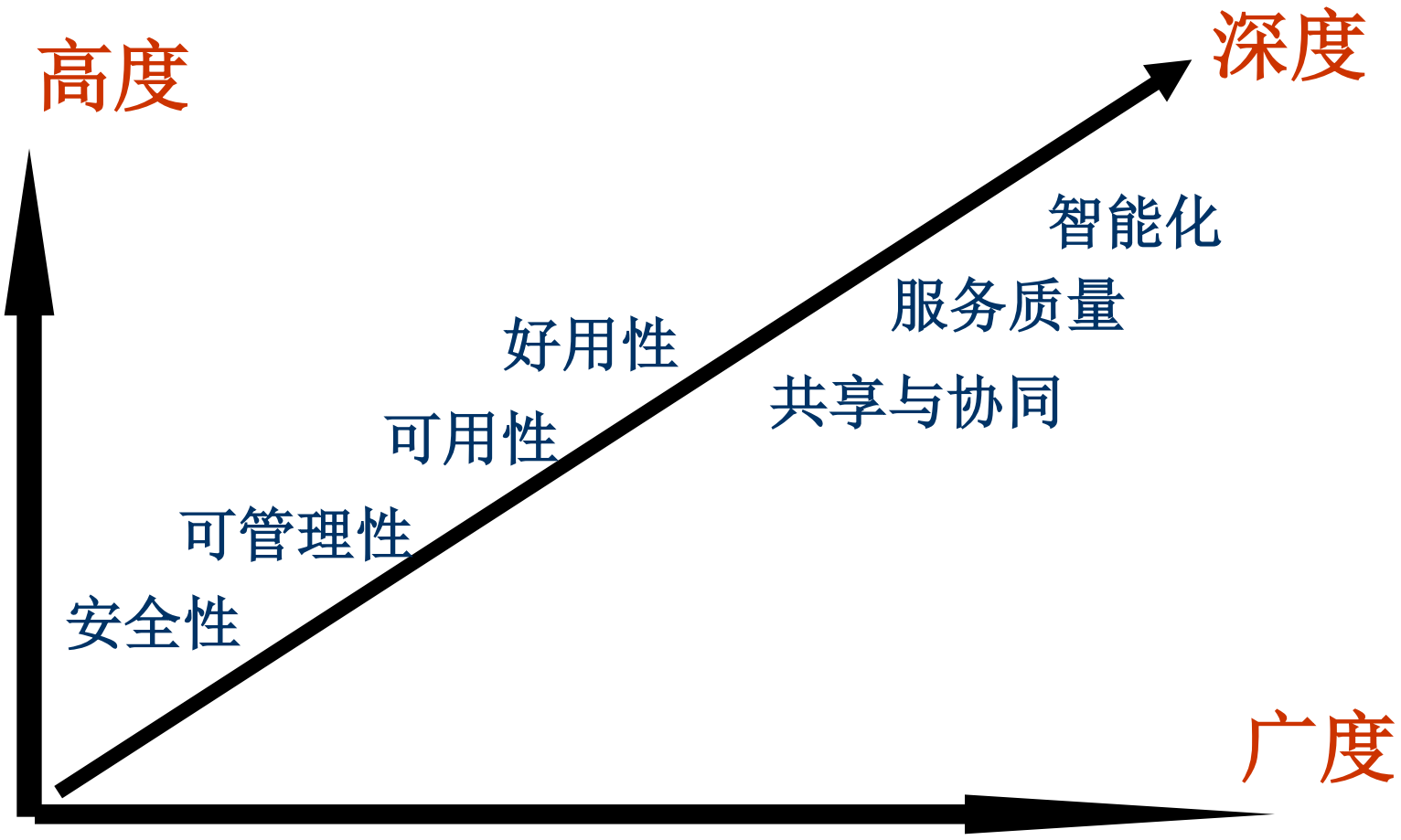
计算机系统的技术挑战

孙凝晖
中国科学院计算技术研究所
中国计算机大会，2010，杭州

国家信息化的高、广、深
都与计算机系统紧密相关

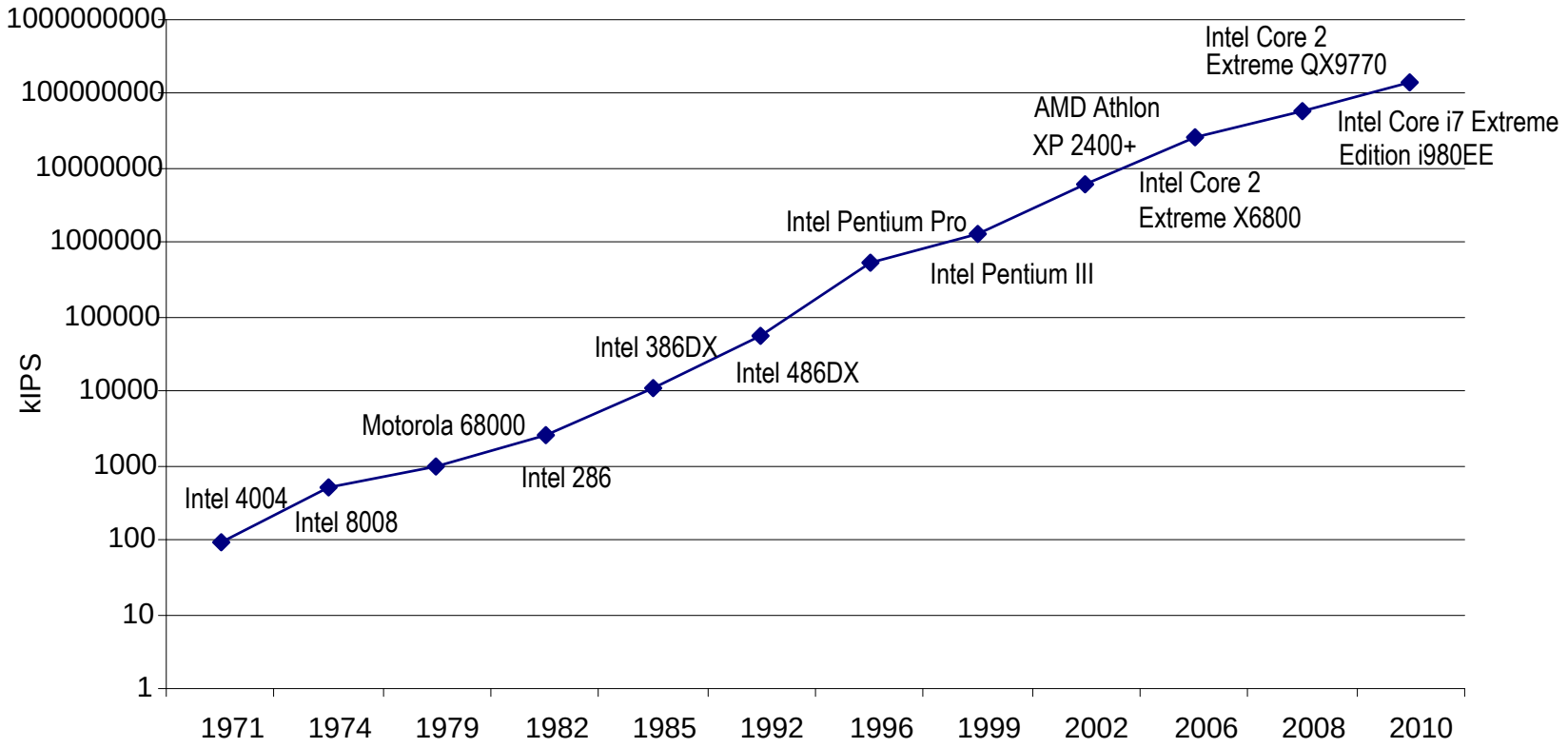
服务器、高性能计算机、存储设备的数量



电视、手机、PC、Internet用户的数量

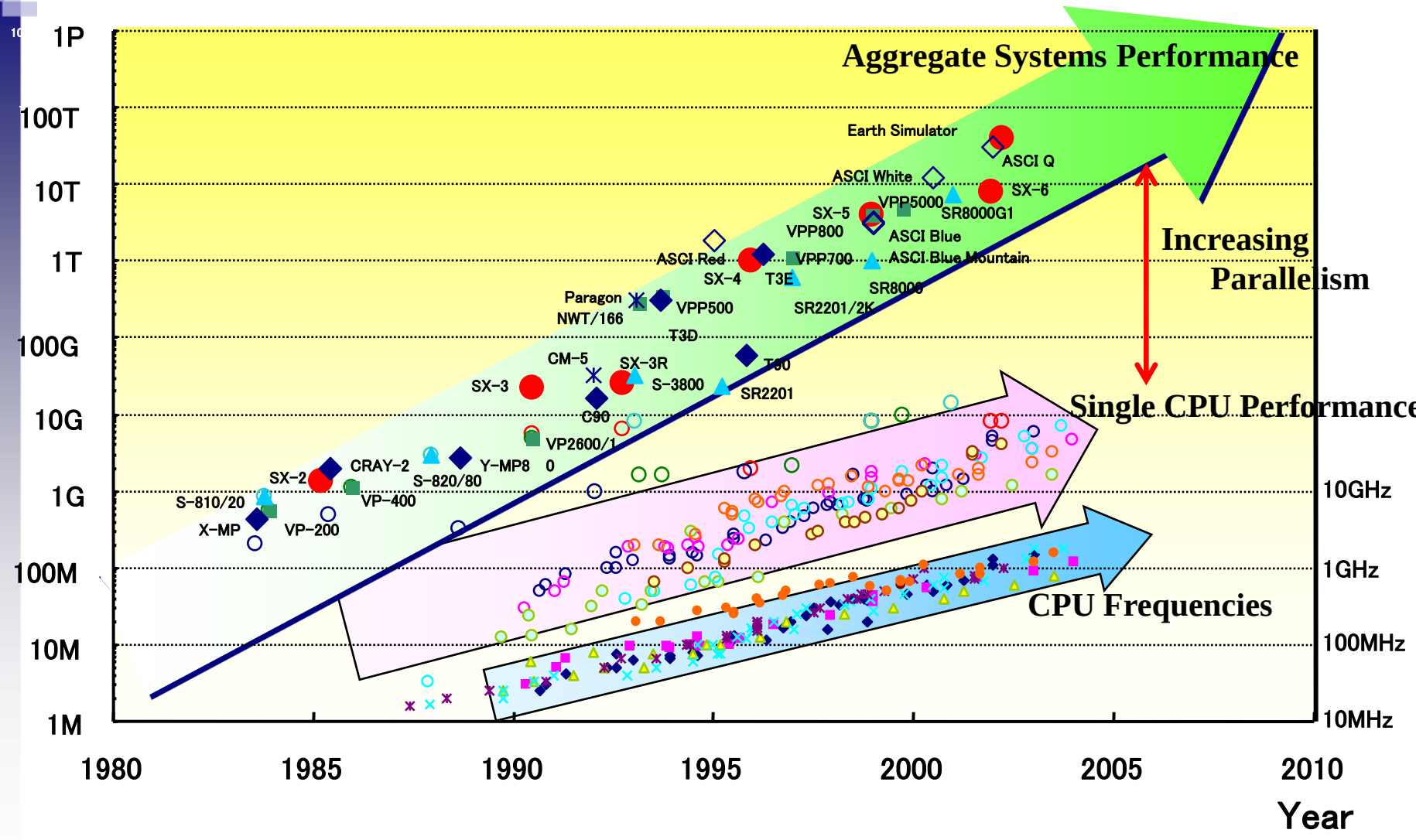
（10亿部电话 、2亿网民、8亿台计算机）

处理器速度持续提高（广度）



从1971年第一颗微处理器Intel 4004问世以来，
40年间处理器芯片集成的晶体管数目从二千三百个
发展到今天的数十亿个，处理器频率从不到1MHz
发展到今天最高接近5GHz，处理器的性能提高了数十万倍。

