



Technical Documentation Version 7.1

Performance



C A D S W E S

Center for Advanced Decision Support for Water and Environmental Systems

These documents are copyrighted by the Regents of the University of Colorado. No part of this document may be reproduced, stored in a retrieval system, or transmitted in any form or by any means electronic, mechanical, recording or otherwise without the prior written consent of The University of Colorado. All rights are reserved by The University of Colorado.

The University of Colorado makes no warranty of any kind with respect to the completeness or accuracy of this document. The University of Colorado may make improvements and/or changes in the product(s) and/or programs described within this document at any time and without notice.

Performance Table of Contents

Introduction	1
Performance Analysis Tools	2
RPL Analysis	2
RPL Timing/Count functions predefined function	2
Windows Task Manager	4
Windows perfmon utility	4
Run RiverWare More Efficiently	7
Reduce Diagnostics processing	7
Disable diagnostics	7
Filter diagnostics	7
Send diagnostics to a file.....	9
Break long runs into multiple runs	9
Close unused windows	9
Run in batch mode	9
Improve your RPL set logic	10
Invest in Hardware	10
Install More Physical Memory (RAM)	10
Purchase additional faster disks	11
Use additional CPUs for multiple runs	12
Purchase a bigger, faster machine	12
Consider running on a 64 bit architecture.....	12
Physical Memory.....	12
Cache Size.....	13
Front Side Bus disk speed.....	13
CPU	13
Manage Memory	14
Close other applications	14
Adjust the size or location of your paging file	14
Configure to use 3 Gig tuning	14

Performance

1. Introduction

This section provides suggestions for improving RiverWare model run times; that is, we try to answer the question: what can you do to make your model run faster?

The short answer:

- Turn off unnecessary diagnostics, close as many RiverWare windows as possible, and close other applications.
- Determine if there is excessive paging during the run. If paging is a problem, purchase more physical memory. If that is not possible, adjust the paging properties.
- Analyze your model run using RiverWare utilities to determine if there are unexpected slowdowns due to excessive dispatching or slow RPL functions. Address these slowdowns by modifying your model or ruleset.
- Improve hardware; sometimes this is the most cost effective method to improve performance.
- For long production runs, use batch mode and/or divide the model into multiple runs on multiple CPU's.
- If a simulation exceeds the system's virtual memory limit (RiverWare exits with an "out of memory" error) there are several possibilities for addressing the issue: enable 3 Gig tuning, or run a 64 bit version of RiverWare on a 64 bit operating system. These options are discussed in more detail [HERE \(Section 5.3\)](#) and [HERE \(Section 4.4.1\)](#). If pursuing these options does not resolve the problem, contact riverware-support@colorado.edu as there could be a modelling or internal RiverWare error.

Of course, RiverWare attempts to perform its operations as efficiently as possible, but run times are often affected by model particulars as well as the configuration of the system on which RiverWare is run. RiverWare can not control these factors, but many of them are under your control and these are the subjects of this section. This document attempts to be general but also attempts to prioritize the steps that will make the most difference to a user. This is very difficult to do as each user has a different system, budget, and institutional requirements. A step that may make the most sense for one user (like buying new hardware) may not even be an option other users.

While some of the details presented here are specific to the Windows operating system, in general our suggestions apply equally well to all platforms in which RiverWare is supported. Where we present information specific to Windows, examples are taken from the XP operating system. Although we introduce some technical concepts, e.g., RAM, CPU, and Paging, this document is not intended to be a complete reference on computer performance issues. For more information, the reader is directed to the internet or reference books for a particular system or topic.

The remainder of this section is divided into three subsections. In the first, we describe a set of tools useful for identifying performance problems or slowdowns in RiverWare. In the second, we describe steps which may be taken within RiverWare to make models run faster. Finally, we discuss several ways to improve your computer system for optimizing RiverWare performance on your model.

2. Performance Analysis Tools

The first step in improving performance is typically information collection. This section describes a number of tools and approaches used to diagnose performance problems, including both tools within RiverWare as well as external utilities. The following topics are discussed:

- the RPL Set Analysis Tool
- the RunTime predefined RPL function
- the Windows Task Manager
- the Windows **perfmon** utility

2.1 RPL Analysis

In simulations that use RPL sets (e.g., RBS or optimization rulesets, the expression slot set, or the user defined accounting method set), the RPL Set Analysis tool can be used to identify functions that execute an inordinate number of times or spend a lot of time in their execution. Please consult the documentation [HERE \(RPLDebugging.pdf, Section 5\)](#) for more information on this utility.

