



Profiling with TAU

Le Yan

User Services HPC @ LSU





2/15/2012





Three Steps of Code Development

- Debugging
 - Make sure the code runs and yields correct results
- Profiling
 - Analyze the code to identify performance bottlenecks
- Optimization
 - Make the code run faster and/or consume less resources









Profiling

- Gather performance statistics during execution
 - Inclusive and exclusive time
 - Number of calls
- Reflects performance behavior of program entities
 - Routines
 - Loops
- Implemented through
 - Sampling: OS interrupts or hardware counters
 - Instrumentation: measurement functions









Profiling with gprof (1)

- gprof is a GNU profiling tool
- How to use
 - Compile the code with "-pg" option
 - Run the program as normal
 - Examine the profile with "gprof <executable>"
 - Or "gprof <executable> <output> | less"









Profiling with gprof (2)

	7]											
	Flat profile:											
	Each sample counts as 0.01 seconds.											
	<pre>% cumulative self time seconds second</pre>		elf	self		total						
			onds	ds calls s		s/call	name					
	100.00	5.71		5.71	1	5.71	5.71	laplace_				
	0.00	5.71		0.00	2	0.00	0.00	initialize_				
		5.71										
Call graph (explanation follows)												
granularity: each sample hit covers 2 byte(s) for 0.18% of 5.71 seconds												
	index	% time	self	childr	ren ca	alled	name					
			0.00	5.71	. 1	1/1	main	[3]				
	[1]	100.0	0.00	5.71	. 1	L	MAIN[1]				
			5.71	0.00	1	1/1	lapl	ace_ [2]				
			571	0 00		 / / 1	 MAIN	· [1]				
	[0]	100.0										
	[2]											
								ialize_ [4]				
			0.00	0.00	2	2/2	set_	bcs_ [5]				
Â												
	Index by function name											
	[1]	$M \Delta T M$			[2]]	laplace_						
	[1] MAIN ■ [4] initialize					set_bcs_						
LSU	[4] INILIAIIZe_				[2] 8	ອີດເ_ກິດຊີ						
CENTER FOR COM												
& TECHNOL	OGY											



2/15/2012





What is TAU

- Tuning and Analysis Utilities
 - Developed at University of Oregon
- Scalable and flexible performance analysis toolkit
 - Performance profiling and tracing utilities
 - Performance data management and data mining
 - Automatic instrumentation through Program Database Toolkit(PDT)
 - Provides an instrumentation API









Availability on LONI and LSU HPC resrouces

- Tezpur and LONI Linux clusters
 - +tau-2.18-intel-11.1-mvapich-1.1
 - +tau-2.18-intel-11.1-mvapich2-1.4
- Philip
 - +tau-2.18-intel-11.1-mpich-1.2.7p1



2/15/2012







How to Use

- Add the softenv key to .soft and resoft
- Compile your code with TAU compiler scripts
 - tau_f90.sh for Fortran, tau_cc.sh for C and tau_cxx.sh for C++
 - The code is instrumented automatically
- Execute the generated executable as normal
 Profile data files: profile.x.x.x
- Analyze/visualize the profiling results with paraprof









Paraprof

- Java-based analysis and visualization tool for performance data
- "pprof" is for text based profile display
- Can work with profile data generated by other profiling tools, e.g. MPIP
- Options
 - -f <file type>: specify type of performance data
 - m: perform runtime monitoring
 - --pack <file>: pack profile data into one file

