UME的配置功能的Review

1、当前系统的配置问题总结

1.关于规划区的使用问题： 15年我们曾经访谈了24个售后和网优人员，对ICM反馈问题整了一个excel，TOP4的几个问题为：
1）流程较多，包含规划创建、导出导入、合法性检查、冲突检查、下载、激活流程等，使用繁琐
2）每一步骤都都有校验，停顿较多，流程很难连贯，不出错。
关于这个ICM本身也做了很多优化，如检查的最大努力方式，并且扁平化也是最大努力方式
3）效率慢、而且锁的问题严重，icm有锁、ommb也有网元锁，到激活阶段失败的概率高

4）全参数模版不支持连带，功能和ommb不对称，如小区修改会自动修改外部邻区、功率更新等自动连带计算都不支持，所以支持的场景有限，没有ommb上好用
5) 规划区范围有限，邻区如果超过这个规划区就无法操作，而规划区太大太多对整体系统容量有影响

这些问题在扁平化EMS网管中我们逐条分析改进过：

如针对流程多，扁平化EMS设计了快速规划区(快速通道)，即小批量操作不需要创建规划区，可基于现网区直接选择网元，自动创建临时规划区，激活完自动删除
如锁问题，用排队机制取消了锁的概念
如规划区支持自动扩展，自动拉入邻区网元，规划区可以分为普通规划区和带邻区的规划区等
这些问题在OES上需继续跟踪、比较，除了易用性提升，主要还保障效率

【张敦华2017-1-6】1主要是配置效率，可靠性和系统架构合理性的问题。

2.关于安全鉴权

授权：属于UEP框架的处理，新系统后授权这块如何设计协作，张敦华总体考虑

鉴权：目前UCF框架：把操作的操作码、用户、资源(网元、MO、参数等)版定封装起来给UEP进行鉴权，根据结果判断是否有权限

弊端：之前用户和plmn不关联，拿到用户不知道plmn，网络共享需求来了导致该部分和UEP一起重写

和产品耦合，没有给产品预留灵活的开发和定制接口，如GU和LTE针对不同用户要分类的MO和参数鉴权逻辑不同，导致UCF写了一堆多余的适配和接口

另外，网络共享类似需求，所有需要查询和展现数据的地方都需要过滤，新系统中针对该需求要重新研讨设计

【张敦华2017-1-6】要解决2个问题，一个是数据隐藏，一个是安全鉴权。隐藏比较麻烦，鉴权相对比较简单。对于虚拟运营商，可以只提供gui界面，其他功能屏蔽。在gui界面中，可以增加一个结果过滤器，对查询的结果进行过滤，隐藏无权限的内容。安全鉴权流程，用户的plmn是知道的，将比较用户plmn和它修改的mo对应的plmn是否一致作为一个合法性检查项就可以实现plmn的鉴权。

【刘梅红2017-1-9】对于数据的隐藏，需考虑通用的过滤算法，各个模版和工具都需要显示和导出过滤，这些过滤算法也可以放到工具层。

3.数据一致性问题

配置 数据从NE-OMMB-ICM-EMS(北向)需要4个系统之间一致：

最常见的是OMMB和ICM的数据一致性，会通过北向网管核查暴露出去，该问题从12年攻关至今，也只是把出现的和能想到的漏洞都防住，但本质多系统多数据的拷贝就是存在这个弊端

其次是NE和OMMB之间的数据，经过几年的改进和优化，业务字段已没有不一致问题，但近2年来2次出现动态字段问题：

如小区的人工操作状态：即是配置属性、又能动态设置生效，和数据激活同步配合起来，出现了前后台不一致，网管同步后导致正常小区被闭塞(新系统该类字段只保留配置更新这一个途径，理论上可避免)

新系统只有一层网管，和网元之间是否还会出现数据一致性问题，也需要在激活服务流程中考虑

【张敦华2017-1-7】ume配置集中管理，去除了资源模块中网元之下的配置数据，状态字段在配置系统中管理。这些改进可以从根本上解决这个问题。

【刘梅红2017-1-9】主要考虑数据同步的一致性，规划区把差异数据发给激活，激活和上次网元的快照数据比较生成增量文件

4.一些易用性竞争力问题
如配置报表，一直被吐槽我们就一个excel导出，很土、也不支持定制扩展，这个除了自定义之外，web呈现和图形统计等也可以研究
如宏站和微站、以及AM、BM的关系等，用户想看完整的TOP全貌等，也可以有一些图形化的呈现和统计

