Microsoft

Microsoft Research Faculty Summit 2012

ADVANCING THE STATE OF THE ART

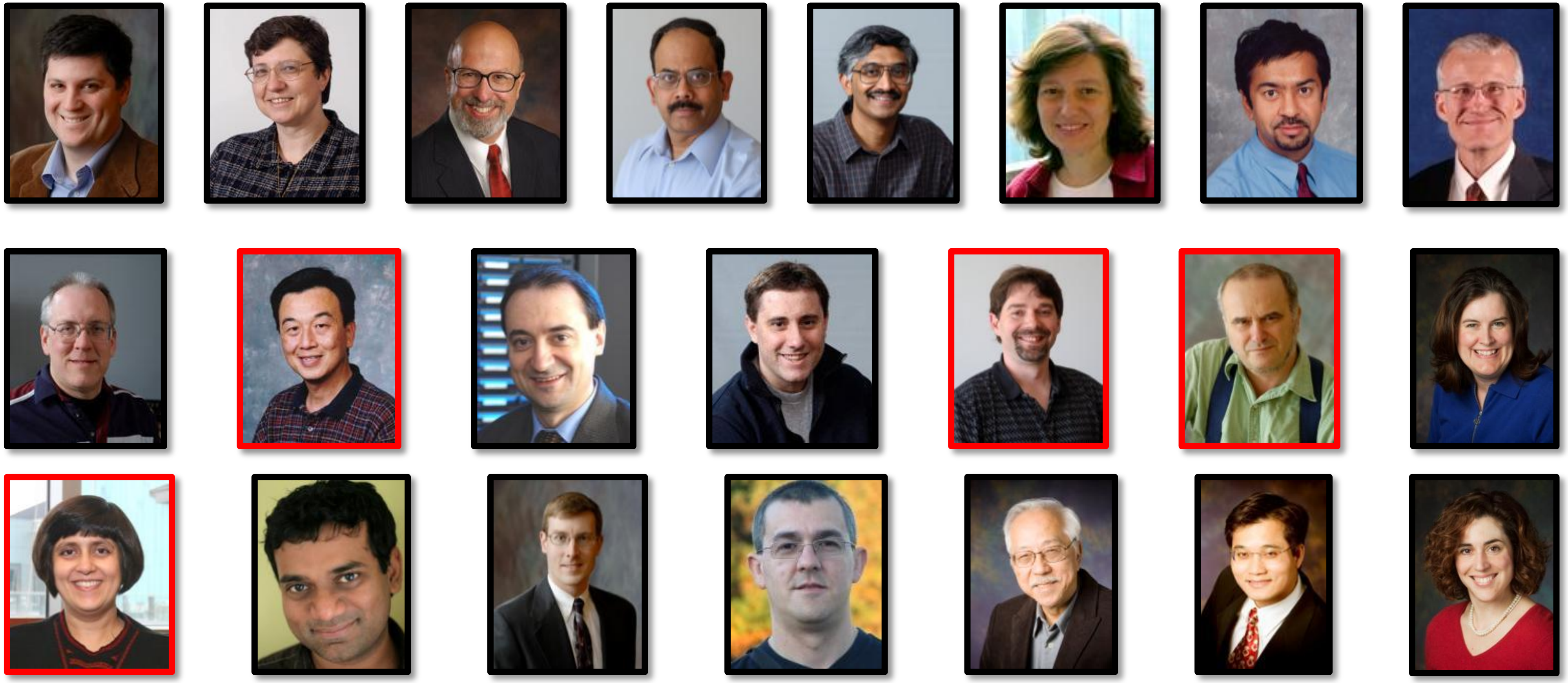
Making parallel programming synonymous with programming

Josep Torrellas,

Vikram Adve, Sarita Adve, Danny Dig, Minh Do, Maria Garzaran, John Hart, Thomas Huang, Wen-Mei Hwu, Ralph Johnson , Laxmikant Kale, Sam King, Darko Marinov, Klara Nahrstedt, David Padua, Madhusudan Parthasarathy, Sanjay Patel, Marc Snir, Craig Zilles,

Universal Parallel Computing Research Center Illinois

Illinois Team of Smiling Professors



Focus: Multicore Client Platforms and Applications

- Hard problem:
  - Need to parallelize individual app algorithms
  - Strict power budget for mobility
  - Large community of developers & use of specialized frameworks
  - “Killer apps” still unclear

The Illinois Vision

Make parallel programming easy

- Easy to write correct programs
- Easy to tune for performance/power

Focus on "easy" parallel programming patterns

- Provide a simple, efficient programming model atop complex HW and SW

Focus on end-goal: Better Applications

Four Pronged Approach

Applications:

- Compelling client apps that can benefit from parallelism
- Practical frameworks, algorithms and libraries

Safe Parallelism:

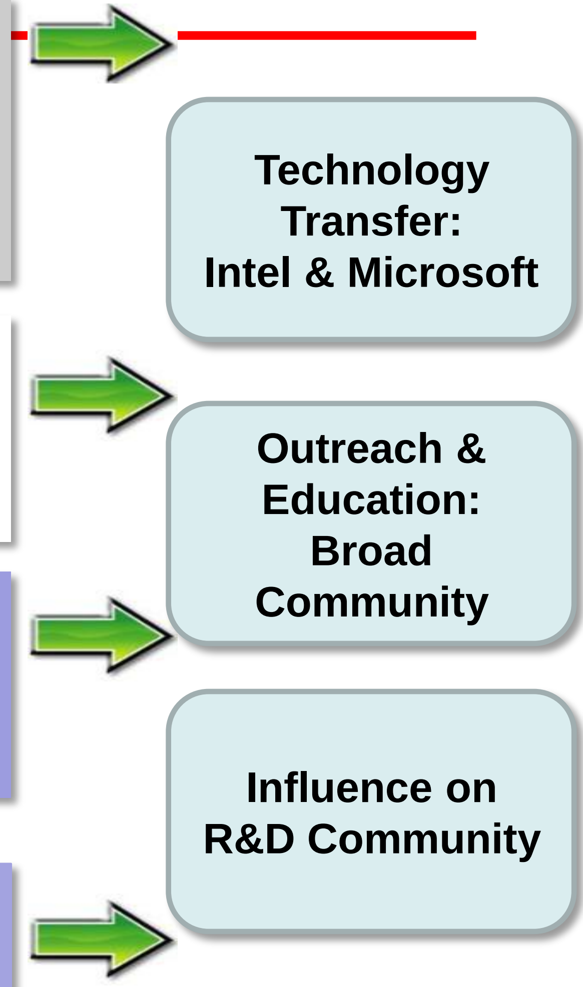
- Parallel programming without mysterious bugs
- Reduced debugging and testing time

Easy Tuning:

- Interactive, deep code transformations for performance and power
- Reduced tuning time

Scalable Hardware:

