多核处理器动态和静态功耗建模方法论

Bhavishya Goel 和 Sally A. McKee

瑞典哥德堡查尔姆斯理工大学
{goelb, mckee}@chalmers.se

***摘要--系统*设计人员和应用程序编程人员必须考虑性能和能耗之间的权衡。在设计应用程序或运行时系统时，要做出能耗感知决策，就必须掌握不同处理器组件的能耗定量信息。我们提出了一种对单个内核和非内核组件的静态和动态功耗进行建模的方法，并在英特尔® Haswell 平台上对不同电压频率对的顺序和并行基准测试验证了我们的功耗模型。**

我们的功率模型为节能扩展提供了以下启示。(1) 我们发现非内核能量占总能量的 74%。特别是非内核静态能耗可高达总能耗的 61%，有可能成为能效低下的主要原因。(2) 我们发现，一个应用程序消耗最低能量的频率取决于它对内存的绑定程度。(3) 我们证明，即使使用更多的内核可以提高性能，但程序串行部分停滞的内核所消耗的能量可以使使用更少的内核更节能。

***关键词--功率*建模；静态和动态功率；能量表征。**

I. 导言

从千万亿次超级计算机到手持设备，现代系统必须在性能和功耗之间取得平衡。这通常要求系统能够获取实时功耗信息。虚拟机管理程序、操作系统和运行时软件都可以利用这些信息更高效地执行工作负载。

软件可从实际电能表或通过估算模型获取用电信息

硬件或软件。例如，英特尔至强 R⁻ 芯片可以使用节点管理器 R⁻ [1]来测量功率，但这种离模基础架构仅提供有关处理器、主内存和系统整体的信息。它无法报告单个芯片组件，即内核、功能单元、其他微架构组件或缓存。

基于模型的功耗估算是实际测量的常见替代方法。例如，英特尔的平均运行功耗限制（RAPL）[2]、AMD 的应用功耗管理（APM）[3]和英伟达的管理库（NVML）[4]都在硬件中实现了能耗估算模型。这些接口提供整个芯片的综合功耗值，但不提供单个计算单元的功耗值。RAPL 提供了内核与非内核的单独总值，但不提供单个内核的功率明细。非核心 "是英特尔公司对 CPU 组件的术语，这些组件位于核心之外，但与核心密切相关（如末级高速缓存、内存控制器和互连）、

实施能量感知调度和软件控制 DVFS 等技术需要更精细的功耗信息。鉴于功率计在商品硬件中的作用有限，人们提出了许多软件功率估算模型。这些模型致力于提供详细的功耗估算，粒度范围从整个 CPU [5]-[7]到单个内核 [8]-[10]、单个微架构组件 [11]-[13]，或单个内核加非内核 [14]。据我们所知，这些模型都没有明确区分静态功耗和动态功耗。

Mair 等人[15]强调了区分静态功率和动态功率的重要性，因为它们的影响和参数有着本质区别。两者都取决于电源电压，但静态功率取决于温度，动态功率取决于频率。这些功率成分从一个电压-频率阶跃到另一个电压-频率阶跃的缩放是不同的，因此在应用 DVFS 时，关于它们之间的细分的准确信息对于了解能量消耗趋势至关重要。

对静态功耗和动态功耗分别建模的另一个原因是为了描述系统能效。静态功耗是晶体管中的漏电流造成的。因此，它无助于执行

有用功。因此，静态能量代表了 "浪费的

系统中的 "能量"。静态能量与系统总能量之比表示系统的能源效率低下，有助于对建筑进行定性比较。

我们提出了一种系统方法，用于推导计算非内核和内核静态和动态功耗的模型。我们展示了如何隔离和量 化 不同处理器组件的功耗。我们在单线程和并行基准的四个不同电压频率阶跃下验证了我们的功耗模型，结果表明我们的模型在所有基准中都能准确估计功耗（平均绝对误差为 3.14%）。

我们使用我们的模型来描述在 Haswell 处理器上扩展时钟频率和线程级并行性的能效。我们发现，非内核静态能耗占系统总能耗的很大一部分（高达 61%），是能效低下的主要原因。我们还表明，以较低的频率运行应用程序并不一定能减少能耗：在选择高能效的 DVFS 策略时，必须考虑应用程序受内存约束的程度。最后，我们证明了并行工作负载的串行部分决定了耗能最少的并发水平。

II. 方法

我们在英特尔酷睿™ i7 4770（Haswell）处理器上运行实验，该处理器有四个物理内核，最高主频为 3.4 GHz。每个内核都有两个八向 32 KB 私有 L1 高速缓存（I 和 D 相分离）、一个 256 KB 私有 L2 高速缓存（I 和 D 相结合）和一个 8 MB 共享 L3 高速缓存，板载物理内存为 16 GB。

由于我们希望将重点放在芯片的 CPU 部分，因此我们的实验没有使用片上 GPU 和 eDRAM（它们是功率门控的，不会影响实验结果）。这些组件的功耗模型将非常有趣和有用，但推导这些模型超出了本文的工作范围。我们在所有实验中都禁用了超线程和 Turbo Boost。在其余讨论中，我们
芯片和 CPU 可以互换使用。

我们测量 ATX CPU 电源的功耗。
我们在以前的工作[10]中建立了一个基础架构。该基础架构使用电流传感器将测量到的电流转换为电压，然后由数据采集单元以高精度和高采样率记录下来。为了尽量减少干扰，我们在被测机器之外的其他机器上记录功率测量值。

我们使用 `libpfm4` 库提供的 `pfmon` 工具收集性能计数器值。我们使用 Linux® 内核驱动程序 `coretemp`，通过
芯片上的数字温度传感器 (DTS) [16]。

