分布式编译的方法和系统研究

童亚拉
(湖北工业大学 理学院,湖北 武汉 430068)

摘 要:软件项目规模越来越大,使大型软件的编译过程漫长,而分布式编译正是目前提高大规模软件项目编译时间及效率的有效方法,目前国内外在此方面研究的文献资料并不多见。以目前市场上两种主流的技术代表性产品 Incredibuild 和 Distcc 为例,详细描述了二者的性能特点、工作原理、框架结构、安装维护等内容,总结了分布式编译器系统的发展现态,分析比较了其优缺点,指出分布式编译系统性能提高问题本质上是一个优化求解问题,因而优化求解的诸多技术和方法均值得借鉴,这将是未来分布式编译器系统的一个发展方向。

关键词:分布式编译;incredibuild;distcc;优化求解
中图分类号:TP311.5 **文献标识码**:A **文章编号**:1673-629X(2010)05-0079-04

Design of Method and System of Distributed Compiling

TONG Ya-la
(School of Science, Hubei University of Technology, Wuhan 430068, China)

Abstract: The scale of software program is getting larger and larger, which causes long process of large scale software compiling, and distributed compiling is presently the available method to improve efficiency in large-scale software project and few literature are finded in this aspect. As an example of two representative products in market--Incredibuild and Distcc, described their knowledge in aspect of performance, operational principle, frame work, installation and maintenance including advantage and deficiency. Finally it also points out the fact that to improve the performance of distributed compiling system is namely to transform seeking optimal performance into seeking optimization solutions of problems which may be one of developing direction for distributed compiling in the future.

Key words: distributed building; incredibuild; distcc; optimizing problem

0 引 言

随着软件项目规模几何数字的增长,百万行代码的系统比比皆是,尽管现在计算机的运算速度不断提高,但大型软件的编译仍然过程漫长,例如某项目软件大小约为200k行,VC6下的编译时间为3分钟(P4 1.8G, 512M),交叉编译时速度更慢,这表明编译时间及效率成为项目开发的瓶颈。分布式编译系统通过提高编译速度有效而直接地提高前期调测的效率,极大地缓解了这种压力,得到市场认可[1,2]。

分布式编译的原理就是将编译的整个工作量通过分布计算的方法分配到多个计算机上执行,以获得极大地效率提升。由于分布式计算技术相对较成熟,各种分布式编译软件也较多,因此基于此原理的分布式编译器的设计具有可靠的基础[3]。一般地,分布式编译软件不是编译器,而是附在某个编译器上的分布计算管理软件,以此特定编译器实现分布式编译。目前市场上分布式编译技术主要分为两种:一种是插件技术,其代表性产品是支持插件技术的 IncrediBuilder,另一种是支持外部直接调用的 Distcc,文中通过详细介绍它们的性能、特点等,比较其优缺点,分析今后分布式编译软件的发展方向。

1 IncrediBuild

IncrediBuild是一个网格编译工具,它利用先进的网格计算技术加快 Microsoft Visual Studio 及其它基于 Windows 复杂过程的编译速度,如单独一台机器编译时需18分钟,而20台机器编译可下降到6分钟。IncrediBuild 采用多线程处理技术,不改变项目文件的代码,可无缝集成到 Visual Studio 开发环境中,有很强的命令行界面,大大增强开发环境,使开发人员可更快地完成工程项目,节省90%的工程开发时间。网格引擎是 IncrediBuild 的核心技术,网格计算是一种分布式计

收稿日期:2009-08-12;**修回日期**:2009-11-03
基金项目:湖北省2009年教育科学研究计划重点项目;湖北工业大学博士科研启动基金(BSQD0830)
作者简介:童亚拉(1966-),女,湖南桃源人,博士,副教授,研究方向为智能计算、分布式计算。

算,一个或多个过程的不同部分并行执行连成网络,根据参与机器的状态和有效性动态调整处理过程,保证处理过程会被正确执行,与机器环境、文件系统及安装基准无关[4]。

1.1 框架结构

IncrediBuild 主要包括两部分:IncrediBuild 协调程序(服务器)和 IncrediBuild 代理(客户),前者类似一个中央任务分派器,用来管理各个编译代理;它们将基于服务器和对等结构的协议相结合以实现健壮的、高效的、可扩展的目的。图 1 是 IncrediBuild 系统结构图。