- Architectures that scale to 1K cores and are programable



Technology Transfer: Intel & Microsoft

Outreach & Education: Broad Community

Influence on R&D Community

Some Statistics

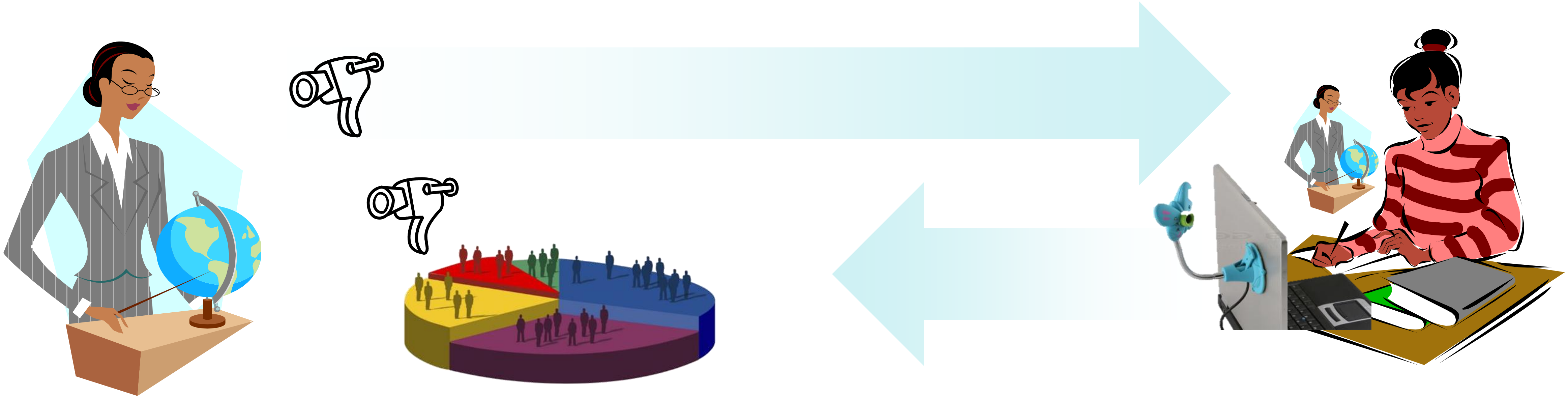
- 19 faculty involved at the peak (not all funded)
- 3 engineers
- 30 graduate students per semester (not all funded)
- Nearly 100 papers published
- 4 software publications (Vivid, ReLooper, DPJ, refactoring)
- Weekly seminars at Illinois, broadcasted
- Yearly summer course on parallelism at Illinois
- Encyclopedia of Parallel Computing
- 3 courses at Boeing and 1 in Singapore
- Keynotes, distinguished lectures, best papers, major awards, demos

APPLICATIONS

AvaScholar: Immersive Virtual Environment for Education

AvaScholar Instructor
Real-Time Deformable Stereo and Shape-from-Motion Reconstruction of Instructor and Visual Aids

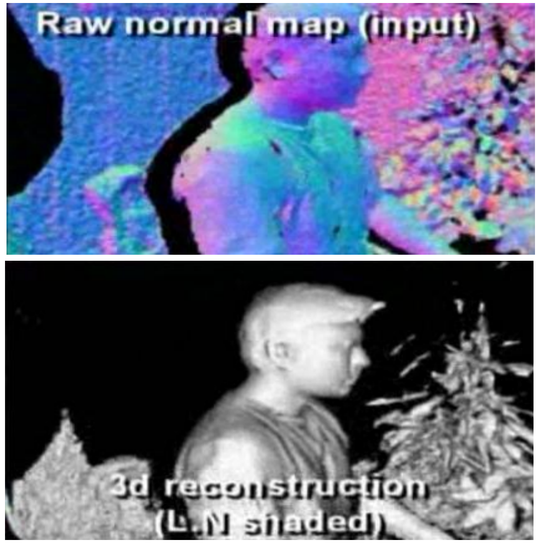
AvaScholar Student
Real-Time Agglomeration of Demographics, Engagement and Confusion of Remote Students



http://i2pc.cs.illinois.edu/

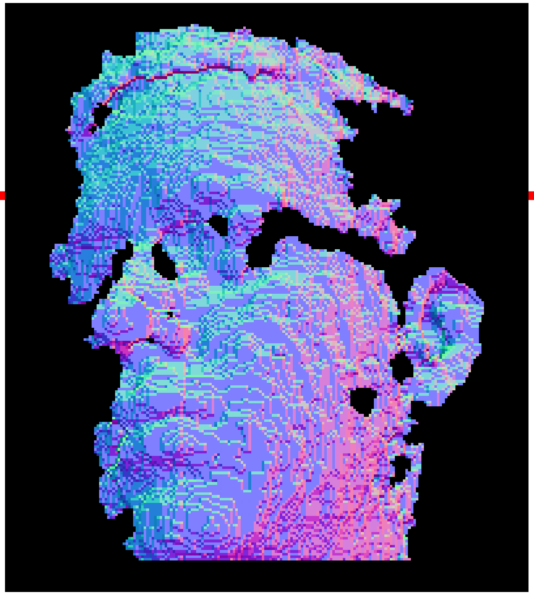
AvaScholar Instructor

- Real-Time shape-from-motion stereo reconstruction

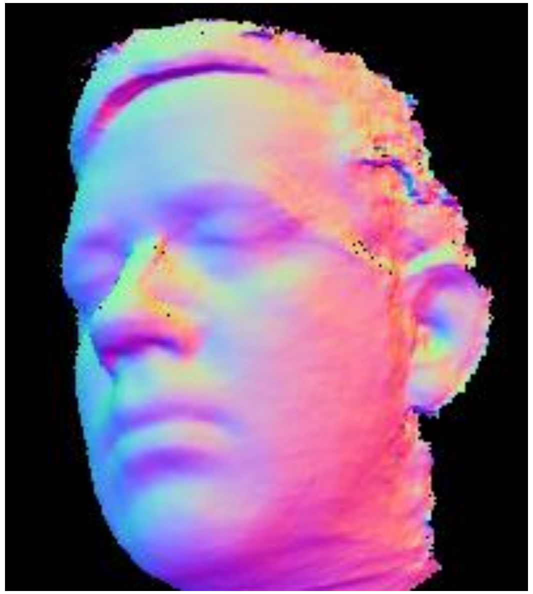


Scan Alignment

- Align multiple scans to refine model, remove noise and fill gaps
- Scaling up via increased parallelism
  to handle deformable models such as faces, hands and general shape articulation

