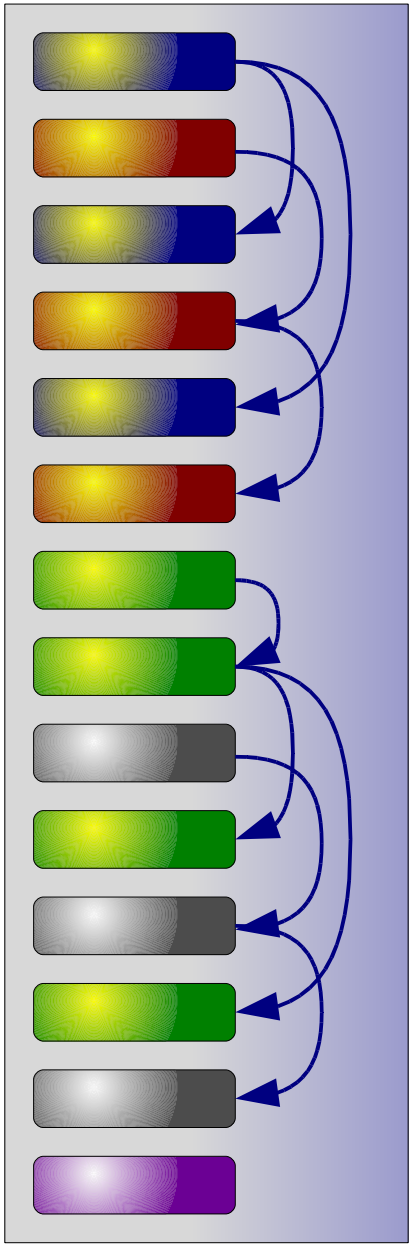
Using Fine Grain Multithreading for Energy Efficient Computing

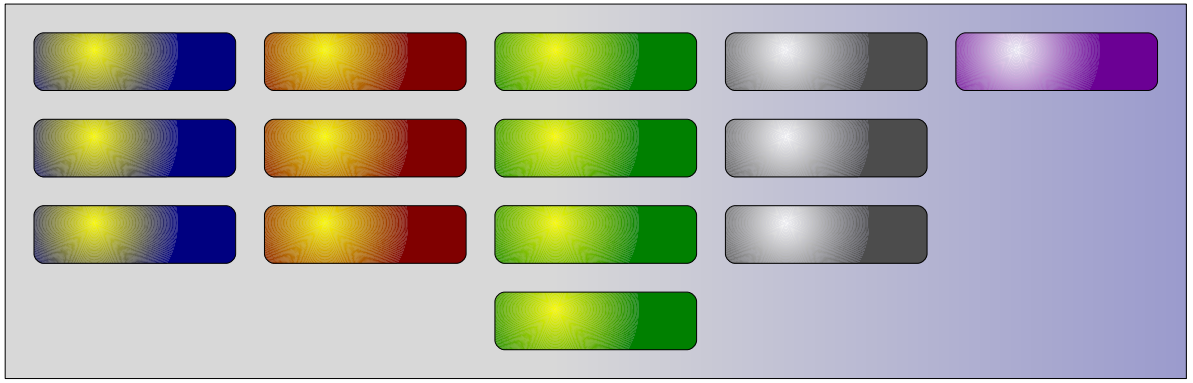
Alex Gontmakher, Technion
Avi Mendelson, Intel Labs
Assaf Schuster, Technion

Instruction Level Parallelism

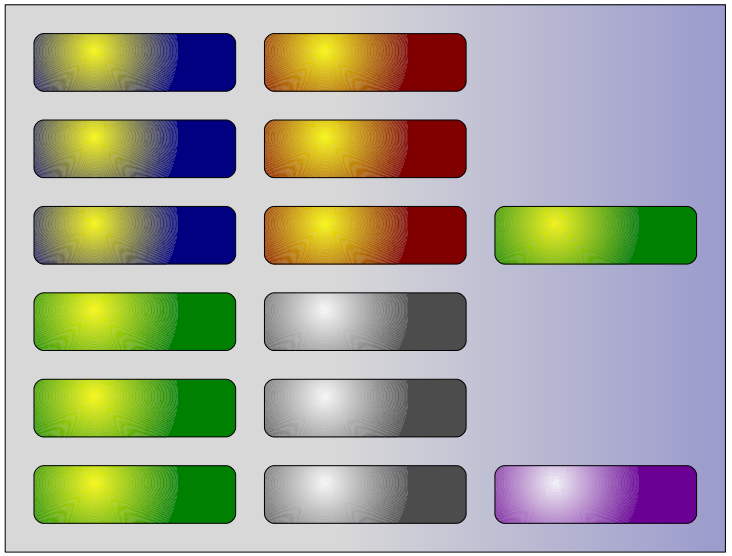
Program



Dataflow Execution

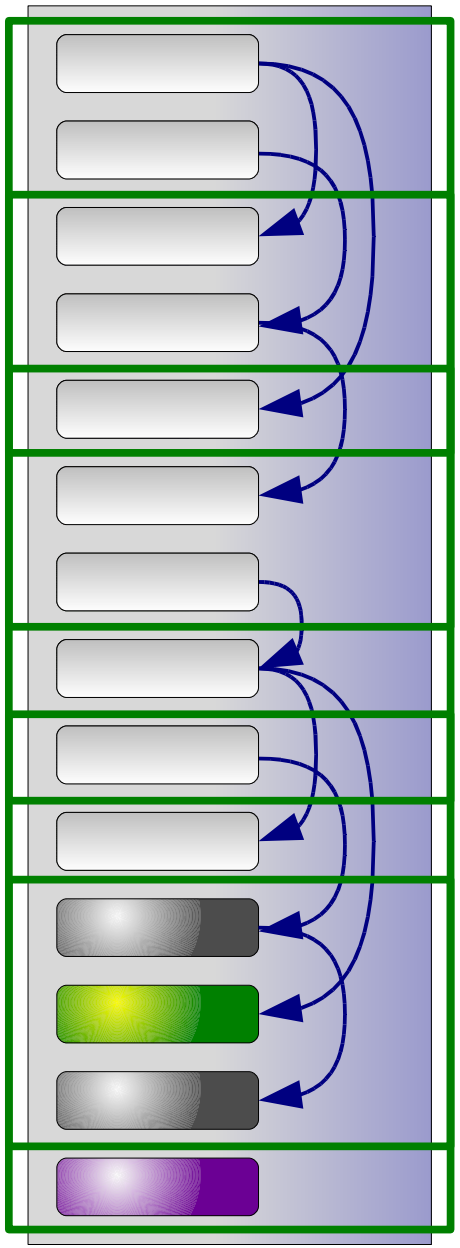


Practical Execution: constrained by processor resources



Out-of-order processing

How Out-of-Order Works



- Sliding window over the program
  - Independent instructions within the window executed
- Much of the parallelism is **beyond the horizon**!
  - Good for extremely fine granularity!
- All instructions within the window must be checked for dependencies
  - High complexity
  - Increased energy consumption

Who's The/A Culprit?

