
AWS Summit
AWS技术峰会2015 · 上海
亚马逊
aws

AWS大数据架构模式和最佳实践

曹玮祺 博士
AWS中国 解决方案架构师团队 高级主管
weiqicao@amazon.com

前所未有的大数据

大数据的演进

- 批处理
- 报表

- 实时
- 警报

- 预测
- 预报

应接不暇的大数据工具

Cassandra

EMR

S3

DynamoDB

SQS

Amazon
Redshift

Amazon
Glacier

RDS

ElastiCache

Amazon Kinesis

Kinesis-
enabled
app

Data Pipeline

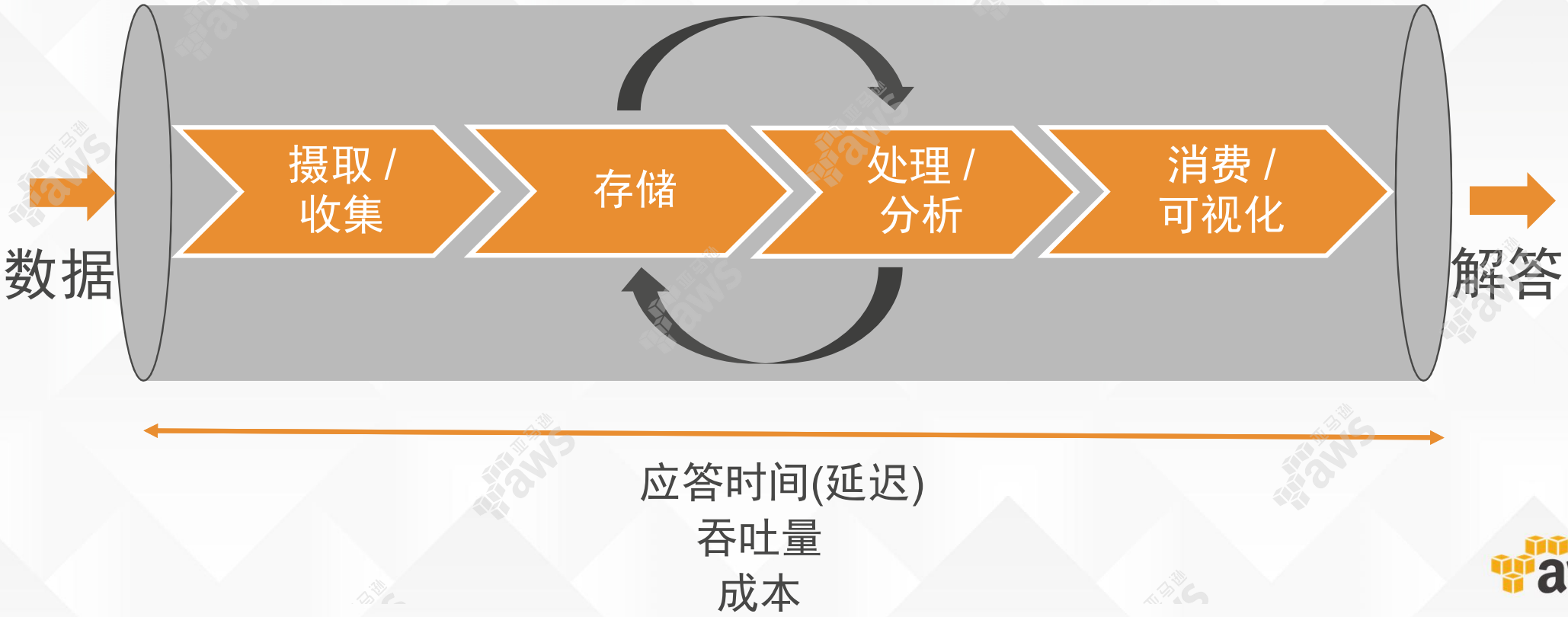
CloudSearch

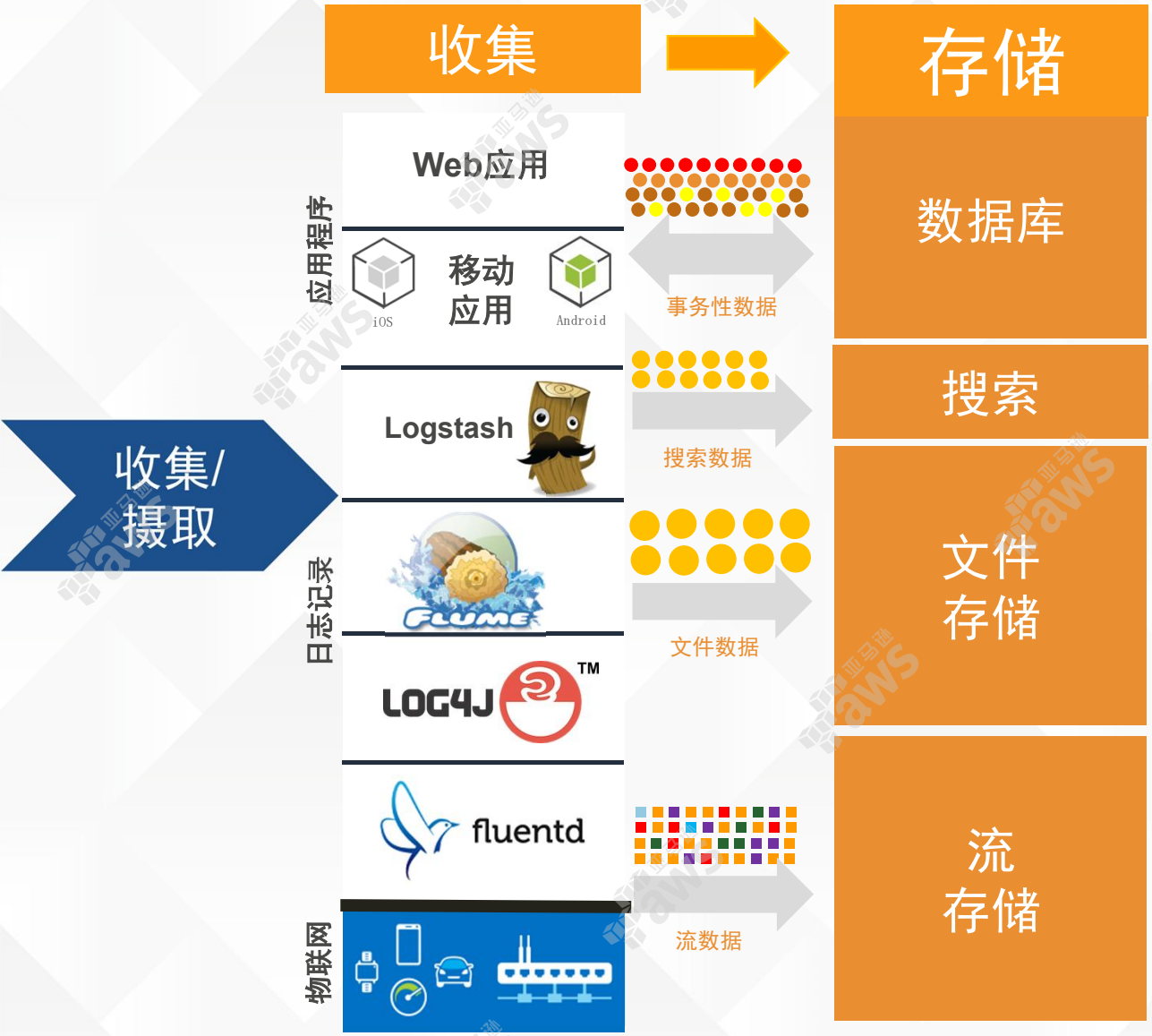
Lambda

ML

DynamoDB
Streams

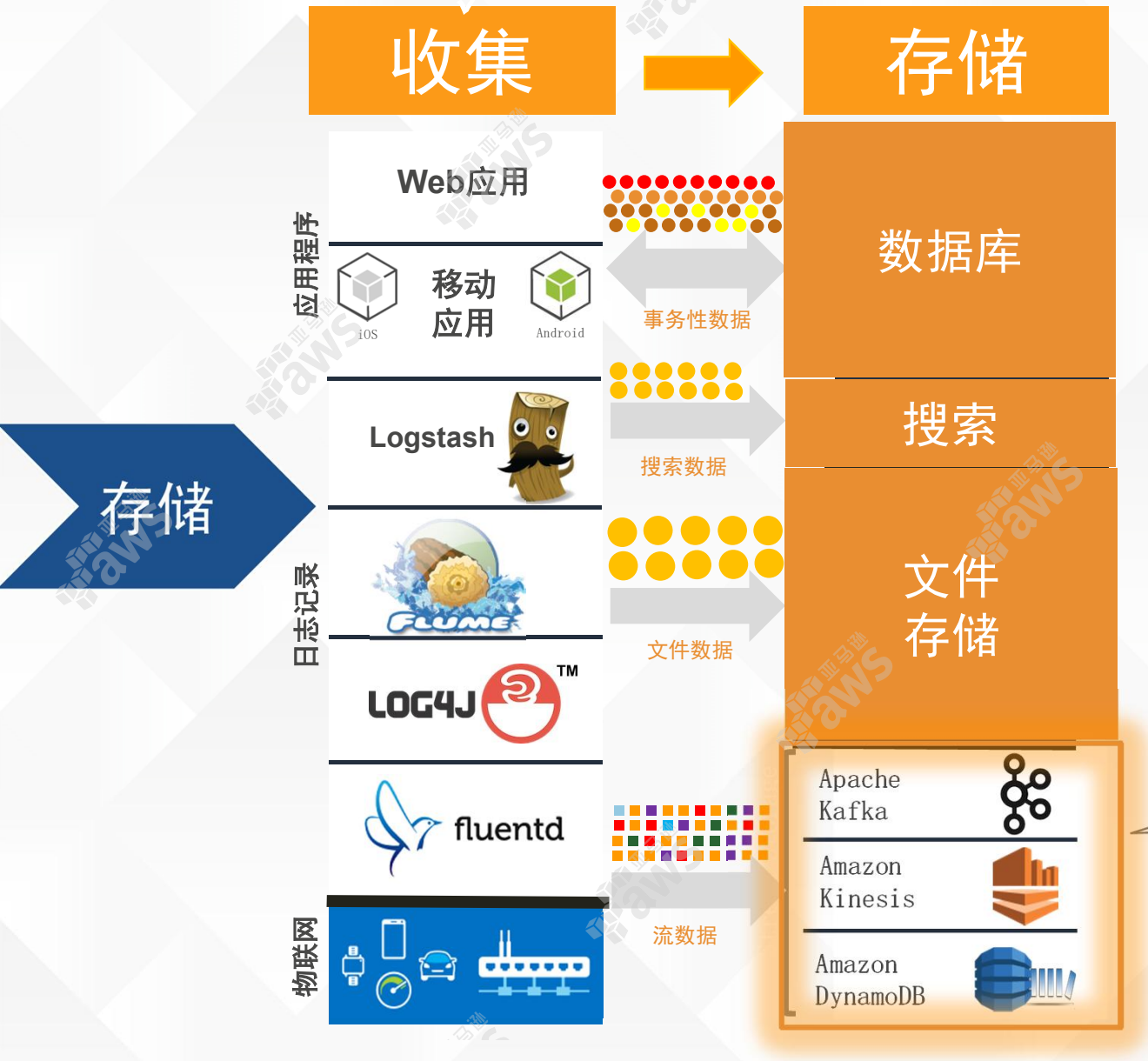
简化大数据的处理





数据类型

- 事务处理
  - 数据库读写 (OLTP)
  - 缓存
- 搜索
  - 日志
  - 流
- 文件
  - 日志文件(/var/log)
  - 日志收集和框架
- 流
  - 日志记录
  - 传感器和IoT数据

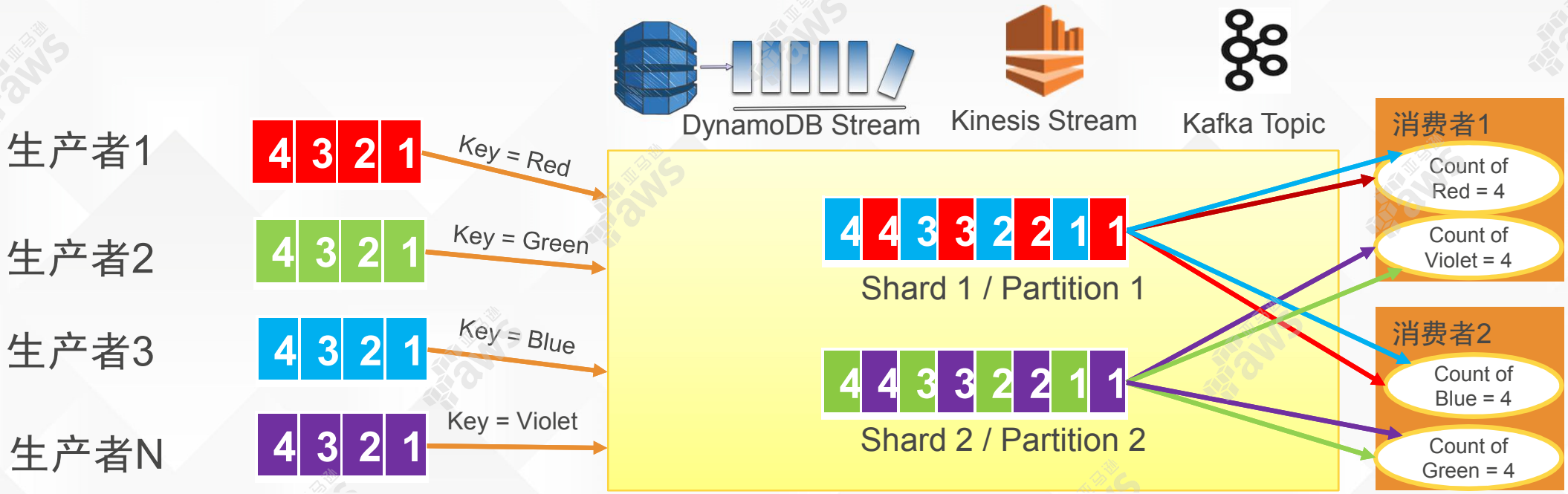


流存储选项

- AWS 托管服务
  - Amazon Kinesis → 流
  - DynamoDB Streams → 表+流
  - Amazon SQS → 队列
  - Amazon SNS → 发布/订阅
- 非托管的
  - Apache Kafka → 流

