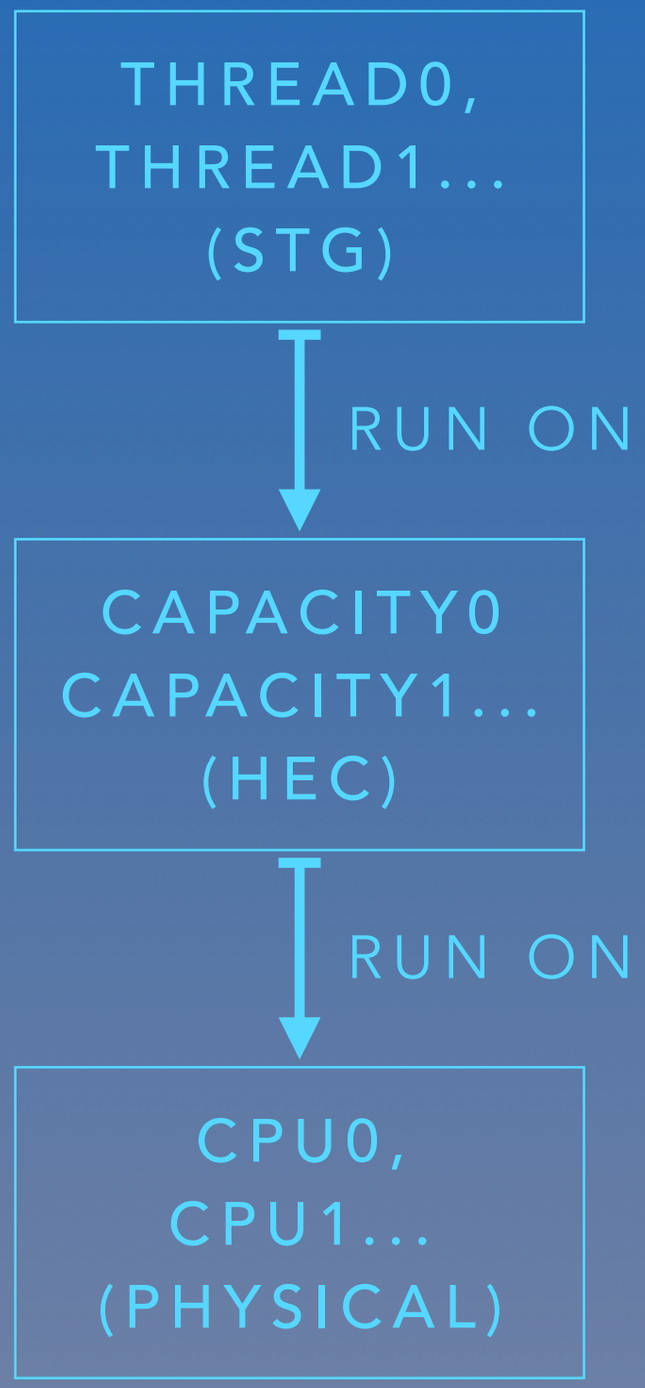
HASKELL PERFORMANCE PATTERN @ 滴滴FP

- HEC
- MEMORY
- FFI
- MVAR
- STM
- IO
- OPTIMIZE

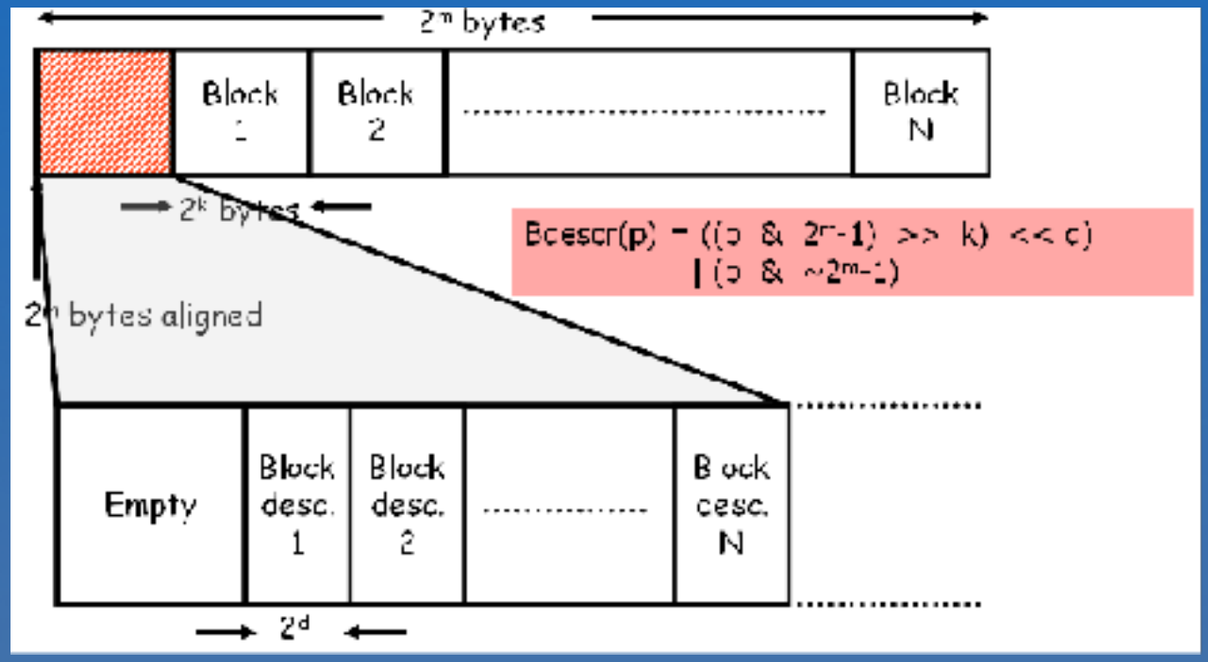
HEC: HASKELL EXECUTION CONTEX

- 为什么要有 HEC
- 操作系统 thread
- Task
- Capacity
- TSO
- Haskell thread



MEMORY

- block allocator
- heap memory
- pinned vs unpinned
- mutable variables
- CAS

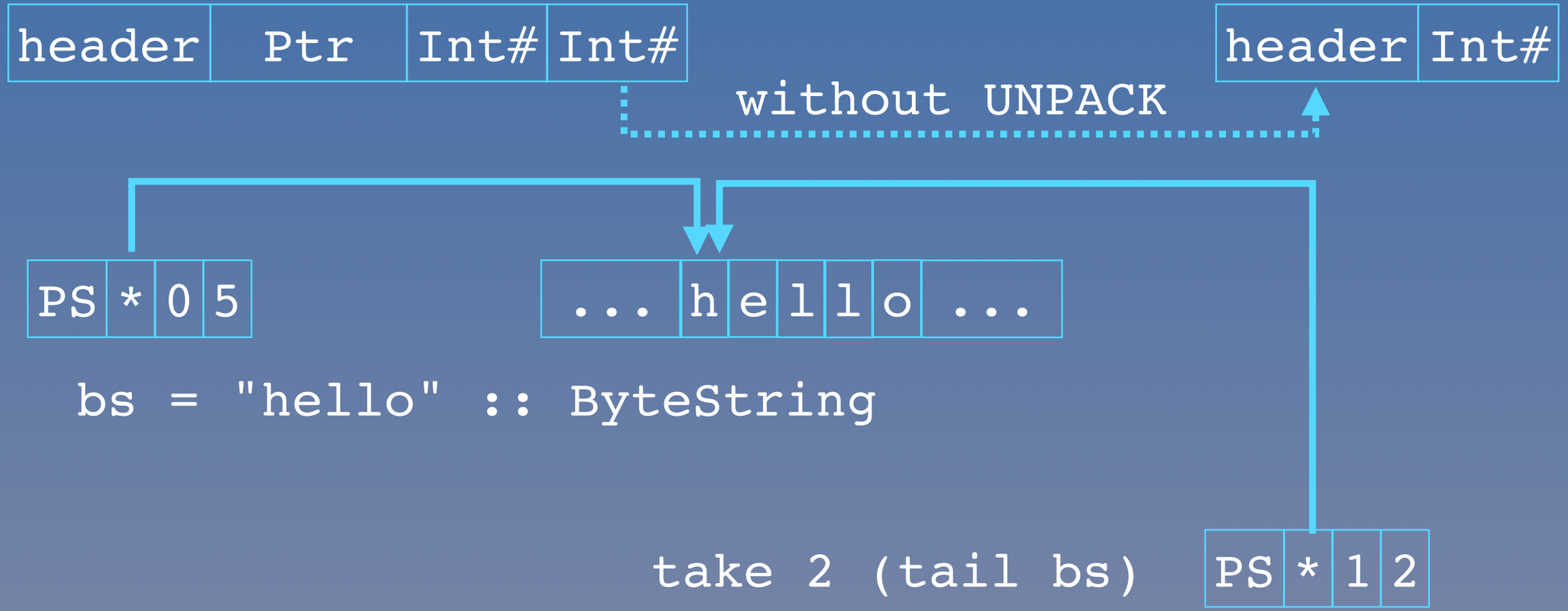


