集算器

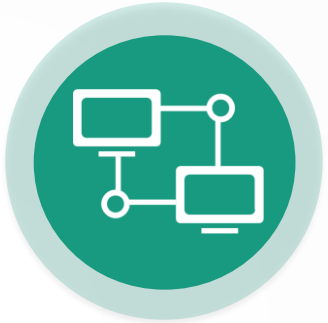
报表数据源

润乾软件出品

现状分析

报表业务不稳定

业务发展过程中，新报表不断、老报表要改，造成了报表开发没完没了



数据准备开发难

为报表准备数据的SQL长达数K，写起来难，维护更难

大报表查询慢

数据量大，报表呈现太慢，搞不好还溢出，用户抱怨

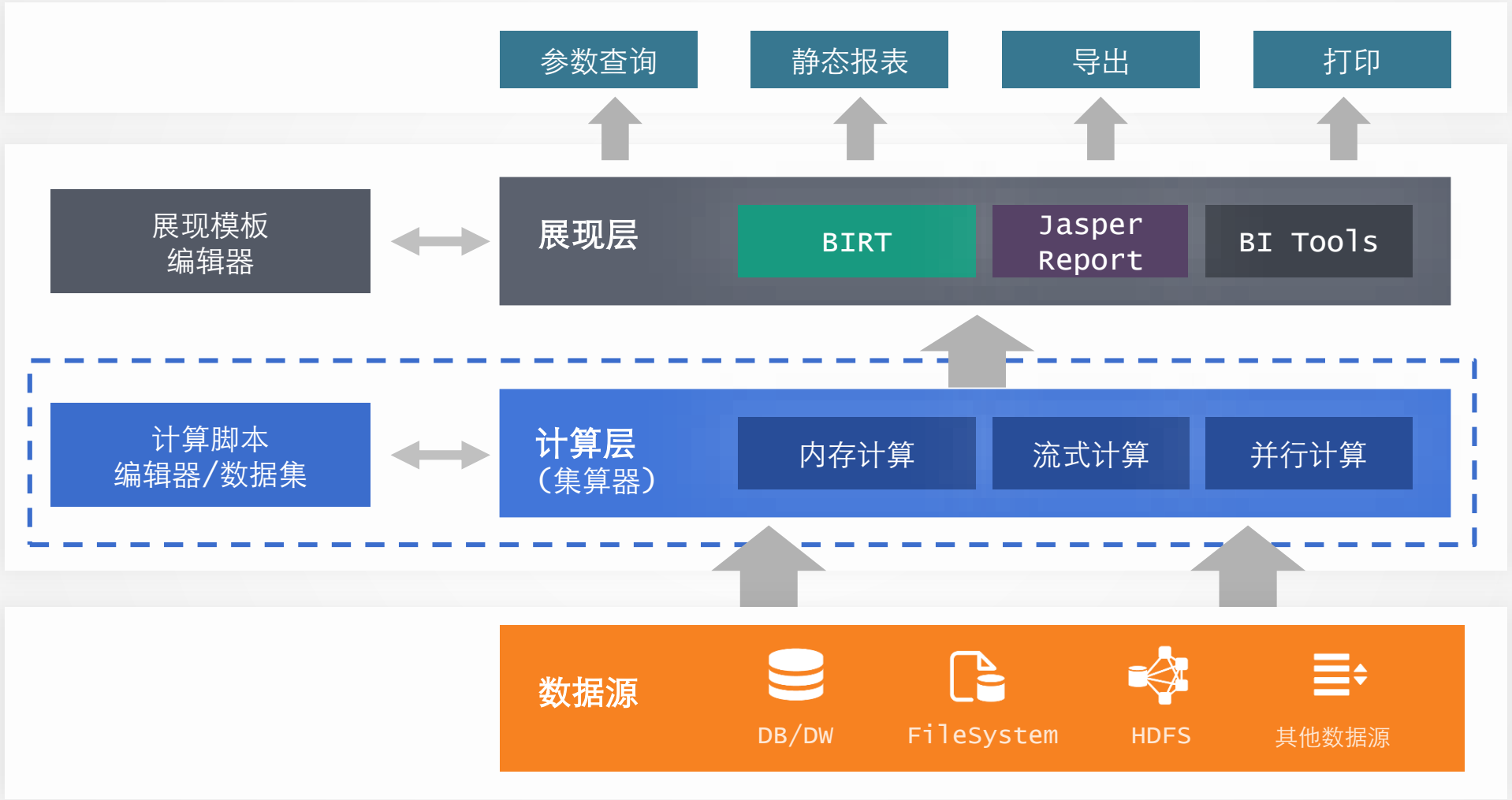
与应用耦合高

应用与报表高耦合，修改报表常常需要重启应用

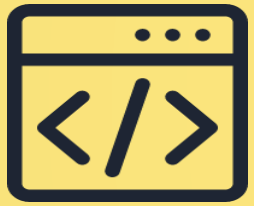
报表开发的难点转移

- ✓ 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

引入计算层 – 集算器


参数查询
静态报表
导出
打印
展现模板
编辑器
展现层
BIRT
Jasper
Report
BI Tools
计算脚本
编辑器/数据集
计算层
(集算器)
内存计算
流式计算
并行计算
数据源
DB/DW
FileSystem
HDFS
其他数据源

核心优势



降低开发难度

提高运算性能

优化应用结构

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

某支股票最长连续涨了多少交易日

```
1  select max(continuousDays)-1
2  from (select count(*) continuousDays
3        from (select sum(changeSign) over(order by tradeDate) unRiseDays
4              from (select tradeDate,
5                    case when closePrice>lag(closePrice) over(order by tradeDate)
6                    then 0 else 1 end changeSign
7                    from stock) )
8        group by unRiseDays)
```

SQL

	A
1	=stock.sort(tradeDate)
2	=0
3	=A1.max(A2=if(closePrice>closePrice[-1],A2+1,0))

SPL

语法体系更容易描述人的自然思维！

思考：按照自然思维怎么做?

