Efficient Software-only Checkpointing Support for Debugging

UofT Connection 2009

Chuck. Zhao

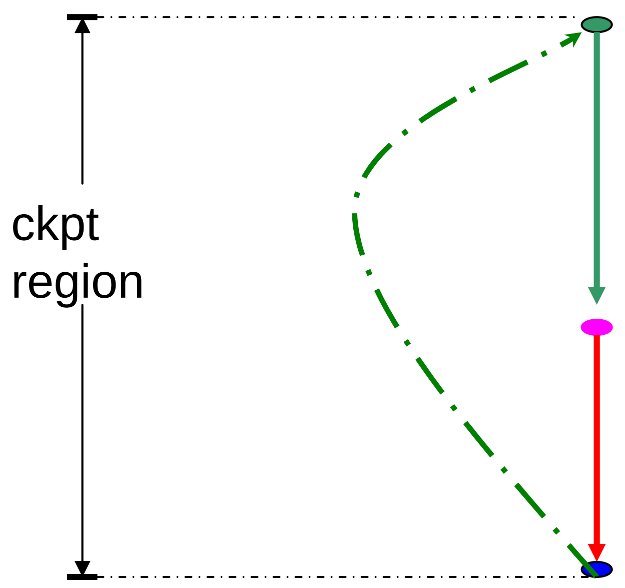
May.14 2009

Checkpointing Support for Debugging

- Efficient Software Checkpointing Framework
  - checkpoint and rollback within any given program region
    - cover arbitrarily large code area
  - software only
  - compiler optimizations for overhead reduction

- Existing Solutions for Ckpt-enabled Debugging
  - hardware-based schemes
  - cover limited program region (limited checkpoint buffer)
  - no program analysis or optimizations

Checkpointing Support for Debugging



T: safe point, literally earlier than P, the program can reach through checkpoint recovery

P: root cause of a bug

Q: place where the bug manifests

(a user or programmer notices the bug at this point)

Programmer can progressively increase the ckpt region’s granularity until the root cause point (P) is covered

Checkpointing Support for Debugging

- Bug Identifying Process
  - bug locations are known to us (from BugBench doc)
  - can trial and error with the buggy input

- Enable Software Ckpt
  - backup: over normal program code region
    - start ckpt: an estimated “good” program point
    - stop ckpt: immediately before/after the bug manifests
  - recovery
    - programmer controlled in debugging mode

Checkpointing Support for Debugging by Example

```
/* buggy code: storage.c:176, bc-1.06, BugBench suite */
for (; indx < v_count; indx++){
    arrays[indx] = NULL;
}
```

original code with buffer overflow bug

```
/* buggy code: storage.c:176, bc-1.06 */
start_ckpt();
for (; indx < v_count; indx++){
    backup_memory(&arrays[indx], sizeof(arrays[indx]));
    arrays[indx] = NULL;
}
stop_ckpt();
```

buggy code checkpointed

Checkpointing Support for Debugging

- Benchmarks
  - BugBench
    - a total of 17 C programs that have known bugs
    - around 10 are buffer-overflow related memory bugs

- Evaluations
  - SUIF Compiler Framework
    - leverage on our existing checkpointing framework
  - Functional
    - program rewinding
    - ckpt locations and granularity
      - buffer size, # of instructions, # of meta entries, …
  - Performance
    - performance difference with ckpt enabled (on non-failing inputs)
    - performance difference with ckpt optimizations

Checkpointing Support for Debugging

- Debugger with ckpt will not find the bug automatically
    - still the programmer's job to find the bug
- Debugger with ckpt will provide additional assistance in finding the bug
    - reverse code to start ckpt location without terminate execution
- Our proposed work will deliver both functionality and performance

take away points

Questions?