- newByteArray#
- newPinnedByteArray#
- newAlignedPinnedByteArray#

- data MutVar# s a
- readMutVar# writeMutVar#
- casMutVar#
- data MutableArray# s
- casArray#

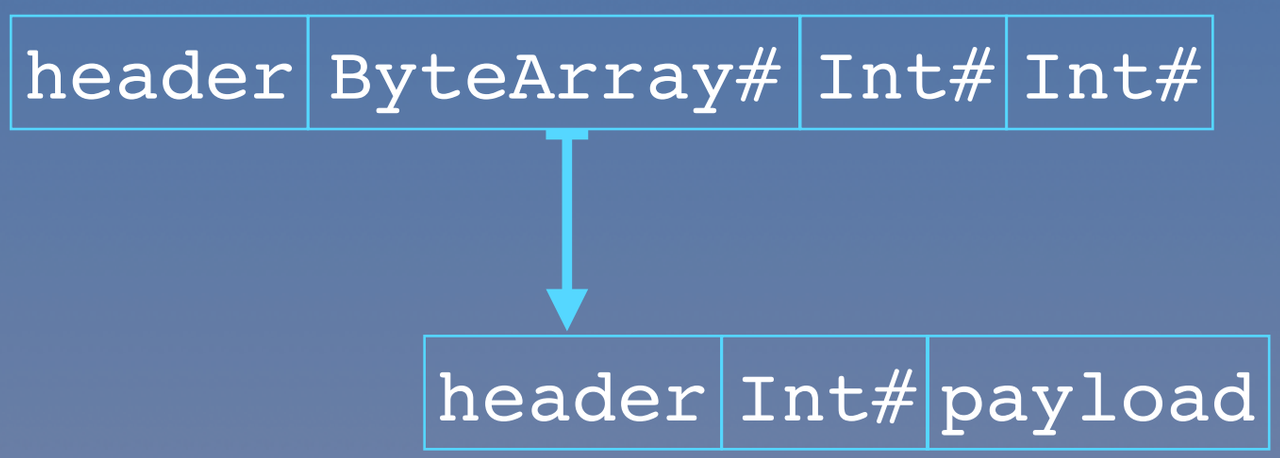
PINNED MEMORY

```
data ByteString = PS
    {-# UNPACK #-} !(ForeignPtr Word8) -- payload
    {-# UNPACK #-} !Int                -- offset
    {-# UNPACK #-} !Int                -- length
```



