openEuler增强特性介绍

- openEuler是一款基于Linux内核的通用操作系统；
- 为了充分发挥鲲鹏处理器的优势，openEuler在多核调用技术、软硬件协同、轻量级虚拟化、指令级优化和智能优化引擎等方面做了增强。

CPU 调度

- 操作系统需要合理安排系统中多个进程的执行顺序，使其轮换占用CPU资源，保证其并发性；
- 当系统中只有一个CPU时，只有一个进程会处于执行状态，而其它待运行进程则处于就绪状态。操作系统在必要的时候会中断当前进程的执行，并选择就绪状态中的一个进程让其占用CPU，这个过程被称为**CPU调度**(也叫**进程调度**)；
- 完成选择进程任务的程序被称为**调度程序**(Scheduler)。调度程序根据**调度策略**(Scheduling Policy)来决定在什么时候以什么样的方式选择一个新进程占用CPU；
- 当系统中拥有多个CPU或多核CPU时，调度程序还要考虑多(核)CPU之间的数据共享和数据同步。

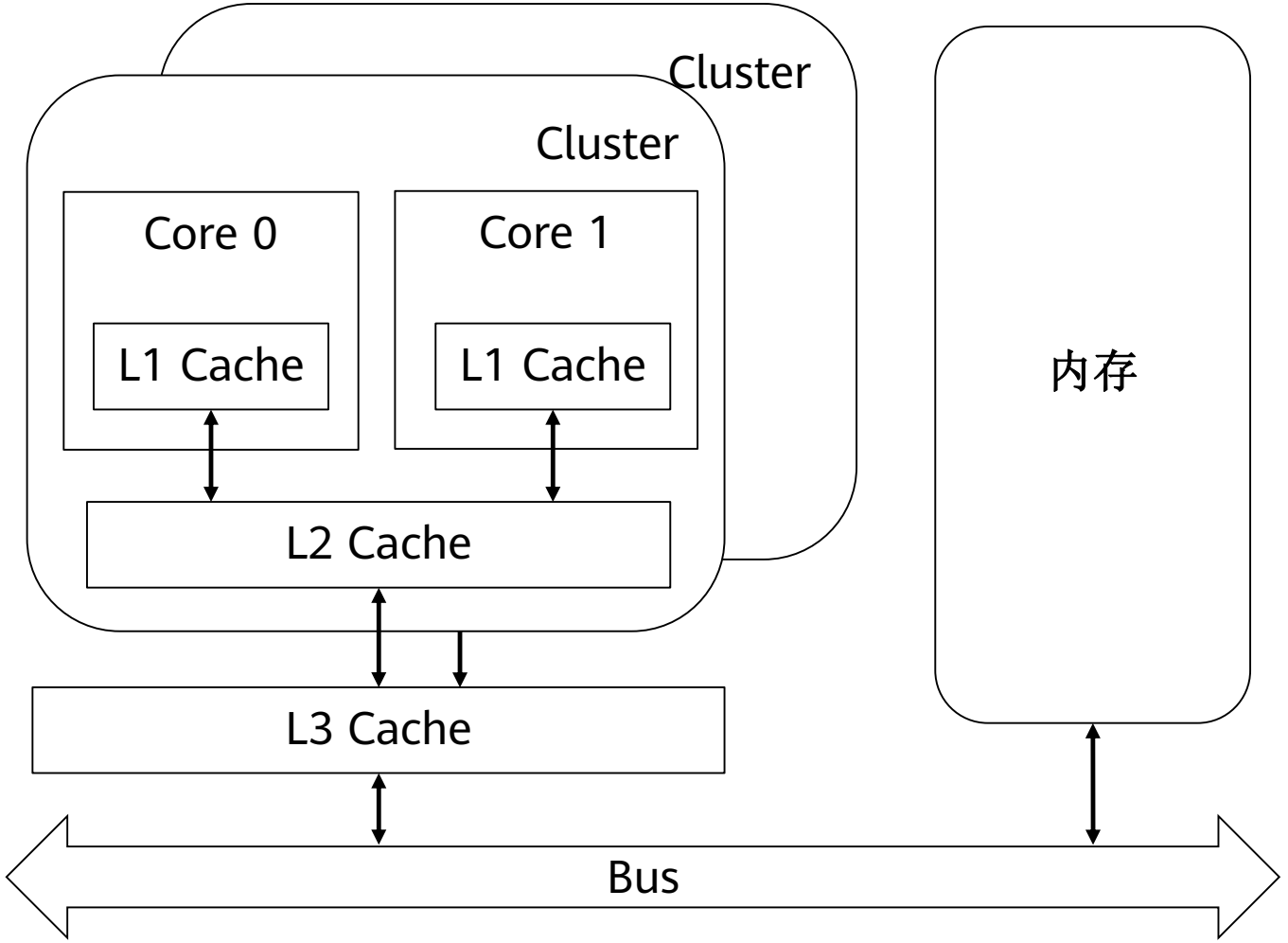
常见调度算法

- 先进先出(First In First Out，FIFO)，或先来先服务(First Come First Served，FCFS)；
- 最短进程优先(Shortest Job First，SJF)；
- 轮转(Round-Robin，RR)调度；
- 优先级调度；
- openEuler使用了上述先进先出、轮转调度以及优先级调度算法。这些调度算法被联合使用之后，用于对实时进程的调度。

CFS 调度

- openEuler中大部分进程是普通进程，它们更需要调度的公平性。
- openEuler中一种核心的进程调度策略为标准轮流分时调度策略，其采用的是CFS(Completely Fair Scheduler)算法。
- CFS调度算法使用了时间片和优先级的概念，并且引入了虚拟运行时间，使得操作系统按照当前系统的负载和普通进程的优先级给该进程分配CPU使用的比例，从而确保了普通进程的相对公平。

