内网负载均衡容错系统
L5设计方案

■ 拟定人：	王先明	■ 审批人：	
■ 发布范围：	内部使用	■ 发布日期：	2015 0803 19:30

■ 版本变更记录

时间	版本	说明	修改人
2015-07-31	V0.1	初稿（2015 1210 14:30起草）	王先明
2015-08-03	V0.2	修订	王先明
2015-08-05	V0.3	修订	王先明

目录

1. 项目介绍

1.1. 背景介绍

典型的互联网分布式系统，几乎每个层次都具备了横向伸缩能力。但分布式系统需要关注以下两个问题：

1) **每个层次的各个服务节点的负载均衡问题。**如果不均衡，在负载上升的时容易导致繁忙节点服务质量下降；
2) **服务节点的临时故障问题。**如何及时探测、屏蔽和恢复服务节点，若不及时处理这些服务节点也会造成服务质量下降。

不同的系统也许有不同的架构和算法来解决这两个问题，不管哪种实现方式都有一些局限性（略）。假如**能够抽象出一个较为通用且高效易用的系统，解决大部分分布式系统的这两个问题，这样避免了各个业务系统各自研发代价，也能集中开发、运营资源，提高业务开发效率和运营质量，降低运维复杂度。**

1.2. 基本原理

L5负载均衡容错系统，主要为了解决以上两个问题而出现基础服务组件，尽量提高业务服务质量，理想目标是达到99.999%级别（即：5个9，意味着该系统运行1年的故障影响只能小于5分钟）。