UNPINNED MEMORY

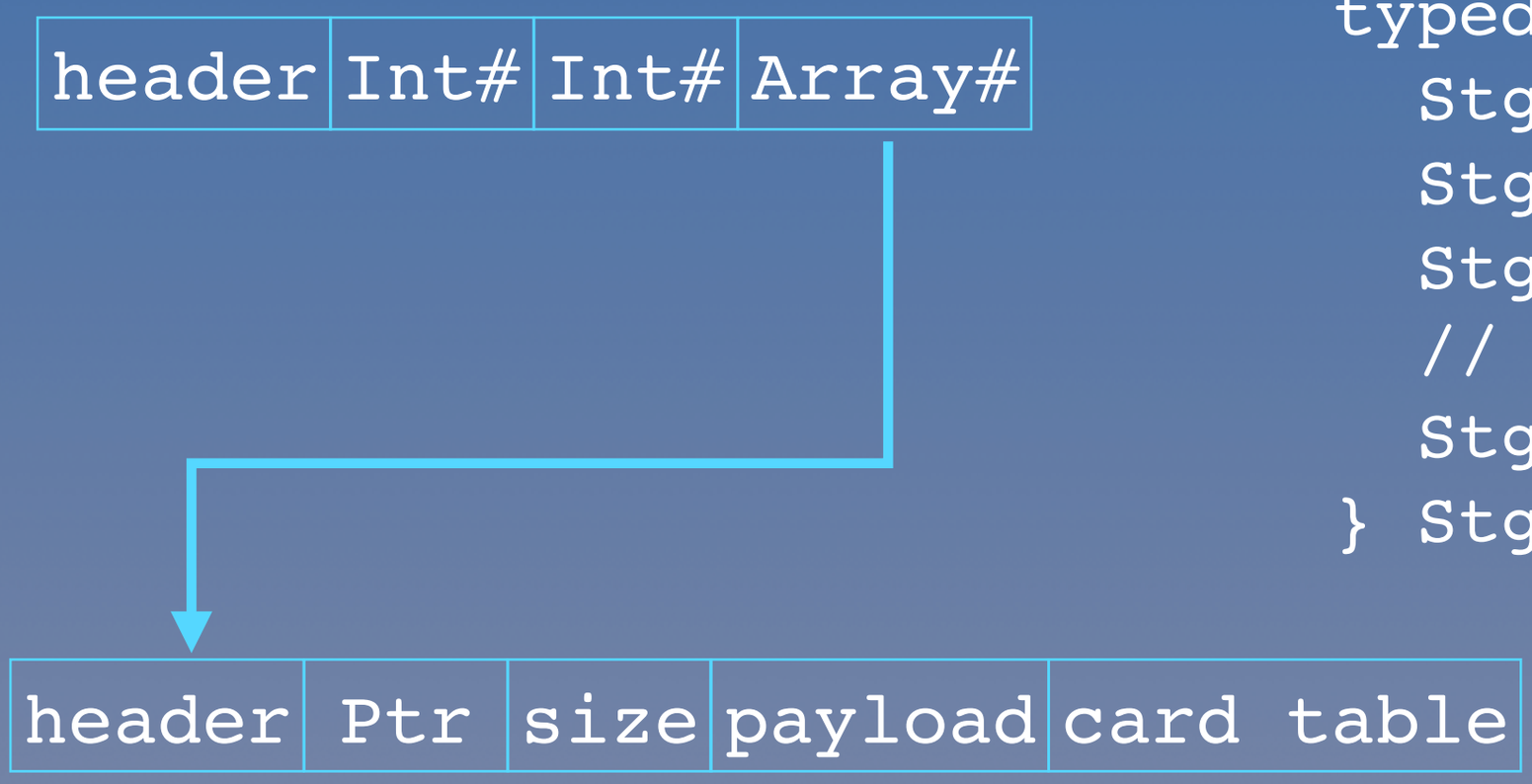
```
data Array = Array { aBA :: ByteArray# }
data Text = Text
     {-# UNPACK #-} !A.Array  -- payload
     {-# UNPACK #-} !Int      -- offset
     {-# UNPACK #-} !Int      -- length
```




```
typedef struct {
  StgHeader    header;
  StgWord      bytes;
  StgWord      payload[];
 } StgArrBytes;
```

UNPINNED MEMORY

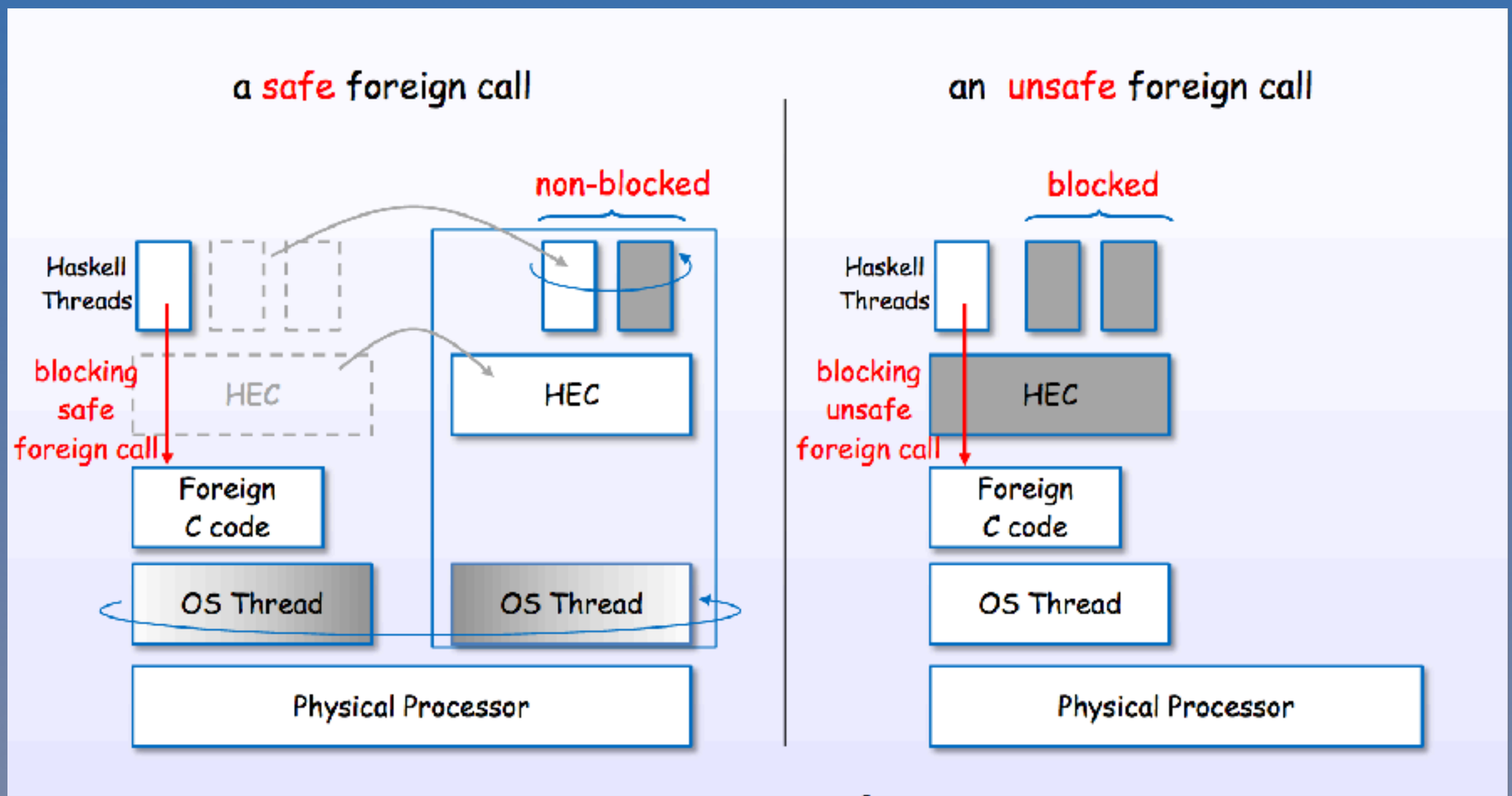
```
data Vector a = Vector {-# UNPACK #-} !Int
                       {-# UNPACK #-} !Int
                       {-# UNPACK #-} !(Array a)
```



