Performance

- How well resource requirements of an application are being met.
- Expressed in terms of latency, memory, bandwidth and (for mobile apps) battery consumption metrics.
- Usually apps involving playout of multimedia content, 2D/3D graphics/animation, AR/VR and data analytics etc. impose resource requirements on the environment.
- Performance Testing involves measuring and/or profiling latency, memory, bandwidth and battery consumption of an app under normal working conditions.
  - Involves use of system calls and/or profilers provided by the software development kits.
  - e.g.SystemClock, System.nanoTime(), Hierarchy Viewer, Debug, Traceview, Runtime, Network Profiler, BatteryManager

Scalability

- Scalability is how performance trends as various system, environment or contextual factors change.
- Scalable architectures aim at ensuring that the application either continues to perform well as these factors stress the application or at least degrades gracefully without resulting in system failure when the system is stressed.
- Conversely, a scalable system is one whose performance improves in proportion to the resources being added.

Scalability Testing

- Load Testing involves incrementing the load progressively and monitoring the resulting trends in the performance in terms of latency or throughput.
- Commonly used system scalability/load testing tools include LoadRunner, JMeter.

Scalability Models

- The increase in computational latency or memory requirements of an algorithm, scheduling scheme or data structure as the size of the input increases is typically expressed using O notation. O(1), O(log N), O(N), $O(N^2)$, O(N!) and $O(N^N)$ represent a deteriorating trend in performance as the input size N increases.
- Queuing models are used for representing statistical trends in the performance in response to a random input process and predict impact of load balancing and scalability of architectures.

Note: The PhotoGallery app is not a queuing system and therefore the appropriate scalability or load test would be the performance of the location, time or keyword search as the number of pictures in the storage increase.

Queuing Models

M/M/1 queue

- For an M/M/1 queue, the average time a request/transaction spends in the system is given as follows:

$$E[T_{system}] = 1 / [\mu - \lambda],$$

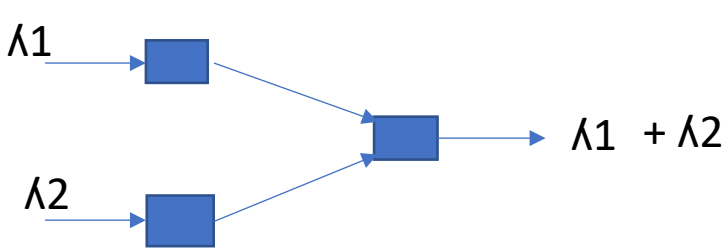
where λ is the average rate at which requests/transactions arrive at the system, μ is average service rate (inversely proportional to the average request/transaction time/length), under the condition that $\mu >> \lambda$.

The arrival process is assumed to be Poisson distributed and the service process is assumed to be exponentially distributed.

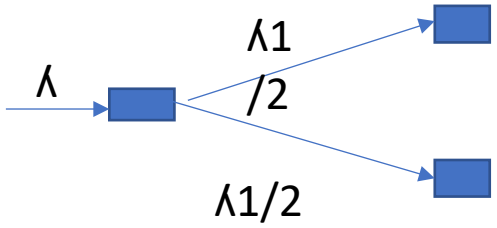
- Example: Assuming that a component that could be modelled as an MM1 queue in a system, can process 10 requests/transactions per second on the average, and the average arrival rate of the requests to the component is 3 requests per second, the average latency or time it takes for a request to be handled by the system is then

$$1/[10-3] = 0.14 \text{ seconds.}$$

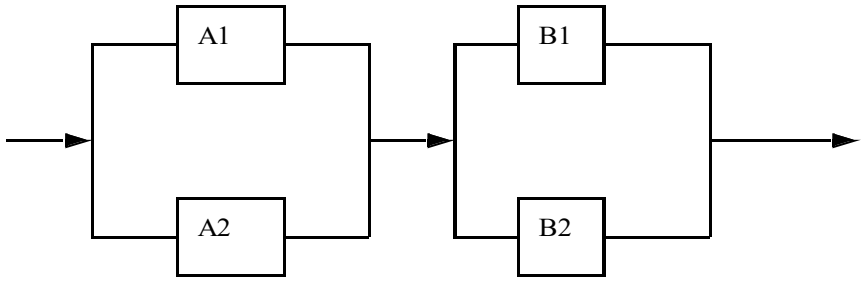
- For Poisson arrival processes and exponentially distributed service processes, the following relationships hold:



and



- The average time a request spends in a system that is composed of multiple such queues, arranged in tandem, is simply the sum of the average times at each queue, as long as the number of queues in such network stays small. The two M/M/1 queues of the previous example connected in tandem will thus have E[Tsystem] = .14 + .14 = .28s
- Example:



- Assuming that each component, in the above system, on the average can process 10 requests/transactions per second and the average arrival rate of the requests to the system is 3 requests per second, the average latency or time it takes for a request to be handled by the system assuming that the system could be modeled as a network of queues is estimated as:

  1/[10-1.5] + 1/[10-1.5] = 0.25 seconds.