Here are **open-source alternatives to VictoriaMetrics** for time-series data storage and monitoring, along with their **key differences, advantages, and tradeoffs**:

---

1. Prometheus + Thanos/Cortex

What It Is:

- **Prometheus**: The de facto standard for metrics collection and alerting (pull-based).
- **Thanos/Cortex**: Scalable extensions for long-term storage and HA.

Differences vs. VictoriaMetrics:

Feature	Prometheus + Thanos/Cortex	VictoriaMetrics
Storage	Requires separate object storage (e.g., MinIO) for long-term data.	Built-in storage (no external dependencies).
Scalability	Thanos/Cortex adds complexity for clustering.	Single binary or cluster mode (simpler scaling).
Query Performance	Good for mid-scale; slower at petabyte scale.	Optimized for high cardinality and large datasets.
Compression	Moderate (depends on backend).	Best-in-class compression (2–5x better than Prometheus).
HA	Requires Thanos Sidecar/Cortex for HA.	Native replication and HA support.

Best For:

- Existing Prometheus users needing long-term retention.
- Multi-cluster/multi-tenant setups (Thanos).

---

2. M3DB (Uber's Time-Series DB)

What It Is:

- Distributed, scalable TSDB built for high cardinality and large-scale workloads.

Differences vs. VictoriaMetrics:

Feature	M3DB	VictoriaMetrics
Architecture	Complex (requires etcd, coordinator nodes).	Single binary or simple cluster setup.
Resource Usage	High (needs dedicated etcd cluster).	Lightweight (lower CPU/RAM usage).
Query Language	M3QL (custom).	PromQL and MetricsQL (compatible with Prometheus).
Compression	Good (similar to Prometheus).	Better compression ratios.

Best For:

- Extremely high cardinality (e.g., millions of unique time series).
- Teams with Kubernetes/container expertise.

---

3. TimescaleDB

What It Is:

- PostgreSQL extension optimized for time-series data (hybrid SQL/TSDB).

Differences vs. VictoriaMetrics:

Feature	TimescaleDB	VictoriaMetrics
Data Model	Relational (SQL) + time-series.	Pure time-series (no joins/transactions).
Query Flexibility	SQL with time-series functions.	PromQL/MetricsQL (metrics-specific).
Compression	Columnar compression (2–4x).	Better compression (5–10x).
Scalability	Vertical scaling or distributed hypertables.	Horizontally scalable with native clustering.

Best For:

- Teams already using PostgreSQL.
- Mixed workloads (e.g., metrics + relational metadata).

---

4. InfluxDB (Open Source)

What It Is:

- Popular TSDB with a focus on metrics and event data.

Differences vs. VictoriaMetrics:

Feature	InfluxDB OSS	VictoriaMetrics
Scalability	Limited clustering in OSS version.	Native clustering in open-source version.
Query Language	InfluxQL (SQL-like).	PromQL/MetricsQL.
Compression	Moderate (depends on data types).	Superior compression for numeric metrics.
Ecosystem	Integrates with Telegraf.	Compatible with Prometheus exporters.

Best For:

- Simple setups with low-to-mid scalability needs.
- Teams familiar with InfluxQL.

---

5. Graphite (Whisper/Carbon)

What It Is:

- One of the earliest TSDBs, designed for static metrics.

Differences vs. VictoriaMetrics:

Feature	Graphite	VictoriaMetrics
Scalability	Limited (vertical scaling only).	Horizontal scaling.
Data Model	Fixed-resolution metrics.	Supports high-resolution, dynamic metrics.
Query Language	Basic (no PromQL support).	PromQL/MetricsQL.
Performance	Struggles with high cardinality.	Built for high cardinality.

Best For:

- Legacy systems already using Graphite.
- Static, low-cardinality metrics.

Summary: Which Should You Choose?

1. **Prometheus + Thanos/Cortex**:
    - **Use Case**: Multi-cluster monitoring with existing Prometheus expertise.
    - **Avoid If**: You want simplicity (VictoriaMetrics is easier to scale).
2. **M3DB**:
    - **Use Case**: Extreme cardinality (e.g., per-user metrics).
    - **Avoid If**: You lack Kubernetes/DevOps resources.
3. **TimescaleDB**:
    - **Use Case**: Combining time-series and relational data (e.g., ISO 8583 metrics + PostgreSQL reference tables).
    - **Avoid If**: Pure metrics storage (VictoriaMetrics is more efficient).

4. **InfluxDB OSS**:
    - **Use Case**: Small-to-mid-scale setups with InfluxQL familiarity.
    - **Avoid If**: You need horizontal scaling (use VictoriaMetrics or M3DB).
5. **Graphite**:
    - **Use Case**: Legacy integration.
    - **Avoid If**: Starting fresh (use VictoriaMetrics or Prometheus).

Why VictoriaMetrics Still Shines

- **Simplicity**: Single binary with built-in clustering.
- **Cost Efficiency**: 5–10x lower storage costs vs. Prometheus/InfluxDB.
- **Performance**: Handles 10M+ active time series on modest hardware.
- **Compatibility**: Drop-in replacement for Prometheus (same scrapers/exporters).

For your **ISO 8583 monitoring system** (100k–200k TPM), **VictoriaMetrics** remains the best fit due to its compression, scalability, and Prometheus compatibility. However, if you need **SQL joins** or **relational data integration**, consider **TimescaleDB**.