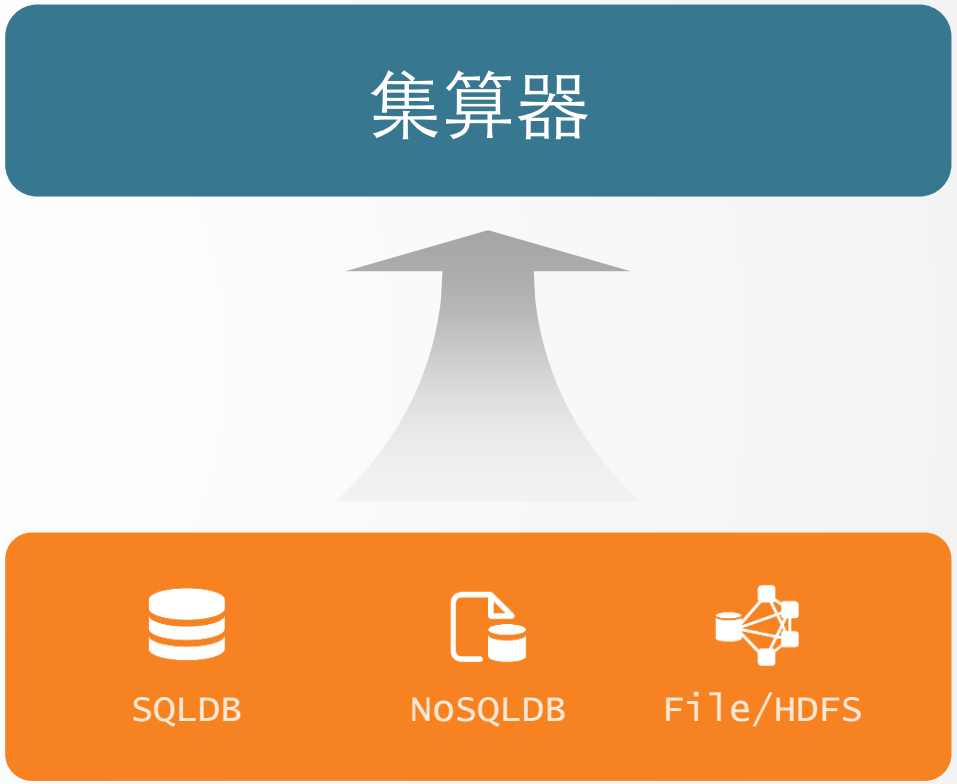
多样性数据源支持

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

结果集容量控制

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

核心优势

降低开发难度

提高运算性能

优化应用结构

并行取数

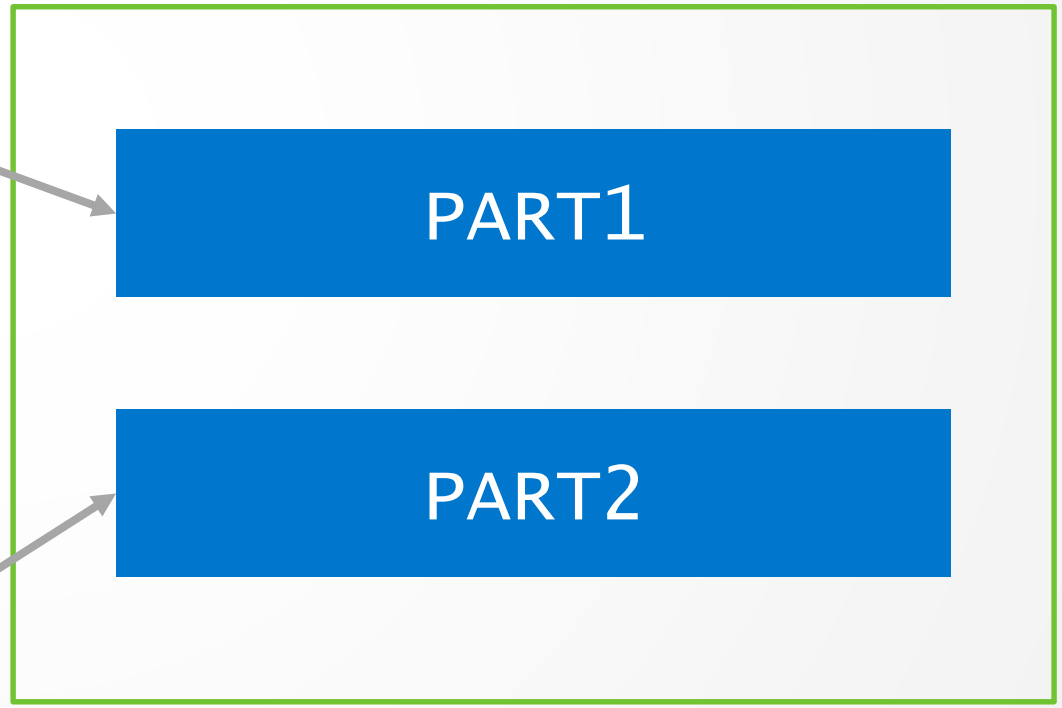
数据库JDBC性能较差，报表性能又严重依赖于取数环节；集算器可以采用多线程并行的方式同时建立多个数据库连接从数据库分段取数，可以获得数倍性能提升

