Polyhedral Mapping Assistant and Visualizer (PUMA-V)
Department of Energy Office of Science, Office of Nuclear Physics, SBIR/STTR Exchange Meeting, August 6-7, 2015

\[ A_{1024} \times 1024 \]

\[ S: \text{for} \quad i \quad \text{for} \quad j \quad \text{for} \quad k \]

\[ S_5 = \{ 0, 0, 0, 0, 1 \} \]

\[ A_{5 \times 5} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \end{bmatrix} \]

1. Order
2. Permutability
3. Locality

\[ x_{i,j} = A \cdot (i \times j) \cdot \text{true only} \]
Polyhedral Mapping Assistant and Visualizer (PUMA-V)

Project and Motivation

- US Department of Energy contract number DE-SC0009678:
  - Phase II STTR, 2014 Award Year.
  - April 15, 2014 – April 15, 2016 term
  - Solic. Number DE-FOA-0001019, Topic Code 39e
  - Reservoir Labs, SUNY StonyBrook, Brookhaven National Lab

- Motivation:
  - Lattice QCD community has traditionally produced very efficient home-grown software and is continuing to do so.
  - However, with the arrival of new hardware architectures, significant efforts are required to optimize the software for the target new architecture.
    - One way to deal with this is to rewrite the software to be “future-proof”.
    - Another way is to see if there are any automation tools that are capable of producing efficient code for a target architecture from generic, high-level, user codes.
  - As a Phase II US DOE SBIR/STTR funded program, PUMA-V explores the second approach, seeking to use the R-Stream source-to-source compiler to optimize the Domain Wall Dirac operator implementation as well as develop accelerated solvers and state-of-the-art visualization methods
  - Columbia Physics System (CPS) as initial target.
Polyhedral Mapping Assistant and Visualizer (PUMA-V)

Goals for Accelerating Code Production for Nuclear Physics

- Algorithms for automatic generation of highly optimized heterogeneous code based on inherent properties of Lattice QCD problem, providing opportunities for substantial speed-ups in computation.
  - Compiler technology (R-Stream) to help with specific computations, especially those related to the proton size puzzle and muon magnetic moment, of particular interest to our domain expert partners (BNL).

- An automated visualization tool-chain to assist with software optimizations and mappings in high dimensional spaces, of great utility not only to the physics community but to the larger scientific computation enterprise.
  - Performance visualizer and IDE plugin development for intuitive code generation (Stonybrook)

- New opportunities to lower cost and power requirements for large computations needed in fundamental physics, fluid dynamics, computational biology and other scientific disciplines.
  - Faster preconditioners and solvers (CMG, Peng-Spielman, etc.)
PUMA-V Teams

PUMA-V Personnel

• Reservoir Labs, Compiler Technology and Solvers:
  – M. Harper Langston, PhD – PI of PUMA-V project
  – Richard Lethin, PhD – President of Reservoir Labs
  – Paul Mountcastle, PhD – Physicist
  – Benoit Meister, PhD – Managing Engineer
  – Muthu Baskaran, PhD – Managing Engineer
  – Tom Henretty, PhD – Senior Engineer

• Brookhaven National Lab, Domain Experts:
  – Taku Izubuchi, PhD
  – Meifeng Lin, PhD
  – Chulwoo Jung, PhD

• StonyBrook University, Visualization and Compiler Technology:
  – Klaus Mueller, PhD
  – Eric Papenhausen
  – Bing Wang
Reservoir Labs, Inc.

Founded 1990 – Offices in New York City and Portland, OR

• Involved in a variety of research projects to explore ways to solve dynamic systems and effectively compile real-world algorithms. Work for and with numerous government institutions, and collaborate closely with other leading researchers and academics around the world. Sample projects:
  - **R-Stream®** – advanced compiler technology enables developers to create program logic once and produce optimized code for multiple parallel computing architectures.
  - **R-Solve®** – An automated reasoning technology that addresses dynamic problems in advanced planning and decision analysis, modeling and simulation.
  - **ENSIGN** – Cutting-edge hypergraph analysis technology for Big Data applications spanning security, finance and biology.

