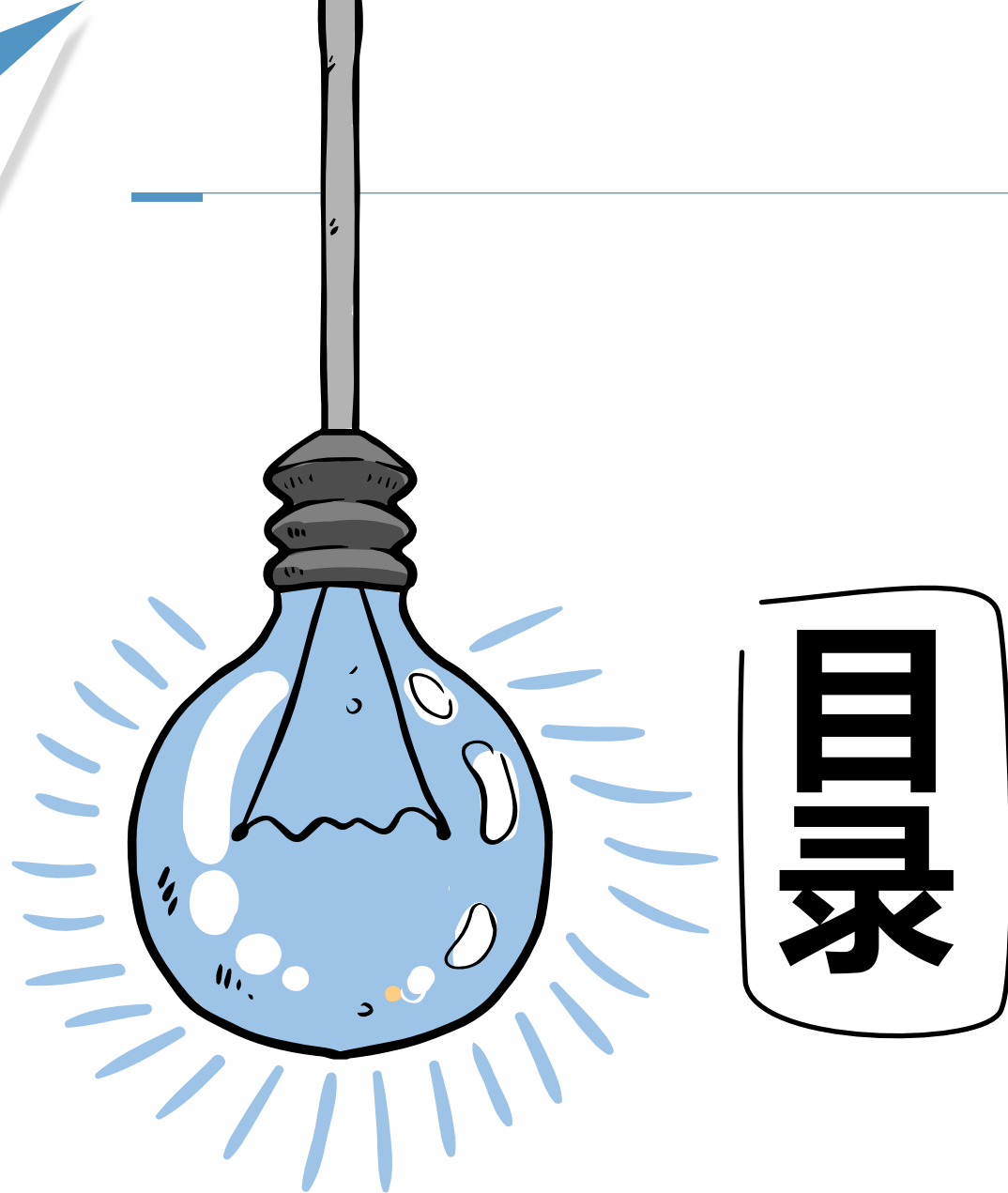


科学 公正 创新 服务

CPU Bench测试指导

中国电子技术标准化研究院

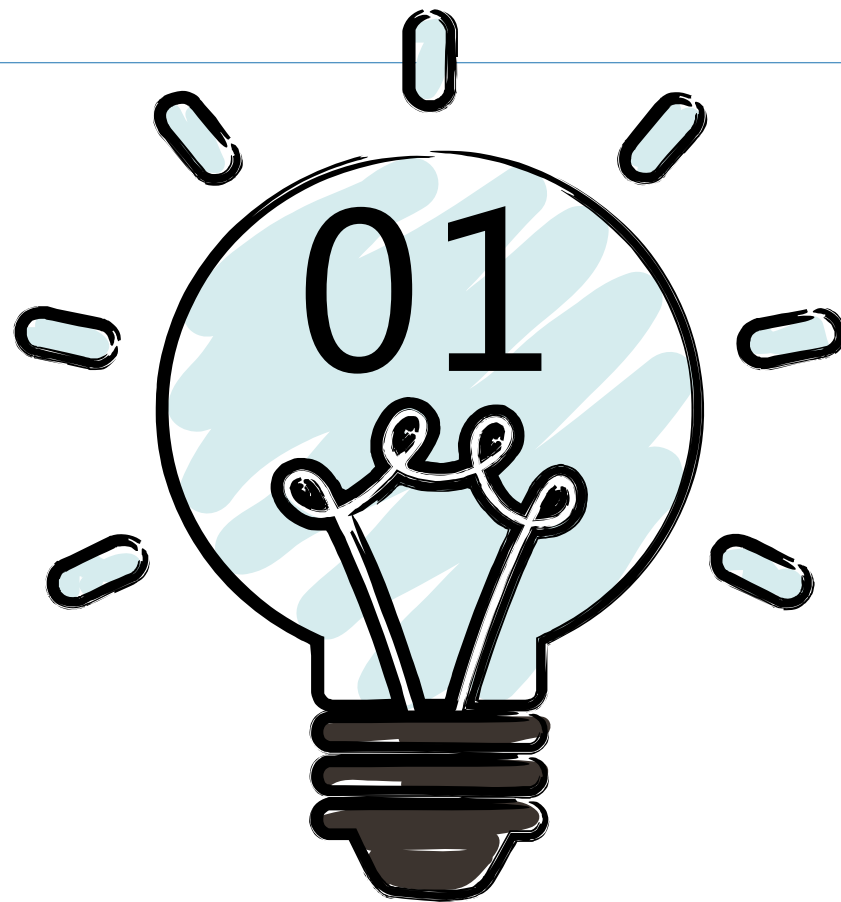
2022.11.05



1.工具介绍

2.工具使用

3.调优说明



工具介绍



计算产品性能基准工作组

Computing Product Performance Benchmark Workgroup

CPPB WG

计算产品性能基准工作组

Computing Product Performance

Benchmark Workgroup

致力于性能基准的

- 顶层设计
- 标准研制
- 工具研发
- 应用推广

- **支撑**计算产品性能迭代优化、**牵引**计算产业能力提升，为**促进**计算产业全链条健康有序发展的提供抓手。
- 工作组由工业和信息化部**电子信息司**、**信息技术发展司**共同领导，相关企事业单位共同参与。
- 工作组下设**秘书处**及**标准组**、**工具组**、**运营组**等专题组。



- 工作组内来自芯片、整机、操作系统、应用等产业链各环节的企业已近80家

CPU厂商

鲲鹏、海光等

用户

中国移动、中国电信
中讯邮电、政府用户等

整机厂商

浪潮、曙光、联想、超聚变、宝德、新华三等

高校

中科院、清华等

OS厂商

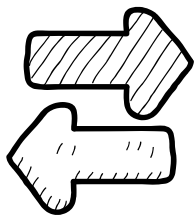
麒麟、统信

测试机构

电子标准院、电子五所
信通院、电科院



◆ 工作组网站提供新闻展示、标准查阅和工具下载。 <http://www.cppb-wg.com>



计算产品性能基准工作组 (Computing Product Performance Benchmark Workgroup , 简称 “CPPB WG”) 由处理器厂商、整机厂商、评测机构及学术机构共同成立。经过长时间的研究、设计、开发、验证等阶段，工作组于**2021年世界计算大会**完成了通用计算CPU性能评测基准工具CPUBench的发布。