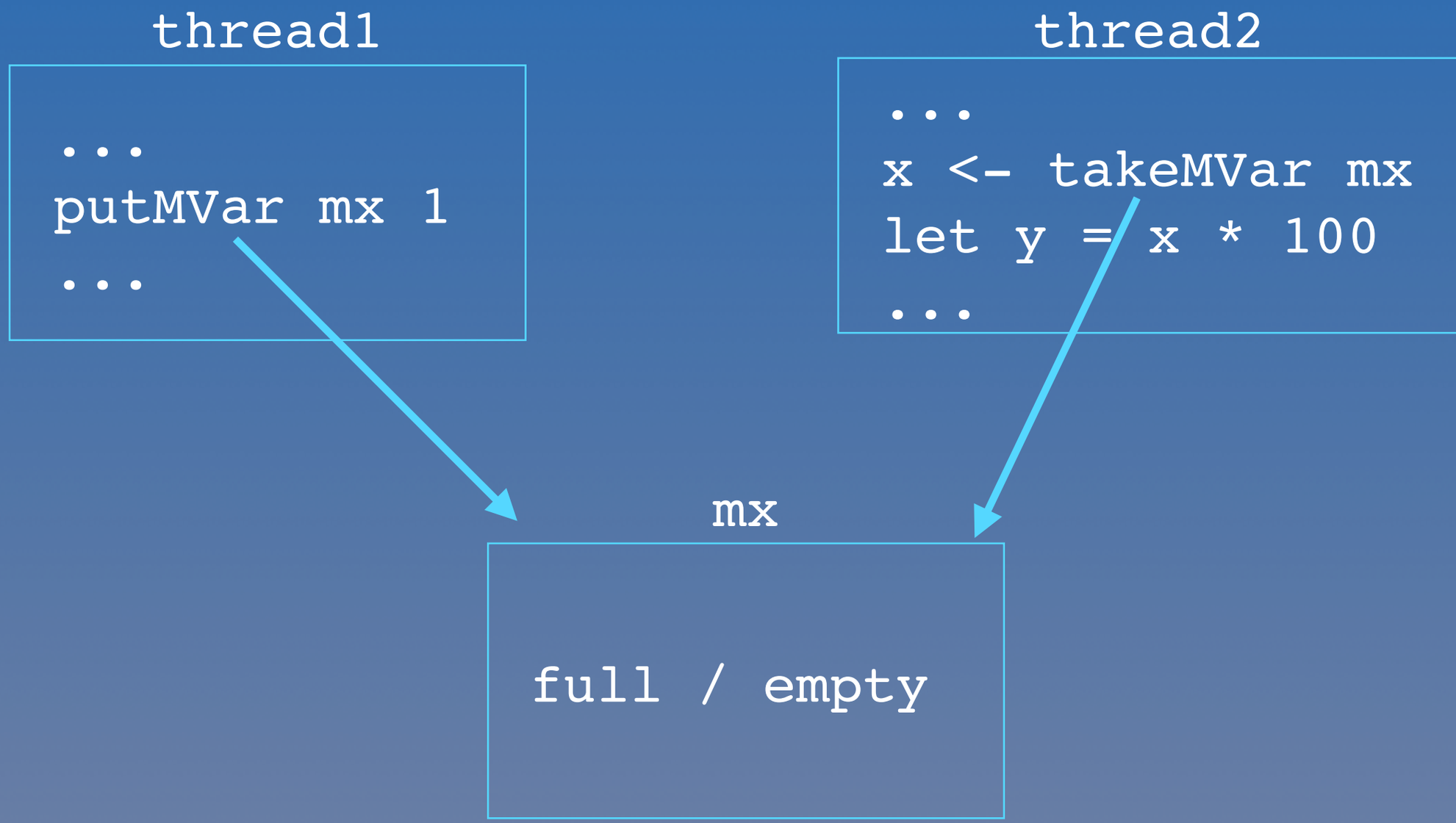

```
typedef struct {
  StgHeader   header;
  StgWord     ptrs;
  StgWord     size;
  // ptrs plus card table
  StgClosure *payload[];
} StgMutArrPtrs;
```

