爱开发 重创新 更智慧

Innovate2011

IBM Rational 软件创新论坛

Software. Everyware.

服务器系统设计发展趋势及对应用程序开发的影响

Kevin Stoodley

IBM 院士、首席技术官：企业现代化工具、编译器与安全

stoodley@ca.ibm.com

Rational 软件部

演讲纲要

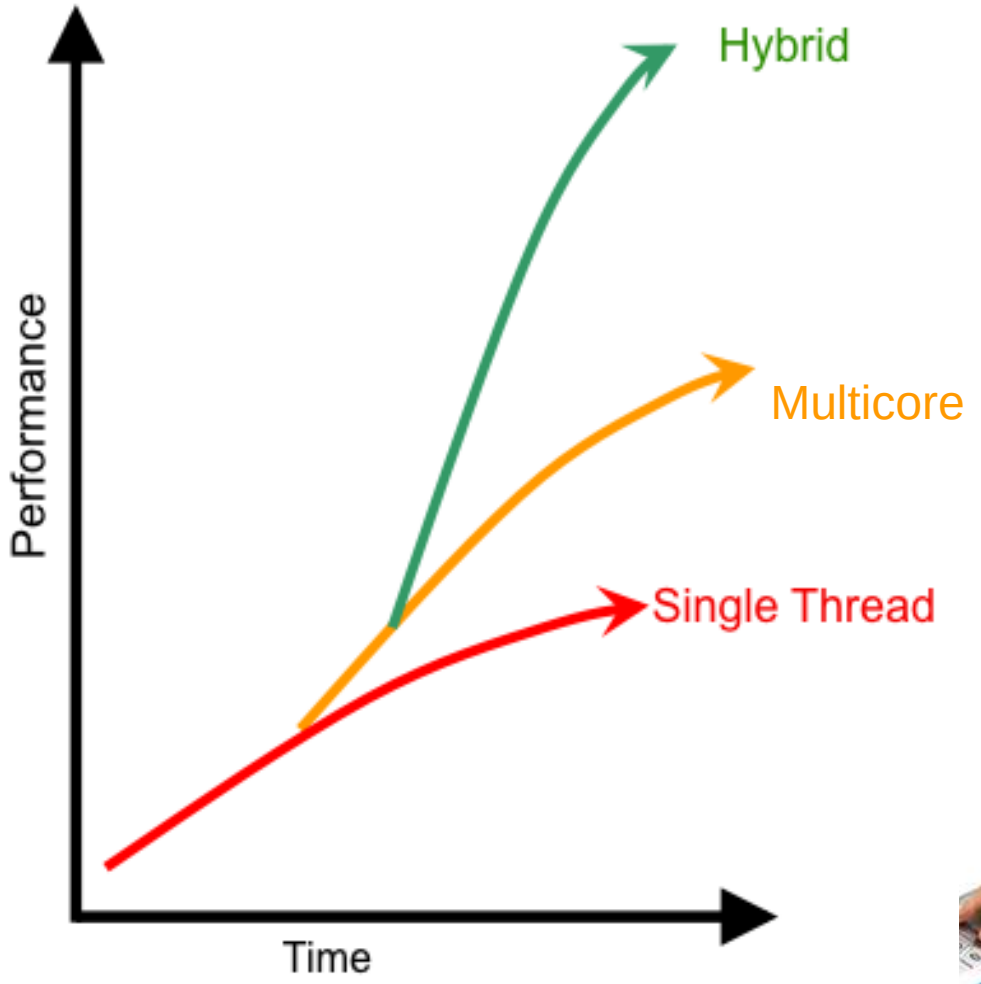
背景介绍

并行计算运用模型

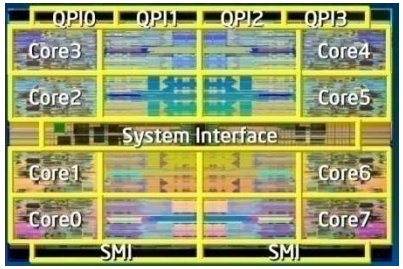
软件性能调优

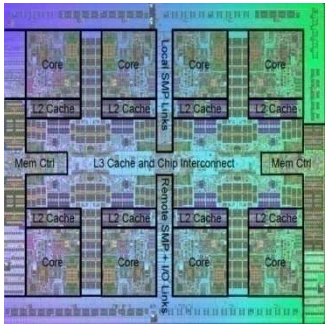
背景介绍：微处理器的发展趋势

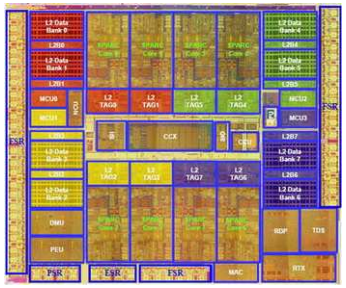
- Single Thread performance power limited
  单线程：有限的性能
- Multi-core throughput performance extended
  多核：扩展的吞吐量及性能
- Hybrid extends performance and efficiency
  混合微处理器：扩展了性能和效率



