A Problem with Large SMTs

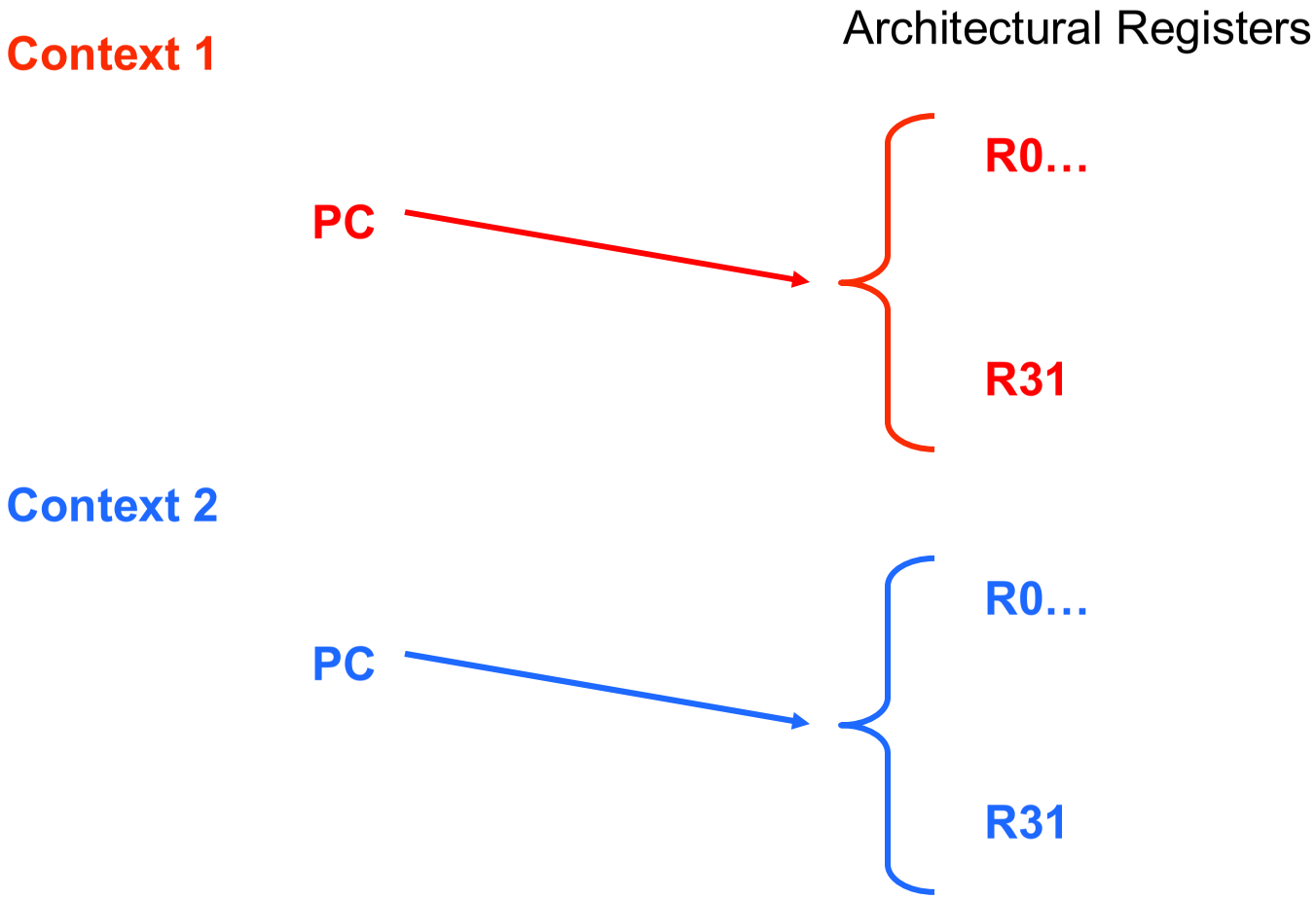
Two factors motivated mini-threads

- Register file will not scale in future technologies
  - e.g., 4-context Alpha 21464:
    - 3 cycles to access
    - 4 times the size of the 64KB instruction cache
- First SMT implementations will be small
  - limited to 2 hardware contexts
  - expect lower throughput gains

