



# K8s與Cluster部署建議



**Cluster version.pdf**  
PDF

參考此pdf，詳細說明cluster需不需要用k8s的建議



以下是根據PDF中提供的內容，關於是否在Cluster版本的VictoriaMetrics上使用Kubernetes（K8s）的建議：

## 何時適合使用K8s來管理Cluster

Kubernetes 是一種用於容器編排的工具，其主要優勢包括高可用性、自動擴展和資源管理。以下是幾種適合在Cluster版本上使用K8s的情境：

### 1. 動態擴展需求

- 當您的系統需要處理大量且變動頻繁的數據流量時，K8s可以自動擴展Pod（如`vminsert`、`vmselect`、`vmstorage`）來應對負載高峰。

### 2. 高可用性

- 根據文檔描述，VictoriaMetrics Cluster架構的每個組件應至少部署兩個節點以實現高可用性。K8s可以監控這些節點並在節點故障時自動重新啟動相關Pod。

### 3. 複雜集群管理

- 當集群包含多個節點且每個節點需要不同的配置時，K8s的Helm chart和Operators可以簡化部署和管理過程。

### 4. 跨多地區或多AZ部署

- 若需在多個可用區（AZ）或地區部署集群，K8s能幫助分配資源並實現負載均衡。

### 5. 持續集成/交付（CI/CD）

- 使用K8s可以方便地集成自動化部署流程，支持滾動升級或無停機更新。

## 何時不適合使用K8s

### 1. 單機部署





- 如果數據量不足（例如每秒小於一百萬數據點），官方建議使用單節點版本，避免配置和管理Cluster版本的複雜性。

## 2. 固定負載

- 當工作負載較穩定且無需頻繁調整資源時，直接使用Docker Compose或手動配置可能更簡單且成本較低。

## 3. 基礎設施有限

- 如果資源有限或組織無法負擔K8s的運維開銷，則可考慮輕量級的部署方案。

## 小結

是否使用K8s來管理Cluster版本應視以下因素而定：

- 集群規模和數據負載。
- 系統是否需要動態擴展。
- 運維團隊對K8s的熟悉程度及資源支持。

若需具體的實施建議，可依需求進一步討論設計方案！



參考"Cluster version.pdf"，不使用k8s能實現動態擴展vminsert, vmselect, vmstorage嗎



根據 PDF 文件的內容，不使用 Kubernetes (K8s) 仍然可以實現動態擴展 VictoriaMetrics Cluster 中的 `vminsert`、`vmselect` 和 `vmstorage` 節點，具體方法如下：

## 動態擴展的實現方法

### 1. 垂直擴展 (Vertical Scaling)

- 增加現有節點的 CPU、RAM 和磁碟 I/O 資源來處理更大的負載。
- 適合數據量增長不大的情況，特別是需要處理高效查詢的 `vmselect` 或高並發數據寫入的 `vminsert`。

### 2. 水平擴展 (Horizontal Scaling)

- 增加新的節點以分擔工作負載：





- `\vminsert`` 節點：
  - 新增 `\vminsert`` 節點，並更新負載均衡器配置（例如使用 Nginx 或其他 HTTP 負載均衡工具），以將數據寫入流量均勻分配至所有 `\vminsert`` 節點。
- `\vmselect`` 節點：
  - 新增 `\vmselect`` 節點，並在負載均衡器中配置其訪問邏輯，確保查詢流量可以均分至多個節點。
- `\vmstorage`` 節點：
  - 啟動新的 `\vmstorage`` 節點，並更新所有 `\vminsert`` 和 `\vmselect`` 節點的配置，使其包含新的 `\vmstorage`` 節點。
  - `\vminsert`` 節點將新數據分配到新的 `\vmstorage`` 節點，實現動態負載均衡。

### 3. 文件或 DNS SRV 基於配置的節點發現

- 文件配置：
  - 在文件中列出所有的 `\vmstorage`` 節點地址，並通過 `-storageNode=file:/path/to/file-with-vmstorage-list`` 傳遞給 `\vminsert`` 和 `\vmselect``。
  - 文件更新後，`\vminsert`` 和 `\vmselect`` 節點會自動感知到新的 `\vmstorage`` 節點。
- DNS SRV：
  - 使用 `-storageNode=svr+<svr_record>`` 來通過 DNS SRV 解析獲取可用的節點地址。
  - 當新的節點添加到 DNS 記錄時，集群會自動更新可用節點。

