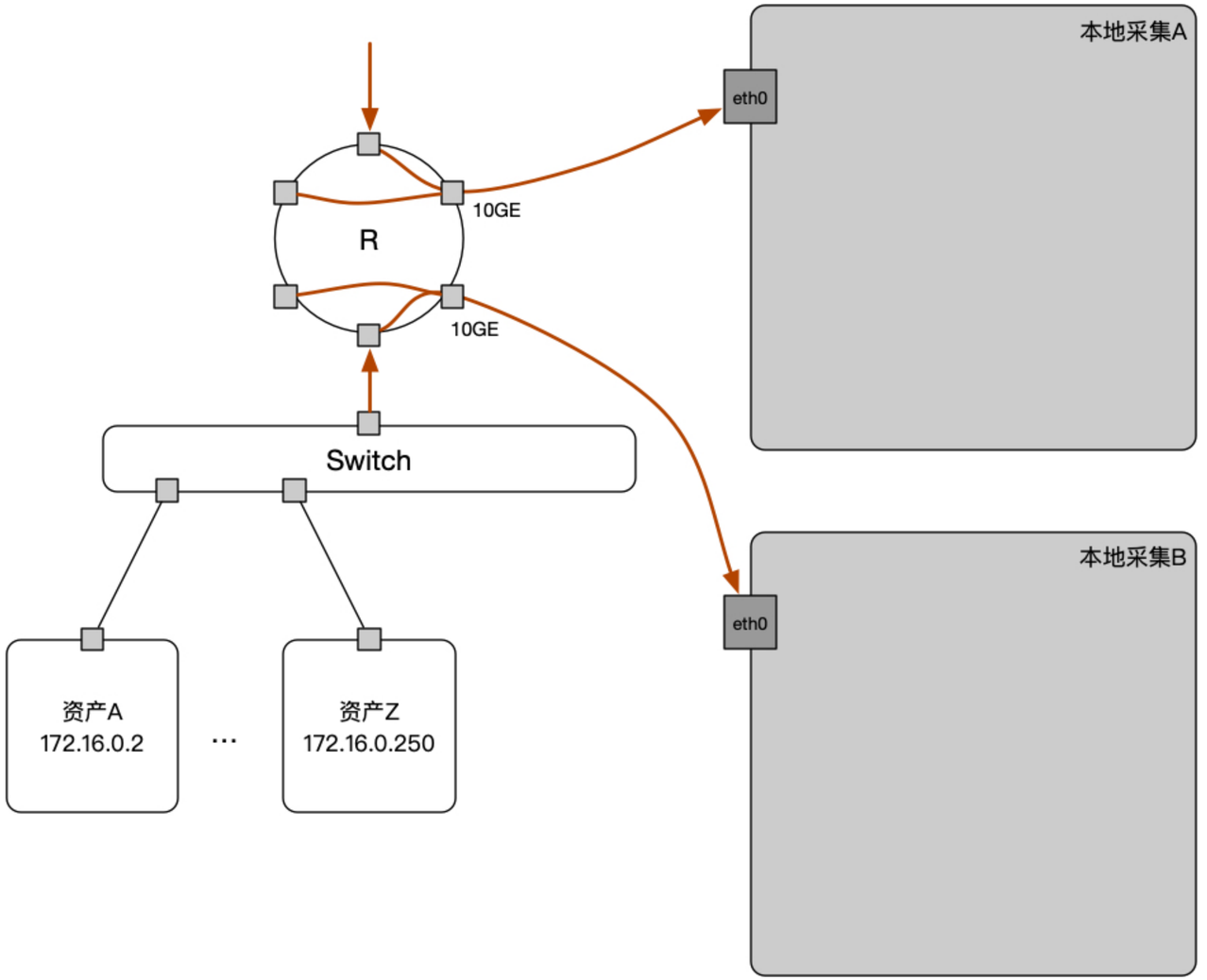
IDS at scaling

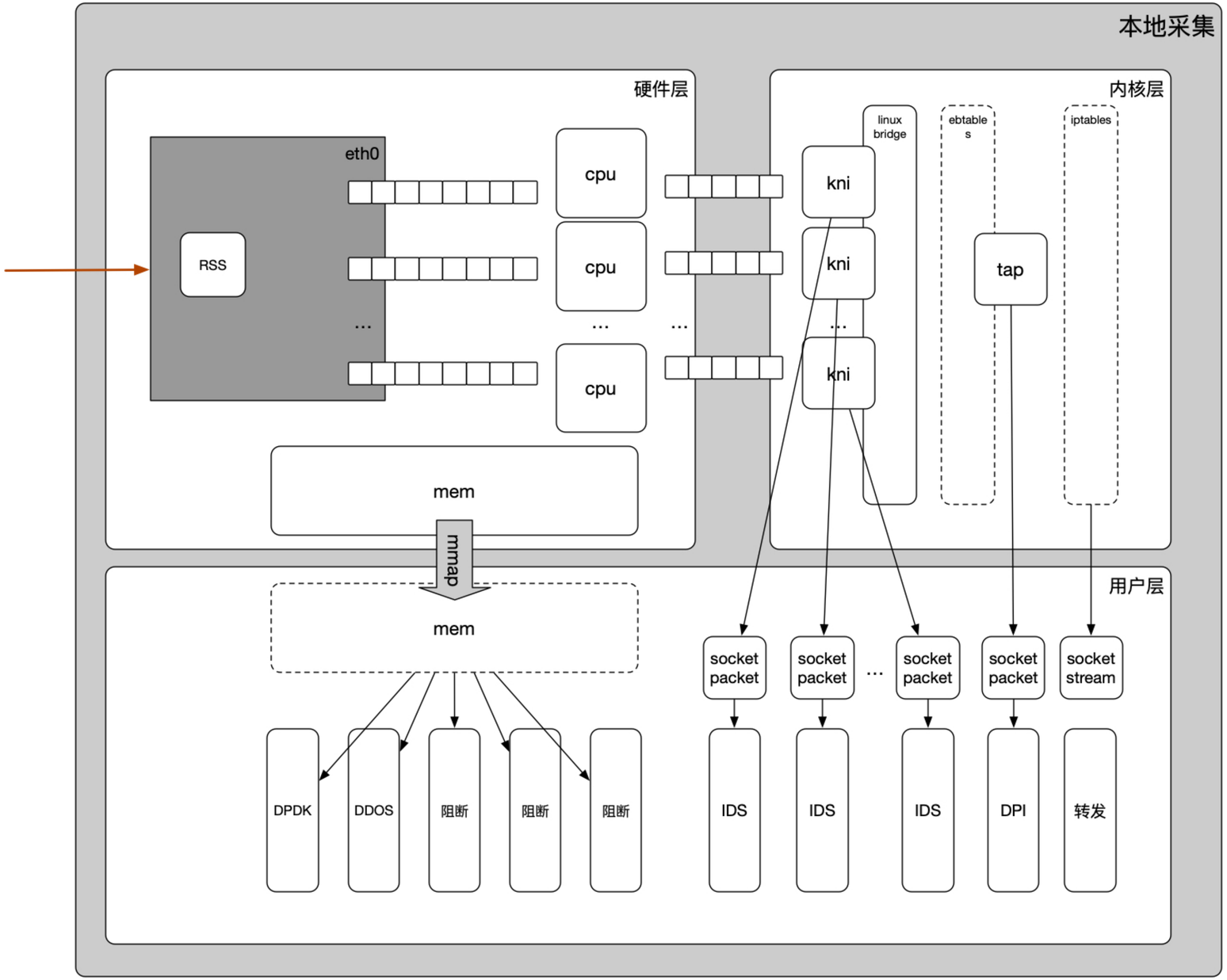
翟增辉 2020.04.22

刃甲是一个以流量为输入，经过本地高性能处理，发往消息队列集群，经分布式实时计算后提供「威胁检测」、「资产分析」、「流量审计」、「旁路阻断」、「动态引流」、「网站防篡改」等业务的微服务系统。