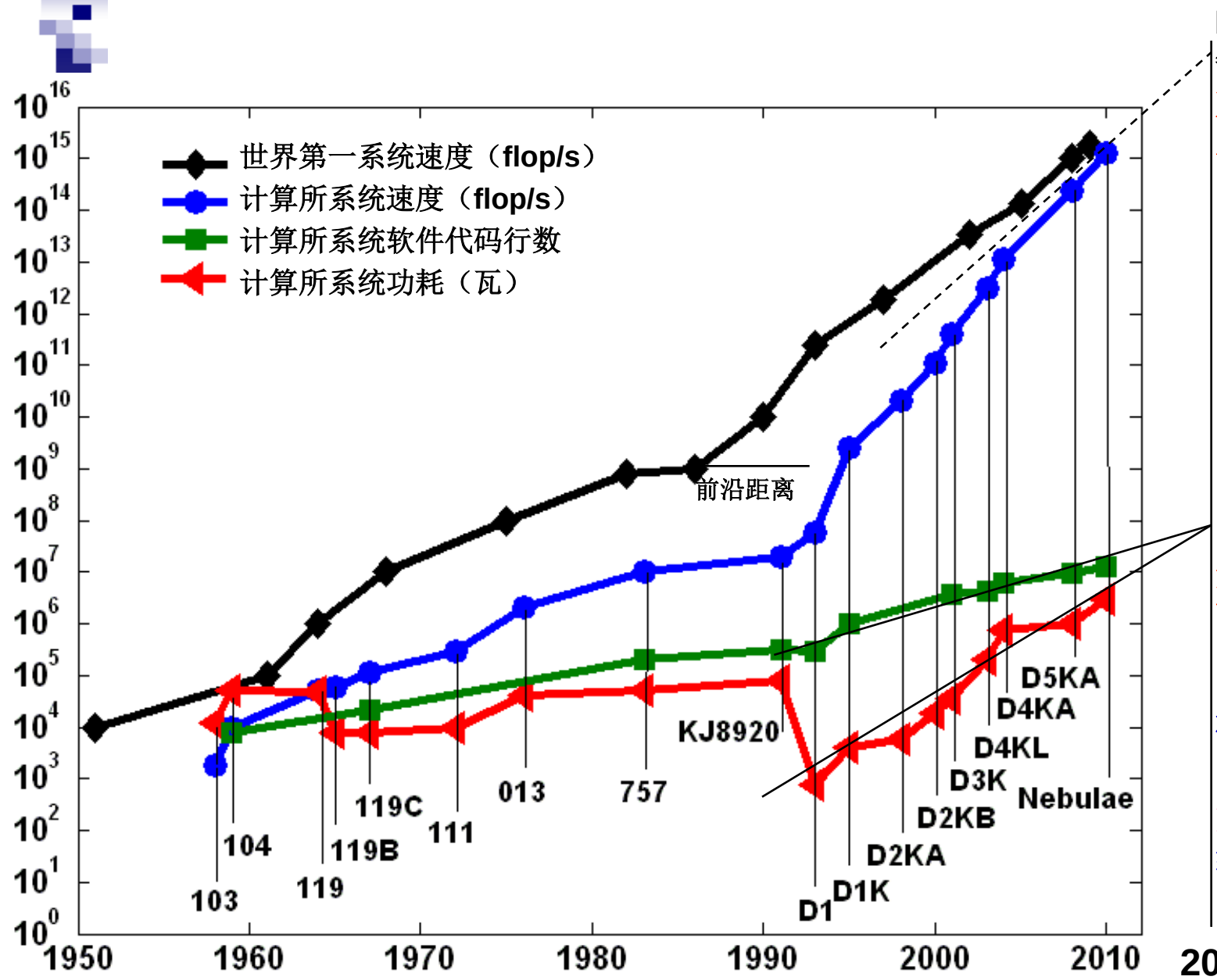
在30年间计算机系统的速度提高了6个数量级



曙光星云3000亿次超级计算机

世界排名第二，Linpack 速度1271万亿次

中科院计算所历代计算机



复杂性是计算机系统结构的第一个代价

- 五十年代：分立元件，结构比较简单
- 六、七十年代：集成电路，IBM 360/370和VAX 11/780采用了复杂指令系统（CISC）和动态流水线结构
- 八十年代：超大规模集成电路，将完整的处理器实现在单个芯片上，为了提高效率精简指令系统（RISC）出现
- 九十年代：摩尔定律为多发射的动态超标量、超流水线结构提供了资源空间，RISC结构复杂化

复杂性为性能付出高昂的代价

- 复杂的集中控制结构给芯片的布线带来了困难
- 给时钟频率的提高带来了困难
- 计算机主频的增长停滞下来
- 多核使处理器结构日趋复杂
  - 共享存储多处理体系结构
  - 分布式共享Cache的访问一致性协议
  - 片上互连网络
  - 多个内存控制器
  - 线延迟大于门延迟
  - 功耗、管脚增加
  - 芯片资源利用率降低、成品率降低

并行处理体系结构是通过增加复杂性提高性能

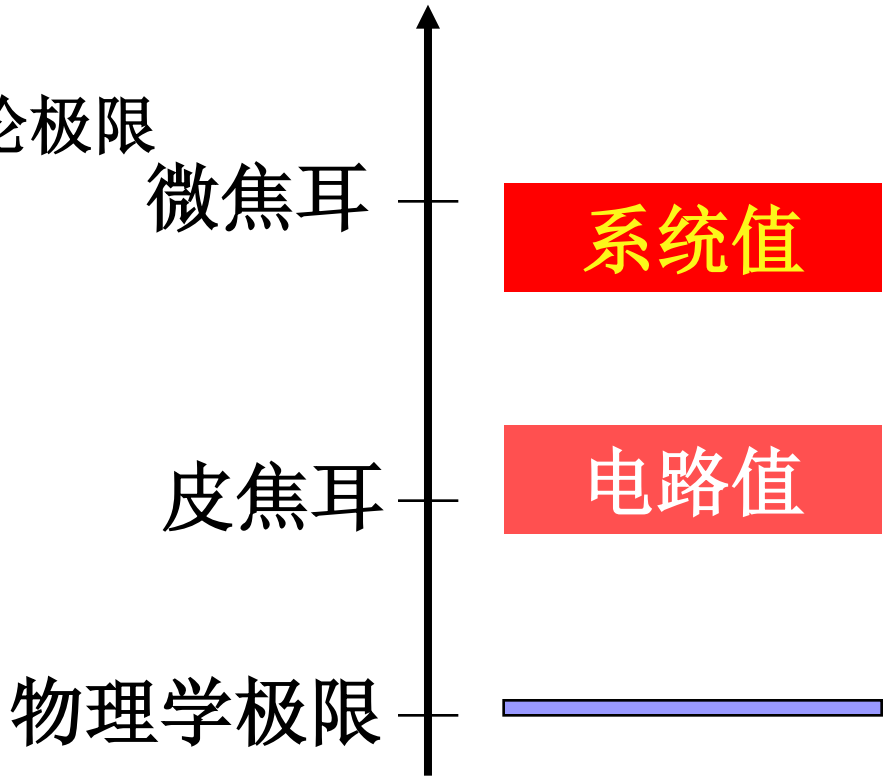
- 微处理器将并行处理计算机系统的复杂性带到片内
- 高性能计算机
  - 片内、节点内、节点间三级并行体系结构，每一级都具有一定的复杂度
  - 总的并行处理规模达到十万个处理器核以上
  - 在得到峰值速度的同时，大大增加了结构组装、系统管理、并行编程、算法设计的复杂性
- 并行处理结构与计算机科学的基本模型的不匹配
  - 计算机应用的PRAM串行程序模型与并行结构不匹配
  - 冯诺依曼体系结构的顺序地址空间与系统实现的分布式存储不匹配

