本文档截取自电子工业出版社出版的图书《千金良方：MySQL 性能优化金字塔法则》的附录部分，该图书由来自沃趣科技的李春、罗小波、董红禹三位作者共同撰写。更多精彩内容可关注下方的微信公众号 "**沃趣技术**"与"**知数堂**"


附录 A 线程状态信息详解

- 线程的状态可以通过 SHOW PROCESSLIST 语句进行查看，输出结果中显示了哪些线程正在运行（您还可以通过查询 INFORMATION_SCHEMA PROCESSLIST 表或 mysqladmin processlist 命令获取此信息）
    - 如果用户具有 PROCESS 权限，则可以查看所有用户的线程。否则，您只能看到当前用户自己的线程（即与您使用的 MySQL 帐户相关联的线程）。
    - 如果不使用 FULL 关键字，每个语句只显示前 100 个字符在 info 字段中。
- 也可以从 performance_schema.threads 表中获取进程信息（前提是参数 performance_schema 设置为 ON，表示开启 performance_schema 功能）。对于此表的访问，不需要互斥量，且对服务器性能的影响最小。 （INFORMATION_SCHEMA.PROCESSLIST 和 SHOW PROCESSLIST 语句会有一定的性能影响，因为它们需要互斥量。），另外，此表除了查看 show processlist 中的线程信息外，还可以查看后台线程的信息（INFORMATION_SCHEMA.PROCESSLIST 和 SHOW PROCESSLIST 不能查看后台线程信息）。这意味着此表可以用于监视更多的一些后台线程活动信息。

- SHOW PROCESSLIST 语句是非常有用的，如果你遇到“too many connections”错误信息，并想知道发生了什么。 MySQL 保留一个额外的连接以供具有 SUPER 权限的帐户使用，以确保管理员始终能够连接和检查系统（前提是您没有向所有用户授予此权限）
- show processlist 查看到的线程可以用 KILL 语句杀死。
- 下面是 SHOW PROCESSLIST 输出的示例：

```
root@localhost Tue Mar 14 00:14:06 2017 00:14:06 [(none)]>show full processlist\G;
*************************** 1. row ***************************
     Id: 4
   User: root
   Host: localhost
     db: NULL
Command: Query
   Time: 0
  State: starting
   Info: show full processlist
1 row in set (0.00 sec)
```

- show processlist 输出字段含义详解
  - Id：连接进程标识符。这与在 INFORMATION_SCHEMA.PROCESSLIST 表的 ID 列，performance_schema.threads 表的 PROCESSLIST_ID 列中显示的值是相同的值，都是由 CONNECTION_ID() 函数返回的值
  - User：执行语句的 MySQL 用户名称。如果显示的是“system user”，它指的是由 MySQL Server 生成的非客户端线程正在执行内部任务。例如主备复制中从库上使用的 I/O 或 SQL 线程或延迟行处理程序的线程。“unauthenticated user”指的是客户端已经和服务端建立了 TCP/IP 连接但是还没有对客户端的用户进行用户密码认证的线程。“event_scheduler”是指监视计划任务调度事件的线程。对于“system user”，在 Host 列中显示为 Null 值

