# SAF2.1 需求整理+设计

工具

被章耿添加，被鲍宁天最后更新于五月 23, 2014



## 注册中心（registry）需求大纲

| 需求 | 需求描述 | 需求类型及优先级 | 细化负责人 | 备注 |
|---|---|---|---|---|
| 服务索引 | index服务，负责指向所有的registry<br><br>将服务与registry分组划分，每个服务（InterfaceID+group？？？）对应一组注册中心。当服务启动时，先连接index服务，获取对应注册中心列表<br><br>对应关系保存在mysql中，每个index服务启动时，从mysql中获取数据，并缓存下来。并且定时（每10分钟）从mysql中更新数据<br><br>在管理端配置服务于registry对应关系 | | 鲍宁天 | |
| 高可用性 | 1. 数据库宕机<br>2. registry宕机，连接不上<br>3. registry死亡，连接上了，但是没反应 | 性能需求高 | 鲍宁天 | |
| 本地数据缓存 | jvm本地缓存，加快查询速度，同时数据库挂了也能容灾<br><br>1.InterfaceMap<String, Provider><br><br>2.实例心跳IPPIDMap<IPPID, timestamp><br><br>3.ServerAlias<br><br>4.global_setting 所有参数配置：monitor global_setting, 心跳时间<br><br>5.proxy_conn<br><br>6.RouteMap<String, RouteObject><br>source->dest | | 鲍宁天 | |
| 服务注册 | 服务节点实时同步注册。注意：服务节点防重注册<br><br>如果mysql挂了，直接抛异常<br><br>服务节点唯一约束，provider是ip+port+group+version+protocol+serviceInfoId（serviceInfoId是interfaceID在mysql的主键），consumer是ip+pid+group+version+serviceInfoId<br><br>节点注册数量限制。防止同一个IP注册无数个节点（由于配置不当，或者性能测试会出现无数个） | | 鲍宁天 | |
| 服务列表订阅/推送 | 从mysql获取服务划分规则，缓存对应的服务列表。并且每1分钟更新数据（可先判断，再更新）<br><br>服务列表订阅（全量订阅/增量订阅？？？）<br><br>服务列表推送（增量推送？？？）<br><br>根据事件查询，事件通知广播其他registry | | 鲍宁天 | |

