Multiple Run Management Performance

Distributed Runs Design

Bill Oakley, CADSWES

1 Introduction

This document describes the requriements and design for distributed (or gridded) multiple (MRM) concurrent runs in RiverWare. In the distributed approach RiverWare will be invoked on multiple CPUs (or cores), with each invocation simulating a subset of the traces and writing a part of the RDF files. When all simulations have finished the partial RDF files will be combined to create the final RDF files. The distributed approach provides two benefits:

- It "solves" the memory growth problem by not requiring a single RiverWare invocation to simulate more traces than possible before exhausting memory.
- It greatly reduces the time necessary to simulate the traces.

Earlier this year a prototype of the distributed approach was implemented and it successfully demonstrated the approach is feasible - a "controller" process on one computer remotely invoked RiverWare in batch mode on another computer and captured and reported RiverWare's simulation status. The remainder of this document discusses the full implementation - key design goals and assumptions and the distributed architecture and design.

2 Requirements

Reclamation now has the need to run stochastic traces of1200 or more years to reflect paleo (tree-ring derived) data available for planning studies on the Colorado River. Previous attempts at running 1200 runs of 1200 years each resulted in running out of computer memory at about 900 runs. By distributing the runs to different processors, it is possible to accomplish all the runs and to reduce the entire runtime.

MRM distributed concurrent runs will allow a user to distribute simulations across multiple computers, with each simulation performing a subset of the traces. When all simulations have finished the partial RDF file will be combined into the final RDF files. In slightly more detail:

- The RiverWare MRM editor dialog will be enhanced to allow a user to specify that concurrent runs should be distributed. If they are, a new "Distributed Runs" tab will be available, in which a user can fully configure the distributed runs.
- When a user starts a distributed run RiverWare will invoke a new "manager" executable which will start the distributed simulations and monitor their status, display their status in its user interface, and combine the partial RDF files into the final RDF files.

3 Design Goals And Assumptions

There are two key design goals:

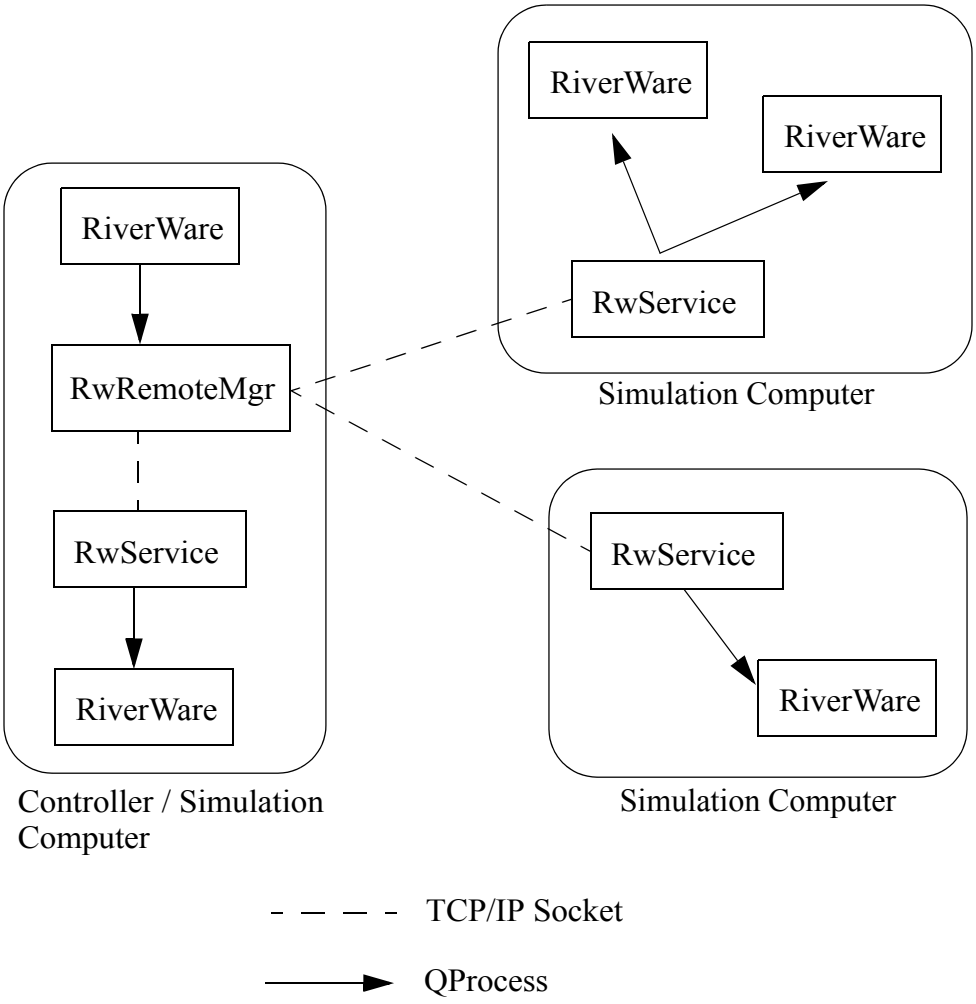
1. To the extent possible performing the concurrent runs in the distributed approach will be identical to performing them on a single computer.

