SOFAStack

文件批处理
技术白皮书

产品版本：AntStack Plus 1.11.0

文档版本：20220929

蚂蚁集团
ANT GROUP

法律声明

蚂蚁集团版权所有©2022，并保留一切权利。

商标声明

免责声明

通用约定

格式	说明	样例
危险	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	危险 重置操作将丢失用户配置数据。
警告	该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。	警告 重启操作将导致业务中断，恢复业务时间约十分钟。
注意	用于警示信息、补充说明等，是用户必须了解的内容。	注意 权重设置为0，该服务器不会再接受新请求。
说明	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	说明 您也可以通过按Ctrl+A选中全部文件。
>	多级菜单递进。	单击**设置**> **网络**> **设置网络类型**。
粗体	表示按键、菜单、页面名称等UI元素。	在**结果确认**页面，单击**确定**。
Courier字体	命令或代码。	执行 `cd /d C:/window` 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	`bae log list --instanceid` *Instance_ID*
[] 或者 [a|b]	表示可选项，至多选择一个。	`ipconfig [-all|-t]`
{} 或者 {a|b}	表示必选项，至多选择一个。	`switch {active|stand}`

目录

1.什么是文件批处理

文件批处理（SOFA Batch）是蚂蚁集团推出的结合文件处理能力和任务调度能力的一站式文件批处理平台，可以对文件进行解析、拆分、合并，同时提供金融行业常见文件的处理模板，具备可复用、可扩展的文件处理能力。能规范化、自动化、可视化、集中化地对金融企业不同业务系统的文件批处理任务进行统一调度和全方位监控运维管理，为用户提供联机与离线互转的成熟解决方案。

概述

文件批处理平台 = 文件工厂（Affilefactory）+ 任务调度（Task Scheduler）。

文件批处理平台提供基于分布式架构的文件处理能力，能够对各种格式和存储介质的文件进行解析、拆分、合并，提供与文件读写、文件大小无关的文件传输和文件加工能力。结合分布式任务调度框架，可以实现对批处理文件任务的统一调度和全方位监控运维管理。

任务类型

文件批处理支持以下多种文件任务类型，以满足不同场景的文件任务处理需求：

- **文件搬运任务**

  指将源存储介质中的文件复制到目标存储介质中，支持在相同或不同的存储介质（OSS、SFTP）之间移动文件。您可以通过搬运任务，让其他系统来处理或备份文件，适用于简单的文件传输场景。

- **文件解析任务**

  指将 TXT、CSV 文件的内容按照预先配置好的模板（文件描述）解析成数据对象。支持对目标文件进行自定义维度分片（目前支持一层拆分），业务客户端拉取分片后，基于文件模版进行解析，所有分片解析完成后，进行汇总。当数据量较大时，可以充分利用业务方集群资源，通过并行处理减少耗时。

- **文件组装任务**

  指将小文件或数据组合成一个大文件，根据组装任务处理的客户端数量可分为简单组装任务和集群组装任务。可以模板化配置生成的文件、对外部不同格式的文件进行解析，并按照内部统一标准重新组装。

  - 简单组装任务

    适用于仅包含单个客户端的组装任务，可以直接将单个客户端的数据转成大文件，因此只包括文件写入逻辑。

  - 集群组装任务

    适用于包含多个客户端的组装任务，可以在单个客户端各自处理数据或小文件后，将处理好的文件组装成大文件。支持自定义维度分片，客户端拉取分片后，基于分片信息适配不同的组装器执行具体的小文件写入逻辑，所有分片写入完成后进行组装。

2.产品优势

一站式文件处理能力

提供文件批处理常用的成熟模式，包括文件的拆分、解析、组装等。支持对外部不同格式的文件进行解析，并按照内部统一标准重新组装。能规范化、自动化、可视化、集中化的对金融企业不同业务系统的批处理任务进行统一调度和全方位监控运维管理，实现一站式文件批处理能力。

高时效

支持在不完全下载原文件、不进行文件切片、不再次上传切片文件和不删除临时文件的前提下进行文件的拆分解析，降低文件切割的耗时，从而大幅降低文件解析过程的整体耗时，提高任务处理速率。