多核处理器系统

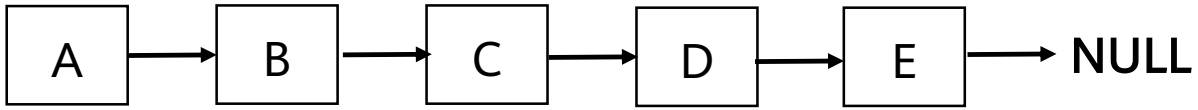


ARMv8架构示意图

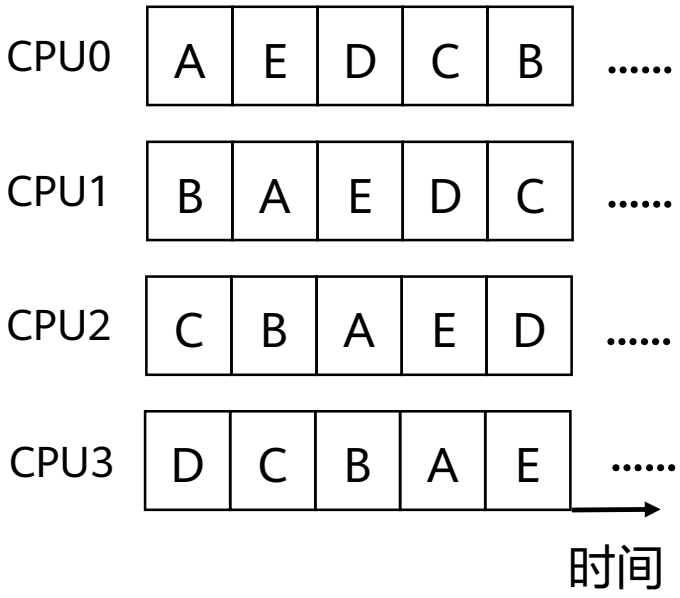
多核处理器系统面临的问题

- 多核处理器缓存和主存的关系发生变化之后，系统主要会面临如下问题：
  - 缓存一致性问题；
  - 缓存亲和性问题；
  - 核间数据共享；
  - 负载均衡。

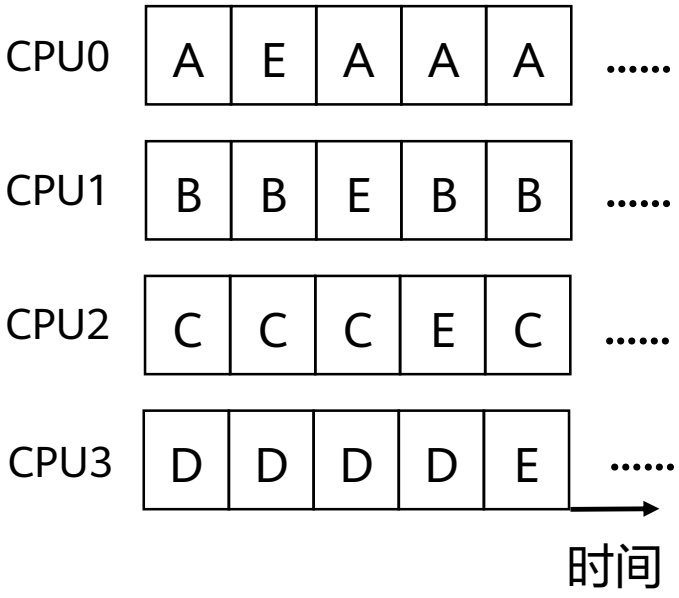
单队列调度



调度队列情况：5个就绪进程



单队列调度情况(策略一)



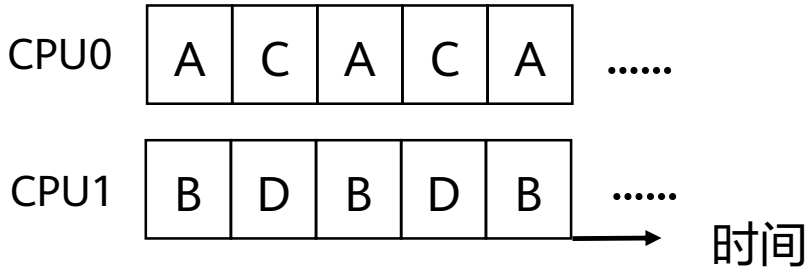
单队列调度情况(策略二)

策略一：每个CPU使用轮转调度算法选择下一个要执行的进程。进程会在不同CPU间转移，违背了缓存亲和性。

策略二：引入亲和度机制尽可能地让进程在同一个CPU上执行。牺牲了某些进程的亲和性(如图所示E)。

多队列调度

openEuler使用多队列调度策略。图中队列Q0由CPU0维护，Q1由CPU1维护。



采用轮转调度算法，调度之初的情况。

一段时间后，假设A、C两个进程都结束，此时负载失衡。

openEuler中的迁移线程

- 解决多CPU负载不均衡问题最直接的方法就是让就绪进程跨CPU迁移。比如上例中将进程D迁移到CPU0上运行后，CPU0和CPU1则实现了负载均衡。
- 在openEuler中，每个处理器都有一个迁移线程(称为migration/CPUID)，每个迁移线程都有一个由函数组成的停机工作队列。如上例：
  1. CPU0向CPU1的停机工作队列中添加一个工作函数，并唤醒CPU1上的迁移线程；
  2. 该迁移线程不会被其他进程抢占，故其第一时间从停机工作队列中取出函数执行，即将进程从CPU1迁移到CPU0；
  3. 此时已实现负载均衡。

NUMA感知队列自旋锁的引入

- 不同线程并发地访问共享地址空间时，如果这些线程占用CPU的时机或者顺序不同，产生的计算结果不同。
- 为了解决这个问题，操作系统提供互斥机制与同步机制。其中互斥机制主要使用自旋锁来实现。
- openEuler提供“NUMA感知队列自旋锁”实现互斥机制以减小MUMA体系结构中使用自旋锁的开销。

