引用格式: 逯海涛, 任翔, 钟伟军, 等. CPUBench: 一款开放的通用计算 CPU 性能基准工具[J]. 微电子学与计算机, 2023, 40(5): 75-83. [LU H T, REN X, ZHONG W J, et al. CPUBench: An open general computing CPU performance benchmark tool[J]. Microelectronics & Computer, 2023, 40(5): 75-83.] **DOI:** 10.19304/J.ISSN1000-7180.2022.0469

CPUBench: 一款开放的通用计算 CPU 性能基准工具

逯海涛, 任 翔, 钟伟军, 赵 鑫, 尹 航
(中国电子技术标准化研究院, 北京 100176)

摘 要: 计算产品性能基准工具是支撑计算产品性能迭代优化和牵引计算产业能力提升的重要保障.为弥补当前国内 CPU 性能评测基准工具的空白,开发了 CPUBench,定位于对通用计算场景下的 CPU、内存子系统以及所依赖的编译器进行综合计算能力评估. CPUBench 测试框架采用 Python 语言开发,模块化设计,兼容 x86_64、aarch64、ppc64le、sw_64 和 loongarch64 等多种 CPU 架构. 测试负载来源于各领域典型业务场景的计算密集型应用,具有良好的实际业务代表性,从指令比例、Top-down 等架构相关或无关特征看,负载之间特征差异明显,整个测试套件的特征覆盖广,能够充分代表目前通用计算场景下的实际业务特征.同时通过 PCA 分析方法对比 CPUBench 和 SPEC CPU2017 在相同测试环境上的微架构特征,CPUBench 基本覆盖 SPEC CPU2017 的特征类,并增加了大数据、数据库等新型应用的特征.此外,在 14 个不同的被测平台上对 CPUBench 和 SPEC CPU2017 两款工具进行了测试比较,测试分数曲线显示出良好的趋势一致性,间接证明 CPUBench 作为一款通用计算 CPU 性能评测基准工具的可用性与合理性.总体上,CPUBench 在业务代表性、易用性、易维护性等方面具备一定的优势,可用于指导通用计算 CPU、服务器等计算产品的设计优化、规格选型和市场采购等工作,对计算产业的发展具有重要的牵引意义.

关键词: 通用计算 CPU; 性能; 基准工具; SPEC CPU2017
中图分类号: TN407 **文献标识码:** A **文章编号:** 1000-7180(2023)05-0075-09

CPUBench: An open general computing CPU performance benchmark tool

LU Haitao, REN Xiang, ZHONG Weijun, ZHAO Xin, YIN Hang
(China Electronics Standardization Institute, Beijing 100176, China)

Abstract: The computing product performance benchmark tool is an important guarantee to support the iterative optimization of computing product performance and improve the capability of the computing industry. In order to fill the gap in the current domestic CPU performance evaluation benchmark tool, CPUBench is developed, which is positioned to evaluate the comprehensive computing capacity of the CPU, memory subsystem and the compiler under the general computing scenario. The framework of CPUBench is developed by Python, modular design, and compatible with x86_64, aarch64, ppc64le, sw_64, loongarch64 and other pocessor architectures. The workloads comes from the computing intensive applications of typical business scenarios in various fields, and has a good representation of the actual business. From the perspective of architecture related or irrelevant characteristics such as instruction proportion and top-down, the characteristics of the workloads are obviously different. The characteristics of the whole test suite cover a wide range, and can fully represent the actual business characteristics under the current general computing scenario. At the same time, the microarchitecture characteristics of CPUBench and SPEC CPU2017 on the same test environment are compared by PCA analysis method.CPUBench basically covers the characteristic classes of SPEC CPU2017, and adds the characteristics of new applications such as big data and database.In addition, CPUBench and SPEC CPU2017 were tested and compared on 14 different tested platforms, and the score curve showed good trend consistency, indirectly proving the availability and

收稿日期: 2022-08-07; **修回日期:** 2022-09-01

rationality of CPUBench as a general CPU evaluation tool.In general, CPUBench has certain advantages in business representativeness, ease of use and maintainability, which can be used to guide the design optimization, specification selection and market procurement of general computing CPU, server and other computing product, and has important guiding significance for the development of the computing industry.

Key words：general computing CPU；performance；benchmark tool；SPEC CPU2017

1 引言

计算性能评测是提升计算机系统能力的重要方法,一款好的性能评测基准工具不仅能够客观、公正地揭示出不同计算机系统的性能差异,还能够用于反映计算机系统处理不同负载任务时的性能特征,对定位系统性能瓶颈、指导系统性能优化、牵引产品设计具有重要帮助[1].

