20211024-系统领域- GHC编译器移植-调研报告-王金龙

相关术语

缩写	全称	描述
GHC	**G**lasgow **H**askell **C**ompiler	类似 **GCC** 的生态编译器
Haskell	**Haskell**	一种函数式高级编程语言
Pandoc	**Pandoc**	**Haskell** 语言编写的文档格式转换程序

一 、问题

1.1 移植背景

UnionTech **OS** (**UOS**) 的用户协议是和 **Pandoc** 高度关联的，而 **Pandoc** 自身是用 **Haskell** 函数式编程语言所编写的。不幸的是 **Haskell** 语言的编译器 **G**lasgow **H**askell **C**ompiler (**GHC**) 并没有被 **GHC** 官方的团队移植到我们使用的 loongarch64 平台上，因此这一部分需要我们自己来完成。 所以当前最根本的问题就是我们要在 **“UOS + LoongArch64”** 的平台中得到可以在 loongarch64 架构下可编译 **Pandoc** 的 **GHC**。

二 、现状

2.1 GHC的相关概念

1. What is **GHC** ?

   GHC 是用于函数式语言 **Haskell** 的最先进的开源编译器和交互式环境。**GHC** 由两个主要部分组成：一个是交互式的解译器 **GHCi**, 另一个是批编译器（参见 Using GHC ）。事实上，GHC 是一个单一的的 程序，只是以两种不同的选项运行，以提供交互功能或批编译功能。批编译器可以与 **GHCi** 一起使用：编译过的模块能够被加载到交互环境中并能够像 解释的代码那样被使用，事实上当你使用 **GHCi** 时，大部分的库都将被预编译，这意味着你可以同时享受预编译的代码带来的执行速度上的提升及开发过程中快速编译所写代码而带来的便利性。**GHC** 支持多种扩展特性，包括：并发、外部函数接口、异常、类型系统扩展（如： 多参数类型类、todo）。**GHC** 提供了全面的优化措施可供选择，所以当你真的使用了这些优化措施时，**GHC** 能够生产出执行速度相当快的代码。**GHC** 默认选择以最快的速度编译代码，而不是对生成的代码做太多的优化。

2. What is **Haskell** ?

   Haskell 是一种高级的纯函数式编程语言。C 语言是发展至今依然是市场主力军，但是由于受限于过程化编程的思想，当软件项目变得足够庞大，C/C++ 的阵营便显得有点力不从心，得益于 GHC 编译器做了大量的微优化，在有些场合 Hakell 语言在其综合效率上明显的胜过 C 语言。

3. Compile **GHC**

GHC 的编译方式类似于 GCC，编译的基本源文件是 Haskell，通常后缀标记为 .hs ，GHC 通过会将 Haskell 源文件编译成 静态库，动态库，可执行文件等，并且在执行时会创建自己的运行态环境。

2.2 GHC的移植原理

GHC 的移植情形目前分为两类：第一类是处架构支持但是操作系统不支持，这种情况下我们不需要去移植架构相关的部分，只需要在有 GHC 的其他同架构的操作系统中直接编译构建即可。第二种是处理器架构不支持，这种情况就比较棘手，难度不亚于移植 GCC ，好在 GHC 团队给这一部分用户留了最后一条路： GHC 源码在底层中加入了纯 C支持，也就是说在这种情况之下，有经验的用户可以使用任何带有 GCC 的 POSIXish 系统，进行跨平台的交叉编译，显然 loongarch64 属于后者。

2.3 影响移植的因素

- GHC 版本的影响：

  由于 Linux Distribution 中的 GHC 的编译依赖于 GHC 自身，并且每个 Distribution 都有不同的出入或有小特性加入或有小版本微调， 所以 GHC 能否成功移植到新的架构受 GHC 的版本影响。

- GCC 版本的影响：

  GHC 是类似于 GCC 的生态编译器，而且正如 2.2 节中提到的要想在新架构中移植，需要编译纯 C 代码，所以不同版本的 GCC 中所包含的 C 编译器对 GHC 移植同样会造成影响。

- OS 版本的影响：

  由于 GHC 的开源特性，使得它被广泛的应用到众多的 Linux Distribution 当中，而这些众多的 Distribution 都根据自己的定位，自身在对 GHC 的管理和功能配置上有着不同的删改和阉割。

- 现有的可用资源：

资源	宿主机	目标机
处理器架构	amd64	loongarch64
运行内存	>= 8GB	>= 8GB
硬盘内存	>= 60GB	>= 60GB
操作系统	UnionTech OS 20 (with GHC)	UnionTech OS 20 (without GHC)
编译器	gcc-8.3.0 (8.3.0.7-1+dde)	gcc-8.3.0 (8.3.0-6-lnd.vec.23)
交叉编译器	loongarch64-linux-gnu-2021-06-19.tar.gz	---
GHC 版本	8.4.4+dfsg1-3	None

2.4 移植的探索历程

GHC 版本	初次移植结果
6.10.4	GHC configuration does not support differing host/target (i.e., cross-compiling)。该版本尚不支持交叉编译，如果想要新架构支持 GHC，必须要从底层从头做起并且还需要很熟悉 GHC。
6.12.3	此版本开始支持交叉编译，config* 加入 loongarch64 支持，编译报错 make [2 of 61]。
7.0.3	config* 加入 loongarch64 支持，编译报错 make [2 of 64]。
7.2.2	config* 加入 loongarch64 支持，编译报错 make [2 of 65]。
7.4.2	config* 加入 loongarch64 支持，编译报错 make [2 of 67]。
7.6.3	config* 加入 loongarch64 支持，编译报错 make [2 of 68]。
7.8.4	config* 加入 loongarch64 支持，编译报错 make [10 of 73]。
7.10.3	config* 加入 loongarch64 支持，编译报错 make [1 of 90]。
8.0.2	config* 加入 loongarch64 支持，并失能 HADDOCK，make [25 of 95]。
8.2.2	config* 加入 loongarch64 支持，并失能 HADDOCK，make 提示 ghc-cabal: Encountered missing dependencies:base >=4.4.1 && <4.11 ghc/ghc.mk:112: ghc/stage1/package-data.mk: 没有那个文件或目录。
8.4.4	config* 加入 loongarch64 支持，并失能 HADDOCK，make 提示 utils/hsc2hs/ghc.mk:24: utils/hsc2hs/dist-install/package-data.mk: 没有那个文件或目录。
8.6.4	同 8.4.4 。
8.8.4	同 8.4.4 。
8.10.4	configure: error: GHC version 8.6 or later is required to compile GHC。
9.0.1 (latest)	configure: error: GHC version 8.8 or later is required to compile GHC。