Qspinlock

- 在队列自旋锁(Qspinlock)机制中，如果跨NUMA(Non Uniform Memory Access Architecture，NUMA)节点进行锁传递，将导致锁变量从一个NUMA节点迁移到另一个NUMA节点。
- 在NUMA体系结构中，访问本地节点比访问远程节点快得多。同时，由于缓存失效，将导致额外的性能开销。因此，如果尽可能连续地在同一个NUMA节点上进行传递锁，将有效地降低使用锁带来的开销。
- 基于这个思想，在NUMA体系结构中，openEuler采用CAN(Compact NUMA-aware Lock)队列代替Qspinlock中的MCS队列。

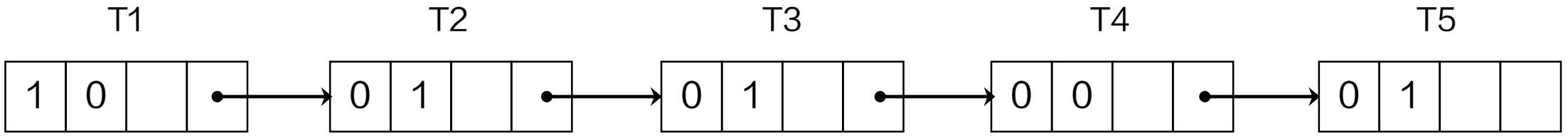
CNA队列

- CNA队列是MCS队列的一种变体。
- MCS将等待获取锁的线程组织在一个队列中，而CNA则将等待获取锁的线程组织为两个队列：一个主队列，一个辅助队列。
- 主队列的线程队头运行在相同的NUMA节点上。辅助队列的线程与主队列队头运行在不同NUMA节点上。

NUMA-aware Qspinlock

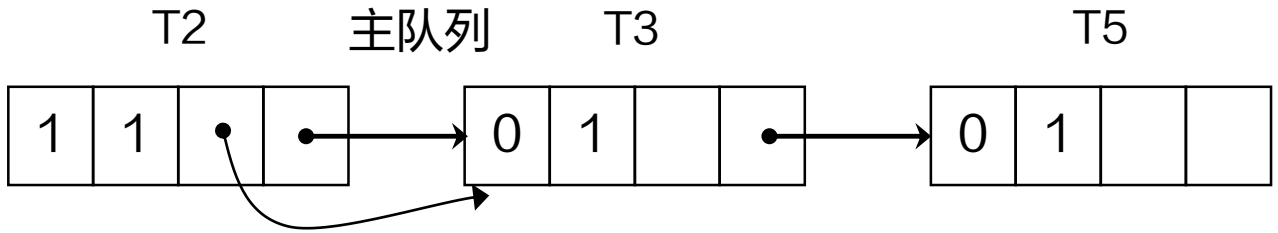
- 当一个线程试图获取CNA锁时，它将先被加入主队列。当锁被释放时，与队头不处于同一个NUMA节点的线程可能将被移动到辅助队列。
- 这种类型的锁称为NUMA感知队列自旋锁，即NUMA-aware Qspinlock。

CAN排队示例 (1)



(a) 主队列的初始状态

(b) 线程T4 成为主队列的头



(c) 辅助队列并入主队列，线程T2 成为主队列的头

CAN排队示例 (2)

- 如上图(a)、(b)、(c)所示：
  - 开始有5个线程排队，线程T1位于序号为0的NUMA节点上并在队头自旋以等待获取锁；
  - 线程T1成功获取锁，遍历主队列寻找队头的继承者。遍历至节点T4时，由于线程T4也位于序号为0的NUMA节点上，故其成为队头的继承者，遍历结束；
  - 线程T1将线程T4的锁状态locked置1，使其成为主队列队头。线程T2与T3位于序号为1的NUMA节点上，故其将被加入到辅助队列中。由于辅助队列当前为空队列，所以线程T2成为辅助队列的队头，线程T3成为辅助队列的队尾；
  - 线程T4成功获取锁时，其遍历主队列寻找继承者。线程T4的后继节点T5位于节点序号为1的NUMA节点上，主队列中没有线程T4期望找到的继承者，辅助队列的节点被并入主队列中。辅助队列的队头线程T2成为主队列的队头，线程T5成为队尾。