	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

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,
                       sub.org_subjection_id as sub_id,
   ......
298  group by l.requisition_id,
299                    l.note,
300                    l.requisition_type,
   ......
364  left join cpl_rwdmx_view ve
365           on ve.requisition_id = a.req_id) t
```

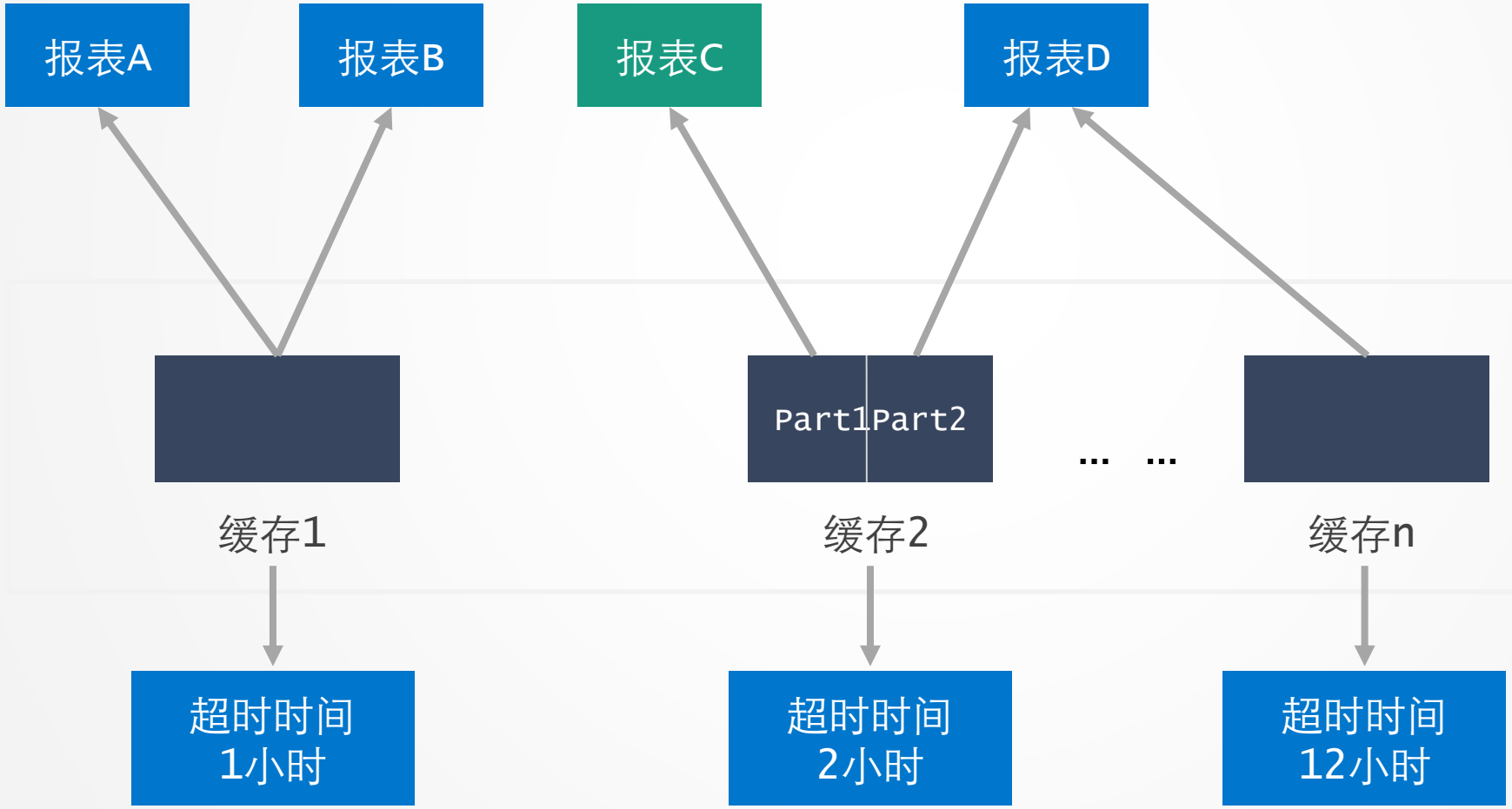


SQL – 442秒

集算器+SQL – 41秒

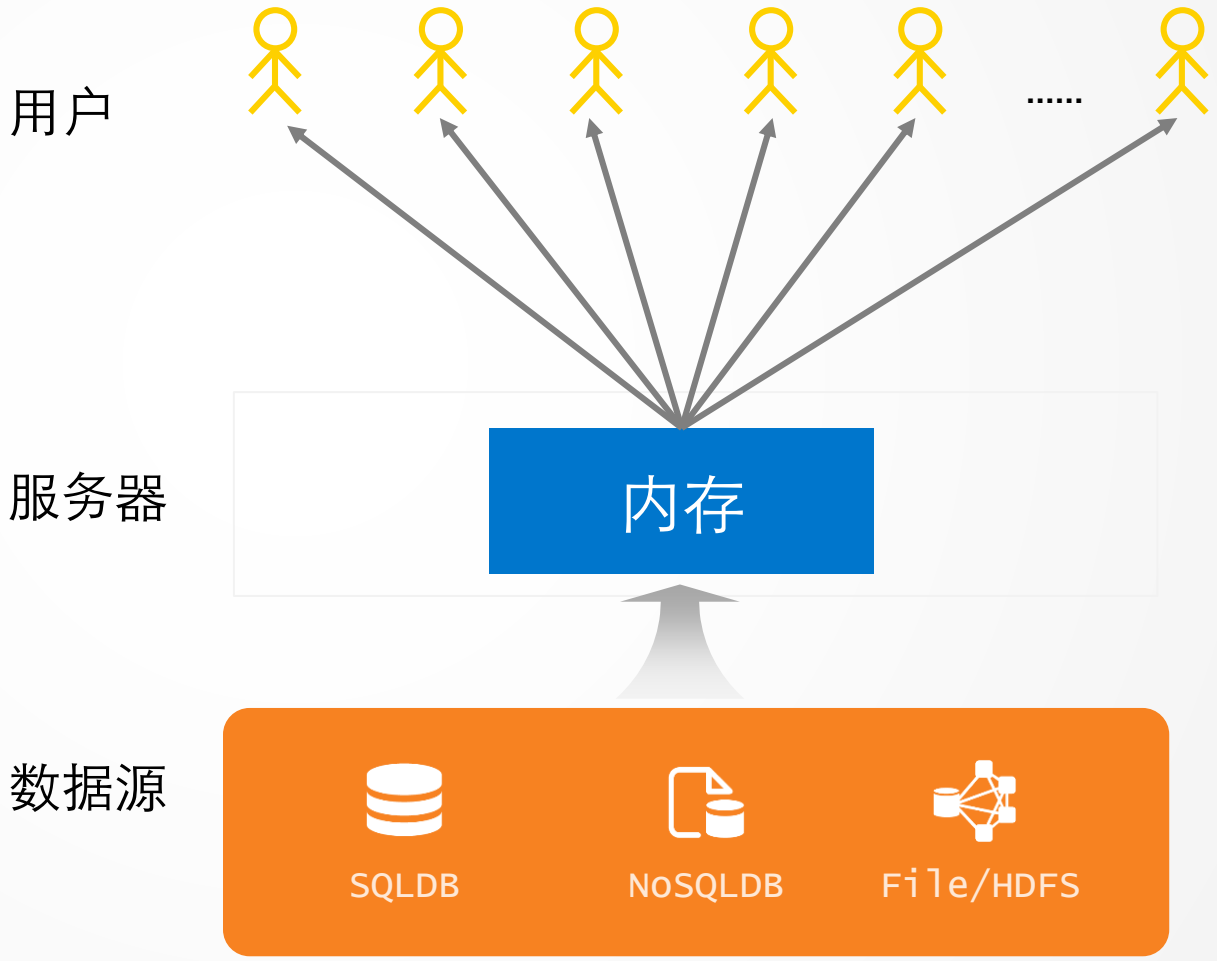
可控缓存

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

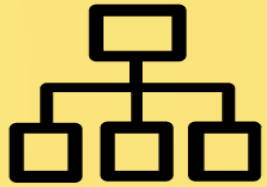
对于**高并发**报表，可以利用内存共享机制，性能更高，而且更方便并行计算



核心优势

降低开发难度

提高运算性能



优化应用结构

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

