# OpenWrt 執行流程與互動之動態視覺化：嵌入式系統可觀測性的深度研究報告

## 執行摘要

本研究報告針對「這些圖表可以動畫化嗎？特別是針對 OpenWrt 的執行流程與互動」這一核心命題進行了詳盡的技術分析與方法論證。傳統嵌入式 Linux 開發領域中，系統架構與執行流程的文檔化長期依賴於靜態工件，如 UML 序列圖（Sequence Diagrams）、呼叫圖（Call Graphs）與靜態火焰圖（Flame Graphs）。然而，OpenWrt 作為一個高度異步、事件驅動（Event-Driven）的嵌入式作業系統，其核心運作依賴於 ubus（OpenWrt Micro Bus）、procd（行程管理守護程式）與 netifd（網路介面守護程式）之間的複雜互動。靜態圖表在捕捉競爭條件（Race Conditions）、啟動風暴（Boot Storms）以及延遲傳播（Latency Propagation）等時序敏感現象時，存在本質上的局限性。

本報告證實，OpenWrt 的執行流程與互動**完全具備動畫化的可行性**，且透過動態視覺化能揭示靜態分析無法察覺的系統行為模式。透過整合 ubus monitor、perf、strace 以及現代化的 eBPF 技術作為資料獲取層，並將其轉換為 Chrome Trace Event Format、Gource Custom Log 或 FlameScope Heatmap 等中間格式，工程師可以利用 **Perfetto**、**Gource**、**FlameScope** 及 **Cytoscape.js** 等先進引擎，重建出具備時間維度（Temporal Dimension）的動態敘事。

本報告將深入探討從資料擷取、格式轉換到渲染呈現的完整技術堆疊，並針對 OpenWrt 特有的硬體限制（如 MIPS 架構對 eBPF 的支援度）提出具體的實施策略與架構建議。

## 第一章：嵌入式架構動態視覺化的理論框架

### 1.1 從靜態快照到動態敘事的典範轉移

在軟體工程的歷史脈絡中，視覺化一直被視為理解複雜系統的關鍵手段。然而，對於像 OpenWrt 這樣的嵌入式 Linux 發行版，傳統的靜態圖表往往只能描繪系統在「理想狀態」下的結構，而非「執行狀態」下的行為。一張標準的 DHCP 協定序列圖可能完美地展示了 DISCOVER、OFFER、REQUEST、ACK 的線性流程，但在真實的路由器運行環境中，這個過程可能與 hostapd 的無線用戶端關聯事件、firewall 的規則重載以及 hotplug 腳本的觸發交織在一起。

動態視覺化（Animated Visualization）引入了時間變數 ，將系統狀態描述函數從  擴展為 。這種轉變對於 OpenWrt 至關重要，原因如下：

* **時序密度（Temporal Density）的揭示**：靜態圖表無法區分兩個事件是發生在 1 毫秒內還是 1 秒內。在 OpenWrt 啟動過程中，數百個服務在數秒內並發啟動，動態視覺化能夠透過時間軸的縮放與重播，讓開發者直觀地看到 CPU 負載的瞬間峰值與服務啟動的擁塞點 1。
* **因果關係（Causality）的驗證**：在非同步訊息傳遞架構中，請求與回應往往是非阻塞的。透過動態流向線（Flow Arrows），視覺化工具可以明確畫出「netifd 發出請求」與「odhcp6c 回傳數據」之間的因果連結，排除並發事件的干擾 3。
* **系統脈衝（System Pulse）的感知**：透過有機的動畫形式（如 Gource），開發者可以「感覺」到系統的活躍程度。例如，當網路介面重啟時，視覺化圖形中應呈現出以 network 子系統為中心的劇烈活動爆發，隨後逐漸平息。這種直觀感知對於快速診斷異常行為（如無限重啟迴圈）具有不可替代的價值 4。

### 1.2 OpenWrt 的異步架構特性

要實現 OpenWrt 的動畫化，首先必須解構其特殊的軟體架構。OpenWrt 與採用 systemd 與 D-Bus 的通用 Linux 發行版不同，它採用了一套極簡、高效的專屬元件：

* **ubus (OpenWrt Micro Bus)**：這是 OpenWrt 的中樞神經系統。它基於 Unix Domain Socket 實現了一種輕量級的 JSON-RPC 機制。幾乎所有的核心守護程式（Daemons）都掛載於此匯流排上。動畫化 ubus 的流量，等同於繪製出了 OpenWrt 的思維過程 6。
* **netifd (Network Interface Daemon)**：負責管理網路介面的狀態機。它的狀態轉換（UP/DOWN/PENDING）是網路設備運作的核心。
* **procd (Process Manager)**：作為 init 程序（PID 1），它負責系統初始化與服務生命週期管理。

