

THE STUDENT
SUPERCOMPUTER
CHALLENGE GUIDE



超算竞赛 导引



《超算竞赛导引》编写组



科学出版社

超算竞赛导引

《超算竞赛导引》编写组

科学出版社

北京

内 容 简 介

ASC 世界大学生超级计算机竞赛由中国发起,得到了中国乃至全球众多大学的积极参与及海内外超算专家和单位的大力支持。大赛从 2012 年以来每年举办一次,目前已经发展成为全球最大规模的超算赛事。

本书以 ASC 超算竞赛为主体,深入浅出地介绍了超级计算机基础、往届超算竞赛情况以及竞赛赛题分析等内容。全书分为学习篇、竞赛篇和进阶篇三部分,共 14 课。学习篇引领读者理解超级计算机及其系统架构、评测方法等;竞赛篇详细介绍了 ASC 世界大学生超级计算机竞赛及规则;进阶篇则以实际超算竞赛赛题为例,介绍了如何针对实际超算应用赛题进行算法分析和性能优化。

本书由亚洲超算协会与浪潮集团组织国内外超算专家共同撰写,是全球第一本系统介绍超算竞赛的专著。本书将帮助青年学生通过超算竞赛了解和热爱超算,帮助教师指导学生研究超算,帮助科研工作者获悉超算如何为科技创新提供计算支持。

图书在版编目(CIP)数据

超算竞赛导引 / 《超算竞赛导引》编写组编著. —北京: 科学出版社, 2016.3
ISBN 978-7-03-047163-5

I. ①超… II. ①超… III. ①超级计算机—研究 IV. ①TP338

中国版本图书馆 CIP 数据核字(2016)第 017354 号

责任编辑: 余 丁 赵艳春 / 责任校对: 包志虹

责任印制: 张 倩 / 封面设计: 迷底书装

科学出版社出版

北京东黄城根北街 16 号

邮政编码: 100717

<http://www.sciencep.com>

新科印刷有限公司 印刷

科学出版社发行 各地新华书店经销

*

2016 年 3 月第 一 版 开本: 787×1092 1/16

2016 年 3 月第一次印刷 印张: 11 1/2

字数: 224 000

定价: 49.00 元

(如有印装质量问题, 我社负责调换)

序 一

超级计算作为当下科学研究的三大支撑手段之一，已经广泛应用于气象预报、石油勘探、地球模拟、宇宙科学、新材料研发、新药研发、汽车与航空器设计等前沿领域。同时，超算也是解决重大经济和社会问题的重要科技手段，如环境保护、基因测序、自然灾害预报、互联网深度学习等也都需要在超算平台上进行计算模拟以得出快速有效的决策。

近年，全球在超算系统建设能力有了大幅提升，最快的超级计算机已经超过 5 亿亿次的计算性能，并将在未来几年进入到百亿亿次时代，超算应用的领域也越来越广，产业发展欣欣向荣。然而在超算硬件快速发展的背后，我们也看到了超算产业面临着应用落后和人才缺乏的困境。

从 2012 年开始，浪潮集团联合亚洲超算协会共同发起世界大学生超级计算机竞赛（ASC），其目的是希望通过大赛的平台，推动中国乃至世界各国及地区间超算青年人才交流和培养，让更多的青年人才有机会接触到最新的超算科技和应用，提供给青年人才系统的超算知识培训和实际动手操作的机会，通过竞赛调动青年人在超算方面的创新热情。这是浪潮作为企业的社会责任和使命使然，希望通过此举为超算行业的长远健康发展贡献力量。

我们很欣喜地看到，ASC 超算竞赛已经得到世界范围越来越多的高校、科研机构、超算专家的积极参与和大力支持，参赛规模和赛事覆盖范围越来越广，逐渐成为世界规模最大的大学生超级计算机竞赛，与美国 SC、德国 ISC 并列为全球三大超算竞赛。ASC 超算竞赛的面向前沿、高端系统、应用导向、动手实践等特点吸引了越来越多的青年学生投入参与，而同学们在竞赛中迸发的创新热情让人惊喜，他们在竞赛中取得的创新成果让人惊叹，给实际超算应用带来了宝贵的启发和促进。更重要的是，越来越多的青年人通过 ASC 竞赛，学习超算、热爱超算，为科技产业创新提供了很好的超算人才动力。

《超算竞赛导引》是全球第一本关于超算竞赛的书，该书集合了众多优秀参赛队伍和超算领域的学者专家撰写而成，内容翔实全面，由浅入深。让超算从“旧时王谢堂前燕，飞入寻常百姓家”一直是 ASC 世界大学生超算竞赛的重要目的，希望通过本书可以帮助到更多有志于超算应用和创新的青年人才。

王恩东

中国工程院院士

高效能服务器和存储技术国家重点实验室主任

浪潮集团首席科学家

2015 年 12 月 10 日

序 二

计算已经和理论与实验一起,成为科学研究和工程设计的基本手段之一。基于超级计算的数值模拟已广泛应用于裂变与聚变能源、新能源开发、新材料设计、天气预报、电磁信息安全、油气勘探、飞行器和高速列车及汽车结构优化设计、大型工程仿真等国民经济重大应用领域和生命科学、材料科学等前沿基础研究领域,达到了缩短研究周期、降低研发成本的目的。当前,超级计算已经和人们的生活息息相关,精准数值天气预报、基于模拟的新药发现、减灾防灾、环境污染治理、大规模信息服务、数据挖掘与知识发现、数字媒体与文化创意等,均可以看到超级计算的身影。

当前,我国超级计算应用水平还相对滞后,迫切需要一批青年才俊投身其中,充分展示超级计算解决实际应用的潜在能力。为此,亚洲超算协会联合浪潮集团主办了 ASC 世界大学生超级计算机竞赛,取得了良好效果,并成为全球具有影响力的赛事。本书由参加了历届超算竞赛的支撑技术团队和多支参赛团队合力撰写,旨在吸引更多的高校团队参与,通过比赛学习超级计算的相关知识和理论,从侧面推动我国超级计算应用水平的提升,达到吸引和培养青年人才的目的。

本书将引领读者理解超级计算及其系统架构、评测方法等,并详细介绍了 ASC 竞赛及规则,以及 ASC 初赛和决赛的试题及其解析和优化方法等。

浪潮集团作为在高性能服务器、容错计算机、海量存储设备、云计算、大型应用软件等多个领域具有重要影响力的企业,与亚洲超算协会合作,赞助并支撑起 ASC 竞赛,付出了艰辛的努力。本书也是在浪潮集团的大力支持下完稿的。

作为 ASC 竞赛的评审委员会主席和我国超级计算科技工作者中的一员,我衷心祝愿 ASC 竞赛越办越好,各参赛团队表现得越来越好,也衷心希望本书可以帮助各参赛团队取得更好的成绩。

莫则尧

北京应用物理与计算数学研究所副所长

ASC 世界大学生超级计算机竞赛评审委员会主席

2015 年 3 月 26 日

序 三

计算科学与工程已经成为与实验观察、理论分析并列的基本科研手段,是推动工程技术创新的重要因素,计算科学与工程支持定量化与精确化描述,对提高科学研究水平具有重要意义。计算科学与工程是 21 世纪最普遍的多学科交叉工作模式,最伟大的科学突破将在其推动下产生,而超级计算是计算科学与工程不可或缺的重要手段。

超级计算在国内外发展迅猛,美国、日本、欧盟等也投巨资研发超级计算机,并大力开展超级计算技术与应用的研究工作。我国将超级计算作为战略高技术,并列入国家中长期发展规划。超级计算长期得到国家 863 计划的支持,并已取得了长足进步。由 863 计划支持研制的天河二号超级计算机,在 2013 年 6 月全球超级计算机 Top500 强登顶后,已连续多次排名世界第一。美国无论是高性能计算机的研制能力,还是高性能计算的应用能力都远远超过其他国家,我国在超级计算应用方面与美国的差距很大,如何提高我国超级计算应用水平是亟待解决的问题。

亚洲超算协会联合浪潮集团举办的 ASC 世界大学生超级计算机竞赛为超级计算技术的推广做了一件十分有意义的工作,该竞赛受到了国内外众多大学的支持,特别是参加竞赛的国内高校越来越多,使众多的大学生了解超级计算、热爱超级计算并投身于超级计算事业。目前迫切需要一本全面介绍超级计算基本方法、基本技术、基本规则的书籍,并能介绍一些重要的竞赛题目及方法。

本书正是基于超级计算机竞赛的需求,一些超级竞赛评委、高水平教练和一些知名学者,完成了编写工作。本书的内容包含学习篇、竞赛篇与进阶篇。学习篇可让读者了解超级计算机的基本概念与基本方法,包括超级计算机系统机构、网络通信、开发环境、应用环境及评测技术,为更好地参加竞赛打下基础;竞赛篇介绍了 ASC 超级计算机竞赛的规则,并对规则进行了详细分析,增强了竞赛的透明性;进阶篇则对超级计算的初赛与决赛的试题进行了分析,给出了对策建议,特别详细介绍了一些特定应用的并行及优化方法,针对性很强,对学生在很短时间内掌握主要的并行方法与优化策略十分有意义。

本书的出版,将对超级计算技术的推广和人才培养起到积极的作用。希望本书对参赛队员的基本训练有意义,对应用的并行优化有帮助,也衷心希望 ASC 世界大学生超级计算机竞赛越办越好,竞赛水平更上一个台阶。

杨广文

清华大学高性能计算研究所所长

2012 年中国大学生超级计算机竞赛组织委员会主席

2015 年 7 月 8 日

寄 语

一

中国的发展正面临一系列重大挑战,例如,极端灾害天气渐成常态、环境污染得不到有效遏制、突发恶性传染病威胁人民健康、能源短缺制约可持续发展、粗放低效的发展模式亟待改变、经济结构的调整转型迫在眉睫。急需通过改革和创新,应对挑战,走出一条绿色、高效、健康的可持续发展之路。高性能计算应该在这个转变过程中发挥其不可或缺的作用。

过去 20 年,在国家 863 计划支持下,我国的高性能计算事业得到了长足的发展。国产超级计算机的运算速度已经多次位居世界第一。但是,必须看到,我国的高性能计算应用与发达国家相比,无论广度和深度都还有很大差距。制约高性能计算应用发展的因素很多,其中一个重要因素就是人才短缺,缺少能够熟练掌握计算工具解决实际问题的高素质人才。

ASC 世界大学生超级计算机竞赛是促进高性能计算人才培养和成长的一种有益尝试,它以比赛的形式,提高大学生的兴趣和竞争意识,培养以计算解决实际问题的能力,训练在约束条件下求最优的技能。这样的比赛经历,会对参赛学生的职业生涯有长远的影响。正因为此,ASC 竞赛举办短短几年,就受到大学生的接受和欢迎,不仅中国参赛的人数逐年提高,而且吸引了其他国家大学生的积极参与,逐渐成为高性能计算界一项有影响力的国际活动。

本书作为参与 ASC 竞赛的导引资料,深入浅出地介绍了超算的概念、竞赛的规则、竞赛平台的构建和典型算例与软件的优化方案,是帮助大学生和指导教师了解、熟悉 ASC 竞赛急需的教材。它的出版,必将进一步拓展 ASC 竞赛的影响,鼓励更多的大学生参与 ASC 竞赛,进而推动高性能计算技术与应用的进步。

钱德沛

国家 863 计划“高性能计算机及其核心软件”重大专项总体组组长

北京航空航天大学教授

ASC 世界大学生超级计算机竞赛专家委员会委员

2015 年 6 月 12 日

二

在信息技术的高速发展之下,高性能计算成为信息领域的前沿高新技术,广泛应用于科学研究、石油勘探、气象预报、航天国防、生命基因等传统领域,以及金融、政府、企业、互联网等新兴应用领域。目前,中国在超级计算机系统架构设计方面已经跃升到国际领先水平。但是,由于自主研发的应用软件和具有交叉学科背景人才的缺乏,我国还不能称为高性能计算的强国。

人才培养作为高性能计算发展的要素之一，亚洲超算协会联合浪潮集团共同举办的ASC世界大学生超级计算机竞赛，为高性能计算人才培养搭建了一个很好的平台，这个平台从中国到亚洲、从亚洲到全球，有力地推动了大学生学习和关注高性能计算的兴趣和热情，非常有助于培养和发现各应用领域优秀的高性能计算人才。

希望通过本书的出版，让更多的国内外大学生掌握高性能计算知识，了解超算竞赛的内容和规则，提高参赛者的技能和水平，真正不断地推动国内高性能产业的发展，更为高性能计算领域的人才培养和挖掘提供力量。

郑纬民

清华大学教授，中国计算机学会理事长
ASC世界大学生超级计算机竞赛指导委员会委员

2015年6月10日

三

计算是人类探索未知世界的最重要手段之一。随着研究对象复杂度的增加和人们对提高认识世界的精准度的不懈追求，对计算能力的需求越来越大。在当代社会，一个国家高性能计算（HPC，或称为超算）的能力往往直接代表并影响着其科技水平。历史已经证明，HPC在科学研究、国民经济和社会发展等方面创造的价值无论如何评估也不为过。作为HPC的最大用户之一，石油物探技术的发展长期以来与HPC技术的发展有着紧密的依赖关系，并彼此相互促进，相信在可见的未来依然是这样。

浪潮集团作为民族厂商，以非凡的眼光和魄力，关注应用层面的创新和HPC人才队伍的培养，投入巨资参与举办目前全球最大规模的ASC世界大学生超级计算机竞赛，不但宣传了品牌、促进了超算知识的普及、促进了各个国家（地区）之间超算青年人才的交流，而且为企业呈现面临的HPC问题、发现优秀的HPC人才提供了卓越的平台。

祝浪潮集团的HPC事业蒸蒸日上，为推动科学和社会的发展作出更大的贡献！

祝ASC竞赛越办越好，进一步推动中国高性能计算教育与人才的培养！

罗国安

中国石油东方地球物理公司研发中心书记
ASC世界大学生超级计算机竞赛评审委员会委员

2015年6月30日

四

面对人类文明进步中不断出现的巨大挑战性问题，超级计算的作用越来越为人们认识到，在不断战胜挑战、实现科技创新的过程中，对超级计算技术研发和应用人才的培养也愈加受到广泛的关注。ASC世界大学生超级计算机竞赛的建立和发展为促进超算领域的技

术交流、提高超算应用水平和推动超算创新人才的培养提供了一个非常重要的交流和实践的舞台，短短四年的实践证明，它使我国大学生对超级计算的兴趣显著提升，国际视野全面拓展，应用能力极大提高。

期望通过本书的出版，进一步普及超级计算的基本知识、培训超级计算的竞赛技能、提高超级计算的应用水平。吸引更多的大学生学习超算，参加竞赛，成为未来各领域超算应用的优秀人才。相信通过不断的努力，我国不仅能造出世界一流的超级计算机，更能在超算应用领域走在世界的前列。

顾一众

上海交通大学网络信息中心主任

2013 年亚洲大学生超级计算机竞赛组织委员会主席

2015 年 6 月 14 日

五

发挥超级计算机在解决人类重大科学工程问题上的推动作用，不仅需要高性能的硬件系统，更需要能真正发挥出系统性能的应用软件，而要实现这一目标，核心在于超算复合型人才的培养。ASC 世界大学生超级计算机竞赛对超算应用创新的鼓励，将加速我国超算人才的培养进程。

ASC 竞赛立足超算应用发展和人才培养，鼓励大学生亲自动手，完成超算系统的搭建和前沿应用的架构迁移、并行计算、性能优化等超算全过程，让竞赛成为超算人才培养的“实战课堂”，进而有效缓解我国当前超算软硬实力发展不对称的局面，推动超算真正成为科技创新的基础设施。

“少年强则国强”，将是未来中国成为真正的超算强国不容有失的关键环节。

许跃生

千人计划国家特聘专家

中山大学国华讲席教授

2014 年世界大学生超级计算机竞赛组织委员会主席

2015 年 6 月 14 日

六

ASC 世界大学生超级计算机竞赛已成功举办四届，应组委会邀请，我作为气候模拟领域的专家，有幸担任总决赛竞赛呈现答辩环节评委，参加了 ASC13 和 ASC14 两届赛事。这是一项集个人智慧与团队力量为一体的精彩赛事，来自世界各国的参赛队伍的蓬勃朝气、团结协作和勇于创新的精神，给我留下了深刻印象。该项赛事对推动我国高校在超算人才培养和超算应用的能力创新等方面，起到了非常积极的作用，已经成为推动超算领域青年领军人才成长的重要平台。

很高兴地看到,在 ASC14 总决赛上,我所在的中国科学院大气物理研究所大气科学和地球流体力学数值模拟国家重点实验室(IAP/LASG)研发的气候海洋模式 LICOM,被选为总决赛的应用算例之一,供各国大学生选手在全球最快的超级计算机“天河二号”上进行并行优化。参赛选手围绕提高 LICOM 并行效率提出的各种优化方案,令人交口称誉。LICOM 是 IAP/LASG 气候系统模式的核心组成部分,气候模拟研究是超算应用领域国际竞争的前沿,ASC 世界大学生超算竞赛将从培养青年超算人才的角度,为提升我国在气候模拟领域的国际竞争力提供基础。

气候模式一直以来是推动高性能计算发展的主要因素。世界上最快的计算机总是最先应用于气候模拟。《自然》在其 2006 年发表的科学计算历史回顾中明确指出,1969 年由美国地球流体实验室(GFDL)研发的世界上第一个海洋-大气耦合的气候模式是科学计算领域的一个重要里程碑。气候模式,简言之就是封装了大量物理定律的计算机程序,是对地球系统中物理、化学和生态过程的数学表达。气候模式的研发,需要软件工程师参与开发高效的、可移植的底层程序代码,发展通用软件平台;需要软件工程师和科学家建立开放的网络端口,支持通过网络技术访问模式输出结果;需要硬件工程师维护超级计算设备,为模拟工作提供硬件保障。但是,当前在世界范围内,气候模式研发都面临计算科学和软件工程人力资源不足的问题。美国国家科学院曾分析了美国在 2000~2008 年间获得计算机科学、大气和海洋科学博士学位的人数,结果表明,近十年间获得大气和海洋科学博士学位的人数稳定在每年 600 人左右,而获得计算机科学博士学位的人数,则从 2000 年的不到 800 人,很快增长到 2008 年的约 1700 人,但这其中选择从事气候模拟工作的比例很低。因此,针对气候模拟领域计算科学和软件工程人力资源不足的问题,美国国家科学院在其《国家战略报告》中呼吁美国应该通过多种方式吸引更多的优秀学生从事气候模式发展事业,提高对气候模式发展的社会认知程度和职业发展机会,提供更丰富的激励方式来吸引那些本来可能选择其他行业的软件工程人员(见《推动气候模拟的美国国家战略》第 7 章)。

我国的气候模式发展和模拟研究工作具有很好的基础,但是研究队伍的总体规模和水平较之发达国家尚有差距。当前我国气候模拟领域,同样面临着计算科学和软件工程人力资源不足的严峻问题。随着气候变化及其影响的加剧,减缓和适应气候变化的模拟和预估需求日益增加,这对气候模式提出了更高的要求。气候模式是迄今为止人类发展的最为复杂的计算机模拟工具之一,是理解、预测气候及其变化的基础,是支撑气候变化相关决策的重要工具。气候模式水平的高低,事关国家可持续发展目标的实现,事关我国在国际气候变化谈判中的话语权问题。气候模拟和预测预估,已经成为超算应用领域国际竞争的前沿。青年学子特别是来自超算领域的杰出青年学子,应该有勇气站到这一国际竞争的前沿,为提升我国的国际竞争力贡献自己的聪明才智!

周天军

中国科学院大气物理研究所 LASG 国家重点实验室
地球气候系统模式研发与应用研究创新团队首席科学家
ASC 世界大学生超级计算机竞赛评审委员会委员

2015 年 7 月 1 日

七

ASC 世界大学生超级计算机竞赛作为大学生超算竞技和交流的平台,吸引了越来越多国内外优秀的大学生参与到超算技术的学习和研究中来。学生们通过对比赛的长期准备,自发地学习到了许多超算的相关知识,从而促进了我国对超算人才的培养,推进了我国超算事业的发展。竞赛命题方向瞄准我国及世界尖端超算应用,既有理论拓展的空间,也有实践创新的机会,学生们可以充分发挥自己的潜能,参与到真实的超算项目中来,这也使参赛学生对高性能计算领域有了更深刻的了解和实践,更清楚自己应该学什么、怎么做。

李 明

太原理工大学数学学院院长

2015 年世界大学生超级计算机竞赛组织委员会主席

2015 年 7 月 10 日

八

高性能计算对国家的科学研究进步和产业发展创新起到越来越重要的技术推动作用。应用发展和人才培养是将超算硬件能力充分转化为技术推动力不可或缺的基石。浪潮希望通过 ASC 大学生超级计算机竞赛的举办推广,使更多青年人才能够提前学习了解并热爱超算科技,推动超算应用得到重视投入和长期健康发展,从而更好地发挥超级计算机的作用,使之更好地服务于各项科学研究和业务应用。

刘 军

浪潮集团高性能计算总经理

ASC 世界大学生超级计算机竞赛组织委员会委员

2015 年 6 月 15 日

前 言

超级计算是信息领域的前沿高新技术。随着信息化社会的飞速发展,人类对信息处理能力的要求越来越高,超级计算已在石油勘探、气象预报、航天国防、科学研究等领域得到广泛应用,金融、企业、基因、深度学习等新兴应用对超级计算的需求也在迅猛增长。目前,百亿亿次级别的超级计算机构建已提上研究日程,人们正期待采用超算解决更大规模和更复杂的挑战性问题。

2015年11月,由中国研发的天河二号在全球最快超级计算机排行榜 Top500 实现了六连冠,峰值计算性能达到 55Pflops。超级计算机在硬件技术方面不断取得突破,但适合大规模超算的应用以及专业人才的欠缺,已成为超级计算发展的瓶颈。应用是超算发展的根本动力,正是各行各业对计算不断增长的需求推动着超级计算技术的持续进步。当前提高并行规模仍然是提高性能的主要手段,能否发掘和提升应用的并行性,成为有效利用大规模超算系统的关键。而发掘和提升应用并行性,也要依赖超算人才对应用的理解和开发。我们在超级计算的普及传播、相关人才的培养储备等方面还有很多工作要做。

为了推进超算科技更好普及发展,特别是在高等学校中广泛传播超算知识,吸引大学生对其感兴趣并加以学习,浪潮集团联合亚洲超算协会于 2012 年举办中国大学生超级计算机竞赛、于 2013 年举办亚洲大学生超级计算机竞赛、于 2014 年开始每年举行 ASC 世界大学生超级计算机竞赛。目前 ASC 已成为与 ISC、SC 并列的具有全球影响力的大学生超级计算机竞赛,在世界范围获得了大学生们的关注和各高校的积极参与。但师生们普遍反映目前还缺少一本深入浅出的好书来系统地介绍如何参与超算竞赛并学习提高。

为了填补该空白,亚洲超算协会联合浪潮集团邀请国内外超算领域的专家、ASC 竞赛参赛队代表等联合撰写了本书。

关于本书

本书分为学习篇、竞赛篇和进阶篇三大内容。学习篇引领读者理解超级计算机及其系统架构、评测方法等,打好竞赛篇和进阶篇的理论基础。竞赛篇详细介绍了 ASC 世界大学生超算竞赛及规则,进阶篇介绍了 ASC 初赛和决赛精选赛题及其解析和优化方法等。

学习篇第 1 课介绍了超级计算机的意义、架构分类和目前发展遇到的挑战。第 2 课阐述了超级计算系统构建的一般原则,包括系统的组成结构、如何构建性能均衡的高效能计算系统,以及如何对它进行管理和调度。第 3 课介绍了超算系统的网络环境及在高性能计算中的应用。第 4 课介绍了如何构建超级计算应用运行环境,包括硬件环境、软件环境和开发环境,同时也介绍了异构系统的应用环境配置方法。第 5 课介绍了超级计算系统的综合评测体系,对高效能计算系统硬件性能的评测方法,对应用性能的评测方法和预测方法做了结构性的说明。

竞赛篇第 6 课和第 7 课介绍了世界三大超级计算赛事 ASC、ISC、SC,第 8 课以 ASC14 竞赛为例详细解析了 ASC 世界大学生超级计算机竞赛的规则。

进阶篇第 9 课详细介绍了参加竞赛的初赛方案如何组织撰写，最后五课则以实际举例分别介绍了如何针对 ASC 的赛题进行分析优化，以及如何针对众核优化赛题进行算法分析和并行优化。

本书全面介绍了大学生超级计算竞赛，结构清晰，条理清楚，对有志于学习和研究超级计算的高校大学生很有裨益。

致谢

本书编写组成员有吕文静、林新华、王渭巍、刘通、袁国兴、姚继峰、Wong Foo Lam、沈立、Jerry Chou、刘晓峰、文敏华、赫洁、金莲、刘羽、陈博文、张清、沈铂、王泽寰、张建峰、吉青等。

负责编写每课内容的作者分别是：第 1 课林新华，第 2 课王渭巍，第 3 课刘通，第 4 课张清、沈铂、王泽寰，第 5 课袁国兴、姚继峰、刘羽，第 6、7 课赫洁、吉青，第 8 课吕文静、金莲、陈博文，第 9 课 Wong Foo Lam，第 10、11 课沈立，第 12 课 Jerry Chou，第 13 课刘晓峰、张建峰，第 14 课文敏华。

对于本书的编写工作，各位作者付出了极大的心血和努力，将自己从业多年积累的超算知识和经验整理归纳，共同完成了此书。在此对所有的作者表示衷心的感谢！

《超算竞赛导引》编写组

目 录

序一

序二

序三

寄语

前言

第一篇 学 习 篇

第 1 课	超算的发展与应用	1
1.1	为什么要研究超算	1
1.2	超算的发展与架构分类	4
1.2.1	超级计算机的发展	4
1.2.2	超级计算机系统的架构分类	5
1.2.3	超级计算机的发展趋势	7
1.2.4	异构并行技术的兴起与发展	7
1.3	超算应用的现状和挑战	9
1.3.1	超级计算的应用现状	10
1.3.2	超级计算应用的挑战	12
第 2 课	超算集群系统的构建及功耗管理	14
2.1	超算集群系统的组成	14
2.2	超算集群功耗监控与管理	18
2.3	构建性能均衡的超算系统	20
2.3.1	性能均衡的超算系统	20
2.3.2	如何管理高效能计算系统	21
2.3.3	监控管理	22
2.3.4	作业调度	23
2.3.5	十万亿次级别高效能计算系统构建	24
第 3 课	超算系统的网络通信	29
3.1	InfiniBand 技术概要	29
3.2	InfiniBand 在 HPC 中的应用现状	31

3.3	InfiniBand 技术核心——RDMA	33
3.3.1	RDMA 的技术简介	33
3.3.2	RDMA 的技术核心	34
3.4	基于 InfiniBand 的 HPC 应用优化	36
3.5	InfiniBand 加速云计算与大数据	42
第 4 课	超算系统的应用环境	45
4.1	CPU 并行系统应用环境	46
4.1.1	硬件环境	46
4.1.2	软件环境	47
4.1.3	开发环境	48
4.2	CPU+MIC 异构并行系统应用环境	51
4.2.1	硬件环境	51
4.2.2	软件环境	51
4.2.3	开发环境	52
4.2.4	资源	54
4.3	CPU+GPU 异构并行系统应用环境	54
4.3.1	硬件环境	54
4.3.2	软件环境	54
4.3.3	开发环境	55
4.3.4	资源	58
第 5 课	超算系统性能评测方法	59
5.1	超算系统性能评测现状	59
5.1.1	系统性能测评的意义和挑战	59
5.1.2	系统性能测评的度量和主要内容	60
5.1.3	系统性能测评步骤及方法	62
5.2	高性能计算系统性能测评现状及方法	63
5.2.1	主要测试程序及应用	64
5.2.2	高性能计算系统排行榜	70
5.3	超算应用特征分析与监测工具	71
5.3.1	应用特征数据提取	71
5.3.2	应用特征分析——实验归纳	73
5.3.3	应用特征分析——理论预测	76
5.3.4	应用特征监控工具——天眼	78

第二篇 竞赛篇

第 6 课	世界大学生超级计算机竞赛介绍	83
-------	----------------------	----

6.1	ASC 世界大学生超级计算机竞赛	83
6.1.1	历史沿革	83
6.1.2	参赛团队	83
6.2	ISC 国际大学生超级计算机竞赛	88
6.2.1	历史沿革	88
6.2.2	参赛团队	89
6.2.3	特点及发展趋势	93
6.3	SC 国际大学生超级计算机竞赛	93
6.3.1	历史沿革	93
6.3.2	参赛团队	94
6.3.3	特点及发展趋势	96
第 7 课	ASC 世界大学生超级计算机竞赛历史与展望	97
7.1	2012 年中国大学生超级计算机竞赛	97
7.1.1	首次进入中国的国际赛事	97
7.1.2	大赛进展	97
7.1.3	大赛三要素：在校大学生、国际接轨、实用技能	97
7.2	2013 年亚洲大学生超级计算机竞赛（ASC13）	98
7.2.1	ASC13 定位及宗旨	98
7.2.2	大赛的赛制、赛程与组织架构	98
7.2.3	进入 ASC13 的十强	99
7.2.4	评审委员会对参赛高校的综合评价	99
7.3	2014 年世界大学生超级计算机竞赛（ASC14）	100
7.3.1	82 所世界高校同台竞技，创历史之最	100
7.3.2	全球超算专家云集	100
7.3.3	初赛：四大赛题遴选超算精英	101
7.3.4	决赛：七大赛题挑战智慧极限	102
7.3.5	黑马异军突起	102
7.3.6	e Prize 计算挑战奖激励全球超算天才	102
7.4	2015 年世界大学生超级计算机竞赛（ASC15）	103
7.4.1	六大洲 152 支参赛队，再创历史新高	103
7.4.2	全球超算专家云集	103
7.4.3	初赛：从分子到宇宙的“两极”考验	104
7.4.4	决赛赛题：紧贴应用	105
7.4.5	汇聚全球青年精英，挑战宇宙级“难题”	105
7.4.6	总决赛设 25 万大奖	105
7.4.7	《三体》作者刘慈欣“出现”在总决赛闭幕式	106
第 8 课	ASC 世界大学生超级计算机竞赛规则解析	107

8.1	详细解析 2014 年世界大学生超级计算机竞赛初赛规则	107
8.1.1	初赛介绍	107
8.1.2	初赛规则解析	107
8.1.3	初赛规则总结	109
8.2	详细解析 2014 年世界大学生超级计算机竞赛决赛规则	109
8.2.1	决赛介绍	109
8.2.2	决赛规则解析	109
8.2.3	决赛规则总结	112

第三篇 进 阶 篇

第 9 课	竞赛方案	113
9.1	简介	113
9.2	组队	113
9.3	基本条件	114
9.4	培训	114
9.5	硬件培训	115
9.6	竞赛方案	115
9.6.1	章节结构	115
9.6.2	图表的呈现	115
9.6.3	其他部分	116
9.6.4	高性能计算活动简介	116
9.6.5	硬件设计与能效	117
9.6.6	(HPL) LINPACK 测试	118
9.6.7	应用软件	118
9.6.8	配置文档	118
9.6.9	代码优化	121
9.7	项目书任务分配	121
9.8	总结	121
第 10 课	竞赛集群的设计与建设	123
10.1	集群的结构	123
10.2	集群的性能和功耗	124
第 11 课	系统浮点性能测试 HPL 优化方案	125
11.1	HPL 简介	125
11.2	安装与测试	126
11.3	常用的 HPL 优化方法	129
11.3.1	修改配置参数	129

11.3.2	第三方库的选择	130
11.3.3	编译器优化	130
11.3.4	确定进程与节点的对应关系	130
11.3.5	优化任务分配	131
11.3.6	散热	132
第 12 课	分子动力学软件 GROMACS 优化方案	133
12.1	应用程序简介	133
12.2	实验环境与硬件规格	133
12.3	测试参数调校	134
12.4	软件库的效能分析	135
12.5	计算资源分配的效能分析	137
12.6	应用 GPU 的效能测试	139
第 13 课	气候系统海洋模式 LICOM 优化方案	140
13.1	气候系统海洋模式 LICOM 介绍	140
13.1.1	什么是 LICOM	140
13.1.2	LICOM 的发展演化	140
13.1.3	LICOM 的优化意义	140
13.2	LICOM 优化的关键点分析	141
13.3	硬件环境搭建	141
13.4	软件环境配置	141
13.4.1	NetCDF C 库安装配置	142
13.4.2	NetCDF Fortran 库安装配置	142
13.4.3	LICOM 的安装配置	142
13.5	优化方案实施过程及结果	144
13.5.1	优化方案实施过程	144
13.5.2	优化结果	145
13.6	ASC14 备赛过程回顾	146
13.7	LICOM 于 ASC14 决赛赛场优化回顾	147
13.8	LICOM 优化总结及评估	149
第 14 课	三维纵横波分离的弹性波方程模拟方法 3D_EW 优化方案	150
14.1	组队参赛情况及参赛准备	150
14.1.1	人员准备	150
14.1.2	技术准备	150
14.2	ASC14 3D_EW 比赛规则解读	152
14.3	三维纵横波分离的弹性波方程模拟算法介绍	153
14.4	3D_EW 程序分析	154

14.4.1	热点分析	154
14.4.2	代码分析	154
14.5	基于 CPU+MIC 集群平台的 3D_EW 并行算法设计与实现	155
14.5.1	基于 CPU+MIC 集群平台的 3D_EW 并行算法整体设计	155
14.5.2	基于 CPU+MIC 集群平台的 3D_EW 并行算法实现	155
14.6	基于竞赛平台的 3D_EW 测试和分析	158
14.6.1	测试平台环境	158
14.6.2	单节点性能	159
14.6.3	跨节点性能	160
14.7	参赛感悟及收获	160

第一篇 学 习 篇

第 1 课 超算的发展与应用

1.1 为什么要研究超算

超级计算机，简单的理解是指世界上运行最快的计算机。为了获取高速度，超级计算机必定是高度并行的计算系统。早先，超级计算机的主要用户是国家级研究实验室、国防部门和大公司；而现在，超级计算机的使用正逐步走向普及。

应用的强烈需求，特别是有关人类社会文明进步的巨大挑战性问题求解的迫切需求和新的挑战性问题不断涌现，促使了超级计算机的不断发展。这是因为这些挑战性的问题求解，有的要求高时效性，如数值气象预报；有的要求高精度，如核爆模拟；有的要求高吞吐率，如海量的事务处理；更有的可能同时在时间、精度和吞吐率上都有要求等。

人们对计算机系统超过当前可能提供计算速度的需求总是不断增长的。需要很高计算速度的领域包括科学和工程问题的数值建模和模拟。这些问题常需要对大量数据进行很多次重复计算以得到有效结果。此外计算必须在合理的时间内完成。在制造业领域，工程计算和模拟如果可能的话必须在几秒或几分钟内完成。在设计环境中，如果进行一次模拟需要两星期才能得到结果，通常是不可接受的。因为只有模拟完成时间足够短，设计者才可高效地工作。当系统变得更复杂时，就需要增加更多时间对系统进行模拟。某些应用问题的计算对时间有指定限制（最著名的要数气象预报）。用两天时间以获取当地第二天精确的天气预报将使得这种预报毫无意义。某些研究领域，如对大型脱氧核糖核酸（deoxyribonucleic acid, DNA）结构建模和进行全球天气预报均属巨大挑战问题。巨大挑战问题是指无法用当今计算机在合理的时间内完成求解的那些问题。

以天气预报（数值气候预测）的应用问题的求解来讲，就是一个被广泛引用的、需要功能非常强大的计算机的一个典型例子。大气建模是通过将大气层划分成三维区域或单元完成的，该模型使用相当复杂的数学方程来估测各种影响。实质上，在某一时间间隔内的每个单元中的条件（温度、压力、湿度、风速和风向等）是通过使用在前一时间间隔中的条件进行计算得到的。每个单元的计算要重复许多次以模拟时间的推移。使此模拟具有有效性的关键点是必须有足够多的单元数。为预报数天的气候，由于大气层会受到远距离事件的影响，所以必须有较大的覆盖范围。假定将整个地球大气层分成大小为 $1\text{mi}^{\text{①}} \times 1\text{mi} \times 1\text{mi}$

① $1\text{mi}=1.609344\text{km}$ 。

的单元，布满从地面直至 10mi 的高度（即十个单元高）。粗略地估算将约有 5×10^8 个单元。假设每个单元的计算需要 200 次浮点操作，则在一个时间步中就必须完成 10^{11} 次浮点操作。如果欲预报 7 天以上的气候，时间间隔为 1min，此时就需要 10^4 个时间步和总计 10^{15} 次浮点运算。那么，对一个运算速度为 1Gflops (10^9 浮点操作/秒)的计算机就将需要 10^6 s，即超过 10 天才能完成上述计算。要想在 5min 内完成这一计算，就需要运算速度为 3.4Tflops (3.4×10^{12} 浮点操作/秒)的计算机。如果将分割单元缩小成 0.1mi $\times$ 0.1mi $\times$ 0.1mi 的大小，则总的浮点运算的次数将增至 10^{19} 次，若同样想在 5min 内完成这一计算，就需要运算速度为 34Pflops (3.4×10^{15} 浮点操作/秒)的计算机。另一个需要巨大计算量的应用问题是预测空间中宇航物体的运动。每个物体由于万有引力而互相吸引，这些远距离的力可用简单公式加以计算，而每个物体的运动可以通过计算物体所受总力的计算加以预测。如果共有 N 个物体，则每个物体将总共需计算 $N-1$ 个力，或约 N^2 计算。在确定这些物体的新位置后，必须重复这种计算。例如，银河系可能有 10^{11} 个星体，此时就需重复计算 10^{12} 次。即使使用一个仅需 $N \log_2 N$ 次计算的高效的近似算法（但每次计算中含有更多的计算量），计算的总量仍异常巨大 ($10^{11} \log_2 10^{11}$)。若在单处理机系统上运算将需很长时间。若每次计算仅需 $1\mu\text{s}$ (10^{-6} s，这是非常乐观的估计，因为一次计算中含有多次乘法和除法)，则使用 N^2 算法时，每次迭代就需 10^9 年，而用 $N \log_2 N$ 算法时，一次迭代也需几乎一年的时间。

全球的天气预报和大量物体的模拟是需要巨大计算的传统应用例子，但是人类会不断想象那些超过计算机系统能力的新的应用，从而需要比目前可提供的更高的速度。例如，虚拟现实，就需要很高的计算速度以实时地获得所需结果。看来不论当前的处理机已达到什么样的计算速度，总是会有新的应用需要更高的计算能力。

美国联邦政府于 1992 年启动高性能计算和通信的一个五年计划，首次提出了迫切要求求解的十二个巨大挑战性应用问题，包括流体和结构集成式的机体模拟、气候建模、流体湍流、污染的弥散、人类基因组、海洋环流、药物设计、量子染色动力学、半导体建模、超级计算机建模、燃烧系统、视觉和认知。其中有些问题已取得了阶段性成果，如人类基因组基因排序的测试，有的则仍需继续。随着时间推进和社会环境的变化，又不断有新的挑战性应用问题出现。进入 21 世纪后，较为公认的有以下 11 个巨大挑战性应用问题：气候预测；交通业中的对汽车、飞机和轮船的设计改进，涉及空气流体动力学、燃料消耗、结构设计和防撞等；生物信息学和计算生物学；社会健康和安全问题；地震预测、地球物理探测和地球科学，如石油勘测；天体物理学，涉及天体演变模拟；材料科学与计算纳米技术；人类及组织系统的研究，涉及大量人口的行为模拟和防恐；核爆模拟；数值风洞；污染检测和防范等。表 1-1 列出了 1992 年和当今巨大挑战性应用问题的对照。

表 1-1 巨大挑战性应用问题示例对照

20 世纪 90 年代	21 世纪	备 注
流体和结构集成式的机体模拟	气候预测	
气候建模	交通业	对汽车、飞机和轮船的设计改进，涉及空气流 体动力学、燃料消耗、结构设计和防撞等
流体湍流	生物信息学和计算生物学	

续表

20 世纪 90 年代	21 世纪	备 注
污染的弥散	地震预测	
人类基因组	社会健康和安全问题	
海洋环流	地球物理探测和地球科学, 如石油勘测	
药物设计	天体物理学, 涉及天体演变模拟	
量子染色动力学	材料科学与计算纳米技术	
半导体建模	人类及组织系统的研究	涉及大量人口的行为模拟和防恐
超级计算机建模	核爆模拟	
燃烧系统	数值风洞	
视觉和认知	污染检测和防范	

在超级计算机系统中, 增加计算速度的一种方法是用多个处理机协同求解一个问题, 这种方法实际上多年来一直在进行研究。在这种方法中, 整个求解问题被分成若干部分, 然后每个部分各由一个处理机并行计算。编写这种形式的程序称为并行编程。计算平台, 即一台并行机, 可以是专门设计的、含有多个处理机的计算机或以某种方式互连的若干台独立的计算机。该方法能提供显著的性能增加。基本的想法是, n 台计算机应能提供 n 倍的单机速度, 不论当前计算机的速度为多少, 可以期待求解问题将以 $1/n$ 时间完成。当然这是一个理想情况, 实际上很难达到。这是因为求解问题经常不能很好地被分解为各个独立部分, 各部分之间必须进行交互, 包括计算中的数据传送和同步。尽管如此, 依赖于欲求解问题和问题中的并行性程度, 仍可达到实质性的性能改进。对并行计算的要求永远不会终止, 这一方面是因为单处理机执行速度的不断升级使并行机速度越来越快, 另一方面是因为总是有一些巨大挑战性的求解问题, 它们在当前计算机上求解时不可能在合理的时间内完成。

除了可使现有的求解问题得到加速的求解, 多计算机/多处理机的使用常常可使一个更大的求解问题或更精确的求解问题能在适当的时间内完成。例如, 许多物理现象的计算, 意味着将求解问题分成离散的求解点。气象预报计算包含着将天空分为三维的求解格点。二维和三维的求解格点出现在许多其他应用中。用多处理机求解时, 常允许在给定的时间内计算更多的求解点, 从而可得到更精确的求解。一个相关的因素是多计算机常比单机有更大的主存容量, 从而可使需要较大主存容量的求解问题得到解决。

超级计算已成为与科学理论和物理实验并驾齐驱的当代人类从事自然界新发现和新探索的第三种手段, 而且这种手段正在变得越来越重要。

经典科学和当代科学的差异: 经典科学基于观察、理论、假设和物理实验, 而当代科学则以观察、理论、假设、物理实验和数值模拟来检验所提出的理论。两者的差异如图 1-1 所示。可以看到, 超级计算机利用数值模拟的手段来代替实际的物理实验完成对理论和假设的验证。这是一种更事半功倍的方法。

在论述了对超级计算机的迫切需求之后, 下面转向讨论研究超级计算机的主要目的。综合起来, 研究超级计算的目的主要有以下三点。

- (1) 促进计算机科学与技术不断发展的强大动力, 特别是对计算机并行体系结构, 高

带宽、大容量的层次存储体系，高带宽、低时延的互连技术，并程序程序设计技术，芯片技术等有着深远的影响。

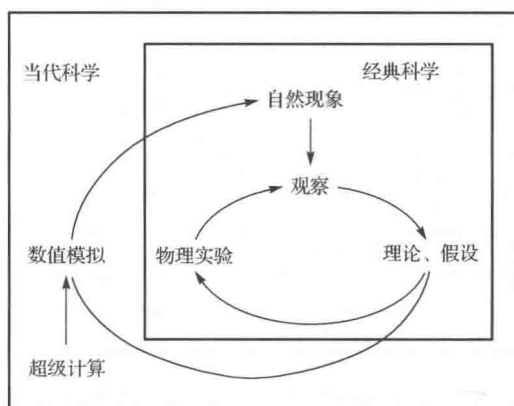


图 1-1 当代科学和经典科学研究手段的差异

(2) 对其他的交叉学科起着巨大的推进作用。相应地产生了计算数学、计算物理、计算化学、计算生物、计算经济学、计算电磁学、计算工程、计算财政、计算流体力学、计算地球物理、计算力学、计算统计学、计算宇宙学等。

(3) 超级计算机的水平反映了一个国家的经济和科技的综合实力。

为了反映当前世界上超级计算机的发展水平，自 1993 年起，每年的 6 月和 11 月分别在德国和美国召开的世界超级计算机会议上，发布世界上超级计算机的 500 强 (Top500) 排行榜，通过测试执行基准程序 LINPACK (线性方程组求解) 所需时间的长短来排行。

长期以来，一直是美国的超级计算机独霸天下，独占鳌头。

进入 21 世纪后，日本的超级计算机技术有了长足的进步，它所研制的“地球模拟器”超级计算机的性能高达 35.6 Tflops，在 2003~2005 年的两年半年时间中连占榜首。以后几年又被美国的超级计算机赶超。到了 2011 年，日本 NEC 公司研制的“京”超级计算机，性能为 8.1624 Pflops，再次占据 Top500 榜首。

在 2014 年 6 月的 Top500 中，我国的超级计算机共有 76 台，总数排名第二。美国的总数为 233 台，仍占据霸主地位。从全球看，超级计算机领域的竞争十分激烈，过几年就有新的超级计算机投入使用，Top500 中的排名变化很快，它是各国尤其是大国必争的战略制高点，是一个国家综合实力与技术实力的主要体现。

需要注意的是，Top500 排行榜，对超级计算机的体系架构的分类是比较简化的，只将超级计算机分为星形、向量、大规模并行处理 (massively parallel processing, MPP) 系统、集群这几种。

1.2 超算的发展与架构分类

1.2.1 超级计算机的发展

超级计算机 (supercomputer) 这一术语，最早出现于 20 世纪 70 年代初期。

第一代的超级计算机是单指令多数据流（single instruction multiple data, SIMD）的阵列处理机。美国于1972年研制成功的ILLIAC-IV阵列处理机，由64个处理器构成8×8阵列，由分立元件组成，运算速度每秒近1亿次浮点运算。

第二代的超级计算机是具有流水结构的向量机。美国Cray公司于1976年研制成功的Cray-1是典型代表，它由12个不同功能的流水运算部件构成，分为4组，可并行流水工作，峰值运算速度达到每秒1.6亿次浮点运算。Cray-1使超级计算机首次进入商品市场。

向量机系统结构按向量操作对象及结果主要存放在寄存器中还是存储器中，可分为存储器-存储器工作方式向量机和寄存器-寄存器工作方式向量机两大类。前者的工作特点是向量操作的源向量都取自主存且操作生成的结果向量也存放到主存中，而后者则都取自或存放到向量寄存器中。早期的向量机都是存储器-存储器方式的，如TI公司的ASC（1972年）、CDC公司的STAR-100（1973年）以及后来CDC公司的CYBER-205（1980年）和ETA-10（1986年）。1976年Cray公司研制的Cray-1向量机首次采用了寄存器-寄存器工作方式，由于它在短向量操作时显示出良好的性能，再加上指令系统的简洁性（如同RISC机，面向寄存器），所以逐步成为向量机的主流。

第三代的超级计算机是具有多指令多数据流（multiple instruction multiple data, MIMD）形式的共享主存多处理机系统。1976年问世的超级计算机Cray-1是单向量机系统，之后为了进一步提高向量机的性能，就在系统中不断增加向量部件的数量，即采用并行向量或多向量部件的技术。1982年推出的Cray X-MP/2有两个向量处理部件，1984年生产的Cray X-MP/4有4个，1988年推出的Cray Y-MP 816最多可有8个向量部件，而此后出现的C90则有16个。进入20世纪90年代后，向量部件数量已达到数百个，而到21世纪时，向量部件数量已达到数千个。

为了使向量和标量处理的性能平衡，当今的并行向量处理机多采用高性能的超标量处理器，具有多路发射指令的能力，如日本于21世纪初，由NEC公司研制的“地球模拟器”（Earth Simulator）并行向量处理机由640个节点处理器组成，每个节点中含有以下部件：8个向量处理器，1个16GB的共享存储器，1个远程访问控制部件，1个I/O处理器。地球模拟器并行向量处理机系统的峰值性能为40.96Tflops。在Top500排行榜的榜首位置上，独领风骚了近3年时间（2002~2005年）。

第四代的超级计算机是MPP系统。它由多至上万个处理机组成，凭借高度并行来获取超高性能。可以用小节点、非一致存储访问结构（non-Uniform memory access, NUMA）、超节点或向量加超标量的方法来构成MPP系统，在当今的Top500排行榜中，MPP系统仍有相当的比例。

第五代的超级计算机是集群系统，这是当今最流行的超级计算机。早先的集群系统采用同构方式，当今的集群逐步转向采用异构方式。主要由CPU+GPU（图形处理器）或CPU+MIC（many integrated core）组成。

1.2.2 超级计算机系统的架构分类

由于超级计算机系统是高度并行的计算机系统，所以完全可以按并行计算机系统架构的分类方法加以分类。

可以按并行计算方式是SIMD还是MIMD，以及存储器是共享的还是分布的这两个准

则加以分类,一般可分为以下四类。

- (1) 共享存储-SIMD (SM-SIMD)。
- (2) 分布存储-SIMD (DM-SIMD)。
- (3) 共享存储-MIMD (SM-MIMD)。
- (4) 分布存储-MIMD (DM-MIMD)。

早期的超级计算机系统以 SIMD 方式工作,由于系统中的各个处理器按阵列方式排列,所以又称为阵列处理机。阵列处理机中的存储器可以是共享式的 (SM-SIMD),也可以是分布式的 (DM-SIMD)。阵列处理机的专用性较强,一般只适合于求解某类算法,工作效率往往很高。

单向量机系统中只有一套向量部件,但存储器为向量部件、标量浮点部件和标量整数部件所共享,因此属于 SM-SIMD 类型。它有较强的通用性,特别是在求解向量应用问题时,有很高的效率。

当今的超级计算机系统大多以 MIMD 方式工作。多向量机 (multi vector processor, MVP) 系统中有多套向量部件,但存储器是共享的,因此属于 SM-MIMD 类型。对称多处理器 (symmetric multiprocessor, SMP) 系统也属于这一类型。MVP 和 SMP 又称为 UMA (uniform memory access) 系统,因为系统中所有处理器对任何存储单元有相同的访问时间。

与 UMA 系统相对的系统称为 NUMA 系统,在 NUMA 系统中,存储器是分布的,各个处理器对同一存储单元的访问时间可能是不同的,依赖于处理器在系统中所处的具体物理位置。显然,NUMA 系统属于 DM-MIMD 类型。应注意的是,NUMA 系统中的处理器可对远程存储器 (即非本地存储器) 以 load-store 指令形式进行直接访问,因此该系统有一个统一的存储器逻辑地址空间。NUMA 并行机系统按是否对 Cache 一致性提供硬件支持可进一步分为 CC-NUMA (cache-coherent) 和 NCC-NUMA (non-cache-coherent)。当存储器全由 Cache 组成时就变为 COMA (cache only memory architecture) 系统。

如果并行计算机系统处理器必须以消息传递方式访问远程存储器,就称为 NORMA (no remote memory access) 系统,它也是 DM-MIMD 类型。与 NUMA 系统不同,它有多个存储器地址空间,且系统中的每个处理器是一个独立的计算机。NORMA 系统按计算机间的互连紧密程度,又分为紧耦合和松耦合两种。集群 (cluster) 系统是松耦合的典型代表,而 MPP 系统则是紧耦合的典型代表。

MPP 系统使用大量的商品化处理节点,用定制的高带宽、低时延互连网络将它们连接起来,存储器在物理上是分布的,必须通过消息传递实现进程间的相互通信,是紧耦合的并行机系统,具有良好的可扩展性。Cray T3E 和 IBM Blue/Gene 系统是它的典型代表。

集群系统中每个节点是一个完整的计算机,可能没有某些外设,节点也可以是一台 SMP 或个人计算机 (personal computer, PC) 等。它采用商品化的互连网设备,节点机通过 I/O 总线与网络接口相连。每个节点机上留驻有一个完整的操作系统,并有一个附加的中间件以支持单一系统映像和高可用性。

图 1-2 中示出了超级计算机系统的架构分类情况。

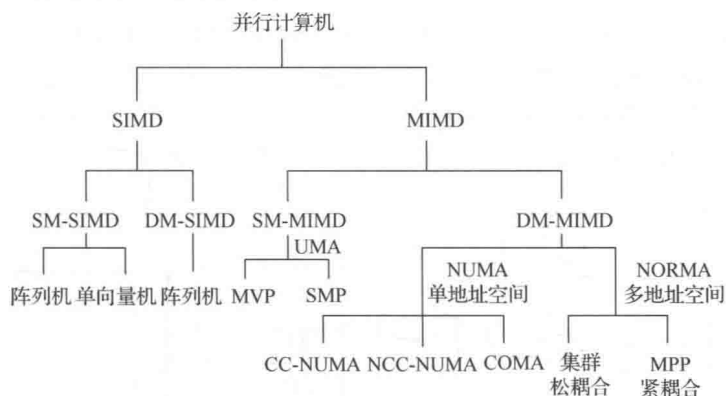


图 1-2 超级计算机系统的架构分类

1.2.3 超级计算机的发展趋势

进入 21 世纪后，由于具有挑战性的应用问题，如气候建模、精密的气象预报、量子染色动力学、海洋环流、污染弥散、模拟核爆炸以及认知和视觉等，要求计算机系统能提供每秒千万亿次（P 级， 10^{15} ）乃至百万万亿次（E 级， 10^{18} ）以上的计算能力。目前 P 级的超级计算机已比较成熟，主要采用 CPU+GPU 或 CPU+MIC 的异构计算架构。E 级的超级计算机的研制则正在激烈的竞争中进行。

打造 E 级的超级计算机有一定的难度，其原因是为了达到 E 级的速度，必须采用更大规模的附属处理器数，这就会导致整个系统功耗的急剧增加，因此研制具有低功耗、高性能的附属处理器就成为一个关键的突破口。此外，更高性能的单处理器，高带宽、低时延的互联网，异构计算架构等也都是需要采用的技术。

最后应指出的是，每秒百万万亿次的超级计算机系统的研制和开发，必须要有相应的优化编译器和并行编程语言的支持，这两者无论对提高系统的性能还是方便用户的使用都具有重要的意义。

1.2.4 异构并行技术的兴起与发展

异构并行技术的研究始于 20 世纪 80 年代。这是一种能使计算任务的并行性类型（代码类型）与机器能有效支持的计算类型（机器能力）最匹配、最能充分利用各种计算机资源的并行技术。

一个异构计算系统通常由以下三部分组成：①一组异构机器，如向量机、MIMD 机、集群、图形处理机等；②将各种异构机互连起来的高速网，可以是商品化网络，也可以是用户定制的网络；③相应的异构计算的支撑软件。

异构并行计算的基本工作原理是，首先析取求解任务的并行性类型，其次将具有相同类型的代码段划分到同一个子任务中，然后根据不同的并行性类型将各子任务分配到最适合执行它的计算资源上加以执行，达到使计算任务总的执行时间为最小的目的。

图 1-3(a)给出了一种分布存储、松耦合的异构计算系统。它由向量机、MIMD 机、SIMD 机和工作站组成。下面举例说明异构并行机是如何获取任务计算的加速比的。假设一个计算任务中的向量、MIMD、SIMD 和 SISD（顺序操作，由工作站执行）各类子任务所占的

比例分别为 30%、36%、24%和 10%，且向量机、MIMD 机、SIMD 机和工作站在执行与之匹配的子任务时的加速比分别为 30、36、24 和 10，TS 为该任务在基准计算机上顺序执行所需的时间，则在上述假设的异构机上执行相同的任务所需的时间 $T_{\text{异}} = (0.3TS/30) + (0.36TS/36) + (0.24TS/24) + (0.1TS/10) + T_c$ ，其中 T_c 为机器间通信时间，假设需要 $0.02TS$ ，则 $T_{\text{异}} = 0.06TS$ ，加速比可到达 16.67。

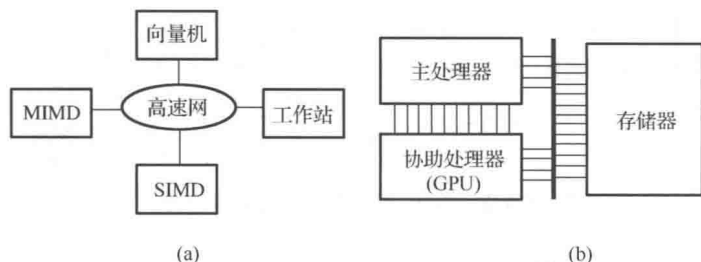


图 1-3 松耦合和紧耦合的异构计算系统

异构计算的应用范围很广，几乎所有涉及巨大挑战性应用问题的求解都可以用异构计算进行经济有效的求解。典型的应用包括图像理解、质点示踪、声束形成、气候建模、湍流对流混合模拟和多媒体查询等，因为这些应用中通常都含有多种不同的计算类型的需求，较适合用异构计算进行求解。一个实际的例子是美国 Hughes 研究所和麻省理工学院(MIT)共同研制的图像理解系统，它按图像理解层次的要求设计成三层，低层是 SIMD 位串网络(4096 位)，用来进行像素操作，中层由 64 个数字信号处理(digital signal processing, DSP)芯片构成，以 MIMD(中粒度)方式执行模式分类等操作，顶层是一个通用 MIMD(粗粒度)机器，完成场景和动作分析等知识处理操作。

将异构并行计算架构引入超级计算机中，采用专用附属处理器的方式并在芯片级加以实现是在进入 21 世纪之后。其主要原因是，自 1996 年以来，由于挑战性应用问题的急切需求，以及超大规模集成电路技术和网络技术的迅速发展，加快了计算机系统结构的演变和发展进程。虽然开发指令级并行性的超标量技术日臻完善，动态猜测执行、显性并行指令计算(explicitly parallel instruction computing, EPIC)等方法也已成功应用到商品化产品中，但随之而来的是超标量处理器的设计越来越复杂，以至于进一步开发指令级并行性已变得相当困难。

另一方面，为了提高性能，微处理器芯片的时钟频率越来越高，导致功耗的急剧增加和组件装配密度的下降。显然，单纯依赖提升单处理器时钟频率和一味开发指令级并行性以提高计算机性能的方法已不再经济有效。于是多线程、多核技术应运而生，实践证明在开发指令级并行性的同时，必须加强对线程级和数据级并行性的开发，唯此，才能使计算机的性能有更大幅度的提高。这种演变的实质表明并行计算技术已成为当今计算机系统结构发展的主流技术。

随着通用微处理器变得越来越复杂和功耗越来越大，长期以来人们已有察觉，即在通用微处理器上倾注如此多的精力可能是低效的。相反，专用的处理器可能功能更强大且节省空间，因此尽可能地卸载更多的工作到专用的附属处理器就显得很有意义。使用附属处理器并不是新事物，微处理器很早就卸载一部分工作给浮点协同处理器。现代的“附属处理器”术语暗示两个处理器间的不对称性质，表明附属处理器代表主处理器完成工作。

用图形处理部件(graphic processing unit, GPU)完成通用计算的概念已非常流行(这

种 GPU 称为通用 GPU, 即 GPGPU), 这是因为 GPU 具有很高的性能和性能价格比。例如, 与同类的高性能微处理器相比, GPU 有大约高出一个数量级的浮点 CPU 能力和存储器带宽。此外, GPU 性能的改进速率远比通用微处理器性能的改进要快, 大约每 6 个月性能就增加一倍, 显著地快于通用微处理器的改进, 由于 GPU 的可编程性正在不断增长, 所以作为通用计算的前景非常广阔。

GPU 之所以能提供如此好的性能有以下几个原因。从经济上讲, GPU 的性能需求是由大型视频游戏工业驱动的。从技术上讲, GPU 是专门处理图形绘制的, 是一种具有大量数据并行的密集计算应用。为此 GPU 能很容易地通过增加更多的处理核来改进浮点性能。此外, GPU 可以极大地忽略许多通用处理器必须处理的问题, 包括为顺序代码保持流水线的畅通等。这样 GPU 就节省了从事转移预测、高速缓存和指令调度工作的大量的器件资源。早期的 GPU 只关注对图形学的支持, 这种系统由固定功能的图形流水线硬件实现。这些早期的 GPU 只支持单精度浮点操作, 没有双精度浮点和整数操作, 也没有位处理操作。但是随着时间的变迁, 由于游戏开发者寻找以不同方式使用硬件, 硬件的可编程性越来越高, 其结果是 GPU 趋向于支持更通用的计算。目前的 GPGPU 均支持双精度浮点和整数操作。

一个现代的 GPU 犹如一个具有片内动态随机存取存储器 (dynamic random access memory, DRAM) 的大规模细粒度多核芯片。例如, NVIDIA GeForce 8800 有 8 个核, 每个核由 16 个 SIMD 处理器组成, 这表明这些处理器以锁步方式运行。

虽然硬件的实现通常是 SIMD, 但程序设计模型并非如此, 就计算统一部件体系结构 (compute unified device architecture, CUDA) 语言而言, 它几乎是 C 程序设计语言的完整子集。不管如何, GPU 的能力在于它能完成大规模并行浮点计算。和其他的多核芯片一样, 一个标准的范例是, 使用多处理器对输入的数据块进行尽可能多的处理, 然后再装载新的数据块。图 1-3(b)给出了这种共享存储、紧耦合的异构计算系统。

MIC 是英特尔 (Intel) 公司开发出的最新并行处理架构, 主要针对高性能计算、工作站和数据中心高度并行处理的应用领域, 推动百亿亿次超算计算技术革新。与传统的处理器相比, Intel MIC 具有更小的内核和硬件线程, 以及更宽的向量单元, 因此能更有效地提升整体性能、满足高度并行化的高性能计算应用。此外, MIC 还具有较强的兼容性, 能够通过使用标准的 C、C++ 和 FORTRAN 源代码进行编程。另外, 由于 MIC 是使用与 Intel 至强处理器相同的工具、编译器和库的全新架构, 而全球超过 80% 的超级计算机采用了 Intel 公司的处理器平台, 所以研发人员在开发针对 MIC 平台的软件时, 能够继续在一个较为熟悉的领域开展研发工作, 较好地解决了重新编程等技术问题。

在 2014 年 6 月举行的国际超级计算大会上所公布的 Top500 中, 有 62 个系统使用了某种形式的附属处理器或协处理器的异构并行计算系统, 其中 44 个采用了 NVIDIA 的 GPU, 17 个采用了 Intel 的 Xeon Phi。

1.3 超算应用的现状和挑战

超级计算机的应用好坏在很大程度上取决于并行编程的好坏, 但并行编程比通常的顺序编程要难得多。其困难主要在于用户 (程序员) 在并行编程时将面临一系列的选择, 而

在顺序编程时几乎不需要进行什么选择。

在并行编程过程中,用户将面临如下的一些选择:采用何种的算法范例来描述并行编程;采用何种并行编程模型;使用哪种并行编程语言;选用何种架构的超级计算机平台等。在当今的超级计算机中,并行性已不对用户隐蔽,软件开发者几十年来享受稳定性能改进的“免费午餐”的时代已经结束,不利用多核芯片的程序将无法实现性能的改进。

1.3.1 超级计算的应用现状

超级计算机在经历了 40 余年的发展后,在并行编程方法和技术等方面取得了不少成果。下面分别加以叙述。

1) 已归纳出 5 种并行程序设计算法范例,供用户选择

如前面所述,并行编程是一项复杂的任务,用户首先面临的将是挑选什么样的并行算法来描述求解任务。如同撰写作文一样,如果能有并行算法范例可供模仿和参照,则无疑可收到事半功倍的效果。值得庆幸的是,虽然并行算法的发展历史比顺序算法要短,但已有一些范例可供用户使用。

(1) 阶段并行算法范例。

在该算法中,并程序由一些超步组成,每一超步分为两个阶段:计算阶段和交互阶段。在计算阶段,多个进程中的每一个完成各自的独立计算,此后是交互阶段,在交互阶段中,这些进程相互间完成一个或多个交互操作。然后便执行下一个超步。图 1-4(a)中对此范例进行了说明。

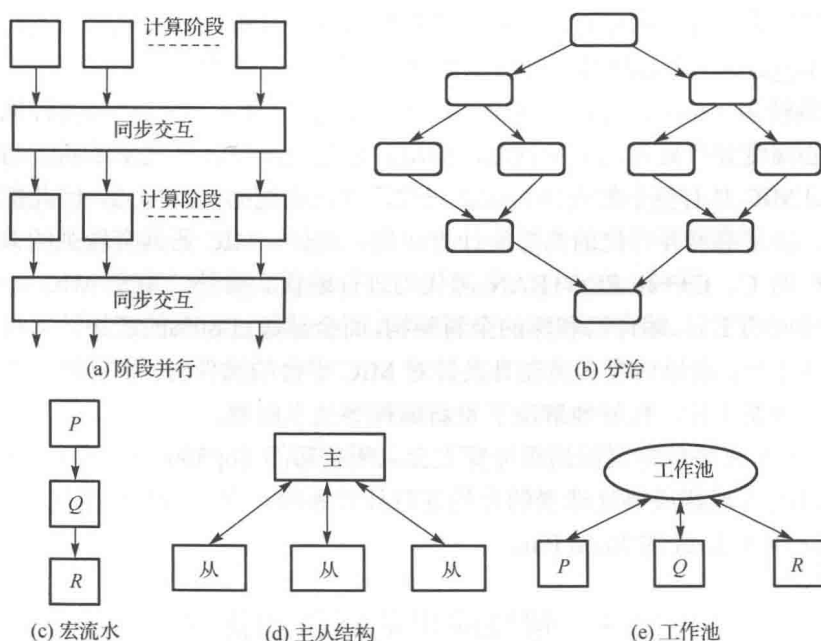


图 1-4 各种并行算法范例

阶段并行算法的主要优点是便于进行并行算法的性能分析,这是因为在计算阶段和交

互阶段互相分离的缘故。其主要缺点一是计算与交互不能进行重叠操作，二是在进程间很难维持工作负载的平衡。

阶段并行算法中有两个特例：同步迭代和异步迭代。在同步迭代中，各个超步成为同一循环中的一串迭代。采用了障栅同步，仅当所有进程均完成了第 j 次迭代后，才能开始循环中的第 $j+1$ 次迭代。异步迭代与同步迭代相反，由于不采用障栅同步，所以允许一个进程在完成本次迭代后马上开始下一次迭代而不用等待其余进程。用户需注意的是异步迭代可能会出现不确定的情况，即存在不同进程间跨迭代的数据相关性。

(2) 分治并行算法范例。

分治并行算法范例非常类似于分治顺序算法范例，即把欲求解的问题分解成若干个子问题，若子问题规模仍过大，则可将子问题再进一步分解。通常这一分解过程一直要进行到所求解的子问题成为最基本的、不能再分的子问题为止。此后将这些子问题分派到各节点机，并行计算各子问题。在求得各子问题的结果后，还需逐级将它们汇总起来，得到对问题的最终求解结果。图 1-4(b)中对此范例进行了说明。

(3) 宏流水并行算法范例。

这种算法范例是建立在功能划分的基础上的。首先需将求解的问题像流水线那样按功能划分成若干子任务，然后使每个节点机执行其中的一个子任务(即流水线中的级或阶段)。这种范例的通信模式非常简单，因为数据只在流水线中的相邻阶段间传送。为了区别于通常细粒度的指令流水线，常称其为宏流水方式。图 1-4(c)中简示了该范例。

(4) 主从并行算法范例。

它由主进程执行并行程序中的基本顺序部分，并负责派生出一些从进程去并行执行各自的工作负载。当一个从进程完成主进程所分配的任务后便通知主进程，让其分派一个新的工作负载。主进程还需承担汇总来自各从进程计算结果的工作。整个程序的并行执行由主进程完成协调，因此是一种很简单的并行算法范例，缺点是主进程很可能成为算法求解性能的瓶颈。应注意的是，这种范例通常使用静态负载平衡方法，即子任务的分配在计算的初始阶段完成，分配可以是一次完成或循环进行。还应注意的是，在一般情况下，只有主从进程间有消息通信。图 1-4(d)中对此范例进行了说明。

(5) 工作池并行算法范例。

这种范例常用于共享变量模型，它以全局数据结构方式加以实现。开始时工作池中可能只有一件工作，而事先所创建的一些进程中的任何空闲进程都可从该工作池中获取一件工作加以执行，执行后可能产生 0 个、1 个或多个新的工作并放到工作池中。当工作池为空，且任一进程均不再生成新的工作时，程序执行便终止。这种范例容易实现负载平衡，因为工作负载是动态分配给各空闲进程的。其缺点是使众多进程高效地访问工作池的具体实现较为困难。工作池通常可以由一个无序集、一个队列或一个优先级队列加以组成。图 1-4(e)示明了此范例。

2) 已开发了各种类型的并行编程语言可供用户选用

— (1) 面向共享存储器的线程编程语言，如 POSIX Threads、Java Threads、Open MP 等。在标准的顺序语言中调用线程库中的各种例程。

(2) 面向分布存储器的消息传递编程语言, 如消息传递接口 (message passing interface, MPI)、并行虚拟机 (parallel virtual machine, PVM) 等。在标准的顺序语言中调用各种消息传递的库例程。

(3) 开发数据并行的并行编程语言, 如 HPF (high performance fortran)。

(4) 面向异构并行架构的编程语言, 如 NVIDIA 的 CUDA 编程语言。

3) 已提出 3 种基本的并行编程模型供用户选用

(1) 共享变量。

(2) 消息传递。

(3) 数据并行。

总的来讲, 随着超级计算机的迅猛发展, 随着并行编程计算的发展、改进和完善, 超级计算机应用的范围已越来越广, 用户日益增多, 已深入人类社会和国民经济的各个领域。在不少研究领域中已取得了阶段成果或关键性成果, 如人类基因组排序的测定等。并行编程技术的不断发展和改进, 方便了用户对超级计算机的使用, 增强了用户使用超级计算机的信心。

1.3.2 超级计算应用的挑战

在超级计算应用中, 应注意开展以下几个挑战性的问题的研究。

1) 具有规模可扩展性和性能可移植性的并行程程序的编写方法

性能不仅和应用相关, 也和硬件相关。特别是当处理器数量增加时, 会越来越难保持相同级别的加速比。但是多核芯片上核的数量似乎增长得很快, 基本上每 18 个月能增长一倍。因此任何能被长期使用的软件, 都是成功的软件。一旦写好一个可扩展的软件, 它就能在没有程序员特别的干预下, 跟随硬件的发展, 就像串行程序在那个 CPU 主频快速增长的年代所做的一样。除了可扩展性, 可移植性对于长期使用的软件也是至关重要的, 这是由于并行计算机硬件的差异性造成的。这里要特别强调性能可移植性这个概念的重要性, 它是指一个并程序在多种不同类型的并行计算机上, 只需要通过一些适当的调谐, 就能获得较好性能的能力。性能可移植性通常可以通过在逼真的抽象机模型上设计算法来获得, 因此选择一个逼真的抽象机模型至关重要。因此, 仅使程序能运行在不同的并行平台上是远远不够的, 程序还必须能很好地运行在这些平台上。在进行编程时, 需要从设计的一开始就考虑可扩展性和性能可移植性, 按此原则设计的并程序将会有很长的生命周期, 这是对并程序中投入的巨大智力和经济投资的合理举措。

2) 在并行编程中如何自觉使用并行程序设计方法学

在并行编程中自觉使用方法学来指导编程, 能起到事半功倍的效果。目前, 一些行之有效的方法包括增量式改进; 侧重并行结构; 以小的存储空间换取性能的较大改进, 如使用 Cache 行的垫衬以消除假共享, 将共享变量变为私有变量, 设置重叠区域等; 提升求解问题的空间维, 如将一维提升为二维、二维提升为三维等, 其目的是增加并行性和减少并程序中通信和同步的需求; 选择适当的数据块分配方式, 如静态和动态、块分配、循环分配、块循环分配等; 注重对并行结构的测试; 尽可能多地编写顺序代码; 乐意书写附加代码等。

3) 并行编程时尽量遵循并行程序设计原理

(1) 为什么要突出设计原理? 这是因为并行的硬件基础在不断变化, 并行的软件设计环境也在不断变化, 但原理永远不会过时。

(2) 应该遵循的设计原理, 包括并行可扩展原理和性能可移植原理; 并行抽象原理; 静、动态数据结构分配原理; 拥有者计算原理; 存储空间换性能原理; 求解问题空间提升原理等。

要真正遵循这些原理来进行并行编程是具有很大的挑战性的。

4) 优化技术的使用

包括多层次存储体系的有效使用、高速缓存的有效使用、汇编级指令的优化技术等。

5) 学科交叉的合作应用

超级计算机的应用因为涉及计算机学科和具体应用的其他学科, 所以只有双方的密切配合, 才能使超级计算机的应用落到实处, 取得最佳的应用效果。

最后, 需要提及的是, 随着超级计算技术的不断发展以及应用的日益普及和细化, 一门称为“计算科学”(computational science)的学科正在兴起。它将会是一门科学, 而非技术。这一学科随着超级计算的日益普及, 正逐步趋向成熟, 并会以与其他学科交叉的方式大力发展计算科学和超级计算技术。开展对计算科学和超级计算技术的研究和应用, 将对增强我国在科技领域的竞争力和提高自主的创新能力起到巨大的促进作用。

第2课 超算集群系统的构建及功耗管理

2.1 超算集群系统的组成

当前 HPC 的主要架构包括集群和 MPP 两种，从 2014 年 6 月份的 HPC Top500 排行榜来看，其中集群架构的计算系统为 427 套，占总量的 85.4%，MPP 架构的系统为 73 套，占总量的 14.6%。集群架构目前已经成为最广泛的超级计算机架构。

典型的 HPC 集群系统主要由五类计算（或网络）设备和三类网络组成。

五类设备主要指管理节点及登录节点、计算节点（包括瘦节点和胖节点）、交换设备、I/O 节点和存储设备。另外，当前很多高性能服务器都采用 CPU+加速处理器异构的方式，因此有些高效能集群系统还包括加速节点。

1) 登录节点

登录节点相当于用户访问集群系统的网关。用户通常登录到这个节点上编译并提交作业。

登录节点是外部访问集群系统强大计算或存储能力的唯一入口，是整个系统的关键点。为了保证用户节点的高可用性，应该采用硬件冗余的容错方法，如采用双机热备份的方式，或至少应该采用 RAID (redundant array of independent disks) 技术保证用户节点的数据安全性。

登录节点一般对计算性能要求不高，整套集群根据需求配置数台机架式服务器即可。

2) 管理节点

管理节点是集群系统各种管理措施的控制节点。它负责监控集群中各个节点和网络的运行状况。通常集群的管理软件也运行在这个节点上。

由于现代服务器强大的处理能力，通常集群的作业调度程序（如 PBS）也可以运行在这个节点上。管理节点是计算网络中的关键点，如果它失效，所有的计算节点都会失效（已提交的作业很有可能会挂起）。所以管理节点也应该有硬件冗余保护。

和登录节点一样，其对计算性能要求不高，整套集群根据需求配置数台机架式服务器即可，如图 2-1 所示。

3) 计算节点

整个集群的计算核心，功能是执行计算。根据需求和预算来决定配置。计算节点又分为瘦节点和胖节点两种。

瘦节点：出于预算考虑，集群的计算主力由配置相同的大量瘦节点构成，主要用于执行并行运算。由于集群中有大量瘦节点，节能和节省空间成为客户的重要需求之一，刀片服务器已成为主流，如图 2-2 所示。

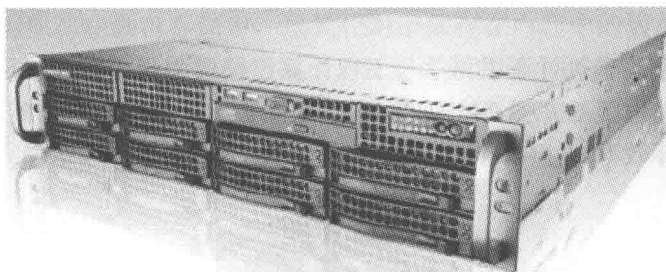


图 2-1 浪潮自主自研的机架式服务器 NF5280M3

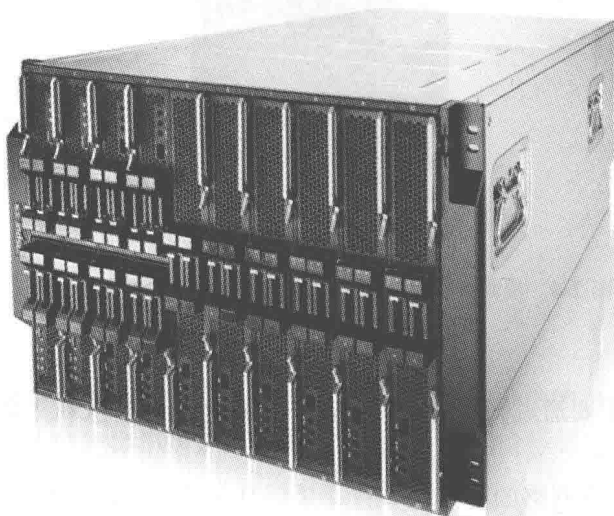


图 2-2 浪潮自主自研的刀片服务器 NX5440

胖节点用于数据划分困难或对内存需求特别大的特殊应用运算等。胖节点一般采用四路以上的服务器，内存容量大，价格也相对比较昂贵，如图 2-3 所示。

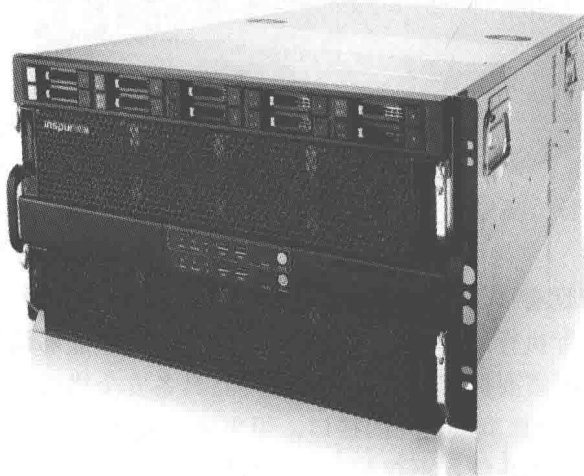


图 2-3 浪潮自主自研的八路服务器胖节点天梭 TS850

4) 异构节点

目前使用异构系统来执行通用科学与工程计算已经成为一个比较热门的方向，目前的异构节点通常同时使用了 CPU、GPU 或 MIC，如图 2-4 所示。利用 GPU 进行并行计算，可以大幅度提高计算效率。

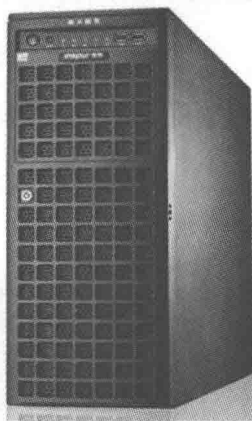


图 2-4 浪潮“倚天”系列异构超算服务器 NF5588M3

5) 交换设备

集群各节点之间需要通过网络连接在一起，节点之间的信息和数据的交换需要通过交换设备进行。

在大型集群中，计算网络的交换设备往往采用具备几百个端口的大型交换机，如图 2-5 所示。

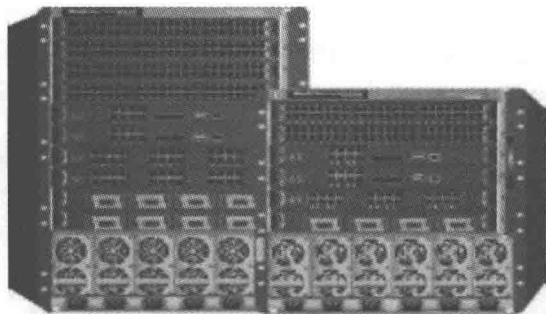


图 2-5 大型全万兆以太网交换机

6) I/O 节点和存储设备

为了使任务可以并行执行，每台执行任务的计算节点必须能够访问同样的数据。同时计算产生的大量重要数据需要拥有较大存储空间，且具备安全措施的专业设备进行存储。I/O 节点提供存储设备与计算节点之间的接口，用来确保数据访问的同步。

高性能计算系统的存储系统，不仅起着存储数据的作用，在计算过程中，还起着提高读/写带宽的作用。大多数软件的计算方法通常是将使用最频繁的数据保存在内存中，而把

使用不太频繁的数据保存在硬盘存储中,虽然相对于保存在内存中的数据,保存在硬盘中的数据使用频率稍低,但是由于其数据量很大,而且计算过程中也会随时更新数据,所以,数据的读/写带宽就要求很高,通常的本地硬盘读/写速度不能满足要求,需要专业的文件存储系统,如图 2-6 所示。

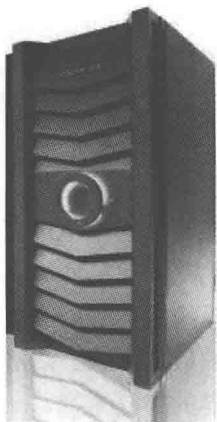


图 2-6 浪潮 AS10000 海量存储系统

7) 管理网络

用于管理节点和各计算节点、I/O 节点的互连。管理网所连接的机器就是集群内部的本地机器,所以高带宽和低延迟都不需要,同时可以容忍一定的过预订率(over-subscription)。由以上可知,千兆网很合适。

8) 计算网络

用于各计算节点间的互连,是并行任务执行时的进程间通信的专用网络,并行计算的核心就是它和集群内的其他节点交换信息的能力,通常称为 IPC(inter-process communications),它需要高性能的网络来进行快速交换,因此要求延迟小、带宽大。系统的互连结构和带宽非常重要,它决定了系统架构、性能和适合的应用等,如 Mesh 和 multidimensional Mesh 架构就比较适合流体力学等科学计算应用,由于这类应用的数据划分和交互方式特点,可以使系统发挥出很好的效率。

对于大规模的计算任务,若要大幅度提高计算速度,并行计算是唯一的解决方法,而随着 CPU 技术和内存技术的不断发展,单节点的计算能力不断增强,因此,为了能最大限度地发挥各个节点间的最大性能,网络的带宽和延迟都要求很高。计算网络的带宽越高、延迟越低,整套系统的计算效率越高。

计算网络一般选用千兆以太网、IB (InfiniBand) 网络或万兆以太网等。

9) 存储网络

存储网络需要向 HPC 集群的节点提供数据访问服务。

在 HPC 领域,根据数据存储和访问方式的不同,有几种不同的存储形式。在最低级别,可以有两种方法访问数据,一是数据由外部文件系统提供文件级别的访问,包括网络附属存储(network attached storage, NAS);二是数据块级别的访问,包括直连式存储

(direct-attached storage, DAS) 或存储区域网络 (storage area network, SAN) 可以分别使用基于 SCSI 或 SCSI RDMA (SRP) 协议的光纤通道或 IB 存储。

2.2 超算集群功耗监控与管理

随着技术创新，服务器的初始购置成本相对下降，电费支出、营业费用和环境影响等能耗相关成本却在持续上升。因此，提高服务器功耗效率、实现高效能计算，是计算机和超级计算的技术发展潮流。

超级计算机的运算能力非常强大，能耗也非常惊人。在能源日益紧张和强调环保的今天，超级计算机的设计越来越强调效能比，即单位能耗的计算能力。超级计算机的功耗管理不是简单地降低功耗或能耗，这需从两个方面入手，一方面在满足性能要求的前提下，优化功耗、能耗与性能的折中，提高系统的能效；一方面是设计发热更低、能源利用率更高的超级计算机部件和设计能耗更低的散热方式。后者不在本书的讨论范围内。

超级计算机的能耗优化已引起业界的广泛重视。一些人致力于新的系统结构，基于低功耗处理器构建高性能的超级计算系统，而更多的研究则是在现有的体系结构基础上探索如何提高能效。

在服务器的主要部件中，处理器功耗占系统功耗的主体地位，管理处理器功耗的方法主要有动态调频 (dynamic voltage frequency scaling, DVFS) 和处理器动态休眠技术。随着处理器低功耗技术的日益成熟，目前的主流操作系统都内置了相关的功耗管理方案。

超算集群系统科学计算的 I/O 量较少，计算、通信和同步是负载的主体，计算节点的运行模式集中体现了高性能计算的应用特征。处理器、供电模块、内存和风扇构成了高性能计算节点的功耗主体。图 2-7 为 idle 状态下的某机型服务器部件功耗分布。不同系统在不同负载条件下，系统功耗在部件间的具体分布比例有所不同，但处理器功耗占最大比例的结论仍不失一般性。

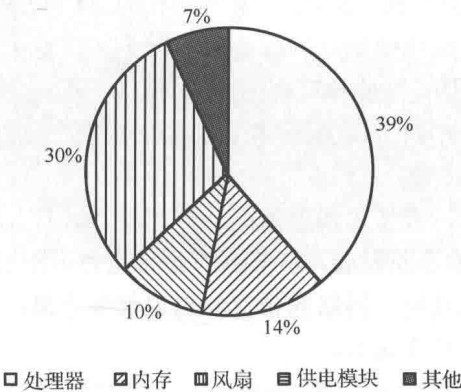


图 2-7 高性能计算节点某机型服务器 idle 状态下的功耗分布

目前已有的低功耗技术很多，并基于不同的低功耗技术衍生出各种行之有效的功耗管理方案。按技术的本质机制将支撑功耗管理的低功耗技术分为两类。

(1) 动态资源休眠 (dynamic resource sleeping, DRS)，即为了节能而休眠或关闭空闲的资源，如组件、设备或节点，需要时再将资源动态唤醒。目前的主流处理器都支持动态

休眠技术，有的处理器还支持多种休眠状态。

高级配置与电源接口（advanced configuration and power interface, ACPI）对处理器休眠状态（C 状态）进行了明确的规范。此外，有的内存也支持动态关闭，外围设备互连（peripheral component interconnect, PCI）功耗管理规范也对设备的动态关闭进行了相关的描述。超算集群系统还可以以节点为单位动态休眠相应的节点。

（2）动态速率调节（dynamic speed scaling, DSS），即动态调节设备的运行速率。并行计算存在大量的通信与同步，快速设备完成其承担的负载后必须等待慢速设备，此时快速设备的高速率是没有必要的，降低快速设备的速率，可以降低系统功耗而不损失系统性能，从而实现系统的能耗优化。处理器 DVFS 是典型的 DSS 机制，有的内存、磁盘也支持频率或转速的动态调节。

目前的商用处理器都支持 DRS 和 DSS 两种低功耗机制。

真实系统中，处理器不可能总是处于繁忙状态。如果不采取任何功耗管理措施，处理器空闲时将在操作系统控制下运行相关内核代码，循环等待，直到内核为其分配相应的用户负载。空闲处理器等待期间的指令运行造成了无谓的功耗浪费。

为此，处理器引入动态休眠技术，即空闲时停止执行指令，进入低功耗状态，直到需要时再被唤醒。超级计算机节点通常采用多核多处理器结构，并行计算中存在大量的通信与同步，这就为处理器休眠提供了潜在的机会。处理器可能支持多个休眠状态，不同休眠状态的逻辑行为相同，但功耗不同，从功耗更低的休眠状态唤醒所需的时间和能耗的开销都更大。举例来说，Intel Xeon 处理器支持增强型空闲电源管理状态转换（enhanced halt state）技术，除了正常运行状态 C0 外，还支持休眠状态 C1 和增强型深度休眠状态 C1E 等。Linux 内核引入 CPUIdle 模块来管理空闲处理器，根据处理器空闲历史记录判断处理器是否休眠，并选择合适的休眠状态。

从一些测试结果看，C1E 完成大部分测试用例的时间比 C1 更长，能耗更少，这与两种休眠状态的特点一致。但是，对于计算非常密集的应用，唤醒时间的影响尤为显著，C1E 尽管功耗更低，但执行时间增加得更多，能耗反而上升。

处理器调频对系统能效的优化效果与应用的特征密切相关。对于通信密集的体现系统通信能力的应用，降频可能会带来 10% 以上能效优化，但对于计算密集的体现处理器浮点计算能力的应用，降频反而降低了能效。

没有任何频率对所有应用都是能效最优的，应该根据并行应用中计算和通信的相对特征动态选择处理器性能。超级计算中，基于处理器性能调节机制来优化系统能耗还有很大的潜力可挖。

超算集群的能耗监控一般有两个层次，第一个层次是整集群的监控，一般以机柜为单位，每个机柜配备支持网络监控功耗功能的电源分配单元（power distribution unit, PDU），由监控软件汇总每个机柜 PDU 的功耗数据。这种能耗监控的数据是根据 PDU 整体的电压电流数据经过计算得出的，连接到该 PDU 的所有设备如交换机、存储等都可以监控到，也是预估电费的直接数据。

第二个层次是以节点为单位的监控，既可以通过专业监控软件监控，也可以通过一些简单的工具或操作系统（operating system, OS）自带命令实现。这里不再赘述。

2.3 构建性能均衡的超算系统

2.3.1 性能均衡的超算系统

Amdahl 定律告诉人们,当 CPU 的性能提高 10 倍而忽略 I/O 的性能时,系统的性能只能提高 5 倍;当 CPU 的性能提高 100 倍而不改进 I/O 的性能时,系统的性能只能提高 10 倍。所以整个系统各部件间的性能平衡显得非常重要,否则如果某些部件的性能较差而成为系统瓶颈,不仅降低了整个系统的性能,同时也不能充分利用其他的部件。

从大的方面来看,超级计算机的系统平衡应该有两个层次。一个层次是体系结构层次上的,这主要是各个部件的硬件性能应达到一个平衡的状态;另一个层次是软件层次上的,主要是指操作系统对各种资源进行有效的管理,达到负载平衡(如不能让一些节点始终处于忙碌状态,而一些节点始终空闲),提高利用率。

超级计算机主要由计算部件、存储部件、互连部件构成,构建一个平衡的系统实质上就是协调这三大部件的主要性能,使系统在某种工作负载下各部件既无冗余又不产生瓶颈。

如何构建性能均衡的高效能计算系统呢?可以从如下几方面考虑来实现系统的性能均衡。

1) 节点内配置均衡

(1) CPU 同构类型。

对于节点内的配置,应尽可能保持一致,以提高数据交换能力和资源利用率。例如,节点内是双路的,则对于每一路处理器配置的内存应尽可能保持一致,以避免两个处理器对应内存差别太大,造成两个处理器的处理能力差距悬殊,导致整个节点处理能力下降。例如,对于双路处理器的节点,两个处理器均配置 24GB 的内存,比一个配置 24GB,另外一个配置 12GB 的内存处理效果要好。

(2) 异构类型。

使用 CPU 与 GPU 异构模式的计算系统,主机内存应大于等于 GPU 内存;创建统一的内存地址空间,让 CPU 和 GPU 完全共享内存以实现无缝的运作;CPU 核数与 GPU 个数保持数量的均衡;GPU 内存带宽与主机内存带宽均衡,以减少数据交换带来的延迟。

2) 节点间配置均衡

同类计算节点保持配置均衡,如内存大小相同,CPU 数量及核数相同,节点结构相同,要么都是同构,要么都是异构,避免同类节点结构不一样(如一个是同构架构,另一个是异构架构)导致数据处理能力降低的情况。

3) 网络均衡

同类节点与节点间的网络连接采用统一的方式,包括同样的网络介质,如千兆以太网、万兆以太网、IB 网络等。

交换机：要么都使用全交换方式，要么都使用半交换方式，否则会由于带宽的不一致，导致信息阻塞。

4) 执行同一应用的节点配置均衡

执行同一应用的节点配置应保持一致，这样在进行数据处理和交换时速度会更快。若由于客观原因实在不能保持一致，则通过集群管理软件来进行作业调度，作业提交后，可以均衡地提交到集群中配置相当的节点上进行运算，避免因某些节点存在瓶颈降低整个应用的运行效率。

5) 不同设备间的均衡

计算节点与 I/O 节点和存储设备间应均衡匹配，高性能计算系统的存储系统，不仅起着备份数据的作用，在计算过程中，还起着提高读/写带宽的作用。

对某些应用主要是进行计算，存储数据和读/写数据不多，那么 I/O 节点机存储配置要求就不用太高。

但对于有些应用，节点上运行的应用比较复杂，涉及大量数据读/写及数据存储，如基因研究、石油勘察等数据之间关联性强（下一步计算要依赖上一步的运算结果）的应用，就需要对应的 I/O 和存储系统配置高。

6) 功耗均衡

在系统构建时需要考虑到机房的实际情况，均衡分布设备，避免出现局部功耗太高或者局部功耗太低的情况。

在满足系统功能的前提下，尽可能将能耗高的设备和能耗低的设备、密度高的设备和密度低的设备组合搭配，如将刀片服务器和机架服务器放到同一个机柜中，避免不同部位间能耗相差太大，造成局部过热或者局部有资源浪费的情况。

另外，在机房中靠近制冷设备的地方可以考虑放置功耗大、密度大的设备等以合理地利用资源。

2.3.2 如何管理高效能计算系统

随着高性能计算机在很多领域中得到广泛的应用，高性能计算机管理也已经引起大多数人的关注，计算机性能不断提高，系统节点规模也不断扩大，计算节点规模巨大，硬件设备繁多，软件配置复杂，用户数量和作业数量增加，作业队列的数目和长度增加，这些量的变化将导致高效能计算系统管理质的变化。

为了达到更高的效能，高效能计算系统需要综合管理计算资源、网络资源、存储资源、能耗资源、作业应用等，如图 2-8 所示。对于计算资源而言，硬件体系结构中的异构技术和多核技术的出现带来了更多的管理复杂性；对于网络资源而言，由于互连网络的规模扩大，互连拓扑结构到应用程序的映射对程序性能的影响不可忽视；对于存储资源而言，系统中存储级别增多，多核系统的 Cache 层次增多，文件系统的分布结构等带来存储的管理需求；对于作业而言，并行编程的模式增加带来了多模式作业的有效管理需求。



图 2-8 浪潮构建的超级计算机生态结构图

为了适应高效能计算的可用性目标，高效能计算系统的管理需要具有灵活易用的功能扩展力，支持系统管理目标和实际应用目标的快速调整。同时，不断增长的并行应用需求给大规模处理系统带来了更重、更复杂的工作负载，而且多领域多业务的应用具有不同的并行特征，其作业并行模式各异，时空规模差异较大，增加了作业调度的复杂性，使系统的作业吞吐率难以随系统规模的增大而持续提升，存在调度策略的可扩展问题，需要研究更适合大规模并行处理系统特点和应用特点的调度方式，增强调度策略的适应性，以达到高产率的目的。

国内的高性能计算机厂商，需要关注客户在应用和产品技术方面的持续创新，要将高性能计算机运用到实际行业，如生命、化学、材料、物理、计算机辅助工程（computer aided engineering, CAE）、石油、气象、环保、高校、科研等领域中。另外，对于高性能计算机的管理需要开发配置一些高效能系统管理软件，如集群监控软件、集群作业调度系统、系统部署软件和系统备份软件等，以此为客户操作带来便利，同时也提高高效能系统的实际利用率，为高性能计算集群系统的正常运行提供强有力的支持。

2.3.3 监控管理

集群的监控管理软件，主要实现对集群系统的单一映射，从而为系统管理员提供监控管理集群中各个节点的统一平台。集群监控软件既可对集群系统中的节点进行分组监控管理，也可针对某一节点进行单一的管理。为系统管理员提供一个单一映像的服务器管理界面。现有流行服务器的集群管理软件主要通过B/S架构实现高效率的远程管理，同时具有分组管理、并行操作、跨平台、单一映像、准三维的图形界面等功能，并且有些监控管理软件还有记账、报警等多项功能，从而可以极大地降低用户的管理成本、提高用户的管理效率。

通过对集群的监控管理，如对处理器、内存、网卡、负载、页面、上下文、交换分区、磁盘容量、磁盘读/写等进行监控，实时采集各项数据，以帮助集群维护人员获得基础数据，以便对系统进行分析和管理等。

集群监控系统提供集中的界面操作方式,如图 2-9 所示,使管理员可以很好地对集群系统和相应资源进行集中管理,如进行用户或用户组的创建、对集群节点的关闭重启等,避免了维护人员一定要通过对物理机器的操作才能完成相应的任务(操作分散),另外通过系统提供的监控日志等能帮助维护人员更好地定位和解决问题。

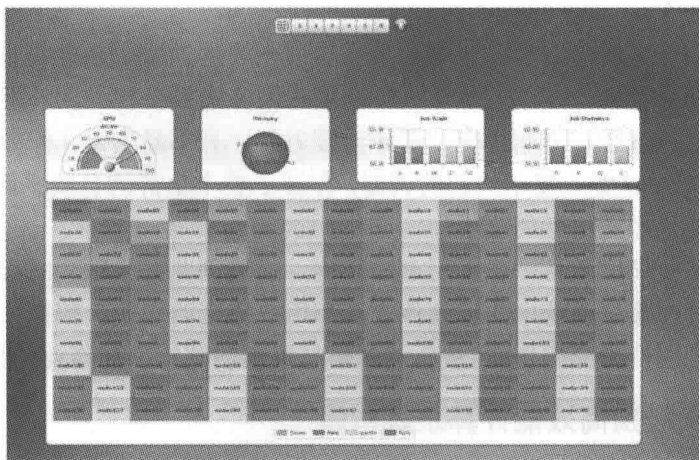


图 2-9 浪潮 ClusterEngine 集群监控页面

高性能集群系统构成复杂,节点和相应设备众多,对于集群系统的日常维护,若通过集群维护人员一一检查的方式来发现问题和对系统进行维护,则面对成百上千的节点和面积庞大的机房,这个工作难以展开,且工作效率低。但通过报警功能,通过相应设备对集群系统的主动扫描,一旦系统发生故障,自动将问题通过声光报警或者电子邮件或者手机短信的方式报告给维护人员,这样维护人员可以做到有的放矢,只针对故障节点和设备进行问题解决,不用关注正常运行的设备,这样将大大减轻维护人员的工作量。报警是“人防、技防、物防”相结合的安全防范体系,是公认的目前最有效的安全防范手段。用户花很少的投入即可获得全方位、全天候的安全防范服务。

2.3.4 作业调度

高性能计算集群平台的应用环境非常复杂,其应用领域遍及基础科学研究(如计算化学、地球物理、人工智能等);社会公益事业(如数值气象预报、地震预报);工业与工程计算(如航空航天、汽车设计、地质勘探等);数据处理(如金融、政务)等多个领域,这就要求高性能计算集群平台的作业调度系统必须能对各个应用软件具有非常良好的支持,不仅能支持各种工程计算应用,还能支持各种各样的基础科学研究。这意味着作业调度系统要与各个商业应用软件具有良好的接口和后台支撑,同时还必须支持各种开源软件,确保各种应用软件在集群环境下的稳定应用。

作业调度系统是提供用户使用高性能平台的通道,通过登录节点作为统一接口访问高性能服务,同时,将用户与高性能后台系统隔离,提高安全性和可管理性。在管理网络层,作业调度系统提供对高性能平台进行部署、监控、调度 and 管理的通信平台,保证系统可控性和可管理性。在计算网络层,作业调度系统提供高性能应用、并行计算应用之间的数据

通信,减少延迟和增大带宽,保证高性能计算应用的高效率 and 高质量。在存储网络层,作业调度系统提供存储服务器与存储设备高 I/O 吞吐和高流量通信访问,保证系统对存储利用的高并发和大吞吐。当然,其他节点对存储访问不仅依赖存储网络,同时还将依赖 I/O 服务器与其他服务器之间的管理网络或计算网络(取决于 I/O 服务器使用管理网络还是计算网络提供 I/O 服务)。

传统的作业调度系统需要通过使用命令来实现作业的调度,命令多且复杂,对使用人员的计算机水平要求较高,使用门槛较高,需要应用专家具备相关的计算机使用水平。目前的作业调度系统通过统一界面提交,操作比较方便,且不易出错,作业调度系统很好地帮助了 HPC 集群解决大型、复杂作业的调度,保证了整个集群的负载平衡,提高了作业运行效率,大大降低了作业运行的成本。

目前主流的作业调度软件一般采用 B/S 架构,通过浏览器(IE、Maxton、Firefox 等)进行操作,可以管理集群系统中的软硬件资源和用户提交的作业,根据集群中的资源使用情况来合理地调度用户提交的作业,从而达到提高资源的利用率和作业的执行效率的作用。

2.3.5 十万亿次级别高效能计算系统构建

每秒计算能力达到十万亿次是现代 HPC 的起步门槛,运算速度达到每秒十万亿次以上的计算机集群已初步具备了高性能计算集群的雏形。由于十万亿次系统 is 高性能计算机的起点,所以其构成不像百万亿次或者计算能力更强的高效能计算机那么复杂,如其管理节点和登录节点可以不是单独的节点,可以与计算节点共同使用一个节点,另外对网络和不间断电源等的要求也相对较低。十万亿次级别高效能计算系统主要可以通过两种方式实现,一种是完全使用 CPU 同构的方式实现,另外一种则是通过异构的方式实现。

1. 构建十万亿次的高效能计算系统

构建十万亿次计算系统主要需要从计算能力、管理、互连网络、存储、操作系统和集群网络安全方面来进行考虑。

1) 计算能力

作为整套集群的主体结构,计算节点是整套系统的基础,其性能直接决定了整套系统的整体性能,所以在处理器的选择上,应尽可能去选择在相同的芯片大小下能容纳更多的运算核心的处理器,同时要注意选择性能更高、功耗更小的处理器,如 22nm 的 Intel Haswell 处理器。

计算节点是一个完整的计算系统,离不开各个部件的协同工作。在高性能计算中,几乎所有的计算软件在计算过程中,都需要 CPU 与内存频繁通信,超高的 CPU 性能只有匹配了相应的内存才可以发挥最佳的优势。为此,系统构建时尽量配置容量大、带宽高的内存,提升性能,为每个 CPU 提供访问本地内存资源的快速通道。

2) 管理

管理方面,高性能计算系统涉及的硬件设备繁多,软件配置复杂,用户数量和作业数量增加,好的集群监控管理系统对系统的管理,可以提高高性能计算系统的管理效率,管

理包括各节点的管理、各节点对应操作系统及软件管理，还包括制冷设备、机房环境等的管理。

3) 互连网络

目前的高性能计算机网络比较主流的是使用 IB 网络和 GbE (Gigabit Ethernet) 网络。

IB 是一个统一的互连结构，既可以用于存储 I/O、网络 I/O，也可以用于进程间通信 (IPC)。它可以将磁盘阵列、SAN、LAN、服务器和集群服务器进行互连，也可以连接外部网络 (如广域网 (WAN)、虚拟专用网络 (VPN)、互联网)。主要是实现高的可靠性、可用性、可扩展性和高的性能。IB 可以在相对短的距离内提供高带宽、低延迟的传输，而且在单个或多个互连网络中支持冗余的 I/O 通道，因此能保持数据中心在局部故障时仍能运转。近年来 IB 网络在高性能计算机中的使用迅猛发展，从 2015 年 11 月的 Top500 排行来看，使用 IB 网络的机器数量已达 237，占总量的 47.4%。

GbE 网络，又称千兆以太网 (GbE、Gigabit Ethernet 或 1 GigE)，由 IEEE 802.3-2005 标准定义。GbE 网络曾是高性能计算机中使用最多的网络，近几年随着 IB 网络的迅猛发展，GbE 所占份额略有降低，从 2015 年 11 月的 Top500 的排行来看，使用 GbE 网络的数量为 62，占总数的 12.4%。

4) 存储

高性能计算系统的存储系统，不仅起着备份数据的作用，在计算过程中，还起着提高读/写带宽的作用，目前常用的存储系统主要包括 DAS 和 NAS、SAN 及并行文件系统存储等。

DAS 是 direct-attached storage 的简称，中文称为直连式存储，以服务器为中心，将 RAID 硬盘阵列直接安装到网络系统的服务器上，存储设备是通过电缆 (通常是 SCSI 接口电缆) 直接到服务器的。I/O 请求直接发送到存储设备，它依赖于服务器，其本身是硬件的堆叠，不带有任何存储操作系统，需相应服务器的操作系统支持。

NAS 是 network attached storage 的简称，中文称为网络接入存储。在 NAS 存储结构中，存储系统不再通过 I/O 总线附属某个特定的服务器或客户机，而是直接通过网络接口与网络直接连接，由用户通过网络访问，该存储将不同系统平台下的文件存储在一台 NAS 设备中，方便网络管理员集中管理大量的数据，降低维护的成本。

SAN 是 storage area network 的简称，中文称为存储区域网络。它是一种高速网络或子网络，提供在计算机与存储系统之间的数据传输。一个 SAN 由负责网络连接的通信结构、负责组织连接的管理层、存储部件和计算机系统构成，从而保证数据传输的安全性和力度。

并行文件存储系统主要用来提供高带宽的存储容量管理。并行文件系统是应用于多机环境的网络文件系统，单个文件的数据采用条带化等形式存放于不同的 I/O 节点之上，支持多机多个进程的并发存取，同时支持元数据和数据的分布存放，并提供单一的目录空间。

5) 操作系统

操作系统方面，从 Top500 的统计结果来看，越来越多地采用开源的操作系统 Linux。不仅是因为 Linux 系统的低软件成本 (Linux 是开放源代码的操作系统)、低人员培训成本、

低移植成本，更重要的是其提供了一个非常稳定的、易扩展的平台，使得用户可以根据自己的需要不断提升系统性能。

6) 集群网络安全

集群管理员为了方便管理集群系统，往往将办公网络与集群的管理网络连接在一起，然而办公网络一般与互联网是互通的，这种“假物理隔离”的连接方式存在着很大的安全隐患，在方便自己的同时也悄然地为黑客打开了一条绿色通道。对集群系统的攻击主要包括面向应用层的攻击、面向网络层的攻击、面对网络基础设施的攻击这三个方面。

种种来自外部的网络威胁可以通过在集群系统前端架设防火墙和 VPN 服务器等手段进行防护。其具备防护功能的架构如图 2-10 所示。

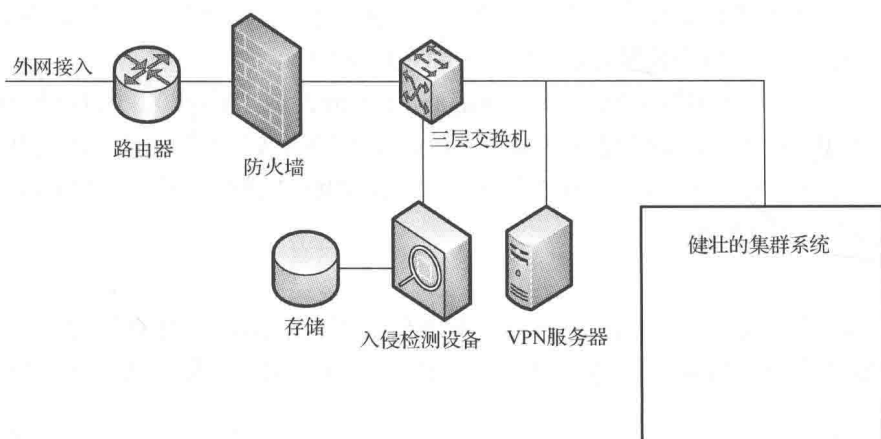


图 2-10 集群系统防护图

(1) 防火墙。

目前市面上的防火墙主要分为两类：硬件防火墙和软件防火墙。软件的防火墙大多都是基于 PC 架构的，可能会采用经过优化的 OS 作为其运行的平台，特点是扩展性好、适应性强、容易升级而且成本远低于基于硬件的防火墙。

基于硬件的防火墙大多采用 ASIC，其特点是速度快、稳定性好、安全系数要高于软件的防火墙，但成本较高、扩展性和易升级性不如软件防火墙。

由于硬件防火墙往往成本很高，在实际应用中相对较少，所以日常使用开源的软件防火墙搭载在服务器上的情况比较多。基于 Linux 的开源防火墙有很多方案，如 Zentyal、pfSense、IPFire、SmoothWall、ClearOS、Iptables 等。其中 Iptables 是嵌入 Linux 内核中的，并且 Iptables 防火墙是一种很强悍的防火墙，只要规则配置好，能大大提高系统安全性。另外，可以通过配置 Iptables 对 IP 欺骗问题和拒绝服务（denial of service, DoS）攻击问题进行基本防范。

(2) VPN 服务器。

VPN 是一种远程访问技术，简单地说就是利用公网链路架设私有网络。其之所以称为虚拟网，主要是整个 VPN 的任意两个节点之间的连接并没有传统专网所需的端到端的物理链路，而是架构在公用网络服务商所提供的网络平台上，如异步传输模式（ATM）、帧中

继 (frame relay) 等之上的逻辑网络, 用户数据在逻辑链路中传输。它涵盖了跨共享网络或公共网络的封装、加密和身份验证链接的专用网络的扩展。OpenVPN 是 Linux 下开源 VPN 的最成熟、最稳定的技术, 允许参与建立 VPN 的节点使用预设的私钥、第三方证书, 或者用户名/密码来进行身份验证。大量使用了 OpenSSL 加密库和 SSLv3/TLSv1 协议。OpenVPN 能在 Linux、xBSD、Mac OS X 与 Windows 2000/XP 上运行。

VPN 主要采用了隧道技术、加解密技术、密钥管理技术和使用者与设备身份认证技术, OpenVPN 与生俱来便具备了许多安全特性: 它在用户空间运行, 不需要对内核和网络协议栈进行修改; 初始化完毕后以 chroot 方式运行, 放弃 root 权限; 使用 mlockall 以防止敏感数据交换到磁盘。

2. 以同构方式构建十万亿次集群

通常十万亿次以上的 CPU 同构集群, 从散热、噪声和供电三方面考虑, 均已对人的正常办公造成很大影响, 必须放置在专门的机房中。关于机房, 可以参考国家计算机机房标准。

1) 计算节点

以支持先进向量扩展 (advanced vector extensions, AVX) 的 Intel Haswell 平台考虑, 最多 20 个双路节点足够支撑十万亿次计算性能, 计算方法如下。

$2 \text{ (每节点 CPU 数量)} \times 8 \text{ (每时钟周期浮点运算次数/每计算核心)} \times 8 \text{ (每 CPU 核数)} = \text{每计算节点浮点运算能力}$ 。以 Intel E5-2650V3 运算频率计算, 一个双路计算节点的运算能力在 600Gflops 以上, 则十万亿次 (10Tflops=10×1000Gflops) 的浮点运算峰值 39 个计算节点就可以满足。

17 个计算节点类型的选择则主要分为机架式和刀片式。

如果采用机架式服务器满足计算能力, 至少需要 39U 的机架空间——以 1U 标准机架式双路服务器来搭建, 则需要至少一个 17U 的标准机柜来容纳。

如果选用刀片式服务器, 以目前流行的高密度刀片服务器浪潮 NX5440 举例, 则 8U 空间就可以容纳 20 个双路节点。一台 NX5440 M4 即可满足全部计算能力需求。

2) 管理节点

十万亿次左右的集群一般不会有大规模的用户使用。用户数量一般在个位数到 20 位左右。对集群的管理监控和作业调度压力都较小, 不需要采用昂贵的商业集群软件, 完全可以采用免费或开源的软件来满足需求, 如开源的 OpenPBS 等。

关于管理节点, 如前面所述, 因为对硬件的压力不大, 可以和登录节点共用同一台机器。

3) 登录节点

如前面所述, 可以和管理节点共用同一台机器。可以采用比计算节点配置稍差的节点, 如单独的一台机架式服务器。

4) 交换网络

如果采用 IB 网络, 需要从计算节点的不同情况来考虑。如果采用机架式节点, 则需要

采用单独的 IB 交换机配置,一般的 1U IB 交换机的接口数量配置为 36,超过 36 个接口还需要再多配置交换机。这种情况下,需要考虑 IB 网络是否需要做全线速。

如果需要做全线速,则还需要考虑交换机之间和服务器之间如何互连最为经济。请详见后面针对网络的专门分析。

如果采用刀片节点,则因为刀片节点自带 IB 交换模块,两台刀片之间直接 IB 互连即可。

如采用以太网网络,如果只用千兆以太网作为计算网络,一台 48 个网络接口的千兆以太网交换机即可满足。在采用刀片节点的情况,将两台 NX5440M4 刀片的千兆交换模块的万兆上行口互连即可。

5) 管理网络

- 一般使用千兆以太网作为管理网络,互连方法如前面所述。

6) 存储

一般十万亿次的高性能计算集群对 I/O 的压力不会很大,在这种情况下,根据不同情况可以采用不同方案。

计算结果数据很小时,可以直接采用管理节点的硬盘进行网络文件系统(network file system, NFS)共享,将计算结果数据定期备份即可。

计算产生的数据较大时,建议配置存储服务器或磁盘阵列等专业存储设备,此时文件系统可以考虑 XFS(X file system)等。

3. 以异构方式构建十万亿次集群

异构方式搭建十万亿次计算集群,相对简单得多,以 NVIDIA Tesla K20M 或者 Intel Xeon Phi 的 1000Gflops 的计算能力,10 块加速卡即可满足计算能力的需求。如果加上 CPU 的计算能力,所需的机器数量较同构方式将大大减少。

采用异构方式构建集群,需要考虑应用特性和性能均衡,要考虑采用何种硬件架构。例如,应用采用 OpenMP 编写,在同一个节点内扩展性比较好时,可以考虑采用高密度的异构服务器,两个 CPU 对应四块 GPU 卡,即一台服务器配置四块 GPU。

如果应用的 MPI 扩展性比较好,希望该集群除了进行 GPU 应用计算,还可以进行较大规模的 CPU 计算,那么可以两个 CPU 对应一块 GPU,即一台服务器只配置一块 GPU。

以节点数最多的后一种模式考虑,仅需要 12 台服务器,每台服务器配置一块 GPU 即可满足计算性能需要。

关于计算节点和网络的考虑与 CPU 同构集群相同。而关于管理节点和登录节点则需要说明,必须采用可以支持对 GPU 进行管理和调度的集群软件。浪潮 ClusterEngine 可以对 GPU 进行监控管理和作业调度,加上中文便捷界面,比较合适。

第 3 课 超算系统的网络通信

主流网络互连技术包括以太网（Ethernet）、FC 和 InfiniBand。FC 是为实现存储互连而设计的存储专属网络。由于其技术局限性，迄今只在存储互连领域得到广泛应用。以太网与 InfiniBand 网络同为开放网络互连技术，以太网更侧重网络协议的通用性，局域网与广域网遵循统一的以太网协议。因此以太网在网络信息传输方面得到最普遍性的采用。随着以太网速度大幅超越 FC，在存储领域也涌现出支持以太网的存储协议，如小型计算机系统接口（internet small computer system interface, iSCSI），以太网开始走进存储网络。然而 InfiniBand 网络的创立目的则是弥补 FC 与以太网的不足，同时满足广义网络传输与存储 I/O 对网络性能与智能性的需求。InfiniBand 拥有大幅领先于以太网与 FC 性能的优势，同时是具备软件定义网络（software defined network, SDN）属性的智能网络。InfiniBand 十多年来逐渐成为对网络性能要求最高的高性能计算领域最为广泛应用的网络，并且高速挺进存储、大数据与云计算领域。

本章将重点讲解 InfiniBand 网络的特点和如何优化应用程序来充分发挥 InfiniBand 的性能优势。

3.1 InfiniBand 技术概要

InfiniBand 由 IBTA（InfiniBand Trade Association）协会制定规范。它是一种开放标准的高带宽、低延迟网络互连技术。InfiniBand 网络可以以极低的 CPU 负荷消耗来实现多种网络流量类型的高性能，包括集群通信网、存储网与管理网。并且所有这些传输可以同时同一 InfiniBand 链路实现。InfiniBand 不仅可以构建高性能数据中心内部高速网，通过远程 InfiniBand 互连设备，还可以实现数据中心与数据中心之间的高速数据传输。

InfiniBand 架构介绍如下。

1) HCA

主机通道适配器（host channel adapter, HCA）是 InfiniBand 网络的最终节点，可以安装在服务器或存储中，实现服务器、存储与 InfiniBand 网络的连接。HCA 之间创建通道实现互通，通常称为队列组（queue pair）。每个队列组由发送队列（send queue）与接收队列（receive queue）组成。队列组可随应用程序的需要而动态创建。通过队列构建信息通道，避免了通信对 OS 的依赖，从而实现应用对 HCA 网络资源的直接访问。

而传统网络应用需要通过操作系统来间接实现对网络资源的访问。额外环节势必影响通信性能。

2) TCA

目标通道适配器（target channel adapter, TCA）主要为内嵌系统定义的适配器架构，不用提供面向应用程序的标准应用程序接口（application program interface, API），迄今应用只局限于很小的领域。

3) 网络交换机

类似于传统网络交换机,实现网络通信的交换。但是 InfiniBand 交换机的设计更侧重于高性能与更佳性价比。它采用直通转发技术(cut-through)来实现网络流量的快速传递,同时降低设计复杂硬件的成本。传统存储转发交换(store-forward)性能低而且硬件成本高。InfiniBand 交换机实现链路层数据流控制机制保证网络数据包不丢失,这个特性对 InfiniBand 实现可靠传输协议非常重要。

4) 路由器

InfiniBand 网络路由器可实现 InfiniBand 多子网直接互连。过去 InfiniBand 的路由需要很低,但随着计算系统的规模不断增加与云计算的出现,单一子网模式也受到挑战,InfiniBand 路由器变得越来越重要。通过路由器可以实现跨域甚至分布于世界各地的 InfiniBand 集群的协同工作。

5) 网线与连接模块

InfiniBand 定义线缆规范,支持铜线与光缆,包括(4x、12x)线宽与不同速度(SDR(single data rate)、DDR(dual data rate)、QDR(quad data rate))。

通过对完整系统的定义,InfiniBand 为数据中心提供了最高效的网络互连技术。InfiniBand 的独特优势能够解决这一挑战。InfiniBand 的典型优势如下。

(1) 最高性能。最新 FDR InfiniBand 提供 56Gbit/s 的主机对外吞吐量。这里的 56Gbit/s 指单芯片、单端口。Mellanox InfiniBand 单芯片网卡,双端口绑定可以实现 100Gbit/s。EDR InfiniBand 交换机单端口可实现 100Gbit/s (Mellanox 公司在 2014 年 6 月宣布首款支持 100Gbit/s 的 EDR InfiniBand 技术交换机)。

(2) 低延迟。InfiniBand 具备极低的通信延迟,FDR InfiniBand 可以实现端到端低于 0.7 μ s 的延迟,极大地加速现代数据中心与 HPC 领域的应用程序。

(3) 高效率。InfiniBand 内置支持更佳优化的网络传输协议,如 RDMAC(remote direct memory access),通过内核旁路(kernel bypass)、零拷贝(zero copy)和通信协议卸载(transport offload)来实现最高效的网络数据传输。

(4) 可靠稳定的网络互连。InfiniBand 的高可靠性设计非常适合关键任务的处理,如全冗余、无损 I/O(lossless I/O)、多路径失效转移(failover)等。

(5) 网络整合实现更低功耗与成本。InfiniBand 通过多网融合(计算、管理、存储),将数据中心的多套网络整合成单一网络。从而降低网络采购与运营成本,同时有效降低数据中心整体能耗。实现最佳数据中心投资回报率。

(6) 数据完整性。InfiniBand 在每个网络中间的每个节点与端到端都具有循环冗余检查(cyclic redundancy check, CRC)功能,从而能够保证最高的数据传输准确性。

(7) 开放性。InfiniBand 具备完整的生态链、完整的开源代码软件支持(OpenFabrics Alliance 提供)。

InfiniBand 可以为目前和未来对于网络性能要求苛刻的应用提供最佳解决方案。图 3-1 是 InfiniBand 的发展路线图。EDR 产品已经问世,2017 年左右将实现 HDR(200Gbit/s)。

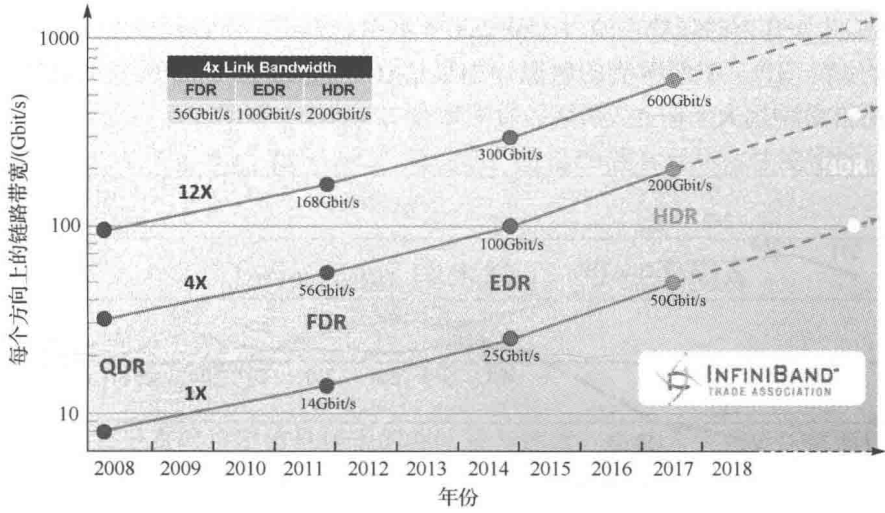


图 3-1 InfiniBand 的发展路线图

3.2 InfiniBand 在 HPC 中的应用现状

集群模式构建的超算系统已经占据 Top500（全球超算系统性能排行榜）的绝对主流。

图 3-2 所示为计算集群在 2014 年 6 月 Top500 排行榜中占比高达 85.4%。计算集群带来的高可扩展性与高性价比无疑是其逐渐取代 MPP 机型的驱动力。为了保证系统发挥尽可能高的整体效率，集群间的网络通信无疑是最关键因素。系统需要高带宽、低延迟、高可扩展性、低系统负荷的网络来提高集群通信的性能。图 3-3 展现了网络互连技术在 Top500 中的使用情况，InfiniBand 超越以太网成为 Top500 中最为主流的网络。由于 Top500 排名

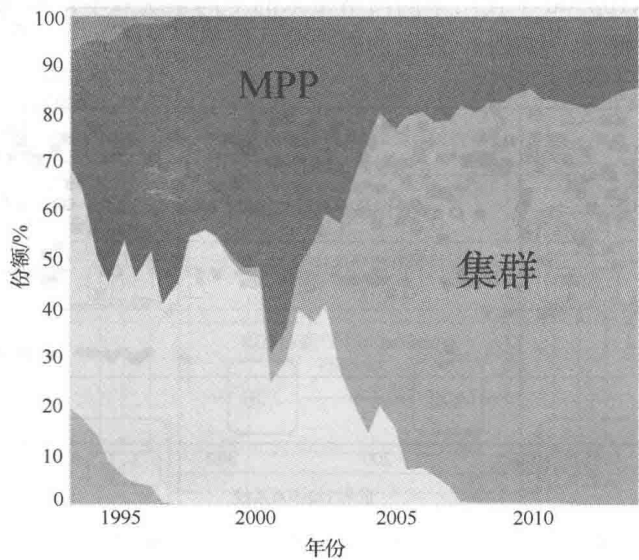


图 3-2 集群在 Top500 发展趋势

中有部分系统并非真正的超算系统，而是普通数据中心系统，所以这里显示的网络分布图并不能完全代表 HPC 使用网络的情况。如果将真正意义的 HPC 系统单独统计，相信 InfiniBand 所占份额远大于图 3-3 所示。

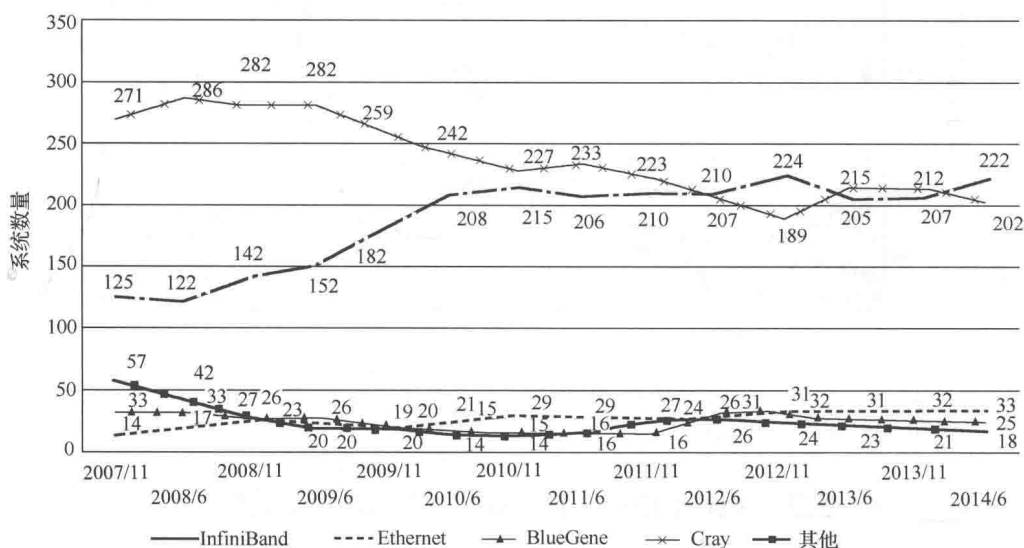


图 3-3 Top500 网络互连趋势图

之所以越来越多的 HPC 系统采用 InfiniBand，其原因就是 InfiniBand 是现在工业标准网络中的最高性能网络，同时也是最佳性价比的网络。InfiniBand 能够为系统带来前所未有的极高效率，最高甚至达到 99.8% 的系统效率。系统效率=实测浮点峰值/理论浮点峰值。其中实测浮点峰值是系统 LINPACK 实测性能值，而理论浮点峰值=CPU sockets × CPU 主频 × 每个时钟周期执行浮点运算的次数 × 系统中 CPU 核心数目。

图 3-4 列举出了 Top500 中使用不同网络带来的不同系统效率。InfiniBand 系统效率大幅优于以太网与私有网络，如 Cray 公司的专用网络。

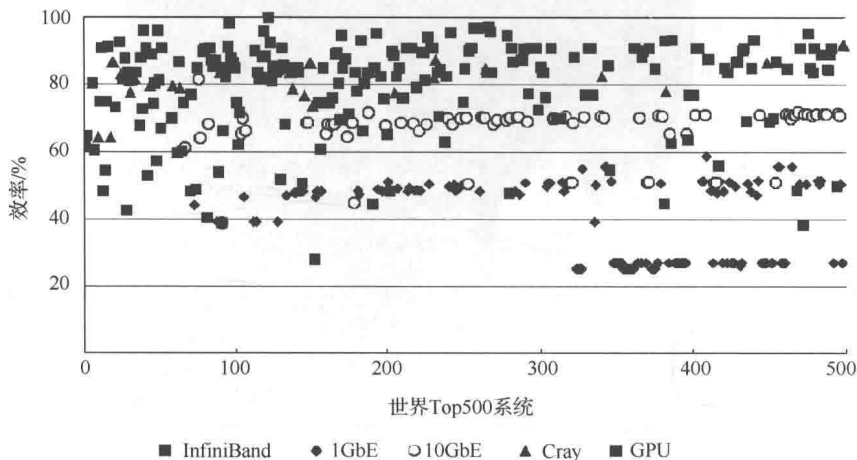


图 3-4 Top500 系统效率分布图

关于 LINPACK 效率之争已经持续很长时间,虽然 LINPACK 效率高不能完全说明系统能够同样对每个应用都提供最好的实际性能(实际应用还要考虑到很多定制细节来满足不同应用的需求),但是反之如果 LINPACK 效率不好的系统,几乎无可能为实际应用程序带来好的性能与可扩展性。所以 LINPACK 对超算的评定仍然具有其他性能指标程序所不具备的通用性与价值。

3.3 InfiniBand 技术核心——RDMA

3.3.1 RDMA 的技术简介

网络带宽与延迟直接决定信息传输的吞吐量与速度,纵然至关重要,然而仅凭这两个指标还不能完全直接评定网络对应用程序的最终影响。网络硬件具备的高带宽与低延迟要转化到应用层面的性能,需要支持优化的网络传输协议——RDMA 远程内存直接访问技术。不同的 InfiniBand 芯片技术支持 RDMA 的方式也各有不同,大体有两种方式:卸载(offload)与加载(onload)。卸载技术是将网络协议(包括 RDMA)处理过程内置在网卡芯片,然而加载技术主要借助 CPU 来处理网络协议。

TCP/IP 网络通信协议是因特网最基本的协议、Internet 的基础,由网络层的互联网协议(internet protocol, IP)和传输层的传输控制协议(transmission control protocol, TCP)组成。协议采用了四层的层级结构(网络接口层、网络层、传输层、应用层),每一层都呼叫它的下一层所提供的协议来完成自己的需求。如图 3-5 所示,应用程序需要传输数据时,需要经过多次缓存复制,同时需要内核介入来完成协议处理。因此传统的 TCP/IP 技术在处理数据传输的过程中需要占用大量的服务器资源,以至于 TCP 传输效率低下。在对数据传输效率要求不高的环境尚可以接受。然而高性能计算或大数据应用是数据传输密集型应用,采用 TCP 无疑大幅限制应用的性能与可扩展性。因此系统不能以软件方式持续处理以太网通信,CPU 资源必须释放专注于应用处理。

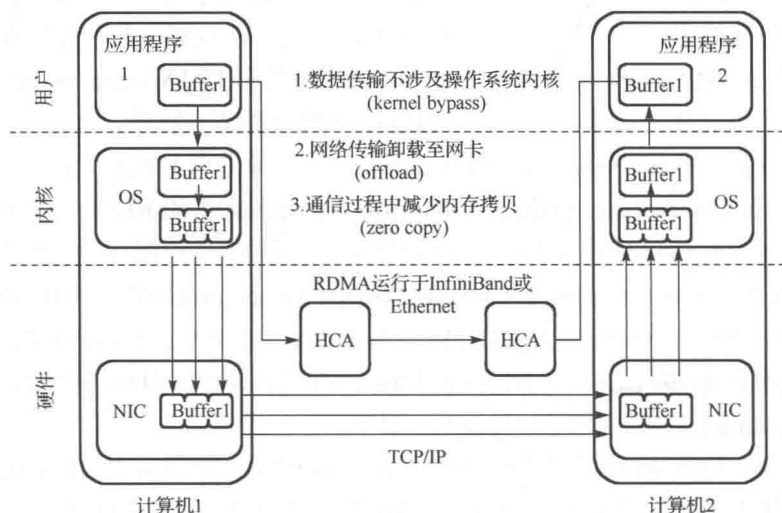


图 3-5 TCP/IP 与 RDMA 的工作原理对比

RDMA 操作使应用可以从一个远程应用的内存中读数据或向这个内存写数据。RDMA 操作用于读/写操作的远程虚拟内存地址包含在 RDMA 消息中传送, 远程应用程序要做的只是在其本地网卡中注册相应的内存缓冲区。远程节点的 CPU 在整个 RDMA 操作中并不提供服务, 因此没有带来任何负载。通过类型值 (键值) 的使用, 一个应用程序能够在远程应用程序对它进行随机访问的情况下保护它的内存。

发布 RDMA 操作的应用程序必须为它试图访问的远程内存指定正确的类型值, 远程应用程序在本地网卡中注册内存时获得这个类型值。发布 RDMA 的应用程序也必须确定远程内存地址和该内存区域的类型值。远程应用程序会将相关信息通知给发布 RDMA 的应用程序, 这些信息包括起始虚拟地址、内存大小和该内存区域的类型值。在发布 RDMA 的应用程序能够对该内存区域进行 RDMA 操作之前, 远程应用程序应将这些信息通过发送操作传送给发布 RDMA 的应用程序。

3.3.2 RDMA 的技术核心

1) 零拷贝技术

零拷贝 (zero copy) 技术, 网卡 (HCA) 可以直接与应用内存相互传输数据, 从而避免了在应用程序内存与内核内存之间的数据拷贝过程。

2) 内核旁路技术

内核旁路 (kernel bypass) 技术, 当执行 RDMA 读/写请求时, 应用程序不需要执行内核调用就可向网卡发送命令。在不需要任何内核参与的条件下, RDMA 请求从用户空间发送到本地网卡, 并通过网络发送给远程网卡, 从而减少了在处理网络传输时内核空间与用户空间之间环境切换的次数。

3) 协议卸载

网卡硬件中部署可靠的传输协议。最大化用网卡实现必要的协议处理, 从而消除通信对 CPU 资源的依赖。现在有些用软件来实现 RDMA 网络传输的技术 (如 iWARP), 因其依然基于 TCP 进行协议处理, 所以没有实现真正意义的协议卸载 (offload), 其 RDMA 性能也自然无法达到硬件 RDMA 技术 (InfiniBand RDMA 或 RoCE (RDMA over converged ethernet))。

RDMA 正在迅速成为高速集群和服务器区域网的一种基本特性。为了让更多领域使用 RDMA 互连服务器与存储, Open Fabrics Alliance 作为一家非营利机构, 致力于开发开源代码软件 OFED (open fabrics enterprise distribution) 来支持 3 大主流 RDMA 网络: InfiniBand、iWARP、RoCE。其软件包括软件驱动、内核代码、中间件、用户层接口, 并支持一系列标准协议如 IPoIB (IP over InfiniBand)、SDP (socket direct protocol)、SRP (SCSI RDMA protocol)、iSER (SCSI extension for RDMA) 与 DAPL (direct access provider library)。同时也支持其他协议, 如多种 MPI、文件系统 (Lustre、GPFS、NFS over RDMA (NFS over RDMA) 等)。该软件有 Linux、FreeBSD 和 Windows 版本。

InfiniBand 与其他网络技术最大的区别就是 InfiniBand 为应用层提供直接 RDMA 消息传输服务, 无论是用户应用程序还是内核应用。而传统网络当应用层需要访问网络服务时, 必须要通过操作系统的支持才能完成。InfiniBand 环境中 (图 3-6 所示), 应用层与 InfiniBand 传输层

交互时借助软件接口 Verbs，它是被 OFED 软件传输接口（software transport interface）的规范定义的。应用使用 POST SEND Verb 函数来请求 InfiniBand 传输层向指定通道发送消息。

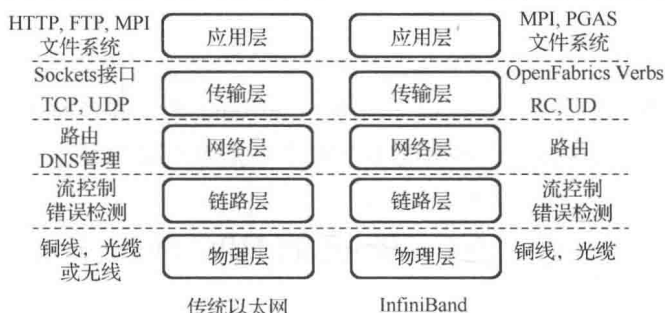


图 3-6 InfiniBand 网络层级架构

程序员通过 Verbs 设计开发的应用程序可以直接运行在 InfiniBand 网络上，并且支持 RDMA。很多高性能计算应用直接基于 Verbs 改写，从而能够最大限度地利用 InfiniBand 等支持 RDMA 技术的网络。然而直接基于 Verbs 编写，要求程序开发人员了解 Verbs 规则，知道如何创建队列对（queue pair），建立连接，通过建立的连接发送与接收消息等。为了使很多通用应用能够受益于 RDMA，而且不用学习 Verbs 更改程序，OFED 提供了一系列的 ULP（upper layer protocols），让已经存在的应用能够在 InfiniBand 网络上使用 RDMA 技术进行消息传输，而不是传统的 TCP 方式。每个 ULP 包括两大接口：面向应用的向上接口与面向 InfiniBand 传输层的向下接口。向上接口提供应用程序能够识别的标准接口如下。

（1）SRP。

通过 SRP，SCSI 文件系统能够直接以 RDMA 方式连接远端块存储，文件系统本身不需任何变动。

（2）SDP。

Socket 应用程序可以在不需要改动的情况下直接利用 InfiniBand RDMA 特性进行通信。

（3）iSER。

iSCSI 协议允许块存储文件系统通过支持 TCP/IP 的网络访问块存储，iSER 使得 iSCSI 协议能够在支持 RDMA 的网络上运行。

（4）IPoIB。

该上层协议允许应用程序在 InfiniBand 网络内部或 InfiniBand 网络与其他网络通过标准 IP 通信方式进行信息传输。IPoIB 支持 TCP/IP、UDP、SCTP 等。该协议可以保证传统应用在不经任何改动的情况下直接以 IP 网络方式运行在 InfiniBand 网络上。

（5）NFSoRDMA。

NFS 是最为广泛使用的文件系统之一，提供文件级 I/O。传统 NFS 基于 TCP/IP 网络运行。NFSoRDMA 延伸了 NFS 协议，使其能够利用 InfiniBand 的 RDMA 特性实现更高的文件系统性能。类似的文件系统支持包括 Lustre、GPFS 等。

（6）RDS。

RDS（reliable datagram sockets）由 Oracle 开发，提供 Berkeley Sockets 接口，允许消

息从一个 Socket 发送到多个目标。数据库通过 RDS 可以实现并行最大化,同时充分发挥 InfiniBand 的高带宽与低延迟特性。

(7) MPI。

MPI (message passing interface) 为高性能计算应用提供了完整 MPI 函数支持。

通过完整的 ULP,应用可以通过单一网络 InfiniBand 来完成多种服务,如存储、网络、IPC 服务等访问。数据中心可以不需要多个网络来完成不同的服务。

3.4 基于 InfiniBand 的 HPC 应用优化

MPI 规范的标准化工作是由 MPI 论坛完成的,其已经成为并行程序设计事实上的工业标准。最新的规范是 MPI3.0,基于 MPI 规范的实现软件包括 MPICH 与 OpenMPI。MPICH 由美国阿贡国家实验室和密西西比州立大学联合开发,具有很好的可移植性。MVAPICH2、Intel MPI、Platform MPI 都是基于 MPICH 开发的。OpenMPI 由多家高校、研究机构、公司共同维护的开源 MPI 实现。

在 HPC 领域,并行应用程序通常基于 MPI 开发。因此要优化 HPC 应用程序,了解 MPI 实现的特性是非常关键的。

1) MPI 通信协议

MPI 通信协议大体可以分为两类: Eager 协议与 Rendezvous 协议。

(1) Eager 协议。

该模式下发送进程将主动发送信息到接收进程,而不会考虑接收进程是否有能力接收信息。这就要求接收进程预先准备足够的缓存空间来接收发送过来的信息。Eager 协议只有非常小的启动负荷,非常适合对延迟要求高的小消息发送。Eager 协议下,可以采用 InfiniBand Send/Recv 或 RDMA 方式发送消息来实现最佳延迟。

(2) Rendezvous 协议。

和 Eager 模式相反,该模式下 Rendezvous 协议会在接收端协调缓存来接收信息。通常适用于发送比较大的消息。该情况下,发送进程自己不能确认接收进程能够有足够的缓存来接收要发送的信息,必须借助协议和接收端协调缓存之后才会发送信息。

Rendezvous 协议与 Eager 协议本身并不局限于 RDMA 操作,可以运行 Socket、RDMA Write 与 RDMA Read。然而 Socket 操作中需要多个消息拷贝过程从而大幅降低通信性能,并且无法实现计算与通信的重叠。RDMA Write 和 Read 通过零拷贝与内核旁路,实现更高的性能同时可以将计算与通信操作同步叠加运行。如图 3-7 所示,发送端首先发送 Rndz_start 控制指令到接收端,接收端随后返回另外一个控制指令 Rndz_reply,该指令包含接收端应用程序的缓存信息和访问其内存地址的 key 信息。发送端收到指令后调用 RDMA_Write 将数据直接写入接收端应用程序的缓存,消息发送完成之后,发送端会发出 Fin 指令到接收端告知自己已经将整个信息放入接收端的应用缓存中。Rendezvous 模式的好处是在没有确切得知发送消息的信息之前,没有预先的 Pre-pin 缓存,因此它是相对于 Eager 模式更节约内存的一种方式。相对负面的是其多重操作会增加通信延迟。因此更适合于传输相对占用

内存较多的大消息。如图 3-8 所示，Eager 协议在消息大小小于 16KB（在 MVAPICH2 中的默认 Eager 阈值）时都可以提供更低的通信延迟，但在消息大小大于 Eager 阈值后，Rendezvous 模式的优势开始显现。

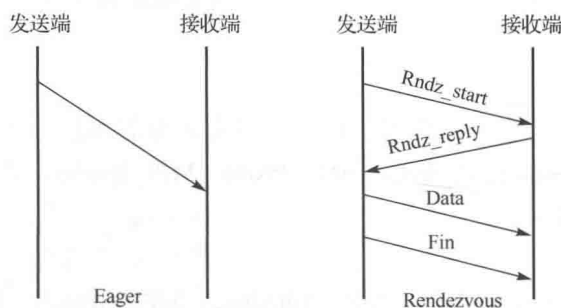


图 3-7 Eager 与 Rendezvous 通信方式

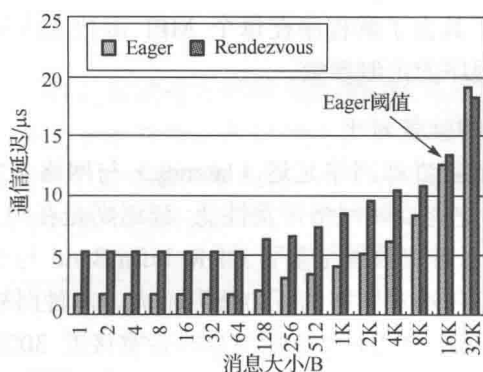


图 3-8 Eager 与 Rendezvous 性能对比

在 MPI 程序运行时，用户可以通过设置不同 Eager 阈值来改变通信特点，从而获得可能的更高应用程序性能。图 3-9 中，在改变 Eager 阈值后，延迟出现波动的转折点也随之改变。通过分析应用程序的 MPI 通信特性后，用户可以准确地设置不同阈值来测试其对应应用性能的影响。

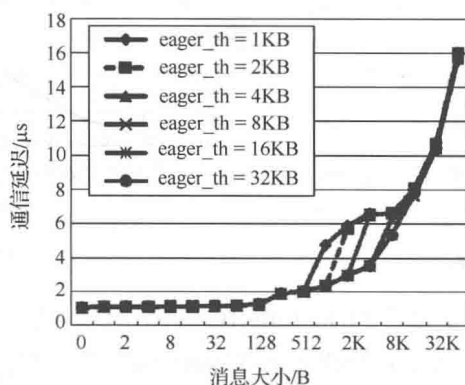


图 3-9 Eager 阈值对延迟的影响

2) MPI 函数

前面介绍的 MPI 底层通信协议会对所有 MPI 通信产生影响。具体到上层的 MPI 函数还会涉及另一层面的优化。MPI 函数分为集群 (collective) 通信与点对点 (point to point) 通信。不同的 MPI 实现对集群通信与点对点通信的实现略有差异。因此针对不同 MPI 实现所采取的优化方式也存在差异。

(1) 点对点通信。

MPI 定义了超过 35 个点对点通信函数。最为主要的包括 MPI_Send、MPI_Recv、MPI_Sendrecv、MPI_Isend、MPI_Irecv、MPI_Probe、MPI_Iprobe、MPI_Test、MPI_Testall、MPI_Wait、MPI_Waitall 等。

(2) 集群通信。

MPI_Allgather、MPI_Allgatherv、MPI_Allreduce、MPI_Alltoall、MPI_Alltoallv、MPI_Barrier、MPI_Bcast、MPI_Gather、MPI_Gatherv、MPI_Reduce、MPI_Scatter、MPI_Scatterv 等。

每个 MPI 实现针对这些函数都有类似或不同的调试参数。为了找到最合适的参数，需要用户首先运行程序分析工具去了解程序在每个 MPI 函数上占用的时间和所使用的消息大小，从而给出最适应该程序的定制参数。

3) MPI 基于不同网络的性能对比

网络最基本的两个性能特性即网络延迟 (latency) 与网络带宽 (bandwidth)。网络带宽通常是消息大于 256B 时更能反映网络带宽性能。延迟则是在消息小于 256B 时更能反映网络延迟特性。图 3-10 所示带宽数据是基于 QDR InfiniBand 与千兆以太网测试所得的。随着消息大小的增加，消息传输对网络的压力逐渐增大，两种网络带来的带宽差距也随之越来越大。QDR InfiniBand 在 2MB 大小的消息时带宽接近 3000MB/s，然而以太网只有 100MB/s。InfiniBand 相对于以太网在网络延迟上的领先优势同样巨大，如图 3-11 所示。InfiniBand 网络延迟基本在微秒级，QDR 延迟达到 1.2 μ s，但是千兆以太网的延迟高达 50 μ s，速度远低于 InfiniBand。

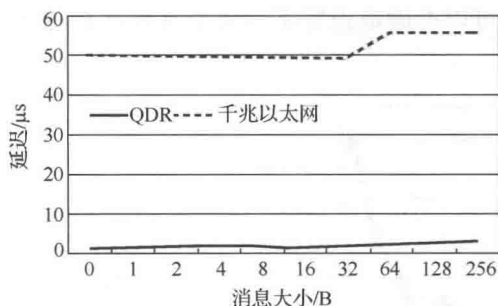


图 3-10 网络带宽图

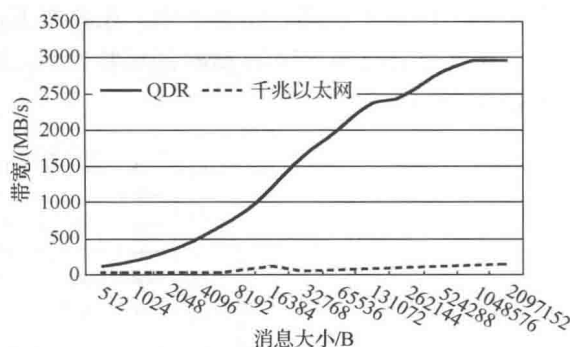


图 3-11 网络延迟

以上网络基本性能对比是基于 QDR InfiniBand 网络的，最新 FDR InfiniBand 具备更高性能。如图 3-12 和图 3-13 所示，Mellanox Connect-IB 网卡单向带宽可达 12GB/s，FDR InfiniBand 网卡单向带宽达到 6.3GB/s，网络延迟低于 1 μ s。

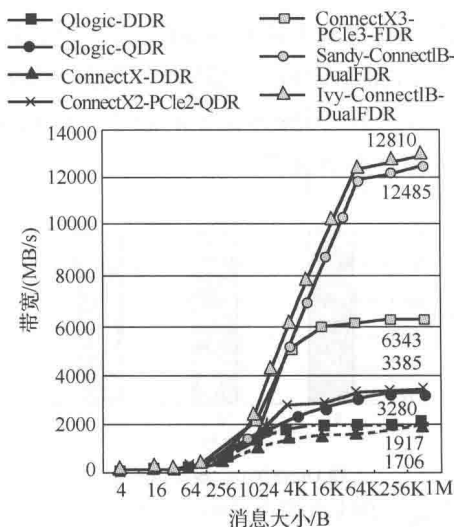


图 3-12 网络带宽

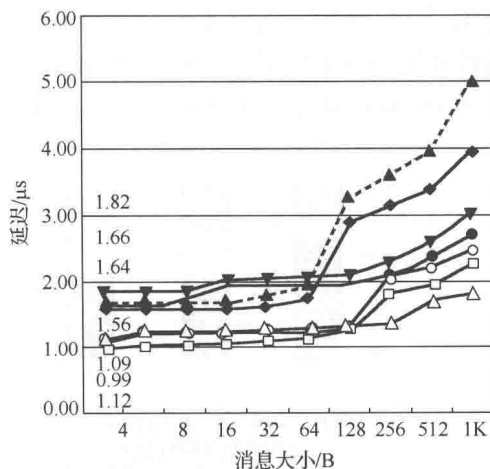


图 3-13 网络延迟

应用程序性能如下。

网络基本带宽与延迟的性能测试报告证明 InfiniBand 具备更高的带宽与更低的网络延迟。其能否为应用程序带来更高的性能是衡量 HPC 系统效率的最终标准。图 3-14 给出了应用程序 NAMD 在两种网络上的直接性能对比。NAMD 是最为流行的分子动力学模拟软件之一，Apoal 是该程序的标准性能测试算例。

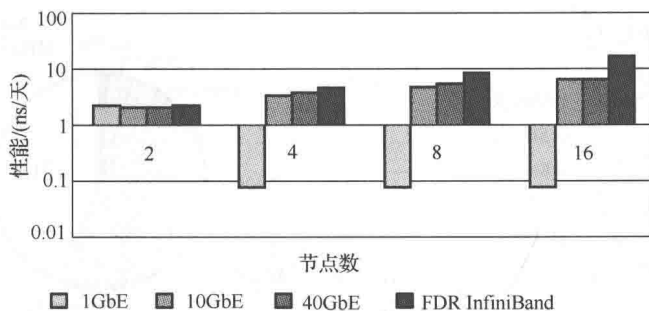


图 3-14 NAMD 性能测试 (Apoal)

性能结果显示从两台服务器开始，InfiniBand 就可以提供比以太网更高的性能。当在 8 个服务器节点时，InfiniBand 能够提供双倍于以太网的性能，随着节点数的增加，InfiniBand 相对于以太网的优势进一步扩大，在 16 节点时，基于 InfiniBand 的 NAMD 性能是以太网性能的 1.5 倍。从 4 个节点开始，基于以太网的 NAMD 性能增长就非常缓慢。万兆以太网与 4 万兆以太网提供相同的 NAMD 性能，其性能高于千兆以太网，但相对 InfiniBand 性能远远落后。要充分发挥 HPC 系统的性能与效率，InfiniBand 网络是不可替代的核心技术。

4) MPI 应用优化

不同的 MPI 实现有不同的特性分析工具，如 OpenMPI 有 IPM (integrated performance monitoring)，Allinea 可以分析多种 MPI。本书中的案例是运行 IPM 分析 MPI 应用。

首先通过 IPM 可以分析 MPI 总体运行时间在整个程序运行时间中的占比。图 3-15 以 NAMD 为例, 相同的服务器运行在千兆以太网时, MPI 通信时间占比随着节点数增加而逐步超越计算所占用时间。而运行在 InfiniBand 网络时, 通信时间占比相对较小。这直观地解释了 InfiniBand 能够为应用程序 NAMD 提供更高性能的原因。

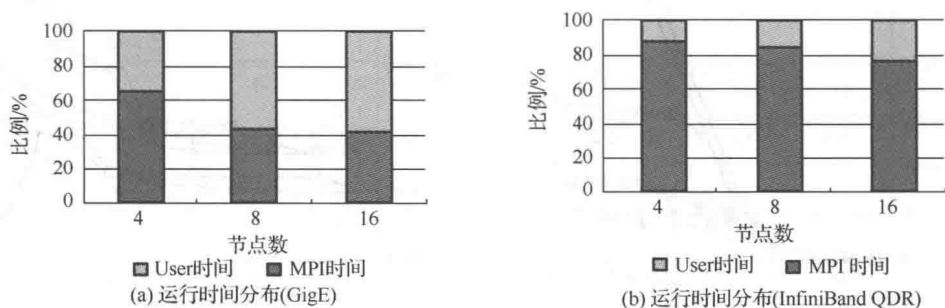


图 3-15 NAMD 运行时间分析

即使 InfiniBand 能够为 NAMD 提供非常好的性能, 但是更加细致的 MPI 优化可以进一步提升其性能与可扩展性。为了有的放矢地优化具体 MPI 函数, 需要分析不同 MPI 函数在通信时间中各自的占比。如图 3-16 所示, 在 NAMD 程序运行时, MPI_Recv 与 MPI_Iprobe 占用时间比例最大, 而且随着进程数增加, MPI_Iprobe 逐渐超越 MPI_Recv 成为占用时间最长的 MPI 函数。因此在 MPI 函数优化时要首要考虑优化这两个函数。

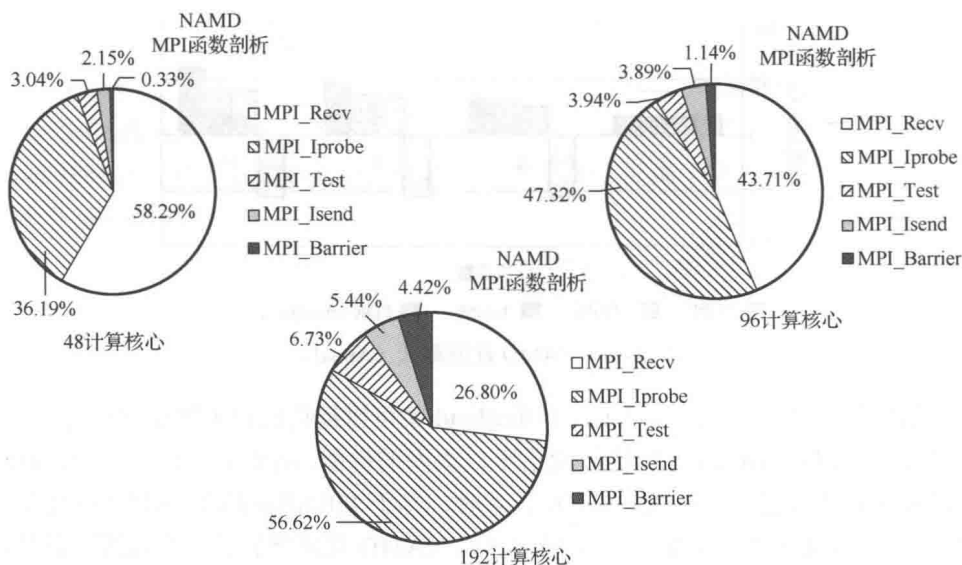


图 3-16 NAMD MPI 通信时间分析

要给出准确的 MPI 优化参数, 除了分析每个函数总体占用的时间, 还需要知道每个函数所传递消息的大小。图 3-17 通过 IPM 分析得知传输的消息主要集中在 512B~64KB 和 256KB~2MB。因此在 MPI 优化时可以以上述分析结论为依据对 MPI_Recv 接收中大型消息和 MPI_Iprobe 为优化重点。

MPI_Recv 与 MPI_Iprobe 都是点对点通信, 影响其通信时间最为重要的 MPI 协议即 Rendezvous 和 Eager 参数。经过调试, 提高到 VIADEV_RENDEZVOUS_THRESHOLD= 50000B 时 NAMD 可以在 16 节点时提升性能 11% (图 3-18)。当小于该消息大小时, MPI 发送消息采用 Eager 模式从而大幅降低通信延迟, 进而缩短整个程序的运行时间。当然该参数的数值设置要和自身系统配置相结合。当采用大量 Eager 运行通信时, 它会事先占用更多内存来保证接收到远端发送来的信息。在系统内存不充裕的情况下, 一味增加 Eager 使用量可能会很快造成整体程序运行速度下降。

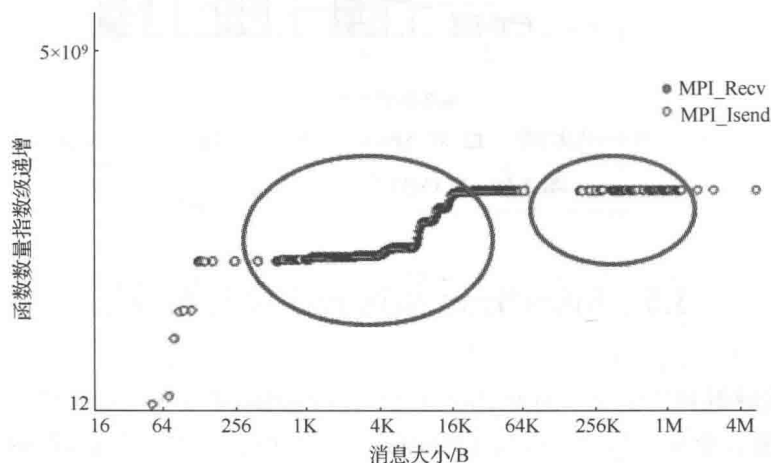


图 3-17 NAMD MPI 消息分布

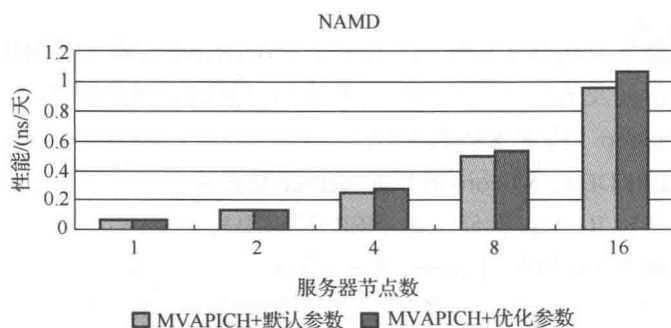


图 3-18 NAMD MPI 函数优化

根据以上原理, 采用 IPM 来分析应用程序 MPI 的通信特点, 然后针对相应函数和相应消息大小进行定制优化, 对应用性能提升有直接帮助。当然有些应用未必采用 MPI 来编写, 这是需要参考不同优化技巧的。如图 3-19 所示, 直接采用 InfiniBand Verbs 开发的 NAMD 版本比 MPI 版本性能提升 10% 左右。

在 HPC 领域, 基于网络互连技术还有众多需要探索的领域, 如运行 MPI 应用时调用扩展可信连接(XRC)或不可靠数据报(UD)模式进行点对点通信。InfiniBand 公司 Mellanox 针对大规模 MPI 应用开发了免费的开源加速软件消息加速器 (mellanox messaging accelerator, MXM) 与网络加速器 (fabric collective accelerator, FCA) 来分别加速点对点

通信与集群通信，通过更多硬件智能卸载来进一步加大应用程序在集群环境中的可扩展性与性能。

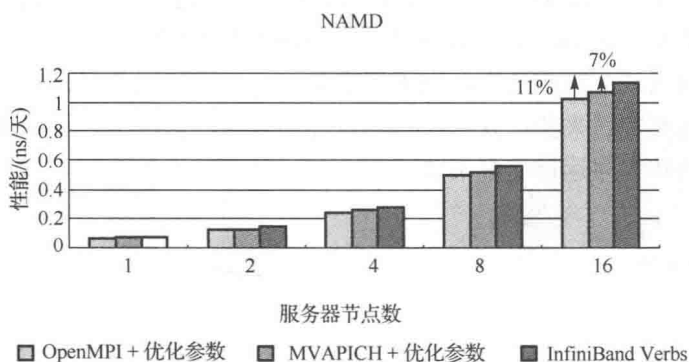


图 3-19 NAMD 多版本性能对比

3.5 InfiniBand 加速云计算与大数据

在 HPC 领域得到广泛应用的 RDMA 网络如 InfiniBand 技术已经延伸到更多领域，如大数据、云计算、企业应用等。高速 InfiniBand 交换机提供纳秒级网络延迟。以太网交换芯片在传输小数据包的情况下性能很难发挥，不可能实现线速。然而 InfiniBand 交换机即使小数据包也可以实现线速。InfiniBand 网卡延迟低于 $0.7\mu\text{s}$ ，是普通以太网网卡延迟的 $1/10 \sim 1/5$ 。

除了网络硬件能够直接带来的低延迟优势，InfiniBand 内置支持 RDMA，该技术可以大幅提升云计算中所涉及的一系列应用（有些应用性能可以数倍提升），详见如下。

- (1) 大数据 Hadoop（内置支持 RDMA）。
- (2) Oracle、IBM DB2、Microsoft SQL Server 数据库。
- (3) 块存储（支持 RDMA 存储协议 iSER 与 SRP）。
- (4) 并行文件系统（GPFS、Lustre、Ceph 等）。
- (5) 行业应用如金融、期货、动漫等，都可实现性能大幅提升。

即使某些应用程序本身不支持 RDMA，使用支持 RDMA 的底层存储或文件系统同样也会为应用程序带来大幅性能提升。即便整个系统没有 RDMA 支持的软件或存储，InfiniBand 上运行传统以太网 TCP 应用性能依然可以超越万兆以太网。

1) InfiniBand 网络具备更好的可扩展性

InfiniBand 是软件定义的扁平网络，具备极佳的可扩展性。可以很容易地构建几万节点的集群，然而这对以太网来说是极具挑战性的。同时 InfiniBand 为应用程序提供最高可扩展性。在并行程序越来越成为主流的大数据时代。对应用可扩展性的影响无疑是新系统设计必须要考虑的因素。这也是先进网络最为重要的一个因素。InfiniBand 拥有拥塞控制、服务质量（quality of service, QoS）机制、传输协议硬件卸载（释放 CPU 资源）、

支持硬件虚拟化 SR-IOV (Single Root-IO Virtualization) 等特性。这些特性直接提升了应用可扩展性。

2) InfiniBand 具备更佳的投资性价比

系统实际性价比 = (100% - 网络在总成本占比) × 系统整体效率, 网络在整个系统成本中占比如表 3-1 所示。

表 3-1 网络在整个系统成本中占比

网络在整个系统 成本中占比	系统效率			
	60%	70%	80%	90%
10%	0.54	0.63	0.72	0.81
20%	0.48	0.56	0.64	0.72
30%	0.42	0.49	0.56	0.63

以实际数据为例, 假设万兆以太网成本在整个系统的采购成本中占 10%, 由其构建的系统整体效率为 60% (基于 Top500 系统效率排名), 其系统性价比为 $(1-0.1) \times 0.6 = 0.54$ 。

如果采用最先进 InfiniBand, 其采购成本约占总成本的 20%, 然而其系统整体效率可达 90%, 其系统性价比为 $(1-0.2) \times 0.9 = 0.72$ 。由此可见高性能 InfiniBand 网络即使采购成本高于万兆以太网, 但其整体系统带来的投资回报率却远高于万兆以太网系统。

3) InfiniBand 拥有完整的云计算平台软件支持

InfiniBand 网络支持所有主流虚拟化软件, 如 VMware、OpenStack (KVM、Xen)、Microsoft Hyper-V。在主流虚拟化平台上, 实现网卡的高性能虚拟化。同时实现多网融合, 在云计算环境里, 再不需要 FC 网络。多网融合架构如图 3-20 所示。

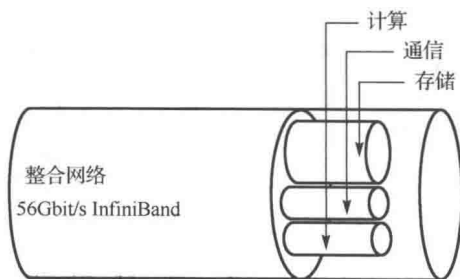


图 3-20 多网融合架构

单一 InfiniBand 网络可以满足计算、通信、管理、存储需求, 从而实现更简单的云网络, 降低整体网络采购与管理成本。

全世界有越来越多的系统开始使用 InfiniBand 构建云计算平台。得益于 InfiniBand 网络能够带来的低能耗、高应用程序性能、高系统扩展性、最高投资性价比、简单网络架构等优秀特性, InfiniBand 开始从高性能计算领域快步挺进云计算与大数据市场。Oracle 基于 InfiniBand 公有云现支持全球 760 万用户、5000 家 PaaS 客户, 仅 2013 年第四季就增加

500 个大型 SaaS 客户。每天承载着 165 亿次交互。通过 InfiniBand 连接 90PB 存储。微软公有云平台内部通过 InfiniBand 实现应用程序的更高性能与可扩展性。更快的 CPU、内存、存储、更多的分布式大数据应用，都需要支持 RDMA 的高速网络。IBM 在 SoftLayer 上推出 InfiniBand 网络。IBM 此举使得自己在网速上领先于其他如亚马逊和谷歌等对手。以满足那些需要计算密集型或者通信密集型应用的用户，并且具有更好的性价比。在云供应商开始提供 InfiniBand 服务后，新一类的应用程序会相继出现，这些新程序将充分利用云的敏捷性和成本效益，同时也会进一步加速云的需求和使用。

InfiniBand 网络不是仅局限于高性能计算的网络，其简单的网络架构、最高的网络性能、最低功耗、最佳可扩展性，以及越来越完整的生态链支持，使得 InfiniBand 逐渐成为现代云计算中心的最佳网络选择。

第 4 课 超算系统的应用环境

从第一台计算机面世到现在,已经超过五十年了,计算机行业经历了数次技术革新。从最初的批处理计算机,到现在基于 SMP、MPP、NUMA、集群等架构的并行处理系统,计算机系统已经发生了很大变化。随着多核 CPU 的普及,并行应用逐渐成为主流应用,因此如何搭建并行应用环境,成了亟待解决的问题。

如今主流的超级计算机系统并行应用环境分为两大类:共享存储的集群系统和分布式存储的集群系统。这里的“存储”指的是内存而非外存储器。共享存储的集群系统是由一台配置较高的多 CPU 组成的,CPU 间共享内存,并通过内部总线互相连接。分布式存储的集群系统是由多台同构或异构的计算机节点组成的,通过网络连接起来协同完成特定任务。由于分布式存储的集群系统拥有很高的性价比,因此得到普遍应用。

从软件方面来说,主要分为操作系统和并行环境两个方面。

操作系统分为 UNIX 系统和 Windows 系统两大类,其中 UNIX 系统占到绝大多数。而 UNIX 系统中,Linux 系统的操作系统占到其中的绝大多数。Linux 系统有几点优势。

(1) 廉价。大多数 Linux 发行版采用免费使用、收费服务的形式,可以按需消费,减少大量软件成本。

(2) 自由开放。自由开放是 Linux 系统最著名的一个特点。Linux 的开发者从寥寥几个人发展到全世界数千万用户群,其扩张速度令人叹为观止。开放源代码为 Linux 提供了广阔的发展空间。使全球的用户为其提供大量的测试,使得 Linux 的系统漏洞可以被及时发现。另一方面,全球的开发者可以充分发挥自己的聪明才智,为 Linux 系统添砖加瓦,使得 Linux 的性能越来越高、越来越适合使用。而对于高性能开发工程师来说,开放源代码意味着工程师可以完全了解自己的程序是如何在操作系统中运作的,并依据这样的观察,做出针对性的调优,提高程序性能。甚至可以修改操作系统,以便更加契合硬件及应用。

(3) 高效。Linux 由于内核与图形界面的分离,以及设计之初的目的等原因,使得稳定性比其他系统要好很多。且 Linux 为单内核结构,使得 Linux 内核的执行效率比微内核操作系统要高。因此,Linux 系统非常适合高性能应用。而广大用户的选择也证明了这一点,在世界上最权威的 Top500 高性能集群排行榜上,全球最快的 500 台集群当中,使用 Linux 系统的占 90%以上。这已经能充分证明 Linux 作为高性能计算集群操作系统的实力。

在并行环境方面,MPI 是目前最广泛使用的并行环境,它具有移植性好、功能强大、效率高等诸多优点。而且有多种不同的免费、高效、实用的实现版本,几乎所有的并行计算机厂商和操作系统都对其提供支持,这是其他并行环境所无法比拟的。MPI 于 1994 年诞生,虽然产生的时间比较晚,但它吸收了其他多种并行环境的优点,同时兼顾性能、功能、移植性等特点,且赶上了并行计算发展的黄金时期,在短短几年内便迅速普及,成为消息传递并行环境的标准。

随着超算应用的增多和应用对计算量需求的增大,CPU 逐渐在某些应用领域显露疲

态。为解决这个问题,异构计算应运而生。异构是指与传统 CPU 不同架构的计算设备,如 GPU、MIC 等新型计算设备。这些新的计算设备通常拥有远高于 CPU 的并行计算能力。GPU 原本并非用于高性能计算,而是用于图形显示。但是 GPU 所特有的硬件架构,天然适合高并发度的计算。但利用 GPU 编程,需要将通用问题转化为图形显示问题,因此编程门槛高,很少有人利用 GPU 进行高性能计算并行应用开发。

2007 年 NVIDIA 公司推出 CUDA 架构的 GPU 体系解决方案。从硬件层面上,对 GPU 进行了高性能计算方向的优化,使之更加适合通用计算。从软件层面上,提供了同名的 CUDA 语言,以类似于 C 语言的方式,为开发者提供了简单易用的使用 GPU 进行高性能计算并行应用的方法。

2012 年底,Intel 公司正式发布基于 MIC 架构的 Intel Xeon Phi 系列协处理器, MIC 架构采用 x86 指令集,使得程序源代码可以无缝移植到 MIC 架构,编程方面采用引语方式,以对源代码改动尽量少的方式快速移植。

在此之前,异构开发环境也有先进精简指令机(advanced RISC machine, ARM)、现场可编程门阵列(field programmable gate array, FPGA)等非 x86 架构环境,但都有使用复杂、与现有 x86 架构环境差异较大、无法结合等缺点。自从可以使用 GPU 进行高性能计算并行应用以来,现有计算资源得到了保留,并且可以很容易地大幅扩充并行计算能力,且编程简易,使得异构计算的方式得到了真正的普及和接受。

异构环境并非抛弃 CPU,而是 CPU 与新的计算硬件相结合,并行进行计算。无论是 GPU 还是 MIC,都是以 PCI-E 接口与现有 x86 节点进行连接,以协处理器的方式与现有节点进行集成。因此可以在不破坏现有集群的情况下,增加一部分计算能力,得到了业界的认可,采用异构集群的高性能计算集群在 Top500 中越来越多,表明异构集群方案越来越多地被业界采用。

虽然在硬件方面,异构集群与传统集群相比,并没有革命性的变化,通常可以在现有集群的基础上在节点内增加异构计算卡的方式,将传统同构集群转换为异构集群。但在软件方面,可能会有较大的改变。由于使用异构硬件的方法并不相同,因此需要针对 CPU 和 GPU/MIC 使用不同的软件环境和编写相应的代码。更重要的是,因为异构集群中计算设备的结构不同,所以需要针对新情况,使用新方法,解决新问题。针对新的 GPU/MIC 设备,将不能使用传统的 CPU 编程思想,而是转而使用高并行性的 GPU/MIC 编程思想。这一思想的转变,有时比方式的转变更难以接受。但是,只有当编程思想从传统的串行思维转变为异构集群并行思维以后,才能充分利用异构硬件, CPU+GPU、CPU+MIC 协同计算,才能达到最好的性能。

4.1 CPU 并行系统应用环境

4.1.1 硬件环境

正如前面所述,最常用的传统并行应用环境为分布式存储的封闭式集群系统。集群由多台计算节点构成。每台计算节点都是共享内存的计算系统。一般来说,集群中的每个节

点都使用相同的配置，以方便管理和性能的优化。节点一般为标准机架式服务器，配有双路或四路的多核服务器版 CPU（如 Intel[®] Xeon[™]系列），每核搭配 4GB 或更多内存。节点之间采用以太网线（千兆网线或万兆网线）或 InfiniBand 网线，通过相应的交换机互相结合。节点与节点属于同一个内网，并各自拥有不重复的内网 IP 地址。通常集群与外网物理隔离，以保证集群的安全性。

具体硬件环境搭建的方法与细节，请参考前面的相关章节，这里不再赘述。

4.1.2 软件环境

1. 操作系统

正如前面所述，高性能计算并行应用的操作系统一般选用 Linux 操作系统。其中 RedHat Linux 发行版以其稳定而著称，因此在商业集群中得到广泛应用。但是，RedHat Linux 由于需要一定的授权费用以进行维护和售后服务，导致一些对成本极其敏感的客户，放弃了 RedHat Linux 发行版。RedHat Linux 的一个再发行版是 CentOS，使用 CentOS 可以在免费的情况下，获得与 RedHat Linux 类似的体验。但 CentOS 并不是官方支持的，因此可能在补丁升级等方面与官方的 RedHat 有一定差距。另外，CentOS 显然也不能获得 RedHat 官方的售后服务。除了 RedHat Linux，另一大商业发行版是 Novell 公司的 SLES（SUSE Linux enterprise server），二者都是 Linux 系统，且都需要付费使用。需要注意的是，高性能计算并行应用集群中使用的都是服务器版的操作系统，这类版本的操作系统对服务器应用进行了针对性的优化，使得其性能、稳定性都最适合高性能计算使用。而一些桌面版的操作系统由于面向个人用户，导致在桌面、显示等方面进行了一定优化，而在网络连接、提供系统服务等方面有所欠缺。另外，在注重安全的领域中，有时会采用基于 UNIX 的 FreeBSD 操作系统，此操作系统是以安全性作为最重要的考量而设计的。

2. 并行运行环境

由于上述高性能计算并行应用环境的硬件环境构建设计，采用的是分布式内存结构，因此并行软件环境需要采用消息传递的网络通信方式。MPI 是使用最普遍的并行编程环境，其优点在于编程简易、可移植性强。MPICH 是 MPI 标准的一种最重要的实现，MPICH 的开发主要是由美国阿贡国家实验室（Argonne National Laboratory）和美国密西西比州立大学（Mississippi State University）共同完成的，它是 MPI 的一种具体实现，该实现可以免费从网上下载。

MPICH 并行编程环境的安装步骤简述如下。

（1）解压、安装。首先从网上下载安装包，根据后缀名，使用相应的命令解压，一般解压命令是 tar。进入 MPI 的解压目录下，使用 ./configure; make 命令安装。

（2）放权。为了能够在多个不同的机器上运行 MPI 程序，首先需要其他机器对启动 MPI 程序的机器放权，即允许启动 MPI 程序的机器能够访问其他的机器。为了简单起见，最好在集群的每一个节点机上都建立相同的账户名，使得 MPI 程序在相同的账户下运行。在放权设置完成以后，可以在任何一个节点，使用同样的账户名，不需要输入密码。这样可以在任何一个节点启动 MPI 程序，并且在不需要输入密码的情况下可以自动运行程序。

(3) 运行。程序在不同节点可以放置在不同目录下,但为了方便管理,建议放置在同样的目录下,放置在共享存储时最佳。使用哪些节点、每个节点内开启几个 MPI 进程,是由配置文件决定的,每个应用可以采用不同的节点配置文件,以便根据不同的应用选择不同的方案,避免闲置计算资源或达不到最好的利用效果。

在节点内,由于是共享内存模式的并行系统,一方面仍然可以选择 MPI 软件环境进行并行,另一方面也可以选择 OpenMP 或 PThread 方式进行并行。PTHread 是一种多线程库,也是 Linux 下的基础多线程库,在其他操作系统中也有 PThread 库的移植。OpenMP 基于 PThread,底层由 PThread 库实现。OpenMP 以在源代码中插入编译制导语句的方式,屏蔽了并行程序开发的底层细节,能够以最简便的方式将程序并行化。但是 OpenMP 不像 PThread 库能够控制一些底层的细节,因此在某些情况下,性能可能会略逊于 PThread。

4.1.3 开发环境

传统高性能计算并行应用的软硬件构建应该基于应用环境,即采用与应用环境相同的架构、软硬件环境和规模。理想情况下,开发平台即应用平台,但在实际应用中,往往需要在一个平台上开发,并应用于多个不同的平台。所以在开发环境的选择上,一般选择架构、软硬件环境与应用环境尽量相同,但规模上可能小于应用环境。但是由于规模不同,性能上的表现也可能出现不同,所以在开发测试时,应尽量模拟应用平台的情况进行测试,以免出现开发时性能很好,但是在实际部署中效果不佳的情况。

开发环境中,包含与应用环境相同的软件环境,并且包括编译器、调试器等软件开发工具和高性能数学库、调优工具等软件优化工具。

1. 编译器和调试器

编译器一般使用 Intel 公司的 Compiler XE 编译器套装,包含 C/C++、Fortran 等语言的编译器(需要分别购买)。一般来说,Intel 公司出品的编译器编译出的应用程序性能比较好,尤其当应用程序运行于 Intel 的 CPU 的平台下时。但是,Intel 公司的编译器需要付费购买使用,因此限制了部分注重成本过于注重性能的用户使用。GNU/GCC 是属于 GNU 的免费编译器套装,包含 C/C++、Fortran 和其他语言的编译器。GNU 编译器在各个操作系统和平台上都有相应的实现,适用范围较广,尤其在 Linux 系统中,已成为默认的编译器。在 Linux 系统中,默认都安装有 GNU 编译器。

Linux 系统中,包含对 PThread 库的运行环境支持。各个常用的编译器,都支持 OpenMP 的语法和 PThread 的函数声明,并且自带 OpenMP 和 PThread 的运行库。因此在安装编译器以后,即可支持 OpenMP 和 PThread 库。

MPI 的运行库和编译环境包含在 MPICH 的安装包中。在 MPICH 的编译方式中,可以选择使用其他编译器,如使用 Intel 公司的编译器,如果不进行指定,则默认使用 GNU 编译器进行编译。

调试器一般使用免费的 GNU/GDB,或 Intel 公司的 IDB,或 TotalView 公司的 TotalView Debugger 调试器。

GDB 通常附带于 Linux 系统中,并与系统紧密集成。其优点在于适用性广、支持广泛,

由于是字符界面的调试器，所以十分适合远程调试和在没有图形界面的环境下调试。GDB 支持常用的设置断点、查看代码、查看堆栈、附着线程等应用，是功能强大的调试器，广泛用于各种环境下的调试。

IDB 是 Intel 公司推出的调试器，是商业软件。IDB 同样拥有命令行界面的调试工具，并同时拥有 Java 支持的图形界面调试工具，因此相对 GDB 来说使用更加简易、学习门槛低。IDB 全面兼容 GDB 的命令，因此对 GDB 用户来说，改变调试工具不会带来多少学习成本。IDB 除了图形界面以外，也对多线程调试等 GDB 比较薄弱的方面进行了改进，因此成为了一个进阶的选择。

TotalView 是一个为开发人员提供对应用程序进行复杂控制的图形调试器。它是一款资源级功能全面的工具，含有使大型或复杂程序开发简单化和使用信息检验、多线程、OpenMP 和其他并行编程技术的程序提供高级支持的功能，这些并行编程技术使程序可以运行于分布式存储器和共享存储器的多处理器计算机。TotalView 拥有广泛和完善的 C/C++ 调试功能集。TotalView 的调试器优势在于强大的图形化界面和对多进程，尤其是并行集群系统的支持，因此对调试并行应用非常有利。

2. 高性能数学函数库

现有的数学函数库可以简化应用程序开发，避免“重复造车轮”的过程，并且使程序简洁，对绝大多数情况来说，使用现成的数学库的性能比程序员自己编写源代码的性能高，因为现成的数学库经过了大量的开发、深度优化和测试，比自己手工写的程序性能好，且可靠性高。因此，如果在程序中用到一些经典算法，一般使用现有的高性能数学库，可以提高代码的工程性，且提高程序性能。

Intel 公司有一套核心数学库，即 Math Kernel Library，简称 MKL 数学库。Intel 数学核心函数库 (Intel MKL) 提供经过高度优化和大量线程化处理的数学例程，面向性能要求极高的科学、工程和金融等领域的应用。Intel MKL 为 Intel C++ 和 Fortran 编译器专业版和 Intel 集群工具套件的组件，也可作为单独产品使用。Intel MKL 针对当前多核 x86 平台进行了深入而全面的优化，如充分利用了 CPU 的向量化指令。MKL 数学库支持 C 语言和 Fortran 的接口，面向多核 CPU，包含线程安全的特性和在多核上的优秀扩展性，MKL 的函数会自动执行处理器检测，并选择调用最适合该处理器的二进制代码。Intel 的 MKL 数学库包含线性代数数学库，即 BLAS 和 LAPACK、ScaLAPACK、稀疏矩阵解算器，快速傅里叶变换 (FFT)，矢量数学库，即幂函数、三角函数、指数函数、双曲函数、对数函数等，随机数生成器等。同样，使用 Intel 的 MKL 数学库需要付费，但分发其运行库则不需要支付额外费用。

FFTW (faster Fourier transform in the west) 是一个快速计算离散傅里叶变换的标准 C 语言程序集，其由 MIT 的 Frigo 和 Johnson 开发。可计算一维或多维实数据或复数据以及任意规模的离散傅里叶变换 (DFT)。FFTW 还包含对共享和分布式存储系统的并行变换，它可自动适应计算节点，充分利用缓存、存储器大小、寄存器个数。FFTW 通常比目前其他开源傅里叶变换程序都要快。FFTW 的编码生成器采用面向对象设计技术和面向对象的语言 CAML 编写；它能自动适应系统硬件，因而可移植性很强。FFTW 为任意大小的模式生成一个计划 (plan)，通过对该计划施行各种运算完成各种模式的转换；内部结构及其复