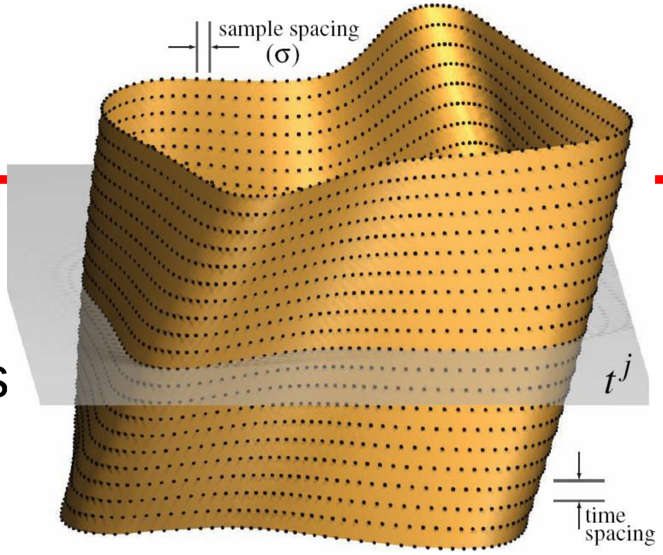


Single Scan

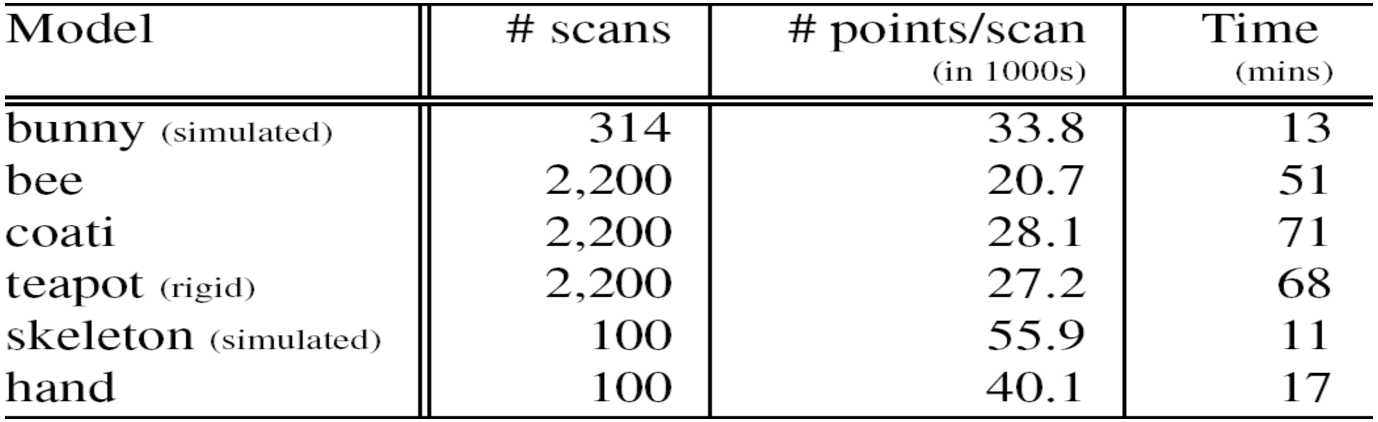


Aligned Scans

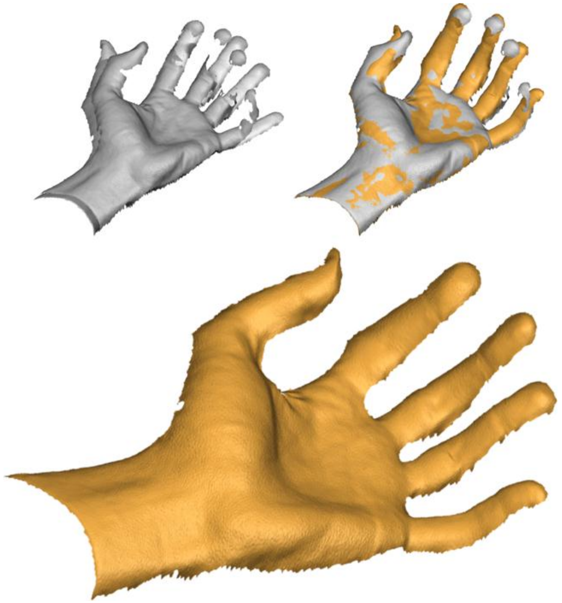
Deformable Alignment



- Segment model into rigid elements
- Use Iterated Closest Point methods to align elements
  → energy minimization
- Solve simultaneously for model, segmentation and motion

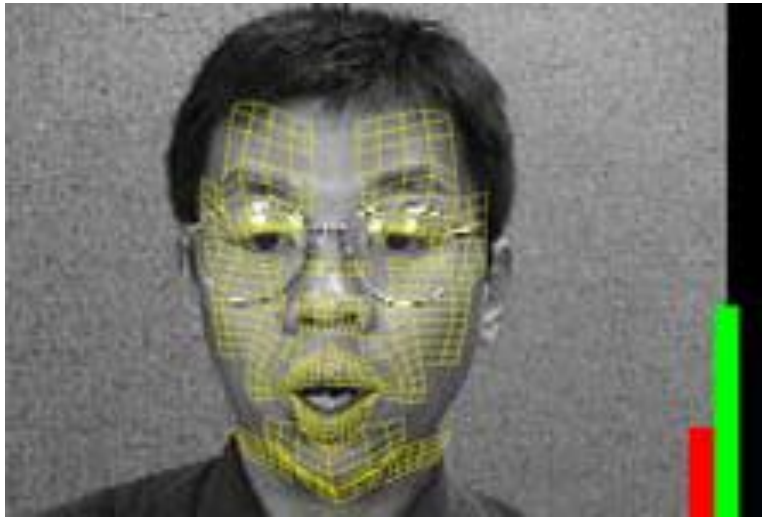


Model	# scans	# points/scan (in 1000s)	Time (mins)
bunny (simulated)	314	33.8	13
bee	2,200	20.7	51
coati	2,200	28.1	71
teapot (rigid)	2,200	27.2	68
skeleton (simulated)	100	55.9	11
hand	100	40.1	17



AvaScholar Student

- Uses ordinary student PC webcams
- Feature vectors based on real-time segmentation & patch fitting
  - Use classification by nearest-neighbor (NN), tree-augmented naïve (TAN) Bayes, hidden Markov model (HMM), Gaussian mixture model (GMM)

