★CACM 中国版★

# 计算机协会通讯

CACM.ACM.ORG

2015 年 5 月第 58 卷第 5 期

## 最优化——机器人动作中的运动选择原则

讲授语言原理

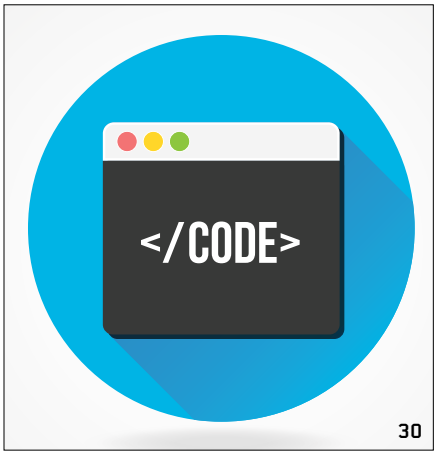
“现在”并不存在

斯诺登之后的隐私行为

编写多核计算机程序

Association for Computing Machinery

## 观点




30 观点
**讲授基础语言原理**
工业界期待迎来更多学习了程序设计语言原理的毕业生。
*Thomas Ball 和 Benjamin Zorn*

## 实践


36 **“现在”是不存在的**
分布式系统的同时性问题。
*Justin Sheehy*

## 投稿文章

48 **斯诺登之后的隐私行为**
尽管媒体进行了持续报道，但公众的隐私行为仍基本没变。
*Sören Preibusch*

## 评论文章

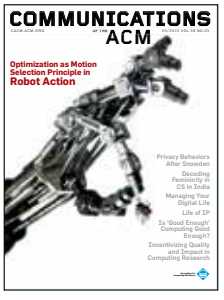

64 **最优化——机器人动作中的运动选择原则**
机器人的动作源于机器人的运动。然而，机器人的动作存在于物理空间，机器人的运动始于电机控制空间。那么，机器人怎么用运动来表达动作呢?
*Jean-Paul Laumond，Nicolas Mansard，*
*Jean Bernard Lasserre*

在本通讯独家视频中，您可以观看作者对本研究的讨论。

## 研究亮点

76 技术视角
**编写多核计算机程序**
*James Larus*

77 **传统编程如何填补并行计算应用程序的忍者性能差距?**
*Nadathur Satish、*
*Changkyu Kim*、Jatin Chhugani*、*
*Hideki Saito、Rakesh Krishnaiyer、*
*Mikhail Smelyanskiy、*
*Milind Girkar 和 Pradeep Dubey*




**关于封面：**
虽然移动是生命系统的基本特征，但是将这种能力融入机器人系统取决于诸多科学领域的实力及专业知识。本期封面故事（第 64 页）展示了最优运动在机器人动作建模中的表达能力。插图由 Peter Crowther Associates 提供

图片由JANE KELLY，TRACI DABERKO，PETER CROWTHER ASSOCIATES提供

Association for Computing Machinery
*Advancing Computing as a Science & Profession*

# ACM 通讯

精选文章中文译文




## ACM中国理事会办公室

中国北京清华大学
东主楼 11-236 室
邮编：100084
电话：+86-10-62785025
电子邮件：acmchina@acm.org
联系人：辛爽

## ACM通讯

(ISSN 0001-0782) 由计算机协会（2 Penn Plaza, Suite 701, New York, NY 10121-0701）按月发行。

Association for Computing Machinery

DOI:10.1145/2663342

Thomas Ball 和 Benjamin Zorn

# 观点
# 讲授基础语言原理

*工业界期待迎来更多学习了程序设计语言原理的毕业生。*

编程人员日益成为紧缺人才，这一问题已得到广泛关注（请查看 www.code.org 网站上的相关内容，或观看 2013 年和 2014 年 12 月计算机科学周期间举办的“编码一小时”(Hour of Code) 课程）。尽管会编程在支持人们发挥创造力方面拥有巨大潜力，但我们应记住，编程语言仍将继续发展，以解决层出不穷的新问题，而*语言原理*总是万变不离其宗。如本观点文章所论述，语言基础在本行业所使用的复杂软件系统的设计与实现中发挥日益重要的作用。工业界亟需更多语言原理方面的人才，为客户提供可靠而高效的软件解决方案。

在过去，为应对复杂系统在设计与实现方面的难题，很多重要语言原理应运而生。John McCarthy 在 1959 年前后为 Lisp 语言提出了垃圾回收机制，而该机制目前广泛应用于 Java 和 C# 等现代编程语言以及 Python 和 JavaScript 等热门脚本语言。[8] Dijkstra 在 *Communications* 上发表了一篇题为《Go To 语句是有害的》(Go To Statement Considered Harmful) 的致读者来信，他在文中提倡使用结构化编程设计，这


被所有现代编程语言奉为圭臬。[5] 类型系统按照程序表达式计算出的值来对其进行分类，[12] 让编译程序可以证明某类错误不存在，更高效地优化编码。Hoare 的断言方法为如何确定程序的正确性提供了框架。[6]

我们希望构建更加复杂的系统，要想以可预见的方式实现可靠的系统，能在更高层次表达编程人员意图的全新机制必不可少。随着新类型的系统不断问世，为了让更多人员能对此类系统编程，就必须设计全新的编程语言。表达编程人员意图的新方法有很多，这些方法以多种形式利用通用语言、特定领域语言和用于验证高层次设计属性的形式规格说明语言中的特性。

近期发生的 OpenSSL TLS 心跳包缓冲区读取泄露（心脏出血漏洞）事件表明，[10] 公司企业和社会团体如果使用无法保护抽象的弱类型系统的旧编程语言（此处为 C 语言）来构建基础架构，就会付出惨重代价。为应对可扩展编程的可靠系统所带来的挑战，多家公司都已开发了全新安全的系统编程语言。例如，Google 的 Go 语言 (http://golang.org/)、Mozilla 的 Rust 语言 (http://www.rust-lang.org/) 以及 Microsoft 的 Sing# 语言 (http://singularity.codeplex.com/)。这些语言通过可为程序执行提供更多安全保障的全新类型系统提高了编程水平。

领域专用语言 (DSL) 进一步提高了编程水平，通过限制表达能力实现了较通用语言更高程度的行为保证。SQL 数据库查询语言是一个经典例子，该语言基于关系代数，[2] 支持复杂的查询优化。

DSL 仍将在行业中派上用场。Google 的 Map/Reduce 数据并行执行模型 [3] 推动了基于 SQL 的 DSL 的大量出现，其中包括 Yahoo 的 Pig (http://pig.apache.org/)。苏黎世联邦理工学院的 Spiral 系统 (http://www.spiral.net/) 利用数学函数和优化规则的声明式规格说明，生成用于数字信号处理的非常高效的平台专用编码。Intel 将 Sipral 生

图片由 JANE KELLY 提供

成的编码加入到 Integrated Performance Primitives 库中提供使用。微软工作人员于近期开发了名为 P 的 DSL，用于异步事件驱动系统的编程设计，[4] 该语言支持使用模型检查来检查设计的响应性，即及时处理每个事件的能力。[1] Windows 8 USB 3.0 设备驱动器堆栈的核心部分采用 P 语言实现和验证。

规格说明语言是另一类重要的语言。当系统和算法设计人员将其设计编码成程序后，设计上的错误便难以发现和修复。如果使用此类语言，他们就能对自己的设计更有把握。最近，AT&T 实验室的 Pamela Zave 揭示了 Chord 分布式哈希表的基础协议存在缺陷 [14]；她以 Alloy 语言对该协议建模，[7] 并使用 Alloy Analyzer 工具展示“在 Chord 论文对故障行为的相同假设条件下，Chord 的所有发布版本都是不正确的。” Emina Torlak 及其同事使用相同的建模方法分析 Java Memory Model (JMM) 的各种规格说明，并对照了 JMM 的已发布测试用例，[13] 揭示了规格说明和测试用例结果存在很多不一致。

我们对此提出了三个方面的建议，以从后向前的方式讨论本观点文章所探讨的三个主题（形式设计语言、领域专用语言以及全新通用编程设计语言）。首先，计算机科学专业的学生大多数都将成为新一代系统的设计者和实现者，他们应掌握逻辑及其在设计形式化方面的应用的基础知识，并且在使用 Alloy 或 TLA+ 等自动化工具创建并调试形式规格说明方面具备一定经验。正如 Leslie Lamport 所说的那样，“对于复杂系统的设计者而言，形式规格说明正如摩天大楼的蓝图一样必不可少。”[9] “形式规格说明”的教学方法、工具和材料即将迎来黄金时期。目前，“契约式设计”等机制已在主流编程设计语言中提供使用，应纳入编程基础课程的教学内容中，卡耐基梅隆大学的编程设计语言序列导论课程就是如此。[11] 学生如果能认识到从原理上思考和相关工具的价值，这些课程会让他们在以后的职业生涯中受益。我们如果不能让本专业的学生了解到形式规格说明的价值，那么我们的计算机科学专业教育就失败了。

**为应对复杂系统在设计与实现方面的难题，很多重要语言原理应运而生。**

其次，无论是计算机科学专业学生，还是非专业学生，只要他们想要成为编程人员，就应尽早接触函数式编程设计语言，以便在声明性编程设计范式方面积累经验。函数式 / 声明性语言抽象的价值是显而易见的，此类语言可以让编程人员事半功倍，并且支持在一系列运行时目标上编译高效编码。我们已经发现，这类抽象已在 DSL 以及 C#、Java、Scala 等指令性语言中成为主流，在 F# 和 Haskell 等现代函数式语言中更是如此。

第三，有志于开发全新编程设计语言的人士应仔细研究类型系统，而 B.C. Pierce 的 *Types and Programming Languages*[12] 是一本不错的入门读物。Microsoft、Google 以及 Mozilla 都在不断加大对强类型系统的系统编程设计语言的投入，这绝非偶然，而是依据我们在构建和部署以弱类型系统语言编写的复杂系统方面积累的数十年经验。为推动行业的发展，我们迫切需要在安全编程设计语言 - 类型系统形式基础领域受过教育的语言设计者。

## 结论

未来的应用程序和系统将日益依赖于原理化和形式化的基于语言的软件开发方法，以便提高编程人员的工作效率以及相关系统的可靠性。软件开发人员必须扎实地掌握语言原理才能在今后大有可为，加强对计算机科学的语言原理的教学是必不可少的。读者如果对编程设计语言的教学感兴趣，请参阅 SIGPLAN 教育委员会的著述 (http://wp.acm.org/sigplaneducationboard/)，该委员会彻头彻尾地改写了“编程设计语言”的“知识领域”，相关内容包含在《2013 年计算机科学课程报告》(http://cs2013.org) 的首版公开初稿中。


参考资料

1. Clarke, E.M., Grumberg, O., and Peled, D. *Model Checking*.MIT Press, 2001, I–XIV, 1–314.
2. Codd, E.F.A relational model of data for large shared data banks.*Commun.ACM 13*, 6 (June 1970), 377–387.
3. Dean, J. and Ghemawat, S. MapReduce:A flexible data processing tool.*Commun.ACM 53*, 1 (Jan. 2010), 72–77.
4. Desai, A. et al.P:Safe asynchronous event-driven programming.In *Proceedings of the 2013 ACM SIGPLAN Conference on Programming Language Design and Implementation*, 2013.
5. Dijkstra, E.W.Letters to the editor:Go To statement considered harmful.*Commun.ACM 11*, 13 (Mar. 1968), 147–148.
6. Hoare, C.A.R.An axiomatic basis for computer programming.*Commun.ACM 12*, vol. 10 (Oct. 1969), 576–580.
7. Jackson, D. *Software Abstractions:Logic, Language, and Analysis*. MIT Press, Cambridge, MA, 2012.
8. Jones, R. *The Garbage Collection Handbook:The Art of Automatic Memory Management*. Chapman and Hall, 2012.
9. Lamport, L. Why we should build software like we build houses.*Wired 25* (Jan. 2013).
10. OpenSSL Project.OpenSSL Security Advisory [07 Apr 2014].(Apr. 7, 2014); http://www.openssl.org/news/secadv_20140407.txt.
11. Pfenning, F. Specification and verification in introductory computer science.Carnegie Mellon University; http://c0.typesafety.net/.
12. Pierce, B.C. *Types and Programming Languages*.MIT Press, 2002, I–XXI, 1–623.
13. Torlak, E., Vaziri, M. and Dolby, J. MemSAT:Checking axiomatic specifications of memory models.In *Proceedings of the 2010 ACM SIGPLAN Conference on Programming Language Design and Implementation*, 2010.
14. Zave, P. Using lightweight modeling to understand Chord.*SIGCOMM Comput.Commun.Rev. 42* (Mar. 2012), 49–57.



Thomas Ball (tball@microsoft.com) 是微软研究院软件工程研究组 (RiSE) 的首席研究员和共同经理人，该研究院位于华盛顿市雷德蒙德。

Benjamin Zorn (zorn@microsoft.com) 是微软研究院软件工程研究组 (RiSE) 的首席研究员和共同经理人，该研究院位于华盛顿市雷德蒙德。


译文责任编辑：陈文光

DOI:10/1145/2733108

文章编写主导者 acmqueue
queue.acm.org

## 分布式系统的同时性问题

作者：JUSTIN SHEEHY

# “现在”并不存在

“现在。”

我撰写本文之日与您阅读本文之时至少相隔数周时间。我们对这种延迟不以为然，这在书面媒体中更不值得一提。

“现在。”

假如我们在同一个房间，当我大声讲话时，您可能会对即时性更有感触。如果凭直觉判断，您可能会认为，在我说话的同时您就能立即听到声音。但这种直觉是错误的。如果您考虑声音的特性，而非仅凭直觉判断，您就能知道我说话与您听到声音之间必须间隔一段时间。我的声音靠空气运动来传递，声音从我的嘴里传递到您的耳朵里要花费一定时间。

“现在。”

如果将写有文字的标牌举起来，即使我们同时向标牌看去，我们也不会同时看到该图像，因为标牌信息靠光来传播，光到达我们所用的时间是不同的。

尽管计算机的某些组成部分是虚拟的，但计算机仍在物理世界中运行，并且无法回避物理世界中的难题。Rear Admiral Grace Hopper 是计算机领域最重要的领军人物之一，其成就包括创建了首个编译器。为阐释上述观点，他曾发给他的每位学生一根长 11.8 英寸的电线，该长度也是电在每纳秒移动的最大距离。这种物理表示形式体现信息、时间和距离之间的关系，可以用来阐述信号（就像标牌的例子）到达目的为何必然需要一定时间。由于存在这些延迟，我们很难弄清楚计算机系统中的“现在”到底意味着什么。

从理论上来讲，如果我们提前仔细计划，没有什么可以妨碍我们就“现在”的常用含义达成共识。（相对论尽管是充满诱惑的干扰，但不是此处的问题。人类当前的所有计算系统共用一个足够封闭的参照系，使得时间知觉上的相对论差别无关紧要。）网络时间协议 (NTP)[14] 用来同步互联网上各系统的时钟，其工作原理包括计算在不同主机之间传送消息所需的时间。当传送时间确定以后，主机就能在调整时钟时对其解析，以匹配更具权威性的源。通过在网络中提供一些非常精确的源（基于原子辐射持续测量的时钟），我们就能使用 NTP 来同步计算机的时钟，并将误差控制在较小范围内。组成全球 GPS 中的每颗卫星都包含数个这种原子钟，所以单个原子钟发生故障不会导致

插图由 TRACI DABERKO 提供


NOW
NOWNOWNOWNOWNOWN
WNOWNOWNOWNOW
NOW

卫星报废。GPS 协议支持任何携带接收器的人员确定接收器位置并获得精确的时间，但前提是接收器能够发现数量足够的卫星以解决所有变量。

我们了解这些协议已有几十年之久，我们很容易就会认为自己已经克服这类问题，并且应该能够构建可以假定时钟是同步的系统。原子钟、NTP 以及 GPS 卫星提供了解析信息传送时间所需的知识和设备。因此，任何地方的系统都应能够就“现在”达成一致，并以同一个通用的视角来看待时间进程。这样一来，网络和计算中的各类困难问题就更容易解决了。如果您关心的所有系统都使用完全相同的时间知觉，那么很多问题都更易于管控和解决，即使一些相关主机发生故障时也不例外。然而，这些问题仍然存在，如何解决这类问题将继续成为热门研究领域，也将成为构建实用分布式系统时的主要考虑因素。

如果您查看了用于理解时间的成熟机制，也许会觉得研究人员和系统构建人员做了大量无用功。既然我们知道如何同步时钟，为什么还要假设时钟存在误差，并在该条件下尝试解决问题？为什么不使用恰当的时间源和协议组合，使时钟达到一致并略过这些问题？这种质疑是站不住脚的，这类问题不仅重要，而且还需直接面对，因为任何事物都会出现故障。

真正的问题并不在于信息在各地点间传送需要时间这一理论观点，而是在计算系统驻留的物理世界中，计算系统的组件经常出现故障。构建系统时的最常见误区就是假设我们可以脱离基本的物理事实，这一点尤以基于商品机器和网络的分布式计算系统为甚，但也绝不仅限于此。

光速就是一个基本的物理事实，但一个更为致命且同样普遍适用的事实就是我们无法制造永远不会出现故障的完美机器。不同时性和局部故障的现实情况相结合，使构建分布式系统变得非常困难。如果我们没有周密的计划，不解析单个组件的故障原因，那么我们的组合系统注定会发生故障。

不可能性结论是分布式系统理论中最著名的结论之一，揭示了构建在事物可能出现故障的环境中工作的系统时受到的一项限制。该结论通常称为 FLP 结论，是以作者 Fischer、Lynch 和 Paterson 而命名的。[8] 他们的著述作为分布式计算领域最具影响力的论文而获得了 2001 年 Dijksatr 奖，他们论证了一些计算问题可以在同步模型中（主机使用相同时钟或共用时钟）解决，却无法在较弱的异步系统模型中解决。这类不可能性结论非常重要，可以指导您在设计系统时避开一些死路。这些结论也可以用作试金石，如果某些产品言过其实，那么您可以为自己的怀疑找到依据。

CAP 定理是与之相关的另一个结论，更为当今的开发人员所熟知。CAP 是 consistency（一致性）、availability（可用性）和 partition tolerance（分区容忍性）的开头字母缩写。CAP 定理最早由 Eric Brewer[5] 非正式提出，其正式版本随后由 Seth Gilbert 和 Nancy Lynch 论证。[9] 从分布式系统理论的观点来看，CAP 定理并没有 FLP 有趣：一个推翻 CAP 定理正式版本的反例假设世界上存在比 FLP 更弱且更具对抗性的模型，并且要求在该模型中实现更多目标。尽管二者并不互为子问题，但 FLP 是更强大、更有趣、或许更令人意想不到的结论。已经熟悉 FLP 的研究人员可能会觉得 CAP 观点有些无聊。

对于那些尚未涉足分布式系统领域的人士，CAP 更加易用易懂，这也许就是 CAP 存在的价值，这种想法似乎更为合理。如此看来，CAP 是值得称赞的。但在过去几年中，相关文章和博文大多误读了 CAP 的基本观点。令人遗憾的是，CAP 并没有为当今开发人员提供一种简单途径，使其接受分布式和非完美环境中的编程设计的现实情况。无论是从 CAP、FLP 还是其他结论的观点出发，这种现实情况就是：在当前环境下，必须假设用作构建模块的组件存在缺陷。（CAP 或 FLP 等任何理论上的“不可能结论”与相应系统模型存在内在联系。结论依赖于环境的理论模型。此类结论实际上要表达的是有些目标（如一致性）在特定模型中无法达到，而不是在一般情况下无法达到。这些结论对于研究哪些路径是死路的从业人员极其有用，但是如果只学习结论而忽视了结论的适用环境，就会受到误导。）

真正的问题在于事物会出现故障。本文中所提及的内容（如 FLP）都是以处理组件可能发生故障的系统为前提的。如果这是问题症结所在，那么我们为何不使用不会出现故障的事物呢，然后使用假设可以保持健康状态的组件来构建更好的系统？

在过去的几年中，人们常常援引 Google 的 Spanner 系统来证明此类假设是合理的。[6] 该系统正是使用上述技术（NTP、GPS 和原子钟）来协调 Spanner 包含的主机的时钟，对各时钟之间的误差和不确定性进行测量并将其降至最低（但无法消除）。Spanner 白皮书及其记录的系统常被用来支撑以下论断：构建单一时间观的分布式系统是可能的。

尽管提及 Google 并以其权威性作为论据具有一定吸引力，但该论断仍是错误的。用 Spanner 来证明同步问题已解决的人要么是在说谎，要么没有读过白皮书。驳斥该论断最简单最有力的证据就是 Spanner 白皮书本身。Spanner 的

TrueTime 组件并不提供简单且统一的共享时间线。相反，由该系统明确提供的 API 直接暴露了各系统时钟之间的知觉时间存在误差。如果向该系统询问当前时间，它不会提供唯一的值，而是以一对数值来描述“现在”可能处在的时间范围。也就是说，TrueTime 的表现与解决该基本问题是相悖的。要有勇气和魄力才能选择直面问题并正视不确定性，相比于假装以一个绝对值来代表“现在”，前一种做法对于工作分布式系统更有意义。

在 Spanner 的生产环境中，任何时刻的时钟漂移一般在 1 到 7 毫秒之间。这是 Google 所能到达的最佳成绩，并且已将旨在最小化偏斜的措施考虑在内，如加入 GPS、原子钟纠正效果，驱逐漂移最严重的时钟等。典型的 x86 时钟的速度会受到各类不可预见的环境因素的影响，比如负载、温度和能源。就连同一机架顶部和底部之间的差异都会导致偏斜不同。如果 Spanner 等极其昂贵的环境的最佳效果就是保持几毫秒的不确定性，那么我们大多都会认为自己的时钟与之望尘莫及。

有人认为在设计分布式系统时采用“假装一切正常”的方法是合理的，他们的依据是以下说法，质量足够好的设备不会发生故障，至少发生故障的几率极小，因此不必对此担心。这种错误的论断如果由设备制造商提出，这是情有可原的。但该论断往往会出自此类设备的用户（如分布式数据库软件的设计者）之口。（GPS 设计者当然不会赞同这种说法，Spanner 和其他系统的都要依靠 GPS 设计者。GPS 卫星的数量已经达到近 30 颗，系统只需要发现 4 颗 GPS 卫星就可以工作。每颗 GPS 卫星都装有多个冗余的原子钟，以便在某个原子钟发生故障时可以继续工作。）