上世纪五十年代,计算性能评测是以每秒百万条指令（MIPS, Million Instructions Per Second）来进行性能评估.随着浮点运算量的增加,每秒百万浮点运算次数（MFLOPS,Million Floating Point Operations per Second）性能指标开始得到使用.由于上述指标不能反映处理器缓存、流水线等较新的体系结构特点,通过统计执行简单指令速度的性能评测方法逐渐不再采用.1964 年 Whetstone[2] 的出现表明评价指标开始从简单指令的执行情况过渡到代表性功能函数或程序片段的执行情况.Whetstone 定义的数据类型包括整型、长整型及双精度型,整型变量用来控制循环,双精度变量用来执行浮点操作,反映了 CPU 处理浮点和控制指令的能力.发布于 1984 年的 Dhrystone[3] 测试的核心为字符串赋值和字符串比较,可用来评估系统的整型任务处理能力.Whetstone 和 Dhrystone 作为代表性的人工合成基准测试工具,也是最早的基于高级计算机语言的基准测试工具,在早期的处理器性能评估上发挥了重要作用.Livermore Loops[4]、Linpack[5] 和 NPB（NAS Parallel Benchmark）[6] 均基于应用程序的内核代码抽取、设计,是内核类基准中使用较多的工具.Livermore Loops 执行 24 个不同的 Fortran 语言编写的数学应用算法,主要用于评估浮点计算性能.Linpack 旨在用高斯消元法求解 N 元一次稠密线性代数方程组,来评价计算机的浮点性能,被用于 TOP500 计算机的榜单排名.NPB 是一款以高并行超级计算机为性能评估对象的测试工具,测试负载主要来自于计算流体力学应用.

随着处理器微架构及业务负载复杂度的提升,上述评测方法已不能满足或不能正确反映计算机系统的实际性能.因此,基于实际业务负载的测试程序开始流行起来,SPEC（Standard Performance Evaluation Corporation）推出的 SPEC CPU 基准工具便凭借其负载多样性、评测全面性和规则透明性等优势逐渐在计算产业界流行,并通过 30 多年共 6 个大版本的演进,发展成为具有全球影响力的通用计算 CPU 性能评测基准工具[7].但 SPEC CPU 目前也面临一些挑战,由于其版本继承时间跨度大,历史包袱较重,程序框架基于 Perl 语言开发,语法晦涩,维护难度高;版本更迭缓慢,导致测试套件缺乏新兴领域的应用代表,不能尽快体现新兴业务对计算系统能力的需求;不兼容部分类型硬件架构,不能满足更广泛的系统评估要求;测试中编译器带来的定向优化也不能反映实际业务性能,例如 ICC 编译器对 SPEC CPU2006 的 462.libquantum 的优化效果显著,但对其它测试负载和实际业务程序并没有表现出应有的效果.

为准确评估计算系统的真实性能,解决当前性能基准工具存在的问题,弥补国内在性能测试技术研究及基准工具研制方面的经验不足,联合国内处理器厂商、整机厂商、评测机构及学术机构等多家企事业单位共同参与了 CPU 性能评测基准工具 CPUBench 的研究、设计、开发和验证等工作,并于 2021 年世界计算大会上完成该工具 v1.0 版本的发布.

2 CPUBench 工具介绍

CPUBench 的测试框架使用 Python 语言开发,模块化设计,增强了工具的易维护性.CPUBench 共包括 22 个测试负载,均来源于实际的应用场景,如编译器、视频编解码、数据压缩、加解密、大数据和数据库等通用计算场景,电磁学、流体力学、基因拼接、气象预报和分子动力学等科学计算场景.

考虑到多种计算架构平台的性能评测需求,CPUBench 已经移植到 x86_64、aarch64、ppc64le、sw_64 和 loongarch64 等指令集架构的处理器平台,可运行在 KylinOS、UOS、openEuler、CentOS 和 Ubuntu 等基于 Linux 的操作系统,兼容 GCC、Clang、AOCC、ICC 和 BiSheng 等编译环境.CPUBench 安装包提供针对不同指令集架构的内置依赖库,简化测试环境部署过程,提高了工具的易用性.

2.1　测试框架

CPUbench 工具总体框架如图 1 所示,主要包括:负载模块、配置模块、控制模块、通用模块和平台模块五个部分.当在被测系统(SUT,System Under Test)上运行测试套件或部分测试负载时,测试框架将自动执行以下流程:(1)检查被测系统是否具备必要的依赖,如 gcc、g++、gfortran 和 jdk 等;(2)提取内置工具,如 cmake 和 python3;(3)解析命令行和配置文件的参数;(4)核心程序代码的防篡改检查;(5)编译、构建和运行所需的测试套件或测试负载;(6)运行后校验每个测试负载结果的正确性;(7)计算性能得分,收集被测对象的软硬件信息,并汇总到测试报告中.

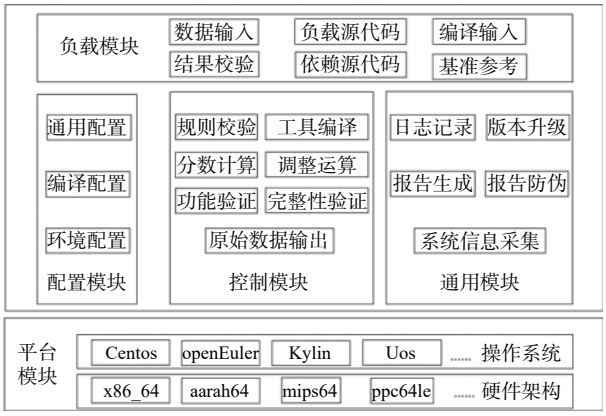

图 1　CPUBench 工具总体框架

Fig. 1 Overall framework of CPUBench

2.1.1　负载模块

该模块用来提供运行所需的测试负载,包含负载源代码、依赖库、输入数据和校验结果等内容.

