



专注创新 化繁为简

# DolphinScheduler 与工业互联网平台xIn<sup>3</sup>Plat 牵手之路

刘慧娟 宝信工业互联网研究院/大数据中心

研发GPS

Make IT Simple

Make IT Possible

Make IT Great

01

## DolphinScheduler与工业互联网平台xIn3Plat牵手

- xIn3Plat介绍
- xIn3Plat对调度系统需求
- 调度系统选型

02

## 基于DolphinScheduler的功能扩展

- 插件类型任务
- 资源缓存
- SQL功能扩展
- 消息触发调度
- 多数据源接入
- 工作流并发控制
- 操作审计
- 告警优化
- 配置管理
- 权限控制
- 运行数据归档

03

## 近期规划

- 任务数据关联
- python功能增强
- 滚动升级
- 工作流运行隔离

# 工业互联网平台xIn3Plat介绍

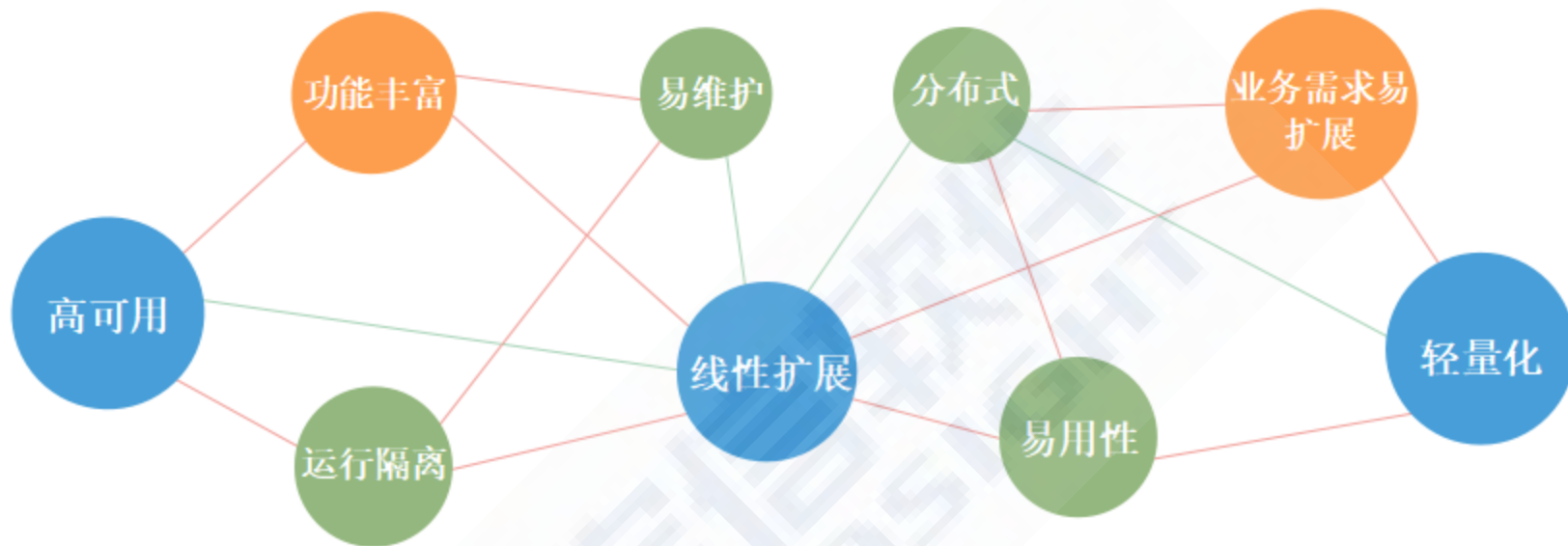
宝信软件基于多年深耕工业领域实践积累自研的工业互联网平台xIn3Plat，依托于大数据、区块链、人工智能、智能控制、工业安全、移动物联、虚拟仿真等七大核心技术，帮助企业实现能力服务化、业务数字化、企业平台化、管理智能化的企业数智化时代“新四化”转型。

