大数据计算引擎

Bonree Ants

**北京博睿宏远数据科技股份有限公司**

# 项目背景

目前越来越多的企业都认识到了大数据对于自身未来发展的重要性，于是纷纷开始使用并逐渐依赖大数据处理相关技术。但随着需要处理的数据越来越多，业务场景也越发复杂，在实际执行过程中也遇到了很多问题，如大数据人才紧缺、导致人力成本高，相关技术缺乏沉淀，短期内又难以培养一支较为成熟的大数据团队，同时不同业务部门的业务需求繁多且各异，导致不同项目代码重复开发、重复造轮子、项目技术架构五花八门的情况也较为常见，给后期维护和迭代带来很大的挑战。

随着大数据处理技术的发展和成熟，由于以上的现实问题，我们认为有必要基于博睿数据过往众多实际大数据项目经验，抽象和设计出一套灵活轻便、场景通用、稳定高效的统一大数据处理引擎框架来解决这些问题。因此，我们设计并推出Bonree Ants大数据处理引擎框架。

# Bonree Ants是什么？

Bonree Ants大数据处理引擎框架是博睿数据基于多年积累的流式大数据处理方面实际经验而设计的一个针对时序指标数据处理的一个通用流式计算引擎框架。此引擎框架负责范围包括原始数据预处理、准实时计算（支持默认算子和自定义算子）、多种时间粒度批量计算（支持天以内的自定义粒度计算）、数据落地及各种容错处理，同时支持高级功能自定义扩展机制，默认支持如时序指标动态基线计算、报警条件判断计算等扩展，不负责范围包括数据存储方案和在线查询支持。

Ants（蚁群）名称源于其寓意：高效协作、小而有力，我们认为其可以诠释我们设计这套引擎框架的初衷：轻便灵活、场景通用、稳定高效。

# 为什么要用Bonree Ants？

Bonree Ants可以帮助大数据技术积累较少的企业，或者项目周期和人力紧张的项目团队便捷快速的实现海量时序指标数据的在线流式处理。我们的目标是，对于常见的时序指标流式处理的业务场景可不需要研发人员参与，只需非研发业务人员对数据应用进行简单配置和业务脚本描述，即可实现其目标；而对于复杂业务场景我们则希望通过研发人员通过引擎的插件机制，进行少量编码来实现相关与业务强关联的逻辑，而将大数据处理中底层复杂的资源调度、任务编排、容错处理交给引擎负责，快速实现相关大数据处理业务开发，极大降低企业相关开发和维护成本。通过在博睿众多内部项目的实践，应用Bonree Ants引擎框架之后，大数据处理开发工作量整体降低了80%，整体项目周期缩短40%以上。

# 什么是Bonree Ants APP？

Bonree Ants将用户在kafka上的一个独立数据源视为一个数据应用APP。

数据源自身数据格式不限，由用户任意定义。但Bonree Ants要求发布到Kafka之上的message需进行统一格式的封装，并标识数据时间戳。具体格式大致如下所示：

{

"message": {

"ts": 13489811122,

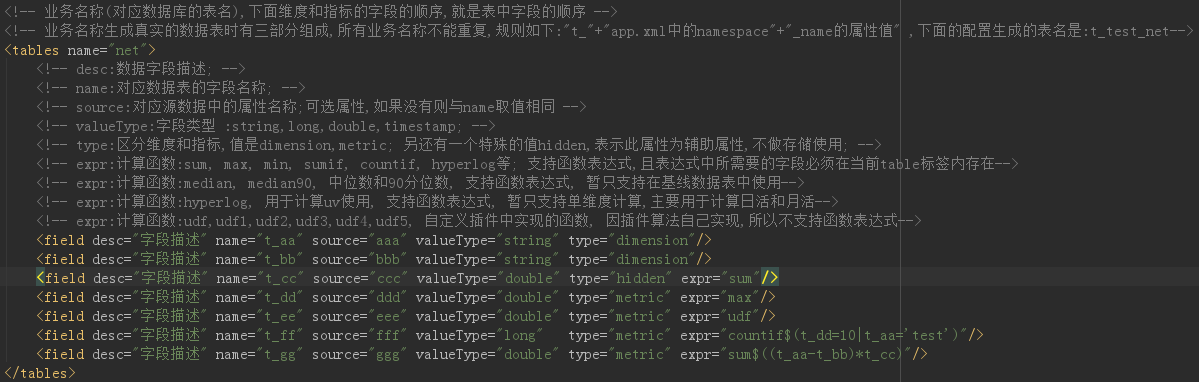
"data": "xxxxxxxxxxxx"

}

}

一般来说，一个Bonree Ants APP包含如下几个基本文件：

1. **App.xml**：用于配置基础依赖集群地址和各计算拓扑运行时的关键控制参数。
2. **Schema.xml：**用于配置数据源中各业务数据具体处理规则，所有数据指标和维度的运算处理规则都由此文件进行描述，如下图示例所示：

