De load- en stress testers te slim af

onthullingen uit de praktijk

Albert Witteveen
10 mei 2011
Testnet voorjaarsevent

Pluton IT

Agenda

- Introductie
- Deel 1: trucks van performance testers
- Deel 2: basis van performance testen en eisen die we aan de test zelf moeten stellen
- Deel 3: Queuing Theorie en hoe we dit kunnen gebruiken
- Een samenvatting over hoe we hiermee controle krijgen

Introductie

- Software tester sinds 1999
- 2000 medeoprichter ProcoliX: High Available en High Performance Webhosting
- 2006 ProcoliX verlaten en Pluton opgericht: gespecialiseerd in technisch testen.
- Sinds 2006 voornamenlijk actief in load- en stress testen en operations

ProcoliX Pluton IT

First contact

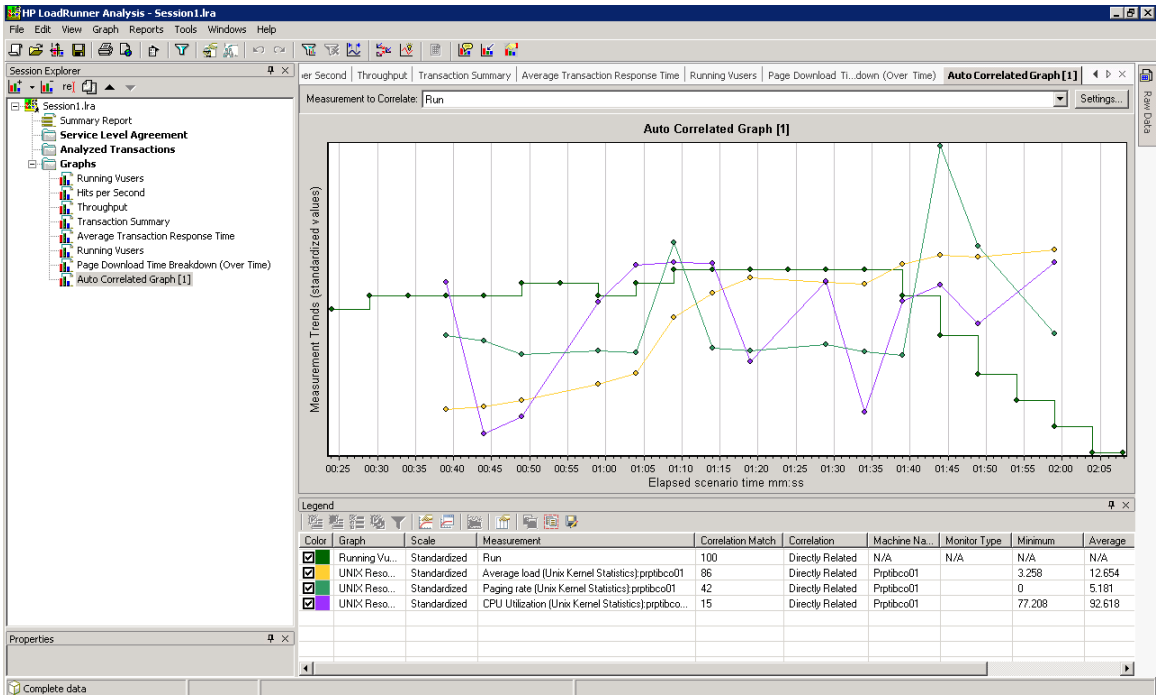
- Eerste ervaring met een load- en stress test rapport:
  - Een willekeurig cijfer gebruikt om aanames op te baseren
  - Het load model (profile) was verzonnen
  - Er was geen gebouwde functionaliteit getest
  - Conclusie viel uit de lucht
- Maar de klant was tevreden!!
- Bij in productiename bleek dat er ca. 10 keer meer hardware nodig was

De praktijk

- De klant heeft moeite te beoordelen of er wel goed wordt getest
- Ondanks performance testen gaat er in productie toch veel mis
- Test managers missen 'handvaten' om de performance testen te sturen en te beoordelen

Onze trucks:

- Indrukwekkende tools
- Dikke rapporten
- Modellen
- Test scope
- Aannames



Pluton IT

De tools

- De load- en stress tools zijn complex, met veel mogelijkheden en metertjes
- Andere tools zien er nog moeilijker uit doordat de tests zelf geprogrammeerd worden
- Als performance tester kan je daardoor makkelijk manipuleren
- Maar ook met de beste bedoelingen kan het misgaan: de tool zegt passed, terwijl de applicatie foutmeldingen gaf.