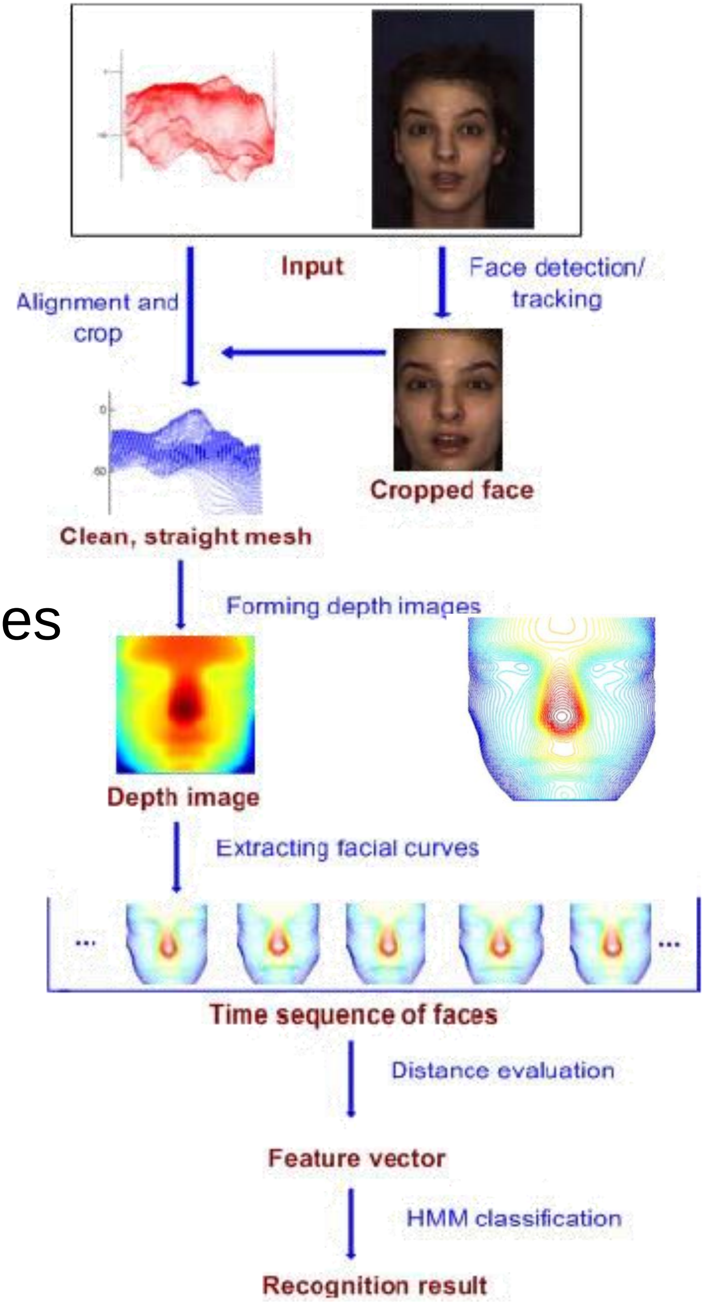


Collecting Soft Biometrics

- Age Estimation
- Gender Estimation
  Zhou et al., “Image Classification Using Super-Vector Coding of Local Image Descriptors. ECCV 2010
- Expression Recognition
  Le et al., “Expression recognition from 3D dynamic faces using robust spatio-temporal shape features. FG 2011
- Shrug Detection: Hough parabola transform
  Ning et al. A Realtime Shrug Detector, FGR 2006

Expression Recognition

- Six expressions: anger, disgust, happiness, fear, sadness, surprise
- 101 subjects, 100 frames of 3-D video
- Face represented by iso-depth curves
- Feature vector = distances between iso-depth curves (chamfer distance)



http://i2pc.cs.illinois.edu/

Where are We Headed?

- Build a robust system for educational purposes
- Apply it to other environments:
  - Corporate meetings
  - Political speeches: change message on-the-fly?
- Applying computer science technologies
  - Scheduling of tasks on a heterogeneous, soft real-time platform
  - Annotations and checks for correctness

PROGRAMMING PRODUCTIVITY

Abstractions

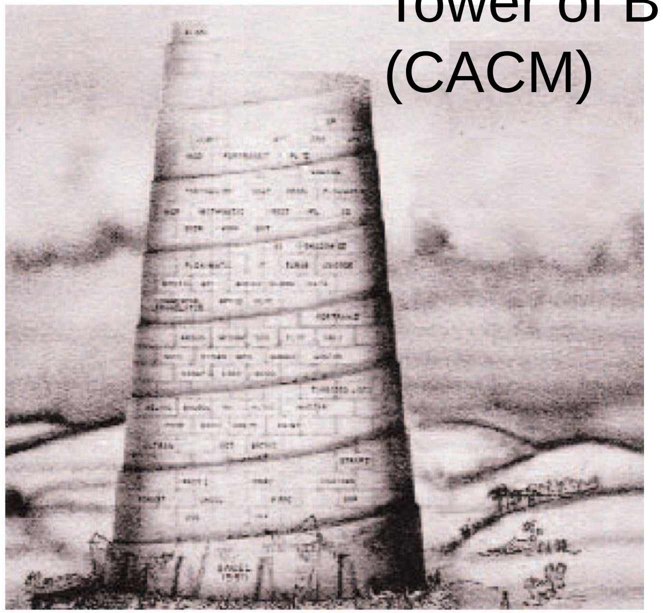
Refactoring

Determinism

Parallel Programming Abstractions

Many low level parallel notations:

- GPU/SIMD
  - CUDA
  - OpenCL
- Shared memory
  - Cray microtasking
  - OpenMP
  - BSP
  - Linda
  - Intel TBB
  - Cilk (Intel)
  - Java threads
  - Pthreads
- Distributed memory
  - CSP (Occam)
  - PVM
  - MPI
  - UPC
  - Co-array Fortran

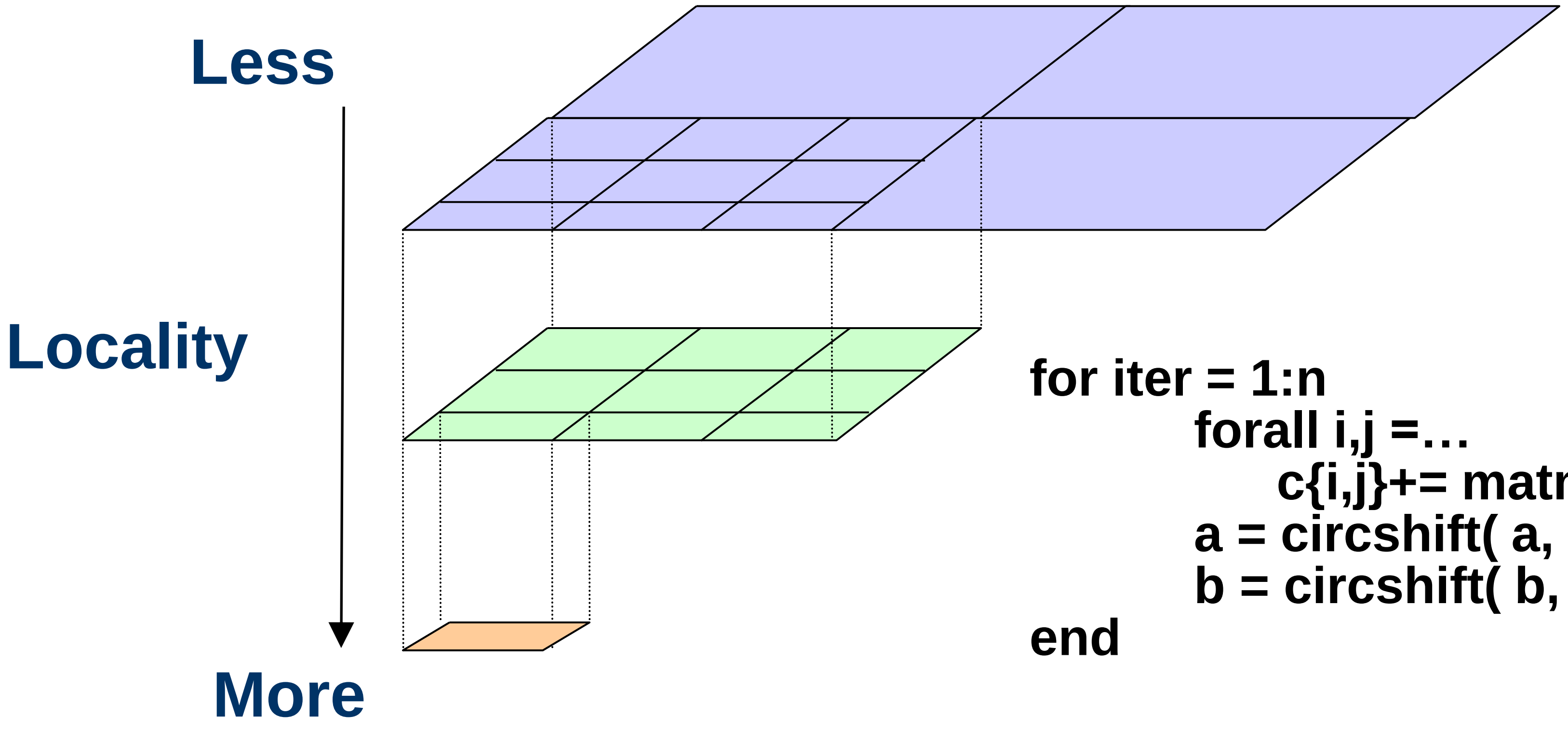


Tower of Babel (CACM)