金融级高可用

支持机房级容灾，分布式模式保证了当一台机器宕机时，受到影响的任务数量有限，任何单机或单机房故障都不影响当前任务的执行，保障业务的正常执行，达到高可用的目的。

高可用高扩展

支持无限水平扩展，无性能容量瓶颈，监控任务的执行情况，可以及时发现执行异常或没有执行的任务，通过可配置的异常处理策略对异常任务进行补偿。

高性能

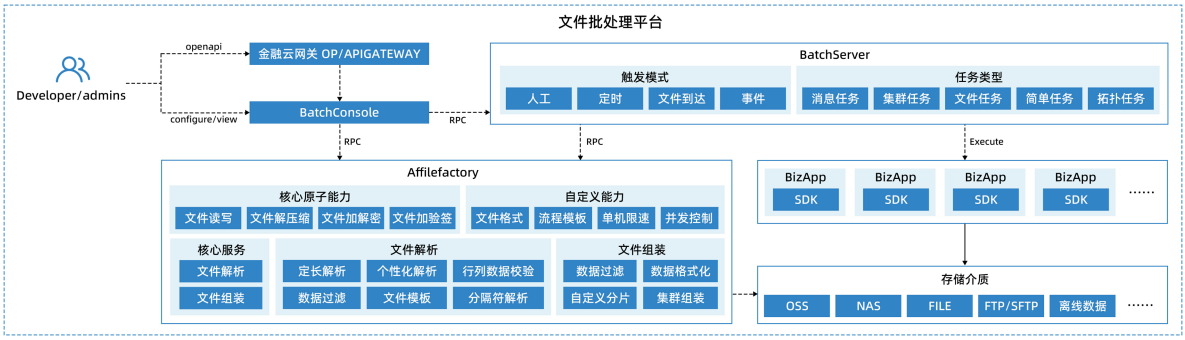
通过分布式以及优化的底层架构，支持多层调度模式进行无限拆分，实现多线程并行处理，显著提升了大数据量的批任务处理的性能。

可视化集中式管理

通过简易操作的可视化集中式管理平台可对上万个任务节点进行集中化管理，以简化运维管理操作，提高处理效率。

3.产品架构

系统架构



文件批处理由以下四个部分组成：

- **BatchConsole**：可视化集中管理平台，方便管理批处理任务，简化运维操作。
- **BatchServer**：批处理服务端，提供任务调度和集群间任务分配等功能。
- **Affilefactory**：文件工厂，提供文件拆分、解析、搬运以及组装等通用功能。
- **Client**：客户端，业务应用通过集成 Client，获取批处理任务调度能力。

文件批处理还需要有以下外部依赖：

DB：负责存储任务元数据、文件存储介质元数据、文件模版元数据以及执行记录等信息，并作为批处理集群注册表。

文件批处理任务调度的执行过程如下：

1. 添加文件存储介质、文件模版等元数据。
2. 通过 BatchConsole 添加任务。
3. 当任务满足执行条件时，批处理服务端会触发任务调度，根据任务类型获取对应的调度执行器执行。
4. 调度执行器会通过事件方式创建触发记录、更新触发记录状态。
5. 调度器通过通讯模块通知客户端执行，RPC 采用 CallBack 模式下，客户端会将执行结果告诉服务端。服务端再根据事件模式更新任务执行状态。
6. 监听器监听漏触发和超时的执行记录，分别根据漏触发策略和超时策略进行处理。

分布式部署

批处理平台集成了文件批处理功能的任务调度系统。通过分布式模式可以提高可用性、水平拓展能力。结合 LDC 模式，文件批处理实现了 zone 级别的容灾能力。文件批处理平台通过以下方式完成分布式部署：

- **xxj-job 分布式模式**

  依赖 quartz 的分布式管理，通过 quartz 管理任务的分配。批处理平台使用 quartz master/slave 架构，由 master 通过 Hash 算法将任务分配到各个机器上，并且定期检查 slave 状态。 slave 状态异常时会把这台机器上的任务分配给其他机器来处理，这样就保证了当一台机器宕机时，受到影响的任务数量有限，达到高可用的目的。