系统结构的复杂性导致了功耗问题

- Google
  - 服务器每年$2亿元电费
- 计算机系统
  - 实际能耗远高于电路值和理论极限
  - 一次32位运算（皮焦耳）

电路级： 2
单机 DSP： 60
RISC： 200
PC： 2000
地球模拟器： 300000

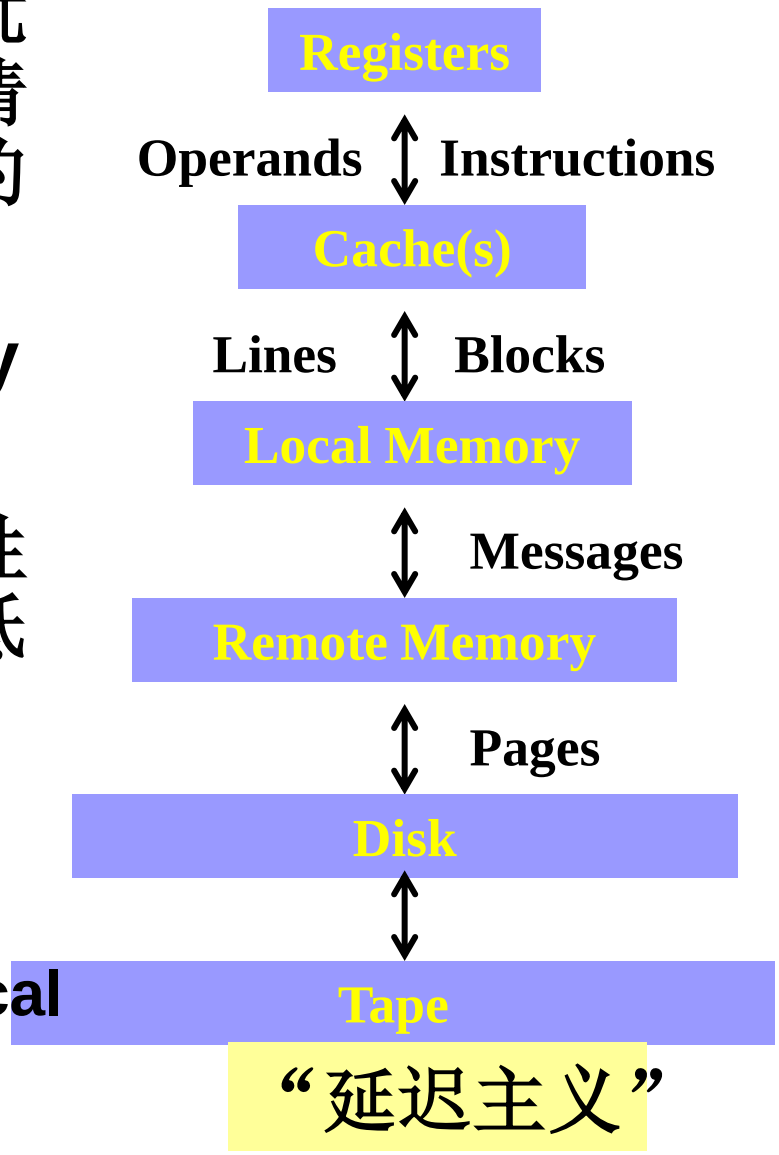
每个运算的平均功耗
（2000-2007数据）



用系统结构的简单降低功耗的例子：BlueGene/P

效率是计算机系统结构的第二个代价

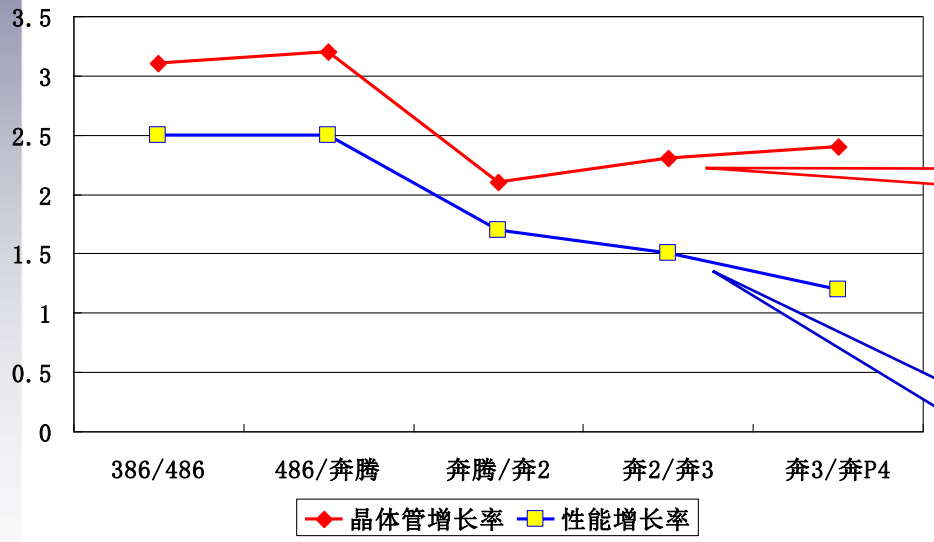
- 计算机应用效率不高一直是困扰计算机系统结构的问题，一般情况应用效率只有峰值运算速度的5%-20%
- 晶体管越多，存储墙（memory wall）越厚，应用效率越低
- 可扩展系统规模越来越大，高性能计算机的并行效率也越来越低
- 唯一的解决方法：Hierarchy
  - 处理器：Regs，L1，L2，L3，Memory
  - 高性能计算机：片内Buffer，Local M，节点内通信，节点间通信



晶体管越多存储墙越厚应用效率越低

安装时间	系统名	峰值速度	峰值比	应用速度比
2002	ASCI Q	20万亿次	1	1
2005	Bluegene/L	360万亿次	18	1.6-3.5

Adolfy Hoisie, Los Alamos National Lab, at Supercomputing 2007



晶体管增长
2.3倍

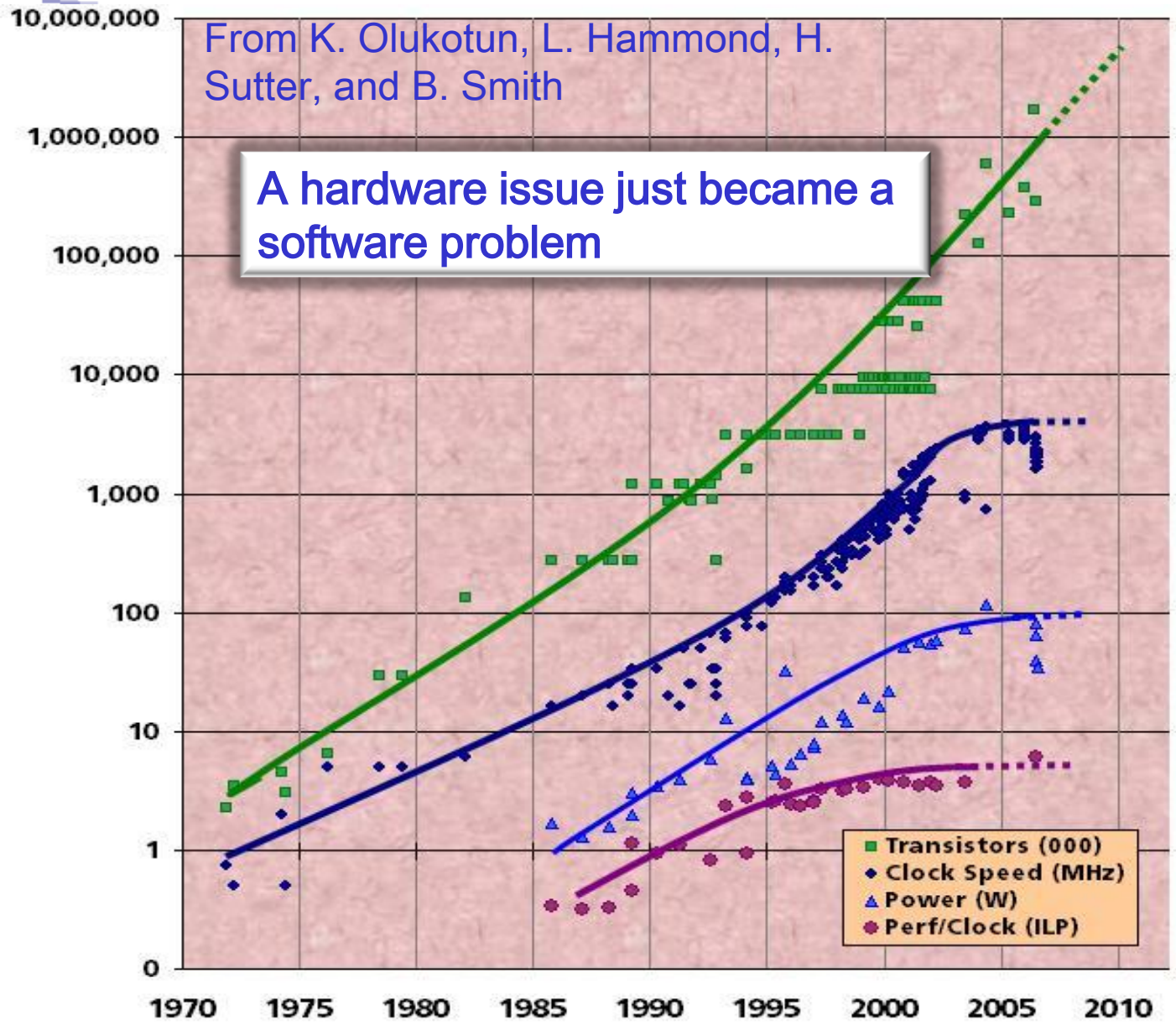
性能提高
1.5倍

Pollack's Rule：性能增长速率大致是晶体管资源增长率的平方根

通用性和效率之间寻找一个平衡点

- 通用处理器：为了满足多种应用的需要，把功能设计得面面俱到，许多逻辑忙碌地工作却不产生真正的应用效率，导致很高的功耗
- 处理器开始走细分之路
  - 嵌入式应用：**ARM、龙芯1**
  - 移动Internet应用：**Intel Atom、龙芯2**
  - 普及应用：**X86-64**
  - 企业数据处理应用：**IBM Power6、Itanium**
  - 网络应用：**Sun Niagara**
  - 游戏应用： **IBM Cell**
  - 图形处理应用：**Intel Larrabee、Nvidia GPU**
  - 科学计算应用：**Sun Rock、ClearSpeed、龙芯3、DSP**

并行编程是计算机系统结构的第三个代价



1Eflops

100Pflops

曙光6000

曙光5000

曙光4000

曙光3000

曙光1000

曙光一号

并行度

并行处理加剧了计算机在编程性和效率上的矛盾

计算机性能的提高伴随着编程效率和易编程性的提高

机器语言 ➡ 汇编语言 ➡ 高级语言(Algo、Pascal、Fortran)

面向性能的过程语言 (Basic、C) ➡ 面向程序员生产率的对象语言 (Java、C#)

- 随着并行处理结构的出现，计算机技术越发展，编程越困难、效率越低
  - pthread
  - OpenMP
  - MPI
  - Data Flow语言
  - PGAS语言
  - HPCS语言

计算机技术越发展编程越困难

Parallelism for the Masses 以后没有不是并行的计算机了

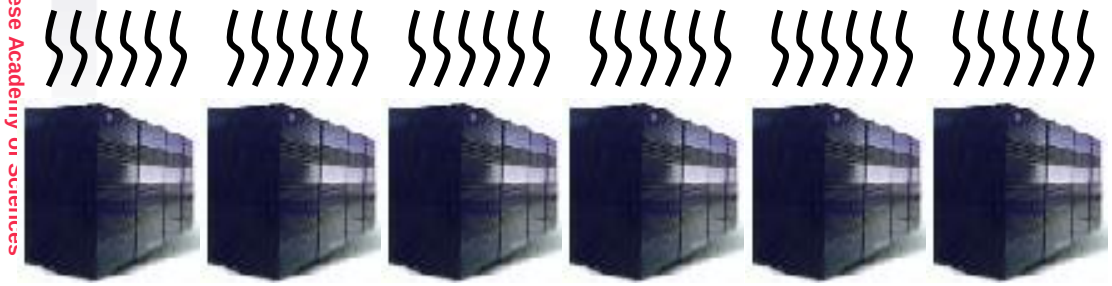
- 面向大众的并行编程模型？
- 10亿程序员用的并行编程语言是什么？

“如果数据中心是计算机，什么是这个计算机的加法指令？”

