vmagent 叢集設定

翻成繁體中文並詳細說明:Scraping big number of targets #
A single vmagent instance can scrape tens of thousands of scrape targets. Sometimes this isn’t enough due to limitations on CPU, network, RAM, etc. In this case scrape targets can be split among multiple vmagent instances (aka vmagent horizontal scaling, sharding and clustering). The number of vmagent instances in the cluster must be passed to -promscrape.cluster.membersCount command-line flag. Each vmagent instance in the cluster must use identical -promscrape.config files with distinct -promscrape.cluster.memberNum values in the range 0 ... N-1, where N is the number of vmagent instances in the cluster specified via -promscrape.cluster.membersCount. For example, the following commands spread scrape targets among a cluster of two vmagent instances:

/path/to/vmagent -promscrape.cluster.membersCount=2 -promscrape.cluster.memberNum=0 -promscrape.config=/path/to/config.yml ...
/path/to/vmagent -promscrape.cluster.membersCount=2 -promscrape.cluster.memberNum=1 -promscrape.config=/path/to/config.yml ...
Shell
The -promscrape.cluster.memberNum can be set to a StatefulSet pod name when vmagent runs in Kubernetes. The pod name must end with a number in the range 0 ... promscrape.cluster.membersCount-1. For example, -promscrape.cluster.memberNum=vmagent-0.

By default, each scrape target is scraped only by a

single vmagent instance in the cluster. If there is a need for replicating scrape targets among multiple vmagent instances, then -promscrape.cluster.replicationFactor command-line flag must be set to the desired number of replicas. For example, the following commands start a cluster of three vmagent instances, where each target is scraped by two vmagent instances:

```
/path/to/vmagent -promscrape.cluster.membersCount=3 -promscrape.cluster.replicationFactor=2 -promscrape.cluster.memberNum=0 -promscrape.config=/path/to/config.yml ...
/path/to/vmagent -promscrape.cluster.membersCount=3 -promscrape.cluster.replicationFactor=2 -promscrape.cluster.memberNum=1 -promscrape.config=/path/to/config.yml ...
/path/to/vmagent -promscrape.cluster.membersCount=3 -promscrape.cluster.replicationFactor=2 -promscrape.cluster.memberNum=2 -promscrape.config=/path/to/config.yml ...
```

Shell

Every vmagent in the cluster exposes all the discovered targets at http://vmagent:8429/service-discovery page. Each discovered target on this page contains its status (UP, DOWN or DROPPED with the reason why the target has been dropped). If the target is dropped because of sharding to other vmagent instances in the cluster, then the status column contains -promscrape.cluster.memberNum values for vmagent instances where the given target is scraped.

The /service-discovery page provides links to the corresponding vmagent instances if -promscrape.cluster.memberURLTemplate command-line flag is set. Every occurrence of %d inside the -promscrape.cluster.memberURLTemplate is substituted

with the -promscrape.cluster.memberNum for the corresponding vmagent instance. For example, -promscrape.cluster.memberURLTemplate='http://vmagent-instance-%d:8429/targets' generates http://vmagent-instance-42:8429/targets url for vmagent instance, which runs with -promscrape.cluster.memberNum=42.

Note that vmagent shows up to -promscrape.maxDroppedTargets dropped targets on the /service-discovery page. Increase the -promscrape.maxDroppedTargets command-line flag value if the /service-discovery page misses some dropped targets.

If each target is scraped by multiple vmagent instances, then data deduplication must be enabled at remote storage pointed by -remoteWrite.url. The -dedup.minScrapeInterval must be set to the scrape_interval configured at -promscrape.config. See these docs for details.

The -promscrape.cluster.memberLabel command-line flag allows specifying a name for member num label to add to all the scraped metrics. The value of the member num label is set to -promscrape.cluster.memberNum. For example, the following config instructs adding vmagent_instance="0" label to all the metrics scraped by the given vmagent instance:

/path/to/vmagent -promscrape.cluster.membersCount=2 -promscrape.cluster.memberNum=0 -promscrape.cluster.memberLabel=vmagent_instance
Shell
See also how to shard data among multiple remote storage systems.