### 4. 負載均衡器

- 配置 HTTP 負載均衡器（如 Nginx、vmauth 或其他工具）：
  - 寫入流量通過負載均衡器分配至不同的 `\vminsert`` 節點。
  - 查詢流量通過負載均衡器分配至不同的 `\vmselect`` 節點。
- 確保負載均衡器動態更新節點地址。

---

## 需要注意的事項

### 1. 配置更新

- 在添加或移除節點時，需要更新負載均衡器的配置以及現有節點的 `-storageNode`` 參數，確保流量正確分配。



## 2. 節點高可用

- 每種節點（`vminsert`、`vmselect` 和 `vmstorage`）應至少部署兩個實例，以避免單點故障。

## 3. 監控和調整

- 建議使用 Prometheus 或其他監控工具來監控資源使用情況和集群健康狀態。
- 根據負載情況調整節點數量或資源分配。

## 小結

雖然不使用 Kubernetes 會增加一些管理和運維的複雜度，但通過負載均衡器和動態配置文件機制，可以實現 `vminsert`、`vmselect` 和 `vmstorage` 節點的動態擴展，滿足 VictoriaMetrics Cluster 的彈性擴展需求。



參考"Cluster version.pdf"，詳細說明使用k8s建置vmstorage、vminsert、vmselect



以下是根據 "Cluster version.pdf" 的內容，詳細說明如何使用 Kubernetes (K8s) 部署 VictoriaMetrics Cluster 的 `vmstorage`、`vminsert` 和 `vmselect`：

## 1. 基礎環境需求

### 1.1 節點需求

VictoriaMetrics Cluster 包含以下核心服務：

- `vminsert`：負責接收數據並將其分配到 `vmstorage` 節點。
- `vmselect`：負責從 `vmstorage` 節點中查詢數據。
- `vmstorage`：負責存儲數據。

### 1.2 前置要求

- Kubernetes 集群版本：建議使用最新的穩定版本。
- 容器運行時：支持 Docker 或 Containerd。
- Helm：建議使用 Helm Chart 來部署。

### 1.3 配置存儲

- 持久存儲：`vmstorage` 節點需要持久存儲，建議使用 Persistent Volume (PV)。



- **內存和 CPU**：根據數據負載需求分配資源（具體配置見下方資源建議）。

## 2. 部署流程

### 2.1 創建命名空間

創建一個專用的命名空間來管理 VictoriaMetrics 資源。

```
kubectl create namespace victoriametrics
```

### 2.2 配置 Persistent Volume (PV) 和 Persistent Volume Claim (PVC)

「vmstorage」需要用到持久化存儲，下面是 PV 和 PVC 的範例：

```
# vmstorage-pv.yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: vmstorage-pv
spec:
  capacity:
    storage: 100Gi
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  hostPath:
    path: "/data/vmstorage"

---
# vmstorage-pvc.yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: vmstorage-pvc
  namespace: victoriametrics
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 100Gi
```

應用 PV 和 PVC：

```
kubectl apply -f vmstorage-pv.yaml
```



```
kubectl apply -f vmstorage-pvc.yaml
```

## 2.3 使用 Helm Chart 部署

VictoriaMetrics 提供官方的 Helm Chart，用於快速部署 Cluster。

### 1. 添加 Helm Repository

```
helm repo add victoriametrics https://victoriametrics.github.io/helm-charts/
helm repo update
```

### 2. 安裝 VictoriaMetrics Cluster 使用以下命令部署 `vmselect`、`vminsert` 和 `vmstorage`：

```
helm install vm-cluster victoriametrics/victoria-metrics-cluster \
  --namespace victoriametrics \
  --set vmstorage.persistence.enabled=true \
  --set vmstorage.persistence.storageClass="standard" \
  --set vmstorage.persistence.size="100Gi" \
  --set replicaCount=2
```

