

The square-root law: a rule of thumb for data dependencies

Application to understanding the
impact of instruction fetching, and
miscellaneous other things

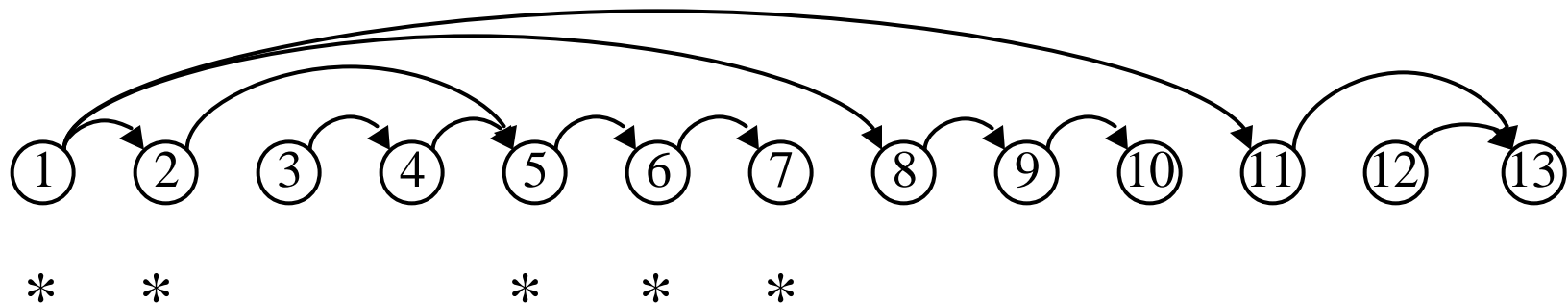
Pierre Michaud, January 2001

Dynamic data-flow graph

- Consider a program instruction trace
- Take a slice of S consecutive instructions in this trace
- Consider the data-flow graph of the slice
 - take into account true data dependencies only
- Determine the length L of the longest chain
 - label each instruction with its latency
- L is the minimum execution time of the slice
 - assuming no value prediction

Example

instruction latency = 1 cycle



$$S = 13$$

Longest chain length = 5

The square-root law

- What is, on average, the length L_S of the longest chain in a slice of S consecutive instructions ?

$$L_S \approx \frac{1}{a} \sqrt{S}$$

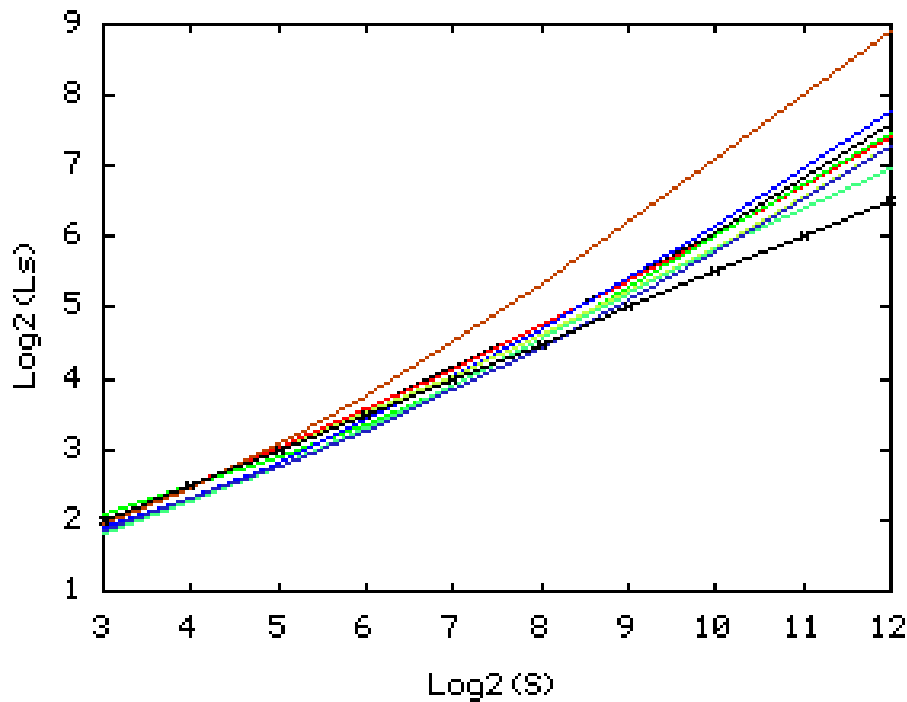
parameter a quantifies the “density” of parallelism

