Kafka在工业大数据管理中的应用
Kafka Meetup Beijing

K²DATA 昆仑数据
释 放 机 器 数 据 价 值

K2Data - 专注工业大数据

– 数据：文本、图片、影像、用户浏览记录、在线交易、企业信息等

– 典型应用：搜索、用户行为分析、舆情分析，社交网络分析

Palantir

cloudera
Ask Bigger Questions

– 数据： IT 数据、日志

– 典型应用：App管理，信息基础建设和运营，合规安全

splunk>

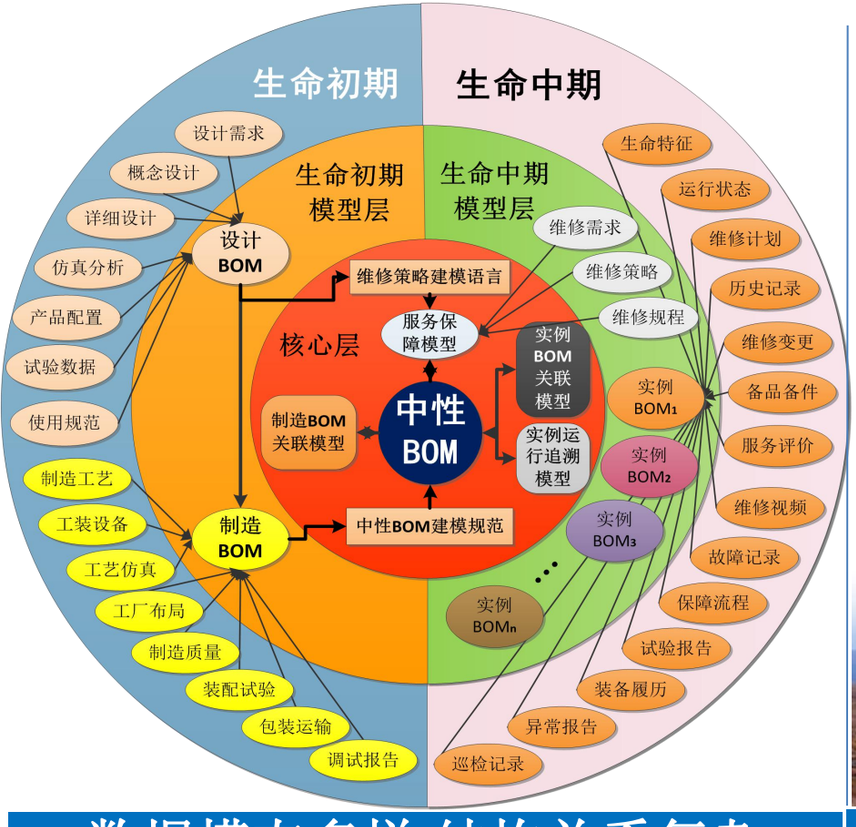
– 数据：传感器、智能装备、科学仪器数据

– 典型应用：装备运维、智能工厂

K²Data

工业大数据的来源和特点

多模态，高通量，强关联



数据模态多样,结构关系复杂

典型高端制造企业数据类型可达300余种,汽轮机35万个零部件数据

海量高速

机器24*7产生，采集频率高，数据量大

数据通量大

50Hz，500测点/台，2万台风机，最高可达数千万数据点/秒

多学科异构数据信息交互模型

协作专业多

飞行器研发相关专业200多类

工业信息化数据　　机器数据　　产业链跨界数据

工业大数据的业务目标

加法

提质增效

乘法

如何在供应链与我的供应商进行更有效的协同

减法

降低成本

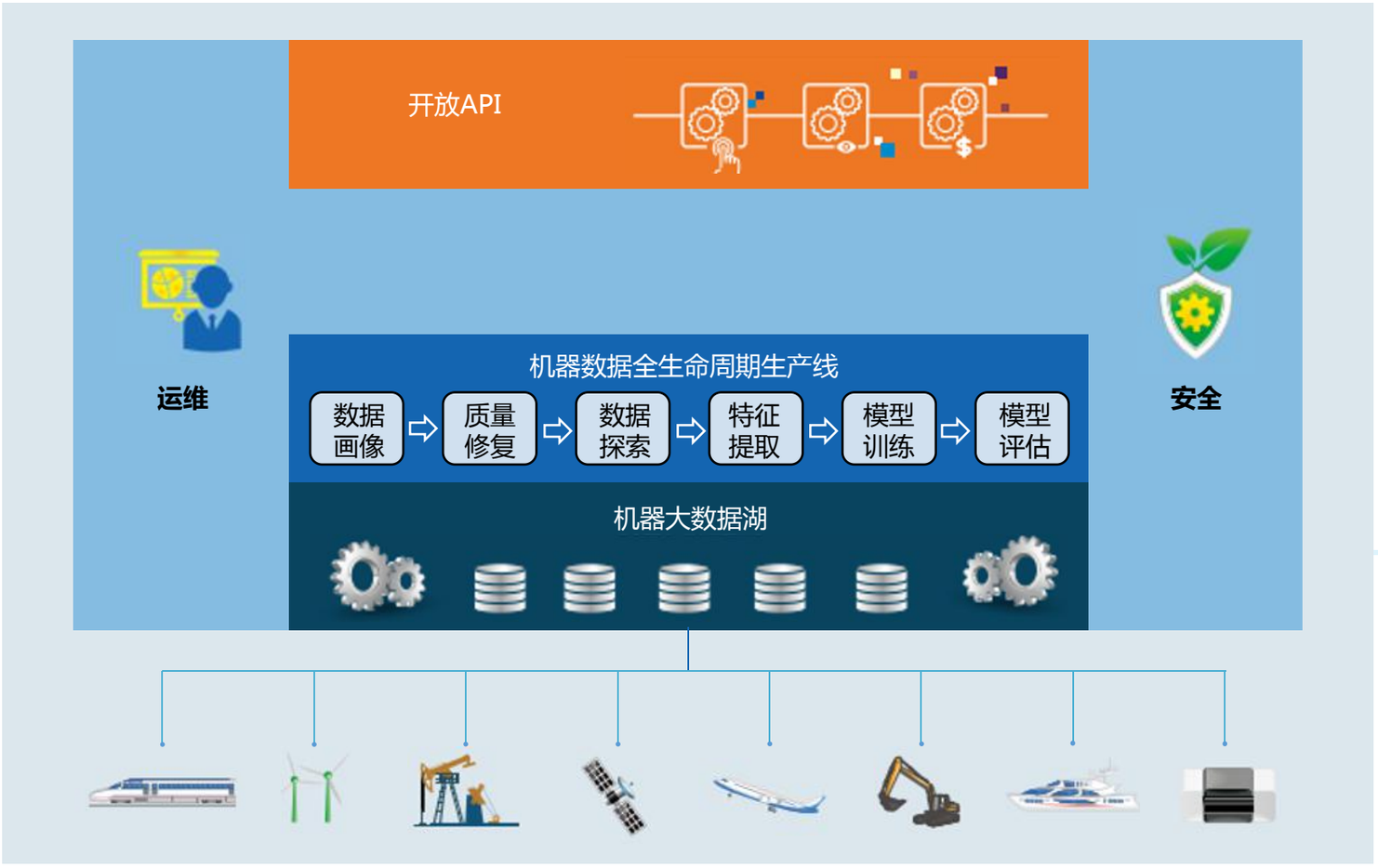
降低次品

降低能耗

除法

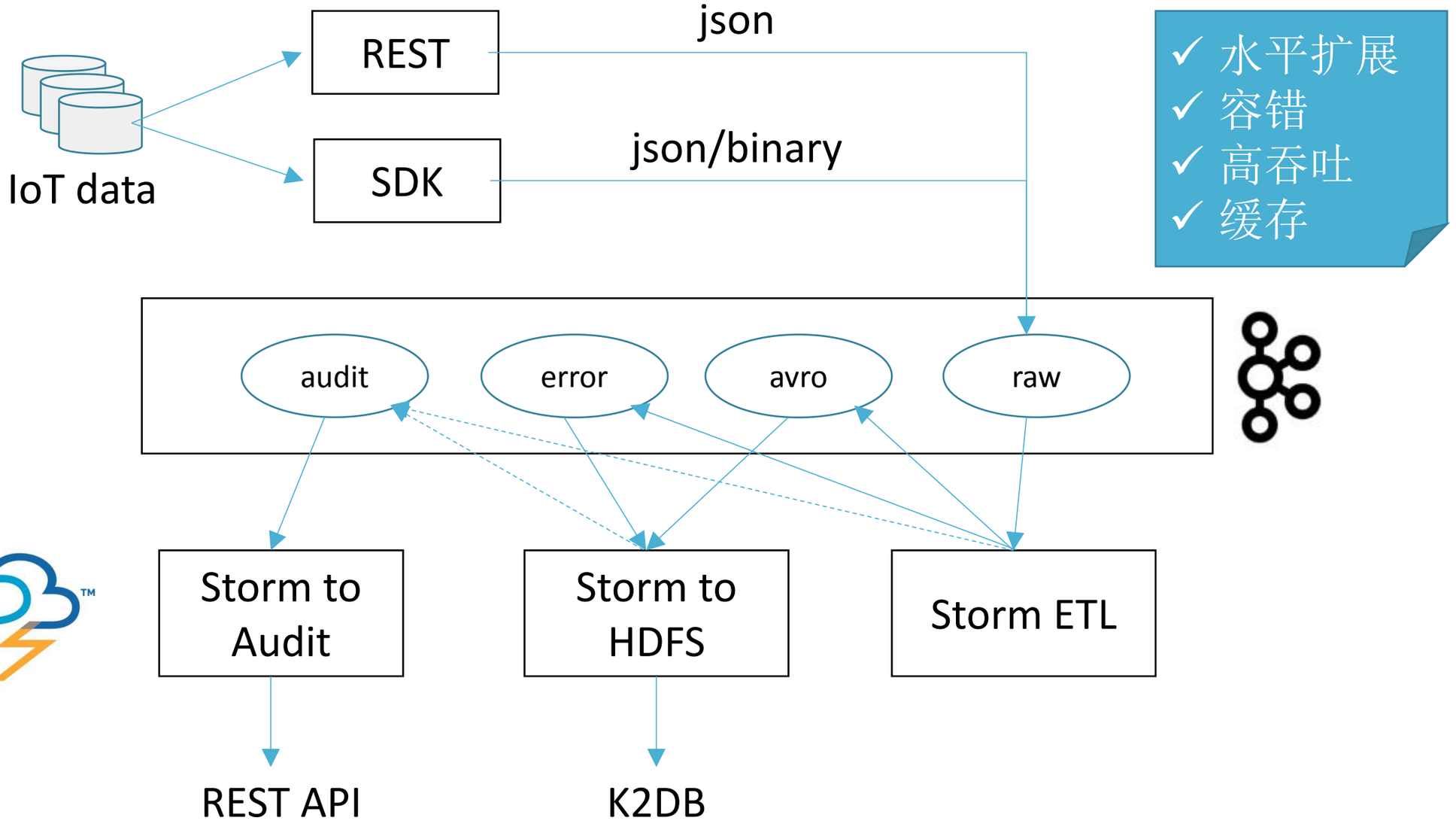
如何在供应链上进行分工，如何实现更轻资产的运营

KMX - 工业大数据平台产品



- 机器数据高效接入和管理
- 内置工业数据查询引擎
- 分析查询一体化
- 服务化的数据访问与分析

Kafka在工业数据接入的应用



实验一

- 实验目的：通过实验验证KafkaSpout能否实时消费掉写入Kafka的消息，以此作为后续设置Kafka日志定期清除机制的前置条件。
- 实验方法：通过KafkaProducer API向kafka中全速写入（消息之间没有sleep）最大消息体（max.message.bytes），利用KafkaSpout构建Topology，利用LocalCluster本地运行消费kafka实时写入的消息。
