



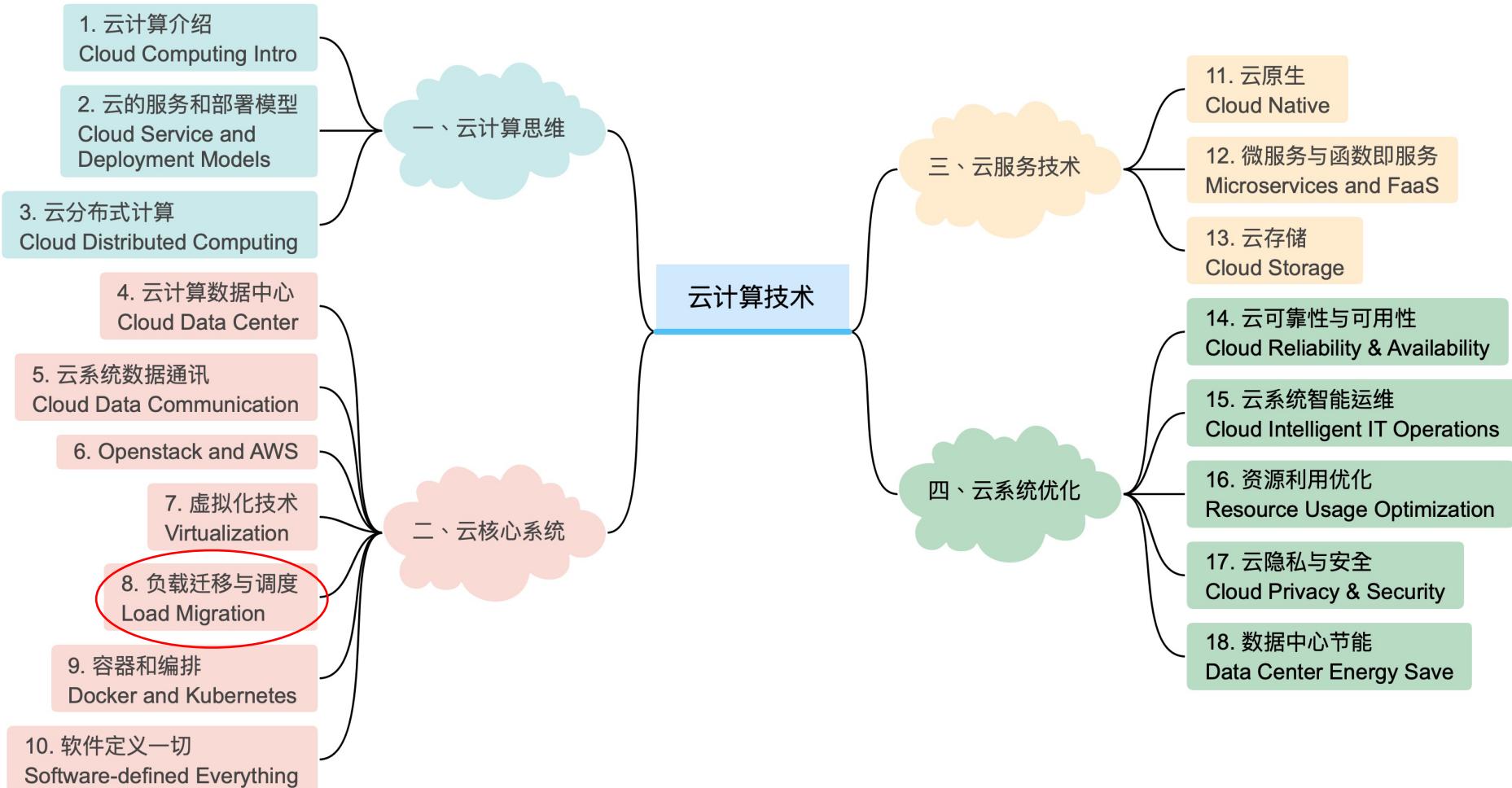
Lecture 07: 云虚拟化技术

SSE316: 云计算技术
Cloud Computing Technologies

陈壮彬

软件工程学院

chenzhb36@mail.sysu.edu.cn



Today's topics

- 虚拟化技术对云的意义

- 虚拟机检查点设置

- 虚拟机调度

- 虚拟机迁移

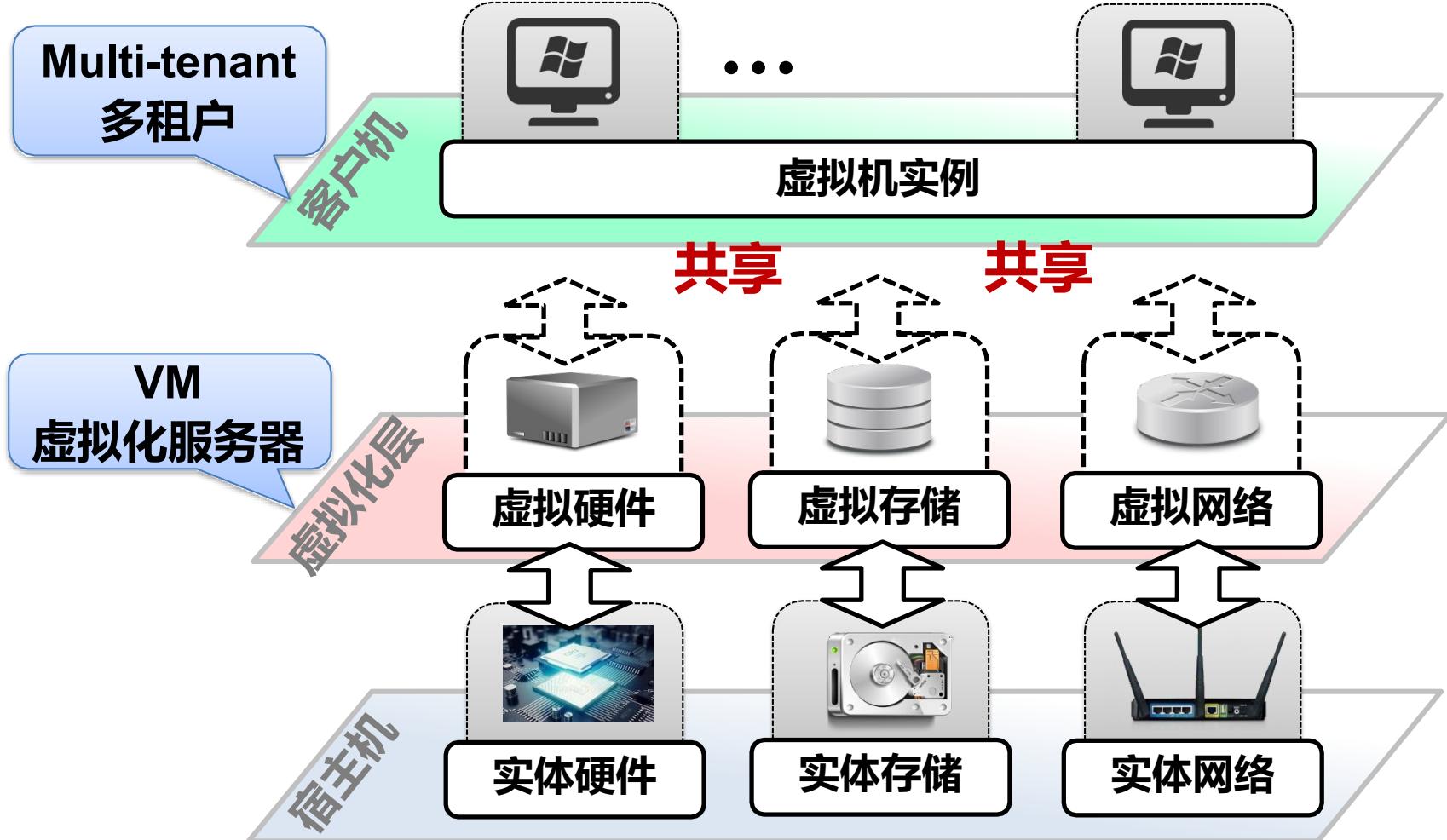
虚拟化技术对云的意义

静态角度：多租户共享

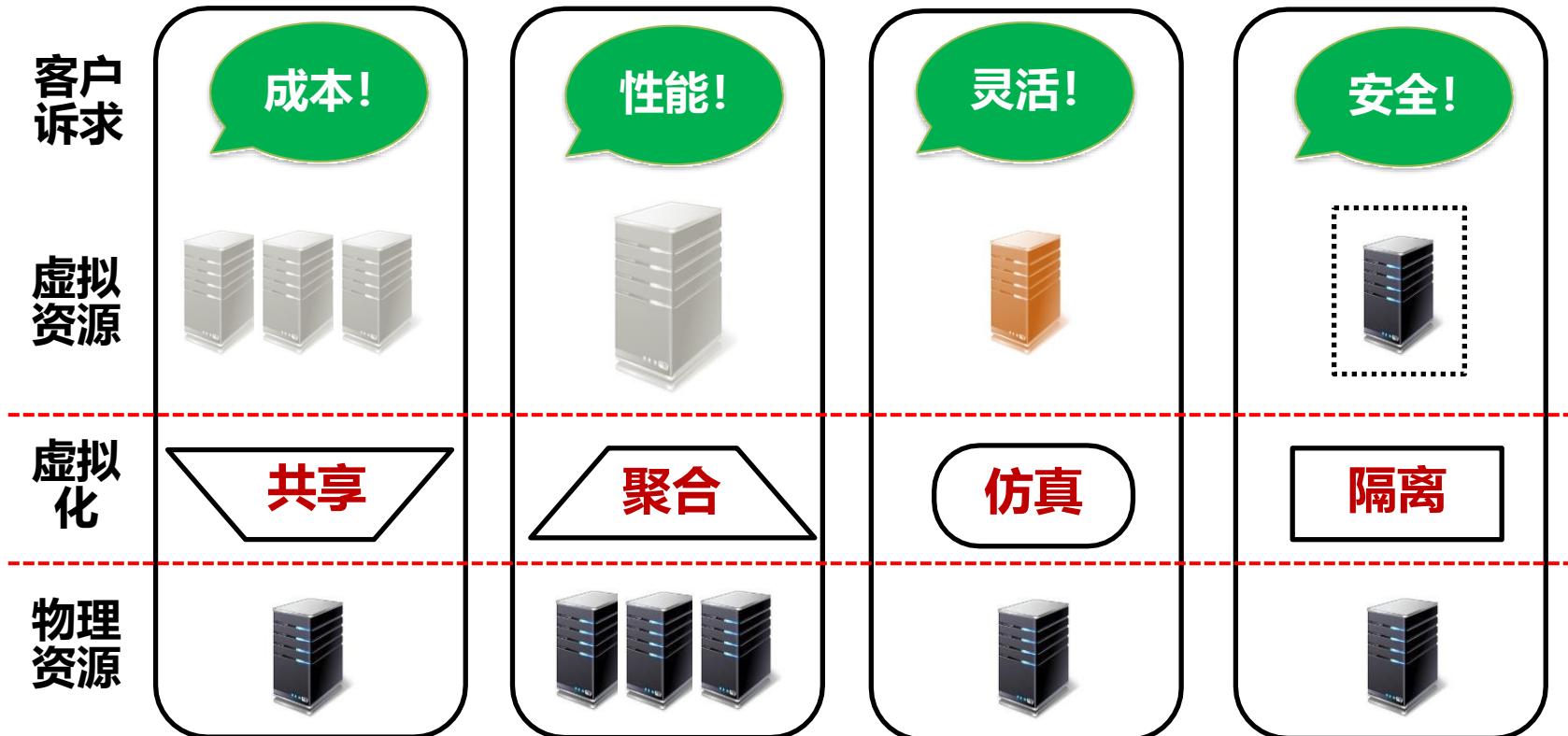


传统：上下层软硬件直接耦合，应用对硬件享有专有权

静态角度：多租户共享

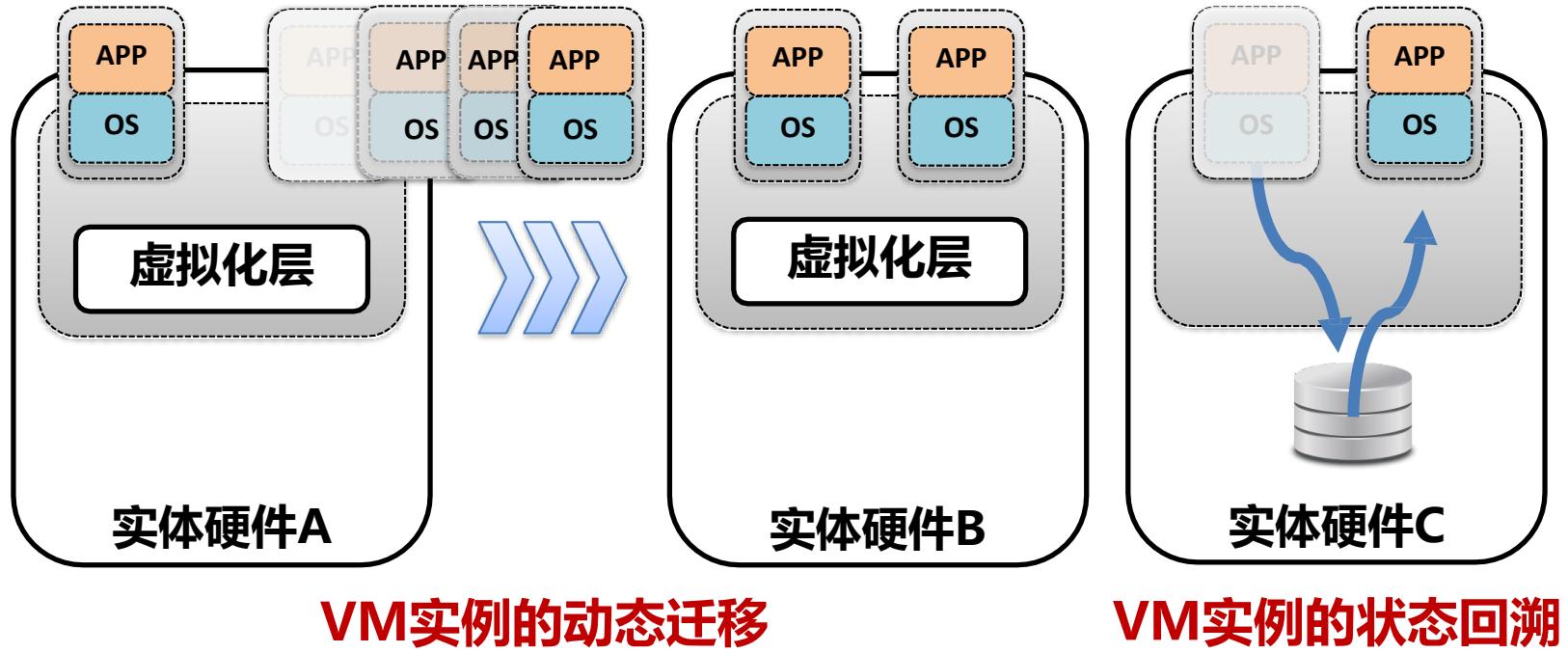


静态角度：优势纵览



虚拟化为IT资源利用提供**全新维度**，是云计算**重要基础**

动态优势



虚拟机状态可以数据形式存储，支持**时空层面**的灵活管理

动态优势：高灵活

虚拟机便于将全部系统状态 (VM states) 封装为镜像 (image)，包括操作系统、应用程序及它们的配置和数据

口虚拟机带来的灵活性

- 用户可以把青睐的应用和计算环境打包在虚拟机中
- 进而动态地将虚拟机信息从一个主机复制到另一个上

动态优势：高可用

虚拟机便于创建检查点 (checkpoint)，也称快照 (snapshot)，包括CPU、内存、设备、磁盘状态等

□ 虚拟机带来的可用性

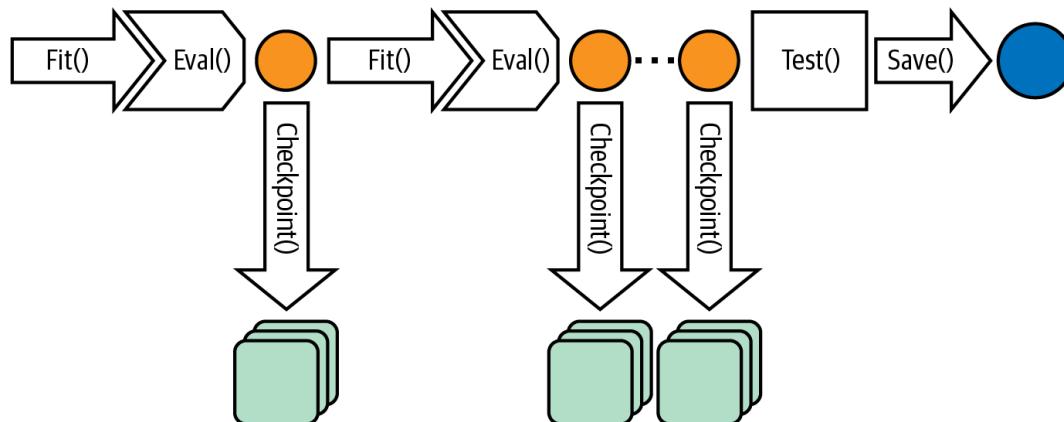
- 通过周期性地制造检查点来捕获虚拟机信息
- 可借助检查点恢复到最近状态 (Checkpoint gives you the ability to roll a VM back to the state in which it existed at an earlier point in time)

检查点 Checkpoint



Overview

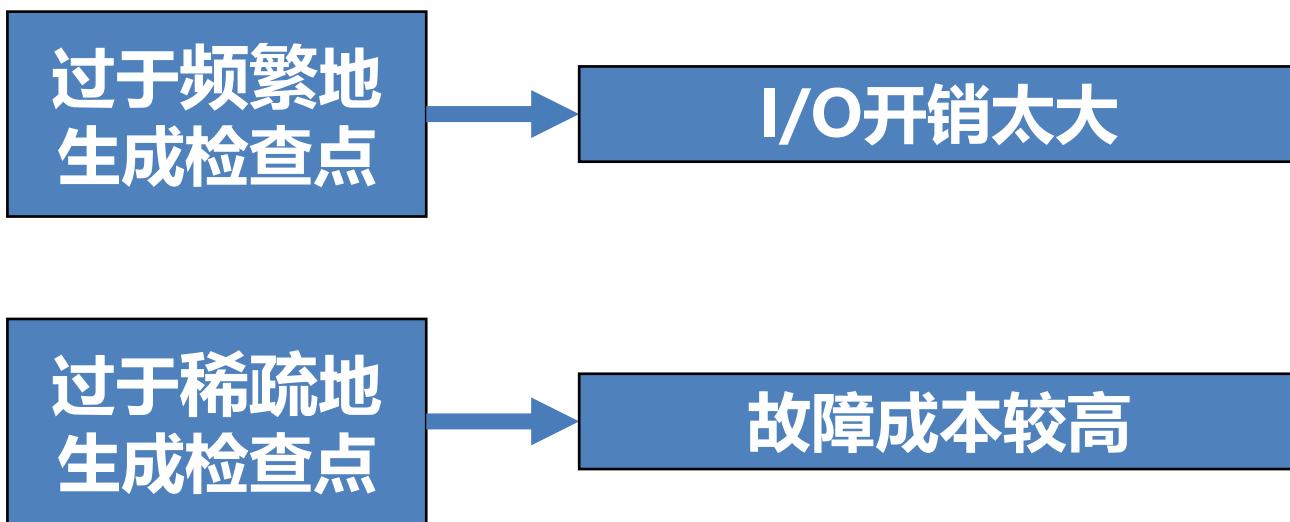
HDFS Snapshots are read-only point-in-time copies of the file system. Snapshots can be taken on a subtree of the file system or the entire file system. Some common use cases of snapshots are data backup, protection against user errors and disaster recovery.