处理器厂商

兆芯、龙芯、海光、华为、申威、飞腾等

整机厂商

联想、浪潮、曙光、长城等

评测机构

中国电子技术标准化研究院
中国软件评测中心

学术机构

清华大学，上海交通大学、华中科技大学等



CPUBench是计算产品性能基准工作组发布的通用计算性能评估套件，其测试负载来源于 HPC、大数据，数据库等常用**通用计算**领域的典型业务场景。

CPUBench定位于评估**CPU**、**内存子系统**的性能，以及与计算性能密切相关的**编译优化**能力，如编译器、JVM虚拟机。

负载设计介绍

CPUBench定位于评估**CPU**、**内存子系统**的性能，以及与计算性能密切相关的**编译优化**能力，如编译器、JVM虚拟机。

CPUBench根据计算类型（整型、浮点型）和并发数（单并发、多并发）设计**四大测试套件**，每个套件支持**typical**和**extreme**两种测试模式，形成八大性能指标，

测试类型	测试套件	性能指标	性能指标含义
整型	IntSingle	int_single_typical int_single_extreme	评估 单核整型 运算能力
	IntConcurrent	int_concurrent_typical int_concurrent_extreme	评估多核整型运算能力
浮点	FloatSingle	float_single_typical float_single_extreme	评估 单核浮点 运算能力
	FloatConcurrent	float_concurrent_typical float_concurrent_extreme	评估多核浮点运算能力

typical模式：要求测试套件中**相同语言**的负载必须采用**相同**的性能优化措施，体现性能优化的**通用性**。

extreme模式：允许**不同的负载**采用**不同**的性能优化措施，体现该系统**极限**优化的性能水平。



★ 整型负载 —— 包含**12个**整型负载，覆盖**大数据、数据库、压缩**等典型业务场景

★ 浮点型负载 —— 包含**10个**负载，覆盖**制造、气象、基因、物理化学**等典型科学计算领域

整型测试负载		
编译器	gcc	最流行的编译器
视频编解码	x264	遵循ITU标准，最好的有损视频编码器之一
数据压缩	gzip	流行的压缩/解压缩软件
大数据-统计	spark-wordcount	大规模数据统计处理
大数据-聚类	spark-kmeans	数据挖掘领域常用算法
基因领域	velvet	比较早也比较有名的短序列拼接软件
OLTP数据库	mysql-tpcc	事务型数据库标准测试模型
OLAP数据库	mysql-tpch	分析型数据库标准测试模型
数据压缩	xz	以高压缩比为特点的压缩软件
JSON 序列化	rapidjson	腾讯开发的高效JSON解析、生成器
加解密	openssl	目前最流行的SSL 密码库工具
语言解析器	python	最流行的解析型编程语言

浮点测试负载		
CAE/制造-计算气动声学	Nektar++	源自帝国理工大学，高精度计算代表方向
CAE/制造-计算流体力学	Phenglei	源自中国气动中心，国家数值风洞工程平台
CAE/制造-计算电磁学	Openfoam	最流行的电磁、流体数值模拟软件
基因领域	Phyml	生信典型算法-最大似然法构建系统发生树的软件
机器学习	Lightgbm	微软开发的分布式梯度提升框架，竞赛分数第一
数字媒体-图像渲染	Povray	典型基于光线追踪3D渲染软件
气象领域-天气预报	Wrf	全球各气象中心均在使用的天气模拟软件
分子动力学-无机物	Lammps	计算材料、无机化学流行应用
分子动力学-有机物	Gromacs	计算材料、有机化学流行应用
天文物理-N体模拟	Cube	上海交通大学+厦门大学共同开发，完成粒子数目最多的宇宙学N体模拟软件



