最佳实践

一、数据源

1.1 命名

命名：数据源类型-IP 等命名规则，见名知意

- 数据源命名要准确，避免产生误会
- 如果一个类型的数据原有多个 ip，可以名称后面添加 ip，例如：mysql-137,mysql-138

二、数据集

2.1 命名

命名：层级分布，按主题-子主题-名字命名数据集层级，并区别数据集类型

- 数据集命名的原则同报表一致，按照主题、子主题、数据集名称的层级去设定分布，名称和报表互相对应
- 如果有集市，创建一个 sql 文件夹和一个集市文件夹分别存储

2.2 sql 查询

- 数据量大：建议分区或分表
- sql 语句优化：使用索引或视图等
- 直连数据库的报表，依赖 sql 在数据库的执行效率，sql 执行的效率越高，报表打开的越快
- select 字段名 from 表名 ，避免 select *

避免用 in 或者 not in ，在子查询中，用 exists 代替 in 等。

- 尽量避免勾掉 SQL 数据集、组合数据集和自服务数据集里的“数据库内计算”

风险：如果勾掉，数据集上所有的计算和报表上所有的计算都会进行内存计算。占用大量内存的同时，也可能在服务器上生成大量临时文件，会从数据库加载大量明细数据到服务器

- SQL 里用参数时，select 里的参数尽量加@，避免索引失效

如： select ?{@param1}, market from table1

避免：select ?{param1}, market from table1

- 过滤里如果某条过滤条件不能 push SQL，会导致整体的过滤条件都不能 push SQL，所以建议先将能 push 到 SQL 的过滤完成后，在数据量减少的情况下，再去做不能 push SQL 的过滤

比如：

组合和自服务数据集下推情况：

同源的数据库中，关联、联合、抽样、排序、行转列、分组和聚合、自循环列、镜像、去重节点一般都可以下推到数据库，不同数据库情况不同。

【知识点】

计算列如果不能下推到数据库（push SQL），会以橙色显示。

过滤里某条过滤条件如果不能 push SQL，会以橙色显示。

组合和自服务数据集里如果有不能 push SQL 的操作，会以橙色显示。工具栏的“性能检测”里提示具体内容。

报表里有不能 push SQL 计算存在，会在工具栏的“性能检测”里提示具体内容。

2.3 集市查询

- 建议先做关联再入集市

如果必须做 Join，在数据库里做 Join，结果再入集市，而不是直接在集市里 Join。

如果是一个细节表和多个维度表的关联，可以使用分布式 Join，也就是在入集市的时候，将维度表的入集市任务勾选维度表；

如果是多个明细表，场景上不能先 join 再入集市，可以将这几个表在入集市时，勾 上“分片列”属性，对要 Join 的联接列来进行分片。

风险：内存计算，耗用访问节点（C 节点）大量内存，易导致卡顿或宕机。

【知识点】分布式 Jion 原理：在分布式系统中，当有星形数据（一个大表，若干个小表）需要 join 的时候，可以将小表的数据复制到每个 Map 节点，执行 Map Side Join，而无须到 Reduce 节点进行连接操作，从而提升表连接的效率。可以参考该文章：http://club.yonghongtech.com/thread-9901-1-1.html

- 先建表达式再入集市，避免入集市后建表达式

风险：入集市后创建的表达式（除了报表侧创建的聚合表达式外），都会基于内存计算，集市将所有的数据文件都抓取到内存中，然后耗用内存进行计算。表达式最好是入集市前就创

建好。

- 避免使用 DATA MART 数据源
- 尽量避免随意将所有业务数据抽取到集市

建议先确定要分析的内容，然后选取数据分析过程中，可能会用到的字段，需要分析的范围的数据抽取到数据集市中。

风险：数据量大，列数多，长字符串列，都会导致抽取到集市时间增加，浪费资源且消耗性能。每个数据文件存储 100W 行数据，列数多或长字符串列的时候，单个文件就会大；行数多的时候，文件数量会增加，读取数据的时间就会增长。

- 大数据量集市查询建议使用文件过滤

入集市尽量做增量追加，增量时最好打上 meta，配合文件过滤使用，提升查询性能。

2.4 关联

- 大宽表：按主题划分，可以更便利的进行相关分析，减少表间计算。常用于集市，不适合直连
- 尽量使用 union 代替 join 关联
- 避免上万条数据以上的跨数据源组合查询

