[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
H04L 12/56 (2006.01)

[12] 发明专利说明书

专利号 ZL 02815366.9

[45] 授权公告日 2007 年 1 月 24 日

[11] 授权公告号 CN 1297113C

[22] 申请日 2002.5.2 [21] 申请号 02815366.9
[30] 优先权
[32] 2001.6.7 [33] GB [31] 0113844.5
[86] 国际申请 PCT/GB2002/002016 2002.5.2
[87] 国际公布 WO2002/099555 英 2002.12.12
[85] 进入国家阶段日期 2004.2.5
[73] 专利权人 马科尼英国知识产权有限公司
地址 英国考文垂
[72] 发明人 P·科利特 B·P·詹森
S·I·马丁
[56] 参考文献
EP1075116A2 2001.2.7 H04L12/56
US5796956A 1998.8.18 G06F13/00
US5497375A 1996.3.5 H04J3/24
WO99/25148A2 1999.5.20 H04Q11/04
审查员 郭风顺
[74] 专利代理机构 中国专利代理(香港)有限公司
代理人 王 岳 王 勇

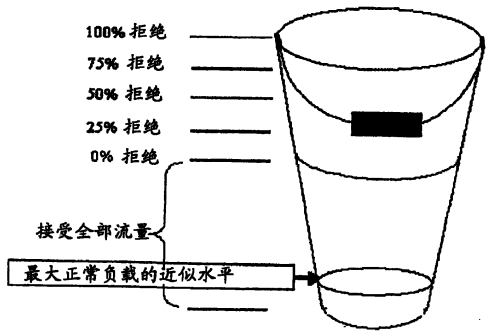
权利要求书 1 页 说明书 18 页 附图 12 页

[54] 发明名称

一种实时处理的系统和方法

[57] 摘要

一种实时业务处理系统，包括漏桶和流量控制装置，该流量控制装置监控每个实时业务的反应时间，并将所述反应时间和预定反应时间相比较，当所述反应时间超过预定反应时间时，则增加漏桶内的内容。 该实时业务处理系统可以根据漏桶内容成比例拒绝业务。 以及操作该实时业务处理系统的一种方法，该方法为监控每个实时业务的反应时间；将该反应时间和预定反应时间相比较；当该反应时间超过预定反应时间时，则增加该漏桶的内容。



1．一种业务处理系统(SCP)，包含漏桶和流量控制装置，所述流量控制装置用来监控每个业务的反应时间，并将每个反应时间和预定反应时间相比较，其中对不同的业务采用所述预定反应时间的至少两个不同值，并且当一个业务的反应时间超过与特定业务对应的预定反应时间时，则增加漏桶内的内容，且所述流量控制装置设计为监视漏桶的内容，并且在一固定值以上按照漏桶的内容成比例地拒绝业务，而在该固定值以下不拒绝任何流量。

2．权利要求 1 中所要求的业务处理系统（SCP），其中所述增加的漏桶内的内容是相应业务的函数。

3．权利要求 1 或者 2 中所要求的业务处理系统（SCP），其中所述业务是实时业务。

4．操作业务处理系统（SCP）的一种方法，该处理系统包含漏桶和流量控制装置，该方法包含步骤:

（a）监控每个业务的反应时间;

（b）将每个反应时间和预定反应时间相比较，其中对不同的业务采用所述预定反应时间的至少两个不同值;

（c）当一个业务的反应时间超过对应于特定业务的预定反应时间时，则增加漏桶内的内容；和

（d）监视漏桶的内容，并按照漏桶的内容成比例地拒绝业务。

5．权利要求 4 中所要求的方法，其中所述增加的漏桶内的内容是相应业务的函数。

6．权利要求 4 或者 5 中所要求的方法，其中所述业务是实时业务。

一种实时处理的系统和方法

本发明涉及包含一个漏桶的实时处理系统。

根据本发明的第一方面，提供了一种包含一个漏桶和流量控制装置的业务处理系统，所述流量控制装置用来监控每个实时业务的反应时间并将每个反应时间和预定反应时间进行比较，其中对不同业务采用所述预定反应时间的至少两个不同值，并且当一个业务的反应时间超过对应于特定业务的预定反应时间时增加该漏桶的内容。且所述流量控制装置设计为监视漏桶的内容，并按照漏桶的内容成比例地拒绝业务。

根据本发明的第二方面，提供了操作业务处理系统的方法。

本发明的进一步的特征在从属权利要求中叙述。

本发明将用举例的方式参考附图进行说明，其中：

图 1 示出了“漏桶”的图解表示以说明该发明的功能；

图 2 说明了为仿真而假定的单箱 SCP（服务控制点）结构；

图 3 说明了为仿真而假定的多箱 SCP（C）/SDP（服务数据点）结构；

图 4 示出了反应时间分配的示意性柱状图。这表示了在单箱 SCP 仿真上大约 70％的处理器占用率时，一次读服务运行的反应时间，该反应时间以毫秒为单位；

图 5 说明了变量 x＝占用率／（1－占用率）和 y＝95％反应时间之间的线性关系，其中反应时间以 ms 为单位。而 x＝1 对应 50％的占用率；

图 6 说明了读取次数 n，＝1，2，3，4，5 和 95％“排队等待”时间 T_q 之间的线性关系；

图 7A－7B 说明了在单箱 SPC 仿真上的一次写服务。对于在 SCP 仿真上测量到的数据，变量 x＝占用率／（100－占用率）和 95％反应时间（y 轴）之间的近似线性关系，该反应时间以 ms 为单位；

图 8A 和 8B 说明了两次写服务（8A）和三次写服务（8B）时如图 7A－7B 中那样的线性拟合。

图 9A－9C 说明了对于在实际单箱 SCP 上测量的数据，变量 x＝占

用率／（100－占用率）和95％反应时间（y轴）之间的近似线性关系，该反应时间以ms为单位，图9A是一次读服务，图9B是一次读，一次写服务，图9C是12次读，一次写服务；

图 10A－10D 说明了分别对于 1 次，3 次，5 次和 8 次读，95％时间延迟（y 轴）与占用率／（1－占用率）（x 轴）间的关系。占用率是（主控机）SDP 的 cpu 占用率，配置是 2 个 SCP（C）（各有 2 个处理器）和 2 个 SDP（主控机／受控机）（各有 2 个处理器）；

图 11 说明了对于 1 次读取，95％时间延迟（y 轴）与占用率／（1－占用率）（x 轴）间的关系。配置是 4 个 SCP（C）（各有 2 个处理器）和 2 个 SDP（主控机／受控机）（各有 2 个处理器）；

图 12A－12C 说明了混合流量 A 和漏桶的执行情况，其中：

图 12A 示出了携带流量（y 轴）和供应流量（x 轴）（呼叫／秒）的对应关系；

图 12B 示出了装置反应时间（ms）和供应流量速度（呼叫／秒）的对应关系；和

图 12C 示出了 SCP（主控机）cpu 占用率（百分比）和供应流量速度（呼叫／秒）之间的函数关系；

图 13A－13C 说明了混合流量 B 和漏桶的执行情况，其中：

图 13A 示出了携带流量（y 轴）和供应流量（x 轴）（呼叫／秒）的对应关系；

图 13B 示出了装置反应时间（ms）和供应流量速度（呼叫／秒）的对应关系；和

图 13C 示出了 SCP（主控机）cpu 占用率（百分比）和供应流量速度（呼叫／秒）之间的函数关系；

图 14A 说明了绘制原始数据表明了 x 和 y 之间的线性关系；和

图 14B 说明了最小二乘法，该方法用来绘制通过图 14A 中的点的最佳拟合曲线 y=a+bx。

首先将描述本发明的实际案例。

问题涉及银行的副总裁。该银行管理 Sunderland 的客户电话中心，该电话中心覆盖整个国家。副总裁想知道该电话中心如何忙，以及大多数客户得到何种服务，但却不相信该电话中心经理给他准确的信息。有一个好办法获得该信息而不需要离开伦敦的办公室。他可以一天几次打电话到该电话中心，就可以知道他需要等多长时间才能得到回答。即他记下铃声＋讨厌的“请等待”的总秒数，再加上处理业务的时间（核对帐单）。

假如他总需要等待很长时间，那么意味着该电话中心非常忙。假如他总是在第二声铃声得到回答，并且业务处理迅速，那么他可以缩减人员。

可能有时时间长，有时时间短，这意味着该电话中心忙，但不是太忙。

即，如果频繁取样的话，通过该延迟时间将可以算出远程服务器的占用率，而不需要直接进行询问。

问题又变成，客户抱怨在一天的某个特定时间要等待很长时间。

副总裁想控制进入该电话中心的工作量。一个问题是由于每小时的咖啡休息时间造成的停机时间。他安装了一个系统，通过该系统他可以在伦敦的办公室控制接线总机。他决定着手该系统：每次电话进来，他就在桶里放一粒豆子。每次客户挂上电话，他就拿出一粒豆子。假如该桶变满了，他就改变电话，向客户播放“稍后回电”。这就保证了，假如所有电话中心员工都在同一时间休息，也不会使太多的客户不便。

每次电话进来就在桶里加上恒定数量的豆子，当远程服务器故障时，该桶最终会满，那么电话就可被拒绝。

但是该系统只有在完全故障时才起作用。副总裁想要在电话中心占用率太高时拒绝流量（播放“稍后回电”）。

他担心假如占用率接近100％会给员工过大的压力，可能会引起工会干预甚至上法庭。他从先前的试验中得知当铃声加音乐等待时间太长时，占用率就高了。他决定记下在最高法定占用率（70％）时的总时间（等待加服务时间）。他发现时间范围在30秒到5分钟，但是95％的电话不超过2分钟。因此他决定对于每个超过2分钟的电话，他放一粒豆子到桶里。这样，假如电话时间太长，该桶就会满，他就会调用关于拒绝流量的规则。

