

David Keil
Framingham State University

7. Models of concurrent computation and multi-stream interaction

1. Models of parallel and distributed computing
2. Parallel Random Access Machine
3. Multi-stream and indirect interaction
4. Other models of computation

David Keil Theory of computing 1/12 1

Inquiry

- Is *communication* part of *computation*?
- Does concurrency require new computational models?
- Is multi-agent, asynchronous interaction more powerful than sequential interaction?
- How does the brain work?

David Keil Theory of computing 1/12 2

Objectives

- 7a. To describe forms of concurrency
- 7b. To describe some models of parallel computation
- 7c. To describe connectionist and multi-stream models

1. Models of parallel and distributed computation

- Serial computing and concurrency
- Circuit model of parallelism: algorithm is hard-wired
- Synchronization in parallel computation
- Parallel Random-Access Machine (PRAM)
- Programming models: shared memory, threads
- Distributed computation

Serial computing

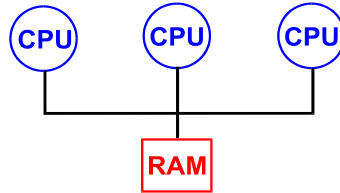


- The 50-year-old *Von Neumann architecture* defines most general-purpose computing today
- One processor; data and program are in memory
- The microprocessor carries out a *fetch/execute cycle*, retrieving and executing machine instructions one at a time
- Concurrent variants include multiprocessing, multitasking, and threads
- Multi-core machines support threads

Machine features to be modeled

- Design of many models of concurrent computing is concerned with:
 - Computational parallelism
 - Communications latency
 - Communications overhead
 - Communications bandwidth
 - Execution synchronization
 - Memory hierarchy
 - Network topology

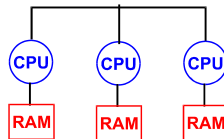
Shared-memory architectures



- All CPUs access all memory as global address space
- Memory access may be uniform (UMA) or non-uniform (NUMA)
- Where processors use cache, *cache coherency* may be an issue

Distributed-memory and hybrid architectures

- Memory addresses in *distributed-memory systems* do not map between CPUs; no global address space exists



- Cache coherency does not apply
- Scalability is in thousands of CPUs
- *Hybrid* distributed-shared memory systems network multiple symmetric multiprocessors

Flynn's taxonomy

Single instruction single data (SISD)

- Sequential machine, e.g., PC

Single instruction multiple data (SIMD)

- All CPUs synchronously execute the same instruction, but on different data items

Multiple instruction multiple data (MIMD)

- Every processor may execute a different instruction stream on a different data stream
- May be synchronous or asynchronous

Multiple instruction single data (MISD)

- Uncommon; used for fault tolerance

Parallel programming models

- *Programming model* is an abstraction lying above hardware and memory architecture and is not architecture specific
- *Shared memory*
 - May be implemented using distributed physical memory as virtual memory
 - Programming interface is simple, using global addresses
- *Threads*
 - Like subroutines, called in parallel, sharing some memory
 - UNIX (POSIX) and Java support threads

Other parallel programming models

- *Message passing*
 - Tasks exchange data on network via messages
 - MPI (1994) is de facto industry standard
 - MPI may be used in shared-memory architectures
- *Data parallel*: parallel tasks work on different parts of a common data structure
- *Hybrid*: combines the four above programming models, e.g., MPI with threads or shared memory

Models of distributed computing

- *Distributed system*: group of computers with local memory that communicate autonomously by message passing via a network.
- *Distributed computing*: “Computing with multiple and potentially asynchronous processors that may be widely dispersed”
- Example of a problem in distr. computing: *Dining Philosophers*, who sit around a table with one chopstick between each pair of neighbors. What protocol avoids starvation?