FFI

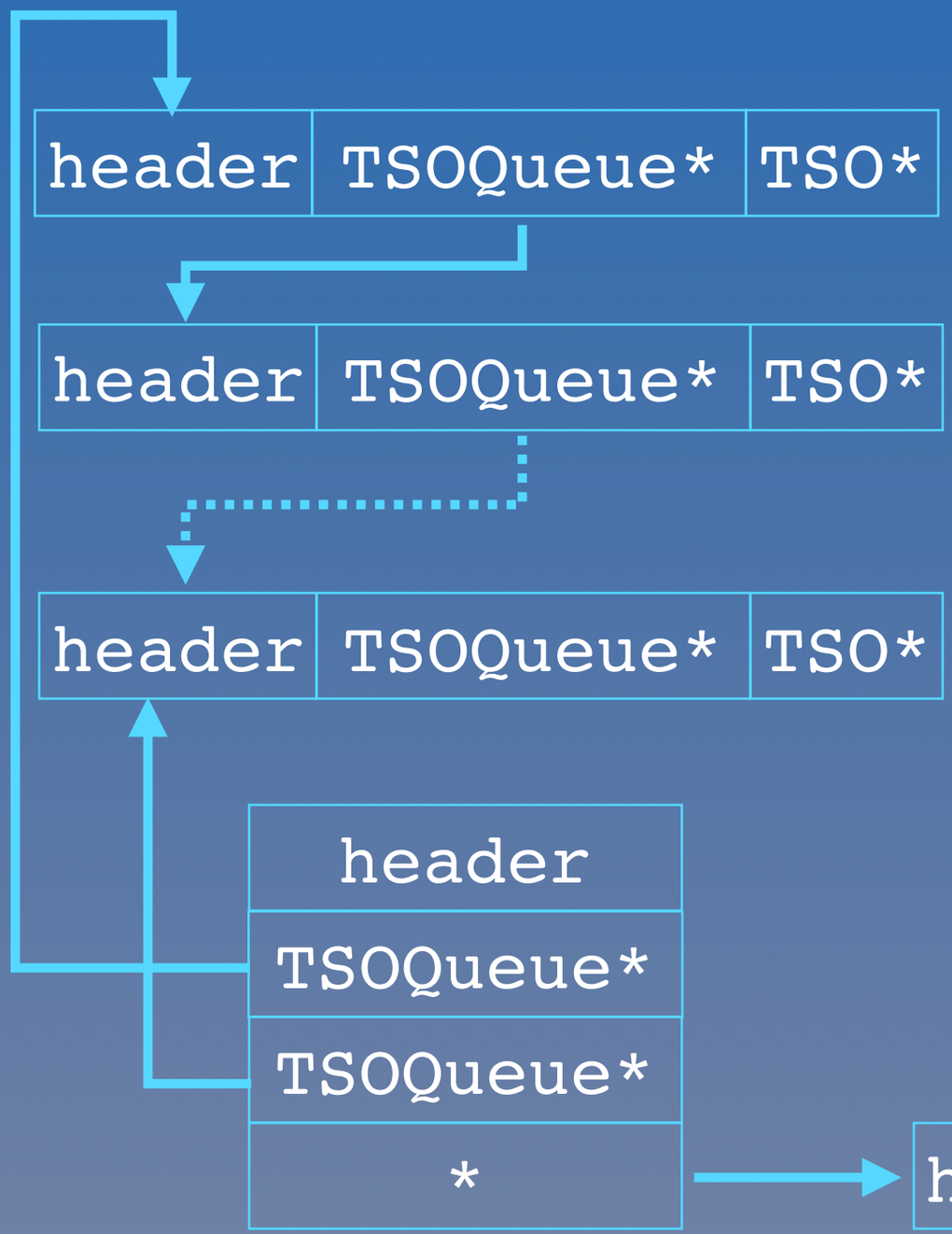
```
foreign import ccall unsafe "_hs_text_memcmp" memcmp
    :: ByteArray# -> CSize -> ByteArray# -> CSize ->
CSize -> IO CInt
```



