文章编号：1005-8451（2015）06-0026-05

基于内存数据库提升铁路货车追踪应用性能的研究

颜昌盛，范娟娟，海　洋，高明星
（中国铁路信息技术中心，北京　100844）

摘　要：介绍铁路货车追踪应用的背景及当前存在的性能问题，针对实际情况提出利用内存数据库解决性能问题的思路，并进行了POC测试。测试结果证明，采用内存数据库解决当前货车追踪应用面临的性能问题，无论是从前期投入还是从后期维护角度看，都具有可行性。

关键词：内存数据库；货物运输；货车追踪
中图分类号：U294：TP39　**文献标识码：**A

Application performance of railway freight car tracking based on in-memory database

YAN Changsheng, FAN Juanjuan, HAI Yang, GAO Mingxing
(China Railway Information Technology Center, Beijing　100844, China)

Abstract: This paper introduced the background of the railway freight car tracking application and its current performance problem. Based on a comprehensive analysis, a totally new method of using the in-memory database was put forward to solved the problem, the POC test was done afterwards. The test result proved that it was practicable to use the in-memory database to improve performance and reduce cost, from the point of new, whether the early stage investment or the later stage of maintenance, the method was feasible.

Key words: in-memory database (IMDB); freight traffic; freight car tracking

铁路运输管理信息系统（TMIS，Transportation Management Information System）是中国铁路第一个覆盖全路的实时信息系统，它通过计算机网络，从全路6 000多个站的2 000多个主要站点中，实时收集列车、机车、车辆、集装箱以及承运货物的动态信息，对列车、车辆、集装箱和货物进行节点式追踪管理，实现货票、确报、编组站、区段站、货运站、货运营销以及运输调度等的信息化管理。

实现货车追踪主要是为了解决货车管理的“黑洞”问题，以此提高运输组织的效率、降低运营管理的成本，同时促进提高运输服务的质量、提升铁路货运的市场竞争力等，其意义之重大不言而喻。

1　当前货车追踪应用的主要问题

当前货车追踪应用所需的原始数据采集于基层站段，并由站段向铁路局和铁路总公司逐级上报，最后在铁路总公司集中综合处理后形成全路完整的货车追踪数据库。货车追踪应用的后台数据库采用了业界主流的Oracle数据库，处理平台采用了小型机作数据库服务器外加SAN存储的典型架构。全路目前共有80多万辆货车，货车的状态和位置处于不断变化之中，每一次变化都要形成一条轨迹记录，这就决定了当前货车追踪应用的主要特点为数据规模庞大、后台综合处理较复杂且处理量大，同时用户对应用的可用性提出了非常高的要求。

当前货车追踪应用的主要问题是性能问题。（1）随着货车追踪密度的不断加大和用户对数据的实时性和准确性要求越来越高，后台数据库处理系统的能力已渐趋饱和、亟待扩能；（2）后台数据库处理系统因技术架构问题，难以简单横向扩展能力，同时受资金投入和建设周期等多种因素制约，纵向扩展能力也难以一蹴而就；（3）针对当前货车追踪应

收稿日期：2015-01-23
作者简介：颜昌盛，高级工程师；范娟娟，高级工程师。

用的性能问题已经进行了多轮优化，在保持现有软硬件架构不变的情况下，进一步提升性能的潜力非常有限。正因为货车追踪应用的后台数据库处理系统当前存在性能问题，尽管各种用户对货车追踪数据库的访问需求量很大，但目前仅针对有限的用户开放了查询访问权限，应用效果和价值受到了一定程度的制约。

2 解决货车追踪性能问题的思路选择

面对日趋紧张的货车追踪数据库系统的性能问题和不断增长的用户对货车追踪数据库的大量查询访问需求之间的矛盾，要想进一步开放货车追踪数据库的查询访问权限，更好地发挥应用的效果和价值，目前解决问题主要有两种思路：(1) 保持原有系统整体架构不变，通过使用一体机、闪存等新技术产品大幅提升硬件设备能力。(2) 另辟蹊径，采用全新技术对数据处理架构进行改造优化，以大幅提升系统整体处理能力。

货车追踪应用的现有架构是基于传统关系型数据库的，虽曾对应用做过多次优化但仍然不能很好地满足现实要求。传统关系型数据库的主要瓶颈在I/O，尤其是磁盘的I/O对性能的提升限制最大。因此，要想彻底解决问题，摒弃原有架构，采用业界日渐成熟的内存数据库技术，将数据挪到内存以便从根本上解决I/O瓶颈问题，不失为一种良策。

