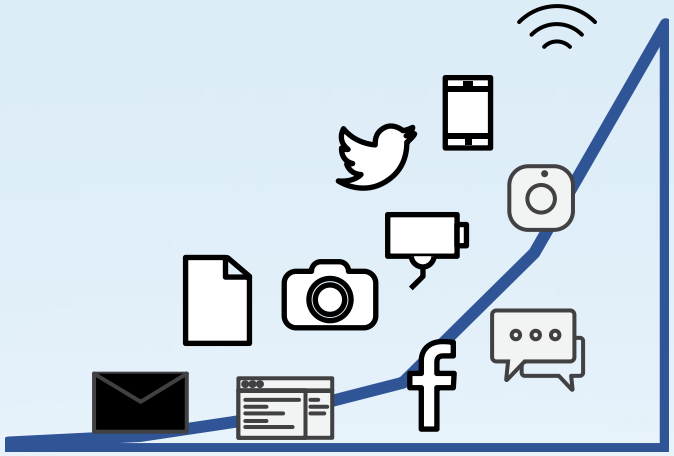
在Kubernetes上运行Apache Spark 进行大规模数据处理的实践

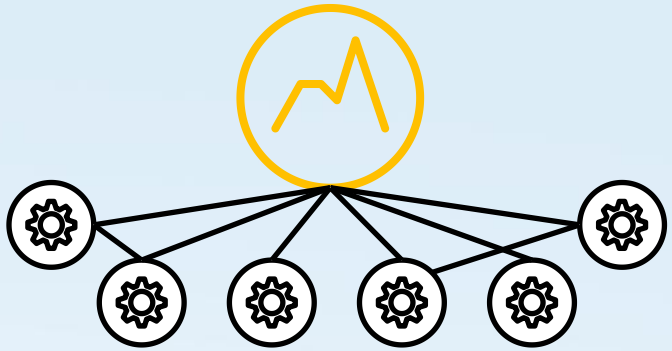
杨冬冬 亚马逊云科技
高级容器专家

Apache Spark



快速增长的数据量

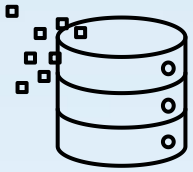
个性化的需求



快速的决策要求近乎实时的数据处理

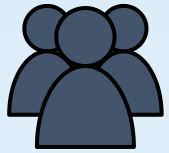
