FROM SCALAR & SERIAL TO VECTOR & PARALLEL

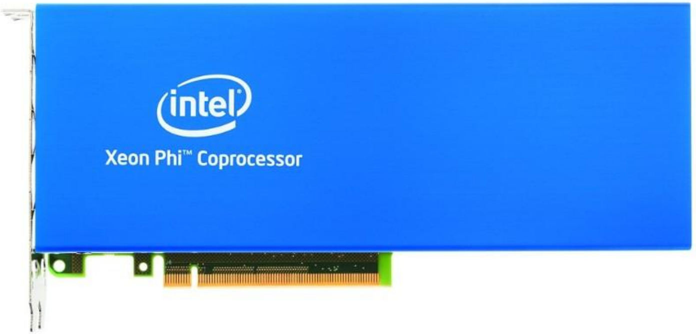
Hands-on Lab

Part 1 of 3

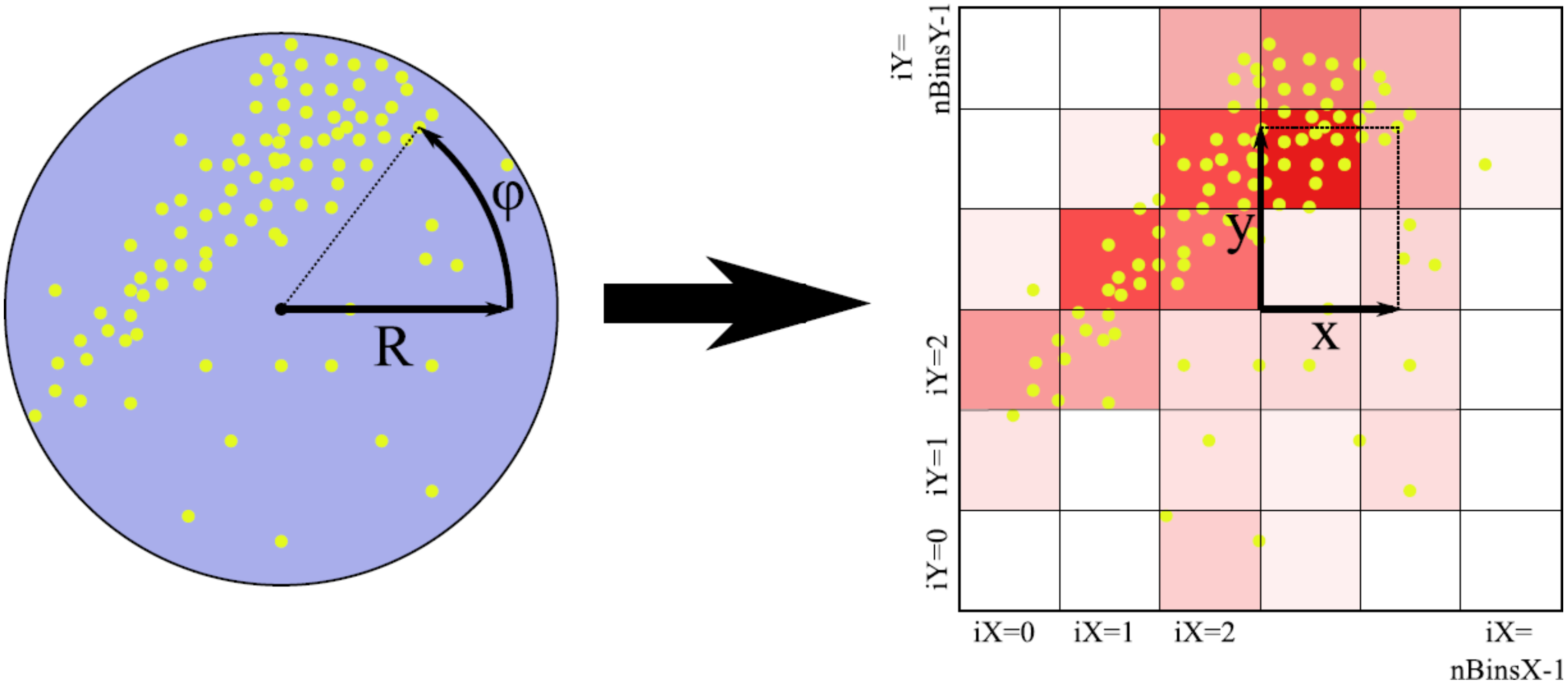
COLFAX
Customized Solutions

GOALS

- **Part 1: see multi-threading on Intel architecture in action**
- Part 2: learn to vectorize & future-proof vectorization
- Part 3: experiment with what it takes to make the memory subsystem happy

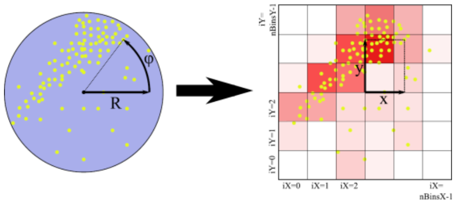


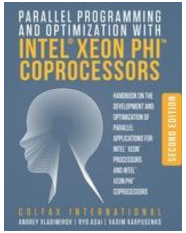
EXAMPLE PROBLEM: BINNING



SERVER FOR EXERCISES

- Instructions at **uni.colfax-intl.com/cdt**
- Find code of lab + exercises from **xeonphi.com/book**





PARALLEL PROGRAMMING AND OPTIMIZATION WITH INTEL® XEON PHI™ COPROCESSORS

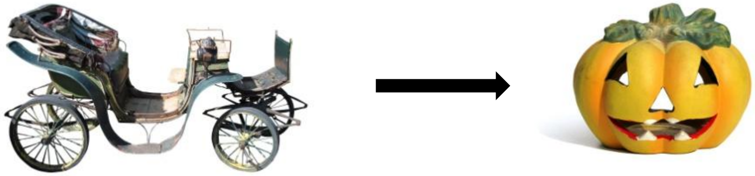
HANDBOOK ON THE DEVELOPMENT AND OPTIMIZATION OF PARALLEL APPLICATIONS FOR INTEL® XEON® PROCESSORS AND INTEL® XEON PHI™ COPROCESSORS

