Open | SpeedShop™

Understanding Performance of Parallel Codes Using Open|SpeedShop on BG/Q

ESP April WorkShop "Code for Q"

May 1, 2012

Jim Galarowicz : Krell Institute

Talk Outline

- **Welcome**

① **Introduction to Open|SpeedShop**

② **Contrast usage models**

  ① Typical Cluster Open|SpeedShop usage

  ② Typical BG/Q Static Executable Open|SpeedShop usage

③ **Compare Facility**

④ **What works now on BG/Q**

⑤ **Future scalability plans**

- **Availability and Contact Information**

OSS Development and Management

- **Jim Galarowicz, Krell**
- **Don Maghrak, Krell**

- **Larger team**
  - William Hachfeld, Krell
  - Dave Whitney, Krell
  - John Ziebarth, Krell
  - David Montoya, David Gunther, Michael Mason, LANL
  - Mahesh Rajan, SNLs
  - Martin Schulz, Matt Legendre, LLNL
  - Dane Gardner, Argo Navis/Krell
  - Dyninst group (Bart Miller, UW & Jeff Hollingsworth, UMD)
  - Phil Roth, ORNL
  - Ciera Jaspan, CMU

Open|SpeedShop Goals

- **Open Source Performance Analysis Tool Framework**
  - Most common performance analysis steps ***all in one tool***
  - ***Extensible*** by plugins for data collection and representation
  - Gathers and displays several types of performance information
- **Flexible and Easy to use**
  - User access through ***GUI***, ***Command Line***, ***Python Scripting, and convenience scripts.***
- **Several Instrumentation Options**
  - All work on ***unmodified application binaries***
  - ***Offline*** and ***online data collection*** / ***attach*** to running codes
- **Supports a wide range of systems**
  - Extensively used and tested on a variety of ***Linux clusters***
  - ***Cray XT/XE*** and ***Blue Gene*** support

Experiment Workflow

Open|SpeedShop Workflow

Application

"Experiment"

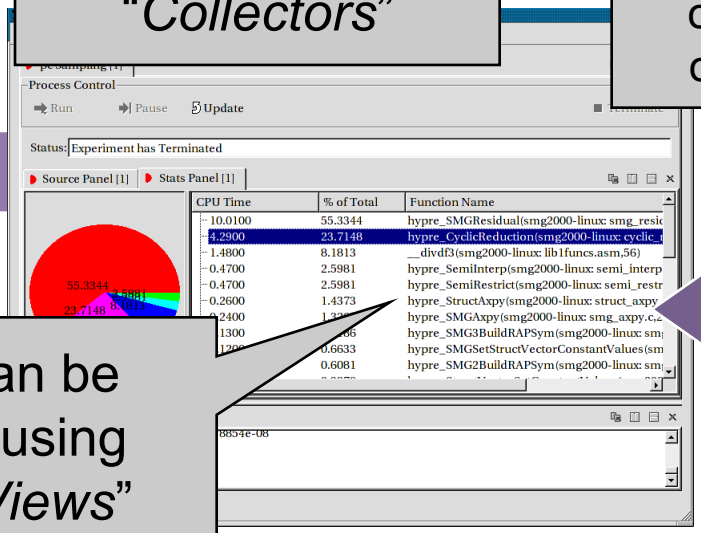
Open | SpeedShop™

Run

Consists of one or more data "*Collectors*"