這些組件之間的互動是高度動態的。例如，一個物理網路線的插入會觸發內核的 Netlink 事件，被 procd 的熱插拔處理器捕獲，進而透過 ubus 通知 netifd，netifd 再調用 odhcpc 獲取 IP，最後觸發 firewall 重載規則。這一連串的連鎖反應，唯有透過動態回放才能完整呈現。

## 第二章：資料獲取層：捕捉系統行為的「幀」

動畫的基礎是數據。為了產生流暢、準確的執行流程動畫，我們需要從 OpenWrt 系統中擷取具備高精度時間戳記（Timestamped）的事件流。根據資料的抽象層級，我們可以將獲取策略分為應用層、系統層與內核層。

### 2.1 應用層遙測：ubus monitor 的深度解析

對於 OpenWrt 的服務互動而言，ubus monitor 是最直接且豐富的資料來源。該指令能夠監聽 ubus 守護程式上的多播群組，並即時輸出所有經過匯流排的訊息。

#### 2.1.1 資料結構與語義

ubus monitor 輸出的原始資料流包含了建構序列圖所需的所有關鍵要素：

* **時間（Time）：** 事件發生的精確時刻。
* **發起者（Peer）：** 發送訊息的行程或物件 ID。
* **路徑（Path）：** 訊息的目標物件（例如 network.interface.wan）。
* **方法（Method）：** 呼叫的函數名稱（例如 up, down, status）。
* **負載（Payload）：** 具體的參數或回傳值（通常為 JSON 格式）。

一個典型的 ubus monitor 輸出一行可能如下所示：

-> [10:22:33.123] 8e00000a network.interface notify { "interface": "wan", "action": "up" }

這些數據點直接對應於動畫中的「關鍵幀」。-> 表示訊息的方向，時間戳記決定了動畫的節奏，8e00000a 是 Peer ID，可以用來區分不同的執行緒或服務泳道（Swimlanes）7。

#### 2.1.2 擷取策略的優化

為了支援高品質的動畫重建，單純執行 ubus monitor 可能不足。我們建議採用以下優化策略：

1. **時間同步化**：OpenWrt 的 ubus 工具輸出的時間戳記可能不包含日期，或僅精確到秒。建議修改 ubus 原始碼或使用包裝腳本（Wrapper Script），在每一行輸出前加上 Unix Epoch 微秒級時間戳記，以便後續與內核日誌（dmesg）進行對齊 6。
2. **過濾與降噪**：在生產環境中，ubus 可能充斥著定期的輪詢（Polling）請求（如 LuCI 介面的自動更新）。在擷取數據用於動畫製作前，應透過 grep 或 jq 排除這些背景雜訊，以免動畫過於混亂。

### 2.2 系統層遙測：strace 與程序軌跡

並非所有 OpenWrt 的行為都會經過 ubus。許多的 shell 腳本（如 /lib/netifd/proto/dhcp.sh）會直接執行二進位檔或讀寫檔案系統。為了捕捉這部分「隱形」的互動，我們需要 strace。

* **捕捉目標**：重點關注 execve（執行新程序）、open/write（檔案操作）以及 socket（網路通訊）系統呼叫。
* **視覺化價值**：透過 strace 捕捉到的 execve 呼叫鏈，可以繪製出詳細的程序樹（Process Tree）生長動畫。這對於理解 OpenWrt 複雜的腳本呼叫層級（Script Nesting）至關重要 4。

### 2.3 內核層遙測：perf 與 eBPF

對於效能瓶頸的視覺化，必須深入內核。

* **perf (Performance Counters for Linux)**：OpenWrt 提供了 perf 套件。透過 perf record -a -g 指令，可以以高頻率（如 99Hz）採樣 CPU 的指令指標暫存器（Instruction Pointer）與堆疊軌跡（Stack Trace）。這些數據是產生 **FlameScope** 熱圖的基礎，能夠將 CPU 的使用情況在時間軸上展開 2。
* **eBPF (Extended Berkeley Packet Filter)**：這是現代 Linux 可觀測性的聖杯。eBPF 允許在內核中動態掛載探針（Probes），以極低的開銷捕捉網路封包流向、檔案系統延遲等深層資訊。
  + *架構限制警語*：OpenWrt 常見於 MIPS 架構（如 Atheros AR71xx/AR93xx）或較舊的 ARM 架構。這些平台往往不支援 eBPF JIT 編譯器，或者內核版本過舊（< 4.9），無法運行現代 eBPF 工具（如 Cilium Hubble）。因此，eBPF 的應用主要侷限於較新的 ARM64（如 Raspberry Pi 4, MediaTek Filogic）或 x86 軟路由平台 10。

