aws

超擴展 Aurora MySQL 相容以處理突然的流量成長

AWS 方案指引

AWS 方案指引: 超擴展 Aurora MySQL 相容以處理突然的流量成長

Table of Contents

超擴展 Aurora MySQL 相容以處理突然的流量成長

Oliver Francis、Shyam Sunder Rakhecha 和 Vikram singh Rai，Amazon Web Services (AWS)

2022 年 10 月

您的網際網路業務可能會面臨前所未有的超速成長，而您現有的應用程式基礎設施無法應付。這種情況可能是由於各種因素而發生的，例如有關您的產品的廣告引起了超出預期的興趣，或者客戶購買模式突然從面對面轉變為線上。

為了解決這些意外負載，您必須超擴展您的基礎設施。擴展應用程式層需要新增更多伺服器或 Pod。但是，實現超擴展最困難的部分是資料庫。您可以將資料庫執行個體擴展至最大的可用執行個體大小，但這可能無法解決您的問題。

如果您在 Amazon Aurora MySQL 相容版上執行資料庫工作負載，本指南提供了您可以實作的建議，以超擴展資料庫來滿足突然的超速成長。並非所有這些建議都是長期最佳實務。如果您預期超速成長且有時間規劃解決，請參閱 資源 一節中列出的部落格文章。這些文章可以協助您實作長期最佳實務。另一方面，如果您面臨意外的超速成長，本指南將協助您管理負載並實現合理的穩定性，同時為您的業務規劃和實作長期解決方案。

請注意，本指南中概述的建議需要您的開發團隊的實作協助。

目標業務成果

本指南中介紹的方法將協助您執行下列操作：

- 在出現意外的超速成長時穩定您的業務。實現足夠的穩定性以實作長期最佳實務進行超速成長。
- 防止經濟損失。超擴展環境中的突然中斷可能會導致客戶在您的應用程式上執行的業務交易量下降。在某些情況下，這可能會導致巨大的經濟損失。穩定的超擴展環境是防止長時間中斷導致業務損失的關鍵。

管理連線

隨著應用程式需求的成長，前端流量也會增加。在一般案例中，您可以在應用程式層設定自動擴展來處理此類爆量的傳入流量。因此，應用程式層開始自動擴展，並新增更多應用程式伺服器 (執行個體) 來滿足流量的增加。由於所有應用程式伺服器都已預先設定資料庫連線集區設定，因此資料庫的傳入連線數量與新部署的執行個體成比例成長。

例如，20 個應用程式伺服器設定有 200 個資料庫連線，每個應用程式伺服器將總共開啟 4,000 個資料庫連線。如果應用程式集區縱向擴展至 200 個執行個體 (例如，在尖峰時間)，總連線數將達到 40,000 個。在典型工作負載下，大多數連線可能處於閒置狀態。但是，連線的激增可能會限制您的 Amazon Aurora MySQL 相容版資料庫的擴展能力。這是因為即使閒置連線也會耗用記憶體和其他伺服器資源，例如檔案描述元。Aurora MySQL 相容版通常使用比 MySQL 社群版更少的記憶體來維護相同數量的連線。但是，閒置連線的記憶體用量仍然不是零。

組態變數

您可以使用以下兩個主要伺服器組態變數來控制資料庫允許的傳入連線數：`max_connections` 和 `max_connect_errors`。

組態變數 max_connections

組態變數 `max_connections` 限制每個 MySQL 執行個體的資料庫連線數。最佳實務是將其設定為稍高於您預期在每個資料庫執行個體上開啟的連線數上限。

如果您也啟用 `performance_schema`，請格外小心 `max_connections` 設定。效能結構描述記憶體結構根據伺服器組態變數自動調整大小，包括 `max_connections`。您設定的變數越高，效能結構描述使用的記憶體就越多。在極端情況下，這可能會導致較小執行個體類型出現記憶體不足問題。請注意，啟用 Performance Insights 將自動啟用效能結構描述。