风险：跨源组合查询，会将两个数据源的数据抓到内存中，在进行计算，占用大量内存的同时，也可能在服务器上生成大量临时文件

- 可以转换为 sql 的查询，尽量不要使用组合查询

多层嵌套的组合查询，可能会遇到无法下推到数据库的情况，会整体拖慢报表的性能。

- 不推荐大数据量的 SQL 数据集和非 SQL 数据集的 Join。

SQL 数据集和非 SQL 数据集组合，无法下推到数据库，会占用大量的内存，以及会拖慢报告的整体性能

三、制作报告

3.1 常规

- 报告命名：层级分布，按主题-子主题-名字命名报告层级，文件夹层级不超过 8 层

历史、备份、制作中以及正式版本报告区分

- 通过业务分离、逻辑分层，分为多页面展示，通过超链接进行逐层展示

风险：不同业务分析内容放到一个页面或者为了达到美化作用，单页面组件过多，可能导致页面展示缓慢，问题排查困难。

- 筛选类组件：用统一的维表，去掉组件之间的关联关系

风险：过滤组件之间是否关联，默认是关联的，关联的话需要计算所有列的组合关系，如果这些字段不存在包含关系的时候，数据量会膨胀的比较厉害（笛卡尔积），另外如果过滤器的不同值很多的时候，当多个筛选组件的时候，勾选其中一个，其他的组件也会跟着进行数据的过滤，这个时候就会执行 sql 重新抓取数据，在报表页面直接感受到的是报表卡顿，无法再次操作。

- 展示类组件：一屏不超过 6 个组件，每个组件单一数据源，报表不超 3 屏
- 隐藏类组件：建议以 hidden_开头，并统一放置右上角，并通过设置样式隐藏，并置于顶层，或者在脚本中备注

风险：若是对隐藏类组件不统一管理或者标识，一旦出现问题，将增加问题的排查难度。

- 当列不同值较多时，避免进行精确不同值计算，可以用近似不同值替代

当制作报告的数据集为集市数据集市，若是字段的不同值较多，建议使用不同值计数，而不是精确不同值计数，集市数据的计算是在内存中，值较多时会计算耗费时间较长，且占用内存多，会造成系统访问卡顿。

- 打开报表尽量设置不加载全部数据，在页面上加参数根据参数过滤后进行少量数据展现

风险：大量的明细数据从数据库执行出来后，通过网络带宽传输到永洪，报表的打开速度，基本上取决于 sql 在数据库执行时间以及网络对数据的传送速度，影响报表的打开速度，慢且占用内存，大量的内存占用会影响永洪的整体性能，并可能引发永洪宕机的风险。

- 显示明细较多时采用显示部分导出全部的方式

风险：报表数据在导出 excel 的时候，会带有 html 的格式，以此来保留再永洪中设置的样式，如合并单元格，颜色等。导出 CSV 格式的时候，文件没有带格式。因此，excel 文件在生成的时候会比较慢，文件也会比较大，所以会出现导出 excel 很长时间，或者导不出的情况，用户体验较差。如果需要导出大量的数据，推荐用户导出 csv 格式。

- 避免大量明细数据导出

风险：导出明细数据，占用大量内存。假设给永洪分配了 20G 内存，存 1T 的数据。那么在每次用户访问报表的时候，会把报表涉及的数据加载到 C 节点的内存里，假设现在有一张表，涉及到 1 个 G 的明细数据。那么 10 个用户同时导出这个数据，最少也有 10G，考虑到并发，这个占用的内存还会更大。

- 多个筛选条件时最好做批量提交，可以减少报表前端向后台发请求的次数，提升性能

风险：批量提交的作用是，在有多个选项的时候，只有当点击提交的时候，才会执行数据的过滤。当没有设置批量提交的时候，每改变一个选择条件，都会发出一条 sql 申请到数据库，当数据量较大的时候，会占用线程时间较长，那么后面再发出的申请只能等待线程释放。最终的后果，可能造成其他的用户在看报表的时候，一直加载，无法显示报表内容。