| 需求 | 需求描述 | 需求类型及优先级 | 细化负责人 | 备注 |
|---|---|---|---|---|
| 服务心跳 | 采用缓存批量更新的方式，每隔1分钟往mysql数据库批量更新心跳时间<br><br>心跳判断参数版本是否需要配置下发<br><br>心跳间隔10秒？ | | 鲍宁天 | |
| 配置下发 | monitor信息配置，心跳间隔，推拉结合 | | 鲍宁天 | |
| WORKER | 1. 服务节点状态扫描，根据服务节点心跳时间判断状态，3分钟没有心跳为死亡状态，5分钟没有心跳就删掉非固定端口节点。下线服务节点的操作也一样。<br>2. zookeeper节点同步到saf2.0的mysql中<br>3. 检查redis与mysql数据一致（废除）<br>4. 状态检查时，如果死亡，就记录上下线日志<br>5. 心跳倍数判断死亡和删除 | | 鲍宁天 | |
| 监控数据 | 记录时间段内，客户端与注册中心连接事件断开的数量，保存到数据库中。<br>单位时间内，注册中心收到的数据包的大小<br><br>统计连接注册中心的客户端订阅数量 | | | |
| 数据库连接数 | 由于mysql连接数有限，默认是150，所以在考虑index服务和registry服务的实例部署时，需要规划下每个实例的mysql连接池数量。需要咨询下mysql dba<br><br>比如：index服务部署5个实例，每个实例可以配置3个mysql连接<br><br>registry服务部署10个实例，每个实例可配置5个mysql连接<br><br>worker服务部署1个实例，每个实例可配置10个mysql连接<br><br>总数就是15+50+10=75。如果有其它用户共享mysql实例，连接数可能超过最大值 | | 鲍宁天 | |
| 服务分组 | 在管理端实现动态分组：<br>1. 手工对服务端进行分组<br>2. 同时将刷新服务端订阅关系<br>3. 将最新的数据推送给调用端，或者等调用端重新订阅 | 高 | 章耿 | |
| 服务ID | 将目前服务通过（服务接口＋版本＋group）的模式进化为通过唯一的服务ID来标示，<br>~~1.可以通过消息摘要等方式（不可逆，有其它可逆的最好）进行生成较短uuid。~~<br>2.生成一段可读的较短的服务ID | 中 | 章耿 | |
| 跨机房容灾 | MySQL:<br>1. 主写从读，主挂了，手动切换从开始写<br>2. 双写 不推荐<br>3. 主从部署的时候，如果有机器，从也可以跨机房部署，一个主可以部署多个从库<br><br>Registry:<br>无状态服务，跨机房部署多实例 | 高 | 章耿 | |
| 水平扩展 | 最好避免IP直连注册中心。在注册中心扩大后，不需要服务修改配置文件：<br>已用VIP实现，域名连接。 | 高 | 张俊峰 | |
| 安全性 | 为了防止恶意访问（非法访问），可以通过以下方式（方法）在访问注册中心或者Index服务前对访问者进行必要的检查：<br><br>方案1：通过黑白IP名单、IP地址检测（只有内部IP可以访问）的方式<br>方案2：权限分配；管理员分配签名或者其他方式 | 中 | 张俊峰 | |
| 数据一致性 | 当在注册中心注册的某个服务的状态发生变化时（或其他变动的情况），所有调用此服务的消费者得到的信息应该是相同的——不管连接的是否是注册中心的同一台机器。<br>目前心跳机制可以保证最迟1分钟的数据状态同步，可以考虑注册中心实例间互通，达到数据实时一致性 | 低 | 张俊峰 | |

# RPC框架需求大纲

| 需求 | 需求描述 | 需求类型及优先级 | 细化负责人 | 备注 |
|---|---|---|---|---|
| Spring集成 / API 方式启动 | 通过Spring注入，或者直接API方式启动服务：<br><br>Spring集成：基于Spring可扩展Schema提供自定义配置进行扩展开发，在启动时利用spring的自动加载方式进行启动；<br><br>API方式：类似于Gaea中的自定义配置文件方式，设置好对应的目录，解析配置文件进行启动。main函数直接启动<br><br>建议采用Spring集成方式，可以最大化利用已有的代码。在连接不上注册中心利用本地配置启动时，可采用API启动方式。 | 高 | 张俊峰 | |
| xml / annotation方式声明 | XML方式：用<namespace:tag 属性1=”值1” 属性2=”值2” />的方式进行显示声明-<br>Annotation方式：定义各种不同含义的注解，在启动时扫描class文件取得相应的注解和属性。 | 高 | 张俊峰 | |
| 注册中心寻址 | 连接注册中心Index服务，提供当前机器环境（IP，机房等）、服务归属（项目、部门等）以及服务配置（saf版本等）给index服务，得到最佳的一组注册中心地址：<br><br>方案1： K-V结构直接寻址。通过提供当前机器环境（IP，机房等）、服务归属（项目、部门等）以及服务配置（saf版本等）作为Key从Index服务中取得最优注册中心地址；<br><br>方案2：多级（三级）列表结构间接寻址。通过提供当前机器环境（IP，机房等）、服务归属（项目、部门等）以及服务配置（saf版本等）进行多级分层查找，从Index服务中取得最优注册中心地址 | 高 | 张俊峰 | |
| 服务发布 | 监听一个端口，一个端口对应一个protocol，一个端口后可以发布多个服务。<br>要有服务方法级的发布限制。<br>方法注册有两种方案<br>1 通过include，exclude标签定义方法是否需要发布<br>2 在方法标签中，通过register="false" \| "true" 指定是否发布 | | 常洪强 | |
| 服务注册 | 1 将发布后的服务信息注册到注册中心<br>根据发布服务的url, 将服务实例信息写入到mysql数据库<br>2. 服务注册前判断服务是否已经发布, 若已发布则直接返回，注册失败要destroy服务。 | | 常洪强 | |
| 多协议发布 | 一个服务端可以同时指定发布多个协议，例如同时发布saf和rest协议，不同协议监听不同端口 | 低 | 常洪强 | |
| 服务分组、服务ID | 1.服务端发布服务需要指定group+version，调用端调用服务需要匹配服务端的group+version<br>2.新增服务别名的方式，对服务端指定服务别名，调用端调用时匹配服务端的服务别名 | 高 | 章耿 | |
| 服务订阅 | 调用端根据配置的group+version或者服务别名，连接注册中心获取服务端列表。<br>1.方案一：长连接/短连接，调用端定时主动获取<br>方案二：长连接，调用端心跳时检查版本，有变化则主动获取<br>方案三：长连接，注册中心主动推送<br>2.最好是服务端推送最新的服务列表（可以不推全量，只推送变化部分）<br>3.必须有保障机制，定时检查功能<br>4.本地需要保持一个文件、内存服务列表。 | | 章耿 | |
| 本地调用 | 本地发布一个服务服务端，同时又配置一个一个该服务的调用端（同一jvm内）<br>此时要求走本地实现，而不走远程rpc调用 | 中 | 章耿 | injvm |
| 异步调用 | 调用端发送请求给服务端后，无需等待结果，返回一个future，可执行向后执行代码<br>通过future.get异步得到返回结果 | 高 | 章耿 | |
| 隐式传参 | 调用端调用的时候，可以传递一些附加参数给服务端（类似上下文里有个额外的Map） | 低 | 章耿 | |
| generic调用 | 1.服务端默认开启接收泛化调用请求<br>2.调用端无需知道服务端的接口描述（不需要接口类），直接通过一个泛化调用将请求参数<br>发送给服务端，服务端直接返回结果 | 高 | 章耿 | 用于网关应用 |