- **Elastic-Job 分布式模式**

  采用 ZK 选主，通过 master 分配任务。批处理平台的选举模式依赖数据库，通过选举得到的 master 负责整个集群的任务分配和容灾。

4.性能指标

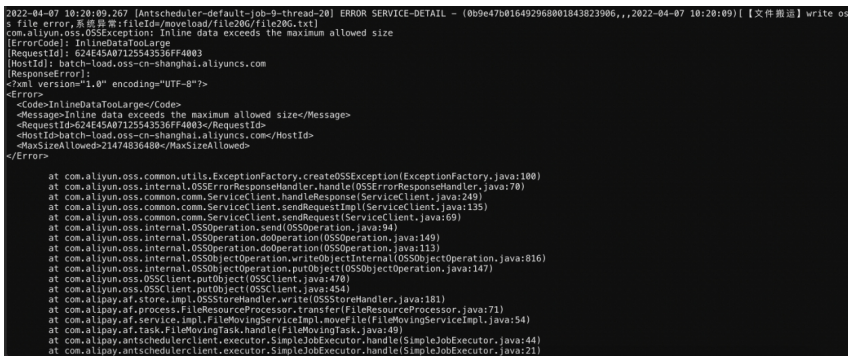
不同业务场景，不同机器配置、性能不一样。性能数据可以参考以下压测报告。

压测场景一：文件搬运任务

- **测试目的**

  测试文件搬运任务的耗时。

- **服务器规格**
  - 文件批处理服务器：2C4GB * 2
  - 调度服务器：2C4GB * 2
- **配置说明**
  - 通知任务使用默认执行线程池，增加耗时挡板，模拟耗时 10ms。
  - 测试场景：文件从 OSS 介质搬运到 OSS 介质。
- **测试结果**

文件大小	测试次数	最低耗时 /s	最高耗时 /s	平均耗时 /s	失败率	设置超时时间 /hour	机器 CPU 用量
1GB	10	19.409	76	55.1	0%	1	
2GB	10	71	117	97.9	0%	1	
5GB	10	174	308	257.1	0%	1	
10GB	10	351	575	438.1	0%	1	
20GB	10	OSS 报错如下： 					

- **结论**

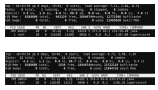
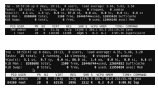
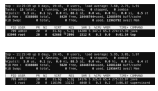
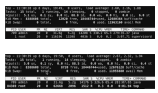
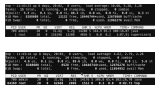
- 文件搬运任务的耗时随文件大小增大而增加，对文件批处理的应用性能消耗较小。
- 当搬运的文件大小为 20GB 时，OSS 出现报错。

压测场景二：文件解析任务

- **时长测试**
  - **测试目的**

    测试文件解析任务的耗时。
  - **服务器规格**
    - 文件批处理服务器：2C4GB * 2
    - 调度服务器：2C4GB * 2
    - 客户端服务器：4C8GB * 2
  - **配置说明**

    通知任务使用默认执行线程池，增加耗时挡板，模拟耗时 10ms。
  - **测试结果**

压测内容	分片长度	拆分耗时 /s	解析耗时 /s	总耗时 /s	客户端机器 CPU 数据
1GB 大小的文件	16140486（70 个分片）	7.08	42.371	71	
2GB 大小的文件	22309824 (100 个分片)	14.42	75	110	
5GB 大小的文件	21914702 (250 个分片)	59.082	165	283	
10GB 大小的文件	19790345 (550 个分片)	95	332	470	
20GB 大小的文件	18087932 (1200 个分片)	154	656	896	

  - **结论**

    对比一期压测最优分片长度，二期压测分片数据解析时客户端负载较高。

- **单机与分布式**
  - **测试目的**

    测试单机与分布式场景下文件解析任务的耗时。
  - **服务器规格**
    - 文件批处理服务器：2C4GB * 2
    - 调度服务器：2C4GB * 2
  - **配置说明**

    通知任务使用默认执行线程池，增加耗时挡板，模拟耗时 10ms。
  - **测试结果**

文件大小	分片长度	客户端机器配置	拆分耗时	解析耗时	总耗时	客户端机器
5GB	21914702 (250 个分片)	4C8GB * 1	39.741	386	451	[redacted]
		4C8GB * 2	37.03	192	255	[redacted]
		4C8GB * 4	39.59	104	179	[redacted]