SECOND EDITION

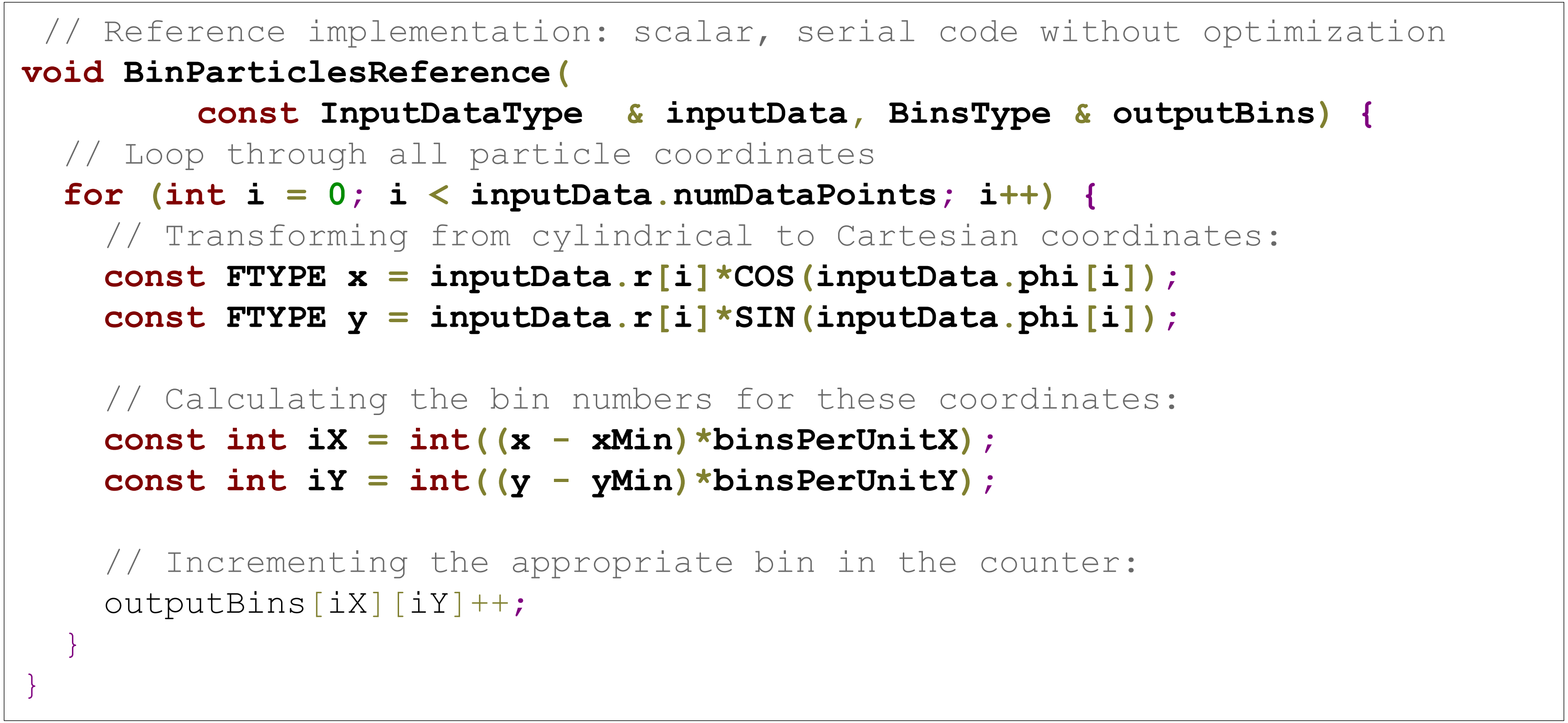
COLFAX INTERNATIONAL

ANDREY VLADIMIROV | RYO ASAI | VADIM KARPUSENKO

- Enjoy remote access until midnight



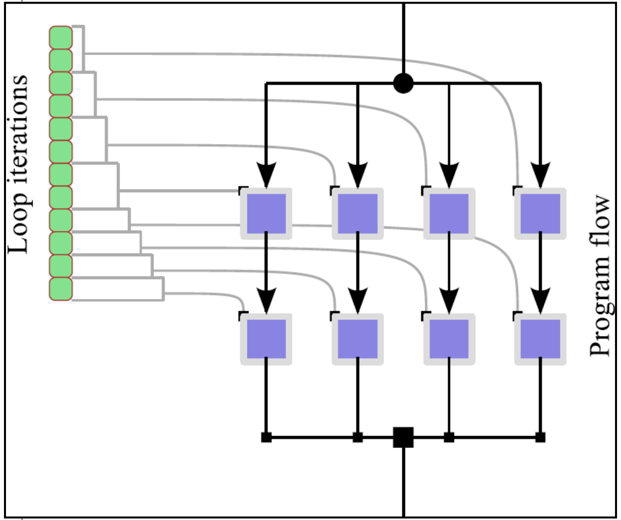
INITIAL APPROACH



```
 // Reference implementation: scalar, serial code without optimization
void BinParticlesReference(
         const InputDataType  & inputData, BinsType & outputBins) {
  // Loop through all particle coordinates
  for (int i = 0; i < inputData.numDataPoints; i++) {
    // Transforming from cylindrical to Cartesian coordinates:
    const FTYPE x = inputData.r[i]*COS(inputData.phi[i]);
    const FTYPE y = inputData.r[i]*SIN(inputData.phi[i]);

    // Calculating the bin numbers for these coordinates:
    const int iX = int((x - xMin)*binsPerUnitX);
    const int iY = int((y - yMin)*binsPerUnitY);

    // Incrementing the appropriate bin in the counter:
    outputBins[iX][iY]++;
  }
}
```