我们的模型将总功率分为四个部分：

1) 非核心动力、
2) 核心动力、

$$P(uncore)_{idle\_dynamic} = \alpha * C * V^2 * F \qquad (1)$$

其中 α = 非核心闲置活动系数

C = 非铁芯电容

V = 无芯电源电压

F = 非核心频率

空闲时，非内核有效电容（公式 1 中的αC）在频率变化时几乎保持不变（我们不考虑操作系统内务线程引起的微不足道的非内核活动）。回想一下，非内核静态功率取决于非内核电压和封装温度。我们通过 BIOS 固定无核电压，并测量相同封装温度下不同无核频率的空闲芯片功率。因此，我们可以有把握地假设静态功率在频率变化时保持不变。因此，测量功率的任何差异必须是由于非核动态功率的变化造成的。我们测量频率为 F_1 和 F_2 时的 CPU 空闲功耗，同时确保两次读数的封装温度相同。然后，非内核频率 F_n 下的 CPU 空闲功耗可计算为
$P_n = \alpha * C * V^2 * F_n + P_{static}$。然后，我们利用公式 2 计算空闲非内核*的* αC。

$$\alpha * C = \frac{P_2 - P_1}{V^2 F_2 - V^2 F_1} \qquad (2)$$

3) 非核心静态功率，以及
4) 核心静态功率。

我们努力为每个组成部分单独开发模型，以防止任何组成部分的模型估计误差影响其他组成部分的估计精度。

A. 非核心动力模型

我们将非内核动态功耗分为非内核空闲动态功耗和非内核活动动态功耗。我们的实验表明，非内核在空闲时既没有时钟门控，也没有功率门控。公式 1 给出了非内核空闲动态功耗的计算公式。

公式 3 显示了我们生成的非内核空闲动态功耗模型。我们通过重复测量 F_1 和 F_2 的不同值来验证 αC 的值。

$$P\,(uncore)_{idle\_\,dynamic} = 1.41 * 10^{-9} * V^2 * F \quad (3)$$

其中 V = 无芯电源电压

F = 非核心频率

接下来，我们分析二级缓存未命中对非内核动态功耗的影响。L2 缺失会导致环形互连和 L3 末级缓存中的活动。为了测量二级缓存未命中对 CPU 功耗的影响，我们创建了一个微基准，将内存访问与足够多的整数和浮点运算交错进行，以隐藏三级延迟。我们在逐步增加基准的工作集大小时测量 CPU 功耗，使内核活动（每个周期执行的微操作）和 L2 活动（每个周期的 L2 请求）保持不变，但 L2 未命中与 L2 请求的比例增加。我们将消耗功率的差异归因于 L2 缺失增加导致的非核心活动增加。我们对 L2 未命中率和功耗增加进行线性回归，以生成 L2 未命中对非内核动态功耗的贡献模型。公式 4 表明

图 1 显示了该模型与测量值的精确度。衡量估计模型拟合度的 R^2 系数为 0.999。

$$P\,(uncore)_{L2\_miss\_\,dynamic} = (3.043 * 10^{-9} * x^2 + 2.881 * 10^{-9} * x) * V^2 * F \quad (4)$$

其中，x = 每个非核心周期的全芯片 L2 未命中次数

V = 无芯电源电压

F = 非核心频率

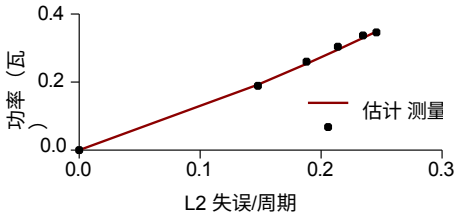


图 1：F=800 MHz 和 V=0.7V 时 L2 缺失导致的功耗模型适配性

我们进行了类似的实验，测量 L3 未命中对非内核功耗的影响。对于实际的 L3 未命中率（≤ 50 MPKI），CPU 功耗的增加可以忽略不计，甚至
尽管 DRAM 功耗会大幅增加。因此，我们的模型省略了 L3 缺失，将非核心空闲动态功耗与 L2 缺失导致的非核心功耗之和计算为非核心动态总功耗。

B. 核心动力模型

为了建立内核动态功耗模型，我们研究了微架构事件的影响，如执行的微操作（uops）、L1 访问和 L2 访问。我们首先量化非内存操作对内核动态功耗的影响。我们倾向于分别分析整数和浮点运算，但 Haswell 微架构没有提供浮点运算的性能计数器。因此，我们对两种指令类型进行了平均。

我们创建了微基准，循环运行以下内容

指令，无论是单独使用还是结合使用：

- x87 浮点乘法
- x87 浮点加法

由于非内核电压、非内核频率和封装温度在各读数之间保持稳定（假设温度变化在极小的采样周期内微不足道），因此测量功率的差异一定是由于内核动态功耗的变化造成的。

现在可以使用公式 2 计算 αC。我们计算多个频率对的 αC 并取平均值，以减少估计误差。我们按照这种方法计算所有微基准的 αC。我们使用线性回归来寻找最佳拟合，以估算非内存指令对内核动态功耗的影响。公式 5 显示了推导出的模型， 图 2 显示了模型与测量值的拟合度。R^2 的拟合系数为 0.801。

$$P\,(core)_{non-mem\_\,dynamic} = (2.448 * 10^{-10} * x + 1.1809 * 10^{-9}) * V^2 * F \quad (5)$$

其中 x = 每个周期的非内存指令

V = 核心供电电压

F = 核心频率

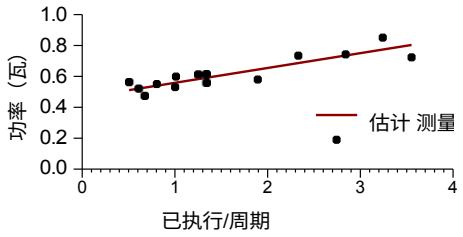


图 2：F=800 MHz 和 V=0.7V 时非内存执行单元功耗的模型适配度

接下来我们研究内存操作对内核动态功耗的影响。我们在非内存执行端口（端口 0、1、5 和 6）上保持恒定活动，并逐步
在负载/存储单元（端口 2、3、4 和 7）上增加内存。我们将内存操作的增加与内核动态功耗的增长联系起来，生成公式 6 中的模型。图 3 显示了模型拟合度。R^2 系数为 0.999。

$$P\,(核心)_{mem\_\,动态} = (2.780 * 10^{-10} * x + \quad (6)$$

- 整数乘法、
- 整数增加、
- SIMD 浮点乘法，以及
- SIMD 浮点加法

对于每个微基准测试，我们使用与计算空闲非内核功耗相同的技术计算 αC：我们从 BIOS 中固定非内核电压、非内核频率和内核电压。然后，我们在所有内核上启动微基准测试，测量功耗，将内核切换到更高频率，并在 10 毫秒内再次测量。

$$1.497 * 10^{-10}) * V^2 * F$$

其中 x = 每个周期的 L1 访问次数

V = 核心供电电压

F = 核心频率

为了研究 L2 访问对内核动态功耗的影响，我们采用了与研究 L3 访问影响相同的方法。在保持内核活动不变的情况下，我们再次使用 mi-crobenchmark 增加 L2 访问次数。随着 L2 访问率的增加，我们测量 CPU 功耗的增加。然后，我们对 L2 访问率和功耗的增加进行线性回归，以生成 CPU 功耗模型。