2. Including a computer in the distributed approach will be “minimally invasive,” ideally requiring only that RiverWare be installed on it (not requiring setting environment variables, downloading files, etc.).

The following key assumptions are made:

- The distributed approach will only be available for paired concurrent runs. (With paired concurrent runs, index sequential runs and input DMIs are paired rather than “multiplied,” with each pair representing a trace.) CRSS uses paired concurrent runs and this assumption will simplify development. (Later enhancements could be done to remove the Index Sequential assumption, which does not always make sense.)
- The computers included in the distributed approach will have access to two network directories - the model directory and an intermediate directory. The model directory is equivalent to the CRSS_DIR environment variable, while the intermediate directory is where the individual simulations will write the partial RDF files.
- The initial simulations will be performed at CADSWES, and the design reflects the CADSWES network (Windows clients which access a Solaris server via Samba). A future revision could address other network configurations; elements of the design which might be affected are identified.
- Although the intent is for the distributed approach to be platform-neutral, the target platform is clearly Windows. If it becomes necessary to write platform-dependent code, the Solaris code may be deferred.
- CRSS doesn’t use the per-trace output DMI and its support in the distributed approach may be deferred.

4 Distributed Architecture



In the distributed architecture there is a “controller” computer and one or more “simulation” computers; notice that:

- The controller computer can also be a simulation computer.
- A simulation computer with adequate capacity can support multiple simulations.

The distributed architecture introduces three new processes - the RiverWare Remote Manager (RwRemoteMgr), RiverWare Service (RwService), and the RiverWare Service Controller (RwSvcCtlr, not shown).

4.1 RiverWare Service

The RiverWare Service runs on each simulation computer; it’s a Windows Service which runs in the background, listening to the network for incoming connections (over a TCP/IP socket). Each connection represents a simulation;

the Service reads the simulation's XML configuration from the socket, creates a batch script file, and invokes RiverWare in batch mode. The Service reads RiverWare's output and writes it to the socket.

The RiverWare Service is generic - it reads an XML configuration, creates a batch script file and invokes RiverWare in batch mode. The XML configuration can define any batch mode invocation.

4.2 RiverWare Service Controller

The RiverWare Service Controller controls the RiverWare Service - it allows a user to install, start, stop, pause, resume, and uninstall the Service. Once installed the Service is a "real" Windows Service and can also be controlled through the Windows Service interface. The Service must be installed and started (and not paused) to receive incoming network connections.

4.3 RiverWare Remote Manager

The RiverWare Remote Manager runs on the controller computer. It parses an XML configuration file which defines the simulations and:

- Creates an XML configuration for each of the simulations.
- Configures its user interface (a dialog which shows the status of the simulations).
- For each simulation, connects to the simulation computer (over a TCP/IP socket), writes the XML configuration to the socket, and reads RiverWare's output from the socket (which it uses to update its status dialog).
- When all simulations have finished, combines the partial RDF files to create the final RDF files.

5 XML Configurations

Previous sections have referred to XML configurations. There are three distinct XML configurations which exist as top-level elements in an XML document; the configurations are identified by their names and can coexist within a document:

- "distrib" identifies an distributed concurrent run (hereafter referred to as the "distributed configuration").
- "RW" identifies a RiverWare batch mode invocation (hereafter referred to as the "RiverWare configuration").
- "GenApp" identifies a generic application invocation.

For example, an XML document which defined a distributed concurrent run, two RiverWare batch mode invocations and a generic application invocation would have top-level elements:

```
<document>
 <distrib>
  ...
 </distrbv>
 <RW>
  ...
 </RW>
 <RW>
  ...
 </RW>
```

```
 <GenApp>
  ....
 </GenApp>
</document>
```

Hopefully this illustrates the flexibility of the architecture.

5.1 Distributed Configuration

The distributed configuration document is created by RiverWare when a user starts a distributed concurrent run. DistribMrmCtlr parses the distributed configuration and creates the RiverWare configurations for the simulations. Some elements are from the MRM configuration, the others are provided by RiverWare. Key elements, preceded by brief descriptions, are:

<distrib>

If required, the login information from the MRM configuration. Missing user or password is prompted for.