$a \approx 0.7$ for 1-cycle latencies

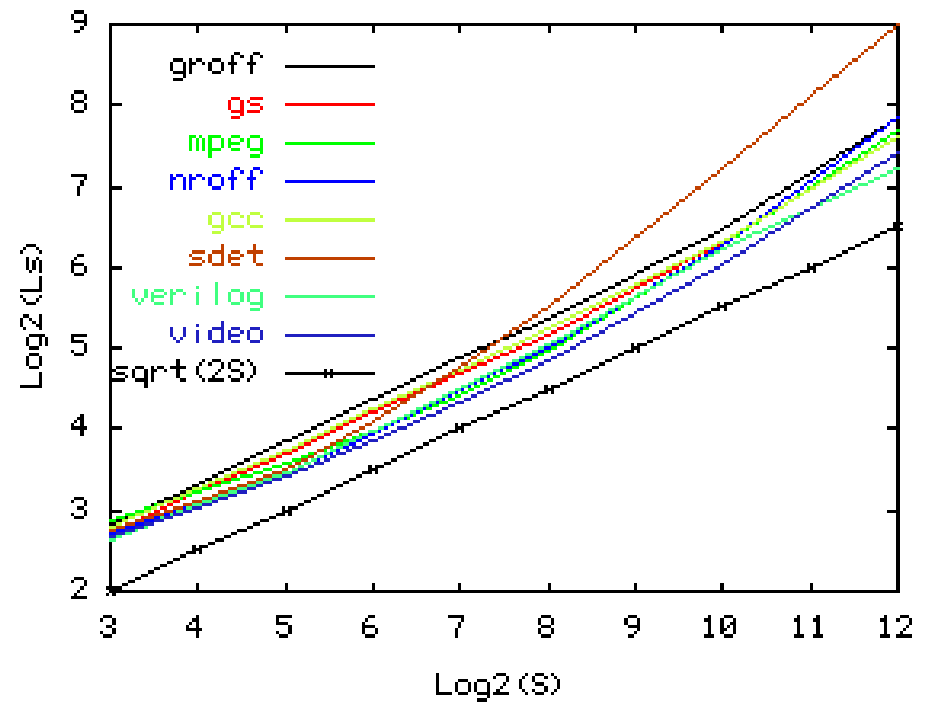
IBS benchmarks

slice size S from 8 to 4k instructions

load latency = 1 cycle



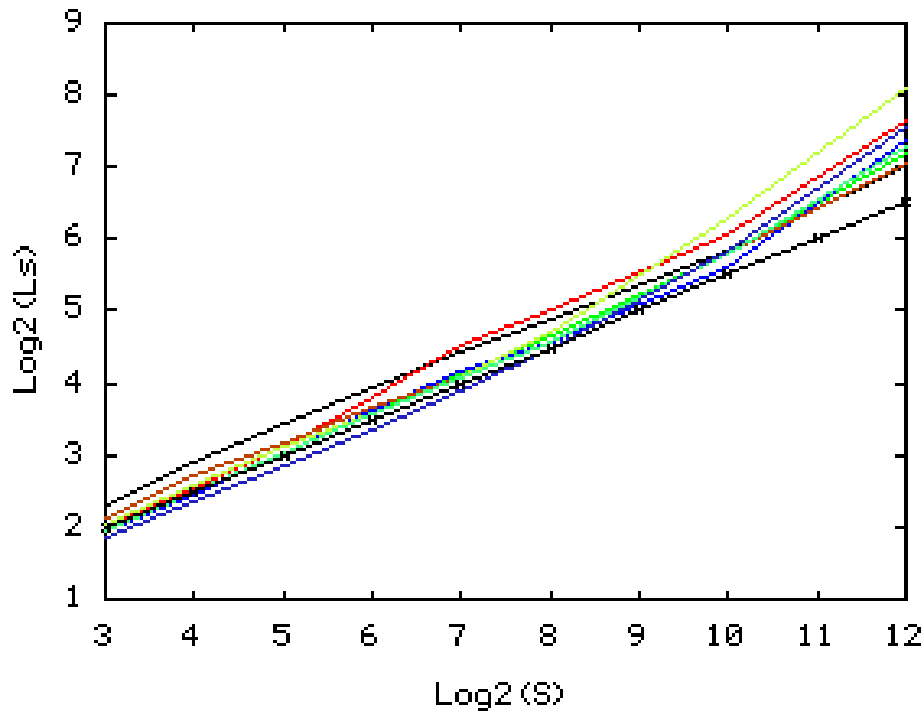
load latency = 4 cycles



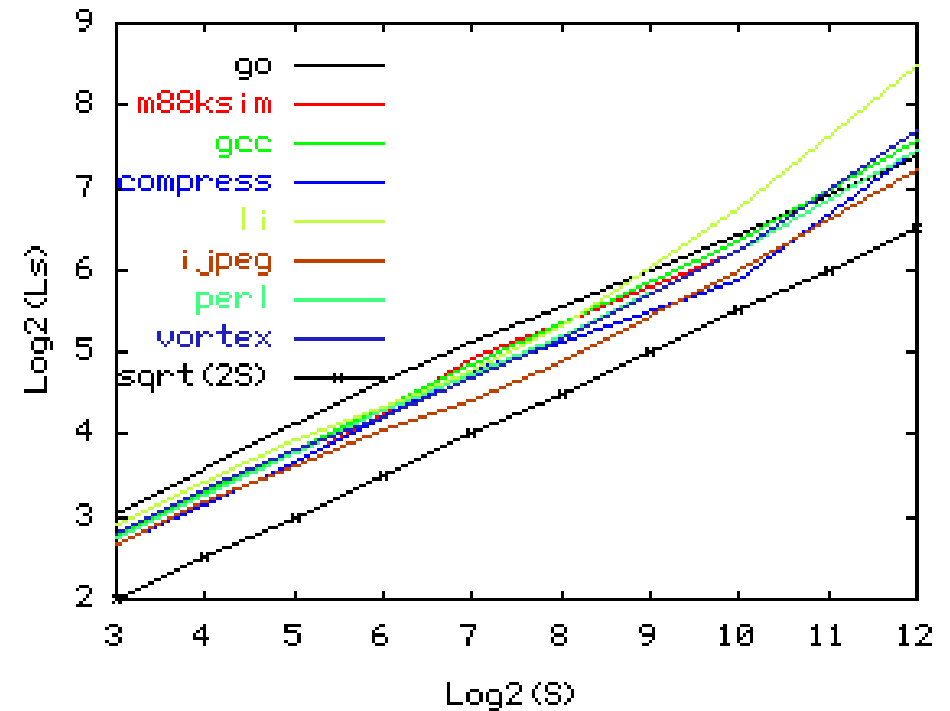
SPECint 95

slice size S from 8 to 4k instructions

load latency = 1 cycle



load latency = 4 cycles



Another way to observe the sqrt law

- Processor with
 - reorder buffer of W entries
 - perfect branch prediction
 - unlimited fetch
 - no resource constraints
- IPC proportional to \sqrt{W}
 - NB: average fetch bandwidth needed equals the IPC