Dikke rapporten

- Dikke rapporten
  - Veel grafieken
  - Een paar formules met griekse letters
  - Nog wat tabellen
  - En iedereen leest alleen nog maar het advies

Maar als het misgaat in productie is hier toch voldoende bewijs in verstopt dat het niet aan ons lag.

Modellen

- Krachtig wapen:
  - Gebaseerd op cijfers die de klant aanlevert
  - Meestal alleen gemiddelden bekend
  - Definitie van concurrent user is van groot belang
  - Subjectief
  - Effect andere processen meestal onbekend
  - Gebaseerd op veel aannames
- Zelfs als een bestaand systeem vervangen wordt is het zeer moeilijk een objectief model te maken

Test scope

- Problemen in productie blijken vaak ongetest op performance
  - De meeste functies leveren geen druk op het systeem
  - Juist de functies die dat wel doen zijn vaak moeilijk te testen
  - Performance testers zijn vaak niet goed thuis in de functionaliteit
  - Tijd per test is hoog, dus er kan maar een zeer beperkte set getest worden

Dus...

- Niet transparant
- Verschuilen ons achter dikke rapporten
- We trekken conclusies op ongefundeerde modellen
- Veel aannames geven veel ruimte voor manipulatie
- We weten niet wanneer het goed is
- We testen het verkeerde

Transparantie

- Transparantie: om te weten of de load tester met zijn tool echt goede dingen doet:
  - De basis van load en stress testen is hetzelfde ongeacht de tool
  - De valkuilen zijn ook hetzelfde.

- Dus kunnen we:
  - Vooraf eisen stellen aan de test
  - Deze eisen hoeven niet technisch te zijn of afhankelijk van de tool

Dan moeten we wel weten wat een load tester doet.

De basis van L&S testen

Je hoeft de tool niet zelf te kunnen bedienen om te snappen wat er gebeurt

- Basis activiteiten:
  - Load generatie
  - Monitoren
  - Resultaat controle

Load generatie

- In productie komen de requests van client applicaties (of browsers)
- De load generator luistert af, neemt op en genereert een testscript
- Tijdens de testuitvoer, wordt dit verkeer nagebootst, vermenigvuldigd met aantal na te bootsen gebruikers: het script wordt simultaan x maal tegelijk gedraaid.

Data parametrisatie

- Bij het nabootsen van meerdere gebruikers kan niet exact hetzelfde worden verstuurd. (bv login gegevens, klant gegevens)
- Deze data wordt (door de tester) geparametriseerd

	A	B	C
1	Naam	Postcode	Huisnummer
2	Jansen	1001 BT	2
3	Pietersen	2004 HG	3
4	Duck	7542 HH	5
5	Klaasen	8541 HH	7
6	Puk	2541 WT	1

- Hoe meer parametrisatie hoe realistischer, maar ook vergroting complexiteit.
- Bron is belangrijk (gegenereerd, of query)

Load scenario

- Drukken alle gebruikers tegelijk op 'de knop' of kiezen we voor een realistisch scenario?
- Scenario definieert ramp-up en eventueel synchronisatie punten

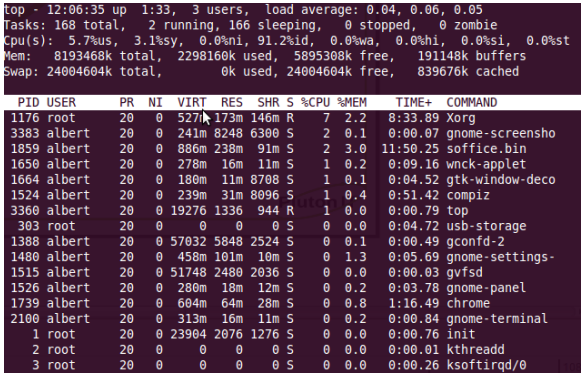
Zeer geschikt om het resultaat te manipuleren

