Concurrent User Modeling

An Alternative Approach to Classic Queuing Theory

Xiaosong Lou

Concurrent Users

Concurrent User is one of the key performance metrics in a system. Many performance issues that can only be exposed under load are related to the increased number of concurrent users.

In the context of load test, we often hear phrases such as “test the system under x concurrent users”. This is a common misunderstanding of the term.

In most cases, Concurrent User is a Random Variable.

We propose an analytical alternative to the classic queuing theory for estimating Concurrent User. This model helps us determine whether the simulated workload is a proper representation of the expected production scenario.

Performance Issues Triggered by Concurrent Users

- ❖ Queuing
  - ➢ Queuing occurs when number of concurrent requests in the system exceeds total available resources to serve them. It directly affects service time.
- ❖ Resources
  - ➢ Concurrent requests in the system requires more caches, buffers, file descriptors, thread pools and other temporary resources.
- ❖ Deadlocks and Race Condition
  - ➢ As the number of parallel resources access increases, the chances of rare and unforeseen state changes increases, and cause problems sometimes hard to reproduce.

Problematic Workload Simulation

Many people setup the simulation with throughput as a target, while concurrent user states are completely ignored.

Many tools are programed to default to the easiest way of achieving the targeted throughput, while ignoring the distribution of concurrent users.

Another common misconception is “stress test”. In many cases, people would increase the throughput by cutting down or completely eliminating think time in a workflow, thereby achieving the desired “stress level”. We prove that such an approach is not sending the proper workload to the system.

Classic Queuing Model

For M/M/C queue

$$p_0 = \left[\left(\sum_{k=0}^{c-1} \frac{(cp)^k}{k!}\right) + \frac{(cp)^c}{c!} \frac{1}{1-\rho}\right]^{-1}$$

$$p_k = \begin{cases} p_0 \dfrac{(cp)^k}{k!}, & 0 < k < c \\ p_0 \dfrac{(cp)^k c^{c-k}}{c!}, & c \leq k \end{cases}$$

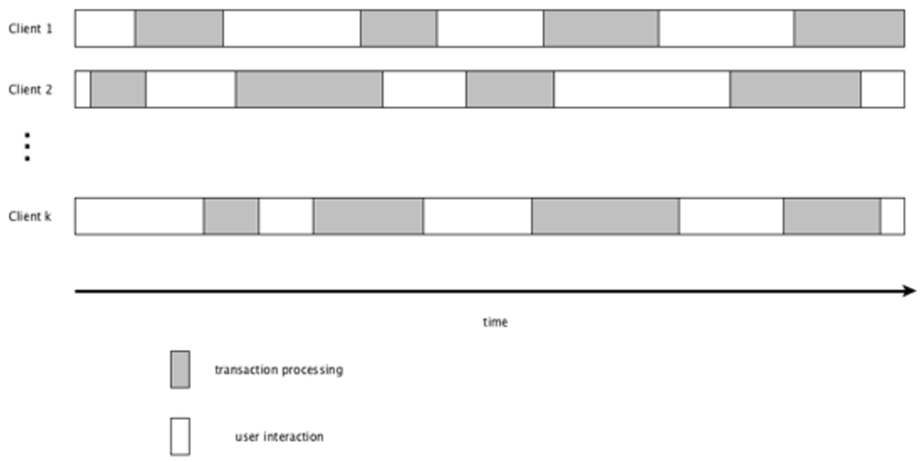
In classic queueing models, the Concurrent User is represented by state probabilities in Markov Chain.

There are many queuing models based on arrival and departure processes, number of servers, limits of queue depth, scheduling strategy etc.

The analytical solutions for the state probabilities are often long and seemingly prohibitive formulas. We believe this has in part contributed to the fact that many test engineers don’t use them.

Most of these models require that arrivals are independent.

Modeling Concurrent User from the Server



Alternatively, we model the concurrent user from observing server states.