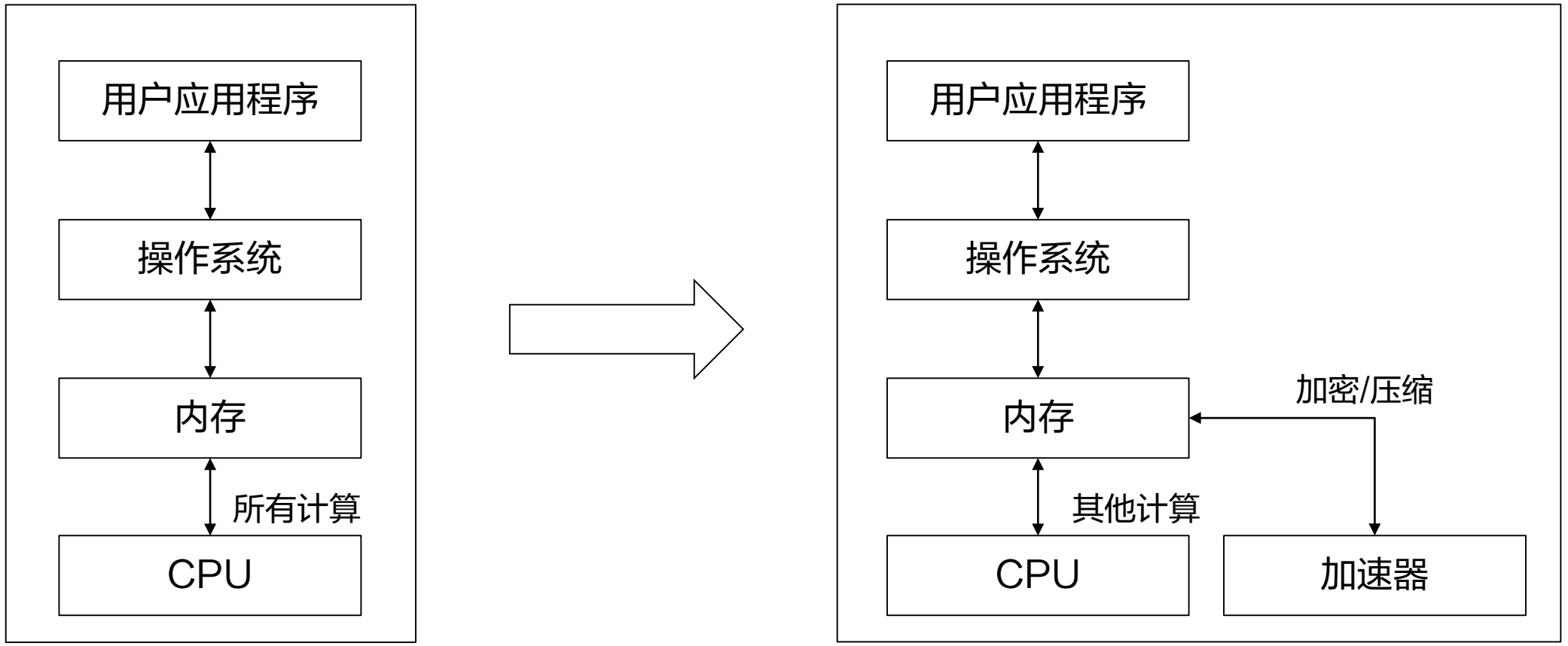
cna_pass_lock()函数 (2)

- CNA与MCS队列的主要区别在于：在锁释放时，选择哪个节点来拿锁；
- 即队头在获取锁后，选择哪个节点作为队头的继承者；
- 在实现上，NUMA-aware Qspinlock将MCS的锁传递函数mcs_pass_lock()替换为CNA的锁传递函数cna_pass_lock()。

KAE

- openEuler通过提供鲲鹏加速引擎(Kunpeng Accelerator Engine，KAE)插件，使能Kunpeng硬件加速能力，包括：
  - 对称/非对称加密
  - 数字签名
  - 压缩解压缩等算法，用于加速SSL/TLS应用和数据压缩

加速器协同的一个例子



加速库

- 加速库结合计算机体系结构，通过改变软件代码的数据结构和算法，从而提升基础软件库的计算效率；或者利用计算机芯片多样性，通过软件开发，充分发挥专有芯片高性能的特点，提升特定领域基础功能的计算效率。
- 从上述加速库的定义上看出，加速库可以分为软件加速库和硬件加速库两大类，硬件加速库也叫做加速器，所以术语加速库有时特指软件加速库。
- 软件加速库的实现只需修改软件代码；而硬件加速库需要专有硬件支持，通过硬件驱动使能。

KAE逻辑架构

- 以鲲鹏920 CPU为例，其加速系统逻辑架构包括：
  - 芯片加速器子系统、BIOS子系统和BMC子系统为TaiShan硬件产品自带子系统；
  - 加速器驱动子系统，向上层提供各子加速器模块统一的驱动接口，是本系统的核心子系统；
  - 应用库子系统(OpenSSL/zlib)，向上层应用提供标准接口；
  - 应用系统(APPs)，指上层应用系统，包括大数据应用、Web应用等，属于用户层面系统。

