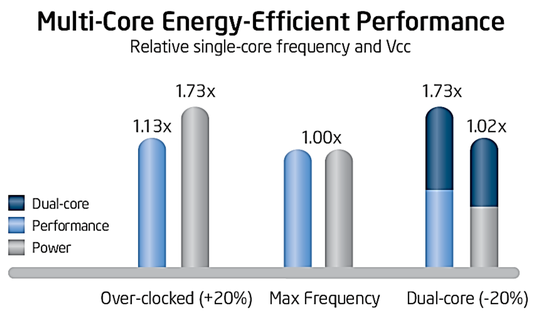
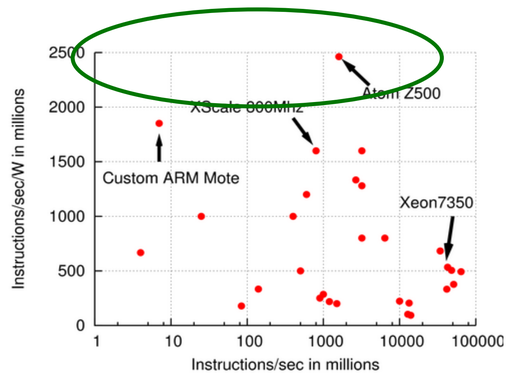
Teaching Parallelism to CS Undergraduates

Aviral Shrivastava, ASU

Why teach Parallelism?

- Need for higher performance continues
- Power-Efficiency is the bottleneck
- Multicores provide the only way to improve performance without commensurate increase in power
- Multicores are not a choice!





- New Moore's law – Number of cores will increase exponentially
- As no. of cores increase, each core becomes simpler, and less capable
- If we do not parallelize applications, then they will become slower with time
- Affects the whole software industry, as opposed to the processor design companies only

Projection: There will be a huge demand for parallel programmers in the coming deacade!

How to teach Parallelism?

- Can barely teach them sequential programming
- Is is possible/worthwhile to teach parallel programming before or in parallel with teaching sequential programming?
- All I learnt in CS undergraduate was recursion and pointers, and that has served me well.
- What will you remove from the current curriculum?
- How to scale the technique to a MOOC.

- What do we need to teach?
  - Multicore architectures, e.g., shared memory multicores, GPUs, NCCs, LLM
  - Thread programming, and issues of concurrency
  - Message-passing based programming, OpenMP
  - Higher-level programming models, e.g., Cilk, Habanero,
- How to teach?
  - Concepts → Exercises does not work as well.
  - Exercises → Concepts works better, but hard to do right.
  - Make sure there is a compelling story to tell.
  - "Flip model" of classroom may be naturally suitable

What not to teach?: Very political, need help in answering this question.