Instead of assume all arrivals are independent and identically distributed, we only require that all clients are independent. Each client could have multiple requests that are dependent of each other.

When user interaction (think time) >> transaction processing, this model approximates classic M/M/C/K model.

When think time approaches zero, this model approximates non-independent arrivals

Binomial Approximation

- Individual client arrival x_i follows Bernoulli distribution

$$\begin{aligned}&\Pr(x_i = 1)\\ &= \Pr(\mathit{server\ is\ processing\ request\ from\ client\ } i)\\ &= \frac{t}{T}\end{aligned}$$

- Number of Concurrent Users N follows Binomial Distribution

$$\Pr(N = i) = \binom{k}{i}\left(\frac{t}{T}\right)^{i}\left(1 - \frac{t}{T}\right)^{k-i}$$

Converges to Normal Distribution when N is sufficiently large

According to Central Limit Theorem, when k is sufficiently large, N converges in probability to normal distribution

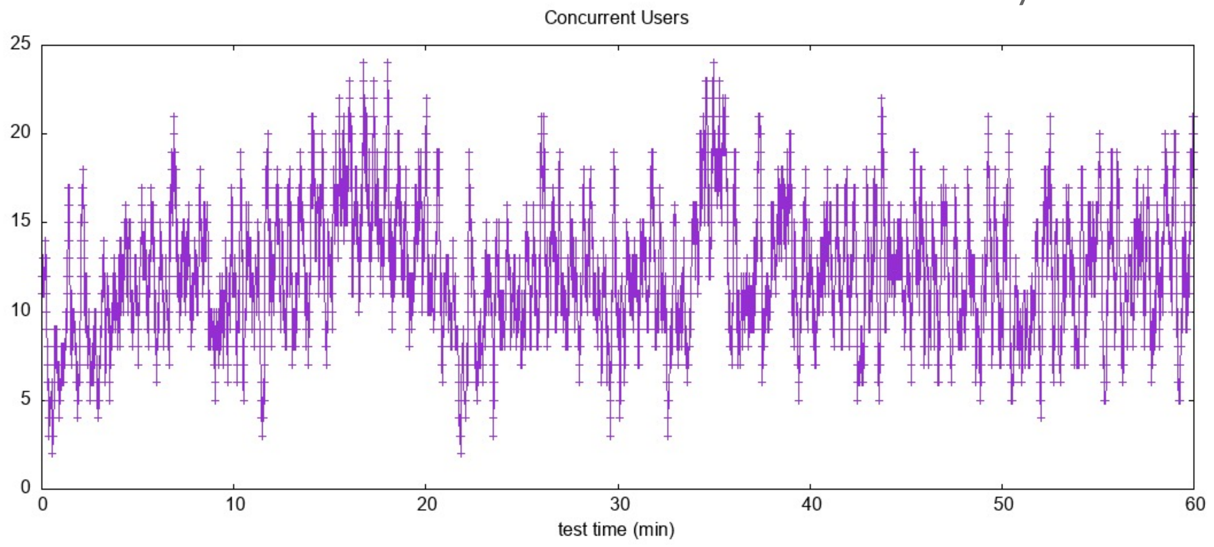
$$\Pr(N = i) \approx \int_{i-1}^{i} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(t-\mu)^2}{2\sigma^2}} dt$$

where $\mu = \frac{kt}{T}$ and $\sigma = \sqrt{\frac{kt}{T}\left(1 - \frac{t}{T}\right)}$

