

azul

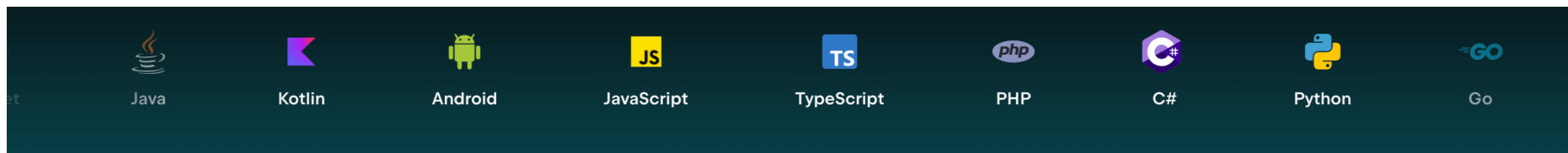
Optimizer Hub 技术与实践

CNC云原生编译器简介

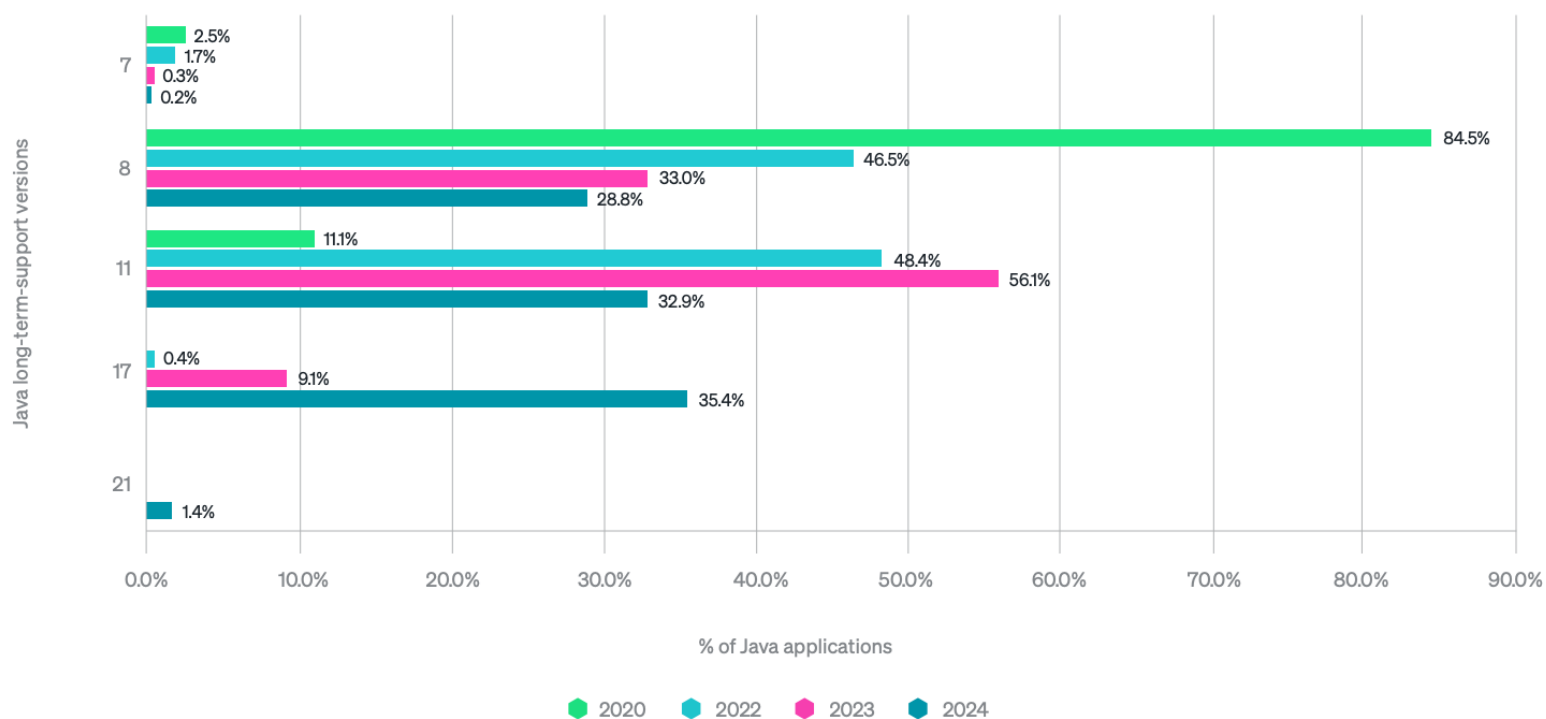
Gavin Yu | July 6, 2024

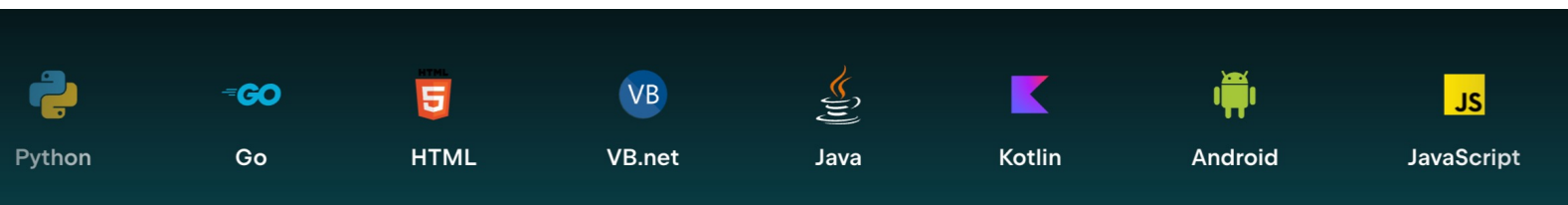


- 28 years of Java
- 25 years of JCP Program
- Java continues to be an **important choice** for enterprises and developers
- The Java technology **innovation pipeline** has never been richer
- Java 21 available now

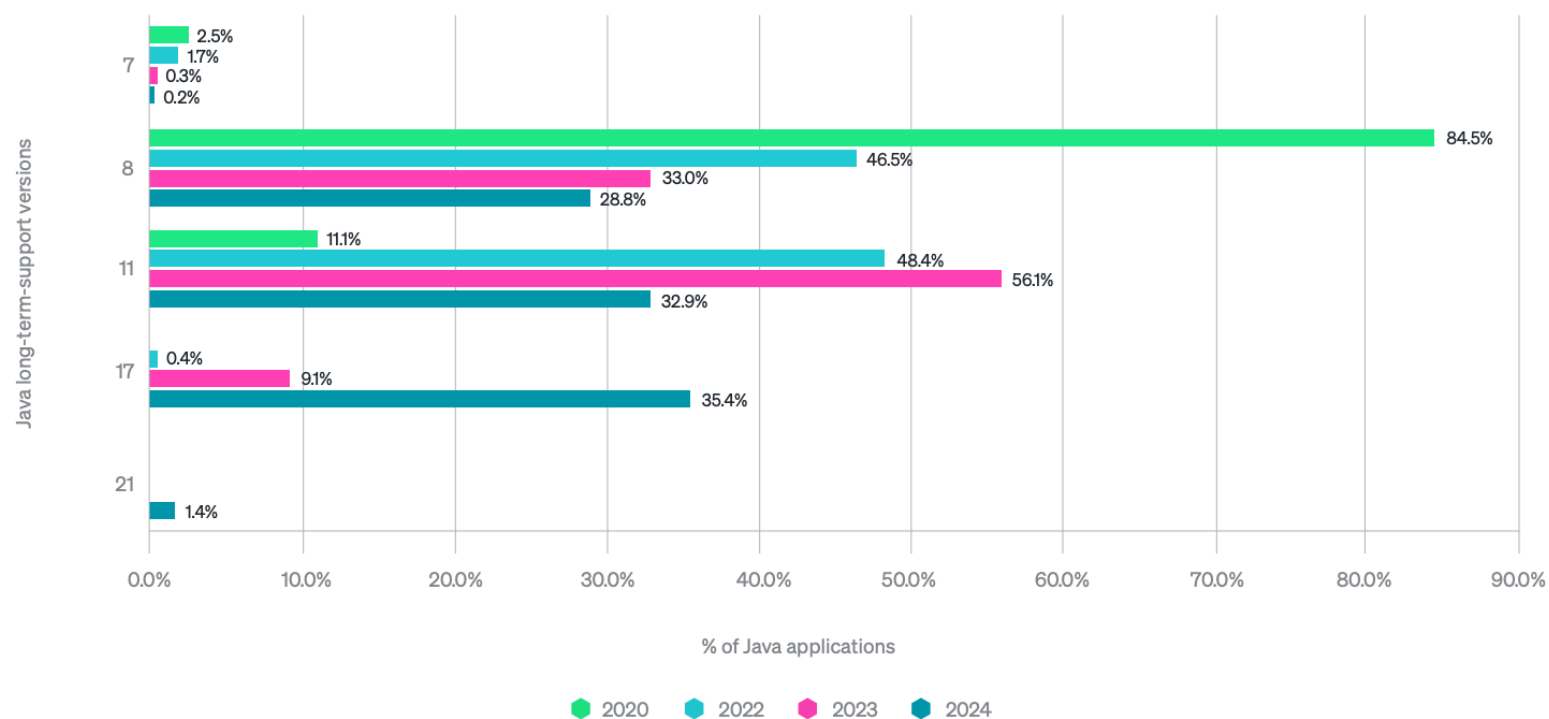


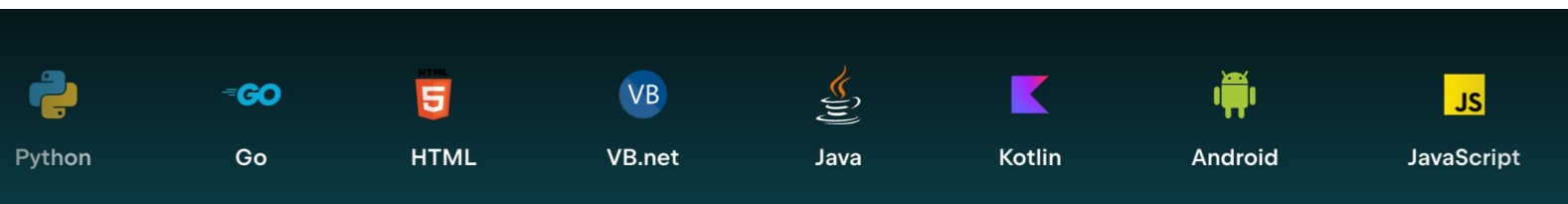
New Relic 05.29 最新发布的《2024 年 Java 生态系统现状》





New Relic 05.29 最新发布的《2024 年 Java 生态系统现状》





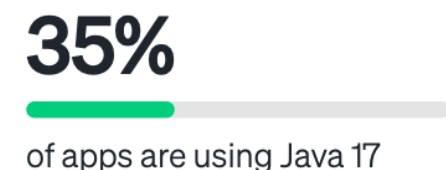
New Relic 05.29 最新发布的《2024 年 Java 生态系统现状》

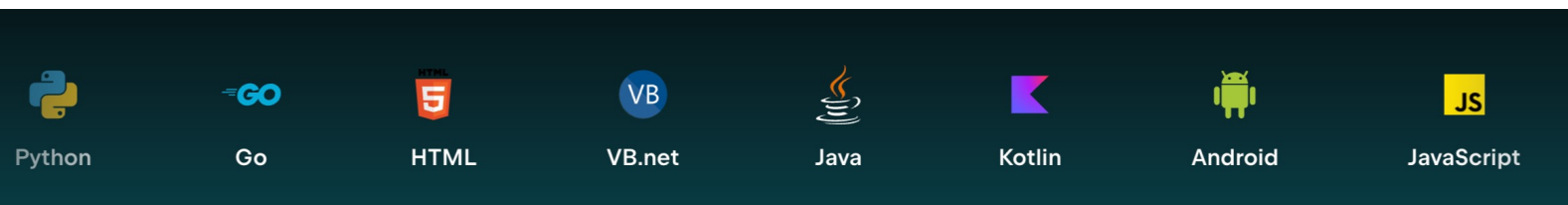
- **Java 21 sees higher adoption than Java 17:** Following the release of Java 21 in September 2023, the report found that the adoption rate for the latest Java LTS version was 287% higher in the first six months after release than its previous LTS counterpart, Java 17.

In the six months after the release of Java 21, 1.4% of applications monitored by New Relic were using it. To put this into perspective, in the six months after Java 17 was introduced, only 0.37% of applications were using it, which is 287% fewer.

In addition, the adoption rate of Java 17 far exceeded what the developer world saw when Java 11 was introduced. About a tenth (9%) of applications were using Java 17 in production in 2023, and now 35% of applications are using Java 17, representing a nearly 300% growth rate in one year. It took years for Java 11 to reach anywhere near that level.

Less than 2% of applications were using Java non-LTS versions, which makes sense since they're usually not used in production.



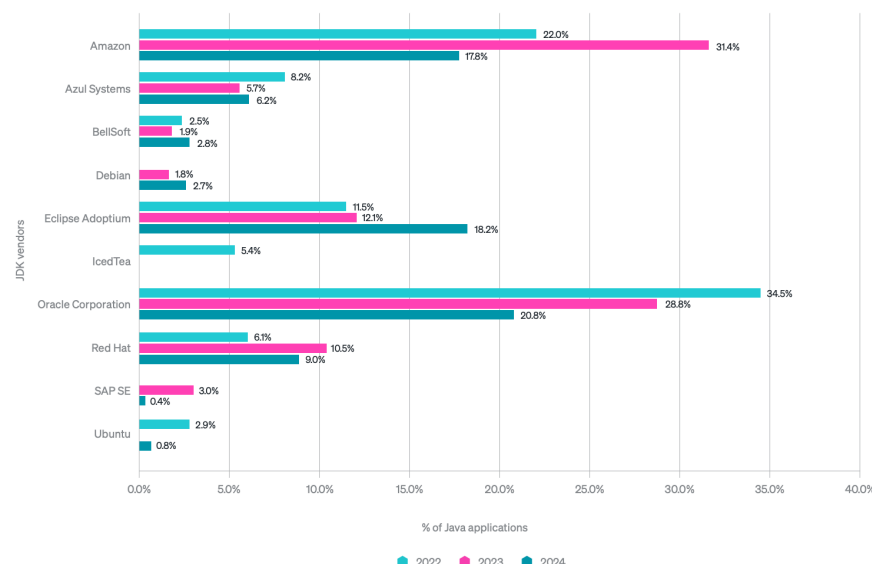


New Relic 05.29 最新发布的《2024 年 Java 生态系统现状》

The use of Amazon increased to 31% of the market in 2023 (up from 2.2% in 2020 and 22% in 2022), but has dropped to 18% in 2024, which represents a 43% decrease YoY.

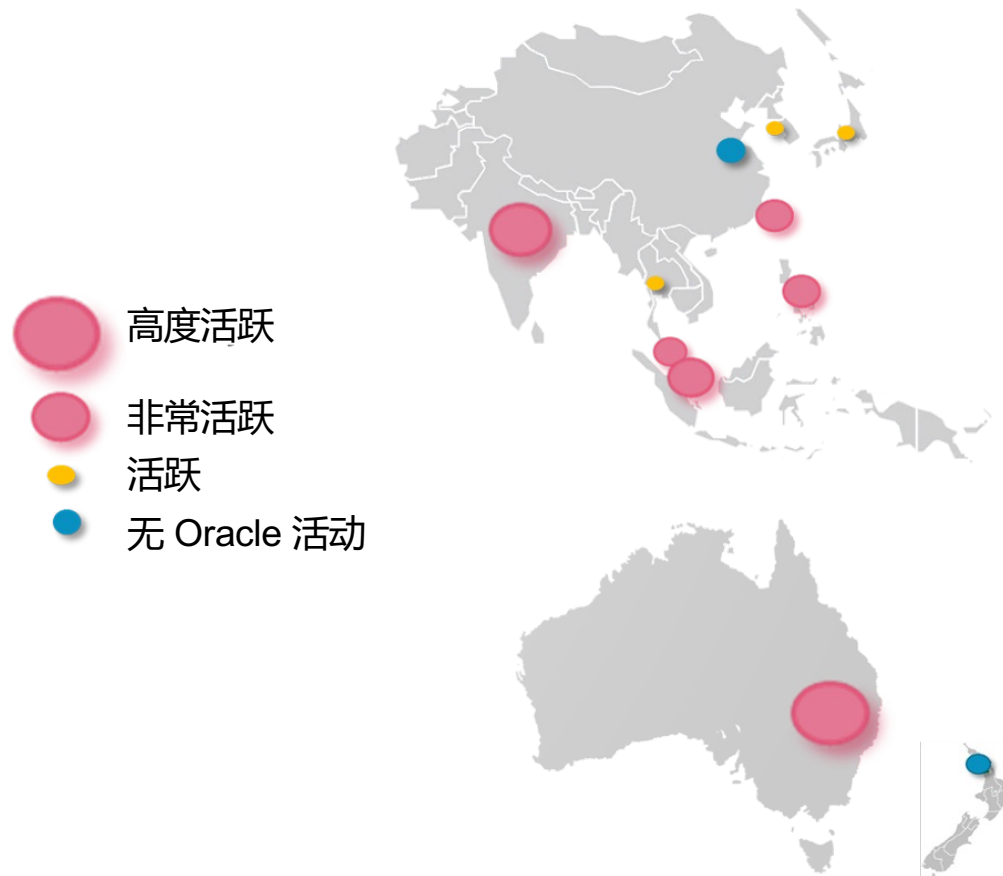
The rising star this year is Eclipse Adoptium, adoption of which rose 50% YoY from 12% to 18%. Because Eclipse Adoptium is community-managed, this JDK tends to be updated more frequently than the Oracle and Amazon JDKs.

