Google App Engine – rakendused pilvedesse

1. Sissejuhatus

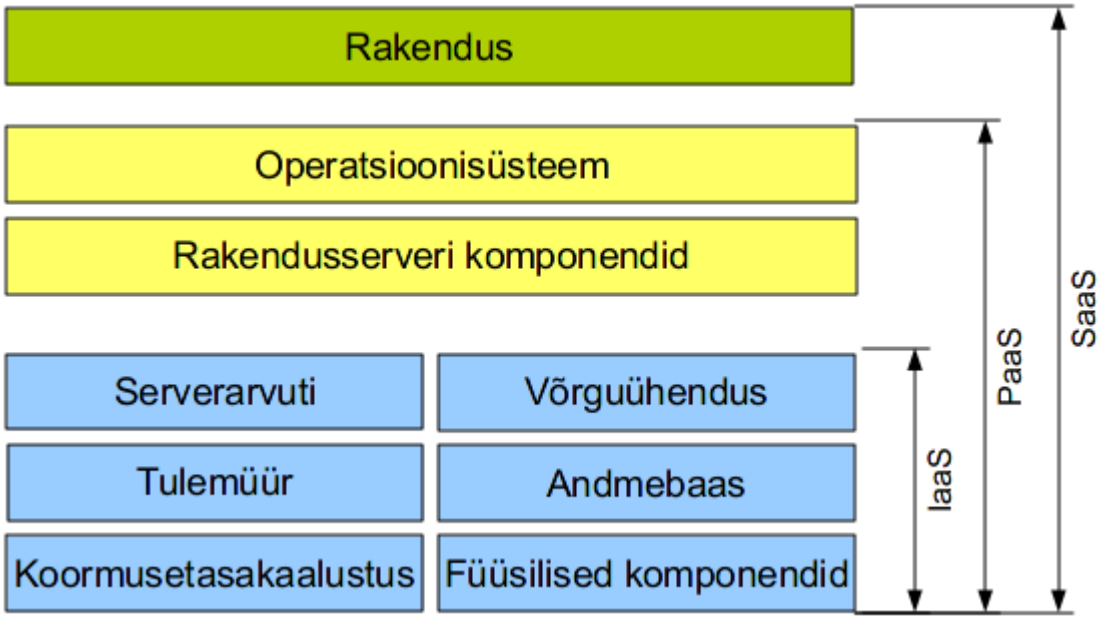
Viimastel aastatel on tugevalt esile kerkinud uus suund andmetöötluses – pilvandmetöötlus (ingl *Cloud Computing*) ning sellega seotud märksõnad nagu *SaaS* või *Amazon AWS*. Küll aga on vähem räägitud, kuidas lisandunud võimalusi reaalselt kasutada, ning kas tegu on vaid turunduslikku sõnumit kandvate teadetega või on tegemist ka millegi reaalselt kasulikuga.

Käesolev artikkel üritab osaliselt tühimikku täita ning tutvustab *Google*'i pakutavat pilveteenust nimega *Google App Engine*. Lühidalt on juttu ka pilveteenustest ning pilvandmetöötlusest üldises plaanis.

2. Google'i pilveteenus

Google App Engine [1] on *Google*'i infrastruktuuril põhinev veebirakenduste platvorm. Tegu on andmetöötluse teenusega, kus andmed paiknevad niiöelda serverite pilves, võimaldades sellega vajaduse korral täiendavalt kasutada teiste pilves paiknevate serverite ressursse. Taoline arhitektuur pakub väga suurt skaleeruvust – üksiku serveri puhul saab ressursse kasutada vaid konkreetse serveri limiitide raames ning kui nendest ei jätku, tuleb kas serveri riistvara uuendada või server hoopis välja vahetada.

Pilveteenuse kasutaja jaoks on pilve näol tegu omamoodi musta kastiga, millel on küll sisendid ja väljundid, kuid mille reaalsest sisust ülevaade puudub. Kõik pilveteenused saab üldjoontes jagada kolmeks suuremaks grupiks (Joonis 1): infrastruktuur kui teenus (*IaaS, Infrastructure as a Service*), platvorm kui teenus (*PaaS, Platform as a Service*) ning tarkvara kui teenus (*SaaS, Software as a Service*), millest igaüks kannab eri abstraheerituse taset. Kui tarkvarateenus (*SaaS*, näiteks *Gmail* [2] e-posti rakendus) võimaldab täita ainult üht konkreetset ülesannet, siis infrastruktuuri teenus (*IaaS*, näiteks *Amazon EC2* [3] virtuaalserverid) annab küll laia võimaluste hulga, kuid praktiliselt mitte mingeid hõlbustusvahendeid nende võimaluste täitmiseks.



Joonis 1. Pilveteenuste erinevate kihtide poolt pakutavad võimalused

Google App Engine on oma olemuselt platvormteenus (*PaaS*), mis paigutub tarkvara- ja infrastruktuuri teenuse vahele. Rakenduste loojad on platvormteenuse puhul surutud küll kindlatesse raamidesse (*Google App Engine* pakub programmeerimiskeelteks vaid *Pythonit* või *Javat*, andmebaasiliidestused on piiratud jne), kuid erinevalt tarkvarateenuse tasemest ei ole rakenduste eesmärgid eeldefineeritud ning seega on võimalik piirangutest hoolimata hõlmata suhteliselt laia skaalat. Infrastruktuuri teenusest erinevalt ei pea platvormteenuse rakenduste arendajad aga muretsema madala taseme komponentide ega süsteemi hooldamise pärast – näiteks puudub vajadus jälgida operatsioonisüsteemi turvauuenduste ajakohasust või paigaldatud tarkvarakomponentide ühilduvust.

Pilveteenuste hinnastamine toimub põhimõttel „maksad vaid selle eest, mida kasutad“ – sama kehtib ka *Google App Engine*'i puhul. Kui teenuse koormus ja ressursikasutus tõuseb, siis maksad vastavalt rohkem, kui aga langeb, siis vähem. Kusjuures kuni esimeste limiitide ületamiseni (limiidid on peamiselt päevapõhised) on *Google App Engine*'i teenus sootuks tasuta. Kuna *Google App Engine*'i limiidid on välja töötatud USA turgu järgides, siis on Eesti veebilehtede puhul, kus koormused on rahvastiku arvust tulenevalt niigi väikesed, isegi tasuta limiite keeruline ületada. Nii võib Eesti oludes tinglikult pidada *Google App Engine*'it tasuta teenuseks. *Google App Engine* pakub seejuures alguses võimalust kasutada just tasuta paketti. Selle peamiseks erinevuseks tasulisest on fakt, et limiitide ületamise korral jääb rakenduse teenindamine kuni limiitide vabanemiseni (st uue päeva alguseni) seisma, samas kui tasulise versiooni puhul esitatakse tasuta limiite ületanud ressursside eest arve.

Google App Engine'i laadsed teenused sobivad eriti hästi automaatse skaleeruvuse tõttu sellistele rakendustele, kus suuri ressursse vajatakse harva, üldjuhul aga on rakenduse kasutamine väike kui mitte olematu. Nii pole tarvis nende harvade tavalisest suurema koormusega hetkede tarbeks ennetavalt osta ja hallata kallimat riistvara ja võrguühendust, mis enamiku ajast istuks lihtsalt jõude, kuna tavaolukorras serveril vastav koormus puudub.

