"The most important motivation for using a parallel system is the reduction of the execution time of computation-intensive application programs."

Performance Evaluation

- CPU Performance -- CPU time
  - user CPU
  - system CPU
  - waiting time -- not often mentioned
- IPS (Instructions Per Second) (MIPS, GIPS)
- FLOPS (FLoating point Operations Per Second) (MFLOPS, GFLOPS, PFLOPS)
- tools: measuring, benchmarking
  - gperf
  - Dhrystone (integer), Whetstone (floating)
  - Livermore Loops, Linpack (floating)
- SPEC benchmarks -- more rounded workload
Performance for Parallel Programs

- Parallel Runtime: $T_p(n)$ $p$ - processors, $n$ - size of problem
  - local computation, communication, synchronization, wait time ...
- Cost: processor, time product
  - $C_p(n) = p \times T_p(n)$
- Speedup: measure of how much parallelism help
  - $S_p(n) = T^*(n) / T_p(n)$, $T^*(n)$ is best sequential implementation
  - Theoretical limit: $S_p(n) \leq p$
  - $S_p(n) > p$?
    - New sequential algorithm!
- Practice, sometimes get $S_p(n)$ just a bit bigger than $p$, not asymptotically larger
  - caching effects
- Efficiency:
  - $E_p(n) = T^*(n) / C_p(n) + S_p(n) / p$
  - $E_p(n) = 1$ is perfect speedup.

- Amdahl’s Law:
  - Fraction $f$ ($0 \leq f \leq 1$) of program is sequential, e.g. $T^*(n)*(1-f)$ is parallelizable
  - $S_p(n) \leq T^*(n) / (f \times T^*(n) + ((1-f)/p) \times T^*(n)) = 1/(f+(1-f)/p) \leq 1/f$
Scalability and communication costs

Scalability -- the idea that $E_p(n)$ can remain constant as $p$ and $n$ increase.
- Increase in problem size and number of processors gives same efficiency
- Scalability discussions in other texts are better ...

Communication costs on various models
- Single-broadcast, scatter, multi-broadcast, total exchange
  - Ring:
  - $d$-dimensional mesh:
  - Hypercube: