Application-driven MPI profiling with P^{n}MPI and mpiP

Martin Schulz
Lawrence Livermore National Laboratory

Workshop on
Extreme Scale Performance Tools
held at SC2012, Salt Lake City

LLNL-PRES-603055

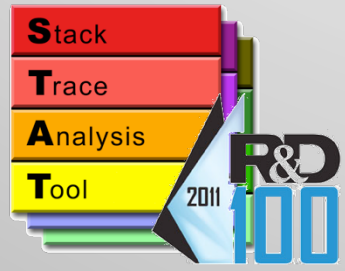
This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

LLNL Tool Contributions to VI-HPS

- Performance Analysis

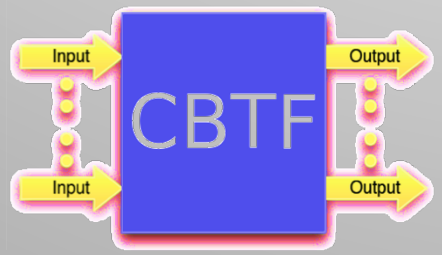
mpiP Open | SpeedShop™

- Debugging / Correctness



AutomaDeD

- Tool Infrastructures



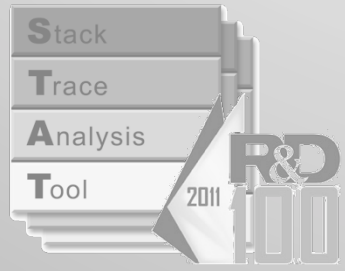
nPMPI

LLNL Tool Contributions to VI-HPS

- Performance Analysis

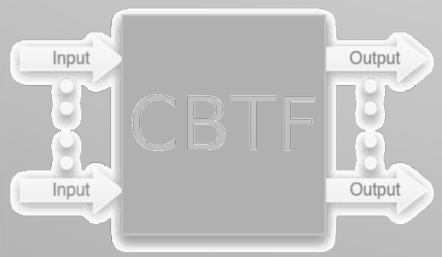
Open | SpeedShop™

- Debugging / Correctness



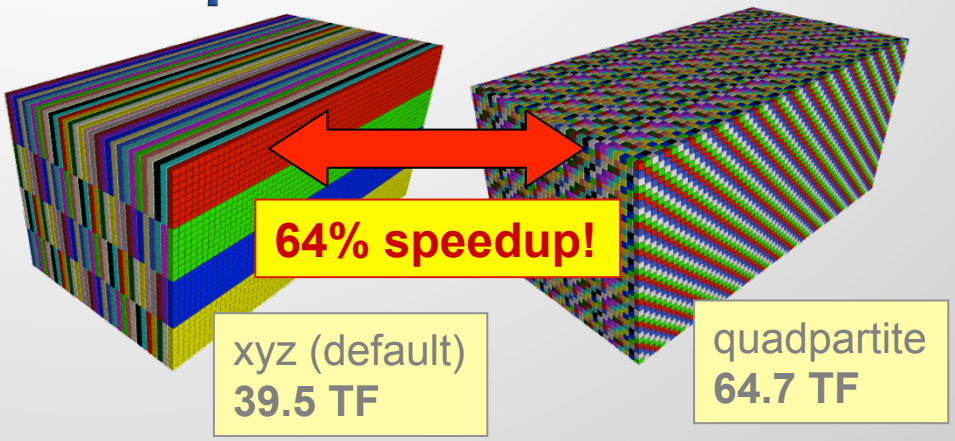
AutomaDeD

- Tool Infrastructures



Understanding Communication Performance is Critical for any Extreme Scale Code Optimization

- **Example: FPMD Code**
  - How to map rows/columns of the underlying 2D matrix?
  - Good node mappings essential for performance
  - Requires understanding of communication patterns