2.1.2　配置模块

该模块定义了如何构建和运行测试套件的详细规则,包含通用配置、编译配置和环境配置等内容,作为测试结果可复现的依据.

2.1.3　控制模块

该模块用于实现对测试套件运行的调度,包括规则校验、工具编译、分数计算、调度运行、功能验证和完整性校验等功能.

2.1.4　通用模块

该模块用于实现测试所应具备的通用功能,包括多种格式的报告生成、版本升级、日志记录、报告防伪以及对被测对象软硬件信息采集等功能.

2.1.5　平台模块

该模块用于保证测试基准框架对被测系统环境的兼容性.

2.2　测试负载

CPUBench 的测试负载选择需要考量诸多因素,首先,测试负载应来自于实际的应用场景,具有较高的业界认可度和影响力.其次,还要综合评估各负载于多架构平台的移植成本、计算密集程度、编程语言、稳定性、公平性和知识产权等因素.另外,基准工具整体的微架构特征覆盖度、业务领域覆盖度以及不同负载之间的差异性同样作为重要的选择依据.基于原始业务应用,设计并实现一个可用的测试负载也是一个复杂的过程,不光要移除掉除业务核心计算逻辑外的其他代码部分,如网络、图形、I/O 操作和依赖于特定架构的汇编代码等,同时要为裁剪后的负载设计输入数据集,并确保运行时长在合理范围内.最后要基于一系列不同的测试环境进行测试验证,考察裁剪后负载的稳定性、兼容性、可重复性、资源利用率和微架构特征等指标.

表 1 和表 2 给出了 CPUBench 的 12 个整型负载和 10 个浮点负载的具体信息描述.可以看出,所选负载业务应用领域广泛,整型测试套件除传统的编译器和数据压缩等负载外,还新纳入了目前云计算领域比较热门的大数据及数据库等领域负载;浮点测试套件主要聚焦在气象预报、基因领域、数字媒体、分子动力学等科学计算领域.负载实现语言包括 C、C++、Fortran、Java 和 Scala,Java 语言应用的引入能够代表服务端主流应用场景的性能需求.

表 1　整型测试套件负载

Tab. 1　Integer test suite workload

整型负载	应用领域	编程语言
x264	视频编解码	C
gcc	编译器	C++
gzip	数据压缩	C
tpcc	事务处理	C,C++
tpch	在线分析	C,C++
kmeans	机器学习	Java,Scala
wordcount	大数据-统计	Java,Scala
velvet	基因拼接	C
openssl	加解密	C
rapidjson	JSON解析/生成	C++
python	Python解释器	C
xz	数据压缩	C

表 2 浮点测试套件负载

Tab. 2 Float-point test suite workload

浮点负载	应用领域	编程语言
lightgbm	机器学习	C++
nektar++	气动声学	C++,Fortran
phenglei	流体动力学	C++
phyml	基因分析	C
gromacs	分子动力学	C++
povray	光线追踪	C++
openfoam	电磁学	C++
lammps	分子动力学	C++
cube	天体物理	C
wrf	天气预报	C,Fortran

2.3 性能指标

CPUBench 共包含四个测试套件：IntSingle、IntConcurrent、FloatSingle 和 FloatConcurrent,分别用于评估计算机系统的单核整型运算能力、多核整型运算能力能、单核浮点运算能力和多核浮点运算能力.每个测试套件均可指定 typical 或 extreme 模式来运行,其中 typical 模式是基本优化下的性能测试,任何优化措施统一应用到所有的测试负载;extreme 模式可针对不同的测试负载采用不同的优化措施.

单核测试套件里每个测试负载的得分是一个比值,即该负载在参考机器上运行得到的参考时间除以被测系统运行该负载的时长,而后将测试套件所有负载的得分取几何平均便是该套件的最终得分,该分数可用于评估被测系统单核运算能力;多核测试套件里的测试负载以多任务并发的形式执行,该负载在参考机器上运行得到的参考时间除以被测系统运行其中最慢任务的时长,再乘以并发的任务数量,最后对所有负载得分进行几何平均处理便是该套件的得分,该分数用于评估多核并发的运算性能.

出具有效测试报告时,指定的测试套件应执行三次,取测试套件里每个负载三次测试结果的中位值作为求几何平均的输入值.测试正常执行完成后,测试结果生成并保存到报告中,如图 2 展示的是在参考机器上运行 IntSingle 测试套件的结果（参考机器理论上单核测试结果为 1）.

CPUBench 的参考机器为一台搭载 Intel Xeon Bronze 3 106 处理器的 HPE 服务器,详细配置见表 3.基准工具中内置的每个负载的参考时间都是从参考机器上测试得到.