2. Parallel Random Access Machine

- Abstracts from synchronization concerns
- Assume all processors have access to *shared* memory in constant time
- Each processor also has private memory
- Shared memory is globally addressable
- PRAMs are {concurrent, exclusive} read, {concurrent, exclusive} write – CRCW, CREW, ERCW, EREW
- PRAM is the standard assumption for theoretical work in parallel computation

PRAM definition

- Unbounded set of processors
- Unbounded global (shared) memory
- Set of input registers
- Finite program
- Each processor has accumulator, program counter, unbounded local memory, on/off flag
- Instructions: *load, store, add, sub, jump, jzero, read, fork, halt*
- *Fork* instruction activates another processor to execute starting at a certain address

Advantages of PRAM model

- *Naturalness*: operations per cycle is same as number of processors
- *Strength*: processors have access to all memory
- *Simplicity*: abstracts from communication and synchronization time
- *Usefulness*: as benchmark and as representative of existing parallel computers

Is PRAM a suitable model?

- *Advantages*:
 - Gives measure of ideal parallel time complexity
 - Encourages maximum possible parallelism
- *Disadvantages*:
 - Strong dissimilarity to actual computers
 - Ignores all costs of exploiting parallelism, such as cost of nonlocal memory access
 - Rigid synchronized execution pattern

Synchronization in parallel computation

- Coordination of concurrent tasks entails synchronization
- A method used in parallel computing is *barrier synchronization*
- A *barrier* is a point in a computation at which any thread or process must stop until all other threads/processes reach barrier
- *Bulk synchronous parallel* (BSP) computing is a model that uses barrier synchronization

The bulk synchronous parallel model

- Designed as a *bridge* between parallel hardware and software for parallel computing, analogous to role of Von Neumann model of serial computing
- *Definition:*
 - *Components* for processing or memory
 - *Router* to distribute messages between pairs of components
 - *Global synchronization* facilities
 - *A computation* is a sequence of *supersteps* that execute, bounded by *synchronization* steps

3. Multi-stream and indirect interaction

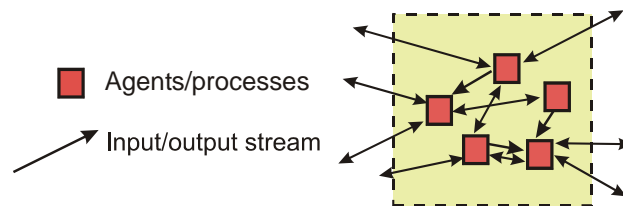
- Mission of a multi-agent system
- Multi-stream interaction
- Indirect interaction
- Stigmergy
- Message passing and its limitations
- Facebook
- Web 2.0
- Sensor networks

Mission of a multi-agent system

- Algebra and propositional logic provide rules for *evaluation* of formulas
- Algorithms compute recursively definable *functions*
- Sequential-interactive agents offer *services*
- Multi-agent systems accomplish *missions* requiring *quality of service* for all users
- Interaction in MASs may be *asynchronous*
- Mission may require a minimum QoS regardless of number of users (*scalability*)

Multi-stream interaction

- Occurs when more than two entities are interacting
- Multi-stream interaction is not just the composition of multiple sequential interactions



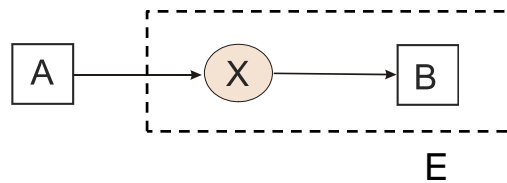
Special features of multi-stream interaction

In contrast to sequential interaction, multi-stream interaction may feature:

- *Nondeterminism* when attempts to write collide
- *Dynamic linking* and unlinking, creation/ destruction of nodes
- *Indirect* interaction via a shared environment

Indirect interaction

- Let A and E interact asynchronously.
- If E may be decomposed into X and B , where $X = E - \{B\}$, then A and B *interact indirectly* via X iff mutual causality holds between the behaviors of A and B .



Direct and indirect interaction

Direct interaction:

interaction via *messages*, where the identifier of the recipient is specified in a message.

Indirect interaction:

interaction via persistent, observable changes to a *common environment*; recipients are any agents that will *observe* these changes.

