Memory System Utilization

- Memory level parallelism:
  - Cell - issued via DMAs, allows a few dynamic instructions to express vast MLP. However, requires predictive knowledge of data access patterns
  - XMT - expressed via multithreading, requires at least one instruction per word of MLP. Requires enough understanding of application to guarantee no RAW hazards - but does not predictive knowledge of data access
- Spatial and Temporal Reuse:
  - Cell - programmer must maximize efficiency of spatial (alignment issues) and temporal (via explicit DMA to local store)
  - XMT - Data cache hardware manages spatial and temporal reuse for local memory accesses.
    - Must organize code to leverage spatial locality at cache line granularity?
- Effectively hide/tolerate latency and utilize bandwidth
  - CELL: Multiple prefetch DMAs outstanding
  - XMT: Requires multiple threads and more than 1 outstanding memory operation per stream

Which application characteristics allow Cell/XMT to use memory subsystem effectively

- **Parallelism**
  - For both systems threading alone is not sufficient to saturate memory system, requires memory level parallelism
- **Predictability of data access**
  - Required for high Cell performance (depends how we define “high” performance)
  - Not required for XMT
- **Computational Intensity**
  - Cell can become computationally bound
    XMT cannot become computationally bound (is not good choice for computationally intense problems)
- **Locality**
  - Cell is structured to benefit from locality,
    XMT small from cache line transfers - but cannot take advantage of locality
- **Memory access regularity**
  - Cell has large effect (benefit unit stride),
    XMT little effect
- **Control flow regularity**
  - Cell has impact have to think about vector-like masks (speculation etc)
    XMT - virtually no impact,

Continued

- **Synchronization mechanisms**
  - Cell is traditional locks + mailboxes - could benefit from DMA synchronization if existed (scatters across DMAs),
    XMT has a rich set of sync primitives
- **Alignment**
  - Cell optimal performance with 128 byte,
  - XMT minor issue
- **Address translation**
  - Cell - conventional approach, local store is like pinned memory+ spe-spe transfer
    XMT uses segmented memory not pages, does not use virtual memory, TLB can cover physical address space of whole machine
- **Bank conflicts/contention**
  - Cell no special feature to avoid this (sequential access are spread over the banks),
    XMT - address hashing
- **Pointer chasing**
  - Cell needs to create virtual threads,
  - XMT inherently doing this with threaded arch

OTHER

- Execution model sometimes lacks needed features
  - Shared queue libraries (perhaps middleware can be utilized)

- Current interconnects lack implementation of global shared address space
  - Switch are not powerful enough to easily implement this