基于 PL/SQL 的批量处理应用的性能优化策略①

张孝斌, 王 超

(中国民航信息网络股份有限公司, 北京 100102)

摘 要: Oracle 数据库系统是目前企业应用最广泛的大型关系数据库管理系统. PL/SQL 是 Oracle 对结构化查询语言 SQL 扩展的过程性语言, 利用其设计和开发触发器、视图、存储过称、包以及函数等实现对数据库的数据处理. 在很多企业应用架构设计中, 会采取 Oracle 端存储过程实现一些用户交互少、逻辑复杂和涉及数据量大的批量处理, 使得数据库具备了数据存储和业务处理的双重角色. 但是在这种应用中, 随着数据量的不断增加, 如何保证存储过程的性能非常关键. 本文结合中国民航国际客运收入管理系统海外版性能优化的成功实践, 简要介绍了 Oracle 的 OEM 工具和 AWR 报告两个发现性能问题的工具, 并从代码结构调整降 I/O、临时表和模拟多线程三方面总结了基于 PL/SQL 的批量处理应用的性能优化策略, 为采用此类架构设计的开发人员和运维人员提供持续优化的思路.

关键词: PL/SQL; 存储过程; 批量处理; 性能优化

Performance Optimization Strategy of Batch Process Application Based on PL/SQL

ZHANG Xiao-Bin, WANG Chao

(Travelsky Technology Limited, Beijing 100102, China)

Abstract: As large-scale RDBMS, Oracle is used most widely in enterprise application. PL/SQL is programming language based on SQL and expanded by Oracle. We could adopt PL/SQL to develop trigger, view, stored procedure, package and function to process data. In many architecture designs of enterprise application, stored procedure is utilized to achieve batch processing which does not need more user interaction, owns complex logic and often involves a large amount of data, making the database have dual roles which are data storage and business process. With the increasing amount of data, how to guarantee performance of stored procedure is very important. In this article, based on successful practice of performance tuning to oversea version of China Aviation International Passenger Revenue Accounting Solution, the author introduces Oracle's OEM and AWR report simply which are used to discover performance problem, and three technologies as optimization strategy adopted in batch processing application based on PL/SQL are summarized, they are code refactoring to decrease I/O, temporary table and simulating multi-thread. The target of this article is to provide thinking for developer and maintainer engaging in continuous optimization about the application utilizing such architecture.

Key words: PL/SQL; stored procedure; batch processing; performance optimization

1 前言

当前, 基于 Oracle 数据库的应用系统越来越广泛, 特别是企业级应用. 随着社会经济发展和企业业务的扩张, 企业级应用系统一般具有数据量大、业务复杂高的趋势, 批量处理具有用户交互少、数据处理量大、业务逻辑复杂等特点, 它和 ONLINE 处理构成了大多数面向数据库企业应用方案, 广泛应用于银行、金融、民航、电子商务、证券、医药等行业, 因此除了确保 ONLINE 处理性能外, 如何保障批量处理的性能对这些行业级应用非常重要.

① 收稿时间:2015-08-21;收到修改稿时间:2015-11-02

PL/SQL 是 Oracle 对结构化查询语言 SQL 扩展的过程性语言, 利用其开发的存储过程可实现对批量数据的加工处理. 因此, 考虑到减小应用服务器和数据库服务器之间的网络压力, 架构设计者会将应用服务器端一些涉及数据量大、业务处理复杂、用户交互少的批量处理逻辑转移至数据库端形成存储过程或包, 从而构成客户端、应用服务器和数据库服务器多层应用开发架构. 在这种架构下, 数据库服务器拥有业务处理和数据存储双重角色, 因此在采取 Oracle 物理设计、逻辑设计、操作系统参数调整、数据库参数调整、索引等优化策略的基础上[1], 优化存储过程的重要性对于整体应用不可低估, 而这一方面鲜有经验总结. 本文在阐述 PL/SQL 与存储过程、基于存储过程的多层应用架构的基础上, 结合中国民航国际客运收入结算系统海外版性能优化的成功实践, 重点探讨了存储过程新的优化策略以及效果, 从而为采用此种架构的开发人员和运维人员的性能优化工作提供借鉴.