Deep learning model training – Checkpoint()

虚拟机检查点设置

在检查点方面，一个主要问题在于确定最佳的生成周期，即两个连续检查点的间隔时间。



检查点确定算法

在检查点和回滚的开销分析：Young一阶模型

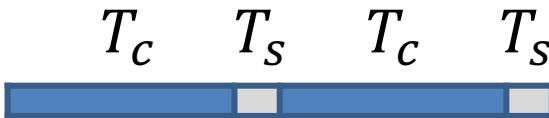
原始应用：



检查点生成间隔 T_c

创造检查点开销 T_s

采用检查点：



如何最小化由于检查点设置以及故障所导致的时间损失？

- 假定生成检查点的时间间隔为 T_c ，所需的时间固定为 T_s ，即检查点文件的大小固定
- 故障可在应用运行过程中的任何时候发生，包括 T_c 或 T_s 期间，无论何种情况，应用将在上一个完整检查点继续运行

检查点确定算法

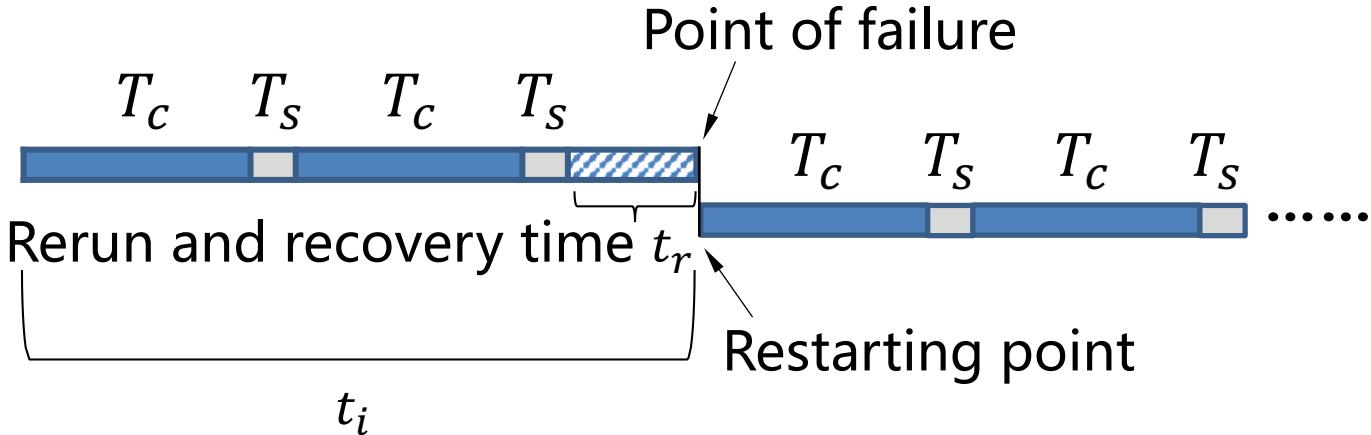
在检查点和回滚的开销分析：Young一阶模型

原始应用：



检查点生成间隔 T_c ■
创造检查点开销 T_s □

采用检查点：



- 假设应用在某一次故障发生前的 t_i 时间内共完成了 n 段执行周期，即 n 次 T_c 和 T_s
- 在第 $n + 1$ 个执行周期中发生故障导致系统回滚。应用需要 t_r 时间加载检查点并重新执行至上一个检查点的部分

检查点确定算法

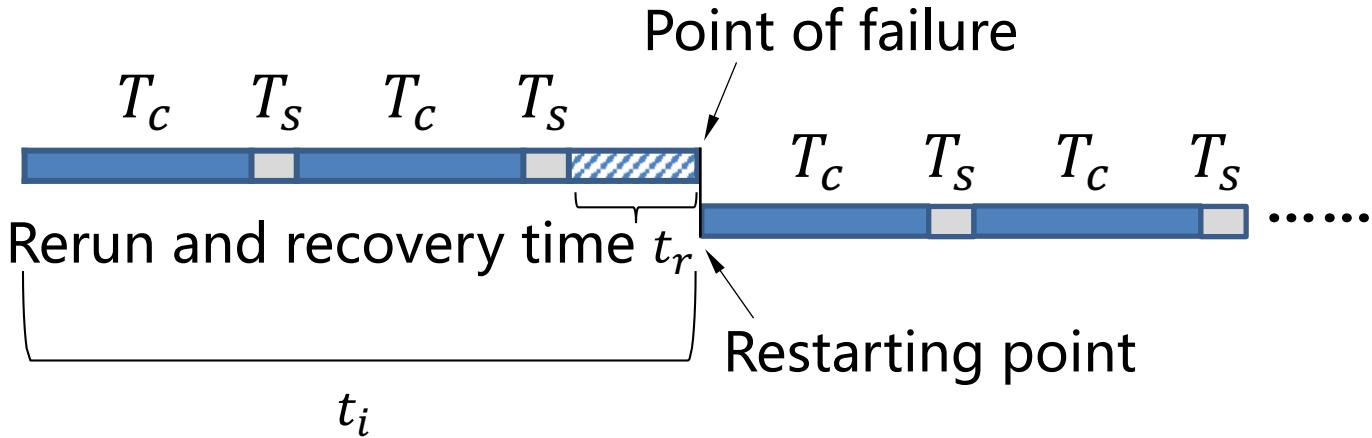
在检查点和回滚的开销分析：Young一阶模型

原始应用：



检查点生成间隔 T_c ■
创造检查点开销 T_s □

采用检查点：



- 相对于原始应用来说，设置检查点的应用在处理上述故障过程中，一共多花费了 $(nT_s + t_r)$ 时间，等于 $(t_i - nT_c)$

检查点确定算法

在检查点和回滚的开销分析：Young一阶模型

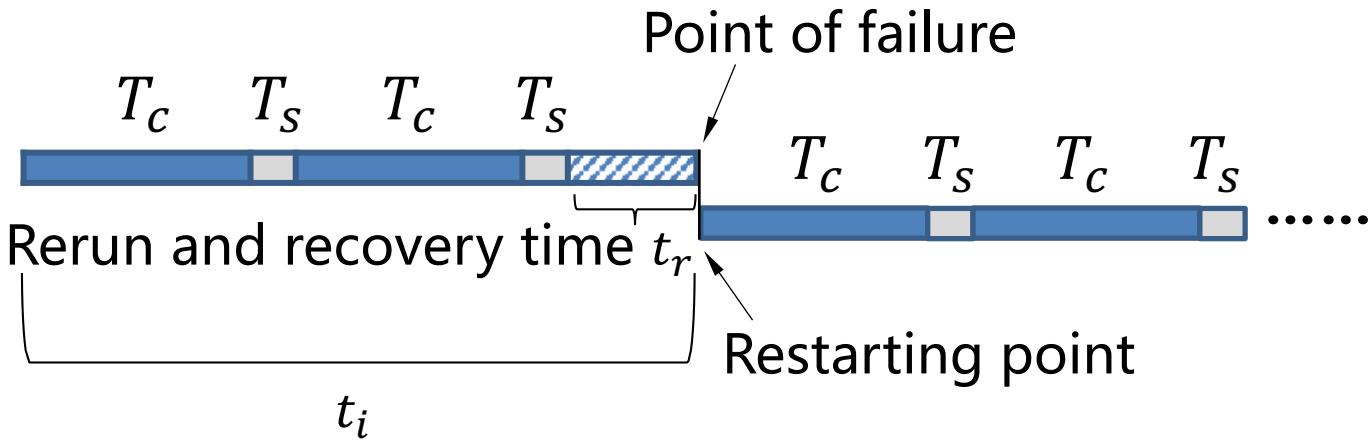
原始应用：



检查点生成间隔 T_c 

创造检查点开销 T_s 

采用检查点：



- 假设故障发生是随机的，符合参数为 λ 的泊松过程，则平均故障间隔 (Mean Time Between Failure) 为 $T_f = 1/\lambda$
- 则故障间隔时间 x 的密度函数为 $P(x) = \lambda e^{-\lambda x}$ ，意味着故障间隔时间为 t_j 的概率为 $\lambda e^{-\lambda t_j} \Delta t$ ，其中 $t < t_j < t + \Delta t$

检查点确定算法

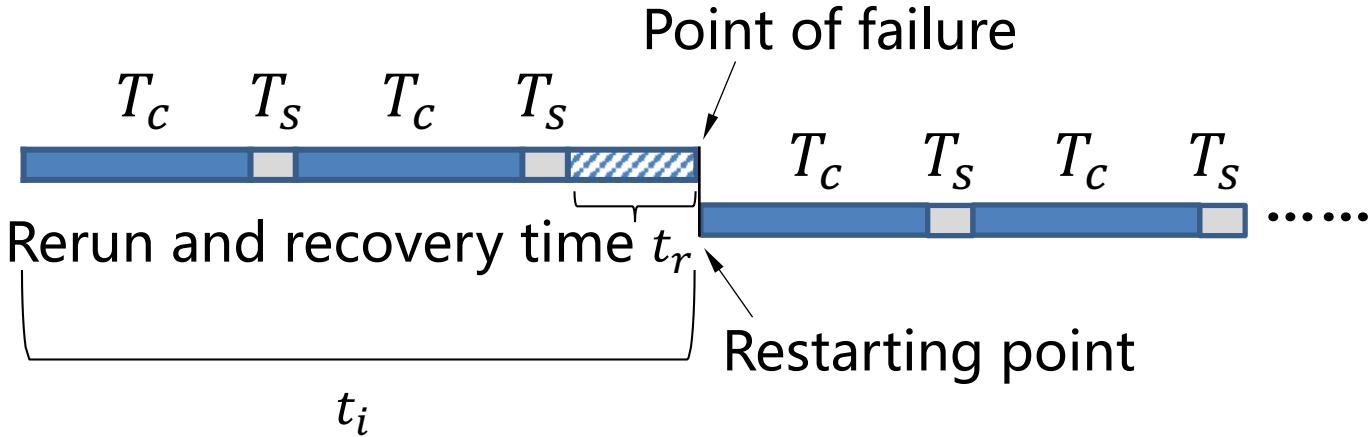
在检查点和回滚的开销分析：Young一阶模型

原始应用：



检查点生成间隔 T_c ■
创造检查点开销 T_s □

采用检查点：



□ 最终，由于检查点设置以及故障所导致的时间损失 T_l 可为：

$$T_l = \sum_{n=0}^{\infty} \int_{n(T_c+T_s)}^{(n+1)(T_c+T_s)} (t - nT_c)(\lambda e^{-\lambda t}) dt$$

检查点确定算法

在检查点和回滚的开销分析：Young一阶模型

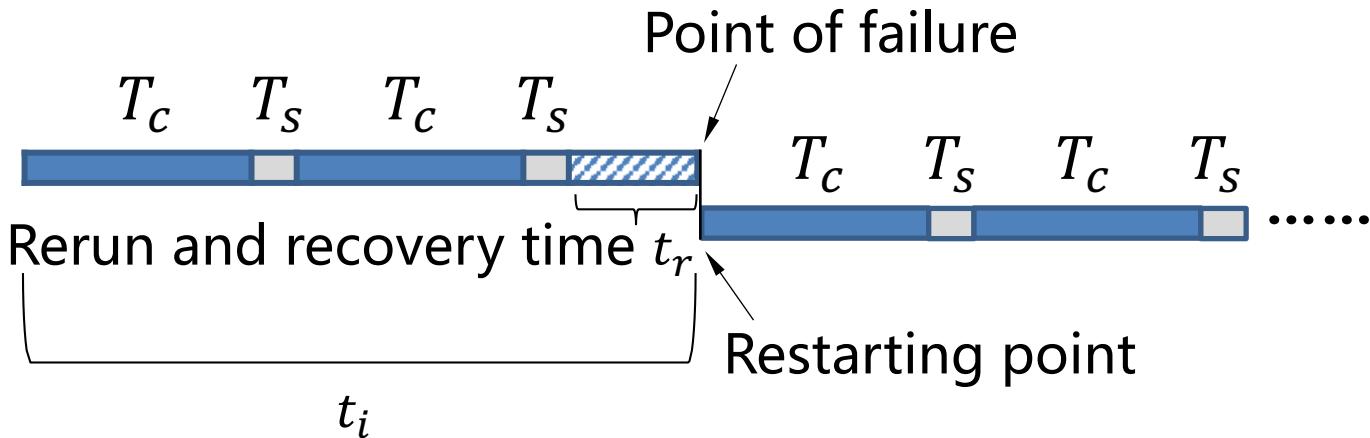
原始应用：



检查点生成间隔 T_c

创造检查点开销 T_s

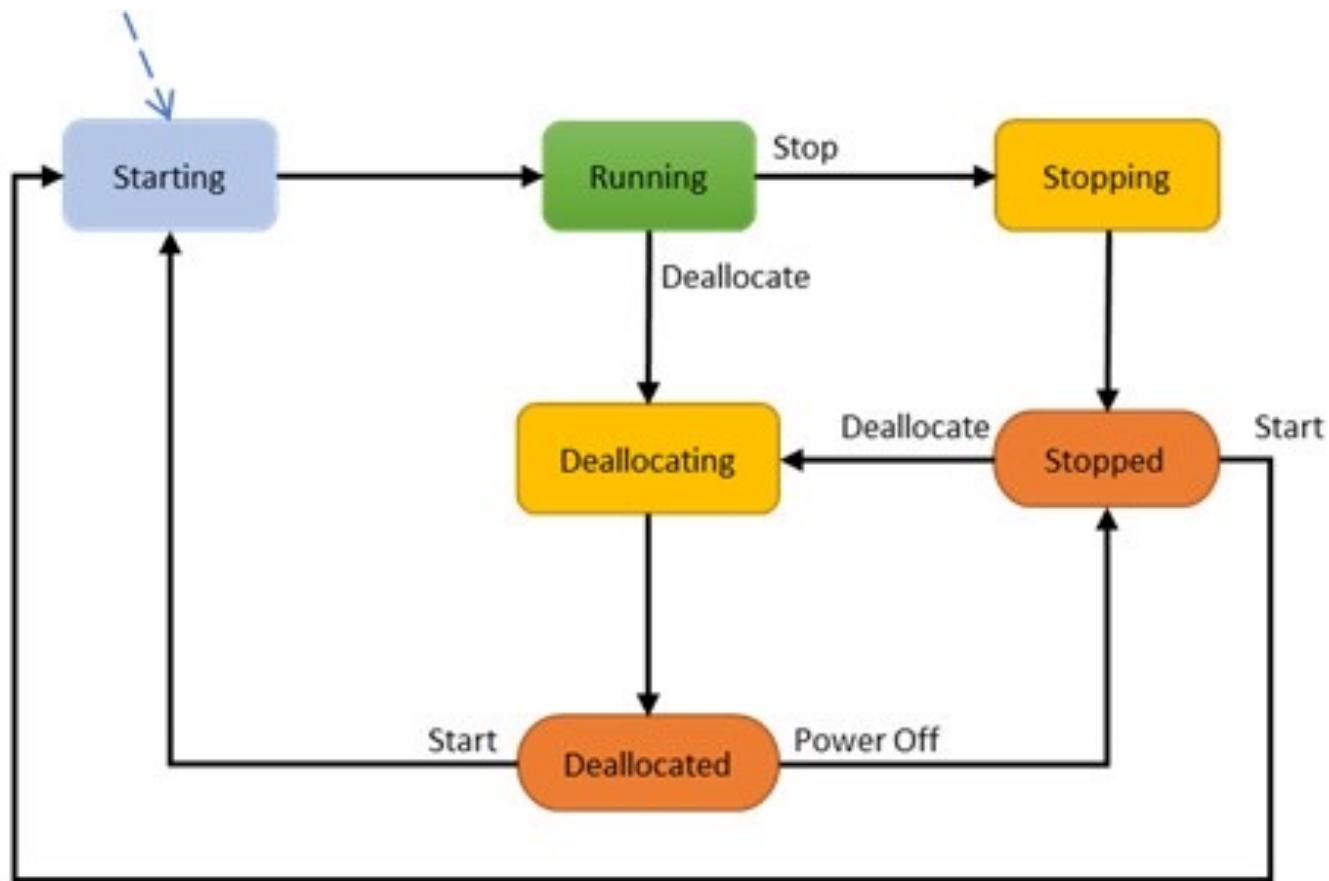
采用检查点：



□ Young 对上式子进一步分析，得出使 T_l 最小的 T_c 值约为：

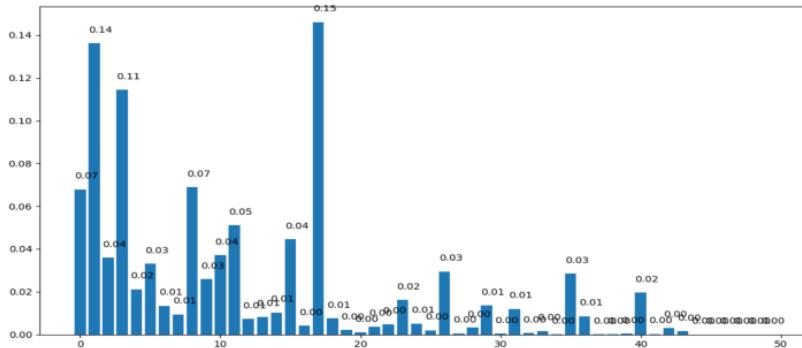
$$T_c = \sqrt{2T_s T_f} = \sqrt{2T_s / \lambda}$$