2x

3x

	MIPS		Alpha	
	R5K	R10K	21164	21264
Pipeline	InO	OOO	InO	OOO
SPECInt95	5.5	10.1	15.4	27.7
SPECFP95	5.5	9.71	21.1	58.7
Power (W)	10	30	28	91
Clock Rate	200	250	500	500
Technology (μ)	0.35	0.35	0.35	0.35
Transistors	3.6M	6.7M	9.3M	15.2M
Transistors (Logic)	0.8M	2.3M	2.6M	6M
Physical Regs		64		80

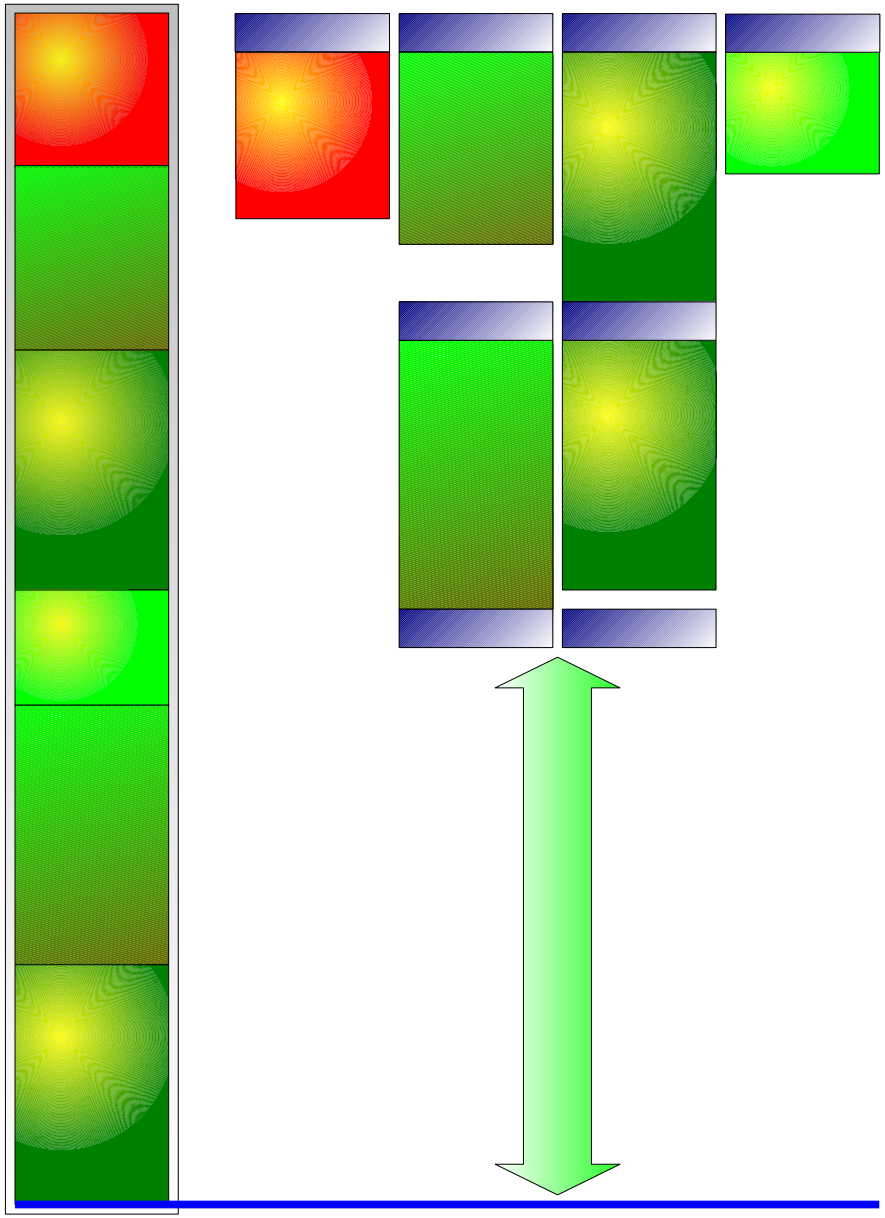
Related Work

- S. Hily, A. Seznec: Out-of-order execution may not be cost-effective on SMT processors
  - **Single thread**: inorder 46% slower than OOO
  - **4 threads**: inorder only 15% slower than OOO

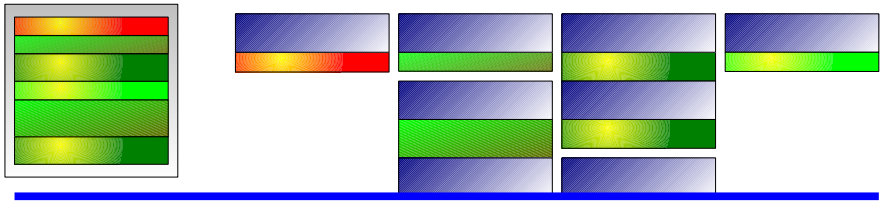
- SUN's Niagara processor:
  Massive SMT support with inorder pipelines, especially for low-power computing (throughput-oriented)

Thread Based Parallelism

Coarse Granularity



Fine Granularity



Parallelism expressed during compilation

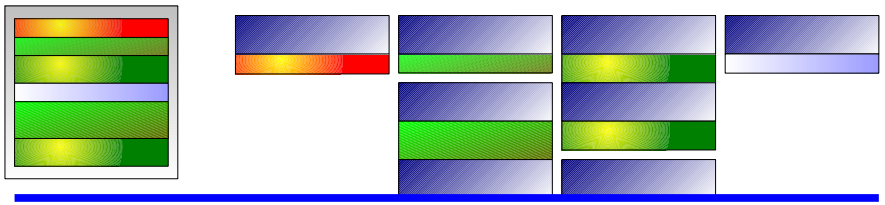
- Communication through shared variables
- OS-based runtime support

Relatively high threading overhead

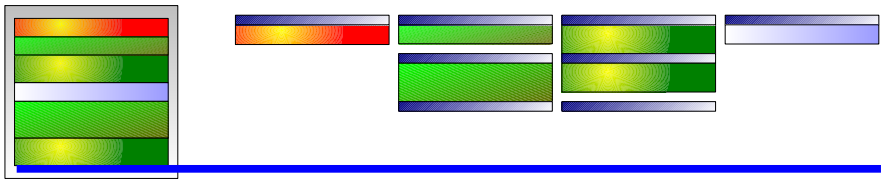
- **Good only for coarse granularity!**

The Main Idea

Regular Threads



Inthreads



Inthreads: An extremely lightweight threading mechanism

- Hardware-based runtime support
- Communication through shared registers

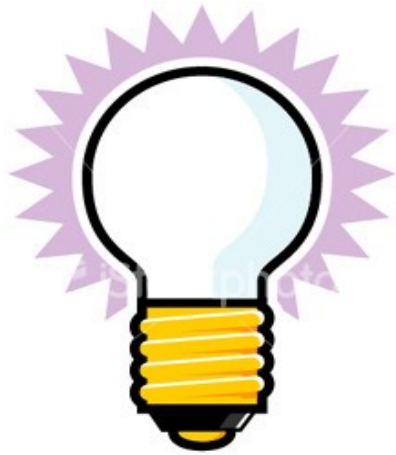
Medium-level granularity

- **Beyond the horizon for OOO**
- **Too fine-grain for regular threads**

Inthreads: Programming Model

Lightweight architecture

- Fixed number of threads
- Shared registers
- Synchronization instructions



Code regions explicitly belong to specific threads

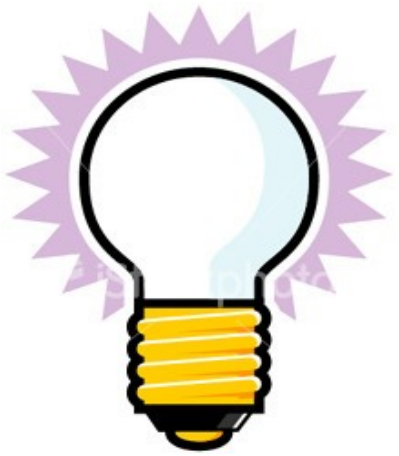
- Opportunity for better compiler optimizations

Fits within a function call frame. Function calls?