• Provide services and products to commercial clients through research technologies for organizations working on novel high-performance systems; package those technologies in turnkey commercial and government solutions that address important science and security issues. Sample products:
  - **R-Scope Network Security Monitoring** – Puts networks under a microscope so customers can respond to both known and zero-day attacks before becoming crises.
  - **R-Check® SCA** – Simplifies and accelerates SCA-compliance testing for defense communication systems worldwide, shortening the timeframe for checking from weeks and months to hours.

• More than 30 full-time researchers and engineers (half with PhDs) and a mature business development department
R-Stream Polyhedral Model Compiler

Developed by Reservoir Labs Inc.

- A high-level source-to-source compiler based on the polyhedral model, a mathematical abstraction for analysis and transformation of computer programs:
  - Darte, Schreiber & Villard, 1985
  - Feautrier 1992
- Performs optimizations in terms of parallelization, memory management, locality etc. and can target a range of hardware architectures.
- Accepts a sequential C program as input and produces code in a variety of formats, including C + OpenMP and CUDA.
  - Meister et. al, 2011
  - Vasilache et. Al, 2013
- Used for PUMA-V in targeting key code kernels in the LQCD formulations
Optimizing the Domain Wall Fermion Dirac Operator
Using the R-Stream Source-to-Source Compiler
Domain Experts at BNL Directed Focus

\[ M_{x,s,x',s'} = \delta_{s,s'} M_{x,x'}^\parallel + \delta_{x,x'} M_{s,s'}^\perp \]

\[ M_{x,x'}^\parallel = -\frac{1}{2} \sum_{\mu=1}^{4} \left[ (1 - \gamma_\mu) U_{x,\mu} \delta_{x+\mu,x'} + (1 + \gamma_\mu) U_{x',\mu}^\dagger \delta_{x-\mu,x'} \right] \]

\[ + (4 - M_5) \delta_{x,x'} \]

\[ M_{s,s'}^\perp = -\frac{1}{2} \left[ (1 - \gamma_5) \delta_{s+1,s'} + (1 + \gamma_5) \delta_{s-1,s'} - 2\delta_{s,s'} \right] \]

\[ + \frac{m_f}{2} \left[ (1 - \gamma_5) \delta_{s,L_{s-1}} \delta_{0,s'} + (1 + \gamma_5) \delta_{s,0} \delta_{L_{s-1},s'} \right] \]

Wilson Kernel
The Wilson Kernel

R-Stream Focus

- We first looked at how R-Stream was able to transform the single-core version, unimproved, implementation of the Wilson Dslash operator in CPS.

\[ D_{\alpha\beta}^{ij}(x, y) = \sum_{\mu=1}^{4} \left[ (1 - \gamma_{\mu})_{\alpha\beta} U_{\mu}^{ij}(x) \delta(x + \hat{\mu} - y) + (1 + \gamma_{\mu})_{\alpha\beta} U_{\mu}^{+ij}(x + \hat{\mu}) \delta(x - \hat{\mu} - y) \right] \]

```c
for(x=0; x<lx; x++){
    for(y=0; y<ly; y++){
        for(z=0; z<lz; z++){
            for(t=0; t<lt; t++){
                //
                printf("wilson_dslash: %d %d %d %d\n", x,y,z,t);
                parity = x+y+z+t;
                parity = parity % 2;
                if(parity == cbn){
```
There are, however, a few transformations required on the input code before R-Stream can process it.

