Brian Crafton

1)

Friday, January 15, 2016 4:41 PM

$T(n) = n \ell T$ (no pipe)

$T(n) = (s + \ell + n - 1) T$ (pipe)

$r(n) = \left(\frac{T(n)}{n}\right)^{-1}$

$T(n) = (n + n_{1/2}) T$

$n_{1/2} = s + \ell - 1$

Pipeline:

$$\boxed{\begin{aligned} r(n) &= \left(\frac{(n + n_{1/2})T}{n}\right)^{-1} \\ r_\infty &= \left(\frac{T}{1}\right)^{-1} \end{aligned}}$$

none pipeline:

$$\boxed{\begin{aligned} r(n) &= (\ell T)^{-1} \\ r_\infty &= (\ell T)^{-1} \end{aligned}}$$

asymptotic speed-up $= \ell$.

$r(n_{1/2}) = \left(\frac{2n T}{n}\right)^{-1}$

$r(n_{1/2}) = (2T)^{-1}$

$r(n_{1/2}) = \frac{r_\infty}{2}$.

---

3) $X[i+2] = a[i+1]\left(a[i] \cdot X[i] + b[i]\right) + b[i+1]$

$T_s(n) = (s + \ell + n - 1) T = (2n + \ell - 1) T$

$T_0(n) = (\ell + n - 1) T$

$s = n$ because # of operations is doubled

initial calculations may be less important if: pipeline forwarding, a and b contain many 0s

---

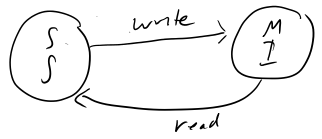
4) the problem would be avoided because the cache would only be flushed once per iteration. If cache address was msb(16) then a[0][i], a[1][i], a[2][i] would all be in same cache line since their msb(16) are x, x+1, x+2 not x, x+8k, x+16k.

using the first 16 bits is generally bad because it eliminates all benefits of spatial locality and because it will take longer to load non-adjacent memory addresses to cache

---

2) $\underline{\text{Speedup} = P}$

$$n^{1/2} = \frac{P}{2\gamma}$$

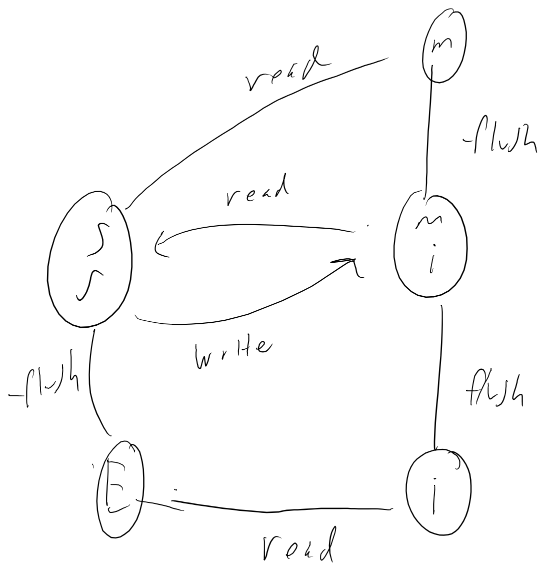


- if X_1 and X_2 are never flushed then there is no need to use memory bandwidth

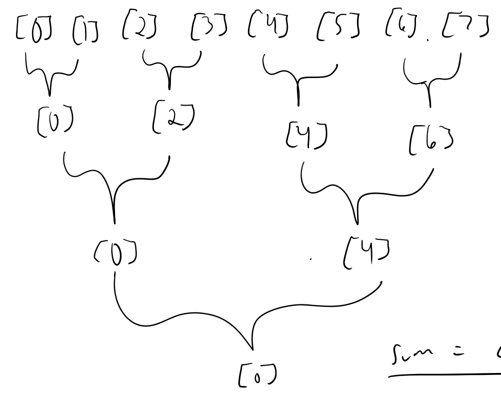
but if flushes happen :

MESI

	M	E	S	I
M	X	X	X	✓
E	X	X	X	✓
S	X	X	✓	✓
I	✓	✓	✓	✓



6



$\underline{\text{Sum} = [0].}$

the algorithm makes good use of both spatial and temporal locality.

- In the early iterations it will use most to all data in each cache line it fetches because the stride is small at first. As the stride increases we access logarithmically less memory so we care less about the spatial locality
- In the later iterations of the algorithm we will be accessing the same indexes in memory we have already accessed meaning we make good use of temporal locality.

— The special algorithm and the standard algorithm have similar spatial locality because in the standard algorithm and first iteration of the special algorithm they access each index sequentially meaning each index of each cache line will be used

— The temporal locality differs. The standard algorithm uses the same 'sum' variable so it will make use of temporal locality. the special algorithm will use the same index <u>many</u> times... this is because it stores temporary sums in (power of 2 strided) indexes of the array itself.

---

7) Reuse: • Indexes in array C will be reused because it will fit in cache and not be flushed since it is used in each operation

• Array A and B are not reused because they are too large for cache and the loop is not written to reuse the values it has cached. Each index in the cache line will be used however... spatial locality

Cache Size: Because it is smaller than A and B, it eliminates reuse we could have obtained

Associativity: The associativity of the cache is important because if A, B, or C are mapped to the same cache lines in a <u>set associative</u> or <u>direct mapped cache</u> then it is possible that on <u>each operation</u> we would be flushing a cache line that would otherwise be used <u>many</u> more times.

It would perform <u>worse</u>.

- C would still be reused and have whole cache line used
- Each index in B would be reused for each element in C and the whole cache line <u>each time</u> would be used
- But: A would have not be reused <u>and</u> would only use <u>1</u> index of every cache fetch. Each index in A would have to be fetched from RAM.