- Host：执行语句的客户端的主机名（除了没有主机信息的“system user”之外）。 SHOW PROCESSLIST 的 Host 列以 host_name：client_port 格式显示 TCP/IP 连接的主机名(如果启用了 skip_name_resolve 参数，则显示为 ip:client_port 格式)，以便更容易确定哪个客户端正在做什么事情
- Db：客户端连接的默认数据库（如果连接时指定了库名），否则显示为 NULL 值。注意：如果客户端线程后续执行了 use db_name 语句指定了默认数据库，则再次执行 show processlist 语句的时候所看到的 Db 列值会发生改变。
- Command：线程正在执行的命令的类型。此列的值对应于 C/S 协议和 Com_xxx 状态变量的 COM_xxx 命令。
- Time：线程处于当前状态的时间数（以秒为单位）。对于从库 SQL 线程，该值是最后复制事件的时间和从库的实际时间之间的秒数（也可以理解为事件等待的时间）
- State：提示线程正在做什么样的操作，事件或状态。大多数状态所对应于的操作都执行的非常快。如果线程停留在某个状态很长时间，则表名该线程可能执行过程中碰到了某个问题，需要进行排查。对于 SHOW PROCESSLIST 语句，State 列的值始终为 NULL
- Info：线程正在执行的语句，如果没有执行任何语句，则显示为 NULL。语句可以是发送到服务器的语句，或者如果语句内部调用执行其他语句，即指的最内层调用的语句。例如，如果 CALL 语句调用存储过程，而存储过程中执行 SELECT 语句，则 Info 值将显示存储过程中的 SELECT 语句

1、线程信息

- 本节针对 show processlist 输出结果中的 State 和 Command 字段进行全面解读，详细介绍了所有 show processlist 能够查看到的线程的所有状态信息和 Command 信息及其代表的含义
- 线程信息可以通过如下几种方法进行查询
  - SHOW [FULL] PROCESSLIST 语句

- SHOW PROFILE 语句（该语句从 MySQL 5.6.7 版本开始被弃用，使用 performance_schema 代替）
- INFORMATION_SCHEMA PROCESSLIST 表
- mysqladmin processlist 命令
- performance_schema.threads 表

1.1. 线程的 Command 值有如下值

- Binlog Dump：这是主库上的一个线程，用于将二进制日志内容发送到从库
- Change user：线程正在执行更改用户操作
- Close stmt：线程正在关闭一个预编译好的语句
- Connect：从库线程已经连接到主库
- Connect Out：从库正在连接到主库
- Create DB：线程正在执行一个建库操作
- Daemon：这个是 server 内部线程，不是客户端连接的线程
- Debug：线程正在生成调试信息
- Delayed insert：是一个延迟插入处理程序的线程
- Drop DB：线程正在执行 drop database 操作
- Error
- Execute：线程正在执行一个预编译好的语句
- Fetch：线程正在执行语句并从中获取结果集
- Field List：线程正在检索表列的信息
- Init DB：线程正在选择默认数据库
- Kill：线程正在杀死其他线程
- Long Data：线程在执行语句并从中检索并返回长字段（大字段）类型的数据结果集
- Ping：线程正在处理服务器 ping 请求
- Prepare：线程正在执行预编译一个语句
- Processlist：线程正在生成有关 server 线程的信息
- Query：线程正在执行查询语句
- Quit：线程正在终止

- Refresh：线程正在刷新表，日志或高速缓存，或重置状态变量或复制 server 信息
- Register Slave：线程正在主库上注册从库
- Reset stmt：线程正在重置预编译语句
- Set option：线程正在设置或重置客户端语句执行选项
- Shutdown：线程正在执行关闭 server
- Sleep：线程正在等待客户端向其发送一个新语句请求
- Statistics：线程正在生成 server 状态信息
- Table Dump：线程正在将表内容发送到从库
- Time：当前未使用

1.2. 普通线程状态

- 这里所说的普通线程指的非后台线程，非 MySQL 内部机制创建的线程，而是指的用户客户端正常访问时创建的线程。这些线程状态仅用于在 server 中排查一些普通线程的问题，下面列出了所有普通线程的状态信息及其代表的含义
    - After create：当线程创建一个表完成时（包括内部临时表），会出现这种状态。即使由于某些错误而导致创建表最终出错，也会出现此状态
    - Analyzing：线程正在计算 MyISAM 表的 key 分布（例如，对于 ANALYZE TABLE）
    - checking permissions：正在 server 中检查线程是否具有执行语句所需的权限
    - Checking table：线程正在执行表检查操作
    - cleaning up：线程已经执行完成了一个命令，并准备释放所占用的内存和重置某些状态变量
    - closing tables：线程正在将表发生更改的数据刷新到磁盘并关闭表。通常情况下这个操作很快。否则，请检查磁盘空间是否满了，或者磁盘的负载是否比较高
    - converting HEAP to MyISAM：线程正在将内部临时表从 MEMORY 引擎表转换为磁盘 MyISAM 引擎的临时表