Heaks näiteks skaleeruva ressursikasutuse vajalikkuse kohta on *Digg* ja *Slashdot* efektid [4]. See tähendab, et veebilehe viide satub mõne populaarse linkija esilehele, olgu selleks siis *Digg.com*, *Slashdot.com* vms – tagajärjeks on see, et piiratud ajaperioodi jooksul (mõned tunnid kuni mõned päevad, niikaua kui link püsib taolise linkija esilehel) külastab teenust suur hulk kasutajaid. Eesti kontekstis võib ekstreemseks näiteks tuua valimised, mille puhul valimispäeva õhtul külastab Vabariigi Valimiskomisjoni veebilehte suur mass tulemusi ootavaid inimesi, samas kui ülejäänud osa aastast on lehe külastatavus praktiliselt nullilähedane.

Valimiste korraldajaid aitab asjaolu, et need on ajaliselt planeeritud, ning seega on külastatavuse järsk kasv eelnevalt prognoositav, aga lingi sattumist populaarsele veebilehele on palju keerulisem ette näha. Klassikalise veebimajutuse korral lakkaks server suure tõenäosusega sellise ootamatu ülekoormuse all töötamast, kuid skaleeruva teenuse puhul ei tekiks rakenduse töös katkestusi – kuu lõpus laekuks lihtsalt harjumuspärasest veidi suurem arve.

Google App Engine'i tasuta paketi peamised limiidid on välja toodud allolevas tabelis (Tabel 1). Juhul kui need limiidid ületatakse, peatub rakenduse teenindamine *Google*'i serverite poolt ning kasutajad näevad lehe avades *Google*'i logoga veateadet.

Ressurss	Päevane limiit	Maksimaalne sagedus
Pöördumisi	1 300 000 päringut	7 400 päringut minutis
Väljuv andmemaht	1 GB	56 MB minutis
Sisenev andmemaht	1 GB	56 MB minutis

Protsessorikasutus	6.5 CPU tundi	15 CPU minutit minutis
Kasutatav kettapind	1 GB	
E-posti saatmine	2000 väljuvat kirja	8 kirja minutis
Memcache päringuid	8 600 000	48 000 päringut minutis

Tabel 1. Tasuta kasutamise limiidid *Google App Engine*'i rakenduste jaoks

Tasulise paketi korral jäävad tasuta limiidid samaks ning maksta tuleb vaid neid limiite ületavate ressursside eest. Muutuvad ka maksimaalsed sagedused, mis tõusevad tasulise paketi puhul märgatavalt. Näiteks kui väljuv andmemaht on tasuta paketis maksimaalse sagedusega 56 MB/minutis, siis tasulises paketis 10 GB/minutis; e-kirjade saatmise limiit tõuseb tasuta paketi 8 kirjalt minutis 5100 kirjani minutis jne.

3. Platvorm

Google App Engine erineb teistest pilveteenuste pakkujatest (näiteks *Amazon AWS* [5]) selle poolest, et pakub mitte üldist laadi, vaid kindlaksmääratud platvormil rakenduste majutust. Kogu arveldamine käib rakenduste põhjal – igal rakendusel on kasutada teatud arv ressursse ning maksta tuleb konkreetse rakenduse ressursikasutusele vastavalt.

Google App Engine'i platvormil ei saa hoida ega käivitada suvalisi faile. Tuleb kinni pidada ja arvestada mitmete reeglite ja piirangutega.

Esiteks programmeerimiskeele valik – valida on vaid kahe keskkonna, *Java* ja *Pythoni* vahel. Kolmandate keelte (näiteks *PHP* või *Ruby*) kasutamine on reeglina võimalik vaid *Java* abil, kui vastava keele jaoks on olemas *Java* interpretaator. See tähendab, et võetakse näiteks *PHP*-keeles kirjutatud skript, tõlgitakse see *Javale* loetavaks ning *Java* käivitab selle *Google App Engine*'i platvormil tavalise *Java*-põhise rakendusena. Sellist funktsionaalsust *PHP*-programmide käivitamiseks *Google App Engine*'i platvormil võimaldab *Quercus* [6]. Käesolevas artiklis on käsitletud peamiselt *Pythoniga* seonduv, kuna *Python* oli esimene keel, mida *Google App Engine* toetas.

Teiseks oluliseks erinevuseks võrreldes muude teenusepakkujatega on andmebaasivalik. *Google App Engine* ei toeta *MySQL*, *MSSQL* ega muid relatsioonilisi andmebaase, vaid ainult *Google*'i poolt välja töötatud mitte-relatsioonilist andmebaasi *BigTable*. *BigTable*'i üheks piiravamaks omaduseks *Google App Engine*'i platvormil on kuni 1 MB suurused andmebaasi kirjed, kuna *Google App Engine*'is puudub rakendustel failisüsteemi kirjutamise õigus ning kõik programmi töö jooksul tekkinud failid tuleb salvestada failisüsteemi asemel andmebaasi. Õnneks on alternatiivina võimalik kasutada suuremate failide salvestamiseks *BlobStore*'i failihoidlat, kuid see on mõnevõrra keerukam.

Kolmandaks teenuse suureks piiranguks on juba eelpool nimetatud 1 MB piirang: andmebaasi iga kirje peab jääma 1 MB raamidesse, teistest serveritest saab alla laadida kuni 1 MB suuruseid faile, välja saadetava e-kirja maksimaalne suurus koos manustega saab olla 1 MB jne.

Neljandaks piiranguks on ligipääs serverile – *Google App Engine* ei võimalda serveris asuvatele failidele harjumuspärast ligipääsu mõne failiedastusprotokolli abil nagu näiteks *FTP*, vaid kasutada tuleb arenduspaketi vahendeid. Failide laadimine on ühesuunaline – ainult üles. Kui staatilisi faile saab veebi kaudu tavakorras siiski ka alla laadida, siis programmifaile serverist enam tagasi laadida ei saa – kui rakenduse lähtefaile enam kohalikus arvutis alles pole, siis neid taastada ei õnnestu. Sellisel juhul pole võimalik serveris edaspidiseid muudatusi teha, kuna failide

üleslaadimisel uuendatakse korraga tervet rakendust, mitte üksikut faili. Taolise probleemi vältimiseks tasub kindlasti kaaluda täiendavalt mõne versioonihalduskeskkonna kasutamist, näiteks *Subversion*, *Git* vms, mis lähtefailid turvaliselt ära varundaks.

Viimase piiranguga kaasneb ometi ka positiivne omadus – kuna kõiki faile tuleb mingil moel varundada ning arendajate vahel sünkroniseerida, ilma et vahepeal saaks failidega otse serveris toimetada, on kõik projektis osalejad sunnitud korrektselt versioonihaldustarkvara kasutama ning kõik muudatused selles registreerima. Suuremates ettevõtetes pole see tõenäoliselt juba praegugi probleemiks, küll aga väiksemates, kus iga töötaja toimetab omapead ning tegevustest puudub reaalne ülevaade. Rääkimata sellest, et kui serveris midagi juhtuma peaks, siis rakenduse taastamine on ülimalt keeruline, kuna failid on tõenäoliselt kõik versioonimata ja varundamata.

Viiendaks oluliseks piirajaks on programmi tööaeg – *Google App Engine*'i poolt on see piiratud 30 sekundiga ning kui programm selle aja täitudes pole veel tööd lõpetanud, ilmub veateade. Andmebaasi puhul võib see aeg olla isegi tunduvalt lühem – viga võib ilmuda näiteks siis, kui andmebaas ei vasta vaid mõne sekundi jooksul.

Probleeme on ka *SSL*-sertifikaatidega. *Google App Engine* jagab kõikide rakenduste ja kasutatavate IP-aadresside vahel ühte ja sama **.appspot.com* suhtes kehtivat *wildcard SSL*-sertifikaati. *HTTPS* protokoll aga ei võimalda ühe IP/pordi kombinatsiooni kohta rohkem kui ühte sertifikaati kasutada [7]. *Google App Engine*'i tööplaanis [8] on antud küsimus kirjas, seega tulevikus antud probleem tõenäoliselt lahendatakse.

