找出系统性能瓶颈：企业级系统性能分析实践

7点测试：Zee

整体目录

理解性能测试

性能测试需求的获取和分析

性能测试执行、控制及分析

可用性统计分析

排队论在分析过程中的应用

性能问题实例

分析一下

并发数	并发时间	平均响应时间 Sec	平均每秒请求数	平均每秒通过的事务数	总错误率	吞吐量 Bytes/sec	CPU 占用%	平均每秒 IO
100	15min	0.074	1283.796	1284.018	0	444184.507	20	49
150	15min	0.116	1244.675	1245.007	0	430642.781	20	47
200	15min	0.157	1233.893	1234.334	0	426908.195	20	47
400	15min	0.318	1239.356	1240.232	0	428780.257	19	44
600	15min	0.487	1208.232	1209.541	0	418048.369	21	46
1000	15min	0.832	1076.145	1077.955	0	372172.537	18	41
1500	15min	1.568	865.551	868.347	0	299448.457	14	21
2000	15min	2.146	822.45	826.009	0	284535.211	13.4	20

频繁GC的现象

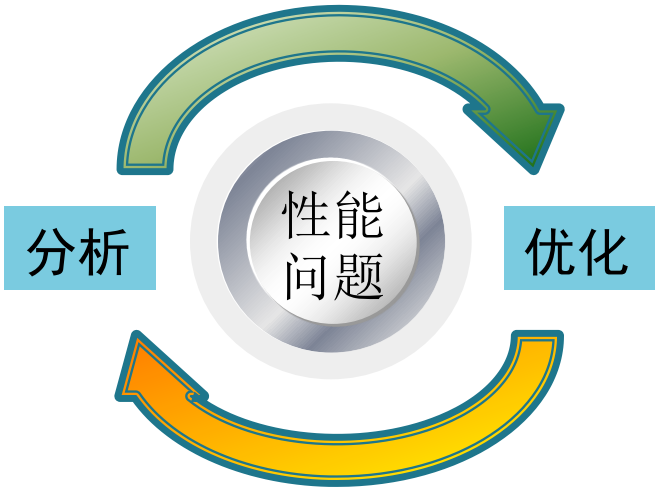
```
1211.963: [GC 1447495K->1351861K(1560768K), 0.0925580 secs]
1212.860: [GC 1448629K->1352979K(1560768K), 0.0574980 secs]
1213.611: [GC 1449747K->1358990K(1560768K), 0.1141340 secs]
1214.507: [Full GC 1455758K->1358140K(1560768K), 13.8713880 secs]
1229.223: [Full GC 1454908K->1360125K(1560768K), 13.7175530 secs]
1243.656: [Full GC 1456893K->1361787K(1560768K), 13.7245990 secs]
1258.030: [Full GC 1458555K->1363511K(1560768K), 18.6801770 secs]
1277.503: [GC 1450279K->1355897K(1560768K), 0.0395160 secs]
1278.322: [Full GC 1452665K->1355949K(1560768K), 13.8096370 secs]
1292.890: [Full GC 1452717K->1363137K(1560768K), 13.8575540 secs]
1307.515: [Full GC 1459905K->1364913K(1560768K), 13.8494200 secs]
1322.107: [Full GC 1461681K->1361004K(1560768K), 19.1481170 secs]
1342.010: [Full GC 1457772K->1362670K(1560768K), 13.9276840 secs]
1356.709: [Full GC 1459438K->1369881K(1560768K), 13.9454060 secs]
1371.413: [Full GC 1466645K->1371542K(1560768K), 13.9195540 secs]
```

理解模型

理解性能测试的目标

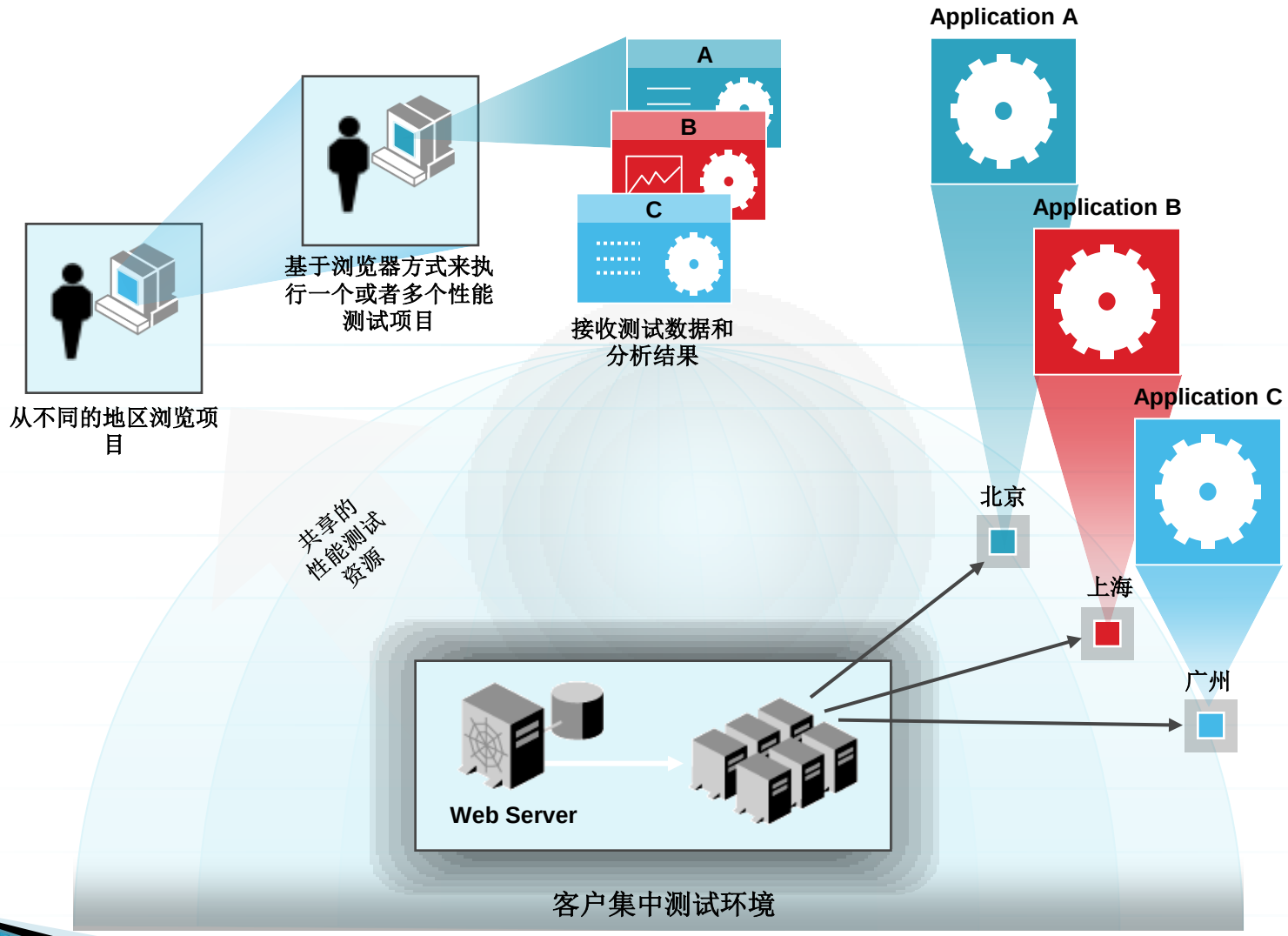

评估系统在性能方面抗风险能力
• 连续性风险
• 长时间
• 多用户
性能测试
• 峰值
• 最大并发量
• 系统性能拐点
满意的用户？
不满意的用户？

理解性能分析

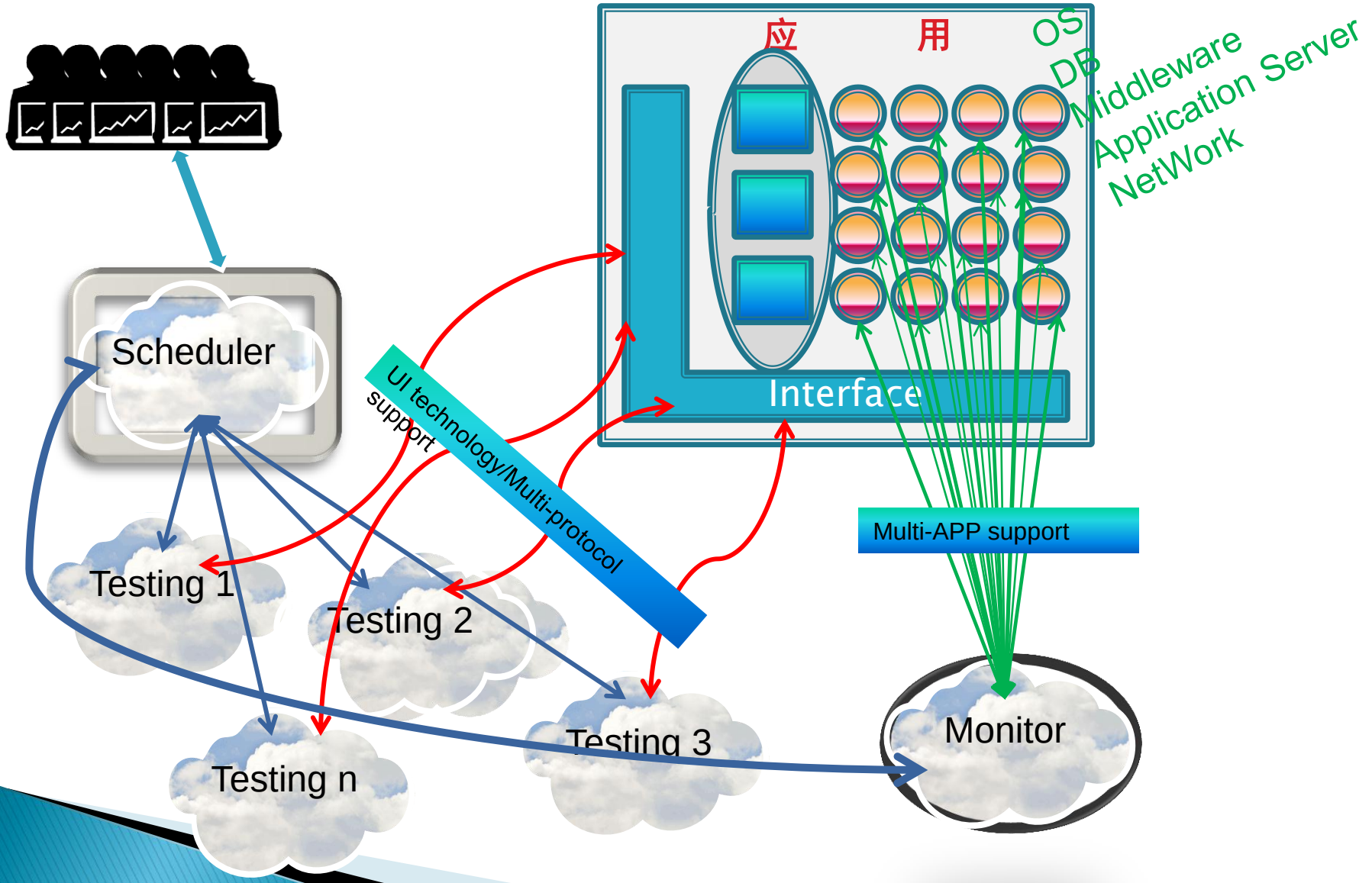


它可以是时间，可以是资料，
也可以是业务！

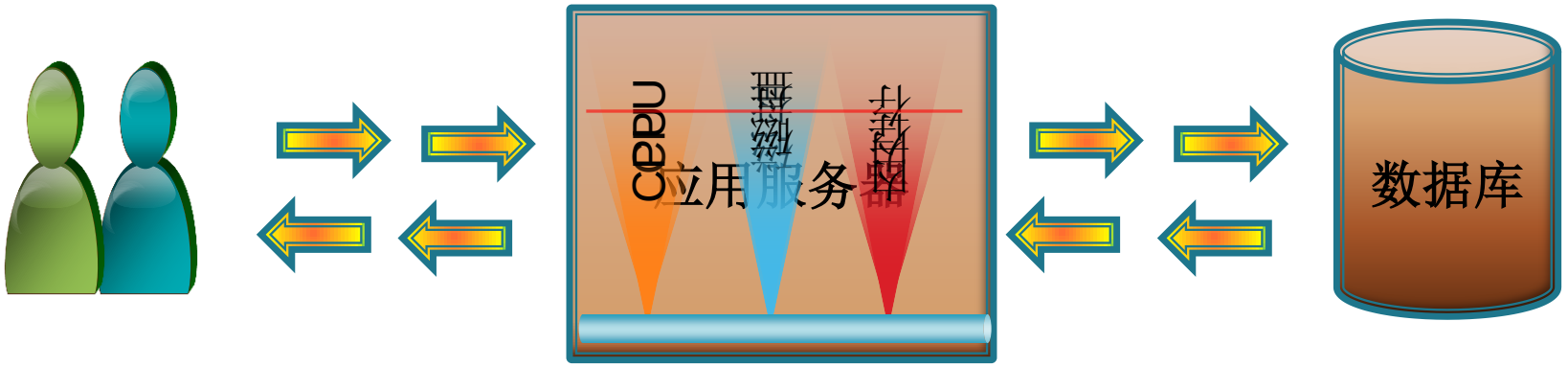
性能实施和控制


Application A
A
B
C
基于浏览器方式来执行一个或者多个性能测试项目
接收测试数据和分析结果
Application B
Application C
从不同的地区浏览项目
共享的性能测试资源
北京
上海
广州
Web Server
客户集中测试环境

理解性能监控


应
用
OS
DB
Middleware
Application Server
NetWork
Interface
Scheduler
UI technology/Multi-protocol support
Multi-APP support
Testing 1
Testing 2
Testing 3
Testing n
Monitor

