

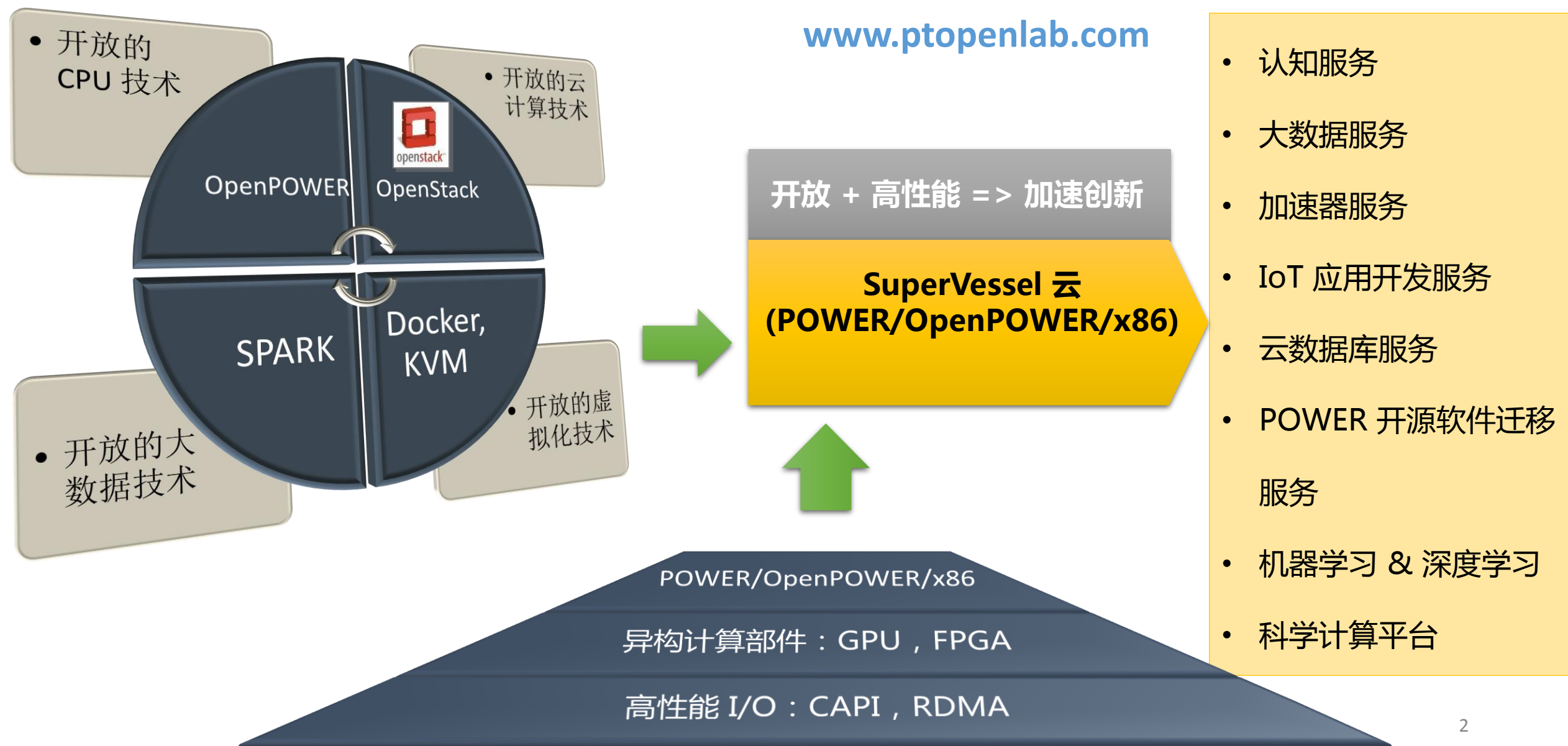
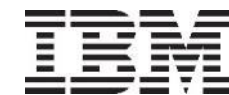
利用和扩展Linux内核功能管理大规模生产环境中的Linux容器资源