性能度量方法

CPUBench的各子项计分规则如下：

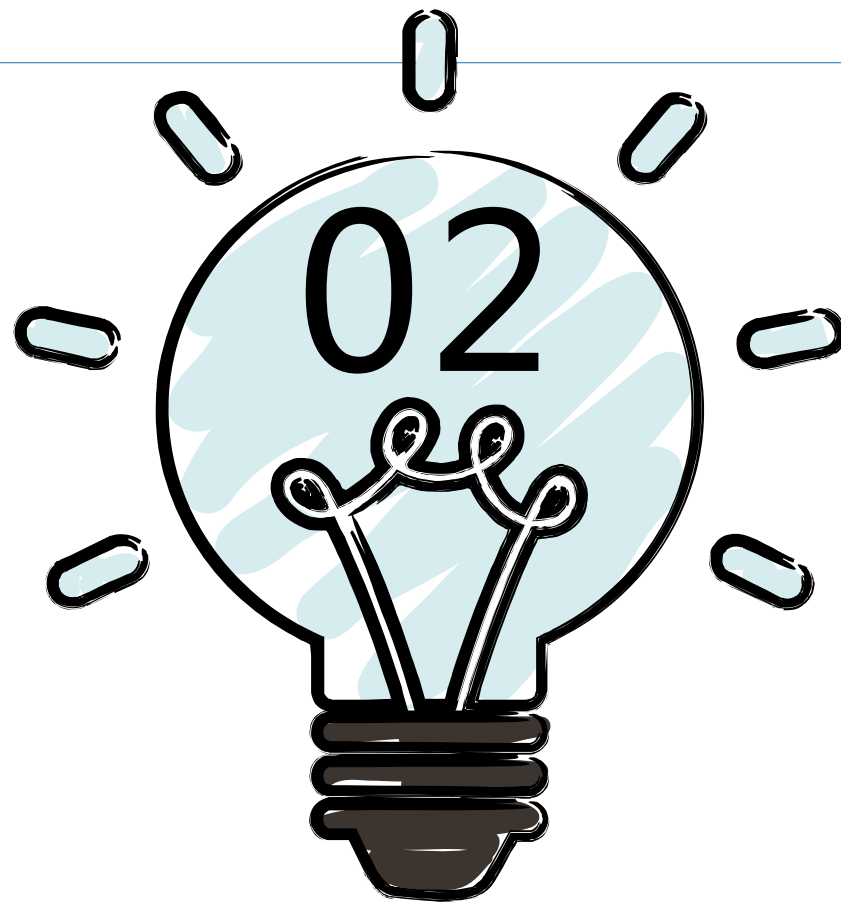
$$W_i = R_{time} / E_i$$



-  W_i 为各子项得分；
-  R_{time} 为参考时间，基于Intel Skylake 3106 CPU (1.7GHz 8Core) 的HPE DL380 Gen10服务器测试得出的时间；
-  E_i 为各个子项实际运行的时间。

CPUBench的测试套总分为各子项得分的几何平均值：

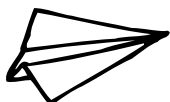
$$G = \sqrt[n]{\prod_{i=1}^n W_i}$$



工具使用

环境要求

软件要求



C、C++、Fortran编译器，版本 $\geq 5.3.0$ 。

Make，jdk等

硬件要求



工具运行需要约**2G内存**每实例。

工具运行时/dev/shm目录至少需要3G空间。

运行**Single模式**时至少需要**20G**硬盘空间

运行**Concurrent模式**时，每实例增加**2G**硬盘空间。

硬件环境一定要提前
检查，如报错，
请查阅测试指导书！

完整测试流程

1

解压安装

CPUbench采用**免安装模式**，
将CPUbench软件包上传到被测
系统并解压即可：

```
# tar -zxvf CPUbench-vx.x.x.tar.gz
```