<login user=”” **passwd**=””**/>**

The RiverWare executable which started the distributed concurrent run. The assumption is that all simulation computers have this executable installed.

<app>C:\Program Files\CADSWES\RiverWare 5.1.6 Patch\riverware.exe**</app>**

The model loaded when the user starts the distributed concurrent run. The assumption is that all simulation computers can access the model using the same path.

<model>R:\\CRSS\\model\\CRSS.mdl**</model>**

The config *attribute is the MRM configuration selected when the user starts the distributed concurrent run. The* mrm *elements are from the MRM configuration and identify the individual simulations - the simulation computer and the traces to simulate. Although the* mrm *element supports per-simulation-computer port numbers, the assumption is that all simulation computers use the same port number (and the MRM configuration dialog allows a single port number to be entered).*

<mrmlist config=”Powell Mead 2007 ROD Operations”**>**
<mrm addr=”lc1.usbr.gov” **port**=”27285” **firstTrace**=”1” **numTrace**=”200”**/>**
<mrm addr=”lc2.usbr.gov” **port**=”27285” **firstTrace**=”201” **numTrace**=”300”**/>**
</mrmlist>

The rdflist *element is a list of the final RDF files, while the* slotlist *element is a list of the slots which are written to the RDF files. Slots can be written to multiple RDF files; they’re associated with the RDF files by the* idxlist *attribute, whose value is a comma-separated list of RDF file indices. RiverWare initializes the RDF DMI and mines its data structures to generate* rdflist *and* slotlist*.*

<rdflist num=”2”**>**
<rdf name=”R:\\CRSS\\results\\Res.rdf” **idx**=”0”**/>**
<rdf name=”R:\\CRSS\\results\\Salt.rdf” **idx**=”1”**/>**
</rdflist>
<slotlist>
<slot name=”Powell.Outflow” **idxlist**=”0,1”**/>**
<slot name=”Powell.Storage” **idxlist**=”0”**/>**
</slotlist>

The envlist *element specifies RiverWare’s runtime environment; RIVERWARE_HOME is from the version of RiverWare which starts the distributed concurrent run, all others are from the MRM configuration.*

<envlist>
<env>RIVERWARE_HOME_516=C:\\Program Files\\CADSWES\\RiverWare 5.1.6 Patch**</env>**
<env>CRSS_DIR=R:\\CRSS**</env>**

</envlist>

The tempdir *element is from the MRM configuration and is the intermediate directory where the individual simulations write the partial RDF files.*

<tempdir>R:\\CRSS\\temp**</tempdir>**

</distrib>

5.2 RiverWare Configuration

The RiverWare configuration defines a RiverWare batch mode invocation. In a distributed concurrent run DistribMrmCtlr creates the RiverWare configuration; in other contexts a user could create the RiverWare configuration. This example shows a RiverWare configuration from a distributed configuration; not all elements are valid in other contexts. Some elements are from the MRM configuration, the others are provided by DistribMrmCtlr. Key elements, preceded by brief descriptions, are:

<RW>

The simulation computer, from the distributed configuration's mrm element.

<host addr=”lc1.usbr.gov” **port**=”27285”**/>**

RiverWare's executable, from the distributed configuration's app element.

<app>C:\Program Files\CADSWES\RiverWare 5.1.6 Patch\riverware.exe**</app>**

Creates the batch script file; the name *attribute is provided by DistribMrmCtlr, with each individual simulation having a unique name.*

<script name=R:\\CRSS\\temp\\script0.rcl”**>**

Adds the OpenWorkspace command to the batch script file.

<openws>R:\\CRSS\\model\\CRSS.mdl**</openws>**

Adds the StartController command to the batch script file; firstTrace, numTrace and ctlFile are new StartController !MRM options. The config *attribute and* firstTrace *and* numTrace *elements are from the distributed configuration. The* ctlFile *element's value is provided by DistribMrmCtlr, with each individual simulation having a unique name. DistribMrmCtlr creates the control file from the distributed configuration's* rdflist *and* slotlist *elements.*

<start>
<mrm config=”Powell Mead 2007 ROD Operations”**>**
<firstTrace>1**</firstTrace>**
<numTrace>200**</numTrace>**
<ctlFile>R:\\CRSS\\temp\\control0.ctl”**</ctlFile**>
</mrm>
</start>

Adds the CloseWorkspace command to the batch script file.

<close/>
</script>

Specifies RiverWare's output file; the value is provided by DistribMrmCtlr, with each individual simulation having a unique name. The value is used with the new “--remout” option, described below.