- 实验环境：单机 Intel(R) Core(TM) i7-4790 CPU @ 3.60GHz　　mem: 31GiB　　SATA 7200rps

```
NetHogs version 0.8.0

  PID USER     PROGRAM                                          DEV        SENT      RECEIVED
23000 pc       /usr/lib/jvm/jdk1.7.0_80/bin/java                lo       67.392   54157.672 KB/sec
24387 pc       /usr/lib/jvm/jdk1.7.0_80/bin/java                lo    94768.031      89.434 KB/sec
 3351 pc       /usr/lib/jvm/java-7-openjdk-amd64/jre/bin/java   lo        0.000       0.000 KB/sec
24400 pc       /usr/lib/jvm/jdk1.7.0_80/bin/java                lo        0.000       0.000 KB/sec
 3282 pc       /usr/lib/jvm/java-7-openjdk-amd64/bin/java       lo        0.000       0.000 KB/sec
26096 pc       java                                             lo        0.135       0.000 KB/sec
?     root     unknown TCP                                               0.000       0.000 KB/sec

  TOTAL                                                                 94835.558   54247.106 KB/sec
```

```
NetHogs version 0.8.0

  PID USER     PROGRAM                                          DEV        SENT      RECEIVED
 4753 pc       /usr/lib/jvm/jdk1.7.0_80/bin/java                lo      122.449   75027.500 KB/sec
 7526 pc       /usr/lib/jvm/jdk1.7.0_80/bin/java                lo    72530.211      66.950 KB/sec
 3351 pc       /usr/lib/jvm/java-7-openjdk-amd64/jre/bin/java   lo        0.000       0.000 KB/sec
 7544 pc       /usr/lib/jvm/jdk1.7.0_80/bin/java                lo        0.000       0.000 KB/sec
 3282 pc       /usr/lib/jvm/java-7-openjdk-amd64/bin/java       lo        0.000       0.000 KB/sec
26096 pc       java                                             lo        0.135       0.000 KB/sec
?     root     unknown TCP                                               0.000       0.000 KB/sec

  TOTAL                                                                 72652.795   75094.450 KB/sec
```

config.setDebug(false)

