How *should* the current and future programming models drive interconnect requirements?

Vinod Tipparaju
Oak Ridge National Laboratory

What do applications see?

- Applications see an abstraction of a parallel machine as provided to them by the compiler and programming model

Some requirements of Programming models

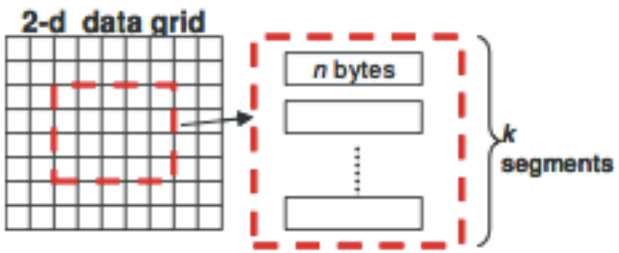
- Well defined data consistency semantics
  - Fence, Order, Read-Write consistency
  - Atomicity
- Support for Data transmissions
  - Vector, Strided, Structured Data Types in addition to contiguous
- Task pools and queues
  - *The* chosen way to scale
    - Most new languages/libraries are trying to do this
- Fault Resilience
  - What do we need to enable fault tolerance
  - Transparent is too complicated
- Collective communication
- Don't prevent interoperability
  - enable it, if you can

Consistency requirements

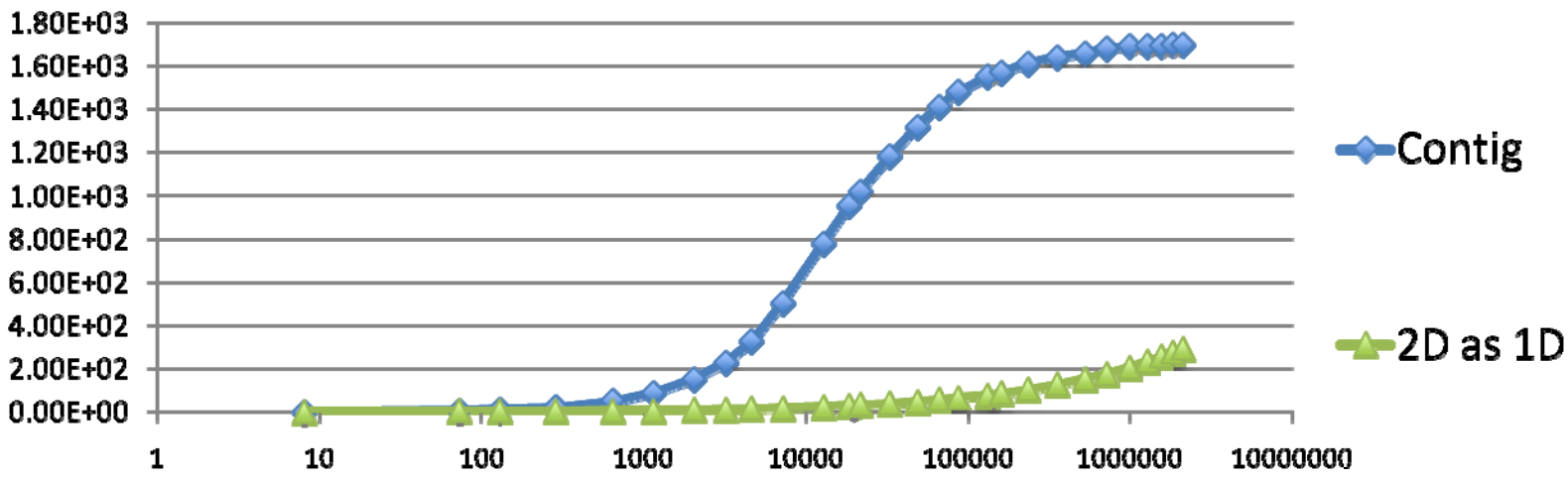
- Extending programming model semantics
  - Ordering
    - Despite the network
    - With-in the node
    - Packetization at different levels
      - Can we remove host level?
      - Makes host level flow control easier
    - Easy with non-contiguous support
  - Fence
    - Definition: An operation that gives you a state of memory that you can define
  - Coherency
    - Programming models guarantees some coherency
    - Relies on what NIC+Host provides
    - Strong, may be better
    - Simple guarantees like sequential consistency are not possible

- With-in Single OS image (OS constructs to share memory)
  - Load Store, Shared Memory, POSIX Threads,
  - OpenMP
- Parallel System (No OS constructs to share memory or your don't want to use them)
  - SHMEM, MPI, MPI-2
- GAS/PGAS (abstractions that let you work on 2 under the assumption that it is 1)
  - Language based
    - CAF, UPC, X10, Chapel,
  - Library based
    - Global Arrays

