PerfExpert.- Workflow for Vectorization and Parallel Scalability

Antonio Gómez-Iglesias, PhD
Research Associate
Texas Advanced Computing Center
Austin, TX

https://www.tacc.utexas.edu/perfexpert
https://github.com/TACC/perfexpert

What's PerfExpert?

- **PerfExpert: an easy-to-use automatic performance diagnosis and optimization tool for HPC applications**
- **Goal: get close to optimality with minimal input from the user**
- **Open source**
- **Modular design that allows different tools to be integrated into the PerfExpert framework**
  - HPCToolkit, VTune, MACPO, MACVEC
- **User friendly, robust, and great recommendation system**
- **Automatic changes in the code**
- **It works with:**
  - Host
  - KNC
  - What about KNL?

What about KNL? Combined Vectorization and Parallel Scalability

- **Vectorization and parallel scalability become more critical with each new generation of chips. PerfExpert for the KNL will combine optimization of vectorization and parallel scalability and later add internode communication optimization**
- **MACVEC, part of PerfExpert, performs vectorization analysis to maximize the vectorization**
- **PerfExpert:**
  - Checks intra-node load balance
  - Determines the optimal number of threads for the application
  - Optimizes the memory access behavior for the optimal number of threads
  - Reruns the scalability analysis
- **Internode communication optimization: integration with MPI Advisor**

Example of MACVEC Vectorization Optimizations

- **MACVEC searches for constructs which prevent the compiler from vectorizing a loop**
- **Specifies what information must be given to the compiler to enable vectorization**
- **Example of Array Alignment**
  1. Analysis - The first-referenced address of the referenced array is used to understand the alignment of the data structure.
  2. Optimization Recommendations:
     - Arrays are aligned and loop is vectorizable, use the #pragma vector aligned directive
     - Arrays are not aligned or mutually aligned to the same alignment value use mm_malloc() for heap memory and using the attribute ((aligned(64))) clause for global, static and stack memory
  3. Nbody Code on KNC – speedup of **30%**

Example of MPI Advisor Optimization

- **MPI Advisor currently provides recommendations addressing frequent MPI-related performance bottlenecks:**
  1. point-to-point protocol (eager vs. rendezvous)
  2. collective communication algorithm
  3. MPI tasks-to-cores mapping
  4. Infiniband transport protocol

- **Example: choice of Infiniband transport – RC/UD**
  - Recommends choice of RC or UD depending on number of tasks and message and buffer sizes
  - Execution time (seconds) for SMG2000 running on 4,096 cores with Intel MPI.

RC 124.09 seconds

UD 87.78 seconds

PerfExpert Dependencies

- **Depending on the desired functionality, PerfExpert has different dependencies with other tools:**
  - Rose compiler
  - HPCToolkit
  - VTune
  - libelf
  - Java
- **Once installed, the integration of these tools into PerfExpert is transparent for the users**
- **Not all the dependencies are required in all the platforms**

Summary

- **Multilevel Scalability Optimization**
  - PerfExpert for the KNL will support attainment of comprehensive parallel scalability spanning vectorization, node level scalability and internode scalability.
- **Ease of Use**
  - A single command line will be necessary to apply the parallel scalability analysis to most of the applications.
- **Limitations**
  - The Rose compiler infrastructure fails on a substantial fraction of applications using newer Fortran dialects. We are looking for more robust solutions.

References

[1] PerfExpert - https://www.tacc.utexas.edu/perfexpert

[2] MACVEC - Ashay Rane, Rakesh Krishnaiyer, Chris Newburn, James Browne, Leonardo Fialho, Zakhar Matveev. Unification of Static and Dynamic Analyses to Enable Vectorization

[3] MPI Advisor - Esthela Gallardo, Jerome Vienne, Leonardo Fialho, Patricia Teller, James Browne. MPI Advisor: a Minimal Overhead Tool for MPI Library Performance Tuning

BACKUP SLIDES

Combining Compiler Analyses and Runtime Measurement to Enhance Vectorization MACVEC

Ashay Rane – UT TACC

Rakesh Krishnaiyer, Chris J Newburn and Zakhar Matveev - Intel

Jim Browne, and Leo Fialho – UT TACC

Motivation

- Chip and node architectures and modern languages are both very complex. Best (most efficient) executables are thus complex to write.
- Compiler static analyses alone cannot generate optimal executable
- Optimization based on runtime measurements is much more effective when guided by static analysis
- Vectorization is becoming a critical factor for both execution speed and power consumption.

