RiverWare Performance

A Study of an URGWOM

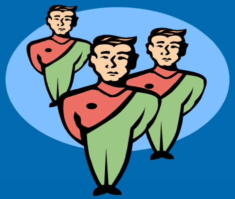
Water Operations Model

Performance Improvements Realized

RiverWare User Group Meeting
March 1-2, 2005

What We Knew When We Started

Users say rules + accounting =

Hypothetical simulation has much overhead in cloning objects. 

We have suspected that free-memory fragmentation might be a culprit, but had no empirical evidence.

The Study

The Model

300 Simulating Objects

1590 Accounts, 14573 Acct Slots, 7.8M acct solves, 84% early

1862 Supplies, set by rules

211 Active Rules

694 Active Functions

21 Hyp Sim Functions

1098 Timesteps

6623 Hyp Sim invocations, 36234 simulations, 11.3M dispatches

The Tools

Quantify ™ - GUI profiling tool

Purify ™ - GUI memory-usage analyzer

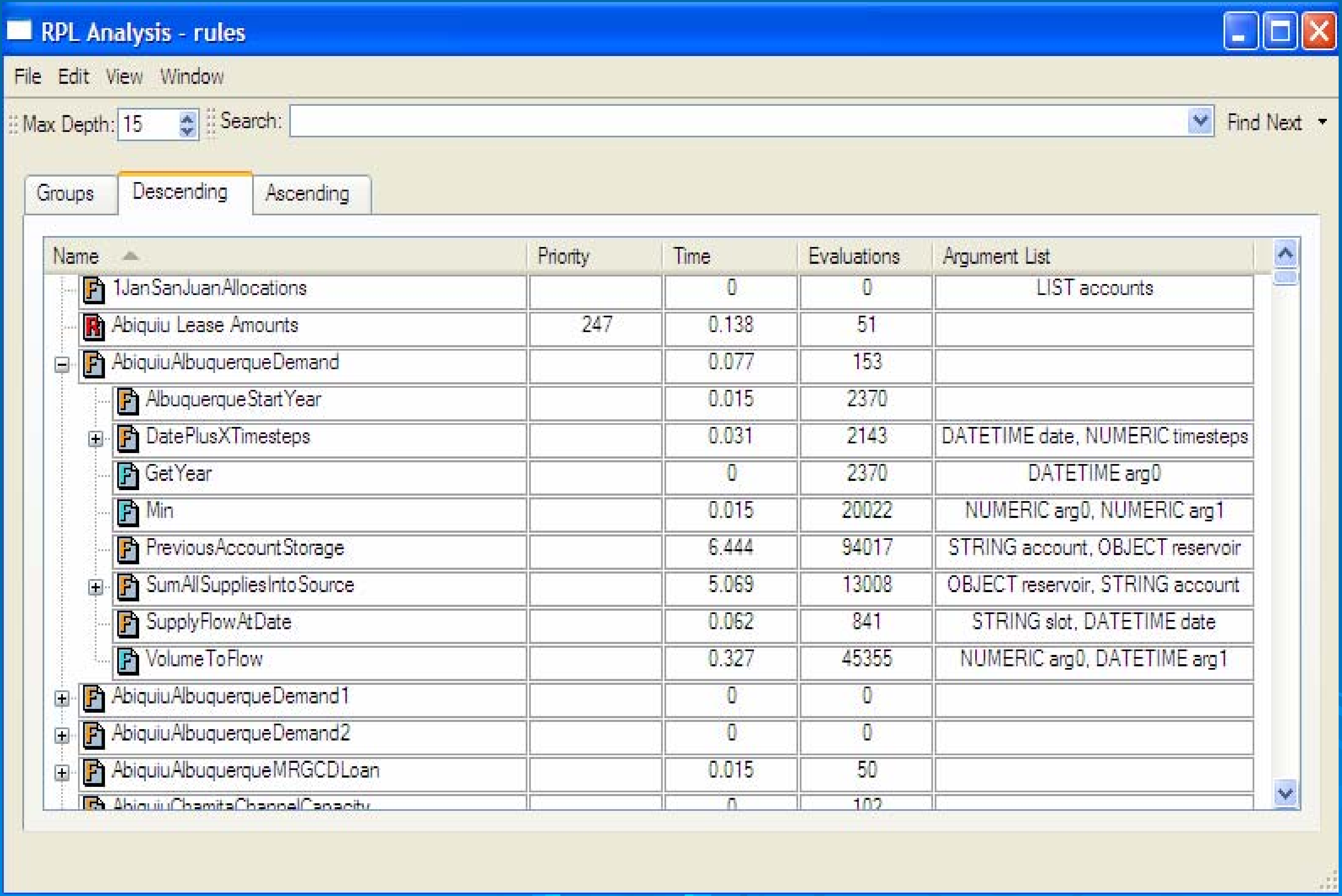
Hand-coded CPU timers

Instrumented memory management library (Solaris)

