TEXAS ADVANCED COMPUTING CENTER
WWW.TACC.UTEXAS.EDU

TEXAS
The University of Texas at Austin

ICAT: Using an Interactive Tool to Adapt Codes to KNL

IXPUG Fall Meeting 2017
Wednesday, September 27th

PRESENTED BY:

Lars Koesterke

Email: lars@tacc.utexas.edu

Ritu Arora

Email: rauta@tacc.utexas.edu

Goals

We want to teach HPC concepts without the burden of teaching syntax first

Tools for (semi)automatic code modification allow for this approach

1. Quick introduction to the topic
2. Presentation of example code
3. Semi-automatic code modification (with our tool)
4. Exploring modified code
    1. Running modified code to assess benefits (usually speed-up)
    2. Inspect modified code
5. Learn syntax elements from modified code

Teaching with IPT and ICAT

IPT: OpenMP

First example

- Spawning threads: parallel region
- Parallel execution: work-sharing
- Race condition: private variables
- Race condition: reduction variables

Second example

- Race condition: Write-after-read (WAR) conflict
- Resolution through code modification

Students are exposed to concepts

Syntax is important, but is better learned later from a book; no time 'wasted' in class

ICAT: High-Bandwith Memory

Goal

- Converting a 'malloc' or allocate' into an HM memory allocation
- Including a decision tree

Example

- Write-after-read (WAR) conflict
- Resolution through code modification

Students are exposed to concepts

Syntax is important, but is better learned later from a book; no time 'wasted' in class

Common Questions

Related to Innovative Hardware and Supporting Software Stack

1. How can I adapt my application to take advantage of the manycore processors and multiple levels of memory hierarchies?
2. Which is the best memory mode for running an application on a KNL processor?
3. What are the band-width critical portions of an application data?
4. Which cluster mode is the best for running an application on a KNL processor?
5. How much effort is required to analyze and adapt an application for the KNL processors?

HPC is Constantly Evolving

The complexity in architecture is increasing. We highlight 2 trends here:

1. More cores, with more threads (Hyperthreads)
   - Pure MPI is not enough. Threads are needed to
     1. Make use of the Hyperthreads
     2. Reduce MPI tasks and MPI communication
     3. Allow for more memory per MPI tasks
2. Knights Landing (KNL) with additional memory layers
   - High-bandwidth memory (MCDRAM)
   - May need additional memory management in code

The users are forced to constantly adapt their code. We are developing tools to help users to:

1. Parallelize code with MPI and OpenMP
   - MPI and OpenMP for new users
   - Upgrade to a hybrid setup (MPI + OpenMP)
   - Memory management; Compile and run code optimally (KNL-specific)

Stampede 2

We are in 'Phase 1'
The KNL's will provide more than 50% of the available SUs

Phase 1: 4200 KNL nodes

- 68 cores
- Up to 272 Hyperthreads
- 96 GB DDR4 memory
- 16 GB MCDRAM
- Most in Cache Mode (tag directories in quadrants)
- Some in Flat Mode (tag directories in quadrants)
- Special queues for other configurations