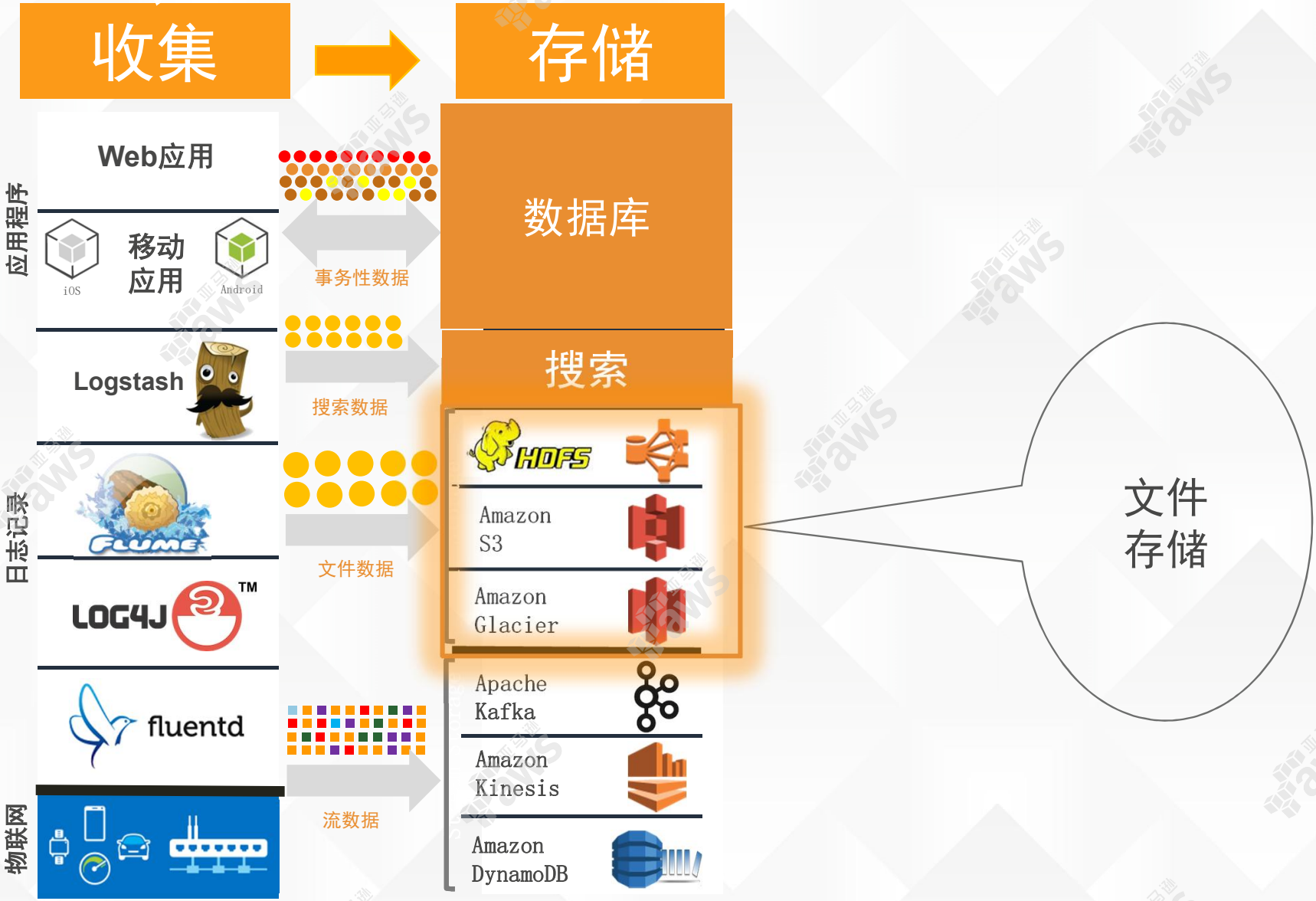
为什么使用流存储?

- 解耦生产者和消费者
- 持久的缓存
- 收集多个流
- 保留客户端顺序
- 流式MapReduce
- 并行消费



我应该使用哪个流存储?

	Amazon Kinesis	DynamoDB Streams	Amazon SQS Amazon SNS	Kafka
托管	Yes	Yes	Yes	No
有序	Yes	Yes	No	Yes
分发	最少执行一次	只执行一次	最少执行一次	最少执行一次
有效期	7 天	24 小时	14 天	可配置
复制	3 可用区	3 可用区	3 可用区	可配置
吞吐量	没有限制	没有限制	没有限制	~ 节点
并行客户端	Yes	Yes	No (SQS)	Yes
MapReduce	Yes	Yes	No	Yes
记录大小	1MB	400KB	256KB	可配置
成本	低	较高(table cost)	低 - 中	低 (+admin)

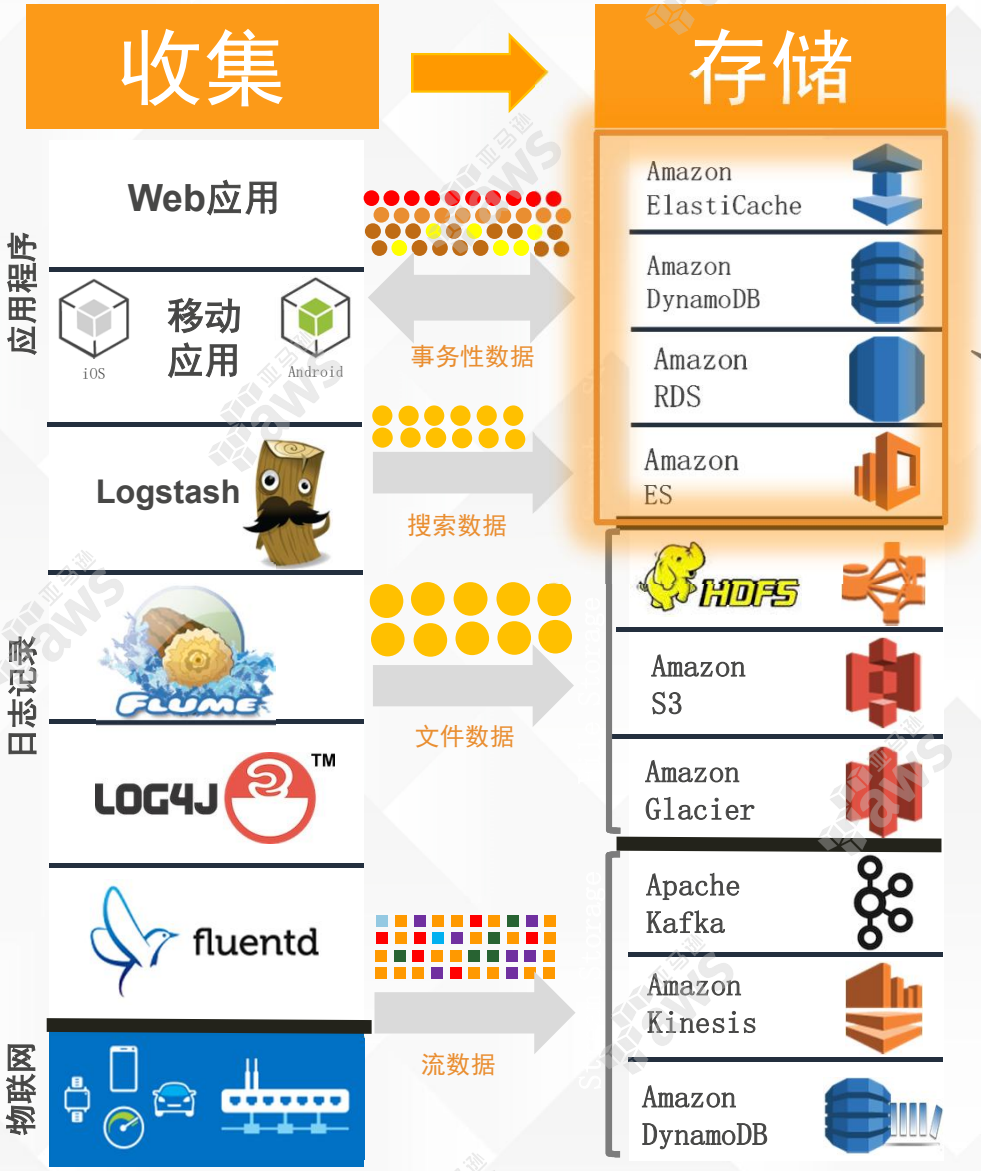

收集
存储
应用程序
Web应用
移动应用
iOS
Android
日志记录
Logstash
FLUME
LOG4J
fluentd
物联网
事务性数据
搜索数据
文件数据
流数据
数据库
搜索
HDFS
Amazon S3
Amazon Glacier
Apache Kafka
Amazon Kinesis
Amazon DynamoDB
文件存储

为什么Amazon S3善于处理大数据?

- 支持大数据框架(Spark, Hive, Presto, etc.)
- 不需要为存储运行计算集群(不像HDFS)
- 能瞬间运行Hadoop集群 & Amazon EC2 Spot 实例
- 多个不同的(Spark, Hive, Presto)集群能使用相同的数据源
- 不限数量的对象存储
- 非常高的带宽 – 没有总吞吐量的限制
- 高可用性 – 可以容忍AZ失败
- 为99.999999999% 持久性设计
- 分级存储 (Standard, IA, Amazon Glacier) 通过 Life-cycle 的策略
- 安全 – SSL, 客户端/服务端数据加密
- 低成本

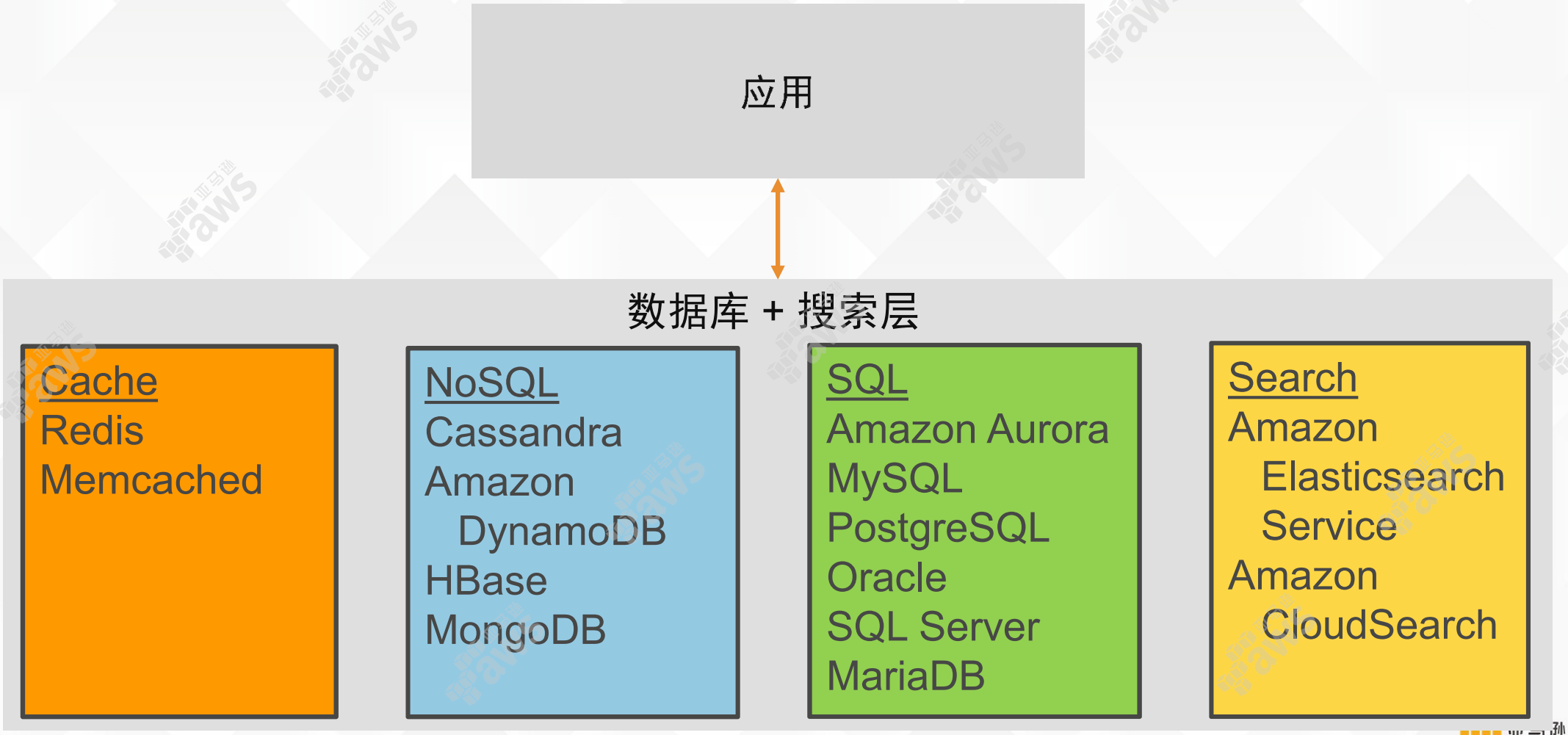
HDFS & Amazon Glacier

- 使用HDFS用于高频繁访问 (热) 数据
- 使用Amazon S3标准用于频繁访问数据
- 使用Amazon S3标准 – IA 用于非频繁访问数据
- 使用Amazon Glacier 归档冷数据

