Checkpoint/Recovery

18-849b Dependable Embedded Systems

John DeVale

February 4, 1999

Required Reading: **Application-Transparent Checkpointing in Mach 3.0/UX - Russinovich and Segall**

Best Tutorial: **Libckpt: Transparent Checkpointing under Unix**
Usenix Winter 1995 Technical Conference

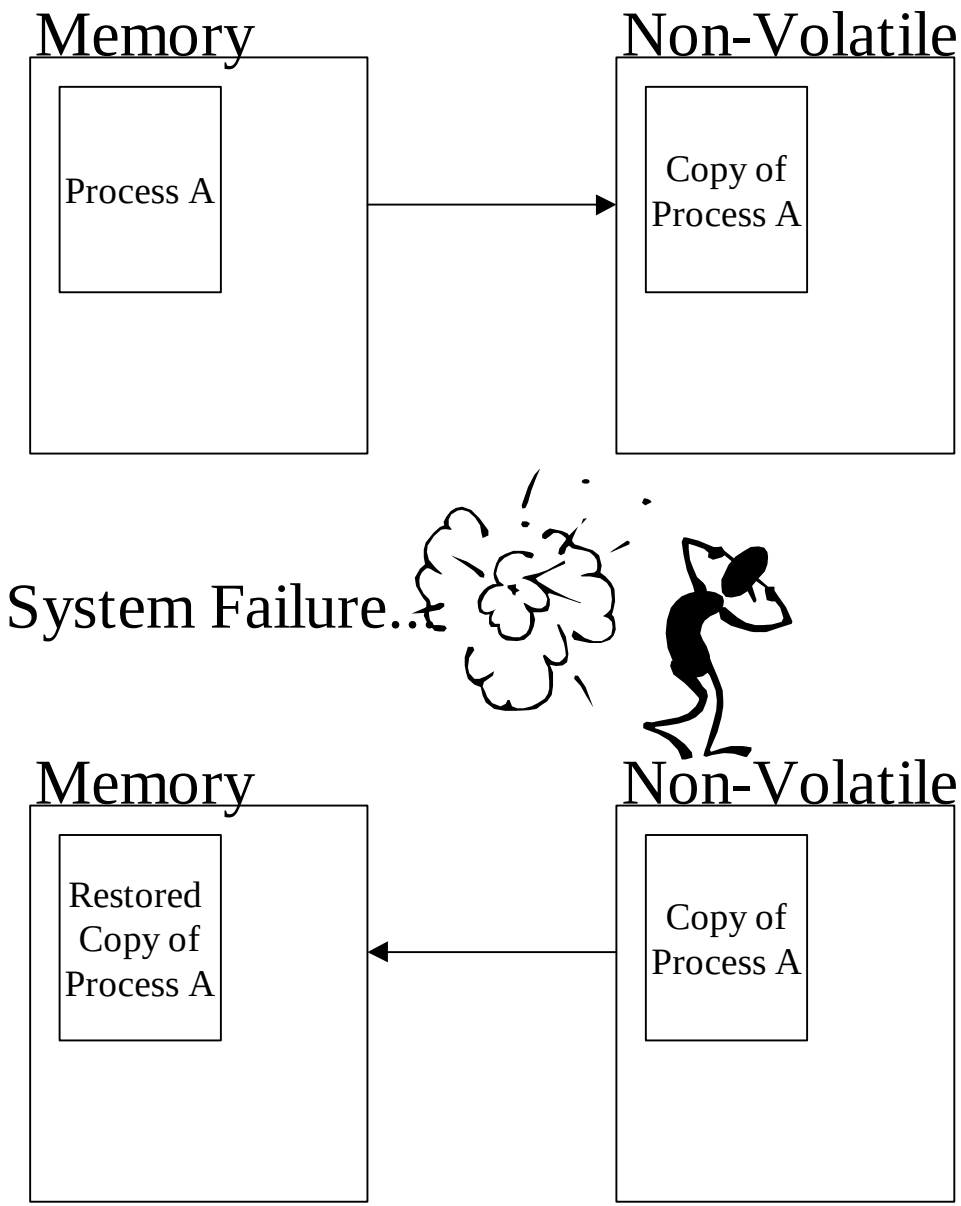
Authoritative Books: **Software Fault Tolerance, Michael R. Lyu (ed)**