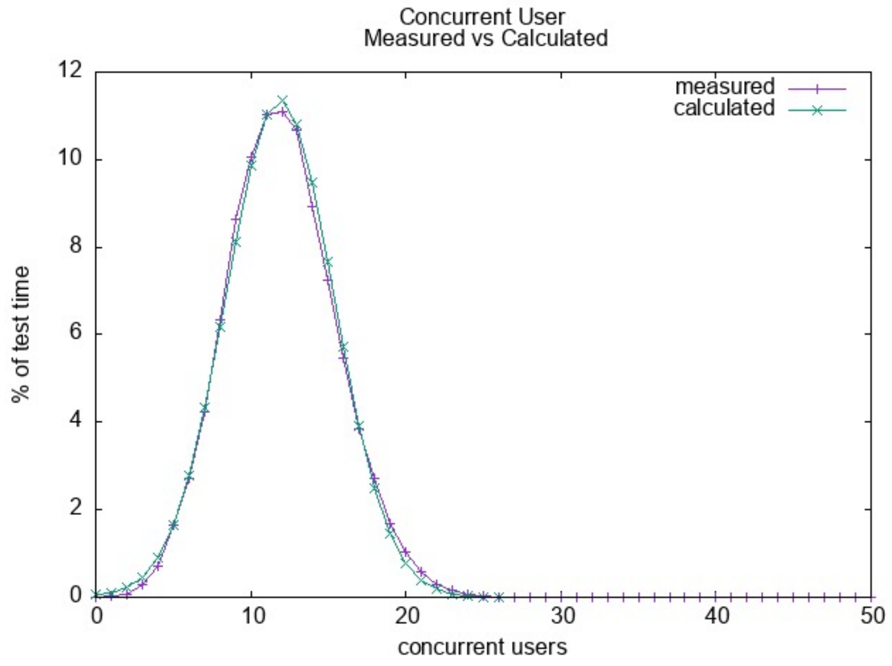
Actual User Scenario

- 1000 clients
- Average response time is 188ms
- Average think time 15 seconds

Concurrent user is a random variable. We record the actual number of concurrent users in the system over the duration of our test.



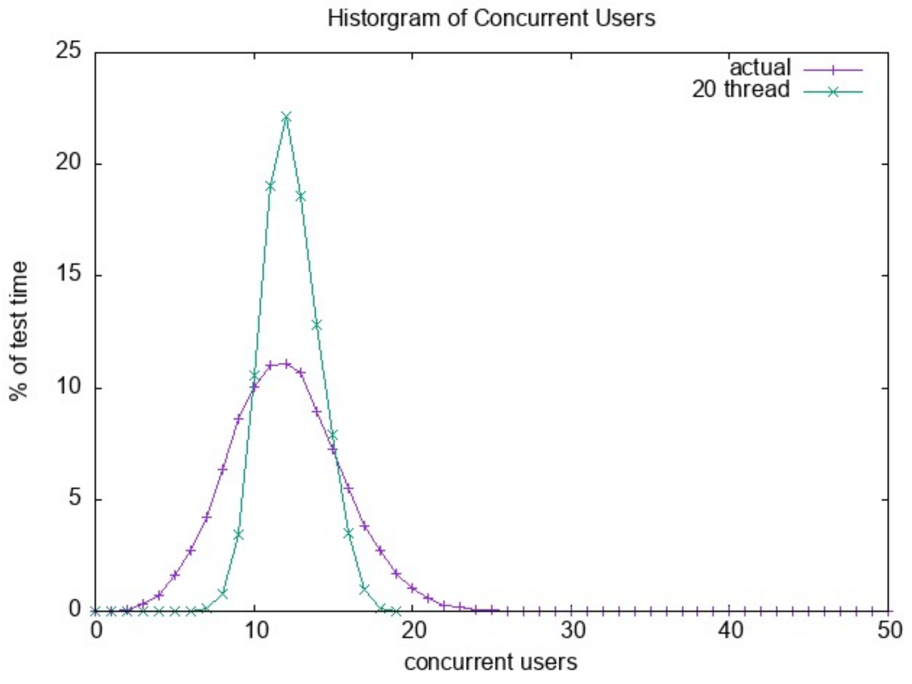
Compare modeling result with actual measurement



Our model suggests that the number of concurrent users follow normal distribution with average = 12.38 and standard deviation = 3.497

After two layers of approximation, our calculation looks reasonably close to the actual measured distribution of Concurrent Users in the system.