2.2 最终版本的确定

当前 **UOS** 仓库中的 GHC 源码版本为 ghc-8.4.4+dfsg1-3 这个是基于 Debian buster 的版本。在经历过多次版本的尝试之后,发现 GHC 这个编译器与 GHC 自身版本， GCC 的版本，以及所使用的 Distrobution 的版本有着密不可分的关系，比如说如果交叉编译完 GHC-8.8.4 并且将其手动打包成 Deb 包，然后再用它在 loongarch64 平台上 DPKG 编译 GHC-8.4.4 会出错，原因是 8.4.4 和 8.8.4 的版本不一样，内部函数的实现方法不一样导致的。此外不同的 GCC 和 Distribution 也会造成多种多样的不同问题。结合以上对所有探索版本的总结，最终选择 ghc-8.4.4 为本次移植的最终版本，这样的折衷选择不仅使仓库源码中的版本一致性得以保持，而且也可以使为移植所做的努力更加有意义。

本篇文档仅仅涉及了成功编译和移植 GHC-8.4.4 所涉及的最顺畅最基本的作业流程，至于更进阶的移植修改由于种种原因不便编写。

三、技术方案

3.1. 整体设计

GHC 官方在对新架构移植的过程给出了如下的参考：

	Stage 0	libs boot	Stage 1	libs install	Stage 2
built on	---	build	build	host	host
runs on	build	build	host	target	target
targets	build	---	target	---	target

- **Stage 0:** the GHC that is already on the build system(the one you specify using --with-ghc when configuring the build), comes with a set of built libs, could be older than the version of GHC being built
- **libs boot:** libs that the current version of GHC being built relies on that are either absent or too old in order versios of GHC that might be being used as Stage 0. These libs are built with Stahe 0 GHC, and linked into the Stage 1 GHC being built.
- **Stage 1:** the first GHC built, compiled by the Stage 0 GHC, and linked with both libs from that GHC installation, and the boot libs.
- **libs install:** libs that will accompany the final compiler, built by the Stage 1 GHC.
- **Stage 2:** the final GHC built,compilked by the Stage 1 GHC, and linked with only the install libs.

上面的描述虽然说是官方给出的可信度最高，最权威的参考资料，但是过于晦涩抽象，对操作者的技术面宽广有一定程度的要求。所以经过多次的考量修改，总结成以下相对更容易理解的过程。要想在 GHC 官方尚未支持的硬件平台上面编译 GHC，这里必须通过交叉编译的方式来进行。总结以下六个阶段：

阶段	简述
01-宿主机交叉编译	交叉编译得到基本 GHC 编译依赖，为 Debian 化提供内容。
02-Deb化第一阶段	Deb 化前面的编译结果，解决 Distribution 依赖的问题。
03-目标机直接编译	目标机直接编译，引入生成的交叉编译时缺失的 Haddock。
04-Deb化第二阶段	Deb化前面的编译结果，解决DPKG时Distribution 的依赖。
05-DPKG自动编译	安装使用 04 步骤编译得到的 GHC-Deb，DPKG 编 GHC。
06-验证编译结果	用变异生成的 GHC 编译 Pandoc。

3.2. 阶段简述

1. 宿主机交叉编译

   该阶段的主要目标所以通过交叉编译工具链直接编译生成适配于 loongarch64 架构下的基本 ghc，为后续的目标机的直接编译以及 Deb 化提供内容。

2. Deb化第一阶段

   该阶段是将阶段 1 的编译结果进行分拆打包，解决 debian-base 系统下 ghc 编译时相关包的编译依赖。

3. 目标机直接编译

该阶段是添加交叉编译时缺失的 haddock 二进制文件（交叉编译时该选项失能，否则无法编译），为 Deb 化第二阶段提供内容。

4. Deb化第二阶段

   将阶段 3 的编译结果 Deb 化，与 Deb 化第一阶段功能相同。

5. DPKG自动编译

   该阶段从 debian 源码仓库中得到 ghc 的 dfsg 规范源码，使用 dpkg 包管理器直接编译。

6. 编译结果验证

   该阶段有两个验证步骤：直接 Haskell 源码验证、用生成的 ghc 编译 Pandoc 用来检测生成的 GHC 是否具备基本的能力。

四、实验验证

4.1. 宿主机交叉编译

- 软硬件环境

硬件环境	Arch	Mem	Hdd
编译宿主机	x86_64	>= 8GB	>= 60GB

软件环境	OS	Compiler
编译宿主机	UnionTech OS 20	cross compiler gcc-8.3.0 (8.3.0-6-lnd.vec.23)

- 期望结果

  本阶段我们使用 **GHC** 官方的发行源码包编译，使用龙芯提供的交叉编译工具链 loongarch64-linux-gnu-2021-06-19.tar.gz。

4.1.1 Prepare

- 配置编译环境

```
# UnionTech OS GNU/Linux 20
sudo apt-get install ghc libncurses5-dev # 安装依赖软件
export GHC_WORKDIR = "准备路径" # GHC 工作路径
echo $GHC_WORKDIR

# 该路径下的资源
uos@uos-VB:~/Workspace$ ls -al $GHC_WORKDIR
总用量 464648
drwxr-xr-x 2 uos uos      4096 9月  27 05:44 .
drwxr-xr-x 5 uos uos      4096 9月  27 03:40 ..
-rw-r--r-- 1 uos uos    120621 9月  27 05:33 0001-port-loongarch.diff # 由龙芯提供
-rw-r--r-- 1 uos uos 475660716 9月  27 01:24 loongarch64-linux-gnu-2021-06-19.tar.gz # 由龙芯提供

# 配置交叉编译工具链
tar zxvf loongarch64-linux-gnu-2021-06-19.tar.gz
export PATH=$GHC_WORKDIR/loongarch64-linux-gnu-2021-06-19/bin:$PATH

# 获取 GHC 源码
```

```
cd $GHC_WORKDIR
wget https://downloads.haskell.org/~ghc/8.4.4/ghc-8.4.4-src.tar.xz
tar xvf ghc-8.4.4-src.tar.xz
export GHC_SRC=$GHC_WORKDIR/ghc-8.4.4
```

