The Spring Simulation Testbed - V2
User's Guide

Elizabeth Gene

May 1990

Contents

1 Overview **2**

2 Configuration Interface **2**

2.1 Command Line Options 2

2.2 Menu Bar 3

2.3 Screens 4

2.3.1 Main Window 4

2.3.2 Node Configuration 5

2.3.3 Dispatching Table 5

2.3.4 Local Scheduling Setup/Guarantee Routines 6

2.3.5 Heuristic Functions/Resource Reclaiming Routines 7

2.3.6 Global Scheduling Setup 7

2.3.7 Signal Enabling/Polling Routines 8

2.3.8 Signal Management Routines 8

3 Simulator **8**

4 Animated Run-Time Display **9**

4.1 Command Line Options 10

4.2 Display 10

4.2.1 System View 10

4.2.2 Node View 11

1 Overview

This document is in the directory:

```
freal:~gene/testbed/doc/user/
```

There is a Makefile to create it.

The Spring Simulation Testbed is divided into three modules:

1. Configuration Interface
2. Simulator
3. Animated Run-Time Display

The Configuration Interface is used to set up the desired system configuration and run the Simulator, which produces an event-based log file. This file is then used to drive the Animated Run-Time Display. This document describes in detail how to run the Configuration Interface and the Animated Run-Time Display.

2 Configuration Interface

2.1 Command Line Options

The executable is in the directory:

```
min:~gene/testbed/src/
```

To invoke the Configuration Interface, type

```
testbed [-d display] [(-i,-r) file]
```

The -d flag is used to specify which display to use. If not specified, the DISPLAY environment variable is used. The -i flag is used to specify a configuration file with which to initialize the system, otherwise some (not particularly intelligent) defaults are used. The -r flag is used to specify a configuration file with which to initialize the system, and that the Simulator should be run immediately, bypassing the interface. This flag overrides the -i flag. The next section describes how to create a file of configuration information.

2.2 Menu Bar

- **Run** - Allows the user to run the Simulator with the current configuration.
- **Files** - Allows the user to either load or save a file of configuration information. Before a configuration file can be loaded, it must be created. This is done by going through all the parameter screens and selecting the desired options, then using the *Save configuration file* option to save the file to disk. A whole series of files can be created this way. An already created file can be loaded by using the *Load configuration file* option. This will change the options on the screens to those set when the file was created. You can enter this module with the command line option *-i file*. This will initialize the parameters to those set up when the file was created. You can then modify some of the parameters if desired, save a new configuration file if desired, rewrite the old configuration file if desired, and run the simulation if desired. Note that the file to be read or saved is specified on the Main Window, and that the system will overwrite existing files after checking with the user. The configuration files can also be used with the *-r* command line option.
- **Other parameters** - Allows the user to switch between different screens of parameters. The screens will be described in detail in the next section. More than one option on this menu may map to the same screen. The current screen's options are disabled.

- **Quit** - Allows the user to exit the system without running the Simulator.

2.3 Screens

Each screen contains entries to fill in that specify the configuration of the simulation. If a box with the word *CHOICES* appears next to an entry blank, it is a menu of selections for that entry, and the entry can be filled either by selecting an item off the menu, or typing in one of the selections exactly as it appears on the menu.

2.3.1 Main Window

Basic Information:

- **Configuration file** - The name of the file to use for loading or saving configuration information. (tilde-username won't work)
- **Log file** - The name of the event-based log file. An existing file will be written over.
- **System tick size** - The system can run in any of the ticks on the menu. All system times are converted to the system tick size.
- **Simulation length** - How long is the simulation - in number of ticks.
- **Snapshot interval** - A snapshot is taken of the system in intervals specified by the number of ticks entered here. Currently, guarantee ratios and processor utilizations are computed and written to the log file.
- **Number of nodes** - The number of nodes in the system.

Network Parameters:

- **Network topology** - The system can simulate any of the topologies on the menu.