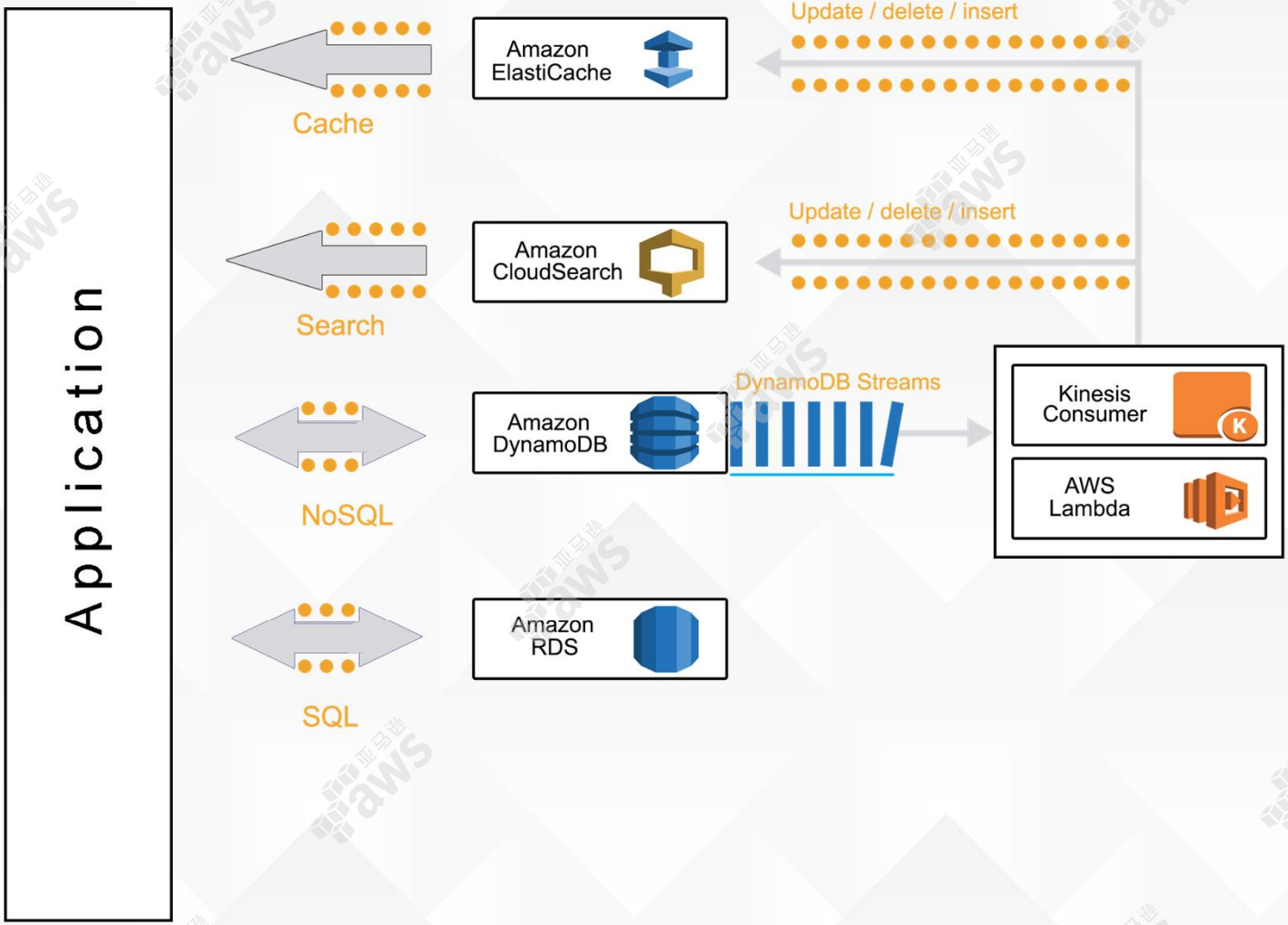


数据库 + 搜索层

最佳实践 — 用正确的工具做正确的事情



实例化视图


Application
Cache
Amazon ElastiCache
Update / delete / insert
Search
Amazon CloudSearch
Update / delete / insert
NoSQL
Amazon DynamoDB
DynamoDB Streams
Kinesis Consumer
AWS Lambda
SQL
Amazon RDS

我应该使用什么数据存储?

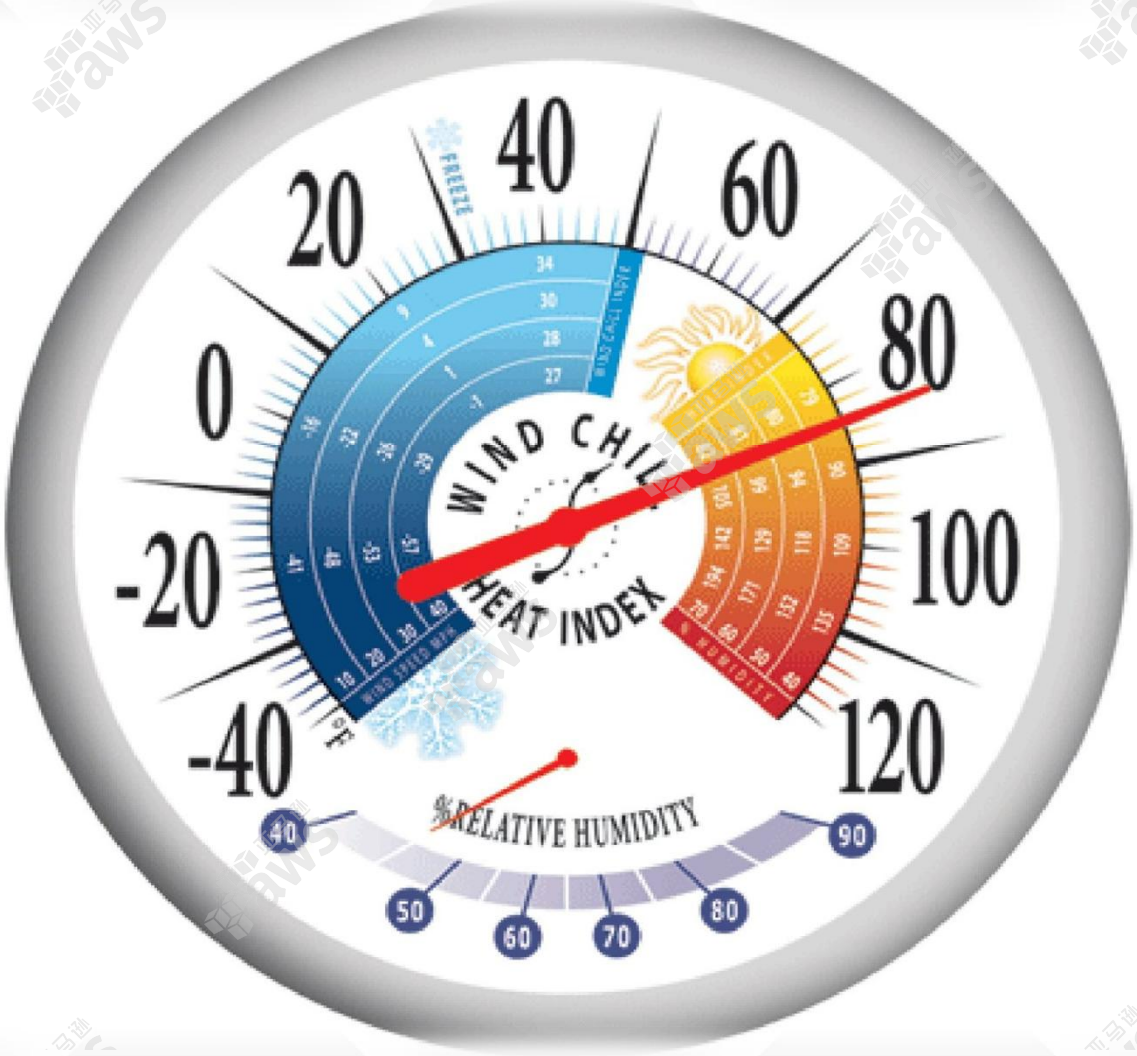
- 数据结构 → 固定的模式、 JSON 格式、键-值对格式
- 访问模式 → 按照你将访问的格式存储数据
- 数据 / 访问特征→ 热数据、温数据、冷数据
- 成本 → 适合的成本

数据结构和访问模式

访问模式	使用什么?
Put/Get (Key, Value)	Cache, NoSQL
Simple relationships → 1:N, M:N	NoSQL
Cross table joins, transaction, SQL	SQL
Faceting, Search	Search

数据结构	使用什么?
Fixed schema	SQL, NoSQL
Schema-free (JSON)	NoSQL, Search
(Key, Value)	Cache, NoSQL

你的数据 / 访问的温度是怎样的？



数据 / 访问特征：热，温，冷

热数据 温数据 冷数据

	Hot	Warm	Cold
容量	MB–GB	GB–TB	PB
数据项大小	B–KB	KB–MB	KB–TB
等待时间	ms	ms, sec	min, hrs
持久性	低–高	高	非常高
请求率	非常高	高	低
成本/GB	$$-$	$-¢¢	¢

我应该使用什么数据存储?

热数据 温数据 冷数据

	Amazon ElastiCache	Amazon DynamoDB	Amazon Aurora	Amazon Elasticsearch	Amazon EMR (HDFS)	Amazon S3	Amazon Glacier
平均等待时间	ms	ms	ms, sec	ms,sec	sec,min,hrs	ms,sec,min (~ size)	hrs
数据容量	GB	GB–TBs (no limit)	GB–TB (64 TB Max)	GB–TB	GB–PB (~nodes)	MB–PB (no limit)	GB–PB (no limit)
数据项大小	B-KB	KB (400 KB max)	KB (64 KB)	KB (1 MB max)	MB-GB	KB-GB (5 TB max)	GB (40 TB max)
请求率	高 – 非常高	非常高(没有限制)	高	高	低 – 非常高	低 – 非常高(没有限制)	非常低
存储成本 GB/month	$$	¢¢	¢¢	¢¢	¢	¢	¢/10
持久性	低 – 中等	非常高	非常高	高	高	非常高	非常高

热数据 温数据 冷数据

成本意识的设计

举例: 我应该使用Amazon S3 还是 Amazon DynamoDB?

- “我们目前正在做一个项目，将大大增加我的团队使用Amazon S3的用量。希望你能回答一些问题。当前迭代的设计将访问很多小文件,峰值也许到十亿。总大小接近**1.5 TB**每月…”

请求率(Writes/sec)	对象大小(Bytes)	总大小 (GB/month)	每月的对象
300	2048	1483	777,600,000

Amazon S3 还是Amazon DynamoDB?

请求速率 (写/秒)	对象大小 (Bytes)	总大小 (GB/月)	每月的对象
300	2,048	1,483	777,600,000

Amazon DynamoDB is a high performance non-relational database service that is easy to set up, operate, and scale. It is designed to address the core problems of database management, performance, scalability, and reliability. It also provides predictable high performance and low latency at scale.

Indexed Data Storage:

Dataset Size: 1483 GB

Provisioned Throughput Capacity *:

Item Size (All attributes): 2 KB

Number of items read per second: 0 Reads/Second

Read Consistency: Strongly Consistent / Eventually Cons cheaper)

Number of items written per second: 300 Writes/Second

Amazon DynamoDB Service (US-East)		$ 644.30
Provisioned Throughput Capacity:	$ 261.69	
Indexed Data Storage:	$ 382.61	

Amazon S3 is storage for the Internet. It is designed to make web-scale computing easier for developers.

Storage:

Storage: 1483 GB

Reduced Redundancy Storage: 0 GB

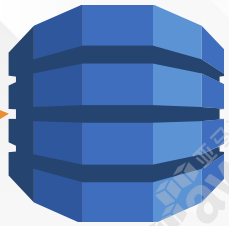
Requests:

PUT/COPY/POST/LIST Requests: 77760000 Requests

GET and Other Requests: 0 Requests

Amazon S3 Service (US-East)		$ 3932.27
Storage:	$ 44.27	
Put/List Requests:	$ 3888.00	

amazon web services SIMPLE MONTHLY CALCULATOR



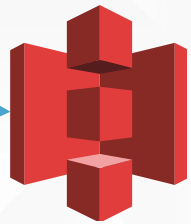
Amazon DynamoDB

选用

Amazon S3 Service (US-East)		$ 3932.27
Storage:	$ 44.27	
Put/List Requests:	$ 3888.00	
Amazon DynamoDB Service (US-East)		$ 644.30
Provisioned Throughput Capacity:	$ 261.69	
Indexed Data Storage:	$ 382.61	
DynamoDB Streams:	$ 0.00	

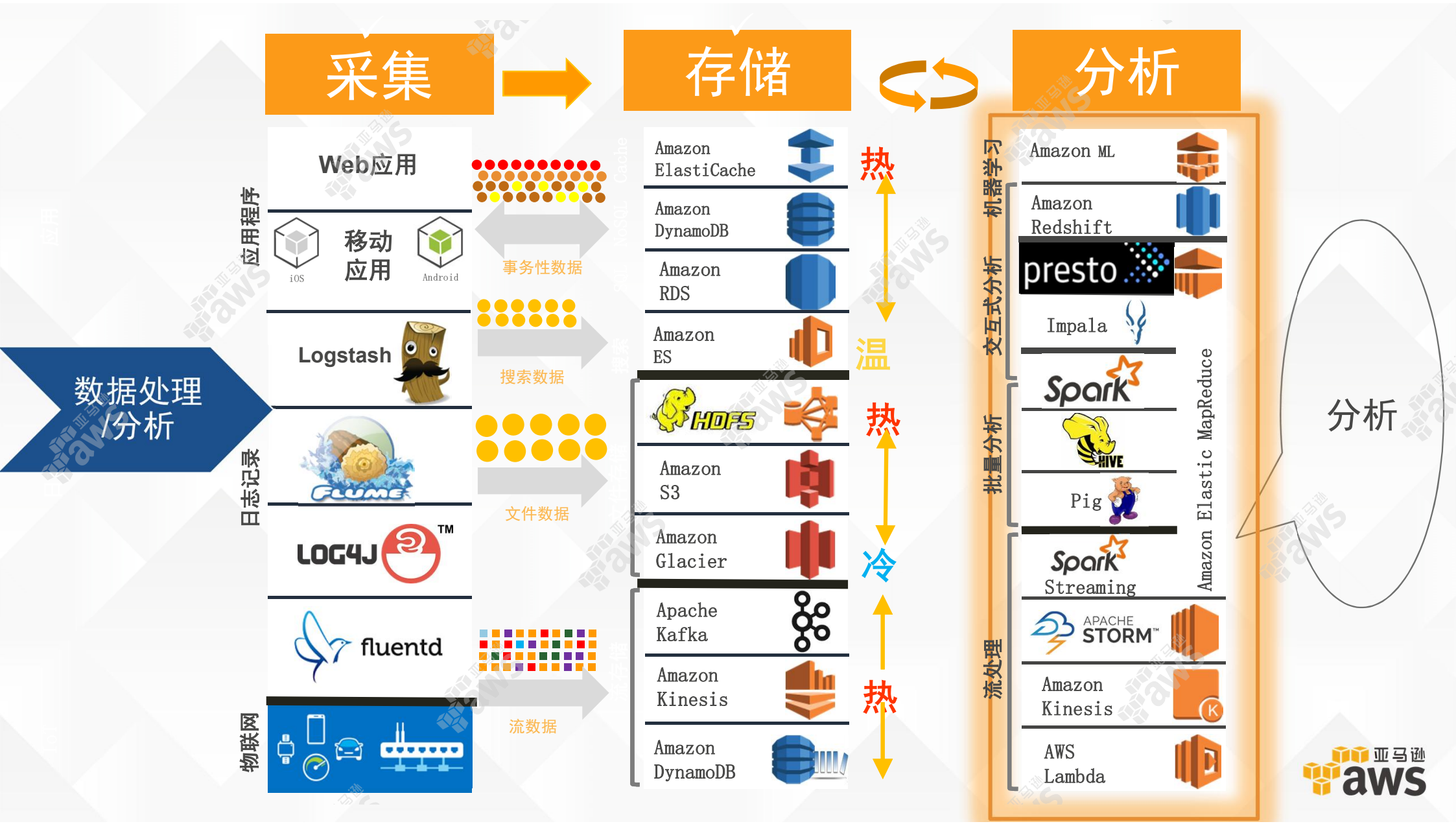
	请求速率 (写/秒)	对象大小 (Bytes)	总大小 (GB/month)	每月的对象
方案 1	300	2,048	1,483	777,600,000
方案 2	300	32,768	23,730	777,600,000