上述第1种思路涉及到投入产出的综合考虑等问题，加上目前机房、电源都比较紧张，难以落地；第2种思路硬件投入相对较少，经过评估我们认为学习成本和实现难度并不大，相对容易落地。下文将基于上述第2种思路，研究解决货车追踪应用的性能问题。

3 内存数据库的发展及应用情况

内存数据库，顾名思义就是将数据放在内存中直接进行管理和操作的数据库。内存数据库抛弃了磁盘数据管理的传统方式，基于全部数据都在内存中重新设计了体系结构，并且在数据缓存、快速算法、并行操作方面也进行了相应的改进，所以数据处理速度比传统数据库的数据处理速度要快很多，能够极大地提高应用的性能。

内存数据库的发展史和数据库的发展史几乎一样长。1969年IBM公司研制了世界上最早的数据库管理系统—基于层次模型的数据库管理系统IMS，并作为商品化软件投入市场。在设计IMS时，IBM考虑到基于内存的数据管理方法，相应推出了IMS/VS Fast Path。Fast Path是一个支持内存驻留数据的商业化数据库软件，同时也可以很好地支持磁盘驻留数据。近年来内存数据库的发展更是突飞猛进，例如出现了eXtremeDB、Oracle TimesTen、SAP HANA、Pivotal GemFire等为代表的一些优秀产品，其中GemFire已在12306网站中得到了很好的应用。

GemFire目前在业界取得的认可主要得益于它在以下几个方面的技术优势：

(1) GemFire支持海量并发请求的能力超群。它可以把客户端的一个查询或是计算请求分发给各个节点，各个节点都查询或是计算本节点的数据，通过协调节点对各个节点返回来的数据进行汇总，再返回给客户端。通过并行计算方式，GemFire可以持续地横向扩展，即增加机器就可以提高支持的并发量。

(2) GemFire采用了标准的TCP/IP协议，对于同一网络下的其它应用没有影响。而有些同类产品采用了私有的TCMP协议，要求交换机支持UDP广播协议，对其它应用可能会产生影响。

(3) GemFire支持大内存，最大实测过支持1 TB ~ 2 TB的内存数据。而其它同类产品一般都是小内存，一个节点只支持1 GB ~ 2 GB内存。如果要缓存的数据规模较大，则需要部署大量的节点。即降低了硬件内存的利用率，也增加了采购的成本，增加了管理的复杂性，比较容易引起网络风暴，同时也会对整个集群的稳定性产生影响。

(4) GemFire提供C++/.net的客户端。其它同类产品对C++/.net应用支持欠佳。

(5)GemFire可以把内存数据持久化在本地应用，同步或是异步均可以。而很多同类产品不支持把数据持久化在本地硬盘，这样若计算机异常断电后可能会导致数据丢失。

因此，下文将基于Gemfire内存数据库产品，研究解决货车追踪应用的性能问题。

4 解决货车追踪应用性能的POC测试研究

4.1 POC测试总体方案设计

本次 POC 测试的主要目的是评估 GemFire 和 Oracle 两种解决方案在大并发环境下的性能差异。本次 POC 测试方案的逻辑架构示意图如图 1 所示。

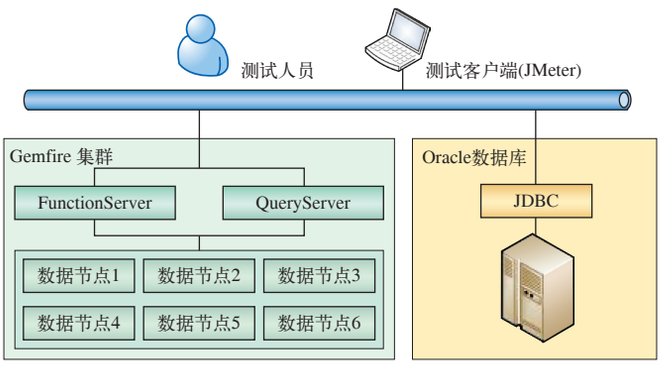

图1 货车追踪应用POC测试方案逻辑架构示意图

图 1 中，虚线左边代表 Gemfire 系统环境，虚线右边代表同一局域网下的 Oracle 系统环境。

在本 POC 测试方案中，通过压力测试工具 JMeter 分别向 Gemfire 和 Oracle 两套系统环境发送查询请求。查询接口分为以下几类：(1) 采用 Gemfire 函数实现查询，JMeter 调用 Gemfire 提供的 FunctionServic, 执行车辆轨迹信息查询的函数；(2) 采用 Gemfire OQL 实现查询，JMeter 调用 Gemfire 提供的 QueryService 执行车辆轨迹信息查询的 OQL (Object Query Language)；(3) 采用 JDBC 实现查询 Oracle，JMeter 采用 JDBC 接口查询 Oracle，实现车辆轨迹信息查询。

