**So sánh VictoriaMetrics và Prometheus cho cơ sở dữ liệu chuỗi thời gian giám sát hệ thống**

**Giới thiệu**

Trong lĩnh vực giám sát hệ thống, Prometheus từ lâu đã trở thành tiêu chuẩn phổ biến cho việc thu thập và lưu trữ dữ liệu dạng chuỗi thời gian (time-series)[influxdata.com](https://www.influxdata.com/comparison/prometheus-vs-victoria/#:~:text=Prometheus%20is%20an%20open,emphasis%20on%20reliability%20and%20scalability). Tuy nhiên, với sự phát triển của quy mô hạ tầng và nhu cầu lưu trữ dài hạn, các giải pháp mới như VictoriaMetrics (VM) đã xuất hiện nhằm khắc phục một số hạn chế của Prometheus về hiệu năng và khả năng mở rộng. Bài viết này trình bày một nghiên cứu chuyên sâu so sánh Prometheus và VictoriaMetrics dưới vai trò hệ thống cơ sở dữ liệu chuỗi thời gian cho giám sát. Các khía cạnh được phân tích bao gồm kiến trúc tổng thể, hiệu suất ghi/truy vấn, khả năng mở rộng, chi phí phần cứng/vận hành, tính dễ triển khai và bảo trì, khả năng tương thích hệ sinh thái, kinh nghiệm triển khai thực tế, và tổng kết ưu nhược điểm của từng giải pháp. Các nhận định trong bài được củng cố bằng dẫn chứng từ tài liệu chính thức và các kết quả benchmark uy tín.

**1. Kiến trúc tổng thể và cách triển khai**

**Kiến trúc Prometheus:**

*Hình 1: Sơ đồ kiến trúc Prometheus (một server đơn lẻ thực hiện thu thập dữ liệu theo mô hình kéo, lưu trữ chuỗi thời gian trong TSDB nội bộ, tích hợp với Pushgateway cho các job ngắn hạn, Alertmanager để cảnh báo, Grafana cho trực quan hóa, v.v.)*. Prometheus có kiến trúc **single-server** (đơn thể) tự quản lý toàn bộ quy trình thu thập, lưu trữ và truy vấn dữ liệu[influxdata.com](https://www.influxdata.com/comparison/prometheus-vs-victoria/#:~:text=Prometheus%20is%20a%20single,to%20share%20and%20aggregate%20data). Server Prometheus định kỳ **kéo (pull)** metrics từ các **exporter** hoặc dịch vụ mục tiêu thông qua HTTP. Dữ liệu được lưu trong cơ chế lưu trữ thời gian thực tích hợp (TSDB) trên đĩa cục bộ với định dạng nén tối ưu cho truy vấn[influxdata.com](https://www.influxdata.com/comparison/prometheus-vs-victoria/#:~:text=Prometheus%20is%20a%20single,to%20share%20and%20aggregate%20data). Kiến trúc Prometheus mang tính mô-đun: bao gồm **Prometheus server** (thực hiện scrape và lưu trữ), **client library** (thư viện nhúng trong ứng dụng để xuất metric), **exporter** cho dịch vụ bên thứ ba (ví dụ: HAProxy, MySQL, Node exporter cho Linux, v.v.), **Pushgateway** cho phép đẩy metric từ các job ngắn hạn, và **Alertmanager** để quản lý cảnh báo[last9.io](https://last9.io/blog/prometheus-vs-victoriametrics/#:~:text=,ephemeral%20and%20batch%20jobs%20to)[last9.io](https://last9.io/blog/prometheus-vs-victoriametrics/#:~:text=or%20systems%20like%20Linux%20system,flexible%20query%20language%20for%20data). Prometheus hỗ trợ nhiều cơ chế **service discovery** (Kubernetes, DNS, v.v.) để tự động tìm các mục tiêu thu thập mới, giúp phù hợp với môi trường động như container/Kubernetes[last9.io](https://last9.io/blog/prometheus-vs-victoriametrics/#:~:text=,the%20targets%20it%20should%20scrape). Tuy nhiên, Prometheus **không có khả năng clustering tích hợp** – mỗi instance Prometheus hoạt động độc lập, không chia sẻ dữ liệu trực tiếp. Để mở rộng quy mô hoặc chịu lỗi, thường phải chạy nhiều instance và sử dụng **federation** (phân cấp truy vấn liên cụm) hoặc nhờ đến các dự án bên ngoài như Thanos, Cortex để ghép các dữ liệu lại[influxdata.com](https://www.influxdata.com/comparison/prometheus-vs-victoria/#:~:text=Prometheus%20is%20a%20single,to%20share%20and%20aggregate%20data). Kiến trúc đơn giản của Prometheus rất hiệu quả cho triển khai nhỏ lẻ, nhưng khi quy mô lớn lên, việc quản lý nhiều instance Prometheus và đảm bảo tính nhất quán trở nên phức tạp.

**Kiến trúc VictoriaMetrics:**

*Hình 2: Kiến trúc VictoriaMetrics ở chế độ cluster, tách biệt các thành phần tiếp nhận (vminsert), lưu trữ (vmstorage) và truy vấn (vmselect), hỗ trợ nhiều định dạng ghi dữ liệu (Prometheus remote\_write, Influx Line Protocol, Graphite, OpenTSDB).* VictoriaMetrics được thiết kế linh hoạt với hai tùy chọn triển khai: **single-node** và **cluster**[influxdata.com](https://www.influxdata.com/comparison/prometheus-vs-victoria/#:~:text=VictoriaMetrics%20is%20available%20in%20two,their%20database%20infrastructure%20as%20required). Phiên bản single-node là một binary duy nhất chứa tất cả chức năng (tương tự mô hình đơn của Prometheus) – thuận tiện cho việc cài đặt và bảo trì ở quy mô vừa và nhỏ[influxdata.com](https://www.influxdata.com/comparison/prometheus-vs-victoria/#:~:text=VictoriaMetrics%20is%20available%20in%20two,their%20database%20infrastructure%20as%20required). Phiên bản **VictoriaMetrics Cluster** gồm ba loại thành phần chính: **vminsert** (nhận và phân phối dữ liệu ghi vào các nút lưu trữ), **vmstorage** (lưu trữ chuỗi thời gian, có trạng thái), và **vmselect** (xử lý truy vấn, tổng hợp kết quả từ các nút lưu trữ)[medium.com](https://medium.com/99dotco/victoriametrics-replace-prometheus-8f17ad32cdf#:~:text=What%27s%20different%20with%20VM%3F). Cách tách biệt này cho phép VictoriaMetrics mở rộng **chiều ngang (scale-out)** dễ dàng: có thể tăng số lượng vminsert/vmstorage/vmselect độc lập tùy nhu cầu tài nguyên ghi, lưu trữ hay truy vấn. Các thành phần giao tiếp qua HTTP hoặc RPC nội bộ, có thể đặt sau load balancer để phân tải (Hình 2). Kiến trúc VM cluster tích hợp sẵn tính năng **replication** giữa các vmstorage (cấu hình sao chép dữ liệu) giúp đạt tính sẵn sàng cao (HA) và chống mất dữ liệu khi có node hỏng[last9.io](https://last9.io/blog/prometheus-vs-victoriametrics/#:~:text=Prometheus%20does%20not%20inherently%20support,requires%20manual%20effort%20and%20coordination). Bên cạnh đó, VictoriaMetrics còn cung cấp các dịch vụ phụ trợ: **vmagent** (một tác nhân thu thập dữ liệu nhẹ, hoạt động như Prometheus scraper, có thể thay thế Prometheus server trong việc thu thập rồi gửi (push) dữ liệu qua remote write[last9.io](https://last9.io/blog/prometheus-vs-victoriametrics/#:~:text=storage%2C%20respectively.%20,or%20any%20other%20compatible%20TSDB)), **vmalert** (xử lý các quy tắc cảnh báo và ghi, tương tự Alertmanager nhưng có thể gửi alert đến Alertmanager hoặc hệ thống khác)[last9.io](https://last9.io/blog/prometheus-vs-victoriametrics/#:~:text=,or%20any%20other%20compatible%20TSDB), **vmui** (giao diện web tích hợp sẵn để truy vấn dữ liệu), và **vmctl** (công cụ nhập liệu di chuyển dữ liệu từ các TSDB khác sang VictoriaMetrics)[last9.io](https://last9.io/blog/prometheus-vs-victoriametrics/#:~:text=%2A%20VictoriaMetrics%20Single,or%20Prometheus%20remote%20write%20protocol). Nhìn chung, kiến trúc VictoriaMetrics linh hoạt hơn, cho phép **triển khai phân tán** khi cần, nhưng vẫn có tùy chọn đơn giản dạng single-node khi khối lượng công việc nhỏ.

**So sánh triển khai:** Ở quy mô nhỏ, cả hai giải pháp đều có thể chạy nhanh chóng dưới dạng một dịch vụ đơn (Prometheus một binary ~> victoriaMetrics single binary). Prometheus có lợi thế **đơn giản** – chỉ cần chạy một server, cấu hình các **scrape config** là có hệ thống hoạt động. VictoriaMetrics single-node cũng đơn giản tương tự, đồng thời có thể **đọc trực tiếp file cấu hình prometheus.yml** để thực hiện việc scrape các target như Prometheus, tức là có thể thay thế drop-in cho Prometheus trong hầu hết các trường hợp thu thập dữ liệu[docs.victoriametrics.com](https://docs.victoriametrics.com/faq/#:~:text=FAQ%20,scraping%20can%20be%20done)[last9.io](https://last9.io/blog/prometheus-vs-victoriametrics/#:~:text=VictoriaMetrics%2C%20on%20the%20other%20hand%2C,Key%20features%20include). Sự khác biệt lớn xuất hiện ở quy mô lớn: Prometheus muốn lưu trữ lâu dài hoặc mở rộng sẽ cần tích hợp thêm tầng bên ngoài (như **Thanos/Cortex** để lưu trữ tập trung) – điều này làm tăng đáng kể độ phức tạp vận hành. Ngược lại, VictoriaMetrics hỗ trợ sẵn cluster nên người dùng có thể **scale-out nội tại** mà không cần công cụ bên ngoài[influxdata.com](https://www.influxdata.com/comparison/prometheus-vs-victoria/#:~:text=VictoriaMetrics%20is%20available%20in%20two,their%20database%20infrastructure%20as%20required). Tuy nhiên, vận hành cụm VictoriaMetrics đòi hỏi hiểu biết về các thành phần VM (vminsert/vmselect/vmstorage) và cơ chế phối hợp giữa chúng. Tóm lại, Prometheus thiên về triển khai độc lập, phù hợp cho hệ thống vừa và nhỏ hoặc thành phần thu thập phân tán; còn VictoriaMetrics cung cấp lộ trình sẵn sàng để mở rộng lên cấp độ cluster khi cần thiết.