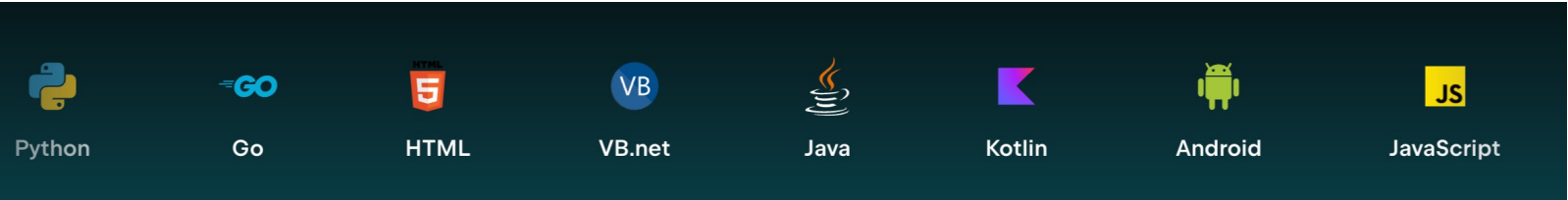
18%
are using the Eclipse
Adoptium JDK



The most popular JDK vendors by year

APAC Oracle 热图



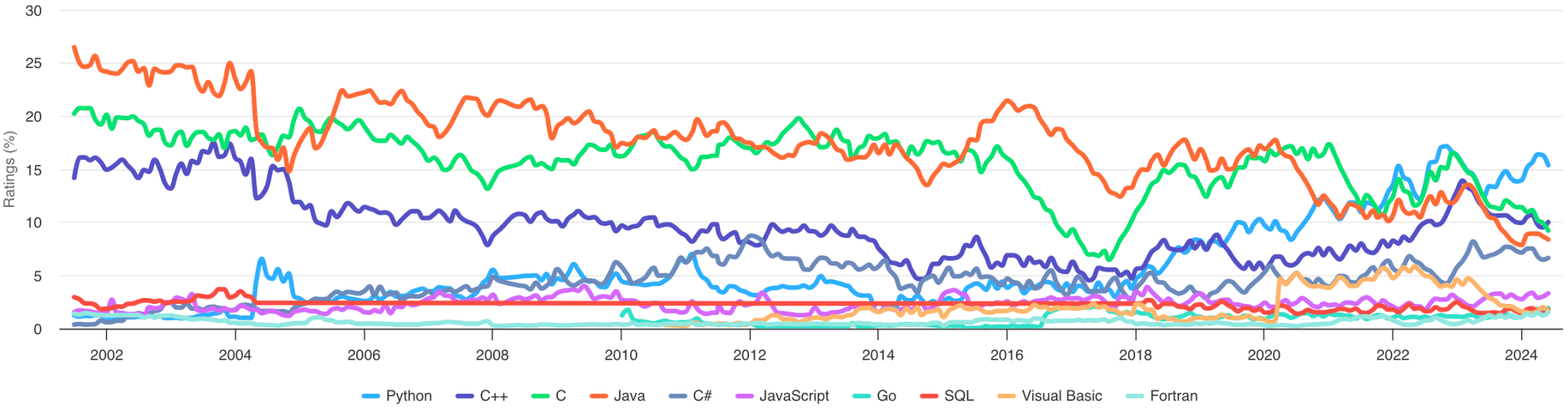


TIOBE 6月榜单

Jun 2024	Jun 2023	Change	Programming Language	Ratings	Change
----------	----------	--------	----------------------	---------	--------

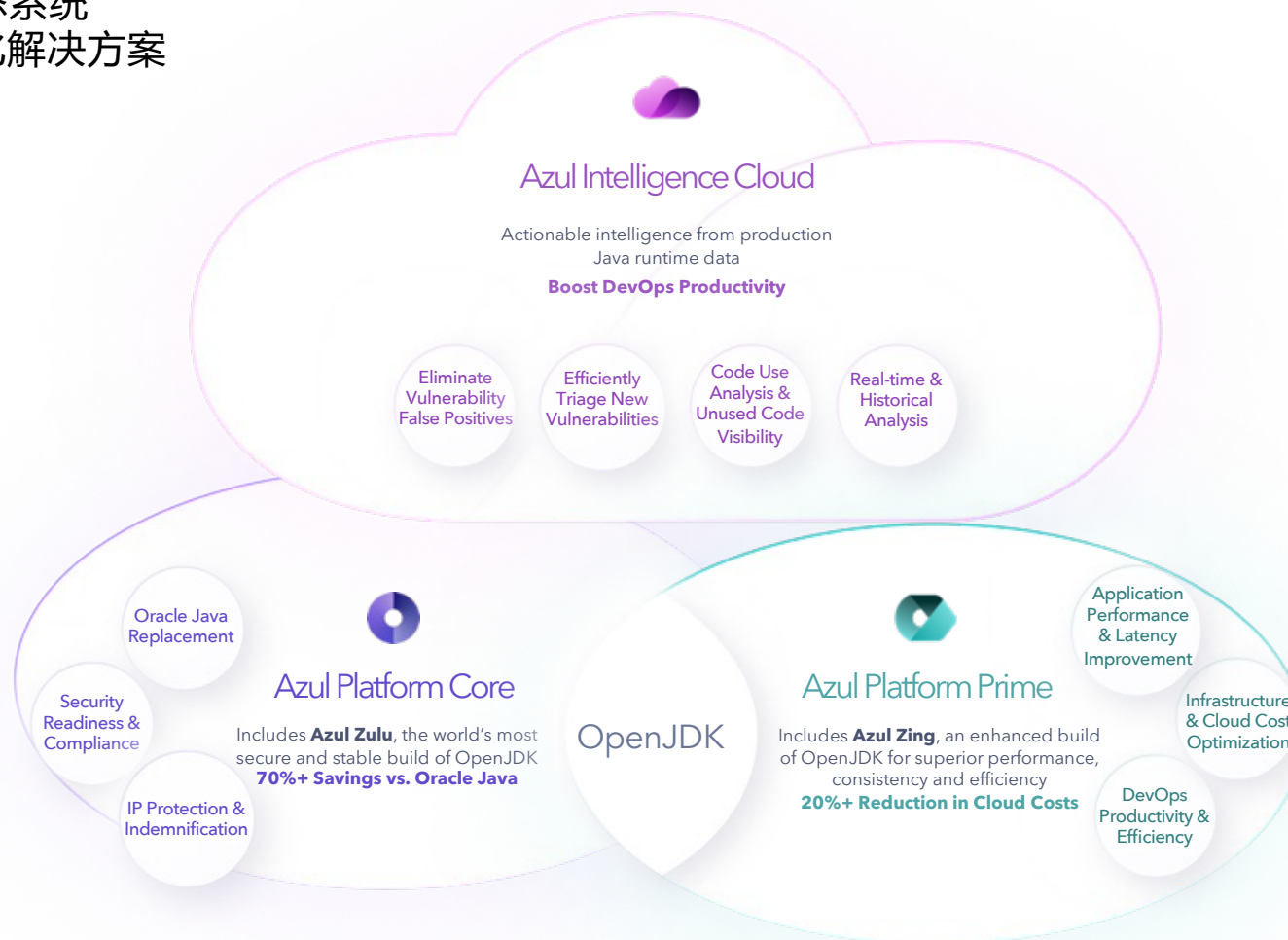
TIOBE Programming Community Index

Source: www.tiobe.com

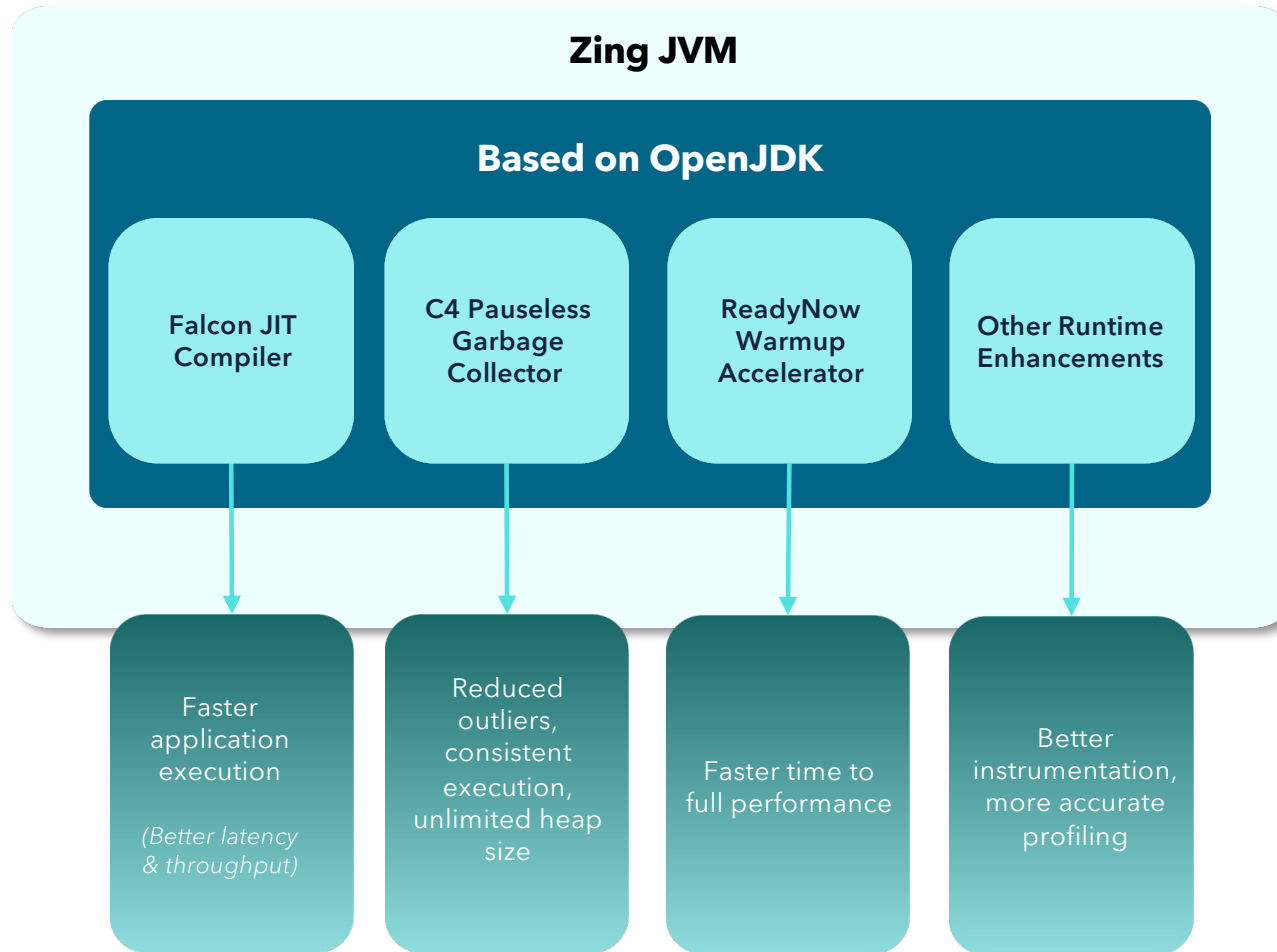


The Azul Platform

Java生态系统
全景优化解决方案

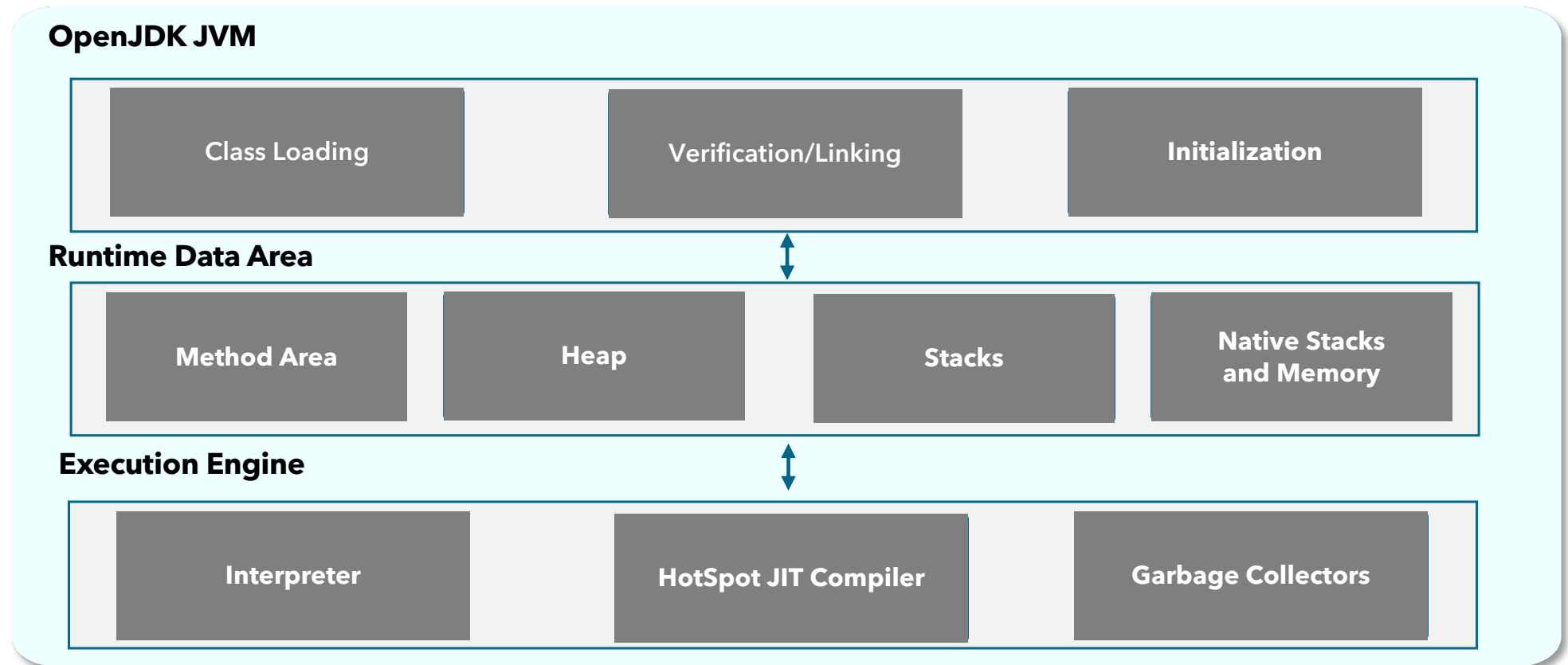


Azul Zing Builds of OpenJDK



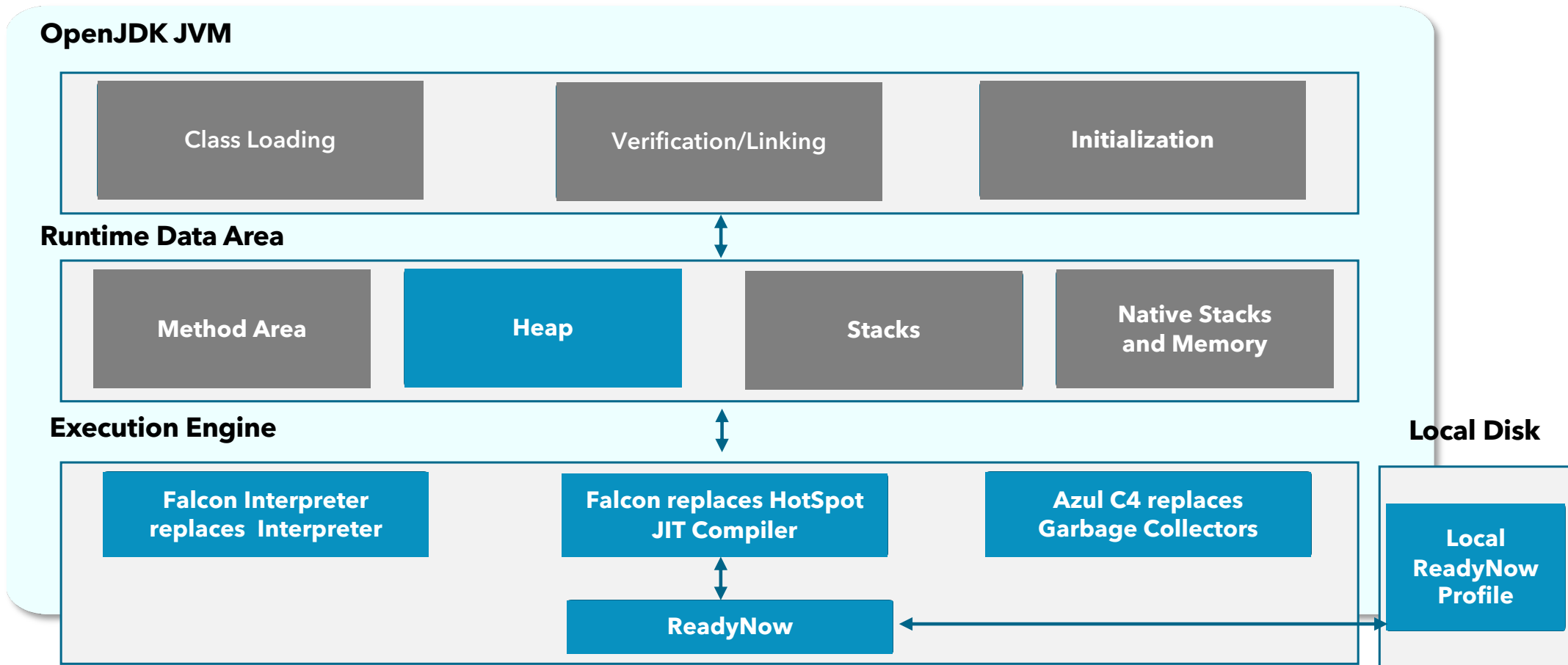
- We start with OpenJDK as the base
- Then we replace key components with our optimized versions
- Taken together, these deliver the best & most consistent performance at the lowest operating cost of any JVM on the planet

How OpenJDK Works



How Azul Zing OpenJDK Works

**Azul Platform Prime
Optimizations**



► 内容概述

1. Optimizer Hub 前世CNC
2. Optimizer Hub 今生CNC+RNO
3. 云成本优化的新武器
(Java Cloud FinOps: 百万核的降本优化之路)
4. AI时代的Java技术展望与FinOps方案总结