2.2 RPL Timing/Count functions predefined function

There are RPL predefined functions available to measure performance as follows:

- RunTime() returns the number of seconds which have elapsed since the run began. Click [HERE \(RPLPredefinedFunctions.pdf, Section 156\)](#) for more information on this function.
- DispatchTime() Returns the accumulated time spent executing object dispatch methods since the beginning of the current run. Click [HERE \(RPLPredefinedFunctions.pdf, Section 29\)](#) for more information on this function.
- DispatchCount() returns the number of object dispatch method executions that have occurred since the beginning of the current run. Click [HERE \(RPLPredefinedFunctions.pdf, Section 27\)](#) for more information on this function.

Note: More specific dispatch times and counts are also available in the RPL Set Analysis tool described [HERE \(RPLDebugging.pdf, Section 5\)](#).

Using expression slots or rules, these function can be used to record the performance. Below is a series of expression slots that record the cumulative number of seconds that have elapsed since the beginning of the run or determine the number of seconds each timestep takes (see Figure 1). This can be used to determine if a particular operation is taking more time than expected. For example, does a timestep that has a flood control operation take significantly longer than a non-flood control timestep?

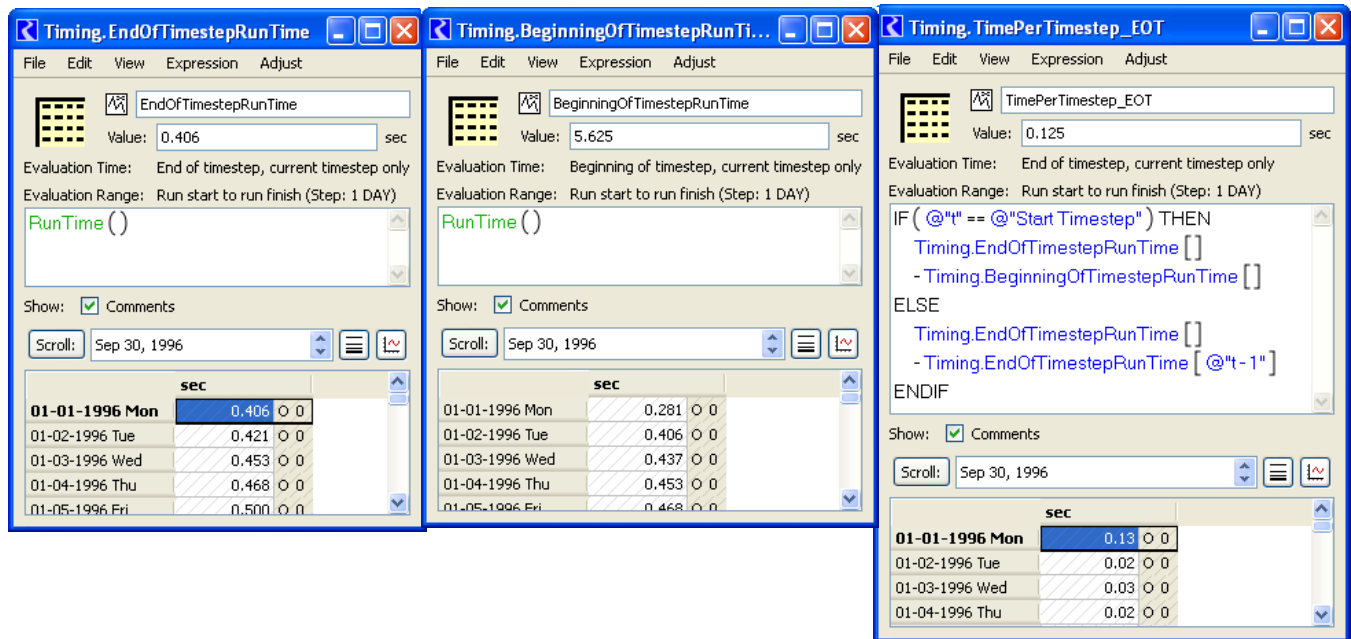


Figure 1: Expression Slots showing how the RunTime function can be used. Note, the left and right ones are configured to evaluate at “End of Timestep, current timestep only”, the center one is configured to evaluate at “Beginning of Timestep, current timestep only”. Together, these give you the “time per timestep” shown in the right expression.

2.3 Windows Task Manager

The **Task Manager** is accessed by right-clicking on the tool bar and selecting **Task Manager**. The **Task Manager** has two tabs that record performance information. The **Performance** tab displays a number of useful items as shown in Figure 2.

- The total CPU usage
- The number of CPU's on this machine and a plot of their usages.
- The total physical memory (RAM) on this machine. In Figure 2, this is approximately 3 GB of RAM.

The **Processes** tab displays the percentage of the CPU used and the CPU Time for each process.

We will refer to this utility in future sections. When using this utility to assess the performance of a RiverWare model, make sure that other processes are not running.

2.4 Windows *perfmon* utility

Within Windows, there is a performance monitoring utility called **perfmon**. This tool can be used to determine how processes including RiverWare are using memory. When using this utility to assess the performance of a RiverWare model, make sure that other processes are not running.

