关于并行程序的安全和性能问题

Pascal 的并行程序主要是两套工具流来开发

FPC+Lazarus+LCL，也可以是 FPC+CodeTyphon+LCL，根据喜好而定。
Delphi，内置提供了 TParallel，并且提供了匿名函数作为辅助

现在我们分别来看一下，这些工具流的并行程序做法

FPC+Lazarus+LCL

目前 freepascal 官方给出的解决方案需要安装一个依赖包:multithreadprocslaz
然后引用 MTProcs 单元
它的程序范例如下

```
program Test;

{$mode objfpc}{$H+}

uses
  {$IFDEF UNIX}
  cthreads, cmem,
  {$ENDIF}
  MTProcs;

// a simple parallel procedure
procedure DoSomethingParallel(Index: PtrInt; Data: Pointer; Item: TMultiThreadProcItem);
var
  i: Integer;
begin
  writeln(Index);
  for i:=1 to Index*1000000 do ; // do some work
end;

begin
  ProcThreadPool.DoParallel(@DoSomethingParallel,1,5,nil); // address, startindex, endindex, optional data
end.
```

安全问题：并行套并行，会出现卡死情况。如果做算法，这种问题让人很头疼，你得控制并行开辟的场景，至少要避免并行里面套并行。
性能问题：在并行程序中如果发生内存的高频率 alloc,realloc，会发生卡顿，导致多核跑不满，失去并行的意义
库问题：因为 fpc 本身支持 nested 回调申明，而 MTProcs 并没有直接使用 nested 回调接口，都是 method 类回调，DoParallel 需要我们给出一个指针，它强制换成 nested 回调，为什么要这样做？我也不知道！结论：蛋疼！

Delphi

Delphi 官方给出的并行方案需要引用 system.threading 单元
由于匿名函数的支持，delphi 的并行程序非常人性化
它的程序范例如下

Using TParallel.For from the Parallel Programming Library

Go Up to Using the Parallel Programming Library

The **Parallel Programming Library (PPL)** includes a loop function, TParallel.For, that allows to run its loops in parallel.

Within parallel loops, you must transform your operations into thread-safe operations. You can use members of the System.SyncObjs unit, such as TInterlocked methods.

Converting a For Loop into a `TParallel.For` Loop

Delphi:

For Loop	TParallel.For Loop

`for I := 1 to 10 do`
`begin`
`  // …`
`end;`
	`TParallel.For(1, 10, procedure(I: Integer)`
	`  begin`
	`  // …`
	`  end);`

C++:

If you are using a Clang-enhanced C++ compiler, you can use lambda expressions:

For Loop	TParallel.For Loop

`for (int i = 1; i <= 10; i++) {`
`  // …`
`}`
	`TParallel::For(0, 10, TProc1<int>([](int I){`
	`  // …`
	`}));`

You can alternatively use a functor, which works with both previous-generation compilers and Clang-enhanced compilers:

For Loop	TParallel.For Loop

`for (int i = 1; i <= 10; i++) {`
`  // (Loop logic)`
`}`
	`class TMyFunctor : public TCppInterfacedObject<TProc__1<int> > {`
	`public:`
	`  void __fastcall Invoke(int I) {`
	`  // (Loop logic)`
	`  }`
	`};`
	
	`// …`
	
	`TParallel::For(1, 10, System::DelphiInterface<TProc__1<int> >(new TMyFunctor()));`

安全问题：和 FPC 一样，并行套并行，会出现卡死情况。因为 thread pool 原因。
性能问题：在并行程序中如果发生内存的高频率 alloc,realloc，会发生卡顿，导致多核跑不满，失去并行的意义

优化并行程序性能

由于 delphi 与 FPC+Lazarus 不提供编译级的内存预分配，导致了分支程序在计算中非常容易发生内存的 alloc,realloc，该机制会出现锁操作，这是导致 cpu 跑不满的罪魁祸首。解决该问题最有效的方法是预申明好内存后，再去做并行计算。

其次是并行计算目标，我们以 fill 为例

```
For i:=1 to 10000*10000 do
   Buff[i]:=0;
```

对比

```
Tparallel.for(1,10000*10000,procedure(pass:Int)
Begin
  Buff[i]:=0;
End);
```

在这种情况下，for 要比 tparallel 快上 cpu 频率/内存频率的值
我们必须知道一点常识：内存的频率非常低，通过多通道来提速，才能达到 2999,3000,4000 这种高频率，真实的单通道内存频率在超频情况一般都是 2000 以下，而 cpu 动辄都是 3000,5000 这种睿频，理论上 cpu 频率会比内存快上 2 倍以上。如果是 cpu 是多核工作，速度比内存会拉的更远。
我们编写程序时，除非有很多 for 分支，且含有大量 cpu 的计算，诸如向量，像素，否则，直接串行即可。

并行程序的安全和稳定问题

躺过很多并行坑，计算循环峰值经常跑到 400 并发线程，delphi 的 Tparallel 和 fpc 的 DoParallel 会等并行结束，线程池满，以及并行中等另一个并行中结束，发生这类情况时非常容易出现卡死情况。（最开始我以为我的程序问题，检查半天！）

另外一个是，并行程序在 ide 很不好调试，delphi 甚至在匿名函数中，出现丢失 debug symbol 的 bug！！
处于以上考虑，我在内核库 CoreComputeThread 中提供了一种安全措施，它不会在大规模并行循环中发生卡死，发生线程池满情况，它会自动将并行调度成串行。当我们以 debug 方式运行程序，它会自动关闭并行，方便调试。而非 ide 模式，则是并行化。

By.qq600585
2020-2