Threads & Concurrency

ICS lec 4

- Modern systems are multithreaded. dont have Single threaded. (Multiple tasks can be done at a time).

- each thread will have own pc, stack and register.

- in Server Server will create new threads upon user requests.

Benefits of muttithreading.

- responsiveness - Several task are executed parraral.
- Resourse Sharing - easy to communicat with ip processes.
- Economy
- Scalability.

Chdlenges of multicore programming.

- Deviding activities.
- Balance.
- Data Splitting.
- Data dependency.
- Testing & depuging.

Parallelism - Can perfor than one task at a time.
Concurrency - Single core each proces going to get a time to execute.

Multicore programming

- have 2 types of parallesim in multicore programming.

01) Data parallelism - Sub set of data given to Cores

02) task parallelism - distributing the data acros Cores. not a subset. Call data shared).

2 type of threads

01) User threads - managed by user application.
02) kernel threads - managed by OS.

Relationship between user and kernel threads

- Many to one
- one to one
- Many to Many

(01) Many to one

- Many user threads mapped to one kernel threads.

- if ~~other~~ one thread making using kernel other threads are blocked. (one thread at a time).

(02) one to one

- each user thread mapped to kernel thread. User thread have a kernel thread.

Date No

(03) Many to many

- many user threads mapped to many kernel threads.
- kernel threads are equal to user or less.
- ~~the~~ no of kernel thread depend on application or OS.

④ Two level model.

- Some as many to many, but one user kernel can be bounded to kernel thread.

How to create and manage threads (thread library)

- Provided API to manage and create threads.

2 ways of creating threads.

01) Entire library in the user space. (Local func) not going to System calls.

02) kernel level - Supported by OS. have System calls.

Pthreads.

- can be user level or kernel level thread.
- Show the behaviour of the thread library.

Implicit threading.

-if no of threads high managment & creations are more difficult. So it will done by Compilers & run time librarys.

5 methods to manage & creat threads

01) thread pool
02) Fork-join
03) open MP
04) Grand central Dispatch
05) Intel threading building blocks.

01) Thread pool.

- Creat no of threads and keep it. if task need a threads allocate a thread and continue.
- Can give sub set of threads from the pool.
- Can schedule tasks to threads.

02) Fork-join parallelism,

- from main thread creating multiple threads by fork() call and allocate task to threads and join together to get output.

Threading issues.

- how to use fork() and exec calls.
- how to handle signals.
- thread concellation
- thread local storage (What going to store)
- Scheduler activation.

Date No

(01) Semantic of fork() and exec().

- duplicating all threads or Calling thread.

depend on the how use fork and exec. (fork() + exec(), fork(), exec() + fork()).

(02) Signal handling.

- Signal used to notify a process to about event.
- Signal handler used to handle the Signals.

- Signal generated
  ↓
  Signal deliver to the process.
  ↓
  handle the Signal.

- 2 Ways of handling a Signal.
  (01) default
  (02) User-defined.

(01) default - every Signal has a default handler running on the kernel..

(02) user-defined - have to overrid the default handler

- if it a Single threaded no problem. Signal Will delivered thoogh using that thread..

- but multi threaded have Som problems.

- going to deliver to all threads, set of threads, assign a specific thread or it choice.

03) thread cancellation.

- terminating a thread before it has finished. the thread going to cancelled is target thread.

Ex: if multiple threads finding a db and one found it other threads will be Cancelled.

2 ways to Cancell a thread.

01) Asynchronous Cancellation
02) Deferred Cancellation.

01) Asynchronous - Cancell the thr thread immidiatly.

02) Deferred - check weather it should be cancelled or not.

04) Thread local storage. (TLS)

- allow all threads to have a own copy of data. usefull when using thread pool. also know thread specific data

- TLS unique to each thread.
- Different from local variable.

Date No

05 Scheduler activation

- M:M and two level has equal amount or less no of kernel level threads. to maintain the no of kernel level threads using an data structure between user and kernel threads (light weigh process) - LWP

- each LWP attached to kernel thread.
- OS Schedule kernel thread to run on physical Process.
- if kernel thread blocked, LWP, and user thread also going to block.
- if a request doesn't have enough thread have to whoit untill one end.

- how the communication happening between kernel and user thread called Scheduler activation.

- kernel inform user application about a event by upcalles. upcall A handled by thread library. and have a upcall handler.

- upcall has to run on the LWP.