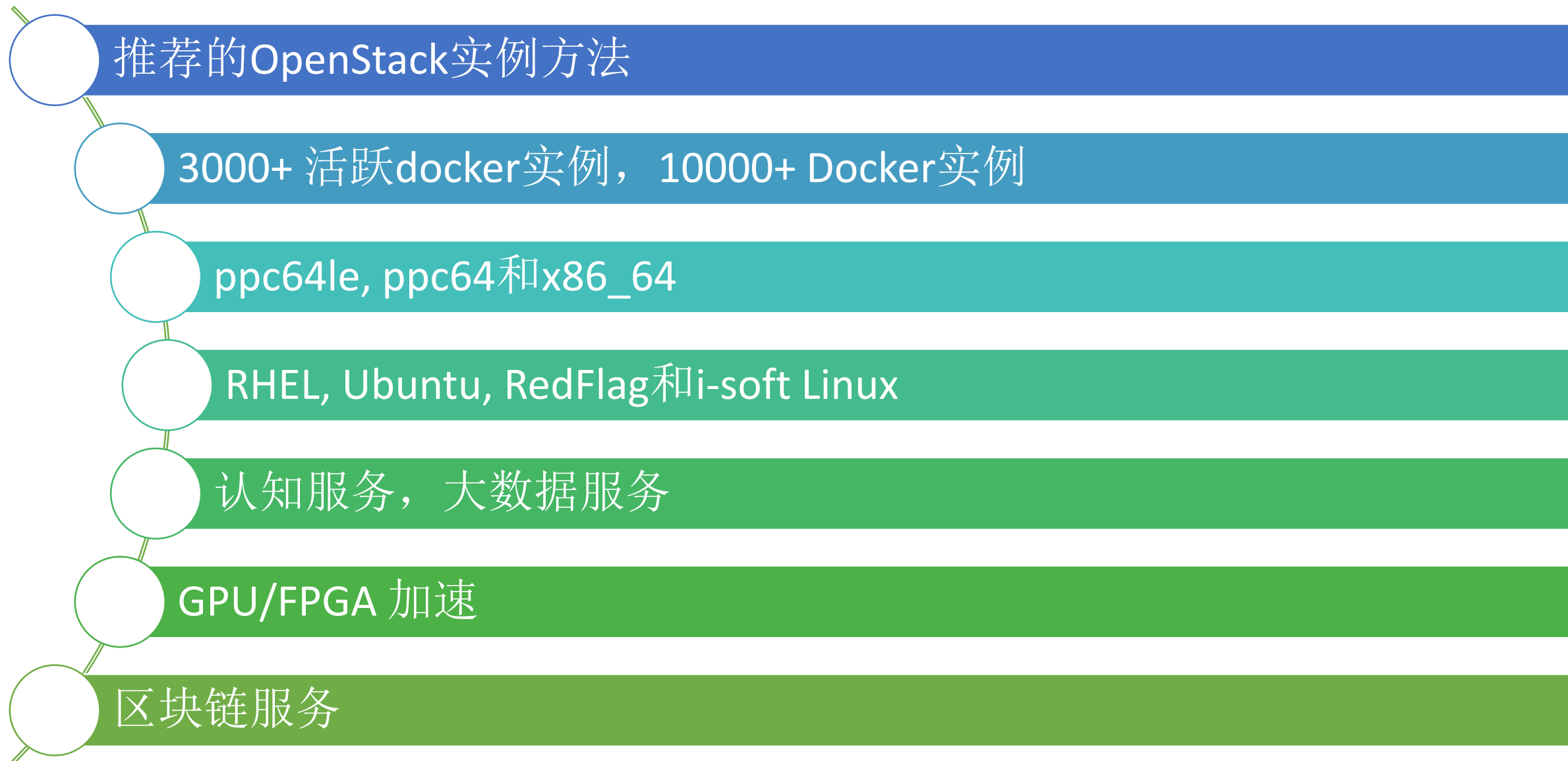
李光成

liguangc@cn.ibm.com

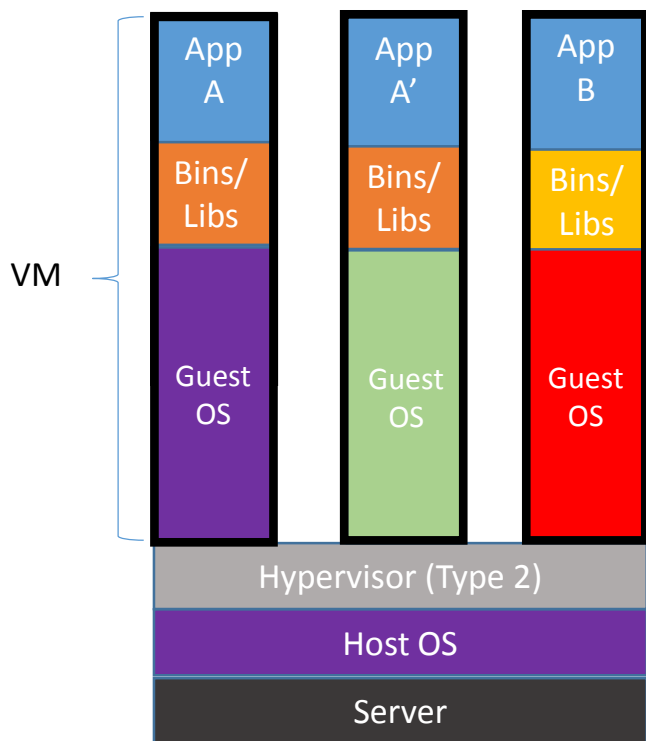
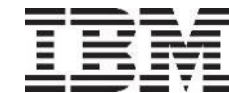
IBM中国研究院