convenience Scripts
osspcsamp, etc.
or
osslink –c pcsamp
ossutil <raw data>

View



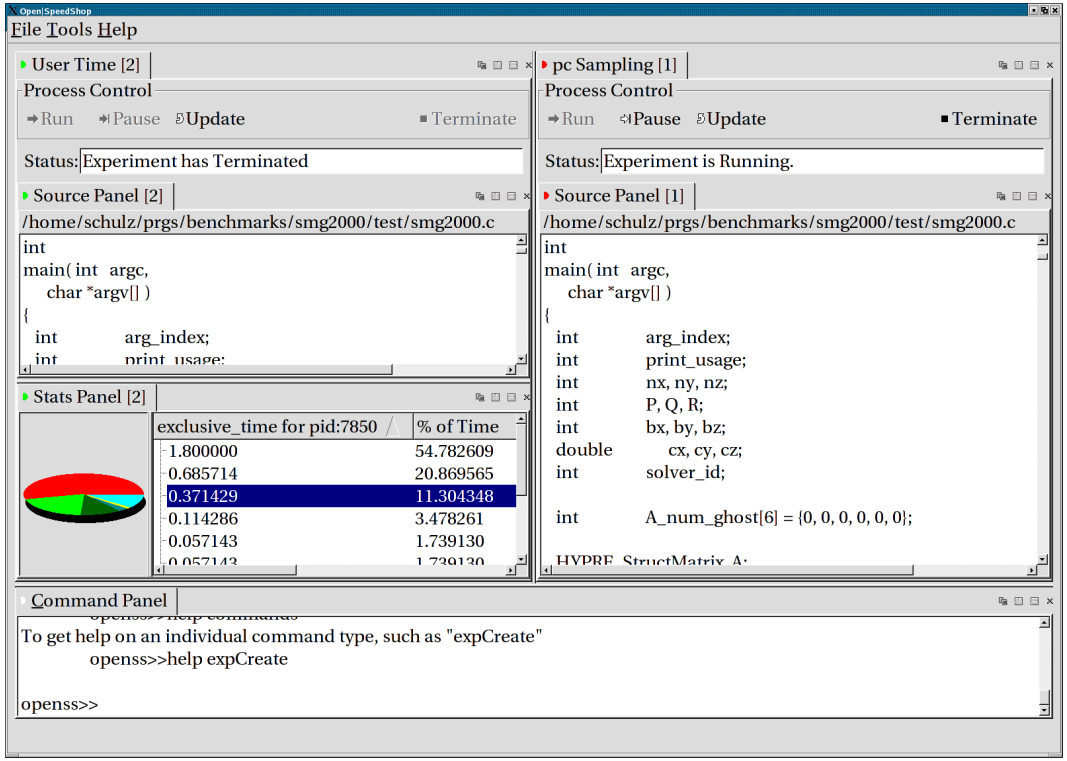
Results

Results can be displayed using several "*Views*"

Stored in SQL database

Basic Interface Usage

- **Step 1**
  - Gather data from command line
  - Example: osspcsamp “<application>” for clusters
  - Re-link and run application for static executables
  - Create database
- **Step 2**
  - Analyze data in GUI
  - Simple graphics
  - Relate data to source
  - openss –f <db file>



Sampling Experiments

- **PC Sampling (pcsamp)**
  - Record PC in user defined time intervals
  - Low overhead overview of time distribution
  - Good first step, lightweight overview
- **Call Path Profiling (usertime)**
  - PC Sampling and Call stacks for each sample
  - Provides inclusive and exclusive timing data
  - Use to find hot call paths, whom is calling who
- **Hardware Counters (hwc, hwctime, hwcsamp)**
  - Access to data like cache and TLB misses
  - hwc, hwctime:
    - Sample a HWC event based on an event threshold
    - Default event is PAPI_TOT_CYC overflows
  - hwcsamp:
    - Sample up to six events based on a sample time (hwcsamp)
    - Default events are PAPI_FP_OPS and PAPI_TOT_CYC
    - Good overview of hardware counter event counts

Tracing Experiments

- **Input/Output Tracing (io, iot)**
  - Record invocation of all POSIX I/O events
  - Provides aggregate and individual timings
  - Provide argument information for each call (iot)
- **MPI Tracing (mpi, mpit)**
  - Record invocation of all MPI routines
  - Provides aggregate and individual timings
  - Provide argument information for each call (mpit)
- **Floating Point Exception Tracing (fpe)**
  - Triggered by any FPE caused by the application
  - Helps pinpoint numerical problem areas

