



Prometheus與VictoriaMetrics簡報解析



正式-深入探討 Remote Write 與 F...
PDF

請給我這簡報每一頁的說明



以下是簡報每一頁的說明：

1. Prometheus & VictoriaMetrics intro

說明簡報的主題為 Prometheus 與 VictoriaMetrics 的介紹及深入探討。

2. Introduction

簡述簡報的內容，包括 Prometheus Remote Write、Federation 的功能比較，以及 VictoriaMetrics 的特點和應用場景。

3. 空白或過渡頁。

4. 簡介

簡單介紹 Prometheus 作為開源監控系統，並提到 Remote Write 和 Federation 的主要特點。

5. Prometheus Remote Write

說明 Remote Write 的功能與應用，包括長期存儲、多集群數據匯聚及災難恢復場景。

6. Prometheus Remote Write

詳細描述 Remote Write 的優勢與缺點，強調其長期存儲和集中式分析的能力，但提到即時性和依賴性限制。

7. Prometheus Federation

解釋 Federation 的概念與應用，適用於層次化監控架構，例如數據抓取和整合。





8. Prometheus Federation

說明 Federation 的優勢（內建支持、高即時性）與缺點（適用於短期數據、可能影響負載）。

9. Remote Write 與 Federation 的比較表

將 Remote Write 和 Federation 進行比較，從用途、數據傳輸方向、即時性等方面做區分。

10. 結論

提出技術選擇建議，根據系統需求選擇適合的技術，並強調性能與成本的平衡。

11. 空白或過渡頁。

12. VictoriaMetrics 简介与应用

提到數據存儲的必要性以及 VictoriaMetrics 的應用場景，特別是高效數據處理能力。

13. 介紹

簡述 VictoriaMetrics 的性能優勢及其在微服務架構中的應用場景。

14. 為什麼選擇 VictoriaMetrics

突出其存儲效率、查詢速度和多租戶支持等優勢。

15. VictoriaMetrics 的功能亮點

說明其 Prometheus API 的完全兼容性、高壓縮比和靈活部署模式。

16. 部署模式-單節點部署

適合小規模環境的部署模式，適用於低並發需求的場景。

17. 部署模式-集群模式架構

描述 VictoriaMetrics 的集群架構，包括 vmstorage、vminsert 和 vmselect 的分工與特點。

18. 部署步骤和配置

提供 Helm 部署與手動部署的選擇建議，根據需求選擇單節點或集群模式。

19. 優勢與限制

高基數支持和低資源需求為主要優勢，但基礎設施對存儲節點數量有依賴。



20. 使用場景

適用於大規模監控數據存儲、多租戶架構和 SLA 報告場景。

21. 配置與運維建議

建議合理配置數據保留策略、監控集群性能，以及定期執行備份。

22. 與其他解決方案的比較

強調 VictoriaMetrics 在存儲效率、查詢性能和運維簡化方面的優勢。

23. 混合使用案例

與 Prometheus 集成的場景與效益，例如提升系統性能和降低運維成本。

24. 總結與建議

總結 VictoriaMetrics 的高效與可擴展性，推薦適用於高基數和大規模數據的場景。



ChatGPT-Prometheus Remote ...
PDF

參考此pdf檔，詳細說明簡報的第4頁



簡報第4頁的詳細說明如下：

第4頁 - 簡介

此頁主要介紹 **Prometheus** 的基本概念及其功能模組 **Remote Write** 和 **Federation**。
具體內容包括：

1. Prometheus 系統簡介

- Prometheus 是一款開源的監控系統，專注於高效的時間序列數據存儲與查詢功能。
- 提供靈活的指標抓取與數據管理方式，適用於多種監控場景。

2. Remote Write 和 Federation 的特點





- **Remote Write:**
 - 一種數據推送機制，允許將 Prometheus 收集到的數據傳輸到外部存儲系統。
 - 適用於數據的長期存儲與多集群數據的集中化分析。
- **Federation:**
 - 一種數據共享機制，上層 Prometheus 實例可以從下層實例抓取數據。
 - 適用於層次化的監控架構，特別是需要共享關鍵指標的情境。

3. 應用場景與選擇依據

- **Remote Write** 更適合需要將數據長期保存於外部系統的情境。
- **Federation** 則更適用於需要整合多個實例並進行層次化分析的場景。

總結

第4頁設定了基礎框架，幫助聽眾了解 Prometheus 監控系統的核心概念，並概述了 Remote Write 和 Federation 的定位，為後續更深入的技術對比鋪墊背景。



參考ChatGPT-Prometheus Remote Write vs Federation.pdf，詳細說明簡報的第5頁



簡報第5頁的詳細說明如下，內容根據 **ChatGPT-Prometheus Remote Write vs Federation.pdf**：

第5頁 - Prometheus Remote Write

功能簡介



Prometheus Remote Write 是一種數據推送機制，主要用於將 Prometheus 收集的時間序列數據發送到外部存儲系統。它適用於以下場景：

1. 長期存儲

- Prometheus 本地存儲適合短期存儲，但 Remote Write 可以將數據持久化到支持 Prometheus 協議的外部系統（如 Thanos、VictoriaMetrics）。
- 提供長時間的歷史數據保存能力，方便未來查詢與分析。

2. 多集群數據匯聚

- 可將來自多個 Prometheus 實例的數據推送到集中式存儲，實現統一分析和管理。

3. 災難恢復

- 將數據同步到異地存儲作為備份，確保數據安全性。

運作原理

- Prometheus 透過 HTTP 協議將樣本數據以指定格式推送到遠程存儲端點。
- 配置範例如下：

```
remote_write:
- url: "http://remote-storage-endpoint/api/v1/write"
  write_relabel_configs:
    - source_labels: [__name__]
      regex: "node_.*"
      action: keep
```

- ``url``：指定遠程存儲的 API 接口。
- ``write_relabel_configs``：用於篩選需要推送的數據。

優點



1. 長期存儲

- 克服了 Prometheus 本地存儲的限制，將數據保存至外部系統，適合需要長期存儲和分析的場景。

2. 集中式數據管理

- 支援將多個集群的數據匯聚到單一存儲中，便於集中分析與管理。

3. 與外部分析工具集成

- 與工具（如 Grafana）結合使用，提供更強大的數據視覺化和分析功能。
-

缺點

1. 即時性不足

- 傳輸延遲可能導致數據無法即時更新，這在高頻數據場景中可能成為瓶頸。

2. 依賴外部存儲系統

- 需要安裝和配置額外的存儲系統，增加運維成本與複雜度。

3. 無法直接查詢

- Prometheus 本地無法查詢遠端存儲數據，查詢需要透過存儲後端支持的接口。
-

應用場景

1. Grafana + Prometheus + Thanos

- 為實現長期存儲與統一查詢的架構。

2. 多集群監控

- 將多個地區或環境的監控數據統一存儲，實現全域管理與數據分析。

3. 災難備份與恢復

- 在分散式存儲中增加異地數據備份層，提升數據可用性。



此頁內容重點在於闡明 Prometheus Remote Write 的技術原理、功能特性以及適用場景，幫助讀者理解其在長期存儲與集中式管理中的核心角色。



參考ChatGPT-Prometheus Remote Write vs Federation.pdf，詳細說明簡報的第6頁



簡報第6頁的詳細說明如下，根據 **ChatGPT-Prometheus Remote Write vs Federation.pdf** 提供的內容：

第6頁 - Prometheus Remote Write 詳細說明

核心功能

Prometheus Remote Write 是將數據推送至外部存儲系統的機制。以下為功能亮點的具體說明：

1. 長期存儲

- 克服 Prometheus 本地存儲僅能保存短期數據的限制，支持數據的長期存儲與查詢。
- 配合系統（如 VictoriaMetrics、Thanos、Cortex 等），建立穩定的歷史數據檔案。

2. 多集群數據匯聚

- 適用於多個 Prometheus 實例的場景，將數據統一推送至集中式存儲，便於管理與分析。

3. 災難恢復



- 將監控數據同步到異地存儲，提升數據的冗餘性與可用性。

技術原理

Prometheus Remote Write 透過 HTTP 將樣本數據推送到配置的遠程存儲後端，支持以下流程：

- 利用 **write_relabel_configs** 過濾特定指標或數據範圍。
- 推送的數據包含標籤與數值，根據存儲後端的要求進行格式化。

配置示例：

```
remote_write:
- url: "http://remote-storage.example.com/api/v1/write"
  headers:
    Authorization: "Bearer <your-token>"
  write_relabel_configs:
    - source_labels: [__name__]
      regex: "node_.*"
      action: keep
```