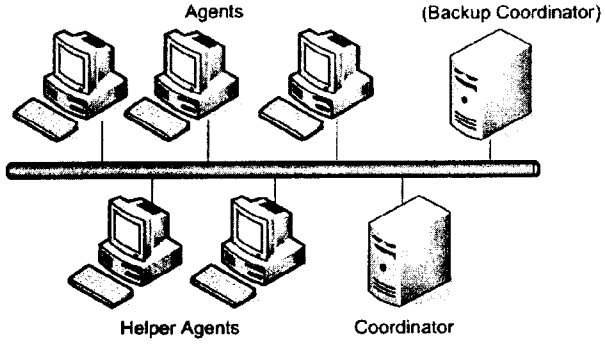

图 1 IncrediBuild 系统结构图

1.1.1 IncrediBuild 协调程序

IncrediBuild 协调程序是服务器,负责跟踪系统状态,分配计算资源给各代理。它根据各代理的软硬件性能与当前的有效性和状态,维护系统的动态和实时的"窗口信息",这些信息可用来确保每个作业被分配到最好资源。另一重要角色是连接代理的单一配置和管理中心,利用协调程序监视器可使版本更新、作业维护、用户设计修改等工作集中完成。

1.1.2 IncrediBuild 代理

IncrediBuild 代理是客户组件,负责跟踪系统的状态,分配任务给初始化分布事务。其最基本的功能是充当"帮助者"的角色,执行由其它代理参与作为远地计算资源而初始化的分布作业。不管远地代理的文件系统的安装基础和环境,虚拟技术确保每项任务能被正确执行。

输出文件与在本地初始化的文件系统相似。因为每个 CPU 能同时完成不同的工作,它还有助提高处理能力[5]。

1.2 性能特点

分布式编译系统可提高执行效率近 20 倍,图 2 描述了 Visual Studio 项目在编译时间上的提高。

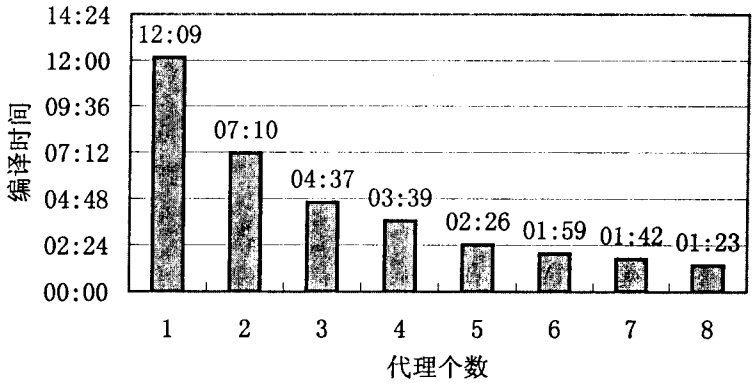

图 2 IncrediBuild 编译效率图

* 文件缓存。为了降低网络通信量,文件缓存保存最近访问的文件,在每个代理的文件系统产生。利用智能文件交换、硬件连接和元数据更新算法可使文件同步最小化。当文件作为分布式任务存储在网络设备上时文件缓存特别有用。

* 实时压缩。为进一步降低网络通信量,可利用实时压缩算法缩小数据。

* 理想的 CPU 操作。办公网站一般用来完成文本编辑、互联网冲浪和数据应用。因为这些工作所占用的 CPU 资源极少,所以通过分布式网络能为办公网站提供优良的资源,因此在办公网站上安装 IncrediBuild 有助于合理利用现有资源。

* 自适应的资源配置。因为网格计算可用性不断变化,采用自适应分配机制适应这种变化。一旦节点 CPU 或网络利用率降低到最小水平,它将自动停止担当网格远程资源的功能,未完成的任务转给其它可用性更高的节点。

* 多 CPU 或核利用率。代理可利用额外处理器和核。CPU 或核被当作独立的计算资源,每个均具有独立并行执行能力以充分利用计算机的硬件。

* 双重任务。当一个执行分布式任务的代理等待单个远程代理完成任务时,系统将作优化处理。满足条件时初始分布式任务的节点开始并行执行。若其中一个代理完成任务,代理的输出被用作分布作业,根据完成情况继续完全剩余任务。

1.3 安装与维护

维护和管理联网的计算机是一个复杂的问题。IncrediBuild 简化了公共维护工作如版本更新,并移去节点管理的任务。除了基本分布式网络计算软件的安装要求外,IncrediBuild 设置非常简单。初始化步骤包括安装一个协调器和在客户节点上安装一个 IncrediBuild 代理。协调程序监视器是系统的监视程序和配置中心,它提供全面的、实时的系统状态记录以显示每个代理的硬件状态、操作系统信息、性能变量和当前操作,也允许系统管理者设置修改重要代理。一旦系统

建立起来,版本更新很容易完成。协调程序监视器可对所有代理节点分配更新版本,无需手工安装。网络性能是一个非常重要的因素,"网络连接性测试"是每个代理的网络性能的基准,它可帮助网络管理者跟踪速度较慢的节点。

