LLMORE: Mapping and Optimization Framework

Michael Wolf, MIT Lincoln Laboratory

11 September 2012
Overview of Mapping and Optimization Challenges

Challenges:
• Realistic simulations of applications
• Support for diverse languages/numerical libraries
• Support for diverse devices and architectures
• LLMORE is MIT Lincoln Laboratory’s Mapping and Optimization Runtime Environment

• Parallel framework/environment for
  – Optimizing data to processor mapping for parallel applications
  – Simulating and optimizing new (and existing) architectures
  – Generating performance data (runtime, power, etc.)
  – Code generation and execution for target architectures

<table>
<thead>
<tr>
<th>Year</th>
<th>pMapper</th>
<th>SMaRT/MORE</th>
<th>LLMORE</th>
</tr>
</thead>
<tbody>
<tr>
<td>2004</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2006</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2011</td>
<td>pMapper: Matlab, dense operations, simple architecture model, executor, serial</td>
<td>SMaRT/MORE: Matlab, sparse operations (limited data size), architecture model, no executor, serial</td>
<td>LLMORE: multiple language support, sparse/dense operations, architecture model, executor, parallel, robust software</td>
</tr>
<tr>
<td>2012</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

◊ pMapper patent issued: “Method and apparatus performing automatic mapping for multiprocessor system”

Three generations of mapping and optimization
Key Features of LLMORE

• Support for multiple languages and numerical libraries
• Ability to solve large problems
  – Written in C++ and runs in parallel
  – Fit larger problems into memory, reduces time to solution
• Support for dense and sparse linear algebra operations
• Production quality research software
  – Easy to use interfaces
  – Designed to support future algorithms/packages/languages

• LLMORE is NOT an autoparallelizing compiler
  – Will not generate optimized parallel code for any set of (serial or parallel) instructions
  – Data layouts optimized in context of maps
Outline

• Motivation/Overview

→• Design and Usage
  – Usage 1: Map Optimization
  – Usage 2: Performance Evaluation

• LLMORE and POEM

• Preliminary POEM Results: 2D FFT

• Next Steps and Summary
LLMORE Framework Overview

LLMORE input
- User Code
- LLMORE Parameters
- Architectural Model

LLMORE

LLMORE output
1. Set of optimized maps or
2. Performance data or
3. Optimized architecture(s) or
4. Generated code or
5. Results from run on target architecture

Key/Novel Features
- Application code to simulator
- Data mapping optimization for user code
- Support for multiple languages and libraries
- Ability to solve large problems
- Production quality software

Output: One or more
LLMORE Design Overview

Language Interface
- Parse Manager
- Map Manager

Parser
Map Converter

AST Builder
Map Builder

Analyzer and Optimizer

Core Functionality

Code Generator
Runtime Engine

Machine Interface

LLMORE
Input

LLMORE
Output

LLMORE
Input

LLMORE
Output

AST = abstract syntax tree
Outline

• Motivation/Overview
• Design and Usage
  – Usage 1: Map Optimization
  – Usage 2: Performance Evaluation
• LLMORE and POEM
• Preliminary POEM Results: 2D FFT
• Next Steps and Summary
LLMORE Usage 1: Map Optimization

- LLMORE produces set of optimized maps for parallel variables specified in user code
- Matrix-vector product example
  - LLMORE computes map for dense matrix
  - LLMORE computes map for two vectors

LLMORE optimizes data mapping to improve parallel performance of key computational kernels
Map Optimization: Input

Input from application: user code for dense matrix-vector product

Dense Matrix-Vector Product

User Code:

\[ y = Ax \]
Map Optimization: AST Representation

Parser converts user code into abstract syntax tree (AST), which is input language/numerical library neutral

Dense Matrix-Vector Product

y = Ax

Parser

Mapper

Analyzer and Optimizer

Optimized Maps

LLMORE

SB

MV

AST

PVar: y

PVar: A

PVar: x
LLMORE computes map for each parallel variable in AST

Map Optimization: Mapping

LLMORE computes map for each parallel variable in AST
Map Optimization: Output

- LLMORE output: optimized maps for parallel variables
- New maps used to redistribute vector and matrix data
- Optimized matrix-vector product calculated with new data distributions

\[ y = Ax \]
Outline

• Motivation/Overview
• Design and Usage
  – Usage 1: Map Optimization
  – Usage 2: Performance Evaluation
• LLMORE and POEM
• Preliminary POEM Results: 2D FFT
• Next Steps and Summary
LLMORE Usage 2: Performance Evaluation

LLMORE simulates user code on specified architecture to produce performance evaluation metrics.
Performance Evaluation: Input

Input from application: user code for dense matrix-vector product, architecture model

Dense Matrix-Vector Product

User Code:
\[ y = A x \]

Architecture Model:

LLMORE

MI = Machine Independent
Performance Evaluation: AST Representation

Parser converts user code into abstract syntax tree (AST), which is input language/numerical library neutral
Performance Evaluation: Mapping

LLMORE computes map for each parallel variable in AST

Dense Matrix-Vector Product

MV

SB

PVar: y
Vector Map
P0: {0,1}
P1: {2,3}

PVar: A
Matrix Map
P0: {0,1}
P1: {2,3}

PVar: x
Vector Map
P0: {0,1}
P1: {2,3}
Performance Evaluation: Machine Independent Code

Mapped AST and architecture model used to generate machine independent code
Machine Independent Code

# rows = 2
# processors = 2

Machine independent code for matrix-vector product
Simulation of user code on target architecture
Outline