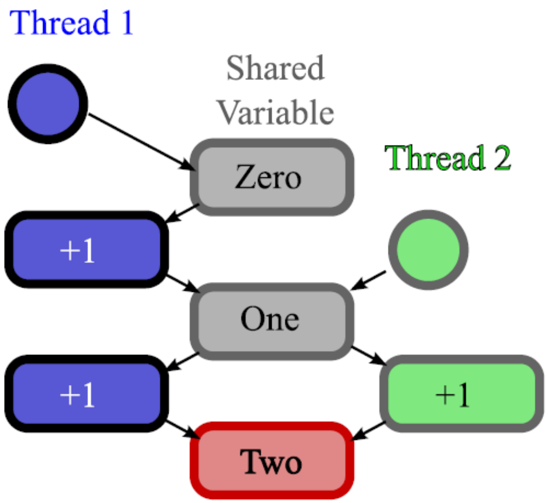
OPENMP THREADS

```
 // Parallel, but INCORRECT implementation
void BinParticlesReference(
         const InputDataType  & inputData,
         BinsType & outputBins) {
  // Distribute loop iterations across threads
#pragma omp parallel for
  for (int i = 0; i < inputData.numDataPoints; i++)
    {

      // …
      // Incrementing the appropriate
      // bin in the counter:
      outputBins[iX][iY]++;
    }
}
```

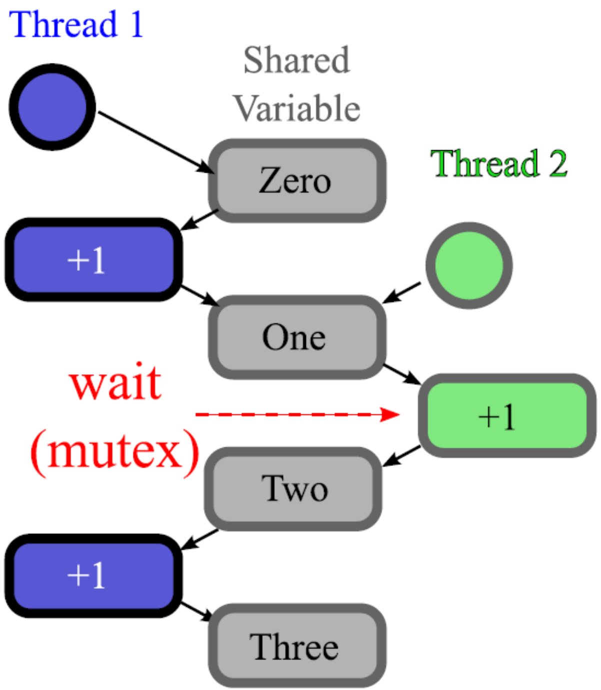


Software threads utilize logical processors in an Intel Xeon or an Intel Xeon Phi processor