- copy to tmp table：线程正在执行 ALTER TABLE 语句。此状态发生在新结构的表已经创建好之后，执行 copy 旧表数据到新表中之前出现，这里对于线程的具体执行详情，可以使用 performance_schema 库来获取有关 copy 操作的进度信息
- Copying to group table：如果语句使用了不同的 ORDER BY 和 GROUP BY 条件列，则按照 group by 对这些行数据进行排序，并将排序结果复制到临时表
- Copying to tmp table：server 正在复制数据到内存临时表
- altering table：server 正在执行 in-place 算法的 ALTER TABLE 的过程
- Copying to tmp table on disk：server 正在复制数据到磁盘临时表。因为临时结果集太大，所以，线程正在将内存临时表转换为基于磁盘的临时表，以节省内存
- Creating index：线程正在执行一个 MyISAM 表的 ALTER TABLE ... ENABLE KEYS 语句
- Creating sort index：线程正在执行 SELECT 且使用到了内部临时表
- creating table：线程正在创建表。包括创建临时表时也会使用此状态
- Creating tmp table：线程正在内存或磁盘上创建一个临时表。如果表在内存中创建，但后来被转换为磁盘表，则该操作期间的状态将为“Copying to tmp table on disk”
- committing alter table to storage engine：server 已执行完成 in-place 算法的 ALTER TABLE 语句，正在提交
- deleting from main table：server 正在执行多表删除语句中的第一部分。看到这个状态表示正在从第一个表中删除，并保存后续用于删除其他表的列数据和偏移量
- deleting from reference tables：server 正在执行多表删除语句的第二部分，从其他表（非第一个表）中删除匹配的行
- discard_or_import_tablespace：线程正在执行 ALTER TABLE ... DISCARD TABLESPACE 或 ALTER TABLE ... IMPORT TABLESPACE 语句

- end：这发生在语句执行结束时，但在清除 ALTER TABLE，CREATE VIEW，DELETE，INSERT，SELECT 或 UPDATE 语句之前出现该状态
- executing：线程正在执行语句中
- Execution of init_command：线程正在执行一个初始化系统变量的语句
- freeing items：线程已经执行完成了一个命令。释放一些涉及到 query cache 状态的 items。这种状态后通常紧随 cleaning up 状态之后
- FULLTEXT initialization：server 正在准备执行自然语言全文搜索
- init：这在 ALTER TABLE，DELETE，INSERT，SELECT 或 UPDATE 语句初始化之前发生的状态。server 在此状态下执行的操作包括刷新二进制日志，InnoDB 日志和一些查询缓存清理操作。对于这个状态结束时，可能会有如下一些操作：
  * 当表中的数据更改后删除查询缓存条目
  * 将事件写入二进制日志
  * 释放内存缓冲区，包括 blob
- Killed：向线程发起一个 kill 操作，线程应该执行终止操作。在 MySQL 的每个主循环中检查线程的 kill 标志，但在某些情况下，杀死线程可能只需要很短的时间。但如果被 kill 的线程被其他线程锁定，则需要等待其他线程释放锁之后，kill 命令才会生效并执行。
- logging slow query：线程正在向慢查询日志写一条语句
- login：连接线程的初始状态，直到客户端成功通过身份验证
- manage keys：server 正在启用或禁用表索引
- NULL：此状态用于 SHOW PROCESSLIST 语句
- Opening tables：线程正尝试打开一个表。打开表操作应该非常快，除非打开操作被阻止。例如，ALTER TABLE 或 LOCK TABLE 语句可以防止打开表，直到该语句完成。另外也可能是 table_open_cache 不够大导致不能打开表。
- optimizing：server 正在对查询执行初始优化
- preparing：此状态发生在查询优化期间