Phase 2: Skylake dual-socket nodes

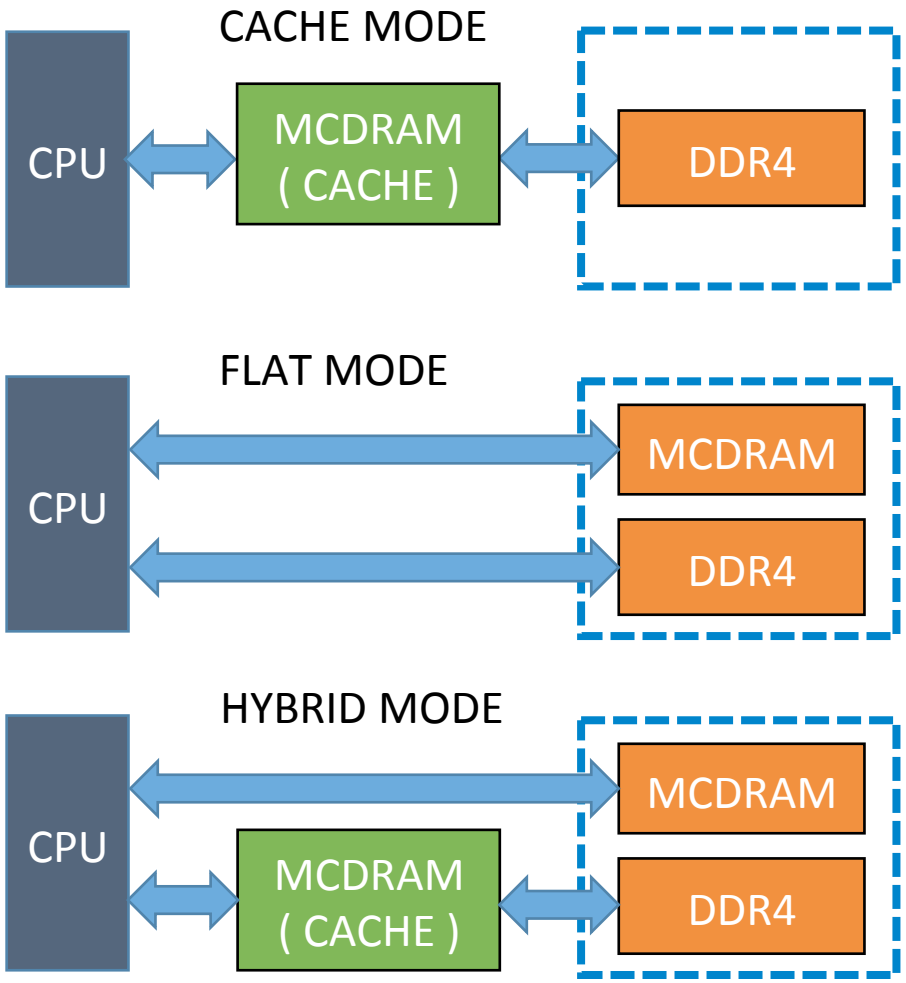
- 40+ cores
- 80+ Hyperthreads

Fast interconnect (Fat-tree)

Fast file systems

Memory Hierarchy and Modes

- Two main memory types
  - DDR4
  - MCDRAM
- Three memory modes
  - Cache
  - Flat
  - Hybrid
- Hybrid mode
  - Three choices
  - 25% / 50% / 75%
  - 4GB / 8 GB / 12GB



Memory Modes

Memory footprint	Flat	Cache
< 16GB	Optimal	Couple of percent lower
>16 GB: many arrays	May work well. May require code modifications (user identifies which arrays to store in MCDRAM)	May work well
>16 GB: few large arrays	Does not work, if no array fits in the MCDRAM	May work well

Using the 'Flat' mode may require code modifications

Cluster Modes

- Each tile tracks an assigned range of memory addresses on behalf of all cores by maintaining a data structure called tag directory
- To maintain cache-coherency, tile-to-tile, and tile-to-memory communication is required
- Cores that read or modify data must communicate with the tiles that manage the memory associated with that data
- Similarly, when cores need data from main memory, the tile(s) that manage the associated addresses will communicate with the memory controllers on behalf of those cores
- **The KNL can do this in the multiple cluster modes: all-to-all, quadrant, and sub-NUMA-4 (or 2)**

More information on clustering modes: https://portal.tacc.utexas.edu/user-guides/stampede#cluster-modes

Why are Tools helpful?

How we all cope with new HPC concepts

1. Concept of parallel execution or memory management
2. Specific parallel paradigm:
    1. Shared vs. distributed memory - OpenMP *vs.* MPI
    2. When to manually modify the code for optimal memory placement
3. Syntax, and good coding practices
4. Actual code modifications

Our tools help with syntax and the actual code modifications

1. Select OpenMP or MPI and a simple toy code
2. Modify the code interactively
3. Test the modified code
4. Inspect code to learn syntax and good coding practices

General Philosophy

We provide a tool and a variety of simple code examples

Alternatively the user may provide code

Profiling of code

Vtune and/or perf

→ list of routines, list of memory objects, etc.