2　PL/SQL与存储过程

PL/SQL 是一种高性能的基于事务处理的语言, 是ORACLE对标准SQL的扩展, 支持所有数据处理命令. 利用该语言可以灵活操作 Oracle 的各种数据资源, 从而开发出各种存储在数据库服务器端的程序对象, 如函数、存储过程、包、视图等. 这种扩展体现在两个方面: (1)通过增加变量和类型、IF 和循环控制结构、过程和函数等用在其他过程性语言中的结构, 提供了在数据库上执行各种业务逻辑的途径; (2)与 SQL 相比, PL/SQL 是将多个 SQL 语句按照一定逻辑形成一个程序块存储于服务器, 从而达到网络通信少和应用程序的执行效率更高的目标[2].

Oracle存储过程是一个PL/SQL程序块, 即一组为了完成特定功能的由流控制和 SQL 语句开发的过程, 经编译和 SQL 优化后存储在数据库中. 同时, 存储过程可以被授权用户在任何需要的地方调用[3]. 采用它不仅起到了保证数据的安全性和完整性作用, 而且与一般的匿名 PL/SQL 块主要区别是: 当客户端应用程序调用时, 只需发送一条调用命令即可触发数据库服务器执行该过程, 大大降低了网络通信的负担.

3　基于存储过程的多层应用架构

Oracle 数据库系统借助 PL/SQL 使其的功能由数据存储扩展到业务处理, 从而使得架构设计师为了提高 Oracle 数据库端业务处理能力、减小应用服务器和数据库服务器间的网络压力以及其他安全考虑, 而将一部分用户交互少、数据量大和业务处理复杂的逻辑转移至 Oracle 的存储过程或包, 形成嵌入在 Oracle 中的“逻辑层”, 最终形成了基于 Oracle 存储过程的多层应用架构, 具体见图 1.

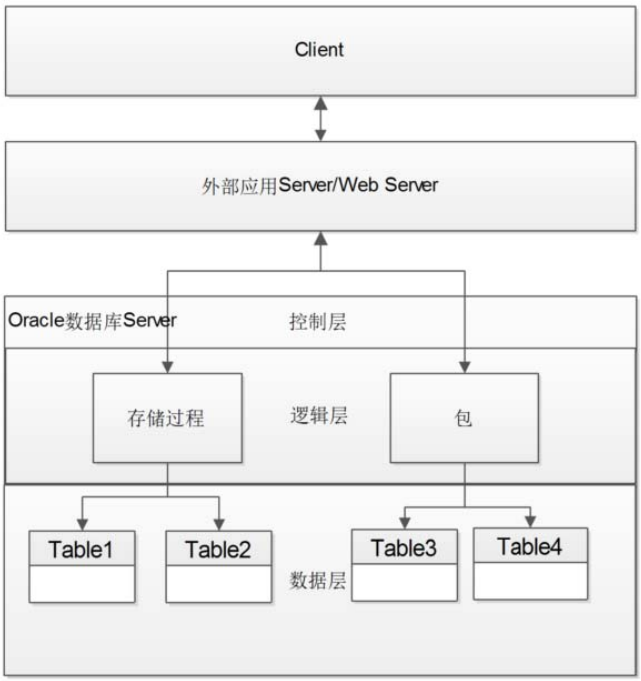

图 1　基于 Oracle 存储过程的多层应用架构

这种架构的优点如下:

① 减少应用服务器和数据库服务器间的网络流量. 存储过程位于服务器上, 被调用时只需要传递存储过程的名称和参数即可.

② 执行效率高, Oracle 中的存储过程、函数等均在创建时进行编译一次即可, 并且其中的 SQL 语句可以通过绑定变量等减少硬解析;

③ 安全性高. 参数化的存储过程可以防止 SQL 注入式攻击, 而且可以将 Grant、Deny 以及 Revoke 权限应用于存储过程提高访问安全.