Hand-coded counters

Rpl Set Analysis Tool (RPLSAT)

RPL Set Analysis Tool



Max Depth: 15

Search:

Name	Priority	Time	Evaluations	Argument List
F 1JanSanJuanAllocations		0	0	LIST accounts
R Abiquiu Lease Amounts	247	0.138	51	
⊟ F AbiquiuAlbuquerqueDemand		0.077	153	
F AlbuquerqueStartYear		0.015	2370	
⊞ F DatePlusXTimesteps		0.031	2143	DATETIME date, NUMERIC timesteps
F GetYear		0	2370	DATETIME arg0
F Min		0.015	20022	NUMERIC arg0, NUMERIC arg1
F PreviousAccountStorage		6.444	94017	STRING account, OBJECT reservoir
⊞ F SumAllSuppliesIntoSource		5.069	13008	OBJECT reservoir, STRING account
F SupplyFlowAtDate		0.062	841	STRING slot, DATETIME date
F VolumeToFlow		0.327	45355	NUMERIC arg0, DATETIME arg1
⊞ F AbiquiuAlbuquerqueDemand1		0	0	
⊞ F AbiquiuAlbuquerqueDemand2		0	0	
⊞ F AbiquiuAlbuquerqueMRGCDLoan		0.015	50	
F AbiquiuChamitaChannelCapacity		0	102	

Where To Start?

- ✓ Is early or excessive account solving a problem?
- ✓ Use Quantify ™ to get time breakdown.
  - % Times for function + descendents, # calls.
  - Reconcile debug/release times (Heisenberg Principle)
  - extremely SLOW…..
  - Re-run after each improvement/change to RiverWare.
- ✓ Add counters, CPU timers to understand rule set, dispatching, and rules engine behavior.
- ✓ Add CPU timers to time the runs (sans load, initialization time).

Account Solving - Conclusion

Early account solving is not a problem.

All account solving accounts for +/- 3% of the run time.

Rule Set / RPL

Examined rule set.

Observed lots of common subexpressions, possible repeated executions of functions with same arguments.

RPLSAT – extended in-house to show more detail.

- □ More function result caching?
- □ Extend WITH ?
- ✓ Modified rule set to test these concepts – 14-18 % gain

Function-Result Caching Study

Implemented caching for functions of up to N arguments.

Subject to command-line arguments for

- Cache size (# prior results to keep),
- Number of arguments (N) that make a function eligible for caching,
- Process list arguments or not?

Cost of cache management may override benefit.

Not obvious if there is a good default configuration.

Might need GUI support for user control of configuration.

No conclusion; study not finished.

Diagnostics

Quantify Study #1 showed time in argument preparation for (possibly filtered) diagnostics.

- ✓ Re-wrote filtering idiom : 1.8% improvement with diagnostics off, 11.6% improvement with diagnostics on.

