编译技术年度进展报告㊀

董渊[1+]　冯晓兵[2]　王生原[1]　陈文光[1]

[1]清华大学计算机科学与技术系，北京 100084

[2]中国科学院计算技术研究所，北京 100190

摘　要

编译技术是信息产业链中沟通处理器和应用程序之间最为关键的环节，对于整个产业的可持续发展具有重要的战略意义。任何高级语言程序都需要经过编译处理才能够运行，而任何处理器性能的充分发挥都必须依赖编译优化技术的支持，因此处理器和编译器的发展相辅相成。未来10年整个计算机领域面临的两个巨大挑战——多核/众核处理器编程和复杂软件系统安全性/可靠性中，编译技术研究都将会起到至关重要的作用，因此具有50多年发展历史的编译方向依然是国际上极为活跃的重要研究领域。同时，编译技术也是当前国家战略规划中急需发展的方向——高性能通用处理器和基础软件的关键支撑。近年来，我国在编译领域取得了长足进步，本文从学术研究、产业发展、软件平台和教育等几个方面给出国内这个领域的情况报告，并给出未来发展展望和建议。

关键词：编译，优化，编程语言，开源软件，教育

Abstract

Compiler technology is the most critical aspect of the link between processors and applications in the information industry chain. Any program in a high-level language need compiling prior to running, and the full performance of any processor relies upon the support of optimizing techniques. Therefore the development of processors and compilers complement each other. Compiler technology will play a role of vital importance in both major challenges to the whole field of computers in the upcoming decade, multi-core programming and security/reliability of complex software systems. Compiler technology, which has a history of development of more than fifty years, remains an area of active research in international academy and industry. Meanwhile, compiler technology is also a key support for high-performance general-purpose processor and basic software, which is in urgent need by the strategic planning of China. China has achieved significant progress in compiler technology in recent years. This paper gives a report on this area in China in aspects of academic research, industrial development, software platform and education, as well as prospects and suggestions for future developments.

Keywords: Compiler, Optimization, Programming language, Open source, Education

---

㊀ 本文工作得到国家自然科学基金（项目号90818019）、国家高技术研究发展计划（863）（项目号2008AA01Z102）和国家核高基重大专项（项目号2009ZX01036－002）的资助。

编译技术和程序设计语言是信息时代软件系统的核心基石。截至2010年，在有“计算机界诺贝尔奖”之称的“图灵奖”43年历史中，约1/3的获奖都在“编译技术和程序设计语言”这个领域，占据绝对领先地位[1]。本领域顶极会议PLDI（ACM Programming Language Design and Implementation）[2]在根据引用数量得到的CiteSeer全部计算机学科会议和期刊影响力排行榜中高居第三[3]。

未来10年整个计算机领域面临的两个巨大挑战——多核/众核处理器编程和复杂软件系统安全性/可靠性中，编译技术研究都将会起到至关重要的作用[85]。编译技术也是高性能通用处理器和基础软件等我国当前国家战略规划急需发展方向的关键支撑，在“核心电子器件、高端通用芯片及基础软件产品”等国家科技重大专项中均有部署。近年来，我国在编译领域取得了长足进步，2009年国内学者先后组织召开了“多核环境下的编译技术研讨会”和“编译课程教学研讨会”[4]等会议，交流编译技术研究进展和教育问题，探讨进一步的发展方向。本文在这几次讨论的基础上整理成文，从学术研究、产业发展、软件平台和教育等几个方面给出国内这个领域近年来的进展情况报告，并初步探讨未来的发展方向和建议采取的措施。

1 编译技术面临的问题

计算机软件逐步渗透到日常生产和生活的各个环节，高端服务器系统和移动嵌入设备构成云计算时代的计算主体，高端服务器普遍采用多核、众核体系结构，同时应用程序日益复杂化、多样化。如何开发、验证软件，保证程序正确、高效地执行，成为大家普遍关注和深入研究的热点。

编译技术面临着来自于三个方面的问题：

1）如何高效开发应用程序？高效而准确地采用计算机语言来描述现实世界的问题是软件开发人员对语言设计人员和编译器开发人员提出的要求，而针对特定应用领域的编程模型是解决这类问题的有效手段。

2）如何提高程序运行效能？在编译过程中采用各种优化方法，全面提升软件性能几乎是编译器诞生以来一直不变的主题，而随着电池供电移动设备的普及，针对代码体积和功耗的优化也逐步成为大家日益关注的问题，因此研究人员正在面临着综合考虑多目标优化以提高计算效能的难题。

3）如何保证程序值得信赖？应用程序的规模和复杂性不断增加，与此同时，运行环境的复杂性也日益增加，多核系统中程序并发执行带来一系列新的问题，如何解决这些问题成为大家关注的又一个问题，而检测工具和形式化验证成为解决这个问题的重要手段。

2　编译技术前沿研究进展

2.1　面向多核架构的编程模型

随着工业水平的不断提高，多核处理器日益普遍。尤其是异构多核，由于它能够在特定领域中针对应用的特点充分发挥不同处理器核的作用，成为当今处理器的主流架构。异构体系结构已成为未来大规模计算硬件的基础，可能包含多核、众核、专用加速设备，多层次的内部互连和结构并行，使得传统 MPI[5]、OpenMP[6]、OpenTM[7]、UPC[8]等单一的消息或者共享的编程模型难以清晰描述各个计算部件的协同关系，而 CUDA[9]、OpenCL[10]等编程模型又存在编程要求高、产能低、调试难、兼容性差等不足。结合应用和硬件架构的新变化，以上述模型为基础开展众核并行编程模型及语言的研究是最近几年编译技术的一个重要方向。

2.1.1　基于 MPI 的并行编程模型

MPI 并行编程模型方面，针对多核平台呈现出多资源的共享竞争和明显的 NUMA 特征，中国科学院计算技术研究所研究人员完成了三个相关的工作。一是对 MPI 程序采用两阶段的度量和调度。度量代码插桩以获得带宽需求爆发的代码段、调度代码插桩进行互斥的并行调度。度量开销低（10% 以内），但是调度的收益不明显，说明调度的粒度过大。二是实现了基于通信切片的 MPI 优化进程映射工具[11]。利用切片程序抽象方法，大大降低了通信模式收集的时间。针对 MPI 应用中普遍使用的迭代算法，该课题组提出激进切片的方法，来保证较高的裁剪度，实验表明激进切片能保持很高的通信拓扑相似度。三是在 MPI 程序的检错方面，运行时方法是比较实用的技术，但是对于死锁检测，普遍存在集中式检测的扩展性问题。提出通信模式循环不变的概念，以及与检测第一迭代相等价的充要条件，并给出了相关的编译分析和运行时验证技术，该技术在一个运行时检测工具和模型检测工具上获得实现，得到几十倍的性能提升[12]。

2.1.2　基于 OpenMP/OpenTM 的并行编程模型

数据重用模型是局部性优化技术的基础，对缓解存储墙问题起到了重要作用。经典的数据重用模型只面向串行程序，随着多核处理器技术的不断发展，并行程序中的局部性优化技术变得越来越重要。吴俊杰等人将经典的数据重用模型扩充到并行领域，分别提出了面向 OpenMP 和 OpenTM 应用的并行数据重用模型[13~18]。针对重用在线程、事务中的关系，系统地讨论了并行应用中重用的分类、判定和求解方法。应用这一模型研究了 OpenTM 循环的优化技术，降低了事务回退的风险。并行数据重用模型给出了并行程序中重用性的定量分析方法，对面向共享存储结构的并行程序分析和编译优化技术的研究具有重要的指导意义。

尉红梅等人根据众核结构特点和应用需求重点研究了加速编程模型，加速编程模型

以加速编译指示为核心，该编译指示以 OpenMP 为基础，增加了发挥众核特性的扩展[19]，用户在应用中插入合适的加速编译指示以标识核心代码，由高层编译生成适合众核架构的高效代码。

随着并行应用复杂度和规模的扩大，非规则核外应用和数据重分解已经成为程序员生产率和应用性能提高的主要瓶颈。充分利用 OpenMP 易于编程、支持增量并行的特点，扩充其并行计算模型支持非规则核外和数据重分解，并在分布式存储系统上进行并行编译通信优化研究具有重要意义。鉴于此，胡长军等人扩充 OpenMP 编程模型、执行模型和存储模型支持分布式存储系统上的数据重分解和非规则核外应用[20,21]。在编程模型方面，扩充 OpenMP 规范定义了处理单元组、模板数组、数组重对齐与数组重分布、核外数组等指导语句来描述数据重分解和非规则核外问题。在执行模型方面，采用扩展的 Fork-Join 模型支持分布式存储和核外应用。在存储模型方面，提出“部分数组共享”存储模型的通信优化技术。在“部分数组共享”存储模型基础上提出的共享数组识别算法，具有算法输入集小和识别信息更全面的特点。首次提出了基于该模型的一系列针对底层 MPI 代码生成的优化，如将零散的 MPI_Send/Receive 操作用 MPI 组通信进行替代，开发了非阻塞的组通信来保障通信/计算重叠，利用周期性来为多维数组打包/解包等。该研究小组提出面向非规则核外应用的通信与 I/O 隐藏技术[22]。首次通过给出 INSPECTOR/EXECUTOR 模型各个组件的相关性分析，将 INSPECTOR 阶段进行分解并提出相应的流水化转换策略。该策略与传统的 INSPECTOR/EXECUTOR 模型相比，不仅能够更有效地重叠通信、I/O 和计算，而且没有增加任何附加的存储空间。还提出面向非线性引用的通信生成技术[23]，将整数格与代数方法相结合提出了编译时通信集和相应 SPMD 代码的生成策略。提出面向数据重分解的通信调度优化[24]，采用通信表和通信调度表来描述数据重分解的实际通信状况。首次将周期性理论引入到数据重分解相关的通信调度优化，并给出相应的通信调度生成算法。该算法不仅减少了通信调度的生成时间以便应用到编译时，而且还有效地避免了通信冲突和减少了实际通信时间。

大规模科学计算中非规则、非结构化并行日益普遍，OpenMP3.0[25]的 task 结构的出现为有效地解决该类问题提供了可能。如何利用现有的 OpenMP3.0 编程模型降低 Cell 处理器[26]这一典型异构多核的编程难度，提高非规则、非结构化并行应用的效率，成为学术界和工业界急需解决的一个问题。胡长军等人实现了一套异构多核 Cell 处理器上支持 OpenMP3.0 的运行时系统。考虑到异构多核本身的优势，利用现有 OpenMP3.0 编程模型降低异构多核处理器的编程难度，提高编程效率；为了对非规则应用进行优化，构建了一个面向异构多核 Cell 处理器的软件 cache 模型，还实现了一个面向非规则应用的软件 cache 的预取方案；建立一套异构多核 Cell 处理器上面向 OpenMP3.0 的 task 的任务调度策略，提出了一套集成广度优先和工作优先的调度策略，既减少了 PPE 与 SPE 之间的通信，又进一步实现了负载均衡。

2.1.3 基于 UPC 的并行编程模型

伯克利的 UPC（Unified Parallel C）是一种全局地址空间语言，是当前国际上并行语

言研究的热点之一。中国科学院计算技术研究所的科研人员基于 UPC 源代码开发了面向国产曙光系列并行机的 UPC 编译系统，并进行层次并行特征扩展，CUDA 集群、Godson-T 平台和曙光的实验结果均有良好表现。自 2006 年起，该课题组开展了分割的全局地址语言和相关编译技术的研究。其目标是针对基于多核处理器芯片的高性能计算机上片内、节点内和节点间的三级并行体系结构，研究能统一多层次并行的并行程序设计模型。2007 年到 2008 年，基于伯克利的 UPC 开源编译器，课题组为曙光 5000 研制了 UPC 编译系统，提供了细粒度访存的多方面支持。第一，利用曙光机器节点内的硬件全局地址空间(GAS)，将 UPC 程序中的远程细粒度通信转换为远程 load/store 指令，显著地降低通信延迟；第二，针对远程 load/store 指令流水的深度受限的问题，实现了软件 caching 的优化。实验表明，对于细粒度通信模式的应用程序，远程 load/store 指令支持和软件 caching 优化能带来平均 10% 的性能加速。第三，针对细粒度的不规则应用，研究结合了指针本地化、探测、通信计算重叠的运行时通信调度方法，很好地隐藏了远程访问的延迟。第四，选择有代表性的带宽受限应用 NPB-FT，实现细粒度通信重叠的优化，得到 48.75% 和 80.34% 的性能加速。与 MPI 的对比表明，PGAS 中高效的单边通信能更好地支持细粒度通信重叠[27]。