但是这种假定有一个问题。当该电话中心在或低于法定占用率时，却有5％的电话时间太长怎么办？假如该电话中心的占用率在70％，那么5％的电话(电话时间超过2分钟)将导致一粒豆子进入该桶内(并呆在桶内）。

最后，银行总裁发现该桶满了，按照它的规则，他必须拒绝电话。他决定在某种程度上改变该系统：他实行“漏出”。

有时他从该桶里拿出一些豆子。他计算出漏出速度，简单地说，就是在最大可承受客户访问速度时，豆子进入该桶内的平均速度。假如他将最大可承受速度定为平衡点，在该点上豆子进入该桶的速度等于出去的速度，那么他肯定，只要客户访问率超过最大值，进入的豆子将超过漏出的豆子，那么该桶将会满。更具体而言，他在电话中心有100个低薪工人。他希望在任何时间最多占用70个。他发现在70%占用率下，平均电话持续1分钟，95%少于2分钟。当呼叫速度达到每分钟70个时，出现了70%的占用率。在这样的呼叫速度下，每分钟有3.5个电话长于2分钟。（3.5是70的5%）。规则是每个长于2分钟的电话必须在桶内放一粒豆子以示惩罚。因此这意味着，根据他的规则，当呼叫速度达到70个电话／分钟时，他每分钟放3.5粒豆子到桶里。他平均每分钟从桶里拿出3.5粒豆子，这样该桶保持同样的水平，即十分之一满。

现在呼叫速度增加到80个电话／分钟。雇员们的压力更大并且效率更低了，更多的客户在等着空线，因此平均电话时间上升到1.1分钟，超过20%的电话长于2分钟。占用率上升到80%。但是现在该桶的调节系统开始生效。因为他仍然应用该2分钟惩罚规则，现在他将20%×80个电话／分钟＝16粒豆子放到桶里。因为只有3.5粒豆子漏掉，所以几分钟后该桶就满了（确切的几分钟取决于桶的大小），然后开始拒绝电话。一旦拒绝电话，（接受的）呼叫速度降低了，占用率和电话时间降低了，情况又稳定在70%的占用率。当呼叫速度是50个电话／分钟时又会怎样呢？在该呼叫速度下，平均时间仍然大约为1分钟，但是只有1%的电话长于2分钟。因此只有0.5（50的1%）粒豆子／分钟进入该桶内。但是仍然有3.5粒豆子／分钟漏出。因此该桶大多数时间几乎是空的。

该算法使用了服务＋等待时间来控制流量。延迟次数意味着在桶里加上以示惩罚的豆子。当该桶满了，就拒绝新的流量。桶里有个漏缝，因此当流量不大时不会因为偶然的长时间而将该桶塞满。

还有一个改进。银行副总裁发现该‘所有或者没有’拒绝算法有点太严格了。因此他在桶里设立了一个拒绝区。拒绝的流量比例由拒绝区的比例给定，拒绝区是桶里低于豆子高度的部分。例如，假定该拒绝区是桶的上半部分。当该桶不到半满时，不拒绝任何流量。当该桶

3/4 满时，拒绝 50％的流量。当该桶 90％满时，拒绝 80％的流量。当该桶 100％满时，拒绝 100％的流量。

根据桶里豆子覆盖的该桶的拒绝区的比例来拒绝流量。

现在引进一些更复杂的问题。平均通话时间为 1 分钟的电话服务是一个简单的帐户查询服务。该银行引进新的服务，允许打电话人在电话中申请新帐户，购买外汇，等等。该流量拒绝系统崩溃了，直到意识到可以对电话进行分类才得以解决：

电话类型	平均时间*	95％的电话短于
帐户查询	1 分钟	2 分钟
贷款状态查询	2 分钟	4.5 分钟
申请新帐户	30 分钟	35 分钟

*这些时间都是在 70％占用率下测得的

因此决定，对于每种类型电话，假如该电话长于该种类型电话的 95％时间，就会有一粒豆子进入该桶内以示惩罚。

现在又有麻烦。因为免费（give-away offer），因此有很多人电话申请新帐户。由于平均时间是 30 分钟，因此 3.3 个电话／分钟的呼叫速度就足以占用该电话中心 100 个员工的全部时间。该电话中的所有时间都完全占满了。但是发现该桶却没有满。

一项调查显示每个电话都延迟了，因此 3.3 粒豆子／分钟进入该桶内。但是该桶仍然以 3.5 粒豆子／分钟的速度漏出。该桶仍然几乎是空的，因此不拒绝电话。

这表示在该保护装置中有漏洞，因此必须进行修改。

决定给惩罚加上“加权”因子，因此比较费时的电话比相对较短的电话应给予更大的惩罚。这样选择加权因子，使得对于每个类型的电话，在最大呼叫速度（即当该中心 70％占用时的呼叫速度）情况下豆子进入该桶内的速度与漏出的速度正好平衡。例如，假定 100％的电话都是新帐户申请。那么在呼叫速度为 2.33 个电话／分钟的呼叫速度下，该电话中心将被占用 70％。假如 5％的电话延迟了，那么这意味着：0.05*2.33**惩罚 = 3.5 或者惩罚 = 30。

对于该种类型的每个延迟的电话，必须往桶里放 30 粒豆子。这很有道理，因为该种类型的电话耗费的时间是简单的 1 分钟电话耗费的时间的 30 倍。

预测的拖延次数和惩罚都取决于电话的类型。该惩罚和电话所用的时间成比例。

使用 SCP/SDP 结构的数值仿真来研究一种开关控制漏桶负载控制算法。该算法考虑一次服务内的单独读写次数，以便预测反应时间。假如该请求回到开关控制的时间延迟了，那么将惩罚标志加到计数器（桶）上。假如该桶内豆子的高度超过固定值，则拒绝一定比例的新流量。该比例随着桶内豆子高度升高而上升，当该桶完全满了，则拒绝所有的流量。由于两种原因，该桶周期性地漏出标志：

1） 在流量较低阶段时积累的不真实的标志必须漏出，和

2） 该桶必须能够漏掉由于流量爆发引起的过载。

用图 2 和 3 中所示的结构对该算法进行检验。在图 2 中，该 SCP 和 SDP 存在于单个 Unix 箱中。在图 3 中，几个双 cpu SCP（C）箱通过通信连接向两个多 cpu SDP 箱（主控机和受控机）传送请求。

该单箱 SCP 仿真模型用于测定作为不同类型流量的流量速度的函数的反应时间和处理器占用率。这里示出了怎样使用这些测量方法来建立该漏桶算法的输入数据。

使用表 1 和 2 中给出的软件程序的初步测量方法，写出用于图 3 结构的简单仿真模型。在切断过载的情况下，运行该模型来测定作为不同类型流量的流量速度的函数的反应时间和处理器占用率。

它也用于研究其它问题。特别重要的是 SDP 客户软件程序数量的选择。该数量是可设定的，假如太小，可能严重影响性能。

轮流使用模型运行得到的数据来计算漏桶模型的输入参数，即目标反应时间和不同服务惩罚公式中的系数。该配置数据反馈到该模型内，然后对于不同类型流量，加上过载以后运行该模型，该不同类型流量包括实际流量混合，不同数量的 SCP 和不同数量的 SDP cpu。

使用简单的线性方程可以计算出目标反应时间和惩罚，这里详细描述了用来寻找这些方程系数的方法，包括必须进行的测量和用来得到最合适方程式的技术。在很大程度上，该程序可以是自动的。

该惩罚的线性方程希望能用于所有情况。目标反应时间的线性方程在某些情况下会崩溃，这稍后再谈。

将处理器占用率为期望最大值时的反应时间分布中 95％那一点选取为反应时间极限 T_{target}。当使用术语“处理器占用率”时，指的是在

单箱结构中的总占用率和多箱结构中 SDP cpu 占用率。即，假定期望的最大占用率为 66%。

当 cpu 占用了 66%，95% 的反应时间小于 T_{target}。例如，在图 4 中，示出了在 SDP 仿真模型上以 220 个电话／秒运行的一次读服务的反应时间的柱状图，95% 时间大约为 120ms。这接近 T_{target}。该图表实际上示出了对应于 70% cpu 占用率的流量速度下的反应时间柱状图。

使用试验和误差来寻找占用率正好为 66% 的流量速度是很困难的。

这里示出了以下公式，作为占用率的函数，通过该公式可以计算出反应时间 $T_{response}$：

$$T_{response} \cong T_0(n_r, n_w) + T_q(n_r, n_w) \times \text{占用率} / (1 - \text{占用率})$$

其中 $T_0(n_r, n_w)$ 和 $T_q(n_r, n_w)$ 取决于读和写的次数，并且必须测量。

第三种类型是失败的读或写。尽管在算法中包括了该种类型，但是在以下讨论中省略了和 n_r 的关系。在以上公式中为了得到 T_{target}，这里将对应于期望 cpu 占用率的占用率设定为 66%。简明地指出，对于有效数据（见图 5），可以得到很好的结果，而不需要在算术上调整该公式。稍后将作详细描述。

假如加到桶里的惩罚标志数量与服务无关，那么在维持前后一致过载的策略上就会出现明显的问题。考虑以下例子，一次读服务假如反应时间大于 142ms，则要放入 100 个惩罚标志。当呼叫速度对应 66% 的 cpu 占用率时，反应时间的 5% 将大于此。该流量速度为 202 个电话每秒。在 202 个电话／秒的情况下，惩罚标志进入桶里的速度为 $0.05 \times 202 \times 100 = 1000$ 个标志／秒。那么假定该桶的漏出速度为 1000 个标志／秒，因此在该流量速度的情况下，该桶的标志高度几乎恒定－每秒钟，进入桶里的标志等于漏出桶的标志。假如流量速度上升，那么该桶就会满，将会切换到过载拒绝算法，并且开始拒绝流量。

