

阿里服务化架构演进

Part I

单一应用架构

All in one

- 整个网站几个应用
- 前台 web + 后台 ops + tasks
- 业务 web + service/dao 各自开发
- 一起集成发布



技术栈

- Webx
- Spring + Ibatis
- Jboss
- Oracle

存在的问题

- 合并时经常代码冲突
- 发布相互制约效率低下
- 应用代码庞大臃肿维护困难

垂直应用架构

按应用拆分

jar

Service / DAO / Impl 都以二方库 jar 的形式提供出去

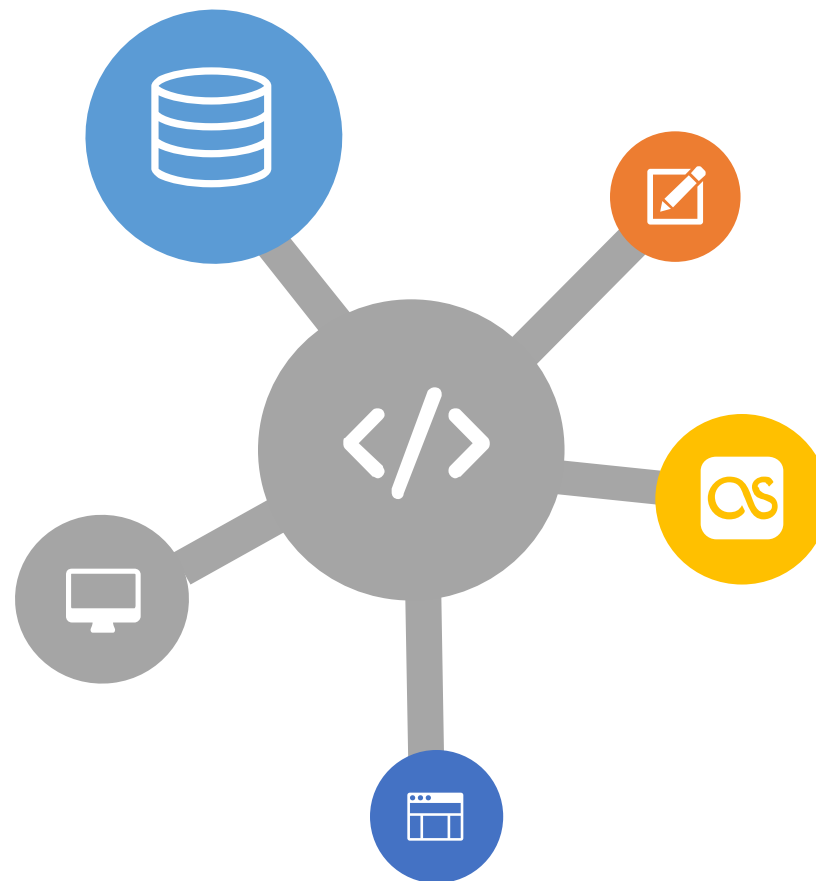
- 代码拆分，独立部署，流程隔离，技术栈没有太大变化
- 应用相互之间直接依赖二方库
- 问题：
 - 升级困难，要全网推动
 - 数据库连接池压力大



微服务架构

Boot 拥抱微服务，提升开发体验和效率

- 应用更轻量、开发更简单
 - 配置
 - 编码
 - 开发
 - 调试
 - 部署
- 技术栈：
 - Pandora Boot
 - Spring Boot



