doi: 10.3969/j.issn.1006-1576.2012.02.024

Bigtable 系统主服务器检查点的实现

王金锁[1]，康林[2]，费江涛[3]，齐学玲[4]
(1. 中国人民解放军 63898 部队，河南 济源 454650；2. 总装驻绵阳地区军代室，四川 绵阳 621000；
3. 北京航天飞行控制中心软件室，北京 100094；4. 中国人民解放军第 306 医院，北京 100101)

摘要：为提高系统的故障恢复速度，通过设置检查点记录主服务器关键数据结构的状态，保存程序运行的关键数据，利用 Google File System(GFS)对检查点数据进行可靠存储，以此来实现主服务器的快速重启。应用结果表明，该方法有效地提高了系统的故障恢复速度。

关键词：Bigtable；检查点；主服务器；故障恢复
中图分类号：TP311.12　　**文献标志码**：A

Implement of Master Checkpoint in Bigtable System

Wang Jinsuo[1], Kang Lin[2], Fei Jiangtao[3], Qi Xueling[4]
(1. *No.* 63898 *Unit of PLA*, *Jiyuan* 454650, *China*;
2. *Representative's Office of General Armament Department Mianyang,* 621000, *China*;
3. *Software Office*, *Beijing Aerospace Control Center*, *Beijing* 100094, *China*;
4. *No*. 306 *Hospital of PLA*, *Beijing* 100101, *China*)

Abstract: In order to improve the failure recovery rate, analyzes the work principle of Bigtable, and compares several checkpoint algorithms, master of Bigtable records the states of key data structure by setting checkpoints, and saves the data into GFS. Using checkpoints, master can restart quickly, the method can increase failure recovery rate effectively.

Key words: Bigtable; checkpoint; master; failure recovery

0　引言

Bigtable[1]是 Google 公司开发的一个用于结构化数据管理的分布式存储系统，已经广泛地应用于 Google 公司的相关产品中。Bigtable 系统包括 3 部分：客户端库、一个主服务器和多个 Tablet 服务器。其中主服务器(master)的作用是管理整个系统的元数据。由于系统运行在普通的商用计算机组成的集群上，机器的可靠性和稳定性较差，同时只存在一个主服务器为系统提供服务，因此，当主服务器出现故障后，整个系统将不能正常运行。主服务器的故障恢复时间是影响系统整体性能的重要因素。

检查点机制是集群系统进行故障恢复的主要手段，计算机系统通过周期性地设置检查点，把程序在运行时的正确状态保存到稳定存储器中。如果在随后运行过程中发生故障，那么系统进行卷回恢复，从稳定存储器中读出前一个检查点时的正确状态．从该点继续执行，这样就避免了由于故障而导致的程序从头重新执行，从而有效地减少了系统故障恢复的时间花费。因此，笔者对主服务器检查点的实现策略和设置方法进行研究。

1　检查点技术对比

设置检查点操作需要增加系统的额外开销，开销太大将严重影响系统的性能。系统在故障情况下运行的开销主要包括 2 个部分[2]：一是系统在每个检查点都必须完成保存程序状态的操作，不可避免地要引入一定的时间和空间开销；二是因故障而引起的卷回，因为每次卷回都会损失已经完成的任务。

目前检查点算法主要是通过 2 种策略来减小算法的开销与延迟，减少检查点时刻所需保存的程序状态信息和提高检查点操作与程序运行的并行性。下面介绍几种优化的检查点算法。

1.1　增量算法

增量(incremental)算法[3]是一种在不删除原有检查点的基础上，利用页式虚存管理机制自动排除内存只读数据的技术。只读数据是指自从上次检查点保存后，数据值没有被修改过。在增量检查点方式下，把程序空间所有页面的访问权限都置为“只读”。如果程序在随后的运行中试图修改一个只读的页面，则会产生一个“访问非法”错误，此时由

收稿日期：2011-09-02；**修回日期**：2011-09-21
作者简介：王金锁(1984—)，男，河北人，硕士研究生，从事网络安全、系统软件研究。

检查点系统在一个表格中记录下该页面，并将它的访问权限重新设置为“读写”，使程序可以修改该页。当需要设置下一个检查点的时候，只需要保存处理器状态和那些所有在表格中有记录的页。