现在设想仅仅包含 3 次读服务的流量。对应 66% cpu 占用率的流量水平是大约 91 个电话／秒。按此速度，假如使用同样的“每次延迟反应 100 个惩罚标志”的规则，则标志进入桶内的速度将约为 500 个每秒。但是该桶的漏出速度设为 1000 个每秒。在该情况下，该桶漏出的速度是填充速度的两倍，那么即使流量水平处于最高状态，该系统也几乎不可能过载。

因此，加进桶里的惩罚标志的数量必须和该服务的 cpu 费时成比例－对于每个延迟反应来说，三次读服务必须放入比一次读服务更多的惩罚标志。

标志大小的公式为：

漏出速度／（最大占用率下的呼叫速度×0.5）＝标志大小

以下假定漏出速度为 1000 个标志／秒。

表 1 示出了对 SCP 仿真模型上 n_r（n_r＝1，...5）次读服务运行的分析。

读服务数	使用的点	T_0	T_q	66％时的 T_{target}	费时(ms)	R_{T66}呼叫／秒	标志大小
1	8	50.2	28.61	107.4	6.61	202	100
2	13	42.43	51.59	145.6	10.57	126	160
3	10	43.69	73.18	190	14.57	91.45	220
4	7	42.74	97.15	237	18.58	71.72	280
5	7	29.5	143.47	316	22.78	58.51	340

表 1

这里 R_{T66} 是在 66％占用率下的流量速度，通过以下公式得出所费时间：

费时＝cpu 数×1000ms／秒×占用率／（以呼叫／秒表示的呼叫速度）

所费时间以毫秒为单位。

正如所希望的那样，该表格显示出 T_0 大致上是恒定的，（忽略 5 次读取的数值）平均为 45ms，而 T_q 是读取次数的线性函数。根据数据的最优接近（通过稍后说明的最小二乘法计算得到），得出 $T_q \cong 27.5n-3.8$。但是，查看图 6 和表 1，可以看出 5 次读服务点似乎有点与众不同，T_q 似乎太高而 T_0 似乎太低。假如该 5 次读服务点不包括在内，那么余下来的 4 点和公式 $T_q = 22.7 + 5.8$ 几乎吻合。

找到通过公式来计算目标反应时间 T_{target} 的方法是我们的一个目标。对于任意数量的读取和 cpu 占用率，计算 95％反应时间的公式是：

$T_{response}$（n，占用率）＝45＋（22.7n＋5.8）×占用率／（1－占用

率）

假定66％的占用率，那么得到表2。在该表中，对于n_r＝1，2，3，4，5，使用公式T_{target}（n）=45+2(22.7n+5.8)计算出的结果和上表中的T_{target}进行了比较。[注意，占用率＝66％，因此（1－占用率）／占用率＝2]

n	45+(22.7n+5.8) × 2	T_{target}
1	102	107.4
2	147.4	145.6
3	192.8	190
4	238.2	237
5	283.6	316

表2

尽管该公式的预测能力不差，但是对于n_r=5该公式似乎有很大误差。一个简单的线性公式一般来说似乎很好，但是不足以用来精确计算所有服务的反应时间极值，必须分别测量其目标反应时间。对于多箱案例，以下部分示出了类似的结论。

假如用n_w次写服务来代替n_r次读服务进行检查，以上分析应当同样正确。即，即使写服务比读服务花费较多时间，但是呼叫速度，占用率和95％反应时间之间应该有相同的函数关系。

这里，示出一次“长读”服务的SCP仿真得到的一些数据。为了这些目的，其被称作写。

对于一次，两次和三次写入，该模型以几种不同的流量速度运行。然后分析该数据，以求出漏桶算法的系数。这样做去除了高占用率下的数据点，以得到合理的线性拟合（图7）。

以上分析完全取决于呼叫速度、占用率和反应时间之间存在着某种特定关系。可以看出该SCP仿真数据非常符合这些关系。可以有趣地看到在SCP上采集到的实际数据是否和模型具有相同类型的关系。这里，在单箱SCP模型上采集的一些测量结果给出了三种不同服务的图表（图9）。该数据是对于一次读（基准5）和一次读一次写服务（基准C）的。第三种服务是对于包含12次读和1次写的长时间且昂贵的服务。下面给出了该数据。图中示出，在占用率和呼叫速度之间确实

存在线性关系，在变量 x = 占用率 /（100 - 占用率）和 95% 反应时间之间也存在线性关系。

呼叫速度（cps）	占用率（%）	95% 反应时间（ms）
14. 048	60	501
14. 88	63	538
17. 066	67	558
17. 959	71	684

表 3

考虑图 3 中示出的结构，电话到达 SCP（C）并被传送到 slp（服务逻辑程序）。

假定该电话需要两次数据库访问：一次读和一次写。该 slp 将该读请求分配到下一个 sdp 客户进程，该进程标号为 k，该请求加入行列 k。该 sdp 客户进程将请求传送到一个 sdp 箱，在主控机和受控机之间交替。

该请求经过软件程序 rcv，$sdb\text{-}serv_k$，$slp\text{-}dba_k$ 和 Informix 数据库访问程序 $oninit_k$。为了仿真，假定除了数据库访问外，每个软件程序占用固定数量的 cpu 时间，这里假定为 Poisson 分布时间。

当然，当程序必须排队等候 cpu 时间时，该请求就会招致排队等候延时。

在 SCP/SDP 模型上进行各种软件程序计时，并将其在下面表 4 - 6 中示出。这是根据一次读服务和一次读 / 一次写服务运行的。一次写服务的计时是考虑其不同之处推断出来的。

SCP（C）	i-sw	o-sw	slp	slp-客户机	rcv	总时间
1 次读 / 0 次写	0.85	0.988	1.45	0.91	0.42	4.618
1 次读 / 1 次写	0.85	0.988	2.20	1.85	0.849	6.737
（0 次读 / 1 次写）	0.85	0.988	1.45	0.94	0.42	4.648

表 4

SDP（主控机）ms	Rcv	sdb-serv	Dba	oninit	总时间
1 次读 / 0 次写	0.42	0.77	0.86	1.14	3.19
1 次读 / 1 次写	0.85	1.81	3.03	9.8	15.49
（0 次读 / 1 次写）	0.42	1.04	2.17	8.66	12.29

表 5

SDP（受控机）ms	rcv	sdb-serv	dba	oninit	总时间
1 次读／0 次写	0.36	0.81	0.89	1.59	3.65
1 次读／1 次写	0.89	1.73	3.14	12.3	18.06
（0 次读／1 次写）	0.53	0.92	2.25	10.71	14.41

表 6

为了计算该桶的输入参数，对于包含 1，3，5 和 8 次读的服务，在不同的流量速度下运行该模型。在图 10 中示出了 95％的反应时间和 X = 占用率／（1－占用率）的函数关系图。

注意，一次读的线性拟合不是很好。原因如下：在图中，占用率是 SDP 的 cpu 占用率。即，假定延迟从排队等候 SDP 上的 cpu 时间开始。但是，对于一次读服务来说，在 SCP（C）上一个电话花费 cpu 时间 4.618ms，在 SDP 上花费 3.19ms，这意味着双箱且每箱 2 个 cpu 的 SCP（C）上的占用率将大于双箱且每箱 2 个 cpu 的 SDP 上的占用率。因此，对于大流量来说，最显著的延迟来自在 SCP（C）上排队等待 cpu 时间，而不是来自在 SDP 上排队等待 cpu 时间。这种影响使得寻找反应时间的最合适公式的程序预测结果有所偏离。使用图 9 中显示的数据（除了来自写过程的数据）得到的该公式的 Mathematica 生成表格和预测结果如下（Mathematica 是由 Wolfram 研究有限公司拥有的软件程序注册商标）：

TR95 的公式为 TR95 = 69.733+1.85971*n 次读

惩罚标志 token 大小的公式为 token = 0.327565+23.8191*n 次读

得到表格：

读取数	T_95（测得）	T_95（预测）	标志（测得）	标志（预测）
1	98.7617	71.5927	23.9156	24.1467
2	…	73.4524	…	47.9658
3	44.0841	75.3121	71.926	71.7849
4	…	77.1718	…	95.604
5	67.6836	79.0315	119.491	119.423
6	…	80.8912	…	143.242
7	…	82.7509	…	167.061
8	100.071	84.6106	191.065	190.88

9	...	86.4703	...	214.699
10	...	88.33	...	238.519

T_95（T_{target}）的预测不太好。假如从输入中去除数据集读取，则表格如下：

T_{target}的公式为 T_{target} = 11.1049+11.1544*n 次读

惩罚标志 token 大小的公式为 token = 0.850048+23.7326*n 次读

得到表格：

读取数	T_{target}（测得）	T_{target}（预测）	标志（测得）	标志（预测）
1	...	22.2593	...	24.5826
2	...	33.4137	...	48.3151
3	44.0841	44.5681	71.926	72.0477
4	...	55.7225	...	95.7802
5	67.6836	66.8769	119.491	119.513
6	...	78.0313	...	143.245
7	...	89.1857	...	166.978
8	100.017	100.34	191.065	190.71
9	...	111.494	...	214.443
10	...	122.649	...	238.176

T_{target}的公式为 T_{target} = 16.212+53.5948*n 次写

惩罚标志 token 大小的公式为 token = 2.38595+90.592*n 次写

得到表格：

写入数	T_{target}（测得）	T_{target}（预测）	标志（测得）	标志（预测）
1	69.3647	69.8067	91.627	92.9779
2	124.065	123.402	184.736	183.57
3	...	176.996	...	273.62
4	230.37	230.591	365.6	364.754

系数为：

k0=1.618

kr=23.7326

kw = 90.592

以下公式是近似的，因此可能需要进行细微调整

Tr（nr）=11.1544*nr

Tw（nw）=53.5948*nw

Cr=11.1049

Cw=16.212

（写数据包括在第二打印输出中。）很明显，假如该程序不使用1次读测量，则其它服务的预测会好很多。解决该问题的一个方法是将一次读服务作为特殊情况来对待，“手工”填入从测量值得到的目标反应时间，并使用Mathematica最小二乘法拟合程序给出n_r次读的目标反应时间（$n_r>1$）公式。那么这里，将使用目标值。