- inlining
- `suspend`/`resume`

Inthreads: Programming Model 2

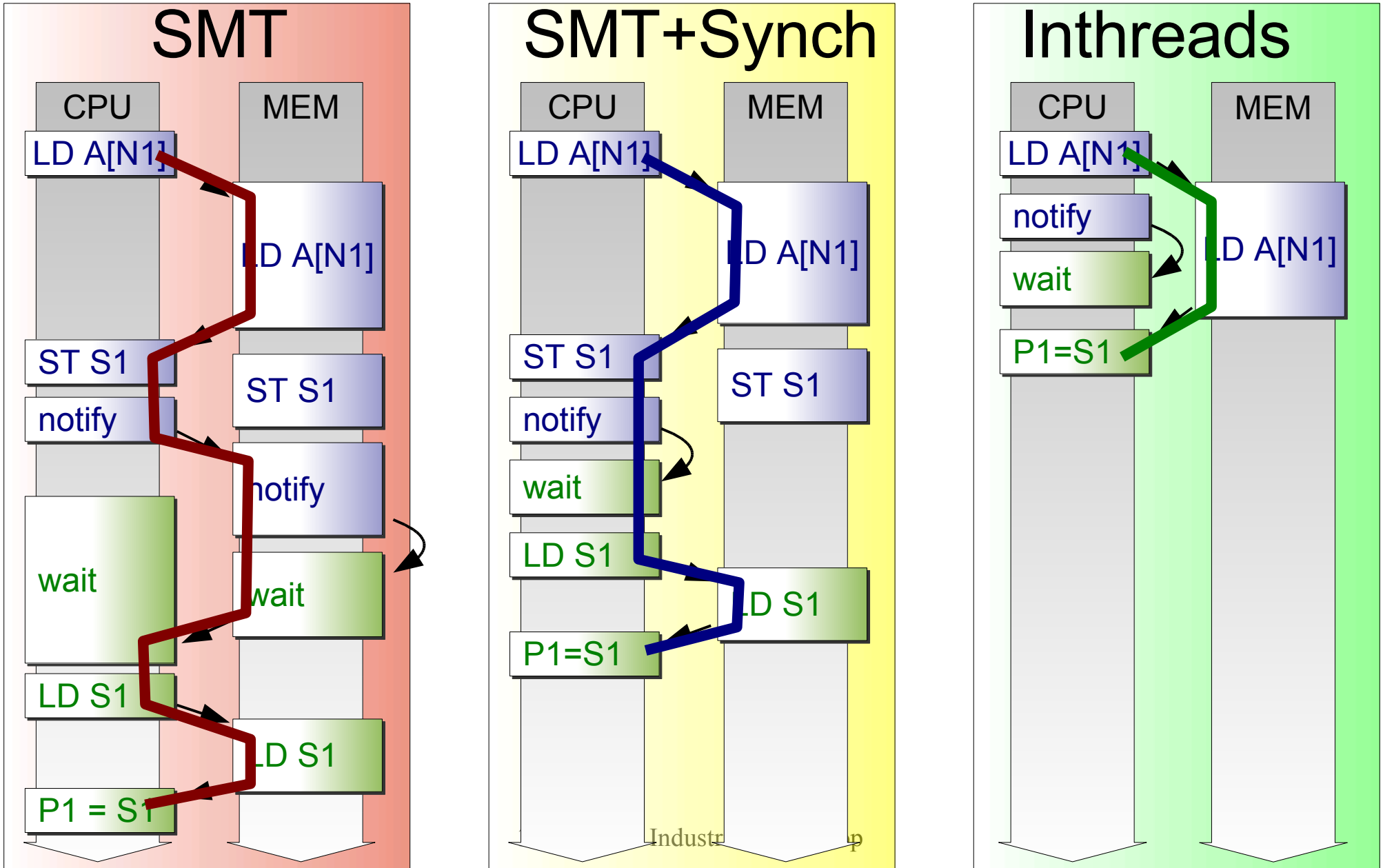
- Threads share the registers cooperatively
  - **Thread-private variables** use different registers in each thread
  - **Shared variables** must be allocated to same register in all threads
  - Accesses to shared variables must be **protected by synchronization**
- *Memory (**+register**) consistency model: Data-Race-Free-1 (DRF1)*
  - Software: no data races
  - Processor: obeys instruction ordering