<output>R::\\CRSS\\temp\output0.log**</output>**

RiverWare's runtime environment, from the distributed configuration.

<envlist>
<env>RIVERWARE_HOME_516=C:\\Program Files\\CADSWES\\RiverWare 5.1.6 Patch**</env>**
<env>CRSS_DIR=R:\\CRSS**</env>**
</envlist>
</RW>

5.3 GenApp Configuration

```
<GenApp>
  <app>C:\\Perl\\bin\\perl.exe</app>
  <arglist>
    <arg>R:\\CRSS\\bin\\CombineRdf.pl</arg>
    <arg>-o</arg>
    <arg>R:\\CRSS\\results\\Res.rdf</arg>
    <arg>R:\\CRSS\\temp\\Res.*.rdf</arg>
  </arglist>
  <output>%scratchfile%</output>
  <envlist>
    <env>CRSS_DIR=R:\\CRSS</env>
  </envlist>
</GenApp>
```

5.4 Configuration Output Options

Both the RW configuration and the GenApp configuration specify “output” files - for RW configurations it’s the script file, the control file and the output file; for GenApp configurations it’s the output file. If the controller computer and the simulation computers have access to the same network directories then the files can be created in a network directory by naming them, e.g.:

<output>R::\\CRSS\\temp\output0.log**</output>**

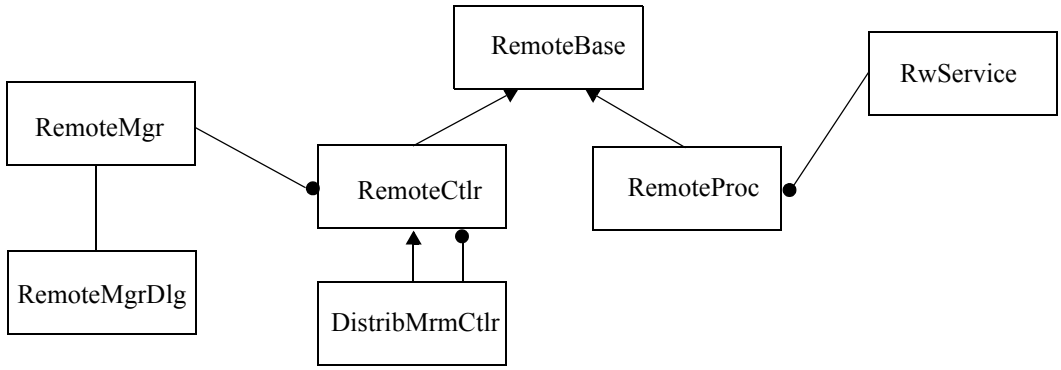
If they don’t have access to the same network directories then the files must be created on the simulation computer, and it’s possible the controller computer (where the files are specified) won’t know where to create the files on the simulation computer. To accommodate this, “output” files can have the following special values:

- %tempfile% - The file is a temporary file which persists.
- %scratchfile% - The file is a temporary file which is removed.
- %devnull% - No file is created (the equivalent of redirecting a process’s output to /dev/null).

A temporary file is a guaranteed unique file in a temporary directory. (The temporary directory might vary, but on Windows it’s commonly C:\WINDOWS\Temp.)

6 Software Architecture

The main classes are RemoteBase, RemoteCtlr, DistribMrmCtlr, RemoteProc, RemoteMgr and RemoteMgrDlg:



6.1 RemoteBase

RemoteBase is the base class for RemoteCtlr, DistribMrmCtlr and RemoteProc. It holds the configuration information common to the derived classes, most notably:

- The XML configuration (as a QDomElement)
- A socket (a QTcpSocket, initialized by the derived classes)

RemoteBase has utility methods to extract information from the XML configuration and to write data to the socket.

6.2 RemoteCtlr and RemoteProc

RemoteCtlr (Remote Controller) and RemoteProc (Remote Process) are peers which interact over a TCP/IP socket:

3. RemoteCtlr writes a simulation's XML configuration to the socket.
4. RemoteProc reads and parses the configuration, writes a batch script and invokes RiverWare in batch mode.
5. RemoteProc reads RiverWare's output and writes it to the socket.
6. RemoteCtlr reads RiverWare's output from the socket and emits a signal.

(Steps 3 and 4 are potentially repeated many times.)

6.3 DistribMrmCtlr

DistribMrmCtlr (Distributed MRM Controller) is-a RemoteCtlr which manages a list of RemoteCtlr. DistribMrmCtlr parses the distributed XML configuration and:

- Creates an individual RiverWare configuration for each simulation.
- Constructs a RemoteCtlr for each simulation.