### 3. 自定義配置 如果需要自定義更多參數，可以使用 `values.yaml` 文件，配置內容包括：

- 資源限制（CPU、RAM）
- 節點副本數量
- 負載均衡器設置
- mTLS 或安全配置

## 2.4 部署檔案範例

如果不使用 Helm，可以直接編寫 YAML 文件來部署：

### 1. `vminsert` 節點

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: vminsert
  namespace: victoriametrics
spec:
  replicas: 2
  selector:
    matchLabels:
      app: vminsert
```



```
template:
  metadata:
    labels:
      app: vminsert
  spec:
    containers:
      - name: vminsert
        image: victoriametrics/vminsert:v1.95.0
        args:
          - -storageNode=vmstorage:8400
        ports:
          - containerPort: 8480
```

## 2. `vmselect` 節點

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: vmselect
  namespace: victoriametrics
spec:
  replicas: 2
  selector:
    matchLabels:
      app: vmselect
  template:
    metadata:
      labels:
        app: vmselect
    spec:
      containers:
        - name: vmselect
          image: victoriametrics/vmselect:v1.95.0
          args:
            - -storageNode=vmstorage:8400
          ports:
            - containerPort: 8481
```

## 3. `vmstorage` 節點

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: vmstorage
  namespace: victoriametrics
spec:
  replicas: 2
  selector:
    matchLabels:
      app: vmstorage
  serviceName: "vmstorage"
  template:
```



```
metadata:
  labels:
    app: vmstorage
spec:
  containers:
  - name: vmstorage
    image: victoriametrics/vmstorage:v1.95.0
    args:
      - -storageDataPath=/storage
    ports:
      - containerPort: 8482
    volumeMounts:
      - name: storage
        mountPath: /storage
  volumeClaimTemplates:
  - metadata:
      name: storage
    spec:
      accessModes: [ "ReadWriteOnce" ]
      resources:
        requests:
          storage: 100Gi
```

#### 4. 服務定義

- 使用 ClusterIP 暴露 `vmstorage`，並使用 LoadBalancer 暴露 `vminsert` 和 `vmselect`。

```
apiVersion: v1
kind: Service
metadata:
  name: vmstorage
  namespace: victoriametrics
spec:
  selector:
    app: vmstorage
  ports:
  - protocol: TCP
    port: 8400
    targetPort: 8482
  clusterIP: None
```

### 3. 監控和擴展

#### 3.1 監控

- 使用 Prometheus 或 Grafana 監控 VictoriaMetrics Cluster。
- 每個節點暴露 `/metrics` 接口，可用於收集指標。





### 3.2 動態擴展

- 修改 `replicas` 值動態擴展節點數。
- 使用 HPA (Horizontal Pod Autoscaler) 基於 CPU 或內存負載自動調整 Pod 數量。

## 小結

使用 Kubernetes 部署 VictoriaMetrics Cluster 提供靈活的擴展性和高可用性。推薦通過 Helm Chart 簡化部署過程，並結合 PVC 確保持久存儲。若需精細控制，可直接使用 YAML 文件設置部署。



參考"Cluster version.pdf"，詳細說明不使用k8s能實現動態水平擴展、動態垂直擴展嗎



根據 "**Cluster version.pdf**"，不使用 Kubernetes (K8s)，仍然可以通過手動或腳本化的方式實現 VictoriaMetrics Cluster 的動態水平擴展 (Horizontal Scaling) 和動態垂直擴展 (Vertical Scaling)。以下是詳細的步驟和實現方式：

## 1. 動態水平擴展 (Horizontal Scaling)

水平擴展是通過增加新的節點來處理更高的負載。以下是針對 `vminsert`、`vmselect` 和 `vmstorage` 的具體方法。

### 1.1 新增 `vmstorage` 節點

`vmstorage` 是存儲數據的核心組件，水平擴展可提高存儲容量和查詢性能。

步驟：

#### 1. 啟動新節點

- 啟動新的 `vmstorage` 節點，並確保使用相同的 `--retentionPeriod` 和 `--storageDataPath` 配置。
- 示例命令：

```
./vmstorage --retentionPeriod=30d --storageDataPath=/data/vmstorage
```

#### 2. 更新 `vminsert` 節點配置





- 在所有現有的 `vminsert` 節點中，新增該 `vmstorage` 節點的地址到 `storageNode` 配置中。
- 示例：

