Communication Performance

Performance characteristics determine usage of operations at a layer

- Programmer, compilers etc make choices based on this

Fundamentally, three characteristics:

- *Latency*: time taken for an operation
- *Bandwidth*: rate of performing operations
- *Cost*: impact on execution time of program

If processor does one thing at a time: bandwidth $\propto$ 1/latency

- But actually more complex in modern systems

Characteristics apply to overall operations, as well as individual components of a system, however small

We’ll focus on communication or data transfer across nodes

Simple Example

Component performs an operation in 100ns

Simple bandwidth: 10 Mops

Internally pipeline depth 10 => bandwidth 100 Mops

- Rate determined by slowest stage of pipeline, not overall latency

Delivered bandwidth on application depends on initiation frequency

Suppose application performs 100 M operations. What is cost?

- op count * op latency gives 10 sec (upper bound)
- op count / peak op rate gives 1 sec (lower bound)
  - assumes full overlap of latency with useful work, so just issue cost
- if application can do 50 ns of useful work before depending on result of op, cost to application is the other 50ns of latency

Linear Model of Data Transfer Latency

Transfer time (n) $= T_0 + n/B$

- useful for message passing, memory access, vector ops etc

As n increases, bandwidth approaches asymptotic rate B

How quickly it approaches depends on T_0

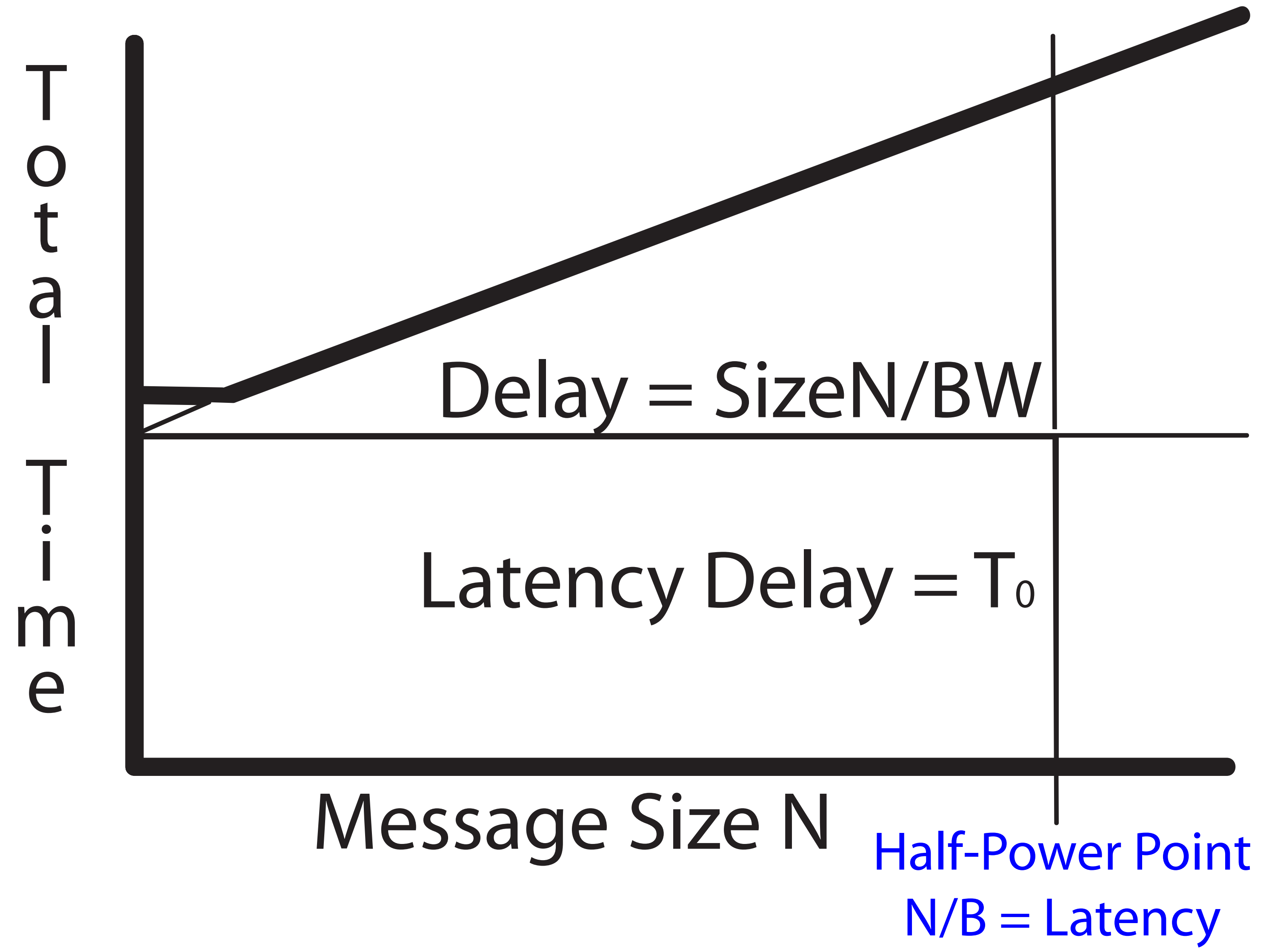
Size needed for half bandwidth (half-power point):

$$n_{1/2} = T_0 / B$$

But linear model not enough

- When can next transfer be initiated? Can delay costs be overlapped?
- Need to know how transfer is performed

Graph of Linear Model of Transfer Latency



Transfer time (N) = T_0 +N/B

Communication Cost Model

Comm Time per message= Overhead + Assist Occupancy + Network Delay + Size/Bandwidth + Contention

$$= o_v + o_c + l + n/B + T_c$$

Overhead and assist occupancy may be *f(n)* or not

Each component along the way has occupancy (1/bandwidth) and delay

- Overall delay is sum of delays
- Overall occupancy (1/bandwidth) is biggest (=>slowest) of occupancies

Comm Cost = frequency * (Comm time - overlap)

General model for data transfer: it also applies to cache misses

Summary of Design Issues

Functional and performance issues apply at all layers

Functional: Naming, operations and ordering

Performance: Organization, latency, bandwidth, overhead, occupancy

Replication and communication are deeply related

- Management depends on naming model

Goal of architects: design against frequency and type of operations that occur at communication abstraction, constrained by tradeoffs from above or below

- Hardware/software tradeoffs

Recap

Parallel architecture is important thread in evolution of architecture

- At all levels
- Multiple processor level now in mainstream of computing

Exotic designs have contributed much, but given way to convergence

- Push of technology, cost and application performance
- Basic processor-memory architecture is the same
- Key architectural issue is in communication architecture
  - How communication is integrated into memory and I/O system on node

Fundamental design issues

- Functional: naming, operations, ordering
- Performance: organization, replication, performance characteristics

Design decisions driven by workload-driven evaluation

- Integral part of the engineering focus

Outline for Rest of Course

Understanding parallel programs as workloads

  – Much more variation, less consensus and greater impact than in sequential

- What they look like in major programming models (Ch. 2)
- Programming for performance: interactions with architecture (Ch. 3)
- Methodologies for workload-driven architectural evaluation (Ch. 4)

Interconnection networks (Ch 10)

Latency tolerance (Ch 11)

Future directions (Ch 12)