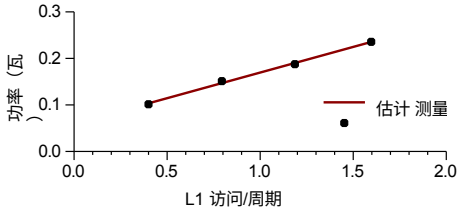


图 3：F=800 MHz 和 V=0.7V 时 L1 访问功耗的模型适配性

L2- 存取对内核动态功耗的贡献如公式 7 所示。图 4 显示了模型与测量值的拟合度。衡量拟合度的 R^2 系数为 0.997。

$$P(core)_{L2\_dynamic} = 2.829 * 10^{-9} * x * V^2 * F \quad (7)$$

其中 x = 每个非内核周期的每个内核 L2 访问次数

V = 无芯电源电压

F = 非核心频率

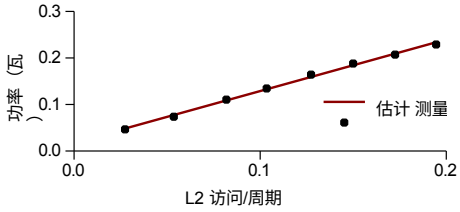


图 4：F=800 MHz 和 V=0.7V 时 L2 访问功耗的模型适配性

现在，内核的总动态功耗可 用 公式 8 表示。

$$P(core)_{dynamic} = P(core)\text{mem\_dynamic} + P(core)_{non\text{-}mem\_dynamic} + P(core)_{L2\_dynamic} \quad (8)$$

C. 非核心静态功率模型

当内核处于闲置状态时，Haswell 微架构会使用深度睡眠 C 状态（C7）。在 C7 状态下，内核处于电源门控状态，因此功耗可以忽略不计。与此相反，我们的实验表明，Haswell 非核心在芯片空闲时既没有电源门控，也没有时钟门控。因此，芯片的空闲功耗几乎完全由非内核造成。芯片空闲功耗可用公式 9 表示。

$$P(CPU)_{idle} = P(uncore)_{idle\_\text{dynamic}} + P_{(非核心)_{静态}} \quad (9)$$

我们使用第 II-A 节中得出的 $P(uncore)_{idle_动态}$模型来分离 $P(uncore)_{静态}$。回顾一下，静态功耗取决于电源电压和温度。我们使用 CPU 封装温度来估算整个非内核

为了验证非核电压和非核温度对非核静态功耗的影响，我们将非核电压设置为 0.7V 至 1.0V，增量为 0.05V。在每个电压下，我们使用热风枪改变 CPU 温度，并以每秒一个样本的粒度测量功耗。我们从测量值中减去非内核空闲动态功耗，然后对其进行非线性回归。

差值、电压和温度来创建公式 10 中的非核心静态功率模型。该公式使用了普尔-弗伦克尔效应 [17]。图 5 显示了三个电压点的测量值与估计非核静态值。我们运行了每个实验，直到温度停止下降（为便于比较，图中显示的时间窗口为 200 秒--在未显示的时间内，运行时间更长的实验继续显示出较高的估计精度）。在所有样本点上，估计模型的 R^2 系数为 0.999。

$$P_{(非核心无电)} = 141.615 * e^{-\left(\frac{-169.093+150.702*\sqrt{V}}{T}\right)} * V^2 \quad (10)$$

其中 V = 无芯电源电压

T = 包装温度

D. 核心静态功率模型

为了生成内核静态功耗模型，我们强制内核保持 C0 状态。在此状态下，Linux 内核运行

一个消耗恒定动态功耗的轮询空闲循环。我们采用计算非内核空闲动态功耗的相同策略计算该功耗。我们使用公式 2 计算 C0 状态下空闲内核的 αC。公式 11 显示了 C0 状态下每个内核的动态功率模型。公式 12 利用公式 11 分离出每个内核的静态功耗。

$$P_{(核心)_{C0\_dynamic}} = 1.28 * 10^{-9} * v^2 * f \quad (11)$$

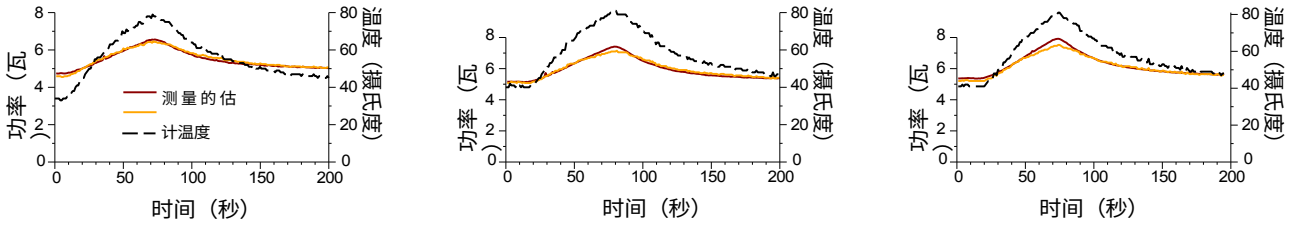
的平均温度。测量影响

其中 V = 核心供电电压

F = 核心频率

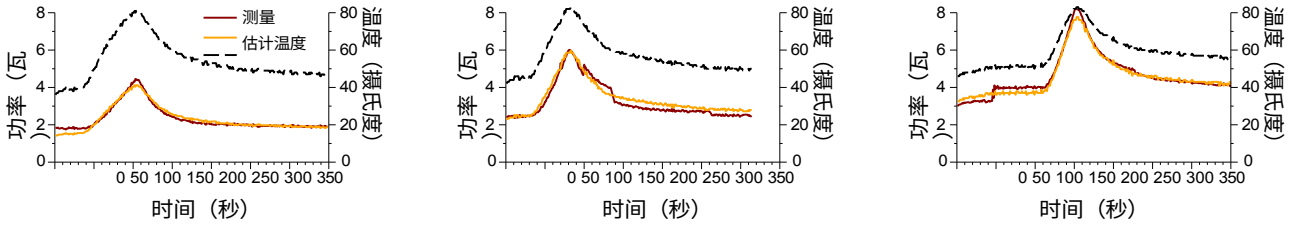
$$P(core)_{static} = \frac{1}{4} * (P(CPU)_{C0} - P(uncore)_{idle\_dynamic} - P(uncore)_{static} - 4 * P(core))_{C0\_dynamic} \quad (12)$$

