🧠 Throughput vs Hits/sec: A Performance Engineer's End-to-End Diagnostic Playbook

🧭 1. Conceptual Deep Dive

🔄 Hits/sec (Requests per Second)

Definition: Total number of HTTP(S) requests sent from client to server **per second**, across all concurrent threads/users.

🔧 Internals:

- **Generated by Tool** (JMeter, Gatling, LoadRunner).
- Includes:
  - HTTP requests (GET/POST/etc.)
  - Static resources (JS, CSS, images) if embedded resource parsing is enabled.
- May include **retries**, **redirects**, **polling**, **AJAX calls**.

💡 Important Points:

- **Not equal to TPS (Transactions per second)** unless:
  - One transaction = one request.
- **Can be inflated**:
  - Due to auto-retries.
  - Recursive calls inside samplers (e.g., recursive polling).
  - Load tool misconfigurations (no pacing, infinite loop).
- **Affects system CPU**, but not necessarily bandwidth.

📦 Throughput (Bytes per second / MBps / Mbps)

Definition: Actual volume of data transferred per second **between server and client**, generally in kilobytes/sec or megabits/sec.

🔧 Internals:

- Data includes:
    - HTTP response body.
    - Headers.
    - Redirected payloads.
    - TCP overhead (sometimes counted).
- Can be measured:
    - From client side (JMeter): bytes received / sec
    - From server side (web server logs, NIC counters)
    - From OS level (using tools like iftop, nload, netstat, ss, sar -n DEV)
    - At CDN or Load Balancer

🧠 **Implication:**

- Reflects **I/O stress**, **network saturation**, **serialization cost**, and **response payload impact**.
- Used to calculate:
    - Bandwidth planning.
    - CDN sizing.
    - Socket throughput per node.
    - Network chokepoint diagnosis.

⚙️ **2. Calculations**

🧮 **Hits/sec**

Hits/sec = Total HTTP requests / Test duration (in seconds)

🧮 **Throughput (JMeter-based)**

Throughput (KB/sec) = (Total Bytes Received + Total Bytes Sent) / Test Duration (sec)

🧮 **Bandwidth required**

Bandwidth (Mbps) = (Throughput KB/sec * 8) / 1024

Example:
100 users sending 10KB per request, 2 requests/sec per user =
Throughput = 100 users * 2 req/sec * 10KB = 2000 KB/sec
= ~15.6 Mbps bandwidth

🔍 3. Tooling View: JMeter

Hits/sec Monitoring

- Aggregate Report → Samples/sec
- Summary Report → Throughput (requests/sec)
- Backend Listener → Time-series of request rate
- **Visuals**: Hits per Second listener (plugin), InfluxDB + Grafana

Throughput Monitoring

- **Bytes Received/sec**, **Bytes Sent/sec**
    - Summary Report
    - Throughput Shaping Timer (for model-based pacing)
- PerfMon Server Agent (plugin) → OS-level network bytes
- Server logs (Apache/Nginx access.log) → Real KB transferred

---

🧪 4. Typical Patterns & Engineer's Diagnostic Flow

🧵 Scenario A: High Hits/sec, Low Throughput

- 🔍 **Symptoms**:
    - High API request rate, but server bandwidth low
    - Low CPU/IO
- 🧠 **Engineer's Thinking**:
    - "Payloads are small — probably health checks or small JSON"
    - "Client sending too many requests per second — check pacing"

- “Are these real-user scenarios or synthetic loops?”

- ✅ Action:
  - Validate each request’s size with Save Responses to a file
  - Add Transaction Controller to model real user flows

📦 **Scenario B: Low Hits/sec, High Throughput**

- 🔍 **Symptoms**:
  - Few requests/sec, but server NIC shows > 100 Mbps
- 🧠 **Engineer’s Thinking**:
  - “Likely heavy download API — media files, PDFs, bulk DB exports”
  - “Response time may be high due to large payload”
  - “Is server compression enabled (gzip)?”
- ✅ Action:
  - Inspect response size per API in listener
  - Use bmon or iftop to validate per-connection bandwidth

🚨 **Scenario C: Throughput Drop, Hits/sec Steady**

- 🔍 **Symptoms**:
  - Hits/sec flat, throughput suddenly dips
- 🧠 **Engineer’s Thinking**:
  - “Server sending smaller responses — maybe errors (500s)”
  - “Downstream services (DB/cache) failing — returning fallbacks”
  - “Rate limiting? Circuit breaker triggered?”
- ✅ Action:
  - Inspect error codes (Summary Report, listener)

- Enable View Results Tree for a sample run
- Check backend logs / Splunk for time-matched events

🧱 5. Infra & App Correlation

Metric	What it Impacts	Root Cause Clues
Hits/sec ↑, CPU ↑	Load-gen CPU or app CPU	Thread saturation, CPU-bound logic
Throughput ↑, IO ↑	App IO subsystem	File read/write, DB BLOB fetches
Throughput ↓, Hits/sec ok	Payload issue or error fallback	204/500s instead of real content
Both ↓	App bottleneck	GC pause, DB connection limit, deadlock

---

🛰️ 6. OS & Network Side Checks

🖥️ Server:

- iostat -dx 1 → Disk I/O
- vmstat 1 → System wait time
- top/htop → CPU steal/wait
- ss -s / netstat -s → TCP retransmits, established connections

🌐 NIC / Network:

- iftop, bmon, nload → Real-time network usage
- sar -n DEV 1 → Interface level throughput
- ethtool -S eth0 → NIC buffer drops
- MTU/fragmentation checks (ping -s 1472 -M do <ip>)

🧩 7. Modeling Impacts

Impact of Think Time:

- Increases spacing between requests
- Reduces both hits/sec and throughput
- Simulates real user behavior

Impact of Payload Compression:

- Throughput reduces significantly
- CPU usage increases on both ends
- Evaluate with gzip off/on (Content-Encoding: gzip)

Impact of Concurrent Connections:

- **Hits/sec limited by concurrent sockets**
- Throughput limited by **TCP window**, **congestion control**, **bandwidth delay product**

🧪 8. What a Senior Perf Engineer Always Checks

Metric/Graph	Reason
📈 Hits/sec	Load generation behavior
📦 Throughput	App/network behavior
🧠 Response Time	Capacity under pressure
🔥 Error Rate	Application stability
❓ Backend logs	DLQ, timeouts, 5xx tracing
🧵 Thread Count vs Active Threads	Thread leakage
📡 Server NIC usage	Bandwidth choke analysis
🔁 GC / Heap	Memory pressure / leaks
⛔ Retry spike	Downstream failure compensation

✅ **Final Thoughts: Decision Tree**

if (hits/sec ↑ && throughput ↓):

-> validate payload size & content

-> check if cache returns empty or static response

if (hits/sec ↓ && throughput ↑):

-> likely large payloads, download-heavy flows

if (both ↓ but no errors):

-> GC pause, network saturation, DB wait

if (throughput flat at max bandwidth):

-> infra saturation; scale horizontally or optimize payload