**构建分布式系统时的最常见误区就是假设我们可以脱离基本的物理事实。**

由该论断衍变而来的最常见说法就是，在本地数据中心网络的环境中，“网络是可靠的”。在网络表现不佳时，很多系统都会发生未定义行为，可能会造成灾难性后果。推销这些系统的人员认为这种情况是可以忽略不计的，他们的理由是此类故障极少出现。他们对客户造成了双重危害。其一，即使故障事件发生几率微乎其微，但可曾设想过，一旦发生此类事件，系统会如何运行，这类事件会对业务造成哪些影响，又该如何做好应对准备？其二，这种说法无异于谎言，这是最糟糕的情况。这种厚颜无耻的谎言其实是分布式计算八个经典谬论中的第一个谬论。[7,15] 有关此类故障的真实情况在之前的 ACM *Queue* 文章中被详细记录，[3] 相关证据是十分明确的，如果有人不带讽刺意味地声称“网络是可靠的”，并以此作为软件设计选择依据，那么根本不能相信这种人可以构建任何计算系统。有些系统和网络发生故障的方式可能不为特定观察者所察觉，但将系统的安全性完全建立在系统不会发生故障这一假设的基础上，这种做法仍是愚蠢至极。

在对待这类物理问题时，人们还会采取一种相反的做法，就是假设任何事物都是不可靠的，只能使用极具对抗性的环境的形式化模型进行设计。验证 FLP 的“异步”模型并不是用来构建工作系统的最具对抗性的模型，但它无疑比大多数开发人员的系统运行环境更加不利。其思路是，如果模型所在环境比它的运行环境更恶劣，那么可以在该模型中做到的事情在真实的实现环境中也应该能够做到。（请注意，分布式系统理论家创建了一些难以应对的模型，比如加入拜占庭故障可能性的模型，其中的系统可能会出现比崩溃或延迟更为严重的故障。要想了解真正的对抗性网络 / 系统模型，您可以查看加密协

议理论家使用的符号模型或象征模型。在这类模型中，系统构建人员确实会假设您的网络出现故障。）

假设一个比真实运行环境更加苛刻的环境，很多著名的一致性协议和协调协议都是在这种环境中设计出来的，例如 Paxos。[12] 对于分布式系统的基本构建模块，了解它们如何在不利于设计人员的环境中提供保障非常有用，在这类环境中，消息丢失、主机崩溃等事件都会随意发生。（例如，Paxos 和相关协议最注重的是“安全性”这一属性，即确保不同参与主机不会做出相互矛盾的决定。）另一个此类工作领域是逻辑时间，表示为向量时钟、版本向量以及事件顺序的其他抽象方式。这种观点承认了以下事实，在一个时钟完全不可靠的环境中，无法假设时钟同步并建立顺序的观念。

当然，应努力确保一切事物都是尽可能可靠的，比如网络。将一成不变的目标与可实现的完美最终状态相混淆是愚蠢的。同样愚蠢的还有单纯认为只有最具说服力、易于理解的理论模型才是构建系统的合理出发点。很多最高效的分布式系统所基于的有价值元素都无法完美地映射到分布式计算的常见模型。TCP 就是典型的这类构建模块。这种普遍使用的协议提供了一些非常有用的属性，其中的真实集合并未精确地映射到用于确立理论结论（如 FLP）的典型网络模型。这令系统构建人员陷入两难境地：在设计时不假设 TCP 等协议真实存在是愚蠢的做法，但在有些情况下，如果他们试图了解分布式系统理论如何运用到工作中，就会无法找到充足的依据。

Zab 协议是著名的 Apache ZooKeeper 协调软件的重要组成部分，也是展示对中间路线的尝试的最好例子。[10] ZooKeeper 的作者了解 Paxos 等现有一致性协议，但仍然决定设计自己的协议，并添加一些额外功能，比如支持一次性处理多个请求，不必等待一项方案完成才开始下一项方案。他们意识到，如果以 TCP 作为基础，就能使用相应的底层系统，获得可在他们的协议中继续利用的重要属性。例如，在单连接 TCP 嵌套中，发送者先生成消息 A，然后生成消息 B，如果接收者读取了消息 B，那么就可以认定接收者在此之前已经读取了消息 A。这种“前缀”属性非常有用，并且是异步模型所不具备的。有关这项优势的具体案例，请了解该领域的研究成果以及特定的构建技术，请不要忽略二者中的任何一个部分。

如果想要务实，必须防范实用主义成为自己草率工作的借口。ZooKeeper 中所实现的 Zab 协议是标准的参考实现，并没有完全遵循 Zab 协议的设计。[13] 您可能自称为“实用主义者”，但大多数其他软件也不与形式规格说明完全一致，因此您可能会认为这并不是值得担心的异常情况。您的错误原因有两个：首先，ZooKeeper 和类似软件的作用不同于其他软件，前者真正提供了必要的安全属性，为其他系统做出强大假设奠定了基础。其次，如果此类协议的安全属性发生问题，那么这些问题（可能很危险）的隐蔽性较强，很容易被忽视。如果用户不坚信系统的实现反映了已被分析好的协议，那么他就没有理由信任系统。系统的“足够正确性”可以从受欢迎程度上判断，如果普通用户无法评估正确性，那么这种说法就是一派胡言。

这些内容并不是在批判 ZooKeeper。实际上，ZooKeeper 是同类产品中质量最好的开源软件之一，众多优秀工程师都在不断对其进行改进。根据最近的分析，ZooKeeper 在监控下的表现好于竞争对手。[1] 我以 Zookeeper 为例，从理论和实践两个方面，展示了采取中间路线的必要性和缺陷。将理论映射到实践是极具挑战性的。

混合逻辑时钟是采用中间路线的另一个例子，[11] 此类时钟将通用逻辑时间技术与物理时钟的时间戳集于一身。用户可以通过查看同一个整体系统中的各个物理时钟来应用一些有趣的技术（不完美，但仍然有用）。正如 Spanner 一样，该方法并不假装创建一个统一时间线，而是允许系统设计人员将有关跨集群感知和传递时间的最佳可用知识作为构建依据。

包括 Paxos、View-stamped Replication、Zab/Zookeeper 以及 Raft 在内的各类协调系统都提供了定义分布式系统中的事件顺序的方法，尽管物理时间不能稳妥地用于该目的。这些协议肯定可以用于该目的：在系统中提供统一的时间线。您可以将协调看作为“现在”提供逻辑代理。如果将协议用于此用途，协议会造成一定代价，这是由它们的相同点所造成的：持续通信。例如，如果对分布式系统中发生的所有事情的顺序进行协调，使响应延迟不少于系统中的往返时间（发送两个连续消息用时）是最好的效果。

根据协调系统的详细信息，吞吐量也可能存在边界。性能要求苛求的系统的设计人员可能希望正确做事，却发现他们无法承受持续协调的过高成本。当主机或网络可能频繁出现故障时，这种情况更为严重，很多协调系统只提供“最终活性”，可以在困难最小的时候取得进展。在极少数情况下，当一切都完美工作时，持续协调的通信成本可能过于高昂。

在很多计算领域，如 CPU 体系结构、多线程编程设计以及 DBMS 设计，放弃严格协调来换取性能的做法都是广为人知的折中手段。很多时候，这已经变成一场游戏，目标是找出可向用户提供正常行为时

所需的最小协调。一些并发本地系统的设计人员发明了一系列管理足够协调的技巧，令您无法想象（例如，RDBMS 领域很早之前就利用有趣的性能入侵，这往往会导致 ACID 严重不足 ( 原子性、一致性、隔离性、持久性 )）[2]，对适用于通用分布式技术的此类技术的探索才刚刚开始。

这是一个激动人心的时代，如何在分布式系统中权衡这些利弊，人们现在才开始认真研究这一领域。该话题得到了加州大学伯克利分校 的 Berkeley Orders of Magnitude (BOOM) 团队的重视，该团队中的多名研究院都采取互不相同但存在关联的方法，试图了解如何才能明智地权衡各种利弊。[4] 他们开辟了新的领域，研究可以完全不考虑“现在”的条件和方式，从而免于承担协调的成本。这类研究工作可能很快就会让更多人了解到同步时间对于开展工作是不必要的。如果分布式系统可以建立在更少的协调的基础上，同时还能提供所需的一切安全属性，那么这类系统在速度、弹性和扩展能力方面都会得到改善。

另一个重要的研究领域是如何避免对无冲突复制数据类型 (CRDT) 所涉及的时间或协调感到担忧。这是一种从来不需要按顺序进行更新的数据结构，因此无需处理时间问题就可以安全地使用。由 CRDT 提供的安全性有时被称作“强最终一致性”：系统中的所有主机不分先后次序，都会收到同一组更新，并进入相同状态。人们很早以前就知道这是可行的，前提是要开展的所有工作都具备特定属性，如交换性、结合性、幂等性。CRDT 令人期待之处在于该领域的研究人员 [16] 正不断扩大我们的认识，即我们可以在限制中发挥表现力，我们能以低成本开展工作，同时提供一套丰富的数据类型，可供开发人员立即使用。

这些研究领域尚处于起步阶段，从以下情况就可以判断出来，更倾向于临时入侵的主流分布式系统得以存在，而处理一致性和协调问题著名选项却被取代。例如，一些分布式数据库采用“最后写入获胜”的策略来解决相互冲突的写入。“现在”并不是一个横跨整个系统的简单值。同理，“最后”本身是一个毫无意义的词语。这个策略实际上是“以不可预测方式选择的多个写入将丢失”，但这种观念不被多数数据库所接受，难道不是吗？科学技术水平仍在迅速提高，任何人都应能做得比这更好。

又如，选择通过临时协调代码来解决群集管理，而不使用正式确立且经过分析验证的一致性协议，这种做法与“大多数写入丢失”一样糟糕。在处理可由知名一致性协议解决的问题时，如果您确实需要采用其他方式，（提示：请勿这样做）那么您至少应该效仿 ZooKeeper/Zab 实现人员的做法，并清楚地记录您的目标和假设。这虽然不会避免灾难发生，但至少有助于考古学家稍后检查灾后现场。

这是一个令分布式系统的构建人员感到兴奋的时代。我们还将迎来很多变化。然而，一些真理仍然会一成不变。认为“现在”是一个横跨一段距离的有意义的简单值，这种想法依旧不妥。我们需要继续开展相关研究，积累更多实践经验，以应对现实问题。我们会做得更好。我们现在就可以行动起来。

**鸣谢**

感谢提供宝贵意见和建议的人士，包 括 Peter Bailis、Christopher Meiklejohn、Steve Vinoski、Kyle Kingsbury 以及 Terry Coatta。

**queue.acm.org 上的相关文章**

If You Have Too Much Data, then “Good Enough” Is Good Enough
*Pat Helland*
http://queue.acm.org/detail.cfm?id=1988603

There’s Just No Getting around It:You’re Building a Distributed System
*Mark Cavage*
http://queue.acm.org/detail.cfm?id=2482856

The Network is Reliable
*Peter Bailis and Kyle Kingsbury*
http://queue.acm.org/detail.cfm?id=2655736

**参考资料**

1. Aphyr.Call me maybe:ZooKeeper, 2013; http://aphyr.com/posts/291-call-me-maybe-zookeeper.
2. Bailis, P. When is “ACID” ACID?Rarely, 2013; http://www.bailis.org/blog/when-is-acid-acid-rarely/.
3. Bailis, P. and Kingsbury, K. The network is reliable. ACM *Queue 12*, 7 (2014); http://queue.acm.org/detail.cfm?id=2655736.
4. BOOM; http://boom.cs.berkeley.edu/papers.html.
5. Brewer, E.A.Towards robust distributed systems, 2000; http://www.cs.berkeley.edu/~brewer/cs262b-2004/PODC-keynote.pdf.
6. Corbett, J.C. et al.Spanner:Google’s globally distributed database.In *Proceedings of the 10th Symposium on Operating System Design and Implementation*, 2012; http://research.google.com/archive/spanner.html; http://static.googleusercontent.com/media/research.google.com/en/us/archive/spanner-osdi2012.pdf.
7. Deutsch, P. Eight fallacies of distributed computing; https://blogs.oracle.com/jag/resource/Fallacies.html.
8. Fischer, M.J., Lynch, N.A. and Paterson, M.S.Impossibility of distributed consensus with one faulty process.*JACM 32*, 2 (1985), 374–382; http://macs.citadel.edu/rudolphg/csci604/ImpossibilityofConsensus.pdf.
9. Gilbert, S. and Lynch, N. Brewer’s conjecture and the feasibility of consistent, available, partition-tolerant Web services.*ACM SIGACT News 33*, 2 (2002), 51–59; http://webpages.cs.luc.edu/~pld/353/gilbert_lynch_brewer_proof.pdf.
10. Junqueira, F.P., Reed, B.C. and Serafini, M. Zab:High-performance broadcast for primary backup systems, 2011; http://web.stanford.edu/class/cs347/reading/zab.pdf.
11. Kulkarni, S., Demirbas, M., Madeppa, D., Bharadwaj, A. and Leone, M. Logical physical clocks and consistent snapshots in globally distributed databases; http://www.cse.buffalo.edu/tech-reports/2014-04.pdf.
12. Lamport, L. The part-time parliament.*ACM Trans. Computer Systems 16*, 2 (1998), 133–169; http://research.microsoft.com/en-us/um/people/lamport/pubs/pubs.html#lamport-paxos.
13. Medeiros, A. ZooKeeper’s atomic broadcast protocol:Theory and practice; http://www.tcs.hut.fi/Studies/T-79.5001/reports/2012-deSouzaMedeiros.pdf.
14. NTP; http://www.ntp.org.
15. Rotem-Gal-Oz, A. Fallacies of distributed computing explained; http://www.rgoarchitects.com/Files/fallacies.pdf.
16. SyncFree; https://syncfree.lip6.fr/index.php/publications.

Justin Sheehy (@justinsheehy) 是 VMware 的 Cambridge MA 研发中心的现场主管，也是该公司的存储和可用性业务的设计师，负责决策制定。在加入 VMware 之前，他曾先后担任 Basho 的 CTO、MITRE 的首席科学家以及 Akamai 的高级设计师。

译文责任编辑：陈海波

DOI:10.1145/2663341

**尽管媒体进行了持续报道，但公众的隐私行为仍基本没变。**

SÖREN PREIBUSCH

# 斯诺登之后的隐私行为

**2013 年 6 月 NSA（美国国家安全局）承包商雇员爱德华 J. 斯诺登（EDWARD J. SNOWDEN）曝光棱镜（PRISM）计划后，大众媒体**报道了美国在全球范围内的监听情况。广泛持续的媒体报道使得这次揭秘事件成为一次天然的实验机会。在 2013 年 5 月到 2014 年 1 月的纵向研究中，我调查了揭秘事件对美国用户网络行为的即时和长期影响。我的研究包含了用户对个人隐私采取的自我保护行为，以及能体现用户对隐私的关注度的行为。PRISM（棱镜）事件的揭露激发了用户对隐私的关注，然而随后虽然媒体有持续报道，这种关注度又回到了甚至低于揭露之前的水平。我发现隐私增强技术（比如匿名代理）的用户基数并没有出现持续增长。而与体育和八卦等爆炸性新闻相比，PRISM 在世界范围内的公开披露对个人关注度的影响较小。我的研究结果挑战了以下假设，——即在出现重大的隐私事件之后，网络用户会开始更多地关心他们的隐私媒体对政府监听的持续报道与公众迅速消失的关注形成了鲜明的对比。

2013 年 6 月 6 日以后，虽然媒体报道相当全面而且十分稳定，但是公众仍然不知道 PRISM 和相关网络情报计划（包括 Tempora 和 XKeyScore）的很多细节。[8,5] 自 2009 年以来，美国的全国性报纸 *USA Today* 首次在头版报道了政府的监听情况，[7] 同时《*华盛顿邮报*》2013 年也首次在头版的文章中报道了侵犯隐私的情况，[16] 这些报道引起了世界范围内包括隐私研究专家，以及政客和隐私保护倡导者之间的持续讨论。

大众媒体可以通过增加某个话题的报道来设定公众议题。[12] 然而，人们基本不知道媒体对隐私的聚焦如何引导公众观点和行为，让他们更为关注隐私。之前对媒体的影响的分析，其缺陷在于它们对隐私关注度的度量是不可靠的、临时的、和零散的 [17]，一些分析甚至简单地忽略了公众的反应。[21] PRISM 的揭露，使得政府 / 公民和公司 / 客户这两对关系存在的隐私差别变得模糊了，因为很多公司被揭露出恰好是政府监听的帮凶。一些公民做出了反应，调整了他们对于基于网络的服务的消费。

2013 年，对于政府和企业数据处理，德国互联网用户的不信任度较 2011 年高了九个百分点 [1]，但是只有少部分人报告说他们改变

## » 重要见解

- **2013 年 6 月，Snowden（斯诺登）揭露了世界范围内的政府监听情况，这是关于隐私的令人震惊的事情，也为隐私行为研究提供了一次很自然的实验机会。**
- **尽管媒体进行了长期的深度报道，但在公众对隐私的关注和行为方面，它的影响有限又短暂。**
- **从多个源头获取的高精度数据说明，除了记者和学术研究人员之间的论战之外，当前时代中最大的隐私灾难只对网络用户产生了相当小的影响。**

插图由 GLUEKIT 提供，斯诺登的照片由 AGE SKIDMORE/FLICKR 提供

了他们管理个人数据的方式。[2] 谷歌趋势（Google Trends ）表明，发出那些有可能“惹上”美国政府的搜索请求的用户倾向有一个激冷效应，该效应因国家不同而不同。[11] 然而，这些通过专项调查获得的公众观点快照，并不能捕捉到消费者多变易逝的反应。年度报告则只能提供粗略的时间精度，而要衡量 PRISM 揭露事件的效果，却需要连续的行为记录。隐私问卷调查也会有其缺陷，比如因为调查对象缺乏责任心而导致的后设理性和符合社会期望、但不可靠的回答。

收集用户行为方面的数据相当难，[4] 而且公司极少披露关于用户使用其网络服务的统计数据。网络搜索引擎 DuckDuckGo（它宣传它拥有超强的隐私保护能力）曾宣称 PRISM 的揭露，增加它的了日查询量，但是它并没有将用户数量的增加考虑进去。[3,20]

在以上几种调查方式以外，PRISM 事件提供了一次对隐私行为进行互联网规模的纵向研究的机会，本文通过使用多个原始数据源，首次实现了高精度的纵向研究分析。 我结合了多个原始数据源的指标来探讨消费者隐私行为的演变。隐私自我保护行为可以直接观察得到，而用户对隐私的关注度则可以解释为用户对相关信息的搜索活动。

2013 年 6 月 6 日，主流媒体首次报道了 NSA 的国家监听计划的详细情况，后面将其称为“PRISM 日”；同时我将之前的三周定义为参照期。 通过对几个关键指标的每日测量，我分析了随后的 6 到 30 周内，每周的即时扩展反应。几个关键指标包括：

*信息搜索量*，包括 Microsoft.com 上公布的隐私策略的访问量（微软是早期媒体报道中提到的公司之一）、与隐私相关的维基百科页面的访问量、以及与隐私相关的其他普通页面的访问量；

*增强隐私的配置*，包括 IE 浏览器中的隐私设置、Tor 匿名网络和火狐的 Anonymox 插件这两种隐藏连接机制的使用；以及

*相关新网页*，作为一个控制变量，它指以新创建的网页数来表示的媒体报道，这些新网页的内容与政府监听和否认 PRISM 的存在（最初的报道之后）这一话题有关。

**媒体对政府监听的持续报道与公众迅速消失的关注形成了鲜明的对比。**

## 方法论

我分析了使用美国英语的消费者的用户行为。美国是 PRISM 和相关计划的发源地，也是第一篇相关媒体报道出现的地方。这显然是我的研究的局限之处，但这是我刻意为之。未来的研究将会调查不同国家之间的区别，但是在本文中，我避免了语言和文化方面的混杂因素。网络搜索量、企业隐私策略和维基百科页面的访问量、以及 Tor/Anonymox 的使用量等方面的数据，可以从美国获得的数据范围最广，因此我考虑了美国境内的所有网络用户，而不是取了其中的一个样本。作为一个例外，在普通的页面访问数据和 Internet Explorer 的隐私设置方面，由于缺少国家信息，我调查了全球范围内的用户样本。因此，进行任何比较时，我都小心谨慎。

通过人工选择得到与 PRISM 相关的关键词，然后我记录了这些关键词的网络搜索行为（见图 1），然后我把它们与从查询重写中推断出的隐私相关关键词进行交叉检测。查询重写是研究网络搜索行为的标准技术。我更倾向于半人工选择的方法，而不是绝对的回溯数据驱动方法，因其会忽略后斯诺登时期公众对于隐私感知的变化。 美国用户的计数来源于在微软搜索引擎——必应中校正拼写的查询的总数（例如，“privcay”也计算在内）。在任何一年，查询词的流行度都会自然地发生波动；例如，在参照期，词“Wikipedia”的日搜索量波动达 33%，但是它的相对份额只波动了 0.01 个百分点。为了处理总搜索量的波动（即工作日 / 周末的差异）、普通流行度方面的差异以及用户之间不同的搜索模式，我记录

了发起给定查询的用户与所有用户的比例，把这一比例的变化作为参照期的指标。我使用时长为一周的时间箱来平均工作日/周末的影响，这样一来，当比较不同事件的反应时，不必对齐开始的日期。

微软的四个主要部分——Bing、Microsoft.com、Xbox 和其他微软产品（合起来占了 99% 以上的页面浏览量）中均发布了微软的隐私声明 http://www.microsoft.com/privacystatement/，我测量了该隐私声明的页面访问量。我还统计了对 PRISM 相关主题页面的普通浏览量。如果该页面的 URL 或标题包含了关键词，我就把该页面的访问量计算在内。我收集的数据来源于同意共享浏览历史的用户样本，这可能会导致结果偏向于不太关注隐私的用户。

Tor 用户定期刷新他们的运行中继列表。在估算 Tor 用户的数量时，这些发往目录镜像的请求相当有用。[23] 与其他的分析一样，我的重点放在美国用户上，在参照期内，美国用户占据了最大的份额（18%）。为了方便地进行匿名浏览，Anonymox 用户安装了火狐插件。[15]Anonymox 是美国排名前 10 的“隐私和安全”插件之一，也是唯一一家宣传自己可以避免政府威胁隐私的插件。通过每天更新的 ping 列表，我统计了美国活跃用户的数量。

通过统计包含三个关键词的在线文件的数量，我评估了媒体报道的情况。这三个关键词是：斯诺登，棱镜和监听（Snowden, PRISM, surveillance）。必应维护了网络索引的一个子集。通过这个子集，我发现了一些东西 [13]，其中的数量是相对于所有的新文件而言的。”隐私（privacy）“这个词产生了假阳而被排除在外，因为现在很多网站（由法律规定）必须包含一个指向其“隐私”策略或声明的链接。