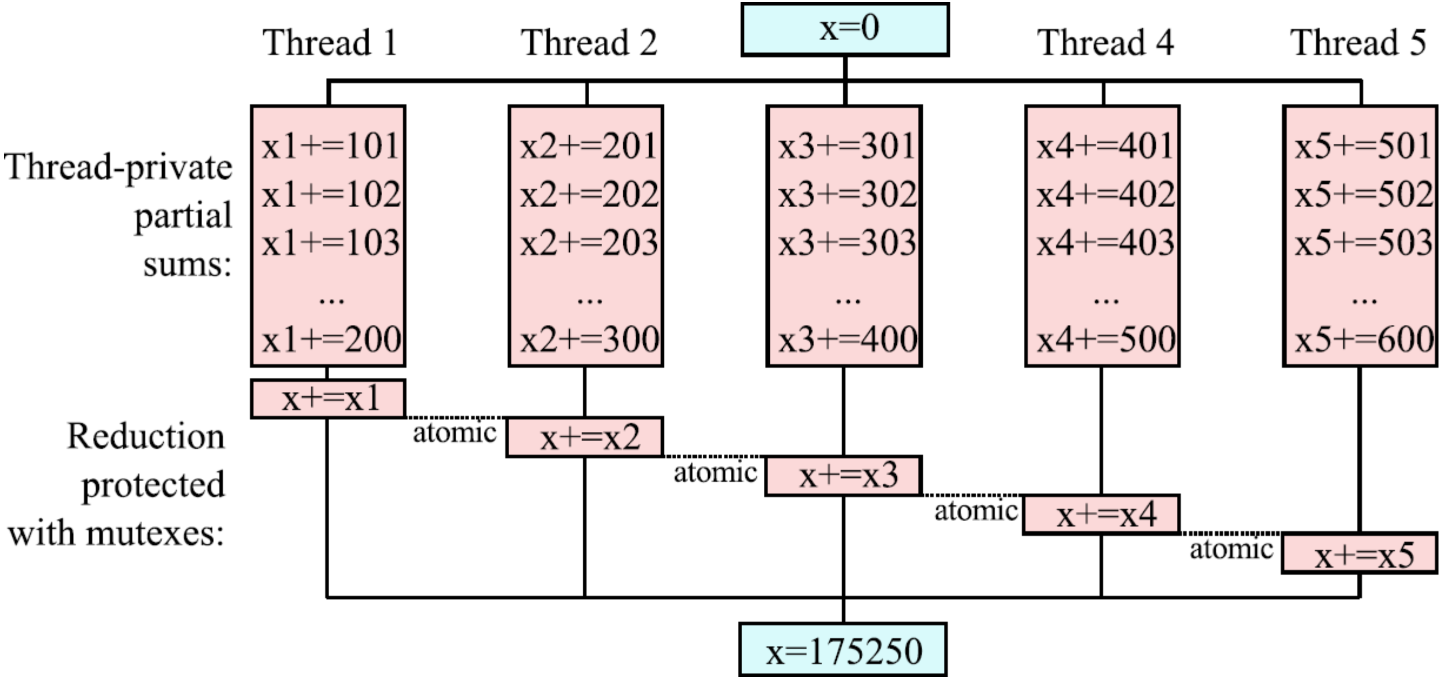
DATA RACES



Race Condition!