Java 隔离容器 Pandora

Part 2

为什么需要隔离？

中间件相互冲突

中间件与应用
相互冲突

应用快速升级
发布的需求

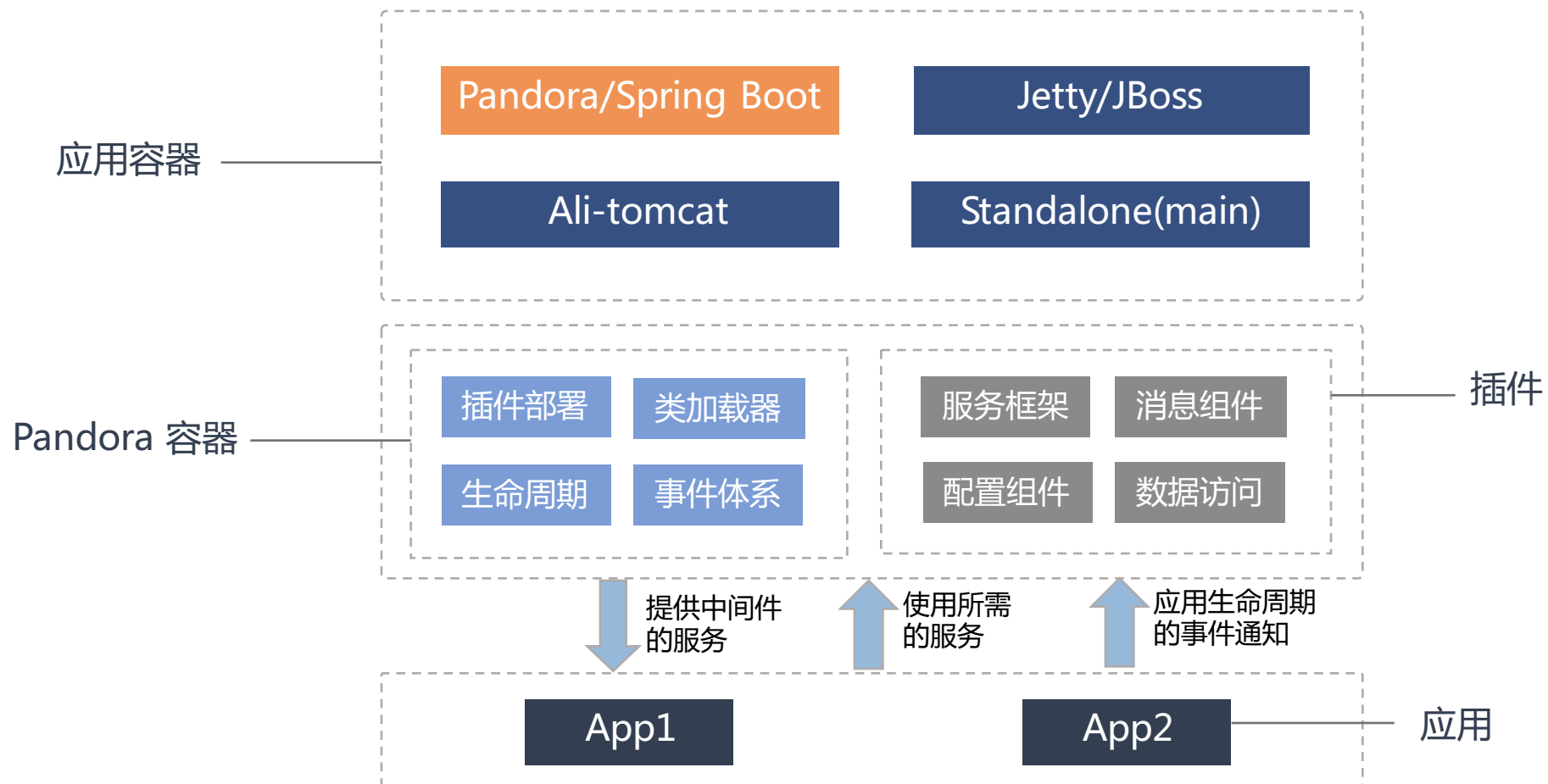
中间件统一升级
统一运维的需求



Pandora 容器架构

基于 ClassLoader实现

- 插件体系
- 生命周期
- 事件体系



Pandora 结构与部署形式

pandora.sar/		~/\${app_name}/
-- conf/		-- bin/
-- lib/	pandora 容器内部依赖	-- setenv.sh
-- pandora.api.jar		-- appctl.sh
-- pandora.archive.jar		`-- preload.sh
`-- pandora.container.jar		
-- plugins/	pandora 插件列表	-- target/
-- hsf/		-- \${app_name}.tgz
-- conf/		-- \${app_name}/
`-- lib/	hsf 插件内部依赖	-- pandora.tgz
-- tddl/		`-- pandora.sar/
-- conf/		
`-- lib/		

