毕昇编译器
2.1.0
Autotuner 特性指南

文档版本	01
发布日期	2021-12-30

华为技术有限公司

商标声明


和其他华为商标均为华为技术有限公司的商标。
本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

华为技术有限公司

地址： 深圳市龙岗区坂田华为总部办公楼 邮编：518129

网址： https://www.huawei.com

客户服务邮箱： support@huawei.com

客户服务电话： 4008302118

目 录

1 概述

1.1 概念介绍

自动调优

一种自动化的迭代过程，通过操作编译选项来优化给定程序，以实现最佳性能。它由两个组件配合完成，毕昇编译器和Autotuner命令行工具。

毕昇编译器

带有自动调优特性的编译器，配合Autotuner可以更细粒度地控制优化。

Autotuner

一个命令行工具，需要与毕昇编译器一起使用。它管理搜索空间的生成和参数操作，并驱动整个调优过程。

1.2 毕昇编译器的职责和功能

自动调优作为毕昇编译器的特性之一，它可以更细粒度地控制编译器的优化。此功能不需要在源代码中注入pragma，而是允许用户在简单的YAML文件中指定优化配置，该文件包含优化信息及其相应的代码结构信息，包括名称和行号。此外，它还可以记录优化结果，生成包含可调优代码结构的列表（tuning opportunities）并以YAML的形式导出。

目的和用途

- 使编译过程更加灵活和可控；

- 细粒度编译控制，提供更多优化机会。

主要功能

- 读取与每个代码区域对应的编译配置；
- 输出可调优代码结构，即目标程序中哪些结构可以用来调优。

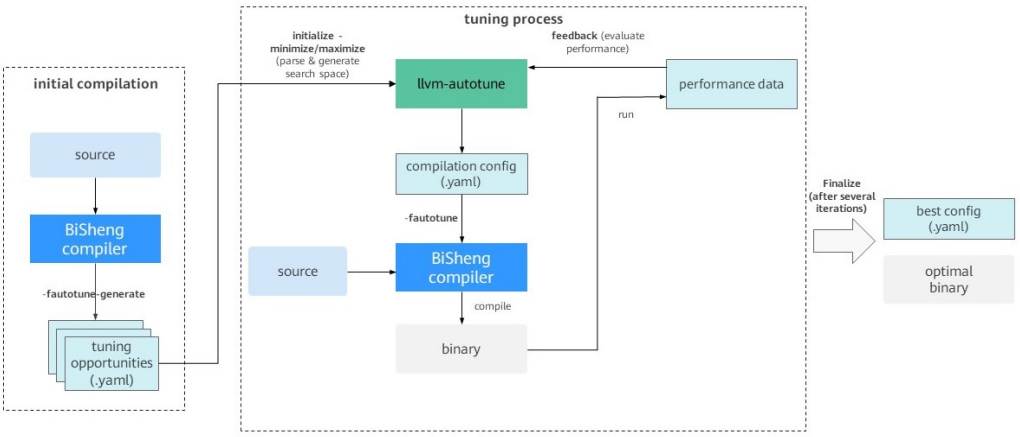
1.3 Autotuner 主要职责和功能

- 与毕昇编译器进行交互：
  - 根据编译器产生的可调优代码结构创建搜索空间（search space）；
  - 生成编译配置并调用编译器来编译源代码；
- 操作调优参数以及应用搜索算法；
  - 自带的遗传算法；
- 获取性能数据。

1.4 Autotuner 调优流程

调优流程（如图一所示）由两个阶段组成：初始编译阶段 （initial compilation）和调优阶段 （tuning process）。

图 1-1 Autotuner 调优流程



初始编译阶段

初始编译阶段发生在调优开始之前，Autotuner首先会让编译器对目标程序代码做一次编译，在编译的过程中，毕昇编译器会生成一些包含所有可调优结构的YAML文件, 告诉我们在这个目标程序中哪些结构可以用来调优，比如文件（module），函数（function），循环（loop）。 例如，循环展开是编译器中最常见的优化方法之一，它通过多次复制**循环体**代码，达到增大指令调度的空间，减少循环分支指令的开销等优化效果。若以循环展开次数（unroll factor）为对象进行调优，编译器会在YAML文件中生成所有可被循环展开的循环作为可调优结构。