PARALLEL REDUCTION

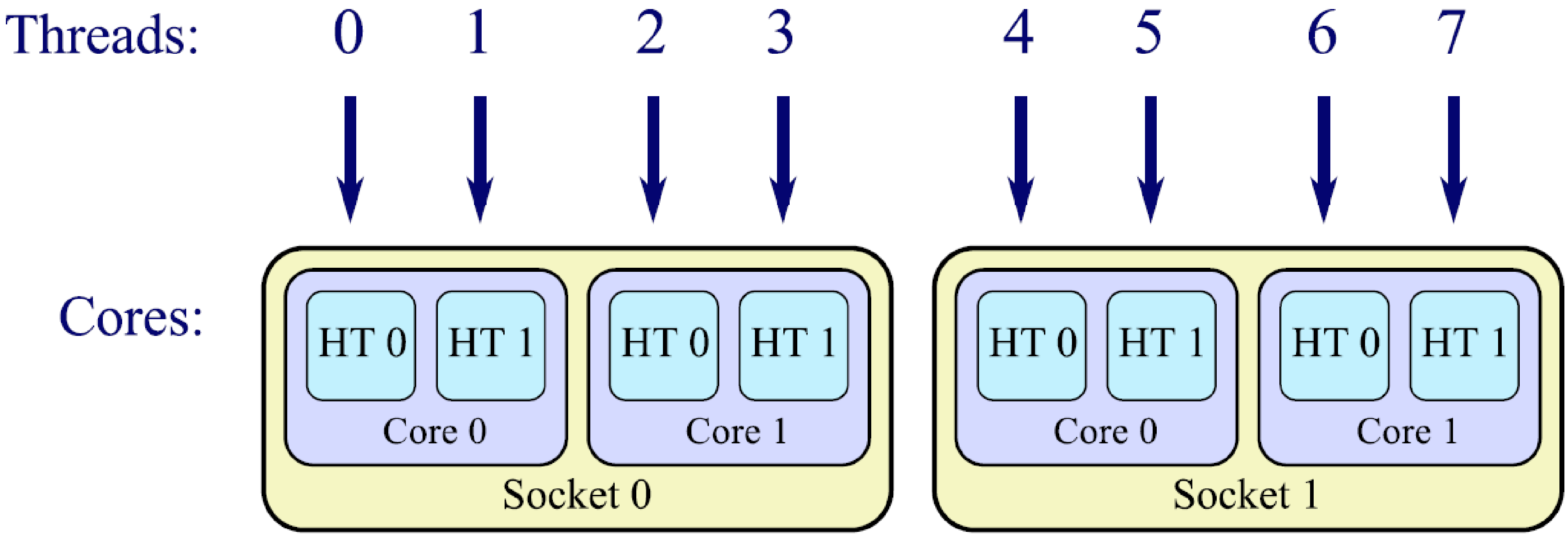


Key: using thread-private containers for partial sums; mutexes only after the parallel loop

