工作流系统性能分析与优化方案①

林加镇, 杨 帆, 涂红兵, 王云福

(深圳中广核工程设计有限公司, 深圳 518172)

摘 要: 随着企业业务流程信息化建设的推进, 对核心业务领域的流程化管理已成为实现信息化水平进一步提升的重要基础, 工作流系统作为基础设施, 其性能在一定程度上决定了业务流程流转的效率. 本文通过对工作流系统性能的量化分析, 得出其性能指标与数据库规模的量化关系, 从而为性能优化工作提出关键因素和技术切入点, 在此基础上, 本文结合数据库机制对工作流性能优化方案进行了研究和实现, 提出了一种有效的算法, 实现了归档机制, 从而达到提升工作流系统性能优化的最终目的.

关键词: 信息化; 工作流系统; 性能优化; 量化关系; 归档机制

Performance Optimization of Workflow System

LIN Jia-Zhen, YANG Fan, TU Hong-Bing, WANG Yun-Fu

(China Nuclear Power Design Co., Ltd (ShenZhen), Shenzhen 518172, China)

Abstract: With the promotion of information construction of enterprise business process, the management of core business process has become an important foundation to further improve the information level. The performance of workflow system determines how efficiently the business process flows to some extent. In this paper, quantitative relationship between performance index and database scale is sought out by quantitatively analyzing the performance of workflow system to put forward key factors and pointcuts of technology for performance optimization. On this basis, performance optimization scenarios of workflow are studied based on database mechanism in this paper and a simple effective algorithm is proposed. As a result, filing mechanism is implemented so that the ultimate goal of workflow system performance optimization can be achieved.

Key words: information construction; workflow system; performance optimization; quantitative relationship; filing mechanism

1 现状及问题

近年来, 随着企业信息化建设的深入, 对核心业务领域的流程化管理已成为实现信息化水平进一步提升的重要基础, 工作流系统, 从某种意义上作为一种信息化基础设施, 因其具备统一的流程管理及配套功能, 在企业级的业务管理系统中正占据越来越重要的地位.

从另一角度来说, 由于工作流系统涵盖了核心业务及海量的流程化数据, 加之其作为基础设施的唯一性, 其本身的性能状况极易演变为企业业务管理系统的整体性能瓶颈, 在这种情况下, 企业的信息化对其依赖程度将变得越来越高, 这已成为不可避免的趋势.

基于上述趋势的研判, 本文认为, 对于工作流系统性能优化技术的研究及实现, 将具有重要的现实意义.

2 工作流性能分析

本文进行工作流系统性能分析的基本思路, 是在流程生命周期分析及系统数据结构复杂度分析的基础上, 得出工作流各项核心功能的运行指标及其与相关因素的变化关系, 从而寻找出性能优化的关键点, 为方案的实施提供依据.

① 收稿时间:2014-04-10;收到修改稿时间:2014-05-09

2.1 工作流生命周期

工作流的流转是由状态驱动的, 其生命周期, 即工作流数据从产生到失效的全过程, 总体上包括了启动, 流转以及关闭三个状态(如图 1).

启动状态(START): 系统根据当前流程模板生成流程实例, 并把新创建的工作流对象托管到工作引擎. 作为一种初始化的状态, 系统为工作流对象分配内存空间以及 CPU 资源, 并在数据库、文件系统等进行数据读写、日志记录等 I/O 操作.

流转状态(ACTIVE): 工作流启动以后即进入流转状态, 大部份情况下, 流转状态并不单指某一个节点, 从逻辑结构讲, 它是由一系列串行或并行的节点组成, 这些节点相互之间存在业务联系, 是业务流程在工作流层面上的实现. 如下图所示, 流转状态的这种结构是在工作流模板中预定义的并在生成工作流对象时被实例化.