MVAR: MUTEX VARIABLE

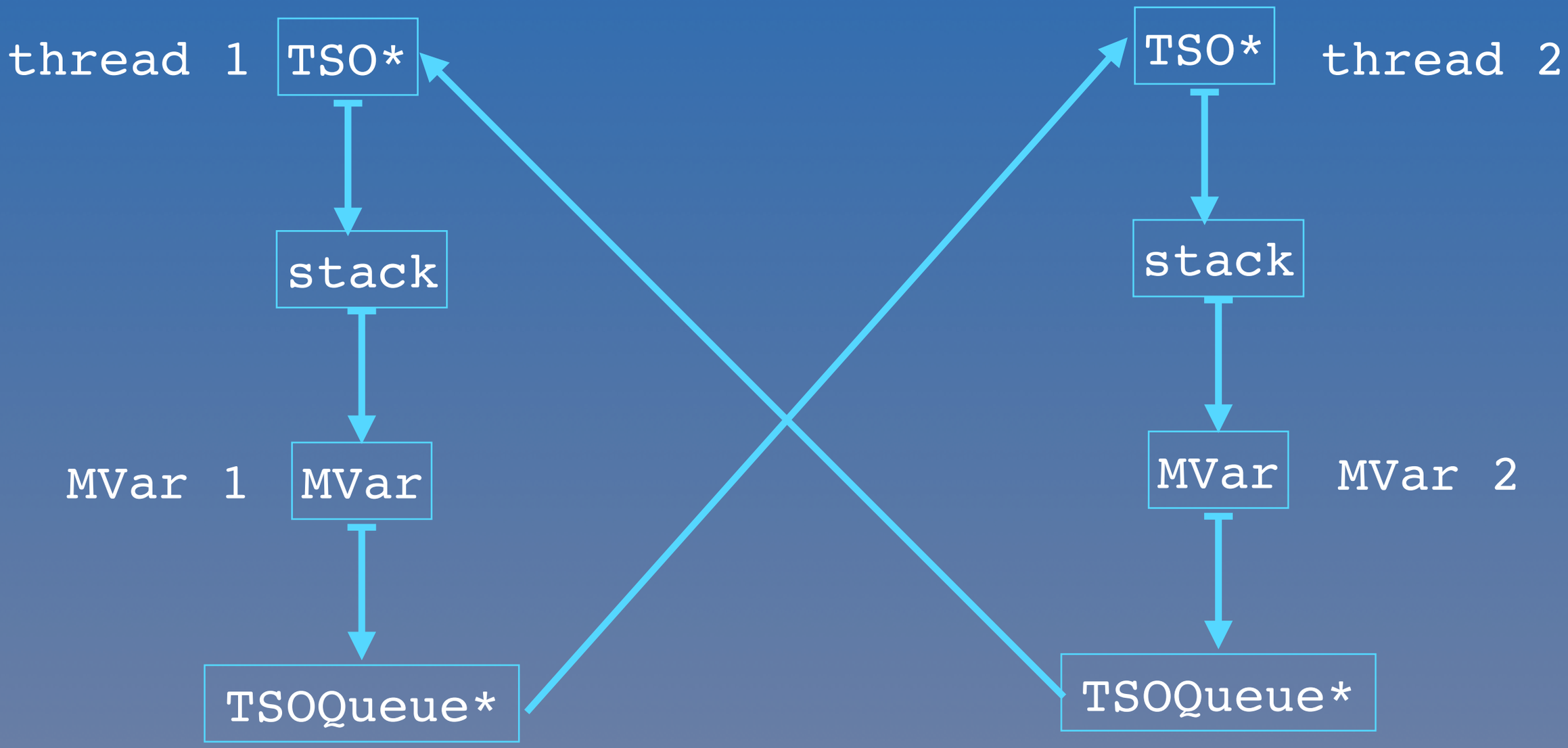


MVAR: MUTEX VARIABLE



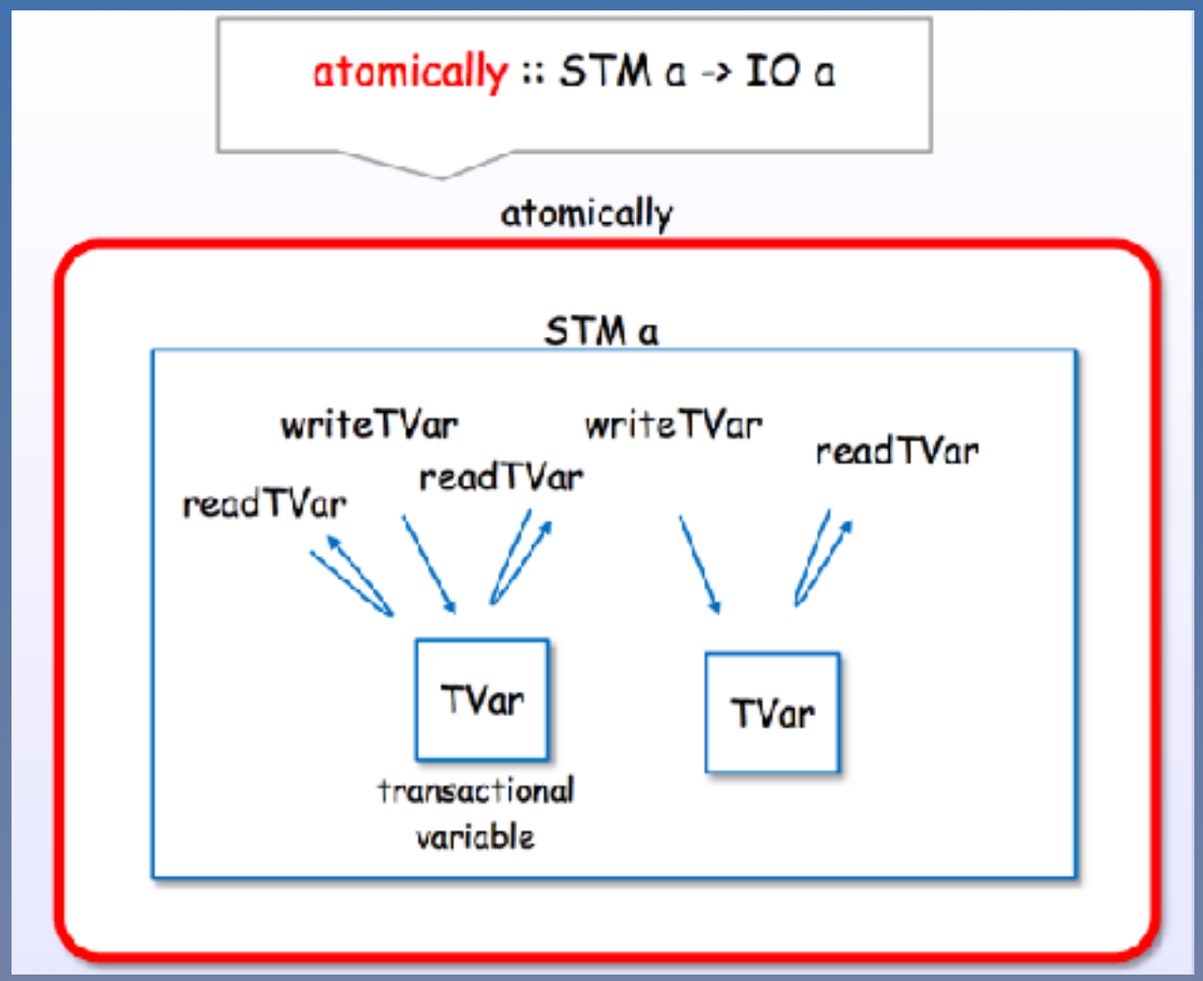

```
typedef struct StgMVarTSOQueue_ {
    StgHeader                   header;
    struct StgMVarTSOQueue_  *link;
    struct StgTSO_           *tso;
} StgMVarTSOQueue;

typedef struct {
    StgHeader                   header;
    struct StgMVarTSOQueue_  *head;
    struct StgMVarTSOQueue_  *tail;
    StgClosure*                 value;
} StgMVar;
```

