GUSTAFSON-BARSIS LAW

(QUINN) focus of Amd's law is minimizing exec. time
(we treat pb size as a constant, and demonstrate how increasing #procs can reduce time

often the goal of //lism is to increase the accuracy of a solution that can be computed in a fixed amount of time.

treat time as a constant,
let pb size increase with the #procs

inherently seq fraction of a computation typically decreases as pb size increases (Amdahl effect)

$$\psi(n,p) \leq \frac{\sigma(n)+\phi(n)}{\sigma(n)+\phi(n)/p} \quad \text{(speedup)}$$

s = fraction of time spent in the parallel computation performing inherently seq. ops

$1-s$ = " " " " " " " " parallel ops

$$s = \frac{\sigma(n)}{\sigma(n)+\phi(n)/p} \qquad 1-s = \frac{\phi(n)/p}{\sigma(n)+\phi(n)/p}$$

$$\sigma(n) = (\sigma(n) + \varphi(n)/p)\, s$$

$$\varphi(n) = (\sigma(n) + \varphi(n)/p)(1-s)p$$

$\rightsquigarrow$ **G-B law**

$$\boxed{\psi(n,p) \leq p + (1-p)s}$$

given a parallel program solving a pb of size n on p procs let s denote the fraction of total exec. time spent in serial code,
the max speedup achievable by this program is:

(ex) an application on 64 procs requires 220 secs to run
5% of the time is spent executing serial portions of the computation on a single processor.
what is the speedup of the application?
$s = 0.05$, $p = 64$, $\psi \leq 64 + 63 * 0.05 = 60.85$

(ex) we want to demonstrate that a new supercomputer with 16384 procs can achieve a speedup of 15000 on a specific pb.
what is the max fraction of the parallel execution time that can be devoted to inherently seq. ops?
$15000 = 16384 - 16383 \cdot s \rightsquigarrow s = 0.084$

KARP-FLATT METRIC

(AMD GB) -> we ignored $k(n,p)$ the parallel overhead term.
~> overestimate speedup

exec. time of a // progr on p proc^s:

$$T(n,p) = \underbrace{\sigma(n)}_{\text{serial}} + \underbrace{\varphi(n)/p}_{\text{parallel}} + \underbrace{k(n,p)}_{\text{com/tion}}$$

serial program: $T(n,1) = \sigma(n) + \varphi(n)$

DEF
experimentally determined serial fraction [e] of the parallel computation is the total amount of idle and overhead time scaled by two factors: # proc^s - 1 and seq. exec. time

$$e = \frac{(p-1)\sigma(n) + p\,k(n,p)}{(p-1)T(n,1)}$$

$$\psi = \frac{T(n,1)}{T(n,p)}$$

K-F metric

given a parallel computation exhibiting speedup ψ on p proc^s ($p > 1$), the exp. determ. serial fraction e is

$$e = \frac{1/\psi - 1/p}{1 - 1/p}$$

for a pb of fixed size, the efficiency of a parallel computation typically decreasses as # proc^s increases.

using e, we can determine the reasons for the decrease:
(1) limited opportunities for parallel [illegible]

ex I) benchmark of a II program on 1, 2, .. 8 proc's give the following speedup results:

p	2	3	4	5	6	7	8
ψ	1.82	2.50	3.08	3.57	4.00	4.38	4.71

why does the II program achieve a speedup of only 4.71 on 8 proc's?

we compute e for each (p, ψ) pair:

e	0.10	0.10	. . .	0.10

since e is not increasing with #proc's, the primary reason for the poor speedup is limited opportunity for parallelism: large fraction of inherently seq's computations

ex II)

p	2	3	4	5	6	7	8
ψ	1.87	2.61	3.23	3.73	4.14	4.46	4.71

(same question as before)

e	0.070	0.075	0.080	0.085	0.090	0.095	0.1

since e increases steadily as #proc's increases, the primary reason for the poor speedup is parallel overhead.

ISOEFFICIENCY METRIC (11)

scalability: measure of the ability of a parallel program to increase performance as #procs increases.

ISOEF. METRIC ~~METRIC~~ ~~DERIVATION~~ (DERIVATION ommitted)

Suppose a parallel program exhibits efficiency $\varepsilon(n,p)$ [speedup divided by p]

define $C = \frac{\varepsilon(n,p)}{1-\varepsilon(n,p)}$

$T_0(n,p) = (p-1)\sigma(n) + p\kappa(n,p)$ → total amount of time spent by all procs doing work not done by seq. alg.

In order to maintain the same level of efficiency as #procs increases, n must be increased so that the following ineq. holds:

$$T(n,1) \geq C\,T_0(n,p)$$

$T(n,1)$: seq exec. time

concl. maintain efficiency when increasing #procs ~> increase size of pb being solved