关于图 1 中的各个组件及其作用简单说明如下：(1) JMeter，是基于 Java 的压力测试工具。JMeter 在测试中模拟对 Gemfire 和 Oracle 系统发起压力请求，在不同压力类别下测试它们的强度并分析整体性能；(2) FunctionService，是 Gemfire 提供的函数服务接口，开发人员通过开发 Gemfire 的函数服务接口，可以实现复杂的业务行为。在 Gemfire 服务端部署了自定义的函数服务后，在 Gemfire 客户端就可以动态调用这些函数；(3) QueryService，是 Gemfire 执行 OQL 查询语言的服务接口；OQL 是 Gemfire 提供的类似 SQL 语言的接口。QueryService 是执行 OQL 的入口，开发人员采用 Gemfire Client API 获取 QueryService 服务，然后通过它发送 OQL 给 Gemfire 集群；Gemfire 集群接受到 OQL 后会解析 OQL，然后把解析后的执行计划发送到集群的每个节点执行并最终汇总结果给客户端；(4) 数据节点 (CacheService)，是 Gemfire 缓存集群的节点，负责存储数据，并对外提供缓存操作的接口，包括 FunctionService 和 QueryService；(5) JDBC，是 Java 语言中用来规范客户端程序如何来访问数据库的应用程序接口，提供诸如查询和更新数据库中数据的方法。采用它实现对 Oracle 关系型数据库的访问。

4.2 POC测试环境及场景说明

4.2.1 软硬件环境配置

在图 1 中，整个测试环境由 6 台物理 PC 服务器、1 台小型机和 1 台笔记本组成。在 6 台 PC 服务器上部署 GemFire 集群，在 1 台小型机上部署 Oracle 数据库。笔记本作为测试客户端，能分别连接 Gemfire 集群和 Oracle 数据库系统进行压力测试。

6 台 PC 服务器的配置均相同。基本配置：(1) 硬件：CPU：Intel Xeon CPU；CPU 核数：16 核；MEM：128 GB；(2) 软件：OS：Redhat 6.4；Gemfire：GemFire_702。

Oracle 数据库服务器为 1 台 Oracle SPARC T5-4 小型机。基本配置：(1) 硬件：CPU 核数：32 核；MEM：32 GB；(2) 软件：OS：SunOS；Oracle 11g。

JMeter 测试客户端为 1 台笔记本。基本配置：CPU：Intel Core i7-3520M；MEM：16 GB。

服务器端网络为开发测试环境区域的局域网，Gemfire 集群和 Oracle 数据库系统之间带宽是 1 000 Mb/s。JMeter 测试客户端部署在办公区时，与 Gemfire 集群和 Oracle 数据库系统之间的带宽是 100 Mb/s，部署在服务器端时，带宽可以达到 1 000 Mb/s。

4.2.2 测试数据说明

在 Oracle 测试数据库中创建货车追踪轨迹表 (gcarevta)，字段与货车追踪生产系统保持一致。表结构如表 1 所示。

在 Gemfire 集群中创建内存表对象，字段与 Oracle 数据库的表结构保持一致。

从货车追踪轨迹表 (gcarevta) 的数据备份中选择了 15 天的历史数据，共 40 107 420 条记录，分别

表1　货车追踪轨迹表结构

Name	Type
PART_ID	NOT NULL VARCHAR2(4)
CAR_ID	NOT NULL VARCHAR2(20)
CAR_FLAG	NOT NULL CHAR2(1)
CAR_OWNER	VARCHAR2(10)
……	

导入 Oracle 测试数据库和 Gemfire 集群系统。其中导入 Gemfire 集群耗时 35 min，内存占用 460 GB 左右（数据冗余 1 份，共 2 份）。