杂性对用户透明；速度快（适合各种机器的内部编译器、代码生成器利用 AST 在运行时生成代码并自我优化，而且不占用编译时间，采用分层存储技术）。FFTW 受到越来越多的科学研究和工程计算工作者的普遍青睐，并为量子物理、光谱分析、音视频流信号处理、石油勘探、地震预报、天气预报、概率论、编码理论、医学断层诊断等领域提供切实可行的大规模 FFT 计算。

基础线性代数程序集（basic linear algebra subprograms, BLAS）是一个 API 标准，用以规范发布基础线性代数操作的数值库（如矢量或矩阵乘法）。该程序集最初发布于 1979 年，并用于建立更大的数值程序包（如 LAPACK）。在高性能计算领域，BLAS 被广泛使用。例如，LINPACK 的运算成绩则很大程度上取决于 BLAS 中子程序 DGEMM 的表现。为提高性能，各软硬件厂商则针对其产品对 BLAS 接口实现进行高度优化。

LAPACK (linear algebra package) 是以 Fortran 编程语言写就，用于数值计算的函式集。LAPACK 提供了丰富的工具函式，可用于如解多元线性方程式、线性系统方程组的最小平解、计算特征向量、用于计算矩阵 QR 分解的 Householder 转换、奇异值分解等问题。在 NetLib 也提供了 API 经简化的 Fortran 95 版本的 LAPACK95。LAPACK 以 BSD 授权的方法释出。

以上 FFTW、BLAS 和 LAPACK 都可以免费使用，且性能优于同类免费的库，因此得到了广泛的应用。

3. 高性能调优工具

高性能调优工具是指通过对已有程序的采样分析，找出现有程序的瓶颈，协助开发人员进行性能调优的工具。由于高性能计算并行应用的程序通常十分复杂，且运行时间很长，所以如何能快速、准确地找到程序热点，并据此进行改进，是性能优化人员长期面临的重要问题。凭借性能优化人员的经验和对源代码的理解推测热点的方式，极端依赖于性能优化人员的个人经验，并且极不准确。通过人工采样的方式测试程序，可以比较客观地反映真实性能情况，但费时费力且容易遗漏，并且需要对代码进行修改。采用高性能调优工具则可以避免这些问题，通过对源代码或者二进制文件进行插桩，可以在不修改源代码的情况下对程序进行准确、客观、全面的测试，并找出真正的热点。性能优化人员可以根据测试结果进行分析，并进行有针对性的性能优化工作。

gprof 是 GNU 的性能调试工具，包含在 Linux 的开发调试套装中。gprof 通过在编译时增加“-pg”的编译选项，自动地在函数调用时进行“插桩”。可以在程序运行后输出每个函数的运行时间和调用关系，给性能优化带来极大方便。打印出程序运行中各个函数消耗的时间，可以帮助程序员找出众多函数中耗时最多的函数。产生程序运行时的函数调用关系，包括调用次数，可以帮助程序员分析程序的运行流程。

Intel VTune 性能分析器是商业软件，甚至不需要重新编译，只需通过 VTune 重新运行程序，即可自动获取程序的各种运行信息。除了对函数耗时进行采样以外，VTune 也可以进行代码行级别的采样，甚至汇编语句级别的采样，并且 VTune 可以对不同的硬件事件进行采样，因此可以进行更有针对性的分析测试工作。除此之外，VTune 可以更好地支持多线程和多进程的分析测试工作，因此对高性能计算并行应用来说显得尤为适合。

4.2 CPU+MIC 异构并行系统应用环境

4.2.1 硬件环境

正如前面所述，异构环境中 GPU 和 MIC 是以附加板卡的形式附着于现有系统的，因此在硬件环境上，不需要对集群的整体结构进行任何改变，只需要在节点内部增加协处理器或将节点更换为可以使用协处理器的节点。

由于 GPU 和 MIC 在硬件环境构建中，对集群结构没有区别，因此以 MIC 为例说明异构系统的硬件环境，异构并行应用环境架构如图 4-1 所示。

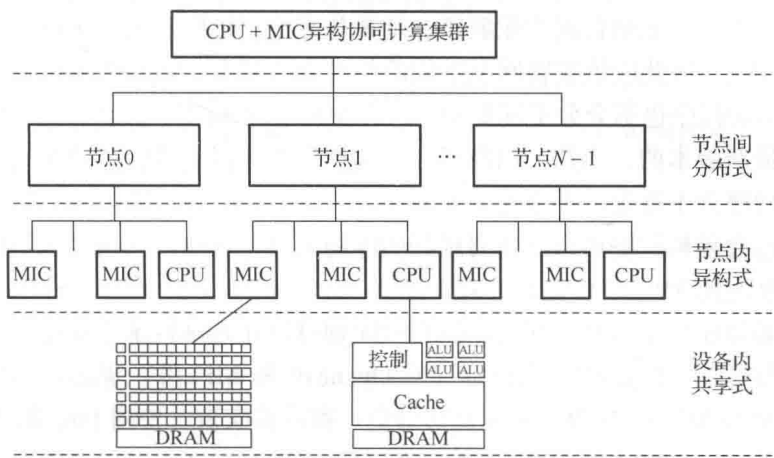


图 4-1 异构并行应用环境架构

使用异构协处理器除了搭建传统高效能并行应用环境需要注意的几个问题以外，还有三点特别之处需要注意。

(1) 传统并行应用环境中的节点，不一定有多余或合适的 PCI-E 的插槽，也就不一定能够附加 GPU 或 MIC 卡。但是异构并行开发环境中，其中的节点必须是传统并行环境中拥有 PCI-E 接口，可以附加 GPU 或 MIC 卡的类型。

(2) 由于增加了协处理器，所以节点的功耗需求大大增加，通常每增加一块协处理器，需要增加 300W 左右的电源供应。

(3) 由于增加了协处理器，所以对节点的散热提出了更高的要求。在搭建异构集群时，不仅需要看节点是否能够支持协处理器，还要注意节点的散热能力。

更详细的细节请参考之前的相关章节。

4.2.2 软件环境

在异构并行开发环境的构建中，由于环境是异构的，因此包含传统的并行开发环境和支持新硬件的开发环境两部分。这二者都是并行程序开发不可或缺的组成部分，其中传统环境对应传统部分的程序，同时作为专用环境的基础。而专用环境针对新硬件开发的软件驱动，构建于传统环境之上。

1. 通用软件环境

通用软件环境与传统并行应用软件环境极其类似，不仅同样可以分为操作系统、并行环境等方面，而且异构模式下的通用软件环境，也基本与传统软件环境相同。

在并行环境方面，一般需要选择 MPI、OpenMP 等通用的并行库，因为异构环境的硬件显然需要支持尽量通用的、常用的并行库。而 InfiniBand 作为高性能领域常用的网络互连方式，现有的异构环境可以支持 InfiniBand 硬件，因此通用软件环境也可以根据硬件需要，增加支持 InfiniBand 的驱动。

2. MIC 专用软件环境

MIC 专用软件环境包含驱动程序和运行时库等部分。其中驱动程序为 MIC 卡的驱动程序，其作用为在软件和硬件间产生联系，使操作系统可以顺利地利用硬件。驱动程序与具体硬件密切相关，因此应该在官网上下载硬件对应的版本。MIC 卡的硬件版本现在还比较单一，因此驱动程序也不会分不同类型。因为 MIC 技术推出时间还不长，所以驱动程序应该尽量选择最新版本的。当然，同样要注意根据需要使用正式版本的驱动程序，因为测试版本的驱动程序并不稳定，不适合工业应用。

运行时库包含异构程序运行时需要用到的库的集合，这些库调用了驱动程序的一些接口，使程序能够运用 MIC 硬件进行计算。

驱动程序必须在每个节点上安装，运行时库则可在共享目录中安装一份。

在并行运行环境方面，MIC 支持标准的 OpenMP 和 MPI 库，但是，如果想充分利用 MIC 的特性，如将 MIC 卡视为一个独立的节点，则需要安装、使用 Intel 的 MPI 库。

4.2.3 开发环境

异构高性能计算并行应用的软硬件构建应该基于应用环境构建，即采用与应用环境相同的架构、软硬件环境和规模，且原则与传统并行环境构建的原则基本相同。但异构开发环境的硬件规模通常由于成本限制无法达到应用环境的规模，应保证节点内配置一致，以免对开发造成太大影响。

开发环境中，包含与应用环境相同的软件环境，并且包括编译器、调试器等软件开发工具。异构开发环境，由于硬件差异较大，通常会有与硬件配套的专用编程语言与接口，而编译器和调试器也会有相应的版本和支持。为了尽量利用异构硬件的计算能力，通常会使用针对异构硬件的高性能数学函数库，并利用针对异构硬件的性能调优工具进行程序优化。

1. 编程语言与接口

1) 通用编程语言与接口

为了提高易用性与兼容性，异构开发环境通常使用已有的编程语言，并在其上加入自己专用的扩展，而不是重新创建一门新的编程语言。因此，一般的协处理器支持常见的如 C、C++、Fortran 等语言。

为了进一步方便用户使用，有些组织开发了通用的编程接口，使编程人员用相同的代

码控制不同的异构设备。其中开放运算语言（open computing language, OpenCL）是使用最广泛的异构编程接口。

OpenCL 是第一个面向异构系统通用目的并行编程的开放式、免费标准，也是一个统一的编程环境，便于软件开发人员为高性能计算服务器、桌面计算系统、手持设备编写高效轻便的代码，而且广泛适用于多核心处理器、GPU、Cell 类型架构和数字信号处理器等其他并行处理器，在游戏、娱乐、科研、医疗等各种领域都有广阔的发展前景。OpenCL 由一门用于编写 kernels（在 OpenCL 设备上运行的函数）的语言（基于 C99）和一组用于定义并控制硬件平台的 API 组成。OpenCL 提供了基于任务分割和数据分割的并行计算机制。

OpenCL 支持 GPU 和 MIC 协处理器，因此可以通用于异构集群当中。程序员可以使用统一的代码，不理睬硬件细节，尽可能地利用异构硬件的性能。

2) MIC 专用编程语言与接口

MIC 支持 C、C++ 和 Fortran，并以类似 OpenMP 的引语方式，对前述语言进行扩展。同样，MIC 的专用接口可以充分利用 MIC 卡的硬件资源，提升程序的执行效率。

2. 编译器和调试器

MIC 的编译器一般使用 Intel 公司的 Compiler XE 编译器套装，包含 C/C++、Fortran 等语言的编译器（需要分别购买）。需要注意，Intel 公司的编译器，支持 MIC 的版本与仅支持 CPU 版本的名称没有区别，因此需要选择能够支持 MIC 的编译器版本（即 2013 版和以后的版本）。Intel 现已将 MIC 的驱动程序和操作系统等部分开源，因此未来可能出现第三方的免费编译器，如果用户对软件成本极其敏感，可以选择未来可能出现的免费版本编译器。

MIC 的编译器虽然同样分为编译 MIC 程序和编译 CPU 程序的两个版本，但由于 Intel 公司将二者集成于同一个软件包当中，所以程序员可以不加区别地加以调用，并且可以不必安装 GCC 编译器。

IDB 是 Intel 公司推出的调试器。IDB 同样拥有命令行界面的调试工具，并同时拥有 Java 支持的图形界面调试工具，因此相对 GDB 来说使用更加简易，学习门槛低。IDB 全面兼容 GDB 的命令。IDB 支持对 MIC 内核的调试，是调试 MIC 程序的首选。

3. 高性能数学函数库

MIC 高性能数学库是基于 MIC 架构的一组高性能数学库，支持使用 MIC 进行高性能数学运算。由于 MIC 与 Intel 的 CPU 一脉相承，所以 Intel 旗下的 MKL 数学库也可以用于 MIC。

Intel 的 MKL 数学库包含线性代数数学库，即 BLAS 和 LAPACK、ScaLAPACK、稀疏矩阵解算器，快速傅里叶变换，矢量数学库，即幂函数、三角函数、指数函数、双曲函数、对数函数等，随机数生成器等。

MIC 版本的高性能数学库可以手工调用，在源代码中设置何时使用 MIC 版本的数学库，也可以在 MIC 函数中直接使用这些数学库的函数，还可以在系统环境变量中设置，使大计算量的算例可以使用 MIC 版本的计算函数，而在小规模的情况下，仍然使用 CPU 版本的计算函数，甚至可以选择 CPU 和 MIC 协同计算的方式进行运算，这是通过事先在环境变量中设定、CPU 和 MIC 各自的工作量比例确定的。

MIC 系统 x86 的架构和开源的特性,使得将来一定会有越来越多的高性能数学库应用于 MIC 之上,也为用户带来更多选择。

4. 高性能调优工具

由于 MIC 是基于 x86 的指令架构,且卡上有基于 Linux 的操作系统,因此绝大部分 Intel 性能调试工具都可以应用于 MIC 中。Intel VTune 性能分析器是商业软件,甚至不需要重新编译,只需通过 VTune 重新运行程序,即可自动获取程序的各种运行信息。除了函数耗时以外,VTune 也可以进行代码行级别的采样,甚至汇编语句级别的采样。并且 VTune 可以对不同的硬件事件进行采样,因此可以进行更有针对性的分析测试工作。最新版本 VTune 完全支持 MIC 的硬件事件采样,因此可以很好地应用于 MIC 程序的性能调优中。

4.2.4 资源

1) IDF

英特尔信息技术峰会(Intel developer forum, IDF)是由 Intel 公司主办的技术讲座,在美国、中国等 7 个地区举办,每年分春、秋举办两次。IDF 主要由主题演讲、技术专题讲座和技术展示组成,主题演讲的演讲者均是 Intel 的高层人士,演讲的题目都具有相当的前瞻性,Intel 作为一家在处理器、网络处理器等领域处于领先地位的公司,IDF 的确是让业界获悉 Intel 最新动向的最佳场合。可以通过以下网址搜索相关信息:<http://www.intel.com/content/www/us/en/library/viewmore.results.html?prTag=rauthorship:inteldeveloperforum>。

2) MIC 计算论坛

论坛是时效性最高的 MIC 技术互动平台,如果无法找到答案可以在此进行交流,在此获得众多领域的专家协助:<https://software.intel.com/en-us/forums/intel-many-inte-grated-core>。

更多资源可以访问 <https://software.intel.com/en-us/mic-developer>。

4.3 CPU+GPU 异构并行系统应用环境

4.3.1 硬件环境

搭建 GPU 异构超算平台与搭建 MIC 平台并无本质区别,请参考前面的相关章节。

4.3.2 软件环境

GPU 专用软件环境,根据生产厂商的不同,软件环境也有所不同。但一般都需要硬件驱动、运行时库等基本的几部分。本章将以 CUDA 为例,讨论相关问题。

CUDA 的软件环境分为驱动程序和运行时库两部分。其中驱动程序为 GPU 卡的驱动程序,其作用为在软件和硬件间产生联系,使操作系统可以顺利地利用硬件。驱动程序与具体硬件密切相关,因此应该在官网上下载硬件对应的版本。驱动程序并非越新越好,因为过新的驱动通常对最新的硬件进行了较好的支持,而对较旧的硬件的支持有所下降。但是一般来说,应该选用比较新的、正式版的驱动程序。因为测试版的驱动程序虽然会对某些性能进行优化,但是由于测试版并不稳定,不适合工业应用。

运行时库包含异构程序运行时需要用到的库的集合，这些库调用了驱动程序的一些接口，使程序能够运用 GPU 硬件进行计算。

需要特别注意的是，驱动程序必须在每个节点上安装，因为每个节点都有 GPU 卡，所以需要在每个节点的操作系统上配置驱动程序。而对于运行时库来说，仅在应用程序需要时才会用到，因此可以在共享目录中安装一份，以供全部节点使用。这种方式的好处是，减少安装和维护的工作量，可以统一管理，一次配置，多次使用。但需要注意对该目录下文件写权限的控制，以避免用户无意识地破坏文件，造成全局软件故障。

4.3.3 开发环境

通过前面的介绍，我们了解了基于 GPU 异构计算程序的软硬件执行环境。本节将概括性地对开发环境进行介绍，意在让读者可以将本书当成一个索引，在 GPU 程序开发的过程中灵活选择各种语言、实用工具和库，做到游刃有余。

1. 语言和库

从 2007 年发展至今，由于行业涵盖广泛，不同应用对于 GPU 软件精度、速度、开发周期的要求不同，且程序员背景多样，掌握编程语言和能力有所不同，为广泛适应市场需要，针对各个不同的市场划分，NVIDIA 及其合作伙伴共同开发了多种多样的编程方式，有 CUDA C/C++、CUDA Fortran、OpenACC、HMPP、CUDA-x86、OpenCL、JCuda、PyCUDA、Direct Compute、MATLAB、Microsoft C++ AMP 等。目的是让所有有 GPU 编程需求的开发者都能找到适合自己和自己项目的语言工具。

多线程编程语言可以分为两类，一种为编译指导，即在串行程序之前添加编译指令，指导编译器将串行指令自动编译为并程序；另一种为显式线程模型，用户可以直接编写并程序，显式地调用 GPU 的线程。相对地，编译指导比较简单，有易于初学者的学习、方便串行程序并行化、开发周期短等优点。但是，因为编译器自动完成串行程序的并行化，所以对 GPU 的操作相对不够灵活。

在下面介绍的语言中，OpenACC 与 HMPP 为编译指导语言，它们与 CPU 程序并行语言 OpenMP 十分相似。而 CUDA Fortran 同时支持编译指导和显式线程模型，其他语言为显式线程模型。其中 CUDA-x86 比较特别，它用于将 CUDA C/C++写成的程序直接编译成为 CPU 多线程程序，不需要修改代码。表 4-1 详细展示了各种语言的情况。

表 4-1 GPU 编程语言

编程语言	支持语言	开发者	分 类
CUDA C/C++	C/C++	NVIDIA	显式线程模型
CUDA Fortran	Fortran	PGI	显式线程模型/编译指导
OpenACC	C/C++ Fortran	PGI/CAPS/GREY/NVIDIA	编译指导
HMPP	C/C++ Fortran	CAPS	编译指导
CUDA-x86	CUDA C	PGI	显式线程模型
OpenCL	C/C++	Khronos	显式线程模型
Direct Compute	C/C++集成于 Direct3D 之中	Microsoft	显式线程模型
Microsoft C++ AMP	C/C++	Microsoft	显式线程模型

续表

编程语言	支持语言	开发者	分 类
JCuda	Java	Personal	显式线程模型
PyCUDA	Python	Personal	显式线程模型

在支持多种语言的同时，为了提升开发者的开发效率，各合作伙伴、NVIDIA 协同各领域开发者，将常用算法进行模块化，形成了一批行业涵盖广泛、功能强大、效率高、富有针对性的库，开发者可以直接调用，不必自己重新开发。

(1) AmgX: 提供了对核心求解器加速的简单方案，对于计算密集的线性求解器有着 10x 的加速，非常适合隐式非结构方法。

(2) cuDNN: 专为深度神经网络设计的 GPU 加速函数库，可以被快速地加入高层机器学习框架中。

(3) cuFFT: 实现了基于 GPU 的快速傅里叶变换，并提供简单接口供开发者使用。

(4) cuBLAS-XT: 基于 cuBLAS 为 Level 3 BLAS 提供了多 GPU 加速支持，不需要开发者自己配置并调度多 GPU 资源。

(5) NPP: 包括了上千个图像处理和信号处理函数，功能强大。

(6) CULA Tools: EM Photonics 提供的线性代数库，为复杂的数学计算提供 GPU 加速。

(7) MAGMA: 支持最新 LAPACK 和 BLAS 标准的线性代数库，为异构计算进行特殊优化。

(8) IMSL Fortran Numerical Library: RogueWave 开发的基于 GPU 的数学和统计函数库，涵盖广泛的内容。

(9) cuRAND: 提供高速的 GPU 加速的随机数生成器。

(10) ArrayFire: 包罗万象的 GPU 函数库，涵盖数学信号处理、图像处理、统计学等领域，支持 C/C++、Java、R 和 Fortran。

(11) cuBLAS: GPU 加速的 BLAS 函数库，提供了相比 MKL BLAS 6x~17x 的加速。

(12) cuSPARSE: 对应 cuBLAS 提供了对稀疏矩阵计算的加速。

(13) Thrust: 提供了高速的并行算法的开源库，用于计算 GPU 加速的排序、扫描、变换和规约等常用算法。

(14) NVBIO: 生物领域，提供了 GPU 加速的高吞吐序列分析，同时支持长短读校准 (long and short read alignment)。

另外除了以上提到的这些，还有提供水体海洋实时模拟的 Triton Ocean SDK、支持 GPU 软硬编解码的 NVIDIA Video Codec SDK、基于 R 开发环境的线性数学库 HiPLAR、计算机视觉领域通用性极强的 OpenCV、计算几何学引擎 GPP、稀疏矩阵迭代求解器 Paralution 等。

2. 开发工具综述

开发工具一般可以归纳为三类：编译工具、调试工具和性能分析工具。下面主要介绍调试工具和性能分析工具，最后介绍集成开发环境 Nsight。

1) 调试工具

(1) CUDA-GDB。

它是 NVIDIA 随 CUDA 开发套件发布的一款基于 GDB 命令行的调试工具。同时支持 CPU、GPU 调试。如果开发者以前有使用 GDB 的经验将可以很快上手。

(2) CUDA-Memcheck。

与 CUDA-GDB 一起可以快速定位 GPU 函数中的内存访问错误,解决程序中 unspecified launch failure 的错误。

(3) Allinea DDT。

支持 GPU 集群和单机,并同时支持 MPI、OpenMP 和 CUDA 程序的调试工具。

(4) TotalView。

基于图形界面的调试工具,可以支持多进程(线程)的 MPI、OpenMP 和 GPU 调试。

2) 分析工具

所有的 CUDA 程序分析工具都是基于 CUPTI (CUDA profiling tools interface) 而开发的,但所支持的程序和分析内容各不相同。其中 NVIDIA Visual Profiler 可以为开发者优化 CUDA C/C++ 程序提供可视化的反馈数据和基本的优化建议。支持 Linux、Mac OS X 和 Windows 平台。目前已经支持远程单节点分析。

NVPROF 是命令行界面下的 CUDA C/C++ 分析工具,可以将结果输出到 Visual Profiler 进行显示。允许通过用户自己编写的脚本实现灵活的功能。

另外众多商业软件如 TAU Performance System、VampirTrace、The PAPI CUDA Component 可以同时支持 CUDA、PyCUDA 和 HMPP 的混合并行程序。相对于 Visual Profiler 功能更加丰富,并支持多节点分析。有助于开发者对程序和集群总体情况有更好的把握。

3) 集成开发环境 Nsight

Nsight 是 NVIDIA 开发的一套集成了编译、调试与性能分析功能的开发环境。目前有两个版本,一个是 Windows 平台下的 Nsight Visual Studio Edition,另外一个是基于 Mac OS 和 Linux 的 Nsight Eclipse Edition。可以从以下网站下载:<http://www.nvidia.com/object/nsight.html>。

(1) 开发。

提供了新的开发模板,并集成了 CUDA SDK Sample,使开发者可以快速熟悉并开始编写 CUDA 应用程序。

提供了代码高亮、补全和导航功能。

(2) 调试。

可以同时进行 CPU GPU 部分的调试工作,使用方法类似。

支持远程调试。

包含了 CUDA-Memcheck,用于检测内存错误。

(3) 分析。

提供了类似于 Visual Profiler 的界面和功能支持各项性能数据的分析,自动产生优化建议。

并且在 Eclipse 版中支持热点分析,可以在代码中自动标注 GPU 函数中的热点语句。

4.3.4 资源

1) CUDA Zone

CUDA Zone 提供了一站式的 GPU 程序开发指导和下载, 涵盖了从入门到开发工具, 到学习资源各方面的信息, 并实时更新最新的 GPU 计算领域的新闻, 如图 4-2 所示。适合于所有 GPU 程序开发者使用: <https://developer.nvidia.com/cuda-zone>。

2) GTC

每年的 GPU 技术大会 (简称 GTC) 是 GPU 计算领域最大规模也是最重量级的技术交流活动。来自世界各地的各领域科学家、工程师、研究员将在此进行学术报告, 交流成功经验。参与者可以在这里获知最新的行业动态和成果。每年的会议报告将被制成录像, 可以通过以下网址访问: <http://on-demand-gtc.gputechconf.com/gtcnew/on-demand-gtc.php>。

3) GPU 计算论坛

论坛是时效性最高的 GPU 技术互动平台, 如果无法找到答案可以在此进行交流, 在此获得众多各领域的专家协助: <https://devtalk.nvidia.com/default/board/53/gpu-computing>。

更多资源可以访问: <https://devtalk.nvidia.com/default/board/53/gpu-computing>。



图 4-2 CUDA Zone 的一站式 GPU 程序开发指导平台

第 5 课 超算系统性能评测方法

5.1 超算系统性能评测现状

5.1.1 系统性能测评的意义和挑战

对高性能计算机或者超级计算机而言，强大的计算性能是其最主要的价值和优势所在。如何评价一台高性能计算机系统的性能，是用户、厂家和研究机构普遍关心的问题。高性能计算机的系统性能测评，在计算机的研制、选型、引进和有效使用等方面，都是不可缺少的一项重要工作。下列的常见问题都可以或都需要通过系统性能测评来解决。

研制的计算机是否达到设计目标？

研制的计算机和应用要求的差距在哪里，如何改进？

硬件升级（CPU、内存、协处理器或其他）能给性能带来多大的提升？

不同架构的高性能计算机哪一种更适合自己的应用需求？

是否有必要为高性能计算机配备加速计算部件？

A 厂商的产品与 B 厂商的类似产品在性能上有何不同？

在采购高性能计算机预算固定的条件下，应该如何获得最佳的性能？

.....

一台计算机系统的性能是硬件、软件等诸多因素综合作用的结果。这里的硬件包括了处理器、内存、存储设备、网络设备、系统总线以及近年来大行其道的协处理器（加速计算设备）等；而软件则主要包括操作系统、编译系统、编程语言、并行通信库、数学库等系统软件。由于测评关注的是系统本身的性能状况，所以这里所述的软件通常不包括实际的应用软件。和普通的个人计算机、工作站或服务器相比，高性能计算机系统无论在硬件上还是软件上都更为庞大、复杂，要想准确地测评计算机性能，给出类似上述问题的解答并不容易。

从体系结构看，除了以出色的性价比占据主流的集群系统外，传统的 MPP 系统依然以其不可替代的架构和性能优势占据一席之地，且 Cray、IBM、Fujitsu 等主流 MPP 厂商的不同产品又可以细分为 SMP、CC-NUMA、向量机等不同种类。不同体系结构的计算机在性能表现方面有着先天的区别。

从计算部件看，当前主流的 CPU 包括了 Intel/AMD x86_64 处理器、IBM 的 Power 处理器、富士通的 SPARC 64 处理器和 ARM 处理器等主要几类，而且即便是同一类型的处理器（如 Intel x86_64 位处理器），不同型号的产品在芯片架构、工艺、核数、缓存、指令集等方面又有或多或少的差异，从而导致了其性能的差异。除了 CPU 外，近几年以 NVIDIA GPU 和 Intel Xeon Phi 为代表的协处理器也在众多高性能计算系统中得到了广泛应用，这又大大增加了考量高性能计算机性能的复杂性。

从系统软件看，操作系统、编译系统、通信库（包括底层协议和上层通信库）、核心数学库等均对计算性能有着明显的影响。例如，Linux 操作系统因为其可定制性和可裁剪性成为高性能计算机的首选操作系统，但仍有少量采用其他操作系统如 UNIX、Windows 的计算系统，甚至现在还有少量基于虚拟机构建的高性能计算系统，不同操作系统有着不同的性能表现（尽管通常差异不大）；又如核心数学库 BLAS 有着 GotoBLAS、OpenBLAS、ATLAS、MKL 等众多版本，在不同系统上性能各有优势。

任何一台计算机系统的计算性能最终都是服务于某个或者某些特定的应用软件的，不同的应用软件通常有着不同的性能表现；即便是同一软件，其计算性能经常也会因计算机硬件、系统软件或者运行参数、运行环境等的不同而不同。

上述高性能计算机系统性能测评的复杂性使得人们很难找到统一的规则或标准去评测所有的计算机。相应地，系统性能测评的途径、方法、所采用的软件就呈现出多样性，并且随着计算机软硬件技术发展和各类用户需求的变化而相应地更新、发展。例如，近年来，由于高性能计算机的规模越来越大（最多可达数百万核），系统的功耗、稳定性和可靠性成为热点问题，这就要求性能测评不仅关心计算性能，还应该关注获得特定计算性能时的功耗和该性能可稳定持续的时间。

高性能计算机或超级计算机，一直被称为计算机领域王冠上的明珠，这一方面是因为其超卓的性能和不菲的价格，另一方面是高性能计算系统通常体现当前最先进的计算技术，并在系统规模和复杂性上远超普通计算机系统。这既使得性能测评对高性能计算系统极其重要，又给测评工作带来了巨大的挑战。

5.1.2 系统性能测评的度量和主要内容

计算机系统的性能通常涵盖了计算性能、存储（I/O）性能、通信性能等多个方面，而对高性能计算机而言，最重要的无疑是计算性能。

由于绝大多数高性能计算机是用来处理浮点计算问题的，所以当前最常用的性能度量是“每秒浮点运算次数”（floating operations per second），这里的浮点操作是指加、减、乘等基本运算。

对一台给定的高性能计算机系统，其理论的峰值性能是固定的。以典型的集群系统为例，其系统理论峰值性能为 CPU 主频 $\times$ CPU 单核每个时钟周期能执行的浮点操作数 $\times$ 系统总 CPU 核数。

考虑如下配置的某个集群系统。

处理器型号：Intel Xeon E5 2692 V2@2.2 GHz，12 核。

单节点处理器数：2。

系统总计算节点数：16。

Intel Xeon E5 处理器单核每个时钟周期能执行 8 个浮点计算，那么这套集群系统的理论峰值性能为 $2.2 \times 8 \times (12 \times 2 \times 16) = 6758.4 \text{ Gflops} = 6.7584 \text{ Tflops}$ 。

这里的 Tflops 是 Tera-flops 的简写，Tera 是计算机中所采用的 Kilo、Mega、Giga、Tera、Peta、Exa 千进位序列中的一级。目前普通的高性能计算机已经是数十或数百 Tflops 量级，

少数最先进的系统则达到 Pflops 量级；百亿亿次（E 级）计算机的研制则是当前众多国内外厂商和研发机构的共同目标。

当计算机系统中配备有 GPU、Intel Xeon Phi 或其他加速卡时，计算理论峰值性能时只需要将 CPU 峰值性能和加速卡峰值性能累加即可。

从上面的计算公式可以看到，一台计算机系统的理论峰值性能只显示在计算机的 CPU 或加速计算部件（协处理器）上，并未体现出和系统的其他硬件如内存、互连网络、I/O 设备等的关系。显然，这和通常所经历和体会的系统性能状况不符：内存大、网络快的系统通常总是会比内存小、网络慢的系统有性能优势。正因为此，性能的理论峰值只是一台计算机系统不能超越的理论性能，其真实的性能还与其他因素有关，只能通过测评获得。

计算机的性能测评是通过在计算机上运行特定的软件来获得该系统的性能参数。除了上面提及的最主要的计算性能 flops 外，还可以包括其他多个项，如系统网络带宽、网络延迟、内存带宽等单项性能，用户也可以自定义度量标准（如单位时间内完成某个任务的次数）等。但无论如何度量，应用都是测评的准则。

绝大多数测评会产生一个简单的指标，其中有的指标是越大越好，如带宽、内存或存储容量等；另外一些指标是越小越好，如运行时间、内存使用量、延迟等。一个好的测评通常具备如下的特点。

- （1）易运行：对测评人、测评设备和测评环境要求不苛刻。
- （2）可重复：这是一个基本的条件，否则测评没有意义。
- （3）易理解：结果可以方便地获得，并很容易被人理解（如输出 flops 一个数值）。
- （4）代表性：能代表一大类实际应用情形，或者很接近系统的实际性能表现。
- （5）可扩展性：测试可在不同并行规模系统上进行。
- （6）可引用：不存在授权问题，结果应可以对外展示。
- （7）透明性：最好有能公开的源程序。
- （8）适当的运行时间：测评时间不能过长。

依据测试内容、测试目标或测试方法的不同，高性能计算机的性能测评通常可分为如下四大类。

1) 单项测评

通常又称为微基准测试（micro-benchmark），对计算机特定部件或单项性能进行测试，常见的包括内存带宽测试、网络性能测试、I/O 性能测试等。其主要用途包括检验系统部件或性能参数是否符合设计及研制要求、排查系统故障、对不同部件（如网络设备）进行横向比较等。

2) 整体计算性能测评

对计算机的整体计算性能进行测评，反映计算机一般情况下能获得的计算性能。常见的测试包括 HPL（high performance LINPACK）、HPCC（high performance computing challenge）、SPEC HPC（standard performance evaluation corporation high performance computing）等。整体测评通常和系统理论峰值关联密切，但理论峰值可能和真实性能有较大差异，这里的测评值是计算机运行特定软件后真实取得的结果，已经能基本反映系统的性能特性。目前国际和国内的超级计算机排行榜都是基于整体计算性能测评结果进行排序的。

3) 领域应用性能测评

由于不同应用在同一计算机系统上通常会呈现较大的性能差异,整体计算性能测评结果常常会与实际应用情况不相符,某些情况下甚至会大相径庭。而同一行业或领域内常用的应用软件、计算模型或者实现算法通常有限,基于此构建测评软件包,能很好地体现计算机系统在该领域应用的性能状况。美国国家航空航天局(NASA)的NAS并行基准测试程序(NAS parallel benchmark, NPB)、中国北京应用物理与计算数学研究所的IAPCM Benchmarks是这类测试的主要代表。

4) 典型应用性能测评

任何一台高性能计算系统都是为了运行某个或某些应用软件,人们所有对性能的考量最终都是期望在运行目标应用软件时能获得最佳的计算性能。因此对特定的用户而言,基于应用软件构建测评软件用于系统性能测评,是性能测评的最佳选择。但如前面所述,一个好的测评需要满足一定的条件,并非将目标软件在待测系统上运行几次就称为性能测评,从实际应用软件到用于性能测评的典型应用,仍需要做不少工作,如准备标准测试数据集、调整应用软件计算参数、制定计算结果正确性检验准则以及设计易于解读和比较的计算结果输出等。

5.1.3 系统性能测评步骤及方法

系统性能测评工作并没有一致的标准或者流程,但通常可分为测试前准备、性能测试和测试后结果整理分析三个步骤。

1. 测试前准备

测试前准备主要包括系统准备、配置记录、拟定测试计划等几项主要工作。

系统准备工作包括待测试的软硬件系统、运行场地、运行环境等各个方面,以保障测试工作顺利进行。配置记录是指详细记录此次测试的软硬件配置情况,通常可包括(但不限于)如下内容。

1) 硬件

(1) 节点型号及节点数。

(2) 互联网络:厂商、型号、主要配置、带宽、延迟。

(3) 节点信息。

① 主要 BIOS 设置。

② CPU: 数量、型号、主频、步进 (stepping)。

③ 内存: 类型、容量 (数量、每条容量)、速度、延迟。

④ 硬盘: 数量、型号、容量。

⑤ 加速卡: 型号、数量。

⑥ 机箱电源: 型号、功率。

⑦ 机箱风扇: 数量、位置、功率。

2) 软件

(1) 操作系统: 型号、版本、主要设置。

(2) 编译系统: 型号、版本、主要编译参数。

(3) 测试软件：版本、所运行的工作负载、使用的哪几个处理器。

(4) 开源软件：编译器的版本、编译选项，使用的软件库的版本等。

测试计划则是指事先定义好此次测试的操作过程，并以表格或其他方式准备好测试结果的记录，使得对整个测试心中有数，测试进程一目了然。表 5-1 即是在用 Iometer 进行存储性能测试时事先准备好的测试计划，测试时每完成一项，填写一项，这样既使得测试工作有序，又对整体进展有很好的把握。

表 5-1 测试计划

访问模式	块大小	IOPS	带宽	延时	CPU 利用率
顺序读	4KB				
	16KB				
	64KB				
	512KB				
顺序写	4KB				
	16KB				
	64KB				
	512KB				
随机读	同上，依据块大小分列各项				
随机写					
随机读写混合					

2. 性能测试

运行测试软件时，最重要的是保证测试过程的稳定和可靠。通常可以遵循如下一些原则。

(1) 最小系统原则，即尽可能关闭不需要的服务、不运行非必要的程序、不启动图形界面。

(2) 排他性原则，只运行测试程序，不运行其他程序，更不与他人共享系统使用。

(3) 不干扰原则，即测试期间不进行其他操作。

(4) 重复性原则，多次运行程序，取稳定时的（最佳）结果。

(5) 单一调整原则，如需修改程序运行参数，一次只更改一个。

3. 测试后结果整理分析

测评结果收集后，首先需确认结果的可靠性，即同一测试每次运行结果差别是否在容许误差范围之内（一般<5%）。确认结果可信后，需要对测试结果进行处理，通常是以图表的方式对结果数据进行呈现。

最后是对结果进行分析，看是否达到预期的目标；如果没有达到，则需要查找原因并加以改进。

5.2 高性能计算系统性能测评现状及方法

上节介绍了测评主要可分为单项测评、整体计算性能测评、领域应用性能测评和典型应用性能测评四大类，这四类均是当前常用的测评方法，并因测试目的的不同在实际应用时各有侧重。

本节将就主要的测试方法和相应的基准测试程序进行介绍，其中单项测评部分介绍内存性能测试程序 Stream 和网络通信测试程序 OMB (OSU micro-benchmarks)；整体计算性能测评部分重点介绍 HPL 和 HPCC；行业应用性能测评部分介绍 NPB、IAPCM Benchmarks 和 Graph 500 Benchmarks。典型应用测试因用户具体应用千差万别，故不单独阐述。

基于性能测评结果的高性能计算系统排名一直为人们所关注，本节最后将介绍 Top500、Top100、Green500 等几个主要排行榜。

5.2.1 主要测试程序及应用

1. 内存性能测试程序 Stream

随着处理器处理核心数量的增多，内存带宽对于提升整个系统的性能越发重要。如果系统不能足够迅速地将内存中的数据传输到处理器中，那么若干处理核心就会因等待内存数据传输而处于闲置状态。这其中所产生的闲置时间不仅会降低系统的效率，同时还会抵消多核心和高主频所带来的性能提升。由美国弗吉尼亚大学开发的 Stream 是当前最为流行的内存带宽测试工具之一。

Stream 有 C、Fortran 语言两个版本，同时还提供 MPI 版本，其测试结果统一以 MB/s 来衡量，反映系统持续内存带宽大小。

它的运行非常简单，单线程串行版本在目标系统上正确编译后直接运行即可。若是运行并行版本（多线程、OpenMP 或 MPI），只需要参照同类并程序进行编译和运行（如 MPI 版本需要预先在系统上配好 MPI 库，使用 mpif77 编译 stream_mpi.f 文件，使用 mpirun 运行程序）。

需要注意的是，尽管综合性能测试程序集 HPCC 中采用了 MPI 版的 Stream，但在实际应用中，由于多节点系统的访存性能通常取决于网络，因此内存带宽测试对单机更为重要，即常用的是 Stream 的单机版和多线程版（包括 PThread 和 OpenMP）。

最新版的 Stream 可从<http://www.cs.virginia.edu/stream>下载获得，该地址还给出了当前 Stream 测试的 Top20 系统，当前排名第一的是 SGI 公司的大型共享内存计算机 Altix UV2000。

2. 通信性能测试程序 OMB

众多节点通过网络互连的集群系统是当前高性能计算机的主流形态，因此通信性能对高性能计算机至关重要。美国俄亥俄州立大学的 OMB 是业内最为知名的通信性能测试工具之一。

OMB 当前支持 MPI、UPC 和 OpenSHMEM 三种通信模型，同时最新版本还提供了对 CUDA 和 OpenACC 的支持。其中最典型和最常用的无疑是 MPI 通信性能测试。

OMB 提供包括点对点通信、集合通信和单边通信在内的丰富测试，并且每个通信类型又提供延迟、带宽、多线程延迟、多线程带宽等多个输出。测试者可以对通信数据大小进行设置（如延迟测试使用 1B、4B、16B、64B、256B 等不同消息大小，带宽测试使用 4KB、64KB、256KB、1MB、4MB 等不同消息大小）。

OMB 采用常见的 GNU 编译系统 (configure & make)，运行时通过形如 osu_latency、osu_bw 的运行参数来执行相应的测试。

OMB 的下载链接及更详细信息参见<http://mvapich.cse.ohio-state.edu/benchmarks>。

3. 浮点计算性能测试程序 HPL

HPL 是当前最广为人知的高性能计算机性能基准测试软件，全球超级计算机性能 Top500 排行榜和中国高性能计算机性能 Top100 排行榜均采用 HPL 的测试结果作为排名依据。

LINPACK 是二十世纪七八十年代由美国 Jack Dongarra 等编写的一个用来求解线性代数问题（方程求解、最小二乘等）的函数库。为了帮助用户估算使用 LINPACK 求解方程所需的时间，LINPACK 开发者编写了 LINPACK Benchmark 的第一个版本，并提供了测试数据，针对一个 100×100 规模的矩阵问题，给出在众多当时流行的计算机系统上（共计有 23 种）的测试结果（运行时间）。这样，用户就可以推断出解决他们的矩阵问题所需的时间。1993 年 6 月，HPL 测评结果被引入全球超级计算机性能 Top500 排行榜，并取代了原来作为排名依据的峰值性能，成为新的排名依据并迅速获得了广泛的认同。其后直至今日，HPL 一直是高性能计算领域最重要的性能测评程序。

HPL 是通过求解一个稠密线性方程组 $Ax = b$ 所花费的时间来评价计算机的浮点计算性能。为了保证评测结果的公平性，HPL 不允许修改基本算法（采用 LU 分解的高斯消去法），即必须保证总浮点计算次数不变。对 $N \times N$ 的矩阵 A ，求解 $Ax = b$ 的总浮点计算次数为 $(2/3 \times N^3 - 2 \times N^2)$ 。因此，只要给出问题规模 N ，测得系统计算时间 T ，则 HPL 将测定该系统浮点性能值为：计算量 $(2/3 \times N^3 - 2 \times N^2)$ / 计算时间 T ，其单位为 flops。

HPL 允许用户选择任意 N 规模，并在不改变总浮点计算次数和计算精度的前提下对算法或程序进行修改。这在一定程度上促使用户为了取得更优的 HPL 值而八仙过海，各显神通。

常用的 HPL 优化策略或途径如下。

(1) 选择尽可能大的 N ，在系统内存耗尽之前， N 越大，HPL 性能越高（这是为什么呢？请读者思考）。

(2) HPL 的核心计算是矩阵乘（耗时通常在 90% 以上），矩阵乘采用分块算法实现，其中分块的大小对计算性能影响巨大，需综合系统 CPU 缓存大小等因素，通过小规模问题的实测，选择最佳的分块矩阵值。

(3) HPL 采用 MPI 进行并行计算，其中计算进程以二维网格方式分布，需要设定处理器阵列排列方式和网格尺寸，这同样需要通过小规模数据的测定获得最佳方案。

(4) LU 分解参数、MPI、BLAS 数学库、编译选项、操作系统等其他众多因素同样对最终测试结果有影响，读者可参阅相关文献获知更详细的信息。

HPL 的安装需要编译器、并行环境 MPI 和基本线性代数函数库（BLAS）支持。其中需要注意的是 BLAS 库的选择。如前面所述，当前常用的 BLAS 库有 GOTO、OpenBLAS、Atlas、MKL、ACML 等多个版本，不同系统上不同实现的性能可能会有较大差异，需参照相关文献或实际测试选择最佳版本。

最新版的 HPL 可从 <http://www.netlib.org/benchmark/hpl> 获得。

4. 综合性能测试程序 HPCC

HPCC (high performance computing challenge) 是由美国高等研究计划署 (defense

advanced research projects agency, DARPA) 的 HPCC (high productivity computing system) 项目所支持发布的一套评价高性能计算系统的基准测试程序集。

HPCC 基准是由若干知名的测试程序(包括单项测试和浮点性能测试)组成的,并刻意选择了有着鲜明空间局部性或时间局部性的典型测试程序,以期望对高性能计算机系统性能给出全面的评价。HPCC 所选择的七个测试程序分别如下。

- (1) HPL, 如前面所述,获得浮点计算性能值;时空局部性均良好。
- (2) DGEMM, 双精度实数矩阵乘,获得浮点计算性能值;时空局部性均良好。
- (3) Stream, 如前面所述,获得系统内存持续带宽值;空间局部性好,时间局部性差。
- (4) PTRANS, 并行矩阵转秩,体现系统通信性能;空间局部性好,时间局部性差。
- (5) RandomAccess, 测试内存随机更新速率(GUPS);时空局部性均不好。
- (6) FFT, 计算双精度一维复数离散傅里叶变换,获得浮点计算性能值;空间局部性差,时间局部性好。
- (7) b_eff, 测定系统通信带宽和延迟。

上述七个测试程序共获得 8 个测试值(b_eff 提供带宽和延迟两个结果),HPCC 常采用如下可视方式形象展现一台计算机的性能表现(不同类型的线条代表不同计算机),如图 5-1 所示。

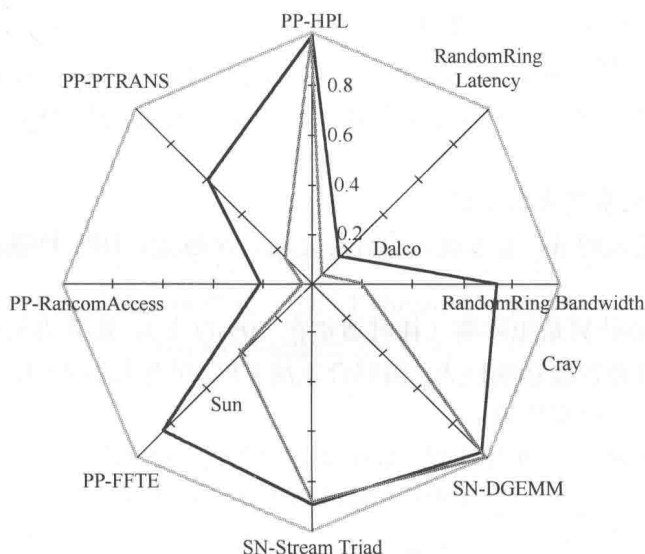


图 5-1 HPCC 结果示意图

HPCC 尽管提供了远超过单个性能测评程序(如 HPL)的丰富测试结果,但并未在高性能计算界获得广泛的支持和认可。其原因是多方面的,测试过程和测试结果过于复杂、无法给出易于比较的单一指标是两个重要因素。曾有一段时间,国际包括国内有一些研究者试图综合上述 8 个指标,依据某种准则计算得到一个类似 HPL 的 flops 那样的单一指标,这些工作尽管不乏亮点,但最终均以失败告终。不管如何,HPCC 仍旧是一个出色的能综合评定高性能计算机系统性能的基准测试软件。

HPCC 可以从 <http://icl.cs.utk.edu/hpcc> 获得。

5. 领域测试程序集 NPB

NPB 是一个科学计算领域的并行计算机性能评测基准程序,它含有 8 个不同的基准测试,都来自计算流体动力学应用软件,每一个基准测试模拟并行应用的一种不同行为,因此,NPB 可以测试出集群系统上计算流体动力学 (computational fluid dynamics, CFD) 并行应用程序的性能和可扩展性。

NPB 由美国航天局的艾姆斯研究中心 (NASA Ames Research Center) 开发,它制定了 NPB 测试程序的规范,可根据不同机器平台的具体特点在规范允许范围内由厂商自行按最优方法来实现程序。

对于并行版本的 NPB,需要根据系统的体系结构,在并行粒度、数据结构、通信机制、处理器映射、内存分配等方面进行有针对性的优化。NPB 2 以上的标准统一提供了用 MPI 实现的并行程序。

NPB 包含 5 个核心程序,它们是应用较频繁的一些算法。

整数排序 (integer sort, IS), 主要测试整数运算性能和聚合通信 (collective communication) 性能。IS 的主要操作是并行排序,并行排序对于物理学粒子研究的程序代码比较重要。IS 程序不包含浮点运算,但是它使用了较多的数据通信。整数排序对通信延迟比较敏感,对于这类应用,使用高带宽、低延迟的通信网络对性能的提高比较关键。

EP (embarrassingly parallel), 几乎没有数据通信,主要测试数学函数的浮点运算性能,通信网络对 EP 程序的测试结果影响甚微,对于 EP 这种通信很少的应用,对通信网络性能的依赖不高,这种类型的应用一般具有较好的可扩展性。

三维多重网格 (3-D multigrid, MG), 采用多重网格算法求解三维泊松方程,要求处理器数目必须为 2 的幂次,主要测试规则 (structured) 的非连续存储访问聚合通信和点到点通信, MG 既有整数运算也有浮点运算。

共轭斜量法 (conjugate gradient, CG), 主要测试不规则 (unstructured) 的聚合通信和点到点通信。

快速傅里叶变换 (fast Fourier transform, FFT), 用 FFT 求解三维偏微分方程,主要测试聚合通信。

NPB 还包含 3 个拟应用程序,它们来自计算流体动力学。

LU (lower-upper triangular, 对称超松弛法 (SSOR) 求解块稀疏方程组), 要求处理器数目必须为 2 的幂次,主要测试细粒度的非连续存储访问点到点阻塞通信,对通信延迟很敏感。

BT (block tridiagonal, 解 5×5 三对角块方程组), 要求处理器网格是方形的,主要测试通信和计算的平衡,以非连续存储访问点到点长消息通信为主。

SP (scalar pentadiagonal, 解五对角线方程组), 要求处理器网格为方形,主要测试通信和计算的平衡,以非连续存储访问点到点长消息通信为主,与 BT 的通信模式很相似,但两者在通信/计算的比率上有较大的差别,SP 通信强度比 BT 高。

NPB 的每个基准测试有 6 种规模 (class), 分别是 A、B、C、D、W (workstation) 和 S (sample), IS 没有 D 类规模。A、B、C 和 D 代表 4 种不同的问题规模,从 A 到 D 问题

规模递增。问题规模指的是测试所用的数据集的大小,较大的数据集需要较长的运算时间和较大的存储空间。

测试所用的处理器数目也需要指定,NPB 的 8 个程序对处理器的数目有不同的要求,BT 和 SP 要求处理器的数目是 n^2 (n 为正整数),LU、MG、CG、FT 和 IS 要求处理器的数目为 2^n (n 为正整数),EP 程序对处理器数目没有特殊要求。如果指定的处理器数目不符合要求,在编译的时候会有相应的错误提示。

为了更准确地表现出系统的性能,需要根据机器系统配置和研究的需要来选用合适的问题规模 and 处理器数目来进行测试。

NPB 可以从 <http://www.nas.nasa.gov/publications/npb.html> 获取。

6. 领域测试程序集 IAPCM Benchmarks

与美国的 NPB 类似,北京应用物理与计算数学研究所(IAPCM)作为中国最主要的高性能计算应用研发和应用机构之一,同样开发了体现本领域需求特点的高性能计算机性能基准测试程序集 IAPCM Benchmarks。

IAPCM Benchmarks 基准测试程序于 20 世纪 80 年代初开始开发,并于 1992 年第 17 期《计算机世界》首次发表了“对若干计算机系统性能的评价”的文章,目前共有 10 个程序,它们分别如下。

(1) 非平衡系统动力学程序(EPT)。EPT 程序用六点差分格式近似非热动平衡分群光子扩散方程组(E 方程),用 GEAR 积分近似束缚电子占据几率方程组(Pn 方程)和自由电子温度方程(T 方程)。这是一个复杂的非常定问题,特别是 Pn 方程为一组非线性强 Stiff 常微分方程组(刚性比达 10^{10})。

(2) 二维跟踪界面活动网络法程序(YGX)。YGX 是一种欧拉方法和拉格朗日方法相结合的二维计算程序。它是以轴对称的流体力学积分方程组出发进行差分离散的。该方法跟踪物质边界的运动,在物质区域内部采用一组近似椭圆型方程的方程组来构造计算网格,因而有一种连续地重分网络的能力。

(3) 粒子输运程序(DOSE)。DOSE 是用蒙特卡罗(Monte Carlo)方法求解粒子输运方程的程序。

(4) 线性松饼势轨道方法程序(LMTO)。LMTO 是用目前国际上应用极其广泛的一种线性能带论方法编写的程序。

(5) 线性代数方程组迭代解法 I 程序(WBR)。热传导方程差分离散后形成的线性代数方程组,它的系数矩阵具有五条非零对角元素。WBR 程序用块超松弛迭代法(BSOR)求解该线性代数方程组。

(6) 线性代数方程组迭代解法 II 程序(KGFM)。带一次导数项的热传导方程差分离散后形成线性代数方程组,它的系数矩阵具有五、七、九条非零对角线元素。KGFM 程序用不完全 LU 分解的共轭梯度法(ILUGG)求解该线性代数方程组。

(7) 二维离散坐标法程序(FSW)。FSW 求解的是二维中子输运方程。角速度空间用离散坐标法,对几何空间用矩形网络上的差分近似,用 GS 迭代法求解。

(8) 二维多介质隐式欧拉法程序(NNFC)。NNFC 是欧拉型数值方法。它是由轴对称

流体力学微分方程出发进行差分离散的。该方法采用固定的欧拉网格，不同物质用网格中的不同标记来表示，并以这些标记在网格之间的运动来实现输运量的计算。

(9) 分子动力学程序 (MD)。MD 程序用 Morse 势描述分子间的相互作用，采用跳点格式进行数值计算，程序设计用链表结构。

(10) 状态方程程序 (DUP)。这是状态方程数据库程序的一部分，其功能是对状态方程数据进行预加工，以备插值时用。该程序有大量数据的存取，要用多个通道进行数据传输，可以用来考察计算机传输通道的性能。

测试结果为 IAPCM 开发、选用计算机提供了必要的评估资料，也为许多中国的科研单位选购计算机提供了有关计算机性能的测试数据。

7. 领域测试程序集 Graph500 Benchmark

Graph500 是对数据密集型应用的高性能计算系统排行榜，其依据的测试程序集即 Graph500 基准测试包。和上述大部分测试程序关注系统浮点计算峰值不同，Graph500 主要利用图论去分析超级计算机在模拟生物、安全、社会以及类似复杂问题时的吞吐量，并进行排名。

Graph500 Benchmark 所计算的问题是在一个庞大的无向图中采用宽度优先算法进行搜索。测试包括两个计算核心：首先是生成待检索的图并以稀疏的 CSR (compressed sparse row) 或 CSC (compressed sparse column) 方式压缩存储；其次是采用并行 BFS (breadth-first search) 方法进行检索。目前有 6 种不同数据规模可选。

- (1) Toy 级， 2^{26} 个顶点，约需 17GB 内存。
- (2) Mini 级， 2^{29} 个顶点，约需 137GB 内存。
- (3) Small 级， 2^{32} 个顶点，约需 1.1TB 内存。
- (4) Medium 级， 2^{36} 个顶点，约需 17.6TB 内存。
- (5) Large 级， 2^{39} 个顶点，约需 140TB 内存。
- (6) Huge 级， 2^{42} 个顶点，约需 1.1PB 内存。

Graph500 依据 GTEPS (10^9 traversed edges per second) 对系统进行排名，当前排名第一的是日本的超级计算机“京”，其性能为 17977.1GTEPS。

8. 浮点计算性能测试程序 HPCG

高性能共轭梯度 (high performance conjugate gradient, HPCG) 是近年来由 HPL 的主要作者之一 Jack Dongarra 等提出的一个期望弥补 HPL 不足的新测试程序。它采用共轭梯度法求解大型稀疏矩阵方程组 $Ax = b$ 。

实际上，这类方程源自非定常非线性偏微分方程组，迭代求解过程中需要频繁地存取不规则数据，因此 HPCG 对计算机系统要求高带宽、低延时、高 CPU (核) 主频。而具备这些特点的计算机通常研制时间长、研制难度大且价格昂贵，即使在美国，也只有极少量“领导级计算机”属于此类系统。

整体而言，HPCG 所代表的问题涉及面较窄，基于此的性能测试程序要想如同 HPL 那样取得广泛的应用和认可，仍有较长的路要走。

5.2.2 高性能计算系统排行榜

1. Top500 排行榜

Top500 是全球已知最强大的计算机系统的排名榜。最早的榜单于 1986 年由德国的 Hans Meuer 发布,其排名依据是系统的理论峰值。1993 年引入 HPL 作为排名依据后,Top500 榜单开始真正得到了全球广泛的关注和认同。目前 Top500 一年发布两次,分别是六月份在德国召开的国际超级计算机会议(international supercomputing conference, ISC)和十一月份在美国召开的超级计算会议(supercomputing conference, SC)上。

关于 Top500 的详细信息可参阅 <http://www.top500.org>。

2. 中国 HPCTop100 排行榜

中国高性能计算机性能 Top100 排行榜,简称 HPC Top100,是指依据 LINPACK 测试性能进行排序的中国最快的 100 台计算机系统的榜单列表,是衡量中国高性能计算机系统及应用发展的重要参考依据。

2001 年,为促进中国高性能计算事业的发展,推动国产高性能计算机的应用与产业化进程,中国软件行业协会数学软件分会(SAMSS)倡议发起中国高性能计算机性能排行榜工作,并于 2002 年由孙家昶、袁国兴、张林波三人发布首次 Top50 榜单。经过十余年发展,Top100 排行榜得到了中国高性能计算机研制单位、系统厂商、应用单位和高性能计算相关科研机构的充分肯定和认可,在国际上有着广泛而深远的影响力。Top100 榜单体现的已不仅仅是中国高性能计算事业的发展变迁,还折射出中国科技和经济的长足进步和日趋强大。

无论是国际的 Top500 排行榜,还是中国的 Top100 排行榜,均选择了 HPL 作为性能排序依据,这主要是因为 HPL 具有如下的一些良好特性。

(1) 应用广泛。HPL 要求计算机有很高的浮点速度和大内存,这类系统有着广泛的应用领域,重大军事装备、能源、航天、航空、高端装备制造、生物医药、石油、海洋环境工程、天气预报和气候预测等主要 HPC 领域都有此类计算问题。

(2) 适应性强。HPL 很容易适应各种不同结构的计算机系统的计算测试要求,如处理器差异、内存结构差异、加速部件差异、网络差异、操作系统差异等。中国 Top100 包含了高性能计算中不同结构的计算机系统。集群、MPP、世界首台 GPU 系统、排名第一的 Intel Phi 系统、Windows 集群等。

(3) 可扩展性好。中国 HPC Top100 自 2002 年至今,计算机系统的性能变化范围覆盖了 6 个数量级(从 13.17Gflops 到 33.86Pflops)。

(4) 能检测计算机系统的可靠性。HPL 是检测计算机全系统能否持续稳定运行并给出正确结果的最有效程序,这对当前有数万个甚至更多计算部件的超大型系统尤为重要。可靠性是大规模系统面临的最大考验,而运行 HPL 测试首要解决的就是系统的可靠性。

中国 HPC Top100 的更多信息可参见 <http://www.hpctop100.cn>。

3. Green500 排行榜

Green500 排行榜将 Top500 排行榜上的系统依据能效进行排名,其排名依据为 flops/W,

即每瓦计算性能。

Green500 自 2007 年开始发布，迄今已有 15 届。目前排名第一的是日本东京工业大学的 TSUBAME-KFC-LX 1U-4GPU/104Re-1G 集群系统（4389.82Mflops/W）。