【张敦华2017-1-7】模版采用可导入，热加载的方式，增加模版定制发布的灵活性。

ume配置改进点	可以解决的当前痛点	

ume微服务的功能划分	1，3	
微服务的弹缩设计	1	
现网区多数据库实例的改进	1	
规划区数据库只记录变更数据	1	
使用用户服务层接口替代mo连带	4	
配置web界面的优化	4	
支持高效的关联查询	1	

2、UME的架构实现及对比

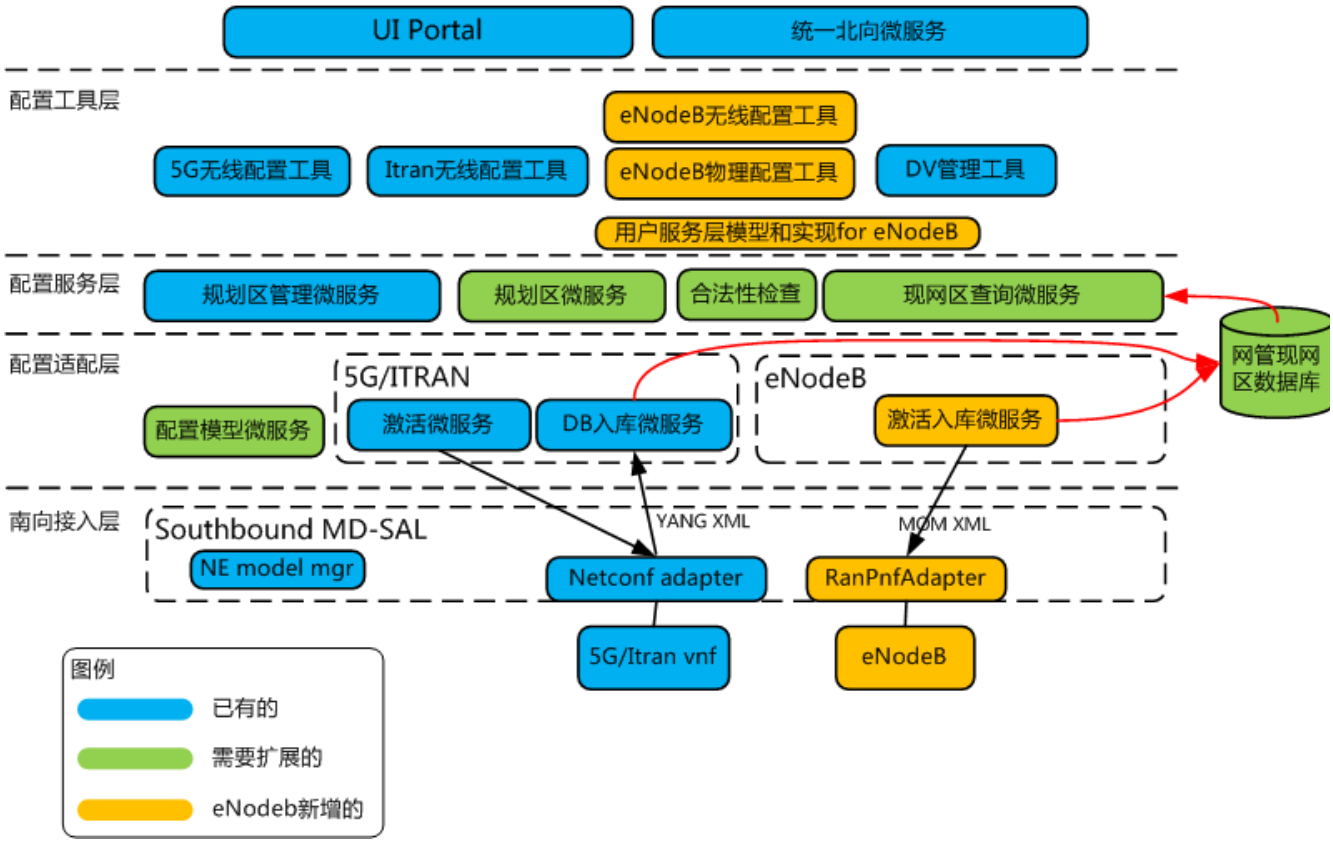
2.1 ume微服务的功能划分

当前系统在架构上的主要问题在于：

1、ommb缺少激活队列，没有一种直接操作运行配置数据的途径。

2、另外icm和ommb功能重复，定位也不清晰，用户操作不便。

ume系统实现了统一的集中配置系统，实现激活的队列管理，从根本上来说解决了当前系统的问题。



我们把微服务分成了多个层，只有上层微服务才能依赖下层的微服务，这样结构比较清晰。

在配置适配层中，激活微服务和db入库微服务负责对网元数据的接入。后续对激活微服务做增强，在不依赖配置数据区的情况下，使之能够支持网元的在线增删改查配置命令。

配置服务层的规划区微服务和现网区查询微服务是通用的数据区服务。由模型驱动，不涉及业务逻辑的硬编码。

配置工具层集中了模版和用户服务层接口。这一层集中处理配置的业务逻辑。大部分业务逻辑都是通过模版算法，用户服务层的模版定义来实现的

2.2 微服务的弹缩设计

微服务的弹缩，主要需要考虑好如下架构设计：

1. 服务集群化，即可使用更多的微服务实例服务更多的用户请求。
2. 微服务接口服务无状态化，即状态跟着任务走，或者通过分布式缓冲或数据库存储任务状态。
3. 服务化，即将一个大的服务拆细，每个小服务使用不同的资源，这样服务可以并发，并且不会因为共享资源而阻塞。
4. 异步化，将微服务接口做成异步接口，通过消息机制联络。从而避免服务间的同步阻塞。
5. 容错设计，在设计过程中要考虑某些服务失效后的善后机制。

配置中各微服务的弹缩设计：

1. 有些微服务对管理网元的数量/用户数量不敏感，这些微服务在系统中是单例。有：配置模型微服务，规划区管理微服务。