Data Transmission

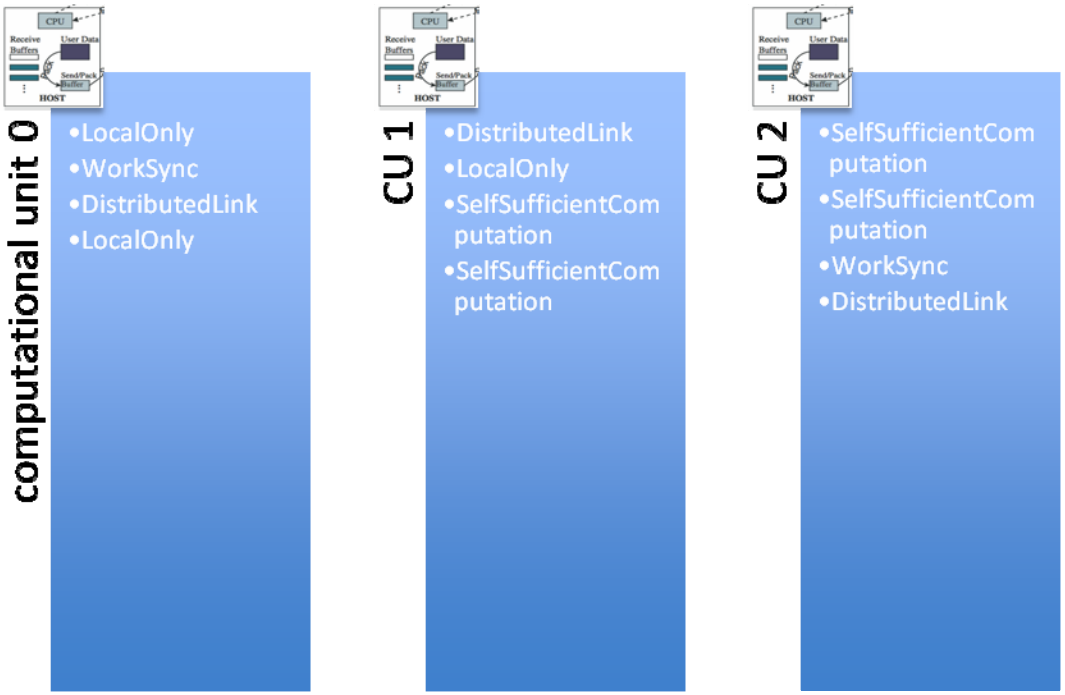


- Support for structured data types
  - Pack/unpack
    - Interrupt involved
    - OS scheduling
    - Cost of copy
    - Good for some shapes
  - Network Support
    - Desirable
  - Hybrid approach
    - Desirable when no alternative
    - Still involves interrupt



Task pools and queues

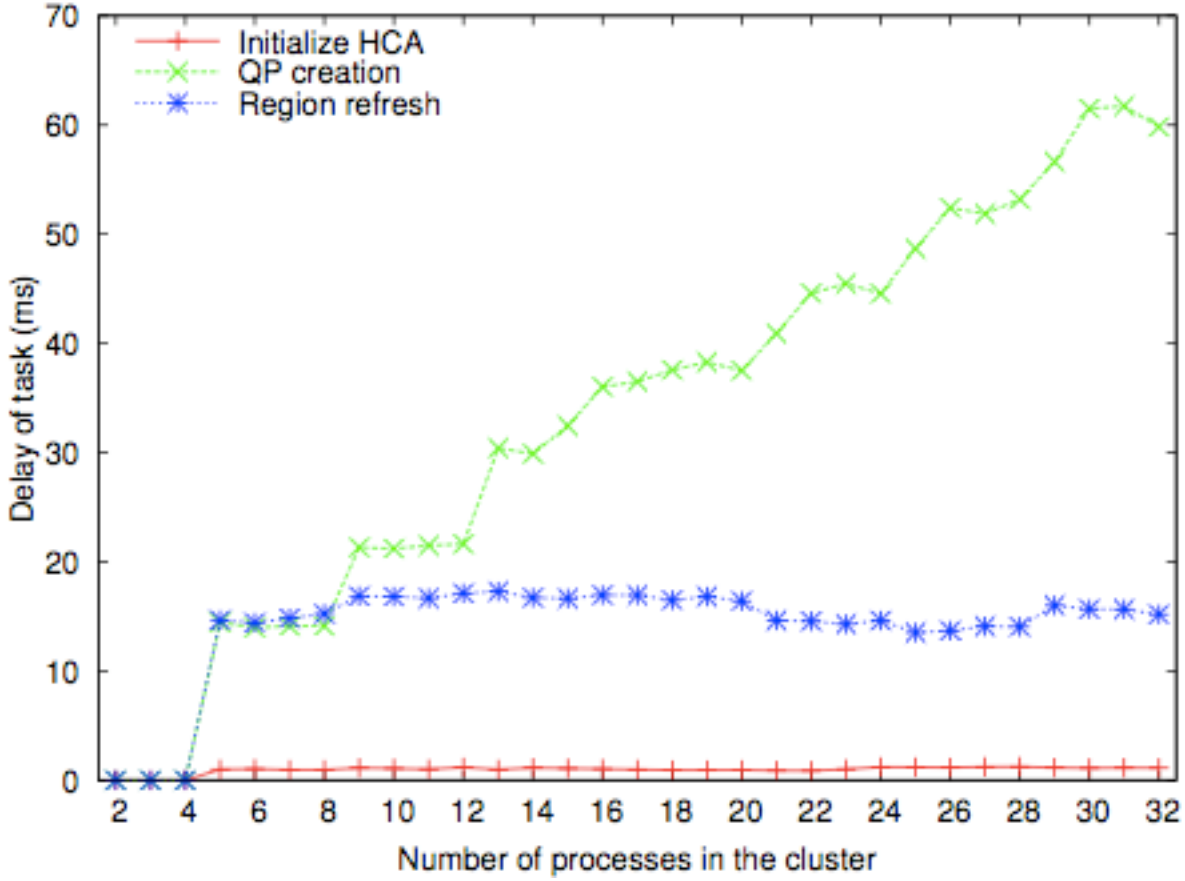
- Several programming model and language developers are looking at this today
  - Work scheduling
    - Who understands the process model the best?
    - OS scheduling
    - Cost of moving
  - Work shipping/stealing
    - Host involvement, locks
    - NIC controlled, serialized
  - Work Sharing



Fault Resilience

What can programming models do?

Can a transparent fault resilient solution possible



Fault Resilience