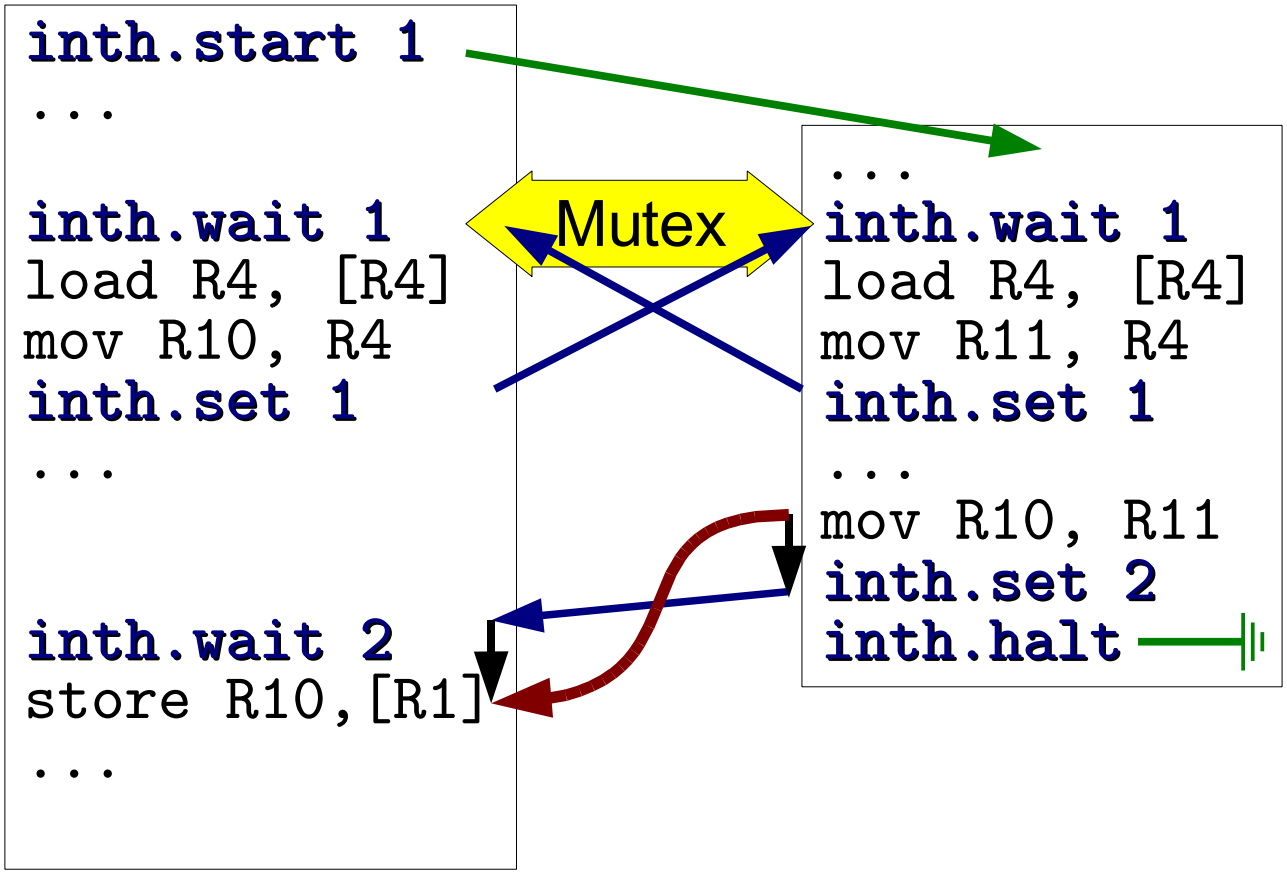
Inthreads vs. SMT: Fast Comm

Thread 0:
```
S1=A[N1]
notify()
```

Thread 1:
```
wait()
P1 = S1
```



Inthreads ISA by Example



Condition Registers

C0	0
C1	0
C2	1
...	...
C31	0

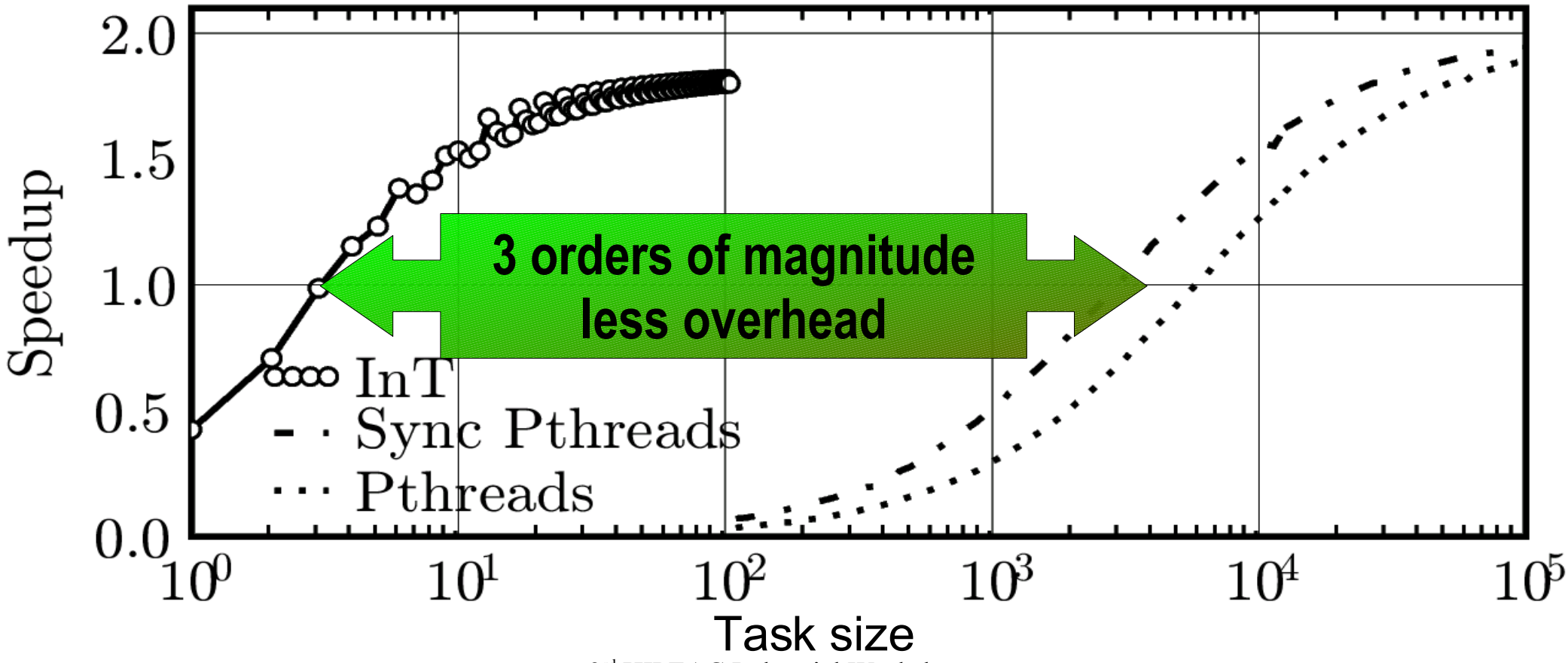
More Instructions:

inth.clr – clear a condition variable

inth.kill – kills a given thread

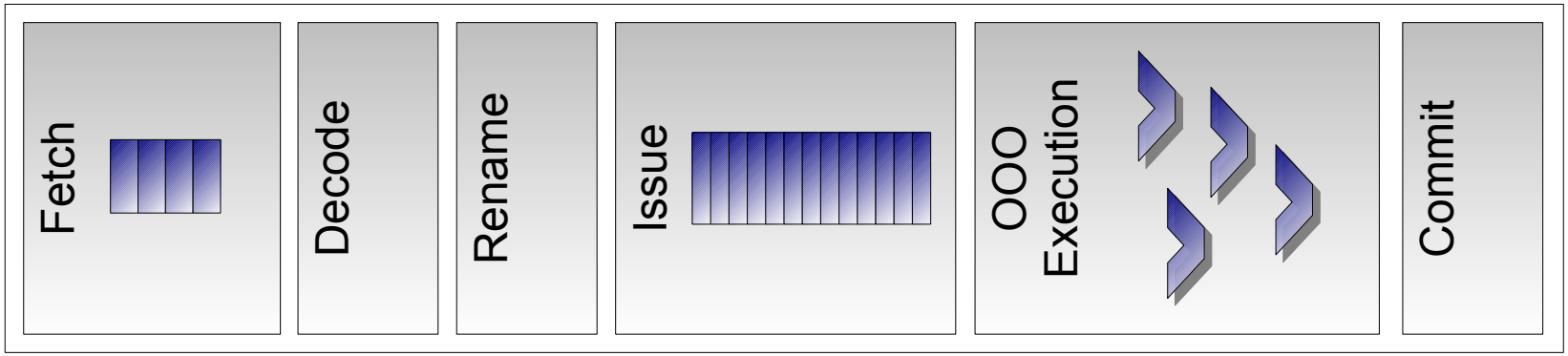
How lightweight it really is?