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

类库少

JAVA程序在结构化和半结构化计算方面的类库较少

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

类库多

更丰富的语法和类库，让结构化数据的计算更有效率

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

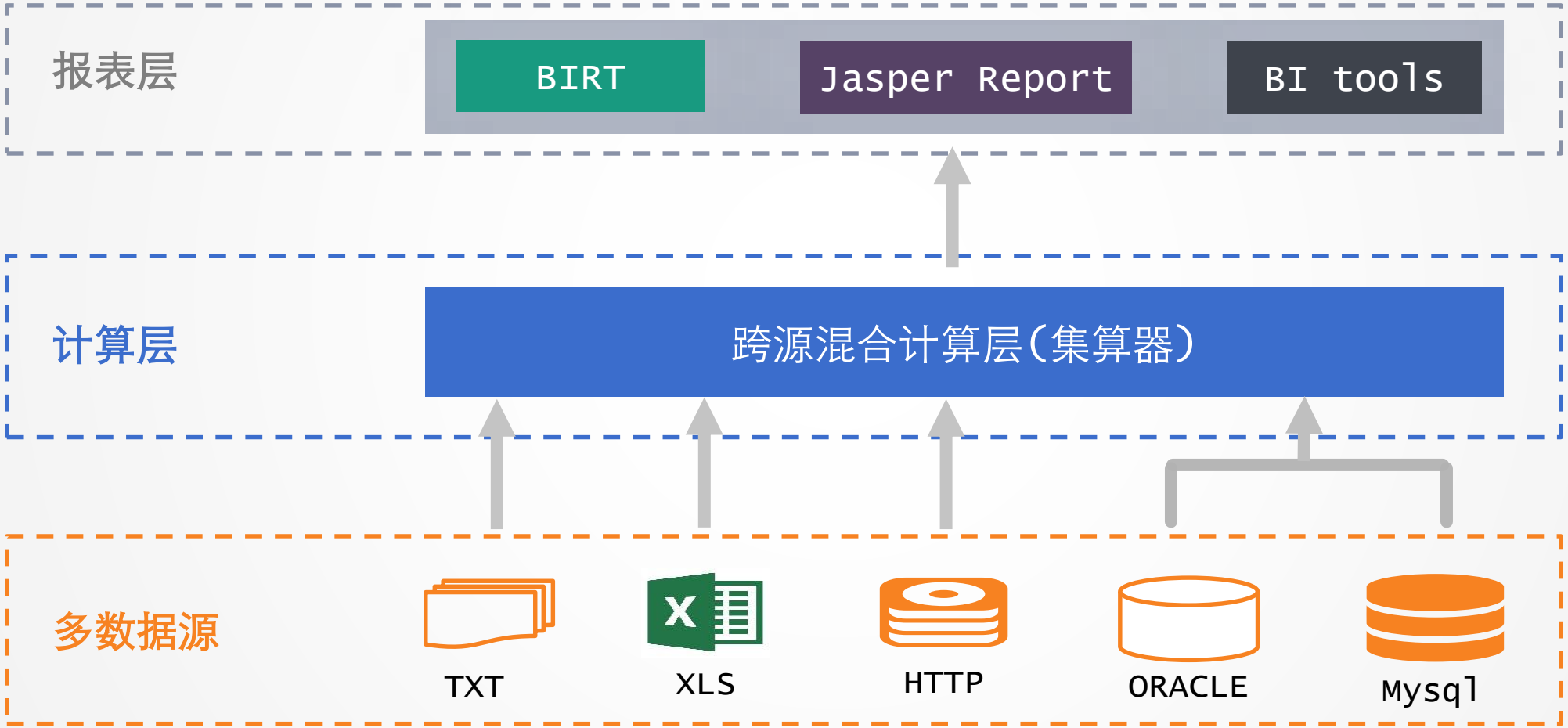
各个应用的累积大量中间表存储在线性结构的数据库中造成数据库管理混乱

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



集算器实现报表数据计算层 – 总结

易开发

敏捷语法体系，提高开发效率

开放性

丰富的多样数据源接口，直接计算

低耦合

报表开发彻底独立化与应用解耦

易集成

对外提供标准JAR包易于嵌入集成

热切换

解释执行无须重启应用

高性能

离散数据集模型支持更高计算性能

*延伸阅读：http://c.raqsoft.com.cn/article/1565142431668

集算器作为BIRT数据源-动态解析CSV

BIRT representation

Showing page 1 of 1

dataset_date	measurement	managedelement	content	last_minute_cp	last_5_minute_	last_30_minute	last_hour_cpu_
2015-05-26 21:45:00.0	cpqHoCpuUtilTaln	ltcom04	20:7:5:5	20	7	5	5
2015-05-26 21:45:00.0	cpqHoCpuUtilTaln	ltcom04	17:8:5:5	17	8	5	5
2015-05-26 21:45:00.0	cpqHoCpuUtilTaln	ltcom01	20:4:7:7	20	4	7	7
2015-05-26 21:45:00.0	cpqHoCpuUtilTaln	ltcom03	8:12:5:5	8	12	5	5
2015-05-26 21:45:00.0	cpqHoCpuUtilTaln	ltcom02	11:5:5:6	11	5	5	6

esProc calculation

	A
1	=file("D:\\DummyData.csv").import@t(dataset_date,measurement,managedelement,content:string;,",")
2	=file("D:\\Mapping.csv").import@t(;,",")
3	=A2(1).content.split(":")
4	=A3.to(2,).("cols("+string(#+1)+"):"+~).string()
5	=A1.derive((cols=string(content).split(":"))(1):${A3(1)},${A4})

Data sources