实验二

- 实验目的：研究在不同replication-factor情况下kafka写入每条消息的耗时，以此推算出kafka的写入速度（基于kafka 2.11-0.10.0.1）。
- 实验方法：
  - 为了避免网络不稳定影响测试结果，我们本地搭建kafka集群(4个broker)；
  - 从本地台式机发送kafka数据，payload约为18 KB；
  - 创建不同replication-factor的topic，分别发送不同规模的message，统计耗时，计算平均处理时间；
  - 本地发送程序直接调用kafkaProducer原生api，单个producer实例，单线程调用，循环中没有sleep；
- 实验环境：单机 Intel(R) Core(TM) i7-4790 CPU @ 3.60GHz　　mem: 31GiB　　SATA 7200rps

replication-factor	#message	time (ms, 测试三组)	timePerMsg (ms)	bandWidth (send)	result
partitions=4 replication-factor=1	10000	3936/3179/2625	0.3246		每条消息写入耗时约为：0.12 ms 每秒写入消息数：1000 / 0.12 = 8333 吞吐量：8333 × 18 KB = 150 MB/s
	100000	12005/11358/13966	0.12443		
	1000000	129301			
partitions=4 replication-factor=2	100000	31268/24078/22255	0.25867		每条消息写入耗时约为：0.26 ms 每秒写入消息数：1000 / 0.26 = 3846 吞吐量：3846 × 18 KB = 70 MB/s
partitions=4 replication-factor=3	100000	36291/35195/35478	0.35654		每条消息写入耗时约为：0.35 ms 每秒写入消息数：1000 / 0.35 = 2857 吞吐量：2857 × 18 KB = 50 MB/s

实验三：破坏性实验

结论：为了保证broker变化过程中没有数据丢失，需要满足如下两个条件：

- ✓ broker变化的数目小于topic的replication-factor值;
- ✓ 设定producer的retries参数大于0。

实验四：Storm ETL性能

方法：单机运行Storm ETL拓扑，依次添加KafkaSpout、TransformBolt和KafkaBolt，并变换它们的并行度，使用本地运行AdapterTopology的方式消费集群中的积攒数据。

数据：测试中使用的数据payload信息如下：fg×10，每个fg中包含的asset×100，6中field type每种类型个数为50，每条json数据的payload约为 18KB；数据提前发送到kafka中累积起来，便于在测试中反复使用，同时避免发送数据客户端速度过低的问题，保证全速消费。

场景：

- ✓ 仅添加KafkaSpout;
- ✓ 添加KafkaSpout和TransformBolt，但是TransformBolt中不包含实际处理逻辑，直接ack;
- ✓ 添加KafkaSpout和TransformBolt，TransformBolt中包含实际处理逻辑，但是向后emit数据时不挂靠input，并直接ack input;
- ✓ 添加KafkaSpout、TransformBolt和KafkaBolt，包含完整的处理路径;

结论：

- ✓ 并行度调整为kspout×4，tbolt×16，kbolt×4，到达当前资源和网络的上限，总吞吐率为 500 × 4 = 2000 tuples /s

踩过的坑:

- zk集群异常后kafka broker无法自动恢复： KAFKA-2729
- partitions分布不均匀
- Kafka升级: 2.10-0.8.2.2 -> 2.11-0.10.0.1
- batch/compression

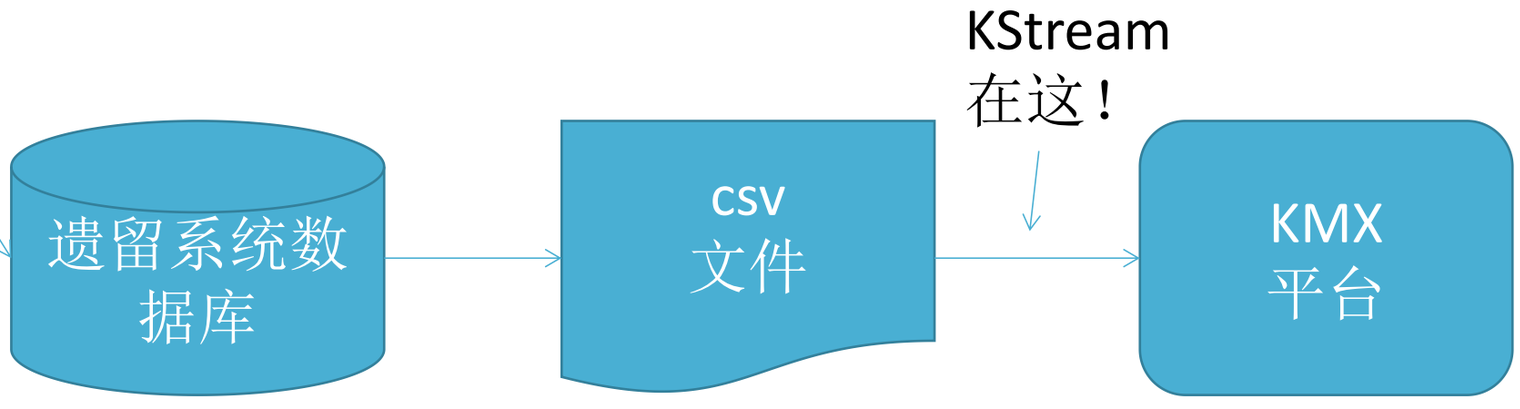
将走的路:

- docker kafka -> cloudera kafka
- 认证/授权
- 流控: Quota
- 实时处理/查询： Storm？ KStream？ Spark Streaming？ Beam？ ...

