fibonacci parallel processing

main source

```haskell
module Main where
import qualified Parallel as P
main = P.parallel_main
```

haskell source

Paralle.hs

```haskell
module Parallel where
import Control.Exception
import Control.Parallel
import Control.Parallel.Strategies
import Data.Time.Clock
import System.Environment
import Text.Printf

fib2 :: (Eq a1, Num a2, Num a1) => a1 -> [a2]
fib2 1 = 1 : []
fib2 2 = 1 : 1 : []
fib2 n = sum (take 2 f) : f
  where f = fib2 (n - 1)

fib3 :: (Eq a1, Num a, Num a1) => a1 -> a
fib3 n = fib2 n !! 0

-- slow fibonacci
fib :: Integer -> Integer
fib 0 = 1
fib 1 = 1
fib n = fib (n - 1) + fib (n - 2)

printNumber :: Show a => [a] -> IO ()
printNumber [] = return ()
printNumber (x : xs)
  = do putStrLn $ show x
       printNumber xs

parallel_main :: IO ()
parallel_main
  = do let test = test_fast
       t0 <- getCurrentTime
       r <- evaluate $ runEval test
```

```
        printTimeSince t0
        printNumber r
        printTimeSince t0

test_fast
  = do x <- rpar $ fib3 1000
       y <- rpar $ fib3 2000
       z <- rpar $ fib3 3000
       z1 <- rpar $ fib3 4000
       return [x, y, z, z1]

printTimeSince :: UTCTime -> IO ()
printTimeSince t0
  = do t1 <- getCurrentTime
       let v = (realToFrac $ diffUTCTime t1 t0) :: Double
       printf "time: %.9fs seconds\n" v
```

result

```
code>stack run
time: 0.000001200s seconds
434665576869374564356885276750406258025646605173717804024817290895365554179490518904038798400792551692959225930803226347752096896232398733224711616429964409065331879382989696499285160037044761377951668492288750

422469633392304878706725602341482782579852840250681098010280137314308584370130707224123599639141511088446087538909603607640194711643596029271983312598737326253555802606991585915229492453904998722256795316982874482472992263901833716778060607011615497886719879858311468870876264597369086722884023654422295243347964480139515349562972087652656069529806499841977448720155612802665404554171717881930324025204312082516817125

410615886307971260333568378719267105220125108637369252408885430926905584274113403731330491660850044560830036835706942274588569362145476502674373045446852160486606292497360503469773453733196887405847255290082049086907512622059054542195889758031109222670849274793859539133318371244795543147611073276240066737934085191731810993201706776838934766764778739502174470268627820918553842225858306408301661862900358266857238210235802504351951472997919676524004784236376453347268364152648346245840573214241419937917242918602639810097866942392015404620153818671425739835074851396421139982713640679581178458198658692285968043243656709796000

399094734350044227920812480949609126007925709828202578526288763265230518186413734335491367694241324422939693065375201182738796280254432353703622509554356541715928979667908648144582231419142725908974684721803706396953344496626503128747355609262982462494041683090642143510444590777494252367776608092260951518520527813529754494825658383698091837717874396608251405028243431319117112963924571388674865939235441778937354286022382122491565646314525076586034000120036853229848384889623514926325777553544529040492412945656625194172350200498738738780273137920789
```

```
321233542348487346908305455632989416726281869259981520958251727796505906823554 3
139459375028276851221435815957374273143824422909416395375178739268544368126894 2
409791353221760803747809980106577107756258560415940784954117242365602425977591 8
554382479833246791961359866700302599371527487 5
time: 0.004743900s seconds
code>
```