▶ Cloud FinOps的新武器- Azul Optimizer Hub

使用 Cloud Native Compiler 演进为云原生 JVM

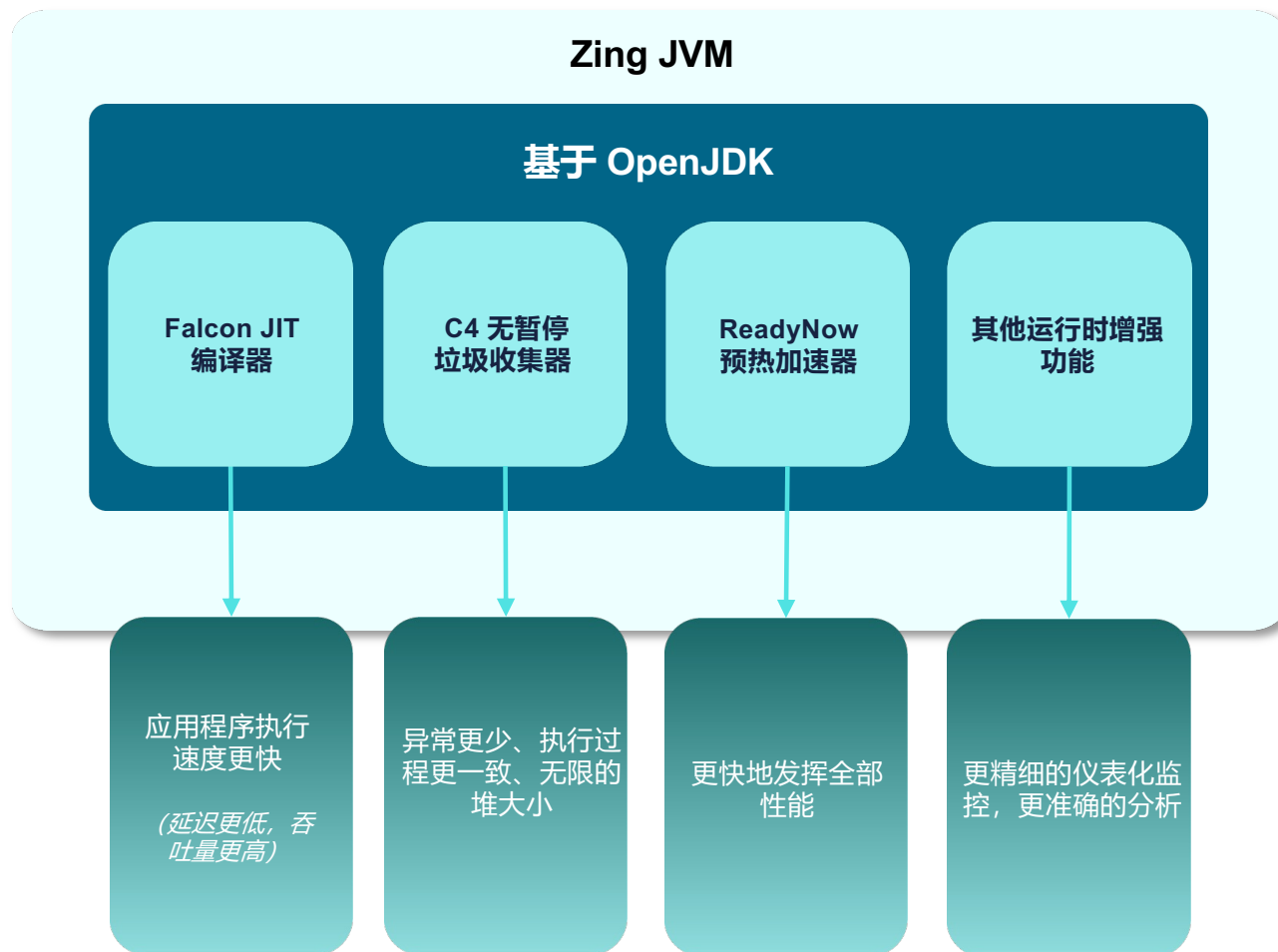
没有
云原生编译器
更大、更慢和更多的实例



使用
云原生编译器
更小、更少的实例和更好的性能



Azul Zing 版本的 OpenJDK



- 我们以 OpenJDK 为基础开始
- 然后将关键组件替换为我们的优化版本

Azul Platform Prime 价值主张

更出色的性能和
客户体验

更快的预热

更低的延迟

其他
运行时增强功能

更低的基础设施成本
和总拥有成本

更少的
计算实例

更平稳的弹性



性能技术应用场景

更快预热	更低的整体延迟	更一致的执行过程
• 动态扩展以满足负载需求	• 更短的响应时间将对业务产生重大影响	• 通常伴随着更低的整体延迟，即使总体延迟并非关键因素，但长时间的暂停会造成业务中断
• 延迟对于游戏和电子商务等业务至关重要	• 面向客户的应用程序或关键业务流程中的瓶颈	• 对延迟峰值极其敏感

20%

CPU 占用率降低 20%，
提升承载能力，
减少玩家遇到的延迟



1 万

每天消除 1 万次
垃圾回收暂停，
在高业务量下，
将欺诈检测加速到微秒级



50 万

广告服务器平台
每秒生成 50 万条推荐，
前端服务器
占用空间减少 30%

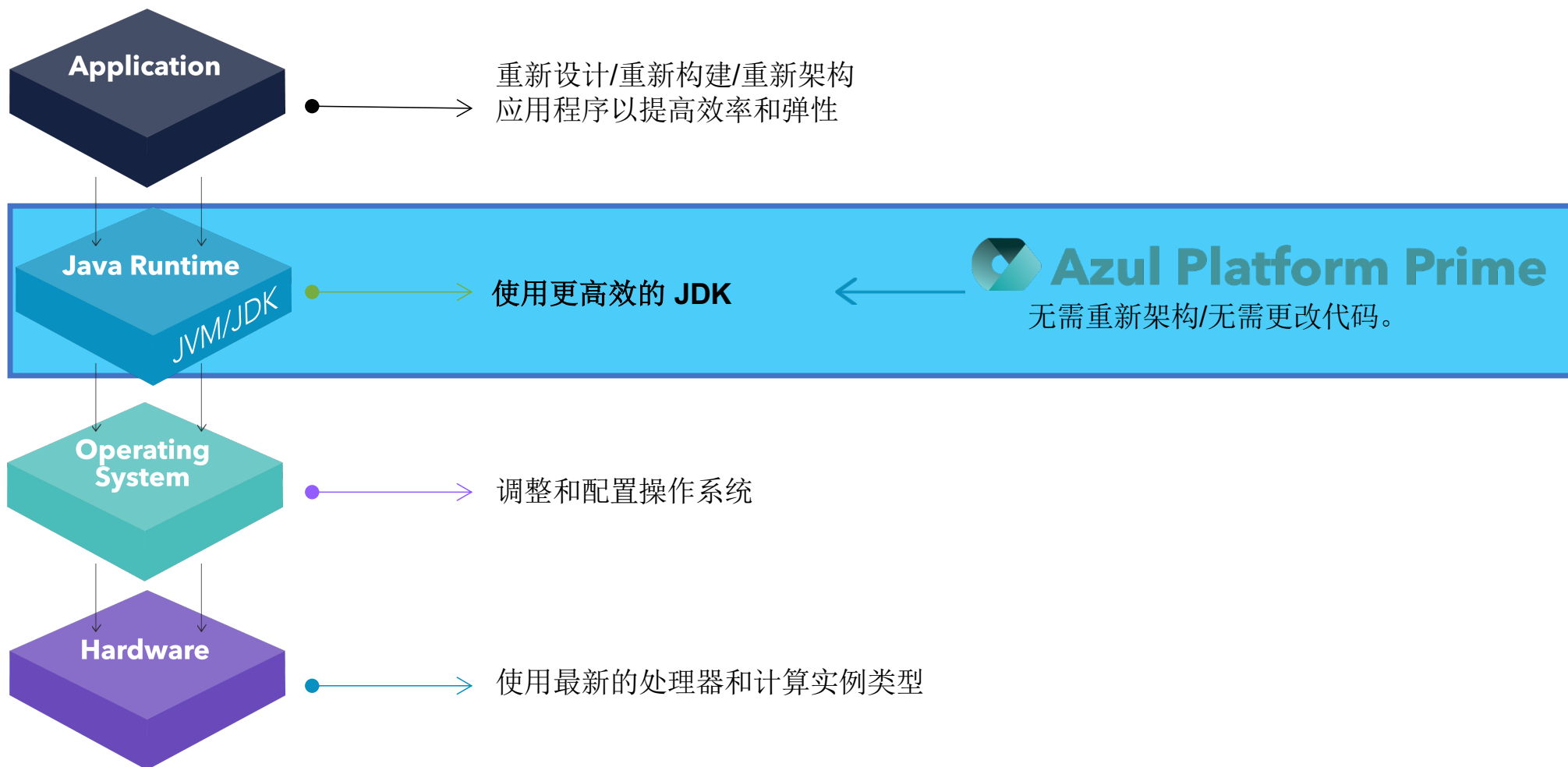




成本优化技术应用场景

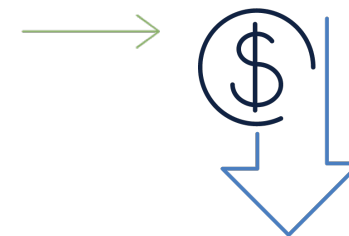
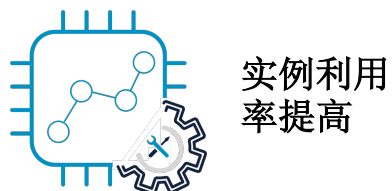
计算实例更少	弹性更平稳
大量集中的 vCore 运行相同代码	
- 以相同方式运行的应用程序：大型电子商务应用程序拆分为多个微服务 - 所有微服务采用相似的技术（如 SpringBoot）；相同的部署模式（具有相似自动扩展规则的 Kubernetes）；以及相同的实例大小（标准 4 vCore K8s 容器）	
- 至少有数十个实例运行相同的代码 - 通过减少实例总数来节省资金 - 我们在一个实例上运行服务，并将镜像大小减半	
- 目标是削减 20% 的基础设施，实现 200% 的投资回报率 - 集群总规模需要足够大，让 20% 的削减具有吸引力	
- 完全不进行自动扩展的客户 - 为了避免启动和停止 JVM 的痛苦，在非高峰时段为浪费的容量付费 - 客户可能会因为其他原因而不进行自动扩展	
既定的云成本优化/FinOps 实践和目标	
能够为所有团队确定方向的强大中央架构角色 - 云卓越中心 (CCoE)	

使用高性能 JVM 减少云计算实例





使用 Prime 降低基础设施成本



 Azul Platform Prime

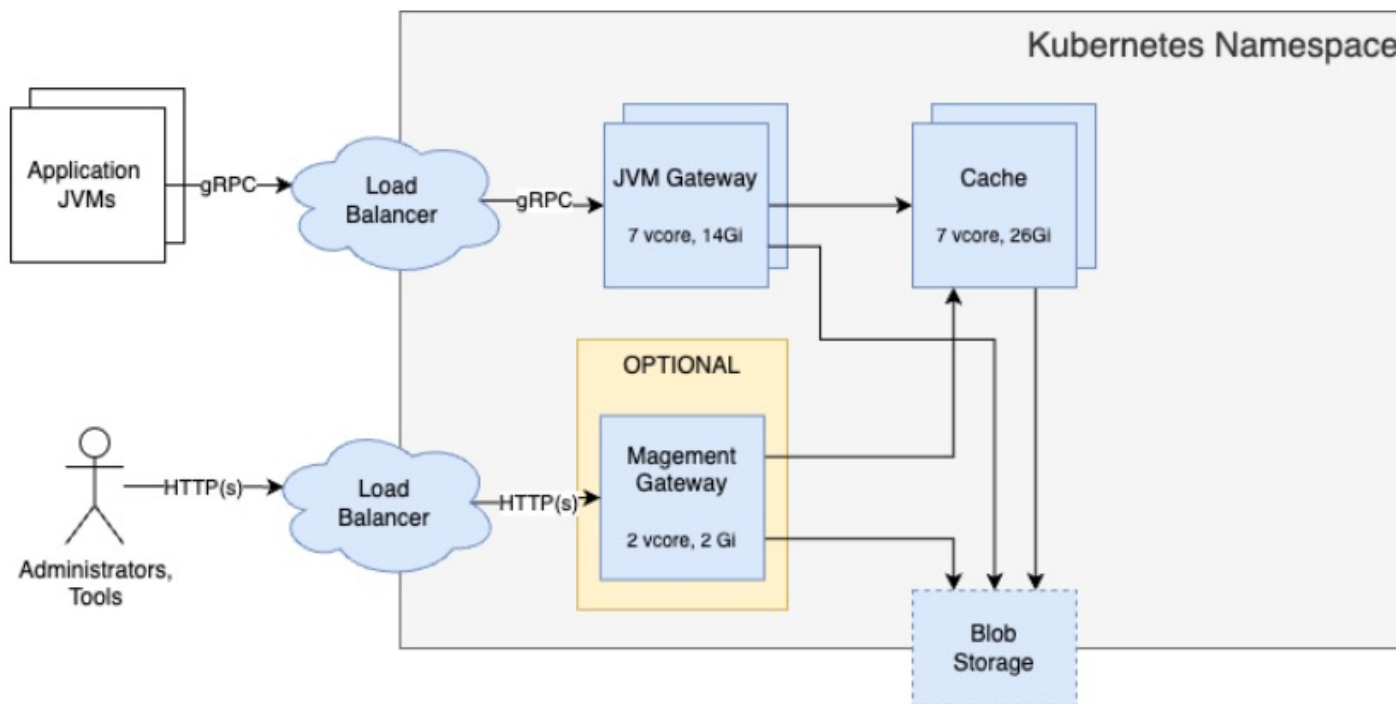


Azul Platform Prime 行业应用场景

数字化工作负载 | 关键业务工作负载 | 需要 SLA 的工作负载

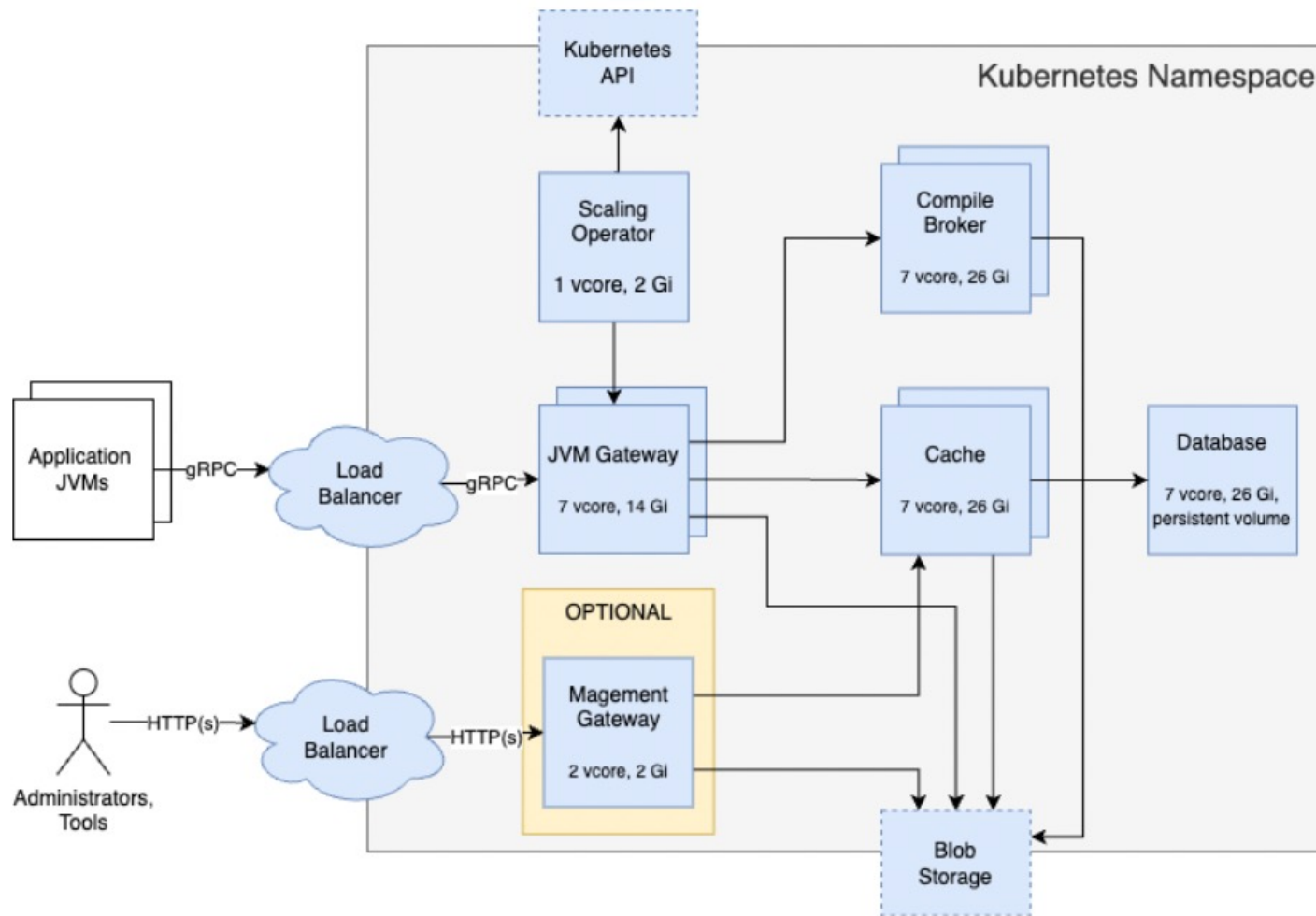
- **SaaS:** 使用更快的交互改善客户体验
- **大数据:** 搜索海量数据集, 获取即时数据或定价信息
- **广告技术:** 实时内容和广告投放建议
- **金融科技:** 实时授权、欺诈检测, 以及更快的交易速度
- **游戏:** 改善玩家体验, 增加游戏内收入, 缩短上市时间
- **交易:** 在开盘时达到性能巅峰
- **数字和电子商务:** 面对客户流量波动, 保持稳定性

➤ 云原生RNO – ReadyNow Orchestrator Only架构图



解决容器预热启动慢
解决结点重复编译

► 云原生CNC – Azul Cloud Native Compiler架构图



降本

避免节点重复编译
全部卸载到远端编译
消除CPU启动风暴
极大缩短启动时间
将集群启动降至最低开销

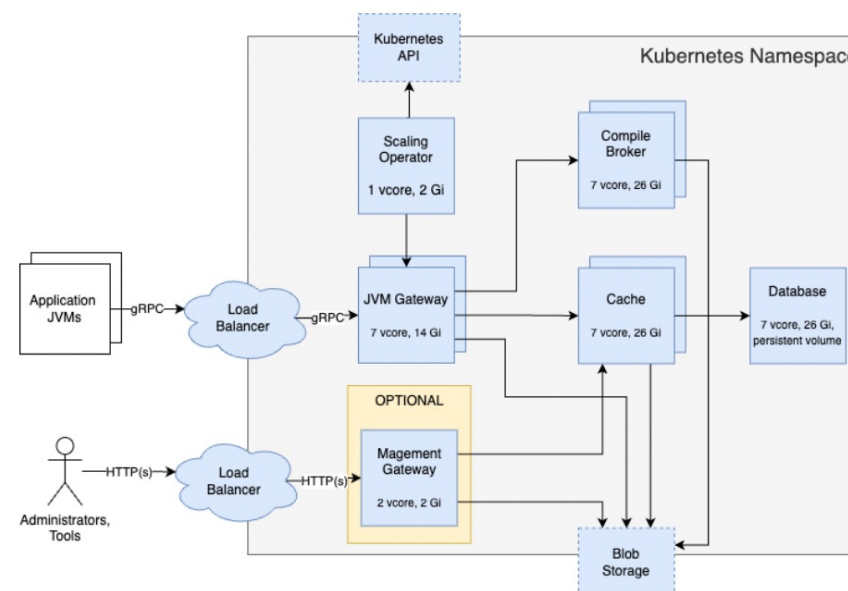
1. 支持不同架构编译请求
2. 支持不同优化编译级别
3. 消除动态代码加载与编译带来的业务RT抖动
4. 对行级代码修改，进行快速性能验证的敏捷能力

➤ Optimizer Hub Component

With the default AWS setup (`values-aws.yaml`), the setup is divided into three node types (four if you also want to use the optional monitoring stack). Each node has a `role` label used to set the affinity for the nodes. If you set up your cluster on AWS EKS using the Azul-provided [cluster config file](#), nodes are created with these labels.