Interaction with user

Identify what (parallelization, vectorization, or memory management), where (routines, loops, allocation calls)

Code modification using bash scripts and ROSE source-to-source compiler

Final code ready for compilation, testing, and inspection.

Extra for KNL: Recommendation for configuration modes

Code Adaptation Using ICAT (1)

Directing array allocation towards MCDRAM

Interactive tool

Works with example code or user code

Example shown in C and Fortran

Tool identifies statements like these:

```
real(8), dimension(:,:), allocatable, :: x
…
allocate (x(n,m))
```

```
double *x;
…
x = (double*)malloc(sizeof(double)*n*m)
```

Tool Development
One example shown here

Modified code, pseudo code shown:

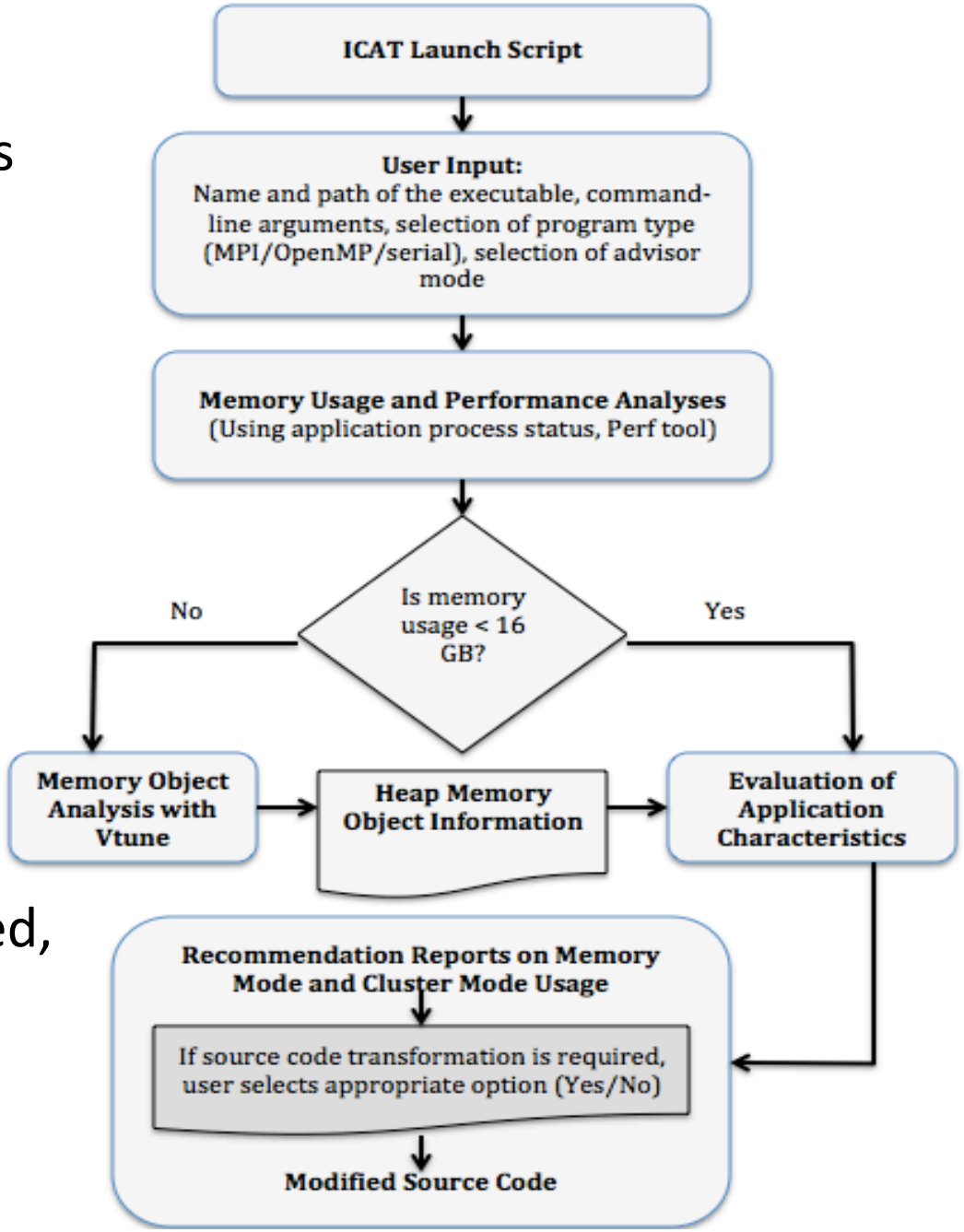
```
    size_gb = 8 * n * m / (1024*1024*1024)

    if (size_gb <= size_of_mcdram & mcdram_is_available)
Fortran:
!DEC$ ATTRIBUTES FASTMEM :: x
      allocate (x(n,m)) or
C:
      x = (double*)hbw_malloc(sizeof(double)*n*m
      if (fail)
        'normal allocate or malloc' (Reason: failure)
      fi
    else
      'normal allocate or malloc' (Reason: size or availability)
    fi
```