- Purging old relay logs：线程正在删除不需要的中继日志文件
- query end：此状态出现在执行查询语句之后但在释放该查询语句相关状态 items 之前
- Reading from net：server 正在从网络读取数据包。在 MySQL 5.7.8 之后该状态叫做“Receiving from client ”
- Receiving from client：server 正在从客户端读取数据包。在 MySQL 5.7.8 叫做“Reading from net”
- Removing duplicates：查询使用 SELECT DISTINCT 语句时，使 MySQL 无法在早期阶段优化掉 distinct 操作。因此，MySQL 需要一个额外的阶段来删除所有重复的行，然后将结果发送到客户端
- removing tmp table：线程在 SELECT 语句执行完成后，正在删除内部临时表。如果 SELECT 语句未创建临时表，则不会出现此状态
- rename：线程正在执行 rename 语句重命名表
- rename result table：线程正在执行 ALTER TABLE 语句重命名表，已经创建完成新表，并正在使用新表替换旧表名称
- Reopen tables：线程获得了表锁，但是获得锁后，发现基础表结构已经被改变了。于是释放表锁，并关闭表，尝试重新打开表
- Repair by sorting：修复代码正在使用排序来创建索引
- preparing for alter table：server 正在准备执行 in-place 算法的 ALTER TABLE 语句
- Repair done：该线程已完成 MyISAM 表的多线程修复
- Repair with keycache：修复代码正在使用通过 key cache 逐个创建 key 的方法修复索引。这比通过排序索引修复的方法慢得多
- Rolling back：线程正在回滚事务
- Saving state：对于 MyISAM 表操作（如修复或分析），线程正在将新表状态保存到.MYI 文件头。状态包括：表数据行数，AUTO_INCREMENT 计数器和 key 分布之类的信息
- Searching rows for update：线程正在进行第一阶段查找所有匹配的行，然后再更新它们。如果 UPDATE 正在更改用于查找涉及的行的索引，则必须先把 update 满足匹配的行先查找出来

- Sending data：线程正在读取和处理 SELECT 语句产生的数据行，并将数据发送到客户端。因为在此状态期间发生的操作可能产生大量的磁盘访问（读取），所以它通常是给定查询的生存期内最长的运行状态
- Sending to client：server 正在向客户端写入数据包。在 MySQL 5.7.8 之前叫做“Writing to net”
- setup：线程正在执行 ALTER TABLE 操作
- Sorting for group：线程正在执行一个 GROUP BY 排序操作
- Sorting for order：线程正在执行一个 ORDER BY 排序操作
- Sorting index：线程正在排序索引页面，以便在 MyISAM 表优化操作期间实现更高效的访问
- Sorting result：对于 SELECT 语句，这类似于“Creating sort index”状态，但是针对于非临时表
- statistics：server 正在计算统计信息以优化查询执行计划。如果一个线程在这个状态很长一段时间，server 可能是磁盘执行其他工作而阻塞了统计信息的操作，也有可能发生了锁等待。
- System lock：线程调用了 mysql_lock_tables()，线程状态从未更新过。这是一个非常常见的状态，出现该状态的原因有很多。例如，线程将请求或正在等待表的内部或外部系统锁定。当 InnoDB 在执行 LOCK TABLES 期间等待表级锁时，可能会发生这种情况。如果此状态是由外部锁请求引起的，如果您不使用多个 mysqld 服务器访问同一 MyISAM 表，则可以使用--skip-external-locking 选项禁用外部系统锁。但是，默认情况下外部锁定是禁用的，因此此选项可能无效。对于 SHOW PROFILE，此状态表示线程正在请求锁定（不等待）
- update：线程准备开始更新表
- Updating：线程搜索且正在更新数据行
- updating main table：server 正在执行多表更新语句的第一部分。该状态表示正在更新第一个表，并保存列值和偏移量以用于更新其他（引用）表
- updating reference tables：server 正在执行多表更新语句的第二部分，更新其他表的匹配行