One aspect of creating the individual XML configurations requires explanation. The distributed XML configuration contains a list of the final RDF files, but the individual simulations can't write to the final RDF files. Instead they write to intermediate RDF files, which are combined after all of the individual simulations have finished. To this end, DisbribMrmCtlr:

- Maps the final RDF files to intermediate files.
- Uses the slot list to create intermediate RDF control files.
- Adds the intermediate control files to the individual XML configurations.
- After all of the individual simulations have finished, combines the intermediate RDF files into the final RDF files.

6.4 RemoteMgr

RemoteMgr constructs a DistribMrmCtlr with the distributed XML configuration. It then iterates over DistribMrmCtlr's RemoteCtlr objects to configure its status dialog. When the RemoteCtlr read RiverWare's output from the socket and emit a signal, RemoteMgr receives the signal and uses the information to update its status dialog.

7 RiverWare Command Line Options

Two command line options have been added to support the distributed approach - "--remote" and "--remout <file>". Both are undocumented - they should only be provided by RemoteProc when it invokes RiverWare.

"--remote" causes RiverWare to write configuration, process and RiverWare run status messages directly to standard output. The configuration status message is:

#c# errorStr

the process status messages are:

```
:s:                          // process started
:f: exitCode exitStatus      // process finished
:e: errorCode                // process error
```

and the RiverWare run status messages are:

```
@s@ state                    // single run state changed
@S@ state                    // multi run state changed
@b@ timestep %d              // single run advance block
@B@ trace numTrace %done     // multi run advance block
@F@                          // multi run finished
```

With "--remote" RclInterp registers to receive "state changed" and "advance block" callbacks from RunInfo. It queries RunInfo and the controller for the relevant status information and writes the status message directly to standard output. RemoteProc reads the status messages and writes them to the socket; RemoteCtlr reads them from the socket and emits signals; RemoteMgr receives the signals and updates its status dialog.

"--remout <file>" causes RiverWare to write diagnostic messages to the file, regardless of how diagnostics are configured. This has the effect of separating normal RiverWare messages from the status messages described above - normal messages are written to the file, the status messages are written to the socket. This greatly reduces the traffic over the socket and allows a user to view RiverWare's diagnostic messages without the status messages interspersed.

8 Batch Mode Changes

The current Rcl syntax for starting a multiple run is:

```
StartController !MRM {config name}
```

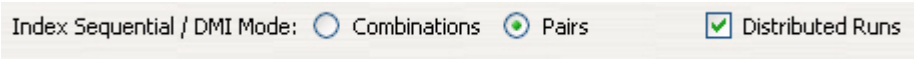
It will be expanded to allow the first trace number, number of traces, and RDF control file to be specified:

```
StartController !MRM {config name} firstTrace=N numTrace=M ctlFile=file
```

firstTrace sets the index sequential initial offset, numTrace sets the index sequential number of runs, and ctlFile sets the output control file. These override the values in the MRM configuration in the model file.