$$a\sqrt{W} \leq IPC \leq 2a\sqrt{W}$$

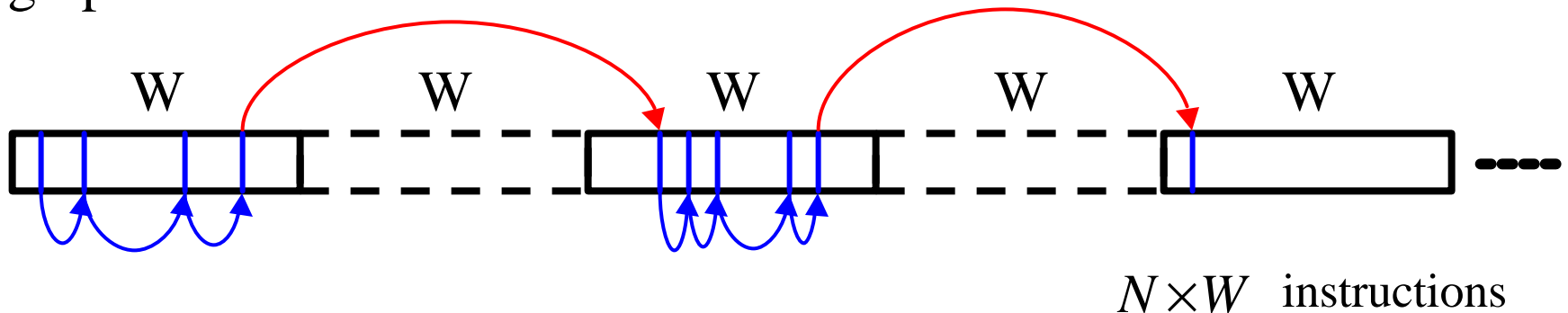
demonstration in following slides...

$$IPC \approx \sqrt{W}$$

for 1-cycle latencies

Demonstration (1)

- Consider the whole instruction trace and its data-flow graph
- Let's add a “window-dependence” edge between every pair of instructions (I,J) such that $(J - I) \geq W$
- The total execution time T is the length of the longest chain in this graph



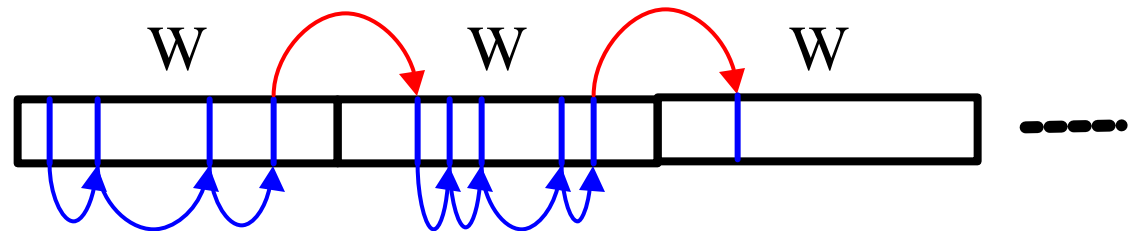
$$T \geq \frac{N}{2} L_w$$

$$IPC = \frac{N \times W}{T} \leq \frac{2W}{L_w}$$

Demonstration (2)

- Consider the whole instruction trace and its data-flow graph
- Let's partition the trace into N slices of W instructions each
- Let's add an edge between every pair of instructions (I, J) such that I and J belong to different slices
- Let's L be the length of the longest chain in this graph

NB: the graph of the previous slide is a subgraph of this one



$$T \leq L = N \times L_w \qquad IPC = \frac{N \times W}{T} \geq \frac{W}{L_w}$$

A good rule of thumb

- Already observed by Riseman and Foster in early 70's
- Works well for slice sizes $< 1k$ instructions
- Models only average behavior
 - several programs or program with complex control flow
 - do not use it to predict the performance of a tight loop
- Why the square root ? I don't really know but ...
 - it is normal to find a sub-linear law
 - many programs have similar instruction-type distributions
 - for $S < 1k$, it is an approximation, for $S > 1k$ it is no longer valid

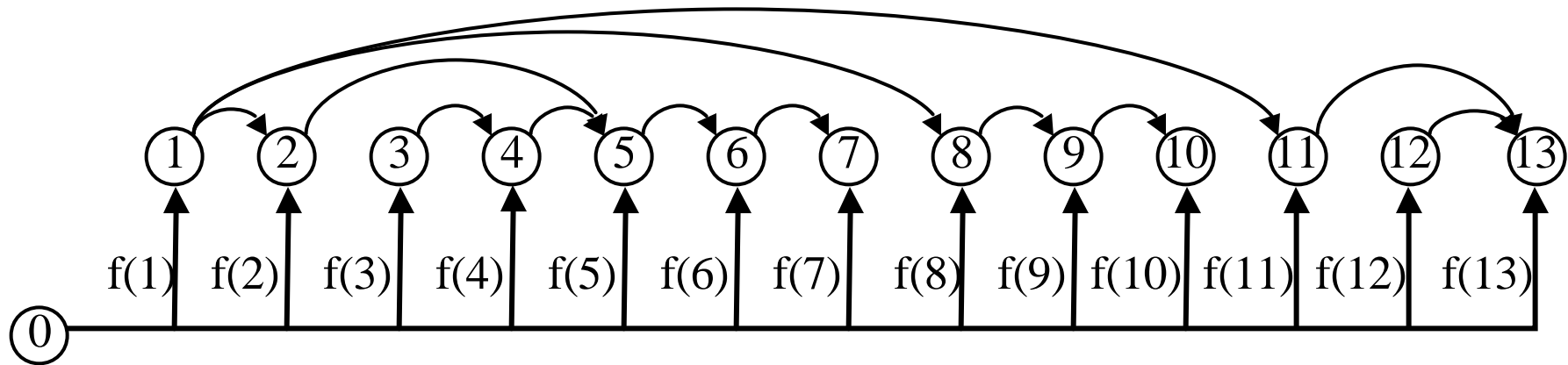
Processor model

- Unlimited execution resources
- Unlimited reorder buffer
- 1-stage pipeline
- Limited fetch bandwidth
- Control-flow breaks (mainly branch mispredictions)
 - processor window flushed after a control-flow break
 - slice = instructions between consecutive control-flow breaks
 - the processor executes slices one after another