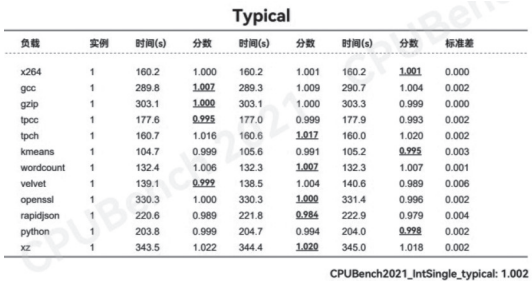

图 2 IntSingle 测试套件在参考机器上的三次运行结果

Fig. 2 Results of three runs of the IntSingle test suite on the reference machine

表 3 参考机器的配置信息

Tab. 3 Configuration information of reference machine

组件	配置信息
CPU	Intel Xeon Bronze 3106 * 1
内存	16 GiB 2666MT/s * 6
硬盘	480 GB SSD * 2
操作系统	CentOS Linux 7 3.10.0-957.el7.x86_64
JDK	1.8.0_292
编译器	GCC 7.3.1

3 负载特性分析

在本节中,基于一台运行 Kylin V10 操作系统的 aarch64 架构服务器,对每时钟周期完成的指令数（IPC,Instruction Per Cycle）、指令占比（Intstruction Mix）及访存带宽等应用程序特征和微架构相关特征进行了考察,并使用 Top-down 微架构分析方法（TMA, Top-down Microarchitecture Analysis）[8] 和主成分分析（PCA, Principal Components Analysis）[9],从不同维度展示了 CPUBench 负载的微架构特征多样性.

3.1 IPC

IPC 是 CPU 主要性能指标之一,不仅与流水线深度、乱序执行、分支预测器的准确性和 cache 配置等微架构设计相关,同时与数据访问模式、指令序列的依赖距离及编译器的优化能力等应用程序特征密切相关.IPC 能够展现 CPU 执行指令的吞吐能力,也能客观反映应用程序的指令并行度.使用 perf 工具采集了 CPUBench 各负载的 IPC,如图 3 和图 4 所示,IntSingle 测试套件的 IPC 范围是 0.76~2.73.其中,IPC 最高的负载是 openssl,主要是因为加解密算法具有访存少执行效率高的特点;第二高的 x264 是

视频编解码负载,其矩阵运算数据访问模式规律,指令可向量化;IPC 最低的 velvet 是一个基因拼接应用,由于需要处理大量的基因序列片段,致使访存流量大,指令并行度较低.

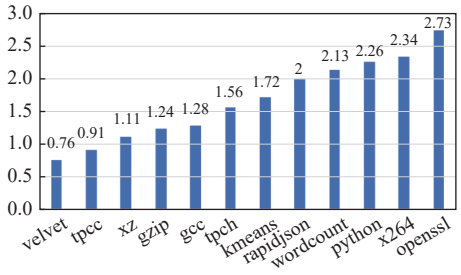

图 3　IntSingle 测试套件 IPC

Fig. 3 IPC of Intsingle test suite

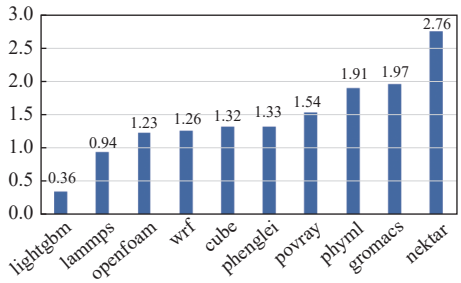

图 4　FloatSingle 测试套件 IPC

Fig. 4 IPC of FloatSingle test suite

FloatSingle 测试套件的 IPC 范围是 0.36~2.76,其中 IPC 最高的 nektar 来自于 CAE 领域,在 CPUBench 中用于气动声学的仿真计算,调用了 BLAS 库的 dgemv 和 dgemm 接口,矩阵向量运算执行效率高;IPC 最低的负载是 lightgbm,属于机器学习决策树模型,计算过程访存流量大.

参考 SPEC CPU2017 IntSpeed 套件的 IPC 范围是 0.70~2.90,FpSpeed 的 IPC 范围是 0.85~3.11[10-11]. 较 SPEC CPU2017,CPUBench 单核测试套件的 IPC 上限略低（不同微架构测试环境的差异）,但覆盖区间基本相似.综上所述,CPUBench 的负载指令并行度覆盖范围广,可用于评估 CPU 微架构执行复杂指令流的效率.