1. 本地处理 - 部署


本地采集A
eth0
10GE
R
10GE
Switch
本地采集B
eth0
资产A
172.16.0.2
...
资产Z
172.16.0.250

1. 本地处理 - 架构


本地采集
硬件层
内核层
eth0
RSS
cpu
cpu
cpu
linux bridge
ebtables
iptables
kni
kni
kni
tap
mem
mmap
用户层
mem
socket packet
socket packet
socket packet
socket packet
socket stream
DPDK
DDOS
阻断
阻断
阻断
IDS
IDS
IDS
DPI
转发

1. 本地处理 - UIO

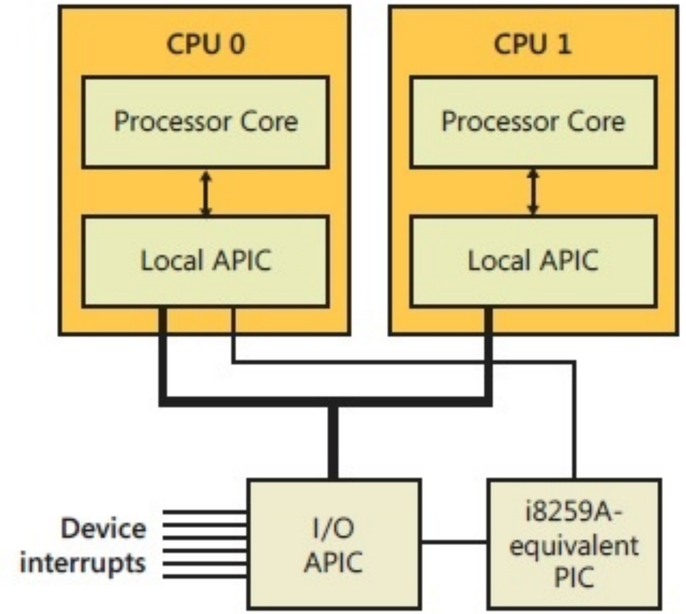


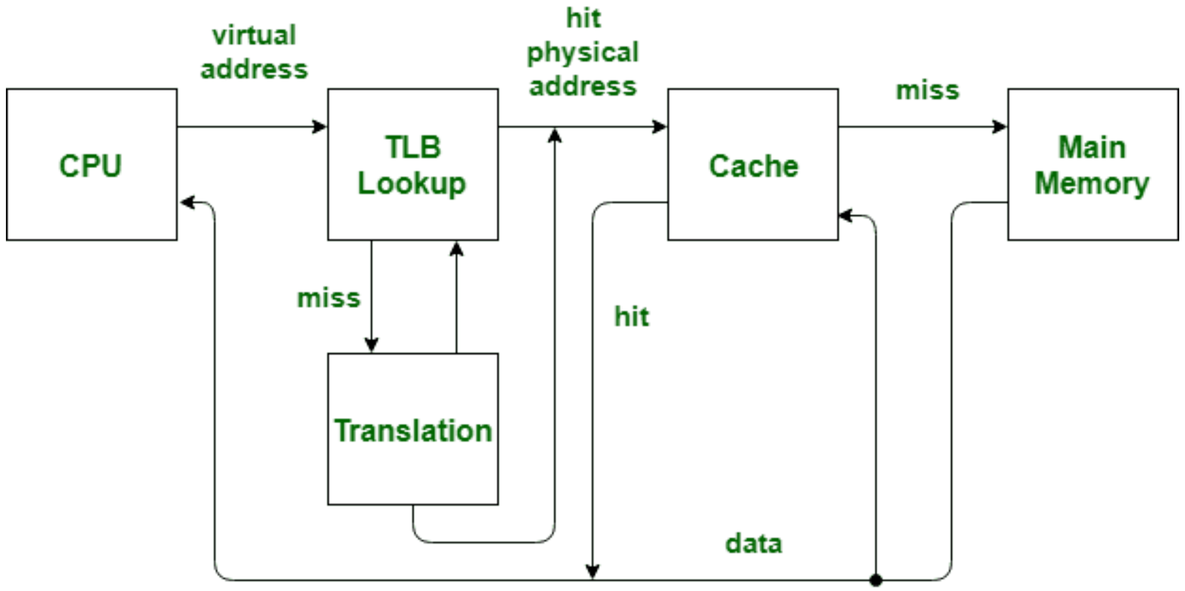
FIGURE 3-2 x86 APIC architecture

1. IRQ
    1. cat /proc/interrupts
    2. 禁止了原硬件中断
    3. 免调用cpu中断例程

2. UIO
    1. lsmod | grep uio
    2. 自定义中断例程
    3. 用户态ioctl

3. MMAP
    1. 零拷贝

1. 本地处理 - hugepages



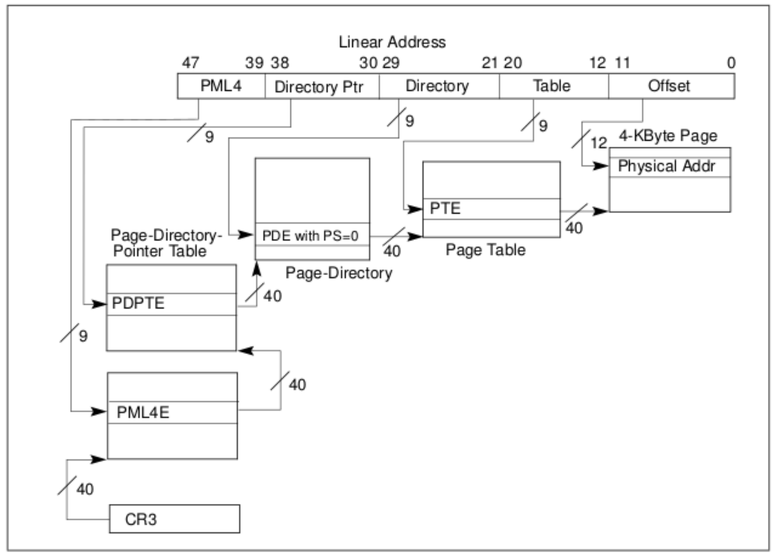


Figure 4-8. Linear-Address Translation to a 4-KByte Page using IA-32e Paging

```
[root@localhost hugepage]# perf stat -B -e dTLB-load-misses,iTLB-load-misses ./hugepage

 Performance counter stats for './hugepage':

             3,336      dTLB-load-misses
             2,375      iTLB-load-misses

       0.952872237 seconds time elapsed

[root@localhost hugepage]#
[root@localhost hugepage]# perf stat -B -e dTLB-load-misses,iTLB-load-misses ./mmap

 Performance counter stats for './mmap':

            45,066      dTLB-load-misses
            19,458      iTLB-load-misses

       1.048320112 seconds time elapsed
```

