Security & CAP Theorem Guidelines

1. Security Best Practices for Key Resolution

SpEL & EL Security (MANDATORY)

To prevent Remote Code Execution (RCE) attacks via configuration or metadata injection, all Key Resolvers must adhere to the following:

- **Restricted Context**: Use SimpleEvaluationContext (Spring) or equivalent restricted contexts.
- **Prohibited Features**: Explicitly disable:
    - Method invocation on arbitrary classes.
    - Variable assignment.
    - Access to java.lang.Runtime, System, or ClassLoader.
- **Whitelisting (Core)**: Only the following variables are permitted by default:
    - #user: The authenticated principal.
    - #ip: The remote IP address.
    - #args: The method arguments array.
    - #headers: Map of incoming request headers.

Custom Variable Registration

To allow for extensibility while maintaining security, the library provides a registration mechanism for additional variables:

- **Registration Interface**: Framework adapters must provide a VariableProvider SPI.
- **Validation**: Registered variables must be checked against a "Forbidden Keywords" list (e.g., class, loader, system) before being injected into the evaluation context.
- **Scope**: Custom variables should be scoped to the specific request context and cleared immediately after the rate-limit decision is made.

2. CAP Theorem Alignment

Distributed Mode (L1 - Redis)

- **Classification**: **CP** (Consistency & Partition Tolerance).
- **Behavior**: Provides strong consistency across the cluster. If Redis is partitioned, the circuit breaker triggers a failover.

Fallback Mode (L1 Fail -> L2 Active)

- **Classification**: **AP** (Availability & Partition Tolerance).
- **Behavior**: Prioritizes service availability. Each node tracks limits independently in local memory.
- **Trade-off**: Total cluster traffic may exceed the global limit by (Node_Count - 1) * Limit.

This is an acceptable trade-off for infrastructure resilience.

3. Clock Synchronization

To prevent window fragmentation in distributed environments:

- **Reference Clock**: All algorithms MUST use StorageProvider.getCurrentTime().
- **Implementation**: In Redis, this calls TIME via the Lua script. In Caffeine, this defaults to System.currentTimeMillis().