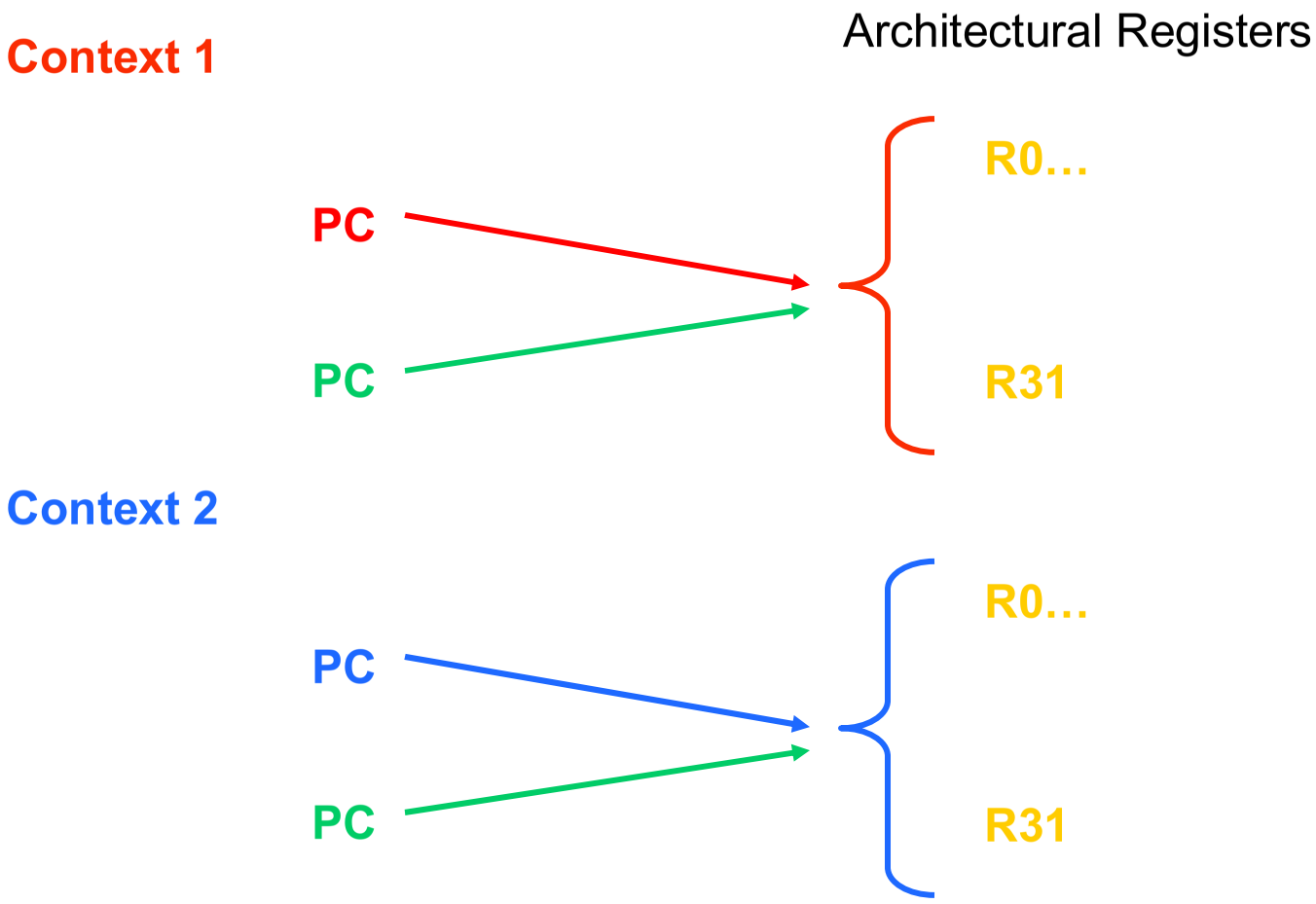
Goal of mini-threads: achieve performance gains of larger SMTs without the register cost

- execute more than one thread (a **mini-thread**) within each context
- mini-threads in a context **share** the context's architectural register set

