



集算器

创新大数据计算引擎

报表后端计算的利器

润乾软件出品



目录

Contents

1

公司介绍

2

现状分析

3

解决方案

4

核心优势

5

集算器技术特征

6

应用案例



COMPANY POSITIONING

公司概况及定位

北京润乾成立于2000年6月，是北京市海淀高科技园区内的**高新技术企业**、国内知名软件企业，拥有一支实力强劲的产品研发团队。

北京润乾**十八年**始终坚持**创新技术推动应用进步**的经营理念，是中国最成熟的数据计算与数据呈现**基础软件提供商**，为客户提供国际一流的**通用软件产品**，我们的产品已经得到了广大开发商用户的认可和应用，在金融、电信、能源、政府、军工等多个行业，已积累了20000多成功案例。

- ◆2015福布斯中国非上市潜力企业100强
- ◆2016年中国软件和信息服务业·十大领军人物（蒋步星）
- ◆2016年度中国大数据 - 数据计算领域最佳产品奖（集算器）
- ◆2016年度 - 大数据领域创新产品奖（集算器）
- ◆2017年4月集算器被中国软件评测中心测定为“优秀大数据基础设施类产品”
- ◆2017年6月集算器获“航天云网”软件应用创新大赛一等奖，润乾报表V5.0获二等奖
- ◆2017年8月中国大数据企业50强！

详细介绍见：<http://www.raqsoft.com.cn/about>



- 2000 ○ 6月，润乾公司成立
- 2002 ○ 推出润乾报表V1.0-V2.0，建立了对国外同类软件的竞争优势
- 2004 ○ 2月，推出革命性的润乾报表V3.0,研发了独创的**非线性报表模型**，实现了对国外产品的技术超越，并逐渐形成解决中国式报表的技术标准
- 2008 ○ 内核重构，大幅提升性能的**润乾报表V4.0**；并开始研发**数据计算引擎-集算器**
- 2010 ○ 推出润乾报表V4.5,增强多维分析功能，提升企业级性能优化
- 2014 ○ 历经4个版本，正式发布**面向大数据的新一代数据计算引擎——集算器**，并朝拥有自主产权的非关系型强计算数据库仓库/云数据库产品迈进
- 2015 ○ 润乾软件被福布斯中文网站评为“2015福布斯中国非上市潜力企业100强”
- 2017 ○ 荣获中国大数据产业生态联盟评选的“**2017年中国大数据企业50强**”。

....



DEVELOPMENT PROCESS

发展历程 情怀与坚持



HONOURS

见证成长的荣耀之路



PATENT

知识产权



10个专利证书



21条著作权



润乾报表软件 (v6.0)

润乾超维报表软件 (v5.0)

集算报表软件 (v5.0)

集算器软件 (v3.0)

计算表软件 (v2.0)

.....

目录

Contents

1

公司介绍

2

现状分析

3

解决方案

4

核心优势

5

集算器技术特征

6

应用案例



现状分析

报表业务稳定性差，开发没完没了

这是常态，要试图适应而不能解决，建立长期应对机制

自助报表并不管用

只能解决10%-20%的需求

零编码制作报表只是一句口号

只限于呈现环节

报表开发中的二八原则

复杂报表在数量上只占二成，但工作量却占八成





现状分析

- ✓ 成熟的报表工具已经能解决**呈现**环节问题
- ✓ 报表开发更多的困难在**数据源**上
- ✓ 大多数性能问题也是**数据源**造成的或者需要由数据源解决
- ✓ 好的数据源处理机制还能**优化应用结构**



现状分析-具体场景



场景	描述与举例
报表特殊格式	报表工具无法实现，子表横向插入主表，多层动态表，横向分栏，补足空行
动态图形/水印/签章	普通报表工具能实现静态图形/水印/签章，实现动态需要自定义类，调试、扩展、维护均不方便
动态数据源/集	根据传参动态连接数据库，动态拼接SQL并对结果集容量进行一定控制
报表过程/关联计算	利用大量隐藏格保存中间结果，变相的/很费力的实现复杂的过程式计算，且性能体验很差
多样性数据源	关系型数据库，文本/文件数据类型，Hadoop，NoSQL，HTTP、ALI-OTS，...
自定义数据集	单纯SQL/存储过程一次逻辑算不完，需要借助JAVA二次加工或反复几轮调用混算后的结果集
大量存储过程/中间表	为业务查询而准备，报表应用与数据库耦合度高、管理混乱、数据库I/O瓶颈，扩容成本高
报表慢/内存溢出	日积月累，数据量上来了，报表访问越用越慢，经常内存溢出，甚至导致服务器宕机
多源跨库计算	实现跨多种类型数据源查询困难，且不分同/异构。比如：Oracle+MSSQL、数据库+Excel
T+0混合计算	要求数据库具有跨库运算能力，但实施复杂度较高，性能较低；当数据库类型不同时难以实现
大清单列表	清单式千万行大报表展现/打印/导出，用户体验差，甚至内存溢出，采用数据库分页方式性能差
离线ETL辅助	自己编码太累，专业etl工具来干又过于浪费，并且有一定的局限性、不够灵活、成本过高等
报表数据准备困难	报表工具解决展现层，但计算层还需编程(Java/复杂SQL/存储过程等)，且需专业程序员
应用耦合度高	为报表数据计算编写的代码，和应用系统的其他部分耦合在一起。一旦调整，且需专业程序员
SQL/存储过程难写	较复杂的计算逻辑，SQL/存储过程难写、难调试，低成本维护人员没办法自主完成
中间件	是否要求具有良好的可管理性、集成性、扩展性、开放性？

目录

Contents

1

公司介绍

2

现状分析

3

解决方案

4

核心优势

5

集算器技术特征

6

应用案例

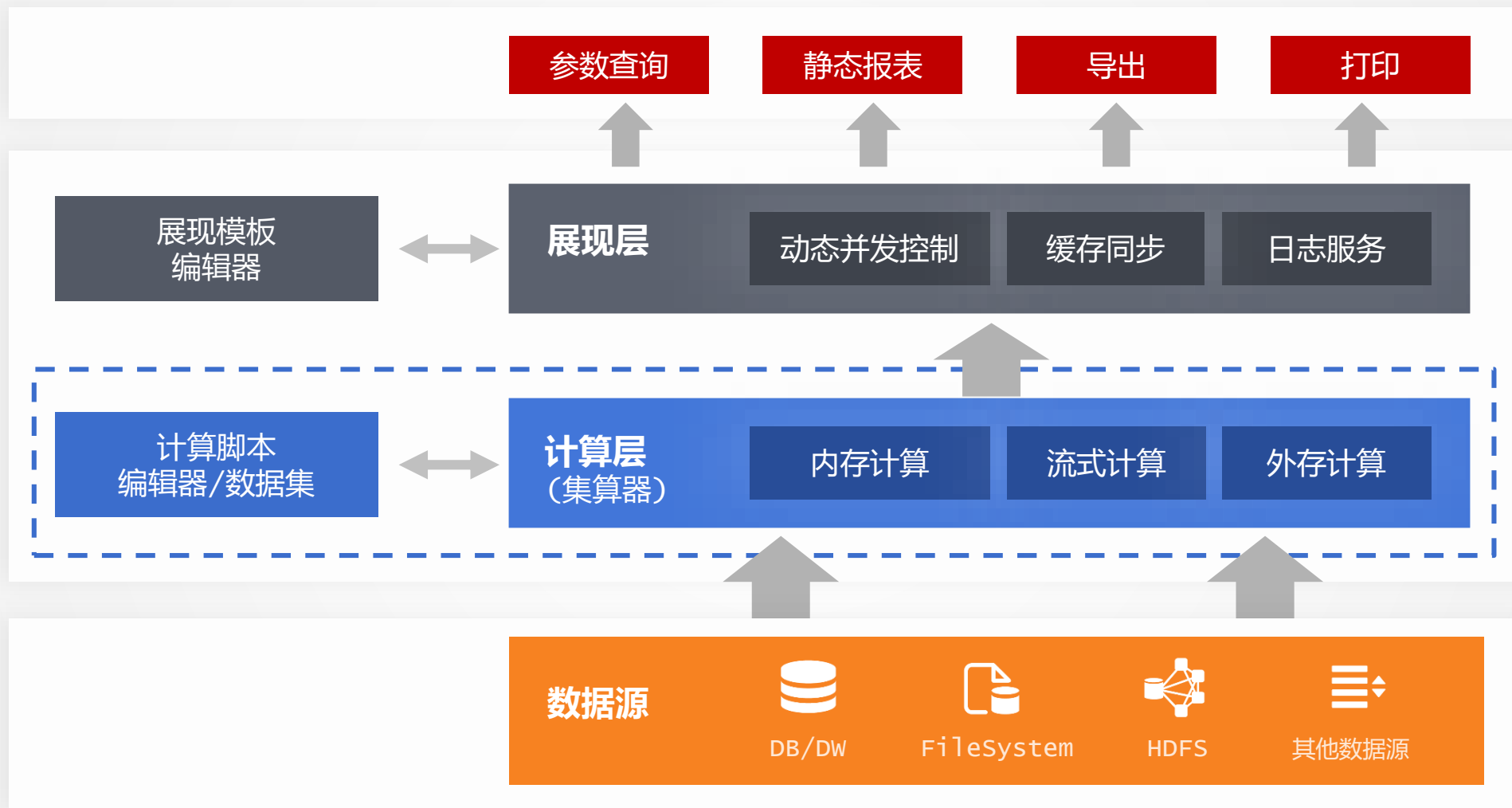
A person wearing a blue suit jacket and a white shirt is standing next to a laptop. The person's hands are in their pockets. The background is slightly blurred, showing what appears to be an office or workspace.