关闭状态(CLOSE): 当对业务节点的处理满足某种预定义的条件时, 将触发其进入关闭状态. 该状态是工作流生命周期的最后一个状态, 也是最持久的一种状态. 对于处于关闭状态的流程, 工作流引擎按照期既定的归档机制, 回收其占用的计算资源并记录日志. 从数据层面上讲, 流程对应的数据并不会被物理删除, 系统只是作逻辑标识并对其进行归档处理.

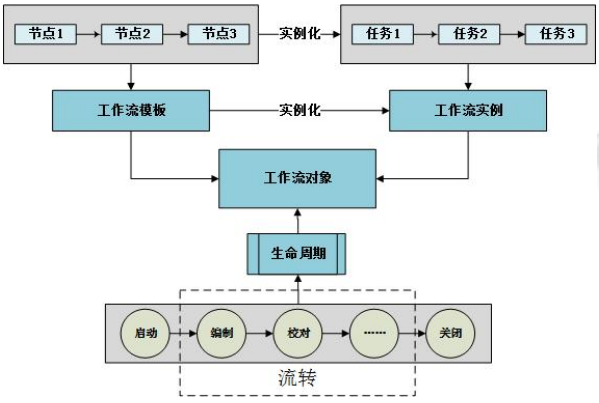

图 1　工作流的生命周期

以上所述关于工作流生命周期的三种状态, 是数据在系统中从产生到归档(即成为历史数据)的不同阶段. 随着流程数据的增加, 数据库中已归档的数据呈倍增的趋势增长, 与此同时, 系统对历史数据的归档机制也逐渐成为工作流系统的性能瓶颈, 这些因素都给系统的稳定运行带来越来越大的压力. 因此, 对历史数据的考察与分析, 是工作流系统性能优化及优化方案设计的一个重要前提.

2.2 流程库及其数据结构

从系统框架的层次结构(图 2)可以看出, 工作流系统作为基础设施, 为业务管理系统提供流程服务和支撑, 在工作流基础设施之上为 API 接口层, 该层次实现业务端与工作流的解耦和分层部署. 工作流 API 接口的主要功能, 是封装底层的数据状态驱动逻辑, 实现工作任务在业务端按固定模版进行流转, 同时为调用方提供任务状态的查询, 管理等程序界面.

在底层架构中, 工作流系统的核心数据库主要分为核心实例库(K2Server)以及流程日志库(K2ServerLog). 其中, 核心实例库存储流程模板以及当前的活动实例数据, (即对应于 START 和 ACTIVE 两种状态的工作流数据), 在工作流引擎运行过程中, 该库的数据被直接读入工作流服务引擎的数据内存中. 流程日志库中记录所有流程的日志信息, 包括活动的以及非活动的流程日志数据(包括部分 ACTIVE 状态与所有的 CLOSE 状态的数据), 该库中所有的日志数据都从核心实例库中归档而来.

从数据结构的层面进一步考察工作流数据的存储方式, 可以看出上述核心数据库(K2Server 及 K2ServerLog)中的流程表类型如表 1.

表 1　流程表类型

序号	关键字	含义
1	Proc	流程数据表
2	Act	流程节点数据表
3	Dest	处理人信息
4	Inst	流程实例表
5	Audit	审计记录表
6	Xml	Xml 类型变量
7	Data	普通变量
8	Slot	任务可用的空缺
9	Line	流程线条
10	Log	日志表
11	Status	状态表
12	WorkList	待办任务表

为了实现流程业务的完备性, 流程数据在数据库中的存储方式呈现出以流程实例为中心的星形结构(其中至少涉及 12 张表, 以实例 ID 进行关联), 这种主从表的关联方式在一定程度上给数据读写带来较大的

开销, 随着数据量的增长, 这种开销变得日益严重.

工作流引擎在上述两个数据库之间建立实时归档机制, 把状态为CLOSE的流程数据写入日志库, 同时, 为了确保数据的完备性, 流程日志库中还存在大量ACTIVE 状态数据, 这种实现方式导致日志库数据规模的快速增长, 对数据库I/O造成巨大压力, 从业务角度来说, 其直接结果是降低了新流程流转的性能指标.

