Are new languages necessary for multicore?

David Callahan
Distinguished Engineer
Parallel Computing Platform Team
Visual Studio
Microsoft

Why languages *evolve*:

- Programming languages capture *design patterns*
  - Function calls/return
  - Objects / Method Dispatch / Interfaces / Generic Programming
  - Iterators (CLU *v.* C++ *v.* C#)
  - Access to structured data (LINQ)
- Design patterns change over time
  - Increasing complexity with new abstraction conventions
  - Domain-specific uses
  - New programmer burdens
- Language constructs support application lifecycle
  - Architecture/development
  - Testing (especially defect analysis)
  - Performance analysis (especially “the dialog”)
  - Maintenance

Multicore is a new burden

- A least three design patterns demand support
  - Services (*aka Actors, CSP)* – asynchronously evolving agents with private state communicating via messages
  - Forall – (opportunistic) nested data parallelism over partially-ordered slices of data collections
  - Transactions – unordered updates to shared state
- And this ignores locality….
- Language bindings facilitate engineering automation
  - Load balancing
  - Resource management
  - Speculation / Deadlock-recovery

But, a new language?

- Yes: Isolation + Communication
  - Many concepts: domains, processes , messages, channels, contracts, choice, joins, data schema, time, new failure modes
  - This suggests a domain-specific language for “coordination” interoperable across many existing languages
- No, new features:
  - Forall must be tightly integrated
    - Fewer concepts but interwoven with data abstractions
  - Transactions should be tightly integrated to support forall
    - Insufficient to handle just at the coordination level


Microsoft®
Your potential. Our passion.™