- Two independent tasks executed in parallel
  - Expected top speedup: 2x
  - Speedup 1x => Overhead is the same as task size

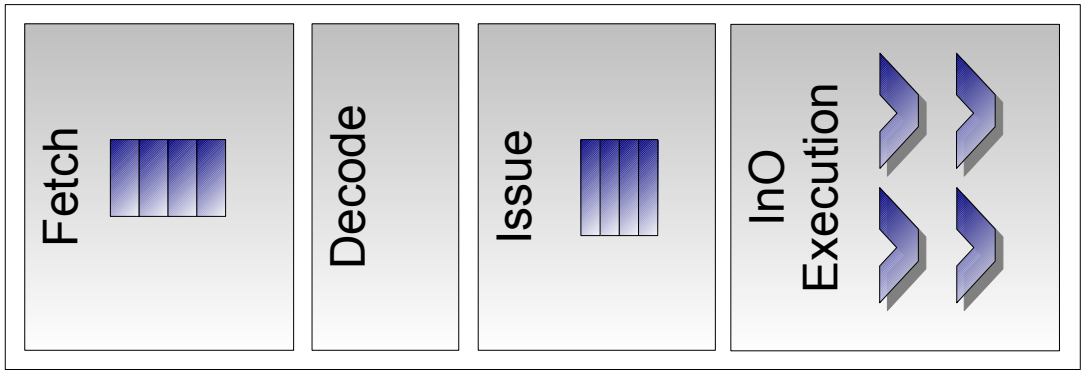


Pipeline Outline

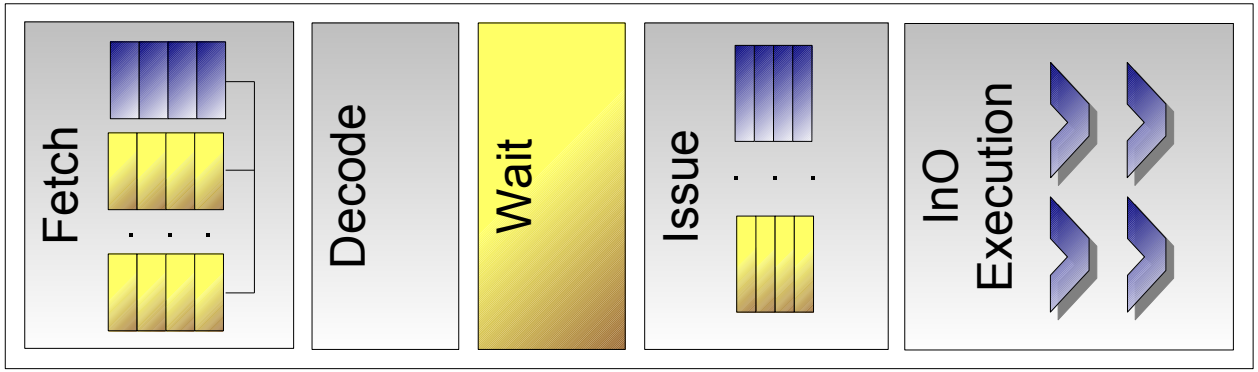
Out Of Order



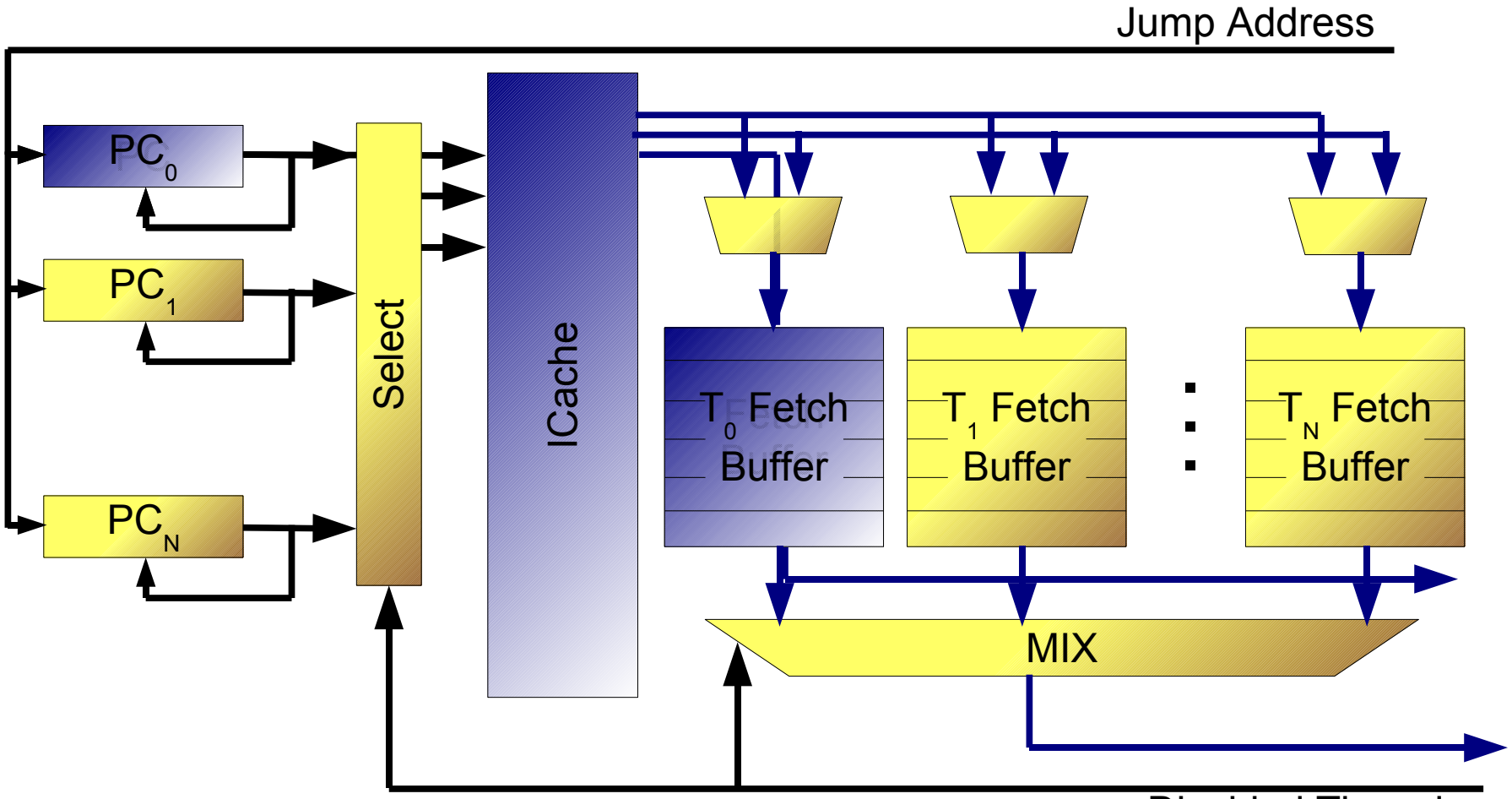
Inorder