組態變數 max_connect_errors

組態變數 `max_connect_errors` 決定所指定用戶端主機允許的連續中斷的連線請求數量。如果用戶端主機超過指定的連續失敗連線嘗試次數，伺服器會封鎖它。該用戶端的進一步連線嘗試會產生錯誤。

```
Host 'host_name' is blocked because of many connection errors. Unblock with
'mysqladmin flush-hosts'
```

如果您遇到「主機遭到封鎖」錯誤，請避免增加max_connect_errors變數的值。相反，請調查 `aborted_connects` 狀態變數和 `host_cach` 資料表中的伺服器診斷計數器。使用收集的資訊來識別

並修正遇到連線問題的用戶端。另請注意，如果 `skip_name_resolve` 設定為 1 (預設)，則此參數無效。

如需有關下列項目的詳細資訊，請參閱 MySQL 參考手冊： 有關以下內容的詳細信息，請參閱《MySQL 參考手冊》：

- Max_connect_errors 變數
- 「主機已封鎖」錯誤
- Aborted_connects 狀態變數
- Host_cache 資料表

實作連線集區

擴展事件可能會新增更多應用程式伺服器，這反過來又可能導致資料庫伺服器超過完整載入的作用中連線數。在應用程式伺服器與資料庫之間新增連線集區或代理層就像漏斗一樣，減少了資料庫上的連線總數。代理的主要目的是透過多工方式重複使用資料庫連線。

一方面，代理透過控制的連線數連接至資料庫。另一方面，代理接受應用程式連線。它還提供其他功能，例如查詢快取、連線緩衝、查詢重寫和路由以及負載平衡。連線集區層需要設定為保持資料庫的連線數上限低於完整載入數。Amazon RDS Proxy 是您可以為此用途實作的全受管代理。Amazon RDS Proxy 不需要對大多數應用程式進行任何程式碼變更，並且您不需要管理任何額外基礎設施來實作解決方案。

您也可以探索下列可與 Aurora MySQL 相容搭配使用的第三方代理：

- ProxySQL
- MariaDB MaxScale
- ScaleARC
- Heimdall Data

避免連線風暴

考慮在資料庫過載或複本落後於主節點太遠的情況下連線集區的行為方式。設定代理伺服器或連線集區時，確保不會因資料庫回應緩慢 (由基礎硬體或儲存問題或資料庫資源限制造成) 而重設整個連線集區。

突然啟動數百個連線會產生連線風暴，因為大量對資料庫的新連線請求都同時起始。風暴是資源密集型。在 MySQL 中建立新的資料庫連線是一項昂貴的操作，因為後端會交換多個網路封包以進行初始交握、產生新程序、配置記憶體、處理驗證等。如果在短時間內收到大量請求，資料庫可能沒有回應。

MySQL 有一種機制可防止連線請求中出現此類激增。back_log 變數可以設定為在 MySQL 暫時停止回答新請求之前短時間內可以堆疊的請求數。此值由連線處理執行緒強制執行，此執行緒本身可能會被連線風暴淹沒。如需詳細資訊，請參閱 MySQL Reference Manual。

如果您的連線設定為在資料庫速度緩慢時重設，您將一次又一次地初始化循環。同樣，如果您預期一天中的某些時間 (例如，股市開盤時) 資料庫流量會突然增加，請預熱您的連線集區，以便您不會嘗試在高流量負載開始的同時開啟許多連線。

調校查詢工作負載

妥善調校的工作負載將協助您實現超速成長的穩定解決方案。如果您的工作負載未妥善調校，無論您使用的 Amazon Aurora 叢集的功能如何，您都會遇到瓶頸，從而降低效能並影響應用程式的使用者體驗。最佳實務是從一開始就制定一個程序，協助您識別和調校系統中有問題的查詢。

