Periscope User's Guide

August 11th, 2009

1. Introduction 1
2. Quick Start 2
1. Installation 2
2. Preparing an analysis run 2
1. Incremental analysis 2
2. Specification of a phase region 3
3. Modify your makefile for instrumentation 3
1. Instrument, compile, and link the application 5
3. Starting an Analysis Run 5
1. Starting the registry 5
2. Starting the Analysis via the Periscope Frontend 5
4. Analyze found properties 6
5. Periscope and batch jobs 6
3. Examples 7
4. Known Issues 7
5. Individual Periscope Components 7
1. Environment Variables 7
2. Periscope Configuration File 8
3. Frontend 8
4. Registry 10
5. High-level Agents 11
6. Analysis Agent 12
7. Periscope Instrumenter F90inst 14
8. psc_instrument 15
9. psc_clean.sh 17

Introduction

Periscope is a scalable automatic performance analysis tool currently under development at Technische Universität München. It consists of a frontend and a hierarchy of communication and analysis agents. Each of the analysis agents, i.e., the nodes of the agent hierarchy, searches autonomously for inefficiencies in a subset of the application processes.

The application processes are linked with a monitoring system that provides the Monitoring Request Interface (MRI). The agents attach to the monitor via sockets. The MRI allows the agent to configure the measurements; to start, halt, and resume the execution; and to retrieve the performance data. The monitor currently only supports summary information.

The application and the agent network are started through the frontend process. It analyzes the set of processors available, determines the mapping of application and analysis agent processes, and then starts the application and the agent hierarchy. After startup, a command is propagated down to the analysis agents to start the search. The search is performed according to a search strategy selected when the frontend is started. At the end of the local search, the detected performance properties are reported back via the agent hierarchy to the frontend. Periscope starts its analysis from the formal specification of performance properties as C++ classes. The specification determines the condition, the confidence value, and the severity of performance properties.

Quick Start

Installation

First insert `module load periscope` into your `.bashrc` file on the Altix and execute `source .bashrc`. Just loading the module on the command line is not sufficient.

Before you first use Periscope, you have to create the configuration file *.periscope* in your home directory. You may copy it from $PERISCOPE_ROOT.

```
cp $PERISCOPE_ROOT/periscope.sample ~/.periscope
```

It should look like:

```
MACHINE          = hlrb2
SITE             = LRZ
REGSERVICE_HOST = hlrb2     //host of registry
REGSERVICE_PORT = 50001     //port of the registry
AGENT_BASEPORT  = 50002     //first port agent hierarchy
APPL_BASEPORT   = 51000     //first port for application
```

You also have to make sure that you can login to nodes of the Altix via a private key. This is essential for startup of the agent hierarchy.

```
1. mkdir ~/.ssh
2. cd ~/.ssh
3. ssh-keygen -t rsa
4. type twice ENTER (for no passphrase)
5. cat id_rsa.pub >> authorized_keys
```

Preparing an analysis run

Incremental analysis

Periscope performs an **incremental analysis**, i.e., it determines performance properties based on measurements, decides on possible new candidate properties and performs a new

experiment to measure those data required to check whether the candidate properties hold. This incremental analysis thus requires execution of multiple **experiments**.

The experiments can be done during the same application run, if a repetitive region is specified as **phase region**. The application is suspended at the end of the phase region, new measurements are requested and the application is released. When the application encounters again the end of the region, it is suspended and the measured values are retrieved.

The experiments can also be done for entire executions of the application. If no phase region is specified, Periscope will automatically **restart the application** to perform new experiments, until no new candidate properties are found and the search terminates.

Specification of a phase region

The phase region can be specified as a user region via directives:

```
!$MON USER REGION
some code
!$MON END USER REGION
```

Modify your makefile for instrumentation

To enable performance measurement, the program has to be instrumented. This is done via a source instrumenter. Therefore, adopt your makefile in the following way:

Replace in the compilation of F90 files the compiler, e.g., `mpif90 <args>`, with `psc_instrument -v mpif90 -c <args>`. Replace also in the link step the compiler with `psc_instrument -v mpif90 <args>`. The "-c" argument will direct the script to instrument and compile instead of linking the application.

You also have to provide a list of those files to be instrumented and the instrumentation requests in `psc_config_inst`. Periscope currently supports only Fortran programs.