- **Step 1: Communication profiling**

mpiP: Efficient MPI Profiling

- **Open source MPI profiling library**
  - Developed at LLNL, maintained by LLNL & ORNL
  - Available from sourceforge
  - Portable across MPI libraries and system architectures
  - Available on most platforms, incl. IBM BG L/P/Q, Cray XT/XK/XE, Clusters
- **Easy-to-use and portable design**
  - Relies on PMPI instrumentation alone
  - Single text file as output
- **Usage:**
  - Compile application with –g for better accuracy mapping to source
  - Option 1: add libmpip.a/.so to the link line
  - Option 2: set preload variable (e.g., LD_PRELOAD) to libmpip.so

mpiP 101 / Running

```
bash-3.2$ srun –n4 smg2000
mpiP:
mpiP:
mpiP: mpiP V3.1.2 (Build Dec 16 2008/17:31:26)
mpiP: Direct questions and errors to mpip-
help@lists.sourceforge.net
mpiP:
Running with these driver parameters:
  (nx, ny, nz)    = (60, 60, 60)
  (Px, Py, Pz)    = (4, 1, 1)
  (bx, by, bz)    = (1, 1, 1)
  (cx, cy, cz)    = (1.000000, 1.000000, 1.000000)
  (n_pre, n_post) = (1, 1)
  dim             = 3
  solver ID       = 0
=============================================
Struct Interface:
=============================================
Struct Interface:
  wall clock time = 0.075800 seconds
  cpu clock time  = 0.080000 seconds
```

Header

```
=============================================
Setup phase times:
=============================================
SMG Setup:
  wall clock time = 1.473074 seconds
  cpu clock time  = 1.470000 seconds
=============================================
Solve phase times:
=============================================
SMG Solve:
  wall clock time = 8.176930 seconds
  cpu clock time  = 8.180000 seconds

Iterations = 7
Final Relative Residual Norm = 1.459319e-07

mpiP:
mpiP: Storing mpiP output in [./smg2000-p.4.11612.1.mpiP].
mpiP:
bash-3.2$
```

Output File

mpiP 101 / Output – Overview

```
---------------------------------------------------------------
@--- MPI Time (seconds) ---------------------------------------
---------------------------------------------------------------
Task    AppTime    MPITime     MPI%
   0       9.78       1.97    20.12
   1        9.8       1.95    19.93
   2        9.8       1.87    19.12
   3       9.77       2.15    21.99
   *       39.1       7.94    20.29
---------------------------------------------------------------
```

mpiP 101 / Output – per Function Timing

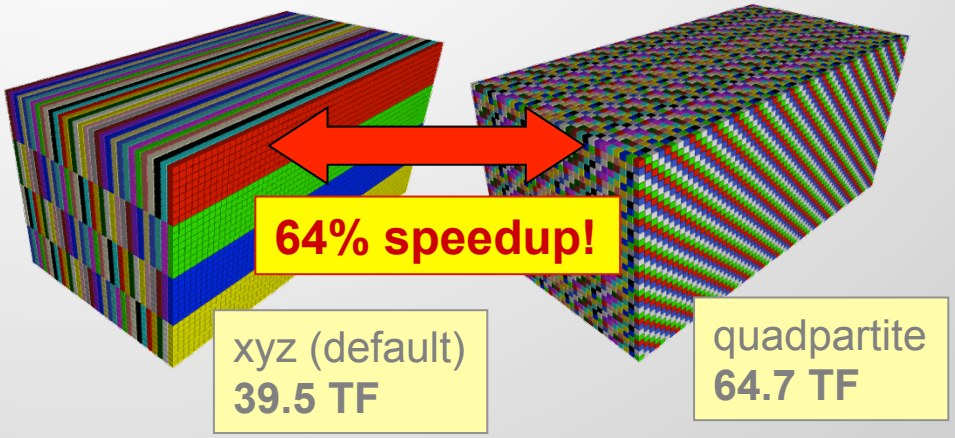
```
---------------------------------------------------------------
@--- Aggregate Time (top twenty, descending, milliseconds) ---
---------------------------------------------------------------
Call                 Site       Time    App%    MPI%      COV
Waitall                10    4.4e+03   11.24   55.40     0.32
Isend                   3   1.69e+03    4.31   21.24     0.34
Irecv                  23        980    2.50   12.34     0.36
Waitall                12        137    0.35    1.72     0.71
Type_commit            17        103    0.26    1.29     0.36
Type_free               9       99.4    0.25    1.25     0.36
Waitall                 6       81.7    0.21    1.03     0.70
Type_free              15       79.3    0.20    1.00     0.36
Type_free               1       67.9    0.17    0.85     0.35
Type_free              20       63.8    0.16    0.80     0.35
Isend                  21         57    0.15    0.72     0.20
Isend                   7       48.6    0.12    0.61     0.37
Type_free               8       29.3    0.07    0.37     0.37
Irecv                  19       27.8    0.07    0.35     0.32
Irecv                  14       25.8    0.07    0.32     0.34
...
```