**2. Hiệu suất ghi và truy vấn dữ liệu**

**Hiệu suất ghi (ingestion rate):** Prometheus có khả năng ghi hàng trăm nghìn mẫu dữ liệu mỗi giây trên một server tùy theo cấu hình phần cứng. Một thử nghiệm cho thấy Prometheus có thể ingest khoảng *240.000 mẫu/giây*, trong khi VictoriaMetrics đạt khoảng *360.000 mẫu/giây* trên cùng một cấu hình, tức cao hơn ~50%[last9.io](https://last9.io/blog/prometheus-vs-victoriametrics/#:~:text=Data%20Collection%20Pull%20based%20Pull,to%20360%2C000%20samples%20per%20second). Khả năng ghi của Prometheus bị giới hạn bởi tốc độ scrape và xử lý mẫu trong một tiến trình đơn. Nếu lượng dữ liệu vượt quá khả năng, Prometheus có thể bắt đầu **drop mẫu** hoặc tăng độ trễ scrape[last9.io](https://last9.io/blog/prometheus-vs-victoriametrics/#:~:text=ingestion%20rate.%20,samples%20or%20experience%20increased%20latency). VictoriaMetrics được thiết kế tối ưu hơn, sử dụng ít CPU và I/O cho mỗi mẫu, cho phép ingest nhanh hơn trên cùng phần cứng[last9.io](https://last9.io/blog/prometheus-vs-victoriametrics/#:~:text=,cardinality%20data%20and%20ingestion%20rates). Đặc biệt, VM hỗ trợ cả mô hình **pull và push** đồng thời: ngoài việc tự scrape như Prometheus, nó cho phép nhận trực tiếp dữ liệu đẩy vào (HTTP API hoặc **Prometheus remote\_write**). Mô hình push này tỏ ra hiệu quả khi số lượng nguồn dữ liệu lớn hoặc phân tán, giúp VM xử lý tốt dữ liệu có **độ phong phú nhãn rất cao (high cardinality)** mà Prometheus thuần pull có thể khó khăn[last9.io](https://last9.io/blog/prometheus-vs-victoriametrics/#:~:text=and%20disk%20space%20for%20the,cardinality%20data%20and%20ingestion%20rates). Trong thực tế, một cụm VictoriaMetrics có thể tiếp nhận lượng dữ liệu khổng lồ – chẳng hạn hệ thống Roblox đã đạt mức **120 triệu điểm dữ liệu mỗi giây** ghi vào cụm VictoriaMetrics mà vẫn hoạt động ổn định[datanami.com](https://www.datanami.com/2023/05/30/why-roblox-picked-victoriametrics-for-observability-data-overhaul/#:~:text=data%2C%20Dai%20said)[datanami.com](https://www.datanami.com/2023/05/30/why-roblox-picked-victoriametrics-for-observability-data-overhaul/#:~:text=%E2%80%9CWe%20have%20120%20million%20data,%E2%80%9D). Những con số này vượt xa khả năng của một instance Prometheus đơn lẻ (thông thường Prometheus bắt đầu quá tải ở mức vài trăm ngàn mẫu/giây tùy phần cứng).

**Hiệu suất truy vấn:** Ngôn ngữ truy vấn của Prometheus là **PromQL**, vốn được thiết kế mạnh mẽ cho việc xử lý chuỗi thời gian, tính toán các toán tử trên vector thời gian. Prometheus có thể xử lý hàng chục nghìn truy vấn mỗi giây (truy vấn đơn giản) và đối với truy vấn phức tạp, độ trễ thường tính bằng hàng chục mili-giây đến vài giây tùy độ dài thời gian quét[victoriametrics.com](https://victoriametrics.com/blog/reducing-costs-p1/#:~:text=Disk%20usage%2083,ile%207s%E2%9C%85%203.6s). Trong thử nghiệm hiệu năng, Prometheus đạt khoảng *80.000 truy vấn/giây* cho các truy vấn đơn giản, trong khi VictoriaMetrics đạt tới *100.000 truy vấn/giây*[last9.io](https://last9.io/blog/prometheus-vs-victoriametrics/#:~:text=Data%20Collection%20Pull%20based%20Pull,to%20360%2C000%20samples%20per%20second). Đối với truy vấn phức tạp trên tập dữ liệu lớn, VictoriaMetrics tỏ ra vượt trội: độ trễ truy vấn trung bình ~0,35 giây so với ~0,50 giây của Prometheus trong một benchmark độc lập[soufianebouchaara.com](https://soufianebouchaara.com/victoriametrics-db-exploring-the-latest-version-and-performance-benchmarks/#:~:text=). Một thử nghiệm khác do chính đội ngũ VictoriaMetrics thực hiện cho thấy độ trễ truy vấn bách phân vị 50 của Prometheus ~70 ms, trong khi VictoriaMetrics chỉ ~4 ms (nhanh hơn ~16 lần), và độ trễ P99 giảm từ 7 giây xuống ~3,6 giây[victoriametrics.com](https://victoriametrics.com/blog/reducing-costs-p1/#:~:text=Disk%20usage%2083,ile%207s%E2%9C%85%203.6s). Sự khác biệt này nhờ VM có engine truy vấn tối ưu hơn cho khối lượng lớn, cũng như cơ chế lưu trữ được thiết kế cho truy xuất nhanh. Ngoài ra, VictoriaMetrics hỗ trợ ngôn ngữ mở rộng **MetricsQL** (mở rộng từ PromQL) với các hàm và toán tử mới giúp viết truy vấn phức tạp dễ dàng hơn[last9.io](https://last9.io/blog/prometheus-vs-victoriametrics/#:~:text=VictoriaMetrics%20is%20backward%20compatible%20with,with%20an%20extension%20called%20MetricsQL). Tuy nhiên, ngay cả khi chỉ sử dụng cú pháp PromQL gốc, hiệu năng query trên VM thường cao hơn do cách tổ chức dữ liệu và thuật toán nén tốt hơn.

**Sử dụng tài nguyên (CPU, RAM, I/O):** Một điểm khác biệt quan trọng là hiệu quả sử dụng bộ nhớ và đĩa. Trong một benchmark **scrape 3400 node\_exporter** (với ~3,4 triệu series), Prometheus tiêu thụ dần đến ~14 GB RAM và có lúc vọt lên tới **23 GB RAM**, trong khi VictoriaMetrics giữ ổn định quanh **4,3 GB RAM**[valyala.medium.com](https://valyala.medium.com/prometheus-vs-victoriametrics-benchmark-on-node-exporter-metrics-4ca29c75590f#:~:text=VictoriaMetrics%20consistently%20uses%204,shutdown%20such%20as%20OOM%20crash). Như vậy Prometheus yêu cầu bộ nhớ nhiều gấp ~5,3 lần so với VictoriaMetrics cho cùng khối lượng dữ liệu trong thử nghiệm này[valyala.medium.com](https://valyala.medium.com/prometheus-vs-victoriametrics-benchmark-on-node-exporter-metrics-4ca29c75590f#:~:text=According%20to%20the%20graph%20above,3GB%29%20more%20RAM%20than%20VictoriaMetrics). Việc sử dụng RAM cao của Prometheus một phần do kiến trúc lưu trữ: Prometheus giữ một **head block** dữ liệu mới trong bộ nhớ để ghi nhanh, và chỉ định kỳ ghi xuống đĩa; khi số lượng chuỗi thời gian tăng, head block phình to dẫn tới tốn RAM[valyala.medium.com](https://valyala.medium.com/prometheus-vs-victoriametrics-benchmark-on-node-exporter-metrics-4ca29c75590f#:~:text=14GB%20of%20RSS%20memory%20with,shutdown%20such%20as%20OOM%20crash)[medium.com](https://medium.com/99dotco/victoriametrics-replace-prometheus-8f17ad32cdf#:~:text=Then%20you%20start%20to%20notice,quotas%2C%20but%20you%20cannot%20do). Ngược lại, VictoriaMetrics quản lý bộ nhớ hiệu quả hơn, ghi dữ liệu xuống đĩa liên tục hơn và có cơ chế nén tốt, nhờ đó tránh các đỉnh RAM đột biến. Về I/O và dung lượng đĩa, sự khác biệt còn rõ hơn: thử nghiệm cho thấy VM chỉ dùng ~33 GB đĩa để lưu cùng lượng dữ liệu mà Prometheus cần ~83,5 GB, tức VM tiết kiệm ~**2,5 lần dung lượng**[victoriametrics.com](https://victoriametrics.com/blog/reducing-costs-p1/#:~:text=Prometheus%20VictoriaMetrics%20CPU%20avg%20used,ile%207s%E2%9C%85%203.6s). Một báo cáo khác thậm chí chỉ ra VM có thể nén dữ liệu hiệu quả gấp **7 lần** Prometheus trong kịch bản lưu trữ dài hạn[valyala.medium.com](https://valyala.medium.com/prometheus-vs-victoriametrics-benchmark-on-node-exporter-metrics-4ca29c75590f#:~:text=VictoriaMetrics%20needs%20up%20to%205x,to%20significant%20infrastructure%20cost%20savings). Lý do là VictoriaMetrics áp dụng thuật toán nén cải tiến (dựa trên Gorilla) cùng các cấu trúc dữ liệu tối ưu, trong khi Prometheus dùng cơ chế nén LZF đơn giản hơn[last9.io](https://last9.io/blog/prometheus-vs-victoriametrics/#:~:text=VictoriaMetrics%2C%20however%2C%20stands%20out%20with,Key%20benefits%20include). Kết quả là **tốc độ ghi/đọc** của VictoriaMetrics cao hơn đồng thời **tiết kiệm tài nguyên** hơn so với Prometheus trong các bài test quy mô lớn[valyala.medium.com](https://valyala.medium.com/prometheus-vs-victoriametrics-benchmark-on-node-exporter-metrics-4ca29c75590f#:~:text=VictoriaMetrics%20needs%20up%20to%205x,to%20significant%20infrastructure%20cost%20savings). Dĩ nhiên, hiệu năng cụ thể còn tùy thuộc cấu hình phần cứng và kiểu workload, nhưng xu hướng chung cho thấy VictoriaMetrics ưu thế ở cường độ ghi cao và truy vấn trên tập dữ liệu lớn, trong khi Prometheus có thể gặp giới hạn về bộ nhớ và I/O sớm hơn.