为什么客户选择Kubernetes运行Spark

高扩展性



更好地资源利用率

计算和存储选择丰富

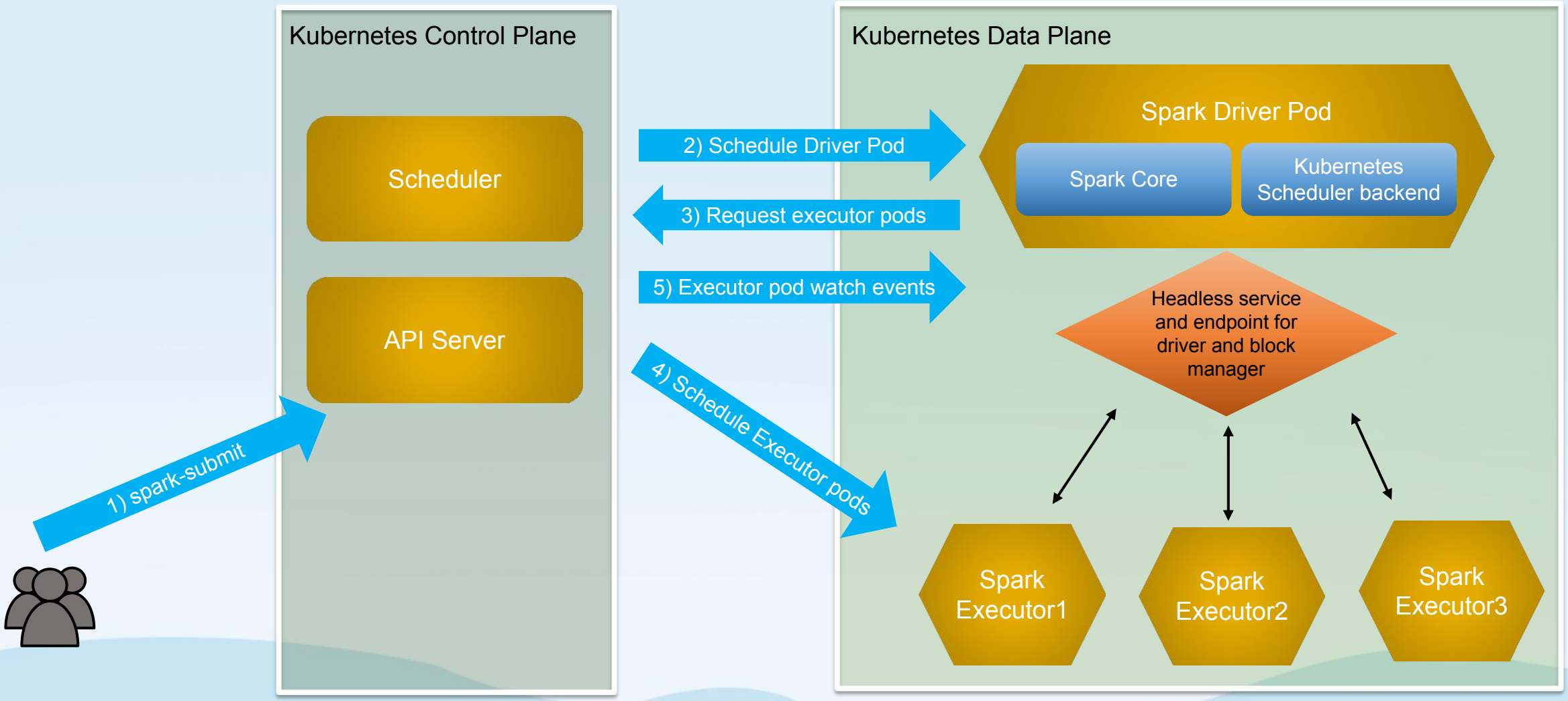


安全和多租户隔离

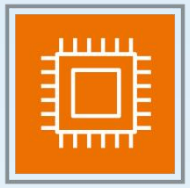
成本性价比

开源社区

Spark on Kubernetes



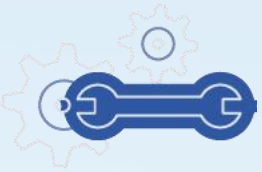
最佳实践



计算与基础设施

弹性扩展

存储



健壮性和灾备

可观测性

