Example pag. 5

① $\frac{4,90}{1,43} = 3,42$

② $\frac{5,69}{1,35} = 4,21$

③ ⓐ $\frac{1,43}{0,695} = 2,05$ ⓑ $\frac{1,35}{0,96} = 1,40$

Esercizio 1 pag 13

$$\frac{1}{(1-0,4)+\frac{0,4}{10}} = \frac{1}{0,6+0,04} = \frac{1}{0,64} = 1,56$$

Esercizio 2 pag 14

Speed: 5x
Cost: 5x

$CPU_{time} = 50\%$
$CPU_{cost} = \frac{1}{3}$ of machine total cost

$$\frac{1}{0,5+\frac{0,5}{5}} = \frac{1}{0,6} = 1,66$$

$$5 \cdot \frac{1}{3} + \frac{2}{3} = \frac{7}{3}$$

costa più del doppio avendo uno speedup di molto inferiore

Esercizio 3 pag 16

$$\frac{1}{(1-x)+\frac{x}{100}} = 80$$

$$80 - 80x + \frac{80x}{100} = 1$$

$$80 - 79,2x = 1$$

$$79 = 79,2x$$

$$x = 0,997$$

$$x = 99,7\%$$

Deve rimanere sequenziale meno dello 0,3% del codice

$$\frac{1}{(1-x)+\frac{x}{20}} = 2$$

$$2 - 2x + \frac{x}{10} = 1$$

$$2 - \frac{19}{10}x = 1$$

$$\frac{19}{10}x = 1$$

$$x = 0,52$$

52 %

$$\text{Speedup} = \frac{1}{(1-x)+\frac{x}{20}} = \frac{1}{\frac{20-20x+x}{20}} = \frac{20}{20-19x}$$

Max speedup (capita da me) :

$$\frac{1}{1-0,52} = 2,08$$

half speedup = 1, 54

$$1,54 = \frac{20}{20-19x}$$

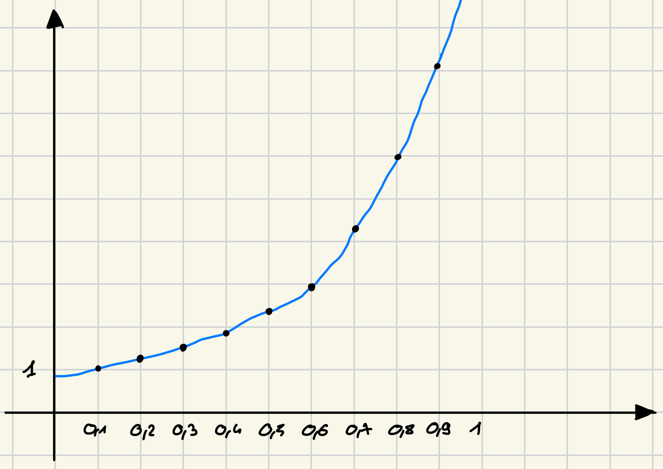
$$0,077 = \frac{1}{20-19x}$$

$$1,54 - 1,463x = 1$$

$$1,463x = 0,54$$

$$0,37 = x$$

Graph:



Max speedup Piccoli :

$$\text{Speedup} = \frac{20}{20-19x} = \frac{20}{1} = 20$$

$$\frac{1}{0{,}3+\frac{0{,}7}{20}} = 2{,}98$$

$$\frac{1}{0{,}3+\frac{0{,}7}{40}} = 3{,}149$$

$$\frac{1}{(1-x)+\frac{x}{20}} = 3{,}149$$

$$\frac{20}{20-19x} = 3{,}149$$

$$20 = (20-19x)\cdot 3{,}149$$

$$20 = 62{,}98 - 59{,}831\,x$$

$$42{,}98 = 59{,}831\,x$$

$$x = 0{,}7183$$

71,83 %

$$\text{Speedup} = \frac{1}{0{,}1 + \frac{0{,}9}{5}} = \frac{1}{0{,}28} = 3{,}57$$