大量目標的爬取（Scraping Big Number of Targets）

單個 `vmagent` 實例可以爬取數萬個 `scrape target`（爬取目標）。然而，受限於 **CPU、網路、記憶體（RAM）等資源**，單個 `vmagent` 可能無法處理所有的爬取目標。在這種情況下，可以將爬取目標拆分（sharding）至 **多個 `vmagent` 實例**，這種方法稱為 `vmagent` **水平擴展（horizontal scaling）、分片（sharding）或叢集（clustering）**。

---

1. 設定 `vmagent` 叢集

要使用 `vmagent` 叢集，必須透過 `-promscrape.cluster.membersCount` 指定 `vmagent` **叢集的總數量**，並確保每個 `vmagent` 實例使用 **相同的 `-promscrape.config` 設定檔**，但分別指定不同的 `-promscrape.cluster.memberNum` 值。該數值的範圍為 `0 ~ N-1`，其中 `N` 為 `vmagent` 叢集的總數量。

示例：建立兩個 `vmagent` 節點的叢集

假設要將爬取目標分配給兩個 `vmagent` 實例，可以使用以下指令：

```
/path/to/vmagent -promscrape.cluster.membersCount=2 -
promscrape.cluster.memberNum=0 -promscrape.config=/path/to/config.yml ...
/path/to/vmagent -promscrape.cluster.membersCount=2 -
promscrape.cluster.memberNum=1 -promscrape.config=/path/to/config.yml ...
```

在 Kubernetes 中使用 `StatefulSet`

當 `vmagent` 執行於 **Kubernetes** 環境時，可以將 `-promscrape.cluster.memberNum` 設定為 **StatefulSet 的 Pod 名稱**。該 Pod 名稱必須以範圍 `0 ~ promscrape.cluster.membersCount-1` 的數字結尾，例如：

```
-promscrape.cluster.memberNum=vmagent-0
```

這樣，每個 `vmagent` 都能根據 Pod 名稱自動分配 `memberNum`。

---

2. 設定 `vmagent` 爬取目標的複製數

預設行為

在 `vmagent` 叢集中，**每個爬取目標（scrape target）只會由其中一個 `vmagent` 負責爬取**。這樣可以減少重複請求，提高爬取效率。

啟用多個 `vmagent` 爬取相同的目標

如果需要 **多個 `vmagent` 節點爬取相同的目標（例如提高可用性）**，可以使用 `-promscrape.cluster.replicationFactor` 指令來指定 **複製數量**。

示例：建立三個 `vmagent` 節點的叢集，每個目標被兩個節點爬取

```
/path/to/vmagent -promscrape.cluster.membersCount=3 -
promscrape.cluster.replicationFactor=2 -promscrape.cluster.memberNum=0 -
promscrape.config=/path/to/config.yml ...
/path/to/vmagent -promscrape.cluster.membersCount=3 -
promscrape.cluster.replicationFactor=2 -promscrape.cluster.memberNum=1 -
promscrape.config=/path/to/config.yml ...
/path/to/vmagent -promscrape.cluster.membersCount=3 -
promscrape.cluster.replicationFactor=2 -promscrape.cluster.memberNum=2 -
promscrape.config=/path/to/config.yml ...
```

在此設定下：

- `vmagent-0` 和 `vmagent-1` 可能負責相同的爬取目標
- `vmagent-1` 和 `vmagent-2` 可能負責相同的爬取目標
- `vmagent-2` 和 `vmagent-0` 可能負責相同的爬取目標

這樣可以確保每個目標至少被 **兩個 `vmagent` 實例爬取**，提高數據的可靠性。

3. 監控 `vmagent` 叢集

檢視 `vmagent` 發現的所有爬取目標

每個 `vmagent` 在 `http://vmagent:8429/service-discovery` 頁面上顯示 **所有已發現的爬取目標**，包含其當前狀態：