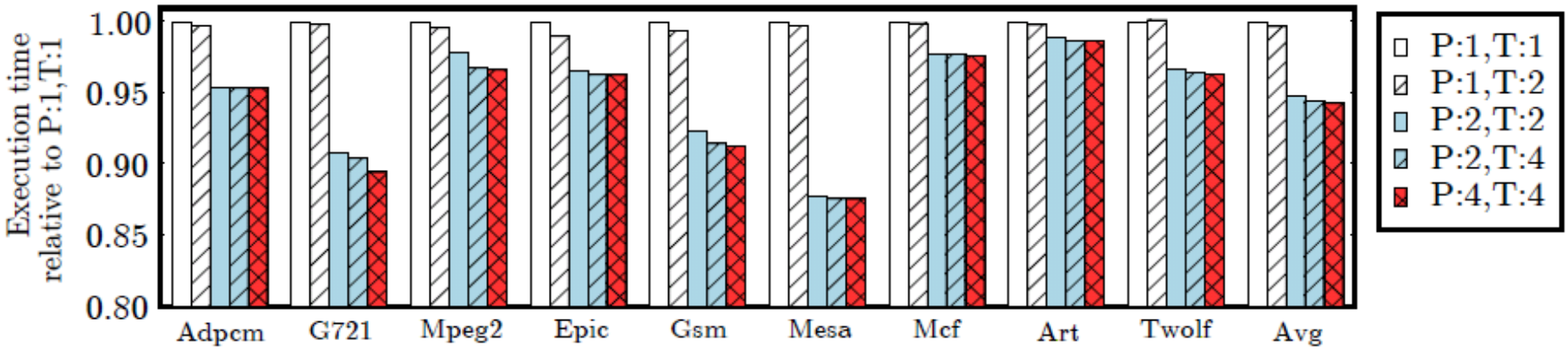
Inorder/ MT



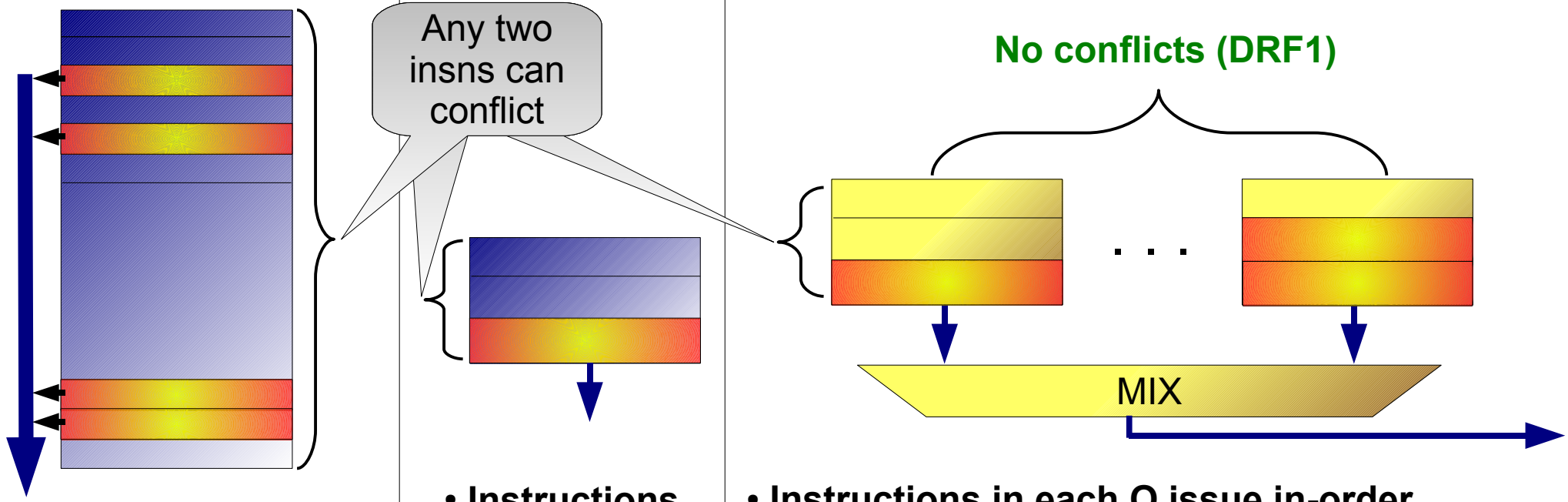
Fetch Stage



With banking, loading from 2 addresses is reasonable



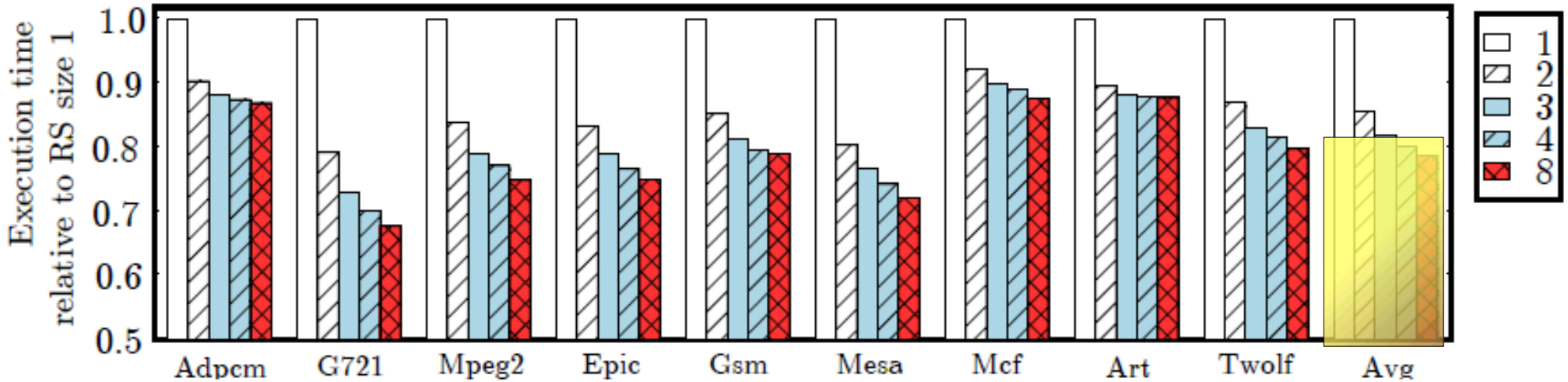
Instruction Issue



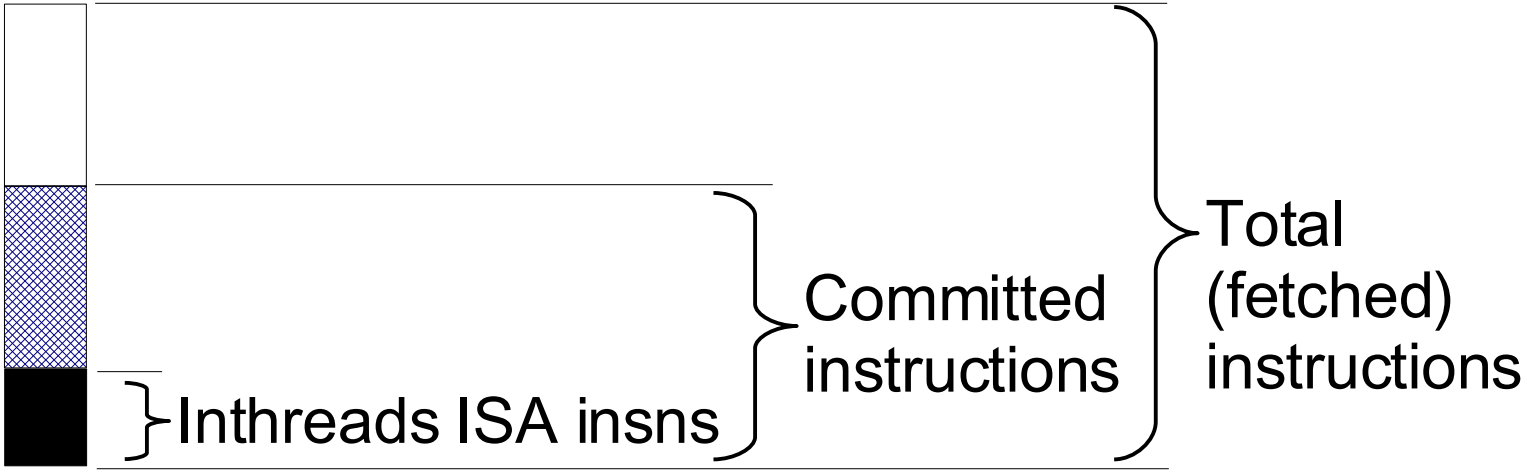
- Any insn can issue
- Long queue