Seatud piirangutest on võimalik mööda pääseda oma *App Engine*'i ülesseadmisega. Kalifornia Santa Barbara ülikoolis on loodud *AppScale*'i [9] nimeline vabavaraline platvorm *Google App Engine*'i rakenduste paigaldamiseks ja majutamiseks kasutaja enda (virtuaalsetest) serveritest moodustatud klastrites. *AppScale* toetab näiteks ka *Amazon EC2* virtuaalserveriplatvormi, võimaldades sellega *Google App Engine*'i rakendusi jooksutada ka *Amazoni* infrastruktuuri teenusel. *AppScale* erineb originaalsest *Google App Engine*'i teenusest muuhulgas valitud andmebaasi poolest – kuigi andmebaasi liides on rakendustele sama nagu *Google App Engine*'i puhul, pole kasutatud *BigTable*'i andmebaasi, vaid mõnd vabavaralist alternatiivi (*Voldemort, MemcacheDB, MySQL* vms).

Boonusena sisaldab *Google App Engine* tasuta sisuedastusvõrgu (*CDN*) teenust. Kõik rakendused kopeeritakse täies mahus erinevates asukohtades olevatesse serveritesse, kusjuures mida aktiivsemalt konkreetset rakendust kasutatakse, seda rohkem antud rakendust pilves ka levitatakse. Nii suunatakse kasutajate päringud alati füüsiliselt lähimal asuva koopia juurde, viies sellega võrgu kiirusest sõltuvad reageerimisajad miinimumini.

Kui praegu võib veel Eesti asukohast tulenevalt kiirustega kohati probleeme esineda, siis *Google*'i Soomes paikneva Hamina [10] andmekeskuse käivitudes peaksid *Google App Engine*'i rakenduste ühenduskiirused muutuma Eestis majutatud lahendustega praktiliselt võrdseteks. Sama kiirelt saab *Google App Engine*'i rakendust kasutada ka näiteks Aasias või Ameerika – *Google App Engine*'i puhul pole vahet, millises maailma otsas kasutaja asub.

4. Kasutamine

Google App Engine'i kasutamiseks on vaja paigaldada arvutisse vastav tarkvara arenduspakett (*SDK*), mille saab alla laadida *Google App Engine*'i kodulehelt [11]. Toetatud on kõik levinumad

platvormid – *Windows, Mac OSX* ja *Linux*. *Windowsi* ja *Maci* versioonide puhul on lisatud graafiline liides, kuid *Linuxi* puhul peab käsureal toimetades hakkama saama.

Arenduspaketti on tarvis peamiselt kahel põhjusel. Esiteks sisaldab see kohaliku veebiserveri näol *Google App Engine*'i emulaatorit, mis võimaldab arendusarvutis midagi üles laadimata proovida, kas programm töötab või mitte. Teiseks toimub failide üleslaadimine *Google*'i serveritesse ainult arenduskeskkonna poolt võimaldatud liidese abil.

Harjumuspäraseid *FTP, SSH* vms failiedastusvahendeid *Google App Engine*'i puhul seega kasutada ei saa ning kogu vajalik funktsionaalsus on realiseeritud *SDK* enda sees. *SDK* poolt kontrollitav üleslaadimine on samas väga mugav – piisab vaid vajutusest nupul *Deploy* või *Linuxi* puhul (kuna graafiline keskkond puudub) üleslaadimise käsu sisestamisest. Uue versiooni paigaldamine *Google App Engine*'i serverisse võtab reeglina alla minuti (sõltub failide koguarvust projektis ning üleslaetavate failide suurusest), kohalikus testserveris ei kulu aga sedagi – rakendus hakkab tööle kohe programmikoodi salvestamise järel.

SDK laeb üles ainult failid, mis on uued või mida on muudetud. Samuti toimub taustal paigalduste versioonimine – juhul kui üleslaadimine ebaõnnestus või programm osutus vigaseks, saab üleslaadimist „tagasi pöörata," misjärel taastab server programmi eelmise oleku. Kuna olemasolevaid faile ei kirjutata kunagi üle, vaid ainult lisatakse, ei saa tekkida probleemi, kus programm on serveris lootusetult rikkis – alati saab naasta mõne eelneva töökorras versiooni juurde.

Arenduspaketi graafilise liidese abil saab luua ka uusi rakendusi, kasutades käsku File→New Application…. Avanenud dialoogiaknas tuleb määrata rakenduse unikaalne identifikaator ning projektifailide asukoha kataloog. Identifikaator tohib sisaldada vaid ladina tähti, numbreid ja sidekriipsu. Kuigi programm ei anna veateadet, kui identifikaator sisaldab näiteks tühikuid, siis hiljem vigase identifikaatoriga rakendust käivitada siiski ei saa.

Google App Engine'i platvormile loodavad rakendused koosnevad reeglina minimaalselt kolmest määratud failist: app.yaml (määrab ära rakenduse konfiguratsiooni), index.yaml (seab andmebaasi indeksid) ning main.py (selles asub rakenduse programmikood).

Rakenduse paigaldamiseks *Google App Engine*'i serverisse tuleb rakendus kõigepealt registreerida *Google App Engine*'i teenuste lehel [12]. Pärast registreerimist eraldatakse rakendusele vajalik serveripind ning domeeninimi kujul identifikaator.appspot.com. Domeeninime määramine selgitabki identifikaatori puhul lubatud sümbolite piiratuse – kõiki sümboleid (näiteks tühikut) ei ole nimelt võimalik domeeninimedes kasutada. Enne rakenduse registreerimist tuleb aktiveerida ka *Google App Engine*'i kasutamise konto, mida saab teha lihtsalt lehte külastades. Vajalik on mobiiltelefoni olemasolu, kuna kasutaja tuvastamine käib SMS-i teel.

Kui rakendus on registreeritud ja kohalikus testserveris on selle töötamine kontrollitud, saab rakenduse paigaldada *Google App Engine*'i serverisse arenduspaketi käsuga *Deploy*. Peale paigaldamisprotsessi lõppu saab rakendust vahetult kasutama hakata rakenduse veebiaadressi kaudu.

Kui on soov kasutada identifikaator.appspot.com domeeninime asemel midagi muud, näiteks www.minudomeen.ee, tuleb antud domeeni suhtes luua *Google Apps*'i [13] konto. *Google Apps*'i peamiseks kasutusalaks on oma domeeni e-posti liikluse suunamine *Google*'i serveritesse, kuid

see võimaldab lisaks domeeni sidumist *Google App Engine*'i rakendusega. *Google Apps Pro* versioon on tasuline, kuid *Google* pakub ka tasuta *Standard* versiooni.

5. Programmi struktuur

Rakenduse töö jaoks tuleks *Pythoni* programmi laadida erinevad abistavad teegid, milleks *Google App Engine*'i puhul oleksid kindlasti vajalikud teegid veebipäringute teenindamiseks. Veebipäringuid oskab hallata *Google App Engine*'i poolt vaikimisi pakutav *WebAPP* raamistik. Võimalik on kasutada ka muid raamistikke, kuid *WebAPP* on kõige lihtsam viis rakendust kiirelt tööle saada.

```
import wsgiref.handlers
from google.appengine.ext import webapp
```

Antud teekide poolt pakutavad meetodid oskavad vastu võtta app.yaml konfiguratsiooni poolt suunatud veebipäringuid (näiteks kui keegi avab aadressi www.server.ee/info, siis /info suunatakse programmile kui sisenev päring) ning leida nendest üles erinevad päringu andmed (*GET*- ja *POST* -muutujad, küpsised jne).