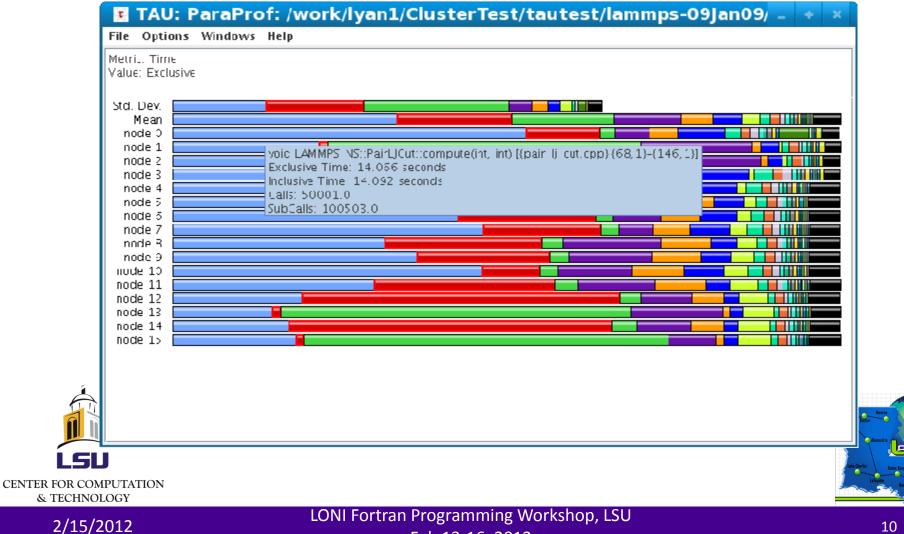








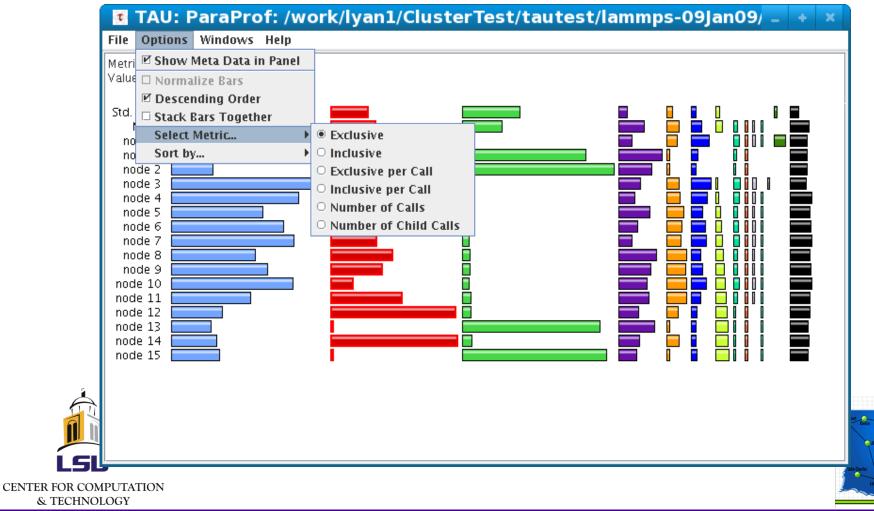
Main Data Window







Main Data Window: Unstacked Bars

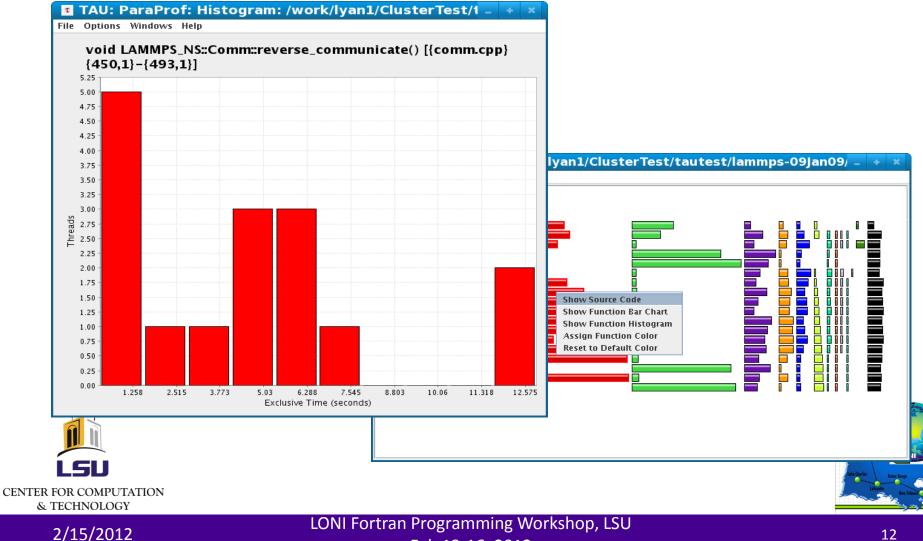


2/15/2012





Function Data Window: Histogram

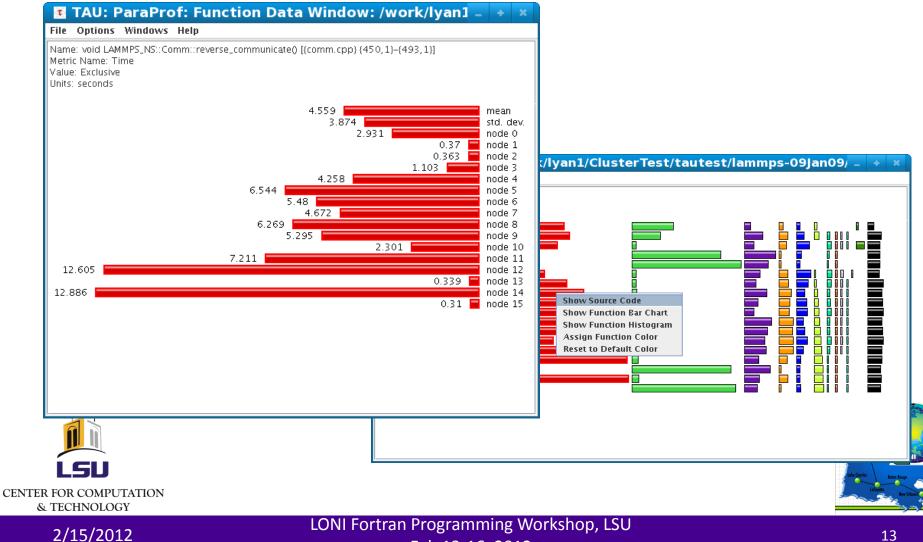


Feb 13-16, 2012





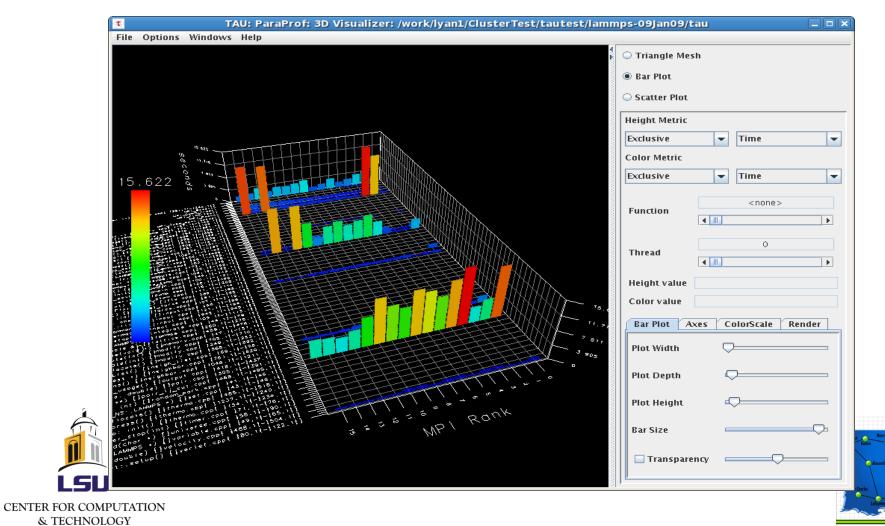
Function Data Window: Bar Chart







3D View







Individual Thread View

TAU: ParaProf: /work/lyan1/C	lust File Options Windows Help
	Metric: Time
Metric: Time /alue: Exclusive	Value: Exclusive
	Units: seconds
Std. Dev.	
Mean Here	
node 0	15.622 void LAMMPS_NS::PairLJCut::compute(int, int) [{pair_
node 1	2.267 MPI_Allreduce()
node 3	2.051 void LAMMPS_NS::Neighbor::half_bin_newton(LAMM 1.244 void LAMMPS_NS::Comm::communicate() [{comm.cp
no(Show Thread Bar Chart	
no Show Thread Statistics Text Window	1.103 🖬 void LAMMPS_NS::Comm::reverse_communicate() [{ 0.735 🔲 void LAMMPS_NS::FixNVE::initial_integrate(int) [{fix_i
not Show Thread Statistics Table	
not Show Thread Call Graph	0.403 🗍 void LAMMPS_NS::FixNVE::final_integrate() [{fix_nve
no Show Thread Call Path Relations	
ode Show User Event Bar Chart	0.214 MPI_Recv0
ode Show User Event Statistics Window	0.207 int LAMMPS_NS::Neighbor::check_distance() [{neigh