The nodes in a Optimizer Hub instance are as follows:

- Compile Broker - Performs JIT compilations.
 - AWS node type: `role=opthubserver`
 - System Requirements: CPU 8, RAM 32GB, HDD 100GB
- Cache - Stores information about the JVM that the compiler needs to perform compilations.
 - AWS node type: `role=opthubcache`
 - System Requirements: CPU 8, RAM 32GB, HDD 100GB
 - There is one pod per Cache node. To scale up, create more replicas.
- Infrastructure - Provides supporting functionality.
 - AWS node type: `role=opthubinfra`
 - System Requirements: CPU 8, RAM 32GB, HDD 100GB. Make sure the disk connection is fast (use SSD) and that the storage volume is persistent between runs.
 - The pods included in this node are:
 - db
 - gateway
 - storage
- Infrastructure - Non-Optimizer Hub supporting functionality, such as monitoring.
 - AWS node type: `role=infra`
 - System Requirements: CPU 8, RAM 32GB, HDD 100GB.
 - Pods included in this node:
 - grafana
 - prometheus



► Install Optimizer Hub – Deployment on Kubernetes

1. [Install Azul Zulu Prime Builds of OpenJDK](#) 21.09.1.0 or newer on your client machine.
2. Make sure your Helm version is `v3.8.0` or newer.
3. Add the Azul Helm repository to your Helm environment:

```
helm repo add opthub-helm https://azulsystems.github.io/opthub-helm-charts/  
helm repo update
```

4. Create a namespace (i.e. `my-opthub`) for Optimizer Hub.

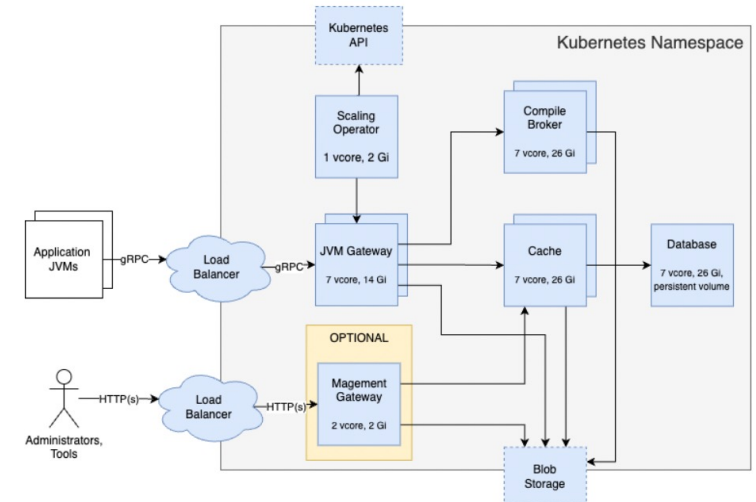
```
kubectl create namespace my-opthub
```

5. Create the `values-override.yaml` file in your local directory.
6. If you have a custom cluster domain name, specify it in `values-override.yaml`:

```
clusterName: "example.org"
```

7. Configure sizing and autoscaling of the Optimizer Hub components according to the [sizing guide](#). By default, autoscaling is on and Optimizer Hub can scale up to 10 Compile Brokers. For example, you could set the following:

```
simpleSizing:  
  vCores: 32  
  minVCores: 32  
  maxVCores: 106
```



► Install Optimizer Hub – Deployment on Kubernetes

8. If needed, configure external access in your cluster. If your JVMs are running within the same cluster as Optimizer Hub, you can ignore this step. Otherwise, it is necessary to configure an external load balancer in `values-override.yaml`.

For clusters running on AWS [an example configuration file is available on Azul's GitHub](#).

9. Install using Helm, passing in the `values-override.yaml`.

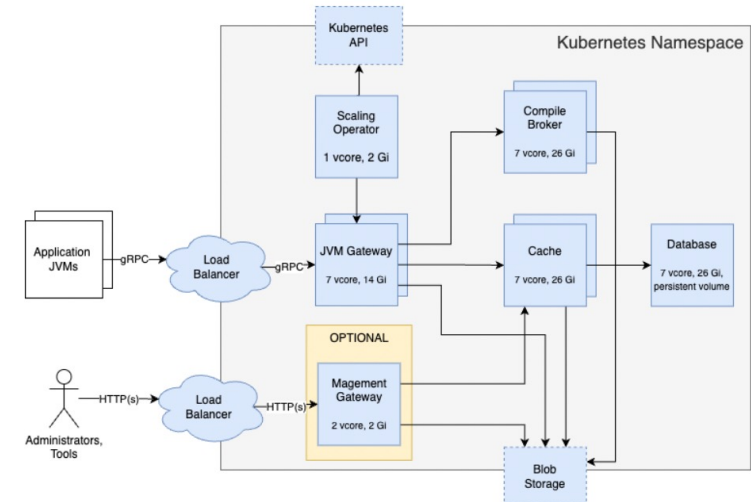
```
helm install ophub ophub-helm/azul-ophub -n my-ophub -f values-override.yaml
```

- In case you need a specific Optimizer Hub version, please use `--version 1.9.2` flag. The command should produce output similar to this:

```
NAME: ophub
LAST DEPLOYED: Wed Jan 31 12:19:58 2024
NAMESPACE: my-ophub
STATUS: deployed
REVISION: 1
TEST SUITE: None
```

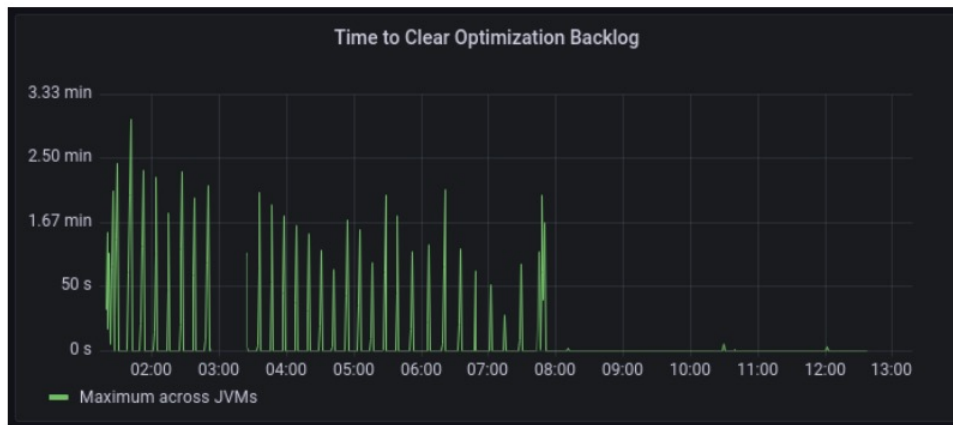
10. Verify that all started pods are ready:

```
kubectl get all -n my-ophub
```



► Sizing and Scaling your Optimizer Hub Installation

Perform a test run of a single JVM against your Cloud Native Compiler. In Grafana, check the Time to Clear Optimization Backlog and Compilations in Progress graphs.

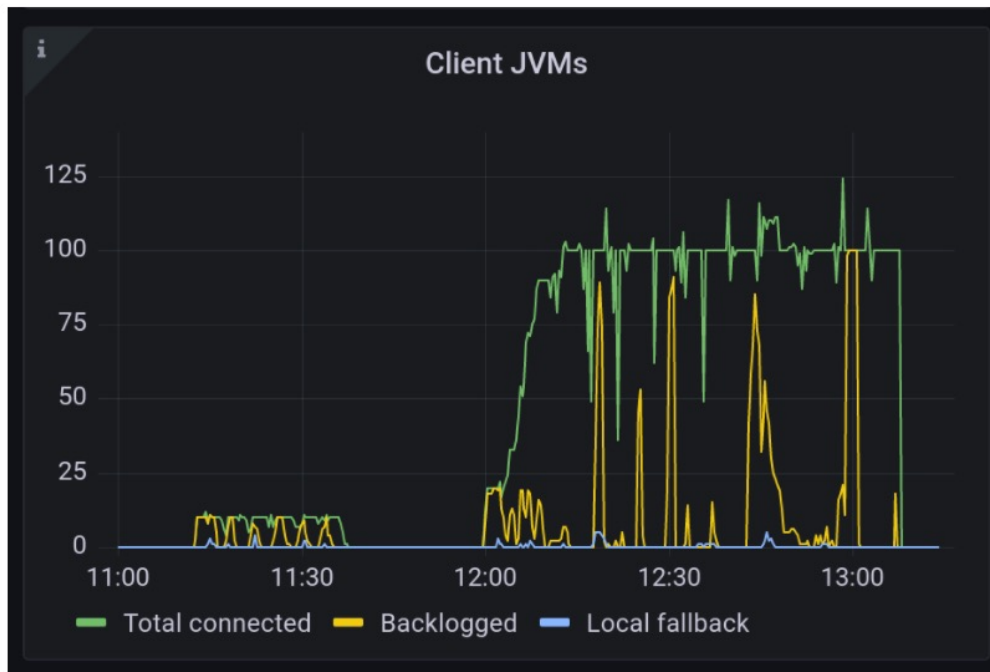


- If the maximum TCOB during your application's warmup is lower than your target, you can scale down the number of vCores provisioned for the job.
- If the maximum TCOB is higher than your target, check the Compilations in Progress metric in Grafana. This metric shows you actual compilations in progress versus Cloud Native Compiler capacity. If you are using the full capacity, add more vCores to the capacity.

You should also check the client JVM logs to see whether the JVM [fell back to using local compilation](#). JVMs switch to local compilation when Cloud Native Compiler becomes unresponsive or tells the JVM that it cannot handle any new requests. You can also see the number of local fallbacks in the Grafana dashboard.

► Sizing and Scaling your Optimizer Hub Installation

You should also check the client JVM logs to see whether the JVM [fell back to using local compilation](#). JVMs switch to local compilation when Cloud Native Compiler becomes unresponsive or tells the JVM that it cannot handle any new requests. You can also see the number of local fallbacks in the Grafana dashboard.



► Sizing and Scaling your Optimizer Hub Installation

Configuring Capacity

Depending on your autoscaling settings, there are three variables you will need to set:

```
simpleSizing:  
  vCores: 32  
  minVCores: 32  
  maxVCores: 106
```

- `vCores` - Total number of vCores that will be allocated. This does NOT include resources required by monitoring, if you enable it. The minimum amount of vCores for provisioning Cloud Native Compiler is 29.
- `minVCores` - The minimum amount of resources that are always allocated when [autoscaling](#) is enabled.
- `maxVCores` - The maximum amount of resources that are allocated when [autoscaling](#) is enabled.

Configuring Autoscaling

Autoscaling is enabled by default in the Helm chart. To disable autoscaling, add the following to `values-override.yaml`:

```
autoscaler: false
```

If you use the Azul-provided [cluster config file](#), the pre-defined node groups for the `gateway`, `compile-broker` and `cache` components already contain instructions to work with Autoscaler. If the Autoscaler Node sees any unused nodes, it deletes them. If a replication controller, deployment, or replica set tries to start a container and cannot do it due to lack of resources, the Autoscaler Node knows which service is needed and adds this service to the Kubernetes cluster. For more information, see <https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/>.

In order to use HPA autoscaling, you need install the [Metrics Server](#) component in Kubernetes.

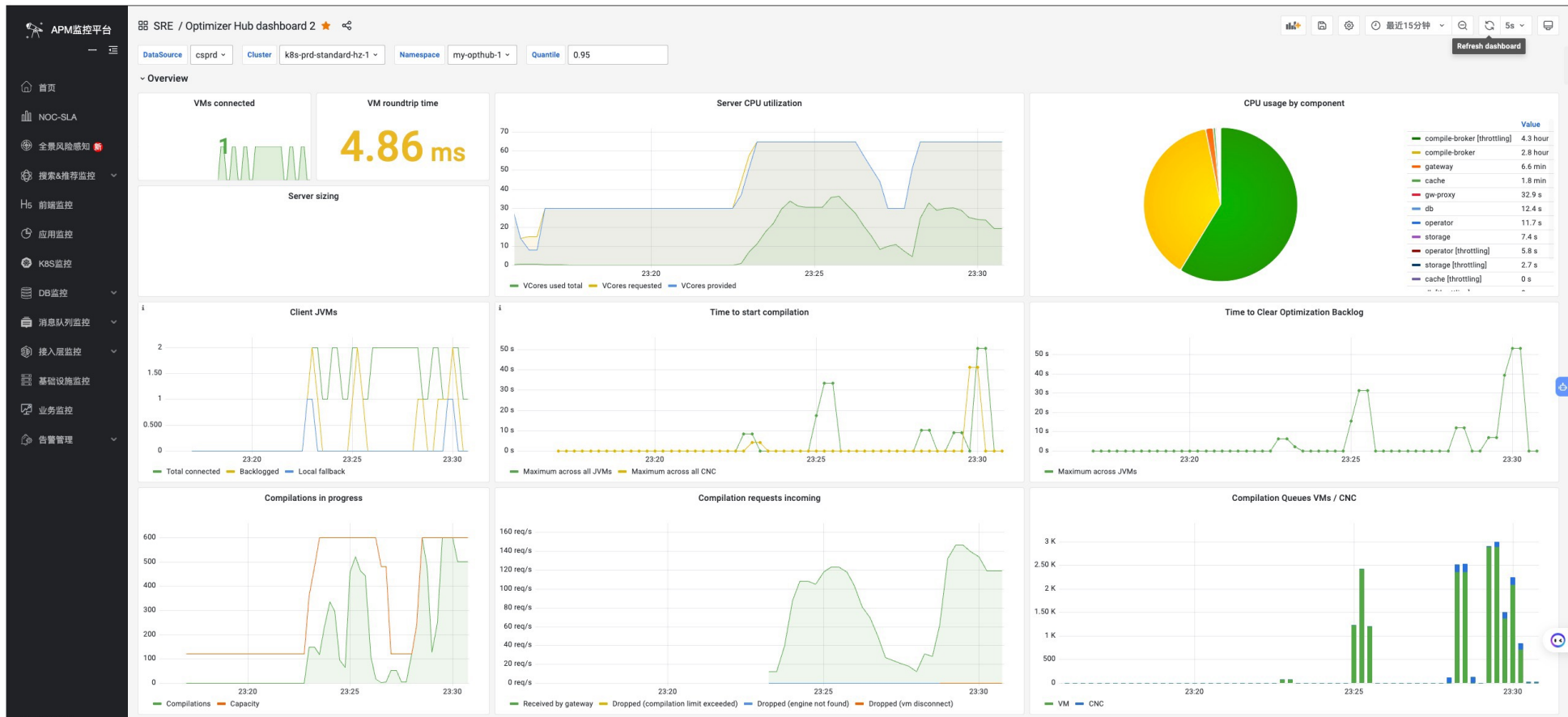
► Monitor Optimizer Hub

Grafana Dashboard

You can find a Grafana configuration file `cnc_dashboard.json` in [opthub-install.zip](#).