另一方面可证明，SCP决不允许比SDP占用更多，因此假如该流量的相当一部分是一次读服务（象一些装置那样），则应当加上更多的SCP箱。这样做的自然效果将是在相同的呼叫速度下减轻每个SCP（C）上的cpu负载。在4个SCP（C）箱的设备上重新运行该测试数据。这次运行，没有出现一次读服务的特别问题：图11示出了很好的线性拟合，Mathematica输出给出

T_{target}的公式为$T_{target}=6.18404+11.5085*n$次读

惩罚标志token大小的公式为token=0.511096+11.7688*n次读

得到表格：

读取数	T_{target}（测得）	T_{target}（预测）	标志（测得）	标志（预测）
1	18.1253	17.6925	11.9921	12.2799
2	…	29.201	…	24.0487
3	40.1035	40.7094	35.9785	35.8174
4	…	52.2179	…	47.5862
5	…	63.7264	…	59.355
6	…	75.2348	…	71.1238
7	…	86.7433	…	82.8925
8	98.4249	98.2518	95.3493	94.6613
9	…	109.76	…	106.43
10	…	121.269	…	118.199

可得出以下结论：假如一个服务在SCP上的费时大于SDP上的费时，那么线性最小二乘法拟合就不正确，必须特别注意。

运行一个例桶，其中有一个具有2个SCP（C）和主控机－受控机SDP的多箱结构。每箱有2个cpu。从最后部分，建立该桶参数：

$T_{target} = 11.15 \times n_r + 53.60 \times n_w + (11.10 \times n_r + 16.21 \times n_w) / (n_r + n_w)$ 和

惩罚 $= 0.16 + 2.4 \times n_r + 9.06 \times n_w$

该流量为以下的呼叫混合（callmix）：

读取数	写入数	百分比
1	0	60%
2	0	5%
3	0	25%
4	0	10%

表 7

读取数	写入数	百分比
2	0	73%
12	1	22%
9	1	5%

表 8

在图 12 和图 13 中示出了结果。执行／供应（carried/offered）曲线遵循由 ITU 推出的模型。随着流量的上升，反应时间迅速上升，直到开始拒绝流量，这时达到最大值大约 400 个电话／秒并开始下降。SLP 占用率上升到最大值，随后保持大致恒定。

对（单箱）SCP 模型的仿真和运行都显示出以下公式是正确的（注意，这里包括了公式中的失败次数 n_f，以便和 SCP 上编写的算法相一致）：

$$T_{target}(n_r, n_w, n_f) = T_r(n_r) + T_w(n_w) + T_f(n_f) + C(n_r, n_w, n_f)$$

其中 T_{target} 是当处理器在规定的最大占用率时的 95% 反应时间，n_r 是服务中的读取数，n_w 是写入数而 n_f 是失败的读取或写入数。在大多数情况下，T_r，T_w 和 T_f 可以分别表示为 n_r，n_w 和 n_f 的线性函数。即：

$$T_r(n_r) \triangleq t_r \times n_r,\ T_w(n_w) \triangleq t_w \times n_w,\ T_f(n_f) \triangleq t_f \times n_f$$

其中 t_r，n_r 和 n_f 是常数，稍后可以用替代公式计算。$C(n_r, n_w, n_f)$ 为“平均截距”（应该是恒量或者近似恒量）：

$$C(n_r, n_w, n_f) = (n_r c_r + n_w c_w + n_f c_f) / (n_r + n_w + n_f)$$

其中 c_r，c_w 和 c_f 稍后也可用最小二乘法求出。在以下两种情况下，目标反应时间和读取数和写入数之间似乎不遵循线性关系：

1）当流量包含单箱结构上的读或写的平均数量很大的电话时。

2）当流量包含多箱结构上的读的平均数量很小（接近1）的电话时。

在这两种情况下，反应时间似乎比由公式预测出的结果大，因此可能需要一些“手工调整”。

如上所述，延迟反应后留在桶里的惩罚标志的大小必须取决于服务。惩罚的公式为：

$$惩罚 = n_r k_r + n_w k_w + n_f k_f + k_0$$

其中 k_r，k_w，k_f 和 k_0 可以通过使用以下程序中的替代公式根据 SCP 模型从测量值得到。

Mathematica 程序把包含呼叫速度、占用率和 95% 反应时间的读取 1，读取 2，…，写入 1，写入 2，…作为输入数据文件，该程序同样也可以用 C 来编写。例如，文件读取 3 具有以下形式：

50	11.74	22.1
100	23.635	23.60
150	35.739	24.20
200	48.028	30.7
250	59.807	38.4

这是 2 次读服务的几种不同运行的呼叫速度，cpu 占用率和 95% 反应时间。然后，对于每种服务，该程序根据 T_{target} 公式计算该服务在期望的最大占用率（maxocc）下的惩罚和 95% 反应时间。然后，使用这些值来寻找 T_{target} 和惩罚的最佳拟合线性公式。具体地说，它给出了 k_0，k_r，k_w 和（线性近似）T_{target} 公式。在大多数情况下，由该方法给出的参数可以直接进入 SCP 的配置数据。

使用最小二乘法分析 SCP 数字模型运行数据，以便寻找所述最佳拟合曲线。

考虑 n 个数据点

$$(x_1, y_1), (x_2, y_2), \ldots, (x_i, y_i), \ldots, (x_n, y_n),$$

假定变量 x 和 y 具有线性关系 $y = a + bx$，其中 a 和 b 是常数。在测量中易受误差影响的各个数据点，不一定都落在一条直线上，而是有少量的随机分布（参看图 14）。

最佳拟合曲线，即使得各点到该曲线的距离的平方和最小的拟合曲

线由 y = a + bx 给出，其中 a 和 b 为：

$$a = \bar{y} - b\bar{x}, \qquad b = \frac{\sum_i x_i y_i - n\bar{x}}{\sum_i x_i^2 - n\overline{x^2}}$$

其中划线的量为 x 和 y 值的一般平均数。

反应时间 T_R，即发出数据库请求消息和接到答复消息之间的时间，可以分成三部分。第一部分是请求／反应来回传送的固定时间，第二部分是数据库访问所花费的（变动）时间。可以假定访问时间遵守固定平均数的泊淞（Poisson）分布。第三，还有排队延迟：假如该数据库在某时处理超过一个电话，则当处理排在前面的请求时，后面的请求必须等候。假如这三个延迟是 T1，T2，T3，则：

（平均反应时间）$= T_{(MR)} = T_1 + T_2 + T_3$

T_1 和 T_2 是恒量，且可以测量到。重要之处在于估算 T_3，这取决于队列的长度。

考虑一种通常的排队系统。客户到达，排队等候，然后接受服务。如果假定有速度为 λ 的到达 Poisson 分布和速度为 μ 的服务时间 Poisson 分布，那么排队等候时间为

$T_{queueing} = \{1/\mu\}\ \{\lambda/\mu/(1-\lambda/\mu)\}$

但是，对于 Poisson 分布流量来说，速度 λ 和 μ 分别等于到达之间的时间和服务时间的倒数。因此，该公式变成

$T_{queueing} = T_s\ \{T_s/T_{ia}/(1 - T_s/T_{ia})\}$

对于该系统来说，T_s 是访问一个数据库的平均时间，而 T_{ia} 则是请求到达之间的时间。假定该处理器只进行数据库访问而实质上不进行其它操作，则 T_s/T_{ia} 仅仅是处理器工作时间的的比例…，即占用率。因此排队等待时间是

$T_3 = T_{queueing} = T_s$ 占用率/（1 - 占用率）

假如 $T_1 + T_2 = T_0$，且用 T_q 代替 T_s，则公式为：

$T_R = T_0(m,n) + T_q(m,n) \times$ 占用率／（1 - 占用率）

尽管要对平均延迟求导数，但是对于 95％延迟来说，这也是正确的，至少是近似正确的。

这里详细地说明了对于一个给定的服务，如何计算惩罚标志的大小。令期望的最大 cpu 占用率为 maxocc（在上述例子中设为 66％）。

对于个别服务来说，对数据进行最小二乘法分析，从而估算出对应于cpu占用率为maxocc的呼叫速度。（注意，占用率不仅仅和呼叫速度成比例，因为即使在有电话时还有后台任务占用该cpu。）