- Sequential interaction is direct
- Preconditions for indirect interaction:
 - Agents share access to parts of the environment
 - Persistence of environment
- *Example of indirect interaction:* use of semaphores in process synchronization (critical section problem)

Preconditions for indirect interaction

- Common malleable environment
- Changes must persist over time
- *Example:* synchronization in operating systems to solve critical-section problem
- *Solution:* semaphore

Properties of indirect interaction

- **Time decoupling (asynchrony):**
State changes persist
- **Anonymity:** Recipient ID not used in access
- **Space decoupling:** Agents need not meet
- **Non-intentionality:** Agents need not have goal of communicating
- **Hybrid nature:**
Physical environment may play role
- **Late binding** of recipient

Indirect interaction and multi-agent systems

- In a MAS characterized by *locality* of interaction and *mobility* of agents, it is only possible for agents to influence overall system behavior by use of indirect interaction
- Richness of multiagent interaction:
 - It is due partly to ability of each agent to interact with multiple others
 - Hence each agent interacts indirectly with *all* others (otherwise system partitions)

Ubiquity of indirect interaction

- **Social biology:** Social insects interact by modifying common structures or through pheromones
- **Operating systems:** Processes communicate via *semaphores* in shared memory
- **Coordination languages:** Shared *tuple spaces* enable coordination in Linda
- **Anatomy:** Cells exchange information via *hormones* in the blood stream
- **Economics:** A *market* is an environment for buyers and sellers that serves as a medium for indirect interaction

Stigmergy in nature

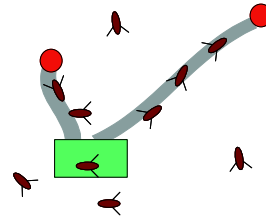
1. Ants foraging

2. Termites gathering chips into pile:

Move at random, pick up chip when encountered, put down when another chip found; the pile structure is used to coordinate creation of pile (*StarLogo*)

3. Slime mold dividing and aggregating:

These amoeba may aggregate by emitting a chemical, migrating toward its greatest concentration



Q: Is stigmergy essential for some missions?

Decentralized, self-organizing systems

- Decentralized and self-organizing systems lend themselves to flexibility and adaptiveness
- *Where required:* in environments that are dynamic, persistent, multi-agent, decentralized, and self-organizing.

Decentralized system: a multi-agent system whose components do not respond to commands from an active director or manager component, and do not execute prespecified synchronized roles under a design or plan.

Self-organizing system: a multi-agent system with a coherent global structure or pattern shaped by local interactions among components, rather than by external forces.

Emergent behavior

- The behavior that comes out of indirect interaction is called *emergent*.
- In emergent behavior, the overall system behavior is richer than the sum of the individual behaviors of the parts
- Emergent behavior is typical of *self-organizing systems*, which are *decentralized* and *non-hierarchical*

Food foraging problem for ants

- Food is scattered randomly
- Task is to take it to the nest
- Ants are small and limited in intelligence and communicating power
- Food may appear or disappear dynamically
- *A solution:*
 - Ants walk semi-randomly dropping pheromone
 - Ants tend also to follow pheromone trails
 - Ants carrying food drop special pheromone
 - Trails evolve toward short paths between nest and food

The message-passing model of concurrency

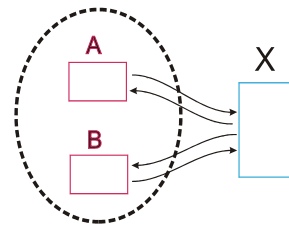
- Due to Robin Milner: CCS, π Calculus; associated with theory of concurrency and with process algebra
- These models capture the notion of *direct interaction by message passing*
- Axiom of concurrency theory:
interaction = message passing
i.e., atomic communication of a *message* from one *process* to another (targeted send/receive)
- Shared variables are deemed *processes*

Limitations of concurrency theory

- The foundation of the formalism of concurrency theory is the atomic communication of a *message* from one *process* to another.
- The message contains the identity of the recipient; in this sense interaction is seen as *targeted send/receive*.