2. 按用户的命令数量进行弹缩。当接入用户多，发起的操作请求多时，增加微服务实例的数量。这类微服务有：规划区微服务，现网区查询微服务，模版工具微服务，激活微服务，合法性检查微服务。这种架构可以将一个用户的命令可以拆分成多个命令，交给多个微服务并发完成，最后汇总结果。设计思路如下：

    - 增加一个总控微服务，总控微服务做3个事情：1、拆分任务文件；2、分发命令；
      3、汇总生成结果文件。总控微服务提供需要费时较长的任务服务，比如1000个网元的数据导入。
    - 总控微服务负责解析任务文件，解析后，如果需要操作的网元数量比较多，就拆分成多个子命令，每个命令只操作少量网元。然后将命令分发给多个执行微服务。
    - 多个执行微服务并发执行同一个用户任务，最终把结果反馈给总控微服务。
    - 总控微服务收集结果信息，生成结果文件，返回给用户。

    执行微服务的弹缩策略是当执行微服务实例正在执行的平均任务数大于2时，增加新的微服务实例。反之当平均任务数小于某个值时，减少执行微服务实例。
    总控微服务虽然是接入点，但是性能压力不大，可以根据网络规模按比例给出一个固定的实例数。

3.
按管理网元数量进行弹缩的微服务。管理的网元数量增多后，就扩展更多的微服务实例。这类微服务有：db入库微服务。这类微服务都是根据网元的数量给出明确的微服务实例个数。

基于Paas微服务弹缩并发架构，可以通过将大任务拆小并发，从而提高了任务处理速度。

2.3 现网区多数据库实例的改进

当前的配置系统，无论ommb还是icm，都是使用单个配置数据库，一套大表。这带来2个问题：

1、网元数量多时，数据库查询效率下降。

2、当管理的网元种类很多时，所有类型网元的数据模型混在一套表中，难于维护，升级困难，甚至会导致模型冲突的问题。

ume配置准备考虑支持多个现网区数据库引擎实例，可以把不同类型的网元放在不同的实例中存储，当一种网元类型数量过多时，也可以拆成多个数据库实例存储。拆成多个库还有一个好处是各数据库之间不会互相影响，比如一个库升级不影响另外一个库。一个库的网元有问题不影响另外库的网元。