从最初公式看出，该惩罚标志和呼叫速度成反比。因此，

（n次读的惩罚）/（1次读的惩罚）=（1次读的最大呼叫速度）/（n次读的最大呼叫速度）。

该惩罚同样可以根据服务的“费时”表达，即cpu执行该服务所花费的时间，以毫秒为单位。很明显，呼叫速度、费时和占用率通过以下公式联系起来：

呼叫速度×（费时／1000）=占用率／100

其中占用率以百分比表示（在0.0到100.0之间）。通过上述公式，得到

惩罚=漏出速度×最大占用率／100×（费时／1000）

其中费时以毫秒为单位，而最大占用率在0.0到100.0之间。很明显，费时是服务的读取数n_r的线性函数，给出

$$惩罚 = k_0 + k_r n_r$$

其中k_0和k_r是常数，可以测量得到。从上述公式得到，k_r是

k_r=漏出速度×（最大占用率／100）×（1次读取／1000）

k_0（大约）是根据不访问数据库的服务费时计算出来的惩罚。不需要直接测量k_s，可以根据不同服务的测量或者占用率与呼叫速度之比计算出来。

以上主要描写了具有n_r次读的服务，但是该说明可以延伸到更普遍的服务，可以包含读取，写入和失败。最终公式变成

$$惩罚 = n_r k_r + n_w k_w + n_f k_f + k_0$$

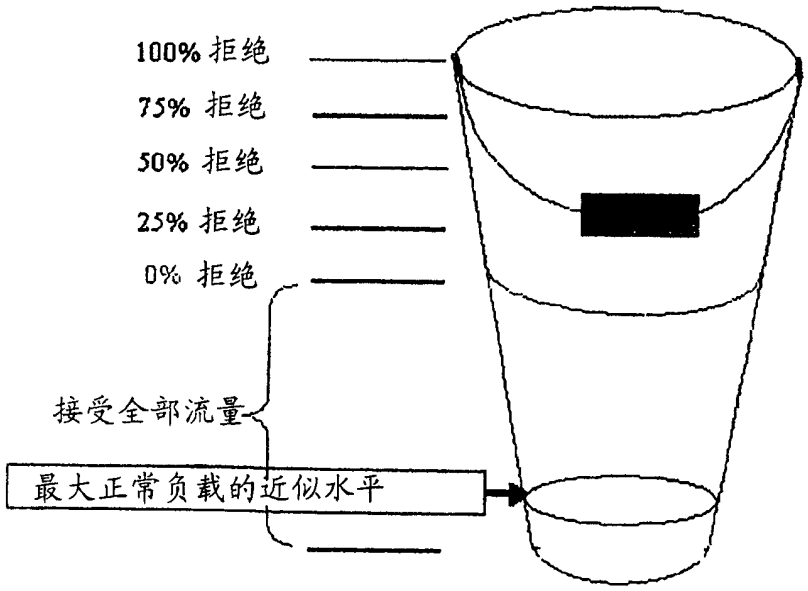


图 1

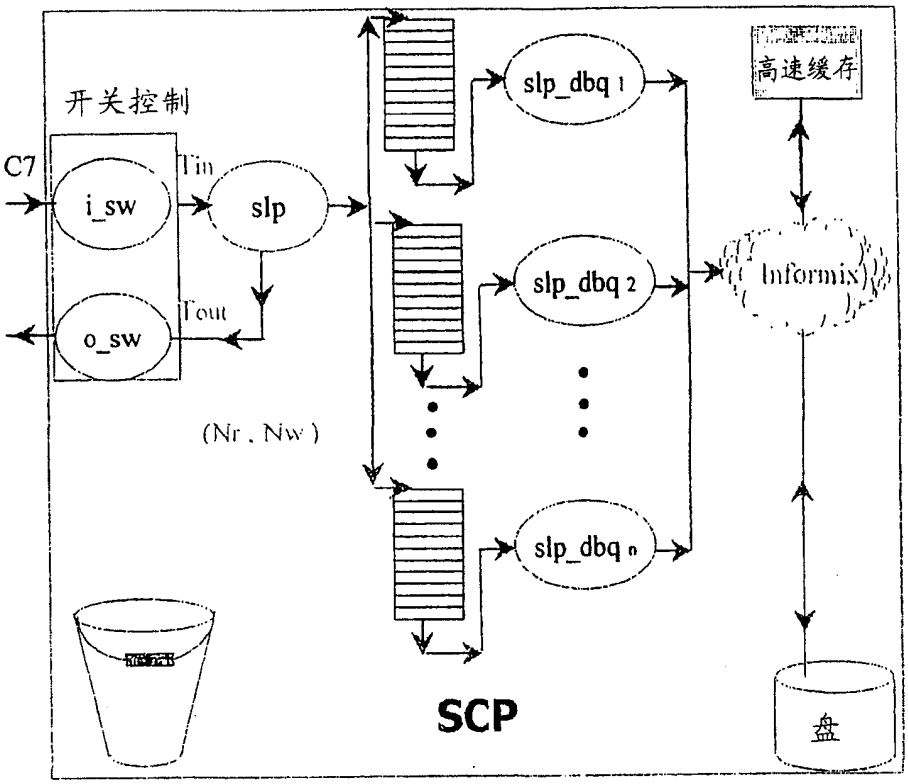

开关控制
C7
i_sw
Tin
slp
Tout
o_sw
(Nr、Nw)
slp_dbq 1
slp_dbq 2
slp_dbq n
高速缓存
Informix
盘
SCP

图 2

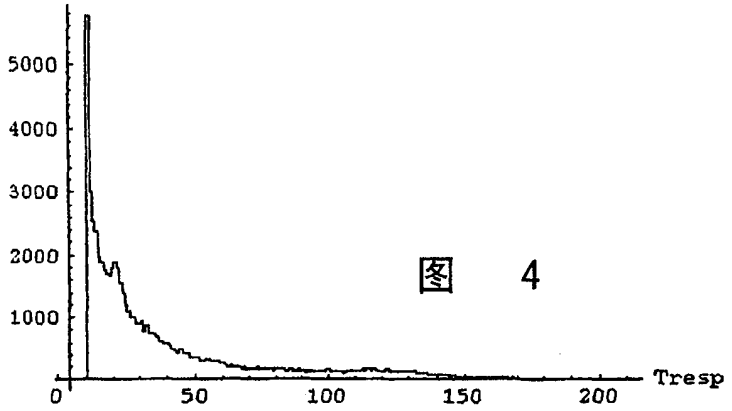

5000
4000
3000
2000
1000
0
50
100
150
200
Tresp

图 4

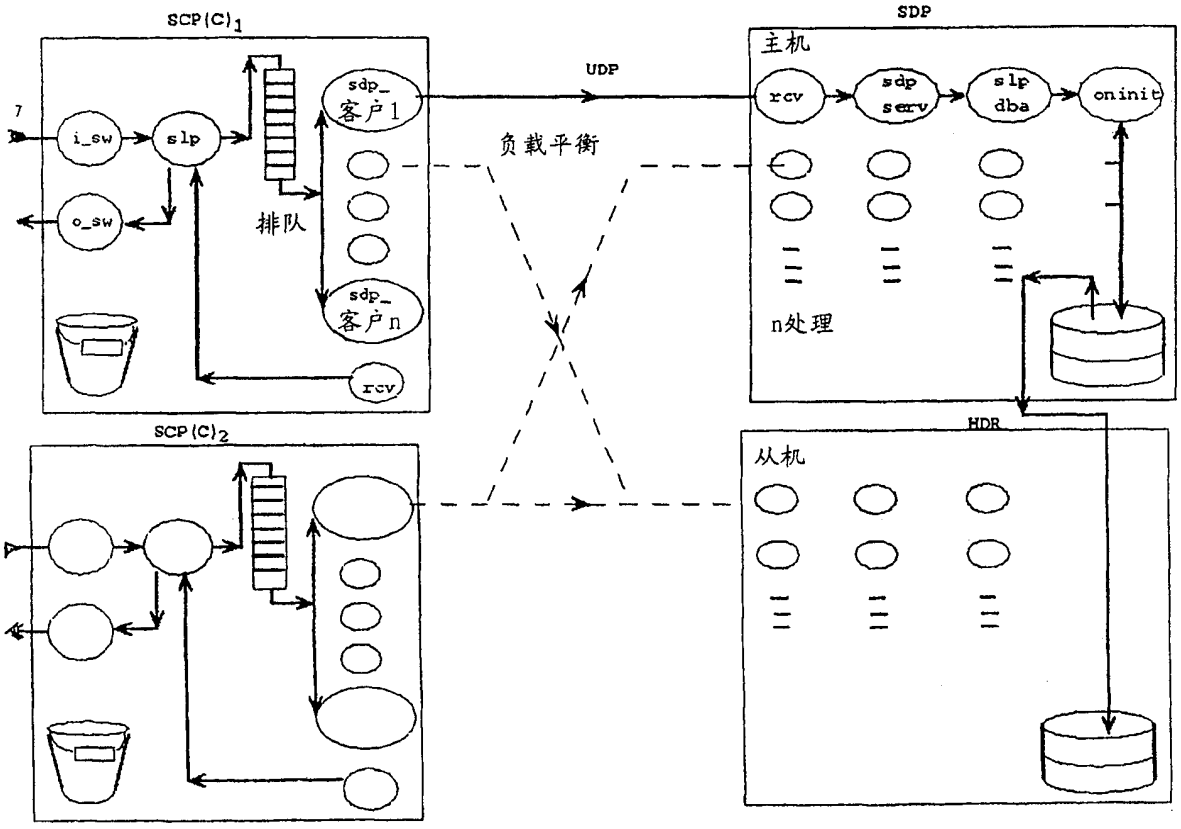

SCP(C)1
SDP
主机
UDP
sdp_
客户1
i_sw
slp
o_sw
排队
负载平衡
rcv
sdp
serv
slp
dba
oninit
sdp_
客户n
n处理
rcv
SCP(C)2
HDR
从机