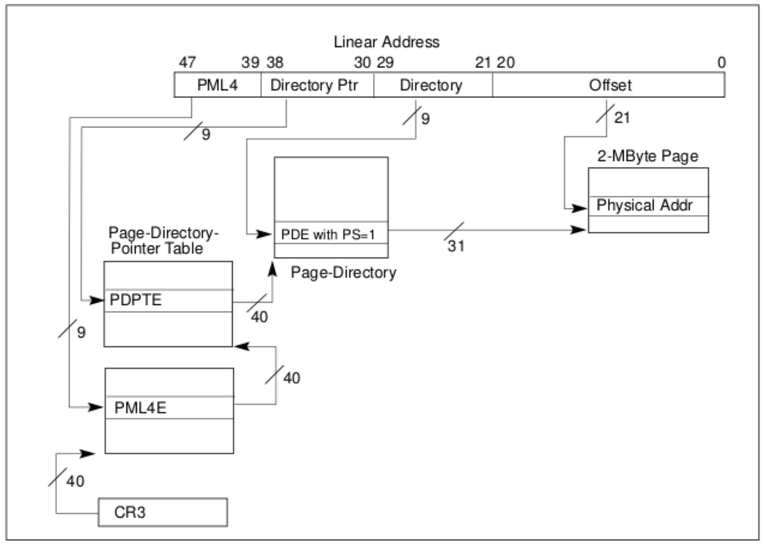
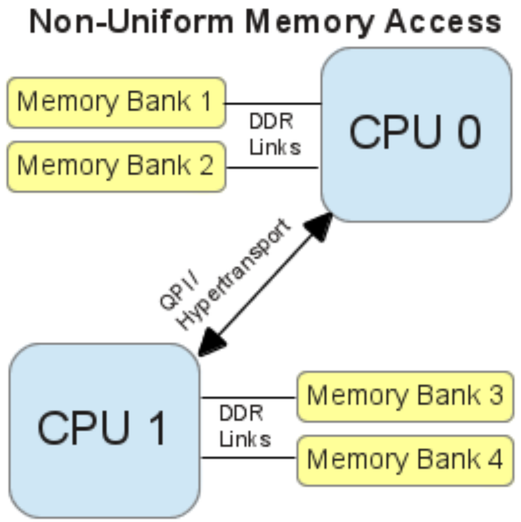
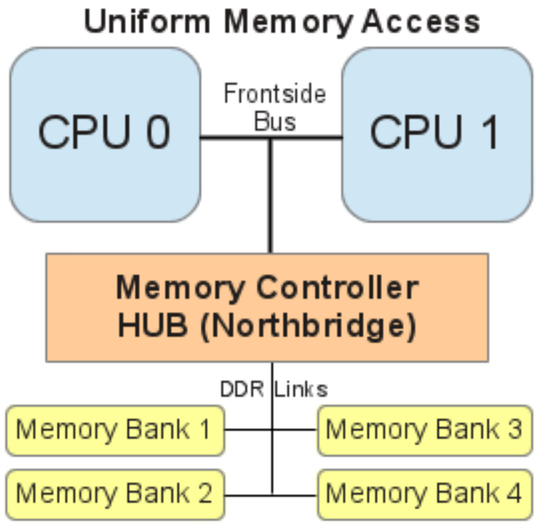


Figure 4-9. Linear-Address Translation to a 2-MByte Page using IA-32e Paging

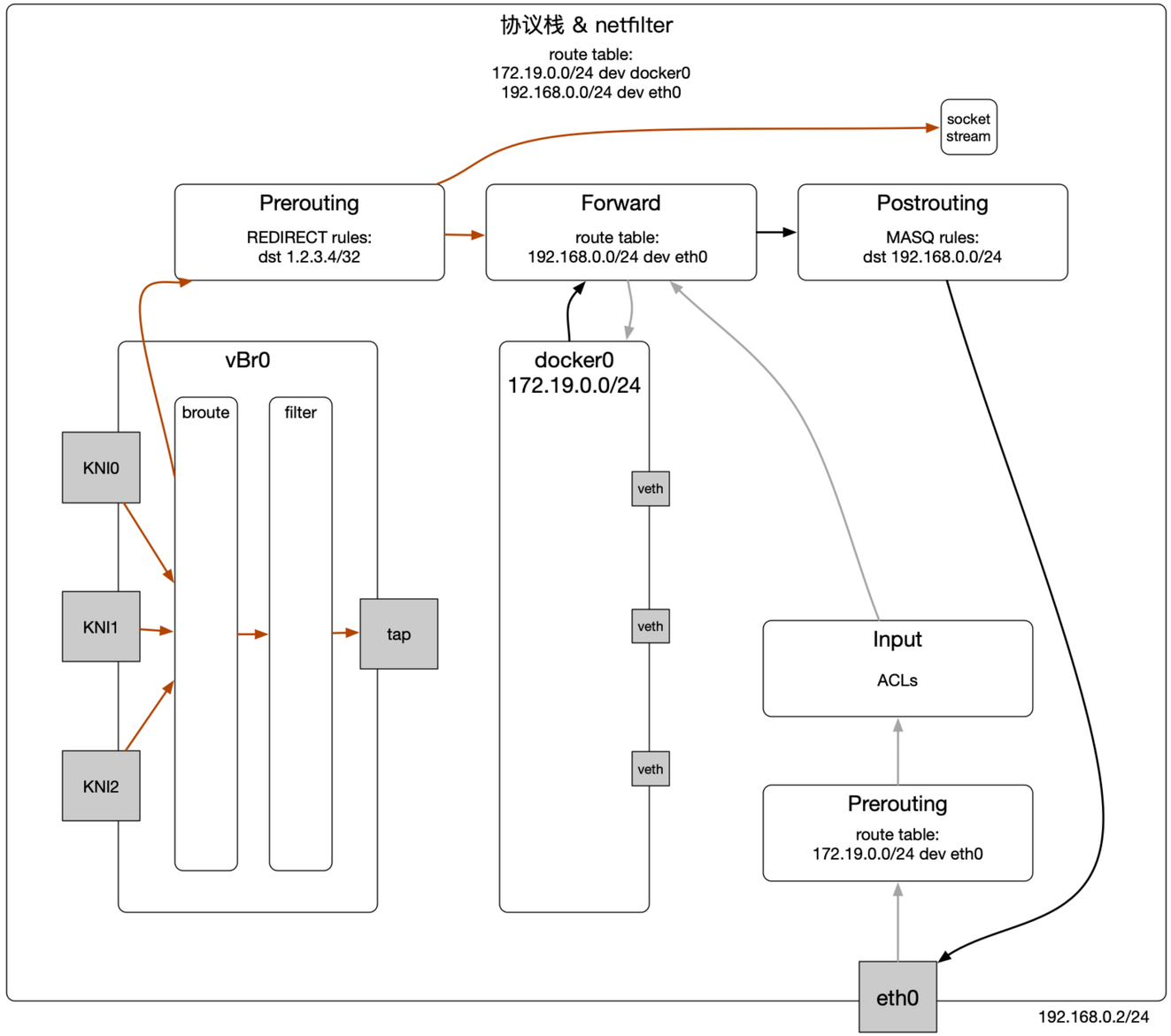
1. 本地处理 - 局部计算





1. NUMA
    1. 非对称内存访问
2. 数据本地化
    1. 网卡offload - rss
    2. 网卡多队列分别绑定核心
3. cpu affinity（绑核）
    1. 减少缓存miss
    2. perf with caches-misses
4. threat local
    1. gcc __thread 关键字