優勢分析

1. 高效數據轉發

- 將數據持久化至高性能存儲系統，支持更複雜的分析。

2. 與外部工具無縫集成

- 可與 Grafana 等分析工具配合，進一步提升可視化與洞察能力。

3. 適用於大規模架構

- 在多集群架構中，集中式存儲使管理更加高效。



潛在局限

1. 傳輸延遲

- 推送數據的過程依賴於網絡與遠端存儲的性能，可能導致數據延遲。

2. 運維複雜性

- 外部存儲系統的安裝與維護需要額外的資源。

3. 查詢限制

- Prometheus 無法直接查詢遠程存儲數據，需依賴存儲系統的查詢功能。
-

適用場景

1. 大型企業監控

- 適合需要長期保存多集群數據的企業。

2. 跨地區監控管理

- 可將分布於不同地理位置的數據集中存儲，統一分析。

3. 歷史數據查詢

- 支持 SLA 報告與趨勢分析。
-

此頁側重於技術與應用的結合，幫助理解 Remote Write 的運行方式及其在實際業務場景中的優勢與限制。

視覺化概念建議

以下圖片展示 Prometheus Remote Write 的架構及應用情境：Prometheus 實例將數據推



送至遠程存儲（如 VictoriaMetrics 或 Thanos），後續通過 Grafana 等工具進行分析。

如果需要，我可以提供一個圖表或視覺化方式來展示 Prometheus Remote Write 的流程和應用場景，請讓我知道是否需要補充！



簡報第6頁的詳細說明如下，根據 **ChatGPT-Prometheus Remote Write vs Federation.pdf** 提供的內容：

第6頁 - Prometheus Remote Write 詳細說明

核心功能

Prometheus Remote Write 是將數據推送至外部存儲系統的機制。以下為功能亮點的具體說明：

1. 長期存儲

- 克服 Prometheus 本地存儲僅能保存短期數據的限制，支持數據的長期存儲與查詢。
- 配合系統（如 VictoriaMetrics、Thanos、Cortex 等），建立穩定的歷史數據檔案。

2. 多集群數據匯聚

- 適用於多個 Prometheus 實例的場景，將數據統一推送至集中式存儲，便於管理與分析。

3. 災難恢復

- 將監控數據同步到異地存儲，提升數據的冗餘性與可用性。



技術原理

Prometheus Remote Write 透過 HTTP 將樣本數據推送到配置的遠程存儲後端，支持以下流程：

- 利用 **write_relabel_configs** 過濾特定指標或數據範圍。
- 推送的數據包含標籤與數值，根據存儲後端的要求進行格式化。

配置示例：

```
remote_write:
- url: "http://remote-storage.example.com/api/v1/write"
  headers:
    Authorization: "Bearer <your-token>"
  write_relabel_configs:
    - source_labels: [__name__]
      regex: "node_.*"
      action: keep
```

優勢分析

1. 高效數據轉發

- 將數據持久化至高性能存儲系統，支持更複雜的分析。

2. 與外部工具無縫集成

- 可與 Grafana 等分析工具配合，進一步提升可視化與洞察能力。

3. 適用於大規模架構

- 在多集群架構中，集中式存儲使管理更加高效。

潛在局限

1. 傳輸延遲



- 推送數據的過程依賴於網絡與遠端存儲的性能，可能導致數據延遲。

2. 運維複雜性

- 外部存儲系統的安裝與維護需要額外的資源。

3. 查詢限制

- Prometheus 無法直接查詢遠程存儲數據，需依賴存儲系統的查詢功能。
-

適用場景

1. 大型企業監控

- 適合需要長期保存多集群數據的企業。

2. 跨地區監控管理

- 可將分布於不同地理位置的數據集中存儲，統一分析。

3. 歷史數據查詢

- 支持 SLA 報告與趨勢分析。
-

此頁側重於技術與應用的結合，幫助理解 Remote Write 的運行方式及其在實際業務場景中的優勢與限制。

視覺化概念建議

以下圖片展示 Prometheus Remote Write 的架構及應用情境：Prometheus 實例將數據推送至遠程存儲（如 VictoriaMetrics 或 Thanos），後續通過 Grafana 等工具進行分析。

如果需要，我可以提供一個圖表或視覺化方式來展示 Prometheus Remote Write 的流程和應用場景，請讓我知道是否需要補充！



參考ChatGPT-Prometheus Remote Write vs Federation.pdf，詳細說明簡報的第7頁



以下是簡報第7頁的詳細說明，根據 **ChatGPT-Prometheus Remote Write vs Federation.pdf** 的內容：

第7頁 - Prometheus Federation

核心功能

Prometheus Federation 是一種數據共享機制，用於在多個 Prometheus 實例之間共享監控數據，特別適合層次化監控架構。其核心功能如下：

1. 多集群數據匯聚

- 上層 Prometheus 實例通過抓取（scrape）下層 Prometheus 實例的數據，進行統一管理與監控。
- 支持跨集群的數據整合與分析。

2. 關鍵指標共享

- 僅抓取關鍵指標（如核心業務相關指標），減少上層實例的負擔。
- 支持動態篩選和調整抓取內容。

3. 層次化監控

- 建立層次化架構：每個子系統或區域的 Prometheus 實例負責收集局部數據，並通過 Federation 將核心數據匯聚至中央監控實例。



技術原理

Federation 使用 Prometheus 標準的抓取機制，從其他 Prometheus 實例的 `/federate` 接口抓取數據。

配置示例：

```
scrape_configs:
  - job_name: 'federation'
    scrape_interval: 30s
    metrics_path: '/federate'
    params:
      'match[]':
        - '{job="node_exporter"}'
        - 'up'
    static_configs:
      - targets:
          - 'prometheus-instance-1:9090'
          - 'prometheus-instance-2:9090'
```

關鍵配置項：

- `metrics_path`: 指定抓取目標的 `/federate` 接口。
- `params`: 使用 `match[]` 參數篩選需要抓取的數據。
- `targets`: 定義下層 Prometheus 實例的地址。

優勢

1. 內建支持

- Prometheus 本身支持 Federation 機制，無需安裝額外的存儲系統或服務。

2. 高數據即時性

- 與 Prometheus 本地數據抓取的延遲一致，確保數據的高即時性。

3. 數據過濾



- 支持通過 `match[]` 參數過濾數據，僅抓取需要的核心指標，減少不必要的負擔。
-

限制

1. 短期數據存儲

- 只適用於 Prometheus 短期數據存儲窗口，無法保存長期數據。

2. 負載影響

- 當上層實例抓取多個下層實例的數據時，可能導致下層實例的性能下降。

3. 有限數據處理能力

- 適合小範圍的數據聚合，無法處理大規模數據的複雜計算。
-

應用場景

1. 多層次監控架構

- 適合分布式系統，讓每個子系統有自己的 Prometheus 實例，並將核心指標匯聚至上層實例進行分析。

2. 跨環境監控

- 在 Dev、QA、Prod 等不同環境部署 Prometheus 實例，通過 Federation 匯聚相關數據進行全局統一分析。

3. 核心指標分析

- 僅抓取關鍵業務指標進行監控，減少抓取頻率對下層系統的影響。
-

此頁重點描述 **Prometheus Federation** 的特點與應用場景，強調其在分層監控和關鍵指標共享中的作用。



如果需要，我可以協助製作一個圖表，展示 Prometheus Federation 的架構與數據流動方式，以更直觀地幫助理解。請告訴我是否需要補充！



以下是簡報第7頁的詳細說明，根據 **ChatGPT-Prometheus Remote Write vs Federation.pdf** 的內容：

第7頁 - Prometheus Federation

核心功能

Prometheus Federation 是一種數據共享機制，用於在多個 Prometheus 實例之間共享監控數據，特別適合層次化監控架構。其核心功能如下：

1. 多集群數據匯聚

- 上層 Prometheus 實例通過抓取（scrape）下層 Prometheus 實例的數據，進行統一管理與監控。
- 支持跨集群的數據整合與分析。

2. 關鍵指標共享

- 僅抓取關鍵指標（如核心業務相關指標），減少上層實例的負擔。
- 支持動態篩選和調整抓取內容。

3. 層次化監控

- 建立層次化架構：每個子系統或區域的 Prometheus 實例負責收集局部數據，並通過 Federation 將核心數據匯聚至中央監控實例。

技術原理



Federation 使用 Prometheus 標準的抓取機制，從其他 Prometheus 實例的 `/federate` 接口抓取數據。

配置示例：

```
scrape_configs:
  - job_name: 'federation'
    scrape_interval: 30s
    metrics_path: '/federate'
    params:
      'match[]':
        - '{job="node_exporter"}'
        - 'up'
    static_configs:
      - targets:
          - 'prometheus-instance-1:9090'
          - 'prometheus-instance-2:9090'
```

關鍵配置項：

- `metrics_path`：指定抓取目標的 `/federate` 接口。
- `params`：使用 `match[]` 參數篩選需要抓取的數據。
- `targets`：定義下層 Prometheus 實例的地址。

優勢

1. 內建支持

- Prometheus 本身支持 Federation 機制，無需安裝額外的存儲系統或服務。

2. 高數據即時性

- 與 Prometheus 本地數據抓取的延遲一致，確保數據的高即時性。

3. 數據過濾

- 支持通過 `match[]` 參數過濾數據，僅抓取需要的核心指標，減少不必要的負擔。



限制

1. 短期數據存儲

- 只適用於 Prometheus 短期數據存儲窗口，無法保存長期數據。

2. 負載影響

- 當上層實例抓取多個下層實例的數據時，可能導致下層實例的性能下降。

3. 有限數據處理能力

- 適合小範圍的數據聚合，無法處理大規模數據的複雜計算。
-

應用場景

1. 多層次監控架構

- 適合分布式系統，讓每個子系統有自己的 Prometheus 實例，並將核心指標匯聚至上層實例進行分析。

2. 跨環境監控

- 在 Dev、QA、Prod 等不同環境部署 Prometheus 實例，通過 Federation 匯聚相關數據進行全局統一分析。

3. 核心指標分析

- 僅抓取關鍵業務指標進行監控，減少抓取頻率對下層系統的影響。
-

此頁重點描述 **Prometheus Federation** 的特點與應用場景，強調其在分層監控和關鍵指標共享中的作用。

如果需要，我可以協助製作一個圖表，展示 Prometheus Federation 的架構與數據流動方式，以更直觀地幫助理解。請告訴我是否需要補充！



參考ChatGPT-Prometheus Remote Write vs Federation.pdf，詳細說明簡報的第8頁



以下是簡報第8頁的詳細說明，基於 **ChatGPT-Prometheus Remote Write vs Federation.pdf** 的內容：

第8頁 - Prometheus Federation 詳細說明

核心功能

Prometheus Federation 是用於實現數據共享和層次化監控的關鍵機制，專注於上層和下層 Prometheus 實例之間的數據抓取。以下是功能的詳細說明：