测试自有计算机系统的功率可通过专门的功率计进行，需要注意的是功率值需取图 5-2 所示性能（功耗）稳定阶段的平均系统功耗。

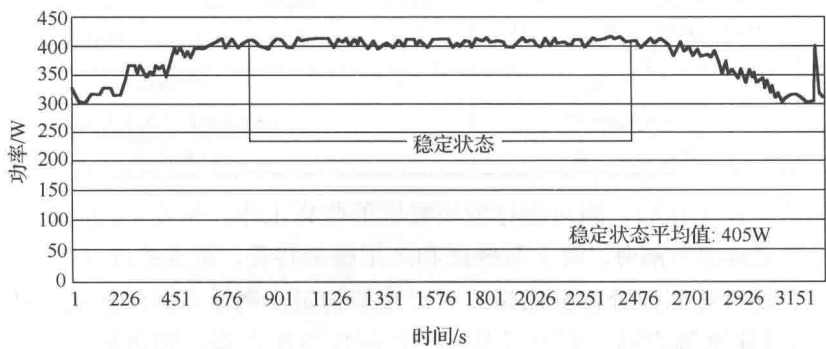


图 5-2 系统功耗记录

5.3 超算应用特征分析与监测工具

人们建造超级计算机的目的是提高科学研究和工农业生产的效率，因此越来越快的计算速度是人们追求的目标。但超级计算机是一个十分复杂的系统，它不仅包含基础的硬件平台、系统中间件平台，还包括在超级计算机上运行的各行各业的应用软件。因此，如何使各个行业的不同应用软件在超级计算机上运行时达到效率的最大化，是一个极其复杂的难题。它涉及对硬件、系统和软件三方面的优化，是一个有机的整体，需要全方位考虑。此外，应用行业的差异也会导致应用软件特征的巨大差别，如何提取这些应用特征，并针对这些应用软件特征进行优化就是问题的关键。这其中涉及对应用特征数据的提取，对所提取数据的分析，以及对应用特征的分析过程，本书将在下面重点讨论。

5.3.1 应用特征数据提取

一般地，应用特征的提取和分析工作是建立在某个固定的硬件平台基础之上，并针对某个特定计算体系的。当运行应用软件的硬件平台发生变化，或者应用所计算的体系发生变化时，往往应用的特征也会随之而变。因此，在固定硬件平台，并选定某应用软件和合适的负载算例后，就可以在应用运行过程中进行应用特征收集工作。表 5-2 给出了一般从事应用特征提取时的硬件平台示例。对于平台的选择，通常是选用主流配置，即选择主流 CPU。在选择 CPU 时应当考虑到 CPU 的主频、QPI（quick path interconnect）、高速缓冲等属性；选择内存时，应该从内存容量、内存通道数、条数等方面考虑；对于文件存储系统则需要考虑磁盘读/写速率；网络方面要了解网络带宽、延迟等属性。在平台确定后，首先应进行算例设置，之后编译应用软件。完成后，需要测试应用软件的运行情况，以确认算例是否可以正常运行，以保证应用特征收集的有效性。

表 5-2 测试平台（常用计算平台）

组成	选配方案	考虑角度
算例	运行时间	算例规模、算例设置
软件	版本	软件版本
机型	计算节点：NX5440（刀片）	根据项目需求选择机型
CPU	Intel E5-2600v3, E5-2600v2	CPU 频率、高速缓存、QPI 频率、Turbo 配置、NUMA 配置
内存	DDR3/DDR4 1600/1866/2133 MHz	内存容量、内存频率、DIMM/Socket
存储/文件系统	I/O 节点, NFS	AS500G3
网络	InfiniBand	InfiniBand、10GB 以太网
OS	RHEL6.4、7.0	OS 版本

在完成上述准备工作后，即可进行应用特征的收集工作。所收集的特征主要分为系统级、应用级和微架构级三部分。对于系统级和应用级的特征，重点关注计算平台的处理器、存储器、网络、硬盘等四部分特征指标；对于微架构级，则重点关注应用的代码指令执行效率、CPI、向量化率等指标。在上述基础特征指标收集之后，即可进行系统平台调优和应用调优，这两部分与特征收集是循环进行的，直到优化后的应用特征没有明显的瓶颈存在。最后，将根据优化后的特征数据进行应用的特征剖析，为设计行业应用的最优化系统构建方案提供依据。图 5-3 是高性能计算应用特征剖析系统示意图，其中性能抓取步骤所重点关注的指标和监控工具在表 5-3 中列出。应用特征剖析系统一般分三个步骤：首先是确定计算平台和软件的算例，然后进行特征收集，最后优化硬件平台和应用软件。

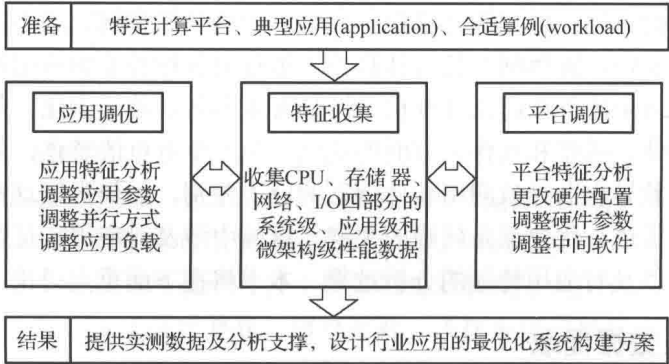


图 5-3 高性能计算应用特征剖析系统

表 5-3 高性能计算性能指标及监控工具

主要指标		说 明	监控工具
CPU	应用计算时间	应用计算时间	ITAC, TAU
	平均负载	CPU 负载，判断是否需要更多资源	TOP, Ganglia, Teye
	CPU 使用率/%	应用软件及操作系统对 CPU 资源的占用比例	TOP, PAPI, TAU, Teye
	L3 高速缓存缺失率/%	相对高的 L3 高速缓存缺失率表示程序数据预取效果较差	PAPI, VTune
	CPI	平均每条指令消耗的时钟，相对高的 CPI 表示性能优化空间较大	VTune, Teye
	浮点计算速度	应用运行时的 CPU 实时浮点计算速度	Teye
	向量化率	应用程序运行时向量化指令所完成的浮点计算的占比	Teye

续表

主要指标		说 明	监控工具
内存	内存容量	应用负载中的内存使用量	free, Nmon, Ganglia, Teye
	内存带宽	应用负载内存带宽占用量	Teye
	SWAP 使用	SWAP 空间使用量	Nmon, Ganglia
内存	NUMA 比例	程序的远程内存访问所占比例	VTune
网络	Communication Time%	即 MPI Time%, MPI 通信时间所占比例	ITAC, TAU
	MPI Wait Time%	MPI 通信中的进程等待时间比例	ITAC, TAU
	Network In	网络进	Nmon, Ganglia, Ethtools, Teye
	Network Out	网络出	Nmon, Ganglia, Ethtools, Teye
	延迟	应用负载下的节点间传输延迟	实测: OSU, IMB
	带宽	应用负载下的节点间网络带宽	实测: OSU, Iperf
I/O	Disk I/O%	程序处理 I/O 时间所占总时间比例	Nmon, Ganglia, IOstat
	文件系统 I/O 带宽	文件系统 I/O 带宽大小	实测: IOzone, IOMeter
	粒度	运行中每次存储数据的大小	Nmon, Ganglia, Teye
	频率	运行中存储数据次数/时间	Nmon, Ganglia

表 5-3 中列出的工具软件可以在应用软件运行过程中监测计算节点的各项指标, 包括 CPU 利用率、平均负载、系统占用率、浮点运算速度、CPI、向量化率、内存容量、内存带宽、文件输入/输出大小、输出频率、输出规律、网络流量、网络带宽延迟、消息块大小、主要消息传递次数、MPI 通信占整个应用的百分比等。

为了能够精确地反映应用软件的特征, 应用特征提取的时间间隔往往非常短, 通常以秒计, 从而导致反映应用特征的数据量巨大。因此, 高效的特征数据收集和数据的分析往往需要借助数据库, 如 MySQL 等来完成。

5.3.2 应用特征分析——实验归纳

在提取了应用的特征数据后将进行应用的特征分析。一般地, 对于现有的冯·诺依曼架构的计算机, 将主要从处理器、内存储器、I/O 和互连四个方面来分析。

(1) 内存储器特征分析。内存可以看成内存控制器与 CPU 之间的桥梁与仓库。显然, 内存的容量决定“仓库”的大小, 而内存的带宽决定“桥梁”的宽窄, 也就是常说的内存容量与内存速度, 两者缺一不可。一些高性能计算中的应用由于算法和算例的特殊性, 需要较大的内存容量(如 256/512GB)才能正常运行; 另外一些应用需要较高的内存带宽(内存速度)才能确保应用程序快速高效地运行。因此, 内存存在高性能计算中发挥着非常重要的作用。

按照高性能计算的应用对内存的不同需求, 应用又具体分为两类: 内存带宽需求型应用和内存容量需求型应用。内存带宽需求型应用是指在程序运行中 CPU 和内存控制器之间数据交换频繁, 需要较高的内存速度, 内存速度就成了应用中的性能瓶颈, 如图 5-4 所示。从图中可以看出, 该应用软件在运行时基本已经用尽了系统平台所能提供的最大内存带宽资源。此时, 内存带宽就是该应用软件的最大瓶颈所在。

因此, 对于某一特定应用软件在内存使用方面的情况, 应该从内存容量和内存带宽两个方面进行分析。针对内存容量的使用情况, 可以通过应用软件所使用的算法来统计, 或

通过使用 Nmon、Teye 等工具软件来监控；针对内存带宽的使用情况，其敏感性可以通过在节点内采用不同的进程运行组合来判断，或通过一些工具软件来实时监控，如 Teye 等。

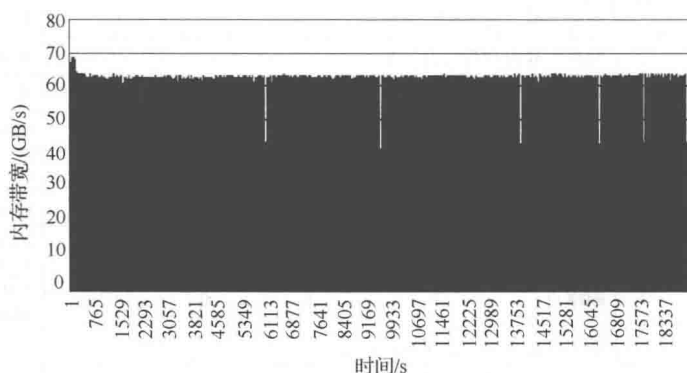


图 5-4 某应用在运行时的实时内存带宽表现 (DDR3 1333MHz)

(2) 输入/输出 (I/O) 特征分析。存储是高性能计算不可或缺的重要组成部分。它往往是超级计算机中技术要求最高、难度最大的系统，其建造的资金投入也是巨大的。存储性能的好坏对应用的性能影响将是本质的，特别是那些对数据敏感的行业。

在某些行业中，如生物信息、气象\气候模拟、流体计算等领域，一些应用的原始输入数据就达数 TB，而且经过计算处理后的最终结果文件可能还会更大。因此，直观地看，存储容量就是高性能计算必须考虑的一项重要因素。此外，一些应用程序在运行中会产生大量的中间数据、临时数据等文件，这些数据的格式种类繁多，文件块的大小也不一，因此就需要具有高每秒的输入/输出操作次数 (input/output operations per second, IOPS) 和高读/写速率的存储来提高应用运行效率。下面在图 5-5 中给出两款不同应用软件在运行时的 I/O 特征。图 5-5(a) 所示的应用软件在运行过程中的实时 I/O 非常稀少；而图 5-5(b) 所示的应用软件在运行进程中却要求大量和频繁的 I/O 读写。

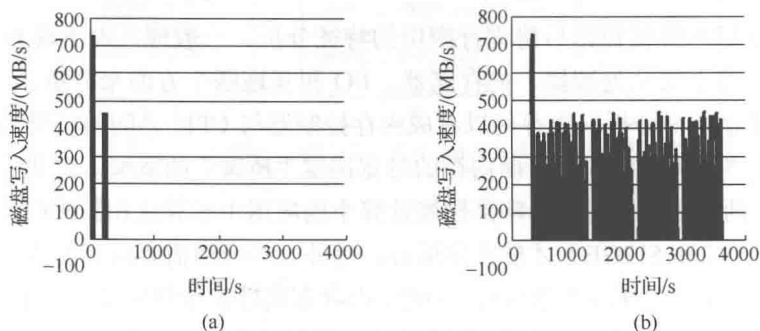


图 5-5 某应用软件在运行时的 I/O 特征

(3) 处理器特征分析。在应用软件运行过程中，处理器的大部分时间都用来处理数值计算和逻辑计算等，因此一般来讲，超级计算机的 CPU 利用率都很高，如图 5-6 所示。为了解一款应用在运行时的 CPU 利用情况，可以通过 TOP 命令来查看 CPU 的利用率、系统占用率、进程等待时间等情况；也可以通过 PAPI、TAU、VTune、Nmon、Teye 等工具软件来监控更详细的处理器特征，从而进行更深入的分析，图 5-7 给出了某应用软件在运行

时的实时浮点运算速度。CPU 使用情况对应用的性能分析是极为重要的，因此正确合理的分析方法和手段就成为分析的重点。

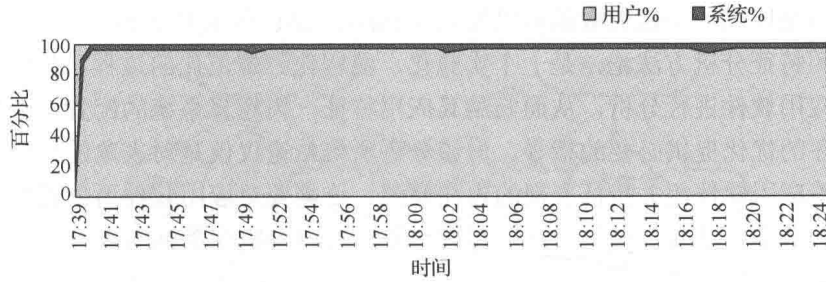


图 5-6 某应用软件在运行时的 CPU 利用率与运行时间的关系

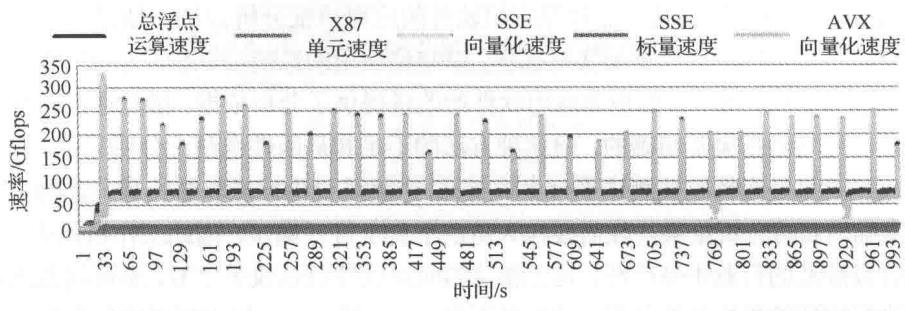


图 5-7 某应用软件在运行时的实时浮点运算速度与运行时间的关系

针对不同的应用，在分析 CPU 需求的时候可以通过监控 CPU 利用率、浮点运算速度、向量指令向量化率等指标，以了解应用的优化程度，进而指导调节应用的性能参数或硬件平台参数，以获取最优性能，从而达到对应用优化的目的。

(4) 互连特征分析。现代的超级计算机动辄拥有数十万乃至上百万的 CPU 核心，因此应用软件要充分利用这些计算资源，各进程间不可避免要进行通信和数据交换。进程间的互连和数据交换方式很多，有基于进程的，也有基于线程的，还有基于共享内存的，但一般在超级计算机上使用最广，对应用性能影响最大的就是消息传递的通信方式，即 MPI。因此，通过在应用运行过程中监控 MPI 的通信时间占比和网络流量，即可对应用在网络方面的特征进行分析，如图 5-8 所示。这些数据可以通过 Ganglia、ITAC 等工具来获得。

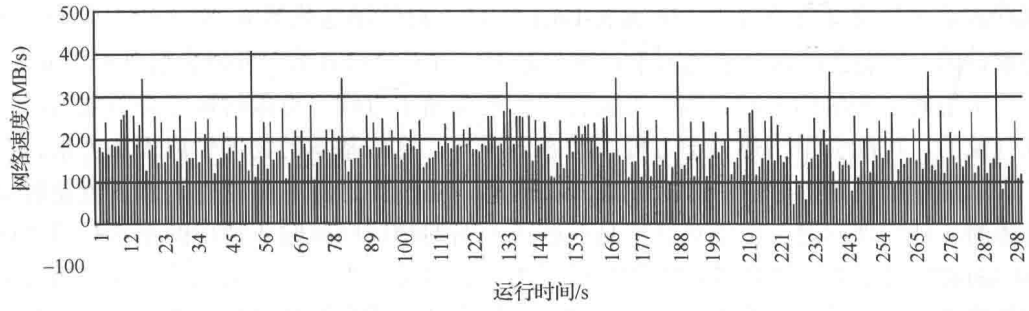


图 5-8 某应用软件在运行时的网络发送速度与运行时间的关系

此外，为了深入地优化基于 MPI 的应用软件，特别是从代码级优化。还可以监控和提

取应用软件在运行时的 MPI 消息事件和 MPI 函数的使用情况，如 MPI_Allgather、MPI_Alltoall、MPI_Scatter 等函数的调用情况。在大规模并行计算情况下，这些函数对应用的性能影响是巨大的。这类数据可以通过 VTune、TAU 等工具获得。

上述应用特征分析方法论，是一个实验化、流程化、规范化的过程。使用该方法可以对任何一款应用软件进行分析，从而归纳其应用特征，为超算系统的配置、集群方案的设计和应用程序的优化提供必要的指导。但该分析所得结论仅仅是对表象的总结，它往往无法解释和回答应用软件相关特征表现的内在原因。这需要对应用软件所依赖的基本理论背景和所使用的基本算法有全面的把握，有关这方面的内容将在后面着重介绍。

5.3.3 应用特征分析——理论预测

前面提出了一整套针对高性能计算应用软件的应用特征分析方法，通过在特定的平台上对应用运行时资源占用量等特征的实时抓取，以定位应用的性能，一方面为专用平台的设计提供事实上的参考依据；另一方面也为应用软件的改进提供了分析依据。虽然通过指标抓取的性能分析方法所得到的结果是精确的，但这种方法的不足和局限性也是十分明显的。首先，抓取的指标所反映的只是在特定被测试平台上的特征，也就是说指标抓取是具有一定的平台相关性的，为了准确地获得分析结果往往需要在大量的平台上进行测试，因此工作量巨大。其次，由于高性能计算涉及的行业十分广泛，其所涵盖的应用软件更是成千上万，不可能做到对所有的应用软件进行特征的指标抓取分析，这也是不现实的。最后，通过指标抓取所获得的分析结果均是应用软件的表面现象，无法仅通过这些现象了解现象产生的原因，而且也很难对同一行业不同应用的迥异表现给出合理的说明和解释。那么是否还有更好的方法解决上面的问题呢？

在这里作者结合自身多年从事高性能计算研究和实践的经验，提出了一整套理论原理性能预测方法。不同于指标抓取的性能分析方法，它不依赖于特定的硬件平台，而是从应用软件所依据的基本物理原理、模型、算法入手来分析应用软件的性能特征。

应用软件无论大小、功能多少、用户界面是否友好，均是由代码实现的，而这些代码又是以特定的物理原理，依据适当的物理模型以合适的数学算法构建、组织起来的，图 5-9 形象地反映了这种依赖关系。

应用软件的理论模型包括基本的物理模型和数学算法两方面。物理模型是指这款应用软件所使用 and 依赖的物理方法，例如，对于研究材料和化学的应用软件，可能会用到第一性原理方法（如密度泛函理论、Hartree-Fock 方法、自洽场迭代等）；对于研究生物分子或药物设计的，可能会用到分子动力学理论；对于从事工业设计的，可能会用到有限元方法；对于从事电磁、光学相关的应用，可能会用到麦克斯韦方程、有限时域差分方法等；对于研究气象的，可能会用到热力学方程、状态方程等。这些基本的物理学原理和方法往往决定了这款应用所使用的算法和并行的方式。因此，高性能计算应用软件的本质就是解决上面所述的各种物理问题。数学算法就是在解决上面的物理问题时所使用的基本数学手段，但不同的数学算法对应用软件性能和并行度的影响也是巨大的。如果能够把这些物理原理和数学算法在运行时对计算资源的占用情况分析清楚，那么不经过指标抓取就能清楚地了解任何一款使用该应用软件的软件的特征。

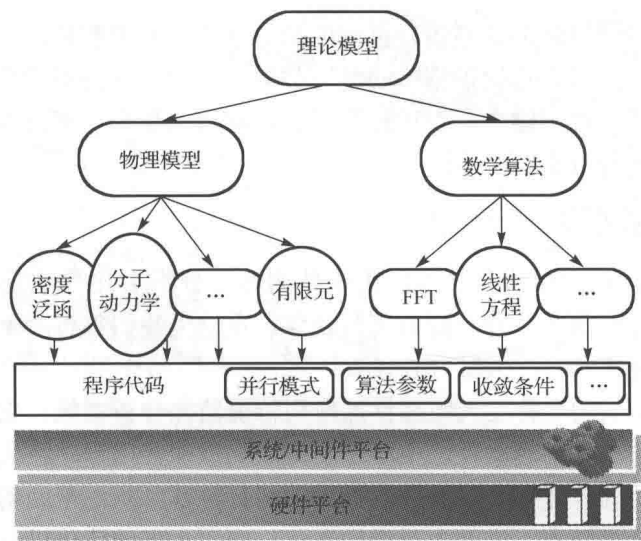


图 5-9 理论原理性能预测方法

为了详细说明，这里以材料和量子化学方面的应用为例。正如前面提到的，材料和量子化学多以基于量子力学的第一性原理为基本原理，如密度泛函理论（DFT），并通过 Kohn-Sham 方程的自洽求解来计算具体的物理量。而 Kohn-Sham 方程的求解方法有很多种，这些不同的方法也决定了不同的并行模式，而正是这些方法和并行模式的不同决定了应用软件的特征有巨大的差异。表 5-4 给出了几款常用的高性能计算应用软件所使用的基本方法。从表 5-4 中可以看出 VASP（Viennaab-initio simulation package）是以平面波为基矢，通过在带间和基矢间的并行来计算的。由于平面波型基矢离真实的轨道函数描述比较远，所以计算时往往需要大量的平面波数（几十万），加之在带间和基矢间并行时每次迭代都需要对所有的基矢进行正交归一化操作，这必然会导致异常频繁的内存读/写和进程间数据交换，即高内存带宽和高网络吞吐。相比较而言，虽然 ABINIT 也使用了平面波基矢，但它在实现并行化时却是使用的 K 点间并行模式，而每个 K 点在空间中是完备的，它不需要像带间和基矢间并行那样在进程间传递大量的归一化参数，因此，可以预见 ABINIT 的特点是大内存带宽、小网络吞吐，这点与 VASP 很不相同。

表 5-4 部分材料、量化应用软件所依据的基本物理原理

应用软件	基矢		并行化				对角化				
	平面波	原子轨道	带间	基矢间	K 点间	自旋极化	CG	PC-CG	DAV	RMM	CP
VASP	√		√	√					√	√	
ABINIT	√				√	√	√	√			
PWscf	√			√	√						√
Siesta		√									
CASTEP		√			√		√				
CPMD	√										√
Gaussian		√									√

从上面的分析过程可以看出，理论原理的预测方法可以快速定位应用的运行特征，对

应用的优化方向和专用平台的搭建均有重要的指导意义。但很明显，仅依靠理论原理的预测方法是远远不够的，它无法提供精确的特征数据。因此，理论原理分析方法应该与前面介绍的指标抓取分析方法相结合，以理论为指引、应用为导向、平台为依据，充分全面地对应用软件的特征进行分析。

5.3.4 应用特征监控工具——天眼

通过前面的讨论我们知道，从事应用软件的特征分析工作离不开特征数据的提取工具。在前面我们已经提到了多种开源的工具软件。在这里我们将再着重介绍另外一种功能强大的应用特征监控工具——天眼（Teye）。

天眼（Teye）工具的全称是天眼高性能应用特征监控分析系统。它是由浪潮专业的高性能计算团队开发的一款卓越软件，主要用于提取高性能应用程序在大规模集群上运行时对系统资源占用的情况，并实时反映应用程序的运行特征，从而帮助用户最大限度的在现有平台挖掘应用的计算潜力，进而为系统的优化、应用程序的优化以及应用算法的调整改进提供科学的指引方向。

天眼是一款可视化的工具软件，它由集群的性能监控和指标提取端以及客户机的性能分析端两个工具软件组成。其中，天眼的性能监控和指标提取端工具主要用于实时监控和提取指定的性能指标，它基于 BS 架构，无需用户安装任何客户端，仅通过网页浏览器即可正常使用天眼软件来监控和提取关键性能指标。图 5-10 和图 5-11 分别给出了天眼软件性能监控和指标提取端的登录界面和主应用界面。此外，天眼还具有体积小、易操作、数值监控精确、实时、资源占用率低等众多优点，即使在系统重负载情况下，天眼对系统资源的需求量也远远不足千分之一，极大程度上保证了所反映的高性能应用程序运行特征的真实性。

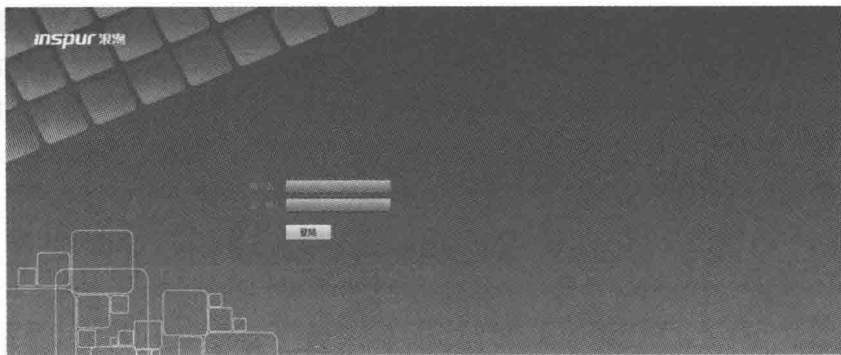


图 5-10 天眼（Teye）软件性能监控和指标提取端的登录界面

天眼软件功能强大，它不仅可以监控处理器、内存储器、互联、I/O（Input/Output）的特征数据，从而反映应用软件随时间变化的运行特征，还可以抓取 40 余项微架构指标，极大的满足高性能应用调优的需要。这些指标包括：节点内 CPU 的单、双精浮点运算速度 GFlops、x87 浮点单元速度、SSE/AVX 单元运算速度、128bit SSE 指令向量化率、256bit AVX 指令向量化率、内存读/写带宽、节点间 NFS 文件系统的吞吐、总 NFS 吞吐、PCI-E 设备与内存间的数据交换带宽等。

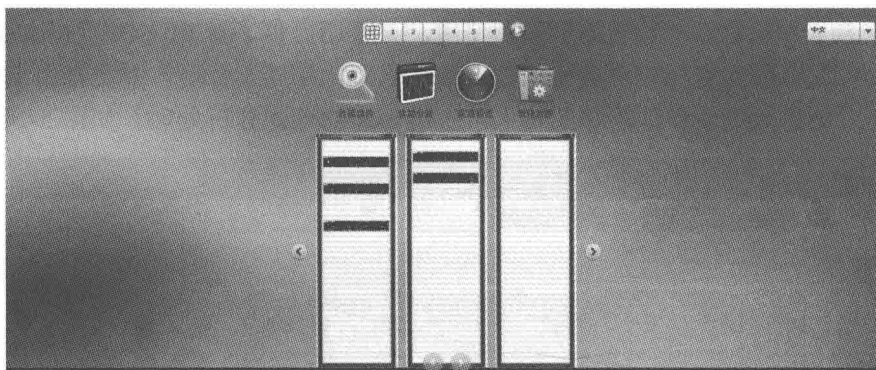


图 5-11 天眼 (Teye) 软件性能监控和指标提取端的主应用界面

图 5-12 给出了天眼软件在监控和提取性能数据时的应用界面。它反映了当前作业所关注节点 (由用户来指定) 的所有关键性能指标, 以缩略简图的形式动态地呈现在监控页面, 并且可以通过简单地点击节点图标以获得该节点的实时性能监控曲线示图, 如图 5-13 所示。

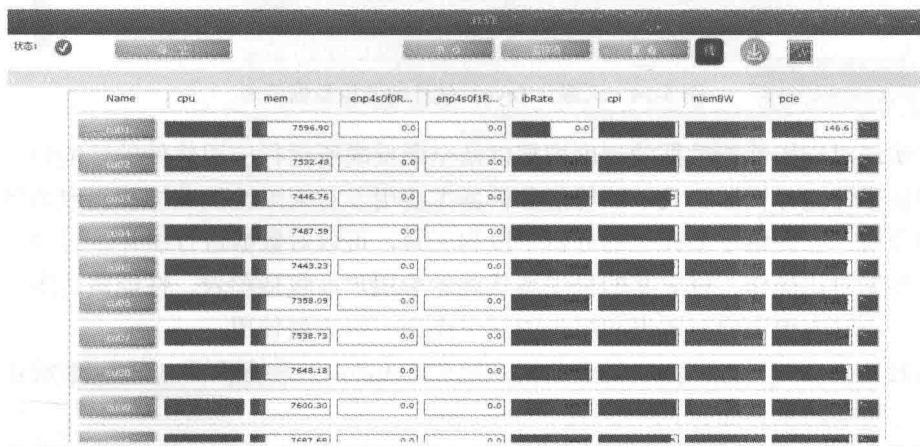


图 5-12 天眼 (Teye) 软件在监控时的应用界面

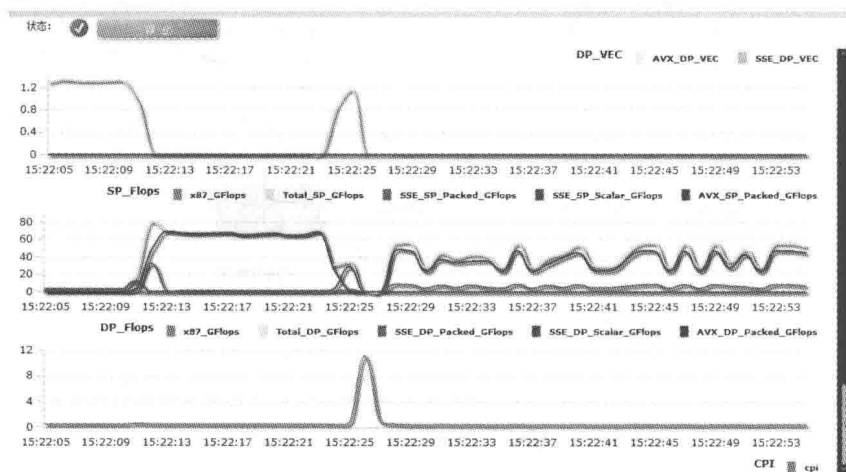


图 5-13 天眼 (Teye) 软件的实时性能曲线示图

此外，为了方便对监控数据的管理和使用，天眼还提供了对 MySQL（Mariadb）数据库的支持，可以方便实现用户对被监测数据的数据库化存储及管理，并支持用户对监控数据的导出，以用作天眼软件分析端的输入，做进一步的性能分析处理。如图 5-14 所示。

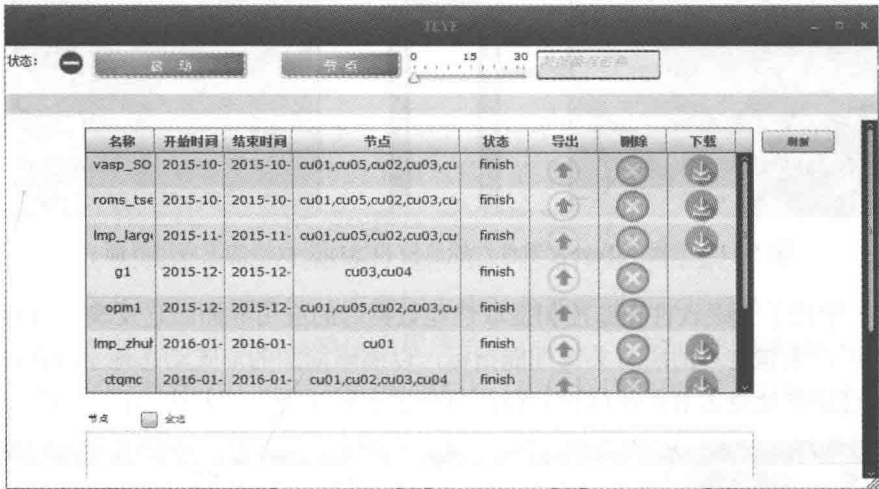


图 5-14 天眼（Teye）软件的监控数据管理

为了实现对应用性能数据的深度挖掘以及全面反映所运行应用软件的性能特征，天眼软件还提供了性能分析器工具。该性能分析器主要用于对天眼性能监控和提取端导出的性能数据的分析。它提供了多样化的分析手段和工具，可以对数据进行统计、过滤、热点定位，以及数据对比操作，以丰富的呈现形式展示给用户，包括图表、性能雷达图，以及报告生成等，同时方便用户对图表和报告的导出操作，以方便使用。

天眼软件分析端可方便的安装于任何主流的 Windows 系统环境，图 5-15 给出了天眼软件性能分析端的主界面。

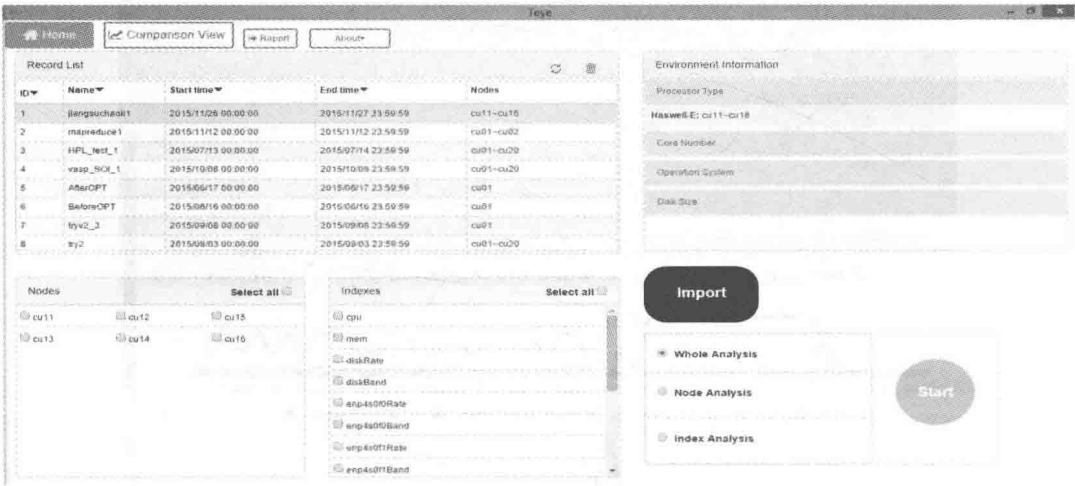


图 5-15 天眼（Teye）软件分析端主界面

图 5-16～图 5-19 分别给出了天眼分析器的整体分析功能、节点分析功能、指标分析功

能以及对比分析功能。整体分析用于对所监控作业的所有节点和所有性能指标的全面展示，用于对应用性能的整体把控；节点分析功能可深入到各个计算节点内，对节点内的运行负载及资源相关性进行分析，并可生成相应的性能雷达图；指标分析功能用于对相同指标在不同节点间的对比，以反映应用的负载的均衡性或进程的异步性等；对比分析功能可按用户需求，自定义所要对比的指标参数，极大地方便了性能的分析 and 优化。

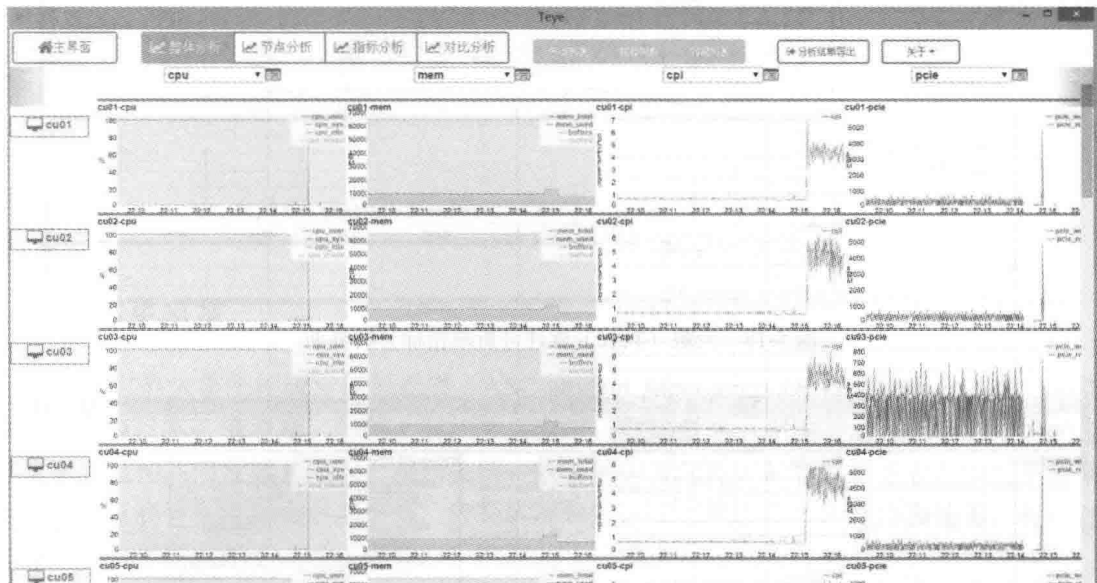


图 5-16 天眼 (Teye) 软件分析端整体分析界面

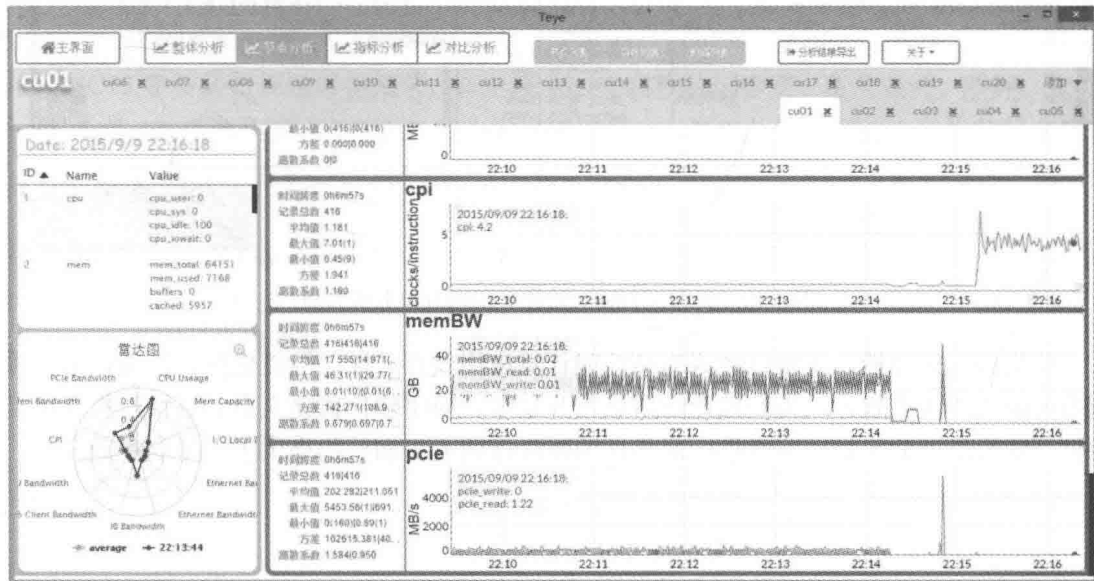


图 5-17 天眼 (Teye) 软件分析端节点分析界面

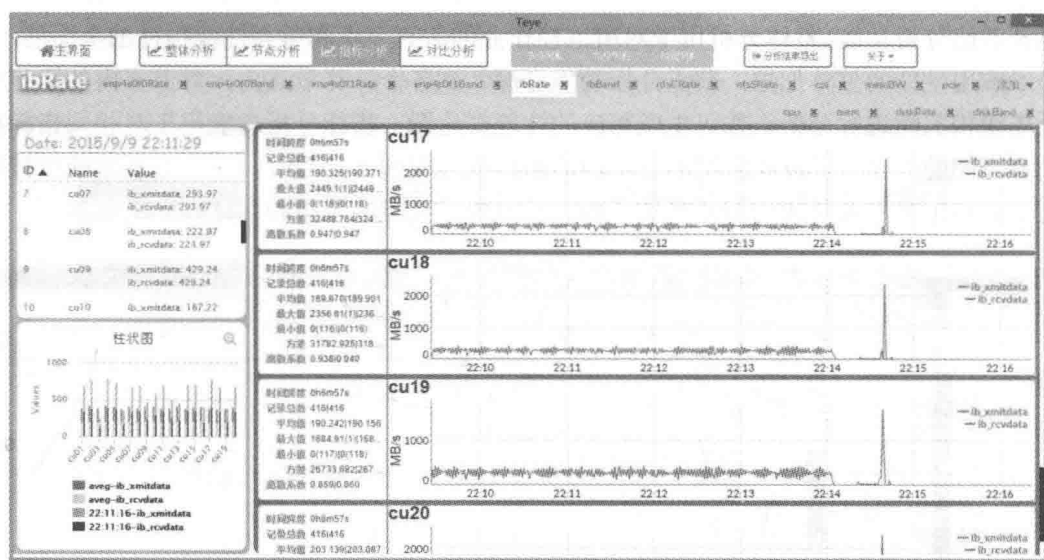


图 5-18 天眼 (Teye) 软件分析端指标分析界面

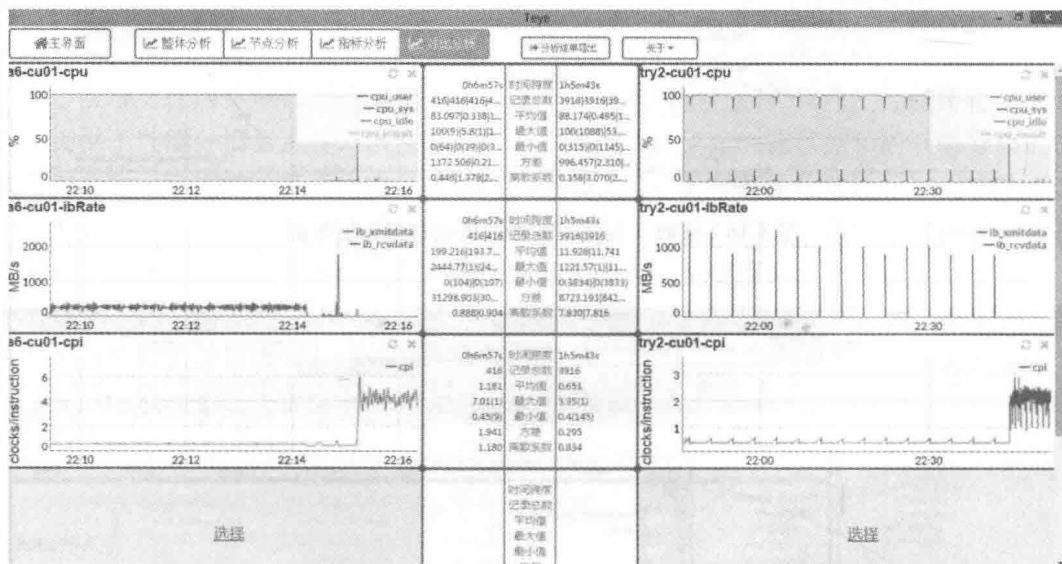


图 5-19 天眼 (Teye) 软件分析端对比分析界面

凭借强大的功能,天眼已经成为高性能计算应用软件开发、性能优化、性能评估等领域不可或缺的重要软件,为高性能应用软件开发人员、应用软件优化人员、高性能集群项目验收系统性能评估人员、高性能应用软件的性能评估人员和机构提供了大量的帮助。

第二篇 竞赛篇

第 6 课 世界大学生超级计算机竞赛介绍

6.1 ASC 世界大学生超级计算机竞赛

6.1.1 历史沿革

ASC 世界大学生超级计算机竞赛（ASC student supercomputer challenge）是由中国倡议成立，与日本、俄罗斯、韩国、新加坡、泰国、中国台湾、中国香港等国家和地区的超算专家和机构共同发起并组织，得到美国、欧洲等国家和地区的超算学者和组织的积极响应支持。竞赛旨在通过大赛的平台，推动各国和地区间超算青年人才交流和培养，提升超算应用水平和研发能力，发挥超算的科技驱动力，促进科技与产业创新。

ASC 肇始发轫于 2012 年，迄今影响力不断攀升。2012 年的大赛共有 27 所高校报名，范围仅局限于中国大陆；2013 年的范围扩展到全亚洲，共有 43 所高校报名参赛；而 2014 年举行的 ASC14 则得到世界范围内众多高校的积极参与，报名参赛的高校队伍达到创纪录的 82 支，参赛国家和地区覆盖亚洲、欧洲、北美洲、南美洲和非洲等五大洲。2015 年举行的 ASC15 参赛队伍规模再创纪录，有全球 152 支高校队伍参赛，覆盖亚洲、欧洲、北美洲、南美洲、大洋洲和非洲 6 大洲，成为世界规模最大、覆盖范围最广的大学生超算竞赛。

6.1.2 参赛团队

2014 年 ASC 超算竞赛在中山大学举办，总决赛规定各参赛队在 3kW 功耗约束下自行设计构建超级计算机并完成 LINPACK、Quantum Espresso、LICOM、SU²、3D_EW、mpiBLAST 应用优化赛题比试，表 6-1 是进入决赛的队伍。

表 6-1 ASC14 十六强名单

学校名称	历史成绩
新加坡南洋理工大学	首次参加国际级超算大赛
美国普渡大学	最早参加美国 SC 大赛，在历届 SC 竞赛中表现不俗
巴西圣保罗大学	首次参加国际级超算大赛
俄罗斯乌拉尔联邦大学	首次参加国际级超算大赛
匈牙利米什科尔茨大学	首次参加国际级超算大赛
韩国蔚山科学技术大学	韩国超算竞赛冠军，曾入围 ASC13 总决赛
香港理工大学	曾参加 ASC13 初赛，表现不俗

续表

学校名称	历史成绩
台湾清华大学	SC10、SC11 竞赛总冠军，ASC13 竞赛亚军
太原理工大学	首次参加国际级超算大赛
清华大学	ASC12 决赛东道主并获总冠军，ISC12 竞赛总冠军，ASC13 竞赛总冠军、最高计算性能奖
浙江大学	首次参加国际级超算大赛
上海交通大学	ASC13 决赛东道主并获 LINPACK 成绩亚军
北京航空航天大学	首次参加国际级超算大赛
华中科技大学	ASC13 竞赛 MIC 应用优化奖，ISC13 最高计算性能奖
国防科学技术大学	ASC12 决赛亚军，ISC12、SC12、SC13 竞赛最高计算性能奖
中山大学	ASC 13 竞赛 GROMACS 应用最佳成绩

最终，上海交通大学获得总决赛冠军，新加坡南洋理工大学获得亚军。中山大学获得最高计算性能奖，首次设立的 e Prize 计算挑战奖由上海交通大学夺得，上海交通大学、清华大学、太原理工大学和北京航空航天大学共同获得应用创新奖，如图 6-1～图 6-6 所示。



图 6-1 总冠军上海交通大学三冠 ASC14



图 6-2 亚军：新加坡南洋理工大学



图 6-3 最高计算性能奖：中山大学



图 6-4 应用创新奖：清华大学



图 6-5 应用创新奖：太原理工大学



图 6-6 应用创新奖：北京航空航天大学

2013 年的 ASC 超算竞赛在上海交通大学举办，共有 43 所高校报名参赛。经过激烈的预赛角逐，清华大学、国防科学技术大学、上海交通大学、华中科技大学、中山大学、台湾清华大学、香港中文大学、圣彼得堡大学（俄罗斯）、蔚山科学技术大学（韩国）、阿卜杜勒·阿齐兹国王大学（沙特阿拉伯）等 10 所高校脱颖而出进入了总决赛。

总决赛中，10 支参赛队伍在现场独立搭建总功率不超过 3kW 的平台，然后进行 LINPACK、GROMACS、BSDE、WRF（weather research and forecasting）、OpenCFD 5 个算例的比赛，决赛采用百分制，每个算例 18 分，最终呈现环节 10 分。

最终，清华大学以优异的表现力压群雄，夺得冠军并获得最高计算性能奖；台湾清华大学以微弱劣势惜败，获得亚军；华中科技大学获得 MIC 应用优化奖，如图 6-7～图 6-10 所示。



图 6-7 两岸清华大学齐聚 ASC13



图 6-8 总冠军和最高计算性能奖：清华大学



图 6-9 亚军：台湾清华大学



图 6-10 MIC 应用优化奖：华中科技大学

2012 年的 ASC 超算竞赛在清华大学举办, 当时报名参赛的 27 所高校中, 国防科学技术大学、中国科学技术大学、清华大学、武汉大学、山东大学、中山大学等六大名校跻身决赛。

参赛队伍在总功率 3kW 的限制下, 根据自己对超级计算机、对应用软件的理解分别组建各自的超级计算机系统, 大赛现场组委会统一发放测试算例文件, 各个参赛队伍现场进行 HPL、OpenForm、CPMD (car-parrinello molecular dynamics)、CP2K、Nemo 等 5 个应用算例的优化。

最终, 清华大学、国防科学技术大学获得了比赛的冠、亚军, 如图 6-11 和图 6-12 所示。



图 6-11 冠军: 清华大学



图 6-12 亚军: 国防科学技术大学

6.2 ISC 国际大学生超级计算机竞赛

6.2.1 历史沿革

在 1986 年, Hans Werner Meuer (时任德国曼海姆大学计算机中心主任, 计算机系教授) 组织成立了曼海姆超级计算机研讨会。当时参会人员有 81 人。这个研讨会随后每年都

举办,逐渐成为了全世界著名的国际超算会议和展览会,有来自全球的超算专家、厂商和研究人员。自1993年来,此研讨会成为Top500的年度发布平台之一。另外一个Top500的发布平台是SC,后面将详细介绍。截至2014年6月,此会议已经召开29届。此会议一直以来由Meuer家族精心经营(<http://www.isc-events.com>)。

ISC是电气和电子工程师协会(institute of electrical and electronics engineers, IEEE)认可的会议。一般此会议提供5天的技术议程,有来自约170家顶尖技术中心和公司的专家学者。这些技术议程包括培训、研讨会、企业环节、赞助商环节等。2015年的ISC将在德国法兰克福举办,ISC近年举办的时间、地点如表6-2所示。

表6-2 ISC近年举办的时间、地点

时间	地点	会议
2015	法兰克福，德国	ISC
2014	莱比锡，德国	
2013		
2012		
2011	汉堡，德国	
2010		
2009		
2008		
2007	德累斯顿，德国	
2006		
2005		
2004	海德堡，德国	
2003		
2002		
1993		曼海姆，德国

自从2009年,ISC组委会扩大了经营范围。新增了ISC Cloud Conference和ISC Big Data Conference,也取得了不错的效果。

6.2.2 参赛团队

ISC大学生超算竞赛是一个年轻的竞赛。至今仅举办了4届。第一届的超算比赛在2012年于德国汉堡在ISC12超算大会期间举办(<http://www.isc-events.com/isc12/student-programs.html>)。两支中国高校代表队——清华大学、国防科学技术大学与来自美国的科罗拉多大学、纽约州立大学石溪分校以及德国卡尔斯鲁厄理工学院三支超算劲旅参加了此次比赛。

此次比赛的试题包括LINPACK、OpenFoam、CPMD、CP2K、Nemo、WRF、GROMACS的应用性能测试,其中对于应用的性能评分占据80%。

最终,中国的清华大学代表队荣获“国际大学生超算竞赛总冠军”,这是中国大陆大学生在国际超级计算机竞赛中取得的第一个世界冠军。同时,国防科学技术大学代表队获得“最高计算性能奖”,如图6-13所示。纽约州立大学石溪分校设备一直在调试,获得问题处理奖。卡尔斯鲁厄理工学院获得了最受欢迎奖。此奖项是通过Facebook、Twitter等中国不方便投票的网站参与的,这给中国队伍带来了很多不便。



图 6-13 国防科学技术大学与清华大学领奖以及国防科学技术大学与清华大学比赛现场

ISC13 于 2013 年 6 月 16~19 日在德国莱比锡会议中心开幕, 来自中国、美国、德国、英国、南非和哥斯达黎加的 8 支大学生超算队伍共同参与到比赛中, 同台竞技。他们是卡尔斯鲁厄理工学院、开姆尼茨工业大学、清华大学、华中科技大学、科罗拉多大学、普渡大学、爱丁堡大学、南非高性能计算中心代表队 (<http://www.isc-events.com/isc13/student-cluster-challenge.html>)。

经过 4 天的激烈角逐终于尘埃落定, 总冠军的争夺战异常激烈, 南非超算中心与清华大学实力接近, 展开了激烈的角逐, 最终南非超算中心获得总冠军, 而清华大学惜败, 与总冠军宝座擦身而过。同样来自中国的华中科技大学将 LINPACK 最高性能奖收入囊中。在这次比赛中, 面试的分值占到了 40%, 这使得本来以十几分优势领先的清华队与冠军失之交臂。因此, 如何更好地提高中国队伍的国际表达能力是值得积极推进的培养点, ISC13 颁奖现场以及清华大学与华中科技大学全家福如图 6-14 所示。

ISC14 也是在德国莱比锡召开的 (<http://www.isc-events.com/isc14/student-cluster-competition.html>)。其参赛队伍如下。

中国: 上海交通大学、清华大学、中国科学技术大学。

南非: 高性能计算中心。

美国: 麻省理工、本特利、东北大学联盟; 科罗拉多大学。

韩国: 蔚山科技大学。

英国: 爱丁堡大学。

德国: 开姆科技大学、汉堡大学。

巴西: 圣保罗大学。

ISC14 颁奖现场以及清华大学与上海交通大学比赛现场如图 6-15 所示。



图 6-14 ISC13 颁奖现场以及清华大学与华中科技大学全家福





图 6-15 ISC14 颁奖现场以及清华大学与上海交通大学比赛现场

其中上海交通大学、清华大学、巴西圣保罗大学、韩国蔚山科技大学 4 所参赛队伍均来自 ASC 世界大学生超算竞赛。

此次比赛的赛题除常规的 HPL 外，还有 OpenFOAM（计算流体动力学软件包）、Gadget（天体宇宙学 N-体系统数值模拟软件）、Quantum Espresso（纳米尺度的电子结构与材料建模软件）；两个现场给定的神秘应用优化——HPCG 性能优化和用 Quantum Espresso 软件测试系统能效和稳定性，要求在 20min 之内运行出。值得强调的是这次给定的 Quantum Espresso 算例，除结果正确外，成功完成任务且运行过程中功耗值最小的队伍获满分，其余成功运行出结果的队伍按和第一名功耗比例给分，没有完成任务的队伍零分。这道神秘应用赛题规则强调超算系统在运算作业和时间限定情况下要求尽可能优化运行功耗，充分体现了对超算实际应用性能功耗比的重视。这种神秘应用的竞赛规则首次在国际性超算竞赛采用，极大地考验了各参赛队伍对超算系统构建的掌控能力、应用性能与功耗优化的平衡设计，以及团队协作和临场应变能力，成为本次 ISC14 超算竞赛的最受关注赛题。

经过紧张的准备和角逐，南非高性能计算中心参赛队以最低 1700W 的峰值功耗成功运行作业获得了本题的满分 16 分，来自中国的清华大学参赛队和上海交通大学参赛队分别以峰值功耗 2400W 和 2700W 成功运行作业，分别获得 11.33 分和 10.07 分，他们成为本次 11 支参赛队中唯一成功完成本赛题的 3 支队伍，出乎意料的是其他 8 支参赛队因为各种原因出错，最终都未能完成这道神秘赛题。

最终，南非超算中心以均衡发展的突出成绩取得冠军。中国科学技术大学和清华大学分获亚军和季军。赛后，清华大学队长李凯威对此次比赛进行了特殊点评，认为享受比赛比名次更重要。这说明激发年轻人对超算的兴趣无疑是极为重要的一环，让他们享受比赛，获得进步，而不仅仅是得到奖牌、奖状，也将是超算竞赛的主办方和参与方需要共同思考和解决的问题。

表 6-3 列出了 ISC13 与 ISC14 的评委名单。很明显，ISC 竞赛的评委比较稳定，绝大部分评委来自 HPCAC（HPC Advisory Council）、ISC。至今尚未发现有来自中国机构的评委。

表 6-3 ISC13 与 ISC14 评委一览表

ISC13	ISC14
Gilad Shainer (HPCAC)	Gilad Shainer (HPCAC)
Brian Sparks (HPCAC)	Brian Sparks (HPCAC)

续表

ISC13	ISC14
Hans Meuer (ISC)	Thomas Meuer (ISC)
Martin Meuer (ISC)	Martin Meuer (ISC)
Gerd Buettner (Airbus)	Gerd Buettner (Airbus)
Tong Liu (HPCAC)	Tong Liu (HPCAC)
Pak Lui (HPCAC)	Pak Lui (HPCAC)
Marcel Schneider (ISC)	Marcel Schneider (ISC)
Shelley Smith (ISC)	Shelley Smith (ISC)
Dan Olds (Gabriel Consulting Group)	Dan Olds (Gabriel Consulting Group)
Happy Sithole (Center for High Performance Computing, South Africa)	Happy Sithole (Center for High Performance Computing, South Africa)
Doug Smith (University of Colorado)	Doug Smith (University of Colorado)
Hussein Harake (HPC Advisory Council Switzerland Center of Excellence)	Hussein Harake (HPC Advisory Council Switzerland Center of Excellence)
Sadaf Alam (Swiss Supercomputing Center)	Sadaf Alam (Swiss Supercomputing Center)
Cydney Stevens (HPCAC)	Martin Hilgeman (HPCAC)
Christophe Mion (Scalable Graphics)	Christophe Mion (Scalable Graphics)
	Volker Springel (Heidelberg University)
	Filippo Spiga (University of Cambridge)

6.2.3 特点及发展趋势

ISC 大学生超算竞赛的宗旨在于将下一代的学生引荐到高性能计算（超级计算）领域。其主办单位 HPCAC 主席 Gilad Shainer 介绍在 2013 届比赛中收到了 20 余份申请。他希望这个比赛成为一个令人神往的优秀的学习机会和展示他们知识技能的机会。作为 Mellanox 鼎力支持的 HPCAC，在经费上比较有保证，这也为此竞赛的长期性提供了基础保证。

从四年比赛的情况来看，组委会、评委和比赛队伍都在不断地学习进步。ISC 也在积极关注和学习 ASC 与 SC 竞赛，如在 ISC14 竞赛的奖杯中就设立和 ASC 一样的奖状一项。

借助 ISC 会议本身的影响力和覆盖范围，ISC 竞赛有着与生俱来的国际影响力。但是组委会如果不能很好地平衡公平性和覆盖范围，将对此比赛的长期发展有根本性的阻碍。

6.3 SC 国际大学生超级计算机竞赛

6.3.1 历史沿革

SC (<http://supercomputing.org>) 超算大会是美国超算界的年会，随着它的与会者水平、数量和范围的不断扩大和提高，SC 已经成为全世界超算的年度盛会。第一届 SC 会议于 1988 年 11 月在美国奥兰多召开。当时与会人数仅逾 100 人。到 2013 年，与会人数已经达到近万人。SC 大会由 ACM SIGARCH、ACM SIGHPC 和 IEEE 计算机协会共同资助举办。

SC 的年度会议活动内容丰富，通常包括邀请报告、论文、海报、培训、展览、研讨会、

颁奖、新技术分享等。每一次活动都有详细的策划和轮转的委员会来安排管理。逐年扩大的参会人数和影响力是对 SC 质量的高度认可。

在 SC05 期间，有几位专家打赌是否学生可以用简单的工具来搭建起一台超级计算机。正是因为这样一个设想，第一届 SC 学生超算竞赛在 SC07Reno 鸣锣比赛，这一比就持续了 8 年。SC 是学生超算竞赛的发源地，正是在它的带动下，才有了 ISC 和 ASC，以及一些区域竞赛如韩国、台湾、南非等。SC 竞赛举办一览表如图 6-16 所示。