- 与应用 tgz 包部署在一起
- 应用可单独升级 pandora.sar
- 应用容器识别 -Dpandora.location , 便于本地开发

现有架构的不足



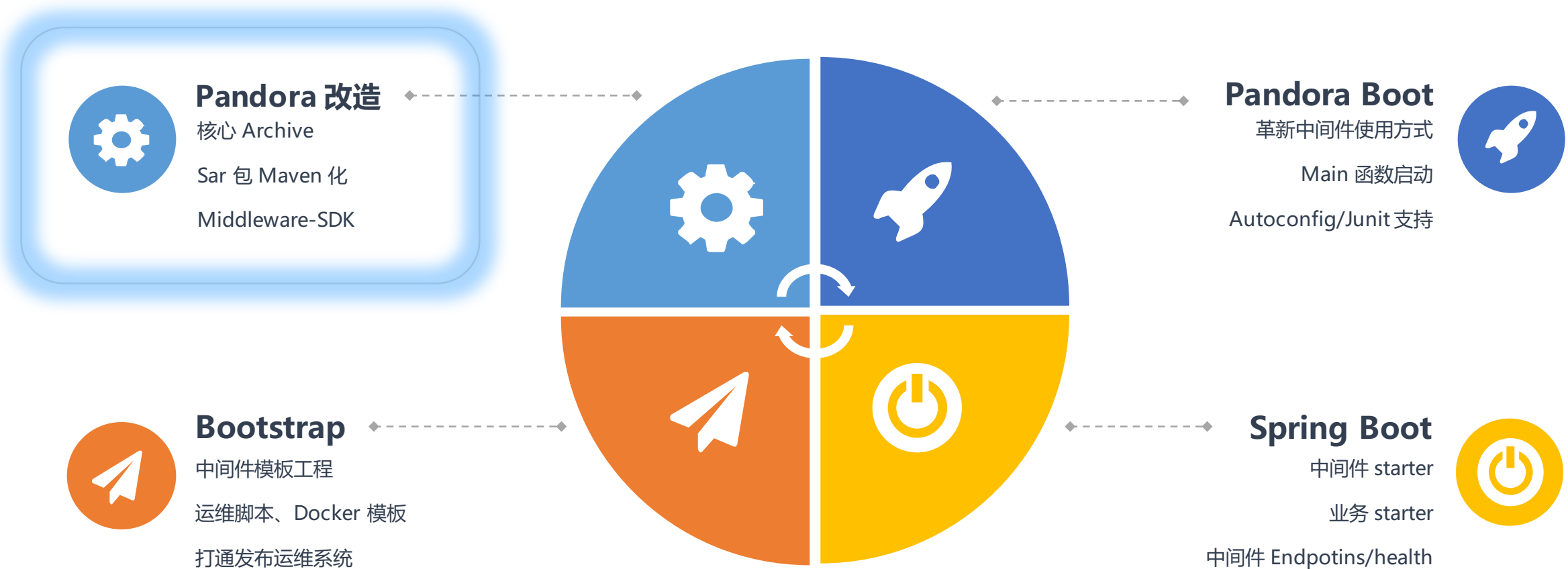
微服务框架 Pandora Boot

Part 3

开发人员的痛点



Pandora Boot 解决方案



Pandora 核心 Archive 化



Archive 抽象

- dir
- Jar
- Jar In Jar



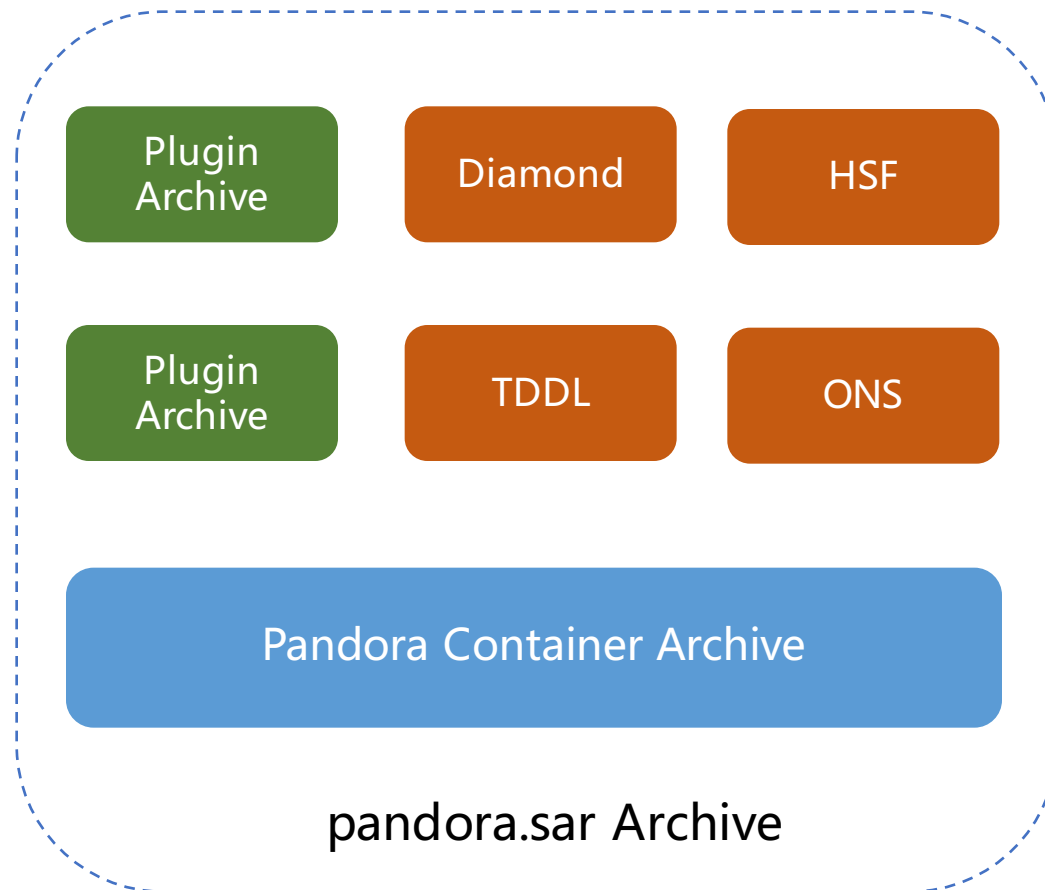
Jar Protocol 扩展 / Jar In Jar / Fat Jar

jar:file:/pandora.sar!/
jar:file:/pandora.sar!/container.jar!/
jar:file:/pandora.sar!/plugins/hsf-plugin.jar!/
整个 pandora.sar 是一个 Jar , 插件是一个 Jar



面向 URL 编程

从面向文件目录转为面向 URL
和标准的 URLClassLoader 对齐
从 URL 扫描 Pandora 插件启动



Pandora sar 包 maven 化



通过 maven 依赖 pandora.sar 包

```
<dependency>
  <groupId>com.taobao.pandora</groupId>
  <artifactId>pandora.sar</artifactId>
</dependency>
```



积木式组合 / 依赖即所得

```
<dependency>
  <groupId>com.alibaba.boot</groupId>
  <artifactId>pandora-hsf-spring-boot-starter</artifactId>
</dependency>
```