一个典型的三层分布式系统以及L5在架构中的位置，如下图1所示：

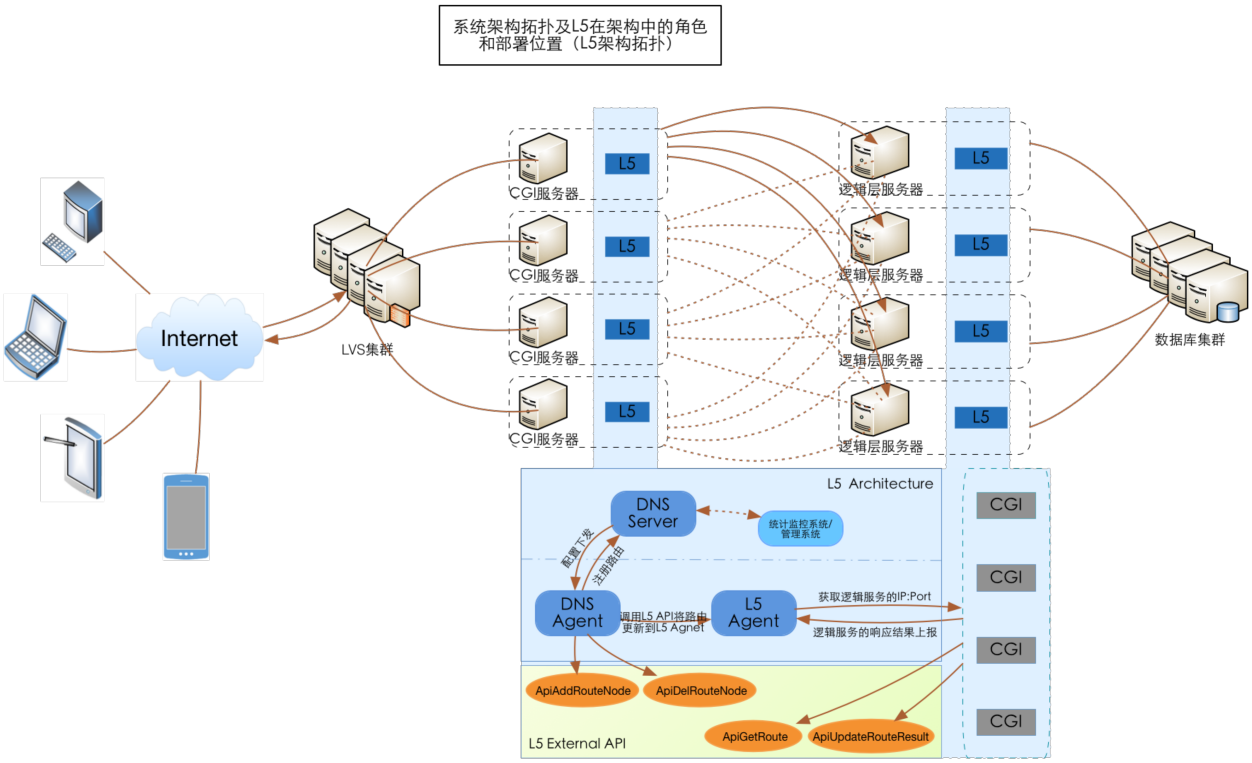

图1

对于任何层次需要进行网络远程调用的节点，都可以使用L5带来的负载均衡和容错服务，典型的如接入层服务器访问逻辑层服务器，或者逻辑层服务器访问存储层服务器，如图1. 如果接入层机器为主调机器，那么逻辑层机器为被调机器；如果逻辑层为主调机器，那么存储层就为被调机器。

L5系统调用关系，如下图2所示：

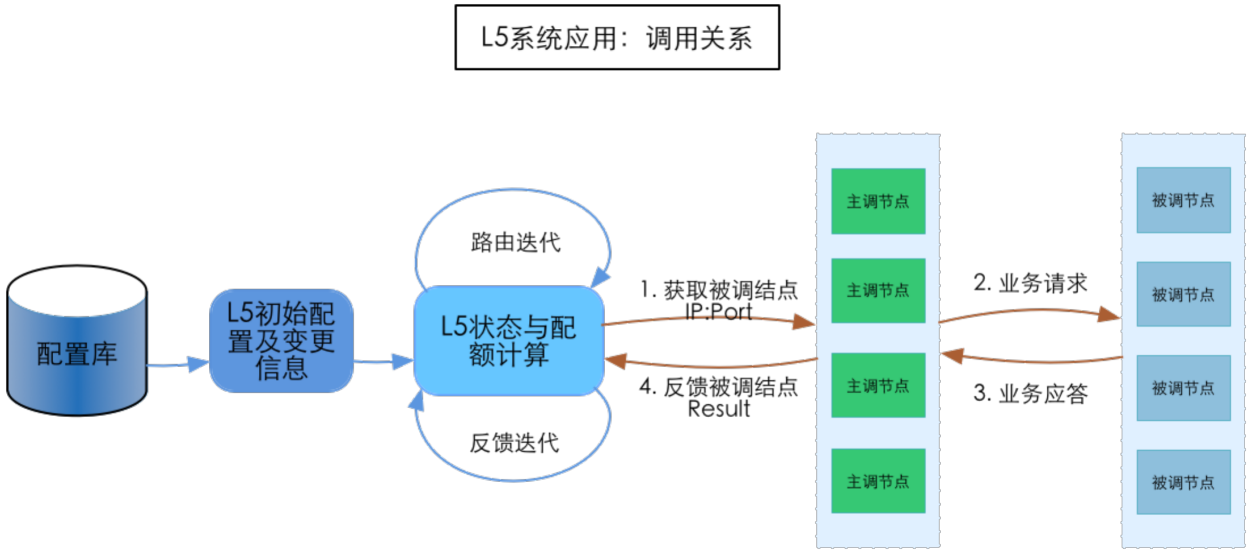

图2

L5**基本工作原理**可以抽象为：

1) 基于机器初始配置信息，通过自适应算法；

2) 以两个关键指标：a. 请求成功率 和 b. 请求成功时延 为依据，周期性计算出每个被调机器的权重，再使用高效的配额算法分配各个被调业务进程的访问路由，主调业务进程通过API来取得这些路由，调用结束时通过API来反馈路由的好与坏。

L5**基本功能特征**：

1) **名字服务：**以SID为关键字，通过SID取得真正的IP和Port，IP和Port配置对调用者透明，集中进行配置管理，运维变更配置更方便；说明：SID 即本文档的Service ID：由模块ID（ Module ID ）和命令字ID（ Command ID ）组成。

2) **负载均衡：**以两个关键指标：a. 请求成功率 和 b. 请求成功时延 进行动态权重计算，动态均衡各个被调服务器的负载，达到较好的整体服务质量；

3) **预防出错：**通过对后端服务状态的监控，收集各类重要指标，及时发现风险区，并根据业务重要性对后端服务访问进行合理规划，预防出现进程单点，机器单点，路径单点等故障；