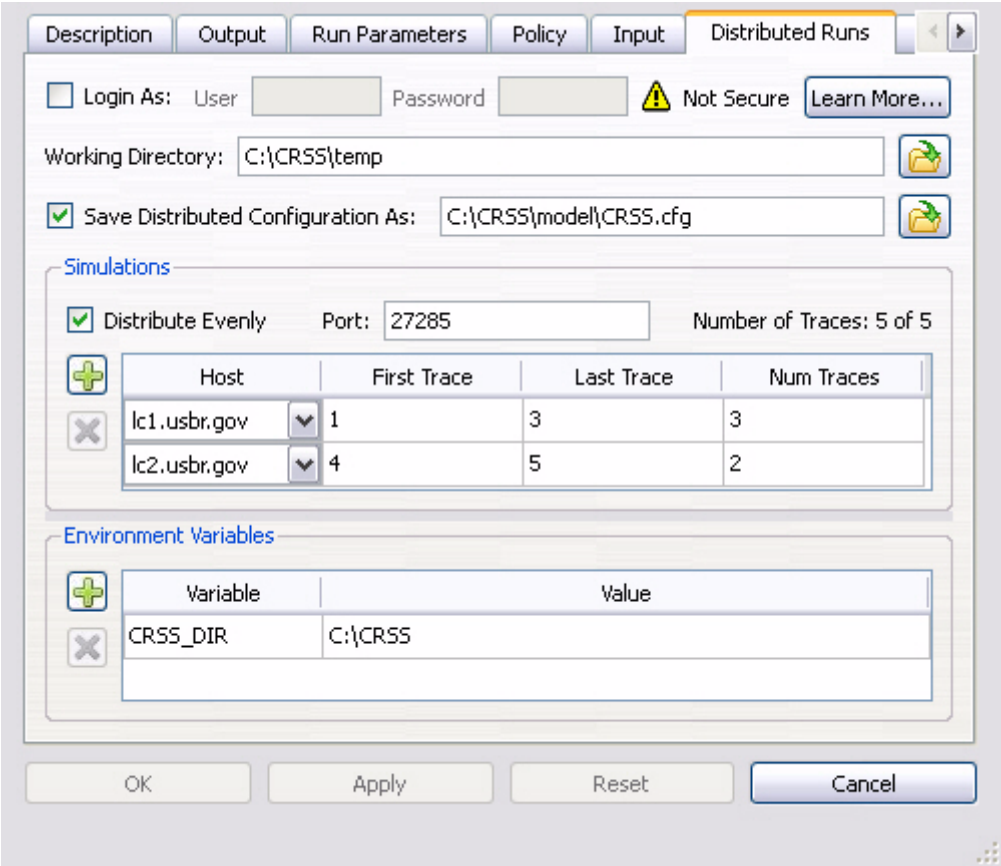
9 RiverWare User Interface

The MRM configuration dialog's "Input" tab will include a "Distributed Runs" checkbox:



If the checkbox is checked the new "Distributed Runs" tab becomes available; it allows a user to specify:

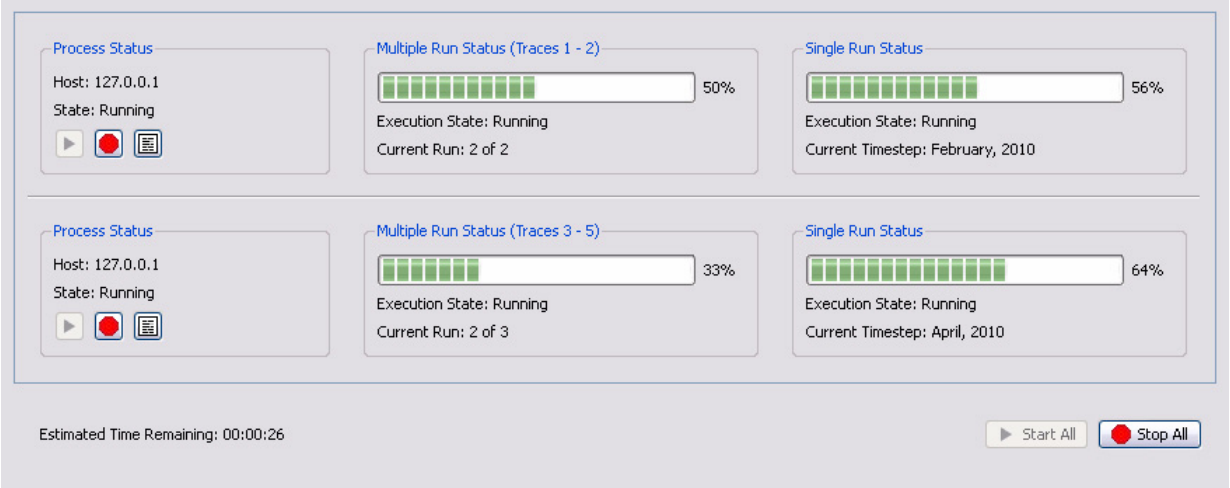
- Whether a login is required on the simulation computers and if so, optionally provide the user and password. (If either the user or password is omitted it will be prompted for when starting the run.)
- The working directory (where the partial RDF files are created).
- Whether to save the XML configuration in a named file and if so provide the name of the file.
- The simulation computers and the traces they will simulate.
- Environment variables.



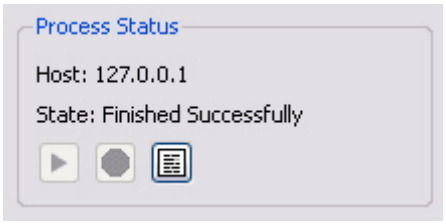
10 Remote Manager User Interface

As mentioned above, the Remote Manager includes a user interface which displays the status of the simulations. More specifically, its user interface allows a user to:

- Start and stop the simulations individually or collectively.
- View RiverWare's diagnostic output for the simulations.
- View the multiple and single run status.
- See an estimated time remaining.
- See the status of the post-processing (combining the RDF files).