**3. Khả năng mở rộng (Scale-up, Scale-out)**

**Mở rộng chiều dọc (scale-up):** Cả Prometheus và VictoriaMetrics đều có thể **scale-up** bằng cách chạy trên máy chủ có cấu hình mạnh hơn (nhiều CPU, RAM, IOPS hơn). Prometheus được thiết kế khá tối ưu nên có thể xử lý hàng triệu series trên một node đơn nếu có đủ tài nguyên. Tuy nhiên, giới hạn vật lý (như kích thước RAM) sẽ sớm đạt tới khi dữ liệu tăng, do Prometheus không thể tách thành nhiều node để chia tải. Người quản trị thường phải tăng giới hạn bộ nhớ cho Prometheus, nhưng điều này có thể dẫn tới tình trạng Prometheus ngốn hết tài nguyên node và bị OOM kill nếu vượt quá giới hạn[medium.com](https://medium.com/99dotco/victoriametrics-replace-prometheus-8f17ad32cdf#:~:text=Then%20you%20start%20to%20notice,issue%20if%20running%20on%20Kubernetes). Thực tế vận hành cho thấy có trường hợp Prometheus tiêu thụ tới ~100GB RAM trong Kubernetes, gây ảnh hưởng đến các pod khác trên node và nếu restart sẽ mất dữ liệu trong bộ nhớ[medium.com](https://medium.com/99dotco/victoriametrics-replace-prometheus-8f17ad32cdf#:~:text=Then%20you%20start%20to%20notice,issue%20if%20running%20on%20Kubernetes). Ngược lại, VictoriaMetrics single-node thường tiêu thụ ít RAM hơn cho cùng khối lượng (như đã nêu, ~4 GB vs 14 GB trong test)[valyala.medium.com](https://valyala.medium.com/prometheus-vs-victoriametrics-benchmark-on-node-exporter-metrics-4ca29c75590f#:~:text=VictoriaMetrics%20consistently%20uses%204,shutdown%20such%20as%20OOM%20crash), nên trên cùng một máy chủ, VM có thể scale-up tiếp nhận nhiều dữ liệu hơn trước khi chạm giới hạn. Bên cạnh đó, do VM ghi xuống đĩa liên tục với pattern tuần tự và nén cao, nó có thể tận dụng ổ đĩa dung lượng lớn (kể cả HDD) hiệu quả hơn Prometheus (thường khuyến nghị SSD do ghi random I/O khi snapshot)[valyala.medium.com](https://valyala.medium.com/prometheus-vs-victoriametrics-benchmark-on-node-exporter-metrics-4ca29c75590f#:~:text=RSS%20Memory%20usage%3A%20VictoriaMetrics%20vs,Prometheus). Tóm lại, về scale-up, VictoriaMetrics tận dụng tài nguyên hiệu quả hơn, cho phép đẩy ngưỡng cao hơn trên một node so với Prometheus.

**Mở rộng chiều ngang (scale-out):** Đây là điểm khác biệt then chốt. **Prometheus không hỗ trợ scale-out trực tiếp** do thiết kế không cluster. Mỗi server Prometheus là một silo dữ liệu riêng. Để mở rộng, cách phổ biến là **shard dữ liệu** bằng cách chạy nhiều Prometheus, mỗi cái phụ trách một tập con mục tiêu (ví dụ phân vùng theo dịch vụ hoặc theo cụm). Tuy nhiên, việc này dẫn đến phân tán dữ liệu – muốn có cái nhìn toàn cảnh, phải **federation** (một Prometheus cấp cao hơn tổng hợp từ các Prometheus cấp dưới) hoặc dùng giải pháp như **Thanos, Cortex, Mimir** để hợp nhất dữ liệu đa instance. Các giải pháp này về cơ bản tạo một lớp cluster bên ngoài Prometheus: ví dụ Thanos thêm các component store, sidecar, query để tập hợp nhiều TSDB; Cortex/Mimir cho phép chạy Prometheus ở chế độ stateless và ghi dữ liệu vào storage chung. Mặc dù những giải pháp đó giúp Prometheus có thể scale-out, chúng **phức tạp để triển khai và vận hành**, yêu cầu hàng chục pod và cấu phần (như một người dùng chia sẻ, cụm Mimir của họ cần >100 pods, hàng trăm GB RAM, kèm hệ thống object storage riêng, ... để ingest hàng triệu series[reddit.com](https://www.reddit.com/r/kubernetes/comments/1351kk7/any_reasons_to_use_prometheus_over_victoriametrics/#:~:text=IronRedSix)). Tựu trung, mở rộng ngang với Prometheus đòi hỏi nỗ lực tích hợp đáng kể, và hiểu biết sâu về các dự án mở rộng.

Trái lại, **VictoriaMetrics hỗ trợ clustering out-of-the-box**. Ở chế độ **cluster mode**, ta có thể tăng số lượng **vminsert**, **vmstorage**, **vmselect** để scale tuyến tính theo tải ghi, lưu trữ và truy vấn tương ứng[influxdata.com](https://www.influxdata.com/comparison/prometheus-vs-victoria/#:~:text=VictoriaMetrics%20is%20available%20in%20two,their%20database%20infrastructure%20as%20required). Ví dụ, nếu cần tăng thông lượng ghi, có thể bổ sung thêm nhiều node vminsert (các node này stateless và có thể đặt sau load balancer để nhận metric từ nhiều nguồn đồng thời). Tương tự, thêm vmstorage nodes để mở rộng dung lượng lưu trữ và chia nhỏ khối lượng dữ liệu (mỗi vmstorage lưu một phần chuỗi thời gian, thường được partition theo hàm băm của metric). Các vmselect nodes cũng có thể tăng để phục vụ nhiều truy vấn song song, chúng sẽ phân tán truy vấn tới tất cả vmstorage cần thiết và tổng hợp kết quả【41†】. Nhờ thiết kế tách biệt này, VictoriaMetrics cluster có thể **mở rộng tuyến tính** khi thêm tài nguyên, đồng thời tích hợp sẵn tính **replication** (ví dụ mỗi dữ liệu có thể ghi vào 2 vmstorage ở 2 zone khác nhau) để đảm bảo HA. Một ví dụ điển hình cho khả năng scale-out của VM: hệ thống Roblox đã triển khai cụm VictoriaMetrics ingest **120 triệu mẫu/giây** với **5 tỷ series đang hoạt động** – một quy mô cực lớn[reddit.com](https://www.reddit.com/r/kubernetes/comments/1351kk7/any_reasons_to_use_prometheus_over_victoriametrics/#:~:text=Roblox%20is%20ingesting%20120Mil%20samples%2Fs,on%20used%20resources%20from%20them). Trong khi đó, rất khó (nếu không muốn nói bất khả thi) đạt được con số tương tự trên một cụm Prometheus truyền thống, nếu không dùng giải pháp thay thế TSDB khác. Cũng cần lưu ý rằng, vận hành một cụm VictoriaMetrics yêu cầu công sức tương xứng (quản lý nhiều thành phần), nhưng nhờ có kiến trúc tích hợp, mức độ phức tạp vẫn được đánh giá là **đơn giản hơn** so với việc dựng cả “vũ trụ” Thanos/Cortex quanh Prometheus.

**Độ co giãn và linh hoạt:** Một khía cạnh mở rộng khác là khả năng co giãn linh hoạt (scalability elasticity). Prometheus khi chạy đơn lẻ khó co giãn tự động – ví dụ không thể dễ dàng thêm bớt node để tạm thời tăng năng lực rồi giảm xuống, vì mỗi node là một thực thể độc lập dữ liệu. VictoriaMetrics cluster có thể co giãn linh hoạt hơn: ta có thể thêm vminsert/vmselect trong giờ cao điểm và gỡ bỏ khi không cần, miễn sao đảm bảo các vmstorage vẫn đủ chứa dữ liệu (hoặc thậm chí thêm vmstorage rồi sau đó re-balance dữ liệu). Mặc dù việc thêm bớt node cluster cũng cần kế hoạch (tránh mất dữ liệu khi tháo node lưu trữ), nhưng kiến trúc phân tán cho phép VM thích ứng tốt hơn với tải thay đổi. Ngoài ra, VM hỗ trợ **phân vùng dữ liệu theo tenant** (multi-tenancy) tốt hơn – có thể thiết lập nhiều **accountID** khác nhau trong cluster để tách biệt dữ liệu (tương tự việc chạy nhiều Prometheus nhưng chung cluster vật lý). Prometheus truyền thống muốn phân tách dữ liệu theo tenant thường phải chạy nhiều instance riêng biệt.