性能分析路径



整体目录

理解性能测试

性能测试需求的获取和分析

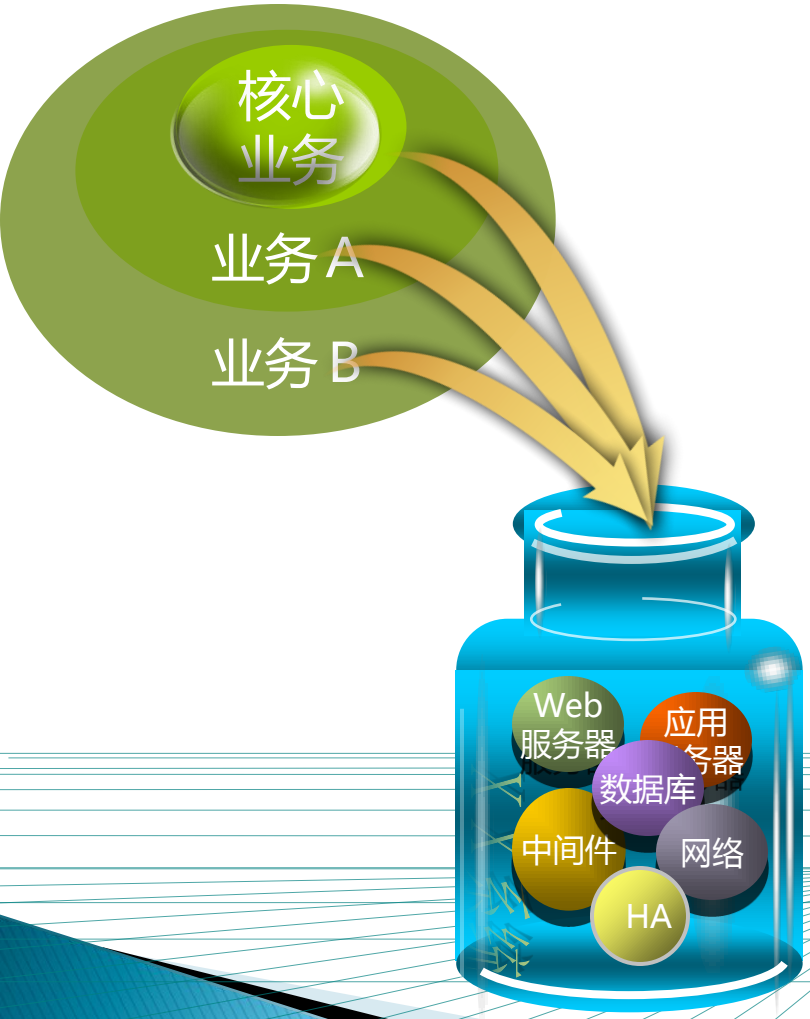
性能测试执行、控制及分析

可用性统计分析

排队论在分析过程中的应用

性能问题实例

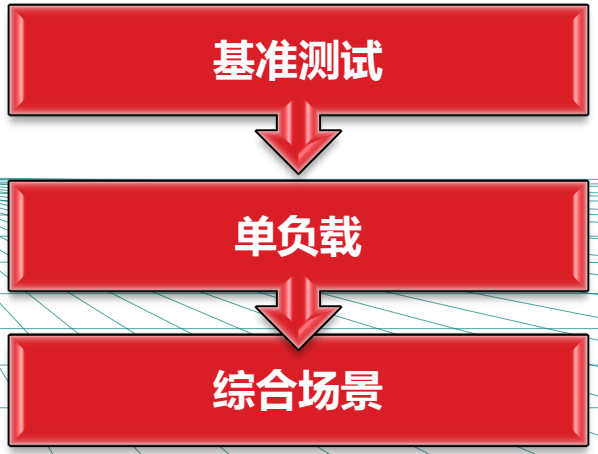
容量测试目标



说明

容量测试关注的性能指标

- ◆最大处理能力（TPS）
- ◆成功率
- ◆并发用户
- ◆在线用户
- ◆每秒点击率（HPS）
- ◆响应时间
- ◆吞吐量
- ◆资源阀值



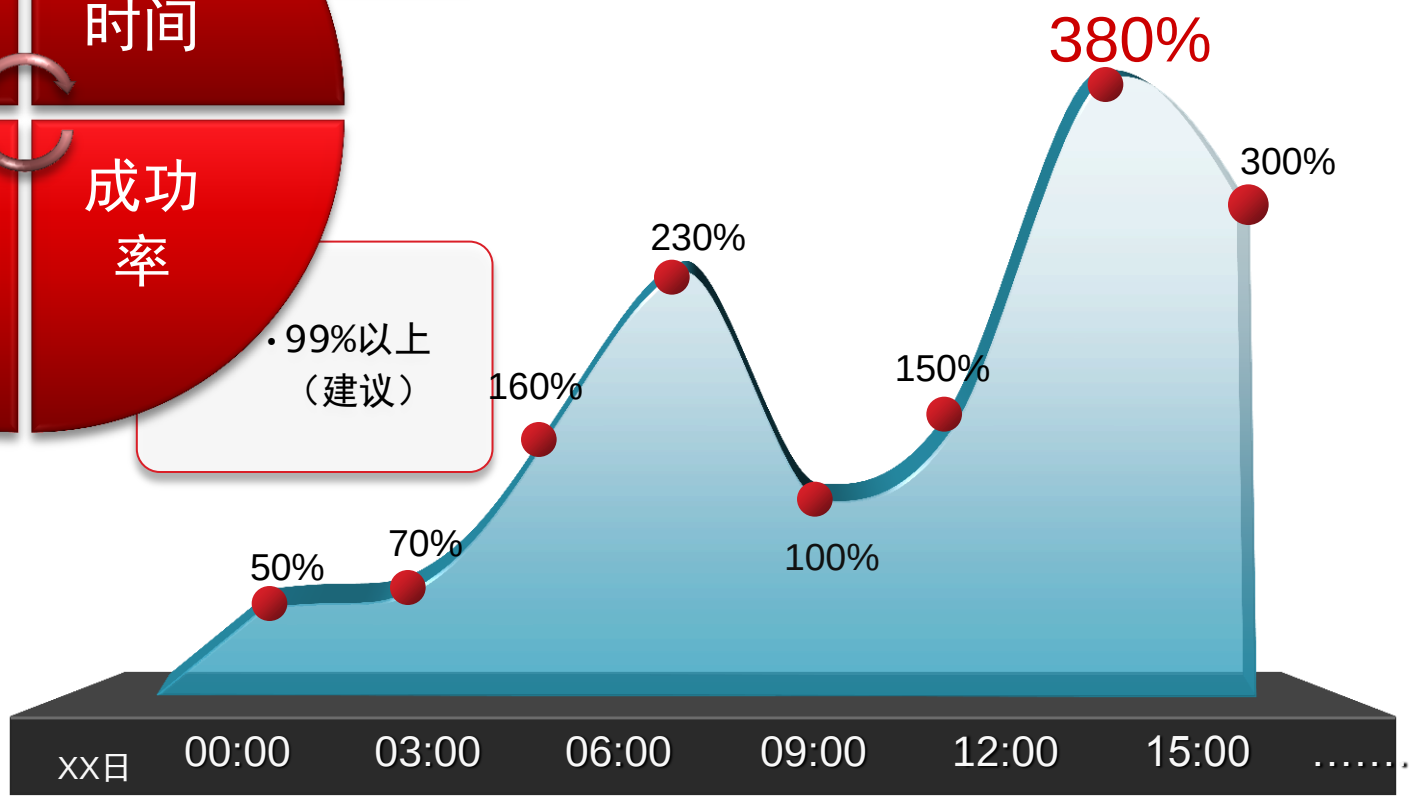
稳定性能力评估目标

- 压力的大小
- 业务配比
- 多场景模拟

- 系统运行特性
- 用户总量

- 各类资源稳定使用
- 合理释放资源

- 99%以上（建议）



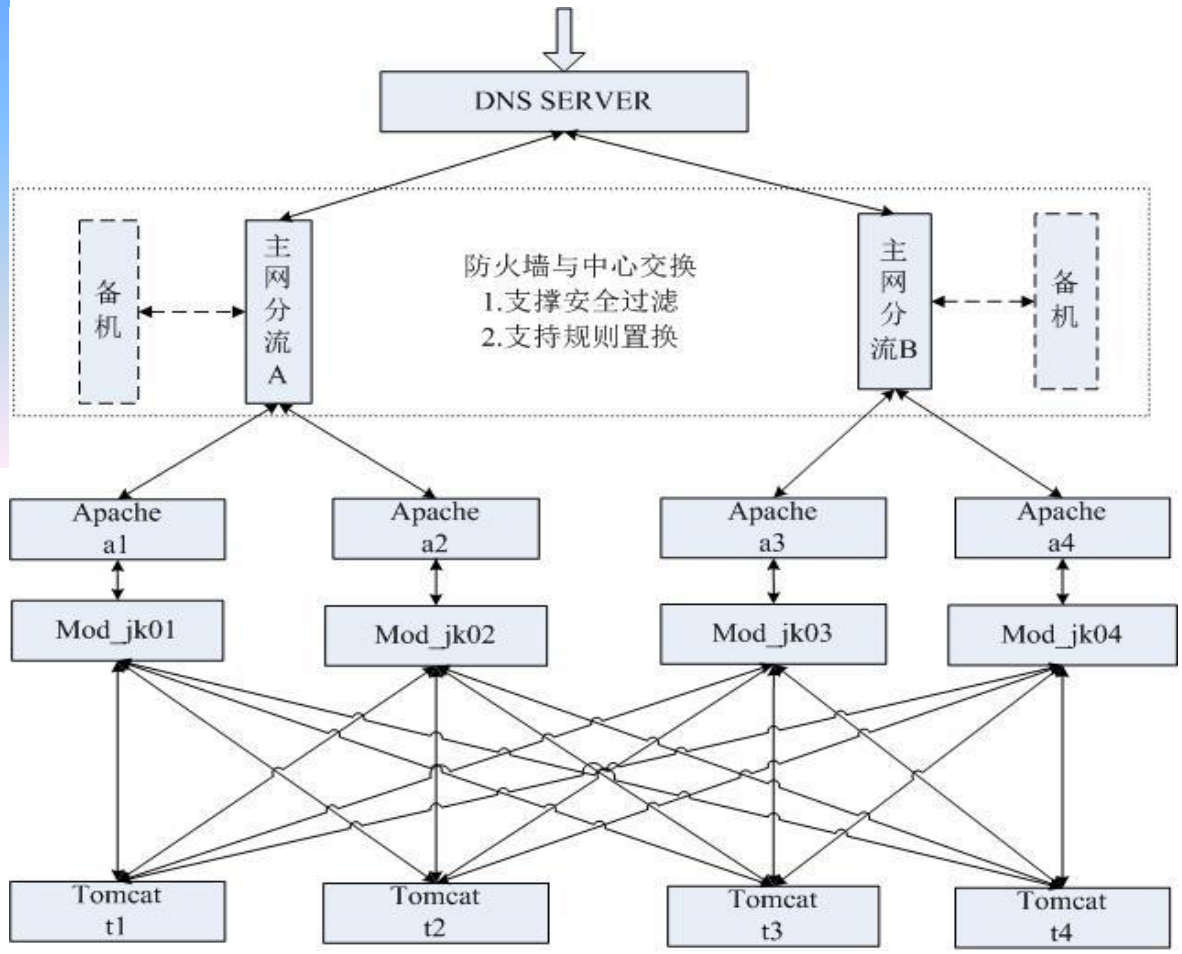
测试目的

来源：
1.业务人员；
2.开发人员；
3.客户需求；

内容：
1.描述环境（软硬件、条件）；
2.描述业务；
3.描述性能指标；
4.描述数据；
5.描述测试策略；

测试范围

- 项目背景
- 系统架构说明
- 系统拓扑说明
- 测试范围说明
- 交易路径描述
- 需要测试的特性
- 不需要测试的特性



业务调研

系统信息调研

- 系统类型：系统的基本特性，如交易处理型系统、数据处理型系统等
- 架构部署：系统的整体架构、服务器部署方式
- 技术信息：系统运行平台、数据库产品、使用的中间件、协议及通讯方式等
- 业务信息：支持的业务类型、业务范围与功能、与其它系统的业务关系等

业务信息调研

- 基本业务与功能：系统的基本业务概念以及系统的业务种类与具体功能
- 关键业务逻辑与处理流程：关键业务的业务流程、交易路径、交易数据、交易流程与时序图
- 交交易列表：调查业务系统全部交易清单，了解交易的组合关系、执行顺序等
- 交易量信息：在不同时间粒度下统计单个交易处理量以及总交易量信息
- 业务目标/业务拓展计划：目前的生产业务量和用户数以及系统预期业务目标和本 次测试预期业务指标

文档资料调研

- 功能规格说明书
- 系统设计文档
- 生产运营统计
- 前期系统测试资料

测试环境调研

- 架构层次
- CPU
- 内存
- 磁盘
- 网络
- 操作系统
- 应用服务器

架构层次	服务器名称	服务器功能描述	CPU数量	内存大小（GB）	主机类型	操作系统
Web层	Web	接入服务器	2	4	Linux	RHEL 4U6
应用层	AP1	XX系统性能测试	4	8	HP PA-RISC	HP-UX 11.11
	AP2	XX系统性能测试	4	8	HP PA-RISC	HP-UX 11.11
数据库层	DB1	在线数据库	4	8	IBM	AIX 5304
	DB2	在线数据库	4	8	IBM	AIX 5304

