Fault-tolerant iterative methods via selective reliability

Mark Hoemmen and Mike Heroux
{mhoemme, maherou}@sandia.gov

Sandia National Laboratories

Householder Symposium XVIII
Tahoe City, CA
12-17 June 2011

---

1Sandia National Laboratories is a multiprogram laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.
“Parity is for farmers.”
– Seymour Cray, on the CDC 6600 (a 1930’s New Deal reference)

“. . . I remarked to Dennis [Richie] that easily half the code I was writing in Multics was error recovery code. He said, ‘We left all that stuff out. If there’s an error, we have this routine called panic, and when it is called, the machine crashes, and you holler down the hall, “Hey, reboot it.” ’ ”

– Tom van Vleck, Multics developer
Motivation

- Correct arithmetic & data cost energy
  - Parity data and checksums
  - Redundant computation and storage
  - Communicating agreement on redundant computations
- Extreme-scale parallelism: correctness is costly
  - More components, so faults more likely
  - Extremely energy-constrained
- Consumer applications drive hardware
  - Many consumer apps tolerate some faults
  - Mobile devices also energy-constrained
- Numerical algorithms have *latent* fault tolerance
  - Iteration, convergence, “inexact Krylov,” . . .
  - *Certain parts* require reliable computation
  - Current algorithms *overconstrain* reliability
Fault terminology

- **Fault** happens inside a function. It may or may not produce correct output as a result.
- **Soft** faults do not interrupt the program immediately. User code can detect them via introspection.
- **Hard** faults interrupt the program. The program that suffers them cannot detect them directly.

A failure is a fault that "leaks out," so the function misbehaves from an outside perspective.

Relative to current level of abstraction.

**Key:**
- Dotted outline: Beyond our scope
Reliability models

- Can’t reason about code behavior without a model
- Current model: “Fail-stop”
  - System tries to detect all soft faults
  - Turn all detected soft faults into hard faults
- Our basic model: “Sandbox”
  - Isolate unreliable computation in a box
  - Reliable code invokes box as a function
- Additional desired features of a model
  - Detection: report faults to application
  - Transience: refresh / recompute unreliable data periodically
  - Embed into type system: compiler can help you reason
Desired properties of a fault-tolerant iterative method

- Converge eventually
  - No matter the fault rate
  - Or it detects and indicates failure
  - Not true of iterative refinement!
- Convergence degrades gradually as fault rate increases
  - Easy to trade between reliability and extra work
- Requires as little reliable computation as possible
- Can exploit fault detection if available
  - e.g., if no faults detected, can advance aggressively
Origin of Fault-Tolerant GMRES

- GMRES converges eventually
  - As long as Krylov subspace keeps growing
  - The algorithm tells you otherwise
- Flexible GMRES allows changing preconditioner
  - It can change a lot (no perturbation theory)
  - Preconditioner fault = “changing” preconditioner
  - Rank-revealing decomposition of upper Hessenberg detects whether Krylov subspace keeps growing
- “Preconditioner” can be an arbitrary solver
  - Whatever iteration + preconditioner you were using before
  - Inner solves run unreliably, outer solver runs reliably
  - Expect inner solves to take most of the time
- Fault-Tolerant GMRES (FT-GMRES) =
  - Flexible GMRES as an inner-outer iteration
  - With unreliable inner iterations
Fault-Tolerant GMRES (FT-GMRES) algorithm

**Input:** Linear system $Ax = b$ and initial guess $x_0$

$r_0 := b - Ax_0$, $\beta := \|r_0\|_2$, $q_1 := r_0 / \beta$

for $j = 1, 2, \ldots$ until convergence do

Inner solve: Solve for $z_j$ in $q_j = Az_j$  \hspace{1cm} $\triangleright$ The only unreliable part

$v_{j+1} := Az_j$

for $i = 1, 2, \ldots, k$ do

$H(i, j) := q_i^* v_{j+1}$, $v_{j+1} := v_{j+1} - q_i H(i, j)$ \hspace{1cm} $\triangleright$ Orthogonalize $v_{j+1}$

end for

$H(j + 1, j) := \|v_{j+1}\|_2$

Update rank-revealing decomposition of $H(1:j, 1:j)$

if $H(j + 1, j)$ is less than some tolerance then

if $H(1:j, 1:j)$ not full rank then

Try recovery strategies discussed in paper

else

Converged; return after end of this iteration

end if

else

$q_{j+1} := v_{j+1} / H(j + 1, j)$

end if

$y_j := \text{argmin}_y \|H(1:j + 1, 1:j) y - \beta e_1\|_2$  \hspace{1cm} $\triangleright$ GMRES projected problem

$x_j := x_0 + [z_1, z_2, \ldots, z_j] y_j$  \hspace{1cm} $\triangleright$ Solve for approximate solution

end for
FT-GMRES can run through faults

- FT-GMRES can run through faults and still converge.
- Standard GMRES, with or without restarting, cannot.

FT-GMRES vs. GMRES on Ill_Stokes (an ill-conditioned discretization of a Stokes PDE).

FT-GMRES vs. GMRES on mult_dcop_03 (a Xyce circuit simulation problem).
Empirical observation: FT-GMRES convergence slows gradually as fault rate increases.

FT-GMRES on Ill_Stokes problem, with different fault rates in inner solves’ SpMVs.

FT-GMRES on mult_dcop_03 problem, with different fault rates in inner solves’ SpMVs.
Observed gradual degradation of convergence (2 of 2)

FT–GMRES(50,300): Fault rate vs. tolerance for mult_dcop_03 problem, with faulty SpMVs in the inner solves (deterministic faults)

Number of outer iterations to convergence for FT-GMRES (50 iterations per inner solve, max 300 outer iterations) on mult_dcop_03 problem, vs. fault rate in the inner solves’ SpMVs, and the outer solves’ convergence tolerance.
Advantages of our approach

Existing approach:
- System overconstrains reliability
- “Fail-stop” model
- Checkpoint / restart
- Application is ignorant of faults

Our approach:
- System lets application control reliability
- Tiered reliability
- “Run through” faults
- Application listens for and responds to faults
Reference and future work

- See our Supercomputing 2011 submission
  - PDF at http://www.sandia.gov/~maherou/
- In progress: Collaboration with systems researchers
  - Allocation of “unreliable memory”
  - Fault injection in user space
    - Can run at scale without making cluster admins angry
  - Trilinos (trilinos.sandia.gov) FT-GMRES prototype
    - MPI and hybrid-parallel already
    - Inner solvers are preconditioned