

MLP yes!

Definitions ILP no

!

**MLP**

ILP = Instruction Level

Work on memory level parallelism. Stop worrying about IPC.

= Memory Level

Parallelism a Parallelism = Number cache misses

simultaneously

IPC metric misleading (Inst. per Clock)

Andy “Krazy” Glew

outstandinga glew@cs.wisc.edu, glew@hf.intel.com

esp. for linked lists!

ASPLOS 98 Wild and Crazy Ideas Session

98/10/09 17:01 Glew: MLP yes! IPC no! (ASPLOS98) 2

98/10/09 17:01 Glew: MLP yes! IPC no! (ASPLOS98) 3

1000 cycles per cache miss Thought Experiment

...

...

Assume:

1 operation

1 operation dispatch

complete

• Memory latency (cache miss latency) = 1000x ALU compute latency

per cycle

*Why IPC is misleading*

per cycle MLP ≠ IPC

• Memory bandwidth easy to obtain

• MLP ≈ 4 cache misses outstanding

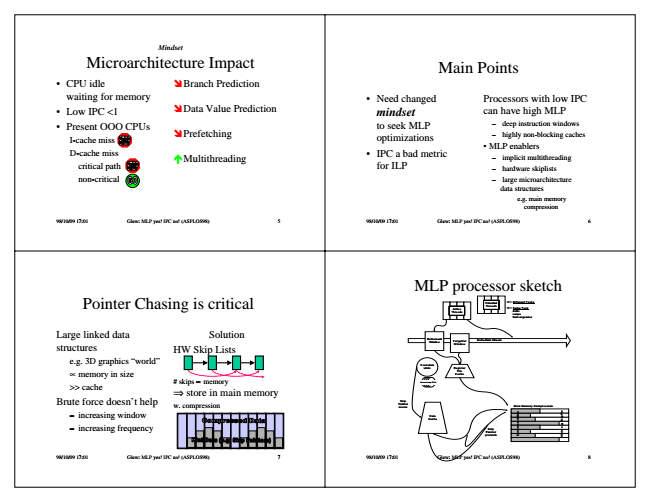
• IPC = 4 / 1004 ≈ 0.004

• Narrow machine ≈ Wide superscalar

• e.g. AXPY (trivial MLP)

98/10/09 17:01 Glew: MLP yes! IPC no! (ASPLOS98) 4

band- width ok



Mindset Microarchitecture Impact

• CPU idle waiting for memory

• Low IPC <1

• Present OOO CPUs

I-cache miss D-cache miss

critical path non-critical

Main Points ÖBranch Prediction

ÖData Value Prediction

• Need changed

Processors with low IPC mindset

can have high MLP

ÖPrefetching

ÑMultithreading

to seek MLP

– deep instruction windows optimizations

• IPC a bad metric for ILP

– highly non-blocking caches

• MLP enablers

– implicit multithreading – hardware skiplists – large microarchitecture

data structures

e.g. main memory compression

98/10/09 17:01 Glew: MLP yes! IPC no! (ASPLOS98) 5

98/10/09 17:01 Glew: MLP yes! IPC no! (ASPLOS98) 6

98/10/09 17:01 Glew: MLP yes! IPC no! (ASPLOS98) 7

MLP processor sketch Pointer Chasing is critical

Potential

**Different Tasks**

Threads Active Threads

Large linked data structures

e.g. 3D graphics “world”