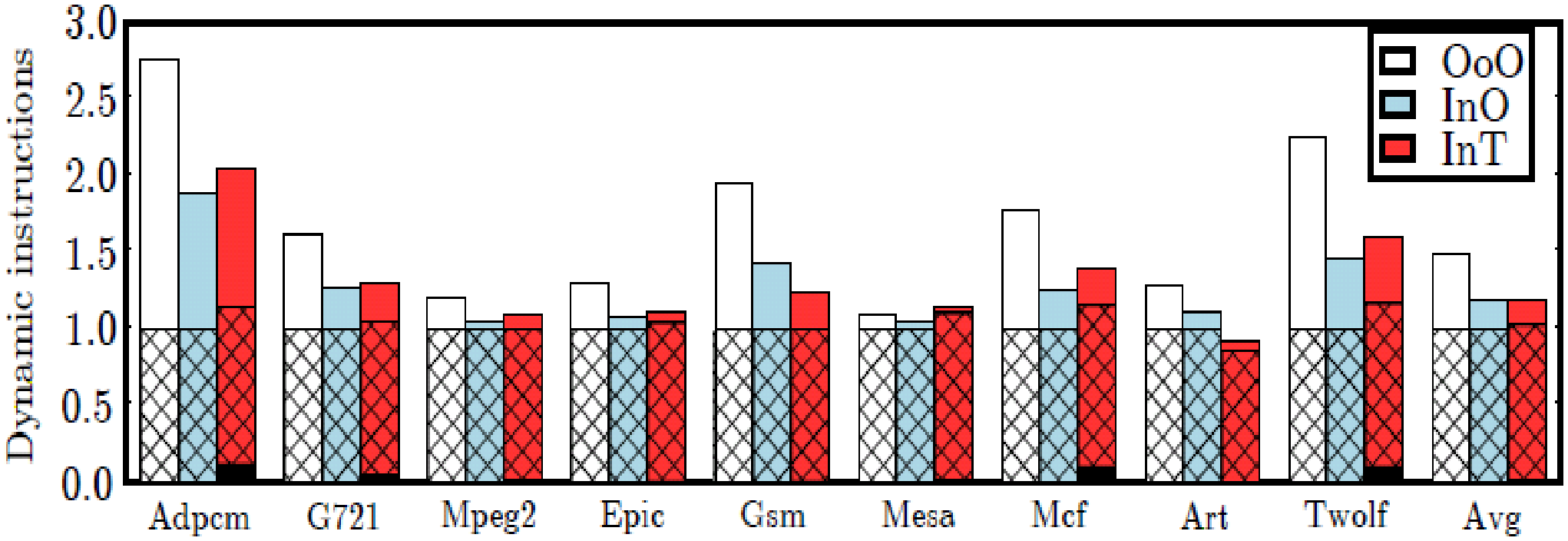
- Instructions issue in-order
- Short queue

- Instructions in each Q issue in-order
- Set of short queues
- IQs of inactive threads disabled by clock gating



Dynamic Instruction Mix

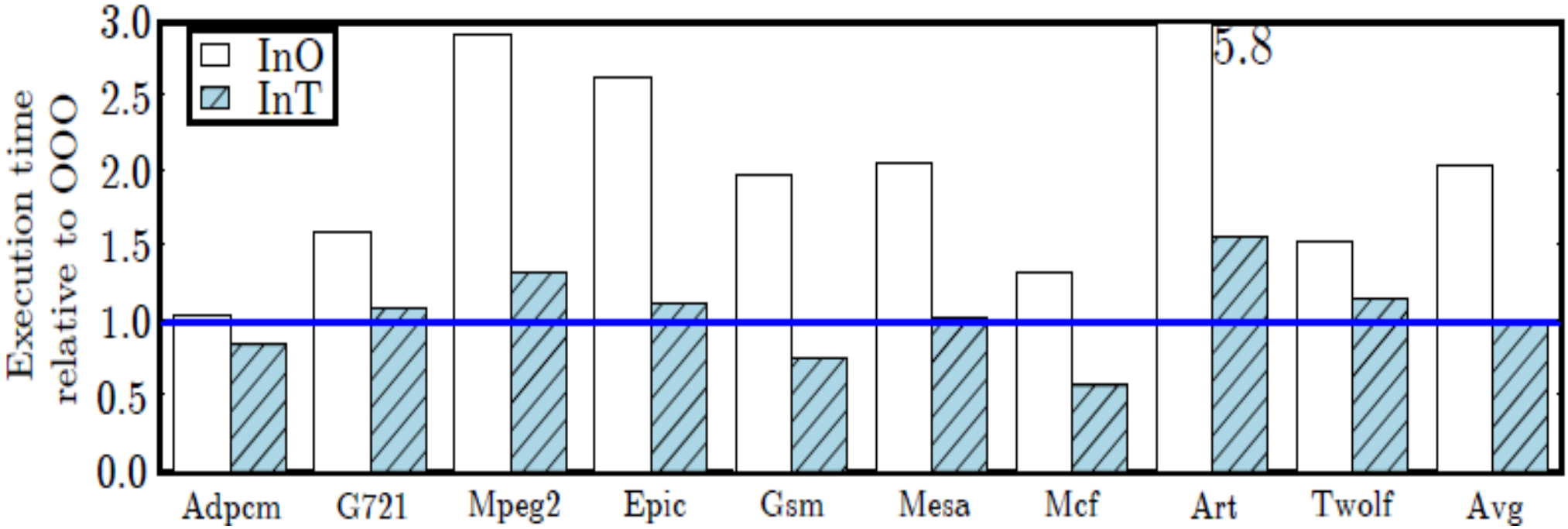




Inorder+Inthreads: less unnecessary work!

