

RW: Multi-Processor

Coherence[✓] vs. Consistency[✓]
based on → board LAN Internet
SMP / cluster / Cloud

* Distributed System

* Multi-core

New Amdahl's law

Dennardian law (?)

Infrastructure 基础架构? / Utility?

1. ESI (MSI) → MESI?

后者有 "E": 是唯一具有缓存的人

"S": 多个 core 都具有缓存

原因: 若是唯一读者, 那么修改无需通知别人 (不在总线上发 sign)

可减少干扰与总线占用

2. Multi-Core

2007 new Amdahl's law

$$T_{old} / T_{new} = \frac{T_s + T_p}{T_s + T_p / S}$$

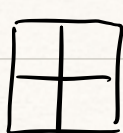
T_p : 可并行部分

S : core count

$T_{para} \sim 30\%$ 时. Speedup ≤ 1.41

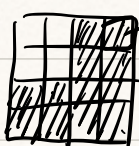
一个程序内可能可并行较少, 但可多个不相干程序 (目前想法)

1974 Dennardian law



4-core

$S=2$
(线性密度)



16-core

die

dark

$$P = c \cdot V^2 \cdot f \cdot N$$

功耗 \uparrow 容性 \downarrow 电压 \downarrow 频率
(各种电容等)

但单位面积散热面积有限

4 → 16 core 摩尔定律 18个月 $S=2$

Dark silicon problem?

Post-Dennardian

ΔN S^2 S^2

Δf S S (集成度高可提速)

ΔC $1/S$ $1/S$

ΔV^2 $1/S^2$ 1

ΔP 1 S^2 散热要 S^2 倍, 有限制

↑
前后比

另一种技术: chiplet?

3. Distributed System

History: H/W Infrastructure

硬件容易坏, 但 availability

Next } - time 时间同步问题

- data (spatial) consistency

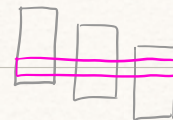
✓ 4. static pipelining 静态调度

Code Movement

4.1 unrolling Loop 循环展开

4.2 SW pipelining 软件流水

分阶段安排指令顺序



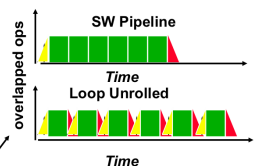
Software Pipelining Example

Before: Unrolled 3 times

```
1 L.D F0,0(R1)
2 ADD.D F4,F0,F2
3 S.D 0(R1),F4
4 L.D F6,-8(R1)
5 ADD.D F8,F6,F2
6 S.D -8(R1),F8
7 L.D F10,-16(R1)
8 ADD.D F12,F10,F2
9 S.D -16(R1),F12
10 DSUBUI R1,R1,#24
11 BNEZ R1,LOOP
```

After: Software Pipelined

```
1 S.D 0(R1),F4 ; Stores M[i]
2 ADD.D F4,F0,F2 ; Adds to M[i-1]
3 L.D F0,-16(R1) ; Loads M[i-2]
4 DSUBUI R1,R1,#8
5 BNEZ R1,LOOP
```



• Symbolic Loop Unrolling

- Maximize result-use distance
- Less code space than unrolling
- Fill & drain pipe only once per loop vs. once per each unrolled iteration in loop unrolling

5 cycles per iteration