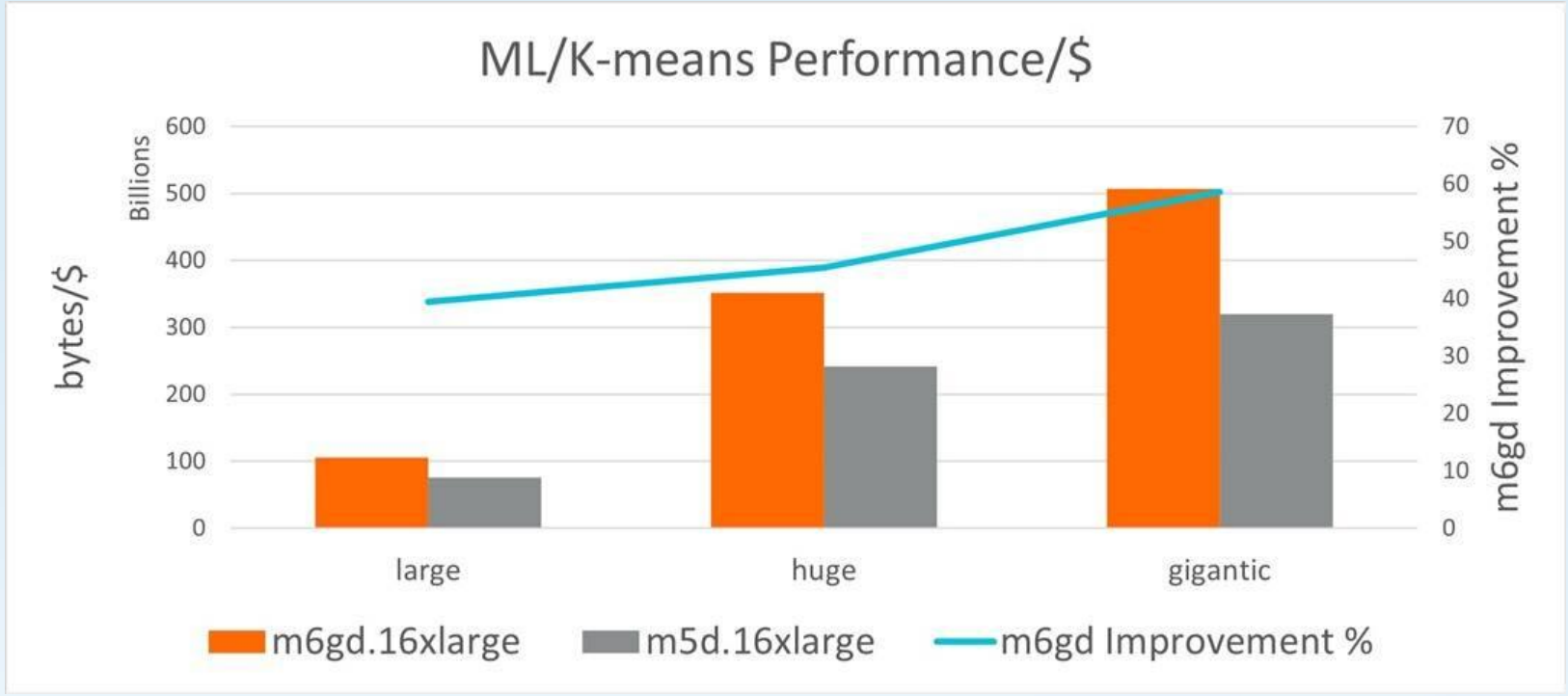
计算与基础设施

尝试采用ARM架构

JDK 11 + Spark

可获得高达58%的成本性价比

采用多架构镜像

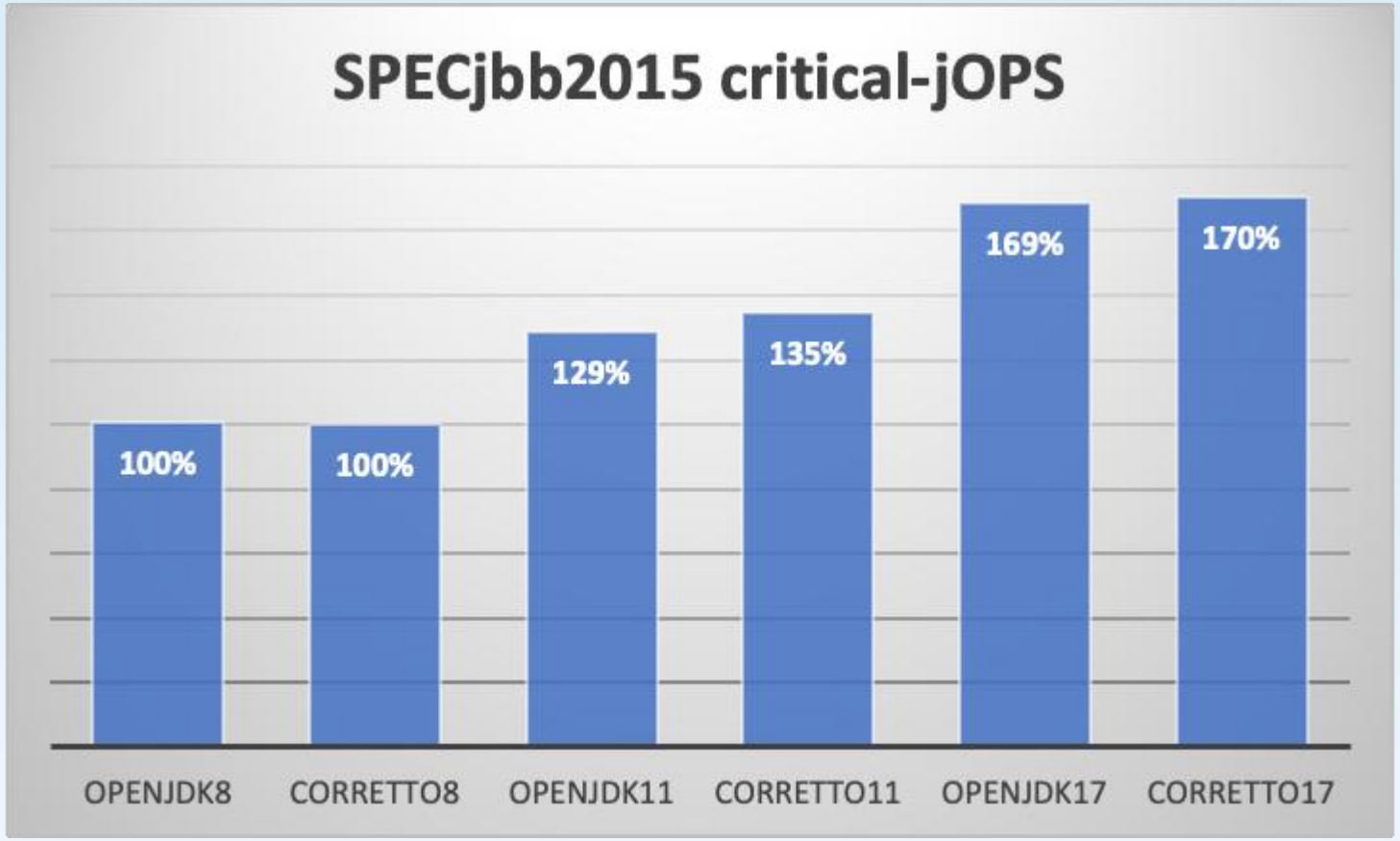


Source: https://community.arm.com/arm-community-blogs/b/infrastructure-solutions-blog/posts/apache-spark-performs-better-on-aws-graviton