1. 本地处理 - 虚拟网络



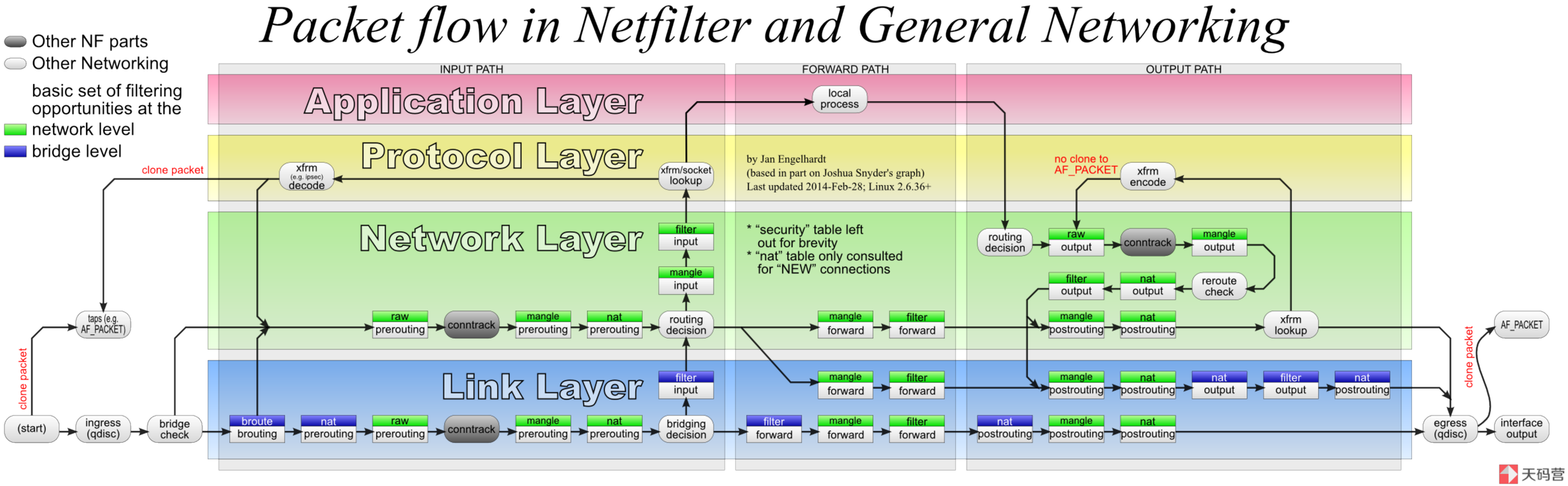
涉及点：

1. dpdk kni
2. linux bridge
3. ebtables (broute, filter)
4. iptables (redirect, nat, masq)
5. tap device
6. docker方案

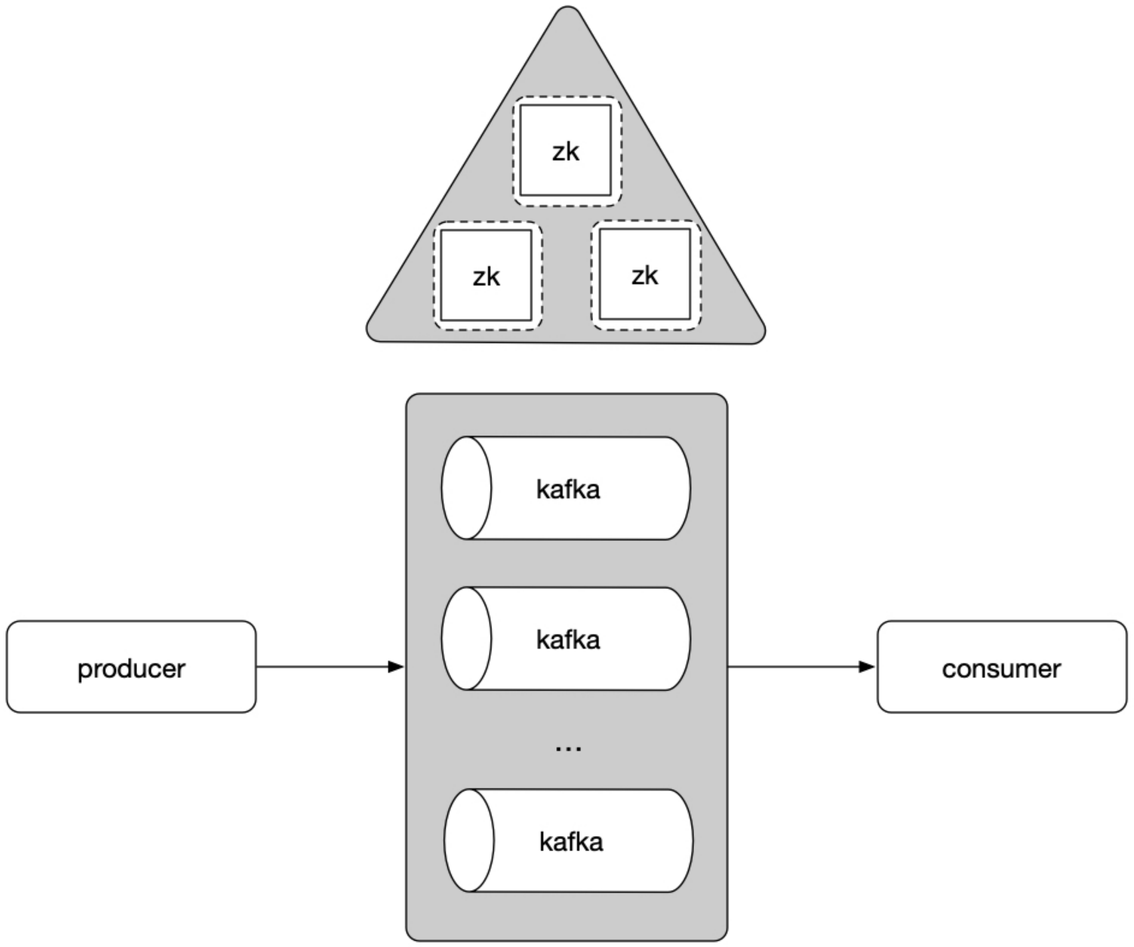
注意点

1. rp_filter: 禁止源地址校验
2. ip_forward: 允许转发
3. bridge: setageing 0; 让不存在的fdb消亡
4. bridge: nf-call-iptables 0