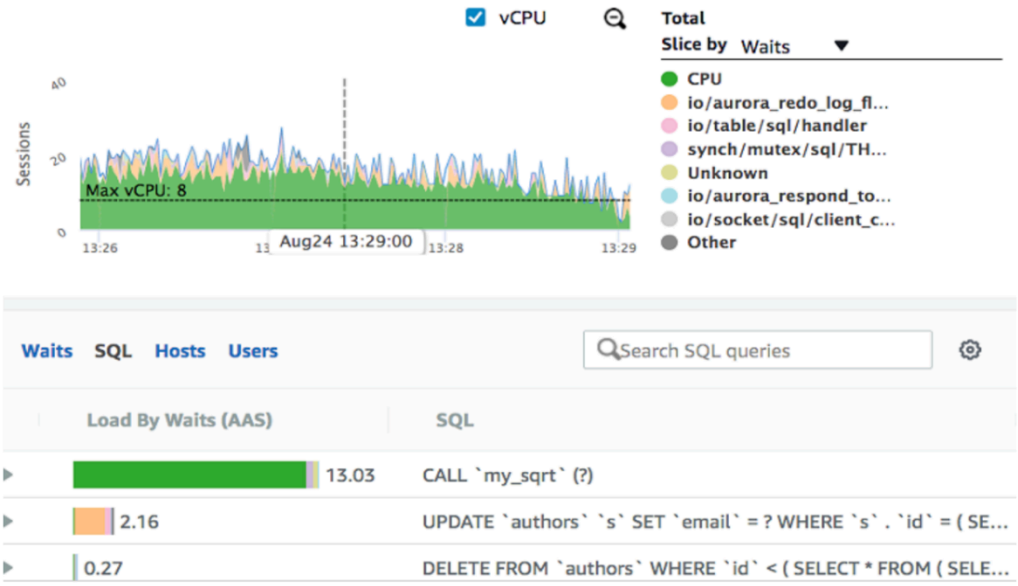
追蹤和調校工作負載中有問題的查詢

發現超速成長時，妥善調校的工作負載就已成功一半。若要了解 Aurora 叢集面臨的即時工作負載和效能問題的性質，請確保您的團隊從寫入器和讀取器執行個體收集有問題的查詢。需要調校這些有問題的查詢，以使您的工作負載以最佳狀態執行。Amazon Aurora MySQL 為您提供了兩種方法來實現此目的：

- Performance Insights
- 慢速查詢日誌

使用 Performance Insights

Performance Insights 根據平均作用中工作階段數 (AAS) 追蹤 Aurora 寫入器或讀取器執行個體上的負載。AAS 值是透過使用取樣和等待 CPU 來取得查詢工作負載並進行處理的作用中工作階段數來計算的。Performance Insights 提供圖形介面，您可以透過等待作用中工作階段來檢查導致最高負載的 SQL 陳述式。



在上一個螢幕擷取畫面中，對儲存程序 `my_sqrt` 的呼叫導致平均 13.03 個工作階段等待其負載得到處理。合乎邏輯的下一步是調校此程序。您應識別讀取器和寫入器中導致各自執行個體負載的 SQL 陳述式，並進行調校以提高效能，直到 AAS 值下降並保持在 Performance Insights 中的 vCPU 數上限虛線以下。如果您的調校工作已達到上限，但仍看到 AAS 超過 vCPU 數上限線，則您可以選擇更大的執行個體類別來處理您的工作負載。在沒有先嘗試調校查詢工作負載的情況下，請勿選擇更大的執行個體，因為不斷增長的流量將開始暴露由工作負載中的錯誤查詢造成的故障線。

使用慢速查詢日誌並將其發佈至 CloudWatch

慢速查詢日誌是原生 MySQL 功能，是 Performance Insights 的補充。最佳實務是同時使用這兩種方法來提前解決可能對您的執行個體造成嚴重破壞的有問題的查詢。慢速查詢日誌記錄比動態變數 `long_query_time` 更慢的任何查詢。無需重新啟動叢集執行個體即可設定此變數。