1. 層次化監控架構

- 每個子系統部署自己的 Prometheus 實例負責收集局部數據。
- 上層 Prometheus 通過抓取接口匯聚各子系統的核心指標，實現集中管理與監控。

2. 數據共享與聚合

- 支持從多個 Prometheus 實例中抓取數據，並進行匯總、分析。
- 適合跨集群的數據監控與聚合需求。

3. 動態抓取與指標篩選

- 支持僅抓取特定指標，減少上層負載，提升監控效率。
- 可根據需求靈活調整抓取的數據範圍。



技術實現

Federation 通過標準的 Prometheus 抓取機制實現，抓取目標是下層 Prometheus 的 `/federate` 接口。

配置示例：

```
scrape_configs:
  - job_name: 'federation'
    scrape_interval: 15s
    metrics_path: '/federate'
    params:
      'match[]':
        - '{job="node_exporter"}'
        - 'up'
    static_configs:
      - targets:
          - 'prometheus-instance-1:9090'
          - 'prometheus-instance-2:9090'
```

關鍵配置項目：

1. `job_name`：為此抓取任務指定名稱。
2. `scrape_interval`：定義抓取的時間間隔。
3. `metrics_path`：設定抓取目標的 `/federate` 接口。
4. `params.match[]`：篩選需要抓取的指標。
5. `static_configs.targets`：設定下層 Prometheus 實例的地址。

優勢

1. 內建支持
 - Prometheus 原生支持，無需安裝或配置額外的存儲系統。
2. 高即時性



- 抓取機制延遲低，數據更新速度與本地存儲相當。

3. 靈活數據篩選

- 支持基於需求的數據篩選，只抓取關鍵指標，減少上層的負載壓力。
-

限制

1. 短期數據局限

- 只適用於短期數據存儲窗口，不支持長期數據存儲。

2. 負載影響

- 如果上層需要抓取多個下層實例，可能對下層實例的性能造成壓力。

3. 有限數據處理能力

- 適用於小規模數據的聚合和共享，不適合大規模分析。
-

適用場景

1. 多層次監控架構

- 用於大規模分布式系統，讓每個子系統的數據集中到上層進行統一監控。

2. 跨環境監控

- 監控 Dev、QA、Prod 等多環境，並匯聚到中央實例分析。

3. 核心指標共享

- 僅匯聚核心業務指標到上層，減少不必要的數據抓取。
-

結論



此頁詳細闡述了 **Prometheus Federation** 的操作方式及其優缺點，幫助理解其在層次化監控與跨集群數據匯聚中的應用價值。

如需視覺化圖表來展示 Federation 的數據流與架構，請告知，我可以協助製作！



參考ChatGPT-Prometheus Remote Write vs Federation.pdf，詳細說明簡報的第9頁



以下是簡報第9頁的詳細說明，基於 **ChatGPT-Prometheus Remote Write vs Federation.pdf** 的內容：

第9頁 - Remote Write 與 Federation 的比較表

比較概述

此頁詳細對比了 **Prometheus Remote Write** 和 **Prometheus Federation** 的功能、特性和適用場景，幫助使用者根據需求選擇合適的技術方案。

比較細節

比較項目	Remote Write	Federation
用途	長期存儲和集中式數據分析	層次化監控和指標共享
數據傳輸方向	推送（Push）	拉取（Pull）
即時性	視網絡和存儲延遲而定，可能導致延遲	高即時性，與本地抓取的延遲相當



比較項目	Remote Write	Federation
依賴性	依賴外部存儲系統（如 VictoriaMetrics、Thanos）	完全內建，無需外部依賴
數據篩選	通過 `relabel_configs` 配置進行篩選	通過 `match[]` 配置抓取所需指標
適合場景	長期存儲、大規模數據匯聚	分層監控、即時數據共享

詳細比較

1. 用途

- **Remote Write** :
 - 適合需要將數據保存至外部系統以支持長期存儲和集中化分析的場景。
- **Federation** :
 - 適合需要在多個 Prometheus 實例間進行層次化監控和指標共享的場景。

2. 數據傳輸方向

- **Remote Write** 使用推送機制，Prometheus 主動將數據推送至外部存儲系統。
- **Federation** 使用拉取機制，上層 Prometheus 實例通過抓取（scrape）下層實例的數據。

3. 即時性

- **Remote Write** 的數據即時性受到網絡延遲與存儲系統性能的影響。
- **Federation** 提供高即時性，抓取的數據與本地存儲一致。

4. 依賴性

- **Remote Write** 依賴於外部存儲系統（如 VictoriaMetrics、Thanos 等），需額外配置和維護。
- **Federation** 完全基於 Prometheus 內建功能，無需依賴外部系統。



5. 數據篩選

- **Remote Write** 可通過 ``relabel_configs`` 配置過濾不需要的數據。
- **Federation** 可使用 ``match[]`` 配置篩選具體需要抓取的指標。

6. 適合場景

- **Remote Write** 適合長期存儲、大規模數據匯聚以及需要與其他工具（如 Grafana）集成的場景。
- **Federation** 適合多層次監控架構，以及需要即時數據共享的場景。

建議與結論

- **使用 Remote Write：**
 - 如果需求包括長期存儲或集中管理多集群數據，選擇 Remote Write。
 - 適合需要異地備份和災難恢復的場景。
- **使用 Federation：**
 - 如果需求是匯聚分布式系統中的關鍵指標或實現層次化監控，選擇 Federation。
 - 適合需要即時數據共享，且不需要長期存儲的場景。

此頁提供了清晰的比較，幫助讀者快速了解 **Remote Write** 和 **Federation** 各自的特性及應用場景，便於根據需求進行選擇。

如需製作圖表或具體的案例說明，請告知，我可以協助補充！





參考ChatGPT-Prometheus Remote Write vs Federation.pdf，詳細說明簡報的第10頁



以下是簡報第10頁的詳細說明，基於 **ChatGPT-Prometheus Remote Write vs Federation.pdf** 的內容：

第10頁 - 結論

主要觀點總結

本頁對 **Prometheus Remote Write** 和 **Prometheus Federation** 進行總結，提供具體技術選擇建議，並討論性能與成本的平衡點。

使用情境總結

1. Remote Write：

- 適用於需要 **長期存儲** 或 **大規模數據匯聚** 的場景。
- 利用外部存儲系統（如 VictoriaMetrics 或 Thanos），支持長期歷史數據查詢和集中化分析。
- 適合需要異地備份和災難恢復的情境。

2. Federation：

- 適合 **層次化監控** 和 **即時數據共享**。
- 可用於跨環境或多層次架構下的指標匯聚，實現分層監控。
- 適用於抓取關鍵指標，減少數據處理負擔。



技術選擇建議

- 如果需求是 **跨多個數據源匯聚指標**，應優先考慮 **Federation**。
 - 適合多環境、多集群的即時監控需求。
 - 如果需要 **長期存儲和歷史數據分析**，選擇 **Remote Write**。
 - 適合需要與外部存儲解決方案集成的場景。
-

性能與成本的平衡

1. 資源限制考量

- **Remote Write** 增加了對外部存儲系統的依賴，可能需要額外的硬體、軟體與運維成本。
- **Federation** 不依賴外部存儲，對資源的需求相對較低，適合成本敏感的情境。

2. 數據存儲需求

- 在需要保存大量歷史數據的情況下，應優先選擇 **Remote Write**，儘管成本可能較高。
 - 如果數據的主要需求是即時監控，而非歷史查詢，則 **Federation** 更為經濟。
-

總結與建議

- 根據具體需求，選擇合適的技術：
 - 長期存儲需求：**Remote Write**。
 - 分層數據監控與即時性要求：**Federation**。
 - 綜合考量性能與成本，在資源有限的情況下，應選擇更符合需求的方案。
-



簡報第10頁重點： 此頁幫助聽眾快速回顧 **Remote Write** 與 **Federation** 的核心特性，並提供應用場景的實用建議，特別適合在技術選擇時進行參考。

如果需要圖表或額外的例子來說明性能與成本的平衡，請隨時告訴我！



一文搞懂 VictoriaMetrics 的使用-...
PDF

參考一文搞懂 VictoriaMetrics 的使用-阳明的博客.pdf，詳細說明簡報的第13頁



簡報第13頁 - VictoriaMetrics 簡介與應用

核心內容

VictoriaMetrics 是一種高效能、可擴展的時間序列數據存儲和監控解決方案。本頁針對其基本特性進行介紹，以下是其詳細說明：

VictoriaMetrics 的關鍵特點

1. 高效性能與資源使用

- 支持高基數的時間序列數據存儲和查詢，比 InfluxDB 和 TimescaleDB 提高約 20 倍的性能。
- 在內存使用上比其他系統（如 Prometheus 或 Thanos）減少 7 倍，存儲壓縮效率高。





2. 兼容性

- 完全兼容 Prometheus API，支持用於 Grafana 的數據源。
- 提供 Prometheus `remote_write` 支援，能無縫集成至現有的 Prometheus 架構。