## 揭露 PRISM 后的网络搜索行为

网络搜索的查询提供了一个探究用户行为的透镜，[4] 可以作为了解用户兴趣的一个入口。例如，流感趋势分析使用网络搜索指标作为医疗状况和行为的预报。[6] 流感趋势模型被指出有不准确之处，这是由于正反馈还引起的，[9] 但这个问题并不存在于我的分析方法中。通过发出的相关查询的数量，我测量了人们对 PRISM 相关话题的关注度。

基于所有相关的和自动生成的隐私相关的关键词，我计算出了搜索行为的综合指标，结果表明，PRISM 事件对人们对隐私的关注程度既没有即时的影响（$p = 0.14$），也没有长期趋势上的影响（$p = 0.84$，线性回归上的 F 检验）。

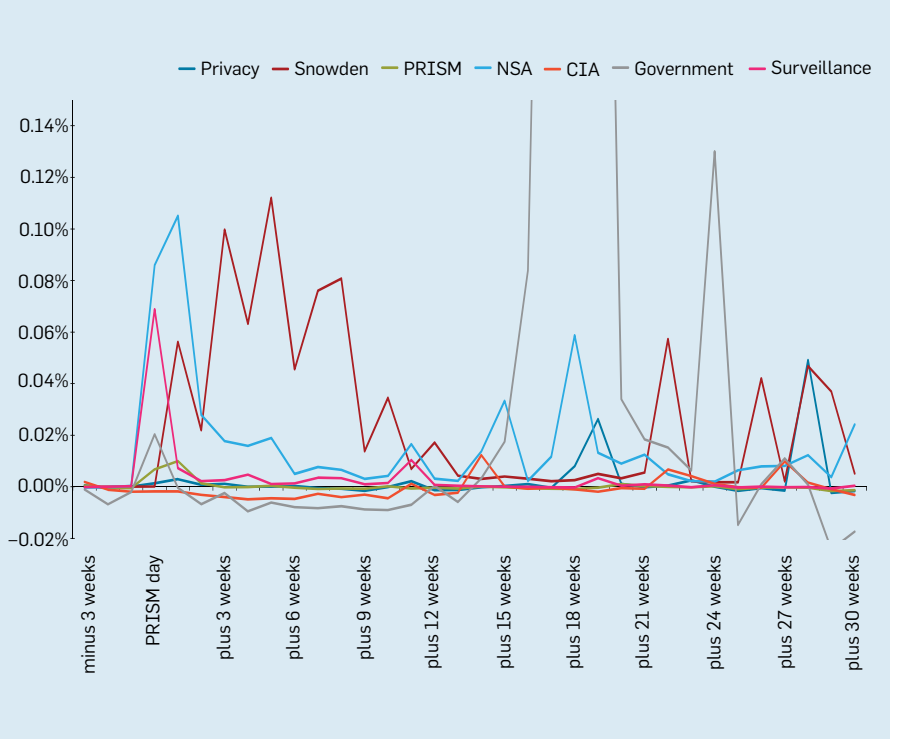
**短期演变。**从单个关键词考虑，在 2013 年 6 月 6 日 PRISM 日，NSA、监听和政府的网络搜索查询出现了小幅增加（所有搜索量增加了 +0.08% 到 +0.02% 个百分点）。两天后，在 6 月 8 日，当“斯诺登（Snowden）”的名字被披露时，Snowden 的搜索量增加了。在 PRISM 日后的六周中，只有 Snowden 和 NSA 的搜索量持续上升，而“NSA”的搜索量稳步下降。人们对“隐私”的关注只是略有增加，在 PRISM 日后的一周内，达到了最大增长 0.003%。在同一周内，PRISM 本身的搜索上升了 0.01%，但随后降回了最初的水平，如图 1 所示。中央情报局（CIA）是另一家收集外国情报的美国联邦机构。它并没有吸引更多的查询，而且在 PRISM 日后，它的相关搜索量有所下降。

在搜索的用户中间，虽然名人 Snowden 仍然流行（$p < 0.0001$，t-检验，PRISM 日之前与之后的对比），但隐私相关的搜索则无法维持更大的用户基础（$p = 0.4$）。把调查范围限于新闻结果的搜索后，我证实了这一观察结果。PRISM 日之前和之后的情况相比，搜索”Snowden“的平均用户基础上升了 0.007%，而对隐私的新闻搜索没有受到影响，变化的幅度低于 0.00001%。

对隐私增强技术的搜索出现了极小的增加，其中通用加密第一周最多增加了 0.001%，Tor 同一周增加了 0.002%，以及 PGP 在同一周增加了 0.0003%。（Tor 是用于隐藏通讯对象的系统，PGP 是用于加密消息传输的系统）

**长期的演变。**要阐释搜索行为的长期演变则更加困难，因为离

**图 1.PRISM日（2013年6月6日）后各搜索词的搜索量的变化，用相对于参照期的百分点表示。**



参照期的时间变长引入了季节性波动和其他重要事件的影响。截止 PRISM 曰后的第 11 周，对于我考虑的所有词语，它们的搜索量的增加幅度不超过 0.02%，如图 1 所示。Snowden 和 NSA 的搜索量分别在 2013 年九月 / 十月以及十一月 / 十二月持续飙升，增加了多达 +0.06%。对于 PRISM、监听（surveillance）或中情局（CIA），则并没有明显的增加

PRISM 曰过去很久之后，，在第 19 周和第 28 周内，”隐私“的搜索量激增，分别增加了 0.03 和 0.05 个百分点，高于参照期，达到了揭秘事件之后未曾达到的水平。除了持续对政府监听进行报道之外，那几周媒体还报道了 Facebook 删除隐私设置，以及 Google 删除 Gmail 和 Android 中的隐私增强功能。与此类似，从第 15 周到第 21 周，”政府“方面搜索量的激增可以归因于 2013 年 10 月美国政府关门事件。

查询量的某些变化可以用一次性和持续性信息需求的对比来解释。那些想跟进斯诺登下落信息或 NSA 最新揭秘的用户会反复的查询相关关键字，而对于关于隐私或监听的背景资料，他们可能只会搜索一次。下列各节讨论了维基百科和普通网页上的网页浏览习惯，在一次性与持续性的信息搜寻的对比方面提供了更多的证据。

**依其他事件来确定基准。**与其他的事件相比，PRISM 的公开揭秘对搜索的影响似乎要小得多。我把它与三个造成社会影响的时事问题进行了对比，它们是我人工选择的在相关媒体报道不久后发生的全球知名事件。2013 年夏，大众媒体报道了哈桑·鲁哈尼（Hassan Rouhani）当选为伊朗总统这个地缘政治问题（2013 年 6 月 15 日），为期四天的大型体育赛事美国高尔夫公开赛（U.S. Open）（6 月 13 至 16 日）以及“王室宝宝”乔治王子诞生在剑桥（7 月 22 日）的事件（见图 2）。

**图 2.与国际政治、体育和八卦新闻中的其他事件的对比——PRISM相关话题的网络搜索量的变化**

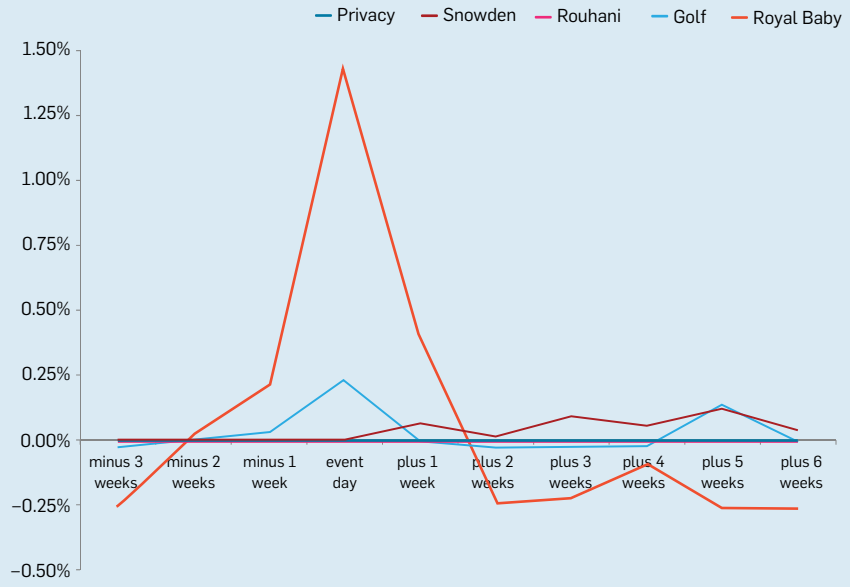
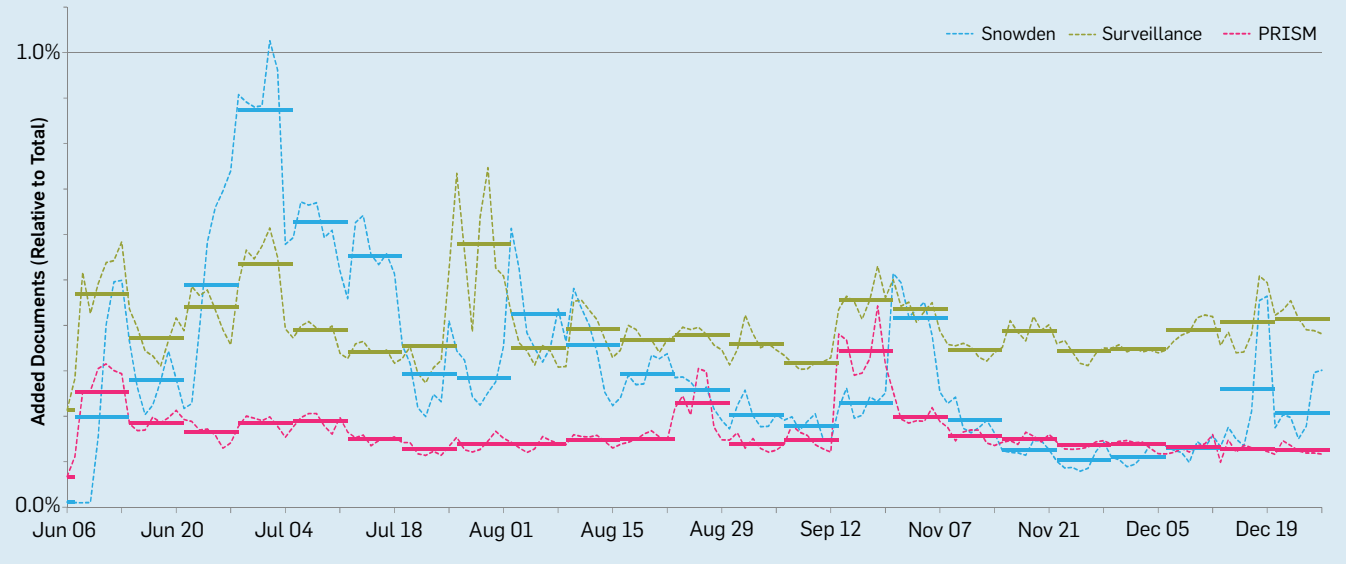



**图 3.初次揭露PRISM（2013年6月6日）后的30周内，分别提到 “Snowden,” “PRISM,” 和 “surveillance（监听）” 的新的在线文档；水平线为6月6日后每7天一个周期的均值；由于用于记录数据的系统出现系统故障2013年10月到12月下旬的数据缺失。**



在当选当天，有关鲁哈尼的查询量增加了0.0039%，低于对“PRISM”的关注，但高于PRISM日中对”隐私“的关注。然而，对鲁哈尼的关注消逝得更快；例如，当选后一周对鲁哈尼的搜索查询量明显高于当选前三周时的情况（$p = 0.0001$），但这种显著差别在当选第二周后基本消失（$p = 0.3$）。在为期四天的美国公开赛期间，高尔夫的搜索量达到了峰值（+0.25%），而且在之前的一周（+0.04%）也有所增加，但在公开赛后，该搜索量迅速下降，低于初始的用户基础。

王室宝宝的出生呈现了最大的日均增长，出生当日（7月22日）增加了+1.5%，后续一周仍在增加，增加幅度达+0.41%；之后，搜索新继承人的用户比例下降。这样算出的关注度可能会是低估的，因为参照期期间数值就已经相当高；如果是与之前的一个月相比，王室宝宝出生当日的用户搜索则多了一百多倍。在他出生前两周，即12天的时间里，所有用户中至少有0.1%搜索过王室宝宝，在7月23日，搜索量的比例达到了峰值，达2%以上。在与PRISM有关的搜索词中，NSA（美国国家安全局）和Snowden（斯诺登）分别在7月7日和6月11日达到了各自搜索量份额的峰值，均为0.4%.

**揭露PRISM后的浏览行为**

与网络搜索一样，浏览行为也能说明问题，因为信息搜寻说明人们对隐私话题的关注。因此，我统计了访问Snowden、PRISM、隐私和监听有关的页面的用户量，使用了与之前相同的时间箱——PRISM日之后时长为一周的时间周期。与之前一样，我把群体数量与PRISM日前三周的情况进行了对比。

浏览与Snowden有关的网页的用户增加了两个数量级。这是本次调查的所有话题中，迄今为止增幅最大的话题。在PRISM日后的所有六周内，Snowden相关网页仍然备受瞩目（$p < 0.00001$，t-检验）。PRISM和监听相关的网页也吸引了更多的用户，其用户数分别增加了95%和250%。公众对PRISM的关注度得到了保持（在所有的六周内，页面访问量明显增加，$p < 0.0001$），但在该期间内，访问监听相关网页的用户数量稳步减少（$(\rho = -0.93$，$R^2 = 0.87)$），在第五周，它与参照期的情况不再有任何区别（$p = 0.22$）。访问隐私相关网页的群体数量没有明显的增加。在PRISM日后的一周内，数量略有增加（+4%），但随后便降低至初始水平以下（降低的幅度高达-13%）。

**Snowden（斯诺登）的揭秘几乎没有给隐私增强技术带来新的用户。**

在较长期趋势方面，只有“Snowden”在持续吸引明显更多的读者，PRISM或隐私网页的访问量降至或低于初始水平，而监听相关的网页的访问人数在八月和十一月、十二月偶尔高于参照期，不过并未体现出清晰的趋势。

维基百科是标准的在线参考，它满足了通用的信息需求。PRISM和Snowden的英语维基百科页面分别创建于2013年6月7日和9日，因此在参照期内没有数据。对于隐私和监听方面的百科条目，它们的页面浏览量显著增加（$p < 0.001$），分别增加了23%和75%。但是，截止第二周，对维基百科中描述“隐私”的文章的页面浏览量已经下降并低于参照期，但稍后在2013年9月有所增加；截止第五周，对描述“监听（surveillance）”的文章的访问量已经回到了初始水平。25虽然描述“Edward Snowden”的文章创建于参照期后，但是，在通过网络搜索来对比分析一次性或持续性的信息搜寻行为方面，维基百科的文章统计数据让我们有了更进一步的认识。没有数据表明Snowden能够持续地吸引读者，而隐私和监听则不能吸引读者。另一方面，对比PRISM日后的即时反应和长期演变时，我发现，与隐私和监听相比（均为-13%），人们对Snowden的每周关注下降得更为猛烈（-80%）。

早期对 PRISM 的报道中媒体提到，NSA（美国国家安全局）会“直接监听九大美国互联网公司的中心服务器”[5]，其中包括微软，所以一些搜索信息的个人可能会因此去查询该企业网站上发布的隐私声明。虽然消费者极少查阅隐私策略，但是他们可能会突然地变得急切起来，想了解企业具体的实施情况，包括“当法律要求或为了应对执法部门或其他政府机构等提起的司法程序或法律请求”而进行的数据共享。[14] 在 PRISM 曰后的第一个六周内，访问量增加了多达 12%，且在整个扩展范围内均明显高于参照期（$p < 0.01$），并呈现了一个向上的趋势（$R^2 = 0.10$，$p = 0.09$，F- 检验）。然而，我只在 PRISM 曰后的第三周观察到了明显更高的数值，在紧接揭秘的那周内数值并没有明显增加（$p = 0.29$）。由于访问量的计算方式发生了变化，无法获得前一年的数据，所以不能排除季节效应。

**在2013年7月2日的顶峰时期，在所有新发现的在线文档中，有1%以上的文档含有“Snowden”。**

## 揭露 PRISM 后的隐私增强技术

人们通过关注度和搜索量表现出的隐私态度问题，和他们所采取的相应行为是有差异的，这在 [18] 中得到了良好的阐释，并通过此次 PRISM 事件得到了验证。例如，Facebook 报告了信任降低的现象，但未影响人们对它的社交服务的使用频率。[24]

Tor 是一种隐私增强技术，通过使用多个中继来进行网络流量的路由选择，它允许用户隐藏他们的位置和浏览习惯。Tor 宣传自己是一个防范网络监听的保护措施；在《*华盛顿邮报*》中，“使用 Tor 进行匿名浏览”号称“阻止 NSA 窥探您的五种方式”之首。[10]Tor 的使用明显增加——$p < 0.01$，增长幅度多达 10%——但是只出现在 PRISM 曰后的第三周至第五周。Tor 仍然是一个小众技术；它的用户群的增长仍然不大（在第四周达到了最大值 15,000 名用户），与 2013 年 9 月相比相当小，当时网络罪犯对 Tor 基础设施的滥用让用户数达到了四倍。

Anonymox 是用于隐藏用户 IP 地址的另一种浏览器插件服务。2013 年，Anonymox 使用增加了一倍以上，在该年年底用户数达到了 200,000 以上；在我的分析中，我校正了这一趋势。PRISM 曰代表了五月到七月之间的临时高水平，但它自己并没有造成使用峰值（第二周出现了最大增长，$p = 0.24$）。

所有网络浏览器的隐私设置配置是 PRISM 曰后行为变化的第三个指标。虽然配置代理仍然需要一定的技术水平，但是浏览器设计可让普通用户也能进行配置。我使用的数据源于同意分享他们的使用指标的 Internet Explorer 用户样本。然后我统计了这些用户中在前一个月有多少人就已经选择了 Internet 选项中的“privacy（隐私）”标签。与参照期相比，在 PRISM 曰之后，该行为增加了多达 2.8 个百分点，可以说是显著的增加（$p = 0.02$）。

隐私选项的配置仍然是一种使用较少的浏览器特性，而且我发现在扩展的期间内它并没有明显增加。对比 Tor/Anonymox 与 Internet Explorer 的数据后，我注意到，使用匿名代理是一种持续进行的隐私活动，而调整浏览器设置，使之更偏向隐私保护则是一次性操作。可用数据仍可能高估了用户的比例。例如，我把打开隐私设置页签的用户计算在内，但他们不一定会修改设置，更不用说激活限制性更强的设置。不仅如此，Internet Explorer 遥测数据的样本规模不大。

## 媒体报道的演变

通过与媒体报道的对比，我解释了之前描述的消费者隐私行为方面的数据。除了 PRISM 曰后第二周出现下降之外，调查中包含的词语呈现了一致的上升趋势。例如，与第一周相比，第六周有更多的文档提到了 Snowden 和监听。在 2013 年 7 月 2 日的顶峰时期，在所有新发现的在线文档中，有 1% 以上的文档含有“Snowden”。在后续的每一天，“PRISM”每天平均在

240,000 多份新文档中出现，与之竞争的有其他的计划（比如 Tempora 和 XKeyScore）（见图 3）。

在 PRISM 日后 30 周的时间内，媒体报道仍在继续，PRISM（$\rho = -0.00002$，F- 检验：$p < 0.0001$）、监听（$\rho = 0.0000$，$p = 0.001$）或 Snowden（$\rho = 0.0000$，$p = 0.004$）没有出现显著的下降趋势。截止本文研究阶段的结束之时——PRISM 日后的第 30 周——有关 Snowden 的相对日均文档数量为 2013 年 6 月 6 日时的 18 倍。

## 结语

本文进行了对美国网络用户的隐私行为的首次纵向研究，原因是 2013 年 Edward Snowden 揭露政府监听和当前网络普遍缺乏通讯隐私的状况后，网络用户可能已经受到了影响。我对比了 PRISM 日前和 PRISM 日后隐私增强技术的使用情况，并使用了网络搜索和浏览活动作为调查人们对隐私的关注以及人们的信息搜寻行为的入口。通过微软必应搜索引擎，我对网络搜索行为进行了分析，但这可能已经引入了无法量化的偏差（如果它存在的话）。然而，外部的证据说明，必应可能更吸引那些了解隐私的人群，[19,22] 这意味着，我观察到的少量增加可能仍然是一种高估。

我整合了来自原始资料的高精度数据，它说明了，新的 PRISM 方面的公众信息导致公众对隐私和保护的关注短暂增加。然而，与其他的新闻事件相比（比如王室宝宝和美国高尔夫公开赛），该峰值要低得多。在 Facebook、Android 和 Gmail 删除隐私增强功能之后，人们对隐私的关注也有所增加，而且比 PRISM 事件的影响大。

虽然在 PRISM 日后的 30 周内，对 PRISM 和监听的媒体报道有所增加，但是很多隐私行为迅速地消散了。微软企业隐私策略页面的访问量一直相当高，但是只有某些隐私相关网页留住了较多的读者——那些与 Snowden 和监听有关的网页——而维基百科中有关 PRISM 话题的文章失去了增加的读者群。Snowden（斯诺登）的揭秘几乎没有给隐私增强技术带来新的用户；2013 年，匿名代理的数量有所增加，但是 PRISM 并没有增强这一势头。虽然 Snowden 本身仍然是吸引大量读者的唯一话题，但是或许这种关注已经和他揭露的隐私侵害情况不再相关。

只有使用高时间精度的纵向研究才能揭示隐私侵害对人们行为的影响。此类研究的数量不多，可能是因为很难获得能说明核心特征的数据。由于无法直接观察用户对隐私的关注度，我必须借助其他方式（比如信息搜索行为）来进行分析。我对数据源的选择部分考虑了实际情况，旨在获取详实的数据，较好地描述相同群体在扩展期间内的情况。

因此，我选择聚焦于美国的英语用户；对于很多消费者服务而言，这个所谓的 "en-us（美国英语）"市场是一个标准的地理过滤器，对于多个不同的数据集，它让我圈定了一致的范围。在范围较窄但质量高的数据之间进行取舍后，导致了一个明显的局限，因为用户可能在他们的软件设置中设置成为 "en-us（美国英语）" 用户。不过，之前的内部分析说明，该群体的比例是可以忽略的。我还计划调查 PRISM 在多个国家已经产生的（以及继续产生的）影响。我的结果说明，有理由对隐私问题方面政府的不道德行为与企业的不道德行为所产生的影响进行对比。 C


