Reproducible BLAS
(Basic Linear Algebra Subprograms)

Jim Demmel, Hong Diep Nguyen, Peter Ahrens

Computer Science Division

UC Berkeley

Motivation (1/2)

- Since roundoff makes floating point addition nonassociative, different orders of summation often give different answers
- On a parallel machine, the order of summation can vary from run to run, or even subroutine-call to subroutine-call, depending on scheduling of available resources, so answers can change
- Why is this important?

Motivation (2/2)

- Email on NA-Digest: Commercial finite-element SW vendor wanted a parallel reproducible sparse linear equation solver, because his customers (civil engineers) had contractual obligations to their customers to get the same answer from run to run: "Will the bridge fall down or not?"
- Responses from ~100 UC Berkeley faculty to email query about the importance of reproducibility:
  - Most common: How will I debug without reproducibility?
  - How do I do fracture mechanics, where I do many random simulations looking for a very rare event, and when one occurs, I need to resimulate it exactly, while computing some side information?
  - What if my "illegal underground nuclear test detector" (funded by the United Nations) says "They did it!" and then "They didn't do it"?

Reproducible BLAS

- First step in longer term goal of making ScaLAPACK, LAPACK etc reproducible
- Simplest algorithm for reproducible sum $s = \Sigma_i x(i)$
  1. Compute $M = \max_i |x(i)|$; exact and so reproducible
  2. Round all x(i) to 1 ulp (unit in last place) of M; error introduced no worse than usual error bound
  3. Add rounded x(i); they behave like fixed point numbers so summation exact and so reproducible
- Drawback: costs 3 passes over data in serial, or 3 reduction/broadcast steps in parallel
- Better: can do it in 1 pass, or 1 reduction, by interleaving all 3 steps

Performance results on 1024 proc Cray XC30

1.2x to 3.2x slowdown vs fastest (nonreproducible) code dasum

data for n=1M summands on up to p=1024 processors

3 reproducible sum algorithms compared, best one depends on n, p

code and papers at bebop.cs.berkeley.edu/reproblas