Big Data Visualization

Tools, Techniques, and an Observability Case Study

Z5008 Big Data Lab

IIT Madras Zanzibar

Tuesday, 9 June 2026



Agenda

1. What is **Big Data Visualization**? (3Vs: volume, variety, velocity)
2. Why visualization matters: decisions, comprehension, engagement
3. Common visualization types (heatmaps, networks, geo maps, treemaps, ...)
4. Tools overview: Tableau, Power BI, Qlik, Superset, Jupyter, D3, Grafana, Splunk, ...
5. Best practices & challenges
6. Why monitoring matters (observability in 2026)
7. Prometheus + Grafana: roles, differences, and how they work together



What is Big Data Visualization?

Definition

Big Data Visualization uses techniques and tools to **graphically represent large and complex datasets** so they are easier to understand and interpret.

- **Volume**: massive data needs summarization without losing key signals
- **Variety**: structured, semi-structured, and unstructured data from many sources
- **Velocity**: fast streams (e.g., IoT, social media) often require real-time views



Why Big Data Visualization Matters

1. **Enhanced decision-making**: see the story quickly and act faster
2. **Improved comprehension**: spot patterns, correlations, and outliers
3. **Increased engagement**: interactive dashboards support drill-down and exploration



Common Visualization Types for Big Data

- **Heatmaps**: density / intensity across regions or categories
- **Geospatial maps**: add location for spatial analysis
- **Tree maps**: hierarchical, part-to-whole relationships
- **Stream graphs**: changing volume over time across categories
- **Network diagrams**: nodes + edges for relationships / flows
- **Parallel coordinates**: multivariate (many dimensions) patterns
- **Chord diagrams**: inter-relationships / flows in a circular layout



pie.png



heatmaps.png



dashbaord.png



What to Look for in Big Data Visualization Tools

- Handle **multiple data types** (structured / semi-structured / unstructured)
- Support **filters** and fast slicing/dicing of large datasets
- Enable **interactivity** (zoom, drill-down, brushing, tooltips)
- **Integrations**: connect to data sources and other software (APIs, DBs, clouds)
- **Collaboration**: sharing, comments, permissions, reproducible views



Big Data Visualization Tools (Overview)

- **Jupyter Notebook**: code-first, reproducible analysis + visualization; great for experimentation and teaching
- **Tableau**: enterprise BI dashboards with strong drag-and-drop workflows
- **Power BI**: Microsoft ecosystem BI; reports + real-time dashboards
- **Qlik**: associative exploration (fast slicing across linked datasets)
- **Apache Superset**: open-source BI + dashboarding (SQL-first)



Big Data Visualization Tools (Web, Observability, Presentation)

- **Google Charts**: free web charts (HTML5/SVG) for quick embedding
- **D3.js**: highly-custom interactive web visuals (requires JavaScript)
- **Grafana**: time-series dashboards for metrics/logs/traces (common in ops)
- **Splunk**: search + analytics for machine data; strong for security/IT ops
- **Visme**: presentation-first visuals (infographics, reports, embeds)



Quick “When to Use What”

- Want **no-code dashboards**? Tableau / Power BI / Qlik / Superset
- Want **code + reproducibility**? Jupyter (plus Python/R plotting libs)
- Want **custom web interactions**? D3.js (or Google Charts for simpler cases)
- Want **ops/monitoring dashboards**? Grafana (often with Prometheus)
- Want **logs + security analytics**? Splunk
- Want **slides/infographics quickly**? Visme



Best Practices & Challenges

Best practices

- Simplify & aggregate (trends > noise)
- Add interactivity (filter, zoom, drill-down)
- Use color wisely (consistency + accessibility)
- Provide context (legends, annotations, benchmarks)

Common challenges

- Data quality
- Scalability / performance
- Complexity (many dimensions)
- Integration across sources



Observability & Monitoring

Big data visualization in production systems

Why Monitoring Matters (Observability in Practice)

The 2:00 AM question

Your service is down. Can you tell **what changed**, **when it degraded**, and **why it failed**?

- Monitoring is **big data visualization** for systems: high-volume time-series + events
- Modern stacks: microservices, Kubernetes, serverless, GPUs/AI workloads
- Goal: shift from **reactive firefighting** to **proactive reliability**
- Unified observability: **metrics + logs + traces** with shared context
- OpenTelemetry (OTel): common instrumentation standard across languages/services



Monitoring vs Visualization

Monitoring

- Continuous tracking of system health
- Uses **metrics/logs/traces** + thresholds/SLOs
- Produces **alerts** and supports incident response

Visualization

- Graphical representation of data (charts/dashboards)
- Helps humans see trends, patterns, and outliers
- Can be used for any domain (business, science, ops)

Key idea

Visualization is a **technique**. Monitoring is a **practice** that *uses* visualization plus collection, storage, querying, and alerting.



What is Prometheus?

Mental model

A high-tech **flight data recorder** for your infrastructure: it records vital signs and raises alarms.