不同JDK上的性能表现

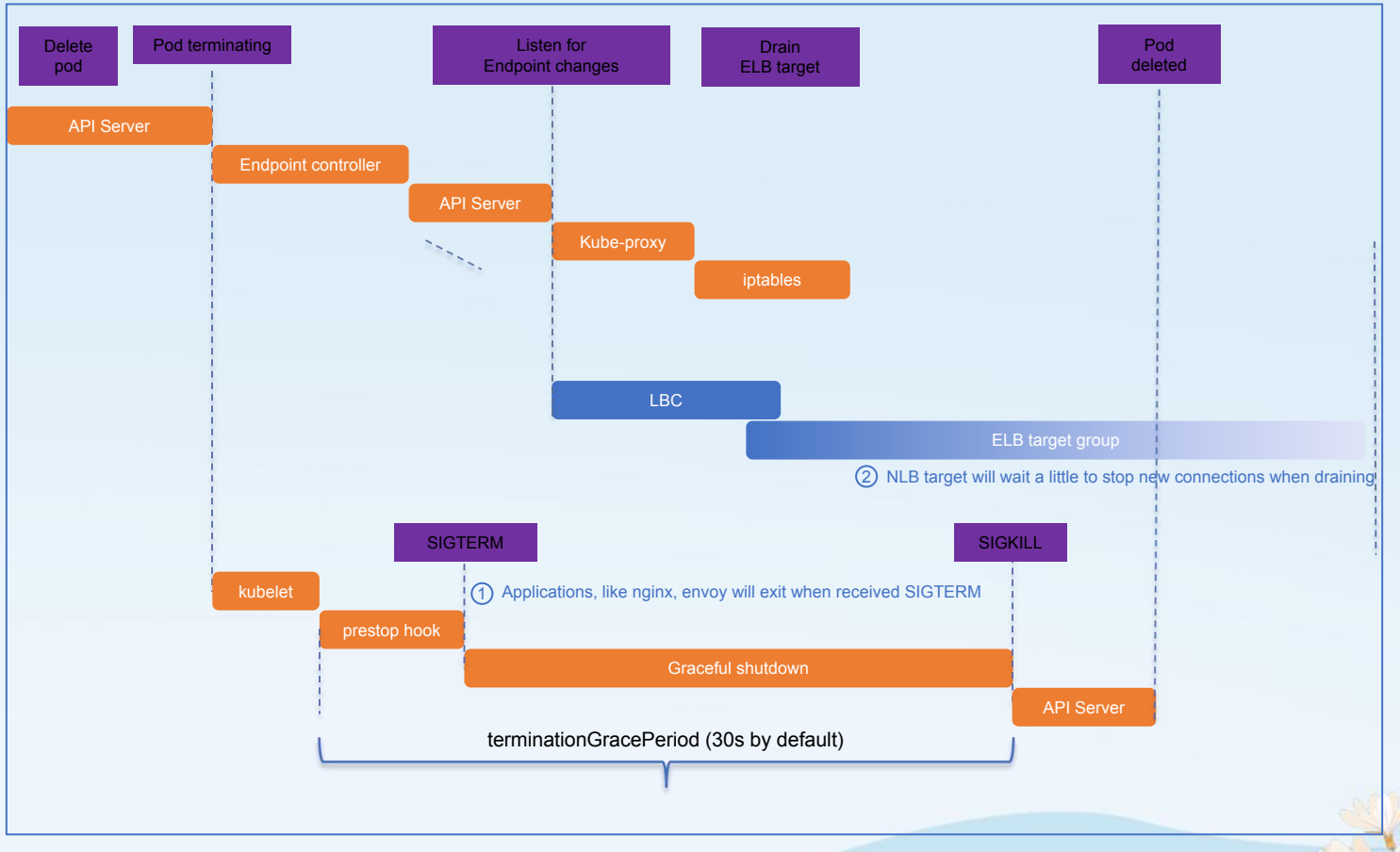
Java Benchmark

Instance Type	m6g.4xlarge
OS	Amazon Linux 2
SPECjbb Version	specjbb2015-1.02