Content controle

- Tools zien uit zichzelf niet of het resultaat klopt:
  b.v. zegt het scherm: 'deze gebruiker bestaat al' of 'gebruiker aangemaakt'
- Controle is noodzakelijk:
  - Tijdens:
    - door middel van content checks in het test script
    - soms door monitoring
  - Achteraf:
    - d.m.v. een steekproef
    - of door een query of controlescript

Monitoring

- Response tijden
  - Doorgaans de load generator zelf
- Hardware resources
  - Vaak door de load en stresss test tool
  - Bij voorkeur ook losse monitoring



De test case

- De test case bevat dus:
  - Functionele teststappen
  - Scenario
    - Rampup
    - Concurrent users
    - Synchronisatie punten
  - Test data en parametrisatie
    - Queries of bron
    - De items die geparametriseerd moeten worden
  - Monitor items
    - Hardware wat gemonitord wordt
    - Monitor definitie: ingebouwd, andere monitors etc.
  - Content controle
    - Content checks en/of
    - Controle achteraf

Wat en hoe te testen

- Bekend terein: Test Risico Analyse
- Maar wie maakt de TRA:
  - Functionele testers en Business analysts weten wat veel geraakt wordt en wat de impact is
  - Developers weten welke onderdelen grotere kans van falen hebben
- Wie maakt de test case:
  - Functionele testers kennen de stappen al
  - Performance testers zijn daar minder goed in

Oftewel: wat en hoe, samen met functionele test teams en development

Resultaat beoordeling

- Probleem:
  - Dikke rapporten
  - Twijfel over aanames en modellen
  - Hoe resultaat te beoordelen?

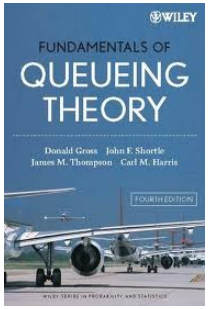
Om resultaat te beoordelen eerst een beschrijving hoe systemen zich gedragen

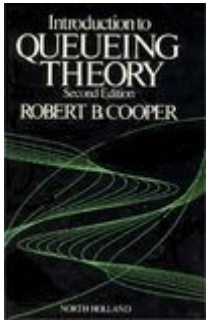
Tester: *“we zittten op 100 % CPU”*

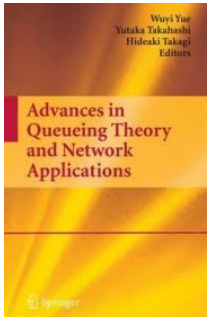
Systeembeheerder: *“mooi, dan gebruiken we ze efficient”*

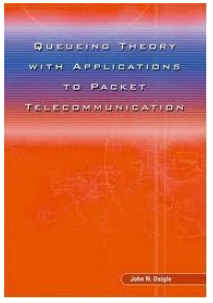
De Queuing Theorie

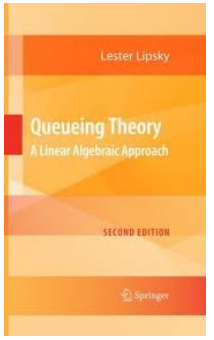
- Anno 1917: beschikbaarheid telefoon maatschappijen
- Elk computer systeem gedraagt zich volgens deze theorie
- Performance experts gebruiken deze theorie
- Toepasbaar op meerdere 'zoom' niveaus.
- Twee goede manieren om te spellen

