Unraveling Real Time

Dr. Christopher Landauer
Information Systems & Cyber Divison

03 December 2018

Approved for public release. OTR 2019-00054

Unraveling Real Time

Dr. Christopher Landauer
The Aerospace Corporation
chris.landauer@aero.org

Flight Software Workshop
03-06 December 2018
Southwest Research Institute
San Antonio, Texas

Outline

- Overview / Summary
- Hardware Assumptions
- Hardware Speculations
- Software Approach
- Software Implications
- Software Issues
- Mathematical Implications and Speculations
- Conclusions and Prospects

Overview / Summary

- Issue
  - *Real-time coordination required for flight software is complicated;*
    but perhaps it is not as necessary as we think.
  - *This presentation is a speculation about the issue.*
- Hypothesis
  - *Numerical computations need recent enough data, not always the most recent data.*
- Implications
  - *There are hardware, design, software and mathematical issues.*
- The main architectural idea:
  - *Assign all separate tasks into many small concurrent processors,*
    one task per processor, running at different speeds.
  - *Communication via direct or targeted multicast writes.*
- There are many experiments to be performed.
  - *All are marked in red (experiment)*

Hardware Assumptions

- Processors that can run at different speeds and save power at slower speeds.
- Concurrent processors with independent clocks:
  - *Incoming data all timestamped (by sender AND receiver clock)*
  - *They can use that to synchronize clocks* *(experiment)*
    (cyclone synchronization algorithm from 1998 or so),
  - *or not (if this idea works, small drift rates will not matter).* *(experiment)*
- Direct Memory Access (DMA) / Multicast rules: *(experiment)*
  - *No interrupts at receiver on writes:*
    - The receiver uses whatever values are there when it accesses them.
  - *Overlapping writes are possible; collect them for consensus.*
    - Complain to sender only if the values are different.
  - *Overlapping reads cannot happen:*
    - There is no parallelism in each processor's main computational task.
  - *Writes take precedence over reads;*
    - reads wait for writes to finish, unless read is far along in its operation. *(experiment)*
- Small Processor memories can be fast:
  - *DDR3 800MHz (beagle bone) has 10-15ns latency, 15-20ns for 8 bytes*

Hardware Speculations

- New idea (maybe) - ECC processors
  - *Many modern computers use ECC memory,*
    - There is a common (72, 64) code used at least since the VAX 11/780, derived from the Hamming (127, 120) code,
  - *but none use ECC processing;*
  - *No modern CPU ``does'' arithmetic.*
    - They interpret table-driven models of finite integer arithmetic, with arithmetic operations are defined by tables.
    - They can easily use similar tables for (72, 64)-arithmetic. *(experiment)*
      - *This is related to homomorphic encryption.*
    - We expect that these processors do not need to be as radiation hardened. *(experiment)*
- New idea (maybe) - complex ECC codes
  - *Think of the memory as a classical communication channel for moving bits from one time to a later time.*
  - *Use some of the well-known concatenated codes. (experiment)*
    - Consider the same possibility for processors. *(experiment)*

Software Approach

- Assign one task per processor.
- Duty cycles for tasks mean duty cycles for processors.
  - *Each task has a relatively well-defined timing need*

    (e.g., 2400Hz, 1200, 400, 300, 100, 50, 20, 1).
- Some classes of tasks:
  - ***Navigation:*** *compute position, velocity, acceleration from sensors;*
  - ***Attitude:*** *compute actual and desired attitude from sensors and mission plan;*
  - ***Guidance:*** *compute steering coefficients for desired trajectory and attitude;*
  - ***Power-phase autopilot:*** *roll, pitch, yaw control during main engine firing via engine gimbals;*
  - ***Coast-phase autopilot:*** *roll, pitch, yaw control during coast via attitude control jets;*
  - ***Propellant management:*** *maintain proper pressures and ratio of liquids;*
  - *Many more.*
- Tasks assigned to multiple processors for computational redundancy.
  - *Each task reads recent enough data for their computations.*
  - *Careful separation within each task of inputs from computations.*
- Task / processor communication is via DMA writes or multicast messages *(experiment)*

Software Implications

- Duty cycle management
  - *Clock synchronization to a definable precision, not necessarily the best possible precision. (experiment)*
- Task assignment
  - *Each task instance also has a priority among the instances of that task (analogous to identifying master and backups).*
- Interfaces / communication data volume
  - *Individual or block transfers (experiment)*
- Filter equations with unequal measurement data arrival intervals.
  - *Well-known mathematics for distributed filters.*
- Fault management
  - *Status writes to a monitor class of tasks;*
  - *Active failover of tasks (including monitor);*
  - *Re-assignment after ``offline'' processor reset, (experiment) or retirement of processors with too many errors.*

Software Issues

- Multicast communication *(experiment)*
  - *Protocols to manage: accepting, interfering, blocking, timing*
- Dependencies defined by shared variables
  - *producers and consumers, volume, timing (experiment)*
- Data discard policies *(experiment)*
- Interference among multiple writes *(experiment)*
  - *Collecting and computing consensus of opinion (experiment)*
  - *Byzantine generals problem*
- Detecting communication anomalies *(experiment)*
  - *Different kinds of problems: silence, lies and spoofing, blathering*
  - *Different detection methods, different responses (experiment)*
- Effects of occasional delays and dropouts
  - *Our hypothesis is that these can be made to be minor (experiment)*
- Task assignment, re-assignment, and interruption
  - *Monitor protocols (experiment)*
- Refinement and reduction of uncertainties *(experiment)*
- Does using more instances of faster task cycles help? *(experiment)*
- Coordination protocols for distributed computing *(experiment)*

Mathematical Implications and Speculations

- Dynamics: coupled state equations with occasional missing / old data
  - *Compute the delay / absence threshold below which results are essentially the same (experiment)*
  - *Map delay / absence level to resulting error covariance (experiment)*
- Filter equations with unequal data arrival intervals
  - *Modify state and covariance update equations for Kalman filter (experiment)*
  - *Other (usually simpler) filters (experiment)*
- Asynchronous interleaved filters
  - *Data is distributed to multiple parallel filters: (experiment)*
    - Alternately, with or without overlap, other distribution patterns. (experiment)
  - *Comparison of results (experiment)*
- Multistage filters and successively refined measurements
  - *Result of one filter used as a cleaned up measurement for the next (experiment)*

Conclusions and Prospects

- This approach / issue is easy to study.
  - *It has the potential to simplify flight software.*
- There is a tremendous amount of experimentation to do:
  - *In hardware;*
  - *In software;*
  - *In mathematics.*

- There is some good news.
- The software system design can start with many mostly known decisions:
  - *The set of tasks is known (we have added a set of monitor tasks).*
  - *Their respective duty cycle requirements or expectations are known.*
  - *Their respective data and computation requirements are known.*
  - *Communication dependencies are known.*