背景 – IBM超能云SuperVessel: 为开发者和生态链打造的公有云





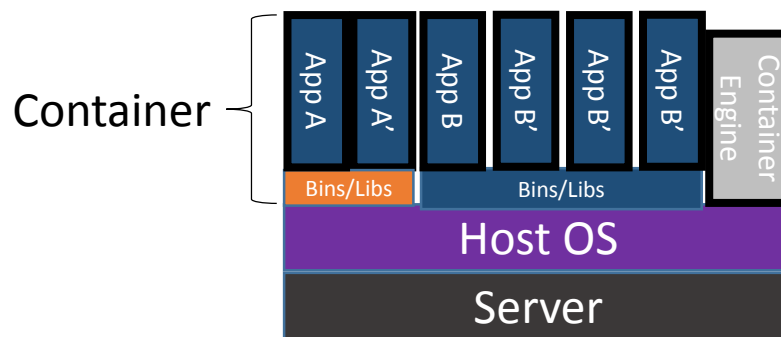
容器 vs 虚拟机- Linux内核需要关心吗？



容器是隔离环境，但是共享Linux内核

带来的好处是超快的部署，极小的开销，易于迁移，能快速重启等

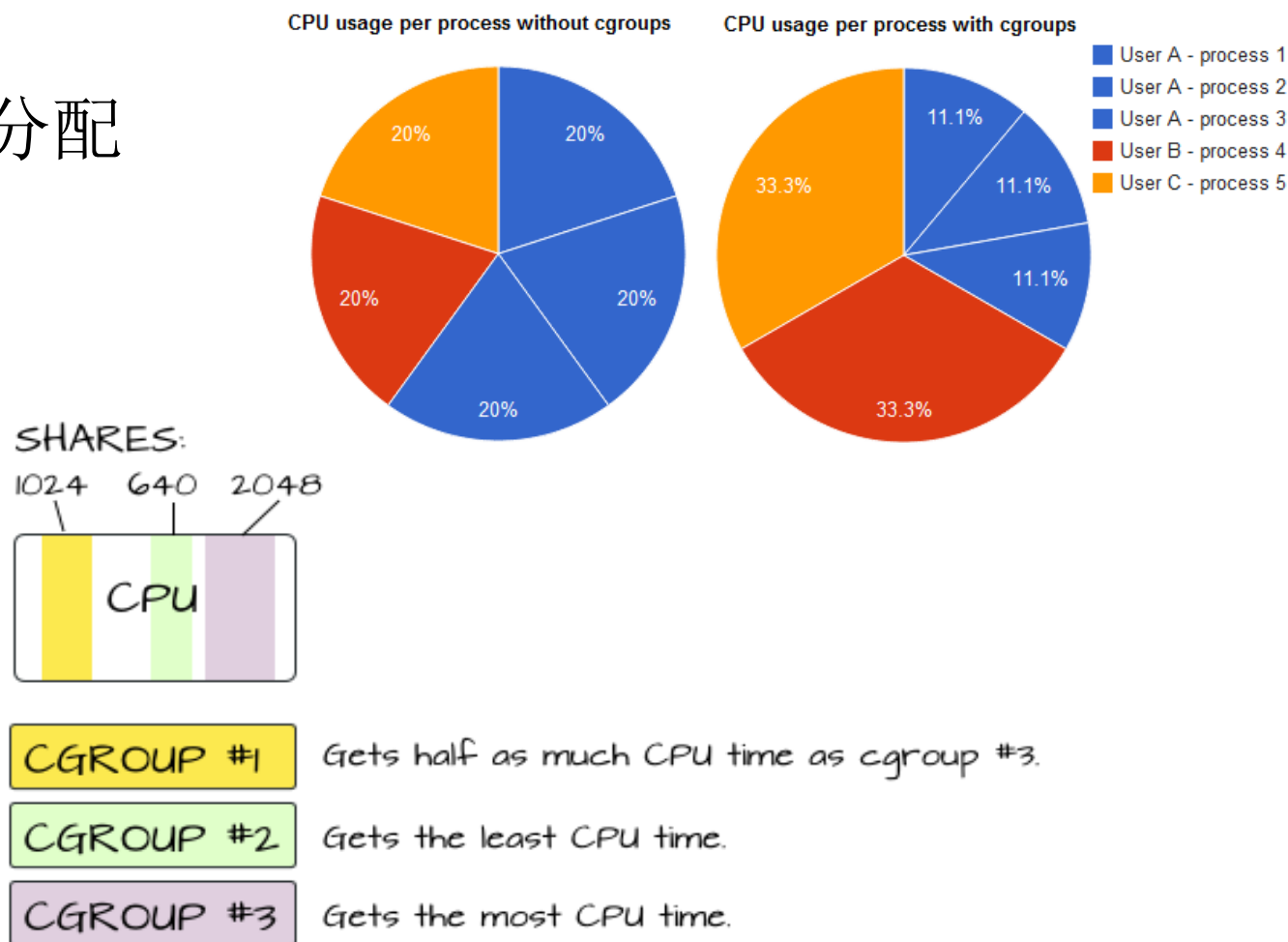
事情另一方面，Linux内核成为了一个事关全局的实体 – 功能性、稳定性和可扩展性的挑战



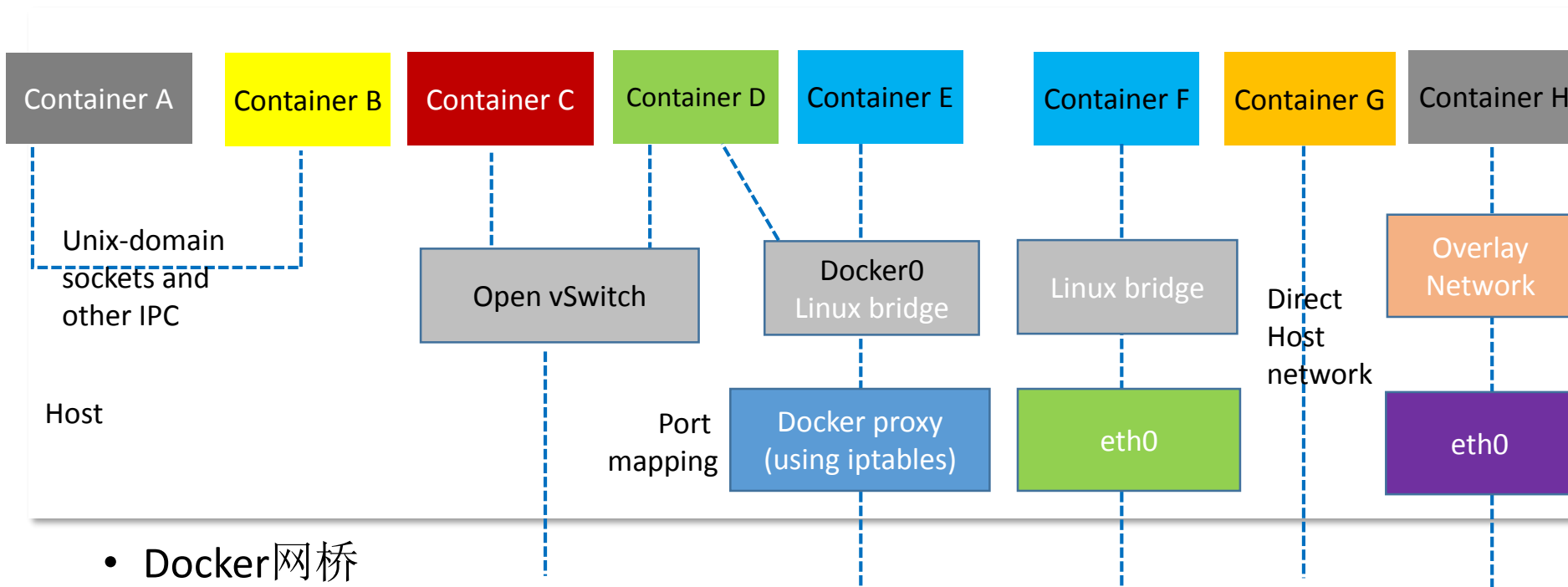
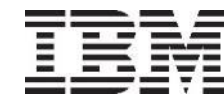
- 改变系统视角
 - Mounts (CLONE_NEWNS)
 - UTS (CLONE_NEWUTS)
 - IPC (CLONE_NEWIPC)
 - PID (CLONE_NEWPID)
 - Networks (CLONE_NEWNET)
 - User (CLONE_NEWUSER)
- 未来可能的新的名字空间
 - Device namespace
 - Time namespace
 - Security namespace