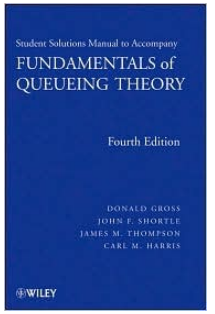








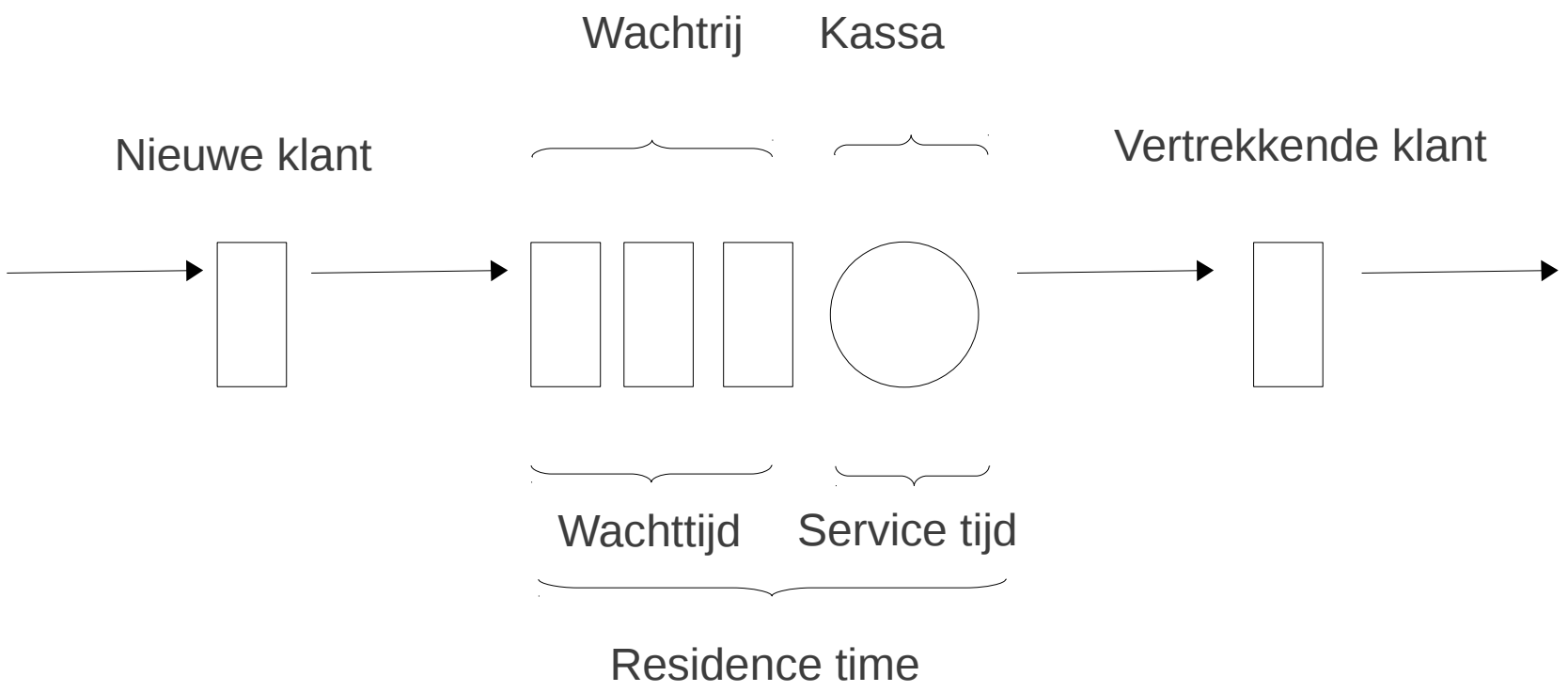




