Metrics, Logs, and Dashboard Approach

Important Metrics for API Endpoint Performance

Following the RED Method (Rate, Errors, Duration) with additional context metrics:

Primary Metrics

1. **Request Rate**: Requests per second, broken down by endpoint and method
    - *Why*: Indicates load and usage patterns; sudden changes may correspond to issues
2. **Error Rate**: Percentage of requests resulting in errors (4xx, 5xx)
    - *Why*: Directly impacts user experience; immediate indicator of service health
    - *Breakdown*: By status code, endpoint, and error type
3. **Duration (Latency)**: Response time distributions with percentiles (p50, p90, p99)
    - *Why*: Shows performance degradation; high percentiles reveal edge case issues
    - *Components*: Break down by middleware, DB, external API calls

Supporting Metrics

4. **Saturation Metrics**: Resource utilization connected to each endpoint
    - CPU, Memory, Connection pools, Thread utilization
    - *Why*: Helps identify resource constraints causing performance issues
5. **Dependency Metrics**: Performance of connected services
    - Database query times, external API latency
    - *Why*: Many API issues stem from downstream dependencies
6. **Business Context Metrics**: Payload size, user count, tenant information
    - *Why*: Adds context to technical metrics for faster debugging

Log Strategy

Log Selection Criteria

1. **Signal-to-noise ratio**: Keep logs that provide actionable information
2. **Unique context**: Prioritize logs that add information not available in metrics
3. **Error details**: Maintain detailed logs for error conditions
4. **Transaction boundaries**: Preserve logs marking start/end of key operations

Logs to Add

1. **Structured Context**: Add JSON context to all logs with:
   - TraceID for correlation with traces
   - UserID/TenantID for request context
   - RequestID for grouping related logs
2. **Error Details**: Enhanced error logs with:
   - Stack traces for unexpected errors
   - Categorized error types for filtering
   - Relevant request parameters (sanitized)
3. **Performance Boundary Logs**: Mark entry/exit of critical operations with timing:
   - Database transactions
   - External service calls
   - Authentication/authorization steps

Logs to Remove

1. **Debug noise**: Remove excessive debug logs in production
2. **Redundant information**: Eliminate logs that duplicate metrics
3. **Periodic status logs**: Replace with metrics for system status
4. **Sensitive data**: Remove or mask any PII or sensitive information

Sample Dashboard Design

The API Performance Dashboard follows these design principles:

1. **Top-down approach**: Start with high-level health, drill down to components
2. **Correlation panels**: Place related metrics adjacently for easy comparison
3. **Context preservation**: Maintain selected time ranges when drilling down
4. **Actionable insights**: Include links to relevant logs and traces

Dashboard Sections

1. **Overview Section**
    - Service health summary (color-coded)
    - Request volume and error trends
    - SLO compliance indicators
    - Top 5 slowest endpoints
2. **Endpoint Performance Section**
    - Response time heatmap by endpoint
    - Error rate by endpoint
    - Resource utilization correlation
    - Apdex score for user satisfaction
3. **Dependency Analysis Section**
    - Database query performance
    - External API call latency
    - Cache hit/miss ratios
    - Connection pool utilization
4. **Debug Acceleration Section**
    - Recent error logs table
    - Slow transaction traces
    - Resource contention indicators
    - One-click filters for common issues