Observations from experiments

- Once the instruction fetch bandwidth is twice the IPC, extra fetch bandwidth is superfluous
- To double the IPC, we should both double the instruction fetch bandwidth and decrease the number of control-flow breaks fourfold
 - the fetch bandwidth requirement grows as the square root of the distance between branch mispredictions

Fetch cycle constraints



$f(i)$ = cycle in which instruction i is fetched

execution time $T = \max_{i=1 \dots S} (f(i) + l(i))$ $l(i) \in [1 \dots L_S]$

$l(i)$ = length of the longest chain emanating from instruction i

Milestone instructions

Instructions fetched in program order : fetch function f is increasing

For a given $l(i)=n$, we only need to consider the instruction fetched the latest

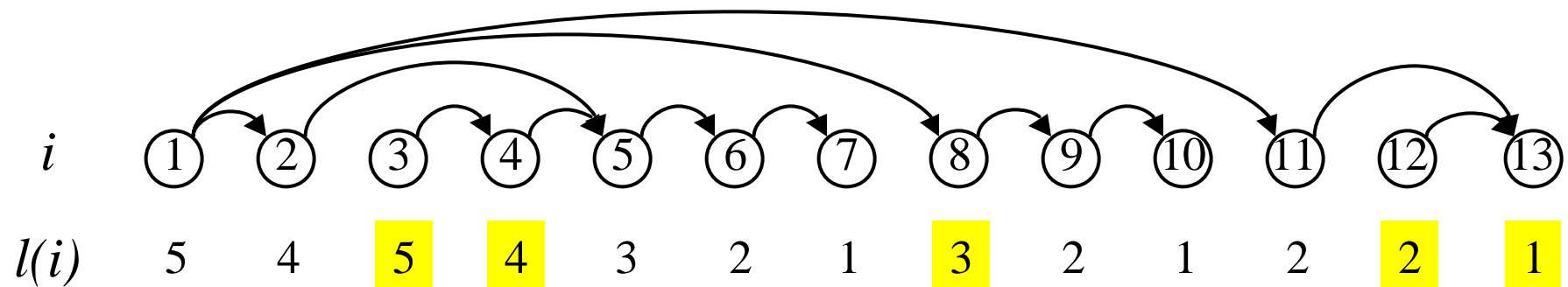
milestone instruction of rank n

$$m_n = \max(i \in [1 \cdots S] / l(i) = n)$$

$$T = \max_{n=1 \cdots L_S} (f(m_n) + n)$$

Optimal fetch: one milestone per cycle

Example



Optimal fetch:

- cycle 0 : fetch $m_5 = 3$
- cycle 1 : fetch $m_4 = 4$
- cycle 2 : fetch $m_3 = 8$
- cycle 3 : fetch $m_2 = 12$
- cycle 4 : fetch $m_1 = 13$

Milestone model with sqrt law

- Let's define a model of the form $m_n = S - g(n)$
- Coherence with $L_S \approx \frac{1}{a} \sqrt{S} \quad \Rightarrow \quad m_n = S - \mathbf{a}^2 n^2$
- The optimal fetch is not uniform
 - the fetch bandwidth requirement (FBR) decreases gradually as we get closer to the end of the slice

$$\boxed{FBR \approx 2\mathbf{a} \sqrt{s}}$$

s = distance from the end of the slice

When half the slice is fetched, the FBR is 30 % lower than at the start

Uniform fetch

F instructions per cycle $f(i) = \frac{i}{F}$

$$T = \max_{n=1 \dots L_S} \left(n + \frac{S - \mathbf{a}^2 n^2}{F} \right)$$

execute what
remains in
the window

fetch the
slice

threshold fetch rate

$$\boxed{F_t = 2\mathbf{a}\sqrt{S}}$$

$$\left\{ \begin{array}{l} F \leq F_t \Rightarrow \\ F > F_t \Rightarrow \end{array} \right.$$