5S大数据套件的研发致力于将大数据技术注入xIn3Plat，激发平台大智慧，赋能智能制造和智慧服务行业应用。





# 对调度系统的需求



## 高可靠性

调度节点高可用  
执行节点分布式



## 丰富易用

支持多种任务类型  
能够轻松定义复杂的任务依赖关系



## 轻量化

非常轻量，用于调度流程、启动任务等的开销非常小



## 业务隔离

支持业务独立的工作空间，不同的工作空间，相互之间不会干扰



## 调度性能线性扩展

调度器使用分布式调度，调度能力随集群规模增长而线性增长



## 业务易扩展

插件化，支持业务自定义插件扩展，可以实现新需求

		DolphinScheduler	oozie	azkaban	airflow
背景	组织	Apache孵化	Apache	Apache	Apache
	公司	易观	N/A	Linkedin	Airbnb
	社区活跃度	近期很活跃	一般	活跃	非常活跃
	软件开发语言	Java	Java	Java	Python
系统	分布式（执行节点）	✓	✓	✓	✓
	高可用（调度节点）	✓（多Master）	✓	✓	✓
	轻量化	✓	×	✓	✓
	过载处理	✓	✓	×	×
	线性扩展（执行节点）	✓	✓	✓	✓
	易维护（日志收集、手动调度、作业暂停/停止/恢复等）	✓	✓	不完善	不完善
用户	易用性（工作流定义）	图形界面	xml	自定义语言	python
	功能丰富	✓	×	✓	✓
	业务扩展性	一般	一般	较好	较好
	运行隔离	×	✓	×	部分支持