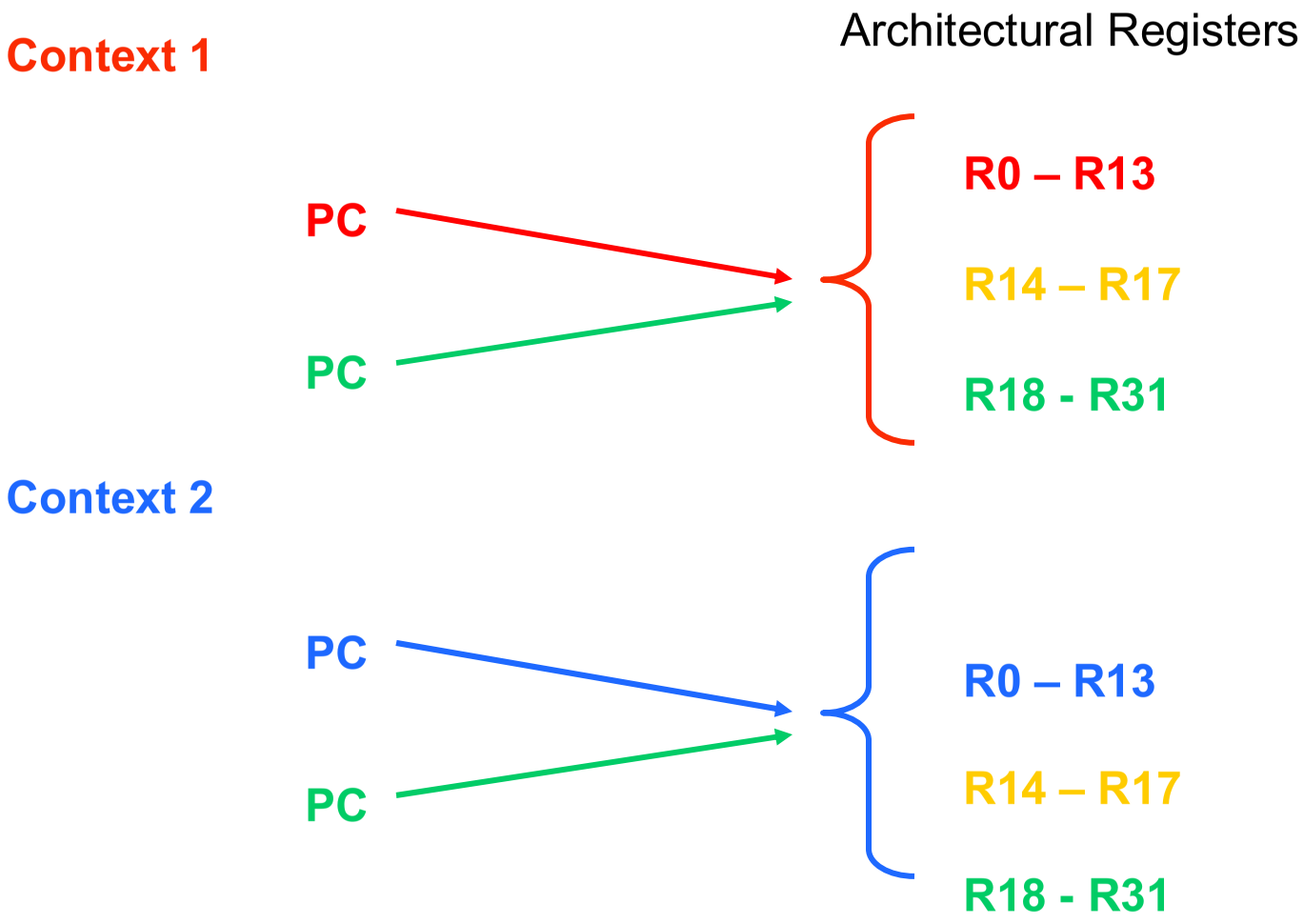
SMT Processor Model



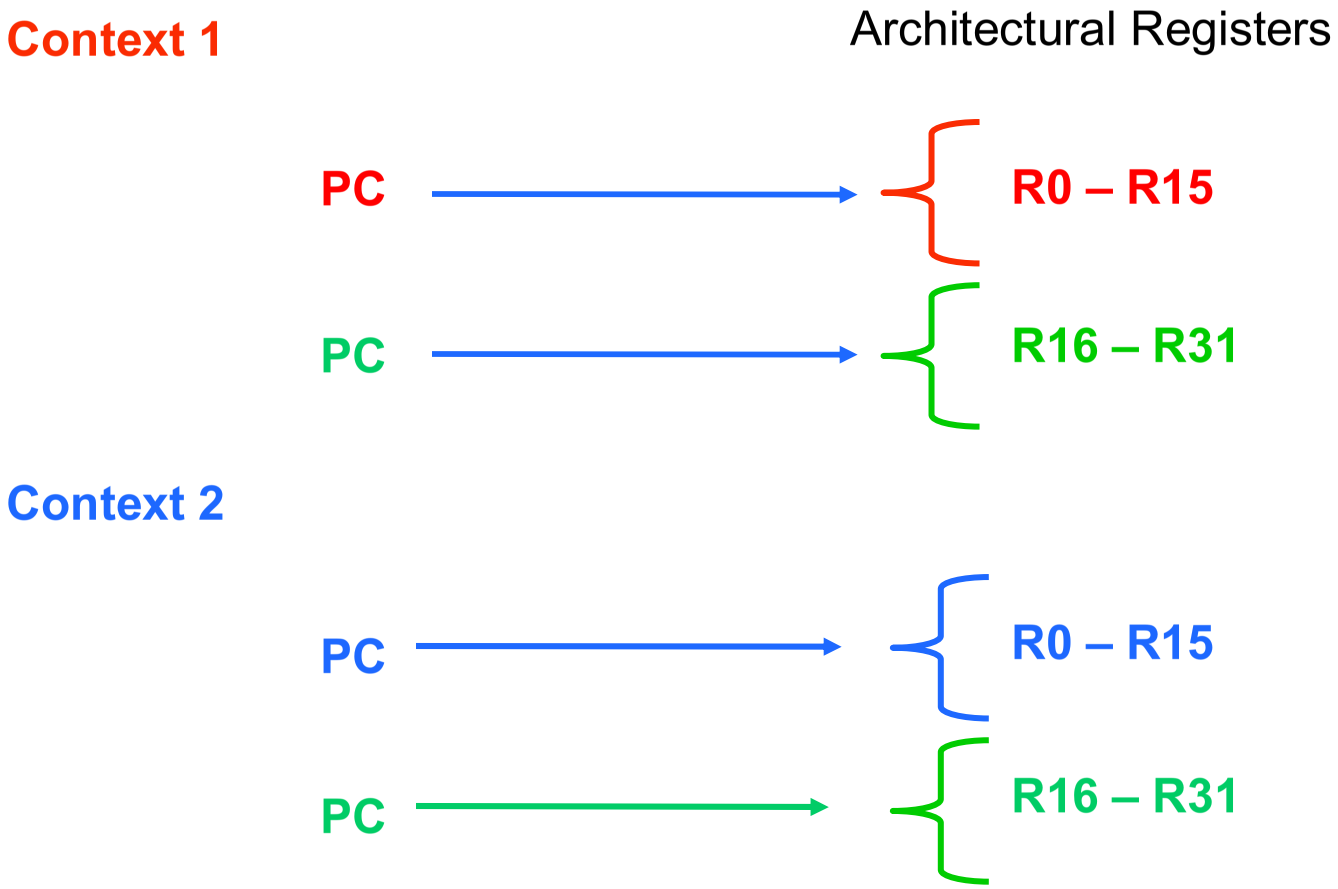
Mini-thread Processor Model