多核处理器的不断演变







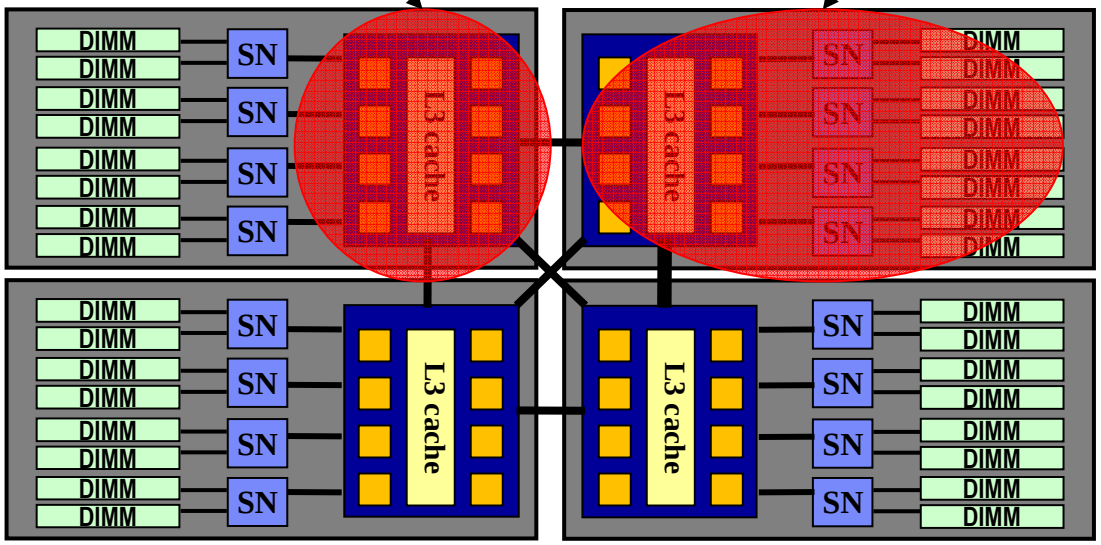
	Nehalem EX	POWER 7	UltraSPARC T2
每个芯片的最大内核数	8	8	8
每个内核的最大线程数	2	4	8
最高的芯片内置缓存	24MB	32MB	4MB
每个芯片的内存控制器数	2	2	4
每个系统的最大芯片数	8	32	4
最大的系统规模（线程数）	128	1,024	256

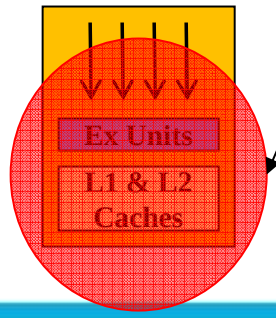
线程速度和吞吐量之间是不断地相互制约平衡的
内存使用量和当前内存页之间也是不断地相互制约平衡的

内存系统：NUMA（*Non-Uniform Memory Access* 非统一内存访问）成为新标准

次紧密的结合：一个芯片的内核之间　芯片与内置的动态随机存取存储器的结合

IBM Power 750
POWER 7
32核，128线程

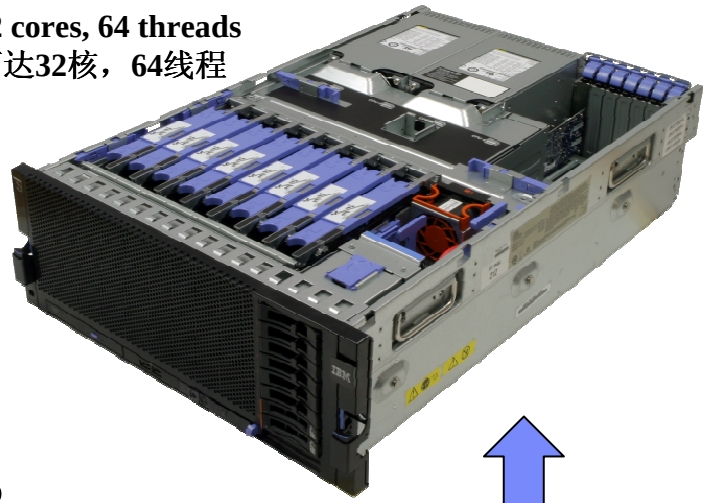




最紧密的结合：一个内核的线程之间

满足海量数据需求的系统

IBM 3850 X5
Nehalem-EX
up to 32 cores, 64 threads
最高可达32核，64线程



超密集动态随机存储器 （MAX5）

极快速度的随机读/写
最高的性格与能耗
有限的容量

企业级的NAND闪存

高速随机读
最低的价格与每秒读写的能耗
高容量(TBs)

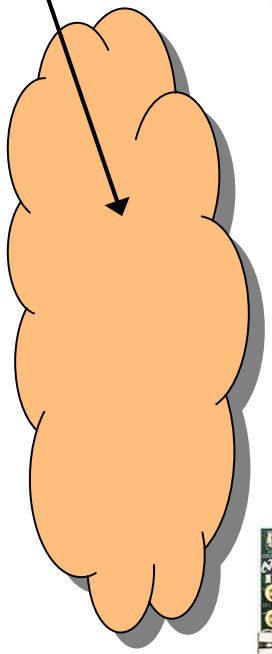
并行磁盘阵列

高速的连续读/写
每单位GB最低的成本
等同于无限容量(PBs)

新兴的混合体架构

互连接
从输入/输出总线进化为更紧的或更松的耦合

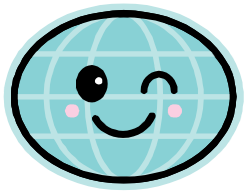
通用的处理器日渐显现出能耗与冷却系统的局限性



计算加速器
通常见于宽数据并行处理器。图形处理器GPU便是一个常见例子。

数据加速器
主要用于数据转换，比如加密或压缩

可重新配置的硬件
可定制的功能，典型例子是现场可编程门阵列Field－Programmable Gate Array



高级软件面临的挑战

- **并行**
  - 增加每个系统的线程数，即使用更多的并行来达到更高的利用率
  - 线程与线程的联结（共享代码、数据或两者）
- **内存管理**
  - 在更多的线程中共享缓存与内存带宽，更加需要内存的高效性
  - 线程与内存的联结（在离相关数据最近的地方执行线程）
- **存储管理**
  - 根据访问频率和访问方式，在动态随机存取存储器、硬盘与闪存中分配数据
- **加速器**
  - 分配代码在加速器或通用处理器上运行
  - 管理共享的数据
  - 异步计算与通信

为下次的露营行动选择选择最好的工具

只做一件事并把它做好

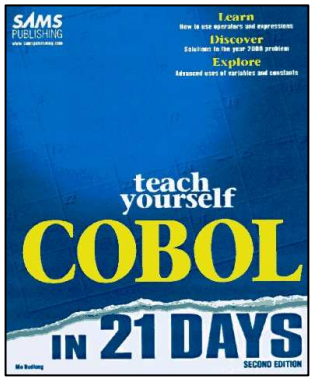
做很多事，但没一件做好的

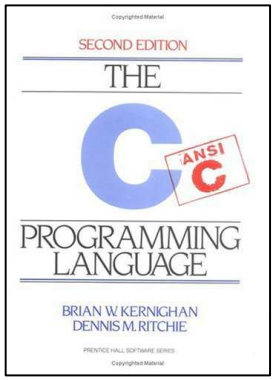
在设计领域，需要在“广而弱”和“专而强”之间进行平衡。象人类一样，工具看上去似乎要么是万事通，要么是某方面的专家。 这种万事通或专家的平衡在所有工具，无论是凿子、螺丝起子、还是自行车、超级计算机的设计中，都有着重大的影响。众所周知，又专业又面面俱到的系统从来就不存在。总要二者择其一。将新出现的技术与合适的设计结合的一个重要方面就是从一开始改变这种情况的潜力。