调优阶段

当可调优结构顺利生成之后，调优阶段便会开始：

1. Autotuner首先读取生成好的可调优结构的YAML 文件，从而产生对应的搜索空间，也就是生成针对每个可调优代码结构的具体的参数和范围；
2. 调优阶段会根据设定的搜索算法尝试一组参数的值，生成一个YAML格式的编译配置文件（compilation config），从而让编译器编译目标程序代码产生二进制文件；
3. 最后Autotuner将编译好的文件以用户定义的方式运行并取得性能信息作为反馈；
4. 经过一定数量的迭代之后，Autotuner将找出最终的最优配置，生成最优编译配置文件，以YAML的形式储存。

2 快速入门

2.1 获取 Autotuner

Autotuner 已经包括在了毕昇编译器的发布软件包里。 可以在以下目录下找到：**bisheng-compiler-2.1.0-aarch64-linux/lib/autotuner。**

2.2 环境要求

必选：

- 操作系统：openEuler21.03、openEuler 20.03 (LTS)、CentOS 7.6、Ubuntu 18.04、Ubuntu 20、麒麟V10、UOS 20
- 架构：AArch64
- Python 3.8.5
- SQLite 3.0

可选：

- LibYAML（ 推荐安装，可提升Auotuner文件解析速度 ）

2.3 安装 Autotuner

Autotuner 已包括在毕昇编译器的发布软件包里。 若您已经安装毕昇编译器，只需配置毕昇编译器的环境变量即可直接使用。否则，请先安装毕昇编译器。

- 配置毕昇编译器的环境变量
  ```
  export PATH=/opt/compiler/bisheng-compiler-2.1.0-aarch64-linux/bin:$PATH
  ```


须知

以上步骤是以/opt/compiler目录举例，若您的安装目录不同，请以实际目录为准。

- 测试是否安装成功

  执行如下命令：

```
install-autotuner.sh
llvm-autotune -h
auto-tuner -h
```

  执行完毕后，界面如果显示相应的帮助信息则表示安装成功。

须知

如果运行过程中出现错误，请确保您的系统满足**2.2 环境要求**。

例如：

```
bad magic number in 'autotuner': b'U\r\r\n'
```

请确保您的 python3 版本的为3.8.2, 安装路径存在于PATH中。请输入 'python3 -V' 命令检查python3版本。

```
No module named '_sqlite3'
```

请确保已安装 SQLite 3.0。

2.4 运行 Autotuner

2.4.1 运行方式说明

Autotuner 目前有两种使用方式，并对应两种不同的命令行工具llvm-autotune和auto-tuner。

- llvm-auotune 采用让用户主导调优过程的方式，提供辅助功能与编译器合作使用。相比auto-tuner繁琐的配置流程，极大简化了配置调优的步骤，可开箱即用，因而更推荐使用。
- auto-tuner 为传统调优方式，主导管理整个调优过程。用户需要先适配好配置文件来设定调优过程中的各种细节，其中包括如何编译、运行、获取性能信息以及可调参数等。

我们将以coremark为示例展示如何运行自动调优，毕昇编译器的发布包里没有自带coremark, 请从社区获取**coremark**。更多详细用法，请参阅**4 使用方法**章节。

2.4.2 llvm-autotune

用户可根据自身需求，编写调优脚本。我们将以coremark为示例展示如何运行自动调优，毕昇编译器的发布包里没有自带coremark, 请从社区获取**coremark**。以下为以20次迭代调优coremark的脚本示例：

```
export AUTOTUNE_DATADIR=/tmp/autotuner_data/
CompileCommand="clang -Ilinux64 -I. -g -DFLAGS_STR=\"\"  -DITERATIONS=300000  core_list_join.c core_main.c core_matrix.c core_state.c core_util.c linux64/core_portme.c -O2 -o coremark"

$CompileCommand -fautotune-generate;
```

```
llvm-autotune minimize;
for i in $(seq 20)
do
  $CompileCommand -fautotune ;
  time=`/usr/bin/time -p ./coremark  0x0 0x0 0x66 300000  2>&1 1>/dev/null | grep real | awk '{print $2}'`;
  echo "iteration: " $i "cost time:" $time;
  llvm-autotune feedback $time;
done
llvm-autotune finalize;
```