Carnegie Mellon

Overview: Checkpointing - Recovery

- **Introduction**
  - Method of creating fault tolerant software systems
- **Key concepts**
  - Periodically saves process/system state
  - In the event of a fault, state is restored via a rollback
  - Scales to distributed/parallelized applications
- **Tools / techniques / metrics**
  - Stratus VOS and Tandem NonStop Kernel
  - libFt , libckpt
- **Relationship to other topics**
  - Fault Tolerant Computing technique
- **Conclusions & future work**

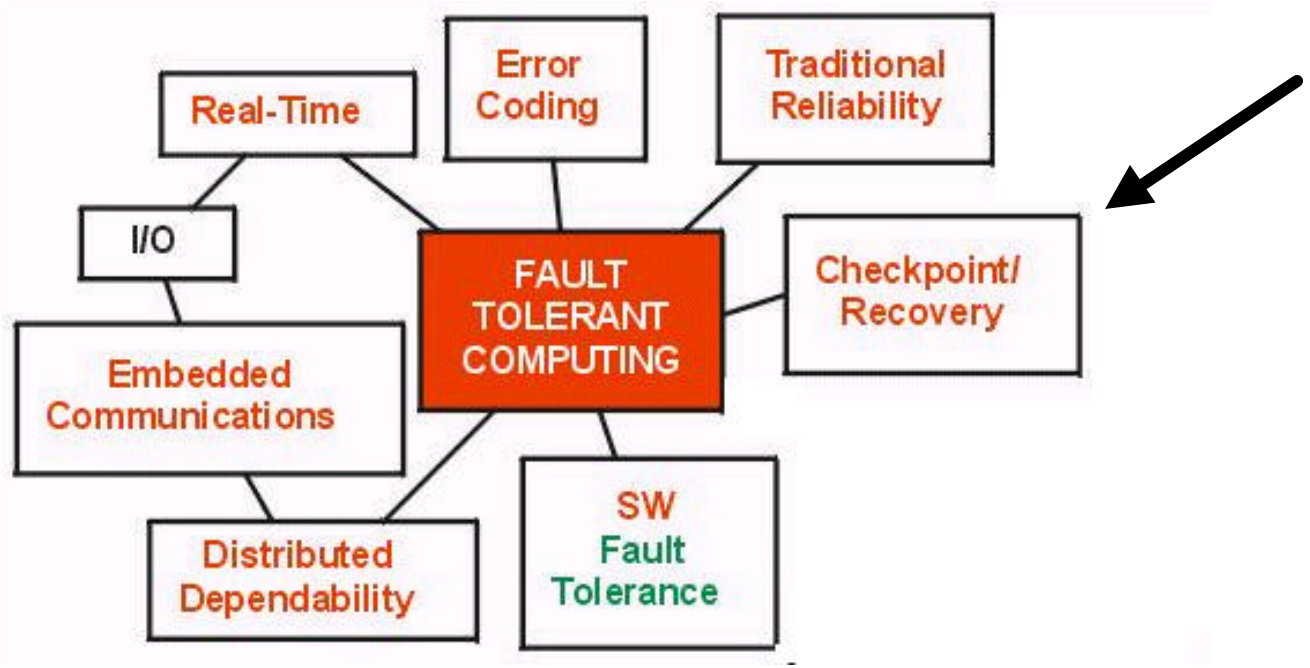
Checkpoint - Recovery… The basic picture



Checkpoint mechanism copies the state of process A into non-volatile storage

Restore mechanism copies the last known checkpointed state of process A back into memory and continues processing. This mechanism is especially useful for application which may run for long periods of time before reaching a solution.