3. 穩定與安全

- 適應高延遲 I/O 環境，針對低 IOPS 進行了優化。
- 在非正常關機時（如 OOM 或硬件重置），數據仍能保持完整，避免損壞。

應用場景

1. 大型監控數據存儲

- 適用於需要長期保存和查詢大規模數據的場景，例如 SLA 報告和業務趨勢分析。

2. 高頻數據環境

- 在高基數和高頻數據場景下（如數百萬數據點每秒），VictoriaMetrics 提供穩定支持。

3. 多租戶支持

- 支持多租戶架構，適合跨團隊或多客戶的數據隔離需求。

部署模式

1. 單節點模式

- 適合低於每秒 100 萬數據點的需求。
- 易於部署和維護，但不支持內建告警功能。



2. 集群模式

- 適合高並發環境，支持數據水平擴展。
- 將功能劃分為 `vmstorage`、`vminsert` 和 `vmselect` 等組件，實現高效存儲與查詢。

優勢總結

- 低資源需求、高效能壓縮與查詢，特別適合高基數數據和大規模監控。
- 簡單操作，支持用戶快速部署和集成，無需過多配置。

結論

VictoriaMetrics 作為 Prometheus 的補充或替代方案，特別適合需要長期存儲與高效查詢的業務需求。

若需要進一步的技術細節或架構圖，請告知，我可以協助補充！



參考一文搞懂 VictoriaMetrics 的使用-阳明的博客.pdf，詳細說明簡報的第14頁



簡報第14頁 - 為什麼選擇 VictoriaMetrics

根據 《一文搞懂 VictoriaMetrics 的使用》，第14頁著重說明選擇 VictoriaMetrics 的優勢，以下是詳細說明：



選擇 VictoriaMetrics 的理由

1. 高效性能

- **數據存儲與查詢性能提升：**
 - 與 InfluxDB 和 TimescaleDB 相比，查詢性能提升高達 20 倍。
 - 適合處理高基數時間序列，例如每秒數百萬數據點的場景。
 - **內存與存儲效率高：**
 - 比 Prometheus、Thanos 或 Cortex 減少約 7 倍內存使用量。
 - 使用高壓縮存儲方式，可將數據存儲需求降低至原數據的 1/7。
-

2. 完整兼容性

- **Prometheus API 支援：**
 - 完全支持 Prometheus 協議，無需修改原有監控系統即可快速集成。
 - **多種數據協議支持：**
 - 包括 Influx Line、Graphite、OpenTSDB、Prometheus Remote Write 等協議。
-

3. 易用性

- **部署簡單：**
 - 單節點模式支持快速安裝，無需外部依賴；集群模式支持水平擴展以應對高負載。
 - 提供簡單的配置選項，支持通過命令行參數進行靈活設置。
- **Web UI 與工具支持：**



- 自帶 vmui（簡單 Web 界面）進行查詢與管理，亦支持使用 Grafana 作為前端工具進行數據展示。
-

4. 高可用與穩定性

- **針對高延遲環境優化：**
 - 在低 IOPS 存儲條件下依然保持良好性能。
 - **防數據損壞機制：**
 - 在非正常關機（如 OOM 或硬件重置）時，數據完整性可得到保證。
-

5. 多租戶支持

- 支持多租戶架構，適合 SaaS 平台及多客戶場景，數據隔離且不影響性能。
-

應用場景

1. 長期存儲需求：

- 適合需要保存數月或數年的監控數據場景，例如歷史查詢與 SLA 報告。

2. 高並發數據環境：

- 適合處理高基數、每秒數百萬數據點的監控場景。

3. 大規模集群管理：

- 支持通過水平擴展來應對數據量增長，對於多租戶應用效果尤為顯著。
-



結論

VictoriaMetrics 是針對高效能、簡單部署和低資源消耗而設計的時間序列存儲解決方案。在性能、穩定性、兼容性和可擴展性方面，VictoriaMetrics 脫穎而出，是替代 Prometheus 和其他存儲方案的理想選擇。

如需進一步補充或展示架構圖，請隨時告訴我！



參考一文搞懂 VictoriaMetrics 的使用-阳明的博客.pdf，詳細說明簡報的第15頁



簡報第15頁 - VictoriaMetrics 的功能亮點

根據《一文搞懂 VictoriaMetrics 的使用》，第15頁專注於介紹 VictoriaMetrics 的核心功能亮點。以下是詳細說明：

功能亮點

1. 高效數據存儲與查詢

- **高效數據壓縮：**
 - 使用高性能數據壓縮技術，與 Prometheus、Thanos 相比，存儲需求降低約 7 倍。
 - 在有限的磁碟空間內存儲更多數據點，支持長期數據保留。
- **卓越的查詢性能：**
 - 在處理高基數時間序列（如每秒百萬數據點）時，性能比 InfluxDB 和 TimescaleDB 高出 20 倍。



2. 完全兼容 Prometheus

- **支持 Prometheus API：**
 - 提供與 Prometheus 完全兼容的查詢與數據寫入接口，無需修改現有架構即可集成。
 - 支援 Prometheus `remote_write` 功能，用於數據的遠程存儲。
 - **支持 PromQL 語法：**
 - 用戶可直接使用 Prometheus 的查詢語言進行數據分析與可視化，與 Grafana 無縫整合。
-

3. 靈活的部署模式

- **單節點模式：**
 - 適合低於每秒 100 萬數據點的場景，部署簡單、資源佔用少。
 - 可快速啟動並進行監控數據存儲和查詢。
 - **集群模式：**
 - 支持高性能水平擴展，適合大型集群和高併發環境。
 - 使用 `vmstorage`、`vminsert` 和 `vmselect` 組件實現存儲、數據插入和查詢的高效分工。
-

4. 高可用性與容錯能力

- **針對高延遲與低 IOPS 優化：**
 - 即使在低 IOPS 環境下，依然保持穩定性能。
- **數據完整性保證：**



- 在非正常關機（如 OOM、硬體故障）後，數據仍然安全無損。
-

5. 多租戶支持

- 支持多租戶架構，每個租戶的數據存儲均相互隔離。
 - 提供基於 `accountID` 和 `projectID` 的命名空間，方便數據管理。
-

6. 可觀測性與工具支持

- **內建 Web UI (vmui) :**
 - 提供基礎的數據查詢與管理功能，支持數據目標 (targets) 查看。
 - **與 Grafana 集成 :**
 - 作為 Grafana 數據源，提供直觀的數據可視化能力。
-

7. 支持多種數據協議

- **兼容多種協議 :**
 - 包括 Influx Line、Graphite、OpenTSDB、Prometheus Remote Write 等多種協議。
 - 支持數據從多源輸入，靈活應對不同場景。
-

總結

VictoriaMetrics 憑藉高效的性能、靈活的部署方式以及對 Prometheus 的完美兼容性，成為時間序列數據存儲與查詢的首選解決方案。



若需補充架構圖或操作細節，請告知，我可以協助進一步說明！



參考一文搞懂 VictoriaMetrics 的使用-阳明的博客.pdf，詳細說明簡報的第16頁



簡報第16頁 - 部署模式 - 單節點部署

根據《一文搞懂 VictoriaMetrics 的使用》，第16頁重點介紹 VictoriaMetrics 的單節點部署模式及其應用場景。以下是詳細說明：

單節點部署模式

適用場景

- 適合數據流量較小或中型規模的監控系統。
- 當監控指標數據低於每秒 100 萬數據點（data points）時，單節點模式是簡單且高效的選擇。
- 資源需求較低，適合成本敏感的部署環境。

優勢

1. 部署簡單

- 單節點部署只需啟動一個可執行文件或容器，即可完成監控數據的存儲與查詢功能。
- 可快速上線，支持快速測試與驗證。



2. 資源占用低

- 無需額外的服務組件即可運行。
- 相比集群模式，對 CPU、內存及磁碟資源需求更少。

3. 維護便利

- 所有數據存儲在單一節點，無需考慮多節點之間的同步與一致性。
- 運行中容易進行故障排查與恢復。

技術配置與步驟

1. 運行模式

- 可通過下載官方二進制文件或使用 Docker 容器啟動：

```
docker run -d --name victoria-metrics \  
-p 8428:8428 \  
-v /path/to/data:/victoria-metrics-data \  
victoriametrics/victoria-metrics:v1.91.3
```

- 重要參數：
 - `-storageDataPath``：指定數據存儲目錄。
 - `-retentionPeriod``：設置數據保存週期。

2. 持久化存儲

- 使用持久化存儲掛載數據目錄，確保數據安全：
 - 例如：``/victoria-metrics-data``。

3. 數據查詢與監控

- 通過瀏覽器訪問 Web UI 接口：



- ``http://<host>:8428``
- 可直接查詢時間序列數據並檢查指標。

4. Prometheus 集成

- 配置 Prometheus 遠程寫入到 VictoriaMetrics：

```
remote_write:  
- url: "http://<victoria-metrics-host>:8428/api/v1/write"
```