参考资料

1. BITKOM (Federal Association for Information Technology).Internetnutzer werden misstrauisch, July 25, 2013; http://www.bitkom.org/de/presse/8477_76831.aspx
2. Dierig, C., Fuest, B., Kaiser, T., and Wisdorff, F. *Die Welt* (Apr. 13, 2014); http://www.welt.de/wirtschaft/article126882276/Deutsche-unterschaetzen-den-Wert-persoenlicher-Daten.html
3. DuckDuckGo.DuckDuckGo Direct queries per day (28-day average), July 2014; https://duckduckgo.com/traffic.html
4. Dumais, S., Jeffries, R., Russell, D.M., Tang, D., and Teevan, J. Understanding user behavior through log data and analysis.Chapter in *Ways of Knowing in HCI*, Springer, New York, 2014, 349–372.
5. Gellman, B. and Poitras, L. U.S., British intelligence mining data from nine U.S.Internet companies in broad secret program.*The Washington Post* (June 7, 2013); http://wapo.st/1888aNq
6. Ginsberg J. et al.Detecting influenza epidemics using search engine query data.*Nature 457*, 7232 (Feb. 19, 2009), 1012–1014.
7. Jackson, D., Davis, S., and Johnson, K. Obama's spy plan includes Internet:NSA's expansive reach revives an unsettled and vexing 9/11 debate.*USA Today* (June 7, 2013), A1.
8. Landau, S. Making sense from Snowden:What's significant in the NSA surveillance revelations.*IEEE Security & Privacy 11*, 4 (July–Aug. 2013), 54–63.
9. Lazer, D., Kennedy, R., King, G., and Vespignani, A. The parable of Google flu:Traps in big data analysis. *Science 343*, 6176 (Mar. 14, 2014), 1203–1205.
10. Lee, T.B.Five ways to stop the NSA from spying on you.*The Washington Post* (June 10, 2013), http://www.washingtonpost.com/blogs/wonkblog/wp/2013/06/10/five-ways-to-stop-the-nsa-from-spying-on-you/
11. Marthews, A. and Tucker, C. *Government Surveillance and Internet Search Behavior.*SSRN Working Paper, Social Science Electronic Publishing, Inc., Rochester, NY, 2014.
12. McCombs, M.E. and Shaw, D.L.The agenda-setting function of mass media.*Public Opinion Quarterly 36*, 2 (Summer 1972), 176–187.
13. Microsoft.Bing Help, 2013; http://onlinehelp.microsoft.com/en-us/bing/ff808447.aspx
14. Microsoft.Microsoft.com Privacy Statement, 2013; http://www.microsoft.com/privacystatement/en-us/core/default.aspx
15. Mozilla. anonymoX:Add-ons for Firefox, 2014; https://addons.mozilla.org/en-US/firefox/addon/anonymox/
16. Nakashima, E. and Markon, J. Dozens of attacks foiled, NSA says.*Washington Post* (June 13, 2013), A1.
17. Phelps, J., Gonzenbach, W., and Johnson, E. Press coverage and public perception of direct marketing and consumer privacy.*Journal of Direct Marketing 8*, 2 (Spring 1994), 9–22.
18. Preibusch, S., Kübler, D., and Beresford, A.R.Price versus privacy:An experiment into the competitive advantage of collecting less personal information. *Electronic Commerce Research 13*, 4 (Nov. 2013), 423–455.
19. Protalinski, E. Microsoft confirms Google privacy campaign to promote Bing is aimed at Apple Safari users.*The Next Web* (Sept. 20, 2012); http://thenextweb.com/microsoft/2012/09/20/microsoft-confirms-google-privacy-campaign-aimed-apple-safari-users/
20. Rosenblatt, S. Escaping Google's gravity:How small search engines define success.*CNET* (July 11, 2013); http://cnet.co/15kOLCY
21. Roznowski, J.L.A content analysis of mass media stories surrounding the consumer privacy issue, 1990–2001.*Journal of Interactive Marketing 17*, 2 (Apr. 2003), 52–69.
22. Stone, B. Facebook radically revamps its search engine.*Bloomberg Businessweek* (Jan. 15, 2015); http://www.businessweek.com/articles/2013-01-15/facebook-radically-revamps-its-search-engine
23. The Tor Project, Inc.*Tor Metrics Portal:Users 2013*; https://metrics.torproject.org/users.html?graph=direct-users&country=us#direct-users
24. Van Grove, J. Zuckerberg:Thanks NSA, now people trust Facebook even less.*CNET* (Sept. 18, 2013); http://news.cnet.com/8301-1023_3-57603561-93/zuckerberg-thanks-nsa-now-people-trust-facebook-even-less/
25. Wikipedia article traffic statistics.2013; http://stats.grok.se/



Sören Preibusch（http://preibusch.de/）现任加州山景城谷歌公司的用户体验研究员。书写本文时，他在英国剑桥微软研究院任职。



译文责任编辑：杨珉

DOI:10.1145/2743132

**机器人的动作源于机器人的运动。然而，机器人的动作存在于物理空间，机器人的运动始于电机控制空间。那么，机器人怎么用运动来表达动作呢？**

JEAN-PAUL LAUMOND，NICOLAS MANSARD，
JEAN BERNARD LASSERRE

# 最优化——机器人动作中的运动选择原则

**运动是生**物系统的根本特性（见图 1）。为了生存，植物和动物都必须运动。动物与植物的区别在于动物必须探索世界来觅食。食肉植物处于固定的位置上，捕捉莽撞的昆虫。这些植物必须利用以自我为中心的运动。同时，猎豹离巢去觅食。

觅食堪称行动的典范。在物理世界中，任何动作均会涉及以自我为中心的运动，探索运动或两者兼有。以此类推，机械手机器人利用以自我为中心的运动，移动机器人通过移动来探索世界，类人机器人结合了这两种类型的运动。

动作发生在物理空间内。运动则源于运动控制空间。机器人与任何生物系统一样，只能通过传感器和马达间接地接触物理空间。机器人运动规划和控制探索了物理、感觉和运动空间之间的关系；这三种空间是几何学的基础。[32] 那么，如何把物理空间中描述的动作变换成为用运动坐标描述的运动呢？这种逆变换（inversion）是机器人学中的根本问题。

在生命科学中，研究人员承认，感觉运动控制中的最优化原理清晰地解释了经验数据，或者说该原理至少比别的原理要好。[40] 二十世纪七十年代，Whitney 开创性地在机器人学中首次提出了把机器人的动作描述为拟优化的运动这一理念。[41] 如今，在经典的机器人控制中，这一理念已经得到了长足的发展[37]，在多学科的方法中，同时还提出了与之共同发展的新范式。[35] 运动优化似乎成了选择动作的自然法则。然而，如本文姊妹篇所述，[24] 最优化方程在大多数情况下难以驾驭，且其数值最优化的求解过程在

## » 重要见解

- **对于机器人和生物而言，物理空间中描述的动作与源于运动空间的运动之间存在着某种关联，这种关联一般会转向几何学求解，具体来说是线性代数。在生命科学中，在感觉运动控制中应用最优化原理后，解开了经验数据的谜团。自二十世界七十年代以来，把机器人的动作描述为拟优化的运动这一理念已经在机器人学中得到了发展。**
- **在执行给定动作的所有可能运动中，最优化算法倾向于根据给定的性能准则选择最佳的运动。不仅如此，它们还支持实现次要动作。**
- **最优运动是动作的特征。如何揭示隐藏在给定动作下面的最优化准则？该问题开启了面向逆最优控制的挑战性议题。**

插图由PETER CROWTHER ASSOCIATES提供

**图 1.石头和锤子自己不能移动。**

移动是生物系统（以及机器人）的特权。植物（以及机械手机器人）通过以自我为中心的移动把世界带向自己。动物（以及移动机器人）通过搜寻道路来探索世界。人类（和类人物体）的动作由两种类型的运动构建。

工程中是出了名地慢。本文旨在简短地概述该领域中的最新进展。首先，我们说明了为什么应该把机器人运动的最优化技术看成用于冗余机器人的运动选择原则。在这一方面，我们回顾了在类人机器人上的最新应用所引发的结果和挑战。本文余下篇幅用于介绍逆最优控制。作为一种方法，逆最优控制可以让我们更好地理解自然现象，并把它们变换成工程实践。这一问题开启了多个难度颇高的挑战性问题。在这方面，基于最近提出的多项式最优化技术的方法似乎与经典的机器学习方法互补。

## 线性化的优点和局限

将机器人动作变换成机器人控制空间运动有许多方法，例如，操作空间描述法（operational space formulation）[19] 任务函数法（task function approach）[34]，以上仅举数例。任务的概念包含了物理空间中描述的动作的概念。任务空间可能是物理空间（比如把机械手的末端执行器放到世界坐标系中定义的某个位置上）或是传感空间（比如跟踪机器人摄影机框架中的一个物体）。所谓的任务函数的作用是建立任务空间和控制空间之间的联系。

由于存在底层的强非线性变换，求解逆问题的代价非常高（求出几秒的运动需要几分钟或几小时的计算）。为了满足机器人控制频率所设定的时间约束，通过考虑任务空间和位形空间两者的切空间，研究人员只能局部解决该问题。这种线性化涉及雅可比矩阵[31]，并需借助于线性代数中的所有技术。由于位形空间的切空间集中了位形速度（通常含有机器人控制输入），所以线性化尤为吸引人。对这一原则进行动态扩展后，还能考虑基于扭矩的控制。[19]

雅可比矩阵随机器人的位形变化而变化，这使轨迹的搜索变成了非线性的。然而，对于给定的位形，它确定了一个关联未知系统速度与任务空间中的速度（作为参考给出）的线性问题。从数值的角度来看，这个问题是线性的，可在每个瞬间轻松求解，获取系统速度。在某一时间区间内从初始位形开始对这一速度进行积分后，便能绘制出趋向于完成任务的轨迹。与此类似，也可以对机器人实时应用该速度，控制它趋向目标。在传感器测量值更新后得到的每个新位形处，重新初始化线性问题，反复执行这一过程。这种迭代的原则对应于下降迭代算法（比如梯度下降或牛顿 - 拉夫逊梯度），这些算法用于从数值方面计算给定函数的零值。然而，这一方法的结果不止这些：假设下降步长相当小，则下降迭代的序列是对从初始位形到目标的实际轨迹的离散化。瞬时线性化的缺点是，它没有给控制提供任何前瞻能力，在靠近非凸障碍物时，通常这可能会引导机器人趋向局部极小值。这是广为人知的线性化诅咒。

## 运动选择

任务空间的维度可能等于、大于或小于控制空间的维度。为简便起见，我们认为任务空间是一个流形，它描述了全驱动机械手的末端执行器的位置。当任务空间和位形空间两者的维度相等时，任务空间中的每个点定义了一个单位形（single configuration）[a]，而任务函数可用于驱动机器人至唯一的位形。这里不存在运动选择的问题。雅可比矩阵是方阵可逆的，求解线性问题相当容易。在这种情况下，任务函数法最初用于定义可采纳性（admissibility）属性，以连接位形空间中的两点，同时避免奇点。[34]

在其他情况下，最优化用作运动选择原则。最优化准则的选择确定了对雅可比矩阵求逆的方式，后文我们将进行论述。

当任务空间的维度大于位形空间时，有时候无法找到满足任务目标的位形：任务函数不是满射（onto），也就是说雅可比矩阵的行数多于列数。此时，无法在位形切空间中找到一个速度与任务切空间中的速度相对应。例如，当需要在摄像机框架中跟踪很多个点时，视觉伺服中存在这样的情况。[4] 当优化相机相对于多个标志的多个位置时，同时定位与地图构建中也存在这种情况。[13] 最优化还被用于找出一个可以极小化任务切空间中的

a 任务函数中的非线性可以生成完成任务所需的离散位形集合，它们与多个选择对应。在这种情况下，该讨论成立，但仅在局部成立。

误差的速度。那么，该问题变成极小化到参考任务向量的距离。通常情况下，参考任务向量不可达。在特殊情况下，当该参考属于雅可比的象空间时，最优化的残差为零。视觉伺服中是这样的情况，此时已经从实际场景中获取到了目标图像，且没有噪声。

与此相反，如果任务空间的维度小于位形空间的维度，那么若干位形会对应一个单独的任务。该任务函数不是一对一映射；也就是说，雅可比矩阵的列数多于行数。例如，在 30- 关节类人机器人用手拾球的场景中，任务空间的维度为 3，而位形空间的维度为 30. 几种运动均可完成任务。对于该任务而言，称该系统存在冗余。那么，会使用最优化作为一种在所有可采纳运动中进行选择的准则。在此情况下，位形切空间中的几个向量都能在任务空间中产生相同的效果。同样，某些速度在任务空间中不会产生效果。该位形切空间的子集为雅可比矩阵的核，称为任务的零空间。零空间中的任何速度都不会改变任务。把满足任务的给定位形切向量与零空间加在一起后，得出了满足任务的所有速度的向量空间。依照某个准则（如极小范数速度）在该空间中选择一个样本时，会有极小值问题。

总体来说，该任务函数可能既不是满射，也不是一对一映射；也就是说，雅可比矩阵可能既不是满行秩（它的各行不是线性无关的），也不是满列秩（换言之，它的各列不是线性无关的）。一般情况下，位形切空间中没有向量能满足任务（因为变换不是满射），但有无数的向量能极小化到该任务切空间中的任务向量的距离（因为变换不是一对一的）。因此，选择问题变成了双重极小值问题：我们需要同时极小化到任务的距离以及位形速度的范数。Moore-Penrose 伪逆 [2]（又称为最小二乘逆）给出了这种双重极小值问题的解法。请注意，通过改变切空间中的度量，可以在相同的框架内考虑其他的极小值准则。例如，我们可以使用加权的伪逆，其中系统输入的分量（雅可比矩阵的列）和任务残差（雅可比矩阵的行）拥有不同的权重。如前所述，最优向量与零空间的和给出了针对第一个问题（到任务的最小距离）的所有最优解的集合，但对于第二个问题（最小速度范数），它只是次优的。

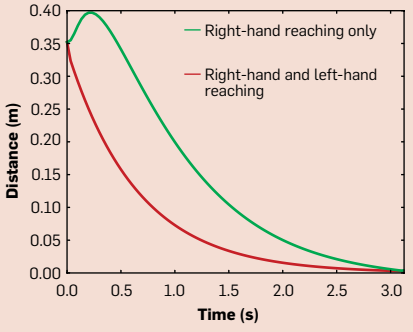
**可以使用反向工程技术从运动的观察结果中识别出动作。**

### 使用最优化作为选择原则

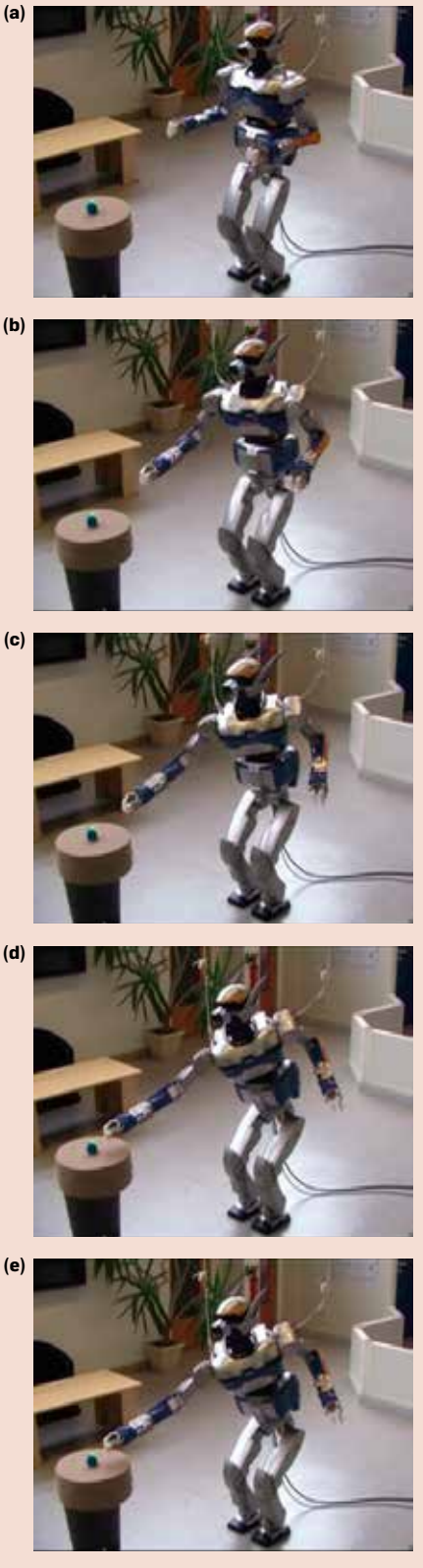
*冗余系统的任务堆栈。*处理某个任务时，如果机器人存在冗余，那么给机器人分配次要任务会相当有趣。对于类人机器人而言，这种情况就是机器人可以同时执行两个任务。让我们来考虑下两个不同的任务函数，它们分别处理右手和左手的位置。如何检查这两种任务是否有冲突？简单的思路是设定这两个任务的顺序。在积分过程的每一个时间步长中，从与第一个任务相关的位形切空间中选择一个向量。然后，仅在受限的速度集合范围内考虑次要任务，该集合应处于第一个任务的核内。应用于第一个任务的推理也可应用于投射出的次要任务：任务函数可能是满射，一对一，或是两者都不是。具体来说，如果它不是满射，称任务为奇异的（从雅可比矩阵是秩亏的意义上说）。现在可以区分出两种情况。如果（非投射的）次要任务函数不是满射，那么称该奇点为运动学奇点：由于有次要任务，它在本质上是运动学的。与此相反，如果（非投射的）次要任务函数是满射，那么称该奇点为算法奇点：因为与主任务存在冲突，次要任务变成了奇异的。[5] 在各算法奇点的范围之外，称两个任务不存在冲突，两个任务的执行顺序不相关。

**图 2通过任务堆栈生成的运动的示例**

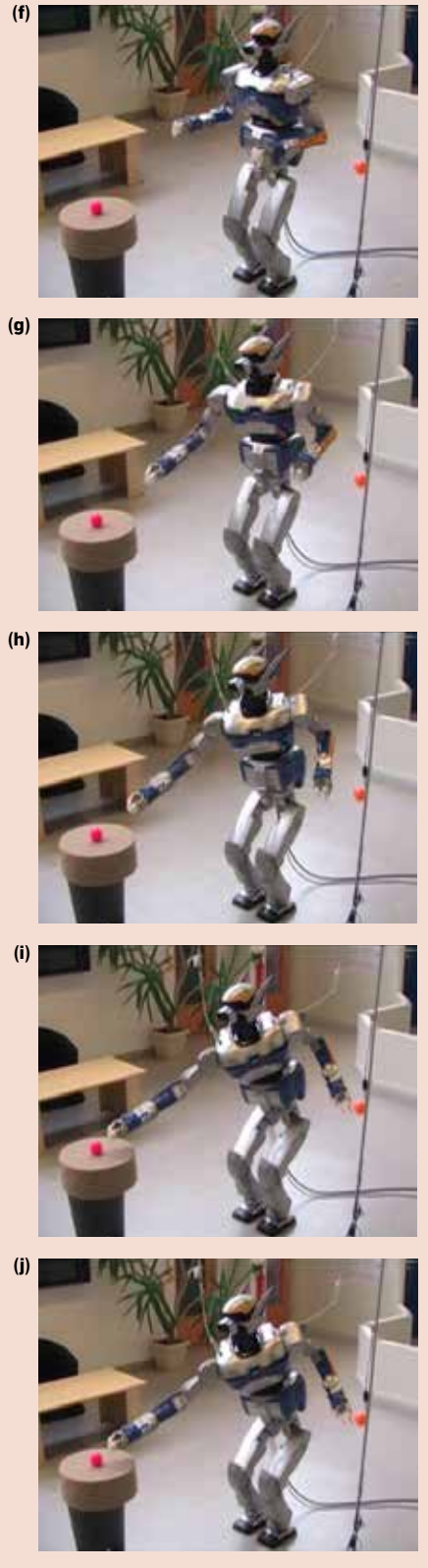
左列：任务堆栈由三个约束（双脚和质心应该处于固定的位置；它们一直都是可行的，且得到了满足）以及两个分别控制凝视和右手的任务组成。在最终的位形（e）处，机器人已经用右手触到了球，且该球位于机器人视场的中心。它的左手进行了移动，仅仅是为了控制质心的位置。右列：运动是相似的，但是添加了一个任务来控制左手的位置：为左手设置的期望位置为前一次运动中左手达到的最终位置。这两种运动看起来非常相似，但是它们的“意义”不同。在正确的运动（a-e）中，为了控制平衡，左手进行了运动；在正确的运动（f-j）中，为了到达特定的位置，左手进行了运动。




左手的各种运动看起来是一样的，但实际不是完全一样。这些曲线描绘了这两种情况下左手从初始位置移动到最终位置时的距离函数。这些曲线是不同的。该距离函数（红线）的指数减少“说明”存在左手取物任务。




右手取物，同时保持平衡：左右进行了移动，仅仅是为了维持平衡。




右手和左手同时取物：为左手设定的目标是前一次（只有右手）运动中左手达到的最终位置。

对于其他任务，也可以反复进行这种推演过程，进而得出所谓的任务堆栈。[26] 这样一来，降低了连续零空间的维度。当所有任务被处理完成后，或者一旦零空间的维度消失后，该过程终止（见图 3）。在后一种情况下，我们仍能从剩余零空间中的多个向量中选出极小范数向量。

零空间首先被在数值分析的框架内，用于引导顺序优化的梯度。[33] 在机器人学中，它被用于使用冗余度机器人执行定位任务，同时考虑关节的限制。[25]Nakamura 把它推广到了任何数量的任务，[30]Siciliano 则提出了它的递回式 [38]（还见计算机动画背景下 Baerlocher 的论述 [1]）。

在本文中，我们把范围限制在逆运动学上，也就是说，从参考速度的任务约束出发计算机器人的速度。在逆动力学中，也可以用相同的方法从同质的操作约束（典型情况下是参考力或加速度）出发计算系统力矩 [19]（典型的情况下是关节力矩，但也可能是腱力或其他的驱动参数）。在这种情况下，欧几里德范数是无关的，通常研究人员更偏向使用加权逆为运动施加最小的能量。

*任务堆栈、二次规划和不等式约束。*任务堆栈堪比二次规划：二次规划是一个涉及线性约束集和二次代价的最优化问题（例如，在最小二乘意义上拟合线性函数）。这样一来，它类似于拥有两个任务的堆栈：第一个（优先级更高）任务是一种约束；次要任务则是需要极小化的代价。然而，这种相似却不是完全的：二次规划中的约束应该是可采纳的（至少存在一个可行解），而主任务的情况并未如此。同时，可以扩展任务堆栈，容纳两个以上的任务。