4) **故障容错：**迅速自动屏蔽错误率高或有故障的机器节点，对故障点进行告警和通知。并进行适时探测，待故障恢复后自动恢复，实现对单台机器或者整个模块机器的过载保护能力，防止雪崩现象。

L5本质上是一个**路由决策系统**，不包含远程过程调用，与业务系统完全没有耦合，不参与到请求的实际处理数据流中，不作为业务系统的关键路径。L5本身从API到客户端Agent都有缓存，所以即使整个L5后端挂了，也不会对业务有什么影响。正因为这样，大量的业务系统可以共用L5系统，唯一需要改动的是在业务代码中对L5 API进行一些必要的调用。

1.3. 对比LVS

目前公司业务在升级改造浪潮之中，内网的负载均衡和故障容错成为主要问题之一， 那有人或许疑问：“为何不用LVS？”，实际上LVS等优秀开源的组件做负载均衡和容错也是可以的，而且运维复杂度更低，但是事物总有两面性，以下是L5与LVS负载均衡/容错功能比较，业务可以根据需要选择合适的方案。

	LVS	L5
功能与应用	外网IP收敛，内外网负载均衡和容错。	内网负载均衡和容错。
是否关键路径	构成业务请求的关键路径，增加一次网络通讯和时延，增加若干次上下文切换，故障对业务是致命的（故障不可接受）。	本质是决策系统，不构成业务请求关键路径，故障几乎无影响。
使用成本	对机房和网络有一定的要求，属于比较稀缺资源。	内网的任何地方可以使用，几乎零成本。
运行方式	以VIP（Virtual IP）方式提供，对业务透明。	需要业务修改代码调用VIP，多两次本地调用。
均衡和容错能力	基于传输层，算法比较简单。	基于业务应用层反馈，可靠性更高；算法更智能，可个性化定制。
申请方式	可能需暴露到外网，一般需要审核接入。	完全自助申请接入，自助变更。
附加功能		具备详细的成功率和时延的统计分析数据和模块间的调用关系。

2. 设计方案

2.1. 系统结构

L5容错系统对后端服务集群提供容错访问（负载均衡和过载保护），L5系统整个架构比较简单清晰，由DNS Server，DNS Agent和L5 Agent三部分组成。