2

修改配置文件

修改 config/config-template.ini文件

3

运行测试

直接执行cpubench.sh

4

查看报告

PDF报告:负载得分，运行情况等

修改配置项

完整的配置项
请参考指导书

修改 config/config-template.ini文件里的配置项。主要关注以下配置选项：

选项	默认值	作用域	含义
action	standard	common	指定本次工具的活动。完整流程： standard 编译：build 运行：run 清理数据：clean
benchmarks	[]	common	指定本次运行的子项集合或套件
tune	typical	common	调优级别，可以选择经典 (typical)或最优(extreme)。
jobs	1	common	运行实例个数，1表示单核单实例，大于1表示多核多实例。
iterations	3	common	运行的迭代次数，最后得分取均值。
taskset	[]	common	绑核列表，由逗号分隔的一系列cpu核心编号构成，同时支持通过-分隔符来标识一个编号范围，如1-5等价于1,2,3,4,5。

选项	默认值	作用域	含义
lang_dir	/usr/bin	namespace	编译器的bin目录
LD_LIBRARY_PATH	[]	namespace	若用户使用非系统自带编译器，需配置编译器lib库完整路径
CC	\${lang_dir}/gcc	namespace	使用的C语言编译器
CFLAGS	-O3	namespace	使用的C语言编译选项
CXX	\${lang_dir}/g++	namespace	使用的C++语言编译器
CXXFLAGS	-O3	namespace	使用的C++语言编译选项
LIBS	[]	namespace	链接库
AR	ar	namespace	使用的静态库制作工具，如不使用系统默认，请注意配置完整路径
RANLIB	ranlib	namespace	静态库的符号索引表更新工具，如不使用系统默认，请注意配置完整路径
FC	\${lang_dir}/gfortran	namespace	使用的fortran语言编译器
FFLAGS	-O3	namespace	配置fortran语言编译选项
java options	[]	namespace	JVM配置参数



通用配置项



编译器相关配置项

主要规则说明

01

若benchmarks配置为**多并发**测试套(IntConcurrent、FloatConcurrent)，jobs必须**大于1**，否则强制运行单并发测试套(IntSingle、FloatSingle)。若benchmarks配置为**单并发**测试套，jobs无论配置为多少，均强制运行单并发测试套。

02

设置taskset时，当该列表长度大于jobs时，只会选前jobs的cpu核进行绑定，若小于jobs，则会在超出列表长度时，从列表头重新绑定。

03

java_options选项传递给JVM的调优参数，通过spark.driver.extraJavaOptions生效。禁止通过spark.driver.extraJavaOptions=-Xmx设置最大堆容量，故**禁止**在本配置项设置**-Xmx**。

04

iterations必须设置为**3**，否则测试报告无效；**action**必须为**standard**，否则测试报告无效；必须**包含typical**模式下的测试结果，否则测试报告无效。

运行说明

切换到CPUBench的安装目录，直接执行cpubench.sh，运行命令：

```
# ./cpubench.sh [options] [arguments]
```

也可通过./cpubench.sh -h查看：

选项	含义
--benchmarks/-b	指定测试项
--action/-a	指定测试阶段
--config/-c	指定配置文件，可以是绝对路径或文件名或去后缀名，如：
	/CPUBench/config/config-template.ini
	或config-template.ini
	或config-template
--jobs	指定运行副本数
--tune/-T	指定测试模式
--iterations/-i	指定运行轮次
--work_dir	指定工作目录
--version/-V	输出工具版本
--verbose/--debug	指定是否开启debug日志
--rebuild/-R	指定负载运行前是否重新编译
--perf	指定是否开启性能采集
--perf_target/--ptarget	指定性能采集类别
--skip_verify	指定是否跳过工具校验
--help/-h	输出帮助信息



若相同的配置项既出现在命令行又出现在配置文件，
以命令行为主。

```
# ./cpubench.sh -c config-template.ini -b x264 gzip xz -i 3 --jobs 8
```

表示指定config目录下的config-template.ini为配置文件，仅测试int_x264，int_gzip，int_xz三个子项，各子项迭代测试3次，各子项的实例为8个。

```
# ./cpubench.sh -c config-template.ini -b x264 -a build
```

表示指定config目录下的config-template.ini为配置文件，仅对int_x264子项进行编译测试

```
# ./cpubench.sh -c config-template.ini -b x264 --skip_verify 1
```