要创建内核静态功耗模型，我们首先要测量内核电压和温度的影响。我们使用封装温度作为内核温度的近似值。我们将 BIOS 中的缓存电压固定为 0.7V，频率固定为 800MHz。我们再次以 0.05V 为单位将核心电压从 0.7V 逐步提高到 1.05V。在每个电压下，我们使用热风枪改变封装温度，并测量功耗和温度的变化。我们对收集到的数据进行非线性回归，以创建公式 13 中的内核静态模型。图 6 显示了测量值与



(a) V=0.70　　(b) V=0.85　　(c) V=1.0

图 5：非核心静态模型拟合度



(a) V=0.70　　(b) V=0.85　　(c) V=1.0

图 6：核心静态模型健壮性

三个电压点的估计磁芯静态值。估计模型与测量值的 R^2 系数为 0.996。

$$P_{(核\ 天电心)} = 1525.07 * e^{-(\frac{1884.1-2523.5T}{V})} * V^2 \quad (13)$$

其中 V = 核心供电电压

T = 包装温度

E.芯片总功率模型

根据我们的研究结果，我们在构建全芯片功耗模型时必须考虑另外两个因素。首先，我们的测量结果表明，当内核频率增加时，芯片功耗在某些频率阶段的增加幅度超出预期。这些增加似乎取 决 于 活动内核的数量，而与内核活动无关。我们认为，这种现象是由于内核 PLL 在特定频率以上切换到更高的电源电压所致。其次，芯片功耗在较高温度下会突然增加--每个活动内核最多增加 0.5W，但我们的实验和研究一直无法确定这一现象的根源。如果没有更多关于其原因的信息，这两个我们的实验表明，$P\ (misc)$ 可占芯片总功耗的 10%，具体取决于内核和非内核活动的水平。因此，芯片总功耗为

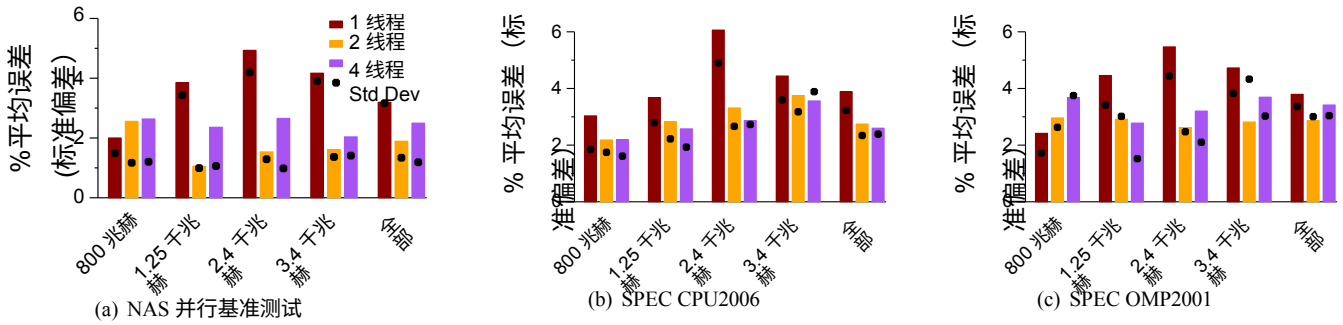
$$P\ (CPU) = P\ (非核心)_{动态} + P\ (非核心)_{静态}$$

功耗部分很难建模。因此，我们构建了一个离线表格，列出了根据经验确定的因这些因素而增加的功耗，并承认需 要 更深入的了解才能准确预测它们的行为。我们将这两个部分统称为 $P\ (misc)$。

$$+P(\text{核心})_{\text{动态}}+P(\text{核心})_{\text{静态}}+P(\text{杂项}) \quad (14)$$

III. 验证

我们在表 I 中的单线程 SPEC CPU2006 [18] 基准、NAS Parallel Bench- marks [19] 和多线程 SPEC OMP2001 [20] 应用上验证了我们的功耗模型。我们省略了 SPEC CPU2006 中的 *bwaves* 以及 SPEC OMP2001 中的 *art* 和 *fma3d*，因为它们不在我们的系统上运行。我们使用 NAS B 类输入，但忽略了 *IS*，因为其运行时间不够长，无法收集可靠的测量结果。我们使用一个、两个和四个线程运行并行应用程序。同样，我们运行一个、两个和四个并发实例的顺序应用程序。我们每秒读取一次封装温度和内核电压，并收集性能计数器，如第二节所述。我们每 1 毫秒测量一次处理器功耗，并在一秒钟内取平均值，以便将功耗值与其他参数同步。为了验证我们的模型在不同电压-频率阶跃时对功耗的估算是否准确，我们在四个 DVFS 点进行了验证：800 MHz（0.7V）、1500 MHz（0.78V）、2400 MHz（0.88V）和
3400 MHz，电压为 1.01V。我们再次禁用超线程和 Turbo Boost。



(a) NAS 并行基准测试　(b) SPEC CPU2006　(c) SPEC OMP2001

图 7：验证芯片总功率

套房	基准
NAS	SP, EP, BT, MG, DC, UA, CG
SPEC OMP2001	quake、swim、wupwise、ammp、API、APIU、MGrid
SPEC CPU2006	cactusADM, calculix, dealII, gamess、GemsFDTD, gromacs, lbm, leslie3d, milc, namd, povray, soplex, zeusmp, astar, bzip2, gcc, gobmk, h264ref, hmmer, libquantum, mcf, omnetpp, perlbench, sjeng, xalancbmk

表 I：用于验证的基准

图 7 显示了基准套件的估计误差。所有 NAS 基准在四个 DVFS 点上的平均绝对误差（MAE）为：单线程 3.19%，双线程 1.89%， 四线程 2.50%。SPEC CPU2006 的 MAE 为：单实例运行 3.88%，双实例运行 2.74%，四实例运行 2.60%。SPEC OMP2001 的单线程平均绝对误差为 3.79%，双线程为 2.87%，四线程为 3.42%。所有基准和电压频率状态下所有样本点的平均绝对误差为 3.14%，标准偏差为 2.87%。