Troubleshooting Optimizer Hub



Troubleshooting Optimizer Hub

compilations:

```
minVmVersionForCNCCompilation: ""
```

parallelism:

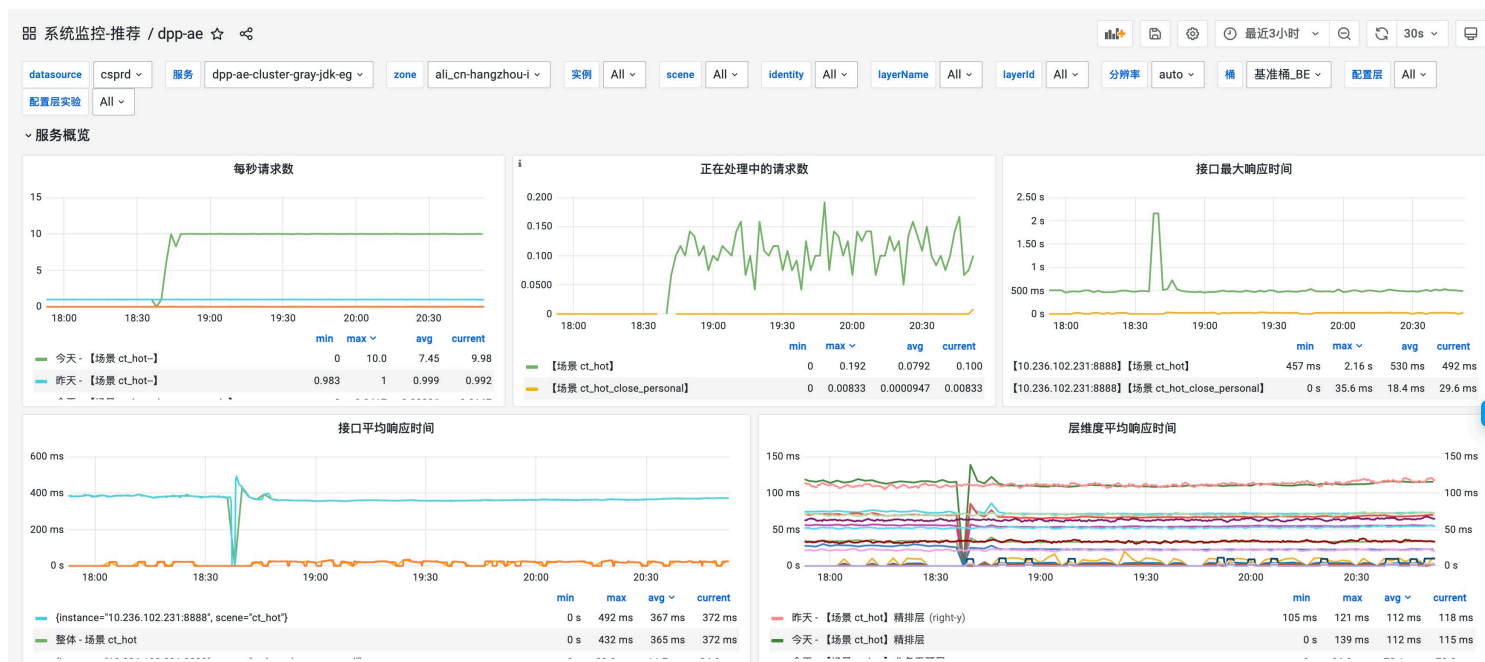
```
# Maximum concurrent compilation limit per VM (VMs will not send more than this amount at the same time).
```

```
limitPerVm: 500
```

```
# Maximum amount of parallel compiler engine processes running at the same time per compile broker.
```

```
limitPerCompileBroker: 30
```

```
lookupParallelism: 120 # calculated as limitPerCompileBroker * 4
```



➤ Benefit of Optimizer Hub



Zing JDK



OpenJDK

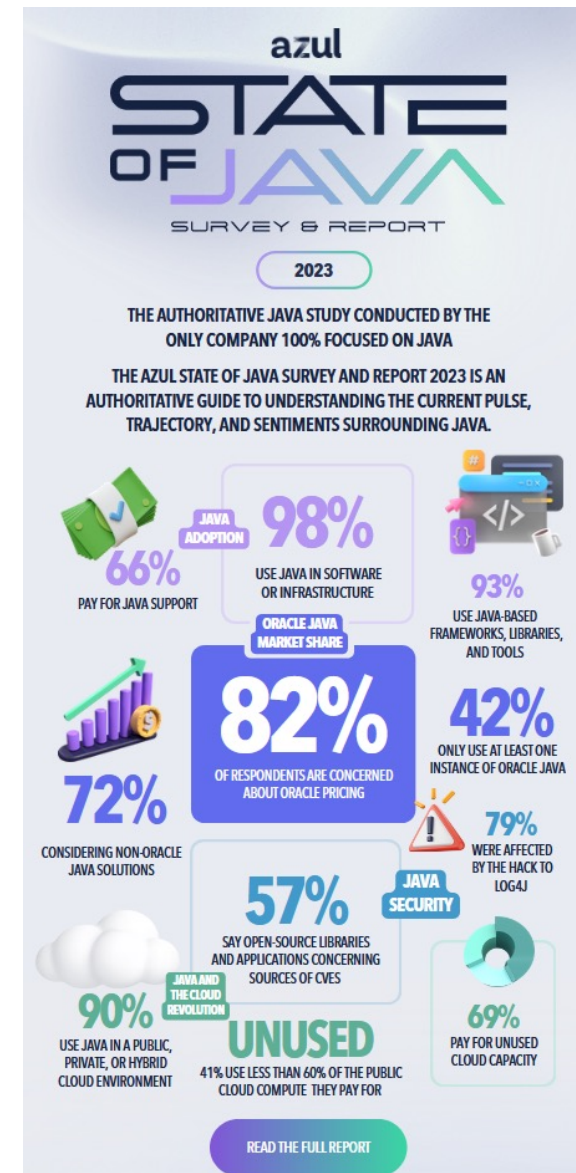
► Interaction Between Optimizer Hub and JVMs

**READYNOW
ORCHESTRATOR**

► 云计算与Java计算的成本 💰

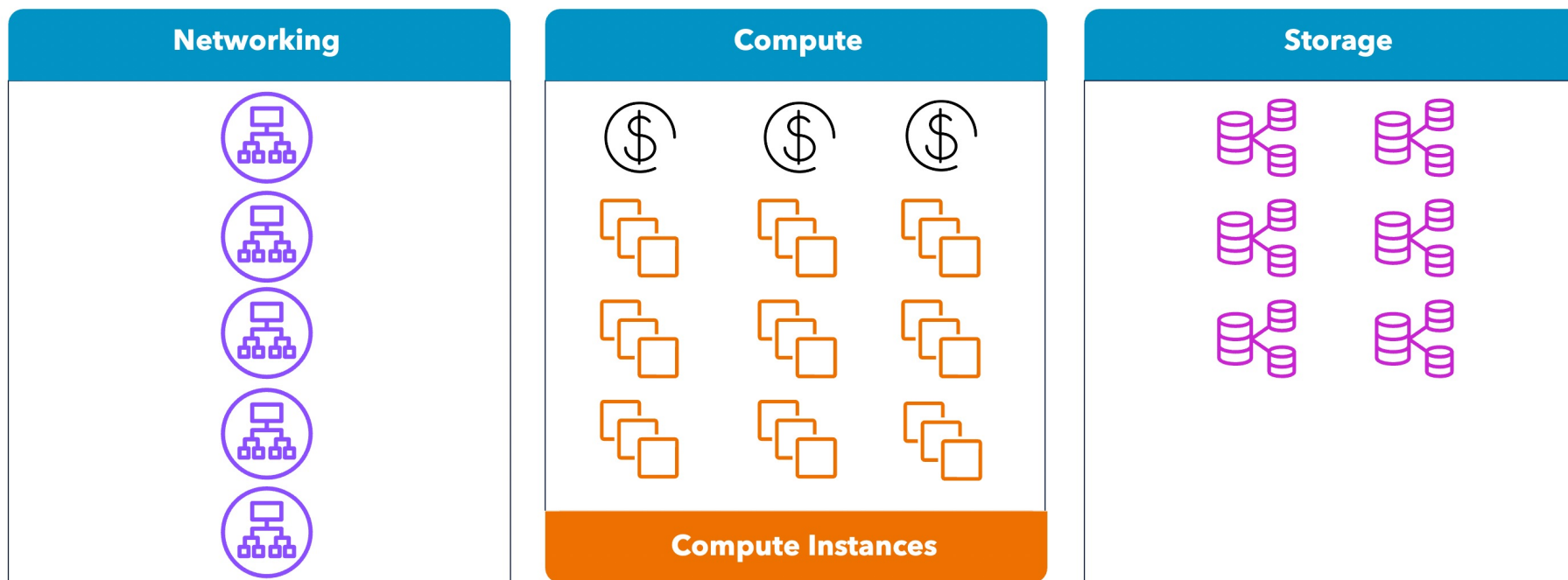
- 90%的受访公司表示，他们在云环境中使用Java
- 70%的受访公司表示，他们认为自己过度调配了云资源
- 95%的受访用户表示，他们正在采取措施降低云成本

Source: "2023 State Of Java Survey Report" Survey commissioned by Azul Systems (Oct 2023)



► 云成本分析

计算资源占据了 ~60%¹ 的云账单

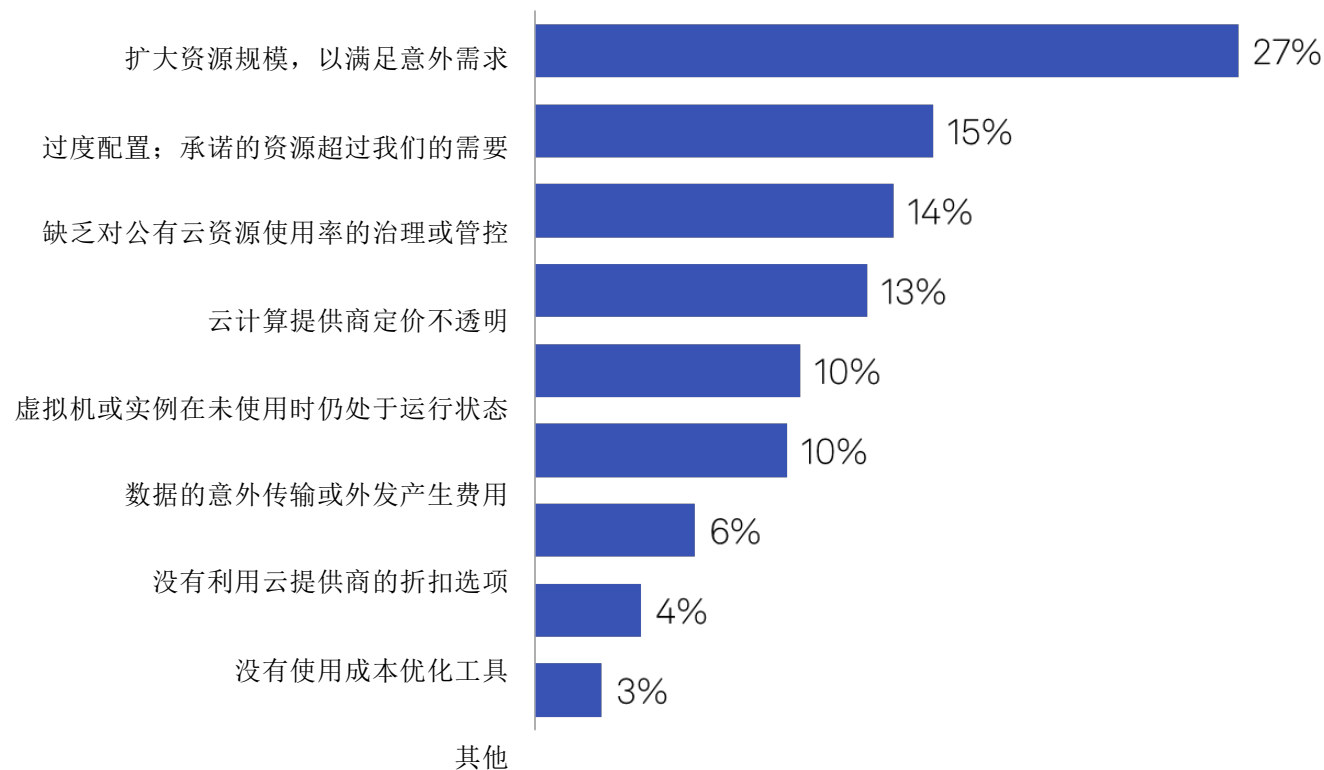


Shared Services



1. <https://www.datamation.com/cloud/cloud-costs/>

造成云计算超支的最大因素



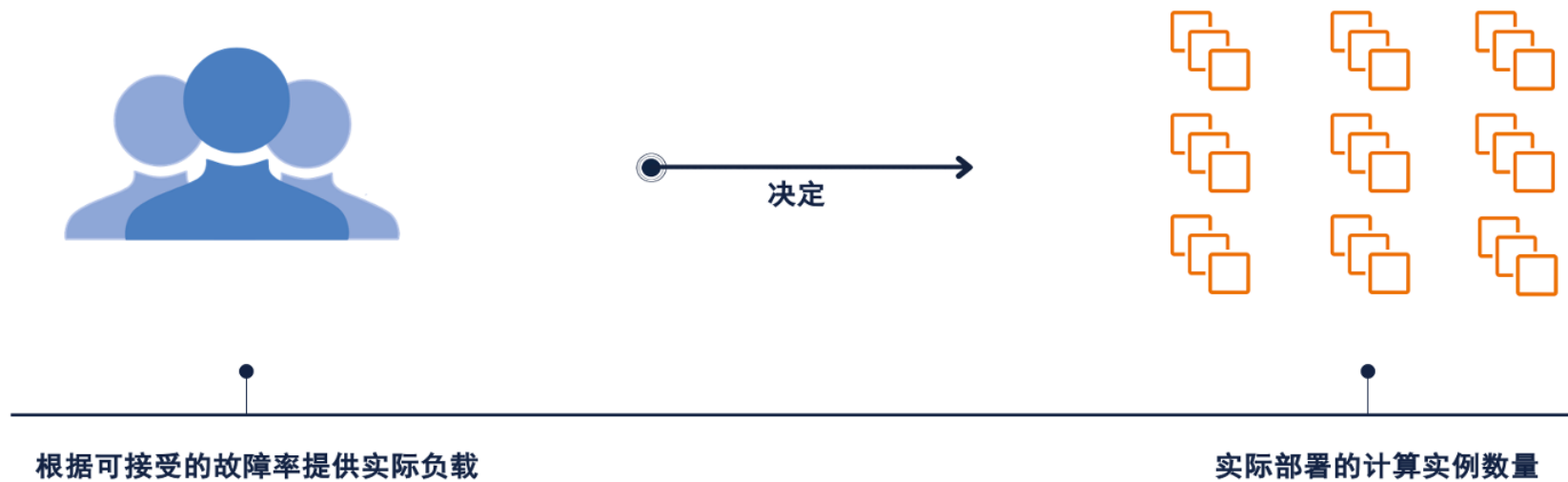
问题：以下哪一项是 2022 年导致 IaaS/PaaS/公有云花费超支的最主要原因？

调查基础：在 IaaS/PaaS/公有云上支出超出预算的企业，简略调查 (n=199)。

来源：451 Research 的《企业之声：云、托管与托管服务，2023 年云支出》。

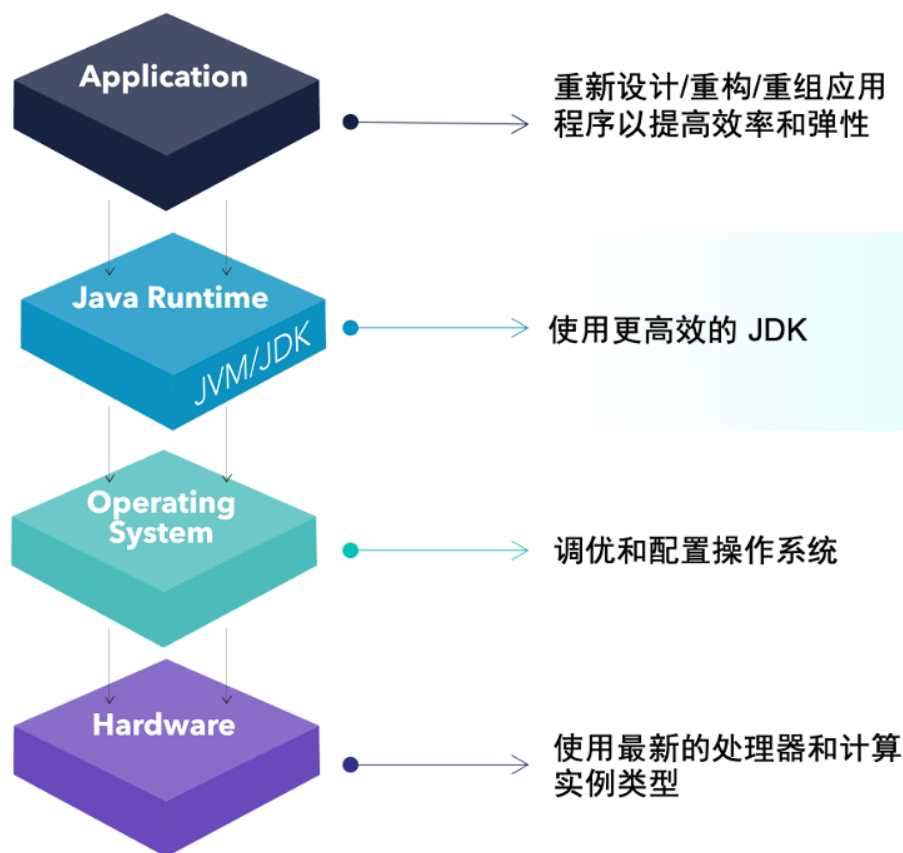
➤ 云实例配置-sizing

所需的云计算实例数量...



云如何节省成本 💰

如何减少云计算实例总数？



Azul Platform Prime

Includes Azul Zing, an enhanced build of OpenJDK for superior performance, consistency, and efficiency.