## 第三章：資料轉換與標準化：通往視覺化的橋樑

原始日誌是無法直接動畫化的。我們需要一個中間層，將 OpenWrt 的異質數據轉換為視覺化引擎能夠理解的標準格式。

### 3.1 目標格式 I：Chrome Trace Event Format (JSON)

這是目前最通用、支援度最高的時序視覺化格式。它由 Google 開發，最初用於 Chrome 瀏覽器的效能分析，現已被 Perfetto 等工具廣泛採用。

* **格式結構**：  
  JSON  
  {  
   "traceEvents":  
  }
* **轉換邏輯**：
  + 將 ubus monitor 的每一條請求開始（Call）映射為 ph: "B"（Begin）事件。
  + 將回應或完成映射為 ph: "E"（End）事件。
  + 將 ubus 的物件名稱（如 network.device）映射為 pid（Process ID）或 tid（Thread ID），以便在時間軸上分層顯示 3。

### 3.2 目標格式 II：Gource Custom Log Format

Gource 原本用於版本控制系統的視覺化，但其靈活的自定義日誌格式使其成為展示系統活動的絕佳工具。

* **格式結構**：timestamp|username|type|filename
* **OpenWrt 映射策略**：
  + **Timestamp**：Unix 時間戳記。
  + **Username**：映射為觸發事件的行程名稱（如 procd, dnsmasq）。
  + **Type**：映射為操作類型（A=新增服務/介面, M=修改配置/呼叫方法, D=刪除服務/介面）。
  + **Filename**：映射為 ubus 的層次路徑（如 network/interface/wan）4。

### 3.3 目標格式 III：FlameScope Profile

FlameScope 需要特定格式的堆疊折疊（Folded Stacks）數據，通常包含時間戳記偏移量。這需要使用 Brendan Gregg 提供的 stackcollapse-perf.pl 等腳本對 perf script 的輸出進行後處理 1。

## 第四章：實施策略 I — 時序與序列動畫 (Perfetto)

針對 OpenWrt 執行流程（Execution Flow）的動畫化，**Perfetto** 是目前最強大的解決方案。它不僅能呈現時間軸，還能透過「流（Flow）」的概念視覺化組件間的因果互動。

### 4.1 為什麼選擇 Perfetto？

* **微秒級縮放**：OpenWrt 的許多互動發生在微秒級別（如中斷處理）。Perfetto 的 UI 支援無限平滑縮放，這是傳統影片格式無法比擬的。
* **互動式查詢**：內建 SQL 查詢引擎，允許開發者在視覺化之上進行數據分析（例如：「統計啟動過程中 netifd 阻塞超過 100ms 的次數」）3。
* **跨平台支援**：基於 Web 的 UI (ui.perfetto.dev) 意味著無需在開發機上安裝複雜軟體即可分析從路由器導出的 Trace 檔案。

### 4.2 實作流程：從 ubus 到 Perfetto

要將 OpenWrt 的 ubus 互動轉化為 Perfetto 動畫，需開發一個轉換器（Converter）：

1. **擷取**：在 OpenWrt 裝置上執行 ubus monitor > /tmp/ubus.trace。
2. **後處理**：在 PC 端執行 Python 腳本讀取該檔案。
3. **事件合成**：
   * **Async Slices (非同步切片)**：ubus 呼叫通常是異步的。轉換器需要維護一個「未完成請求表」，當讀到 call 時記錄開始時間，讀到回傳時計算持續時間並輸出一個完整的 Async Slice。這在視覺上會形成一個橫跨時間軸的長條，直觀顯示請求的生命週期。
   * **Flow Events (流事件)**：這是動畫化的核心。當 netifd 發送一個通知觸發 dnsmasq 重載時，我們在 Trace 中插入一個 Flow 事件（ph: "s" start, ph: "t" step, ph: "f" finish）。在 Perfetto UI 中，這會被渲染為一條連接兩個不同行程軌道的貝茲曲線箭頭，生動地展示出「A 觸發了 B」的因果流動 3。

