Report

* Goal:
  + Efficient automatic interface of OMPD to Cetus
* Motivation:
  + Allow existing OMPD to scale beyond shared memory architectures
  + Allow OMPD interface to be used with distributed memory architectures

An OMPD Input

* Annotated sequential loops to indicate parallelism
* Implicit synchronization: no explicit data-thread movement

for (step=1; step<STEPS; step++){

#pragma omp parallel for

for (i = 1; i <= N; i++ )

A[ i ] = …

#pragma omp parallel for

for (i = 1; i <= N; i++ )

… = A[ i - 1 ] + A[ i + 1 ]

}

IR generation

* OmpAnalysisForOMPD.java
  + Parses OpenMP directives
  + Identifies shared variables
  + Identifies barriers
* OMPDHelper.java
  + Converts parallel loops into πfor loops
  + Identifies outer serial loops
  + Creates the control flow graph (CFG)
  + Adds range analysis information to the CFG nodes
  + A node in the CFG corresponds to a statement
  + Tags barriers
  + Determines if the program is repetitive or non-repetitive

Array Data Flow Analysis

* OMPDReachAnalysis.java
* Compute DEFlocal sets
* OMPDLiveAnalysis
* Compute USEGlobal sets
* An SPMD block represents the code region between consecutive barriers
* Patches the OMPD to Cetus
* SectionSet.java
* New array section representation:
* Set of regular section descriptors (RSD)
* Delayed symbolic evaluation is implemented (ERSD)
* Array section operations
* Advanced symbolic manipulation

Code Generation and Interface

* GenerateCode.java
* For each barrier
  + Inserts functions calls to pass DEFlocal and USEGlobal sets
  + Detect if communication can be removed at compile-time
  + Hoist up function calls for repetitive communications
  + Generate communication for reduction operations
* Insert function calls to partition loops
* Output is:
  + Interfaced version of OMPD to Cetus