测试指标及测试数据

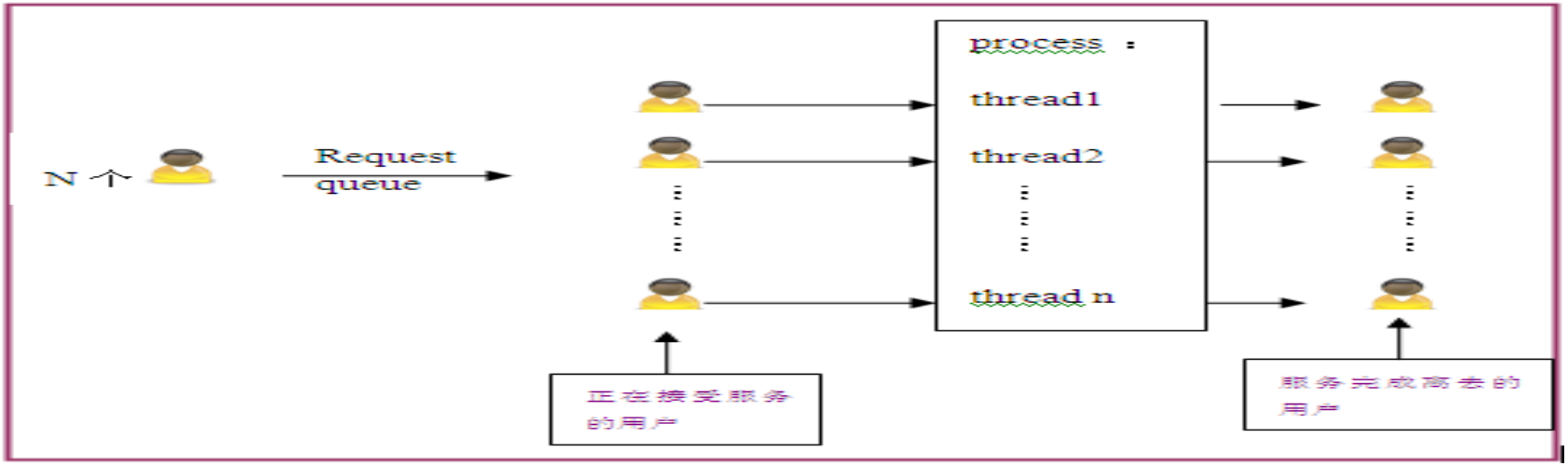
测试指标：

交易处理能力TPS（笔/秒）：500
平均响应时间：webservice交易<2秒；报表<5秒
系统资源使用率：<75%
失败率（不包括返回错误的正常交易）：<1%

测试数据：

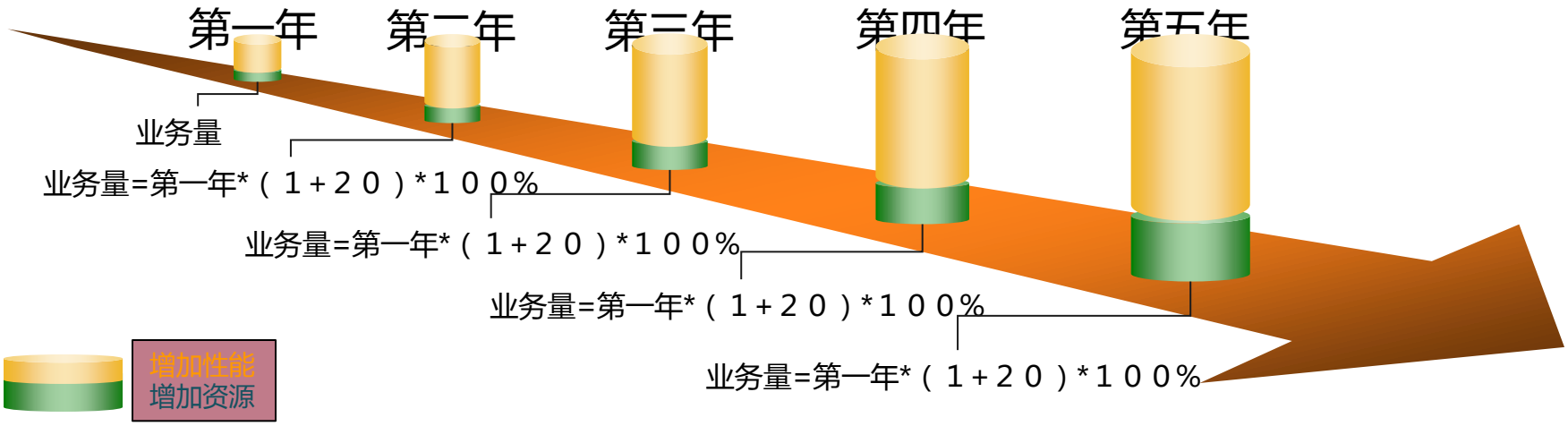
数据类型
数据量
表名称
表关系

并发用户数



并发用户数指在同一时刻内，登录系统并进行业务操作的用户数量。
并发用户数对于长连接系统来说最大并发用户数即是系统的并发接入能力。对于短连接系统而言最大并发用户数并不等于系统的并发接入能力，而是与系统架构、系统处理能力等各种情况相关。

系统可扩展性指标



计算公式为：（增加性能/原始性能）/（增加资源/原始资源）*100%。

扩展能力应通过多轮测试获得扩展指标的变化趋势。
一般扩展能力好的系统，扩展指标应是线性或接近线性的，且扩展能力应该在70%以上。

整体目录

理解性能测试

性能测试需求的获取和分析

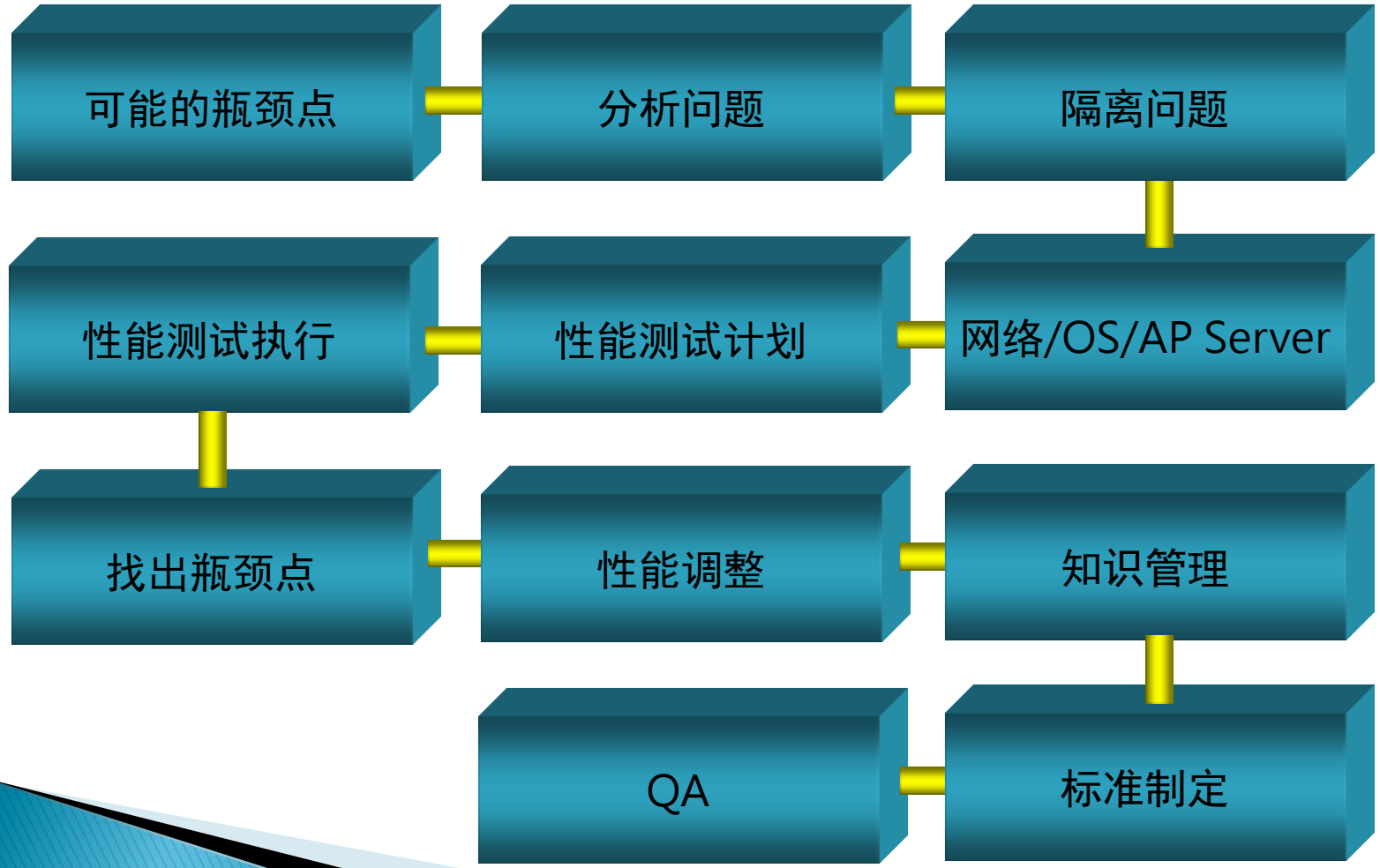
性能测试执行、控制及分析

可用性统计分析

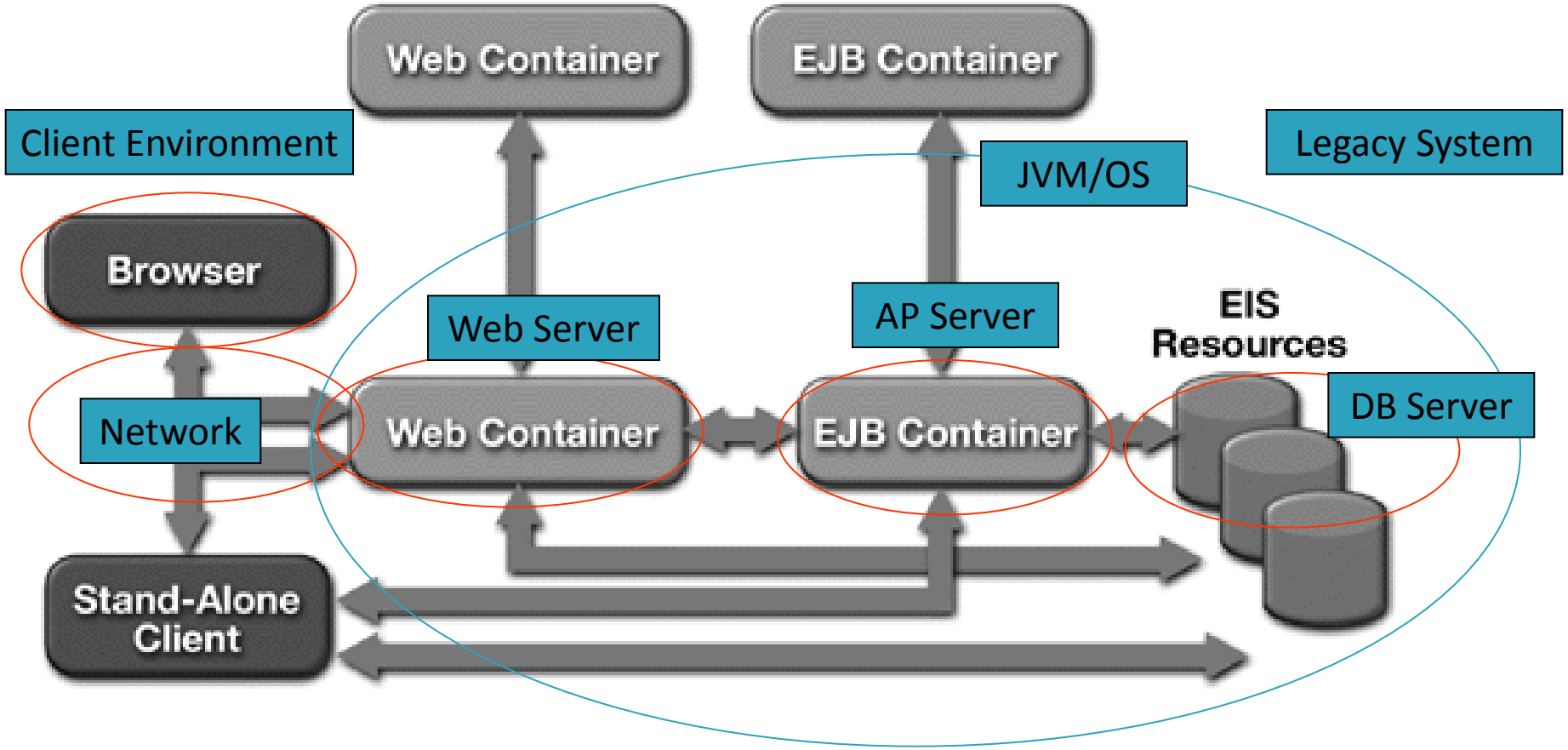
排队论在分析过程中的应用

性能问题实例

性能问题分析流程


可能的瓶颈点
分析问题
隔离问题
网络/OS/AP Server
性能测试计划
性能测试执行
找出瓶颈点
性能调整
知识管理
标准制定
QA

从系统架构分析瓶颈点



Client Environment/Networking/Web or AP Server/EJB/DB Server/OS/JVM/Legacy System

系统故障征兆

- **故障征兆**

- ■ 持续运行缓慢。时常发现应用程序运行缓慢。通过改变环境因子（如负载量、数据库连接数等）也无法有效提升整体响应时间。
- ■ 系统性能随时间的增加逐渐下降。在负载稳定的情况下，系统运行时间越长速度越慢。可能是由于超出某个阈值范围，系统运行频繁出错从而导致系统死锁或崩溃。
- ■ 系统性能随负载的增加逐渐下降。随着用户数目的增多，应用程序的运行越发缓慢。若干个用户退出系统后，应用程序便能够恢复正常运行状态。
- ■ 间发性的系统挂起或异常错误。有时候可能受负载或其它因素影响，当面不能完全显示，又或者是追踪到异常和堆栈的错误页，用户此时会看到系统挂起的情况。系统挂起的次数可能会稍有不同，然而即便是经过预烧（“burn in”）试验也无法完全消除。
- ■ 可预见的死锁。系统挂起或系统出错发生的频率随着时间的推移明显增多，直到系统完全死锁。通常借助自动重启的故障管理模式解决上述问题。
- ■ 系统突然出现混乱。系统正常运行，且在或多或少的一段时间内（譬如，可以是1小时也可以是3天）拥有一致优越的运行性能，却突然毫无征兆地出错甚至死锁。

