Waar moet je aan denken bij het opzetten van een performancetest? Wat komt daar bij kijken en hoeveel voeten heeft dat in de aarde? Die vragen zijn niet altijd direct en gemakkelijk te beantwoorden. Om een performancetest uit te kunnen voeren, zijn er de nodige randvoorwaarden waaraan voldaan moet worden. Deze randvoorwaarden noem ik **de zes pijlers onder performancetesten**.

Wat zijn de 6 pijlers onder performancetesten?

De zes pijlers zorgen er samen voor dat een performancetest tot een succes wordt. Je kunt dus ook niet zeggen dat de een belangrijker is dan de ander. Het is van belang om bij de opzet van een performancetest alle pijlers in het oog te hebben.

Wees ervan bewust dat de keuze voor de ene pijler van invloed kan zijn op een andere. Sommige vullen elkaar aan en de ene keer is de een belangrijker dan de ander en de andere keer weer andersom. Dit heeft alles te maken met welk doel je voor ogen hebt. Wil je een eenmalige test voor live-gang als check, of wil je performance testen inbedden in het ontwikkelproces?

Wat je ook wilt, je zult altijd te maken krijgen met de volgende **6 pijlers onder performancetesten**:

- Testdata
- Testomgeving
- Testscripts
- Load profiel
- Test analyse
- Test tools

Testdata

Waarom is bij performancetesten de testdata belangrijk? Daarvoor gaan we ons eerst de vraag stellen: wat is testdata bij performancetesten eigenlijk? Bij performancetesten heb je grofweg met twee kanten te maken. Aan de ene kant is er de data die er nodig is voor het afspelen van scenario’s en aan de andere kant de aanwezigheid van productie-like data in het systeem.

Bij het naspelen van real-life scenario’s kun je er al snel tegenaan lopen dat je niet genoeg hebt aan een paar testgebruikers. Vaak heb je honderden of zelfs duizenden gebruikers nodig. Bovendien mogen dat niet allemaal nieuwe gebruikers zijn, maar moeten er ook gebruikers zijn met een geschiedenis. In de meeste situaties heb je niet een gebruiker op een systeem die altijd hetzelfde doet. Een gebruiker komt misschien maar één keer per dag langs. Elke gebruiker heeft een andere geschiedenis of andere rechten op het systeem en

dus worden er ook andere zaken opgehaald en getoond. Om ervoor te zorgen dat er voldoende testdata is moet er dus goed nagedacht worden over welke data er nodig is voor elke situatie.

Om gebruikers te hebben met een geschiedenis, heb je een behoorlijke databasevulling nodig. Dat is op zich logisch. Er is nog een reden voor een goed gevulde database. Een lege database reageert anders dan wanneer deze gevuld is. Dus de database moet niet alleen vulling bevatten voor het kunnen uitvoeren van de testscenario's, zoals vaak op een testomgeving het geval is, maar ook zoveel gevuld zijn, dat de grootte van de database vergelijkbaar is aan productie.

Bij het opzetten van een performancetest moet je dus rekening houden met hoe de testdata ingericht gaat worden, en bij het gebruik van dit soort grote datasets hoe dit het beste beheerd kan worden. Het beheer van testdata kan behoorlijk wat voeten in de aarde hebben, aangezien het vaak niet eenvoudig is om grote hoeveelheden testdata te genereren of te kopiëren.

Testomgeving

In principe kun je elke omgeving gebruiken voor een performancetest. Dit is alleen sterk afhankelijk van het doel van de performancetest. Wil je echter iets kunnen zeggen over de verwachting van de performance op productie, dan zit je al vrij snel vast aan een productie-like omgeving. Wat dan vaak gebruikt wordt is de acceptatieomgeving. Op zich is dat prima, mits dit een productie-like omgeving is.

Wat is dat een productie-like omgeving? Een van de belangrijkste eigenschappen van een productie-like omgeving is, is dat de architectuur moet overeenkomen met productie. Dit houdt in dat je ervoor moet zorgen dat alle lijnen tussen de systemen hetzelfde lopen, de verdeling van de componenten gelijk is en de verhouding van machines gelijkwaardig is. Dat is essentieel als je op zoek bent naar bottlenecks en op zoek bent naar de verwachting van de performance in productie.

Naast dat de testomgeving qua architectuur gelijkend moet zijn aan productie, is ook de software van belang. De testomgeving moet niet alleen stabiel zijn, want een performancetest is er niet om functionele fouten op te sporen, maar hij moet ook draaien op de juiste versies van de gebruikte software. In dit geval gaat het niet alleen om wat de gebruikers gebruiken, maar ook om de software die gedraaid wordt op de servers, zoals bijvoorbeeld de juiste Java versie. Verschillende versies kunnen andere karakteristieken vertonen, welke van invloed kunnen zijn op het cpu-gebruik of het geheugenverbruik. Hierdoor verschuiven bottlenecks en kan de performance verschillen.