2.3 性能分析

性能分析的关键在于找出工作流性能指标与数据库规模的量化关系, 从而为优化方案的设计及实施提供依据.

假设:

①系统中仅存在一种工作流程模版 *P*;

②P 中定义了一种具有 *n* 个节点的流程;

③数据库中与流程实例相关联的表数量为 *C*(*C* 为常量);

④系统中已有的流程数为 *N*;

⑤流程库中实际存储的数据量为 *S*, 即流程库规模;

⑥系统的性能指标为 *T*, *E*, 其中 *T* 表示系统新产生一条流程所需要的平均时间开销(单位: 毫秒), *E* 表示在特定条件下系统生成新流程实例的失败概率.

由于流程中每个节点的数据都需要在 *C* 张表中进行数据存储, 因此: 系统流程库中实际存储的数据量(*S*)与上述各个参数的量化关系可以表示为:

$$S \approx Cn*N. \quad (1)$$

其物理意义为: 系统中每产生一条具有 *n* 个节点的流程, 流程数据库中则需要存储 *Cn* 条数据, 即为正比例关系. 这是在理想假设下得出的结论, 而实际情况为: 由于业务操作的复杂性, 在大部分情况下, 系统中存在多个流程模版(其节点数*n*不同), 模版的流程结构也更为复杂(包括串行, 并行结构), 并且各个流程实例都有可能存在反复流转(如节点回退等)的情况, 因此, 在实际条件下, 流程数据量的增长要远远大于依据公式(1)得出的数据.

为了验证流程库规模 *S* 与流程系统 *T* 和错误率 *E* 之间的关系, 本文在工作流系统的开发环境下展开数据实验, 目的在于通过实验测试得出 *T*, *E*, *S* 三者之间的数量关系.

本实验采用典型的由“编-校-审-批”四个节点组成的流程模版作为测试样本, 在不同的数据规模条件下生成流程实例, 得出如表 2 所示的实验数据.

表 2 *S*/*T*/*E* 数量关系

流程库规模(*S*)	平均时间开销(*T*/*ms*)	错误率(*E*)
35000	0.679	0
35010	0.928	0
35050	1.89	0
35100	1.906	0.17
35150	1.965	0.15
35200	1.805	0.30
35250	1.712	0.31
35300	1.732	0.33
35500	1.887	0.22
36000	1.835	0.30
36500	1.981	0.30
37000	2.322	0.29
37500	2.356	0.29
38000	2.625	0.29
38500	2.671	0.29

根据实验数据可以作出流程系统的性能曲线, 如图 2 所示.

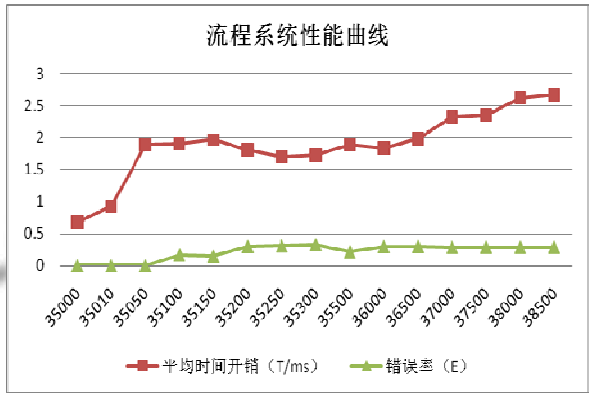

图 2 性能曲线

随着流程库数据规模的增长, 数据库 I/O 由于操作频繁而成为性能瓶颈, 这导致工作流引擎生成新流程的平均时间开销迅速增加, 与此同时, 系统生成新流程实例的失败概率也在增大.

基于上述对流程系统的性能分析, 可以看出, 在给定软硬件条件的情况下, 流程系统性能优化的关键在于控制数据规模. 下文将根据这一结论提出系统的性能优化方案.