### 4.3 案例分析：DHCP 續約流程

在 Perfetto 中，一個 DHCP 續約的動畫將呈現為：

1. **時間點 T0**：odhcpc 軌道出現一個事件區塊，表示收到核心封包。
2. **T0+1ms**：一條流動箭頭從 odhcpc 射向 netifd 軌道（ubus call）。
3. **T0+2ms**：netifd 軌道出現處理區塊，隨後發射多條箭頭指向 firewall 和 dnsmasq（觸發配置更新）。
4. **T0+50ms**：firewall 軌道顯示密集的 CPU 活動（重載 iptables 規則）。

這種視覺化方式將抽象的日誌轉變為可互動的、具備因果關係的動態劇本。

## 第五章：實施策略 II — 拓撲與有機動畫 (Gource & Cytoscape)

除了線性的時間軸，系統的「結構」如何隨時間演變也是重要的視覺化維度。這類動畫強調的是**拓撲結構（Topology）與群體行為（Swarm Behavior）**。

### 5.1 Gource：軟體執行的有機生態系

Gource 通常被用來展示程式碼庫的演進，但其本質是一個通用的樹狀結構視覺化引擎。

* **OpenWrt 的樹狀映射**：
  + 根節點：OpenWrt System
  + 第一層分支：Network, System, Service, Kernel
  + 葉節點：具體的介面（wan, lan）、設定檔（/etc/config/wireless）或執行緒。
* **動態演繹**：  
  透過將 ubus 和 fschange（檔案系統變更）日誌轉換為 Gource 格式，我們可以看到一個生動的動畫：
  + 系統啟動時，代表 procd 的「使用者」飛入畫面，快速「射擊」生成大量的服務節點（樹枝生長）。
  + 當發生網路故障時，代表 netifd 的使用者會頻繁地對 network/interface 分支進行操作（發光束），如果介面當機，該分支可能會枯萎消失（Delete 事件），隨後在重連時重新生長。
* **價值**：這種動畫極具觀賞性與宏觀洞察力，非常適合向非技術利害關係人展示系統的複雜度與即時狀態 4。

### 5.2 Cytoscape.js：力導向服務相依圖

對於需要更嚴謹的拓撲分析，Cytoscape.js 提供了強大的 Web 圖論視覺化能力。

* **力導向佈局（Force-Directed Layout）**：利用 cose 或 cola 佈局演算法，可以模擬節點間的物理斥力與引力。
* **應用場景**：
  + 建立一個網頁儀表板，即時讀取 ubus list。
  + 將每個 ubus 物件渲染為圓點。
  + 當物件 A 呼叫物件 B 時，在兩者間繪製一條動態連線（例如：流動的粒子效果）。
  + **動畫效果**：隨著系統運作，相關聯的服務會自動聚攏（Clustering），而孤立的服務會被推向邊緣。這種動態佈局能揭示出系統中的「上帝物件」（God Objects，即與眾多服務互動的核心組件）16。

## 第六章：實施策略 III — 效能與熱圖動畫 (FlameScope)

在嵌入式系統中，單純的執行流程動畫可能掩蓋了效能問題。CPU 的瞬間凍結（Stutter）在標準圖表中往往被平均化而消失。

### 6.1 FlameScope 的時序熱圖

FlameScope 結合了「亞秒級熱圖（Sub-second Heatmap）」與「火焰圖」。

* **視覺化機制**：
  + X 軸為時間（秒），Y 軸為該秒內的毫秒偏移量。
  + 顏色的深淺代表 CPU 的忙碌程度。
* **OpenWrt 應用**：
  + 透過 perf 錄製一段時間的系統行為。
  + 在 FlameScope 中，週期性的背景任務（如 wifi 掃描或 collectd 統計）會呈現為熱圖上規則的垂直條紋。
  + **互動動畫**：使用者可以選取熱圖上的任意一個「亮點」（高負載區間），系統會即時生成該時間切片的火焰圖。這讓開發者能從「宏觀的時間分佈」瞬間鑽取到「微觀的堆疊呼叫」，是診斷間歇性延遲的神器 1。

## 第七章：前沿技術 — eBPF 與進階可觀測性

隨著硬體效能的提升，OpenWrt 正逐漸擁抱 eBPF 技術，這將徹底改變執行流程的視覺化方式。

### 7.1 Cilium Hubble 的啟示

Cilium Hubble 是 Kubernetes 生態中的 eBPF 網路視覺化工具，它能自動繪製服務間的通訊地圖（Service Map）。雖然 OpenWrt 通常不運行 Kubernetes，但 Hubble 的底層 eBPF 探針（Probes）原理完全適用於 Linux 系統。