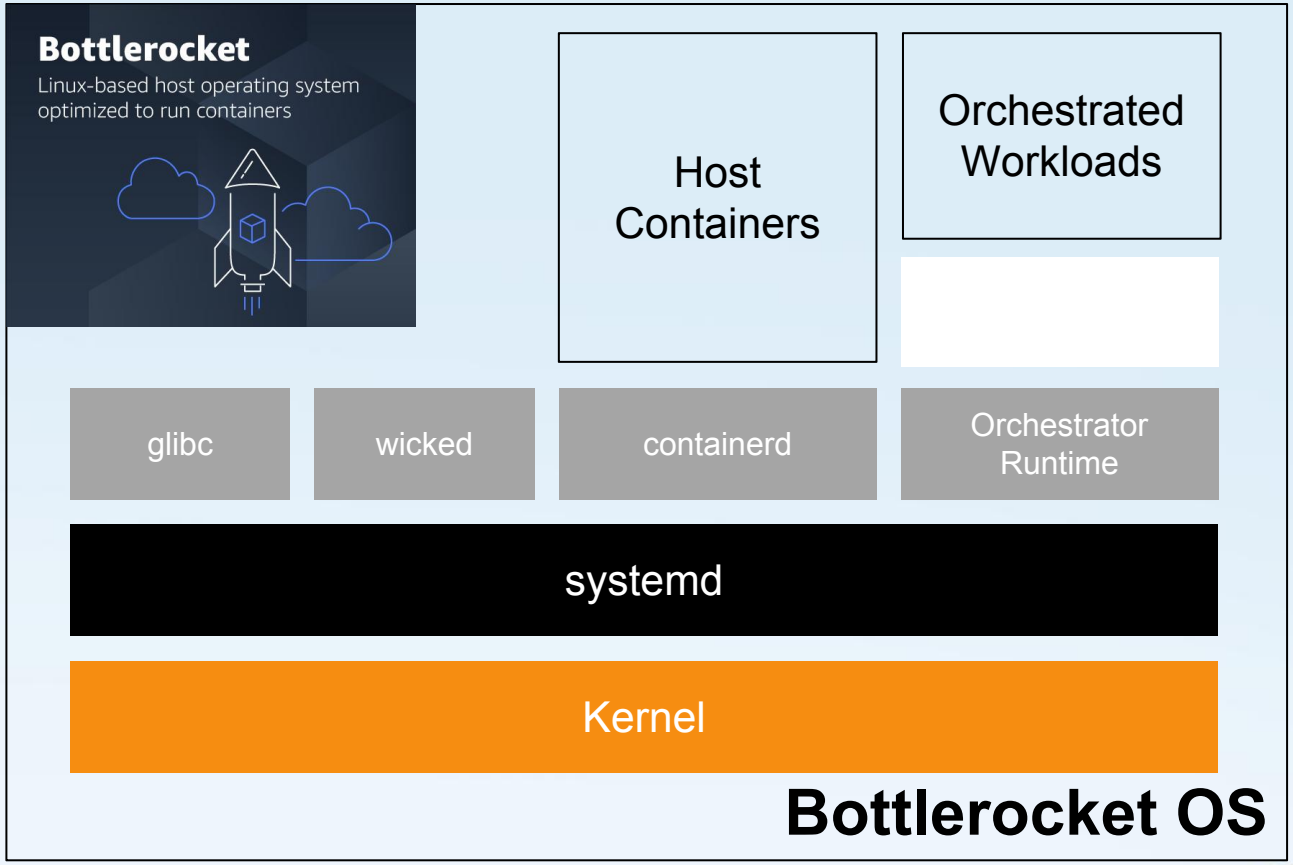
处理Spot中断以及Pod优雅退出

- **Spot**
  - Spot性价比优势高
  - 模拟中断
- **Pod优雅退出**
  - Executors Pod Detach块存储



采用容器化操作系统，降低运维成本和提高敏捷性

