

ML-RSIM User Manual

Lambert Schaelicke
Department of Computer Science and Engineering
University of Notre Dame

Mike Parker
School of Computing
University of Utah

May/05

1 Introduction

ML-RSIM is an execution-driven cycle-accurate simulator that integrates detailed processor, cache and memory models with a complete I/O subsystem. Combined with the Unix-compatible Lamix operating system, ML-RSIM provides a unique tool that allows researchers to study the interaction of I/O activity, computer architecture and applications. The accuracy of the simulation platform has been validated against an SGI Octane workstation and both the hardware characteristics and the Lamix kernel performance were found to closely match the validation system. This validation increases the confidence that conclusions drawn from experiments with ML-RSIM

ML-RSIM simulates the Sparc V-8 instruction set. Applications that are compiled for this instruction set and are statically linked can in most cases execute both on the simulator and on a native Sparc system without modifications.

The simulator includes detailed models of the following hardware components:

- dynamically-scheduled processor
- split level-1 instruction and data caches, uncached buffer
- unified level-2 cache
- split-transaction coherent system bus
- multi-bank memory controller
- real time clock
- SCSI controller with SCSI bus model
- SCSI disk drive

While the system architecture is representative of a wide variety of modern workstation and server-class system, it does not model any one particular system but instead combines common characteristics of many systems.

The Lamix operating system kernel is modeled after BSD and to a lesser extent after Linux, and is partly based on NetBSD v1.4.3b source code. The system call interface is Solaris compatible, thus applications compiled for the simulator can execute on a native Sparc/Solaris host without modification. Other features of the Lamix kernel include:

- limited process management, including fork, exec and multitasking
- SVR4 signal handling
- basic virtual memory support
- time and timer system call support
- vnode-based file system layer
- host file system imports simulation host file systems into Lamix kernel
- BSD FFS filesystem with statically allocated file cache
- BSD-based device drivers for SCSI adapter and disk, RAIDframe software RAID and CCD

- basic Unix and Internet domain socket support

Simulated disk devices provide persistent storage in files on the simulation host, thus disks created or modified during a simulation can be used for later simulations. The ML-RSIM distribution includes BSD-based utilities to initialize disk labels and file systems, RAID and CCD devices. In addition, initialized disk files for individual disks, RAID-5 sets and CCD sets can be downloaded from the ML-RSIM web site.

2 System Requirements

Generally, successful compilation of the simulator requires a C and C++ compiler, an assembler, a library archive tool, a C++ aware linker and the GNU make utility. The simulator has been tested on the following systems:

- Sun UltraSPARC II/III based systems running Solaris 2.7/2.8 with Sun Workshop compiler 5.0/6.0 and Sun assembler
- SGI Origin/Octane running Irix 6.5 with MIPSpro 7.3 compiler/assembler
- Intel Pentium-III based platform running Redhat, Debian or Mandrake Linux with gcc 2.96

Significantly different systems may require modifications to compiler flags and possibly to source code. In addition, an ELF library is required on Linux hosts.

The Lamix kernel must be compiled for the Sparc V8 instruction set and requires a C compiler, assembler, library archive tool and linker. It has been tested with the Sun Workshop 5.0/6.0 compiler and linker. GNU make is currently required by the makefile. Different compilers such as the GNU tools may require modifications to assembler code and C-code pragmas.

Applications must be compiled for the Sparc V8 or V8plus instruction set and must be statically linked. A variety of compilers has been tested, including the Sun Workshop C, C++ Fortran compilers as well as various GNU compilers.

3 Installation

ML-RSIM can be downloaded from the ML-RSIM web page at <http://www.cse.nd.edu/~mlrsim>, either as one compressed TAR file or as a set of smaller files. To install ML-RSIM, decompress and unpack the downloaded files using one of the following commands as appropriate:

```
tar -xf ml-rsim.tar
gunzip -c ml-rsim.tar.gz | tar -xf -
uncompress -c ml-rsim.tar.Z | tar -xf -
```

If ML-RSIM was downloaded in multiple files, each file is decompressed and unpacked in the same way.

The source code will be unpacked in a *ml-rsim/* directory. The following instructions assume that ML-RSIM is installed in the users home directory, indicated by *~/*.