1. 本地处理 - 参考

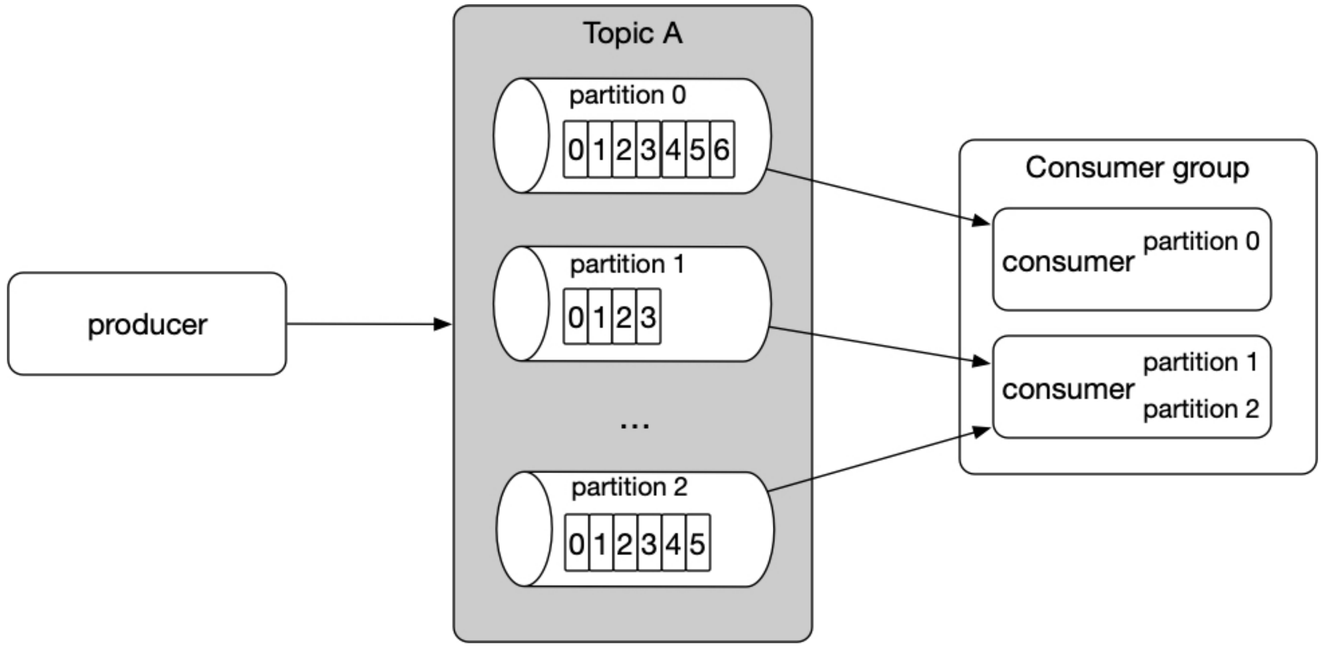

Packet flow in Netfilter and General Networking
Other NF parts
Other Networking
basic set of filtering opportunities at the network level
bridge level
INPUT PATH
FORWARD PATH
OUTPUT PATH
Application Layer
Protocol Layer
Network Layer
Link Layer
by Jan Engelhardt
(based in part on Joshua Snyder's graph)
Last updated 2014-Feb-28; Linux 2.6.36+
* "security" table left out for brevity
* "nat" table only consulted for "NEW" connections
clone packet
no clone to AF_PACKET
local process
xfrm (e.g. ipsec) decode
xfrm/socket lookup
xfrm encode
taps (e.g. AF_PACKET)
(start)
ingress (qdisc)
bridge check
broute brouting
routing decision
reroute check
egress (qdisc)
interface output
AF_PACKET
raw
conntrack
mangle
nat
filter
bridging decision
xfrm lookup

2. 消息队列集群 - Why MQ



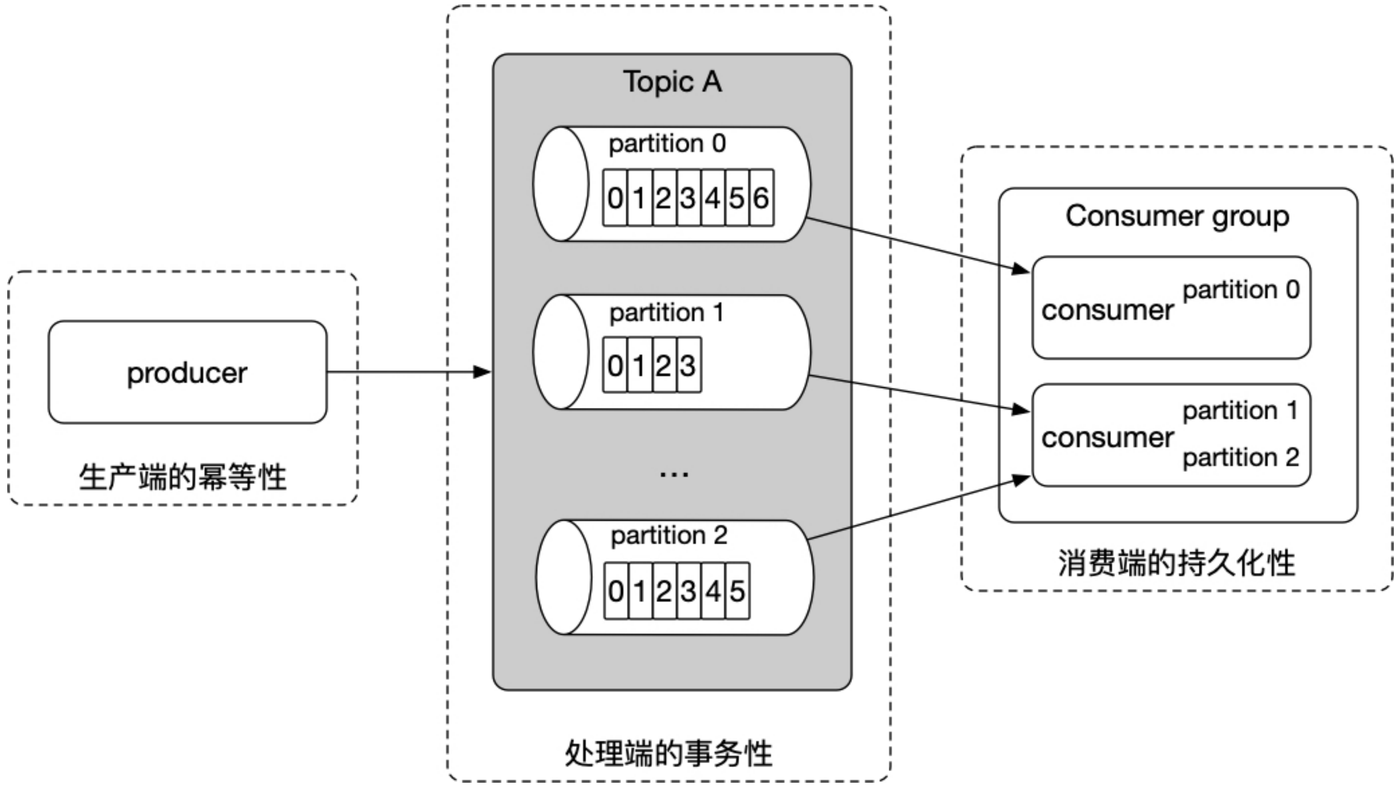
1. 异步化： 消峰、提升性能
2. 集群化：scale out
3. 解耦合组件：以生产/消费取代长连接、rpc、rest等协议
4. 消息总线语义：分布式系统里的数据面表达
5. 系统之间耦合：消费代替api，表达增量数据

