Bilag 8

Fouriertransformation

Formålet med at teste fourier udregningen i 80 og 100 procent løsningen er at finde den opnåede forbedringen ved at skifte fra DFT til FFT.
Følgende to tabeller viser udsnit fra program profileringen (se bilag).

self seconds	calls	Name	Beskrivelse
2270.65	55440	Fourier::iDFT()	Funktion der udregner invers DFT.
1129.59	55440	Fourier::DFT()	Funktion der udregner DFT.

Tabel 1 Statistik for DFT algoritmen

self seconds	Calls	Name	Beskrivelse
41.14	166320	Fourier::FFTsub()	Funktion der rekursivt udregner FFT
0.0	55440	Fourier::FFT()	Starter den rekursive FFT udregning.
2.33	55440	Fourier::iFFTsub()	Funktion der rekursivt udregner invers FFT
0.0	55440	Fourier::iFFT()	Starter den rekursive invers FFT udregning.

Tabel 2 Statistik for FFT algoritmen

Navn	Beskrivelse
self seconds	Tallet angiver hvor lang tid der alt i alt er brugt i funktionen.
calls	Tallet angiver hvor mange gange funktionen er kaldt.
Name	Funktionen der er tale om.

Fra Tabel 1 kan man se at DFT og invers DFT står for 33,07% + 66,48% = 99,55% af den samlede udførsels tid, og at den inverse DFT tager længere tid at udregn end DFT.
Det beløber sig til 2270,65 sek. + 1129,59 sek. = 3400,24 sekunder for udregningerne.

Fra Tabel 2 kan man se tidsforbruget for FFT og invers FFT. Det er værd at bemærke at FFT iflg. tabellen bruger ca. 42 sekunder og invers FFT bruger ca. 3 sekunder. Forholdet mellem de 2 funktioners tidsforbrug er faktisk korrekt men ikke logisk. De burde bruge ca. lige lang tid.

Med de følgende 2 tabeller forklares tidsforbruget (se bilag 100profile-fft og 100profile-ifft)

self seconds	Calls	Name
21.48	55440	Fourier::FFTsub()
0.0	55440	Fourier::FFT()

Tabel 3 Statistik for FFT algoritmen

Fra tabellen kan man se at udregningen udføres præcist som man kunne forvente, FFT() bliver kaldt én gang for hver frame og den kalder FFTsub(), der er 55440 frames i alt.

self seconds	Calls	Name
19.14	110880	Fourier::FFTsub()
2.30	55440	Fourier::iFFTsub()
0.0	55440	Fourier::iFFT()

Tabel 4 Statistik for iFFT algoritmen

Her ser man at der under udregningen af iFFT bliver kaldt FFTsub() funktionen som faktisk var skrevet til at udregne FFT delen. Dvs. compiler der har bygget vores program har optimeret koden så følgende sker:
iFFT() kaldes én gang for hver frame, som forventet og kalder selv iFFTsub() én gang for hver frame, ganske som forventet. Her kommer så det sjove iFFTsub() som er skrevet til at være en rekursiv funktion benytter den anden rekursive funktion vi har skrevet til FFT-delen. Så FFTsub() kaldes to gange for hver frame (her regnes ikke med de kald funktionen selv laver). Der er 55440 frames * 2 = 110880, som er det antal gange FFTsub () bliver kaldt.

Det er nu blevet umuligt at sammenligne DFT med FFT og iDFT med iFFT ud fra koden til 100% løsningen. Men på baggrund af de data der findes for FFT og iFFT hver for sig har vi lavet en sammenligning. Resultaterne er meget tætte på det korrekte:
41.14 sek. i den samlede og 19.14 + 21.48 = 40.62 sek. i de to funktioner hver for sig, hvilket er en forskel på 0,52 eller ca. 1,26%.

Original wave fil:
27720 k

Original wave filanvn:
Aerosmith-stor

80% Løsning
Tid:
3415.72 sek

100% Løsning
Tid:
61.17 sek

Sammenligning:
Tidsforbedring:
55.84 gange

80% Løsning	100% Løsning	Sammenligning:
Tidsforbrug i DFT: 1129.59 sek	Tidsforbrug i FFT: 21.48 sek	Tidsforbedring: 51.92 gange*
Tidsforbrug i iDFT: 2270.65 sek	Tidsforbrug i iFFT: 21.44 sek	Tidsforbedring: 104.57 gange*
Tidsforbrug i alt: 3400.24 sek	Tidsforbrug i alt: 43.47 sek	Tidsforbedring: **78.22** gange

* Korrigeret for Afvigelse på
1.26%

Tabel 5 tabellerne viser tidsforbrug i de enkelte løsninger ved konvertering af ”aerosmith.wav”.