The Threaded VM is a “conventional” processor model

- API is much like existing threaded MP programming in C

- References to memory mapped by upper compiler are explicit

- Includes explicit primitives for IPC
  - Locks and barriers for traditional shared-memory programming
  - Simple message passing and DMA primitives for MPI-like frameworks
  - Explicit cache control for software-generated primitives

The Streaming VM is a model of a streaming engine that works only under the control of a nearby TVM node

- There is an API for stream kernels themselves

- Most of “SVM” API is for the controlling TVM thread

- Model looks like:

**Compiling Brook via Metacompilation**

- Our existing “upper-level” compilation framework

  - **Brook** source code is just an extension to C
    - Threads are actually in C itself
    - Stream kernels look like C function calls
    - “Calls” to these kernels are expanded into threaded control code
    - An example, with **Brook** additions in bold:

      ```c
      typedef stream Start * DoThread;  
      kernel void Add(Sources A, Sources B, out Sources C)  
        * C = A + B; 
      ```

  - **Metacompilation** adds functionality to an existing compiler
    - Brook code is mostly just C
    - So just add the ability to call occasional user routines within a thread
    - Compiler uses metadata to determine when user routines are necessary:
      ```c
      annotations { 
        decl ('out', 'int', 'reduce', ..); 
        function ('kernel'); 
        typedef ( 'stream' ); 
      }
      ```
    - Allows extensive reuse of existing compilation technology
    - Can also be used to add other extensions, like fixed-point arithmetic
    - **PCA applications** (FIR, FFT, beamforming) have been written in Brook and are now being tested

**SVM: Control Thread API**

- Threads can use several routines to send nonblocking commands to a nearby streaming node
  - **Stream LD/ST operations**
    - Load or store a strided stream from/to memory
      ```c
      DONE finished = VM_STREAMLOAD(int source_region, STREAM +address_to_srf, int #records, int recordSize); 
      ```
    - Synchronization can be performed earlier, if required
      ```c
      DONE finished = VM_STREAMMOVE(int source_srf_number, STREAM +address_from_srf, STREAM +address_to_srf, int #records, int recordSize); 
      ```
  - **Stream scatter/gather operations**
    - Load or store an indexed stream from/to memory
      ```c
      DONE finished = VM_STREAMGATHER(int dest_srf_number, STREAM +address_to_srf, PUB_STREAM +record_ptr_srf, int source_region, int #records, int #recordSize); 
      ```
  - **Stream move from node-to-node**
    - Efficient DMA of stream that avoids main memory
      ```c
      DONE finished = VM_STREAMMOVE(int dest_srf_number, STREAM +address_to_srf, PUB_STREAM +record_ptr_srf, int source_region, int #records, int #recordSize); 
      ```
  - **Kernel load and execute**
    - Sets up and starts kernels, the actual stream “operations”
      ```c
      DONE finished = VM_KERNELLOAD( ... ); 
      ```
    - **Inorder** output enforces in-order processing of stream elements
    - Thread becomes the “control” of a software-controlled O-O-O pipeline
      ```c
      DONE finished = VM_STREAMLOAD(int source_region, STREAM +address_to_srf, int #records, int recordSize); 
      ```
  - **Conditional streams**
    - Look at nearby stream elements, with a constant offset
      ```c
      DONE finished = VM_STREAMCOND_LOAD(int #records, int recordSize, int flags); 
      ```
  - **Conditional streams**
    - Optional marking API for potentially expensive stream accesses that do not occur for every stream element
      ```c
      _IS_COND, _OUT_COND, and _OUT_COND_IGNORER 
      ```
  - **Direct stream access**
    - It may be necessary to enable full addressing of streams occasionally
      ```c
      DONE finished = VM_STREAMACCESS(int addr, offset); 
      ```
    - Additional versions for accessing parameters to kernels
      ```c
      _SRC_RW, _SRC_RW, _SRC_RW, and _SRC_RW; 
      ```

Rest of code looks essentially like C now

- We may choose to restrict this to aid compilation