Pluton IT

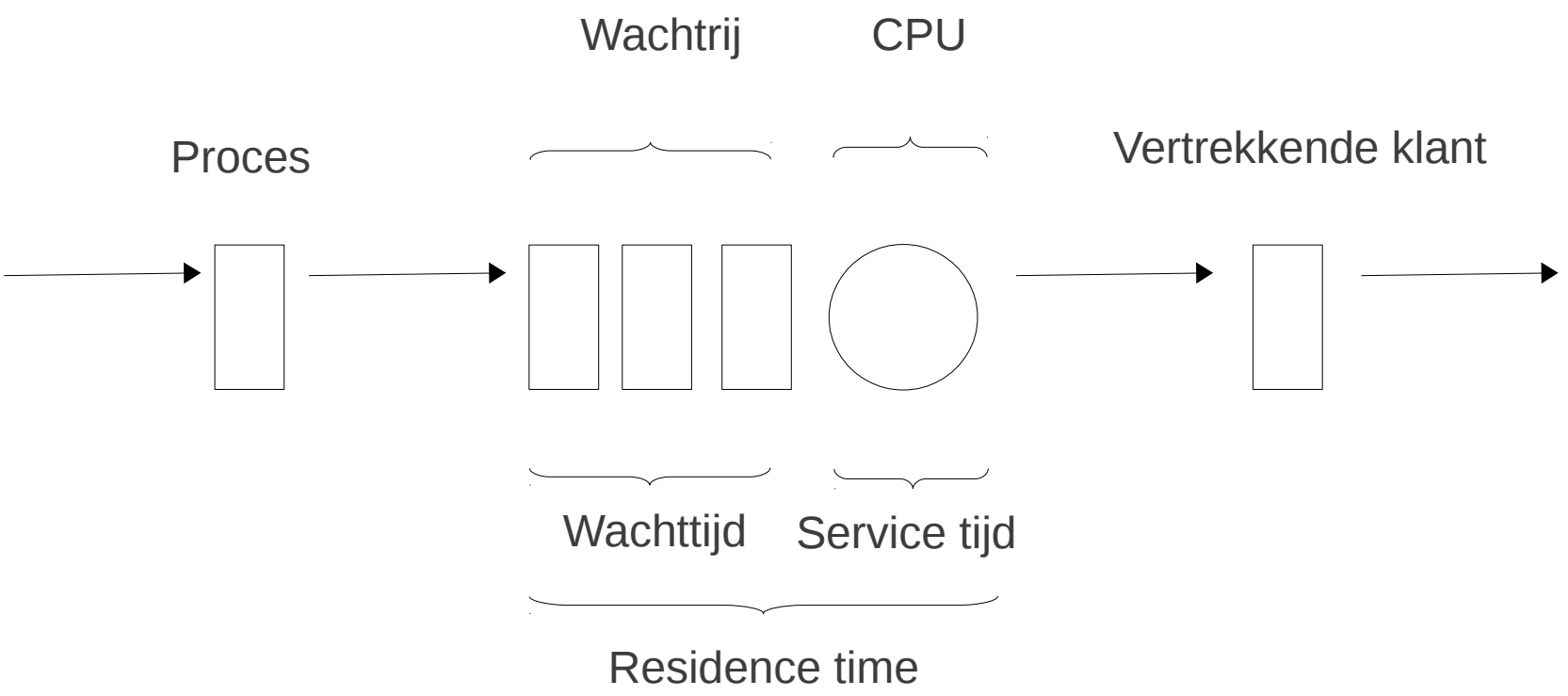
Een simple rij (que)