| 需求 | 需求描述 | 需求类型及优先级 | 细化负责人 | 备注 |
|---|---|---|---|---|
| 直连调用 | 无需从注册中心订阅服务端列表，直接通过配置设置服务端列表，进行连接调用 | 中 | 章耿 | |
| 连接控制 | 1.延迟连接：调用端启动时候不主动和服务端建立长连接，等到调用时再建立长连接<br>2.粘滞连接：如果一个服务端可用，则一直使用，直到这个服务端不可用再去建立其它服务端 | 低 | 章耿 | |
| 集群策略选择 | 1.failover，失败重试，当出现失败则自动切换重试其它服务器，通常用于读操作（推荐使用），<br>缺点是重试会带来更长延迟<br>2.failfast，快速失败，只发起一次调用，失败立即报错,通常用于非幂等性的写操作，<br>缺点是失败几率大用户体验差<br>3.failsafe失败安全，出现异常时，直接忽略，通常用于写入审计日志等操作，缺点是数据会丢失<br>~~4.failback失败后台自动重试~~<br>~~5.forking并行调用多个服务器，一个成功即成功~~ | 高 | 章耿 | |
| 过滤器扩展 | 方案1：类似于Netty中的encoder和decoder，基于netty的handler类封装一个Filter基类，要求开发者实现指定的过滤方法来完成过滤功能。<br><br>方案2：基于Spring中的拦截器扩展实现。<br><br>方案3：自定义一个过滤器的基类，让用户实现其指定的方法完成过滤功能 | 高 | 张俊峰 | |
| 序列化方式 | 二进制序列化（MessagePack）、JSON序列化：<br><br>默认采用MessagePack序列化，可以修改为指定方式。<br><br>MessagePack：序列化的类需要在MessagePack中注册或者加上注解@Message。<br><br>方案1：项目启动时扫描配置的package，自动注册序列化类；<br><br>方案2：项目启动时扫描所有jar文件，自动注册Serializer的子类或某一接口的子类；<br><br>方案3：需要序列化的类都加上@Message注解<br><br>JSON序列化：建议使用方案1<br><br>方案1：使用FastJSON实现；<br><br>方案2：使用Gson实现 | 高 | 张俊峰 | |
| 通信协议 | TCP、HTTP协议支持：<br><br>TCP协议：默认方式。在TCP传输的数据包中自定义协议：包括协议头和协议数据包<br><br>HTTP协议：扩展方式。HTTP协议采用RESTful规范 | 高 | 张俊峰 | |
| 服务端业务处理线程模型 | 在IO线程（EventLoop）完成基本的截包及包头解析操作，截完之后调用用户线程池完成业务相关数据decode操作、以及具体业务操作；<br>为某接口，或者接口类中的某个方法定义可使用线程数，如果超过并发上限则直接报已无线程资源的异常；<br>在请求积压时，根据请求的优先级对请求进行排序，优先级高的先执行，同时执行时判断请求是否已过期，如果已过期则直接丢弃； | | 李鑫 | |
| telnet运维 | telnet取回SAF服务的相关信息，包括过去一分钟执行情况，每个方法被调用了多少次，成功执行了多少次，数据包的大小是怎样的；<br><br>服务接口名称、服务的版本、服务当前建立的链接数以及具体与哪些调用者建立了链接； | | 李鑫 | |
| 数据压缩 | 业务数据在序列化之后，再经过一次数据压缩以缩小体积，适合大数量调用时使用； | 性能需求低 | 李鑫 | |
| 自动先从缓存中读取 | 调用时先根据调用参数生成的key在redis缓存中取一下，取不到对应缓存值的情况下再真正进行远程调用； | 性能需求低 | 李鑫 | |