目前考虑按照子网为粒度拆分现网区。即一个子网内的所有网元只存储在一个数据库中，多个子网的网元数据可以存储在不同数据库中。

当用户发起配置请求时首先确定请求对应的子网，然后根据子网查找对应的数据库引擎。利用2.2的总控微服务，可以使多个库的使用对用户透明。

按照上述设计，ume的配置具有无限扩展的能力。可以做到管理的网元数量无上限。

2.4 规划区数据库只记录变更数据

当前的icm，创建规划区需要首先选择网元，然后从现网区拷贝网元数据到规划区，然后才能由用户对规划区进行操作。这种方式存在多个问题：

1、如果用户选择很多网元，即使只修改一点点参数，也需要花费大量时间来拷贝现网区数据。拷贝数据需要花费大量时间，给用户感受不好，特别是全网网元修改一个参数时，操作效率非常低。

2、网元之间某些参数会有关联，比如修改一个网元的小区，要求同步修改其他相关网元的外部小区。如果这些网元在最初选择时没有包含在

规划区中时，就不好操作了。

3、规划区拷贝的数据存在过期的情况（其他用户的修改导致现网区变更），时间越长，过期的情况越严重。这导致合法性检查失败的概率大大增加。

邬春会研究发现如果规划区数据表和现网区数据表在同一个mysql引擎中时，可在规划区中只记录用户增删改的mo变更信息，通过一种复杂的sql查询语句可直接看到现网区数据叠加规划区数据后的结果。这样我们将可以在创建规划区时，不需要用户选择网元，而是创建一个空白的规划区，用户查询规划区数据时，看到的是完整的现网区数据。当用户修改某些网元的数据后，则可以看到修改后的规划区数据。

这个功能实现后，就可以很容易的解决上述2个问题，并大大提高规划区的处理效率。

梅红在上面说的icm问题： 1）流程较多，包含规划创建、导出导入、合法性检查、冲突检查、下载、激活流程等，使用繁琐。每一步骤都容易失败。

在新的架构下，流程变成 规划创建（只输入规划区名字，不选网元，速度很快）、导入导出（如果需要用模版的话）、合法性检查、激活。流程步骤失败的概率会低很多。

2.5 使用用户服务层接口替代mo连带

在模版工具微服务中，使用groovy实现一套dsl语法，实现复杂配置业务逻辑的描述和定义，比如创建超级小区，一键邻区增删。从而替代了现有系统的mo连带。这样使得系统结构更清晰，也便于故障定位和保证效率。