集算器

轻松为您解决数据计算难题！

- 1、一款面向应用程序员和数据分析员，专注于**结构化**数据分析与处理的**快速**开发工具。
- 2、一套基于Java**解释执行**的动态语言，采用了**先进**的计算**模型**和设计思想，让开发**更易**于实现且性能更好。
- 3、一个完备的类库和**轻量级**架构，让工程应用更灵活、更高效。

引入数据计算层-集算器





引入数据计算层-工具化、独立化

开发工具化

- 1、不仅呈现层要工具化，报表数据计算层也要工具化，降低对开发人员要求。
- 2、不必做复杂的环境配置(数据源等)，可编写简单代码实现复杂计算。
- 3、多样性数据源(比如EXCEL/文本)，也必须支持简单脚本计算。

模块独立化

- 1、数据计算层也要和呈现层一样，完全和应用系统解耦合，实现独立维护。
- 2、报表需求变更和新增时，仅修改报表模块就可以了，不会影响应用系统其他部分。

目录

Contents

1

公司介绍

2

现状分析

3

解决方案

4

核心优势

5

集算器技术特征

6

应用案例

核心优势



降低开发难度



提高运算性能



优化应用结构



大数据报表



过程计算

报表采用直观的状态式计算

不合适进行过程计算

过程计算在复杂报表中很普遍

采用隐藏格

采用Java或存储过程



过程计算举例

销售额占前一半的大客户

	A	B	C
1	客户	金额	=ds.sum(金额)/2
2	=ds.select(客户)	=ds.金额	=C2[-1]+B2
3	大客户数量	=count(B2{C2[-1]<C1})	
4	平均销售额	=avg(B2{C2[-1]<C1})	

隐藏格：C列要隐藏，且在第2行设置显示条件为C2[-1]<C1。不能用更简单地C2<=C1，否则刚跨过一半销售额的那一行会出错
条件会被计算多次，还要用到格集过滤等报表工具特有功能

舍位平衡

四舍五入后明细与合计不一致，需要根据合计重新舍入明细



集算器实现

先准备出数据集

	A	B
1	=db.query(“select 客户,金额 from 客户销售 order by 金额 desc”)	
2	=A1.sum(金额)/2	=0
3	=A1.pselect((B1+=金额)>=A2)	return A1.to(A3)

再做报表就很简单

	A	B
1	客户	金额
2	=ds.select(客户)	=ds.金额
3	大客户数量	=ds.count()
4	平均销售额	=ds.avg(金额)

多了一步，但更为清晰；

任务报表工具都可以完成



比JAVA的优势

集算器采用**集合化**语法，代码要比没有直接提供结构化计算的JAVA更加**短小**

写的更快更短

- 集算器基于Java提供了更高层的类库和方法

容易理解和排错

- 伪实代码的比例大约是只有1:1.5，大多数报表数据准备算法可以在**一个屏幕内**显示出来
- 一个页面内能看到更多代码，能更完整地理解代码的含义与排错



比SQL/存储过程的优势

销售额占前一半的大客户

1	SELECT CUSTOMER, AMOUNT, SUM_AMOUNT
2	FROM (SELECT CUSTOMER, AMOUNT,
3	SUM(AMOUNT) OVER(ORDER BY AMOUNT DESC) SUM_AMOUNT
4	FROM (SELECT CUSTOMER, SUM(AMOUNT) AMOUNT
5	FROM ORDERS GROUP BY CUSTOMER))
6	WHERE 2 * SUM_AMOUNT < (SELECT SUM(AMOUNT) TOTAL FROM ORDERS)

分步计算，调试开发更方便

离散性支持下实现更彻底的集合运算和有序运算



多样性数据源支持

报表工具的计算能力和容量都难以胜任多样性数据源

计算层处理多样性数据源

无需入库，减少工作步骤

支持多层数据格式，减少开发工作量





动态数据源/集

动态数据源

根据参数决定连接的数据库 `${pds}.query("select * from T where F=?", pF)`

动态数据集

动态SQL，需要程序逻辑来拼接

	A	
1	<code>=sums.array().("sum("+~+") as "+~).string()</code>	/把a,b变成sum(a) as a,sum(b) as b
2	<code>=db.query("select G,"+A1+" from T group by G")</code>	

结果集容量控制

	A	B	
1	<code>=db.cursor("select * from T")</code>	<code>=A1.fetch(1000)</code>	
2	<code>if B1.fetch@0(1)</code>	<code>>B1.insert(0,"继续")</code>	/未完成则插入标记
3	<code>>A1.close()</code>	<code>return B1</code>	

特殊格式



解决报表工具不支持的格式

▶ 横向分栏

员工号	姓名	部门	员工号	姓名	部门	员工号	姓名	部门
1	Rebecca	R&D	2	Ashley	Finance	3	Rachel	Sales
4	Emily	HR	5	Ashley	R&D	6	Matthew	Sales
7	Alexis	Sales	8	Megan	Marketing	9	Victoria	HR
10	Ryan	R&D	11	Jacob	Sales	12	Jessica	Sales
13	Daniel	Finance	14	Alyssa	Sales	15	Alexis	Sales
16	Christopher	Production	17	Hannah	Marketing	18	Jonathan	Administration

	A	B	C
1	=db.query("select a,b,c from T ")		
2	=A1.step(3,1)	=A1.step(3,2) [null]	=A1.step(3,3) [null]
3	=A2.derive(B2(#).a:a2,B2(#).b:b2,B2(#).c:c2,C2(#).a:a3,C2(#).b:b3,C2(#).c:c3)		

▶ 补足空行

	A	
1	=db.query("select * from T")	
2	=pn-A1.len()%pn	/计算应补的行数
3	=A1 if(A2!=pn,A2*[null])	/补充空行数后的结果集

核心优势



降低开发难度



提高运算性能



优化应用结构



大数据报表



报表的性能问题

大部分是数据源导致的

报表本身一般是个小数据任务

数据进入报表之前就很慢， 优化报表环节意义不大

报表环节的性能也可由数据源环节解决

采用不同的计算方式

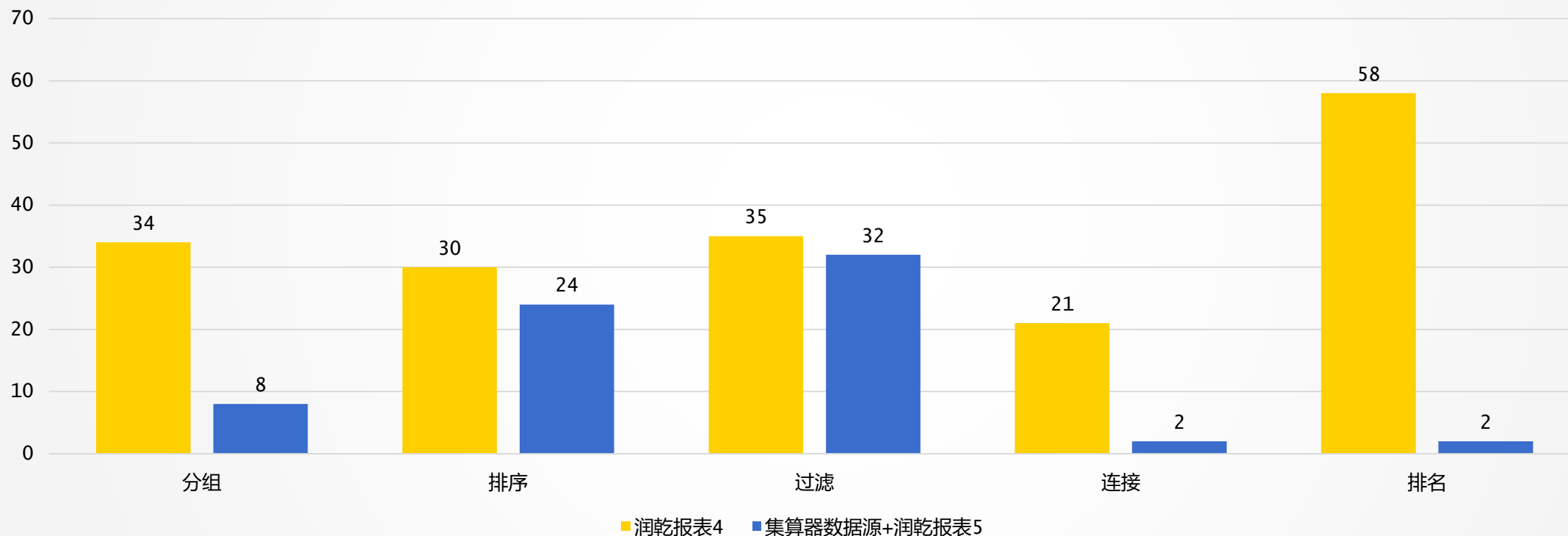
减少隐藏格

数据传输、可控缓存、内存共享

替代报表计算 - 高效算法



新/旧报表性能对比



集算器中多数数据源对齐采用Hash算法，优于报表中的遍历方式；复杂度由平方级降为线性级
这个优势可以服务于任何报表工具

替代报表计算 – 减少隐藏格



隐藏格作用

用于保存中间结果

隐藏格缺点

单元格带有展现属性会占用过多内存，出现性能瓶颈

改进方法

- ✓ 在独立的计算层中准备数据，方便复用中间结果
- ✓ 不涉及隐藏格和外观属性，内存使用效率更高



减少隐藏格实例

统计各地区前五的销售业绩，第六名以后全部归并为其他

单纯报表工具实现

	A	B	C	D	E	F
1(头)						
2(头)		地区	排名	销售	订单数量	订单金额
3		=ds1.Group(地区,false,...)	=&D3	=ds1.Select(雇员ID,true,订单金额)	=ds1.订单数量	=ds1.订单金额
4		ds1.sum(订单金额),true)	--	其他	=sum(E3{C3>5})	=sum(F3{C3>5})
5				=B3+"地区合计"	=ds1.sum(订单数量)	=ds1.sum(订单金额)

按订单金额降序排列

大于5的行隐藏，表达式：if(value())>5,true,false)

销售不超过5人则隐藏行，表达式：if(count(D3{>})<6,true,false)

"其他"销售人员订单数量和订单金额汇总。

集算器+报表工具实现

	A	B	C	D	E	F
1(头)						
2(头)		地区	排名	销售	订单数量	订单金额
3		=ds1.Group(地区)	=&D3	=ds1.Select(姓名)	=ds1.订单数量	=ds1.订单金额
4				=B3+"地区合计"	=ds1.sum(订单数量)	=ds1.sum(订单金额)

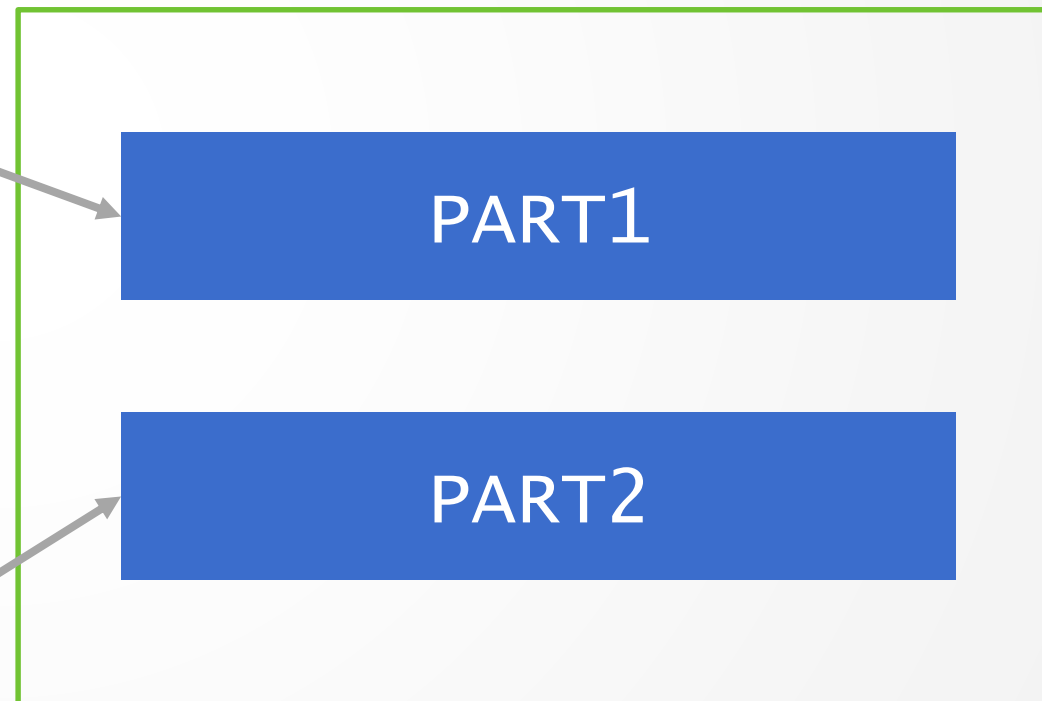


控制SQL执行路径

- 数据库的透明性为用户带来方便的同时，使得优化SQL执行路径非常困难
- 集算器就可以**自由控制执行路径**，部分运算可以移出数据库实施，方便完成性能优化

```
1 select t.*
2   from (select *
3         from (select syb.org_abbreviation as syb,
4               max(xmb.org_abbreviation) as
5               xmb,
6               sub.org_subjection_id as sub_id,
7               .....
298          group by l.requisition_id,
299                   l.note,
300                   l.requisition_type,
301                   .....
364          left join cpl_rwdmx_view ve
365                on ve.requisition_id = a.req_id) t
```