```
./vminsert -storageNode=vmstorage1:8400 -storageNode=vmstorage2:8400
```

### 3. 更新 `vmselect` 節點配置

- 在所有 `vmselect` 節點中，新增該 `vmstorage` 節點的地址到 `storageNode` 配置中。
- 示例：

```
./vmselect -storageNode=vmstorage1:8400 -storageNode=vmstorage2:8400
```

### 4. 負載均衡器調整

- 更新負載均衡器（如 Nginx）配置，使其可以均衡分配流量至所有節點。

---

## 1.2 新增 `vminsert` 節點

`vminsert` 負責接收數據寫入，新增節點可以提高數據寫入吞吐量。

步驟：

#### 1. 啟動新節點

- 啟動新的 `vminsert` 節點，並配置現有的所有 `vmstorage` 節點地址。
- 示例命令：

```
./vminsert -storageNode=vmstorage1:8400 -storageNode=vmstorage2:8400
```

#### 2. 更新負載均衡器

- 在負載均衡器中新增該節點，使其分配數據寫入流量到新的 `vminsert` 節點。

---

## 1.3 新增 `vmselect` 節點

`vmselect` 負責查詢數據，新增節點可以提高查詢並發能力。

步驟：

#### 1. 啟動新節點



- 啟動新的 `vmselect` 節點，並配置現有的所有 `vmstorage` 節點地址。
- 示例命令：

```
./vmselect -storageNode=vmstorage1:8400 -storageNode=vmstorage2:8400
```

## 2. 更新負載均衡器

- 在負載均衡器中新增該節點，使其分配查詢流量到新的 `vmselect` 節點。

## 2. 動態垂直擴展（Vertical Scaling）

垂直擴展是通過增加現有節點的硬體資源（如 CPU、RAM、磁碟 I/O 等）來提升性能。

### 2.1 增加節點的硬體資源

- 在雲環境中（如 AWS、GCP 或 Azure），可通過調整實例類型增加 CPU 和 RAM 資源。
- 在本地環境中，可以升級硬體（如更換更快的磁碟或增加內存）。

### 2.2 針對 `vminsert` 的優化

- 增加 CPU 和 RAM：
  - `vminsert` 處理數據壓縮和傳輸，增加資源能提高寫入性能。
- 配置優化：
  - 增加 `maxConcurrentInserts` 參數來允許更多並發寫入。
  - 示例：

```
./vminsert -storageNode=vmstorage1:8400 -maxConcurrentInserts=200
```

### 2.3 針對 `vmselect` 的優化

- 增加 CPU 和 RAM：
  - `vmselect` 處理查詢，特別是複雜的查詢，增加 CPU 核心數可以顯著提高性能。
- 配置優化：
  - 增加 `search.maxConcurrentRequests` 來允許更多併發查詢。
  - 示例：



```
./vmselect -storageNode=vmstorage1:8400 -search.maxConcurrentRequests=100
```

## 2.4 針對 `vmstorage` 的優化

- 增加磁碟性能：
  - 使用 SSD 替代 HDD 或選擇更高性能的磁碟陣列。
- 配置優化：
  - 增加 `-storage.maxDailySeries` 來允許更多數據序列。
  - 示例：

