Need to Scale: Estimates

Thought Process

- For each microservice
  - Check whether **each tier needs to scale**
  - A deterministic set of reasons can be posed across all interviews
    - Need to scale for storage (storage and cache tiers)
    - Need to scale for throughput (CPU/IO+network)
    - Need to scale for reducing API latency
    - Need to remove hotspots
    - Availability and Geo-distribution
  - The constraints or numbers change from problem to problem
  - Solve algebraically first and then put numbers
    - Algebraic solution is same for all problems
  - If dataplane phase takes more time,
    - do not spend time on calculations/estimations here u**nless asked for** and get out fast

Storage

- Storage: One server cannot hold all data
  - Size of a single K-V pair: **B**: comes from your dataplane proposal
  - Number of lifetime K-V pairs: **A**: comes from interviewer or guesstimate
    - Number of **new** K-V pairs inserted /sec: **C**:
    - **Storage : (A*B) or (C* number of seconds in a couple of years * B) (capacity plan for a few years) or (C*TTL in sec*B) (if data has TTL), min of all 3**
    - With replications: number of servers = number from above * replication factor
    - Memory is similar to persistence
      - What changes is amount of data to be stored in memory
      - What changes is the data structure
    - A typical commodity 8-core server -> 1-2TB of storage, 128GB of RAM memory
    -

CPU throughput

- CPU throughput
  - How many API calls per second system needs to handle: **Y:** comes from interviewer or guesstimate
    - API1 : 10000/s, API2: 5000/s: API3: 2000/sec: Y= Y1+Y2+Y3 = 17000/sec
  - Let's say that the latency (processing time) of an API in any server in a single thread is **X ms** (weighted avg of P50, P90, P99):
    - API1 : 10 ms, API2: 20ms, API3: 30 ms
    - X = (10000/17000)*10 + (5000/17000)*20 + (2000/17000)*30
  - Number of APIs handled by a single thread = 1000/X ops/sec
  - Number of concurrent threads in the server: in a commodity server (4-8 cores) : 100-200 application threads: Z
  - 1000*Z/X ops/sec per server (assuming no hotspots, best case)
    - Assuming max hotspot, 1000/X ops/sec: if banging on the same data structure
  - 30-40% (operate at resting pace)
  - (300-400)*Z/X ops/sec from one server = T per server
  - Y divided by T = number of servers
    - K-V workloads:

IO Throughput

- Sequential IO throughput
  - A single server can provide 100-200 MBytes/sec I/O throughput (medium is spinning disk) = Z
  - A single server can provide 1-2GBytes/sec I/O throughput (medium is flash SSDs) = Z
  - Let’s say the total I/O throughput required from your system = Y MB/s
  - Number of servers = Y/Z

Contd

- Availability
  - I should be able to return results 99.999% of time
- Geo-location
- API parallelization
  - For some problems, especially when APIs are bulky, large response times
  - Let's say total time taken by API in single thread is X = 1 hr
  - Client requires time to taken per API to be Y = 10 minutes
  - Number of parallel threads required= X/Y = Z'
  - How many threads in a single server: Z (comes from experiments)
  - Number of servers = Z'/Z

Generic tips

Number of writes per second: when human generates a workload: thousands to tens of thousands/sec

Number of reads per second in a read heavy system: hundreds of thousands/sec

Number of writes per second: when system is generating: millions per second