Mini-thread Processor Model



Mini-thread Processor Model

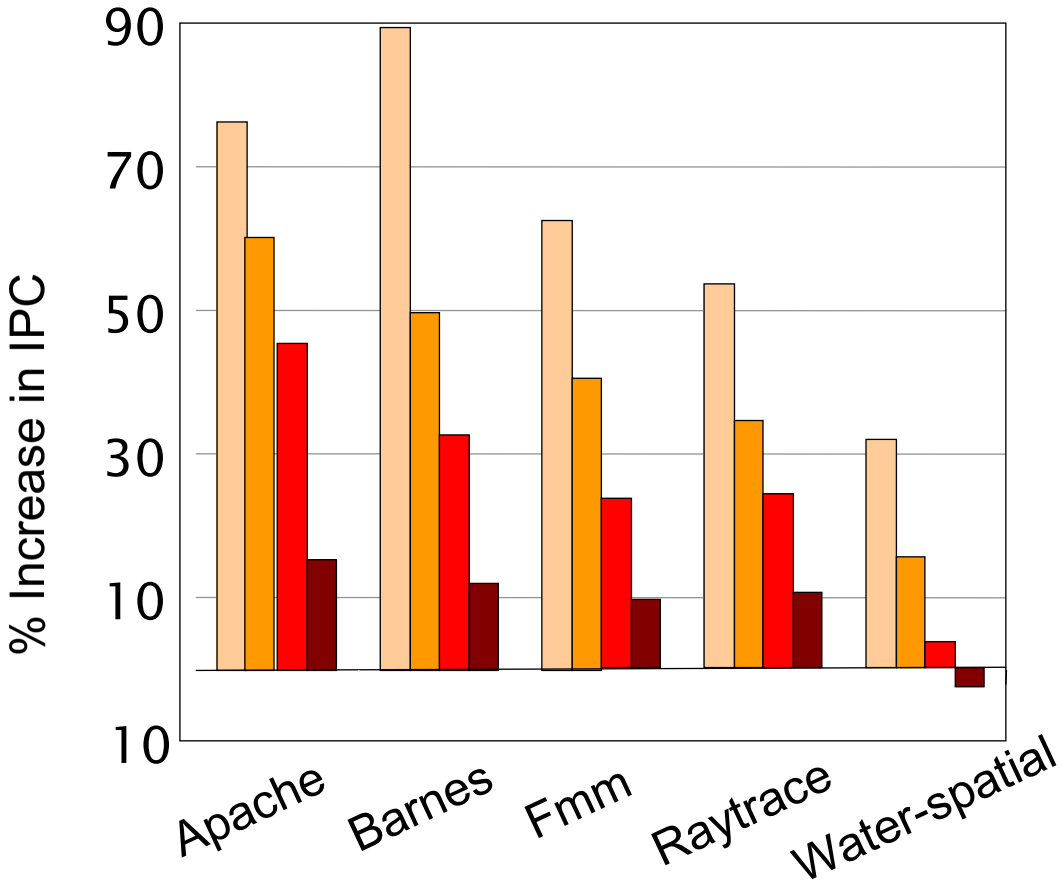


The Register/TLP Trade-off

+ Throughput benefit of extra mini-threads (more TLP)

-- Instruction cost of fewer registers per mini-thread (spill code)