当然这种架构也存在一定的缺点, 因此选择时需要做出一些平衡. 具体如下:

① PL/SQL 代码移植性不高. 因为 PL/SQL 的语法仅在 Oracle 数据库中适用, 一旦转为其他数据库系统如 DB2, 则需要重写或改造.

② 不支持并发. 因此大批量数据处理时可采用优化对策很有限.

③ 尽管在创建时已经编译, 但有时也需要重新编译. 如果带有引用关系的对象发生改变时, 受影响的存储过程则需要重新编译.

4 存储过程的优化策略

4.1 工具

(1) AWR 报告

AWR 报告全称是“Automatic Workload Repository”, 是 Oracle 10g 版本开始引入的一个重要组件, 是一种性能收集和分析工具. 通过该报告可以了解应用在一个时间段内的整体运行状况, 包括 CPU、I/O、内存资源利用情况以及耗时 SQL 情况等, 借助该报告很容易发现存储过程中影响性能的 SQL 语句和代码设计.

AWR 报告涵盖的内容非常多, 此处重点介绍仅涉及存储过程本身性能的关注点——耗时 SQL 列表(SQL ordered by Elapsed Time)章节, 见图 2.

图 2 SQL ordered by Elapsed Time 列表示例

通过分析该章节可知涉及 SQL 语句执行次数、单次性能是否正常等, 这些可更好的辅助性能优化, 表 1 则简单介绍其中表格的各字段含义和具体用途, 有助于正确理解和分析.

表 1 SQL ordered by Elapsed Time 列表字段解释

编号	字段	含义	用途
1	Elapsed Time (s)	总执行耗时	执行时间长的 SQL 语句, 通常意味着问题所在
2	Executions	执行次数	异常的执行次数应该引起重视
3	Elapsed Time per Exec (s)	每次执行的耗时	单次执行时间长的 SQL 语句, 通常意味着问题所在
4	%Total	该语句的执行时间占总数据库时间的百分比	性能参考
5	%CPU	该语句消耗的 CPU 占总数据库 CPU 消耗的百分比	性能参考
6	%IO	该语句造成的 IO 占总数据库 IO 的百分比	较高的 IO 比例说明 SQL 语句获取数据的方式可能出了问题
7	SQL Id	执行 SQL 的编号	
8	SQL Module	执行 SQL 的进程或程序	
9	SQL Text	执行 SQL 部分内容	通常可以根据这一项调整 SQL 语句, 或者优化索引

(2) OEM

AWR 报告适用批量处理应用的事后性能分析, 而 OEM 则属于事中、实时分析. OEM 全称是 Oracle Enterprise Manager, 利用 OEM 控制台可以实时了解 Oracle 执行批量处理应用的性能状况, 可细化到 SQL 级, 即采用 SQL Monitor 功能监控正在执行的存储过程里各个 SQL 语句的执行情况, 如图 3 所示.

图 3 中不同颜色标记出了不同 SQL 语句所消耗的时间, 面积越大表示消耗的时间越多, 可通过点击右侧相应 SQL_ID 可以查看相应 SQL 的执行计划以及所消耗的各类系统资源情况, 从而确定是否需要优化.

(3) 代码走查

代码走查是拥有多年开发经验的技术人员通过对代码进行逐行分析以及整体代码层次、架构、调用等研判, 从而确定代码逻辑设计中的冗余、繁琐和其他不合理之处, 也可以采用 CODE REVIEW 会议的形式. 对于某特定批量处理应用, 代码走查的人员可以由两种角色的人进行两轮甚至多轮完成, 具体见下表. 其中, 原开发人员的走查非常必要, 因为其对业务流程和代码设计最为熟悉, 最容易找到优化的办法.