以下为分步说明：

步骤1 配置环境变量

使用环境变量**AUTOTUNE_DATADIR**指定调优相关的数据的存放位置。

```
export AUTOTUNE_DATADIR=/tmp/autotuner_data/
```

步骤2 初始编译步骤

添加毕昇编译器选项**-fautotune-generate**，编译生成可调优代码结构。

```
cd  examples/coremark/
clang -Ilinux64 -I. -DFLAGS_STR=\"\"  -lrt"\" -DITERATIONS=300000 core_list_join.c core_main.c
core_matrix.c core_state.c core_util.c linux64/core_portme.c -O2 -g -o coremark -fautotune-generate
```

须知

建议仅将此选项应用于需要重点调优的热点代码文件。若应用的代码文件过多（超过500个文件），则会生成数量庞大的可调优代码结构的文件，进而可能导致步骤3的初始化时间长（可长达数分钟）；以及巨大的搜索空间导致的调优效果不显著，收敛时间长等问题。

步骤3 初始化调优

运行**llvm-autotune**命令，初始化调优任务。生成最初的编译配置供下一次编译使用。

```
llvm-autotune minimize
```

minimize表示调优目标，旨在最小化指标（例如程序运行时间）。也可使用maximize，旨在最大化指标（例如程序吞吐量）。

步骤4 调优编译步骤

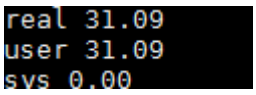
添加毕昇编译器选项**-fautotune**，读取当前**AUTOTUNE_DATADIR**配置并编译。

```
clang -Ilinux64 -I. -DFLAGS_STR=\"\"  -lrt"\" -DITERATIONS=300000 core_list_join.c core_main.c
core_matrix.c core_state.c core_util.c linux64/core_portme.c -O2 -g -o coremark -fautotune
```

步骤5 性能反馈

用户运行程序，并根据自身需求获取性能数字，使用**llvm-autotune feedback**反馈。例如，如果我们想以coremark运行速度为指标进行调优，可以采用如下方式：

```
time -p ./coremark  0x0 0x0 0x66 300000  2>&1 1>/dev/null
```




```
llvm-autotune feedback 31.09
```


须知

建议在使用 llvm-autotune feedback 之前， 先验证步骤4编译是否正常，及编译好的程序是否运行正确。若出现编译或者运行异常的情况，请输入相应调优目标的最差值（例如, 调优目标为minimize, 可输入llvm-autotune feedback 9999；maximize 可输入 0 或者 -9999 ）。

若输入的性能反馈不正确，可能会影响最终调优的结果。

步骤6 调优迭代

根据用户设定的迭代次数,重复步骤4和5进行调优迭代。

步骤7 结束调优

进行多次迭代后，用户可选择终止调优，并保存最优的配置文件。配置文件会被保存在环境变量**AUTOTUNE_DATADIR**指定的目录下。

```
llvm-autotune finalize
```

步骤8 最终编译

使用步骤7得到最优配置文件，进行最后编译。在环境变量未改变的情况下，可直接使用**-fautotune**选项：

```
clang -Ilinux64 -I. -DFLAGS_STR="\"  -lrt"\" -DITERATIONS=300000 core_list_join.c core_main.c core_matrix.c core_state.c core_util.c linux64/core_portme.c -O2 -g -o coremark -fautotune
```

或者使用 -mllvm -auto-tuning-input= 直接指向配置文件。

```
clang -Ilinux64 -I. -DFLAGS_STR="\"  -lrt"\" -DITERATIONS=300000 core_list_join.c core_main.c core_matrix.c core_state.c core_util.c linux64/core_portme.c -O2 -g -o coremark -mllvm -auto-tuning-input=/tmp/autotuner_data/config.yaml
```