- 数据集上的样本行数避免选择全量数据

风险：制作报告时，每一次操作时间上都会有一次查询，如果使用全量数据制作报告，每次计算都用全量数据，也会由于数据量太大导致每次操作都会有查询等。

3.2 计算列

- 计算复杂度越高效率越低
- 避免多层嵌套

风险：如果更改引用字段，或进行名称变更，增加排查问题难度。如，脚本从组件获取数据->将数据赋给参数->参数传递给表达式->表达式绑定到组件中。

- 维度字符串过长时计算效率低
- 能在创建数据集处理的字段，尽量不要在报表端处理

风险：在制作报告新建的表达式无法给其他报表共用，造成重复性工作。

- 性能上 sql 表达式优于 js 表达式

风险：产品支持数据库内计算，使用 sql 函数（sql 表达式），可以将计算逻辑下推到数据库执行；而 js 函数（js 表达式）处理的会在数据抓到内存后再进行计算，占用大量内存的同时，可能在服务器上生成大量的临时文件。

3.3 脚本

- 特别影响性能的：循环，二次计算，复杂逻辑判断，动态效果
- 变量一定用 var 声明，且变量命命名有意义

不声明的话就是全局变量，会给排查问题带来困难，对于没有严格初始化的脚本代码，会带来诡异的问题。

- DataGrid 脚本规范

一般 getData，getViewGrid 返回的都是 DataGrid 的对象，这个对象是支持流式处理的，所以访问指定的行数据之前，需要用 exists 方法判断是不是已经执行完了，并确保有这一行数据。

grid.exists(Integer.MAX_VALUE, c, bool); // 确保执行完，exists：Integer.MAX_VALUE，检测指定行数的数据，当 c 的值为 -1 时，检测所有列数据,当 bool 为 true 时，需要检测的数据不存在时，则继续等待