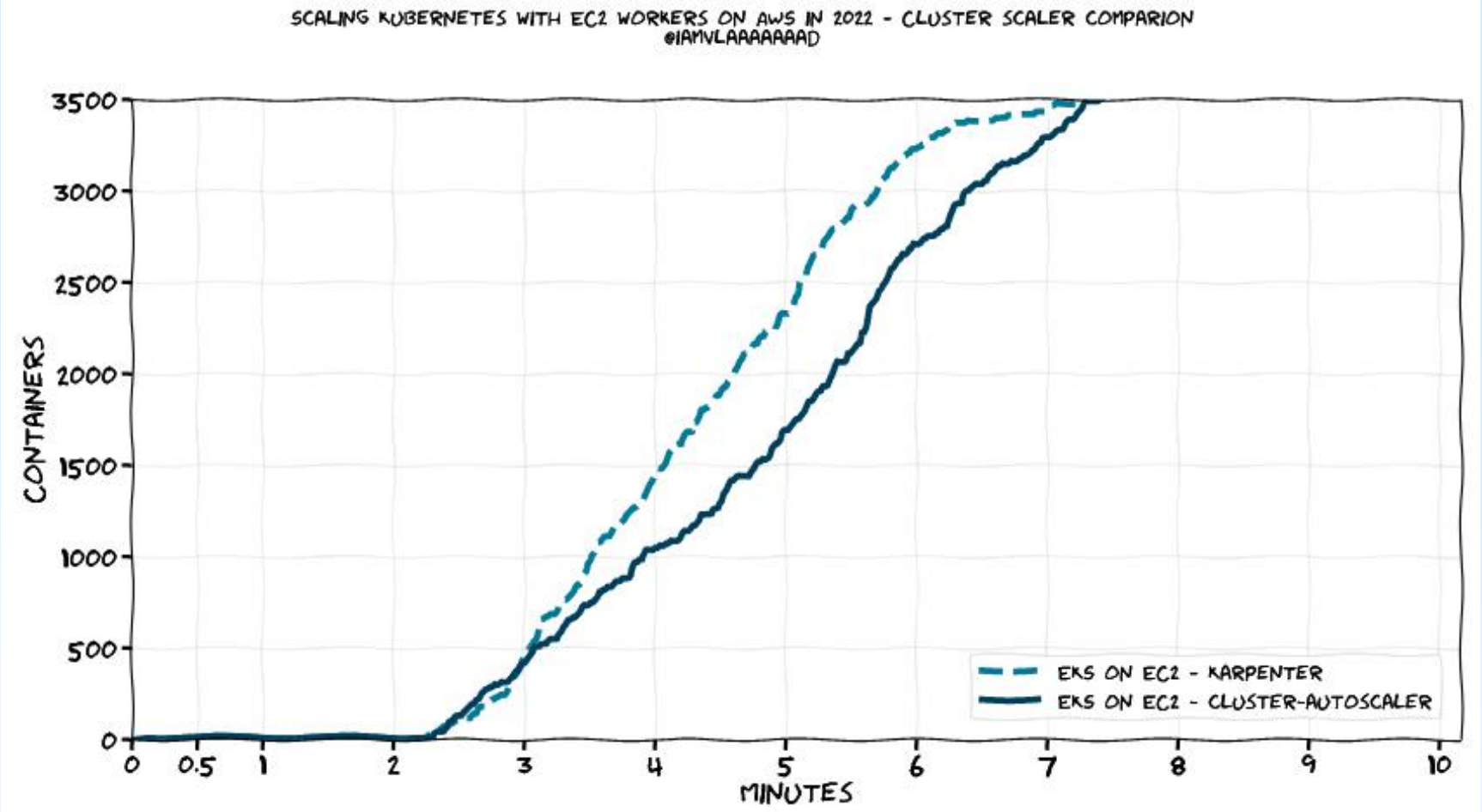
- 最小化
  - 只包括运行容器的包
  - 针对不同workload场景
  - 不可变
- 更新
  - 基于镜像的原子更新
  - 支持回滚和Operator
- 安全
  - SELinux