4.1.2 Config

- 增加对 loongarch64 支持

```
diff -Nuar ghc-8.4.4-orig/config.sub ghc-8.4.4/config.sub
--- ghc-8.4.4-orig/config.sub   2017-11-29 00:39:14.000000000 +0800
+++ ghc-8.4.4/config.sub        2021-09-27 06:21:34.105684732 +0800
@@ -267,6 +267,7 @@
        | ip2k | iq2000 \
        | k1om \
        | le32 | le64 \
+       | loongarch64 \
        | lm32 \
        | m32c | m32r | m32rle | m68000 | m68k | m88k \
        | maxq | mb | microblaze | microblazeel | mcore | mep | metag \
diff -Nuar ghc-8.4.4-orig/configure ghc-8.4.4/configure
--- ghc-8.4.4-orig/configure    2018-10-14 03:54:34.000000000 +0800
+++ ghc-8.4.4/configure 2021-09-27 06:23:02.957684732 +0800
@@ -3930,6 +3930,9 @@
   ia64)
     TargetArch="ia64"
     ;;
+  loongarch64)
+    TargetArch="loongarch64"
+    ;;
   m68k*)
     TargetArch="m68k"
     ;;
@@ -9630,6 +9633,9 @@
         alpha)
             test -z "$2" || eval "$2=ArchAlpha"
             ;;
+       loongarch64)
+            test -z "$2" || eval "$2=ArchLoongarch64"
+            ;;
         mips|mipseb)
             test -z "$2" || eval "$2=ArchMipseb"
             ;;
diff -Nuar ghc-8.4.4-orig/libraries/base/config.sub ghc-
8.4.4/libraries/base/config.sub
--- ghc-8.4.4-orig/libraries/base/config.sub    2017-11-29 00:39:14.000000000
+0800
+++ ghc-8.4.4/libraries/base/config.sub 2021-09-27 06:23:31.653684732 +0800
@@ -267,6 +267,7 @@
        | ip2k | iq2000 \
        | k1om \
        | le32 | le64 \
+       | loongarch64 \
        | lm32 \
        | m32c | m32r | m32rle | m68000 | m68k | m88k \
        | maxq | mb | microblaze | microblazeel | mcore | mep | metag \
```

```
diff -Nuar ghc-8.4.4-orig/libraries/integer-gmp/config.sub ghc-
8.4.4/libraries/integer-gmp/config.sub
--- ghc-8.4.4-orig/libraries/integer-gmp/config.sub     2017-11-29
00:39:14.000000000 +0800
+++ ghc-8.4.4/libraries/integer-gmp/config.sub  2021-09-27 06:24:06.113684732
+0800
@@ -267,6 +267,7 @@
        | ip2k | iq2000 \
        | k1om \
        | le32 | le64 \
+       | loongarch64 \
        | lm32 \
        | m32c | m32r | m32rle | m68000 | m68k | m88k \
        | maxq | mb | microblaze | microblazeel | mcore | mep | metag \
diff -Nuar ghc-8.4.4-orig/libraries/unix/config.sub ghc-
8.4.4/libraries/unix/config.sub
--- ghc-8.4.4-orig/libraries/unix/config.sub    2017-11-29 00:40:34.000000000
+0800
+++ ghc-8.4.4/libraries/unix/config.sub 2021-09-27 06:24:32.801684732 +0800
@@ -267,6 +267,7 @@
        | ip2k | iq2000 \
        | k1om \
        | le32 | le64 \
+       | loongarch64 \
        | lm32 \
        | m32c | m32r | m32rle | m68000 | m68k | m88k \
        | maxq | mb | microblaze | microblazeel | mcore | mep | metag \
diff -Naur ghc-8.4.4-orig/compiler/utils/Platform.hs ghc-
8.4.4/compiler/utils/Platform.hs
--- ghc-8.4.4-orig/compiler/utils/Platform.hs   2017-11-29 00:39:14.000000000
+0800
+++ ghc-8.4.4/compiler/utils/Platform.hs        2021-09-27 07:54:46.133684732
+0800
@@ -61,6 +61,7 @@
           }
         | ArchARM64
         | ArchAlpha
+       | ArchLoongarch64
         | ArchMipseb
         | ArchMipsel
         | ArchJavaScript
```

- 应用龙芯的 libffi patch

```
cd $GHC_SRC/libffi-tarballs
cp $GHC_WORKDIR/0001-port-loongarch.diff ./
tar zxvf libffi-3.99999+git20171002+77e130c.tar.gz
cd libffi-3.99999/
patch -p1 < ../0001-port-loongarch.diff
cd ../
tar zcvf libffi-3.99999+git20171002+77e130c.tar.gz libffi-3.99999
```

- 增加 Debian-base 对 GHC 编译结果验证的 patch

```
diff -Nuar ghc-8.4.4-orig/docs/users_guide/flags.py ghc-
8.4.4/docs/users_guide/flags.py
```

```
--- ghc-8.4.4-orig/docs/users_guide/flags.py    2017-11-29 00:39:29.000000000
+0800
+++ ghc-8.4.4/docs/users_guide/flags.py 2021-09-27 06:33:09.129684732 +0800
@@ -46,9 +46,11 @@

 from docutils import nodes
 from docutils.parsers.rst import Directive, directives
+import sphinx
 from sphinx import addnodes
 from sphinx.domains.std import GenericObject
 from sphinx.errors import SphinxError
+from distutils.version import LooseVersion
 from utils import build_table_from_list

 ### Settings
@@ -597,14 +599,18 @@
 ### Initialization

 def setup(app):
+    # The override argument to add_directive_to_domain is only supported by
>= 1.8
+    sphinx_version = LooseVersion(sphinx.__version__)
+    override_arg = {'override': True} if sphinx_version >=
LooseVersion('1.8') else {}

     # Add ghc-flag directive, and override the class with our own
     app.add_object_type('ghc-flag', 'ghc-flag')
-    app.add_directive_to_domain('std', 'ghc-flag', Flag)
+    app.add_directive_to_domain('std', 'ghc-flag', Flag, **override_arg)

     # Add extension directive, and override the class with our own
     app.add_object_type('extension', 'extension')
-    app.add_directive_to_domain('std', 'extension', LanguageExtension)
+    app.add_directive_to_domain('std', 'extension', LanguageExtension,
+                                **override_arg)
     # NB: language-extension would be misinterpreted by sphinx, and produce
     # lang="extensions" XML attributes
```