对于大多数基准，我们的模型都能准确预测功率估计值。对于相位变化较快的基准（NAS 中的 *DC* 和 SPEC OMP2001 中的 *ammp*），我们必须以 100 毫秒的较高测量粒度重新运行验证实验，以准确估计其快速变化的功耗值。

我们知道，模型错误至少来自两个方面。首先，Haswell 微架构没有提供跟踪浮点运算执行情况的每核事件，这使我们无法为浮点和整数指令的执行创建单独的模型。这导致计算密集型阶段的模型略有误差。其次，我们为 P（*misc*）创建的离线表中的值并 不总是准确的，尤其是在较高频率的快速阶段变化期间。尽管存在这些已知的误差来源，我们的模型仍能准确预测频率和线程数变化时的工作负载能耗趋势。

IV. 能量表征

动态电压频率扩展允许硬件或软件调整时钟速度和/或电压水平，以尝试提高性能或节省功耗。还可以增加工作负载的并行程度（即并行线程数），以提高性能，但这种扩展往往需要付出能耗代价。找到既能满足应用性能目标，又能保持给定能耗预算的线程频率或数量，并不一定是一件简单的事情。通过展示内核和非内核组件的静态和动态能量如何随 DVFS 和线程级并行性的扩展而扩展，我们利用我们的功耗模型来描述能效。

我们对不同电压频率点和线程并发水平下的能耗进行了敏感性分析。在这些研究中，我们采用了参数化的合成工作负载，使我们能够改变内存访问强度和线程级并行性。我们将这种合成工作负载的结果与实际应用的结果进行了验证。

A. DVFS 的能量效应

我们首先研究电压-频率扩展如何影响能耗。传统观点认为，以更高的频率运行可提高性能，但代价是消耗更多的能量[14]，因为动态功耗与电压成二次方关系（公式 1）。但当我们考虑到非内核消耗能量的影响时，就会发现以较低的电压-频率阶跃运行有时会消耗更多能量。图 8 显示了当我们改变内存强度和执行频率时，合成工作负载的能耗变化情况。能耗数据以最低频率（800 MHz）下的能耗为标准。

图 8(a) 显示，在最低频率下运行的完全顺序、CPU 绑定的应用程序（因此随频率线性扩展）消耗的总能量最高，尽管在此频率下内核动态能量最低。出现这种情况的原因是非内核能量占总能量的 74%，但它对总能量没有任何贡献。

对应用性能的影响。具体来说，非内核静态能量占总能量的 61%。当频率增加时，内核动态能量就会增加。然而，缩短执行时间可大幅减少非内核静态能量，从而降低整体能量。频率达到一定程度后，（内核和非内核）动态能量的增加使非内核静态能量的减少相形见绌，从而导致总能量消耗再次上升。

应用程序的内存占用越多，从高频率运行中获得的性能就越低，因此非核心静态能量的减少随着频率的增加而变得不那么重要。当单线程应用程序的内存约束部分超过 40% 时，我们就会看到预期的趋势：频率的提高会增加能量消耗。图 8(b) 显示了四线程工作负载的能量缩放结果。由于内核功率与芯片总功率的比率增加，多线程、CPU 绑定应用的非内核静态能耗并不明显（我们测得两个线程时为 49%，四个线程时为 35%）。因此，当双线程应用的内存束缚率为 30%，四线程应用的内存束缚率为 20%（而单线程应用的内存束缚率为 40%）时，能量缩放与频率缩放相关。这一分析表明，在选择运行应用程序的频率时， 必须考虑到应用程序受内存约束的程度，从而使其消耗最少的能量。

B. 增加线程数对能量的影响

在能效理想的处理器上，无论使用多少个内核，执行一定量的工作所消耗的能量应该大致相同。但是，现代处理器能效低下的原因导致其能耗趋势与这种 "完美 " 情况不同。我们使用功耗模型来分析并发级别的扩展如何影响总能耗。图 9 显示了在保持工作量不变的情况下，当我们改变串行代码的比例时，不同并行度下的能耗缩放情况。能量值已归一化为单线程运行的能量消耗。当工作负载完全并行时，使用所有可用内核消耗的能量最少，因为性能的 提高会减少非内核静态能量和非内核空闲动态能量。使用更多内核时，由于封装温度升高，内核静态能量略有增加，但内核动态能量保持不变。在 3.4 GHz 下，从一个线程到两个线程的能耗降低了 26%，从一个线程到四个线程的能耗降低了 40%。

随着串行执行相对比例的增加，即使工作量没有变化，内核动能也会增加。即使处理器执行的指令数量相同，但由于内核间通信增加导致流水线停滞，内核总活动时间也会增加。这增加了内核闲置的动态能量。在 3.4 GHz 时（图 9(a)），当应用的串行部分执行完后，内核闲置动态能耗会增加。

超过 20%时，使用两个内核比使用四个内核更高效。当串行应用超过 40% 时，运行单线程比运行并行线程更节能，尽管从一个内核到四个内核的速度提升接近 100%。

800 MHz（图 9(b)）时，非内核静态能耗与总能耗的比率远高于 3.4 GHz，因此增加线程数会使降低非内核静态能耗带来的节能效果更加突出。在 800 MHz 频率下，即使 30% 的应用是串行应用（而在 3.4 GHz 频率下为 20%），运行四个线程仍然是最节能的。工作负载必须达到近 70% 的串行比例，串行版本才会比多线程更节能（3.4 GHz 时为 40%）。

C. 基准能量扩展

为了验证利用模型敏感性分析预测的能量趋势，我们检查了六个 NAS 基准的能量消耗。图 10 显示了 NAS 基准在不同频率和不同并发水平下的能量消耗和执行时间比例。请注意，这些都是测量值。