应用依赖无污染

PandoraClassLoader 才会识别加载
普通 ClassLoader 不会加载



Pandora middleware-sdk

maven 平板化依赖



依赖冲突严重

hsf/tddl/tair/ons , 60M , 196 jar



中间件功能不能保证

Netty/thrift/logger



大量答疑

排查问题先要排查依赖



难以升级

应用依赖固化 / 升级困难

VS

middleware-sdk



一个 maven 依赖

使用简单 , 4M



ASM字节码生成

新增插件 / API 无维护成本
只有导出的 API



解决 Debug 问题

sources.jar / UTF-8 无乱码



随 pandora.sar 包升级发布

保证API和sar包对应

自动生成 sdk.jar / sources.jar

```
public class HSFPackageScanner implements BeanFactoryPostProcessor {
    protected final static Logger logger = LoggerFactory.getLogger(HSFPackageScanner.class);
    public static final String PACKAGE_DELIMITER = ",";

    private String packageName;

    public void setPackageName(String packageName) {
        this.packageName = packageName;
    }

    @Override
    public void postProcessBeanFactory(ConfigurableListableBeanFactory beanFactory) throws BeansException {
        if (!(DefaultListableBeanFactory.class.isAssignableFrom(beanFactory))) {
            throw new RuntimeException("scanner not supported in this bean factory");
        }
        doScan((DefaultListableBeanFactory) beanFactory);
    }
}
```



保留 public/protected
删除 private field/method
函数抛出 RuntimeException

```
public class HSFPackageScanner implements BeanFactoryPostProcessor {
    protected final static Logger logger = LoggerFactory.getLogger(HSFPackageScanner.class);
    public static final String PACKAGE_DELIMITER = ",";

    public void setPackageName(String paramString) {
        throw new RuntimeException("Can not load this fake sdk class, please check your pandora env!");
    }

    public void postProcessBeanFactory(ConfigurableListableBeanFactory beanFactory) throws BeansException {
        throw new RuntimeException("Can not load this fake sdk class, please check your pandora env!");
    }
}
```

SDK生成

hsf plugin 60+ jar/ 22M

SDK源码



扫描所有的lib jar
只保留导出类
可选的关联导出类

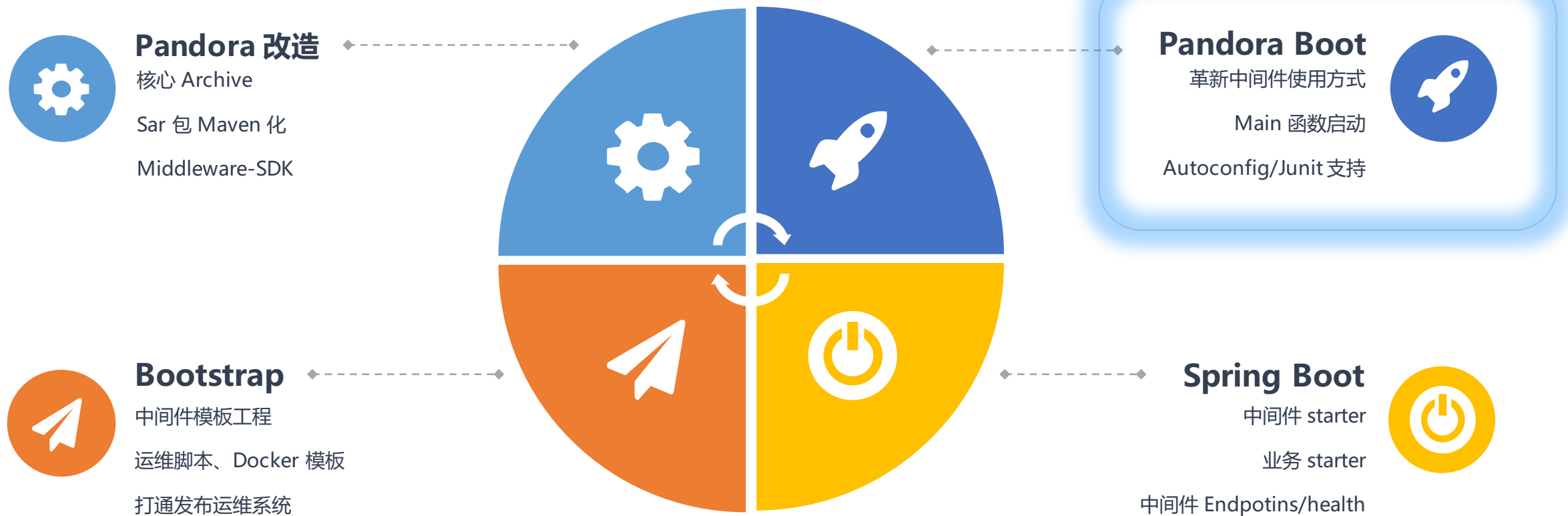
hsf-sdk.jar



通过maven自动生成
UTF-8编码
无乱码

hsf-sdk-sources.jar

Pandora Boot 解决方案







Pandora Boot 应用开发



```

6 public class ServerRun {
7
8   public static void main(String[] args) {
9     PandoraBootstrap.run(args);
10    GenericXmlApplicationContext ctx = new GenericXmlApplicationContext();
11    ctx.load("context-hsf-server.xml");
12    ctx.refresh();
13    PandoraBootstrap.markStartupAndWait();
14  }
15 }
  