虚拟机的“一生”经历多个不同的状态

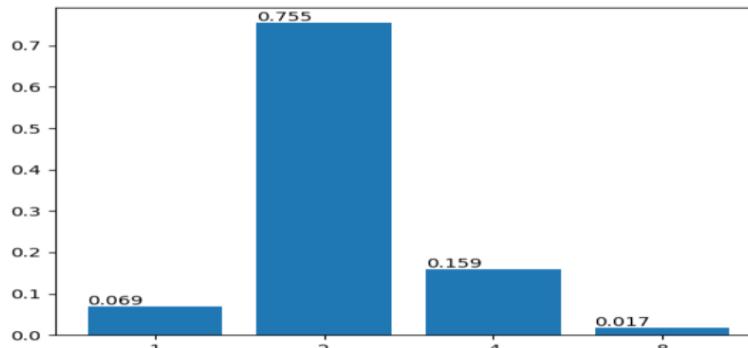


Azure Virtual Machines Power States

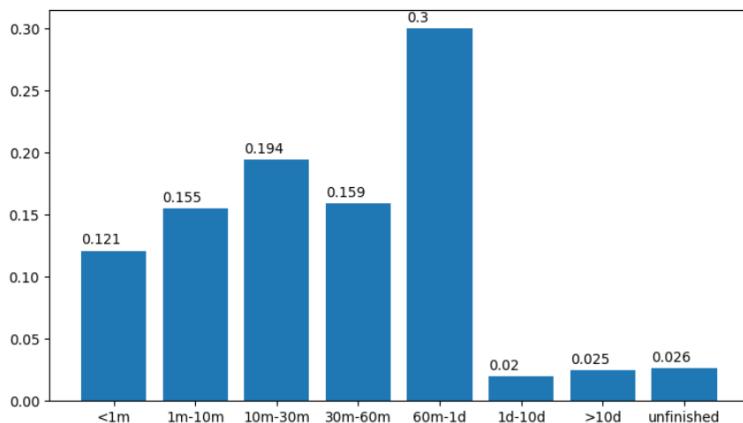
华为某实际虚拟机运行数据集分析



50余种VM产品的运行分布呈现不均匀性



某些资源类型的虚拟机需求更大



虚拟机的运行时长分布

1. 超过一半的虚拟机时长小于一小时
 2. 绝大多数虚拟机时长小于一天
 3. 有约10%的虚拟机时长在一分钟之内
 4. 对于长生命周期虚拟机，往往大于10天

VM 生命周期

云中存在大量围绕VM的管理负载 (management workload)
--- 一般涉及到维护和监控物理主机和虚拟主机的操作

操作	日操作数均值	日操作数峰值
虚拟机重新配置	2.3	699
虚拟机开机	90	1576
虚拟机关机	35	1535
虚拟机重置	4.6	176
虚拟机补丁安装	5.3	250
虚拟机生成快照	4.8	56
虚拟机快照还原	7	101
虚拟机克隆	6	44
虚拟机自动热迁移	51	3156

VM 生命周期

不受控的虚拟机繁殖被称为虚拟机蔓延 (VM Sprawl)

--- 回收虚拟化的计算资源或清理非正常虚拟机变得越来越困难

虚胖
虚拟机

CPU/内存/外存被过度配置，而
实际部署后并未使用

幽灵
虚拟机

一些非活跃VM，难删除、回收，
不得不任其消耗资源

僵尸
虚拟机

虚拟机虽停机，但相关镜像文件/
副本依然保留在硬盘上

VM 生命周期

解决虚拟机蔓延有哪些途径?
--- 对 VM 生命周期要**增强管控**



降低 (Reduce)

重用 (Reuse)

回收 (Recycle)

虚拟机调度

服务器整合

Andreas Schindler – IT Strategy and Architecture Services Leader CEEMEA

May 2009



Server Consolidation Services from IBM: Optimizing your IT infrastructure for greater savings, flexibility and resiliency

OPINION

Server Consolidation: Steps to IT and Business Rationalization



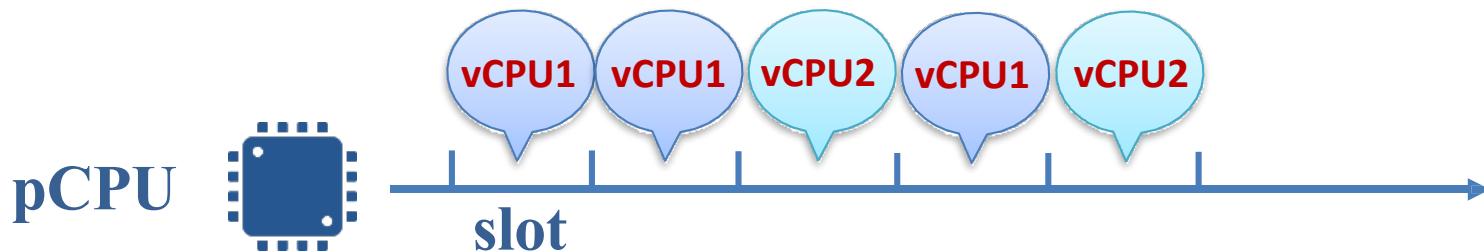
By Christopher Burry, Avanade

Computerworld | JUN 18, 2003 12:00 AM PST

虚拟机调度基本概念

虚拟处理器调度 (Virtual CPU Scheduling)

两个层面的调度问题 { 客户操作系统负责将进程调度于 vCPU 上
Hypervisor 负责将 vCPU 调度于 pCPU 上



物理 CPU 在固定长度时间单位下分配给虚拟 CPU。虚拟 CPU 有权重。若某 vCPU 权重为 0.6，意味着每 10 个 slots 中只有 6 个是能用于执行该程序线程的。

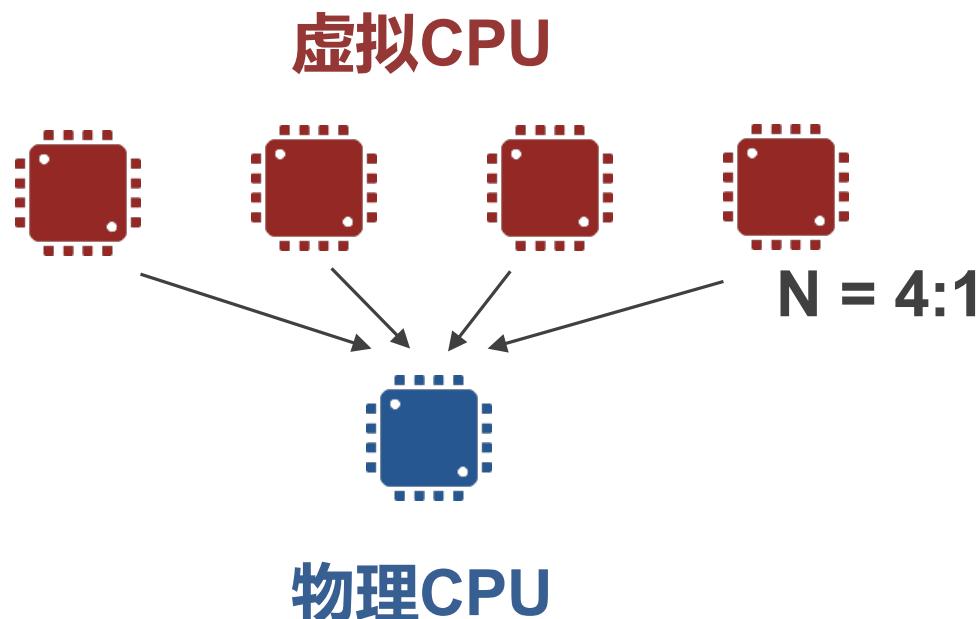
问题：在一台服务器上，虚拟机如何调度？

我们关注的调度对象是虚拟处理器(vCPU)。
vCPU替代了物理处理器，更多 vCPU意味着更强的线程并行处理能力

虚拟机调度基本概念

虚拟机整合与超分比

超分比英文即 Oversubscription Ratio, 或者说 vCPU-to-pCPU ratio。超分比为 N 时, 每台客户机实际的计算性能缩减到了 $1/N$ 的物理机最大性能



虚拟机调度基本概念

公平调度 (Fair-Share, FS)

将计算资源公平地分配给每一个虚拟机用户

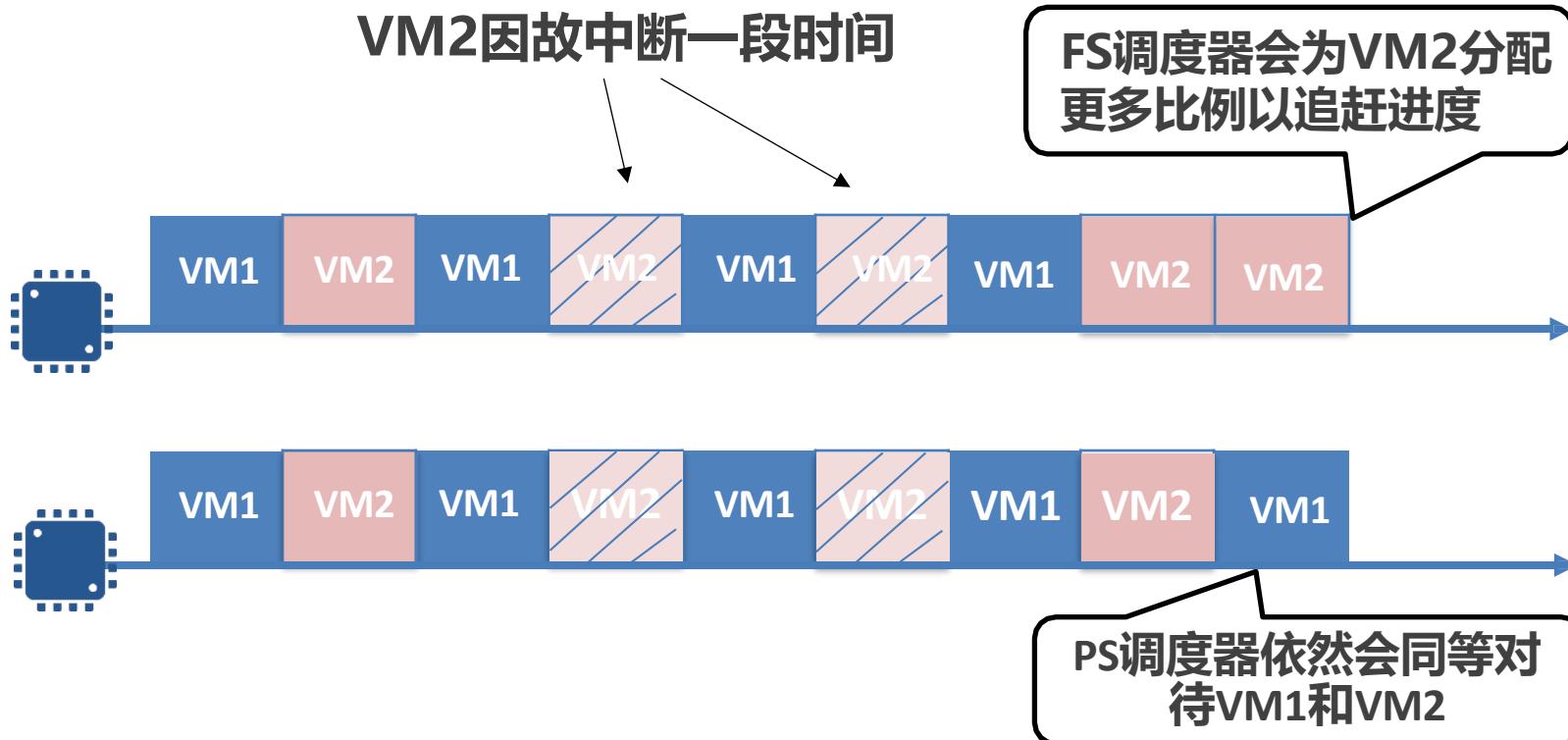
比例分享调度 (Proportional Share, PS)

系统根据虚拟机所拥有的分享指标（权重），
按比例地分配 CPU

虚拟机调度基本概念

FS 调度器作用于一个较粗时间粒度下的比例分配共享

PS 调度器目标是根据权重提供一个即时的等比例分享

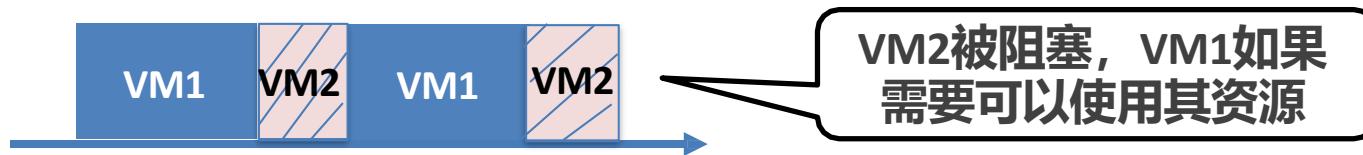


虚拟机调度基本概念

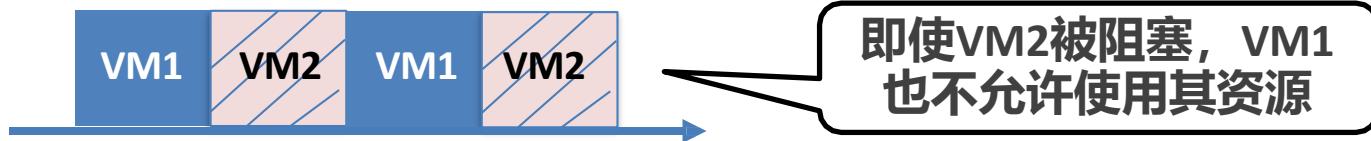
Work-Conserving vs. Non-work-conserving

工作保留 vs. 非工作保留

WC模式下，虚拟机获得的资源是基本保证



Non-WC模式下，虚拟机获得的是资源限额



虚拟机调度基本概念

Preemptive vs. Non-Preemptive

抢占式: 若新用户具有高于目前用户的“优先级”，则系统会抢夺走当前用户的资源，转而执行新用户需求

非抢占式: 允许当前执行的客户完成其正在使用的CPU配额，除非当前客户机主动放弃了CPU资源

Xen Credit 调度器

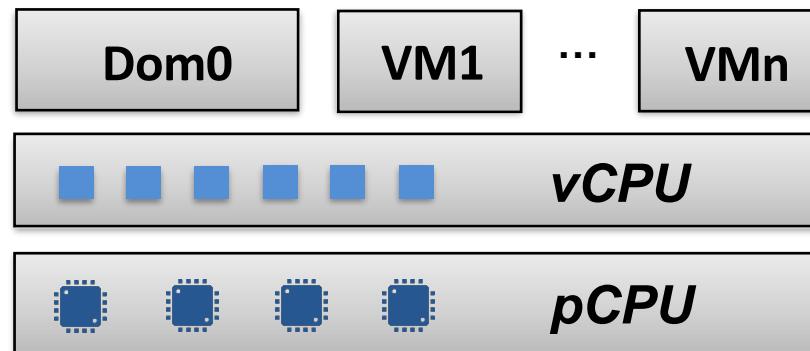
Xen Credit is a virtual CPU scheduler

- 1 以共享CPU资源为目标
- 2 以最大化硬件资源利用率为目
- 3 以满足延时敏感型任务为目

Guest OS run on vCPUs

Xen Hypervisor
(emulated HW interface)

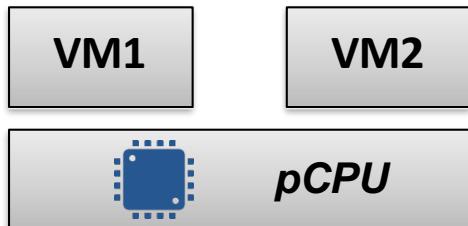
Physical Hardware



Every vCPU consumes credits when running: the hypervisor implements a global accounting thread to compute credits

Xen Credit 调度器

Xen Credit has two key parameters



VM1获得2倍于VM1的
CPU时间

Name	Weight	Cap
VM1	512	0
VM2	256	0

Legal weights range from 1 to 65535
and the default value is 256

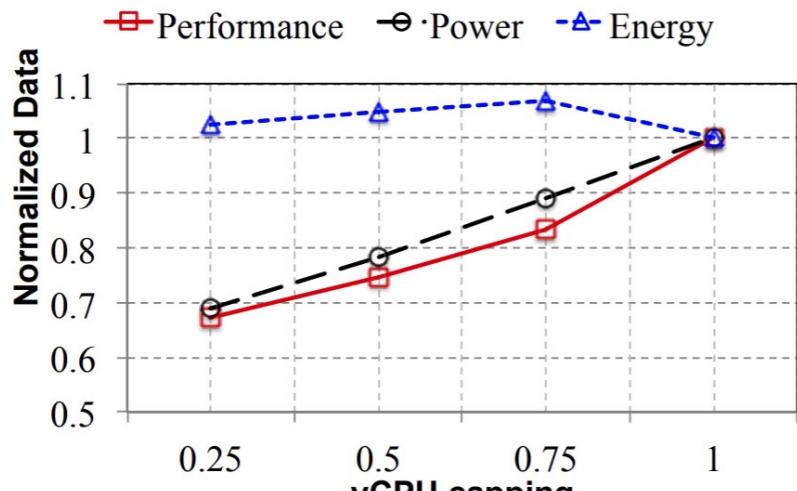
Weight: to grant a specific VM more CPU time than others.

Cap: specifies the maximum cap% of CPU a VM can use.

