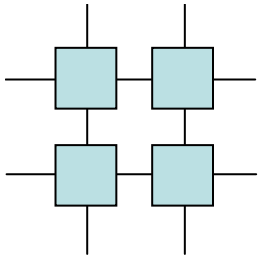
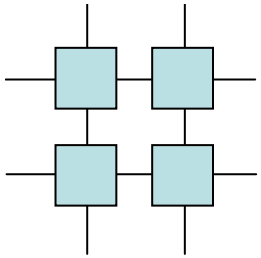
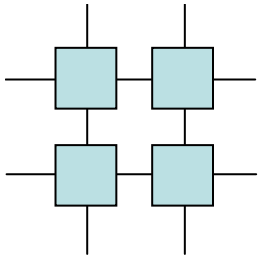


Performing experiments in Computing



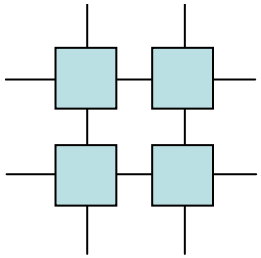
Purpose

- The target for much computer science is to provide the required result faster
  - Algorithmic improvements are measured in O
  - Experimental CS uses time, page faults, cache misses, etc
- In this class performance is all important and we measure in wall clock time



How

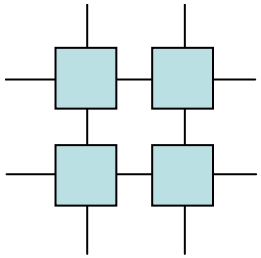
- Use wall clock – CPU time is easy to get down
- Consider your timing resolution
  - In this class wall-clock with seconds will do just fine



Speedup

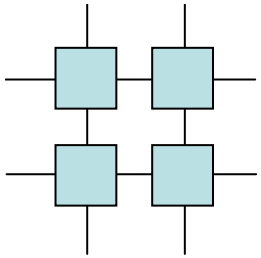
- Take a given problem an measure its runtime on a single CPU
- Measure the runtime of your parallel version

- Speedup = t_{seq}/t_{par}



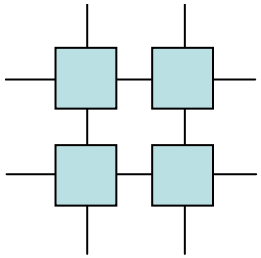
CPU Utilization

- How effectively are we using the extra CPUs
- Important metric
- Utilization = $(N_{CPU} * t_{seq})/t_{par}$



Gustavson

- Sometimes speedup is not the real mark
  - Rather we wish to solve larger problems in the same time
- Harder since we need to know the exact complexity of the algorithm



Cheating

- Do not do this
  - Compare your parallel version on 1 CPU with your parallel version on N CPUs and call it speedup
  - Use a better algorithm for your parallel version than your sequential version
    - The reverse is OK
  - Cheat with problem sizes
    - Like using SP and compare to DP