  - **结论**

    文件解析任务的耗时随着客户端机器的增多缩短，符合预期。

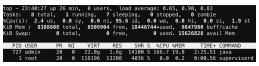
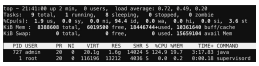
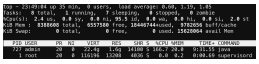
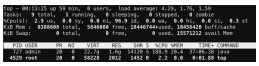
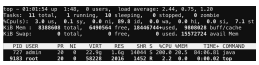
压测场景三：简单文件组装任务

- **测试目的**

  测试在不同文件大小的情况下，执行简单文件组装任务的性能。
- **服务器规格**
  - 文件批处理服务器：2C4GB * 4
  - 调度服务器：2C4GB * 2
  - 客户端服务器：4C8GB * 1
- **配置说明**

  增加 5ms 耗时挡板。

- **测试结果**

文件大小	测试次数	耗时	客户端机器
100MB	1	25 秒	
500MB	1	123 秒	
1GB	1	243 秒	
5GB	1	19 分钟 4 秒	
10GB	1	39 分钟 7 秒	
20GB	1	1 小时 26 分钟 42 秒	

- **结论**
  - 随着文件大小增大，简单文件组装任务的耗时增加，基本成线性增长。
  - 文件组装任务执行时，客户端机器负载较低。

压测场景四：集群文件组装任务

- **测试目的**

  测试不同数据量的情况下，集群组装任务的性能。

- **服务器规格**
  - 文件批处理服务器：2C4GB * 4
  - 调度服务器：2C4GB * 2
  - 客户端服务器：4C8GB * 2

- **配置说明**
  - 集群组装时重置线程池大小为 50 个线程，2 个节点共 100 个线程。
  - 文件解析阶段 Chunk 执行采用异步执行模式。
  - 组装方式为 mock 数据拼装。

- **测试数据**

  文件组装任务耗时如下：

压测内容	分片数量 / 个	拆分耗时 /s	写耗时 /s	大文件合并耗时 /s	总耗时 /s
5 千条数据	5	0.131	7.792	0.709	9.103
1 万条数据	10	0.086	6.647	1.327	8.564
10 万条数据	10	0.1	19.121	1.897	21.248
100 万条数据	200	0.214	66	23.322	89
500 万条数据	800	0.642	202	100	303

- **结论**

  对比一期压测最优分片数量，二期集群文件组装任务的耗时随文件数据量增加而增加。

5.功能原理

5.1. 文件搬运任务

搬运任务指将文件从源存储介质移动到目标存储介质，同时支持文件的加密、解密、压缩、解压缩操作。具体逻辑由文件批处理平台完成，使用时需要提供源文件和目标文件的存储介质及文件完整路径。

5.2. 文件解析任务

文件解析任务包括拆分阶段、解析阶段、分片结果通知阶段、汇总阶段四个阶段。

拆分阶段

拆分阶段分为 chunk 拆分、chunk 上报、chunk 入库三个步骤。

Chunk 拆分

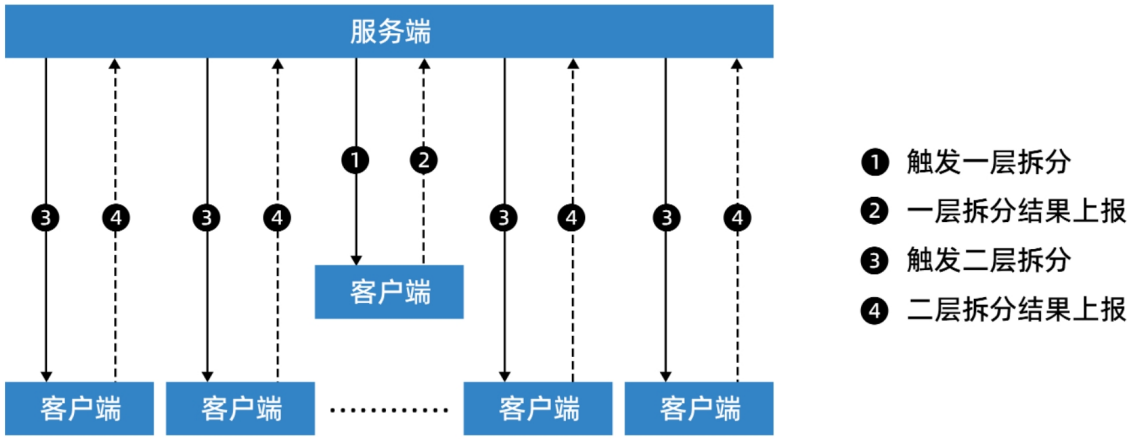
拆分逻辑为·：框架将拆分的数据封装为对象，上报到服务端并存储到数据库中。目前只有一层拆分，对应的 IChunkData 类型为 ShardingChunkData，该 chunk 只包含一个分片规则。IChunkData 支持存储一些自定义属性到 extensionMap，这些数据会一起存储到数据库中，大小限制为 1024 个字符。