	SC13 丹佛，科罗拉多州 2013 年 11 月 18~21 日		SC12 盐湖城，犹他州 2012 年 11 月 10~16 日
	SC11 西雅图，华盛顿州 2011 年 11 月 12~18 日		SC10 新奥尔良，路易斯安那州 2010 年 11 月 13~19 日
	SC09 波特兰，俄勒冈州 2009 年 11 月 14~20 日		SC08 奥斯丁，德克萨斯州 2008 年 11 月
	SC07 里诺，内华达州 2007 年 11 月 10~16 日		SC06 坦帕，佛罗里达州 2006 年 11 月

图 6-16 SC 竞赛举办一览表

6.3.2 参赛团队

下面介绍近两届的 SC 竞赛。

SC12 (<http://sc12.supercomputing.org/content/student-cluster-competition.html>) 的竞赛于当年 10 月在美国盐湖城召开。参赛队伍是美国波士顿大学、台湾清华大学、美国太平洋大学、美国得克萨斯大学、国防科学技术大学、中国科学技术大学、美国得克萨斯理工大学、美国普渡大学。

SC12 共进行了 7 个大项 24 个小项的比赛。除常见的 LINPACK 测试，还有 LAMMPS、QMCPACK、CAM、PFLOTRAN 等 4 个应用。

美国得克萨斯大学代表队以 79.86 分的成绩获得了总冠军。图 6-17 所示为国防科学技术大学获得最高 LINPACK 奖。

SC13 (<http://sc13.supercomputing.org/content/student-cluster-competition>) 竞赛在美国科罗拉多州丹佛市举行。国防科学技术大学参赛队将代表中国出赛，与来自美国本土的田纳西大学诺克斯维尔分校、科罗拉多大学巨石城分校、太平洋大学、得克萨斯大学奥斯汀分

校、马萨诸塞格林高性能计算中心、德国的弗里德里希·亚历山大埃兰根·纽伦堡大学，以及澳大利亚 5 所高校组成的联盟队伍一起参加了此次比赛。



图 6-17 国防科学技术大学领奖

此次比赛的赛题包括 WRF、Nemo5、GraphLab 和一个神秘应用。GraphLab 是个很受欢迎的开源项目，GraphLab 开发者不断追求图计算的创新和发展，使其能迎合海量数据处理的要求。SFrame 的亮相显得低调而神秘，不过其功能不可小觑，它将 GraphLab 扩展到了表格，使其可以轻松管理 TB 级数据。

最终国防科学技术大学参赛队以每秒 8.22 万亿次的 LINPACK 成绩获得 SC13 全球大学生超级计算机竞赛最高计算性能奖。这也是国防科学技术大学继 ISC12、SC12 之后第三次荣获该项成绩冠军。得克萨斯大学奥斯汀分校则获得了总冠军，如图 6-18 所示。



图 6-18 得克萨斯大学领奖现场

6.3.3 特点及发展趋势

SC 作为国际超算的领军会议和风向标，有着超算界无可匹敌的影响力和国际认可度 (http://www.scientific-computing.com/news/news_story.php?news_id=2525)。Dustin Leverman, SC13 和 SC14 学生竞赛主席，美国橡树岭国家实验室系统主管，就是 SC07 的受益者，并且希望把这样的好机会带给更多的新同学。Brent Gorda, Intel 高性能数据部总经理，表示非常愉快与商家合作参与竞赛，可以促进校企合作，并且为学生带来工作机会。

在 SC 超算竞赛的影响下，不少学生获得了知名企业的实习机会，并且让更多学生了解到高性能计算。他们也鼓励参赛的学生与更多没有机会参赛的学生交流心得。不少高校也开始设置高性能计算相关课程，如美国科罗拉多大学波尔得分校和中国的中山大学。

第 7 课 ASC 世界大学生超级计算机竞赛历史与展望

7.1 2012 年中国大学生超级计算机竞赛

7.1.1 首次进入中国的国际赛事

2012 年,在科技部、教育部的指导下,在浪潮集团与国际超级计算机大会组委会(ISC)、国际高性能计算咨询委员会(HPC-AC)的共同推动下,“中国大学生超级计算机竞赛暨 ISC 国际大学生超级计算机竞赛中国区选拔赛”在中国拉开序幕。该项赛事是与 SC 大学生超算竞赛并列的两大全球性赛事之一,得到了国家和科技界的大力支持,中国工程院金怡濂、沈绪榜、何新贵、陈左宁等四位资深院士分别担任大赛专家委员会主席和副主席,科技部高新司、清华大学等给予了全力支持。

7.1.2 大赛进展

赛事自 2012 年 2 月拉开战幕,截至 3 月 2 日,共有 27 所高校报名,经过激烈角逐,3 月 22 日,经过竞赛评审委员会客观公正的评定,国防科学技术大学、中国科学技术大学、清华大学、武汉大学、山东大学、中山大学六大名校跻身决赛。4 月 16 日,六支参赛队伍齐聚北京进行决赛,4 月 19 日,根据决赛方案的测试结果和竞赛委员会根据现场呈现打分的结果,清华大学、国防科学技术大学最终赢得 6 月于德国汉堡举办的 ISC12 国际大学生超级计算机竞赛的两张入场券,代表中国与世界超级计算强队同场竞技。

7.1.3 大赛三要素:在校大学生、国际接轨、实用技能

本次大赛的目的就是以国际赛事的规则、算例来考察在校大学生超级计算机实用技能。参赛者要求以高校为基本参赛单位,每所高校只能有一支队伍参赛,每支参赛队伍由 6 名学生与 1 名指导老师组成,参赛学生必须为具有该校正式学籍的全日制在校本科生。

参赛队伍在总功率 3kW 的限制下,根据自己对超级计算机、对应用软件的理解分别组建各自的超级计算机系统,大赛现场组委会统一发放测试算例文件,各个参赛队伍现场进行相关的应用测试和程序优化以及竞赛呈现。大赛所用的 HPL、OpenForm、CPMD、CP2K、Nemo 等 5 个应用算例,不仅与 ISC 国际大学生超级计算机竞赛的比赛项目相同,而且上述算例在实际应用中也最为广泛,大赛的实用性导向可见一斑。

7.2 2013 年亚洲大学生超级计算机竞赛 (ASC13)

7.2.1 ASC13 定位及宗旨

该赛事由中国倡议成立,日本、俄罗斯、韩国、新加坡、泰国、中国台湾、中国香港等国家和地区的超算专家和机构联合发起和组织,由亚洲超算协会联合浪潮集团主办。大赛旨在推动亚洲国家及地区间超算青年人才交流和培养,提升超算应用水平和研发能力,发挥超算的科技驱动力,促进亚洲科技与产业创新。

作为亚洲首次举办的国际性超算赛事,ASC13 吸引了亚洲高校的热烈关注和积极响应,报名参赛的高校全面覆盖东亚、西亚、北亚和南亚,包括中国、俄罗斯、韩国、印度、哈萨克斯坦、沙特阿拉伯、中国台湾、中国香港等在内的多个国家及地区,共计 43 所高校报名参赛。

参赛高校除了前面在第 6 章提到的初赛十强以外,还有曾获得美国 SC12 国际大学生超算大赛“最高计算性能奖”的国防科学技术大学等实力强队,来自亚洲各国及地区的韩国釜庆大学、莫斯科国立鲍曼技术大学、印度孟买大学、哈萨克国立大学、沙特国王大学等著名高校,以及来自中国大陆的南京大学、武汉大学、同济大学、复旦大学、西北工业大学等国内重点高校。

7.2.2 大赛的赛制、赛程与组织架构

本次赛事由浪潮集团主办,联合国际超级计算机大会组委会、国际高性能计算咨询委员会共同承办。高效能服务器与存储技术国家重点实验室、上海交通大学两家单位协办。中国科技部高新司、教育部科技司、上海超级计算中心、中国科学院计算机网络信息中心超级计算中心、国家超级计算天津中心、济南中心、长沙中心和深圳中心给予了大力支持和帮助。

赛程分为三个阶段:第一阶段是高校报名,组织参赛团队,报名将截止于 1 月 25 日;第二阶段初赛将在 2 月进行,根据竞赛的命题要求,各参赛队提交相关书面方案,由评审委员会评选出进入决赛的团队;第三阶段决赛的举办时间为 4 月,进入决赛的团队将在浪潮提供的超级计算机设备上,在满足竞赛规则的要求下,根据自己对应用的优化理解组建各自的超级计算机系统,现场进行相关的应用测试和程序优化以及竞赛呈现,由评审委员会专家根据现场实际运算结果,评选出最终名次。根据规则要求,每个高校只能有一支参赛队,且参赛队伍的 6 名学生必须是本科生。

赛制要求各参赛队在主办方提供的超级计算机设备上自行构建参赛平台,并在 3kW 的额定功率约束下自行设计超级计算机系统方案,进行应用优化测试,并特别设置了体现亚洲特色的基于期权定价应用 BSDE 的 MIC 众核并行优化试题,最终评委根据各个参赛队伍提交的书面文档进行打分,评出 10 支初赛胜出队伍。

7.2.3 进入 ASC13 的十强

进入 ASC13 的十支强队在高性能计算方面都有着比较丰富的经验。

阿卜杜勒·阿齐兹国王大学（沙特阿拉伯王国）：沙特阿拉伯王国的超级计算机发展深受国王的重视，它们在石油、生命、气候等方面有着深入的超算应用研究。

圣彼得堡大学（俄罗斯）：圣彼得堡大学开设了超算相关课程，超算教学健全。本次代表俄罗斯圣彼得堡大学参赛的学生都接受过高性能计算的训练。

蔚山科学技术大学（韩国）：重视超算学科的教育，在冬季和夏季学校都组织并行计算的课程，吸引了一批同学的参加。本次代表蔚山科学技术大学参赛的队员也曾在 KSC 大学生超算竞赛中取得优异成绩。

台湾清华大学（中国台湾）：曾多次夺得美国 SC 大赛的总冠军，在海内外超算界引起关注。

香港中文大学（中国香港）：香港中文大学有着健全的超算教育体系，学校设置计算机科学中心，服务于全校超算相关的学科研究，在本科阶段和研究生阶段，学校都分别设置相应的超算课程。

清华大学：清华大学拥有与浪潮集团共同研制的运算峰值超百万亿次的超级计算机，2012 年，清华大学代表队在 ISC12 全球大学生超算大赛中夺得总冠军，成绩优异震惊了超算业界。

上海交通大学：拥有健全的超算相关教育模式，学校在本科生教育中开设超算相关课程，邀请国际上高性能计算领域著名专家讲课，为本科生系统地介绍高性能计算领域的相关内容，学校还设置高性能计算奖学金，对成绩优异的学生进行奖励。这次竞赛由上海交通大学高性能计算中心组织，得到了网络信息中心和电子信息与电气工程学院的大力支持。

国防科学技术大学：他们曾蝉联 ISC12、SC12 超算大赛“最高计算性能奖”，并一再打破该奖在超算大赛的纪录，学校也建设了非常优越的超算学习环境。

华中科技大学：华中科技大学高性能计算中心开设有并行处理、并行程序设计等高性能计算相关的必修课程，并定期组织用户培训，实验室学生自发组成了 HPC 兴趣小组，参加过金融计算等与超级计算相关的比赛，获得了不错的成绩。

中山大学：拥有中国华南高校领先的超算平台，对于高性能计算在学科建设和科研创新中的应用非常重视，他们一直注重人才培养，因此在校内通过课程教育、专业培训和兴趣小组等形式，进行超算专业人才的培育。

7.2.4 评审委员会对参赛高校的综合评价

评审委员会对参加 ASC13 的参赛队给出了非常高的评价。

严谨系统的方法论：从各校提交的整体方案来看，这些高校队伍对高性能计算具有比较深入的理解，从系统平台建设方案到实际应用软件的运行优化，再到众核平台上的代码加速，都表现出了严谨系统的方法论和高超的技术技能，通过实验提出的结论也具有总结性和概括性。

系统架构成熟：部分高校的系统平台方案比较成熟，架构新颖，充分考虑了应用的特征差异、系统性能与功耗比，在理论上有助于极大地发挥机器性能。

性能测试优良：各高校提交的测试及优化方案分别从系统、硬件、软件参数等方面给出了详尽的介绍；设计使用的性能调优方法非常全面，获得了较好的优化性能，效果明显；对其中的期权定价应用 BSDE，给出了程序代码的详细分析和热点测试，设计了合理的 MIC 并程序，采取了全面、有效的优化手段，循环进行测试、分析和优化工作，达到较高的加速比。

应用优化深入到算法及代码：很多高校对平台整体集群软、硬件架构，系统和编译优化等都有非常准确地把握，方案中同时给出了对可能出现的问题的预估及其解决方案；部分高校对应用软件从算法和代码的角度进行了较为详细的分析、阐述，并在算法和代码的层面进行了一定程度的改进。

7.3 2014 年世界大学生超级计算机竞赛（ASC14）

7.3.1 82 所世界高校同台竞技，创历史之最

2014 年 ASC 世界大学生超级计算机竞赛（ASC14）由亚洲超算协会、中山大学和浪潮集团共同主办。大赛报名启动于 2013 年 11 月，预赛在 2014 年 1~2 月举行。总决赛于 2014 年 4 月 21~25 日在广州中山大学进行。

本次大赛共有来自北美洲、南美洲、非洲、亚洲、欧洲五大洲、海内外 82 所高校报名参赛，创下了全球超算赛事参与数量最多、覆盖国家和地区最广的新纪录。

在 82 所参赛高校中，不乏世界强队。清华大学连续获得 ASC12、ISC12、ASC13 总冠军；中山大学在应用优化方面有较强能力，曾在 ASC13 中取得过 GROMACS 算例的最佳优化成绩；获 SC、ISC “最高计算性能奖”三冠王的国防科学技术大学；取得 ASC13 MIC 应用优化性能奖和 ISC13 最高计算性能奖的华中科技大学；台湾清华大学曾蝉联两届美国 SC 大赛冠军并获 ASC13 大赛亚军；最早就参加美国 SC 超算竞赛并获得不俗成绩的美国普渡大学比赛经验丰富，是公认的世界级劲旅；美国科罗拉多大学曾多次参加 SC、ISC 竞赛并取得好成绩；南非西开普大学、韩国蔚山科学技术大学分别是南非和韩国国家超算竞赛的冠军队。

7.3.2 全球超算专家云集

ASC14 超算竞赛的专家委员会主席由 LINPACK 创始人、Top500 发起者之一美国田纳西大学教授 Jack Dongarra，美国劳伦斯·利弗莫尔国家实验室计算副主任 Dona L. Crawford 等世界顶尖超算专家联合担任。中国科学院院士沈绪榜、中国工程院院士何新贵、中国工程院院士陈左宁，美国 ACM 院士、日本东京工业大学教授 Satoshi Matsuoka，以及俄罗斯科学院会员、计算科学总监 Abramov Sergei M. 出任副主席。

委员包括（排名不分先后）北京航空航天大学教授、中德联合软件研究所所长、863 计划重点专项专家组组长钱德沛，中国工程院院士、国防科学技术大学教授、计算机学院

院长廖湘科, 华中科技大学教授、计算机学院院长、ChinaGrid 专家组组长金海, 国家超级计算天津中心主任刘光明, 中国科学院软件研究所研究员孙家昶, 北京应用物理与计算数学研究所研究员袁国兴, 上海超级计算中心主任奚自立, 国家超级计算济南中心主任顾卫东, 国家超算深圳中心副主任张月光, 清华大学高性能计算研究所所长杨广文, 中国科学院软件研究所研究员张云泉, 长沙超级计算机中心主任李肯立, 香港知识管理学会副总裁、香港理工大学教授 Tsui Eric Y.H., 日本筑波大学计算机科学系教授 Boku Taisuke, 日本工业大学信息学院院长 Oyanagi Yoshio Lee, 日本工学院大学信息系主席 Oyanagi Yoshio, 韩国科学技术信息研究院 (KISTI) Woo Joon, 新加坡 A*star 计算资源中心高级主任 Michalewicz Marek, 上海交通大学超算中心顾问 See Simon, 泰国农业大学计算科学与工程中心主席 Uthayopas Putchong, ACIS 实验室主任 Fortes, 国际数据公司 (IDC) 副总裁 Earl C. Joseph II, 印度海德拉巴大学教授 Rajeev Wankar, 美国 AT&T 杰出学者 José A.B. Fortes, 英国科学和技术设施委员会实验室计算科学与工程中心副主任、瑞士卢加诺大学信息学院副教授 Walter Binder, 美国国家高级计算机基础设施联盟执行主任 Peter Arzberger, 香港大学信息技术中心 IT 助理总监 Kwan Wing Keung。

竞赛指导委员会主席由国家科技部高新司司长赵玉海担任, 成员包括国家科技部高新司副司长杨威武, ASC 发起人、高效能服务器与存储技术国家重点实验室主任、浪潮集团首席科学家王恩东, 清华大学教授、中国计算机学会理事长郑纬民, 中国科学院计算机网络信息中心超级计算中心主任迟学斌, 香港中文大学教授、顺兴高级工程研究院副院长 ChingPak-Chung, 国际超级计算大会执行总裁 Martin Meuer, HPCAC 咨询委员会主席 Gilad Shainer。

北京应用物理与计算数学研究所副所长、研究员莫则尧担任 ASC14 的竞赛评审委员会主席。

7.3.3 初赛：四大赛题遴选超算精英

ASC14 初赛将完成 ASC14 竞赛组委会给定的四大命题, 参赛队伍将在总功耗 3kW 的限制条件下做出超算系统设计方案, 完成 HPL、Quantum Espresso 和三维纵横波分离的弹性波方程模拟方法 (3D_EW) 共 3 项应用测试与优化。

参赛队伍首先需要展现超算系统的设计构建能力。在总功耗 3kW 的限制下, 各参赛队伍对计算性能、功耗控制、高速网络、加速器、软件环境综合考量的超算系统设计方案的综合性与创新性, 将成为评审委员会评估方案优劣的重要依据。

HPL 测试是超算 Top500 排名的重要依据, 将主要考察参赛队对硬件平台的浮点计算性能优化能力。参赛队伍要取得好成绩, 需要对超算系统体系结构、多级存储一致性、高速网络、算法优化等方面进行全方位的综合研究, 在混合架构、单一架构等方案之间比较选择, 并结合性能功耗比加以反复优化和测试。竞赛组委会特意指出鼓励各参赛队伍从算法和代码角度深入分析, 从底层算法上对 HPL 软件进行优化。

Quantum Espresso 应用软件是目前应用较为广泛的开源量子分子动力学计算软件, 包含了 PWscf、FPMD、CP、PWgui、Atomic 五大功能模块, 包括电子自洽计算、晶格动力学计算、后续数据处理、电子输运性质计算、分子动力学等模块, 能够完成超现实的材料性能模拟, 主

要应用在新材料开发领域。目前量子分子动力学计算软件普遍具有扩展性较差的弱点，而 Quantum Espresso 可以根据不同的应用体系扩展上千核到上万核。

3D_EW 是 ASC14 指定的 MIC 应用优化试题，参赛队在组委会搭建的超大规模 CPU+MIC 异构并行计算集群上完成应用优化。3D_EW 是与 ASC14 报名同时启动的“中国好应用”征集活动中最终入选的中国自主研发应用软件。该软件由中国石油东方地球物理公司自主研发，在石油勘探等领域具备广泛应用前景。

7.3.4 决赛：七大赛题挑战智慧极限

前 16 强参赛队将在总决赛现场在 3kW 功耗约束下自行设计构建超级计算机并完成要求的各项应用优化赛题比试，最后再完成竞赛呈现答辩。

在初赛四大命题的基础上，ASC14 总决赛增加了 LICOM、SU² 两大应用，并在决赛当天公布一项神秘应用。

LICOM 来自中国科学院大气物理研究所自主研发的海洋环流模式，是一个 Eta 坐标原始方程海洋环流模式，也是大气科学和地球流体力学数值模拟国家重点实验室（LASG）海气耦合模式的海洋分量，能够很好地重现合理的大尺度海洋环流特征。LICOM 是 LASG 科学家发展的第三代全球海洋环流模式，并且已经广泛地应用于研究海洋环流、气候变化模拟、海气相互作用等多个研究领域。

SU² 是由斯坦福大学开发的一个开源的计算流体力学软件，可进行偏微分方程组求解，以及飞行器气动力学研究及外形优化，支持并行计算。同时，SU² 使用 C++ 语言编写，具有较高的可读性。

7.3.5 黑马异军突起

首次参赛的新加坡南洋理工大学和太原理工大学表现极为出色，分列海外海内队伍初赛成绩第一名，双双成为本次竞赛最大黑马。

新加坡南洋理工大学在开源量子分子动力学计算软件 Quantum Espresso 赛题中表现出对凝聚态物理的深刻认识，从物理的角度出发全方位把握应用优化。

太原理工大学则取得了三维纵横波分离的弹性波方程模拟方法 3D_EW 在 CPU+MIC 异构并行计算赛题的最好成绩，应用运行时间从原始串行的 9000 多秒大幅并行优化到 21s，在 4 个运算节点上实现 400 余倍的惊人性能加速比。

太原理工大学取得 MIC 应用优化第一名的成绩并非源自侥幸。据组委会统计的 MIC 大赛平台登录时长显示，太原理工大学的登录时长达到了惊人的 500h，为参赛的 82 所大学中最高的。从 1 月 2 日初赛开始到 2 月 27 日初赛方案提交截止，初赛总共 56 天共计 1344 小时，其中包括春节、元宵、寒假等假期，这意味着太原理工大学参赛队花在大赛上的时间超过三分之一。

7.3.6 e Prize 计算挑战奖激励全球超算天才

为鼓励青年超算人才成长，ASC 组委会宣布将设立 e Prize 计算挑战奖，颁给竞赛中获得超算应用最高扩展性与性能优化的队伍。该奖项旨在鼓励青年人才勇于解决超算应用难

题，敢于挑战计算性能极限，激励青年人才利用超级计算机推动科学与工程领域实现应用突破。

e 是科学中最重要的自然常数，同时也代表着人类下一个超算性能挑战目标 exascale 百亿亿次，ASC 希望 e Prize 计算挑战奖能成为青年人才的戈登·贝尔奖 (Gordon Bell prize)。

7.4 2015 年世界大学生超级计算机竞赛 (ASC15)

7.4.1 六大洲 152 支参赛队，再创历史新高

ASC15 共有来自北美洲、南美洲、大洋洲、非洲、亚洲、欧洲六大洲的 135 所高校报名参加，参赛队伍数量为 152 支，创下了全球超算赛事参与数量最多、覆盖国家和地区最广的新纪录。

首次参赛的国家增加了 6 个，日本、德国、法国、荷兰、印度、印度尼西亚等均首次参加 ASC15。同时，世界各国的著名高校纷纷报名参加，如全球排名第一的麻省理工学院、中国大陆和巴西各自排名第一的清华大学和圣保罗大学等，如表 7-1 所示。

表 7-1 ASC15 知名高校

ASC15 知名高校	国家	排名
麻省理工学院	美国	世界第一
南洋理工大学	新加坡	新加坡第二
清华大学	中国	中国第一
万隆理工学院	印尼	印尼第二
博特拉大学	马来西亚	马来西亚排名第五
圣彼得堡大学	俄罗斯	俄罗斯第二
圣保罗大学	巴西	巴西第一
莱顿大学	荷兰	荷兰第二
东京工业大学	日本	日本第四
台湾清华大学	中国台湾	中国台湾第二
香港中文大学	中国香港	中国香港第二

注：数据来源于 2015QS 世界大学排名

7.4.2 全球超算专家云集

ASC15 超算竞赛的专家委员会主席由 LINPACK 创始人、Top500 发起者之一美国田纳西大学教授 Jack Dongarra，美国劳伦斯·利弗莫尔国家实验室计算副主任 Dona L. Crawford 等世界顶尖超算专家联合担任。中国科学院院士沈绪榜、中国工程院院士何新贵、中国工程院院士陈左宁、美国 ACM 院士、日本东京工业大学教授 Satoshi Matsuoka，以及俄罗斯科学院会员、计算科学总监 Abramov Sergei M.出任副主席。

委员包括（排名不分先后）华中科技大学教授、计算机学院院长、ChinaGrid 专家组组

长金海,中国科学院软件研究所研究员孙家昶,北京应用物理与计算数学研究所研究员袁国兴,上海超级计算中心主任奚自立,国家超算深圳中心副主任张月光,清华大学高性能计算研究所所长杨广文,中国科学院软件研究所研究员张云泉,香港知识管理学会副总裁、香港理工大学教授 Tsui Eric Y.H., 日本筑波大学计算机科学系教授 Boku Taisuke, 日本工业大学信息学院院长 Oyanagi Yoshio Lee, 日本工学院大学信息系主席 Oyanagi Yoshio, 韩国 KISTI Woo Joon, 新加坡 A*star 计算资源中心高级主任 Michalewicz Marek, 上海交通大学超算中心顾问 See Simon, 泰国农业大学计算科学与工程中心主席 Uthayopas Putchong, ACIS 实验室主任 Fortes, IDC 副总裁 Earl C. Joseph II, 印度海德拉巴大学教授 Rajeev Wankar, 美国 AT&T 杰出学者 José A.B. Fortes, 英国科学和技术设施委员会实验室计算科学与工程中心副主任、瑞士卢加诺大学信息学院副教授 Walter Binder, 美国国家高级计算机基础设施联盟执行主任 Peter Arzberger, 香港大学信息技术中心 IT 助理总监 Kwan Wing Keung。

竞赛指导委员会主席由国家科技部高新司司长赵玉海担任,成员包括国家科技部高新司副司长杨威武,ASC 发起人、高效能服务器与存储技术国家重点实验室主任、浪潮集团首席科学家王恩东,清华大学教授、中国计算机学会理事长郑纬民,中国科学院计算机网络信息中心超级计算中心主任迟学斌,香港中文大学教授、顺兴高级工程研究院副院长 ChingPak-Chung, 国际超级计算大会执行总裁 Martin Meuer, HPCAC 咨询委员会主席 Gilad Shainer。

北京应用物理与计算数学研究所副所长、研究员莫则尧担任 ASC15 的竞赛评审委员会主席。

7.4.3 初赛:从分子到宇宙的“两极”考验

ASC15 超算竞赛初赛试题分别为 HPCC、纳米尺度的分子动力学(NAMD)和 Gridding。ASC15 是国际首个用 HPCC 代替 HPL 的竞赛。另外两道试题一个研究分子级的微观世界,一个研究宇宙级的宏观世界。

HPCC:目前国际上把亚洲 ASC、美国 SC 和德国 ISC 并称为三大国际超算竞赛,HPCC 是一套面向高性能计算机的综合测试程序包,能够测试高性能计算机系统多个方面的性能,包括处理器速度、存储访问速度和网络通信速度,对各种应用都有一定的代表性和参考价值。

NAMD:能够在大规模并行计算机上快速模拟大分子体系,广泛应用于生命科学等领域,如抗癌药物开发等。NAMD 曾获得过 2002 年超算界最高荣誉的戈登·贝尔奖和 2012 年的西德尼·费恩巴赫奖,能够从几百个内核的典型模拟扩展到超越 500000 内核的最大的模拟。

Gridding 是平方公里射电望远镜阵列(square kilometer array, SKA)数据处理过程中非常重要的一个步骤,也是最耗时的步骤之一。为了生成天空图像,科学家需要对射电望远镜采集的数据进行一系列的操作,但望远镜产生的数据是不规则的,需要被映射到规整的二维网格上,随后才能进行傅里叶变化,这个映射的过程就是 Gridding。本次 ASC15 竞赛要求参赛队在 MIC 平台上完成 Gridding 的应用移植。

7.4.4 决赛赛题：紧贴应用

参赛队伍将在总功耗 3kW 的限制条件下做出超算系统设计方案,在预赛两大赛题外,完成 HPL、WRF-Chem、PalaBos 应用赛题,并在决赛当天现场公布一项神秘应用(HPCG),最后再完成竞赛呈现答辩,决赛赛题涉及生命、雾霾、汽车外观设计等应用,紧贴实际应用。

HPL 测试是超算 Top500 排名的重要依据,主要考察参赛队对硬件平台的浮点计算性能优化能力。参赛队伍要取得好成绩,需要对超算系统体系结构、多级存储一致性、高速网络、算法优化等方面进行全方位的综合研究,在混合架构、单一架构等方案之间比较选择,并结合性能功耗比加以反复优化和测试。竞赛组委会鼓励各参赛队伍从算法和代码角度深入分析,从底层算法上对 HPL 软件进行优化。

WRF-Chem 是目前最先进、应用最广泛的空气质量数值预报模式,是世界各国预测雾霾和 PM2.5 颗粒、PM10 颗粒、二氧化硫等有害空气成分变化的主要技术手段。

PalaBos: 计算流体力学软件,主要依靠计算机进行物体与气流方面的模拟计算工作,常见的应用领域如飞机、汽车等的外观流线设计,有助于在这类产品开发时尽可能地预计空气对物体的阻力。

HPCG 为高性能计算三大基准测试之一。

7.4.5 汇聚全球青年精英,挑战宇宙级“难题”

ASC15 超算竞赛和世界最大的射电阵列望远镜 SKA 合作,汇聚全球超算天才挑战 SKA 最复杂的计算过程 Gridding。

SKA 是由中国、澳大利亚等多个国家参与的国际天文学项目,其目标是建造全球最大的综合孔径射电望远镜,来解决人类认识宇宙的一些基本问题,特别是关于第一代天体如何形成、星系演化、宇宙磁场、引力的本质、地外生命与地外文明、暗物质和暗能量等。

SKA 将由大约 3000 个碟形天线组成,每个碟形天线的直径超过 15m,是家用的卫星接收天线的 10 倍,其灵敏度比目前最大厘米波综合孔径望远镜 JVLA 高 50 倍、搜寻速度高 10000 倍。SKA 每秒采集的数据量计划将大于 12TB,相当于 2013 年年底中国互联网国际出口带宽的 3.5 倍,是全球最大的“大数据”项目。SKA 将采集到的庞大数据动用超大规模的超级计算机来处理,这个项目对高性能计算性能的需求比全球 Top500 中所有超级计算机加起来的计算力还要高。

ASC15 采用 SKA 中的 Gridding 作为竞赛试题,这是 SKA 数据处理部分最复杂的过程之一。目前,Gridding 的串行处理版本速度无法达到理想状态,SKA 国际大科学工程的科学家希望能够通过 ASC15 大赛,对 Gridding 进行 MIC 众核并行优化,而本次经过各参赛队伍优化后的结果,反馈给 SKA 组织参考采用。

7.4.6 总决赛设 25 万大奖

ASC15 总决赛在中国山西的太原理工大学正式拉开帷幕,这也是 ASC 超算大赛首次将决赛场地设置在中国内陆城市。总决赛设置总冠军、亚军、e Prize 计算挑战奖、最高

算性能奖、最佳呈现奖和应用创新奖等六项大奖，奖金总金额约 25 万元人民币，这也让 ASC15 大赛成为有史以来奖金额度最高的超算赛事。

7.4.7 《三体》作者刘慈欣“出现”在总决赛闭幕式

ASC15 与全球最大的天文学国际合作项目 SKA 合作，将该项目中计算耗时最多、最复杂的数据处理软件 Gridding 的优化作为大赛“e Prize 计算挑战奖”赛题。

著名科幻作家刘慈欣在其科幻名著《三体》中对宇宙本质和未来命运的大胆想象，或许也将通过 SKA 得到更多的验证，因此对宇宙的好奇与探究让刘慈欣与超算大赛结缘，刘慈欣专门为总决赛闭幕式录制“视频讲话”，这也带来了一场“科学与科幻的火花碰撞”。

第 8 课 ASC 世界大学生超级计算机竞赛规则解析

2014 年世界大学生超级计算机竞赛（ASC14）是首次面向全球各大洲高校在校大学生的超算竞赛，报名参赛高校来自各大洲，共一百多所高校。下面将以 ASC14 为例，详细解析初赛和决赛规则。

8.1 详细解析 2014 年世界大学生超级计算机竞赛初赛规则

ASC14 初赛阶段，各个参赛队需要根据组委会给定的命题，在规定时间内完成书面方案和测试优化的结果，通过竞赛评审委员会的评定才能入围决赛。ASC14 共有上百所高校参加初赛，最终择优 16 支队伍进入决赛，竞争非常激烈。如何从初赛中脱颖而出，仔细研读比赛规则必不可少。

8.1.1 初赛介绍

时间：2014 年 1 月 2 日～2 月 27 日上午 9:00。

形式：竞赛评审委员会给定统一命题，竞赛组织委员会发送给报名参赛队伍，报名参赛队在规定时间内按命题要求提交书面方案和测试优化结果，由竞赛评审委员会统一评定入围决赛队伍。

8.1.2 初赛规则解析

（1）首先要明确初赛截止时间，在规定的时间内提交组委会要求的内容，其中包含书面方案和测试优化结果。因此对竞赛委员会命题必须仔细研读！

（2）初赛要求提交书面方案，书面方案包含三大部分。

① 参赛部门的超级计算技术水平（5 分）。

介绍学校超算平台，介绍学校的超算教育（课程、培训班、兴趣小组）、应用情况、超算成果等。超算平台介绍可以通过学校的网络中心，超算教育可以通过教务处等获得相关信息；做好调研、沟通，各项都有涉及或介绍即可。

② 参赛部门的竞赛组织情况（5 分）。

本部分包含两个内容，一是介绍竞赛组织和参与情况，二是参赛队伍构成部分介绍。首先介绍获取竞赛信息的时间、方式，学校对竞赛的重视，老师同学的响应，人员征集等各个层面的情况；其次对指导老师以及各个参赛学生一一从年龄、性别、专业背景、兴趣爱好等方面进行介绍。

③ 参赛技术方案（90 分）。

该部分是整个书面方案中权重最大的一项，重点考察参赛队对高性能计算的理解程

度。不仅包含硬件层考察、系统层考察,更重要的是要让参赛学员了解应用的重要性、应用优化的必要性。

技术方案包含系统平台建设方案、HPL 测试方案、Quantum Espresso 测试方案、CPU+MIC 的应用优化四个方面。

a. 系统平台建设 (15 分)。

必须要求在 3kW 的功耗下搭建整套系统,在这一前提条件下就要先计算组成一套机器的每个部件的功耗,如 CPU 的功耗、每个内存条的功耗、机器中内存条的个数、HCA 卡功耗、硬盘功耗、硬盘个数;并从实际应用考虑是否使用加速卡;最后各个部件确定后,一台机器的功耗可以预估到,在 3kW 功耗限制下,计算出机器的个数。最后确定出系统方案。该部分不仅需要全局考虑,还要考虑各个细节。不仅要了解硬件特点,还要掌握系统知识。各个参赛队运筹帷幄,方可拿出优秀的系统平台建设方案。

b. HPL 测试方案 (15 分)。

要求在通过正确性检查的前提下,以浮点运算性能最高为目标。

测试方案首先要求描述测试 HPL 的操作系统、编译器、数学库、MPI 等。并给出测试的方法,以及对 HPL 进行优化。评审委员会建议从算法原理层进行讨论展开。这是考察参赛队对 HPL 的算法、模型的掌握程度,参赛队如果能从这一层级入手,对 HPL 的测试和优化大有裨益。在理解掌握算法后,从代码层级优化也将成为可能。因此针对该部分的方案最好能给出 HPL 测试的软硬件环境、HPL 基本原理分析、HPL 的测试方法、测试结果、性能分析、优化方法、总结等,这样可较强地涵盖方案要求。

c. Quantum Espresso 测试方案 (20 分)。

要求在通过正确性检查的前提下,以运行时间最短为目标。

测试方案也是要求描述测试时的操作系统、编译器、数学库、MPI 等,并给出测试、优化的方法。测试方法中一般需要详述软件编译的方法或步骤,列出编译使用的关键参数。Quantum Espresso 不像 HPL 是评估计算机的计算性能,它是一款材料科学类的应用软件,用于解决实际的科研问题。因此参赛队掌握软件的功能、用途是基础要求。在掌握基本的功能后,如果能从原理或算法层级进行研究则可以大大提高测试和优化的效率。针对该部分的方案最好能给出 Quantum Espresso 测试的软硬件环境、Quantum Espresso 简介、Quantum Espresso 基本原理分析、并行方法分析、测试方法、测试结果、性能分析、优化方法、总结等,这样可较强地涵盖方案要求。

d. CPU+MIC 程序设计与实现 (40 分)。

即在通过正确性检查的前提下,完成组委会给定代码的 CPU 并行化或 CPU+MIC 并行的设计与优化。组委会为各个参赛队提供了测试平台,包含纯 CPU 集群并行和 CPU+MIC 集群。参赛队可以在这两种平台中任选一个进行代码的并行化和优化。可以采用 MPI、OpenMP、PThread、OpenCL 等 MIC 支持的并行编程模型;若采用 CPU+MIC 集群并行, MIC 程序必须采用 offload 模式;由于涉及代码的并行化和优化,参赛队必须对物理模型、数学算法、并行程序设计有所掌握。因此在方案中要包含应用的介绍、物理模型的分析、数学方法分析、代码分析、向量化分析、并行编程实现过程、并行后的测试,验证、优化的思路,方法等。

8.1.3 初赛规则总结

初赛要求提供的是书面方案,并且规定了方案的内容和不同部分的权重。一些是基本信息收集,可以视为必得分项。技术方案部分占总比分的90%,包含四部分,考察点不同,需要参赛队进行全方位的准备。不仅考察参赛队的计算机水平,还考察实际应用的能力。参赛队可以按照个人所长进行分工,各司其职,准备各自负责的方案,讨论汇总,最终整理出风格统一的完整的技术方案。

8.2 详细解析 2014 年世界大学生超级计算机竞赛决赛规则

通过对预赛中全球各大洲一百多所高校的书面方案及其中所述测试的优化结果,最终选拔出16所高校参加2014年世界大学生超算竞赛(ASC14)决赛。

8.2.1 决赛介绍

时间:2014年4月21日~4月25日。

形式:各队按照3kW功耗要求(具体见规则解析)提出所需硬件方案,由组委会按方案提供相应硬件;各队有两天时间进行硬件搭建、操作系统及其他相应配置,比赛具体内容由竞赛评审委员会在比赛时间统一发放(LINPACK除外),在规定时间内及3kW功耗要求下进行计算,计算结果正确,且时间越短者,分数越高;此外,在所有应用比赛结束后,进行竞赛呈现,各队以PPT展示竞赛思路、竞赛策略、优化方法。竞赛评审委员会按照应用成绩和呈现成绩,最终评定决赛结果。

8.2.2 决赛规则解析

(1)首先要明确决赛时间安排,在规定的时间内提交组委会要求完成的内容,仔细阅读相关应用算例比赛时间和提交最终结果的方法。

(2)决赛主要包括四大部分。

① 硬件搭建、操作系统安装、并行环境配置、应用软件编译。

a. 在21日和22日的决赛准备期间,如果对机器的默认硬件(CPU、内存、硬盘)进行更改,请及时通知工作人员并做好配件保管,在比赛结束后需恢复原状。

b. 自23日决赛开始后,硬件配置不可更改,机柜上架的机器必须全部开机使用。如果发生硬件故障,会有备机和备件可以替换,备机和备件由全部参赛队伍共享,队伍不能单独申请备机。

c. 比赛规定,参赛队现场搭建集群,每个参赛队各自有一个供电PDU(power distribution unit),由组委会监控其功耗。在决赛期间,各队的实际功耗不得超过3kW。在决赛过程中,实际功耗连续超过3kW达1min以上时,各队伍需将应用停掉,当时的运行结果作废。因为硬件搭建、操作系统安装、并行环境配置、应用软件编译是比赛的基础,所以各队应在环境许可的条件下充分演练、准备。

② LINPACK、Quantum Espresso、SU²、LICOM、神秘应用比赛规则。

a. 应用算例提供内容及规则。

LINPACK: LINPACK 测试将于 23 日 08:00~11:00 进行, 最晚 11:00 务必提交测试结果, 各队也可提前提交结果, 但已提交的 LINPACK 结果不可再修改或重新提交 (提交 LINPACK 结果后即可收到 Quantum Espresso、SU² 应用的算例)。

Quantum Espresso: 将提供赝势文件, 输入参数.in 文件, 其中除了用于指定赝势文件位置的目录和指定临时文件输出的目录两个选项外, 一律不允许修改。

SU²: 将提供*.su² (或*.cgns 文件) 及*.cfg 文件, 所有算例文件 (*.su²、*.cgns、*.cfg) 均不允许修改。

LICOM: 将提供模式运行的基本数据文件 (INDEX.DATA、TSinitial、MODEL.FRC 和 dncoef.h1) 和 namelist (ocn.parm), 如模式需要 continue 运行, 则提供 rpointer.ocn 文件, 而相应的重启文件 (fort.22.yyyy-mm-dd) 需要从前一个个例得到。前 3 个例子使用编译好的 licom2 可执行文件即可, 后 3 个例子需要替换程序 param_mod.F90 重新编译。其中除了与目录相关或并行计算核数相关的选项外, 一律不允许修改。

神秘应用: 神秘应用将在 4 月 24 日 08:00 提供软件、算例和 readme 文件, 并会在纸质成绩单上附上应用规则; 运行前请详细阅读对应的 readme 文件。

b. 应用计时及屏幕输出要求。

a) 各应用需使用 time 命令记录实际运行时间, 并将屏幕输出结果一并重定向到文件中, 作为计时和结果正确性判定的标准。

b) 命令及命名规则如下:

\$ time -p (your command here) > teamname.appname.workloadNo.time.txt 2>&1

c) 参赛队提供的重定向文件命名规则必须严格按照组委会指定的格式进行, 否则每个文件扣除 0.5 分。

c. 应用代码及文档规则。

a) 如果参赛队对某应用程序源代码有修改, 需通过 U 盘一同提交修改后的代码包。

b) 竞赛结束前, 需汇总一篇操作说明文档。文档内容包括但不限于: 测试环境简述 (包括 OS 版本、MPI 软件版本、应用软件版本等)、调优操作 (包括调优内容、调优方法等)、主要运行命令等。

d. 应用算例提供和应用结果提交方式及规则如表 8-1 所示。

表 8-1 应用算例提供和应用结果提交方式及规则

应用	U 盘提供的输入文件	上传到 FTP 的文件	提交到 U 盘的输出文件
LINPACK	无	所有的 teamname.appname.workloadNo.time.txt 文件	teamname.appname.workloadNo.time.txt, 应用操作说明文档
Quantum Espresso	赝势文件, *.in 文件		teamname.appname.workloadNo.time.txt, 应用操作说明文档
SU ²	*.su2(or *.cgns), *.cfg		teamname.appname.workloadNo.time.txt, 所有结果文件, 应用操作说明文档
LICOM	INDEX.DATA, TSinitial, MODEL.FR, dncoef.h1, namelist		每个算例的最后一个 fort.22.yyyy-mm-dd 文件, teamname.appname.workloadNo.time.txt, 应用操作说明文档

③ 3D_EW 决赛规则。

a. 比赛时间。

常规赛：2014 年 4 月 23 日 8:00~18:00。

决赛：2014 年 4 月 24 日 8:00~14:00。

b. 硬件规则。

常规赛：将提供天河-2 的登录方式，并使用 128 个节点进行性能测试，每个节点包含 2 个 CPU (Intel® Xeon E5-2692 v2, 12 核, 2.20GHz) 和 3 块 MIC 卡 (Intel® Xeon Phi 31S1P, 57 核, 1.1GHz, 8GB 内存)。

决赛：将在天河-2 平台上提供 1024 个节点进行性能测试。每个节点包含 2 个 CPU (Intel® Xeon E5-2692 v2, 12 核, 2.20GHz) 和 3 块 MIC 卡 (Intel® Xeon Phi 31S1P, 57 核, 1.1GHz, 8GB 内存)。

c. 比赛流程。

应用测试 (2014 年 4 月 21~22 日)。

a) 8:00 发布常规赛的竞赛平台账户信息。

b) 向每个队伍提供 128 个节点用于测试。

常规赛 (2014 年 4 月 23 日)。

a) 8:00 发放测试用例。

b) 18:00 之前提交结果，18:00 之后竞赛平台上申请的计算节点将不能再使用。

c) 各参赛队在当天的竞赛结束前，需要将优化后的代码、makefile 文件、输出文件 (record.dat)、log 文件复制到竞赛平台自己的用户下：~/3d_ew_out/中，并且把优化后的代码、makefile 文件、log 文件下载到 U 盘的相应文件夹中，并交至工作人员（注：输出文件 (record.dat) 放到竞赛平台自己用户下即可，不需要下载）。

d) 关于计时。各队在运行各试题时，需使用 time 命令记录实际运行时间，并将其重定向到该小题目下的文件 time.txt 中，如 \$ time -p (your command here) 2> time.txt，该时间将作为最终成绩。该时间文件需连同规则 3 中的文件一同上交（复制到~/3d_ew_out/，并下载到 U 盘中）。

e) 文件命名规则如表 8-2 所示。

表 8-2 文件命名规则

算例	输出	日志文件	时间文件	job 标识
para11.in	record11.dat	log11.txt	time11.txt	
para12.in	record12.dat	log12.txt	time12.txt	
para13.in	record13.dat	log13.txt	time13.txt	

注：提交作业运行的 job 标识，大赛方将根据 job 标识统计作业的运行时间

决赛 (2014 年 4 月 24 日)。

a) 8:00 发布竞赛平台账户信息，发放测试用例。

b) 14:00 之前提交结果，14:00 之后竞赛平台上申请的计算节点将不能再使用。

c) 各参赛队在当天的竞赛结束前，需要将优化后的代码、makefile 文件、输出文件

(record.dat)、log 文件复制到竞赛平台自己的用户下：~/3d_ew_out/中，并且把优化后的代码、makefile 文件、log 文件下载到 U 盘的相应文件夹中，并交至工作人员（注：输出文件（record.dat）放到竞赛平台自己用户下即可，不需要下载）。

d) 关于计时。各队在运行各试题时，需使用 time 命令记录实际运行时间，并将其重定向到该小题目下的文件 time.txt 中，如 \$ time(your command here) 2> time.txt，该时间将作为最终成绩。该时间文件需连同规则 3 中的文件一同上交（复制到~/3d_ew_out/，并下载到 U 盘中）。

e) 文件命名如表 8-3 所示。

表 8-3 文件命名

算例	输出	日志文件	时间文件	job 标识
para21.in	record21.dat	log21.txt	time21.txt	

注：提交作业运行的 job 标识，大赛方将根据 job 标识统计作业的运行时间

④ 竞赛呈现规则。

a. 呈现时间：2012 年 4 月 25 日 08:30~12:30。

b. 呈现地点：中山大学南校区数学楼。

c. 呈现规则。

呈现顺序由 24 日下午 14 点抽签决定。

呈现以 PPT 展示，由两名参赛队员演讲，要求呈现文字和语言均使用英文。

各队呈现时间为 10min，超时酌情扣分。每队呈现后评委进行 3~5min 现场提问。评审委员根据各参赛队实际呈现情况进行打分，满分 10 分。

各队呈现过程中，该参赛队指导老师和该队其他队员可全程旁听。其他参赛队队员和指导老师均不得入场。

8.2.3 决赛规则总结

决赛要求在给定时间内完成硬件、系统、环境的配置；而应用要求在给定最大功耗条件（最大功耗 3kW）、给定时间内完成，对参赛队伍的硬件安装、系统安装、环境配置熟练度都有非常高的要求，参赛队应在赛前对这些相关操作进行训练，确保在竞赛现场万无一失；而对于应用备战，参赛队应提前熟悉相关应用，并可根据输入条件粗略判断运行时间，并根据应用的特点，有选择、有针对性地进行应用优化；而 3D_EW 是要求进行 MIC 移植的操作，这就需要参赛选手深入分析应用代码，并熟练掌握 MIC 的架构及编程方法，将应用移植到 MIC 上并发挥最大功效；在呈现环节，要注意思路清晰、条理分明、前后照应，也要注意呈现方式要直观、生动、易于理解。

第三篇 进阶篇

第9课 竞赛方案

9.1 简介

ASC 初赛阶段，各参赛队需按要求提交一份竞赛方案，通过评审委员会的评定来决定能否入围决赛阶段。本章将讨论在准备竞赛方案中需要用到的方法和可能遇到的挑战，帮助参赛队提交一份优秀的竞赛方案。需要指出的是，这份竞赛方案可以反映出参赛学生对于集群竞赛理论知识的掌握情况，如设计与架构、并行编程技能、软件应用和代码优化等方面的理论知识。同时，初赛阶段是一个既相互竞争又相互学习的过程，这也为参赛学生提供了一个非常好的接触高性能计算和超级计算的机会。

在培训阶段，参赛学生将会学习到一系列的专业技能，对他们将来科技写作、事业规划、团队合作和面对媒体等方面都有极大的帮助。在组委会提供的一些课程与咨询环节上，他们还会有机会与相应领域的专家进行交流沟通。

千里之行、始于足下，通往决赛的道路，就是从准备这份竞赛方案开始的。这份竞赛导引的主要初衷是指导那些未来想要参加竞赛的学生，为他们提供写好科技材料、提交专业方案的方法以及具有实战意义的方法与技能。接下来将详细介绍如何一步步实现这一目标。

9.2 组队

对于有兴趣参赛的学生来说，首要任务就是要组成一支具有竞争力的队伍。队伍最好能涵盖不同的专业背景，如计算机工程学、计算机科学等。因此，队员最好可以来自科学和工程等不同领域，从而可以分工应对不同方面的挑战（其实，组成一支涵盖所有学科的队伍是不可能的，原因这里就不讨论了）。

由于 ASC 中所涉及的问题对实践动手能力要求很高，因此前期的准备工作就显得尤为重要。这一点与两大传统竞赛（ISC、SC）还是不同的。不过准备竞赛方案的过程，主要还是从入门的角度来为具有超算天赋的学生提供一个进入超算世界的通道。每个队伍都会配备指导教师和指导团队，他们可以帮助组建一支多样性的队伍，并且提交具有竞争力的方案给竞赛组委会。

9.3 基 本 条 件

第一眼看上去,准备这样一个任务似乎毫无头绪,因此需要提供一些指导来帮助参赛者应对这第一个挑战。一般情况下,掌握一定的 Linux 基础知识和集群操作方法对参赛者来说是非常有益的。学生在课程学习中或者在公司实习(对于部分人来说)中可能会接触到一定的基础知识。这些知识之所以非常有益,是因为它有助于对竞赛应用环节中代码编译过程的深入了解。在这个过程中,还涉及软件层面的培训从而全面覆盖基础的核心概念。

因此,参赛队伍如果能争取到学校管理层的大力支持,并且配备一支专业权威的指导队伍,对队伍的发展非常重要。当然,学生靠自己的努力去准备这份竞赛方案也不是不可能的。这些都要根据学生的个人素质等情况来决定。

9.4 培 训

作为一个重要的环节,前期培训能够让参赛队员了解一些基础的知识以更好地准备竞赛方案。由于比赛时间紧迫,中间又有圣诞、新年等节假日,制定一个高效的培训计划就显得尤为重要。一个高效的训练计划,应该涵盖对不同领域、不同知识的培训,如计算机理论知识、并行编程技术、计算机硬件、数学、软件应用等。作为培训课程的一部分,可以适当地布置一些小作业,用来加深理解并增强兴趣。对比赛提供的题目,来自工程与科学领域的学生可能一开始的时候理解起来略有困难,所以起步阶段,学生可以先主要从他们各自的特长入手,逐步将特长与题目的背景应用结合起来。

制订训练计划的时候可以分成两个阶段:第一个阶段是在初赛题目还未公布之前的基础知识训练,即以课堂培训的形式为主的知识学习阶段;第二个阶段就必须有针对性地去培训如何写好竞赛方案,包含动手实践能力的培训(提示:可以参考往年竞赛的题目细节来找出需要了解哪些方面的知识,如可能的软硬件系统、LINPACK 测试、应用背景和代码优化等方面)。

由于大部分参赛学生都没有在超级计算环境下工作的经历,因此在前期培训的过程中,使用入门级的材料就显得尤为重要。简单易懂的介绍与词汇解释等更有助于加深理解和培养兴趣,从而为接下来的知识学习打下基础。

这种通过线下咨询进行培训的方式有助于根据不同学生的不同情况来帮助他们答疑解惑,同时培训人员也应该在此之前把一些重要的知识点先准备出来。

为了保证参赛学生能够得到全面的培训,需要所在的大学与某一科研机构或者设备供应商,或者学校不同学院之间的充分合作与配合。目前世界上已经有三大超算竞赛(SC、ISC、ASC),所以很容易找到一些可以借鉴的针对前期培训的信息。

9.5 硬件培训

竞赛方案要求合理的涵盖设计与实现的过程，因此要求参赛队员能够学以致用，将学到的知识应用到程序设计中，并在最后的提交结果中加以体现。一般来说，能够越早确定一家设备提供商并建立良好的合作关系，就越有利于完成硬件培训。维持与设备提供商的良好关系是一个很重要的起点。另一方面，在学校的实验室里也应该配备一套服务器系统，用来让队员上手熟悉搭建集群等实践操作。

根据使用目的的不同，可以有不同类型的硬件平台供使用：笔记本电脑，可以用来练习操作系统的安装和应用程序的编译；两个节点的集群，可以用来进行分布式代码（MPI）的测试与编译。在使用两节点集群时一定要确保实验结果的可复现性，以备将来能够迅速地将代码移植到 ASC 提供的平台上。队员最好能够利用这一段时间去掌握搭建双节点服务器的方法。

在不同的平台上，程序运行性能的差异是难以避免的，所以需要队员去了解集群搭建、软件堆等知识，并能够将不同系统的规格说明与配置联系起来分析。

9.6 竞赛方案

竞赛方案的要求将会随着竞赛细节的正式发布而公布。准备竞赛方案需要理论知识与实践动手能力的合理搭配。这份竞赛方案将被视为一份正式的设计报告，需清晰易懂地涵盖各种技术信息。竞赛报告应该包含不同的部分，以方便不同背景的读者阅读。

9.6.1 章节结构

竞赛方案应该包含主体部分和为主体提供补充说明的附录。对于概念、实验结果等信息，最好能用图表等更直观的方式表现出来，以帮助读者或者评委理解。

9.6.2 图表的呈现

接下来将介绍在方案中如何使用图表的形式将结果呈现出来。通过作图能够很好地反映出两个变量之间的关系（如加速比与 CPU 核数）。人们一般用 x 轴来表示自变量（如 CPU 核数），用 y 轴来表示因变量（如消耗时间）。坐标轴上应该标出尺度与单位来。图的范围应该以覆盖整个页面为宜，如图 9-1 所示。

在图表中使用合适的字体有助于更美观的信息呈现。图中的对象与字体一般用来表示某一主题以及解释一些想法与概念等。在使用线条时，应使用彼此之间易于区分的线条。例如，如果用虚线来表示以太网连接，就不要再使用相同样式的虚线来表示其他的连接了。

在图的范围内，合理地布局各个对象的分布对于信息更好地呈现来说至关重要。一般需要特别强调的对象会放到相对空白的区域以加以突出。空白区域一般都使用白色的背景

而不使用黑色或者深色的背景。黑色的背景往往会产生阴影等效果，降低了图的可阅读性，图中对象布局以及不同线性的示例如图 9-2 所示。

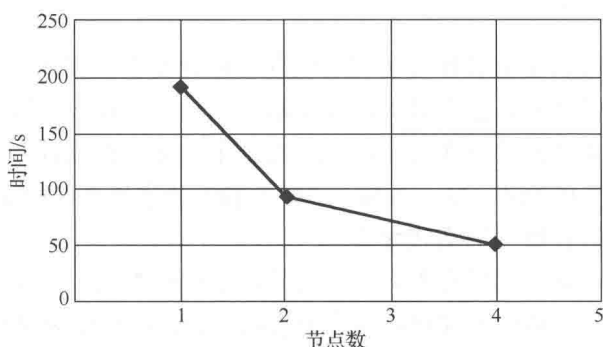


图 9-1 性能结果

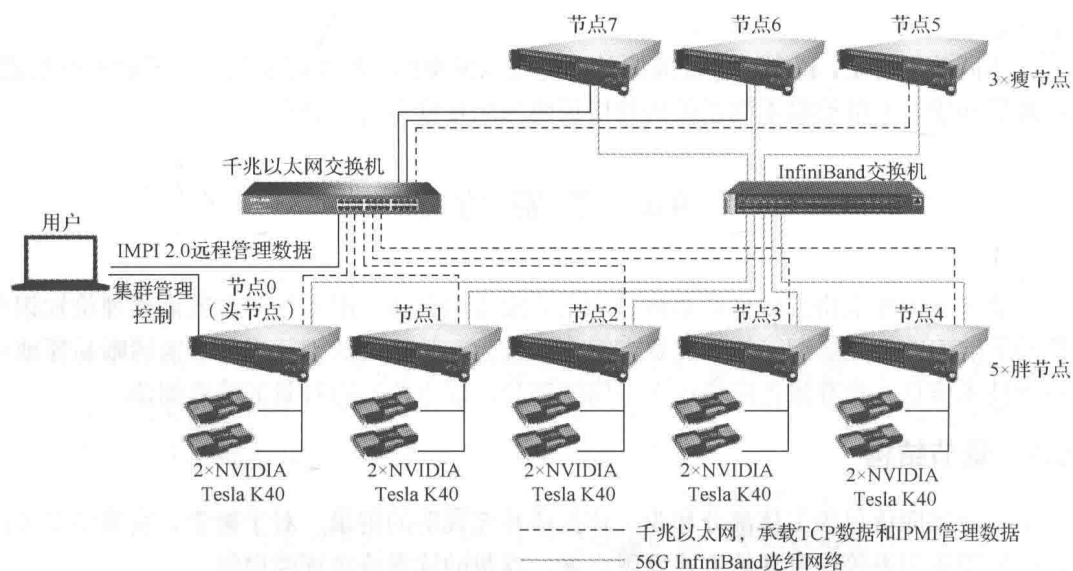


图 9-2 图中对象布局以及不同线性的示例

9.6.3 其他部分

在致谢部分，可以对一些提供过很大帮助，但在方案中没有提及的个人或者集体进行感谢等。

9.6.4 高性能计算活动简介

这一部分内容相对简单，需要选手介绍一下所在学校或者机构关于高性能计算的一些活动。可以介绍一套硬件系统和该系统的主要用途，还可以提供一些在系统上运行的规模较大应用的相关信息。

9.6.5 硬件设计与能效

由于不同的集群的硬件资源可能不相同，所以在进行硬件设计时并不需要与软件应用部分直接挂钩。选手在硬件上跑任务的时候并不需要完全按照与理论设计一样的设置来进行。选手需要明白安装编译器与数学库的重要性，最好还能在一个小规模集群上亲自试装一下。安装编译器与数学库是将来得到很好 LINPACK 测试结果的基础。

选手可以在不同的硬件集群上进行测试。同时，最好还能介绍一下在测试的过程中做了哪些工作来保证系统在 3kW 规模下运行。一般可以使用功率计来获得能耗的相应信息。表 9-1 列出了功率与效率的相关细节。

表 9-1 不同 CPU 核的结果

核数	加速器	计算峰值/Gflops	能耗/W	效率/（Gflops/W）
24	1 K40 GPU	1440	700	2.057
18	1 K40 GPU	1364	635（预估）	2.148
12	1 K40 GPU	1253	570	2.198
6	1 K40 GPU	1116	500	2.232
4	1 K40 GPU	1046	475	2.202
2	1 K40 GPU	895	465	1.924

注：HPL 配置与单核、单 GPU 测试一致

服务器一般都会配有一些冗余的特性（如备用电源）用以保证在商用环境中的可靠性。还可以考虑使用一些节省功率的设备和水冷技术等来取得更好的性能输出。水冷装置的电源功率也应该在 3kW 能耗限制以下（在进行 LINPACK 测试的时候，可能需要牺牲一定的 CPU 核吞吐量以获得更好的浮点计算能力（flops））。