Execution ∝ memory in size

Units

>> cache

Inner Loop EUs

Brute force doesn’t help

Main Memory Compression – increasing window – increasing frequency

C Same Task Calls Loops Convergences

Solution HW Skip Lists

Retirement Window

**Forgetful**

Instruction Stream Window

Register File Cache # skips ∝ memory ⇒ store in main memory w. compression

Skip Pointer create

Data

**a t a D**

**s**

Cache

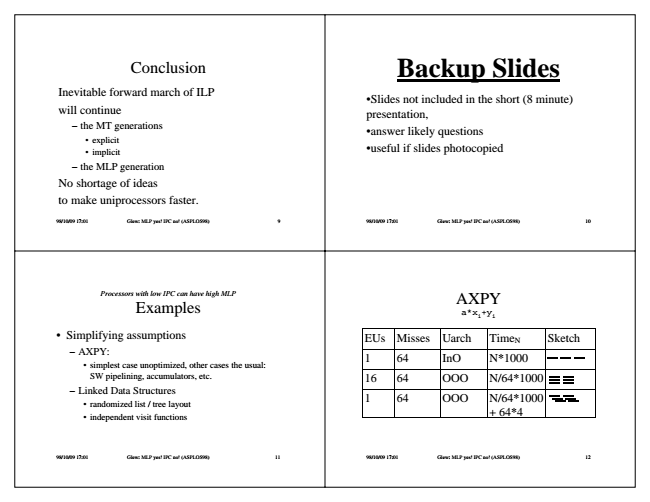
**Skip Pointer prefetch**

98/10/09 17:01 Glew: MLP yes! IPC no! (ASPLOS98) 8

**t P**

**d e s s e r p m o**

**p i k S . g . e , s t n i H**

****

Backup Conclusion

**Slides**

Inevitable forward march of ILP will continue

•Slides not included in the short (8 minute) presentation, – the MT generations

• explicit

• implicit

•answer likely questions

•useful if slides photocopied

– the MLP generation No shortage of ideas to make uniprocessors faster.

98/10/09 17:01 Glew: MLP yes! IPC no! (ASPLOS98) 9

98/10/09 17:01 Glew: MLP yes! IPC no! (ASPLOS98) 10

98/10/09 17:01 Glew: MLP yes! IPC no! (ASPLOS98) 11 Processors with low IPC can have high MLP Examples

AXPY

a\*x

i

+y

i

• Simplifying assumptions

EUs Misses Uarch Time

N

Sketch – AXPY:

• simplest case unoptimized, other cases the usual:

1 64 InO N\*1000

SW pipelining, accumulators, etc.

16 64 OOO N/64\*1000 – Linked Data Structures

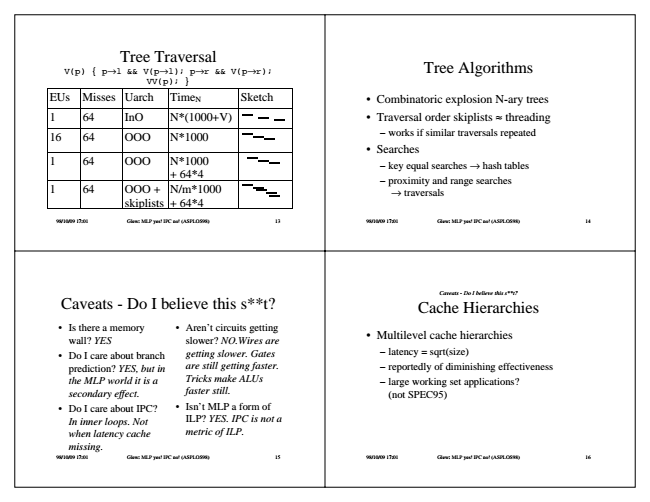
• randomized list / tree layout

• independent visit functions

1 64 OOO N/64\*1000

+ 64\*4

98/10/09 17:01 Glew: MLP yes! IPC no! (ASPLOS98) 12



Tree Traversal V(p) { p→l && V(p→l); p→r && V(p→r); VV(p); } EUs Misses Uarch Time

N

Tree Algorithms

Sketch

• Combinatoric explosion N-ary trees 1 64 InO N\*(1000+V)

• Traversal order skiplists ≈ threading

16 64 OOO N\*1000

1 64 OOO N\*1000

+ 64\*4 1 64 OOO +

skiplists

– works if similar traversals repeated

• Searches

– key equal searches → hash tables

N/m\*1000 + 64\*4

– proximity and range searches

→ traversals

98/10/09 17:01 Glew: MLP yes! IPC no! (ASPLOS98) 13

98/10/09 17:01 Glew: MLP yes! IPC no! (ASPLOS98) 14

98/10/09 17:01 Glew: MLP yes! IPC no! (ASPLOS98) 15 Caveats - Do I believe this s\*\*t?

*Caveats - Do I believe this s\*\*t? Cache Hierarchies*

• Is there a memory wall? YES

• Multilevel cache hierarchies

• Do I care about branch

– latency = sqrt(size) prediction? YES, but in

– reportedly of diminishing effectiveness the MLP world it is a

– large working set applications? secondary effect.

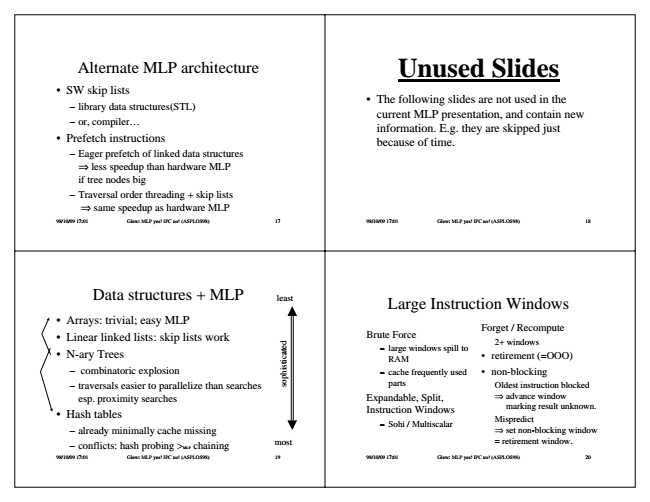
(not SPEC95)