迄今为止，我们已经探讨了与位形切空间中的等式相对应的任务，在最小二乘意义上可得出满足该式的最佳解。让我们来考虑一下由不等式集定义的区域：机器人可以在一个区域内自由活动，但是必须处于该区域内；当任务变得不可行时，它应该在最小二乘意义上极小化到区域的距离。这种不等式约束无法通过本文描述的方法直接求解。

当不等式约束满足时，第一种解法历来在任务空间中设置一个零速度。这是由 Khatib 提出的人工势场法：[18] 目标区域通过低或零代价描述，越靠近区域的限制，代价越高，这沿袭了内点数值算法中使用的障碍函数的行为。那么，当机器人靠近区域边界时，该函数的梯度会作为一种虚拟的力量把机器人推向区域的内部；在处于区域内部时，该函数的影响为零或极低。

对于机器人控制而言，当机器人被推向限制时，一般倾向于使用惩罚函数，而不是障碍函数来阻止不良的数值行为。对于简单的任务，或者当不等式任务的优先级最低时，获取的行为总是令人满意的：当不等式的条件满足时，该机器人不需要移动。然而，使用这种方法来施加一种层级相当难。具体来说，在冗余度非常高的场景中（例如，使用六个关节的机器人臂在三维空间中取物）施加机器人约束时，如果不等式任务的重要性居次，则可以使用梯度投影法 [27]。当不等式任务的优先级较高时，该任务区域的某个边界的饱和（saturation）将对应于分配了一个自由度，[b] 它被分配用于确定与该边界正交的速度。因此，对于任何次要任务，无法再获得此自由度。不仅如此，当机器人在区域内部自由移动时（距各边界相当远），次要任务可使用此自由度。为了利用在由不等式约束确定的区域内提供的冗余，应该动态分配对应的自由度。如果不等式约束得到了满足，自由度未分配且可供次要任务使用：此时称约束无效（inactive）。如果违反了约束，则使用对应的自由度来尽可能地满足约束：此时称约束有效（active）。

所有有效的约束的集合成为有效集。有效集搜索算法是一种迭代的求解方案，它在所有可能有效的约束范围内进行搜索。在每次迭代中，它会求出满足有效约束的候选解。根据有效和无效约束相对于候选解的状态，修改有效集并反复进行这一过程。有效集搜索算法是求解含不等式约束的二次规划时经典算法。还可引入多个螺旋（coning）目标之间的优先顺序，进而引出层次性的二次规划（hierarchical quadratic program）。[9]

接下来，我们用 HPR2 类人机器人同时执行两个取物任务并遵守平衡约束的范例来说明任务堆栈框架。所有的基本任务被嵌入进了一条单全局轨迹。我们将会看到，二次规划中引入的层次引出了任务向量空间中的一种结构。这样一来，全局轨迹看起来就像基本运动的组合体，其中每个运动刻画了一个给定的任务（或子任务）。然后，可以使用反向工程来确定运动的“意义”，也就是说，运动中正在纳入的各种任务。

## 运动是行为的特征

我们此处回顾在类人机器人 HRP2 上任务堆栈的两种实际应用。[c] 第一种应用说明了在涉及所有的体节并遵守物理约束时，如何描述复杂的动作。第二种应用说明了如何能够通过反向工程技术从运动的观察结果中识别出动作。

*从动作到运动：在实践中使用基于最优化的选择原则。*任务堆栈是生成和控制机器人运动时使用的通用工具。给定初始位形后，把任

b 自由度是位形切空间中各种控制手段的线性组合。

c 详细的描述参见Hak 等人的著作。[12]

务集加入堆栈，整合由此得出的速度，直到所有有效任务聚集在一起，从而生成运动。任务堆栈可在多种不同的机器人学场景中使用。在类人机器人中，经典的任务处理取物（用末端执行器的放置位置来描述）、视觉伺服（用对机器人摄像头的象平面中物体位置的凝视的控制来描述），或准静态平衡（用对质心的控制描述，控制方式保证它在地板上的投影处于脚的支撑多边形范围内）。

例如，生成图2中的运动（左列）时，把两只脚和质心约束在它们的初始位置上，并设置了两个任务，以把右手和凝视均控制在机器人前方的球上。机器人向前俯身取球。这样做时，机器人质心前移。它的左手向后移动，以补偿质心的这种运动。左手的运动并未对应任何特定的动作。它只是保持平衡的副产品。

通过设置新的任务或改变有效任务的期望值，可以轻松地修改该运动。例如，图2中的运动（右列）可通过添加控制左手位置和方向的任务生成，该任务把左手放到第一个场景中的最终位置处。这个新任务是取物：左手必须触及目标。在两种场景中，左手的两次移动看起来非常相似，但是它们的意义不同。在第一种场景中，运动是无意的：左手移动的目的是控制质心的位置；那么，它的运动是其他任务的意外结果。在第二种场景中，运动是有意的：左手进行明确地移动，拾取给定的目标。对两种左手运动之间的细微差别进行详细分析后，消除了模棱两可之处。通过反向工程法可以实现这一目标。

*从运动到动作：动作识别的反向工程法。*这种层级结构人为地对堆栈中的任务解耦，以避免两个不同的任务之间存在任何冲突。它的不良效果是，进入给定有效任务空间的轨迹不受任何其他任务的影响。例如，在图2中（右侧）任务堆栈对左手进行了解耦，这样它可以独立于其他的两个任务运动。那么，在机器人运动的生成过程中，某个任务空间中的轨迹构成了该任务的活动的特征。

考虑下面的问题：我们观察可能的控制器已知的系统运动。仅仅观察关节的轨迹时，问题是如何进行重现，从可能的控制器中识别出哪些是有效的控制器，哪些是相关的参数。重现一个任务相当简单：使用对应的任务函数，把位形轨迹投射到所有的候选任务空间中。通过匹配投射出的轨迹与该任务模型，可以选择出最佳的任务（匹配和随后的选择再一次通过最优化完成）。

然而，如果任务堆栈人为地解耦了有效任务，则某些候选任务之间可能会出现耦合：例如，由于手腕和肘的运动链距离较近，它们的轨迹存在很多相似之处。这些相似之处会导致检测中的假正。为了避免这一问题，首先选择最相关的任务。然后，通过把该位形轨迹投射在检测到的任务的零空间内，可以取消由于该任务产生的运动。随后，反复执行检测算法，直到找出了所有的任务，也就是说，直到连续推演后运动的剩余数量为零。[12]

该检测算法可用于区分图2中执行的那两种外观类似的运动，而不需要使用任何背景信息。图3给出了连续推演的说明。这些任务按检测算法给出的顺序进行移除。首先移除了右手的任务（第二行），然后是质心（第三行）：这取消了左手的大部分运动，因为三个任务之间的耦合是相当重要的；然而，仍有一小部分左手的运动存在。与此相反，头部的运动（它几乎实现了与右手和质心解耦）仍然相当重要。在移除凝视任务（第四行）后，它完全为零。左手剩余的运动只能通过左手的任务来解释，检测出该任务是有效的，随后该任务被移除。最后，检测并移除了双脚的约束。移除第一只脚的效果（第六行）相当明显。

在识别动作并说明类似外观的机器人运动的区别方面，该算法达到了非常好的性能。除了机器人学之外，该方法还可以用于人类动作的识别。然而，它有一个关键的先决条件：最优化原理的知识立足于有意动作的运动生成。实际上，算法基于若干动作特征，这些特征是特定代价函数的典型结果。该方法还要求利用人类使用的协调策略的计算模型，以设计若干同时发生的运动原语。对于结合了计算神经科学和机器人学的未来研究而言，这些道路前景远大。

在这一阶段，我们已经看到了最优化原理和任务的概念能够帮助研究人员从运动出发来奠定动作的符号表示基础：动作被视为最优化过程的结果，它的代价代表了动作的特征。下一部分处理了从运动中识别动作特征时存在的双重问题。

## 逆最优控制

在本部分中，我们用源自类人机器人的案例研究作为开篇。假设我们希望类人机器人像人一样行走，也就是说，遵循类似人的轨迹。那么，问题就是：人类运动轨迹的计算基础是什么？在第一个阶段，我们说明了在多次重复和多个实验对象中，运动轨迹均是高度模式化的。该方法基于对多个轨迹的大量运动捕获数据基础的统计分析（七个实验对象，1，500多个轨迹）[15]。然后，在第二个阶段，假设人类的运动轨迹遵循了某条最优化原理。在人类或动物运动的研究中，这是一个常见的假说。那么，问题就是：在人类运动中，极小化了哪种代价泛函？在实践中，我们认为人类和机器人遵循了相同的模型，也就是说，我们精确地知道那个描述了某

**图 3.检测到七个任务中的每个任务后连续对运动进行的推演。**

从上到下依次为：最初的运动；移除右手任务；移除质心任务；移除凝视任务；移除左手任务；移除左脚任务；移除右脚任务。在最后一行，所有的任务都被取消了，推演出的运动被完全取消了。

些控制动作下的运动的微分方程，以及系统状态应该满足的约束。还可获得轨迹的数据基础。根据这一知识，确定在人类运动中被极小化的代价泛函变成了一个逆最优控制问题。

控制中的逆最优化方面的开创性工作可追溯至二十世纪六十年代应用于经济学的系统论。[29] 与此类似，对于最优稳定性问题，研究人员知道最优稳定性问题的每个值函数也是闭环系统的李雅普诺夫（Lyapunov）函数。Freeman 和 Kokotovic[10] 证明，互逆为真：也就是说，每个稳定闭环系统的每个李雅普诺夫（Lyapunov）函数也是有意义的最优稳定性问题的值函数。

在静态最优化中，直接的问题在于从可采纳解的某个集合 $K$ 中找出极小化某个给定代价函数 $f$ 的可行点 $x$。

我们将相关逆优化问题描述如下：给定 $K$ 中的可行点 $y$，找出极小化差值 $(g - f)$ 的范数的代价准则 $g$，确定 $g$ 时，需保证 $y$ 是使用代价准则 $g$（而不是 $f$）后直接最优化问题的最优解。当 $f$ 为零函数时，这成了静态版的逆最优控制问题。开拓性的工作可追溯至二十世纪九十年代的线性问题，以及 Manhattan 范数。对于后者，逆问题再次成为相同范数的线性问题。类似的结果对于拥有无限范数的逆线性问题也成立。有兴趣的读者可以从 Heuberger 的著作中找到一篇不错的综述，它描述了线性规划和组合最优化问题使用的逆最优化。[14] 请注意，在逆最优化中，主要的难点在于为某个给定点和某个候选代价准则设定一个可驾驭的全局最优化特性。这是为什么上述的大多数研究处理线性规划或组合最优化问题，因为它们可以获得全局最优化的某个特性，而且有时候能够有效地用于实际计算。这也解释了为什么之前逆（非线性）最优化没有得到很多的关注。

最近，在逆多项式最优化中取得了某些进展；换言之，这些是使用多项式目标函数和半代数集作为可行解集合的逆最优化问题。[22] 强大的表征导致了实代数几何 [21]，它通过某个正性证书来描述全局最优化约束。这些可被称为在多项式代价函数的系数的未知向量上的线性矩阵不等式（LMI）。后者中的集合是一个凸集，我们可以通过半定规划在该集合上进行有效的最优化，[23]，半定规划是一种强大的凸优化技术。然后，我们可以证明，逆最优解的计算可被归约成求解规模不断增大的半定规划层级。

现在我们回到用于拟人运动的逆最优控制问题上来，我们可以考虑一组函数基来描述候选代价函数。Mombaur 等人 [28] 提出的该方法基于两个主要的算法：高效的多目标直接打靶法，用于处理最优控制问题，以及最新的最优化技术，用于确保（直接）最优控制问题和测量值之间匹配良好。一旦（从给定类型的基函数中）识别出了最优代价函数，我们就可以在类人机器人上实现直接的最优控制解。迄今为止，该方法至少在一些样本实验问题上相当高效。然而，它需要定义先验类型的基函数。不仅如此，在该算法的每次迭代中，直接打靶法只提供了局部最优解。

因此，无法保证全局最优。

考虑该问题另一种的方法是拓展为逆多项式最优化开发的方法论 [22]，将其用于逆最优控制的场景。请注意，当最优值函数已知时，Hamilton-Jacobi-Bellman（HJB）方程是用于证明给定状态控制轨迹的全局最优化的一个完美工具。基本的思路是，对于在数据库保存的实验轨迹，使用松弛后的 HJB- 最优化方程作为全局最优化的证书。最优值函数（通常假设为连续的）可通过多项式在紧域上拟合。如果我们搜索积分型代价泛函（该泛函的被积函数 $h$ 也是多项式的），那么通过求解半定规划，可把这种全局最优化的证书用于计算 $h$ 以及相关的（多项式）最优值函数。按照 Lasserre 的著作中的步骤 [22]，我们求解了规模不断增大的半定规划的层级。在该层级的每个步骤中，或是半定规划无解，或是任何最优解 $h$ 会使得数据库中的多个轨迹成为使用多项式代价函数 $h$ 作为被积函数的问题的全局最优解。在层级中的位置越高，解的质量也就越高（但是计算的代价也越高）。

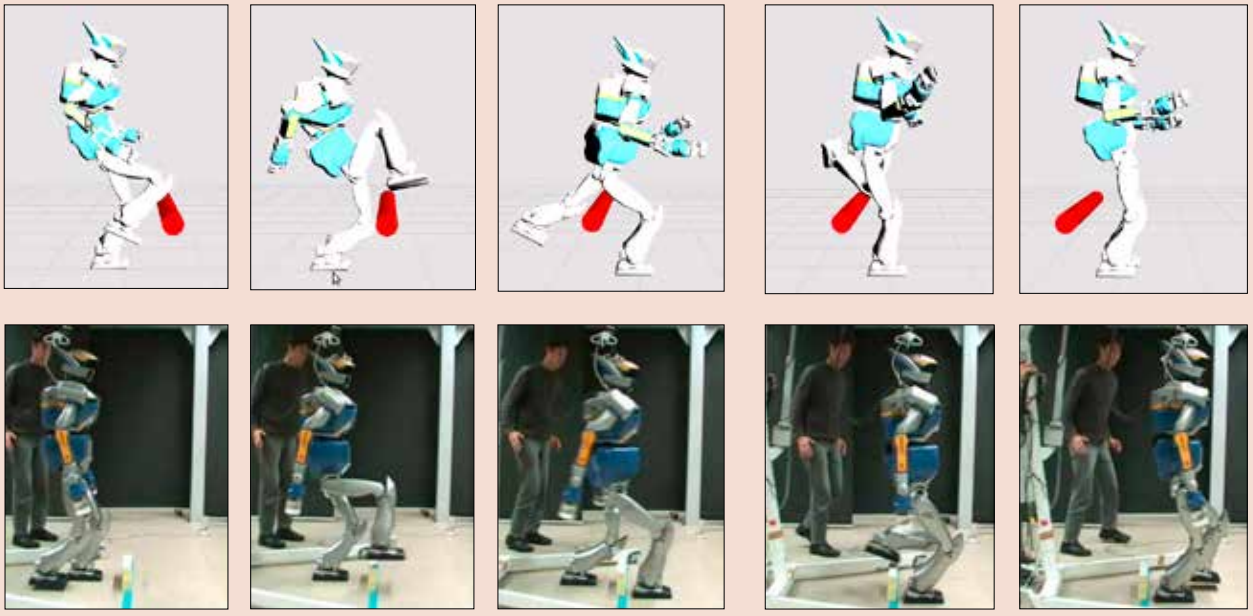
除了多项式最优化技术外，最近还从最优控制理论的几何学角度引入了其他的方法。在指向运动的场景中，研究人员取得了重大成果：[3] 基于 Thom 的可横向性理论（transversability theory），代价的结构可从实验数据强调的定性属性中推导得出。例如，这些可能是运动中肌肉无活动时的时间段的特征。这种定性方法还被成功应用于人类运动。[6]

我们已经从机器人控制的仿生方法的角度介绍了逆最优控制问题问题是：如何综合自然运动法则，从中推导出用于机器人的最优控制模型？我们强调了逆多项式最优化中出现的最新发展。我们应该注意到，本文远不能涵盖逆最优控制的所有方法。逆最优控制也是机器学习中的活跃研究领域。在强化学习的场景中，[16,20] 逆强化学习构成了基于马尔可夫决策过程的另一个求解范式，在挑战性的问题方面（如直升机控制），它取得了令人惊叹的结果。[7] 该方法库源于随机控制（见 Kober 等人的著作 [20] 及其中的参考文献）。

**计算：是实际问题，还是理论问题？**

在计算机动画中，从模仿自然的现实主义角度衡量，基于最优化的运

**图4.（顶部）使用整个身体的轨迹最优化时的两步运动[36]（由K. Mombaur提供）和（底部）基于线性化倒立摆的步行模式生成器[17]（由O. Stasse.提供[39]）。整个身体的最优化能让机器人达到更高的性能，但是数值解决方案仍然太慢，无法让人得到有效的控制器。**



动生成给出了良好的结果。例如，可以使用数值最优化来为仿人制品模仿出真实感很强的每一次步行，跨越或跑步的动作。这些复杂的身体结构包括多达 12 个体节和 25 个自由度。[36] 初看上去，该方法可先验地应用于类人机器人。图4（顶部）提供了一个例子，描述了 HRP2 跨越非常大的障碍物的方式。

然而，机器人学设定了在虚拟世界中不存在的物理约束，对计算的性能也有要求。双足步行是一个典型的例子，此时技术的局限暗示人们需要寻找其他的公式。此时的瓶颈是控制算法满足实时约束的能力。

在当前基于模型的仿真实验中，计算的时间以分钟来衡量。但是分钟不是能与实时相比的时间尺度。例如，计算时间需要达到几毫秒的上界，以保证站立的类人机器人的稳定性。所以，利用通用的最优化技术来进行必要的实时控制时，需要构建简化的模型或者开发专用方法。这一问题构成了一个活跃的研究方向，该方向结合了机器人控制和数值最优化。

类人机器人的步行运动生成研究给出了一个例子。最流行的步行模式生成器基于拟人身体的简化模型：线性化倒立摆模型。它由 Kajita 等人提出[17]，并被开发用于 HRP2 类人机器人。该方法基于两大假设：（1）第一个假设通过设定恒定高度的质心简化了控制模型；（2）第二个假设假定了足迹的知识。假设（1）的优势在于把最初的非线性问题变换成了线性问题。对应的模型是低维度模型，而且可以通过一个最优化公式来处理（1）。[8] 有了这个公式后，不再需要假设（2）。这样一来，该方法产生了一个即时的步行运动生成器，其自动确定脚步位置。它通过线性模型预测控制（同 Kajita 等人阐述的原始控制回路相比，与该控制相关的二次规划所支持的控制回路要快得多）。[17] 实际上，使用最新的求解器后，运行完整的二次规划耗时不到 1 毫秒。不仅如此，在特定的场景中，可以设计一种优化算法，把求解的计算时间缩短 100 倍。[8]

图 4（底部）给出了该方法的例子，它让真实的 HRP2 跨过了一个障碍。基于模型降阶的方法支持实时控制机器人。然而，降阶后的模型无法完全使用机器人动力学。与对机器人整个身体的轨迹进行优化相比，生成的运动不是最优的。因此，无法达到相同的性能（在这种情况下为相同的障碍物高度）：整个身体的最优化能让机器人达到更高的性能，但是只能是非即时的。如果即时情况下也要达到相同的性能，则需要更强大的计算机（运行相同的算法）或更精巧的算法。

## 结语

机器人运动最优化的概念繁多，有各种各样的定义和应用域。本文的目标之一是总结遍布在下列多个领域内中的若干观点和参考：机器人学、控制、微分几何、数值最优化、机器学习，甚至是神经生理学。

该目标强调了最优运动在机器人动作建模中的表达能力，并提出了当前在复杂机器人（如类人机器人）的实时控制中使用数值最优化时面临的挑战。第二个目标是报告逆最优控制的最新问题。虽然在机器学习中它的随机公式相当流行，但如今在微分几何控制理论和多项式最优化中也涌现了其他的范式。

如本文的姊妹篇中证明的那样，[24] 机器人学为最优控制理论提供了丰富的基准。由于有效应用时存在实时的计算约束，机器人学还对数值最优化提出了挑战。机器人学家的难点是在一般性和特殊性之间找出合适的折衷。通用的算法受到了经典的维度诅咒的困扰，它构成了机器人控制的瓶颈。因此，可把它们用于非即时的运动生成，但是它们不足以满足实时应用的需要。机器人实时控制需要非常快的计算。它需要专用的数值最优化方法。在双足步行的例子中，我们已经看到了一般性和特殊性之间的紧张局面。如今，机器人学家向最优化理论家寻求更有效的算法，同时他们也在为此开发特定的专有技术。

最后但同样重要的是，让我们用神经生理学家 K. Friston 提出的争议作为结尾。在最近的论文中，[11] 他提出了带有挑战性质的问题："理解运行行为时，最优控制理论是否有用，它是否一种误导？"他反对最优控制，支持与其竞争的主动推理概念。虽然该论文的主要篇幅放在生命科学的运动控制上，但对机器人学家而言，这一问题相当关键，利益攸关，它要求生命和工程科学加强合作。

## 鸣谢

本文受益于 Quang Cuong Pham 的指点和 Joel Chavas 的细心审阅。最重要的是，本文得到了高质量的评审。本文的部分资助来源于 ERC（欧洲研究委员会） Grant 340050 Actanthrope、Jacques Hadamard 数学基金会最优化和运筹学研究 Gaspar Monge 项目【Gaspar Monge Program for Optimization and Operations Research of the Fondation Mathématique Jacques Hadamard (FMJH)】以及 ANR 13-CORD-002-01 Entracte。

