选择实例 ID（可输入和搜索），切换至目标实例。
3. 在页面中选择要查询的时间，若此实例在该时间段中有慢 SQL，SQL 统计会以柱形图的方式展示慢 SQL 产生的时间点和个数。
单击柱形图，下方的列表会显示对应的所有慢 SQL 信息（模板聚合之后的 SQL），右方会显示该时间段内 SQL 的耗时分布。
4. 针对 SQL 列表中 SQL 执行的数据进行判断和筛选，下面简单介绍一种判断方式：
1. 先按照平均耗时（或者最大耗时）降序，重点关注耗时较大的 SQL，不推荐使用总耗时，容易受到执行次数多而累加的干扰。
2. 然后关注返回行数和扫描行数的值。
若发现**返回行数**与**扫描行数**值相等的 SQL，大概率是全表查找并返回了。
若发现几行 SQL 都有很多扫描行数，但返回行数都为0或特别小，说明系统产生了大量的逻辑读和物理读。当查找的数据量过大且内存不足时，该请求必然会产生大量的物理 I/O 请求，导致 I/O 资源大量消耗；大量的逻辑读便会占用大量的 CPU 资源，导致 CPU 利用率过高。

方式三：使用“实时会话”功能 Kill 扫表慢查

MongoDB 内核会记录当前正在执行的 currentOp 信息，DBbrain 的实时会话功能可以看到数据库正在执行中的所有操作，并且支持 Kill 指定会话，可以把耗时 SQL 直接 Kill 掉来释放 CPU、磁盘 IO 等资源。另外还提供持续 Kill 功能，可以根据条件或信息持续 Kill 会话，在数据库产生堵塞的异常情况下，您可以使用持续 Kill 功能，紧急做异常处理。
获取会话信息，Kill 会话。
根据**诊断优化** > **实时会话** > **活跃会话**，选择需要 Kill 的会话信息，然后执行 Kill 操作。
持续 Kill 会话。
持续 Kill 会话功能可以针对 DB、HOST、Type、TIME 等维度进行配置，两个触发机制“超时自动退出”和“手动关闭”：
如需停止持续 Kill 会话，则单击**停止**，即可提前关闭未到超时时间的定时任务，或终止手动触发的任务。

如何解决 MongoDB 节点 Oplog 保存时间太短

最近更新时间：2022-08-13 20:18:46

问题描述

MongoDB 副本集结构中，从节点会保存从主节点同步过来的数据日志，这种方式和 MySQL 数据库的 bin-log 主从复制的方式类似，MongoDB 采用的是根据 oplog 来从主节点同步数据到从节点。但是 oplog 不是无限大，超过其配置大小，oplog 会被覆盖。
在日常运维场景中，当从库发生故障时。主节点仍正常运行，还不会影响整个副本集运行状态，但如果重新启动从节点，那么从节点需要从主节点上恢复数据，这时，如果 oplog 被覆盖，从节点需要恢复的数据就会有遗漏，中间被覆盖的数据就会丢失，这种情况下，从节点无法恢复正常。另外一些情况下，数据库重启、主从延迟高都可能会出现 oplog 不够用的情况。

解决方案

使用数据库智能管家 DBbrain，可帮您7 * 24小时实时观察生产所有节点 oplog 保存问题相关的风险项，从而避免此类故障产生。

使用“异常诊断”功能排查数据库异常情况（推荐）

异常诊断功能提供故障主动定位和优化，不需要任何数据库运维经验，不仅能发现 CPU 利用率过高的异常情况，经过多年 MongoDB 运维专家经验沉淀，结合机器学习、大数据与智能分析算法的运用，快速复制资深数据库专家经验，赋能您的数据库，智能运维 MongoDB 数据库，几乎可以实时发现 MongoDB 生产数据库中所有的异常与故障问题。
操作步骤及示例如下：
1. 登录 DBbrain 控制台，在左侧导航选择**诊断优化**，在上方选择**异常诊断**页。
2. 概览卡片中，呈现黄色标记，告知此时间点发现异常风险项。
3. 右侧的**诊断详情**列表中，同时出现 oplog 保存时间太短的风险项。
4. 单击该风险项查看，可进一步获得更详细的异常信息。优化建议会根据当前情况给出合理的处理措施。