為了在調校練習中提供靈活性和隔離性，請為寫入器和讀取器執行個體使用單獨的參數群組。如果您使用讀寫分割，這一點尤其重要。根據您的需求為叢集執行個體中的 `long_query_time` 設定適當的限制。調校負載時，您能以 `long_query_time` 變數中的主動值為目標，因為您能以毫秒級設定閾值。透過高度並行和妥善調校的工作負載，幾乎所有查詢都應在幾毫秒內執行。

在您設定 Amazon Aurora MySQL 相容版以將慢速查詢記錄至檔案時，Aurora MySQL 相容會將慢速查詢日誌寫入 Aurora MySQL 相容檔案系統並保留 24 小時。若要將慢速查詢日誌保留更長時間以進行分析，請將其發佈至 Amazon CloudWatch。您也可以建置 CloudWatch 儀表板來監控慢速查詢。如需詳細資訊，請參閱部落格文章建立 Amazon CloudWatch 儀表板來監控 Amazon RDS 和 Amazon Aurora

MySQL。除了分析 Amazon CloudWatch 上的慢速查詢之外，您還可以使用 Percona Toolkit 中的工具 pt-query-digest 來分析慢速查詢日誌。

您也可以選擇自動執行此下載和分析查詢的程序，以提高團隊的效率。您的團隊應檢查頻繁執行且間隔較長的查詢，並排定調校的優先順序。目標是在慢速查詢日誌中記錄很少的查詢，並且在了解和調校工作負載時，您可以減少 `long_query_time` 以變得更積極。

查詢調校指引

在識別工作負載中有問題的查詢後，應調校每個查詢。使用下列有關調校的指引來協助您的工作負載更有效率地執行。

最小化掃描的資料列數

儘管看起來很簡單，但在調校查詢時這是一個很好的建議。使用 EXPLAIN 陳述式，並檢閱資料列欄以查看優化工具在每個聯結處掃描的資料列數。嘗試透過建立最佳索引來減少掃描的資料列數，然後重新解釋您的查詢以確認您的工作。如需詳細資訊，請參閱 MySQL 文件。

如果您使用分割的資料表，請永遠使用啟用分割區清除的 WHERE 子句進行查詢，以便優化工具無需掃描每個分割區。如果您的 WHERE 子句包含已分割資料欄的常數，則優化工具知道要尋找的分割區，這會使您的查詢更有效率。

此建議的另一個方面是資料庫的設計。查詢中的資料表越少，查詢的速度就越快。如果您可以對資料庫設計反正規化，則可以讓優化工具掃描更少的資料列，從而提高查詢效能。

最大限度地減少暫時資料表用量和磁碟上的暫時資料表

如果 Aurora MySQL 相容優化工具無法直接從索引取得所需的查詢結果，則會同時在 RAM 和磁碟上建立暫時資料表。因此，調校的很大一部分是具有適合您的工作負載的正確索引。但是，工作負載中可能存在不能僅依賴索引的查詢，因此某些操作可能在暫時檔案中執行。只要將其保持在最低限度，並確保在磁碟上建立很少的資料表，就可以了。當暫時資料表太大而無法放在記憶體中時，MySQL 會建立磁碟資料表。MySQL 用於檢查內部暫時資料表大小的邏輯是兩個變數值 tmp_table_size 和 max-heap-table-size 中較小的一個值。您可以根據工作負載將這些變數調校為最佳值，以便在無法阻止暫時資料表時，僅在極少數情況下將這些資料表推送至磁碟。

避免檔案排序

如果您的工作負載具有大量 `ORDER BY` 查詢，解決這些查詢的最佳方法是在資料表上使用正確索引。確保您的多資料欄索引設計良好，以避免在檔案中排序。如果未使用常數掃描前面的資料欄，則無法對

資料欄進行排序 (in、>、<、!= 和 BETWEEN 不允許對右側的下一資料欄進行排序)。在 MySQL 中排序的最佳方法是放置多資料欄索引，該索引會將包含查詢中提供的常數值的資料欄定位至連續結構中排序資料欄的左側。如果實在沒有別的辦法，在沒有檔案排序的情況下您的查詢無法傳回結果，請將排序移至應用程式。