Tóm lại, về khả năng mở rộng: **Prometheus phù hợp cho quy mô nhỏ đến trung bình**, với kiến trúc đơn giản, nhưng **gặp hạn chế ở quy mô rất lớn hoặc yêu cầu HA**, đòi hỏi bổ sung nhiều thành phần ngoài. **VictoriaMetrics** được thiết kế hướng đến **quy mô lớn và dài hạn**, có sẵn cụm phân tán để mở rộng, đồng thời single-node của nó cũng đủ nhẹ cho trường hợp nhỏ. Lựa chọn giữa hai giải pháp sẽ phụ thuộc vào việc hệ thống của bạn có dự kiến tăng trưởng lớn về dữ liệu hay không và mức độ chấp nhận phức tạp khi vận hành cụm.

**4. Chi phí phần cứng và vận hành**

**Hiệu quả tài nguyên và chi phí hạ tầng:** Do những khác biệt về hiệu năng và sử dụng tài nguyên đã phân tích, VictoriaMetrics thường mang lại **chi phí phần cứng thấp hơn** cho cùng khối lượng dữ liệu so với Prometheus. Cụ thể, trong thử nghiệm với hàng ngàn target node\_exporter, VM dùng ít hơn ~5 lần RAM và ~7 lần dung lượng đĩa so với Prometheus[valyala.medium.com](https://valyala.medium.com/prometheus-vs-victoriametrics-benchmark-on-node-exporter-metrics-4ca29c75590f#:~:text=According%20to%20the%20graph%20above,3GB%29%20more%20RAM%20than%20VictoriaMetrics)[valyala.medium.com](https://valyala.medium.com/prometheus-vs-victoriametrics-benchmark-on-node-exporter-metrics-4ca29c75590f#:~:text=VictoriaMetrics%20needs%20up%20to%205x,to%20significant%20infrastructure%20cost%20savings). Điều này có nghĩa một cụm VM có thể chạy trên số máy chủ ít hơn hoặc cấu hình thấp hơn mà vẫn lưu trữ được lượng dữ liệu tương đương, từ đó **giảm chi phí đầu tư phần cứng** (đặc biệt là RAM và ổ đĩa NVMe/SSD vốn đắt tiền). Đội ngũ VictoriaMetrics tuyên bố việc tối ưu cấu trúc dữ liệu và thuật toán giúp giảm đến **90% chi phí lưu trữ** và đem lại *10 lần* tiết kiệm chi phí cloud trong các trường hợp sử dụng của khách hàng[victoriametrics.com](https://victoriametrics.com/blog/victoriametrics-slashes-data-storage-bills-with-worlds-most-cost-efficient-monitoring/#:~:text=A%20combination%20of%20VictoriaMetrics%E2%80%99%20optimized,up%20to%20x4%20network%20costs). Dù con số này có thể mang tính quảng cáo, thực tế cũng cho thấy các tổ chức chuyển sang VM để cắt giảm đáng kể hóa đơn cơ sở hạ tầng – ví dụ Grammarly báo cáo giảm đến 90% chi phí lưu trữ sau khi dùng VictoriaMetrics[victoriametrics.com](https://victoriametrics.com/blog/victoriametrics-slashes-data-storage-bills-with-worlds-most-cost-efficient-monitoring/#:~:text=A%20combination%20of%20VictoriaMetrics%E2%80%99%20optimized,up%20to%20x4%20network%20costs).

Một yếu tố chi phí khác là **năng lượng và hiệu suất**. VictoriaMetrics tiêu thụ CPU ít hơn cho mỗi đơn vị dữ liệu, đồng nghĩa với việc cùng một công việc giám sát, VM có thể chạy trên lượng CPU nhỏ hơn (hoặc server ít core hơn) so với Prometheus[last9.io](https://last9.io/blog/prometheus-vs-victoriametrics/#:~:text=,cardinality%20data%20and%20ingestion%20rates). Điều này giúp **tiết kiệm điện năng** và giảm nhiệt tỏa, nhất là trong các data center lớn. Với xu hướng các công ty quan tâm đến chi phí vận hành và dấu chân carbon, việc chuyển sang giải pháp monitoring tối ưu hơn như VM có thể góp phần giảm hóa đơn điện và làm mát. Một báo cáo cho biết kết hợp tối ưu của VM giúp **giảm tải xử lý và lưu trữ đến 90%** so với giải pháp tương tự, từ đó gián tiếp cắt giảm tiêu thụ năng lượng[victoriametrics.com](https://victoriametrics.com/blog/victoriametrics-slashes-data-storage-bills-with-worlds-most-cost-efficient-monitoring/#:~:text=A%20combination%20of%20VictoriaMetrics%E2%80%99%20optimized,up%20to%20x4%20network%20costs).

**Chi phí vận hành (nhân sự, phức tạp hệ thống):** Bên cạnh phần cứng, chi phí vận hành bao gồm công sức duy trì hệ thống. Ở phương diện này, Prometheus có ưu thế ở quy mô nhỏ vì nó **đơn giản**, nhiều kỹ sư đã quen thuộc, và có hệ sinh thái công cụ hỗ trợ phong phú. Việc cài đặt, cấu hình Prometheus khá nhanh; nhiều biểu đồ giám sát, cảnh báo có sẵn (do cộng đồng đóng góp) có thể import dùng ngay. Chi phí vận hành Prometheus chủ yếu tăng cao khi hệ thống phình to: khi đó phải quản lý nhiều instance và giải pháp tầng trên, đòi hỏi kỹ sư chuyên sâu về Thanos/Cortex hay tuning Prometheus để tránh quá tải. Ngược lại, VictoriaMetrics single-node cũng đơn giản không kém để vận hành ở mức nhỏ. Khi chuyển sang chế độ cluster, độ phức tạp cao hơn Prometheus đơn lẻ, nhưng vẫn nhẹ hơn so với việc quản lý cả một “stack” mở rộng Prometheus. Thay vì phải điều phối nhiều thành phần khác hãng (Prom + DB ngoài hoặc object storage + query layer), với VM bạn làm việc trong một hệ giải pháp thống nhất do một nhóm phát triển cung cấp. Theo kinh nghiệm người dùng, việc **di trú từ Prometheus sang VictoriaMetrics khá dễ dàng** – “không có thành phần nào thay đổi, cú pháp query PromQL giữ nguyên, chỉ cần triển khai VM và trỏ Grafana tới đó, mọi dashboard sẵn có tiếp tục hoạt động”[medium.com](https://medium.com/99dotco/victoriametrics-replace-prometheus-8f17ad32cdf#:~:text=Here%E2%80%99s%20one%20solution%20for%20using,no%20difference%20in%20data%20storage). Điều này giúp giảm chi phí chuyển đổi và đào tạo.

**Chi phí sự cố và độ tin cậy:** Một dạng chi phí ẩn là khi hệ thống gặp sự cố (downtime, mất dữ liệu). Prometheus có thể gặp trường hợp **mất dữ liệu khi OOM hoặc tắt đột ngột** do dữ liệu trong bộ nhớ chưa kịp flush ra đĩa[valyala.medium.com](https://valyala.medium.com/prometheus-vs-victoriametrics-benchmark-on-node-exporter-metrics-4ca29c75590f#:~:text=14GB%20of%20RSS%20memory%20with,shutdown%20such%20as%20OOM%20crash). Sự cố này dẫn đến lỗ hổng trong chuỗi thời gian, ảnh hưởng đến phân tích giám sát. Để khắc phục, thường phải tăng nhiều bộ nhớ dự phòng cho Prometheus hoặc chạy 2 instance song song (HA) ghi trùng lặp dữ liệu – cả hai giải pháp đều tốn kém (thêm RAM hoặc tăng gấp đôi tài nguyên). VictoriaMetrics nhờ sử dụng bộ nhớ ổn định và flush liên tục nên giảm thiểu nguy cơ mất dữ liệu khi crash; hơn nữa, ở chế độ cluster với replication, nếu một node gặp sự cố, node khác đã có bản sao dữ liệu. Do đó, **VM có độ tin cậy cao hơn trong việc giữ dữ liệu**, giúp tiết kiệm chi phí khắc phục sự cố. Dĩ nhiên, việc chạy cluster VM cũng không tránh khỏi chi phí (ví dụ phải giám sát thêm các node VM, lo cân bằng tải, etc.), nhưng tổng thể, nhiều tổ chức cho rằng chi phí này **đáng đánh đổi** để có hệ thống quan sát ổn định và mở rộng.

Tổng kết lại, **chi phí phần cứng** cho giải pháp VictoriaMetrics thường **thấp hơn** do hiệu suất tài nguyên tốt hơn[valyala.medium.com](https://valyala.medium.com/prometheus-vs-victoriametrics-benchmark-on-node-exporter-metrics-4ca29c75590f#:~:text=VictoriaMetrics%20needs%20up%20to%205x,to%20significant%20infrastructure%20cost%20savings). **Chi phí vận hành** tùy thuộc quy mô: nhỏ thì Prometheus và VM tương đương; lớn thì VM cluster có thể đơn giản hơn việc duy trì nhiều cụm Prometheus + lớp mở rộng. Khi tính tổng chi phí sở hữu (TCO), các tổ chức lớn như CERN, Roblox đã chọn VictoriaMetrics vì bài toán kinh tế khi lưu trữ hàng tỷ chuỗi thời gian trong nhiều năm trở nên khả thi hơn so với dùng Prometheus thuần[dcnnmagazine.com](https://dcnnmagazine.com/data/victoriametrics-monitoring-cern/#:~:text=The%20CMS%20experiment%20has%20long,began%20the%20search%20for%20alternatives)[datanami.com](https://www.datanami.com/2023/05/30/why-roblox-picked-victoriametrics-for-observability-data-overhaul/#:~:text=By%20all%20accounts%2C%20the%20move,amount%20of%20data%2C%20Dai%20said).