4.1.3 Make

- 强制 Debian-base 使能交叉编译

```
diff -Nuar ghc-8.4.4-orig/utils/hsc2hs/ghc.mk ghc-8.4.4/utils/hsc2hs/ghc.mk
--- ghc-8.4.4-orig/utils/hsc2hs/ghc.mk  2017-11-29 00:40:35.000000000 +0800
+++ ghc-8.4.4/utils/hsc2hs/ghc.mk       2021-09-27 06:29:28.537684732 +0800
@@ -21,7 +21,7 @@
 # won't run on the host.
 ifeq "$(CrossCompiling)" "YES"
 # compile with stage 0 (bootstrap compiler)
-$(eval $(call build-prog,utils/hsc2hs,dist-install,0))
+$(eval $(call build-prog,utils/hsc2hs,dist-install,1))
 else
 $(eval $(call build-prog,utils/hsc2hs,dist-install,1))
 endif
```

- 交叉编译失能部分功能

```
diff -Nuar ghc-8.4.4-orig/mk/build.mk ghc-8.4.4/mk/build.mk
--- ghc-8.4.4-orig/mk/build.mk  1970-01-01 08:00:00.000000000 +0800
+++ ghc-8.4.4/mk/build.mk       2021-09-27 06:25:58.157684732 +0800
@@ -0,0 +1,116 @@
+# -----------------------------------------------------------------------------
---
+# A Sample build.mk
+#
+# Uncomment one of the following BuildFlavour settings to get the desired
+# overall build type.
+
+# -------- Build profiles ------------------------------------------------
---
+# Uncomment one of these to select a build profile below:
+
+# Full build with max optimisation and everything enabled (very slow build)
+#BuildFlavour = perf
+
+# As above but build GHC using the LLVM backend
+#BuildFlavour = perf-llvm
+
+# Perf build configured for a cross-compiler (using the LLVM backend)
+#BuildFlavour = perf-cross
+
+# Perf build configured for a cross-compiler (using the NCG backend)
+#BuildFlavour = perf-cross-ncg
+
+# Fast build with optimised libraries, no profiling (RECOMMENDED):
+#BuildFlavour = quick
+
+# Fast build with optimised libraries, no profiling, with LLVM:
+#BuildFlavour = quick-llvm
+
+# Fast build configured for a cross compiler (using the LLVM backend)
+#BuildFlavour = quick-cross
+
+# Fast build configured for a cross compiler (using the NCG backend)
+#BuildFlavour = quick-cross-ncg
+
+# Even faster build.  NOT RECOMMENDED: the libraries will be
+# completely unoptimised, so any code built with this compiler
+# (including stage2) will run very slowly, and many GHC tests
+# will fail with this profile (see Trac #12141):
+#BuildFlavour = quickest
+
+# Profile the stage2 compiler:
+#BuildFlavour = prof
+
+# Profile the stage2 compiler (LLVM backend):
+#BuildFlavour = prof-llvm
+
+# A development build, working on the stage 1 compiler:
+#BuildFlavour = devel1
+
+# A development build, working on the stage 2 compiler:
+#BuildFlavour = devel2
+
+# A build with max optimisation that still builds the stage2 compiler
```

```
+# quickly. Compiled code will be the same as with "perf". Programs
+# will compile more slowly.
+#BuildFlavour = bench
+
+# As above but build GHC using the LLVM backend
+#BuildFlavour = bench-llvm
+
+# Bench build configured for a cross-compiler (using the LLVM backend)
+#BuildFlavour = bench-cross
+
+# Bench build configured for a cross-compiler (using the NCG backend)
+#BuildFlavour = bench-cross-ncg
+
+# Use the same settings as validate.
+#BuildFlavour = validate
+
+ifneq "$(BuildFlavour)" ""
+include mk/flavours/$(BuildFlavour).mk
+endif
+
+# -------- Miscellaneous variables ----------------------------------------
---
+
+# Set to V = 0 to get prettier build output.
+# Please use V=1 (the default) when reporting GHC bugs.
+#V=0
+
+# Should all enabled warnings (see mk/warnings.mk) be turned into errors
while
+# building stage 2?
+#WERROR=-Werror
+
+# After stage 1 and the libraries have been built, you can uncomment this
line:
+#stage=2
+
+# Then stage 1 will not be touched by the build system, until
+# you comment the line again.  This is a useful trick for when you're
+# working on stage 2 and want to freeze stage 1 and the libraries for
+# a while.
+
+# Enable these if you would like DWARF debugging symbols for your libraries.
+# This is necessary, for instance, to get DWARF stack traces out of programs
+# built by the produced compiler. You must also pass --enable-dwarf-unwind
to
+# `configure` to enable the runtime system's builtin unwinding support.
+#GhcLibHcOpts += -g3
+#GhcRtsHcOpts += -g3
+
+# Build the "extra" packages (see ./packages). This enables more tests. See:
+#
https://ghc.haskell.org/trac/ghc/wiki/Building/RunningTests/Running#Additiona
lPackages
+#BUILD_EXTRA_PKGS=YES
+
+# Uncomment the following line to enable building DPH
+#BUILD_DPH=YES
+
```

```
+# Uncomment the following to force `integer-gmp` to use the in-tree GMP
6.1.2
+# (other sometimes useful configure-options: `--with-gmp-
{includes,libraries}`)
+#libraries/integer-gmp_CONFIGURE_OPTS += --configure-option=--with-intree-
gmp
+
+# Enable pretty hyperlinked sources
+#HADDOCK_DOCS = YES
+#EXTRA_HADDOCK_OPTS += --quickjump --hyperlinked-source
+
+# Don't strip debug and other unneeded symbols from libraries and
executables.
+STRIP_CMD = :
+HADDOCK_DOCS = NO
+WITH_TERMINFO=NO
```

- 开始第一阶段编译

```
cd $GHC_SRC
./configure --target=loongarch64-linux-gnu
make
```

4.1.4 Install

```
cd $GHC_WORKDIR
mkdir ghc-install-dir
export GHC_INSTDIR=$GHC_WORKDIR/ghc-install-dir
cd $GHC_SRC
make install DESTDIR=$GHC_INSTDIR
```

4.2 Deb化第一阶段

- 软硬件环境

硬件环境	Arch	Mem	Hdd
编译宿主机	x86_64	>= 8GB	>= 60GB

软件环境	OS	Compiler
编译宿主机	UnionTech OS 20	cross compiler gcc-8.3.0 (8.3.0-6-lnd.vec.23)

- 期望结果

  这里的一步操作是为了解决目标平台上源码编译 GHC 编译会出错的依赖问题，在 Debian-base 中对于 GHC 的包管理是分解为三个独立 package 的：ghc、ghc-prof、ghc-doc；本次将 GHC_INSTDIR 中的内容手动打包成这三个 deb 包。

```
cd $GHC_WORKDIR
mkdir deblization-stage1
cd deblization-stage1/
```

