Fundamentals of Parallel Computers

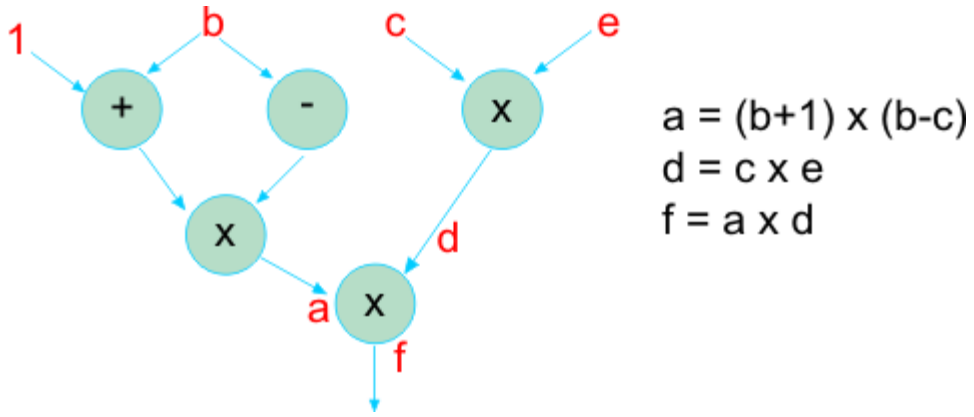
- Dataflow architecture
- Systolic arrays
- A generic architecture
- Design issues
- Naming
- Operations
- Ordering
- Replication
- Communication cost
- ILP vs. TLP

[From Chapter 1 of Culler, Singh, Gupta]

Previous Next

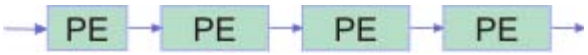
Dataflow architecture

- Express the program as a dataflow graph
- Logical processor at each node is activated when both operands are available
  - Mapping of logical nodes to PEs is specified by the program
- On finishing an operation, a message or token is sent to the destination processor
- Arriving tokens are matched against a token store and a match triggers the operation



Systolic arrays

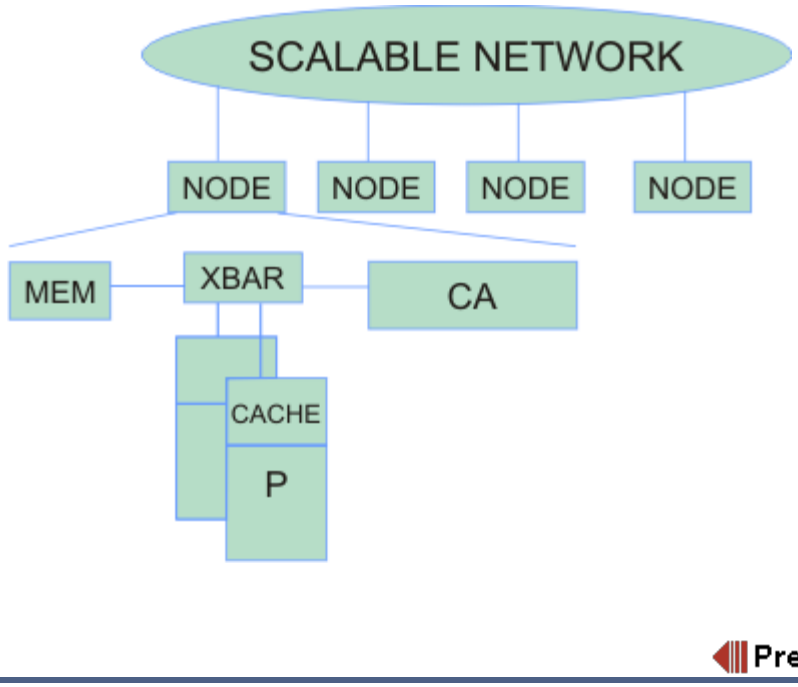
- Replace the pipeline within a sequential processor by an array of PEs



- Each PE may have small instruction and data memory and may carry out a different operation
- Data proceeds through the array at regular “heartbeats” (hence the name)
- The dataflow may be multi-directional or optimized for specific algorithms
  - Optimize the interconnect for specific application (not necessarily a linear topology)
- Practical implementation in iWARP
  - Uses general purpose processors as PEs
  - Dedicated channels between PEs for direct register to register communication

A generic architecture

- In all the architectures we have discussed thus far a node essentially contains processor(s) + caches, memory and a communication assist (CA)
  - CA = network interface (NI) + communication controller
- The nodes are connected over a scalable network
- The main difference remains in the architecture of the CA
  - And even under a particular programming model (e.g., shared memory) there is a lot of choices in the design of the CA
  - Most innovations in parallel architecture take place in the communication assist (also called communication controller or node controller)