表示指定config目录下的config-template.ini为配置文件，仅对int_x264子项进行测试，且**跳过工具校验**，一般修改工具源代码，进行调试时，增加该编译选项，但会导致最终的测试**报告无效**。



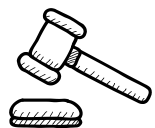
报告说明

CPU Bench测试报告包含**六个部分**：

- 运行日志（log）：CPU Bench运行日志与终端日志相同，记录工具的运行情况。
- 环境信息文件（CPU Bench_Environment.txt）：记录当次运行所采集的环境信息，方便用户更全面的了解运行环境各参数的配置情况。
- 当次运行配置文件（ini）：用户使用--config或-c指定的配置文件，若用户未指定，则默认为config-template.ini文件。
- **结果报告**：支持json、csv、pdf格式的最终报告，json用于上传官网进行审核发布，用户可以直接查看PDF进行性能分析。PDF报告包含了**测试套得分**、用户配置测试信息、软硬件信息、负载运行信息、构建运行信息以及优化选项。



报告有效性说明



PDF报告会标识本次运行是否有效，若需要合法的报告，须确保以下行为：



必须**包含typical**模式下的测试结果；



禁止**跳过工具完整性校验**；



必须运行**整个测试套**action=standard；



必须将**迭代次数**设置为**3** iterations=3；

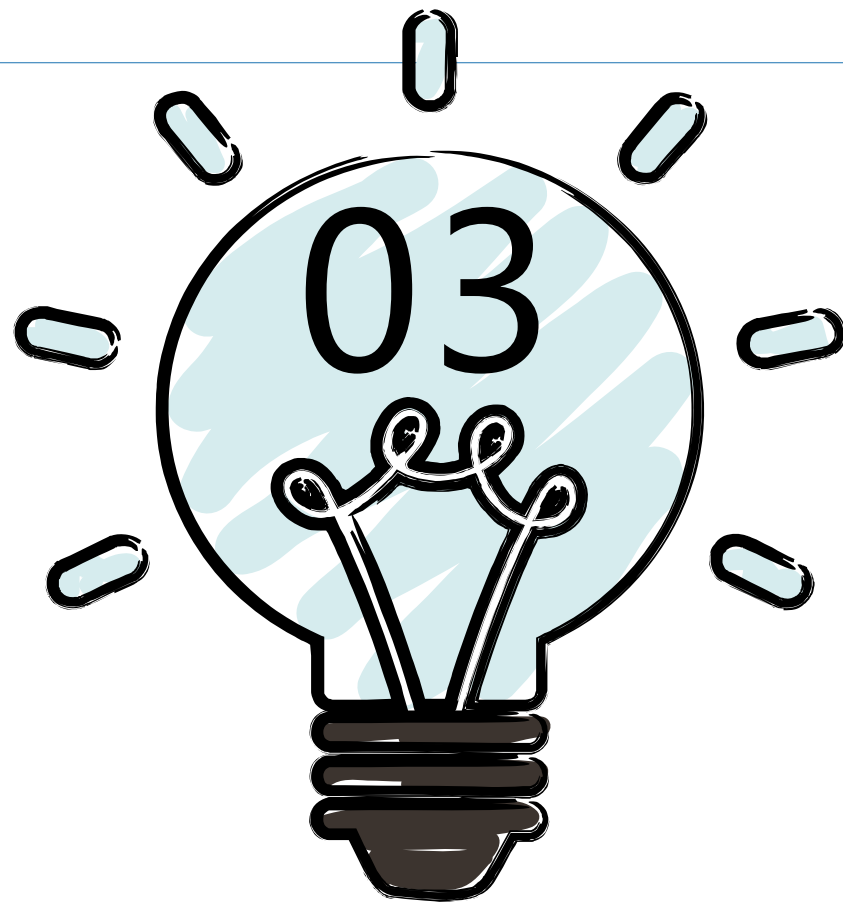


测试套中所有负载的运行**结果**必须**有效**；



typical模式下同语言测试负载必须拥有**相同的优化选项**。

若测试过程中有子项报错，请查阅指导书“常见问题”进行排查。



调优说明

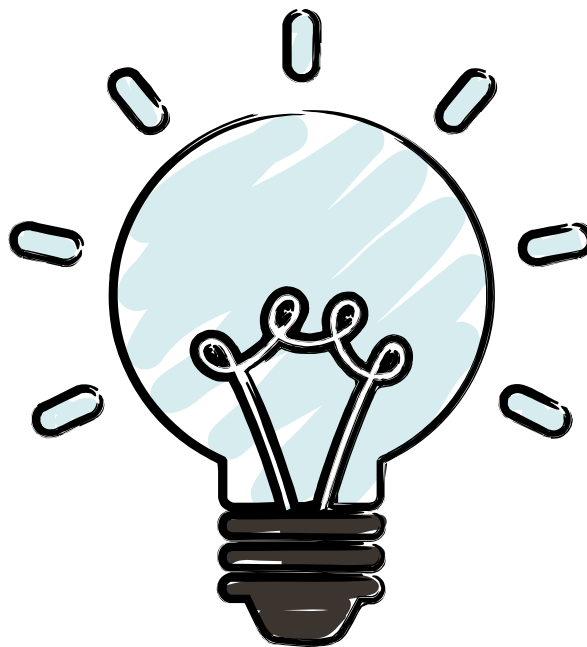


常用的调优手段



OS调优

通过修改OS启动参数，运行时参数等方法，提高CPU的性能。



编译器调优

不同类型的编译器，编译优化选项



数学库加速库等其它调优

Intel -MKL , AMD-ACML ,
鲲鹏-KML

OS调优介绍

OS调优手段，主要通过修改OS**启动参数**，**运行时参数**，对进程进行**绑核**等操作，从而提升CPU的性能。以下为常用的调优手段：



建议仅安装必须的软件，关闭无用的服务

评测时手动关闭非必要的后台服务，减少后台服务进程切换引起干扰。如：

```
# systemctl disable auditd.service
# systemctl disable libvirtd
# systemctl disable firewalld
# systemctl disable irqbalance
.....