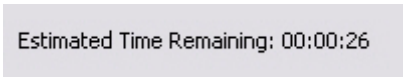


The dialog is by necessity large and difficult to fit in a document. From left to right are the “Process Status” panel, the “Multiple Run Status” panel and the “Single Run Status” panel. In better renditions, here is the “Process Status” panel with the “start”, “stop” and “view diagnostics” buttons:



and the “Multiple Run Status” and “Single Run Status” panels (which are very similar to RiverWare’s run status dialog):

The bottom of the dialog shows the estimated time remaining:



11 Combining RDF Files With CombineRdf.pl

Each simulation writes an intermediate RDF file which is a portion of the final RDF file; when all simulations have finished the intermediate files are combined into the final file. A Perl script, CombineRdf.pl, combines the intermediate RDF files. Its syntax is:

```
CombineRdf.pl <number of simulations> <number of traces> <intermediate RDF files> <final RDF file>
```

DistribMrmCtlr invokes the Perl script, so a user needn't know the specifics of it.

Note that the choice of Perl requires that Perl be installed on the controller computer. If this is not desirable, the program could easily be reimplemented in C++.

12 CRSS Input DMI Changes

The CRSS input DMIs have trace directories (which contain the data files) and a Temp directory:

```
Temp
trace1
trace2
...
```

The DMI executables, which are Perl scripts, are invoked with the trace number as a command line option. The scripts copy data files from the appropriate trace directory to the Temp directory:

```
my $tempdir = "$TRACES/Temp";
```

The control files refer to the Temp directory:

```
DoloresRiver.Inflow: file=$(CRSS_DIR)/dmi/NFSinput/Temp/%o.%s
```

and RiverWare imports data from the Temp directory.

This is not adequate when there are multiple simulations. For example, one simulation might be copying the data files for trace157 to the Temp directory at the same time another simulation is copying the data files for trace 1306 to the Temp directory, leading to unpredictable results. To resolve this, the MRM_NUM environment variable was introduced, with the first simulation having MRM_NUM=0, the second simulation having MRM_NUM=1 and so on. With this, the scripts now copy data files to the TempN directory, where N is the simulations MRM_NUM:

```
my $mrm_num = $ENV{MRM_NUM};
my $tempdir = "$TRACES/Temp$mrm_num";
```

The control files now refer to the Temp$(MRM_NUM)

```
DoloresRiver.Inflow: file=$(CRSS_DIR)/dmi/NFSinput/Temp$(MRM_NUM)/%o.%s
```

Multiple simulations can now run the input DMIs simultaneously, without conflicts.

(This also means that in order to run the DMIs outside of the distributed framework a user is required to set MRM_RUN=0 in her environment, as she is required to set CRSS_DIR to the appropriate value. There are alternative approaches which can be discussed, but this is probably the best approach.)

13 Installing The Distributed Framework

It's likely that installing the distributed framework will be incorporated into InstallShield, but for now the installation is manual and requires administrator permissions. There are four new executables which are co-located with riverware.exe:

- RwService.exe
- RwSvcCtlr.exe
- RwRemoteMgr.exe
- CombineRdf.pl

although for the most part users will only interact with RwRemoteMgr.exe.

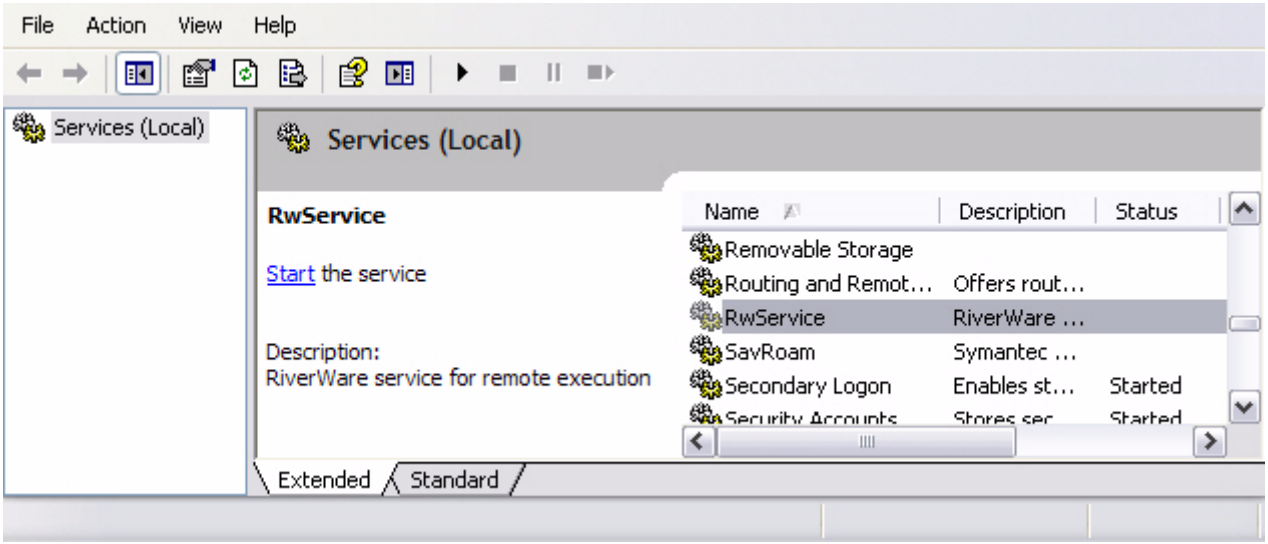
For this section we'll assume the installation directory is "C:\Program Files\CADSWES\RiverWare 5.2".

