# SEJITS and the quest for ubiquitous parallel software

Tim Mattson
Intel Labs
timothy.g.mattson@intel.com

Kayaker: Tim Mattson. Photo by P. Welle.

# OPL Pattern Language (Keutzer & Mattson 2010)

## Applications

### Structural Patterns

- Pipe-and-Filter
- Agent-and-Repository
- Process-Control
- Event-Based/Implicit-Invocation
- Arbitrary-Static-Task-Graph
- Model-View-Controller
- Iterative-Refinement
- Map-Reduce
- Layered-Systems
- Puppeteer

### Computational Patterns

- Graph-Algorithms
- Dynamic-Programming
- Dense-Linear-Algebra
- Sparse-Linear-Algebra
- Unstructured-Grids
- Structured-Grids
- Graphical-Models
- Finite-State-Machines
- Backtrack-Branch-and-Bound
- N-Body-Methods
- Circuits
- Spectral-Methods
- Monte-Carlo

### Finding Concurrency Patterns

Task Decomposition
Data Decomposition
Ordered task groups
Data sharing
Design Evaluation

### Parallel Algorithm Strategy Patterns

- Task-Parallelism
- Divide and Conquer
- Data-Parallelism
- Pipeline
- Discrete-Event
- Geometric-Decomposition
- Speculation

### Implementation Strategy Patterns

Program structure:

- SPMD
- Kernel-Par.
- Fork/Join
- Actors
- Vector-Par
- Loop-Par.
- Workpile

Algorithms and Data structure:

- Shared-Queue
- Shared-Map
- Parallel Graph Traversal
- Distributed-Array
- Shared-Data

### Parallel Execution Patterns

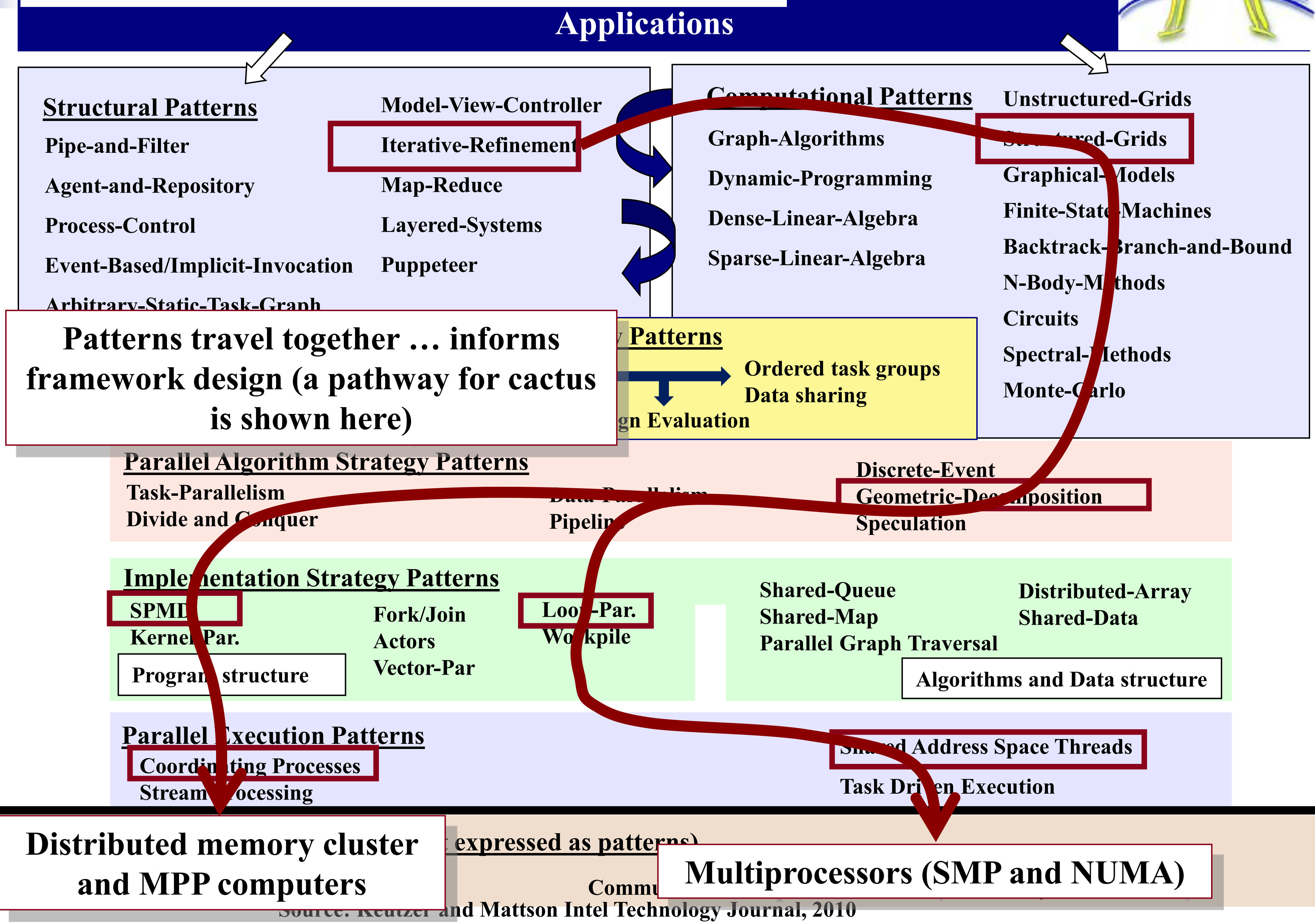
- Coordinating Processes
- Stream processing
- Shared Address Space Threads
- Task Driven Execution