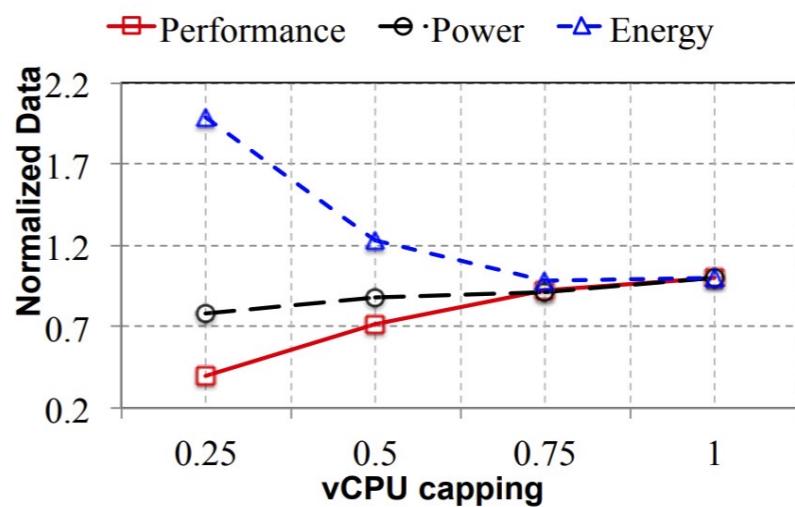
1. Credit 是 Proportional Share, 意味着它通过权重实现比例分享
2. Credit 可以是 WC 或 NWC, 取决于 Cap 值 (Cap 为 0, WC mode)

Xen Credit 调度器

Cap对不同类型应用的综合影响



计算密集型



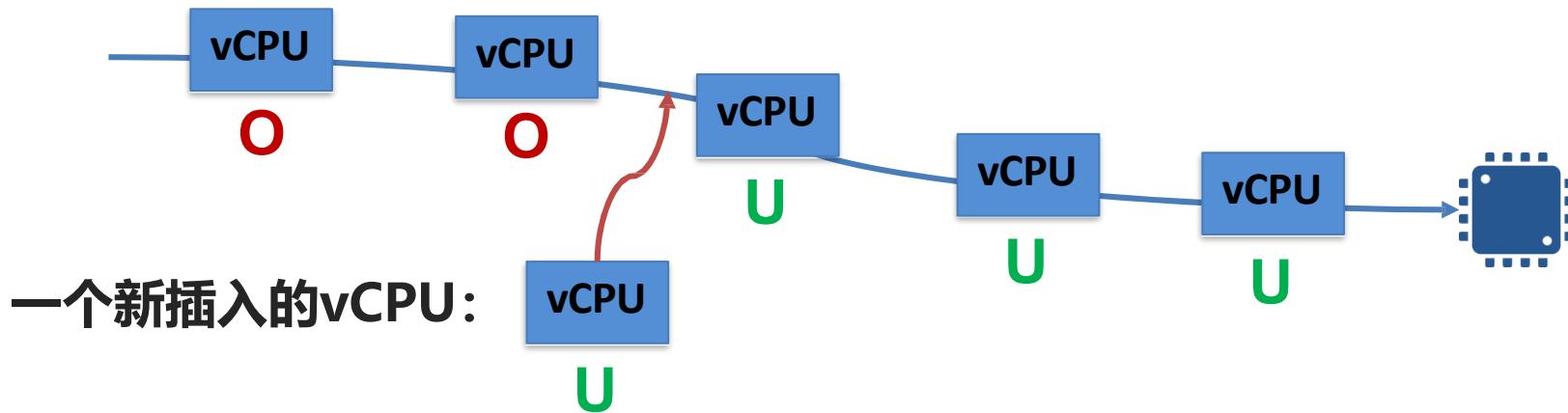
数据密集型

Xen Credit 调度器

Xen Credit maintains a queue of vCPU

该算法下，每个物理处理器核都有一个本地的vCPU调度队列，队列中vCPU依照优先级排序

- Two vCPU priorities: **OVER** and **UNDER** fair share.
- **OVER**: 该vCPU已耗尽该周期其公平应有的CPU资源
- **UNDER**: 该vCPU在当前周期还余留可用的CPU资源



Xen Credit 调度器

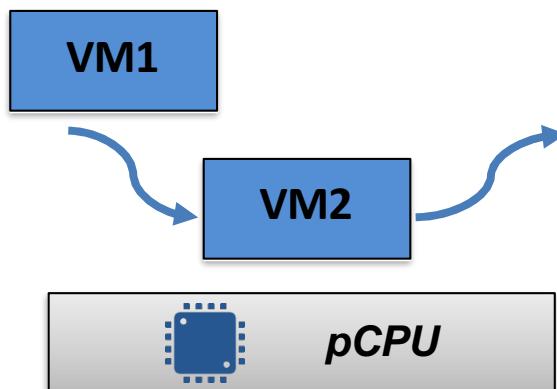
Xen Credit also defines time-slice (时间片)

也被称为调度量子位，指的是一个vCPU可以在pCPU上运行的最短时间，即使在其它vCPU存在抢占的情况下。

当一个vCPU完成其时间片后，调度器会从调度队列中选择下一个

Timeslice防止过于频繁的系统切换，能避免CPU/Cache Trashing

为优化性能，tslice的值可以重配置，常见的有10ms, 5ms, 1ms

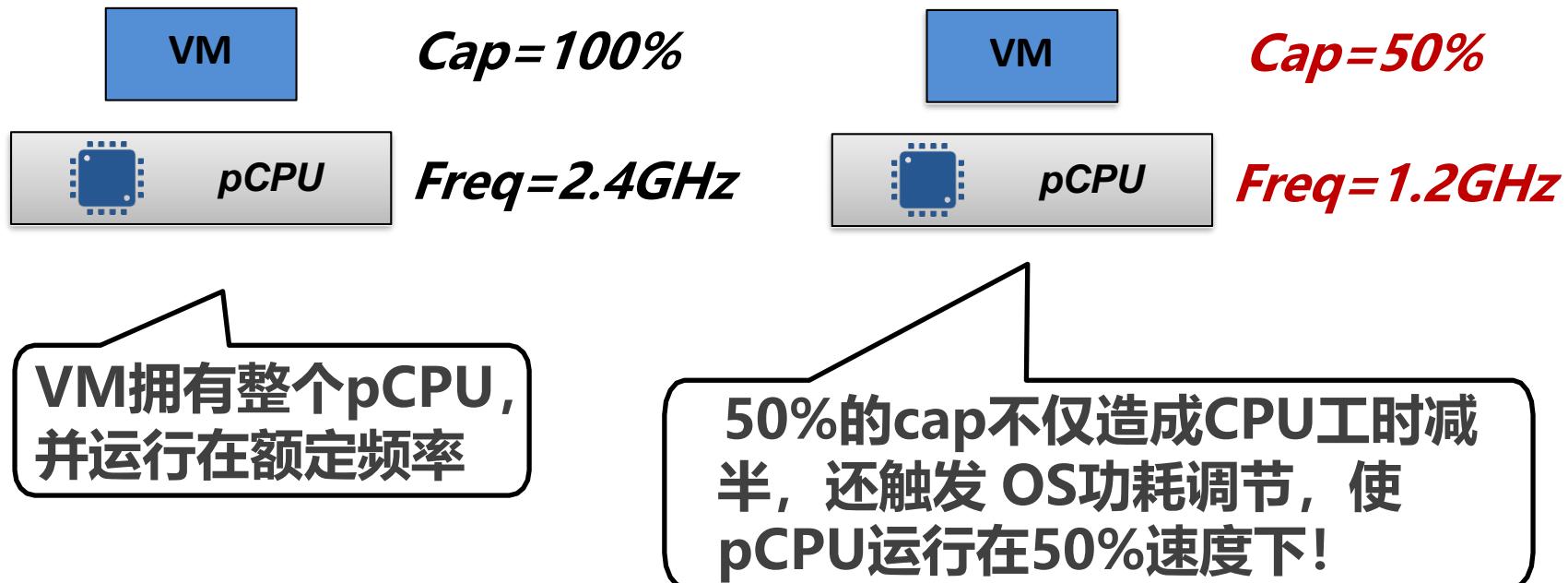


默认tslice是30ms，即VM2一旦分配至pCPU，将至少获得30ms运行时长

Xen Credit 调度器

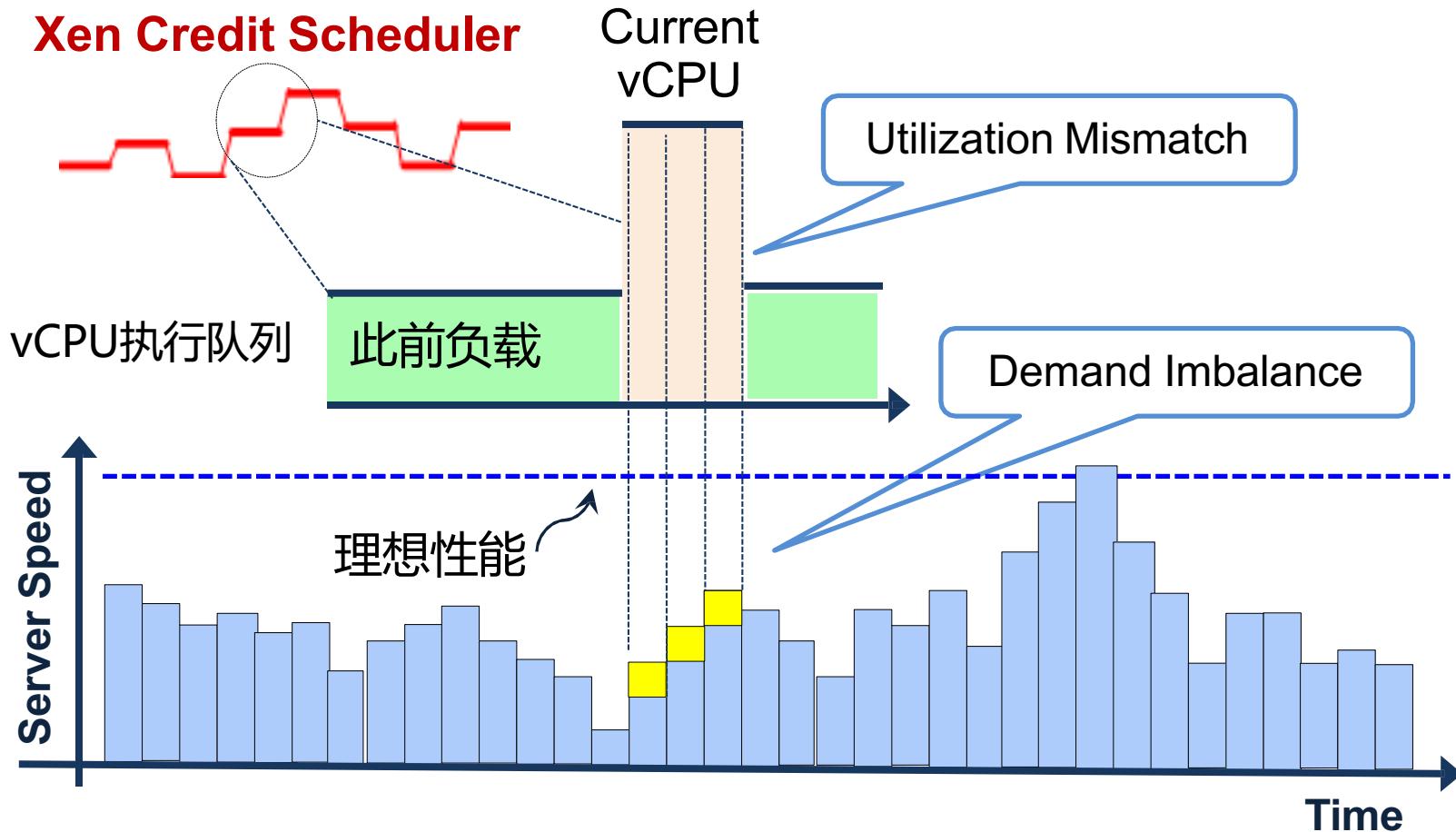
Xen Credit + Linux CPU Scaling Governor

操作系统定义了一系列功耗管理 (power management) 机制，能够通过底层ACPI接口，实现处理器频率的自适应调整



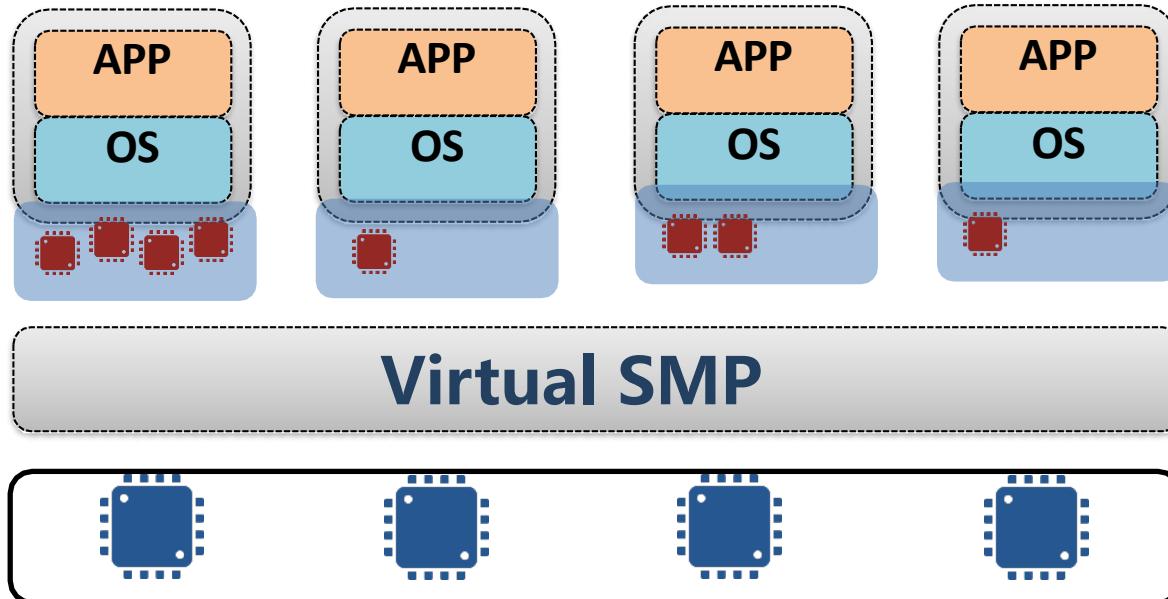
Xen Credit 调度器

调度复杂性例子：如果 vCPU 默认 30ms 调度周期的话，在特殊情况下可能无法应对负载的波动变化，从而始终停留在次优状态



vSMP 调度

虚拟对称多处理器调度 (Virtual Symmetric Multi-Processing)



vSMP允许一个虚拟机使用多个物理处理器

例如，VMware ESXi 允许一个VM最多配置64个vCPU

vSMP 调度

虚拟对称多处理器调度对大数据并行任务尤其有用

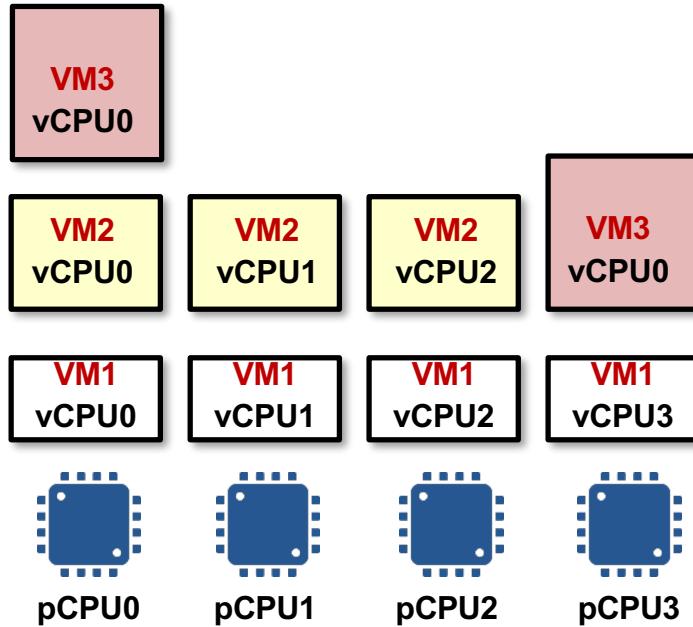
1. **增强性能**: Ability to run resource-intensive applications in virtual machines

2. **负载均衡**: Automatic shifting of tasks among available processors to balance workload

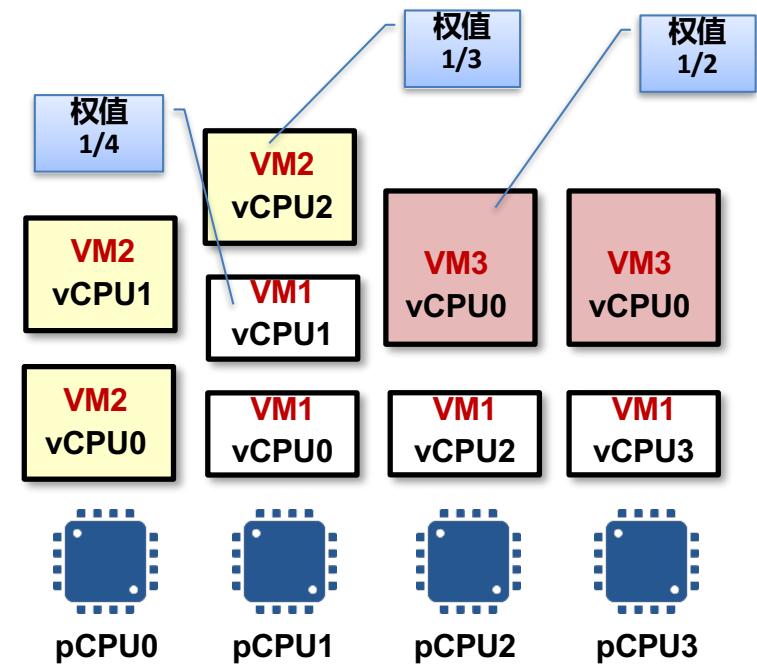
3. **优化利用**: Scale computing environment and increase utilization of existing resources

vSMP 调度

虚拟对称多处理器映射方法举例



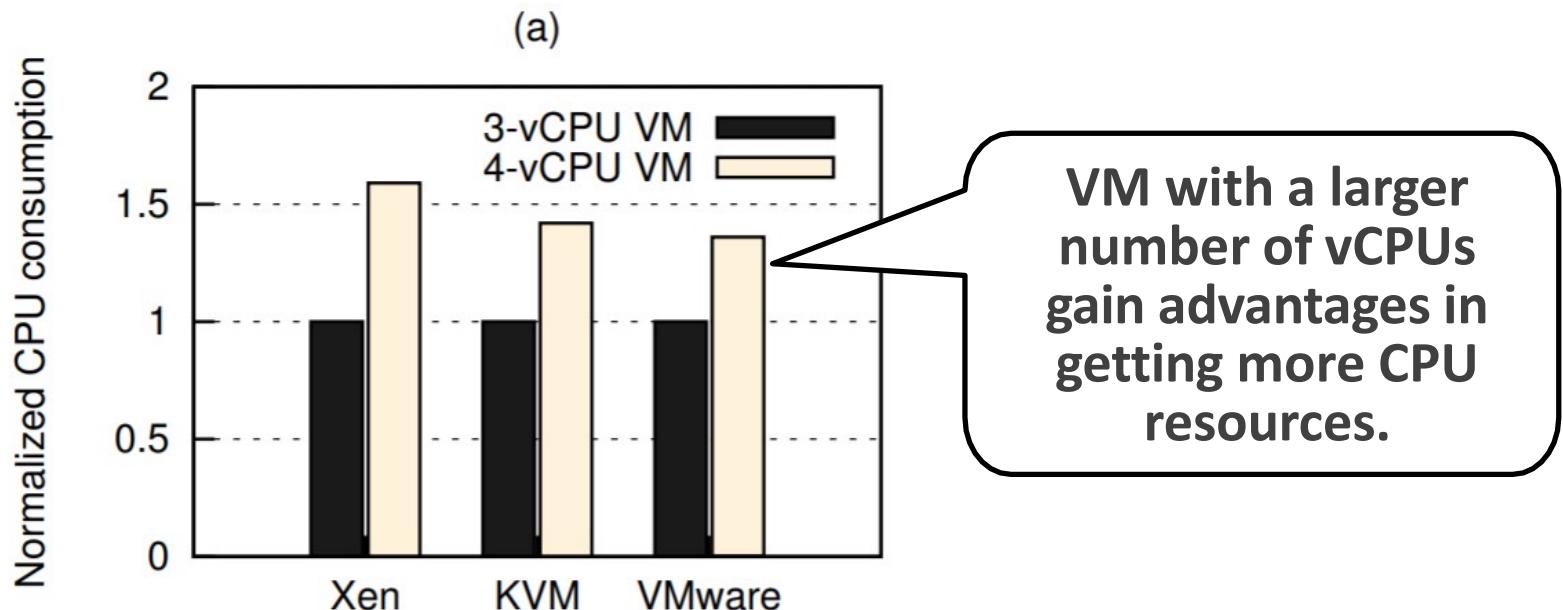
基于队列长度的负载均衡
(不考虑每个VM占有几个vCPU)