SQL - 442秒



集算器+SQL - 41秒



并行取数

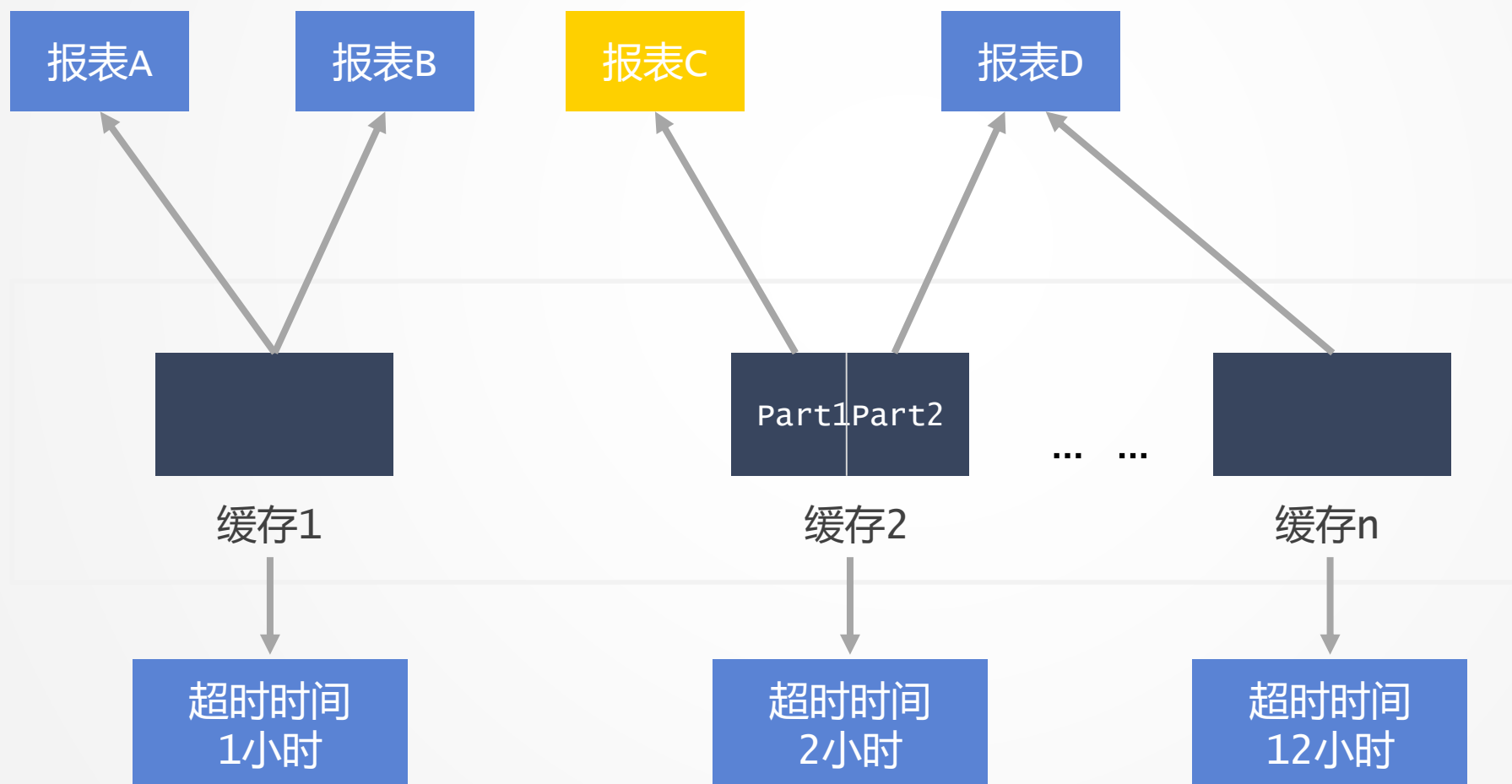
数据库JDBC性能较差，报表性能又严重依赖于取数环节；集算器可以采用**多线程并行**的方式同时建立多个数据库连接从数据库分段取数，可以**获得数倍性能提升**

	A	B	C
1	fork 4	=connect(db)	/分4线程，要分别建立连接
2		=B1.query@x("select * from T where part=?",A1)	/分别取每一段
3	=A1.conj()		/合并结果



可控缓存

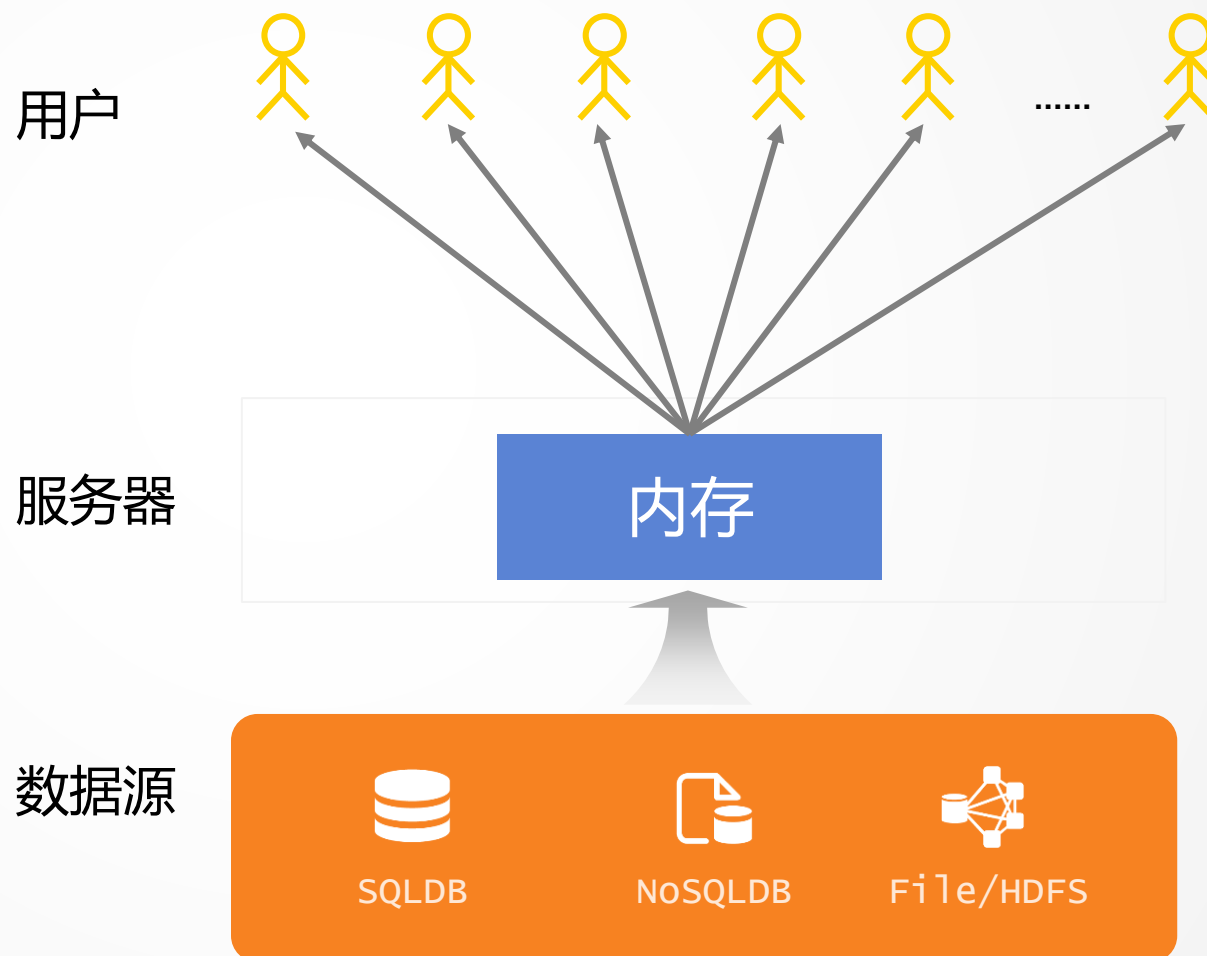
集算器可以实现报表的**部分缓存**、多个报表之间**缓存复用**、以及不同缓存的**不同生存周期**