### Concurrency Foundation constructs (not expressed as patterns)

Thread/proc management | Communication | Synchronization

Source: Keutzer and Mattson Intel Technology Journal, 2010

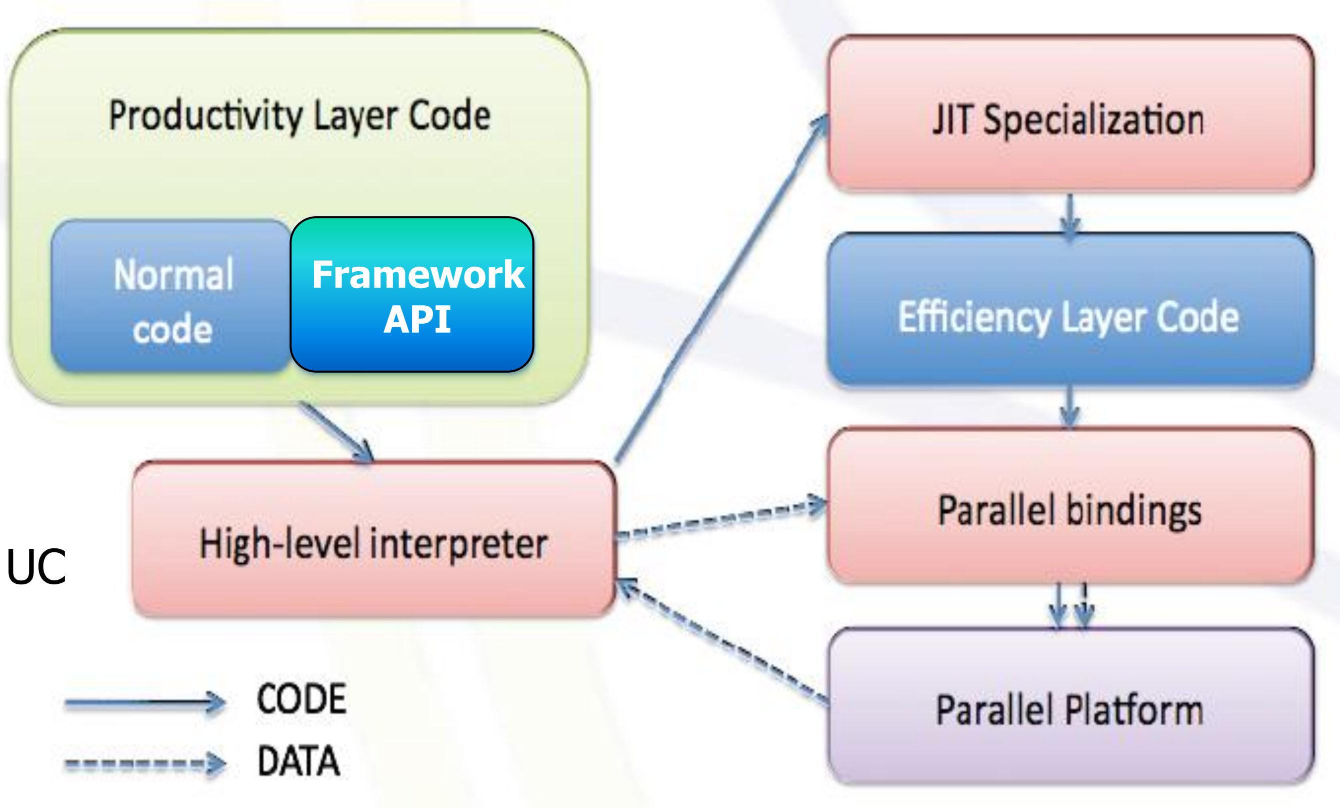
# OPL Pattern Language



# How do we get performance from frameworks?

- SEJITS: Scalable, embedded, just in time specialization
  - Code with a high level language (e.g. Python or Ruby) that is mapped onto a low level, efficiency language (e.g. OpenMP/C or CUDA).
  - SEJITS system to embed optimized kernels specialized at runtime to flatten abstraction overhead and map onto hardware features.

