Multicore Computing

Dr. A. Sahu
CSE, IIT Guwahati

Why Multicore?

• Saturation of single processor performance
  – Speed limit not to crosses : 4GHz
    – The ultimate point
    – Power consumption is proportional to square of frequency
      \[ P = \frac{1}{2} C \cdot V \cdot f^2 \]
  – Single-processor
    – Branch prediction accuracy gone upto 95%
    – L1 Cache hits gone upto 80%
    – ILP exploited by uniprocessor is upto 8 (mostly)
    – Thread/Data level parallelism needs to exploit

Why Multicore?

• Application specific IC (ASIC)
  – High performance, low power than Processor
  – But complexity of ASIC design is very High
  – Example: 12MP+HDVideo, GPS Camera in side mobile handset
  – It is fixed for an application

Terminology

• All are similar terminology
  – Multicore (Connected Internally inside a Chip)
  – Multiprocessor (Connected Via BUS/ On a Mother board)
  – Multicomputer (Connected Via LAN)
• Parallel Architecture

Power Vs frequency

Era of Parallelism

Work Required

Free Speedup

Evolutions of Total Platforms

Steady state main processors [MHz] vs

CPU beginning [MHz]
**Why Multicore?**

- VLSI technology offering high integration density
- Moore’s Law (In 1965, Gordon Moore Prediction)
  - Exponential growth of the number of transistors on an IC
  - Doubled every 26 months for the past three decades
- Why more transistors per IC? Smaller transistors, Larger dice

---

**Why Multicore?**

- Many applications are highly parallel
  - Take benefit of all parallelism (instruction, data and thread)
- Multiprocessors
  - Flexible, programmable, high performance
  - Take benefit of all parallelism (instruction, data and thread)
  - Likely to be cost/power effective solutions

---

**Why Multicore?**

- Multiprocessors are
  - Flexible, programmable, high performance
    - Processor are programmable as compared to ASIC
    - Flexible in terms of portability as compared to ASIC
    - Higher Performance than single processor

---

**Example of Multicore**

- CRADLE TECHNOLOGIES
  - CT3600
  - CT3600

---

**Multicore Difficulties**

- Multiprocessors are likely to be cost/power effective solutions
  - Because it share lots of resources
    - **Personal room is costlier than dormitory**
    - **You can’t allocate a Bungalow to each student: it will too costly**
      - Hostel room with shared facility is sufficient
  - Need not require very high frequency to run
  - Lots of replication makes easy to manage and cost effective in design
**Multicore Difficulties II**

- Many applications are highly parallel
  - Take benefit of all parallelism (instruction, data and thread)
  - Most of the coder write sequential code
  - Who will extract parallelism from applications?
  - There is no successful auto-parallelisation tool till date
    » Attempts: Cetus, SUIF, SolarisCC

---

**Multicore Difficulties III**

- Task scheduling in multiprocessors
  - Deterministic task scheduling on multiprocessor with more than 2 processor is NP-Complete problem
  - 4 Tasks \( \{A,B,C,D\}\)\} {}, \( \{A,B,C\}\)\} {}, ......Exponential Solutions

---

**Function Parallel Architectures**

- Instruction level PAs
- Thread level PAs
- Process level PAs
- Built using general purpose processors
- Pipelined processors
- VLIWs
- Superscalar processors
- Shared Memory MIMD
- Distributed Memory MIMD

---

**Grain size and performance**

- Overhead limited
- load imbalance and parallelism limited
- Speed up
- Fine grain
- Opt grain size
- Coarse grain

---

**Speed up and efficiency**

\[
T_p = \text{Time on uniprocessor} \\
T_p = \text{Time on } p \text{ processors} \\
\text{Speed up} = S_p = \frac{T_p}{T_p} \\
\text{Efficiency} = E_p = \frac{T_p}{p \cdot T_p}
\]

Usually, \( S_p < p \) or \( E_p < 1 \) due to overheads
Sometime superlinear speedup \( (S_p > p \) or \( E_p > 1) \) is reported
This may be due to
- failure to use the best uniprocessor algorithm
- advantage due to larger memory

---

**Amdhal’s law**

- Work 500, Time 500, Sp=1X
- Work 500, Time 400, Sp=1.25X
- Work 500, Time 350, Sp=1.4X
- Work 500, Time 300, Sp=1.7X
Amdahl’s Law

Serial fraction \( s = \frac{T_s}{T_i} \)

\[
T_p = T_s + \frac{T_i - T_s}{p}
\]

\[
S_r = \frac{T_i}{T_p} = \frac{T_i}{T_i + \frac{T_i - T_s}{p}} = \frac{T_i}{T_i(1 - \frac{1}{p}) + \frac{T_i}{p}}
\]

\[
= \frac{1}{s(1 - \frac{1}{p}) + \frac{1}{p}} = \frac{p}{s(p-1) + 1}
\]

\[
Lt \quad p \to \infty \\
S_r = \frac{1}{s}
\]

Amdahl’s Law: Assumption, Difficulty, Superlinerity

- Next Class
  - Critical section access in Parallel Region need to be serialized
  - Computer are becoming powerful to handle larger task...
    - Task multiplication instead of division
  - In presence of Memory hierarchy it behaves differently