- 管理一组进程的资源分配
 - CPU
 - 内存
 - 磁盘
 - 网络
- 可嵌套

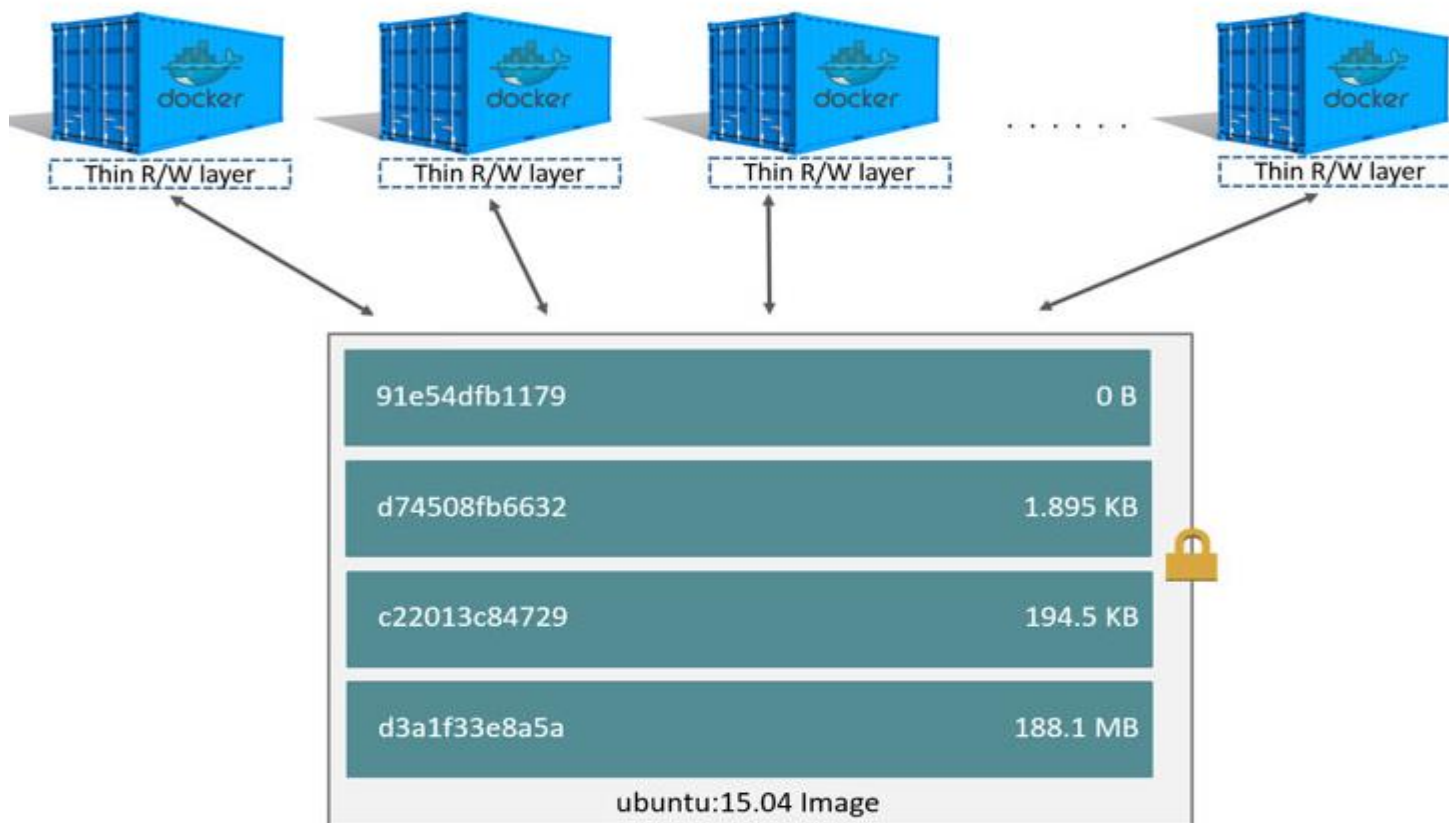
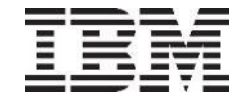