- High-level programming abstractions:
  - Facilitate the development of efficient parallel programs
  - Enable portability across classes of machines

Hierarchically Tiled Arrays

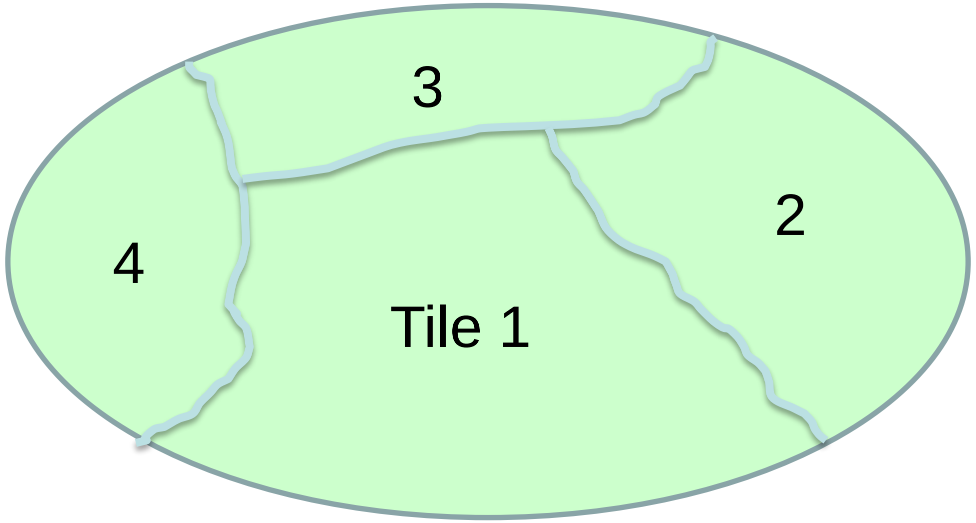
- Tile structure gives programmer control over locality, granularity of parallelism, and communication [PPoPP’06, PPoPP ’08]



```
for iter = 1:n
        forall i,j =…
            c{i,j}+= matmul(a{i,j},b{i,j});
        a = circshift( a, [0, -1] );
        b = circshift( b, [-1, 0] );
end
```

Abstractions for Irregular Computations

- Notations to tile sets, trees, graphs [HotPar’09]
- Result: highly efficient & readable programs



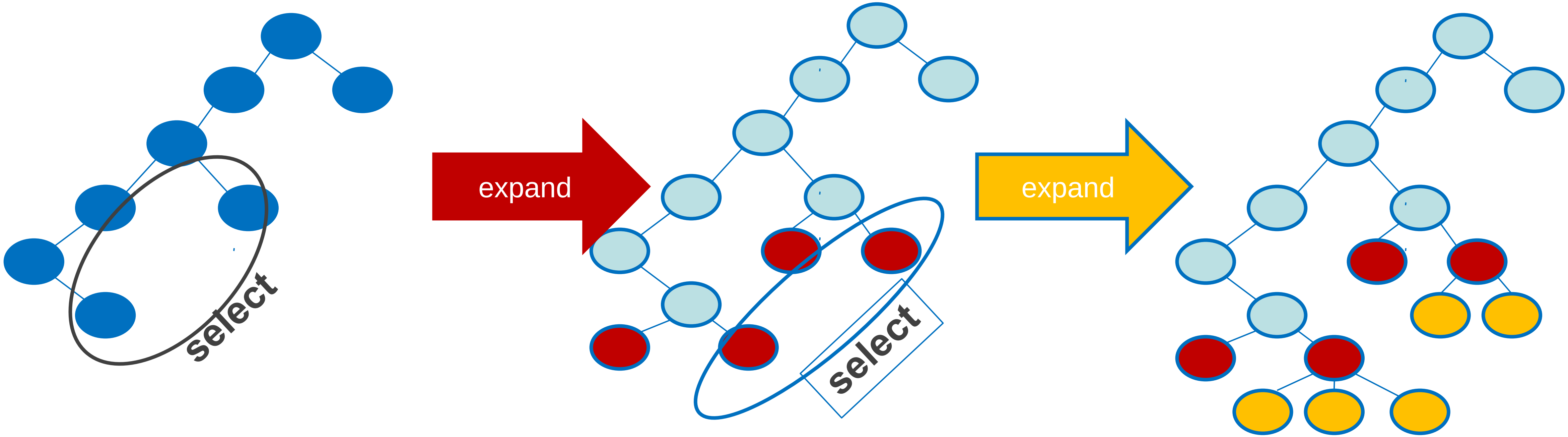
Application of Tiled Sets in a Tree Search

```
TiledSet work_list[ # of Tiles ]

while ( work_list not empty )
    n = SELECT( work_list )
    If ( n contains GOAL ) break
    work_list = work_list – n
    successors = expand( n )
    update( work_list, successors )
```

Code looks sequential
Operators can be parallel

Application of Tiled Sets in a Tree Search



http://i2pc.cs.illinois.edu/

Interactive Refactoring

- Tools for incremental, interactive parallelization of sequential code
- Approach:
  - User focus on key decisions, tool does the tedious code analysis and rewriting
- Refactorings for thread-safety
  - Make class immutable [ICSE'11]
  - Convert to Atomic* classes [ICSE'09]
  - Use concurrent collections [ICSE'09]
  - Infer region annotations [ASE'09]
  - Privatize shared variables
- Refactorings integrated in official release of Eclipse 4.2.1 this summer

Deterministic Parallel Java (DPJ)

- Nondeterminism makes parallel programming harder
  - Concurrent task interleavings
  - Task schedules
- Results in concurrency bugs (races, deadlocks, …)
- This isn’t necessary: Many parallel algorithms are deterministic
  - Same input always produces same (visible) result

Language guarantees determinism as default case

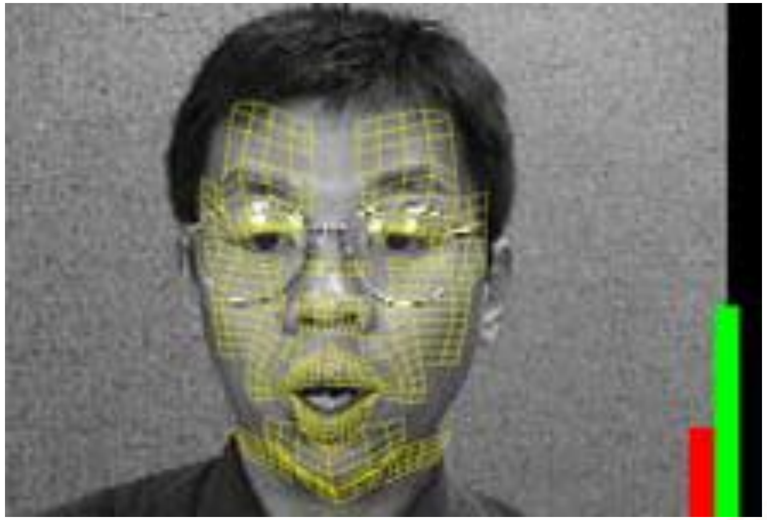
Nondeterminism is explicit and controlled