图 3

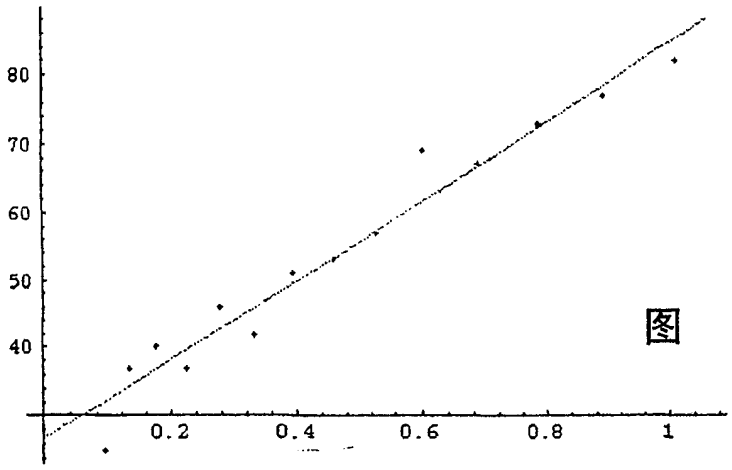


图 5

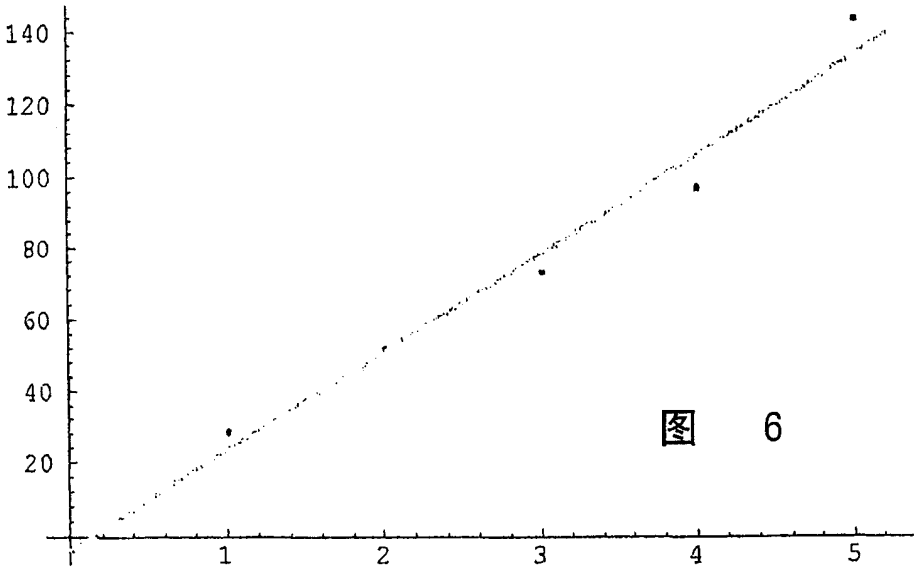


图 6

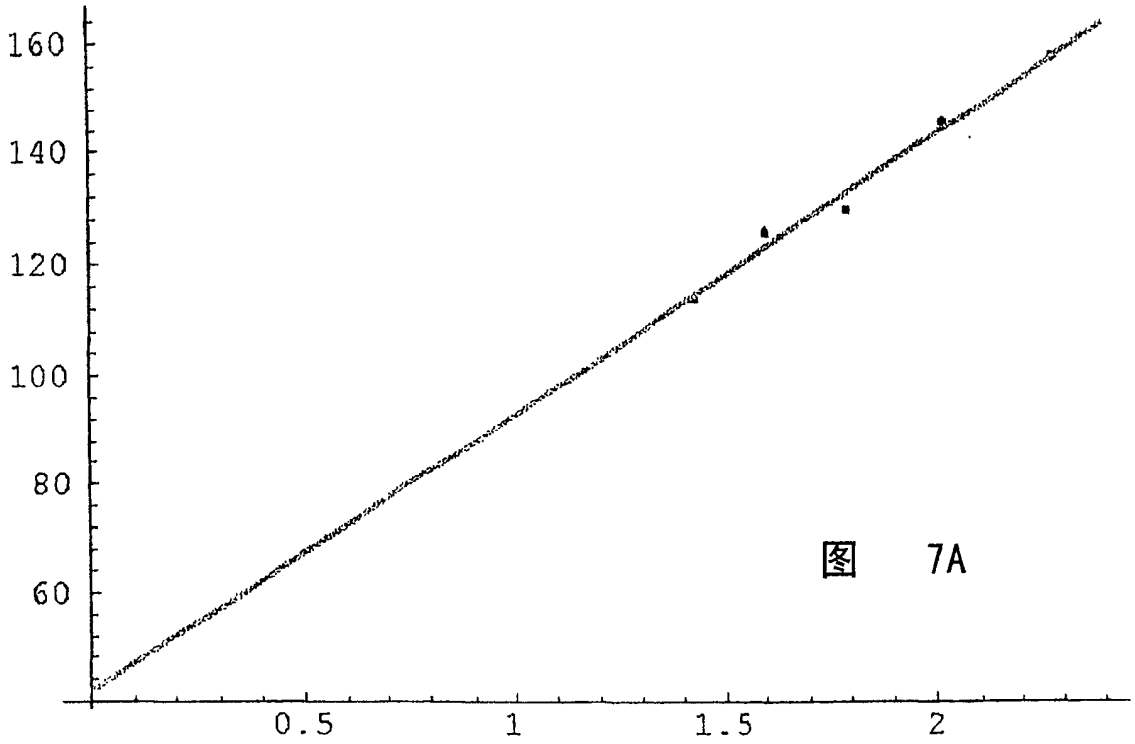


图 7A

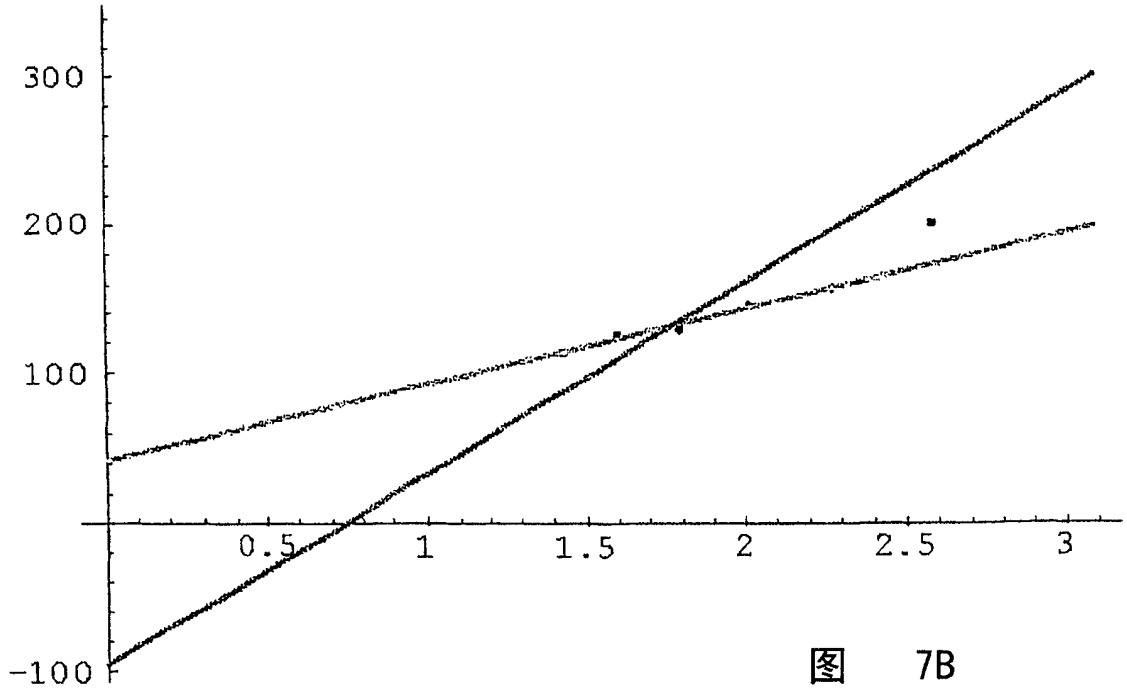


图 7B

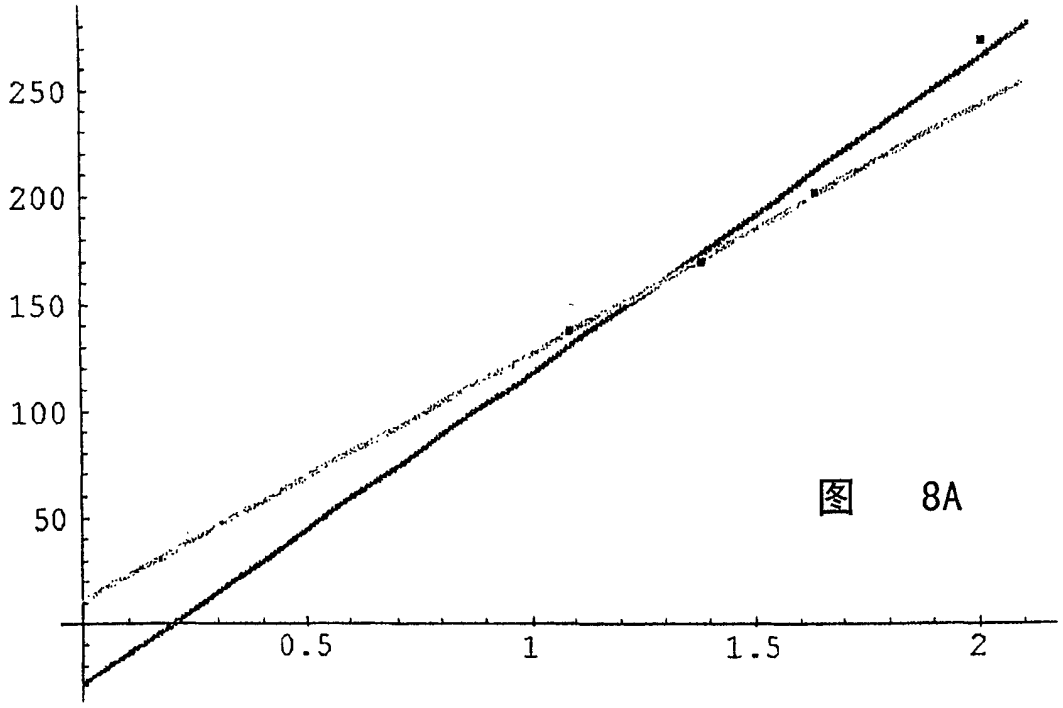


图 8A

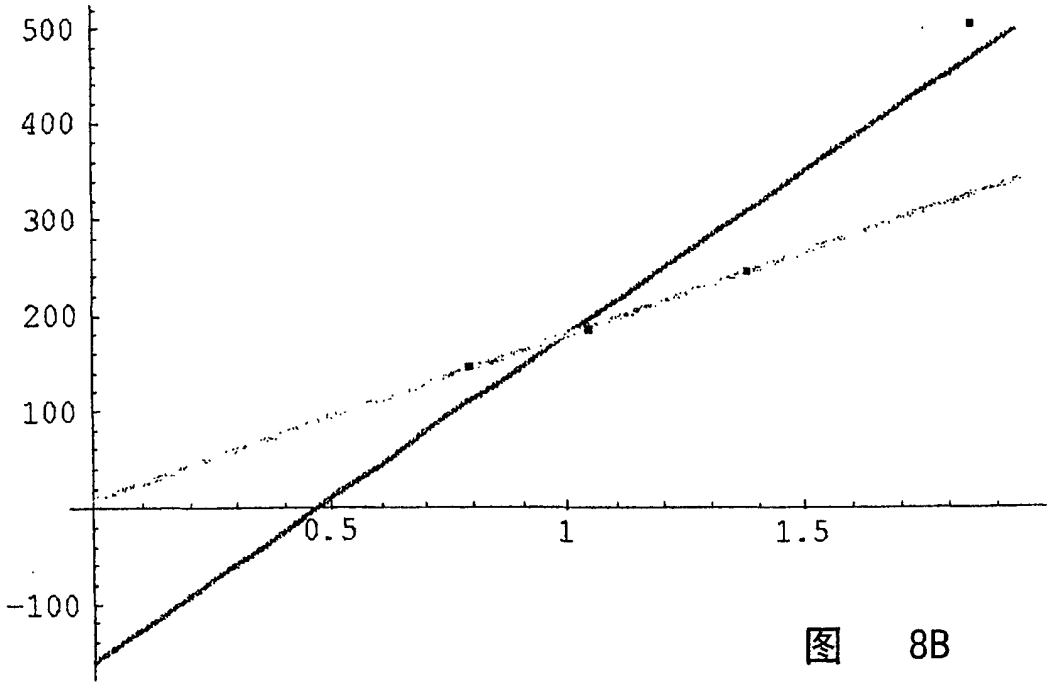


图 8B

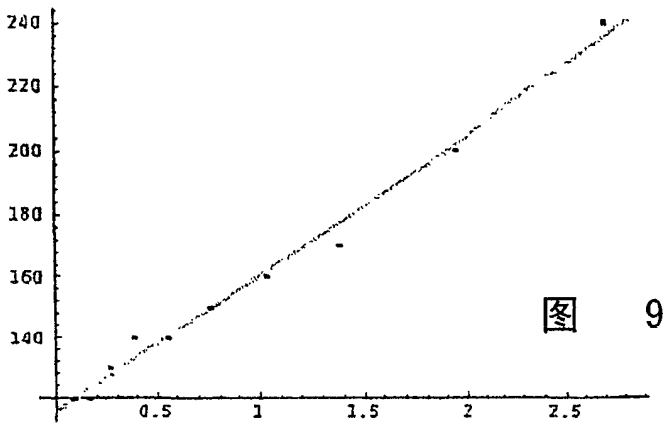

240
220
200
180
160
140
0.5
1
1.5
2
2.5

图 9A

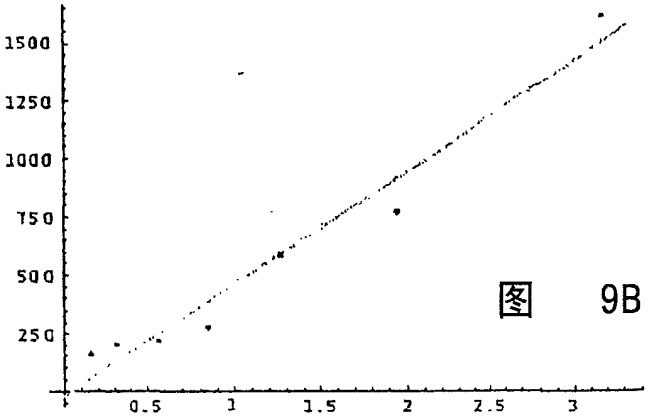

1500
1250
1000