3 优化方案

从上述分析中可以得出: 性能优化方案的核心思想在于把流程日志数据从日志库中移出, 从而减少系统的 I/O 操作频数. 因此, 本文采用的主要方法, 是通过SQL 语言和数据库自身机制为流程日志库建立工作流数据的归档机制.

归档机制包括历史数据库创建以及归档算法实现两大要点:

(1)历史数据库创建

根据流程日志库(K2ServerLog)的数据结构, 创建历史数据库(K2ServerLog_His), K2ServerLog_His 库中各数据表的的 Id 列使用 int 数据类型(不使用自增列), 该数据库用于存储流程日志库中状态为 CLOSE 的完整流程数据.

(2)归档算法实现

为了保证流程归档的完备性和可靠性, 采用迭代的方式编制归档算法, 即, 对单条流程归档进行功能封装, 实现单条流程归档功能的原子性, 并在此基础上实现流程归档的批量操作.

单流程归档与批量归档的伪代码如下:

SingleArchive:	BatchArchive:

SingleArchive(需要归档的流程 ID)
{
//基本参数初始化
//开启归档事务
BEGIN TRAN SingleArchiveTran
//对流程实例关联的所有表进行归档
@Error=ArchiveToHis(ID);
//如果归档失败, 回滚事务
If(@Error <> 0)//失败
ROLLBACK TRAN SingleArchiveTran
//删除日志库的数据
deleteLogData(ID);
//提交事务
COMMIT TRAN SingleArchiveTran
Log();//记录日志
}
	BatchArchive()
	{
	ConnectDB();//建立数据库会话
	OpenDB();
	//基本参数初始化
	//设置归档流程数
	SELECT @Count=COUNT(ID) FROM _ProcInst] WHERE status ='close'
	IDs=getIDs();//获取待归档流程 ID 集合
	Index=0;//指示器
	While(index < @count)
	{
	//每次批量归档 batchNum 条数据
	For(i=0;i<=batchNum;i++) {SingleArchive(IDs[i]);index++}
	Log();//记录日志
	}
	CloseDB();//回收资源
	}

上述算法中:

ArchiveToHis 实现将 K2ServerLog 中单条流程实例所关联(通过 ID)的所有表进行逐条归档, 与入 K2ServerLog_His 中对应的表.

deleteLogData 则在 ArchiveToHis 成功执行的条件下, 把 K2ServerLog 中对应的数据作物理删除, 从而减少流程日志库的数据量.

事务性描述: 为了保证流程归档的完备性, 在算法中加入了数据库事务特性, 实现单条流程归档的的原子性.

为了实现自动归档功能, 首先利用上利算法在数据库中建立SQL 脚本, 然后创建定期执行脚本的作业, 通过作业的定期运行, 把 K2ServerLog 库的流程日志定期自动归档到 K2ServerLog_His 库, 从而达到控制流程日志库的规模的目的, 最终通过减少了数据库的 I/O 操作实现流程性能优化的目标.

4 总结与展望

本文从数据层面对流程系统的性能作了分析, 通过SQL 语言脚本和数据库自身机制实现了一种性能优化的方案, 实现表明, 这种方案能够明显提升流程系统性能, 特别是在降低平均时间开销(T)和提交失败概率(E)两个性能指标方面, 具有一定的优势.

为了进一步提高工作流系统的性能, 在未来的研究中, 还需要对流程机制的其它方面作深入全面的研究, 本文认为, 对流程数据结构以及流程实例的生成机制方面的研究, 将是今后该领域的一个重要方向之一.

参考文献

1 李建强,范玉顺.一种工作流模型的性能分析方法.计算机学报,2003,26(5):513−523.

2 林闯,田立勤,魏丫丫.工作流系统模型的性能等价分析.软件学报,2002,13(8):1472−1480.

3 苏子林,赵轩臣.电力企业工作流管理系统的设计与实现.电脑开发与应用,2004,12:5−6.

4 范玉顺,吴澄.工作流管理技术研究与产品现状及发展趋势.计算机集成制造系统-CIMS,2000,1:1−7.