3.2　指令占比

指令占比是程序的主要微架构无关特征之一,不同的微架构具有不同的计算单元数量,执行不同指令所需时钟周期也不相同.因此,测试负载能够涵盖丰富的指令类型便至关重要.采集了 CPUBench 各个负载的指令占比,如图 5 和图 6 所示,IntSingle 测试套件和 FloatSingle 测试套件涵盖了丰富的指令类型.其中,IntSingle 测试套件的平均指令分布分别为 Integer 46.7%, FP 0.1%,SIMD 1.4%,Control 18.8%,Load 23.5%, Store 9.4%;FloatSingle 测试套件的平均指令分布分别为 Integer 30.02%, FP 22.9%,SIMD 2.8%,Control 11.8%,Load 25.8%,Store 6.6%.不同负载的指令占比具有明显的差异,比如 tpcc 和 tpch,由于代码规模大、代码控制流复杂,Control/Branch 指令频率较高,可用于评估复杂场景下分支预测器的执行效率;视频编解码负载 x264,由于矩阵运算频繁、数据访问模式规律,SIMD 指令占比相对较高,可用于评估不同编译器向量优化能力及不同微架构处理器的向量化执行效率.此外,两个测试套件都包含各种比例的 Load 和 Store 指令,反应了实际应用不同的数据访问读写模式[12],以及对 CPU 内存子系统的压力.通常高内存读写占比会导致 CPU 后端瓶颈,下一节的 Top-down 分析 lightgbm 等也反映了这一现象.

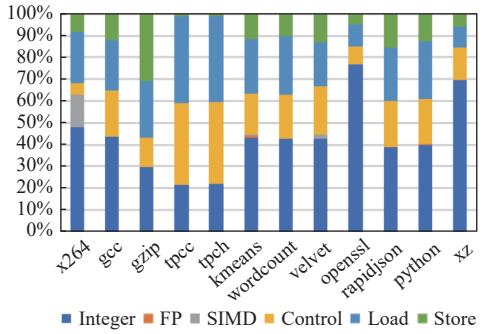

图 5　IntSingle 测试套件的指令混合统计

Fig. 5 Instruction mix statistics of IntSingle test suite

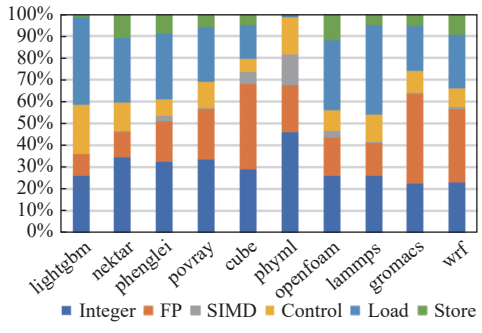

图 6　FloatSingle 测试套件的指令混合统计

Fig. 6 Instruction mix statistics of FloatSingle test suite

3.3　Top-down 微架构分析

使用 Top-down 微架构分析方法分析了 IntSingle 和 FloatSinle 两个测试套件运行时的微架构特征.如图 7 和图 8 所示,这两个测试套件均覆盖了 Frontend

Bound、Backend Bound、Bad Speculation 和 Retiring 类.其中,IntSingle 测试套件特征覆盖较全面,既有 Frontend Bound 占比高的 tpcc 和 tpch 负载,也有 Backend Bound 占比高的 velvet 和 rapidjson 负载,且该套件内各个负载的 Bad Speculation 占比程度不同.tpcc 和 tpch 均属于 Frontend Bound 类型的代表,两个负载的代码规模较大,控制流较复杂,从而提升了 footprint 和 cache miss.velvet 属于 Backend Bound 类型负载,该负载处理的基因片段数据量较大,内存访问带宽的需求较大.而 FloatSingle 测试套件,整体上 Backend Bound 占比较高(Backend Bound 又细分为 Memory Bound 和 Core Bound).通过进一步采集性能监控单元(PMU,Performance Monitor Unit)的事件进行详细分析,如表 4 所示,lightgbm、phenglei、wrf 和 openfoam 等负载内存读写操作频繁,属于 Memory Bound 类;nektar 负载的访存带宽同样较大,由于 BLAS 库矩阵运算高效,Retiring 占比较高,IPC 也较高;gromacs 和 cube 属于 Core Bound 类,依据上一节指令占比分析,两个负载的浮点指令占比较高;另外,lammps 的 Bad Speculation 占比较高.总体上,从 Top-down 来看 CPUBench 对微架构的特征覆盖较全面.