L5实际数据流，如下图3所示：

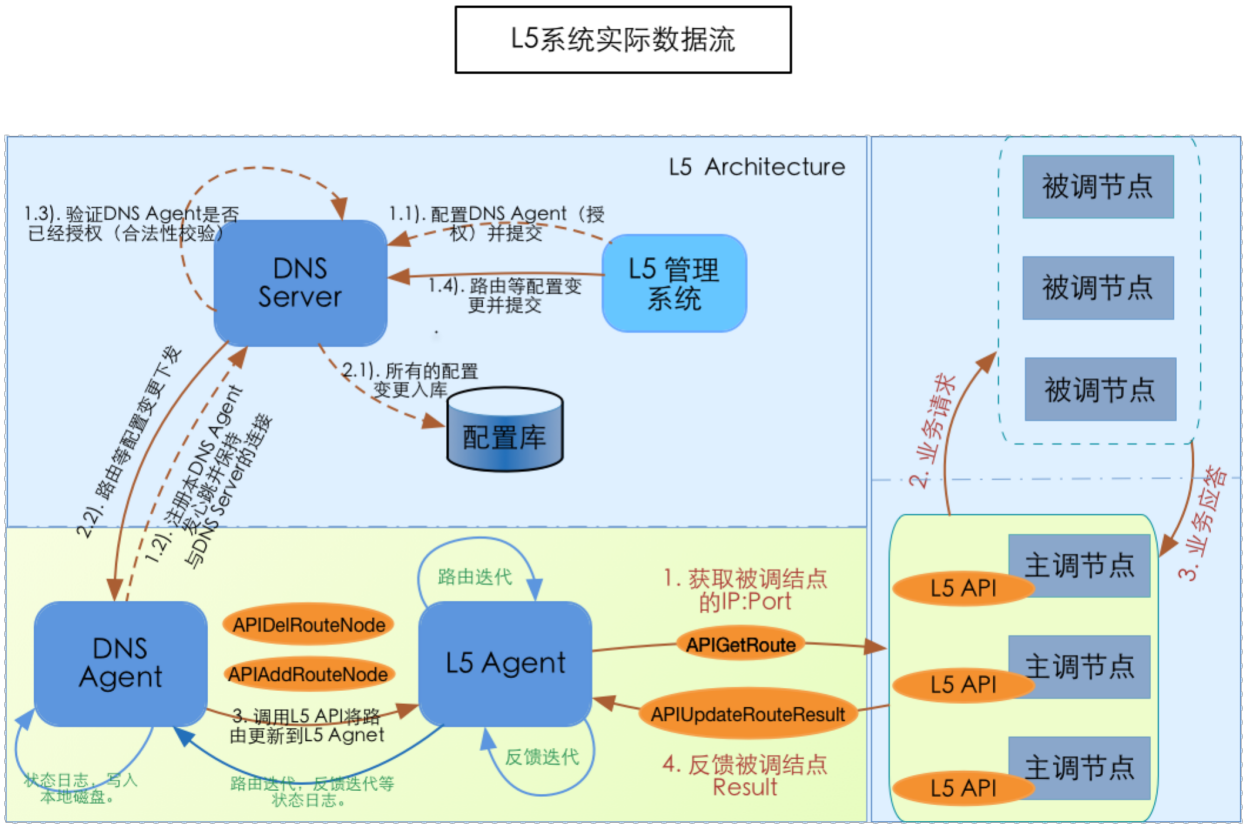

图3

相关**模块职能**说明：

1) DNS Server负责下发配置：a. 无状态路由配置（如无特别说明，都属这类配置）；b. 有状态路由配置（暂不实现）。

2) DNS Agent接收来自DNS Server的配置，保证 L5 Agent获取到完整的最新的路由配置信息。通过调用L5 API（APIAddRouteNode和APIDelRouteNode）将路由配置信息更新到L5 Agent。

3) L5 Agent将后端Server的IP/Port分配给业务CGI，并根据业务CGI上报的路由响应结果来决定下一个周期的路由分配。 说明：业务CGI对后端服务进行业务请求，首先，通过调用L5 API（APIGetRoute）从L5 Agent获取后端服务的IP/Port；其次，直接对该 IP/Port上的后端服务进行业务请求，最后，通过调用L5 API（APIUpdateRouteResult）将本次后端服务应答的成功与否，成功时延信息告知L5 Agent。

相关**模块部署**说明：

1) DNS Server部署在一个机器上，可根据需要进heartbeat，以防DNS Server或所在机器故障。

2) DNS Agent部署在业务机器上，并且部署在L5 Agent同一个目录，在启动L5 Agent之后，再启动DNS Agent。

2.2. 过载保护

L5使用时间片内（一般为1分钟）的访问作为统计单位，时间片内所有访问的a. 请求成功率 和 b. 平均请求成功延时 等信息作为下个时间片内请求的参照。 L5过载保护技术有以下假设条件：

1) 收集x时间片内后端服务响应结果成功，延时信息判断后端服务x+1时间片对前端请求的处理质量，是否适合继续服务。
2) 大部分服务质量的下降跟当前时刻处理压力有关系，当服务响应结果失败增大时，降低访问量，有助于它进行正确处理。
3) 根据服务前一个时刻处理结果的成功率增加下一个时刻对其的请求，能减少服务发生故障的可能性。
4) 当服务有故障时服务失败比率很高，对其的访问量按上个周期的失败率调整，如果周期设置合适，能够迅速检测到故障服务，并对其仅发象征性请求。

2.3. 核心算法

L5主要核心算法：

负载均衡算法：以请求成功率和请求延时构成的动态权重，结合预先配置的静态权重，最后构成了被调节点的调度权重，然后以周期性建立的分配模型分配给各个主调节点；另外还需要考虑机器伸缩的时候，负载的变化需要平滑过度的问题。

门限收缩算法：考虑到系统运行是一个动态过程，任何静态的最大阀值最小阀值限制都是不科学的，所以需要根据延时和成功率情况对阀值进行动态伸缩，这样才能更好的契合业务的需要。