针对众核处理器上普遍存在的分布存储体及片上存储受限的情况，该课题组在 2009 年对 UPC 语言扩展了层次并行的特征，目前，该特征已经在 CUDA 集群和 Godson-T（中国科学院计算技术研究所设计的 64 核处理器）两个实验平台上得到了实现[28,29]。对多个规则科学计算应用的测试表明，该扩展能简化加速器平台的并行编程，而通过编译的数组区域分析、软件 DSM 等技术，可以自动实现分离存储和 SPM 存储的自动管理和访问优化，获得与专家编程模式相当的性能。目前该扩展正在曙光 6000 的编译系统中进行实现。

江南计算技术研究所方燕飞等人基于 UPC 源代码开发了面向国产神威系列并行机的 UPC 编译系统[30~32]。他们通过实现多种并行编译优化措施，在大规模分布内存的并行系统上高效实现了 UPC 并行语言，提高 UPC 并行应用效率；针对国产并行机系统结构特点进行了编译优化的算法改进和移植，并根据实际应用课题的特点，提出了包括并行循环迭代划分优化、本地共享访问私有化、动静结合的共享访问消息向量化、共享指针计算优化、全局共享空间优化在内的一系列优化技术，最大限度地实现计算局部化、远程访问向量化，达到高效实现的目的。测试结果表明，优化后的基于国产并行系统的 UPC 编译器性能得到了较大的提升。一些典型并行应用课题的运行效率提高 60% 以上，实现了高效的 UPC 运行时支持系统。根据国产并行机系统的体系结构特点，在并行系统底层消息通信库的基础上实现高带宽与低延迟的 UPC 运行时支持系统；同时实现蝶型同步和基于请求队列的软件分布锁，明显改善 UPC 并行应用在大规模环境下运行的同步互斥性能。按照 UPC-IO 1.0 文本和 UPC 集合操作 1.0 文本实现了并行 IO 和集合操作功能，使得 UPC 功能更丰富，可用性更好。

2.2　大规模并行计算的管理

随着计算机系统和应用程序规模的扩大，如何进行有效的任务调度、错误恢复成为

日益突出的问题，而功耗，往往也成为大型计算中心关心的话题。

2.2.1 调度管理

随着多线程技术的广泛普及，多线程编程语言逐步提供了任务级并行支持，例如，Cilk[33]、OpenMP3.0 和 X10[34] 通常使用 Work-stealing 调度技术帮助负载平衡。现有 work-stealing 技术存在一些问题：1）在一些应用中实现代价比较高；2）需要大量的存储空间进行数据复制来避免潜在的数据竞争；3）对于一些应用程序（有不平衡的调用树，或者没有明确工作集），处理效率低下。Tascell 使用新的基于 backtracking 的调度技术解决上述部分问题，但它又引入了新问题，即对于拥有不平衡的调用树应用程序，不能达到负载平衡。

王蕾等人提出一种新的自适应任务创建策略（AdaptiveTC）来改进现有的 work-stealing 技术[35]。在 AdaptiveTC 中，先创建一定数量的任务使得所有线程忙碌。在执行过程中，除了当空闲的线程需要任务时，每个忙碌的线程避免产生更多的任务放到双向任务队列中。另外，AdaptiveTC 引入一个新的数据属性“taskprivate”，这个新属性主要针对回溯搜索和分支界限搜索中的“解空间”变量。Taskprivate 这个新数据属性不仅方便程序员写任务级并行的多线程程序，而且该属性结合 AdaptiveTC 中新的调度方法，能大量减少解空间的复制，从而提高性能。实验结果显示，对于 16-queen 问题，8 个线程时，AdaptiveTC 比 Cilk 快 171%，比 Tascell 快 72%。

近年来，GPU 在高性能计算领域起到越来越重要的作用，但是如何提高 CPU 与 GPU 组成的异构系统的计算效率仍是编译优化所面临的难题。GPU 在不同负载情况下性能差异明显，发挥多核 CPU 与 GPU 组成的异构系统的总体计算能力需要解决 CPU 与 GPU 之间的负载平衡问题，杨灿群等人提出了一种动态任务调度方法，在程序运行时根据 CPU 和 GPU 在不同负载情况下的实际计算性能动态决定两者之间的任务分割比率，从而实现 CPU 和 GPU 之间的负载平衡。另外，在 CPU 和 GPU 的异构系统中，GPU 需要通过 PCIe 与 CPU 通信以获得输入数据和返回计算结果。这样通信开销将严重制约 GPU 计算性能的发挥，为此，他们提出一种用于 GPU 计算的流水方法，把数据从主存储器传输到 GPU 设备存储器、GPU 上计算核心的执行，以及数据从 GPU 设备存储器传回主存当作流水线的三个阶段，使得多个计算任务能以流水方式执行，达到隐藏输入和输出开销的目的。在天河一号的 Linpack 测试中，采用动态任务调度、GPU 计算流水方法，以及以 GPU 为中心的任务映射、流式数据存取、亲和调度、存储分配优化等技术，使单颗 Intel 多核 CPU 与单颗 AMD GPU 的计算效率由 20% 提升到 70%。相关研究成果发表于 UCHPC-09、ICPADS-09、CIT-09、计算机工程与科学等国际会议和期刊[36~42]。

在并行编程模型与运行时环境方面，陈海波等人提出并设计了分块 MapReduce 模型[43]，实现了原型系统 Ostrich。Ostrich 能根据 CPU 上的核间拓扑结构和缓存体系架构等信息，有效地划分和调度子任务，使得 MapReduce 应用程序能充分利用多核环境中的数据局部性。

2.2.2　容错管理

随着并行计算机系统规模的增加，必然有更为频繁的硬件故障，系统的平均无故障时间远低于许多大规模科学计算程序的运行时间。因此，大规模科学计算程序必须能够容忍硬件错误，但系统级容错由于开销太大往往难以承受，需要研究低开销的应用级容错技术。最近的研究集中在容错模型、容错并行算法设计方法以及编译辅助工具方面。杨学军等人提出了容错并行算法的概念——容错并行算法是一种能够实现快速故障恢复的应用级容错技术，给出了面向 MPI 程序的容错并行算法形式化的设计方法，实现了一个源到源的容错并行算法设计支撑工具 GiFT。GiFT 基于编译指导语句划分计算段，通过面向 MPI 程序的数据流分析技术求解需要保存的变量，并且使用自动并行化方法生成故障恢复代码，从而实现了容错并行算法设计过程的自动化。容错并行算法为提高大规模并行计算机系统的可靠性提供了一种新的方法。相关研究成果发表于 IEEE Transactions on Parallel and Distributed Systems、ICDCS-08、HPCC-08 等国际期刊和国际会议[44~52]。针对这一问题，江南计算技术研究所正在研究如何提炼出适合超大规模计算系统的容错程序模型、如何在并行语言中提供容错描述手段、编译指导的保留恢复等技术。

在程序的动态演化方面，陈海波等人设计并实现了 POLUS 系统[53,54]，支持对程序不同版本进行语义分析以生成动态补丁，并且提供运行时环境来实现对正在运行的应用软件进行动态更新。与同类型系统相比，POLUS 具有如下特点：对二进制文件兼容（不需要对程序进行转换），支持多线程的软件，并且可以使软件从一个错误的状态中恢复，同时具有良好的可用性。

2.2.3　功耗管理

功耗问题是未来高性能技术发展面临的主要瓶颈之一。陈娟、董勇等人研究了并行系统功耗模型和功耗评价、并行任务调度和异构系统的任务调度、处理器及服务器的功耗特征评测等方面的问题，在异构系统的任务调度方面有所突破。异构系统已成为高性能计算机的发展趋势，而传统低功耗优化技术只面向同构系统，难以高效应用于异构系统。陈娟、董勇等人以类 OpenMP 的并行程序为研究对象，研究异构系统并行循环调度和功耗优化的关系。他们首先建立了异构系统功耗感知的并行循环调度问题基本模型。然后通过分析方法给出异构系统并行循环调度的能耗下界，进而将异构系统并行循环调度问题归纳为整数规划问题，并提出了合并同构处理器调度的方法和基于处理器内不同运行级的循环再调度方法。最后，以 CPU-GPU 异构系统为平台，评测了 10 个典型核心计算子程序，实验结果表明该方法可以有效降低系统功耗，提高系统效能。相关研究成果发表于 Software Practice and Experience、HPCC-08、ISPA-08、ICYCS-08、CIT-10、计算机学报、软件学报等国际会议和学报[55~63]。

2.3　程序性能优化

程序的性能优化是编译研究人员和程序开发人员永恒的热门话题，根据应用程序的

特征进行合适的语义保持变换，最大限度地发掘硬件设备计算能力，更快、更好地完成计算，是研究人员一如既往的追求目标。

2.3.1 静态优化

数据布局。如何发掘并行性、如何提高数据访存的局部性是并行编译技术的关键问题。现有编译器的并行性识别能力差、并行化后的数据局部性差，无法在当今存储已经成为性能瓶颈的多核和众核系统上取得很好的性能，开发具有良好数据局部性优化能力的并行编译系统是当前的研究热点。为此，刘雷等人提出了一种将并行性划分与数据局部性优化融合的循环变换编译框架，利用数学模型对循环变换进行抽象和分析，通过以tiling 变换划分的“块”为单位来进行并行化分析，不仅可以降低并行划分算法的时间复杂度，而且也能提高并行粒度，增加数据局部性，并降低处理器的通信开销[65]。利用限定形状的“非矩形”tiling 变换，可以对诸如 stencil 等科学计算类应用程序，划分出同步开销最优的粗粒度并行性，而且其代码生成的时间复杂度也和“矩形”形状 tiling 的一样简单[64]。此外，该研究团队还提出了一种新的循环最优变换序列的求解算法，称为循环自适应变换技术[66]，该方法利用辅助的使能循环变换，自动修正一个循环变换后引入的违法依赖关系，使得编译器可快速准确地分析出性能优化的循环变换序列。同时，为了解决传统编译时自动并行化技术并行性分析能力差的问题，给出了一种动、静态结合的并行化方法。该方法可以提高编译器的私有化分析能力，并可以有效地解决别名给并行化带来的问题[66]。和纯粹的动态并行化方法（如投机并行化）相比，该方法几乎没有运行时开销，而且利用编译器的程序变换技术可以发现更多的并行性。

2.3.2 动态优化

动态优化中非对齐访存的优化。现在二进制翻译技术已成为软件移植的重要手段，在一些体系结构中（例如 ×86）允许访存地址非对齐的内存访问，但是大多数 RISC 机器结构的机器上要求访存地址必须是对齐的。因此在二进制翻译器中，如果源体系结构支持非对齐访存，但目标体系结构不支持非对齐访存，那么我们在翻译访存指令时必须特别处理。一般的翻译方法将导致移植程序在运行时产生大量的非对齐访存异常，从而影响程序性能。通过对 ×86 体系结构上可执行程序中非对齐访存的统计，并对已有的在二进制翻译器中处理非对齐访存的方法进行了深入的分析。在已有方法的基础上李建军等人提出了一种利用异常处理机制处理非对齐访存的方法，它可以实时地修改程序执行过程中发生非对齐内存访问的指令，有效地减少因非对齐内存访问带来的时间开销。在基于 SPEC[67] CPU2000 和 CPU2006 的性能评价中，该方法比其他方法有 14% ~73% 的性能提升[69]。

动静结合的二进制编译方法。在二进制翻译方面，徐超豪等人设计并实现了动静结合的二进制编译方法[68]，编译器在静态编译过程中收集内存访问和寄存器分配信息，并形成注解，在动态二进制翻译过程中通过结合这些注解信息来提高二进制翻译的效率。

动态内存池优化。在 C 和C++ 等程序中，经常会出现通过对库函数 malloc、calloc 和

realloc 的调用，从堆上为程序动态分配内存。这些函数都属于系统库函数，通过依据程序中相应函数的执行顺序和所需空间大小，来选择合适大小的内存空间，分配给程序。该分配策略不考虑程序对内存的使用情况，关系紧密的动态数据对象可能会被分配在不相干的位置，影响数据的局部性。而通过截获堆空间分配函数，将关系紧密的数据分配在一起，叫做“池分配”。池分配可以改善数据的局部性。已有的动态方法是基于分配点的，即每个动态内存分配点有一个专门的池，这个分配点的所有分配请求都在这个池里得到满足。