弹性扩展

快速响应大数据负载

- Spark Pod的特点
  - 数量大
  - 频率高
  - 多个Job同时运行
- Cluster Autoscaler
  - 与节点组/池绑定
  - Namespace/CA
- Karpenter
  - 开源
  - 速度快
  - 为大规模和成本优化而构建



Source: https://www.vladionescu.me/posts/scaling-containers-on-aws-in-2022/

Karpenter 最佳实践

- 机器类型选择顺序：
  - arm spot
  - x86 spot
  - arm on-demand
  - x86 on-demand
- 当遇到Capacity的问题时，Karpenter会自动fallback
- 采用多架构

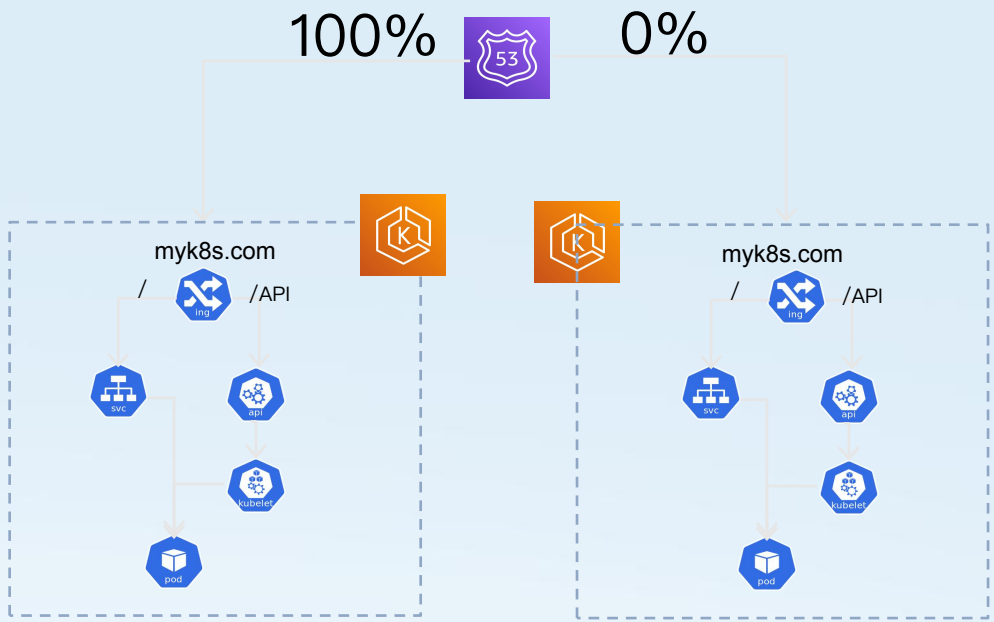
```
- key: karpenter.sh/capacity-type
  operator: In
  values:
    - on-demand
    - spot
- key: kubernetes.io/arch
  operator: In
  values:
    - amd64
    - arm64
```

```
nodeAffinity:
  preferedDuringSchedulingIgnoredDuringExecution:
  - weight: 1
    preference:
      matchExpressions:
        - key: beta.kubernetes.io/arch
          operator: In
          values:
            - arm64
```

健壮性和灾备

构建Spark Clusters

- 多集群
  - for testing and prod purpose
  - EKS Active/Active Clusters
- 使用节点组运行Systems critical pods
- 使用 On-Demand 节点组运行Driver pods (Single AZ)
- 规划VPC(/16 65k IPs)
- 调优CoreDNS



可观测性

SLI/SLO建设的必要性

Status	SLI	SLO
Official	Latency of mutating API calls for single objects for every (resource, verb) pair, measured as 99th percentile over last 5 minutes	In default Kubernetes installation, for every (resource, verb) pair, excluding virtual and aggregated resources and Custom Resource Definitions, 99th percentile per cluster-day <= 1s
Official	Latency of non-streaming read-only API calls for every (resource, scope) pair, measured as 99th percentile over last 5 minutes	In default Kubernetes installation, for every (resource, scope) pair, excluding virtual and aggregated resources and Custom Resource Definitions, 99th percentile per cluster-day (a) <= 1s if scope=resource (b) <= 5s if scope=namespace (c) <= 30s if scope=cluster

Source: https://github.com/kubernetes/community/blob/master/sig-scalability/slos/api_call_latency.md

Spark 可观测性

指标

- Prometheus and Grafana for monitoring
- 开启Prometheus serve的VPA
- 尝试使用RemoteWrite，将Metrics写到托管Prometheus
- Spark history Server for Spark events

日志

- FluentBit for logging
- 使用对象存储导出日志
- FluentBit *extraFilters config*
  - *使用Kubelet 获取元数据而非apiserver*

成本可视化

If you can’t measure it, you can’t manage it.

