Executive Summary

- Interest in patterns is exploding and the frontier of research problems grows monthly if not weekly
- Let’s take some time to inventory the frontier of problems and see who is interested in working on them
- Even with a Par Lab sized research program the frontier of problems is bigger than UCB can, or wants to, manage
- HELP!
Outline

1. Our Pattern Language
2. Application patterns
3. Methodological patterns
4. From patterns to frameworks
5. How do implementation constraints get reflected in patterns?
6. Implications of patterns on architecture and micro-architecture
7. Using patterns as a mechanism to analyze and organize parallel programming languages
8. Software exemplars/embodiments of patterns
9. Patterns and education
10. Upcoming workshops and events
11. (Design patterns for building hardware: CS250)
OPL: Current status

- Overview paper in second draft form – should be finished by 10/1/2009
- Drafts of patterns at wiki:
  - http://parlab.eecs.berkeley.edu/wiki/patterns/patterns
- Momentum on patterns everywhere – hence this talk …
OPL: Current research frontier

- Name?
  - OPL -- Overall Pattern Language, Our Pattern language,
  - MOAPL -- mother of all pattern languages
  - SAPL -- software architecture pattern language, system-architecture pattern language
  - PSAPL -- parallel software architecture pattern language
  - FSAP -- fundamental software architecture patterns -- software architecture foundational patterns
- Some pattern descriptions missing in wiki, some incomplete, no forcing function for revisions …
- Ultimately we’ll need a book … no?
- More case studies architecting parallel sw using OPL
- New classes of patterns – “efficient implementation patterns,” data layout patterns etc.

Owners: Kurt Keutzer and Tim Mattson … Ralph Johnson?

Outline

1. Our Pattern Language
2. Application patterns
3. Methodological patterns
4. From patterns to frameworks
5. How do implementation constraints get reflected in patterns?
6. Implications of patterns on architecture and micro-architecture
7. Using patterns as a mechanism to analyze and organize parallel programming languages
8. Software exemplars/embodiments of patterns
9. Patterns and education
10. Upcoming workshops and events
11. (Design patterns for building hardware: CS250)
NI Vision Assistant

- Image Analysis and Processing
  - Analysis: Histogram, line profiles, intensity measurements
  - Processing: Lookup tables, convolution kernels, spatial filtering, grayscale morphology
  - Operators
  - Frequency domain: FFT, Lowpass/highpass/mask FFT filters
- Particle Analysis
  - Image Segmentation: Thresholding, morphology segmentation
  - Binary Morphology
  - Particle Measurements
- Machine Vision
  - Edge Detection
  - Pattern Matching
  - Geometric Matching
  - Dimensional Measurements
  - Color Inspection
  - Binary Particle Classification
  - Golden Template Comparison
  - OCR
  - Instrument Readers
- Custom step/code entry (LV or library code)
- Code Generation
Application Patterns Frontier

• Still have no application domain in which we have defined and elaborated a set of “application patterns”
• Computer vision seems like a natural first area
• Need to form working group to define and elaborate a set of patterns for computer vision

• Owners: Andrade, Catanzaro, Murphy, Su, Sundaram, Keutzer

Outline

1. Our Pattern Language
2. Application patterns
3. Methodological patterns
4. From patterns to frameworks
5. How do implementation constraints get reflected in patterns?
6. Implications of patterns on architecture and micro-architecture
7. Using patterns as a mechanism to analyze and organize parallel programming languages
8. Software exemplars/embodiments of patterns
9. Patterns and education
10. Upcoming workshops and events
11. (Design patterns for building hardware: CS250)
Need to bring our thinking about methodological patterns to some conclusion
- Methodological patterns at each layer
- Single methodological back-plane?

HELP!
 Owners: Tim Mattson? Hank Hoffman?
Outline

1. Our Pattern Language
2. Application patterns
3. Methodological patterns
4. From patterns to frameworks
5. How do implementation constraints get reflected in patterns?
6. Implications of patterns on architecture and micro-architecture
7. Using patterns as a mechanism to analyze and organize parallel programming languages
8. Software exemplars/embodiments of patterns
9. Patterns and education – Michael Wrinn
10. Upcoming workshops and events – see wiki.
11. (Design patterns for building hardware: CS250)

People, Patterns, and Frameworks

<table>
<thead>
<tr>
<th>Application Developer</th>
<th>Application-Framework Developer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uses application design patterns (e.g. feature extraction) to architect the application</td>
<td>Uses programming design patterns (e.g. Map/Reduce) to architect the application framework</td>
</tr>
<tr>
<td>Uses application frameworks (e.g. CBIR) to develop application</td>
<td>Uses programming frameworks (e.g MapReduce) to develop the application framework</td>
</tr>
</tbody>
</table>