选用



Amazon S3

Amazon S3 Service (US-East)		$ 4588.55
Storage:	$ 700.55	
Put/List Requests:	$ 3888.00	
Amazon DynamoDB Service (US-East)		$ 10131.40
Provisioned Throughput Capacity:	$ 4187.04	
Indexed Data Storage:	$ 5944.36	
DynamoDB Streams:	$ 0.00	

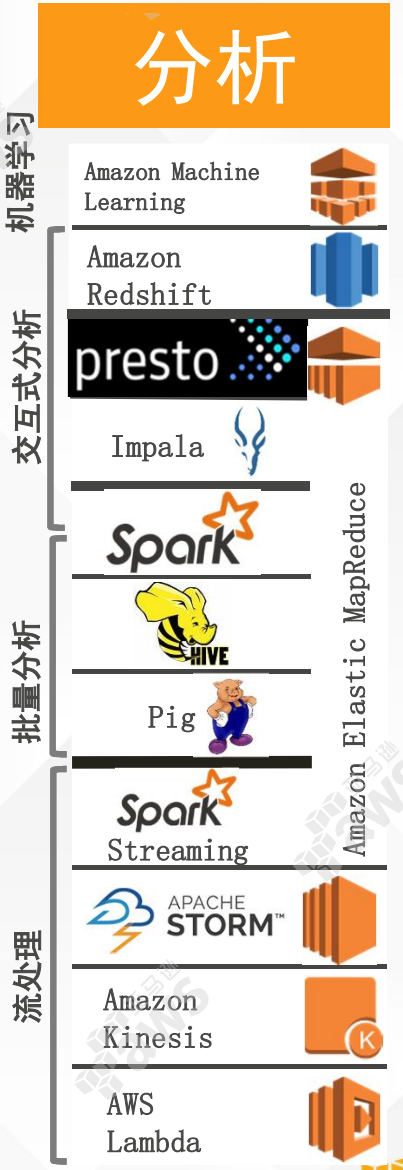

采集
存储
分析
数据处理/分析
应用程序
Web应用
移动应用
iOS
Android
日志记录
Logstash
FLUME
LOG4J
fluentd
物联网
事务性数据
搜索数据
文件数据
流数据
Amazon ElastiCache
Amazon DynamoDB
Amazon RDS
Amazon ES
HDFS
Amazon S3
Amazon Glacier
Apache Kafka
Amazon Kinesis
Amazon DynamoDB
热
温
热
冷
热
机器学习
交互式分析
批量分析
流处理
Amazon ML
Amazon Redshift
presto
Impala
Spark
HIVE
Pig
Spark Streaming
APACHE STORM
Amazon Kinesis
AWS Lambda
Amazon Elastic MapReduce
分析

数据处理 / 分析

- 交互式分析
  - 需要大量的数据（温数据/冷数据）
  - 秒级得到结果反馈
  - 样例: 自服务仪表板
- 实时分析
  - 需要小量热数据和提出问题
  - 短时间内得到结果反馈（毫秒级或者是秒级）
  - 实时 (事件)
    - 数据流实时响应事件
    - 样例: 账单/欺诈警报
  - 近实时 (微-批量)
    - 近乎实时的小批量数据流事件处理
    - 样例: 1分钟指标
- 批量分析
  - 需要大量的数据（温数据/冷数据）
  - 分钟级或者是小时级得到结果反馈
  - 样例: 生成每天，每周。或者是每月的报告
- 通过机器学习预测
  - 机器学习给计算机学习的能力，并且不需要显示的编程
  - 机器学习算法：
  - 监督式学习 ← “teach” 程序
    - 分类 ← 这是交易欺诈吗？ (Yes/No)
    - 回归分析 ← 客户终身价值?
  - 非监督式学习 ← 让它自己学习
    - 聚类 ← 市场划分

分析工具和框架

- 机器学习
  - Mahout, Spark ML, Amazon ML
- 交互式分析
  - Amazon Redshift, Presto, Impala, Spark
- 批量分析
  - MapReduce, Hive, Pig, Spark
- 流处理
  - 微-批量: Spark Streaming, KCL, Hive, Pig
  - 实时: Storm, AWS Lambda, KCL



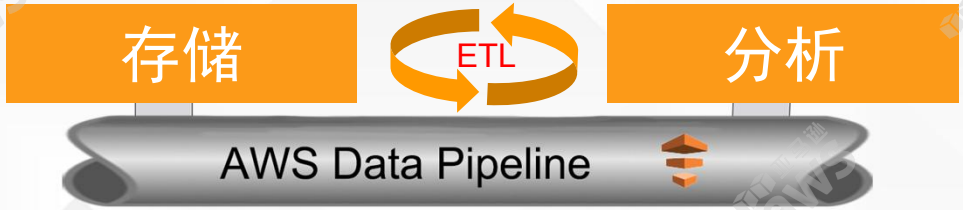
我应该使用什么流处理技术?

	Spark Streaming	Apache Storm	Amazon Kinesis Client Library	AWS Lambda	Amazon EMR (Hive, Pig)
规模 / 吞吐量	~ Nodes	~ Nodes	~ Nodes	Automatic	~ Nodes
批处理 or 实时	实时	实时	实时	实时	批处理
可管理性	Yes (Amazon EMR)	Do it yourself	Amazon EC2 + Auto Scaling	AWS managed	Yes (Amazon EMR)
容错性	Single AZ	可配置	Multi-AZ	Multi-AZ	Single AZ
编程语言	Java, Python, Scala	Any language via Thrift	Java, via MultiLangDaemon (.Net, Python, Ruby, Node.js)	Node.js, Java	Hive, Pig, Streaming languages

我应该使用什么数据处理技术?

	Amazon Redshift	Impala	Presto	Spark	Hive
查询延迟性	Low	Low	Low	Low	Medium (Tez) – High (MapReduce)
持久性	High	High	High	High	High
数据卷	1.6 PB Max	~Nodes	~Nodes	~Nodes	~Nodes
托管型	Yes	Yes (EMR)	Yes (EMR)	Yes (EMR)	Yes (EMR)
存储	Native	HDFS / S3A*	HDFS / S3	HDFS / S3	HDFS / S3
SQL 兼容性	High	Medium	High	Low (SparkSQL)	Medium (HQL)

如何ETL?



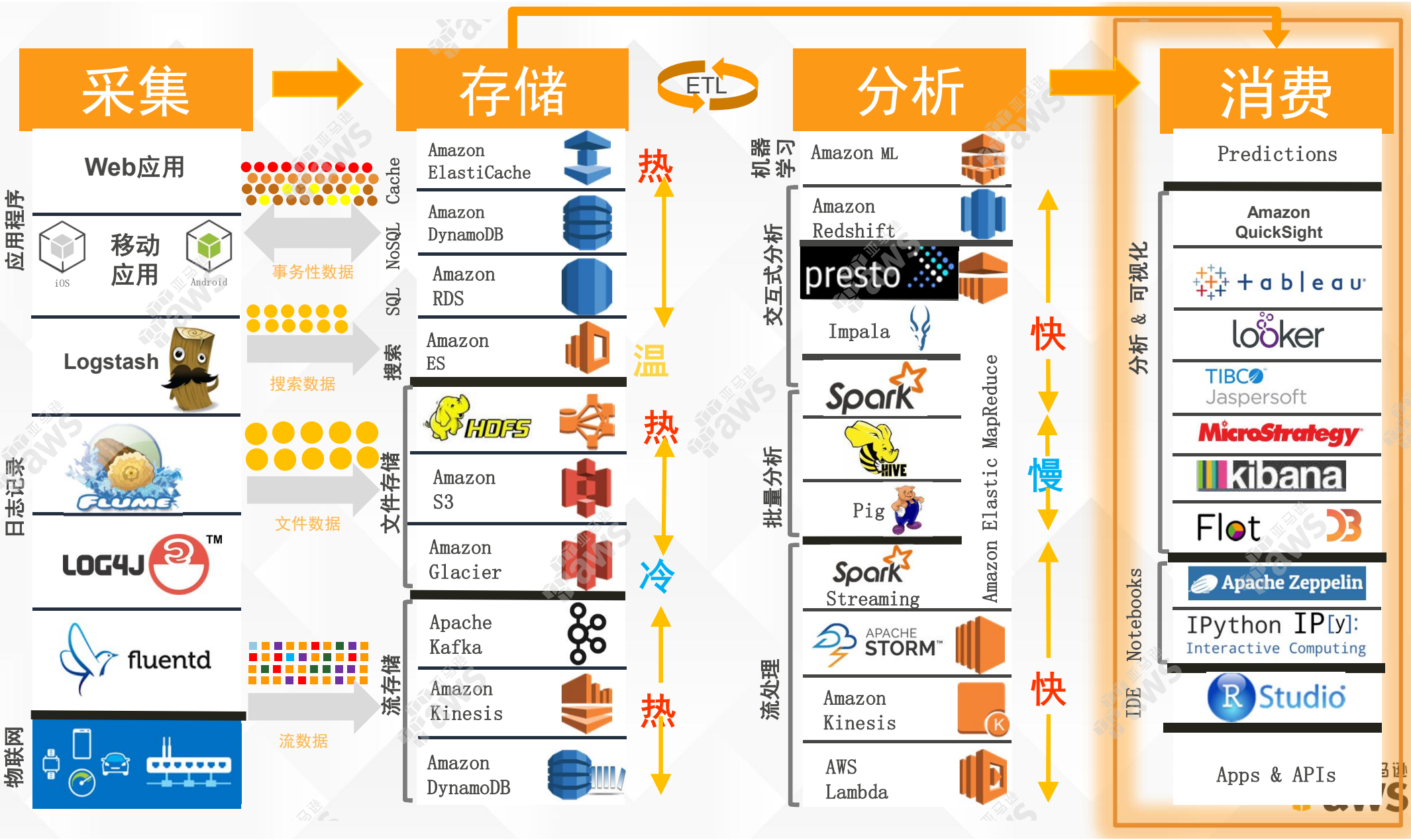
Data Integration

Reduce the effort to move, cleanse, synchronize, manage, and automatize data related processes.

Attunity CloudBeam	Informatica Cloud	Matillion ETL for Redshift	snapLogic
alteryx			

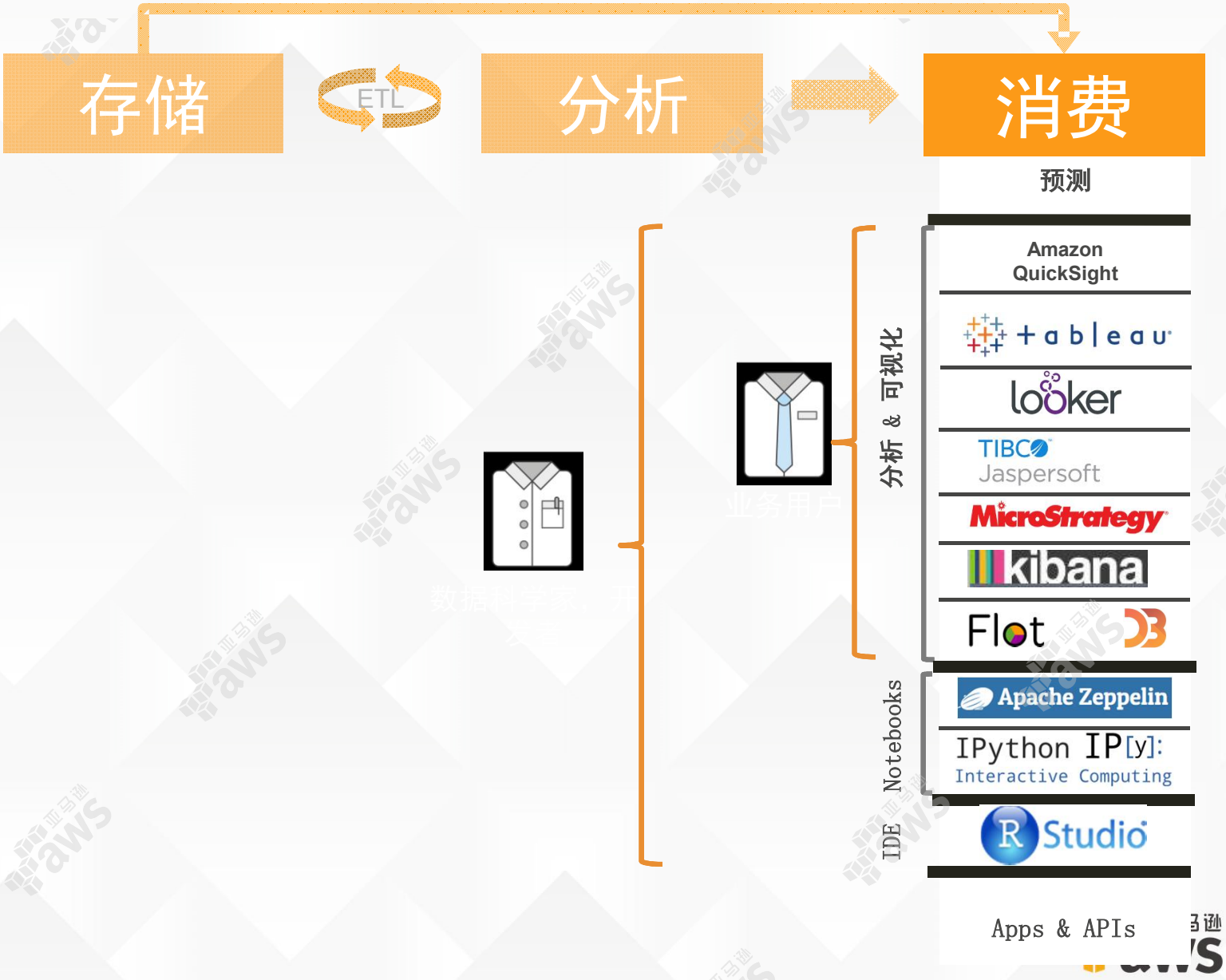
https://aws.amazon.com/big-data/partner-solutions/