750
500
250
0.5
1
1.5
2
2.5
3

图 9B

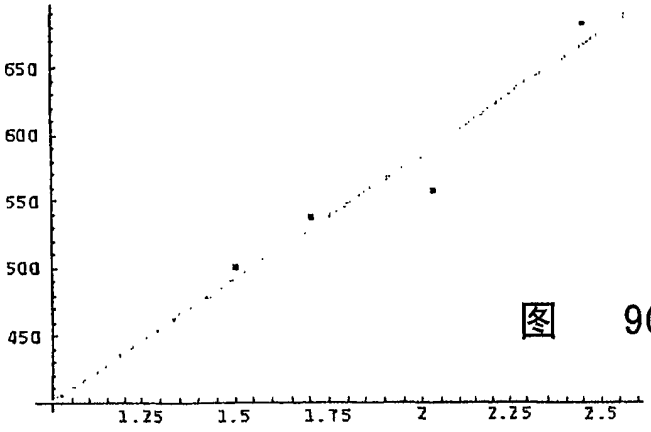

650
600
550
500
450
1.25
1.5
1.75
2
2.25
2.5

图 9C

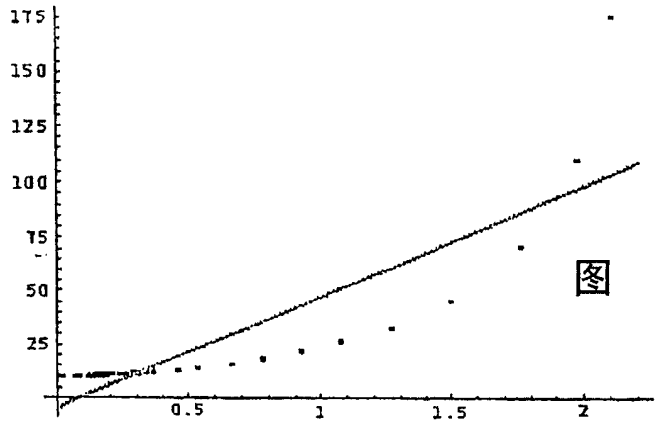


图 10A

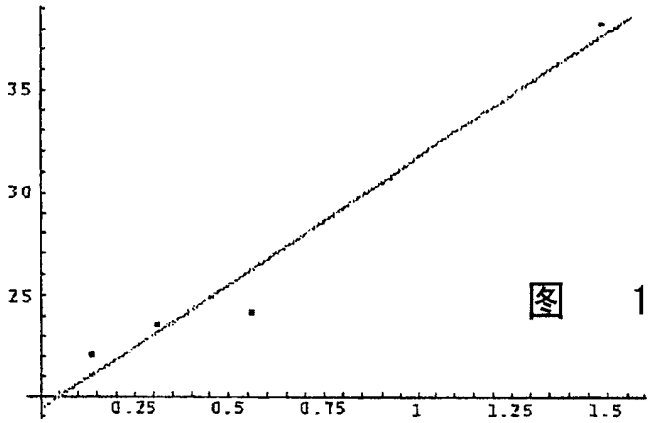


图 10B

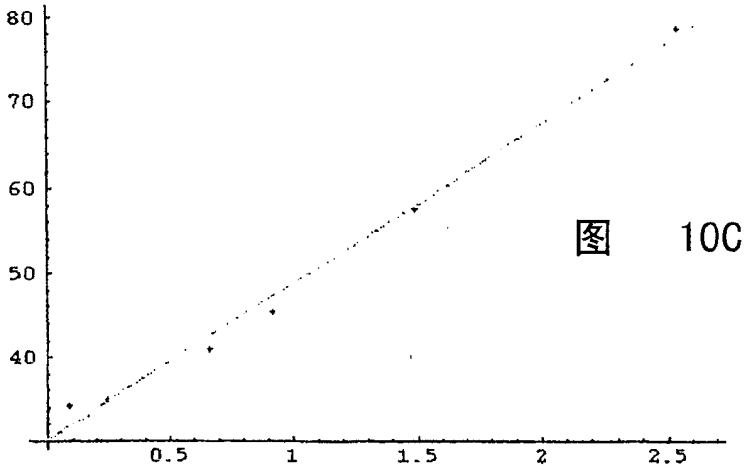


图 10C

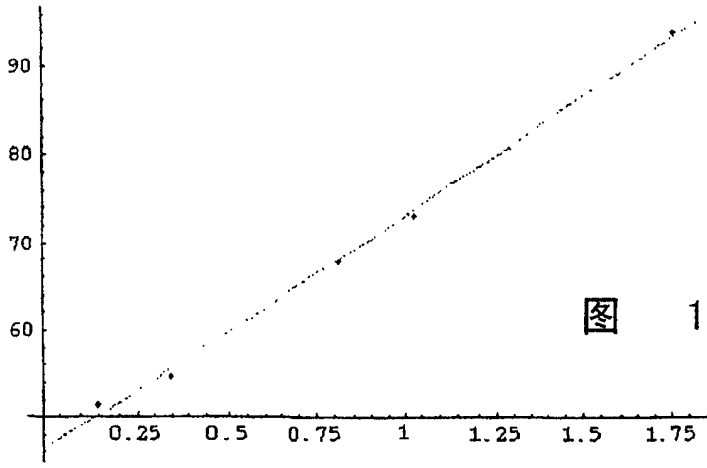


图 10D

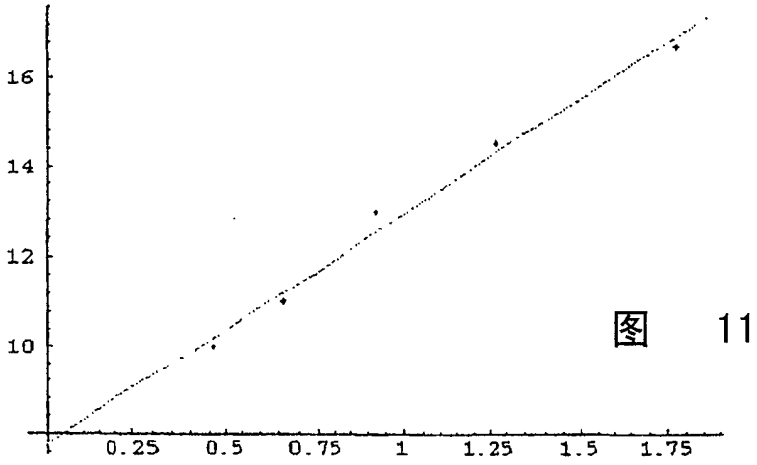


图 11

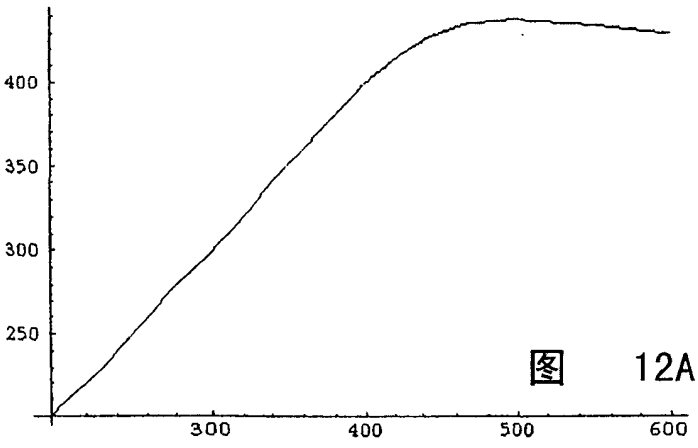


图 12A