Design Patterns | Frameworks
Full Application Framework for LVCSR

**Top Level Attributes**
- Customizable attributes:
  - Recognition network structure
  - Input waveform format
  - Output word sequence

**Data Structure**:
- Feature vector info

**Fixed Structure**:
- Feature extractor
- Inference engine

**Inference Engine**
- Customizable Function:
  - Observation/Arc probability computation
  - Pruning heuristics
  - Track back data logging

Framework architecture customization:
- Active states vs active arc based traversal
- Push vs Pull state access

**Feature Extractor**
- Customizable Function:
  - Feature extraction algorithm

Patterns travel together

### Applications
- Structural Patterns
  - Agent and repository
  - Layered systems
- Combinatorial patterns
  - Monte Carlo
- Parallel Algorithm Strategy Patterns
  - Data parallelism
  - Pipeline
  - Discrete event
  - Recursive splitting

### Implementation Strategy Patterns
- Processor
  - Task queue
  - Shared data
- Message passing
  - Master/Worker
  - Shared queue
  - Distributed array
  - Loop parallelism
  - Shared data
  - Graph partitioning
  - Shared hash table
  - Memory parallelism

### Concurrent Execution Patterns
- Implementation methods
  - Advanced program counters
  - Coordination
  - Message passing
  - Mutual exclusion
  - Digital circuits
  - Speculation
  - Collective communication
  - Transactional memory
  - Data flow
  - Collective synchronization
  - PP synchronization
Premise

We need programming frameworks which comprehend a complete pattern flow, not just individual patterns. Framework can optimize across layers for efficiency.

Copperhead is a framework supporting our most common flow:

- **Data Parallel**
- **Strict Data Parallelism**
- **SIMD**

A framework for Data Parallelism

Copperhead is a subset of Python
- Collaboration with Michael Garland, Nvidia

Copperhead comes with “productive constraints” which ensure fidelity to pattern flow, allowing efficient compilation.
- Construct computation using only data parallel primitives and their compositions:
  - map, reduce, scan, sort, split, join, scatter, gather...

Copperhead includes runtime, JIT compiler.
Patterns and Frameworks Frontier

- Identify new
  - Application frameworks -- UCB computer vision team
  - Programming frameworks --
- Implement pattern-oriented frameworks:
  - Application framework: speech recognition: Jike Chong
  - Programming framework: Copperhead: Catanzaro et al.
- Apply
  - Programming frameworks to build application frameworks
  - Application frameworks to build applications … ???

Like to see these efforts expand beyond berkeley – Ralph Johnson?

Outline

1. Our Pattern Language
2. Application patterns
3. Methodological patterns
4. From patterns to frameworks
5. How do implementation constraints get reflected in patterns?
6. Implications of patterns on architecture and micro-architecture – Mark Murphy -- today
7. Using patterns as a mechanism to analyze and organize parallel programming languages
8. Software exemplars/embodiments of patterns
9. Patterns and education – UCB and Michael Wrinn
10. Upcoming workshops and events – see wiki
11. (Design patterns for building hardware: CS250)
Implementation constraints

- Hardly any piece of software is conceptualized or architected without some notion of the constraints under which it will be used
- How do we integrate this observation in the pattern language?
- Two broad approaches have been identified:
  - “softly” represent the constraints in the forces section of a pattern language – alerts architect and implementer to these forces
  - Quantitatively express the constraints in the structural patterns themselves

Constraints in Pipe and Filter

• Arrival constraints
  - Filter 1
  - Filter 2
  - Filter 3
  - Filter 4
  - Filter 5
  - Filter 6
  - Filter 7

• Back propagate arrival and destination constraints into individual filter constraints …

• Destination constraints
Implementation Constraints Frontier

- Low hanging fruit
  - Beef up forces section of patterns to reflect implementation forces which give more detailed implementation trade-offs
- Experiment with incorporating quantitative constraints into structural patterns and do a case study –
- Alexander’s A Pattern Language doesn’t help us here -- can Biomimicry help?
  - http://www.asknature.org

Can National Instruments/Hugo own this?

Outline

1. Our Pattern Language
2. Application patterns
3. Methodological patterns
4. From patterns to frameworks
5. How do implementation constraints get reflected in patterns?
6. Implications of patterns on architecture and micro-architecture – Mark Murphy -- today
7. Using patterns as a mechanism to analyze and organize parallel programming languages
8. Software exemplars/embodiments of patterns
9. Patterns and education – UCB and Michael Wrinn
10. Upcoming workshops and events – see wiki .
11. (Design patterns for building hardware: CS250)
Outline

1. Our Pattern Language
2. Application patterns
3. Methodological patterns
4. From patterns to frameworks
5. How do implementation constraints get reflected in patterns?
6. Implications of patterns on architecture and micro-architecture – Mark Murphy -- today
7. Using patterns as a mechanism to analyze and organize parallel programming languages
8. Software exemplars/embodiments of patterns
9. Patterns and education – UCB and Michael Wrinn
10. Upcoming workshops and events – see wiki.
11. (Design patterns for building hardware: CS250)

Can patterns organize this chaos?