Perfmon is accessed from a command prompt (Start ➔ All Programs ➔ Command Prompt or) by typing **perfmon** (or: Start ➔ Run..., then type in **perfmon**). **Perfmon** displays counters of variables. The following instructions are provided to show you how to use **perfmon** for simple analysis. This document is not comprehensive; use **perfmon**'s help for more information.

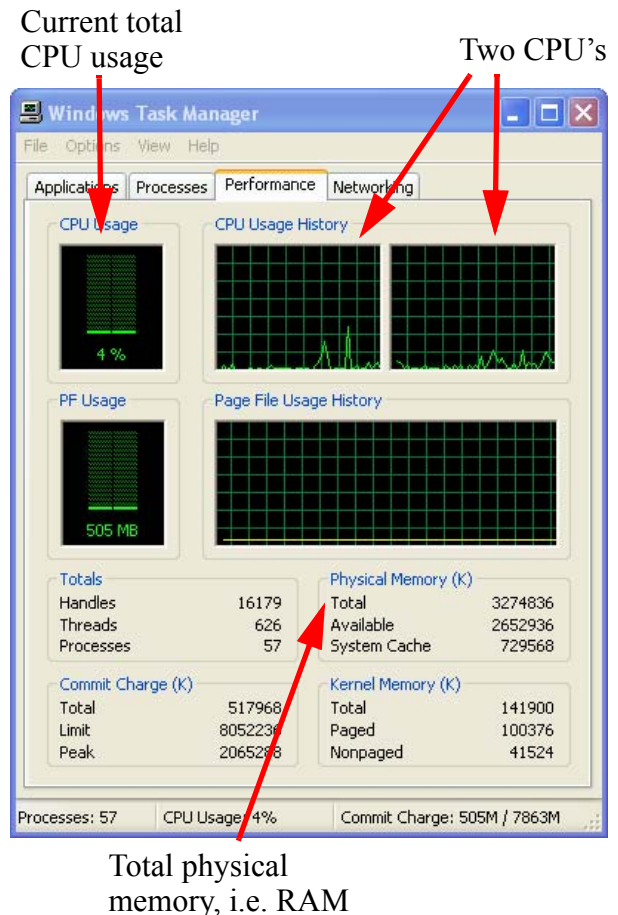


Figure 2: Windows Task Manager, Performance tab

When you first start **perfmon**, a graph of the selected or default counters is provided as shown in Figure 3.

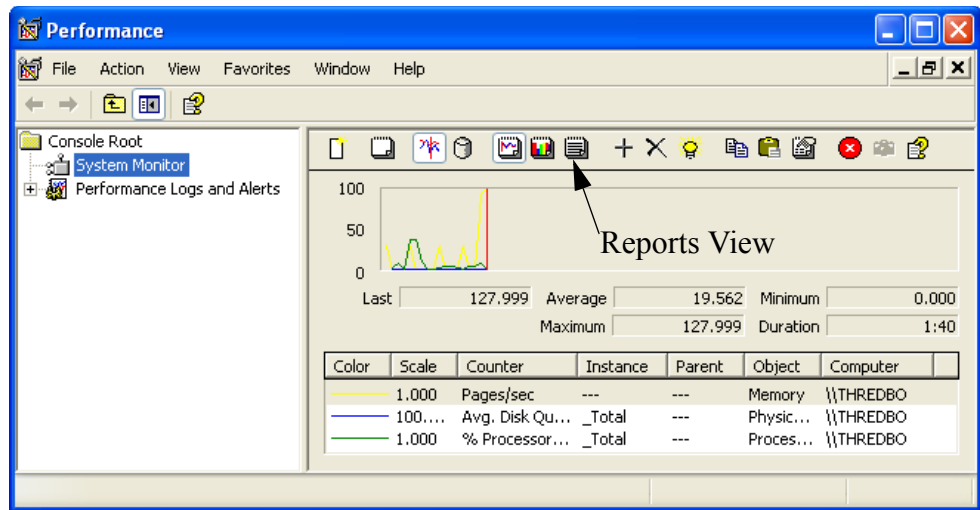



Figure 3: Perfmon default view

The most useful view is the Reports view. Click on the Reports icon  to switch views. Figure 4 shows the reports view.

