Scala

oo, functional, compile to JVM bytecodes, high throughput or process large data set, embed Domain-specific language

everything is an expression

every value is an object

every operation is a method call

Scala 的 class instantiated using new 其中 primary constructor 最重要, 直接内嵌且 default。其他 alternative constructor 【def this(xx) = {this(xxx)}】可 chains 起来，但最终都需用 this(xxx)形式 call primary constructor

field 与 constructor arguments 不是一回事儿，fields contain data items. 它可能由几个参数构成，暴露给用户一个更有意义的整体名，如 name = firstname + lastname

method :【def funcname(arg : type) : returnType = x / {x}】
可 override
func body last expression 需 be a value
无 return type 则用 Unit 代替

val: immutable，evaluated once at init, 之后不变了
def: 每次 referenced 时都重新 evaluated 一次， 会变
lazy val: init 以后还在变，但第一次 referenced 后值被 fixed 了

Concurrency

multi task logically proceeding in time interval

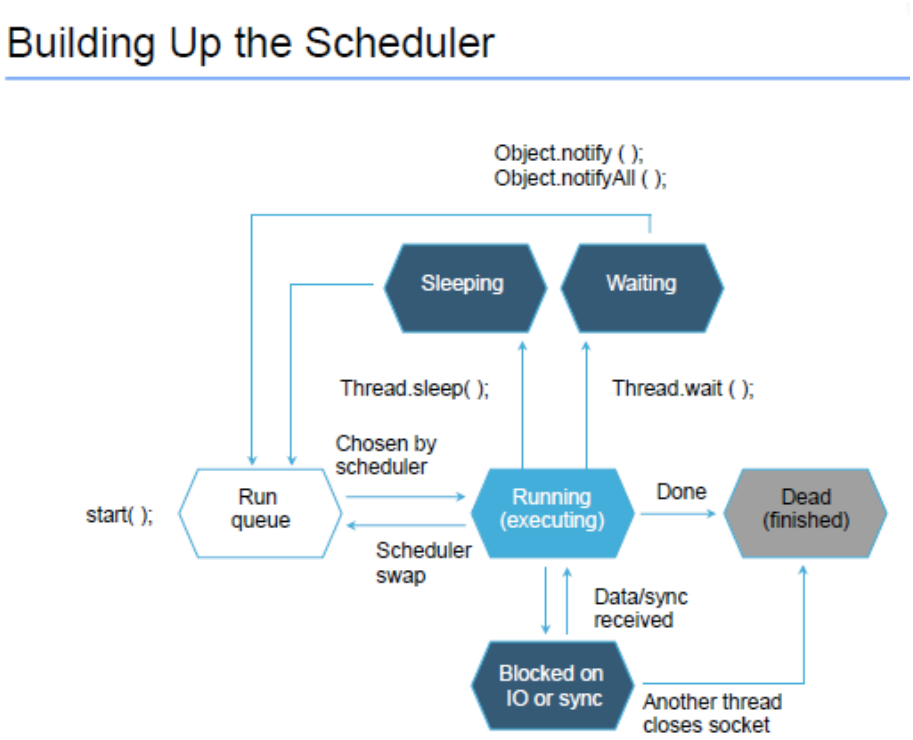
vs. Parallel

multi task execute at the same instant

Thread

sequence of instructions that execute independency

exe concurrently, multicore processor, singlecore processor, time sliced



Process

exe program 变成 process, fork() 创 new process

at least one thread executes concurrently

provide resources (address space, security context, environment variables) needed to execute a program

Thread pool

threads created ahead of time and used when tasks are submitted

Concurrent code 得

1. Synchronize operations 操作同步
   condition variable
   events
2. Protect Data 数据 lock
   race condition/deadlock
   mutex/semaphores
   lock free programming

Condition variable

each requires an associated lock to protect its use, from the lost wakeup race condition

Semaphore

an (>=0)integer variable, its value represent a number of available resources of some kind. Only 2 operations wait() (substract value)or signal() (add value). When value = 0, it blocks the calling thread until value above 0 again.

Daemon threads

if a thread has its daemon flag set, then it will be auto terminated when all other non-daemon threads have terminated.

a thread that does not prevent the JVM from exiting when the program finishes but the thread is still running.

Java

Callable -- call()	return value, 用于 asynchronous calculation, 可= lambda
vs. Runnable – run()	no return value, take no arguments, don't raise exception

multiprocessing	at least 1 thread, 多 process
vs. multithreading	多 threads 可在 same application, within a process

Java thread:

Define	implement Runnable interface -- use static method () of thread Or extend Threads – call () directly 因是 subclass of thread
Execute	public void run() {}

notify()	wake up only one thread
notifyall()	wake up all threads that blocked in wait() on the same object

Task submit() to ExecutorService, returns Future<T> to reperesent result

Actor

a self-contained processing unit

is composed of(optionally) state, behavior and a mailbox that allows it to receive messages.

communicate with each other by sending asynchronous messages and reacting to them.

share and communicate using immutable message 分享 const 信息

keep any mutable state internally, but nothing exposed 保留可变

Thread Scheduling

Scheduler 根据 thread associated priority 来 determine which thread can access CPU at any time.

