Project 3 Systeembeheer (Linux)

Bert Van Vreckem

2015-09

Contents

1 Doelstellingen

- Een performante webserver opzetten met high availability.
  - HA: servers die wegvallen hebben geen invloed op beschikbaarheid voor de gebruikers
  - performant: ook veel gebruikers kunnen tegelijkertijd de webserver gebruiken
- Monitoren van netwerkservices, opvangen van metrieken
- Stress-testen van netwerkservices
- Reproduceerbare experimenten opzetten, resultaten correct analyseren en rapporteren

2 Opstelling 1: Eenvoudige LAMP stack

- Zet eerst een LAMP-stack op met een PHP-webapplicatie en een databank in de back-end (bijvoorbeeld Drupal, Wordpress, of een andere complexe applicatie). Vul deze op met data (bv. een honderdtal blog-artikels opgevuld met Lorem Ipsum vultekst en commentaren).
- Zet een server op voor monitoring (bv. `collectd`)
- De monitoring server capteert relevante metrieken van de webserver.
  - Dat zijn minstens:
    * aantal queries per seconde (op alle componenten van de opstelling),
    * aantal fouten vs. geslaagde queries,
    * bandbreedte en
    * responstijd (zie ook Mariadb “Slow query log”).
  - Andere kunnen ook, bv.
    * cpu load,
    * netwerk/disk-IO,

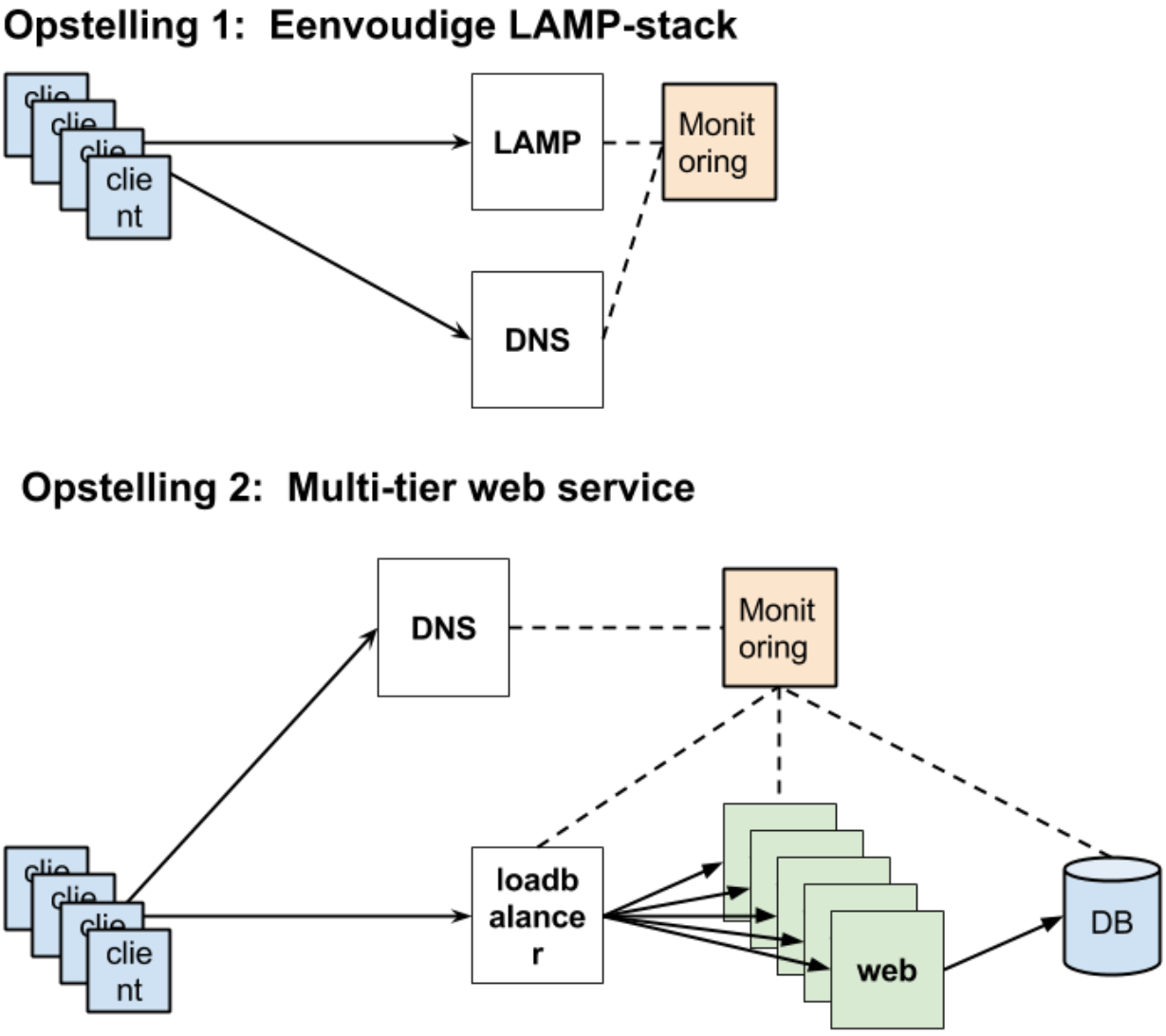

Figure 1: Opstellingen

    * geheugen- en schijfgebruik,
    * enz.