基于队列权值的负载均衡
(每个虚拟机给定同等权值)

vSMP 调度

vSMP 调度在虚拟机层面可能造成不公平资源分配



两个具有相同权重的异构VM共享4个CPU的资源利用测试

虚拟机迁移

虚拟机热迁移

用户全程“零感知”？UCloud是这样做云数据中心整体热迁移的

本文作者：AI研习社 2018-12-26 16:06

“ 导语：在UCloud成立的6年多时间里，产品线持续扩张，产品质量和性能也在不断提升。而在每一次的升级迭代背后，都代表着UCloud后台技术能力一次质的提升。

机房热迁移，用户无感知

UCloud在全球部署了29个可用区，韩国首尔就是其中之一。但是由于韩国机房供应商的原因，UCloud不得不面临需在3个月内完成对整座机房层面的数据迁移的严峻局面。为了在最大程度上保证用户业务正常运行，在系列调研分析之后，UCloud决定开始对机房进行热迁移工作。

UCloud应用云产品中心总监吴斌炜现场介绍到，UCloud机房主要有两类产品，一类是承载用户的产品，这部分主要是一些主机网络产品，用于承载用户业务；还有一类是控制产品的管理模块，这些模块控制产品的创建、删除、扩容操作。要将两个机房合成一个机房，就需要在一个机房部署这两类产品。因此，迁移的过程也是分两步进行的，首先是控制系统的迁移，然后再迁移上层的产品。

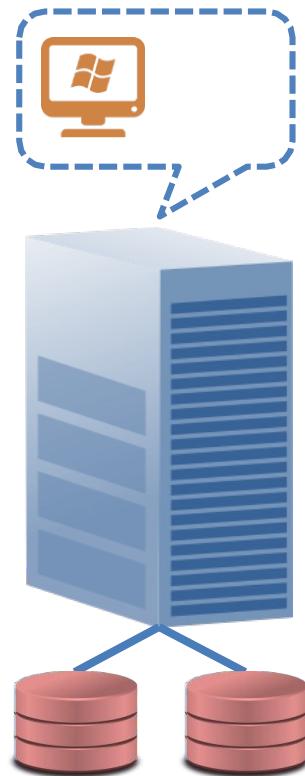
迁移需求

设施维护角度出发



迁移需求

硬件更换角度出发

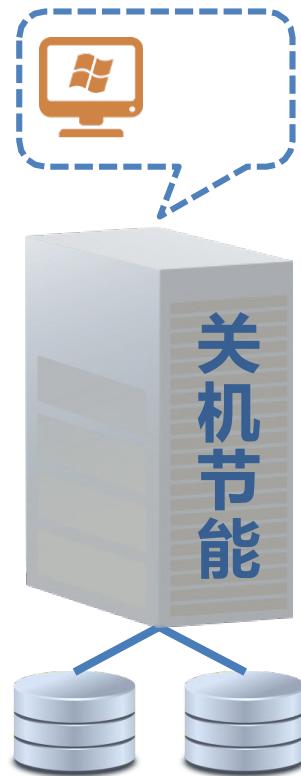


更换硬盘



迁移需求

资源利用角度出发

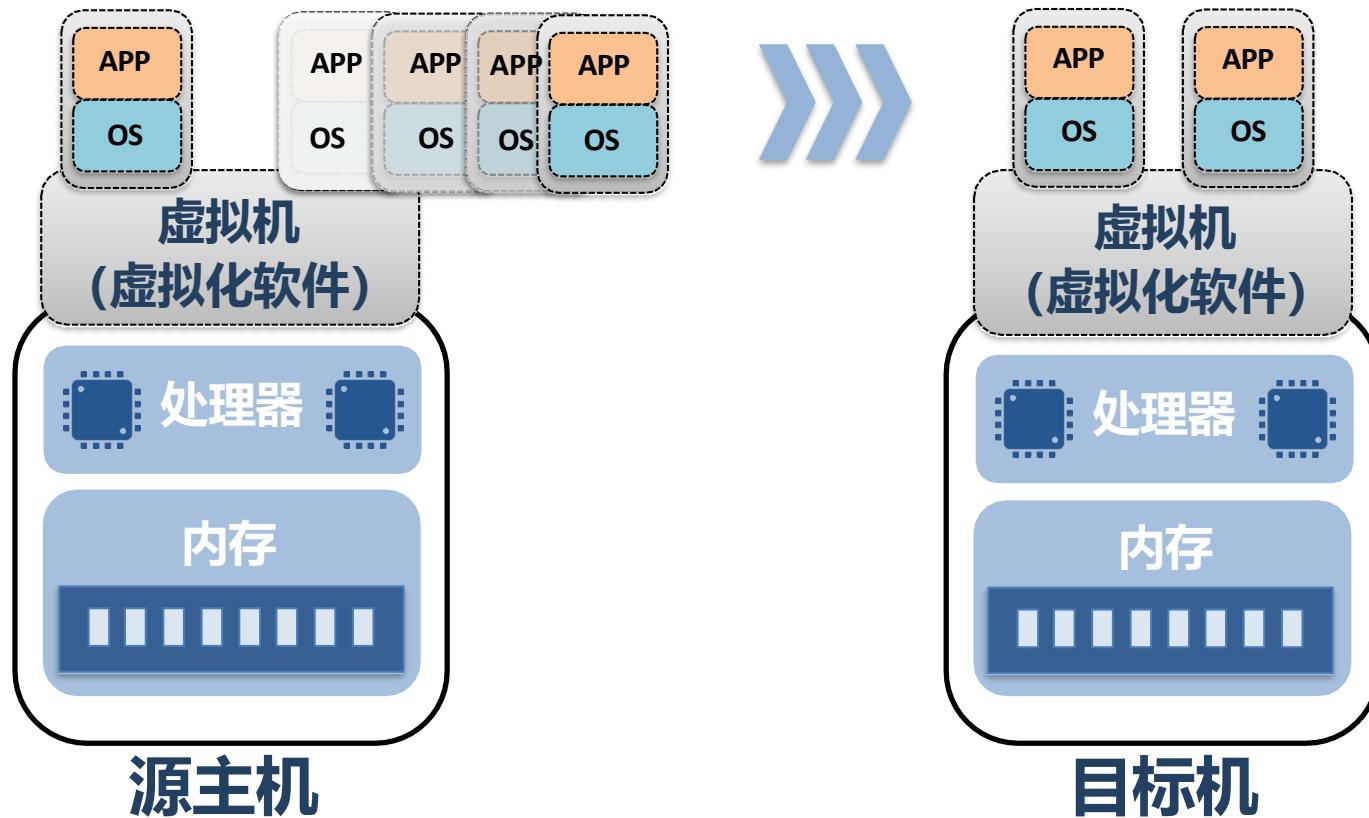


问题2：在一个主机集群中，虚拟机如何调度？

我们接下来关注的调度对象是虚拟机实例(instance)。在虚拟机迁移过程中，物理机也伴随着开关状态的改变。

迁移条件

负载迁移能力靠虚拟化软件支撑



迁移条件

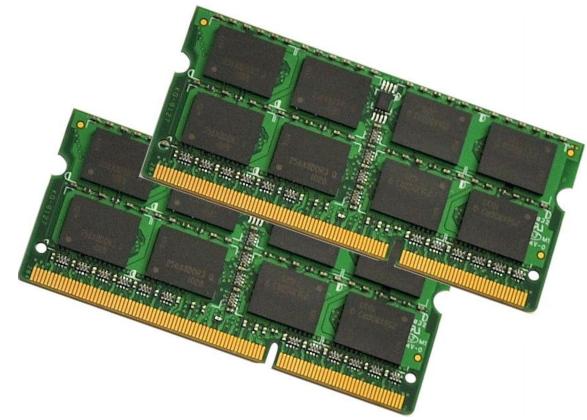
□ 三种需要迁移的资源



Storage



Network



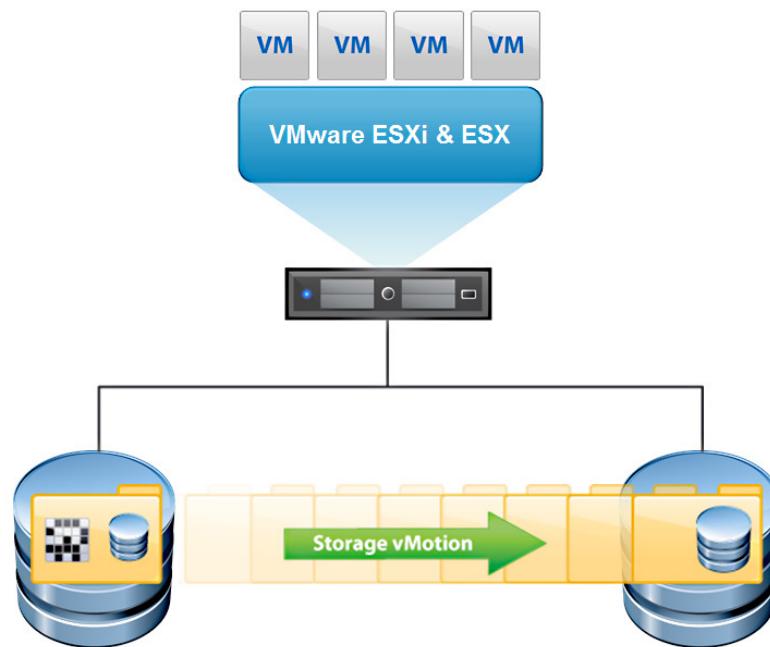
Memory

存储迁移

口迁移存储的最大障碍在于需要**占用大量时间和网络带宽**

口通常的解决办法是**共享数据和文件系统**，而非真正迁移

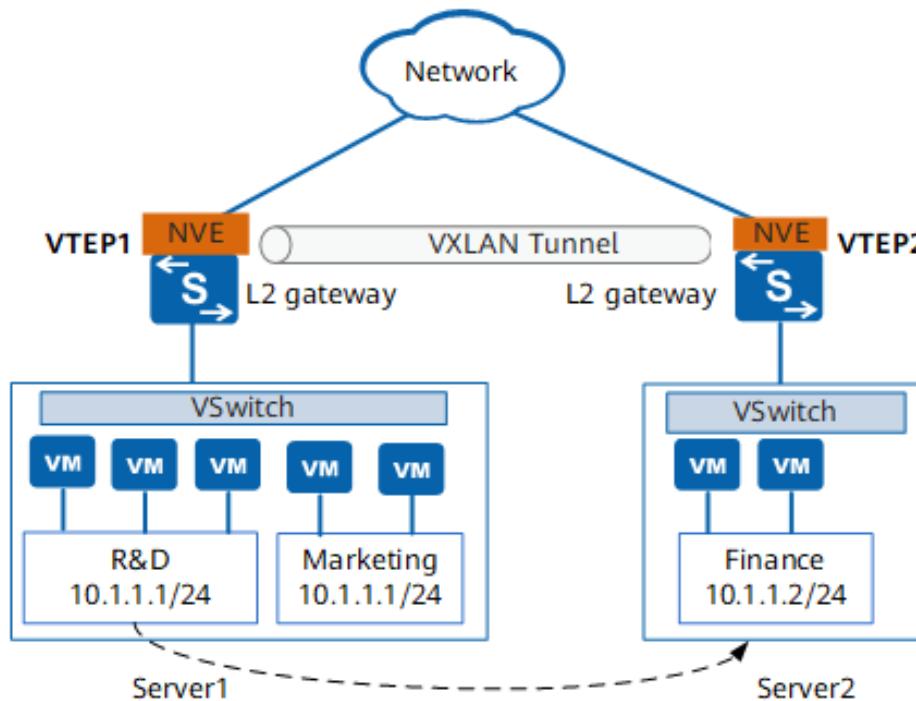
口目前大多数集群使用 NAS (Network Attached Storage, 网络连接存储) 作为存储设备共享数据



网络迁移

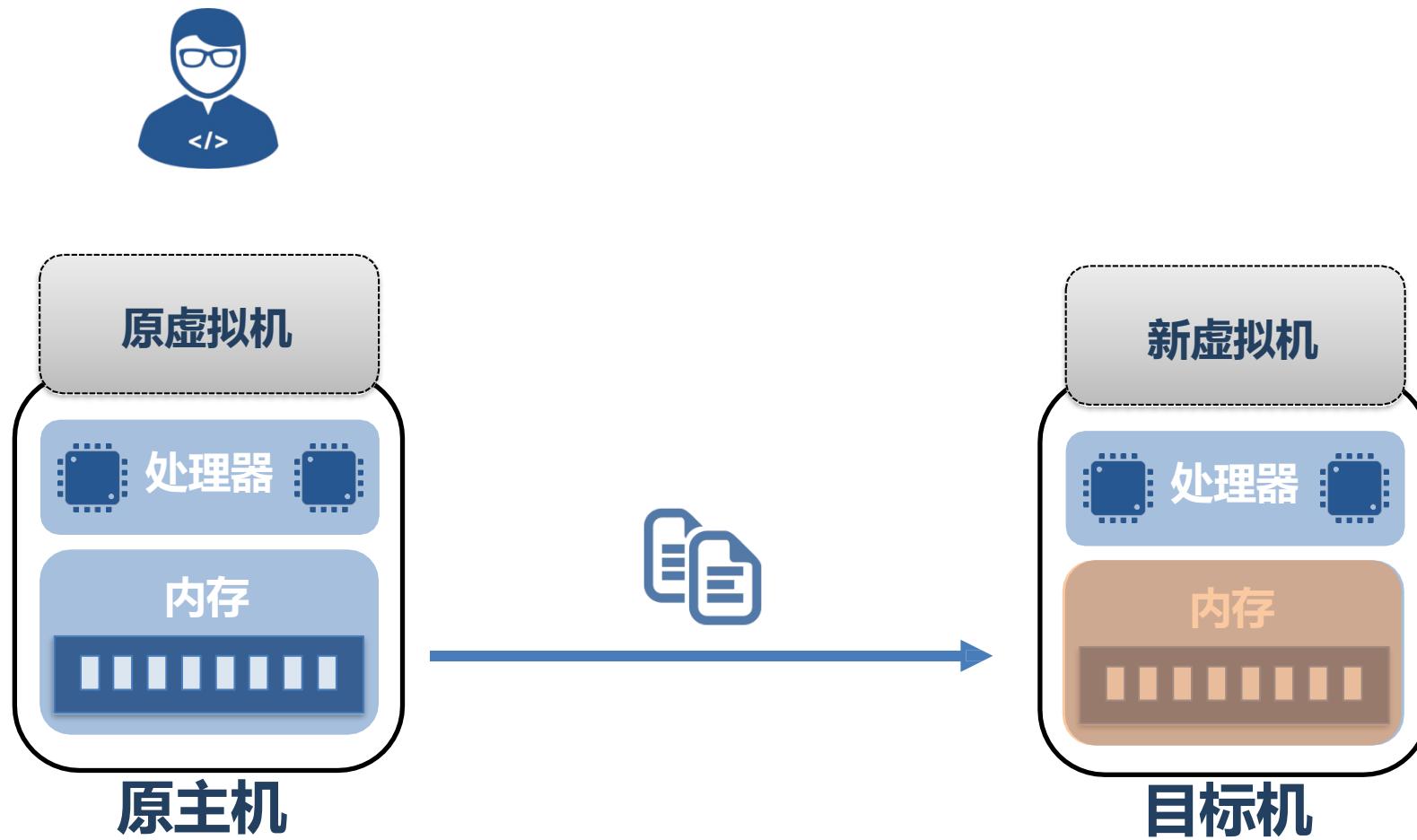
口虚拟机的封装方式意味着迁移时 VM 的所有网络设备，包括协议状态（如 TCP 连接状态）以及 IP 地址

- 方法一：局域网内，可通过 ARP 重定向包将 VM 的 IP 地址与目的机器的 MAC 地址绑定，后续数据包即可发送到目的机器上
- 方法二：通过虚拟化平台的**网络虚拟化技术**实现，如VxLAN



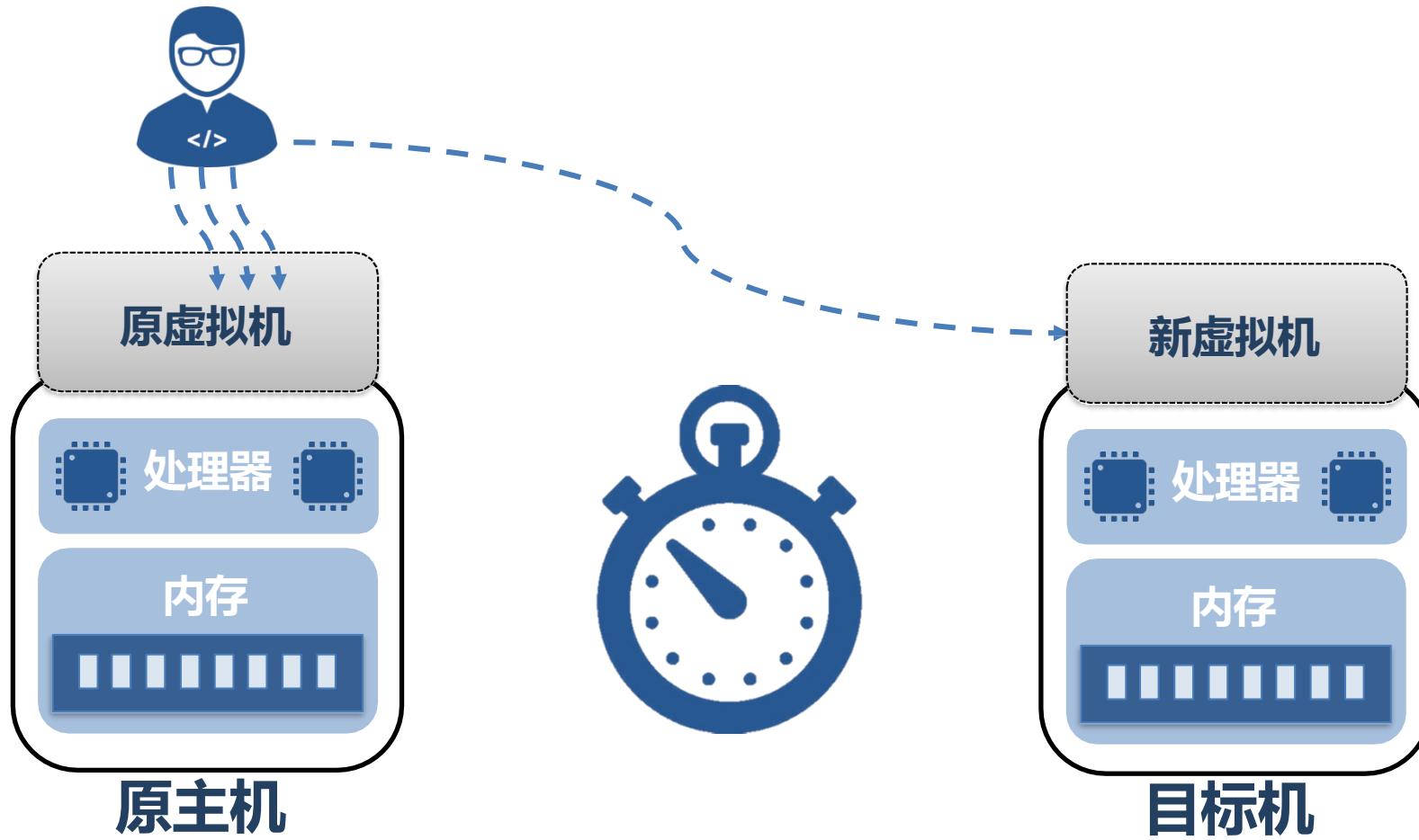
内存迁移

负载的关键在于重建“内存数据”