避免在高度並行下執行彙總查詢

您的工作負載可能具有少量彙總查詢來滿足應用程式內的某些功能。此使用案例需要非常謹慎。InnoDB 引擎適合適當的線上交易處理 (OLTP) 負載，但即使是幾個高度並行的分組查詢也會對 CPU 造成很大的負擔，並且會迅速降低叢集的效能。若要解決需要彙總結果集的使用案例，請將就緒資料預先彙總至讀取的資料表，以便完全避免分組查詢。

測試您的查詢的並行性

調校個別查詢時，記住這些查詢在 Aurora MySQL 相容中的多個 vCPU 上同時執行。單次執行時，您的查詢可能會在測試環境中執行幾毫秒。但這還不是全部。請務必在生產叢集上使用預期的並行層級測試您的查詢並對效能進行基準測試。僅當查詢滿足並行目標時，才將查詢發行至生產。確保在測試指令碼中使用優化工具 `hint sql_no_cache`，以避免從快取中擷取結果。您可以使用 mysqlslap 等工具來執行並行測試並對結果進行基準測試。

調校寫入工作負載

實作負載平衡並釋放寫入器執行個體將有助於您的寫入工作負載在高峰值期間更好地執行。若要在高度並行下獲得更佳寫入效能，請遵循下列其他步驟。

將參考完整性移至應用程式層

雖然參考完整性檢查很重要，但對於超擴展來說，其可能會對您的負載產生不利影響。對於每次寫入，必須在寫入本身執行之前執行額外掃描，這會導致效能不佳。如果您的應用程式需要嚴格的完整性檢查，請將其建置到應用程式層以防止其限制您的寫入。

避免使用重度主索引鍵

讓您的主索引鍵保持輕鬆。InnoDB 儲存引擎會將主索引鍵附加至您在資料表中建立的每個其他索引。當主索引鍵很大時，它會影響索引的大小。如果主索引鍵很大，資料頁面的儲存和擷取將會變慢。常見範例是使用通用唯一識別碼作為主索引鍵。如果您的目標是超擴展環境中的效能，那麼這不是一個好方法。

使用分割區交換將資料載入到分割的資料表

如果您要將大量資料寫入分割的資料表，則 LOAD DATA FROM S3 和分割區交換的組合可以提高效能，因為插入時不會存取主資料表。分割區交換涉及資料定義語言 (DDL)，它會在資料表上放置中繼資料鎖定。請確保在資料表上執行的查詢最少或沒有時執行此操作，以便分割區交換 DDL 無需等待即可取得中繼資料鎖定。交換本身只需幾毫秒即可完成。

移除未使用的索引

InnoDB 根據資料的成長來優化查詢計畫，最好檢查資料庫中未使用的索引並將其移除。在應用程式將資料寫入資料表時，未使用的索引會耗用 IO。檢查未使用的索引清單，並確認其不是在極少數情況下使用的索引，例如季度報告。另請注意，某些索引用於強制唯一性或排序，也必須予以考慮。

確保有效地清除舊資料列版本

在多版本並行控制 (MVCC) 的 InnoDB 實作中，當修改記錄時，要修改的資料的目前 (舊) 版本首先記錄為復原日誌中的復原記錄。不斷增長的歷史記錄清單長度 (HLL) 值表示 InnoDB 垃圾回收執行緒 (清除執行緒) 跟不上寫入工作負載，或清除被長時間執行的查詢或交易封鎖。在垃圾回收已封鎖或延遲時，資料庫可能會出現嚴重的清除遲時，從而對查詢效能產生負面影響。您可以使用下列建議來優化清除程序。

- 保持較小的交易。
- 對於讀取查詢，請使用 READ COMMITTED 隔離層級。
- 增加清除執行緒數量 (innodb_purge_threads 和 innodb_purge_batch_size)。請注意，調校這些參數需要重新開機。
- 定期監控 HLL，並解決任何阻止垃圾回收進行的工作負載問題。