1.2 用户参与的内存排除算法

如果用户了解程序的结构，则让用户直接参与检查点的保留过程是最简单和最有效的方法，因为这样可以把保存到检查点文件中的一些与故障恢复无关的数据(如以后不再访问的临时变量或在以后先进行写访问的变量等)排除掉。显然，该方法对于那些大型数组和大的内存块时效果很好，但对于单个变量的处理，上述辅助操作会引入更大的开销，反而得不偿失。并且该方法的人为因素过多，如果用户错误地排除掉必要的数据，则会造成检查点不可恢复的严重后果。

1.3 检查点压缩

磁盘读写速度比内存读写要慢许多，若采用合适的数据压缩算法把检查点所要存储的信息写到磁盘之前在内存中先进行压缩，则可既减少时间开销又减少空间开销。不过使用压缩技术减少开销必须满足一定的条件：只有用于压缩的时间小于因数据量减少而节约的时间，使用检查点压缩才有意义。由于压缩比通常与具体的应用有密切关系，所以压缩方法的通用性不是很强。

1.4 主存算法

主存(main memory)算法[3]可以在很大程度上提高检查点操作和程序执行的并行性，降低检查点算法的延迟。主存算法的执行过程为：首先，冻结用户进程的运行，把进程运行状态复制到一个内存备份空间上；然后恢复进程的运行，同时启动一个专门的复制进程负责把内存备份空间的内容复制到磁盘上。该算法把检查点操作的延迟，从进程运行状态的磁盘复制时间减少到内存复制的时间。该算法取得良好效果的前提条件是机器的物理内存空间足够大，否则，内存空间与磁盘交换区之间的频繁对换会大大增加算法的延迟。

1.5 写时复制算法

检查点主存算法的有效实现需要大量的系统内存，而且在内存空间较小的系统上会明显降低系统的性能。写时复制(copy-on-write)算法[4]只复制需要修改的数据内容，避免了只读数据的复制，从而减少了内存的使用量。该方法是对主存算法的进一步改进，它使检查点的延迟减小到保存寄存器状态和置页面为“只读”方式的时间，或者是处理一次页面访问错误时间。

2 主服务器的工作原理

如图 1，Bigtable 系统包括 3 个主要组件。客户端库、Tablet 服务器和主服务器。其中 Tablet 服务器用于管理系统的数据块，处理用户对表数据的读写请求；主服务器用于管理系统的元数据、Tablet 数据块的分配情况和服务器工作情况；客户端库用于对数据进行缓存和提供统一的操作接口。除此之外，Bigtable 使用 GFS 存储 Bigtable 的海量数据，使用 Chubby 提供锁服务和对表结构数据的存储。

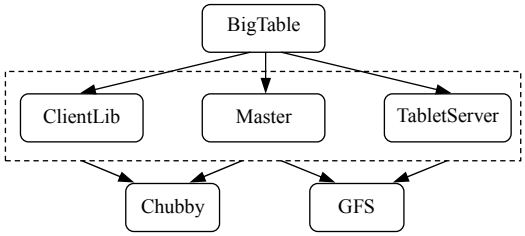

图 1 Bigtable 系统的体系结构

主服务器的功能包括：启动整个系统、创建和删除表、注册和卸载 Tablet 服务器、管理表结构数据、分配 Tablet 数据块和垃圾文件的回收。为了实现这些功能，Master 服务器需要保存的数据结构有：Tablet 服务器信息列表、表基本信息列表、未分配的 Tablet 数据列表和 Tablet 数据分配情况。在系统启动时，主服务器需要从不同位置读取这些数据信息，这个过程将花费大量的时间和系统资源。图 2 描述了这些数据系统的分布情况。