系统常见问题及解决方法

“疾病”	描述	征兆	原因或解决方法
内存泄漏呈线性增长	各单元（如每个事务或每个用户等）的内存泄漏，导致内存使用率随时间或负载的增加呈线性增长。系统性能随时间或负载的增加大幅下降，重启后系统可恢复正常	系统性能随时间的增加逐渐下降, 系统性能随负载的增加逐渐下降	虽然有许多外在因素存在（如各单元数据的链表存储、尚未回收的缓冲中的回收或增长操作等），但最为常见的原因还是资源泄漏
内存泄漏呈指数级增长	内存泄漏呈双倍增长，导致系统内存消耗随时间呈指数曲线变化	系统性能随时间的增加逐渐下降,系统性能随负载的增加逐渐下降	虽然有许多外在因素存在（如各单元数据的链表存储、尚未回收的缓冲中的回收或增长操作等），但最为常见的原因还是资源泄漏
导致无限循环的编码缺陷	线程在while语句返回值为真的情况下发生阻塞， 将最终演变成为CPU密集型和等待密集型或螺旋等待变量	可预见的死锁	需要进行侵入式循环切除
资源泄漏	JDBC语句、CICS事务网关连接等出现资源泄漏，引发Java桥接层和后台系统出现严重性能问题	系统性能随时间的增加逐渐下降, 可预见的死锁, 系统突然出现混乱	通常是由于遗漏了Finally模块，或者只是没有用close函数关闭代表外部资源的对象
外部瓶颈	后台或其它外部系统（如用户验证）运行缓慢，大大影响J2EE应用服务器及应用程序的运行速度	系统持续缓慢运行, 系统性能随负载的增加逐渐下降	向专家（包括可靠的第三方或系统管理员等）咨询特定外部瓶颈问题的有效解决方法

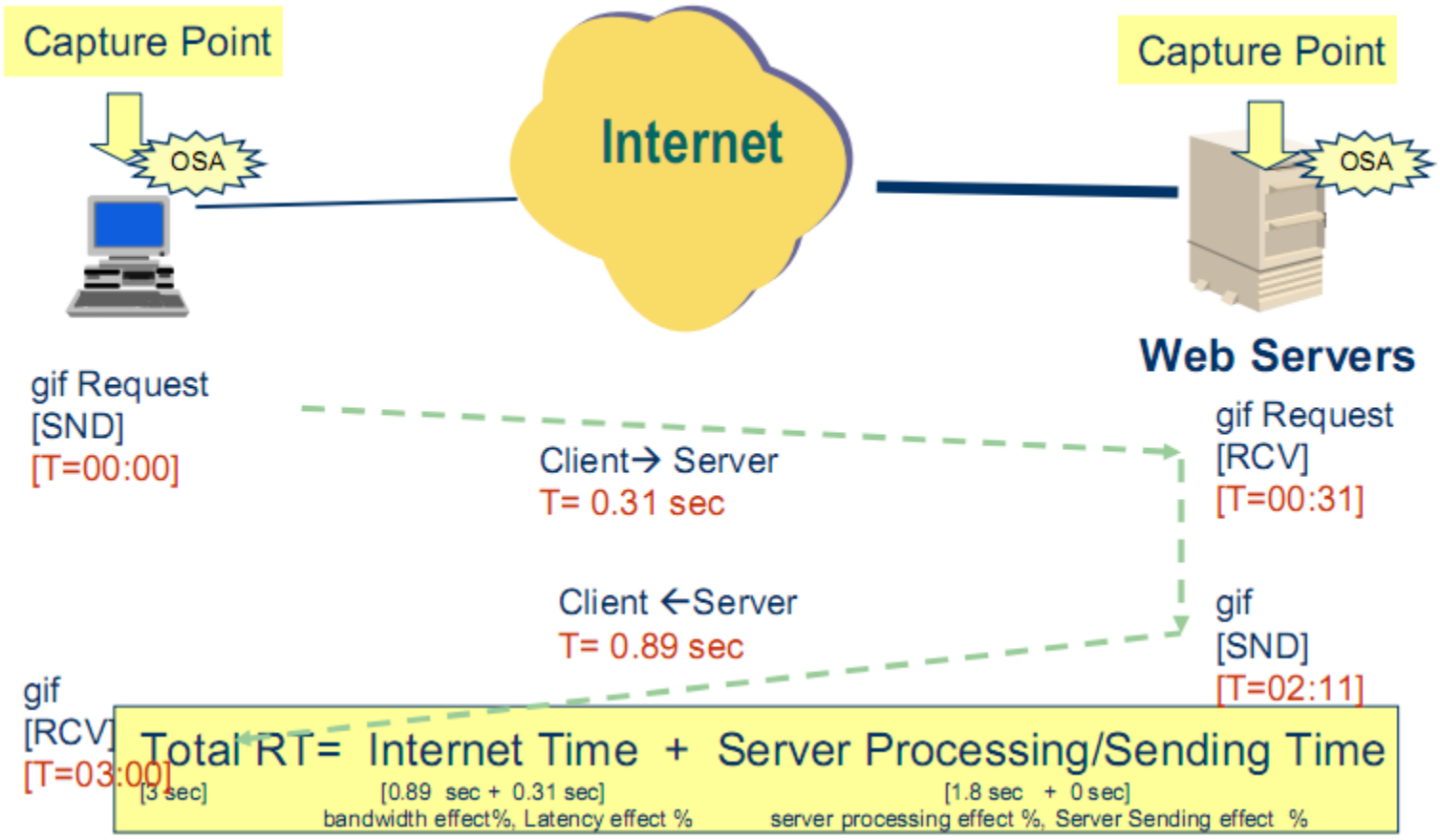
系统常见问题及解决方法

“疾病”	描述	征兆	原因或解决方法
外部系统的过度使用	J2EE应用程序发送的请求过大过多，滥用后台系统资源	系统持续缓慢运行, 系统性能随负载的增加逐渐下降	消除冗余的工作请求，分批处理同类工作请求，把大请求细分为若干个小请求，调整工作请求或后台系统（例如为公共查询的关键字建立索引）等
频繁调用CPU密集型组件的编码缺陷	J2EE领域的通病是：些许编码缺陷或少量编码的交互失败，都会令CPU挂起，从而将数据流量速度降至蜗行	系统持续缓慢运行, 系统性能随负载的增加逐渐下降	最好的解决方法是将数据存储在本地缓存中，或者为执行算法配备高速缓冲存储器
桥接层本身存在的问题	桥接层（如JDBC驱动、CORBA到遗留系统的连接等）存在执行缺陷。需要不断对数据和请求作编组和取消编组操作，导致该层的数据流量速度降至蜗行。这种故障现象在早期阶段与外部瓶颈极为相似	系统持续缓慢运行, 系统性能随负载的增加逐渐下降	检查桥接层与外部系统的版本是否兼容。如果可能的话，对不同桥接供应商进行评估。通过重新规划设计系统架构，则可完全旁路桥接层
内部资源瓶颈：资源的过度使用或分配不足	内部资源（如线程、放入存储池的对象等）匮乏，却无法判断是正常情况下随负载增加而引起的资源过度使用，还是由于资源泄漏引起	系统性能随负载的增加逐渐下降, 间发性的系统挂起或异常错误	若因资源分配不足引起，则可依照最大预期负载值上调存储池的最大容量；若因资源过度使用引起，请参看本表“外部系统的过度使用”一栏

系统常见问题及解决方法

“疾病”	描述	征兆	原因或解决方法
不断重试	失败请求的频繁重试（某些极端情况下将无休止重试）	可预见的死锁 系统突然出现混乱	后台系统可能已经完全宕机。监控系统可用性对这样的状况有所帮助，也可以只是将多次重复尝试的请求从成功的请求中筛选出来
线程阻塞点	线程退回到无法完成的同步点，造成通信阻塞	系统性能随负载的增加逐渐下降, 间发性的系统挂起或异常错误, 可预见的死锁, 系统突然出现混乱	或许根本没有必要进行同步（只需对系统重新设计稍加改良），当然也可以定制一些外在的锁定策略（如读取器或写入器的读/写锁）
线程的死锁或活锁	通常只是“获取顺序”问题	系统突然出现混乱	解决方法选项包括主锁、即定的获取顺序以及银行家算法
网络饱和	等待时间长或基本无法将任何请求打包，导致系统异常停运、系统挂起或活锁等情况的出现	系统持续缓慢运行, 系统性能随负载的增加逐渐下降, 间发性的系统挂起或异常错误	如此棘手的问题正在侵蚀着网络系统，如果不及时升级系统的基础架构，提升网络及路由器的运行速度，将无法扼制日后类似问题的出现

网络两点之间数据分析


Capture Point
OSA
Internet
Capture Point
OSA
Web Servers
gif Request
[SND]
[T=00:00]
Client→ Server
T= 0.31 sec
gif Request
[RCV]
[T=00:31]
Client ←Server
T= 0.89 sec
gif
[SND]
[T=02:11]
gif
[RCV]
[T=03:00]
Total RT= Internet Time + Server Processing/Sending Time
[3 sec]
[0.89 sec + 0.31 sec]
bandwidth effect%, Latency effect %
[1.8 sec + 0 sec]
server processing effect %, Server Sending effect %

分析性能问题需要的信息

- 引导使用者提供完整的信息
  - When(何时发生？)
  - What(那个操作？)
  - Who(谁操作的？)
  - Where(在何处操作？)
  - How(操作程序？)

SQL性能分析

- 首先获得数据库系统中每一条 SQL语句在数据库中执行的平均时间

- 然后将效率低下并且频繁调用的SQL语句的执行时间划分为四部分：解析时间（Parse Time）、执行时间（Execute Time）、读取时间（Fetch Time）和其他时间（Other Time），其中其他时间包括数据库中消耗的一些时间，例如绑定时间（bind time）

- 优化SQL语句

SQL性能分析

- 数据库锁管理
- 数据库lock管理
- 数据库latch管理

- sql语句的优化
- sql语句的定位
- sql语句的执行计划
- 全表扫描语句的定位
- sql语句的优化建议

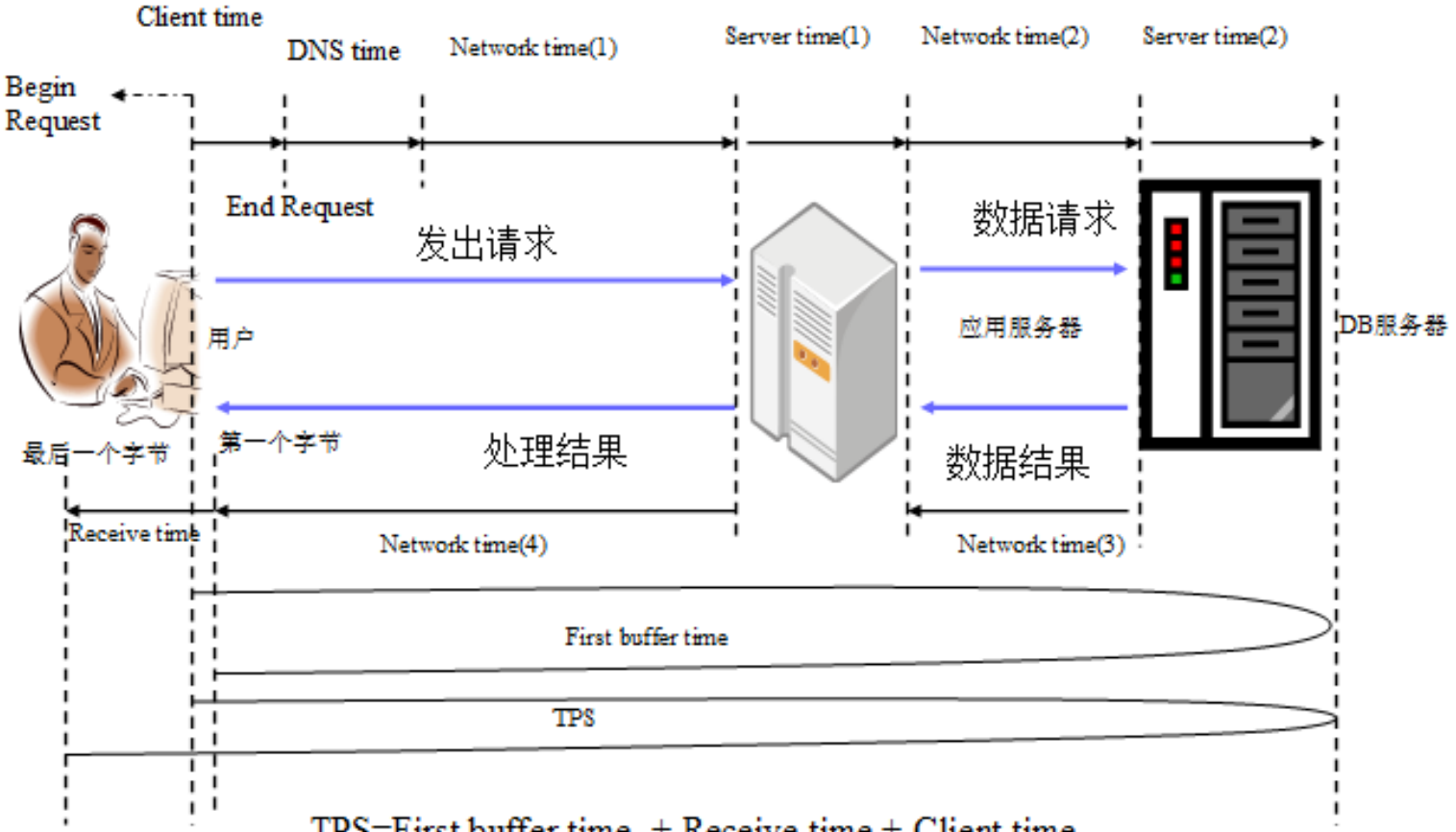
Process Time & Queue Time

- Response Time = Process Time + Queue Time
  - Process time : average response time of single thread
  - Queue time : average response time of multi threads
- Queue Time = Queue Length * Process Time

图形中的拐点

- **表示响应时间突然增加**
- **意味着一种或者多种系统资源的利用达到了极限**
- **调优的方法**
  - 每次改变一个系统参数或者一个应用逻辑
  - 使用固定的负载
  - 测试另一个设置之前收集本次性能测试的数据。
  - 重复测试过程，直到应用程序的性能达到了期望的状态。
- **例如：**
  - 很多Web服务器可以设置固定数量的threads来处理用户同时发出的请求。
  - 当这些并发的请求数量超过当前有效的threads数量时，任何新到的请求将会被放入一个队列中等待系统的处理。
  - 这个在队列中等待的时间将会使响应时间大大的增加。