- User lock：线程将请求或正在等待通过 GET_LOCK() 调用请求的建议锁。对于 SHOW PROFILE，此状态表示线程正在请求锁定（无需等待）
- User sleep：线程已调用 SLEEP() 调用
- Waiting for commit lock：FLUSH TABLES WITH READ LOCK 语句正在获取提交锁
- Waiting for global read lock：FLUSH TABLES WITH READ LOCK 正在等待获取全局读锁或全局 read_only 系统变量设置
- Waiting for tables：线程获取到一个通知，表的底层结构已经改变，它需要重新打开表以获得新的结构。但是，要重新打开表，它必须等待，直到所有其他线程都关闭了旧数据结构的表的访问。如果另一个线程已在表中使用 FLUSH TABLES 或下列语句之一，则就会出现这个通知：
  * FLUSH TABLES tbl_name
  * ALTER TABLE
  * RENAME TABLE
  * REPAIR TABLE
  * ANALYZE TABLE
  * OPTIMIZE TABLE
- Waiting for table flush：线程正在执行 FLUSH TABLES，并且正在等待所有线程关闭所访问的表，或者线程得到一个表的底层结构已经改变的通知，它需要重新打开表以获得新的结构。但是，要重新打开表，它必须等待，直到所有其他线程都关闭了旧表结构的访问。如果另一个线程已在表中使用 FLUSH TABLES 或下列语句之一，则就会出现这个通知：
  * FLUSH TABLES tbl_name
  * ALTER TABLE
  * RENAME TABLE
  * REPAIR TABLE
  * ANALYZE TABLE
  * OPTIMIZE TABLE
- Waiting for lock_type lock：server 正在等待获得一个 THR_LOCK 锁或者从元数据锁定子系统中获取一个 MDL 锁，其

中 lock_type 表示正在等待获得的 MDL 锁的类型，THR_LOCK 只有一种（Waiting for table level lock），MDL 锁有如下几种：

* Waiting for event metadata lock
* Waiting for global read lock
* Waiting for schema metadata lock
* Waiting for stored function metadata lock
* Waiting for stored procedure metadata lock
* Waiting for table metadata lock
* Waiting for trigger metadata lock

- Waiting on cond：线程正在等待条件变为 true 的通用状态。没有特定的状态信息可用
- Writing to net：server 正在向网络写入数据包。从 MySQL 5.7.8 之后叫做“Sending to client”

1.3. Query Cache Thread States

- 以下是查询缓存相关的线程状态
  - checking privileges on cached query：server 正在检查用户是否具有访问缓存的查询结果集的权限
  - checking query cache for query：server 正在检查当前查询请求的数据是否存在于查询缓存中
  - invalidating query cache entries：因为表结构的修改导致的对应的查询缓存条目正在被标记为无效
  - sending cached result to client：server 正在从查询缓存中获取查询需要的结果肌，并将其发送到客户端
  - storing result in query cache：server 正在将从存储引擎层返回的查询结果集存储在查询高速缓存(内存)中
  - Waiting for query cache lock：当会话等待获取查询高速缓存(内存)锁时，会发生此状态。这可能发生在需要执行一些查询高速缓存(内存)操作时被其他可能导致数据变更的查询加锁，如使 INSERT、DELETE 或 RESET QUERY CACHE 等语句先执行，然后执行 SELECT 时就会导致锁等待（注意：query cache 只有一