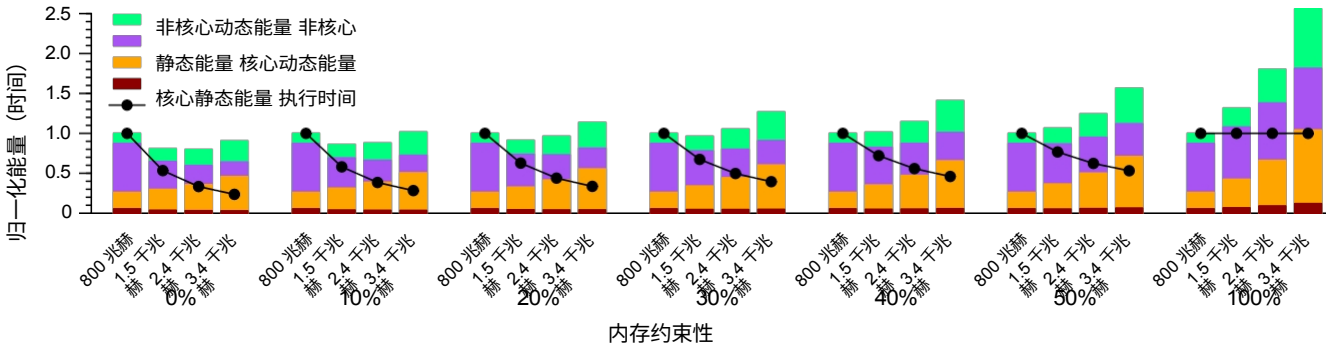
EP，即 "令人尴尬的并行"（Embarrassingly Parallel）基准，受 CPU 约束：它的工作集很小，没有串行部分。图 10（a）显示，正如我们从图 8（a）中得出的趋势一样，顺序运行在最低频率下消耗的能量最大。对于四线程运行，*EP 在* 3.4 GHz 频率下能耗最高，在 1.5 GHz 频率下能耗最低，这与图 8（b）中的趋势一致。当扩展到更多线程时，能量消耗会减少，这验证了我们的分析。

BT 的工作集比 *EP* 大。对于顺序执行，图 10(b)中的能耗缩放趋势与图 8(a)中 10%内存绑定工作负载的能耗缩放趋势相似。在四线程情况下，能量消耗与图 8（b）中 10% 内存绑定工作负载的趋势非常接近。其余基准（*CG*、*SP*、*MG* 和 *LU*）的工作集甚至更大，它们的能耗缩放趋势与图 8 中 20% 内存绑定工作负载的能耗缩放趋势一致。

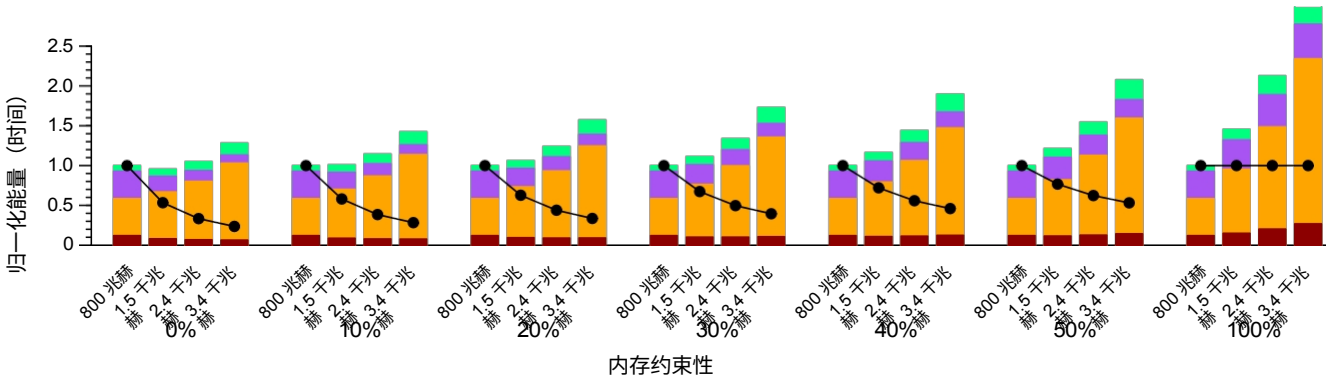
在线程扩展方面，大多数 NAS 基准的执行时间几乎呈线性扩展，能耗随着并发量的增加而降低。运行频率为 3.4 GHz 的 *SP* 是个例外。从一个线程扩展到两个线程几乎是线性加速（执行时间减少 48%），但从两个线程扩展到四个线程，由于流水线停滞，执行时间仅减少 36%。正如我们的模型所预测的那样，内核动态能量增加，使得 *SP* 在四个线程时比两个线程时消耗更多的总能量。

V. 相关工作

由于难以获得准确、实时的功耗测量结果，再加上对绿色计算的日益重视，引发了对功耗估计模型的大量研究。Tivari 等人 的早期研究[21]开发了一种基于测量的方法，用于建立指令级功耗模型。