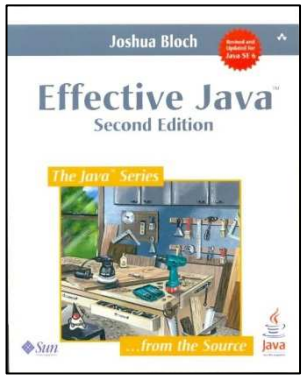
Bill Buxton

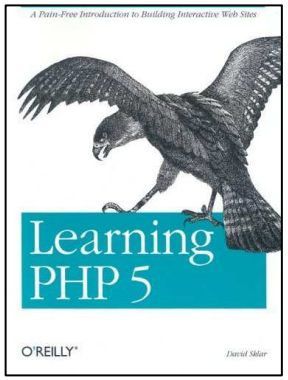
少即是多（多或少）： 《非凡的灵感与计算机设计》

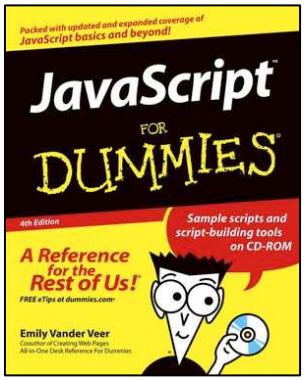
史上最成功的编程语言是什么？







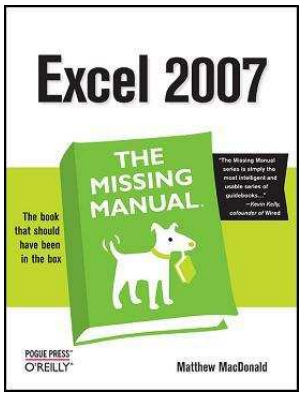




。。之最　代码量最少　最好的网页指南　发行的书最多　最利于找工作　最大众化

这是个诡异的问题

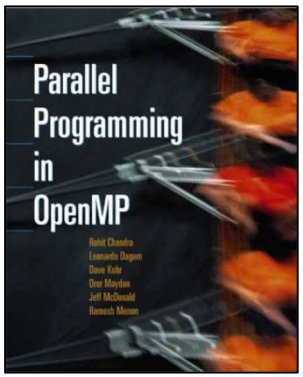
答案是…

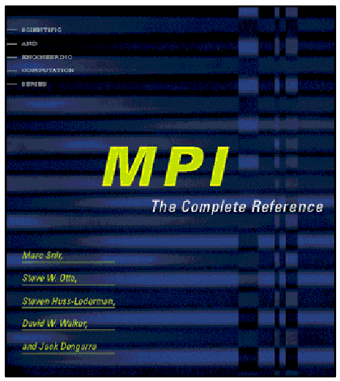


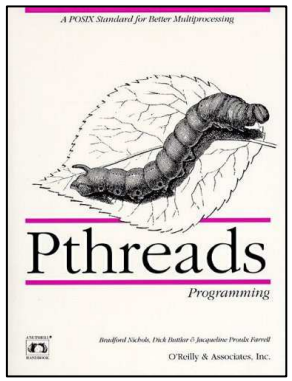
Spreadsheets!
Excel 电子工作表！

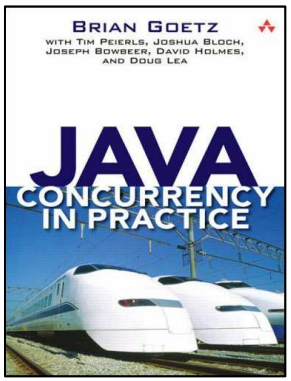
拥有不计其数的高级用户！

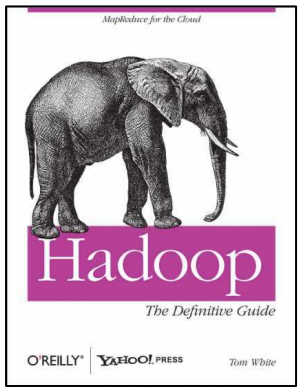
史上最成功的并行编程语言是什么？





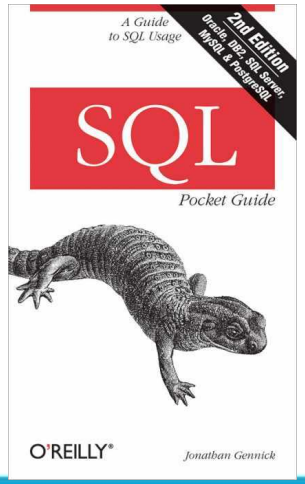






好吧，这次不吊大家的胃口了

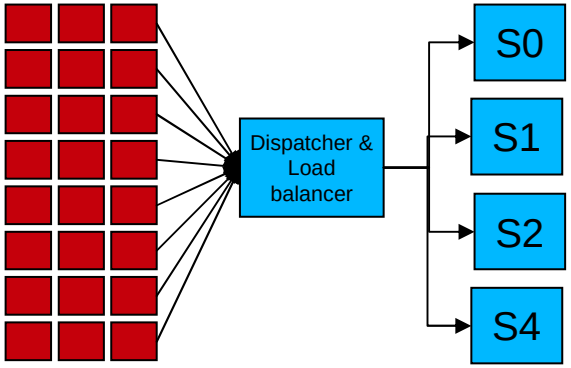
答案是…



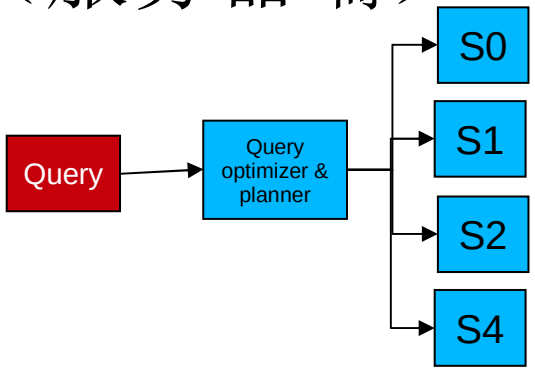
SQL, 数据库访问的标准语言，是一种声明式语言. 它把用户从访问和操作数据的复杂的过程化的细节中解放出来。特别是，使用并行机器并不需要用户去学一门新的语言或重写现有的SQL代码.