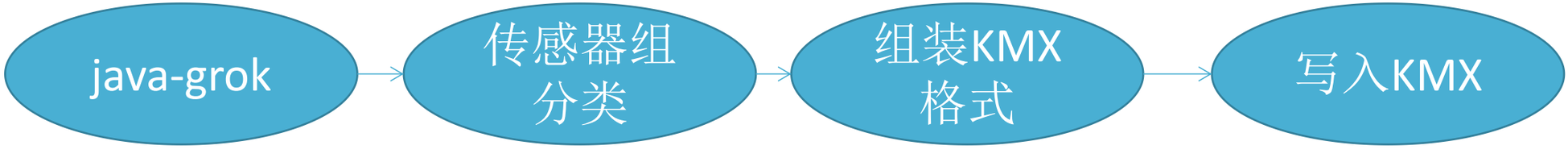
数据管理案例1：使用KStream做通用ETL工具

K²DATA

阶段1：按需开发KStream APP



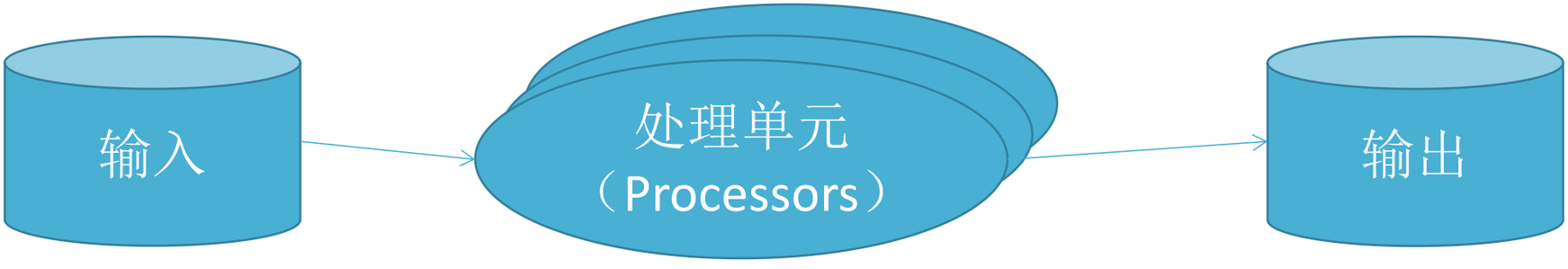
...
2016-10-08
16:57:43.291,132,3,1501:0.0;1502:0.0;1503:0.0;1504:0.0;1505:0.0;1506:0.0;1507:0.0;1508:0.0;1509:0.0;1510:0.0;1511:0.0;1512:0.0;1513:0.0;1514:0.0;1515:0.0;1516:0.0;1517:0.0;1518:0.0;1519:0.0;1520:0.0;1521:0.0;1522:0.0;1523:0.0;1524:0.0;1525:0.0;1526:0.0;1527:0.0;1528:0.0;1529:0.0;1530:0.0;1531:0.0;1532:0.0;1533:0.0;1534:0.0;1535:0.0;1536:0.0;1537:0.0;1538:0.0;1539:1.0;1540:0.0;1541:0.0;1542:0.0;1543:0.0;1544:0.0;1545:0.0;1546:0.0;1547:0.0;1548:0.0;1549:0.0;1550:0.0;1551:0.0;1552:0.0;1553:0.0;1554:0.0;1555:0.0;1556:0.0;1557:0.0;1558:0.0;1559:0.0;1560:0.0;1561:0.0;1562:0.0;1563:0.0;1564:0.0;1565:0.0;1566:0.0;1567:0.0;1568:0.0;1569:0.0;1570:0.0;1571:0.0;1572:0.0;1573:0.0;1574:0.0;1575:0.0;1576:0.0;1577:0.0;1578:0.0;1579:0.0;1580:0.0;.472;1570:412.902;1571:0.0;1572:3.0;1573:177.0;1574:0.0;1575:0.0;1576:0.0;1577:0.0;1578:0.0;1579:0.0;1580:0.0;.0;1580:0.0;;;;;;;;;0;;;;;
2016-10-08
16:57:43.291,134,3,1501:0.0;1502:0.0;1503:0.0;1504:0.0;1505:0.0;1506:0.0;1507:0.0;1508:0.0;1509:0.0;1510:0.0;1511:0.0;1512:0.0;1513:0.0;1514:0.0;1515:0.0;1516:0.0;1517:0.0;1518:0.0;1519:0.0;1520:0.0;1521:0.0;1522:0.0;1523:0.0;1524:0.0;1525:0.0;1526:0.0;1527:0.0;1528:0.0;1529:0.0;1530:0.0;1531:0.0;1532:0.0;1533:0.0;1534:0.0;1535:0.0;1536:0.0;1537:0.0;1538:0.0;1539:1.0;1540:0.0;1541:0.0;1542:0.0;1543:0.0;1544:0.0;1545:0.0;1546:0.0;1547:0.0;1548:0.0;1549:0.0;1550:0.0;1551:0.0;1552:0.0;1553:0.0;1554:0.0;1555:0.0;1556:0.0;1557:0.0;1558:0.0;1559:0.0;1560:0.0;1561:0.0;1562:0.0;1563:0.0;1564:0.0;1565:0.0;1566:0.0;1567:0.0;1568:0.0;1569:0.0;1570:0.0;1571:0.0;1572:0.0;1573:0.0;1574:0.0;1575:0.0;1576:0.0;1577:0.0;1578:0.0;1579:0.0;1580:0.0;.472;1570:412.902;1571:0.0;1572:3.0;1573:177.0;1574:0.0;1575:0.0;1576:0.0;1577:0.0;1578:0.0;1579:0.0;1580:0.0;.0;1580:0.0;;;;;;;;;0;;;;;
2016-10-08 16:57:43.571,5,1,1024:0.0;
2016-10-08 16:57:43.571,7,1,1024:0.0;
2016-10-08 16:57:43.571,9,1,1024:0.0;
...



