Programming Models Research Questions Working Group Out-brief

Session Chair: Thomas Sterling

March 11, 2015

Charter: Session V – Research Questions for Programming Models

- What are the key **research questions** in programming models that
  - must be answered by the research community
  - before widespread adoption of **new programming models** and/or
  - consideration of forming standards?
  - including
    - **semantic constructs** and
    - **intrinsic facilities** like interoperability with runtime

Definition of Terminology

Programming Model (PM):

A **set of abstractions** that simplify and structure the way the programmer thinks about and **expresses a parallel algorithm**.

It embodies the **principles** and **semantics** of **computational parallelism**, their interrelationships with runtime systems, and **name spaces** of all globally referenced objects.

Programming Model

abstractions and their semantics

- Parallelism
- Global name space
- Locality/distribution/memory hierarchy
- Synchronization
- Introspection
- Resilience
- Communication
- Time/space/energy management

Enumerating what the abstractions or implementations are

Reformulating the Questions around

abstractions to automate tedious or complex tasks

- **Data layout** : Domain-Decomp/Tiling/cache-blocking, data placement (*Raja, TiDA Kokkos)*
- **Thread Affinity**: (thread pinning, associate with data and with computing elements)
- **Threading** (Task, SPMD, DAG dependencies)
- **Memory spaces** (virtualize or not, different kinds, manual copying between)
- **Communication** (lightweight, MPI+x or all threads peers)
- **Handling nonuniformity** (async, tasking, runtimes)
- **Resilience**: (user-driven vs. system driven)
- **Modularity**: (kernel/driver model in frameworks)
- **Data Storage Model:** break assumption of contiguity? What abstraction to store data this way (post-posix/ object storage models`?)

Strategic Research Questions

- What issues/questions are actionable and what are those actions?
- What's *Next*? How do we make the next steps to planning?
- Topics are not questions: what unknowns or challenges have to be resolved?
- What are suitable programming models that will address the challenges of **exascale** programming?
- Note: A programming model is NOT a Lego-set: it comprises a complex semantic structure of mutually supportive constructs that addresses performance, energy, reliability, portability, and user productivity; hopefully enables generality, performance portability, and interoperability for composability

Sterling's list(1):

- What is the **performance model** that will drive the development of the programming model(s)?
- What are the forms, granularities, and controls for **parallelism** actions (tasks?) to achieve exascale?
- What is the abstract **name space** and **hierarchy** for IDing referential objects? What are **first-class objects**?
- What is the **parallel/distributed control state** and what objects comprise them? (beyond IP counters)
- How is **latency** addressed for minimization and hiding?
- What PM attributes can be incorporated to advance **productivity**?

Sterling's list(2):

- How does the PM represent information about **distribution of data** and **locality** of objects?
- In what ways does it rely on and interoperate with the runtime systems? What support does the runtime system have to provide? How does the PM ***inform*** **the runtime system**?
- What information can the PM provide to influence optimization of **energy usage**?
- What information can the PM provide to influence **fault tolerance** strategies?
- **I/O**?
- **Interoperability** with other PMs, similar and different?

latency

- Specialized threads
  - Continue to use what you were doing
  - Do something special
  - Future request
  - Break for subset task
  - Divides strategy in XSTACK programming
  - Purest model versus …

Focused general framework

- 3 year time frame
  - Not going to solve productivity
- What are the unique problems of platforms
- How do our codes differ from conventional commercial codes based on mathematical
- High priorities above plus to beyond exascale
- Considering other domains mining for valuable methods and technology

Short term long term

- In short term worry about operational issues, link up to portability workshops
- What is the right resilience model
- Debugging for non determinism
- Factors and limits of interoperability
- Data management abstraction model
- How do we make exascale program easier to use?

- How can programming models capture programmer intent
- How do PM serve as general research
- Quantitative and qualitative performance model for relative value; numbers evolve
- Essential aspect of abstract machine model
- Possibly more than one PM at each level
  - How to separate concerns and share information

- Code generation on or between nodes is one level
- How transferrable across various domains is a DSL useful and across a wide set of applications
- Can there be a portable abstract model?
- How to accelerate the creation of DSLs
- Analyzability for different PM; what are design decisions, automatic verification

- Functional programming: not only your father's Haskel
- Apps and algorithms are more restrictive than the language; how to take advantage
- E.g., C++ objects with restrictions
- Machine independent parallelism built into compilers, not just sequences?
- How express constraints on references to facilitate analysis?

- Language can build in constraints that are ambiguous to compiler. How?
- Should squash: will DSLs solver enough problems?
- What are the areas we know we can push it and what does the DSL need to look like? What are missing pieces?
- Distributed memory with a global name space?
- Roles between PM and runtime?

- Remote versus local: write in one model – symmetric of semantics for accesses? Oblivious to shared memory versus distributed memory.
- Can cache oblivious be used on large range of problems. E.g., global array model.
- Is it possible to avoid explicit communications management
- Can think of machine as hierarchical, flat, nodes?

- Difference between scientific locality and machine locality: do we think about, should they provide information from high model to low level?
- Programming model across disparate platforms.
- Who tunes codes
- Does runtime exploit info to tune code to machine?

- How general is a class of algorithms that we can build a generic code?
- What is the programming notation/model to map to all levels of hierarchy?
- How do you bridge the information between levels and across domain spaces?
- How to effectively perform aggregation automatically?

I/O

- How to automatically tile? Express dimensionality of the array.
- What restrictions might be placed on model task and underlying behavior?
- How do we do interoperability in the large? What are the ways of composing applications?
- Resource sharing and consistent data models as well as bindings depending on level?

- I/O persistent models for task based asynchronous?
- How do we make I/O land work with computer world?
- Marrying
- How can programming model capture resilience