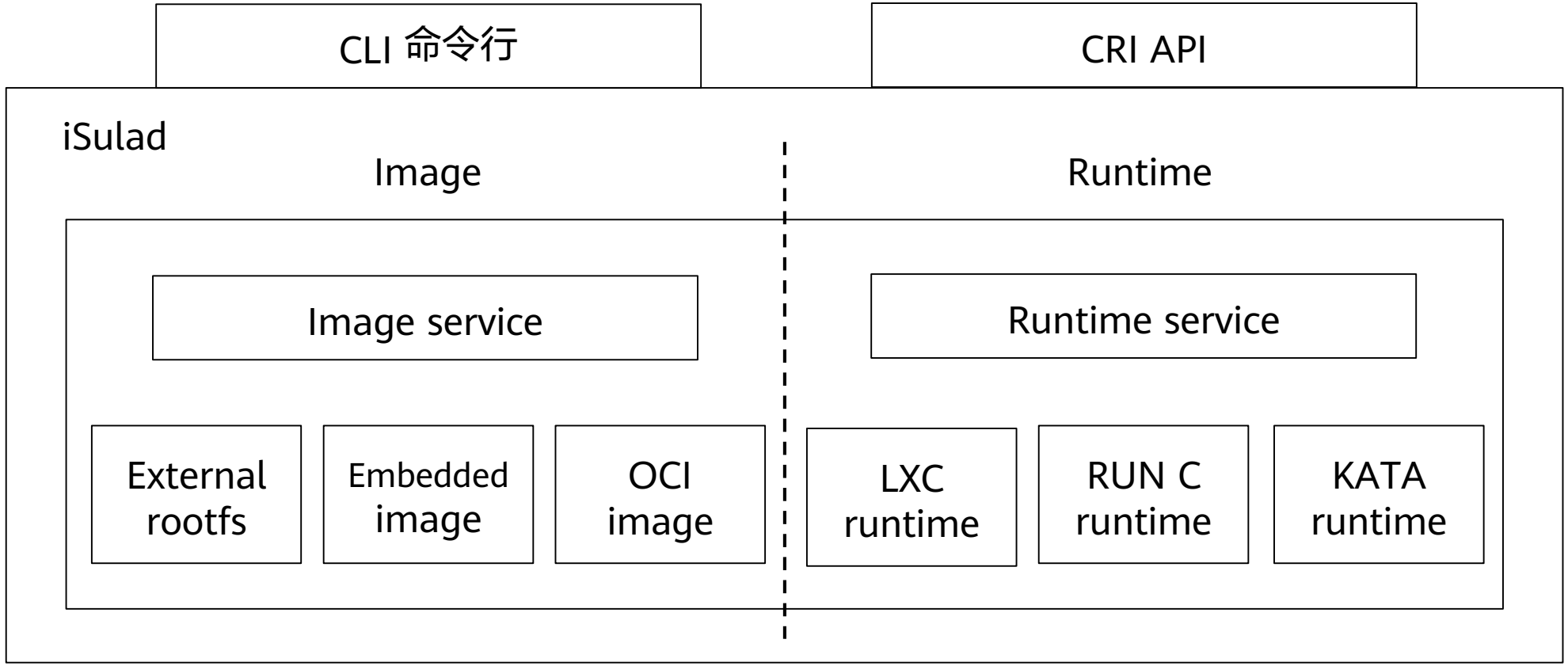
KAE 的安装

- 可以选择以下任何一种方式安装KAE：
  - rpm安装；
  - dpkg安装；
  - 源码安装。通用的源码安装方式，使用configure进行编译及安装配置，使用make进行源码编译，使用make install进行安装。

iSula到底是啥?

- 在居住中南美洲亚马逊丛林的巴西原住民眼里，iSula是一种非常强大的蚂蚁，学术上称为“子弹蚁”，因为被它咬一口，犹如被子弹打到那般疼痛，它是世界上最强大的昆虫之一;
- iSula为全量的容器软件栈，包括引擎、网络、存储、工具集与容器OS；iSulad 作为其中轻量化的容器引擎与子弹蚂蚁"小个头、大能量"的形象不谋而合。

iSulad 统一架构



容器统一架构图

iSulad 架构简介

- iSulad是一个轻量级容器引擎，采用C/C++语言实现，相比其它容器引擎，它的内存开销更小，并发性能更高；
- iSulad容器引擎主要包括以下几个模块：
  - 通信模块：支持gRPC/RESTFUL两种通信方式，提供对外通信的能力；
  - 镜像模块：支持OCI标准镜像，提供content/metadata、rootfs及snapshot管理能力；
  - 运行时模块：支持轻量级Runtime(lcr)和OCI标准的Runtime(runc，kata等)。
- 从接口划分：
  - 北向接口：提供CLI(Command Line Interface)接口和CRI接口；
  - 南向接口：提供统一的Runtime管理接口Plugin，支持lcr和OCI两种类型的Runtime；

容器运行与镜像功能简介 (1)

- 现在简要介绍iSuald容器与镜像生命周期管理的基本操作，详细步骤可查看openEuler社区《容器用户指南》中“iSulad容器引擎”节。
- 运行容器：运行容器指创建一个新的容器，并启动该容器。即使用指定的容器镜像创建容器读写层，并且为运行指定的命令做好准备。在创建完成后，使用指定的命令启动该容器。

```
$ isula create -it busybox
9c2c13b6c35f132f49fb7ffad24f9e673a07b7fe9918f97c0591f0d7014c713b
$ isula start 9c2c13b6c35f
```

容器运行与镜像功能简介 (2)

- 暂停/恢复容器：暂停容器指通过freezer cgroup挂起指定容器中的所有进程，恢复容器为暂停容器的逆过程，用于恢复被暂停容器中所有进程。

```
$ isula pause 9c2c13b6c35f
9c2c13b6c35f132f49fb7ffad24f9e673a07b7fe9918f97c0591f0d7014c713b
$ isula unpause 9c2c13b6c35f
9c2c13b6c35f132f49fb7ffad24f9e673a07b7fe9918f97c0591f0d7014c713b
```

容器运行与镜像功能简介 (3)

- 销毁容器：销毁容器指停止并删除容器。首先向容器中的首进程发送SIGTERM信号以通知容器退出，如果在指定时间(默认为10s)内容器未停止，则再发送SIGKILL信号时主动杀死容器进程。无论容器以何种方式退出，最后都会回收和删除该容器所占用资源。

```
$ isula stop 9c2c13b6c35f
9c2c13b6c35f132f49fb7ffad24f9e673a07b7fe9918f97c0591f0d7014c713b
$ isula rm 9c2c13b6c35f
9c2c13b6c35f132f49fb7ffad24f9e673a07b7fe9918f97c0591f0d7014c713b
```

容器运行与镜像功能简介 (4)

- 从镜像仓库拉取容器镜像：拉取容器镜像指的是从远程镜像仓库拉取镜像到本地主机。这里的“远程”和“本地”是相对的，可能都是本地主机。

```
$ isula pull localhost:5000/official/busybox
Image "localhost:5000/official/busybox" pulling
Image "localhost:5000/official/busybox@sha256:bf510723d2cd2d4e3f5ce7
e93bf1e52c8fd76831995ac3bd3f90ecc866643aff" pulled
```

容器运行与镜像功能简介 (5)

- 删除容器镜像：删除容器镜像指的是从本地保存的容器镜像中删除指定的容器镜像。

```
$ isula rmi busybox
Image "busybox" removed
```

openEuler智能调优 — A-Tune

- 当Linux或Windows这种通用操作系统的某个功能机制无法保证对所有场景均有益时，设计者就会在系统中提供一个可配置参数，并确保该参数的默认配置对大部分通用场景有益，而使用者通过更改参数配置来满足特定的使用场景需求；
- openEuler正是基于Linux内核的，举例来说，其sysctl命令的参数超过1000个，而完整的IT系统从最底层的CPU、加速器、网卡，到编译器、操作系统、中间件框架，再到上层应用，可调节对象超过7000个；
- 自调优工具A-Tune旨在让操作系统能够满足不同应用场景的性能诉求，降低性能调优过程中反复调参的人工成本，提升性能调优效率。