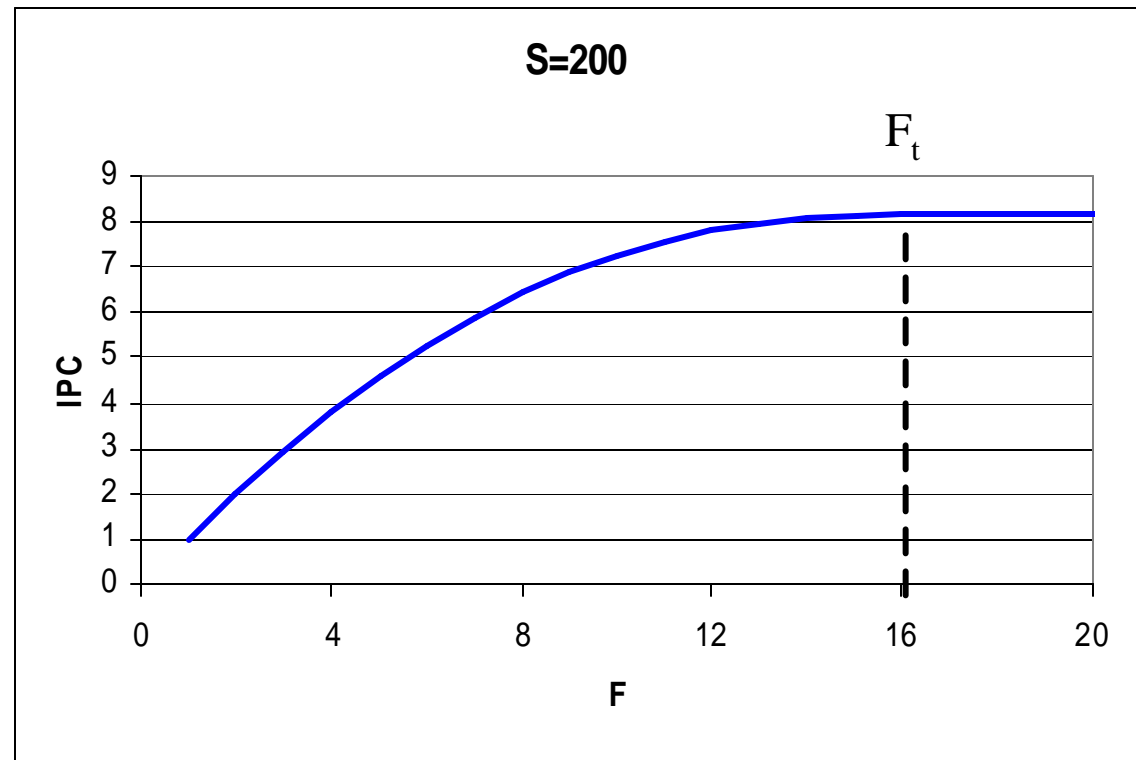
$$\boxed{T = \frac{S}{F} + \frac{F}{4\mathbf{a}^2}}$$

$$T = L_S = \frac{1}{\mathbf{a}}\sqrt{S}$$

$$F_t = 2 \text{IPC}_{\max}$$

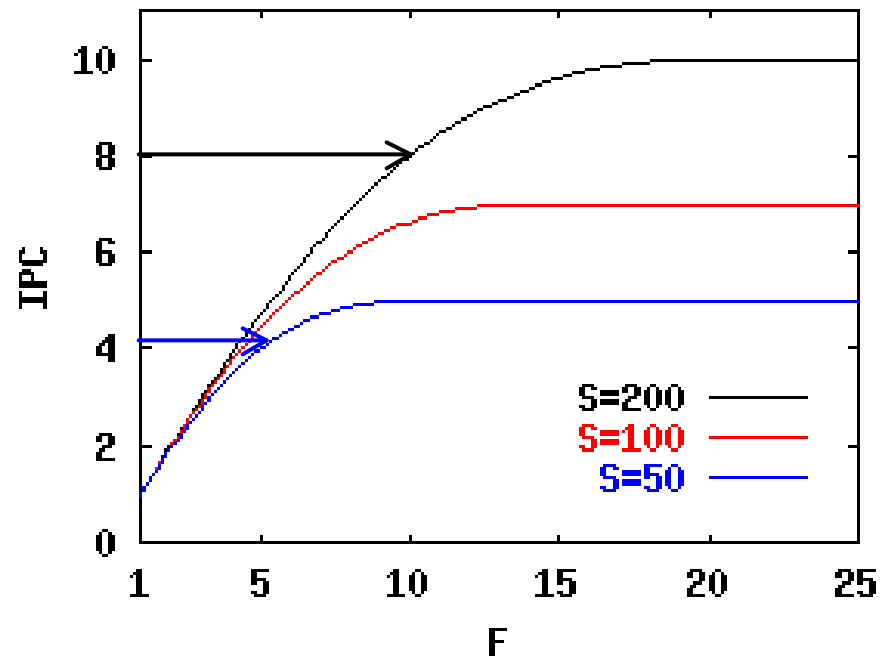
$$\text{IPC} = \frac{F}{1 + \left(\frac{F}{F_t}\right)^2}$$

$$\text{IPC}_{\max} = \frac{F_t}{2}$$



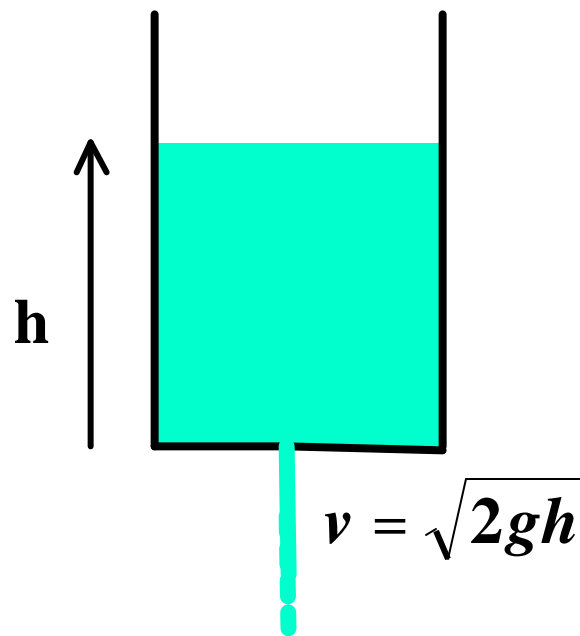
Fetch bandwidth / mispredict ratio

Curves are homothetic



To double the IPC, we should both double the fetch rate and decrease the mispredict ratio four-fold

Variant: “leaky bucket” model



N_t = number of instructions waiting for execution in the window in cycle t

$$ILP_t = \sqrt{2gN_t} \quad g = \text{“gravity” constant}$$

$$\left\{ \begin{array}{l} t < \frac{S}{F} \quad \Rightarrow \quad N_{t+1} = N_t + F - \sqrt{2gN_t} \\ t > \frac{S}{F} \quad \Rightarrow \quad N_{t+1} = N_t - \sqrt{2gN_t} \end{array} \right.$$