Parallel Result Analysis GUI views

- **Default views**
  - Values aggregated (summed) across all ranks
  - Manually include/exclude individual ranks/processes/threads
- **Comparing ranks**
  - Use the Customize Stats Panel View
  - Create compare columns for process groups or individual ranks
- **Load Balance (min, max, average) across ranks**
  - For hybrid openMP/MPI codes, focus on rank to see load balance across openMP threads
- **Cluster analysis (finding outliers)**
  - Automatically creates groups of similar performing ranks or threads
  - Available from the Stats Panel toolbar or context menu

Typical Cluster: Running a First Experiment

1. **Picking the experiment**
   - What do I want to measure?
   - We will start with pcsamp to get a first overview
2. **Launching the application**
   - How do I control my application under O|SS?
   - Enclose how you normally run your application in quotes
   - **osspcsamp** "mpirun –np 256 smg2000 –n 65 65 65"
3. **Storing the results**
   - O|SS will create a database
   - Name: smg2000-pcsamp.openss
4. **Exploring the gathered data**
   - O|SS will print a default report
   - Open the GUI to analyze data in detail (run: "**openss**")

Typical Cluster: Example Run with Output

❖ osspcsamp "mpirun –np 2 smg2000 –n 65 65 65"

```
osspcsamp "mpirun -np 2 ./smg2000 -n 65 65 65"
[openss]: pcsamp experiment using the pcsamp experiment default sampling rate: "100".
[openss]: Using OPENSS_PREFIX installed in /opt/OSS-mrnet
[openss]: Setting up offline raw data directory in /opt/shared/jeg/offline-oss
[openss]: Running offline pcsamp experiment using the command:
"mpirun -np 2 /opt/OSS-mrnet/bin/ossrun "./smg2000 -n 65 65 65" pcsamp"

Running with these driver parameters:
 (nx, ny, nz)    = (65, 65, 65)
 (Px, Py, Pz)    = (2, 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 = 0.049847 seconds
 cpu clock time  = 0.050000 seconds
```

Typical Cluster: Example Run with Output

- **osspcsamp "mpirun –np 2 smg2000 –n 65 65 65"**

```
==========================================
Setup phase times:
==========================================
SMG Setup:
 wall clock time = 0.635208 seconds
 cpu clock time  = 0.630000 seconds
==========================================
Solve phase times:
==========================================
SMG Solve:
 wall clock time = 3.987212 seconds
 cpu clock time  = 3.970000 seconds
Iterations = 7
Final Relative Residual Norm = 1.774415e-07
[openss]: Converting raw data from /opt/shared/jeg/offline-oss into temp file X.0.openss

Processing raw data for smg2000
Processing processes and threads ...
Processing performance data ...
Processing functions and statements ...
```

Typical Cluster: Example Run with Output

- **osspcsamp "mpirun –np 2 smg2000 –n 65 65 65"**

```
[openss]: Restoring and displaying default view for:
        /home/jeg/DEMOS/demos/mpi/openmpi-1.4.2/smg2000/test/smg2000-pcsamp-1.openss
[openss]: The restored experiment identifier is:  -x 1

 Exclusive CPU time      % of CPU Time  Function (defining location)
      in seconds.
      3.630000000       43.060498221  hypre_SMGResidual (smg2000: smg_residual.c,152)
      2.860000000       33.926453144  hypre_CyclicReduction (smg2000: cyclic_reduction.c,757)
      0.280000000        3.321470937  hypre_SemiRestrict (smg2000: semi_restrict.c,125)
      0.210000000        2.491103203  hypre_SemiInterp (smg2000: semi_interp.c,126)
      0.150000000        1.779359431  opal_progress (libopen-pal.so.0.0.0)
      0.100000000        1.186239620 mca_btl_sm_component_progress (libmpi.so.0.0.2)
      0.090000000        1.067615658  hypre_SMGAxpy (smg2000: smg_axpy.c,27)
      0.080000000        0.948991696  ompi_generic_simple_pack (libmpi.so.0.0.2)
      0.070000000        0.830367734  __GI_memcpy (libc-2.10.2.so)
      0.070000000        0.830367734 hypre_StructVectorSetConstantValues (smg2000:
struct_vector.c,537)
      0.060000000        0.711743772  hypre_SMG3BuildRAPSym (smg2000: smg3_setup_rap.c,233)
```

