Runtime 系统综述*)

A Survey of Runtime Systems

张宏莉　胡铭曾　方滨兴
(哈尔滨工业大学计算机系　哈尔滨 150001)

Abstract Runtime systems play an important role in parallel programming and parallel compilation. In this paper, goals and key techniques of runtime systems are presented. And some experiences and its trend are given in the end.

Keywords Parallel compiling, Runtime, Message passing, Multithreads

1 Runtime System 概述

自 70 年代中期以来，并行处理一直是推动计算机发展的主流技术[1]。经过二十年的发展，超级计算机的速度以从亿次提高到万亿次。但实践表明：其实际运行速度仍在一亿次左右徘徊，究其主要原因在于并行软件已成为发挥并行计算机性能的瓶颈，尤其是系统软件，如并行编译器。

因此研究高性能的并行编译器已成为国内外研究的热点。其中，作为并行编译系统(如图 1)的重要组成部分，Runtime System(简称 RTS)介于并行语言编译器与计算机硬件之间[1]，更进一步地说，是介于并行语言编译器与操作系统之间。对上，它是并行语言编译器的运行支撑环境；对下，它是计算机操作系统向用户方向的延伸。它旨在充分利用计算机的硬件资源，为并行编译器开发者或并行程序员提供高效、便捷、可移植的、模块化的编程及运行环境，

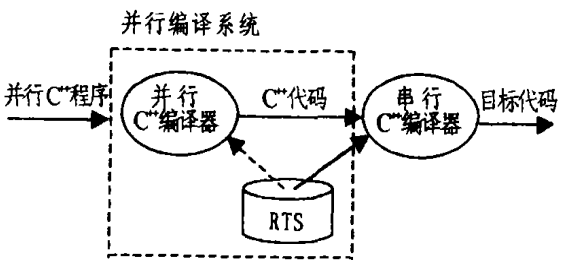

图 1　RTS 在并行编译系统中的地位

即在功能上服务于并行语言编译器，在性能上追求并行加速比。简言之，Runtime 对并行程序加速比的提高至关重要。

下面我们将从两方面分析 RTS：功能和性能。即分析 RTS 为上层并行语言编译器提供的主要功能和为提高性能而采取的关键技术，分别回答“What to do”和“How to do”的问题。最后，给出前人的开发经验和我们未来的研究方向。

2 RTS 的功能

由 Runtime System 的地位与作用可以看出：在功能方面 RTS 与并行语言编译器密切不可分，所以我们首先从并行语言编译器的研究现状说起。自八十年代以来，并行语言的研究与开发蓬勃发展，被并行的语言涉及逻辑推理、科学计算、事务处理等诸多方面，目前林林总总已不下几十种，现就几种有代表性的语言列表比较如下，见表 1。

从被并行的语言来看，除逻辑语言 Lisp 等以外，主要分为 Fortran 和 C++ 两大流派。Fortran 一直是科学计算领域的主导语言，在经历了若干并行化的尝试以后，推出了 Fortran 90, Fortran D, Fortran 90 D 等版本，最后这些开发者们组织在一起制定了并行 Fortran 的标准——HPF[2]。它体现了目前最成熟的数据并行思想，其主要特色如下：在语言级提供数据划分、数据对准、并行循环的指导，在编译时为所有节点生成相同的节点程序。这是典型的

*)本文受国防科工委九五预研项目资助。张宏莉　博士生，主要研究方向为并行编译技术、面向对象技术。胡铭曾　教授，博士生导师，主要研究方向为并行体系结构、并行处理。

分布存储即SPMD的编程风格。并行化的实现主要在编译时完成(包括同步),运行时主要解决通讯问题。

相应的RTS功能较为单一,以通信为主,故又称之为通信库。其中,有的是厂家专门开发的,如:IBM的EUI,Intel的Nx,Meiko的CS,nCUBE的PSE,Thinking Machine的CMMD等;有的是专门的研究项目,如:P4,PVM,Zipcode;也有商用产品如:由Parasoft公司推出的Express。在94年,来自工业界、政府、学术界人士联合开发的消息传递接口标准:MPI[3],按此标准实现的通信库如:MPICH。另外还有一些为解决某些专门问题,如不规则计算,而建立的RTS:马里兰大学的CHAOS[4],TreadMarks等。