```



修改OS启动参数

vim /etc/default/grub，找到GRUB_CMDLINE_LINUX=xx
添加需要的grub启动参数，如selinux=0

```
GRUB_TERMINAL_OUTPUT="console"
GRUB_CMDLINE_LINUX="rhgb quiet crashkernel=
0me selinux=0"
GRUB_DISABLE_RECOVERY="true"
```

OS调优介绍



绑核优化

测试时对整机进行绑核可以提升性能，如服务器有128核可以按照taskset=0-127绑定。

CPUbench可以通过修改配置文件的方式绑核：

```
#  
[common]  
action = standard          # options: standard, build, run, clean  
benchmarks = IntConcurrent  # Example: x264,gcc or lightgbm  
arch = auto                # arch: auto, aarch64, x86_64, mips64el  
tag = test  
tune = typical             # options: typical, extreme, both  
jobs = 128                 # 1: single-core performance. 1+: Multi-Core Performance.  
iterations = 3  
verbose = 0               # options: 0(close), 1(open)  
work_dir = bench_line     # Any directory with the write permission.  
rebuild = 0               # options: 0(close), 1(open)  
taskset = 0-127           # Set the number of the bound CPU core.
```

编译器调优介绍

编译调优主要是从编译器的角度对应用进行优化，在编译过程中的优化器阶段，gcc 提供了近百种优化选项，通常考虑3种优化编译选项：

01 O2/O3及相关flag优化

GCC提供了不同的优化级别，O0为默认的编译选项，减少编译时间和生成完整的调试信息。O1对程序做部分编译优化O2 比O1更加优化，尝试更多的**寄存器级**的优化以及指令级的优化。O3在包含 O2 所有的优化的基础上，使用**普通函数的内联**，以及针对**循环**的更多优化。