Example Makefile:

```
  FC  = mpif90

# Instrument with phase region for faster analysis
  IFC = psc_instrument -s cx.sir -t "user sub loop call" -
v mpif90

# Instrument without phase region for application restart
#  IFC = psc_instrument -s cx.sir -t "sub loop call" -v
mpif90

cx: global.o init.o b_node.o csendxs.o main.o sindex.o
velo.o bound.o curr.o maxv.o temp.o crecvxs.o konst.o
n_node.o testin.o
    $(IFC)   -autodouble  -o $@ *.o

.f.o: global.o
    $(IFC)  -autodouble  -O3  -c $<

clean:
    rm -rf *.o cx cx.sir global.mod compmod inst instmod
prep
```

Example `psc_config_inst` file:

```
#
#  which files are to be instrumented for periscope?
#
#  id  filename      [all sub loop call]  # if any
#
    1  global.f
    2  init.f          sub loop user call
    3  b_node.f        sub loop user call
    4  csendxs.f       sub call
    5  main.f          sub user loop call
    6  sindex.f        sub user loop call
    7  velo.f          sub user loop call
    8  bound.f         sub user loop call
    9  curr.f          sub user loop call
    10 maxv.f          sub user loop call
    11 temp.f          sub user loop call
    12 crecvxs.f       sub call
    13 konst.f         sub user loop call
```

```
14 n_node.f         sub user loop call
15 testin.f         sub user loop call
```

Instrument, compile, and link the application

The instrumented application has to be linked with several libraries. Everything is done automatically after you modified your makefile.

Starting an Analysis Run

An analysis can be executed in interactive and batch mode at HLRB2.

In an interactive job the number of analysis agents is determined according to the frontend parameter `maxcluster`. The number of high level agents results from the maxfan specification.

In a batch job, the number of agents is again computed based on `maxcluster`. For each host (a01..a19) with processors allocated for the batch job one high-level agent is started. All the high-level agents are children of the master agent (the root of the agent hierarchy).

Starting the registry

The Periscope agents and the application processes register with a registry. The registry is started via:

regsrv.ia64 &

The port of the registry will be taken from the environment variable PSC_REGISTRY or from the REGSERVICE_PORT in the configuration file. It will run on the host where it was started.

Starting the Analysis via the Periscope Frontend

Periscope is started via the frontend. It will first contact the registry and then start the application. After all application processes registered with the registry, the agent hierarchy will be started, the analysis agents connect to the application processes and the search starts. The command is:

```
~/psc/frontend/frontend.ia64 --apprun=~/psctest/add/add --
mpinumprocs=4 --strategy=MPI --debug=1
```

--apprun=<command line>	Specify the command line to start the application. It will be passed to the mpirun command.
--mpinumprocs=<np>	Specify number of MPI processes for the application.
--strategy=<strategy>	Specify one of the following strategies: MPI, StallCycleAnalysis, StallCycleAnalysisBreadthFirst.
--debug=<level>	Level of debug output.

Periscope will automatically restart the application for multiple experiments if no phase region is specified, i.e., either there is no user region or it is not instrumented.

Analyze found properties

The frontend will write the properties found into the file properties.psc. This file is in XML format and can be opened with Excel 2007 after it was renamed into properties.xml. A graphical user interface based on Eclipse will be provided soon.

Periscope and batch jobs

Periscope can be used in batch jobs. It is recommended to start a local registry in a batch job to ensure that the registry is running when the batch job is started.

Example batch script:

```
#!/bin/bash
#PBS -j oe
#PBS -S /bin/bash
#PBS -l select=80:ncpus=1
#PBS -l walltime=0:20:00
#PBS -N cx64
#PBS -M gerndt@in.tum.de
#PBS -m e
. /etc/profile
cd psc/test/cx_parallel/
regsrv.ia64 50004&
sleep 10
sudo /lrz/sys/lrz_perf/bin/lrz_perf_off_hlrb2
export PSC_REGISTRY=$HOSTNAME:50004
export PSC_APPL_BASEPORT=52300
 ~/psc/frontend/frontend.ia64 --registry=$HOSTNAME:50004 -
-apprun=cx  --mpinumprocs=64 --maxcluster=16 --
strategy=StallCycleAnalysis --debug=1
```

Examples

You can find two examples with the adapted makefile in `~/psc/test/add` and `~/psc/test/cx_parallel`. Both directories include a file `makefile.psc_instrument`.

