# Program flöde

## 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**.

## Thresholding

Först startas trådarna med argument som talar om vilken del av bilden som ska beräknas. Varje tråd börjar med att beräkna medelvärdet på de delar som de har blivit tilldelade. Medelvärdet beräknas genom att summera ihop alla pixlar i varje del av bilden. Dessa summor måste sedan läggas ihop till en global summa för hela bilden. Detta utförs genom att skapa en delad variabel som håller summar och ett lås om skyddar denna. För att se till att inte trådarna rusar vidare så fort de har lagt till sin summa till den globala, så har vi en annan delad variabel som håller koll på hur många trådar som inte har tagit sig igenom medelvärdes summeringen ännu. I vår kritiska sektion ser vi till att kontrollera så att denna variabel är större än 0 och i sådana fall se till att tråden väntar där tills alla andra trådar är klara. När detta har skett så kan alla trådar lugnt rusa vidare och utföra förändringen av bilden givet det totala medelvärdet på bilden.

# Exekveringstider

Resultaten visade att det gick snabbare med fyra trådar än åtta. Anledning till det är att eftersom alla trådar körs på samma CPU med sekventiell åtkomst till minnet så kommer de övriga trådarna att behöva vänta när processorn läser minnet för en tråd. Om vi ökar till lika många trådar som kärnor så kommer varje tråd att vänta flera gånger om på att CPU spenderar sin största tid med att hämta data än vad det tar för varje kärna att bearbeta den.