Distributed Computer Systems

Issued by:
Dr. Ameer Mosa Thoeny Al-Sadi

lecture 2 (Parallelism)

Reference:
ADVANCED COMPUTER ARCHITECTURE AND PARALLEL PROCESSING

Outlines

- Some Elementary Concepts.
- Parallelism physical effect.
- Parallelism Reasons (Motivations).
- Challenges.
- Transparency in Distributed Systems.

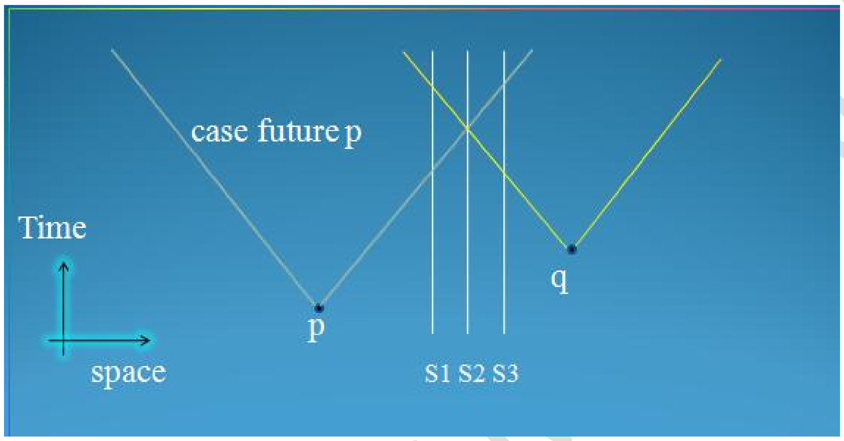
SOME ELEMENTARY CONCEPTS

- The Concept of Program: From the programmer's perspective, roughly a program is a well-defined set of instructions, written in some programming language, with defined sets of inputs and outputs. From the operating systems perspective, a program is an executable file stored in a secondary memory. Software may consist of a single program or a number of programs. However, a program does nothing unless its instructions are executed by the processor. Thus a program is a passive entity.

- The Concept of Process: Informally, a process is a program in execution, after the program has been loaded in the main memory. However, a process is more than just a program code. A process has its own address space, value of program counter, return addresses, temporary variables, file handles, security attributes, threads, etc.
- Process defined as a sequence of events (instructions, sending messages, receiving messages).

- The Concept of Thread: Thread is a sequential flow of control within a process. A process can contain one or more threads. Threads have their own program counter and register values, but they are sharing the memory space and other resources of the process. Each process starts with a single thread. During the execution other threads may be created as and when required.

Parallelism physical effect



- Event *p* and *q* are independent, can't have mutual effect because no one from them can happen in case future of another.
- Look in s1 see at first event *p* and then *q*.
- Look in s3 will be change order in opposite.
- Look in s2 can form that both events happen together.
- Note: have never information about which event occur before.

- For exact describe event in distributed system can't use full (linear) ordering (format), while for a lot of practical cases like this model is suitable.
- Not each two events show ordering in time.
- Not each distributed system give:

$$T(p) < T(q) \vee T(q) < T(p) \vee T(p) = T(q).$$

- *Models actually parallelism use partially ordering.*

Parallelism Reasons

- Physical restriction.
- Reliability, Accessibility.
  -HA (High Availability)
- Efficiency
  -HPC (High Performance Computing)
- Synchronization restriction.

Physical Restriction:

1. System, which are his spirit (matter)- “from nature” physical distribution.
2. Sequence system here can’t use.
3. Not interest about performance, neither about reliability, but about capability execute with multiple inputs together.
4. System reflects natural distribution inputs, which are data and control process.
5. For example, // system control breaks when haven’t response for his input immediately not until finish his assigned time.
6. Distributed database:
   -Haven’t reason “Bear” data on one place.
   -Branches of big organization.

   <u>Note:</u>
   Control system has wide technological processes:

   - Especially remote sensor and actuators.
   - Monitoring & control system from away (remote control)
   - Multi input data flow must keeping execute together.

Reliability:

1. Distribution system can be open additional redundant component for raising all dependency.
2. Reduction components in distributed system have less effect on all performance system than in system with unique resources.
3. Can increase quality service and efficiency, decrease response time but functionality remains.
4. For example, // dropout DNS service in network can't full access to files in servers.

Performance:

1. May be major reason examination and exploitation.
2. When one processor not provides adequate execution;
    - Time critical application (ex. Reactive system with strait time restriction, weather, finances).
    - Time-consuming; big data or long loop.
    - More rising demand application than efficiency machine.
3. When solution on sequence architecture takes loss long time.
4. When we want solve complex problems.
5. Efficiency processes long rang rising (but for when?).
6. In past was efficiency supercomputing comparing with latest processors.
7. Always will be enough problem overlaps capability (grand challenge).
8. Most problems are naturally parallel.
9. Highest execute is can obtain from usable multiple processors at same time (simultaneously).

Understand Performance Limitation and Today Technology Form Hardware perspective.

Moore's Law

- Gordon Moore:
  -count transistors on chip duplicate each 18 months (1965); duplicate each two years (1995).

  -" Moore's Law" get from direct conflicts with natural laws (1997).
- Simultaneously technology:
  -90nm,65nm,45nm,32nm.
  -In 2012, 22-nanometer (nm) processor.
  -In 2014, Intel launched an even smaller, more powerful 14nm chip;
  -and today, the company is struggling to bring its 10nm chip to market.

- For perspective, one nanometer is one-billionth of a meter, smaller than the wavelength of visible light. The diameter of an atom ranges from about 0.1 to 0.5 nanometers.

Synchronies circuit:

- What do processor between two edge clocks?
  **** Wait***** (Clock restriction)

- How get clock suddenly for all part chip?
  ****By specific clock buses**** (More Buses)

Asynchronies circuit:

- Never clock:
  1- Smaller needed (mobile equipment).
  2- Less emitting (noise).
  3- More execute (mean speed component).
  4- Different work can execute simultaneously, different speeds.
  5- Less dimension, more complex application.
    - Negotiable solution: mixed circuits.
    - Intel Pentium 4 has some parts asynchronous.
    - SUNFLEETzero- prototype asynchronous chips.

  6- Better scalability.

Challenges (Differences from Sequential Computing)

- Heterogeneity
- Latency
- Remote Memory Vs Local Memory
- Synchronization
  Concurrent interactions the shared resources.
- Partial failure
  Applications need to adapt gracefully in the face of partial failure.
- Need for “openness”
  Open standards: key interfaces in software and communication protocols need to be standardized
- Security
- Denial of service attacks
- Mobile code
- Scalability
- Transparency

Transparency in Distributed Systems

- Access transparency: enables local and remote resources to be accessed using identical operations.
- Location transparency: enables resources to be accessed without knowledge of their physical or network location (for example, which building or IP address).
- Concurrency transparency: enables several processes to operate concurrently using shared resources without interference between them.
- Replication transparency: enables multiple instances of resources to be used to increase reliability and performance without knowledge of the replicas by users or application programmers.
- Failure transparency: enables the concealment of faults, allowing users and application programs to complete their tasks despite the failure of hardware or software components.
- Mobility transparency: allows the movement of resources and clients within a system without affecting the operation of users or programs.
- Performance transparency: allows the system to be reconfigured to improve performance as loads vary.
- Scaling transparency: allows the system and applications to expand in scale without change to the system structure or the application algorithms.