W. Hasan, “在并行机器上优化 SQL 查询”

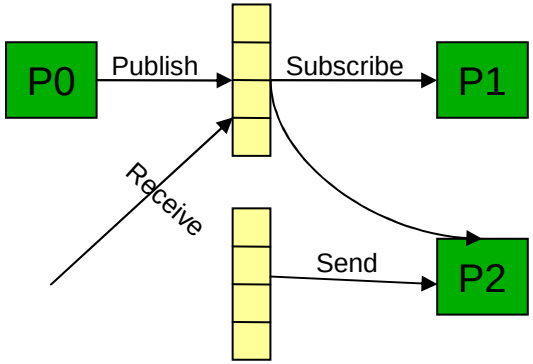
成熟的并行编程模式：主流（服务器端）



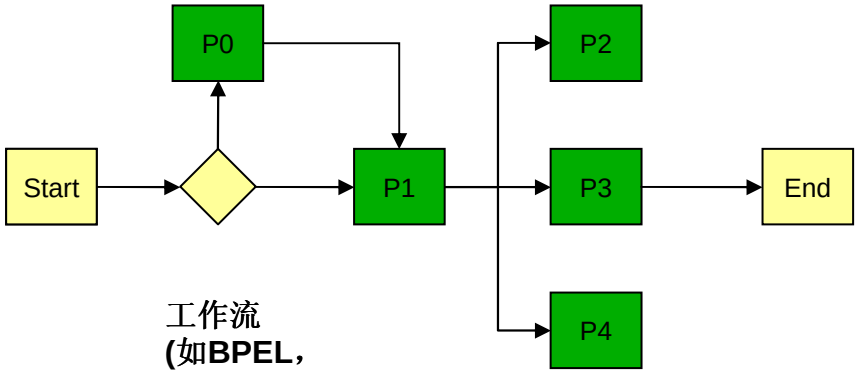
客户-服务器
如：**REST**
表述性状态转移



声明式查询
(如SQL)

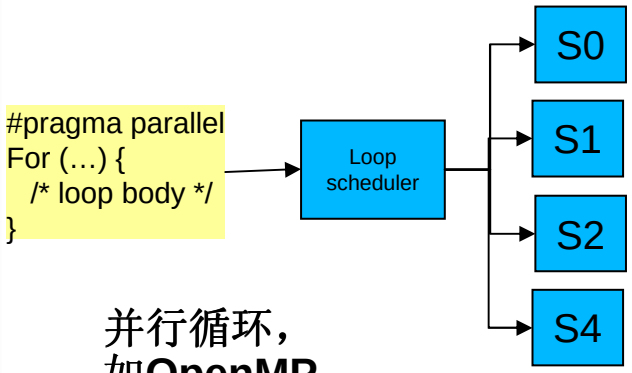


消息队列
(如：Java消息服务)

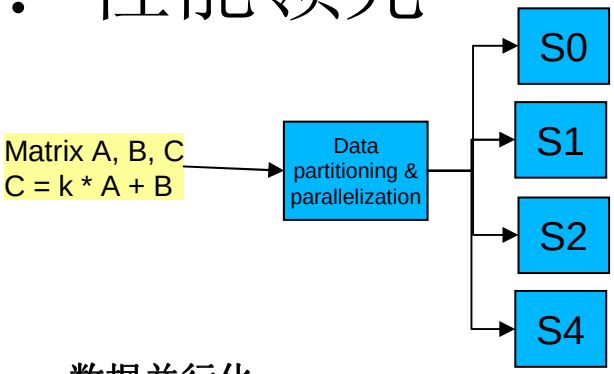


工作流
(如BPEL，
Business Process Execution Language
业务流程执行语言)

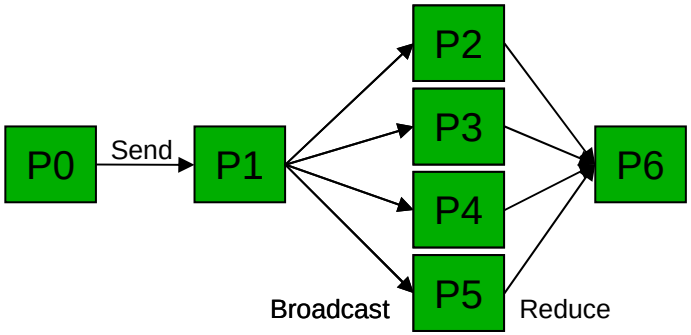
成熟的并行编程模式：性能领先



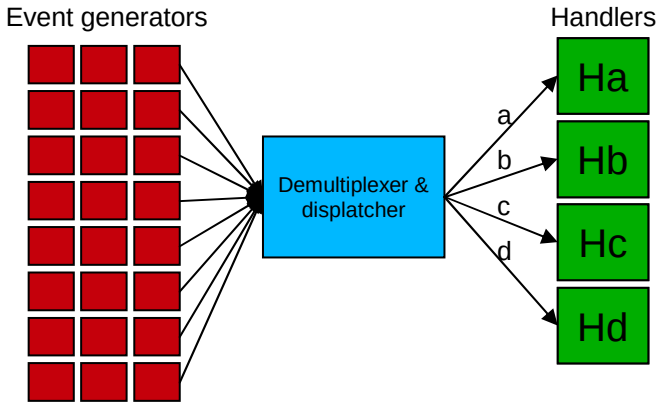
并行循环，
如**OpenMP**



数据并行化
如 可扩展线性代数库
(ScaLAPACK)



SPMD (消息传递)
Single Program Multiple Data
(e.g. MPI消息传递标准接口)