Fine Tuning a Profile Run

- **Tweak parameters to gain better result and/or reduce data**
  - Default settings provide reasonable first overview
  - Change parameters to pin point more details
- **Options**
  - Stack trace depth
  - Selective profiling using MPI_Pcontrol(…)
  - Reduced output options for large scale runs
- **In mpiP controlled by MPIP environment variable**
  - Set by user before profile run / command line style argument list
  - Example: MPIP = "-c –o –k 4" (stack trace 4, include callsites)

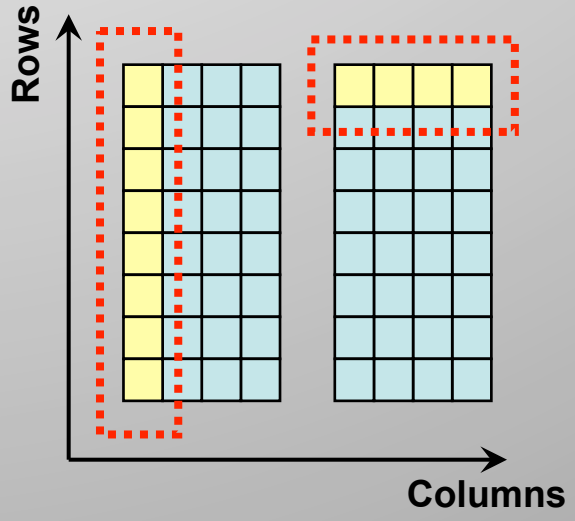
Flat Profiles Can Only Take You so Far

- **Example: FPMD Code**
  - How to map rows/columns of the underlying 2D matrix?
  - Good node mappings essential for performance
  - Requires understanding of communication patterns



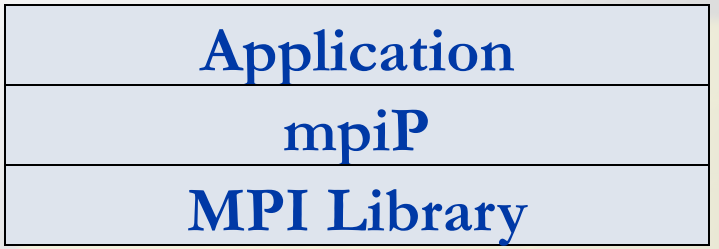
- **Flat profile shows quite balanced execution**

Operation	Sum
Send	317245
Allreduce	319028
Alltoallv	471488
Recv	379265
Bcast	401042