David Patterson

***CACM* 2008.1**

如何编程让数据中心的上万个线程高效工作？



- **BigTable**
- **MapReduce**
- **GSML**

可靠性是计算机系统结构的新代价

可扩展系统的规模不断增长，使得高性能计算机的平均无故障时间可能小于一个真实应用的运行时间

- 摩尔定律的延续使芯片集成了数十亿个晶体管
- 缺陷密度增加
- 芯片的成品率急剧下降
- 需要从缺陷容忍、故障容忍和差错容忍三个层次研究芯片从故障中自动恢复的方法

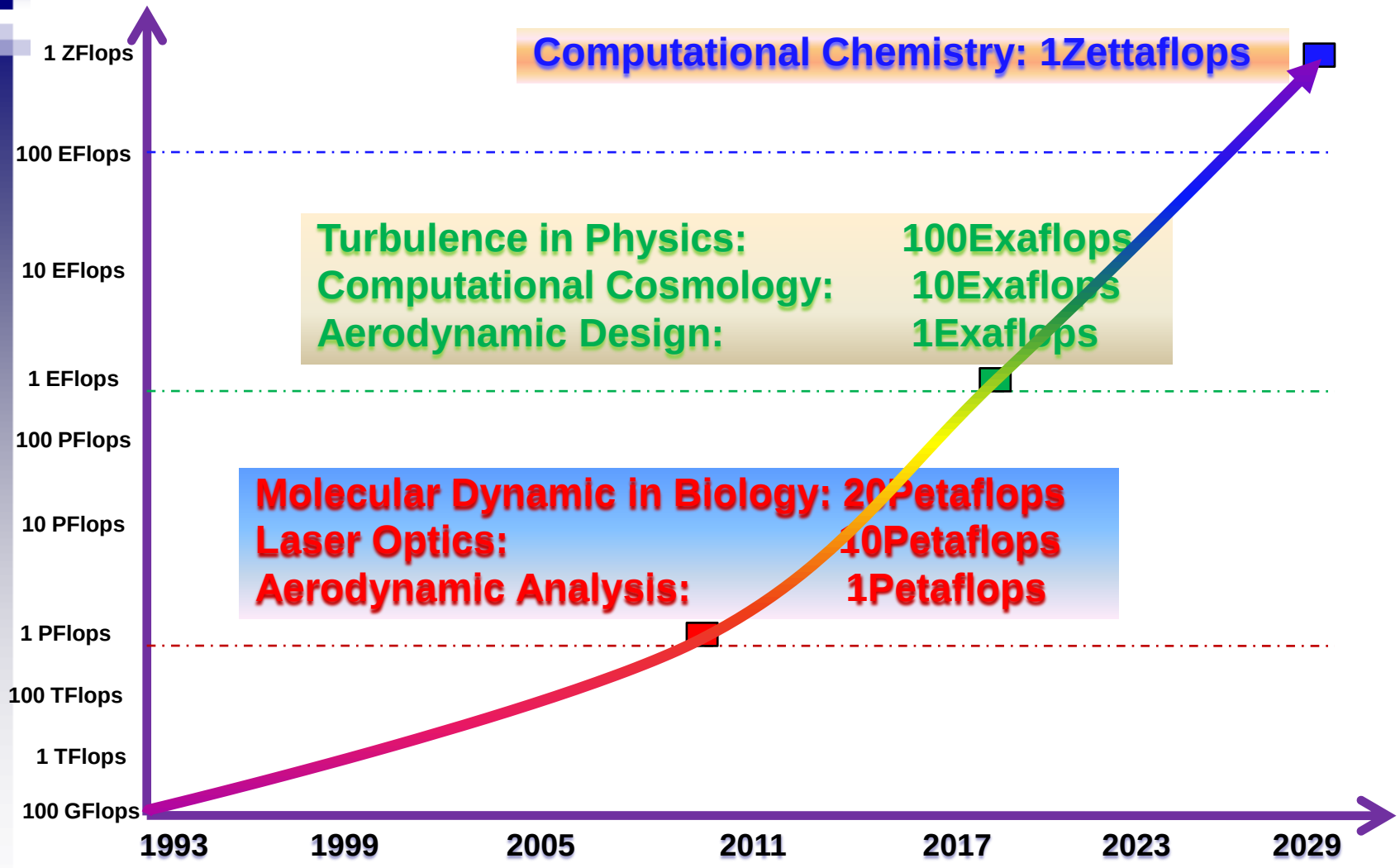
- 制造过程中工艺参数的涨落、内部原子级效应引起器件参数离散性不断增加
- 越来越薄的栅氧化层导致隧穿的机率不断增加
- 越来越多的片内存储器使单粒子效应等因素导致的软失效越来越多

What is computing good for?

— Butler Lampson

Simulation **IT1.0: 模拟时代**	**1950-today**	通过数学模型模拟物理世界和人类活动，进行科学规律、工程实现、经济和社会规律的仿“真”
Communication (storage) **IT2.0 : 数字社会**	**1980-today**	通过移动数据(communication)，将人类社会“真实”的工作与生活(talking, reading, writing, watching, selling, buying, management …)移到Cyberspace（网域）
Embodiment (physical world) **IT3.0 : 泛在社会**	**2010-future**	通过具有一定的人的智能(事先无法预知)的计算机算法，让计算设备最终能够隐形地长入到“真实”的物理世界而无处不在