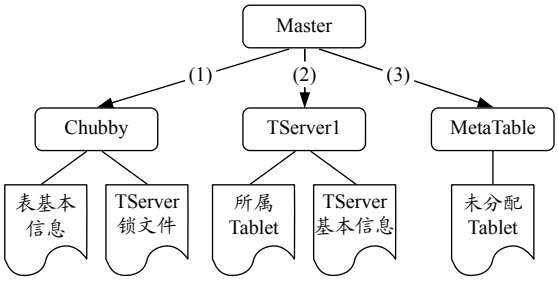

图 2 Master 所需数据的分布图

在主服务器重新启动后，它需要分别从 Chubby、TServer和元数据表(MetaTable)上分别读取数据，进行数据结构的创建和恢复。主服务器的重新启动流程如下：

1) 创建和 Chubby 的连接；初始化 Chubby 操

作指针和会话句柄。

2) 从配置文件读取属性信息。

3) 建立与底层文件系统 GFS 的连接。

4) 检查并创建 Master 的相关文件和目录。

5) 抢占 Master 的互斥锁。打开原 Master 服务器的锁文件，抢占互斥锁，如抢占失败，程序退出。

6) 从读取 Master 锁文件，获取最后的表 id(last_table_id)值。

7) 扫描 Tablet 服务器目录，将有效的 TServer 基本信息添加到服务器信息列表。

8) 和所有的 TServer 服务器进行通信，读取其状态信息；获取其所加载的 Tablet 列表。

9) 扫描元数据表，从 RootTablet 开始扫描，将未加载的 Tablet 信息放置到未分配 Tablet 信息列表中，读取未分配块的表 ID，Tablet 的起止行。当未加载的是元数据块时，先将其进行加载，之后继续扫描。

10) 启动库文件的扫描线程，将无效的目录和文件删除。

从主服务器的启动流程可以看出：从第 6)步操作到第 9)步操作，主服务器读取不同的数据进行数据结构的重建。它需要从 Chubby 服务器上获取表基本信息和所有 TServer 的基本信息；从每台 TServer 服务器上获取状态信息和所属 Tablet 的信息；从元数据表获取未加载的 Tablet 信息。这是一个非常复杂的执行过程，将耗费大量的资源和时间。

3　基于检查点的主服务器恢复

3.1　设计思路

通过对 Bigtable 系统的工作原理进行分析，笔者发现主服务器重新启动过程需要花费大量的时间，其中数据结构的重建是时间花费的主要部分。如果系统能够在正常运行时，不断保存这些数据，当主服务器重启时，直接从文件中读取这些数据，从而将数据结构恢复到故障之前的状态。这样系统就会减少因数据读取带来的时间和资源的开销，达到快速启动的目的。

笔者将检查点技术应用到主服务器的故障恢复中，可以提高其的重启速度。主服务器正常运行时不断设置检查点，从而将数据结构保存。主服务器设置检查点需要确定状态信息的存储位置和检查点的设置策略。

首先，如果主服务器出现硬件故障，机器上的数据可能在很长时间内不能使用，此时需要重新启动一个新服务器作为主服务器。如果将检查点数据保存在主服务器本地计算机上，这些数据将会丢失，无法进行有效的数据恢复。因此系统需要将检查点数据保存在稳定的存储器上。在系统中，Chubby 和 GFS 是支持系统正常运行的 2 个组件，它们都能够提供可靠的文件存储的功能。Chubby[5]在提供锁文件的同时也提供了小型数据的存储能力，锁文件最大可以存储 256 B 的数据，锁文件将会备份到 5 台服务器上，具有很高的可靠性。GFS[6]提供了对大文件进行并发存储的能力，具有高可靠性和较高的读写速度，系统中的每个数据块(chunk)大小为 64 MB，每个数据块被复制到 3 台机器上进行数据备份。对比 2 个存储组件，Chubby 由于其文件的存储能力太小，不能满足检查点数据存储的需要，因此系统使用 GFS 对检查点进行存储。

由于系统只提供一个主服务器，所有的客户端都将访问此服务器进行数据的读写，主服务器的执行效率决定着整个系统的性能。检查点的设置不能占用太多的资源和产生较大的时间延迟。为了达到这个要求，检查点的设置策略至关重要。首先，笔者借鉴增量算法，将检查点数据分步写入，使用内存复制策略降低时间延迟，利用有效的检查点压缩算法减少写入磁盘的数据。

