• Based on Encyclopedia of Parallel Computing: Entry on Parallel Debugging by …

• Debugging vs testing
  o Steps in the program development and lifecycle.
  o Testing is used to verify the reliability and functionality of software.
  o Debugging is used to track the reasons why software fails.

• Definition: Bug and debugging
  o A bug is the commonly used term for an error with originally unknown location and reason.
  o Debugging is the process of locating, analyzing and correcting suspected errors.

• Approaches
  o Cyclic debugging
    ▪ Loop through
      □ Set breakpoint
      □ Execute program
      □ Inspect state
    ▪ Until the bug is detected. Then correct the bug.
    ▪ It requires that the behavior is reproducible.
  o Reverse debugging
    ▪ Recorded information from the execution of the program enables to execute the program in reverse to a limited extend.

• Differences between sequential and parallel debugging
  o Increased complexity: due to multiple processes and threads.
  o Amount of debugging data: Computational speed due to parallelism leads to more data.
  o New parallel bugs: deadlocks, race conditions, nondeterminism
  o Scalability: Increased number or processes/threads requires scalable tools and approaches for debugging.

• Nondeterminism in parallel programs
  o Limitation for cyclic debugging but also for testing.
  o Due to: random number generation, race conditions in combination with OS and network jitter.
  o Elimination through instant replay.
    ▪ Record phase: collecting ordering information for critical events in an initial program run.
    ▪ Replay: recorded information is used to replay the previously observed behavior.

• Breakpointing in parallel programs
  o Definition: Breakpoint
    ▪ A breakpoint is a controlled way to force a program to stop its execution.
    ▪ Breakpointing may occur on
      □ software interrupt calls
      □ calls to a program subfunction
      □ selected points within the program
  o General types of breakpoints
    ▪ Instruction breakpoint: Interrupts the execution before a
certain instruction is executed.

- **Data breakpoints or watch points**: interrupts the execution when a memory address is accessed or a specified value is assigned.
- **Conditional breakpoint**: The breakpoint is combined with a condition. Only if the expression is true the application is stopped. A special condition is the so-called ignore count.

  - Breakpoint types for parallel programs
    - **Local breakpoint**: applied to a single process. Identical to the sequential case.
    - **Message breakpoint**: Stops all processes involved in single communication event, either p2p or collective.
    - **Global breakpoint**: involves multiple processes.
      - **Single global breakpoint**: single breakpoint owner process. Its state determines to break. The other processes are stopped immediately in undefined state.
      - **Global breakpoint set**: Consists of multiple local breakpoints. When the global breakpoint is reached, the involved processes are stopped at defined states. The processes will respond to resume only after all in the set reached the global breakpoint (barrier breakpoint in Totalview9).

- **Tools for debugging**
  - Runtime checking tools
  - Program visualization tools
  - Traditional debuggers

- **Runtime checking tools**
  - MPI Checker: Automated identification of errors such as misuse of MPI routines or race conditions.
    - Use when porting MPI applications to new systems or introducing new MPI operations.
    - **MUST** RWTH Aachen (Predecessor Marmot, HLRS), Umpire (LLNL). MUST currently provides correctness checks for the following classes of errors:
      - Constants and integer values
      - Communicator usage
      - Datatype usage
      - Group usage
      - Operation usage
      - Request usage
      - Leak checks (MPI resources not freed before calling MPI Finalize)
      - Type mis-matches
      - Overlapping buffers passed to MPI
      - Deadlocks resulting from MPI calls
  - Memory checker detecting standard errors such as memory leaks as well as race conditions due to unprotected shared data access.
    - **Intel Inspector XE**
      - Memory Errors
        - Memory leaks
        - Memory corruption
        - Allocation / de-allocation API mismatches
        - Inconsistent memory API usage
- Illegal memory access
- Uninitialized memory read
- Threading Errors
  - Data races
  - Deadlocks

- Visualization tools
  - VAMPIR (TU Dresden): trace visualization.
    - Used to detect wrong communication patterns, analyze deadlocks etc.

- Traditional debuggers
  - Most standard debuggers support thread level parallelism, e.g., setting breakpoints for threads or analyzing shared and private variables.
  - Few debuggers support process level parallelism. Allinea DDT and Rogue Wave Totalview.
    - Support for applying operations to sets of processes and threads.
    - Comparison means for data structures across process groups.
    - Array inspection and visualization tools for distributed multidimensional arrays.
  - Important aspect: scalability of those debuggers
    - Challenges for launching the tools and communication in the tools, but also in
    - STAT - The Stack Trace Analysis Tool (University of Wisconsin)
    - Based on MRNet infrastructure.