内存共享

对于**高并发**报表，可以利用内存共享机制，性能更高，而且更方便并行计算



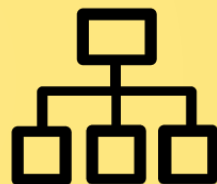
核心优势



降低开发难度



提高运算性能



优化应用结构



大数据报表



解释执行降低应用耦合度

使用JAVA和集算器准备报表数据源会有以下不同：

JAVA

模块化困难

Java程序必须和主应用一起编译打包，耦合度高

难以热切换

使用Java编写的报表数据准备算法有修改后会导致整个应用重新编译部署，很难做到热切换。

集算器

模块化简单

集算器脚本文件可以和报表模板一起管理维护，从而使报表功能模块化

容易热切换

集算器是解释执行的语言，很容易做到热切换



算法外置减少存储过程

采用存储过程实现数据准备算法，会造成报表与数据库的耦合问题

存储过程和报表的存放位置不同，导致对应难度很大

存储过程修改需要分配相应的数据库权限，存在安全隐患

存储过程容易被其他应用使用，造成多个应用间的耦合

使用集算器替代存储过程完成报表数据准备，会极大减少存储过程，算法外置后与报表模板一起存放管理，完全归属于应用本身，降低报表与应用其他部分或其他应用的耦合



数据外置减少中间表

由于数据量或计算复杂度原因，经常需要在数据库中创建中间表，中间表会带来如下问题：



数据库管理混乱

各个应用的累积大量中间表存储在线性结构的数据库中造成数据库管理混乱



数据库资源浪费

不用的中间表，仍然需要相应ETL过程向其更新数据，浪费数据库资源

将中间表数据外置存放到文件系统，**便于管理**，而且通过集算器获得计算能力，可以获得更高效的IO性能和计算能力，**充分减少数据库中间表**，梳理数据库结构



直接使用多数据源及跨库计算

集算器直接使用各类型数据源可以带来以下好处：

1、无须入库，体系结构更精简，减少中间表

2、直接取数，实时性好，减少不一致风险

3、充分利用各数据源固有的优势

核心优势



降低开发难度



提高运算性能



优化应用结构



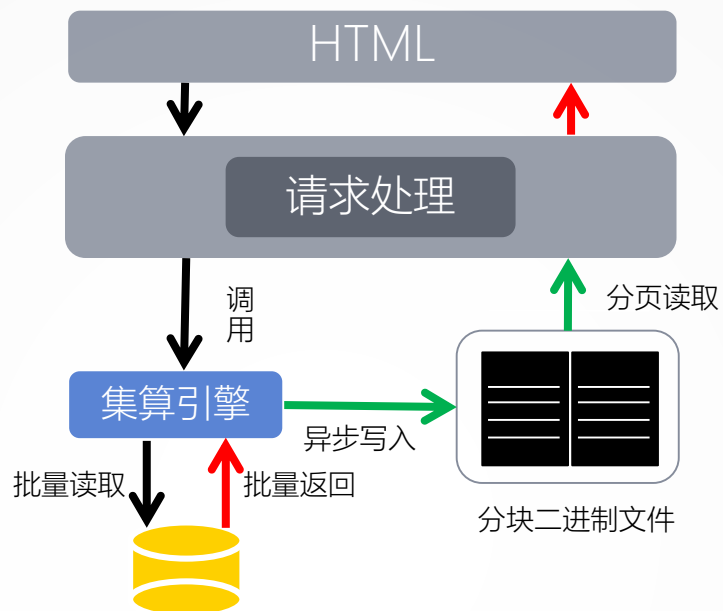
大数据报表



大清单报表

传统报表模式

- 全部取出后再呈现，用户体验恶劣
- 全内存计算的报表可能溢出
- 由数据库实现分页取出，性能很差

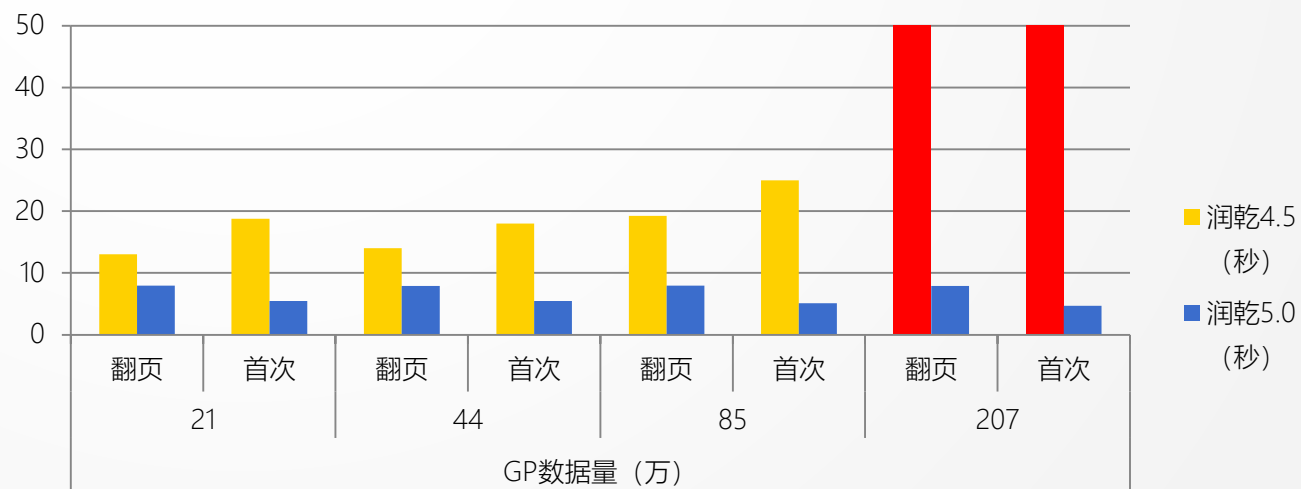


基于大报表的在线分析

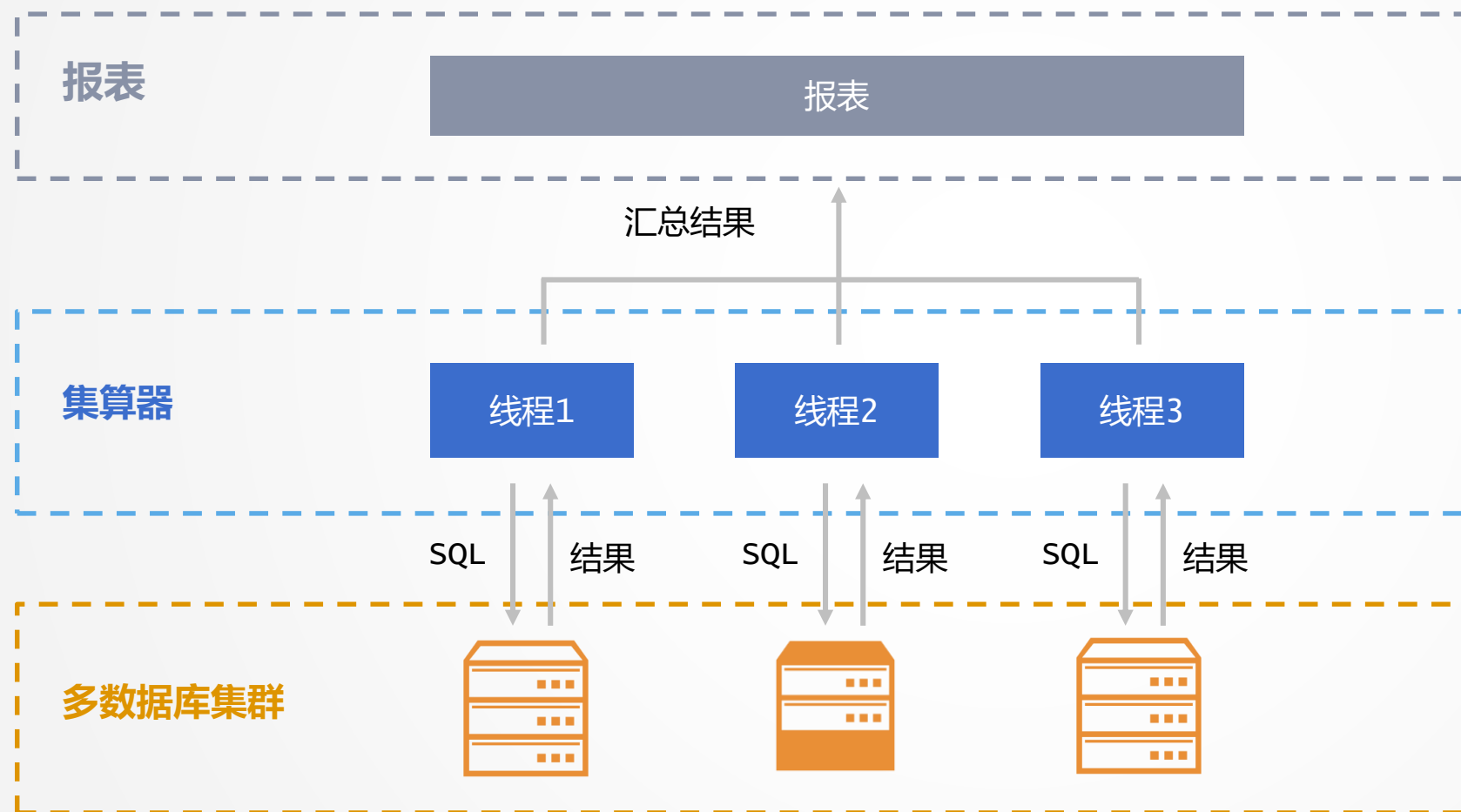


数据计算层实现分页取出

- 异步线程**边取数边呈现，快速响应
- 本地缓存**，按页随机取数
- 不依赖于**数据库的分布取出能力



分库计算汇总



集算器可以基于同构或异构数据库集群进行结果汇总，为报表输出汇总后的结果集



混合运算实现T+0报表

目前T+0报表常用方式与存在问题：

①

历史和当期数据同库存储

大量的历史数据会导致高昂的数据库成本
(存储成本和性能成本)

②

历史和当期数据分库存储

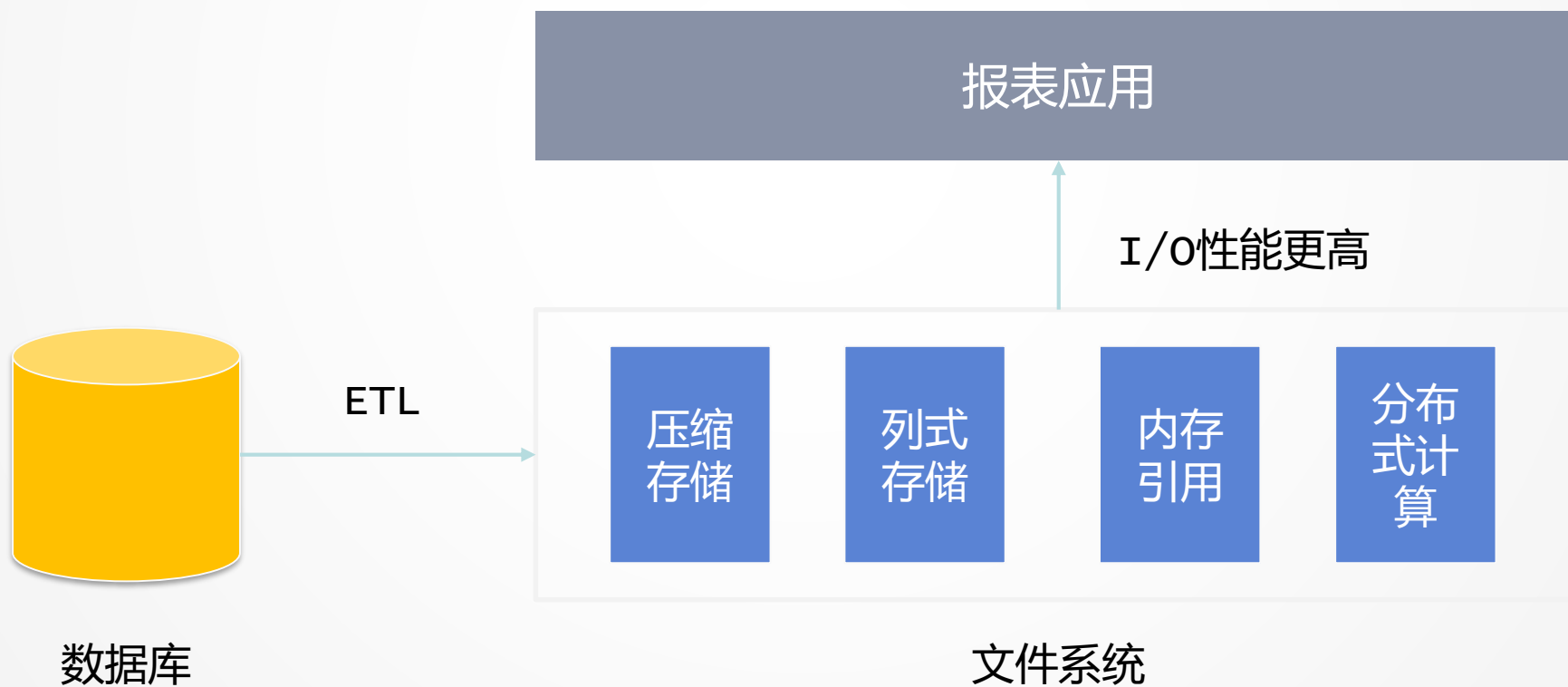
需要数据库具有跨库运算能力，但实施复杂度较高，性能较低；当数据库类型不同时难以实现