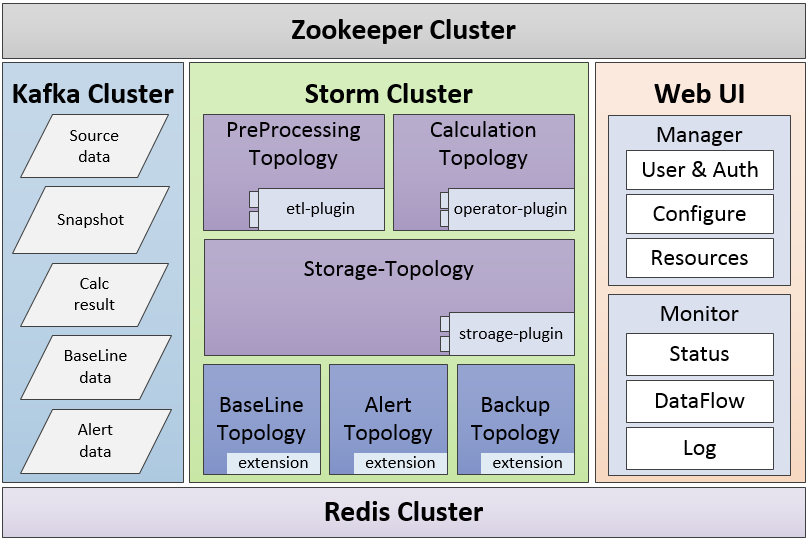


1. **Etl-plugin.jar**：数据预处理插件。由用户开发，运行在数据预处理拓扑中，负责对每一条原始数据执行具体的清洗策略；
2. **Operator-plugin.jar**（可选）：自定义算子插件。由用户开发，运行在数据计算拓扑中，负责实现用户对数据指标和维度处理所需的自定义算子；
3. **Storage-plugin.jar**（可选）：数据处理结果落盘策略插件。由用户开发，运行在数据存储拓扑中，负责实现数据处理结果具体的落地入库操作；
4. Ants引擎自带框架可执行Jar文件；

Bonree Ants的众多APP可共享同一组基础集群运行，引擎框架会以命名空间（默认以APPName为命名空间名称）的方式对各APP的资源进行彼此隔离。

# Bonree Ants的整体架构

Bonree Ants引擎框架整体架构如图所示：



（**Bonree Ants整体架构图**）

Bonree Ants引擎框架依赖如下开源组件运行：ZooKeeper、Kafka、Storm、Redis，依赖少，部署简单，架构轻便，也是其核心优势之一。

Bonree Ants底层计算框架基于Storm，之所以选择Storm作为底层计算框架，主要考虑到Storm自身的实时性高、资源开销小，外部依赖少、纯内存计算、容错性好等特性。Bonree Ants将时序指标大数据处理过程抽象为以下几个主要流程，即：预处理、准实时计算、小批量计算、大批量计算、落地入库等5个流程。以上流程均由运行于Storm之上的Preprocessing Topology、Calculation Topology、Storage Topology三类拓扑负责完成。

Bonree Ants整个数据处理过程中数据不落地，均在内存中完成。由于需支持大时间粒度批量计算业务场景，Bonree Ants引入Redis作为Storm集群辅助内存存储集群，以降低大批量计算时对Storm集群内存的开销。由于采用内存计算的方式，Bonree Ants实时性高，对磁盘I/O几乎无影响，但这也带来一定的数据处理吞吐量上的牺牲。

Bonree Ants批量计算是基于时间粒度的聚合计算，对于天粒度以内的聚合计算支持自定义配置，默认支持1秒、1分钟、10分钟、1小时、1天等五个不同时间粒度的聚合。由于各粒度计算之间存在递进依赖关系，因此为减少计算资源开销，加速计算过程，Bonree Ants在Redis集群缓存了各粒度中间计算结果，以备下一时间粒度计算直接使用，从而减少数据处理量级。

Bonree Ants对于最终计算结果落地只提供了基本框架支持。由于不负责最终数据存储，因此也不会对最终数据落地存储组件有任何限制。Bonree Ants默认内置支持Mysql存储方案，如采用Mysql作为最终落地数据库，则建表和结果数据入库过程均可由Bonree Ants自动完成。如需采用其他落地方案，如HBase等，则由用户开发Storage-Plugin插件进行自定义支持。