----结束

2.5 卸载 Autotuner

编辑环境变量中的"PATH"，删除新增毕昇编译器的路径"/opt/compiler/bisheng-compiler-2.1.0-aarch64-linux/bin"。

3 准备工作

步骤1 请先安装 Autotuner。 更多信息请看**《 快速入门 》**。

步骤2 Autotuner 必须与支持调优的编译器配套使用。

在运行Autotuner之前，请先确认编译器的环境变量是否设置正确（ 或者可以尝试把环境变量放在配置文件中，**《 使用方法 》**章节会做详细的介绍 ）。

----结束

4 使用方法

4.1 llvm-autotune

4.1.1 工具简介

Autotuner 目前有两种使用方式，并对应两种不同的命令行工具llvm-autotune 和 auto-tuner。

llvm-auotune 采用让用户主导调优过程的方式，提供辅助功能与编译器合作使用。相比auto-tuner繁琐的配置流程，极大简化了配置调优的步骤，可开箱即用，因而更推荐使用。

4.1.2 帮助信息

帮助命令：**llvm-autotune -h**。llvm-autotune 执行格式如下所示：

```
llvm-autotune [-h] {minimize,maximize,feedback,dump,finalize}
```

可选指令：

- minimize：初始化调优并生成初始的编译器配置文件，旨在最小化指标（例如运行时间)。
- maximize：初始化调优并生成初始的编译器配置文件，旨在最大程度地提高指标（例如吞吐量）。
- feedback：反馈性能调优结果并生成新的编译器配置。
- dump：生成当前的最优配置，而不终止调优（可继续执行feedback）。
- finalize: 终止调优，并生成最佳的编译器配置（不可再执行feedback）。

帮助信息：

- --help/-h

```
usage: llvm-autotune [-h] {minimize,maximize,feedback,dump,finalize} ...

positional arguments:
  {minimize,maximize,feedback,dump,finalize}
    minimize            Initialize tuning and generate the initial compiler
```



```
                configuration file, aiming to minimize the metric
                (e.g. run time)
  maximize        Initialize tuning and generate the initial compiler
                configuration file, aiming to maximize the metric
                (e.g. throughput)
  feedback        Feed back performance tuning result and generate a new
                test configuration
  dump          Dump the current best configuration without
                terminating the tuning run
  finalize      Finalize tuning and generate the optimal compiler
                configuration

optional arguments:
  -h, --help          show this help message and exit
```

4.1.3 编译器相关选项

llvm-auotune 需要与毕昇编译器选项**-fautotune-generate**和**-fautotune**配合使用。

- **-fautotune-generate**：
  - 在“autotune_datadir”目录下生成可调优的代码结构列表，此默认目录可由环境变量**AUTOTUNE_DATADIR**改写；
  - 作为调优准备工作的第一步，通常需要在**llvm-autotune minimize/maximize**命令执行前使用；
  - 此选项还可以赋值来改变调优的细颗粒度（可选值为Other, Function, Loop, MachineBasicBlock）。例如**-fautotune-generate=Function**会开启类型仅为函数的可调优代码结构，每个函数在调优过程中会被赋予不同的参数值；而Other表示全局，生成的可调优代码结构对应每个编译单元（代码文件）。

    -fautotune-generate默认等效于**-fautotune-generate=Function,Loop**。通常建议使用默认值。
- **-fautotune**：
  - 使用“autotune_datadir”下的编译器配置进行调优编译（此默认目录可由环境变量**AUTOTUNE_DATADIR**改写）；
  - 通常在调优迭代过程中，**llvm-autotune minimize/maximize/feedback**命令之后使用。

说明

具体示例，请参阅**2.4.2-llvm-autotune运行方式说明** 章节。

5 附录

5.1 问题反馈

在使用过程中遇到问题，需要技术支持时，请反馈问题信息至**毕昇论坛**。

5.2 修订记录

发布日期	修订记录
2021-12-30	第一次正式发布。文档内容更新如下： 新增llvm-autotune工具的依赖安装脚本描述 删除auto-tuner使用方法的描述