- 集算器可以基于多个**异构数据库**完成报表T+0查询；
- 还可以将历史数据存放到IO性能更佳的**文件系统**中采用集群运算获得更高性能和更低成本



数据外置与并行计算

文件系统IO性能高于数据库，还可以使用压缩数据存储、列式存储、内存记录引用，以及分布式集群等获得更高性能，**减轻数据库负担**





Hadoop数据源

Hadoop计算能力尚不完善

开发复杂

经常把计算后的数据再导入关系数据库

使用集算器实现Hadoop上报表计算

开发更轻松

支持集群并行计算，比Hadoop原生方法性能更好

无需导入数据库



集算器实现数据计算层-总结

应对报表的业务不稳定性，集算器的手段和达到的目标



高效率

更丰富的语法和类库



低耦合

报表模板与算法一体化



工具化

报表开发彻底独立化与应用解耦



热切换

解释执行无须重启应用



易开发

无须环境配置与应用层代码引用



低成本

非专业程序员自主完成

目录

Contents

1

公司介绍

2

现状分析

3

解决方案

4

核心优势

5

集算器技术特征

6

应用案例



集算器-开发环境

即装即用，调试功能完善

执行、调试执行、单步执行

设置断点

The screenshot displays the Jisuanqi 3.1 development environment. The main window shows a script editor with the following code:

```
A2 = 1 =file("../demo\\zh\\txt\\Sale.txt").import@t().select(month(Datetime)==6)
```

Below the script editor, a data grid is visible with the following columns: Datetime, Commodity, and Volume. The grid contains data for the period from 2009-06-01 08:0 to 2009-06-01 09:0.

Datetime	Commodity	Volume
2009-06-01 08:0	20077	28
2009-06-01 08:1	20056	47
2009-06-01 08:1	20094	34
2009-06-01 08:2	20020	19
2009-06-01 08:4	20013	42
2009-06-01 08:4	20077	1
2009-06-01 08:5	20069	19
2009-06-01 09:0	20011	22
2009-06-01 09:0	20007	22

At the bottom of the window, a system information output window is visible, showing the following text:

```
[2017-09-28 10:38:11]  
DEBUG: Esproc Function Points = 1000 0001 1111 1101
```

网格结果所见即所得，易于调试；方便引用中间结果

语法简单，符合自然思维，比其他高级开发语言更简单

系统信息输出，异常随时查看



集算器-面向过程计算

完整的循环分支控制

	A	B	C	D	E	F
1	=esProc.query("SELECT 订单ID AS 合同,订购日期 AS 日期,客户 ,订单金额 AS 金额,员工ID AS 销售 FROM 销售记录表 WHERE year(订购日期)=? OR year(订购日					
2	=esProc.query("select * from 员工表")					
3	>A1.run(销售=A2.select@1(编号:A1.销售))		序段值是记录			
4	=A1.group(销售)					
5	=create(销售,今年销售额,去年销售额,增长率,客户数,大客户数,大客户占比)					
6	for A4	=A6(1).销售.姓名				
7		=A6.select(year(日期)==年份).sum(金额)		/今年销售额		
8		=A6.select(year(日期)==年份-1).sum(金额)		/去年销售额		
9		=B8/B7-1	/增长率			
10		=A6.group(客户).(~.sum(金额))				
11		=B10.count()	客户数			
12		=B10.count(~.大客户数)	大客户数			
13		=B10.count(~.大客户占比)	大客户占比			
14		>A5.insert(0,B6,B7,B8,B9,B11,B12,B13)				
15	result A5					

天然分步、层次清晰、直接引用单元格名无需定义变量



集算器-敏捷语法体系



某支股票最长连续涨了多少交易日

```
1 select max(连续日数)
2 from (select count(*) 连续日数
3       from (select sum(涨跌标志) over(order by 交易日) 不涨日数
4             from (select 交易日,
5                   case when 收盘价>lag(收盘价) over(order by 交易日)
6                       then 0 else 1 end 涨跌标志
7                   from 股价表) )
8       group by 不涨日数)
```

SQL

	A
1	=股价表.sort(交易日)
2	=0
3	=A1.max(A2=if(收盘价>收盘价[-1],A2+1,0))

集算脚本



思考：按照自然思维怎么做？

语法体系更容易描述人的思路
数据模型不限制高效算法实现



集算器-丰富的运算类库

专门针对结构化数据表设计

	A	B	C
1	=esProc.query("SELECT 订单ID AS 合同,订购日期 AS	/读取销售记录表	
2	=A1.group(销售)		
3	=create(销售,今年销售额,去年销售额,客户数,大客户数)		
4	for A2	=A4(1).销售	
5		=A4.select(year(日期)==年份).sum(金额)	
6		=A4.select(year(日期)-=年份-1).sum(金额)	
7		=A4.group(年份).sum(金额)	
8		=B7.count()	
分组、循环			
	A	B	C
1	=esProc.query("select * from 员工表")		
2	=A1.select(性别=="男")		
3	=A1.select(出生日期>=date("1970-01-01"))		
4	=A2^A3	/交运算,统计晚于1970年出生的男员工	
5	=A2&A3	/并运算,统计男员工或者晚于1970年出生的员工	
6	=A2\A3	/运算,统计早于1970年出生的男员工	
7	=A4.sum(工资)		
8	=A5.avg(年龄)		
9	=A6.sort(出生日期)		
10	/集合作为基本数据类型		
11			
集合运算			

	A	B	C
1	=file("交易记录.txt").import@t0		
2	=A1.sort(客户编码,交易日期)		
3	=A2.select(车辆型号=="捷达" 车辆型号=="迈腾").dup@t0		
4	=A3.derive(interval(交易日期[-1],交易日期):间隔)		
5	=A4.select(车辆型号[-1]=="捷达" && 车辆型号=="迈腾" && 客户编码==客户编码[-1])		
6	=A5.avg(间隔)		
排序、过滤			
	A	B	C
1	=esProc.query("select * from 员工表")		
2	=A1.sort(入职日期)		
3	=A2.pmin(出生日期)	/出生最早的员工的记录序号	
4	=A2(to(A3-1))	/直接用序号访问成员	
5	=esProc.query("select * from 股价表 where 股票代码='000062")		
6	=A5.sort(交易日期)		
7	=A6.pmax(收盘价)	/收盘价最高的那条记录的序号	
8	=A6.calc(A7.收盘价/收盘价[-1]-1)		
9			
10	/直接用序号访问成员		
11			
有序集合			



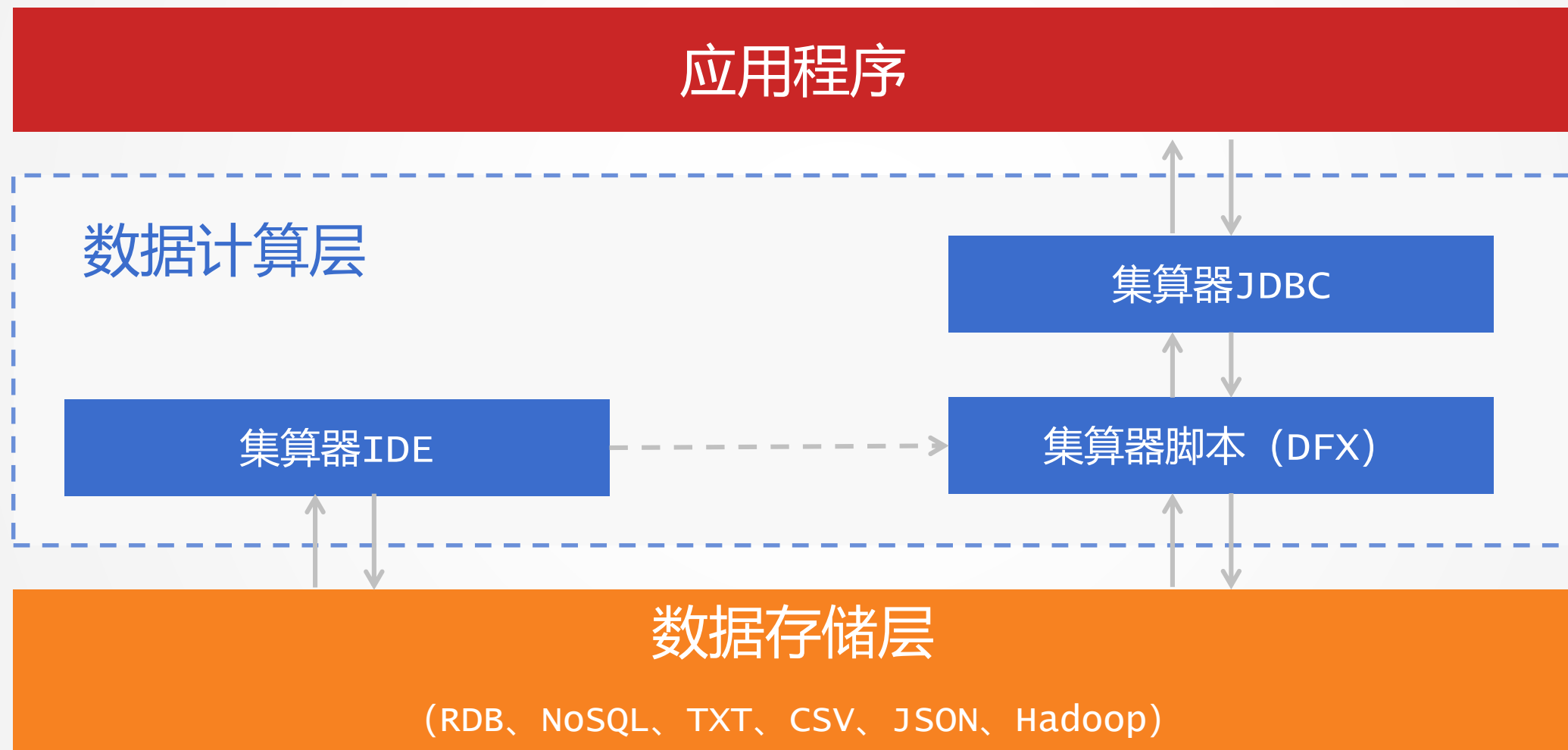
集算器-多样性数据源接口

- ▶ 高效二进制压缩文件、列式存储
- ▶ RDB: Oracle,DB2,MS SQL,MySQL,PG,....
- ▶ TXT/CSV, JSON/XML, EXCEL
- ▶ Hadoop: HDFS, HIVE, HBASE
- ▶ MongoDB, REDIS, ...
- ▶ HTTP、ALI-OTS
- ▶
- ▶ 内置接口, 即装即用