4.2.1 ghc

```
# 创建 ghc 目录, 下载 ghc 包
cd $GHC_WORKDIR/deblization-stage1/
mkdir ghc
cd ghc/
wget http://ftp.cn.debian.org/debian/pool/main/g/ghc/ghc_8.4.4+dfsg1-3_amd64.deb

# 解开 ghc 包
ar x ghc_8.4.4+dfsg1-3_amd64.deb
mkdir CONTROL DATA
mv control.tar.xz CONTROL
mv data.tar.xz DATA

# 先打包 control.tar.xz
cd CONTROL
tar xvf control.tar.xz
rm control.tar.xz
## 适配修改
diff -Naur control/control control-orig/control
--- control/control     2021-09-28 01:49:28.621684732 +0800
+++ control-orig/control        2021-09-28 01:48:29.481684732 +0800
@@ -1,6 +1,6 @@
 Package: ghc
 Version: 8.4.4+dfsg1-3
-Architecture: loongarch64
+Architecture: amd64
 Maintainer: Debian Haskell Group <pkg-haskell-
maintainers@lists.alioth.debian.org>
 Installed-Size: 733324
 Pre-Depends: dpkg (>= 1.16.1)
diff -Naur control/postinst control-orig/postinst
--- control/postinst    2021-09-28 01:50:14.253684732 +0800
+++ control-orig/postinst       2021-09-28 01:48:29.481684732 +0800
@@ -3,8 +3,8 @@
 #
 set -e

-execdir=/usr/local/bin
-libdir=/usr/local/lib/ghc
+execdir=/usr/bin
+libdir=/usr/lib/ghc
 bindir=$libdir/bin
 mandir=/usr/share/man
 vardir=/var/lib/ghc
diff -Naur control/prerm control-orig/prerm
--- control/prerm       2021-09-28 01:50:49.021684732 +0800
+++ control-orig/prerm  2021-09-28 01:48:29.481684732 +0800
@@ -5,8 +5,8 @@

 set -e

-execdir=/usr/local/bin
-libdir=/usr/local/lib/ghc
+execdir=/usr/bin
+libdir=/usr/lib/ghc
 bindir=$libdir/bin
```

```
 vardir=/var/lib/ghc
## 压缩
tar Jcvf ../control.tar.xz ./*

# 再打包 data.tar.xz
cd DATA
tar xvf data.tar.xz
rm data.tar.xz
## 移除 x86 架构 ghc
rm -rf usr
cp -r $GHC_INSTDIR/* ./
## 压缩
tar Jcvf ../data.tar.xz ./*

# 生成完整的 ghc 的 deb 包 (版本号保持一致)
cd $GHC_WORKDIR/deblization-stage1/ghc
ar r ghc_8.4.4+dfsg1-3_loongarch64.deb debian-binary control.tar.xz data.tar.xz
cp ghc_8.4.4+dfsg1-3_loongarch64.deb ../
```

4.2.2 ghc-prof

```
# 创建 ghc-prof 目录, 下载 ghc-prof 包
cd $GHC_WORKDIR/deblization-stage1/
mkdir ghc-prof
cd ghc-prof/
wget http://ftp.cn.debian.org/debian/pool/main/g/ghc/ghc-prof_8.4.4+dfsg1-
3_amd64.deb

# 解开 ghc-prof 包
ar x ghc-prof_8.4.4+dfsg1-3_amd64.deb
mkdir CONTROL DATA
mv control.tar.xz CONTROL
mv data.tar.xz DATA

# 先打包 control.tar.xz
cd CONTROL
tar xvf control.tar.xz
rm control.tar.xz
## 适配修改
diff -Naur control-orig/control control/control
--- control-orig/control        2021-09-28 03:29:41.773684732 +0800
+++ control/control     2021-09-28 03:30:07.309684732 +0800
@@ -1,7 +1,7 @@
 Package: ghc-prof
 Source: ghc
 Version: 8.4.4+dfsg1-3
-Architecture: amd64
+Architecture: loongarch64
 Maintainer: Debian Haskell Group <pkg-haskell-
maintainers@lists.alioth.debian.org>
 Installed-Size: 563001
 Depends: ghc (= 8.4.4+dfsg1-3)
## 压缩
tar Jcvf ../control.tar.xz ./*

# 再打包 data.tar.xz
cd DATA
```

```
tar xvf data.tar.xz (rm data.tar.xz)
rm data.tar.xz
## 移除 x86 架构 ghc-prof
rm -rf ./usr/lib
## 压缩
tar Jcvf ../data.tar.xz ./*

# 生成完整的 ghc-prof 的 deb 包 (版本号保持一致)
cd $GHC_WORKDIR/deblization-stage1/ghc-prof
ar r ghc-prof_8.4.4+dfsg1-3_loongarch64.deb debian-binary control.tar.xz
data.tar.xz
cp ghc-prof_8.4.4+dfsg1-3_loongarch64.deb ../
```

4.2.3 ghc-doc

```
# 创建 ghc-doc目录, 下载 ghc-doc 包
cd $GHC_WORKDIR/deblization-stage1/
mkdir ghc-doc
cd ghc-doc/
wget http://ftp.cn.debian.org/debian/pool/main/g/ghc/ghc-doc_8.4.4+dfsg1-
3_all.deb
cp ghc-doc_8.4.4+dfsg1-3_all.deb ../
```

4.3. 目标机直接编译

- 软硬件环境

硬件环境	Arch	Mem	Hdd
目标平台机	loongarch64	>= 8GB	>= 60GB

软件环境	OS	Compiler
目标平台机	UnionTech OS 20 (with deblization-stage1's ghc)	gcc-8.3.0 (8.3.0-6-lnd.vec.23)

- 期望结果

  在 4.2 中在 `$GHC_WORKDIR/deblization-stage1/` 下得到 三个 deb 包，将这三个包拷贝到未安装 GHC 的目标机上，通过 dpkg 安装它们，然后在目标机器上进行直接编译 GHC，这一步操作是为了加入宿主机交叉编译时缺失的 haddoc 二进制文件。

