Algemeen

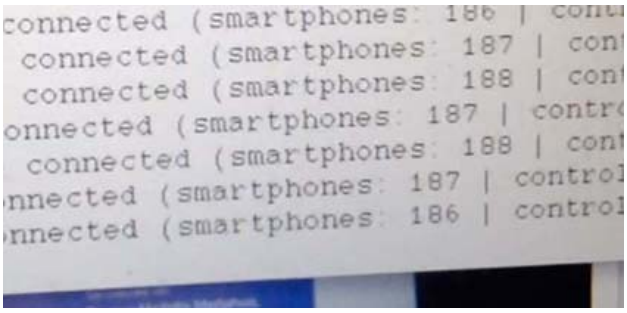
- Voting-app draaide op zelfde machine als mixapp.be
  - voting-app liet dus machine crashen waardoor mixapp.be onbereikbaar was.
  - hadden we dit op verschillende machines gedraaid, dan ging mixapp.be nog laden en konden we een 'even geduld, we zitten met een probleem'-scherm laten zien ;-)

Netwerk

- Netwerk was dik OK! Ward (InAnyEvent) heeft dat super gedaan.
- max 260 connected devices.
- geen overschrijdingen van up/down
  - max 13 MB/s down
  - max 5 MB/s up

mixapp.be

- 139 unieke toestellen op de online versie van mixapp.be
- 214 unieke toestellen op lokale versie van mixapp.be
- 191 maximum gelijktijdige verbindingen op mixapp.be (gebeurde tijdens de hackathon)



"too many open files"

Naast de problemen met de voting-app, ondervonden mixapp.be en 3 hacks ook een probleem. Namelijk `"Error: EMFILE, too many open files" in Node.js`.

Apple heeft namelijke een limiet op het aantal open bestanden gezet. Telkens er een request voor een pagina binnenkwam werd hiervoor een bestand geopend. Bij mixapp.be werd dit snel gefixt door die limiet te verhogen met `ulimit -S -n 10240` maar bij de hacks werd dit pas naar het einde van het optreden toe ontdekt.

Een betere oplossing hiervoor is om de apps in productie modus te zetten, waardoor bestanden 1 keer geladen worden en daarna in geheugen worden bijgehouden.

Dit verhoogt de performantie enorm: wanneer we bijv. requests afvuren op de clientpagina van MediaGoo draaiende op de MacMini, krijgen we (met verhoogde ulimit) iets van een 90req/s die correct afgehandeld worden (=HTTP 200) indien we NIET in production mode staan. (Zonder verhoogde ulimit krijgen we direct de too many open files error; niet zeer zinvol)

Als we wel in production mode staan, verhoogt het aantal HTTP 200/s naar >2000. Het lijkt echter het veiligst ook hier de ulimit te verhogen; anders krijgen we toch nog errors (~30 timeouts/s bij het opzetten van een nieuwe connectie (geen HTTP error), vermoedelijk een timeout aan te passen in de benchmarking tool (tsung))

Hacks die hierop crashten:

- MediaGoo
- Pale Eyes (Phiemel)
- Hexamusic

Hacks die zonder problemen blijven draaien zijn:

- Epleptic
- Sound Defender
- Ossciloscoop (geen pc voor nodig)

Oplossingen

Limieten verhogen:

```
ulimit -S -n 10240
```

Node instellen op production mode

```
NODE_ENV=production node app.js
```

Zorgt ervoor dat de templates (jade, stylus) maar 1x gelezen worden (anders gebeurt dit blijkbaar elke request!). Epleptic en Sound Defender gebruikten html ipv jade en daar kwam de fout niet voor.