3.1 Lamix Kernel Compilation

The first step to compile ML-RSIM is to configure and compile the LAMIX kernel, as follows:

```
cd ~/ml-rsim/lamix/arch/lamix_ws/conf
config LAMIX_WS
cd ~/ml-rsim/lamix
make
```

Note that the Lamix kernel must be compiled on a Sparc/Solaris system with the native Sun Workshop C-compiler and assembler, while the makefile requires *gmake*. Upon successful compilation, a binary file *lamix* will exist in the *lamix/* subdirectory. Alternatively, a precompiled Lamix kernel can be downloaded from the web site. Currently, the makefile does not consider header files for its dependency checks. When modifying a header file, the user should remove any object files by running *'make clean'* and then rerunning *make*.

3.2 Simulator Compilation

Next, the simulator binary needs to be compiled as follows:

```
cd ~/ml-rsim/bin
make
```

Object files and libraries can be removed by executing *'make clobber'*, but the automatically generated dependency files should make this action unnecessary in most circumstances. The simulator can be compiled and executed on a variety of systems. Compilation requires a C and C++ compiler, an assembler and *gmake*. For each platform, a subdirectory is created in *bin/* that contains an optimized binary (*mlrsim*) and an unoptimized binary with additional correctness checks and debugging symbols (*mlrsim_d*).

3.3 Simulator Compilation on Linux

Most Linux hosts do not provide the required ELF library by default. If the linker phase fails with a message that *libelf.a* does not exist, the library must be downloaded and installed, either as root using RPMs, or in the users home directory from source code.

If it is desired that the ELF library is installed in the default library directory, download a suitable RPM, for instance from *www.rpmfind.net*. Then install the RPM as usual. e.g.

```
rpm --install <name>
```

If an installation as root from an RPM is not possible or not desired, download the source code from <http://www.stud.uni-hannover.de/~michael/software/>, unpack the source code and follow the instructions in the *INSTALL* file. If installation with root permission in the default library directory is desired, type:

```
./configure
make
make install
```

Otherwise, the library must be installed in the users home directory. In the directory that contains the unpacked libelf source code, type:

```
./configure --prefix=~/.ml-rsim/libelf
make
make install
```

This will install the compiled ELF library in a *libelf/* directory in the *ml-rsim/* root directory. Next, the simulator makefile must be modified to include the libelf installation directory in the library search path. Open *~/bin/Makefile.defs* in an editor, go to the section for Linux hosts and modify the *LIBELFDIR* variable to point to the correct directory. Following these steps, the simulator should compile and link without error messages.

4 Running Simulations

The ML-RSIM distribution includes a number of sample applications that test various aspects of the simulator and kernel. The following example shows how to compile and simulate the *filetest* application.

ML-RSIM/Lamix applications must be compiled for the Sparc V8 or V8plus instruction set and must be statically linked. The generic makefile provided in the *apps/* subdirectory specifies the required compiler and linker flags. This makefile is intended to be included in an applications makefile. It assumes that several subdirectories exist that contain the source code, the compiled object code and the linked executable. Source files, header files, libraries and target executable are specified by the top-level makefile in variables. Alternatively, applications may also be compiled without the provided generic makefile, as long as the abovementioned requirements are fulfilled.

In this example, the *apps/filetest/* directory contains the following subdirectories:

- *src/* contains source code and header files
- *obj/* contains compiled object files and assembler output for debugging purposes
- *execs/* contains the compiled and linked executable

The makefile specifies the variables *HEADERS*, *SRC* and *TARGET* and then includes the generic makefile. To compile the *filetest* application, simply type *make*. Note that like the Lamix kernel, applications must be compiled on a Sparc/Solaris system. Alternatively, a cross-compiler may be used, but this option has not been tested.

When simulating an application, three sets of configuration parameters may be used to control the simulator, kernel and application behavior:

- simulator parameters
By default, these parameters are specified in a *rsim_params* file in the current working

directory. Only parameters that deviate from the default value need to be present in this file. Alternative parameter file names can be specified with the ‘-z’ command line flag. A complete list of parameters and their default values is available in the appendix of the ML-RSIM Reference Manual. The *bin/* subdirectory contains a template *rsim_params* file that lists all configuration parameters and their default values.