```

 Run 'ServerRun.main()'  ^⬆R
 Debug 'ServerRun.main()'  ^⬆D

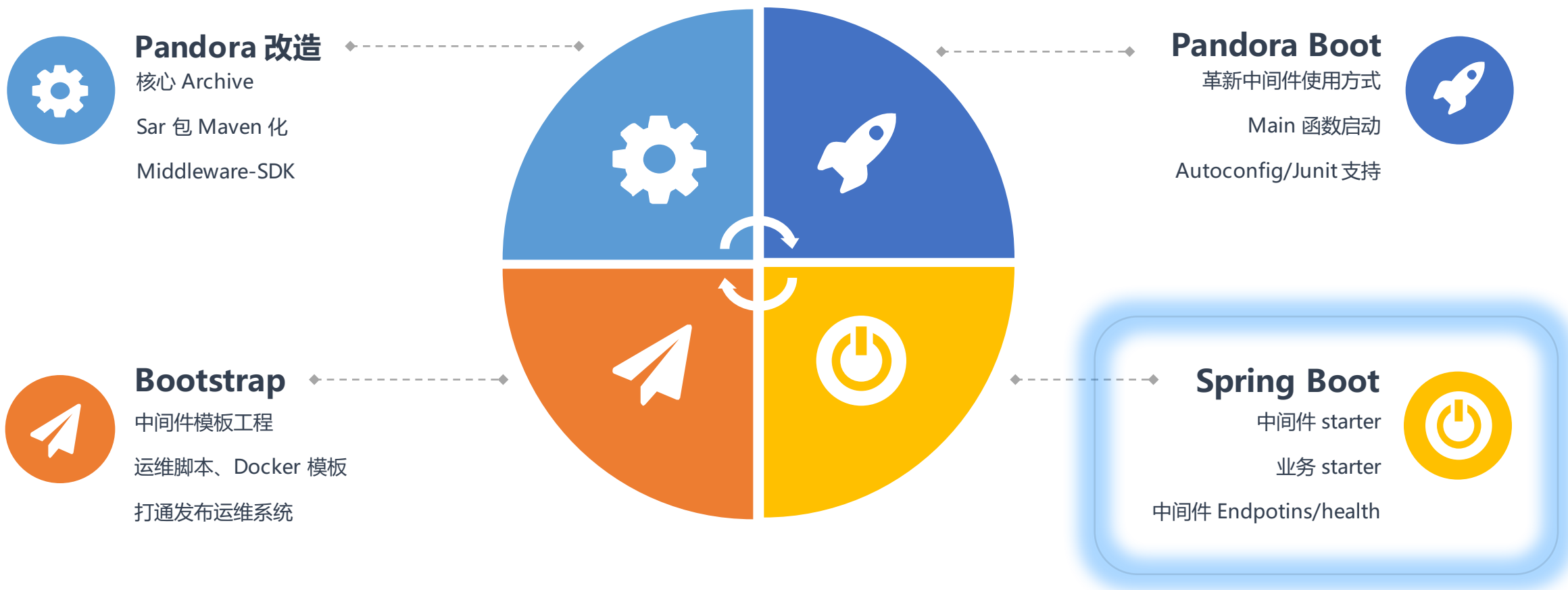
```

% java -jar hsf-server.jar
.....
Service(pandora boot) startup in 8484 ms
  
```

```

% mvn pandora-boot:run
.....
Service(pandora boot) startup in 12043 ms
  
```

Pandora Boot 解决方案



与 Spring Boot 集成

Spring Boot starter 生态在 Pandora Boot 上无缝使用



Pandora Boot 与 Spring Boot 无缝集成

```
@SpringBootApplication
public class Application {
    public static void main(String[] args) {
        PandoraBootstrap.run(args);
        SpringApplication.run(Application.class, args);
        PandoraBootstrap.markStartupAndWait();
    }
}
```



Pandora Boot 类隔离 + Spring Boot 快速开发

Spring Boot 的依赖是平板化的，难以解决依赖冲突的问题。Pandora Boot 提供类隔离技术，中间件与业务之间不会存在类冲突



Pandora Boot 提供完整的解决方案

Pandora Boot 是一套完整的解决方案，本地开发时无缝集成 autoconfig，日常、线上部署无缝打通内部发布、运维等系统

```
22 <dependencyManagement>
23 <dependencies>
24 <dependency>
25 <!-- Import dependency management from Spring Boot -->
26 <groupId>org.springframework.boot</groupId>
27 <artifactId>spring-boot-dependencies</artifactId>
28 <version>1.5.9.RELEASE</version>
29 <type>pom</type>
30 <scope>import</scope>
31 </dependency>
32
33 <dependency>
34 <groupId>com.taobao.pandora</groupId>
35 <artifactId>pandora-boot-starter-bom</artifactId>
36 <version>2018-01-release</version>
37 <type>pom</type>
38 <scope>import</scope>
39 </dependency>
40 </dependencies>
41 </dependencyManagement>
42 <dependencies>
43 <dependency>
44 <groupId>com.alibaba.boot</groupId>
45 <artifactId>pandora-hsf-spring-boot-starter</artifactId>
46 </dependency>
47
48 <dependency>
49 <groupId>com.taobao.pandora</groupId>
50 <artifactId>pandora-boot-test</artifactId>
51 <scope>test</scope>
52 </dependency>
53
54 <dependency>
55 <groupId>org.springframework</groupId>
56 <artifactId>spring-test</artifactId>
57 <scope>test</scope>
58 </dependency>
59
```

在 Spring Boot 中使用 HSF

使用 annotation 简单的发布、消费、测试 HSF 服务

```
@HSFProvider(serviceInterface=HelloWorldService)
public class HelloWorldService {
    @Override
    public String sayHello(String name) {
        return "Hello " + name;
    }
}
```