可通过**gcc -Q -O2 --help=optimizers**查看对应O2级别开启的优化列表，同理可查其他级别。

可通过**gcc -Q -O2 --help=params**命令进行查看参数型的编译优化选项。

不同的编译选项开关，编译选项参数，能够达到不同的优化效果。如：

```
CFLAGS = -O3 -mcpu=native -fno-default-inline -finline-limit=400
```

表示开启编译器-O3级别的优化，使编译器自动检测处理器，在函数体内定义的函数不做内联处理，限制决定函数是否能被 inline 的伪指令长度为400。



<https://gcc.gnu.org/onlinedocs/gcc/Optimize-Options.html>

编译器调优介绍

02 LTO优化

LTO (Link Time Optimization) : 传统的编译方式只能将优化局限于单个编译单元 (如.c文件) , 链接时优化可以**跨编译单元全局优化**, 从而实现跨文件函数内联、函数特化、常量传播等优化。链接时优化是对整个程序的分析 and 跨模块的优化, 因此**会增加链接编译时间**。

- ★ gcc编译时添加-flto。
- ★ 设置并发增加编译速度, -flto=N。

03 使能硬件feature

一些架构本身有对应的特性能够使能优化, 比如ARM的CRC循环冗余校验对一些内核态有大量网络通信且调用CRC校验的场景workload, 能够使用该硬件特性进行优化。

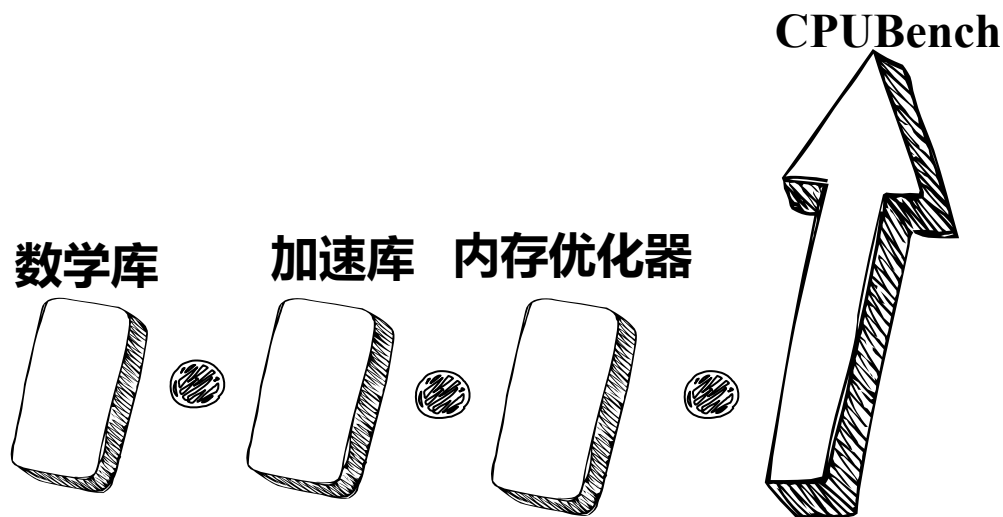
可用 “**cat /proc/cpuinfo**” 命令查看ARM硬件是否支持, 在Features一行有crc32即为支持硬件指令加速。

要使用该特性, 只要在ARM平台中增加-march=armv8-a+**crc**编译选项即可, 其它服务器可查看对应的硬件特性进行调优。

其它优化手段

除了上述的一些调优方法外，还可以借助一些工具，如**perf top**查看热点函数，分析**热点函数**，针对性地对热点函数进行优化。比如，当分析热点函数malloc和free占比较高时，可以考虑使用其它内存分配器优化，常用的有jemalloc/tcmalloc。

另外，一些厂商也提供了对应的**数学库**和**加速库**，可以对计算密集型HPC进行优化，如Intel提供的MKL，AMD提供的ACML，鲲鹏提供KML对浮点测试套件有所提升。



思考？

编译优化选项这么多，
如何找到**最优的组合**？

编译器类别如此多，
如何选择**最优的编译器**？

如何找出**优化空间**最大的负载？

JAVA应用应该如何
配置**JVM参数**？