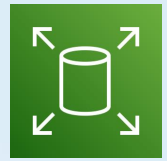
Peter Drucker

合适的Pod规格　　超售比　　**任务执行时间**　Shuffle成本

存储

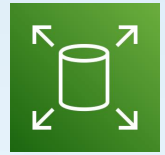
Spark Shuffle 存储选项

当开启DRA以及HA时，结合PVC使用块存储


Amazon EBS

当对性能和吞吐有高的要求时，使用NVMe/SSD

NVMe 可采用RAID0配置

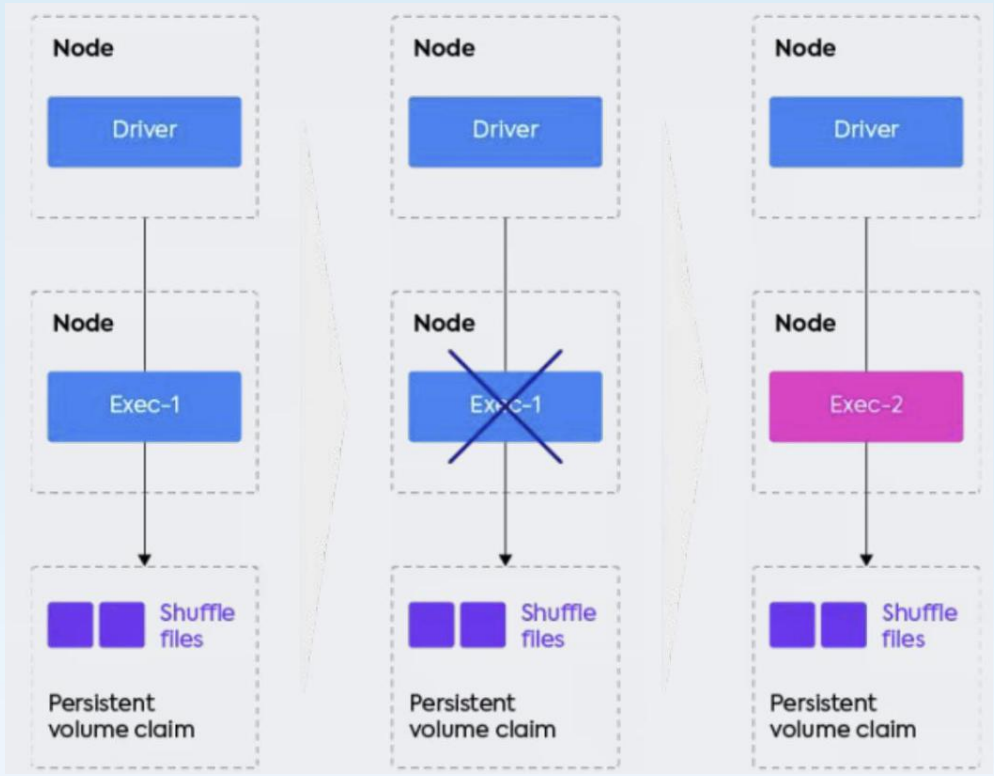

Instance Store


Amazon FSx for Lustre

尝试使用RSS

Spark PVC reuse

- Spark 3.2 [SPARK-35593]
- 降低重算
- 注意处理中断




```
"spark.kubernetes.driver.ownPersistentVolumeClaim": "true"
"spark.kubernetes.driver.reusePersistentVolumeClaim": "true"
```

块设备使用最佳实践

- **当PVC数量较大时:**

当运行大规模的 Spark 作业时，我们需要运行许多 Spark 执行器 Pod，每个执行器 Pod 都需要一个独立的PVC 用于 shuffle 。 有时并行创建或删除速度可能会受到CSI API的限制，我们可以增加CSI的以下值以提高效率：

csi-attacher – --kube-api-qps=100 – --kube-api-burst=200 – --worker-threads=100

- **处理Spot中断**

在Spot中断期间，我们需要确保所有应用程序Pod可以在2分钟内迁移到另一个工作节点。 PVC 的绑定信息也可能保留在原始节点上，并且需要几分钟才能释放。

Thanks.