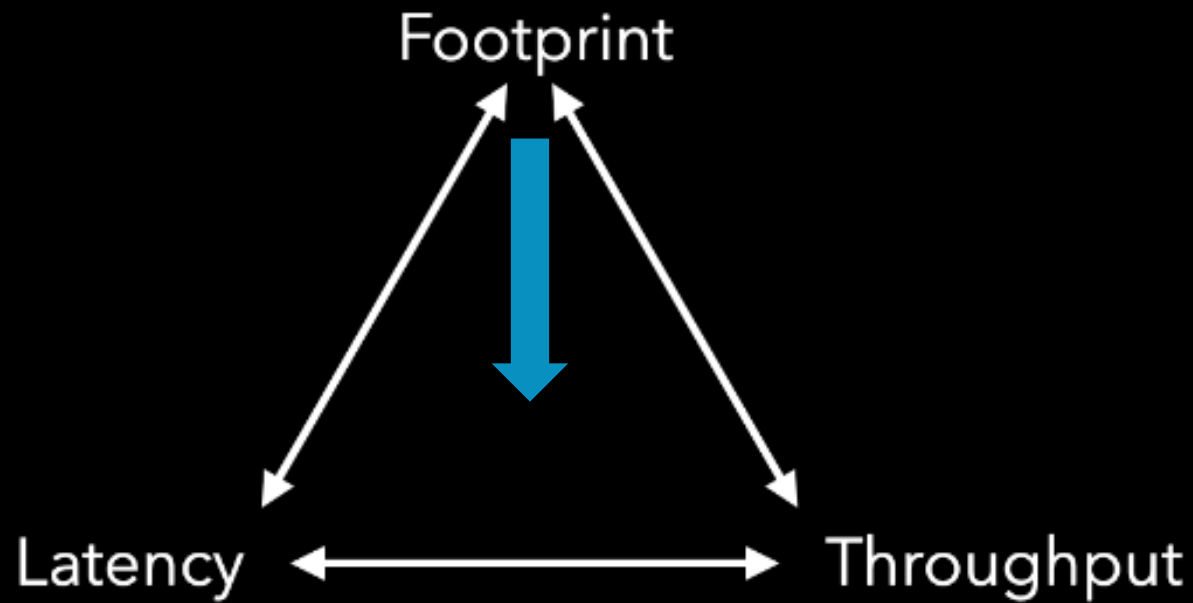
Azul Zing Builds
of OpenJDK

Optimizer Hub

Monitoring Tools

Support and
Custom Builds

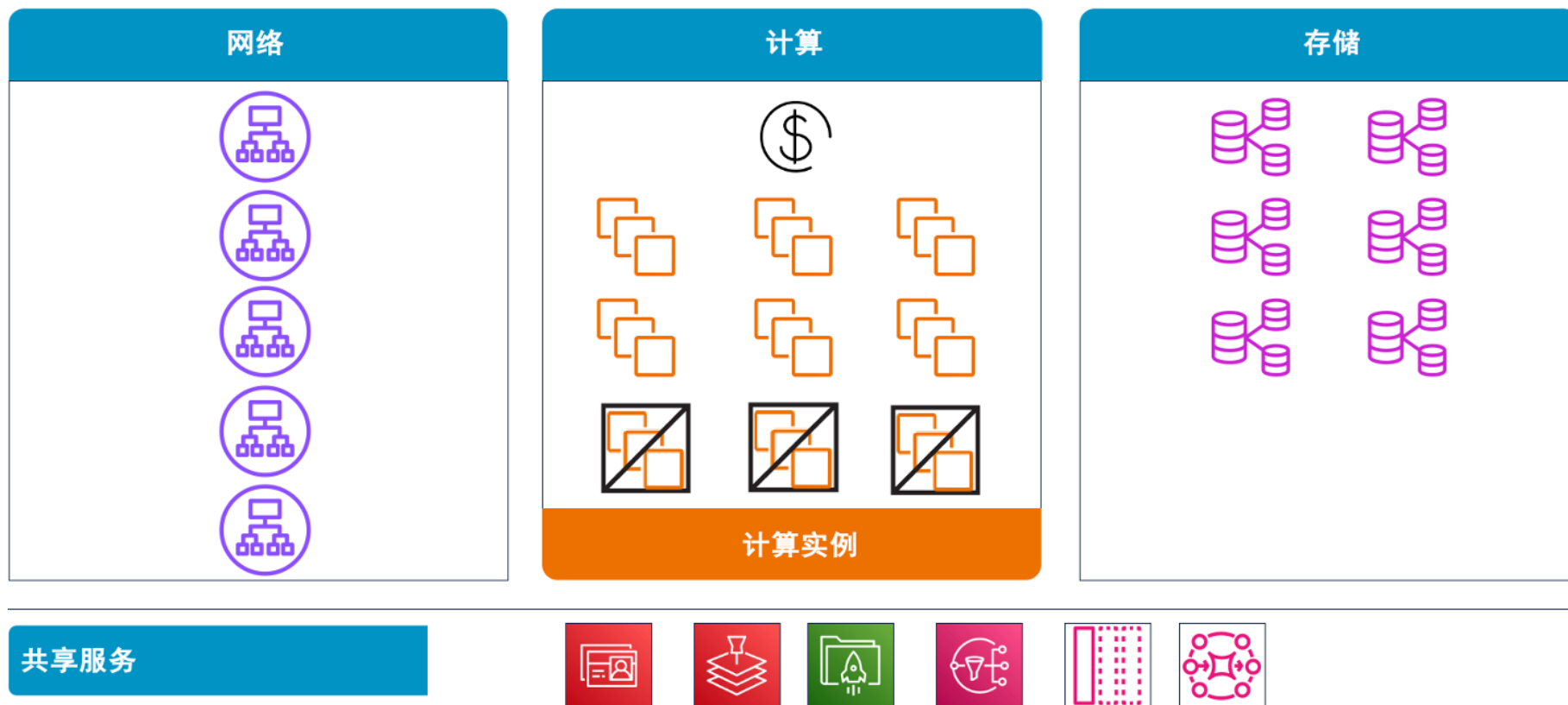
➤ 性能不可能三角形 🙅 既要快又要多还要省!



Performance Triangle

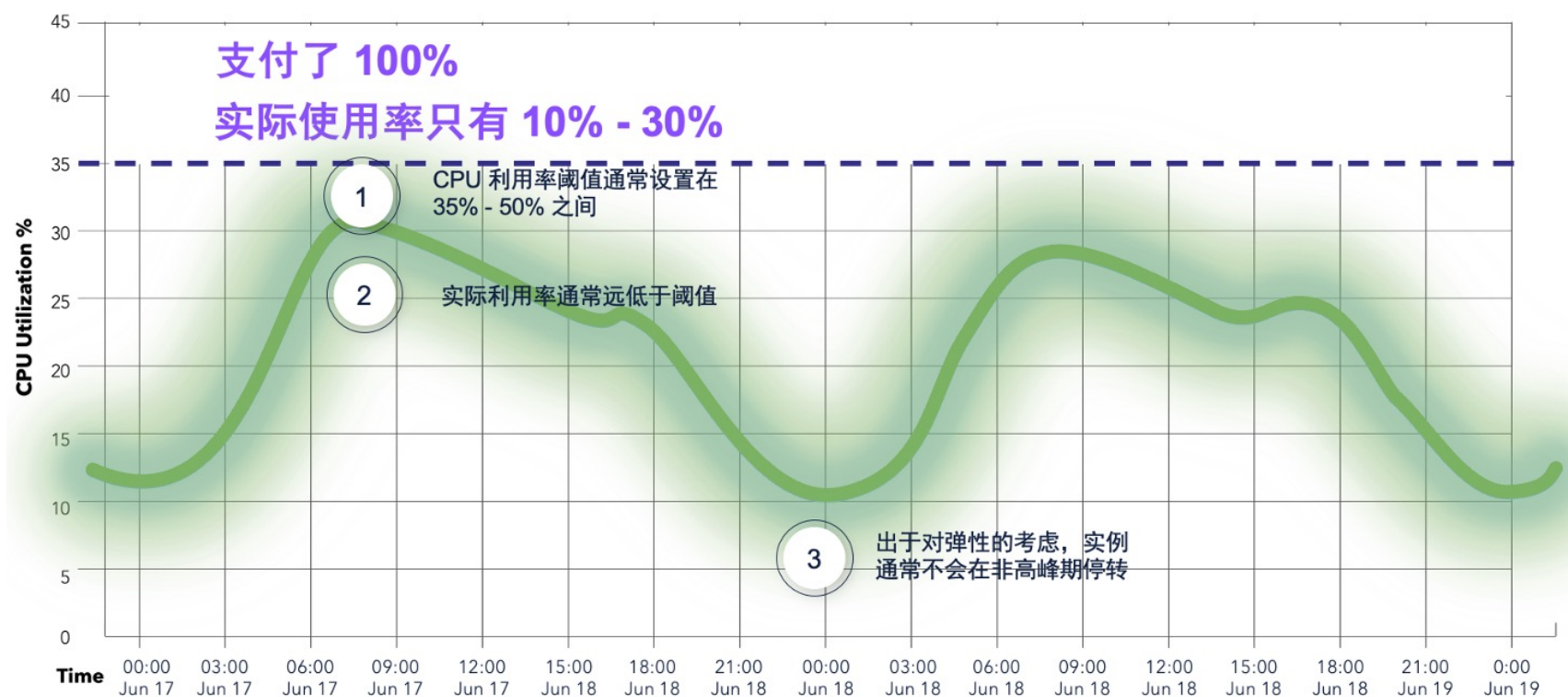
▶ 打磨好的利器-(十年磨一剑)

无需任何代码更改即可将云计算减少 20% - 50%



非常打脸的现实-(实际使用率不到30%)

云计算的普遍现实



azul

*匿名客户真实案例

Confidential | Do Not Distribute

▶ 直接上链接 - (<https://www.azul.com/products/prime/>)

 **Azul Platform Prime 全球最快、最高效的 JVM**
它是 OpenJDK 的增强版本，具有卓越的性能、一致性和承载能力



代码执行更快，Java 应用程序更快

运行 Java 代码的速度提高 20% - 50%

优化代码执行以充分利用最新处理器

使用更好的基于 LLVM 的 JIT 编译器改进
推测优化



一致性更高，利用率更高

在较高负载下保持一致的
服务水平

随着时间的推移减少 CPU 使用中的噪声

消除暂停、抖动和停滞

利用 C4 – 世界上最好的无暂停垃圾回收器



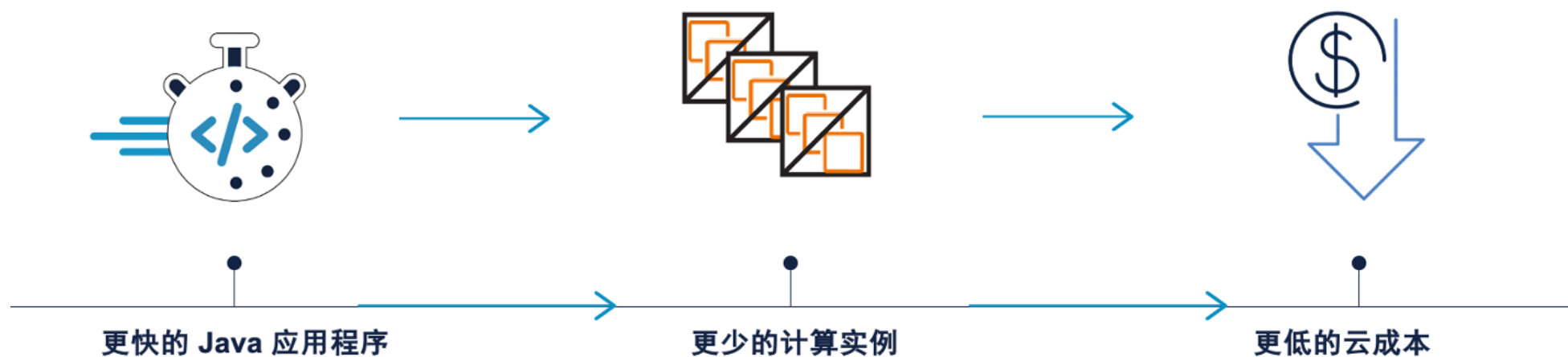
预热改善，弹性增加

显著减少因添加
新实例而引起的
中断和故障

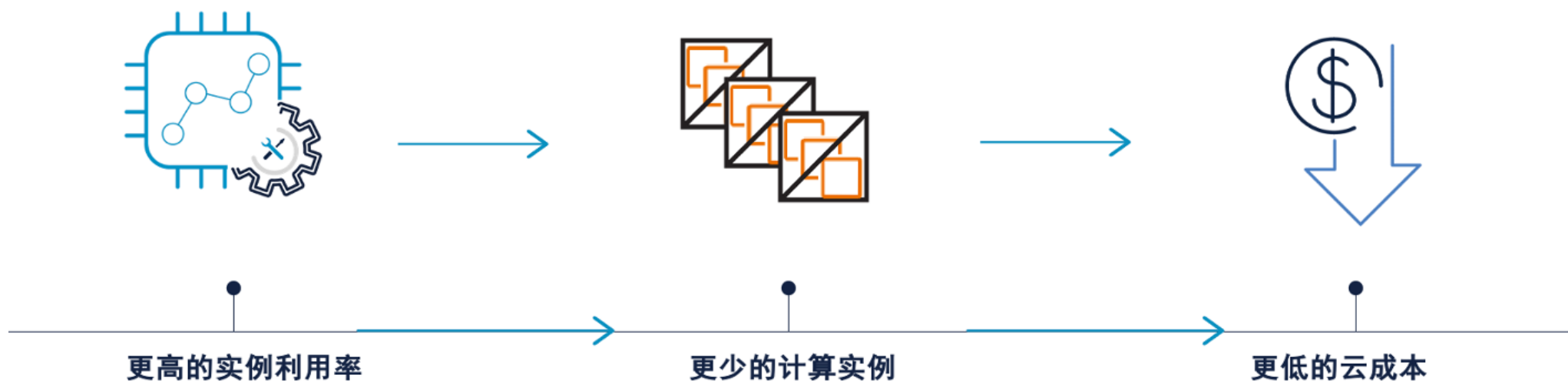
从一开始便全速运行

使用 ReadyNow! 技术调整、
学习和传播优化配置文件

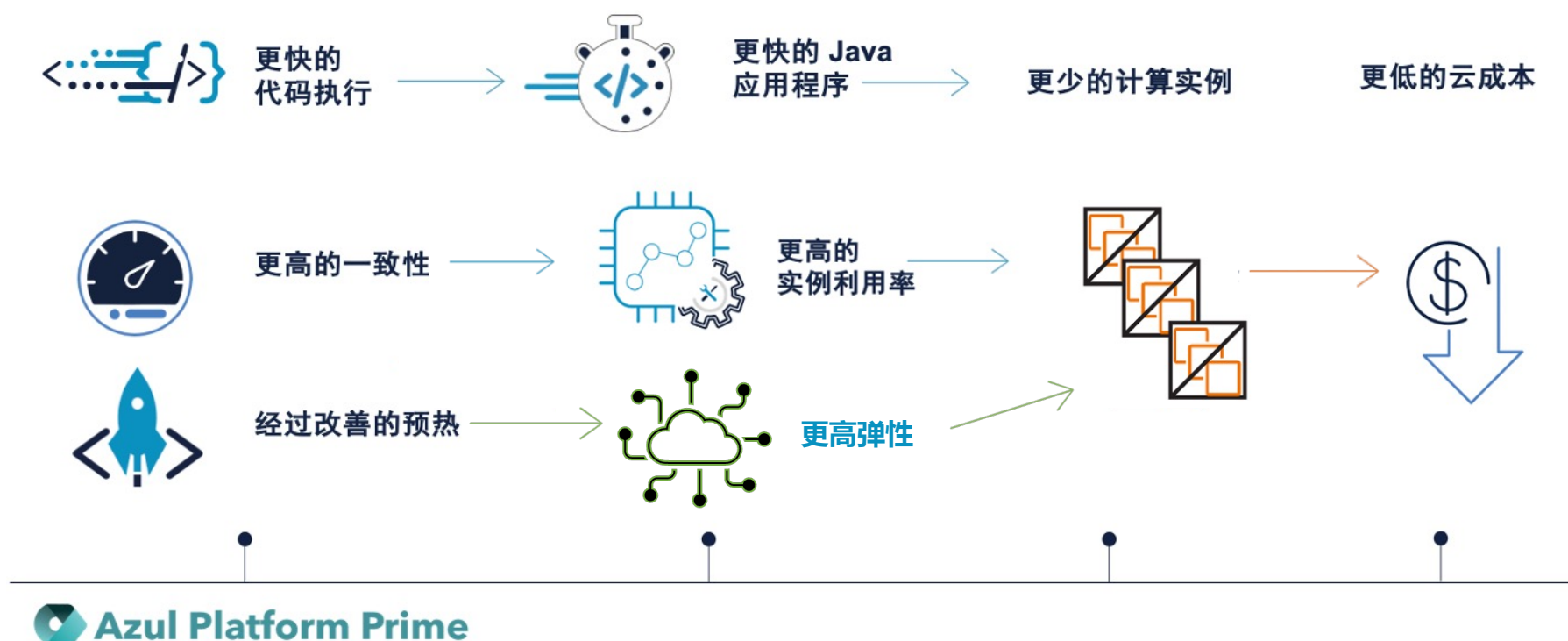
▶ 天下武功唯快不破-第一步（更快运行效能）



► 增加稳定性与抗压能力-第二步（加压增加CPU水位）

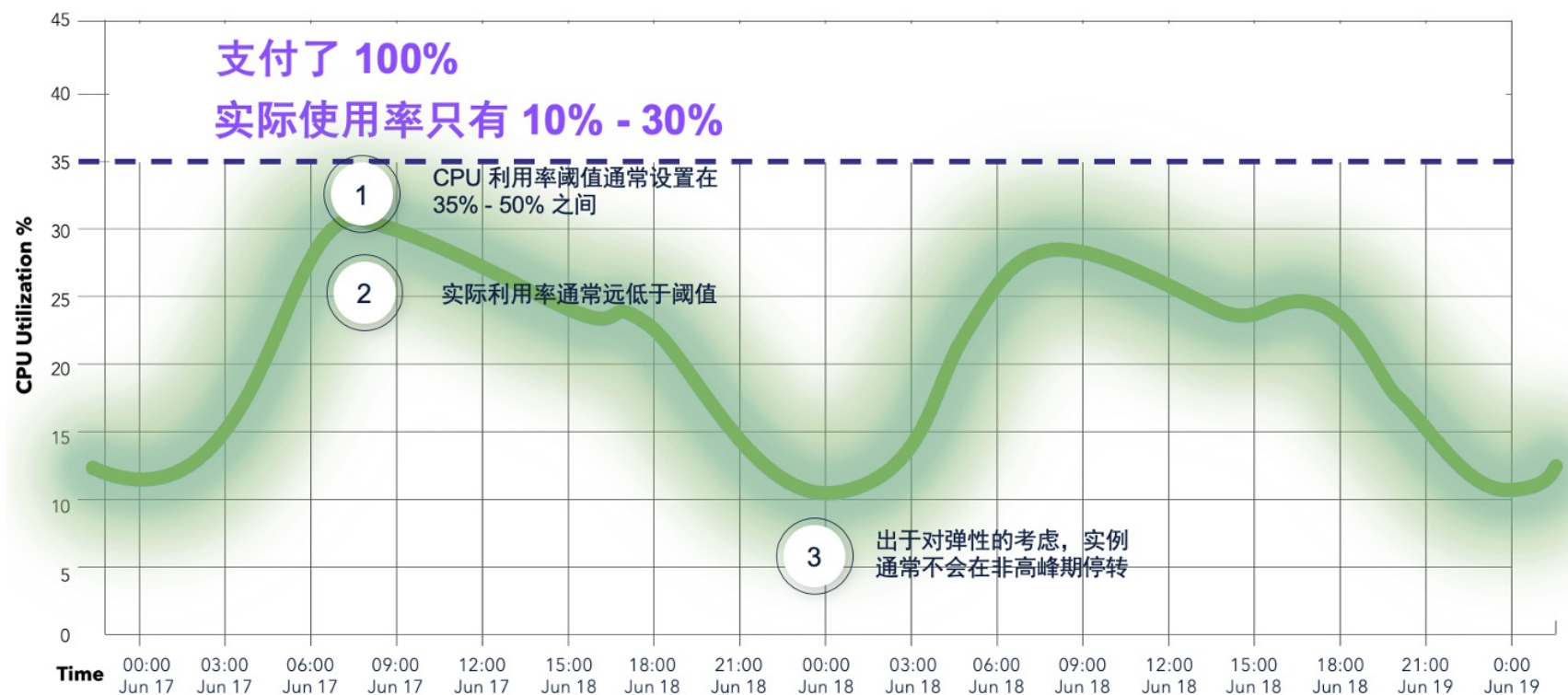


➤ 更高弹性-第三步 (更加高效启动提升弹性)



