Application of KGen and KGen-kernel

Youngsung Kim and John Dennis

Sep. 14, 2016

NCAR
Contents

• Introduction
• KGen kernel in practice
  – Optimization and Porting
  – Validation, Test collection, Profiling, etc.
• Extension of KGen
• Limitations and Plans
• Demo
Let’s Share It!

**Usual Tasks**

- Configuration
- Input data preparation
- Building dependencies
- Compilation/linking
- Submitting to queue
- Post-processing

**KGen does**

- Parsing Fortran Code
- Analyzing on ASTs
- Extracting code
- Instrumenting for data

**KGen Kernels**

- Stand-alone Executable
- In/out data from app.
- Verification
- Timing
- Perturbation
- Makefile for build/run

**ASTs: Abstract Syntax Trees**

**User specifies**

- Where to extract
- How to clean/build/run application

**KGen**

Generates

**Kernel**

- Kernel Files
- Makefile
- In/Out State

**Application**

- Source Files
- External Libraries
- Building System

**Data**
KGen Kernel Github Repository
(https://github.com/NCAR/kernelOptimization)
Case 1: MG2 Kernel Optimization

- Takes about 10% computation time
- Optimized together with compiler experts.
  - Easy to reproduce results on both-sides
- Explored various optimizations with
  - CPUs, compilers, clusters
- Easy to apply profiling (Folding: on KNL)

[Graph showing MG2 Vectorization Improvement - PAPI counters on KNL]

KGen and KGen-kernel
Case 2: WACCM Chemistry kernel Optimization

- Auto-generated from chemistry equations
- Memory-bound

\[
\begin{align*}
lu(k, 85) &= lu(k, 85) \times lu(k, 82) \\
lu(k, 1133) &= -lu(k, 83) \times lu(k, 1101) \\
lu(k, 1372) &= lu(k, 1372) - lu(k, 83) \times lu(k, 1366)
\end{align*}
\]

- Explored different vector lengths, FP precisions, array sizes, blocking, and so on
- Folding Analysis on Intel KNL
More use-cases

- MPAS Porting onto GPU
  - Dynamic core was extracted as a kernel
  - Applied OpenAcc and being optimized and verified

- Model Validation
  - Extracted “suspicious” region of code
  - Tried many evaluations quickly with a kernel
More use-cases - continued

- **Procurement Benchmark**
  - MG2 kernel as a part of NCAR’s NWSC-2 production

- **Compiler Regression Test Suite**
  - KGen-kernels are being considered to be a part of PGI compiler test suite

- **Quick profiling**
  - Extracted a small DO loop from WRF and performed analyses related to compiler options and assembly code on Intel KNL

- **Extension of KGen**
  - Trying to incorporate KGen-kernel extraction as a part of unit-test framework.
Support for KGen Users

- Who tried KGen or KGen kernels from
  - Compiler vendors: Cray, Intel, Nvidia
    - Optimization, compiler bug report
  - Universities: U. of Hamburg, ETH Zurich, U. of Tennessee, U. of Utah
    - KGen bug reports, GPU porting, Tool development, Unittest case generation
  - Research/Public centers: GFDL, NOAA, ORNL, DKRZ
    - Analysis on new CPUs, Optimizations
- Online support
  - Online documentation: https://ncar.github.io/kgendocs/
  - Discussion group: https://groups.google.com/forum/#!forum/kgen-discuss
Getting-started

>> git clone https://github.com/NCAR/KGen.git

>> cd KGen/example/simple-region

>> vi src/Makefile

>> make
Getting-started

>> make

kgen \n  src/update_mod.F90 \n  --cmd-clean "cd src; make clean" \n  --cmd-build "cd src; make build" \n  --cmd-run "cd src; make run"

# target source file that contains callsite
# Linux command(s) to “clean” target application
# Linux command(s) to “build” target application
# Linux command(s) to “run” target application
Output from KGen kernel execution

*************** Verification against 'calc.0.0.1' ***************

Number of output variables: 3
Number of identical variables: 3
Number of non-identical variables within tolerance: 0
Number of non-identical variables out of tolerance: 0
Tolerance: 1.000000000000000E-014

Verification PASSED

calc : Time per call (usec): 1.9999995296516E-002

************************************************************************

calcul kernel execution summary: calc
************************************************************************

Total number of verification cases : 3
Number of verification-passed cases : 3

Average call time (usec): 0.233E-01
Minimum call time (usec): 0.200E-01
Maximum call time (usec): 0.300E-01

************************************************************************
### KGen structure

<table>
<thead>
<tr>
<th>KGen</th>
<th>Uses</th>
</tr>
</thead>
<tbody>
<tr>
<td>F2PY Parser</td>
<td>- cpp</td>
</tr>
<tr>
<td></td>
<td>- strace</td>
</tr>
<tr>
<td></td>
<td>- make</td>
</tr>
</tbody>
</table>

- **Written in Python**
  - No external compiler or analysis framework
  - Performs static analysis on ASTs
- **Uses F2PY* Fortran Parser**
  - Integrated and extended in KGen
  - Produces ASTs for analysis
- **Uses three Linux utilities**
  - cpp: preprocessing
  - strace: collect macros and include paths
  - make: executes KGen-generated kernel

- **F2PY***: Fortran to Python interface generator (https://github.com/pearu/f2py)

---

**KGen and KGen-kernel**
KGen Github Repository
(https://github.com/NCAR/KGen)
KGen as a source-to-source compiler

• Event-driven source modification

Kernel Extraction Plugin
Output Verification Plugin
Perturbation Plugin

Event register with callback
Call callback with nodes
Modify nodes and ASTs

KGen Plug-in Framework
• Goes through all nodes in ASTs to check if the node matches with registered events

• Applications of KGen
  – Current KGen is based on the plug-in framework
  – A group of Oak Ridge Nat’l Lab/U of Tennessee tries to use KGen to replace their kernel generation component in their unittest framework for ACME Land Model
  – Automated optimization (Experimental)
Limitations and Plans

• Limitations and Development Plans
  – Providing metrics for representation
    • Record PAPI counters from original application and compare with those from KGen-kernels
  – Enhancing MPI support
    • Allows (or imitates) MPI library calls within KGen-kernel
  – Open APIs for KGen-extension
    • General purpose src-to-src compiler
• General Improvement
  – Supporting better Fortran specification
  – Resolving known-limitations
  – Setting-up continuous integration Tests
• User Support
  – Improving documentation
  – KGen discussion group
Demo

1. Extraction MG2 from CESM
2. MG2 Kernel Execution
Thank You

Q & A