確保日誌記錄不會導致其他爭用

一般查詢日誌記錄用戶端連線和中斷連線，以及伺服器依接收順序接收的所有陳述式。啟動時，日誌記錄是同步的，這可能會導致忙碌系統的效能大幅下降。除非需要，我們建議停用一般日誌。

慢速查詢日誌記錄執行時間超過 long_query_time 秒數的陳述式，預設設定為 10 秒。在此設定設為 0 時，所有陳述式都會同步記錄，這可能會導致忙碌資料庫的效能下降。

消減負載

縮短回應時間並增加關鍵工作流程的可用資源可能需要消除無關負載。本節介紹的許多解決方案都是權衡決策。其會對應用程式產生影響，必須仔細考慮。在下列情況下，請考慮使用這些解決方案：

- 您已處於最大執行個體上，尤其對於主要可寫入資料庫執行個體。
- 作為最後的手段，在短期內提供足夠的預留空間來實作其他變更。

立即變更包括下列項目：

- 將非關鍵讀取流量移離主要資料庫執行個體。
- 封存或清除舊資料。
- 移除參考完整性。
- 停用 binlog (如果使用中)。
- 延遲非關鍵擷取、轉換、載入 (ETL) 程序。
- 暫停或降級不必要的應用程式功能。

在執行這些動作之前，根據長期業務目標和風險進行評估。

分隔讀寫工作負載

在執行由 MySQL 提供支援的應用程式時，常見技術是將讀取操作從寫入器 (主要) 資料庫執行個體卸載到一或多個唯讀資料庫複本上。透過卸載讀取，您可以減少主要資料庫執行個體上的整體負載並為寫入騰出空間。請確保僅針對不依賴複本的立即 read-after-write 一致性的讀取。更有效的方法是將所有讀取流量移至複本，並規劃在複寫延遲時重試 read-after-write。存在可以卸載的獨立讀取工作負載，例如報告服務。其他讀取將需要在應用程式層級進行變更，其中發出讀取的內容是眾所周知的。

另一種方法是實作資料庫代理解決方案作為應用程式與資料庫之間的媒介，它可以提供讀寫分割和查詢路由的功能，而無需應用程式感知。ProxySQL、MariaDB MaxScale、ScaleARC 和 Heimdall Data 是一些可用的 MySQL 相容代理解決方案。這些產品提供多種其他功能，例如快取或連線多工。在您遇到流量突然激增並在應用程式堆疊中實作新技術時，我們建議您從分割讀取/寫入查詢的基本功能開始，並在使用可能產生意外副作用的其他功能之前測試行為和效能。

封存或清除較舊資料

另一種提高資料庫效能的技術是將歷史資料卸載至另一個資料表、資料庫或 Amazon Simple Storage Service (Amazon S3)。許多資料庫保留應用程式整個歷史記錄的所有內嵌資料。在一般情況下，在典

型的面向使用者的應用程式中，這可讓使用者查看其所有歷史訂單。需求突然激增時，應用程式上的許多作用中使用者可能是新使用者或專注於下新訂單。如果歷史資料在線上駐留在包含數十億資料列的單一資料表中，則情況會更加複雜。此資料表還可能具有大型索引。資料表資料和索引都以樹狀目錄結構儲存。資料表中的項目越多，樹狀目錄的層級就越高，這需要更多 I/O 操作來存取資料列。這會增加尋找個別記錄的存取時間。更重要的是，它會導致索引的大量不必要的部分駐留在資料庫頁面快取 (InnoDB 緩衝集區) 中。

將較舊資料封存至個別資料表、個別資料庫或 Amazon S3 可以減少最終使用者的存取時間、釋放寶貴的快取並提高整體應用程式效能。在事件發生之前封存較舊資料 (例如，僅保留 30 天或 90 天) 會限制關鍵資料表上的資料表和索引的大小。此變更需要修改應用程式，以僅針對明確要求查看歷史資料的一小部分使用者從次要位置查詢較舊資料。