Known Issues

- **include <mpif.h> with Altix MPI.** If mpif.h is included in a file with a user region, the code for the instrumentation of the user region is inserted in the declaration part. The problem is a nested include for mpif_parameters.h in the Altix MPI environment. **Solution**: Replace include <mpif.h> with #include 'mpif_parameters.h'

Individual Periscope Components

Environment Variables

PERISCOPE_ROOT	Root directory of the Periscope installation. It includes Periscope's configuration file.
PSC_REGISTRY <hostname>:<port>	Specifies the host and port of the registry service.
PSC_APPNAME	Specifies the name of the application. It is either set by the frontend if it starts the application or can be set by the programmer before starting the application. If it is not set, the default *appl* will be used.
PERISCOPE_DEBUG	0..6 0=quiet 1=startup, found properties in each search 2=candidate properties and found properties in each strategy step 3=details on refinement 4= 5=very detailed info including the values recieved by the agents from the application monitor. 6=individual measurements coming from the application monitoring.

PSC_APP_BASEPORT	It is used by the application monitor and determines the first port used by MPI process with rank 0.
PSC_AGENT_BASEPORT	It defines the port of the frontend and the analysis agents, if it is not specified as command line parameter.

Periscope Configuration File

The configuration of Periscope can be loaded from a configuration file. Its name is `.periscope`. It has to be located in your home directory. The precedence is: command line parameters, environment variables, specification in the configuration file, and finally defaults hardcoded in the program's sources.

REGSERVICE_HOST	Specifies the host of the registry. It is ignored by the registry itself. The host will be the one were the registry is started.
REGSERVICE_PORT	Specifies the port at which the registry is waiting for connections.
APPL_BASEPORT	Specifies the base port for the application monitor. The monitor linked to each process will listen at the baseport+rank.
AGENT_BASEPORT	Specifies the base port for the frontend and the agent hierarchy. The base port will be used by the frontend. The agents will increment the baseport to obtain unique ports.

Frontend

The frontend starts up the application and the agent hierarchy.

--help	Help information
--registry=<Hostname>:<port>	If registry is not specified on the command line, the information is taken from the Periscope configuration file. An error message is generated if it does not exist. Default: Periscope configuration file
--port=<port>	The port to be used by the frontend. It is also used as base port for other analysis agents.

	Default: 30000
--maxfan=<n>	Determines the fan-out of the tree of high-level agents in interactive mode. Default: 4
--maxcluster=<n>	Maximum number of processors (MPI processes * OpenMP threads) analyzed by a single analysisagent. Default: 4
--phase=<fileid:rfl>	Specifies the phase region via the *fileid* and the *region first line* number. If no phase region is specified, a user region is selected if at least one is given in the code. If multiple are given, it is undefined which is selected. If no user region is given, the main program is the user region and the program will be restarted for each strategy step. If you mark the phase region via a user region and would like to use user regions also to guide analysis, you have to give the *fileid* and *rfl* for the phase region.
--appname=<name>	It specifies the application to be searched for in the registry. If the value is defined, it will passed to the application processes via PSC_APPNAME and to the analysis agents via a command line parameter. This variable is set by the frontend. Default: appl<pid> is constructed based on the pid of the frontend process
--apprun=<appl cmdline>	This is the command line used by pbsdsh to start an application process. It should be the same as in `mpirun -np procs <appl cmdline>`.
--ompnumthreads=<n>	Number of OMP threads to be started per MPI process.
--mpinumprocs=<n>	Number of MPI processes to be started.
--timeout=<secs>	Timeout for startup of the agent hierarchy. Default: varying depending on the number of processes

--debug=level	Level of debugging. Default: PERISCOPE_DEBUG or 0
--dontcluster	Online clustering is currently not supported. Passed to master agent.
--strategy=<strategyname>	Strategy used by analysisagent. Currently one of MPI StallCycleAnalysis StallCycleAnalysisBreadthFirst
--sir=<filename>	SIR file of the application to be analyzed. Default: The file name is composed of the executable's name and the extension `.sir`. If --apprun is omitted, the default is appl.sir.
--propfile=<filename>	Specify the file to use when exporting the properties. Default: *properties.psc*
--srcrev=<source revision>	Specify the source code revision. It will be written in the output file.
--delay=<n>	Number of phase executions that are skipped before the search is started. This is useful for applications that have a different behavior at the beginning.