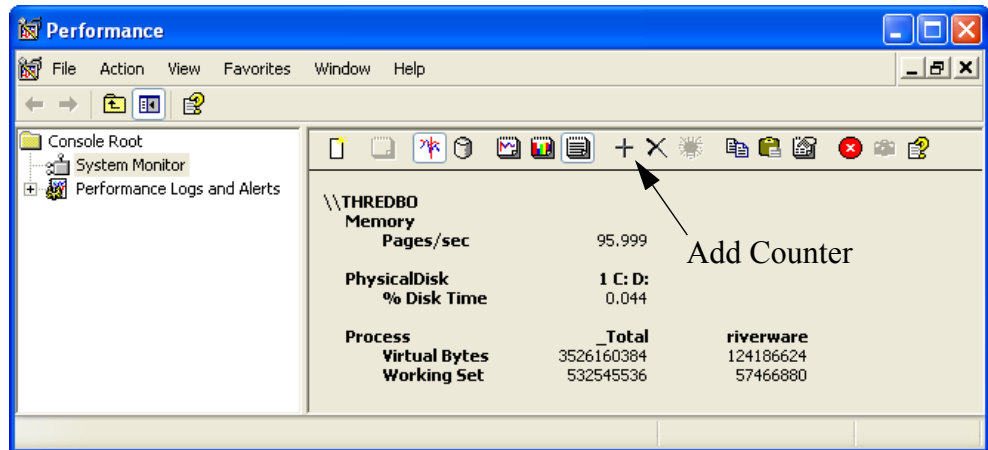



Figure 4: Perfmon Reports view

There may or may not be counters defined. To add counters, click on the plus icon  to open the **Add Counters** dialog, Figure 5. In the **Add Counters** dialog, shown to the right, select a **Performance Object** (1), highlight the counter from the list (2), select the **instance**, i.e. riverware (3), click **Add** (4). Repeat for other counters. Click **Close** when finished.

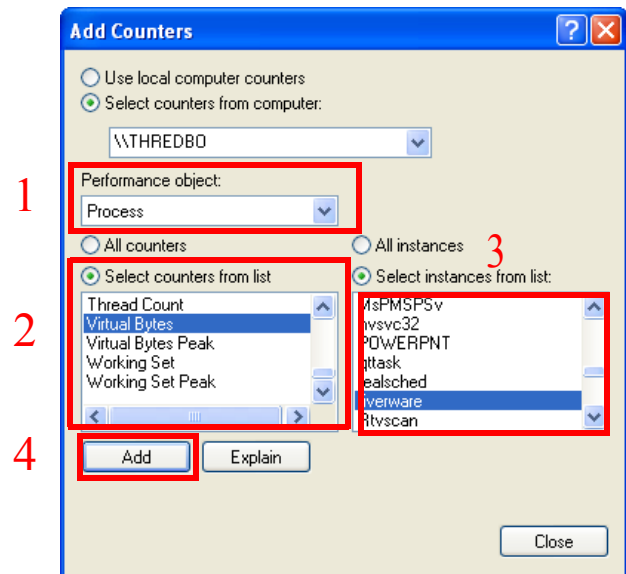


Figure 5: Perfmon Add Counters dialog

The most useful counters, as displayed in Figure 4, are as follows:

Performance Object	Counter	Instance	Description
Memory	Pages/Sec	NA	Hard Page Faults per second
Physical Disk	% Disk Time	The disk that contains the paging file	Percentage of time the selected disk is reading or writing.
Process	Virtual Bytes	_Total	Total size of all process
Process	Virtual Bytes	riverware	Total size of the RiverWare process
Process	Working Set	_Total	Size of memory that was recently touched by all processes
Process	Working Set	riverware	Size of memory that was recently touched by RiverWare

Note: For more information or a description of any of these counters, highlight the counter in the **Add Counter** dialog and click **Explain**. Once the **perfmon** tool is configured, it can be saved as a .msc file and reloaded as necessary.

Now that we have tools to collect performance information, we present recommendations for acting on this information to improve performance.

3. Run RiverWare More Efficiently

This section describes steps that can be taken to run RiverWare more efficiently and thus improve runtime performance.

3.1 Reduce Diagnostics processing

When diagnostics are enabled, RiverWare must process diagnostics as well as perform the primary calculations of the simulation. The overhead associated with diagnostics consists of the following:

1. Deciding whether or not each individual diagnostic should be issued
2. Collecting the data to be posted
3. Posting the diagnostic

In some contexts, steps 1 and 2 can require significant computation, but in most situations it is step 3 which dominates diagnostic processing time. For a general discussion of diagnostics, click [HERE \(Diagnostics.pdf, Section 1\)](#).

3.1.1 Disable diagnostics

Since the overhead associated with processing diagnostics can represent a significant fraction of a run, if they are not necessary, and performance is an issue, then it is best to disable them. To disable, make sure the **Enable Informational Diagnostics** option is NOT checked in the **Diagnostics Manager**.