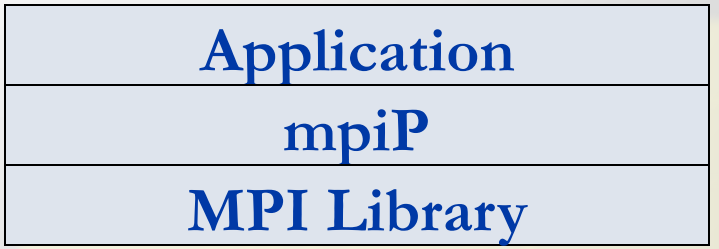
Customizing Profiling with PNMPI

- **Distinguish row/column**
  - MPI_Pcontrol option insufficient
  - Changing profiler too complex



Customizing Profiling with P^{N}MPI

- **Distinguish row/column**
  - MPI_Pcontrol option insufficient
  - Changing profiler too complex
- **P^{N}MPI virtualized PMPI**
  - Multiple tools concurrently
  - Dynamic loading of tools
  - Configuration through text file
  - Tools are independent
  - Tools can collaborate

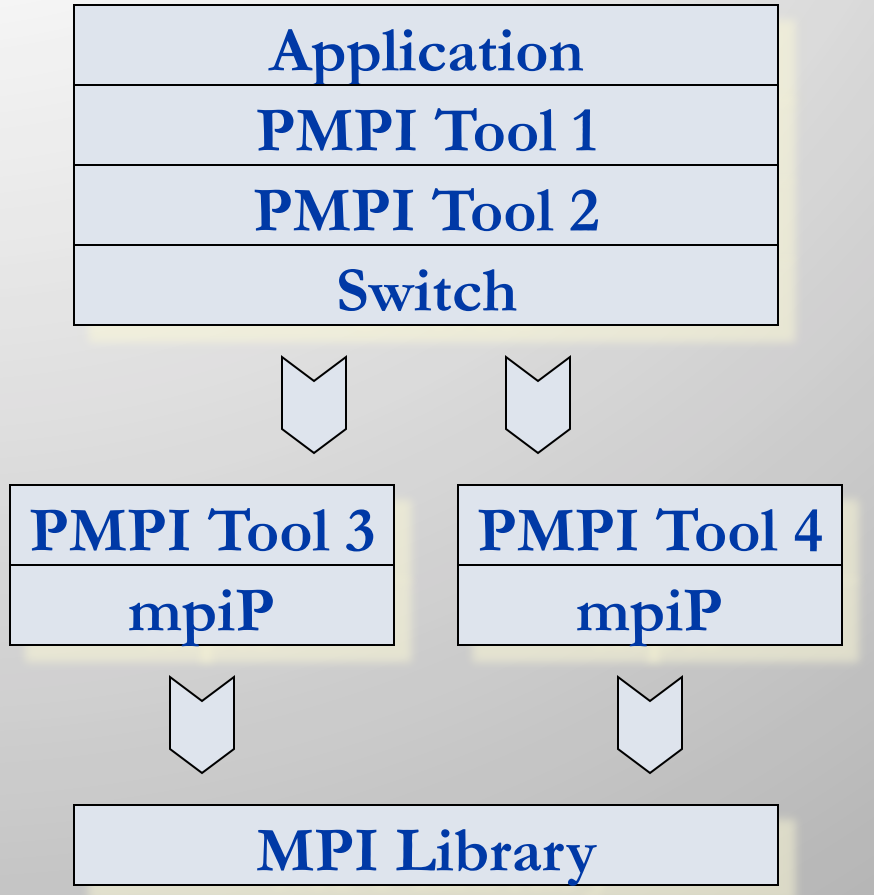


Customizing Profiling with P^{N}MPI

- **Distinguish row/column**
  - MPI_Pcontrol option insufficient
  - Changing profiler too complex
- **P^{N}MPI virtualized PMPI**
  - Multiple tools concurrently
  - Dynamic loading of tools
  - Configuration through text file
  - Tools are independent
  - Tools can collaborate