Registry

The registry collects information about the application processes and analysis agents. It is started via `regsrv.ia64&`
The default port is 31337.

Arguments

<port>	Specification of the port to be used. It can also be defined via the environment variable PSC_REGISTRY or via the specification o REGSERVICE_PORT in the PSC configuration file.

Commands

List	Show the entries
Clean	Removes all entries
Help	Shows list of commands
Liststr <id>	Shows strings attached with entry id.
quit	Disconnect

High-level Agents

The root agent and all intermediate agents in the hierarchy are high-level agents. In interactive mode the hierarchy is determined via maxfan and maxcluster. In batch mode for each node a separate high-level agent is allocated.

Arguments

--help	Help information
--registry=<Hostname>:<port>	If registry is not specified on the command line, the information is taken from the Periscope configuration file. An error message is generated if it does not exist. Default: Periscope configuration file
--port=<port>	The port to be used by the agent. Default: 30000
--tag =<tag>	All debug messages and the registry entry are marked by tag.
--parent=<Hostname>:<port>	Port of the parent agent in the agent hierarchy.
--dontcluster	Properties reported to agent are not clustered.
--timeout=<secs>	Timeout for startup of agent hierarchy. Default: 20
--debug=level	Level of debugging. Default: PERISCOPE_DEBUG or 0
--dontcluster	Passed to master agent.

Analysis Agent

The Periscope analysis agent is searching for performance bottlenecks in a subset of the application's MPI processes. It can be started from the hierarchy of agents but also be run as a standalone tool.

If used as a standalone tool, the application has to be running already and the processes have to be registered in the Periscope registry. The tool searches for entries tagged with the application name. It then attaches to those application processes and starts the bottleneck search. The agent itself does not have the ability to restart the application. Therefore a *user region* has to mark an iterative phase of the program.

If the analysis agent is started within the hierarchy, the ids of the processes are passed via a program argument to the agent. It connects to the processes and starts the analysis. If a restart of the application is required to continue the search, a request is propagated to the fronend, the frontend restarts the application and informs the agent of the ids of the same MPI processes. Thus the agent will be responsible for the processes with the same ranks.

Arguments

--help	Help information
--registry=<Hostname>:<port>	If registry is not specified on the command line, the information is taken from the environment variable PSC_REGISTRY or from Periscope's configuration file. If registration is required, i.e., dontregister is not specified, an error message is generated if it does not exist.
--dontregister	Suppresses registration of the agent in the Periscope registry.
--port=<port>	The port to be used by the agent. If it is not specified, the port is taken from PSC_AGENT_BASEPORT or from the configuration file. Default: 30000
--appname=<name>	It specifies the application to be searched for in the registry. Default: appl
--parent=<Hostname>:<port>	High level agent which is the parent of this analysis agent.
--tag=<string>	Tag to be used in debug or error messages. Default: local

--debug=level	Level of debugging. Default: PERISCOPE_DEBUG or 0
--strategy=<strategyname>	Strategy used by the analysis agent. Default: RegionNestingStrategy
--phase=<fileid:rfl>	Specifies the phase region. A detailed description can be found for the frontend.
--sir=<filename>	SIR file of the application to be analyzed. Required.
--threads=<n>	Number of threads for application startup. In standalone mode it is used to instruct the agent to search in this number of threads.
--id=<id1>, <id2>…	List of MPI process ids from the registry. If missing, the agent searches for processes in the registry tagged with the application name.
--searches=<n>	Analysis agent performs this number of successive searches. The results of the searches are compared and additional and missing properties are highlighted.
--propfile=<filename>	Specify the file to use when exporting the properties. Default: *properties.psc*
--srcrev=<source revision>	Specify the source code revision. It will be written in the output file.
--delay=<n>	<n> instances of the phase will be skipped.