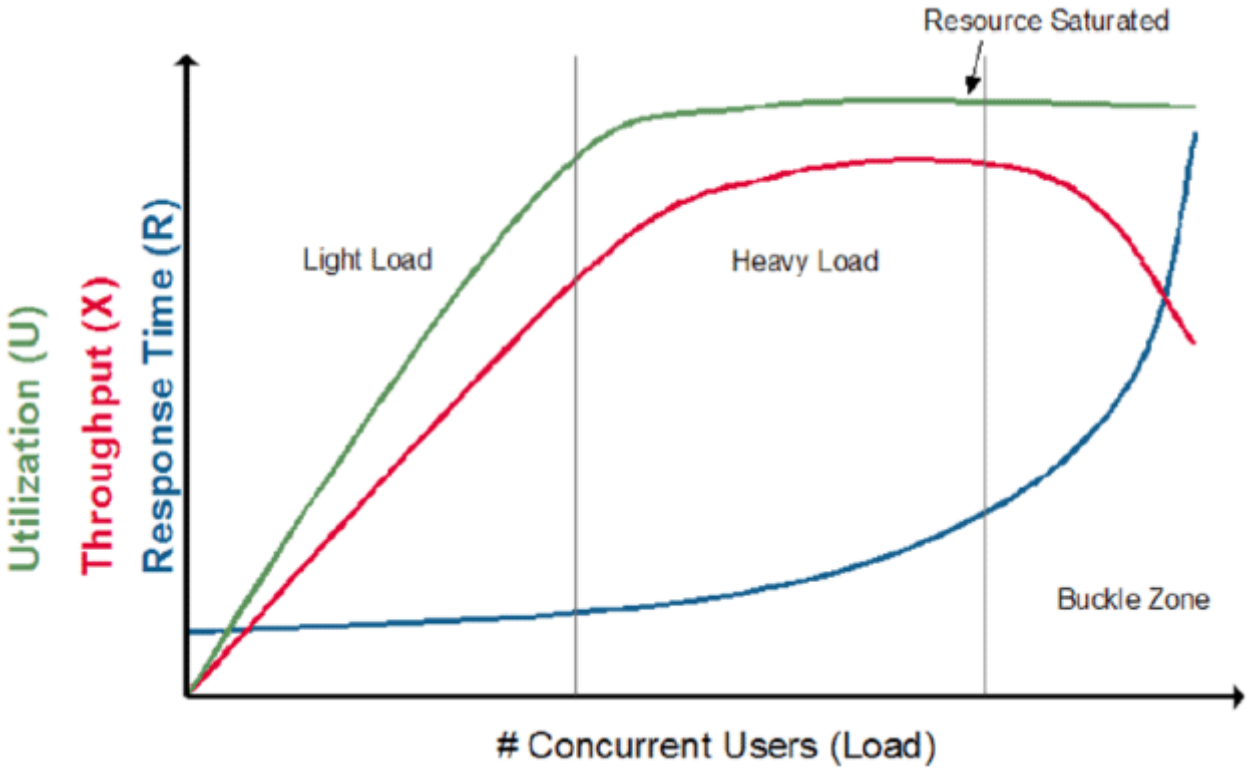
响应时间的分解



TPS=First buffer time + Receive time + Client time

（用户体验到的响应时间）End-to-end GUI TPS=TPS + 浏览器解释时间

性能指标VS用户数


Resource Saturated
Light Load
Heavy Load
Buckle Zone
Utilization (U)
Throughput (X)
Response Time (R)
Concurrent Users (Load)

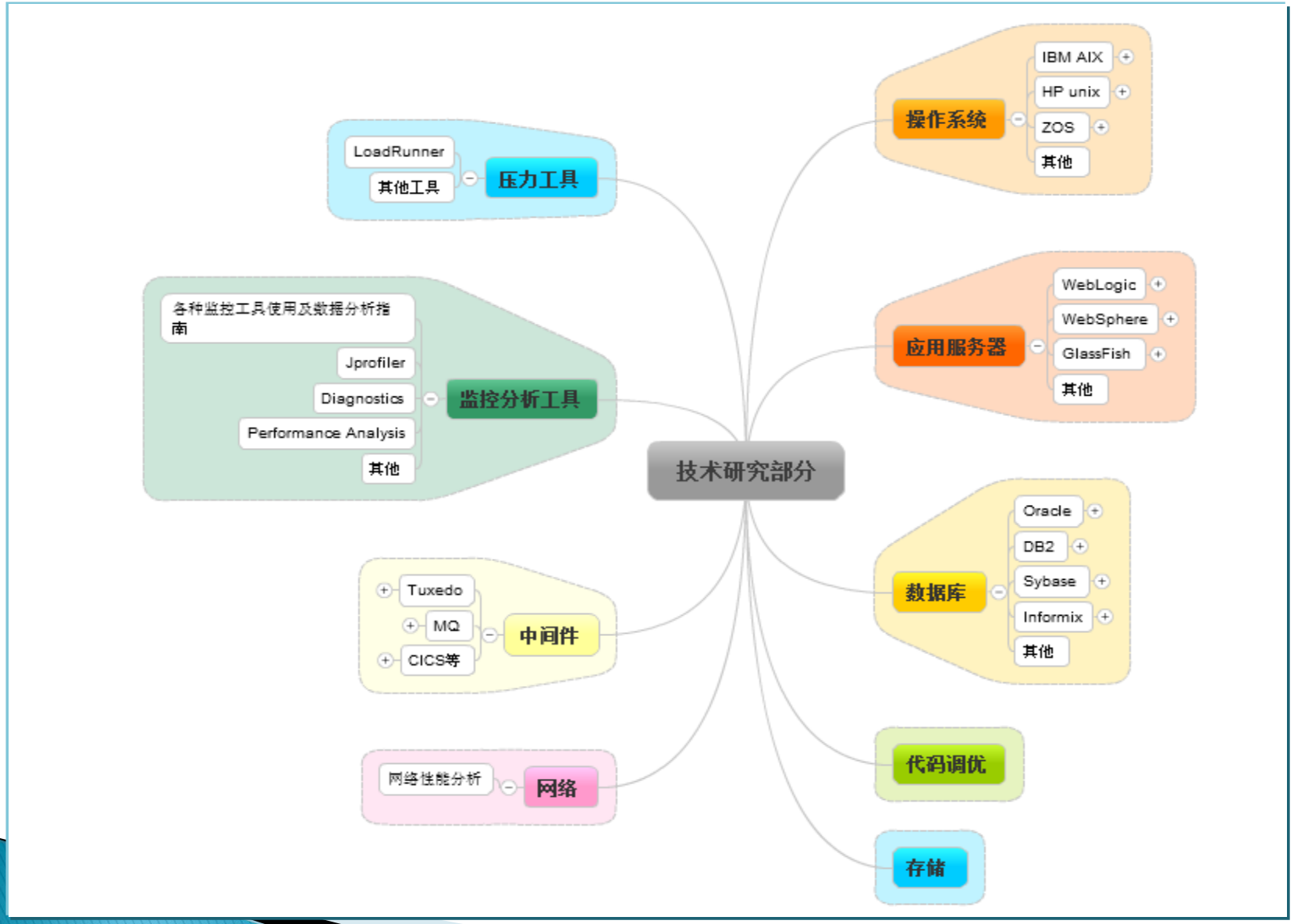
数据敏感性（性能测试人员）

- 1秒的影响

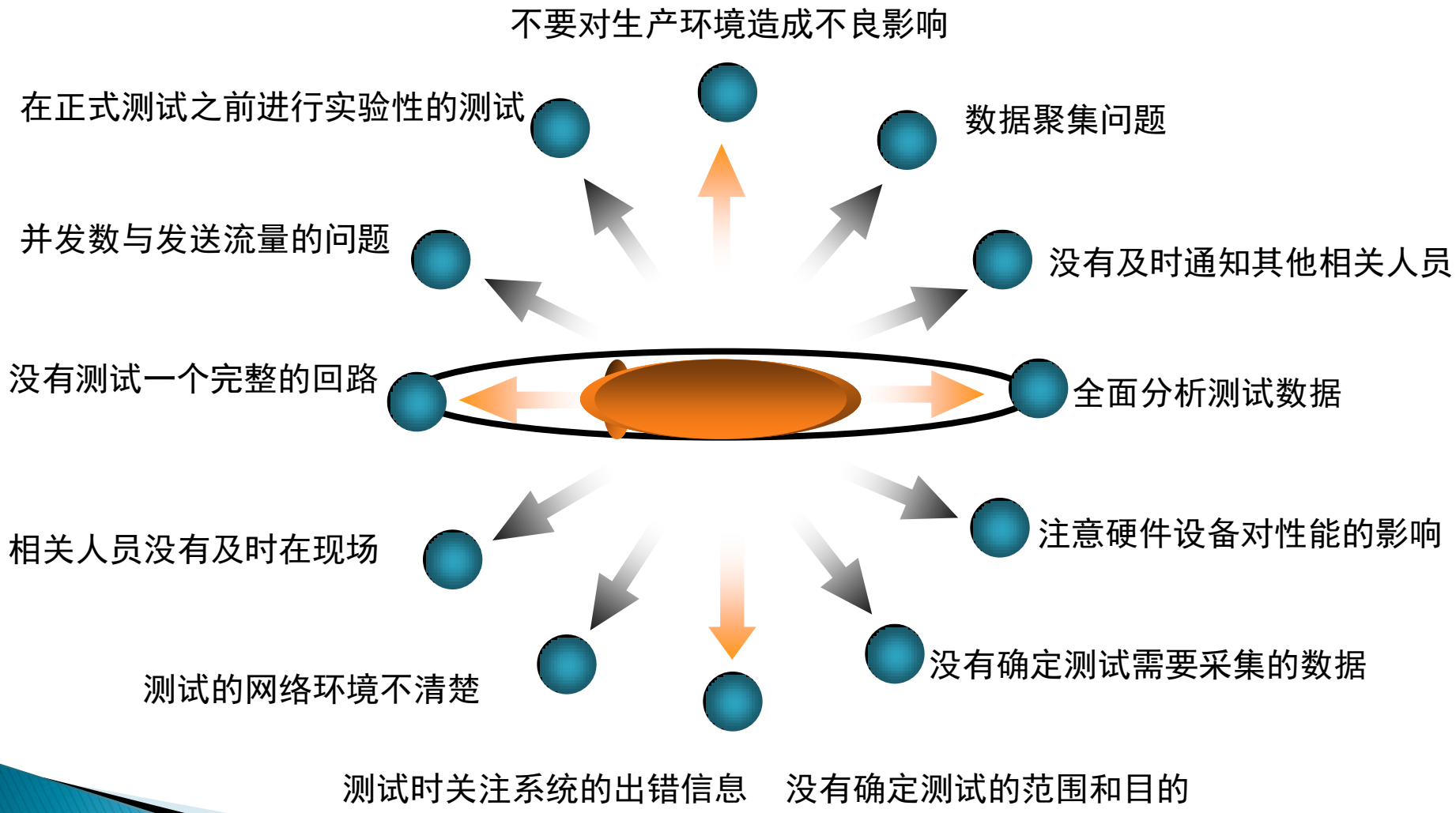
对于单个用户的响应时间，1秒的差距可能不显著，甚至被忽视。但是对于一个不间断的被成百上千用户同时访问的服务器来说，每个用户的响应时间都缩短1秒将是非常巨大的区别。

举这个例子是为了说明性能测试人员应该对数据敏感。

与性能调整有关的议题


技术研究部分
操作系统
IBM AIX
HP unix
ZOS
其他
应用服务器
WebLogic
WebSphere
GlassFish
其他
数据库
Oracle
DB2
Sybase
Informix
其他
代码调优
存储
压力工具
LoadRunner
其他工具
监控分析工具
各种监控工具使用及数据分析指南
Jprofiler
Diagnostics
Performance Analysis
其他
中间件
Tuxedo
MQ
CICS等
网络
网络性能分析