Chunk 上报

Chunk 数据的长度为 10*1000 个字符，超出长度后会休眠 3s 后再继续上报。若客户端未上报完信息时发生宕机，服务端可以感知到客户端的长连接断开，会再选择一台客户端重新进行拆分。

Chunk 入库

服务端在收到上报的 chunk 信息后，将数据入库。若服务端未把 chunk 的完整信息入库时发生宕机，会自动分配一台服务端重新触发拆分动作。

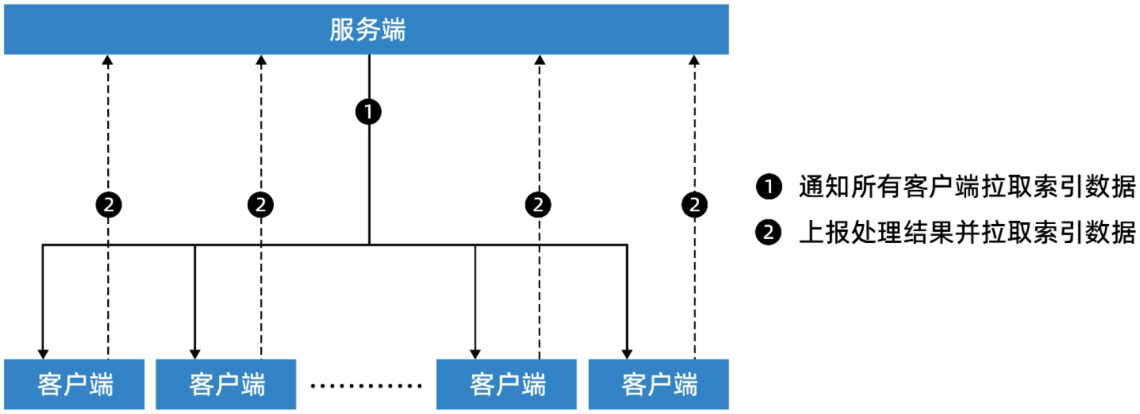


解析阶段

解析阶段按照预先定义好的文件模版对源文件的头、体、尾进行解析，分为拉取 chunk 和执行 chunk 两个步骤。

拉取 Chunk

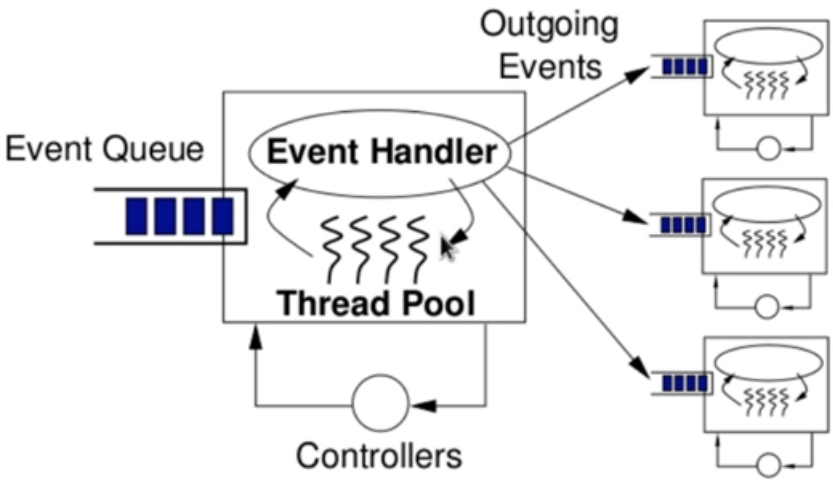
当数据拆分完毕后，服务端会通知客户端集群的所有机器来服务端拉取 chunk 信息，当拉取到的 chunk 处理完后，客户端会将结果上报到服务端，并且继续拉取新的 chunk。