DPJ: Overview

- Uses Fork-Join model of task parallelism (cobegin, foreach)
- Annotates code with regions and effects to statically describe the dynamic effects of a program.
  - Region annotations group mem locations into hierarchy of sets
  - Loads and stores have effects on regions.
- Methods are annotated with effect summaries, which are checked by the compiler.
  - Allows modular checking of the code
- Non-interference of tasks derived from region disjointness

Parallelizing AvaScholar with DPJ Annotations

- DPJ annotations to detect and remove bugs
  - Variables and fields assigned to regions
  - Read and write effects tracked per region
  - Each method annotated with effect summary
  - Effect summary checked by DPJ



Example of FaceExpr Module

```
double likelihoodClass[] =
    new double[numberLabels];

foreach (int j in 0, numberLabels) {
    record[0] = j;
    double prod = (pfs[0]).get_value(j);
    for (int ii=1; ii<pfs.length; ii++) {
        prod *= (pfs[ii]).evaluate(pvs, record);
    }
    likelihoodClass[j] = prod;
}
```

0th elem. changed in parallel

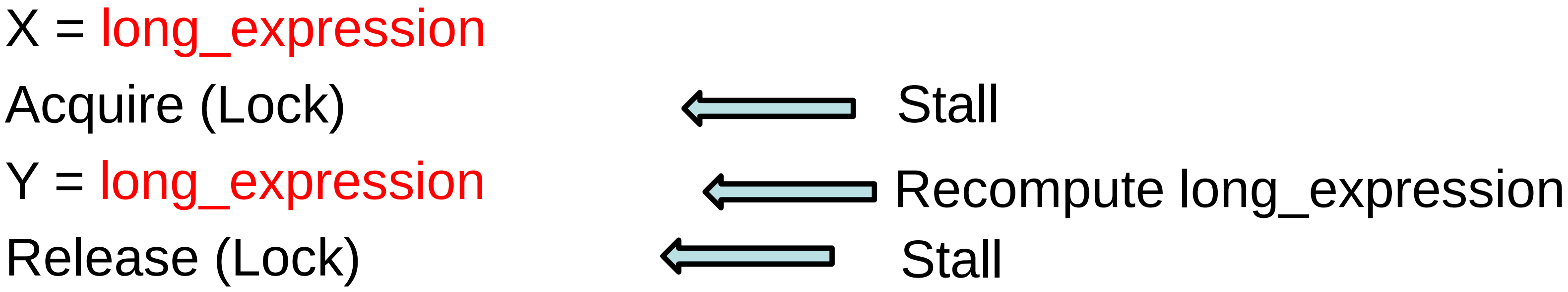
get_value, evaluate unknown effects:
could “write *”

Likelihood output could be in overlapping area

ARCHITECTURES

Architecture for Programmability: Bulk Multicore

- Current processors commit in-order one instruction at a time
  - Disables many “unsafe” hardware & compiler optimizations
  - Most of time, not needed



X = long_expression

Acquire (Lock) ← Stall

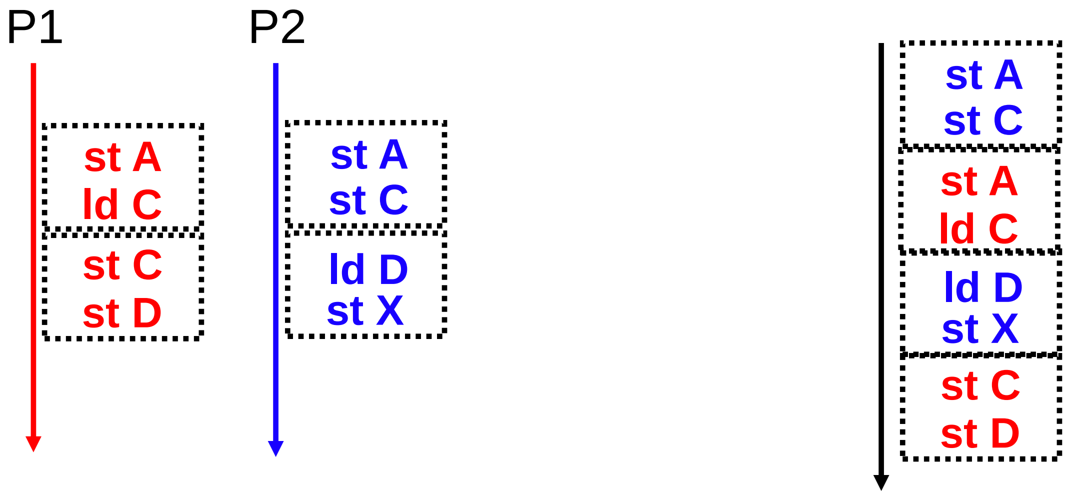
Y = long_expression ← Recompute long_expression

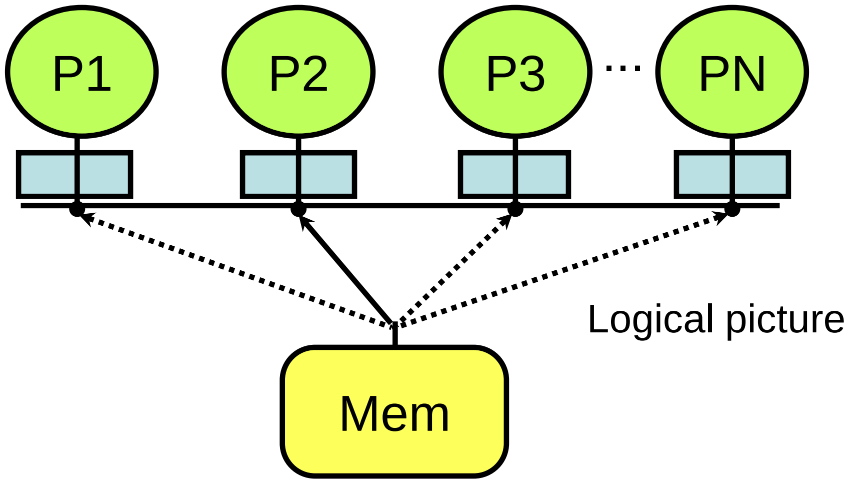
Release (Lock) ← Stall

Bulk Multicore: a Continuous-Block Architecture

[CommACM-09]

- Processors continuously commit chunks of instructions at a time
- Each chunk is executed atomically and in isolation
- Protocol ensures a total order of chunk commits