Konkreetsete päringute haldamiseks *WebAPP* raamistikus tuleb luua igale päringutingimusele webapp.RequestHandler tüüpi päringuklass. Sellise klassi meetod get oskab vastata *GET*-päringutele ning post vastavalt *POST* -päringutele.

```
class InfoHandler(webapp.RequestHandler):
  def get(self):
    self.response.out.write("See on infotekst!")
```

Päringuklassid saab päringutingimustega siduda webapp.WSGIApplication meetodi abil – selle parameetriks tuleb anda massiiv, mis määrab ühesed seosed päringute ja päringuklasside vahel. Näites seotakse päringuaadress /info haldajaga InfoHandler.

```
def main():
  application = webapp.WSGIApplication([('/info', InfoHandler)])
  wsgiref.handlers.CGIHandler().run(application)
if __name__ == '__main__':
  main()
```

webapp.RequestHandler tüüpi päringuklassides asuvad kõik sisenevate päringutega seotud andmed objektis self.request ning väljuvate andmetega seotud andmed objektis self.response. Näiteks kui on vaja midagi väljastada, saab seda teha objekti self.response omadusega out:

```
self.response.out.write("see läheb ekraanile")
```

Kui on vaja lugeda *GET*- või *POST*-päringuga edastatud parameetri väärtust, saab seda teha objektiself.request meetodiga get:

```
nimi = self.request.get("nimi")
```

Lisaks *WebAPP* raamistikule võimaldab *Google App Engine* ilma täiendavaid faile üles laadimata kasutada *Django* [14] versiooni 0.96. *Django* on mahukas raamistik veebirakenduste loomiseks, kuid *Google App Engine*'i platvormil asuvate rakenduste jaoks piisab tihtipeale ka vaid mõnest üksikust *Django* pakutavast osast. Näiteks saab kasutada *Django* väga head lehemallisüsteemi [15] või samas paketis sisalduvat *JSON* teeki.

Üldiselt saab *Google App Engine*'i platvormil kasutada kõiki *Pythoni* versiooni 2.5 võimalusi – lisaks *Google App Engine*'i poolt pakutud teekidele saab kasutada ka teisi, kui need programmi koodi hulgas üles laadida. Arvestada tuleb vaid piirangutega, et kasutatavad teegid peavad olema puhtas *Pythonis* (eelkompileeritud C-keeles kirjutatud teegid *Google App Engine*'i platvormil ei tööta) ning järgima erinevaid keskkonna poolt ette kirjutatud reegleid (näiteks keeldu avada ühendusi teistesse serveritesse muudmoodi kui *URLFetch* teenuse abil).

6. Konfiguratsioonifailid

Google App Engine'i rakenduste konfiguratsioonifailides (rakenduse andmed, andmebaasi indeksite deklaratsioonid, perioodiliste tööde kirjeldused jms) on kasutusel *YAML* formaat [16]. Tegu on inimloetava formaadiga, mida saab toimetada tavalise tekstiredaktoriga, võimaldades mugavalt määrata erinevaid seadeid ühes kindlas asukohas. Kuna *Pythoni* skriptid saavad *Google App Engine*'i rakendustes yaml teegi ise sisse laadida (käsklus `import yaml`), saab sama formaati kasutada ka rakenduse spetsiifiliste seadete salvestamiseks.

Nimetus *YAML* tähendab rekursiivset akronüümi sõnadest *YAML Ain't Markup Language*. Tegu on millegi *INI*-faili laadsega, kuid *YAML* on erinevalt *INI*-failist tunduvalt laiemate võimalustega. Kui *INI*-fail on sisuliselt vaid „võti-väärtus" paaride kogum, kus väärtuseks on number või tekst ning maksimaalseks liigendatuseks saab olla nende väärtuste sektsioonidesse jagamine, siis *YAML* võimaldab teksti kujul esitada ka väga keerulisi andmestruktuure nagu näiteks programmi objektid.

Juhul kui on soov rakendusesiseselt *YAML* faili kasutada, toimub failide lugemine mooduli yaml abil, kus *YAML* formaadis dokument teisendatakse *Pythoni* objektiks.

Konfiguratsioonifail *seaded.yaml*:

```
# See on kommentaar
Rakendus:
  pealkiri: Näidisrakendus
  sildid:
  - GAE
  - Python
  - YAML
```

Programmikood andmete lugemiseks:

```
import yaml
fp = open("seaded.yaml")
seaded = yaml.load(fp, Loader=yaml.Loader)
pealkiri = seaded["Rakendus"]["pealkiri"] # Näidisrakendus
sildid   = seaded["Rakendus"]["sildid"]   # ["GAE", "Python", "YAML"]
```

Nagu näha, on *YAML* formaadis lihtne edastada ka massiive, mida *INI* failiformaadi puhul on teha suhteliselt raske, rääkimata keerulisematest andmestruktuuridest.

Kuigi *Google App Engine* ise *INI*- faile ei kasuta, on soovi korral siiski võimalik rakendusesiseselt neid kasutada. Selleks annab mugava võimaluse *Pythoni* moodul `ConfigParser`.

Konfiguratsioonifail *seaded.ini*:

```
; See on kommentaar
[Rakendus]
pealkiri = Näidisrakendus
```

Programmikood andmete lugemiseks:

```
import ConfigParser
config = ConfigParser.RawConfigParser()
config.read('seaded.ini')
pealkiri = config.get('Rakendus', 'Pealkiri') # Näidisrakendus
```

Kuna *YAML* on samas tunduvalt võimekam kui *INI* ning *Google App Engine*'i poolt nõutav konfiguratsioon peab niikuinii asuma *YAML* formaadis, on mõistlik eelistadagi vajadusel just seda.

7. Andmebaas

Google App Engine ei kasuta mõnd tavakäibes olevat andmebaasi nagu näiteks *MySQL*, *MSSQL* või *Oracle*, vaid *Google*'i enda poolt välja töötatud mitte-relatsioonilist andmebaasi *BigTable*. Andmete salvestamiseks pole alternatiivina võimalik kasutada isegi mitte tekstifaile (*Google App Engine* on keelanud rakendustele kõik kettale kirjutamise õigused) ning seega tuleb kogu püsiv info (ajutise info jaoks saab kasutada ka näiteks memcache teeki) teenuse poolt määratud andmebaasi salvestada.

Google App Engine'i käsutuses on *BigTable* andmebaasiga suhtlemiseks *Datastore API*. Antud *API* võimaldab defineerida andmebaasitabeleid, lisada andmebaasi uusi kirjeid ja neid sealt lugeda. Andmete lugemine ongi *BigTable*'i üheks suurimaks plussiks – sõltumata andmebaasis olevate andmete hulgast, on lugemine alati kiire. Üks peamine kiiruse võimaldaja on tõenäoliselt andmebaasi lihtsakoelisus võrreldurna näiteks *MySQL* andmebaasiga: puudub võimalus kasutada *JOIN*-lauseid, *WHERE* -parameetrid on äärmiselt piiratud ulatusega, kõik päringud peavad olema eelnevalt indekseeritud (lihtsamad indeksid lisab *Google App Engine* ise, kuid keerukamad tuleb defineerida failis `index.yaml`) jne.

Vaatamata sellele, et lugemine on kiire, on andmebaasi kirjutamine ometi väga aeglane. *Google App Engine* on hajutatud teenus, mis tähendab, et rakenduse andmed asuvad korraga mitmes eri serveris – seega igasuguste andmete kirjutamisel tuleb need erinevate sihtkohtade vahel tiražeerida. Lisaks nõuab oma aja indekseerimine. See aga teeb kirjutamise suhteliselt aeglaseks, muutudes suuremate andmemahtude korral tõsiseks probleemiks. Juhul kui on vaja korraga lisada palju kirjeid, on üsna tõenäoline, et enne lõpeb skriptile antud aeg, kui kirjed baasi saavad sisestatud. Kusjuures juttu ei ole mitte tuhandetest, vaid sadadest kirjetest. Kohati muudab andmebaasi kasutamise ebamugavaks asjaolu, et kirje maksimaalseks suuruseks saab olla 1 MB.