var rsize = grid.size(-1); // 拿到可用的行数

- 若是修改组件视图值，尽量将脚本写在组件的脚本上，影响范围更小

【知识点】脚本的执行顺序（简要理解）：

a. 报表打开时，定义在仪表盘上的装载时运行的脚本，是最先执行的，且只在报表第一次打开的时候执行。

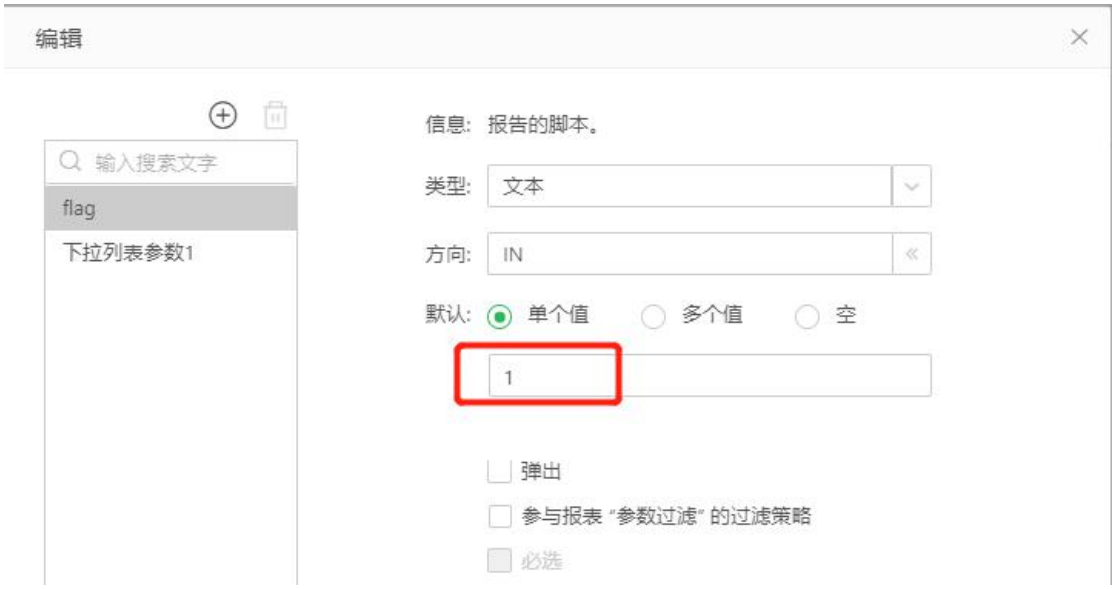
b. 当仪表盘上有任何变化时，设定在变化时运行的脚本被执行。

c. 最后组件上的脚本被执行。

补充：设置默认值

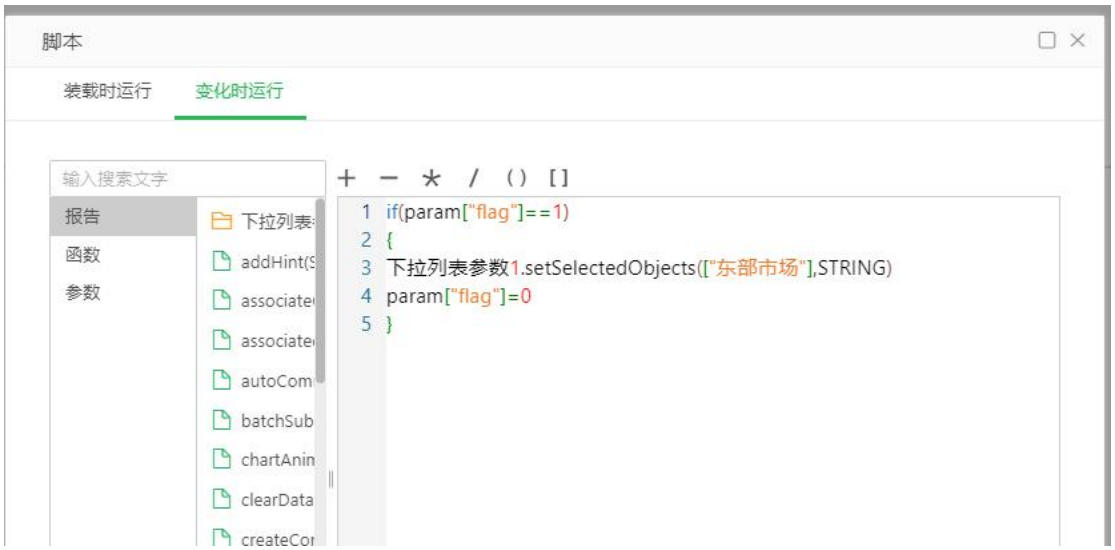
在 BI 中，脚本中设置默认值，通常都是写在报告脚本的装载时运行，但是组件加载顺序和脚本顺序是不可控的，有可能会出现设置默认值不生效的问题，所以最好在变化时运行也将设置默认值的脚本写上，让其执行一次，可以做如下操作：

在编辑参数-新建参数-比如 flag-再将其默认值设置为 1：



并在报告脚本-变化时进行判断，当参数值符合要求时，执行脚本，执行后并将 flag 改成非设置默认值即可：

```
if(param["flag"]==1)
{
下拉列表参数 1.setSelectedObjects(["东部市场"],STRING)
param["flag"]=0
}
```



- 编写脚本时，尽量带上脚本注释

报表完成后，如果带有脚本或者表达式，尽量整理个文档，注明用了什么脚本，原因，作用

等，编写脚本时，尽量带有代码注释。

风险：在进行报表后期运维的时候，如果没有文档记录，制作理念，变更原因等情况，无法了解制作报表的逻辑，增加了运维的难度。

3.4 表格

- 尽量展示汇总数据，避免展示明细数据超过 1W 条

新版本中，明细表格支持真分页，若是在必须展示情况下，可以选择表格分页展示。

- 自由式表格使用太多脚本表达式会导致加载慢

JS 表达式在内存中进行计算，表达式越多，加载越慢。

- 复杂的二次计算可能会导致组件加载慢

计算越复杂，返回数据时间越长，组件加载相应变慢。

- 自由式表格支持多数据源（8.5），数据源越多越慢

自由式表格区别于普通表于交叉表，自由式表格每个单元格都会发起一次查询，自由式表格数据源越多，跨源计算也便越多，则会导致打开报告速度变慢。

3.5 一定要关注静态性能提示和动态性能报告

四、定时任务

- 命名：层级分布，按主题-子主题-名字命名，建议数据集规则保持一致
- 定时任务安排在闲时分散执行，避免任务集中运行

所有任务的执行时间尽量安排在闲时，并且把时间分散开来，避免同一时间执行多个任务，以免在忙时影响用户使用。如果在忙时导数，可以采用单独节点导数，和用户访问区分开。

风险：导数的定时任务，非常耗用资源，占用线程以及内存，如果在忙时导数，用户所发出的请求会出现排队的情况，内存不足也会导致数据处理速度慢，直接导致永洪访问慢，甚至

是宕机。

- 执行任务的 C 节点和制作报表访问的 C 节点分开
- 入集市尽可能做增量追加，增量时尽量打 meta（标签）

meta 可以对数据进行分类，打了 meta 我们在读取数据的时候，就不需要读取所有的文件，只需要读取有 meta 过滤后的文件就可以，减少查询文件的个数，减少花费的总时长。