A-Tune的整体架构

- A-Tune整体上是一个C/S架构；
- 客户端atune-adm是一个命令行工具，通过gRPC协议与服务端atuned进程进行通讯；
- 服务端中atuned包含了一个前端gRPC服务层(采用golang实现)和一个后端服务层；
- gRPC服务层负责优化配置数据库管理和对外提供调优服务，主要包括智能决策(analysis)和自动调优(tuning)；
- 后端服务层是一个基于Python实现的HTTP服务层，包含了MPI(Model Plugin Interface)/CPI(Configurator Plugin Interface)和AI引擎；
- MPI/CPI负责与系统配置进行交互，AI引擎负责对上层提供机器学习能力。

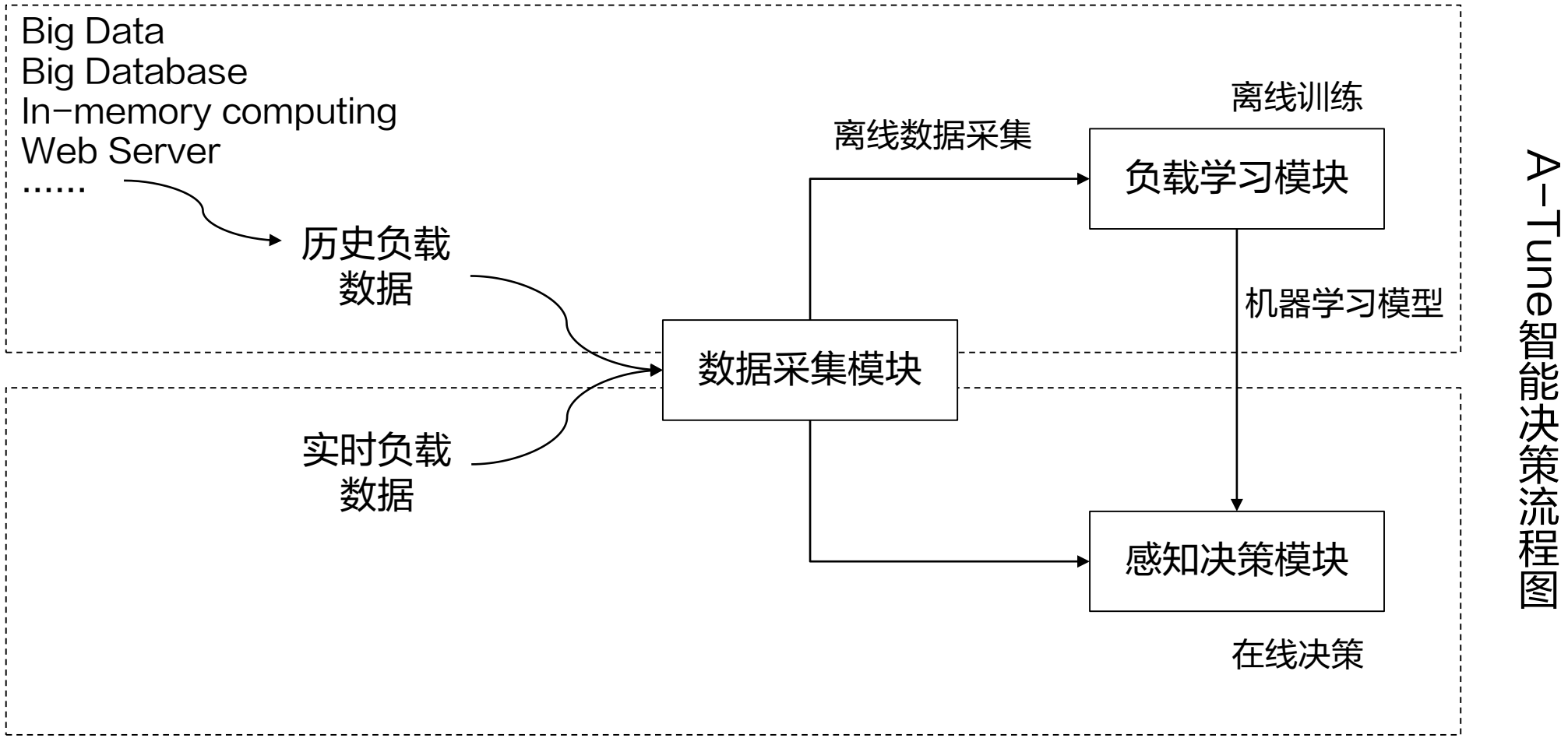
A-Tune的两个能力

- A-Tune目前主要提供智能决策和自动调优两个能力；
- 智能决策是通过采集系统数据，并通过AI引擎中的聚类和分类算法对采集到的数据进行负载识别，然后从优化配置数据库中提取优化配置，选取适合当前系统业务负载的优化配置；
- 自动调优是基于系统或应用的配置参数及性能评价指标，利用AI引擎中的参数搜索算法，反复迭代，最终得到性能最优的参数配置；

两个能力模块的不同

- 自动调优模块应对的业务场景和负载的历史数据样本较小(甚至无历史数据样本)，需要探索最佳的参数配置来优化系统性能；
- 智能决策模块的参数调优通常是针对某一种类型的业务场景和负载，其优化程度取决于历史数据，粒度也相对较大；
- 自动调优模块可为单一业务场景和特定负载实现定向参数调优，其优化更具有针对性，粒度也相对较小，能实现系统参数配置的进一步优化。