Vergelijk met een kassa bij de supermarkt

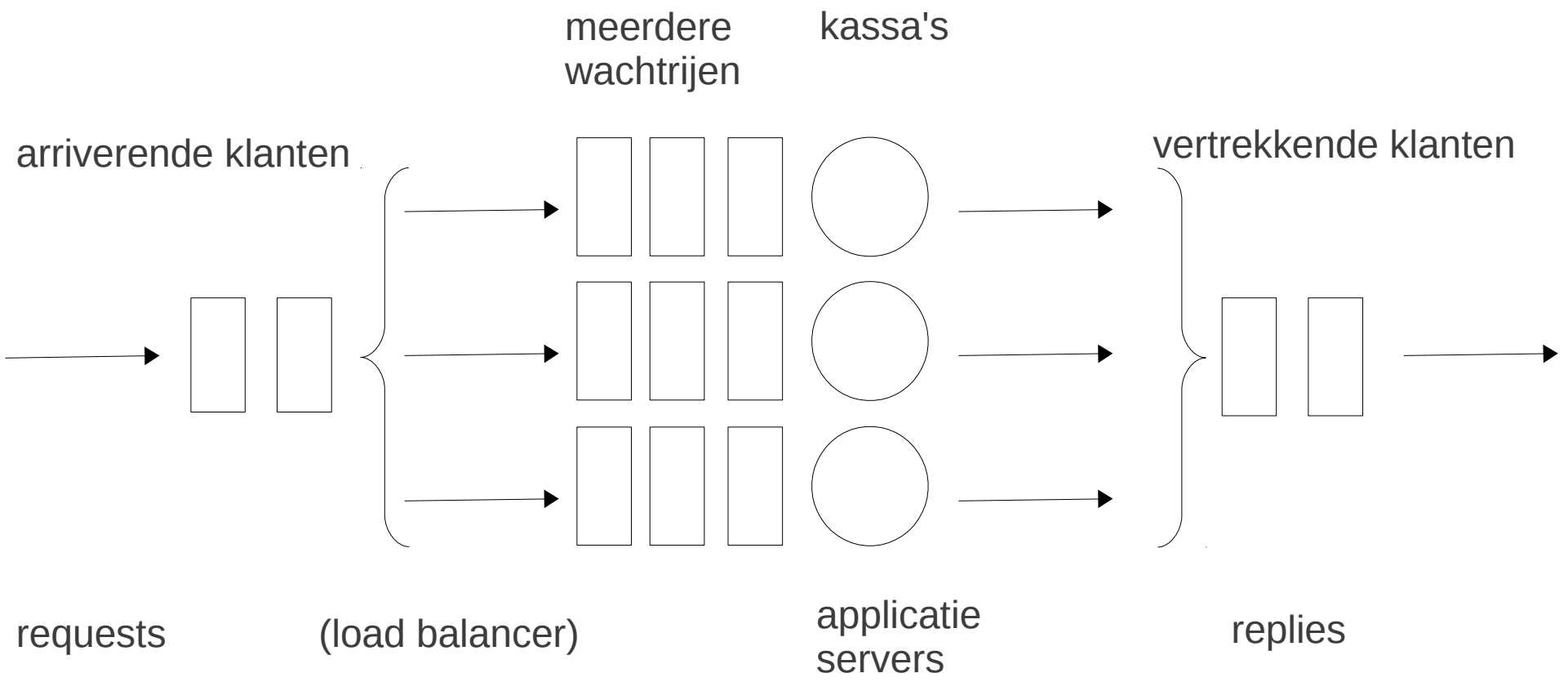


Residence time = totale wachttijd + servicetijd

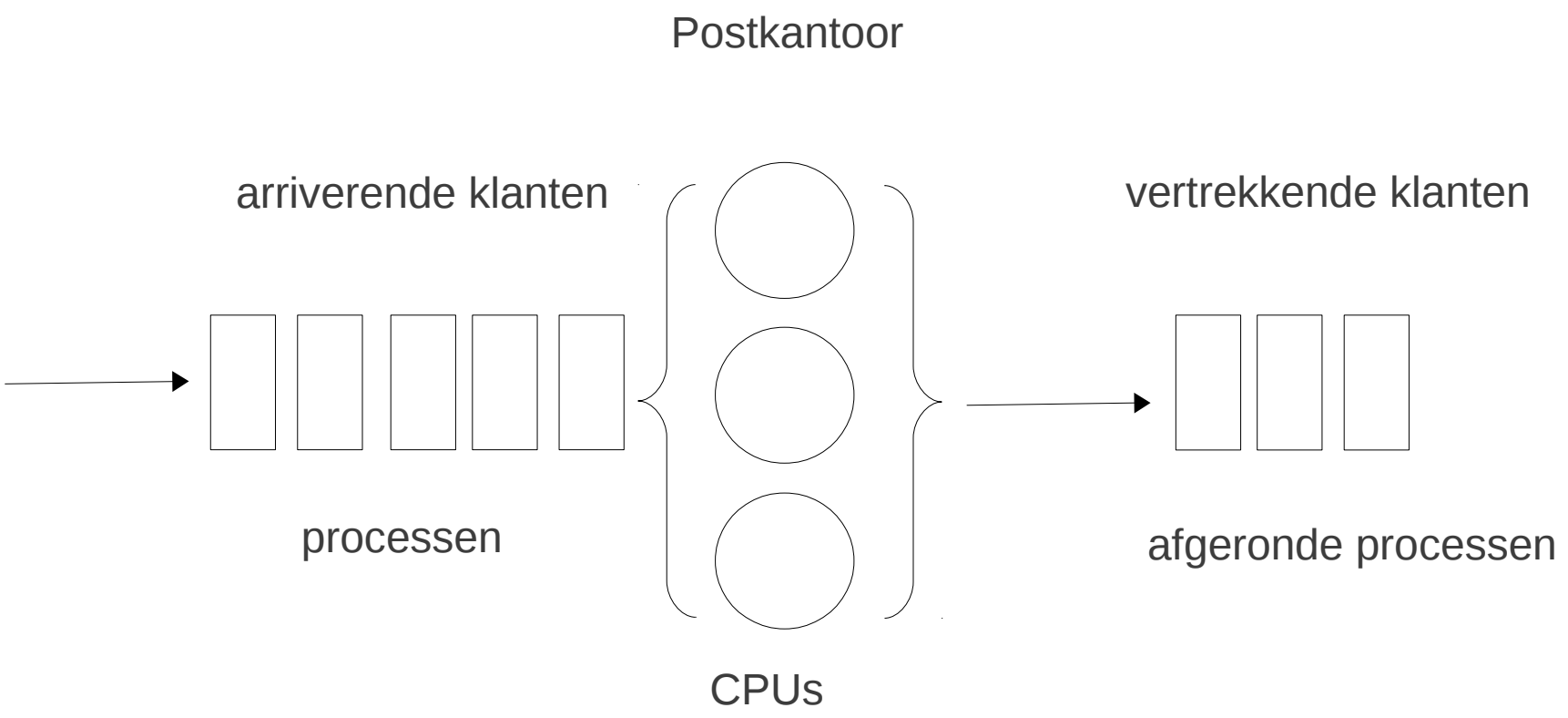
Vergelijk met een CPU



Meerdere kassa's



Nummer trekken



Queuing models

- Beschijven op meerdere zoomniveau's: op infrastructuur niveau, maar ook dieper (server niveau, CPU niveau, netwerk etc.)
- Residence time: totale tijd van wachten en verwerken
- Gebruikt door performance experts bij ontwerp
- Bruikbaar bij test voor resultaat beschrijving

Queuing centres

- Een queuing centre is een locatie in het systeem waar een proces wacht op een ander om af te ronden (vgl: bottleneck)
- Een queuing centre kan van alles zijn: CPU, geheugen, netwerk etc.
- Queuing centres bepalen de schaalbaarheid van de applicatie
- Er zijn vele queuing centres, maar in elk niveau is er minimaal één
- Service time en wait time bepalen de echte performance

Resultaat beoordeling

- Onderscheid tussen twee testdoelen:
  - Gebaseerd op gebruiker requirements: halen we de vereiste respons tijden na go-live.
  - Technische focus: hoe gedraagt het systeem zich. Doorgaans met weinig of zelf vast te stellen requirements

Met requirements

- Focus op gebruiker requirements (response tijden)