2. 消息队列集群 - Why Kafka



1. 消费组模型：负载均衡
2. 持久化：下游组件数据安全性
3. poll模型：下游组件不会过载，保障稳定性
4. 高性能：sendfile保障零拷贝
5. 大数据亲和性：zookeper, spark, flink等原生适配
6. 其他：单partition保障顺序性、流处理能力

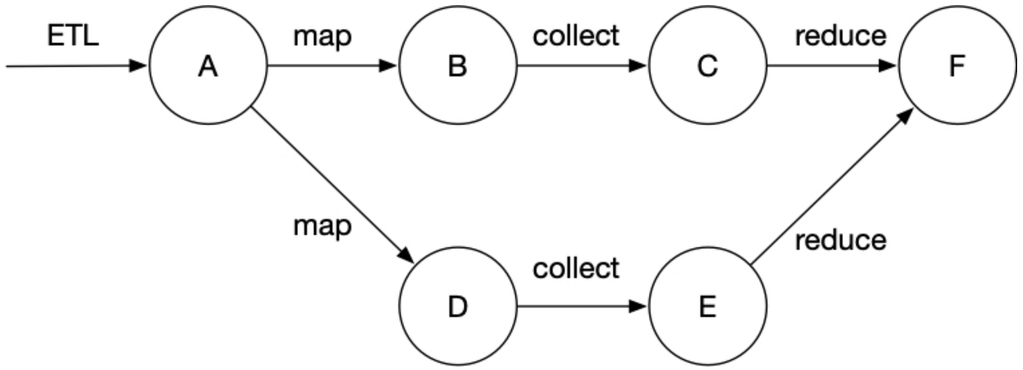
2. 消息队列集群 - 端到端一致性



生产消费模型消息处理语义：

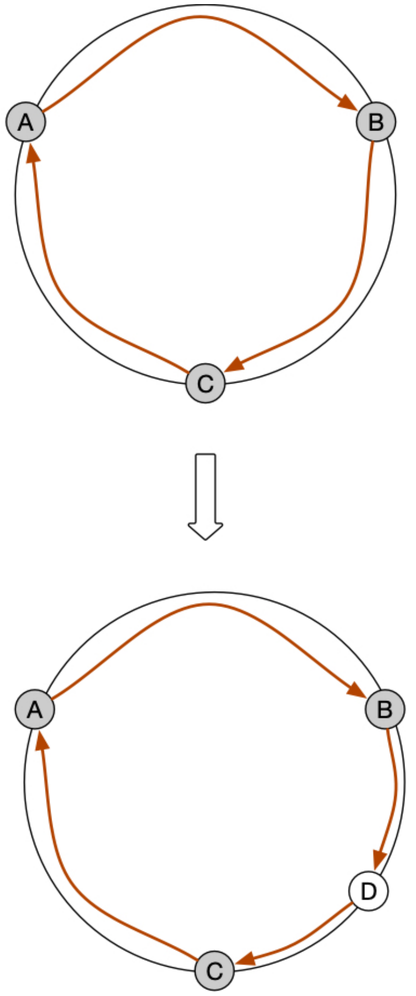
1. 最多一次（at-most-once）：高性能处理通道
2. 最少一次（at-least-once）：确定性处理通道
3. 确定一次（exactly-once）：准确处理通道