智能决策系统流程



A-Tune智能决策流程图

智能决策系统

- 智能决策系统基于openEuler的实时负载数据，识别负载特征并调整系统相关参数。其主要包含数据采集模块、负载学习模块和感知决策模块，并包括离线训练阶段和在线决策阶段；
  - 在离线训练阶段，智能决策系统通过数据采集模块，收集openEuler中不同业务场景运行时的历史负载数据，整理为有监督的离线负载数据集。负载学习模块则在离线负载数据集的基础上，进行聚类分析及业务负载特征分类训练，生成对应的机器学习模型，并将不同类型的负载映射到其最优的系统参数配置；
  - 在在线决策阶段，智能决策系统首先通过数据采集模块采集操作系统当前的实时负载数据，并根据维度将数据整理为若干组在线数据样本。感知决策模块将在线数据样本作为机器学习模型的输入，推理出当前系统负载的聚类、分类结果，并识别出业务负载的瓶颈点，根据业务当前的负载瓶颈点及类型，来调节对应的操作系统参数。

负载学习模块

- 负载学习模块实现数据处理、负载瓶颈点识别聚类和负载特征分类建模三项功能；
  - 数据处理：经过数据预处理、统计分析、特征选择，建立可供训练学习的标准数据集；
  - 瓶颈点聚类：根据操作系统中不同的资源维度进行瓶颈点聚类分析；
  - 负载特征分类：基于聚类分析结果建立负载特征分类模型。

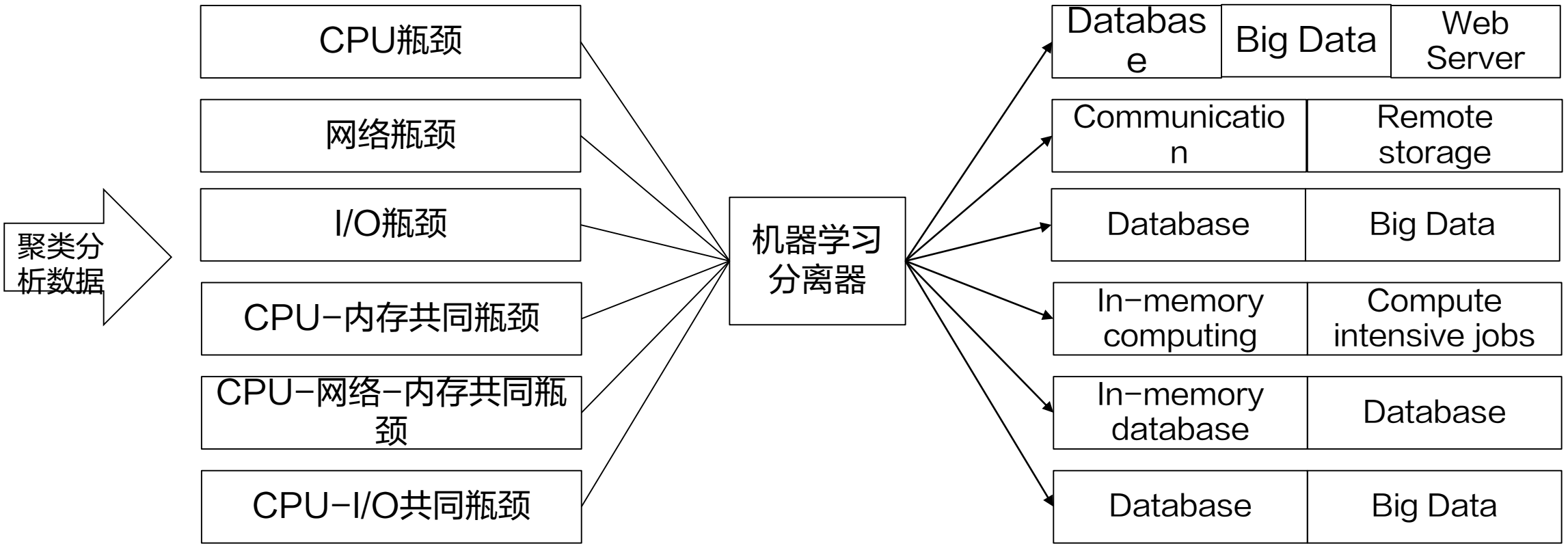
负载学习模块的数据处理

- openEuler上运行的常见业务场景包括大数据、内存密集型计算、数据库、网络服务器等；
- 负载学习模块将从软件资源和硬件资源两个角度，收集不同业务在运行过程中所涉及的特征以及具体生命周期数据；
- 数据样本集涉及的主要特征维度包括CPU、内存、网络等资源的利用率、饱和度及性能等数据。

负载学习模块的瓶颈点聚类分析

- 操作系统对于硬软件资源分配的高压力与瓶颈点可能会直接造成业务的性能下降：
- 操作系统可以根据应用负载数据中所反映出的资源瓶颈点实现合理化、最优化配置；
- 智能决策系统首先根据训练样本集进行无标签分析，通过无监督的聚类学习来分析训练样本集中不同业务场景的瓶颈点位置；
- openEuler根据资源类别，从CPU、I/O、网络、内存四个角度分别对将训练样本集进行聚类分析，将训练样本根据维度数据是否达到瓶颈点做出相似性归类，这样可以得到可解释性强的聚类分析结果。

负载学习模块的负载特征分类



负载特征分类图

负载特征分类

- 对于瓶颈点相似的业务场景，不同的业务负载特征对系统参数的最优配置有较大的影响。比如，内存计算和网络服务器在高压力场景下的业务都存在CPU维度的性能瓶颈，但其负载特征显然不是完全一致的；
- 因此，负载特征分类针对相同性能瓶颈的业务，使用监督学习分类模型，基于离线负载数据集，训练出可以对当前负载数据进行业务分类的模型，进而将模型传递到感知决策模块，由感知决策模块调节配置与该负载特征相关的参数。