```
./vmstorage -storageDataPath=/data/vmstorage -storage.maxDailySeries=1000000
```

## 3. 資源監控與動態調整

為了確保水平或垂直擴展後的系統穩定性，需要持續監控資源使用情況：

### 1. 監控工具

- 使用 Prometheus 收集 VictoriaMetrics 的指標數據。
- 每個節點的 `/metrics` 路徑提供詳細的性能指標。

### 2. 調整策略

- 根據資源使用率（CPU、RAM、磁碟）設置閾值，當超過一定限制時觸發擴展操作。
- 配置腳本來自動執行新增節點或調整配置參數。

## 4. 小結

不使用 Kubernetes 的情況下，可以通過以下方式實現動態擴展：

- **水平擴展**：手動新增節點，並更新負載均衡器和現有節點的配置。
- **垂直擴展**：升級節點硬體資源，並調整相關配置參數。

雖然此方法不如 K8s 的自動化程度高，但結合腳本化管理和監控，仍可實現高效的擴展和穩定運行。





參考"Cluster version.pdf"，詳細說明不使用k8s實現動態水平擴展



根據 "**Cluster version.pdf**"，以下是詳細說明不使用 Kubernetes (K8s) 的情況下，如何實現 VictoriaMetrics Cluster 的動態水平擴展（Horizontal Scaling）。這包括如何為 ``vminsert``、``vmselect`` 和 ``vmstorage`` 添加新節點並確保其穩定運行。

## 1. 動態水平擴展的基本原則

水平擴展的目的是增加新的節點來分擔數據負載或提高查詢性能。VictoriaMetrics Cluster 的架構設計支持在運行時通過更新配置實現水平擴展。

## 2. ``vmstorage`` 節點的水平擴展

``vmstorage`` 是存儲數據的核心組件，新增節點可以提高存儲容量和查詢性能。

步驟：

### 1. 啟動新節點

- 配置新節點並啟動一個新的 ``vmstorage`` 實例，確保使用相同的數據保留策略（``-retentionPeriod``）。
- 示例：

```
./vmstorage -retentionPeriod=30d -storageDataPath=/data/vmstorage
```

### 2. 更新所有 ``vminsert`` 節點

- 在所有現有的 ``vminsert`` 節點中，新增新 ``vmstorage`` 節點的地址到 ``-storageNode`` 配置中。
- 示例：

```
./vminsert -storageNode=vmstorage1:8400 -storageNode=vmstorage2:8400
```

### 3. 更新所有 ``vmselect`` 節點

- 在所有現有的 ``vmselect`` 節點中，新增新 ``vmstorage`` 節點的地址到 ``-storageNode`` 配置中。
- 示例：



```
./vmselect -storageNode=vmstorage1:8400 -storageNode=vmstorage2:8400
```

#### 4. 數據分配調整

- 新增節點後，`vminsert` 會自動將數據分配到新的 `vmstorage` 節點。可以使用以下方法進一步優化：
  - 配置 `vminsert` 僅將新數據寫入新的 `vmstorage` 節點，直到磁碟使用率平衡。
  - 更新配置後，重新允許所有節點均分負載。

#### 5. 監控與測試

- 確保新增的 `vmstorage` 節點正常運行並均勻分擔負載。
- 檢查 `/metrics` 界面，確認數據寫入和查詢分佈是否均勻。

### 3. `vminsert` 節點的水平擴展

新增 `vminsert` 節點可以提高數據寫入吞吐量。

#### 步驟：

##### 1. 啟動新節點

- 啟動一個新的 `vminsert` 實例，並配置現有所有 `vmstorage` 節點地址。
- 示例：

```
./vminsert -storageNode=vmstorage1:8400 -storageNode=vmstorage2:8400
```

##### 2. 更新負載均衡器

- 在負載均衡器（如 Nginx、HAProxy）中新增該節點，使數據寫入流量均衡分配至所有 `vminsert` 節點。
- Nginx 配置示例：

```
upstream vminsert_nodes {
    server vminsert1:8480;
    server vminsert2:8480;
}

server {
    location /insert/ {
        proxy_pass http://vminsert_nodes;
    }
}
```



```
}  
}
```

### 3. 測試與監控

- 測試數據寫入是否均勻分佈到所有 `vminsert` 節點。
- 監控新節點的資源使用情況和性能。

## 4. `vmselect` 節點的水平擴展

新增 `vmselect` 節點可以提高查詢吞吐量和性能。

### 步驟：

#### 1. 啟動新節點

- 啟動一個新的 `vmselect` 實例，並配置所有 `vmstorage` 節點地址。
- 示例：

```
./vmselect -storageNode=vmstorage1:8400 -storageNode=vmstorage2:8400
```

#### 2. 更新負載均衡器

- 在負載均衡器中新增該節點，使查詢流量均衡分配至所有 `vmselect` 節點。
- Nginx 配置示例：