通过对程序特征的分析，王振江等人提出了一种基于调用链的池分配策略。当一个内存分配点在成功得到内存对象后，不进行有意义的操作，而只是简单地把指针传递给上层调用点，那么这个内存分配点就没有专门的池，而是需要和它的上层调用点一起形成一个调用链。不同的调用链才有不同的内存池。另外，池中对象有紧密关系的两个池可以合并。对 SPEC2000 和 SPEC2006 的一些程序的测试表明，该方法能够带来平均 10% 以上，最高达 82% 的加速比[70]。

JVM 和即时编译。史晓华等人围绕 Java 程序的运行环境开展工作，提出了一种为即时编译器和时空受限系统设计的轻量级线性复杂指令调度算法[71]。该算法进行指令调度时，不基于传统的 DAG 图或表达式树，而是基于一种独创的数据结构扩展关联矩阵，其时间复杂性在最坏情况下也能与全部指令长度构成严格的线性关系，仅占用不到 1 KB 的内存空间。该算法已被 Intel 为 Xscale 设计的高性能 J2ME 虚拟机 XORP 采用为即时编辑器中的缺省指令调度算法。他们还提出一种可以有效地处理开放世界所面临问题的逃逸分析架构[72]。逃逸分析（Escape Analysis）是一种可以有效减少 Java 程序中同步负载和内存堆分配压力的跨函数全局数据流分析算法。此前绝大多数逃逸分析的实现都基于一个所谓“封闭世界”（Closed World）的前提：所有可能被执行的方法在做逃逸分析前都已经得知，并且程序的实际运行不会改变它们之间的调用关系，真实的 Java 程序拥有的许多特性，例如动态类加载、调用本地函数以及反射程序调用等，都将打破所谓“封闭世界”的约定，这样的真实运行环境被称为“开放世界”。该逃逸分析架构可以有效地处理上述开放世界所面临的问题，有效地降低程序运行负载，该架构已经在 Intel 的开放式 Java 虚拟机系统 ORP 中实现，实际测试表明，可以有效地消除 SPECjbb2000 和 SPECjvm98 的 db 等基准测试中 70% ~94% 的同步操作，大幅度地提高 15% ~31% 的程序运行速度。

2.3.3　迭代优化

迭代编译优化已经成为一种重要的优化策略，因为该方法应用相对简单，又能有效提高程序的性能。该方法本质上是基于对编译优化空间的搜索，通过多次实验运行，测量程序的运行时间等指标来评价不同优化的效果，直到找到（近似）最优的优化为止[73]。但是该方法所依据的一个前提却从来没有被真正证明过：那就是可以找到在不同数据集上都有效的最优编译优化组合。到现在为止，大多数迭代编译优化的研究是在同样的数据集上重复执行多次，从而获得最佳的优化以及相应的加速比。

一些研究尝试在多个数据集上使用迭代编译优化，但是这些研究通常使用少于真实情况中收敛所需要的迭代次数（数十至数百次）和数据集。陈洋等人在迭代编译方面的主要贡献是首次在多达 1000 个数据集上对迭代编译技术进行了评价，第一次为 MiBench 基准测试程序集（包含 32 个程序）中的每个程序收集/构造了 1000 个数据集，并首次使用该数据集评价迭代编译优化。收集过程中只选择有自由版权的数据来源，以便公开发布收集到的数据集，与其他研究人员共享工作成果。研究发现，在不同的数据集上得到一个鲁棒性好的迭代编译优化策略是可能的：对所有的 32 个程序，至少存在一个编译优化组合，使得可以在每一个数据集上获得最优加速比的 86% 或以上（使用 ICC，若使用 GCC 为 83%），而且该选项组合是程序相关的。这一结果在两个最广泛应用的编译平台上得到确认：Intel ICC 和 GNU GCC，针对两个编译平台的最高级别的优化（分别是-fast 和-O3）。在 ICC 上有最高达 1.71 的加速比，在 GCC 上有最高达 2.33 的加速比。这个发现意味着在多数据集上进行程序优化比以前人们所认为的要简单得多，这为迭代编译优化方法的实用性和可靠性铺平了道路[74]。基于此还提出了针对软件发布前和发布后的优化策略。

2.3.4 基于反馈的程序优化

在优化过程中，很多情况下需要运行时的信息才能作出最优决策。而这些运行时信息在静态编译过程中无法得到。动态编译虽然能提供非常准确的运行时信息，但是由于动态优化及代码生成具有大量的开销，而优化带来的好处可能无法弥补这些开销。因此，基于反馈的编译技术得以推广。该项技术的特点是，用户需要进行两次编译。第一次编译产生的可执行文件能采集程序运行时的一些信息；程序员使用训练数据集对程序进行训练，得到程序执行过程的一些行为剖析（Profile）；在第二次编译时，运用这些剖析信息对程序进行相应的静态优化。

漆锋滨等人在研究 Open64 编译器反馈式编译优化技术的基础上，针对 Alpha 结构的特点，实现和扩展了反馈式编译优化在寄存器分配中的应用[76]，并利用反馈编译技术提高了 Alpha 中的转移预测命中率[77]，获得了较好的优化性能。还利用反馈式数据预取优化技术解决链式结构的预取问题，对针对链式结构的反馈式数据预取进行了优化，SPEC2000 测试表明，平均性能提高了 4.1%[75]。

传统领域内基于反馈的优化需要对程序进行插桩，从而在程序的训练过程中能记录下程序的每一次行为。这个方法存在以下几个问题：1）开销大，通常开销可以达到 20% 到 10 倍不等；2）需要自制训练数据集；3）插桩后的程序行为可能改变。因此，虽然基于反馈的程序优化能带来很好的加速比（15% 左右），但是它仍没有在工业界被广泛应用。

为了解决以上几个问题，陈德颢等人提出了一种采用基于程序硬件计数器的采样技术来获取程序行为剖析的编译技术[78]。其核心思想是，在硬件计数器溢出时产生系统中断，在中断程序中记录下当前程序指针的地址，从而实现对程序行为的采样分析，得到每条指令的执行频度。之后，将采样分析的结果与源代码建立映射，从而使程序行为剖

析能够为编译器所使用。然而，基于采样得到的指令执行频度并不精确。通过微型程序研究，发现导致不精确的一些可能原因，并提出通过采样多个不同事件来提高采样的精度。通过以上工作，基于反馈的程序优化能够达到 80% 以上的峰值加速比，很好地解决了该方法可用性的问题。

与其他需要程序运行时反馈信息指导优化的编译技术一样，结构体数据布局优化也面临着获得反馈代价高、构建训练测试环境困难以及程序执行路径变更导致程序反馈不具代表性等问题。Levin 等人提出的 PMU 反馈修正算法[79]由于面向通用编译优化技术，在修正 PMU 反馈过程中需要保持“回路一致性”。“回路一致性”对于对流信息敏感的编译优化技术（如分支预测等）具有重要意义。然而结构体数据布局优化仅需要基本块执行频率反馈，强制保持 PMU 反馈的回路一致性反而有害于指导结构体数据布局优化。继摆脱回路一致性的限制之后，闫家年等人的研究工作[80]在尝试利用 PMU 指导结构体数据布局优化的过程中还发现：合理选择 PMU 反馈类型并有针对性地对 PMU 反馈进行加权处理，虽然会使 PMU 反馈偏离真实插桩获得的反馈，但是在指导结构体数据布局优化时却取得进一步的性能改善。实践经验揭示，根据不同编译优化技术的特点，寻找有针对性的 PMU 反馈修正算法，有利于利用 PMU 反馈指导优化。在两个平台上的实验显示，使用修正的 PMU 反馈指导数据布局优化能够达到使用程序插桩获得程序反馈效果的 97.6% 和 99.4%。使用 PMU 反馈的时间开销只有原程序执行时间的 12.3%，而程序插桩方法的开销达原程序执行时间的 341.5%。该方法大大提高了结构体数据布局优化的执行效率。

2.4　检测与分析

高效的流敏感上下文敏感指针分析。近年来，随着程序检错领域的研究广泛开展，对程序分析的精度要求越来越高。然而高精度的程序分析必然带来分析效率的降低，以致不可忍受。C 语言含有大量的指针使用，指针分析是很多程序分析的基础。现有流不敏感上下文不敏感的指针分析精度不够，不能满足精确程序检错的需要。而现有流敏感上下文敏感的指针分析精度虽高，但是分析效率较低，不能处理超过百万行以上的程序。于洪涛等人提出一种高效的流敏感上下文敏感指针分析，该指针分析按指针的指向层次从高到低逐层分析各层指针，并利用程序的 SSA 形式提高流敏感分析效率，利用 BDD 紧凑地表示上下文信息，提高上下文敏感分析效率。为了保证算法的正确性，该指针分析预先利用一种简单快速的 Steensgaard 风格的指针分析来计算指针的指向层次。与已有最新的流敏感上下文敏感指针分析相比，该算法在分析效率上有非常大的提高，并能分析超过 100 万行的 C 程序[81]。

动态检测违反顺序一致性错误。随着多处理机（如片上多核）的普及，并行程序的编写也将更加普遍。与消息传递并行程序（如 MPI）相比，共享存储的并发程序（也叫多线程程序，如 pthread）支持传统的单地址编程空间，更容易编写，所以更受程序员的欢迎。编写并发程序时，为了获得高性能，程序员会故意引入数据竞争（Data Race），

例如大部分 lock-free 算法。然而，此类程序在弱一致性的模型中执行时，由于编译器的指令调度、处理器的乱序执行等，可能得到非顺序一致性的结果，比如 Double-Checked-Locking 算法。为了使得这类程序正确执行，需要在程序中适当地插入 fence 指令，来保证前后指令的执行顺序，从而使结果是顺序一致的。

段月路等人提出一种动态检测方法[82]，基于现有的数据竞争检测工具，以很小的额外开销（平均 3.3%）找出潜在的违反顺序一致性的错误（如 MySQL、Apache 以及 Cilk 并发库等应用中的错误），并通过向源程序插入 fence 指令消除这种隐患，最大限度地保证程序得到顺序一致性的结果，同时使得性能损失减到很少（SPLASH-2 平均下降 6.1%）。与编译器静态分析技术相比，该工具更为精确，插入的 fence 对程序性能影响很小；与验证技术相比，该方法可以处理如 MySQL、Apache 等更大规模的实际应用。

卢锡城等人面向高可信软件提出了一种二进制级高危整数溢出错误的动态全自动检测方法（Dynamic Automatic Integer-overflow Detection and Testing，DAIDT）[83]。该方法无需任何源码甚至是符号表支持即可对二进制应用程序进行全面测试，并自动发现高危整数溢出错误，在理论上形式化证明了该技术对高危整数溢出错误测试与发掘的无漏报性、零误报性与错误可重现特性。利用原型系统 IntHunter 对 3 个最新版本的高可信应用程序（微软公司 Windows 2003 和 2000 Server 的 WINS 服务、百度公司的即时通信软件 BaiDu Hi）分别进行了 24 小时测试，共发现了 4 个高危整数溢出错误，其中 3 个错误可导致任意代码执行。

2.5 形式化验证和可信编译

2003 年 Tony Hoare 的论文“验证式编译器：计算研究的伟大挑战”[84]，是近年来人们关注验证技术的一个缩影。2009 年 ACM 通讯关于编译器研究的讨论中认定，软件自动化验证将是未来几十年极为重要的挑战之一[85]。随着国家和社会对软件系统依赖程度的日益增长，安全关键软件系统是否具备正确、安全和可靠等性质至关重要。

经典的程序验证方法大都建立在抽象语言、代数演算或结构、状态机以及软件规范等基础之上。这些方法大都具有较高的抽象层次，如何将其应用至实际的（源语言和目标语言）代码级并发编程模型中去，则需要许多额外的工作。此外，即便是在较高抽象层次的语言和规范之上经过可信性验证的程序，若要确保将其映射（编译）到机器语言后仍然保持可信，还必须保证每一层的变换过程都要可信，而且还需要考虑到和底层低级语言之间的接口验证。底层代码验证和可信编译技术是解决这一问题的两个主要途径。

底层代码和中间代码验证。王生原等人提出一种基于有色网的可验证低级并发编程模型[86,87]，尝试将经典的并发系统形式规范模型直接作为并发代码的一部分，从而改善并发编程的可信任度；同时，相关的工作也已经扩展至基于事务内存的事务型并发编程模型。朱允敏等人给出一种基于 Spin 锁的多核并发程序验证逻辑框架[88]。

从高级语言到汇编的编译过程中会大量使用各种不同的中间表示语言，而虚拟机解释执行通常也采用某种中间语言，最为通用的是以 Java Bytecode 为代表[89]的字节码语