De keuze van de omgeving voor performancetesten kan dus een behoorlijke impact hebben

op dat wat er nodig is. Het is daarom van belang om in een vroeg stadium duidelijk te hebben wat de eisen zijn die gesteld worden aan de omgeving.

Testscripts

Testscripts zijn er om de gedefinieerde scenario's te kunnen nabootsen. De scenario's die worden gebruikt zijn zogenaamde "happy flows". Dit zijn de scenario's die door de gebruikers van het systeem het meest uitgevoerd worden. Dus vraag je af en ga op onderzoek uit om te ontdekken wat 80 tot 90% van de gebruikers op het systeem doen. Deze "happy flows" zijn een belangrijk onderdeel van de test en worden in testscripts nagebootst. Het belangrijkste van een testscript is dat je de stappen kan naspelen van de "happy flow".

> **Wat doen jouw gebruikers eigenlijk allemaal op je systeem?**

Naast het afspelen van "happy flows", kan het zijn dat er nog andere scenario's zijn die niet mogen ontbreken in een testscript. Dat zijn de scenario's waarvan de verwachting is dat deze veel resources van het gebruikte systeem claimen. Hierdoor hebben deze scenario's een onevenredig grote impact op de performance van het systeem en mogen daarom ook niet ontbreken tijdens de uitvoer van een performancetest.

Speciaal aan performance testscripts is, dat deze, indien mogelijk, op protocol niveau worden gemaakt. Dit is nodig om met weinig resources veel load te genereren. Je hoeft dan namelijk niet een browser per gebruiker te starten. Het starten van een browser en op browser niveau testen, wordt alleen gedaan om client-side performance testen te doen, of als het niet mogelijk is om op protocol niveau te scripten. Om op protocol niveau een script weer goed te kunnen afspelen is het ook nodig om na te gaan of er bepaalde onderdelen van een request zonder meer opnieuw afgespeeld kunnen worden. Als dat niet kan, dan moeten deze onderdelen, zoals bijvoorbeeld een token, geparameteriseerd worden.

Wat óók een uitdaging aan het maken van testscripts kan zijn, is dat deze gelijktijdig meerdere gebruikers moeten kunnen simuleren, en dat deze gedurende de test continu herhalend afgespeeld worden. Daarom worden de testscripts voor een performancetest gemaakt in een tool. Hierdoor kunnen we de testen continu herhaalbaar maken en is het mogelijk om te variëren in het aantal gebruikers en de duur van de test. Vele smaken en varianten zijn hiervoor mogelijk.

Load profiel

Load profiel is een typische performancetest term. Ik denk niet dat je deze snel ergens anders zult tegen komen. Dus wat is dat dan? Een load profiel is een beschrijving van hoe

een test scenario uitgevoerd moet worden. Het doel van een performancetest is om een bepaalde load op het systeem na te bootsen. Dus met het load profiel geef je per script aan hoeveel gebruikers er gesimuleerd worden en welke load, dus de hoeveelheid transacties, dit tot gevolg heeft op een bepaald tijdstip.

Voor het opstellen van een load profiel is er inzicht in het gedrag van de applicatie vereist. De vraag is dus: “Hoe ziet het gedrag van de gebruikers er op productie er uit?” In het testscript heb je de “happy flow” nagemaakt en vaak zijn er meerdere paden in een “happy flow”. Om een load profiel te kunnen maken, moet je erachter zien te komen hoe het zit met de verhouding tussen de paden. Hierbij is het van cruciaal belang om inzicht te krijgen over het gebruik van het systeem. Aan de hand van de gegevens op productie en de gekozen scenario’s kan er dan een load profiel worden opgesteld. Dit load profiel bepaalt vervolgens hoeveel gebruikers en hoeveel requests er gedaan moeten worden. Uiteindelijk dient het opstellen van het load profiel als input voor de performancetest. Een load profiel kan geconfigureerd worden in de load test tool, zodat deze de gewenste load kan afvuren op het systeem.

Het instellen en het kunnen uitvoeren van een load profiel is iets specifieks van performancetesten.

Test analyse

Als de performancetest klaar is, begint het echte werk. Het analyseren van de resultaten van de resultaten. Dit is een vak op zich en vereist de nodige voorbereidingen al voordat de test is gestart. De meeste performancetest tools hebben weliswaar de nodige analysemogelijkheden in zich. Dit is echter nooit voldoende. Om goed inzicht in de performance te krijgen, moet je de performance vanuit verschillende kanten kunnen bekijken en deze ook aan elkaar kunnen relateren, zoals de transactietijden gemeten door de performancetesttool en door de systemen zelf. De transactietijden geven aan hoe de gebruikers de performance ervaren en de systemen geven aan hoeveel moeite het kost om deze transactietijden te halen.