$$(4 * 0{,}25) + (0{,}75 * 1{,}33)$$

$$1 + 0{,}9975 = 1{,}997$$

$$2 - 0{,}02 * (20 - 2) = 1{,}64$$

$$0{,}75 * 1{,}33 + 0{,}25 * 2{,}5 =$$

$$0{,}9975 + 0{,}625 = 1{,}625$$

$$\text{Speedup} = \frac{2{,}00}{1{,}625} = 1{,}23$$

$CPI_{time} = (0{,}75 * 1{,}33) + (0{,}25 * 4) = 2$

$CPI_{FPSQR} = 2 - 0{,}02\,(20{,}00 - 2) = 1{,}64$

$CPI_{FP} = 2 - 0{,}25\,(4 - 2{,}5) = 1{,}625$ $\leftarrow$ better

Speedup $\frac{2}{1{,}625} = 1{,}23$

$CPI = 0{,}43 * 1 + 0{,}21 * 2 + 0{,}12 * 2 + 0{,}24 * 2$

$= 0{,}43 + 0{,}42 + 0{,}24 + 0{,}48$

$= 1{,}57$

$$\frac{(0{,}43 - 0{,}43 * 0{,}25)*1 + (0{,}21 - (0{,}25*0{,}43))*2 + (0{,}43*0{,}25)*2 + 0{,}12*2 + 0{,}24*3}{(1 - 0{,}25*0{,}43)} =$$

$$\frac{0{,}3225 + 0{,}205 + 0{,}215 + 0{,}24 + 0{,}72}{0{,}8925} = 1{,}907$$

$0{,}30 * 1 + 1 * 0{,}70 = 1$

$0{,}1 * 1{,}05 + 1 * 0{,}70 = 0{,}805$

$CI_{OTT} = CI_{NON\,OTT}\,(1 - 0{,}3) + 0{,}3 * (1 - \frac{1}{3}) = 0{,}9 * CI_{NON\,OTT}$

```
__shared__ int smem [blockDim][blockDim]

int Raw = blockId.y * BlockDim.y + threadId.y
int Col = blockId.x * BlockDim.x + threadId.x

if (Raw < N && Col < N) {
    smem [threadId.y] [threadId.x] = input [Raw * N + Col]

    __synchthreads()

    Raw = BlockId.x * BlockDim.x + threadId.y
    Col = BlockId.y * BlockDim.y + threadId.x

    output [Raw * n + Col] = smem [threadId.x] [threadId.y]
}


__shared__ int mat A [TILE] [TILE]

__shared__ int mat B [TILE] [TILE]

int tx = threadId.x ,   int ty = threadId.y ;

int bx = threadId.x ,   int ty = threadId.y ;

int Raw = by * blockDim.y + ty

int Col = bx * blockDim.x + tx


for ( int K = 0 ;  K < Width / TILE ;  K++ ) {
    smem A [ ty ] [ tx ] = input [ Raw * N + Tile * k + Tx ]
    smem B [ ty ] [ tx ] = input [ Col + ( Tile * K + ty ) * N ]
    __synchthreads()

    int Pvalue = 0
    for (int i = 0 ; i < Tile ; i++ ) {
        Pvalue += matrix A [ty] [i] * matrix B [i] [tx]
    }
    __synchthreads()

    output [Raw * N + Col] = Pvalue ;
}
```

```
__ shared __  int smem[1024]
int global_id = blockId.x * blockDim.x + threadId.x
smem [threadId.x] = input [global_id]
__ synchthreads ()
for (int i=1; i< blockDim.x; i*=2 ) {
    if ( threadId.x % (i*2) == 0) {
        smem [threadId.x] += smem[threadId.x + i]
    }
    __ synch threads ()
}
if (threadId.x == 0) {
    output [blockId.x] = Smem [0]
}
```

```
__shared__ int smem[1024]
int global_id = blockId.x * blockDim.x + threadId.x
smem[threadId.x] = input[global_id]
__syncthreads()
for (int i=1 ; i < blockDim.x ; i *= 2) {
    if ( threadId.x % (i*2) == 0) {
        smem[threadId.x] += smem[threadId.x + i];
    }
    __syncthreads()
}
if (threadId.x == 0) {
    out[blockId.x] = smem[0]
}
```

```
__shared__   int smem[1024];
int global_id = blockId.x * blockDim.x + threadId.x
smem[threadId.x] = input[global_id]
__syncthreads()
for (int i = 1; i < blockDim.x; i*=2) {
    if (threadid.x % (i*2) == 0) {
        smem[threadId.x] += smem[threadId.x + i]
    __syncthreads();
    }
}
if (threadId.x == 0)
    output[BlockId.x] = smem[0]
```