01

## DolphinScheduler与 工业互联网平台 xIn3Plat牵手

- xIn3Plat介绍
- xIn3Plat对调度系统需求
- 调度系统选型

02

## 基于DolphinScheduler 的功能扩展

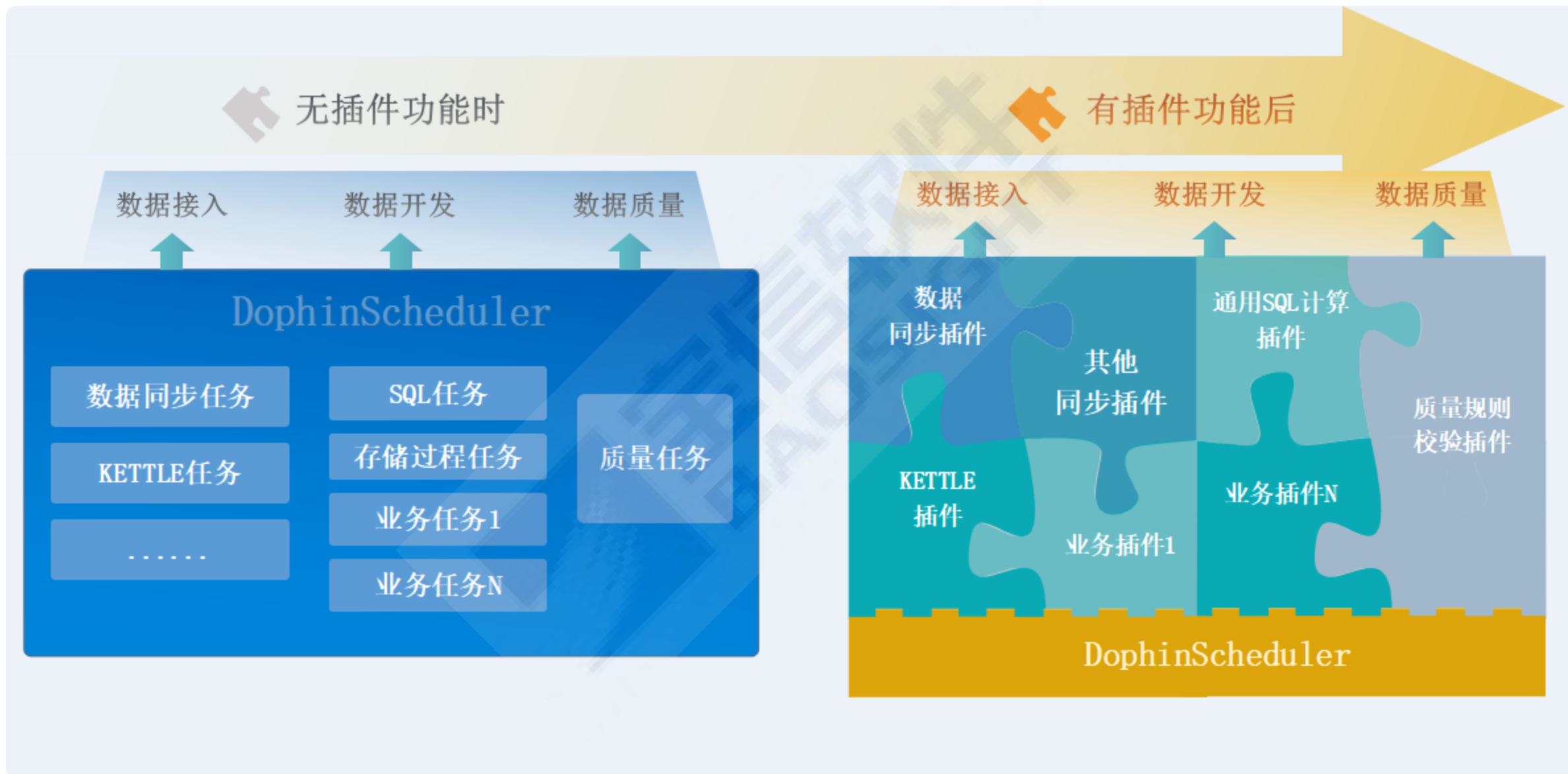
- 插件类型任务
- 资源缓存
- SQL功能扩展
- 消息触发调度
- 多数据源接入
- 工作流并发控制
- 操作审计
- 告警优化
- 配置管理
- 权限控制
- 运行数据归档

03

## 近期规划

- 任务数据关联
- python功能增强
- 滚动升级
- 工作流运行隔离

# 功能扩展1：插件功能



环境管理

插件管理

数据源管理

资源池管理

Worker分组管理

队列管理

资源池管理

## 插件

创建插件

编号	插件名称	资源名称	描述	插件配置
1	tss-extractor	tss-extractor-resource	tss extractor... <a href="#">点击查看</a>	
2	tss-sql	tss-sql-resource	tss sql plugi... <a href="#">点击查看</a>	

## 创建插件

\* 插件名称

tss-sql

资源名称

p\_tss-sql

文件名称

\_tss-sql-resource.zip

描述

SQL计算插件

插件配置

```
query_fetch_count=1000
insert_batch_count=1000
enable_batch_commit=false
```

\* 上传资源

上传

取消

创建



## 功能扩展2：资源缓存

A

资源文件大

B

资源被多任务引用

C

任务调度频率高

\* 插件名称

tss-stdkettle

资源

kjb (DS\_RS\_00\_00\_24eef681155043398be29bb6ca9d011f) ×

数据源

+

环境配置

test(TEST4) ▾

自定义参数

TSS_STD_KETTLE_LOCAL_PAR,	kettle/job-sts-load-test.kjb	🗑
table	gxyg02_test_01	🗑
schema	db10211	🗑