Start Analysis Run with a Single Analysis Agent

The analysis can also be done by simply starting a single analysis agent. This is helpful for debugging purposes. The application will have to be started separately via *mpirun*. The entries of the application processes are either passed to the analysis agent or the application name is used to search the registry. The application name is by default *appl* or can be set for the application processes via the environment variable PSC_APPNAME. The analysis agent takes the application name from a program argument, from PSC_APPNAME, or uses the default *appl*.

```
export PSC_APPNAME=add
mpirun -np 4 add

analysisagent.ia64 --appname=add --sir=add.sir  --
strategy=MPI --debug=0
```

```
or

analysisagent.ia64 --sir=add.sir  --strategy=MPI --
id=1,2,3,4
```

Periscope Instrumenter F90inst

F90inst is the source instrumenter. It allows selective instrumentation of OpenMP F90 programs. The instrumentation can be done separately for each source file.

Syntax:

f90inst <options> <file> <file-id> [<region-specifier>]**

Arguments

-f	Source file is in fixed format.
-I <path>	Search path for include files and module files.
-M <path>	Location where modeule files are placed.
-S	Generate SIR file with static program information
-P <string>	Postfix to the file name of the generated file. The default is *_inst*.
-d <n>	n=1: Switch on debug information.
-h	This information.
-i <n>	Switch on information about the instrumentation process. n is the sum of the requested information according to the following table: 1: command line arguments 2: NAGf90 syntax tree 4: NAGf90 symbol and scope table 8: current node number 16: current region 32: Jump addresses and references 64: exception handling 128:OMP and instrumentation directive handling 256: region tree
<file>	File to be instrumented.

<file-id>	file number used to identify the region's position.
<region-specifier>	Specifies the region type to be instrumented. **all**: all regions **call**: call statements **forall**: forall statements **io**: IO statements **loop**: outermost loops only **nestedloop**: non-perfectly nested loops **sub**: subroutines **vect**: vector statements **par**: OMP parallel and worksharing constructs **sync**: OMP synchronization statements **user**: user regions

Instrumentation of User-defined Regions

Single entry and exit program regions can be defined by the user via the monitoring directives.

```
!$MON USER REGION
   S1
   S2
   ...
!$MON END USER REGION
```

User regions are instrumented via the region specifier *user*. Multiple user regions can be specified in the code. If a user region is the phase region, you can omit the specification for the frontend if this is the only user region in the code.

psc_instrument

This command allows to prepare applications for analysis with Periscope. In the existing makefile, the compilation step generating the object files has to be modified such that the compiler is replaced with `psc_instrument`. The script will preprocess the file, instrument it, and finally call the compiler for generation of the instrumented object file. In addition, the compiler has to be replaced in the link step by `psc_instrument`. Here `psc_instrument` will link also the monitoring library to the executable as well as generate the SIR file with the program's static information.

The instrumentation is controlled by a file called `psc_inst_config` in which the file id

and the region types to be instrumented can be determined for each file individually.

```
psc_instrument [-t <regions>] [-s <sir>] [-n] [-v] <compiler>
                  [<options>] <file> [<libs>]
```

psc_instrument will instrument and compile the given file if "-c" is specified in the options list. Otherwise it will link the application.

Arguments:

-t <regions>	List of region types to be instrumented. This overwrites the specification in psc_inst_config. **all**: all regions **call**: call statements **forall**: forall statements **io**: IO statements **loop**: outermost loops only **nestedloop**: non-perfectly nested loops **sub**: subroutines **vect**: vector statements **par**: OMP parallel and worksharing constructs **sync**: OMP synchronization statements **user**: user regions
-n	Dryrun: run the makefile without executing the commands
-v	verbose
<compiler>	compiler for final compilation of the instrumented files, e.g., mpif90
-s <SIR file>	This file name will be used for the static program information. It is recommended to name the sir file according to the executable with an extension `.sir`. Default: appl.sir
<options>	List of compiler options used in the original call to the compiler. These are passed to the compiler.
<file>	Name of the file to be instrumented. File extensions *.f90* and *.F90* determined free source format while *.f* determines fixed source format.
<libs>	Libraries for linking.

psc_clean.sh

When an analysis is not properly terminated, some agents might continue working and so using system resources and possibly interfering with the toolkit. Moreover, some application entries might be left in the registry. These old entries will prevent the next execution of Periscope.
As a workaround, a shell script called `psc_clean.sh` was created. It will terminate all periscope agents, connect to all monitored applications and request their immediate termination, and finally clean the registry. This script is available in the *bin* folder of Periscope and currently does not support any arguments. It reads the user's configuration directly from the *~/.periscope* file.