Chunk 执行

chunk 的执行分为三个子阶段：Read、Process 和 Write，类似于 SpringBatch，执行逻辑非常契合 SEDA（阶段性事件驱动）架构思想，因此执行阶段采用如下所示的 SEDA 架构进行了实现。SEDA 模型有以下特点：

- 将一个任务分成了多个阶段，每个阶段拥有一个独立的事件缓冲队列和线程池，各个阶段之间通过事件进行通信。
- 整合了多线程的服务器模型和事件驱动的服务器模型的优势，可以高效地管理和控制服务器资源，良好地适应高并发环境。

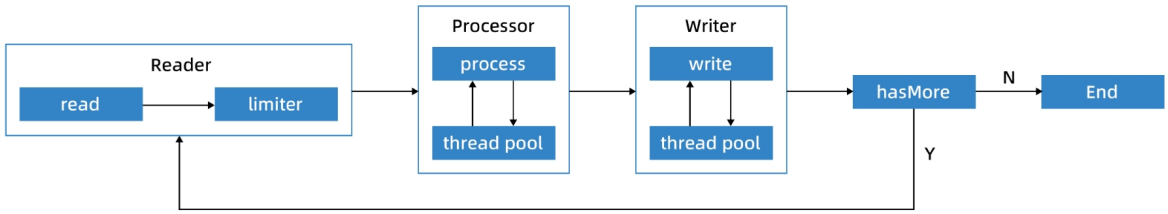


Chunk 执行模式

Chunk 执行模式有同步模式和异步模式两种。

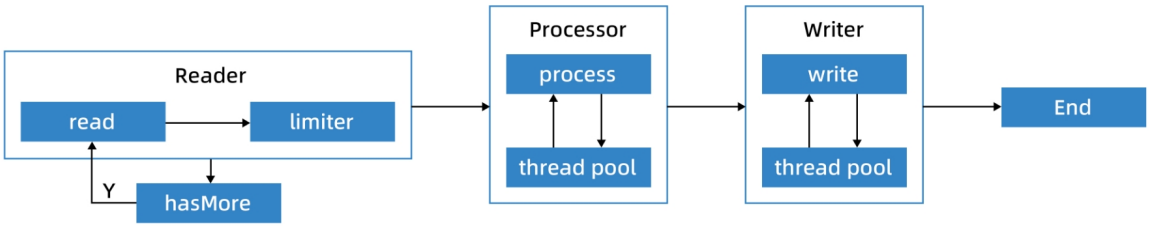
同步模式

该模式下，需要在 read、write 完成之后，再开始下一次 read 操作。



异步模式

该模式下，read 的数据全部放置到队列后即可开始下一次 read 操作，不需要等待 write 的执行结果。适用于读操作比较耗时的场景，可以显著提高处理性能。



分片结果通知阶段

分片结果通知阶段将具体的分片解析结果落入文件工厂数据库中，同样执行解析阶段的拉取 Chunk 步骤，基本流程与解析阶段一样。

汇总阶段

汇总阶段基于简单任务实现，文件工厂根据解析结果执行具体的汇总逻辑，将汇总结果返回。

5.3. 文件组装任务

文件组装任务是指将小文件或数据组合成一个大文件。可以模板化配置生成的文件、对外部不同格式的文件进行解析，并按照内部统一标准重新组装。

简单组装任务

简单组装任务适用于仅包含单个客户端的组装任务，可直接将单个客户端的数据转成大文件，因此只包括文件写入逻辑。

集群组装任务

集群组装任务适用于包含多个客户端的组装任务，可以在多个客户端各自处理自身数据或小文件后，将处理好的文件组装成大文件。执行过程分为以下几个步骤：

1. 单个客户端对目标文件进行拆分，将拆分结果上报批处理服务端。
2. 客户端拉取分片，根据分片信息适配不同的组装器执行具体的小文件写入逻辑。
3. 所有分片执行完成后，文件工厂基于简单任务执行最终的组装合并逻辑。

6.单元化能力

在单元化架构下，批处理平台可以识别到任务客户端所在的单元信息，指定单元进行任务触发。 具体架构实现流程如下：

1. 客户端连接所有的 server，注册连接时携带单元信息。
2. 任务触发时，会根据启动的单元信息找到对应的连接，触发任务。