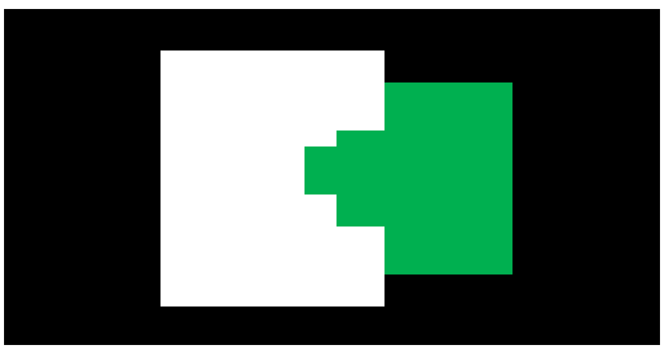
SEJITS comes from Armando Fox's group at UC Berkeley.



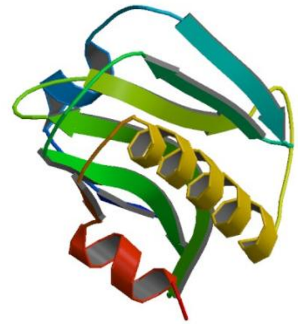

Bryan Catanzaro, Armando fox, Yunsup Lee, mark Murphy and Kurt Ketuzer of UC Berkeley, Mickael Garland of NVIDIA

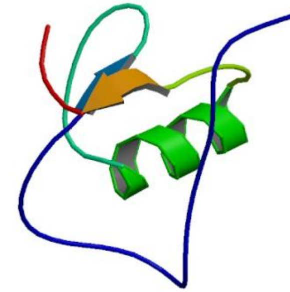
# Proof of Concept project: Shape Fitting