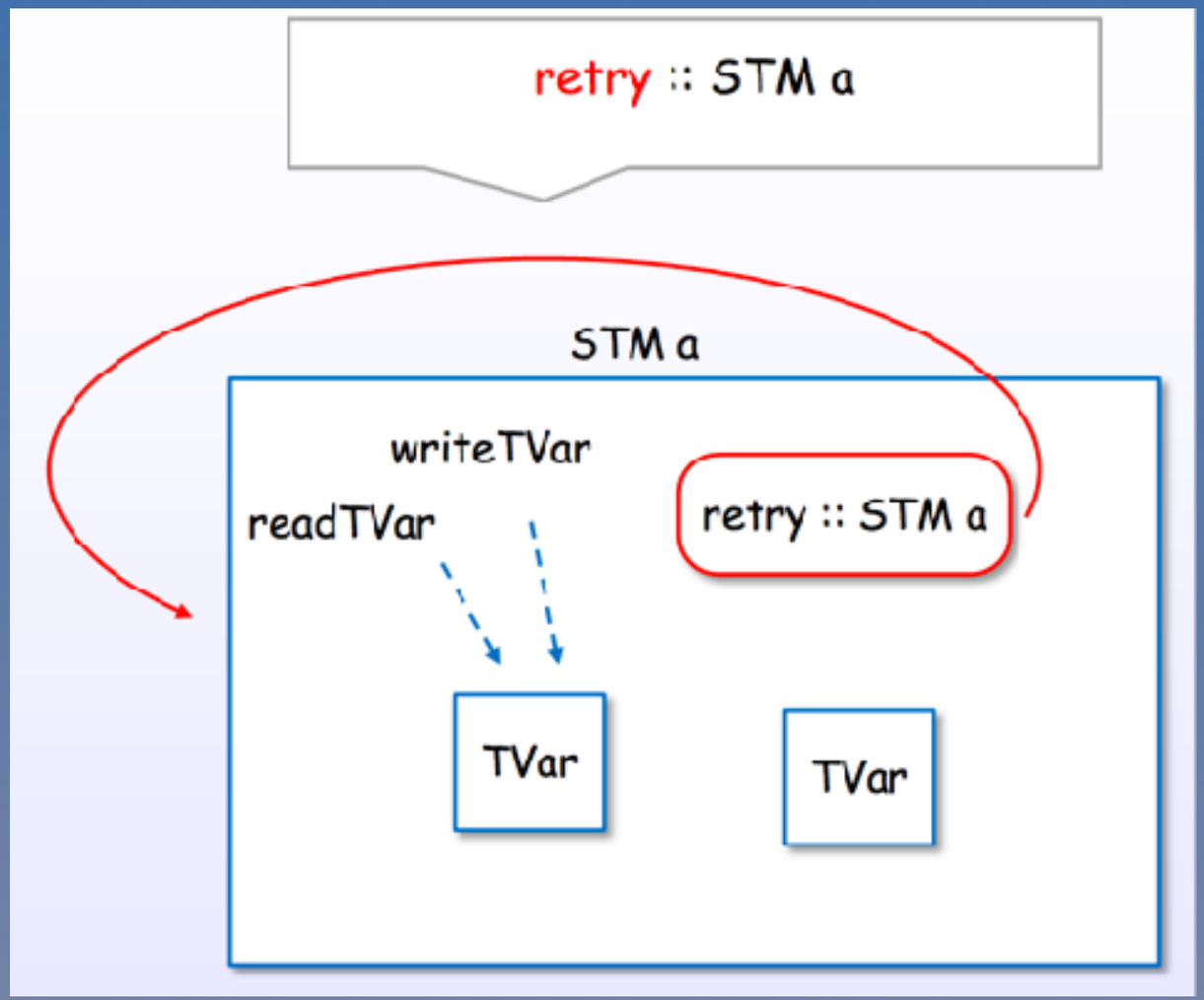
MVAR: BLOCKED INDEFINITELY



STM: Software Transactional Memory

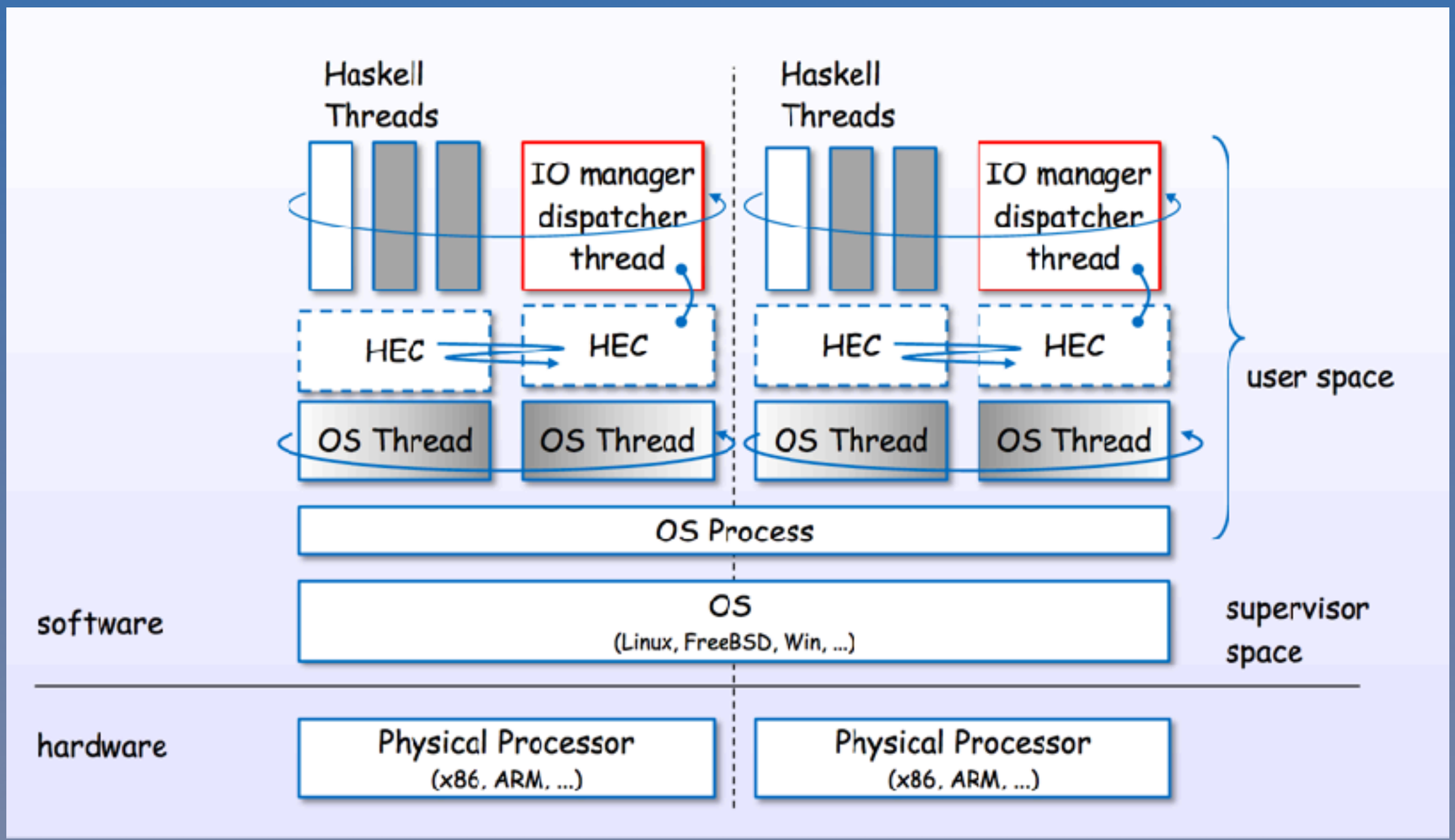
```
atomically :: STM a -> IO a
retry :: STM a
orElse :: STM a -> STM a -> STM a
```