参考资料

1. Baerlocher, P. and Boulic, R. An inverse kinematic architecture enforcing an arbitrary number of strict priority levels.*The Visual Computer 6*, 20 (2004), 402–417.
2. Ben-Israel, A. and Greville, T. Generalized inverses:Theory and applications.*MS Books in Mathematics.* Springer, 2nd edition, 2003.
3. Berret, B. et al.The inactivation principle:Mathematical solutions minimizing the absolute work and biological implications for the planning of arm movements.*PLoS Comput Biol.4*, 10 (2008), e1000194.
4. Chaumette, F. and Hutchinson, S. Visual servo control, Part I:Basic approaches.*IEEE Robotics and Automation Magazine 13*, 4 (2006), 82–90.
5. Chiaverini, S. Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators.*IEEE Trans. on Robotics and Automation 13*, 3 (1997), 398–410.
6. Chitour, Y., Jean, F. and Mason, P. Optimal control models of goal-oriented human locomotion.*SIAM J. Control and Optimization 50* (2012), 147–170.
7. Coates, A., Abbeel, P. and Ng, A. Apprenticeship learning for helicopter control.*Commun.ACM 52*, 7 (July 2009), 97–105.
8. Dimitrov, D., Wieber, P.-B., Stasse, O., Ferreau, H. and Diedam, H. An optimized linear model predictive control solver.*Recent Advances in Optimization and its Applications in Engineering.* Springer, 2010, 309–318.
9. Escande, A., Mansard, N. and Wieber, P.-B.Hierarchical quadratic programming.*The Intern.J. Robotics Research 33*, 7 (2014), 1006–1028.
10. Freeman, R. and Kokotovic, P. Inverse optimality in robust stabilization.*SIAM J. Control Optim.34* (1996), 1365–1391.
11. Friston, K. What is optimal about motor control?*Neuron 72*, 3 (2011), 488–498.
12. Hak, S., Mansard, N., Stasse, O. and Laumond, J.-P. Reverse control for humanoid robot task recognition. *IEEE Trans.Sys.Man Cybernetics 42*, 6 (2012), 1524–1537.
13. Harris, C. and Pike, J.3d positional integration from image sequences.*Image and Vision Computing 6*, 2 (1988), 87–90.
14. Heuberger, C. Inverse combinatorial optimization:A survey on problems, methods and results.*J. Comb. Optim.8* (2004), 329–361.
15. H.Hicheur, H., Pham, Q., Arechavaleta, G., Laumond, J.-P. and Berthoz, A. The formation of trajectories during goal-oriented locomotion in humans.Part I:A stereotyped behaviour.*European J. Neuroscience 26*, 8 (2007), 2376–2390.
16. Kaelbling, L., Littman, M. and Moore, A. Reinforcement learning:A survey.*J. Artificial Intelligence Research 4* (1996), 237–285.
17. Kajita, S. et al.Biped walking pattern generation by using preview control of zero-moment point.In *Proceedings of the IEEE Int.Conf. on Robotics and Automation* (2003), 1620–1626.
18. Khatib, O. Real-time obstacle avoidance for manipulators and mobile robots.*The Intern.J. Robotics Research 5*, 1 (1986), 90–98.
19. Khatib, O. A unified approach for motion and force control of robot manipulators:The operational space formulation.*The Intern.J. Robotics Research 3*, 1 (1987), 43–53.
20. Kober, J. Bagnell, J. and Peters, J. Reinforcement learning in robotics:A survey.*The Intern.J. Robotics Research 32*, 11 (Sept. 2013).
21. Lasserre, J.B.*Moments, Positive Polynomials and Their Applications.* Imperial College Press, London, 2010.
22. Lasserre, J.B.Inverse polynomial optimization.*Math. Oper.Res.38*, 3 (Aug. 2013), 418–436.
23. Lasserre, J.B. and Anjos, M., eds.*Semidefinite, Conic and Polynomial Optimization.*Springer, 2011.
24. Laumond, J.-P., Mansard, N. and Lasserre.J.B.Optimality in robot motion (1):Optimality versus optimized motion. *Commun.ACM 57*, 9 (Sept. 2014), 82–89.
25. Liégeois, A. Automatic supervisory control of the configuration and behavior of multibody mechanisms. *IEEE Trans.Systems, Man and Cybernetic 7* (1977), 868–871.
26. Mansard, N. and Chaumette, F. Task sequencing for sensor-based control.*IEEE Trans. on Robotics 23*, 1 (2008), 60–72.
27. Marchand, E. and Hager, G. Dynamic sensor planning in visual servoing.In *Proceedings of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems* (1998), 1988–1993.
28. Mombaur, K., Truong, A. and Laumond, J.-P. From human to humanoid locomotion:An inverse optimal control approach.*Autonomous Robots 28*, 3 (2010).
29. Mordecai, K. On the inverse optimal problem:Mathematical systems theory and economics, I, II. *Lecture Notes in Oper.Res. and Math.Economics 11*, 12 (1969).
30. Nakamura, Y., Hanafusa, H. and Yoshikawa, T. Task-priority based redundancy control of robot manipulators.*The Intern.J. Robotics Research 6*, 2 (1987), 3–15.
31. Paul, R. *Robot Manipulators:Mathematics, Programming, and Control.* MIT Press, Cambridge, MA, 1st edition, 1982.
32. Poincaré, H. On the foundations of geometry (1898). *From Kant to Hilbert:A Source Book in the Foundations of Mathematics.* W. Ewald, Ed. Oxford University Press, 1996.
33. Rosen, J. The gradient projection method for nonlinear programmimg.Part II, Nonlinear constraints.*SIAM J. Applied Mathematics 9*, 4 (1961), 514–532.
34. Samson, C., Borgne, M.L. and Espiau, B. *Robot Control:The Task Function Approach.* Clarendon Press, 1991.
35. Schaal, S., Mohajerian, P. and Ijspeert, A. Dynamics system vs. optimal control:A unifying view.*Prog.Brain Research 165* (2007), 425–445.
36. Schultz, G. and Mombaur, K. Modeling and optimal control of human-like running.*IEEE/ASME Trans. Mechatronics 15*, 5 (2010), 783–792.
37. Siciliano, B., Sciavicco, L. Villani, L. and Oriolo, G. *Robotics:Modeling, Planning and Control.* Springer, 2009.
38. Siciliano, B. and Slotine, J.-J.A general framework for managing multiple tasks in highly redundant robotic systems.In *Proceedings of the IEEE Int.Conf. on Advanced Robot*, 1991.
39. Stasse, O., Verrelst, B., Vanderborght, B. and Yokoi, K. Strategies for humanoid robots to dynamically walk over large obstacles.*IEEE Trans.Robotics 25* (2009), 960–967.
40. Todorov, E. Optimality principles in sensorimotor control.*Nature Neuroscience 7*, 9 (2004), 905–915.
41. Whitney, D. Resolved motion rate control of manipulators and human prostheses.*IEEE Trans.Man-Machine Systems 10*, 2 (1969), 47–53.

Jean-Paul Laumond（jpl@laas.fr）是法国图卢兹国家系统分析与系统结构实验室（LAAS）的CNRS研究主任。

Nicolas Mansard（nmansard@laas.fr）是法国图卢兹分析与架构实验室（LAAS）的CNRS 研究员。

Jean Bernard Lasserre（lasserre@laas.fr）是法国图卢兹国家系统分析与系统结构实验室（LAAS）的CNRS研究主任。

译文责任编辑：李洪波

在本通讯独家视频中，您可以观看作者对本研究的讨论。

DOI:10.1145/2742908

如需查阅相关论文，
请访问 /10.1145/2742910

# 技术视角
# 编写多核计算机程序

作者：James Larus

**编写**并行计算机程序的最佳方式是什么？常见的答案是让编译器将顺序程序转换为并行程序，或者利用并行语言或库编写并行程序。

在并行计算机的早期岁月中，并行化编译器呈现了一种撩人心弦的前景，实现在新兴的并行计算机上运行未经修改的“沾满灰尘”的顺序 FORTRAN 程序。尽管对这些编译器的研究造就了现代编译器中采用的许多程序分析和表示法创新，但出现的工具并不能成功地并行化大多数应用程序，而开发人员则将目光转向了 pthread 和 MPI 等库。

在这一方法中，程序采用并行构造；无论是 fork-join 等显式并行运算，还是 map 和 reduce 等隐式并行运算。这些理论中的抽象概念应当能激励开发人员“并行”思维，编写并行程序；但实际上，即使有了这些概念，编写并行程序依然颇具挑战性，因为存在数据竞争等新错误类型，而且并行机器繁杂多样（如消息传递、共享内存和 SIMD 等）。

那么，开发人员能够做些什么来提高其代码在具有多个内核和向量处理单元的现代并行微处理器上的性能呢？下面这篇论文提倡在开发人员和编译器之间进行有益的分工，通过人工改造代码和数据结构并强制并行执行一些循环，从而增加编译器生成和优化并行机器代码的机会。

这当中的结果令人惊叹。对于 11 个计算密集型内核，采用这种方式开发的代码的性能达到了最佳手工优化代码的 30% 以内，而且不需要开发人员使用低级编程结构，也无需其了解机器的架构和指令集。

不过，这样的分工有什么必要性？为何编译器无法并行化和向量化这些（相对简单的）程序？作者们把矛头暗暗指向“依赖性分析、内存别名分析和控制流分析等难题”。在现实中，编译器运用了大量的局部优化，每一种优化分别都能改进代码的一小部分。能够改变程序对其结果的计算方式的大而普遍的改造则不在传统编译器的范畴之内。在最近对程序合成的研究之前，对探索可能变革的大型空间的有效技巧的研究少之又少。此外，即便是局部优化，编译器被保守的程序分析拖了后腿，后者最多只是估计程序的潜在行为，[a] 而且必须禁止可能对程序结果造成负面影响的优化。

a 许多程序分析若能完全精确，就能实现图灵停机问题的解决方案。

**此篇论文认为，当重构和注解由开发人员执行时应并不具有难度，应当成为每一位程序员为现代计算机编程时的一项必要任务。**

此篇论文认为，当改造和标注由开发人员执行时并不具有难度，应当成为每一位程序员为现代计算机编程时的一项必要任务。这些变化包括将结构体的数组转变为数组的结构体，通过对循环分块来改善数据重用，注解并行循环，以及运用更多并行算法。从概念上讲，这些变化都不难理解，尽管找到一种新算法或许存在难度。然而，这些改变可为程序带来错误，也可能比较复杂而难以应用到数据结构或许由许多例程共享的大型应用程序中。

当然，程序优化总体上可能会对程序结构和可读性存在类似的有害影响，所以这些顾虑并不限于并行程序。与直接编写正确的、高效运行的并行程序的挑战相权衡，改造和注解似乎是生成可维护程序的合理方法。不过，假如生成的程序无法显著提高运行速度，这一方法也就几无价值。

该论文的主要贡献为展示人工与编译器之间的分工能够达到其利用硬件并行度提高性能的目标。成熟的现代编译器在重构和注解的辅助下，可以生成非常高效的并行代码。无论是编译器还是人工，都不能独自出色地实现这一目标。 C

James Larus 是瑞士洛桑 EPFL 的计算机与通信科学系的教授兼主任。

译文责任编辑：翟季冬

DOI:10.1145/2742910

# 传统编程如何填补并行计算应用程序的忍者性能差距?

作者：Nadathur Satish、Changkyu Kim*、Jatin Chhugani*、Hideki Saito、Rakesh Krishnaiyer、Mikhail Smelyanskiy、Milind Girkar 和 Pradeep Dubey

**摘要**

**当前的处理器倾向于将多个核心与单指令多数据 (SIMD) 单元以及更深又复杂的内存层次结构相整合，使得从应用程序中抽取性能变得愈加困难。一部分人认为，传统的编程方法不适用于这些现代的处理器，所以必须设计激进的新语言。在本论文中，我们对这种想法提出了质疑，通过支持传统的编程方法和性能与编程工作效用比的证据，展示多核处理器和未来众核架构为常用并行计算工作负载提供大幅速度提升和接近于最佳的性能。**

**我们首先量化“*忍者差距*”(Ninja gap) 的范围，这是编写幼稚的 C/C++ 代码（无并行感知能力，通常为串行）和最佳优化的代码在现代多核 / 众核处理器上的性能差距。通过一组代表性的吞吐量计算型基准测试，我们展示出对于 6 核 Intel® Core™ i7 X980 Westmere CPU 平均*忍者差距* 为 24X（最多 53X）；而且，如果不加以解决的话，此差距无疑还会增大。我们也展示出，一组知名的算法更改与现代编译器技术进步的结合如何能将平均忍者差距缩小到仅仅 1.3X。与生成忍者代码所需的大量投入相比，这些更改通常仅需少量的编程工作。我们也展示了即将到来的 Intel® Xeon Phi™ 架构上同样喜人的结果，这一架构拥有更多核心、更宽的 SIMD。因而表明，我们能够控制原本无法操控的忍者差距的扩大，而且能对未来的架构提供更加稳定的可预测性能增长，通过坚实的证据表明并不需要激进的语言更改。**

## 1. 简介

处理器代次之间的性能扩展在过去依赖于提升时钟频率。程序员本可以随波逐流，不必通过大幅更改代码来获得代码性能的提升。然而，时钟频率扩展已经遇到了功率壁垒 [16]，程序员的免费午餐已成历史。

旨在提升处理器性能的最近技术侧重于整合更多核心和更宽的单指令多数据 (SIMD) 单元，同时使内存层次结构更深、更复杂。虽然最新处理器上的峰值计算和内存带宽一直在提高，从这些平台中获取性能的难度却越来越高。这造成了这样的局面，只有少数的专家级程序员（“忍者程序员”）能够完整驾驭现代多核 / 众核处理器的能力，而资质平平的程序员只能获得这一性能的一小部分。我们将“忍者差距”(Ninja gap) 这一术语定义为编写幼稚的 C/C++ 代码（无并行感知能力，通常为串行）和最佳优化的代码在现代多核 / 众核处理器上的性能差距。

近期发表的许多论文 [8, 14, 17, 24] 表明，通过优化特定于平台的并行实施可以实现 10–100X 的性能提升，而这证实了巨大忍者差距的存在。这通常要求大量的编程工作，可能也必须针对各代处理器重新进行优化。然而，这些论文都没有对此类优化中涉及的工作发表看法。在本篇论文中，我们将量化忍者差距的范围，分析这一差距的原因，以及研究在使用传统的 C/C++ 编程语言时少量的工作能够填补这一差距的多少部分。[a]

我们首先来量化忍者差距的范围。我们使用一组真实应用程序，它们要求很高的吞吐量（因而天生有大量并行可被利用）。我们选择吞吐量型应用程序的原因在于，它们组成了应用程序中一个愈加重要的类别 [7]，而且它们也为利用体系结构资源提供了最大的机会 — 如果幼稚代码没有充分利用这些资源，就会造成巨大忍者差距。我们在不同代次的各种平台上利用基准测试测量性能：2 核 Conroe、4 核 Nehalem、6 核

* 本研究进行期间这些作者在 Intel 就职。

[a] 由于对编程容易度的衡量（如编程时间或代码行数）都非常主观，我们通过获得性能所需的代码更改来演示代码片段。

本文的原始版本发表于 *Proceedings of the 39th Annual International Symposium on Computer Architecture*（2012 年 6 月）。IEEE Computer Society, Washington, D.C., 440–451.

Westmere、Intel® Xeon Phi™ [b]，以及 NVIDIA C2050 GPU。图 1 显示了我们基准测试在以下三种 CPU 平台上的忍者差距：2.4 GHz 2 核 E6600 Conroe、3.33 GHz 4 核 Core i7 975 Nehalem，以及 3.33 GHz 6 核 Core i7 X980 Westmere。该图显示幼稚的 C/C++ 代码和最佳优化的代码在最新 6 核 Westmere CPU 上的性能差距高达 53X。图中还显示，此差距在处理器代次之间也有增大 — 2 核 Conroe 系统上为 5 - 20X（平均 7X），Westmere 上则为 20–53X（平均 25X），尽管微架构增强已经缓减了执行各种优化的需求和影响。

我们接下来分析巨大性能差距的来源。幼稚代码性能欠佳的原因有许多。首先，代码可能无法并行化，编译器也不能自动识别并行区域。这意味着，幼稚代码没有利用到核心数量的增加，而优化的代码则对此进行了充分利用。其次，代码可能无法向量化，因而导致缺乏对 SIMD 宽度增加的利用。虽然对自动向量化的研究已有很长时间，但存在诸如依赖性分析、内存别名分析和控制流分析等诸多难题，它们阻碍了编译器向量化外层循环、带有 gather 的循环（不规则内存访问），甚至最内层循环（依赖性和别名分析失败的位置）。存在巨大性能差距的第三个原因可能在于代码受到内存带宽的限制 — 例如，如果代码没有针对缓存层次结构而分块，就可能会出现这种情况 — 导致缓存未命中。

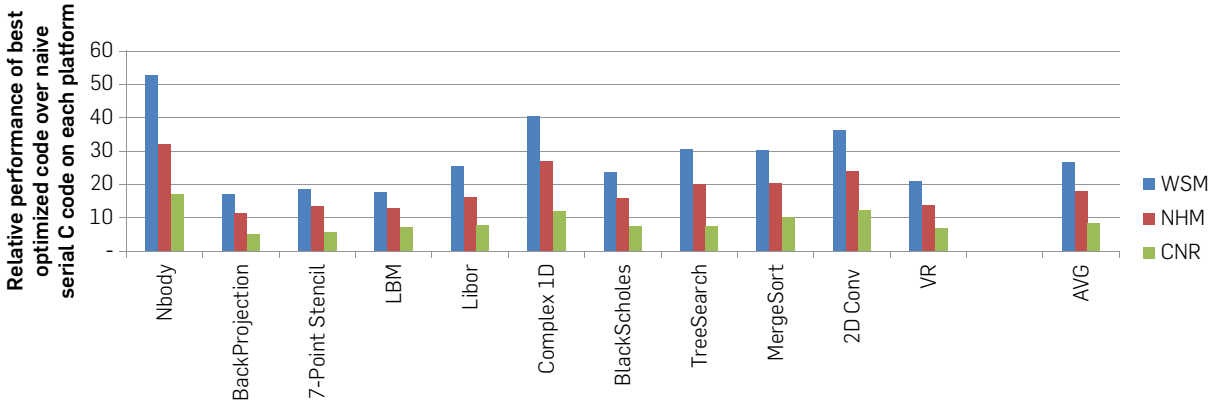
我们对忍者程序员编写的代码进行了分析，结果表明这些程序员投入了大量精力，使用 pthread 等线程技术和用于向量化的低级 instrinsics 函数集来获得性能。这可能会导致代码非常复杂，尤其是向量化不规则循环时。我们的研究表明，可以利用最新的编译器技术，通过相对较少的程序员工作实现代码的并行化和向量化。通过将 OpenMP 编译指导语句标注要并行化的循环来实现并行化，程序员也可避免编写复杂的多线程代码。对于没有依赖项的简单循环，也可以实现自动循环并行化 — 本文中我们假设使用编译指导语句。对于向量化，Intel® Composer XE 2011 版等最新的编译器引入了对编译指导语句的使用，程序员可以通过规避执行依赖性分析和别名分析的需求来强制循环向量化。此版本的编译器也能够向量化外层循环，而 Intel® Cilk™ Plus 功能 [11] 可以帮助程序员使用这一新功能（如果没有自动触发）。[c] 这些功能允许程序员抛弃对更低级 intrinsics 函数集和 / 或汇编代码的使用，并且大幅提高性能。我们发现，在使用这些功能时忍者差距**在 Westmere 上缩小到平均 2.95X**。剩余的差距或许来自代码中的带宽瓶颈，又或者源自代码因为不规则内存访问而导致的部分向量化。然而，虽然遗留的差距相对较小，但随着 SIMD 宽度的增加以及带宽与计算比的减小，在未来的架构上不可避免地会增大。

为填补遗留的差距，就需要程序员的干预。当前的编译器不会在算法级别上自动进行涉及内存布局变化的更改，这些必须由程序员手动完成。我们确定并建议以下三种重要的算法更改：为缓存**分块**、对带宽 /SIMD 友好的**数据布局**，以及在某些情形中选用**对 SIMD 友好的替代算法**。算法更改中一个重要的类别涉及对数据结构进行分块以装入缓存中，从而减轻内存带宽压力。另一类更改则涉及剔除对内存 gather/scatter 运算的使用。此类不规则内存运算不仅会增加延迟和带宽用量，还会限制编译器向量化的范围。

[b] Intel、Xeon 和 Intel Xeon Phi 是 Intel Corporation 在美国和/或其他国家/地区的商标。

[c] 有关编译器优化的更多完整信息，请参阅以下网址上的优化声明：http://software.intel.com/en-us/articles/optimization-notice/。

**图 1：2 核 Conroe (CNR)、4 核 Nehalem (NHM) 和 6 核 Westmere (WSM) 系统上幼稚的串行 C/C++ 代码与最佳优化代码相比性能差距不断变大**



常见的数据布局更改是将以结构数组 (AOS) 表示法编写的数据结构转换为数组结构 (SOA) 表示法。这有助于防止在访问跨越数组元素的一个结构区域时出现 gather，也有助于编译器向量化在数组上迭代的循环。最后，在某些情形中，由于存在循环迭代之间的背对背依赖，代码无法被向量化；在这些情形中，或许需要选用其他对 SIMD 友好的算法。

执行算法上的更改确实需要程序员的投入和见解，我们希望并行编程方面的教育和培训能够发挥较大的作用，帮助程序员开发出更优秀的并行算法。回报是巨大的 — 我们的研究表明，**在执行了算法更改后**，最佳优化的代码和编译器生成的代码之间**平均性能差距仅仅为 1.3X**。此外，这一投入也可以在不同的处理器代次之间，以及 GPU 等不同计算平台之间分摊。由于底层硬件的发展趋势是增加核心数和 SIMD 宽度，并对逐渐增加的带宽进行优化，未来的架构上依然会存在**较小的可预测**性能差距。我们在最新的 86X 型众核平台 **Intel® Xeon Phi™ Knights Corner 协处理器架构**重复了我们的实验，对此进行了展示。结果表明，**忍者差距几乎相同 (1.2X)**。实际上，硬件 gather 支持的增加让可编程性对至少一种基准测试而言变得更加轻松。算法更改和现代编译器技术的结合是向前迈出的重要一步，使得程序员能够利用传统的编程顺应并行处理的趋势。

## 2. 基准测试说明

在我们的研究中，我们分析了近期推出的基准测试套件 [2, 3, 5] 的计算和内存特征，从**吞吐量计算型应用程序**中选择了代表性的一组。吞吐量型工作负载负责在给定的时间内处理大量的数据，因而处理的所有数据都需要快速响应时间。它们包括来自高性能计算、金融服务、图像处理和数据库等领域的工作负载。[5] 吞吐量计算型应用程序拥有许多数据级和线程级的并行，因而被确定为未来应用程序中在*计算*和*内存特征*上影响当前和未来多核 / 众核处理器设计的一大重要类别。它们也为利用架构资源提供了最大的机会 — 如果幼稚代码没有充分利用不断增加的计算资源，就会造成巨大的忍者差距。我们论证了下文介绍的一组代表性基准测试，它们涵盖了吞吐量计算的**这一广阔应用程序领域**。我们采集了关键的计算内核，其中大部分时间花在了吞吐量计算型应用程序上。因此，减小我们基准测试中的忍者差距也就能转化到应用程序本身上。

