编程 Haskell 修改

Haskell有哪些提高效率的技巧? 修改

在学习Haskell，Monad,STRef等特性和很多内置函数用得不太溜，做很多事情都是从头自己写起来，可能执行效率会很慢。有什么能提高程序执行效率...显示全部

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 194 被浏览 4,828

他们也关注了该问题

查看全部 5 个回答

Felis sapiens
函数式编程、编程语言、编程 话题的优秀回答者

酿酿酿酿酿泉、彭飞、阅千人而惜知己、开源哥、圆角骑士魔理沙等 62 人赞同了该回答

问题问得太混乱。我只回答前半个问题（提高程序执行效率/优化程序的思路）吧。。

1. 使用合适的数据结构

比如尽量多用Vector代替List（除非需要大量cons/uncons操作）；Writer monad中用DList代替List；尽量用unordered-containers代替containers，用ByteString/Text代替String，等等；

2. 不要过早使用IORef/STRef

能用StateT的场合尽量避免直接用IORef/STRef，实际上性能一般都是前者更佳。。（

3. 使用合适的并行/并发抽象

比如需要共享状态时，基本的primitive有IORef、MVar、TVar，其overhead是逐渐递增的，而能力也是逐渐递增的（IORef支持单变量CAS操作，MVar有blocking的get/put，隐含了condition variable，而TVar背后的STM支持多变量读写的事务），选择能够满足需要的最弱的一个；Haskell线程和array/dataflow parallel编程相关的，这个要讲也太多了，强烈推荐Simon Marlow的《Parallel and Concurrent Programming in Haskell》

4. 适时使用Streaming I/O

这个请自行查阅《九评Lazy I/O》，哦不对，pipes/conduit文档。。

5. 动静结合调试性能问题

静：学会读ghc编译输出Core代码（尤其是经过simplifier之后的），看哪些地方inline了，哪些地方specialize了，哪些地方应用了strict/unbox优化，等等。。

动：使用RTS的profile功能；使用threadscope工具查看eventlog；使用criterion进行benchmark，比较性能，用weigh来检测各种内存分配行为；对于生产环境的服务器，用ekg监视内存占用，等等

6. 适量使用strict求值

说到性能，是不是惰性求值就是性能的天敌呢？非也。。这个展开来说就长了，总之，需要引入strict求值的工具我们还是有的，seq/deepseq，bang pattern，以及ghc 8引入的Strict语言扩展。应适量使用strict求值，并且通过benchmark来衡量效果，而不是无脑默认全部lazy/strict。

7. LLVM后端有奇效

尤其是一些并行度高的数值计算代码，LLVM后端比默认的ASM后端生成代码效率有明显提升（编译时间也是）。RTS支持的primop里，有一组SIMD primop就是只在LLVM后端才支持的（想看RTS到底支持哪些primop，自行查看ghc-prim的haddock文档）。

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 181 文章 40 关注者 14,872

已关注 发私信

被收藏 22 次

Programming Languages 3,642 人关注
彭飞 创建

Haskell 10 人关注
parker liu 创建

CS 8 人关注
Caesar Julius 创建

haskell 3 人关注
baozii 创建

8. 理解一些语言特性的编译器实现
比如一条最重要的常识：Haskell中，多态是有性能代价的（很多人误以为type class编译完了以后就自动specialize掉了，实际上不总是如此）；另外还有喜闻乐见的Generics/Template Haskell之争，要性能不要优雅？那就手写Template Haskell（

编辑于 2016-07-01

更多回答

baozii
有空来我家吃饭吧

12 人赞同了该回答

长期在reddit上围观别人撕逼，互怼后……我总结出有几个基础的性能提升要诀是应该了解的，不然很容易被打脸 (·̀ω·́)

01 不要用尾递归，要么尾递归和严格求值一块用。这条可以扩展成多用foldr，少用foldl。

大部分从其他函数式语言转过来人尤其容易中招。尾递归后性能不升反降，为什么？构建太多 thunk

一看foldr，直觉要遍历整个表，其实不是的，惰性求值保证结果一出来就立即返回，有时候就读个表头

02 表操作 ++ 注意其右结合属性。参考手册list fusion一节，了解“good producers”和“good

展开阅读全文

方泽图
热爱编程语言

3 人赞同了该回答

楼上 TJ（毕竟匿名了，我也不好直接 @..）已经说的很好了，我再补充几点：

- 怎么写执行效率高？

在可读性差不多的情况下，尽量选择依赖较少/语义较弱的写法，因为这样做（据说）更易为编译器所优化，而且通常也更为简单易懂一些。
比如如果某件事情用 Applicative 就能精确描述的话，那就可以不用 Monad （例如用 parser combinator 来处理 context-free grammar）。再比如能用 List.{foldr,unfold} 甚至是 fixpoint functor 的时候那就别手写递归，前者语义比较弱，GHC 里面有对应的 rewrite rules 来进行 stream fusion，楼上所提到的 streaming IO 库也是充满 rewrite rules。手写递归就没有这个好处。当然具体的情况还是要具体分析，效率分析工具很重要。

- 过程式的写法好不好？

展开阅读全文

PL 2 人关
lsdsjy 创建

相关问题

如何写出优美的 Haskell 代码? 8 个回答
Haskell 这段代码该如何理解? 7 个回答
阻挡你学会 Haskell 最大的两个问题是什么? 37 个回答
Haskell 最有代表性的一段程序是什么? 6 个回答
如何理解下面这段Haskell代码? 4 个回答

相关推荐

淼懂物理学：理解世界的极简指南
共 31 节课 ▶ 试听

数学妙啊！妙！
张英锋 等
289,069 人读过 阅读

查看全部 5 个回答