个全局锁，任何数据修改都会导致对应记录的查询缓存失效，同时锁住整个查询缓存，其他需要使用查询缓存的语句阻塞）

1.4. Replication Master Thread States

- 以下列出在主节点 Binlog dump 线程的 State 列中可能会出现的最常见的状态。如果在主库上没有看到 Binlog Dump 线程，这意味着可能没有运行复制或者说当前没有从库连接着主库
    - Finished reading one binlog; switching to next binlog：线程已经完成读取 binlog 文件，并切换到下一个 binlog 文件
    - Master has sent all binlog to slave; waiting for more updates：线程已经从二进制日志中读取了所有剩余的更新日志，并将它们发送到从库。线程当前处于空闲状态，正在等待新的更新数据的事件写入二进制日志中
    - Sending binlog event to slave：线程已经从二进制日志中读取了一个事件，现在将其发送到从库（二进制日志由事件组成，一个事件通常是由发生更新的数据和一些其他信息组成）
    - Waiting to finalize termination：线程停止时发生的非常短暂的状态，线程正在执行停止线程相关的动作

1.5. Replication Slave I/O Thread States

- 以下列表显示了在从库 I/O 线程的 State 列中可能看到的最常见状态。此状态也出现在 SHOW SLAVE STATUS 显示的 Slave_IO_State 列中，因此您可以通过使用该语句获得当前从库 I/O 线程的状态
    - Checking master version：在建立与主库的连接之后非常短暂的状态，表示正在检查主库的版本号
    - Connecting to master：线程尝试连接到主库
    - Queueing master event to the relay log：线程已读取一个事件，并将其复制到中继日志，以便 SQL 线程进行重放
    - Reconnecting after a failed binlog dump request：线程正在尝试重新连接到主库

- Reconnecting after a failed master event read：线程正在尝试重新连接到主库，当重连连接成功时，状态将变为“Waiting for master to send event”
- Registering slave on master：在连接到主库成功之后非常短暂的状态，表示正在向主库注册从库的连接信息（如从库的 IP 和端口信息等）
- Requesting binlog dump：在与主库建立连接成功之后非常短暂的状态，使用当前的 I/O 线程位置，向主库发送从当前位置开始的二进制日志的内容的请求
- Waiting for its turn to commit：如果启用了 slave_preserve_commit_order 参数，则表示从库 I/O 线程正在等待较旧的工作线程提交数据
- Waiting for master to send event：线程已经连接到主库并且正在等待新的二进制日志事件，如果主库空闲，这可能持续很长时间。如果等待时间持续超过 slave_net_timeout 秒，则从库 I/O 线程发生超时。此时，从库 I/O 线程认为主库的连接断开，会尝试重新连接主库
- Waiting for master update：连接到主库之前的初始状态
- Waiting for slave mutex on exit：线程停止时短暂发生的状态，表示正在回收 I/O 线程的相关互斥资源
- Waiting for the slave SQL thread to free enough relay log space：如果 relay_log_space_limit 变量设置值不为 0，那么当中继日志总大小增长到超过此值时。 I/O 线程会等待，直到 SQL 线程通过重放中继日志内容并删除重放完成的中继日志以释放中继日志占用的空间，使其满足中继日志中大小不大于 relay_log_space_limit 变量的值时，I/O 线程才可以继续写入中继日志操作。
- Waiting to reconnect after a failed binlog dump request：如果二进制日志 dump 请求失败（由于断开连接），那么线程在进入 sleep 状态，此时出现此状态，然后 I/O 线程定期尝试重新连接主库。重试之间的间隔时间可以使用 CHANGE MASTER TO 语句的 MASTER_CONNECT_RETRY 选项指定

  * 要注意，从库的 I/O 线程连接主库是有心跳机制的，当主库超

过这个心跳时间没有发送新的 event 到 slave 上时，I/O 线程就对主库发起一个心跳请求，如果请求成功就重置心跳时间，当主库有新的 event 发送到 slave 时，这个心跳时间也会进行重置。心跳时间由 change master 语句的 MASTER_HEARTBEAT_PERIOD 选项设置（以秒为单位），范围 0 到 4294967 秒，分辨率（毫秒） 最小非零值为 0.001，表示 1 毫秒。将间隔设置为 0 时表示禁用心跳。 默认值是 slave_net_timeout 配置参数的二分之一。so，理论上是不会出现主从数据库正常的情况下因为主库没有写数据而导致从库 I/O 线程断开的情况。