4.3.1 Prepare

```
# Loongnix GNU/Linux 20 Beta9 目标机环境
export T_GHC_WORKDIR = "工作路径" # 目标机下的 GHC 工作路径
echo $T_GHC_WORKDIR

# 该路径下的资源
uos@uos-VB:~/Workspace$ tree -a
.
├── 0001-port-loongarch.diff # 由龙芯提供
└── loongarch64-linux-gnu-2021-06-19.tar.gz # 由龙芯提供
```

```
0 directories, 2 files

# 获取 GHC 源码
cd $T_GHC_WORKDIR
wget https://downloads.haskell.org/~ghc/8.4.4/ghc-8.4.4-src.tar.xz
tar xvf ghc-8.4.4-src.tar.xz
export T_GHC_SRC=$T_GHC_WORKDIR/ghc-8.4.4
```

4.3.2 Config

- 增加对 loongarch64 的支持

```
cd $T_GHC_SRC/
# 按照下列 patch 内容修改 GHC 源码

diff -Naur ghc-8.4.4-orig/config.guess ghc-8.4.4/config.guess
--- ghc-8.4.4-orig/config.guess 2017-11-29 00:39:14.000000000 +0800
+++ ghc-8.4.4/config.guess      2021-09-28 13:28:02.977398451 +0800
@@ -976,6 +976,9 @@
     k1om:Linux:*:*)
        echo ${UNAME_MACHINE}-unknown-linux-${LIBC}
        exit ;;
+    loongarch64:Linux:*:*)
+       echo ${UNAME_MACHINE}-unknown-linux-${LIBC}
+       exit ;;
     m32r*:Linux:*:*)
        echo ${UNAME_MACHINE}-unknown-linux-${LIBC}
        exit ;;
diff -Naur ghc-8.4.4-orig/configure ghc-8.4.4/configure
--- ghc-8.4.4-orig/configure    2018-10-14 03:54:34.000000000 +0800
+++ ghc-8.4.4/configure 2021-09-28 13:27:35.981295358 +0800
@@ -9630,6 +9630,9 @@
         alpha)
             test -z "$2" || eval "$2=ArchAlpha"
             ;;
+        loongarch64)
+            test -z "$2" || eval "$2=ArchLoongarch64"
+            ;;
         mips|mipseb)
             test -z "$2" || eval "$2=ArchMipseb"
             ;;
diff -Naur ghc-8.4.4-orig/libraries/base/config.guess ghc-
8.4.4/libraries/base/config.guess
--- ghc-8.4.4-orig/libraries/base/config.guess  2017-11-29 00:39:14.000000000
+0800
+++ ghc-8.4.4/libraries/base/config.guess       2021-09-28 15:31:24.349568841
+0800
@@ -976,6 +976,9 @@
     k1om:Linux:*:*)
        echo ${UNAME_MACHINE}-unknown-linux-${LIBC}
        exit ;;
+    loongarch64:Linux:*:*)
+       echo ${UNAME_MACHINE}-unknown-linux-${LIBC}
+       exit ;;
     m32r*:Linux:*:*)
        echo ${UNAME_MACHINE}-unknown-linux-${LIBC}
        exit ;;
```

```
diff -Naur ghc-8.4.4-orig/libraries/integer-gmp/config.guess ghc-
8.4.4/libraries/integer-gmp/config.guess
--- ghc-8.4.4-orig/libraries/integer-gmp/config.guess   2017-11-29
00:39:14.000000000 +0800
+++ ghc-8.4.4/libraries/integer-gmp/config.guess        2021-09-28
15:30:52.925451346 +0800
@@ -976,6 +976,9 @@
     k1om:Linux:*:*)
        echo ${UNAME_MACHINE}-unknown-linux-${LIBC}
        exit ;;
+    loongarch64:Linux:*:*)
+       echo ${UNAME_MACHINE}-unknown-linux-${LIBC}
+       exit ;;
     m32r*:Linux:*:*)
        echo ${UNAME_MACHINE}-unknown-linux-${LIBC}
        exit ;;
diff -Naur ghc-8.4.4-orig/libraries/unix/config.guess ghc-
8.4.4/libraries/unix/config.guess
--- ghc-8.4.4-orig/libraries/unix/config.guess  2017-11-29 00:40:34.000000000
+0800
+++ ghc-8.4.4/libraries/unix/config.guess       2021-09-28 15:31:52.557674329
+0800
@@ -976,6 +976,9 @@
     k1om:Linux:*:*)
        echo ${UNAME_MACHINE}-unknown-linux-${LIBC}
        exit ;;
+    loongarch64:Linux:*:*)
+       echo ${UNAME_MACHINE}-unknown-linux-${LIBC}
+       exit ;;
     m32r*:Linux:*:*)
        echo ${UNAME_MACHINE}-unknown-linux-${LIBC}
        exit ;;
diff -Naur ghc-8.4.4-orig/compiler/utils/Platform.hs ghc-
8.4.4/compiler/utils/Platform.hs
--- ghc-8.4.4-orig/compiler/utils/Platform.hs   2017-11-29 00:39:14.000000000
+0800
+++ ghc-8.4.4/compiler/utils/Platform.hs        2021-09-28 15:33:52.358122515
+0800
@@ -61,6 +61,7 @@
           }
         | ArchARM64
         | ArchAlpha
+        | ArchLoongarch64
         | ArchMipseb
         | ArchMipsel
         | ArchJavaScript
```

- 应用龙芯 libffi patch

```
cd $T_GHC_SRC/libffi-tarballs
cp $T_GHC_WORKDIR/0001-port-loongarch.diff ./
tar zxvf libffi-3.99999+git20171002+77e130c.tar.gz
cd libffi-3.99999/
patch -p1 < ../0001-port-loongarch.diff
cd ../
tar zcvf libffi-3.99999+git20171002+77e130c.tar.gz libffi-3.99999
```

- 增加 Debian-base 对编译结果验证的 patch

```
cd $T_GHC_SRC/
# 按照下列 patch 内容修改 GHC 源码

diff -Nuar ghc-8.4.4-orig/docs/users_guide/flags.py ghc-
8.4.4/docs/users_guide/flags.py
--- ghc-8.4.4-orig/docs/users_guide/flags.py    2017-11-29 00:39:29.000000000
+0800
+++ ghc-8.4.4/docs/users_guide/flags.py 2021-09-27 06:33:09.129684732 +0800
@@ -46,9 +46,11 @@

 from docutils import nodes
 from docutils.parsers.rst import Directive, directives
+import sphinx
 from sphinx import addnodes
 from sphinx.domains.std import GenericObject
 from sphinx.errors import SphinxError
+from distutils.version import LooseVersion
 from utils import build_table_from_list

 ### Settings
@@ -597,14 +599,18 @@
 ### Initialization

 def setup(app):
+    # The override argument to add_directive_to_domain is only supported by
>= 1.8
+    sphinx_version = LooseVersion(sphinx.__version__)
+    override_arg = {'override': True} if sphinx_version >=
LooseVersion('1.8') else {}

     # Add ghc-flag directive, and override the class with our own
     app.add_object_type('ghc-flag', 'ghc-flag')
-    app.add_directive_to_domain('std', 'ghc-flag', Flag)
+    app.add_directive_to_domain('std', 'ghc-flag', Flag, **override_arg)

     # Add extension directive, and override the class with our own
     app.add_object_type('extension', 'extension')
-    app.add_directive_to_domain('std', 'extension', LanguageExtension)
+    app.add_directive_to_domain('std', 'extension', LanguageExtension,
+                                **override_arg)
     # NB: language-extension would be misinterpreted by sphinx, and produce
     # lang="extensions" XML attributes
```