Limitations of the message-passing model

- Message passing does not support properties of indirect interaction: anonymity, asynchrony, space decoupling, non-intentionality, and late binding
- Embedded and situated systems aren't supported
- Suppose agents A and B communicate via shared variable X
 - The message-passing model accounts for *direct* $A \leftrightarrow X$ and $B \leftrightarrow X$ interaction .
 - ...but not between A and B via X



An anomaly and a gap

- The practical reality is that indirect interaction is ubiquitous
- The existing theoretical modeling is of message passing, i.e., only direct interaction

4. Other models of computation

- Connectionist models (neural networks)
- Cellular automata
- Quantum computing
- DNA computing

Neural networks

- Human brain: 10^{11} neurons, each connected to average of 10^4 others. Max. switching time: 10^{-3} sec
- Each neuron “fires” (output pulse) if summed inputs exceed a threshold
- Artificial NN ANVINN drove car at 70 mph, 1993
- Good for problems with noisy, complex sensory information (cameras, microphones)
- Most common algorithm: Backpropagation
- Slow learning, fast application
- Hard for humans to understand correspondence between problem and NN solution

Cellular automata

- A CA is a vector or matrix of values (cells) whose values change according to rules that set new values according to its value and values of neighboring cells
- For a CA with 2 possible states per cell, and a cell's neighbors defined to be the cells on either side of it, $2^3 = 8$ initial patterns exist and $2^8 = 256$ possible rules
- CAs model local interaction among concurrent processors and display emergent chaotic behavior

Quantum theory

- Observation shows that both matter and energy are in discrete units (quanta) and energy travels at select frequencies only
- Quantum state of a particle may be seen as a wave function describing probabilities
- Another view is that a particle is spread out across space
- Particles may be entangled (interacting at great distances) and superposed (overlapping in space)

Quantum computation

- In a quantum computer, tape symbols can be either 0, 1 or a superposition of 0 and 1, i.e., 0, 1, and all values between, at the same time
- It is theorized that *superposed* quantum particles, stored as qubits (quantum bits) will enable quantum computers to work at astronomical speeds with only a few qubits
- It is known that Turing machines can simulate quantum computers

DNA computation

- Computational device is composed of enzymes and DNA molecules
- Implements basic logic gates (AND, OR, NOT) in DNA
- Catalytic DNA (deoxyribozyme or DNAzyme) catalyzes a reaction when interacting with certain inputs
- Faster and smaller than any other computer built so far
- Much lower power consumption than traditional silicon computers

Concepts

bulk synchronous	mission
parallelism	multi-stream interaction
cache coherence	multiprocessing
concurrent read	multitasking
direct interaction	parallel computation
distributed computing	PRAM
emergent behavior	self-organization
exclusive read	serial computings
Flynn's taxonomy	shared-memory architecture
indirect interaction	stigmergy
message-passing model	thread
MIMD	

References [Parallel]

- S. Akl. *Parallel Computation: Models and Methods*. Prentice Hall, 1997.
- B. Barney. *Introduction to Parallel Computing*. Lawrence Livermore National Laboratory, 2006.
- G. Cooperman. Lecture notes and drawings, COM 3640, Northeastern University, Winter 1998.
- Richard M. Karp and Vijaya Ramachandran. A survey of parallel algorithms for shared-memory machines. Tech. Report CSD-88-408, UC Berkeley, 1988.
- B. M. Maggs et al. Models of parallel computation: A survey and synthesis. Proc. HICSS'95.
- L. Valiant. A bridging model for parallel computation. CACM 33:8 (1990).

References [MAI]

- E. Bonabeau, M. Dorigo, G. Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford, 1999.
- J. Ferber. *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*. Addison Wesley, 1999.
- D. Goldin and D. Keil. Toward Domain-Independent Formalization of Indirect Interaction. *TAPOCS Workshop, Proc. WET ICE 04*, 2004.
- D. Keil and D. Goldin. Adaptation and Evolution in Dynamic Persistent Environments. In *Proc. FInCo2005, Edinburgh*, 2005.
- D. Keil and D. Goldin. Modeling Indirect Interaction in Environments for Multi-Agent Systems In *Proc. E4MAS*, 2005.
- R. Milner. *Communicating and Mobile Systems: The π Calculus*. Cambridge, 1999.
- M. Resnick. *Turtles, Termites, and Traffic Jams*. MIT Press, 1994.