3.1.2 Filter diagnostics

When diagnostics are being sent to the **Diagnostic Output Window**, the time to post a diagnostic is related to the number of diagnostics in the diagnostic output window. In other words, the more diagnostics that have already been posted, the more time it requires to post an additional diagnostic. This means that the time spent posting diagnostics within a total run is proportional to the square of the number of diagnostics posted. Thus even though posting an individual diagnostic is not expensive, it can take a lot of time to post a large number of them. By filtering diagnostics to post only those diagnostics

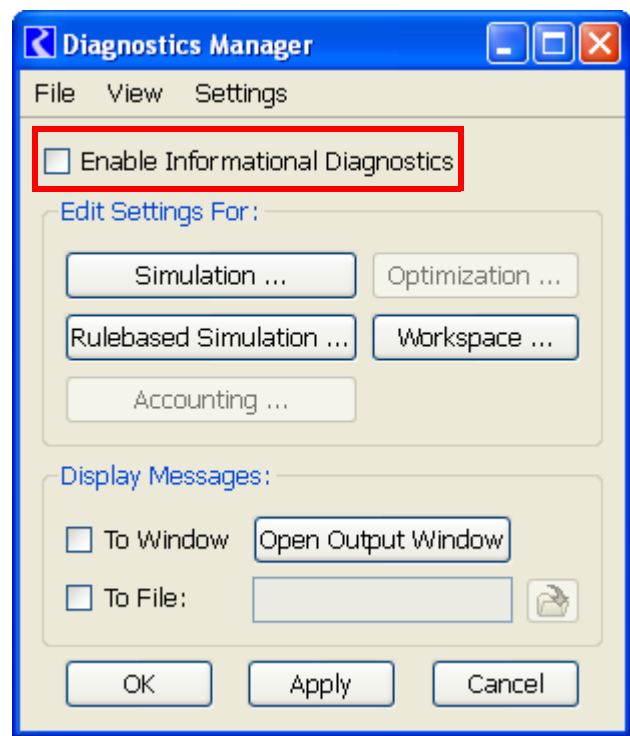


Figure 6: No check means that diagnostics are disabled

currently of interest, run times are decreased and less effort is required to sift through diagnostics for the desired information. Figure 7 shows the diagnostics filtering dialog with a number of filters enabled.

