Challenges: System Perspective (1)

- Communication mechanisms: E.g., Remote Procedure Call (RPC), remote object invocation (ROI), message-oriented vs. stream-oriented communication
- Processes: Code migration, process/thread management at clients and servers, design of software and mobile agents
- Naming: Easy to use identifiers needed to locate resources and processes transparently and scalably
- Synchronization
- Data storage and access
  - Schemes for data storage, search, and lookup should be fast and scalable across network
  - Revisit file system design
- Consistency and replication
  - Replication for fast access, scalability, avoid bottlenecks
  - Require consistency management among replicas

Challenges: System Perspective (2)

- Fault-tolerance: correct and efficient operation despite link, node, process failures
- Distributed systems security
  - Secure channels, access control, key management (key generation and key distribution), authorization, secure group management
- Scalability and modularity of algorithms, data, services
- Some experimental systems: Globe, Globus, Grid

Challenges: System Perspective (3)

- API for communications, services: ease of use
- Transparency: hiding implementation policies from user
  - Access: hide differences in data rep across systems, provide uniform operations to access resources
  - Location: locations of resources are transparent
  - Migration: relocate resources without renaming
  - Relocation: relocate resources as they are being accessed
  - Replication: hide replication from the users
  - Concurrency: mask the use of shared resources
  - Failure: reliable and fault-tolerant operation

Challenges: Algorithm/Design (1)

- Useful execution models and frameworks: to reason with and design correct distributed programs
  - Interleaving model
  - Partial order model
  - Input/Output automata
  - Temporal Logic of Actions
- Dynamic distributed graph algorithms and routing algorithms
  - System topology: distributed graph, with only local neighborhood knowledge
  - Graph algorithms: building blocks for group communication, data dissemination, object location
  - Algorithms need to deal with dynamically changing graphs
  - Algorithm efficiency: also impacts resource consumption, latency, traffic, congestion

Challenges: Algorithm/Design (2)

- Time and global state
  - 3D space, 1D time
  - Physical time (clock) accuracy
  - Logical time captures inter-process dependencies and tracks relative time progression
  - Global state observation: inherent distributed nature of system
  - Concurrency measures: concurrency depends on program logic, execution speeds within logical threads, communication speeds

Challenges: Algorithm/Design (3)

- Synchronization/coordination mechanisms
  - Physical clock synchronization: hardware drift needs correction
  - Leader election: select a distinguished process, due to inherent symmetry
  - Mutual exclusion: coordinate access to critical resources
  - Distributed deadlock detection and resolution: need to observe global state; avoid duplicate detection, unnecessary aborts
  - Termination detection: global state of quiescence; no CPU processing and no in-transit messages
  - Garbage collection: Reclaim objects no longer pointed to by any process

Challenges: Algorithm/Design (4)

- Group communication, multicast, and ordered message delivery
  - Group: processes sharing a context, collaborating
  - Multiple joins, leaves, fails
  - Concurrent sends: semantics of delivery order
- Monitoring distributed events and predicates
  - Predicate: condition on global system state
  - Debugging, environmental sensing, industrial process control, analyzing event streams
- Distributed program design and verification tools
- Debugging distributed programs

Challenges: Algorithm/Design (5)

- Data replication, consistency models, and caching
  - Fast, scalable access;
  - coordinate replica updates;
  - optimize replica placement
- World Wide Web design: caching, searching, scheduling
  - Global scale distributed system; end-users
  - Read-intensive; prefetching over caching
  - Object search and navigation are resource-intensive
  - User-perceived latency

Challenges: Algorithm/Design (6)

- Distributed shared memory abstraction
  - Wait-free algorithm design: process completes execution, irrespective of actions of other processes, i.e., $n - 1$ fault-resilience
  - Mutual exclusion
    - Bakery algorithm, semaphores, based on atomic hardware primitives, fast algorithms when contention-free access
  - Register constructions
    - Revisit assumptions about memory access
    - What behavior under concurrent unrestricted access to memory? Foundation for future architectures, decoupled with technology (semiconductor, biocomputing, quantum ...)
  - Consistency models:
    - coherence versus access cost trade-off
    - Weaker models than strict consistency of uniprocessors

Challenges: Algorithm/Design (7)

- Reliable and fault-tolerant distributed systems
  - Consensus algorithms: processes reach agreement in spite of faults (under various fault models)
  - Replication and replica management
  - Voting and quorum systems
  - Distributed databases, commit: ACID properties
  - Self-stabilizing systems: "illegal" system state changes to "legal" state; requires built-in redundancy
  - Checkpointing and recovery algorithms: roll back and restart from earlier "saved" state
  - Failure detectors:
    - Difficult to distinguish a "slow" process/message from a failed process/ never sent message
    - algorithms that "suspect" a process as having failed and converge on a determination of its up/down status

Challenges: Algorithm/Design (8)

- Load balancing: to reduce latency, increase throughput, dynamically. E.g., server farms
  - Computation migration: relocate processes to redistribute workload
  - Data migration: move data, based on access patterns
  - Distributed scheduling: across processors
- Real-time scheduling: difficult without global view, network delays make task harder
- Performance modeling and analysis: Network latency to access resources must be reduced
  - Metrics: theoretical measures for algorithms, practical measures for systems
  - Measurement methodologies and tools