- `UP`（正常運行）
- `DOWN`（無法爬取）
- `DROPPED`（目標被丟棄，並顯示丟棄的原因）

如果某個爬取目標因為 **負載分片（sharding）** 被指派給另一個 `vmagent`，則 `status` 欄位會顯示該目標的 `-promscrape.cluster.memberNum` 值，表示哪個 `vmagent` 正在負責爬取該目標。

設定 `-promscrape.cluster.memberURLTemplate`

如果 `-promscrape.cluster.memberURLTemplate` 指定了一個 `vmagent` **的 URL 模板**，則 `/service-discovery` 頁面會自動產生對應 `vmagent` 的連結。

例如：

```
-promscrape.cluster.memberURLTemplate='http://vmagent-instance-
%d:8429/targets'
```

則 `vmagent` `memberNum=42` 會對應到：

```
http://vmagent-instance-42:8429/targets
```

調整 `-promscrape.maxDroppedTargets`

`/service-discovery` 頁面 **預設最多顯示** `-promscrape.maxDroppedTargets` **個被丟棄的目標**。如果想顯示更多被丟棄的目標，可以提高該值，例如：

```
-promscrape.maxDroppedTargets=1000
```

---

4. 避免重複數據：啟用去重（Deduplication）

當 **每個爬取目標被多個 `vmagent` 爬取** 時，需要 **在遠端存儲端（remote storage）啟用數據去重**，避免數據重複。

- **`-remoteWrite.url`**：設定遠端存儲的地址
- **`-dedup.minScrapeInterval`**：應該與 **`-promscrape.config`** 中的 **`scrape_interval`** **相同**

---

5. 設定 `vmagent` 的標籤

可以使用 **`-promscrape.cluster.memberLabel`** 設定 **`vmagent`** **的標籤名稱**，並自動在所有爬取的指標數據中添加對應的 **`vmagent`** 實例編號。

示例：在指標中加入 `vmagent_instance` 標籤

```
/path/to/vmagent -promscrape.cluster.membersCount=2 -
promscrape.cluster.memberNum=0 -
promscrape.cluster.memberLabel=vmagent_instance
```

則該 **`vmagent`** 爬取的所有數據都會自動帶上：

```
vmagent_instance="0"
```

如此一來，便可區分 **不同 `vmagent` 實例所爬取的數據**。

6. 資料分片至不同的遠端存儲

如果需要將數據 **分片（sharding）至不同的遠端存儲系統**，可以參考官方文件的相關設定。

總結

1. **單個 `vmagent` 可爬取數萬個目標，但受限於 CPU、記憶體、網路**。
2. **使用 `-promscrape.cluster.membersCount` 水平擴展 `vmagent`**，並為每個 `vmagent` 設定 `-promscrape.cluster.memberNum`。
3. **可使用 `-promscrape.cluster.replicationFactor` 讓多個 `vmagent` 爬取相同目標**，以提高可用性。
4. **透過 `/service-discovery` 監控 `vmagent` 狀態，並設定 `-promscrape.cluster.memberURLTemplate` 提供連結**。
5. **避免數據重複，需在遠端存儲啟用數據去重**。
6. **可使用 `-promscrape.cluster.memberLabel` 在指標中標記 `vmagent` 實例編號**。

這樣可以高效管理大量爬取目標，並確保數據一致性與高可用性。

翻成繁體中文並詳細說明:
Replication and data safety #
By default, VictoriaMetrics offloads replication to the underlying storage pointed by -storageDataPath such as Google compute persistent disk, which guarantees data durability. VictoriaMetrics supports application-level replication if replicated durable persistent disks cannot be used for some reason.

The replication can be enabled by passing -replicationFactor=N command-line flag to vminsert. This instructs vminsert to store N copies for every ingested sample on N distinct vmstorage nodes. This guarantees that all the stored data remains available for querying if up to N-1 vmstorage nodes are unavailable.