再次面对现实

云计算的普遍现实

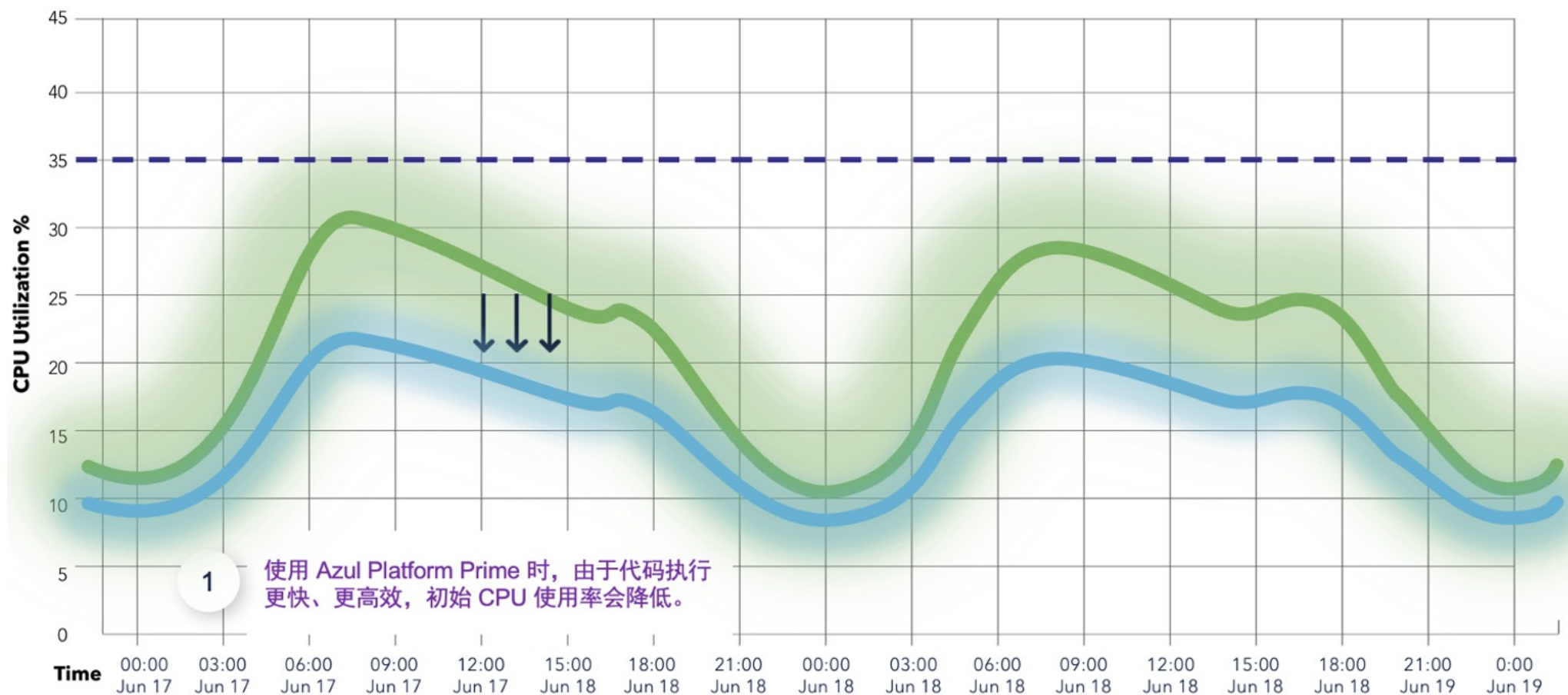


azul

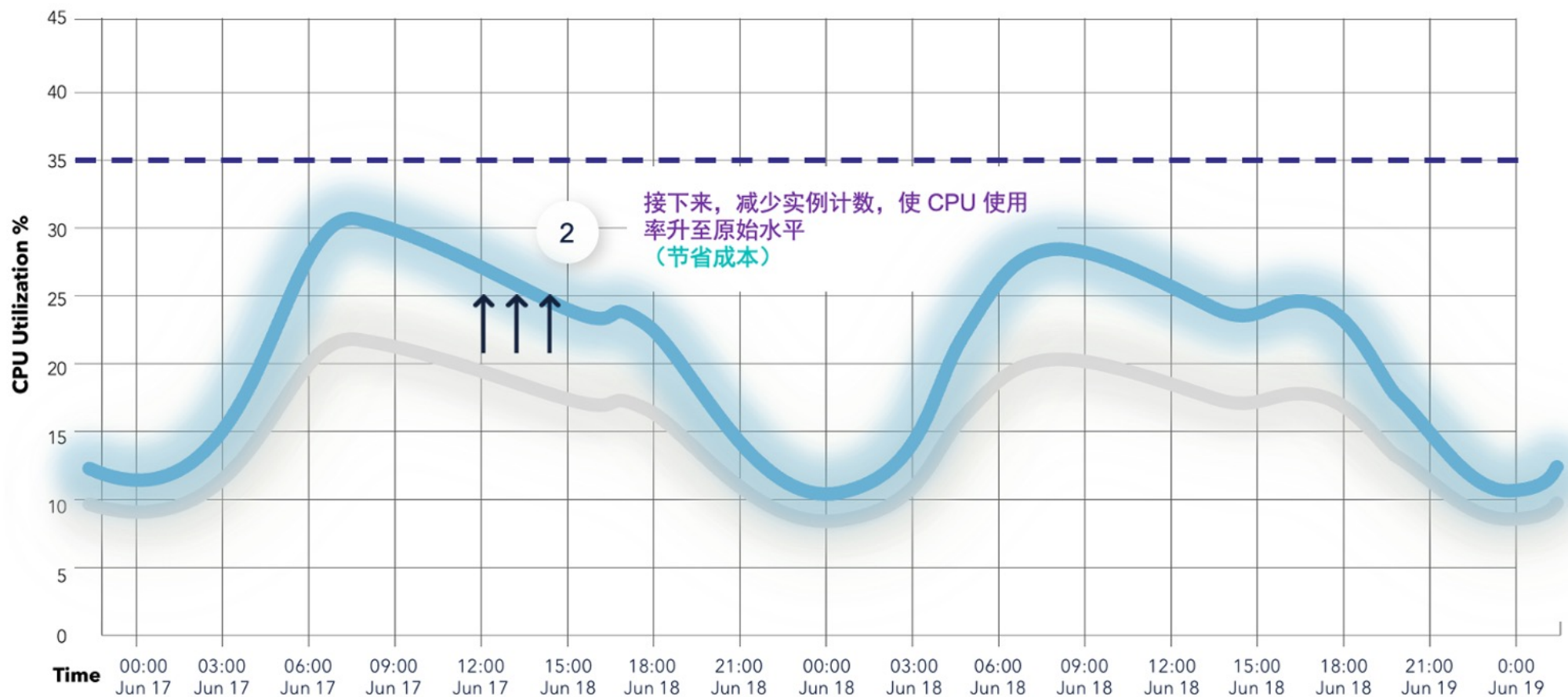
*匿名客户真实案例

Confidential | Do Not Distribute

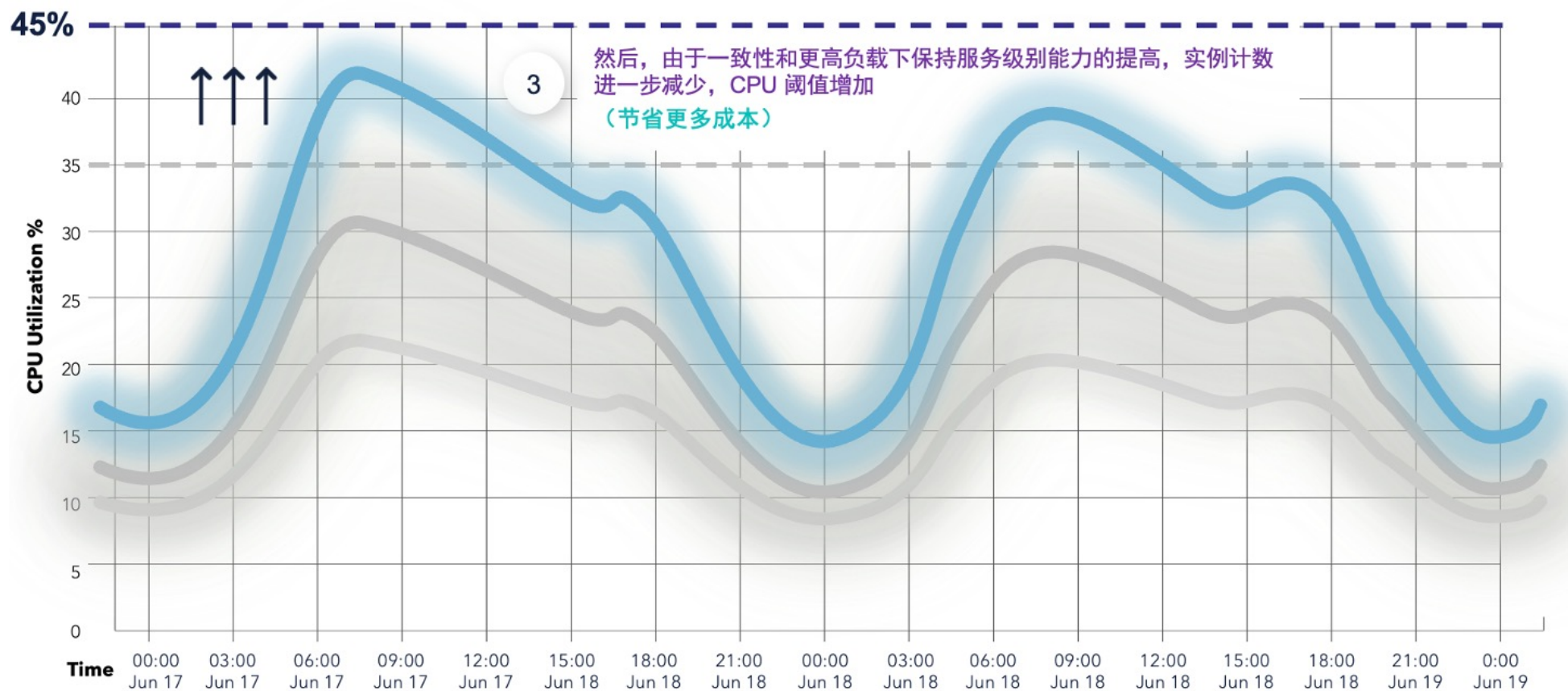
► 如何降低云成本-更快编译更快运行



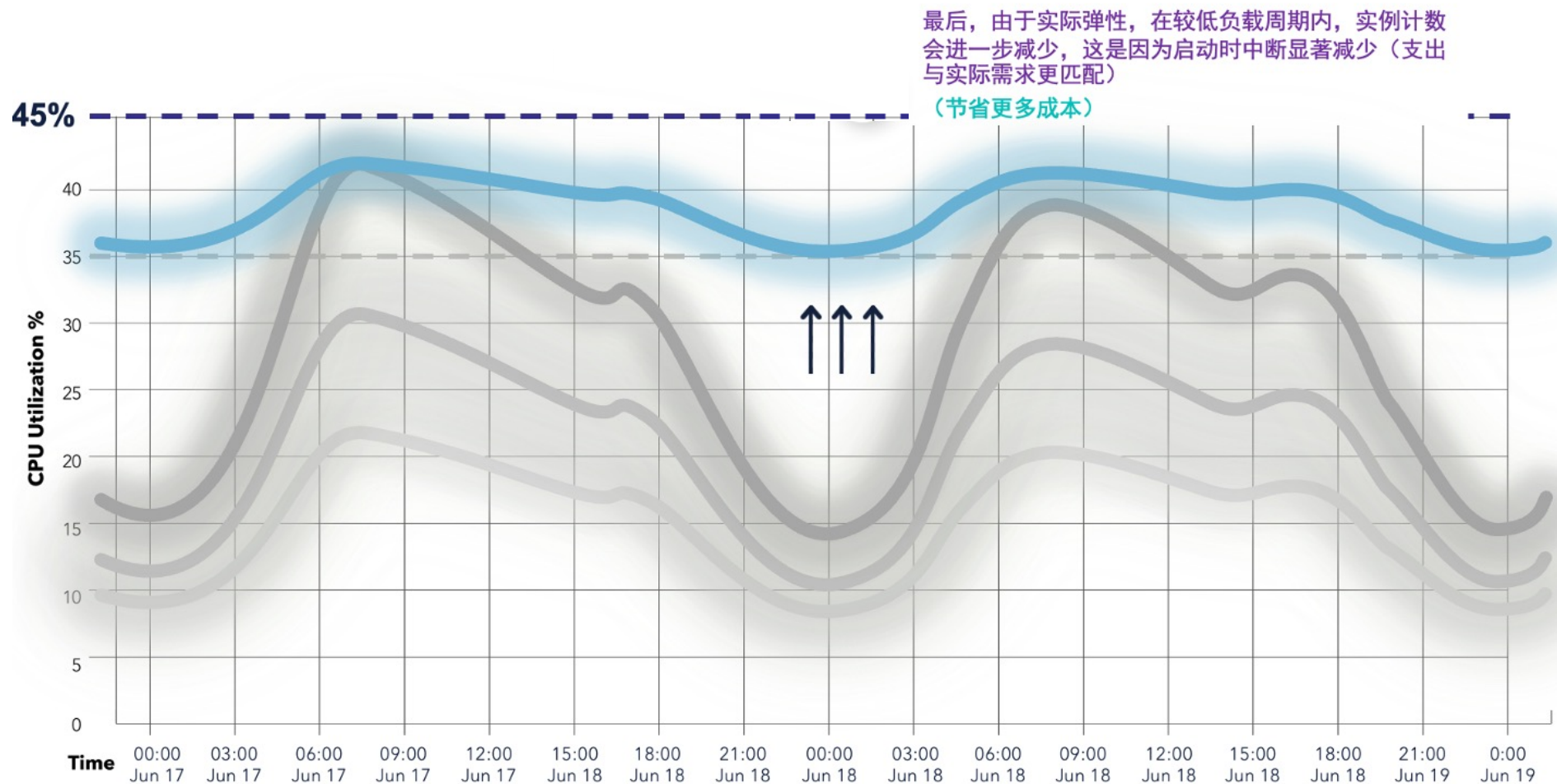
► 如何降低云成本- CPU水位降低



► 如何降低云成本- 加压增加CPU水位

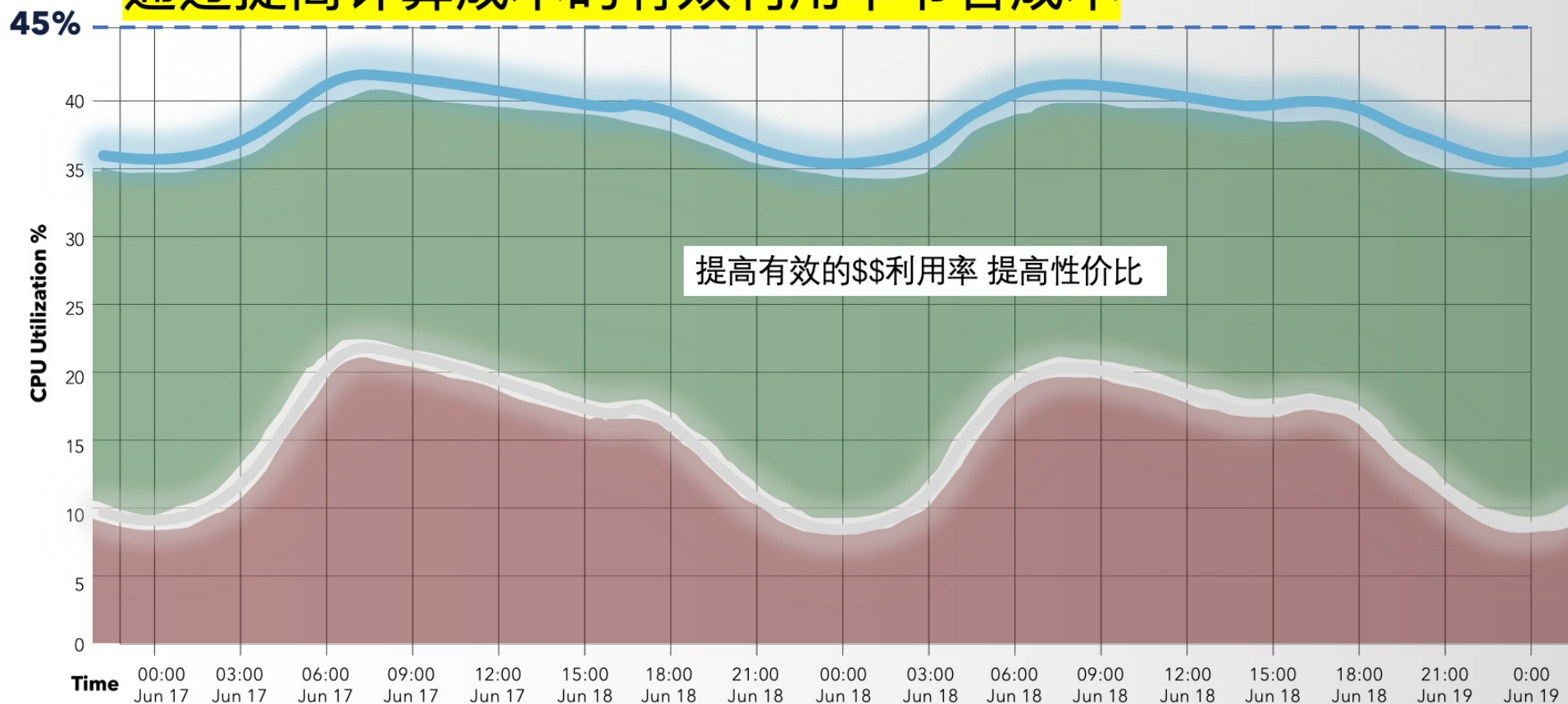


► 如何降低云成本- 叠加更好的弹性



► Azul Optimizer Hub: 如何降低大规模云成本

通过提高计算成本的有效利用率节省成本





A Large Online Retailer

Search everything online and in store



📍 US-based



Contemporary Java-based services (Cloud, K8s, Service Meshes, Microservices)

- 使用Azul Platform Prime的60多个核心应用程序
 - 在大约12个月内逐步完成
 - 不更改代码
 - 不更改DevOps workflow
 - 开箱即用方式替换JDK
- 平均将CPU效率提高约20%
 - 转化为立即减少每个应用程序所需的占用资源
- 将平均利用率从10%-35%提高到35%-45%
 - 通过提高单个实例的CPU%阈值
 - 通过提高弹性采用率并减少每个应用程序的最小Pod数量
 - 随着时间的推移而实现，每个应用程序通常需要2-4个月
 - 通过负责任的循序渐进地提高利用率
- 实现了跨过渡的基于Java的应用程序的总体计算量平均减少40%以上

Transitioned
60+
Core applications
to Azul Platform
Prime

Increased CPU
efficiency by
20%
on the average

Improved average
utilization from
10% - 35% to 35% - 45%

>40%

An overall compute reduction
across the Java-based apps

更多可公开的落地案例- <https://www.azul.com/customer-success/>

客户降低云成本



基础设施将增加 5 倍

无需借助 Azul Platform Prime

Azul Platform Prime



回收 20% 以上未使用的
CPU

承载能力，改善玩家体验



服务器占用空间
减少 30% 以上

节省了数百万美元



借助基于云的财务系统

每笔交易的成本降低 85%



通过提高性能

基础设施支出减少 38%



回收 50%

未使用的基础设施用于
电子交易

可公开分享的落地案例



云成本优化的新武器-ZingJDK | 得物技术

Azul 可以基于源码做特性增强构建, 生成 ZuluJDK 和 ZingJDK。前者属于开源版本, Zin...