3. 分布式处理 - 架构



1. ETL（数据的提取、转换、加载）
2. DAG（算子的有向无环图）
3. 计算算子化（统计、回归、分类、聚类...）
4. 分布式基石（scale out based on map/reduce）
5. 流处理/离线处理的基础架构不同

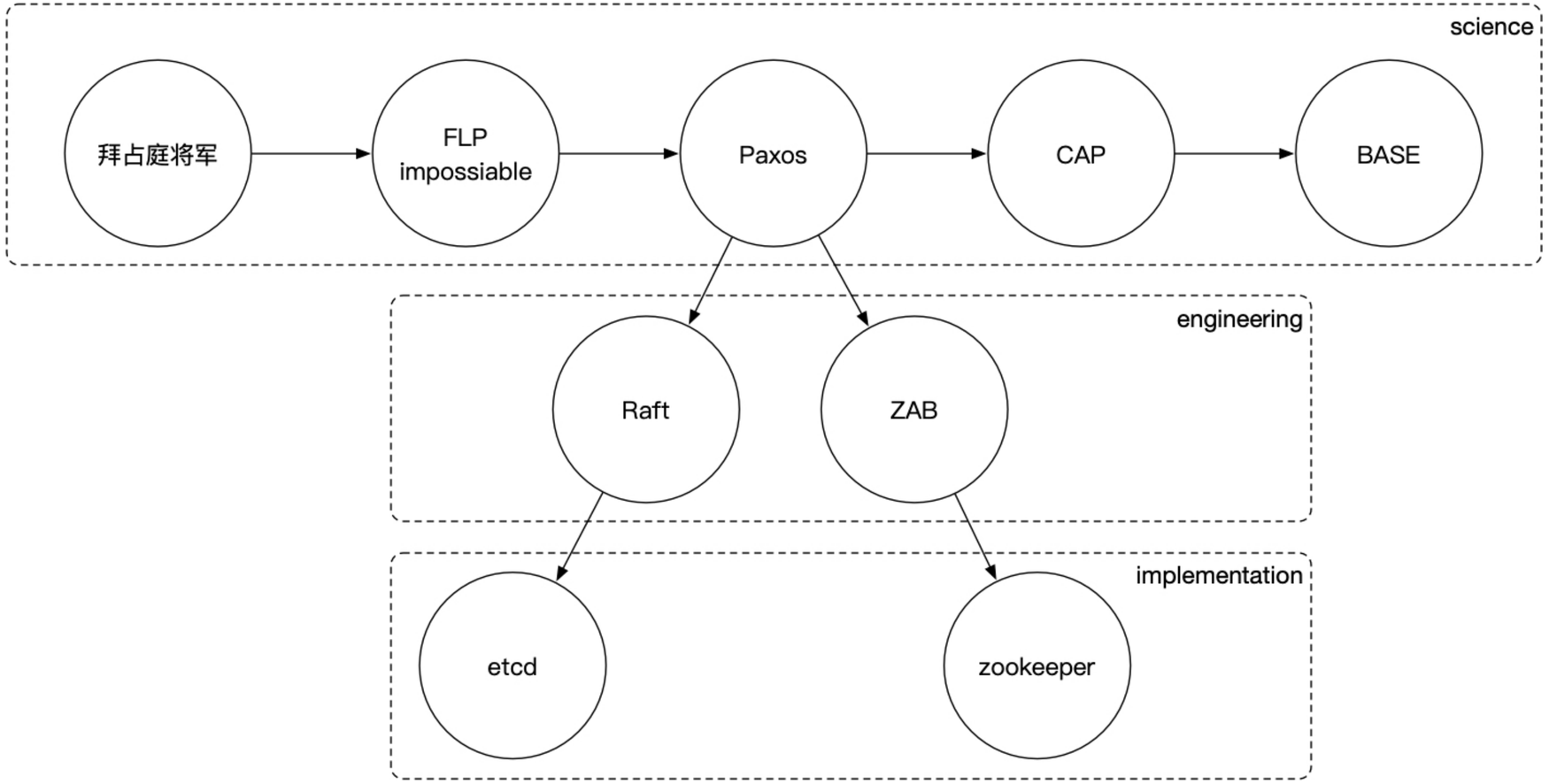
3. 分布式处理 - 一致性哈希的map方案



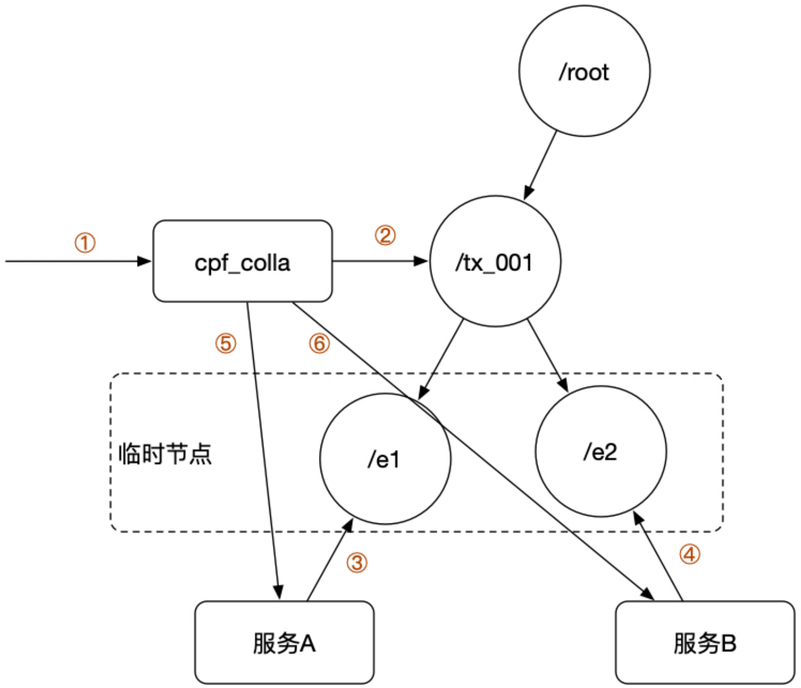
1. 线性空间 -> 环形空间
2. 最小化新增/消减节点带来的影响
3. 无变更下的幂等性，下游计算不受影响
4. 集群化：scale out
5. 哈希：拆分混杂在一起的状态

3. 分布式处理 - 基础理论

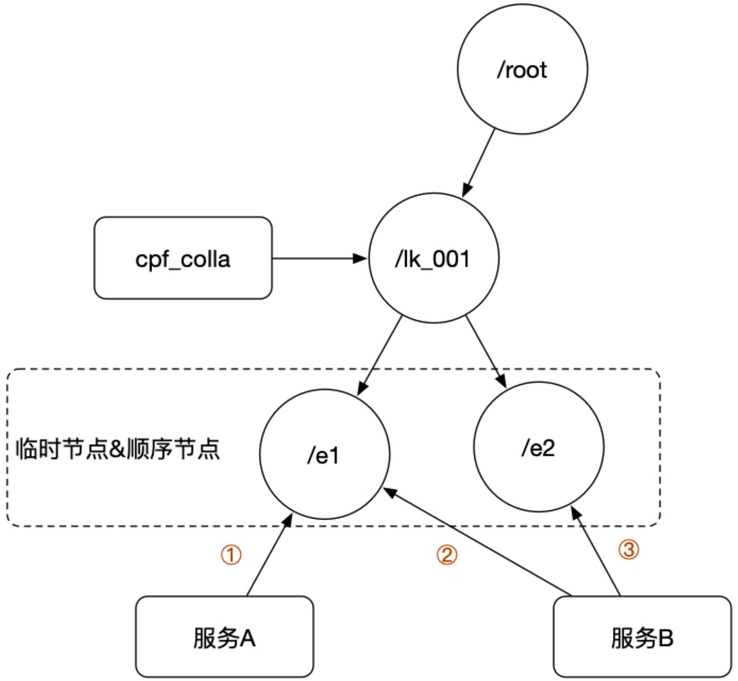
1. **Termination(liveness): 所有存活的进程最终会结束并选取一个值**
2. **Agreement(safety): 所有存活进程必须同意同一个值**
3. **Validity(safety)：最终达成的值必须是是某个进程提议的值**