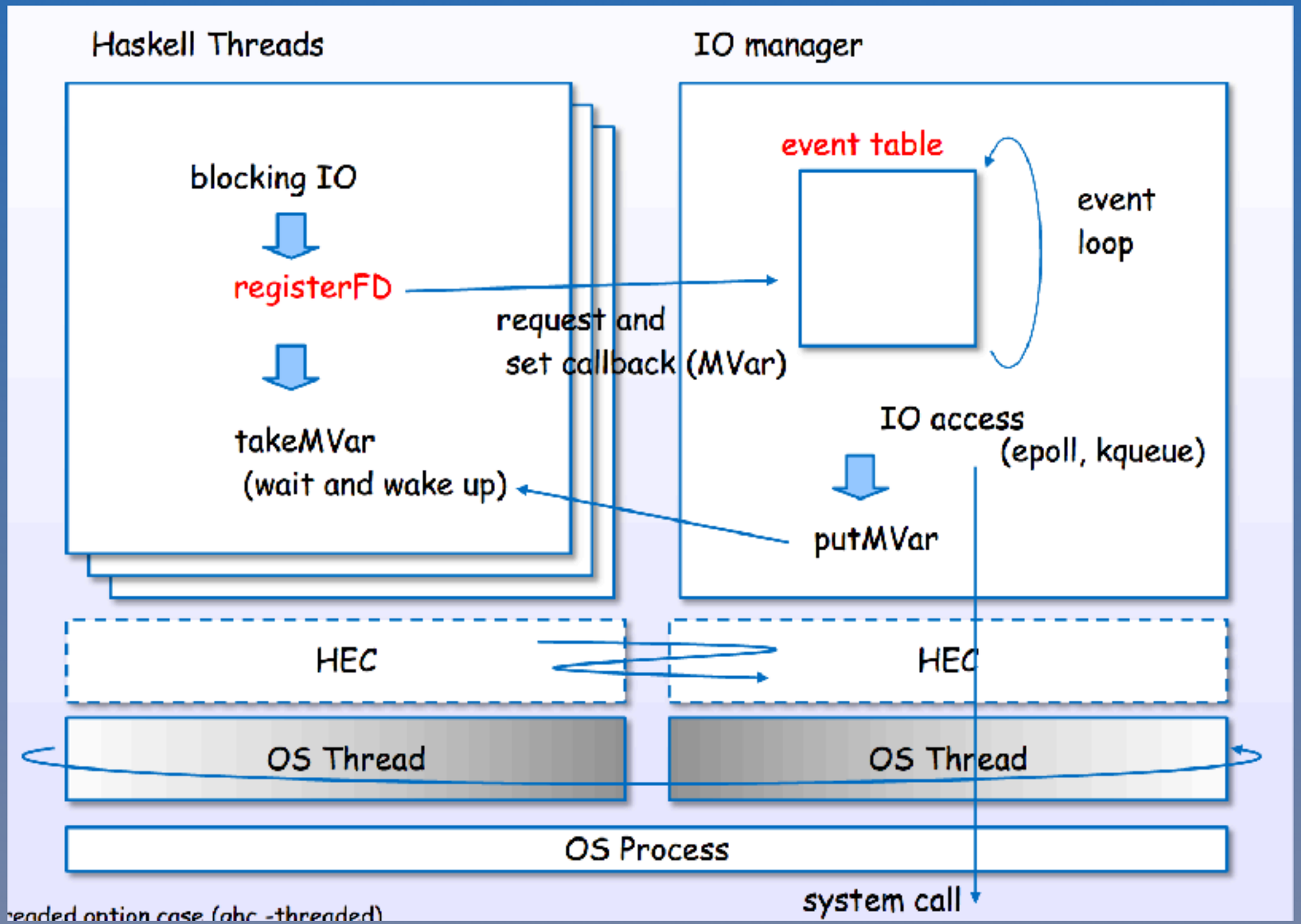


IO MANAGER

```
threadWaitRead :: Fd -> IO ()
threadWaitWrite :: Fd -> IO ()
timeout :: Int -> IO a -> IO (Maybe a)
```



IO MANAGER


Haskell Threads
IO manager
blocking IO
registerFD
takeMVar
(wait and wake up)
event table
event
loop
request and
set callback (MVar)
IO access
(epoll, kqueue)
putMVar
HEC
HEC
OS Thread
OS Thread
OS Process
system call

GHC 8.2: hs_try_putmvar()

```
extern void hs_try_putmvar ( int capability
                           , HsStablePtr sp)
```

- Fast, non-blocking, asynchronous, interface to tryPutMVar that can be called from C/C++.
- 适合任意需要回调的 FFI 调用。

OPTIMIZE

```
when :: (Applicative f) => Bool -> f () -> f ()
when p s  = if p then s else pure ()
{-# INLINEABLE when #-}
{-# SPECIALISE when :: Bool -> IO () -> IO () #-}
{-# SPECIALISE when :: Bool -> Maybe () -> Maybe () #-}
```

- INLINE
- SPECIALIZE
- SPEC-CONSTR
- REWRITE RULES