- Queuing model en centres:
  - Bieden een maat van diepgang
  - Maken modellen 'op het randje' lastig.
  - Geven een reëel beeld van de performance
  - Geven een onderbouwd beeld van de risico's
  - Geven aan of groei van het gebruik kan worden opgevangen

Gehaald op het randje of echt?

Voorbeeld: Batch proces

- Klant vraag:
  - Batch proces te traag, svp verspreiden op meerdere servers
  - Risico: batch proces mag online processen niet verstoren
- Test toonde aan:
  - Op drie server i.p.v. één batch proces drie keer zo snel
  - Maar geen queuing centre gevonden.
- Door gezocht tot centre gevonden: er zat een 'wait' in de code.
  - We konden met zekerheid zeggen dat de online processen geen hinder zouden ondervinden
  - We konden een veel betere oplossing voorstellen

Zonder requirements

- Geen of zachte requirements
  - Door virtualisatie minder nodig
  - Organisatie weet dat ze dit niet hard krijgen en vragen om 'je professionele oordeel'
- Focus op schaalbaarheid en kwaliteit
  - Kwaliteit: betaalt zich terug door minder onderhoudskosten, risico en herbruikbaarheid
  - Schaalbaarheid: zelfs met een zeer ruim hardwarebudget kunnen we niet alle groei opvangen
  - Doorgaans weinig vastgestelde meetbare requirements