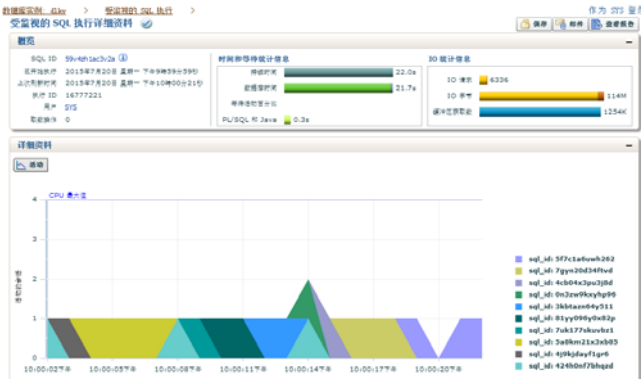

图 3 SQL Monitor 执行某存储过程涉及 SQL 执行详细资料示例

表 2 代码走查各轮次策略

角色	关注点	轮次
原开发人员	业务逻辑设计整体考虑是否更优、冗余代码是否存在等	第 1 轮
PL/SQL 多年开发经验的人员	SQL 语句是否更优、代码结构是否更优、索引使用是否合理等	第 N 轮 (N>1)

4.2 优化技术

无论采用 Oracle 提供的各种监控工具还是人工走查, 最终在发现问题之后均需要采取一定的优化技术来解决. 对于实现批量处理的存储过程, 在不考虑网络性能和已有硬件资源如 CPU、存储磁盘、内存、系统参数设置等前提下, 通常可以采取索引、动态 SQL、批量绑定等常见技术对存储过程进行优化. 但在实际性能优化过程中发现某些应用在使用这些常见技术的基础上依然无法改善批处理性能, 因此经过不断实践和探索又总结出三种新技术: 代码结构调整降 I/O、临时表和模拟多线程, 这三种技术均是在解决不同场景下应用性能欠佳的问题. 当然无论那种技术, 存储过程性能优化的目标都是为了达到提高其检索效率和减少 I/O 操作, 最终提高执行效率.

(1) 代码结构调整降 I/O

代码结构调整降 I/O 指在不影响业务逻辑准确性、特定的更新 SQL 语句本身无法再优化以及该 SQL 语句的 COST 很高的前提下调整实现批处理应用的存储过程的代码结构, 减少该 SQL 语句的执行次数, 从而提高整个批量处理的执行效率. 使用该技术有以下两个前提: ① 识别特定的更新 SQL 语句, 其具有“自身无法再优化而单次执行的 COST 很高”和“执行次数异常多”两个特点, 一般需要通过分析执行该存储过程的某个时间段 AWR 报告, 利用其中的“耗时 SQL 列表”中“Elapsed Time per Exec (s)”和“Executions”进行研判, “Elapsed Time per Exec (s)”表示单次执行的耗时, “Executions”表示该存储过程中该 SQL 被执行的次数, 而“Executions”是否异常需要凭借数据分析和业务背景综合判断; ② 在识别特定 SQL 的基础上, 原开发人员需要仔细研究存储过程的代码设计架构, 并同业务人员一同分析调整代码结构的方案, 从而达到在不影响业务逻辑准确性的基础上降低该 SQL 的执行次数目标.

采用该技术可能会使得原有代码结构冗余或松散, 但是目的是改善应用程序的性能, 这将直接影响到用户体验乃至对产品成败的认知, 因此不能一度追求代码结构紧凑和简明而忽略的性能. 而这一点在系统设计和开发阶段不容易被重视, 在性能测试阶段乃至投产阶段才会显现出来, 所以优化的难度较大, 尤其是优化前的分析工作量较大.

(2) 临时表

在存储过程处理某数据表中大批量数据的时, 如果该表中已有的数据量规模已经很大, 那么可以使用临时表技术可以确保存储过程性能不受人为或其他进程访问相同数据的干扰. 在存储过程中采用该技术指: 在代码逻辑中增加“将要处理的数据备份至临时表”步骤, 进而从该临时表读取相关数据, 后续逻辑操作针对该临时表, 但若有更新操作依然作用在原表上. 另外, 在创建临时表时需要将原表的虚拟列“ROW ID”一同建立, 若有更新原表操作时则可利用原表的“ROW ID”作为 WHERE 条件从而极大提高执行效率, 因为“ROW ID”是原表的系统级定义的唯一性索引. 采用该技术所应对的场景应具有如下两个特点:

①一般通过分析 AWR 报告或 OEM 检测发现该存储过程执行过程中某检索数据的 SELECT 语句耗时较大且执行次数较多, 而且该语句本身的执行计划已经最优;

②该 SELECT 语句在每次执行时是从千万级别规模的数据表中选取百万或更小级别规模的数据.