## 功能扩展2：资源缓存



### 场景

- 资源文件大
- 资源被多任务引用
- 任务调度频率高



### 问题

- 同一资源多次重复下载
- 下载资源耗时导致任务执行变慢
- 任务执行目录存储空间占用大
- 额外的清理工作
- 系统IO负荷高



### 方案

- 资源版本管理
- 资源版本检查
- 并发下载问题
- 资源压缩支持
- 资源引用
- 旧版本资源缓存清理

## 功能扩展3: SQL功能扩展

### 问题

- 不能支持多条SQL语句
- 不能调用存储过程
- 不能使用事务控制
- 不支持一些数据类型

### 功能

- 更好支持数据处理业务
- 存储过程管理
- 用户定义函数管理

```
CREATE OR REPLACE
FUNCTION TABLE_EXISTS ( SCHEMANAME VARCHAR(128),
TABLENAME VARCHAR(128) ) RETURNS INTEGER SPECIFIC SQL150416170754985 I
CASE
WHEN EXISTS (
SELECT
TABLENAME
FROM
SYSCAT.TABLES
WHERE
TABSCHEMA = SCHEMANAME
AND TABLENAME = TABLENAME
AND TYPE = 'T' WITH UR ) THEN 1
ELSE 0
END
--定义分隔符结尾
//
--提交事务
COMMIT;
--定义分隔符开始
delimiter //
--创建一个判断表是否存在，若存在，则删除该表的存储过程
CREATE OR REPLACE
PROCEDURE MYDROPTABLE ( IN para1 VARCHAR(50),
IN para2 VARCHAR(50) )
BEGIN
IF TABLE_EXISTS(para1,
para2) = 1 THEN PREPARE stmt
FROM
'DROP TABLE ' || para1 || '.' || para2;
EXECUTE stmt;
END IF;
END
--定义分隔符结尾
//
--提交事务
COMMIT;
--调用存储过程，若表DB2INST1.TEST存在，则删除该表
CALL MYDROPTABLE('DB2INST1', 'TEST');
--提交事务
COMMIT;
--创建表DB2INST1.TEST
```

### 01

#### 场景

- 不是按固定时间或周期调度，调度由事件触发
- 触发任务执行需要使用事件内容做参数

### 02

#### 方案

- 事件启动工作流的配置
- 从消息内容中获取工作流运行参数
- 在任务实例中引用参数
- 参数优先级

## 功能扩展5：多数据源接入



### 场景

- 多样化的数据源
- 历史悠久的业务系统  
(老版本)



### 问题

- 数据源之间依赖jar包冲突
- 数据源不提供JDBC驱动



### 方案

- 添加数据源子项目
- 按数据源类型 (无JDBC)  
功能封装
- 按数据源类型打包相关jar包
- 按需动态加载/卸载相关jar包



# 其他功能扩展

## workflow并发控制

- 同一工作流同一时刻只能有一个实例运行
- 按一定规则保证同一时刻只有一个实例运行

## 监控告警

- 告警信息需要能够方便运维定位问题
- 任务指标信息（eg：采集指标）
- 智能化监控（eg：根据历史运行情况判断）

## 环境配置管理

- 多种测试环境的任务上线至生产环境时，任务配置不变，但是引入的其它环境或服务信息要随之变更

## 权限控制

- 公共数据源
- 私有数据源按项目粒度控制权限
- 资源文件按项目粒度控制权限
- 环境配置管理

## 操作审计

- 用户的操作日志要有迹可循

## 运行数据归档

- 任务实例太多

## 工作流实例

名称

状态









选择开始日期

选择结束日期

Q

编号	工作流别名	工作流名称	工作流实例	运行类型	调度时间	开始时间	结束时间	host	状态	操作
1		citest-all-parallel	citest-all-parallel-0-1598720400206	调度执行	2020-08-30 01:00:00	2020-08-30 01:00:00	2020-08-30 01:24:45	168.2.8.200	成功	     