Two Mini-threads/Context

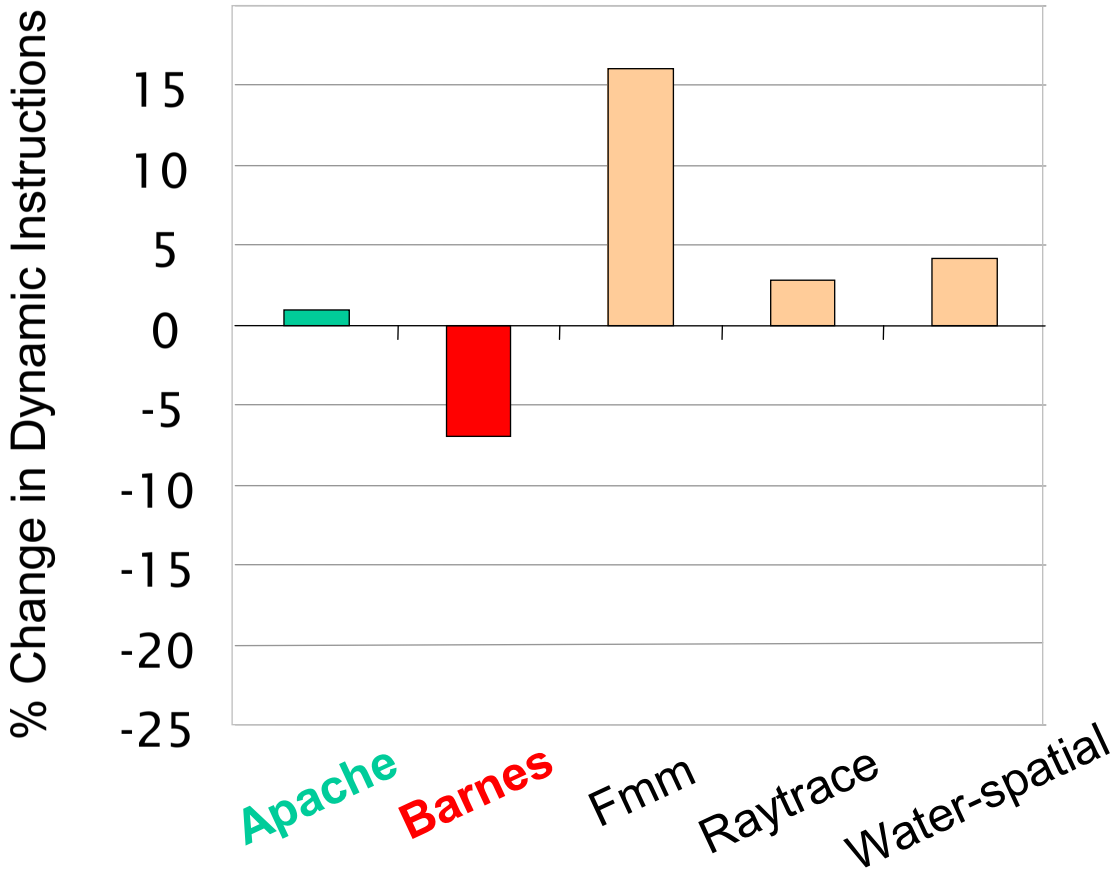


Workload Average

- 1 Context: 63%
- 2 Contexts: 40%
- 4 Contexts: 26%
- 8 Contexts: 9%

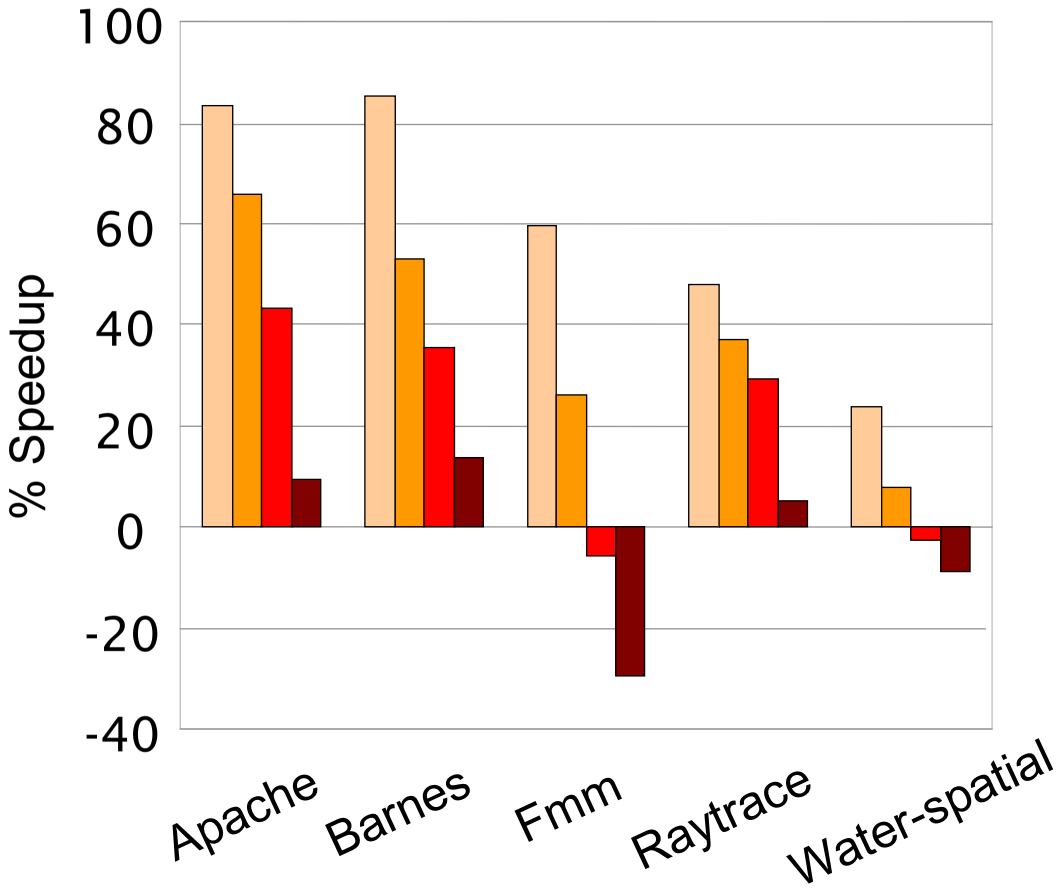
1 Context
2 Contexts
4 Contexts
8 Contexts

16 Registers per Mini-thread



- Only 3% loss on average
- Barnes: 7% savings
- Apache: 8%, mostly kernel execution