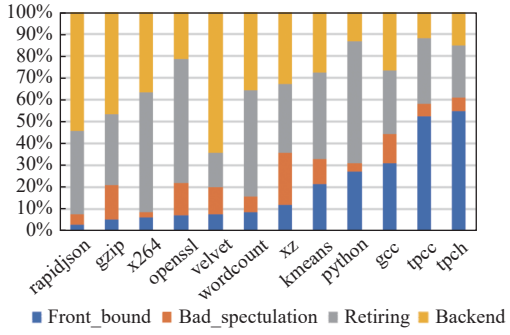

图 7 IntSingle 测试套件的 TMA 分析

Fig. 7 TMA analysis of IntSingle test suite

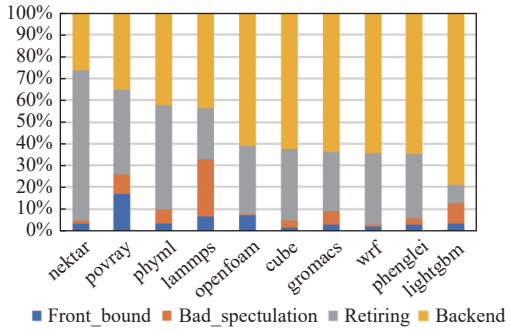

图 8 FloatSingle 测试套件的 TMA 分析

Fig. 8 TMA analysis of FloatSingle test suite

表 4 内存访问带宽及缓存未命中率

Tab. 4 Memory access bandwidth and cache miss rate

负载	内存读 MB/s	内存写 B/s	L1 d miss rate	L2 d miss rate	L3 d miss rate
lightgbm	8 000	10	20.89	46.78	00.57
nektar	1 200	80	01.08	19.51	05.05
phenglei	1 400	500	01.47	07.44	30.61
povray	10	10	00.27	01.13	11.06
cube	640	360	00.98	40.62	51.50
phyml	900	251	01.38	25.98	28.42
openfom	2 552	161	02.40	29.63	3.55
lammps	286	84	00.40	19.38	42.73
gromacs	171	31	00.35	31.35	18.95
wrf	1 847	381	03.18	32.04	19.60

3.4 主成分分析

通过主成分分析方法,分别对 CPUBench 和 SPEC CPU2017 测试负载采集 26 维特征,包括 IPC、L1/L2/L3 cache miss rate、TLB miss rate、working size set、footprint、条件分支跳转、直接跳转、间接跳转、指令占比和访存带宽等进行负载的相关性分析.如图 9 所示,两个工具的整型测试套件聚类成 6 类欧氏距离大于 10 的新特征,其中 SPEC CPU2017 和 CPUBench 均覆盖 5 类.不同之处在于 SPEC CPU2017 有一类访存密集型负载 505.mcf,单独覆盖 1 类.而 CPUBench 增加了数据库负载 tpcc 和 tpch,单独覆盖 1 类; 如图 10 所示,浮点测试套件聚类成 5 类欧氏距离大于 10 的新特征,SPEC CPU2017 覆盖了 4 类,CPUBench 覆盖了 3 类.总体来说,这两个基准的测试套件都有一些特征相似的测试负载,也有各自独有的应用特征,而 CPUBench 引入的数据库和大数据负载对反映当前流行的服务端应用特征有巨大价值.

4 测试对比分析

本文用 CPUBench 工具对 14 款配置不同处理器的计算机产品进行了实测,同时为了侧面验证 CPUBench 工具测试结果的合理性,也采用 SPEC CPU2017 工具进行了测试,测试用到的处理器产品信息如表 5 所示,处理器具体的主频、核心数量和缓存等参数信息可从各产品官网获取.