本章主要为了加深对硬件能耗效率的认识。最后所有优化后的配置参数和系统正常运行的证据都应以类似表 9-2 的形式给出。

表 9-2 不同架构的能耗效率

节点数	核数	加速器	并行技术	N/NB	P×Q	峰值 计算能力 /Gflops	功率/W	效率 /（Gflops/W）
1	24	None	OpenMP within node	87000/ 224	1×1	473.3	450 ^	1.051
1	24	1 Xeon Phi	OpenMP within node, offload to Xeon Phi	87000/1024	1×1	1185.6	700 ^	1.693
1	24	1 K40 GPU	OpenMP within node, CUDA to GPU	87000/1024	1×1	1440	700 （预估）	2.057
2	48	1 Xeon Phi per node	OpenMP within node, MPI across nodes, offload to Xeon Phi	123000/1024	1×2	2216.9	1400 （预估）	1.583
2	48	1 K40 GPU per node	OpenMP within node, MPI across nodes, CUDA to GPU	123000/1024	1×2	2471	1400 （预估）	1.765

— CPU，Intel Xeon E5-2695v2: 2.4 GHz × 12 cores × 2 socket × 8 flops/clock = 460.8Gflops per node（每节点）。

MIC, Intel Xeon Phi 5110P: $1.053\text{GHz} \times 60 \text{ cores} \times 16 \text{ flops/clock} = 1010.88\text{Gflops per accelerator}$ (每卡)。

GPU, NVIDIA Tesla K40: (根据说明书) = $1430\text{Gflops per accelerator}$ (每卡)。

^功耗是在电源插口使用功率计测得的。

9.6.6 (HPL) LINPACK 测试

LINPACK 是一组用来求解稠密矩阵线性计算的测试集合,也是用来测试高性能计算系统性能的重要指标。

本章针对 HPL 的介绍、参数设置和优化方法等对于参赛队伍之后的优化基本适用。表 9-2 以表格的形式进行了一下总结。

9.6.7 应用软件

首先需要从源代码中安装软件。下载的源代码中会伴有指导安装的说明文件,最好先花一段时间预读一下这些说明文件。另外也可以在一些开源软件开发网站上找到一些有用的信息来帮助程序的编译与安装。对于软件的安装来说,应该先从串行版本入手,完成编译、安装、测试等并得到正确的运行结果,这样有助于对整个安装过程的了解。接下来再开始并行的安装。对于新手来说,一开始安装并行版本会有一些困难,不过多练习几次就会慢慢掌握了。

为了取得最好的性能结果,在编译的过程中可能要混合使用不同的编译器与 MPI 库函数。一般需要通过不断迭代才能找到最佳的组合,所以取得最佳性能的过程很麻烦,但是结果是值得的。当然也可以在一些高性能计算指导网站上找到一些使用的信息作为调优的指导。在开始优化的时候,可以直接按照网站上的指导信息进行操作,以尽快地熟悉代码编译等过程。

9.6.8 配置文档

在应用优化的层面,针对公布的题目,需要选手充分发挥他们的基本技能并与应用相结合。像 LINPACK 这样的测试文件往往用来加深对并行计算系统的理解,而真实应用可能需要更加全面的考虑。

事实上,了解与应用相适应的实际需求(如内存、CPU 与加速卡的计算能力)要比硬件本身计算能力有多强大更重要。即使拥有最先进的硬件设备,如果应用在设备上跑不起来,那性能也就无从谈起了。另外需要记录下对配置文件修改的信息并在提交的方案中加以体现,以便与原始的文件进行比较。

下面是一些建议。

1) 使用建议的参数进行安装

从有关 Quantum Espresso 的网站上可以查到,Quantum Espresso 必须按照正确的设置才能取得最佳的性能。表 9-3 中提供的就是进行测试时的初始设置。

表 9-3 初始配置信息

配置	值
使用设置	设置 OpenMP 为 enable，设置 Scalapack 为 enable
编译器	使用 C/C++ – Intel C/C++ (icc), Fortran 77/90 – Intel Fortran (ifort)和 Intel MPI (mpiicc and mpiifort)编译
库	使用 MKL BLAS, LAPACK, SCALAPACK, BLACS 和 FFT 等库进行编译

2) 找出影响性能的关键参数

下面一组数据就是可以用来提高 Quantum Espresso 性能的重要参数，括号里面列出的是测试时使用的值。

- (1) OpenMP 线程数 (1, 2, 3, 4)。
- (2) 运行的总处理器核数，即 OpenMPI 核数×OpenMP 线程数 (12, 24, 48)。
- (3) npool。
- (4) ndiag。

国家能源研究科学计算中心建议使用较少的 OpenMPI 线程。OpenMP 并行的级别与 MPI 不同。在 OpenMP 与 MPI 之间不发生冲突的情况下，将工作任务合理地分配到 MPI-OpenMPI 机制下会有效地提高性能。

运行的总处理器核数是由任务运行的 CPU 卡与核数来确定的。

3) 对问题更深入的了解

Quantum Espresso 程序本身的可扩展性很高，可分为五个层次的并行：图并行、池并行、平面波、任务组和线性代数并行。在两个负载的情况下，这里只需要用到池并行（将 K-points 分布到 CPU 的 npool 池）与线性代数并行（将矩阵对角化与矩阵乘并行），其他的三种并行由于需要的系统规模太大，在这里就不再赘述了。

4) 了解输入文件类型和应用描述

分析给定的输入文件可知，负载 1 主要用来进行结构松弛，而负载 2 主要用来进行一致性检测。

从图 9-3 中原子的位置信息可以看出，负载 2 实际上是在解决氧化锆超晶体中的 O-Vacancy 固有缺陷。由于这种缺陷往往会引发强烈的冷光效应，所以深入了解这一过程对物理学家和材料科学家设计优化一些纳米材料来说至关重要。

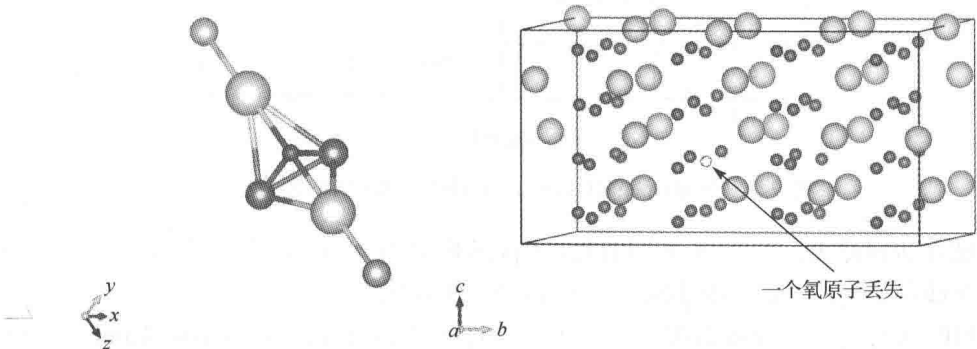


图 9-3 负载 1 与负载 2 的结构

事实上，这是一种用来处理发生在倒晶格空间上的布里渊区（Brilloin-zone）的计算。人们经常使用 K-points 网格来均衡地划分布里渊区，又称为 Monkhorst-Pack 方法。在负载 1 上，使用了 $4 \times 4 \times 4$ K-points，也就是说，在每一个晶体格上有 4 个点。在负载 2 上，只有中心 Γ 点会被考虑到。

也就是说，对性能影响较大的主要是原子的数目与 `encut`、`nspin`、`con_thr`、K-points 等参数。

5) 讨论与结果分析

我们可以通过负载凝聚来降低通信量，这一过程可以由 `npool` 决定。从以上的数据来看，从 `npool1` 到 `npool2`，运行时间降低了，这是因为 `npool2` 的情况下在两个节点之间的通信大大降低了。图 9-4 中标黑方块是可能的额外花销，它是线性递增的（随着 `npool` 的递增，分别是 10s、20s、30s、40s）。

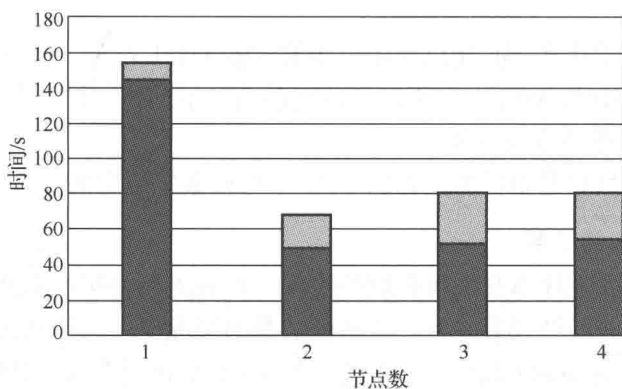


图 9-4 负载 1 上的 `npool` 比较

图 9-5 中的性能波动目前还不能用理论来解释，猜测有以下几点原因。

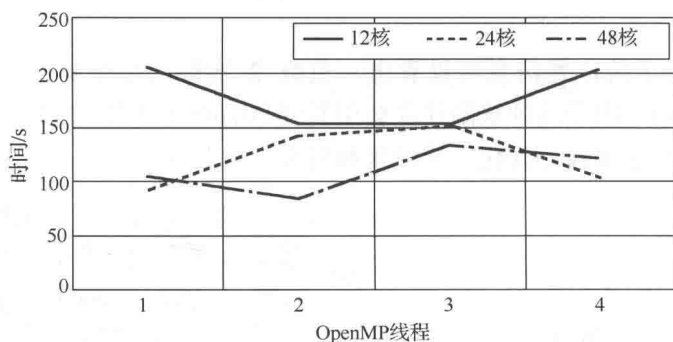


图 9-5 不同 OpenMP 线程、使用核数对时间的影响

(1) 硬件访问冲突：与 MPI 不同的是，OpenMP 线程是在同一段内存区域上工作的，所以当多个线程访问同一段地址的时候，可能会产生冲突。

(2) MPI 未优化：在服务器配置的时候，没有使用 Intel MPI Automatic Tuning Utility，所以不能保证 MPI 通信均衡。

9.6.9 代码优化

由于代码优化需要同时具备跨学科的知识,因此对分别来自科学学科和计算机学科的学生来说,都是一个极大的挑战。代码优化也是在评分过程中占比例最大的部分,所以必须引起足够的重视,事先进行重点的辅导。队员之间的紧密配合,对于代码优化的顺利完成至关重要。

代码优化可以从程序入手直接优化,也可以重写数值算法。重写数值算法一般需要花费更多的精力,所以在重写之前一定要慎重考虑,根据程序情况和比赛规则来评估是否值得重写算法还是直接优化。

首先需要分析整个代码,可以先跑一遍测试程序,产生一组正确的输出。想要取得好的可扩展性能,就需要很好地掌握性能分析方法、了解不同架构硬件性能上的区别等(如CPU与GPU+MIC之间的区别)。这样就可以从一小段代码中找出制约性能的瓶颈,然后去逐步解决更大的瓶颈,最终提高整个应用的性能。

下一步就需要记录不同阶段优化之后的性能情况。需要将不同的优化信息(如核数、编译器选项、配置参数等)整理成文,并写到方案报告中。最好还能涵盖代码优化前后的区别和性能优化前后的区别等,以方便进行比较。对优化结果分析并提出合理的解释是很重要的一个指标,可以体现出参赛队伍的技术水平。

9.7 项目书任务分配

在准备项目书的时候,最好能将不同的章节合理地分配给不同的队员,这样就能同时进行准备,大大提高工作效率。

队长需要发挥两个作用:负责完成任务的队员、负责激励队友的领袖。应该与指导老师以及培训人员保持紧密的联系,以确保进度,确保提交的内容合理并且专业。根据不同任务强度的不同,所需要花费的时间从10小时到200小时不等。

9.8 总 结

提交一份优秀的竞赛方案离不开团队合作。对于参加ASC的学生来说,比赛开始阶段确实会感觉到很困难,所以就应该先制定一个合理的训练提纲。队员的积极性高低也是决定队伍强弱、提交方案好坏的一个重要因素。另一个重要因素就是队员对基础知识、重点知识掌握的好坏。

本章介绍的主要是建议与导引,是为第一次参加竞赛的参赛者提供帮助,告诉他们如何去制订计划、组队等。将竞赛方案以合理科学的方式呈现出来是取得高分的重要因素。技术说明和技术发现需要特别标注并合理地解释,简言之,以下几个建议会对读者提供一定的帮助。

- (1) 掌握扎实的基础知识。
- (2) 团队合作。
- (3) 合理安排时间。
- (4) 自学能力。
- (5) 与指导老师、专家的建设性沟通。

竞赛方案将反映出参赛队伍对基本概念的掌握、实践动手能力、对重要发现的记录、分析，以及佐证能力等。报告最终将提交到评审委员会，由评委进行评判。

第 10 课 竞赛集群的设计与建设

起源于 20 世纪 90 年代中期的集群结构凭借低廉的价格、极强的灵活性和可扩展性 (scalability)，成为近年来发展势头最为强劲的系统结构。尽管 1997 年 6 月才有第一台集群结构的计算机进入 Top500 排名，但进入该排名的集群计算机系统的数量逐年稳步增加。2003 年 11 月，这一数字已达到 208，集群首次成为 Top500 排名中比例最高的结构，截至 2014 年 11 月，已经连续 23 期位居榜首。毫不夸张地说，集群已成为当今构建高性能计算机系统最常用的结构。

ASC 竞赛要求各参赛队搭建一个总功耗不超过 3kW 的集群，这里简要介绍集群的结构和搭建方法。

10.1 集群的结构

集群是一种价格低廉、易于构建、可扩展性极强的并行计算机系统。它由多合同构或异构的独立计算机通过高性能网络或局域网互连在一起，协同完成特定的并行计算任务。从用户的角度来看，集群就是一个单一、集中的计算资源。图 10-1 给出了一个简单 PC 集群的逻辑结构，4 台 PC 通过交换机 (switch) 连接在一起。其中 NIC 表示网络接口，PCI 表示 I/O 总线。这是一种无共享的结构，大多数集群都采用这种结构。如果将图中的交换机换为共享磁盘，则可以得到共享磁盘的集群系统结构。

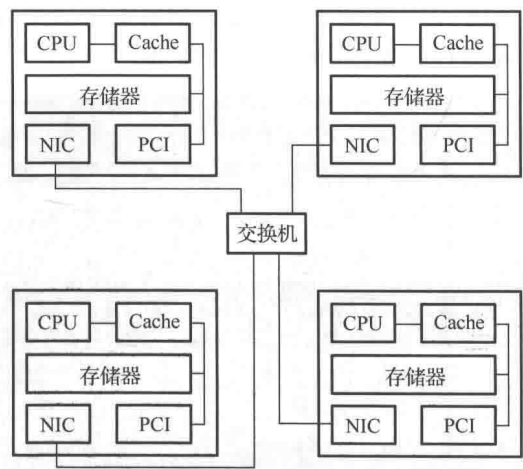


图 10-1 一个包含 4 个节点的简单 PC 集群

构成集群的每台计算机都称为节点。每个节点都是一个完整的系统，拥有本地磁盘和操作系统，可以作为一个单独的计算资源供用户使用。除了 PC 外，集群的节点还可以是工作站，甚至是规模较大的对称多处理机。

集群的各个节点一般通过商品化网络连接在一起，如以太网、Myrinet、InfiniBand、Quadrics 等，部分商用集群也采用专用网络连接，如 SP Switch、NUMAlink、Crossbar、Cray Interconnect 等。网络接口与节点的 I/O 总线以松耦合的方式相连，如图 10-1 中的 NIC 与 PCI。

InfiniBand 和千兆以太网是 ASC 参赛队所构建集群的主要互连方式。在 ASC 竞赛中，浪潮集团提供了完整的 InfiniBand 环境和千兆以太网环境。

10.2 集群的性能和功耗

根据集群的节点数量和每个节点的配置，可以估算出集群的峰值性能。若某集群共有 8 个节点，每个节点上有两块 Intel Xeon E5 2692v2 CPU（12 核）和 3 块 Intel Phi（57 核），则单个节点的峰值性能如下。

CPU 的峰值性能为 $12 \text{ core} \times 8 \text{ flops} \times 2.2 \text{ GHz} = 211.2 \text{ Gflops}$ 。

Phi 的峰值性能为 $57 \text{ core} \times 4 \text{ threads} \times 4 \text{ flops} \times 1.1 \text{ GHz} = 1.003 \text{ Tflops}$ 。

单节点的峰值性能为 $0.2112 \times 2 + 1.003 \times 3 = 3.4314 \text{ Tflops}$ 。

集群的峰值性能为 $3.4314 \times 8 = 27.4512 \text{ Tflops}$ 。

集群系统功耗的估算方法与此类似，也是把各个部件的功耗加在一起，所得的值就是系统的总功耗（估计值）。在总功耗约束一定的情况下，可以据此确定集群中的节点数量。

有的时候，参赛队会选用一些低功耗的部件降低系统功耗，如用固态硬盘（SSD）代替机械硬盘。虽然这样做降低的功耗不足以向系统中增加一个新的节点，但在比赛中却会得到不小的好处。因为在 ASC 竞赛中，组委会一般会使用电流计对各参赛队所搭建集群的实时功耗进行监控。

图 10-2 是 2014 年 4 月的 ASC14 竞赛上，部分参赛系统的实时功耗曲线。显然，SSD 所降低的功耗可以为系统在运行时提供更大的功耗空间。

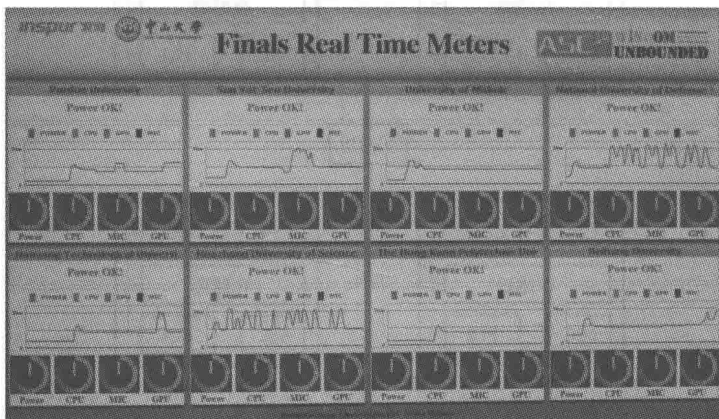


图 10-2 ASC14 部分参赛队的实时功耗监测曲线

比赛时，为了提高效率，将更多的准备时间用于调试和测试，可以将集群的安装和配置步骤做成脚本，也可以事先将系统安装在 SSD 上。

第 11 课 系统浮点性能测试 HPL 优化方案

HPL 测试是评估现代并行计算机系统性能的主流方法,它主要反映出系统的浮点计算能力。每年发布两次的全球超级计算机系统 Top500 排名的依据就是各系统完成 HPL 的性能。

HPL 基准程序采用高斯消元法求解一元 N 次稠密线性代数方程组。当矩阵规模为 N 时,浮点运算次数为

$$2/3 \times N^3 - 2 \times N^2 \quad (11-1)$$

因此只要给出问题规模 N 并测得系统计算时间 T ,就可以根据下式估算出系统性能(单位:flops,每秒完成的浮点运算次数)

$$\text{性能} = \text{计算量} / \text{计算时间} = (2/3 \times N^3 - 2 \times N^2) / T \quad (11-2)$$

受到 3kW 的系统总功耗^①限制,ASC 竞赛中 HPL 成绩一般为数 Tflops,2014 年已经超过了 10Tflops。

测试时,对矩阵的大小 N 没有限制,即求解规模是可变的。为了获取最佳性能,系统可以采用任何不改变基本算法(即式(11-1)反映的计算量)的优化策略,如改变 N (调节问题规模)、修改使用 CPU 的数目、采用高性能库等。

可从<http://www.netlib.org/benchmark/hpl>下载 HPL 源代码。

11.1 HPL 简介

HPL 通过求解一个稠密线性方程组 $Ax = b$ 来测试计算机系统的浮点计算能力,其中 A 和 b 已知, x 为待求向量。

HPL 采用高斯消元法求解,其基本思想为:首先对矩阵 A 进行 LU 分解($A = LU$),其中 L 为单位下三角矩阵, U 为非奇异上三角矩阵。令 $Ux = y$, $Ly = b$, 求解 $Ux = y$ 。

具体实现为:矩阵 A 的分解表示为

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} = \begin{pmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{pmatrix} \cdot \begin{pmatrix} U_{11} & U_{12} \\ 0 & U_{22} \end{pmatrix} = LU \quad (11-3)$$

式中, A_{ij} 和 L_{ij} 已知, U_{ij} 待求($i, j = 1, 2$)。

由式(11-3)可得

$$A_{11} = L_{11}U_{11} \quad (11-4)$$

式(11-4)中 L_{11} 和 U_{11} 由列主元分解得到,这样 L_{21} 、 U_{12} 也可通过下式求得

$$L_{21} = A_{21}U_{11}^{-1} \quad (11-5)$$

^① 有时候总功耗是 $2 \times 13A \times 110V = 2860W$ 。

$$U_{12} = A_{12}L_{11}^{-1} \quad (11-6)$$

此外, 由式 (11-3) 还可得

$$A_{22} - L_{21}U_{12} = L_{22}U_{22} \quad (11-7)$$

从上述算法 (式 (11-4) ~ (11-7)) 中可以看出, HPL 中的主要运算集中在矩阵运算, 包含矩阵乘加和矩阵逆乘, 因此其优化策略主要是针对提高矩阵运算进行的。

11.2 安装与测试

本节以 Ubuntu 14.04 为例, 介绍 HPL 的安装与测试方法。所用的计算机配置为 2 个 Intel Core i7 CPU (双核), 内存 8GB, 编译器为 GFortran。HPL 以多进程方式执行, 因此还需安装 MPICH2, 构建 MPI 运行环境。其他平台和系统上 HPL 的安装及测试方法与此类似。

1) 编译 ATLAS

HPL 默认使用 ATLAS 库, 从 <http://sourceforge.net/projects/math-atlas/files> 下载 ATLAS 并解压缩, 可得到一个名为 ATLAS 的目录。之后, 按照下面的步骤编译 ATLAS。

```
cd ATLAS
mkdir build
cd build
../configure
Make
```

若编译正确结束, 则会在 build/lib 目录下看到 libatlas.a、libcblas.a 等库, 编译 HPL 时会用到这些库。

2) 编译 HPL

解压缩下载得到的 HPL 包 (以 2.1 版为例), 可以得到一个名为 hpl-2.1 的目录。

要编译 HPL, 首先要创建一个 Makefile 文件。对于该文件, HPL 的命名规则为 Make.arch, 这里 arch 表示体系结构名。本节假设 arch=Ubuntu, 因此文件名应为 Make.Ubuntu。

这一工作可以通过修改 setup 子目录下的 Make.* 文件完成。因为本节所用的平台使用的是 Intel 处理器, 所以可以在 setup/Make.Linux_PII.CBLAS_gm 文件的基础上完成, 将该文件重命名为 Make.Ubuntu 即可。需要完成的修改如下。

```
ARCH = Ubuntu
TOPdir = ... # TOPdir 表示 hpl-2.1 所在的目录

LAdir = ... # LAdir 表示 libatlas.lib 的库文件所在的目录
LAinc = ...
LAlib = $(LAdir)/libcblas.a $(LAdir)/libatlas.a
```

得到 Make.Ubuntu 文件之后, 就可以编译 HPL 了, 命令如下。

```
make arch=Ubuntu
```

如果编译正确结束，在 bin/Ubuntu 目录下将会生成两个文件，即 HPL.dat 和 xhpl，其中 HPL.dat 是配置文件，记录了一些配置参数，xhpl 则是要执行的测试程序。

3) 修改配置

下面是一个 HPL.dat 文件的内容样例，其中定义的主要参数如图 11-1 所示。

行数	HPLinpack benchmark input file	
1	HPL.out	output file name (if any)
2	6	device out (6=stdout,7=stderr,file)
3	1	# of problems sizes (N)
4	8192	Ns
5	1	# of NBs
6	192	NBs
7	0	PMAP process mapping (0=Row-,1=Column-major)
8	1	# of process grids (P x Q)
9	2 1 4	Ps
10	2 4 1	Qs
11	16.0	threshold
12	1	# of panel fact
13	0 1 2	PFACTs (0=left, 1=Crout, 2=Right)
14	1	# of recursive stopping criterium
15	2 4	NBMINS (>= 1)
16	1	# of panels in recursion
17	2	NDIVs
18	1	# of recursive panel fact.
19	0 1 2	RFACTs (0=left, 1=Crout, 2=Right)
20	1	# of broadcast
21	0	BCASTs (0=1rg,1=1rM,2=2rg,3=2rM,4=Lng,5=LnM)
22	1	# of lookahead depth
23	0	DEPTHS (>=0)
24	2	SWAP (0=bin-exch,1=long,2=mix)
25	64	swapping threshold
26	0	Ll in (0=transposed,1=no-transposed) form
27	0	U in (0=transposed,1=no-transposed) form
28	1	Equilibration (0=no,1=yes)
29	8	memory alignment in double (> 0)

图 11-1 HPL.dat 文件实例

配置文件的每一行左边的数值为参数值，右边的文字为参数名或参数的含义，其中很多参数是成组出现的，如第 3、4 行。第 3 行 # of problems sizes (N)，指出测试时应使用几个不同的矩阵规模，图中值为 1 表明只有一个矩阵规模。第 4 行则指明了具体的规模数值，即 N=8192。测试时，对于每个矩阵规模 N 将进行一次测试，获得一个测试结果。

例如，若 HPL.dat 文件的第 3、4 行为下面的内容，将进行两次测试，测试的矩阵规模分别为 8192 和 4096。不同的参数值之间用空格隔开。

```
2          # of problems sizes (N)
8192 4096  Ns
```

类似的参数组还有，第 5、6 行指明了分块大小，第 8、9、10 行指明了进程阵列等。通过指定每个参数的数量，一个 HPL.dat 文件可以指定多次测试的参数，测试次数等于各个参数数量的乘积。例如，一个 HPL.dat 中对矩阵规模和块大小的定义分别如下。

```
2          # of problems sizes (N)
8192 4096  Ns
3          # of NBs
192 96 256  NBs
```

则它指定了 6 次测试。

4) 测试 HPL

执行下面的命令，即可运行 HPL。

```
mpiexec -n 4 ./xhpl
```

执行时部分屏幕输出如图 11-2 所示，省略的内容为 HPL.dat 文件中定义的各个参数及其数值。

```
... ..
-----
- The matrix A is randomly generated for each test.
- The following scaled residual check will be computed:
      ||Ax-b||_oo / (eps * ( || x ||_oo * || A ||_oo + || b ||_oo) * N)
- The relative machine precision (eps) is taken to be 1.110223e-16
- Computational tests pass if scaled residuals are less than 16.0
=====
T/V          N    NB    P    Q          Time          Gflops
-----
WR00L2L2      8192      192    2    2          30.63          1.197e+01
HPL_pdgesv() start time Sat Dec 13 01:55:16 2014
HPL_pdgesv() end time   Sat Dec 13 01:55:47 2014
-----
||Ax-b||_oo / (eps * (||A||_oo * ||x||_oo + ||b||_oo) * N) = 0.0023562 ..... PASSED
=====
Finished      1 tests with the following results:
              1 tests completed and passed residual checks,
              0 tests completed and failed residual checks,
              0 tests skipped because of illegal input values.
-----
End of Tests.
=====
```

图 11-2 一次 HPL 测试的部分输出（HPL.dat 如图 11-1 所示）

可以看出，测试得到的性能为 1.197×10^1 Gflops，即 11.97Gflops。

11.3 常用的 HPL 优化方法

11.3.1 修改配置参数

HPL 的配置文件 `hpl.dat` 中所声明的各个参数的值对于性能测试结果有较为明显的影响,影响较大的有问题规模 N 、分块大小 NB 、进程阵列 $P \times Q$,而且根据被测系统的配置参数可以比较容易地确定这些参数的最优值。其他一些参数,如处理器阵列排列方式、矩阵分解方式、横向广播方式等,对于测试结果的影响相对较小,且其取值不容易确定,往往需要进行多次测试才能确定。

1. 矩阵规模

一般来说,矩阵的规模 N 越大,有效计算所占的比例也越大,系统浮点处理性能也就越高。但与此同时,矩阵规模 N 的增加也会导致内存消耗量的增加,一旦系统内存空间不足,性能会大幅度降低。因此,可以参考系统内存的容量确定矩阵规模,在尽量增大矩阵规模 N 的同时,又保证不超过内存容量。

考虑到操作系统本身需要占用一定的内存,除了矩阵 A (大小为 $N \times N$) 之外,HPL 还有其他的内存开销,另外通信也需要占用一些缓存(具体占用的大小视不同的 MPI 库而定)。根据相关资料来看,矩阵 A 占用系统内存容量的 80% 左右为最佳。

当然,这只是一个参考, N 的具体选择还与实际的软硬件环境密切相关。对于 ASC 竞赛来说,3kW 的功耗将系统限制在一个较小的规模,节点数量多在 8~14,所以 N 的取值并不像大规模系统那样严格。

2. 分块大小

为了提高数据的局部性,从而提高整体性能,HPL 将矩阵分块进行求解,因此块大小 NB 对性能有很大的影响。 NB 的选择和许多软硬件因素有关。虽然 NB 值的选择主要是通过实际测试得到的,但还是有一些规律可寻,如 NB 不可能太大或太小,一般在 512 以下; NB 一般取 64 的整倍数等。

3. 二维进程阵列

二维进程阵列 ($P \times Q$) 的选择主要遵循以下几个要求。

(1) $P \times Q$ = 进程数,即命令 “`mpiexec -n 4 ./xhpl`” 中通过命令行参数 $-n$ 指定的进程数。

(2) 对于使用 GPU 且仅用 GPU 完成 HPL 的异构系统, $P \times Q$ = 系统中 GPU 数量 = 进程数,即一块 GPU 卡完成一个进程。

(3) $P \leq Q$, 这是一个测试经验值。一般来说, P 的值尽量取得小一点,因为列向通信量的通信次数和通信数据量要远大于横向通信。

需要强调的是,以上仅是选择参数 N 、 NB 、 P 和 Q 取值的一些参考经验。对于不同的被测系统,最终的取值也不相同,甚至还会有比较大的差别。

11.3.2 第三方库的选择

这里的第三方库指的是 MPI 库和 BLAS 库,常用的 MPI 库有 MPICH2、IMPI、OpenMPI 等, BLAS 库包括 ATLAS、GOTO、MKL 等。HPL 中 CPU 上进行的绝大部分计算都是交给 BLAS 库完成的,所以 BLAS 库的选择也是相当重要的。若要在 CPU+GPU 异构平台上测试 HPL,一般需要使用 GPU 厂商提供的 BLAS 库。

在先前的安装过程中,我们使用了 MPICH2 环境和 ATLAS 库。可以根据需要选择使用不同的 MPI 实现或 BLAS 库。SCC (Student cluster competition) 的大多数参赛队都选择使用 Intel Xeon 系列 CPU,因此也往往会选用 Intel 提供的配套 MPI 库和 BLAS 库。从实际情况看,所获得的性能也比较理想。

11.3.3 编译器优化

借助编译器自身的优化能力,可在不修改程序的条件下获得性能提升。这也是提升程序性能最简单的一种方法。采用编译优化的手段提高程序性能,可以选用优化能力强的编译器,或选择适当的优化选项。

目前 SCC 竞赛的参赛队大都使用 Intel 生产的 Xeon 系列多核 CPU,因此使用 Intel 开发的 C/C++编译器 icc/icpc 或 Fortran 编译器 ifort,有助于获得高质量的目标代码。可以在 Intel® 下载中心免费下载其试用版 (非商业版本)。表 11-1 列出了部分 icc 支持的编译优化选项。

表 11-1 部分 icc 支持的编译优化选项

编译选项	功能描述
-O2	最大化目标代码执行速度。包括全局指令调度、软件流水、预测与数据预取等
-O3	在-O2 的基础上加上更加激进的循环和内存访问优化,包括标量替换、循环展开、代码复制以及循环分块等
-ax_	生成代码指定与用于指定的处理器,也同时生成通用的 IA-32 指令
-funroll-loops	循环展开
-static	静态链接库
-fomit-frame-pointer	开启 EBP 作为通用寄存器

需要指出的是,编译器的各种优化选项对 HPL 性能的影响很小,因为在 HPL 中绝大部分的运算 (矩阵乘) 都是交给 BLAS 库来完成的,而库的性能一般无法通过更改编译选项得到提升。对于其他应用程序,改变编译选项的效果会更加明显一些。

11.3.4 确定进程与节点的对应关系

11.2 节介绍的测试命令用于单个节点。要在多个节点的集群中测试 HPL,还需要指明参与计算的各个节点,如使用下面的命令。

```
mpirun -np 3 -hosts node01 node02 node03
```

MPI 默认按照命令行中节点的输入顺序来分配进程。如果使用上面的命令,则进程 0、

1、2 将分别被分配到节点 node01、node02、node03 上。但在测试中会发现，改变节点的输入顺序也有可能对集群性能和其运行稳定性带来极大影响。

就经验而言，要注意以下几个方面。

(1) 注意不要将共享文件夹主节点作为 0 号进程节点，因为其本身已经承担共享文件夹的管理。

(2) 当性能未达到预期时，可以尝试交换两个性能皆不佳的节点的输入顺序，也许会有所改善。

(3) HPL 中每个进程的任务事实上并不是完全平均的，因此可以通过进程任务量的大小来安排进程与节点的对应关系。

11.3.5 优化任务分配

在只有 CPU 的同构集群上测试 HPL 的任务分配优化工作较简单，而在 CPU+GPU 的异构集群上，这一工作就比较复杂。在异构平台上，CPU、GPU 混合计算，协同实现 DGEMM，即将矩阵乘法中的一部分放到 GPU 上进行，调用 NVIDIA cuBLAS 库中的 DGEMM 函数；同时，另一部分调用 CPU 上 BLAS 库中的 DGEMM 函数。因此，需要考虑 CPU 和 GPU 之间的任务划分。

1) DGEMM 的 CPU、GPU 并行化实现

图 11-3 介绍了矩阵 $A(M \times N)$ 与矩阵 $B(N \times K)$ 相乘得到矩阵 C 的两种任务分割方法，分成了图(a)（竖切 B ）和图(b)（横切 A ）两种情况。

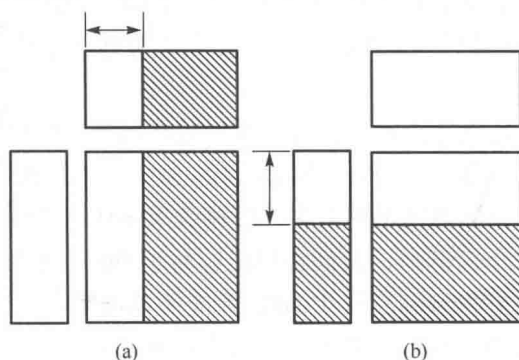


图 11-3 矩阵列划分和矩阵行划分方法

之所以需要进行这两种划分，是因为在 DGEMM 的计算中，矩阵乘法的参数 M 、 N 、 K 存在两种情况，一个是 K 较小， M 与 N 较大，如图 11-3(a) 的情况，此时划分矩阵 B 会得到较好的性能；另一种是 K 与 N 相等并较小，而 M 相对较大，如图 11-3(b)，此时划分矩阵 A 会得到较好的性能。

通过调节比例因子 R ，可以达到 CPU 与 GPU 间负载的均衡。最佳的情况是，CPU 上的计算时间=数据从 CPU 到 GPU 的传输时间+GPU 计算时间+结果从 GPU 传回 CPU 的传输时间。最佳的比例因子 R 可以通过多次的实验获得。

从实现细节上，有两点需要注意的地方：①cuBLAS 的 DGEMM 函数，使用了 Volkov

等的算法,对于参数 M 、 N 、 K 的不同取值性能变化明显,一般来说,在 M 为 64 的整数倍、 N 和 K 为 16 的整数倍时,性能最佳,因此在划分的时候应尽量使 GPU 上的矩阵满足此规则;②在较小的数据规模下, GPU 并不能发挥作用,所以当规模小于一定阈值的情况下,可以直接调用 CPU 的 DGEMM 函数完成计算。

2) CPU、GPU 负载均衡

在 1) 中, CPU 与 GPU 完成 DGEMM 和 DTRSM 计算时采取了静态的任务划分策略,比例因子 R 是固定的。但实际测试时, GPU 与 CPU 的计算性能并不是恒定不变的,而是在一定规模内随矩阵规模的减小而降低的,因为 HPL 的计算过程是一个迭代的过程,计算规模随着迭代的进行、迭代次数的增加而减小,因此静态负载均衡不可能实现完全的负载均衡。使用动态划分策略,即根据本次 CPU 与 GPU 的实际运行性能,更新下一次的 R 值,也许能够解决这一问题。

但这需要对 HPL 的代码进行一定的修改,才能实现。

11.3.6 散热

除了上面介绍的一些优化方法,还有一些情况需要注意。例如, GPU 的工作状态对于温度极其敏感,因此要特别关注系统的散热能力和散热情况。我们在实际测试中发现,一旦 GPU 工作温度过高,其性能将会大幅度下降,甚至造成程序的运行结果错误。

对于这个问题,一是可以通过调整风扇转速来加强散热,风扇转速可在 BIOS 中设置;二是可根据实际情况对系统的风道进行大胆改进。例如,我们就曾借助硬纸板等简易材料改变系统风道,在不影响其他器件工作的情况下将风扇吹出的风尽可能地引向 GPU,不仅使每块 GPU 的性能提高了接近 40Gflops,而且还大大降低了系统因为 GPU 过热而崩溃的概率。

竞赛时,由于天气、场地等因素,有可能遇到散热问题,使得调机时的 HPL 性能远远低于竞赛准备时的性能。例如,在 2013 年 11 月的 SC13 竞赛中,由于两个参赛队的机器是背靠背摆放的,风扇吹出的热风不易散去,散热效果不佳,导致 HPL 测试性能很低而且结果极不稳定。在发现这个问题后,我们将机柜旋转了 90°, HPL 性能很快稳定了下来,达到了准备时的水平。我们也顺利获得了 HPL 性能的单项第一。

不过,要从根本上解决系统散热的问题、提高散热效率,还应依赖赞助厂商的帮助,由他们提供风道设计更加合理、风扇转速调节更加智能的平台。

第 12 课 分子动力学软件 GROMACS 优化方案

本章将先简短地介绍 GROMACS 这套科学计算应用程序,然后利用一组 4 个计算节点的集群系统,进行 GROMACS 的效能测试与分析,并根据实验结果,进一步讨论 GROMACS 这套科学计算应用程序的效能优化方案。

12.1 应用程序简介

GROMACS (groningen machine for chemical simulation) 是一个根据经典分子动力学理论开发的仿真应用程序,它可以模拟在各种场和边界条件情况下,固态、液态或气态的粒子集合体,并随着时间的变化来观察分子的反应机制和特征,仿真的规模可以从几百到几百万个分子。GROMACS 的主要应用是研究生化分子,尤其像蛋白质 (proteins)、脂类化合物 (lipids)、核酸分子 (nucleic acids) 等具有复杂且高连接性的分子结构。

GROMACS 的实现是基于 MPI 的架构将 GROMOS 这套模拟经典分子动力学的软件并行化至多台计算机的执行环境。GROMACS 可模拟三种力: 分子间的键结力 (bounded forces)、非键结力 (unbounded forces) 和外力 (external forces)。分子动力的模拟主要是应用 leap-frog 和 Verlet 算法,计算牛顿耦合方程式来完成,并使用其他演算方法处理各种边界条件。GROMACS 使用区域分解法,将计算工作分配到不同的计算节点。但在真实仿真计算中,往往会由于区域间不同的交往密度或粒子在空间分布不均等因素,造成计算节点负载不平均的问题,而影响计算效能。

为了实现高效能计算, GROMACS 对于程序代码和算法都进行了许多优化,如自行开发函数库计算开根号的倒数,以及使用内在函数 (intrinsic function), 让编译程序 (compiler) 可以充分利用指令集 (instruction-level) 的执行并行度等技巧。从 4.6 版之后更开始提供以 CUDA 程序语言开发的计算模块,实现使用 GPU 来加速计算。详细相关的软件开发信息和功能可以参考相关文献。

GROMACS 是遵守 GNU 许可的免费软件,官方的软件版本和使用说明都可以从以下站点下载: <http://www.gromacs.org>。在以下效能测试实验中,我们使用 GROMACS 4.6.1, 进行参数调校的测试则是使用 GROMACS 官方提供的 “DPPC in water”。

12.2 实验环境与硬件规格

GROMACS 的效能测试与调优是使用一组 4 个计算节点的集群系统进行的。每个计算节点都拥有两个 10 核心的 Intel CPU, 以及一个 NVIDIA Tesla K20X 加速卡, 计算节点由 InfiniBand 所串接, 提供高效能的网络链接。详细的服务器硬件规格如表 12-1 所示。

表 12-1 详细的服务器硬件规格

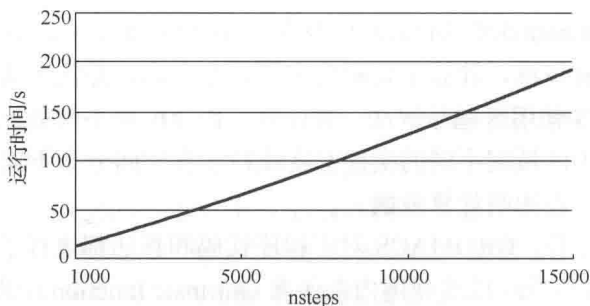
设备	配件	配置	数目
Server	CPU	Intel® Xeon® CPU E5-2670 v2 @ 2.50GHz, 10 core	2
	Memory	16GB, DDR3, 1600MHz	10
	Hard disk	300GB, SAS, 15000rpm	2
Accelerator card	GPU	NVIDIA Tesla K20X (2688 cores, 6GB GDDR5 Memory)	2
HCA card		Infiniband Mellanox Technologies MT27500 Family [ConnectX-3]	1

12.3 测试参数调校

首先对测试的参数进行效能影响的分析。GROMACS 的输入参数中对运行时间有影响的并不多，其中影响最显著的参数之一就是 `nsteps`。`nsteps` 控制模拟时的计算步数，因为 GROMACS 每一次计算步数所需的时间是固定的，所以程序的运行时间与 `nsteps` 有固定正比的关系。测量结果如图 12-1 所示，当 `nsteps`=1000 时，整个模拟的运行时间是 14s，当 `nsteps`=5000 时，运行时间也相对成长至 64s。因此，GROMACS 运行时间可以用下面的等式估算：

$$\text{运行时间} = \alpha \times \text{nsteps} + \beta$$

以图 12-1 中的测试算例为例， $\alpha = 0.0127$ ， $\beta = 1.32$ 。

图 12-1 `nsteps` 参数对运行时间的影响

另一个对程序运行时间有影响的是控制产生输出结果的参数。GROMACS 可以通过这些参数的设定值控制各种信息的写入至输出文件的间隔步数，各参数的名称与其对应的输出数据如下。

- (1) `nstxout`: 设定写入当下坐标至输出文件的间隔步数。
- (2) `nstvout`: 设定写入当下速度至输出文件的间隔步数。
- (3) `nstxtcout`: 设定写入当下位置至输出文件的间隔步数。
- (4) `nstlog`: 设定写入当下能量至输出文件的间隔步数。

图 12-2 显示各输出参数在不同设定值时所需的间隔步数，间隔步数=100 表示每 100 个模拟步数后输出一次结果，所以间隔步数越小表示输出越频繁。从结果中可观察到随着输出写入输出文件的间隔步数增加，程序的运行时间也会相对略为减少，这是可预期的结果，因为输出数据至文件需要额外的时间，尤其大量频繁的硬盘写入往往成为程序的效能

瓶颈。所以若不需记录输出数据，可以选择将这些参数的值设为 0 来避免任何输出动作，并缩短程序运行时间。

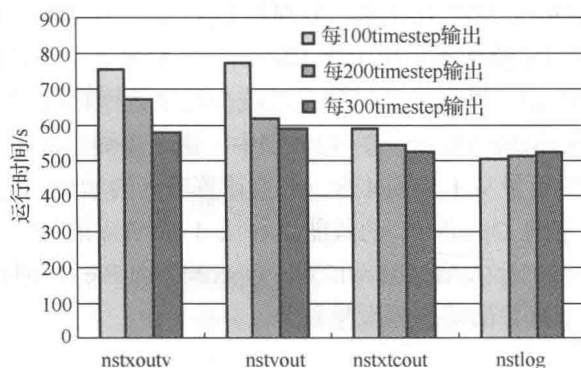


图 12-2 输出参数对运行时间的影响

12.4 软件库的效能分析

GROMACS 执行时需要用到许多第三方的软件库和软件，包括编译程序的选择、MPI 的版本以及负责计算 FFT 的软件库等。以下将逐一针对不同的选择进行比较，观察对程序效能的影响。根据前面的测试结果，以下实验中也将固定 `nsteps` 设为 xxx，并且关闭所有输出文件的动作。

1) 编译程序 (compiler)

编译程序的选择对于程序执行的效率往往也有一定程度的影响。由于我们的测试机器使用 Intel 的 CPU，因此除了选择一般所使用的 GNU 编译器，也可以选择使用 Intel 为它们的专属 CPU 所开发的 Intel 编译器。图 12-3 比较使用不同编译程序所产生的执行文件的运行时间。如图 12-3 所示，不论使用 8 个核心还是 16 个核心来执行程序，使用 Intel 编译器都可以明显减少约两成的运行时间。使用 8 个核心时，GNU 编译器的执行文件需要 6710s，Intel 编译器的执行文件却只要 5409s，相当于近 20% 的效能改善。使用 16 个核心时，虽然两个版本的执行文件都有因为使用更多计算资源而缩短运行时间，但与 GNU 编译器相较之下，Intel 编译器仍然可将运行时间从 3827s 减至 3102s，约 20% 的效能改善。所以在 Intel 的 CPU 机器上使用 Intel 编译器的确对 GROMACS 的执行效能也有相当大的帮助。

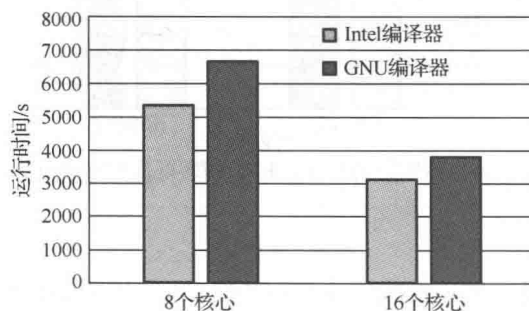


图 12-3 不同核心数下，使用不同编译器版本的效能比较

2) MPI 工具

GROMACS 使用 MPI 达成跨计算机的并行处理。目前可执行于 InfiniBand 上, 并广泛使用于 Top500 超级计算机的 MPI 软件库, 分别是 OpenMPI 和 MVAPICH2。因此将使用不同 MPI 软件库的运行时间实验结果呈现于图 12-4(a)中, 结果显示不论使用 8 个核心还是 16 个核心来执行程序, 使用 MVAPICH2 都可以稍微减少程序运行时间, 但差距并不大, 约 200s (MVAPICH2=3102, OpenMPI=3321)。图 12-4(b)中, 进一步固定在每个计算节点使用 8 个核心, 但增加计算节点的数量从 1 台到 4 台。随着计算节点的增加, 程序花在数据传输的时间也相对增加, 因此观察到 OpenMPI 的效能改善从 1 个节点的 6% (MVAPICH2=3102, OpenMPI=3321) 增长到 20% (VAPICH2=1274, OpenMPI=1596), 因此使用 MVAPICH2 对网络效能的改善会随着节点数的增加越来越显著。

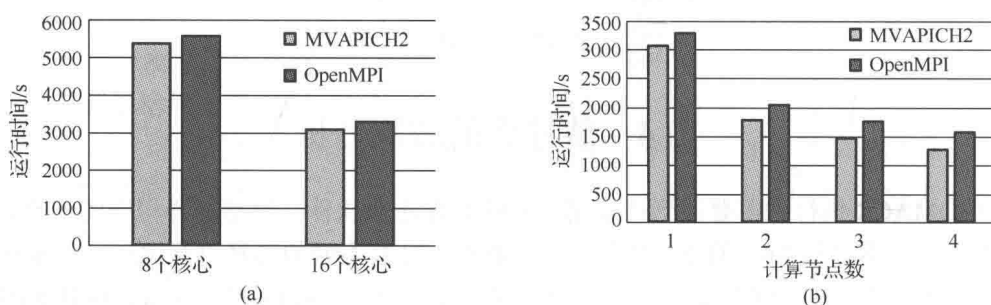


图 12-4 不同核心数节点数下, 使用不同 MPI 版本的效能比较

3) FFT 库

在 GROMACS 的运算过程中会需要用到大量的 FFT 计算, 而目前提供 FFT 的软件库有 FFTW 和 MKL 两种。我们同样地也测试了这两者的效能差异, 结果如图 12-5 所示, 不论使用 8 个核心还是 16 个核心来执行程序, MKL 都只稍微比 FFTW 慢一点点。因此我们并没观察到 FFT 的软件库对 GROMACS 的效能有影响。

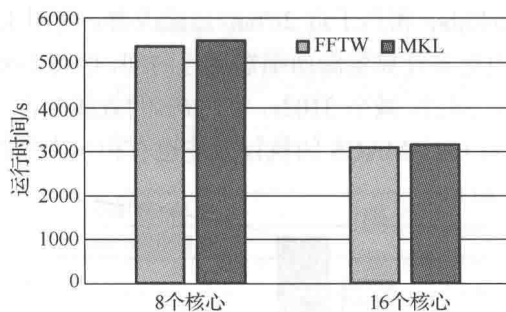


图 12-5 不同核心数下, FFT 的软件库的效能比较

4) 总结

根据上述的实验观察结果, 将能优化 GROMACS 效能的最佳配置整理于表 12-2 中。

表 12-2 优化 GROMACS 效能的最佳配置

	编译器	MPI 库	FFT 库
最佳设定	Intel 编译器	MVAPICH2	FFTW
效能改善	20%	随节点数增加 1 个节点 6% 4 个节点 20%	<1%

12.5 计算资源分配的效能分析

本节使用 12.3 节和 12.4 节所找到的最佳参数与软件库设定，并利用改变计算资源分配来观察效能的影响。针对以下三项计算资源的配置进行效能分析：①使用核心数量；②核心与节点的配置比例；③使用计算节点数量。

1) 核心数量

为了测试计算核心数量对 GROMACS 效能的影响，首先针对单台机器进行测试。结果如图 12-6(a)所示，随着使用的计算核心从 2 增加到 16，程序运行时间也随之减少。图 12-6(b)进一步比较加速比 (speedup factor)，也就是与使用两个核心的时间相比的比例，从此图更可以观察到时间减少的比例与核心数量有线性的正比关系，因此增加使用的计算核心量不仅可以减少运行时间，更可以根据使用少量核心数的结果来准确地预测使用更多核心数时所需的运行时间。我们所观察到的平行效率 (parallel efficiency) 约有 75%(=0.36/0.5)，虽然未达到理想的线性可扩展性 (linear scalability)，但也算是相当高的平行效率了，可见 GROMACS 在单一计算机有理想的平行度。

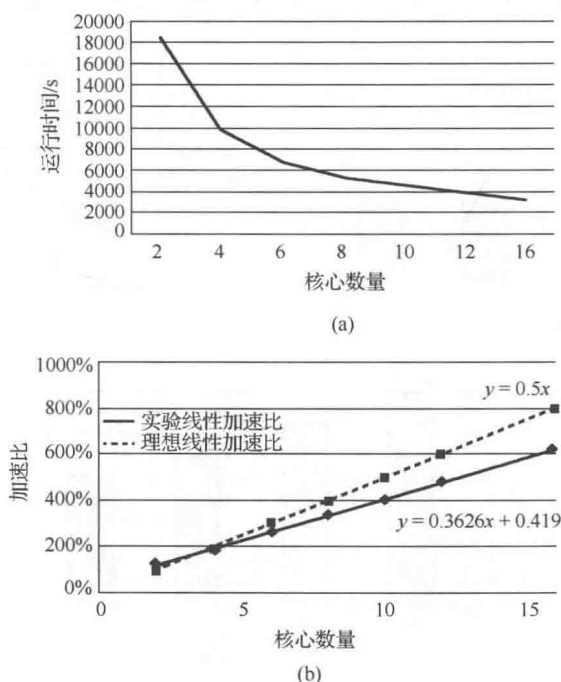


图 12-6 使用单台机器下，增加使用核心数的运行时间和效能加速比

2) 核心与节点的配置比例

接下来我们测试在固定的核心数情况下，当核心被分散在多台机器时会造成的效能影响，这里固定使用核心数为 16，再分别比较配置到 1 台、2 台、4 台机器中的结果。如图 12-7 所示，可以明显地看出若将使用核心分散在 4 台机器，运行时间会明显变长。这主要是因为计算机间所需的网络沟通导致的结果。尤其可以看到使用单一节点，避免所有网络沟通时运行时间会大幅减少。相对使用越多节点，网络的沟通时间会缓慢但持续地增长。所以与单一计算机的平行度相比，跨计算机的平行度会因网络沟通的效能影响而下降。因此执行 GROMACS 时，为了达成最佳效能，应使用单一机器所能支持的最大核心数，减少网络沟通，并充分利用单一计算机的计算资源。

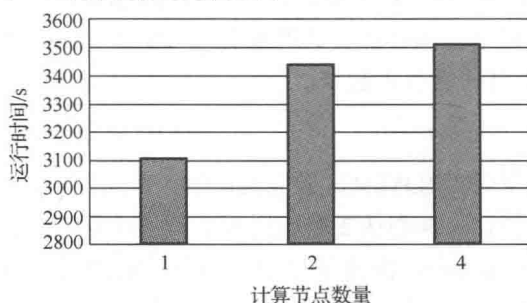


图 12-7 固定使用核心数，将使用核心散布到 1、2、4 台机器上的运行时间

3) 计算节点数量

最后固定每个机器使用最多的核心数，并通过增加使用的计算节点数量，来测试 GROMACS 的效能是否能获得增加。结果如图 12-8(a)所示，随着使用的机器数量增加，程

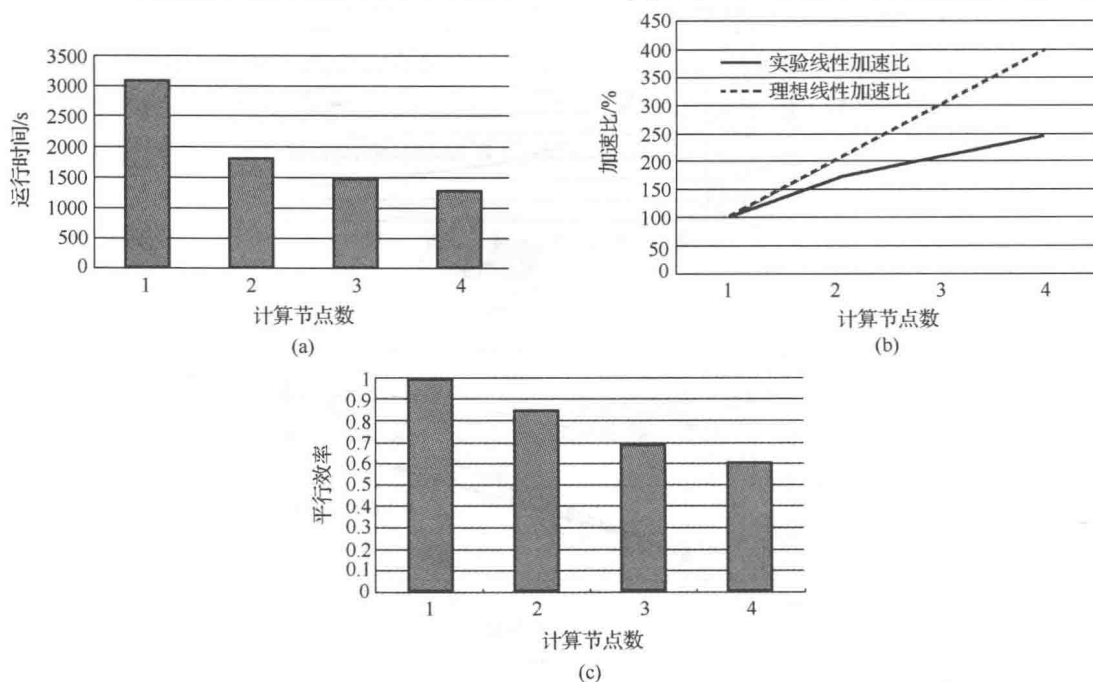


图 12-8 增加使用机器数的运行时间、运行时间加速比、平行效率

序运行时间也相对减少。不过从加速比的比较图中可观察到程序的平行效率会随着机器数量的增加而缓慢下降。如图 12-8(c)所示, 当使用 4 个节点时, 平行效率更降至只有 60%。这结果再次验证 GROMACS 的网络沟通会造成效能的降低, 因此网络的效能相对重要。

4) 总结

由以上结果可看出计算资源的量与配置方法对效能有显著的影响。尤其 GROMACS 在不同进程 (processes) 间会产生一定的数据沟通与传输量, 因此网络的带宽与效能会影响程序的效能, 并且应充分使用单一计算机中的计算核心, 以减少网络的影响。但整体来说 GROMACS 展现不错的效能扩展性, 平行效率不会因计算资源数量的增加而大幅下降, 所以可以通过不断增加使用的核心数和计算节点数来减少运行时间, 并以线性方程式准确地估算不同资源使用量下所需的运行时间。

12.6 应用 GPU 的效能测试

GROMACS 从 4.6 版开始通过 CUDA 编程使用 GPU 的性能优化。此版中的架构是一个 CPU 核心对应一个 GPU, 彼此间利用 OpenMP thread 来进行沟通。以我们的实验环境为例, 因为每个计算节点只配有一张 GPU 卡, 所以也代表每个计算节点只有一个核心会参与至实际的计算当中。不过因为 GROMACS 4.6 版是第一个支持 GPU 的版本, 仍有许多改善的空间, 所以预期未来的版本也可能会针对这项限制进行改变。根据现有的版本, 在图 12-9 中比较单一计算机使用一张 GPU 卡与原先只用所有 16 个核心但不用 GPU 卡的效能。从图中可以看出, 即使 GPU 的版本只使用一个计算核心, 运行时间仍然可以优于使用 16 个核心的 CPU 版本, 由此可看出 GPU 对 GROMACS 有相当不错的加速效果。但是目前当使用多台计算节点与 GPU 卡时, 却发现 GPU 版本的加速比大幅下降。原因推测是 GPU 卡通过 InfiniBand 传送内存中的数据至其他计算机时遇到效能瓶颈。这个问题不仅 GROMACS 的新版本会持续进行进一步的改善, NVIDIA 也已经在新的硬件设计和驱动程序中, 提出缩短 I/O 路径的解决方案, 更有许多进行中的相关研究。所以预期未来 GPU 卡间的网络沟通问题会是影响效能的关键, 虽然无法完全避免效能上的影响, 但应该可以达成类似于 CPU 版本的成果。可见使用 GPU 加速 GROMACS 应成为未来的效能优化的必备工具与趋势。

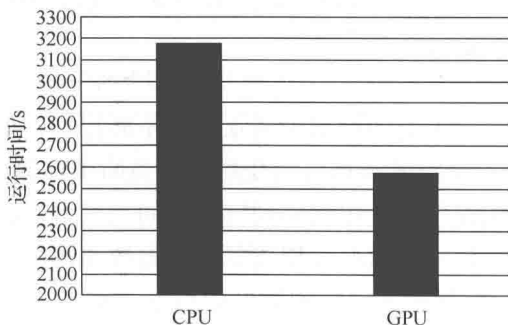


图 12-9 GPU 版本效能测试

第 13 课 气候系统海洋模式 LICOM 优化方案

13.1 气候系统海洋模式 LICOM 介绍

13.1.1 什么是 LICOM

LICOM 是中国科学院大气物理研究所 (IAP) 大气科学和地球流体力学数值模拟国家重点实验室 (LASG) 气候系统 (climate system) 海洋模式 (ocean model) 的英文缩写, 即 LICOM (LASG/IAP climate system ocean model)。LICOM 是由 LASG 全球海气耦合模式研究组设计和发展的, 目的是提供能够模拟大尺度海洋风生环流和热盐环流, 用以作为耦合气候模式的海洋分模式 (简称为气候海洋模式)。目前的耦合气候模式除了海洋和大气模式之外, 还包括海冰和陆面模式, 是一个多圈层相互作用的气候系统模式。LICOM 是 LASG 为发展气候系统模式所做努力的一个重要组成部分。