因此, 在存储过程的中采取该技术主要从三个方面达到提高执行效率的目的: ① 避免在批量处理过程中人为或其他进程修改数据导致降低批量处理的效率, 因为 Oracle 为了保持读一致性而串行读 UNDO 空间; ② 避免了对大数据表的频繁读操作, 而将其转换为对小表的读操作, 从而提升批处理的效率; ③从临时表中读取数据时不需要使用 UNDO 空间, 因而降低 UNDO 空间的使用. 当然, 这种技术也不能过度使用, 否则会对 TEMP 空间依赖性过大.

(3) 模拟多线程

PL/SQL 的语法是不支持多线程的, 所以在数据库操作系统的 CPU 利用率不高和批量处理效率不高的情况下, 对某些存储过程可以采用模拟多线程来实现并发达到提高数据处理能力的目的. 模拟多线程指借助外部多线程技术对 Oracle 的存储过程同时发起多个请求, 而多个请求一定需要具备两个条件: ① 每个

请求所涉及的数据范围不同, 即作用在不同的数据记录; ②每个请求的逻辑操作互相无影响, 即业务逻辑方面不能存在交叉更新操作或逻辑依赖, 两个条件任何一个不能满足则不应使用该技术. 该技术总结为: 外部多线程调用 Oracle 的多进程, 例如利用 Java 或 NET 的多线程技术传递不同的参数并发调用存储过程, 具体原理见图 4.

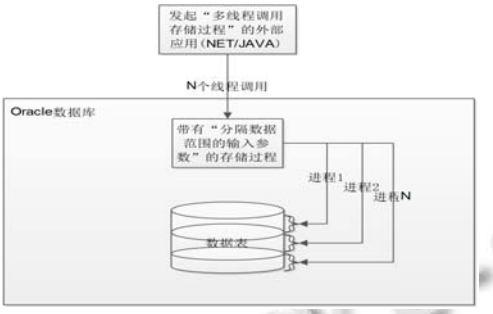

图 4 模拟多线程原理图

该技术的出发点就是在将大数据表切块和并发的情况下充分利用 CPU 的利用率, 通过并行执行存储过程从而达到提高执行效率的目的. 采用该技术需要做三方面工作: ① 修改存储过程的参数以及锁定数据范围的条件, 目的是让每个线程传递不同的参数对不同的数据记录进行逻辑操作; ② 在已知操作系统 CPU 的核数前提下, 需要通过多次实验寻求最佳的并发线程数, 并非线程越多效率就越高, 因为操作系统在较多的线程之间切换就会有一定耗时, 切换越频繁则耗时越多; ③ 将大数据表切块的范围字段必须有索引且有效, 否则会增加 IO 负担而导致效率不理想.

此外, 使用该技术后需要进行充分的数据准确性验证, 即针对相同的数据范围和单一线程执行后的结果进行对比, 对比需要关注两点: ① 多线程之间是否存在数据范围上的缝隙, 即确保大数据表切块的完整性; ② 执行结果的各个数据维度是否和单一线程一致, 即再次验证多线程间的逻辑操作无影响. 总之, 存储过程优化前后的结果必须是一致的.

5 优化策略的应用

中国民航国际客运收入管理系统海外版是面向航空公司收入结算部门的内部管理系统, 即典型的 ONLINE 操作和批量处理操作相结合 OLTP 系统, 处理对象是旅客在国际航线旅行过程中产生的以电子客票为主的各类票证销售数据和运输数据, 该系统采用 Oracle 存储过程的多层应用架构. 近些年来随着社会经济的不断发展和民众旅行消费需求日益高涨, 国际出行的旅客量不断攀增, 使得该系统所处理的数据量不断增长, 年处理量在 TB 级, 所以该系统最大的特点是处理的数据量大. 因此, 用户对该系统的性能尤其是批量处理要求更高.