为客观比较两款工具的测试结果,同一被测平台的软硬件配置保持不变,不针对任一工具采用任何优

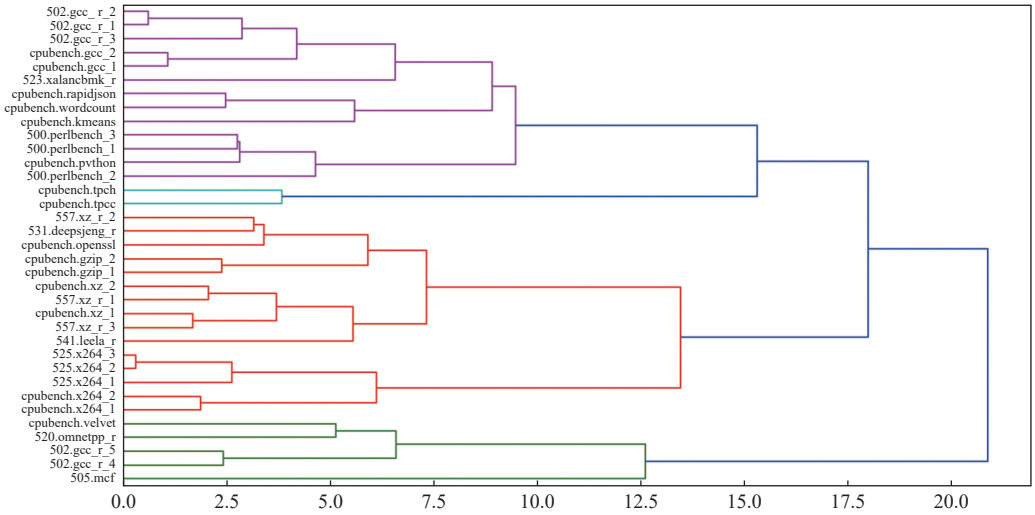

图 9　整型测试套件的 PCA 分析

Fig. 9 PCA analysis of integer test suite

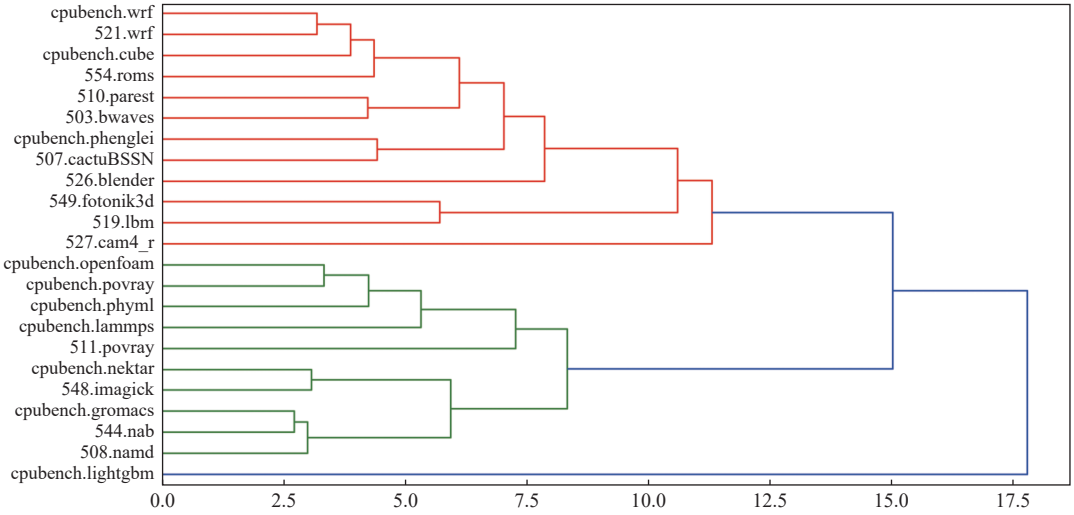

图 10　浮点测试套件的 PCA 分析

Fig. 10 PCA analysis of float-point test suite

化措施（均采用默认配置）.在比较单核测试结果方面,SPEC CPU2017 中 SPECrate 指标的副本数（copies）设置为 1,而不采用 SPECspeed 指标,这是因为 SPECspeed 指标中部分负载是以单任务多线程的方式进行执行.两款工具分别以 base 和 typical 基础模式进行测试,从图 11 和图 12 的对比结果可以看出,这两款工具在 14 个不同被测系统上的得分曲线趋势非常相似.鉴于 SPEC CPU2017 工具的成熟度,测试结果相似也一定程度上表明 CPUBench 能够正确、合理地评估不同被测对象的计算性能.但与此同时,考虑到两款工具测试负载的选择、相同负载使用的版本、输入数据集的设计、编译器对基准工具的优化效果以及新的 Java 类负载的引入等影响因素,在更广泛的测试配置条件下,测试结果出现一定的个体差异性也是可以想象的.总之,SPEC CPU2017 的分数有一定参考性,但与其完全一致并不能作为评判 CPUBench 工具合理、正确的唯一指标.

5　结束语

本文展示了一款通用计算 CPU 性能评测基准工具 CPUBench,对其测试框架、测试负载和性能指标等进行了介绍.CPUBench 具有良好的易用性和易维护性,对不同的处理器架构平台都具有很好的兼容性.通过 IPC、指令占比及 Top-down 等架构相关或无关特征发现,不同负载之间特征差异明显,整个基准测试套件的特征覆盖广,能够充分代表目前通用计

表 5 处理器信息描述

Tab. 5 Information description of the processor

指令集架构	产品型号
x86_64	Hygon C86 3250
	Hygon C86 7285
	KX-U6780A
	KH-37800D
	Intel Xeon Gold 6248
	Intel Xeon Bronze 3106
aarch64	Kupeng920 5250
	Kunpeng920 5251K
	FT-D2000
	FT-S2500
loongarch	LS3A5000
	LS3C5000L
ppc64le	POWER8
sw_64	SW3231

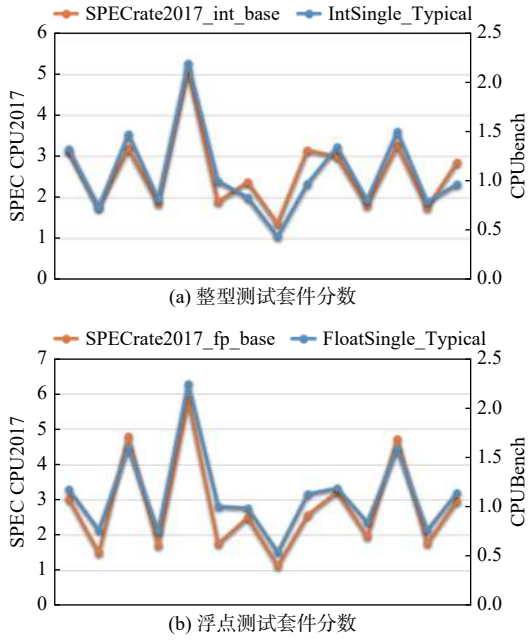

图 11 CPUBench 和 SPEC CPU2017 单任务分数趋势

Fig. 11 Single-task score trends for CPUBench and SPEC CPU2017

算场景下的实际业务特征.另外,也采用 PCA 方法对 CPUBench 和 SPEC CPU2017 在相同测试环境上的微架构特征进行了对比,结果表明 CPUBench 可基本覆盖 SPEC CPU2017 的特征类,并增加了大数据和数据库等新型应用的特征.CPUBench 和 SPEC CPU2017