```
upstream vmselect_nodes {  
    server vmselect1:8481;  
    server vmselect2:8481;  
}  
  
server {  
    location /select/ {  
        proxy_pass http://vmselect_nodes;  
    }  
}
```

### 3. 測試與監控

- 測試查詢是否能被均勻分配到所有 `vmselect` 節點。
- 監控新節點的查詢性能和資源使用。



## 5. 自動化和工具支持

雖然不使用 K8s，但可以使用腳本或工具實現擴展自動化：

### 1. 腳本化操作

- 編寫 Shell 腳本來啟動新節點並更新配置。
- 例如：

```
# 新增 vmstorage 節點
./vmstorage -retentionPeriod=30d -storageDataPath=/data/vmstorage &

# 更新所有 vminsert 配置
sed -i 's/-storageNode=.*-/storageNode=vmstorage1:8400 -storageNode=vmstorage2:8400/' vminsert_config

# 重啟 vminsert
pkill vminsert
./vminsert -config=vminsert_config &
```

### 2. 使用配置服務

- 如果有多節點分佈式架構，可以使用配置管理工具（如 Consul 或 etcd）動態管理和分發節點信息。

### 3. 監控與警報

- 使用 Prometheus 收集每個節點的性能數據，並設置警報來提示何時需要擴展。

## 6. 小結

不使用 K8s 的情況下，可以通過手動或腳本化的方式實現水平擴展：

- **新增節點**：啟動新的 `vminsert`、`vmselect` 或 `vmstorage` 節點。
- **更新配置**：更新現有節點的 `storageNode` 配置，確保新節點可用。
- **負載均衡**：通過 Nginx 或其他工具，均衡分配數據寫入和查詢流量。

雖然此方法需要一定的手動操作，但結合自動化腳本和監控工具，可以有效實現動態水平擴展。



參考"Cluster version.pdf"，詳細說明不使用k8s實現動態垂直擴展







根據 "**Cluster version.pdf**", 不使用 Kubernetes (K8s) 的情況下, 可以通過手動或自動化的方式對 VictoriaMetrics Cluster 的節點進行動態垂直擴展 (Vertical Scaling)。垂直擴展的目的是增加現有節點的資源 (如 CPU、RAM 或磁碟 I/O), 以提升性能和處理能力。

---

## 1. 動態垂直擴展的核心原則

### 1.1 增加資源的情境

- **CPU 負載高**：查詢或數據壓縮操作消耗過多的計算資源。
- **內存不足**：節點因查詢數據、緩存或內存不足而崩潰。
- **磁碟 I/O 壓力大**：數據寫入或讀取速度受磁碟性能限制。

### 1.2 垂直擴展的主要方法

- 升級硬體 (例如, 增加 CPU 核心數、內存大小或更換 SSD)。
  - 動態調整應用參數 (例如, 增加併發查詢或寫入限制)。
  - 減少資源競爭 (例如, 優化數據分佈或減少不必要的操作)。
- 

## 2. 垂直擴展具體步驟

### 2.1 針對 `vmstorage` 的垂直擴展

`vmstorage` 是負責數據存儲的組件, 主要受磁碟性能和內存影響。

方法：

#### 1. 升級磁碟性能

- 替換更高性能的磁碟 (如從 HDD 升級到 SSD)。
- 若使用雲存儲, 選擇 IOPS 較高的存儲類型 (例如 AWS 的 io2 類型 EBS 或 GCP 的高性能磁碟)。

#### 2. 增加內存

- `vmstorage` 需要內存來維護活躍時間序列 (active time series), 更多內存可以支持更高的數據負載。
- 例如, 在雲環境中可以動態升級實例類型。

#### 3. 優化配置

- 使用以下參數限制資源競爭並提高效率：
  - `-storage.maxDailySeries``：設置每日最大序列數。



- ``-search.maxMemoryPerQuery``：限制單次查詢的內存使用量。
- 示例：

```
./vmstorage -storageDataPath=/data/vmstorage \  
-storage.maxDailySeries=1000000 \  
-search.maxMemoryPerQuery=2GB
```

#### 4. 監控並調整

- 通過 ``/metrics`` 查看 ``vm_storage_*` 指標，評估磁碟和內存使用情況。
- 如果磁碟空間不足，可以使用分片機制將數據分佈到多個磁碟或節點。

## 2.2 針對 ``vminsert`` 的垂直擴展

``vminsert`` 負責接收數據並壓縮後寫入 ``vmstorage``，其主要受 CPU 和內存影響。

方法：

#### 1. 增加 CPU

- 增加 CPU 核心數，提高數據壓縮和寫入性能。
- 適合數據流量高、壓縮負載重的情境。

#### 2. 優化配置

- 增加最大併發寫入數量：
  - 使用 ``-maxConcurrentInserts`` 調整併發寫入限制。
  - 示例：