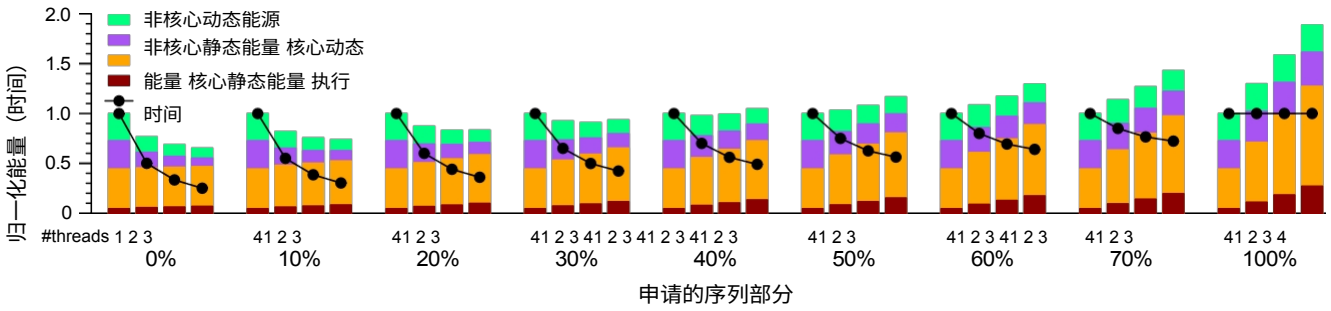


(a) 1 线程

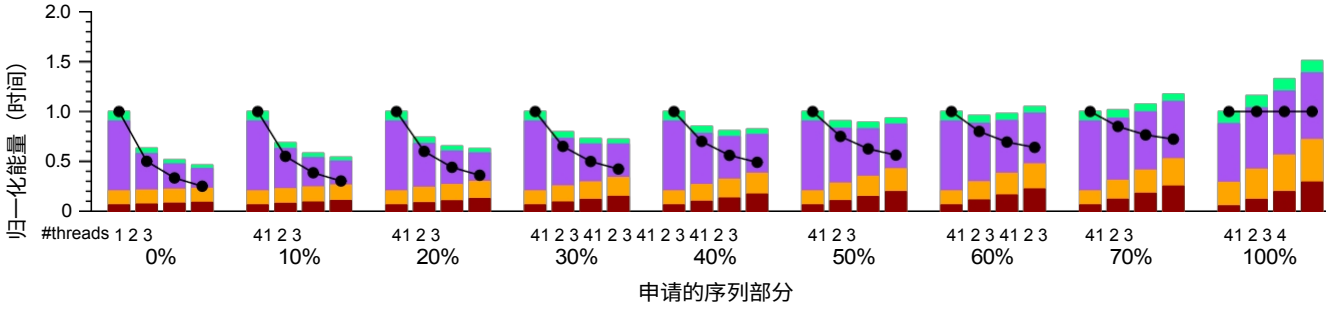


(b) 4 线程

图 8：DVFS 对总能耗和性能的影响

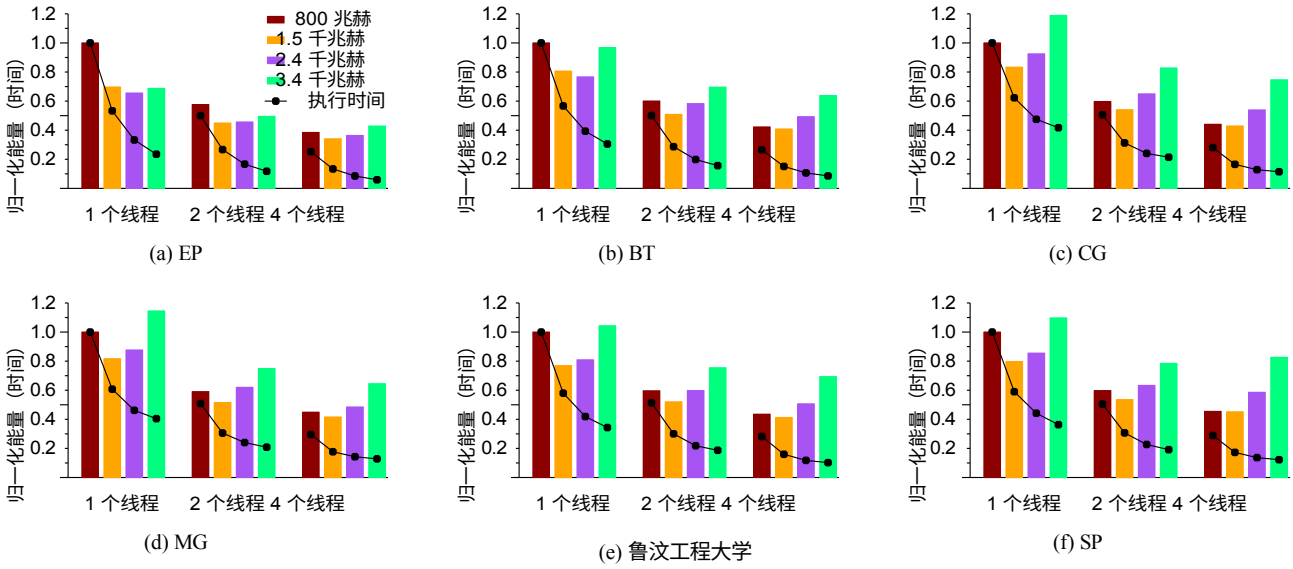


(a) CPU 频率=3.4 千兆赫



(b) CPU 频率=800 MHz

图 9：线程缩放对总能量的影响



(a) EP (b) BT (c) CG

(d) MG (e) 鲁汶工程大学 (f) SP

图 10：NAS 基准的能量扩展

功率效应。他们将每种指令、指令对和指令间效应（如流水线停滞和高速缓存缺失）的能耗成本联系起来。与我们不同的是，他们没有区分静态和动态功耗成分。他们的方法要求在每个处理器频率下分别建立模型。

最近的研究越来越多地依赖硬件性能监测计数器（PMC）来建立处理器功耗模型。Joseph 和 Martonosi [11]通过选择与组件利用率直观相关的性能计数器，并推导出每个组件活动因子的权重，建立了单个微架构组件 的功耗模型。他们在单一频率下进行实验（因为 DVFS 在 2001 年还不常见），并没有区分静态功耗和动态功耗。Singh 等人[8]通过对四个 PMC（代表整数运算、浮点运算、内存访问和流水线停滞）进 行 多元线性回归，估算出 AMD Phenom [22] 的每核功耗。辛格[23] 后来又加入了温度和频率缩放功能。Goel 等人[9]进一步完善并验证了这一通用方法，在多个体系结构中始终保持较高的模型精度。Goel 等人[10] 研究了不同的功率测量基础设施对模型精度。

与我们一样，Bertran 等人[13]也开发了一种将动态功率和静态功率相加的功率模型。不过，他们将静态功率定义为空闲功率，并使用对所选 PMC 值的回归来得出静态（空闲）和动态功率。他们的功率模型不考虑电压和温度。Spiliopoulos 等人 [24] 在功率估算方法中将静态功率和动态功率分开处理，但他们也将静态功率定义为空闲功率。他们创建了一个表格

(在所有频率阶数和 "典型核心温度范围 "下的闲置功率值。对于其模型中的动态功耗部分，他们仅根据退役指令来估算有效的处理器活动因子。我们的经验表明，忽略内存操作的模型在内存绑定的工作负载上很容易出错。

Su 等人[14]也将空闲功耗和动态功耗分开处理。与我们一样，他们也使用电压和温度来开发静态功耗模型，但他们将空闲动态功耗归入静态功耗模型，从而无法反映内核和北桥（相当于我们模型中的非内核）的真实静态功耗。他们在九个 PMC 上使用多元线性回归来推导动态功耗模型，但依靠如此多的计数器需要时间复用，从而牺牲了精度。他们
得出的结论是，他们的 AMD FX-8320 R 芯片总是在最低电压频率状态下消耗最少的能量，而我们的
哈斯韦尔的研究结果与此相悖。