消费 / 可视化


采集
存储
ETL
分析
消费
应用程序
Web应用
移动应用
iOS
Android
日志记录
Logstash
FLUME
LOG4J
fluentd
物联网
事务性数据
搜索数据
文件数据
流数据
Cache
NoSQL
SQL
搜索
文件存储
流存储
Amazon ElastiCache
Amazon DynamoDB
Amazon RDS
Amazon ES
HDFS
Amazon S3
Amazon Glacier
Apache Kafka
Amazon Kinesis
Amazon DynamoDB
热
温
热
冷
热
机器学习
交互式分析
批量分析
流处理
Amazon ML
Amazon Redshift
presto
Impala
Spark
HIVE
Pig
Spark Streaming
APACHE STORM
Amazon Kinesis
AWS Lambda
Amazon Elastic MapReduce
快
慢
快
Predictions
分析 & 可视化
IDE Notebooks
Amazon QuickSight
tableau
looker
TIBCO Jaspersoft
MicroStrategy
kibana
Flot
Apache Zeppelin
IPython IP[y]: Interactive Computing
RStudio
Apps & APIs

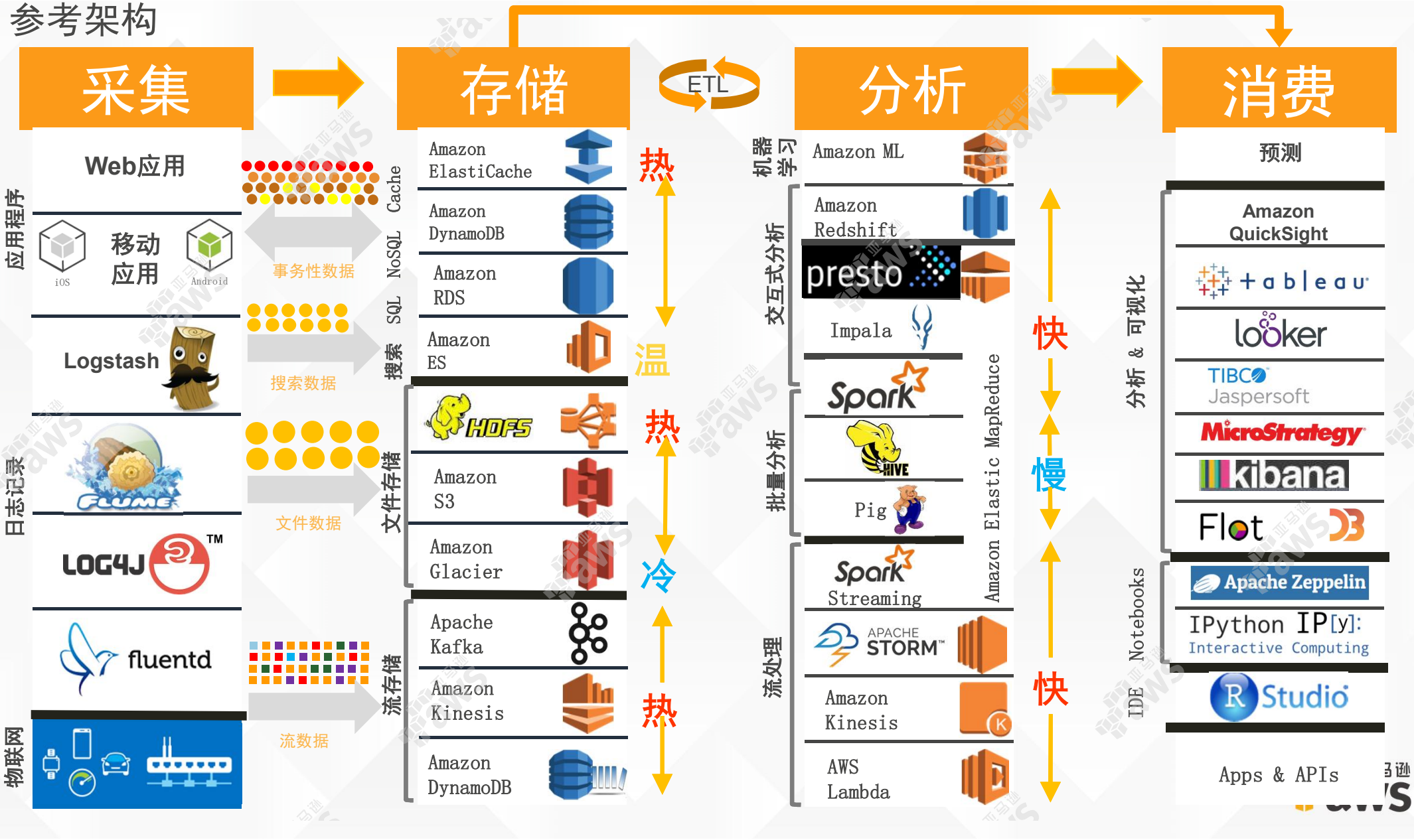
消费使用

- 预测
- 分析和可视化
- Notebooks
- 集成开发环境
- 应用 & API



完整参考架构

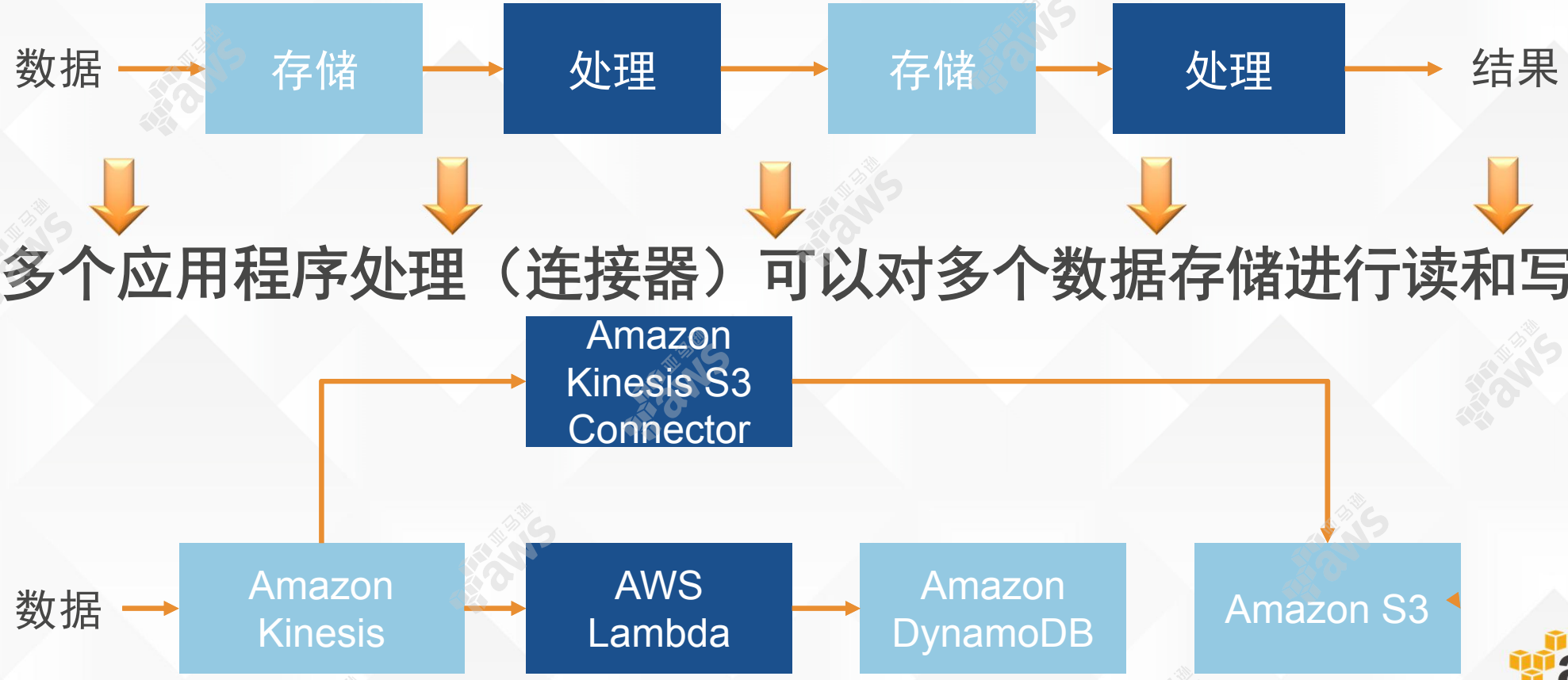
参考架构


采集
存储
ETL
分析
消费
应用程序
Web应用
移动应用
iOS
Android
日志记录
Logstash
FLUME
LOG4J
fluentd
物联网
事务性数据
搜索数据
文件数据
流数据
Cache
NoSQL
SQL
搜索
文件存储
流存储
Amazon ElastiCache
Amazon DynamoDB
Amazon RDS
Amazon ES
HDFS
Amazon S3
Amazon Glacier
Apache Kafka
Amazon Kinesis
Amazon DynamoDB
热
温
热
冷
热
机器学习
交互式分析
批量分析
流处理
Amazon ML
Amazon Redshift
presto
Impala
Spark
HIVE
Pig
Spark Streaming
APACHE STORM
Amazon Kinesis
AWS Lambda
Amazon Elastic MapReduce
快
慢
快
预测
分析 & 可视化
Amazon QuickSight
tableau
looker
TIBCO Jaspersoft
MicroStrategy
kibana
Flot
Notebooks
Apache Zeppelin
IPython IP[y]: Interactive Computing
IDE
RStudio
Apps & APIs

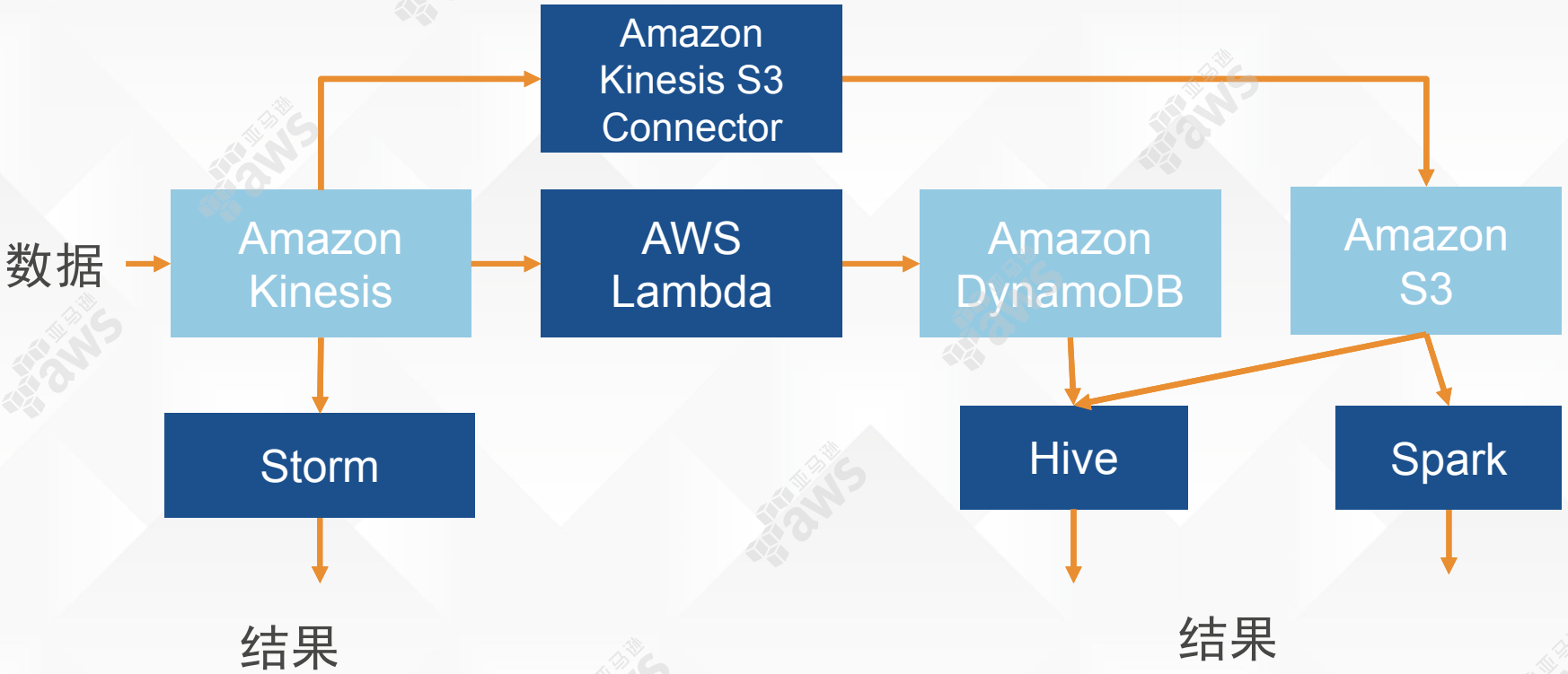
设计模式

多阶段解耦的“数据总线（Data Bus）”