**How do these two shapes fit together?**



**Pretty obvious.**





**How do *these* two shapes fit together?** Not as obvious when dealing with complex, 3D molecular structures.

**Why does it matter how molecules fit together?** Because most biological processes involve molecular binding.

**Henry Gabb**: productivity, application programmer
**Tim Mattson**: specializer writer

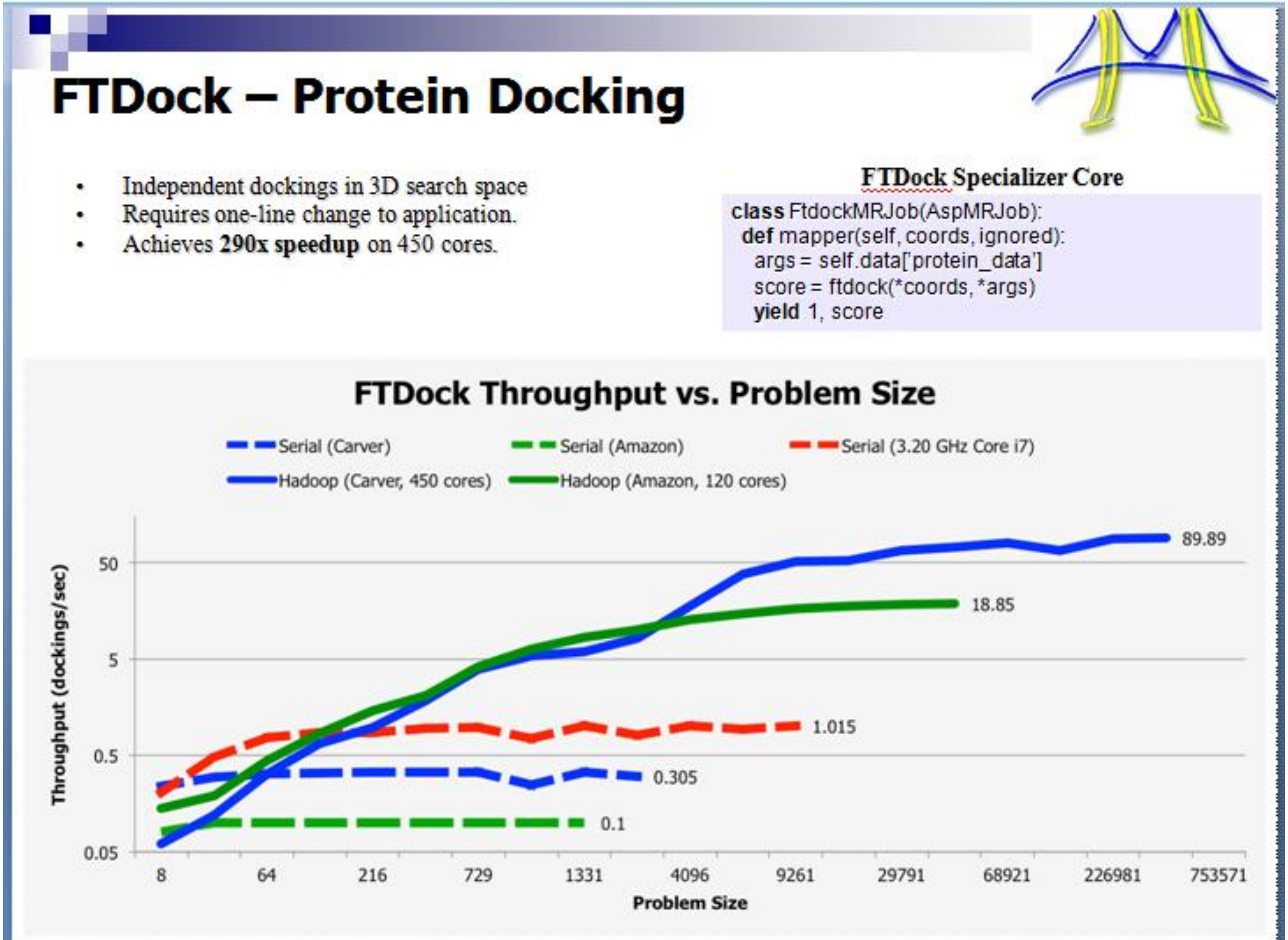
# Proof-of-Concept Results

- **For the productivity programmer:**
  - Pattern-based design of application
  - Significantly easier development:
    - Original version: 4,700 lines of C and Perl
    - New version: 500 lines of Python
  - Performance (16-core Xeon):
    - Serial: ~24 hours
    - Parallel: ~3 hours

Kayaker: Pat Welle. Photo by T. Mattson.

- **For the specializer writer**
  - Documentation was a work in progress. Training materials inadequate
  - Error feedback did not track original source code … required a SEJITS expert to find and fix bugs.
  - Assumed specializer writer was a hardcore python programmer (scipy, numpy, etc.).

# My Ah-ha moment!!!!

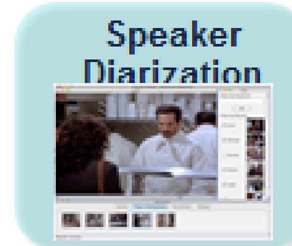



Source: M. Driscoll, E. Georgana, P. Koanantakool, 2012 ParLab winter Retreat.

# The Ah-ha moment for others at Intel

## PyCASP and Speaker Diarization

- Speaker Diarization … 50 lines of python/Pycasp code!!!! Highly productive programming model
  - Average faster-than-real-time factor &error rate
  - Averaged across 12 meetings (AMI corpus) [1]
  - Intel Westmere




| Implementation | Diarization Error Rate | Faster-than-real-time factor |
|---|---|---|
| State-of-the-art C++ | ~22% | 1X |
| PyCASP | 24.7% | 56x |

… and it could generate CUDA too if you wanted to run on a GPU (where it was 2X faster than the CPU)

[1] E. Gonina, G. Friedland, H. Cook and K. Keutzer. "Fast Speaker Diarization Using a High-Level Scripting Language" In Proceedings of IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU), Dec 11-15, 2011, Waikoloa, Hawaii

# The future of SEJITS

- Patterns → frameworks → SEJITS works as advertised.
  - I'm excited and eager to watch where you go with SEJITS.
- But … Great technology has users, not collaborators.
  - SETJITS is in the collaborator stage.  It needs users.
- SEJITS will disappear into the dustbin of computing history joining numerous parallel computing failures unless:
  - Show that one can build frameworks of reusable specializers.
  - Make SEJITS easier to use for the specializer writer.
  - Allow programmers isolated from the SEJITS team to use it.
- We don't need a product … we need a research prototype to validate the idea for application developers.
  - You aren't there yet.