Where we are


Real-Time
Error Coding
Traditional Reliability
I/O
FAULT TOLERANT COMPUTING
Checkpoint/ Recovery
Embedded Communications
Distributed Dependability
SW Fault Tolerance

Description of Topic

- ***Checkpoint-Recovery* gives an application or system the ability to save its state, and tolerate failures by enabling a failed executive to recover to an earlier safe state.**

- **Key ideas**
  - Saves executive state
  - Provides recovery mechanism in the presence of a fault
  - Can allow tolerance of any non-apocalyptic failure
  - Provides mechanism for process migration in distributed systems for fault tolerance reasons or load balancing

Saves Executive State

- **When a checkpoint is executed, a snapshot of all program state is saved into some non-volatile, machine accessible medium.**

  - Time
  - Space
  - Memory Exclusion - *new idea*
    - Ref:Memory Exclusion: *Optimizing the Performance of Checkpointing Systems* James Plank, submitted for publication, SP&E
    - Allows the checkpointing mechanism to be told, and/or dynamically determine what memory structures are an important part of program state, and only save those structures.
    - Saves time *AND* space

Provides Recovery Mechanism

- **Once a fault has occurred, the recovery mechanism restores the program to the last checkpointed state**

  - Current automatic Unix based tools wait for the process to abort, and restore it after abort.
    - Time constraint may not allow for this length of recovery
    - In the presence of a software design fault, rollback mechanism needs more complexity to allow rollback to a previous state, yet retain knowledge of faulted path of execution.

  - Stratus, Tandem seem to handle this, but details are sketchy

Failure Tolerance

- **Faults can be tolerated, even those which may physically destroy the processing site**

  - Geographically distant sites with a synchronized distributed systems can perform coordinated checkpoints and process migration.

  - Transient faults and glitches tolerated as a matter of course through the normal checkpoint-recovery system

Tools / Techniques

- **libFT - AT&T research labs**
  - Provides checkpoint recovery and watchdog demons
  - http://www.research.att.com/sw/tools/reuse/packages/ft.html
- **libCKPT - University of Tenn. Knoxville**
  - Provides incremental checkpoint recovery library, with memory exclusion
  - http://www.cs.utk.edu/~plank/plank/www/libckpt.html
- **PMCKPT**
  - The Poor man's checkpoint
  - http://warp.dcs.st-andrews.ac.uk/warp/systems/checkpoint/source.html
- **CONDOR**
  - Process migration for load balancing
  - http://www.cs.wisc.edu/condor
- **General Links**
  - http://warp.dcs.st-andrews.ac.uk/warp/systems/checkpoint/

Metrics

- **Key Metrics in Checkpoint - Recovery**
  - Snapshot Time
    - How long it takes to identify and copy (to intermediate storage) all required program state
  - Commit Time
    - How long it takes to copy snapshot into non-volatile storage
  - Recovery Time
    - How long it takes to restore state to a failed process

- **Dependant on state size and system performance**

Relationship To Other Topic Areas

- **Fault Tolerant Computing**
  - Checkpoint - Rollback is a technique which can be used to build fault tolerance into a computing system
  - In its current form it very capably saves process state and can create a new process and restore old state to it in the case of a process failure

- **SW fault tolerance**
  - Related to SW fault tolerance by sharing a common goal
  - Scope of the solutions are on a much different scale
    - SW fault tolerance focuses more on making software not crash
    - Traditional checkpointing focuses on recovering from the crash in a graceful manner while preserving computational state and critical data.

Conclusions & Future Work

- **Checkpoint-Recovery provides**
  - Ability to save and restore state for critical applications
  - Useful for single computer systems and large distributed or parallel systems
  - Can incur large time penalties during checkpointing

- **Future Work**
  - Design for Checkpoint-Recovery
    - Design critical systems to have as small a critical state as possible
      - » Breakdown task into smaller subsystems which can be checkpointed separately
      - » Self recovering state
  - Task restart may not be possible in small RT/Embedded systems
    - Support at the OS level to allow micro checkpoints and rollbacks at a task level

Application-Transparent Checkpointing in Mach

- **Paper presents methodology for checkpoint-recovery**
- **Performance varied with memory footprint**
  - Typically <5 sec checkpoint cost (first) less for subsequent
  - Larger commit delays - 10 to 30 sec of degraded performance
  - Recovery times 5 to 10 sec for reasonably sized applications
- **Major Contributions**
  - Provides roadmap on how one might build in transparent checkpointing
  - Can checkpoint and restore entire system state in X
- **Limitations**
  - Time costly
  - Requires custom pager in OS
  - Does not address memory exclusion (trade-off for transparency)