Summary: Light-Weight Contexts

Keywords:

- Security
- Memory isolation
- Snapshots & Rollback
- Reference monitor
- Resource sharing
- Context switching

Problem Statement:

- provide data isolation, privilege separation and multiple execution states.

Problems arising using fork for above tasks:

- The problem with using different processes was the involvement of the kernel scheduler in processes' switching and communication. Actual hardware imposed cost of isolation and privilege separation is much smaller.

How are lwCs different from threads:

- LWCs are orthogonal to threads that may execute within a process.
- Threads provide separate units of execution, but share the same address space.
- LWCs provide different execution states, separate virtual address space, set of page mappings, file descriptor bindings and credentials.

Design:

- A process can have multiple lwCs.
- Each process starts with 1 root lwC.
- Within process, thread executes within 1 lwC at a time but can switch between lwCs.
- Each lwC stores execution state of all threads which enter it.
- LWCs provide methods like lwCreate, lwSwitch for creation and switching among lwCs which are much faster than creation of processes and switching through kernel scheduler.
- lwSwitch provides coroutine semantics.
- Parent lwC may modify visibility of its resources via resource-spec argument
- child lwC receives a copy-on-write snapshot of all its parent's resources by default
- lwC may dynamically map(overlay) resources from another lwC.
- lwRestrict, restricts the set of resources that may be overlayed or accessed by any context that holds the lwC descriptor

API Summary:

- Create lwC :
  - new, caller, args <- lwCreate(spec, options)
- Switch to lwC :
  - caller, args <- lwSwitch(target, args)
- Resource Access
  - status <- lwRestrict(lwc, spec)
  - status <- lwOverlay(lwc, spec)
  - status <- lwSyscall(target, mask, syscall, args)

Uses:

- LWCs can be used to efficiently implement
  - Snapshots
  - Rollback
  - Sensitive data isolation
  - Protected reference monitors

Performance:

- On the benchmarks on which performance was tested, lwC switch takes less than half the time of a process or kernel thread switch.
- Cost of maintaining a separate lwC is linearly dependent on the number of unique pages it creates.
- Apache provides security when it is configured to fork a new process for each connection. LwC provides slightly better performance than this configuration of apache, providing same level of security.
- For both HTTP and HTTPS, with isolation and reference monitoring, lwC-augmented nginx performs comparably to native nginx without these features.
- LwC snapshot is able to provide significant performance benefit to highly optimized end-to-end applications such as web frameworks, while adding isolation between user requests. The speedup in FCGI is almost double.
- With lwCs we were able to isolate SSL private keys without adding any cost.

Check your understanding:

1. How is security an issue in the case of Apache preforked processes configuration?
2. Why do we observe speedups on using lwCs in PHP FCGI?
3. Difference between process-driven, thread-driven, event-driven servers.
4. What is reference monitoring? How lwCs can provide in process reference monitors?
5. On increasing the number of requests, performance of both lwCs and basic-nginx server decreases. Why?