事件处理
(如. ACE)
Adaptive Communication Environment

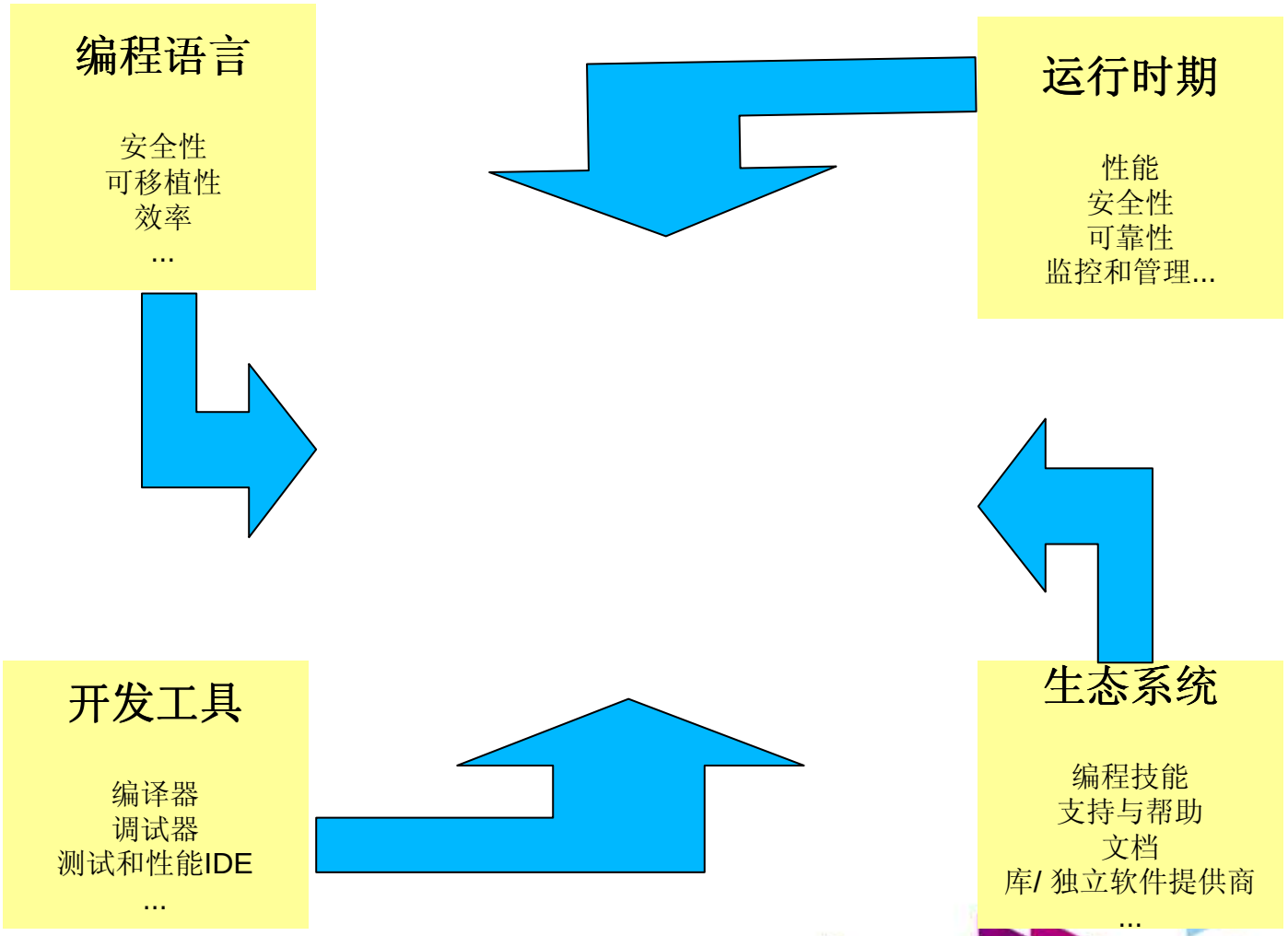
程序员的分类

应用程序员	主流应用	**通常开发如下程序** 数据库应用 Web 应用 桌面应用 应用集成
	性能领先应用	高可扩展Web应用 低延迟或实时应用 科学计算与工程程序 游戏 通迅类程序
系统程序员		操作系统 中间件 编程语言 开发工具