言。最初针对 Java 字节码验证的主要研究工作集中于 JVM 内部检查器，其目标是代码的类型安全性检查，从而保证存储安全性。随着研究工作的深入，近年人们开始关注超越类型的诸如部分正确性等程序特征。董渊等人综合考虑虚拟机的特性，提出一种程序验证逻辑框架，为字节码代码模块化验证问题提供一个良好的解决方案[90,91]，同时大幅度提升逻辑系统的表达能力。关键点包括：1）借鉴汇编程序的验证方法，设计出一种用于字节码程序验证的逻辑系统，完成其合理性证明和一组代表性实例程序的模块化证明，并实现机器自动检查；2）采用形式化方法，证明机器代码实现的虚拟机程序，以解决虚拟机的可信问题；3）由于字节码的平台无关性，可以实现“一次验证，处处可信”，即对于开发人员来讲，只需要在字节码层面证明一次，就可以在不同的已验证虚拟机上正确运行。同时，这部分工作为底层机器语言和中间语言的多语言编程建模和验证提供了一个良好的研究基础。

出具证明的编译技术。出具证明的编译（certifying compilation，验证式编译）是提高软件可信程度的一种重要方法。从概念上说，出具证明的编译器利用自动定理证明技术和工具，在源代码级证明程序具备所期望的性质，在目标代码级证明所生成的目标程序保持源程序的语义，并且把这两方面证明都附加在目标程序上，使得目标程序的使用者可以用工具来检查该程序是否具备所期望的性质。

国内近几年在出具证明的编译技术方面研究取得如下一些主要进展。陈意云等人设计了便于指针程序验证的程序逻辑，称为指针逻辑，并把它用到出具证明编译器的原型中[92~94]，最近又把指针逻辑重新设计为更直观且表达力更强的形状图逻辑。王志芳等人设计了能输出证明的线性整数自动定理证明器和指针逻辑自动定理证明器。输出证明是为出具证明编译器设计的自动定理证明器和普通自动定理证明器的一个重要区别[95~97]。基于前两点的成果，开发了依赖于程序员提供函数前后条件和循环不变式的出具证明编译器的原型[98~100]，并在进一步研究代码优化对出具证明的编译方法的影响。

多核、多处理器等以共享存储为特征的系统结构的迅猛发展及其对并行的全面支持，刺激着对高产能（Productivity）并行编程语言的需求，给并行编程语言的设计、实现和验证等方面带来许多新的挑战。近几年研究主要集中在共享资源的各种同步控制机制的实现技术和形式验证，以及基于共享资源使用声明的高产能并行编程语言的设计和实现。张昱等人研究并设计了两种不含事务嵌套的事务内存（Transactional Memory）实现技术及其正确性验证方法[101,102]，付明等人研究并设计了基于不可重入读写锁或基于可重入互斥锁的并行程序的安全性验证方法[103,104]。以 Java SE 6 中的并发库为基础，设计并实现了融合事务内存与锁方式的程序库，作为对高级并行编程语言中原子区（Atomic Section）构造的一种实现方式[105]。研究在并行编程语言中，用共享资源及其使用特征的声明来取代对共享资源访问控制操作的直接编程，以避免锁方式的编程复杂性。它也用来取代原子区方式，以克服原子区方式的实现困难性。

面向 Java 平台的携带模型代码方法。携带模型代码（Model Carrying Code）是针对移动代码安全问题的一种综合解决方法[106]，该方法同时从移动代码生产者和使用者角度出发来保证安全执行非信任应用程序。结合静态程序分析、抽象解释和运行监控，金

英等人提出顺序和多线程 Java 程序安全相关行为模型的自动生成方法[107,108]，首次给出采用扩展的参数化上下文无关文法来表示安全相关行为模型，具有良好的扩展性和复合性；提出安全相关行为模型的静态检查方法，将安全相关行为模型与基于有限自动机的安全策略定义相比较，判定安全相关行为模型是否满足安全策略[109]。基于这些方法设计并实现了 MCC 方法在 Java 平台上的运行环境，完成携带模型代码 MCC 方法在 Java 平台上的实现，在保障 Java 移动代码安全方面做了有益的尝试，主要工作包括：1）安全相关行为模型自动生成程序，该过程可以作为编译程序的一部分，将生成的安全相关行为模型作为扩展部分写入 .class 文件中；2）安全相关行为模型的检查器[110]；3）安全策略监控程序，通过修改 Java 虚拟机增加安全策略强制实施功能[109]。

2.6 量子程序设计语言及其处理系统

量子程序设计语言自 1996 年出现以来，颇受业界重视，2008 年，在 PLDI 会议 Alfred Aho 教授的特邀报告“Quantum Computer Compilers”（量子计算机的编译器）中，专门介绍了这一领域的重要问题和进展。在分析几种典型量子程序语言，着重阐明量子计算、语言风范、程序结构、输入输出、异常机制等特性的基础上，徐家福等人自行设计提出量子程序设计语言 NDQJava[111,112]，并给出了包括设计准则、语言风范、硬件平台、基本成分以及示例等。同时给出量子程序设计语言 NDQJava 的一个处理系统[113]，其特点是：程序中经典部分之处理借助 Java 系统，着重考虑量子部分之处理。该处理系统遵循编译－解释的途径，由词法分析程序、语法分析与代码转换程序以及量子汇编与解释程序三部分组成，并给出了系统在经典计算机上的模拟实现。

3 产品编译系统进展

编译器是信息产业链中位于处理器和应用程序之间的关键环节，对于整个产业的可持续发展具有重要的战略意义。任何高级语言程序都需要经过编译处理才能够运行，而任何处理器性能的充分发挥都必须依赖优化编译的支持，伴随着国产高性能通用处理器的发展，近年来，我国在编译领域取得了重要进展，为国产处理器的推广、普及提供了最有力的支持。

3.1 支持国产处理器的高质量编译系统

3.1.1 龙芯编译系统

龙芯编译系统是为国产高性能通用处理器芯片——龙芯开发的具有高性能、高可靠性的编译系统及工具链，包括：预编译器、编译器、汇编器、链接器及运行时信息收集、

自动调优等相关工具。龙芯编译系统由中国科学院计算技术研究所系统结构重点实验室编译组开发。该编译系统从2002年开始开发，先后为龙芯1号、2号、3号处理器系列服务，为龙芯处理器性能提高和市场推广起到了重要作用。

龙芯编译系统基于国际知名高性能开源编译器Open64[114]，支持C、C++、Fortran等多种语言。龙芯编译系统继承了Open64的多种特性，如健壮性好、性能好、代码风格较好、有固定社团支持等优点，同时，由于提供了大量龙芯处理器特征相关的优化和支持，龙芯编译器在龙芯平台上的性能明显好于GCC编译器[115]。

龙芯编译系统的高性能、高可靠、功能强大等特点，可以具体地从如下几个方面体现：

1）龙芯编译器全方位地包括了全局标量优化、循环优化、过程间分析和优化、机器相关等众多经典传统优化，能大幅度地提高目标代码的性能，简单介绍如下：

- **全局标量优化**。基于先进的静态单赋值（Single Static Assignment）和全局值计数（Global Value Numbering）技术，在函数范围内，实现了部分冗余删除、无用代码删除、常数传播、复写传播等若干经典优化。
- **循环优化**。采用先进的数据分析模型，能对应用程序进行全局分析，得出应用程序的数据分布、计算分布等特征，并根据目标机特点，制定适合于该应用程序的循环优化方案和数据分布方案，使多种优化能有机结合，更大程度地提高应用程序性能。具体地说，这部分提供的优化有数组压缩、转置、展平、对齐等数据布局优化和循环合并、交换、翻转、分块等循环优化。
- **过程间分析和优化**。龙芯编译器提供实用灵活的过程间分析和优化框架，并在此基础上提供全局常数传播、全局无用代码和无用函数删除、过程间别名和函数副作用分析、函数内联、结构体重排等分析和优化。
- **机器相关优化**。这部分包含的是和编译目标平台紧密相关的优化，如寄存器分配、指令调度和机器相关的控制流优化、窥孔优化。

2）龙芯编译器的中间表示共分为5个层次，其中越高层的中间表示越接近高级语言，越低层的中间表示越接近目标机器的汇编代码，各个相邻的层次之间有专门的转换模块将高层的中间表示向较低的层次转换。丰富的中间表示层次使得龙芯编译器中的所有优化都能在其合适的层次上发挥最大的作用，用户也可以很容易地找到合适的中间表示层次添加新的模块。

3）龙芯编译系统中提供了若干相关工具，极大地丰富了龙芯编译系统的功能。具体地讲，龙芯编译系统中提供了高、低两个层次上的运行时信息收集工具（Profiling）、中间表示向多种源语言的转换工具（C、Fortran）、中间表示的图形化显示工具、优化自动调优工具等多种高效实用的工具。

4）提供若干针对龙芯处理器特征的机器相关优化，有效地提高了龙芯系统的性能。

现在，Open64编译器的龙芯后端已经发布并合并进入Open64主分支，代码可以在该软件的开源网站免费下载。目前，龙芯编译器的研发团队仍然在继续研发下一代龙芯处理器的编译系统。

3.1.2 银河编译系统

银河编译系统 YH-CS（YH-Compiler Suite）是面向国防科技大学自主高性能多核飞腾处理器和银河高性能系列并行计算机的编译器及工具链。YH-CS 语言支持丰富、平台支持广泛，具有强大的编译优化能力，提供配套调试和性能分析工具，能高效支持高精度浮点计算，2008 年获部委级科技进步一等奖。其主要特点有：

1）兼容性高。支持 C、C++ 、Fortran 和 Java 等语言，支持 OpenMP API3.0，支持 MPI2.0 接口；除支持飞腾处理器外，支持主流通用 CPU，包括 Itanium、Xeon 和 SPARC 等。

2）采用了先进的优化技术。支持跨文件的过程间优化，包括过程内联、过程（部分）克隆、过程间常数传播等；利用飞腾处理器硬件性能计数器的剖视指导的优化；通过自动向量化、固有函数（Intrinsic）等方式支持国产飞腾 CPU 的 SIMD 指令；针对飞腾 SIMD 指令的寄存器分配和指令调度；通过亲和调度、数据 padding，提高多线程程序性能；利用投机并行，支持非规则程序的自动并行化。

3）支持高精度浮点计算。支持 80 位扩展双精度浮点计算，提供 80 位浮点数据类型和固有函数。80 位扩展双精度浮点计算性能高，在 Itanium 系统上，性能是 Intel 编译器 64 位浮点计算的 71%；在 Xeon 系统上，性能是 Intel 编译器 64 位浮点计算的 51%。

4）支持低功耗优化。利用 OpenMP 语言机制进行低功耗优化，降低多线程并行程序功耗；支持利用 DVS 技术和 Profile 技术进行串行/并行程序的能量和性能权衡，满足性能需求，降低能耗。

5）调试和性能分析支持。支持 OpenMP 与 MPI 程序的性能分析，支持动态二进制代码插桩，支持基于剖视和事件跟踪的性能监控，提供可视化数据分析方法；支持灵活、低开销的硬件性能计数器（PMU）性能监控，适用于飞腾、Itanium、×86/×86_64 等主流通用 CPU；提供命令行和 GUI 的调试工具，支持多线程、多任务并行程序的断点调试。

3.1.3 神威睿智编译系统

国产神威睿智 CPU 采用了一套全新的体系结构以及指令系统，需要为之开发相适应的编译系统。由于 GCC 本身的开源以及可移植的特性，进行针对国产 CPU 的 GCC 移植成了比较方便和现实的途径，基于 GCC 开发 SWGCC 编译器在功能上可以满足国产 CPU 平台上编译整套 Linux 发行版软件的需要。另外，为了进一步提高 SWGCC 编译目标码的性能，江南计算技术研究所还对 GCC 的编译优化进行了许多改进，包括软件流水、数据预取等，并根据国产 CPU 的特点，在 SWGCC 中增加了向量指令的后端优化，性能在原有基础上得到了较大幅度的提升。

为了给神威睿智国产 CPU 系统提供一个性能更高的编译器，该所还以 Open64 为基础开发了另一套 SWCC 编译器。在 SWCC 编译器中，针对国产 CPU 的指令集结构扩展了一套 SIMD 内部函数接口，同时也在 Open64 原有优化措施的基础上进行重点补充。Open64 的编译优化简单可以分为两种：一种是与机器无关的优化，比如常数传播、不变