Poolikute andmete sisestamise vastu on mõningane abi andmebaasi transaktsioonidest – juhul kui andmebaasiga suhtlemine on vormistatud transaktsioonina, siis ühe päringu ebaõnnestumisel võtab server tagasi kõik transaktsiooni käigus tehtud muudatused.

7.1. Kasutamine

Andmebaasiteenuse kasutamiseks tuleb laadida teek db.

```
from google.appengine.ext import db
```

Järgmisena tuleks defineerida andmebaasitabelite mudelid. Seda saab teha vastavate klasside loomise abil, kus klassi omadustest saavad tabeli väljad. Kui *PHP* ja *MySQL* puhul ollakse

harjunud selle jaoks kasutama mõnd tööriista nagu *phpMyAdmin*, siis *Google App Engine*'i puhul pole vaja nii sügavuti minna – piisab vaid klassi defineerimisest ning kogu „raske töö" teostab andmebaasiteenus.

```
class Teade(db.Model):
  saatja = db.StringProperty()
  sisu = db.TextProperty()
  aeg = db.DateTimeProperty(auto_now_add = True)
```

Selliselt luuakse db.Model klassist pärinev andmebaasitabel nimega Teade, millel on kolm välja – stringiväli saatja, tekstiväli sisu ja ajaväli aeg. Parameeter auto_now_add, mis asub väljas aeg, tähendab, et uue kirje lisamisel seatakse selle välja väärtuseks automaatselt hetke aeg. Mudeli alusel loodud andmebaasiobjekt sisaldab lisaks defineeritud omadustele veel meetodeid put ja delete, mis võimaldavad objektiga seotud andmebaasikirjet kas lisada, muuta või kustutada.

7.2. Andmete lisamine

Uute kirjete lisamine andmebaasi on mugav andmebaasielemendi objekti kujul. Selleks tuleb luua tabeli klassi tüüpi objekt (näites Teade) ning määrata selle omaduste väärtused. Pärast seda on võimalik kirje andmebaasi salvestada.

```
teade = Teade()
teade.saatja = "Peeter Meeter"
teade.sisu = "See on teade"
db.put(teade)
```

Sellise tegevuse tulemusena lisatakse andmebaasi kirje:

key	key_name	Id	saatja	sisu	aeg
…	…	…	Peeter Meeter	See on teade	23.02.2010 17:59:00

Tabel 2. Sisestatud kirje andmebaasis

Tabelist on näha, et kirjega on seotud mitte üks, vaid lausa kolm erinevat võtmeväärtust. *Google App Engine*'i andmebaas võimaldabki kasutada kolme tüüpi võtmeid, mille alusel päringuid teha. Nendest kaks on seatud andmebaasi enda poolt ja ühte saab seada programm.

- key – tekstiline võti. Seatud igale kirjele andmebaasi poolt automaatselt ning on unikaalseks võtmeks terve andmebaasi raames.
- id – numbriline võti, mis on seatud andmebaasi poolt ja on unikaalne konkreetse tabeli raames.
- key_name – tekstiline võtme nimi, mis on seatav programmi poolt ning on unikaalne tabeli raames. Juhul kui luuakse uus element, millel on sama key_name väärtus nagu mõnel olemasoleval kirjel, siis olemasolev kirje kirjutatakse üle.

7.3. Andmete lugemine

Andmete lugemiseks saab kasutada *SQL* laadseid *GQL* [17] päringuid, võtme alusel kindla kirje laadimist või alternatiivina aktiivsete salvestusobjektide (ingl *Active Records*) põhist filtreerimist. Kõikidel juhtudel on tulemuseks andmebaasi tabeli tüüpi andmebaasiobjektid, mida saab vahetult toimetada ning put ja delete meetodite abil ka andmebaasis uuendada.

GQL päring:

```
query = db.GqlQuery("SELECT * FROM Teade WHERE saatja = :1 ORDER BY aeg DESC",
"Peeter Meeter")
teated = query.fetch(100) // kuni 100 esimest elementi
```

Aktiivse salvestusobjekti päring:

```
query = Teade.all()
query.filter('saatja =', 'Peeter Meeter')
query.order("- aeg")
teated = query.fetch(100) // kuni 100 esimest elementi
```

Võtme järgi kindla kirje päring:

```
teade = Teade.get(key)
```

Kuna näitena kasutatud *GQL* ning aktiivse salvestusobjekti päringute puhul tuleb arvestada korraga enam kui ühe väljaga (oluline on nii `saatja` kui `aeg`), peab `index.yaml` faili lisama ka järgmise indeksi deklaratsiooni. Ilma vastava indeksita tekitab päring veateate.

```
- kind: Teade
  properties:
  - name: saatja
  - name: aeg
    direction: desc
```

Vaikimisi toimub andmebaasist lugemine tugeva konsistentsuse põhimõttel – rakendus pöördub kõikide andmebaasioperatsioonide sooritamiseks ühe primaarse andmebaasiserveri poole, tagades sellega alati andmete ühtsuse. Kui aga kasutatav server ei ole parasjagu kättesaadav, tähendab see automaatselt rakenduse töö peatumist. Alternatiivina on võimalik andmeid lugeda ka edaspidise konsistentsuse põhimõttel, kus andmete lugemiseks pöördutakse mõne sekundaarse andmebaasiserveri poole. Sellisel juhul võivad andmed olla siiski veidi vananenud – tiražeerimisele kulub mõnesajast millisekundist paari sekundini [18].

7.4. Andmete varundamine ja taastamine

Andmebaasi sisu on võimalik varundada vastava tööriistaga, milleks on *SDK* paketiga kaasatulev skript `bulkloader.py`. Antud programm suudab rakenduse andmebaasist korraga kõik kirjed alla laadida ning need faili salvestada. Samuti oskab sama programm varundatud andmetega serveris endist seisu taastada.

Andmete varundamise võimaldamiseks tuleb lisada `app.yaml` faili kirje, mis oskab suunata varundamispäringud õige programmi juurde. Juhul kui vastavat konfiguratsioonikirjet seatud pole, ei saa andmeid programmiliselt varundada ega taastada.

```
- url: /remote_api
  script: $PYTHON_LIB/google/appengine/ext/remote_api/handler.py
  login: admin
```

Andmete varundamine faili toimub järgmise käsklusega:

```
bulkloader.py --dump --app_id=<app-id> --url=<host>/remote_api --filename=<file>
```

kus `<app-id>` on rakenduse identifikaator, `<host>` rakenduse serveri aadress, näiteks *http://app_id.appspot.com* ja `<file>` on failinimi, kuhu andmed salvestatakse.

Andmete taastamine varundusfailist:

```
bulkloader.py --restore --app_id=<app-id> --filename=<file> <kataloog>
```

kus <app-id> on rakenduse identifikaator, <file> on failinimi kus asuvad salvestatud andmed ning <kataloog> on kataloog, kus asuvad rakenduse failid (sellest kataloogist otsitakse faili app.yaml).

Varunduseks kasutatav fail on *SQLite* 3 [19] tüüpi andmebaas ning selles asub kõiki varundatud andmeid sisaldav tabel `results`. Tabeli read koosnevad kirje unikaalsest võtmest ning *Blob*-tüüpi väärtusest, milles seisab kirje tegelik sisu serialiseeritud andmebaasiobjektina *Protocol Buffers* formaadis.

Täiendava konfiguratsiooniga on andmeid võimalik importida ja eksportida *Google App Engine*'i andmebaasi ka *CSV* formaadis. Samuti saab käidelda kõiki tabeleid ükshaaval – eelnevalt näidatud käsud varundavad ja taastavad kõiki rakenduse tabeleid korraga.