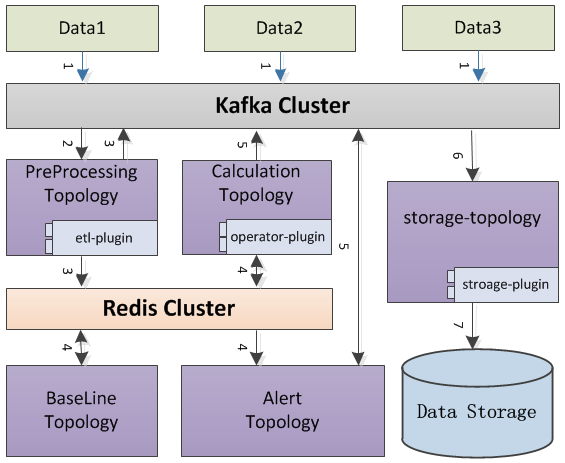
Bonree Ants整体架构设计大量采用插件（plugins）和扩展（extensions）机制，其将与业务强关联的个性化处理，如数据预处理策略抽象为Etl-plugin，将数据和维度指标处理的算子（内置支持sum、max、min等基本算子）开放为Operator-plugin，而将处理结果落盘策略抽象为Storage-plugin。

同时，在支持插件技术外，为了丰富Bonree Ants引擎框架功能，还支持extensions扩展机制。用户可以在现有引擎框架基础上开发自己需要的extension，并以独立计算拓扑的方式提交给Ants，由于Ants加载运行，从而实现Ants功能的延伸。目前，Bonree Ants默认内置动态基线扩展（base-line extension）和报警条件判断扩展（alter-judge extension）。

此外，Bonree Ants还支持**插件**动态更新和S**chema.xml**动态更新功能，这样能够帮助用户实现不重启即可更新业务处理逻辑的支持，对用户复杂多变的业务场景带来极大的灵活性。

# Bonree Ants的运行机制

Bonree Ants内部数据流转如下图所示：



（**Bonree Ants运行机制图**）

* **数据预处理拓扑(PreProcessing Topology)**

此拓扑负责从kafka订阅原始数据源并调用etl-plugin插件对数据进行预处理（etl-plugin插件由客户自己来实现），并对将etl后的结果进行准实时聚合处理（计算规则由用户在schema.xml中描述）。

如果用户想要备份etl后的原始明细数据，则只需在Schema.xml中开启相关配置，则由此拓扑在etl后把数据镜像一份发往kafka集群指定topic，由用户自行消费进行后续备份处理。

* **数据计算拓扑(Calculation Topology)**

此拓扑负责对etl之后的数据进行按不同时间粒度的批量聚合(规则由Schema.xml描述)。此过程内部有两个子计算拓扑：小批量计算（分钟级）， 大批量计算（小时级和天级）。计算过程中小粒度的中间计算结果都会缓存到redis集群中，供下一大粒度计算使用。同时，各粒度计算结果会落地到kafka相应的topic，由“数据存储拓扑”订阅进行后续存储操作，从而实现数据计算和落地之间的解耦。

* **数据存储拓扑(Storage Topology)**

此拓扑负责数据落地入库，落地数据包括两种，即时序指标数据（结构化）和快照文件数据（非结构化，如业务中存在此类数据的话）两种，用户需根据自身业务需要开发Storage-plugin来实现具体的落盘策略。

除此之外，Bonree Ants默认实现了两个重要的extension扩展，即基线计算extension和报警条件判断extension。

# Bonree Ants可视化管理和监控

Bonree Ants提供强大的在线运行支持，可支持可视化管理和监控，具体功能包括如下：

1、APP管理

APP在线创建、Schema在线配置、APP发布上线、下线；

2、基础环境监控

zookeeper、storm、kafka、redis等基础集群运行状态监控；

3、日志监控

APP运行关键流程节点状态日志收集和监控；

# Bonree Ants优点及不足

## 优势

1、简洁开放的架构，较少的组件依赖，开发部署及维护成本低；

2、引擎框架自身与业务无耦合，数据处理流程高度抽象，通用性强；

3、秒级时延，实时性好，同时内置批量计算支持；

4、支持extension机制，用户可自行丰富业务场景功能支持；

5、内置多种容错策略，保证稳定与数据安全；

6、支持可视化管理和监控；

## 缺点

1、完全基于内存进行计算，吞吐量不如传统批量计算框架，如Spark、MapReduce等；

1. 目前聚合的时间粒度无法支持天粒度以上更大粒度聚合；
2. 对MQ的支持仅限于Kafka，后期可考虑支持其他MQ；
3. 仅适合做结构化时序指标数据处理，对其他如非结构化大数据处理场景并不支持；

# 怎样使用Bonree Ants？

1. 安装部署基础环境，主要包括Zookeeper、Redis（需支持cluster版本）、Storm、Kafka等基础依赖组件；
2. 修改APP配置文件App.xml和编写业务描述文件Schema.xml；
3. 开发相关插件，主要是etl-plugin和storage-plugin的开发；
4. 将配置文件和插件可执行Jar文件进行提交试运行，并调试直至功能稳定；
5. 提交正式环境上线运行；