13.1 Installing RwService.exe

The RiverWare service, RwService.exe, must be running on each simulation computer. The RiverWare Service Controller, RwSvcCtlr.exe, is used to install RwService.exe from a Windows Command Prompt:

```
cd C:\Program Files\CADSWES\RiverWare 5.2
RwSvcCtlr.exe -i "C:\Program Files\CADSWES\RiverWare 5.2\RwService.exe"
```

Once the service has been installed it can be started using the Windows Service dialog:



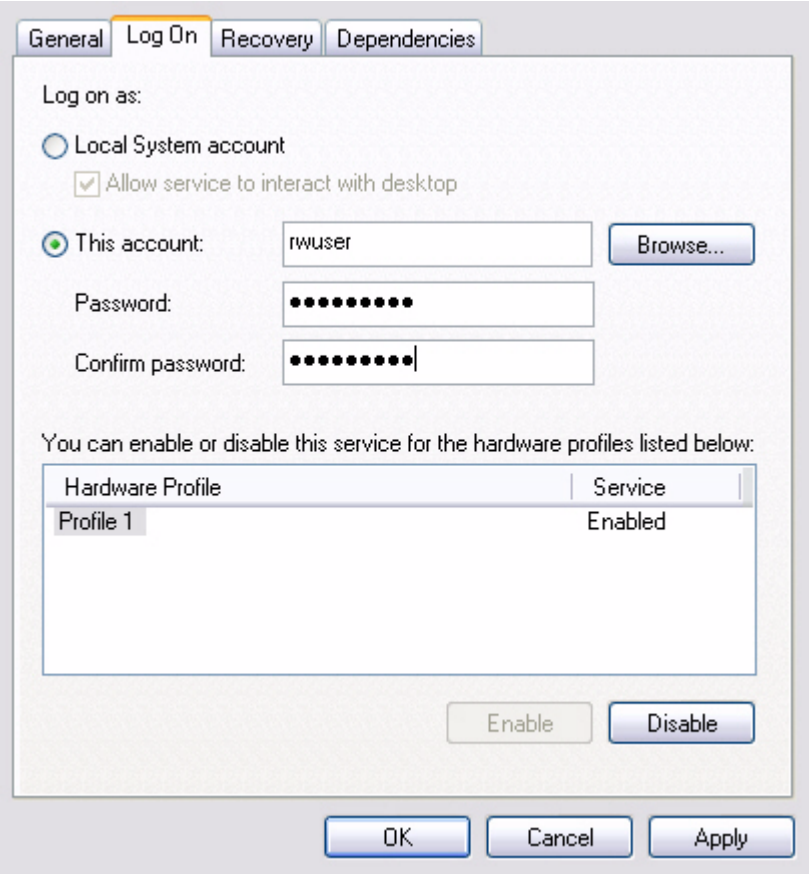
Although not shown, the service can be stopped, paused and restarted using the Service dialog (although it must be started and not paused for the distributed simulations to run).

RwSvcCtlr.exe is also used to uninstall the service:

```
cd C:\Program Files\CADSWES\RiverWare 5.2
RwSvcCtlr.exe -u "C:\Program Files\CADSWES\RiverWare 5.2\RwService.exe"
```

13.2 Setting The Service Log On

By default a program started by a service has very limited permissions. RiverWare is started by the RiverWare service on the simulation computers, so by default it has very limited permissions. For example, it might not be able to access network directories or write in certain directories. To alleviate this, it might be necessary for the RiverWare service to run as a specific user. To set this up, right-mouse-click on the "RwService" line in the Service dialog, select "Properties" and select the "Log On" tab. Here you can specify the user the RiverWare service will run as:



Although not shown, in the Properties dialog, in the "General" tab, it's possible to specify whether RwService is started manually (through this dialog, the default) or automatically (when the computer starts up).