| 需求 | 需求描述 | 需求类型及优先级 | 细化负责人 | 备注 |
|---|---|---|---|---|
| callback回调 | 此调用应为异步调用，调用端的调用参数中应有一个参数继承并实现由SAF预先定义好的一个callback接口，相当于client端也发布一个，<br><br>服务端在执行业务逻辑完毕后回call客户端； | 功能需求低 | 李鑫 | |
| 参数验证 | 制定参数检验规则，不符合规则的参数不发送，遵照JSR303规范，客户端在发送请求前对调用参数（数据bean）进行验证，如<br><br>验证失败则直接抛异常不进行远程服务调用； | 功能需求低 | 李鑫 | |
| 兼容1.0的RPC协议 | 为了兼容线上已有的SAF服务，对SAF1.0所使用的dubbo通信协议进行兼容，但部分参数与功能不支持，如dubbo的回调； | | 李鑫 | |
| 2.0 RPC 协议 | 根据调用的需求设计自有协议，在head中包括如下信息：megic code（？1个字节）、msg length、msg type、 msg id、interface Id、 msg 压缩方式（0不压缩）；head后面是具体业务的消息体 msg body | | 李鑫 | |
| 接口类上传，并自动生成接口描述 | saf客户端自动将所发布的服务接口，上传到服务器，服务器接收后，予以解析； | 帮助提升用户体验的功能 | 李鑫 | |