A newly created thread 的 priority 来自于 created 的 thread. 如 setPriority() 可用于设置 priority

Thread.yield() hints scheduler that if there is another runnable thread at the same priority, then we are equal priority, if no runnable thread then no-op, scheduler may ignore the call.

join() 等 until thread object died

interrupt() 用于 cancel thread, 中断一个 waiting thread(如 thread.sleep()) and raise an InterruptedException, Or if not waiting, set its interrupt bit. (clear interrupted status flag back to false) 需要 reset, 要不其实没终止程序运行

blocking call

sleep() ,wait(), join() 均可 interrupt thread

Synchronize (lock for share data)

the lock is per object

only one synchronized method of a given object may be executed by a thread at a time.

volatile

告诉 compiler 不优化，该 thread 可 shared 可变

Lock	one mutex lock per object
ReentrantLock	for recursion, same thread call lock 多久, fair lock
ReadWriteLock	可多 read 1 次 write，better throughout, fair
Atomic Variable	safe, atomic way to update, fast hardware support
positive lock	全 lock,别的都不能动我
optimist lock	无 lock, check timestamp, 当 timestamp 有变化则重新 read

seed.compareAndSet(origSeed, nextSeed) //set the seed to nextSeed only if the current value is origSeed. 无 thread blocking

Condition

wait() notify() notifyall() = await() signal() signalall()

new objects now become possible to manage several condition queues in association with a single object or lock.

Latches

open/close , all thread 还未 ready 不可 run

all thread 都 ready 后才 open latch, 只有 open latch 以后,才可 all thread start.

value 变为 0, countDown

Cyclic Barrier reusable synchronization barrier

Threads can't proceed until all have reached the point/some state.

在 exe 其间，得 wait,然后 continue

await

Phaser reusable synchronization barrier

同上，可 restart

Lock free

blockqueue q 空 q 满会 block

non-blocking queue 更大 throughout

tail – head = 0 则 empty

tail – head >= arraylength 则 full

element added at tail index

element removed at head index

Scala

同 java, Runnable interfece encapsulating works for thread to do

Define extends Runnable

Exe override def run = {}

Asynchronous Programing

在不等的期间，the calling thread 用 future 先来 represent result, 等 obtain result 后， 通过 some form of callback mechanism 把 result 回传给 calling thread. (这个 thread 叫 ExecutionContext)

Future

a construct/placeholder represent a value that will be available at some time in the future.

read only value(another context 写它的 value)

当 future complete 后，返回 Try[T] type object 传给 function， 可能 success 可能 failure，所以 we define callback onComplete function, take Try[T] object (T is encapsulated in the Future), examine both success and failure case.

到 future complete 为止，block thread 的方法有 await.result(future, duration) 和 await.ready(future, duration). Result() return value of future, Read() return future object.

Promise

represent a placeholder, a value of particular type will be written at some point in the future.

可 watch many future objects, as soon as a value is written to it, all these futures will complete and success. 以 pname success value 的格式来 complete a promise, 再要 write a value 会有 IllegalStateException.

only be written to once.

Actor

Message Actor messages can be of any type

Define class xxxActor extends Actor with ActorLogging

override def receive = {} //how actor behaves in response to messages

no return type, : Unit

! send message to an actor

? send message to another actor and expect a reply with some data in it, a Future act as place holder for the reply, 且必须 map to correct Future type， 如 (xx ? xxx).mapTo[Int], then after we can handle.

relpy order 不一

Create&Init

val system = ActorSystem(“XXX”)

system.actorOf(Props[…]) //Props[] actor configuration

system.terminate()

Blocking on each request until response arrives need to import scala.concurrent.Await

val a : Int = Await.result(xx? xx()).mapTp[Int], Duration(1, seconds))

C++

std::thread t{xxx}

xxx 可以是 function, lambda [v][&v][move(v)], function objects, 如下：

```
stuct f{
    void operator()() const { … }
};
```

unique_ptr 不可 copy, 需 pass by reference 或 move

make_unique<xxx>是把 xxx 用 unique_ptr wrap 起来了

join() parrent 必须等 child 结束，才可 terminate

detach()　把当前 thread turn into daemon thread, runtime system 会 safely terminate it when the main terminates.

Pass arguments to thread

by reference　&, std::ref()可行但会使得 data shared by other threads

by value　X

use move semantics　&&, std::move()

mutex

guarantees that only one thread will hold the lock at a time

mechanism for synchronizing access to date from multithreads

有 2 种可选：the standard mutex ， the recursive mutex

lock_guard<mutex> guard(mutex)

RAII, encapsulate the acquisition of a lock, construct and destruct it

unique_lock<mutex>

during the block, allow release lock

std::promise

store a value for asynchronous retrieval

std::future

waits for a value that is set asynchronously

std::async

launch a function in a new thread and retrieve its return value asynchronously.

Deadlock avoidance

用 std::defer_lock 来 deferred locking,

然后 std::lock(m1, m2) to acquire all locks together.

Condition Variables

a point of synchronization of multiple thread

threads wait on CV and notify on CV

all threads wait until the CV is notified

cv.wait(lock), cv.notify_one()与 cv.notify_all()

if the test for the event condition and the wait operation are not performed in one atomic step, the potential race condition would occur. So, a condition variables is always associate with a lock, guaranteeing that the test for the event and the wait operation can't be separated in terms of another thread waning to notify on the variable.

std::atomic

lock-free

std::atomic_flag (bool, init=false)

.test_and_set() fip

.clear() =false

aa.compare_exchange(expected, desirable)

double-checked locking (singleton 要 atomic instance variavle, istance() {}里 double check 确保 this static initialisation 是 thread safe)

Monitor’s methods manage the lock for a single object of reference type.

task the task class represents an asynchronous operation

async mark a method that execute code asynchronously, use await to execute, can not have ref, out parameters, can call method that have them

async method will execute synchronously on the calling thread until the first call to await, at which point it becomes asynchronous.

await introduces a suspension point, is used to execute asynchronous code within a asyn method

Barrier user defined synchronization primitive, each threads exe until they reach the barrier