Previous Next

Design issues

- Need to understand architectural components that affect software
  - Compiler, library, program
  - User/system interface and hw/sw interface
  - How programming models efficiently talk to the communication architecture?
  - How to implement efficient primitives in the communication layer?
  - In a nutshell, what issues of a parallel machine will affect the performance of the parallel applications?
- Naming, Operations, Ordering, Replication, Communication cost

Naming

- How are the data in a program referenced?
  - In sequential programs a thread can access any variable in its virtual address space
  - In shared memory programs a thread can access any private or shared variable (same load/store model of sequential programs)
  - In message passing programs a thread can access local data directly
- Clearly, naming requires some support from hw and OS
  - Need to make sure that the accessed virtual address gets translated to the correct physical address

Operations

- What operations are supported to access data?
  - For sequential and shared memory models load/store are sufficient
  - For message passing models send/receive are needed to access remote data
  - For shared memory, hw (essentially the CA) needs to make sure that a load/store operation gets correctly translated to a message if the address is remote
  - For message passing, CA or the message layer needs to copy data from local memory and initiate send, or copy data from receive buffer to user area in local memory

Ordering

- How are the accesses to the same data ordered?
  - For sequential model, it is the program order: true dependence order
  - For shared memory, within a thread it is the program order, across threads some “valid interleaving” of accesses as expected by the programmer and enforced by synchronization operations (locks, point-to-point synchronization through flags, global synchronization through barriers)
  - Ordering issues are very subtle and important in shared memory model (some microprocessor re-ordering tricks may easily violate correctness when used in shared memory context)
  - For message passing, ordering across threads is implied through point-to-point send/receive pairs (producer-consumer relationship) and mutual exclusion is inherent (no shared variable)

Previous Next

Replication

- How is the shared data locally replicated?
  - This is very important for reducing communication traffic
  - In microprocessors data is replicated in the cache to reduce memory accesses
  - In message passing, replication is explicit in the program and happens through receive (a private copy is created)
  - In shared memory a load brings in the data to the cache hierarchy so that subsequent accesses can be fast; this is totally hidden from the program and therefore the hardware must provide a layer that keeps track of the most recent copies of the data (this layer is central to the performance of shared memory multiprocessors and is called the cache coherence protocol)

Communication cost

- Three major components of the communication architecture that affect performance
  - Latency: time to do an operation (e.g., load/store or send/recv.)
  - Bandwidth: rate of performing an operation
  - Overhead or occupancy: how long is the communication layer occupied doing an operation
- Latency
  - Already a big problem for microprocessors
  - Even bigger problem for multiprocessors due to remote operations
  - Must optimize application or hardware to hide or lower latency (algorithmic optimizations or prefetching or overlapping computation with communication)
- Bandwidth
  - How many ops in unit time e.g. how many bytes transferred per second
  - Local BW is provided by heavily banked memory or faster and wider system bus
  - Communication BW has two components: 1. node-to-network BW (also called network link BW) measures how fast bytes can be pushed into the router from the CA, 2. within-network bandwidth: affected by scalability of the network and architecture of the switch or router
- Linear cost model: Transfer time = T_0 + n/B where T_0 is start-up overhead, n is number of bytes transferred and B is BW
  - Not sufficient since overlap of comp. and comm. is not considered; also does not count how the transfer is done (pipelined or not)
- Better model:
  - Communication time for n bytes = Overhead + CA occupancy + Network latency + Size/BW + Contention
  - $T(n) = O_V + O_C + L + n/B + T_C$
  - Overhead and occupancy may be functions of n
  - Contention depends on the queuing delay at various components along the communication path e.g. waiting time at the communication assist or controller, waiting time at the router etc.
  - Overall communication cost = frequency of communication x (communication time – overlap with useful computation)
  - Frequency of communication depends on various factors such as how the program is

written or the granularity of communication supported by the underlying hardware

ILP vs. TLP

- Microprocessors enhance performance of a sequential program by extracting parallelism from an instruction stream (called instruction-level parallelism)
- Multiprocessors enhance performance of an explicitly parallel program by running multiple threads in parallel (called thread-level parallelism)
- TLP provides parallelism at a much larger granularity compared to ILP
- In multiprocessors ILP and TLP work together
    - Within a thread ILP provides performance boost
    - Across threads TLP provides speedup over a sequential version of the parallel program

Previous Next