Universiteit Gent

Vakoverschrijdend project

Project Mobiliteitsbedrijf Gent

Performantie

Groep 7:
Trésor Akimana
Nick De Smedt
Tom Hoet
Dean Parmentier
Simon Scheerlynck
Xavier Seyssens
Jonas Van Wilder
Lennart Vermeir

Lesverantwoordelijken:
Jan Goedgebeur
Annick Van Daele
Prof. Dr. Marko Van Dooren

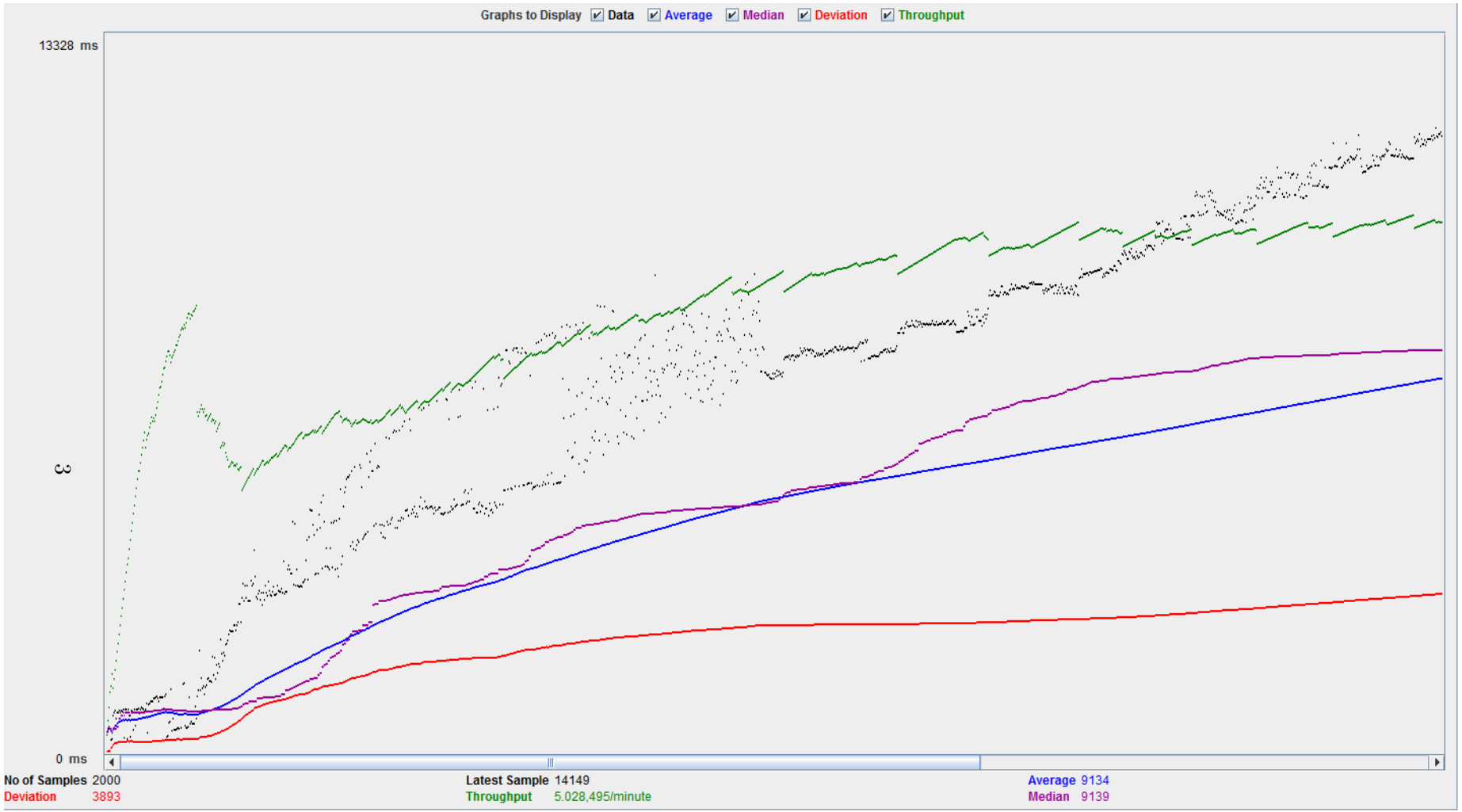
2015-2016

Backend

Om de performantie van de applicatie live te monitoren op de server hebben we gebruik gemaakt van Java Melody. Deze bevindt zich ook in de jar van de applicatie en wordt dus tegelijk uitgevoerd. De resultaten van deze monitoring zijn beschikbaar op de web-gui (https://vopro7.ugent.be/monitoring) en in pdf formaat (Git repository: documentation/milestone3/ op Master). De resultaten in de pdf zijn van een periode van circa 2 weken. Bij het analyseren van deze resultaten moet wel rekening worden gehouden, dat de monitoring niet gebeurde op een volledig systeem in gebruik, maar op een applicatie in ontwikkeling dat nog niet veel data bevatte. Vaak worden bepaalde gemiddelde uitvoeringstijden in het rood aangeduid door Java Melody. We moeten deze echter negeren omdat dit vaak berekend is op basis van n enkel request naar een url van een bepaalde user.

Toch zien we hier al dat er een bottleneck is in de Postgres databank, namelijk bij de query die events en users met elkaar linkt, en dan vooral bij het verwijderen van de overbodige links uit de databank. (**DELETE FROM matchedEvents WHERE event_id = ?**). Deze heeft een gemiddelde uitvoeringstijd van 1,5 seconden, wat ten opzichte van de totale tijd gespendeerd aan sql queries neerkomt dat ongeveer 97% hieraan gespendeerd wordt. Dit komt omdat er extreem veel events gematcht moeten worden en dat deze vaak toegevoegd of verwijderd worden in de databank. We hebben geprobeerd om de query minder vaak uit te voeren door niet alle matches voor een event telkens te verwijderen, maar enkel diegene die niet meer van toepassing zijn voordat de nieuwe matches worden toegevoegd. Er zijn dus nog zeker verdere optimalisaties mogelijk aan de matching en processing van events. Daarnaast kunnen we uit deze resultaten ook afleiden dat vooral het opvragen van events het meeste tijd vraagt. We zien dat 80% van de tijd uitgevoerd in de SpringControllers besteed is aan het opvragen van events.

Verder hebben we ook performantietesten gedaan met behulp van JMeter. We simuleerden de situatie waarin meerdere gebruikers tegelijk een lijst van alle events opvragen. In de praktijk is deze exacte situatie wel niet mogelijk aangezien deze beperkt is tot de rollen operator en admin. Een gelijkaardige situatie kan wel bekomen worden, wanneer veel gebruikers tegelijk de voor hen relevante events opvragen. Ze kunnen nog altijd bijvoorbeeld een interessepunt aanmaken voor heel Gent, zodat ze dus alle actieve events kunnen opvragen. In zo een situatie zou het bijvoorbeeld kunnen voorvallen dat 2000 gebruikers in een tijdspanne van 10 seconden hun events opvragen (circa 80). Het resultaat wordt weergegeven in onderstaande figuur.


Graphs to Display
Data
Average
Median
Deviation
Throughput
13328 ms
0 ms
No of Samples 2000
Latest Sample 14149
Average 9134
Deviation 3893
Throughput 5.028,495/minute
Median 9139