Programming Models for NOW

Anoop Gupta
Computer Systems Laboratory
Stanford University

gupta@cs.stanford.edu

Programming Models

- **Two driving forces:**
  - programming convenience
  - what the hardware can support efficiently

- **Components:**
  - naming model + operations
  - performance model

==> Prog. model can not be separated from the underlying architecture

- otherwise, everything is Turing complete anyway
- need to examine potential evolution paths for NOW

Choices for Programming Models

- **Message Passing**
  - explicit replication; e.g., AM, MPI
- **Shared Object Space**
  - implicit replication; e.g., CST, SAM
- **Shared-Address-Space**
  - explicit replication; e.g., Split-C
  - implicit replication (cache coherent); e.g., ANL-C

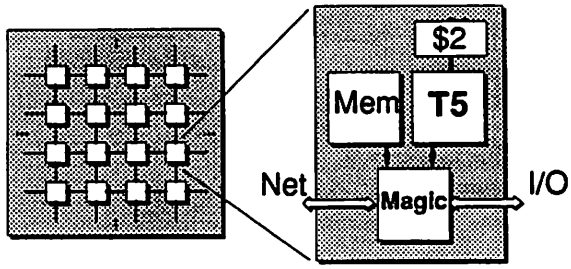
==> Model can substantially impact programming effort

Evolution Paths for NOW

- **Key issue:**
  - **NOT** geographic distribution,
  - **BUT** degree of coupling
- **Degree of coupling**
  - network interfaces directly to memory or I/O bus
  - naming model; translation and protection-checks
  - how and who manages replication and coherence
- **NOW exploits commodity; coupling depends on what is commodity:**
  - processor (e.g., FLASH)
  - processor + memory system (e.g., IBM SP-2, DASH)
  - processor + memory system + network (e.g., SS-10s over ATM)

Commodity: P

- **Useful to establish baseline for later comparison with NOWs**
- **A candidate architecure is Stanford FLASH**



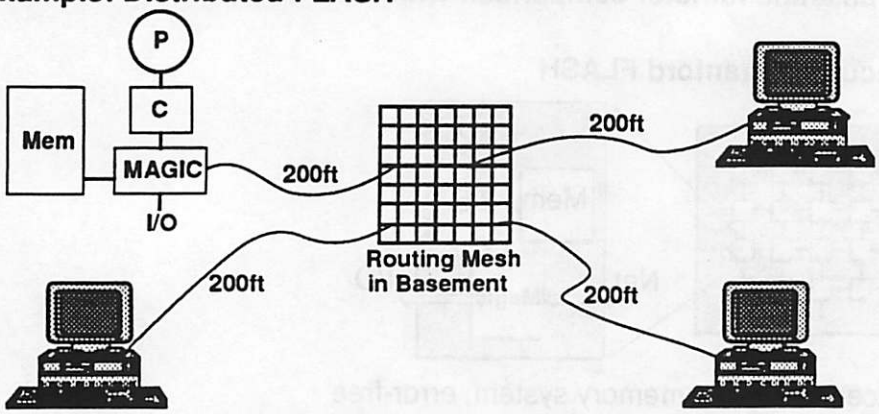
  - network interfaces directly to memory system; error-free
  - cache-coherent shared memory; TLB for translation and protection
  - replication and coherence: T5-CC + MAGIC
- **Expected performance profile:**
  - latency: ~1 µs for remote read of 128 bytes (including overheads)
  - bandwidth: several hundred MB/s

Commodity: P + M + N

- **Driving commodity applications**
  - video, audio, distributed database access, ...
  - latency: 10s or 100s of µs OK
  - bandwidth: 10s of MB/s is plenty
  - errors: substantial redundancy present
- **Today's ATM networks reflect this trend**
  - switch latencies are large
  - 622 Mbps translates to about 50 MB/s delivered
  - tremendous resistance to hop-by-hop flow control ==> dropped packets
- **Implications for NOW**
  - latency ~10 times larger, bandwidth ~10 times smaller than MPPs
  - mixed-bag programming model: f(application, performance goals)
  - heterogeneity will complicate matters further

Alternative Scenario: I

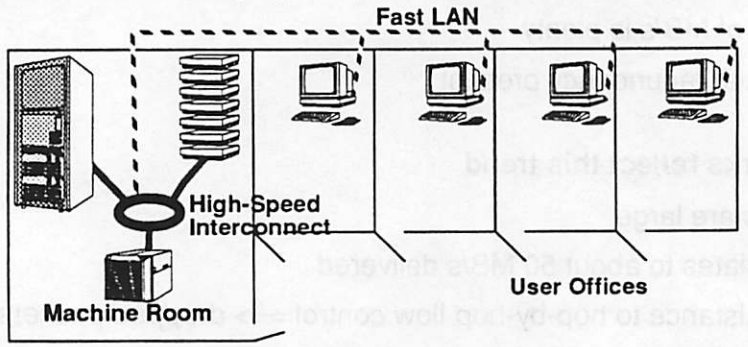
- **Example: Distributed FLASH**



  - network: switched, reliable, multi-Gbps links
  - programming model: cache-coherent sh-mem + msging
  - latency ~2.5us for 128 bytes; bandwidth 100-200 MB/s
  - will support limited heterogeneity and incremental purchase model
  - key challenge is in SW: resource management + fault containment

Alternative Scenario: II

- **Server-based computing environments:**
  - **on the desktop:** cheap display- and communications-centric boxes
  - **behind the scenes:** tightly-coupled MP compute servers + file servers



  - prog. model within a server should be cache-coherent shared-memory
  - fine-grain parallel programming across servers will be rare

- **Tight-coupling within servers can lead to better cost-performance**
  - basic queueing theory: a powerful server is better than lots of tiny ones

Closing Remarks

- **In the short term, NOWs obviously make sense**
  - people have workstations on their desks; good to use them when idle
  - however, I don't see a single winner programming model
  - not a general substitute for MPs

- **In the longer term, I believe:**
  - cool I/O-centric (display, audio, ...) computers on the desktop
    ==> these will **not** be the major source of shared cycles

  - dedicated servers will provide shared cycles, memory, and file storage
    ==> individual compute-servers will likely be tightly-coupled MPs

- **Key issue is NOT geographic distribution, BUT degree of coupling**