XPRESS Interfaces Meeting

January 23, 2014

- Adrian Serio, Alex Duchene, Hartmut Kaiser, Steve Brandt, Ezra Kissel, Thomas Sterling, Kevin Huck, Allan Porterfeild, Ben Martin, Kevin Bohan, Maciej Brodowicz, Dylan Stark, Ron Brightwell, Martin Swany, Abhishek Kulkarni, Bryce Lelbach, Jeremy Kemp
- Welcome: Hartmut Kaiser
    - This is the starting point of the rest of the project
    - Goals
        - Structure for HPX
        - Boundaries
        - Interfaces
        - Functionalities for domains
        - Assign responsibilities
    - Modules
        - Parcel Subsystem
        - AGAAS
        - Threading Subsystem
        - LCOs
    - Boundaries
        - Applications (XPI)
        - Measurement (APEX)
        - OS (RIOS)
- Charter: Thomas Sterling
    - Imperative to work together as one team
    - 2014 will be a pivotal year in American exascale work
    - We are in a Metastable state
        - Could fail or succeed
    - We need to come away in the next few days with the knowledge of how you and your team will coordinate with others to reach our goal
        - Goal: Build XPRESS HPX
    - We need to come away with the interfaces identified, categories defined, and we need a plan to integrate our projects
    - Rules of engagement
        - Needs to be a working workshop
- Ron Brightwell
    - XPRESS's success will depend on this meeting
- Allan P- It would be nice to have documentation of the interface of the Parcel/LCO interface which is available for the group even though it will be an internal interface
- Runtime Architecture
    - Hartmut
        - Description of software stack
            - XPI (IU/LSU)
            - HPX
                - LCOS (IU)
                - Threads (LSU)
                - Parcels (IU)

        - AGAS (TRON)
    - RIOS (SNL)
    - APEX (OU)
    - RCR (RENCI)
  - AGAS
    - We don't have a solid structure yet
    - Assume for now PGAS
    - Hartmut: that is dangerous
      - Agreed
  - Migrating threads
    - Hartmut: Do we want to do that
    - Thomas: We need to have it for consistency
    - Allan: We will have to have an AGAS to do this
  - Threading
    - Who is doing the scheduling, OS or runtime
  - Do threads have stack?
  - Terminating Threads
    - We have to have a method to kill threads or runtimes
    - Do we have a distinction between OS and user level threads?
    - LXK is not aware of other nodes
  - LCOs
    - Remove "Wait on Remote LCO" line
    - Wake a remote LCO
    - Trigger remotely wait locally
    - Linked pairs of LCOs?
    - We need to clarify what we mean by LCO
  - Questions from discussion
    - Changing Scheduler behavior
    - Terminating remote threads
    - Thread migration
    - Interaction with GAS
    - Performance monitoring
    - OS interface
      - Upcalls
      - Performance events
      - Power events
- RIOS: Dylan
  - Parts of LXK
    - Memory
    - NIC
    - Thread
      - How will thread systems interact?
      - Does runtime system maintain ownership of lightweight threads?
      - How can OS and the thread package interact?
      - Hartmut: OS needs to be aware of runtime system
    - Instrumentation control
      - Allan- talked to Kevin about this

        - Job Management
        - Topology
- Allan:
    - APEX should get most of its information from RCR
    - Asynchronous methods of RCR receiving information
    - Should RCR replace HPX performance counters?
    - Kevin: APEX is an interface between HPX and TAU for data
    - RCR is bound to OS and lives forever, APEX is bound to runtime
    - RCR is global state
        - Is it knowledgeable about other nodes
    - Should RCR be remote callable
- Break
- Thomas Sterling: We need to put names on these red lines
- Threads and Parcels: Kevin and Hartmut
    - Threading subsystem
        - Two distinct blocks of functionality
            - Managing single threads
                - Create new ones
                - Cancel threads
                    - OS might want a gun
                    - Do we want a cancellable LCO for fault tolerance
                - Change thread state
                    - Define number of thread states (IU has a document)
                        - Different names: Suspended, Yield, pending
                            - LSU Suspend= IU Pending
                - Provide means of synchronization
            - Schedule threads
                - Scheduling single threads and/or groups of threads
                    - Create now and run now? Vs create now run later?
                - Scheduling policies
                    - Different schedulers (LIFO,FIFO, work stealing, etc.)
                    - Which cores to use
                    - Enforce limits for how many threads can be created
    - Parcels->Threads interface
        - Create a new thread
            - Destination
            - Function pointer (Function ID)
            - Arguments
        - Cancel threads
        - Create Direct Action
        - Kill threads
    - Do we need to define more types of parcels
        - Thomas: There are use cases that we would want different types of parcels
        - Hartmut: All a parcel does is spawn a thread
        - Different states
        - Parcels can have attributes which allow us to have optimization purposes
        - Dylan: Why not give more access to thread API