ode Show Context Event Window	0.189 MPI_Cart_create()
ode Show Metadata for Thread	0.189 void LAMMPS_NS::Verlet::iterate(int) [{verlet.cpp} {1
ode Add Thread to Comparison Window	0.17 void LAMMPS_NS::Verlet::force_clear() {verlet.cpp}
	0.134 MPL Send()
	0.101 MPLSendrecv()
	0.094 void LAMMPS_NS::FixSetForce::post_force(int) [{fix_s
	0.084 Void LAMMPS_NS::Neighbor::bin_atoms() [{neighbor
	0.084 void LAMMPS_NS::Comm::borders() {{comm.cpp} {6
	0.058 void LAMMPS_NS::Comm::exchange() [{comm.cpp}
Â.	0.051 void LAMMPS_NS::AtomVecAtomic::unpack_reverse
	0.046 int LAMMPS_NS::AtomVecAtomic::pack_comm(int, in
	0.043 void LAMMPS_NS::Timer::stamp(int) [{timer.cpp} {5}
	0.043 void LAMMPS_NS::Timer::stamp() [{timer.cpp} {43,1
	0.037 MPI_Irecv()
LSU	0.035 void LAMMPS_NS::Integrate::ev_set(int) [{integrate.c
	0.025 void LAMMPS_NS::Modify::initial_integrate(int) [{mod
TER FOR COMPUTATION	
& TECHNOLOGY	







Comparing Multiple Threads

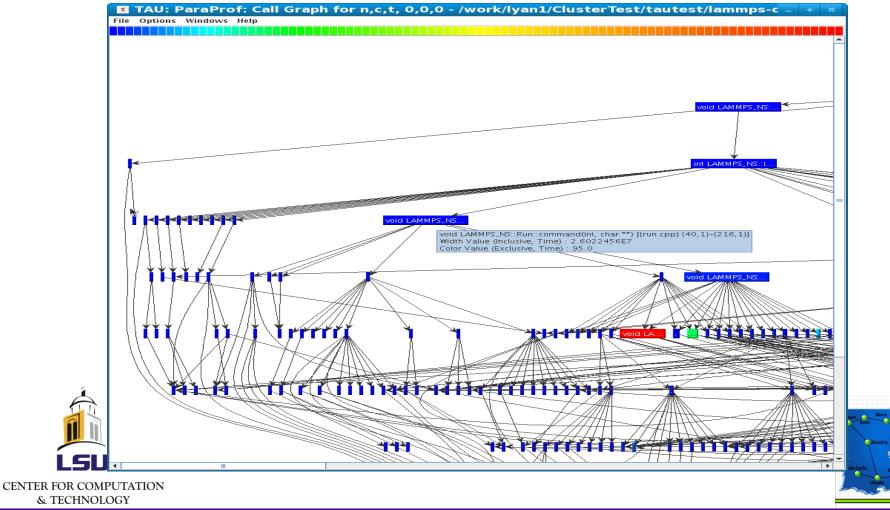
	T	ParaProf: Comparison Window 📃 🔸 🗙
TAU: ParaProf: /work/lyan1/	C File Options Windows Help	
File Options Windows Help Metric: Time Value: Exclusive	Metric: Time Value: Exclusive Units: seconds	tau/lammps-09Jan09/tautest/ClusterTest/lyan1/work/ - node 2 tau/lammps-09Jan09/tautest/ClusterTest/lyan1/work/ - node 3