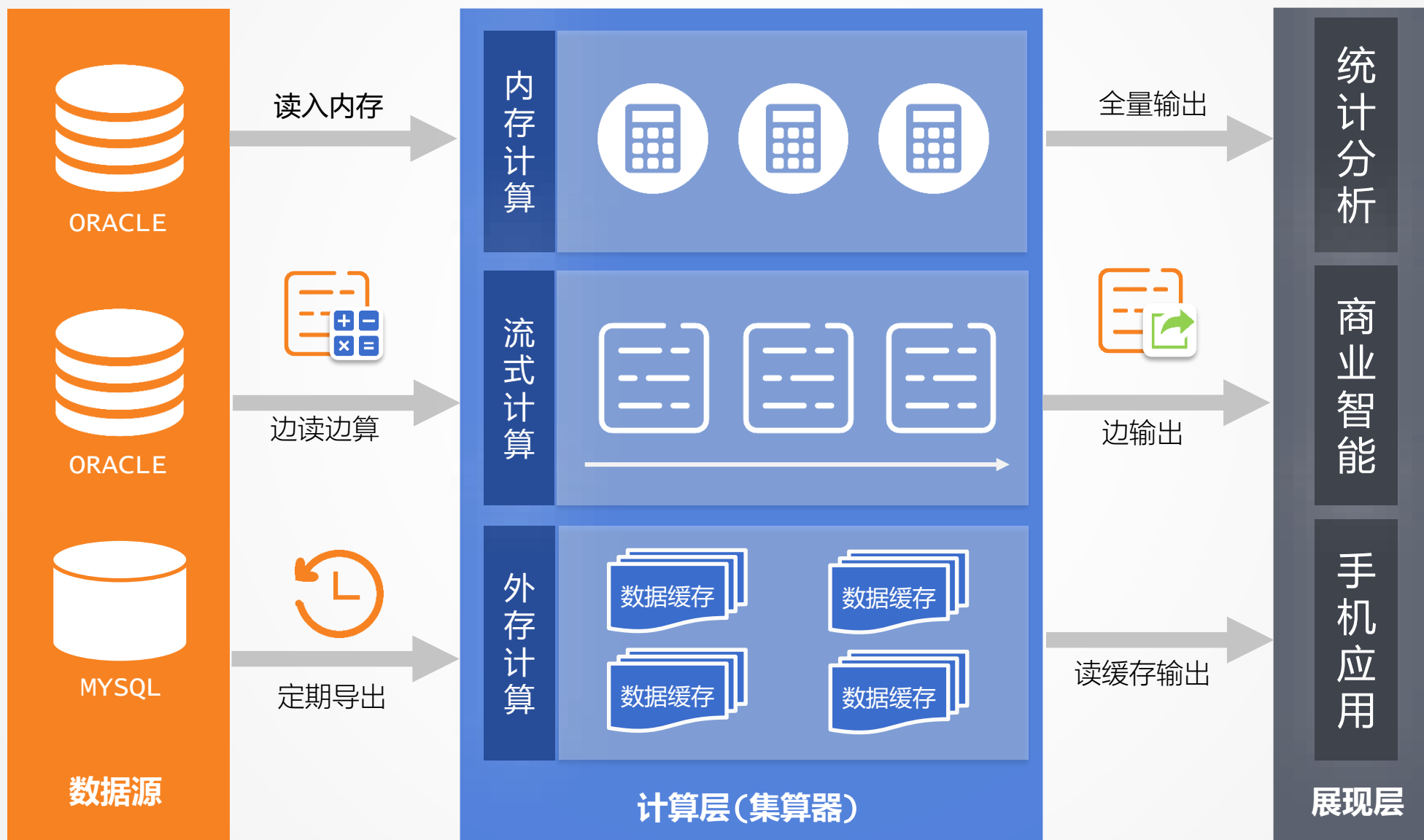


集算器-集成与管理

应用无缝集成，代码易于管理



集算器-数据流向模式



目录

Contents

1

公司介绍

2

现状分析

3

解决方案

4

核心优势

5

集算器技术特征

6

应用案例

案例1

Application case

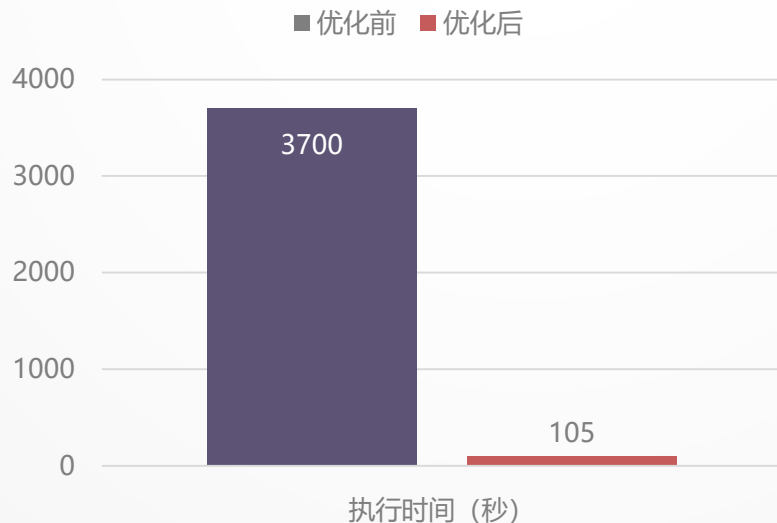
某金融机构-报表加速器

【案例背景】该金融机构需要维护的报表超过7500张，其中20%查询速度为分钟级，5%的查询速度为小时级，这5%非常慢的查询SQL代码行数有几百行，存储过程更是达到几千行，关联运算多。用通俗的语言描述报表慢的原因主要是数据量大，业务逻辑复杂。

【选取场景】某交易情况统计，该表运行时间为3700秒，4个SQL数据集，涉及6个数据库表的关联运算，其中3个表的数据量是千万级（1708万-4886万）。

【优化方案】通过润乾集算器的高效关联算法和多线程并行计算予以优化。

【优化效果】不改变硬件环境的条件下，报表查询速度从3700秒降到105秒。



提升查询效率

35.2 倍

案例2

Application case

某石油-多样性数据源

【案例背景】报表需要从油井传感器收集数据并实时展现，由于源数据是多层Json，报表工具无法解析，只能编写大量JAVA类解析计算。由于JAVA缺乏结构化计算函数，因此开发工作量巨大。

【优化方案】用集算器做计算层，直接访问Http Json流，并对多层数据直接计算。

【优化效果】原来200多张报表，90%的时间和人力都花在写JAVA类。现在可以把主要精力用于设计报表，计算json的工作量忽略不计。

案例3

Application case

某铁塔-并行取数

【案例背景】每天晚上需要对数据进行大批量的汇总统计，数据写回数据库中，给下一个部门第二天查询使用。数据库负担重，因此采取库外计算，使用java程序从数据库中抽取数据，计算后把结果写回数据库中。

【现状分析】Jdbc读数异常慢。33张事实表要做group聚集，然后再和维表关联，group和关联算法都挺慢。自己实现hash算法关联，工作量巨大。源数据量3千万条左右，用时37分钟。

【优化效果】集算器并行读数，极大提升jdbc读数效率。循环代码实现33张表分组和关联算法，hash算法在函数内部已封装，代码异常简短。并行计算提升计算效率。用时11分钟。

案例4

Application case

某银行-大数据报表

【案例背景】随着客户业务数据量级的日益增长，使原有业务报表在查询和导出时的效率明显降低，已经跟不上业务的发展。并且在引入非关系型数据库（HBase）后，原有报表工具无法连接。

【优化方案】定时调度集算器脚本，脚本中使用invoke调用外部接口从HBase取数存为**高性能二进制文件**，报表对二进制进行查询。查询时采用**大数据集**技术进行异步加载。

【优化效果】原有回单报表查询一个分行一个月的数据需要**十几分钟**，甚至造成内存溢出，升级为集算报表后，每页展示2000条数据，首页加载时间为10秒，一个分行一个月的数据全部加载完成缩短至**5分钟**，并且不会造成内存溢出的问题。

案例5

Application case

某铁路局-动态数据源/集

【案例背景】根据查询条件不同，报表需要生成动态SQL，并进行跨库计算，比如主数据库+京九路段探伤数据库，或主数据库+沪昆路段+武九路段。由于现有报表工具无法实现动态跨库计算，只能用JAVA硬写，开发工作量巨大，代码性能差。

【改善方式】使用集算器的流程控制语句、动态语法生成动态SQL，利用高性能结构化计算函数进行跨库计算。

【应用效果】原来6个月没开发完第一张报表，现在40天完成十八张报表，实现了从无到有的飞跃，并顺利完成验收

案例6

Application case

某银行-集算器辅助报表实现动态水印

集算器与报表完成配合实现

- 1、**集算器动态绘制水印图片**：根据报表传递的用户编号参数，动态生成该用户的水印图片
- 2、**报表引用集算器数据集**：报表中设置集算器数据集，引用二进制图片字段
- 3、**报表引用动态水印图片**：报表背景图中直接表达式引用集算器生成的水印图片数据集的图片字段，实现报表动态水印的效果

集算器的优势

- 1、快速开发、易维护
- 2、动态灵活
- 3、可视化水印图片调试
- 4、扩展和复用性高

- 1、**动态绘制水印图片**：普通报表工具只能实现静态图片水印的设置，实现动态需要自定义类，调试、维护均不方便
- 2、**报表插件式引用集算器脚本**：集算器的脚本支持可视化调试、编辑，报表可以直接引用集算器类型数据集，插件式接入，独立引用、易于扩展

应用优势与效果

序号	分行/管理岗	管理岗	所属行/所属行名称	所属行/所属行名称	姓名	性别	证件类型	证件号码	客户归属
1	总行	00001	00001	和平支行	董国生	男	身份证	56881002180000	金卡
2	总行	00001	00001	和平支行	王岩梅	女	身份证	56881002180000	白金卡
3	总行	00001	00001	和平支行	陈三	男	身份证	56881002180000	白金卡
4	总行	00001	00001	和平支行	陈三	男	身份证	56881002180000	白金卡
5	总行	00001	00001	和平支行	陈三	男	身份证	56881002180000	白金卡
6	总行	00001	00001	和平支行	陈三	男	身份证	56881002180000	金卡
7	总行	00001	00001	和平支行	陈三	男	身份证	56881002180000	财富客户
8	总行	00001	00001	和平支行	陈三	男	身份证	56881002180000	白金卡
9	总行	00001	00001	和平支行	陈三	男	身份证	56881002180000	金卡
10	总行	00001	00001	和平支行	陈三	男	身份证	56881002180000	金卡
11	总行	00001	00001	和平支行	陈三	男	身份证	56881002180000	金卡
12	总行	00001	00001	和平支行	陈三	男	身份证	56881002180000	金卡
13	总行	00001	00001	和平支行	陈三	男	身份证	56881002180000	白金卡
14	总行	00001	00001	和平支行	陈三	男	身份证	56881002180000	白金卡
15	总行	00001	00001	和平支行	陈三	男	身份证	56881002180000	白金卡
16	总行	00001	00001	和平支行	陈三	男	身份证	56881002180000	金卡
17	总行	00001	00001	和平支行	陈三	男	身份证	56881002180000	白金卡
18	总行	00001	00001	和平支行	陈三	男	身份证	56881002180000	白金卡
19	总行	00001	00001	和平支行	陈三	男	身份证	56881002180000	金卡
20	总行	00001	00001	和平支行	陈三	男	身份证	56881002180000	白金卡
21	总行	00001	00001	和平支行	陈三	男	身份证	56881002180000	财富客户
22	总行	00001	00001	和平支行	陈三	男	身份证	56881002180000	白金卡
23	总行	00001	00001	和平支行	陈三	男	身份证	56881002180000	白金卡
24	总行	00001	00001	和平支行	陈三	男	身份证	56881002180000	白金卡
25	总行	00001	00001	和平支行	陈三	男	身份证	56881002180000	白金卡
26	总行	00001	00001	和平支行	陈三	男	身份证	56881002180000	财富客户
27	总行	00001	00001	和平支行	陈三	男	身份证	56881002180000	金卡
28	总行	00001	00001	和平支行	陈三	男	身份证	56881002180000	白金卡
29	总行	00001	00001	和平支行	陈三	男	身份证	56881002180000	白金卡
30	总行	00001	00001	和平支行	陈三	男	身份证	56881002180000	白金卡