*• Do I care about IPC? In inner loops. Not when latency cache missing.*

*• Aren’t circuits getting slower? NO.Wires are getting slower. Gates are still getting faster. Tricks make ALUs faster still.*

*• Isn’t MLP a form of ILP? YES. IPC is not a metric of ILP.*

98/10/09 17:01 Glew: MLP yes! IPC no! (ASPLOS98) 16



Unused Alternate MLP architecture

**Slides**

• SW skip lists

– library data structures(STL) – or, compiler...

• Prefetch instructions

• The following slides are not used in the current MLP presentation, and contain new information. E.g. they are skipped just because of time. – Eager prefetch of linked data structures

⇒ less speedup than hardware MLP if tree nodes big – Traversal order threading + skip lists

⇒ same speedup as hardware MLP

98/10/09 17:01 Glew: MLP yes! IPC no! (ASPLOS98) 17

98/10/09 17:01 Glew: MLP yes! IPC no! (ASPLOS98) 18

Data structures + MLP

least

Large Instruction Windows

• Arrays: trivial; easy MLP

• Linear linked lists: skip lists work

• N-ary Trees

Brute Force

– large windows spill to

RAM

Forget / Recompute

2+ windows

• retirement (=OOO) – combinatoric explosion

– cache frequently used

• non-blocking – traversals easier to parallelize than searches

parts

Oldest instruction blocked esp. proximity searches

• Hash tables

– already minimally cache missing

Expandable, Split, Instruction Windows

⇒ advance window

marking result unknown.

– Sohi / Multiscalar

Mispredict ⇒ set non-blocking window – conflicts: hash probing >

MLP

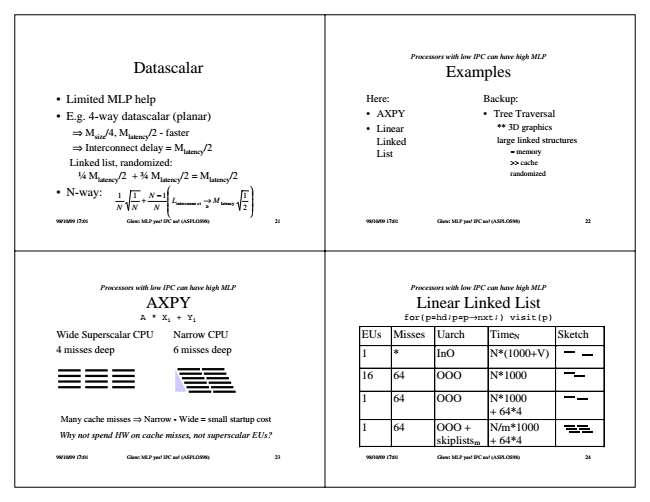
chaining

most

= retirement window.

98/10/09 17:01 Glew: MLP yes! IPC no! (ASPLOS98) 19

98/10/09 17:01 Glew: MLP yes! IPC no! (ASPLOS98) 20



*Processors Datascalar*

*with low IPC can have high MLP Examples*

• Limited MLP help

Here:

• E.g. 4-way datascalar (planar)

• AXPY

⇒ M

size

• Linear Linked List

Backup:

• Tree Traversal

/4, M

latency

/2 - faster ⇒ Interconnect delay = M

latency

/2

\*\* 3D graphics large linked structures

≈ memory Linked list, randomized:

>> cache 1⁄4 M

latency

/2 + 3⁄4 M

latency

/2 = M

latency

/2

randomized

• N-way:

*11 NN*

+

*N*

*N*

−

1     

*L*

interconne

ct →

≥

M latency 1 2

     98/10/09 17:01 Glew: MLP yes! IPC no! (ASPLOS98) 21

98/10/09 17:01 Glew: MLP yes! IPC no! (ASPLOS98) 22

*Processors with low IPC can have high MLP AXPY A \* X*

i

98/10/09 17:01 Glew: MLP yes! IPC no! (ASPLOS98) 23

*Processors with low IPC can have high MLP Linear Linked List + Y*

i

for(p=hd;p=p→nxt;) visit(p)

Wide Superscalar CPU 4 misses deep

Narrow CPU

EUs Misses Uarch Time

N

Sketch 6 misses deep

1 \* InO N\*(1000+V)