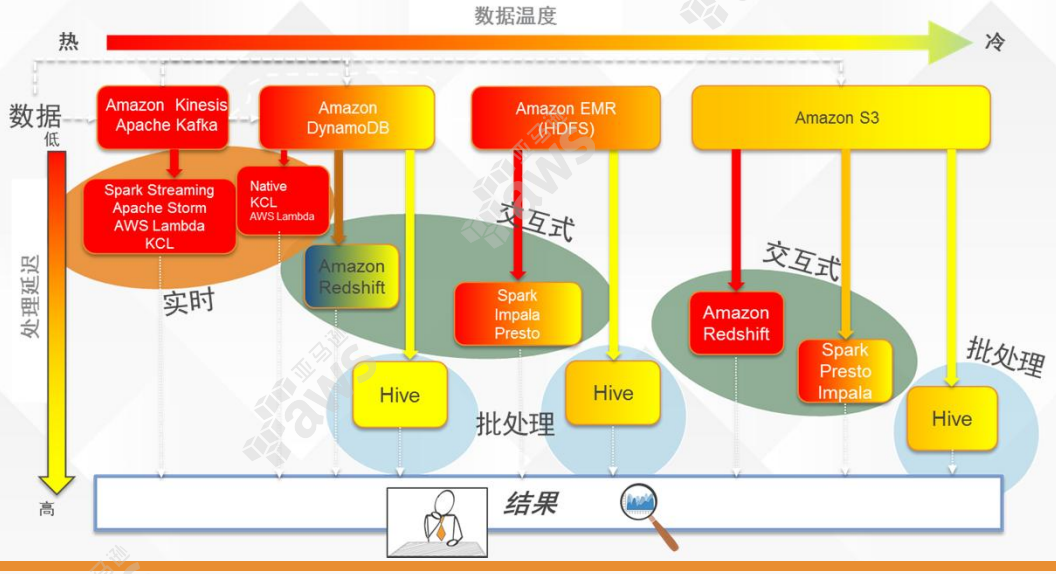
- 多阶段
- 数据存储与数据处理解耦合



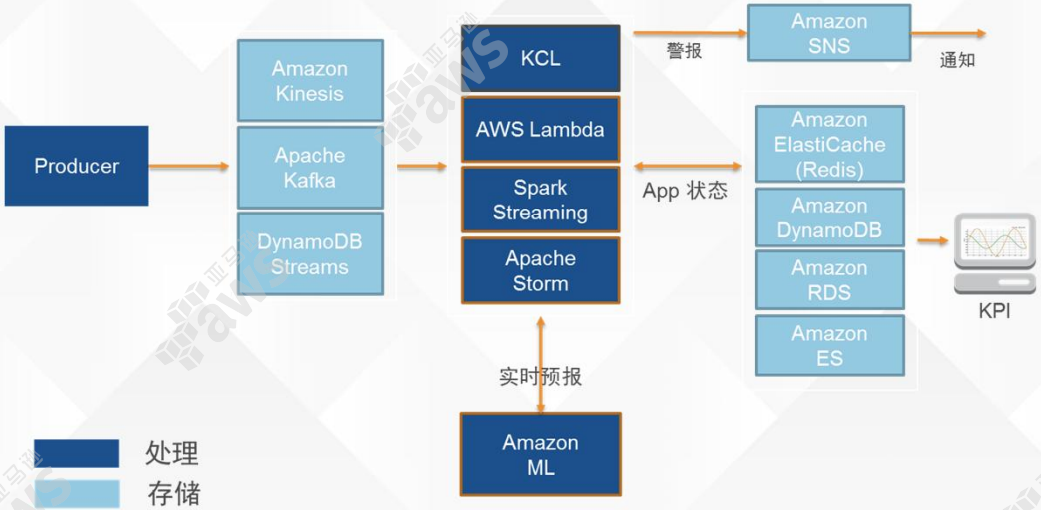
处理框架 (KCL, Storm, Hive, Spark, etc.) 能够从多个数据存储上读取数据



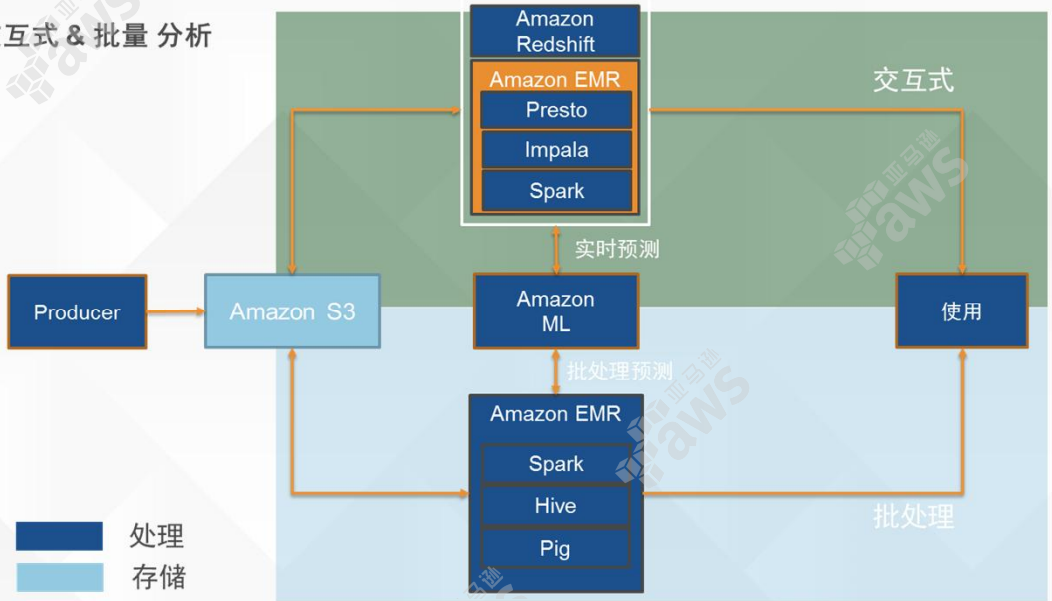
数据温度 VS 处理延迟



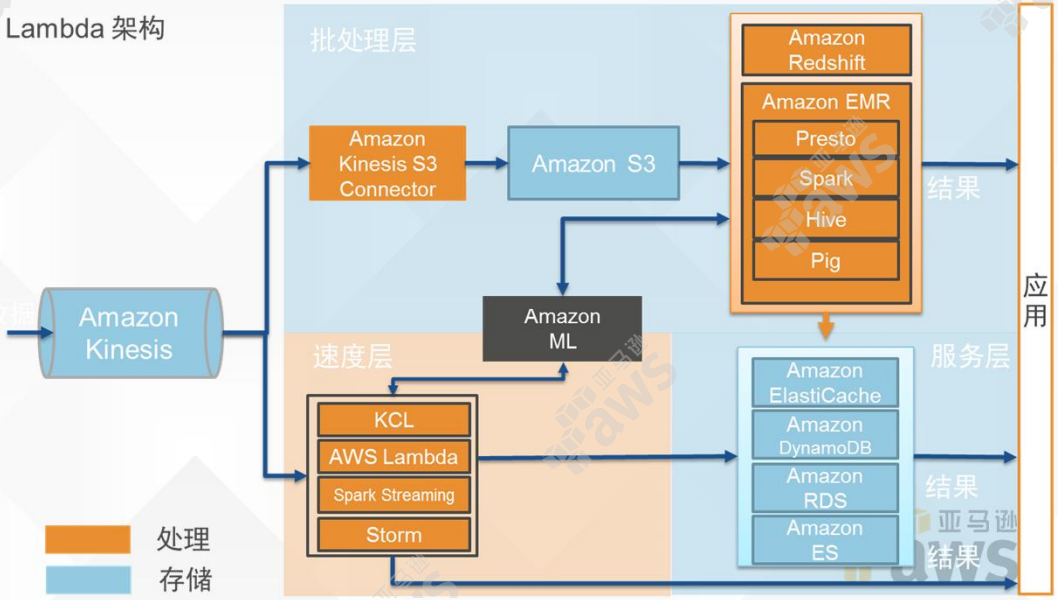
实时分析

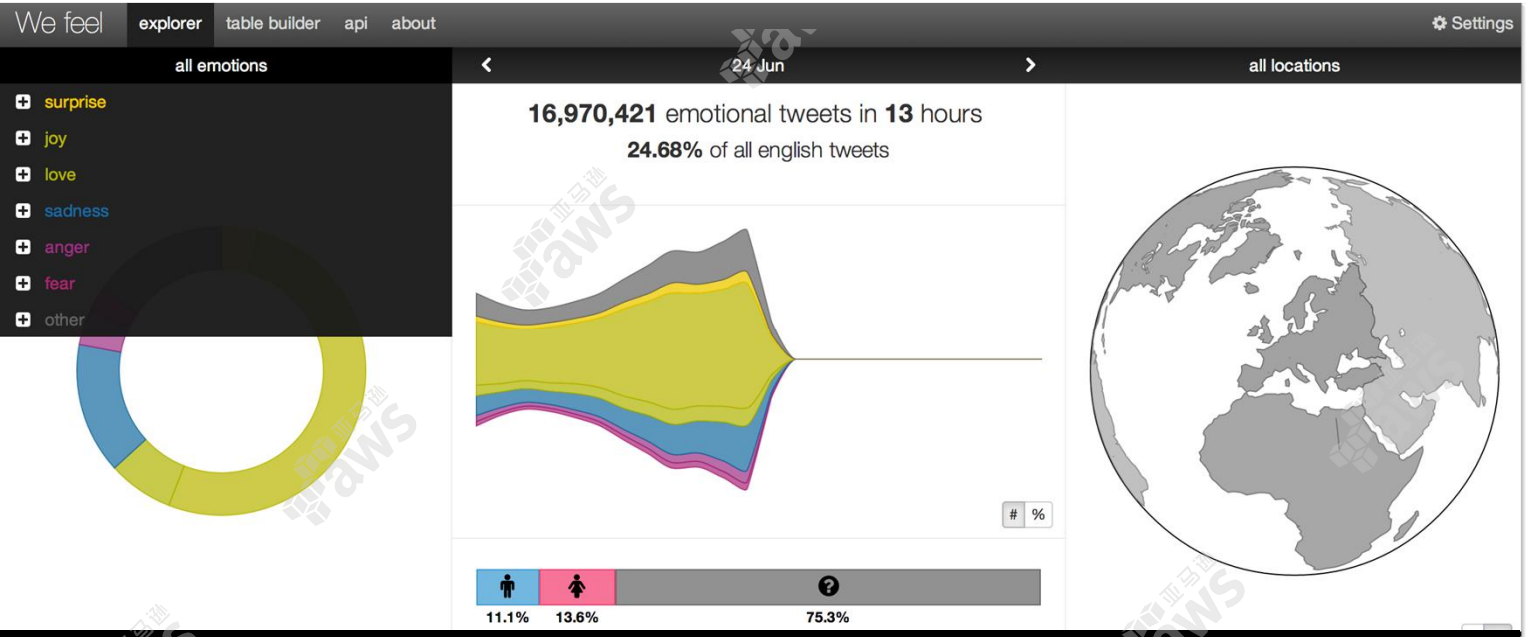


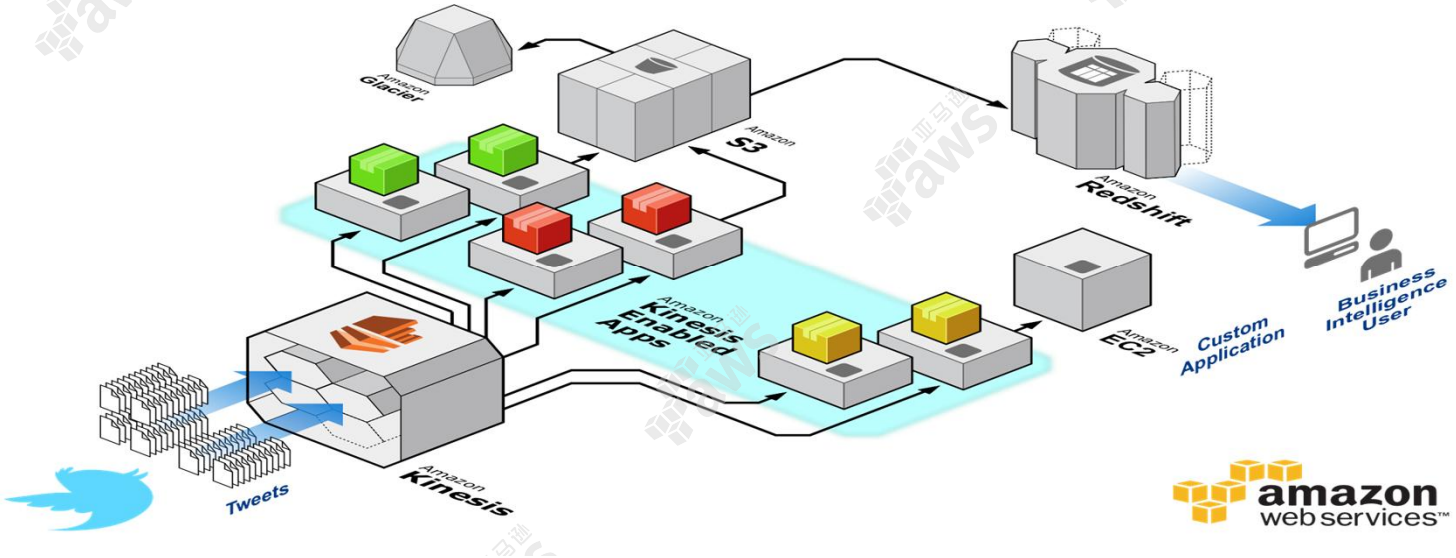
交互式 & 批量 分析



Lambda 架构







成本 & 规模

- 500M 推文/日 = ~ 20.8M 推文/小时 = ~ 5.8K 推文/秒
- 每条推文2KB，就是 ~12MB/秒, ~1TB/日（需6个分片）
- 每个分片成本：$0.015/小时，$0.028/百万PUTS
- Kinesis成本： $0.765/小时
- Redshift成本： $0.850/小时 (以 2TB dw1.xlarge

