Performance Testing Strategy and Implementation

External Pressures that Start the Conversation

- Sales has told you they have clients in the pipeline and need to make sure they can be handled
- Existing clients are asking for more
- Need to determine if the current infrastructure can handle being reduced without affecting users
- **Site went down due to load and no one knows what would keep it from failing again except adding more hardware**
- Transitioning from an old system to a new one
- *Start trading bitcoin futures…(CBOE) ;)*

First Tendencies

- **Solve with hardware**
- Ask everyone in the office to use the system and see what happens
- Let it ride and only react if the system fails
- Find some way to put load on a couple endpoints at random and call it a day
- Test individual endpoints in isolation
- Blame that thing that is always a problem. The DB is always slow, language is slow…but do not do anything.

What Should You Care About?

- Remember why you are in business...it is your customers
- How does a typical user browse your site?
- What does the typical daily usage look like?
- How predictable will be the load increase be?
- What services are in use/how easy is it to scale up/down?
- Does your system understand the difference between test data and real data?

- **Convince the leadership by showing the site can go down with X users.**
  - Ideally this can be done on the production environment off-hours or during a pre-determined maintenance window.

Setting Expectations

- Performance testing is NOT functional testing
- You need support from developers and devops
- This is not a one time thing, if you care about it now, it will continue to be a point of concern
- **Functional testing gets all the glory, but functionality does not matter if it does not scale**
- *Someone has likely tried to performance test in the past*
  - Find out why it was unsuccessful.
- Explaining what metrics to care about and what it means to fail performance testing
- Need of an environment that is *production-like* is important for quickly iterating on performance

Types of Performance Testing

- **Load** - Load testing is performed to determine a system's behavior under both normal and anticipated peak load conditions
- **Stress** - activity that determines the robustness of software by testing beyond the limits of normal operation
- **Soak/Endurance** - involves testing a system with a typical production load, over a continuous availability period, to validate system behavior under production use

Identifying Your Typical User

- **Data needs to be mined to find out how most users go through the site**
  - Google analytics, DB timestamps, tracking/logging of any kind
  - Duration on pages
  - Order of browsing

- **<u>This is the single most important step in performance testing!!!</u>**

Measurements

- Average, 90th, 95th, 99th percentile response times
- Key is to pick one and stick with it to ensure apples to apples
- Servers are temperamental and can **vary by up to 20% each run**. This makes it harder to identify performance issues, but is a point of education and consideration when analyzing results.
- Error rates on the calls.
- Transaction rates requests/sec
- **I went with <3s response time and <5% error rate.**

The Script

- **Hardest part is always authentication**
- Consistency in the script between runs is vital
- Tool is not as important as you might think
- Image/Asset downloads are not really a concern (unless you are Google) and can be made a non-issue through use of CDNs
- Need to only run off-hours if performing the test in production
- Often hard to produce the amount of load needed with just one computer
- Tools: JMeter, Gatling, Locust, etc.

Monitoring

- Identify what monitoring is already in place and build on it
- Logs that include timing
- Focus on transactions that are known or “feel” like they might take time
- APM (Application Performance Management) is helpful when starting from nothing. Will point you in the right direction of what to focus on for low hanging improvements.
  - Examples: New Relic, Data Dog, Dynatrace, etc.

After Test Run

- **Gather the developers and devops to review the results**
- Talk about the type of run that happened
- Assist with any input in the logging/output of the performance testing tool
- Typical culprits: Reflection, queries, missing indexes, web server settings, firewalls (these types of tests will set off alarms), ELB ramp-up, etc.
- **Identify gaps in monitoring and create alerting**

It Works, Now What?

- Performance problems have a nasty habit of showing up when code is changed
- Treat it the same as functional test automation. Regular updates, affects the decision to release code to production, listed as a consideration on new functionality
- **Stand firm on the performance of the application as it now is just as important as the quality of the features being delivered**