Course Review

Kai Shen

Parallel Computing and Distributed Systems

- Parallel computing - compute sub-tasks simultaneously so that work can be completed faster.
  - SOR, Gaussian Elimination, MapReduce data processing, ...
- Distributed systems - a set of autonomous computers working together to appear as a single coherent system (to achieve unified goals).
  - World wide web, networked file system, instant messaging, ...
- Not mutually exclusive
  - Parallel computing more identified with what it does
  - Distributed systems more identified with what they are

Why Parallel Computing?

- The speed of uniprocessor will catch up with the need of computing?
  - Semiconductor feature size approaches physical limit
- Not always desirable to use the fastest hardware
  - FAWN
- Multiprocessors increasingly ubiquitous
  - Multicore revolution

Hardware Support for Parallelism

- Superscalar processors with instruction-level parallelism
  - Resolve artificial data dependencies
- Multiprocessor cache coherence and memory consistency
  - Bus snooping-based protocol
  - More complex for architecture without a shared bus
  - Also restriction on instruction level parallelism
Challenges of Shared Memory Parallel Computing

- Task decomposition and load balancing
  - Fine-grained decomposition and dynamic task assignment
- Synchronization
  - Performance and scalability
  - Ease of programming
  - Fine-grained/coarse-grained locking, transactional memory, ...
- Multicore resource management
  - Competition on shared resources (cache space and memory bandwidth) lead to unstable performance

Multicore Operating System

- Today's commodity OS is inadequate
  - Little consideration for multicores
  - Not scalable for many cores (competing for locks, expensive TLB shootdowns when changing page mapping)
- Less sharing is often better
  - Treating multicore systems as a distributed system - having independent OS instances, force explicit communications (message passing) when necessary
  - Use dedicated OS server to handle high conflict tasks serially

Distributed Memory Parallel Computing

- Benefits
  - Less requirement on hardware support
  - More scalable
  - More portable
  - Better performance?
- But more difficult to program
- MPI

Server

- Computer application serving (potentially many) interactive clients
  - Parallelism: many requests run concurrently in a server
- But unlike the traditional parallel applications
  - Easily partitioned to fine-grained requests without inter-request dependencies, no need for synchronization ⇔ embarrassingly parallel
  - Highly multiprogrammed, many context switches
  - Performance (quality-of-service) and resource accounting at each request
Future of Parallel Computing

- Increasingly common for low-end applications due to ubiquity of multicores
- Ease of programming will be more and more critical
  - Ease of writing correct parallel programs
  - Ease of creating high-performance parallel programs
  - MPI has been incredibly resilient so far ...
- Power/energy considerations will be more important
  - Low-power parallel systems
  - Power-aware adaptive parallelism management
- Parallel computing on heterogeneous platforms
  - Mapreduce on a heterogeneous server cluster
  - Specialized processors (GPU), asymmetric multicore

Why Distributed Systems?

- With advances in networking, including local-area network, Internet, and wireless sensor network
  - Distributed computers and people are increasingly connected
- Desired services in networked world:
  - Information exchange and sharing,
  - Resource sharing,
  - Collaboration,
  - ...
- Help achieve:
  - Scalability
  - Fault tolerance
  - Data availability, durability

Theoretical Foundations

- Synchronization
  - Time and clocks
  - Distributed snapshot
- Fault tolerance
  - Distributed consensus
  - Roll-back and recovery
- Replication
  - Consistency
  - Scalability
  - Performance

Internet Data Centers

- Large, cluster-based server farms
  - Unprecedented scale, simple commodity hardware
- Software system infrastructure
  - Distributed locking (Chubby Lock at Google)
  - Distributed file system (Google file system)
  - Distributed structured data (Bigtable)
  - Data center parallel programming (MapReduce/Hadoop)
- Cloud computing
  - Centralized resource provisioning
## Future of Distributed Systems

- Future of cloud computing?
  - Will we see fewer, more centralized data centers? Or is there room for custom-purpose not-so-giant systems?
- Has peer-to-peer computing delivered its promises?
- Increasingly user-driven
  - Penetration into handheld, mobile devices
  - Security and privacy
- Future driven by new, not-yet-known killer applications!
  - Location-aware services
  - Personal information manager and context-based search