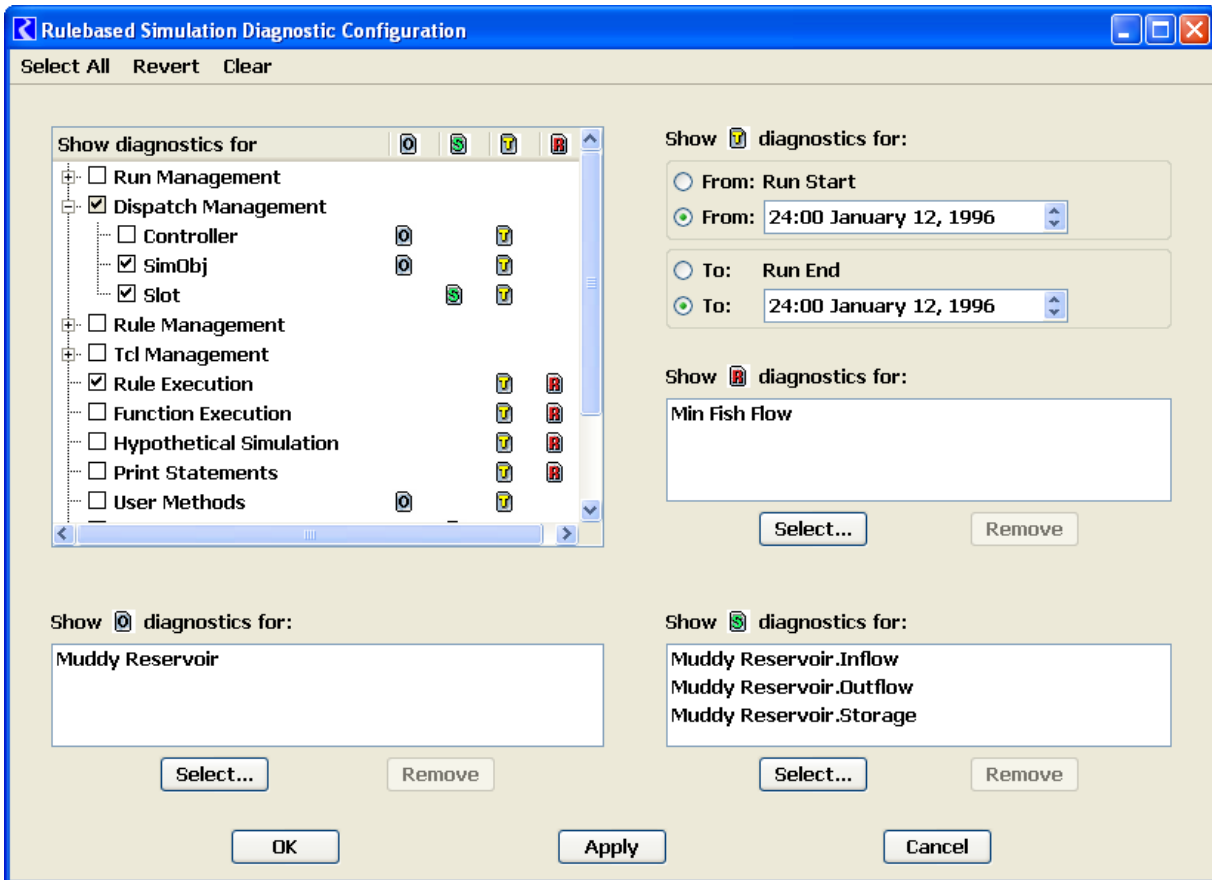


Figure 7: Diagnostics Filtered to show only those for
Timestep = January 12, 1996
Rules = Min Fish Flow
Object = Muddy Reservoir
Slots = Muddy Reservoir.Inflow, Outflow and Storage

3.1.3 Send diagnostics to a file.

If it is useful to see a large number of diagnostics, then consider sending them to a file instead of to the Diagnostic Output Window. File I/O is relatively fast, and one can still view the diagnostics in a text editor, perhaps one with more sophisticated searching and filtering capabilities than those provided by the Diagnostic Output Window.

Note: Posting a diagnostic to the diagnostic output window requires additional memory usage, so sending diagnostics to a file instead of a window will reduce RiverWare's memory use which can have a positive impact on run times as well.

3.2 Break long runs into multiple runs

For most models, the size of the RiverWare process is dominated by series data. If the model is too large to fit into virtual memory (or too large for the 32 bit 3 Gb Windows limit, see below), there are two ways to break up the model, either in time or in space. To break the model up by time, divide the total run into multiple runs of shorter duration, configure DMIs to exchange data between runs, and write RCL scripts to conduct these runs in sequence.

Another possibility is to simulate upstream portions of the model for the full time range of interest and then import the results into a model that simulates the remainder of the basin. Of course, operational dependency of upstream objects on downstream objects may prevent this from being an option.

3.3 Close unused windows

Updating the graphical user interface during a run requires CPU time and increases the size of the RiverWare process. When it is desirable to run RiverWare interactively, you can reduce the amount of time updating dialogs during the run by closing them. This is especially important for series data such as that displayed by an SCT, a SeriesSlot dialog or a model run analysis dialog. Each time a slot value changes internally, the corresponding slot's dialog might need to be updated if it is open. In general, if performance is critical, we recommend that all unused dialogs be closed.

3.4 Run in batch mode

Consider completely avoiding updating the user interface during a run by writing a RCL script to conduct a batch mode run. After a run, you can then bring up RiverWare interactively to analyze the results.

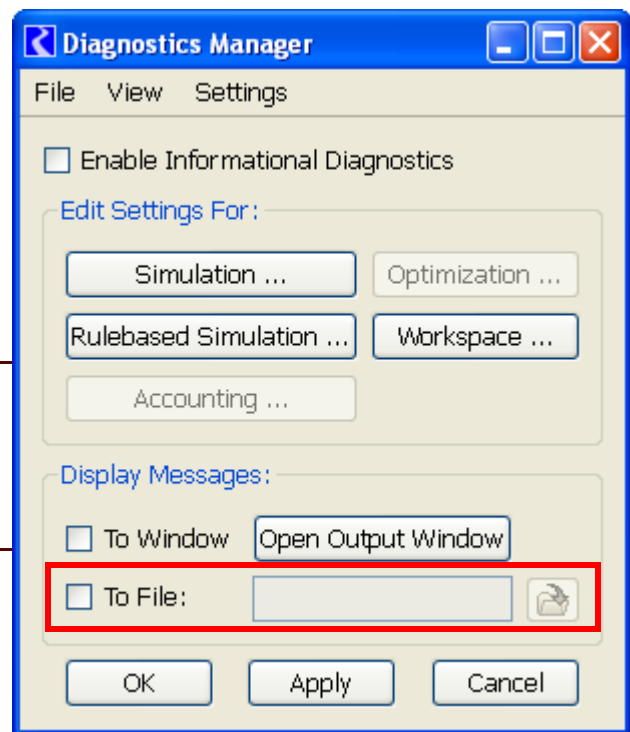


Figure 8: Check box to send diagnostics to a file. Then click the select file button to choose a file

3.5 Improve your RPL set logic

Although RiverWare often spends much of its time dispatching, RPL statement execution can represent a significant factor of the time spent during a rulebased simulation. Some RPL set logic is more computationally intensive than others, and changing RPL logic with this in mind can often improve model run times. Use the RPL Analysis tool described [HERE \(RPLDebugging.pdf, Section 5\)](#) to identify functions that are taking a long time or execute too many times. Then, determine if there is a more efficient way to write these functions. A list of suggestions to writing efficient RPL expressions is presented [HERE \(RPLUserInterface.pdf, Section 2.7\)](#).