数据管理案例1：使用KStream做通用ETL工具

K²DATA

阶段2：通用工具（深受Logstash启发）：根据配置定制KStream数据流



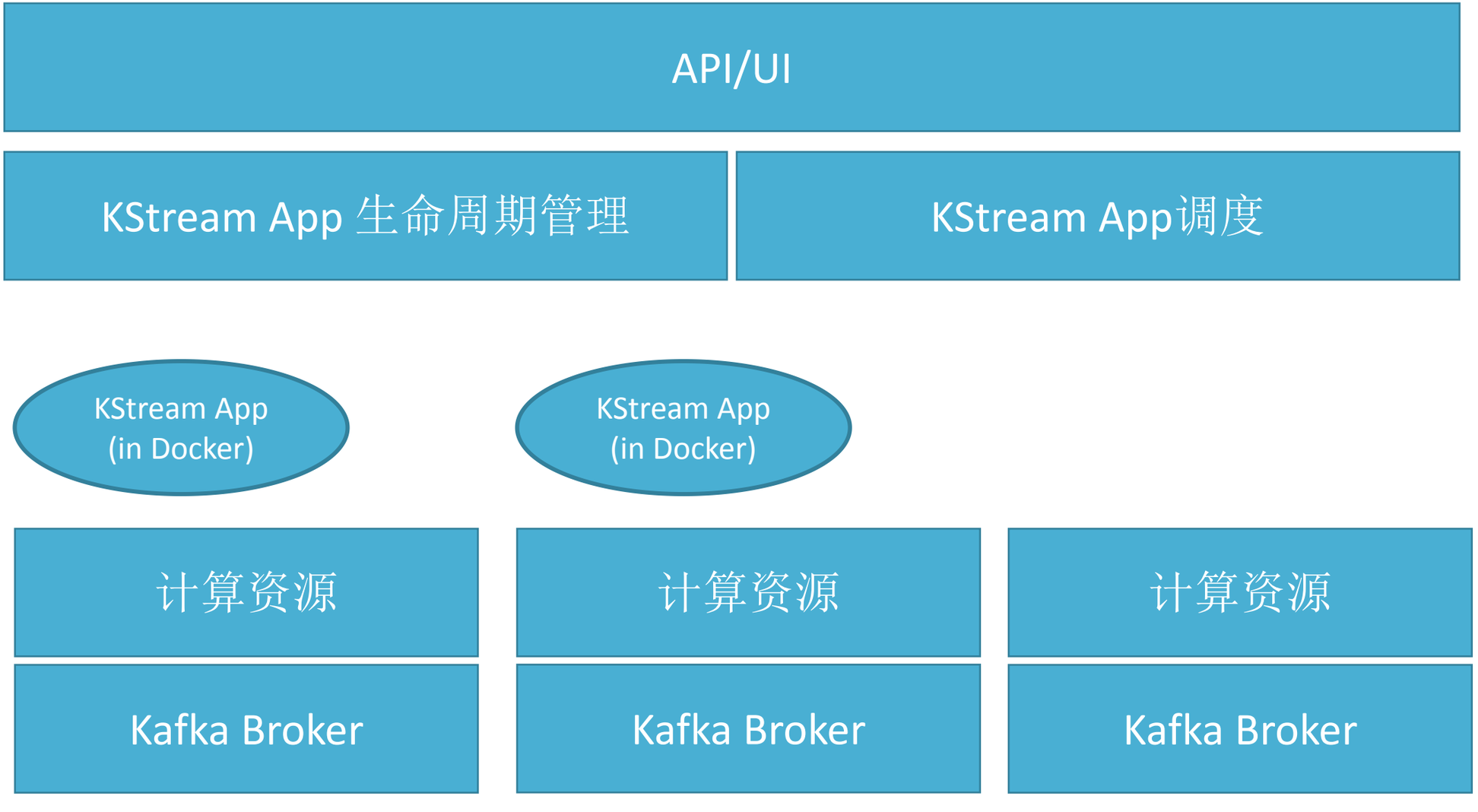

```
{
  "input": {
    "kafka": {
      "bootstrap_servers": "internal",
      "topics": [
        "csv-raw"
      ],
      "auto_offset_reset": "earliest",
      "client_id": "client1",
      "group_id": "group1"
    }
  }
}
```

```
{
  "filter": [
    {
      "grok": {
        "patterns_dir": [
          "foo/bar"
        ],
        "match": {
          "message": "%{YEAR:year}-
%{MONTHNUM:monthnum}-%{MONTHDAY:monthday}
%{TIME:time},%{BASE10NUM:assetId},1,(%{BASE10NUM:
field}:%{SENSORVALUE:value};)(%{BASE10NUM:field}:%{S
ENSORVALUE:value};)?"
        },
        "keep_empty_captures": true
      }
    },
    {
      "translate": {
        "field": "group",
        "dictionary_path": "/path/to/mapping.json"
      }
    }
  ]
}
```

```
{
  "output": [
    {
      "kafka": {
        "bootstrap_servers": "internal",
        "topic_id": "gw-twincat-kmx-debug",
        "codec": "json"
      }
    },
    {
      "kmx-stream-in-json": {
        "bootstrap_servers": "192.168.130.115:9092",
        "field_group": "%{group}",
        "timestamp": "%{year}-%{monthnum}-
%{monthday}T%{time}+08:00",
        "fields": "%{field}",
        "values": "%{value}"
      }
    }
  ]
}
```

