# Comprehensive Approach to Performance Testing Without SLAs, Production Data, or Benchmarks

When conducting **performance testing from scratch** for a **application**—without production logs, SLAs, benchmarks, or competitive references—**hardware sizing, workload modeling, KPI estimation, and baseline testing** must be **systematically planned**.

---

**Step 1: Scenario Identification & Confirmation**

**Objective: Identify Business-Critical Workflows Without Existing Data**

- **How to identify key scenarios without logs or benchmarks?**
    - Engage directly with **product owners, developers, business analysts, and IT teams**.
    - Identify **business-critical transactions** that impact customers and regulatory compliance.
    - Consider both **internal banking operations** and **customer-facing interactions**.
    - Identify areas with **high database interaction, security-sensitive operations, and high-frequency transactions**.

**Scenario Categorization for Applications**

- **Frequent Transactions** (high user interactions, low complexity): Login, balance check, mini statement.
- **Moderate Transactions** (medium interaction, medium complexity): Funds transfer, bill payment.
- **Heavy Transactions** (high resource consumption, complex workflows): Loan application, credit score retrieval, KYC verification.

**Example - Application Test Scenarios**

| Scenario | Complexity | Assumed Frequency | Business Impact |
|---|---|---|---|
| User Login (Web/Mobile) | Low | Very High | Critical |
| Balance Inquiry | Low | Very High | High |
| Funds Transfer (Own Accounts) | Medium | High | High |
| Funds Transfer (Interbank) | High | Medium | Critical |
| Bill Payment | Medium | Medium | High |

| Scenario | Complexity | Assumed Frequency | Business Impact |
|---|---|---|---|
| Loan Application | High | Low | Medium |
| Credit Score Retrieval | High | Low | Medium |

---

**Step 2: Identify User Load Patterns**

**Objective: Estimate Traffic Without Historical Data**

Without **estimated DAU (Daily Active Users)** or **business insights**, we need a rational approach:

1. **Assume a range of potential customers.**
2. **Estimate concurrent users based on logical session duration assumptions.**
3. **Differentiate between peak and non-peak hours.**

**Estimating Concurrent Users**

- **Assumptions:**
  - **Total banking customers:** Unknown (assume 100,000 for modeling).
  - **Peak-time user percentage:** 10-15% of customers.
  - **Avg. session duration:** 5 minutes.
  - **Concurrent Users Formula:**

$$\text{Concurrent Users} = \frac{\text{Estimated Peak Users} \times \text{Avg. Session Duration (min)}}{\text{Total Minutes in Peak Period}}$$

$$= \frac{10,000 \times 5}{180} \approx 278 \text{ concurrent users}$$

**Load Distribution Over Time**

| Time Slot | Assumed % Traffic | Estimated Concurrent Users |
|---|---|---|
| 8 AM - 10 AM | 20% | 500 |
| 10 AM - 1 PM | 35% | 875 |
| 1 PM - 4 PM | 25% | 625 |
| 4 PM - 7 PM | 15% | 375 |

| Time Slot | Assumed % Traffic | Estimated Concurrent Users |
|---|---|---|
| 7 PM - 10 PM | 5% | 125 |

---

**Step 3: Define Performance KPIs**

**Objective: Establish Specific Performance Goals Without SLAs**

- Applications like banking must ensure **low latency**, **high availability**, and **security compliance**.
- Define **KPIs based on practical assumptions**, regulatory needs, and security guidelines.

**Derived KPIs**

| KPI | Estimated Target | Acceptable Deviation | Critical Limit |
|---|---|---|---|
| **Response Time** | < 2 sec | 2 - 3 sec | > 4 sec |
| **Throughput** | 1,200 TPS | 1,000 - 1,500 TPS | < 800 TPS |
| **Error Rate** | < 0.2% | 0.2 - 0.5% | > 1% |
| **CPU Utilization** | < 65% | 65 - 80% | > 90% |
| **Memory Usage** | < 60% | 60 - 75% | > 85% |
| **DB Query Execution Time** | < 150 ms | 150 - 500 ms | > 1 sec |

---

**Step 4: Workload Planning**

**Objective: Define Workload Distribution Without Prior Data**

- Assume **each user performs multiple transactions per session**.
- Define logical transaction mix.