- Gebruik een framework voor load testing van webservers (bv. Apache JMeter, Siege, locust.io, Apache AB, HTTPerf, Low Orbit Ion Cannon, ...), creëer load op je webserver en meet dit op. Werk verschillende herhaalbare scenario's uit, bv.
- weinig load (random pagina's, of browsen van ene link naar de andere)
- veel load (in verschillende grootte-ordes, bv 10, 1.000, 100.000, ... requests per seconde en parallelle gebruikers)
- sommige gebruikers geven af en toe commentaar op artikels
- "Slashdot"- of "Hacker News"-effect (een specifiek blogartikel wordt vermeld op een populaire nieuws-site en ineens komen er enorm veel requests binnen voor die ene pagina, naast de "normale" load voor de rest van de site.)
- ...

Bestudeer het gedrag van de server: vang de metrieken op (`collectd`), en visualiseer ze (dashboard, bv. Graphite of Grafana). Sla de ruwe data op in een vorm die je later nog kan verwerken.

Vanaf hoeveel requests per seconde begint de performantie van de server gevoelig af te nemen? Dit wil zeggen dat de responstijd boven een aanvaardbaar niveau komt. Wat is de verdeling van de metrieken over het experiment? Bestudeer dit a.h.v. de technieken uit de cursus onderzoekstechnieken (centrum- en spreidingsmaten, box-plot, ...)..

Wat betekent een "aanvaardbare" responstijd trouwens? Is dat een welbepaalde tijdsduur? Of ga je het in termen van een spreiding uitdrukken (bv. variantie, percentielen, ...)? Ga op zoek in de vakliteratuur naar hoe dit soort zaken concreet aangepakt wordt. Zie de referenties achteraan dit document voor enkele voorbeelden en startpunten.

Wanneer gaat de server helemaal ten onder? Waar zit de bottleneck? In Apache, de databank, netwerk I/O, processor, geheugen, ...?

Zorg er altijd voor dat je opstelling reproduceerbaar is, zowel de server als de experimenten. Probeer het verwerken van resultaten ook zoveel mogelijk te automatiseren (bv. meteen de statistieken berekenenen, belangrijkste grafieken genereren, enz.).

3 Opstelling 2-n: Multi-tier webserver

Er zijn verschillende methoden om een webserver performant te maken. Hieronder vind je enkele ingrepen die je één voor één kan uitproberen. Herhaal telkens ook de experimenten uit de eerste opstelling en bekijk het verschil. Laat je leiden door de resultaten van je monitoring bij het kiezen van de volgende stap. Probeer meer bepaald de belangrijkste bottleneck te bepalen en weg te werken.

- De webserversoftware vervangen door "performantere" (bv. probeer uit of Nginx statistisch significant beter presteert dan Apache).
- Database op een aparte machine zetten.
- Parallelle webservers + load balancer: vóór de webservers wordt er een "load balancer" (bv. Nginx) geplaatst die de HTTP-requests verdeelt over de verschillende webservers. Er zijn verschillende strategieën om load balancing te doen (bv. round robin, random, failover, ...)
- DNS als load-balancer: de DNS-server geeft af en toe andere IP-adressen terug zodat clients ook andere webservers ondervragen
- Cache-systeem (bv. Memcache, Varnish): vóór de webservers wordt er een cache geplaatst die een kopie bijhoudt van het antwoord van de server op bepaalde HTTP-requests. In plaats van de request uit te voeren (en dus ook de DB te bevragen) wordt de gecachete pagina doorgegeven. Dit zou de load op de database gevoelig moeten verminderen.

Welk effect hebben deze ingrepen? Bijvoorbeeld, hoeveel requests per seconde kan je extra verwerken door webservers toe te voegen? Is het verschil statistisch significant t.o.v. opstelling 1 of andere ingrepen? Toon dit telkens aan aan de

hand van de geschikte statistische toetsen. Hoe verplaatst de bottleneck zich? Hoe verhoudt de load van de loadbalancer zich t.o.v. die van de web- en db-servers? Hoeveel parallelle webservers zou je moeten toevoegen eer de loadbalancer een bottleneck wordt? Kan je met een cache het “HN-effect” tegenhouden?