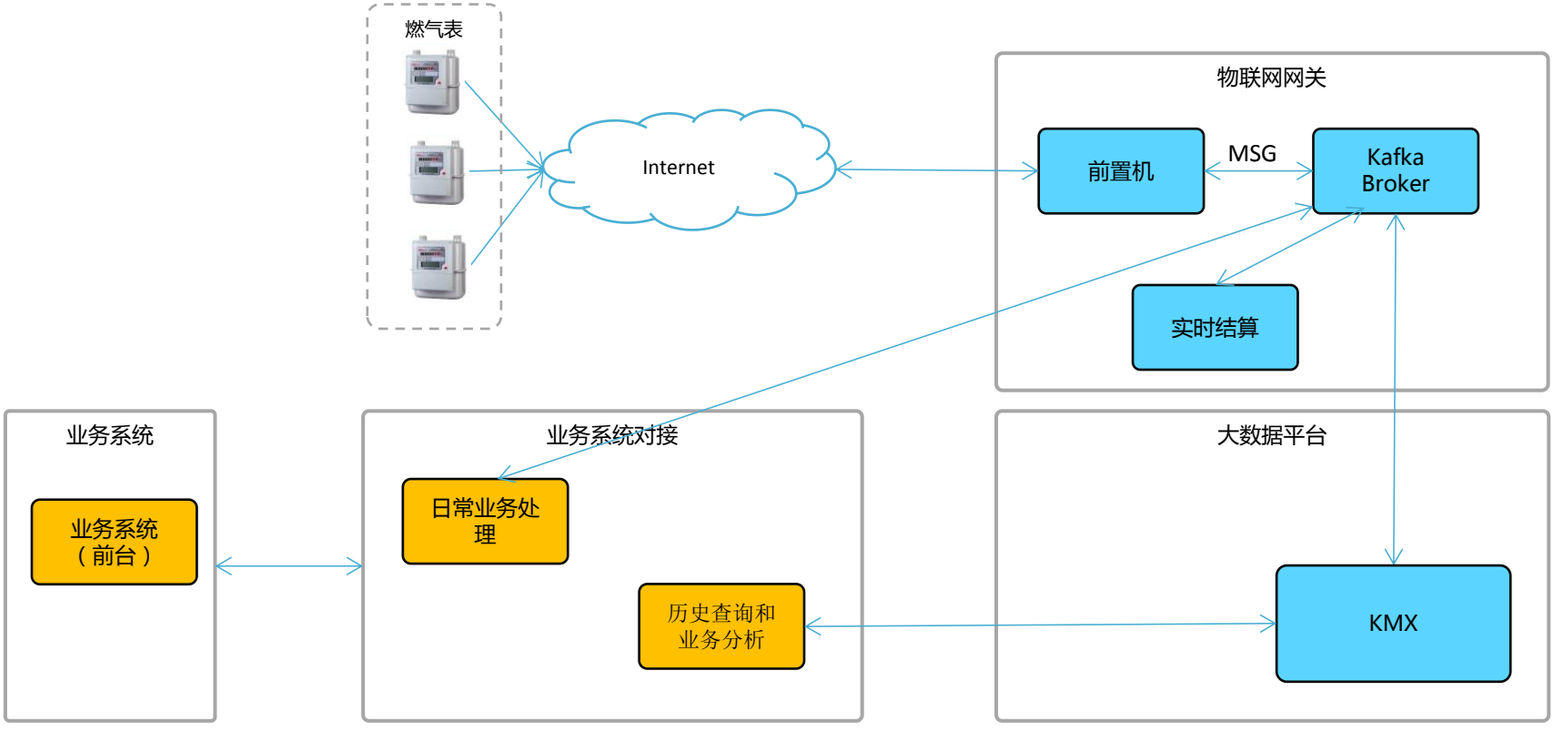
数据管理案例1：使用KStream做通用ETL工具

架构

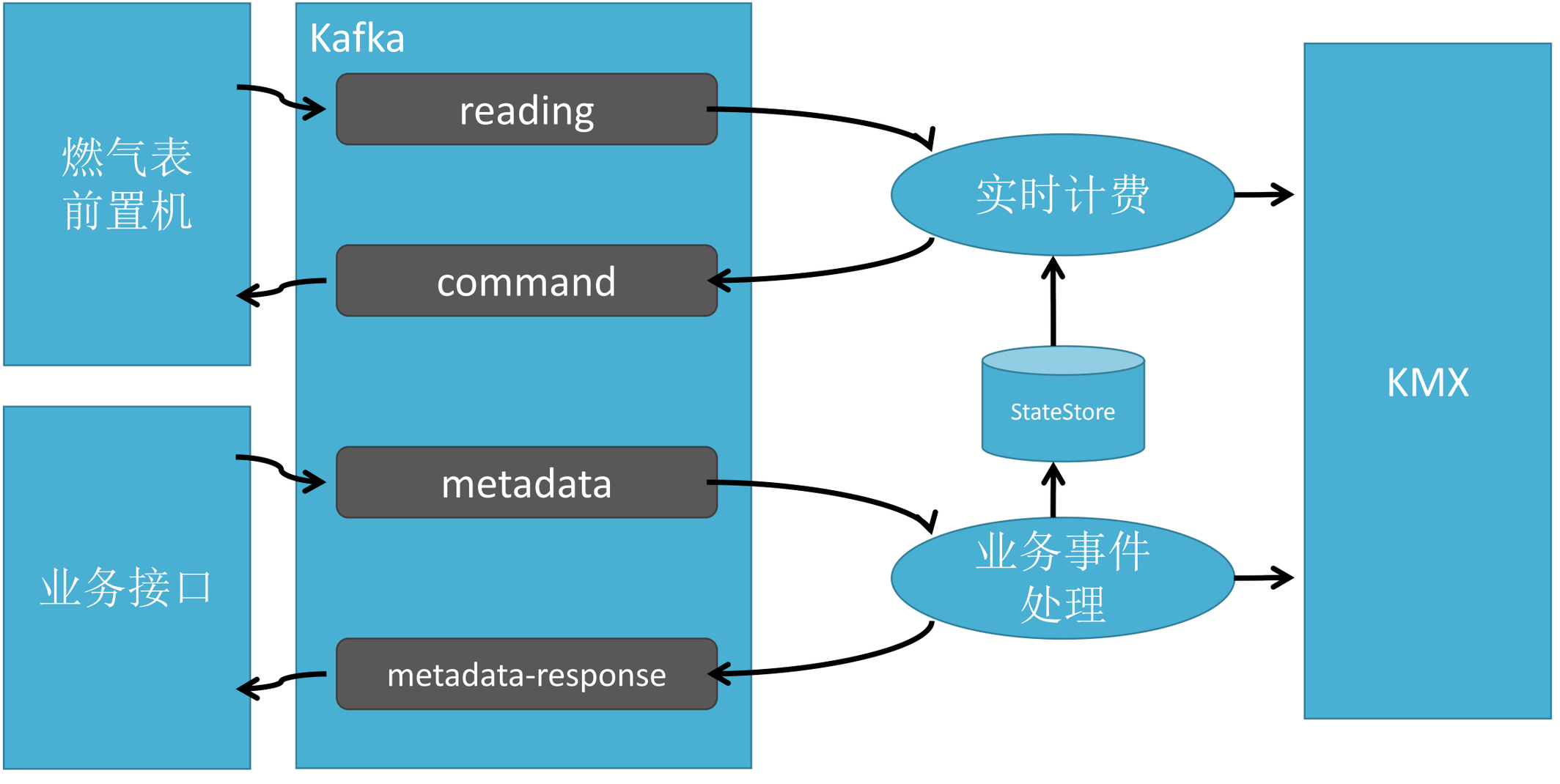


数据管理案例2：使用KStream做燃气表的实时计费

数据管理案例2：使用KStream做燃气表的实时计费



数据管理案例2：使用KStream做燃气表的实时计费



数据管理案例2：使用KStream做燃气表的实时计费

未来趋势： 云+端