**5. Tính dễ triển khai, bảo trì và nâng cấp**

**Triển khai ban đầu:** Cả Prometheus và VictoriaMetrics single-node đều dễ dàng triển khai. Prometheus có sẵn **docker image** chính thức, chart Helm, v.v. và tích hợp chặt với hệ sinh thái Kubernetes (ví dụ Prometheus Operator) giúp việc cài đặt trong cluster rất thuận tiện. Chỉ cần định nghĩa các *ScrapeConfig* và rule, Prometheus có thể đi vào hoạt động trong vài phút. VictoriaMetrics cũng cung cấp image và chart Helm; ngoài ra còn có **VictoriaMetrics Operator** để chạy VM (tương tự Prom Operator). Triển khai VM single-node không khác biệt nhiều so với Prometheus – thậm chí bạn có thể trỏ VM dùng trực tiếp file prometheus.yml quen thuộc[docs.victoriametrics.com](https://docs.victoriametrics.com/#:~:text=VictoriaMetrics%20can%20be%20used%20as,according%20to%20the%20specification)[docs.victoriametrics.com](https://docs.victoriametrics.com/faq/#:~:text=FAQ%20,scraping%20can%20be%20done). Do đó, về mặt khởi động, **độ dễ dàng tương đương**. Điểm lợi của VM là nếu bạn muốn chạy chế độ cluster cho HA, VM cũng đã có sẵn hướng dẫn triển khai (docker-compose, Kubernetes manifests) để dựng cụm với các thành phần vminsert/vmselect/vmstorage. Mặc dù cấu hình cluster phức tạp hơn single node, so với việc thiết lập Thanos/Cortex thì VM cluster vẫn ít bước hơn (do không cần tích hợp thêm storage bên ngoài hay sidecar).

**Bảo trì định kỳ:** Prometheus là dự án CNCF lâu đời, có lịch phát hành phiên bản đều đặn và quy trình upgrade tương đối đơn giản (thay binary hoặc container image rồi chạy lại). Một ưu điểm của Prometheus là tính **ổn định đã được kiểm chứng** – các phiên bản mới hiếm khi thay đổi đột ngột gây lỗi, và cộng đồng rộng lớn nhanh chóng phát hiện vá lỗi. VictoriaMetrics phát triển với tốc độ nhanh (nhiều tính năng mới)[docs.victoriametrics.com](https://docs.victoriametrics.com/#:~:text=How%20to%20upgrade%20VictoriaMetrics%20), người dùng được khuyến cáo nâng cấp thường xuyên để nhận tối ưu và vá lỗi mới nhất. Việc nâng cấp VM (single và cluster) nhìn chung **an toàn và ít downtime**: nhà phát triển khẳng định có thể dừng tiến trình VM bằng tín hiệu SIGINT để nó flush dữ liệu rồi nâng cấp nhịp nhàng, Prometheus hoặc vmagent gửi dữ liệu đến VM sẽ buffer một chút và **không làm mất dữ liệu** trong quá trình restart nhanh[docs.victoriametrics.com](https://docs.victoriametrics.com/#:~:text=,Start%20the%20upgraded%20VictoriaMetrics)[docs.victoriametrics.com](https://docs.victoriametrics.com/#:~:text=Prometheus%20doesn%E2%80%99t%20drop%20data%20during,applies%20also%20to%20vmagent). Đối với cluster VM, quy trình nâng cấp có thể cần tuần tự (upgrade lần lượt các thành phần, đảm bảo tương thích), nhưng tài liệu cho biết có thể **skip nhiều phiên bản** nếu cần và hỗ trợ downgrade khi lỗi[docs.victoriametrics.com](https://docs.victoriametrics.com/#:~:text=VictoriaMetrics%20is%20developed%20at%20a,page%20and%20performing%20regular%20upgrades). Nhìn chung, **nâng cấp VM không quá phức tạp**, chỉ cần chú ý ghi nhật ký phát hành để xem thay đổi tương thích.

Về bảo trì hiệu năng, Prometheus đòi hỏi quản trị viên phải thường xuyên theo dõi các chỉ số như dung lượng TSDB, thời gian query, độ lớn của **tsdb chunks**… để điều chỉnh trước khi gặp sự cố (ví dụ điều chỉnh thời gian lưu retention, cấu hình số chuỗi tối đa, v.v.). VictoriaMetrics do tự động xử lý phân mảnh và tối ưu nén nên giảm bớt một số công việc bảo trì (như không cần lo lắng về compaction vì VM làm nội bộ). Tuy nhiên, VM có thêm thành phần thì cần giám sát thêm chúng: ví dụ đảm bảo vminsert không bị nghẽn, vmstorage đủ dung lượng đĩa, các vmselect cân bằng tải tốt.

**Khả năng nâng cấp/hỗ trợ tương lai:** Prometheus có một **cộng đồng lớn và nhiều bên đóng góp**, do đó người dùng có thể tin tưởng vào lộ trình phát triển dài hạn, cũng như dễ tìm sự trợ giúp (qua diễn đàn, Github). Hệ sinh thái Prometheus cũng phong phú, hầu như mọi dịch vụ phổ biến đều có **exporter** và **dashboard mẫu** cộng đồng chia sẻ. Đây là điểm mạnh về “bảo trì mở rộng”: khi cần giám sát thứ mới, khả năng cao là đã có sẵn plugin và kiến thức cho Prometheus. VictoriaMetrics tuy đang nổi lên nhưng cộng đồng riêng còn nhỏ hơn nhiều. Một số ý kiến từ cộng đồng Prometheus chỉ ra rằng VictoriaMetrics hiện chưa đóng góp ngược lại nhiều cho hệ sinh thái chung, và họ lo ngại VM đang “tự làm mọi thứ theo cách riêng”[reddit.com](https://www.reddit.com/r/kubernetes/comments/1351kk7/any_reasons_to_use_prometheus_over_victoriametrics/#:~:text=VictoriaMetrics%20makes%20a%20lot%20of,had%20a%20decimal%20place%20off). Điều này có thể ảnh hưởng đến việc **tương thích dài hạn** – ví dụ nếu Prometheus/PromQL có thay đổi lớn, VM phải tự cập nhật theo mà không có sự phối hợp trực tiếp; người dùng VM cũng phụ thuộc vào đội ngũ VM cho các tính năng mới thay vì cộng đồng rộng. Tuy vậy, VictoriaMetrics là mã nguồn mở và đội ngũ phát triển rất năng động (các issue, thảo luận trên Github của VM được phản hồi nhanh). Về khía cạnh hỗ trợ doanh nghiệp, cả hai dự án đều có công ty đứng sau (Prometheus có nhiều công ty trong CNCF hỗ trợ, VictoriaMetrics có công ty cung cấp dịch vụ hỗ trợ thương mại). Do đó, lựa chọn nào cũng có con đường hỗ trợ chính thức nếu cần.

Tóm lại, **dễ triển khai/bảo trì** cho hệ thống nhỏ: Prometheus và VM đều đơn giản, Prometheus quen thuộc hơn với nhiều người. **Cho hệ thống lớn**: bảo trì Prometheus đòi hỏi tích lũy kinh nghiệm điều chỉnh và có thể cần tích hợp phức tạp, trong khi VM cung cấp một giải pháp tương đối trực quan (dù cần học cách vận hành cluster VM). Việc nâng cấp cả hai đều khá thuận tiện, nhưng với VM cần chủ động cập nhật để tận dụng cải tiến hiệu năng liên tục.

**6. Tính tương thích với hệ sinh thái (PromQL, Grafana, Kubernetes, v.v.)**

**Ngôn ngữ truy vấn và API:** Prometheus sử dụng PromQL làm ngôn ngữ truy vấn chuẩn và có **HTTP API** phong phú để truy xuất chuỗi thời gian, metadata, quản lý (ví dụ API /api/v1/query, /api/v1/series, v.v.). Đây đã trở thành **chuẩn giao tiếp** cho hầu hết công cụ giám sát hiện nay. VictoriaMetrics hoàn toàn **tương thích ngược với PromQL** – mọi truy vấn PromQL hợp lệ đều có thể chạy trên VictoriaMetrics và trả về kết quả tương đương[last9.io](https://last9.io/blog/prometheus-vs-victoriametrics/#:~:text=VictoriaMetrics%20is%20backward%20compatible%20with,with%20an%20extension%20called%20MetricsQL). VM cũng **implement đầy đủ API Prometheus** (đọc và ghi)[last9.io](https://last9.io/blog/prometheus-vs-victoriametrics/#:~:text=VictoriaMetrics%20provides%20a%20comprehensive%20HTTP,parts%20based%20on%20the%20functionality), do đó có thể được sử dụng như một **drop-in replacement** trong các môi trường đang dùng Prometheus. Ví dụ, Grafana khi kết nối đến VM sẽ coi nó như datasource loại Prometheus (sử dụng chung giao thức truy vấn). Thử nghiệm cho thấy việc thay thế data source từ Prometheus sang VictoriaMetrics trong Grafana hoàn toàn suôn sẻ, các dashboard biểu đồ không cần chỉnh sửa[medium.com](https://medium.com/99dotco/victoriametrics-replace-prometheus-8f17ad32cdf#:~:text=Here%E2%80%99s%20one%20solution%20for%20using,no%20difference%20in%20data%20storage). Không chỉ hỗ trợ PromQL, VictoriaMetrics còn mở rộng qua **MetricsQL** – bổ sung hàm mới (như xử lý nhiều chuỗi cùng lúc, làm việc với chuỗi dạng string, tổng hợp nâng cao) giúp người dùng có thể viết truy vấn phức tạp một cách ngắn gọn hơn[last9.io](https://last9.io/blog/prometheus-vs-victoriametrics/#:~:text=VictoriaMetrics%20is%20backward%20compatible%20with,with%20an%20extension%20called%20MetricsQL). Lưu ý rằng MetricsQL là siêu tập của PromQL, người dùng không bắt buộc phải dùng nếu muốn giữ tính di động của query giữa các hệ thống.