Research Frontiers of Patterns
Keutzer
### Computational Patterns in TBB

<table>
<thead>
<tr>
<th>Pattern Type</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Backtrack Branch and Bound</td>
<td><code>task_group + mutex?</code></td>
</tr>
<tr>
<td>Monte Carlo Methods</td>
<td><code>parallel_for</code> (random number generation lacking!)</td>
</tr>
<tr>
<td>Circuits</td>
<td><code>out of scope</code></td>
</tr>
<tr>
<td>N-Body Methods</td>
<td>use loop parallelism or recursive parallelism</td>
</tr>
<tr>
<td>Dense Linear Algebra</td>
<td><code>out of scope (see Intel® Math Kernel Library)</code></td>
</tr>
<tr>
<td>Sparse Linear Algebra</td>
<td><code>=</code></td>
</tr>
<tr>
<td>Dynamic Programming</td>
<td><code>?</code></td>
</tr>
<tr>
<td>Spectral Methods</td>
<td><code>write FFT in TBB?</code></td>
</tr>
<tr>
<td>Finite State Machine</td>
<td><code>atomic&lt;T&gt;::compare_and_swap</code></td>
</tr>
<tr>
<td>Structured Grids</td>
<td>use loop parallelism</td>
</tr>
<tr>
<td>Graph Algorithms</td>
<td>use loop parallelism or recursive parallelism</td>
</tr>
<tr>
<td>Unstructured Grids</td>
<td>use loop parallelism or recursive parallelism</td>
</tr>
<tr>
<td>Graphical Models</td>
<td><code>=</code></td>
</tr>
</tbody>
</table>

### Summary of Pattern Support in TBB

**Strong points**
- Tasks
- Nested parallelism
- Explicit invocation

**Weak points**
- Patterns where complexity is in the waiting, not the computing.
- No direct support for SIMD

---

Arch Robison, Intel
Prog Langs and Patterns

- Need to leverage Arch Robison’s work on TBB to get the new generation of programming language developers to evaluate their languages using OPL
- Need some coarse grain organization of the current generation of popular programming languages … Cilk, OpenCL etc.
- Requires a senior dude … and/or perhaps a major corporation … Ralph, Tim, Michael Wrinn ….

Outline

1. Our Pattern Language
2. Application patterns
3. Methodological patterns
4. From patterns to frameworks
5. How do implementation constraints get reflected in patterns?
6. Implications of patterns on architecture and micro-architecture – Mark Murphy -- today
7. Using patterns as a mechanism to analyze and organize parallel programming languages
8. Software exemplars/embodiments of patterns – Strohmeir and Microsoft …
9. Patterns and education – UCB and Michael Wrinn
10. Upcoming workshops and events – see wiki.
11. (Design patterns for building hardware: CS250)
Outline

1. Our Pattern Language
2. Application patterns
3. Methodological patterns
4. From patterns to frameworks
5. How do implementation constraints get reflected in patterns?
6. Implications of patterns on architecture and micro-architecture – Mark Murphy -- today
7. Using patterns as a mechanism to analyze and organize parallel programming languages
8. Software exemplars/embodiments of patterns – Strohmeir and Microsoft …
9. Patterns and education – UCB and Michael Wrinn
10. Upcoming workshops and events – see wiki.
11. (Design patterns for building hardware: CS250)

Education and Patterns

- Half-day (ICCAD 2008) and full day (DAC 2009) public tutorials have been delivered on applying OPL
- OPL populated 2-3 lectures in Par Lab Bootcamps in 2008 and 2009
- Still need standard ½ day tutorial on “Architecting Parallel Software using OPL”
- Need to refactor UC Berkeley CS267 using OPL
- Can the Par Lab bear more pattern language work camps camouflaged as CS294 seminars?
- Should we pursue any of today’s research topics in CS294 seminars?
- Updates on other educational activities

- Owners: Keutzer, Mattson, Wrinn, Johnson, U of Victoria crowd …?
Outline

1. Our Pattern Language
2. Application patterns
3. Methodological patterns
4. From patterns to frameworks
5. How do implementation constraints get reflected in patterns?
6. Implications of patterns on architecture and micro-architecture – Mark Murphy -- today
7. Using patterns as a mechanism to analyze and organize parallel programming languages
8. Software exemplars/embodiments of patterns – Strohmeir and Microsoft …
9. Patterns and education – UCB and Michael Wrinn
10. Upcoming workshops and events – see wiki .
11. (Design patterns for building hardware: CS250)
But wait ....

• Prof. David Wessel is exploring the use of patterns in codifying music composition!