Total: $1.615/hour

Case Study

亚马逊AWS机器学习

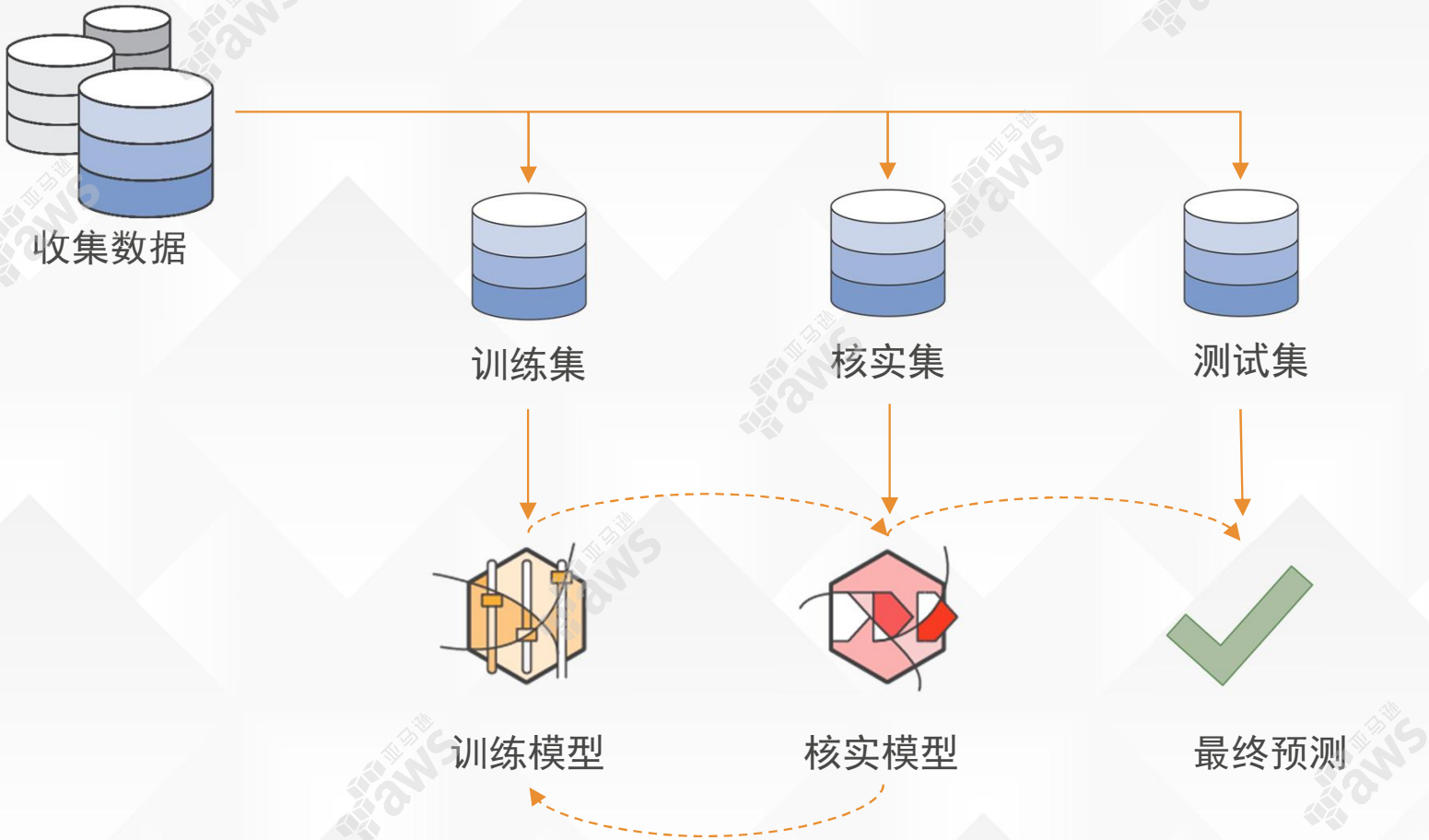
亚马逊aws

机器学习

机器学习 – 过程

- 从已经知道答案的数据开始
- 确定目标 – 你想通过数据预测什么
- 选取用来识别模式的变量或特征，进而预测目标
- 通过已知目标答案的数据集，训练机器学习模型
- 用训练好的模型来给未知目标变量的数据做预测
- 评价模型的准确率
- 提高模型的准确率

机器学习



机器学习经典算法

推荐

协同过滤CF(基于用户、基于项目)
斜坡算法　　SVD++
矩阵分解(Matrix Factorization w/ ALS)

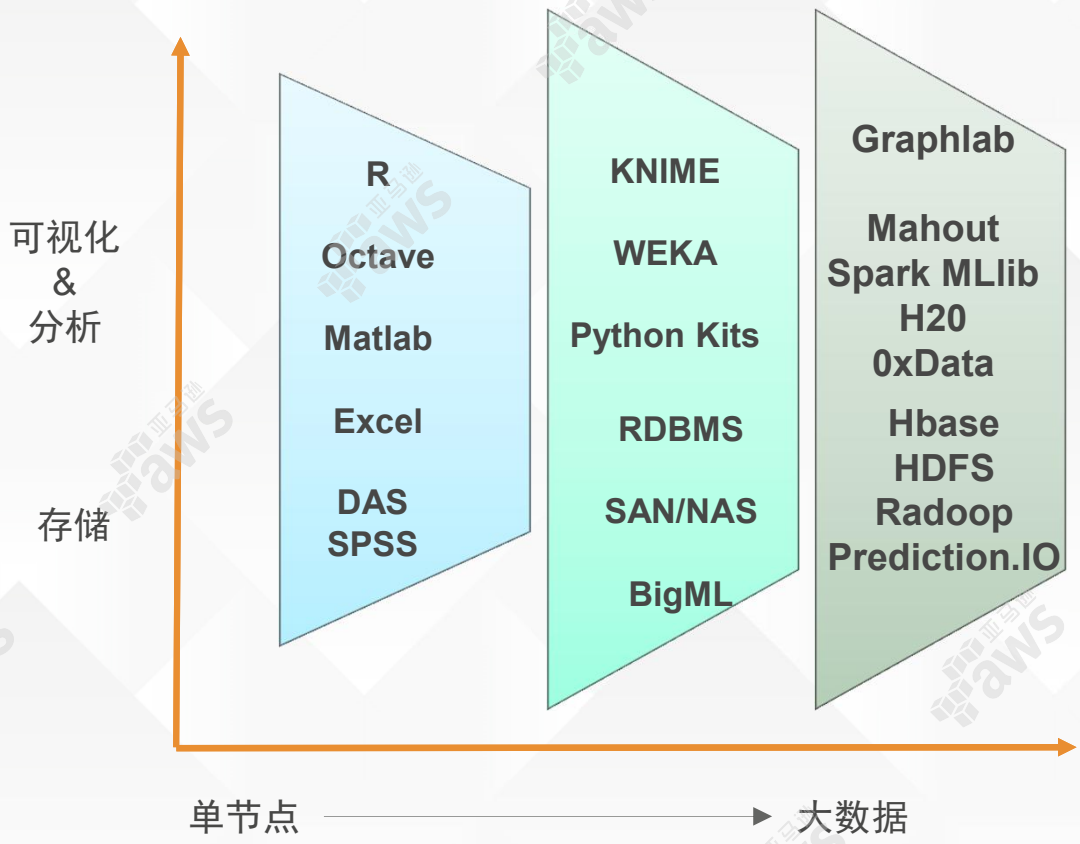
聚类

聚类算法(Canopy、 K-Means)
模糊　　流式　　光谱分析

分类

决策树(Decision Trees)
线性回归(Linear Regression)
逻辑回归(Logistic Regression)
贝叶斯模型(Naïve Bayes)
随机森林算法(Random Forest)
隐马尔可夫模型(Hidden Markov Models)
多层感知器(Multilayer Perceptron)

AWS平台上的机器学习



机器学习的弹性伸缩

- 用正确的工具做正确的事情

Amazon ML介绍

为开发人员打造的，简单易用的机器学习服务

- 通过直观而强大的服务控制台来发现和建构学习模型
- 通过全功能的API和SDK来完成模型生命周期的自动化管理 - Java, Python, .NET, JavaScript, Ruby, Javascript
- 通过AWS Mobile SDK快速建构 iOS ， Android智能应用

基于Amazon内部系统的，健壮的，强大的机器学习技术

- 基于Amazon内部众多经过实战考验的系统
- 不仅仅是算法:智能数据转换、输入数据的质量警告以及模型的质量警告、内置的业界最佳实践
- 和AWS的数据生态系统完美集成：S3、Amazon Redshift、RDS MySQL、IAM

完全托管的模型和预测服务

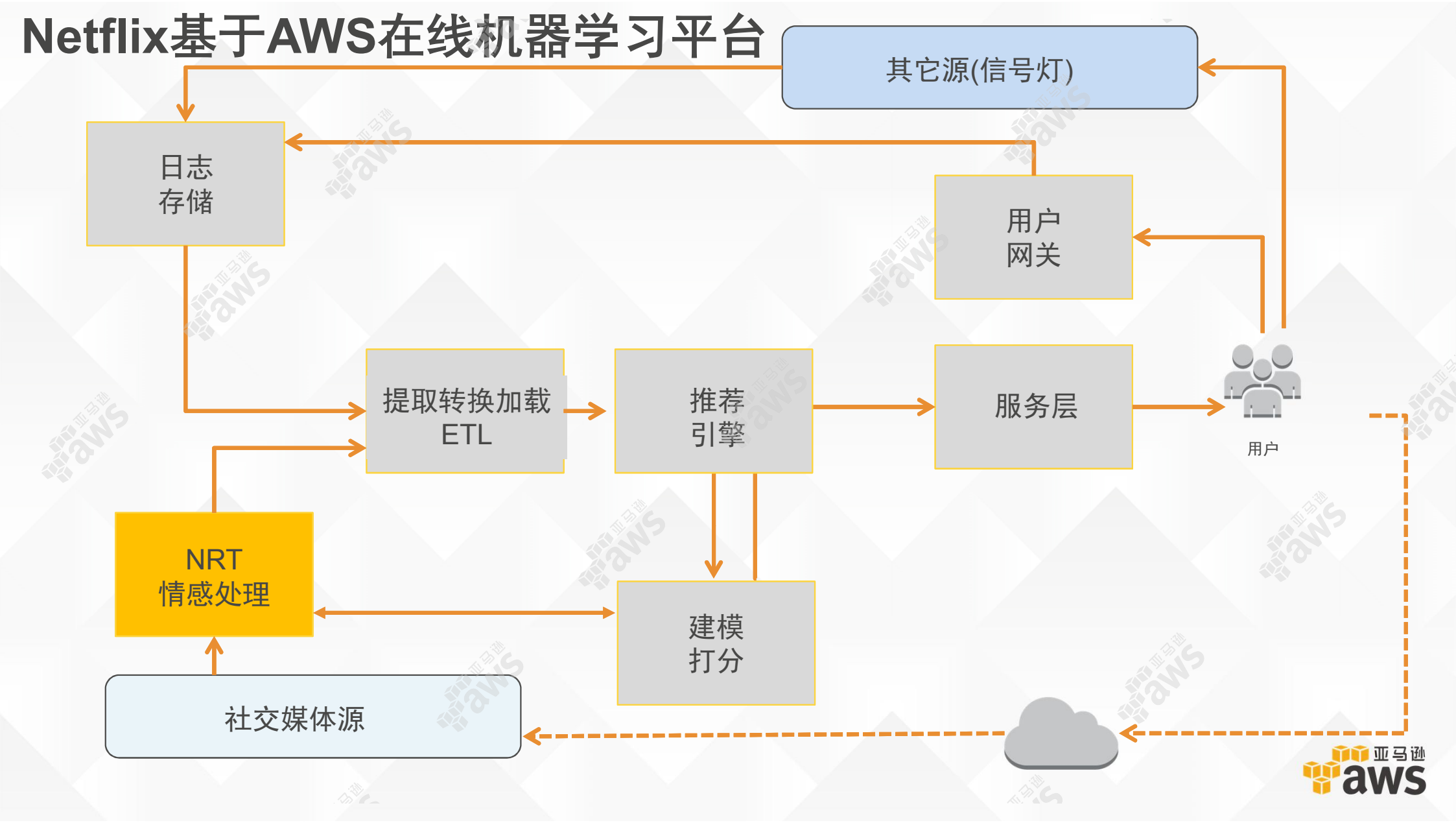
- 端到端的服务，不需要对底层服务器进行管理
- 预测模型一键部署
- 可以通过程序获得模型的元数据，使数据获取流程自动化成为可能
- 可以通过Amazon CloudWatch监控预测使用模式