4 Deliverables

- Geef een **demo** van de verschillende opstellingen, toon de resultaten van de analyse van de meetgegevens.
- **Github/Bitbucket repository** met de code om de opstellingen te reconstrueren. Dit impliceert het gebruik van config management (Ansible). Gebruik waar mogelijk bestaande Ansible rollen (kijk bv. naar https://github.com/bertvv/).
- Schrijf een **verslag** over jullie ervaringen, zorg er in het bijzonder voor dat de hierboven opgesomde vragen aan bod komen.
- Hou de **ruwe data** van de metingen bij in een vorm die kan gebruikt worden voor latere analyse.
- Optioneel: bereid een presentatie in het Engels voor (20 min) waarin je jullie werkwijze en de belangrijkste resultaten toelicht. De bedoeling is deze presentatie te geven op Config Management Camp 2016 (1-2 februari 2016) voor een internationaal publiek.

5 Interessante lectuur

Monitoring en het opvangen van metrieken is een “hot topic” in het werkveld. Hieronder vind je enkele links en artikels over het onderwerp.

Berkholz, D. (2015) *Time for sysadmins to learn data science.* Donnie Berkhoz’s Story of Data (Blog). Opgehaald op 2015-09-08 van http://redmonk.com/dberkholz/2015/01/06/time-for-sysadmins-to-learn-data-science/

Czebotar, J. (2014) *Ten things we forgot to monitor.* Bit.ly Engineering Blog. Opgehaald op 2015-09-08 van http://word.bitly.com/post/74839060954/ten-things-to-monitor

Faustino, K. (2012) *Benchmarking and Load Testing with Siege.* Remarkable Labs Blog. Opgehaald op 2015-09-08 van http://blog.remarkablelabs.com/2012/11/benchmarking-and-load-testing-with-siege

Fisher-Ogden, P., et al. (2015) *Tracking down the Villains: Outlier Detection at Netflix.* The Netflix Tech Blog. Opgehaald op 2015-09-08 van http://techblog.netflix.com/2015/07/tracking-down-villains-outlier.html

Graziano, P. (2013) *Speed Up Your Web Site with Varnish.* Linux Journal. Opgehaald op 2015-09-08 van http://www.linuxjournal.com/content/speed-your-web-site-varnish

Lam, Y. (2015) *Day 5 - How To Talk About Monitors, Tests, and Diagnostics.* SysAdvent. Opgehaald op 2015-09-08 van http://sysadvent.blogspot.be/2014/12/day-5-how-to-talk-about-monitors-tests.html

Rankin, K. (2010) *Hack and / - Linux Troubleshooting, Part I: High Load.* Linux Journal. Opgehaald op 2015-09-08 van http://www.linuxjournal.com/article/10688

Rankin, K. (2012) *The Sysadmin’s Toolbox: sar.* Linux Journal. Opgehaald op 2015-09-08 van http://www.linuxjournal.com/content/sysadmins-toolbox-sar

Saive, R. (2015) *20 Command Line Tools to Monitor Linux Performance.* Tecmint Linux Howto’s Guide. Opgehaald op 2015-09-08 van http://www.tecmint.com/command-line-tools-to-monitor-linux-performance/

Saponaro, J.-M. (2015) *How to monitor Varnish.* Datahog Blog. Opgehaald op 2015-09-08 van https://www.datadoghq.com/blog/top-varnish-performance-metrics/

Souhrada, E. (2015) *MySQL performance optimization: 50% more work with 60% less latency variance.* Pinterest Engineering Blog. Opgehaald op 2015-09-08 van https://engineering.pinterest.com/blog/mysql-performance-optimization-50-more-work-60-

Watson, C., et al. (2013) *Observability at Twitter.* The Twitter Engineering Blog. Opgehaald op 2015-09-08 van https://blog.twitter.com/2013/observability-at-twitter

Watson, C. (2014) *Post mortem: the one where we accidentally DDoSed ourselves.* Keen IO Blog. Opgehaald op 2015-09-08 van https://keen.io/blog/101427090951/post-mortem-the-one-where-we-accidentally-ddosed

Yati, S. (2014) *Load balancing server for dummies: Tutorial part 1*. ServerMom Blog. Opgehaald op 2015-09-08 van http://www.servermom.org/load-balancing-guide/1411/

Websites:

- Monitoringscape: https://www.bigpanda.io/monitoringscape/. Een overzicht van open source monitoring tools
- Monitorama videos: https://vimeo.com/monitorama. Monitorama is een conferentie over monitoring. Videos van de lezingen worden gepubliceerd op Vimeo.