```
1 dataset_date,measurement,managedelement,content
2 2015-05-26 21:45:00.0,cpqHoCpuUtilTable,nltcom04,20:7:5:5
3 2015-05-26 21:45:00.0,cpqHoCpuUtilTable,nltcom04,17:8:5:5
4 2015-05-26 21:45:00.0,cpqHoCpuUtilTable,nltcom01,20:4:7:7
5 2015-05-26 21:45:00.0,cpqHoCpuUtilTable,nltcom03,8:12:5:5
6 2015-05-26 21:45:00.0,cpqHoCpuUtilTable,nltcom02,11:5:5:6
```

DummyData.csv

```
1 managedelement,content
2 nltcom,last_minute_cpu_utilization_kpi:last_5_minute_cpu_u
  tilization_kpi:last_30_minute_cpu_utilization_kpi:last_hou
  r_cpu_utilization_kpi
```

Mapping.csv

集算器作为JasperReport数据源-跨库关联

Jasper representation

Designer | XML | Preview

OID	EID	NAME	DEPT	AMOUNT
10262	8	Megan	Marketing	583.199999392
10265	2	Ashley	Finance	1170.0
10268	8	Megan	Marketing	1098.0
10276	8	Megan	Marketing	400.0

esProc calculation (Cross-database associative filtering)

	A
1	=mysql.query("select OID, EID, AMOUNT from orders")
2	=pg.query("select EID,NAME,DEPT from employee")
3	=A1.switch(EID,A2:EID)
4	=A3.select(["Marketing","Finance"].pos(EID.DEPT))
5	=A4.new(OID,EID.EID:EID,EID.NAME:NAME,EID.DEPT:DEPT,AMOUNT)

Data sources

OID	EID	AMOUNT
10248	5	428.0
10249	6	1842.0
10250	4	1523.4999898076...
10251	3	624.94999976083...

MySQL table:orders(partial field)

EID	NAME	DEPT
1	Rebecca	R&D
2	Ashley	Finance
3	Rachel	Sales
4	Emily	HR

PG table:employee(partial field)

外部JAVA程序通过JDBC调用集算器脚本

JDBC class stored procedure calls SPL script file

```
...
Connection con = null;
Class.forName("com.esproc.jdbc.InternalDriver");
con=
DriverManager.getConnection("jdbc:esproc:local://");
// Calling stored procedures , CountName is the file name of dfx
st =(com.
esproc.jdbc.InternalCStatement)con.prepareCall("call CountName()");
// Execute stored procedures
st.execute();
//Get  result set
ResultSet rs = st.getResultSet();
...
```

JDBC query files directly using SQL

```
...
Connection con = null;
Class.forName("com.esproc.jdbc.InternalDriver");
con=
DriverManager.getConnection("jdbc:esproc:local://");
// Calling stored procedures , CountName is the file name of dfx
st =(com.
esproc.jdbc.InternalCStatement)con.createStatement();
//Query files using SQL,get result set
ResultSet rs =  st.executeQuery("$select name,count(*) from /home/user/duty.txt group by name");
...
```

创新技术
推动应用进步！


各设备故障分析
近7天设备的总故障率
近7天
近1天
近6小时
1.53%
0.95%
2.40%
近7天故障情况
全年故障综合分析
近7天故障次数前5的设备
服务器CPU使用率
今日
昨日
今天硬盘情况