Std. Dev. Mean Note Note Note Note Note Note Note Note	15.622 (366.584%)	4.261 void LAMMPS_NS::PairLJCut::compute(int, int) [{pair_lj_cut.cpp} {68,1}-{1
node 1	15.346	0.624 (4.063%) MPI_Wait()
noc Show Thread Bar Chart		3.391 MPI_Allreduce()
nor Show Thread Statistics Table		2.051 (379.868%) void LAMMPS_NS::Neighbor::half_bin_newton(LAMMPS_NS::NeighList *) [{r
not Show Thread Call Path Relations note Show User Event Bar Chart note Show User Event Statistics Window		0.33 🔋 void LAMMPS_NS::Comm::communicate() [{comm.cpp} {397,1}-{443,1}]
node Show Context Event Window node Show Context Event Window node Show Metadata for Thread		0.363 🛿 void LAMMPS_NS::Comm::reverse_communicate() [{comm.cpp} {450,1}-{·
Add Thread to Comparison Window		0.198 🚦 void LAMMPS_NS::FixNVE::initial_integrate(int) [{fix_nve.cpp} {63,1}-{103, 0.735 (370.682%) 🗧
	-	0.115 0.403 (348.977%) - void LAMMPS_NS::FixNVE::final_integrate() [{fix_nve.cpp} {107,1}-{140,1]
		0.37 0 0.367 (99.193%) MPI_Init()
		0.178 MPI_Recv() 0.214 (120.127%)
		0.075 0.207 (274.588%) int LAMMPS_NS::Neighbor::check_distance() [{neighbor.cpp} {915,1}-{930
		0.189 MPI_Cart_create() 0.189 (100.003%)
ĹSU		0.186 Noid LAMMPS NS: Verlet: iterate/int) //verlet.com//128.12/222.131
NTER FOR COMPUTATION		
& TECHNOLOGY		