Stress test done wrong

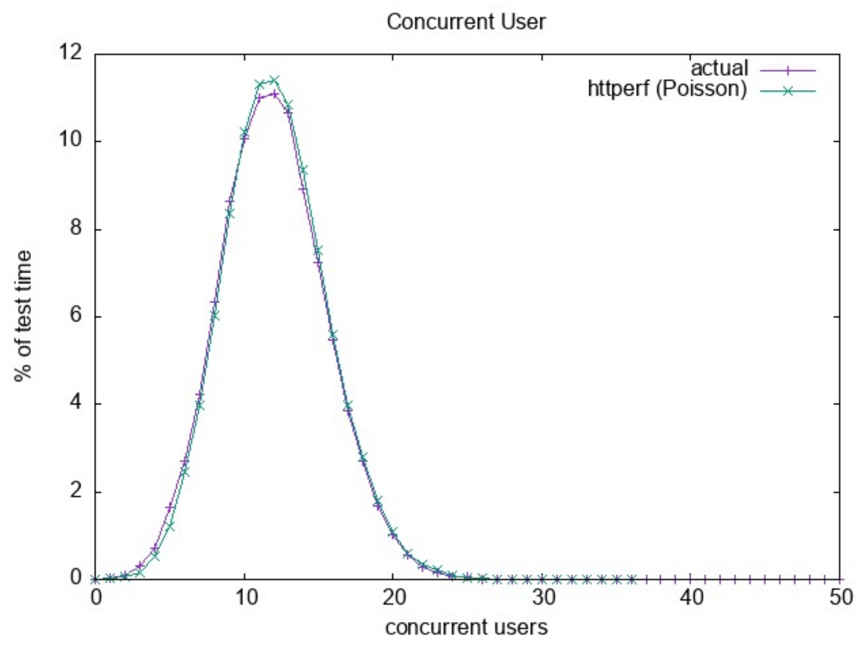


One common approach for “stress test”, is to reduce or remove think time in simulation so that a single load generator thread can send more requests to the server.

This is a common mistake. Because by reducing think time, the resulted distribution actual concurrent users vary drastically compared to the system under actual stress.

In this case, by shrinking think time to 120ms, we only need 20 threads to get the same throughput. However, the resulted distribution of concurrent users is much more concentrated around the mean value, and much less of high-concurrency stress is being tested.

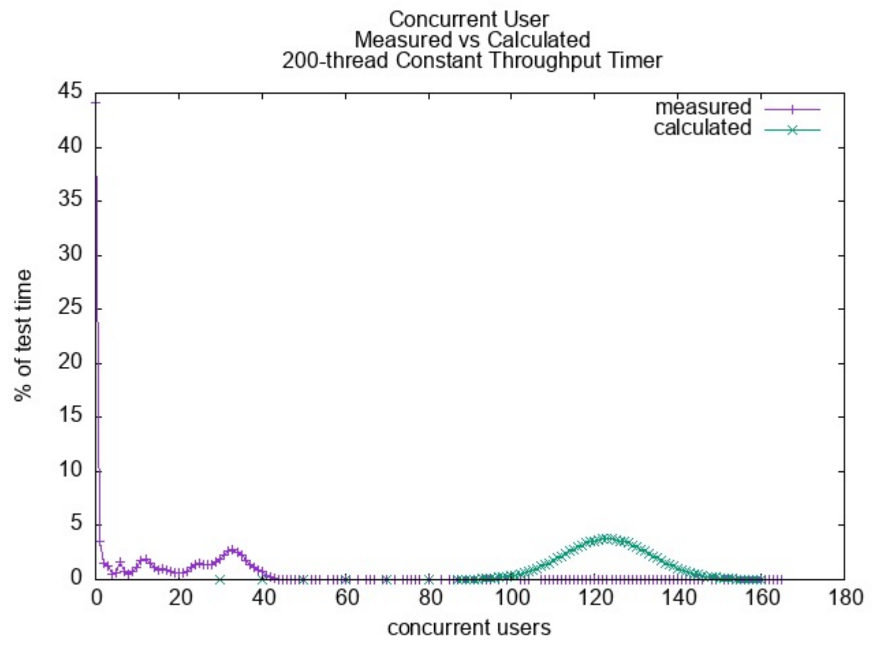
Classic Poisson Arrival



In our actual load scenario, the 1000 clients are independent. Furthermore, because of the large think time between requests within a single client compared to response time, this workload is a perfect example for Poisson arrival process.