**Tích hợp với Grafana và công cụ quan sát:** Prometheus tích hợp mật thiết với Grafana – hầu hết dashboard mẫu trên Grafana Labs đều giả định data source là Prometheus. Như đã đề cập, do VM hỗ trợ API tương thích, Grafana cũng tương thích hoàn toàn với VM. Ngoài ra, VMUI (giao diện web tích hợp trong VictoriaMetrics) cung cấp một số tiện ích truy vấn và khám phá metric, nhưng nhìn chung Grafana vẫn là lựa chọn chính cho trực quan. Về **Alertmanager**: Prometheus đi kèm Alertmanager để gửi thông báo (qua email, Slack, PagerDuty, v.v.) khi rule cảnh báo thỏa mãn. VictoriaMetrics không bundling Alertmanager, nhưng **vmalert** của VM có thể đánh giá các **cảnh báo theo định nghĩa Prometheus rules** và gửi tới một Alertmanager bên ngoài hoặc các webhook khác[docs.victoriametrics.com](https://docs.victoriametrics.com/vmalert/#:~:text=This%20service%20simplifies%20the%20setup,as%20the%20integration%20with%20Alertmanager). Như vậy, người dùng có thể tiếp tục dùng Alertmanager quen thuộc với Prometheus ngay cả khi backend lưu trữ chuyển sang VM. Ở khía cạnh **Pushgateway**: Prometheus dùng thành phần này để các job ngắn hạn push metric khi kết thúc. VictoriaMetrics có thể **nhận push trực tiếp** mà không cần qua pushgateway – ví dụ một ứng dụng có thể gọi API /api/v1/import của VM để gửi metric ngay khi hoàn thành. Tuy nhiên, nếu tổ chức đã có quy trình với Pushgateway, VM cũng có thể tương thích bởi vmagent/VM có thể scrape dữ liệu từ Pushgateway tương tự Prometheus (vì pushgateway cũng cung cấp endpoint /metrics cho Prom scrape). Thực tế, do VM hỗ trợ cả mô hình push, nhiều trường hợp có thể loại bỏ thành phần Pushgateway và cho job đẩy thẳng vào VM, giúp kiến trúc đơn giản hơn.

**Kubernetes và service discovery:** Prometheus nổi tiếng với khả năng tự động phát hiện dịch vụ trong Kubernetes (thông qua Kubernetes API) để tự cấu hình target cần scrape. VictoriaMetrics hỗ trợ **Prometheus service discovery** giống hệt – trong file cấu hình VM, ta có thể định nghĩa kubernetes\_sd\_configs, file\_sd\_configs... giống như trong prometheus.yml[github.com](https://github.com/VictoriaMetrics/VictoriaMetrics/blob/master/docs/vmagent.md#:~:text=VictoriaMetrics%2Fdocs%2Fvmagent.md%20at%20master%20,node%20VictoriaMetrics%20can%20also)[docs.victoriametrics.com](https://docs.victoriametrics.com/faq/#:~:text=FAQ%20,scraping%20can%20be%20done). Điều này có nghĩa VM có thể **thay thế trực tiếp Prometheus trong các cluster Kubernetes**: bạn chỉ cần trỏ operator hoặc config map sang chạy VM (hoặc vmagent) với cấu hình SD tương tự. Tính tương thích này mở rộng đến cả **relabeling**: các luật relabel trong Prometheus cũng hoạt động tương tự trong VM (VM triển khai trình **Prometheus-compatible service discovery and relabeling** theo FAQ nhà phát triển). Nhiều tổ chức đã thực hiện migration mà các workload trên Kubernetes hầu như không nhận ra sự khác biệt, do VM “giả lập” toàn bộ hành vi SD của Prometheus[dcnnmagazine.com](https://dcnnmagazine.com/data/victoriametrics-monitoring-cern/#:~:text=%E2%80%9CWe%20were%20searching%20for%20alternative,member%20of%20CMS%20collaboration).

**Hệ sinh thái exporter, tích hợp khác:** Prometheus có hàng trăm **exporter** và tích hợp chính thức/không chính thức cho mọi loại cơ sở hạ tầng (từ hệ điều hành, database, middleware đến ứng dụng). Vì VictoriaMetrics thu thập dữ liệu qua cùng giao thức Prom (format /metrics và PromQL), **toàn bộ exporter hiện có đều tương thích** với VM. Bạn có thể tiếp tục chạy node\_exporter, mysqld\_exporter, v.v. y như khi dùng Prometheus. Hơn nữa, VM còn cung cấp **tương thích với nhiều giao thức khác**: ví dụ hỗ trợ ingest qua **InfluxDB line protocol**, **Graphite plaintext protocol**, **OpenTSDB put protocol**【41†】. Điều này mở ra khả năng hợp nhất nhiều nguồn metric khác nhau vào một nơi (Prometheus thì chỉ hỗ trợ định dạng Prom). Một ví dụ: bạn có hệ thống cũ dùng Graphite – thay vì vận hành cả Graphite và Prometheus, bạn có thể gửi dữ liệu Graphite vào VictoriaMetrics, đồng thời gửi dữ liệu Prometheus vào đó, tạo thành kho chung. Khả năng “đa ngôn ngữ” này là ưu thế rõ rệt của VM về tích hợp.

**Cộng đồng và tài nguyên hỗ trợ:** Prometheus là dự án **CNCF Graduated**, có cộng đồng khổng lồ, tài liệu và hướng dẫn phong phú. Bất kỳ thắc mắc hay vấn đề nào với Prometheus hầu như đều đã có người gặp và chia sẻ cách giải quyết. VictoriaMetrics tuy không thuộc CNCF nhưng có cộng đồng người dùng riêng đang lớn mạnh; công ty đứng sau VM cũng tích cực xuất bản hướng dẫn, case study (ví dụ blog VM, case CERN, Roblox...). Về mặt này, Prometheus vẫn chiếm ưu thế do tính phổ biến (một *kỹ sư SRE/DevOps trung bình* thường đã quen Prometheus hơn). Tuy nhiên, mức độ **trưởng thành** của VM cũng đang tăng – nhiều diễn đàn bắt đầu có chuyên mục VM, và người dùng lớn như Roblox, Cloudflare... đứng ra chia sẻ kinh nghiệm, nghĩa là hệ sinh thái VM dần hình thành.

**Tương thích trong tương lai:** Do VM cam kết hỗ trợ PromQL, việc đồng tồn tại với Prometheus trong môi trường lai là hoàn toàn khả thi. Bạn thậm chí có thể **chạy song song** Prometheus và VM: Prometheus tiếp tục scrape và gửi *remote\_write* vào VM làm storage dài hạn, trong khi vẫn phục vụ alert thời gian thực ngắn hạn. Cách kết hợp này cho phép tận dụng ưu điểm cả hai (Prometheus cho cảnh báo nhanh, VM cho lưu trữ dài hạn) và rất nhiều tổ chức đã áp dụng (ví dụ CMS tại CERN ban đầu dùng VM làm **remote storage** cho Prometheus trước khi chuyển hẳn sang VM)[dcnnmagazine.com](https://dcnnmagazine.com/data/victoriametrics-monitoring-cern/#:~:text=Initially%20implementing%20VictoriaMetrics%20as%20backend,removing%20cardinality%20issues%20with%20Influx). Tóm lại, về hệ sinh thái, VictoriaMetrics được thiết kế để **đồng hành và thay thế dần Prometheus một cách êm ái**, thay vì tạo ra một hệ sinh thái xa lạ. Điều này giúp người dùng yên tâm hơn khi cân nhắc chuyển đổi, vì họ không bị khóa vào một hệ sinh thái hoàn toàn mới.

**7. Trường hợp sử dụng thực tế và kinh nghiệm người dùng**

Để có cái nhìn cụ thể, sau đây là một số trường hợp sử dụng Prometheus và VictoriaMetrics trong thực tế cũng như phản hồi từ cộng đồng:

