EXAMENSARBETE Multi-threaded execution of Cypher queries
STUDENT Ragnar Mellbin, Felix Åkerlund
HANDLEDARE Per Andersson (LTH), Johan Svensson (Network Engine for Objects in Lund AB)
EXAMINATOR Flavius Gruian (LTH)

Parallelliserad sökning i grafdatabas

POPULÄRVETENSKAPLIG SAMMANFATTNING **Ragnar Mellbin, Felix Åkerlund**

Grafdatabaser blir allt vanligare, samtidigt som antalet processorer i moderna datorer ökar mer och mer. Vi tittar i detta arbete på hur parallelliserad sökning kan leda till prestandavinster i den populära grafdatabashanteraren Neo4j.

Datorer är idag tusentals gånger kraftfullare än de var för tjugo år sedan, mycket tack vare de framsteg som gjorts i tillverkningen av centralprocessorn, den komponent som står för utförandet av alla logiska beräkningar i maskinen. På senare tid har dock fysikaliska begränsningar satt hinder för hur fort man kan köra processorn, därför har man istället börjat bygga datorer som innehåller flera processorer. Detta gör det möjligt för datorn att arbeta på flera uppgifter samtidigt. För att dra full nytta av denna sortens design, krävs dock att mjukvaruutvecklare skriver sina program så att de kan delas upp i mindre beståndsdelar som kan utföras parallellt.

En typ av databas som vuxit i popularitet på senare tid är den så kallade grafdatabasen. Grafdatabaser gör sig av med den traditionella tabellstrukturen och använder istället noder och bågar för att representera data. Ett exempel på data som lämpar sig för denna struktur är sociala nätverk som Facebook eller Twitter. Varje användare i nätverket kan representeras av en nod, medan vänskapsrelationer eller följare representeras av bågar som binder samman noderna.

Den populära grafdatabasen Neo4j har i dagsläget stöd för att besvara flera sökningar parallellt och klarar på så sätt av att utnyttja en modern processor fullt ut. Varje enskild sökning körs dock bara som en enda uppgift, vilket innebär att processorn kan ha delar som står outnyttjade om antalet aktiva sökningar är för lågt.

För simpla frågor som tar millisekunder att besvara så är detta sällan ett problem, speciellt när det finns tusentals användare som använder databasen samtidigt. Vill man däremot beräkna något tyngre, så som att analysera väldigt stora mängder data, är det vanligt att man sitter som ensam användare och kör något som tar flera timmar eller till och med dagar att få svar på. Det är i detta fallet intressant att se om man på något sätt kan få sökningen att utnyttja all den extra processorkraft som annars går till spillo.

För att ta reda på om det går att parallellisera en enskild sökning i en grafdatabas och hur stor påverkan detta då har på svarstider, skapade vi vår egen modifierade version av Neo4j. Vi började med att ta reda på vilka delar av mjukvaran som bäst lämpade sig för parallellisering, med hänsyn till hur ofta de förekom i sökningar samt hur pass stora krav de ställde på processorn. Efter att ha valt ut ett antal av dessa så gick vi vidare med att ta fram metoder för att dela upp dem i mindre uppgifter som kunde köras i olika delar av processorn samtidigt, för att slutligen införa dessa ändringar i Neo4j.

Resultatet är en version av Neo4j som under rätt förhållanden ger upp till 15 gånger snabbare svar på enskilda sökningar.