```
./vminsert -storageNode=vmstorage1:8400 \  
-maxConcurrentInserts=200
```

- 禁用壓縮（在 CPU 資源緊張時）：
  - 使用 ``-rpc.disableCompression`` 禁用數據壓縮以節省 CPU。
  - 示例：

```
./vminsert -storageNode=vmstorage1:8400 \  
-rpc.disableCompression
```

#### 3. 監控並調整

- 查看 ``/metrics`` 中的 ``vminsert_rpc_*` 指標，檢查數據寫入的延遲和吞吐量。



## 2.3 針對 `vmselect` 的垂直擴展

`vmselect` 負責數據查詢，主要受 CPU 和內存影響。

方法：

### 1. 增加 CPU

- 查詢性能與 CPU 線性相關，增加 CPU 核心數可提升查詢速度。
- 尤其在處理複雜查詢或大範圍數據時效果明顯。

### 2. 優化配置

- 增加查詢併發數：
  - 使用 `-search.maxConcurrentRequests` 設置最大併發查詢數。
  - 示例：

```
./vmselect -storageNode=vmstorage1:8400 \  
-search.maxConcurrentRequests=100
```

- 限制每次查詢的資源使用：
  - 使用 `-search.maxMemoryPerQuery` 設置單次查詢的內存限制。
  - 示例：

```
./vmselect -search.maxMemoryPerQuery=4GB
```

### 3. 調整緩存大小

- 使用 `-cacheDataPath` 設置磁碟緩存路徑，減少內存壓力。
- 示例：

```
./vmselect -cacheDataPath=/tmp/vmcache
```

### 4. 監控與調整

- 查看 `/metrics` 中的 `vm\_search\_\*` 指標，分析查詢的延遲和資源使用情況。
- 對於頻繁執行的重複查詢，可以設置更大的緩存大小以提升效率。

## 3. 自動化垂直擴展

### 3.1 在雲環境中實現



## 1. 動態調整實例類型

- 使用 AWS EC2、GCP Compute Engine 等雲服務的自動調整功能，在高負載時升級實例類型。

## 2. 監控資源使用

- 通過監控工具（如 CloudWatch、Stackdriver）檢測 CPU 或內存使用情況，自動觸發升級操作。

---

## 3.2 使用腳本自動化

### 1. 資源升級腳本

- 編寫 Shell 腳本檢查資源使用情況，根據條件動態調整節點參數。
- 示例：

```
# 檢查 CPU 使用率
CPU_USAGE=$(grep 'cpu' /proc/stat | awk '{print $2}')
if [ "$CPU_USAGE" -gt 80 ]; then
    echo "CPU 使用率高，增加並發查詢數"
    ./vmselect -search.maxConcurrentRequests=200
fi
```

### 2. 配置管理工具

- 使用 Ansible 或 SaltStack 自動化節點配置和調整參數。

---

## 4. 垂直擴展的監控與優化

### 1. 監控工具

- 使用 Prometheus 監控各節點的 `/metrics`，關注 CPU、內存和磁碟使用指標。
- 重點關注：
  - `vm_storage_*`（磁碟性能）
  - `vminsert_rpc_*`（數據寫入）
  - `vm_search_*`（查詢性能）

### 2. 調整策略

- 根據監控數據動態調整節點資源。
- 設置警報來提前檢測資源瓶頸。



---

## 5. 小結

不使用 K8s 的情況下，可以通過手動或腳本化的方式進行動態垂直擴展：

- `vmstorage``：增加磁碟性能和內存，優化存儲吞吐量。
- `vminsert``：增加 CPU 並調整併發寫入參數。
- `vmselect``：增加 CPU 和內存，優化查詢性能。

通過監控和自動化工具，可以實現資源的高效利用和動態調整，確保集群穩定運行。