案例7

Application case

某石化-生成多层动态表

业务需求

生成多维度多层动态叉乘中间表

存在问题

- 难以简化：**本算法用SQL/存储无法直观表达层级关系，使用复杂的嵌套语句或难懂的多级关联来实现。
- 工作量大：**SQL表达能力弱，无法用精简的语法实现算法。
- 性能差：**在SQL中，需要大量临时表储存嵌套的临时数据，耗费了大量的时间。
- 难以维护：**SQL实现该算法，代码长且复杂，解读困难。
- 难以优化：**使用SQL/存储实现该算法，代码量巨大，无法耗费更多时间来优化。

解决方案

使用集算器中的高性能函数，对象访问，内存计算等功能，实现本算法，有效的提升了执行性能。

- 快速开发**
- 代码精简 (11行) 易维护**
- 可优化 (基于中间层再计算)**
- 高性能**

指标对比

指标	存储过程	集算器	提升
工作量	120人天	1人天	120倍
执行性能	1800秒	40秒	45倍
可维护性	无	容易	质变
可优化性	无	容易	质变
代码量	1212行	11行	11倍

案例8

Application case

某著名数据服务商-多源跨库计算

【案例背景】

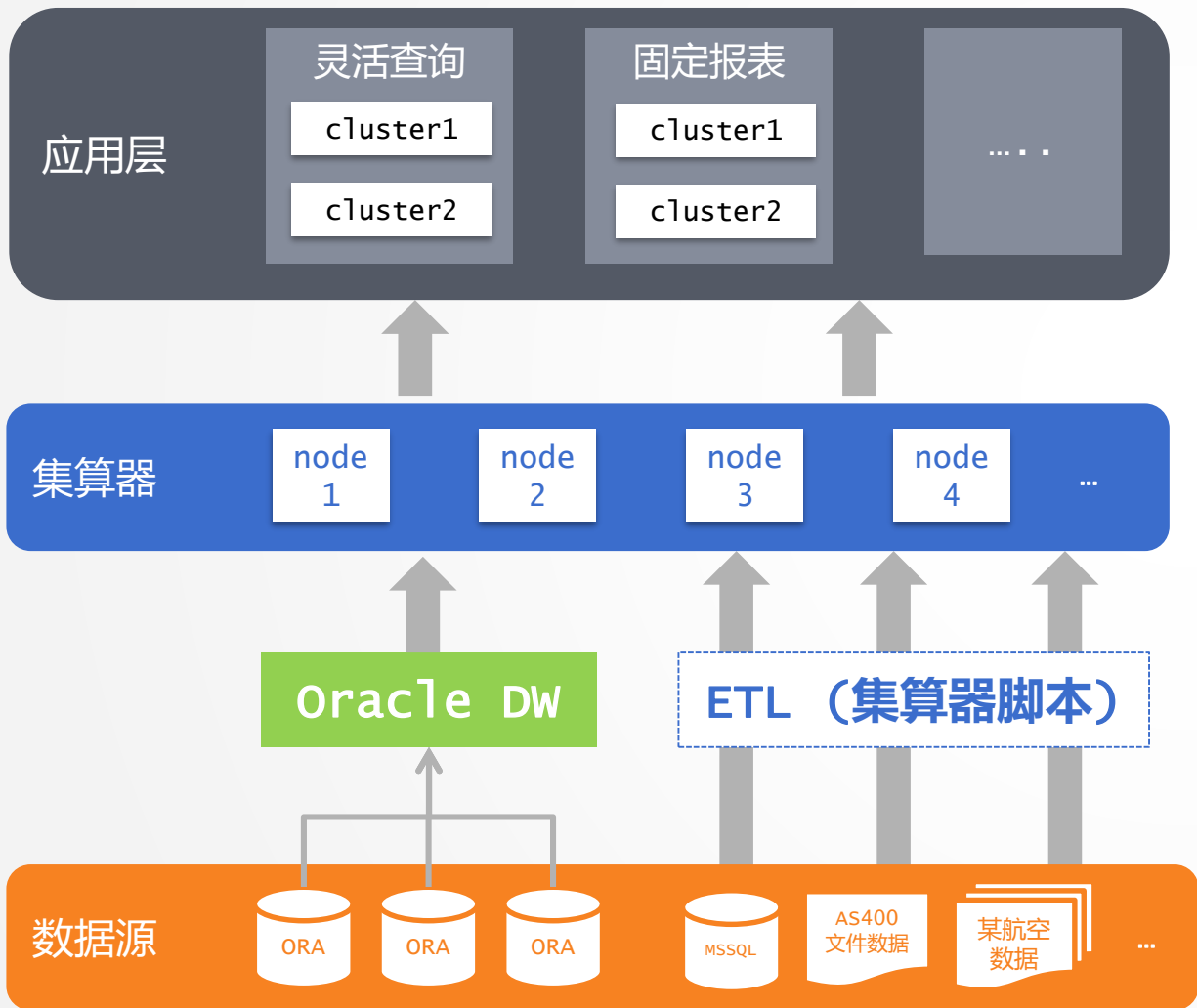
经过多年信息化建设目前存在多种类型的数据源，包括ORALCE、DB2、MSSQL以及文件系统数据等，现有应用（业务系统和分析系统）直接基于多种数据源开发。

- 1、数据源管理非常复杂
- 2、应用与数据源耦合度过高，数据源发生变化应用要随之变动较大
- 3、实现跨多种类型数据源查询困难
- 4、RDB计算性能不足，扩容成本过高

案例8

某著名数据服务商-多源跨库计算

Application case



使用集算器作为统一数据服务接口，通过标准JDBC接口为上层应用提供数据输出服务

面向分析系统，提供了固定报表查询和为业务人员提供的查询工具

集算器作为统一数据服务平台带来的效果：

- 1、应用直接基于集算器开发，屏蔽底层数据源之间的差异，规范了数据管理
- 2、实现多数据源混合计算
- 3、实现历史数据归档，并基于归档数据做分析
- 4、提供多线程并行计算机制，加速数据计算性能
- 5、提供分布式计算，支持横向扩容 (scale out) ，降低扩容成本

案例9

Application case

某大型电信集成商-Excel财务报表优化

项目背景

- 1、多样性数据源：报表数据来源繁杂，数据无法正常关联，报表需要逐级汇总。
- 2、可维护性差：可复用性和维护性极差，严重影响了客户报表输出体验。

面临困难

- 1、数据脏：数据质量不高
- 2、算法复杂：业务逻辑复杂度较高
- 3、数据量大：数据量较大，不适合Excel操作
- 4、效率低：需要逐级汇总，Excel执行效率低

案例9

某大型电信集成商-Excel财务报表优化

Application case

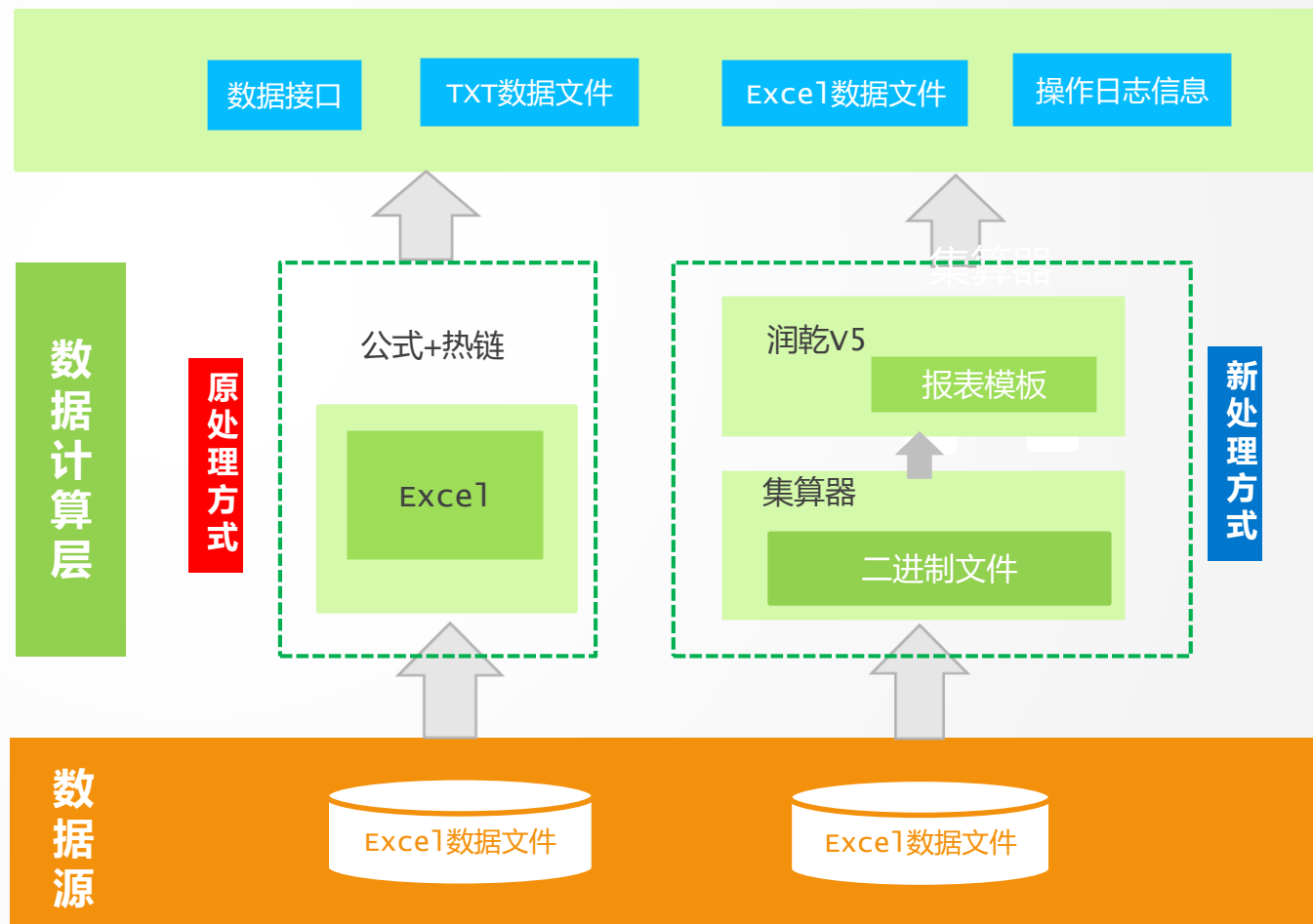
解决思路

- 1、集算器高性能技术：并行技术
- 2、计算层关联：丰富的内置(半)结构化计算函数
- 3、报表模板：采用SDK接口直接导出Excel

优化结果

- 1、文件处理：40秒 -> 3秒
- 2、导出时间：半小时/失败->12分36秒
- 3、速度提升：提升多倍以上
- 4、其他提升：复用性提升，便于数据维护

架构对比



案例10

Application case

某金农项目-报表优化

项目背景

- 1、**查询导出慢**：某金农项目，报表查询导出很慢，经常失败，严重影响了业务用户使用报表系统。
- 2、**优化无果**：尝试了一些优化手段，但效果不理想。