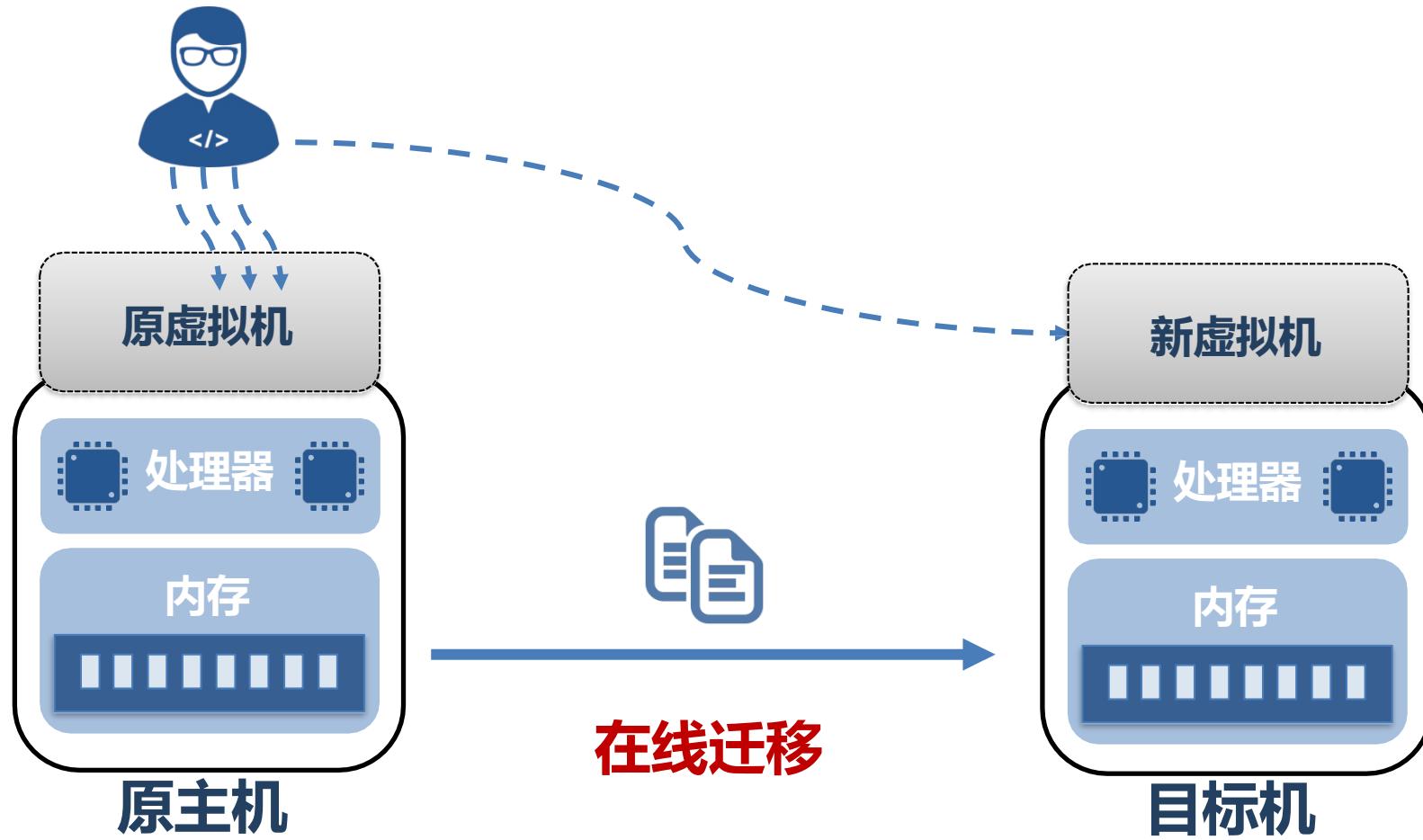
内存迁移

最简单暴力的迁移：停机迁移



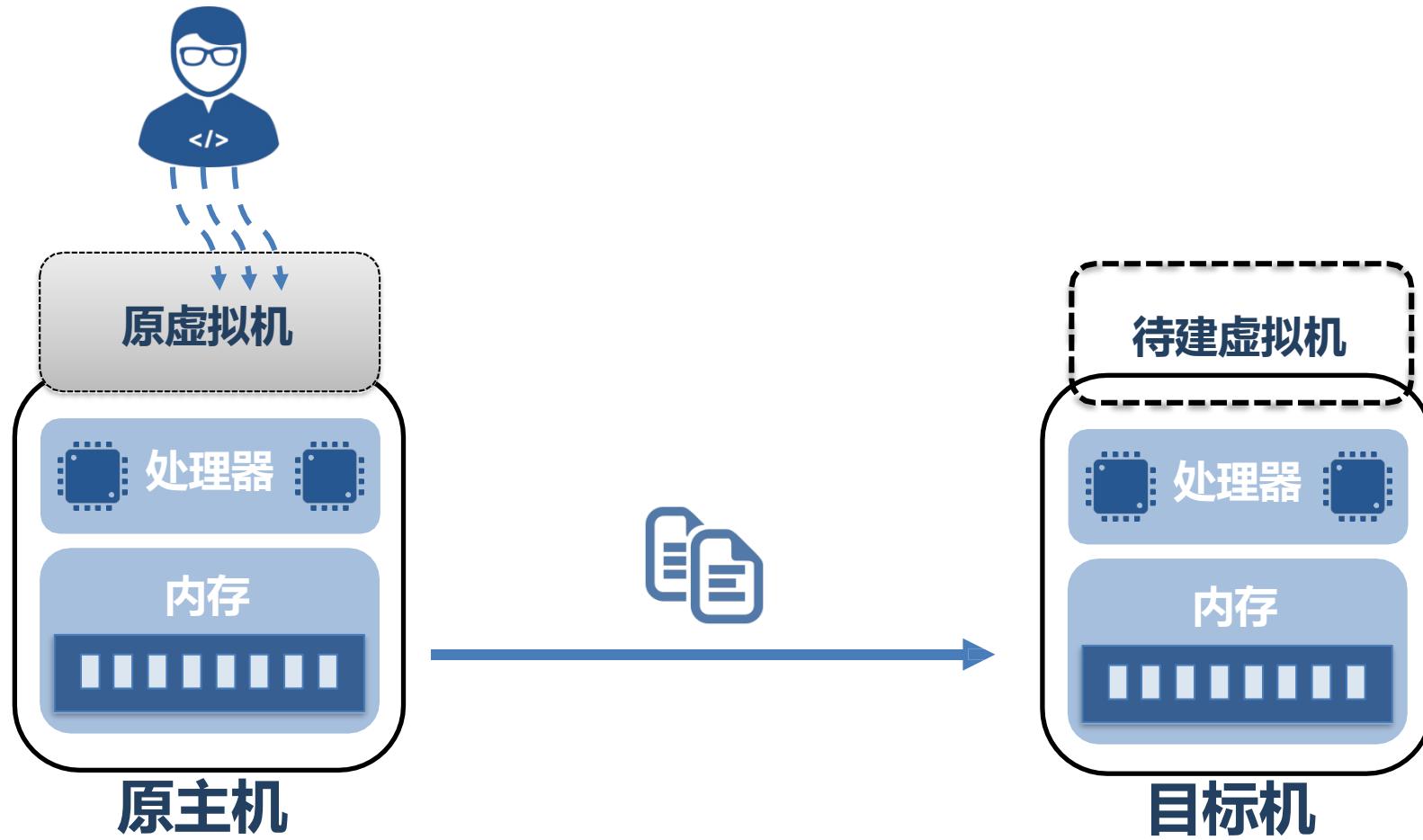
内存迁移

需要避免长时间停机



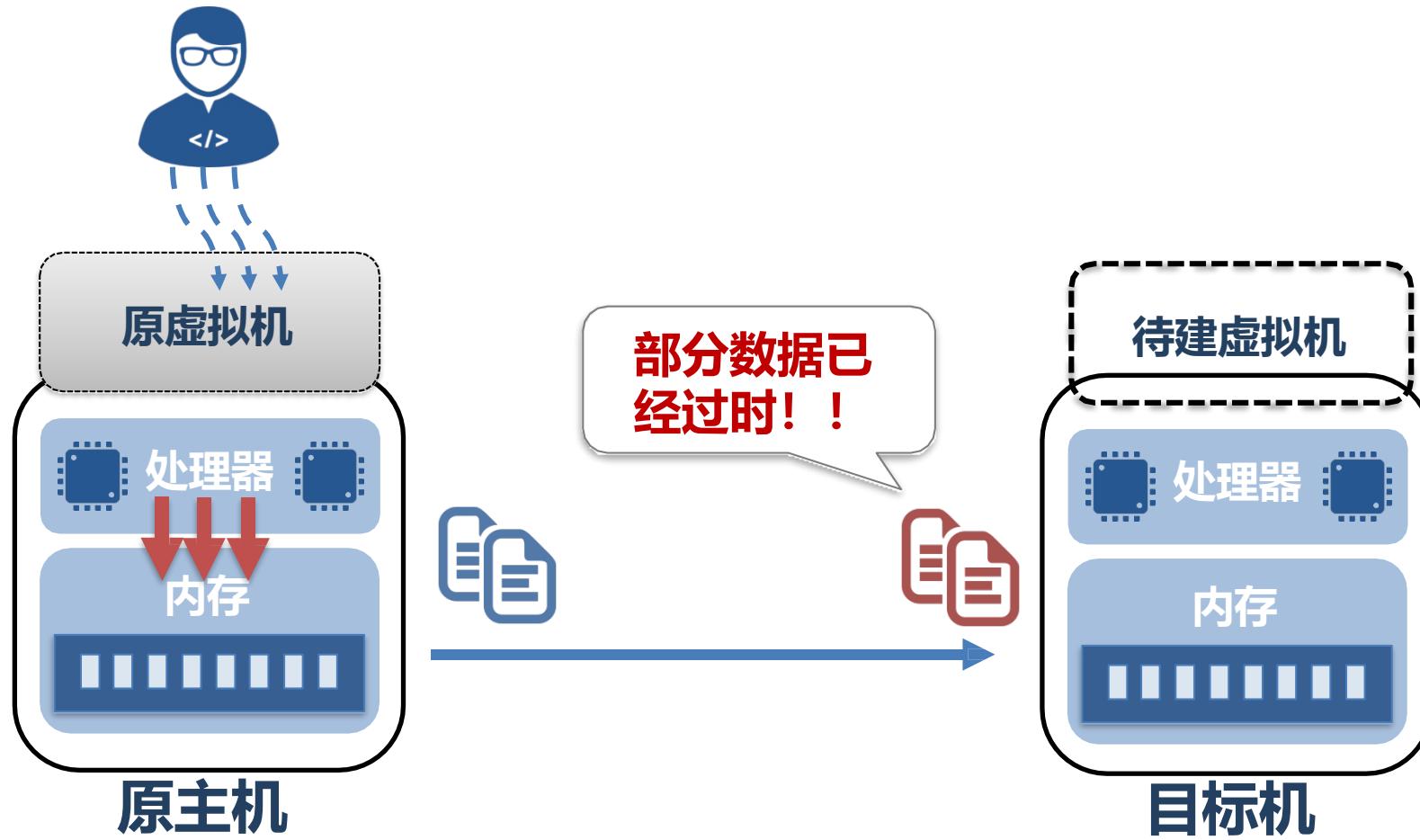
内存在线迁移

服务用户的同时重建数据



内存在线迁移

存在数据一致性问题！



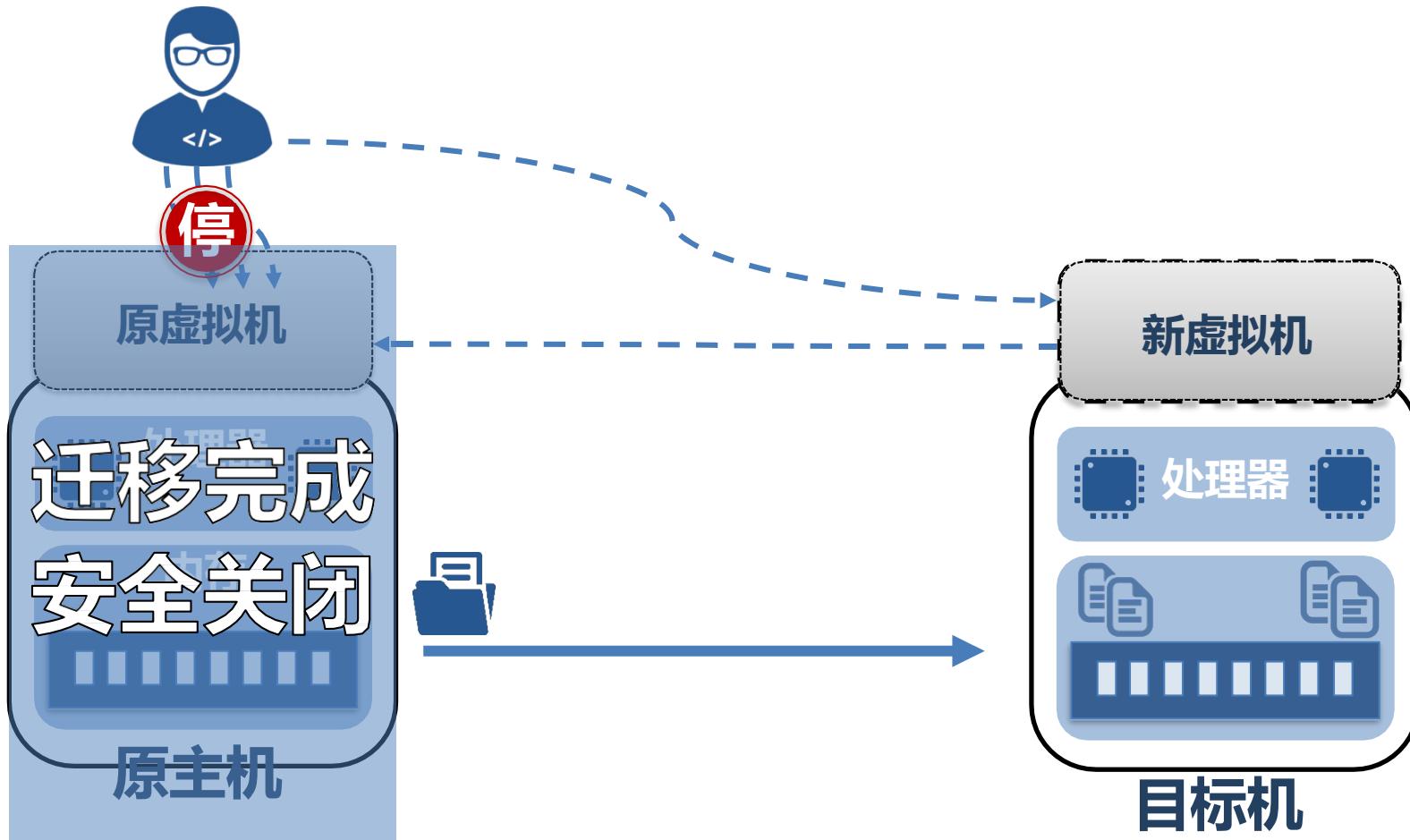
内存在线迁移

关键思路：对内存数据特征进行划分



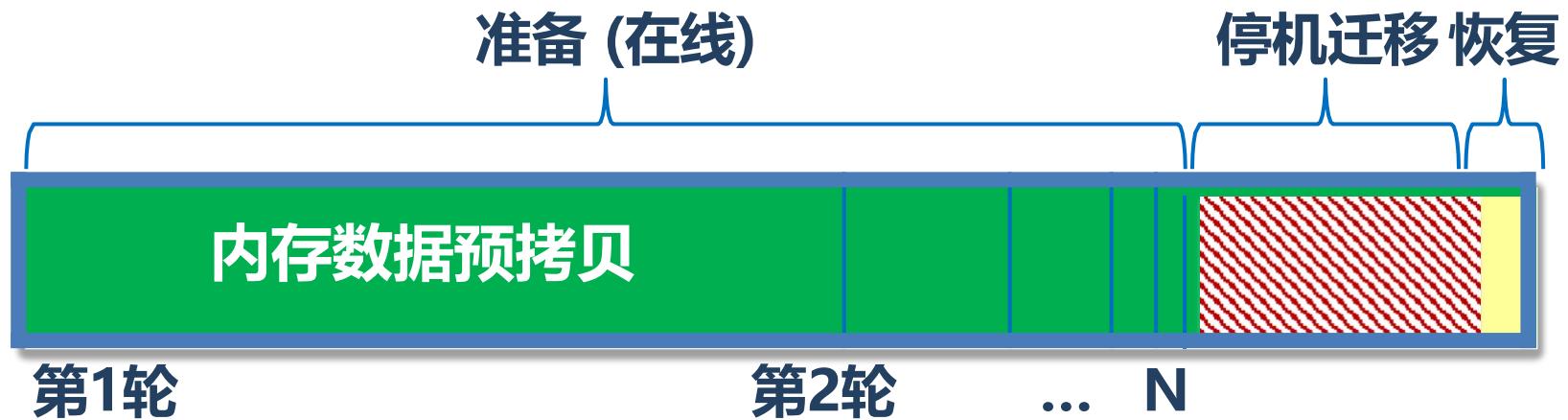
内存在线迁移

对最后少部分数据采用停机传输



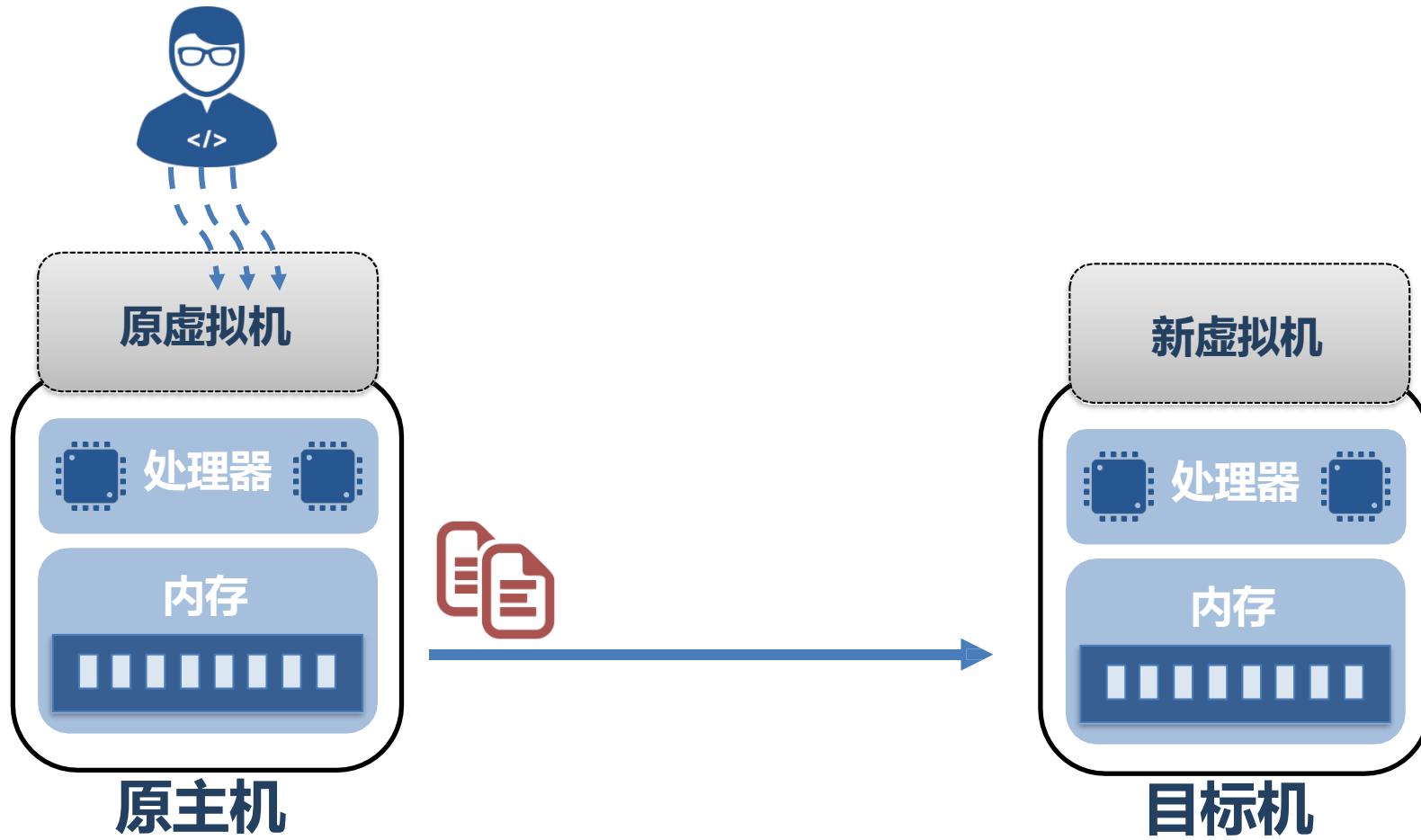
内存在线迁移

“预拷贝”（Pre-Copy）迁移法



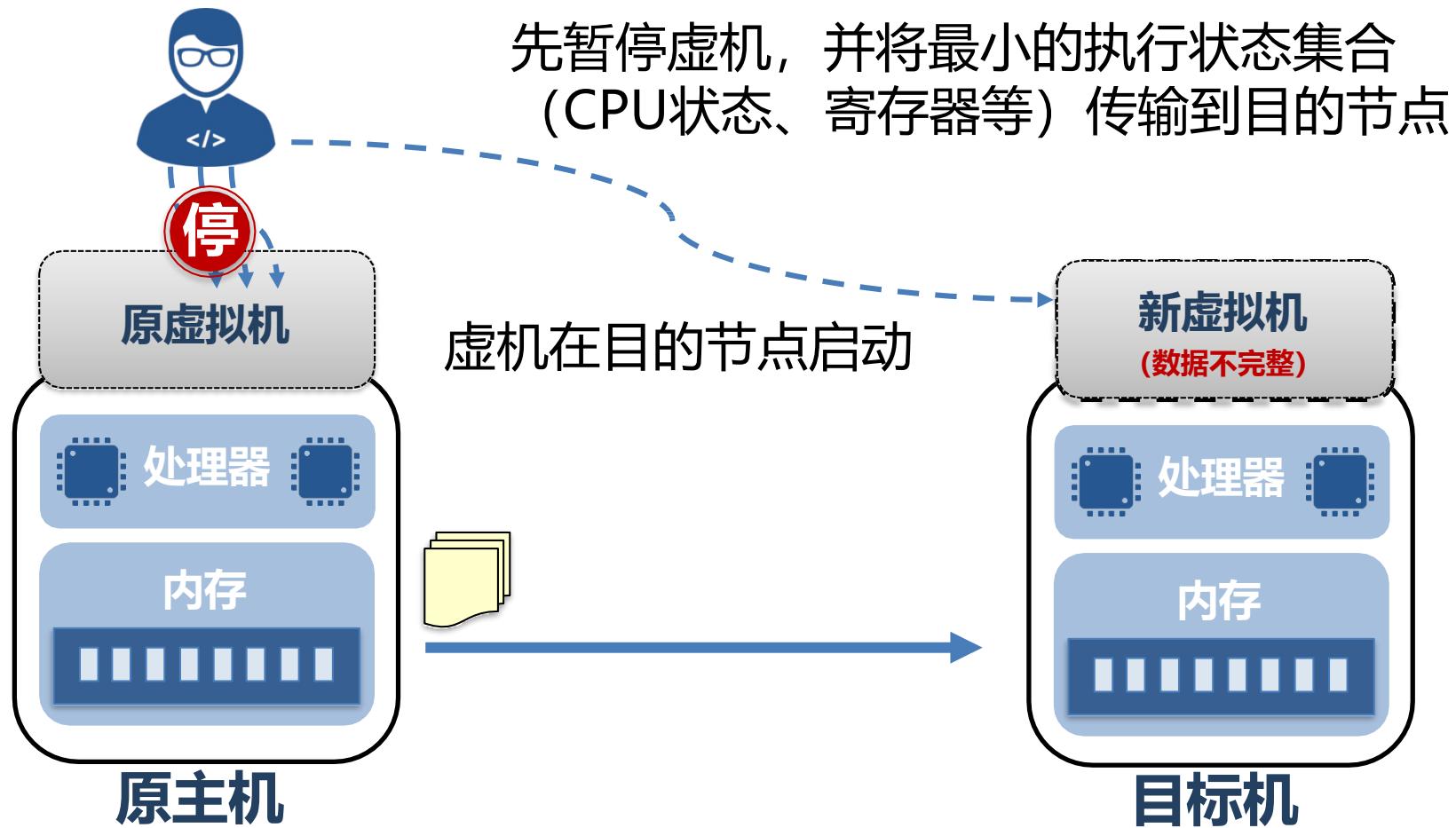
内存在线迁移

预拷贝迁移难免造成部分页面的反复传输



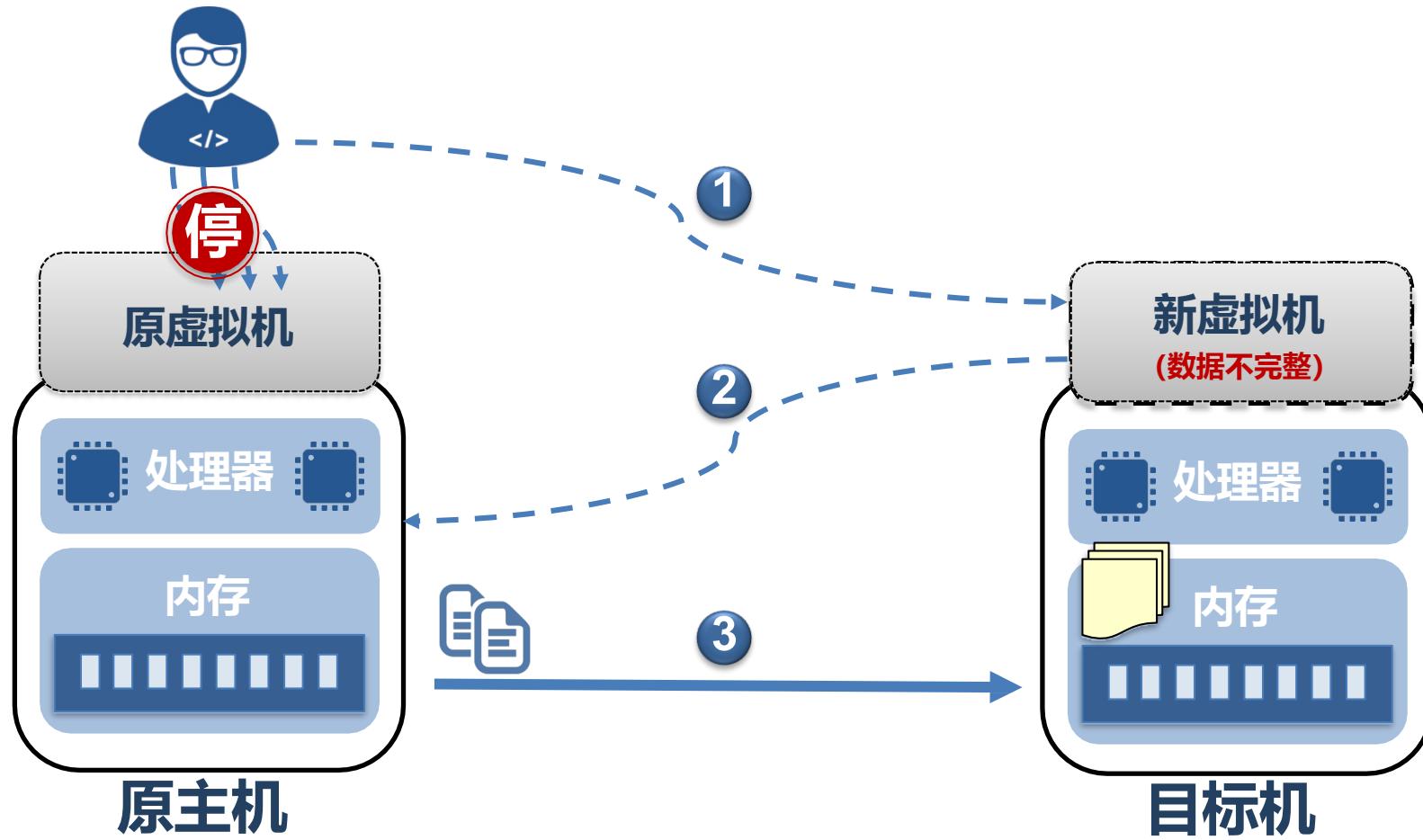