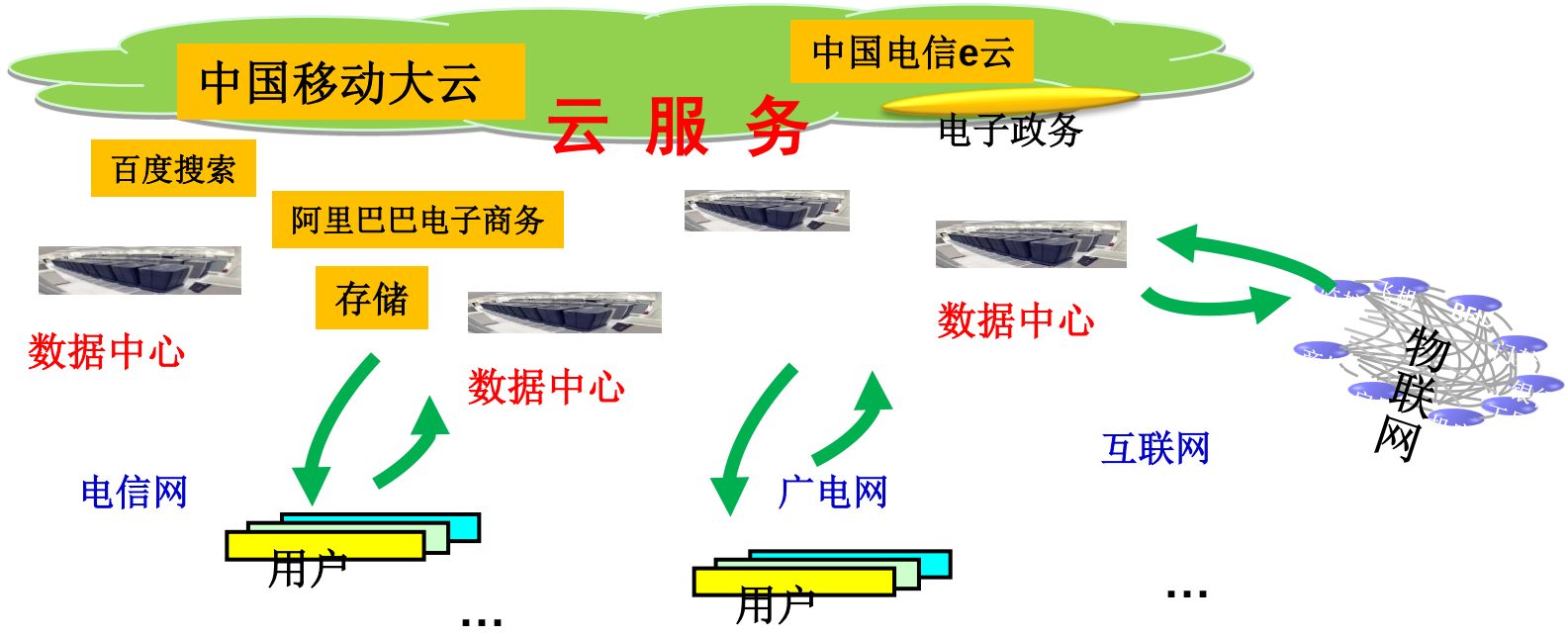
对计算能力的需求持续增长



以计算推动科学研究为例

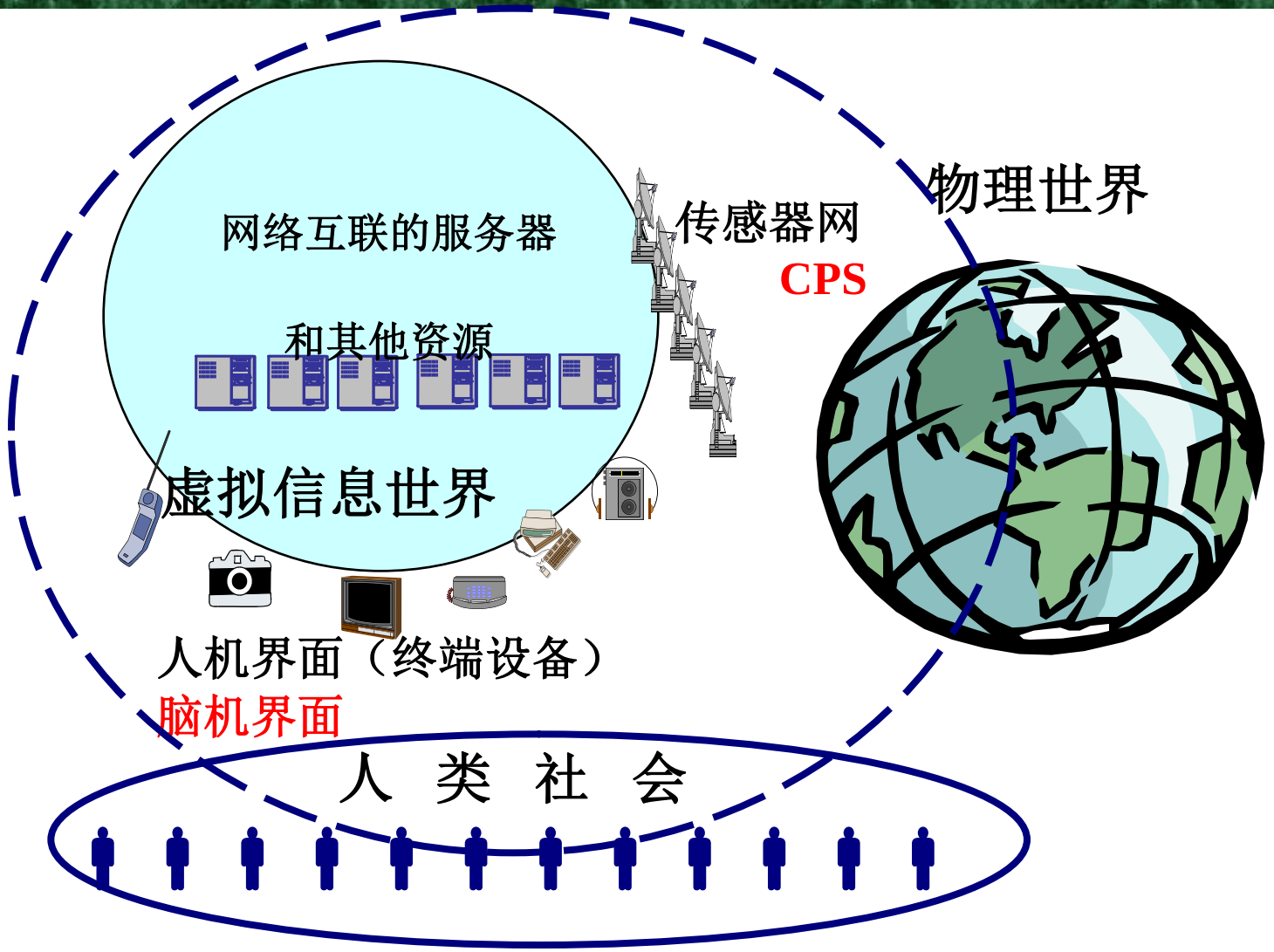
source: isc'06

云计算是计算机和通信技术与产业发展的大趋势



- 以**集中服务、虚拟化**和动态调度资源为主要特征的**云计算**具有**降低信息化成本和能耗**的优势，符合信息技术集中/分布模式波浪式发展的规律，是我国信息化的**必经阶段**。

物理世界、信息世界、人类社会组成三元世界—新信息世界观

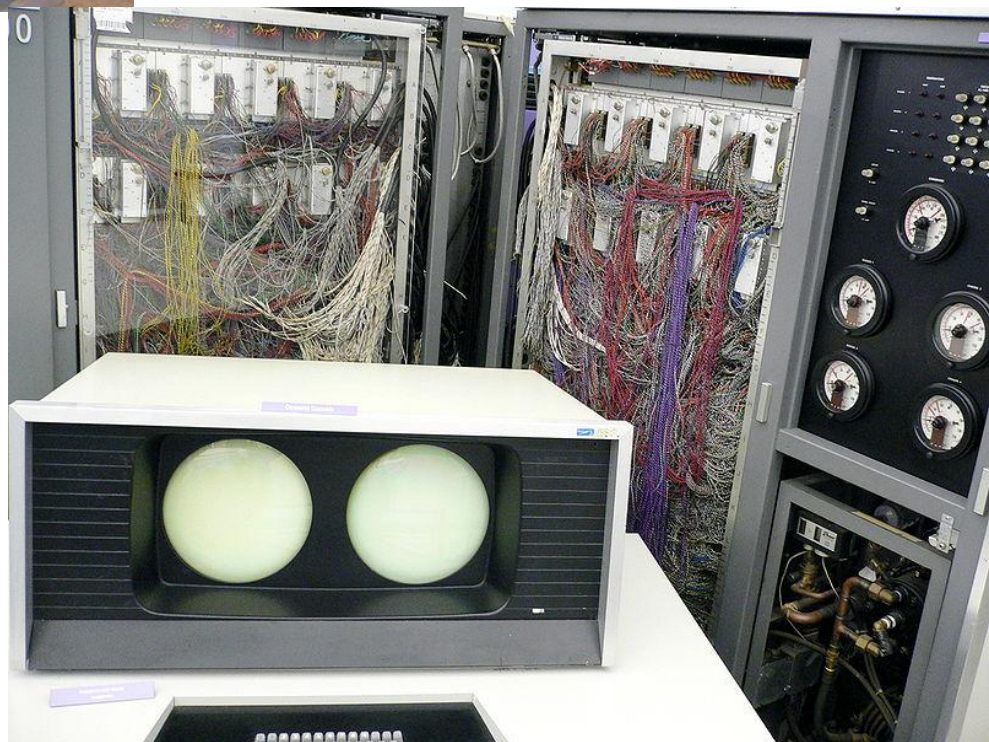


计算机体系结构的技术起源 (1960s)

- 1960年代的IBM System/360 (1964~1969)
  - 兼容机理念(相同指令集、数据格式)
  - 通用机理念(整合科学计算与商业应用)
  - 提出了“以8bit为1Byte”
  - 乱序执行技术(Tomasulo算法)
  - Cache技术
  - 标准外围设备接口
- CDC 6600/7600 (1964~1969)
  - 乱序执行技术(Scoreboard)
  - 流水线技术
  - 独立浮点运算单元

← IBM System/360

CDC 6600 →



并行体系结构的技术起源 (1980s)

- SIMD结构、MPP结构
  - ILLIAC IV (1976)
  - TM (1982)、CM-1 (1983)
- 并行向量机
  - Cray-1 (1976)、Cray X-MP (1982)、Cray-2 (1985)
- 互联技术
  - Wormhole路由 (1987)
  - FatTree结构 (1985)、 CM-2's HyperCube (1987)
- 存储一致性
  - MESI协议 (1983)

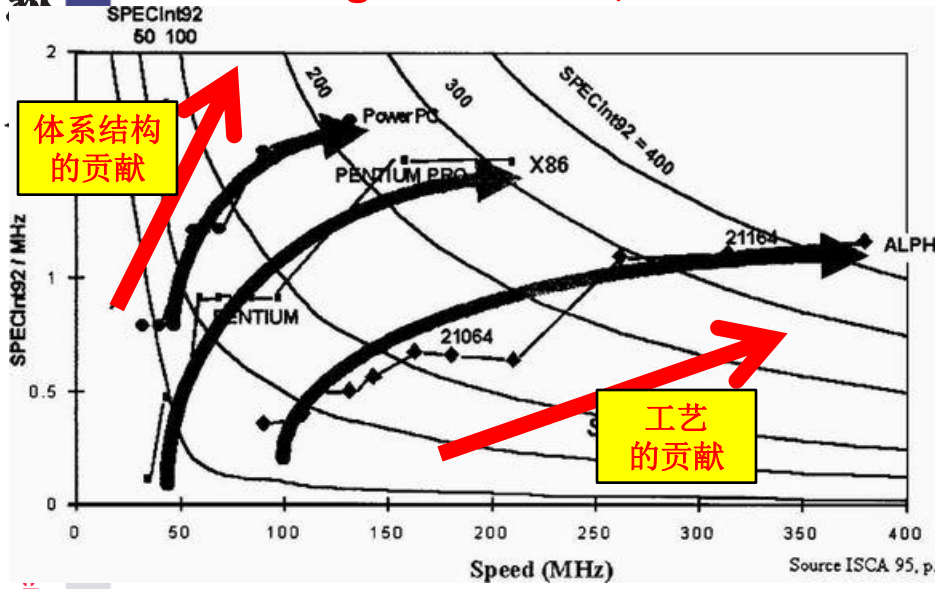
↑

ILLIAC IV

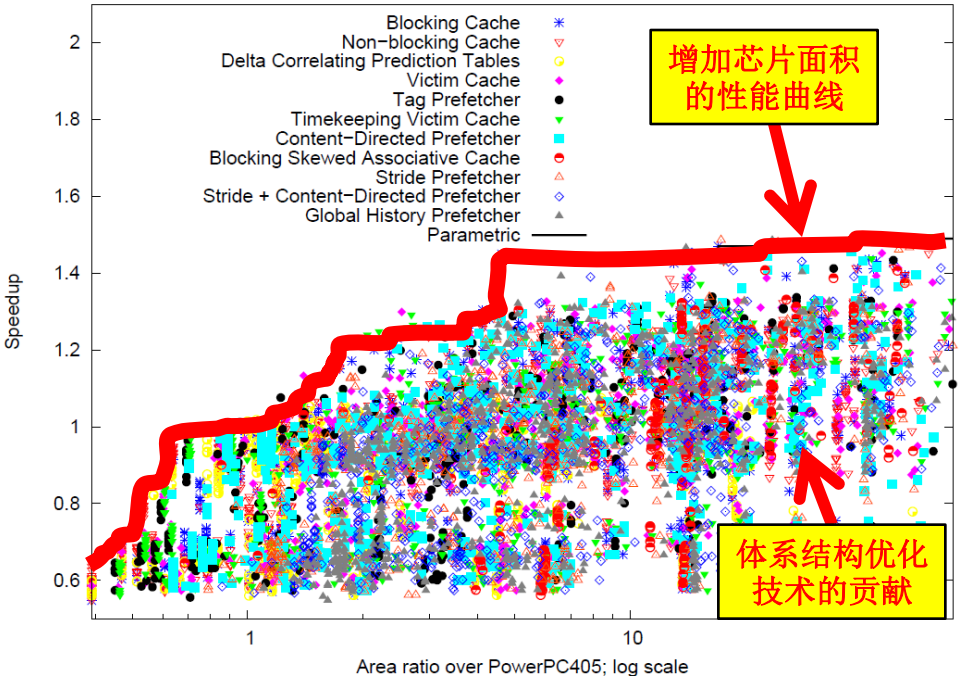
Cray-1 →

is architecture dead?

R. Ronen, et.al, Coming challenges in microarchitecture and architecture, Proceeding of the IEEE, 2001



Olivier Temam, INRIA, www.archexplore.org



- 从1990年代中期开始，工艺提高已经开始对计算机性能增长的贡献占主导地位

- 最新研究发现，近二十年来对处理器Cache的优化技术竟不如“增加芯片面积+调节参数”简

一些观点

Microprocessors are dead, … to be replaced by computers built onto a single chip.