- View with GUI: openss –f smg2000-pcsamp-1.openss

BG/Q: Running a First Experiment

1. **Picking the experiment**
   - What do I want to measure?
   - We will start with pcsamp to get a first overview
2. **Re-linking and Launching the application**
   - **osslink –c pcsamp** ${CC} -o smg2000-**pcsamp** smg2000.o ${LFLAGS}
   - Pass **OPENSS_RAWDATA_DIR** with path to shared file system location
   - Execute smg2000-pcsamp as you normally would
     - Performance raw data files will be written to OPENSS_RAWDATA_DIR
3. **Creating Database and Storing the results**
   - Running **ossutil $OPENSS_RAWDATA_DIR** will create a database
   - **ossutil** creates X.n.openss but can rename: smg2000-pcsamp.openss
4. **Exploring the gathered data**
   - O|SS will print a default report
   - Open the GUI to analyze data in detail (run: "**openss**")

BG/Q: Example Run with Output

❖ Modify makefile

```
smg2000: smg2000.o
        @echo  "Linking" $@ "... "
        ${CC} -o smg2000 smg2000.o ${LFLAGS}

# Target to create a program counter sampling experiment that
# uses a timer to periodically interrupt the application and record where it is.
smg2000-pcsamp: smg2000.o
        @echo  "Linking" $@ "... "
        osslink -c pcsamp  ${CC} -o smg2000-pcsamp smg2000.o ${LFLAGS}
```

❖ Re-link application

```
make smg2000-pcsamp
Linking smg2000-pcsamp ...
osslink -c pcsamp /bgsys/drivers/ppcfloor/comm/gcc/bin/mpixlc -o smg2000-pcsamp
smg2000.o -L. -L../struct_ls -L../struct_mv -L../krylov -L../utilities -lHYPRE_struct_ls -
lHYPRE_struct_mv -lkrylov -lHYPRE_utilities -lm
```

BG/Q: Example Run with Output

- **Run the application passing OPENSS_RAWDATA_DIR**

  rm -rf /veas-fs0/jgalaro/RAW_PCSAMP_SMG2000
  mkdir /veas-fs0/jgalaro/RAW_PCSAMP_SMG2000
  qsub -A PEACEndStation -t 40 -n 128 --mode c2 --proccount 256 --env BG_SHAREDMEMSIZE=32MB:PAMID_VERBOSE=1:**OPENSS_RAWDATA_DIR**=/veas-fs0/jgalaro/RAW_PCSAMP_SMG2000 /veas-fs0/jgalaro/demos/smg2000/test/smg2000-pcsamp -n 50 50 50

- **Create database file by running ossutil**

  ossutil /veas-fs0/jgalaro/RAW_PCSAMP_SMG2000
  Processing raw data for smg2000
  Processing processes and threads ...
  Processing performance data ...
  Processing functions and statements ..

- **Rename the** X.n.openss **database file, if desired**

- **View with GUI: openss –f smg2000-pcsamp.openss**

- **View with CLI: openss –cli –f smg2000-pcsamp.openss**

Default Output Report View

Performance Data
Default view: by Function
(Data is sum from all processes and threads)
Select "Functions", click D-icon

Toolbar to switch Views

Open|SpeedShop

File Tools

pc Sampling [1]

Process Control

Run Cont Pause Update

Status: Process Loaded: Click on the "Run" button to begin the experiment.

Stats Panel [1] ManageProcessesPanel [1]

I U CL D SL OV LB CA CC Showing Functions Report:

View/Display Choice: Functions Statements Linked Objects

Executables: smg2000 Host: localhost.localdomain Processes/Ranks/Threads:(2) 0 ...