Voor de transactietijden ben je aangewezen op de performancetesttool. Hoe deze tijden worden gemeten hangt af van hoe je deze in het script hebt vast gelegd. Hier moet je dus ook aan denken bij het maken van een script. Als je bepaalde transactietijden wilt meten, moet dit ook in het testscript gefaciliteerd worden.

Voor het systeemperspectief is monitoring nodig van de systemen. Dit moet van tevoren ingeregeld en getest zijn. Als ik mijn test draai, zie ik dan in de systeemmonitoring terug waar ik naar op zoek ben? Dit gaat dan vaak om zaken als cpu-gebruik, geheugen-gebruik, netwerk en i/o.

Het beste resultaat verkrijg je wanneer de resultaten uit de test gecombineerd kunnen worden in een tool met de metrieken vanuit de performanctesttool en de monitoringtools. Op deze manier kunnen er directe verbanden gelegd worden. Voor een goede analyse kan dit essentieel zijn.

Het is voor de analyse van de testresultaten ook van belang om van tevoren al te bepalen wanneer de performance goed is. Het is zeker niet zo gemakkelijk als bij functioneel testen, waar iets goed of fout is. Dat maakt de analyse juist zo belangrijk, maar dat geldt ook voor het opstellen van performance-eisen.

Test tools

Het uitvoeren van een performancetest kan alleen met een tool of met meerdere tools. De mate van succes wordt mede bepaald door een goede toolkeuze.

Zoals bij het maken van scripts. Hierbij is dat wat ik vertelde onder het kopje ’testscripts’ van belang, namelijk dat de tool de juiste taal spreekt om jouw applicatie te kunnen scripten. Er zijn vele tools en vaak zijn er meerdere tools die kunnen wat jij nodig hebt. Daarom is belangrijk om te kijken naar wie de tool gaat gebruiken. De verschillende tools gebruiken verschillende manieren om scenario’s op te nemen en verschillende vormen van scripts, van volledig via een gui tot programmeren in Java.

Als duidelijk is welke scripts er zijn, dan moeten deze ook nog afgespeeld kunnen worden. Dit is het maken van scenario’s aan de hand van het load profiel. Ook hier zie je grote verschillen per performancetesttool en de manier waarop ze je hierin kunnen faciliteren. Bekijk dus goed wat best binnen jouw omgeving.

Om deze load scenario’s vervolgens te kunnen afspelen, zijn er load generatoren nodig. Er zijn verschillende mogelijkheden op dit gebied. Maar de twee hoofdsmaken zijn on-premise of in de cloud. On-premise betekent dat je zelf load generatoren neerzet en dat deze zo dicht mogelijk bij de testomgeving staan. Cloud loadgeneratoren worden vaak door de testtool aangeboden. Zij leveren dat dan als een dienst.

Na het uitvoeren van de test is het tijd voor de analyse van de resultaten. Hoe deze inzichtelijk gemaakt worden, verschilt ook enorm per tool. Let bijvoorbeeld op de mogelijkheden om de resultaten te kunnen exporteren naar andere tools. Of nog beter; of het mogelijk is om de resultaten direct door te sturen naar de in de organisatie gebruikte tooling voor monitoring van de systemen. Zodoende kun je ook meteen de monitoring tooling testen en ermee leren omgaan. Daar heb je direct profijt van bij het analyseren van productie-issues.

Kort samengevat

Voordat er gestart kan worden met performancetesten, is de nodige voorbereiding nodig. Naast het opzetten van een testomgeving en het vullen ervan met de juiste testdata is er inzicht nodig. Inzicht in wat er op productie gebeurt en gaat gebeuren. Zonder inzicht in wat de gebruikers doen op het systeem, kunnen er geen load profielen worden opgesteld, geen “happy flows” worden bedacht en dus geen testscripts worden opgesteld en gemaakt.

Daarnaast is er nog de keuze van de juiste tooling. Er zijn legio mogelijkheden op het gebied van performancetest tooling, waarbij rekening moet worden geworden met wie gaat het gebruiken, wat past in de organisatie en in de omgeving.

Dan besef je ineens dat het starten van de test zelf nog de minste voeten in de aarde heeft, want daarna of wellicht al tijdens het testen begint pas waar het om gaat: het analyseren van de resultaten.

Om performancetesten tot een succes te maken is aandacht nodig, aandacht voor alle 6 pijlers onder performancetesten.

Rob Leijenaar, performancetest expert