性能测试中需要注意的一些问题



整体目录

理解性能测试

性能测试需求的获取和分析

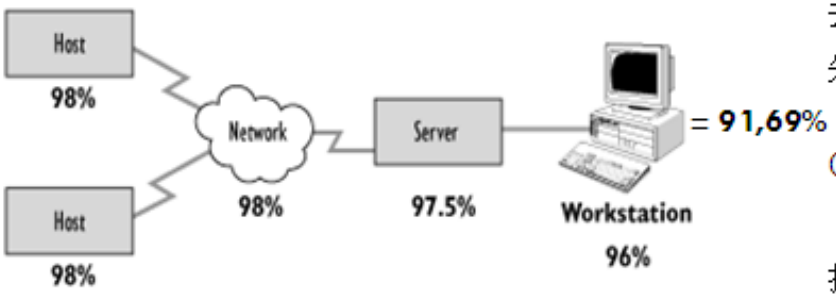
性能测试执行、控制及分析

可用性统计分析

排队论在分析过程中的应用

性能问题实例

可用性基本计算公式



并联关系计算方法：

先计算两个主机的可用性＝ 1 － （1－Host）×（1－Host）

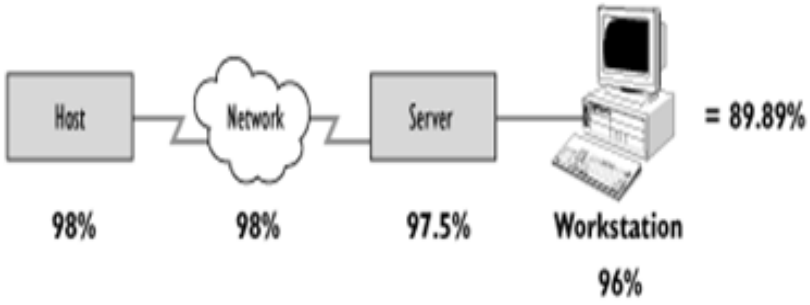
＝ 1 － （1－0.98）×（1－0.98）

＝ 0.9996

按照串联关系计算可用性＝ 0.9996 * 0.98 * 0.975 * 0.96

= 0.9169

总的可用性等于91.69%



串行关系计算方法：

服务的可用性＝ Host × Network × Server× Workstation

等于 ＝ 0.98 × 0.98 × 0.975 × 0.96 = 0.8989

总的可用性等于 89.89%

N+M并行系统可用性计算公式

串行节点计算公式 → $R_n = R_1 * R_2 * R_3 * R_n$

并行两个节点计算公式 → $R_{1+1}=1-(1-R_1) \times (1-R_2) = 1-(1-R)^2$

并行n个节点计算公式 → $R_{n+m}(t) = \left[1-(1-R)^{m+1}\right]^{C_{n+m}^{m+1}}$

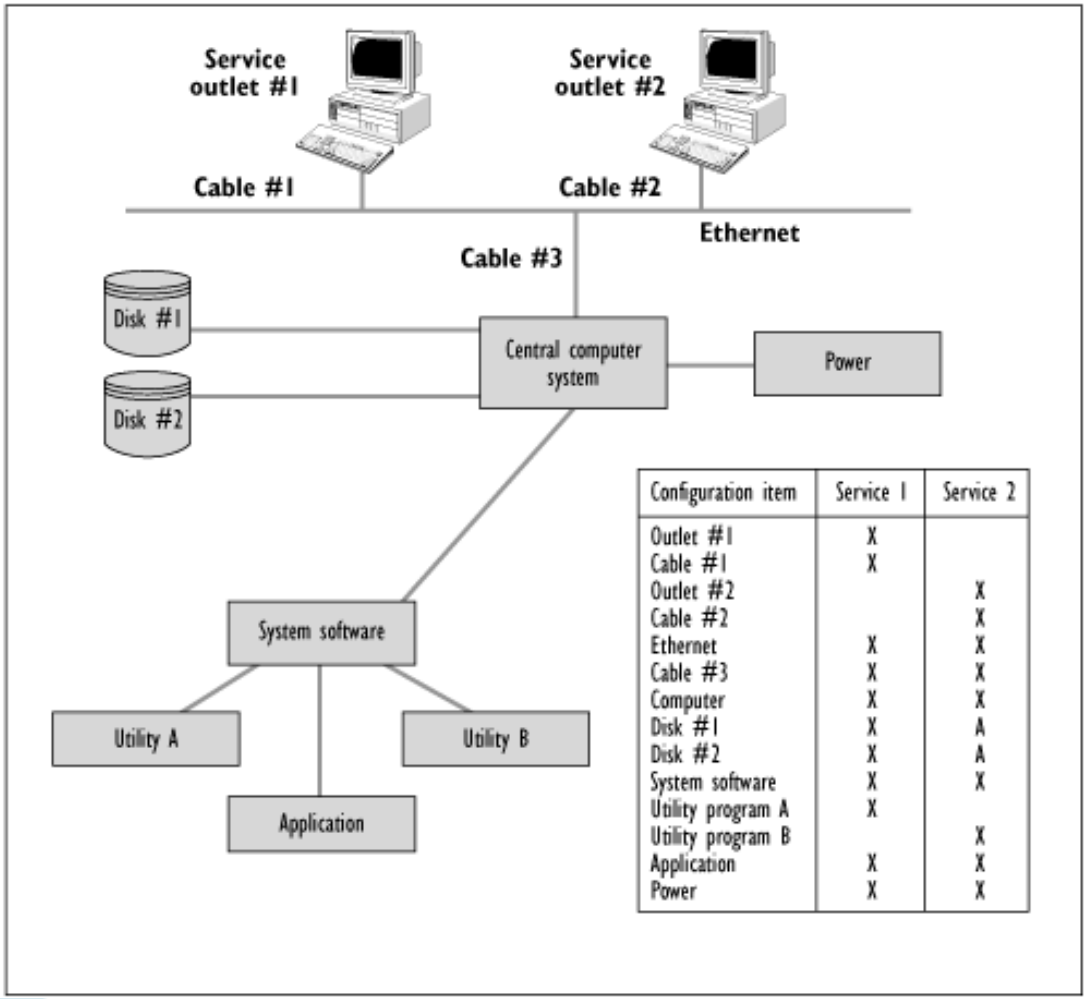
并行m个节点的幂 → $C_{n+m}^{m+1} = \frac{(n+m)!}{(m+1)!(n+m-m-1)!} = \frac{(n+m)!}{(m+1)!(n-1)!}$

说明:

- *R 为单个模块的可用性*
- *N为n 个模块*
- *M为m 个并行或冗余模块*

可用性分析方法

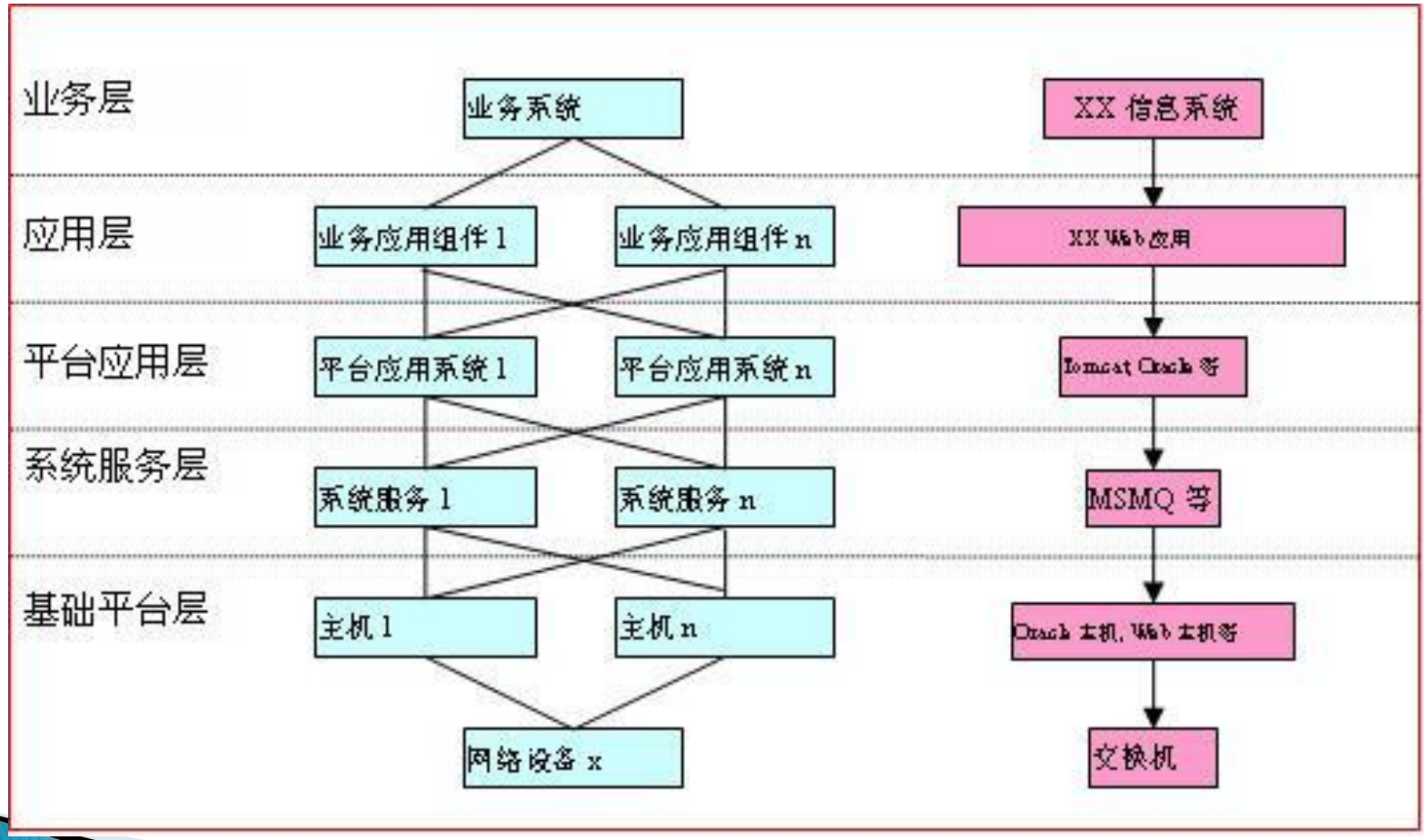
- **组件故障影响分析 (Component Failure Impact Analysis, CFIA)**



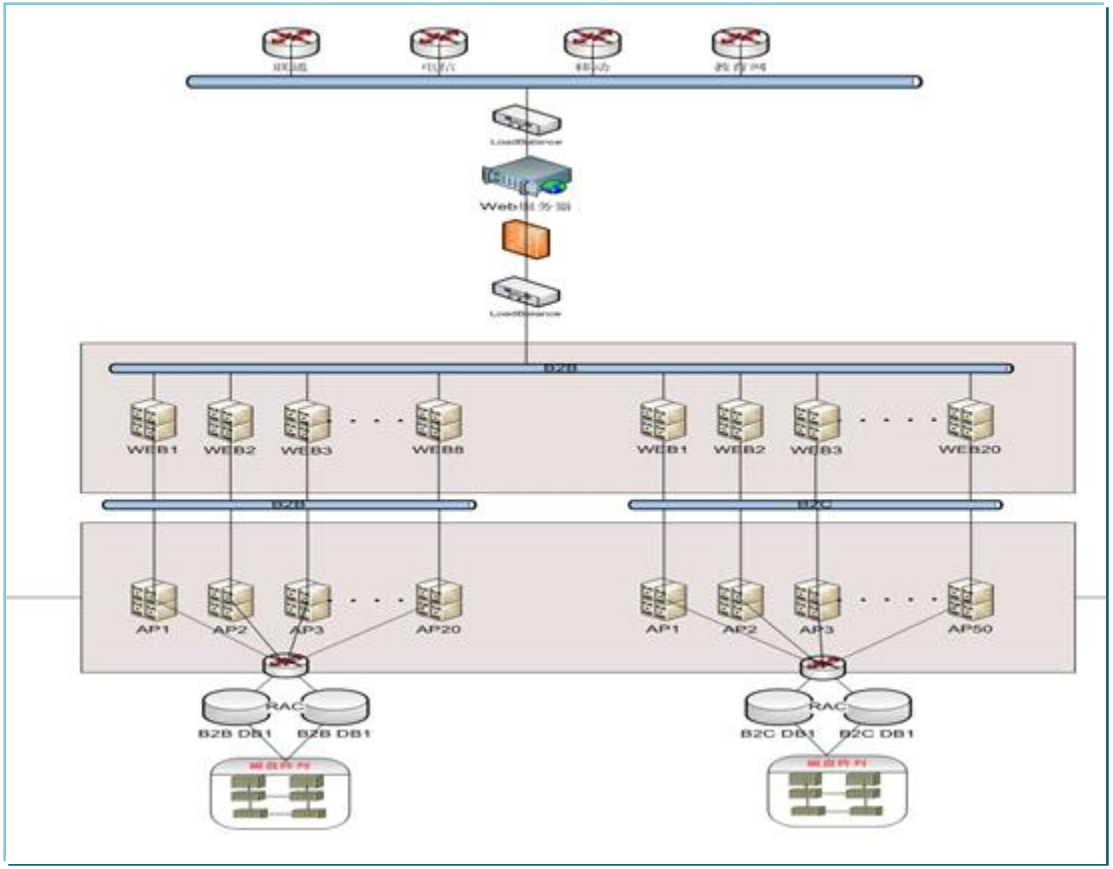
Configuration item	Service 1	Service 2
Outlet #1	X	
Cable #1	X	
Outlet #2		X
Cable #2		X
Ethernet	X	X
Cable #3	X	X
Computer	X	X
Disk #1	X	A
Disk #2	X	A
System software	X	X
Utility program A	X	
Utility program B		X
Application	X	X
Power	X	X

业务应用分析模型

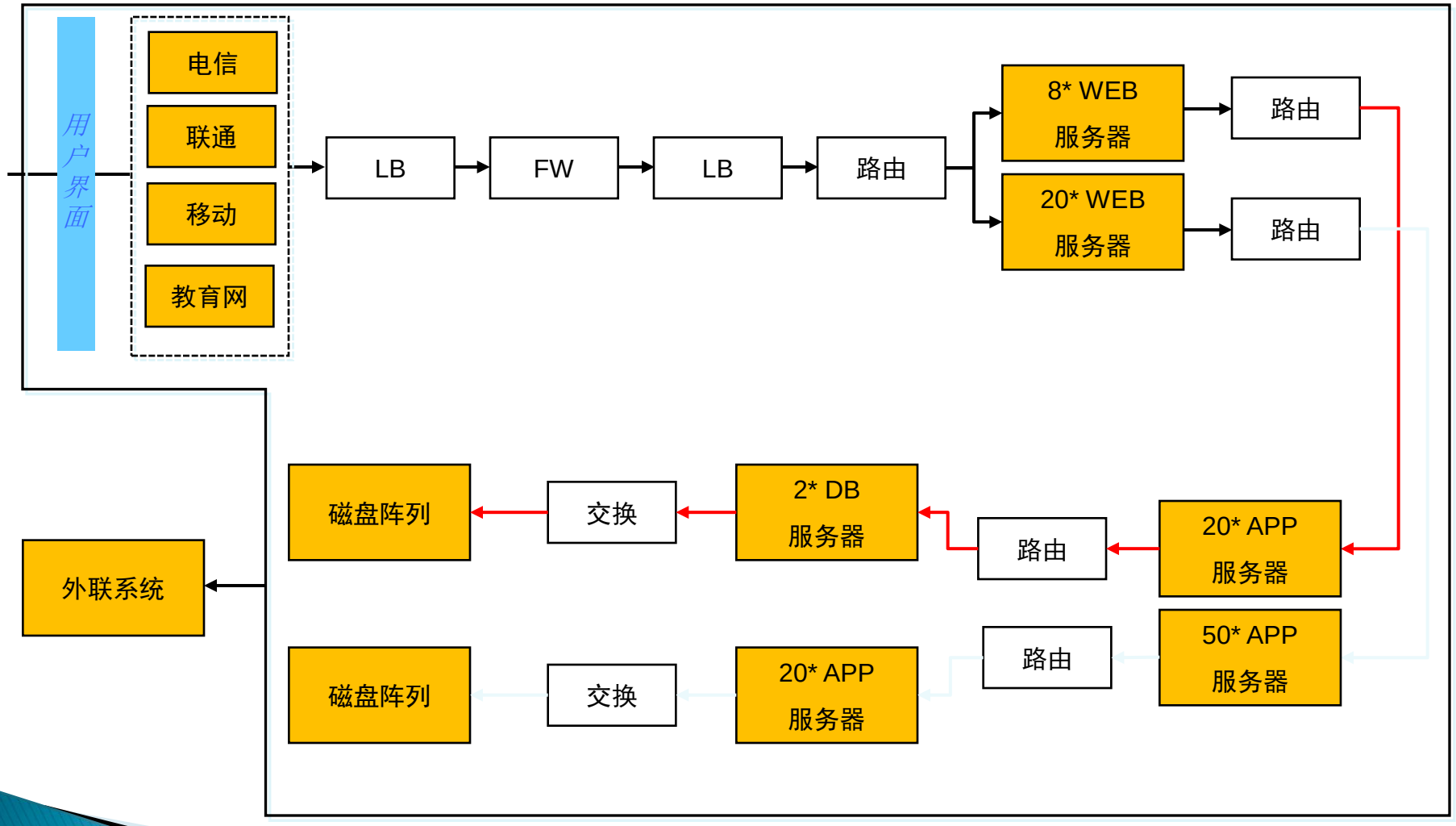
- 按照组件分解的方法，将业务应用分解为处于不同的层次的组件或模块，并在每个组件上定义不同的评估指标来衡量组件的可用性。