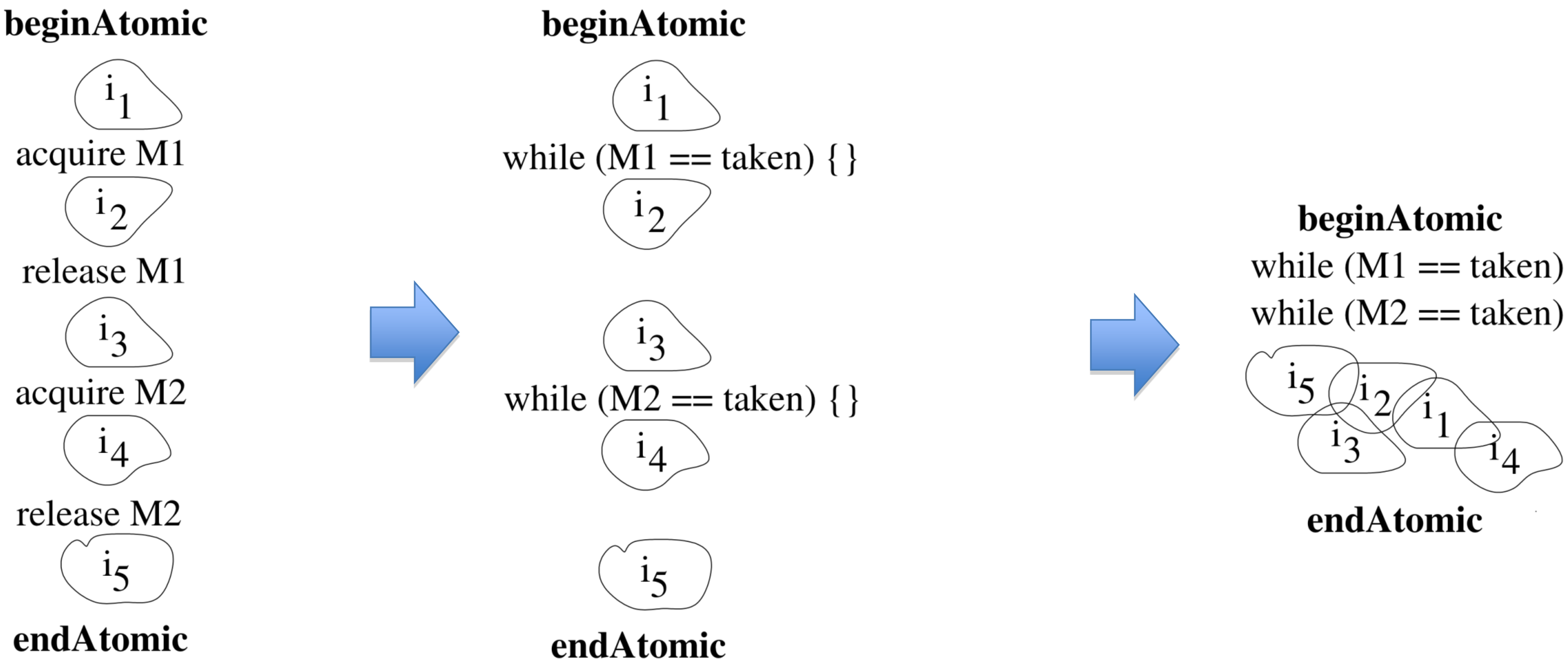




High performance and low power:

- Instructions are reordered by HW inside chunks
- Compiler can aggressively optimize the code inside chunk

BulkCompiler Transformations [MICRO-09]



beginAtomic
i_1
acquire M1
i_2
release M1
i_3
acquire M2
i_4
release M2
i_5
endAtomic

beginAtomic
i_1
while (M1 == taken) { }
i_2
i_3
while (M2 == taken) { }
i_4
i_5
endAtomic

beginAtomic
while (M1 == taken) { }
while (M2 == taken) { }
i_5 i_2 i_1 i_3 i_4
endAtomic

- BulkCompiler outperforms conventional multicore with the relaxed Java Memory Model by 37% (avg 10 Java apps)

DeNovo: Hardware for Disciplined Parallelism

- Idea: Rethink hardware as driven by disciplined software
    - Discipline = structured parallel control + explicit effects
    - Better simplicity, performance, power
- Reduce complexity of cache coherence:
    - Subtle races and numerous transient states in the protocol
    - Hard to extend for optimizations
- Eliminate performance and power inefficiencies
    - Invalidation, ack messages
    - Indirection through directory
    - False sharing (cache-line based coherence)
    - Cache pollution (cache-line based allocation)

What Have We Accomplished?

- Technical insights and advances that go into nearly 100 papers
  - Use of parallelism for compelling tele-immersive env.
  - Improvements in programming productivity (abstractions, refactoring, determinism,etc)
  - Developed many tools for parallel code test & debugging
  - Architectures that scale to 1K cores and are programmable

- Lot of interaction with sponsor researchers

What Have We Accomplished?

- Educated 800+ participants:
  - 4 Summer Schools at Illinois and 1 at Singapore
  - 3 one-week training courses at Boeing
- Tutorials at OOPSLA and ICSM and other workshops
- Encyclopedia of Parallel Computing
- Revamping the undergraduate Computer Science Curriculum for parallelism at U of Illinois
- Refactorings integrated in official release of Eclipse 4.2.1 at the end of this summer
- Graduated many students

Moving Forward

- Goal: Make parallel programming synomymous with programming
  - Enable consumer apps with high-perf. computing on mobile devices

- Focus areas:
  - The browser as the driving application
  - Ecosystem for advanced visual computing
  - Very low power and energy substrates

Universal Parallel Computing Research Center Illinois

Making parallel programming synonymous with programming

Josep Torrellas,
Vikram Adve, Sarita Adve, Danny Dig, Minh Do, Maria Garzaran, John Hart, Thomas Huang, Wen-Mei Hwu, Ralph Johnson , Laxmikant Kale, Sam King, Darko Marinov, Klara Nahrstedt,
David Padua, Madhusudan Parthasarathy, Sanjay Patel,
Marc Snir, Craig Zilles,