**1.NBody：** NBody 计算在许多科学应用程序中使用，如天体物理学和统计学习算法等领域。[1] 对于给定的 $\mathcal{N}$ 个 body，基本的计算为 $O(\mathcal{N}^2)$ 运算，它对这些 body 有两个循环，计算并累积它们之间的成对作用。

**2.BackProjection：** BackProjection 是执行 CT 数据锥束重建时常用的内核。[13] 将一组 2D 图像 “反投影” 到一个 3D 体积上，从而构建密度值网格。对于每个输入图像，各 3D 网格点投影到图像上，相邻的 2 × 2 像素的密度进行双向线性插值并累积。

**3.7 点模板：** 模板计算运用于广泛的科学学科。[8] 这种计算涉及对空间输入 3D 点格网的多次扫描，每次扫描计算各个网格点和其邻居的加权和，并将计算值存储到输出网格中。

**4. 晶格波兹曼法 (LBM)：** LBM 是一个计算流体力学类别，能够对复杂的流体问题进行建模。[25] 它模拟粒子分布函数在许多时间步中对 3D 晶格的演变。在每个时间步上，对每个网格点执行的计算涉及该网格点及其面 (6) 和边 (12) 邻居的方向密度值（也称为 D3Q19）。

**5.LIBOR Monte Carlo：** LIBOR 市场模型 [4] 用于互换期权组合的定价。它将一组期货汇率作为对数正态分布进行建模。典型的 Monte Carlo 方法将为此分布生成随机的样本，然后使用的大量的独立路径计算衍生价格。

**6.1D 复数卷积：** 这在图像处理和雷达跟踪等应用领域广泛使用。它通过一个大型复数滤波器对 1D 复数图像执行 1D 卷积。

**7.BlackScholes：** Black-Scholes 模型为期权价格演变提供了一种偏微分方程 (PDE)。对于欧式期权，PDE 的求解为闭型表达式。[20] 这涉及一组数学运算，即幂、对数、平方根和除法的计算。

**8.TreeSearch：** 内存中树结构索引搜索常用于商业数据库。[14] 此应用程序通过不同的查询对树进行多个并行搜索，每个查询根据各个树级别上节点值的比较结果对树进行路径跟踪。

**9.MergeSort：** MergeSort 常用于数据库等领域，[6] 也是未来架构选用的排序算法。[22] 它利用完整数组的 $\log\mathcal{N}$ 个 *merge pass* 对由 $\mathcal{N}$ 个元素组成的数组进行排序。

**10.5 × 5 2D 卷积：** 卷积是用于模糊和浮雕等特效的常用图像过滤运算。[15] 对于给定的 2D 图像和 5 × 5 空间滤波器，各个像素计算和存储 5 × 5 相邻像素的加权和，其中权重为滤波器中的对应值。

**11. 立体渲染：** 立体渲染通常用于医学成像等领域。[24] 给定 3D 网格和 2D 图像位置时，该基准测试生成垂直于图像平面并穿过 3D 网格的光线，累积颜色和不透明性以计算图像的各个像素的最终颜色。

**忍者性能：** 表 1 提供了各个基准测试的代表性数据集大小的详细信息。各自都有对应的性能最佳代码，前文中已阐述过我们研究中不同的平台上的性能数据 [d]。为了进行公平的比较，我们**实施并且激进地优化了（包括使用 intrinsics 函数集 / 汇编代码）这些基准测试，**获得了与对应平台上最佳报告数据的*可比较性能*。此代码而后在我们的平台上执行，从而获取我们在此

论文中使用的对应最佳优化性能数据。表 1（第 3 列）显示所有基准测试在 Intel® Core ™ i7 X980 上的最佳优化（*忍者*）性能。在本文的后续部分中，**忍者性能均指在我们平台上执行此代码时获得的性能数据**。

### 3. 填补忍者差距

在这一章中，我们取第 2 章中所述的各个基准测试，尝试通过少量编程工作开始对编写幼稚的代码填补忍者差距。如需详细的性能分析，建议读者阅读我们的 ISCA 论文。[23]

**平台：**我们在 3.3 GHz 6 核 Intel® Core ™ i7 X980（代号 Westmere，峰值计算：158GFlops，峰值带宽：30 GBps）测量了性能。这些核心具有乱序的超标量微架构，以及 2-way 同步多线程 (SMT)。它还具有支持广泛指令的 4-wide SIMD 单元。每个核心都有 32 KB L1 和 256 KB L2 缓存。这些核心共享 12 MB 终极缓存 (LLC)。我们的系统配有 12 GB RAM，运行 SuSE Enterprise Linux (ver.11)。我们采用了 Intel® Composer XE 2011 编译器。

**方法学：**对于每个基准测试，我们尝试首先通过指令和数据级并行获得良好的单线程性能。为利用数据级并行，我们在启用 / 禁用（-no-vec 编译器旗标）自动向量化的前提下运行代码，测量各个基准测试的 SIMD 扩展。如果 SIMD 扩展没有接近于峰值，我们通过分析生成的代码来确定架构瓶颈。然后，我们通过添加 OpenMP 编译指导语句并行化基准测试来获得线程级并行，再评估线程扩展 — 再次评估瓶颈。最后，我们进行必要的算法更改，以克服瓶颈。

**使用的编译器杂注：**我们使用 OpenMP 以实现线程级并行，同时使用自动向量化器或最新的技术（如作为 Intel® Cilk ™ Plus 一部分推出的数组标记，此后称为数组标记）来实现数据级并行。关于特定编译器技巧的详细信息可参见 Tian 等人的文章。[26] 我们向代码和命令行添加的编译器指令如下所示：

- ILP 优化：我们在最内层循环之前使用 **#pragma unroll** 指令，并在外层循环的外部使用 **#pragma unroll_and_jam** 原语。两者均可选地接受循环要展开的次数。
- 最内层循环层面的向量化：如果自动向量化失败，则程序员可以利用 **#pragma simd** 强制向量化。这是 Cilk Plus 中最新推出的功能。[11]
- 外层循环层面的向量化：这可以通过两种不同的方式来实现：(1) 直接在外层循环层面向量化；以及 (2) 剥离外层循环迭代，并更改循环体内的各条语句以操作剥离。在本研究中，我们通过数组标记使用第二种方法。
- 并行化：我们使用 OpenMP **#pragma omp** 并行化循环。我们通常通过 **#pragma omp parallel for** 构造对外层循环使用此方法。
- 快速数学：根据精度需求，选择性地将 **-fimf-precision** 编译选项用于我们的基准测试。

**1.NBody：**我们对包含 100 万个 body 的数据集（16 MB 内存）实施了 Nbody。图 2 显示了各种优化的细分情况。代码包含对所有数对迭代的两个循环。我们首先执行了展开优化来改进 ILP，这提供了 1.4X 的增益。编译器对代码的**自动向量化**效果不错，无需程序员干预，而且提供了 3.7X SIMD 扩展。我们获得了 3.1X 的并行扩展，这激发了我们算法优化中对数据结构进行分块以装入 L3 缓存（**1-D 分块**，代码见第 4.1 节）的需求。执行分块后，我们又获得了 1.9X 线程扩展，编译后代码和最佳优化代码之间的性能差距为 1.1X。

**2.BackProjection：**我们将尺寸为 2048 × 2048 像素的 500 幅图像反投影到 1024 × 1024 × 1024 的 3D 网格上。反投影需要每个网格 80 次运算。图像 (16 MB) 和立体 (4 GB) 都太大，无法装入缓存中。图 2 表明，我们从自动向量化获得的 1.2X 的 SIMD 扩展不甚理想。此外，并行扩展仅仅为 1.8X。这是因为代码**受带宽限制**（1.6 字节 /flop）。我们对 3D 立体执行分块来减小带宽（图 2 中的 **3D 分块**）。由于空间局部性，图像工作集也会相应地减小。这导致代码变为**受计算限制**。然而，由于 gather 无法在 CPU 上向量化，SIMD 扩展仅仅额外增大了 1.6X（总计 1.8X）。我们额外获得了 4.4X 线程扩展（总计 7.9X），展示出 SMT 的优势。净性能比最佳优化的代码差 1.1X。

**3.7 点 3D 模板：**应用程序在 3D 点网格上迭代，每个点执行 8 flop 计算。使用的 3D 数据集为 512 × 512 × 512 网格点。图 2 中显示，我们从自动向量化获得的 1.8X 的 SIMD 扩展不甚理想（**受带宽限制**）。为了改善扩展，我们执行了**空间**和**时间分块**来提高性能。[17] 生成的代码可同时执行*四个*时间步，将 DLP 进一步提高 1.7X（净 SIMD 扩展 3.1X — 由于对边界的重

**表 1：各种基准测试和所用的对应数据集，以及 Core i7 X980 上的最佳优化（忍者）性能。**

| 基准测试 | 数据集 | 最佳优化的性能 |
|---|---|---|
| NBody[1] | $10^6$ body | 7.5 × $10^9$ 对/秒 |
| BackProjection[13] | 500 图像，$1K^3$ | 1.9 × $10^9$ 投影/秒 |
| 7 点 3D 模板[17] | $512^3$ 网格 | 1.9 × $10^9$ Up./秒 |
| LBM[17] | $256^3$ 网格 | 2.3 × $10^8$ Up./秒 |
| LIBOR[10] | 10 M 路径，15 期权 | 8.2 × $10^5$ 路径/秒 |
| 1D 复数卷积[12] | 8K，1.28 M 像素 | 1.9 × $10^6$ 像素/秒 |
| BlackScholes[20] | 1 M 认购 + 认沽期权 | 8.1 × $10^8$ 期权/秒 |
| TreeSearch[14] | 100 M 查询，64 M 树 | 7.1 × $10^7$ 查询/秒 |
| MergeSort[6] | 256 M 元素 | 2.1 × $10^8$ 数据/秒 |
| 5X5 2D 卷积[15] | 2K × 2K 图像 | 2.2 × $10^9$ 像素/秒 |
| 立体渲染[24] | $512^3$ 立体 | 2.0 × $10^8$ 光线/秒 |

[d] 最佳报告数值来源于数据库、HPC 和图像成立等领域中的最新顶级信息公布。据我们所知，任何这些基准测试*都不存在*执行速度更快的代码。

复计算的开销而低于 4X）。线程扩展进一步增强了 2.5X（总计 5.3X）。净性能是最佳优化代码的 10.3% 以内。

**4. 晶格波兹曼法 (LBM)：**计算模式与模板内核类似。我们使用了 256 × 256 × 256 数据集。图 2 中显示，我们的初始代码 (SPEC CPU2006) 没有获得任何 SIMD 扩展，核心扩展则为 2.9X。没有 SIMD 扩展的原因在于 AOS 数据布局导致了 gather 运算。为提高性能，我们进行了以下**两个算法更改**。首先，我们对数据执行了 AOS 至 SOA 转换。生成的自动向量化代码将 SIMD 扩展提高到 1.65X。其次，我们执行了 3.5D 分块。生成的代码使 SIMD 扩展进一步提高 1.3X，使得净扩展达到 2.2X。生成的线程扩展进一步提高 1.95X（总计 5.7X）。但是，编译器产生了额外的 spill/fill 指令，导致了 1.4X 的性能差距。

**5.LIBOR：**LIBOR[4] 有一个外层循环覆盖 Monte Carlo 模拟的所有路径，以及一个内存循环覆盖单条路径上的期货汇率。图 3 显示来自自动向量化的性能增益仅仅为 1.5X，因为当前的**编译器仅尝试向量化内层循环**，而后者具有背对背依赖项，仅可部分向量化。为解决此问题，我们执行了算法更改，将布局从 AOS 转换为 SOA。我们使用数组标记技术表示外层循环向量化（代码见图 7b）。执行这些更改使得外层循环可以向量化，提供额外 2.5X 的 SIMD 扩展（净值为 3.8X）。性能与最佳优化代码类似。

**6.1D 复数卷积：**我们使用了包含 1280 万个点的图像，以及 8 K 的内核。图 3 中的第一个柱形表示编译器得到的展开所获得的性能，实现了 1.4X 的扩展。自动向量化器仅实现了 1.1X 的扩展。TLP 获得了 5.8X。为提高 SIMD 性能，我们执行了 AOS 至 SOA 格式的**数据重排**。结果，编译器生成了高效的 SSE 代码，性能进一步扩展了 2.9X。我们的总体性能比最佳优化代码的数值慢 大约 1.6X（编译器无法分块内核权重）。

**7.BlackScholes：**BlackScholes 将认购期权和认沽期权一起计算。总计算量为 200 次运算，而带宽则为 36 字节。图 3 显示，使用自动向量化时 SIMD 加快 1.1X。扩展较低的原因主要为 AOS 布局，它造成了 gather 运算。为提高性能，我们将*数据布局*从 AOS 改为 SOA。结果，自动向量化器生成的 SVML（短矢量数学库）代码使扩展提高了 2.7X（总计 3.0X）。净性能是最佳性能代码的 1.1X 以内。

**8.TreeSearch：**双字节树以宽度优先的方式分布。自动向量化器使 SIMD 加快了 1.4X（图 4a）。这是因为它同时运算 4 个查询，每个查询可能向下遍历一个不同的路径 — 导致 gather 运算。为提升性能，我们进行了**算法更改**，一次遍历 2 个级别（类似于 SIMD 宽度分块[14]）。然而，编译器没有生成所述的代码序列，造成了 1.55X 忍者差距。

**9.MergeSort：**我们通过排序包含 25600 万个元素的输入数组来进行分析。图 4a 表明，我们从自动向量化获得了 1.2X 的扩展。这是因为合并*四*对列表需要 gather 运算。并行扩展仅为 4.1X，因为最后几个合并阶段受到了带宽限制。为提高性能，我们进行了以下*两个***算法更改**。首先，我们利用合并网络实施了列表合并[6]（代码见第 4.1 节）。其次，为了降低带宽要求，我们将多个合并阶段一起执行。所生成代码的并行扩展进一步加快了 1.9X。所生成代码的性能是最佳优化代码的 1.3X 以内。

**10.2D 卷积：**我们通过 5 × 5 内核执行了 2 K × 2 K 图像的卷积。代码中包含四个循环。图 4b 显示我们从循环展开获得的 1.2X 增益。我们通过数组标记技术实施了两个内层循环。这启用了外层循环的向量化，通过 SIMD 宽度扩展了 3.8X。线程级并行是 6.2X。净性能为最佳优化代码的 1.3X 以内。

**11. 立体渲染：**如图 4b 中所示，我们获得了 8.7X 的 TLP 扩展（SMT 1.5X）。对于 DLP 而言，早期的编译器版本不能向量化该代码，因为存在各种控制密集

**图 2：算法更改之前和之后 NBody、BackProjection、模板和 LBM基准测试在指令 (ILP)、任务 (TLP) 和数据级并行 (DLP) 方面的忍者性能差距细分情况。算法更改涉及了分块。**

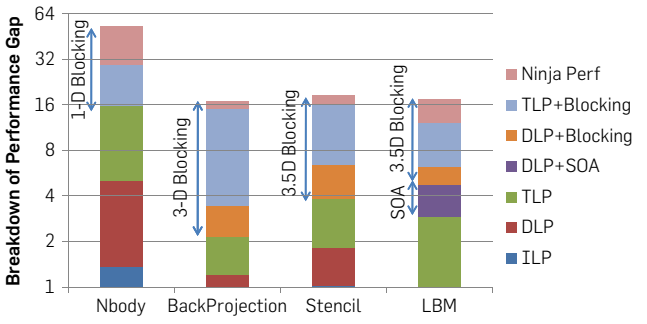
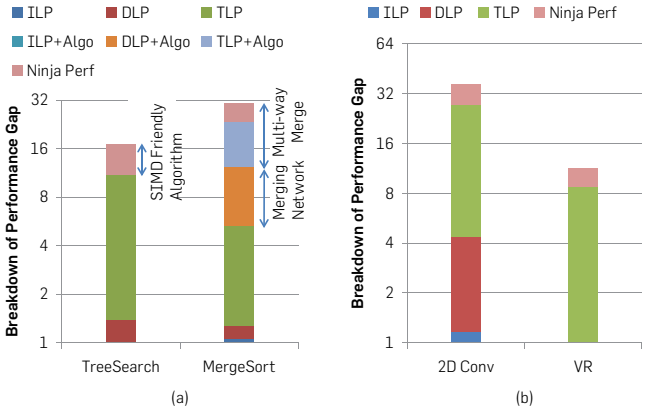



**图 3：忍者差距细分情况：(a) Treesearch 与 Mergesort；(b) 2D 卷积和 VR。(a) 中的基准测试需要重新考虑使算法对 SIMD 更加友好；(b) 中的基准测试则不需要任何算法更改。**



型语句。不过，最新的编译器可通过为各个分支指令使用掩码值，并使用正确的掩码执行各个分支的执行路径，从而对代码进行向量化。仅存在小小的 1.3X 忍者性能差距。

**总结：**在这一章中，我们分析了各个基准测试，通过运用必要的算法更改和最新的编译器技术将忍者差距缩小到 1.1–1.6X 范围以内。

## 4. 分析和总结

在这一章中，我们将确定所需的步骤，通过较少的程序员工作来填补忍者差距。在执行的关键步骤中，首先是执行一组著名的算法优化来克服扩展瓶颈，接着利用最新编译器技术来实现向量化和并行化。现在，我们从各个步骤中获得的增益角度总结我们的成果。

### 4.1 算法更改

算法更改的确需要程序员的投入和一些见解，但这是避免向量化问题和带宽瓶颈所不可或缺的。图 5 显示了我们下文中介绍的著名算法优化而带来的性能增强。

**AOS 至 SOA 转换：**在向量化的代码中避免 gather 和 scatter 的常见优化是将数据结构从结构数组 (AOS) 转换为数组结构 (SOA)。分离各个域的数组可以在执行向量化时实现连续内存访问。AOS 结构需要 gather 和 scatter，这不仅影响 SIMD 效率，也会给内存访问增加额外的带宽和延迟。存在硬件 gather/scatter 机制并不能消除这种转换的需求 — gather/scatter 访问通常需要比连续加载高得多的带宽和延迟。此类转换得到了包括 GPU 在内的多种架构的支持。[19] 图 5 显示，在我们的基准测试中 AOS 至 SOA 转换帮助提高了平均 1.4X 的性能。

**分块：**分块是众所周知的优化，有助于在许多应用程序中避免带宽瓶颈。其主要理念是通过确保数据在多次使用之间保留在缓存中（包括空间域（1-D、2-D 或 3-D）和时间域），利用应用程序中固有的数据重复使用机制。

在代码更改方面，分块涉及结合使用循环拆分和互换。图 6a 中的代码片段演示了 NBody 代码分块的示例。有两个循环（body1 和 body2）在所有 body 上迭代。顶部的*原始代码*在内层循环的整个 body 集合中流动，必须在每一次迭代时从内存中加载 body2 值。*分块后的代码*通过将 body2 循环拆分为以下两个循环而获得：在 *BLOCK* 的倍数的 body 上迭代的外层循环，以及在块内部的元素上迭代的 body22 循环。此代码在 body1 循环的多个迭代间重复使用一组 *BLOCK* 个 body2 值。如果 *BLOCK* 的选择可以确保这一组值能够装入缓存中，内存流量就可以 *BLOCK* 为系数降低。在内存方面（图 5），我们获得了平均 1.6X 的提升（LBM 和 7 点模板高达 4.3X）。

**对 SIMD 友好的算法：**在一些情形中，幼稚的算法无法轻松地向量化，原因可能在于循环迭代之间的背对背依赖，或者代码中大量使用了 gather 和 scatter。因此，可能需要另一种对 SIMD 更加友好的算法。图 6a 中的代码片段演示了 NBody 代码分块的示例。左侧代码演示传统的算法，其中一次仅合并两个元素，并且写出最少。由于数组递增运算的关系，存在背对背依赖，所以该代码无法向量化。右图显示对 SIMD 友好的合并网络的代码，[6] 它运用一个 min、max 和 interleave 运算序列合并两组 SIMD 宽度 *S* 大小的元素。此代码可以自动向量化，其中每一个突出显示的行对应于一条 SIMD 指令。不过，此代码必须要执行更多的计算（常量系数 log(S)），但依然能为 4-wide SIMD 产生 2.3X 的增益。由于这些算法更改涉及了总计算量和 SIMD 友好度之间的权衡，是否使用它们的决定应当由*程序员有意识地做出*。

**总结：**运用众所周知的算法技巧，我们可以在 6 核 Westmere 获得平均 **2.4X 的性能增益**。随着核心数、SIMD 宽度和计算量与带宽比的增加，算法更改带来的增益还会进一步扩大。

**图 4：Libor、1D 复数卷积和 BlackScholes 的忍者性能差距细分情况。所有基准测试都需要 AOS 至 SOA 转换，以获得良好的 SIMD 扩展。**

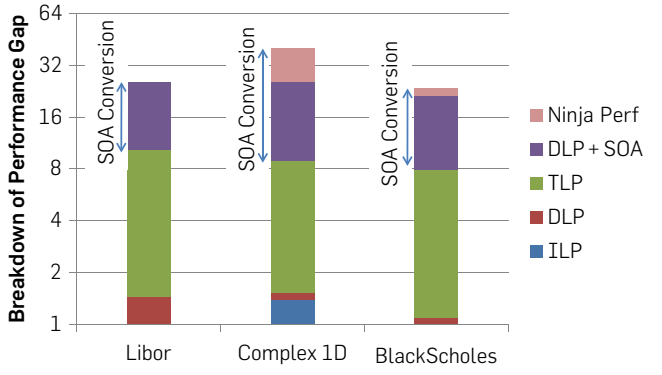



**图 5：三种不同算法更改对我们基准测试的益处，已正规化为任何算法更改之前的代码。算法更改的影响是累积的。**

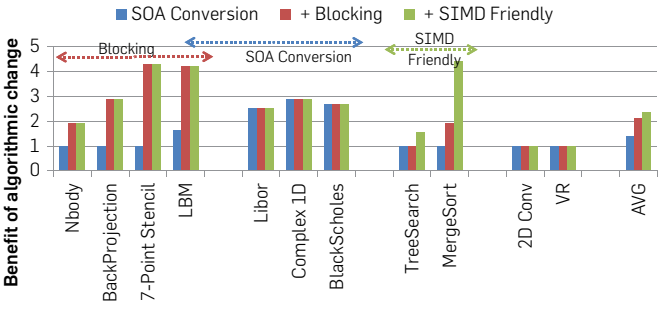
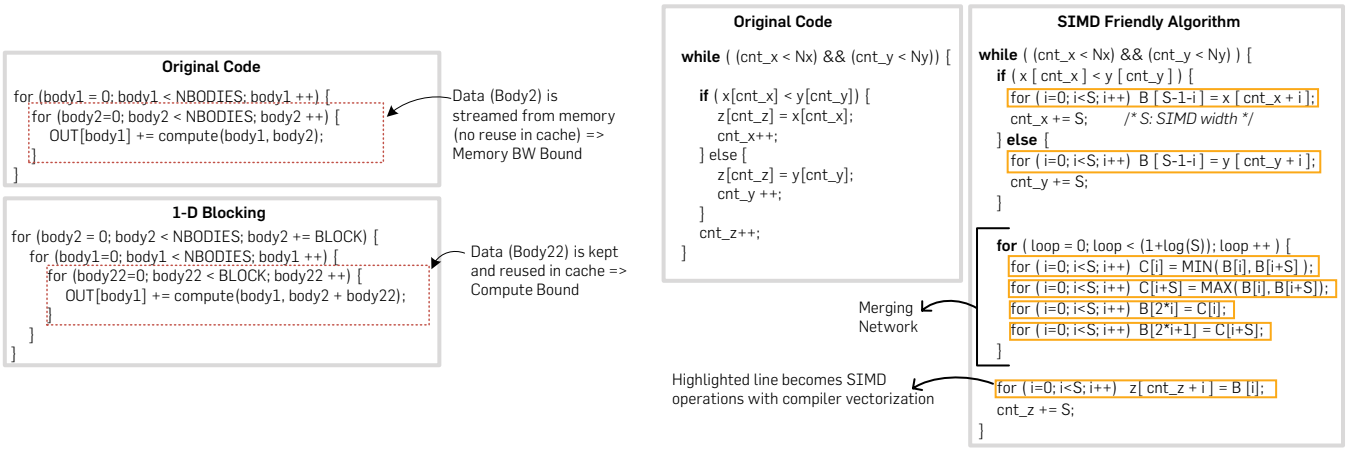