- Pull-based scraping of metrics endpoints
- Time-series database (TSDB) optimized for metrics
- PromQL for querying and aggregation
- Alert rules + notification routing (Alertmanager)

Prometheus is a *data owner*

It collects and stores metric history.



Prometheus: Core Components

Building blocks

1. **Prometheus server**: discovery, scraping, local storage
2. **Exporters**: translate system/app stats to Prometheus metrics
3. **Alertmanager**: grouping, silences, inhibition, routing
4. **PromQL**: queries + recording rules
5. **Pushgateway**: short-lived batch jobs (when pull doesn't fit)

Typical metrics you scrape

- CPU, memory, disk, network
- Request rate, error rate, latency (including tail latency)
- Queue depth, DB pool usage, saturation indicators



What is Grafana?

Mental model

A **mission control center**: turns raw numbers into interactive dashboards people can act on.

- Dashboards for metrics, logs, and traces (multi-source)
- Rich visualization: time series, heatmaps, tables, annotations
- Collaboration: sharing, permissions, wallboards
- Alerting UI (often unified across multiple backends)

Grafana is a *data consumer*

It queries backends (e.g., Prometheus) on demand; it doesn't own your metric history.



Grafana Capabilities (2026-era stacks)

- "Big tent" data-source model: metrics + logs + traces + SQL/business data
- Drill-down workflows: click a spike in a chart and jump to related logs/traces
- Unified alerting view across multiple backends
- Increasing use of assisted/AI features for dashboard creation and investigation
- Extensible plugin ecosystem (dashboards, panels, data sources)



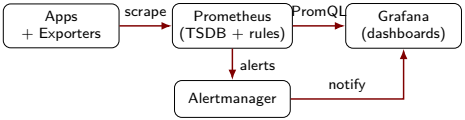
Prometheus vs Grafana: The Real Differences

Dimension	Prometheus	Grafana
Primary role	Collect + store metrics; evaluate rules	Visualize + explore; correlate across sources
Data ownership	Owns TSDB (metric history)	Stateless; queries other systems
Scope	Deep on metrics	Wide: metrics + logs + traces + more
Alerting	Rule evaluation + Alertmanager routing	Unified alert UI; can alert from many sources
UX	Engineer-focused query UI	Shareable dashboards, access control, collaboration



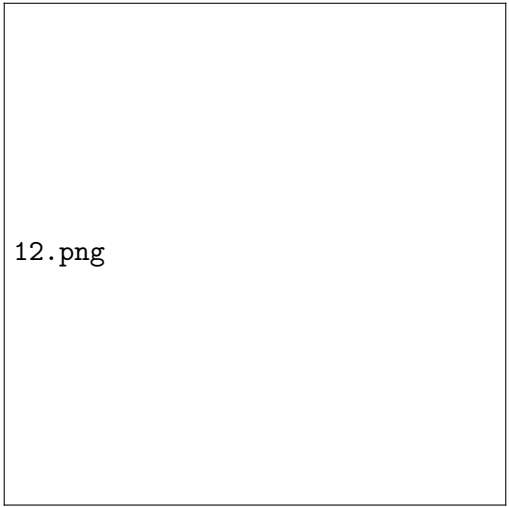
How They Work Together (Prom–Graf Stack)

1. Instrument services (often with OTel SDKs)
2. Expose /metrics or use exporters
3. Prometheus scrapes and stores time series
4. Grafana queries Prometheus using PromQL
5. Dashboards + alerts drive action (humans and automation)





Prometheus vs Grafana)





Real-World Scenario: Flash Sale (E-commerce)

What you see

Traffic spikes; latency threatens checkout; database pool saturates.

- Prometheus: captures rate/latency/saturation metrics; evaluates alert rules
- Alertmanager: routes pages to on-call, groups duplicates, respects silences
- Grafana: highlights affected services on a NOC dashboard and supports drill-down
- Correlation: jump from a latency spike to related logs/traces for root cause



Where It's Going: AIOps and Automation

- Predictive capacity planning (forecasting saturation before it happens)
- Cost and sustainability dashboards (cloud spend, utilization, efficiency)
- Self-healing runbooks: alerts trigger automated remediation (with guardrails)
- Assisted investigation: summarize anomalies, suggest likely causes, propose next queries



Security Observability (DevSecOps)

Detect + visualize

- Authentication failures, rate anomalies, suspicious traffic patterns
- Compliance dashboards (config drift, access patterns)
- Risk heatmaps from high-cardinality labels (used carefully)

Respond + preserve evidence

- Webhook-driven mitigations (e.g., temporary rate limiting)
- Incident snapshots: capture metrics/logs/traces context at alert time
- Role-based access control for sensitive security telemetry



Conclusion: Choose the Stack, Not a Side

- Prometheus: precise **metrics collection**, **storage**, and **rule-based alerting**
- Grafana: powerful **visualization**, **correlation**, and **collaboration**
- Together: a practical foundation for unified observability in 2026-era systems

One-liner

Prometheus answers: **“What happened (in metrics)?”**
Grafana answers: **“How do we see and explore it (across systems)?”**