* **Standalone 模式**：透過編譯支援 eBPF 的 OpenWrt 內核（Kernel 5.10+），並移植 hubble-relay 與 gops 等組件，可以在 OpenWrt 上實現類似 Hubble 的即時服務依賴圖。這將自動顯示 uhttpd 與 php-fpm 之間的 TCP 連線，甚至是 dnsmasq 的 DNS 解析請求，並以動態拓撲圖呈現 20。

### 7.2 嵌入式系統的限制與對策

在 MIPS 或低階 ARM 設備上，eBPF 的 JIT 編譯可能不被支援，導致效能開銷過大。在此情境下，建議回退使用 ftrace 或 perf 的靜態追蹤點（Tracepoints），雖然靈活性不如 eBPF，但在視覺化標準的內核行為（如排程器切換、封包傳輸）時仍能提供足夠的數據密度 10。

## 第八章：綜合分析與建議

### 8.1 工具選擇矩陣

為了協助開發團隊選擇最適合的視覺化方案，本報告整理了以下矩陣：

| **視覺化目標** | **推薦工具** | **資料來源** | **動畫能力** | **OpenWrt 適用性** |
| --- | --- | --- | --- | --- |
| **執行流程除錯** | **Perfetto** | ubus monitor, JSON | **極高** (時間軸回放, 因果流) | **最佳** (適合深度工程分析) |
| **系統活動概覽** | **Gource** | ubus, strace | **高** (自動播放影片) | **良** (適合展示與簡報) |
| **效能瓶頸分析** | **FlameScope** | perf record | **中** (互動式熱圖選取) | **極佳** (診斷間歇性問題) |
| **服務拓撲監控** | **Cytoscape.js** | ubus list | **高** (力導向模擬) | **中** (需自行開發 Web 前端) |
| **即時網路地圖** | **eBPF (Hubble)** | Kernel Probes | **高** (即時流量動畫) | **低** (僅限高階硬體) |
| **代碼依賴分析** | **AppMap** | Language Agents | **無** (目前不支援 C++) | **差** (不適用於 OpenWrt 核心) |

### 8.2 結論與展望

回到最初的問題：「這些圖表可以動畫化嗎？特別是針對 OpenWrt 的執行流程與互動？」答案是肯定的，但這並非單一工具可以完成的任務。它需要一個工具鏈的整合：

1. **資料擷取**：利用 ubus monitor 與 perf 作為底層感測器。
2. **格式轉換**：透過腳本將原始日誌標準化為 Chrome Trace Event Format 或 Gource Log。
3. **渲染呈現**：利用 **Perfetto** 進行微觀的工程分析，或利用 **Gource** 進行宏觀的架構展示。

這種動態視覺化方法，將使 OpenWrt 開發者從「閱讀靜態日誌」進化為「觀看系統運作」，從而更有效地捕捉稍縱即逝的競爭條件與效能異常。隨著 eBPF 在嵌入式領域的普及，未來的 OpenWrt 路由器將有望內建即時的、零開銷的 3D 執行流程儀表板，徹底改變嵌入式系統的開發與維運體驗。

*本報告基於對現有開源工具、OpenWrt 架構文件及效能分析最佳實踐的綜合研究。*

**參考資料索引：**

* **Perfetto & Trace Formats:** 3
* **OpenWrt Architecture (ubus/netifd):** 7
* **Gource Visualization:** 4
* **FlameScope & Profiling:** 1
* **eBPF & Embedded Limitations:** 10
* **Graph Visualization (D3/Cytoscape):** 16
* **AppMap Limitations:** 26