8. Teenusespetsiifilised teegid

Kui programmeerimiskeel *Python* 2.5 sisaldab juba iseenesest suhteliselt laias ulatuses erinevaid abistavaid teeke ning vaikimisi on kasutada suur hulk *Django* mooduleid, siis sellele lisanduvad mõned teenusespetsiifilised teegid, abistamaks rakendusi *Google App Engine*'i platvormil paremini hakkama saama.

Näiteks saavad rakendused kasutada mugavat võimalust *Users* teegi abil tuvastada sisseloginud kasutajaid *Google'i Kontod* liidese abil – kasutaja saab rakendusse sisse logida sama kasutajanime ja parooliga, millega ta loeb oma *GMail* e-posti. Mainimist väärivad ka andmete puhverdamise teek *Memcache* ja suuremate tööde tegumiteks jagamise teek *Tasks*. Järgmisena vaatamegi põgusalt mõningaid selliseid teeke.

8.1. Google'i Kontod

Google App Engine pakub arendajatele mugavat kasutajate haldust *Google'i Kontod* [20] liidese abil, nii pole endal vaja kogu vajalikku infrastruktuuri üleval pidada. Sama kasutajanime ja parooliga, millega kasutaja siseneb oma *GMail* postkasti või kirjutab *Blogger* teenuses blogipostitusi, saab ta sisse logida ka ükskõik millisesse *Google App Engine*'i platvormil paiknevasse rakendusse.

Google'i Kontod kasutamiseks tuleb sisse laadida `users` teek.

```
from google.appengine.api import users
```

Programmi käigus saab kontrollida sisseloginud kasutaja staatust käsuga `users.get_current_user()` – sellega saab kätte sisseloginud kasutajaga seotud info või väärtuse `False`, kui kasutaja pole sisse logitud. Sisselogimisvormi aadressi, millele kasutaja enda tuvastamiseks suunata, saab genereerida meetodiga `users.create_login_url(sihtkoha_url)`.

8.2. Memcache

Memcached [21] on lihtne „võti-väärtus" paaridel põhinev andmete puhverdamise süsteem, mis pakub võimalust hoida rakendustel puhverdatavaid andmeid mälus sessioonüleselt. Näiteks kui laetakse andmebaasist andmeid, siis on neid mugav säilitada järgmiste lehevaatamiste jaoks juba mälus, nii pole vaja kulukaid andmebaasipäringuid puhvri kehtivusaja jooksul teha.

Google App Engine kasutab modifitseeritud *Memcached* versiooni nimega *Memcache*. Selle kasutamiseks tuleb sisse laadida memcache teek.

```
from google.appengine.api import memcache
```

Andmete puhverdamiseks tuleb kasutada meetodit `memcache.set(võti, väärtus)` ja lugemiseks `memcache.get(võti)`. Puhverdada saab suvalisi andmestruktuure, sealhulgas näiteks andmebaasiobjekte.

8.3. E-post

Google App Engine võimaldab nii e-posti saatmist kui vastuvõtmist. Vastuvõtmiseks tuleb registreerida haldaja, millele suunatakse kõik tingimusele vastavad sisenevad e-kirjad (eri aadressidele laekuvad e-kirjad saab suunata eri haldajatele). Nii saatmine kui vastuvõtmine on tehtud programmeerija jaoks mugavaks – pole vaja näiteks spetsiifilisi teadmisi e-kirja tehnilisest ülesehitusest, e-kiri on *Google App Engine*'i rakenduse jaoks lihtsalt järjekordne kindlate omadustega objekt.

Google App Engine'i infrastruktuuri kuritarvitamise vältimiseks saab kirju välja saata ainult rakenduse administraatorite või hetkel sisse loginud kasutaja e-posti aadressidelt. See tähendab, et kuigi saatja nimeks võib panna suvalise tekstiväärtuse, siis kasutatavad saatja e-posti aadressid on väga piiratud.

8.4. Veebiaadressidelt sisu laadimine

Veebiaadressidelt sisu laadimiseks on kasutusel *Google App Engine*'i spetsiifiline teek `urlfetch`. Tugi on olemas ka teiste sarnaste *Pythoni* teekide jaoks (näiteks `urllib` või `urllib2`), kuid nende meetodid veebist sisu laadimiseks on tegelikult asendatud `urlfetch` teegi meetoditega – kõik andmete laadimised *Google*'i infrastruktuuris käivad ühtedel ja samadel alustel ning samu vahendeid kasutades.

```
from google.appengine.api import urlfetch
url = "http://www.neti.ee/"
result = urlfetch.fetch(url)
print result.content
```

Näites tehakse *GET*-päring aadressile http://www.neti.ee/ ning väljastatakse päringu vastusena saadud sisu ekraanile.

8.5. Piltide redigeerimine

Pildifaile on images teegi abil võimalik redigeerida suhteliselt piiratud ulatuses. Saab muuta faili tüüpi, suurust, pilti lõigata ja pöörata. Samuti saab kasutada automaatset värvide korrigeerimise funktsiooni. Pildifaile saab hoida nagu muidki andmeid andmebaasis, seega ei tohi fail ületada 1 MB suurust. Sellest piirangust on võimalik mööda saada *BlobStore*'i failihoidlat kasutades. Samas peab arvestama, et kuigi *BlobStore* failihoidlas olev pilt võib olla väga suur, siis sellist pilti images teegi abil töödeldes peab tulemus jääma ikkagi 1 MB piiridesse.

8.6. Tegumid

Tegumid on väikesteks ühikulisteks ülesanneteks jagatud suured tööd, mis edastatakse serverile ükshaaval täitmiseks. Sisuliselt on tegu millegi perioodiliste tööde laadsega, selle vahega, et kuigi tegumid käivitatakse samuti ükshaaval, siis mitte lõputult korrates kindla perioodi tagant, vaid nii

kiiresti, kui serveri hetkevõimalused lubavad, ning kuni kõik ülesanded täidetud saavad. Mõlemal juhul teeb server kindlaksmääratud veebiaadressidele päringuid, mis vajalikud programmid käivitavad.

Näiteks saab ühe suure töö jagada mitmeks väikeseks tööks ja need ükshaaval tegumitena serverile ette anda. Kohe kui serveril tekib vabu ressursse tööde tegemiseks, hakkab ta neid tegumeid ükshaaval täitma. Kui perioodilise töö puhul tehakse skriptile *GET*-päringuid, siis tegumi puhul tehakse *POST*-päringud, võimaldades sellega anda tegumile ette suuremahulisi parameetreid.

8.7. BlobStore'i failihoidla

BlobStore'i failihoidla võimaldab rakendustel hallata suuremaid faile, kui andmebaasikirjete 1 MB limiit seda muidu lubaks. Nii saavad kasutajad mahupiirangutest hoolimata suuri faile üles ja alla laadida. Teenuse ebamugavaks küljeks on asjaolu, et kuigi faili vastuvõttev rakendus pääseb üleslaetud faili meta-andmetele probleemideta ligi, siis selle sisu saab lugeda maksimaalselt 1 MB suuruste lõikude kaupa.

Kui kasutaja soovib faili alla laadida, edastab päringuga tegelev programm faili identifikaatori *BlobStore*'i teenusele ning *BlobStore* asendab päringule vastust genereerides identifikaatori esialgse failiga. *BlobStore*'i faili on võimalik kasutada ka pilditöötluse juures – sellisel juhul tuleb pildiobjekti luues edastada parameetrina faili sisu asemel viit *BlobStore*'i kirje juurde. Arvestada tuleb vaid sellega, et pilditöötluse tulemus peab jääma ikkagi 1 MB raamidesse.