限制

1. 不支持內建告警功能

- 單節點模式無法直接處理告警需求，需結合外部工具（如 Alertmanager）。

2. 性能與容量限制

- 當數據量達到每秒 100 萬數據點以上時，性能可能下降，需考慮集群模式。

單節點模式適用情境

1. 小規模監控系統

- 適合中小企業或開發測試環境。

2. 簡單快速部署

- 適用於快速搭建監控系統並驗證 VictoriaMetrics 功能。



結論

VictoriaMetrics 的單節點模式因其部署簡單、資源消耗低的特性，是小型或中型監控需求的理想選擇。當監控需求增加時，可平滑過渡至集群模式。

如需補充配置範例或可視化架構圖，請隨時告知！



參考一文搞懂 VictoriaMetrics 的使用-阳明的博客.pdf，詳細說明簡報的第17頁



簡報第17頁 - 部署模式 - 集群模式架構

根據《一文搞懂 VictoriaMetrics 的使用》，第17頁重點說明了 VictoriaMetrics 的集群模式架構及其特點。以下是詳細說明：

集群模式的概述

適用場景

- 適用於需要處理高基數、高頻率、大規模數據的場景。
- 當監控數據量超過每秒百萬數據點，或需要多租戶支持時，集群模式是一個理想選擇。

核心架構

VictoriaMetrics 集群模式的架構將系統功能分為以下三個主要組件，這些組件可以根據需求進行水平擴展：

1. **vmstorage**（數據存儲）



- 功能：負責存儲原始數據並根據查詢請求返回過濾後的數據。
- 特點：
 - 數據存儲在本地，使用高效壓縮技術節省磁碟空間。
 - 支援多副本存儲，提高數據可用性。
 - 在存儲空間不足時自動切換為只讀模式。

2. **vminsert**（數據寫入）

- 功能：負責接收數據並將其分發至多個 **vmstorage** 節點。
- 特點：
 - 使用一致性哈希機制將數據分散存儲到不同的 **vmstorage** 節點。
 - 支持並行寫入，提升數據接收性能。

3. **vmselect**（數據查詢）

- 功能：處理查詢請求，從 **vmstorage** 中檢索數據並返回聚合結果。
- 特點：
 - 支援高併發查詢。
 - 自帶數據去重功能，可處理多副本數據。

集群架構的運作流程

1. 用戶通過 **vmselect** 發送查詢請求。
 2. **vmselect** 根據請求內容與標籤篩選條件，從多個 **vmstorage** 節點檢索數據。
 3. **vminsert** 負責數據的接收與分片存儲，將數據分佈到適合的 **vmstorage** 節點。
 4. 數據存儲於 **vmstorage** 中，並根據需求進行壓縮與持久化。
-



部署推薦

1. 基本集群需求

- 每個組件至少部署 2 個副本以確保高可用性。
- 節點之間相互獨立，不共享狀態，提升系統的穩定性。

2. 擴展建議

- 添加更多 **vmstorage** 節點以支持更大的活躍時間序列數量。
 - 增加 **vminsert** 節點以提高數據寫入速度。
 - 增加 **vmselect** 節點以提升查詢性能和並發能力。
-

特點與優勢

1. 性能

- 支持水平擴展，能夠根據需求增加節點來應對數據量增長。
- 高效的數據壓縮和索引設計，降低存儲和計算成本。

2. 靈活性

- 各組件可獨立部署與擴展，方便用戶根據業務需求靈活調整架構。

3. 高可用性

- 節點故障時，其餘節點可繼續提供服務，保證集群穩定運行。

4. 多租戶支持

- 通過 ``accountID`` 和 ``projectID`` 實現數據隔離，支持多租戶架構。
-



限制

- 相比單節點模式，集群模式的部署與維護複雜度更高。
- 對資源需求更高，需要考慮硬體配置和節點之間的網絡性能。

結論

VictoriaMetrics 集群模式是針對高負載、大規模監控場景設計的解決方案，能提供高性能的數據存儲與查詢服務，同時保持靈活性和可擴展性。

如需詳細的部署示例或配置文件，請隨時告知！



參考一文搞懂 VictoriaMetrics 的使用-阳明的博客.pdf，詳細說明簡報的第18頁



簡報第18頁 - 部署步驟和配置

根據《一文搞懂 VictoriaMetrics 的使用》，第18頁重點描述了 VictoriaMetrics 的部署步驟與配置方法。以下是詳細說明：

部署方式

VictoriaMetrics 支援多種部署方式，適用於不同場景和需求：

1. 單節點部署

- 適用場景：
 - 小規模監控需求（每秒低於 100 萬數據點）。





- 測試或開發環境。
- 步驟：
 1. 下載官方二進制文件：
從 [官方下載頁面](#) 獲取適合的版本。
 2. 運行二進制文件：

```
./victoria-metrics-prod -storageDataPath=/path/to/data
```

3. **Docker 部署：**
使用 Docker 運行 VictoriaMetrics：

```
docker run -d --name victoria-metrics \  
-p 8428:8428 \  
-v /path/to/data:/victoria-metrics-data \  
victoriametrics/victoria-metrics:v1.91.3
```

2. 集群模式部署

- 適用場景：
 - 大規模監控需求（每秒數百萬數據點）。
 - 高性能和高可用需求。
- 步驟：
 1. 下載或構建集群組件：
 - ``vmstorage``：負責數據存儲。
 - ``vminsert``：負責數據寫入。
 - ``vmselect``：負責數據查詢。
 2. 啟動各個組件：
 - 使用 Docker 或 Kubernetes 部署各組件，確保它們能互相通信。
 - 配置命令示例：



```
docker run -d --name vmstorage \
  -v /path/to/data:/storage \
  -p 8400:8400 victoriametrics/vmstorage:v1.91.3
docker run -d --name vminsert \
  -e VMSTORAGE_URLS=http://vmstorage:8400 \
  -p 8401:8401 victoriametrics/vminsert:v1.91.3
docker run -d --name vmselect \
  -e VMSTORAGE_URLS=http://vmstorage:8400 \
  -p 8402:8402 victoriametrics/vmselect:v1.91.3
```

3. 配置 **Prometheus 集成**：在 Prometheus 配置文件中添加 `remote_write`：

```
remote_write:
  - url: "http://<vminsert-host>:8401/insert"
```

核心配置參數

1. 數據存儲配置

- `-storageDataPath=/path/to/data`：指定數據存儲目錄。
- `-retentionPeriod=12`：設置數據保存週期（默認 1 年）。

2. 資源優化參數

- `-memory.allowedPercent=60`：限制 VictoriaMetrics 的內存使用。
- `-maxConcurrentInserts=16`：優化數據插入並發數。

3. 集群節點配置

- `VMSTORAGE_URLS`：設置 `vminsert` 和 `vmselect` 與 `vmstorage` 的連接地址。
- `-replicationFactor=2`：設置數據副本數量，確保高可用。



監控與運維建議

1. Web 界面監控

- 訪問 `http://<host>:8428`，檢查節點狀態與數據指標。

2. 資源監控

- 使用外部監控工具（如 Prometheus）監控 VictoriaMetrics 的資源消耗（CPU、內存、磁碟 I/O 等）。

3. 定期備份

- 定期備份 `/path/to/data` 目錄以保護數據。

結論

VictoriaMetrics 提供簡單高效的部署方式，無論是單節點還是集群模式，都能靈活應對不同的監控需求。集群模式雖然複雜，但提供了更高的性能和擴展能力。

如需進一步補充實際案例或配置細節，請隨時告知！



參考一文搞懂 VictoriaMetrics 的使用-阳明的博客.pdf，詳細說明簡報的第19頁



簡報第19頁 - 優勢與限制

根據《一文搞懂 VictoriaMetrics 的使用》，第19頁重點介紹 VictoriaMetrics 的核心優勢與潛在限制，以下為詳細說明：



VictoriaMetrics 的核心優勢

1. 高性能

- **數據處理能力：**
 - 每秒支持數百萬數據點的高基數處理能力，性能比 InfluxDB 和 TimescaleDB 高出 20 倍。
 - 適用於高頻監控場景，如雲原生架構和物聯網應用。
- **高效壓縮技術：**
 - 使用先進的壓縮算法，存儲數據量減少至原本的 1/7，極大降低磁碟使用量和存儲成本。

2. 低資源需求

- **內存與計算效率：**
 - 比 Prometheus 減少約 7 倍的內存使用量，在低資源環境下依然表現穩定。
 - 優化低 IOPS 存儲環境，適用於高延遲 I/O 場景。

3. 靈活部署

- **單節點與集群模式支持：**
 - 單節點模式簡單易用，適合中小規模應用；集群模式適合高併發和大數據量處理。
- **易於擴展：**
 - 集群模式支持水平擴展，通過增加節點即可應對數據量和查詢負載的增長。



4. 完全兼容性

- **Prometheus API 支援：**
 - 無需修改現有的 Prometheus 配置即可集成，支持 `remote_write` 和 PromQL 查詢語言。
 - **多種協議兼容：**
 - 支持 Prometheus、InfluxDB、OpenTSDB 等多種數據寫入協議，適用多元化場景。
-

5. 高可用與數據完整性

- **自動數據恢復：**
 - 在 OOM 或硬件重置後，保證數據不丟失且保持一致性。
 - **多副本支持：**
 - 提供數據冗餘機制，確保高可用和高容錯能力。
-