Passing -replicationFactor=N command-line flag to vmselect instructs it to not mark responses as partial if less than -replicationFactor vmstorage nodes are unavailable during the query. See cluster availability docs for details.

The cluster must contain at least 2*N-1 vmstorage nodes, where N is replication factor, in order to maintain the given replication factor for newly ingested data when N-1 of storage nodes are unavailable.

VictoriaMetrics stores timestamps with millisecond precision, so -dedup.minScrapeInterval=1ms command-line flag must be passed to vmselect nodes when the replication is enabled, so they could de-duplicate replicated samples obtained from distinct vmstorage nodes during querying. If duplicate data is pushed to VictoriaMetrics from identically configured vmagent instances or Prometheus instances, then the -dedup.minScrapeInterval must be set to scrape_interval from scrape configs according to deduplication docs.

Note that replication doesn’t save from disaster, so it is recommended performing regular backups. See these docs for details.

Note that the replication increases resource usage - CPU, RAM, disk space, network bandwidth - by up to -replicationFactor=N times, because vminsert stores N copies of incoming data to distinct vmstorage nodes and vmselect needs to de-duplicate the replicated data

obtained from vmstorage nodes during querying. So it is more cost-effective to offload the replication to underlying replicated durable storage pointed by -storageDataPath such as Google Compute Engine persistent disk, which is protected from data loss and data corruption. It also provides consistently high performance and may be resized without downtime. HDD-based persistent disks should be enough for the majority of use cases. It is recommended using durable replicated persistent volumes in Kubernetes.

資料複製（Replication）與數據安全（Data Safety）

1. VictoriaMetrics 預設的資料複製機制

在 VictoriaMetrics 中，**預設的資料複製（Replication）機制** 是依賴 **底層存儲（underlying storage）**，這些存儲系統可以透過 `-storageDataPath` 參數指定。例如：

- **Google Compute Persistent Disk** 提供了 **數據持久性保障**，確保數據不會因為硬體故障而丟失。

然而，在某些情況下，可能無法使用 **底層具備持久性保障的存儲系統**，此時 VictoriaMetrics **支援應用層級的資料複製（Application-Level Replication）**，以提高數據的可用性。

---

2. 啟用應用層級的資料複製

可以透過 `-replicationFactor=N` 參數來啟用 **應用層級的複製**。這項設定適用於 `vminsert` **節點**，當啟用時：

- **每個接收到的數據樣本（sample）都會被存儲 `N` 份**，並分別儲存於 `N` **個不同的 `vmstorage` 節點**。
- **即使最多 `N-1` 個 `vmstorage` 節點發生故障，數據仍然可用**，不會影響查詢。

示例：設定 3 份資料複製

```
/path/to/vminsert -replicationFactor=3
```

此設定表示：

- 每筆數據都會存儲 **3 份**（分別存放在 3 個不同的 `vmstorage` 節點）。
- **即使有 2 個 `vmstorage` 節點故障**，數據仍然可用。

---

3. 設定 `vmselect` 的複製行為

在查詢時，可以對 `vmselect` **節點** 設定 `-replicationFactor=N`，用於處理查詢時 `vmstorage` 的可用性問題。

- **若少於 `-replicationFactor` 個 `vmstorage` 節點無法訪問**，則 `vmselect` **不會將查詢結果標記為部分結果（partial result）**。
- 如果超過 `N-1` 個 `vmstorage` 節點發生故障，則仍可能會出現 **部分結果**。

示例：設定 `vmselect` 查詢時不標記部分結果

```
/path/to/vmselect -replicationFactor=3
```

這樣即使最多 2 個 `vmstorage` 節點故障，查詢結果仍不會被標記為 **partial result**。

---

4. 確保 `vmstorage` 節點數量足夠

為了確保新寫入的數據 **始終符合 `N` 份複製的要求**，當最多 **`N-1`** 個 **`vmstorage`** 節點故障時，叢集內的 **`vmstorage` 至少應包含 `2*N-1` 個節點**。