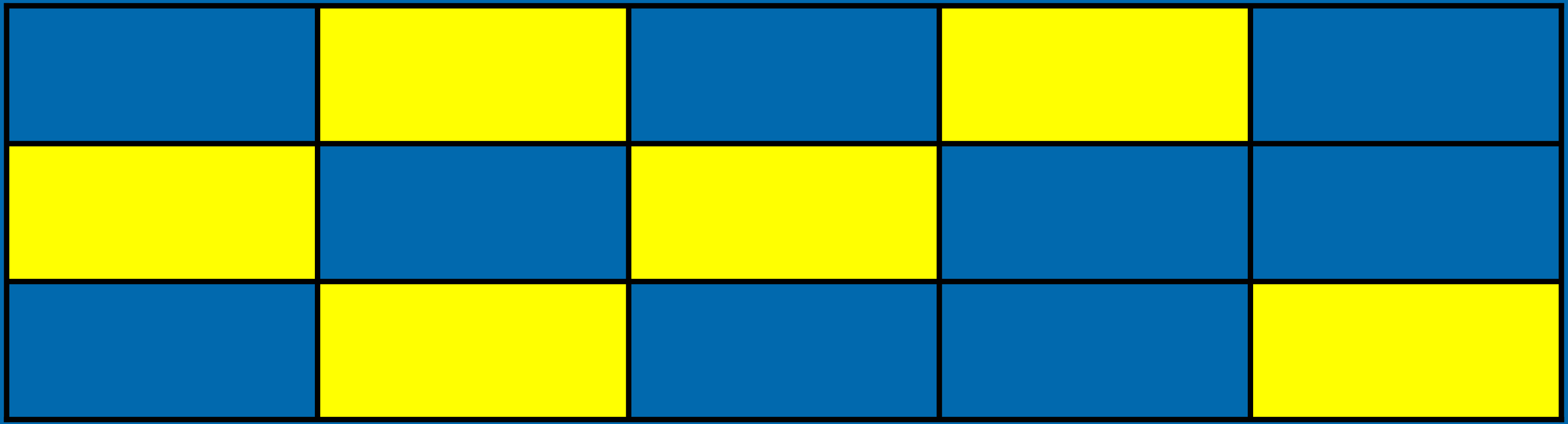
Quantify study #2 showed time in context management.

- ✓ Rewrote Context class – work in progress.

Memory Management - I

- Timing consecutive runs showed increasing time.
- Purify showed no memory leaks.
- Process size not increasing according to OS calls (sbrk(0)).
- Instrumented mm library (-lmalloc) study showed small loss of usable memory with each run.
- Our first observed evidence of memory fragmentation.

What is Fragmentation ?



Caused by requests that are quasi-random in size requested and order of freeing.

May not find large contiguous blocks, despite adequate total available memory. Must request more memory from operating system.

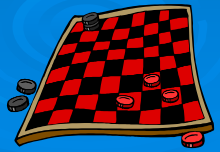
Library has to work increasingly harder to merge/split blocks.

Hypothetical Simulation

CPU Timers showed that model was spending 26-27% of run time in hypothetical simulation.

Quantify ™ study showed that approximately ½ the hyp sim time was spent cloning objects.

- ✓ Save cloned objects, attached them to the subbasin.
- ✓ Expected larger process size, improved times up to 13%
- ✓ Got smaller process size, 30+% improved time. ➔ FRAGMENTATION



Memory Management - II

- ✓ Implemented class factories: a way to reduce the randomness of the load on the mm library.

  Chose two classes used frequently throughout RiverWare, particularly in rules.

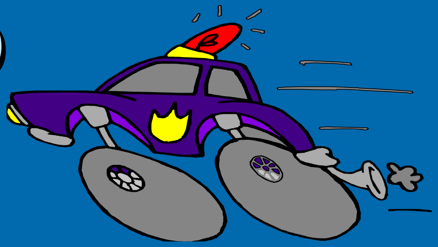
  10% improvement (3-13%)

- □ Apply to more classes in rules processing.