量删除、循环外提等；另一种就是与机器结构紧密相关的优化，比如寄存器分配、指令调度、反馈优化等。与机器结构无关的优化具有一定的通用性和普适性，重点还把这些优化应用到扩展的 SIMD 中间表示类型上，充分利用已有的优化手段。与机器结构相关的优化是 Open64 移植的重点，针对国产 CPU 结构的特点编译优化的算法改进和移植，并根据实际应用程序的特点，采用了一些新的优化技术，包括转移预测的反馈优化技术、基于缓冲寄存器的需求预估和胖点计算的寄存器分配技术、多维指令空间优化技术、数据预取技术、基于过程间分析的寄存器分配优化和对界优化技术、基于循环级和函数级的迭代编译技术等，取得了很好的优化效果。

3.2　服务于应用兼容的二进制翻译系统

二进制翻译的是通过软件手段将一种指令集体系结构（ISA）上的可执行代码翻译到另一种 ISA 上运行。对于国产处理器来讲，二进制翻译的一个重要的现实意义是可以帮助它们摆脱丰富应用软件。由于 Intel 公司专利的限制，国产微处理器大多都采用与 ×86 不同的指令集体系结构。×86 系列机器作为当前的主流机型，占有绝对的市场份额，拥有极其丰富的应用软件资源。与之不兼容则会失去相应软件的支持，在市场竞争中处于劣势。通过二进制翻译技术，可以将 ×86 处理器上的软件迁移到国产处理器上，提升国产处理器的竞争力。随着国产 CPU 的发展，二进制翻译还可以解决新型处理器向下兼容问题，减轻硬件兼容设计负担，进而达到简化硬件设计的目的。

3.2.1　龙芯相关的二进制翻译系统

中国科学院计算技术研究所的研究人员针对“龙芯”处理器，设计了一个从 ×86 到“龙芯”的进程级二进制翻译系统——DigitalBridge，以移植 ×86 应用程序到龙芯上，丰富龙芯的应用软件。

性能是决定一个二进制翻译系统能否真正得到应用的关键因素。因此，性能提升技术的研究非常重要。经过研究人员的努力，Digital Bridge 系统的性能获得了较大的提升。经 SPEC CPU 2000 的 ×86 二进制码进行测试，几何平均性能分别为 MIPS 本地编译器（Gcc4.3.1-O3）所产生的代码的 80.2%。并且在龙芯机器上，已经能够成功实现 Firefox、Acrobat Reader、Flashplayer、Apache 和 Mysql 等实际应用程序的翻译执行。

最近的两年里，着重探索了如下几个方面的技术：

动静结合的翻译运行策略。设置一个静态翻译模块，该模块将静态时刻所能够分析清楚的代码进行预先翻译。这样可以减少动态时刻的翻译开销。另外，由于静态翻译不占用运行时间，可以进行较为深入的代码优化，如窥孔优化、寄存器分配等，以改善本地码的质量。

标志位模式识别。在 ×86 机器中，几乎所有的运算指令都会影响标志位，如果全部模拟标志位，显然会带来极大的开销。为了减少标志位，需要对基本块内的标志位的定值引用关系进行分析，消除不必要的计算。同时，还需要进行模式识别，利用龙芯本地

类似功能的指令组来替换×86 指令组，进一步改善标志位相关代码的效率。

非对齐访存处理。×86 的硬件支持非对齐访存操作，当程序中发生非对齐访存时，基本上不会对程序的性能造成影响。而龙芯机器采用的是 MIPS 架构，硬件上不支持非对齐的访存，每次遇到非对齐操作时，会陷入系统态，由操作系统来处理，这样就带来很大的开销。设计了一种基于剖面分析和异常处理相结合的策略，尽量避免误判的情况下，发掘出所有可能造成非对齐访存操作的代码，而后采用一个不会引起非对齐异常的访存操作序列来替代非对齐操作。

基于内存池的动态优化。部分程序中会大量地申请堆上数据，并对这些数据进行操作。系统所提供的堆数据分配器，主要关注的是对堆空间的充分利用，并且避免碎片的产生。无论是程序员还是堆分配器，很少会关注这部分数据的局部性。为了改善堆数据的局部性，定义两种数据亲和关系。一些节点具有相同类型，并且用于形成同一数据结构（如链表、树、图等），这些节点间具有第 I 类亲和关系。具有第 I 类亲和关系的节点，如果它们的某一个域是指针类型，则称同一个域所指向的所有节点具有第 II 类亲和关系。当出现数据分配操作时，可以将具有相同亲和关系的数据分配在同一个内存池中，这样起到提升数据局部性，降低 cache 和 TLB 缺失率的作用。

栈变量的提升。×86 寄存器的数量比较少，仅仅有 8 个通用寄存器，而龙芯则提供了 32 个定点和 32 个浮点寄存器，优化中利用龙芯上多出来的寄存器，来提升×86 的栈上变量，将访存操作变换成寄存器操作，改善程序的性能。

3.2.2 神威睿智相关的二进制翻译系统

江南计算技术研究所针对国产 CPU 研制二进制翻译系统，主要开展了如下研究工作：

对二进制翻译框架、精确异常、代码自修改、间接跳转、轻量级运行时优化等关键技术进行了研究；确定了动静结合的二进制翻译方法，综合利用动态翻译、静态翻译的优点，在运行时收集信息的基础上离线进行深度优化，这样既降低了动态翻译开销又提高了代码的翻译质量；针对兼容源系统、目标系统结构特征的优化研究，如大小端镜像置换、特殊寄存器映射等；针对二进制翻译调试困难，研究并开发了一些辅助工具。

二进制翻译系统已经完成了第一版的研制，可在国产 CPU 上高效翻译运行在 PowerPC、×86 上生成的串行应用程序；可在国产主机系统上翻译运行在 PowerPC 上生成的标准 MPI 并行程序；系统经过上千个各类应用测试，完备性好；系统经过深入的性能调优，翻译运行时间与在国产 CPU 上直接运行相比，比值小于 2.3，性能优良。

3.3 编译系统测试验证工具

编译器本身是一个复杂的软件，包含了大量的优化转换，对其进行全面的测试和验证是一项复杂而又具有挑战性的任务。C/C++ 语言广泛应用于各个领域，从嵌入式系统到大型分布式数据库应用。C/C++ 编译器的质量和鲁棒性对应用程序的正确性和安全性

至关重要。国防科技大学研发的银河编译器验证工具 YH-CVT（YH-Compiler Validation Tool）是 C/C++ 编译器进行自动测试验证的工具，能够对编译的整个过程进行验证。YH-CVT 由测试用例集和测试工具组成。YH-CVT 已经成功运用于航空航天等安全关键领域的编译器测试验证。YH-CVT 的主要特点有：

1）测试用例覆盖语言标准 ISO 9899：1990 C、ISO 9899：1999 C99 和 ISO 14882 C++，以及 GNU C 扩展语法和 GNU C++ 扩展语法，包含正例和反例测试，测试用例 6.5 万多个。

2）对规范中涉及的未指定行为和未定义行为也给出测试用例，方便用户进一步了解目标编译器的特性。

3）包含表达式测试用例自动生成工具，能够对表达式进行深度测试，支持所有运算符和数学库函数，覆盖定点、浮点、单精度、双精度及其混合运算。

4）采用了模块化结构，具有良好的可扩展性，用户可基于 YH-CVT 开发自己的测试用例。

5）支持交叉编译器测试，支持编译器优化选项组合的自动测试。

6）支持类 UNIX 操作系统和 Windows 操作系统。

3.4 嵌入式编译系统和其他系统

嵌入式系统正在成为我国整个信息领域最活跃的一部分。以手机、汽车电子和智能家电为代表的嵌入式系统已经越来越深入到每个家庭和个人，嵌入式系统已经成为人们生活中不可或缺的部分。

针对嵌入式处理器的编译器。在嵌入式领域，常常需要为特定应用程序定制具有特殊指令的处理器以满足其性能需求。这时，具备良好重构（重定向）能力的编译系统是必不可少的支持工具。随着我国集成电路设计领域的不断进步，嵌入式处理器设计能力不断提高，随之而来对可重定向编译器的需求也日趋强烈。具备良好重定向能力的编译系统是嵌入式领域重要的支持工具。针对这一问题，张铎等人基于开源高性能编译器 Open64，以 PowerPC 嵌入式处理器为例，开展重定向关键问题研究和代码实现，自主开发完成一款具有工业产品水准的高性能开源编译器后端[116,117]。实测结果表明在正确性和性能方面，所完成的编译器均接近或达到和 GCC 编译器相当的水平。代码已通过严格测试，并成功合并进入 Open64 主分支，作为重要的新特征之一随 Open64 4.2.3 发布。这项工作为进一步研究和应用提供了良好的编译工具支持以及实现参考，在此工作基础上，该研究团队正在进行后续的编译自动重定向研究工作。

在面向嵌入式编译技术方面，张为华等人设计并实现了针对流处理器和网络处理器存储系统的编译优化算法[118,119]，通过利用流处理器的互联特性和网络处理器的软件 Cache 管理功能，提升 Cache 的访问性能，降低共享总线的竞争。

通用标准测试语言 ATLAS 及其实现技术。ATLAS 是面向自动测试领域的通用测试语言，可以看做是设备和平台无关的自动测试系统规格说明语言[120]，其语法比较接近自

然语言，语言比较庞大，广泛应用于航空航天、工业等领域自动测试系统的研制。在对ATLAS 语言深入研究的基础上，郭德贵等人定义了 ATLAS 语言的指称语义[121]，为理解和实现该语言奠定了基础；给出了 ATLAS 语言的程序分析方法，将程序切片和部分求值等技术应用于 ATLAS 程序，从而实现对 ATLAS 源程序的优化和异常检测；给出了 ATLAS 语言到 C 语言的转换规则，并设计实现了 ATLAS 编译系统[122]，包括：

1）ATLAS 语言的编译器。它含有词法分析系统、语法分析系统、语义分析系统和 C 代码转换器。

2）测试环境描述语言。利用该语言可以完整描述测试环境中的资源配置及资源、适配器和 UUT 间的连接关系。

3）测试程序的运行系统。针对测试系统设备分配问题，提出了基于有向图的启发式双重回溯静态设备分配算法，在一定程度上提高了 ATLAS 测试程序的运行效率[123]。

4 开源软件平台和教育

4.1 开源编译软件平台

经过 20 多年的不断发展，开放源代码的软件日益普及。而作为其中的基石和代表，开源编译器已经成为行业中的主流，未来将会得到进一步普及。目前最为主流的开源编译器是 GCC 和 Open64，前者普及程度高，后者优化性能好，是国内研究、产品开发和教育中普遍使用的开源编译软件平台。

GCC 是应用最广泛的开源可重定向编译系统[115]。GCC 全称 GNU Compiler Collection（GNU 编译器集），发源于自由软件基金会的 GNU 计划，基于 GNU 公共许可证授权。GCC 是 GNU 工具链里面最基本的东西，在 GNU 工具中处于核心地位，它是一个功能强大的编译器。该系统是整个开源社区的基石，具有多语言多目标支持能力，是所有 Linux 系统和部分 UNIX 系统的默认编译工具，但其设计初衷导致其并不适合用来做编译研究。在 2005 年 GCC4 发布之后，其整体结构做了很大的改动，增加了两种语言无关、体系结构无关的中间表示，方便在中间表示层面开展优化。该方案不仅增强了 GCC 的优化能力，也使得前端和中端的耦合度大大降低。GCC 编译器具有非常优秀的可重定向能力，几乎覆盖了我们能想到的所有领域，从高性能计算，到 IBM 的商用服务器，到桌面 PC 和笔记本，再到手机等移动设备，一直到 GCC 占支配地位的工业控制等嵌入式系统。

几乎所有国产处理器开发单位均基于 GCC 开发完成了高质量的编译系统，不过，目前国内研究机构在 GCC 社区中并不活跃，贡献的代码量不多，而进入主分支的代码则非常少，从 2010 年最新发布的 GCC 4.6 文档中，我们只看到国防科技大学杨灿群等人因在 Fortran 方面的工作而进入贡献者名单[124]。

Open64[114]是全球最好的开源高性能编译系统。该系统源自 SGI 的 MipsPro 编译器，之后在 2000 年开源发布并改名为 Pro64，随后 Intel 增加了针对 Itanium 处理器的各种高级

优化，并命名为 ORC（Open Research Compiler），目前定名为 Open64。其前端和 GCC4 兼容，支持多种高级语言，后端支持包括 MIPS（含国产龙芯）、Itanium、ARM、AMD64、×86-64、IA32、PowerPC、NVIDIA Cuda 在内多种体系结构的处理器。该系统最初定位是高性能计算，采用 5 层中间表示，便于各种优化算法的实现，可以生成高质量的代码，因此得到研究领域和工业界的普遍认可和使用。目前它已经在嵌入式领域获得大量应用，在性能方面具有非常明显的优势。