宕机探测算法：由于机器临时性故障难免，通过成功率的逐渐下降可以预知宕机风险，并及时排除；另外需要以极低的代价，及时探测被调节点，并自动恢复负载。

3. 实现要点

3.1. DNS Server模块功能点

1) 收到新的 DNS Agent 发消息（注册）时，判定是否在配置DNS Agent列表中，如果是，则允许该IP/Port的DNS Agent接入。
2) 定时遍历在 DNS Agent列表中的DNS Agent，a. 超时没有收到DNS Agent发送heartbeat进行告警；b. 收到DNS Agent发送的heartbeat时，回应heartbeat ack。
3) 提供DNS Agent 获取数据的接口，a. 初始化拉取pull所有DNS Agent 关注的配置；b. 在DNS Agent 关注的配置范围内，允许DNS Agent拉取pull或增量拉取pull指定某些数据节点的数据的请求，并给出正确的应答。
4) 收到配置管理系统的路由配置变更通知（如：业务列表、后端服务地址发生变化）Push包时，Push 相关变更信息到 需关心该变更通知的 DNS Agent，对Push失败的 DNS Agent 进行告警。
5) 所有的操作记录，以日志的形式记录到本地。
6) **设计难点**：a. 管理系统（含权限）的设计（暂不实现）；b. 增量通知的相关（如:变更时事务粒度的控制）；c. 重启后对增量状态的恢复。

3.2. DNS Agent模块功能点

1) 向已知的DNS Server 发消息（注册），并对注册失败进行告警。
2) 定时向DNS Server发送heartbeat包，并对超时没收到DNS Server 对 heartbeat ack包进行告警。
3) 提供获取DNS Server数据的请求接口：a. 初始化拉取pull所有DNS Agent 关注的配置；b. 在DNS Agent 关注的配置范围内，允许DNS Agent拉取pull或增量拉取pull指定某些数据节点的数据的请求。并把DNS Server应答的数据及时同步给L5 Agent，DNS Agent 通过共享内存队列与L5 Agent 通讯。
4) 收到DNS Server路由配置变更通知（如：业务列表、后端服务地址发生变化）Push包时，把Push的数据调用L5 API（APIAddRouteNode和APIDelRouteNode）及时同步给L5 Agent。对同步给L5 Agent 的失败信息，进行告警，并重试。
5) DNS Agent接收来自DNS Server的配置，保证 L5 Agent获取到最新、最完整的路由信息。通过调用L5 API（APIAddRouteNode和APIDelRouteNode）将路由更新到L5 Agent。
6) 记录从L5 Agent 产生的状态日志（比如：路由迭代，反馈迭代）输出到本地磁盘保存，L5 Agent将这些日志通过共享内存通道递交给DNS Agent。
7) 所有的操作记录，以日志的形式记录到本地。
8) **设计难点**：a. 重启后对增量状态的恢复；b. 大量的日志写带来的一些问题。

3.3. L5 Agent模块功能点

1) L5 Agent将后端Server的IP/Port分配给业务CGI，并根据业务CGI上报的路由响应结果来决定下一个周期的路由分配。
2) 业务CGI对后端服务进行业务请求，首先调用L5 API（APIGetRoute）从L5 Agent获取后端服务的IP/Port，接着对该 IP/Port后端服务进行业务请求，得到业务应答后，调用L5 API（APIUpdateRouteResult）将访问结果、时延信息告知L5 Agent。
3) L5 Agent 产生的状态日志（比如：路由迭代，反馈迭代），通过共享内存通道递交给DNS Agent，由后者落地存盘。
4) 所有的操作记录，以日志的形式记录到本地。
5) 路由分配算法（另外讨论）。
6) **设计难点**：a. 路由分配算法的准确，稳定，高效，智能；b. 重启后对增量状态的恢复。

3.4. 主要数据结构

此处未给出详细的数据结构（按关系数据库MySQL来定，至少有15张表），考虑到存储的选择，如：一些数据结构类似层次目录结构，用树形存储，那更为方便，简洁。目前只是列出主要的数据结构和数据元素，供大家参考，开发人员可自行发挥，比如更换存储类型，自行添加必须的元素等。