Voting-app

- Werkt met Apache/MySQL/PHP
- We gebruikten hier een MAMP-stack op een iMac

Load testen

Enkele load testen met ab:

```
ab -r -k -c 25 -n 1000 http://10.100.11.206:8080/
```

- machine
  - 2.7Ghz Intel Core i5
  - 4GB RAM 1333Mhz DDR3
- not loaded
  - Apache neemt 102,4 MB RAM in beslag
  - 9 `httpd`-processen draaien
- 25 concurrent requests
  - max load 327,68 MB RAM
  - max 39 concurrent `httpd`-processen
  - werkbaar
- 50 concurrent requests
  - max load 645,12 MB RAM
  - max 71 concurrent `httpd`-processen
  - werkbaar, resultaten laden wel traag
- 100 concurrent requests
  - max load 1230,24 MB RAM
  - max 134 concurrent `httpd`-processen
  - gaat traag, resultaten laden in schokjes
- 150 concurrent requests
  - max load 1563,24 MB RAM
  - max 165 concurrent `httpd`-processen
  - gaat traag, niet echt werkbaar
- 300 concurrent requests
  - machine hangt vast!
  - herstarten is nodig

(tussen de testen heen werden de services herstart)

Mogelijke oplossingen

- MaxClients hoger zetten
  - Om te testen op 600 gezet
  - Hiermee raken we aan 600 connecties. De machine loopt dan wel een beetje vast, maar komt er na de load test wel terug door.
  - langste request doet er 98 seconden over
- Op een zwaardere machine draaien (want het probleem is dat het geheugen opgelokt wordt bij iedere connectie)
- Verdelen over meerdere servers met load balancers

Interval polling

In de client code werd `setInterval(jsApp.fetchCurrent, 1000)` gebruikt. Dat betekent dat hij elke seconde een request naar de server lanceert. Tijdens het voten werd dit nog verzet naar `setInterval(jsApp.fetchCurrent, 2000)` waardoor hij om de 2 seconden een request naar de server lanceert. Maar dat mocht niet baten.

Het probleem is, als de server vast hangt, hij sowieso na 2 seconden opnieuw een request doet naar de server. Onafhankelijk van het feit of hij de vorige keer een antwoord gekregen heeft of niet!

Zo kunnen de requests zich vlug opstapelen. Als de server een halve minuut niet antwoord, dan zijn er ondertussen 15 requests naar de server gelanceerd die de server allemaal nog moet afhandelen. Als bij 50 mensen de verbinding geblokeerd is voor 30 seconden, komen hier 50x15 = 750 connecties bij die de server nog allemaal moet afhandelen. De load stapelt zich dus alleen maar op!

Een betere oplossing was geweest om telkens na een antwoord van de server een `setTimeout(jsApp.fetchCurrent,2000)` te gebruiken. Zo wordt gewacht op de server vooraleer een nieuwe request te lanceren.

Een nog betere oplossing is om long polling of zelfs websockets te gebruiken. Waarbij maar 1 request per client wordt gemaakt en alle data over die websocket naar de client vloeit. Zo vermijd je de overhead die een interval-request met zich meebrengt.

MIDI signaal

Op het podium stond een macmini die het MIDI-signaal over het netwerk to bij de hacks bracht. Dit werkte perfect, net zoals op de hackathon.

Het enige nadeel was dat de Compact Disk Dummies niet voor ieder liedje MIDI genereren. Maar dat had Janus op de doorloop de week ervoor nog verteld.

Hacks die gebruik maakten van MIDI:

- Epleptic
- Sound Defender
- Hexamusic (zonder MIDI-signaal werkt het ook, maar dan 'danst' de visualisatie niet op het ritme van de muziek)

AB lights project

Heeft heel de tijd gewerkt, was goed voorzien op mogelijkheden van licht, maar er was te weing tijd om de lichttafel volledig te programmeren, niemand heeft dat gemerkt.

Hadden we nog moeten doen

- op afstand toegang tot iedere iMac zodat we niet telkens de visualisatie moesten minimaliseren om de hack te herstarten.
- elke app op een apparte machine
- kritische apps (voting in dit geval) op backup machine en een manier om er eenvoudig naar de switchen
- SVO switchen en smartphone-switchen loskoppelen zodat SVOs appart gezet kunnen worden.
- Minder snel van hack switchen. Probleem was hier dat er geswitcht werd telkens een hack het begaf.
- load testen ;-)