HSF 服务

@HSFProvider

@HSFProvider(serviceInterface=HelloWorldService.class) 标注需要暴露出去的远程服务

```
@SpringBootApplication
public class Application implements ApplicationRunner {
    @HSFConsumer
    HelloWorldService helloWorldService;

    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
        PandoraBootstrap.markStartupComplete();
    }
}
```

Spring Boot 自动装配

@HSFConsumer

@HSFConsumer 注入服务调用方
@SpringBootApplication 自动装配

```
@RunWith(PandoraBootRunner.class)
@DelegateTo(SpringJUnit4ClassRunner.class)
@SpringApplicationConfiguration(classes = ApplicationTest.class)
@Component
public class ApplicationTest {

    @HSFConsumer(generic = HelloWorldService.class)
    HelloWorldService helloWorldService;
}
```

单元测试

@RunWith

@RunWith(PandoraBootRunner.class) 进行基于 Pandora 的单元测试

中间件 Endpoints / Health

Endpoints

```
4 {
5   "tDataSources": [
6
7   ],
8   "unit": "CENTER",
9   "sequenceProperties": {
10    "type": null,
11    "table": "sequence",
12    "innerStep": 1000,
13    "adjust": false,
14    "nameColumn": "name",
15    "valueColumn": "value",
16    "gmtModifiedColumn": "gmt_modified"
17  },
18  "dataSourceProperties": {},
19  "version": "2018-01-release",
20  "properties": {
21    "app": "TEST_WEB_APP",
22    "groupKey": "TEST_WEB_GROUP",
23    "url": null,
24    "sharding": true,
25    "ruleFilePath": null,
26    "dynamicRule": null,
27    "healthCheckSql": null,
28    "apps": null,
29    "groupKeys": null,
30    "shardings": null,
31    "ruleFilePaths": null,
32    "urls": null,
33    "dynamicRules": null,
34    "healthCheckSqls": null,
35    "tddlProperties": {
36
37    }
38  }
39 }
```



运行时信息
环境信息
中间件内部信息
用户配置等

Health

```
4 {
5   "status": "UP",
6   "diskSpace": {
7     "status": "UP",
8     "total": 64424509440,
9     "free": 58640568320,
10    "threshold": 10485760
11  },
12  "db": {
13    "status": "UP",
14    "database": "MySQL",
15    "hello": 1
16  },
17  "tddl": {
18    "status": "UP"
19  }
20 }
```



运行时状态
应用 / 组件状态
中间件状态

Metrics

```
4 {
5   "data": {
6     "diamond-client": [
7       {
8         "interval": 15,
9         "metric": "middleware.diamond.defaultEnv.listenerSize",
10        "metricLevel": "NORMAL",
11        "metricType": "GAUGE",
12        "tags": {},
13        "timestamp": 1521535122383,
14        "value": 21
15      },
16      {
17        "interval": 15,
18        "metric": "middleware.diamond.envSize",
19        "metricLevel": "NORMAL",
20        "metricType": "GAUGE",
21        "tags": {},
22        "timestamp": 1521535122383,
23        "value": 1
24      },
25      {
26        "interval": 15,
27        "metric": "middleware.diamond.request.count",
28        "metricLevel": "NORMAL",
29        "metricType": "COUNTER",
30        "tags": {},
31        "timestamp": 1521535122383,
32        "value": 63550
33      }
34    ]
35  }
36 }
```



黑盒 -> 白盒
查看实时数据 curl
与外部对接查看历史数据

Pandora Boot starter 列表



中间件 18+

- ✓ hsf
- ✓ diamond
- ✓ Message
- ✓ Cache
- ✓ Data
- ✓ Monitor
- ✓ Metrics
- ✓ Scheduler
- ✓

