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

Program flöde

Stationary Heat Conduction

För att parallellisera problemet delade vi upp problemet i flera delar. Där varje del hade N / P + 1 kolumner att beräkna, sista delen fick eventuellt lite färre rader beroende på avrundning.

Vi delade upp problemet innan den stora iterationsloopen. Detta av anledning att det är billigare att starta alla trådar samtidigt. Väll där passade vi även på att beräkna var trådarna skulle börja och sluta sina delar, beroende på problemets storlek och antalet trådar vi tilldelats av OpenMP.

Den parallelliserade delen kör samma sak i flera olika trådar och för att inte vissa variabler ska krocka med varandra fick vi se till att variablerna *id, istart, iend, tmp1, tmp2, tmp3*, samt loop variablerna *k* och *j* var privata.

Vi börjar med att spara undan den sista kolumnen i föregående del, samt den första kolumnen i nästa del. Efter det ser vi till att synkronisera med en barriär så att ingen tråd börjar utföra beräkningar innan vi har försäkrat oss om att varje del har sparat undan de kolumner som kolliderar vid beräkningen.

När vi beräknar kolumnerna i varje del ser vi till att flytta med oss föregående rad hela tiden, så att vi inte förlorar originalvärdet av den rad vi beräknar. När vi når sista raden ser vi till att beräkna den genom att använda den undansparade första raden i nästa del ifall den tråd som ansvarar för beräkningen av den delen skulle ha förstört den kolumnen redan.

Vi beräknar felet genom att ta den största skillnaden mellan varje element i den nya och gamla kolumnen. Vi sparar detta värde i en felmarginalsvektor.

Vi ser till att synkronisera med en barriär precis efter att alla kolumner har beräknats för att garantera att alla trådar har uppdaterat felmarginalsvektorn.

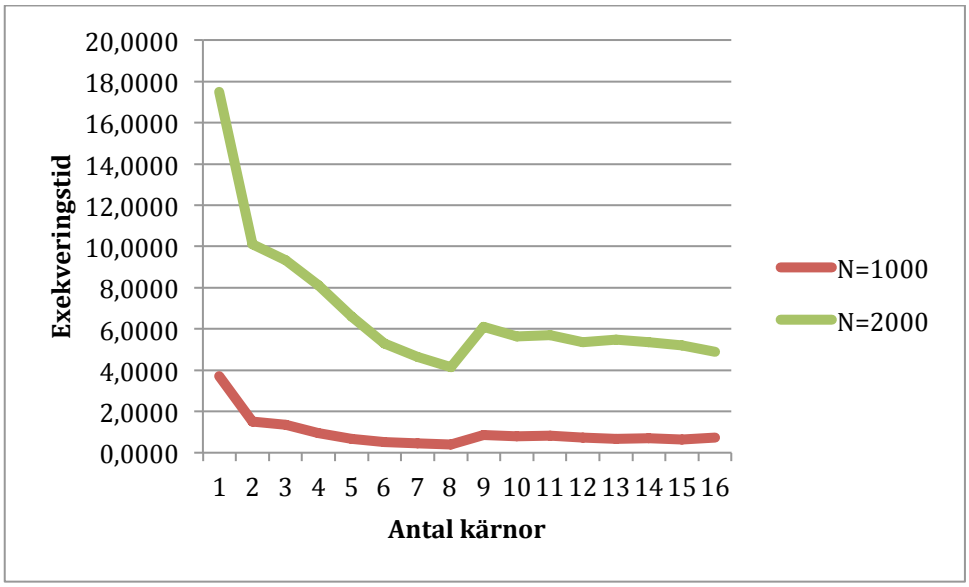
Därefter tar vi fram det största felet genom att utföra reduktion av största värden på felmarginalsvektorn. Detta värde jämförs sedan med det totala accepterade felet, och ifall felmarginalen är mindre än totala felet så är vi klara.

Annars utför vi allt ihop igen tills föregående villkor är uppfyllt eller att vi använt alla tillåtna iterationer.

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

Exekveringstider

Våra resultat visar tydligt att programmet är som mest optimalt att köras på åtta kärnor. Detta beror på samma sak som i föregående labb, eftersom processorerna delar minne och vi endast har åtta kärnor till förfogande har vi inget att vinna på att introducera fler trådar. Fram till 8 trådar har speedup en linjär ökning.



Figur 1 Exekveringstid för laplsolv

Figur 2 Speedup för laplsolv