* **Quy mô vừa và nhỏ:** Phần lớn các công ty khởi đầu với Prometheus cho giám sát cụm Kubernetes và hạ tầng cơ bản. Ví dụ, một doanh nghiệp triển khai ~20 microservice và dùng Prometheus thu thập cả metrics hệ thống lẫn ứng dụng. Prometheus đáp ứng tốt nhu cầu, nhưng khi lượng dữ liệu tăng (do traffic tăng), người ta bắt đầu thấy dấu hiệu như **trễ trong biểu đồ Grafana, lỗ hổng dữ liệu** do Prometheus phải khởi động lại khi OOM[medium.com](https://medium.com/99dotco/victoriametrics-replace-prometheus-8f17ad32cdf#:~:text=Prometheus%2C%20one,restart%20and%20start%20debugging%20Prometheus)[medium.com](https://medium.com/99dotco/victoriametrics-replace-prometheus-8f17ad32cdf#:~:text=Then%20you%20start%20to%20notice,issue%20if%20running%20on%20Kubernetes). Trong các trường hợp này, nếu quy mô chưa quá lớn, giải pháp thường là **tuning Prometheus** (tăng tài nguyên, giảm retention) hoặc phân tách nhiều Prometheus. Nhiều người cho biết “nếu dữ liệu ít, Prometheus hoàn toàn ổn. Nhưng khi data nhiều lên, Prometheus sẽ nhanh chóng ngốn bộ nhớ và *kẹt cứng* với lượng dữ liệu đó”[reddit.com](https://www.reddit.com/r/kubernetes/comments/1351kk7/any_reasons_to_use_prometheus_over_victoriametrics/#:~:text=We%20made%20the%20switch%20from,46%20clusters). Điều này khớp với kinh nghiệm: Prometheus tuyệt vời cho vài chục ngàn series, nhưng đến hàng trăm ngàn hoặc vài triệu series thì việc vận hành đòi hỏi kỹ năng cao.
* **Chuyển đổi sang VictoriaMetrics do tăng trưởng:** Một người dùng trên Reddit chia sẻ họ đã **chuyển từ ~10 instance Prometheus (mỗi cluster một cái) sang một cụm VictoriaMetrics duy nhất cho 46 cluster**. Lý do là khi số liệu tăng, việc chạy nhiều Prometheus trở nên cồng kềnh và tốn tài nguyên, trong khi một cụm VM lớn có thể đảm đương tất cả[reddit.com](https://www.reddit.com/r/kubernetes/comments/1351kk7/any_reasons_to_use_prometheus_over_victoriametrics/#:~:text=We%20made%20the%20switch%20from,46%20clusters). Kết quả là hệ thống đơn giản hơn (thay vì quản lý 10 Prom riêng rẽ, giờ chỉ quản lý một cluster), và không còn cảnh Prometheus “choking” trên đống data. Đây là xu hướng chung: **khi kiến trúc microservice ngày càng nhiều dịch vụ, lượng metric bùng nổ**, nhiều công ty đã tìm đến VM như một giải pháp tập trung, tối ưu.
* **Case study: CERN CMS (thí nghiệm hạt nhân):** Nhóm vận hành hạ tầng cho thí nghiệm **CMS tại CERN** từng dùng Prometheus và InfluxDB để lưu trữ metrics khoa học và hạ tầng. Họ gặp vấn đề về **dung lượng lưu trữ và khả năng mở rộng** với giải pháp cũ khi cần lưu dữ liệu dài hạn[dcnnmagazine.com](https://dcnnmagazine.com/data/victoriametrics-monitoring-cern/#:~:text=The%20CMS%20experiment%20has%20long,began%20the%20search%20for%20alternatives). Sau khi đọc bài benchmark của VictoriaMetrics, nhóm quyết định thử dùng VM. Ban đầu, họ tích hợp **VictoriaMetrics làm backend storage** cho Prometheus (tức Prometheus vẫn scrape, nhưng ghi dữ liệu dài hạn vào VM)[dcnnmagazine.com](https://dcnnmagazine.com/data/victoriametrics-monitoring-cern/#:~:text=The%20CMS%20monitoring%20team%20has,by%20the%20detail%20on%20display). Nhận thấy hiệu quả, họ tiến tới thay thế hẳn InfluxDB và Prometheus bằng **VictoriaMetrics làm front-end** thu thập luôn. Kết quả, hệ thống CMS **không còn gặp vấn đề cardinality** (số lượng nhãn rất lớn vốn làm InfluxDB chậm) và **ổn định trong vận hành** – theo lời nhóm kỹ sư, từ khi cài VM họ “không còn gặp bất kỳ sự cố nào về cardinality hay vận hành phần mềm”[dcnnmagazine.com](https://dcnnmagazine.com/data/victoriametrics-monitoring-cern/#:~:text=Since%20installing%20VictoriaMetrics%2C%20the%20CMS,vmalert%2C%20the%20solution%E2%80%99s%20alerting%20system). Việc triển khai được hỗ trợ nhờ tính tương thích ngược của VM với Prometheus, giúp chuyển đổi **mượt mà, liền mạch**[dcnnmagazine.com](https://dcnnmagazine.com/data/victoriametrics-monitoring-cern/#:~:text=%E2%80%9CWe%20were%20searching%20for%20alternative,member%20of%20CMS%20collaboration). Đây là ví dụ cho thấy VictoriaMetrics có thể đáp ứng những môi trường đòi hỏi cao (dữ liệu nhiều petabyte, phân tán toàn cầu) mà các giải pháp trước đó không kham nổi.
* **Case study: Roblox (nền tảng trò chơi trực tuyến):** Roblox có hệ thống quan sát rất lớn phục vụ hàng trăm triệu người dùng. Họ từng sử dụng nhiều công cụ khác nhau và gặp khó khăn trong việc tập trung hóa dữ liệu quan sát. Sau quá trình benchmark, Roblox đã quyết định chọn **VictoriaMetrics + Grafana** làm nền tảng mới[datanami.com](https://www.datanami.com/2023/05/30/why-roblox-picked-victoriametrics-for-observability-data-overhaul/#:~:text=%E2%80%9CWe%20also%20choose%20to%20use,%E2%80%9D). Kết quả được lãnh đạo kỹ thuật Roblox chia sẻ: hệ thống mới giúp họ đạt **120 triệu điểm dữ liệu mỗi giây** ingest vào VictoriaMetrics, đồng thời cung cấp một nền tảng tập trung để kỹ sư dễ dàng truy cập metrics (thay vì phân mảnh nhiều hệ thống trước kia)[datanami.com](https://www.datanami.com/2023/05/30/why-roblox-picked-victoriametrics-for-observability-data-overhaul/#:~:text=don%E2%80%99t%20have%20to%20hunt%20around,amount%20of%20data%2C%20Dai%20said). Thậm chí, Roblox đạt **100% uptime trong 3 quý liên tiếp** cho hệ thống quan sát mới – điều rất ấn tượng với lượng dữ liệu khổng lồ họ xử lý[datanami.com](https://www.datanami.com/2023/05/30/why-roblox-picked-victoriametrics-for-observability-data-overhaul/#:~:text=By%20all%20accounts%2C%20the%20move,amount%20of%20data%2C%20Dai%20said). Họ đánh giá cao tính linh hoạt và hiệu năng của VM, cho phép công ty tiếp tục tăng trưởng nhanh mà hệ thống quan sát vẫn theo kịp. Trường hợp Roblox khẳng định khả năng của VM ở mức “enterprise khổng lồ”, và cũng minh chứng rằng Grafana+VM có thể thay thế hoàn toàn Grafana+Prometheus trong thực tiễn.
* **Ý kiến từ cộng đồng Prometheus:** Bên cạnh những lời khen, cũng có những ý kiến thận trọng về VictoriaMetrics. Một contributor của Prometheus đã bình luận rằng anh *giữ lựa chọn Prometheus vì cộng đồng phát triển mạnh* và sự minh bạch của dự án[reddit.com](https://www.reddit.com/r/kubernetes/comments/1351kk7/any_reasons_to_use_prometheus_over_victoriametrics/#:~:text=As%20a%20Prometheus%20contributor%2C%20the,Prometheus%20is%20the%20developer%20community). Anh cho rằng VictoriaMetrics mặc dù hiệu năng cao nhưng **ít nói về các trade-off** – ví dụ khi VM tối ưu mạnh cho ghi/đọc, có đánh đổi gì về độ chính xác, độ nhất quán? (Hiện chưa có cáo buộc cụ thể về sai lệch kết quả, nhưng đây là lo ngại tiềm ẩn). Anh cũng phê phán VM từng đưa ra vài benchmark “quảng cáo” mà sau đó phải cải chính do lỗi tính toán[reddit.com](https://www.reddit.com/r/kubernetes/comments/1351kk7/any_reasons_to_use_prometheus_over_victoriametrics/#:~:text=VictoriaMetrics%20makes%20a%20lot%20of,had%20a%20decimal%20place%20off). Ngoài ra, việc VictoriaMetrics phát triển nhiều thành phần tương tự nhưng tách biệt (như MetricsQL thay vì tham gia cải tiến PromQL) bị xem như “đi riêng” thay vì đóng góp chung[reddit.com](https://www.reddit.com/r/kubernetes/comments/1351kk7/any_reasons_to_use_prometheus_over_victoriametrics/#:~:text=decimal%20place%20off). Từ góc nhìn người dùng, điều này có nghĩa nếu chọn VM, bạn phần nào **phụ thuộc vào nhà phát triển VM** cho các cập nhật, thay vì có hàng trăm công ty cùng xây dựng như Prometheus. Dẫu vậy, nhiều người phản biện rằng **cạnh tranh** thúc đẩy sáng tạo – sự xuất hiện của VM buộc hệ sinh thái Prometheus phải cải tiến (ví dụ gần đây Prometheus cũng giới thiệu thử nghiệm hỗ trợ **push** một phần để bắt kịp nhu cầu[victoriametrics.com](https://victoriametrics.com/blog/reducing-costs-p1/#:~:text=The%20push%20model%20is%20the,it%20behind%20a%20feature%20flag)).

Tổng kết kinh nghiệm thực tế: Prometheus vẫn là lựa chọn hàng đầu cho hệ thống vừa và nhỏ, với rất nhiều **thành công trong sản xuất**. Khi hệ thống lớn dần, một số tổ chức chọn mở rộng Prometheus bằng giải pháp như Thanos/Mimir, số khác quyết định **chuyển sang VictoriaMetrics** để đơn giản hóa kiến trúc và tiết kiệm tài nguyên. VictoriaMetrics đã chứng minh hiệu quả ở vài môi trường khổng lồ (CERN, Roblox, Grammarly, Cloudflare...), cho thấy độ tin cậy và hiệu năng của nó không chỉ là lý thuyết. Tuy nhiên, mỗi môi trường đều khác, nên kinh nghiệm chung là: hãy thử đánh giá trên dữ liệu và workload cụ thể của mình. Nhiều nơi thực hiện POC (Proof of Concept) chạy song song Prometheus và VM để so sánh trước khi quyết định. Nhìn chung, phản hồi người dùng về VM đa phần tích cực ở khía cạnh kỹ thuật (nhanh, nhẹ, dễ dùng), còn về mặt cộng đồng/hệ sinh thái thì Prometheus vẫn vượt trội ở thời điểm hiện tại.