业务类 10+

- ✓ 登录
- ✓ 会员
- ✓ ACL
- ✓ LOG
- ✓ 地址库
- ✓ 类目
- ✓

Pandora Boot 解决方案



Pandora 改造

核心 Archive

Sar 包 Maven 化

Middleware-SDK

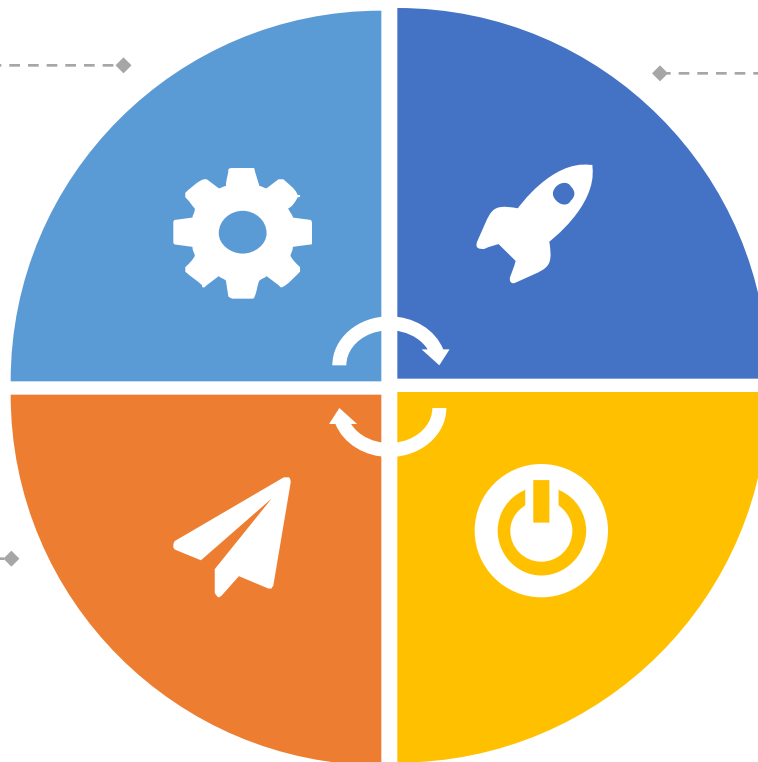


Bootstrap

中间件模板工程

运维脚本、Docker 模板

打通发布运维系统



Pandora Boot

革新中间件使用方式

Main 函数启动

Autoconfig/Junit 支持



Spring Boot

中间件 starter

业务 starter

中间件 Endpotins/health



Pandora Boot 模板工程 Bootstrap

Aone应用名

Group

Artifact

Java Version

应用类型 ☒ 微服务应用(Pandora Boot) ☒ 集成Spring Boot (推荐)
☐ 传统War应用

Docker模板 ☒ 是 ☐ 否

Web框架
请按需勾选 ☐ Spring MVC (推荐)
Spring Web框架

中间件
请按需勾选 ☐ HSF
阿里高性能服务框架
☐ Diamond
分布式持久配置系统

☐ TDDL
分库分表数据库服务
☐ Tair
分布式缓存

☐ ONS
分布式消息服务
☐ Notify
分布式消息服务



标准模板

避免复制旧代码，依赖干净
创建工程数 3W+



中间件示例

按需勾选，代码质量保证
最正确用法，持续更新



Release 相关

打包发布规范
脚本、健康检查等
可以直接发布，30分钟上线

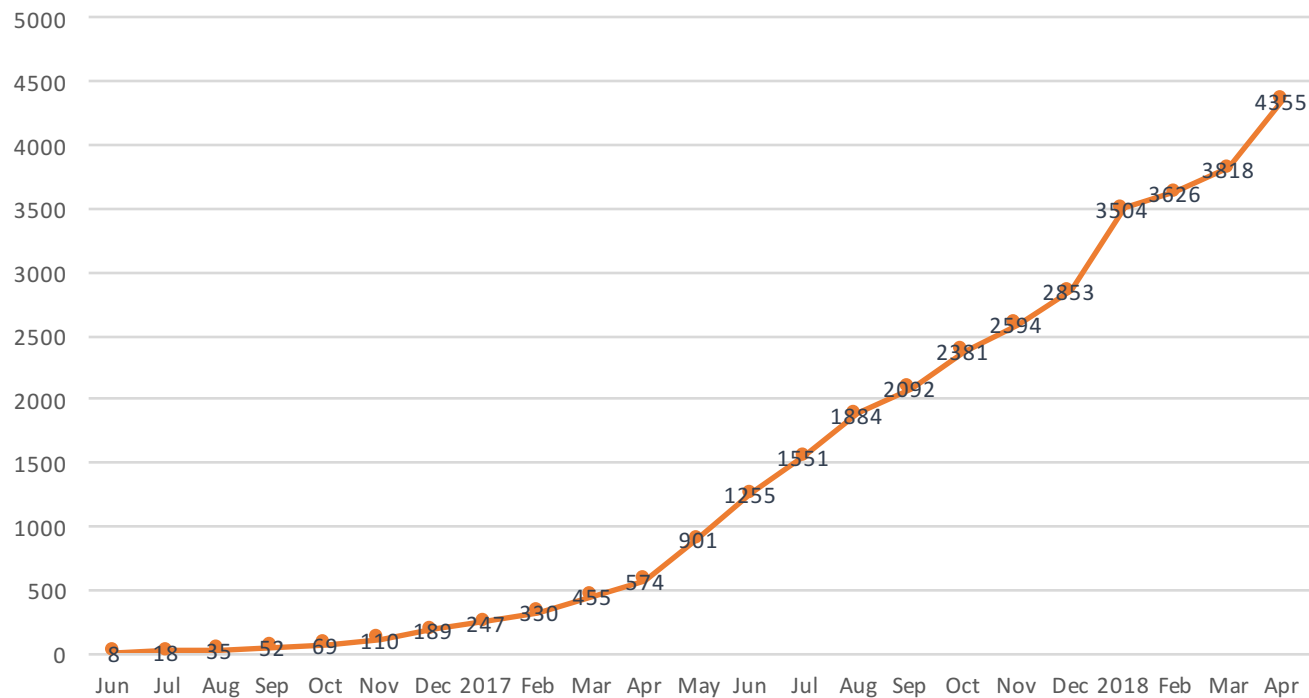
Pandora Boot 应用增长数



30% 全网占比

集团微服务开发框架标准
主打开发体验，成为业务架构升级的首选
年应用增长 10 倍

Pandora Boot 应用增长趋势



微服务运维及诊断

Part 4

微服务中心



能力透出

应用健康

应用资源

应用管控

应用诊断



平台能力

健康状态

资源状态

中间件管控

日志分析

环境配置

资源申请

容器管控

在线诊断



基础能力

Metrics

Health

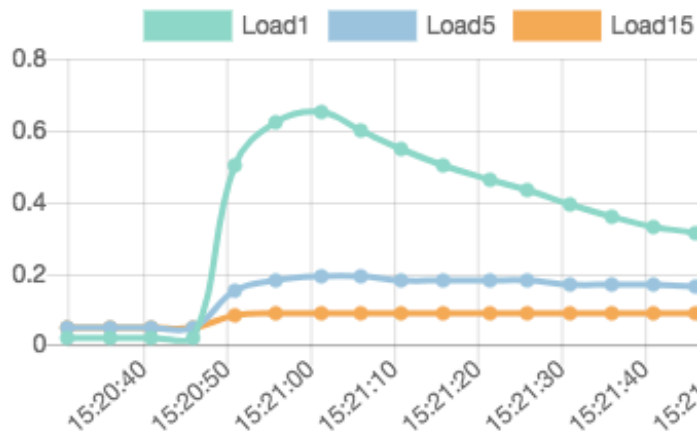
Config

JMX

Arthas

实时大盘

System Load & CPU



2.39%
user cpu%

1.45%
system cpu%

88.06%
idle cpu%

7.68%
steal cpu%

0.42%
iowait cpu%

采集时间: 2018-04-03 15:22:05

System Memory, Disk, and Network

Mem usage	Mem used	Mem total
29.9%	2G	8G
Disk usage	Disk free	Disk total
12.6%	52G	60G
Net in/s	Net out/s	Tcp retrans
3.2K	4.3K	1.3%

采集时间: 2018-04-03 15:21:31

JVM Heap

Eden	1.1G / 1.7G	Survivor	5M / 171M	Old	70M / 2.0G	Heap	1.2G / 3.8G
70%		0%		0%		30%	
Metaspace	134M / 512M	Code Cache	9M / 240M	CCS	17M / 1.0G	Non-Heap	161M / 1.7G
30%		0%		0%		10%	

JVM GC

YGC	FGC
0 +0	0 +0
YGCT (ms)	FGCT (ms)
0 +0	0 +0

线程堆栈

TOTAL: 124 RUNNABLE: 21 WAITING: 55 TIMED_WAITING: 48

122353	pool-2-thread-2759 main	cpu: 55%	RUNNABLE
163	com.taobao.diamond.client.Worker.default main	cpu: 21%	TIMED_WAITING
159	server-timer1 main	cpu: 4%	TIMED_WAITING