国内研究机构在 Open64 社区中非常活跃，贡献颇多。中国科学院计算技术研究所、清华大学、江南计算技术研究所等单位较早参与 Open64 开源编译器的开发和维护活动，曾多次在重要的国际会议中组织相关学术活动，已经成为 Open64 的主要代码贡献者，而基于 Open64 所完成的国产处理器编译系统均表现出优于 GCC 的良好性能。中国科学院计算技术研究所是 Open64 IA64 后端的主要贡献者，目前是龙芯后端的维护者。清华大学目前是 Open64 编译器的主要维护者之一，2003 年在 Open64（ORC）的基础上率先开发出了支持 IA64 的开源 OpenMP 编译器[125,126]，相关成果被多所国外大学和研究机构使用，而且从 2005 年起，为 Open64 编译器贡献了C++ 异常处理、内嵌汇编支持、编译器自举支持、PowerPC 后端等重要代码，并在别名分析、数据预取、数据布局等方面开展了深入的研究工作。

4.2　编译课程教育

编译课程是计算机专业的一门核心课程，也是师生普遍反映难教、难学的一门课程，一度有“计算机系学生要不要开设编译原理课程”的讨论。

我们认为编译相关课程非常重要，是计算机专业人士的标志性课程之一。编译系统在计算机科学技术的发展历史中发挥了巨大作用，是计算机系统的核心支撑软件。编译相关课程是编译人才培养的重要途径，应当从战略高度来看待编译人才的培养。编译原理一直以来是国内外大学计算机相关专业的重要课程，其知识结构贯穿程序设计语言、系统环境和体系结构，其理论基础是联系计算机科学和计算机系统的典范。编译系统构造的基本原理和技术，将为学生深入学习计算机系统相关的专业知识，以及今后从事科学研究或技术开发工作打下扎实的基础。

全国编译课程教学研讨会是进行编译教学讨论的重要平台。2009 年，由中国计算机学会教育专业委员会主办，北京航空航天大学承办的“2009 年全国编译课程教学研讨会”在京召开。来自清华大学、国防科技大学、浙江大学、北京航空航天大学、复旦大学、西安交通大学等国内 40 余所院校的 50 余名活跃在编译课程教学科研一线的教师参会，在会上就编译课程教学的定位、课程内容组织、教学改革、课程实践、学生培养等方面进行了热烈的交流与讨论。会议为一线编译课程任课教师提供了一个交流的平台，类似的教学研讨会今后将定期举行，为课程教学经验的交流推广，为促进我国高校在编译课程上的教学改革和课程建设方面做出重要贡献。

这次会议中，来自北京工业大学、国防科技大学和北京航空航天大学的国家级精品

编译课程责任教授蒋宗礼[127]、王挺[128]和张莉[129]做了精彩发言。复旦大学臧斌宇教授做了编译技术研究热点和教学思考的特邀报告。来自不同学校的近20位一线编译课程教师做了交流报告。其中，吉林大学在教育部－微软和吉林省精品课程“编译原理与实现技术”课程建设和教学方面开展了一系列工作，包括编写系列教材，实施双语教学、采用面向问题、“实例”驱动的模块化授课方式，开设“多层次－多目标－多效果”的实践课程等[130]，取得了较好的效果；清华大学的“编译原理”系列课程自2005年率先在高校教学实践领域引入博客系统为同学交流协作服务，教学中引入业界最重要的编译基础设施GCC和Open6编译器，结合国际研究前沿开展实践教学[131]，其教学方法和实践组织都颇具特色。

5 总结和展望

基础软件产品方向是《国家中长期科学和技术发展规划纲要》确定的国家科技重大专项第一项的三个方向之一，提高基础软件产品的自主知识产权拥有量和自主品牌的市场占有率是国家战略要求[132]。国家对软件基础能力建设尤其重视，包括基础软件的战略规划、标准研制、测试评价、知识产权和公共服务等各个方面。目标是开展共性支撑能力建设，实现在新兴领域加速技术成果整合推进和产业推动，以大幅提高软件企业的核心竞争力。

编译技术应当得到更多的重视和支持。编译系统是信息产业链中沟通处理器和应用程序最为关键的环节。从这个角度来看，编译系统是基础软件的基础，属于战略必争领域，是信息产业的核心竞争力之一，对于我国信息产业的可持续发展具有重要的意义。

伴随着我国信息产业的不断发展壮大，近年来，在围绕国产处理器支持、多核/众核处理器编程模型和优化等方面的多年深入研究中，我国取得了长足进步。逐步培养出多支成熟稳定的科研队伍；开发完成了龙芯、飞腾、神威睿智等国产处理器和PowerPC等商用处理器的高性能产品级编译系统；在Open64等国际主流开源编译社区中的参与程度不断提高，贡献了大量高质量的代码；完成了一系列高水平的优化方法和检测工具；连续几年在PLDI、CGO（Code Generation and Optimization）[133]等重要国际顶级会议上发表高水平研究论文。所有这些成就都表明，经过几代研究人员的共同努力，在编译这一信息产业发展急需的关键技术领域中，我国正在逐步走向世界前沿技术的制高点，这些工作将为我国信息技术发展走向国际前列奠定坚实基础。

我们认为，在未来5到10年中，针对多核/众核架构和应用领域的编程模型与优化、针对嵌入式应用系统的优化、提高软件的可信程度仍将是编译技术发展的三个重要方向，研究人员将面临来自这三个方向的诸多挑战和难题。针对特定计算密集领域的研究工作，如气象和环境模拟、电力系统模拟、地球物理数据处理等领域的编程模型和优化，将为新型体系结构计算设备的充分利用提供支持；以完整产业链建设为目标，围绕国产多核处理器和国产操作系统的编译优化、集成开发环境和运行时支持系统也将持续发展。而

面向移动计算的嵌入式系统编译优化，以代码体积、功耗为目标的编译优化将更为普及，同时针对嵌入式设备浏览器应用的 Javascript 等语言的动态编译和优化技术也将受到更多的关注。此外，针对航空航天、核电等安全关键领域的可信软件构造也将吸引众多研究人员的目光，形式化验证和已验证编译器等方法和工具在这里将大有作为。

为了迎接上述挑战，建议从以下 4 个方面开展深入细致的工作：

1）**夯实理论基础**。编译的理论基础是联系计算机科学和计算机系统的典范，而从目前掌握的文献情况来看，国内研究人员有“重编译技术和系统，轻语言基础和设计”的倾向，一个明显的例子是，最近几年国内研究人员在编译技术顶级会议 PLDI 和编译后端优化顶级会议 CGO 上屡有斩获，而在语言理论的顶级会议 POPL（Principles of Programming Languages）[134] 基本还是空白。如果这样的势头无法扭转，很可能会成为未来国内编译技术和语言研究发展的瓶颈。因此，我们建议将支持和关注适当投向编程语言理论研究。

2）**掌握系统平台**。基于开源系统平台开展编译技术研发是目前的最佳选择，而掌握开源软件核心技术和开源社区话语权是发展的关键。编译器是最为复杂的基础软件之一，其代码量在数百万行量级，既包括大量精巧的优化模块，又包括很多具体、繁琐的语言细节处理，因此，完全重写不太现实，也不是十分必要。掌握优化等开源软件核心技术，参与并引导开源社区朝着有利于自身的方向发展，并能够基于开源编译器开发自主产权的优化模块，以支持国内实际需求是其中的核心问题。但是目前国内比较普遍的情况是“重开源平台的使用和改进，轻开源社区参与和建设”。典型的例子是，国内开发人员基于 GCC 为几乎所有国产处理器开发了配套编译系统，而真正在 GCC 社区中活跃的开发人员却非常少，所贡献的代码合并进入 GCC 主分支的也微乎其微。开源社区中的基本准则是“有贡献才有发言权”，如果这种情况持续下去，国内研发人员将丢失话语权，这将非常不利于国内开源编译器系统的可持续发展。值得欣慰的是 Open64 开源社区的国内研发人员相当活跃，所贡献的几十万行代码已经合并进入主分支。因此，建议从以下三个方面努力，通过贡献更多、更好代码的方式获取开源社区的话语权：开发人员提高参与意识，积极融入开源社区；同时加强代码质量控制并遵循开源社区游戏规则；通过在国内建立开源平台镜像站点或者分支站点，加强国内的社区建设。

3）**建设评估体系**。编译优化评估非常重要。编译优化技术具有很强的实用性，有可能在相同的硬件平台上实现软件性能的成倍提升，而优化性能往往和应用程序本身的特点密切相关，为此，基准测试程序成为编译技术评价的关键。目前广泛采用的以高性能 SPEC CPU 和嵌入式 EEMBC[135] 为代表的基准测试程序，以及正确性相关的测试程序，主要来自于国外，很难体现国内规模巨大的信息产业的特点，也使得针对国内应用程序的优化评估较为困难。因此，建议针对国内应用发展需求，采用产学研联盟共享、共建的方式，收集并建设具有中国特色、具有自主知识产权的基准测试程序集合，建设标准测试流程和评估框架体系。

4）**培育人才队伍**。人才是编译技术这一高技术领域的核心竞争力。应当建立适当的交流和培训机制，在继续办好编译学术和教学研讨会的同时，积极组织和参加 PLDI、

CGO 等国际会议的培训课，加强国内教师和研究人员的跨单位合作与访问，为学生跨单位实习提供便利条件，通过 CCF 相关专业委员会等平台开展学术前沿讲座等。更为重要的是，应当定期开展开源基础软件培训，利用暑期等时间，邀请国内外重要基础软件开源社区领袖开设培训课程，从技术前沿和社区组织等不同角度开展深入交流。通过多种形式的交流和培训活动，吸引最优秀的人才加入到编译研究行列，并通过这些活动开阔视野、了解前沿、参与社区，为后续研发工作奠定基础。

相信在 5 到 10 年后重新审视编译技术进展的时候，透过研发人员辛勤的汗水和清晰的代码，我们将欣喜地看到更多沉甸甸的收获，而那些收获将意味着在编译技术这个信息产业的关键领域中，中国研究人员已经走向世界前沿技术的制高点。

致谢 本报告的编写得到国家自然科学基金、国家科技重大专项和国家高技术研究发展计划（863）相关项目资助。北京大学梅宏教授提出编写本报告的最初设想，报告内容组织受到台湾中央研究院资讯所游本中教授的启发，复旦大学臧斌宇教授为本报告撰写工作机制提出宝贵意见，英特尔中国公司陈兴中先生给出很多关于开源编译器的建议，国防科技大学杨灿群教授、江南计算技术研究所李中升高工、中国科技大学陈意云教授、北京科技大学胡长军教授、吉林大学金英教授、复旦大学陈海波博士提供了大量准确的第一手资料，北京工业大学蒋宗礼教授审阅初稿并给出重要建议，还有很多同行给出了建设性的意见，在此向所有为本报告提出建议、提供帮助的人表示最真诚的谢意。报告成文仓促，限于编者水平和文献渠道，错误和遗漏在所难免，恳请读者指正，编者愿为此负责。

参考文献

[1] ACM Turing Award, http://awards.acm.org/homepage.cfm? awd=140, 2010.

[2] Programming Language Design and Implementation, http://www.sigplan.org/pldi.htm, 2010.

[3] Estimated impact of publication venues in Computer Science—2003, http://citeseer.ist.psu.edu/impact.html, 2010.

[4] 全国编译课程教学研讨会, http://scse.buaa.edu.cn/news/140.html, 2009.

[5] The Message Passing Interface (MPI) standard, http://www.mcs.anl.gov/research/projects/mpi/, 2010.

[6] The OpenMP API specification for parallel programming, http://openmp.org/wp/, 2010.

[7] The OpenTM Transactional API, http://opentm.stanford.edu/, 2010.

[8] Berkeley UPC-Unified Parallel C, http://upc.lbl.gov/, 2010.

[9] CUDA, Compute Unified Device Architecture, http://www.nvidia.com/object/cuda_home_new.html, 2010.

[10] OpenCL (Open Computing Language), http://www.nvidia.com/object/cuda_opencl_new.html, 2010.

[11] 卢兴敬，商磊，陈莉. POM：一个 MPI 程序的进程优化映射工具，HPC China 2009.

[12] 王毅. MPI 程序通信行为循环不变分析，中国科学院研究生院硕士学位论文，2010.