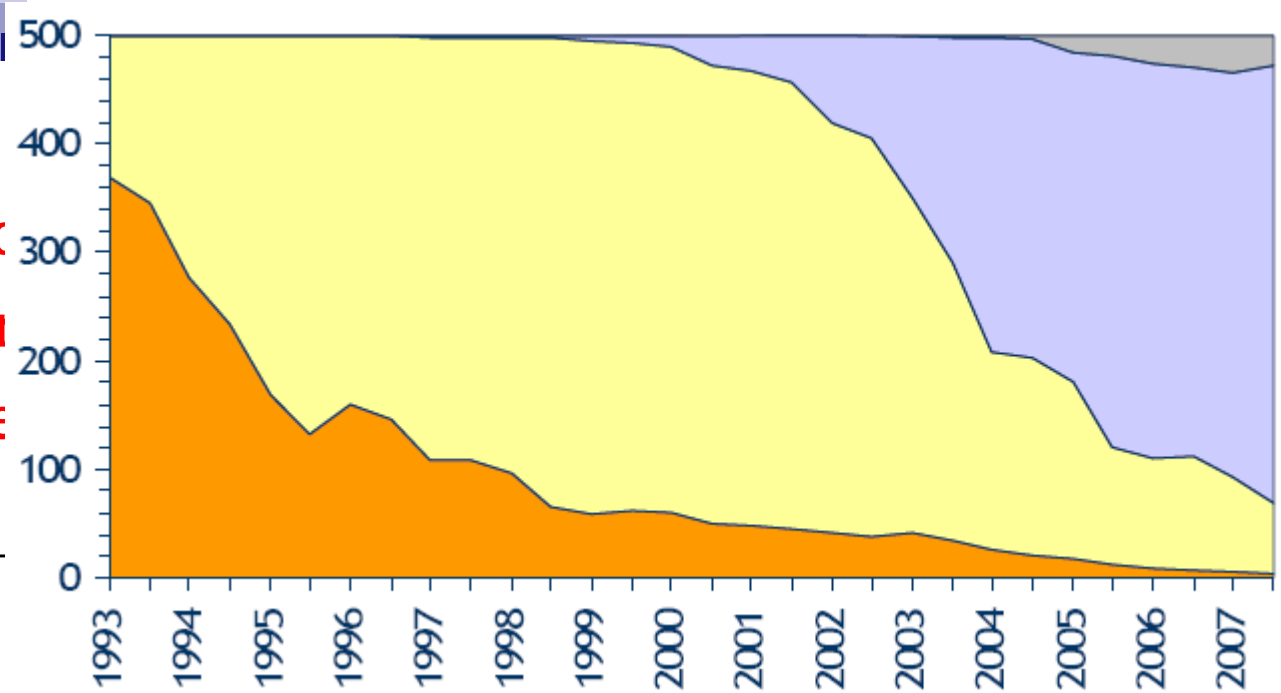
- ***Greg Papadopoulos, Sun CTO, 2003***

Processor is the new transistor!

- Intel 4004 (1971): 2312个晶体管
- RISC II (1983): 40,760个晶体管
  - 3um工艺 → 60mm²
  - 65nm工艺 → 0.02mm²
- 125 mm², 65nm工艺的芯片 = 2312个RISC II

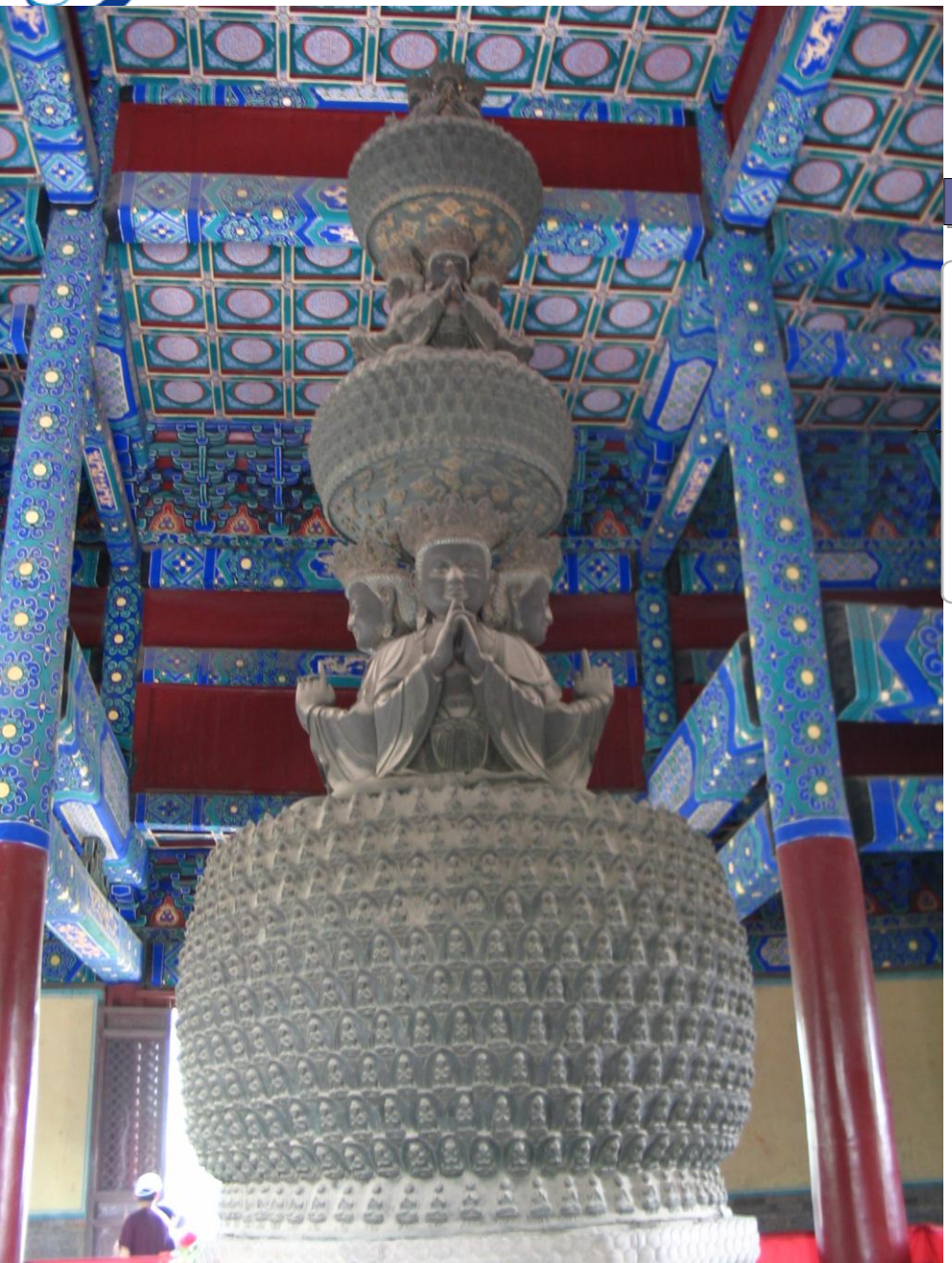
- **Multicore architecture looks more or less like the hardware designers have run out of ideas.**
  - *Donald Knuth, Stanford University,2008*

