Key Capacity Planning Formulas for Performance Testing & Engineering

Capacity planning helps determine the **maximum load a system can handle** while maintaining acceptable performance. These formulas are commonly used in performance testing and system sizing.

---

1. Little's Law (Concurrency Calculation)

🔹 **Formula:**

$$N = X \times R$$

🔹 **Where:**

- **N** = Average number of concurrent users
- **X** = Throughput (Transactions Per Second, TPS)
- **R** = Average response time (seconds)

🔹 **Example:**

- If the system processes **50 TPS**, and the average response time is **4 seconds**:

$$N = 50 \times 4 = 200 \text{ concurrent users}$$

🔹 **Use Case:**

- To estimate **how many users** can be active in the system at any time.

🔹 **Source:**

- **John D.C. Little**, "A Proof for the Queueing Formula: L = λW," *Operations Research*, 1961.
- Used in **queueing theory and system modeling** to estimate concurrency based on throughput and response time.
- **Widely applied in performance testing and system design.**

---

2. Utilization Formula (CPU/Memory/Disk)

🔹 **Formula:**

$$U = \frac{X \times S}{C}$$

🔹 **Where:**

- **U** = Utilization (%)
- **X** = Arrival rate (TPS)
- **S** = Service time per request (in seconds)
- **C** = Number of available processing units (CPU cores, threads, etc.)

🔹 **Example:**

- If a system handles **500 TPS**, each request takes **50ms (0.05 sec)**, and there are **4 CPU cores**:

$$U = \frac{500 \times 0.05}{4} = 6.25\% \text{ CPU Utilization}$$

🔹 **Use Case:**

- Helps **estimate system resource utilization** at different loads.

🔹 **Source:**

- Derived from **CPU Scheduling & Queueing Theory**.
- Similar to the **M/M/1 queue model** in Markov processes.
- **Applied in computer architecture and system performance analysis** (e.g., Hennessy & Patterson's "Computer Architecture: A Quantitative Approach").

---

3. Throughput Calculation (Max Requests Per Second)

🔹 **Formula:**

$$X_{\text{max}} = \frac{C}{S}$$

🔹 **Where:**

- **Xmax** = Maximum throughput (TPS)
- **C** = Number of available processing units (CPU, threads)
- **S** = Service time per request

🔹 **Example:**

- If a server has **8 CPU cores**, and each request takes **100ms (0.1 sec)**:

$$X_{\max} = \frac{8}{0.1} = 80 \text{ TPS}$$

- **Use Case:**
  - Helps **predict the system's throughput limit**.
- **Source:**
  - Derived from **capacity analysis principles in computer systems**.
  - **Hennessy & Patterson, "Computer Architecture: A Quantitative Approach".**
  - Used in **server sizing and system scalability planning**.

---

4. Response Time Degradation Under Load

- **Formula:**

$$R' = \frac{R}{1 - U}$$

- **Where:**
  - **R'** = New response time under increased load
  - **R** = Baseline response time
  - **U** = System utilization
- **Example:**
  - If **baseline response time = 1 sec**, and system utilization increases to **80% (0.8)**:

$$R' = \frac{1}{1 - 0.8} = 5 \text{ sec}$$

- **Use Case:**
  - Helps **predict response time under heavy load**.
- **Source:**
  - **Queueing Theory – M/M/1 Queues**, widely studied in **computer networks and operating systems**.
  - Referenced in **Kleinrock's Queueing Systems, Volume 1: Theory**.

5. Queueing Theory (Impact of Load on Response Time)

🔹 **Formula:**

$$R_{\text{queue}} = \frac{S}{1 - U}$$

🔹 **Where:**

- **Rqueue** = Response time including queuing delays
- **S** = Service time per request
- **U** = Utilization

🔹 **Example:**

- If **service time = 100ms (0.1 sec)** and system is **70% utilized (U = 0.7)**:

$$R_{\text{queue}} = \frac{0.1}{1 - 0.7} = 0.33 \text{ sec}$$

🔹 **Use Case:**

- Helps **estimate delays due to resource saturation**.

🔹 **Source:**

- **M/M/1 Queue Model in Performance Analysis**.
- Used in **server queuing models** to determine **waiting time in single-server systems**.
- **Referenced in Raj Jain's "The Art of Computer Systems Performance Analysis"**.

---

6. Scaling Formula (Vertical Scaling)

🔹 **Formula:**

$$C_{\text{new}} = C_{\text{current}} \times \frac{U_{\text{current}}}{U_{\text{target}}}$$

🔹 **Where:**

- **Cnew** = Required new CPU capacity
- **Ccurrent** = Existing CPU capacity
- **Ucurrent** = Current utilization

- **Utarget** = Target utilization

◆ **Example:**

- If the system is running **8 cores at 80% utilization**, and we need to reduce utilization to **50%**:

$$C_{\text{new}} = 8 \times \frac{0.8}{0.5} = 12.8 \approx 13 \text{ cores}$$

◆ **Use Case:**

- Helps **determine new system capacity for vertical scaling**.

◆ **Source:**

- **Derived from CPU Utilization and Resource Allocation Theories**.
- Used in **system provisioning and enterprise IT planning**.
- Referenced in **AWS Auto Scaling Best Practices & Google's Site Reliability Engineering (SRE) handbook**.

---

7. Load Distribution Across Servers (Horizontal Scaling)

◆ **Formula:**

$$N_{\text{servers}} = \frac{X_{\text{required}}}{X_{\text{per server}}}$$

◆ **Where:**

- **Nservers** = Number of servers required
- **Xrequired** = Required system throughput
- **Xper server** = Throughput per server

◆ **Example:**

- If total required TPS is **5000**, and each server handles **1000 TPS**:

$$N_{\text{servers}} = \frac{5000}{1000} = 5 \text{ servers}$$

◆ **Use Case:**

- Helps **determine how many servers are needed**.

🔹 **Source:**

- Used in **distributed computing & cloud scaling strategies**.
- **Referenced in Amazon Web Services (AWS) Well-Architected Framework** & **Google SRE Handbook**.

---

8. Memory Capacity Planning

🔹 **Formula:**

$$M_{\text{required}} = U_{\text{sessions}} \times M_{\text{per session}} \times F$$

🔹 **Where:**

- **Mrequired** = Total memory required
- **Usessions** = Number of concurrent sessions
- **Mper session** = Memory per session
- **F** = Safety factor (e.g., 1.2 for 20% buffer)

🔹 **Example:**

- If each session needs **20MB**, there are **500 concurrent sessions**, and we apply a **20% buffer**:

$$M_{\text{required}} = 500 \times 20MB \times 1.2 = 12,000MB \text{ (12GB)}$$

🔹 **Use Case:**

- Helps **estimate memory requirements** for high-concurrency workloads.

🔹 **Source:**

- **Derived from memory allocation theories in operating systems.**
- Referenced in **"Operating System Concepts" by Silberschatz, Galvin, and Gagne**.
- Used in **Kubernetes, JVM tuning, and database scaling**.

How to Use These Formulas?

1. **Use a Practical Scenario** → "We needed to estimate the concurrent users for our application..."
2. **Choose the Right Formula** → "Using Little's Law, we estimated concurrency as N = X × R..."
3. **Give a Realistic Example** → "With 500 TPS and a 4s response time, our system handled ~200 concurrent users."

These formulas **show technical depth** and **help in capacity planning for real-world systems**. 🚀