该系统在分析设计、开发阶段以及性能测试阶段已经做了大量的性能优化工作, 但是在投产初期性能依然表现不佳, 主要存在两个问题: (1) 每天夜间的批量处理作业无法在早上八点前完成, 进而影响到用户白天的 ONLINE 操作; (2) 某定期执行的大型作业运行时间远远超过用户预期, 进而影响后续用户多个系统间数据交换和其他业务处理的如期完成. 而根本问题是: 这些采用 Oracle 存储过程实现的作业已经采取了分区表、优化索引、动态 SQL、批量绑定等常见技术进行了优化, 但依然无法达到用户的预期. 对此, 除进行磁盘阵列优化、系统参数配置等非应用方面调整外, 我们在应用层面采用分析 AWR 报告、监控 OEM 以及人工走查等手段分析耗时较长作业存在的问题, 在不断实践中探索出临时表、代码结构调整降 I/O 和模拟多线程三种新的优化技术, 单独或综合使用使得某些场景下批量处理应用的性能改善取到了明显效果.

5.1 临时表的应用

某大型作业执行过程出现了异常慢的现象, DBA 通过 OEM 实时监控发现该作业涉及的存储过程在执行过程中出现了几个“坑”, 如图 5 所示见浅绿色区域, 坑是运行 SQL_ID 为 6cpghhf1x2mj9 时出现的. 经分析发生如下情况: 在存储过程执行过程中由于人为或其他进程修改了其所读取的数据, 导致 Oracle 为了保持读一致性而串行读 UNDO 空间, 直观上来看就是 Oracle 数据库“罢工”, 实际上一直卡在一条读取数据的 SQL 语句上, 从而使得作业效率异常降低, 正常情况下该 SQL 语句应该很快执行完毕.

图 5 OEM 监控出现“坑”的罢工图示

经过 DBA 和高级数据库架构师的分析, 该存储过程中相关 SQL 语句的执行计划已经很好, 并且索引和动态 SQL 等技术也已使用, 因此建议对该存储过程中采取临时表技术: 增加将所要操作的数据转移至临时表的额外步骤, 并确保该进程执行过程中仅针对该临时表中的数据. 程序优化后, 通过 OEM 监控再未出现过“坑”, 系统正常运行, 并且大大缩减了等待时间, 下图是代码优化前后示例.

图 6　临时表优化代码前后对比图

在图 6 示例中, 其中表 uplbcc 采用字段 upaprm 进行分区且存储了海量数据, 分区字段是月份, 尽管检索一个月份的数据量并不算太大, 但采用临时表可以避免该作业运行过程中其所作用的数据被其他进程或人为干扰而减低运行效率.

5.2 代码结构调整降 I/O

针对已采取了常规技术优化后的某作业依然存在运行效率低的问题, 通过对该作业运行时间段的 AWR 报告分析, 发现 SQL 语句“update a_tkt tset t.a_code = :a_code where t.prm=:prm .....”单步执行耗时大且执行次数多, 而且该 SQL 语句的执行计划已经很好, 单步执行耗时大是因为表 a_tkt 的数据量太大所致, 所以只有减少该 SQL 的执行次数才能提升效率. 在此情况下由经验丰富的开发人员多次走查执行该SQL语句的存储过程, 发现在确保业务逻辑正确的前提下该 SQL 语句独立于原循环结构而新建循环结构后, 其被执行的次数可以降低一个数量级, 因此决定对代码结构进行调整, 图 7 是代码修改前后示例. 对于 600 万条记录的业务数据, 经过优化的该存储过程执行时间由 3 个小时缩短至约 30 分钟, I/O 从近万次降到几百次, 效率提高了约 90%.