75317	nioEventLoopGroup-3-1 system	cpu: 3%	RUNNABLE
175	sso-client_timer main	cpu: 3%	TIMED_WAITING
182	Abandoned connection cleanup thread main	cpu: 3%	TIMED_WAITING
213	NioBlockingSelector.BlockPoller-2 main	cpu: 3%	RUNNABLE

在线诊断

诊断权限 已开启 Arthas Server状态 连接成功

停止诊断

彻底退出

关闭诊断权限

登录机器

ID	NAME	GROUP	PRIORITY	STATE	%CPU	TIME	INTERRUPTED	DAEMON
122381	Timer-for-arthas-dashboard-95a3be6	system	10	RUNNABLE	50	0:0	false	true
75319	pool-19-thread-1	system	5	WAITING	27	0:0	false	false
163	com.taobao.diamond.client.Worker.d	main	5	TIMED_WAIT	16	88:45	false	true
182	Abandoned connection cleanup threa	main	5	TIMED_WAIT	2	13:0	false	true
183	Abandoned connection cleanup threa	main	5	TIMED_WAIT	2	12:59	false	true
75313	AsyncAppender-Worker-arthas-cache.	system	9	WAITING	0	0:0	false	true
97	AsyncAppender-Worker-hsf.taobao.hs	main	5	WAITING	0	0:0	false	true
98	AsyncAppender-Worker-hsf.taobao.hs	main	5	WAITING	0	0:0	false	true
99	AsyncAppender-Worker-hsf.taobao.hs	main	5	WAITING	0	0:0	false	true
100	AsyncAppender-Worker-hsf.taobao.hs	main	5	WAITING	0	0:0	false	true
47	AsyncAppender-Worker-mtop.appkey_s	main	5	WAITING	0	0:0	false	true

Memory	used	total	max	usage	GC	
heap	1312M	3925M	3925M	33.44%	gc.parnew.count	489
par_eden_space	1237M	1706M	1706M	72.53%	gc.parnew.time(ms)	15328
par_survivor_space	5M	170M	170M	2.93%	gc.concurrentmarksweep.count	0
cms_old_gen	69M	2048M	2048M	3.40%	gc.concurrentmarksweep.time(ms)	0
nonheap	167M	212M	1776M	9.44%		

Runtime		Tomcat	
os.name	Linux	connector	http-nio-7002
os.version	3.10.0-327.ali2011.alios7.x86_64	QPS	NaN
java.version	1.8.0_112	RT(ms)	18.48
java.home	/opt/taobao/install/ajdk-8.3.6_fp	error/s	NaN
	2-b10/jre	received/s	0B
systemload.average	0.12	sent/s	0B

未来发展方向

Part 5

未来发展



01

Spring Boot 2.0

02

JDK9 Jigsaw

03

Serverless/RxJava

A solid orange vertical bar on the left side of the slide.

THANKS/感谢聆听

----- Q&A Section -----