表1 几种有代表性的语言比较

并行语言	开发年代	开发者	编程模式	主要特色
Actors	1986	MIT	共享存储	开控制并行之先河,异步通信
Linda	1988	Yale大学	共享存储	控制并行,基于模式匹配的异步通信方式
pC++	1990	Indiana	分布存储	数据并行思想与C++语言的巧妙结合
HPF	1995	HPF工作组	分布共享	成熟的数据并行语言,并行Fortran标准
Mentat	1993	Virginia	共享存储	对象内、对象间并行
HPC++	1996	HPC++工作组	共享存储	制定中的并行C++标准
CC++	1992	CalTech	共享存储	控制并行,多线程支持

与分布编程风格相对应的是共享编程风格,它适于解决诸如事物处理等异步性较强、结构较复杂的问题,并行C++语言尤其关注此类问题。

C++语言具有其良好的封装性、继承性,为并行化提供了有利的支持,所以一出现就成了并行化的热点。相比之下,并行C++关注的问题要多得多,除了数据并行,还包括控制并行、不规则计算问题等,而且运行环境也不单纯是并行机,因其良好的性能价格比工作站机群,LAN也跻身于并行硬件环境之列。由此产生的并行C++语言更是异彩纷呈、门类繁多,如COOL,ABC++,PC++[5],CC++[6],AC++,CHARM++,Mentat,MPC++,等等。在语言一级大多都定义了专门的对象类,如全局/远程对象类、Collection类、共享类,这相当于将任务做了分类,运行时各节点机各司其职,由此带来了动态调度、负载平衡、异构处理等依赖运行时解决的问题。与并行Fortran类似,人们也希望融合各种并行思想于C++语言,达成并行C++的语言标准——HPC++[7]。目前已经拟定了RTS模型,语言标准尚未形成。

相应的RTS功能较为综合化,涉及并行任务的调度、并行任务间的通信与同步、异步处理等。按实施并行的基本单位可分为两类:基于进程的RTS,如并行语言COOL、Orca、CHARM++的RTS,和基于线程的RTS,如Nexus[8]、pSystem、Cilk、Split-Threads、TAM等。对于上层编译器,它们大多支持以对象为单位的并行编程,定义有远程对象类、共享对象类等。在操作系统级构成了一类特殊的以软件方式实现的基于对象的、松散耦合的DSM(Distributed Shared Memory)[9]。

3 RTS的关键技术

在这一部分我们将主要分析现有RTS为提高性能而采取的关键技术,包括通信技术、同步技术、调度技术、以及多线程技术。

3.1 同步与通信技术

并行的关键是将一个程序分成多个程序由多机执行,那么多个程序之间必然要发生同步与通信,所以同步与通信是RTS的关键技术之一。首先我们来看一下在不同硬件体系结构上并行带来的同步与通信问题,然后再分析其中的关键技术。在共享存储系统上,所有进程共享一个存储空间,进程之间通过访问同一结构而达到通信的目的。在访问共享资源时,多个进程发生竞争。所以同步代替了通信。这很类似于多道程序的操作系统,解决方法也多同出一辙,如互斥锁、临界区、管程、信号灯等,此处不再赘述。目前,采用共享存储(MPMD)编程风格的并行语言有:ActorS(1986,MIT),Mentat(1993,Virginia),CHARM++,CC++(1992,CalTech),等等。

对共享存储系统而言,其缺点是硬件实现比较复杂,扩展性较差;优点是用户编程简单。分布存储系统则恰好相反,每个节点都有各自的局部存储器,所执行的程序和数据都分布在局部存储器中,数据的一致性由程序员负责,此时通讯替代了同步。通常采用两类通信方式[1]:

1)双边通信方式。收发双方必须先在时间空间

上都准备好，互相间"全知全觉"。特点是，通信与同步结合在一起，即同步在编译或编程时完成。但当网络传输时间较长时，往往比较浪费时间。如 send-receive 通信原语，通信双方分别执行 send()，receive()函数。根据收方函数的接收方式，又可分为阻塞接收与非阻塞接收。收方接到发方的消息即返回，否则等待，该情况称为阻塞接收；收方接不到发方的消息也返回，该情况称为非阻塞接收。以上两种方式在 PVM 都已有实现。对于用户，这是一种双边操作，send、receive 函数必须在两个进程中配对使用，编程较复杂。单边通信方式的出现改变了这一点，它为用户提供了使用简单的单边操作。