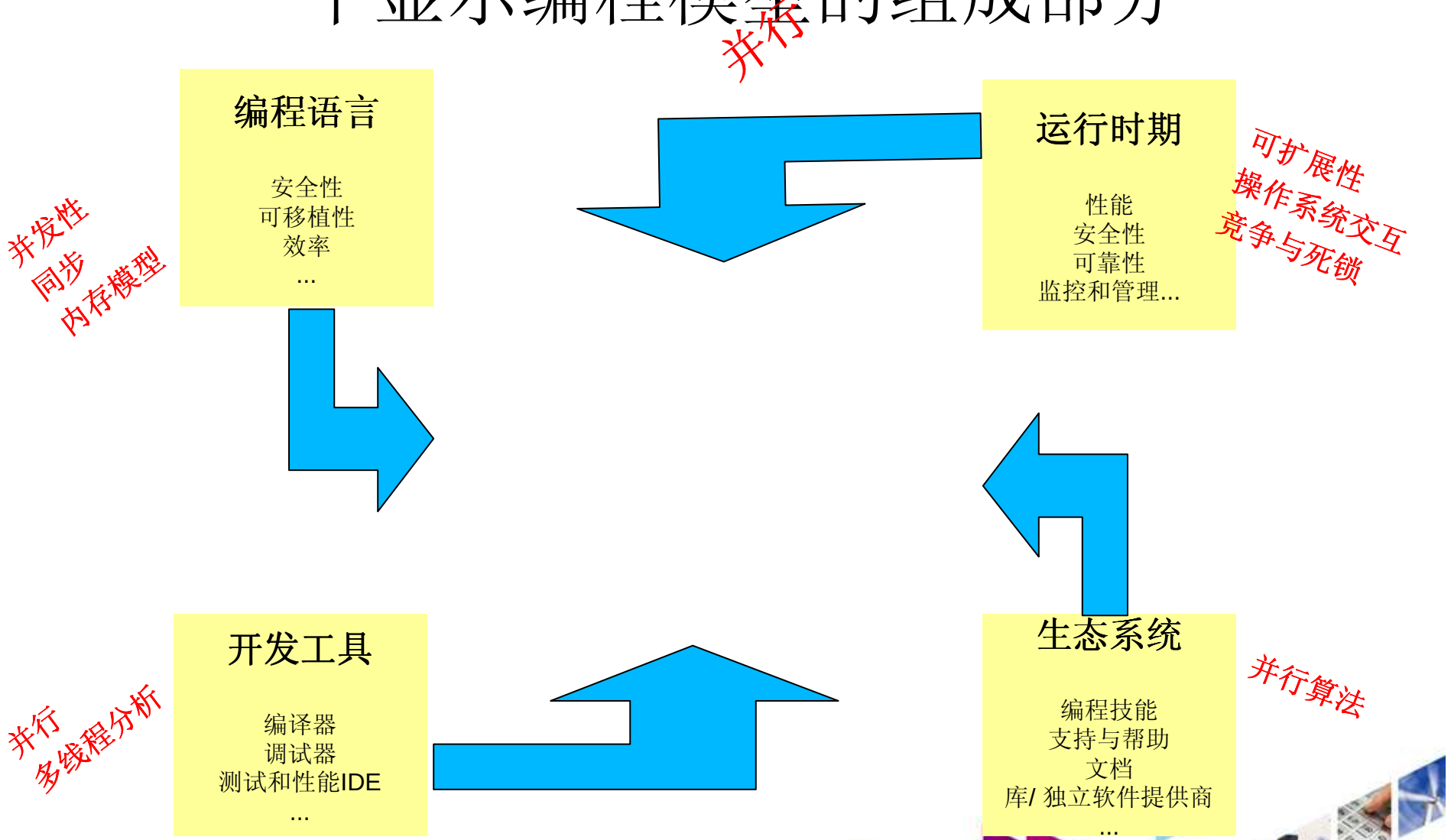
这些程序员的共同点有哪些？

- 应用程序员不需要接触通用的并行编程模式
- 并行化仅由少数几位高技能的程序者来完成，他们开发一些强壮的且可扩展的并行化的实时实现
  - 这是中间件的核心价值之一
- 使用上的重大突破是用一种简单的易学的形式来实现复杂的功能，从而使并行技术能广泛地被非并行专家的人士采用
  - 非程序员也可以用Excel“编程”来解决复杂的问题
  - 没有并行编程经验的程序员也能使用SQL, publish-subscribe, JEE等开发强大的、可扩展的、高可用的并行程序

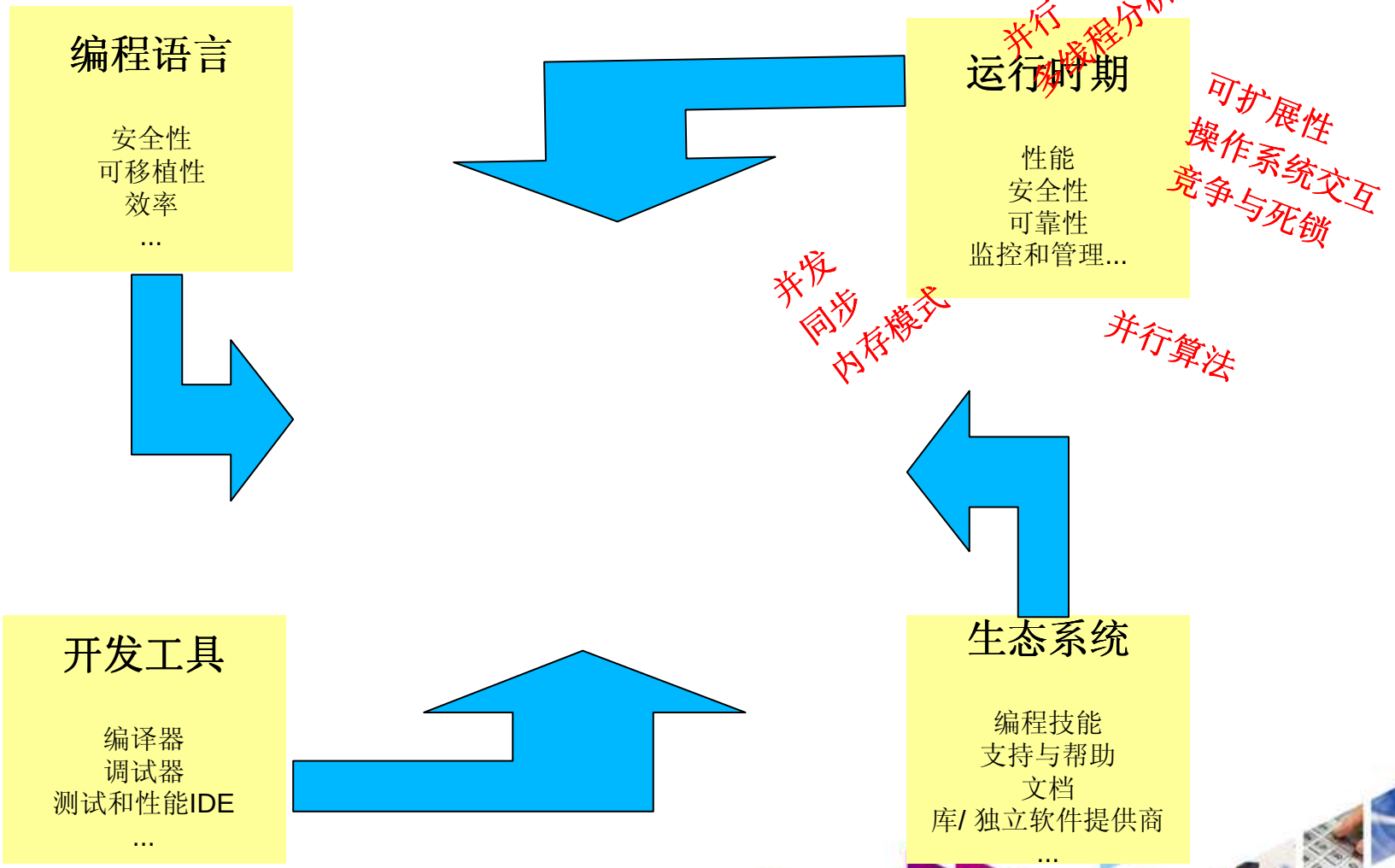
一个编程模型的组成部分



一个显示编程模型的组成部分



一个成功的主流编程模型的组成部分



一些新兴的并行编程模型

下列每一种都有它自己的技术特点和用户群，但是
a) 没有一种可以在普及性与使用方面与前面主流编程模型相比较
b) 在前面提及的编程语言成熟度的四个重要方面（编程语言、运行时间、工作和生态系统），这些模型都有一到多个薄弱点。