**Example Workload Distribution**

| Scenario | Assumed % of Total Users | Avg TPS (Transactions per second) | Peak TPS |
|---|---|---|---|
| User Login (Web/Mobile) | 30% | 10 TPS | 50 TPS |
| Balance Inquiry | 40% | 15 TPS | 75 TPS |

| Scenario | Assumed % of Total Users | Avg TPS (Transactions per second) | Peak TPS |
|---|---|---|---|
| Funds Transfer (Own Bank) | 15% | 5 TPS | 25 TPS |
| Funds Transfer (Interbank) | 10% | 3 TPS | 15 TPS |
| Loan Application | 5% | 1 TPS | 5 TPS |

---

**Step 5: Sensible Hardware Sizing for NFT (On-Premises Environment)**

**Objective: Configure an On-Premises Setup That Can Handle Peak Loads**

Since **scalability is limited in on-premises**, planning must ensure:

1. **Over-provisioning of critical resources (DB, Authentication)**
2. **High-availability for failure scenarios**
3. **Separate compute, network, and storage layers**

**Proposed On-Premises Hardware Setup**

| Component | Configuration |
|---|---|
| **Load Balancer** | HAProxy / F5 LTM (2x Intel Xeon, 128GB RAM) |
| **Web Servers** | 6x Dell PowerEdge (16-core, 64GB RAM each) |
| **Application Servers** | 4x Dell PowerEdge (32-core, 128GB RAM each) |
| **Database (Oracle/PostgreSQL)** | 2x 32-core, 256GB RAM, 10TB RAID-10 SSD |
| **Redis Cache** | 3-node Redis cluster (64GB RAM each) |
| **Storage Layer** | NAS (10TB RAID 10, 1Gbps Network) |
| **Test Load Generators** | 4x JMeter Machines (16-core, 128GB RAM each) |

---

**Step 6: Baseline Tests and Controlled Load Increase**

**Objective: Establish a Reliable Baseline and Gradually Increase Load**

1. **Start with a single-user baseline** to verify API response times under **zero load**.

2. **Gradually increase users in increments**:
    - **Baseline Test:** 1-10 users.
    - **Initial Load Test:** 50-100 users.
    - **Scalability Test:** 500-1000 users.
    - **Peak Load Test:** 2500+ users.
    - **Stress Test:** Beyond expected peak loads.

**Monitoring for Performance Bottlenecks**

- **Application Metrics:** JVM Heap, API Latency, Thread Usage.
- **Database Metrics:** Query Latency, Locks, Connection Pooling.
- **Infrastructure Metrics:** CPU, Memory, Network I/O.

---

**Summary**

1. **Identify test scenarios logically based on assumed user actions.**
2. **Estimate user load patterns using time-based distributions.**
3. **Define KPIs based on transaction speed, security compliance, and availability.**
4. **Plan workload using estimated transaction mix.**
5. **Size hardware sensibly for an on-premises NFT environment.**
6. **Execute baseline tests and gradually increase load.**

📌 **Disclaimer Dictionary for Assumptions**

| Term | Definition |
|---|---|
| **Assumed Concurrent Users** | The estimated number of users active at the same time. These values are approximations and will be refined once real user analytics or business insights become available. |
| **Assumed DAU (Daily Active Users)** | A placeholder value used for initial estimations. Actual DAU should be gathered from production logs or business analytics. |
| **Estimated Peak Load** | A hypothetical assumption based on typical traffic behavior. Actual peak load may differ based on customer demographics, transaction volume, and seasonal variations. |

| Term | Definition |
|---|---|
| **Response Time Expectations** | Industry-based assumptions for acceptable response times. The final SLAs will depend on performance testing results, business needs, and user expectations. |
| **Throughput Targets** | A theoretical TPS (Transactions Per Second) value, subject to change based on real-world testing, backend optimizations, and system architecture. |
| **Hardware Sizing Estimates** | Proposed on-premises server specifications based on projected load. The actual infrastructure requirements will be determined through iterative testing and system monitoring. |
| **Workload Distribution** | The estimated percentage of users performing different transactions. This is a rough approximation that should be refined based on real transaction patterns. |
| **Load Distribution Over Time** | Assumed peak and non-peak hours derived from common usage trends. The actual distribution will be confirmed once system analytics are available. |
| **Baseline Test Assumptions** | Initial low-load testing values used to establish system behavior. These may need adjustment as more test cycles are conducted and real-world data is collected. |
| **Error Rate & CPU/Memory Thresholds** | Assumed best-practice values. Real thresholds should be defined based on actual system capabilities, monitoring, and stress test results. |



- **All values mentioned are initial assumptions** based on **generic performance testing principles and industry experience**.
- These values **should be validated and refined** based on **interactions with Business Analysts (BAs), SMEs, Developers, and actual application performance data.**
- As **more data becomes available**, the **test strategy, user load modeling, KPI benchmarks, and infrastructure sizing must be dynamically adjusted** to ensure **realistic performance expectations**. 🚀