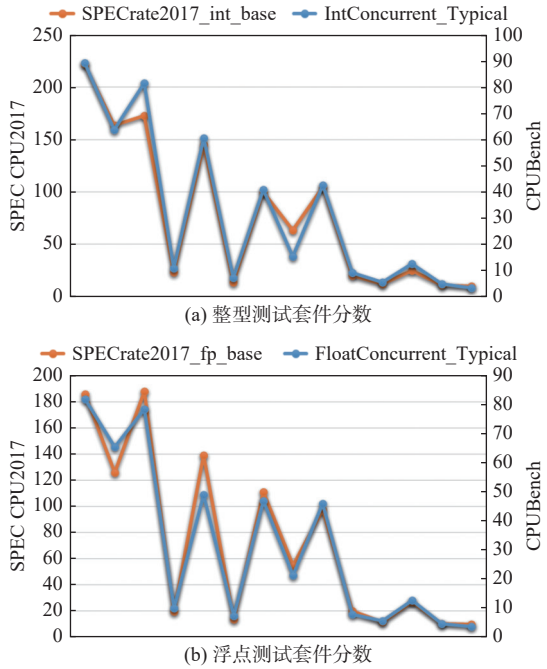

图 12 CPUBench 和 SPEC CPU2017 多任务得分趋势

Fig. 12 Multitasking score trends for CPUBench and SPEC CPU2017

在 14 个不同的测试平台上的结果表明,两款工具的测试分数曲线显示出良好的趋势一致性,间接验证了 CPUBench 作为一款通用计算 CPU 评测基准工具的可用性与合理性.总之,CPUBench 可用于衡量不同计算机系统的性能差异,指导通用计算 CPU、服务器等计算产品的设计、优化和规格选型等工作,对国内计算产业的发展具有重要牵引意义.

参考文献:

[1] KOUNEV S, LANGE K D, VON KISTOWSKI J. Systems benchmarking: for scientists and engineers[M]. Cham: Springer, 2022.

[2] LONGBOTTOM R. Whetstone benchmark history and results. 2017. DOI: 10.13140/RG.2.2.26267.77603.

[3] WEICKER R P. Dhrystone: a synthetic systems programming benchmark[J]. Communications of the ACM, 1984, 27(10): 1013-1030. DOI: 10.1145/358274.358283.

[4] WEICKER R P. An overview of common benchmarks[J]. Computer, 1990, 23(12): 65-75. DOI: 10.1109/2.62094.

[5] DONGARRA J J, LUSZCZEK P, PETITET A. The LINPACK benchmark: past, present and future[J]. Concurrency and Computation:Practice and Experience, 2003, 15(9): 803-820. DOI: 10.1002/cpe.728.

[6] WONG P, VAN DER WIJNGAART R. NAS parallel benchmarks I/O version 2.4[R]. Moffet Field, CA:

NASA Ames Research Center, 2003: 91.

[7] PANDA R, SONG S, DEAN J, et al. Wait of a decade: did SPEC CPU 2017 broaden the performance horizon?[C]//2018 IEEE International Symposium on High Performance Computer Architecture (HPCA). Vienna: IEEE, 2018: 271-282. DOI: 10.1109/HPCA.2018.00032.

[8] YASIN A. A top-down method for performance analysis and counters architecture[C]//2014 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS). Monterey: IEEE, 2014: 35-44. DOI: 10.1109/ISPASS.2014.6844459.

[9] MAĆKIEWICZ A, RATAJCZAK W. Principal components analysis (PCA)[J]. Computers & Geosciences, 1993, 19(3): 303-342. DOI: 10.1016/0098-3004(93)90090-R.

[10] SONG S, WU Q Z, FLOLID S, et al. Experiments with SPEC CPU 2017: similarity, balance, phase behavior and SimPoints[R]. Austin: LCA Group, Department of Electrical and Computer Engineering, The University of Texas at Austin, 2018.

[11] LIMAYE A, ADEGBIJA T. A workload characterization of the SPEC CPU2017 benchmark suite[C]//2018 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS). Belfast: IEEE, 2018: 149-158. DOI: 10.1109/ISPASS.2018.00028.

[12] SINGH S, AWASTHI M. Memory centric characterization and analysis of SPEC CPU2017 suite[C]//Proceedings of the 2019 ACM/SPEC International Conference on Performance Engineering. Mumbai: ACM, 2019: 285-292. DOI: 10.1145/3297663.3310311.

作者简介:

逯海涛　男,(1993-),硕士,工程师.研究方向为集成电路测试评价技术及相关标准化.

任　翔　男,(1974-),高级工程师.研究方向为集成电路测试评价技术及相关标准化.

钟伟军(通讯作者)　男,(1974-),博士,高级工程师.研究方向为集成电路测试评价技术及相关标准化.

E-mail: zhongwj@cesi.cn.

赵　鑫　男,(1981-),硕士,高级工程师.研究方向为集成电路测试评价技术及相关标准化.

尹　航　男,(1990-),硕士,工程师.研究方向为集成电路测试评价技术及相关标准化.