任务导向的
更丰富、更安全
多线程编程

Java fork-join, Intel TBB, Microsoft TPL, Cilk

参与者导向的
消息在流程中或激活的对象之间传递

Erlang, Scala Actors, ActorFoundry, Kilim, Microsoft Axum

混合数据并行
在多种类型的处理器之间进行数据并行处理，尤其是图形处理器

Nvidia CUDA, OpenCL, Microsoft DirectCompute, Intel Ct/RapidMind

聚类数据并行
海量并行处理
（多为非结构化的）数据

Hadoop (+ Pig/Hive/JAQL), StreamSQL, Streams SPL

分割全局地址空间模型 (PGAS)
适用于大规模群的共享内存模型

Unified Parallel C, Co-Array Fortran, X10

硬件的前景正在改变

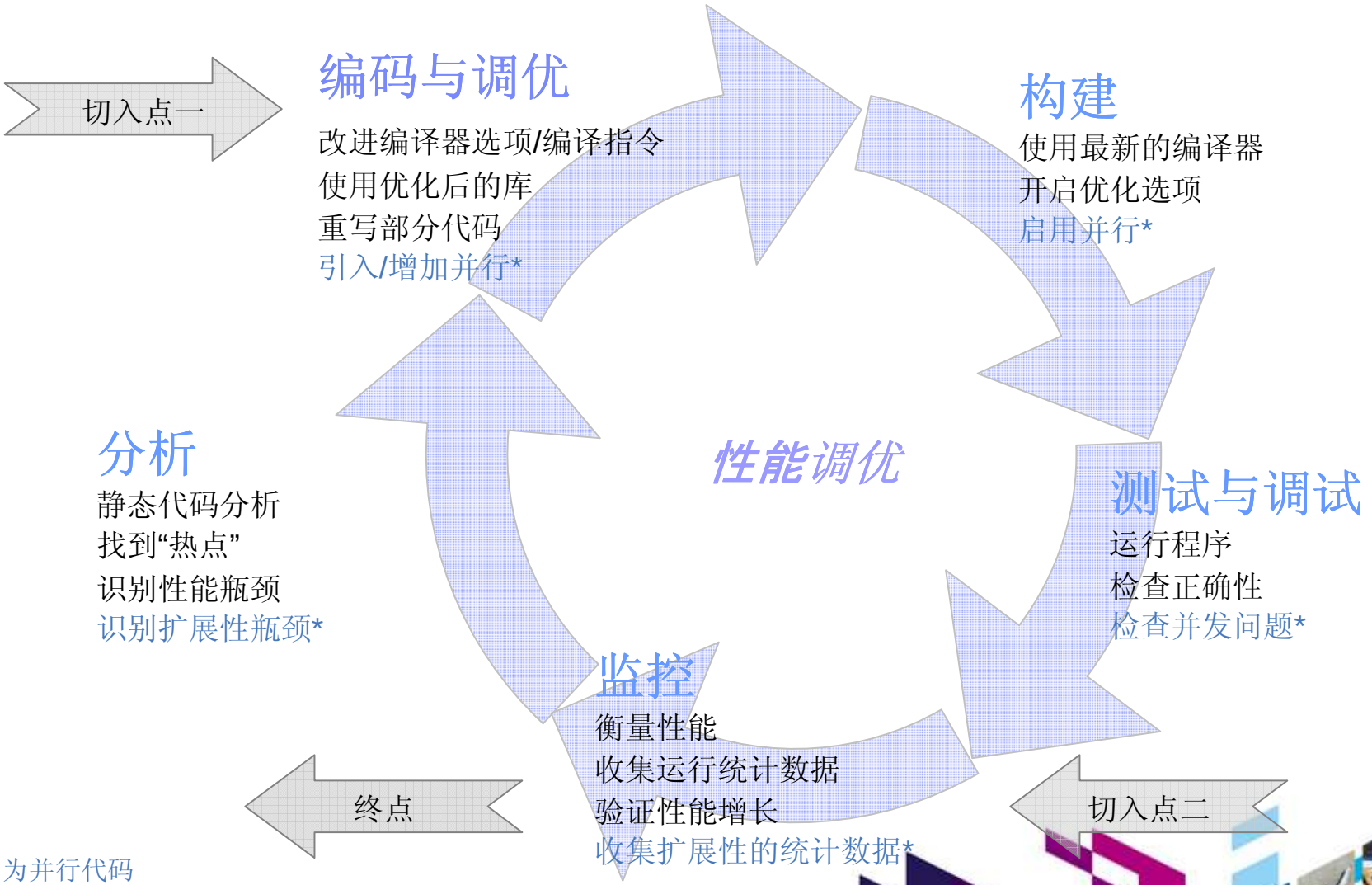
- **单线程的性能的发展正在放慢脚步**
  - 将来会变得更加慢
  - 业界已经转向多核系统的设计，内核的数量正在成为新的吉赫（GHz）
  - 更多的挑战如NUMA非一致访问分布共享存储技术、带宽、混合体/可配置的系统进一步使得软件开发者需要具备更复杂的技能去充分挖掘系统的能力
- **多核系统不会自己提高应用程序的性能**
  - 大多数IBM的IT客户将依赖于可扩展的的中间件提高响应时间和吞吐量
  - 其他客户将需要调优他们的应用程序以提高系统性能
- **许多客户将需要对如何提高多核系统性能进行不同的思考**
  - 调试系统以优化应用程序性能（串行的）
  - 逐渐增加并行化
  - 用并行模式重新设计和编写应用程序