- **Channel speed** - The network channel speed, in KBits per second.
- **Channel protocol** - Which channel protocol to simulate.

2.3.2 Node Configuration

Each of the items can be specified separately for each node.

- **Number of processors** - The number of system and application processors on the node.
- **System processor ID** - The ID of the processor to use as the system processor on the node. It must be between 1 and the number of processors on the node, but should not be the same as the ID of a processor specified as an application processor in the workload file.
- **Number of STTs** - The number of system task tables on the node. Only tasks on processors that share an STT compete for resources. The processor-STT assigments are set up on the *Dispatching Table* screen. Most likely, the system processor should have its own STT.
- **Workload file** - The path name of the workload file for the node. This must be a file of the format of those created by Fuxing Wang's workload generator.

2.3.3 Dispatching Table

Each divided box in the table is a pair of entries (STT id, dispatching option) for a particular processor on a particular node:

- **STT ID** - The ID of the STT that the dispatcher on this processor uses. It must be between 1 and the number of STTs specified on the *Node Configuration* screen.

- **Dispatching Option** - If 0, tasks are bound to a particular processor. If 1, tasks can be dispatched to any processor whose dispatcher uses the STT.

2.3.4 Local Scheduling Setup/Guarantee Routines

- **Local Scheduling Routines** - Select the desired algorithm to use for scheduling on each node. Make sure you select the correct resource reclaiming routine:
  - For *Heuristic,* the selection for no resource reclaiming is *Start Task at SST.*
  - For *Earliest Deadline First,* the usual selection is *Start Task When Has Earliest Deadline and Resources are Available.*
  - PER = periodicity, can be 0 if not used, or if invoked through the signalling mechanism.
  - WCCT = worst case computation time, used by the (not yet implemented) periodic-task scheduler. (Not to be confused with the periodic task-scheduler)
  - ACT = actual computation time, how long the processor works when running the task, can be 0 if the code takes zero time or if the code consumes time internally.
  - PRE = preemptions, number of allowed interrupts, determines how many times the poller can interrupt the task to run the interrupt handler.
- **Guarantee Routines** - Select the desired guarantee routine for the node. If the local scheduler does not invoke a guarantee routine, the choice does not matter.
- **Overhead** - The basic overhead of the local scheduler, used to calculate how much time it takes. (in microseconds)
- **Cost/Task** - How long the local scheduler takes for each task. (in microseconds)

- **Max tasks** - If doing heuristic scheduling with an O(n) guarantee routine, the maximum number of tasks to try, otherwise ignored.

- **Max h-func** - If doing heuristic scheduling with an O(n) guarantee routine, the maximum number of times to calculate the heuristic function, otherwise ignored.

- **Max backtrack** - If doing heuristic scheduling with backtracking, the maximum number of backtracks, otherwise ignored.

2.3.5 Heuristic Functions/Resource Reclaiming Routines

- **Heuristic Function Routines** - The algorithm to use as a heuristic function in scheduling for the node. If the guarantee routine (if any) does not invoke a heuristic function, the choice does not matter. If the function uses a weight, it can be specified in the *H-func weight* entry.

- **Reclaiming Routines** - Select the routine to use for resource reclaiming on the node. It is called before the dispatcher dispatches a task, and must return the time the task can start, but can have any number of side effects. Make sure you select the correct resource reclaiming routine for the scheduling routine:

  - For *Heuristic,* the selection for no resource reclaiming is *Start Task at SST.*
  - For *Earliest Deadline First,* the usual selection is *Start Task When Has Earliest Deadline and Resources are Available.*

2.3.6 Global Scheduling Setup

- **Global Scheduling Routines** - Select the routine to use for global scheduling on the node. See the description of *Local Scheduling Routines* for a description of the parameters *PER/WCCT/ACT/PRE.*

- **Bidding Routines** - Select the routine to use for bidding on the node if so indicated by the choice of global scheduling routines. (Not implemented) See the description of *Local Scheduling Routines* for a description of the parameters *PER/WCCT/ACT/PRE.*

- **RFB size/RFB time out/Bid threshold/Surplus threshold** - Used for the as yet unimplemented bidding global scheduling routines.