内存在线迁移

后拷贝(Post-Copy) 是对预拷贝的补充和改进

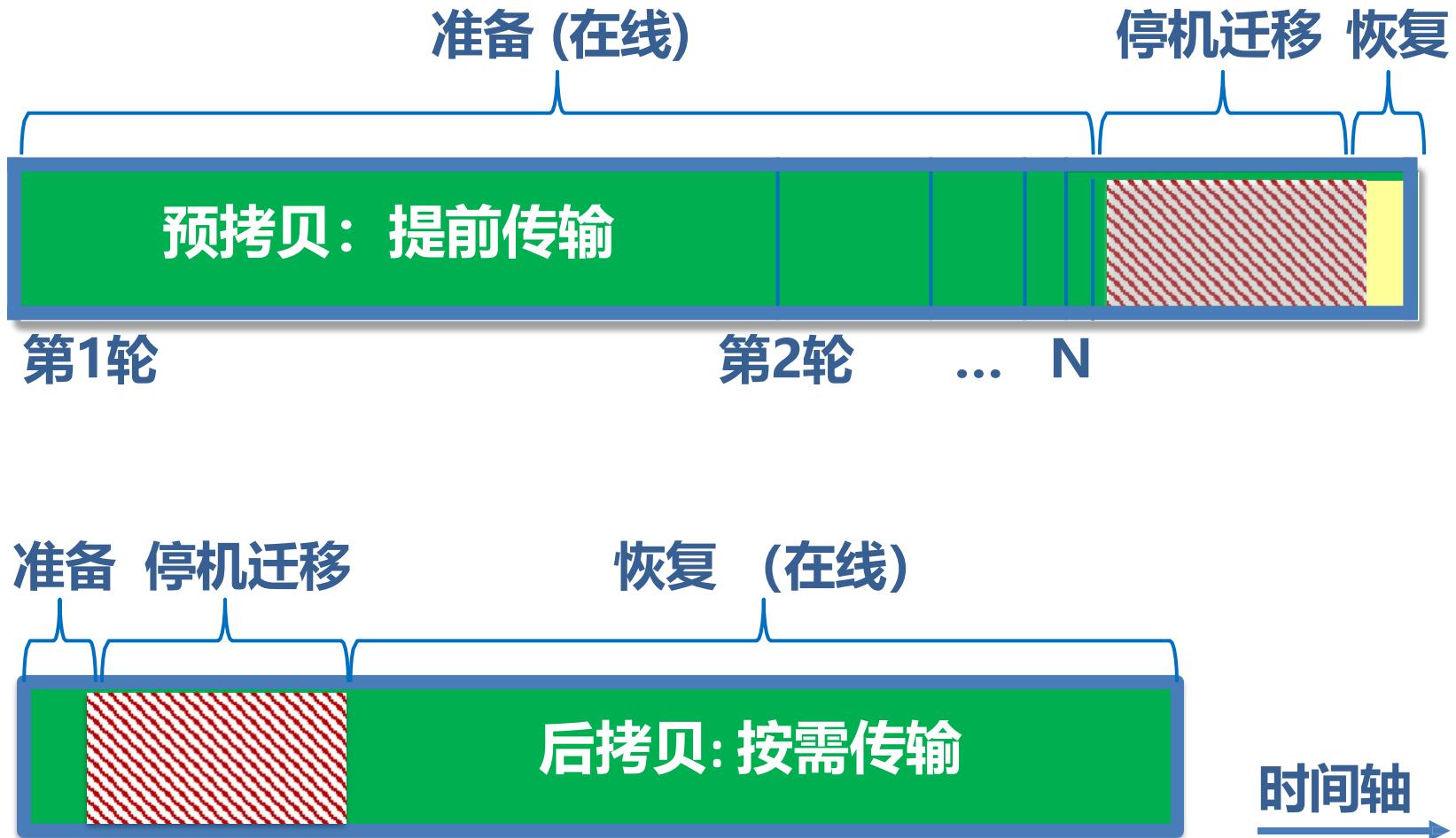


内存在线迁移

后拷贝技术对部分内存页实施**按需抓取**

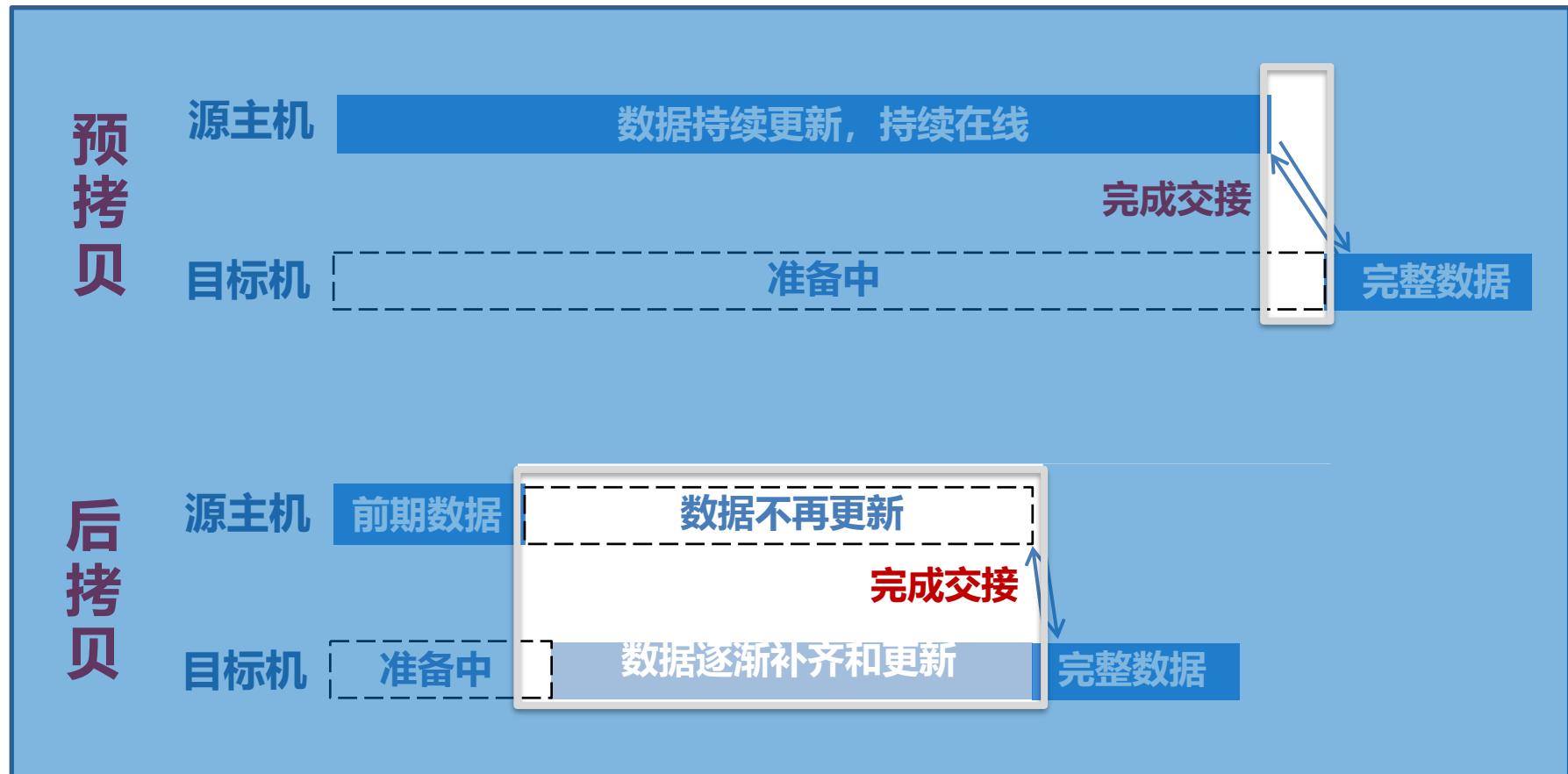


内存在线迁移



内存在线迁移

拷贝传输过程中存在数据可靠性问题



内存在线迁移

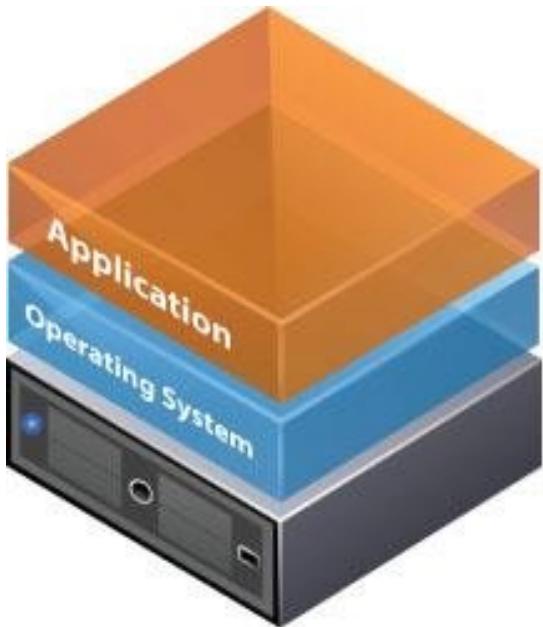
后拷贝传输过程中存在数据可靠性问题

- Pre-Copy's overriding goal is to **keep downtime small** by minimizing the amount of VM state that needs to be transferred during downtime
- Post-copy implementation **cannot recover** from failure of the target node during migration