4.2.3　测试场景说明

本次 POC 测试使用随机读测试场景，从 1 个数据节点开始保持测试交易一直执行，逐步增加至 6 个数据节点并分别进行如下验证：（1）增加数据节点（Cache server）后，交易继续保持执行，观察处理能力是否增加；（2）增加数据节点（Cache server）后，观察缓存集群中的数据是否会自动重新分布，且数据不丢失。

本次 POC 测试共设计了 4 种随即读场景，分别说明如下：

（1）GemfireOQL。此场景是指压力测试客户端通过 Gemfire OQL 编程接口实现查询车辆轨迹业务。OQL 是 Gemfire 提供查询缓存的高级接口，用户可以采用类似 SQL 的语句来查询缓存中的数据。

（2）GemfireFun。此场景是指压力测试客户端通过 Gemfire 的 Function 编程接口实现车辆轨迹查询业务。Function 是 Gemfire 为开发人员提供的编程接口，用户可以通过它实现复杂的业务逻辑。

（3）Oracle。此场景是指压力测试客户端通过 SQL 语句访问 Oracle 数据库实现车辆轨迹查询业务。此场景主要是用于与其它 3 种场景进行性能对比评估。

（4）GemfireFun 单独测试。此场景与上面介绍的 GemfireFun 场景基本相同，不过测试环境做了微调，即将压力测试客户端与 Gemfire 集群部署在相同网段，因此压力机本身的网络带宽可以达到 1 000 Mb/s，避免因 100 Mb/s 网络带宽限制导致不能进一步加压的问题。GemfireFun 单独测试时为 400 并发用户。

在前面 3 种测试场景中，因压力测试的客户端部署在办公网，它与 Gemfire/Oracle 系统之间的带宽只有 100 Mb/s。为了避免网络带宽不足限制对系统性能的充分测试和评估，在执行 Fun/OQL/SQL 查询测试时仅选取返回 5 个字段，分别如下：

（1）Fun：返回的参数是：CAR_NO#RPT_DATE#TRAIN_NO#CUR_STN_NAME#

DEST_STN_NAME；查询条件是 CAR_NO 和 RPT_DATE。

（2）OQL：select CAR_NO,RPT_DATE,TRAIN_NO,CUR_STN_NAME,

DEST_STN_NAME from /gcarevta where CAR_NO=? And

RPT_DATE=?

（3）SQL：select CAR_NO,RPT_DATE,TRAIN_NO,CUR_STN_NAME,

DEST_STN_NAME from gcarevta where CAR_NO=? And

RPT_DATE=?

其中对 CAR_NO 和 RPT_DATE，从测试的原始数据文件中抽取了 500 万不重复的键值对，并且保证都能查询到数据。

4.3　POC测试结果

POC 测试过程是从测试客户端向目标端发起请求，并接收响应目标端返回的数据。为评估测试效果，选取了 5 项技术指标，分别是吞吐量（目标端每秒处理的请求数）、响应时间（目标端处理每次请求的平均时间）、CPU（目标端服务器的 CPU 平均使用率）、测试端网络带宽（测试机的网卡带宽）、Gemfire/Oracle 端网络带宽（目标端服务器的网卡带宽）。本次 POC 测试，选择了 100、200、400 这 3 种不同的并发用户规模，针对每种场景逐一进行测试。详细测试结果如下。

4.3.1　100并发用户对比测试结果

表2　100并发用户对比测试结果

	吞吐量	响应时间	CPU	测试端网络带宽	Gemfire/Oracle端网络带宽
Oracle	813 TPS	123 ms	23%	0.28 Mbit/s	0.21 Mbit/s
GemfireOQL	3 125 TPS	32 ms	69%	2 Mbit/s	4.2 Mbit/s
GemfireFun	11 930 TPS	7 ms	3.9%	10 Mbit/s	1.6 Mbit/s

说明：表 2 测试结果基于相同的查询条件，100

个并发用户。Oracle SQL 查询场景下的响应时间是 123 ms；使用 Gemfire OQL 查询场景下的响应时间为 32 ms；而使用 GemfireFun 即查询接口场景下的响应时间可以达到惊人的 7 ms。这是因为 GemfireFun 没有象 OQL 查询语言那样有语句解析过程且对查询进行了深度优化，因此速度更快。下面 200 和 400 并发用户的情况与此类似，都是 Gemfire 集群比 Oracle 数据库系统性能优异。