Using a load generator that can simulate Poisson arrivals, we see that not only the throughput is properly simulated, concurrent users are also very accurately simulated.

When this model doesn't work



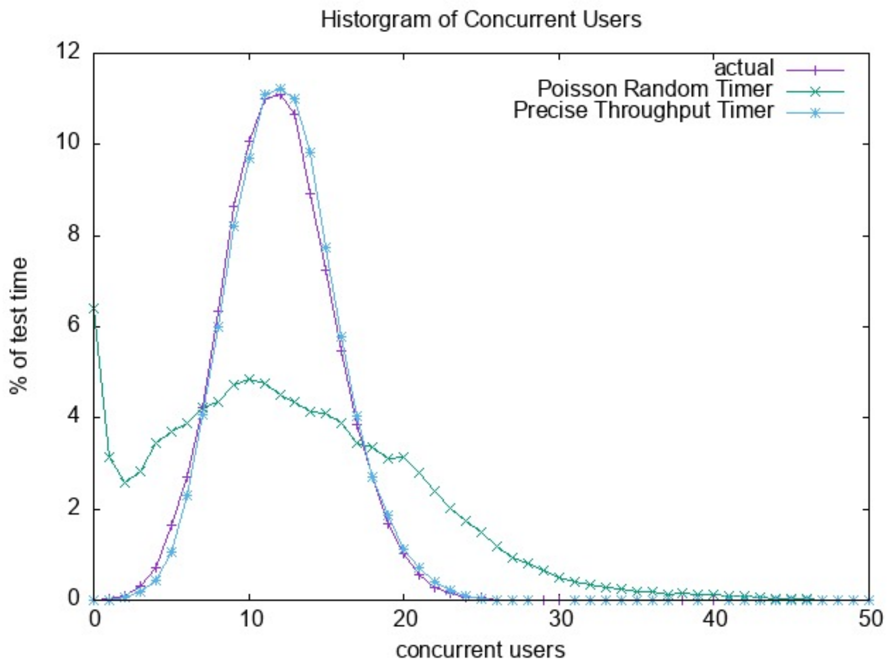
In previous example, we show that the proposed model is accurate in describing the distribution of concurrent users when each clients are independent of each other.

When the clients are dependent of each other, this model doesn't work.

There is "Constant Throughput Timer" in Jmeter. Under this setting, with 200 threads, we achieve the same throughput. However concurrent user distribution doesn't fit the model at all.

Possible explanation: All 200 users are in sync and requests come in waves.

Don’t be fooled by load generators



Jmeter is probably the most popular free load generator. However, most of the settings in Jmeter fails to ensure a proper simulation of concurrent users.

Jmeter actually has a “Poisson Random Timer”, but it doesn’t simulate Poisson process. Instead, it sets think time to follow Poisson distribution. This approach makes the resulted simulate very different from the expected concurrent user distribution.

The “Precise Throughput Timer” in Jmeter simulates Poisson process, if there are suffice threads so that among these threads, the requests can be considered “independent and identically distributed”

Summary

- We use Normal Distribution to model Concurrent User
  - It is a more familiar distribution
  - Less restrictive than classic queuing model
- We show that our model is equivalent to queuing model when applicable
  - When requests are independent and identically distributed, both Poisson model and our proposed Normal distribution are very close to the actual measurements
- There are many ways of improper simulations
  - Not setting think time in a close-network simulation is one of the most common mistake
  - Make sure you understand what the tool is doing. Don't be fooled by the name of the settings.