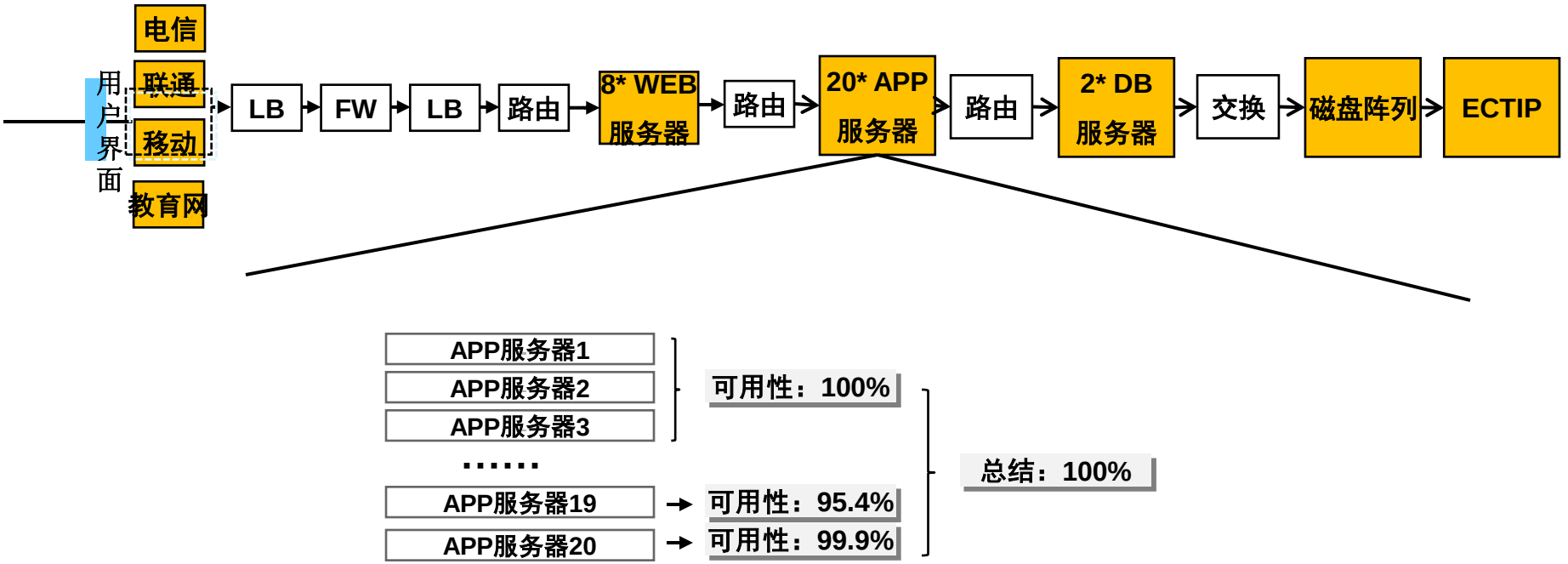
可用性示例－网络拓扑


B2B
WEB1
WEB2
WEB3
WEB8
WEB1
WEB2
WEB3
WEB20
B2B
B2C
AP1
AP2
AP3
AP20
AP1
AP2
AP3
AP50
RAC
B2B DB1
B2B DB1
RAC
B2C DB1
B2C DB1

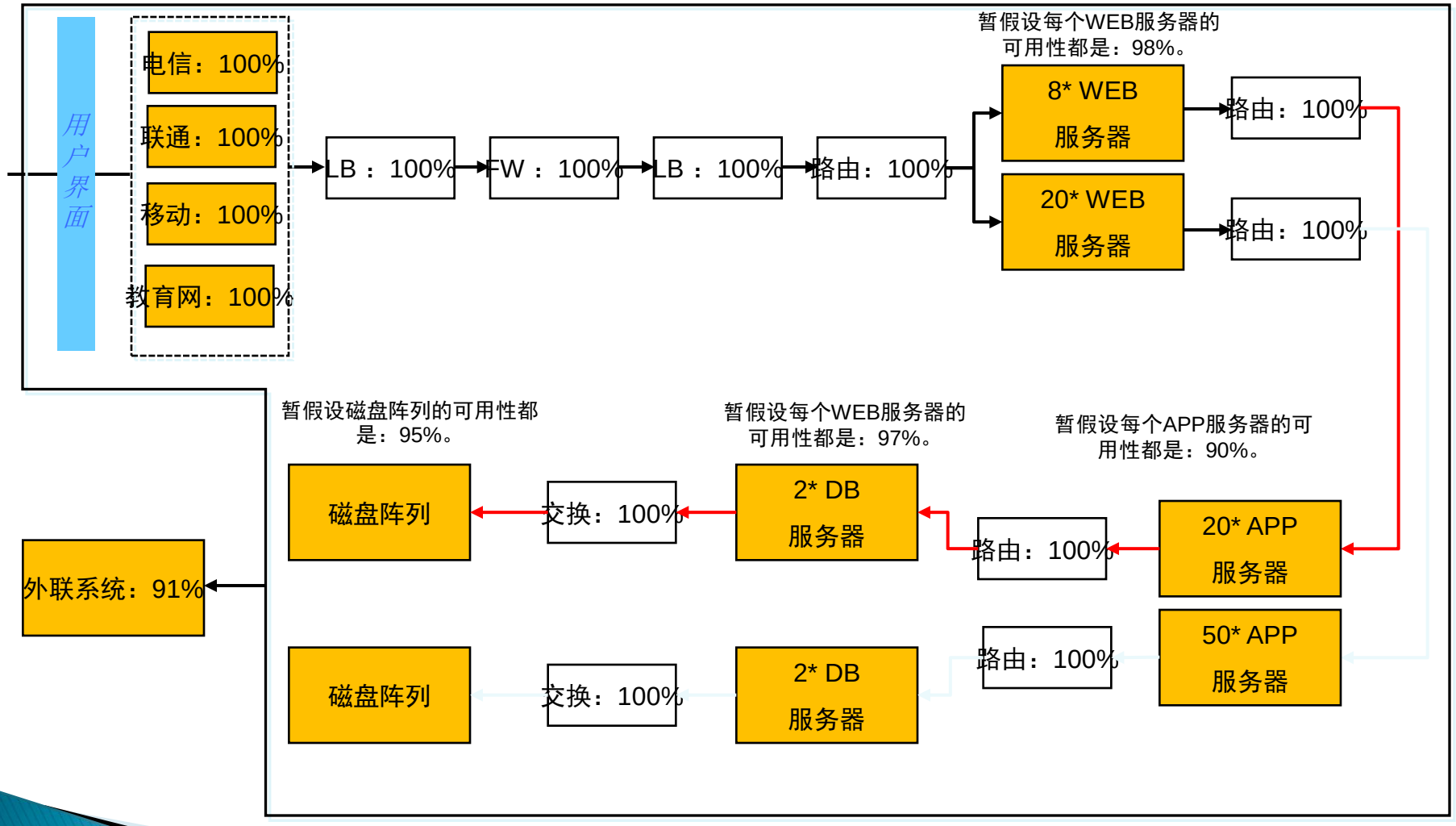
组件故障关系分析（1）


用户界面
电信
联通
移动
教育网
LB
FW
LB
路由
8* WEB 服务器
20* WEB 服务器
路由
路由
磁盘阵列
交换
2* DB 服务器
路由
20* APP 服务器
外联系统
磁盘阵列
交换
20* APP 服务器
路由
50* APP 服务器

组件故障关系分析（2）


电信
联通
移动
教育网
用户界面
LB
FW
LB
路由
8* WEB 服务器
路由
20* APP 服务器
路由
2* DB 服务器
交换
磁盘阵列
ECTIP
APP服务器1
APP服务器2
APP服务器3
.....
APP服务器19
APP服务器20
可用性：100%
可用性：95.4%
可用性：99.9%
总结：100%

计算示例：假设各节点可用率



计算示例（续）：代入公式

可用性 =

(100%*100%*100%* 100%)*

(100%*100%*100%* 100%)*

(98%*98%)*

(100%*100%)* (90%*90%)*

(100%*100%)*

(97%*97%)*

(100%*100%)*

(95%*95%)*

91%

= 60.113116%

整体目录

理解性能测试

性能测试需求的获取和分析

性能测试执行、控制及分析

可用性统计分析

排队论在分析过程中的应用

性能问题实例

排队论的应用

- 什么是排队论？

  排队论是20世纪初由丹麦数学家Erlang应用数学方法在研究电话话务理论过程中而发展起来的一门学科，排队论也称随机服务系统理论，它涉及的是建立一些数学模型，以对随机发生的需求提供服务的系统预测其行为，它已应用于电讯、纺织、矿山、交通、机器维修，可靠性，计算机设计和军事领域，都已取得了显著的成绩。

- 排队论对性能测试分析的作用（举例说明）

  分析用户队列

  分析系统队列（各层面）

  分析网络队列

公式

M/M/C/∞(输入分布类型/服务时间分布类型/服务台数量/系统容量）

排队论公式

符号	定　　义
$\overline{Rs}$	被服务需求的平均网站反应时间
Rt	被服务需求的总网站反应时间
λ	需求到达率
L_n	未被服务的需求数
C	网站连接时间
n	被服务的需求数
N	全部需求数
$W_q(i)$	第 i 个需求在等待队列中的时间(秒)
t_1	第一个需求到达等待队列的时间(秒)
t_2	最后一个需求到达的等待队列的时间(秒)

定义Wq(i)有两种情况,
第一种为Wq(i)小于网站连接时间, 有被服务的需求平均等待时间如下:
Rs= Rt/n, 当Wq(i) < c

另一种情况是当Wq(i)大于网站连接时间时,这
些需求将无法被网站服务器服务, 因此未被服务的
需求数可以用以下的公式表示:
Ln =N- n , 当Wq(i) ≥c
因λ= N/t1- t2
故N=(t2- t1)λ
最后修改后的Little 公式为
Ln=N- n=λ(t2- t1)- Rt/Rs

实例

- 下面以某大型网站系统为例来简要说明修改后的公式的具体应用。
  (一) 数据收集
  数据收集的对象是用修改后的Little 公式计算出没有被服务的需求数目, 在这个测试环境中, 我们设定全部的需求在一秒内进入网站服务器中, 进入需求的数目分别是500、1000 与1500。使用工具分析和Ln, 结果如表2:

N	t_2- t_1	λ	Rt	$\overline{Rs}$	L_n	N- L_n
500	1	500	1358	3	47.334(47)	453
1000	1	1000	15412	17	93.412(93)	907
1500	1	1500	21634	20	118.3(118)	1081

接上实例

- 分析结果
  根据修改后Little 公式以及表2 的数据, 我们计算出在不同网站连接时间C 和需求数N 下的服务率及反应时间, 如表3:

全部需求	C=30		C=40		C=50	
	服务率（%）	反应时间（s）	服务率（%）	反应时间（s）	服务率（%）	反应时间（s）
500	90.6	3	95.5	11	97.4	19
1000	90.7	17	94.7	26	92.3	34
1500	72	20	79.8	28	86.1	39

整体目录

理解性能测试

性能测试需求的获取和分析

性能测试执行、控制及分析

可用性统计分析

排队论在分析过程中的应用

性能问题实例

性能问题及建议— 数据库死锁导致交易失败

示例语句：

HASH_VAL	SQL_TEXT	USERNAME	RES	TAB	OWNER	LMODE
2.14E+09	delete from t_claim_policy wher	ULPRDAPP	TM - DML Enqueue	T_CLAIM_CA	ULPRD	Row Share
2.14E+09	delete from t_claim_policy wher	ULPRDAPP	TM - DML Enqueue	UL_T_PROD_	ULPRD	Row Share
2.14E+09	delete from t_claim_policy wher	ULPRDAPP	TM - DML Enqueue	T_PRODUCT_	ULPRD	Row Share
2.14E+09	delete from t_claim_policy wher	ULPRDAPP	TM - DML Enqueue	T_PRODUCT_	ULPRD	Row Share
2.14E+09	delete from t_claim_policy wher	ULPRDAPP	TM - DML Enqueue	T_PREM_STA	ULPRD	Row Share

Alert信息：

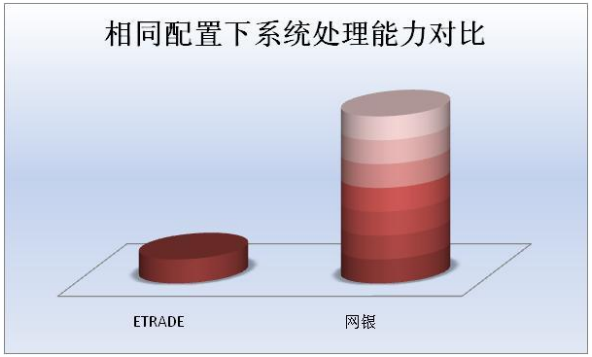
Tue Nov 16 15:32:52 2010

ORA-00060: Deadlock detected. More info in file /home/oracle/app/admin/ulprd/udump/ptest_ora_4223614.trc.