Possibilities

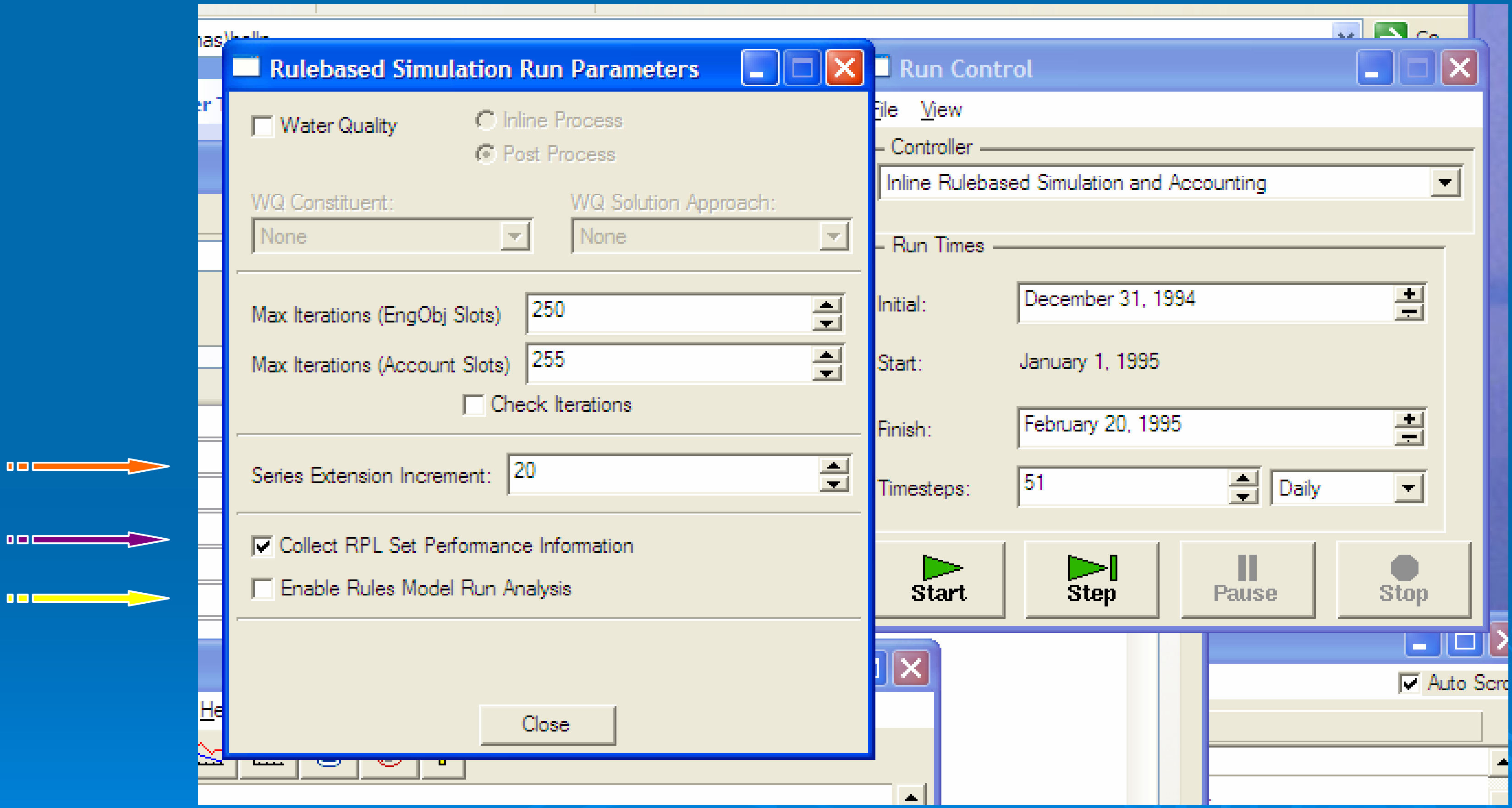
- WITH – extend across entire rule
- Apply class factories to other classes
- Find reasonable configuration for RPL function result caching or design UI to give user control over this
- More predefined functions &/or operators for rules
- Hashing - we know of some improvements in our hash tables to reduce dynamic memory allocation
- More Quantify ™ studies
- Extend RPLSAT for users
- Provide instrumentation to users (GUI)

What Users Can Do Today



- ✓ Turn off diagnostics when not needed.
- ✓ Use argument-less functions when apropos.
- ✓ Attend to Series Extension Increment (new).
- ✓ Turn off RPL Set Performance Information when not needed (new).
- ✓ Turn off Rules Model Run Analysis when not needed (new).

Series Extension Increment, RPL Set Performance, Rules Model Run Analysis



Rulebased Simulation Run Parameters

□ Water Quality

○ Inline Process
● Post Process

WQ Constituent: None

WQ Solution Approach: None

Max Iterations (EngObj Slots): 250

Max Iterations (Account Slots): 255

□ Check Iterations

Series Extension Increment: 20

☑ Collect RPL Set Perfomance Information

□ Enable Rules Model Run Analysis

Close

Run Control

Controller: Inline Rulebased Simulation and Accounting

Run Times

Initial: December 31, 1994

Start: January 1, 1995

Finish: February 20, 1995

Timesteps: 51 Daily

Summary

Performance studies are time-consuming.

Net gains:

- ✓ 11-43% over last release for models w/ hypothetical simulation
- ✓ 13-22% for models w/o hypothetical simulation

More to be gained by work on:

- ❑ WITH
- ❑ RPL function caching
- ❑ More class factories
- ❑ RPL operators, predefined functions

Other potential benefits:

- ❑ Expanded RPLSAT
- ❑ Other instrumentation for users