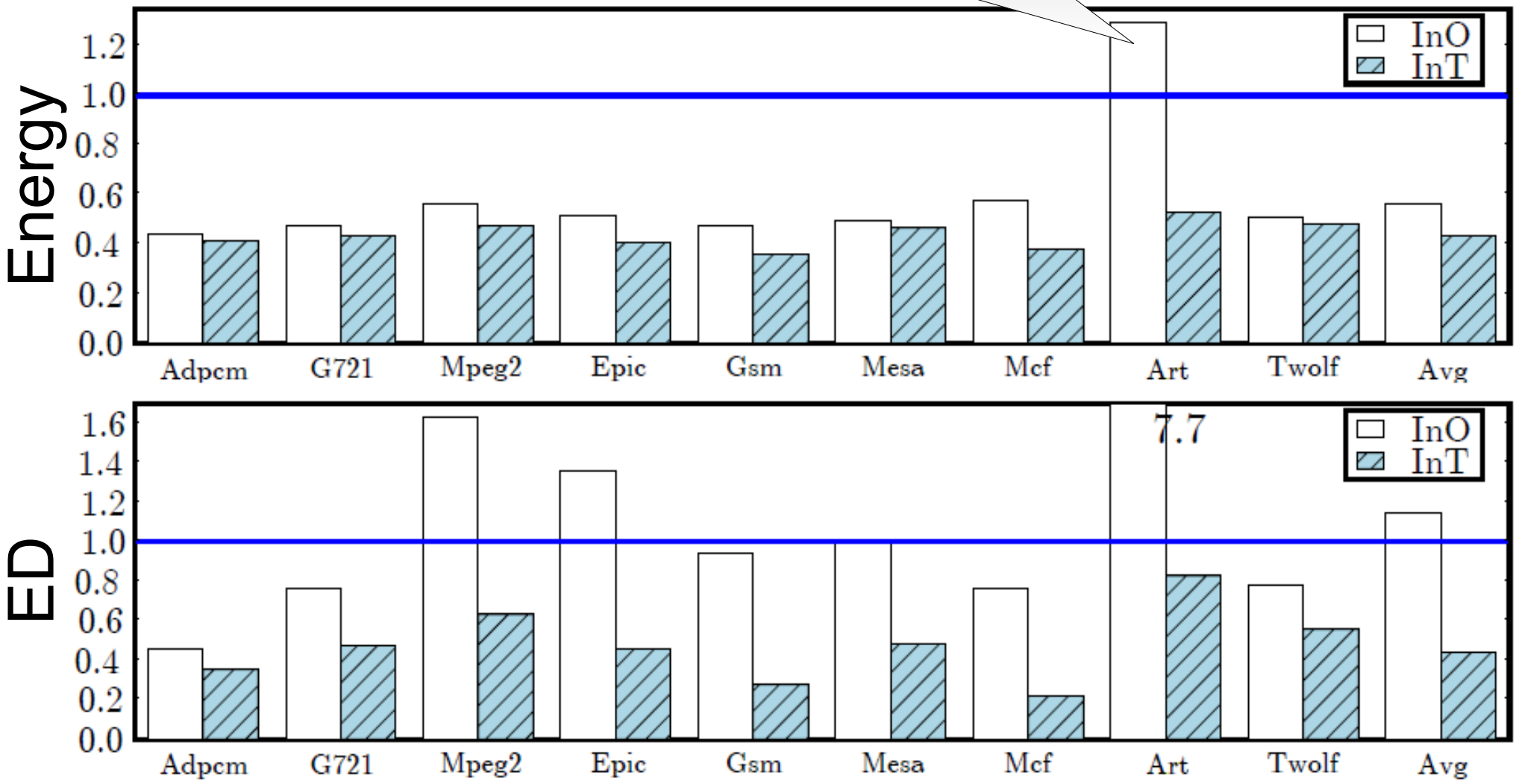
Execution Time

- Inorder is much worse than OOO...
- But Inorder+Inthreads isn't!
  - Although by different methods, Inorder+Inthreads achieves the same latency-tolerance as OOO



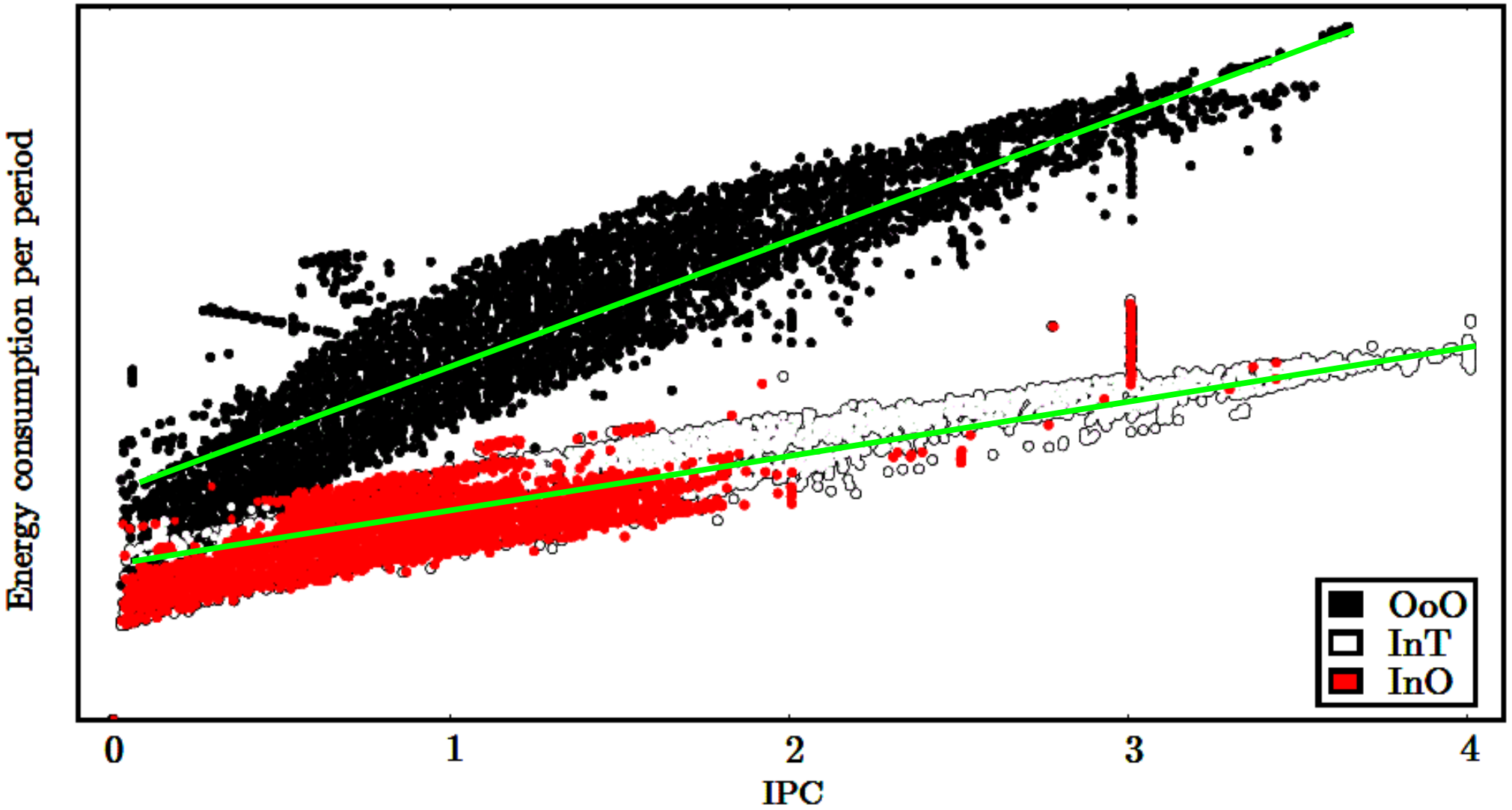
Energy, Energy-Delay Results

Processor mostly idle, waiting for memory. Little progress occurs, but power is consumed.



How IPC Affects Energy

- Divide execution into periods 1000 cycles long
- For each period, plot the energy consumed as a function of the IPC



Conclusions

OOO: **fast single task ↔ high-power**

MT: **energy efficient ↔ throughput-oriented**

Inthreads:

Shared Register Threads + Data-Race-Free-1

=

Best of both

Questions