3.2　具体实现

根据主服务器检查点的设置策略和数据存储方式，笔者对检查点的具体实现进行设计。

首先，为了使检查点需要保存的数据最少，要先分析主服务器管理的数据结构。如上所述，主服务器管理的数据包括：Tablet 服务器信息列表、表基本信息列表、未分配的 Tablet 数据列表和 Tablet 数据分配情况。笔者将这些信息进行分类合并得到如下 3 种信息(状态信息)：表基本信息(最大 ID 号)，TServer 相关信息列表(TServer 状态信息、所属表和 Tablet 数据块信息)、未分配 Tablet 数据列表。这样既可以降低数据的冗余，又可以使数据恢复速度提高。

然后，笔者分析主服务器更新状态信息的功能操作。在文中提到的主服务器功能中，下述功能可能会更新状态信息：创建和删除表、注册和卸载 Tablet 服务器、分配 Tablet 数据块。系统借鉴增量算法的思想，在检查点中只保存状态信息的更新情

况。如果对于每个操作都设置检查点保存状态信息到磁盘，开销将会很大，特别是系统需要将数据远程保存到 GFS 文件中。因此，系统分配一个内存空间用于保存这些状态信息的更新情况，当内存空间用完后，系统分配一个新的内存空间，同时冻结原内存空间，利用系统提供的压缩算法对数据进行压缩，并加上数据校验信息，写回到文件中。

4 结束语

笔者借鉴了增量检查点的思想，利用检查点技术对 Bigtable 系统的关键数据进行存储，有效地提高了服务器的重启速度。该方法保存了最少的状态信息，降低了检查点设置对程序运行带来的延迟，提高了数据的存储速度，降低了系统开销，在保证系统正常运行的情况下，提高了主服务器故障恢复的速度。

参考文献：

[1] Chang F, Dean J, Ghemawat S, et al. A Distributedstructured Data Storage System. In 7th OSDI (2006).

[2] 梁蓓, 等. 用时间序列分析方法动态确定检查点时间间隔[J]. 系统仿真学报, 2004, 16(10).

[3] 魏晓辉, 鞠九滨. 分布式系统中的检查点算法[J]. 计算机学报, 1998, 21(4).

[4] 李凯原, 杨孝宗. 检查检查点开销的一种方法[J]. 计算机工程与应用, 2000(2).

[5] Burrows M. The Chubby lock service for looselycoupleddistributed systems. In Proc. of the 7th OSDI, 2006(11).

[6] Ghemawat S, Gobioff H, Leung S T. The Google_le system. In Proc. of the 19th ACM SOSP, 2003(12): 29-43.

****

（上接第 84 页）

由图 7、8 可以看出：预测漂移与实际信号吻合较好，小波神经网络的收敛速度极快，当神经元个数为 10 时，仿真精度 MSE=4.874 41e-006，在网络权系数学习的过程中，为了加快训练速度，采用漂移数据对神经网络的权系数进行初始化，训练过程中消除了无效迭代，从而可以使网络输出很快地逼近期望输出。

3 结论

基于小波神经网络的陀螺漂移的预测模型利用小波分析对原始信号进行趋势提取，降低了人为因素影响，并采用小波神经网络避免了其它网络存在的局部最小化的缺陷。建模验证的结果证明了该方法有效，并显著地提高了漂移的预报效果。小波神经网络非线性预测算法为补偿一般陀螺的随机漂移提供了一种新的有效途径，还可为一般陀螺仪表的漂移模型建立方法提供借鉴。利用该算法对导航设备的输出进行预测，从而建立智能故障诊断系统，可以有效提高导航系统精度，具有较好的应用前景。

参考文献：

[1] 张良钧, 曹晶, 蒋世忠. 神经网络实用教程[M]. 北京: 机械工业出版社, 2005: 216-217.

[2] 李春鑫, 李天伟, 方建平, 等. 基于小波神经网络非线性预测方法的研究[J]. 光电技术应用, 2005, 20(2): 44-46.

[3] 蔡烽, 万林, 石爱国. 基于相空间重构技术的舰船摇荡极短期预报[J]. 水动力学研究与进展 A 辑, 2005, 20(6): 780-784.