#### 引用的著作

1. Flame Scope - Brendan Gregg, 檢索日期：2月 7, 2026， <https://www.brendangregg.com/flamescope.html>
2. FlameScope is a visualization tool for exploring different time ranges as Flame Graphs. - GitHub, 檢索日期：2月 7, 2026， <https://github.com/Netflix/flamescope>
3. All my favorite tracing tools: eBPF, QEMU, Perfetto, new ones I built and more - Tristan Hume, 檢索日期：2月 7, 2026， <https://thume.ca/2023/12/02/tracing-methods/>
4. Gource custom log format - Google Code, 檢索日期：2月 7, 2026， <https://code.google.com/archive/p/gource/wikis/CustomLogFormat.wiki>
5. How to use Gource to visualize your code repos (and more) | Opensource.com, 檢索日期：2月 7, 2026， <https://opensource.com/business/16/8/intro-to-gource>
6. Getting started with OpenWrt Micro Bus - ubus - Zilogic Systems, 檢索日期：2月 7, 2026， <https://zilogic.com/blog/ubus-service-development-with-rpcd.html>
7. [OpenWrt Wiki] ubus (OpenWrt micro bus architecture), 檢索日期：2月 7, 2026， <https://openwrt.org/docs/techref/ubus>
8. OpenWRT UBUS - HackMD, 檢索日期：2月 7, 2026， <https://hackmd.io/@rYMqzC-9Rxy0Isn3zClURg/H1BY98bRw>
9. Linux perf Examples - Brendan Gregg, 檢索日期：2月 7, 2026， <https://www.brendangregg.com/perf.html>
10. An eBPF overview, part 4: Working with embedded systems - Collabora, 檢索日期：2月 7, 2026， <https://www.collabora.com/news-and-blog/blog/2019/05/06/an-ebpf-overview-part-4-working-with-embedded-systems/>
11. einat-ebpf/docs/guide/openwrt.md at main - GitHub, 檢索日期：2月 7, 2026， <https://github.com/EHfive/einat-ebpf/blob/main/docs/guide/openwrt.md>
12. Real-World Use Cases of eBPF (And Why You Should Care) | by Samiksha Khadka, 檢索日期：2月 7, 2026， <https://medium.com/@swabhimankhadka2001/real-world-use-cases-of-ebpf-and-why-you-should-care-bbd813ff5e81>
13. Trace Event Format - Google Docs, 檢索日期：2月 7, 2026， <https://docs.google.com/document/d/1CvAClvFfyA5R-PhYUmn5OOQtYMH4h6I0nSsKchNAySU/preview>
14. Track events (Tracing SDK) - Perfetto Tracing Docs, 檢索日期：2月 7, 2026， <https://perfetto.dev/docs/instrumentation/track-events>
15. Chrome trace startup got an unreadable json file - Google Groups, 檢索日期：2月 7, 2026， <https://groups.google.com/a/chromium.org/g/chromium-dev/c/LkvpZMbys34>
16. The Best Libraries and Methods to Render Large Force-Directed Graphs on the Web, 檢索日期：2月 7, 2026， <https://weber-stephen.medium.com/the-best-libraries-and-methods-to-render-large-network-graphs-on-the-web-d122ece2f4dc>
17. How can I configure a continuous layout simulation in Cytoscape force-directed graphs?, 檢索日期：2月 7, 2026， <https://stackoverflow.com/questions/60151940/how-can-i-configure-a-continuous-layout-simulation-in-cytoscape-force-directed-g>
18. Using layouts - Cytoscape.js, 檢索日期：2月 7, 2026， <https://blog.js.cytoscape.org/2020/05/11/layouts/>
19. eBPF by Example: Building a GPU Flamegraph Profiler with CUPTI - eunomia-bpf, 檢索日期：2月 7, 2026， <https://eunomia.dev/tutorials/xpu/flamegraph/>
20. Introduction to Cilium & Hubble, 檢索日期：2月 7, 2026， <https://docs.cilium.io/en/stable/overview/intro.html>
21. eBPF-Based Network Observability: Exploring Cilium Hubble and Alternatives - CloudRaft, 檢索日期：2月 7, 2026， <https://www.cloudraft.io/blog/ebpf-based-network-observability-using-cilium-hubble>
22. A script for converting uftrace data to perfetto trace format · Issue #2021 - GitHub, 檢索日期：2月 7, 2026， <https://github.com/namhyung/uftrace/issues/2021>
23. Visualizing external trace formats with Perfetto, 檢索日期：2月 7, 2026， <https://perfetto.dev/docs/getting-started/other-formats>
24. How to do animation of sequential data with d3.js? - Stack Overflow, 檢索日期：2月 7, 2026， <https://stackoverflow.com/questions/45258853/how-to-do-animation-of-sequential-data-with-d3-js>
25. Dynamic message sequence charts for d3. - GitHub, 檢索日期：2月 7, 2026， <https://github.com/koudelka/d3-message-sequence>
26. Troubleshooting - AppMap, 檢索日期：2月 7, 2026， <https://appmap.io/docs/troubleshooting/>
27. AppMap | AppMap, 檢索日期：2月 7, 2026， <https://appmap.io/>
28. Visualization in your code editor - AppMap, 檢索日期：2月 7, 2026， <https://appmap.io/product/appmap-in-the-code-editor.html>