- **Delinearized array access:**
  - In CPS, arrays are linearized. However, the polyhedral model requires that the array indices are affine functions of the outer loop indices so that it can perform dependence analysis:

    \[
    \chi[NT*NZ*NY*NX/2*4*3*2] \Rightarrow \chi[NT][NZ][NY][NX/2][4][3][2]
    \]

- **Introduction of padding at boundaries:**
  - In the original CPS code, the wrap-around at the boundary is handled with array index modulo, e.g., \( \chi[(x+1)\%NX] \), which R-Stream can't analyze directly.
  - For each dimension, padding was added for the two boundaries, leading to new dimensions for the fermion vector arrays:

    \[
    \chi[NT][NZ][NY][NX/2][4][3][2] \Rightarrow \chi[NT+2][NZ+2][NY+2][NX/2+2][4][3][2]
    \]

- **Despite several improvements made to R-Stream, it wasn't able to produce optimally efficient code. Some hand tuning was needed to achieve any speedup.**
Hand Tuning *After* R-Stream

Hand-tuning necessary for more optimal code

- First hand tuning done was to vectorize the code, with both **SSE** and **AVX** intrinsics
  - Intel's Streaming SIMD Extension (SSE) can pack two doubles into one vector register and perform two double operations per instruction.
  - For Wilson Dslash, the vectorization was done by packing real and imaginary parts into one register.

- **SSE** version achieved 2X speedup compared to the input code.
- The **AVX** extension allows to compute 4 doubles with a single instruction:
  - A data layout transform was needed to pack the four spinor components into one vector register:
    - \( \text{chi}[\text{NT}][\text{NZ}][\text{NY}][\text{NX}/2][4][3][2] \rightarrow \text{chi}[\text{NT}][\text{NZ}][\text{NY}][\text{NX}/2][3][2][4] \)
Performance of Wilson Dslash

Results from Lattice 2015: M. Lin, "Optimizing the DWF Dirac Operator Using the R-Stream Compiler", Kobe Japan, July 2015

- CPS-SSE is a heavily hand optimized version of the Wilson Dslash written by Taku Izubuchi.
- Tests were run on Intel i5-2520M dual core @ 2.5 GHz.

Best performance with 1 thread/core
Vectorization for the DWF Dslash

The core piece of the DWF Dslash operator is inlining the Wilson Dslash over the loop of NS

- for (s=0; s<NS; s++) //call Wilson Dslash
  - To adapt the code for the AVX instructions, the data layout had to be transformed:
    - chi[NS][NT][NZ][NY][NX/2][4][3][2] → chi[NT][NZ][NY][NX/2][3][4][NS][2]
  - The AVX register stores the real and imaginary parts of two sites in the s dimension.

- Initial tests show that the single-core AVX version for the DWF Dslash performs reasonably well.
  - One a single core of Intel Xeon E5–2670 @ 2.60GHz, the 4D DWF Dslash achieved a performance of 5.2 Gflops in double precision, or 25% peak (Lattice 2015)
    - Tests were performed with a local volume of 8^4x16.
    - Incorporating MPI communications into the current code creates significant performance drop.
    - We are currently investigating this issue.

<table>
<thead>
<tr>
<th># cores</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5.2 Gflops</td>
</tr>
<tr>
<td>2</td>
<td>4.0 Gflops</td>
</tr>
<tr>
<td>16</td>
<td>1.2 Gflops</td>
</tr>
<tr>
<td>32</td>
<td>0.7 Gflops</td>
</tr>
</tbody>
</table>
PUMA-V Visualizer

Visualizer Eclipse Plugin with Performance Tuning (Stonybrook Focus)

- Automated toolchain to allow user-in-the-loop to make more intuitive decisions for parallelization and R-Stream compiler optimizations. (Video)

- Performance Profiling

- Tool and results accepted and to be published in Proceedings of IEEE VISSOFT 2015
Spectral Support Preconditioning and Nearly-Linear Time Solvers, Combinatorial Multigrid

Solving the linear system $Ax = b$ with actual solution $x := A^{-1} b$.