% of CPU Time	Exclusive CPU time in se	% of CPU Time	Function (defining location)
43.060498221	3.630000000	43.060498221	hypre_SMGResidual (smg2000: smg_residual.c,152)
33.926453144	2.860000000	33.926453144	hypre_CyclicReduction (smg2000: cyclic_reduction.c,757)
3.321470937	0.280000000	3.321470937	hypre_SemiRestrict (smg2000: semi_restrict.c,125)
2.491103203	0.210000000	2.491103203	hypre_SemiInterp (smg2000: semi_interp.c,126)
1.779359431	0.150000000	1.779359431	opal_progress (libopen-pal.so.0.0.0)
1.186239620	0.100000000	1.186239620	mca_btl_sm_component_progress (libmpi.so.0.0.2)
1.067615658			
other			

Command Panel

openss>>

Graphical Representation

Statement Report Output View

Performance Data
View Choice: Statements
Select "statements, click D-icon

Open|SpeedShop

File Tools Help

pc Sampling [1]

Process Control

Run Cont Pause Update Terminate

Status: Process Loaded: Click on the "Run" button to begin the experiment.

Stats Panel [1] ManageProcessesPanel [1]

I U CL D S OV LB CA CC Showing Statements Report...

View/Display choice: Functions Statements Linked Objects

Executables: smg2000 Host: localhost.localdomain Processes/Ranks/Threads:(2) 0 ...

% of CPU Time	Exclusive CPU time in sec	% of CPU Time	Statement Location (Line Number)
29.655990510	2.500000000	29.655990510	smg_residual.c(289)
10.320284698	0.870000000	10.320284698	cyclic_reduction.c(1130)
7.591933571	0.640000000	7.591933571	cyclic_reduction.c(910)
7.591933571	0.640000000	7.591933571	cyclic_reduction.c(998)
7.117437722	0.600000000	7.117437722	smg_residual.c(238)
3.795966785	0.320000000	3.795966785	smg_residual.c(287)
3.202846975			
2.491103203			
1.897983393			
other			

Command Panel

openss>>

Statement in Program that took the most time

Associate Source & Performance Data

Double click to open source window

Use window controls to split/arrange windows

Open|SpeedShop

File Tools Help

Terminate

Status: Process Loaded: Click on the "Run" button to begin the experiment.

Stats Panel [1]

Showing Statement: »

Executables: smg2000 Host: localhost.localdomain Processes/

Exclusive CPU time i	% of CPU Time	Statement Location (I
2.500000000	29.655990510	smg_residual.c(289)
0.870000000	10.320284698	cyclic_reduction.c(113
0.640000000	7.591933571	cyclic_reduction.c(910
0.640000000	7.591933571	cyclic_reduction.c(998
0.600000000	7.117437722	smg_residual.c(238)
0.320000000	3.795966785	smg_residual.c(287)
0.270000000	3.202846975	semi_restrict.c(262)
0.210000000	2.491103203	topo_unity_componen

Source Panel [1]

Exclusive C /home/jeg/DEMOS/demos/mpi/openmpi-1.4.2/smg2000/struct_ls/smg_residual.c

```
281                 hypre_BoxLoop3Begin(loop_size,
282                                     A_data_box, start, base_stride, Ai,
283                                     x_data_box, start, base_stride, xi,
284                                     r_data_box, start, base_stride, ri);
285 #define HYPRE_BOX_SMP_PRIVATE loopk,loopi,loopj,Ai,xi,ri
286 #include "hypre_box_smp_forloop.h"
287                 hypre_BoxLoop3For(loopi, loopj, loopk, Ai, xi, ri)
288                 {
289                   rp[ri] -= Ap[Ai] * xp[xi];
290                 }
291                 hypre_BoxLoop3End(Ai, xi, ri);
292             }
293         }
```

0.320000

>> 2.500

0.010000

Command Panel | ManageProcessesPanel [1]

Processes:	Rank
30947	0
30948	1

Process Sets	PID	Rank	Thread
Dynamic Process Set			
All	All		
Disconnected	Disconnected		