4.3.3 Make

- 开始第三阶段编译

```
cd $T_GHC_SRC
./configure
make
```

4.3.4 Install

```
cd $T_GHC_WORKDIR
mkdir t-ghc-install-dir
export T_GHC_INSTDIR=$T_GHC_WORKDIR/t-ghc-install-dir
cd $T_GHC_SRC
make install DESTDIR=$T_GHC_INSTDIR
```

4.4. Deb化第二阶段

- 软硬件环境

硬件环境	Arch	Mem	Hdd
目标平台机	loongarch64	>= 8GB	>= 60GB

软件环境	OS	Compiler
目标平台机	UnionTech OS 20 (with deblization-stage1's ghc)	gcc-8.3.0 (8.3.0-6-lnd.vec.23)

- 期望结果

  使用阶段三中包含 haddock 的 GHC 编译结果，按照 4.2 中同样的 Deb 化思想，分拆 deb 包。

```
cd $T_GHC_WORKDIR
mkdir deblization-stage2
cd deblization-stage2/
```

4.4.1 ghc

```
# 创建 ghc 目录，下载 ghc 包
cd $T_GHC_WORKDIR/deblization-stage2/
mkdir ghc
cd ghc/
wget http://ftp.cn.debian.org/debian/pool/main/g/ghc/ghc_8.4.4+dfsg1-3_amd64.deb

# 解包 ghc
ar x ghc_8.4.4+dfsg1-3_amd64.deb
mkdir CONTROL DATA
mv control.tar.xz CONTROL
mv data.tar.xz DATA

# 先打包 control.tar.xz
cd CONTROL
tar xvf control.tar.xz
rm control.tar.xz
## 适配修改
diff -Naur control/control control-orig/control
--- control/control     2021-09-28 01:49:28.621684732 +0800
+++ control-orig/control        2021-09-28 01:48:29.481684732 +0800
@@ -1,6 +1,6 @@
 Package: ghc
 Version: 8.4.4+dfsg1-3
-Architecture: loongarch64
```

```
+Architecture: amd64
 Maintainer: Debian Haskell Group <pkg-haskell-
maintainers@lists.alioth.debian.org>
 Installed-Size: 733324
 Pre-Depends: dpkg (>= 1.16.1)
diff -Naur control/postinst control-orig/postinst
--- control/postinst    2021-09-28 01:50:14.253684732 +0800
+++ control-orig/postinst       2021-09-28 01:48:29.481684732 +0800
@@ -3,8 +3,8 @@
 #
 set -e

-execdir=/usr/local/bin
-libdir=/usr/local/lib/ghc
+execdir=/usr/bin
+libdir=/usr/lib/ghc
 bindir=$libdir/bin
 mandir=/usr/share/man
 vardir=/var/lib/ghc
diff -Naur control/prerm control-orig/prerm
--- control/prerm       2021-09-28 01:50:49.021684732 +0800
+++ control-orig/prerm  2021-09-28 01:48:29.481684732 +0800
@@ -5,8 +5,8 @@

 set -e

-execdir=/usr/local/bin
-libdir=/usr/local/lib/ghc
+execdir=/usr/bin
+libdir=/usr/lib/ghc
 bindir=$libdir/bin
 vardir=/var/lib/ghc
## 压缩
tar Jcvf ../control.tar.xz ./*

# 再打包 data.tar.xz
cd DATA
tar xvf data.tar.xz
rm data.tar.xz
## 移除 x86 架构 ghc
rm -rf usr/
cp -r $T_GHC_INSTDIR/* ./
## 压缩
tar Jcvf ../data.tar.xz ./*

# 生成完整的 ghc 的 deb 包 (版本号保持一致)
cd $T_GHC_WORKDIR/deblization-stage2/ghc
ar r ghc_8.4.4+dfsg1-3_loongarch64.deb debian-binary control.tar.xz data.tar.xz
cp ghc_8.4.4+dfsg1-3_loongarch64.deb ../
```

4.4.2 ghc-prof

```
# 创建 ghc-prof 目录，下载 ghc-prof 包
cd $T_GHC_WORKDIR/deblization-stage2/
mkdir ghc-prof
cd ghc-prof/
```

```
wget http://ftp.cn.debian.org/debian/pool/main/g/ghc/ghc-prof_8.4.4+dfsg1-
3_amd64.deb

# 解包 ghc-prof
ar x ghc-prof_8.4.4+dfsg1-3_amd64.deb
mkdir CONTROL DATA
mv control.tar.xz CONTROL
mv data.tar.xz DATA

# 先打包 control.tar.xz
cd CONTROL
tar xvf control.tar.xz
rm control.tar.xz
## 适配修改
diff -Naur control-orig/control control/control
--- control-orig/control        2021-09-28 03:29:41.773684732 +0800
+++ control/control     2021-09-28 03:30:07.309684732 +0800
@@ -1,7 +1,7 @@
 Package: ghc-prof
 Source: ghc
 Version: 8.4.4+dfsg1-3
-Architecture: amd64
+Architecture: loongarch64
 Maintainer: Debian Haskell Group <pkg-haskell-
maintainers@lists.alioth.debian.org>
 Installed-Size: 563001
 Depends: ghc (= 8.4.4+dfsg1-3)
## 压缩
tar Jcvf ../control.tar.xz ./*

# 再打包 data.tar.xz
cd DATA
tar xvf data.tar.xz
rm data.tar.xz
## 移除 x86 架构 ghc-prof
rm -rf ./usr/lib
## 压缩
tar Jcvf ../data.tar.xz ./*

# 生成完整的 ghc-prof 的 deb 包 (版本号保持一致)
cd $T_GHC_WORKDIR/deblization-stage2/ghc-prof
ar r ghc-prof_8.4.4+dfsg1-3_loongarch64.deb debian-binary control.tar.xz
data.tar.xz
cp ghc-prof_8.4.4+dfsg1-3_loongarch64.deb ../
```