ILP: rising, constant, falling

$$\sqrt{2gN_{t+1}} - \sqrt{2gN_t} \approx -g$$

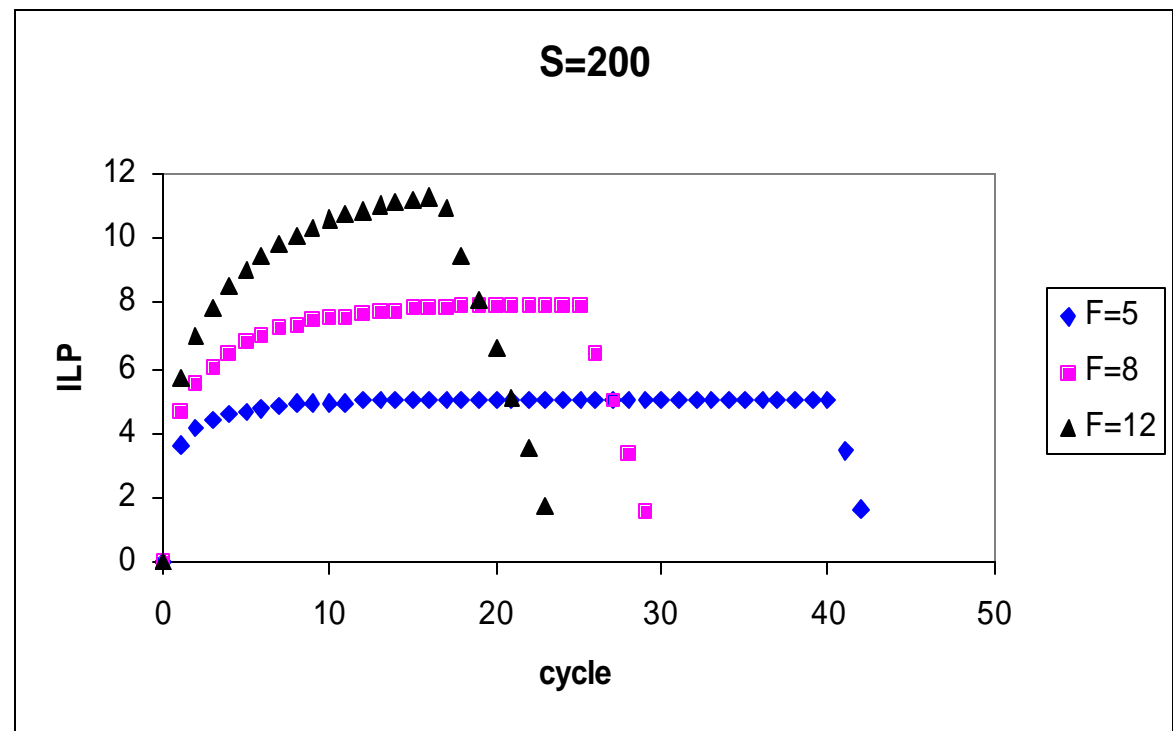
NB: instructions on correct path only

$$T \approx \frac{S}{F} + \frac{F}{g}$$



$$g = 4a^2$$

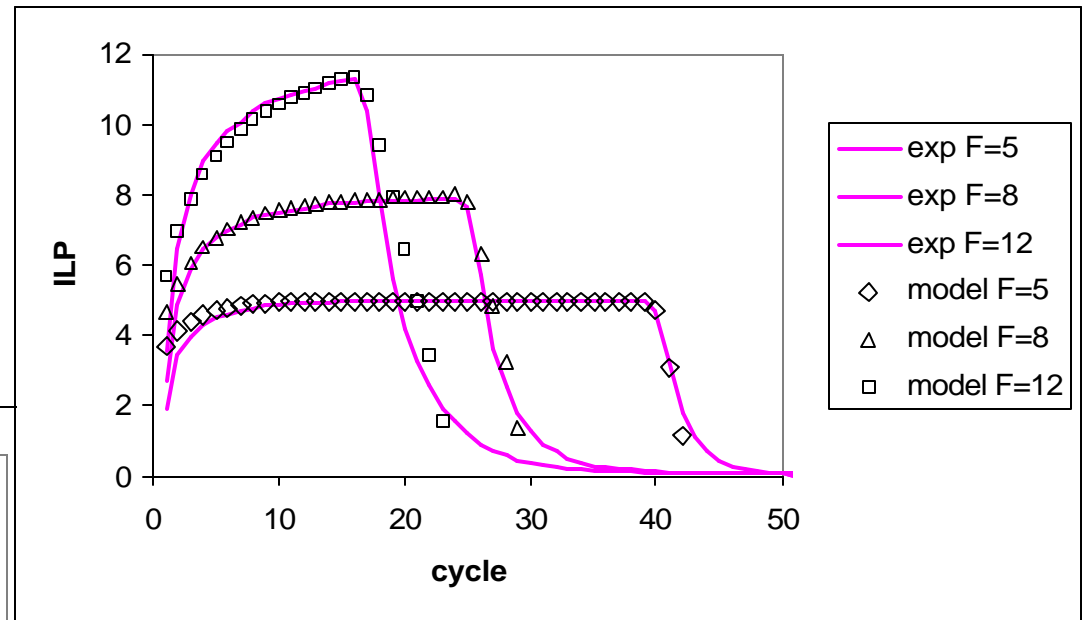
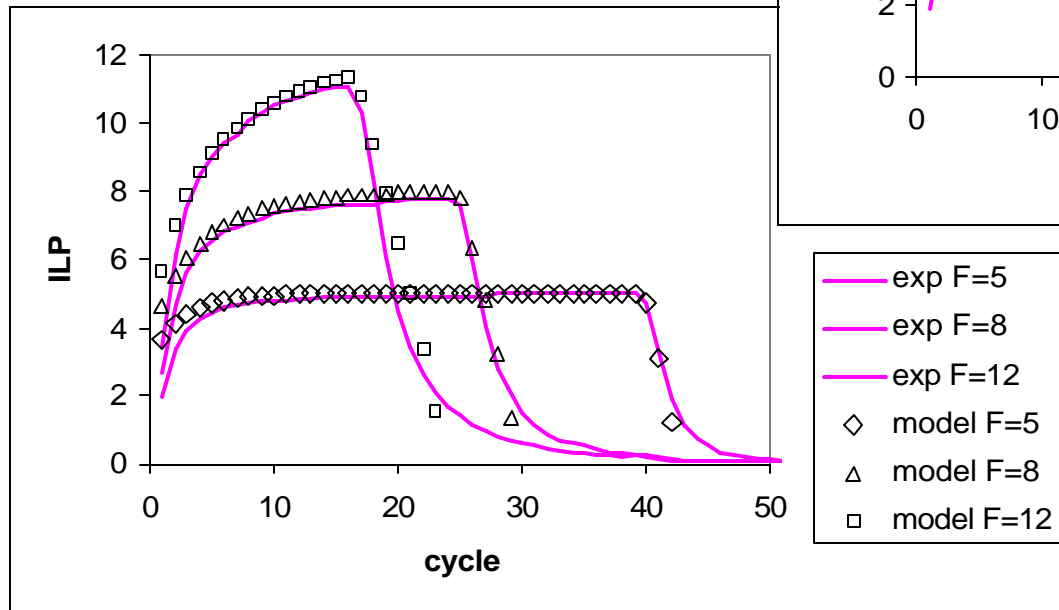
$$\approx 2$$