- simulator command line parameters

These parameters specify output file names and directories and other global behavior and are listed in the following table.

Parameter	Description	Default
-D <dirname>	input/output directory	current directory
-S <subject>	prefix for input/output files	rsim
-z <configfile>	configuration file path and name	rsim_params
-e <username>	send mail to user when simulation completes	NULL (don't send mail)
-c <cycles>	maximum number of cycles to simulate	+INF
-d b	dump flag; enable bus trace file	NULL (no trace file)
-m <procs>	parallel simulation on up to N processors	off
-n	'nice' process - lower simulator priority	off
-r <i-count>	reset statistics when graduating instruction i-count	0 (never)
-s <i-count>	write statistics and abort simulation when graduating instruction i-count	0 (never)
-t <cycles>	enable debug output after N cycles (must be compiled with COREFILE set)	0
-F <file>	simulation executable path and name	-
-u	enable fast functional units (1 cycle latency)	off
-X	static scheduling	off

For a more detailed description of these command line flags consult the ML-RSIM Reference Manual.

- kernel command line arguments

These arguments specify the behavior of the Lamix kernel, including the application or applications to simulate. The following table summarizes all kernel command line arguments.

Parameter	Description	Default
-root	run user process as root (useful when running system administration tools)	off
-nomount	do not mount simulator disk partitions during startup	on (mount disks)
-mtasync	mount filesystems in asynchronous mode	off

Parameter	Description	Default
-bufcache=N	percentage of physical memory used for buffer cache	200 KB + 5% of main memory
-bufpages=N	number of pages used for buffer cache	none
-bufs	number of buffer structures in buffer cache	none
-input=<file>	file from which to read as standard input	/dev/null
-output=<file>	file to which to write standard output and standard error messages	<executable>.stdout <executable>.stderr
-batch	read commands and arguments from specified file and execute them sequentially	none
-parbatch	read commands and arguments from specified file and execute them in parallel (multiprogrammed)	none
-cwd=<dir>	change to specified directory before starting user processes	none
-noclock	disable clock interrupt handler	off
-ccdconf=<file>	pathname of concatenated device configuration file	ccd.conf
-lfs_cleanerd=<file>	pathname of LFS cleaner daemon executable	lamix/lfs_cleanerd

Any arguments following ‘-F’ are passed to the Lamix kernel, which treats all arguments starting with a dash (‘-’) as kernel arguments and the first non-dashed argument as executable name. Following the executable name, application-specific arguments may be listed as they would in a regular shell.

The general format of a simulator invocation is:

```
mlrsim <sim_args> -F <kernel_args> <application> <app_args>
```

In the filetest example, a simulation may be started as follows:

```
~/mlrsim/bin/mlrsim -D outputs -F -nomount execs/filetest /tmp
```

This command line specifies that all output generated by the simulation should go to the *outputs/* directory. The kernel argument *-nomount* prevents the time-consuming mounting of simulated disks. The *filetest* executable located in the *execs/* directory takes one argument that specifies a location for temporary files.

The simulation produces four output files in the *outputs/* directory. The *log* file contains simulator output, including clock interrupts, warnings and completion messages as well as progress messages generated by the kernel. The statistics file is written at the completion of a simulated application and contains a large number of statistical information. The kernel creates output and error files that correspond to *stdout* and *stderr* on Unix systems. These files contain output and error messages generated by the simulated application. The example application simulates for approximately 18400000 cycles. Simulator progress can be observed in the log file, for instance by running ‘*tail -f outputs/filetest.log*’.

5 Known Limitations

While the ML-RSIM simulator is able to model multiple processors per node in an SMP-like architecture, the current version of the Lamix kernel does not support multiprocessing. Adding multiprocessing capabilities to the Lamix kernel would require careful modifications to the synchronization and locking mechanisms used throughout the kernel.

The Lamix kernel currently supports only Unix sockets and a slightly modified version of Internet sockets. Internet sockets allow applications to communicate only with peers outside the simulation environment. The lack of a network adapter hardware model prevents the use of sockets between individual simulated nodes. Furthermore, many simulated socket system calls are reflected in blocking host system calls and thus stall the simulation. This mechanism allows applications to communicate with applications running outside the simulator, but it does not accurately model the timing and memory effects of socket communication in the simulated systems.