4.4.3 ghc-doc

```
# 创建 ghc-doc 目录, 下载 ghc-doc 包
cd $T_GHC_WORKDIR/deblization-stage2/
mkdir ghc-doc
cd ghc-doc/
wget http://ftp.cn.debian.org/debian/pool/main/g/ghc/ghc-doc_8.4.4+dfsg1-
3_all.deb
cp ghc-doc_8.4.4+dfsg1-3_all.deb ../
```

4.5. DPKG自动编译

- 软硬件环境

硬件环境	Arch	Mem	Hdd
目标平台机	loongarch64	>= 8GB	>= 60GB

软件环境	OS	Compiler
目标平台机	UnionTech OS 20 (with deblization-stage2's ghc)	gcc-8.3.0 (8.3.0-6-lnd.vec.23)

- 期望结果

  到现在为止，目标平台上已经存在了可以使用 GHC, 但是这时的 GHC 还不是一个完全符合 Debian DFSG 规范的 GHC 发行包，因此要从 Debian 仓库获取 GHC 的源码来编译，这一步是非常必要的。

```
cd $T_GHC_WORKDIR/
mkdir dfsg-ghc
cd dfsg-ghc
```

4.5.1 获取源码

```
wget http://deb.debian.org/debian/pool/main/g/ghc/ghc_8.4.4+dfsg1-3.dsc
wget http://deb.debian.org/debian/pool/main/g/ghc/ghc_8.4.4+dfsg1-3.debian.tar.xz
wget http://deb.debian.org/debian/pool/main/g/ghc/ghc_8.4.4+dfsg1.orig.tar.xz

dpkg-source -x ghc_8.4.4+dfsg1-3.dsc
cd ghc-8.4.4+dfsg1
```

4.5.2 移植修改

- 基本移植（必选）

```
diff -Naur ghc-8.4.4+dfsg1-orig/aclocal.m4 ghc-8.4.4+dfsg1/aclocal.m4
--- ghc-8.4.4+dfsg1-orig/aclocal.m4     2021-09-29 16:54:50.957429444 +0800
+++ ghc-8.4.4+dfsg1/aclocal.m4  2021-09-30 10:09:31.755150184 +0800
@@ -211,6 +211,9 @@
         alpha)
             test -z "[$]2" || eval "[$]2=ArchAlpha"
             ;;
+        loongarch64)
+            test -z "[$]2" || eval "[$]2=ArchLoongarch64"
+            ;;
         mips|mipseb)
             test -z "[$]2" || eval "[$]2=ArchMipseb"
             ;;
@@ -1842,6 +1845,9 @@
   ia64)
     $2="ia64"
     ;;
+  loongarch64)
+    $2="loongarch64"
+    ;;
```

```
   m68k*)
     $2="m68k"
     ;;
diff -Naur ghc-8.4.4+dfsg1-orig/compiler/utils/Platform.hs ghc-
8.4.4+dfsg1/compiler/utils/Platform.hs
--- ghc-8.4.4+dfsg1-orig/compiler/utils/Platform.hs     2021-09-29
16:54:50.725428450 +0800
+++ ghc-8.4.4+dfsg1/compiler/utils/Platform.hs  2021-09-30 11:28:51.600879070
+0800
@@ -61,6 +61,7 @@
           }
         | ArchARM64
         | ArchAlpha
+        | ArchLoongarch64
         | ArchMipseb
         | ArchMipsel
         | ArchJavaScript
```

- 进阶移植（可选）

```
compiler/codeGen/CodeGen/Platform/ # 目录下为架构平台相关的信息
compiler/codeGen/CodeGen/Platform.hs # GHC 平台信息
compiler/nativeGen/AsmCodeGen.hs # 架构优化的汇编指令
compiler/nativeGen/TargetReg.hs # 寄存器映射

include/CodeGen.Platform.hs # 代码生成器支持新架构
include/rts
include/rts/StgCRun.c
include/rts/Linker.c
include/stg
include/stg/HaskellMachRegs.h
include/stg/MachRegs.h #寄存器映射，需要芯片手册
include/stg/RtsMachRegs.h
```

注意：该文档仅涉及基本移植，进阶移植的部分由于种种原因不便编写。

4.5.3 编译

```
fakeroot dpkg-buildpackage -us -uc -nc
# 得到最终版本的 ghc,ghc-prof,ghc-doc.(建议在完全没有 GHC 装过的环境下安装)。
```

4.6 验证编译结果

4.6.1 实例验证

- 读取版本

```
ghc --version
The Glorious Glasgow Haskell Compilation System, version 8.4.4
```

- 程序示例

```
touch Main.hs
```

```
module Main where
main = putStrLn "Hello, world!"
```

```
ghc -o Main Main.hs
./Main
Hello, world!
```

4.6.2 编译Pandoc

Pandoc 顺利编出，用户显示协议正常。

五 、小结

本次的 GHC 移植 输入的部分只有一个 GCC 和 OS 编译环境，因此所得到的 GHC 高度依赖于 GCC 编译器 和 OS 的实现，GHC 团队对平台相关部分做了比较好的分离，所以在移植过程中就算没有直接添加底层特性的部分，GHC 也会保证上层应用可以编译通过，但尽管如此，也会有一些这样或者那样的的问题存在，比如无法很好的使用 CPU 架构的特性，某些硬件优化，算力无法得到提升，这对于 GHC 本身偏重优化的定位，会产生一些硬限制，如果遇到一些底层错误需要重新编译 GHC 等，但我们使用合编译 GHC 仅仅是为了编译 Pandoc，所以在这种情况下已经足够使用。如果为了得到最终的“全功能 GHC“，让龙芯团队跟 GHC 官方团队一起来做，从最底层修改，这才是根本的解决之道。

六. 参考资料

1. GHC 官网 (https://www.haskell.org/ghc/)。
2. GHC 交叉编译 (https://gitlab.haskell.org/ghc/ghc/-/wikis/cross-compilation)。
3. Debian GHC 源码包 (https://packages.debian.org/buster/ghc)
4. Haskell 编程语言 (https://wiki.haskell.org/Language_and_library_specification)。
5. GHC 下载地址 (https://downloads.haskell.org/~ghc/latest/docs/html/users_guide/)。