### 步骤实例

编号	步骤别名	步骤名称	工作流实例	节点类型	状态	提交时间	开始时间	结束时间	host	重试次数	操作
1				SUB_PROCESS	成功	2020-08-30 01:10:40	2020-08-30 01:10:42	2020-08-30 01:24:43	-	0	 
2				SHELL	成功	2020-08-30 01:00:38	2020-08-30 01:02:28	2020-08-30 01:10:32	168.2.8.200	0	 
3				SHELL	成功	2020-08-30 01:00:50	2020-08-30 01:02:28	2020-08-30 01:03:03	168.2.8.200	0	 
4				SHELL	成功	2020-08-30 01:00:13	2020-08-30 01:00:14	2020-08-30 01:15:30	168.2.8.200	0	 
5				SHELL	成功	2020-08-30 01:00:13	2020-08-30 01:00:14	2020-08-30 01:04:25	168.2.8.200	0	 
6				SHELL	成功	2020-08-30 01:00:13	2020-08-30 01:00:14	2020-08-30 01:05:26	168.2.8.200	0	 
7				SHELL	成功	2020-08-30 01:00:13	2020-08-30 01:00:14	2020-08-30 01:02:11	168.2.8.200	0	 
8				SHELL	成功	2020-08-30 01:00:13	2020-08-30 01:00:14	2020-08-30 01:00:38	168.2.8.200	0	 
9				SHELL	成功	2020-08-30 01:00:13	2020-08-30 01:00:14	2020-08-30 01:02:46	168.2.8.200	0	 
10				SHELL	成功	2020-08-30 01:00:13	2020-08-30 01:00:14	2020-08-30 01:01:31	168.2.8.200	0	 
11				SHELL	成功	2020-08-30 01:00:13	2020-08-30 01:00:14	2020-08-30 01:09:40	168.2.8.200	0	 
12				SHELL	成功	2020-08-30 01:00:12	2020-08-30 01:00:13	2020-08-30 01:06:28	168.2.8.200	0	 
13				SHELL	成功	2020-08-30 01:00:12	2020-08-30 01:00:13	2020-08-30 01:02:27	168.2.8.200	0	 
14				SHELL	成功	2020-08-30 01:00:12	2020-08-30 01:00:13	2020-08-30 01:06:30	168.2.8.200	0	 
15				SHELL	成功	2020-08-30 01:00:12	2020-08-30 01:00:13	2020-08-30 01:00:49	168.2.8.200	0	 
16				SHELL	成功	2020-08-30 01:00:12	2020-08-30 01:00:13	2020-08-30 01:00:58	168.2.8.200	0	 
17				SHELL	成功	2020-08-30 01:00:13	2020-08-30 01:00:13	2020-08-30 01:00:52	168.2.8.200	0	 



01

## DolphinScheduler与 工业互联网平台 xIn3Plat牵手

- xIn3Plat介绍
- xIn3Plat对调度系统需求
- 调度系统选型

02

## 基于DolphinScheduler 的功能扩展

- 插件类型任务
- 资源缓存
- SQL功能扩展
- 消息触发调度
- 多数据源接入
- 工作流并发控制
- 操作审计
- 告警优化
- 配置管理
- 权限控制
- 运行数据归档

03

## 近期规划

- 任务数据关联
- python功能增强
- 滚动升级
- 工作流运行隔离

# 未来规划：任务间数据依赖

1



## 任务间数据依赖

- 任务产生输出
- 任务引用其他任务的输出

2



## Python支持

- python版本问题
- 依赖包问题

3



## 滚动升级

- 升级造成业务暂停
- 手工补数复杂

4



## 运行隔离

- 按项目进行运行时隔离
- 按 workflow 进行运行时隔离





专注创新 化繁为简

# THANKS

[www.baosight.com](http://www.baosight.com)

上海宝信软件股份有限公司

中国（上海）自由贸易试验区郭守敬路515号

86-21-20378899 / 20379999



研发GPS  
Make IT Simple  
Make IT Possible  
Make IT Great