3. 分布式处理 - 协同



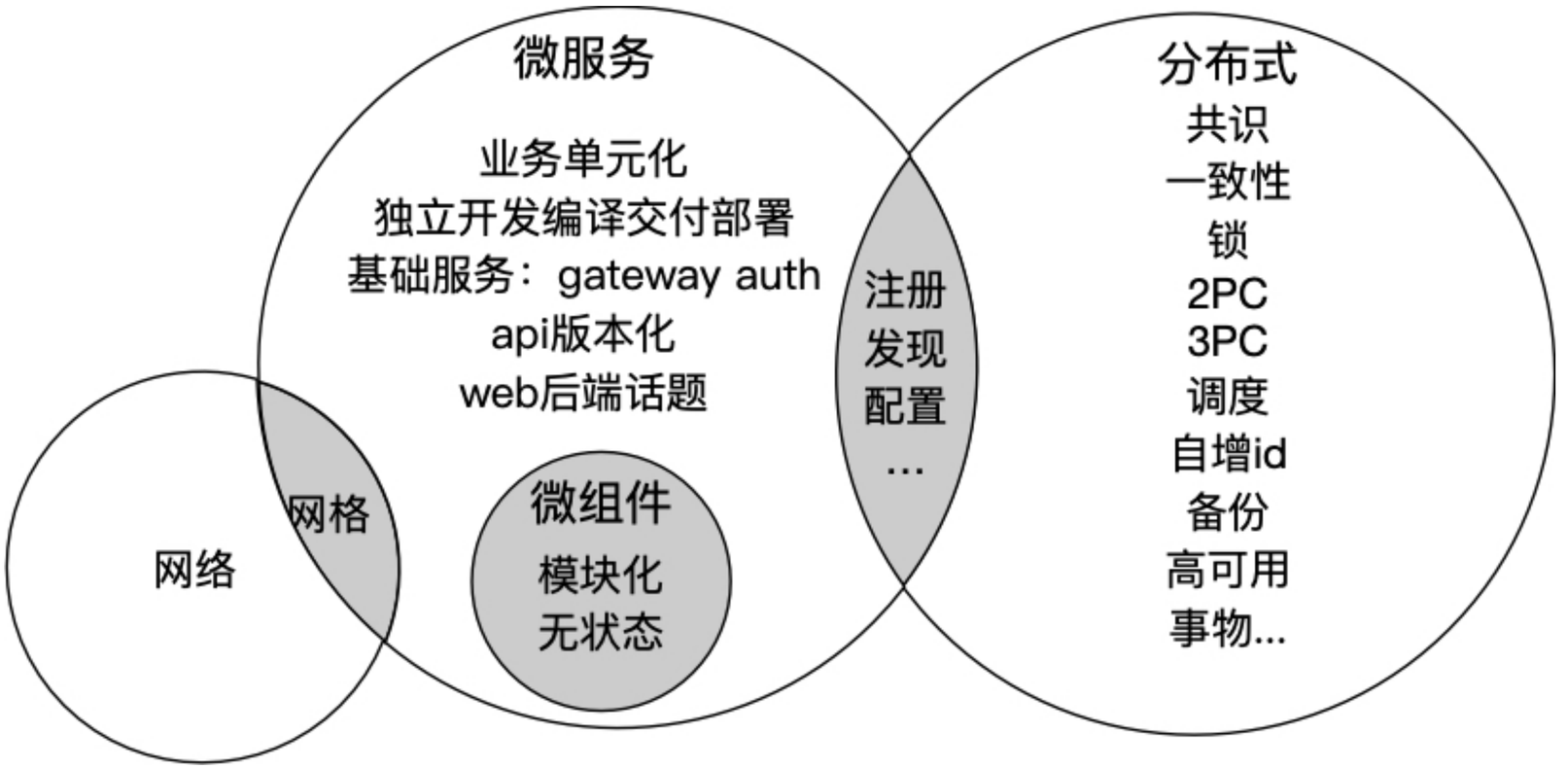
2PC



分布式锁

1. 服务注册/发现
2. 服务配置
3. 互斥调度/定时调度
4. 分布式锁
5. 多阶段提交（2PC, 3PC...）
6. 全局自增唯一id
7. ...

4. 微服务 - 理念


微服务
业务单元化
独立开发编译交付部署
基础服务：gateway auth
api版本化
web后端话题
网络
网格
微组件
模块化
无状态
注册
发现
配置
...
分布式
共识
一致性
锁
2PC
3PC
调度
自增id
备份
高可用
事物...

4. 微服务 - 无状态化

无状态：本次请求与上次或者下次请求无关

1. 业务对扩容/缩容无感知
2. 把状态集中给存储
3. 多实例部署
4. 无缝高可用
5. 无缝容器化
6. 无缝集群化

4. 微服务 - 基础抽象

API Gateway

1. 统一入口
2. 路由
3. 负载均衡
4. Qos
5. 熔断
6. 服务降级
7. 调用日志
8. 请求聚合
9. 超时控制
10. rest/graphql/rpc

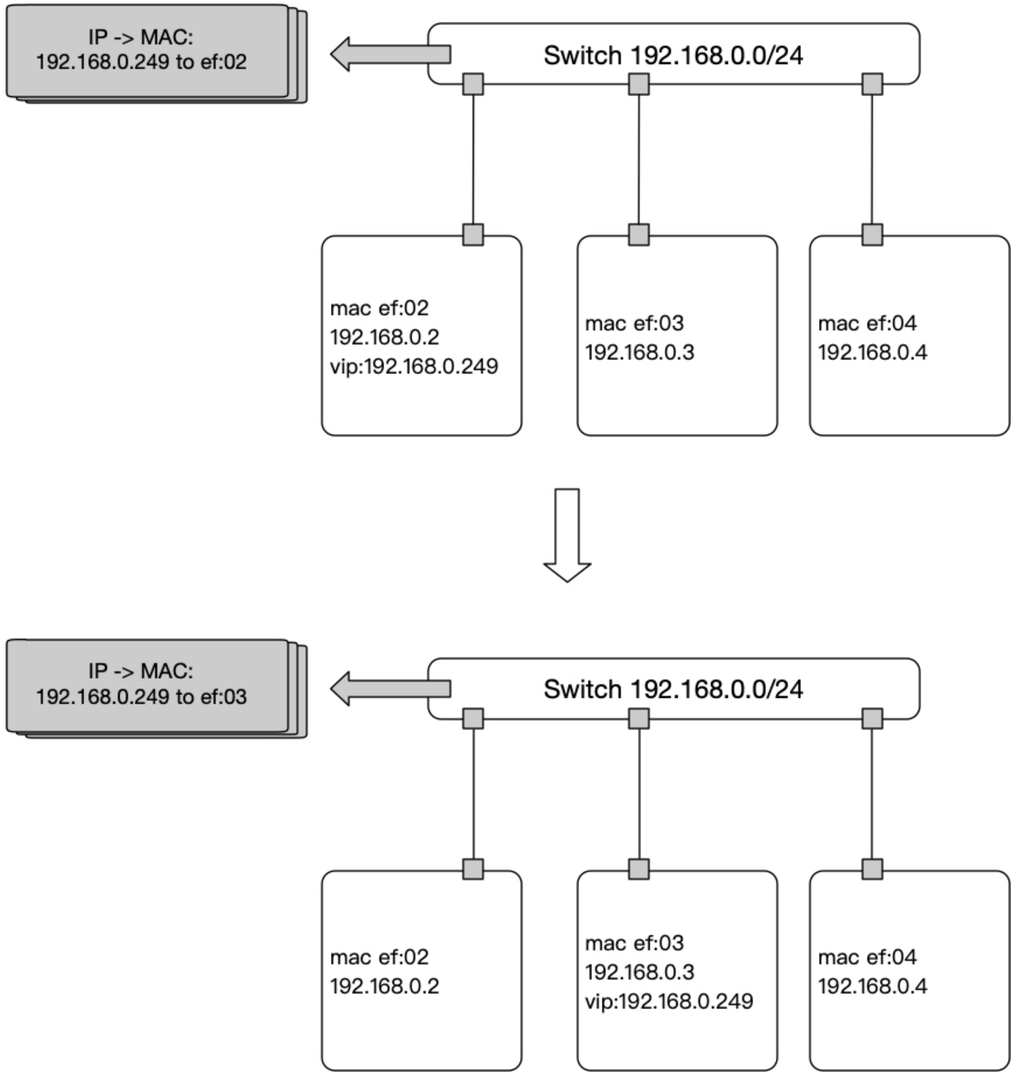
Auth

1. 用户认证
2. 权限鉴别
3. 访问控制
4. license控制

版本控制

1. API版本控制
2. Schema版本控制

4. 微服务 - 高可用原理



可选中间件

1. keepalived
2. pacemaker
3. heartbeat
4. haproxy
5. lvs