Productivity and Performance Working Group

Breakout Summary

Kathy Yelick (co-chair)
Jeff Vetter (co-chair)
Sam Williams
Richard Murphy
Kunle Olukotun
Lenny Oliker

Josep Torrellas
Michael Perrone
Ananta Tiwari (Scribe)
Robert Harrison
Mike Merrill

Cell

- **Performance**
  - Efficiency of architecture
  - Ability to manage BW
  - Strong performance on compute intensive codes

- **Productivity**
  - High initial switching cost to new programming model/system
    - Selecting among models
  - Performance transparency
    - SPEs performance is relatively easy to predict
  - Should improve as software stack matures

XMT

- **Performance**
  - Works well with dependent pointer-chasing streams of references
  - Works well with irregular control flow
  - Load balance automatically managed
  - Abundant concurrency
  - Strong performance on graph algorithms

- **Productivity**
  - Simplified programmer goal (MTA)
    - Expose fine-grained parallelism
    - Keep thread pipes busy (>21?)
    - Start with serial code
  - Global memory helps with data orchestration
    - Random Access example

Cell and XMT

- **Both platforms can be productive for specific applications**
  - XMT/MTA compiler and model is straightforward
  - Cell software stack still maturing

- **Performance instability is a problem for diverse workloads for both platforms**
  - High variance in absolute performance across a range of applications
  - Key is to know this before you spend months trying all optimizations…

Observations (1)

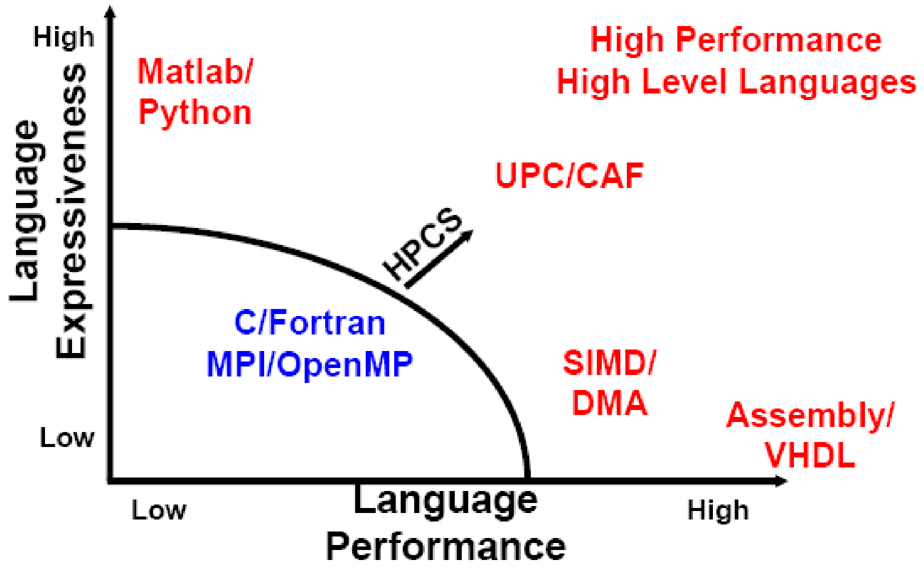
- **Management of concurrency and locality**
  - XMT provides hardware mechanism and policy to implement load balancing and [lack of] locality
  - Cell provides multiple hardware mechanisms for concurrency and locality; policy left to application
- **Maturity of tools for porting, optimizing**
  - Compilers often fall short
  - Move application work into runtime

Observations (2)

- **Application characteristics**
  - Computational intensity
  - Parallelism (multiple levels)
  - Data movement (spatial, temporal)
  - Predictability of computation and communication
    - Prefetching, thread management
- **With TCO of systems today, is it realistic to expect transparent application performance?**
- **Is it possible to build one architecture to satisfy all of these application domains without 'bad' productivity?**

Productivity of Languages -> Architectures

- What features of an architecture make it more productive?
- Performance complexity/instability



Source: HPCS Program