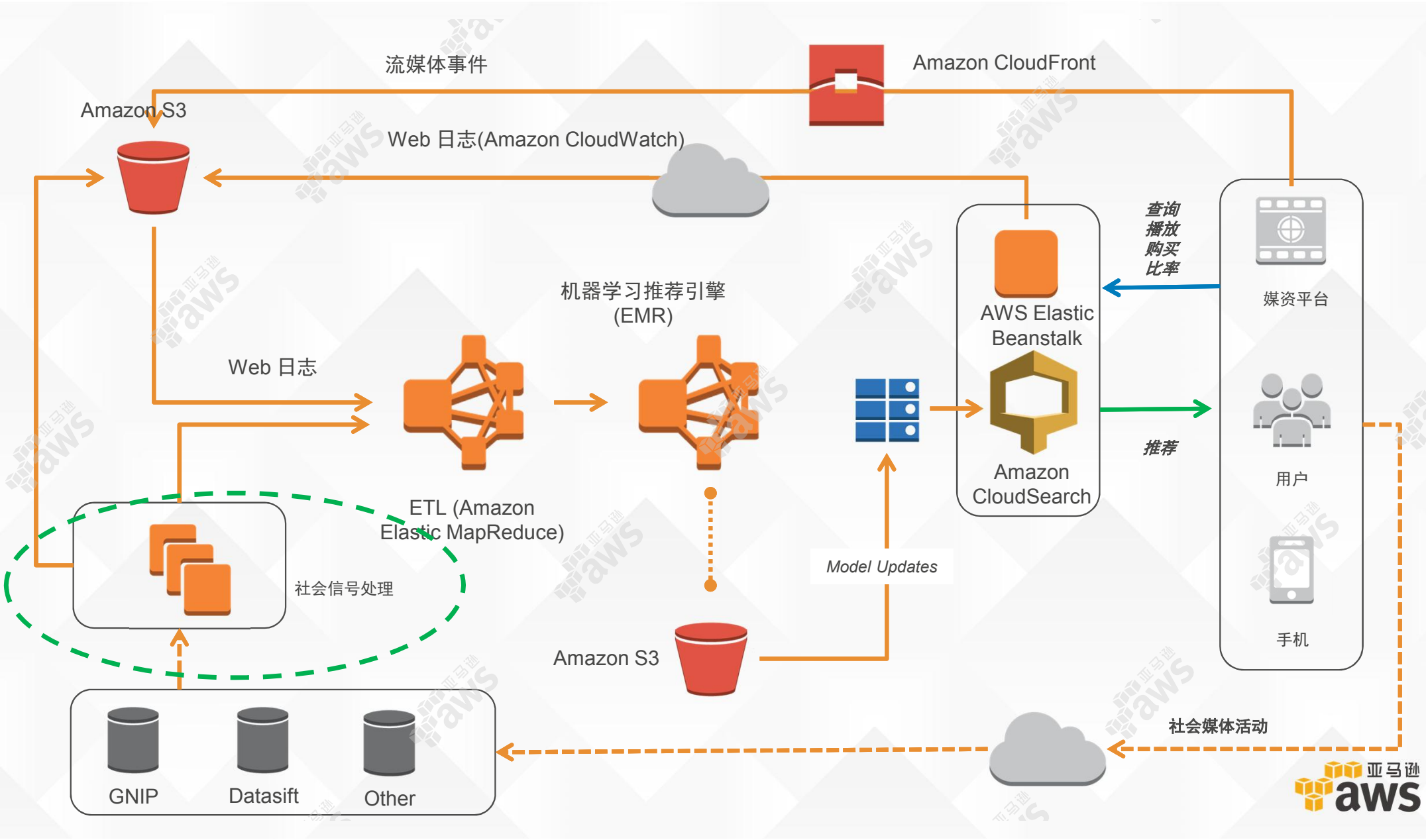
按使用量付费，非常便宜

- 数据分析，模型训练和校验: $0.42/实例小时
- 批量预测: $0.10/1000条
- 实时预测: $0.10/1000条

<每小时的容量预留费用>

Netflix基于AWS在线机器学习平台




流媒体事件
Amazon CloudFront
Amazon S3
Web 日志(Amazon CloudWatch)
查询
播放
购买
比率
媒资平台
机器学习推荐引擎
(EMR)
AWS Elastic
Beanstalk
Web 日志
推荐
用户
ETL (Amazon
Elastic MapReduce)
Amazon
CloudSearch
社会信号处理
Model Updates
手机
Amazon S3
社会媒体活动
GNIP
Datasift
Other
亚马逊
aws

演示 — 客户评分系统

客户评分系统是用来评估客户的潜在价值，以实现精确营销的智能系统。它根据历史数据训练机器学习模型，对客户进行打分，分数高的客户就被认为是更有可能产生利润的客户。

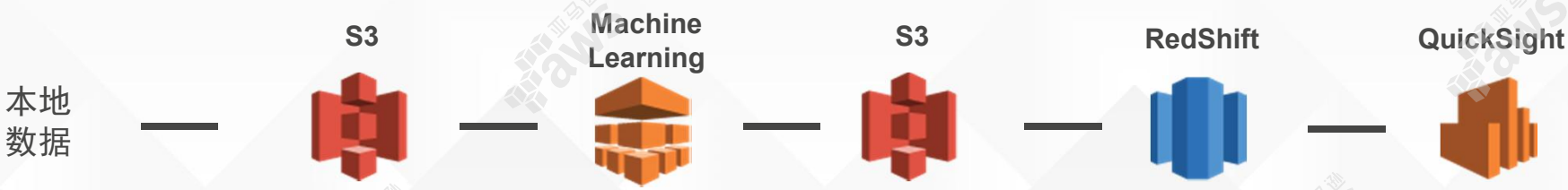
原始数据样例：

Create_Date	First_Name	Last_Name	Title	Company	Phone	Email	Email_Option	City	Province	Industry	Job	Attend_Act_1	Attend_Act_2	Member_Status
2014/10/9	Liang	Li	CEO	**Company	*******0941	123456789@qq.com	1	Beijing	Beijing	Software & Internet	Director	0	1	hot
2015/4/3	小明	王	工程师	**软件有限公司	*******3545	wangxiaoliang@163.com	0	深圳	广东	Telecommunications	Developer / Engineer	1	0	cold

结构化后的数据样例：

预测目标

Create_Date	First_Name	Last_Name	Title	Company	Phone	Email	Email_Option	City	Province	Industry	Job	Attend_Act_1	Attend_Act_2	Member_Status	Account_Status
2014/10/9	Liang	Li	CEO	**Company	*******0941	@qq.com	1	BEIJING	BEIJING	Software & Internet	Director	0	1	hot	1
2015/4/3	xiaoming	wang	gong cheng shi	**you xian gong si	*******3545	@163.com	0	SHENZHEN	GUANGDONG	Telecommunications	Developer / Engineer	1	0	cold	0



客户评分系统：逻辑回归与随机梯度法

逻辑回归模型：

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1+e^{-\theta^T x}}$$

$$P(y=1 \mid x;\theta) = h_\theta(x)$$

$$P(y=0 \mid x;\theta) = 1 - h_\theta(x)$$

似然函数：

$$L(\theta) = \prod_{i=1}^{m} P(y_i \mid x_i;\theta) = \prod_{i=1}^{m} (h_\theta(x_i))^{y_i} (1-h_\theta(x_i))^{1-y_i}$$

取Log：

$$l(\theta) = \log L(\theta) = \sum_{i=1}^{m} \left(y_i \log h_\theta(x_i) + (1-y_i) \log(1-h_\theta(x_i)) \right)$$

随机梯度法求解

For 循环次数 {
    For 每一个 data_i {

$$\theta_j := \theta_j - \alpha (h_\theta(x_i) - y_i) x_i^j$$

    }
}

客户评分系统：Amazon ML的应用流程

训练

创建数据资源：训练集

- 将结构化的训练数据从S3导入Amazon ML。
- 设定数据各维度的**类型**，包括数值型，布尔型，类别型和文本型。
- 指定**目标维度**：数值型（回归），布尔型（二分类），类别型（多分类）。

模型建立

- 设定训练模型时用到的**特征**，并指明各个特征的**处理方式**，如特征的合并，文本型特征的N-Gram处理，数值型特征的归一化等。
- 设定模型的复杂度及**正则化**方式。
- 设定训练集与核实集的比例，默认用70%的数据做训练。

模型评估

- Amazon ML的二分类模型通过**召回率**，**精确度**，**准确率**，**AUC**值等指标来评估模型。
- 在客户评分系统中，我们关注的是召回率，所以我们将根据召回率来确定我们的评分阈值。
- 这里，我们分别找出召回率在90%，95%，99%时的评分阈值，以便预测。

预测

创建数据资源：测试集

- 将结构化的测试数据从S3导入Amazon ML。
- 测试数据的每个维度应与训练数据有相同的数据类型。

模型预测

- 指定**模型ID**和**测试数据资源**，即可开始预测。
- 将预测结果（即客户的评分）输出到S3存储桶。

客户评分系统：Amazon ML模型训练核心代码

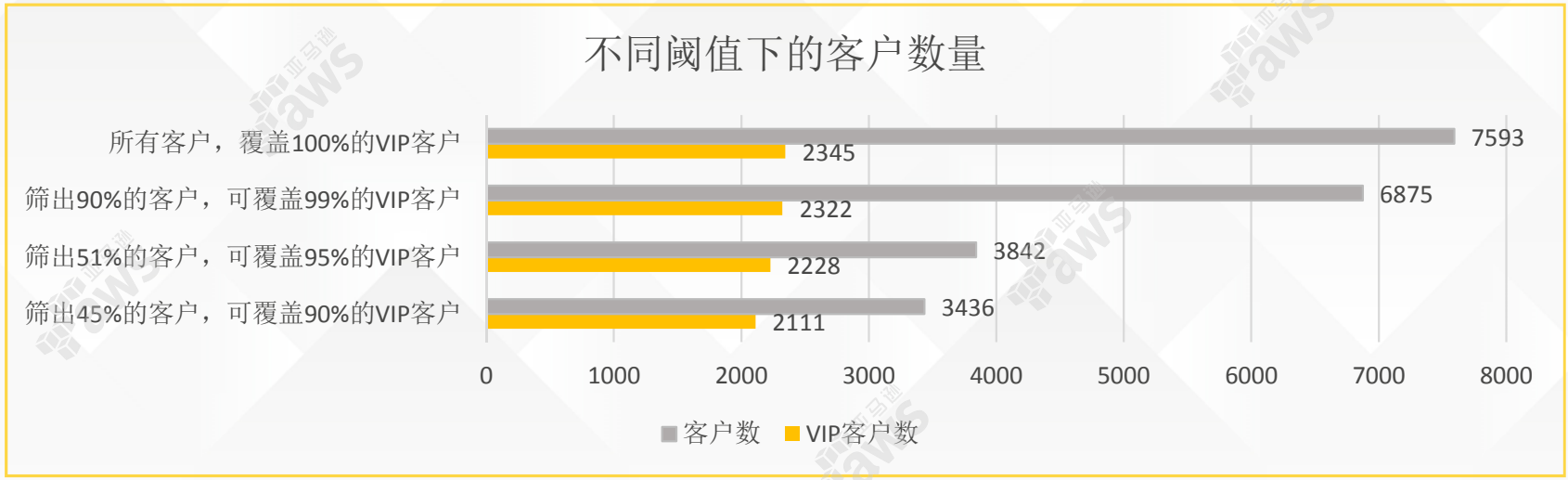
```
String dataSchemaPath = “data/leadsInfo.csv.schema.txt”;   // schema文件路径，schema用来设定结构化的数据各维度的数据类型
String recipePath = “data/recipe.txt”;   // recipe文件路径，recipe用来设定训练模型的时候，各维度的处理方式

AWSCredentials credentials = new ProfileCredentialsProvider("default").getCredentials();
AmazonMachineLearning ml = new AmazonMachineLearningClient(credentials);
ml.setRegion(Region.getRegion(Regions.US_EAST_1));

CreateDataSourceFromS3Request createDataSourceFromS3Request = new CreateDataSourceFromS3Request();
String id = "k3y0";
createDataSourceFromS3Request.setDataSourceId(“ds-SummitMLdemo-train-”+id);   //  命名训练数据资源的id
S3DataSpec dataSpec = new S3DataSpec();
dataSpec.setDataLocationS3(“s3://bucketName/SummitMLdemo/leadsInfo-Structured.csv”);   //  结构化数据在S3存储桶中的位置
dataSpec.setDataSchema(ReadtoString.jsonStr(dataSchemaPath));   //  设定schema
dataSpec.setDataRearrangement(“{\”splitting\“:{\”percentBegin\“:0,\”percentEnd\“:70}}”);   //  用70%的数据进行训练
createDataSourceFromS3Request.setDataSpec(dataSpec);
ml.createDataSourceFromS3(createDataSourceFromS3Request);   //  生成Amazon ML支持的训练数据资源

CreateMLModelRequest createMLModelRequest = new CreateMLModelRequest();
createMLModelRequest.setMLModelId(“ml-SummitMLdemo-”+id);   //  命名训练模型id
createMLModelRequest.setMLModelType(“BINARY”);   //  设定模型种类
Map<String, String> modelParameters = new HashMap<String,String>();
modelParameters.put(“sgd.l2RegularizationAmount”, “1.0E-03”);   //  设定正则化方式
modelParameters.put(“sgd.maxMLModelSizeInBytes”, “1000000000”);   //  设定模型复杂度
modelParameters.put(“sgd.maxPasses”, “20”);   //  设定随机梯度下降的循环次数
createMLModelRequest.setParameters(modelParameters);
createMLModelRequest.setRecipe(ReadtoString.jsonStr(recipePath));   //  设定recipe
createMLModelRequest.setTrainingDataSourceId(“ds-SummitMLdemo-train-”+id);   //  设定训练数据资源的id
ml.createMLModel(createMLModelRequest);   //  开始训练模型
```

客户评分系统：模型成果展现



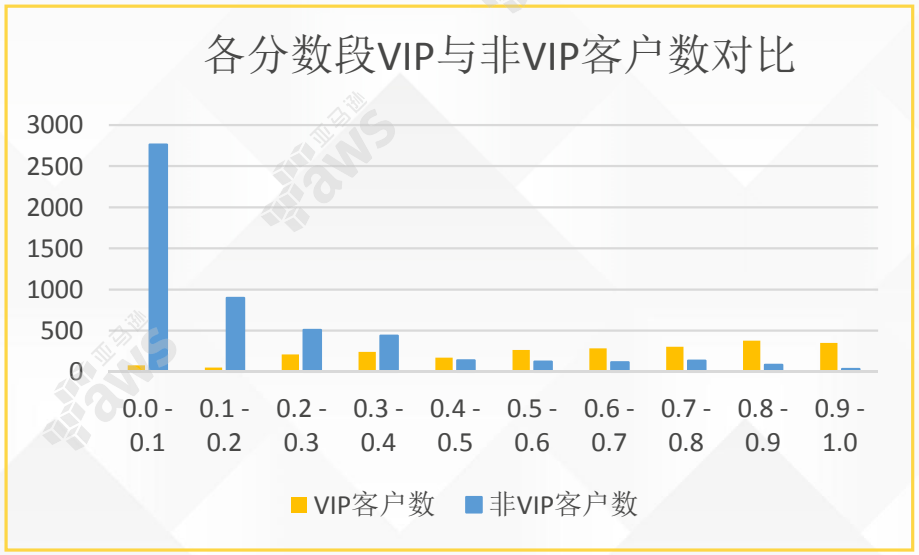
不同阈值下，VIP客户比例

31%

34%

58%

61%



有奖问答

1.

问：请举出一种用于流存储的AWS托管服务？

答：Amazon Kinesis 或 DynamoDB Streams 或 Amazon SQS 或 Amazon SNS

2.

问：AWS云端基于事件驱动的“无服务器”托管服务叫什么？

答：AWS Lambda

3.

问：AWS最新推出的全托管的搜索服务叫什么？

答：Amazon Elasticsearch Service

Thank You