- Users can easily test without 'knowing' all the syntax
- Modified code ready for 'cut-and-paste'

Interactive Code Adaptation Tool (ICAT) - Overview

- Command-line tool that helps in decision-making, and code modernization and migration
- Runs the code, collects hardware counter data, determines some characteristics of applications' (e.g., L1 cache misses)
- Presents reports and if needed, modifies source code



ICAT in Action (1): Launching ICAT

```
c455-022.stampede2(6)$  bash -i ./src/icat.sh

------------------------------------------------------------------
-------- Welcome to ICAT :: Interactive Code Adaptation Tool --------
------------------------------------------------------------------

hello
/scratch/01698/rauta/testICAT
/scratch/01698/rauta/testICAT/src/
bye
Step 1
Purpose          : Acknowledge usage of compiler option '-h'
Question         : Please acknowledge that you have compiled the code with the '-g' option

                   Answer with y/n (y is the default) :: y
                   You have answered with             :: y

Step 2
Purpose          : Provide the name of the executable, the path, and optionally the program arguments
Question         : Name of the executable?                          rose_md_omp
                   Path to the executable? You may use . (dot)      /scratch/01698/rauta/testICAT/example/
                   Command line arguments, separated by commas?     2,2000,2000,0.01


Step 3
Purpose          : Select advice topic
Question         : Please select from one of these options

Option           : Advice                  Description
-------------------------------------------------
1                : Memory mode             Exploit memory hierarchies
2                : Cluster mode            Exploit clustering of cores
3                : Vectorization mode      Enable vector instructions
4                : Code adaptation         Assign individual arrays to different memory types
5                : Memory optimization     Is this the AoS to SoA transformation?
6                : All                     Get all available advice at once

0                : Quit ICAT


                   Answer with a number between 0 and 6 (0 is the default)       :: 6
                   You have selected option                                      :: 6


        Option 6: All advice available
        ---------------------------------------
```

ICAT Future Work

- Adding support for advanced vectorization
  - For situations in which the static-code analysis for vectorization opportunities is inconclusive. Users may add extra hints to support explicit vectorization.
    - `#pragma novector, #pragma ivdep`
- Supporting memory optimization
  - Improve memory alignment, scalar-to-vector conversions
  - Loop-tiling, array-of-structure versus structure-of-arrays, recalculating versus accessing from memory
- Conversion: AoS to SoA

Thank You!

Ritu Arora
Lars Koesterke

We are grateful for the support received through:

- NSF Grant # 1642396
- NSF Grant # 1359304
- TACC STAR Scholars program
- Extreme Science and Engineering Discovery Environment (XSEDE) - NSF grant # ACI-105357