延遲非關鍵 ETL 程序

在超擴展條件下，從主要資料庫系統中擷取 ETL 程序可能會對高度交易和並行工作負載帶來穩定性風險。ETL 程序具有下列特性：

- 長時間執行的交易
- 廣泛的封鎖
- CPU、記憶體和 I/O 耗用
- 與服務關鍵最終使用者路徑的交易工作負載爭用。

可變的或不可預測的 ETL 程序 (例如，`INSERT INTO ... SELECT FROM ...;`) 等查詢) 會新增至負載和爭用的整體可變性，從而增加穩定性風險。

在可能的情況下，延遲或降低 ETL 程序執行的頻率，尤其是在其不提供關鍵功能的情況下。對於重要 ETL 程序，進行修改，以便其在有界限的工作單位中操作，例如處理固定資料列數的批次 (例如，使用 `LIMIT` 子句)、使用獨立的交易，或者在較長的時間內提取少量資料，以減少資料庫執行個體的峰值資源需求。

關閉不太重要的應用程式功能

在處理大量請求時適當地降低使用者體驗或移除非核心功能，可以為關鍵功能節省資料庫資源。這可能需要對客戶體驗進行稍微變更，但不同的流程比網站關閉要好。也許可以暫時停用永遠進行資料庫呼叫的網站首頁上的個人化設定。或者，您可以在選擇產品時停止向回頭客提供自訂優惠。暫時關閉功能可以在不需要存取資料庫的情況下快取或交付頁面。

其他範例包括具有輪詢機制的前端，用於檢查需要資料庫呼叫的資料變更。降低輪詢頻率將立即導致資料庫呼叫減少。需要分頁或為使用者提供擷取遠離熱門搜尋結果的排序結果集的選項的使用者介面需要越來越昂貴的資料庫呼叫。限制結果集中的頁數以保護資料庫免受最昂貴的資料庫呼叫的影響。在應用程式層使用功能旗標，以讓營運團隊在應用程式或資料庫負載增加時關閉或降級功能。

參數調校

除了與連線相關的參數之外，您還可以調校一些參數，以便在面對超成長時改善 Amazon Aurora MySQL 相容版本叢集的效能。變更任何資料庫參數時，請務必使用下列最佳實務：

- 每次變更一個參數，以便您可以衡量變更的影響。
- 某些參數變更後可能需要預熱一段時間才能顯現效果。當您觀測和測量 Aurora MySQL 相容叢集的效能時考慮這一點。
- 避免不正確的參數值，這可能會阻止資料庫執行個體啟動。

這些參數包括：

- `sync_binlog`
- `table_open_cache`
- `innodb_sync_array_size`
- `innodb_flush_log_at_trx_commit`
- `autocommit`
- `tmp_table_size`
- `max_heap_table_size`

如需設定這些參數的指導方針，請參閱 AWS 文件中的 Aurora MySQL 組態參數、Aurora MySQL MySQL 中的 MySQL 功能建議，以及 Aurora MySQL 中的暫時資料表行為。

資源

- 採用雲端原生架構之旅系列：#1 – 為應用程式的超速成長做好準備 (部落格文章)
- 採用雲端原生架構之旅系列：#2 – 最大化系統輸送量 (部落格文章)
- 使用 Amazon RDS Proxy
- 快取策略
- 開啟和關閉 Performance Insights
- InnoDB 儲存引擎
- Aurora 複本
- MariaDB 連接器/J
- MariaDB MaxScale
- ProxySQL
- Heimdall Data

文件歷史紀錄

下表描述了本指南的重大變更。如果您想收到有關未來更新的通知，可以訂閱 RSS 摘要。

變更	描述	日期
初次出版	—	2022 年 10 月 5 日

本文為英文版的機器翻譯版本，如內容有任何歧義或不一致之處，概以英文版為準。