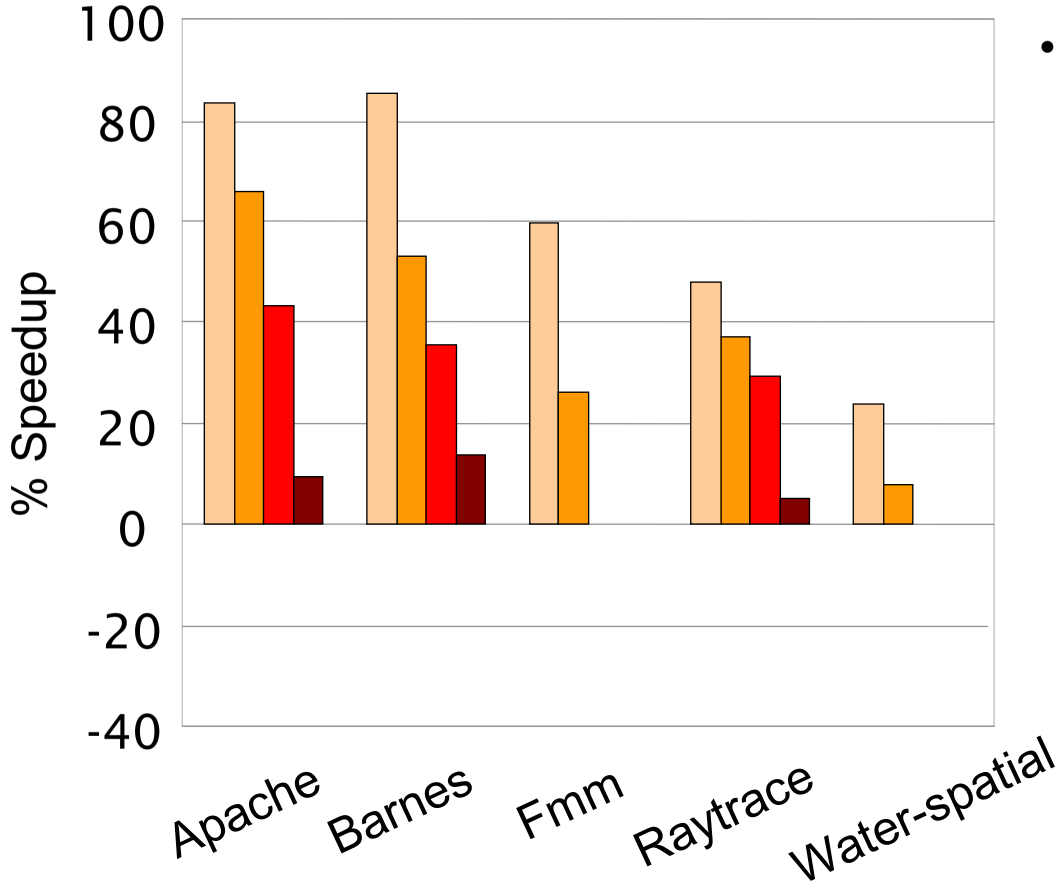
Bottom Line for a 2 Mini-threads on SMT



Workload Average

- 1 Context: 60%
- 2 Contexts: 38
- 4 Contexts: 20%
- 8 Contexts: -2%

Bottom Line for a 2 Mini-threads on SMT



Workload Average

- If free to choose:
  - 1 Contexts: 60%
  - 2 Contexts: 38%
  - 4 Contexts: **22**%
  - 8 Contexts: **6**%

Operating System & Runtime Support

The issues:

- Runtime and/or OS support for managing mini-threads
- Compatible register usage between mini-threads & OS/runtime

Operating System & Runtime Support

1. Dedicated server, e.g., web server
    - Goal: maximum OS performance
    - Solution:
        - 1 version of OS/runtime, recompiled for half the register set
        - each mini-thread uses different registers
        - hardware partition bit steers register access to high or low set

\+ Both mini-threads in a context can execute in the OS at the same time

-- Only 1 partition allowed & no sharing of values

Operating System & Runtime Support

2. Multiprogrammed environment
    - Goal: maximum flexibility for register use
    - Solution:
        - standard OS compile, multiple versions of the runtime
        - runtime schedules mini-threads
        - hardware blocks 1 mini-thread when the other enters the OS

\+ No restrictions on register usage and sharing among mini-threads

-- Lower throughput because mini-threads are blocked