[13] 吴俊杰，杨学军，刘光辉，唐玉华. 面向 OpenMP 和 OpenTM 应用的并行数据重用理论[J]. 软件

学报（已录用）.

[14] 吴俊杰，潘晓辉，杨学军．面向非一致 Cache 的智能多跳提升技术[J]．计算机学报，2009，32(10)：1887-1895.

[15] Xue-Jun Yang, Jun-Jie Wu, Kun Zeng, Yu-Hua Tang. Managing Data-Objects in Dynamically Reconfigurable Caches[J]. Journal of Computer Science and Technology, 2010, 25(2):232-245.

[16] Junjie Wu, Xuejun Yang. Optimizing the Management of Reference Prediction Table for Prefetching and Prepromotion[J]. Journal of Computers, Academy Publisher, Finland, 2010, 5(2): 242-249.

[17] Junjie Wu, Xiaohui Pan, Xuejun Yang. Software Prepromotion for Non-Uniform Cache Architecture[J]. Journal of Software, Academy Publisher, Finland, 2010, 5(1): 11-19.

[18] Junjie Wu, Xiaohui Pan, Guanghui Liu, Baida Zhang, Xuejun Yang. Parallel Data Reuse Theory for OpenMP Applications. In Proceedings of the 10th ACIS International Conference on Software Engineering, Articial Intelligence, Networking and Parallel/Distributed Computing, IEEE Press, 2009: 516-523.

[19] 尉红梅，姚建华．并行语言及编译技术现状和发展趋势[J]．计算机工程，2004 年第 30 卷增刊．

[20] Changjun Hu, Jue Wang, Jianjiang Li. Language Support for Multi-Paradigm and Multi-Grain Parallel on SMP-CLUSTER. International Journal of Computers and Applications. Actapress. Canada. issue 2, 2007.

[21] Jue Wang, Changjun Hu, Jianxin Lai, Yudi Zhao, Suqin Zhang. Multi-Paradigm and Multi-Grain Parallel Execution Model Based on SMP-Cluster. IEEE John Vincent Atanasoff International Symposium on Modern Computing, IEEE Society Press, 2006.

[22] Changjun Hu, Guangli Yao, Jue Wang, Jianjiang Li. Transforming the Adaptive Irregular Out-of-Core Applications for Hiding Communication and Disk I/O. International Conference on Grid Computing, High-Performance and Distributed Applications, Lecture Notes in Computer Science 4804, Springer publisher, 2007.

[23] 胡长军，李静，王珏，姚广利，李永红，丁良，李建江．一类非规则并行应用问题的通信集生成算法[J]．计算机学报，2008.

[24] 王珏，胡长军，张纪林，李建江．一种数据并行中的群通信优化策略[J]．计算机学报，2008.

[25] OpenMP Application Program Interface, Version 3.0, OpenMP Architecture. Review Board, May 2008.

[26] H. Peter Hofstee. Power Effieient Proeessor Architecture and The Cell Processor. Proceedings of the 11th Intel's Symposium on High-Performance Computer Architecture. IEEE Computer Soeiety, 2005.

[27] 商磊，陈莉，李恒杰．UPC 细粒度重叠优化，HPC China 2008.

[28] 李鹏程．众核平台上以层次数据分布为基础的编译并行优化研究，中国科学院研究生院硕士学位论文，2010.

[29] 刘雷．CUDA 平台上的 UPC 语言扩展与优化，中国科学院研究生院硕士学位论文，2010.

[30] 方燕飞，姜小成，漆锋滨．UPC 并行循环优化的研究与实现[J]．计算机工程与应用，2006，(42):65-68.

[31] 方燕飞，王俊，漆锋滨．UPC 共享访问消息向量化[J]．计算机应用与软件，2008，25(6).

[32] 文延华，黄传信，漆锋滨．Berkeley UPC 编译技术分析[J]．高性能计算技术，2004，2:01-05.

[33] Cilk-5.4.6 Reference Manual, http://supertech.csail.mit.edu/cilk/manual-5.4.6.pdf, 1998.

[34] X10, Language Specification, http://dist.codehaus.org/x10/documentation/languagespec/x10-latest.pdf, Version 2.0.3, 2010.

[35] Lei Wang, Huimin Cui, Yuelu Duan, Fang Lu, Xiaobing Feng, Pen-Chung Yew. "An Adaptive Task

Creation Strategy for Work-Stealing Scheduling", CGO2010.

[36] Canqun Yang, Zhen Ge, Juan Chen, Feng Wang, Qiang Wu. Accelerating PQMRCGSTAB Algorithm on GPU. Proceedings of the Combined Workshops on UnConventional High Performance Computing Workshop Plus Memory Access Workshop (UCHPC-09), pp. 11-16. Italy, 2009.

[37] Canqun Yang, Zhen Ge, Juan Chen, Feng Wang, Yunfei Du. Solving 2D Nonlinear Unsteady Convection-Diffusion Equations on Heterogeneous Platforms with Multiple GPUs. The Fifteenth International Conference on Parallel and Distributed Systems (ICPADS'09), Shenzhen, 2009: 961-966.

[38] Canqun Yang, Qiang Wu, Juan Chen, Zhen Ge. GPU Acceleration of High-speed Collision Molecular Dynamics Simulation. IEEE Ninth International Conference on Computer and Information Technology (CIT2009), Xiamen, 2009: 254-259.

[39] 葛振, 杨灿群, 吴强, 陈娟. 线性系统求解中迭代算法的 GPU 加速方法[J]. 计算机工程与科学, 2009, 31(A1): 179-182.

[40] 吴强, 杨灿群, 葛振, 陈娟. 使用 GPU 加速分子动力学模拟中的非绑定力计算[J]. 计算机工程与科学, 2009, 31(A1): 46-49.

[41] 刘来国, 徐炜遐, 杨灿群, 陈娟. 基于 GPU 的 LARED-P 算法加速[J]. 计算机工程与科学, 2009, 31(A1): 59-63.

[42] Xuejun Yang, Xiangke Liao, Weixia Xu, Junqiang Song, Qingfeng Hu, Jinshu Su, Liquan Xiao, Kai Lu, Qiang Dou, Juping Jiang, Canqun Yang. Heterogeneous System and Cooperatin Computing. Frontiers of Computer Science in China. (已录用).

[43] Rong Chen, Haibo Chen, Binyu Zang. Tiled MapReduce: Optimizing Resource Usages of Data-parallel Applications on Multicore with Tiling. The Nineteenth International Conference on Parallel Architectures and Compilation Techniques (PACT 2010, to appear). Vienna, Austria, September, 2010.

[44] Xuejun Yang, Yunfei Du, Panfeng Wang, Hongyi Fu. FTPA: Supporting Fault Tolerant Parallel Computing through Parallel Recomputing. IEEE Transactions on Parallel and Distributed Systems, 2009, 20 (10): 1471-1486.

[45] Hongyi Fu, Xuejun Yang. Fault Tolerant Parallel FFT Using Parallel Failure Recovery. 2009 International Conference On Computational Science and Its Applications (ICCSA), pp. 257-261, 2009.

[46] Xuejun Yang, Panfeng Wang, Hongyi Fu, Yunfei Du, Zhiyuan Wang, Jia Jia. Compiler-Assisted Application-Level Checkpointing for MPI Programs. The 28th International Conference on Distributed Computing Systems (ICDCS 2008), pp. 251-259, Beijing, China, June 17-20, 2008.

[47] Panfeng Wang, Yunfei Du, Hongyi Fu, Xuejun Yang, Haifang Zhou. Static Analysis for Application-Level Checkpointing of MPI Programs. The 10th IEEE International Conference on High Performance Computing and Communications (HPCC2008), pp. 548-555, Dalian, China, 2008.

[48] Hongyi Fu, Yunfei Du, Panfeng Wang, Jia Jia, Xuejun Yang. GiFT: Automating FTPA Implementation for MPI Programs. The 14th IEEE International Conference on Parallel and Distributed Systems (ICPADS'08), pp. 91-98, Melbourne, Victoria, Australia, 2008.

[49] Panfeng Wang, Zhiyuan Wang, Yunfei Du, Xuejun Yang, Haifang Zhou. Optimal Placement of Application-Level Checkpoints. 2008 International Workshop on Parallel Algorithm and Parallel Software (IWPAPS2008), pp. 853-858, Dalian, China, 2008.

[50] 杜云飞, 唐玉华. 容错并行算法的性能分析[J]. 计算机科学, 2009, 36(9): 248-251.

[51] 王攀峰, 杜云飞, 富弘毅, 杨学军, 周海芳. 并行复算: 一种面向高性能计算的新的容错方法

[J]. 计算机科学, 2009, 36(3).

[52] 杜云飞, 王攀峰, 富弘毅, 周海芳, 杨学军. 矩阵 LU 分解的容错并行算法设计与实现. 微电子学与计算机, 2008, 25(10):1-4.

[53] Haibo Chen, Jie Yu, Rong Chen, Binyu Zang, Pen-chung Yew. POLUS: A POwerful Live Updating System. In Proceedings of 29th International Conference on Software Engineering (ICSE-2007), pp. 271-281. Minneapolis, MN, USA, May 2007.

[54] Haibo Chen, Jie Yu, Chengqun Hang, Binyu Zang and Pen-chung Yew. Dynamic Software Updating Using a Relaxed Consistency Model. IEEE Transactions on Software Engineering (to appear), 2010.

[55] Juan Chen, Yong Dong, Xuejun Yang, Panfeng Wang. Energy-Constrained OpenMP Static Loop Scheduling. In the Proceedings of the 10th IEEE International Conference on High Performance Computing and Communications (HPCC-08). p. 139-146. IEEE Computer Society.

[56] Yong Dong, Juan Chen, Xuejun Yang, Lin Deng, Xuemeng Zhang. Energy-Oriented OpenMP Parallel Loop Scheduling. In the Proceedings of the seventh International Symposium on Parallel and Distributed Processing and Applications (ISPA-08), p. 162-169.

[57] Yong Dong, Juan Chen, Xuejun Yang, Canqun Yang, Lin Peng. Low Power Optimization for MPI Collective Operations. In the Proceedings of the 9th International Conference for Young Computer Scientists (ICYCS-08). p. 1047-1052. Zhang Jia Jie, Hunan, China - November 18-21, 2008.

[58] Yong Dong, Juan Chen, Tao Tang. Power Measurements and Analyses of Massive Object Storage System. In the Proceedings of the International Symposium on Frontier of Computer Science, Engineering and Applications (CSEA-10), be held in conjunction with the 10th IEEE International Conference on Computer and Information Technology (CIT-10). June 29-July 1, Bradford, UK.

[59] Yongpeng Liu, Hong Zhu. A Survey of Research on Power Management Techniques for High Performance Systems. Software Practice and Experience. 10.1002/SPE.952. 13 Jan. 2010.

[60] 易会战, 刘永鹏. 改善系统能量效率的体系结构方法: 并行处理[J]. 计算机学报, 2009, 32(12):2475-2481.

[61] 王桂彬, 杨学军, 徐新海, 林一松, 李鑫. 异构系统功耗感知的并行循环调度方法[J]. 软件学报 (已录用).

[62] 董勇, 陈娟. 并行存储系统功耗优化[J]. 计算机科学与工程, 2009, 11 期.

[63] 刘勇鹏, 卢凯, 刘勇燕, 武林平. 高性能计算中处理器功耗特征的评测与分析[J]. 计算机工程与科学, 2009, 31(11).

[64] Lei Liu, Li Chen , Chengyong Wu, Xiaobing Feng. Global Tiling for Communication Minimal Parallelization on Distributed Memory Systems. Euro-Par 2008: 382-391.

[65] Lei Liu, Dingfei Zhang, Hengjie Li, Li Chen. Automatic Implementation of Multi-partitioning Using Global Tiling. ICPADS 2008: 673-680.

[66] 刘雷. 提高并行性和数据局部性的循环变换技术研究. 博士论文.

[67] Standard Performance Evaluation Corporation, http://www.spec.org/, 2010.

[68] Chaohao Xu, Jianhui Li, Tao Bao, Yun Wang, Bo Huang. Metadata Driven Memory Optimizations in Dynamic Binary Translator. In SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE 2007). 2007: 148-157.

[69] Jianjun Li, Chenggang Wu, Wei-Chung Hsu. An Evaluation of Misaligned Data Access Handling Mechanisms in Dynamic Binary Translation Systems, CGO 2009.

[70] Zhengjiang Wang, Chenggang Wu, Pen-Chung Yew. On Improving Heap Memory Layout by Dynamic Pool Allocation, CGO2010.