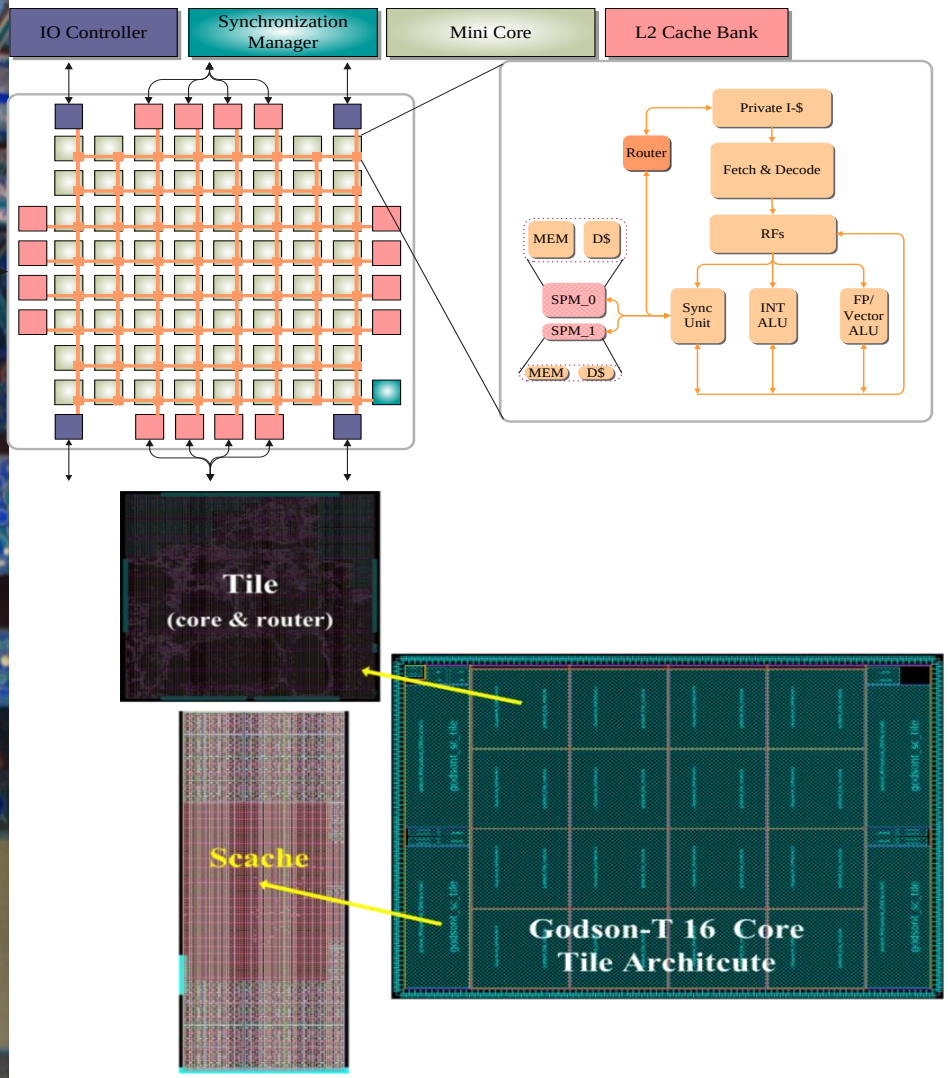
Data
Custo
Comi
Powe



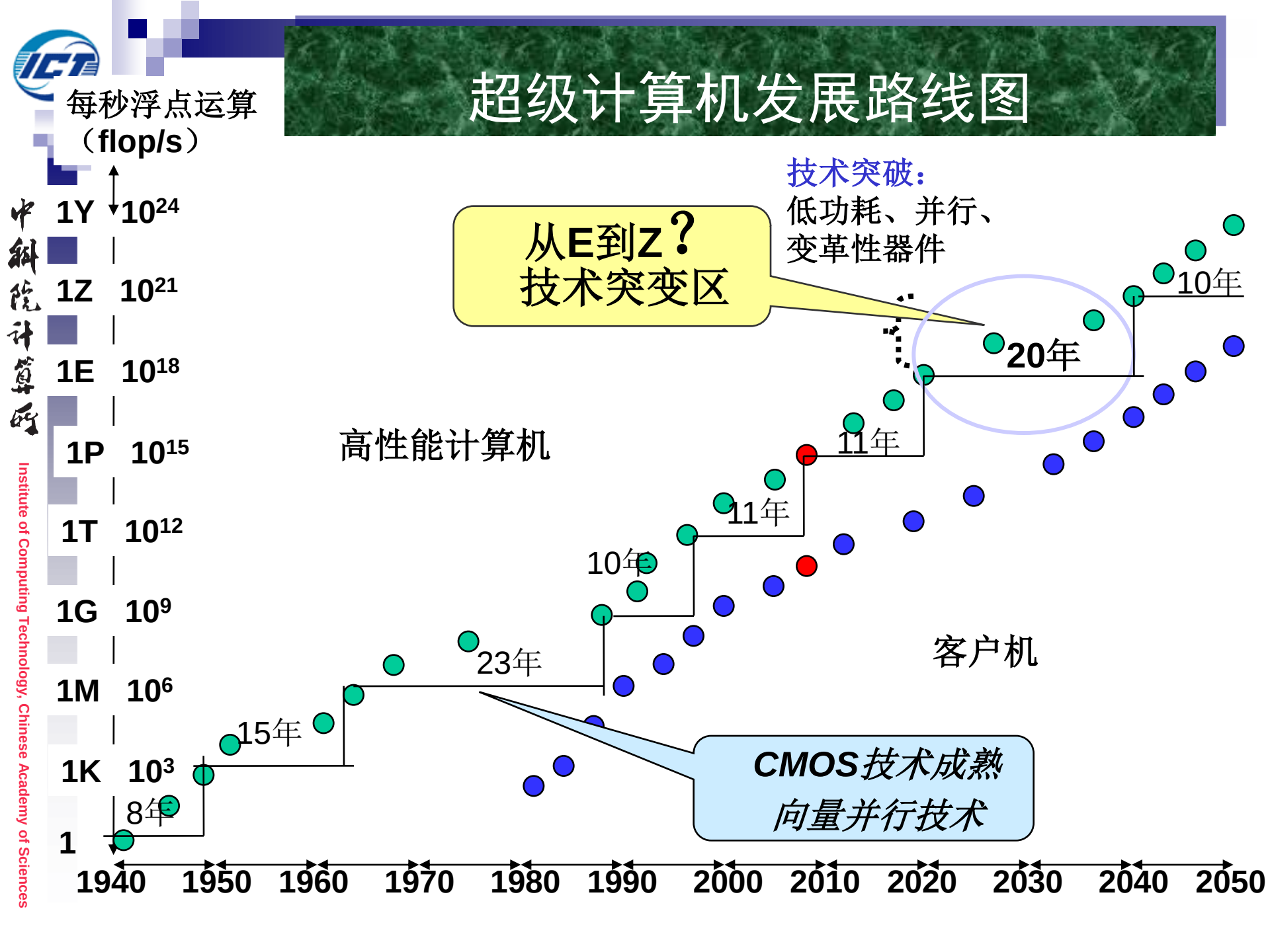
Class			
Data parallel systems	Mid 70's	Mid 80's	Mid 90's
Custom scalar systems	Mid 80's	Mid 90's	Mid 2000's
Commodity clusters	Mid 90's	Mid 2000's	Mid 2010's ???
BG/L or BG/P	Mid 2000's	Mid 2010's ???	Mid 2020's ???

← Many Integrated Core (MIC)?





超级计算机发展路线图

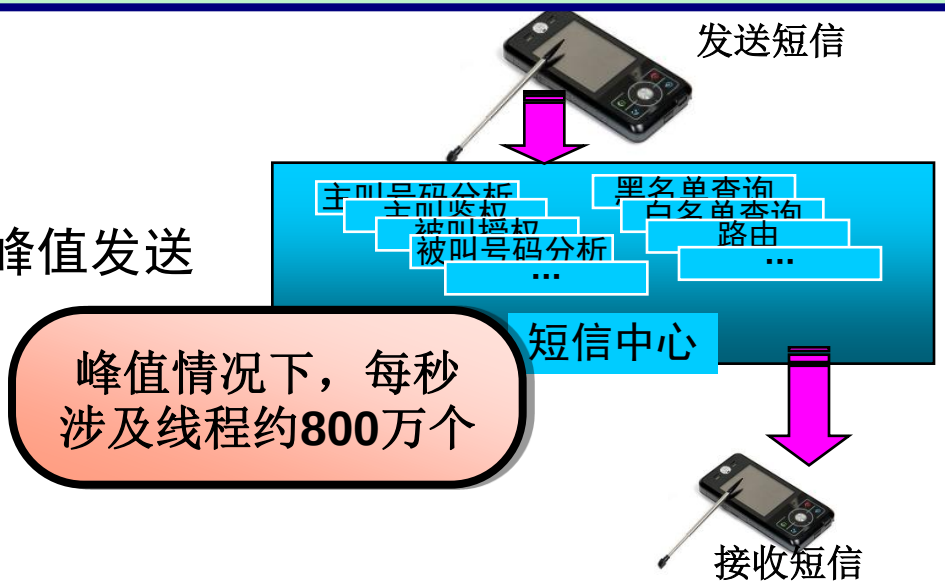


海量数据和高通量计算是新兴的应用负载

并发用户将从目前的数千万发展到数亿规模，数据规模从目前的10PB(10^{16}B)量级发展到EB(10^{18})量级，数据中心面临吞吐能力能耗、服务质量保证等巨大挑战

中国移动短信服务

- 2009年发送6812亿条，每秒峰值发送约40万条
- 处理一条短信涉及20个线程

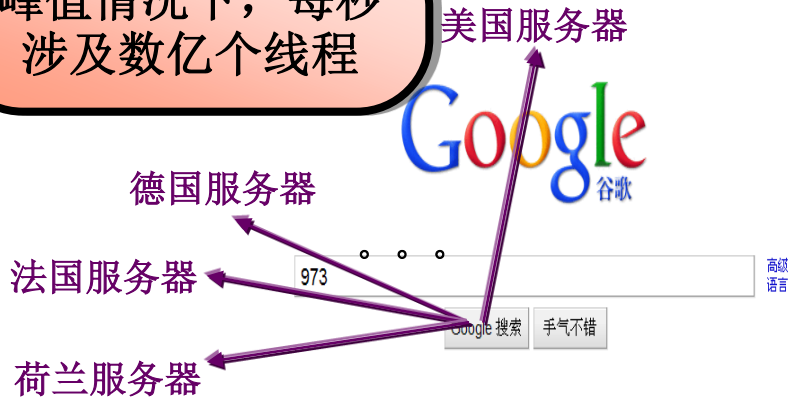


峰值情况下，每秒涉及线程约800万个

Google搜索服务

- 每个搜索派发数千个线程
- 每秒全球有上十万次搜索
- 全球部署超过100万台的服务器
- 搜索引擎服务器上的文件大小为TB~PB数量级

峰值情况下，每秒涉及数亿个线程



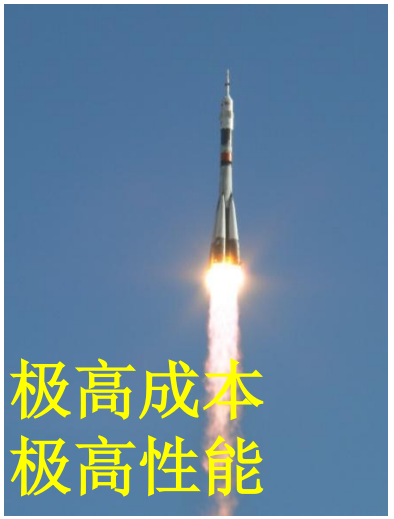
何谓高通量？

低延迟

高带宽

通用

高通量



极高成本
极高性能

专用

极低成本
极低性能

需要研究满足云计算数据中心需求的新一代计算系统

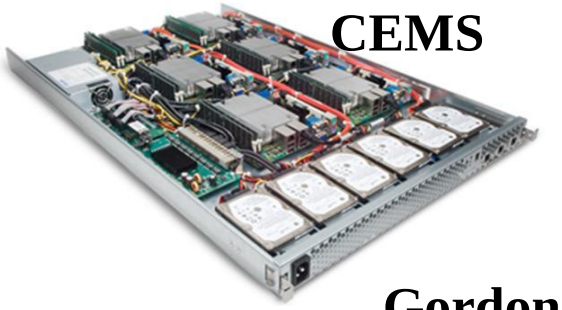
	应用负载特点	任务的并行度	性能成本要求	可靠性要求	性能目标
下一代数据中心需求	● 网络服务 ● 海量规模：EB级数据，亿级并发 ● 任务多样 ● 局部性差：服务内相关性高，不同服务间相关性低	固有充足的线程级并行性	成本决定生存和收益	单部件失效影响不大 不同性质数据有不同的可靠性要求	高通量：提高单位时间内处理的并发任务数目
高性能计算系统	● 科学和工程计算 ● 任务单一 ● 局部性好：计算集中于若干核心任务	需挖掘可用的并行性	优先追求性能兼顾成本	单部件失效可能导致计算停顿 需检查点和恢复	高速度：缩短单个并行计算任务的运行时间

已有的一些尝试

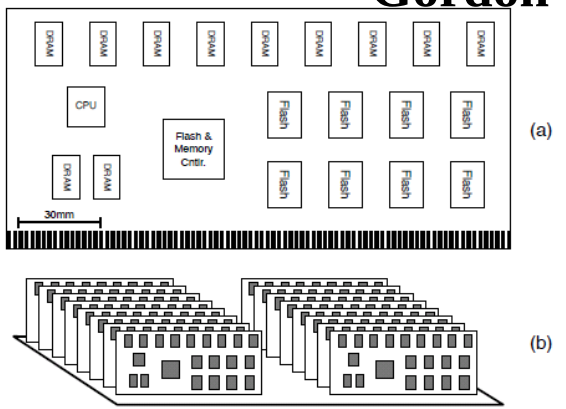
参考系统：CEMS（CIDR'09）、Gordon（ASPLOS'09）、FAWN（SOSP'09）

- 低成本、低功耗、低性能的CPU
- 高成本、低功耗、高性能的SSD
- 定制OS、新的设备驱动、新的buffer cache、新的local FS

CEMS



Gordon

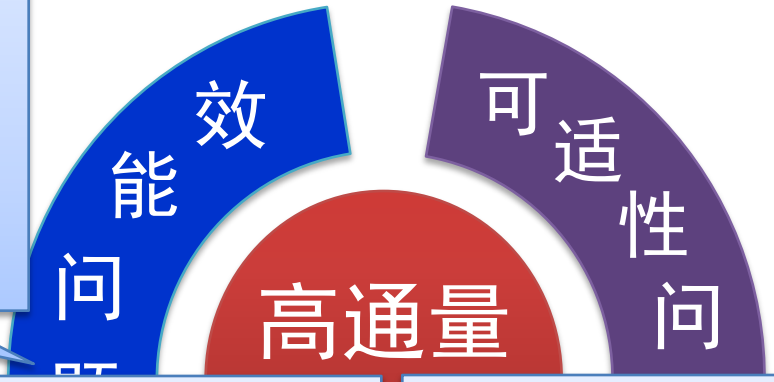


- ✓ CEMS：真实Web Server负载下，RPS/dollar 3.7x, RPS/Joule 3.9x, RPS/Rack 9.4x
- ✓ Gordon：19w/server (vs. 81w/传统server)，能 1.5x , 性能/ watt 2.5x
- ✓ FAWN: 6w/node, 21个节点36,000 QPS，364 queries/Joule (vs. 磁盘2.3 queries/Joule), 100x