9. Platvormide hübriidlahendus

Kuna *Google App Engine* sobib oma omadustelt kõige paremini siiski vahetult külastajaid teenindama, siis erinevate taustatööde jaoks tasub kaaluda vastavalt oludele mõne muu platvormi kasutamist. Taustatöödeks võivad olla näiteks pidevad tegevused, mis võtavad suhteliselt palju aega (rohkem kui 30 sekundit) ning nõuavad kiiret (kirjutamise) ligipääsu andmebaasile. Kuna taustatöid on võimalik planeerida – need ei teki reeglina ootamatult ning neid saab ka järjestada, võivad nõudmised vastavate serveriressursside suhtes jääda üsna tagasihoidlikeks.

Kulude kokkuhoiuks oleks mõistlik taustatööde tegemiseks kasutada mõnd odavat virtuaalserverit või virtuaalserverite komplekti (virtuaalserverite keskmine kuutasu on umbes 100 krooni), millel on küll palju võimalusi, kuid mis jääksid tavaolukorras suurte koormuste all hätta. Kui taustatööd on ajaliselt jaotatud, ei teki ühelgi ajahetkel serverile korraga väga suurt koormust ja see võimaldabki taolise odava lahenduse kasutusele võtta. Nii saab moodustada teenuste hübriidlahenduse, kus iga osapool teeb seda, mida paremini suudab.

Illustratiivseks näiteks võiks siinkohal olla internetiteenuse *Twitter* andmevoo [22] töötlus, kus *Twitter* edastab jälgitavate kasutajate kirjutatud sõnumid andmevoo kaudu selleks soovi avaldanud klientrakendustele. Klientrakendus saab sõnumid *Twitteri* andmevoo kaudu kätte praktiliselt reaalajas ning saab sellega võimaluse sõnumeid kiirelt töödelda, kuvamaks neid veebilehel või lihtsalt indekseerides tuleviku päringute tarbeks.

Andmevoo vastuvõtmiseks tuleb avada *Twitteri* serverisse püsiv ühendus üle *HTTP* protokolli, kus ühenduse avamise järel saab jääda kuulama selle kaudu saabuvaid andmeid. Andmevoog kujutab endast sisuliselt lõputu suurusega veebilehte, mille allalaadimine võtab lõputult aega. Algab see tavalisest *HTTP* päisest, kus on jäänud märkimata *Content-Length* ehk laetava faili suuruse väärtus – kuna sõnumeid edastatakse reaalajas ning ühenduse avamise hetkel neid veel ei

eksisteeri, ei saagi see väärtus enne ühenduse lõpetamist teada olla. Pärast päist hakkavad reaalajas saabuma kasutajate kirjutatud sõnumid ning kui parasjagu ühtegi sõnumit näidata pole, saadetakse kliendile ühenduse elushoidmiseks reavahetuse sümboleid. Vastuvõttev programm laeb andmevoogu pidevalt alla, ise samal ajal uurides, kas viimati saabunud lõigu sees pole ka mõnd sõnumit.

```
HTTP/1.1 200 OK
Connection: Keep-Alive
....

sõnum1
\n
\n
sõnum2
....
```

Google App Engine sellist andmevoo töötlemist ei võimalda, kuna *Google App Engine* lubab programmil töötada maksimaalselt vaid 30 sekundit ning välisühendusi saab teha ainult läbi *URLFetch* teenuse, mis laeb veebist alla terve faili (kuni 1 MB mahus), sulgeb ühenduse ning edastab alles siis kõik laetud andmed korraga programmile. Andmevoo ühendust on aga vaja üleval hoida mitte 30 sekundit, vaid soovituslikult lõpmatult. Kui *Google App Engine*'i korral on antud töö sooritamine probleemne, siis ometi sobib see hästi suvalisele virtuaalserverile, millel on *PHP* tugi ning mis võimaldab programmil ise seada ja eemaldada oma käivitusaja piirangut.

```
set_time_limit(0);
ignore_user_abort(true);
....
```

Antud *PHP*-keelne näide eemaldab programmilt käivitusaja piirangu (`set_time_limit(0)`) ning võimaldab tööd jätkata isegi juhul, kui programmi käivitanud kasutaja on omalt poolt juba töö lõpetanud (`ignore_user_abort(true)`). Kui selline programm käivitada, nii et see avab ühenduse *Twitteri* andmevoo teenusele, siis programm jääbki käima, sobides sellega suurepäraselt oma eesmärki täitma.

Loomulikult ei pruugi ühendused kesta lõputult. Katsetused näitavad, et ühendus kaob vähemalt korra päevas, kuid ühenduse olemasolu saab perioodiliste töödega erinevate meetodite abil pidevalt jälgida ning katkemise korral kiirelt taastada. Kuna *Twitter* võimaldab ajatempli alusel ka tagantjärele sõnumite voo taastamist, saab nii garanteerida programmi pideva töö, kuna ükski edastatud sõnum ei lähe kaotsi ning jõuab halvemal juhul kohale vaid paariminutilise viivitusega. Tänapäevased virtuaalserverite pakkujad suudavad tagada üsnagi hea serverite töövõimeaja – seda ei kinnitata küll *SLA* lepinguga, kuid praktilisest küljest on tegu täiesti piisava ressursiga.

Kui programm võtab andmevoo kaudu vastu uue sõnumi, saab ta esialgse töötluse teostada ise ning edastada selle siis *Google App Engine*'i serverile, mis teenindab juba lõppkasutajaid. Üksikute sõnumite sisestamine *Google App Engine*'i andmebaasi ei ole teenusele eriti koormav ning andmebaasist sõnumite lugemine ja kasutajatele esitamine käib väga kiirelt.

Küsimus on, millises formaadis oleks kõige parem andmeid serverilt serverile edastada. Kuna *HTTP* päringutega saab edastada ainult teksti, tuleb andmeobjektid eelnevalt kuidagi serialiseerida. Arvestada tuleb sellega, et kiiruse (nii andmeedastuse kui programmeerimise mõistes) saavutamiseks peaks tegu olema võimalikult väikesemahulise ning kiirelt töödeldava tulemusega.

Üheks laiemalt levinud formaadiks on kujunenud *SOAP* protokolli abil *XML*-formaadis andmete edastus. *Google App Engine* ise kasutab platvormisiseselt vastavaks otstarbeks *Protocol Buffers* formaati, mis on ülikompaktne ning väga kiirelt töödeldav (*Google*'i andmetel 20 kuni 100 korda kiirem [23] kui samade andmete *XML*-formaadis töötlemine). *Python* toetab andmete serialiseerimiseks vaikimis *pickle*-formaati [24], *Google App Engine* pakub *Pythonile* lisaks *YAML* tuge. Laiemat kasutuspinda on hakanud koguma ka *Facebooki* poolt arendatud *Thrift* [25].

Kuna eelpoolmainitud näite puhul peaksid omavahel suhtlema *PHP* ja *Python* ning tõenäoliselt ka veebibrauseri poolel *JavaScript*, oleks mõistlik valida nende platvormide madalaim ühisnimetaja. Andmevahetusformaadi puhul võiks selleks olla *JSON* [26], mida toetavad kõik mainitud platvormid ning mis on väga kergelt ja kiiresti töödeldav. Andmete saatmiseks teenuselt teenusele oleks täpsemalt tegu *JSON-RPC* [27] kaugkutse protokolliga (hetkel aktiivne versioon 1.0). Antud protokolli suureks plussiks on üsna väike protokolli ballast, sisaldades endas vaid paari juhtimisväärtust, millest peamine on kaugkutse meetodi nimetus.