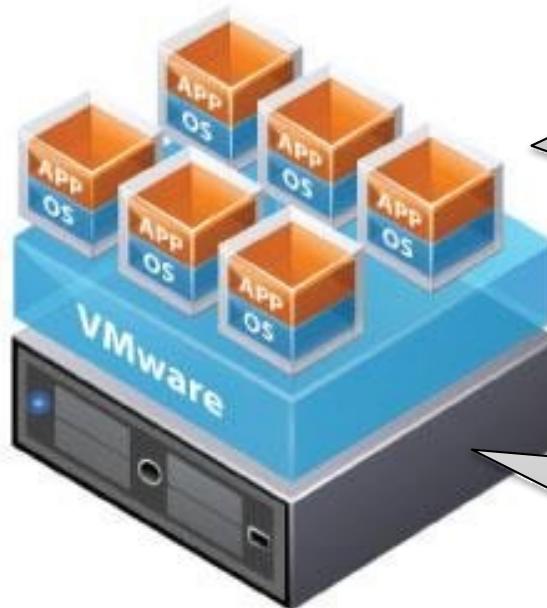
负载整合

□ Server Consolidation (服务器整合)

- 技术上讲，是因为目前服务器利用率不高、能效不好



Traditional Architecture

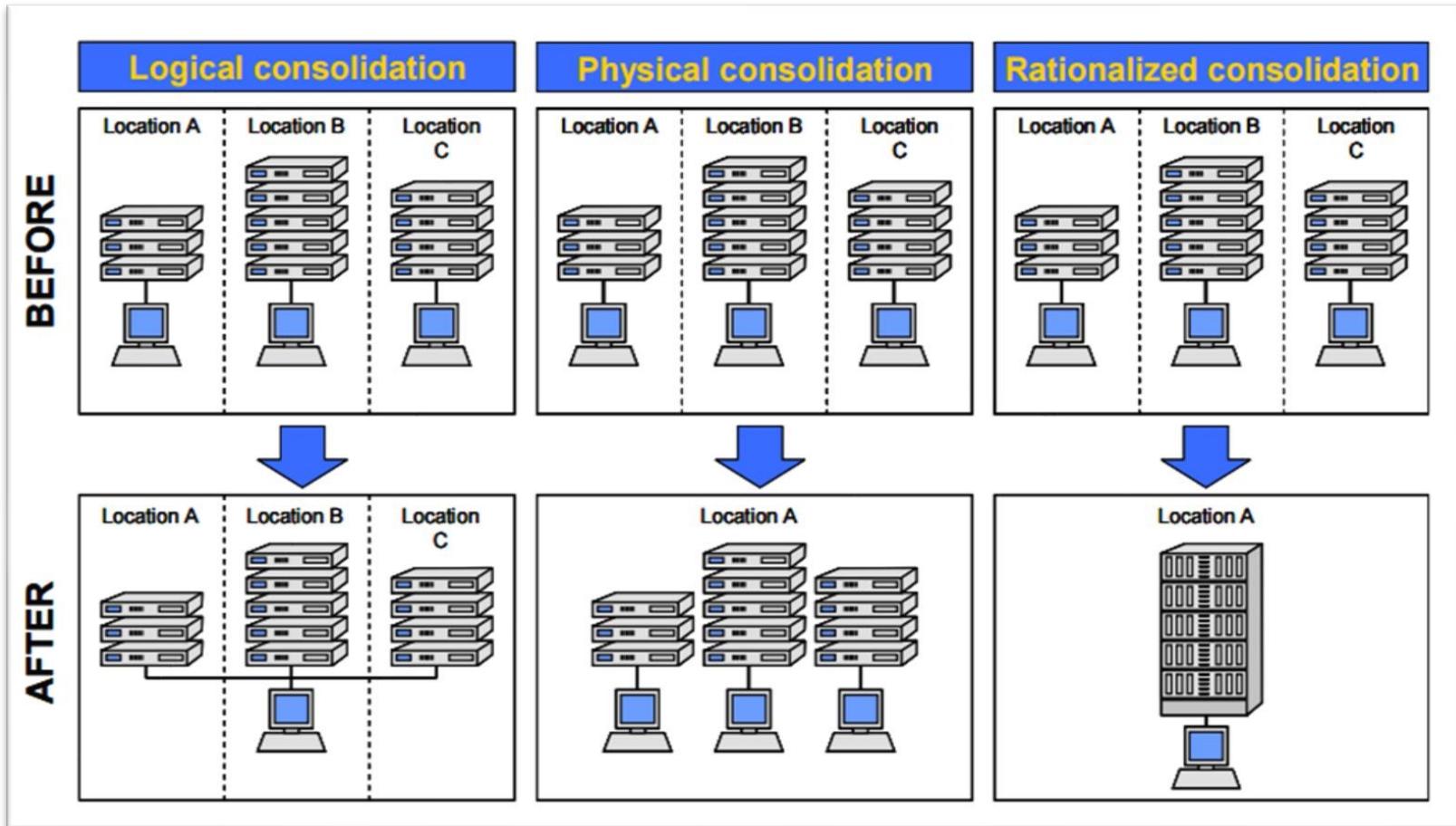


Virtual Architecture

将多个子系统部署到一个物理主机上

达到提升服务器硬件利用率的目的

负载整合



仅仅是逻辑上
实现集中管理

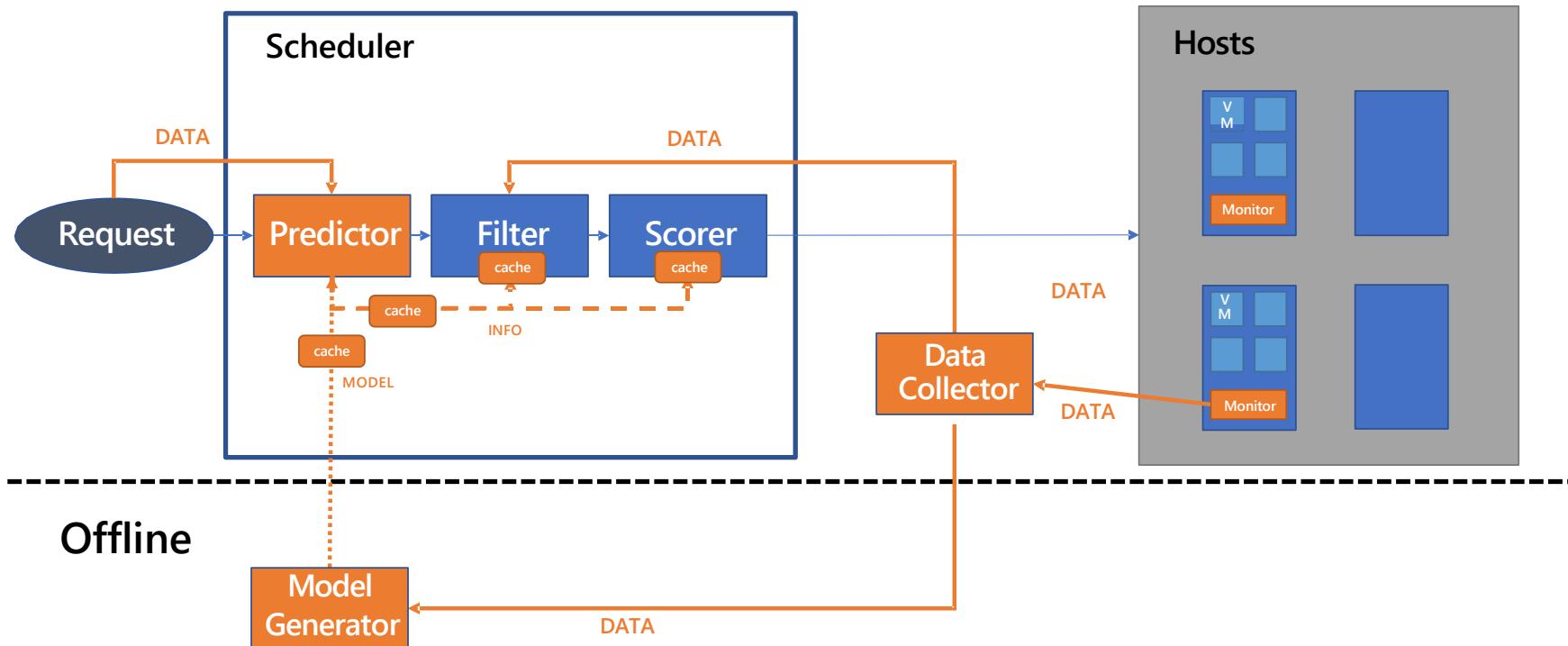
削减数据中心数但
不削减服务器数目

优化整体硬件设
施和管理成本

负载整合

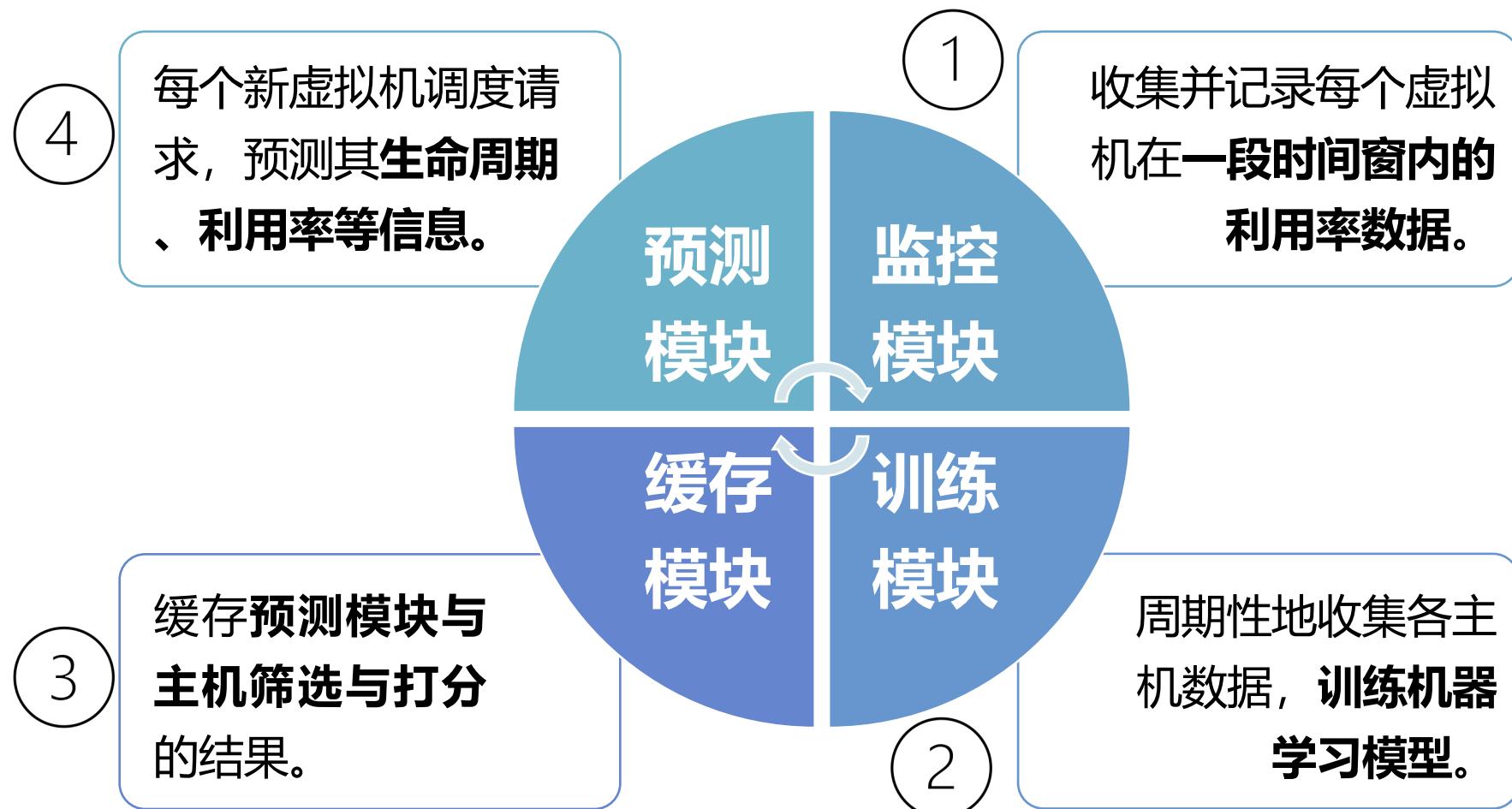
基于预测的智能调度引擎

Online



负载整合

基于预测的智能调度引擎



负载整合

虚拟机整合过程的若干重要参考指标

口峰值平均比 (Peak Average Ratio)

- 资源的峰值需求和平均需求之间的比值

静态整合往往需要从峰值需求出发展开，而动态整合只需要从平均需求展开即可。

口离散系数 (Coefficient of Variability)

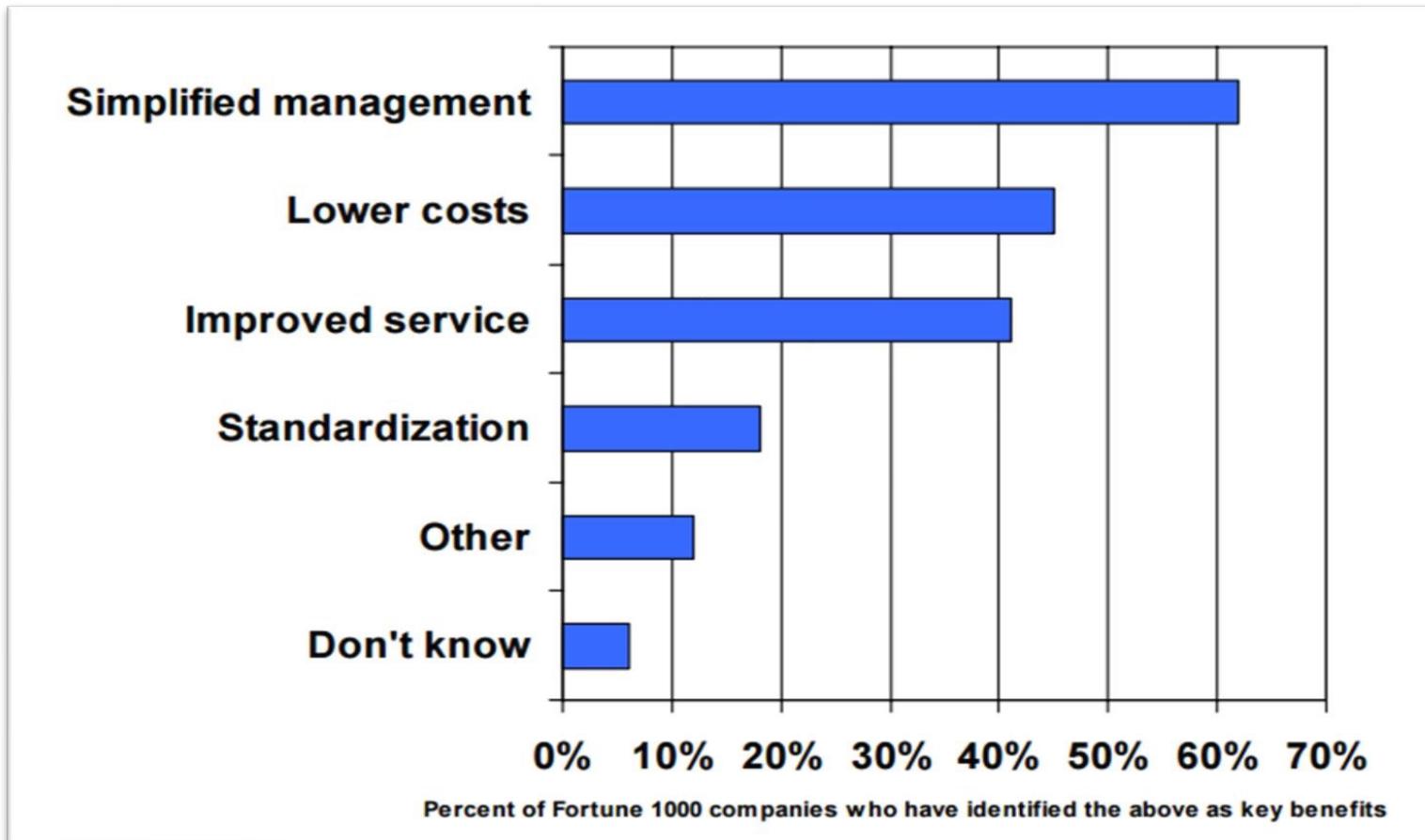
- 标准差和平均值的比值。CoV 值较高往往意味着长尾

$$r_{xy} = \frac{N \sum x_i y_i - \sum x_i \sum y_i}{\sqrt{N \sum x_i^2 - (\sum x_i)^2} \sqrt{N \sum y_i^2 - (\sum y_i)^2}}$$

x_i 和 y_i 表示两个虚拟机在 N 个时间节点下的资源需求

负载整合

云中实施虚拟机负载调度整合的好处





中山大學 软件工程学院
SUN YAT-SEN UNIVERSITY SCHOOL OF SOFTWARE ENGINEERING

谢谢

陈壮彬
软件工程学院
chenzhb36@mail.sysu.edu.cn