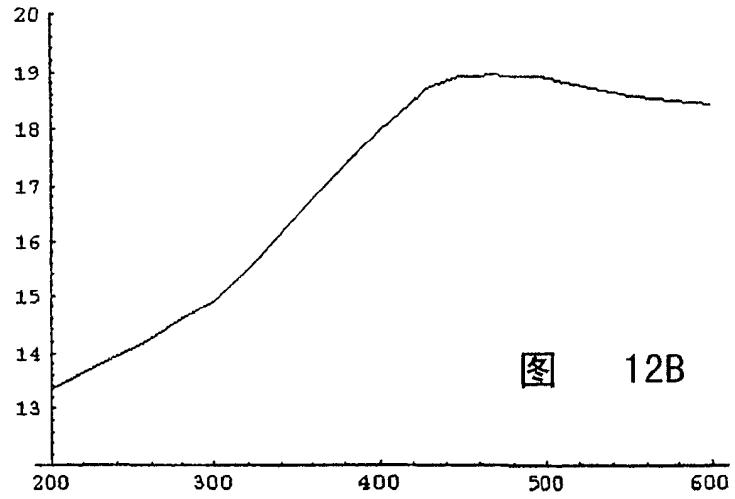


图 12B

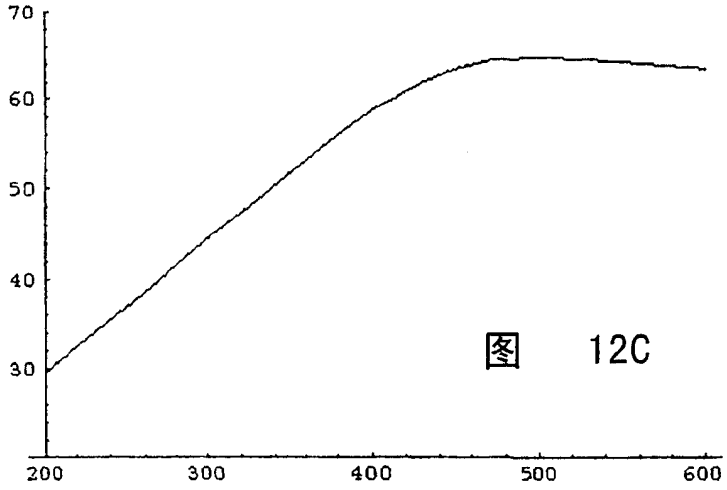


图 12C

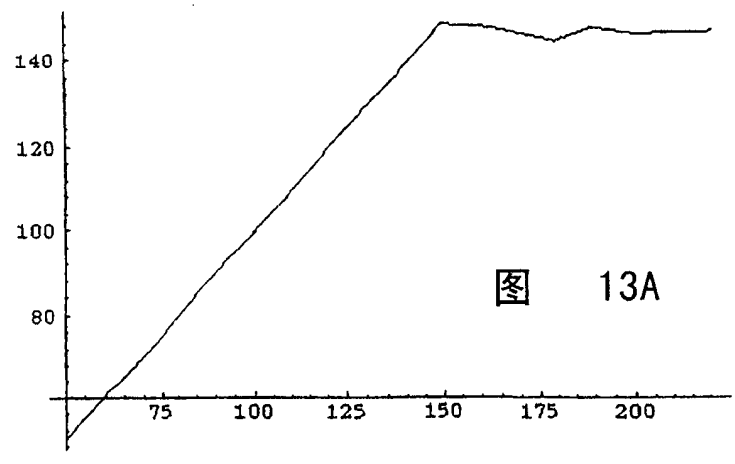


图 13A

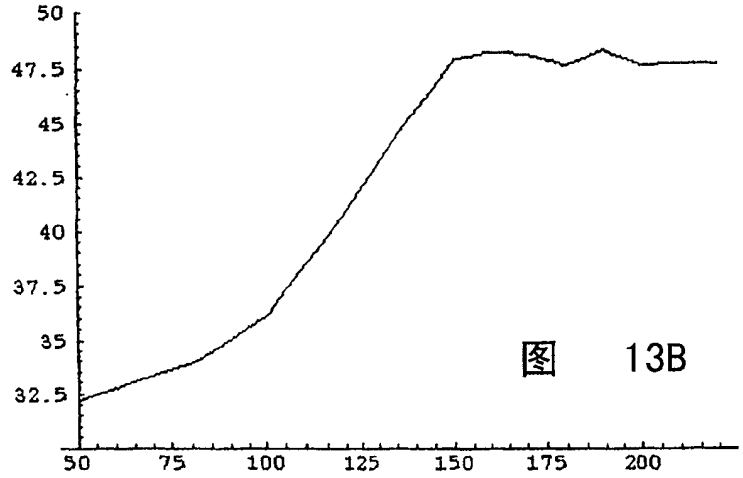

50
47.5
45
42.5
40
37.5
35
32.5
50
75
100
125
150
175
200

图 13B

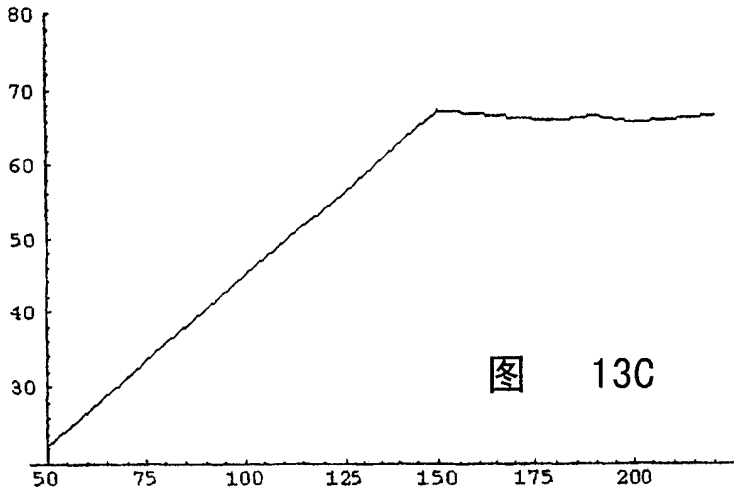

80
70
60
50
40
30
50
75
100
125
150
175
200

图 13C

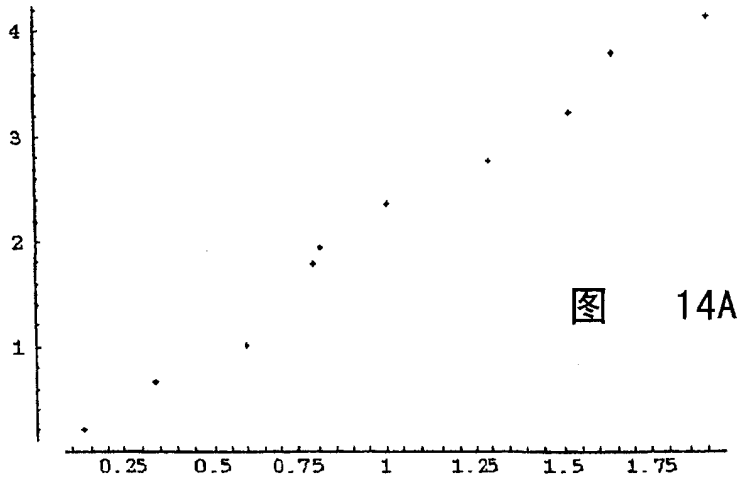


图 14A

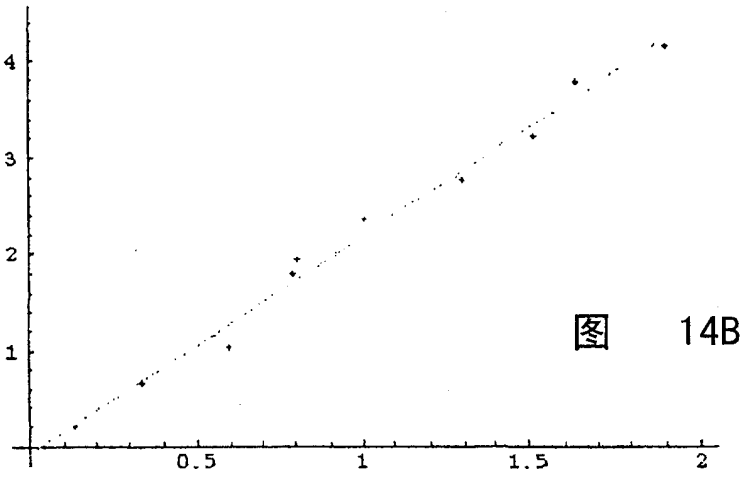


图 14B