PHP võimaldab *JSONi* serialiseerimist käskudega `json_encode(object) → string` ja `json_decode(string) → object`. *Google App Engine*'i *Pythoni* puhul saab kasutada *Django* teeki `simplejson`, mille saab importida paketist `django.utils` ja millega saab serialiseerimist läbi viia käskudega `simplejson.dumps(object) → string` ja `json.loads(string) → object`.

JSON-RPC päringud on nimele vastavalt *JSON*-formaadis, sisaldades kolme parameetrit:

- `method` – kaugkutse meetodi tekstiline nimetus
- `params` – meetodile edastatavate parameetrite massiiv
- `id` – päringu ID, mis tagastatakse koos vastusega

Vastus võib sisaldada samuti kolme parameetrit:

- `result` – vastusobjekt (juhul kui ilmnes viga, peab see väärtus olema seadmata)
- `error` – veaobjekt juhul kui ilmnes viga, vastasel korral seadmata
- `id` – päringu ID, mis anti päringut tehes

Näiteks *Twitteri* sõnumi edastamine *JSON-RPC* protokolli abil võiks välja näha järgnevalt:

```
-->{"method":"uusTekst",params:["Peeter Meeter","Tere õhtust"],"id":1}
<--{"result":"Sõnum on lisatud","error": null,"id":1}
```

Sellisel juhul kutsub *PHP* esile *Google App Engine*'i rakenduses kaugkäsu `uusTekst` käivitamise parameetritega „Peeter Meeter" (kasutaja nimi) ja „Tere õhtust" (sõnumi sisu). *Google App Engine* vastab teatega „Sõnum on lisatud".

Lihtne *JSON-RPC Pythoni* implementatsioon *Google App Engine*'i platvormil näeks välja järgmine. Defineeritud on kaks klassi – `RPCMethods`, mis sätestab kasutatavad kaugkutsekäsud ning `RPCHandler`, mis võtab vastu *HTTP POST*-päringu ja käivitab vastavalt päringule kaugkutsekäsu `RPCMethod` klassist.

```
from django.utils import simplejson

class RPCMethods:
  def uusTekst(self, args, rpc_id):
```

```
    nimi = args[0]
    sonum = args[1]
    ....
    return {"result":u"Sõnum on sisestatud",
      "error": None,
      "id":rpc_id}

class RPCHandler(webapp.RequestHandler):
  def post(self):
    args = simplejson.loads(self.request.body)
    func, args, rpc_id = args["method"], args["params"], args["id"]
    func = getattr(RPCMethods(), func, None)
    result = func(args, rpc_id)
    self.response.out.write(simplejson.dumps(result))
```

Antud lahenduse puhul tuleb siiski meeles pidada, et tegu on küll töötava, kuid siiski lihtsustatud ja ebaturvalise näitega – kaugkutsekäsu käivitamisel tuleks muu hulgas kontrollida, kas käsu nimetus ei alga allkriipsuga _ (klassi privaatsed meetodid), samuti puudub igasugune veahaldus.

10. Kokkuvõtteks

Kuigi *Google App Engine*'il on veel mitmeid vajakajäämisi, tasub uue veebirakenduse loomisel selle platvormi kasutamist kindlasti kaaluda. Viimase kahe aasta jooksul on teenuselt lihvitud maha nii mõnigi teravam nurk ja see trend paistab jätkuvat. Seega tänane piirang võib täiesti vabalt olla homseks ajalugu – nii on juhtunud näiteks algselt täielikult puudunud suurte failide toega ning *wildcard* alamdomeenidega. Samuti on tööplaanis ette nähtud praeguse hetke ühe suurima piirangu lahendamine, milleks on *HTTPS* toe lisamine oma domeeni kasutamisel.

Positiivse külje pealt võimaldab *Google App Engine* luua väga suurelt skaleeruvaid rakendusi minimaalsete investeeringutega. Rakenduse tööks pole vaja rentida ega osta kallist riistvara, pole tarvis seda platvormi hooldada ega maksta liitumis- või perioodilisi hooldustasusid. Hinnastamine on täiesti läbipaistev ning mingi piirini on teenus hoopis tasuta.

Mina jõudsin *Google App Engine*'i juurde, kui otsisin sobivat platvormi SMS-maksete kogumise rakendusele *SMS-Publisher* [28]. *SMS-Publisheri* näol on tegemist omamoodi vahekihiga mobiilimaksete platvormi Fortumo [29] ja kasutajate vahel, kes saavad läbi antud teenuse kasutada Fortumo keerulisemaid võimalusi ilma programmeerimisest midagi teadmata. Teenust kasutatakse praeguseks aktiivselt Lätis, Horvaatias, Rumeenias ning isegi Malaisias – mitte ükski teine platvorm peale *Google App Engine*'i poleks suutnud pakkuda sama hinna ja kvaliteedi suhtega niivõrd laia geograafilist ala katvat lahendust.

Rakenduse arendamisega alustamine võib olla küll suhteliselt tülikas – tuleb näiteks väga täpselt jälgida ja arvestada andmebaasi piiranguid, mida muude platvormide puhul teha vaja ei ole, kuid selle tulemuseks on väga töökindel rakendus. Juhul kui rakendus töötab juba saja andmebaasi kirjega, siis saab see sama hästi hakkama ka miljoniga.

Andris Reinman
Tarkvara disainer, NETI
TLÜ IT-juhtimise magistrant

11. Viited

[1] http://code.google.com/intl/et/appengine/ – *Google App Engine*
[2] http://www.gmail.com/ – *Google Mail*
[3] http://aws.amazon.com/ec2/ – *Amazon Elastic Compute Cloud (EC2)*
[4] http://en.wikipedia.org/wiki/Slashdot_effect – *Slashdot effect*
[5] http://aws.amazon.com/ – *Amazon Web Services*
[6] http://www.caucho.com/resin-3.0/quercus/ – *Querqus*
[7] http://en.wikipedia.org/wiki/Https#Limitations – Wikipedia kirje *HTTPS* piirangute kohta
[8] http://code.google.com/intl/ko-KR/appengine/docs/roadmap.html – *Google App Engine* tööplaan
[9] http://appscale.cs.ucsb.edu/ – *AppScale*
[10] http://www.google.com/datacenter/hamina/index.html – *Google*'i andmekeskus Soomes
[11] http://code.google.com/intl/et/appengine/downloads.html – *Google App Engine SDK*
[12] http://www.appspot.com/ – *Google App Engine*'i rakendused
[13] http://www.google.com/apps/intl/en/group/index.html – *Google Apps Standard edition*
[14] http://www.djangoproject.com/ – *Django web framework*
[15] http://www.djangoproject.com/documentation/0.96/templates/ – *Django template language*
[16] http://www.yaml.org/ – *YAML Ain't Markup Language*
[17] http://code.google.com/appengine/docs/python/datastore/gqlreference.html – *GQL*
[18] http://googleappengine.blogspot.com/2010/03/read-consistency-deadlines-more-control.html – *GAE Datastore Read consistency*
[19] http://www.sqlite.org/version3.html – *SQLite* versioon 3
[20] https://www.google.com/accounts/ – *Google'i Kontod* teek kasutajate tuvastamiseks
[21] http://www.memcached.org/ – *Memcached* andmete puhverdamise moodul
[22] http://apiwiki.twitter.com/Streaming-API-Documentation – *Twitter Streaming API*
[23] http://code.google.com/intl/et/apis/protocolbuffers/docs/overview.html#whynotxml – *Protocol Buffers Developer Guide*
[24] http://docs.python.org/library/pickle.html – *Python object serialization*
[25] http://incubator.apache.org/thrift/ – *Apache Thrift*
[26] http://www.json.org/ – *JavaScript Object Notation*
[27] http://json-rpc.org/ – *JSON-RPC*
[28] http://www.sms-publisher.com/ – *SMS-Publisher*
[29] http://fortumo.ee/ – Fortumo