从计算机软件的角度看, LICOM 是一套求解偏微分方程组数值解的程序, 这套方程组描述了海洋环流和温盐的变化。LICOM 是用 Fortran90 语言编写的, 运用了当今常用的高性能计算并行方式, 可以在各种并行计算机和计算机集群上运行。

LICOM 所有的源码程序共 60 个, 其中 59 个 F90 程序, 1 个 C 程序, 总代码量大约有 1.8 万行。主程序为 licom.F90, 直接或间接调用其他子程序。

13.1.2 LICOM 的发展演化

随着 LASG 在过去十余年中先后推出的 4 层、20 层和 30 层海洋模式, LICOM 也是一个版本不断更新的模式系列, 最初推出的 LICOM 版本称为 LICOM1.0。LICOM1.0 是一个水平格距 $0.5^{\circ} \times 0.5^{\circ}$ 、垂直方向 30 层、覆盖了从 65°N 到 75°S 的准全球范围的海洋环流模式, 它是在原有的水平格距为 $1.875^{\circ} \times 1.875^{\circ}$ 的 30 层海洋模式的基础上发展起来的。2014 年世界大学生超级计算机竞赛决赛期间所用的 LICOM 为其最新版本 LICOM2.0。

13.1.3 LICOM 的优化意义

在高速发展的今天, 全球人类的物质生活水平在不断提升, 但人类活动对气候的影响也在加大, 特别是 20 世纪后半段的全球气候变暖对人类的生存提出了挑战。虽然全球变暖是整个气候系统的变化, 但在年际、年代际到百年时间尺度上, 海洋环流的变化在气候系统的变化中起到了决定性的作用, 而海洋模式是我们理解并预估海洋的变化的重要工具, 因此 LICOM 作为气候系统海洋模式研究的软件应用意义重大。关于 LICOM 的产学研, 其并行优化的工作亟待进一步开展与提高。

13.2 LICOM 优化的关键点分析

LICOM 的优化属于 ASC14 决赛的应用题之一,从赛制要求理解考虑, LICOM 的优化是对参赛队所搭建的硬件集群平台性能高低的基本体现,也反映了相应的软件环境配置的优劣,当然,关于 LICOM 的优化还有很关键的一点就是并行时的方案。由此,可以预先分析出 LICOM 优化的关键点如下。

(1) 试图将硬件集群的性能在不超过赛制规定的 3kW 功耗的前提下发挥到最大值。

(2) 选用更适合于 Intel Xeon 平台和 LICOM 应用的编译器与 MPI 库,如推荐使用 Intel 编译器、Intel MPI。

(3) 了解 Fortran90 语言,理解主程序 licom.F90 及相关子程序和 shell 脚本,以此逐步确定并行方案和决赛赛场上的最佳策略。

13.3 硬件环境搭建

我们综合 LINPACK 测试和包括 LICOM 在内的所有应用优化题目的情况,最终选择了搭建如表 13-1 所示的硬件平台,尝试以这样的方式将硬件资源的性能尽可能地发挥到最大。

表 13-1 ASC14 决赛期间太原理工大学 TUT 超算团队搭建的硬件平台环境

项目	名称		数量
服务器	Inspur NF5280M3	CPU: Intel Xeon E5-2692v2 $\times$ 2, 2.2GHz, 12 cores 内存: 16G $\times$ 8, DDR3, 1600MHz 硬盘: 1TB SATA $\times$ 1	10
HCA 卡	FDR		10
交换机	FDR-IB 交换机		1
网线	IB 网线		10

13.4 软件环境配置

对应于已经搭建好的硬件平台,因为服务器是 Intel Xeon 的 CPU,并根据以往的优化实践经验,最终集群的基础软件环境配置情况如表 13-2 所示。

表 13-2 ASC14 决赛期间太原理工大学 TUT 超算团队所建集群的软件环境配置

项 目	名 称	版 本
操作系统	Red Hat Enterprise Linux Server release 6.2 (Santiago)	2.6.32-220.el6.x86_64
编译器	Intel 编译器	12.10
	GUN 编译器	4.4.4
数学库	Intel MKL	11.1.0.080
MPI	Intel MPI	4.1.3.049
	MPICH2	2-1.5

至此，以下关键的步骤是需要是在集群基础环境之上独立地安装配置 NetCDF 库并安装配置 LICOM，详细步骤如下。

13.4.1 NetCDF C 库安装配置

在当前用户目录下创建 LICOM 的操作目录（~/LICOMtest/ASC14_LICOM/）和 NetCDF 的安装目录（~/local/netCDF/），NetCDF C 库使用最新的 netcdf-4.3.1.1 版本（可从 <http://www.unidata.ucar.edu/software/netcdf> 下载），安装配置步骤如下。

（1）configure 操作，等待该过程正常结束（注：经实践得出 configure 操作时的选项参数不止一种，最后都是可以得到正确结果的，这里仅写出一种操作简便的方法）。

```
[Speedyfeng@node1 netcdf-4.3.1.1]$ ./configure --prefix=/home/Speedyfeng/local/netCDF --disable-netcdf-4 --disable-dap
```

（2）make 操作，等待这一过程正常结束。

```
[Speedyfeng@node1 netcdf-4.3.1.1]$ make check install
```

（3）编辑.bashrc 文件，添加/home/Speedyfeng/local/netCDF/lib 至动态链接库，即修改环境变量，如果这一步没有操作，紧接着安装 NetCDF Fortran 库会报错。

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH: /home/Speedyfeng/local/netCDF/lib
```

13.4.2 NetCDF Fortran 库安装配置

在 NetCDF C 库的配置基础上，配置 NetCDF Fortran 库，NetCDF Fortran 库使用最新的 netcdf-fortran-4.2 版本，具体步骤如下。

（1）configure 操作，等待该过程正常结束。

```
[Speedyfeng@node1 netcdf-fortran-4.2]$ CPPFLAGS=-I/home/Speedyfeng/local/netCDF/include LDFLAGS=-L/home/Speedyfeng/local/netCDF/lib ./configure --prefix=/home/Speedyfeng/local/netCDF
```

（2）make 操作，等待该过程正常结束。

```
[Speedyfeng@node1 netcdf-fortran-4.2]$ make check install
```

（3）添加剩下的 NetCDF 相关环境变量。

```
export PATH=$PATH: /home/Speedyfeng/local/netCDF/bin
export NETCDF=/home/Speedyfeng/local/netCDF
```

13.4.3 LICOM 的安装配置

在确保基本软件环境如 Intel 编译器、Intel MPI 配置无误和 NetCDF 库配置无误后，编辑 LICOM2.0 的 case.sh 脚本，其中，需要相应修改的有 LICOM 的主目录项、运行模式、

进程数、编译器、并行库、NetCDF 库等 (LICOM2.0 的 OpenMP 未进行调试, 这里不介绍), 修改后的 case.sh 主要内容如下。

```
#set TESTNUM=01
setenv CASENAME test01 # 根据操作测试次数情况, 依次为 01、02、03...
setenv LICOMROOT /home/Speedyfeng/LICOMtest/ASC14_LICOM # LICOM dir
setenv SRCPATH $LICOMROOT/src
setenv BLDPATH $LICOMROOT/bld
setenv DATAPATH $LICOMROOT/data
setenv EXEROOT $LICOMROOT/$CASENAME
setenv EXESRC $EXEROOT/src
setenv EXEDIR $EXEROOT/exe
setenv RUNTYPE initial
set HISTOUT = 1
set RESTOUT = 1
set XNTASKS = 10
set YNTASKS = 20
set NTHRDS = 1
set LID = "'date +%y%m%d-%H%M%S'"
@ NTASKS = $XNTASKS * $YNTASKS

echo '-----'
echo '      Produce Makefile      '
echo '-----'

\cat >! Makefile << EOF
#####
# modify this part according to compiler and system you are using
# CPPFLAGS CPP INCLUDE NLIB FFLAGS FC
#####
CPPFLAGS = -P -traditional
CPP      = /usr/bin/cpp $(CPPFLAGS) $(INCLDIR)
INCLDIR  = -I. -I/usr/include -I/usr/local/include -I/home/Speedyfeng/
intel/mpi/4.1.2.040/include -I/home/Speedyfeng/local/netCDF/include
NLIB     = -L/home/Speedyfeng/intel/mpi/4.1.2.040/lib64 -lmpi -L/home/
Speedyfeng/local/netCDF/lib -lnetcdf -lnetcdff
FFLAGS   = -mcmmodel=large -shared-intel -O2 -r8 -i4 -convert big_endian
-assume byterecl -no-vec
FC       = /home/Speedyfeng/intel/mpi/4.1.2.040/intel64/bin/mpiifort $(FFLAGS)
#####
```

case.sh 中关键配置编辑的部分对应如图 13-1 所示。

通过编辑 vim case.sh, 重写上述脚本内容之后, 保存并退出。以命令 “./case.sh” 执行该脚本, 进行相应的编译、链接后生成对应的 test 目录, 进入 test/exe/目录, licom2 等文件已经生成, 接下来便可以进行测试和对应负载的运算了。



```
\cat >! Makefile << EOF
```

图 13-1 case.sh 关键配置部分

13.5 优化方案实施过程及结果

13.5.1 优化方案实施过程

对于 LICOM2.0, 我们参赛队在决赛现场进行了 MPI 的并行运算, 结合所搭建的硬件平台为 10 个节点的纯 CPU 架构, 每个节点 24 个核, 最终的并行执行的方案主体为 `mpiexec -ppn 20 -n 200 ./licom2 <ocn.parm`, 当然事先已经 `mpdboot` 了 `node1~node10` 共 10 个节点的 `mpd ring`, 这样的并行执行方案对于决赛现场四个负载中的 A、B、C 三个效果良好, 功耗监控显示基本维持在 2960W 上下做少许波动, 而负载 D 的运算很快超出 3000W, 我们紧急换用了备选方案, 将进程总数改为 180。其中采取这样的 MPI 并行方案主要是根据主程序 `licom.F90` 调用到的子程序运算规则, 以及多次测试后, 实践得出的论证结果, `param_mod.F90` 主体部分如下。

```
module param_mod
#include <def-undef.h>
use precision_mod
integer,parameter::jmt_global=JMT_GLOBAL ! Number of the EndGrid for Tracer
in Latitude.
integer,parameter::jmm_global=jmt_global-1
integer,parameter::jstart = 3 ! Starting Grid for Tracer in Latitude.
integer,parameter::imt_global=362 ! Number of Grid Points in Longitude
integer,parameter::km=30 ! Number of Grid Points in Vertical Direction
#ifdef SPMD
integer,parameter::nx_proc=NX_PROC ! Number of MPI tasks in zonal direction
integer,parameter::ny_proc=NY_PROC ! Number of MPI tasks in meridional
direction
integer,parameter::n_proc=nx_proc*ny_proc ! Total number of Processors for
MPI
integer,parameter::num_overlap=2 ! Number of overlapping grids for subdomain.
integer,parameter::jst_global=jstart ! Number of the Starting Grid for Tracer
in Latitude.
```

```

!Number of grids in the each subdomain
integer,parameter:: jst=1      ! Number of the Starting Grid for Tracer in
Latitude.
integer,parameter:: jsm=jst+1 ! Number of the Starting Grid for Momentum
in Latitude.
integer,parameter::
jjet=(jmt_global-jst_global+1-num_overlap)/ny_proc+1+num_overlap
integer,parameter:: jem=jjet-1 ! Number of the End Grid for Momentum in Latitude.
integer,parameter:: jmt=jjet
integer,parameter:: imt=(imt_global-2)/nx_proc+num_overlap
integer :: j_loop              ! Loop index of J cycle for the each subdomain.
#else
integer,parameter:: jst=jstart ! Number of the Starting Grid for Tracer in
Latitude.
integer,parameter:: jsm=jst+1 ! Number of the Starting Grid for Momentum
in Latitude.
integer,parameter:: jjet=jmt_global ! Number of the End Grid for Tracer in
Latitude.
integer,parameter:: jem=jjet-1 ! Number of the End Grid for Momentum in Latitude.
integer,parameter:: jmt=jmt_global
integer,parameter:: imt=imt_global
integer,parameter:: nx_proc=1
integer,parameter:: ny_proc=1
#endif

```

其中，定义整型变量 $nx_proc=NX_PROC$ (number of MPI tasks in zonal direction, 纬线方向的 MPI 任务进程数) 和 $ny_proc=NY_PROC$ (number of MPI tasks in meridional direction, 经线方向的 MPI 任务进程数)，与脚本 `case.sh` 中的设置是一致的 (`#define NX_PROC $XNTASKS`, `#define NY_PROC $YNTASKS`)，由本身定义和上述运算规则得知在 MPI 进程数设定时 `XNTASKS` 和 `YNTASKS` 都整除 360 的情况下便于数据分块 (如果 `XNTASKS`、`YNTASKS` 的值设置得不正确，运算过程中会直接报错)，相对于同样的硬件平台和环境配置，`XNTASKS` 和 `YNTASKS` 的乘积越大，即 `NTASKS` (MPI 运行的进程数) 越大，运算速度越快，后续的实际运算测试也证明了这一点。

结合所用的 10 个节点每个节点为两路 CPU 共 24 核的情况，最终 MPI 优化的情况是在 `XNTASKS=10`、`YNTASKS=20` 的情况下达到最佳，针对 MPI 节点间进程级的优化最好，加速比最大。当然，也预先准备了相应的备选方案，详见 13.7 节。

13.5.2 优化结果

根据在决赛正式开赛之前的调试情况和上述提到的主体优化方案，我们关于 LICOM 四个负载的运算不到中午就基本完成了，大致情况如下。

(1) 负载 A 采取 `initial` 运算模式，将编译、链接后生成的 `licom2` 复制至负载 A 所在目录，200 个进程进行 MPI 并行运算，功耗接近 3000W，最终耗时为 1008.17s。

(2) 负载 B 从负载 A 复制 `fort.22.0002-01-01` 文件，采取 `continue` 运算模式，将编译、

链接后生成的 licom2 复制至负载 B 所在目录, 200 个进程进行 MPI 并行运算, 功耗接近 3000W, 最终耗时为 534.85s。

(3) 负载 C 采取 initial 运算模式, 将编译、链接后生成的 licom2 复制至负载 C 所在目录, 200 个进程进行 MPI 并行运算, 功耗接近 3000W, 最终耗时为 1437.98s。

(4) 负载 D 采取 initial 运算模式, 但操作比之前更显复杂, 按照其中的 readme 中的要求, 进行了正确操作后重新编译程序, 将编译、链接后生成的 licom2 复制至负载 D 所在目录, 180 个进程进行 MPI 并行运算, 功耗接近 3000W, 最终耗时为 3794.27s。

13.6 ASC14 备赛过程回顾

2013 年底, TUT 超算团队组建之前, 从太原理工大学数学学院院长李明老师的手中接过 2014 年 ASC 世界大学生超级计算机竞赛的参赛邀请, 感觉到这既是一项挑战, 也是一种机遇, 而我作为整个团队的指导教师也感到任重道远。

2014 年 ASC 竞赛分为初赛和决赛两个阶段。初赛阶段为 2014 年 1 月 2 日~2 月 27 日, 根据竞赛命题要求, 各参赛队进行相关书面方案准备和应用优化, 由评审委员会评选出进入决赛的团队; 决赛阶段为 2014 年 4 月 20~25 日, 进入决赛的团队将利用浪潮集团提供的设备组建各自的超级计算机, 在满足竞赛规则的要求下, 根据自身对应用的理解进行优化设计, 现场进行性能测试、应用优化和竞赛呈现, 由评审委员会专家综合实际运算结果和呈现, 评选出最终名次。

初赛前的队员招募工作, 就是一个难题, 面对新的未知领域, 没有合适的队员, 比赛无从谈起。李明老师建议: 组建赛队不仅是基于眼前的比赛, 要长远地考虑, 必须储备竞赛的后备力量, 构建超算团队。因此, 在太原理工大学学生处的配合下, 我们及时地在校园网上发布了竞赛的队员招募信息, 找到了一批对超算感兴趣、有时间、有能力的同学, 他们来自不同的学院、不同的专业、不同的年级, 但都拥有对超算的向往和激情。

针对比赛要求, 我建议队员要有长远的学习计划, 使得队员没有盲目备赛, 而是令他们先思考, 让他们自己去探索问题难点及突破点, 理顺比赛的流程和框架, 至少清楚了哪些技术需要现学, 哪些技术有现成的硬件支持, 哪些环境需要先行配置, 哪些问题需要和大赛组委会进行进一步沟通了解, 以及需要准备哪些相关书籍资料等。所有这些都需要进行规划。有了规划, 接下来就得有强而有效的执行。队长曹晨曦每周定期组织学习例会, 来总结与分享队员各自的一周收获和心得。

为了更快、更好地备赛, 给队员分配了不同的赛题, 每位队员各有侧重, 遇到需要学习讨论的时候, 就在太原理工大学科学云计算中心的休息厅内工作, 一张小圆桌, 三个沙发, 一个白板, 队员亲切地称之为“思维殿堂”。为了共同进步和分享资料, 建立了超算 QQ 群, 每个队员都有权利和义务上传、下载文献资料, 并且将 QQ 等级改为“猜算”“手算”“珠算”“机算”“超算”等, 既显得创意无限, 又非常应景。对于部分队员来说, 整个备赛的过程就是一个从无到有、不断丰满的学习过程, 队员非常享受这种学习的乐趣, 并且更加热爱这种纯粹的状态。

由于初赛赛程较长, 中间有一个寒假的休息时间, 这既是一个良好的备赛时间段, 也

是容易松懈的时候,要克服队员的惰性,实为不易。为了保持队员的积极性,引入了淘汰机制,因为赛制中规定参赛队伍为“1 名指导教师+5 名学生”的模式,谁能最终代表学校参赛,完全取决于学生自己的表现,这不仅给同学们提供了一个公平参赛的机会,展现自己的舞台,也保证了最终参赛队伍的整体水平。当然这些学生我也都没有看错,他们寒假仅休息了一周左右,就提前返校继续投入了备赛。

初赛要求提供集群的配置方案,虽然大赛组委会提供了远程登录计算平台,但是考虑到实战的因素,队员需要在真实的集群环境中学习硬件知识和熟悉搭建流程,经过和李明老师商量,我们团队拆了科学云计算中心的两个计算节点,以此为平台,同时充分利用远程资源,做到了硬件资源的最大化利用。

对于占分值最大的 3D_EW 程序优化,虽然有所进展,却一直困于数据集的大小而无法移植到 MIC 上,队长曹晨曦和队员高燕军在“思维殿堂”相互讨论,高燕军拿立方体作比喻,队长的思维很敏锐,立刻想到了缩减数据集,基于这个思路以及后来的向量化优化和 MPI 动态调度,队长曹晨曦带头率领团队连续几个通宵奋战,为了模拟昼夜颠倒后的时区,专门把作息时间调成加拿大圣约翰时间。正是源于队员们的团结拼搏,最终在截止日期前 3 天成功地将 3D_EW 应用提速了 400 倍以上,凭借惊人的加速比,TUT 超算团队在全球五大洲 82 支参赛劲旅中,以海内参赛高校第一名的成绩成功晋级,入围决赛 Top16。

晋级的消息是喜悦的,但也是吹响继续比赛的号角。面对即将在广东中山与国内外多支超算强校现场 PK 的情况,更大的压力向队员们悄然袭来。这时我和队员们说得最多的就是“抱着积极参与和学习的心态来认真参赛就可以了,不要去管比赛的结果如何,即使是最后一名,我们能和清华等知名高校同台竞技,已经成功了”。

13.7 LICOM 于 ASC14 决赛赛场优化回顾

作为闯入决赛的“黑马”,对于决赛丝毫不敢懈怠,队员张建峰具体负责 LICOM 优化,开展了多项实践操作,决赛赛场回顾如下。

4 月 21 日,正式比赛前第一天调试,上午队员们进行了 MIC 卡的拆卸、集群的组装搭建、Linux 系统的安装配置、IB 网络配置、NFS 配置等工作,下午进行 LINPACK 和各项应用测试,期间遇到了一个问题就是 mpd ring 总是创建不起来,也就意味着 MPI 的并行测试无法开展下去,傍晚当其他队伍都离开赛场之后,队员张建峰还在利用那仅剩的一点时间测试自己的想法。这个问题一直困扰团队所有人直到深夜,根据报错的信息大家在 Intel 官网论坛等处“疯狂”搜寻解决方案,并提出了多种预备解决方案。

4 月 22 日,正式比赛前第二天测试,启动机器,结果发现 MPI 并行又完全没问题了,大家郁闷的同时也很激动,之后迅速进入状态,开始各项题目的测试工作,其中 LICOM 的测试很顺利,先后测试了 XNTASKS=10、YNTASKS=24, XNTASKS=8、YNTASKS=24, XNTASKS=10、YNTASKS=20, XNTASKS=20、YNTASKS=10, XNTASKS=10、YNTASKS=18, XNTASKS=18、YNTASKS=10, XNTASKS=9、YNTASKS=20, XNTASKS=20、YNTASKS=9 等多种情况,最终确定了以 XNTASKS=10、YNTASKS=20 的 200 个进程的 MPI 并行作为第一方案, XNTASKS=10、YNTASKS=18 为备选方案,其中不断调试的过程中,我们总结

出了根据运算产生的积分文件的数量和速度，并结合 top 显示的运算所耗时间来推测所在运算方案下的最终预计耗时，为整个调试节省了不少时间，也有着不错的成效。张建峰在 LICOM 正式比赛前一天专门写了名为“明天 LICOM 操作规划”的详细技术文档，为 LICOM 在正式比赛期间的运行节省了时间，其中对 case.sh 配置进行了详细说明部署，对 MPI 并行方案分情况进行了分析，同时将“`mpiexec -ppn 20 -n 200 ./licom2 <ocn.parm`”写成可执行脚本 LICOM-A.sh，并提前写好了运行计时的命令行操作“`[Speedyfeng@node1 A]$ /usr/bin/time -p ./LICOM-A.sh > TUT.LICOM.A.TIME-200-001.TXT 2>&1`”，方便直接粘贴使用。

4月23日，正式比赛第一天，进行了 LINPACK 测试，QE、SU² 两项应用和 3D_EW 项目比拼。我们的 LINPACK 测试水平还是和超算强校有一定的实力差距，面对现实，综合考虑整个赛程的时间和赛题分值分布，决定要尽快完成 LINPACK 测试，为后面的其他应用腾出宝贵的测试时间，可是负责 LINPACK 测试的队员张翥茜花了半年的时间备赛，仅几分钟就结束测试，这对初次参赛的队员来说，是很难让人接受的，但这就是比赛，一切为了比赛，就得有所取舍，最终做通了张翥茜的思想工作，在 LINPACK 测试上节省了大量宝贵的时间。队长曹晨曦负责的 3D_EW 战况更加激烈，凭借初赛时的战果和决赛时的努力，3D_EW 取得了不错的成绩。

4月24日，正式比赛第二天，进行 LICOM 和神秘应用的比赛项目，按照预先的调试，LICOM 很顺利地测试完毕，当然期间也出过一些小问题，就是运算到负载 B 时，可能由于张建峰太紧张，没仔细看清 readme 的要求，运算模式还是 initial，首先显示报错，根据报错的结果显示，将运算模式改为了 continue，提示找不到积分文件 fort.22.0002-01-01，查看了其他负载的 readme 也没找到相关提示，误认为是组委会这道题出得有问题，就直接找了大赛组委会进行了讨论。最后还是证明了自己的失误，过度的精神集中在一定程度而言也是把双刃剑。

4月25日，上午进行结果呈现，即答辩，我们抽签的顺序还比较理想。结合自身的情况，在应用介绍方面，稍偏重了 3D_EW 和 LICOM，这样的方案也取得了不错的成效，答辩结束时所有的队员都松了一口气。下午当我们两次站上领奖台高举奖杯时，所有队员的激动心情都溢于言表。

在广州的每个日日夜夜，我都能感受到队员们的团结和努力。由于白天赛事紧张，我们的团队总是分批轮流吃饭，甚至达到了忘食的境界，每天比赛快要结束而组委会开始清理赛场时，我们总是最后一个离开赛场，争取每一刻的宝贵时间，晚上更是抓紧时间来调试程序，几乎每天都是凌晨两点以后休息，早上 6 点多就被闹钟叫醒，因此睡眠严重不足，尤其是准备在竞赛平台上开跑 3D_EW 之前，队长曹晨曦通宵调试程序，最终使我们的应用排名第二；最后答辩前夜，所有队员又是熬夜修改 PPT，队员田友琳从进入赛场，抽空就准备英文讲稿，这样的搏命付出，为团队最终优异的成绩奠定了基础。

最终我们取得的成绩，离不开学校的支持、团队的合作和队员们的拼搏。一路走来，回想赛程，我认为团队的分工协作至关重要，因为没有谁是全能的，而且新手只能靠团队协作取胜。我们做到了“有福同享”，初赛前队员们一起去远足、野餐，工作玩乐都在一起；“有难同当”，遇到难题，就有策略地发起全员进攻，经常一起通宵奋战，这样逐渐培养起了团队默契和协作的能力。当然在任何比赛中，积极和认真是参赛队员必须具备的素质，

整个比赛进程中，我一直强调和要求队员们必须认真。ASC 14 作为一项世界级的赛事，其评委由国内外多位知名专家组成，评价的指标和水准必然很高，面对高要求、高指标，团队始终保持着高度认真的态度，面对实际情况，我们从技术测试运行、优化实践，到最后的数据整理、英文文档书写将“认真”二字一路执行到底。

13.8 LICOM 优化总结及评估

回看 LICOM 整个优化过程，我们的实践经历得出 LICOM 前期的配置工作和后续的优化各占了 50%，尤其是配置工作中的 NetCDF C 库的配置，我们经过多次尝试，发现 NetCDF 的配置方式不是唯一的，我们成功试验了其中的两种，并选取了较为简洁的一种作为常用配置方法（前面 13.4 节已写到）。另外，配置了 NetCDF Fortran 库后，如果在 case.sh 脚本 NLIB 一项中不添加“-lnetcdf”，执行 case.sh 进行编译会报错，编译不成功。

关于 LICOM 的优化，虽然太原理工大学 TUT 超算团队取得了不错的成绩，但我们深知，LICOM 的优化还远不止这些，或者说我们在决赛期间所做的优化也是远远不够的，大赛结束后，我们又对 LICOM 进行了 OpenMP 的并行优化，将 MPI+OpenMP 混合并行优化的技术在 LICOM2.0 上进行了实现，又取得了新的进步，太原理工大学 TUT 超算团队合影如图 13-2 所示。

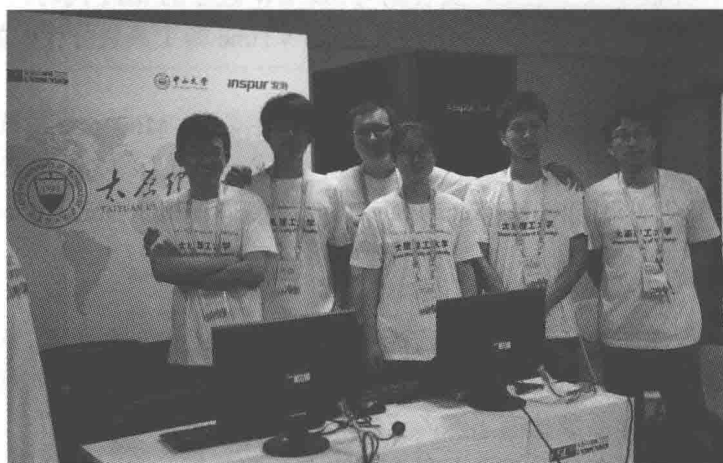


图 13-2 太原理工大学 TUT 超算团队合影

最后，感谢大赛组委会给我们乃至全球的超算人才提供了一个展现自我的平台，同样感谢 ASC 14 竞赛期间，浪潮集团高性能服务器产品部的工程师对所有参赛队伍在技术方面提供的咨询和指导。我们关于超算领域的应用优化经验还不是特别丰富，水平还有待提高，希望各位专家读者和志同道合之士多多交流指导，对本章中的不当之处及时指正，我们将继续努力。

第 14 课 三维纵横波分离的弹性波方程

模拟方法 3D_EW 优化方案

14.1 组队参赛情况及参赛准备

14.1.1 人员准备

上海交通大学 ASC14 队伍负责 3D_EW (3D elastic wave modeling) 的团队总共包括 5 人, 由高性能计算中心副主任林新华、工程技术专员文敏华提供技术指导, 3 名 ASC14 学生队员进行代码移植和优化的具体工作。3 名学生的背景及分工如下。

(1) 赵嘉明, 计算机系大四学生, 代码移植主要负责人, 负责 3D_EW 代码在 MIC 平台上的移植和优化。

(2) 袁野, 计算机系大三学生, 负责编写 3D_EW 程序的 MPI 代码。

(3) 肖迪, 软件学院大三学生, 负责使用 Intel VTune 等工具进行代码的剖析 (profile) 工作, 为上述两位同学提供技术支持。

同时, 3D_EW 团队还获得了高性能计算中心的硬件支持: MIC 程序测试平台不仅基于浪潮提供的 MIC 节点, 还包括上海交通大学高性能计算中心 π 集群上的 5 个 MIC 节点, 节点配置为 2×Intel Xeon E5-2670 CPU+2×Intel Xeon Phi 5110p, 节点间使用 56Gbit/s Infiniband FDR 互连。

14.1.2 技术准备

由于 3D_EW 程序需要在天河二号竞赛平台进行并行优化, 而天河二号 85% 以上的计算能力来自基于 KNC (knights corner) 架构的 MIC 加速卡。我们对 KNC 体系架构进行了一系列的调研, 对它的体系架构有了深入了解。

KNC 与 CPU 相比, 在体系架构上有 3 个特点。正是基于这 3 个特点, 我们采取了与之相对应的并行优化方法。

1) 宽度为 512bit 的向量处理单元 (vector processing unit, VPU)

(1) VPU 对 KNC 性能至关重要。

宽度为 512bit 的向量可容纳 8 个 64bit 的双精度浮点数或 16 个 32bit 的单精度浮点数。KNC 峰值性能的计算公式为 $\text{Peak Performance} = \text{Clock Frequency} \times \text{Cores} \times \text{Lanes} \times 2(\text{FMA}) \text{ Flops/Cycle}$, 其中 FMA (fused multiply and add), 即乘加融合, 是指在 1 条指令中同时执行 1 个乘法浮点操作和 1 个加法浮点操作。以上海交通大学超级计算机 π 上配置的 KNC 5110p 为例, 双精度峰值性能为 $1.05\text{GHz} \times 60\text{cores} \times 8 \text{ Lanes} \times 2(\text{FMA}) \text{ Flops/Cycle} = 1008\text{Gflops}$, 单精度则为 2016Gflops 。由此可见, 要在 KNC 上达到高性能, 必须把 VPU 充分利用起来。

(2) 数据不对齐会降低 VPU 利用率。

KNC 每个线程有 32 个寄存器 (register)，每个长度均为 512bit。因此读入对齐的数据只需 1 次操作，而不对齐的数据往往需要两次操作。同样，写回时也是如此。

(3) gather/scatter 是 VPU 的性能瓶颈。

KNC 使用 IMCI-512 指令集，直接支持 gather/scatter 操作，但这往往成为性能瓶颈。究其原因是 gather 操作的实现使用了循环的方式，将分散在各处的 CL (cache line) 逐个读入。因此数据越分散，所需读入的 CL 就越多，延时也就越大。

综上所述，宽度为 512bit 的 VPU 对数据对齐要求更苛刻，而且应当尽可能避免 gather/scatter 操作。

2) 硬件无法隐藏 cache 延迟

(1) 高 cache 延迟。

以上海交通大学超级计算机 π 上配置的 KNC 节点为例，如图 14-1 所示，Intel Sandy Bridge E5-2670 的 L1、L2、L3 cache 的延迟分别为 1.5ns、4.6ns、81ns，而 KNC 5110p 的 L1、L2 cache 和主存的延迟分别为 2.9ns、22.9ns 和 295ns。

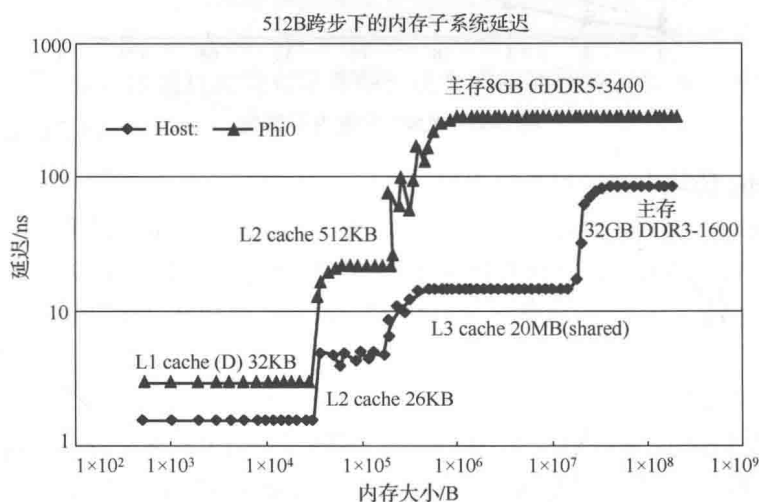


图 14-1 KNC 与 CPU 的 cache 延迟对比

(2) 低 ILP。

根据经验公式，能耗=1.73×性能，使用乱序执行 (Out-of-Order, OoO) 的 P6 比使用顺序执行 (in-order core) 的 P54C 能耗提高 3 倍，但性能只提升了 1.9 倍。受限于能耗和工艺，KNC 采用了循序执行的 P54C，但因此损失了指令集并行 (instruction level parallelism, ILP)。

(3) 少硬件 cache prefetcher。

受限于能耗与空间，与 Intel Xeon E5-2670 相比，KNC 不仅没有 L1 cache prefetcher，就连 L2 cache prefetcher 也打了对折，即只有 16 个 stream prefetcher，而没有 stride prefetcher。

综上所述，CPU 可以通过乱序执行的 core 和硬件 cache prefetcher 等方法隐藏 cache 延迟，而 KNC 却不行，甚至配合编译器之后都收效有限，因此程序员需要关注 cache 预取。

3) 内存带宽有限、cache 大小有限

(1) 实测带宽不到理论带宽的一半。

KNC 有 8 个内存控制器，每个能访问两条通道，每条通道的接口速度为 5.5TB/s，每个内存事务 (transaction) 大小为 4B，因此 KNC 的理论带宽为 $8 \times 2 \times 5.5 \times 4 = 352\text{GB/s}$ 。如图 14-2 所示，但使用 Stream 实测的带宽为 140GB/s，还不到理论值的一半。以 256 个线程计算，平摊下来每个线程所能占用的内存带宽仅为 0.5GB/s，远小于 CPU。

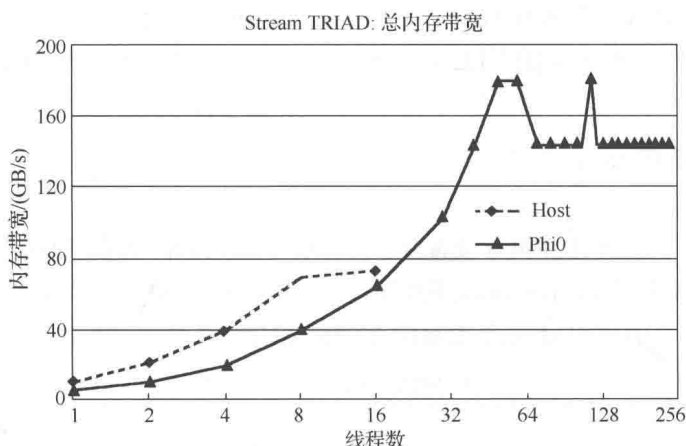


图 14-2 KNC 实测内存带宽

(2) L2 cache 伪共享及远程访问很慢。

KNC 5110p 每个核有 512KB 的 L2 cache，聚合起来有 30MB。但每个核不能直接访问其他核的 L2 cache 中的 CL，必须要把这条 CL 复制到自己的 L2 cache 中才行。雪上加霜的是访问其他核的 L2 cache 仅比访问主存快 17%，且与双向环 (Bi-direction ring) 上两核之间的物理距离无关。

(3) 没有 L3 cache。

KNC 没有 L3 cache 是由于片上空间不够。根据经验公式，下一级 cache 大小应该为上级 cache 大小总和的 8 倍。例如，KNC 单核的 L1 cache 为 32KB (数据) + 32KB (指令) = 64KB，则 L2 cache 为 $64\text{KB} \times 8 = 512\text{KB}$ ，以此类推，L3 cache 应为 $(64\text{KB} + 512\text{KB}) \times 8 = 4.5\text{MB}$ ，聚合 L3 cache 将会达到 270MB。这么大的 cache，就算用上 Intel Knights Landing (KNC 的下一代产品) 的 14nm 工艺，其面积也会超过 500mm^2 ，而目前 Intel 最大的芯片面积不超过 200mm^2 。

综上所述，由于 KNC 内存带宽有限，cache 大小有限，所以应当尽可能减少 cache 与主存间的通信，需要采用 NT-Store 优化。

14.2 ASC14 3D_EW 比赛规则解读

3D_EW 是 ASC14 比赛最有挑战的题目，参赛队伍需要针对给定的真实应用 3D_EW 程序在目前世界上最快的超级计算机天河二号上进行优化。优化的平台不止包括传统的

CPU, 还包括 Intel 近两年推出的 MIC 加速卡。这意味着我们要考虑的不仅有单节点内的 CPU 多核并行和节点间的 MPI 并行, 还将包括节点内的异构计算优化, 即 CPU、MIC 的协同计算。涉及的并行概念包括共享内存、消息传递和异构计算等。这对于参赛本科生来说具有一定的难度。

3D_EW 也是 ASC14 比赛分值最大的题目, 达 20 分, 而且这 20 分的构成与传统比赛也不一样, 包括 17 分的常规赛 (regular season) 和 3 分最终决赛 (finals)。其中常规赛所有队伍均有资格参加而最终决赛仅允许常规赛前两名参加。因而进入最终决赛的队伍相对于其他队伍在总成绩方面将很有优势。

综上, 不管从难度还是从重要性来讲, 3D_EW 都是 ASC14 比赛准备的重点, 因而我们将整个团队的主要资源放在该题目上。

14.3 三维纵横波分离的弹性波方程模拟算法介绍

3D_EW 由中国石油东方地球物理公司自主研发, 是用于石油勘探领域的真实应用。它是一种用波场延拓的方法来模拟弹性波在各向同性的弹性介质中传播的方法。在这个程序中, 纵波 (P 波) 和横波 (S 波) 被分别模拟, 这样可以更好地得知纵波和横波在弹性介质中的传递。该方法可以通过高阶有限差分方法来模拟弹性波的传递。其求解的方程为弹性波波动方程, 如下:

$$\rho \frac{\partial^2 S}{\partial t^2} = (\lambda + \mu) \nabla \theta + \mu \nabla^2 S$$

串行算法流程如图 14-3 所示。在输入数据并完成初始化工作后, 程序进入 ishot 循环, 即计算不同震源的影响。对于不同的 ishot, 计算任务完全独立, 因而这里属于易并行问题。之后进入 l 循环, 这里是时间推进方法, 不同的 l 表示不同的时间步, 前一次迭代的结果将会用于后一步的计算, 因而不同的 l 之间存在极强的数据依赖。在 l 以内, 则是不同网格的计算, 对于显式格式的有限差分来说, 不同网格的计算是完全独立的, 因而这里也具备很强的并行性。

```
initialize the wave field in the 3D domain
for(ishot=0;ishot<nshot;ishot+1)
//different ishot means different source of wave
{
    ...
    for(l=0;l<lt;l++)//lt means time step
    {
        ...
        for i,j,k in x,y,z direction
        {
            compute P-wave and S-wave in each grid of the 3D domain
        }
    }
}
```

图 14-3 3D_EW 程序流程

14.4 3D_EW 程序分析

14.4.1 热点分析

3D_EW 程序结构相对简单, 并没有复杂的函数调用, 主要的计算都是在循环内, 而 l 之间数据依赖, l 循环内部则是图 14-4 所示的 kji 三层循环, 在最内层的 i 循环中, 仅有简单的加法和乘法运算, 很显然, 程序的热点就是 kji 循环内的计算。

```
ntop = ntop-1;
nfront = nfront-1;
nleft = nleft-1;
for(k=ntop;k<nbottom;k++)
    for(j=nfront;j<nback;j++)
        for(i=nleft;i<nright;i++)
        {
            if(i==ncx_shot-1&&j==ncy_shot-1&&k==ncz_shot-1)
            {
                px=1.;
                sx=0.;
            }
        }
```

图 14-4 3D_EW 程序内层 kji 循环

14.4.2 代码分析

对 3D_EW 仔细阅读后, 我们发现它有如下 3 个主要特点。

1) 程序并行性非常好

最外层的 $ishot$ 循环计算部分完全没有数据依赖, 属于易并行问题; 内层的 kji 循环在计算和写数据时也完全没有数据依赖, 也属于易并行问题。因而整个程序具有非常好的并行性, 非常适合在超级计算机上并行实现。

2) 内存访问存在大量的单元跨步访问 (unit stride access)

在内层的 kji 循环中, 在计算每个网格的物理量时, 要读取周围 5 个点的数据以进行计算, x 方向上的数据读取时空局局部性较好, 而在 y 、 z 方向上的跨步访问则地址跨度非常大。前面提到 KNC 的硬件无法隐藏 cache 延迟, 因而这种数据读取方式对于 KNC 优化来说是非常不利的。

3) 数据结构规则且负载均衡

数据结构在高性能计算中的影响非常大。3D_EW 全部采用简单的数组结构, 这对于并行编程来说是非常有利的。而且计算域是简单的三维立体拓扑结构, 循环内几乎没有分支, 虽然计算域随着 l 的增加而增大, 但是 l 增大到一定程度使得计算域到达设定的边界时, 每一次 l 迭代的计算量趋于稳定, 而 kji 循环内的所有计算量几乎一致, 因而整个程序的负载也非常均衡。

14.5 基于 CPU+MIC 集群平台的 3D_EW 并行算法设计与实现

14.5.1 基于 CPU+MIC 集群平台的 3D_EW 并行算法整体设计

考虑到处理的数据和计算量庞大,我们使用 MPI+OpenMP 实现 3D_EW 程序在多节点异构平台的并行实现。其中, MPI 用于节点间并行,考虑到 ishot 循环计算完全独立,使用 MPI 并行将计算任务分配到不同的计算节点。而 OpenMP 用于节点内并行和处理器内向量化处理。对于 l 循环,由于数据依赖,无法并行,所以只能串行执行。而对于内层的 kji 循环,计算完全独立,我们使用 OpenMP 实现该计算的并行化。

14.5.2 基于 CPU+MIC 集群平台的 3D_EW 并行算法实现

1) OpenMP 实现线程级并行

OpenMP 是一种基于共享内存的并行编程标准。在使用 OpenMP 时,我们通过生成多个线程(thread)来使用现代处理器的多核系统,达到并行加速的效果。相对于多进程(process)的并行方法,OpenMP 的优势在于,线程级并行属于轻量级并行方法,所有线程共用一个存储空间,因而数据共享的开销比较小。因而对于节点内的并行非常适合用 OpenMP。

对于 3D_EW 程序,最外层的 ishot 循环不存在数据依赖,技术上也可以使用 OpenMP 并行,然而考虑到我们的并行还要基于多节点,而且对于不同的 ishot 要开辟独立的存储空间,这对于内存的压力非常大,因而使用 MPI 并行该循环,相关技术细节后续详细讲述。对于 l 循环,由前面所述,这个迭代过程前后数据依赖非常强,完全不适合并行,因而也不予考虑。对于 kji 循环,具有较好的并行性,且大部分计算量均在该循环,比较适合使用 OpenMP。而最内层的 kk/kkk 循环则是 kji 循环内比较耗时的地方,虽然并行性也比较好,然而循环次数较少,每个循环只执行 5 次,不管对于多核的 CPU 还是众核的 MIC,都不适合将其展开并行。综上, kji 循环是最适合使用 OpenMP 实现线程级并行的循环,最终决定在 kji 循环实现 OpenMP 并行。

通过对 kji 循环内的代码仔细分析,发现里面的计算是完全独立的,考虑到内层循环之后还要进行计算核心内的进一步优化(如向量化计算),我们将 OpenMP 并行放在最外层的 k 循环外面。之前讲过 OpenMP 的一个优点在于它是轻量级并行方法,它的另一个优点是它的编程难度较低,程序员只需要在要并行的循环外面添加一些制导指令即可以实现并行。

我们的 OpenMP 线程级并行如图 14-5 所示。

```
#pragma omp parallel for default(shared) private(px, k, i, j, kk, kkk, mvvp2, mvvs2, mtempux2, mtempuy2, \
mtempuz2, mtempvx2, mtempvy2, mtempvz2, mtempwx2, mtempwy2, mtempwz2, mtempuxz, mtempuxy, mtempvyz, \
mtempvxy, mtempwxz, mtempwyz) num_threads(TCPU)
for(k=ntop; k<nbottom; k++)
{
    if(k + zlow - 5 < 285)
    {
        mvvp2 = 2300 * 2300;
        mvvs2 = 1232 * 1232;
    }
}
```

图 14-5 使用 OpenMP 实现线程级并行

2) SIMD 实现数据级并行

SIMD (single instruction multiple data) 是一种数据级 (data level) 并行方式, 是一种采用一个控制器来控制多个处理器, 同时对一组数据的每一个分别执行相同的操作, 从而实现空间上并行性的技术。现代许多基于高速缓存的处理器都支持 SIMD 操作的指令集。在我们使用的处理器中, Sandy Bridge 架构的 CPU 具有 256bit 宽度的向量处理器, 而 MIC 的 VPU 则支持 512bit 向量化处理。SIMD 技术的使用极大地增加了程序的并行性能, 以 MIC 为例, 512bit 的 VPU 能同时处理 8 个双精度数据/16 个单精度数据。

我们在实现 3D_EW 并行时对 CPU、MIC 部分的计算均使用了 SIMD 优化。由于这两个平台均基于 x86 架构, 向量化的方法也比较类似, 在要并行的代码前添加“#pragma simd”指令即可。

由于 SIMD 优化时处理单元基于相同的缓存系统, 所以同一个 SIMD 处理器要处理的数据越密集、读取越规则, 则越有利于获得更高的性能。在 3D_EW 程序中, 处理的大部分数据是三维数组, 而 C 程序是基于行存储的, 源程序 *kji* 循环内数组的索引也是基于最内层 *i* 循环的, 因而对 *i* 循环进行了向量化处理, 如图 14-6 所示。这样可以使得同一条 SIMD 指令处理的数据跨度更小, 增加数据读/写的空间局部性, 更加高效地利用读/写带宽。

```
for(k=ntop;k<nbottom;k++)
{
    for(j=nfront;j<nback;j++)
    #pragma simd
        for(i=nleft;i<nright;i++)
        {
            for(kk=1;kk<=mm;kk++)
            {
                for(kkk=1;kkk<=mm;kkk++)
                {
```

图 14-6 使用 SIMD 实现数据级并行

使用 SIMD 指令在 CPU 程序上能获得很好的加速效果, 然而在基于 KNC 架构的 MIC 加速卡上性能却不理想。如前面所述, 相对于 CPU, KNC 具有计算无法隐藏数据读取延迟的缺陷。对于 3D_EW 程序, 其热点部分存在大量的单元跨步访问, 即读取的数据存储非常分散, 这导致在向量化后, VPU 读取数据时存在大量的 gather/scatter 操作 (这可以从生成的汇编代码看到, 如图 14-7 所示), 而这种操作对分散在各个 CL 的数据采取逐个读入的方式, 因而效率非常低。在我们的 MIC 实现中, 使用了底层的编程指令集 C Intrinsic (如图 14-8 所示) 来控制向量化的数据读取操作, 从而有效减少了 gather/scatter 操作, 使得 MIC 平台上程序性能获得了质的提升。

除此以外, 我们还在向量化时也进行了数据对齐的优化。在 MIC 上开辟内存时使用 `_mm_malloc` 语句可以保证数据对齐, 这样可以将一个 CL 的数据一次性读入向量寄存器, 如图 14-9 所示。

而如果没有对齐, 则可能需要两次, 如图 14-10 所示。


```

..LN2796:
    .loc    1 256 is_stmt 1
    vscatterpf0hintdpd (%r14,%zmm0,8){%k1}          #256.21 c13
..LN2797:
    vscatterpf0hintdpd (%r14,%zmm0,8){%k1}          #256.21
..LN2798:
    nop                                              #256.21
..L300:
    #
..LN2799:
    vscatterdpd %zmm2, (%r14,%zmm0,8){%k1}          #256.21
..LN2800:
    jkzd     ..L299, %k1    # Prob 50%              #256.21
..LN2801:
    vscatterdpd %zmm2, (%r14,%zmm0,8){%k1}          #256.21

```

图 14-7 汇编代码中的大量 gather/scatter 操作

```

_a = _mm512_load_pd(mic0_wp1 + k*ny*nx+j*nx+i);
_b = _mm512_load_pd(mic0_wp2 + k*ny*nx+j*nx+i);
_c = _mm512_add_pd(_mtempuxz, _mtempvyz);
_c = _mm512_mul_pd(_tem, _c); _c = _mm512_add_pd(_c, _mtempwz2);
_a = _mm512_mul_pd(_arrayc, _a); _c = _mm512_add_pd(_a, _c);
_c = _mm512_sub_pd(_c, _b);
_tem = _mm512_load_pd(pxtem);
_c = _mm512_add_pd(_c, _tem);
_mm512_mask_store_pd(mic0_wp + k*ny*nx+j*nx+i, _mask, _c);

```

图 14-8 使用 C Intrinsic 优化

```

_a = _mm512_load_pd(mic2_vp1 + k*ny*nx+j*nx+i);

```

图 14-9 数据对齐读取方式

```

_a = _mm512_loadunpacklo_pd(_a, mic2_u+ k*ny*nx+(j+kkk)*nx+(i+kk));
_a = _mm512_loadunpackhi_pd(_a, mic2_u+ k*ny*nx+(j+kkk)*nx+(i+kk+8));

```

图 14-10 数据不对齐读取方式

3) CPU、MIC 协作计算

由于测试所用平台为双路 CPU 节点，CPU 计算能力达到 0.3~0.4Tflops，相对于 MIC 加速卡的 1Tflops 也非常可观，而如前面所述， kji 循环内的计算并行性很好，因而我们将部分计算任务也分配在 CPU 上。而且 ASC 比赛所给的部分算例所需内存非常大，超过 MIC 卡的 8GB 内存，因而一块 MIC 卡往往无法单独处理 l 循环内的所有计算，所以在并行实现中对整个计算域进行区域分解，考虑到数据的存储方式为先按 xy 平面连续存储，以及考虑到编程难度的控制，我们只是简单地在 z 方向上进行了分解。如图 14-11 所示，竞赛平台上单节点为双路 CPU 加 3 块 MIC 卡，在每次 l 循环内， kji 循环的计算任务按 CPU 和 MIC 的计算能力进行一定比例分配，在执行完一次 kji 循环后各 MIC 卡和 CPU 之间进行同步及数据交换，用于下一步 l 循环的计算。

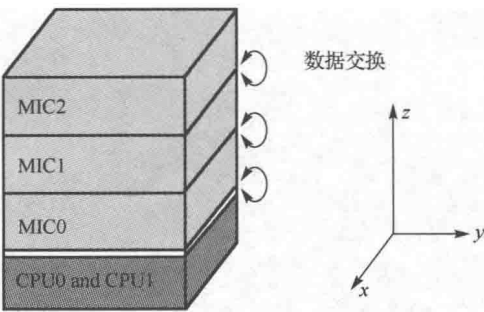


图 14-11 单节点内 CPU 和 MIC 的数据交换

4) MPI 实现

MPI 是一种基于消息传递的并行编程标准，该标准支持现在的大部分并行计算机，具有良好的可移植性。通过 MPI，可以创建多个进程（process），并将粗粒度的计算任务分配到这些进程。MPI 的一个特点是不同进程的数据无法直接共享而只能通过消息传递来达到不同进程的数据通信，相比于共享编程模型来说成本更高，然而这种并行编程模型具有更好的可扩展性。

对于 ishot 循环，计算部分完全没有数据依赖，而且处在最外层循环，非常适合使用 MPI 跨节点并行。对于 ishot 很大的大规模算例，可以将不同的 ishot 计算通过 MPI 分配给多个进程，图 14-12 所示的是 ishot=8192 的算例被分配在 1024 个计算节点并行计算。

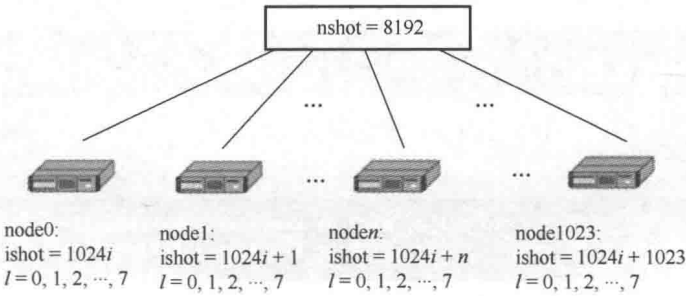


图 14-12 计算任务扩展到 1024 个节点

14.6 基于竞赛平台的 3D_EW 测试和分析

14.6.1 测试平台环境

我们在进行 3D_EW 并行和优化工作时主要基于π 集群（中小规模算例），最终决赛（大规模算例）则在天河二号上运行。

π 集群是浪潮集团搭建的 CPU+GPU+MIC 的异构超级计算系统，部署于上海交通大学高性能计算中心，峰值性能 263Tflops，HPL 实测 192Tflops，目前为教育部所属高校最快的超级计算机。单节点配置如表 14-1 所示。

表 14-1 π 集群单节点配置

软硬件	配置
CPU	Intel SNB E5-2670 @2.7GHz
KNC	Intel KNC 5110p @1.05GHz
编译器	Intel Composers XE 2013

天河二号是国内自主研发的超级计算系统，峰值计算速度 54.9Pflops，HPL 实测 33.9Pflops 的双精度浮点性能三次蝉联 Top500 榜首。天河二号由 16000 个节点组成，累计包含 32000 个 CPU，48000 块 Intel MIC 加速卡，其中超过 85% 的计算性能来自 MIC 卡，是典型的异构计算平台。天河二号单节点配置如表 14-2 所示。

表 14-2 天河二号单节点配置

软硬件	配置
CPU	Intel IVB E5-2692 @2.2GHz
KNC	Intel KNC 31S1P @1.1GHz
编译器	Intel Composers XE 2014

14.6.2 单节点性能

单节点使用两个算例进行了测试，结果分别如图 14-13 和图 14-14 所示。

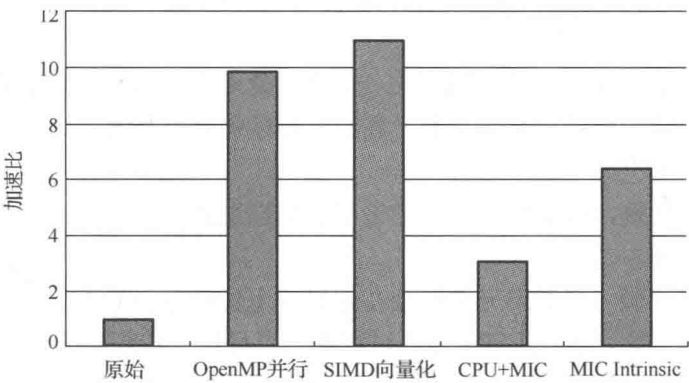


图 14-13 小规模算例加速比

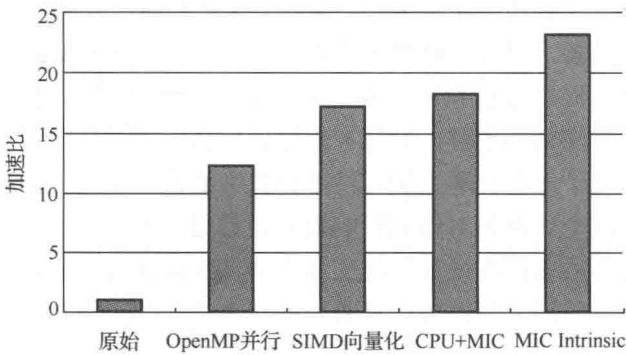


图 14-14 中等规模算例加速比

- (1) 原始指最初的串行程序。
- (2) OpenMP 并行指在 CPU 平台线程级并行后的加速比。
- (3) SIMD 向量化指的是添加 SIMD 指令后的 CPU 程序加速效果。
- (4) CPU+MIC 指的是在 CPU+MIC 协作不使用 C Intrinsic 优化的结果。
- (5) MIC Intrinsic 指的是 MIC 计算部分使用 C Intrinsic 优化的结果。

可以看出,对于小规模算例,MIC 参与运算后性能反而下降,这是因为 MIC 与 CPU 之间的数据传输花了大量时间,MIC 承担计算减少的时间被数据传输时间所掩盖了。

而对于中等规模的算例,MIC 参与运算后只有微小的性能提升,这是因为 MIC 上的计算性能比较差,而 MIC 代码使用 C Intrinsic 优化后,性能则有了十分显著的提升,其原因我们已在讲述并行实现时进行了解释。

14.6.3 跨节点性能

最终决赛平台为天河二号,对于给定的大规模算例,ishot 为 8192,我们使用了 1024 个节点,占总节点的 1/16。每个节点完成 8 个 ishot 的计算,由于不同节点的数据在计算时不存在数据依赖,只有在输出结果时需要汇总,可以知道程序的可扩展性非常好,为线性加速。

14.7 参赛感悟及收获

ASC 比赛是目前世界三大超算比赛之一,今年是上海交通大学队伍第二次参加这种高级别的比赛,从比赛的备战到最终决赛,我们一直以提升自身实力为主要目的,并秉承如下理念。

(1) 通过组织比赛,吸引更多的学生参与到高性能计算领域来,为上海交通大学尽可能多地培养超算人才。

(2) 增加比赛经验,多学习兄弟院校在超算应用优化方面的先进理念。

正是基于这种理念,整个队伍在整个过程都一直保持比较轻松的状态。对我本人来说,这次比赛收获不仅是获得了冠军,更重要的是发现了这么勤奋聪明的 5 名学生,他们对超算领域感兴趣,并且愿意在这个领域花费大量的精力。这 5 名队员对超算都非常有兴趣,比赛准备过程中能够自发积极地了解应用,并尝试优化应用,为了在比赛中获得好的成绩,我们做了非常充分的准备,进行了非常多的调研,尤其在 3D_EW 这个项目,由于要在目前最快的超级计算机天河二号的 MIC 集群上运行,对 MIC 加速卡进行了非常深入的调研,对其体系结构有了非常深刻的理解,同时在程序优化过程中进行了大量的努力,而通过题目的优化,也增强了上海交通大学高性能计算中心在 MIC 众核编程方面的研究水平。对于参赛学生来说,这个比赛也对他们影响很大,通过比赛,将学生吸引到了这个领域,大部分参赛学生在比赛后选择继续在上海交通大学高性能计算中心,并参与到科研项目中,其中一名学生也已经在美国读高性能计算领域的博士。



超算竞赛导引

THE STUDENT SUPERCOMPUTER
CHALLENGE GUIDE

信息技术分社
联系电话: 010-64004332
E-mail: it@mail.sciencep.com

销售分类建议: 计算机技术/超级计算

www.sciencep.com

ISBN 978-7-03-047163-5



9 787030 471635 >

定价: 49.00 元