用户服务层接口的介绍：用户服务层框架实现方案

2.6 配置web界面的优化

配置的操作界面中，将原来的规划区操作，现网区查看，网元批量修改界面合为一个统一的数据编辑查看组件，使用模型驱动的方式进行展现。

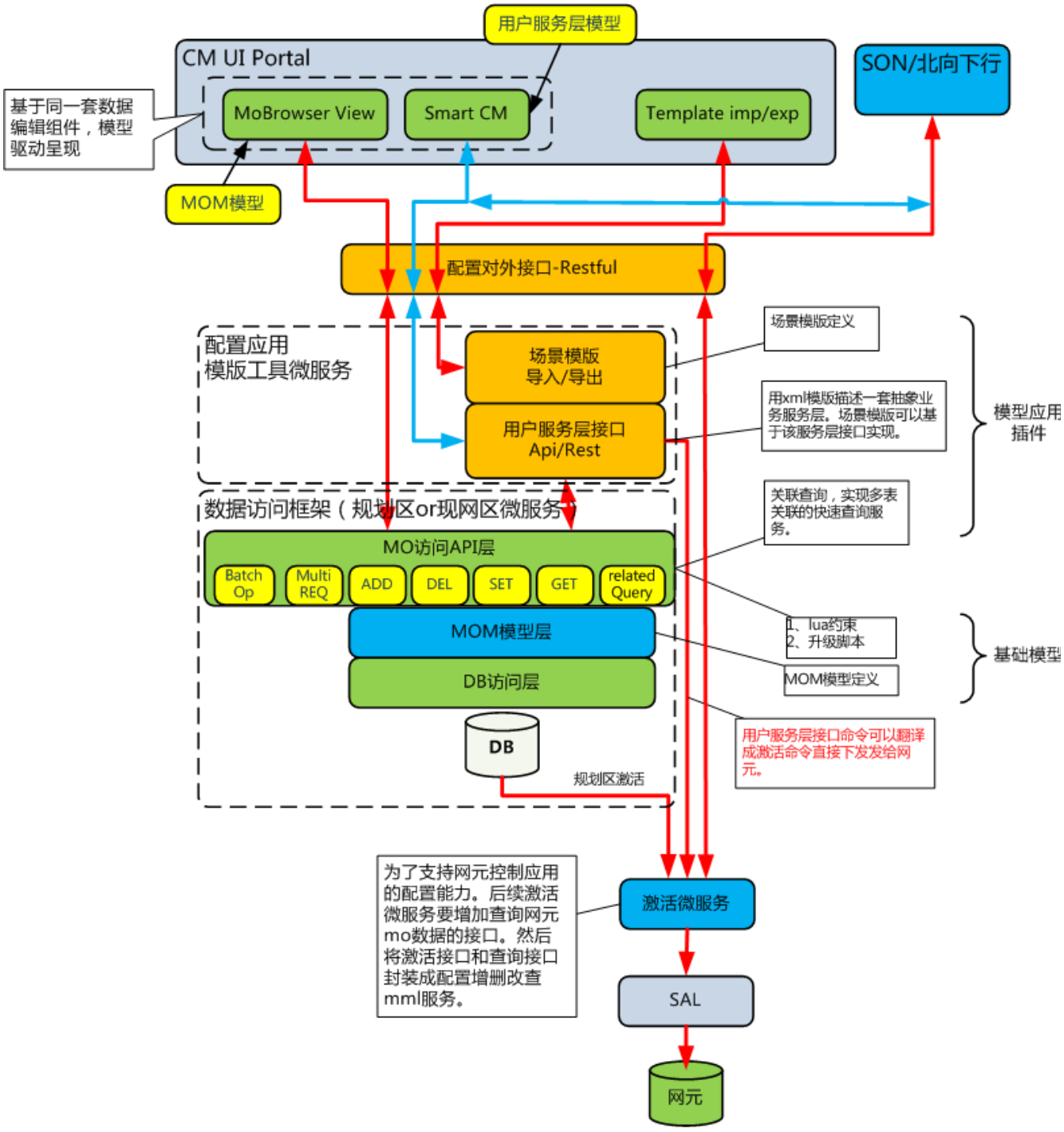
特别是规划区单网元的操作和批量网元的操作合并到一起，方便用户在界面上的操作。

ume的数据编辑查看组件是模型驱动方式工作，所以还可以为其他业务提供数据操作展现。比如后续定义了很多用户服务层的接口服务，这些接口当前只能通过restful和场景模版调用。我们还可以把这些用户服务层接口服务按照界面组件要求定义成一套新模型，然后用数据编辑查看组件来加载这个新模型，这样就可以实现在界面上直接操作用户服务层的命令了。

另外，enodeb网元把它的master默认值和连带功能集成到一个新的配置工具层服务中，那么它可以把这个新的服务按照界面组件要求定义出一套模型，就可以用界面展现支持master默认值和连带的界面了。

2.7 支持高效的关联查询

在现网区查询微服务中，提供了关联查询的框架，支持用户定制一些高效的关联查询服务。


用户服务层模型
CM UI Portal
MoBrowser View
Smart CM
Template imp/exp
SON/北向下行
基于同一套数据编辑组件，模型驱动呈现
MOM模型
配置对外接口-Restful
配置应用
模版工具微服务
场景模版
导入/导出
用户服务层接口
Api/Rest
场景模版定义
用xml模版描述一套抽象业务服务层。场景模版可以基于该服务层接口实现。
模型应用插件
关联查询，实现多表关联的快速查询服务。
数据访问框架（规划区or现网区微服务）
MO访问API层
Batch Op
Multi REQ
ADD
DEL
SET
GET
related Query
MOM模型层
DB访问层
DB
1、lua约束
2、升级脚本
MOM模型定义
基础模型
用户服务层接口命令可以翻译成激活命令直接下发给网元。
规划区激活
为了支持网元控制应用的配置能力。后续激活微服务要增加查询网元mo数据的接口。然后将激活接口和查询接口封装成配置增删改查mml服务。
激活微服务
SAL
网元