得物技术

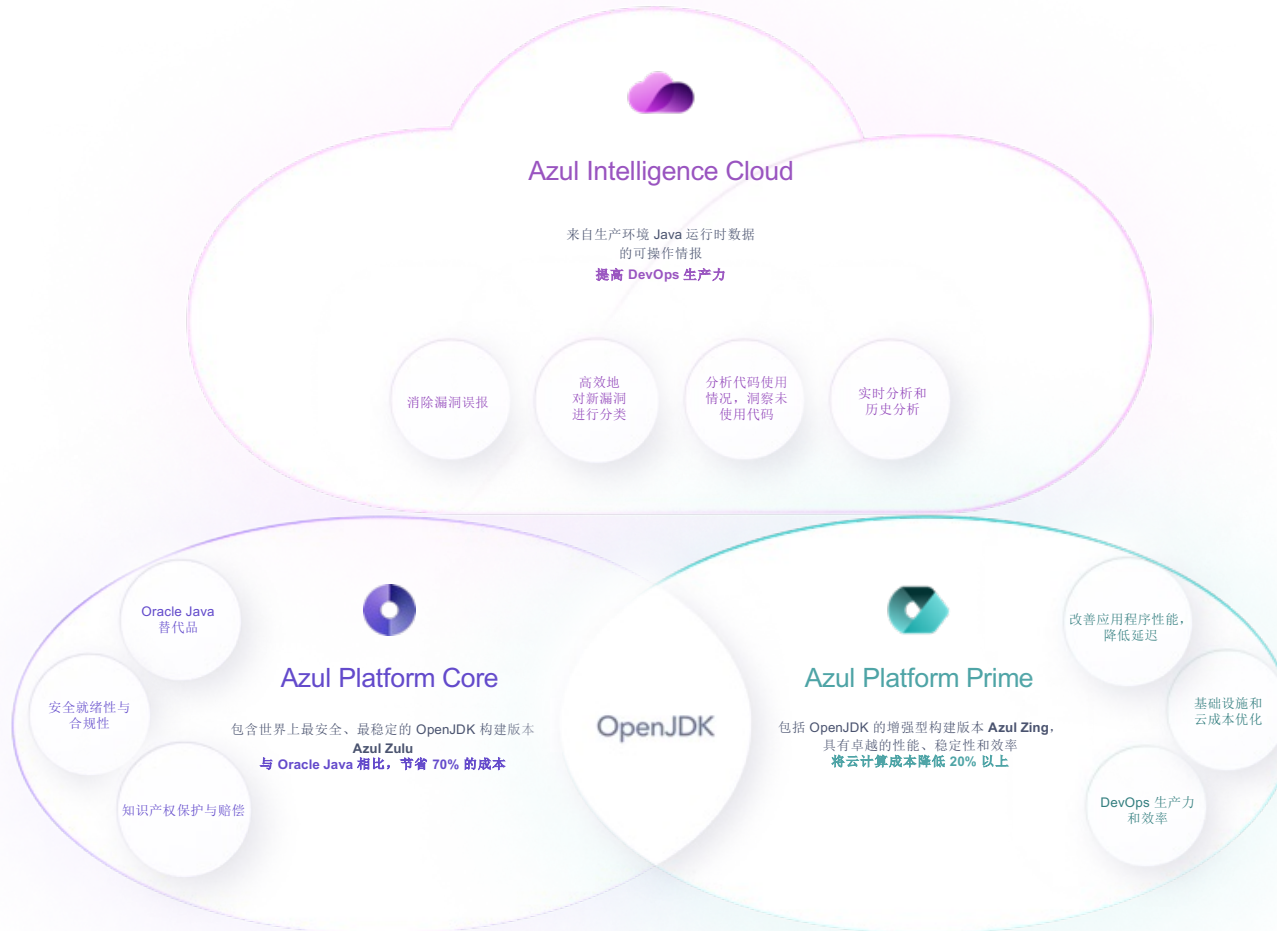
QCon 全球软件开发大会

GTLC北京2020|全球技术领导力峰会 - InfoQ



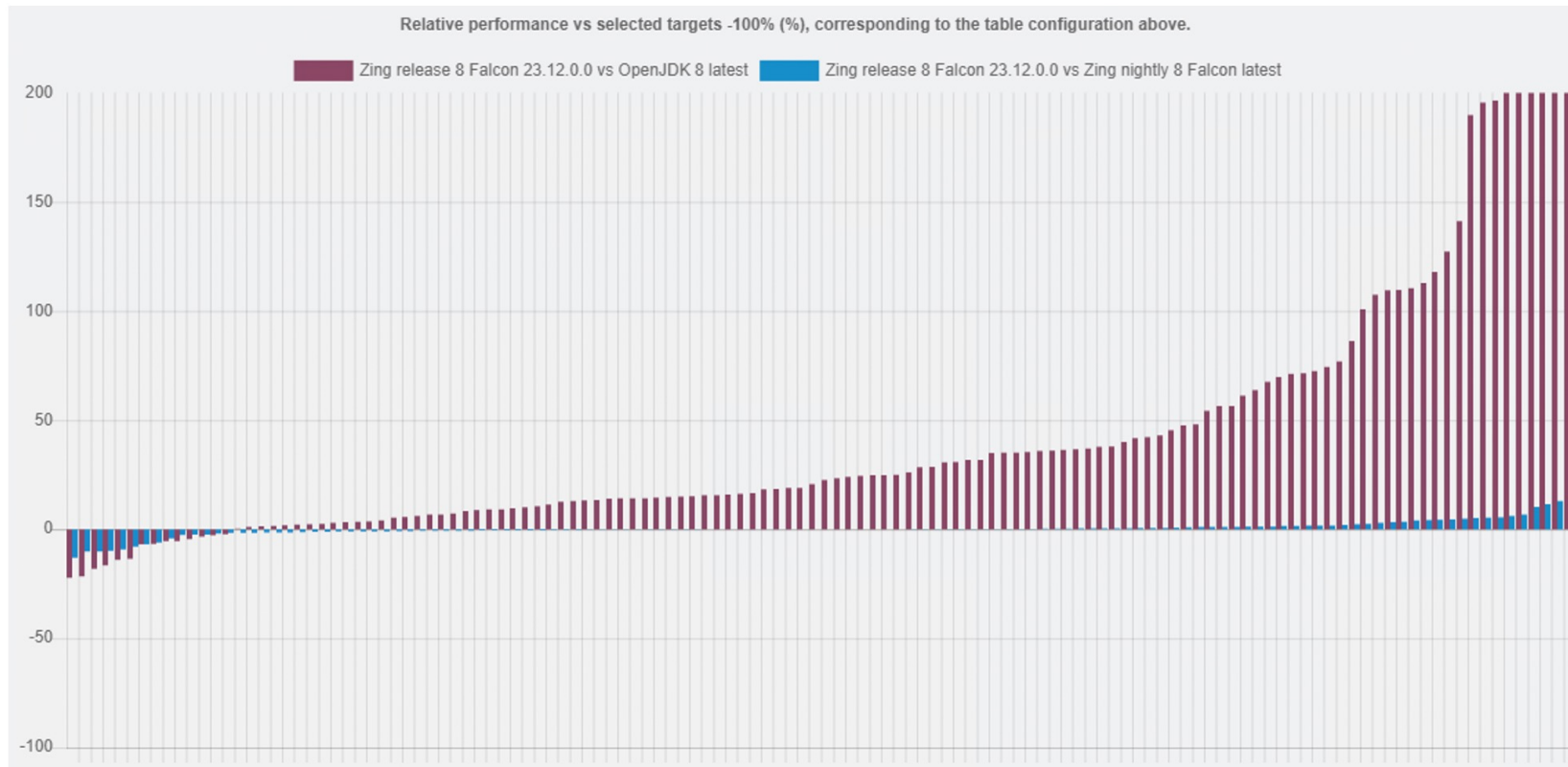
LOUIS VUITTON	1.18B	37
GUCCI	770m	24
CHANEL	710m	22
HERMÈS	640m	20
BALENCIAGA	330m	10
DIOR	240m	8
CELINE	230m	7
PRADA	140m	4
SAINT LAURENT		
YVES SAINT LAURENT		

Azul 产品组合



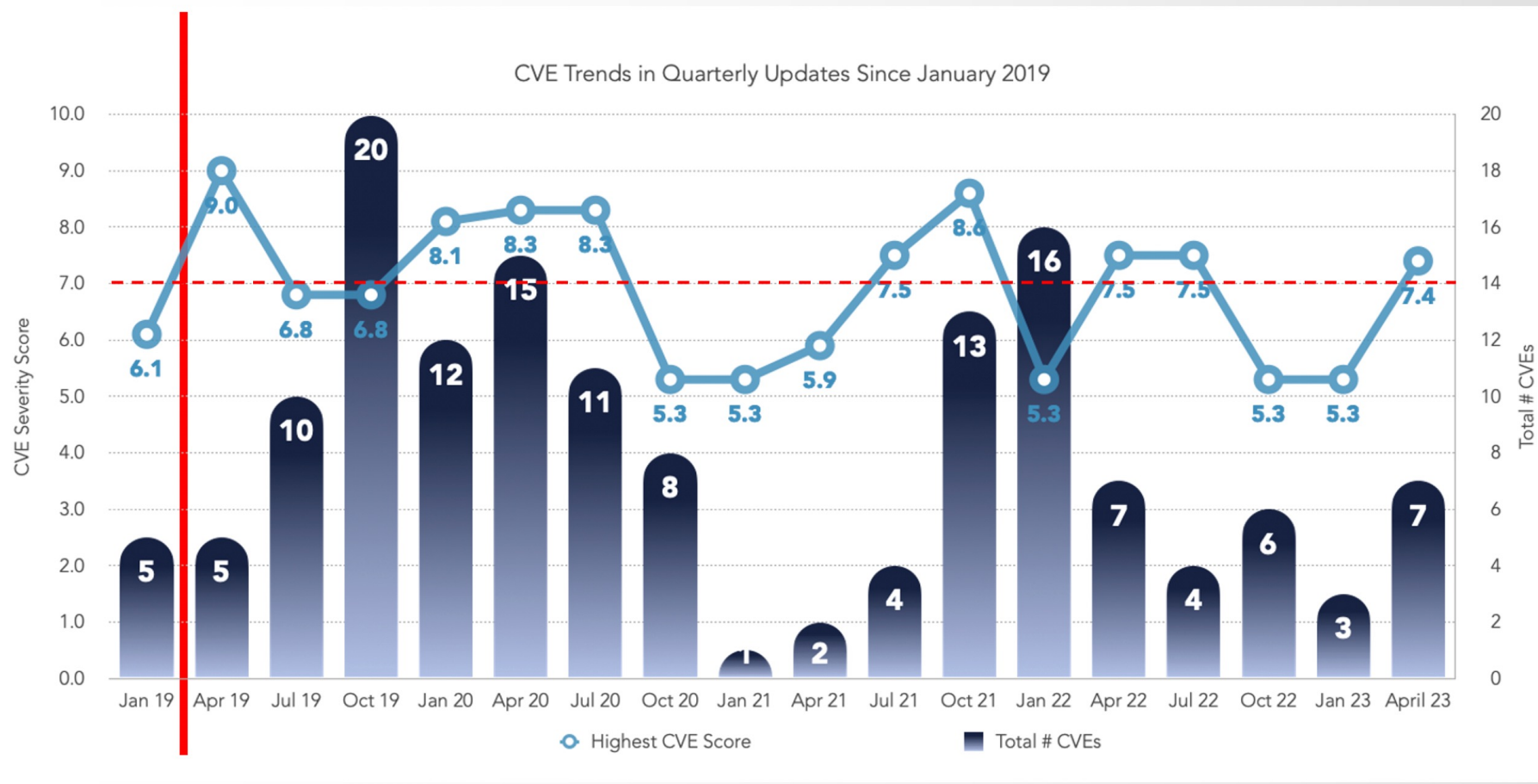
▶ 通过AI不断迭代优化的JVM优化能力 – 5% YoY以上性能提升

Performance S-curve Chart



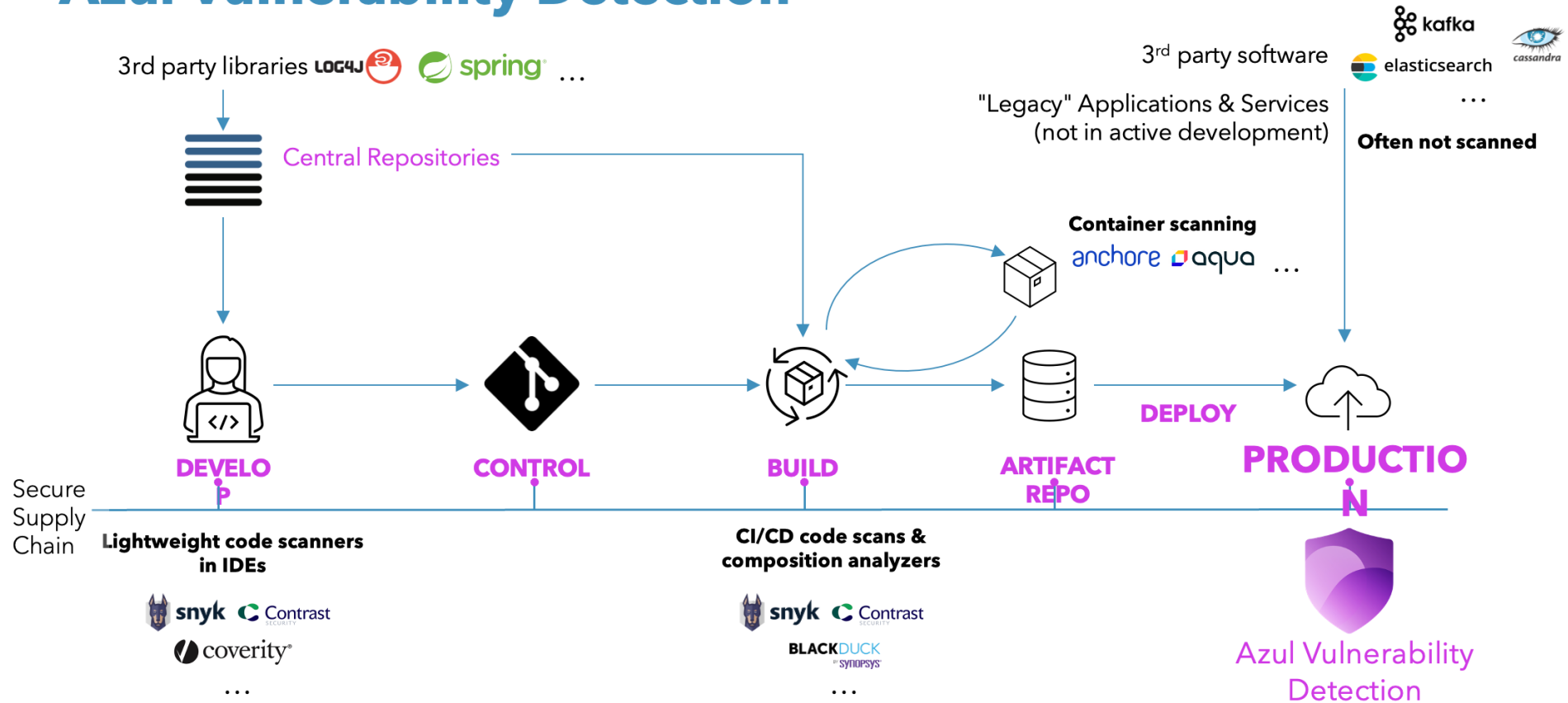
用AI分析CVE漏洞并解决Java安全加固问题

自2019起按季度发布的安全漏洞



检测供应链真正加载和运行的Java高危漏洞

Azul Vulnerability Detection



▶ AIC识别并替换昂贵的O记JDK-收费标准与规则

	Monthly Subscription Price	Subscription Metric	Volume	Pricing Notes
Java SE Universal Products				
Java SE Universal Subscription	15.00	Employee for Java SE Universal Subscription	1-999	1,2,3
	12.00	Employee for Java SE Universal Subscription	1,000-2,999	1,2,3
	10.50	Employee for Java SE Universal Subscription	3,000-9,999	1,2,3
	8.25	Employee for Java SE Universal Subscription	10,000-19,999	1,2,3
	6.75	Employee for Java SE Universal Subscription	20,000-29,999	1,2,3
	5.70	Employee for Java SE Universal Subscription	30,000-39,999	1,2,3
	5.25	Employee for Java SE Universal Subscription	40,000-49,999	1,2,3
		Contact for Details	50,000+	1,2,3

Oracle JDK and Oracle OpenJDK LTS License Guide		
Java Version	Releases	License
Java 21	Oracle JDK, releases through September 2026	Oracle No-Fee Terms and Conditions License for All Users
	Oracle OpenJDK, releases through January 2024	GPLv2+CPE for All Users
Six month releases, Java 18 and later	Oracle JDK, all releases	Oracle No-Fee Terms and Conditions License for All Users
	Oracle OpenJDK, all releases	GPLv2+CPE for All Users
Java 17	Oracle JDK, releases through September 2024	Oracle No-Fee Terms and Conditions License for All Users
	Oracle OpenJDK, releases through January 2022	GPLv2+CPE for All Users
Java 11	Oracle JDK, all releases	My Oracle Support (Oracle Customers Only) and Oracle Technology Network License Agreement for Java SE for Personal, Development and other Users only.
Java 8	Oracle JDK 8u211, April 2019 and later	My Oracle Support (Oracle Customers Only) and Oracle Technology Network License Agreement for Java SE for Personal, Development and other Users only.

▶ 下一步展望

总结 (you can take away)

- 1. Java+AI 还是AI+Java?
 **Azul Intelligence Cloud**
- 2. 如何解决Java应用的三大难题 (GC卡顿, 预慢热, 编译执行代码效能)
 **Azul Platform Prime**
- 3. Azul platform大规模降本且增效的FinOps方案
Azul Optimizer Hub = Azul Cloud Native Compiler + ReadyNow Orchestrator

▶ AI时代的Java进化与FinOps（欢迎进行更多深入交流）

Azul是支持Java栈解决方案的最佳合作伙伴

Thank You.