2.3.7 Signal Enabling/Polling Routines

- **Signals** - Select which signals are enabled for each node. See the description of *Local Scheduling Routines* for a description of the parameters *PER/WCCT/ACT/PRE.*

- **Polling Routines** - Enable/Disable the Poller on the node. See the description of *Local Scheduling Routines* for a description of the parameters *PER/WCCT/ACT/PRE.*

2.3.8 Signal Management Routines

- **Signal Selection Routines** - Select the routine to choose a signal to service when there is more than one pending.

- **Signal Request Honoring Routines** - Select the routine to determine if the selected signal is honored.

- **Task Resumption After Interrupt Routines** - Select the routine to determine whether an interrupted task is resumed or terminated.

3 Simulator

When the Simulator is run, two things are happening, one visible, the other not.

1. Several things are written to the screen:

   - At simulation start, a summary of the system configuration is written.
   - At each snapshot time, the simulation time is written.
   - At simulation end, three things are written:

     (a) A list of finished application tasks with important time statistics. These are tallied by *task_collect_stats,* which is called by the dispatcher.

     (b) The tasks on the STTs for application processors on each node are listed. The task status can be *executing, ready* (scheduled, but no attempt has been made to dispatch it), or *pending* (an attempt has been made to dispatch it, but the resource reclaiming algorithm determined that it cannot be dispatched yet).

     (c) A chart summarizing the guarantee ratios on each node. The left half is for tasks arriving externally, the right for tasks arriving from other nodes. After this, there is a system-wide summary.

2. Events are written to the log file.

All the output is produced through the routine *io_event.* Make sure that all new output is produced there also. Note that for scheduling routines without guarantee, the scheduled start time reported for each task at the end of the simulation is not relevant. Neither are the guarantee ratios.

4 Animated Run-Time Display

Copy the log file to the machine with the Animated Run-Time Display module. I will describe the animation as I have set it up, but this is intended only as a jumping-off place for animation. My design is not as flexible as it could be - there is a limit of 5 nodes and 5 processors on each node.

4.1 Command Line Options

The executable is in the directory:

```
freal:~gene/testbed/src/
```

Before running this module, you must execute the following two commands:

```
source /usr/local/dataviews/DVinit
setenv DVDEVICE X:d:128
```

These could be placed in a *.login* file, since they only need to be executed once. To invoke the Animated Run-Time Display, type

```
DVtestbed filename
```

The filename is the name of the event-based log file.

4.2 Display

4.2.1 System View

The display represents the configuration of a network of Spring nodes. On top, there is a stop watch and digital counter displaying the simulation time. Below this are the nodes. Each node has a graph representing the guarantee ratio over time. The nodes are connected to the bus. The connectors flash different colors as follows:

- **White** - Nothing happening.
- **Green** - A new task has arrived.
- **Blue** - A task has arrived from another node.
- **Red** - A task is being sent to another node.

The user has the following options:

- Click on a node - opens a view into that node.
- Click on *PAUSE* - Temporarily suspend the display.
- Click on *CONTINUE* - Continue updating the display.
- Click on *RESTART* - Start display from time 0. You can *restart* while *paused*, but will not see anything happen until you *continue*.
- Click on *QUIT* - Leave the system.

4.2.2 Node View

The display represents one Spring node. On top left, there is a stop watch and digital counter displaying the simulation time. On top right, there is a graph representing the guarantee ratio on the node. Below these are the processors. Each processor has a graph representing the utilization over time. The user has the following options:

- Click on *RETURN* - Return to the system view.
- Click on *PAUSE* - Temporarily suspend the display.
- Click on *CONTINUE* - Continue updating the display.
- Click on *RESTART* - Start display from time 0. You can *restart* while *paused*, but will not see anything happen until you *continue*.
- Click on *QUIT* - Leave the system.