2 Distcc

Distcc 是一个程序,开发于 2002 年,是 GNU 的分布式 C/C++编译器,通过网上若干机器分布式编译 C、C++、目标 C 或目标 C++代码程序,将编译任务分布到网络中参与主机,产生如同本地编译一样的输出,但常比本地编译要快很多,并且很容易安装和使用。Distcc 能可靠并成功地编译大型的、复杂的和重量级的软件系统,具有很好的跨平台特性,支持 Linux, XFree86, CygWin 等平台,使用范围相当广泛。Distcc 不要求所有机器共享文件系统、同步时钟或安装同样的库和头文件。安装了交叉编译器后可使用不同的处理器和操作系统[6]。

2.1 工作原理

Distcc 的设计目的是与 GNU make 的并行编译(-j)选项一起使用。它本身不是一个编译器,只是用作 g++的一个前端,几乎 g++的所有选项都可按原样传递给 Distcc。对于每项任务,Distcc 发送预处理完的源代码到其他指定机器,守护进程确保编译在远程机器上完成,每个运行 Distcc 守护程序的机器安装了合适的编译器(最好版本相同)。Distcc 和 IncrediBuild 的差别在:Distcc 不使用仲裁者,直接由客户端对其他客户端发起编译请求,所以每个客户端都需知道其他客户端的位置,并且当多个客户发起编译请求时不易做平衡处理。

2.2 关键点

* 机器的配置必须一致。所有机器必须安装相同版本的 g++编译器及编译工具,如 ar、ranlib、libtool 等,操作系统的类型和版本也应相同。

* 由于客户机上 Distcc 将预处理代码发送给服务器,因此需验证守护进程是否在服务器上运行。

* 默认时 Distcc 在单台机器上的调度作业数是 CPU 的个数+2,单核机器是 CPU+3。对于命令行 make -j10 CC=distcc(其中只有三个主机),触发进程时最初触发编译作业数为 9。

* 需保证底层机器访问存储源文件必备的文件系统。基于网络文件系统的系统的一些源区域不能被挂载,否则将导致编译失败和网络堵塞。

* Distcc 在网络上编译源代码时链接可能不是并行的。

2.3 编译过程的监视

Distcc 安装了一个基于控制台的监视工具 distccmon-text。编译前打开一个单独的终端窗口并发出 distccmon-text 5,然后此终端每 5 秒钟显示网络多个节点的编译状态。表 1 为一个监视窗口示例。

表 1　distccmon-text 的输出

2167	Compile	memory.c	tintin[0]
2164	Compile	main.cxx	tintin[1]
2192	Compile	ui-tcl.cxx	asterix[0]
2187	Compile	traverse.c	asterix[1]
2177	Compile	reports.cxx	pogo[0]
2184	Compile	messghandler.c	pogo[1]
2181	Compile	trace.cpp	localhost[0]
2189	Compile	remote.c	localhost[1]

2.4 加快编译速度的方法

由于在 C/C++框架下修改头文件时一般基于 make 的系统最终会重新编译所有源文件,而更改头文件仅影响源文件的子集,因此不需要作耗时的编译清理。此外,用 ccache 可大大减少编译清理时间(为原来的五分之一至十分之一)[7]。工作方式是从预处理源代码和用于编译源代码的编译器选项创建一个哈希表。重新编译时,如果 ccache 在预处理源代码和编译器选项中未检测到任何更改,则检索以前编译输出的缓存副本。安装 ccache 时先下载,转到 ccache 目录后发出命令./configure -prefix=/usr/bin 以及 make && make install。如果 ccache 没有装在/usr/bin,则检查 ccache 的位置是否定义为 PATH 环境变量的一部分。可自定义一些 ccache 的环境变量,如 CCACHE_DIR 是用来指定 ccache 存储预编译输出的文件夹,默认文件夹为 E/.ccache 等。使用 ccache 时,可以带有 distcc,也可以不带。ccache 最简单的用法如下:ccache g++ -o <executable name> <source file(s)>。当它与 makefile 一起使用时,就会覆盖 CC 变量[8]。

3 分布式编译的发展现状

随着软件项目规模的增长,编译时间及效率成为项目开发的瓶颈,同时复杂度也明显增强。现在一个大型软件项目,常会采用多种编程语言开发,不同的编程语言完成不同的功能,而现有的编译方式往往首先需要将这些用不同编程语言编写的模块分别单独编译,这样大大增加了系统编译的成本和维护的复杂程度。而软件产业的快速发展要求对软件项目规模的增长以及对软件项目成本加以控制。因此研究分布式编译系统能同时支持多种编程语言的编译,成为未来分布式编译系统发展的一个重要课题。

目前,国内外对分布式编译系统支持多种编程器的研究尚处于起步阶段,还没有一个成熟的产品。IncrediBuilder 将每个编译器看作一个插件,各编译器自成体系,功能和操作在各自插件内部完成,优点是方便编译系统扩展,每增加一种编译器,只需要在系统中增加一个编译器插件。支持外部直接调用的 Distcc 只是一种最直接、简单的多编译器支持方法,效率不高。这些多编译器技术存在如下缺陷:

1) 由于将各种编译器看作单独的个体,每个个体之间无法做到资源共享和统一调度,使得有限的资源下各个编译器为获取资源相互竞争,导致系统效率低下;

2) 资源的获取及释放没有统一的调度缓冲算法,频繁的获取及释放资源成为系统效率的瓶颈。

在上述的背景下,研究分布式编译系统在有限资源环境里多编译器同时工作的编译效率问题,具有理论研究上的深远意义和现实应用中的巨大潜力。分布式多编译器效率低的直接原因是多个编译器同时工作时存在资源竞争的情况,为实现多编译器的资源共享,降低资源竞争冲突,必须在资源的申请、分配、注销上有统一的算法支持,这涉及到资源申请的优先级、资源空闲上下阈值、资源预留原则、资源互斥原则、资源注销的时机等因素,因此要求系统具有自适应自主演化的特点,即:

(1)资源分配上的可预见性。对可分配资源应该合理调度,动态配置可分配资源的数量;

(2)资源划分的智能化。要求资源划分尽量减少资源碎片,提高资源利用率。

鉴于此,分布式多编译器系统的诸多问题本质上可转化为优化问题,提高分布式多编译器系统性的性能,也就是将寻求分布式多编译器系统性最佳转化为优化问题求解,这将是未来提高多编译器性能的一个发展方向。

4 结束语

编译时间和效率问题是大型软件项目开发的一个瓶颈,分布式编译系统能极大地缓解了这种压力,并得到市场认可。文中介绍了目前主要流行的两种分布式编译系统 Incredibuild 和 Distcc 的技术性能,分析了它们的优缺点,指出分布式多编译器系统的诸多问题本质上可转化为优化问题,而优化求解的技术和方法层出不穷,已运用到各个方面,取得良好效果,今后要提高分布式多编译器系统性能也可借鉴这类经验。

参考文献:

[1] Beter. IncrediBuild 加速原理[EB/OL]. 2009 - 10. http://blog.sina.com.cn/s/blog_4cd5d2bb0100gmgx.html.

[2] Elbeltagi E, Hegazy T, Grierson D. Comparison among Five Evolutionary - based Optimization Algorithms[J]. Advanced Engineering Informatics, 2005(19):43 - 53.

[3] Tanenbaum A S, Streen M V. Distributed Systems: Principles and Paradigm[M]. [s.l.]: Prentice Hall, 2004.

[4] Xoreax Ltd. High - Performance Grid Computing[EB/OL]. 2009. http://www.xoreax.com/features_main.htm.

[5] 李西宁. 分布式系统[M]. 北京:清华大学出版社,2006.

[6] distcc: a fast, free distributed C/C++ compiler[EB/OL]. 2006. http://distcc.samba.org/.

[7] 程 巍,陈 前. 神经网络状态监测和诊断的分布式实现[J]. 计算机技术与发展,2006,16(7):111 - 113.

[8] 陈 涛,陈启买. 分布式计算机系统负载平衡研究[J]. 计算机技术与发展,2006,16(5):33 - 35.

(上接第 78 页)

据分析技术的主要功能模块的实现,并且在某学生信息管理数据库系统中进行了尝试性的应用。相信该课题的研讨对目前 IT 业界所关注的商业管理分析/商业智能决策在国内中小型企业中的应用与推广颇具现实意义。

参考文献:

[1] 林 宇. 数据仓库原理与实践[M]. 北京: 人民邮电出版社,2003.

[2] 施伯乐. 数据库与智能数据分析[M]. 上海: 复旦大学出版社,2002.

[3] 李启炎. 企业商业智能教程[M]. 上海: 同济大学出版社,2007.

[4] Balsters H. Database schema evolution and meta - modeling [M]. [s.l.]: Springer, 2001.

[5] Delaney K. Microsoft SQL Server 2000 技术内幕[M]. 北京:北京大学出版社,2002.

[6] Geiger K. ODBC 深入剖析[M]. 北京: 电子工业出版社,1996.

[7] 张玉芳,熊忠阳. 数据仓库数据模型的设计[J]. 计算机应用,1999,19(9):10 - 12.

[8] 王 珊. 数据仓库技术与联机分析处理[M]. 北京: 科学出版社,1998.

[9] Arnold K, Gosling J, Holmes D. THE Java™ Programming Language[M]. 4th Ed. [s.l.]: Addison Wesley Professional, 2005.

[10] 周 龙,郑 诚. 一种 OLAP 应用系统的设计与实现[J]. 计算机技术与发展,2006,16(6):101 - 103.