**8. Tổng kết ưu điểm – nhược điểm của từng giải pháp**

Để giúp quyết định lựa chọn phù hợp, bảng dưới đây tóm tắt các **ưu điểm (✔️)** và **nhược điểm (⚠️)** chính của Prometheus và VictoriaMetrics:

**Prometheus:**

* ✔️ **Đơn giản, phổ biến:** Dễ triển khai một server đơn; rất nhiều tài liệu, cộng đồng đông đảo hỗ trợ. Được CNCF hậu thuẫn, tích hợp sẵn vào nhiều nền tảng (Kubernetes, Grafana...). Có hàng trăm exporter và dashboard mẫu[influxdata.com](https://www.influxdata.com/comparison/prometheus-vs-victoria/#:~:text=Prometheus%20is%20an%20open,emphasis%20on%20reliability%20and%20scalability)[influxdata.com](https://www.influxdata.com/comparison/prometheus-vs-victoria/#:~:text=applications.%20It%20uses%20a%20pull,term%20data%20storage).
* ✔️ **Tính năng cảnh báo mạnh:** Cung cấp Alertmanager và cơ chế cảnh báo linh hoạt. PromQL là ngôn ngữ truy vấn mạnh mẽ, quen thuộc với giới DevOps.
* ✔️ **Phù hợp giám sát thời gian thực:** Thiết kế tối ưu cho dữ liệu ngắn hạn, truy vấn nhanh trên khung thời gian hẹp (vài giờ tới vài ngày). Độ trễ thấp, thích hợp cho cảnh báo tức thì.
* ⚠️ **Giới hạn mở rộng và lưu trữ:** Không có clustering tích hợp, khó mở rộng ngang khi dữ liệu lớn. Lưu trữ dài hạn kém hiệu quả, thường cần tích hợp remote storage hoặc chịu mất chi tiết (downsample)[last9.io](https://last9.io/blog/prometheus-vs-victoriametrics/#:~:text=Prometheus%20has%20an%20efficient%20storage,term%20data%20retention). **Không lý tưởng cho dữ liệu high-cardinality hoặc retention nhiều năm**[influxdata.com](https://www.influxdata.com/comparison/prometheus-vs-victoria/#:~:text=applications.%20It%20uses%20a%20pull,term%20data%20storage).
* ⚠️ **Sử dụng tài nguyên cao:** Cần nhiều RAM khi số series tăng, có thể dẫn tới OOM nếu không kiểm soát[valyala.medium.com](https://valyala.medium.com/prometheus-vs-victoriametrics-benchmark-on-node-exporter-metrics-4ca29c75590f#:~:text=14GB%20of%20RSS%20memory%20with,shutdown%20such%20as%20OOM%20crash). Dung lượng đĩa sử dụng lớn hơn do nén kém hơn (theo test tốn hơn ~7x so với VM)[valyala.medium.com](https://valyala.medium.com/prometheus-vs-victoriametrics-benchmark-on-node-exporter-metrics-4ca29c75590f#:~:text=VictoriaMetrics%20needs%20up%20to%205x,to%20significant%20infrastructure%20cost%20savings).
* ⚠️ **Mở rộng phức tạp:** Muốn HA hoặc scale-out phải chạy nhiều instance độc lập hoặc dùng giải pháp ngoài (Thanos/Cortex), làm tăng độ phức tạp vận hành.
* ⚠️ **Gián đoạn khi nâng cấp:** Instance đơn lẻ, khi nâng cấp hoặc restart có thể bị mất dữ liệu tạm thời chưa kịp ghi (trừ khi chạy 2 instance HA để bù nhau). Cần cẩn trọng khi nâng phiên bản để không mất series do format thay đổi (hiếm xảy ra nhưng cần lưu ý qua changelog).

**VictoriaMetrics:**

* ✔️ **Hiệu năng và hiệu quả cao:** Ghi nhận và truy vấn rất nhanh trên dữ liệu lớn. Thử nghiệm cho thấy dùng ít RAM (ít hơn 5x) và ít dung lượng đĩa (ít hơn 2.5-7x) so với Prometheus cho cùng workload[valyala.medium.com](https://valyala.medium.com/prometheus-vs-victoriametrics-benchmark-on-node-exporter-metrics-4ca29c75590f#:~:text=According%20to%20the%20graph%20above,3GB%29%20more%20RAM%20than%20VictoriaMetrics)[valyala.medium.com](https://valyala.medium.com/prometheus-vs-victoriametrics-benchmark-on-node-exporter-metrics-4ca29c75590f#:~:text=VictoriaMetrics%20needs%20up%20to%205x,to%20significant%20infrastructure%20cost%20savings). Phù hợp lưu trữ dài hạn do thuật toán nén vượt trội (có trường hợp giảm 90% dữ liệu lưu trữ)[victoriametrics.com](https://victoriametrics.com/blog/victoriametrics-slashes-data-storage-bills-with-worlds-most-cost-efficient-monitoring/#:~:text=A%20combination%20of%20VictoriaMetrics%E2%80%99%20optimized,up%20to%20x4%20network%20costs).
* ✔️ **Khả năng mở rộng linh hoạt:** Hỗ trợ chạy cluster phân tán, scale-out gần như tuyến tính bằng cách thêm node vminsert/vmstorage/vmselect[influxdata.com](https://www.influxdata.com/comparison/prometheus-vs-victoria/#:~:text=VictoriaMetrics%20is%20available%20in%20two,their%20database%20infrastructure%20as%20required). Dễ dàng cấu hình sao chép dữ liệu để HA. Single-node cũng có thể scale vertical rất tốt, đã được chứng minh trong nhiều case lớn (hàng triệu series).
* ✔️ **Tương thích hệ sinh thái Prometheus:** Hỗ trợ đầy đủ PromQL, API, service discovery, v.v. Có thể thay thế Prometheus mà **không cần thay đổi** các công cụ xung quanh (Grafana, Alertmanager, exporters)[medium.com](https://medium.com/99dotco/victoriametrics-replace-prometheus-8f17ad32cdf#:~:text=Here%E2%80%99s%20one%20solution%20for%20using,no%20difference%20in%20data%20storage)[dcnnmagazine.com](https://dcnnmagazine.com/data/victoriametrics-monitoring-cern/#:~:text=%E2%80%9CWe%20were%20searching%20for%20alternative,member%20of%20CMS%20collaboration). Bổ sung hỗ trợ nhận dữ liệu từ InfluxDB, Graphite... giúp hợp nhất nguồn metric.
* ✔️ **Dễ chuyển đổi và vận hành dài hạn:** Người dùng báo cáo quá trình migrate không đau đớn, VM chạy ổn định, ít sự cố về bộ nhớ hay cardinality[dcnnmagazine.com](https://dcnnmagazine.com/data/victoriametrics-monitoring-cern/#:~:text=Since%20installing%20VictoriaMetrics%2C%20the%20CMS,vmalert%2C%20the%20solution%E2%80%99s%20alerting%20system). Việc bảo trì (compaction, downsampling) được VM xử lý nội bộ tự động.
* ⚠️ **Cộng đồng nhỏ hơn:** Ít người dùng hơn Prometheus, tài liệu chủ yếu do nhà phát triển cung cấp. Các tính năng mới, fix bug phụ thuộc vào team VictoriaMetrics; cộng đồng đóng góp chưa nhiều (dù dự án mở mã nguồn).
* ⚠️ **Tính “đặc thù”:** MetricsQL là mở rộng hữu ích nhưng cũng tạo sự khác biệt so với chuẩn chung (nếu lạm dụng MetricsQL, sau này khó quay lại PromQL nguyên bản). Một số giải pháp xung quanh Prometheus có thể không hỗ trợ trực tiếp VictoriaMetrics (ví dụ các tool/query Prom-specific).
* ⚠️ **Triển khai cluster phức tạp hơn Prometheus đơn:** Dù dễ hơn Thanos nhưng so với chạy 1 binary Prom, việc vận hành 3 loại node (insert/select/storage) yêu cầu kỹ năng quản lý phân tán. Với team nhỏ, đây có thể là trở ngại.
* ⚠️ **Chưa được thử thách theo thời gian như Prometheus:** Dù có case lớn, VM vẫn là “người chơi mới” so với thâm niên >10 năm của Prometheus. Một số người lo ngại về tính đúng đắn và bền vững dài hạn (dù chưa có sự cố nghiêm trọng nào được báo cáo công khai)[reddit.com](https://www.reddit.com/r/kubernetes/comments/1351kk7/any_reasons_to_use_prometheus_over_victoriametrics/#:~:text=VictoriaMetrics%20makes%20a%20lot%20of,had%20a%20decimal%20place%20off).

**Kết luận**

Prometheus và VictoriaMetrics đều là các giải pháp mạnh cho cơ sở dữ liệu chuỗi thời gian trong giám sát hệ thống, mỗi cái có triết lý thiết kế riêng. **Prometheus** đem lại sự đơn giản, tin cậy đã được khẳng định, cùng hệ sinh thái rộng lớn – thích hợp cho hầu hết triển khai giám sát thông thường và thời gian thực. **VictoriaMetrics** nổi bật ở hiệu năng và khả năng mở rộng, giải quyết những điểm yếu cố hữu của Prometheus khi đối mặt với dữ liệu “big data” quan sát (độ phân giải cao, cực nhiều nhãn, lưu trữ dài hạn). Sự lựa chọn giữa hai phụ thuộc vào nhu cầu cụ thể: Nếu hệ thống của bạn vừa phải, yêu cầu không quá lớn về lượng metric và bạn muốn tận dụng tối đa cộng đồng có sẵn – Prometheus vẫn là lựa chọn số một. Nhưng nếu bạn dự kiến hạ tầng sẽ *bùng nổ* về số lượng metric, hoặc đang gặp trần về hiệu năng với Prometheus, thì VictoriaMetrics đáng để thử nghiệm nhờ ưu thế về tối ưu tài nguyên và scale-out. Cũng không nhất thiết phải “chọn một bỏ một” – nhiều tổ chức kết hợp cả hai, dùng Prometheus làm công cụ cảnh báo nhanh và dùng VictoriaMetrics làm kho lưu trữ dài hạn phía sau[dcnnmagazine.com](https://dcnnmagazine.com/data/victoriametrics-monitoring-cern/#:~:text=Initially%20implementing%20VictoriaMetrics%20as%20backend,removing%20cardinality%20issues%20with%20Influx). Trong mọi trường hợp, hiểu rõ ưu nhược điểm của từng giải pháp (như phân tích ở trên) sẽ giúp bạn thiết kế hệ thống quan sát cân bằng giữa hiệu năng, chi phí và độ phức tạp, sẵn sàng hỗ trợ mục tiêu kinh doanh trong ngắn hạn lẫn dài hạn.