软件性能调优

- **硬件升级已经成为提高性能最常用的方式**
  - 硬件升级的过程相对来说比较直接，且能达到可靠的单线程（串型的）的性能提升。
- **经验表明纯软件的单线程性能工程可以获得大于40%的性能提高**
  - 例如，提高FileNet产品性能的MARS项目，DB2 的TPC-C benchmarking
- **大多数软件开发者都不善于性能调优**
  - 软件性能调优大大复杂于升级（兼容的）硬件
  - 需要很高的技能，极可能增加成本、风险和推迟上市时间
  - 并不是开发流程中的典型部分，除了在一些特殊的群体，如高性能计算，系统（类似高性能计算的）和独产软件供应商。
- **需要一套完整的性能调优工具**
  - 业界在这方面的投入是技术为主导的和相对零散的
    - 针对为性能专家，提供了一些互相不协调的开源工具
  - 目前的工具集是针对于性能专家的，并不适用于普通程序员
    - 例如: IBM Visual Performance Analyser and HPCS Toolkit, Sun Studio, Intel VTune, HP Caliper

迭代式应用程序性能调优

软件交付过程中日益重要的组成部分



* 为并行代码

采用并行模式不同阶段所需要的能力

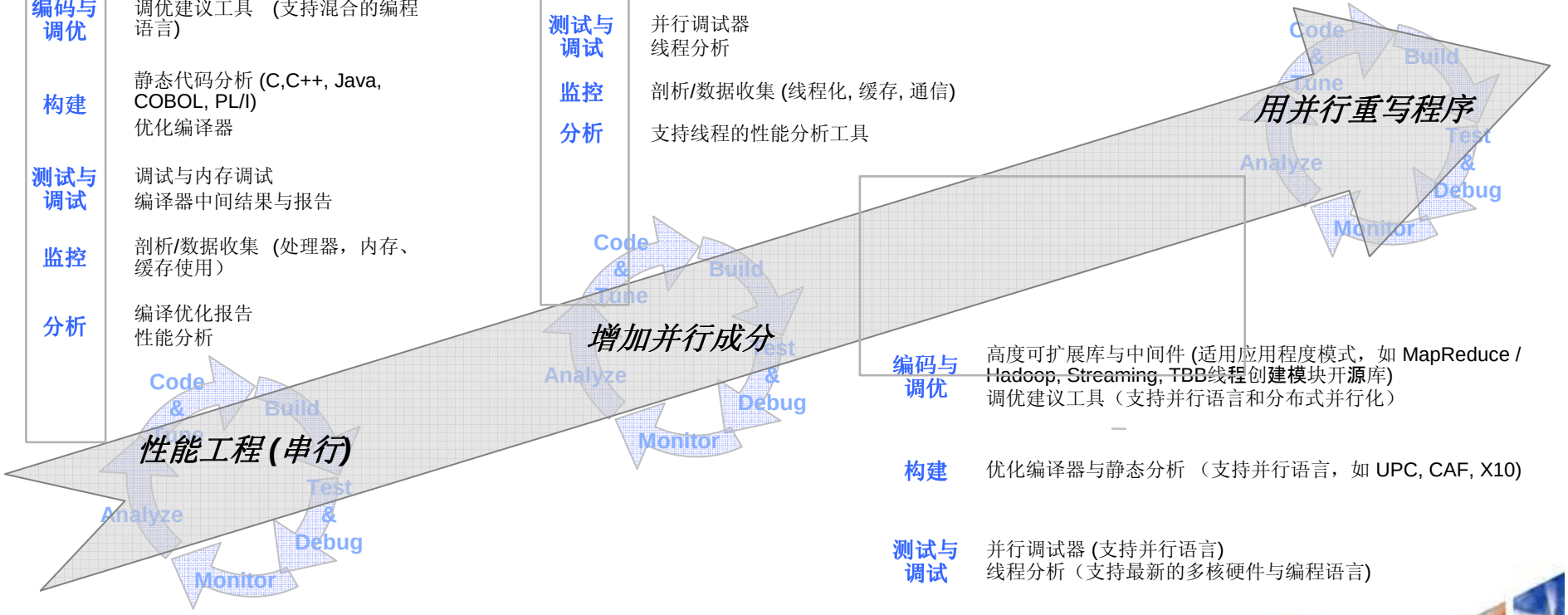
编码与调优 调优建议工具 (支持混合的编程语言)

构建 静态代码分析 (C,C++, Java, COBOL, PL/I)
优化编译器

测试与调试 调试与内存调试
编译器中间结果与报告

监控 剖析/数据收集 (处理器，内存、缓存使用）

分析 编译优化报告
性能分析



编码与调优 高度可扩展的并行库与中间件 (DB2, WebSphere, MQ, PESSL, Amino, MASS...)
支持线程的调试建议工具(**也支持并行和硬件特性**)

构建 支持线程的静态分析工具
优化编译器(support parallelism, SIMD, trans memory)

测试与调试 并行调试器
线程分析

监控 剖析/数据收集 (线程化, 缓存, 通信)

分析 支持线程的性能分析工具

编码与调优 高度可扩展库与中间件 (适用应用程度模式，如 MapReduce / Hadoop, Streaming, TBB线程创建模块开源库)
调优建议工具（支持并行语言和分布式并行化）

构建 优化编译器与静态分析 （支持并行语言，如 UPC, CAF, X10)

测试与调试 并行调试器 (支持并行语言)
线程分析（支持最新的多核硬件与编程语言)

监控 剖析/数据收集（支持分布式并行）

分析 性能分析（支持分布式并行）

一些IBM在多核软件开发方面的战略

- 持续扩展中间件产品来开发新系统
  - 提高内核数目，使用同步多线程（SMT)，减少每个线程的缓存，使用非统一内存访问技术，解决带宽不足的挑战，使用加速器
- 持续开发新的并行编程语言
  - X10, UPC, CAF, Thorn, Java, OpenCL
- 持续开发新兴的编程模式
  - APGAS, Map-Reduce, Actor, etc
- 减慢开发全套性能调试工具
  - 起步阶段
    - 显式并行程序的程序员
      IBM中间件及其他运行时间
      独立软件开发商的代码
      客户的代码
    - 支持非专家进行串行性能工程
  - 研发新技术来支持非专家构建可扩展的并行程序


Questions


Thank
You