- Waiting to reconnect after a failed master event read：读取主库 binlog 时发生错误（由于断开连接）。I/O 线程在尝试重新连接主库之前，线程正在以 CHANGE MASTER TO 语句的 MASTER_CONNECT_RETRY 选项（默认为 60）设置的秒数进行 sleep(该时间是重连失败之后的重试间隔时间)

1.6. Replication Slave SQL Thread States

- 以下列出了在从库 SQL 线程的最常见状态“状态”列
  - Killing slave：线程正在处理 STOP SLAVE 语句
  - Making temporary file (append) before replaying LOAD DATA INFILE：线程正在执行 LOAD DATA INFILE 语句，并将从库将要读取的数据添加到临时文件中
  - Making temporary file (create) before replaying LOAD DATA INFILE：线程正在执行 LOAD DATA INFILE 语句，且正在创建临时文件，临时文件中包含了从库将要读取行数据。注意：只有在 MySQL 5.0.3 之前的版本中，主库记录了原始 LOAD DATA INFILE 语句时，才能遇到此状态
  - Reading event from the relay log：线程正在从中继日志中读取事件，以便进行重放

    - Slave has read all relay log; waiting for more updates：线程已重做完所有的中继日志文件中的所有事件，正在等待 I/O 线程向中继日志中写入新的事件
    - Waiting for an event from Coordinator：从库使用多线程复制时（slave_parallel_workers 大于 1），此状态表示一个 slave works 线程正在等待协调器线程（Coordinator 线程）分配日志事件
    - Waiting for slave mutex on exit：线程停止时发生的非常短暂的状态
    - Waiting for Slave Workers to free pending events：当 Workers 线程处理的事件的总数量大小超过 slave_pending_jobs_size_max 系统变量的大小时，会发生等待操作（协调器线程不进行分配事件给 worker 线程）。当 Workers 线程处理的事件的总数量大小低于 slave_pending_jobs_size_max 限制时，协调器恢复调度。只有当 slave_parallel_workers 设置为大于 0 时，此状态才会出现
    - Waiting for the next event in relay log：“Reading event from the relay log”状态之前的初始状态
    - Waiting until MASTER_DELAY seconds after master executed event：SQL 线程已读取事件，但并没有进行应用，而是正在等待从库设置的延迟复制时间失效。此延迟时间使用 CHANGE MASTER TO 的 MASTER_DELAY 选项设置
- SQL 线程的 Info 列也可以显示语句的文本。这表示线程已经从中继日志中读取了一个事件，并从中提取了 SQL 语句，当前可能正在执行这个语句对应的事件。

1.7. Replication Slave Connection Thread States

- 这些线程状态发生在从库上，但是是与连接主库的连接线程相关联，而不是与 I/O 或 SQL 线程相关联
    - Changing master：线程正在处理 CHANGE MASTER TO 语句
    - Killing slave：线程正在处理 STOP SLAVE 语句

- Opening master dump table：此状态发生在主库创建 dump 表之后
- Reading master dump table data："Opening master dump table"状态之后出现的状态，表示正在从主库 dump 表读取数据
- Rebuilding the index on master dump table： “Reading master dump table data”状态之后出现的状态，表示正在重建主库 dump 表索引

1.8. Event Scheduler Thread States

- 以下列出了事件调度程序线程的状态
  - Clearing：调度程序线程正在停止执行事件
  - Initialized：调度程序线程已初始化完成，将要执行调度事件
  - Waiting for next activation：调度程序具有非空事件队列时，正在等待未来某个时间点激活队列中的某个事件，以便进行调度并执行
  - Waiting for scheduler to stop：线程发出 SET GLOBAL event_scheduler = OFF 并等待调度程序停止
  - Waiting on empty queue：调度程序的事件队列为空，因此调度程序处于休眠状态