• Motivation/Overview
• Design and Usage
  – Usage 1: Map Optimization
  – Usage 2: Performance Evaluation
• LLMORE and POEM
• Preliminary POEM Results: 2D FFT
• Next Steps and Summary
POEM will bridge the gap between innovations in chip-scale photonics and fielded military-critical systems. It will offer a complete architecture design and analysis for numerous real-world military-critical applications.
LLMORE’s Role in POEM Program

LLMORE used to study chip-scale photonics and its impact on applications.
LLMORE and POEM: Applications

- LLMORE provides framework for analyzing POEM applications
  - LLMORE supports many key numerical kernels (FFT, sparse matrix-vector product, vector updates, etc.)
  - Applications supported through composition of these kernels
  - Easy to extend to analyze new applications
- Initially analyzing synthetic aperture radar (SAR) application

LLMORE enables the analysis of many applications on many different architectures (existing and proposed)
LLMORE and POEM: Architectures

- LLMORE supports simulation of applications on POEM architectures (e.g., electronic mesh and photonic bus)
- Framework for simulating user code
  - LLMORE simulator for understanding big picture trends
  - Interface to third party simulators (e.g., PhoenixSim) for higher fidelity performance data

LLMORE enables the analysis of many applications on many different architectures (existing and proposed)
Optimization of maps

- Good application data to processor mapping crucial to achieving peak parallel performance on target machines
- Not difficult for SAR applications (simple maps sufficient)
- Challenging for applications with irregular communication (DNA sequence analysis and sparse matrix computations)

LLMORE provides automatic map optimization
Outline

- Motivation/Overview
- Design and Usage
  - Usage 1: Map Optimization
  - Usage 2: Performance Evaluation
- LLMORE and POEM
- Preliminary POEM Results: 2D FFT
- Next Steps and Summary
LLMORE and Performance Evaluation

LLMORE used to produce performance evaluation data for 2D FFT, an important kernel in SAR processing chain.
Two architectures modeled (electronic mesh and photonic bus)
Processors same, shared memory same
Network parameters (latency, bandwidth) set to allow for apple to apples comparison between networks

LLMORE framework allows direct comparison of electronic mesh and photonic bus architectures
Preliminary POEM Simulations

1D FFT for each row of dense matrix

For data locality of FFT

1D FFT for each column of dense matrix

1D FFT for each row of dense matrix

For data locality of FFT

1D FFT for each column of dense matrix

EM = Electronic mesh
PB = Photonic bus
SCA = Synchronous coalesced access

• Initially, simulated 2D FFT kernel
• Important kernel for SAR applications
Photonic Synchronous Coalesced Access Network (PSCAN)

PSCAN leverages the differences between electronic and photonic interconnect to achieve large efficiency gains in critical operations.

- Utilizes distance independence to rapidly re-organize spatially separate data
- Large gains in efficiency even when bandwidth is equalized
- SCA write
  - Synthesizes matrix row-to-column transpose/write into a single transaction by interleaving data from spatially separate data producers
  - Novel ISA construct enabled by a highly synchronous photonic waveguide

Credit: Dave Whelihan, MITLL
Preliminary LLMORE Results

Experiment:
• Number of memory controllers: 4
• Matrix size: 4096 x 4096
• Equalize bandwidth to memory in photonic and electronic models
• Run traditional 2D FFT with block transpose for electronic mesh
• Run 2D FFT with SCA for PSCAN
• Scale number of processors

Examine performance of photonics and electronics in an increasingly work-starved environment

PSCAN architecture with SCA yields significant speed-up over electronic mesh architectures using block transpose
Data Reorganization Bottleneck in Modern Manycore Systems

As the number of cores in modern CMPs grows, the data reorganization becomes an increasing performance bottleneck. Performance is limited by the inability to keep processors supplied with data.

SCA operation performs data reorganization step of 2D FFT more efficiently as number of processors grows, alleviating traditional block transpose bottleneck.
Outline

• Motivation/Overview

• Design and Usage
  – Usage 1: Map Optimization
  – Usage 2: Performance Evaluation

• LLMORE and POEM

• Preliminary POEM Results: 2D FFT

• Next Steps and Summary
LLMORE Next Steps

• Extend architecture model, LLMORE simulator
  – Support memory hierarchy
  – Model network contention more accurately
• Support for external simulators
  – E.g., PhoenixSim (Columbia), SST (Sandia)
  – Needed for higher fidelity simulations
• Better power modeling (e.g., dynamic power)
• Additional parallel numerical library/language support
  – Additional languages: Matlab, Python
  – Additional libraries: e.g., VSIPL++/PVTOL
  – Additional kernels: e.g., Sparse matrix operations
• Code generator and runtime engine for execution/emulation on target architectures
Summary

• LLMORE: Parallel framework/environment for
  – Optimizing data to processor mapping for parallel applications
  – Simulating and optimizing new (and existing) architectures

• LLMORE used to compare photonic and electronic architectures of interest to POEM project
  – Support for many applications (through composition of kernels)
  – Support for different architectures

• Preliminary results for 2D FFT kernel
  – Support thesis that photonics can improve performance for these applications
  – SCA instruction mitigates performance impact of corner turn in 2D FFT operation

LLMORE allows for exploration of architecture design space in context of real application constraints
Acknowledgements

• LLMORE
  – Michelle Beard
  – Anna Klein
  – Sanjeev Mohindra
  – Julie Mullen
  – Eric Robinson
  – Nadya Bliss (Manager)
  – Minna Song (MIT student intern)

• POEM
  – Dave Whelihan
  – Jeff Hughes
  – Scott Sawyer