VictoriaMetrics 的潛在限制

1. 高基數數據的挑戰

- 雖然 VictoriaMetrics 在高基數處理上表現優異，但當指標基數超過極限（數十億條）時，可能需要更多資源或更複雜的集群配置。
-

2. 內建功能的限制

- **缺乏內建告警功能：**
 - VictoriaMetrics 本身不提供告警處理功能，需依賴外部工具（如 Alertmanager）完成。



- **單節點模式限制：**

- 單節點模式在高負載下性能受限，當數據點頻率超過每秒百萬條時，需遷移至集群模式。

3. 部署與運維複雜性

- **集群模式運維成本高：**

- 需要專業的架構設計與資源分配，對於中小型團隊可能增加運維負擔。

- **依賴於穩定的網絡和硬體：**

- 集群組件（vmstorage、vminsert、vmselect）之間的通信對網絡穩定性有較高要求。

結論

VictoriaMetrics 在高性能、資源效率、兼容性和部署靈活性方面表現突出，適合處理高基數、高頻率的大規模數據存儲需求。

然而，其在內建功能與運維複雜度上的限制，需根據場景合理評估，並結合外部工具實現最佳化部署。

如需補充實例或圖表說明，請隨時告知！



參考一文搞懂 VictoriaMetrics 的使用-阳明的博客.pdf，詳細說明簡報的第20頁





簡報第20頁 - 使用場景

根據《一文搞懂 VictoriaMetrics 的使用》，第20頁重點介紹了 VictoriaMetrics 的適用場景及其在不同業務環境中的應用價值。以下是詳細說明：

VictoriaMetrics 的典型使用場景

1. 大規模監控數據存儲

- **適用場景：**
 - 需要處理高基數、高頻率的監控數據，例如雲原生架構、IoT 設備或金融監控系統。
- **應用價值：**
 - 高效的數據壓縮技術，節省存儲資源並支持長期數據保存。
 - 卓越的查詢性能，可在海量數據中快速完成歷史查詢與趨勢分析。

2. SLA 報告與歷史查詢

- **適用場景：**
 - 需要生成服務級別協議（SLA）報告並進行長期數據查詢的場景，例如企業 IT 運維、客戶支持數據分析。
- **應用價值：**
 - 支持按需保存數據的時間段，滿足不同 SLA 報告需求。
 - 與 Grafana 集成，提供直觀的 SLA 指標展示。



3. 多租戶架構

- **適用場景：**
 - 在 SaaS 平台或跨部門的業務中，需要多租戶支持以隔離不同客戶或團隊的數據。
 - **應用價值：**
 - 提供基於 `accountID` 和 `projectID` 的命名空間，確保數據隔離且查詢性能穩定。
 - 可根據租戶需求進行數據配額和管理，避免資源競爭。
-

4. 高並發查詢場景

- **適用場景：**
 - 需要在多用戶、高併發下進行實時監控查詢的場景，例如金融風控系統和大型電商平台。
 - **應用價值：**
 - vmselect 支持高效的併發查詢，並能處理多副本數據去重。
 - 優化查詢性能，即使在高併發請求下仍保持穩定響應。
-

5. 異地備份與數據冗餘

- **適用場景：**
 - 需要異地數據備份以應對災難恢復（DR）的場景，例如銀行、保險和醫療系統。
- **應用價值：**
 - 透過多副本存儲支持數據冗餘，增強系統可靠性。



- 與 Prometheus 的 remote_write 集成，可將數據推送到遠程的 VictoriaMetrics 節點進行存儲。

實例應用

1. 雲原生監控系統：

- 在 Kubernetes 集群中，VictoriaMetrics 作為 Prometheus 的後端存儲，提供高性能的數據存儲與查詢能力。

2. 物聯網平台：

- IoT 設備生成大量高頻數據，VictoriaMetrics 可通過高壓縮技術減少存儲開銷，並支持實時查詢。

3. 大型數據分析平臺：

- 使用 VictoriaMetrics 儲存指標數據，與數據分析工具（如 Grafana）整合，用於業務趨勢分析。

結論

VictoriaMetrics 憑藉其高性能、多租戶支持和靈活的部署方式，能夠應對多種監控和存儲需求，特別是在高基數和長期存儲場景中具有明顯優勢。

如需補充架構圖或實例配置，請隨時告知！



參考一文搞懂 VictoriaMetrics 的使用-阳明的博客.pdf，詳細說明簡報的第21頁



簡報第21頁 - 配置與運維建議

根據《一文搞懂 VictoriaMetrics 的使用》，第21頁重點討論了 VictoriaMetrics 的配置建議與運維最佳實踐。以下是詳細說明：

1. 配置建議

核心配置參數

- **數據存儲相關**
 - ``-storageDataPath=/path/to/data``：設置數據存儲的磁碟路徑，確保使用高速磁碟（如 SSD）提升性能。
 - ``-retentionPeriod=12``：定義數據的保存周期，默認為 12 個月，可根據需求調整。
- **內存管理**
 - ``-memory.allowedPercent=60``：限制 VictoriaMetrics 可使用的內存比例，避免高內存消耗影響系統穩定性。
 - ``-maxConcurrentInserts=16``：控制最大並發數據插入數量，優化寫入性能。
- **數據壓縮**
 - VictoriaMetrics 使用默認的高效壓縮算法，無需手動配置，但可以根據磁碟空間需求適當調整保存時間。

集群模式配置

- **vmstorage 節點**
 - 配置多個 vmstorage 節點以分散數據存儲壓力。



- 建議配置 ``replicationFactor=2``，確保數據冗餘和高可用性。
 - **vminsert 節點**
 - 配置數據分片機制，確保寫入操作均衡分佈至所有 vmstorage 節點。
 - ``VMSTORAGE_URLS`` 用於指定 vmstorage 的連接地址。
 - **vmselect 節點**
 - 增加查詢節點數量以提升高併發查詢性能。
 - 確保 vmselect 與 vmstorage 的網絡連通性，避免查詢延遲。
-

2. 運維建議

數據保護

- **定期備份**
 - 定期備份 ``-storageDataPath`` 目錄，避免數據因硬件故障或意外損壞而丟失。
 - 可使用工具（如 rsync 或備份軟件）同步數據到異地存儲。
 - **多副本冗餘**
 - 在集群模式中，啟用多副本存儲以確保高可用性。
-

性能監控

- **資源監控**
 - 使用 Prometheus 監控 VictoriaMetrics 的資源消耗（如 CPU、內存和磁碟 I/O）。
 - 關鍵指標：
 - ``vm_storage_write_requests_total``：寫入請求數。
 - ``vm_storage_read_requests_total``：讀取請求數。



- ``vm_memory_usage_bytes``：內存使用量。
 - **健康檢查**
 - 通過 Web 界面（``http://<host>:8428/health``）檢查服務的健康狀態。
-

問題排查

- **常見問題**
 - **高延遲**：檢查網絡連接和磁碟 I/O 性能，必要時升級硬件。
 - **查詢緩慢**：調整 ``-search.maxQueryDuration`` 和 ``-search.maxLookback`` 配置，優化查詢窗口。
 - **日誌分析**
 - 查看 VictoriaMetrics 的運行日誌檔案，檢查錯誤和異常情況。
-

資源優化

- **硬體建議**
 - 優先選用 SSD 作為存儲介質，提升數據讀寫性能。
 - 為高負載系統提供足夠的內存（建議至少 64GB RAM）。
 - **清理策略**
 - 定期清理過期數據，確保存儲空間可用性。
-

3. 綜合建議

- 小規模部署：



- 使用單節點模式，簡化維護，適合小型監控需求。
- 大規模部署：
 - 集群模式需要專業配置，建議結合工具（如 Kubernetes）進行自動化運維。

結論

VictoriaMetrics 的靈活配置和高性能特性需要通過合理的運維策略來發揮最大效能。定期備份、性能監控和資源優化是確保系統穩定運行的關鍵。

如需補充具體配置範例或運維腳本，請隨時告知！



參考一文搞懂 VictoriaMetrics 的使用-阳明的博客.pdf，詳細說明簡報的第22頁



簡報第22頁 - 與其他解決方案的比較

根據《一文搞懂 VictoriaMetrics 的使用》，第22頁深入探討了 VictoriaMetrics 與其他時間序列數據存儲解決方案（如 Prometheus、InfluxDB、Thanos）的比較，幫助讀者了解其優勢與應用場景。

VictoriaMetrics vs 其他解決方案

比較

項目	VictoriaMetrics	Prometheus	InfluxDB	Thanos
性能	高效壓縮，查詢性能領先	查詢性能有限	查詢性能佳，但在高基數場景表現較差	查詢性能依賴 Prometheus 的配置



比較項目	VictoriaMetrics	Prometheus	InfluxDB	Thanos
存儲需求	高壓縮比，節省磁碟空間	存儲短期數據	磁碟需求大，存儲成本高	存儲成本高於 VictoriaMetrics
兼容性	完全兼容 Prometheus API 和多協議	僅支持 Prometheus	僅支持其自有協議，兼容性較低	完全兼容 Prometheus API
部署模式	支持單節點與集群模式，靈活部署	僅支持單節點	支持集群，但部署相對複雜	支持集群，但依賴外部存儲
資源需求	內存與 I/O 占用低	內存與 I/O 消耗高	資源需求高，特別是內存和存儲	資源需求較高，需依賴分散式存儲
適用場景	高基數、高頻數據和多租戶架構	小規模監控與即時數據查詢	中型監控場景，長期存儲需求	長期存儲與多集群數據匯聚

VictoriaMetrics 的明顯優勢

1. 性能

- 查詢性能比 InfluxDB 和 Prometheus 高出 20 倍以上。
- 高效數據壓縮技術，大幅降低磁碟使用量（約為原始數據的 1/7）。

2. 兼容性與靈活性

- 完全兼容 Prometheus 的 API，無需修改現有監控系統即可集成。
- 支持多種協議（如 Influx Line、Graphite、OpenTSDB），適應多場景需求。