MachineNode（机器IP节点）主要数据

Field	Type	Key	Default	Extra
machine_id	int(10) unsigned	PRI	0	机器 id，注册机器IP（一一对应）时，可通过auto_increment 生成。
machine_ip	varchar(255)	UNI	‘0’	机器IP，唯一。
modify_time	int(10) unsigned		0	更新时间

ModuleNode（后端模块）主要数据

Field	Type	Key	Default	Extra
moudle_id	int(10) unsigned	PRI	0	模块id，注册模块name（一一对应）时，可通过 auto_increment 生成。值域范围*0～65535*
moudle_name	varchar(512)	UNI	‘0’	模块name，唯一，如：order（订单模块）。
modify_time	int(10) unsigned		0	更新时间

ServiceNode（后端服务）主要数据

Field	Type	Key	Default	Extra
service_id	int(10) unsigned	PRI	0	service id，注册module_id & command_id（一一对应）时，可通过auto_increment 生成。建议通过***module_id<<16+command_id***生成
module_id	int(10) unsigned	UNI	0	业务模块Id，来自***Module***列表。
command_id	int(10) unsigned			业务命令字id（协议号），和模块Id 一起，用于标示SID。值域范围***0～65535***
service_name	varchar(512)	UNI	‘0’	service name，可通过模块名+command id所对应的command name，如：order_getorderlist
modify_time	int(10) unsigned		0	更新时间

ServiceDeployNode（后端部署）主要数据

Field	Type	Key	Default	Extra
service_id	int(10) unsigned		0	service Id，来自***Service***列表。
~~module_id~~	~~int(10) unsigned~~		~~0~~	~~业务模块id，来自***Module***列表。~~
~~command_id~~	~~int(10) unsigned~~			~~业务命令字id（协议号）~~
machine_id	int(10) unsigned	PRI	0	service所部署的机器id，来自***Machine***列表。
~~machine_ip~~	~~varchar(255)~~		~~'0'~~	~~service所部署的机器IP，来自***Machine***列表。~~
machine_port	int(10) unsigned		0	service 所部署的机器Port
~~environment_id~~	~~int(10) unsigned~~		~~0~~	1. ~~开发环境~~ 2. ~~测试环境~~
quota	int(10) unsigned		100	配额 权重 请求上限
failure	int(10) unsigned		0	失败率（百分比值）
~~state~~	~~int(10) unsigned~~		~~0~~	1. ~~create~~ 2. ~~modify~~ 3. ~~delete~~
~~switch~~	~~int(10) unsigned~~		~~0~~	1. ~~enable~~ 2. ~~disnable~~
modify_time	int(10) unsigned		0	更新时间

AgentDeployJoinServiceNode（关注的路由配置）主要数据

Field	Type	Key	Default	Extra
machine_id	int(10) unsigned	PRI	0	agent 所部署的机器Id，来自*Machine*列表。
~~machine_ip~~	~~varchar(255)~~		~~'0'~~	~~agent 所部署的机器IP，来自*Machine*列表。~~
machine_port	int(10) unsigned		0	agent 所部署的机器Port
service_id	int(10) unsigned		0	service Id，来自*Service*列表。
~~module_id~~	~~int(10) unsigned~~		~~0~~	~~业务模块号，来自*Module*列表。~~
~~command_id~~	~~int(10) unsigned~~			~~业务命令字id（协议号）~~
modify_time	int(10) unsigned		0	更新时间

ClientJoinServiceNode（Client接入）主要数据

Field	Type	Key	Default	Extra
service_id	int(10) unsigned	UNI	0	service Id，来自*Service*列表。
~~module_id~~	~~int(10) unsigned~~		~~0~~	~~业务模块id，来自*Module*列表。~~
~~command_id~~	~~int(10) unsigned~~		~~0~~	~~业务命令字id（协议号）~~
machine_id	int(10) unsigned		0	client id，来自*Machine*列表。
~~machine_ip~~	~~varchar(255)~~		~~'0'~~	~~client IP，来自*Machine*列表。~~
machine_port	int(10) unsigned		0	client 访问Port
modify_time	int(10) unsigned		0	更新时间

4. 致谢

感谢 张善祥 的支持并提供L5的需求（为本文档原型），感谢 刘黄乐 积极提问，使我不断的巩固，总结，完善我的知识体系并最终形成本文档。