感知决策模块

- 感知决策模块基于预训练的机器学习模型，实现智能决策系统的在线决策功能：
  1. 将实时采集的系统负载数据输入到训练生成的聚类模型中，判断当前系统负载的瓶颈点；
  2. 将实时数据输入到相应的分类模型中，推理出当前系统负载的具体分类结果；
  3. 根据前面得到的系统负载瓶颈点和业务分类结果，找到对应最优的系统参数配置，并在操作系统中设置生效。

感知决策模块处理流程示例

- 上图展示了openEuler运行内存密集型计算业务specjbb的感知决策处理流程：
  - 智能决策系统实时采集当前负载数据并传递到感知决策模块，将其作为聚类模型的输入；
  - 通过聚类模型分析后，该业务被判定为CPU-内存瓶颈型业务。然后，将负载数据输入到面向CPU-内存瓶颈型业务的支持向量机分类器中，得到推理的分类结果为内存密集型计算类型；
  - 最后，决策模块将根据模型输出的结果，设置内存大页、刷新率等相关的操作系统参数优化配置以实现该应用的性能提升。

自动调优

- 作为A-Tune工具中的另外一个核心功能，自动调优主要针对实时业务场景和负载，利用AI引擎来搜索最佳的系统参数配置。它主要解决如下两类问题：
  - 当前的业务场景和负载的历史数据样本较小(甚至无历史数据样本)，无法通过有效的离线训练获得此类负载的优化配置经验；
  - 自动调优对于系统参数的优化粒度更细，可为单一业务场景和特定负载实现定向参数调优，能实现系统参数配置的进一步优化。
- 操作系统的可调参数数量巨大且业务复杂度极高。当前硬件和基础软件组成的应用环境涉及了高达7000多个配置对象。随着业务复杂度和调优对象数量的增加，参数调优所需的时间成本呈指数级增长(如下图所示)。

自动调参算法

- 传统的基于人工经验的调优方法在应对上述挑战时变得力不从心，亟需一个高效的自动调参算法。当前常见的自动调参算法有Grid Search(网格搜索)、 Random Search(随机搜索)、 Bayesian Optimization(贝叶斯优化)：
  - 网格搜索：又叫穷举搜索，即搜索整个参数空间(在高维空间会遇到维度灾难)；
  - 随机搜索：在不同的参数维度上随机选取参数值进行组合，可能出现效果特别好/差的参数配置。随机搜索的不同尝试之间是相互独立的，无法利用先验知识来选择下一组参数组合；
  - 贝叶斯优化：在调优过程中形成对参数设置和性能之间的关系认知，利用部分先验知识来优化选择下一组试验参数，可以使用尽量少的试验次数找到最优的性能。
- openEuler中采用了基于贝叶斯优化的自动调优技术。

A-tune自动调优流程 (1)

- 上图是A-Tune中自动调优基本流程的一个示例。其目标是对运行在服务端的业务进行自动优化，即服务端需要调节哪些参数，并通过自动调节这些参数，来提升业务的性能；
- openEuler操作系统是通过用户配置好并上传到服务端的yaml文件进行参数空间的配置的。下图是一个优化mysql的场景的客户端配置文件示例，其中有指定客户端benchmark的拉起脚本和获得benchmark结果的命令，并且指定了调节的次数。

A-tune自动调优流程 (2)

- 首先，我们在客户端运行下面的命令：

```
atune-adm tuning mysql-client.yaml
```

- 拉起tuning之后：
  - 客户端会给服务端发送一个请求，继而服务端将可以调节的参数空间发送给负责运行贝叶斯优化的服务器；
  - 贝叶斯优化算法随机选出新的参数设置发给服务端；
  - 服务端将收到的参数进行自动设置，并通知客户端拉起benchmark进行测试；
  - 客户端完成测试之后，收集测试的benchmark结果发送给服务端；
  - 服务端再将结果发给负责运行贝叶斯优化的服务器；

A-tune自动调优流程 (3)

- 拉起tuning之后 (续)：
  - 贝叶斯优化根据收到的benchmark结果和可调节的参数空间，计算出新的参数设置，并发给服务端；
  - 服务端将收到的参数进行自动设置，并通知客户端拉起benchmark进行测试；
  - 以上过程不断循环，直到达到规定的调节次数，完成调优过程；
  - 调优结束后，负责运行贝叶斯优化的服务器将最优的性能和参数设置发回给服务端，服务端再发给客户端显示出来。

思考题

1. 以下那一项说法是错误的？（ ）

   A. openEuler使用了先进先出、轮转调度、优先级调度算法以及 Completely Fair Scheduler 调度算法。

   B. 在openEuler中，每个处理器都有一个迁移线程(称为migration/CPUID)，每个迁移线程都有一个由函数组成的停机工作队列。

   C. 在 NUMA 体系结构中， openEuler 采用 CAN(Compact NUMA-aware Lock) 队列代替Qspinlock中的MCS队列。

   D. openEuler通过提供鲲鹏加速引擎(Kunpeng Accelerator Engine，KAE)插件，使能Kunpeng硬件加速能力。

   E. iSulad是一个轻量级容器引擎，相比其它容器引擎，它的内存开销更小，并发性能更高。

   F. A-Tune从整体上看并不是一个C/S架构。

本章总结

- 本章介绍了 openEuler 相对通用 Linux 操作系统做的增强。涵盖了多核调度技术、NUMA-aware Qspinlock、鲲鹏加速引擎、iSulad 轻量级容器以及智能调优(A-Tune)等几个部分，以供大家参考。

学习推荐

- 《 openEuler操作系统 》·清华大学出版社
- 《 openEuler内核编程技术 》·中国科学院软件研究所
- 《 openEuler应用编程技术 》·中国科学院软件研究所

Thank you.