09/18/18

Distributed Computing in MATLAB

* In MATLAB each server is referred to as a lab
* labindex is which server/thread something is running on
* x=5 //x is 5 on each thread
* y=rand; y2=5 //y on thread 2 is 5, all other y is random
* If(labindex==4) x=5; on thread 4, x is 5, null on others
* When splitting sequential tasks [pipelined/systolic architecture]
  + Speedup when number of inputs(runs) is **MUCH** greater than number of steps(labs) [1 billion inputs to 3 steps]
  + Need time for each step(lab) to be close, otherwise bottleneck
  + With 3 labs/steps **ideally** you can approach 3x speed

Lecture 2 - reading Van Steen CH2 (55-102), Erl Ch11(255-279) due tues

* System models -
* Physical models: types of computers, no details to tech
  + Most basic HW & SW components located at networked computers
  + Early systems
    - LAN, small range of servers
    - Printers, NAS, email, file transfer
    - Mostly homogeneous
  + Internet-scale DS
    - Set of nodes over the internet
    - More nodes, global organizations
    - heterogeneous hardware, network, OS….
    - Typically desktop computers, static
  + Contemporary(now) DS
    - Not static, mobile
      * Need discovery
    - Not discrete
      * Nodes can be embedded in everything
    - Not Autonomous
      * Nodes act as part of a service
    - End result More heterogeneity
  + Distributed systems of systems
    - Ultra-large-scale (ULS) DS with new levels of complexity
    - System of systems(S0S)
      * Complex system consisting of a series of subsystems that come together to perform tasks
* Architectural models: system in terms of computational and communication
  + Focus of this class
  + Goal: meet present and future demands
  + Concerns: reliability, manageability, adaptability, cost-effective
  + 3 stage approach
    - Elements - diversity of approaches
      * 4 questions
        + Communicating entities - typically process in DS is a thread

Process can spawn threads(threads are more lightweight)

Primitive environments(sensors) don’t support abstraction

* + - * + Communication paradigm

Interprocess - low-level support

Remote invocation - most common - 2 way exchange

Request-reply protocol

Remote procedure calls(RPCs

Remote method invocation(RMI)

Indirect communication

Group

Publish-subscribe

Message queues

* + - * + Roles and Responsibilities

Client-server

Clients request from server and receive info

Most common, very simple

Basically internet

Peer-to-peer

All process act like the same thing, all running same program

Non-centralized

Scales better than client-server

* + - * + Placement

Mapping services to multiple servers

Determine how to distribute or replicate data

Cache - store recently used data closer to client(s)

Needs to be updated periodically

Applet - code downloaded from web works on local machine(can run multiple times), reduces lag

* + - Patterns -
    - Middleware
* Fundamental Model
  + Only most essential pieces needed to understand system
  + Interaction Model - reflects communication delays and accuracy of processes due to those delays
  + Failure models - classify failures

TUNED OUT FOR REST OF CLASS

Test question on slide 24 split application we all know into 3 parts(1 was data/messages)