VI. 结论

准确估算处理器组件的功耗对于支持更好的功耗感知资源管理非常重要。我们提出了一种估算内核和非内核处理器组件静态和动态功耗的方法。该方法使用内核和非内核电压、封装温度和性能计数器创建模型，可估算所有系统频率下顺序和并行应用的功耗。我们在 NAS、SPEC CPU2006 和 SPEC OMP2001 的基准测试中验证了我们的模型，结果表明，我们的模型可以在所有电压频率对和不同并发水平上高精度地估算功耗（平均绝对误差为 3.14%）。

我们利用功率模型研究了 DVFS 对能耗的影响，结果表明，与传统观念相反，以最低频率运行应用程序并不总是最节能的。必须考虑非核心静态能量效应。应用程序的最低能耗频率取决于其 内存约束程度和并发线程数量。我们研究了线程扩展对能耗的影响，结果表明程序的相对串行部分会影响程序能耗最低的并发水平。

致谢

这项工作的部分资金来自欧盟 FP7 项目 EUROSERVER，资助协议号为 610456。

参考资料

[1] D.Jenkins, "Node Manager - a dynamic approach to managing power in the data center," White Paper, Intel Corporation. http://communities.intel.com/docs/DOC-4766, Jan. 2010.

[2] E.Rotem, A. Naveh, D. Rajwan, A. Ananthadrishnan, and E.Weissmann，"第 2.0 版的电源管理架构—代英特尔®酷睿™ 微体系结构，原代码为命名为 Sandy Bridge，"*第 23 届高性能芯片 HotChips 研讨会论文*，2011 年 8 月。

[3] Advanced Micro Devices，"AMD Opteron™ 6200 系列处理器 Linux 调整指南"，2012 年，http://developer.amd.com/wordpress/media/2012/10/51803A_OpteronLinuxTuningGuide/_SCREEN.pdf。

[4] *Nvidia NVML API 参考手册》*，英伟达公司，2012 年。

[5] K.Rajamani, H. Hanson, J. Rubio, S. Ghiasi, and F. Rawson, "Application-aware power management," in *Proc. IEEE International Symposium on Performance Analysis of Systems and Software,* October 2006, pp.

[6] G. Contreras 和 M. Martonosi，"利用性能监控单元事件预测英特尔 XScale 处理器的功耗"，《*IEEE/ACM 低功耗电子与设计国际研讨会论文集》*，2005 年 8 月，第 221-226 页。

[7] D.Economou, S. Rivoire, C. Kozyrakis, and P. Ranganathan, "Full-system power analysis and modeling for server envonments," in *Proc.建模、基准测试和仿真研讨会*，2006 年 6 月。

[8] K.Singh, M. Bhadauria, and S. McKee, "Real time power estimation and thread scheduling via performance counters," in *Proc.芯片多处理器设计、架构和仿真研讨会*，2008 年 11 月。

[9] B.Goel, S. McKee, R. Gioiosa, K. Singh, M. Bhadauria, and M.Cesati, "Portable, scalable, per-core power estimation for intelligent resource management," in *Proc. 1st International Green Computing Conference,* Aug. 2010, pp.

[11] R.Joseph 和 M. Martonosi，"高性能微处理器的运行时功耗估计"，《*IEEE/ACM 低功耗电子与设计国际研讨会论文集》*，2001 年 8 月，第 135-140 页。

[12] F.Bellosa, S. Kellner, M. Waitz, and A. Weissel, "Event-driven energy accounting for dynamic thermal management," in *Proc.低功耗编译器和操作系统研讨会*，2003 年 9 月。

[13] R.Bertran, M. Gonzalez, X. Martorell, N. Navarro, and E.*Ayguade*："使用性能计数器的多核处理器可分解和响应式功耗模型"，《*第 24 届 ACM 超级计算国际会议论文集》*，2010 年 6 月，第 147-158 页。

[14] B.Su, J. Gu, L. Shen, W. Huang, J. Greathouse, and Z. Wang, "PPEP: Online performance, power, and energy prediction framework and DVFS space exploration," in *Proc. IEEE/ACM 47th Annual International Symposium on Microarchitecture,* Dec. 2014, pp.

[15] J.Mair, D. Eyers, Z. Huang, and H. Zhang, "Myths in power estimation with Performance Monitoring Counters," *Elsevier Sustainable Computing：信息学与系统*》，第 4 卷，第 2 期，第 83-93 页，2014 年 6 月。

[10] B.Goel, S. McKee, and M. Själander, *Techniques to Measure, Model, and Manage Power,* ser.Advances in Computers.Elsevier, Nov. 2012, vol. 87, ch. 2, pp.

[16] M.M. Berktold 和 T. Tian，"使用 DTS/PECI 监控 CPU"，英特尔公司白皮书 322683。 http://download.intel.com/design/intarch/papers/322683.pdf，2010 年 9 月。

[17] R.B. Hall, *Thin Solid Films*.Elsevier, Oct. 1971, vol. 8, ch. 2, pp.

[18] 标准性能评估公司，"SPEC CPU 基准套件"，http://www.specbench.org/osg/cpu2006/，2006 年。

[19] D.Bailey, T. Harris, W. Saphir, R. Van der Wijngaart, A. Woo, and M. Yarrow, "The NAS parallel benchmarks 2.0," NASA Ames Research Center, Report NAS-95-020, Dec. 1995.

[20] 标准性能评估公司，"SPEC OMP 基准套件"，http://www.specbench.org/hpg/omp2001/，2001 年。

[21] V.Tivari, S. Malik, A. Wolfe, and M.-C. Lee, "Instruction level power analysis and optimization of software," Kluwer Journal of VLSI Design, Vol.Lee，"指令级功耗分析和软件优化"，《*Kluwer VLSI 设计杂志*》，第 13 卷，第 3 期，第 223-238 页，1996 年。3, pp.

[22] AMD 公司，"型号和功能比较 - AMD Phenom.处理器"，http://www.amd.com/us-en/Processors/ProductInformation/0、,30_118_15331_15332\%5E15347,00.html，2007 年 12 月。

[23] K.辛格：《多核处理器上功率感知计算的预测策略》，康奈尔大学博士论文，2009 年 6 月。

[24] V.Spiliopoulos, S. Kaxiras, and G. Keramidas, "Green governors：*第二届国际绿色计算会议论文集*》，2011 年 7 月，第 1-8 页。