Callpath Profile



2/15/2012





Options for TAU Compiler Scripts

- Display available options with "tau_xxx.sh help"
- Options
 - optVerbose: display verbose debugging information
 - optKeepFiles: keep intermediate files (instrumented source files)
 - -optDetectMemory: trace malloc/free calls









Keeping Intermediate Files (1)

```
[lyan1@poseidon2 single_file]$ 11
total 16
-rwxr-xr-x 1 lyan1 loniadmin 2163 Apr 17 09:23 mat_trans_alt.f90
-rw-r--r-- 1 lyan1 loniadmin 10300 Apr 17 09:50 mat_trans_alt.o
[lyan1@poseidon2 single_file]$ tau_f90.sh -optKeepFiles mat_trans_alt.f90
...
[lyan1@poseidon2 single_file]$ 11
total 1032
-rwxr-xr-x 1 lyan1 loniadmin 1578296 Apr 17 10:18 a.out
-rwxr-xr-x 1 lyan1 loniadmin 2163 Apr 17 09:23 mat_trans_alt.f90
-rw-r--r-- 1 lyan1 loniadmin 2493 Apr 17 10:18 mat_trans_alt.inst.f90
-rw-r--r-- 1 lyan1 loniadmin 10300 Apr 17 10:18 mat_trans_alt.o
-rw-r--r-- 1 lyan1 loniadmin 2019 Apr 17 10:18 mat_trans_alt.pdb
```



2/15/2012







Keeping Intermediate Files (2)

```
[lyan1@poseidon2 single file]$ cat mat trans alt.inst.f90
          ! Matrix dimension
          data ndim /16, 12/
          character(len=*), parameter :: FMT1="(12(1x,i4))"
          character(len=*), parameter :: FMT2="(16(1x,i4))"
              integer profiler(2) / 0, 0 /
               save profiler
               call TAU PROFILE INIT()
               call TAU PROFILE TIMER (profiler, '
          &
             &MATRIXTRANS ALT1 [{mat trans alt.f90} {1,1}-{90,28}]')
             call TAU PROFILE START(profiler)
               call mpi init(ierr)
          call mpi comm size(mpi comm world, nprocs, ierr)
          call mpi comm rank(mpi comm world, myrank, ierr)
CENTER FOR COMPUTATION
   & TECHNOLOGY
                             LONI Fortran Programming Workshop, LSU
    2/15/2012
```

Feb 13-16, 2012



20





Notes for Fortran Programmers

- Use include `mpif.h' instead of use mpi
- If free format is used with .f files, use the `- optPdtF950pts=-R free' option
- If more than one module files are used, use the `-optPdtGnuFortranParser' option
- If C preprocessor directive are used, use the `optPreProcess' option









TAU Environment Variables

- TAU provides many environment variables
 - TAU_MAKEFILE
 - TAU_THROTTLE
 - TAU_OPTIONS
 - PROFILEDIR
 - TRACEDIR









TAU_MAKEFILE

- Different TAU makefiles corresponds to different configurations
- The default is icpc-mpi-pdt-openmp-opari
- There are quite a few others

