Miina Rautakorpi, 291552

Markus Hautala, 283551

Valto Moisio, 268644

Rinnakkaisuus harjoitustyö 3: Threads and Mutexes

Löysimme ohjelmasta muutaman ison rinnakkaisuuteen liittyvän ongelman. Ensimmäinen oli selvä, sillä ohjelman ajamiseen kului kohtuuttoman pitkä aika, jopa 30 sekuntia. Tämä johtui yksisäikeisestä ohjelmasta, joka joutui toteuttamaan käskyt peräkkäin, jolloin ohjelman kestokin venyi. Tämän ongelman sai ratkaistua jakamalla ohjelman moneen eri säikeeseen. Säikeistyksen toteuttamiseksi loimme ThreadPool –luokan, joka huolehtii tehtävien suorittamisesta eri säikeissä. Eri säikeet pystyvät toimimaan saman aikaisesti, jolloin myös ohjelman ajo lyhenee merkittävästi. Neljän säikeen kanssa mitatun ohjelman ajoaika oli keskimäärin 5 sekuntia. Yhdellä säikeellä ajettu ohjelma oli siis noin kuusi kertaa säikeistettyä ohjelmaa hitaampi. Ajat mitattiin Tunin Linux-virtuaalityöpöydällä.

Toinen havaitsemamme ongelma oli detect-funktio, joka ei ollut säieturvallinen (thread safe). Eri säikeet pääsivät muuttamaan funktiossa olevia muuttujia, minkä vuoksi ASSERT(DetectorCounter.load() == 1) lopetti ohjelman suorittamisen, kun ehto ei täyttynyt. Ongelma ratkaistiin laittamalla detect-funktion kutsuminen WhereIsAhto-funktiossa lukon sisälle. Näin eri säikeet eivät pääse vaikuttamaan ristiriitaisesti detectin muuttujiin ja ohjelma toimii kuten pitääkin.

Kolmas löytämämme ongelma oli CountLocation-funktiossa. Funktiossa lisätään havaintojen määriä toiseen tietorakenteeseen. Laitoimme myös tähän funktioon lukon, jotta eri säikeet eivät pääse vaikuttamaan havaintojen määrään samaan aikaan.

Ohjelman dokumentaatio löytyy gitistä.