Selected performance data point

Controling the Experiment: Parameters

- **A few examples of environment variables that control the rate and what is gathered for each experiment**
  - For pcsamp: OPENSS_PCSAMP_RATE=100, is default 100 samples per second
  - For usertime OPENSS_USERTIME_RATE=35, is default 35 samples per second
  - For hwc, hwctime: OPENSS_HWC_EVENT=PAPI_TOT_CYC, is default PAPI event
  - For hwcsamp: OPENSS_HWCSAMP_EVENTS=PAPI_TOT_CYS,PAPI_FP_OPS
  - For io: OPENSS_IO_TRACED=<all I/O functions>, can specify subset
  - For iot: OPENSS_IOT_TRACED=<all I/O functions> , can specify subset
  - For mpi: OPENSS_MPI_TRACED=<all MPI functions>, can specify subset
  - For mpit: OPENSS_MPIT_TRACED=<all MPI functions>, can specify subset
  - For fpe: OPENSS_FPE_EVENT=<all FPE exceptions>, can specify subset

- **Need to pass the environment variables you want changed on the qsub command, similar to OPENSS_RAWDATA_DIR.**

Comparing Performance Data (1)

- **Convenience Script: osscompare**
  - Compares Open|SpeedShop databases to each other
  - Compare up to 8 at one time
  - Produces side-by-side comparison listing
  - Optionally create "csv" output for input into spreadsheet (Excel,..)
    - export **OPENSS_CREATE_CSV**=1

- **osscompare syntax:**
  - osscompare "db1.openss, db2.openss" [time, percent, <metric name>] [rows=nn] [viewtype=<view type>] [oname=<output_name>]
    - Where <metric name> means metrics that the cli accepts: "–m <metric name>". This means you can compare on a variety of items.
    - Where <view type> equals either: functions, statements, or linkedobjects
    - Where nn is the number of rows of compare output you want to see
    - Where <output_name> is the name of the file containing the comma separated list (csv) output

Comparing Performance Data (2)

osscompare "smg2000-pcsamp.openss,smg2000-pcsamp-1.openss"

openss]: Legend: -c 2 represents smg2000-pcsamp.openss

[openss]: Legend: -c 4 represents smg2000-pcsamp-1.openss

-c 2, Exclusive CPU time in seconds.	-c 4, Exclusive CPU time in seconds.	Function (defining location)
3.870000000	3.630000000	hypre_SMGResidual (smg2000: smg_residual.c,152)
2.610000000	2.860000000	hypre_CyclicReduction (smg2000: cyclic_reduction.c,757)
2.030000000	0.150000000	opal_progress (libopen-pal.so.0.0.0)
1.330000000	0.100000000	mca_btl_sm_component_progress (libmpi.so.0.0.2: topo_unity_component.c,0)
0.280000000	0.210000000	hypre_SemiInterp (smg2000: semi_interp.c,126)
0.280000000	0.040000000	mca_pml_ob1_progress (libmpi.so.0.0.2: topo_unity_component.c, 0)

What works on BG/Q as of 4/26/12

- **Of the performance information experiment types that OpenSpeedShop supports, three are currently working:**
  - pcsamp - Program counter sampling
  - hwc - Hardware counter overflow
  - hwcsamp - Hardware counter sampling
- **Because of call path unwinding issues these experiments are not currently operational:**
  - usertime - Call path profiling, Hot call path detection, inclusive and exclusive CPU time
  - hwctime - Hardware counter overflow with call path profiling
  - io, iot - I/O function tracing experiments
  - mpi, mpit - MPI function tracing experiments
  - fpe - Floating point exception tracing

Future scalability improvements

- Current version of OpenSpeedShop does not scale to the necessary level needed for BG/Q
- We are currently in the process of connecting a scalable tree based network based compute node collection mechanism based on MRNet and CBTF to the OpenSpeedShop front-end client
  - http://ft.ornl.gov/doku/cbtfw/start
- This means the writing of files will be eliminated
- Data will be filtered (reduced or combined) on the way from the compute nodes to the front end tool
- Viewing performance data will be faster
- Initial prototype exists, working version Sept/Oct 2012

Availability and Contact

- **Current version: 2.0.2b3**
  - On cetus: resoft openspeedshop
- **Open|SpeedShop Website**
  - http://www.openspeedshop.org/
- **Open|SpeedShop Forum**
  - http://www.openspeedshop.org/forums/
- **Download options:**
  - Package with install script
  - Source for tool and base libraries
- **Feedback**
  - Bug tracking available from website
  - Contact information on website
  - oss-questions@openspeedshop.org