[lyanl@philip1 lib]\$ ls Makefile.tau-intel-11.1-mpich-1.2.7p1-* Makefile.tau-intel-11.1-mpich-1.2.7p1-callpath-icpc-mpi-compensate-pdt-openmp Makefile.tau-intel-11.1-mpich-1.2.7p1-callpath-icpc-mpi-pdt-openmp Makefile.tau-intel-11.1-mpich-1.2.7p1-depthlimit-icpc-mpi-pdt-openmp Makefile.tau-intel-11.1-mpich-1.2.7p1-icpc-mpi-compensate-pdt-openmp Makefile.tau-intel-11.1-mpich-1.2.7p1-icpc-mpi-pdt-openmp Makefile.tau-intel-11.1-mpich-1.2.7p1-icpc-mpi-pdt-openmp-opari Makefile.tau-intel-11.1-mpich-1.2.7p1-icpc-mpi-pdt-openmp-trace Makefile.tau-intel-11.1-mpich-1.2.7p1-icpc-pdt-openmp Makefile.tau-intel-11.1-mpich-1.2.7p1-icpc-pdt-openmp Makefile.tau-intel-11.1-mpich-1.2.7p1-icpc-pdt-openmp Makefile.tau-intel-11.1-mpich-1.2.7p1-icpc-pdt-openmp Makefile.tau-intel-11.1-mpich-1.2.7p1-icpc-pdt-openmp









TAU_CALLPATH

- Enables callpath profiling
 - Recored callpath for each event
 - Need to set TAU_MAKEFILE to one of those with callpath in their names
- TAU_CALLPATH_DEPTH
 - Level to which callpath is recorded
 - Default is 2
 - Overhead increases with the depth of callpath









Other Environment Variables

• TAU_THROTTLE

- Enable event throttling
- Purpose: reduce profiling overhead
- If a function executes more than \$TAU_THROTTLE_NUMCALLS times and has an inclusive time per call of less than TAU_THROTTLE_PERCALLS microseconds, then profiling of that function will be disabled after the threshold is reached
- PROFILEDIR
 - Controls where the profile files are written to (the default is current directory)
- TAU_OPTIONS
 - Override the default instrumentation options









Selective Profiling (1)

- Instruct TAU
 - Which part(s) of the code to profile
 - How they are profiled
- -optTauSelectFile=<file>
 - The select profiling file specifies files, functions and sections that will be included or excluded in the profiling
 - Wildcards can be used









Selective Profiling (2)

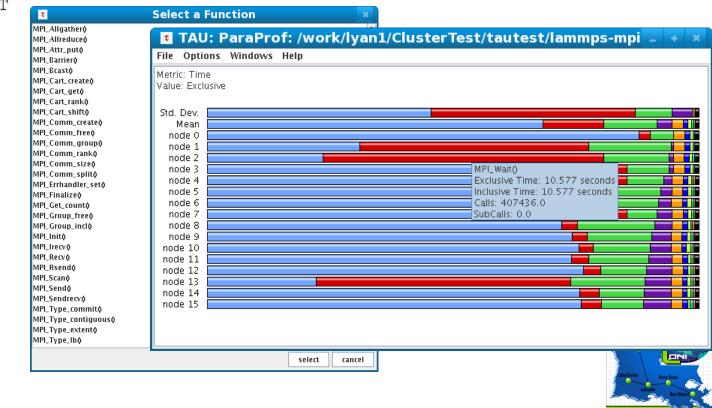
[lyan1@poseidon2 src]\$ echo \$TAU_OPTIONS
-optVerbose -optTauSelectFile=/work/lyan1/ClusterTest/tautest/lammpsmpi-only/src/select.tau

[lyan1@poseidon2 src]\$ cat select.tau

BEGIN_INCLUDE_LIST

MPI#	
mpi#	
Mpi#	

END_INCLUDE_LIST









Tracing (1)

- Recording of information about events during execution
 - Entering/exiting code region (function, loop, block...)
 - Thread/process interactions (send/receive message...)
- Save information in event record
 - Timestamp
 - CPU identifier
 - Event type and event-specific information
- Event trace is a time-sequenced stream of event records









Tracing (2)

- Pick the correct makefile using TAU_MAKEFILE (those with "trace" in the file name")
- Compile with TAU compiler scripts and run the program
- Use external utilities to analyze the trace files
 - JUMPSHOT
 - VAMPIR
- Be careful: trace files can grow very big!







Not Covered

- Database management
- Phase based profiles
- Track memory and I/O
- Instrumentation API









Questions?



2/15/2012



LONI Fortrar