2)单边通信方式。收发双方不必同时准备好，收方异步地接收并处理发方的消息，可通过硬件中断响应或软件轮询方式实现。特点是，通信与计算结合在一起。根据收方响应方式的不同，又有 RPC，Active Message[10]，Message-driven 等。消息由远程函数名、函数参数、发方地址组成。RPC 的目的是用户的远程过程调用同本地调用一样，已用于支持 Client-Server 网络服务。RPC 全靠软件实现，故具有一定通信延迟；倘若有硬件的参与，以硬件中断方式检测消息的到来则可大大降低通信开销，于是出现了 Message-driven 与 Active Message 通信模式。二者的主要区别是：接到消息后，在 Active Message 中，收方激活称为 handler 的函数提取出消息的各部分，然后派生新的进程处理消息，该方式适于开发细粒度并行性，已为 TAM 采用，Nexus 也采用了类似 Active Message 的方式——Remote Service Request 实现通信；而在 Message-driven 方式下，收方提取出消息的各部分，然后直接对消息做以响应，该方式要求有较大的缓冲区，适于开发粗粒度并行性，已为 Concurrent Smalltalk，CHARM^{++}采用。

3.2 调度技术

根据并行加速比的计算公式：如果有 p 个处理器参与并行，每个处理器的计算时间为 T_p，总的通信时间为 T_c，则在不考虑负载平衡的情况下，p 个处理器所能得到的并行加速比为：

$$S_p = p * T_p/(T_p+T_c) = p/(1+T_c/T_p)$$

最终，S_p 的大小由 T_c/T_p 决定，这里我们又称 T_p 为并行任务的粒度，它是调度的主要依据之一。下面，我们首先讨论粒度与并行性的开发。

粒度是衡量软件进程或线程所含计算量的尺度[1]。最简单的测量方法是数一下颗粒(程序段)中指令的数目，颗粒的规模决定了并行处理的基本程序段，一般用细中粗来描述，这与处理级别有关。

针对不同粒度级别的任务，应采用各自适合的开发方法，才能得到理想的加速比。一般来讲，细粒度并行性常在指令级、循环级上借助于并行化或向量化编译器来开发，任务或作业步级中粒度并行性的开发要求程序员及编译器一起发挥作用。开发程序级的粗粒度并行性主要取决于高效的操作系统和所用算法的效率。

目前，大规模并行性常在细粒度级上开发，如 SIMD，MIMD 机上开发的并行性，粒度越细，并行潜力越大，通信和调度的开销也越大。相应的 RTS 有：Cilk，Nexus，TAM 等。消息传递多计算机已用于中粒度和粗粒度计算，工作站群机网络即属此范畴。相应的 RTS 有：Jade，PVM，ActorSpace 等。接下来，我们分析两种已有的调度策略：静态调度策略，动态调度策略。

静态调度策略：即采取均分方案，将多个相同粒度的任务平均分配到物理节点机上。静态调度策略在编译阶段生成节点程序时实现，在单用户的、各节点计算能力相当的计算机系统上能得到较好的并行加速比，对大规模的规则的矩阵计算问题效果尤为显著，已为 MPI 所用。

动态调度策略：当硬件环境比较复杂，各节点机具有不同的计算能力、负载状况时，静态调度策略显然无能为力了，由此产生了动态调度策略。即，根据各节点机运行时负载轻重，分别分配以不同粒度的任务。动态调度策略在运行阶段实现，具有一定的延迟。显然根据前面的分析，工作站群机系统必须采用动态调度策略。目前已为 Dome 采用。

3.3 多线程技术

多线程已被广泛应用于实现动态、异步、并发程序，尤其适合开发细粒度并行。多线程优于传统的进程，主要在于线程共享共同的进程地址空间，上下文切换开销较小。

IEEE 已制定了多线程的标准：POSIX 1003.4a[1]，在 FSU-threads 线程包中已有实现。另外的线程包还有：C threads，Quickthreads，DCE Threads，等等。目前很多操作系统都已支持多线程，如 Solaris，SunOS，AIX，等。