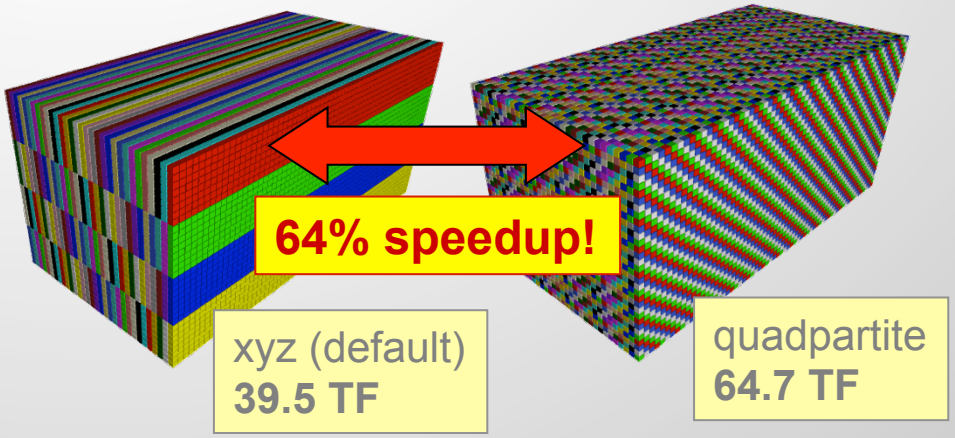
Customizing Profiling with PNMPI

- **Distinguish row/column**
  - MPI_Pcontrol option insufficient
  - Changing profiler too complex
- **PNMPI virtualized PMPI**
  - Multiple tools concurrently
  - Dynamic loading of tools
  - Configuration through text file
  - Tools are independent
  - Tools can collaborate
- **Transparently adding context**
  - Select tool based on MPI context
  - Pick appropriate profiler



Importance of Context Specific Profiling

- **Example: FPMD Code**
  - How to map rows/columns of the underlying 2D matrix
  - Good node mappings essential for performance
  - Requires understanding of communication patterns

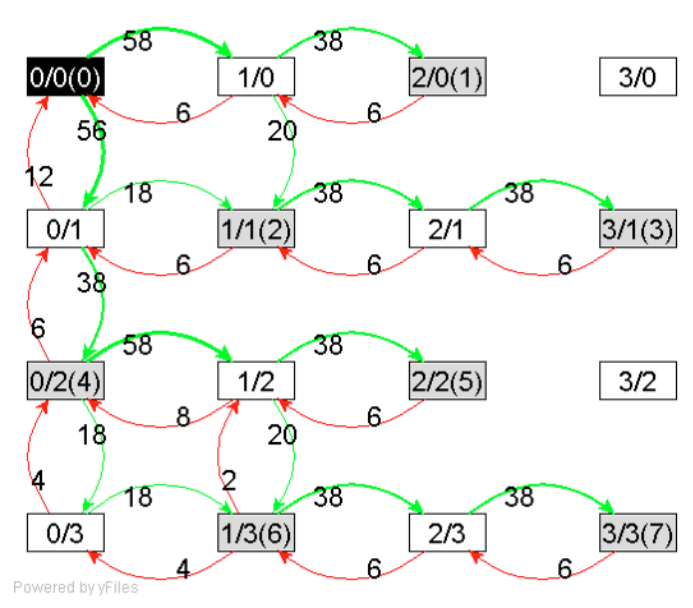


- **Transparently split profiling using P^{n}MPI**
  - Modifying 60+K LOC of a profiler vs. writing 100 LOC for a P^{n}MPI module

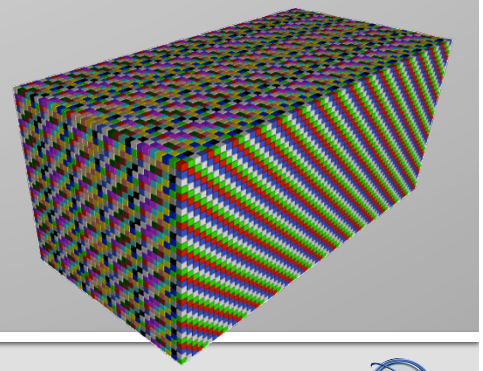
Operation	Sum	Global	Row	Column
Send	317245	31014	202972	83259
Allreduce	319028	269876	49152	0
Alltoallv	471488	471488	0	0
Recv	379265	93034	202972	83259
Bcast	401042	11168	331698	58176