Experimental corroboration

Average on slices of
S=200 instructions

IBS gs



IBS gcc

Scaling the pipeline length

- We wish to multiply the IPC by k
- We multiply the pipeline width by k and we divide the mispredict ratio by k^2
- What pipeline lengthening can we tolerate ? answer : k

$$CPI = \frac{P}{S} + \left(\frac{1}{F} + \frac{F}{gS} \right) = CPI_{depth} + CPI_{width}$$

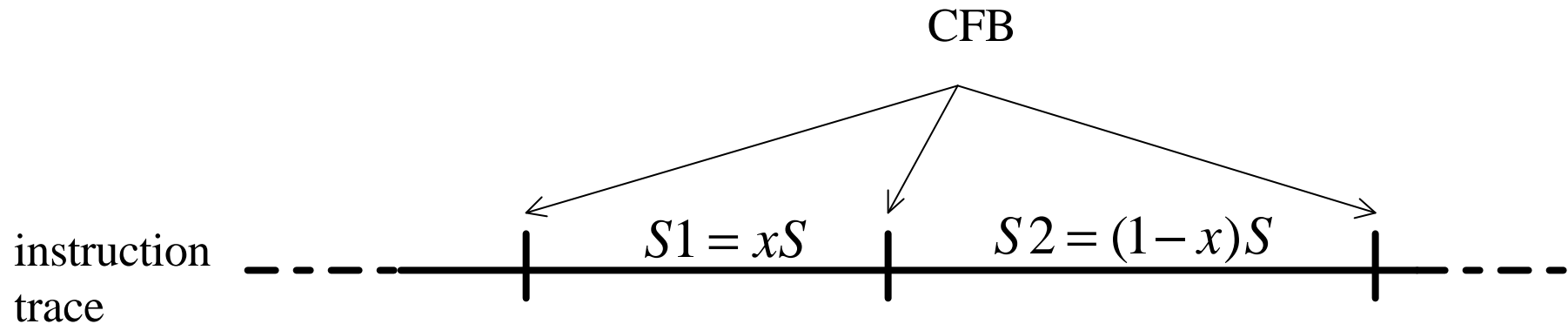
P = pipeline penalty on a control-flow break

Keep $CPI_{depth} = \frac{P}{S}$ proportional to $\frac{1}{\sqrt{S}}$

we tolerate

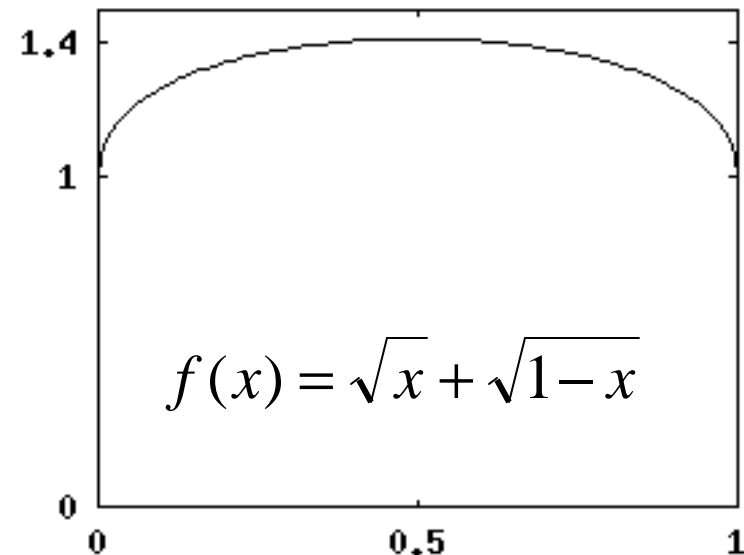
$$P \propto \sqrt{S}$$

A closer look at control-flow breaks ...