开发人员最初将该SQL语句放在循环结构考虑到代码的简洁而忽视了性能, 原循环结构中 SQL 语句“insert into a values rec_acc”被执行很多次是正确的且无需再优化的, 但是 SQL 语句“update a_tkt tset t.a_code = :a_code where t.prm=:prm .....”则不应被执行相同的次数, 因为从业务分析它的变量“a_code”所适用的数据范围经过聚合后是前者的一个很小的子集, 尽管放在一个循环中也能保证结果的准确, 但由于该 SQL 自身 COST 很高和执行次数过多产生叠加效应, 最终导致效率低.

图 7　代码结构调整降 I/O 优化前后对比图

5.3 模拟多线程

该系统中生成账务的作业最为耗时, 每次执行一般要对约 160 万的数据记录进行加工并产生 700 多万的数据记录. 经过分析其代码结构和相关 SQL 语句已经使用了常规优化技术依然, 但耗时依然超过了用户预期, 因此必须采用新的技术才能改善性能, 同时发现该作业存在如下两个特点:

(1) 该作业运行时其他运行的作业和用户 ONLINE 操作很少, 即 Oracle 服务器 CPU 的利用率不高.

(2) 该作业的存储过程逻辑大致是三步: a.锁定源表中一定范围的数据; b.对源表中每一个记录进行简单加工; c.向目标表中插入若干数据记录, 然后重复 b 和 c. 即执行过程中源表中各条记录间逻辑操作无影响, 且源表被作用的记录可以用飞行日期进行均匀切分.

因此, 受到 NET 技术通过多线程实现并发提高效率的启发后我们决定采用模拟多线程技术, 即 NET 的多线程调用 Oracle 的多个进程. 我们对该存储过程及其调用做了如下修改:

(1) 原输入参数基础上增加参数 i_vc_where 表示“起始飞行日期”和“终止飞行日期”, (2)在原有锁定数据范围的条件的基础上增加了 i_vc_where 参数, (3).NET 端利用多个线程同时触多个调用, 传递不同的日期范围. 下图是该存储过程修改前后的对比示例.

图 8 模拟多线程改造前后代码对比

在既定核数的 CPU 情况下, 经过多次实验确定 4 个线程调用优化后存储过程性能改善最佳, 对于 160 万的数据量, 其执行耗时由之前的 120 分钟缩短至 90 分钟, 效率提高了约 25%, CPU 的利用率则由 50%提高至 85%, 达到了性能优化的目标. 由上图可以看到在该存储过程优化过程中也同时使用了临时表技术, 因此性能优化常常是需要使用多种技术不断的达到预期.

6 结束语

基于数据库的企业级应用系统的性能优化是一项系统工程, 该工程应贯穿系统需求分析阶段、设计阶段、开发阶段、运维阶段和升级改造阶段等全生命周期[4]. 针对采用 PL/SQL 的存储过程实现批量处理的多层应用架构, 可充分利用 AWR 报告和 OEM 等工具等监控性能问题, 采取恰当的技术来提高实现批量处理的存储过程性能. 本文以作者的实际工作经验总结出临时表、代码结构调整降 I/O 和模拟多线程三种技术优化存储过程, 并且取得了非常明显的优化效果, 希望给此类应用遇到类似性能问题提供帮助.

参考文献

1 任玉辉,王成良.企业级应用系统中数据库性能优化策略.计算机系统应用,2008,17(3):7−11.
2 王秋生.基于 PL_SQL 的 Oracle 数据库性能优化.微机发展,2003,(12):46−49.
3 孔明华.基于 PL/SQL 的自动数据处理方法研究.信息技术,2013,(3):124−126.
4 茆德柱,钱晓东,闫训超.性能优化在信息系统全生命周期中的研究.计算机系统应用,2014, 23(4):214−217.
5 蒲宇.基于 Oracle 的应用软件系统检索性能的优化.电子技术与软件应用,2013,(7):99.
6 李晓勤.多进程在银行联机事务批量处理中的应用.中国金融电脑,2006,(12):37−42.