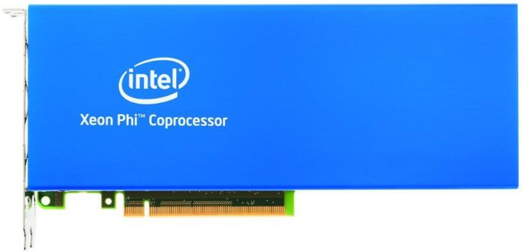
OPTIMIZED VERSION

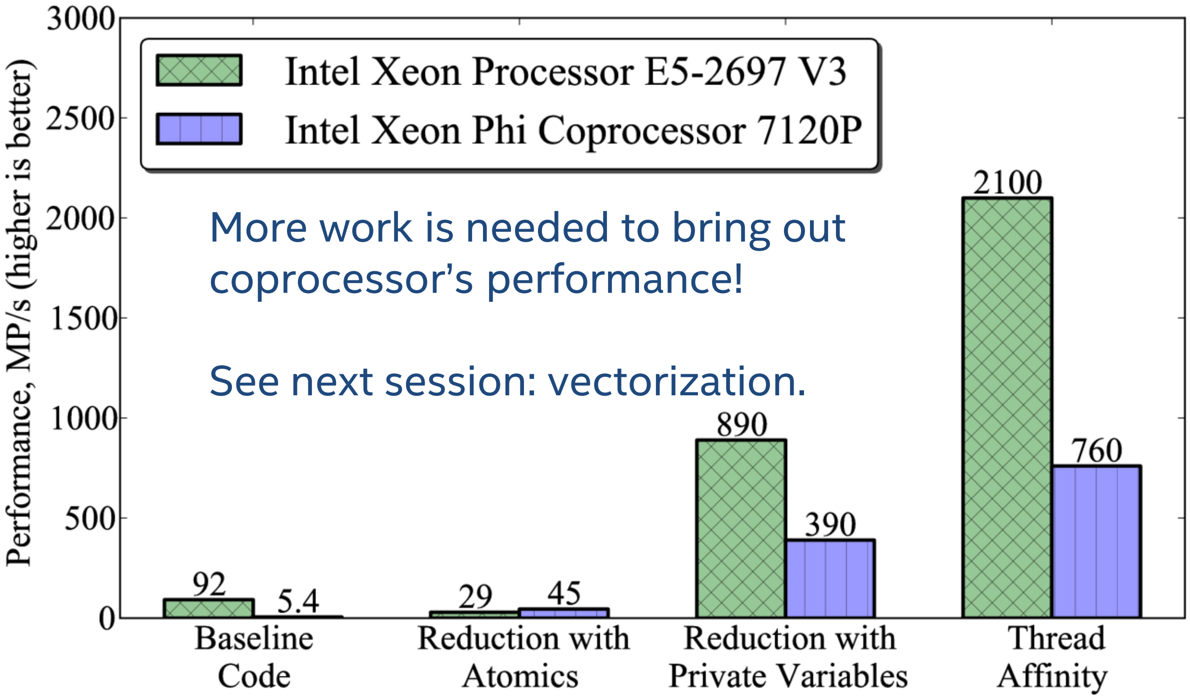
```
void BinParticles_2(const InputDataType& inputData, BinsType& outputBins){
#pragma omp parallel
  { BinsType threadPrivateBins; // Declare thread-private containers
    for (int i = 0; i < nBinsX; i++)
      for (int j = 0; j < nBinsY; j++)
        threadPrivateBins[i][j] = 0;
#pragma omp for
    for (int i = 0; i < inputData.numDataPoints; i++) {
      // …transforming from cylindrical to Cartesian coordinates:
      // …calculating the bin numbers for these coordinates:
      // Incrementing the appropriate bin in the thread-private counter:
      threadPrivateBins[iX][iY]++;
    }
    for(int i = 0; i < nBinsX; i++) // Reduction outside the parallel loop
      for(int j = 0; j < nBinsY; j++)
#pragma omp atomic
        outputBins[i][j] += threadPrivateBins[i][j];
  } }
```

THREAD AFFINITY



PERFORMANCE RESULTS





More work is needed to bring out coprocessor's performance!

See next session: vectorization.

NEXT SESSION

- Part 2: vectorization – another level of parallelism
- Part 3: making the memory subsystem happy

By the way:
download this tutorial
colfaxresearch.com

Log In/Out or Register

COLFAX RESEARCH
CONTRIBUTING TO INNOVATIONS IN COMPUTING

/ READ WATCH LEARN CONNECT JOIN

Optimization Techniques for the Intel MIC Architecture. Part 3 of 3: False Sharing and Padding

To search, type and hit enter

Popular

The Hands-On Workshop (HOW) Series

Archive: Hands-On Workshop, October 2015

Syllabus: Hands-On Workshop, November 2015

Parallel Programming Book

Introduction to parallel programming, deep discussion of optimization techniques, exercises. © 2015, Colfax International. 508 pages.

Research and Educational Publications

Introduction to Intel DAAL, Part 1: Polynomial Regression with Batch Mode Computation

Optimization Techniques for the Intel MIC Architecture. Part 3 of 3: False Sharing and Padding

Software Developer's Introduction to the HGST Ultrastar Archive Ha10 SMR Drives

Optimization Techniques for the Intel MIC Architecture. Part 2 of 3: Strip-Mining for Vectorization

Optimization Techniques for the Intel MIC Architecture. Part 1 of 3: Multi-Threading and Parallel Reduction

Performance to Power and Performance to Cost Ratios with Intel Xeon Phi Coprocessors (and why 1x Acceleration May Be Enough)

Featured Video

CDT V01 - Episode 1.1 - Introduction

Events

Syllabus: Hands-On Workshop, November 2015

Archive: Hands-On Workshop, October 2015

Delivering a hands-on tutorial at IDF

Presentations

The Hands-On Workshop (HOW) Series

Scientific Computing with Intel Xeon Phi Coprocessors

Services

Face-to-face training on parallel programming, Intel architecture

Web-based hands-on training with remote access

Video courses on parallel programming,