FAWN

关键科学问题一：效能问题

海量并发处理单元的**结构、执行模型**，支持高通量的系统**数据通道和存储模型**，从芯片到系统和应用的功耗控制方法



现状：

1. 输入输出“函数”式的计算理论，不能很好刻画互联网“不停机”交互式并发过程
2. 以任务执行时间为最主要目标，难兼顾海量数据处理和高并发
3. 效能低，能量未得到有效利用

关键问题：

1. 严格的并发计算数学理论
2. 计算系统中的能量复杂性理论
3. 数据传输和能量使用中的有效性甄别和去冗
4. 请求驱动的并行微结构和高并发处理的系统结构

关键科学问题二：可靠性问题

现状：

1. 海量节点中单节点失效常态化
2. 依赖多模冗余在性能、吞吐和能量上代价较大
3. 缺乏系统科学中的“整体优于部分之和”的原理保障

关键问题：

1. 故障在不同层次传播效应模型和分级
2. 故障的检测、隔离、处理、自愈等技术
3. 不同层次故障处理的协同

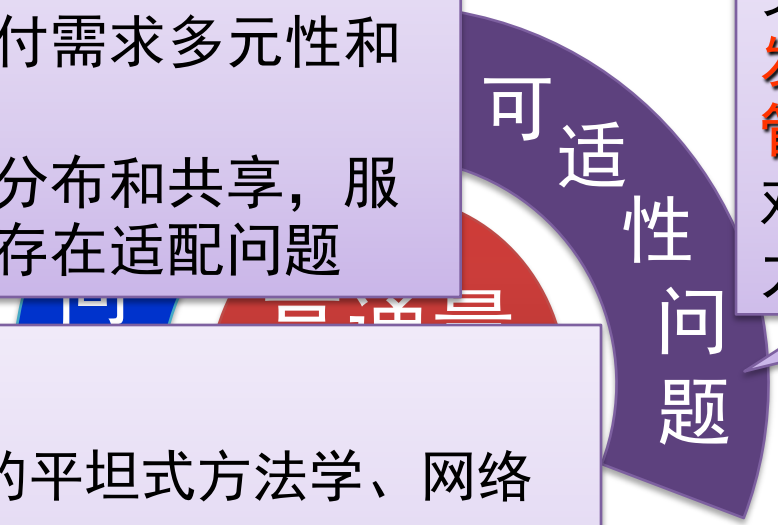
可靠性问题

从芯片到系统和应用的**故障自我管理原理和方法**，在故障为常态情况下，提供有质量保证的高通量服务

关键科学问题三：可适性问题

现状：

1. 计算系统结构和资源模式提供的复杂难控，难应付需求多元性和动态变化性
2. 资源和服务难以分布和共享，服务与运行环境间存在适配问题



面向复杂、动态的业务环境，支持**亿级并发编程模型、云资源管理模型**，增强服务对运行环境的适应能力，提升服务质量

关键问题：

1. 克服计算机系统的平坦式方法学、网络系统的协议栈层次方法学和服务参与者交互方法学之间的矛盾，建立资源消耗与功耗限制下的**可持续网络服务理论**
2. 满足个性化和时变性需求的移动服务按需聚合、智能协同机制
3. 高并发、分布编程模型和数据处理

可能的创新：高通量处理芯片

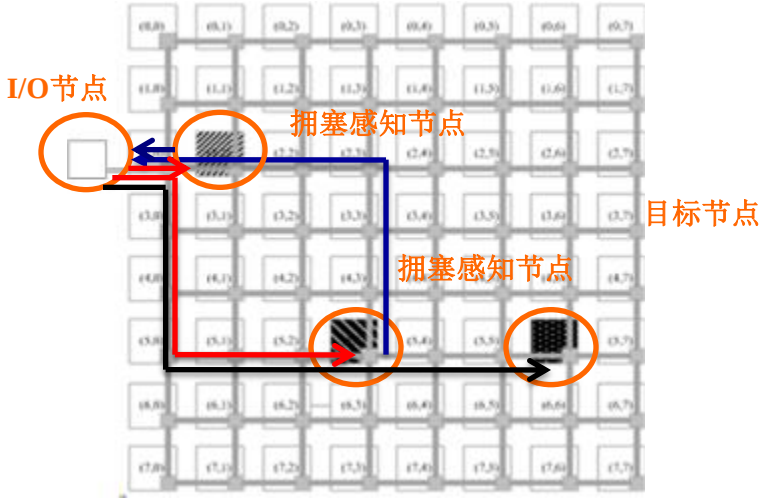
- 单片千线程
  - 高通量处理芯片所支持的线程数与当前支持多线程的芯片相比，**提高近2个数量级**

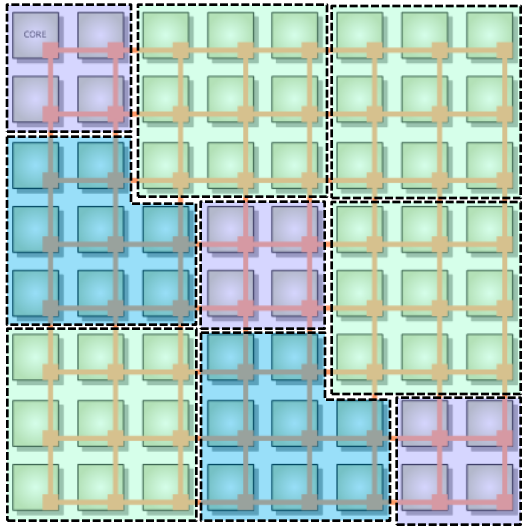
处理器	最大线程数	物理结构
AMD Opteron	12	12个处理器核
Intel Xeon	8	2~8个处理器核
Intel SCC	48	48个处理器核
IBM Power7	32	8个处理器核
IBM Wire-Speed Processor	64	16个处理器核，集成了很多的IP单元，如硬件加速单元
Sun UltraSparc T2	64	8个处理器核，每个处理器核支持8个硬件线程
Tilera TILEPro	64	64个处理器核
高通量处理芯片	1024	单芯片最大支持1024个线程

表: 当前商业多线程处理器与高通量处理芯片的对比

可能的创新：高通量处理芯片

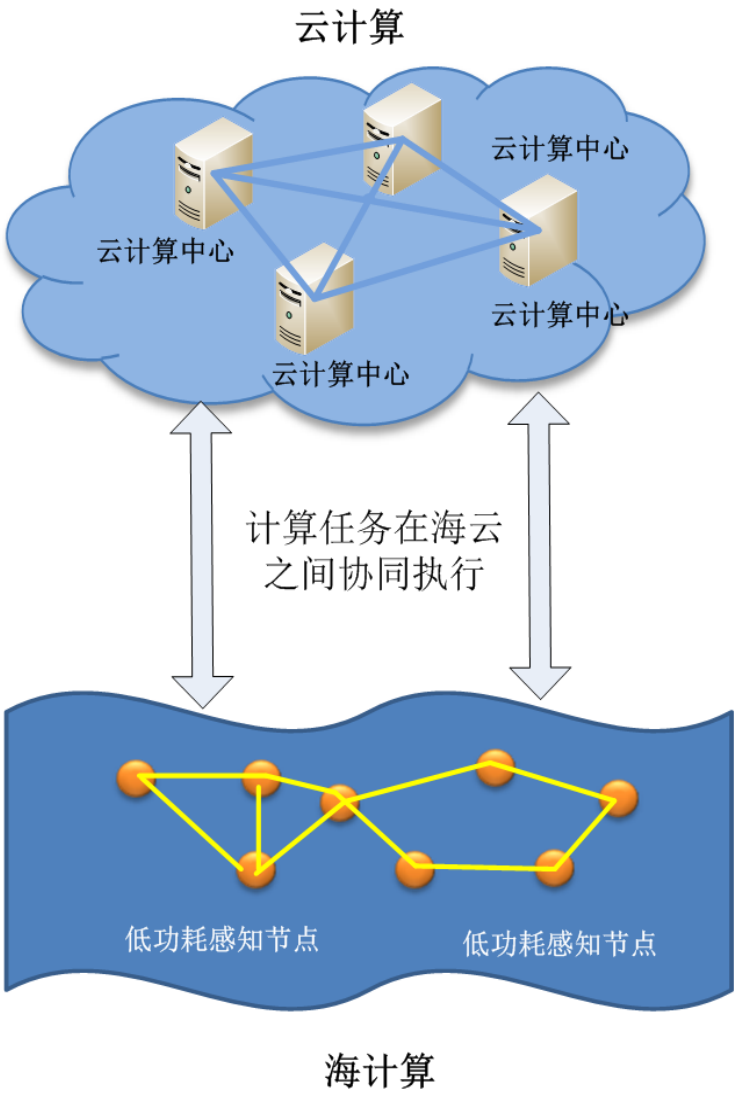
- 指令系统扩展，如：
  - 动态指定片上互连路由策略的指令，强化处理芯片的I/O数据处理能力
- 新型硬件特征
  - 利用高通量线程之间相似性，提高片上存储资源复用的能力
  - 片内共享资源或处理器核的分区管理机制





海计算

- 以物理世界为中心的思维
  - 强调物理世界的智能交流 **(interaction of things)**
  - 强调物理性质涌现**(physical emergence)**
- 四个特征
  - 融入性（**embodiment**）：信息装置融入物体
  - 自主性：物体不只是被动地被控制，而具有自主性（**autonomous and autonomic**）
  - 局部交互：充分利用局部性原理
  - 群体智能：分布式交互产生智能



海计算的潜在优点

- 节能高效。充分利用局部性原理，可以有效地缩短物联网的业务直径，即覆盖从感知、传输、智能决策到控制的路径，从而降低能耗，提高效率。
- 通用结构。融入信息装置的“自主物体”的引入，有利于产生通用的、可批量重用的物联网部件和技术。（有利于克服“昆虫问题”）
- 分散式结构。海计算物联网更强调分散式（decentralized）结构，较易消除单一控制点、单一瓶颈和单一故障点，扩展更加灵活

海计算引出的计算机系统问题和创新机会

- 海计算系统的SPEC或Linpack是什么？
- 海计算系统的处理器体系结构
  - “计算机家族”概念需要回顾吗？
  - 还需要支持虚拟存储、硬件中断吗？
- 海计算系统的软件栈
  - 是否需要支持big-Bit?
  - 还需要OS吗？
  - OS是否用tinyOS或嵌入式OS就够了？
  - 系统软件复杂度能否降低1000倍？

请批评指正！