**图 6：代码片段演示算法更改效果：(a) NBody 中分块；(b) 对 SIMD 友好的 MergeSort 算法。**




### 4.2 编译器技术

在阐述了算法更改之后，我们将演示运用最新编译器中的并行化和向量化技术对填补忍者差距的影响。

**并行化。** 我们通常对最外层循环运用 OpenMP 编译指导语句来并行化我们的基准测试。OpenMP 提供了可移植解决方案，允许指定要启动的线程数、线程与核心关系，以及线程私有 / 共享变量的规格。由于吞吐量型基准测试提供大量的 TLP（通常是循环的外层），我们基本上使用了 **omp parallel for** 杂注。图 7a 中显示了 1D 复数卷积的一个示例。使用 SMT 线程有助于隐藏代码中的延迟性 — 因此，有时可以在我们的 6 核系统中获得 6X 的扩展。

**向量化。**

**SSE 与 AVX：**图 8a 显示，进行了正确的算法更改后 Westmere 上从内层和外层循环向量化获得的增益。我们也将它与手动最佳优化的代码的 SIMD 扩展相比较，在图 8b 中显示利用 4 核 - 3.4 GHz Intel® Core i7-2600 K Sandybridge 系统时对 AVX (8-wide SIMD) 的扩展。我们使用了相同的编译器，只是将编译选项从 **-xSSE4.2** 改为 **-xAVX**。在性能方面，使用编译后的代码时我们获得了平均 2.75X 的 SIMD 扩展，这在使用最佳优化代码时 2.9X 扩展的 10% 范围内。运用 8-wide AVX 时，我们的编译后代码和最佳优化代码分别获得了 4.9X 和 5.5X 的扩展（两者又非常接近）。

在我们大多数的基准测试中，最佳优化代码的总体 SIMD 扩展都不错，但 MergeSort、TreeSearch 和 BackProjection 例外。正第如 4.1 节中所述，我们在 MergeSort 和 TreeSearch 中进行的算法更改以执行更多运算为代价，导致了低于线性加速的结果。BackProjection 没有线性扩展，因为代码中存在不可避免的 gather/scatter 运算。对于 BackProjection 而言，这将 SSE 上的 SIMD 扩展限制在 1.8X（AVX 上为 2.7X）。

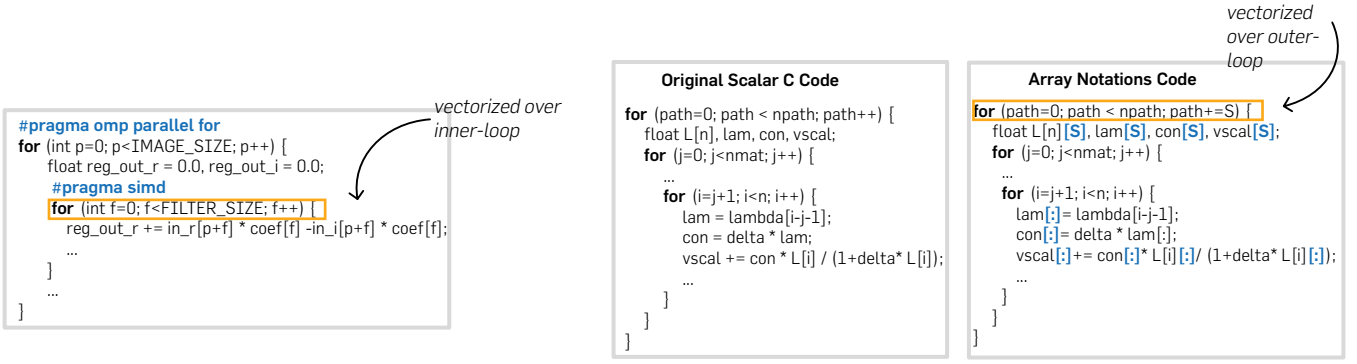
**内层循环向量化：**我们的基准测试中大部分对代码的内层循环进行向量化，或者是使用编译器自动向量化，或者是当依赖性或内存别名分析失败时使用 **#pragma simd**。增加这一编译器标注语句的作用是向编译器发出该循环必须（并且也是安全的）向量化的命令。图 7a 显示了使用这一编译器标注的示例。内层循环向量化在 SSE 上的平均加速为 2.2X（AVX 上为 3.6X）。

**外层循环向量化：**对外层循环进行向量化颇具挑战性。需要对多个循环级别进行归纳变量分析；zero-trip 测试和退出条件等循环控制流必须转换为对多个矢量元素的有条件 / 判定执行。数组标记技术可以帮助程序员避免这些复杂性，而且无需费心更改代码。我们在 LIBOR 中从外层循环向量化获得了性能增益，其中内层循环仅可部分向量化。我们目前使用数组标记来向量化（完全独立）外层循环（图 7b）。对标量代码进行了修改，从而改变外层循环索引以反应向量化，并计算并行的多次迭代的结果。请注意，程序员声明了尺寸为 *S*（simd 宽度）的数组，而 X[*a*:*b*] 标记则代表从索引 *a* 起访问 X 的 *b* 个元素。将标量代码更改为数组标记代码通常直接明了。这一更改使得 SSE 上性能加快 3.6X（AVX 上为 7.5X）。

### 5. 总结

图 9 显示了最佳优化代码与编译器生成的代码（算法更改之前和之后）相比的相对性能。我们假设程序员已经努力引入前述小节中介绍的编译器指导语句和编译器指令。**在算法更改之前**，编译后代码和最佳优化代码之间存在**平均 3.5X 的差距**。此差距主要因为编译后代码受到了内存带宽或 SIMD 效率不足的限

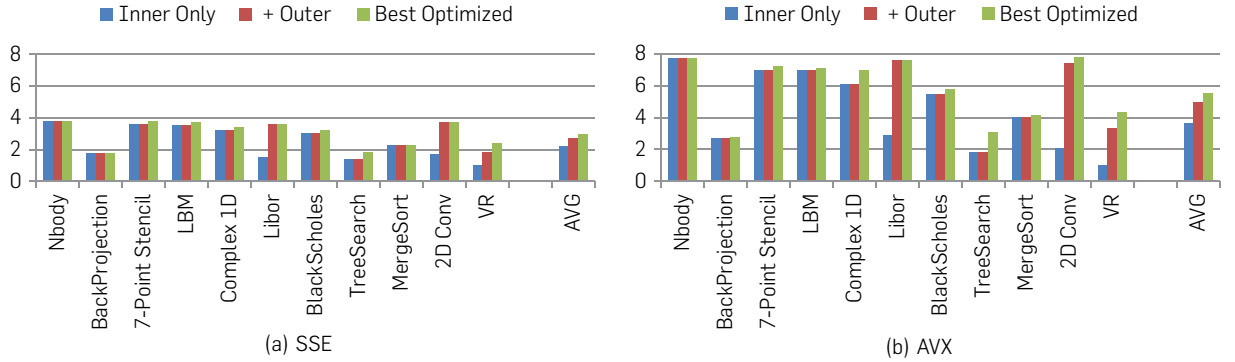
**图 7：代码片段演示编译器技巧：(a) 1D 复数卷积中并行化与内层循环向量化；(b) LIBOR 中外层循环向量化。请注意所需的代码更改量很小，通过少量程序员工作即可实现。**




(a) Code from Complex 1D Conv. Example of inner-loop vectorization

(b) Code from Libor. Example of outer-loop vectorization

**图 8：从内层和外层循环向量化获得的益处：(a) SSE 和 (b) AVX。我们也与最佳优化代码的性能进行了比较。**




(a) SSE

(b) AVX

制。在进行了第 4.1 节中所述的算法更改后，此差距缩小到平均 1.4X。唯一存在显著差距的基准测试为 TreeSearch，其中编译器向量化了带有 gather 的外层循环。其余的基准测试显示了 1.1–1.4X 的忍者差距，原因主要在于额外的 spill/fill 指令和 load/store 指令增加了指令数量 — 这些是因为编译器依赖启发法而导致的疑难问题。

**众核架构的影响：** 为测试众核平台上的忍者差距，我们在 Intel® Xeon Phi™ “Knights Corner” 协处理器 (KNC) 上进行了相同的实验；该处理器在一个晶片上拥有 60/61 个核心，每个核心具有带 4-way SMT 和 512-bit SIMD 单元的顺序微架构。

图 10 显示了 KNC 以及 Westmere 的忍者性能差距。KNC 的平均忍者差距仅为 1.2X，而这几乎与 CPU 相同（稍微小一些）。两个性能差距的主要差别来自 TreeSearch，它受益于 Xeon Phi 上的硬件 gather 支持，其性能与最佳优化的代码接近 (1.1X)。

算法更改后剩余的忍者差距在 KNC 和 CPU 之间保持较小而稳定，尽管 KNC 上的核心数和 SIMD 宽度要大得多。这是因为我们的算法优化侧重于解决代码中的向量化和带宽瓶颈。一旦这些问题得以解决，未来的架构将能够利用越来越多的硬件资源，实现稳定而可预测的性能增长。

## 6. 探讨

我们在第 4.1 节中介绍的算法优化适用于包括 GPU 在内的多种架构。之前发表的多篇论文 [19, 21] 探讨了在 GPU 上获得最佳性能所需的优化。在本章中，我们展示相同算法优化对 GPU 性能的影响。本研究中我们使用了 NVIDIA C2050 Tesla GPU。

虽然 GPU 支持硬件 gather/scatter，最佳的编码实践（如 CUDA C 编程指南 [19]）指出需要避免非合并的全局内存访问 — 包括将数据结构从 AOS 转换为 SOA 以缩短延迟性并降低带宽用量。GPU 也要求分块优化，这指的是数据传输到 GPU 的共享内存（或缓存）/ 在其中进行管理。最后，使用对 SIMD 友好的算法可以让 SIMD 宽度大于当前 CPU 的 GPU 大幅获益。从算法优化获得的平均**性能增益为 3.8X**（图 11）— 高于

CPU 上的 2.5X 增益，因为 GPU 有更多的 SM 和更宽的 SIMD，所以次优化算法选择可以对性能产生很大的影响。

## 7. 相关研究

多篇已发表的论文显示出比过去利用精心调节的代码所做的研究高出 10–100X 的性能。[1, 8, 14, 17, 24] Lee 等人 [15] 总结了相关的硬件架构特征，以及针对 CPU 和 GP 的平台相关软件优化指南。虽然这些研究指出了巨大忍者性能差距的存在，但它们没有介绍相关的编程工作，或者如何填补这种忍者差距。

在本研究中，我们分析了忍者差距的来源，并通过传统的编程模型在较少程序员投入的前提下填补这一差距。本论文的上一版本由 Satish 等人发表。[23] 产品级编译器最近已开始支持编译器研究中公开的并行化和向量化技术。此类技术的例子包括 OpenMP（在最近的 GCC 和 ICC 编译器中用于并行化），以及处理对齐约束和外层循环向量化的自动向量化技术。[18] 这些技术利用了简单明了的杂注，以及作为 Intel® Cilk™ Plus 的一部分的数组标记 [11] 等技术。

然而，即便有了编译器支持，编写幼稚的代码仍然可能无法扩展，因为它们受到了体系结构特征的制约，如内存带宽、gather/scatter 等，或者因为算法本身无法被向量化。在这些情形中，需要诸如分块、SOA 转换和对 SIMD 友好的算法等算法更改。为解决这些算法更改人们提出了各种各样的技巧，或者使用编译器协助的优化，或者使用无视缓存的算法或特殊语言。此类更改通常要求程序员的干预和投入，但可以在多个架构和代次之间利用。例如，多篇论文演示了类似算法优化对 GPU 的影响。[21]

虽然我们的研究关注的是传统的编程模型，但人们也提出了激进的编程模型变化以填补这一差距。近期的建议包括域特定语言（Fowler 提供了调查问卷 [9]）、Berkeley View 项目，[2] 以及为异构系统编程的 OpenCL。也提出了以库为导向的方法，如 Intel® Threading Building Blocks、Intel® Math Kernel Library 和 Microsoft Parallel Patterns Library (PPL) 等。我们相信这些都是正交的，可与传统模型结合使用。

此外，也有大量的文献关注于将自动调节用作填补这一差距的方法。[8, 25] 自动调节的结果可能会比最佳优化的代码差得多。例如，对于自动调节的模板计算，[8] 我们最佳优化代码的性能要好 1.5X。既然对于模板的忍者差距仅仅是 1.1X，因此我们编译后的代码的性能要比自动调节代码好 1.3X。我们希望编译后的结果

**图 9：最佳优化的代码、编译器生成的代码（算法更改后）以及编译器生成的代码（算法更改前）之间的相对性能。性能数据已正规化为算法更改后的编译代码。**

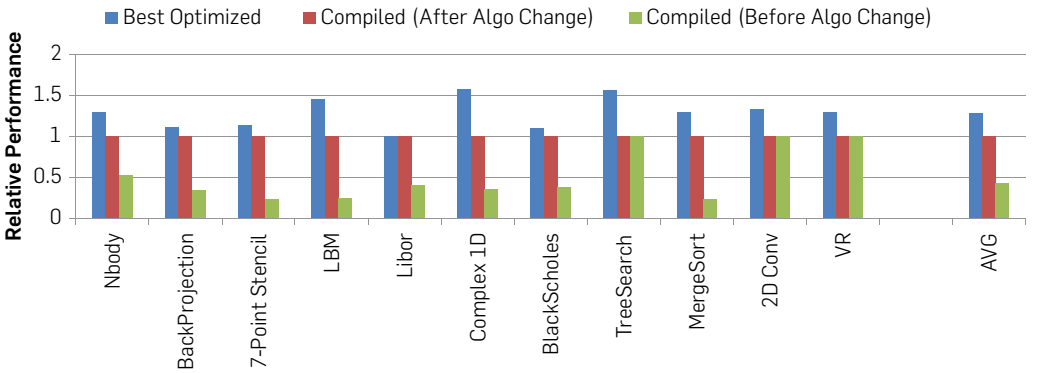



**图 10：Intel® Xeon Phi™ 和 CPU 上最佳优化代码和编译器生成的代码（算法更改后）之间的性能差距。**

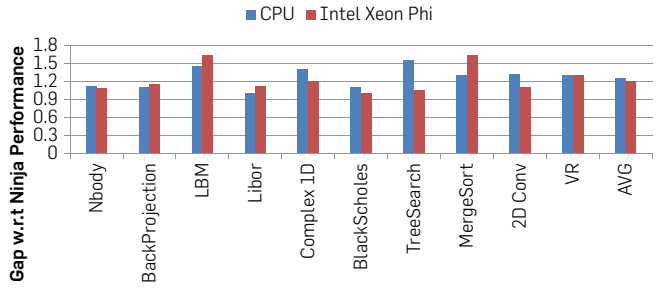
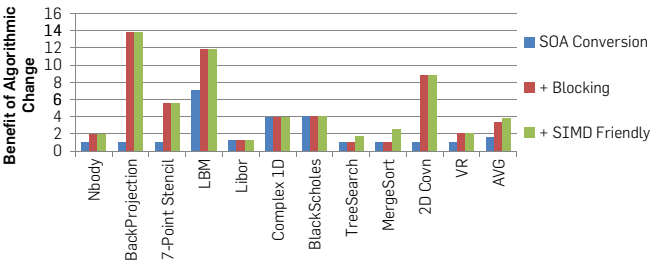



**图 11：图 6 中所述算法更改在 NVIDIA Tesla C2050 GPU 上的效果。**



大体上能与自动调节的结果相匹敌，同时也提供能够利用在处理器代次之间轻松移植的标准工具链的优势。

## 8. 结论

在本研究中，我们展示了一组真实吞吐量计算型基准测试在最新的多核处理器上存在 **24X** 的巨大忍者性能差距。如果不解决的话，这一差距将毫无意外地继续扩大。我们演示了一组简单而闻名的算法技巧与现代编译器技术进步的结合如何能将平均 **Ninja** 差距缩小到**仅仅 1.3X**。与生成忍者代码所需的大量投入相比，这些更改通常仅需少量的编程工作。 C


参考资料

1. Arora, N., Shringarpure, A., Vuduc, R.W.Direct N-body Kernels for multicore platforms.In *ICPP* (2009), 379–387.
2. Asanovic, K., Bodik, R., Catanzaro, B., Gebis, J., Husbands, P., Keutzer, K., Patterson, D.A., Plishker, W.L., Shalf, J., et al.*The Landscape of Parallel Computing Research:A View from Berkeley*.Technical Report UCB/EECS-183, 2006.
3. Bienia, C., Kumar, S., Singh, J.P., Li, K. The PARSEC benchmark suite:Characterization and architectural implications.In *PACT* (2008), 72–81.
4. Brace, A., Gatarek, D., Musiela, M. The market model of interest rate dynamics.*Mathematical Finance 7*, 2 (1997),127–155.
5. Chen, Y.K., Chhugani, J., et al.Convergence of recognition, mining and synthesis workloads and its implications.*IEEE 96*, 5 (2008),790–807.
6. Chhugani, J., Nguyen, A.D., et al.Efficient implementation of sorting on multi-core simd cpu architecture.*PVLDB 1*, 2 (2008), 1313–1324.
7. Dally, W.J.The end of denial architecture and the rise of throughput computing.In *Keynote Speech at Desgin Automation Conference* (2010).
8. Datta, K. Auto-tuning Stencil Codes for Cache-based Multicore Platforms. PhD thesis, EECS Department, University of California, Berkeley (Dec 2009).
9. Fowler, M. *Domain Specific Languages*, 1st edn.Addison-Wesley Professional, Boston, MA 2010.
10. Giles, M.B.*Monte Carlo Evaluation of Sensitivities in Computational Finance*.Technical report.Oxford University Computing Laboratory, 2007.
11. Intel.A quick, easy and reliable way to improve threaded performance, 2010. software.intel.com/articles/intel-cilk-plus.
12. Ismail, L., Guerchi, D. Performance evaluation of convolution on the cell broadband engine processor.*IEEE PDS 22*, 2 (2011), 337–351.
13. Kachelrieb, M., Knaup, M., Bockenbach, O. Hyperfast perspective cone-beam backprojection. *IEEE Nuclear Science 3,* (2006), 1679–1683.
14. Kim, C., Chhugani, J., Satish, N., et al.FAST: fast architecture sensitive tree search on modern CPUs and GPUs.In *SIGMOD* (2010).339–350.
15. Lee, V.W., Kim, C., Chhugani, J., Deisher, M., Kim, D., Nguyen, A.D., Satish, N., et al.Debunking the 100X GPU vs.CPU myth:An evaluation of throughput computing on CPU and GPU.In *ISCA* (2010).451–460.
16. T.N. Mudge.Power:A first-class architectural design constraint.*IEEE Computer 34*, 4 (2001), 52–58.
17. Nguyen, A., Satish, N., et al.3.5-D blocking optimization for stencil computations on modern CPUs and GPUs.In *SC10* (2010).1–13.
18. Nuzman, D., Henderson, R. Multi-platform auto-vectorization.In *CGO* (2006).281–294.
19. Nvidia.*CUDA C Best Practices Guide 3*, 2 (2010).
20. Podlozhnyuk, V. Black–Scholes option pricing.*Nvidia,* 2007. http://developer.download.nvidia.com/compute/cuda/1.1-Beta/x86_website/projects/BlackScholes/doc/BlackScholes.pdf.
21. Ryoo, S., Rodrigues, C.I., Baghsorkhi, S.S., Stone, S.S., Kirk, D.B., Hwu, W.M.W. Optimization principles and application performance evaluation of a multithreaded GPU using CUDA. In *PPoPP* (2008).73–82.
22. Satish, N., Kim, C., Chhugani, J., et al.Fast sort on CPUs and GPUs: A case for bandwidth oblivious SIMD sort.In *SIGMOD* (2010).351–362.
23. Satish, N., Kim, C., Chhugani, J., Saito, H., Krishnaiyer, R., Smelyanskiy, M., et al.Can traditional programming bridge the Ninja performance gap for parallel computing applications?In *ISCA* (2012).440–451.
24. Smelyanskiy, M., Holmes, D., et al. Mapping high-fidelity volume rendering to CPU, GPU and many-core.*IEEE TVCG*, 15, 6(2009), 1563–1570.
25. Sukop, M.C., Thorne, D.T., Jr.*Lattice Boltzmann Modeling:An Introduction for Geoscientists and Engineers*, 2006.
26. Tian, X., Saito, H., Girkar, M., Preis, S., Kozhukhov, S., Cherkasov, A.G., Nelson, C., Panchenko, N., Geva, R..Compiling C/C++ SIMD extensions for function and loop vectorizaion on multicore-SIMD processors. In *IPDPS Workshops* (Springer, NY, 2012).2349–2358.



Nadathur Satish、Mikhail Smelyanskiy 和 Pradeep Dubey 就职于 Intel 公司的并行计算实验室

Changkyu Kim 就职于 Google 公司

Jatin Chhugani 就职于 Ebay 公司

Hideki Saito、Rakesh Krishnaiyer 和 Milind Girkar 就职于 Intel 公司的 Intel 编译器实验室


译文责任编辑：翟季冬