- 增量导入数据（动态增量）和同步数据集的区别和适用场景

	同步数据集	增量导入数据
区别	a. 全量入集市，数据更新周期长 b. 创建数据集位置可操作 c. 无需新建集市数据集即可使用	a. 可传参，勾选追加增量导入数据 b. 可设置维度表，用于分布式join c. 可分割列入集市 d. 需新建集市数据集引用集市数据
场景	a. 没有支持增量导入的传参字段 b. 数据全量更新，历史数据在变化	a. 数据量大，历史数据不变化 b. 实时性要求不高 c. 数据中存在可识别增量部分的字段，如日期

五、集群相关

- 并发高

多个 C 节点做负载。

- 数据量大

多个 M 节点同时运算或使用直连数据库。

- 最佳节点搭配

N 单独作为一个节点；

RC 最好在一起；

M:R=3:1，集市请求并发高的时候，MR 放在一起，该情况下 MR 压力大，一定要保证内存

充足。

六、服务器

- CPU：核数越小，性能越差，物理核数*2 小于许可核数，则取小值，可能出现性能瓶颈
- 内存：永洪分配的内存为服务器可用内存 1/2-2/3

内存的分配并不是将所有的内存都给到 BI 就是最好的，在服务器运行过程中，服务器自身程序也会占用一部分，一般建议，永洪 BI 的服务器内存分配为：服务器可用物理内存的 1/2-2/3。

- 磁盘空间：保证磁盘空间充足，永洪安装在磁盘空间最大的盘符上，推荐 500G 以上

需要磁盘空间充足是由于我们的 Yonghong/temp/serial 会比较大，serial 文件是序列化到文件系统的数据，serial 文件可以删除，删除该文件夹不会影响到系统的正常启动，不过该文件夹一般在系统重启的时候会自动删除再新建，除非是磁盘空间满了之后需要手动进行删除。大数据量入集市、组合数据集等明细查询有可能会生成大量临时文件，并不会一直无限的增加，以入集市为例，当数据量比较大时 产品内存不够用会临时存储到磁盘空间上，当任务入集市完毕，serial 里的临时文件就自动删除了。

- 网络传输：推荐千兆网卡以上，zb 文件备份数多可能会导致网络 IO 占用高

数据集市是存在 M 节点，将数据存入集市时是通过网络传输的，若 ZB 文件的备份数多，则传输的数据也是多份，在传输的时候就会占用 IO，带宽不够还可能影响集群间的通信。

- 线程数：许可限制总线程数，例如：许可是 C4-E4，对应的总线程数是 4，可并行运行的 sql 查询或者定时任务有 4 个，超过会排队等待

在新版本中，产品线程有自动扩充逻辑（针对 SQL 查询线程），但也只是临时缓解压线程压力，最终的解决方式还是优化 SQL，若是线程充足，线程数为 CPU 的 2 倍核数和许可核

数的最小值。

- 语言环境：确保中文显示正常，需要有中文字体

新服务器中，一般是没有字体库的，确保安装正常，需要先将服务器字体库准备好，另外，若是服务器中不存在中文字体，永洪 BI 的前端展示也可能会出现显示不全等异常情况。

七、其他

- 浏览器最好使用 ie11,chrome 最新 3 个版本和 Firefox 最新 3 个版本
- 权限区分，查看用户，开发用户，创建数据集的用户，权限严格区分，只开放对应权限，避免误操作
- 系统设置参数配置：

join.grid.maxrow=10000000（默认 1000 万），后期处理的 join，数据行数超过最大值后预警提示， 建议默认设置小点例如 100000，尤其是给大量业务用户使用时；

max.load.rows=5000000（默认 5000 万），设置组件默认加载的数据行数，不影响导出，导出的数据 为全量数据；

max.export.control=true，控制报表和组件默认导出的数据行数。默认 true 代表控制导出，设置成 false 表示不控制导出数据量。跟随 max.load.rows 的查询逻辑，查询能控制量的地方导出都能控制住。

- **数据库配置**

管理系统-系统设置-数据空间配置的数据库，不能和管理系统-系统设置-数据库连接配置中配置的库是同一个。