Workflow

- 1. Profile application for hotspots using production inputs.
- 2. Parse compiler vectorization reports to find why loops not fully vectorized and what information the compiler needs.
- 3. Instrument hot-loops that are not fully vectorized for information compiler could not determine
- 4. Gather measurements, analyze results and generate recommendations for enhancing vectorization.
- 5. Verify validity of the recommended changes.
- 6. Implement changes, measure performance gains.
  - Automated step
  - Manual step

Potential Benefit

Rodinia Benchmarks for Accelerator Performance

Application	Time
heartwall	07.43%
euler	12.42%
kmeans	19.54%
backprop	32.52%
leukocyte	35.01%
lavaMD	37.42%
srad_v1	48.45%
pre_euler_double	71.60%
pre_euler	75.94%
euler_double	78.99%
streamcluster	85.58%

Fraction of Execution time spent in incompletely vectorized loops.

Safety of Recommendations

- **Are recommendations independent of standard compiler optimizations?**
- **Will recommendations be applicable across multiple program inputs?**
  - Seven of the nine recommendations are guaranteed to be safe.
  - O(1) runtime checks guarantee safety for remaining recommendations.
  - Source-code level measurements ensure that measurements are valid across compilers.

MPI Advisor

a Minimal Overhead Tool for MPI Library Performance Tuning

Esthela Gallardo and Pat Teller
Dept. of Computer Science, UT-El Paso
{egallardo5@miners.utep.edu, pteller@utep.edu}
and
Jerome Vienne, Leonardo Fialho and Jim Browne
Texas Advanced Computing Center (TACC), UT-Austin
{viennej@tacc.utexas.edu, fialho@tacc.utexas.edu, browne@cs.utexas.edu}

Motivation/1

- MPI is pervasive.
- Features to optimize performance are library dependent.
- Most users employ default library- and cluster-specific parameters.
- Many jobs may have MPI-related performance issues.
- Needed: an easy-to-use tool for non-experts to optimize MPI communication performance.

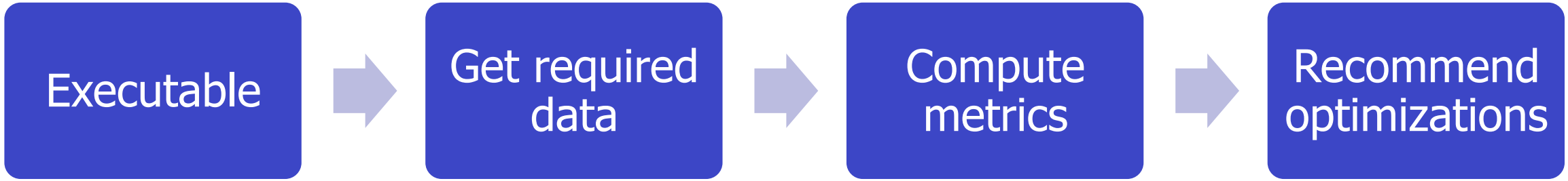
Motivation/2

Available MPI Performance Tools

	MPI Advisor (TACC/ UTEP)	OPTO (PSTL/ UH)	Atune (UIBK)	MPITune (Intel)	Periscope (TUM)
Single run	X				
Multiple libraries	X				
Basic privileges	X	X	X	X	
Inter-node optimization	X	X	X	X	X
Intra-node optimization				X	X
Message passing optimization	X	X	X	X	X
Does not require expert knowledge	X				

Approach - Conceptual

Approach - Operational



Currently Supported Tuning Strategies

Point-to-Point Protocol Threshold

- Eager vs. Rendezvous

Choice of Algorithms for Collective Operations

- Depends on system size, message size, and task properties

Mapping of MPI Tasks to Cores

- Map Task 0 to socket that shares the PCI Express bus with the HCA card
- Default mappings vs. custom mappings

Infiniband Transport: RC and UD

- Tradeoff between memory footprint and message size

Point-to-Point Protocol Threshold: Demonstration/1

- **Application: CFOUR-based benchmark**
  - Reads and writes fixed records in random order.
  - Messages are mainly point-to-point with sizes around 128 KB or less.
- **Recommendation: Since the default eager threshold is 17 KB, MPI Advisor recommends increasing the eager threshold to more than 131,072 bytes.**