$$T_{\min} = L1 + L2 = \sqrt{xS} + \sqrt{(1-x)S}$$

For ILP, clustered CFBs are better than evenly-spaced ones



Impact of CFB distribution

Each CFB defines a slice

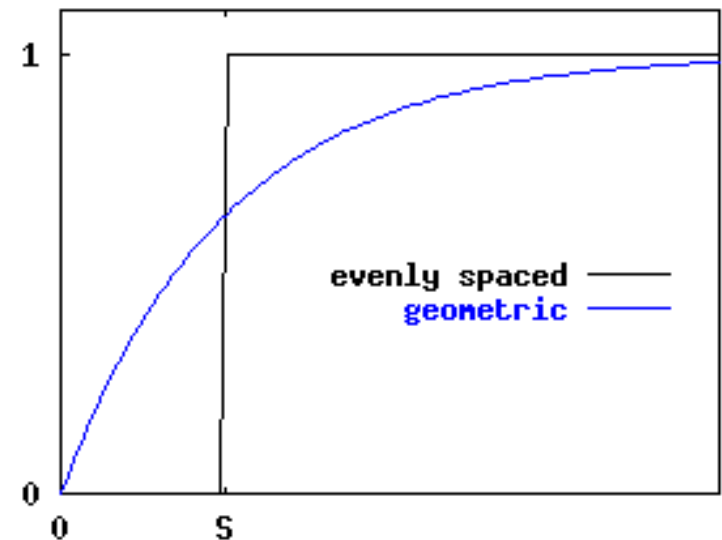
Distribution $F(s)$ = probability that slice size $\leq s$

$$T_{slice} = \int_0^{\infty} \sqrt{s} \times F'(s) ds = \int_0^{\infty} \frac{1}{2\sqrt{s}} (1 - F(s)) ds$$

$$T_{evenly_spaced} = \sqrt{s}$$

$$T_{geometric} \approx \frac{\sqrt{p}}{2} \sqrt{s} \quad 2 / \sqrt{p} \approx 1.13$$

Experiment with real branch predictor:
 ILP ~10 % higher than constant S model



Simultaneous multithreading

- SMT processor with fixed instruction window W
- We wish to double the global IPC
- How many threads ?

N independent threads $IPC \approx N \sqrt{\frac{W}{N}} \approx \sqrt{WN}$

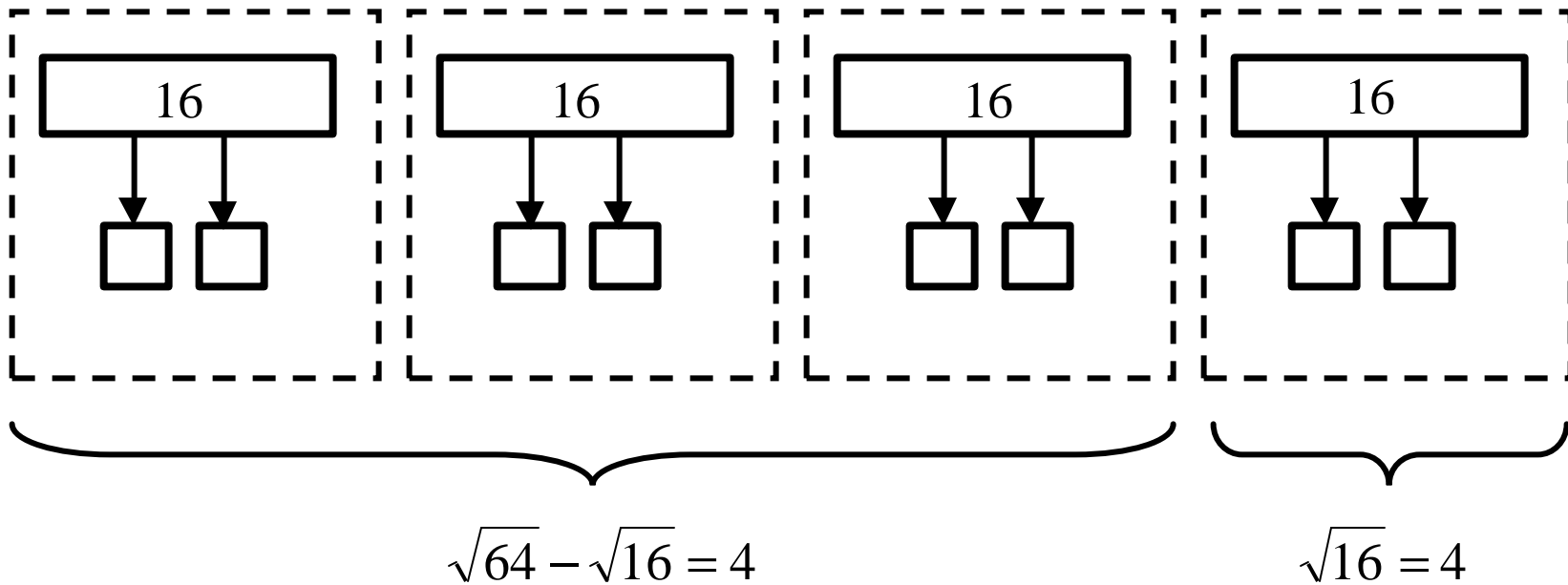
To double the global IPC, the number of threads should be multiplied by 4

Alternative: double both the number of threads and the window

Window partitioning

recent
instructions

oldest
instructions



$$IPC_{unified} = 8$$

$$IPC_{partitioned} = 4 + 2 = 6$$



30 % CPI increase from partitioning

Conclusion

- The impact of data dependencies is not always intuitive
- Square-root law useful for having a rough idea of what simulation results should be expected
- Use it for predicting relative rather than absolute behavior
 - “ what speedup processor A will have on processor B ? ”