Tue Nov 16 15:32:52 2010

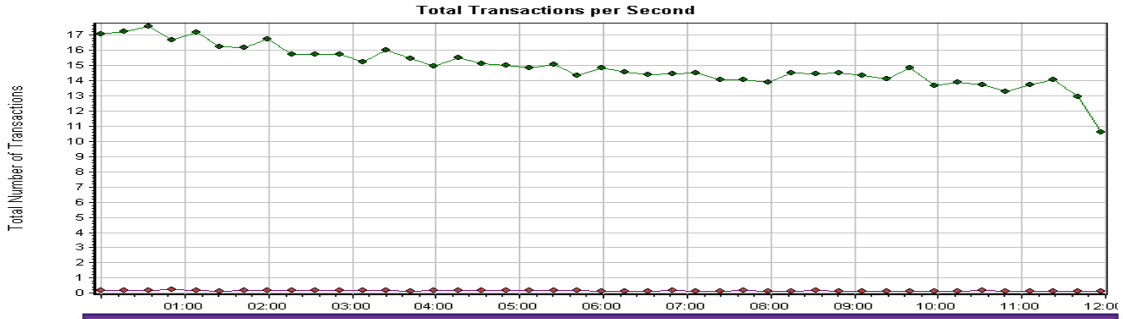
ORA-00060: Deadlock detected. More info in file /home/oracle/app/admin/ulprd/udump/ptest_ora_4305574.trc.

性能问题及建议— 性能衰减



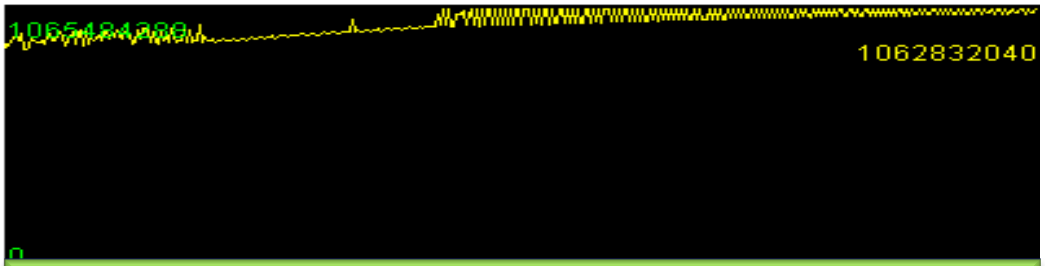
系统仅能支持10万用户并发交易处理能力同比最低

1



系统无法长期稳定运行，存在安全隐患

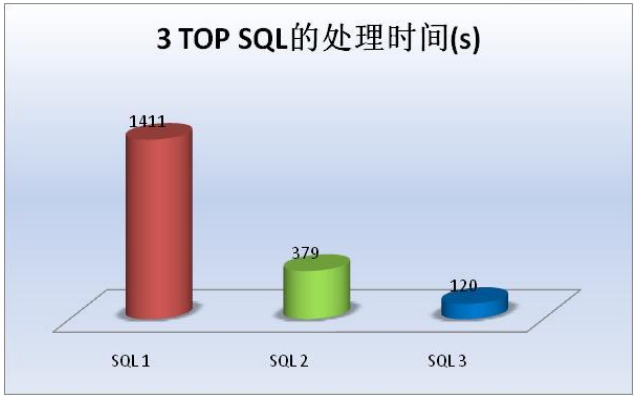
2



内存泄露问题未得到修复，可能造成生产事故

系统问题分析

3



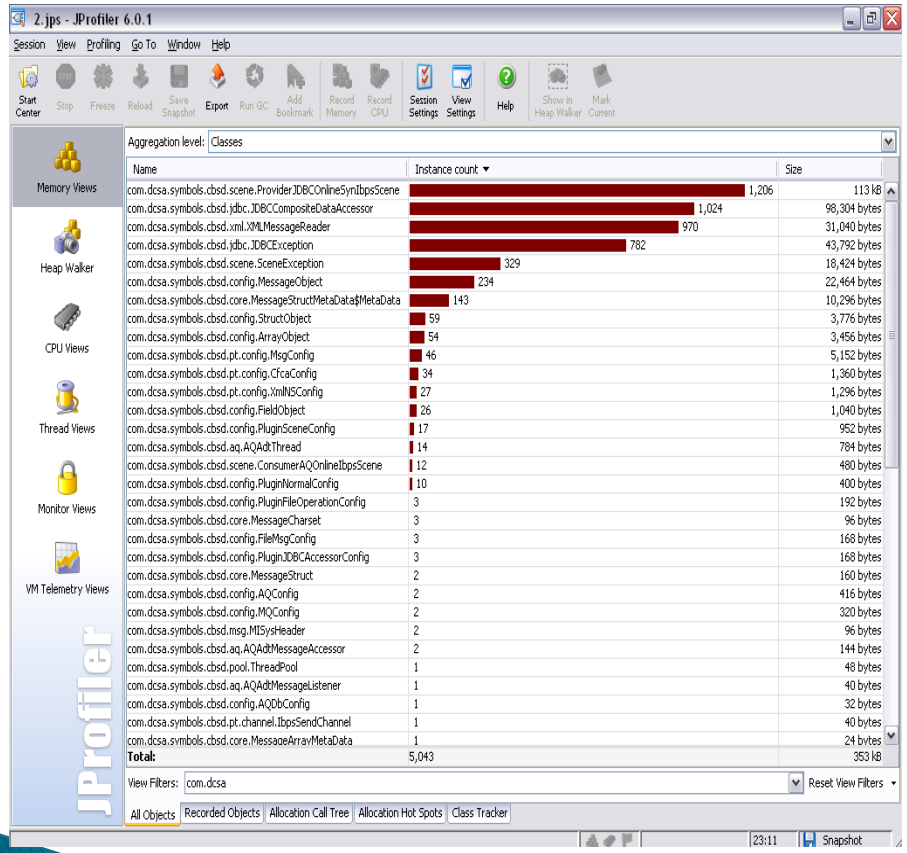
4

CHAR字段6000多个，LONG字段28个

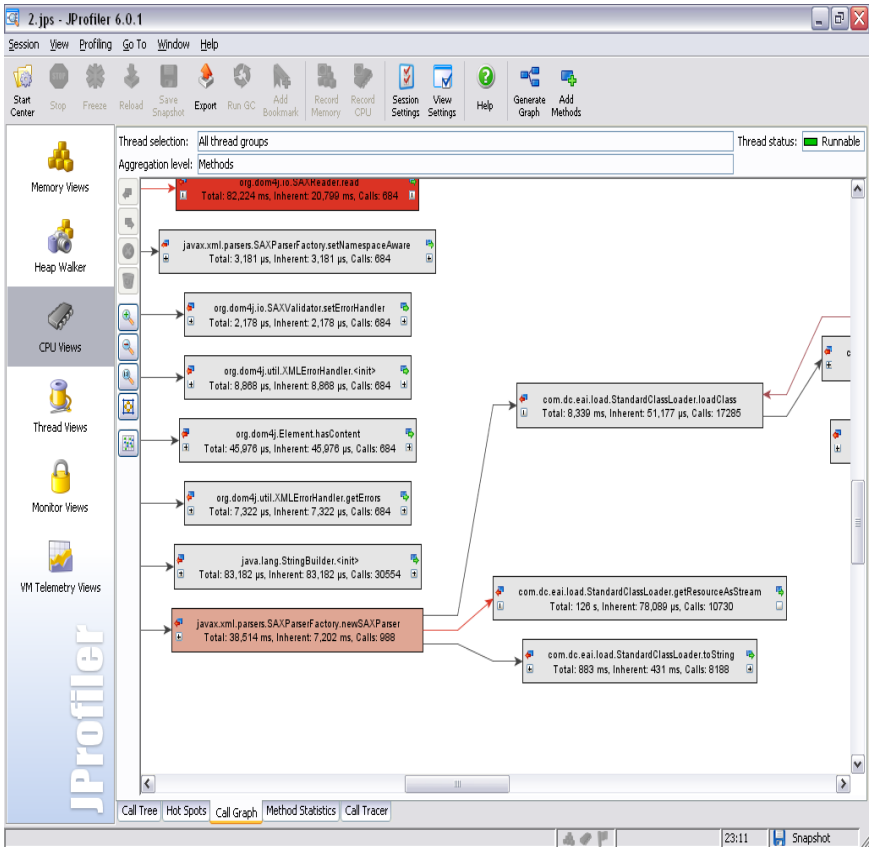
数据库技术过时，性能低下

性能问题及建议－ 内存问题定位及方法调用时间分析

内存问题定位

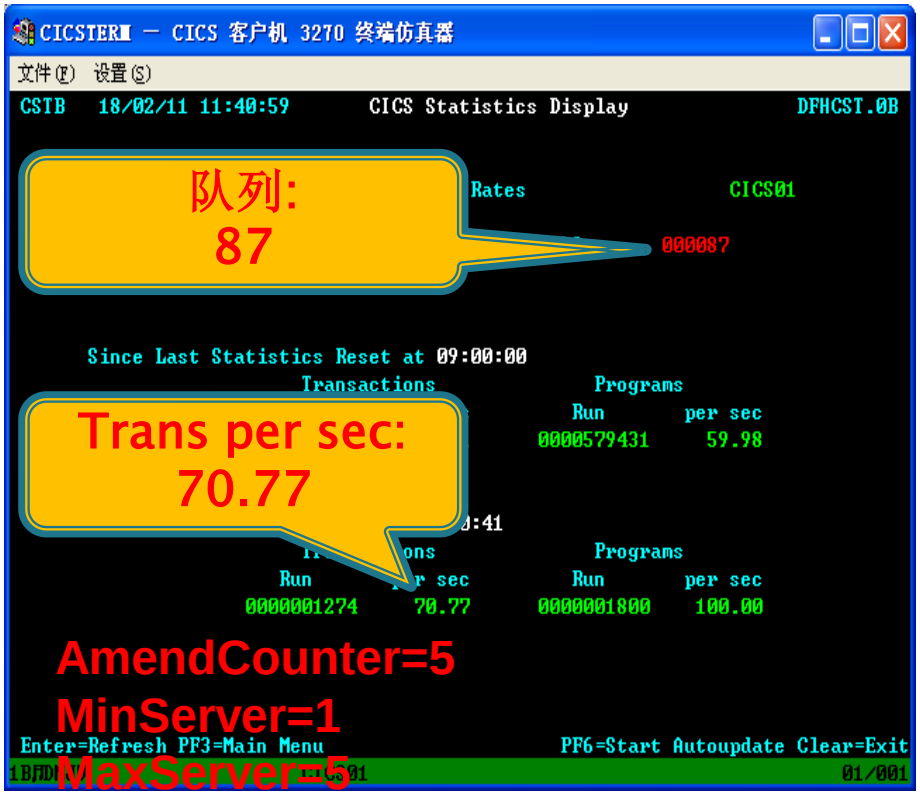


方法调用时间分析



性能问题及建议— 中间件问题

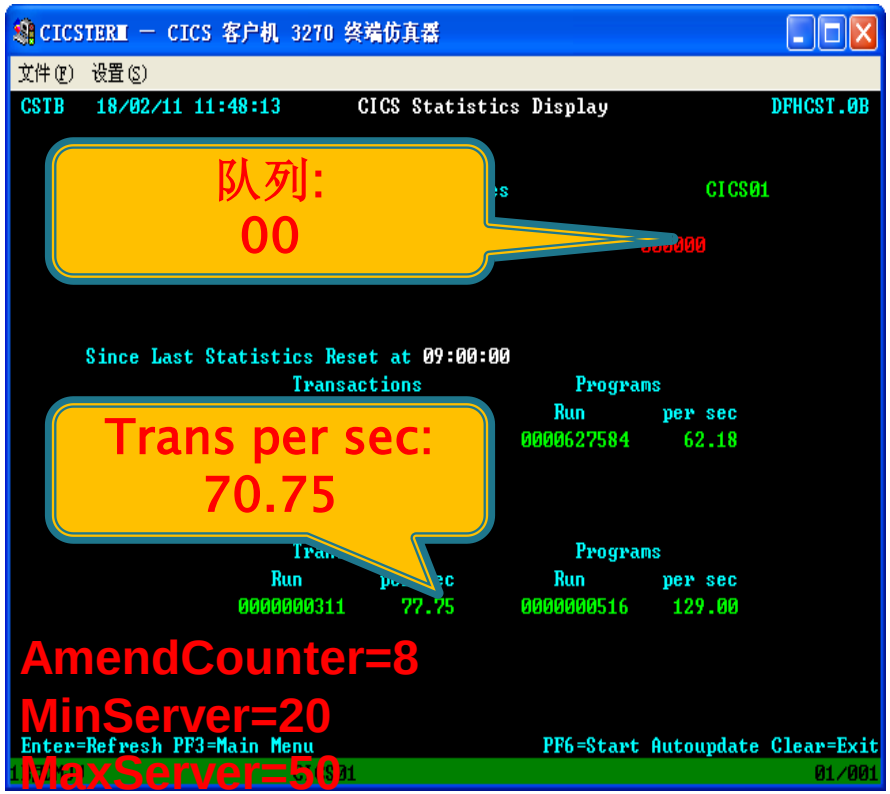
120个用户



AmendCounter=5
MinServer=1
MaxServer=5

MaxRegionPool=2097152
MaxTaskPrivatePool=1048576
MaxTSHPool=1048576

74个用户



AmendCounter=8
MinServer=20
MaxServer=50

MaxRegionPool=80971520
MaxTaskPrivatePool=20485760
MaxTSHPool=40485760

Questions

- Do you prefer to ask me some questions?

The End

http://www.7dtest.com
http://bbs.7dtest.com