# 服务治理需求大纲

| 需求 | 需求描述 | 需求类型及优先级 | 细化负责人 | 备注 |
|---|---|---|---|---|
| 服务归属 | 服务联系人<br>1 服务注册(发布)时需要提供相关负责人, 部门, 接口描述等相关信息;<br>2 (?)服务订阅需要提供consumer端负责人及部门; | | 常洪强 | |
| 服务查询 | 主要功能在管理端实现：<br><br>1 按照关键字查询服务提供，查看api等<br>2 查询服务归属, 服务下发布的方法, 及相关参数.<br>3 根据服务器IP查新相应的注册服务, 以及所属端 | | 常洪强 | |
| 服务路由 | 在管理端配置路由规则，调用端根据路由刷新服务端列表<br>在管理端可配置：动态路由, IP路由, 服务权重, 方法路由等. | | 常洪强 | |
| 服务上下线 | 1 由worker定时扫描所有服务提供端实例，3分钟没有心跳为死亡状态，<br>5分钟没有心跳就删掉随机端口节点<br>2 服务上下线记录, 查询, 判断是否需要报警.<br>3 管理端对服务动态上下线 | | 常洪强 | |
| 负载均衡 | 1 调用端可以知道负载均衡算法，包括随机，轮询<br>每次invoke的时候根据调用端url得到负载均衡配置, 实例话相应的负载对象进行select<br>2 管理端动态管理负载策略, 下发配置 | | 常洪强 | |
| 并发限流 | 服务端可以限制调用并发送，保护自己<br>服务端Filter实现，每次调用时，在服务端判断当前配置的并发数是否超过已缓存过的并发数，<br>是: 异常返回; 否: 继续执行 | | 常洪强 | 服务自我保护 |
| 服务能力水平扩展 | 动态添加、删除服务节点 | | 常洪强 | |
| 调用次数限制 | 服务端可以限制单位时间内某IP的调用次数，在单位时间内调用次数达到了以后，对于后面收到的请求，直接返回异常<br><br>或错误 | | 李鑫 | 服务自我保护 |
| 调用者优先级 | 可以设置服务调用者的优先级，请求积压时优先执行优先级高的请求；请求中加入优先级这个属性 | | 李鑫 | |

| 需求 | 需求描述 | 需求类型及优先级 | 细化负责人 | 备注 |
|---|---|---|---|---|
| 服务授权（token） | 在SAF管理端进行服务预授权后才允许进行调用； | | 李鑫 | |
| 服务黑白名单 | 只有白名单中的ip才能进行调用，此需求点与上一个需求有重复，考虑融合到上一个需求之中； | | 李鑫 | |
| 服务降级 | 服务分为强依赖服务以及弱依赖服务，强依赖服务不可降级，而弱依赖服务可降级－－通过开关可直接bypass此服务，<br><br>而调用端此时直接返回事先设定好的默认结果； | | 李鑫 | |
| 服务质量指标收集 | 收集服务调用的质量指标，发送给监控服务或者输出到日志供监控服务收集 | | 鲍宁天 | |
| 服务质量数据处理 | 存储至hbase、生成统计数据。<br><br>统计数据按小时、天进行统计 | | 鲍宁天 | |
| 服务能力评估 | | | 鲍宁天 | |
| 服务质量dashboard | 展示所收集到的服务质量指标 | | 鲍宁天 | |
| 在注册中心提供服务调用页面 | 方便服务开发人员debug服务 | | 鲍宁天 | |
| 根据1.0配置文件自动生成2.0配置文件 | saf2.0配置项可能改变，需要提供1.0到2.0的xml转换工具类。 | 低 | 章耿 | |
| 线程过载保护 | 1.服务端在线程池已耗尽时直接返回错误，客户端减少请求此服务端<br>2.服务端一个端口如发布多个服务，则每个端口后的线程池可以考虑按服务进<br>行定量分配，不要因为一个其中一个服务慢堵住了线程池 | 中 | 章耿 | 服务自我保护 |
| ~~服务编排~~ | saf是原子级别的调用，是否需要开saf-bus项目，集成服务编排功能（jos?） | 无 | 章耿 | |
| ~~分布式服务追踪~~ | 1.通过日志的方式记录下服务质量指标，保护关键字（配置或者指定）<br>2.通过统一的agent进行数据收集<br>3.进行图表展示 | 低 | 章耿 | |

# 参考文档

1. SOA 治理简介 https://www.ibm.com/developerworks/cn/architecture/ar-servgov/
2. 探讨服务存储库和注册中心在面向服务的体系结构（SOA）中的角色 http://www.ibm.com/developerworks/cn/architecture/ar-servrepos/
3. Dubbo: http://alibaba.github.io/dubbo-doc-static/User+Guide-zh.htm http://alibaba.github.io/dubbo-doc-static/Developer+Guide-zh.htm
4. Rational Edge: SOA 质量管理在 SOA 服务生命周期管理中的角色 http://www.ibm.com/developerworks/cn/rational/rationaledge/content/apr07/mcbride/
5. Venus http://wiki.hexnova.com/display/Venus/HOME
6. Gaea https://github.com/58code/Gaea

无