MPI Advisor Output

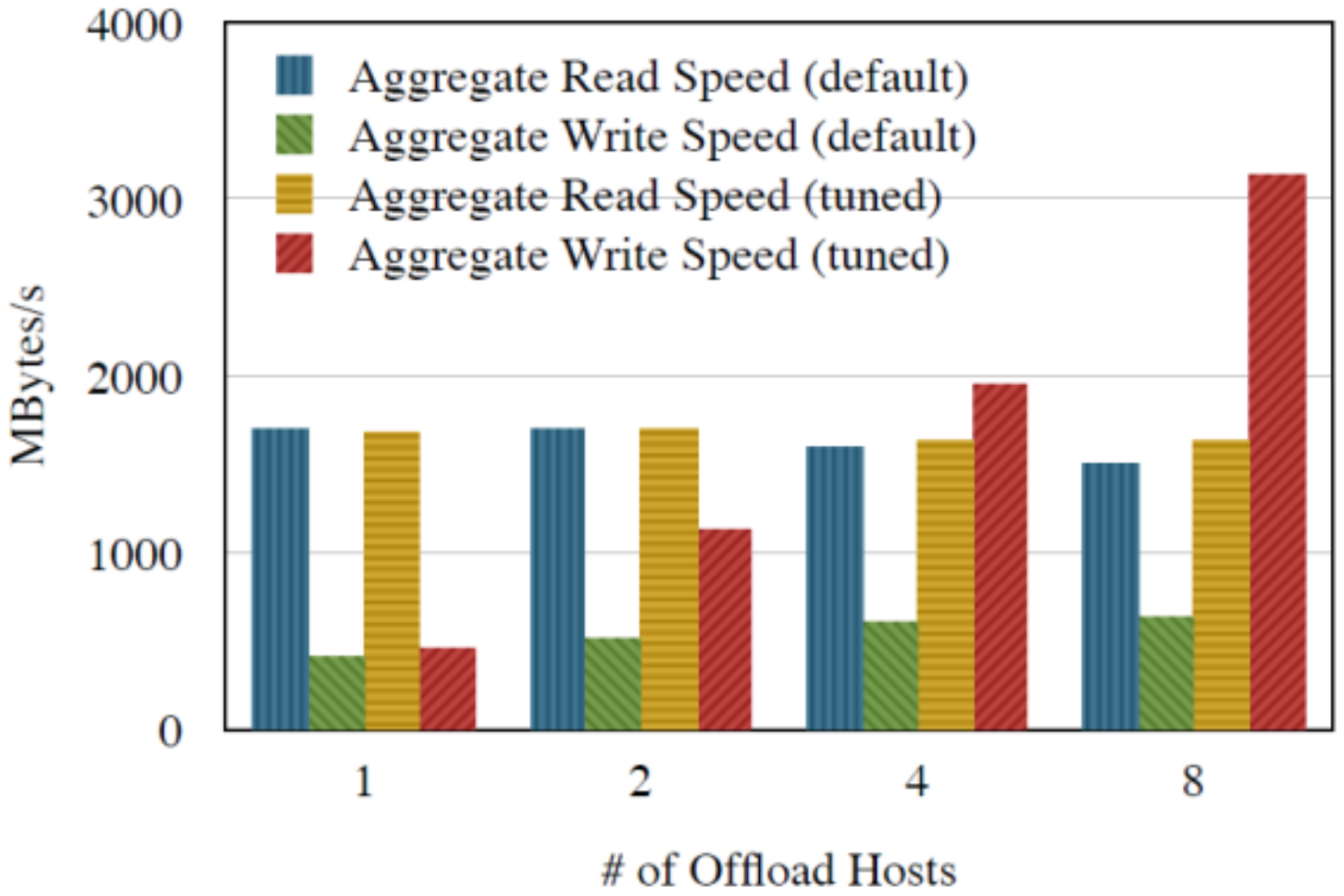
```
Eager vs. rendezvous program details:
- Number of call sites that used MPI_Send: 1
- Maximum median size (bytes) of messages sent
  through MPI_Send: 131072
- Eager threshold of MPI library (bytes): 17408
- For more details on the messages sent,
  consult the mpiP report: ./cfour.88089.1.mpiP

Eager vs. rendezvous suggestions:
- POSSIBLE OPTIMIZATION: The maximum of the
  median messages sent is 131072 bytes, but
  the eager threshold of the MPI Library is
  17408. Consider increasing the eager thres-
  hold to a value higher than 131072 bytes.
- WARNING: Increasing the eager threshold will
  also increase MPI library memory footprint.

MVAPICH2 command that can be used to change the
  eager threshold:
- MV2_IBA_EAGER_THRESHOLD=<nbytes>
- Related documentation can be found in:
  http://mvapich.cse.ohio-state.edu/support/
```

