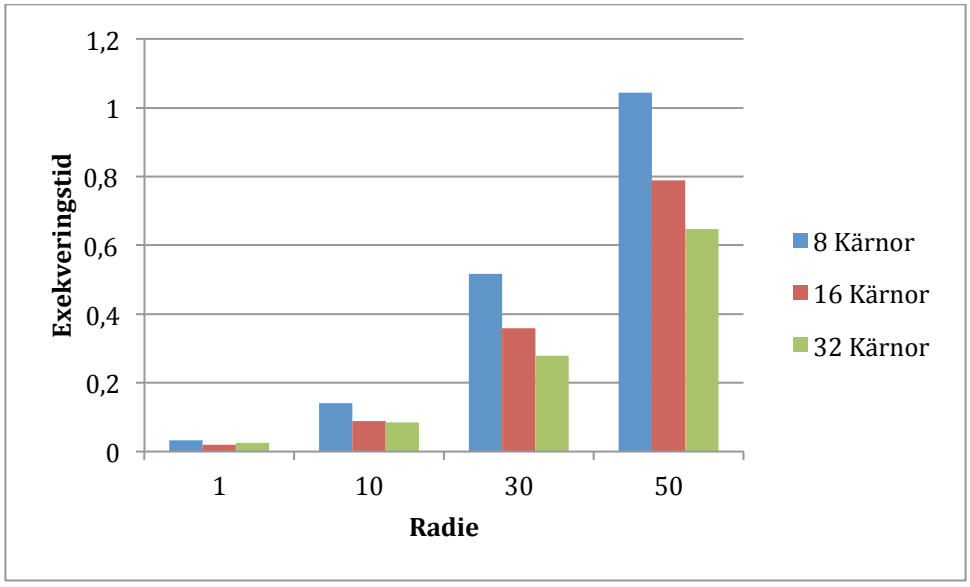
Christoffer Nylén, chrny719@student.liu.se
Alexander Norström, aleno278@student.liu.se

Avaraging filter

Först startas alla processer med hjälp av **MPI_Init**. Därefter kan rot-noden ladda in bilden och hämta ut storlek och data. Storleken skickas till alla andra processer med **MPI_Bcast**, så att de kan beräkna sina displacement och count vektorer. Dessa används för att sprida ut delar av bilden till de olika processerna med **MPI_Scatterv**. Vi använder vektor versionen för att vi behöver sprida ut lite extra rader runt de delar som processerna ska beräkna så att den inte tappar den data som skulle finnas där radien går utanför del-bilden. Nu kan alla processer beräkna blur faktorn på sin del av bilden. När detta är gjort, så återstår det att samla in alla delar av bilden exklusive de delar som las till för att radien skulle kunna kollas utanför. Alla dessa delar skickas till rot-noden med **MPI_Gatherv**. Efter detta är gjort så sparar rot-noden bilden till fil, och alla processer kan avslutas med **MPI_Bcast**.

Exekveringstider

I diagrammet Figur 1 ser vi att exekveringstiden blir bättre när vi ökar antalet kärnor. Vi ser även att vi tjänar ungefär lika mycket oberoende av vilken radie vi använder. Anledning till detta har att göra med att det krävs relativt mycket beräkningskraft för att räkna ut varje pixel. Eftersom vi inte har så mycket kommunikation mellan de olika delarna av bilden gör detta att problemet är praktiskt att parallellisera.



Figur 1: Exekveringstid för blurfiltret

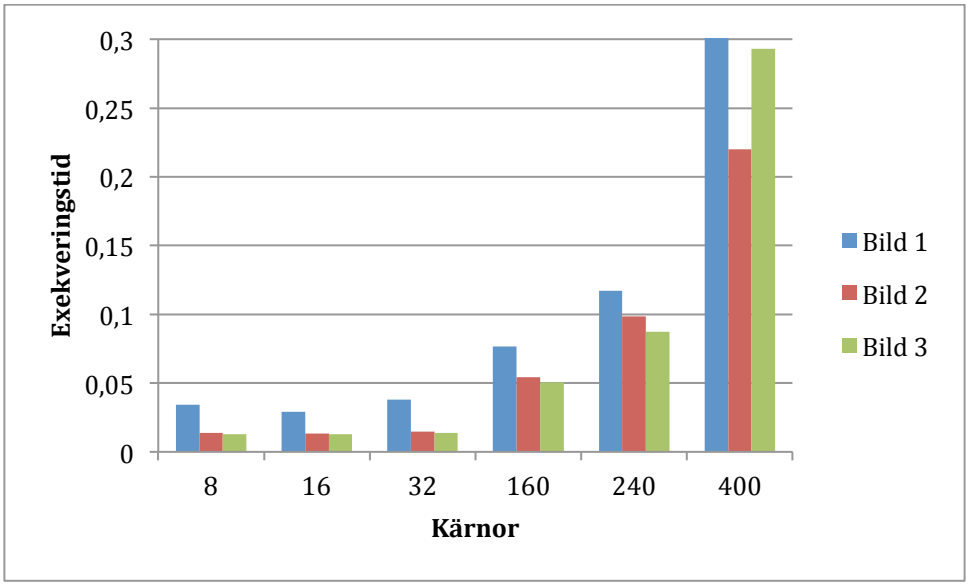
Christoffer Nylén, chrny719@student.liu.se
Alexander Norström, aleno278@student.liu.se

Thresholding

Först skickar vi delar av bilen med hjälp av **MPI_ScatterV**, till alla kärnor. Sedan beräknas medelvärdena på de olika delarna parallellt, vilka skickas tillbaka som en summa till roten med hjälp av **MPI_Reduce**. Därefter beräknar roten det slutgiltiga medelvärdet och använder **MPI_Bcast** för att skicka detta till alla noder. Varpå noderna skapar sina svartvita bilder vilka skickas tillbaka till roten med hjälp av **MPI_GatherV**.

Exekveringstider

Våra resultat visar att desto fler kärnor vi använder så ökar exekveringstiden enligt Figur 2. Detta beror på att vi inte har så mycket individuella beräkningar som måste utföras för varje pixel. Beräkningarna utförs så snabbt att den största delen av tiden spenderas till att kommunicera med de andra processerna för att dela med resultatet.



Figur 2: Exekveringstid för thresholdfiltret