Docker的网络 – 有很多方式是跟虚拟机类似的

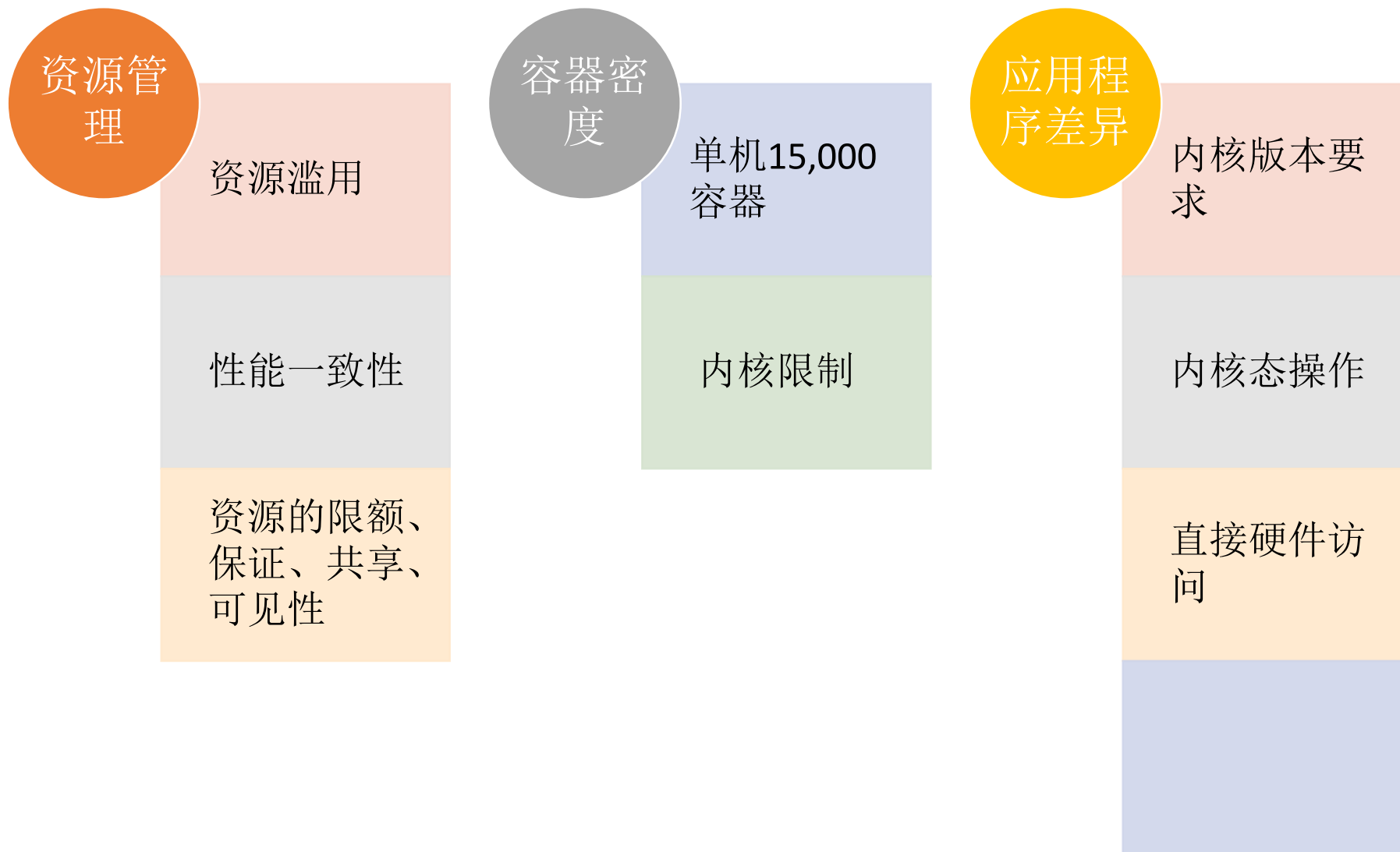






- Docker网桥
- 主机端口绑定
- 主机网络栈 (--net=host)
- 利用其它容器的网络(--net=container)
- 跟其它容器链接(--link)
- 使用OVS
- 使用overlay网络





Copy on Write Storage (CoW)



- 无法使用类似虚机的镜像
 - 秒级启动
 - 在单个主机上启动上数以千计的容器
- Copy-on-Write 是一种自然的方式
- Aufs, OverlayFS, devicemapper, zfs, btrfs



需求		对策
CPU资源限额		cgroup - cpu.cfs_period_us cgroup - cpu.cfs_quota_us
CPU资源保证		cgroup - cpu.shares
CPU绑定		cgroup - cpuset.cpus
QoS vs Best-effort		CPU绑定 + CPU资源限额 + CPU资源保证

需求		对策
内存资源限额		cgroup - memory.limit_in_bytes cgroup - memory.memsw.limit_in_bytes cgroup - memory.oom_control
内存超配		SSD和NVME作为交换分区设备 zswap
内存资源保证		进行中，备选方案：memmap，Huge Page，NUMA模拟。新的cgroup能力需求
内存带宽管理		调研中...

- 磁盘空间管理

- 需要文件系统支持
- 目前在用ZFS支持磁盘空间限额和磁盘空间保证
- 正在尝试一种通用的磁盘空间管理方法 – 通过用户的文件系统配额和用户命名空间
- 内核能提供进一步的能力来支持吗？

- 磁盘带宽管理 - cgroup

- blkio.throttle.read_bps_device
- blkio.throttle.write_bps_device