Point-to-Point Protocol Threshold: Demonstration/2

- **Improvement: Running the CFOUR-based benchmark with an increased eager threshold of 256KB resulted in a ~5x improvement for write operations.**



Infiniband Transport : Demonstration/1

- **Application: SMG2000**
  - Parallel semi-coarsening multi-grid solver.
  - Can be run with different node counts.
- **Recommendation: MPI Advisor recommends using UD when over 4K tasks are used.**

MPI Advisor Output

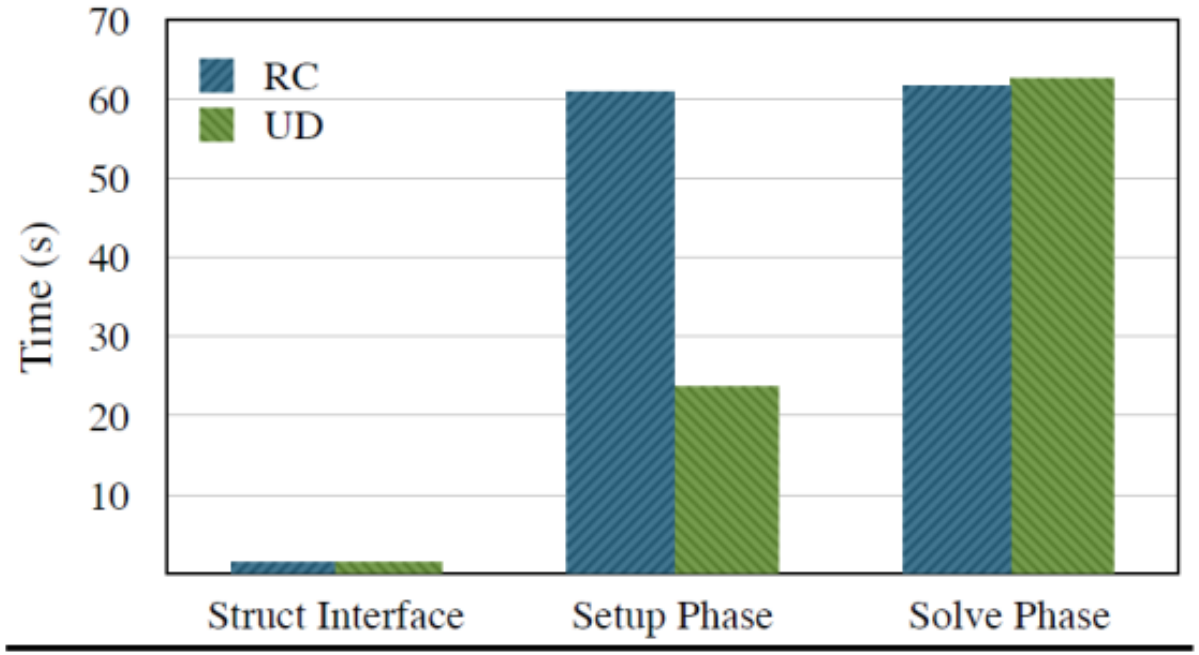
```
Infiniband transport selection details:
- Number of MPI Tasks launched: 4096

Infiniband transport suggestions:
- POSSIBLE OPTIMIZATION: You are using over 4K
  MPI tasks
- Consider using UD instead of RC

Intel MPI variables that can be used to modify
  the Infiniband transport:
- I_MPI_DAPL_UD_PROVIDER=ofa-v2-mlx4_0-1u
- I_MPI_DAPL_UD=enable
- Related documentation can be found in:
  https://software.intel.com/en-us/articles/
  intel-mpi-library-documentation
```

Infiniband Transport : Demonstration/2

- **Improvement:**
  - SMG2000's global performance was improved by 29%.
  - SMG2000's setup phase was improved by 61%.



Transport Method	Struct Interface	Setup Phase	Solve Phase	Total Runtime
RC	1.42	61.04	61.63	124.09
UD	1.44	23.63	62.71	87.78