Many operations of the HostFS file system are implemented by calling native system calls on the simulation host. If the simulation host is not a Solaris system, the resulting mismatch between Lamix (Solaris) system call arguments and return values and the native format can lead to unexpected behavior. The simulator translates all arguments and return values into the correct format, but occasionally missing system files (such as */etc/mnttab*) result in unexpected error codes. For instance, in Solaris the *getcwd* library routine scans the mount points in the */etc/mnttab* system file. If the simulation is running on a different system, this file does not exist and the *getcwd* routine fails.

The Lamix kernel does not support dynamic linking. As a result, all simulated applications must be statically linked, but can then be executed both in the simulator and on a native Sparc/Solaris system. Since Solaris does not provide a static socket library, applications can not use socket system calls without linking a Lamix-specific socket library. Since DNS name lookups using *gethostbyname* or similar routines rely heavily on dynamically linked code, the socket library implements its own variants of the routines that rely on a Lamix-specific system call. While this design allows applications compiled for Lamix to perform dynamic host lookups, these applications can not execute on native Solaris hosts.

6 Acknowledgements

ML-RSIM was developed by Lambert Schaelicke and Mike Parker and is derived from RSIM and NetBSD. The processor model is based on the RSIM simulator released by a group at Rice University under the direction of Sarita Adve and is now maintained at the University of Illinois at Urbana-Champaign. Lixin Zhang has contributed the cache and memory controller models. The LAMIX kernel is partly based on NetBSD source code for the SPARC architecture. The networking subsystem has been ported from NetBSD by Branden Moore.

This work was supported in part by the Defense Advanced Research Projects Agency under agreement number N0003995C0018 and F306029810101, by an NSF Research Infrastructure

Grant Number CDA9623614 and by a Graduate Research Fellowship from the University of Utah Graduate School for the academic year 2000/2001.

7 License and Copyright Terms

7.1 Simulator License

Copyright (c) 2002 The Board of Trustees of the University of Illinois and William Marsh Rice University

Copyright (c) 2002 The University of Utah

Copyright (c) 2002 The University of Notre Dame du Lac

All rights reserved.

Based on RSIM 1.0, developed by:

Professor Sarita Adve's RSIM research group

University of Illinois at Urbana-Champaign and William Marsh Rice University

<http://www.cs.uiuc.edu/rsim> and <http://www.ece.rice.edu/~rsim/dist.html>

ML-RSIM/URSIM extensions by:

The Impulse Research Group, University of Utah

<http://www.cs.utah.edu/impulse>

Lambert Schaelicke, University of Utah and University of Notre Dame du Lac

<http://www.cse.nd.edu/~lambert>

Mike Parker, University of Utah

<http://www.cs.utah.edu/~map>

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal with the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimers.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimers in the documentation and/or other materials provided with the distribution.
- Neither the names of Professor Sarita Adve's RSIM research group, the University of Illinois at Urbana-Champaign, William Marsh Rice University, nor the names of its contributors may be used to endorse or promote products derived from this Software without specific prior written permission.
- Neither the names of the ML-RSIM project, the URSIM project, the Impulse research group, the University of Utah, the University of Notre Dame du Lac, nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE CONTRIBUTORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR

OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS WITH THE SOFTWARE.

7.2 Lamix License

Copyright (c) 2002 University of Notre Dame du Lac

Copyright (c) 2002 University of Utah

Copyright (c) 1982, 1986, 1989, 1991, 1992, 1993

The Regents of the University of California. All rights reserved.

(c) UNIX System Laboratories, Inc.

Portions of this file developed by:

Lambert Schaelicke, University of Utah and University of Notre Dame du Lac

<http://www.cse.nd.edu/~lambert>

Mike Parker, University of Utah

<http://www.cs.utah.edu/~map>

All or some portions of this file are derived from material licensed to the University of California by American Telephone and Telegraph Co. or Unix System Laboratories, Inc. and are reproduced herein with the permission of UNIX System Laboratories, Inc.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. All advertising materials mentioning features or use of this software must display the following acknowledgement: This product includes software developed by the University of California, Berkeley, the University of Notre Dame du Lac, the University of Utah and their contributors.
4. Neither the name of the Universities nor the names of their contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDERS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.