Haskell Erlang（编程语言） 修改

Erlang VM上的light-weight process和GHC RTS上的相比有何优缺点? 修改

Eralng VM和GHC RTS都支持light-weight process，两者相比各有何优劣?

修改

关注问题 写回答 邀请回答 添加评论 分享 邀请回答 举报 ...

关注者 94 被浏览 3,722

他们也关注了该问题

查看全部 3 个回答

Felis sapiens
函数式编程、编程语言、编程 话题的优秀回答者

Belleve、彭飞等 24 人赞同了该回答

数据同步的方式不同。Haskell thread 共享一个 heap 和一个全局 gc，线程间可以通过 MVar#、TVar# 等机制直接传递任意 Haskell value，同时也可以利用 MVar# 和 TVar# 实现消息传递。而 Erlang 的 process 使用隔离的 heap，只能进行消息传递。

从一名 Haskeller 的角度看，我个人其实更推崇 Erlang 风格的隔离 heap、只支持消息传递的 approach。从底层实现的角度看，写分布式应用最终都得做成消息传递，就算在单个服务器上，NUMA 架构日渐流行，维持原本的运行时实现方式显然会增加不小的开销。从用户的角度来看，如果直接实现分布式的运行时，语言统一只提供消息传递的接口，显然比依赖 Cloud Haskell 一类的框架要来得省心。

Haskell thread 之所以做成共享 heap 的设计，一大原因是历史惯性：Concurrent Haskell 在 1996 年刚实现时，就提供了 forkIO 接口。现在 base 中其类型为 forkIO :: IO () -> IO ThreadId，能够将任意的 IO action 作为新的 thread 的入口，而 Haskell 中的 IO action 和普通函数一样完全是 first-class，可以动态创建的，也就很有可能在闭包中包含了不可序列化的 context 信息。如果要做成分离 heap 的设计，就必须禁止对任意 IO action 的 fork，要么运行时碰到不可序列化的自由变量时抛异常（不安全、不优雅），要么只能指定某个 module 顶层定义的函数以及一系列可序列化参数作为入口（实际上可以用 StaticPointer 来实现）。

编辑于 2017-05-16

▲ 赞同 24 ▼ 7 条评论 分享 收藏 感谢 ...

更多回答

祖与占
函数式编程 话题的优秀回答者

28 人赞同了该回答

LWT只是实现, 关键是在LWT 之上的并发模型不同.

在听过名字而且有 LWT 的语言有三个 Erlang, Haskell, Go, 但是它们的并发模型并不一样. 不妨把 LWT想象成在 OS 提供的硬件抽象上为了实现想要的并发模型语义再上一层的包装, 共同点只是几个都想执行调度单元变得轻量罢了.

回顾 GHC IO管理器的进化历程, 就会发现 Haskell 就像 @Canto Ostinato 所说的那样一开始没想着像 Erlang 那样, 都是按共享内存的思路来搞的. 而 Erlang, 看 Joe Armstrong Making Reliable Distributed Systems in the Presence of Software Errors 就知道, 人家思路完全不一样, 整个系统都搭在 Actor + 消息传递的模型上. 这就是 @Canto Ostinato 所说的数据同步的方式不同.

关于作者

Felis sapiens

函数式编程、编程语言、编程 话题的优秀回答者

电影旅行敲代码、Antokha Yuuki、暮无井见铃也关注了她

回答 249 文章 40 关注者 14,871

已关注 发私信

被收藏 5 次

写代码相关 3,364 人关注
Sen Zhang 创建

haskell 3 人关注
baozii 创建

Functional Programming 0 人关注
孙晓光 创建

电影 0 人关注
杨丹 创建

Erlang 这样的数据同步方式优点还是有不少的:

每个"进程"一个堆的话相当于这个进程里面的资源的声明周期跟这个进程绑定起来, 一旦这个进程生命周期结束, 里面的资源就要释放, 资源里面最重要的是内存, 这样 Erlang RTS 里 GC 的成本就会均

展开阅读全文

baozii
有空来我家吃饭吧

1 人赞同了该回答

1个相同之处：ghc和erlang都支持极低开销的进程/线程，可以单机百万乃至千万级的并发

3个不同之处：

01 erlang强调软实时，保证每个actor分到的时间片尽可能公平，而且使用抢占式调度，保证没有任何一个actor能抓着CPU不放

ghc只能在绝大部分情况下保证调度的公平性，不支持抢占式调度，但在绝大多数情况下线程不会抓着CPU不放。这样的好处是线程切换的开销极低，对性能的影响几乎测不出来

02 erlang使用消息传递。好处是这带来了极其简单的异常处理。坏处是当消息大到一定程度的时候复制的开销很大

展开阅读全文

查看全部 3 个回答

CS 0 人关
喵了个咪的 创建

相关问题

Haskell的编译器GHC是Haskell写的吗? 4 个回答

如何评价最新推出的 Glasgow Haskell Compiler (GHC) 8.0.1 版本? 5 个回答

讨论Erlang VM中的GC，它是一个伪命题吗? 4 个回答

为啥 Erlang 没有像 Go、Scala 语言那样崛起? 43 个回答

erlang有哪些好用的库? 8 个回答

相关推荐