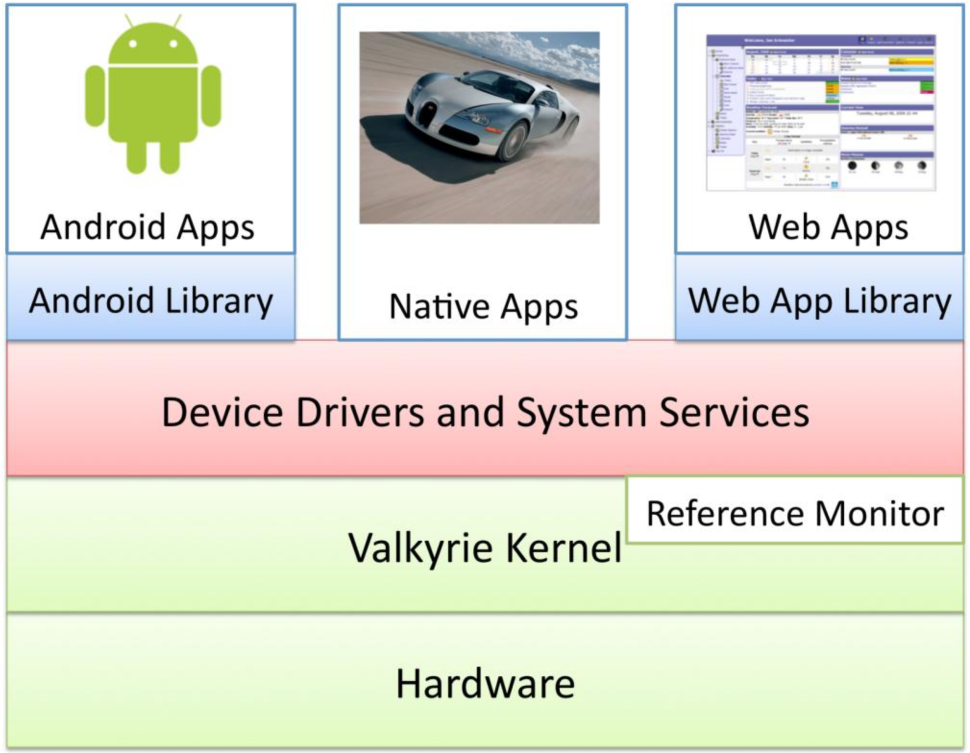
ILLINOIS

UPCRC Illinois
Universal Parallel Computing Research Center

I2PC

Valkyrie OS and Browser: simple and secure

- A new mobile OS and a new mobile browser
- Faster and more secure

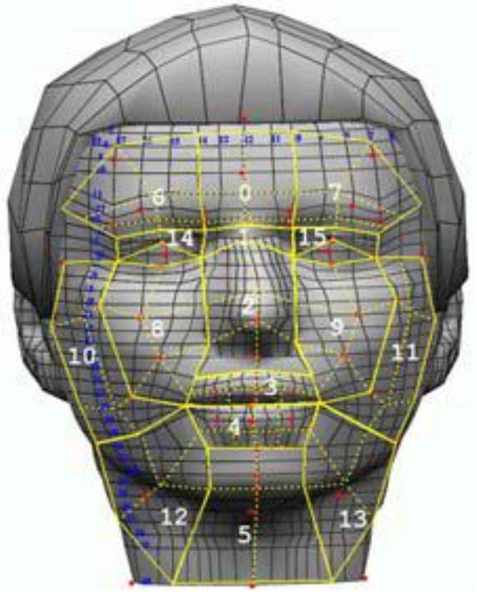


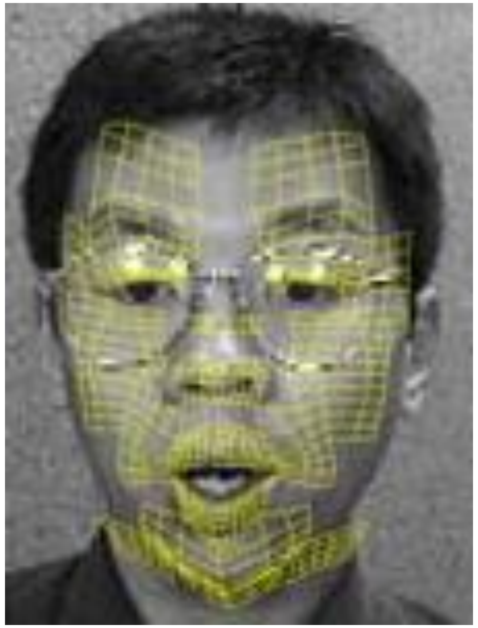
Underlying technology

- Type safe kernel written in C
  - Compiled using LLVM
- Use formal methods in implementation
  - Shows that provable security invariants are possible
- Entire OS is less than 10k LOC
  - Exports subset of the Linux syscall interface
- Processing web pages with static and dynamic parallelism
- Hardware architecture for energy efficiency and reliability

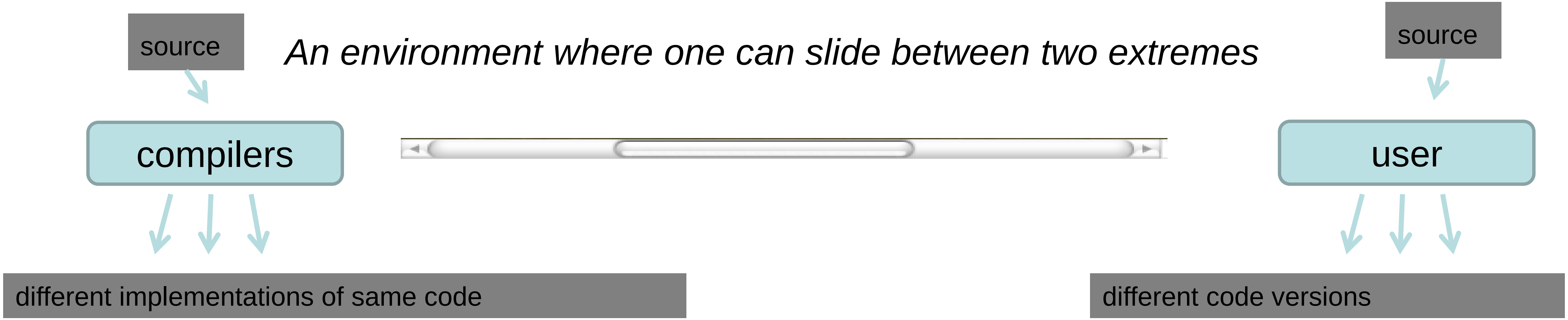
Face Tracking

- Patch model of face is fit to image feature points
- Patch model is deformed according to a robust multiresolution optical flow extracted from image video sequence





Where are We Headed?
A Refactoring Approach to Code Evolution



source

An environment where one can slide between two extremes

source

compilers

user

different implementations of same code

different code versions

- Maintain different versions as one object:
  - Equivalence established formally, via testing, runtime checks...
  - System “knows” what changes occurred, which changes are safe, can undo changes, can reconcile different versions...
  - System can display different views of “same” code
  - System can ingest performance data, can trigger automatic transformations, can trigger autotuning, *and can show the results to the user*.

Valkyrie OS and browser

- A new mobile OS
  - New architecture to achieve high security assurance
- A new mobile browser
  - Novel approach to speed up mobile web browsing
- Faster and more secure
  - Improved web surfing experience
  - Guaranteed end user security and privacy

Outreach Actions

- Yearly summer course on parallelism at UIUC
  - One in Singapore
- Various workshops and technical meetings
- Encyclopedia of Parallel Processing
- Revamping the Illinois undergraduate Computer Science Curriculum for paralrellsims
- Many technocal papers, invited talks
- Many studnets graduated
- Lots of collaboration with Intel and other industial sponsors

Educated 800+ participants:

4 Summer Schools at Illinois and 1 at Singapore

- 3 one-week training courses at Boeing

- 2 half-day tutorials at OOPSLA and ICSM

Our refactorings integrated in official release of Eclipse 4.2.1 at the end of this summer

Main Achievements Applications

- We clearly demonstrated that one needs parallelism and one can use parallelism for compelling tele-immersive environments
  - Need all forms of parallelism (messaging, coarse grain shared memory, SIMD); we learned which applies where
- Developed improved parallel algorithms and libraries to support key components of the video processing and rendering pipelines
  - Using ViVid to drive performance portability work and used the apps to drive work on DPJ

- Developed browser architecture of major import to Microsoft

Main Achievements Software

- Validated and publicized the Illinois approach for concurrency safe, deterministic by default parallel programming environments
  - Promises qualitative improvements in productivity with shared memory parallel programming
- Pushed data parallelism into new application domains
- Developed an approach for tighter user-compiler synergy for code optimization
- Developed a clear vision for the role of refactoring in parallel programming
- Developed improved techniques for parallel code testing
- Used the software work to improve the application codes

Major Achievements Hardware

- Demonstrated multi-core scalability to 1K cores.
- Developed practical mechanisms for improving scalability and programmability of current architectures
  - Use of atomic block execution for compiler optimization
  - Hardware support for race detection and deterministic replay
- Developed designs for fundamentally new multi-core architectures

- Architecture and Software were co-designed: tools and programming models are supported by architectural enhancements; better architectures are enabled by software enhancements.

Microsoft

The information herein is for informational purposes only and represents the current view of Microsoft Corporation as of the date of this presentation. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information provided after the date of this presentation. MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS PRESENTATION.