4.3.2 200并发用户对比测试结果

200 并发用户对比测试结果如表 3 所示。

表3 200并发用户对比测试结果

	吞吐量	响应时间	CPU	测试端网络带宽	Gemfire/Oracle端网络带宽
Oracle	1 401 TPS	142 ms	40%	0.49 Mbit/s	0.38 Mbit/s
GemfireOQL	4 166 TPS	48 ms	88%	2.6 Mbit/s	4.2 Mbit/s
GemfireFun	12 430 TPS	14 ms	4.1%	10 Mbit/s	1.7 Mbit/s

4.3.3 400并发用户对比测试结果

400 并发用户对比测试结果如表 4 所示。

表4 400并发用户对比测试结果

	吞吐量	响应时间	CPU	测试端网络带宽	Gemfire/Oracle端网络带宽
Oracle	2 080 TPS	192 ms	64%	0.91 Mbit/s	0.76 Mbit/s
GemfireOQL	4 938 TPS	81 ms	90%	2.8 Mbit/s	4.5 Mbit/s
GemfireFun	12 766 TPS	20 ms	8%	10 Mbit/s	1.7 Mbit/s

4.3.4 GemfireFun 400并发用户单独测试结果

因测试客户端开始时部署在办公网进行测试，网络带宽受限，当测试机加压后网络出口带宽被占满，无法对 GemFire 集群继续加压。为了测试出 GemFire 的真实性能，将测试机调整部署在与 GemFire 相同的网段，测试结果如表 5 所示。

表5 400并发用户单独测试结果

	吞吐量	响应时间	CPU	测试端网络带宽	Gemfire端网络带宽
GemfireFun	30 769 TPS	13 ms	36 %	114 Mbit/s	19 Mbit/s

说明：表 5 测试结果是将千兆网带宽占满后的效果。可以看到性能达到 30 769 TPS/s，而响应时间仅仅是 13 ms，CPU 占用率才 36%；也就是说千兆网带宽被占满后服务器性能依然有很大的余量，如果使用万兆网进行测试性能会更好。

4.3.5 POC测试总结

从本次 POC 测试结果看，Gemfire 内存数据库集群系统的性能要远高于传统的 Oracle 数据库系统。

当压力测试客户端部署在仅具有百兆接入能力的办公网时，在 Gemfire OQL 场景下集群系统的性能比 Oracle 数据库系统提高 1 倍以上，在 GemfireFun 场景下其性能提高 6 倍以上。GemfireOQL 测试场景的性能瓶颈是 Gemfire 集群的服务器 CPU 使用率，当在 400 并发用户下已达到 90%；而 GemfireFun 测试场景的性能瓶颈是测试客户端与 Gemfire 集群之间的网络带宽，在 3 种并发用户场景下均达到 10 Mbit/s（百兆带宽）。

当压力测试客户端部署在与 Gemfire 集群相同的网段后，基本消除了压力机的带宽瓶颈。在 GemfireFun 场景下集群系统的性能比 Oracle 数据库系统提高 15 倍左右，并且最后性能瓶颈不是 Gemfire 集群本身，而是测试客户端使用的网络带宽已达到限定值（千兆带宽）。

5 结束语

通过本次 POC 测试，看到 GemFire 内存数据库集群系统的性能比 Oracle 数据库要高出 15 倍左右（千兆网环境下），如果是万兆网环境，性能提升会更加可观。而且 GemFire 集群的运行环境是基于 X86 服务器的 Linux 环境，且不需要共享存储，因为它性能的提升完全依赖于分布式运算的强大处理能力，而不依赖于高性能共享存储。这与传统的关系型数据库提升性能很大程度上依赖于存储性能的提升完全不同。本次 POC 测试的代码开发与调试不到两周，经过评估，结论是：使用 GemFire 作为后端数据库，涉及到的修改货车追踪应用的代码工作量不大；GemFire 提供了与 SQL 语句类似的查询语言 OQL，降低了研发人员与后期运维人员的学习成本。

因此，无论从前期投入，还是从全生命周期的效益来看，使用 GemFire 内存数据库改善货车轨迹追踪应用的性能都是非常可取的解决方案。

参考文献：

[1] 盖国强．内存数据库的基本原理与应用 [EB/OL]. http://www.eygle.com/digest/2011/11/memdb_timesten_altibase.html.2011.

责任编辑 方 圆