Moderne omgeving

- Nieuwe technieken leiden tot andere vraag
  - O.a. door virtualisatie wordt een 'dutch auction' achtige tuning goedkoper i.p.v. performance testen
- Geïdentifceerde queuing centres bieden:
  - Maat van schaalbaarheid
  - Risico bepaling
  - Tuning handvaten

Schaalbaarheid en risico's

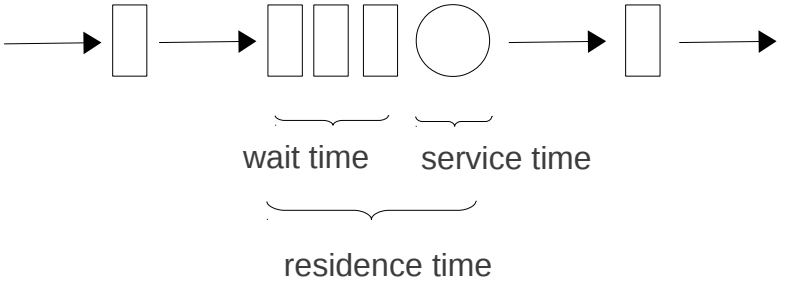
- Schaalbaarheid:
  - de mate waarin we realistisch gezien d.m.v. systeem uitbreiding de performance kunnen verbeteren en/of groei kunnen opvangen
  - Service time is een harde grens van performance en schaalbaarheid!
- Belangrijkste risico's
  - Risico dat bij aanvang stagnatie ontstaat door te weinig performance
  - Risico dat performance problemen of groei van het gebruik niet betaalbaar zijn op te vangen met systeem uitbreiding

Voorbeeld: middleware

- Test: een performance regressie test
- Resultaat:
  - Responsetijden voor de client hetzelfde
  - Maar in middelware verdriedubbelde een residence tijd
  - Responsetijden hetzelfde omdat de client niet wachtte
- Probleem: in productie zou een build up ontstaan die wel tot stagnatie zou hebben geleid
- Alleen dure, gerichte en complexe load test had dat ook aangetoond.

Times

- Residence time = wait time + service time
- Residence time: maat van werkelijke performance
- Service time:
  - Grens van schaalbaarheid
  - Maat van kwaliteit van het proces
  - Verbetering in de software



- Wait time:
  - Afhankelijk van hardware resources en service time
  - Eerste aandachtspunt voor tuning
- Verhouding wait time t.o.v. service time geeft efficientie van de sizing aan

Queuing en testen

- Het totale queuing model geeft handvaten
  - Om performance te beoordelen
  - Om schaalbaarheid te beoordelen
  - Om het risico niveau te bepalen
  - Om opties voor verbetering en tuning te bepalen
- Het toont ook of er goed genoeg is gemonitored om uitspraken te doen
  - Geen queuing centre gevonden: verder zoeken

Modeleer de werkelijkheid

- Effectief geeft een queuing model een model van de werkelijkheid
- Het model kan gebruikt worden om:
  - Risico's te detecteren
  - Schaalbaarheid te kwantificeren
  - Bij gebrek aan requirements, discussie starten over enkel de hoge risico en twijfel gevallen
  - Een hulpmiddel om diepgang in testen te verkrijgen

Het eind rapport

- Stel vooraf eisen aan rapport
  - Risico's onderbouwd met een queuing model
  - Grafieken en cijfers alleen ter illustratie
  - Beperkt aantal pagina's
  - Een model van het gedrag
  - Appendix is voor cijfers, grafieken en techneuten
- Laat vooraf een 'lege' versie opleveren

Samenvatting

Controle over de test door:

- Test case eisen
  - Functionele stappen (samen met functionele test team en development)
  - Scenario eisen
  - Test bevat omschrijving testdata en parametrisatie
  - Monitor items beschreven
  - Content controle
- Queuing model
  - Queuing centres bieden veel houvast als test resultaat
  - Bij traditionele test levert het gegevens op over testdiepte, risico's en schaalbaarheid
  - Bij test voor moderne omgevingen: schaalbaarheid, risicopeil, tuning handvaten
- Rapport eisen
  - Vooraf vast stellen
  - Eisen aan beperking informatie

Vragen

Vragen?