示例：設定 3 份複製所需的 `vmstorage` 節點數

如果 **`-replicationFactor=3`**，則 **`vmstorage`** 節點數應至少為：

```
2 * 3 - 1 = 5
```

因此，在 **最糟糕情況下（最多 `2` 個 `vmstorage` 故障）**，仍然能確保新數據可以存儲 **`3`** 份，滿足複製要求。

---

5. 啟用去重（De-duplication）

VictoriaMetrics **存儲的時間戳（timestamp）具有毫秒級精度**，當啟用 **複製（Replication）** 時，可能會導致多個 **`vmstorage`** 返回 **相同的樣本（Sample）**。為了避免 **重複數據影響查詢結果**，應在 **`vmselect`** 節點啟用 **去重（De-duplication）**，設定：

```
/path/to/vmselect -dedup.minScrapeInterval=1ms
```

這樣 **`vmselect`** 會在查詢時 **去除來自不同 `vmstorage` 的重複數據**，確保數據一致性。

當來自多個 `vmagent` 或 `Prometheus` 的數據重複時

如果數據來自 **多個 `vmagent` 或 `Prometheus`**，則應將 **`-dedup.minScrapeInterval`** 設定為 **`scrape_interval`**，以確保不同來源的數據正確去重。例如：

```
/path/to/vmselect -dedup.minScrapeInterval=10s
```

其中 `10s` 代表 **與 `scrape_interval` 相同的時間間隔**。

---

6. 複製無法防止數據災難，建議定期備份

雖然複製機制能夠提供 **數據冗餘（redundancy）**，但它**無法完全防止數據災難**，例如：

- **整個叢集故障**
- **錯誤的數據刪除**
- **人為操作失誤**

因此，**建議定期執行備份**，確保數據的完整性。可參考官方文檔進行數據備份與恢復。

---

7. 資源消耗與效能考量

啟用 `-replicationFactor=N` 會 **顯著增加資源消耗**：

- **CPU 負載**
- **記憶體（RAM）使用量**
- **磁碟空間**
- **網路頻寬**

這是因為：

1. **`vminsert` 需要將每條數據存儲 `N` 份到不同 `vmstorage` 節點**。
2. **`vmselect` 在查詢時需要去重處理，增加計算開銷**。

最佳做法

如果 **底層存儲系統本身已經具備持久性保障**（如 **Google Compute Persistent Disk**），則 **應優先利用底層存儲的複製機制，而非啟用 `-replicationFactor`**。這樣：

- 可以**降低 `vminsert` 和 `vmselect` 的負載**。
- 提供**穩定且高效的讀取/寫入效能**。
- **磁碟空間可動態調整，無需停機擴展**。

HDD 與 Kubernetes 的建議

- **對大多數應用場景而言，HDD 型的持久性磁碟（Persistent Disk）已經足夠**。
- **在 Kubernetes 環境中，建議使用「具備持久性複製」的存儲卷（Persistent Volume）**。

---

8. 總結

項目	詳細說明
預設機制	使用 `-storageDataPath` 指定底層存儲，如 Google Compute Persistent Disk
應用層複製	可透過 `-replicationFactor=N` 設定 `vminsert` 進行數據複製
查詢行為	`vmselect` 透過 `-replicationFactor=N` 來確保 `N-1` 個 `vmstorage` 節點故障時仍能查詢
最低 `vmstorage` 節點數	需至少 `2*N-1` 個 `vmstorage`，以確保 `N-1` 節點故障時仍能寫入數據
去重機制	`vmselect` 需設定 `-dedup.minScrapeInterval=1ms` 以去除重複數據
備份重要性	複製無法防止數據災難，建議定期備份
資源消耗	`-replicationFactor` 會增加 **CPU、RAM、磁碟空間、網路頻寬**
優先考量	優先使用**底層持久性存儲的複製機制**（如 Google Persistent Disk），避免不必要的應用層複製

透過這些設定，可以在 **數據安全、效能與資源使用之間取得最佳平衡**。