        - Thread interface: Send Parcel
        - Parcel Interface: Parcels create thread
        - Hartmut: We can add functionality
- Lunch
- Dinner will be at Chimes
- During lunch a realization about APEX was
- When thinking about Optimizations; if the optimization needs information across boundaries we need to talk about it
- LCO/Threads interface
    - Commonly used LCOs: Dataflows and futures
    - Thread->LCO interfaces
        - Thread changes the state of an LCO
    - LCO->Thread
        - LCO invokes thread
    - LSU HPX LCOs
        - Creates a new thread or resumes a suspended thread
        - Threads and parcels are semantically the same
            - If the task is local it is packaged as thread
            - If the task is remote it is packages as parcel
    - IU LCOs
        - Create and initialize
        - Utility functions for types
        - Set
            - Set state
            - Set value
            - Set state/value
        - Query
            - Async state
            - Sync state
            - Value
        - Helpers
        - Destroy
        - Predicates
    - Should we be able to cancel a future?
        - Future is not an operation but a handle
            - Take this conversation offline
    - Should we have hints on how to schedule threads?
        - ????
    - LCO->Threads
        - Create a thread to ?????
    - Suspended threads are LCOs
    - A depleted thread is not managed by the runtime system
- APEX/RCR
    - Over lunch we realized that APEX could be integrated with HPX
    - RCR would be incorporated with RIOS
    - HPX->APEX
        - Timer start/stop

        - Sample counter
        - Init
        - Finalization
        - Phase start/stop
        - reset
    - APEX->RIOS
        - Needs a way to communicate
        - What happens if we are trying to get data after an HPX instance is over if we are reliant on parcels
        - **Semantics of the policy engine needs to be sketched out**
    - APEX->RCR
        - Write to a common memory space
        - Should it use parcels?
            - If I am a compiler do I want parcels?
    - APEX writes software data to RCR and RIOS writes hardware data to RCR
        - APEX last for an instance of HPX
        - RCR is persistent over multiple instances of HPX
    - RCR does not make policies
- Break
- ROIS
    - Where we are at
    - Portals 4, LXK (Kitten)
    - Don't have APIs yet
    - Want to create a list of requirements
        - Have use cases
    - Parts of RIOS
        - Threads
            - What are the differences between the HPX threads and the OS threads
        - Schedulers
            - In HPX allows for a number of schedulers to be used
            - These schedulers are directed by a resource manager
                - Resource manager assigns the scheduler resources
            - How would the resource manager handle multiple instances of HPX on one core?
        - Current implementation of HPX uses dedicated OS threads for timers, IO, etc.
    - Hartmut: We should try to avoid having two scheduler scheduling tasks
        - Give the OS the ability to do different policies based on the runtime requirements
    - What is shared between OS and runtime system
        - Task descriptor?
            - If there is shared functionality that systems on top of LXK use wouldn't that make things simpler?
            - HPX4 is not supposed to be restricted to LXK and LXK cannot be restricted to HPX4
            - Middle ground: could LXK implement useful things such as:
                - growable stacks
                - shared queues

        - Should users be able to override an OS scheduler?
        - There needs to be a negotiation protocol between the OS and the RTS
            - The RTS can request resources (it knows what it needs)
            - The OS can grant request or can tell RTS to wait
        - Ron: There is a concept of an "enclave" which is dedicated RTS hardware resources
    - LCOs
        - Bettor Memory allocator
    - AGAS
        - Restricted page tables
- Hartmut: Migration will implemented by summer
    - We need to talk about networking
        - Portals4
        - LXK to LXK communication
        - IO
    - Allen: we should focus on an introspective runtime
        - Tron: OS will have to have to have some introspection
        - Allen: Yes but most should be in runtime
- RIOS is still very fluid
- Tron: Threads should follow data
- Bryce: LSU is concerned about being cornered by the OS
- Tron: RTS can't know what the work load is
- Tron: Lets Discuss integration tomorrow in one group
    - Please think about the next steps to do and miles stones to think about
        - Think about what you need from other intuitions
- We will convene tomorrow at 9:00am