16 64 OOO N\*1000

1 64 OOO N\*1000

Many cache misses ⇒ Narrow - Wide = small startup cost

*Why not spend HW on cache misses, not superscalar EUs?*

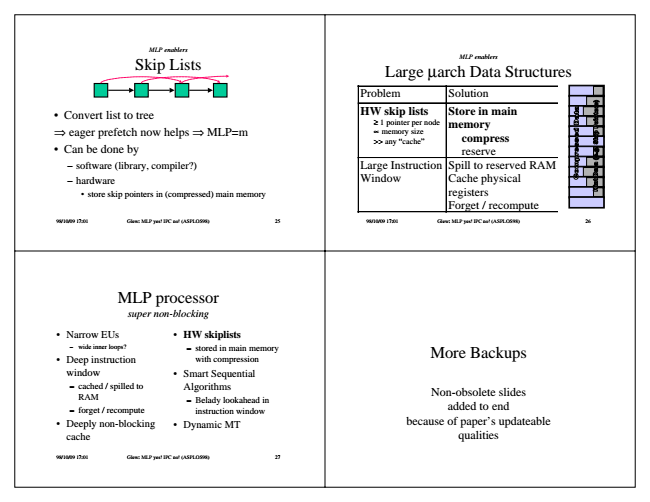
+ 64\*4 1 64 OOO +

N/m\*1000 skiplists

m

+ 64\*4

98/10/09 17:01 Glew: MLP yes! IPC no! (ASPLOS98) 24



*MLP enablers Skip Lists*

MLP enablers Large μarch Data Structures

Problem Solution

• Convert list to tree ⇒ eager prefetch now helps ⇒ MLP=m

• Can be done by

**HW skip lists**

Store in main ≥ 1 pointer per node ∝ memory size >> any “cache”

**memory**

compress reserve – software (library, compiler?)

Large Instruction

Spill to reserved RAM – hardware

Window

Cache physical

• store skip pointers in (compressed) main memory

registers Forget / recompute

98/10/09 17:01 Glew: MLP yes! IPC no! (ASPLOS98) 25

98/10/09 17:01 Glew: MLP yes! IPC no! (ASPLOS98) 26

*MLP processor super non-blocking*

• Narrow EUs

• HW skiplists – wide inner loops?

– stored in main memory

• Deep instruction

with compression window

• Smart Sequential – cached / spilled to

Algorithms RAM

– Belady lookahead in – forget / recompute

instruction window

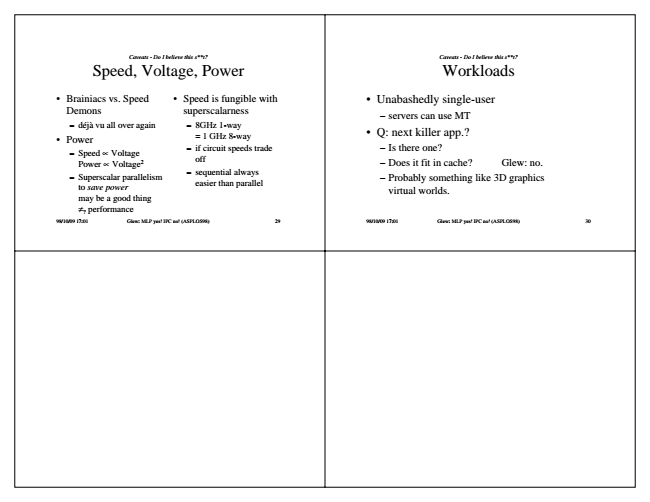
• Deeply non-blocking

• Dynamic MT cache

98/10/09 17:01 Glew: MLP yes! IPC no! (ASPLOS98) 27

More Backups

Non-obsolete slides added to end because of paper’s updateable qualities



*Caveats - Do I believe this s\*\*t? Speed, Voltage, Power*

• Brainiacs vs. Speed Demons

– déjà vu all over again

• Power

– Speed ∝ Voltage

Power ∝ Voltage2 – Superscalar parallelism

to save power may be a good thing ≠

?

*Caveats - Do I believe this s\*\*t? Workloads*

• Speed is fungible with

• Unabashedly single-user superscalarness

– servers can use MT – 8GHz 1-way

= 1 GHz 8-way

• Q: next killer app.?

– if circuit speeds trade

– Is there one? off – sequential always

easier than parallel

– Does it fit in cache? Glew: no. – Probably something like 3D graphics

virtual worlds.

performance

98/10/09 17:01 Glew: MLP yes! IPC no! (ASPLOS98) 29

98/10/09 17:01 Glew: MLP yes! IPC no! (ASPLOS98) 30