Key Observations to Drive the Optimization

- **Observation 1:**
  - Need balance rows and columns
  - Take interference into account
- **Observation 2:**
  - Need to optimize for Bcast bandwidth
  - Drive as many links as possible

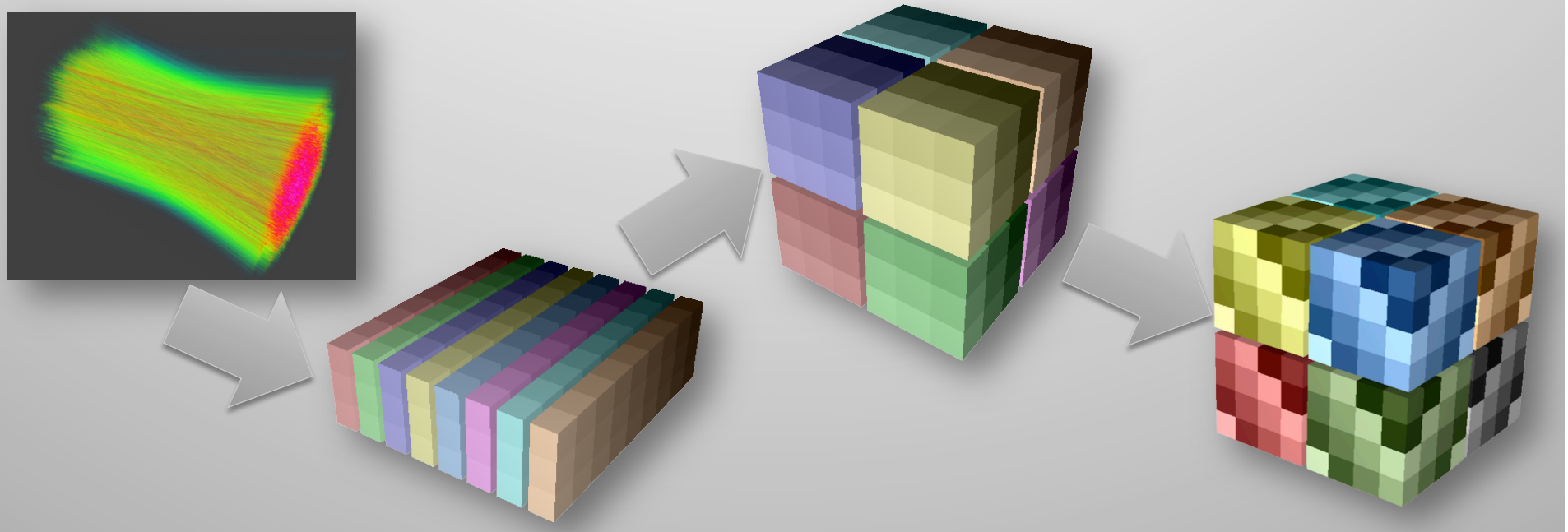


- **Summary and Lessons Learned**
  - mpiP is an easy-to-use and highly portable and scalable MPI profiler
  - Customizing the profiling is essential to understand performance
  - Modifying profiler for each application is impractical
  - PnMPI can help to add transparent context to mpiP
  - Experiments had to be at scale
  - Optimization itself only requires new map files



Rubik: a Flexible Tool to Create Map Files

- **Map files for Cartesian applications to N-dim tori network**
  - Ability to specify application and machine partitions
  - Map one into the other
  - Transformations within partitions to increase bandwidth



http://scalability.llnl.gov/