[71] 史晓华，刘超，金茂忠，郭鹏．即时编译器中的轻量级指令调度算法[J]．计算机工程，2007，33(15)：3-6.

[72] 史晓华，吴甘沙，金茂忠，LUEH Guei-Yuan，刘超，王雷．在开放世界中实现逃逸分析[J]．软件学报，2008，19(03)：522-532.

[73] 王正华，陆平静，车永刚．迭代编译优化技术综述[J]．计算机工程与应用，2008，44(32)：1-5.

[74] Yang Chen, Yuanjie Huang, Lieven Eeckhout, Grigori Fursin, Liang Peng, Olivier Temam, Chengyong Wu. Evaluating Iterative Optimization across 1000 Data Sets. PLDI 2010.

[75] 漆锋滨，王飞，李中升．反馈指导的链式结构预取优化[J]．软件学报，2009，第20卷增刊．

[76] 漆锋滨，姜军，王超．反馈式编译优化在寄存器分配中的应用技术[J]．计算机应用与软件，2009，126卷．

[77] 白书敬，李中升，漆锋滨．反馈式编译优化在转移预测中的研究[J]．计算机工程，2006，第4期．

[78] D. Chen, N. Vachharajani, R. Hundt, S. Liao, V. Ramasamy, P. Yuan, W. Chen, W. Zheng. Taming Performance Counters for FDO Compilation. In Prog. Code Generation and Optimization (CGO'10), pages 42-52, Toronto, ON, Canada. April 2010. ACM Press.

[79] R. Levin, I. Newman, and G. Haber Complementing Missing and Inaccurate Profiling Using a Minimum Cost Circulation Algorithm. In: Proceedings of the High Performance Embedded Architectures and Compilers, Third International Conference. Berlin: Springer, 2008: 291-304.

[80] J. Yan, W. Chen, W. Zheng. PMU Guided Structure Data- Layout Optimization. Tsinghua Science and Technology. Accepted.

[81] Hongtao Yu, Zhaoqing Zhang, Xiaobing Feng, Wei Huo, Jingling Xue. Level by Level: Making Flow- and Context- Sensitive Pointer Analysis Scalable for Millions of Lines of Code, CGO 2010.

[82] Yuelu Duan, Xiaobing Feng, Lei Wang, Chao Zhang, Pen-chung Yew. Detecting and Eliminating Potential Violation of Sequential Consistency for Concurrent C/C++ Programs, CGO2009.

[83] 卢锡城，李根，卢凯，张英．面向高可信软件的整数溢出错误的自动化测试[J]．软件学报，2010，02：179-194.

[84] Tony Hoare. The Verifying Compiler: A Grand Challenge for Computing Research. In Proc. 2003 International Conference on Compiler Construction (CC'03), LNCS Vol. 2622, pages 262-272, Warsaw, Poland, Springer-Verlag Heidelberg, 2003.

[85] Mary Hall, David Padua, Keshav Pingali. Compiler Research: the Next 50 Years, Communications of the ACM, 2009, 52(2):60-67.

[86] Shengyuan Wang, Yuan Dong. A Verifiable Low-level Concurrent Programming Model Based on Colored Petri Nets. the Proceedings of Petri Nets and Distributed Systems 2008 (a satellite workshoip of 29th ATPN conference), Xi'an, China, June 23-24, 2008.

[87] 梁英毅，王生原，董渊．一种基于P/T网的低级并发程序验证模型[J]．中国科学-信息科学（中国科学F辑），2010，40(1)：13-32.

[88] Yunmin Zhu, Liwei Zhang, Shengyuan Wang, Yuan Dong, Suqin Zhang. Verifying Parallel Low-level Programs for Multi-core Processor. In Proc. NASAC'08, 282-287, 2008.

[89] T. Lindholm, F. Yellin. The java Virtual Machine Specification (second edition), 1999.

[90] Yuan Dong, Kai Ren, Shengyuan Wang, Suqin Zhang. Certify Once, Trust Anywhere — Modular Certification of Bytecode Programs for Certified Virtual Machine. APLAS2009. Seoul, Korea 14- 16 December 2009.

[91] Yuan Dong, Kai Ren, Shengyuan Wang, Suqin Zhang. Construction and Certification of a Bytecode Virtual Machine (In Chinese). Journal of Software. 2010, 21(2): 305-317.

[92] 陈意云，华保健，葛琳，王志芳．一种用于指针程序安全性证明的指针逻辑[J]．计算机学报，2008，31(3):372-380.

[93] 陈意云，李兆鹏，王志芳，华保健．一种用于指针程序验证的指针逻辑[J]．软件学报，2010，21(3):415-426.

[94] 梁红瑾，张昱，陈意云，李兆鹏，华保健．处理指针相等关系不确定的指针逻辑[J]．软件学报，2010，21(2):334-343.

[95] Zhifang Wang, Yiyun Chen, Zhenming Wang, Baojian Hua. Automated Verification of Pointer Programs in Pointer Logic Frontiers of Computer Science in China, 2008, 2(4):380-397.

[96] Zhen-ming Wang, Yi-yun Chen, Zhi-fang Wang. Automated Theorem Prover for Pointer Logic. Journal of Software, 2009, 20(8):2037-2050.

[97] 杨思敏，李兆鹏，庄重，张臻婷．出具证明编译器中线性整数命题证明的自动生成[J]．小型微型计算机系统，已录用．

[98] Yiyun Chen, Lin Ge, Baojian Hua, Zhaopeng Li, Cheng Liu. Design of a Certifying Compiler Supporting Proof of Program Safety. In Proceedings of 1st IEEE/IFIP International Symposium on Theoretical Aspects of Software Engineering, IEEE CS press, June 2007: 127-136.

[99] Yiyun Chen, Lin Ge, Baojian Hua, Zhaopeng Li, Cheng Liu, Zhifang Wang. A Pointer Logic and Certifying Compiler. Frontiers of Computer Science in China, 2007, 1(3):297-312.

[100] 李兆鹏，陈意云，葛琳，华保健．一种汇编程序的形式验证框架[J]．计算机研究与发展，2008，45(5):825-833.

[101] Long Li, Yu Zhang, Yi-yun Chen Yong Li. Certifying Concurrent Programs Using Transactional Memory. Journal of Computer Science and Technology, 2009, 24(1):110-121.

[102] Yong Li, Yu Zhang, Yiyun Chen, Ming Fu. On the Verification of Strong Atomicity of Programs Using STM. In Proc. of 3rd IEEE International Conference on Secure Software Integration and Reliability Improvement (SSIRI2009), pages 117-125, IEEE CS press, July 2009, Shanghai, China.

[103] Ming Fu, Yu Zhang, Yong Li. Formal Reasoning about Concurrent Assembly Code with Reentrant Locks. Proc. of 3rd IEEE International Symposium on Theoretical Aspects of Software Engineering (TASE 2009), pages 233-240, IEEE CS press July 2009, Tianjin, China.

[104] Ming Fu, Yu Zhang, Yong Li. Formal Verification of Concurrent Programs with Read-Write Locks. Frontiers of Computer Science in China, 2010, 4(1): 65-77.

[105] Lei Zhao, Yu Zhang. Implementing Atomic Section by Using Hybrid Concurrent Control. Proc. of 2007 IFIP International Conference on Network and Parallel Computing Workshops, Sep. 18-21 2007, Dalian, China. IEEE Computer Society Order Number P2943: 642-647.

[106] R. Sekar, et. al. Model-carrying Code: a Practical Approach for Safe Execution of Untrusted Applications. SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles, 2003, 15-28, ACM.

[107] 金英，张晶等．多线程 Java 程序安全相关行为模型静态检查方法[J]．计算机学报，2009.

[108] Ying Jin. Formal Verification of Protocol Properties of Sequential Java Programs. Compsac2007, Beijing. 2007.
[109] 魏达，金英，等．基于开源 JVM 的安全策略强制实施[J]．电子学报，2008，37(12A):35-41.
[110] 李泽鹏，金英．基于 Java 平台实现安全行为模型验证[J]．计算机工程与科学，2007.
[111] 徐家福，宋方敏，钱士钧，戴静安，张云洁．量子程序设计语言 NDQJava[J]．软件学报，2008(01)：1-8.
[112] 徐家福，宋方敏．量子程序设计语言初探[J]．中国科学 E 辑，2008(06).
[113] 宋方敏，钱士钧，戴静安，张云洁，徐家福．量子程序设计语言 NDQJava 处理系统[J]．软件学报，2008 (01)：9-16.
[114] Open64 Compiler Website, http://open64.net, 2010.
[115] GNU Compiler Collection, http://gcc.gnu.org, 2010.
[116] 张铎．面向 PowerPC 体系结构的 Open64 编译器实现．申请清华大学工学硕士学位论文，2009.
[117] Ming Lin, Zhengyang Yu, Duo Zhang, Shengyuan Wang, Yuan Dong. Retargeting the Open64 Compiler to PowerPC Processor, IEEE CS Press, ICESS2008, 2008: 152-157.
[118] Weihua Zhang, Xinglong Qian, Ye Wang, Binyu Zang, Chuanqi Zhu. Optimizing Compiler for Shared-Memory Multiple SIMD Architecture. LCTES 2006: 199-208.
[119] Qin Wang, Junpu Chen, Weihua Zhang, Min Yang, Binyu Zang. Optimizing Software Cache Performance of Packet Processing Applications. LCTES 2007: 227-236.
[120] IEEE Std 716-1995, IEEE Standard Test Language for All Systems – Common/Abbreviated Test Language for All Systems (C/ATLAS), http://grouper.ieee.org/groups/scc20/td/ATLAS%20Standard.htm, 2010.
[121] Guo Degui, Liu Lei. The Denotational Semantics of the Signal Statement in ATLAS. Proceedings - 4th ACIS International Conference on Software Engineering Research, Management & Applications, SERA2006, Seattle, Washington, USA, August 9-11, 2006: 177-182.
[122] Guo De-Gui, Liu Lei, Li Wen-Gin. Transformation from Test Language ATLAS to C++. Proceedings - Fifth International Conference on Computer and Information Technology, CIT 2005, 2005: 848-852.
[123] 郭德贵，刘磊，金英，程斌．ATLAS 语言实现中设备分配算法研究[J]．电子学报，2007(11).
[124] Contributors to GCC (4.6), http://gcc.gnu.org/onlinedocs/gccint/Contributors.html#Contributors, 2010.
[125] 陈永健，李建江，王生原，郑纬民．基于 ORC 的 OpenMP 编译器设计与实现[J]．清华大学学报，2005，45(1)：69-72.
[126] 陈永健，李建江，王生原，王鼎兴．ORC-OpenMP: An OpenMP compiler based on ORC. Proc. of ICCS2004, Lecture Notes in Computer Science 3038, Springer-Verlag, 2004: 414-423.
[127] 北京工业大学编译原理网站，http://vod.bjut.edu.cn/web/jp/06sb/byyl/index.htm，2010.
[128] 国防科学技术大学编译教学网站，http://jpkc2007.nudt.edu.cn/byyl/，2010.
[129] 北京航空航天大学编译教学网站，http://www.sei.buaa.edu.cn/compile/，2010.
[130] 吉林大学编译教学网站，http://softlab.jlu.edu.cn/2005/default.php? compilation，2010.
[131] 清华大学编译教学网站，http://soft.cs.tsinghua.edu.cn/blog，2010.
[132] 国家中长期科学和技术发展规划纲要（2006—2020 年），http://www.gov.cn/jrzg/2006-02/09/content_183787.htm，2010.
[133] Code Generation and Optimization, http://www.cgo.org/, 2010.

[134] Principles of Programming Languages, http://www.sigplan.org/popl.htm, 2010.
[135] Embedded Microprocessor Benchmark Consortium, http://www.eembc.org/home.php, 2010.

作者简介

董渊 CCF 系统软件专委会委员。博士，清华大学计算机系副教授，主要研究领域为操作系统、编译系统、基于语言的可信软件。E-mail: dongyuan@tsinghua.edu.cn。

冯晓兵 CCF 高级会员。博士，中国科学院计算技术研究所研究员，主要研究领域为编译技术及相关工具环境。E-mail: fxb@ict.ac.cn。

王生原 CCF 高级会员。博士，清华大学计算机系副教授，主要研究领域为程序设计语言与系统、Petri 网应用。E-mail: wwssyy@tsinghua.edu.cn。

陈文光 YOCSEF 学术委员会副主席，CCF 高级会员。博士，清华大学计算机系教授，副系主任，北京市计算机学会常务理事、副秘书长，中国软件行业协会数学软件分会常务理事。E-mail: cwg@tsinghua.edu.cn。