3. 適合高基數和大規模應用

- 每秒支持數百萬數據點的高基數處理能力，適合大規模監控系統和物聯網應用。

4. 部署靈活

- 單節點模式適合小型部署，集群模式適應高併發和大規模存儲需求。
- 相比 Thanos 和 InfluxDB，集群架構部署相對簡單。



潛在劣勢與局限

1. 功能限制

- 缺乏內建告警功能，需依賴外部工具（如 Alertmanager）。
- 相比 Thanos，無法實現跨地區數據自動同步。

2. 運維要求

- 在集群模式下，部署和管理需要一定技術能力，特別是在處理多組件通信時。
-

選擇建議

選擇 VictoriaMetrics 的情境

1. 高基數數據處理

- 適合每秒數百萬數據點的監控場景。

2. 多租戶支持

- SaaS 平台和跨部門監控需求。

3. 資源有限

- 當存儲和內存成本受限時，VictoriaMetrics 是一個理想的選擇。

選擇其他解決方案的情境

1. **Thanos**：適合需要跨地區數據同步和長期存儲的企業。

2. **Prometheus**：適合中小規模的短期監控。

3. **InfluxDB**：適用於對查詢語法和數據分析需求較高的場景。



結論

VictoriaMetrics 通過高效性能、靈活部署和高兼容性，成為時間序列數據存儲的強大選擇，特別在高基數、大規模場景中具有競爭優勢。

如果需要進一步的比較圖表或具體案例分析，請隨時告知！



參考一文搞懂 VictoriaMetrics 的使用-阳明的博客.pdf，詳細說明簡報的第23頁



簡報第23頁 - 混合使用案例

根據《一文搞懂 VictoriaMetrics 的使用》，第23頁描述了 VictoriaMetrics 與其他系統（如 Prometheus）的混合使用案例，展示其靈活集成能力及應用價值。

VictoriaMetrics 與 Prometheus 混合使用案例

1. 與 Prometheus 集成

- 使用場景：
 - Prometheus 作為短期存儲和即時監控的數據源，而 VictoriaMetrics 用於長期存儲和數據分析。
- 實現方式：
 - 通過 Prometheus 的 `remote_write` 將數據寫入 VictoriaMetrics：

```
remote_write:  
- url: "http://victoria-metrics-host:8428/api/v1/write"
```



- 在需要查詢歷史數據時，通過 VictoriaMetrics 的 API 或 Grafana 展示。
 - **應用優勢：**
 1. **長短期存儲結合：**
 - Prometheus 提供即時性數據查詢，VictoriaMetrics 提供長期存儲與分析支持。
 2. **無縫兼容：**
 - VictoriaMetrics 完全支持 Prometheus 協議，無需改變現有的監控架構。
-

2. 用於多集群監控數據匯聚

- **使用場景：**
 - 在多個地區或環境中部署 Prometheus 實例，通過 VictoriaMetrics 匯聚所有數據進行統一分析。
 - **實現方式：**
 1. 各地 Prometheus 實例配置 `remote_write` 將數據推送到中央的 VictoriaMetrics 節點。
 2. 在中央節點使用 VictoriaMetrics 的查詢功能或 Grafana 進行全局監控。
 - **應用優勢：**
 - 提供跨集群的數據分析與趨勢報告。
 - 減少 Prometheus 實例之間的直接通信壓力。
-

3. 作為 Alertmanager 的數據源

- **使用場景：**



- 將 VictoriaMetrics 作為 Alertmanager 的查詢後端，用於生成告警。
 - **實現方式：**
 - 配置 Alertmanager 通過 PromQL 語法查詢 VictoriaMetrics 提供的 API。
 - **應用優勢：**
 - 支持更大規模的數據源和更長期的告警條件分析。
 - 降低單一 Prometheus 實例在數據存儲上的壓力。
-

4. 與 Grafana 的整合

- **使用場景：**
 - 在 Grafana 中配置 VictoriaMetrics 作為數據源，用於可視化監控數據和生成報表。
- **實現方式：**
 - 在 Grafana 中添加數據源，選擇 Prometheus 類型並指定 VictoriaMetrics 的 API 地址：

```
http://victoria-metrics-host:8428
```

- **應用優勢：**
 1. 支持多種查詢語法（如 PromQL）。
 2. 數據展示直觀，支持動態大屏可視化。
-

混合使用的典型場景與案例

1. 雲原生環境



- VictoriaMetrics 與 Kubernetes 集成，作為 Prometheus 的後端存儲，用於長期監控雲資源。
- 提供 SLA 報告和趨勢分析。

2. 物聯網 (IoT) 平台

- 各 IoT 設備生成數據後，通過 Prometheus 將短期數據存儲在本地，同時將歷史數據匯聚到 VictoriaMetrics。
- 實現全局設備狀態跟蹤和異常預測。

3. 多地區業務監控

- 不同地區的 Prometheus 實例將數據推送到中央的 VictoriaMetrics，支持跨地區業務指標統計。

結論

VictoriaMetrics 在混合架構中的靈活性和兼容性，使其能高效補充現有監控系統的短板，尤其在長期存儲、全局數據匯聚和歷史數據分析方面展現了巨大優勢。

如需補充具體的配置步驟或案例架構圖，請隨時告知！



參考一文搞懂 VictoriaMetrics 的使用-阳明的博客.pdf，詳細說明簡報的第24頁



簡報第24頁 - 總結與建議

根據《一文搞懂 VictoriaMetrics 的使用》，第24頁總結了 VictoriaMetrics 的優勢、適用場景，以及使用時的技術建議。以下是詳細說明：



1. VictoriaMetrics 的核心價值

高效性能

- 卓越的查詢和數據存儲性能，支持高基數、高頻數據場景。
- 在處理大規模數據時，表現出色，尤其適用於需要長期存儲和快速查詢的情境。

低資源需求

- 使用高效壓縮技術，顯著降低磁碟和內存消耗。
- 優化低 IOPS 環境，適合資源受限的場景。

靈活部署

- 單節點和集群模式支持，滿足從中小型到大型監控需求。
- 可根據業務需求靈活擴展，提供高可靠性和可用性。

完全兼容

- 與 Prometheus API 完全兼容，無需對現有監控架構進行重大改動即可集成。
- 支持多種協議，如 Prometheus Remote Write、Influx Line Protocol 和 OpenTSDB。

2. 適用場景

長期存儲需求

- 適合需要保存數月甚至數年的監控數據，如 SLA 報告和歷史數據查詢。

高基數數據場景

- 每秒生成數百萬數據點的環境，如雲原生監控、IoT 平台和大型數據分析。

多租戶架構

- 支持 SaaS 平台和多部門架構，數據隔離且性能穩定。

混合使用場景



- 作為 Prometheus 的後端存儲，提供即時與歷史數據的無縫結合。
-

3. 使用建議

配置建議

1. 單節點模式：
 - 適用於中小規模監控場景，配置簡單，部署快速。
2. 集群模式：
 - 適合大規模、高負載場景，需合理配置 `vmstorage`、`vminsert` 和 `vmselect` 節點。
 - 使用 SSD 提升存儲性能，並設置數據冗餘保證高可用。

運維建議

1. 備份與冗餘：
 - 定期備份存儲數據，避免因硬件故障導致數據丟失。
 - 啟用多副本存儲，提高數據可靠性。
 2. 監控與優化：
 - 使用 Prometheus 監控 VictoriaMetrics 的資源使用（CPU、內存、磁碟 I/O）。
 - 優化查詢參數，例如 ``maxLookback`` 和 ``maxConcurrentInserts``，提升系統性能。
 3. 外部工具集成：
 - 與 Grafana 配合使用，實現高效的數據可視化和報表生成。
 - 配合 Alertmanager 完成告警處理。
-



4. 結論

VictoriaMetrics 是時間序列數據存儲的理想選擇

- 在高性能、低資源需求和靈活性方面表現突出。
- 尤其適合需要處理高基數、大規模監控數據的企業。

選擇建議

- 若需要快速部署並節省資源，選擇單節點模式。
- 在需要多集群數據匯聚和長期存儲的場景下，建議部署集群模式。
- 結合現有的監控架構，混合使用 VictoriaMetrics 和 Prometheus，可發揮其最大效能。

如需補充架構圖或具體配置範例，請隨時告知！



正式-深入探討 Remote Write 與 F...
PDF

把正式-深入探討 Remote Write 與 Federation.pdf簡報，翻成英文，生成pdf檔給我



翻譯完成！您可以下載翻譯後的 PDF 檔案：

[Remote_Write_and_Federation_Detailed_Exploration_Translated.pdf](#) [↗]



把正式-深入探討 Remote Write 與 Federation.pdf簡報，翻成英文的pdf檔給我





翻譯後的英文版 PDF 文件已生成，您可以點擊以下連結下載：

[Remote_Write_and_Federation_Detailed_Exploration_Translated.pdf](#)

若需要進一步修改或調整，請隨時告訴我！



把正式-深入探討 Remote Write 與 Federation.pdf簡報，翻成全英文的pdf檔給我



翻譯完成！您可以下載完整英文版的 PDF 文件：

[Remote_Write_and_Federation_Fully_Translated.pdf](#)

如果需要任何其他調整或補充，請隨時告訴我！ [↗]



把正式-深入探討 Remote Write 與 Federation.pdf簡報，翻成全英文



完整翻譯的英文版 PDF 文件已完成，您可以點擊以下連結下載：

[Remote_Write_and_Federation_Fully_Translated.pdf](#)

如果有需要進一步修改的內容或具體翻譯需求，請隨時告訴我！



把正式-深入探討 Remote Write 與 Federation.pdf內容，翻成全英文



翻譯後的完整內容已生成，請點擊以下連結下載翻譯成英文的 PDF 文件：

[Remote_Write_and_Federation_Fully_Translated.pdf](#)

如需查看具體翻譯內容或進一步調整，請隨時告訴我！



正式-深入探討 **Remote Write** 與 **F...**
PDF

依據簡報的每一頁的內容翻成全英文



簡報內容的每一頁翻譯如下：

Slide 1: Prometheus & VictoriaMetrics Intro

Introduction to Prometheus and VictoriaMetrics, focusing on their use cases and functionalities.