- Find $B$ that approximates $A$ in a spectral sense, so solving $By = c$ is easier
- Based on Spielman and Teng (2003-current), solve a system of linear equations with a symmetric diagonally dominant (SDD) discrete operator:

$$Ax = b, \quad A \in \mathcal{R}^{n \times n}, \quad A = A^T, \quad A_{ii} \geq \sum_{i \neq j} |A_{ij}|$$

- SDD systems have clear connection with graphs and Laplacians:

- Low-stretch trees approximate most distances to within $O(\log m)$ using only $m-1$ edges:

- Approaches incorporate elements of direct and iterative solvers for class of problems with graph clustering, spectral sparsification, partial factorizations, minimal trees for state of the art towards $O(m \log^c n)$ solver for $m$ edges and $n$ vertices (Koutis et. al 2009: CMG, Kelner et. al 2013, Peng-Spielman 2014)
Combinatorial Multigrid Issues

- Sparsifiers generate a “smaller” matrix or analogous graph structure, which preserves many graph parameters.

- Computationally very expensive and only seemingly practical in the theoretical realm! Further, computing on the fly tricky. The idea is to maintain “cliques” and “cycles”.
  - Can be satisfied if the general structure remains in a simpler fashion using simpler strategies.

- Complex-valued data for NP problems
- NP problems not diagonally-dominant, though CMG can still handle this
Preconditioner Sample Results

- Small non-LQCD SDD system for testing MATLAB CMG:
  - Significantly-reduced iteration counts as condition numbers grow

- Incomplete Cholesky-based preconditioners on small LQCD system:

<table>
<thead>
<tr>
<th>Test Type</th>
<th>$N_{iter}$</th>
<th>$T(s)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Preconditioner ($P = I$)</td>
<td>3352</td>
<td>35</td>
</tr>
<tr>
<td>IC(0) (no-fill Incomplete Cholesky)</td>
<td>1780</td>
<td>36</td>
</tr>
<tr>
<td>ICT (thresholding with small values)</td>
<td>22</td>
<td>6</td>
</tr>
<tr>
<td>IC with diagonal compensation ($\alpha \approx 1e^{-4}$)</td>
<td>652</td>
<td>17</td>
</tr>
</tbody>
</table>

- IC-based preconditioners show great promise, but LQCD matrices are computed at run-time, so computing incomplete factorization may require too much memory in practice
Summary and Ongoing Goals

• In the past year we have explored the possibility of using the R-Stream source-to-source compiler to optimize the Wilson Dslash in CPS.
  – So far R-Stream hasn't been able to produce highly optimized code for CPS.
    • Working on stabilizing R-Stream scheduler to handle more complex codes.
  – Further manual optimizations based on the R-Stream output include incorporating SSE and AVX intrinsics, which give the expected 2X and 4X speedup, respectively, compared to the un-optimized single-core input code.
  – Currently working to fix implementation of the MPI communications causing significant performance drop for DWF Dslash.
  – Single-precision AVX implementation.
  – Incorporate new code into physics production runs.

• Incorporate newer R-Stream versions into Visualizer Tool
  – More performance tuning implementations and auto-tuning/optimizations in tool for more optimal compiler options.

• Preconditioners and Solvers
  – Complete C implementation of CMG and modify for LQCD systems.
  – Work on memory-efficient Incomplete factorizations.
Thank You!
Questions/Comments
R-Stream Compiler Additional Slide

R-Stream 3.0 Compiler

- Raising
- High-Level Dependence Analysis
- Parallelization-Fusion
- Tiling
- Locality Enhancement
- Placement
- Communication Generation
- Synchronization Generation
- Thread Generation

Capabilities beyond previous compilers

- Input programs: increased scope of optimizations
  - Imperfect loop nests.
  - Parametric affine programs.
- Out: increased scope of targets
  - Optimizations and transformations for multi-core machines.
- Based on mathematics:
  - Unique implementation of capabilities that would otherwise just be theory.
  - Less "magic"
  - First opportunity to use theoretical techniques on C

Leveraged Investment

One high-level optimizer leveraged across multiple architectures and implementations.

Virtual Machine Abstractions (language + library calls)

Programming Challenges

Multi-core Hardware:
- 10-100x FLOPS/Watt advantage
- Parallelism
- SIMD
- Heterogeneous Functional Units
- Coarse grain communication ops
- Streaming Engines
- Tiled Chip Multiprocessing
- Distributed local memories
- Starved for pin bandwidth

Cell GPU FPGA TILE64

Reservoir Labs 8.6.15