4. Invest in Hardware

Often, the most significant improvements in RiverWare model performances can be obtained by investing in hardware. When investing in hardware, the following should be considered (in order of most significance):

- Invest in a 64 bit machine
- Physical Memory (RAM) - purchase enough to eliminate paging
- If completely eliminating paging is impossible, purchase multiple faster disks.
- If making multiple runs, purchase additional machines or use multiple CPUs.
- Purchase a machine looking at the following attributes:
 - Memory size
 - Cache size
 - Bus speed
 - Disk speed
 - CPU speed

4.1 Install More Physical Memory (RAM)

If your model is causing paging, then installing more RAM will lead to less paging and faster run times. Most modern operating systems allow programs to access more memory locations than are available to that program in physical memory. When RiverWare attempts to access a location that is not in physical memory, a page fault occurs and the operating system brings the page containing that location into physical memory. Since the time to access a memory location is likely to be orders of magnitude quicker if that memory location is currently in physical memory than when it is located in disk storage, reducing the number of page faults during a RiverWare run will decrease the run time.

To determine if your model leads to page faults and the associated disk I/O delays, use the Windows **perfmon** tool, described [HERE \(Section 2.4\)](#), to display the **Memory: Pages/Sec** counter and the **PhysicalDisk: % Disk Time** for the disk which contains your system page file. These counters will be non-zero when the system is processing (hard) page faults. You will also notice that when RiverWare is doing significant I/O, the CPU Usage for the RiverWare process as displayed in the Windows **Task**

It also helps if the different operations running on the machine don't compete for use of the same disk. For this reason it helps to have your paging file(s) on different physical disks from your normal data and programs. This topic is discussed in the "Manage Memory" section [HERE \(Section 5.2\)](#).

4.3 Use additional CPUs for multiple runs

If you are making multiple runs, i.e. alternative analysis, and you have access to multiple machines (or a single machine with multiple CPUs), then you can run multiple RiverWare processes simultaneously. For example, if you are completing an alternatives analysis and are ready for three production runs, use three machines to run alternatives 1, 2 and 3 in parallel. The entire analysis will take the same time as running one run and should be three times faster than running them in series. Batch processing can help to manage many parallel different runs

Also remember that a machine with multiple CPU's will speed execution of parallel runs but the total physical memory is shared among all CPU's and can limit run speed. In that regard, it may still be better to run parallel runs on completely separate machines.

4.4 Purchase a bigger, faster machine

Above we have presented some approaches to upgrade existing hardware so that RiverWare performs better. Often it may be more cost effective to replace hardware with new machines. When selecting a new machine look at the following attributes, in order: 64 bit, physical memory size, cache size, bus speed, disk speed, and CPU speed.

4.4.1 Consider running on a 64 bit architecture

If RiverWare is encountering an "out of memory" error, then you will need to increase the RiverWare's virtual memory limit in one of several ways:

- On 32 bit Windows XP, you can increase the limit from 2 GB to 3 GB by configuring the system as discussed [HERE \(Section 5.3\)](#).
- If a simulation requires more than 3 GB of virtual memory, then you will need to run the 64 bit RiverWare executable under a 64 bit Windows operating system, effectively permitting an unlimited address space.

4.4.2 Physical Memory

Depending on the operating system you are running, you might benefit from having more than 4 GB of physical main memory, especially on a 64 bit operating system. Most workstation or personal computer operating systems do **not** make use of more than 4 GB, and purchasing more would waste money. That said, you might be wise to purchase a machine with enough slots and large enough memory boards (e.g., 2 2 GB rather than 4 1 GB boards) to allow you to expand in the future to more than 4 GB.

Most operating systems configured for "servers" do allow large banks of main memory to be used. At CADSWES we have not yet experimented with Windows server configurations, so we cannot attest to

the potential speed benefits of purchasing such a machine, nor can we guarantee that there will not be attendant problems. This is an option worthy of consideration.

4.4.3 Cache Size

Data accesses are faster if the data reside in memory than if they reside on disk. Modern computers further speed up memory accesses by employing caches of faster memory. Cache memory is physically different from main memory, making it more expensive, and cache memory physically resides closer to the CPU (the heart of the computer) than does main memory.

Thus, if the data needed resides in a memory cache, the access will be faster than if it resides in main memory. Consequently, you will benefit from having the largest caches available on the machines you purchase. Some architectures have Level 1 (L1), Level 2 and Level 3 caches. Some have only L1 and L2 and often only the lower-level (L2, L3) cache sizes are configureable when you purchase a machine.