- blkio.throttle.read_iops_device
- blkio.throttle.write_iops_device

- Linux tc(traffic control)
- cgroup net_cls, net_prio
- 入流量和出流量
- *tc qdisc add dev eno1 root tbf rate 1mbit latency 50ms burst 2048*

- cgroup: devices.allow devices.deny
- 设备共享

- CPU，内存
- 设备列表
- 网络带宽
- 磁盘信息
- cgroup aware /proc file system?

- 银弹 - 在虚机中运行容器
- 内核版本要求
 - 调整Docker主机内核
- 内核态操作 – 内核模块，驱动程序，特殊操作
 - 特权模式 + 用户命名空间
- unikernel?
- 终极目标：内核改进 – 多内核实例(前期调研中)

➤ 单机15,000容器

➤ ulimits

- `echo "root soft nofile 1048576" > /etc/security/limits.conf`
- `echo "root hard nofile 1048576" >> /etc/security/limits.conf`
- `echo "root soft nproc 1048576" >> /etc/security/limits.conf`
- `echo "root hard nproc 1048576" >> /etc/security/limits.conf`

➤ `/proc/sys/kernel/pty/max`

➤ Linux bridge logical ports

- Linux 内核能力对于容器至关重要
 - cgroup
 - namespace
 - TCP/IP协议栈
 - 文件系统
- 容器对Linux内核的进一步需求
 - 内存保证
 - 内存带宽管理
 - 网络带宽控制
 - 时间命名空间
 - 资源视图
 - 多内核实例
 - ...

Thank You !