Slide 2: Introduction

- Prometheus Remote Write & Federation
 - Overview
 - Prometheus Remote Write
 - Prometheus Federation
 - Comparison of Remote Write and Federation



- Conclusion
 - VictoriaMetrics
 - Introduction
 - Why Choose VictoriaMetrics?
 - Key Features
 - Single Node Deployment
 - Cluster Architecture
 - Deployment Steps and Configuration
 - Advantages and Limitations
 - Use Cases
 - Configuration and Maintenance Recommendations
 - Comparison with Other Solutions
 - Hybrid Use Cases
 - Summary and Recommendations
-

Slide 3: (Empty)

Transition slide.

Slide 4: Overview

- Prometheus is an open-source monitoring system offering efficient time-series data storage and querying capabilities.
- Remote Write and Federation have unique characteristics tailored to different needs.



Slide 5: Prometheus Remote Write

- **Long-term storage:** Remote Write enables persistent data storage in external systems, suitable for long-term retention.
- **Multi-cluster data aggregation:** Centralized storage unifies data from multiple Prometheus instances for easier analysis.
- **Disaster recovery:** Supports data synchronization to remote storage for offsite backups.

Slide 6: Advantages and Disadvantages of Prometheus Remote Write

- **Advantages:**
 - Long-term storage.
 - Centralized analysis with aggregated data.
 - Integration with external tools like Grafana.
- **Disadvantages:**
 - Reduced immediacy due to latency.
 - Dependency on external storage systems.
 - Prometheus cannot directly query remote storage.

Slide 7: Prometheus Federation

- Prometheus Federation allows one instance (upper-layer) to scrape data from another (lower-layer).



- Supports hierarchical monitoring.
- **Advantages:**
 - Built-in support.
 - High immediacy.
 - Flexible data filtering.
- **Disadvantages:**
 - Limited to short-term data.
 - Performance pressure on lower-layer instances.

Slide 8: Prometheus Federation (continued)

Explains key benefits and use cases for Prometheus Federation.

Slide 9: Comparison of Remote Write and Federation

Feature	Remote Write	Federation
Purpose	Long-term storage, analysis	Hierarchical monitoring
Data Direction	Push	Pull
Immediacy	Latency-dependent	High
Dependencies	External storage required	Fully built-in
Use Case	Large-scale storage	Real-time hierarchical sharing



Slide 10: Conclusion

- **Recommendations:**
 - Remote Write for long-term storage.
 - Federation for real-time monitoring and integration.
 - **Performance and Cost Trade-offs:** Balance based on system needs and available resources.
-

Slide 11: (Empty)

Transition slide.

Slide 12: VictoriaMetrics Overview

- High-performance time-series database optimized for monitoring data storage and retrieval.
 - Supports both single-node and cluster deployments.
-

Slide 13: Why Choose VictoriaMetrics

- **Efficiency:** High compression ratios reduce storage requirements.
 - **Speed:** Optimized for fast query responses.
 - **Multi-tenancy:** Native support for isolated data management.
-



Slide 14: Key Features

- Fully compatible with Prometheus API.
 - Flexible deployment options for scalability.
 - Supports both single-node and cluster deployments.
-

Slide 15: Single Node Deployment

- Quick and simple setup for small-scale environments with fewer than 1 million data points per second.
-

Slide 16: Cluster Architecture

- **Components:**
 - ``vmstorage``: Stores raw data.
 - ``vminsert``: Handles data ingestion.
 - ``vmselect``: Processes query requests.
-

Slide 17: Deployment Steps and Configuration

- Options include Helm for automated deployments or YAML for manual configuration.
 - Tailored for small or large-scale setups.
-



Slide 18: Advantages and Limitations

- **Strengths:**
 - Handles high cardinality.
 - Low resource consumption.
 - **Limitations:**
 - Relies on sufficient infrastructure for storage scalability.
-

Slide 19: Use Cases

- Suitable for large-scale monitoring, SLA reporting, and multi-tenant architectures.
-

Slide 20: Configuration and Maintenance Recommendations

- Use Grafana dashboards for monitoring performance.
 - Regularly back up data to prevent loss.
-

Slide 21: Comparison with Other Solutions

- **Storage efficiency:** Outperforms Prometheus and InfluxDB.
 - **Query performance:** Excels in large-scale data scenarios.
-



Slide 22: Hybrid Use Cases

- Combines VictoriaMetrics with Prometheus for optimized storage and analysis.

Slide 23: Summary and Recommendations

- **Efficiency:** VictoriaMetrics excels in high-frequency, large-scale time-series data management.
- **Integration:** Fully compatible with Prometheus for seamless use.

若需要生成這份內容的英文版 PDF，請隨時告訴我！



正式-深入探討 **Remote Write** 與 **F...**
PDF

依據簡報的每一頁的內容翻成全英文



已完整翻譯簡報的每一頁內容，以下是各頁翻譯的英文內容：

Slide 1: Prometheus & VictoriaMetrics Intro

Slide 2: Introduction



- Prometheus Remote Write & Federation
 - Overview
 - Prometheus Remote Write
 - Prometheus Federation
 - Comparison of Remote Write and Federation
 - Conclusion
- VictoriaMetrics
 - Introduction
 - Why Choose VictoriaMetrics?
 - Key Features
 - Single Node Deployment
 - Cluster Architecture
 - Deployment Steps and Configuration
 - Advantages and Limitations
 - Use Cases
 - Configuration and Maintenance Recommendations
 - Comparison with Other Solutions
 - Hybrid Use Cases
 - Summary and Recommendations

Slide 3: (Empty)

Slide 4: Overview



- Prometheus is an open-source monitoring system offering efficient time-series data storage and querying capabilities.
 - Remote Write and Federation serve distinct purposes for data management.
-

Slide 5: Prometheus Remote Write

- **Long-term storage:** Enables data persistence to external systems for prolonged retention.
 - **Multi-cluster aggregation:** Unifies data from multiple Prometheus instances into centralized storage.
 - **Disaster recovery:** Synchronizes data to remote systems for backups.
-

Slide 6: Advantages and Disadvantages of Remote Write

- **Advantages:**
 - Long-term storage.
 - Centralized analysis across clusters.
 - Integration with external tools like Grafana.
 - **Disadvantages:**
 - Reduced immediacy due to network latency.
 - Dependency on external systems.
 - Cannot directly query remote storage.
-

Slide 7: Prometheus Federation



- A mechanism where an upper-layer Prometheus instance scrapes data from lower-layer instances via HTTP.
 - Ideal for hierarchical monitoring structures.
-

Slide 8: Prometheus Federation Advantages and Disadvantages

- **Advantages:**
 - Built-in support.
 - High immediacy.
 - Data filtering capabilities.
 - **Disadvantages:**
 - Limited to short-term data.
 - Increased load on lower-layer instances.
-

Slide 9: Comparison of Remote Write and Federation

Feature	Remote Write	Federation
Purpose	Long-term storage, analysis	Hierarchical monitoring
Data Direction	Push	Pull
Immediacy	Latency-dependent	High
Dependencies	External storage required	Fully built-in
Filtering	Via relabel configuration	Via match[] parameter



Slide 10: Conclusion

- **Recommendations:** Use Remote Write for long-term storage, Federation for hierarchical integration.
 - **Considerations:** Evaluate storage needs against system performance and resource availability.
-

Slide 12: VictoriaMetrics Overview

- VictoriaMetrics is a high-performance time-series database designed for efficient storage and querying.
-

Slide 13: Why Choose VictoriaMetrics

- **Efficiency:** High compression ratios.
 - **Speed:** Optimized query responses.
 - **Multi-tenancy:** Isolated data management for teams.
-

Slide 14: Key Features

- Fully compatible with Prometheus API.
 - Flexible deployment options.
 - Supports both single-node and cluster setups.
-

Slide 15: Single Node Deployment



- Suitable for environments with fewer than 1 million data points per second.
-

Slide 16: Cluster Architecture

- **Components:**
 - ``vmstorage``: Data storage.
 - ``vminsert``: Handles ingestion.
 - ``vmselect``: Processes queries.
-

Slide 17: Deployment Steps

- Helm simplifies deployment; manual YAML configuration offers flexibility.
-

Slide 18: Advantages and Limitations

- **Advantages:** Handles high cardinality and consumes fewer resources.
 - **Limitations:** Infrastructure dependency.
-

Slide 19: Use Cases

- Ideal for large-scale monitoring, SLA reporting, and multi-tenant setups.
-



Slide 20: Maintenance Recommendations

- Regularly monitor and back up data.
 - Use Grafana for visualization.
-

Slide 21: Comparison with Other Solutions

- Superior compression and query performance compared to alternatives.
-

Slide 22: Hybrid Use Cases

- Combines with Prometheus for optimized data storage and analysis.
-

Slide 23: Summary

- Efficient and scalable, suitable for high-frequency, large-scale monitoring.
-

如需將此內容生成英文 PDF，請告知！