4.4.4 Front Side Bus disk speed

Accesses to main memory are slower than accesses to cache memory. This is because the cycle time of the cache memory is lower than that of main memory, but also because the electrons have to travel farther from main memory than from cache memory. On many architectures, data (and instructions) flowing from main memory to the CPU have to travel across a central system bus. Consequently, the speed of the bus determines the speed of main-memory accesses, and a good rule of thumb is to purchase the fastest bus you can get. Most Intel-based computers call this bus the “Front-Side Bus”.

4.4.5 CPU

Most object-oriented applications, RiverWare included, are not limited by CPU speed, due to the nature of their memory access patterns. Consequently, CPU speed is not the most important factor in purchasing a new machine as most new machines have adequate CPU speeds.

One of the ways modern computers are made more powerful is by giving them more than one CPU. Some applications are “multi-threaded”, which allows them to run somewhat faster by using more than one CPU in parallel. RiverWare is not multi-threaded, however, you might still benefit from having multiple CPUs because the operating systems typically run more than one program, even if you have nothing but RiverWare running. Having another CPU can reduce the amount of time that RiverWare has to give up the CPU and its cached data for use by another process. If you tend to use your machine for things other than running RiverWare (that is to say, you do not have a dedicated machine for RiverWare), then you most certainly can benefit from additional CPUs.

Note: In so-called “multi-core” machines, two or more CPUs might share some cache, but be unable to share all of the cache. For example, a 4 core machine might in fact be two 2 core machines, where two CPUs share half the L2 cache and the other two CPUs share the other half of the L2 cache. This kind of architecture might be labelled as having, say 8 MB of L2 cache, when in fact it has two 4 MB L2 caches (2 x 4 MB). In this example, the most L2 cache that RiverWare could use is 4 MB. This also means that a process scheduled on the CPU that shares cache with RiverWare’s CPU can have a negative impact on RiverWare’s

performance, whereas the processes scheduled on the other CPUs have no effect on RiverWare's available cache. The upshot: a 2 CPU machine with caches dedicated to each CPU might perform better than a 2 CPU machine with shared cache, all else being equal.

5. Manage Memory

When you run a RiverWare model on a computer, there are other applications that are competing for the computer's memory resources (such as the CPU or a disk). If your model is small and your machine is "big/fast", there is no conflict. If the opposite is true, your model is big and your computer is "small/slow", (this is more likely if you are reading this), then there may be competition between applications for the limited system resources. This section describes three approaches to make the RiverWare model run faster by adjusting other applications or changing the configuration of the operating system.

5.1 Close other applications

When other applications are running at the same time as RiverWare, it is possible that competition for system resources will degrade RiverWare's performance. The obvious solution is to close the other applications. But, if you do not want to close out of other applications, it is possible to tell the operating system to give preference to RiverWare by increasing its priority. To change the priority:

On the **Task Manager**, go to the **Processes** tab. Locate the **riverware.exe** process, (note, there may be more than one if you are running multiple models). Right click on **riverware.exe** and select **Set Priority** ➔ **High**. It will warn that system instability may occur. Click **OK**.

5.2 Adjust the size or location of your paging file

If RiverWare is paging and you can not reduce the amount of paging, then you can sometimes configure your system to reduce the overhead associated with paging. First, move the page file off the disk that holds your system partition to another fast, dedicated hard disk. By moving your page file to a dedicated disk, you can reduce the time to write and read pages. In addition, if you can spread the page file across several non-system disks, you can decrease page file competition.

It is beyond the scope of this discussion to provide all the potentially relevant details for various systems configurations, however your system administrator or IT department should be able to assist if necessary. For other resources, search the internet for "windows page file".

5.3 Configure to use 3 Gig tuning

This is not so much a tip for improving performance as a tip to use if you run out of memory when running your model. Typically when you run out of memory, the process crashes and sometimes posts an out-of-memory error depending on how you are running. If this happens, you may possibly be able to use 3 Gig tuning to allow your model to continue running.

In the default XP windows configuration, windows reserves 2 GB of the 4 GB of addressable space for its use. This leaves 2 GB for all other applications. The 3 Gig Tuning feature of XP allows the user to increase the amount of available memory (physical + virtual) for applications by 50% - from 2 GB up to 3 GB. The operating system then uses only 1 GB of the addressable space for its processing. Again, this is a fairly advanced topic and we encourage you to use online Windows documentation for more information on how to configure this feature:

- Available switch options for the Windows XP Boot.ini files: <http://support.microsoft.com/kb/833721>
- How to edit the Boot.ini file in Windows XP: <http://support.microsoft.com/kb/289022>
- How to use the /USERVA switch in the Boot.ini file to tune /3GBconfigurations: <http://support.microsoft.com/kb/316739>

3 Gig tuning only applies to 32 bit, not 64 bit operating systems. On 64 bit operating systems, the full set of available addressable space is available for each process.