面临困难

- 1、**逻辑复杂**：业务逻辑复杂度较高，数据量较大
- 2、**报表关联计算**：报表表达式逻辑不合理，用报表单元格关联计算
- 3、**重复查询**：重复查询数据库
- 4、**内存计算**：全部加载到内存计算导出

案例10 某金农项目-报表优化

Application case

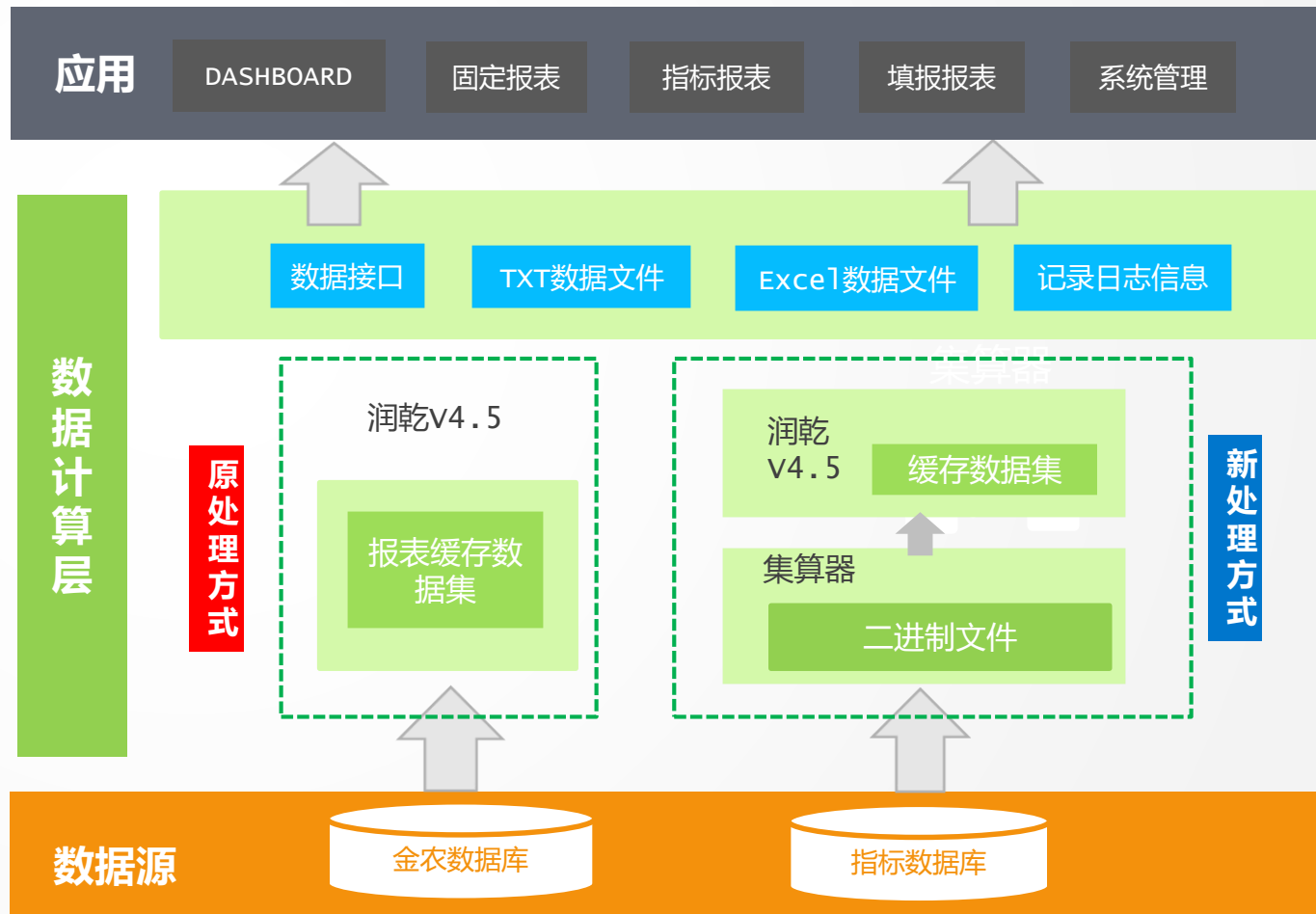
解决思路

- 1、集算器高性能技术：并行技术、游标技术
- 2、计算层关联：内置结构化计算函数
- 3、报表缓存：避免重复查询
- 4、集算器大数据集：异步计算导出

优化结果

- 1、查询速度：330秒->3秒/10秒
- 2、导出时间：半小时/失败->100秒
- 3、展现速度：提升10倍以上

架构对比



案例11

Application case

木槿生活-资源整合、敏捷开发

现状分析

- 1、资源整合难：多个业务系统互不关联，造成信息孤岛。
- 2、需求响应慢：企业的响应速度追不上数据的生产速度。
- 3、报表查询慢：ERP中报表查询越来越慢，影响业务交易。
- 4、报表数量多：ERP中已维护近百张报表数量，而且还在不断增加。
- 5、数据产生价值：完成资源整合，让数据产生价值，决策层很关注。

面临困难

- 1、数据库性能差：计算作业过多过重，数据库性能达到瓶颈。
- 2、优化成本高：提高性能需庞大的人员开发成本、硬件扩容成本。
- 3、算法难度大：数据结构为交易而设计，实现报表难度很大。
- 4、时间紧：平台+70余张报表，需1个月内初验。

案例11

木槿生活-资源整合、敏捷开发

Application case

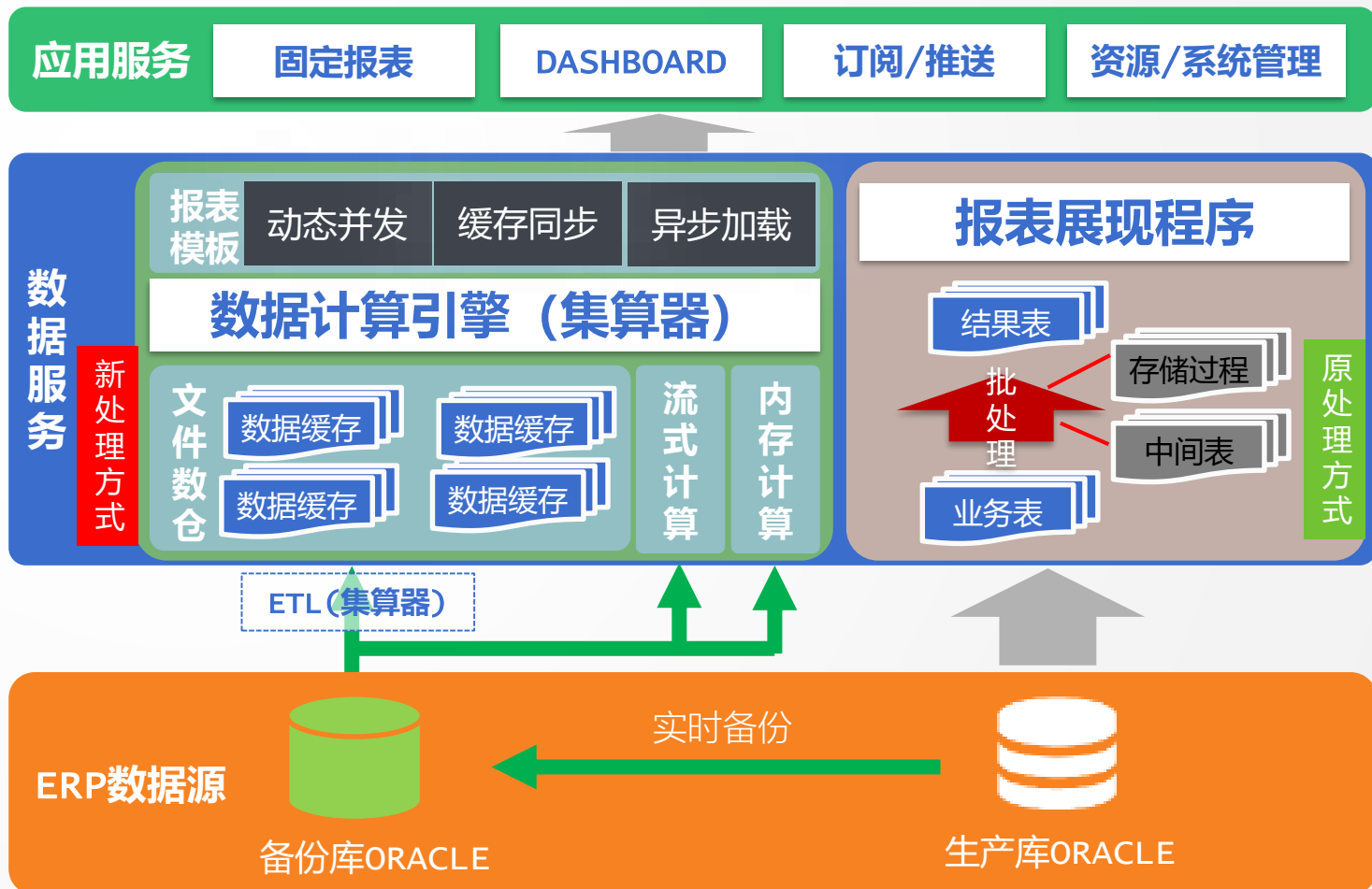
项目价值

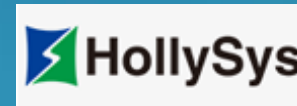
- 1、快速响应:及时满足业务需求,实现数据驱动运营
- 2、低成本方案:解决报表性能问题,给数据库减压
- 3、工具化:报表开发实现工具化,IT人员就可以解决
- 4、扩展性:二期项目扩建,实现资源共享,奠定基础

集算器优势

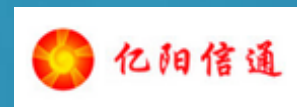
对比指标	SQL/存储过程	集算器	变化
代码行	253	26	代码量减至10%
开发周期	5人天	2人天	工作量减至40%
执行性能	2分45秒	38秒	平均性能提升5倍
应用耦合	高	低	易管理、热切换
维护性	较差	较好	维护成本低

解决方案





ISV DEVELOPER
合作伙伴 与 开发者



金融



政府



电力



CUSTOMERS

部分用户列表

超过 8000+ 用户

电信



能源



交通



外企



创新技术 推动应用进步!