在 Runtime 系统中采用多线程主要是出于功能和性能的考虑。性能方面，多线程能屏蔽计算-通信延迟。在传统的进程中、通信-计算-通信循环导致了空闲周期，但用多线程则可以高效率地使用 CPU，即一个线程阻塞了，又转而执行另一个线程，

且这种开销比进程少得多。Sundraw 等人的初步实验表明，采用多线程可提高通信性能 35%[11]。

从功能角度看，基于线程的模型有两点优势[7,11]：①基于小粒度的数据分解可以被无效率损失地实现。这对通信间计算量较小的问题如树搜索算法、整数计算、数据库查询系统格外重要。②这种模式与 Client-Server 计算很靠近，服务可通过使用线程登记机制输出、用类似 RPC 的方式调用；这种办法对非数值计算应用程序很有用，尤其那些数据库和事务处理领域。

目前，支持多线程的 Runtime System 有：ABC++的 RTS，pSystem，TPVM，Awesime，Panda，Concert，Presto，Nexus 等。

结论 从以上几方面的综述中我们还得出如下经验：

经验一：相近的通信模型与并行编程模式利于并行性能的提高[12]；

经验二：异构处理是有代价的——并行加速比的下降。如在 Nexus 中，当采用三种不同的低层通信库通信时，并行执行时间增长了 2-3 倍[13]。

经验三：系统的可伸缩性是影响并行性能的主要因素之一，差的可伸缩性将限制大规模问题并行性的开发，失去并行的意义[1]。

经验四：在多用户或异构环境下，实时的负载平衡策略对获得较为理想的并行加速比尤为必要。

经验五：多线程对开发细粒度并行效果好，甚至在粗粒度多处理机上[13]。

但到目前为止，还没有通用的在功能上同时为数据并行、任务并行提供支持的、针对工作站群机系统的高性能 R untime System 的研究成果见诸文献，尽管这已成为当前的研究热点之一[7,14]。

我们将研究在工作站群机系统上通用的高性能 Runtime System，它主要涉及以下问题：整体执行模式问题、通信开销问题、动态调度问题、性能的进一步提高等。

参考文献

1 Hwang. 高等计算机系统结构：并行性，可扩展性，可编程性. 见：王鼎兴等译. 北京：清华大学出版社，1995

2 High Performance Fortran Form. High performance Fortran language specification. 1995

3 MPIF. A Message Passing Interface Standard, May 1994

4 Ponnusamy R, Saltz J. A Manual for the CHAOS Runtime Library. UMIACS University of Maryland, May 1994

5 Gannon D, Lee J K. Object-Oriened Parallelism: pC++ ideas and experiments. In: Proceedings of Japan Society of Parallel Processing. 1991. 113～123

6 Chandy K M, Kesselman C. CC++: a declarative concurrent obejct oriented programming notation. Research Directions in Object Oriented Programming, MIT Press, 1993

7 Gannon D, et al. HPC++: A Draft White Paper. In preparation. 1995

8 Foster I, et al. Nexus: Runtime Support for Task Parallel Programming Languages. In: Intl. Workshop on Parallel Processing-1994. India, 1994. 12

9 Tanenbaum A S. Distributed Operating Systems. Prentice Hall, 1996

10 Eicken T V, et al. Active Messages: A Mechanism for Integrated communication and Computation. In: Proc. of 19th Intl. Symposium on Computer Architecture. May 1992. 256～266

11 Draft Version, TPVM: A Threads-based Interface and Subsystem for PVM. 1994. Available at: http://uvcs.cs.Virgnia.edu/~aifij/tpvm.html

12 Spertus E, Dally W. Evaluating Locality Benefits of Active Messages. ACM SIGPLAN Notices, 1995, 30 (8): 329～351

13 Neves R, Schnabel R B. Threaded Runtime Support for Execution of Fine Grain Parallel Code on Coarse Grain Multiprocessors. J. Parallel and Distributed Computing, 1997, 42: 128～142

14 PORTS Form. PORTSO, 1994