



IEEE RADAR Conference 2007

Load Balancing on a Multicore Processor

Authors:

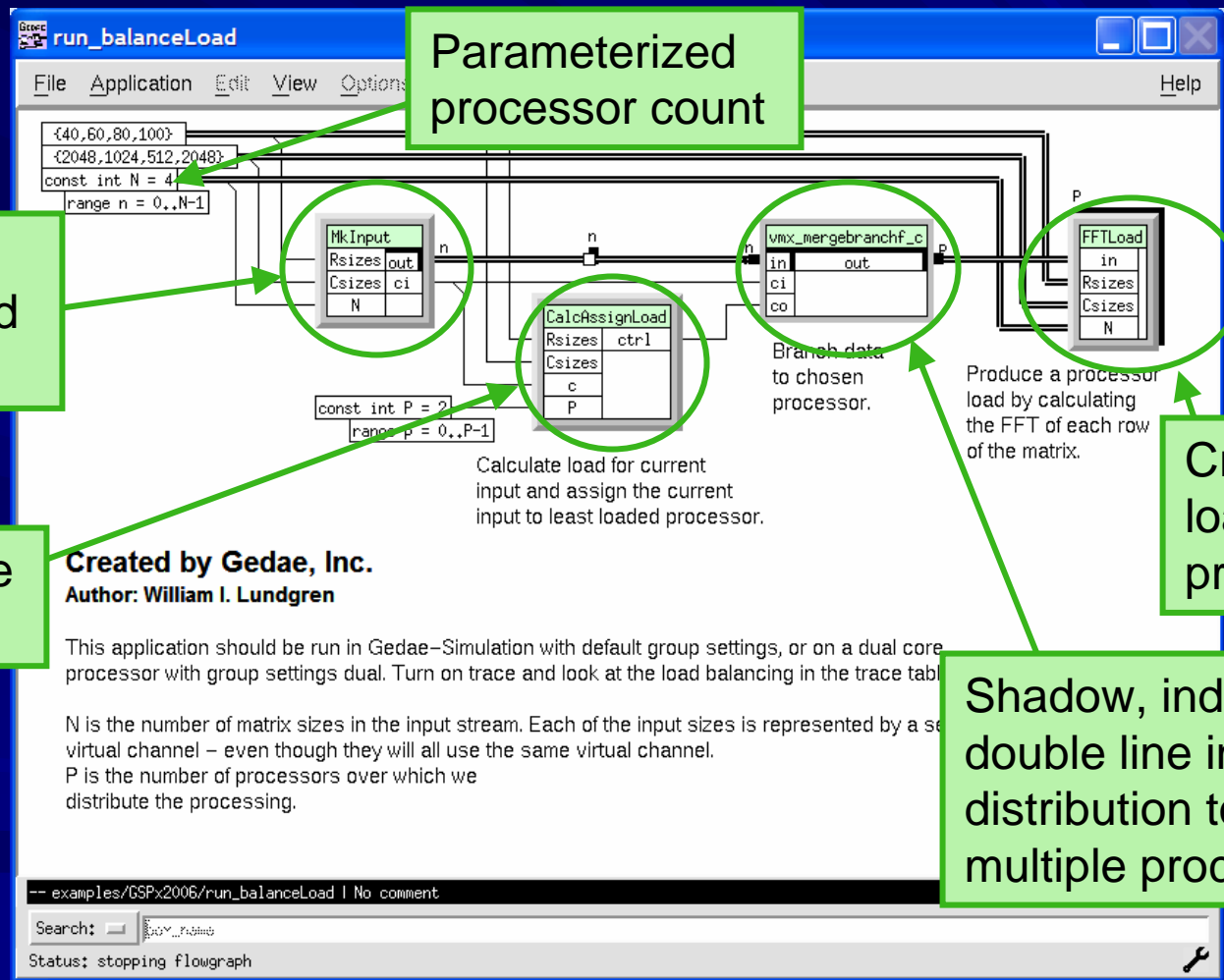
Kerry Barnes, Bill Lundgren, Jim Steed
Gedae, Inc. (Email: bill.lundgren@gedae.com)

Overview of Presentation

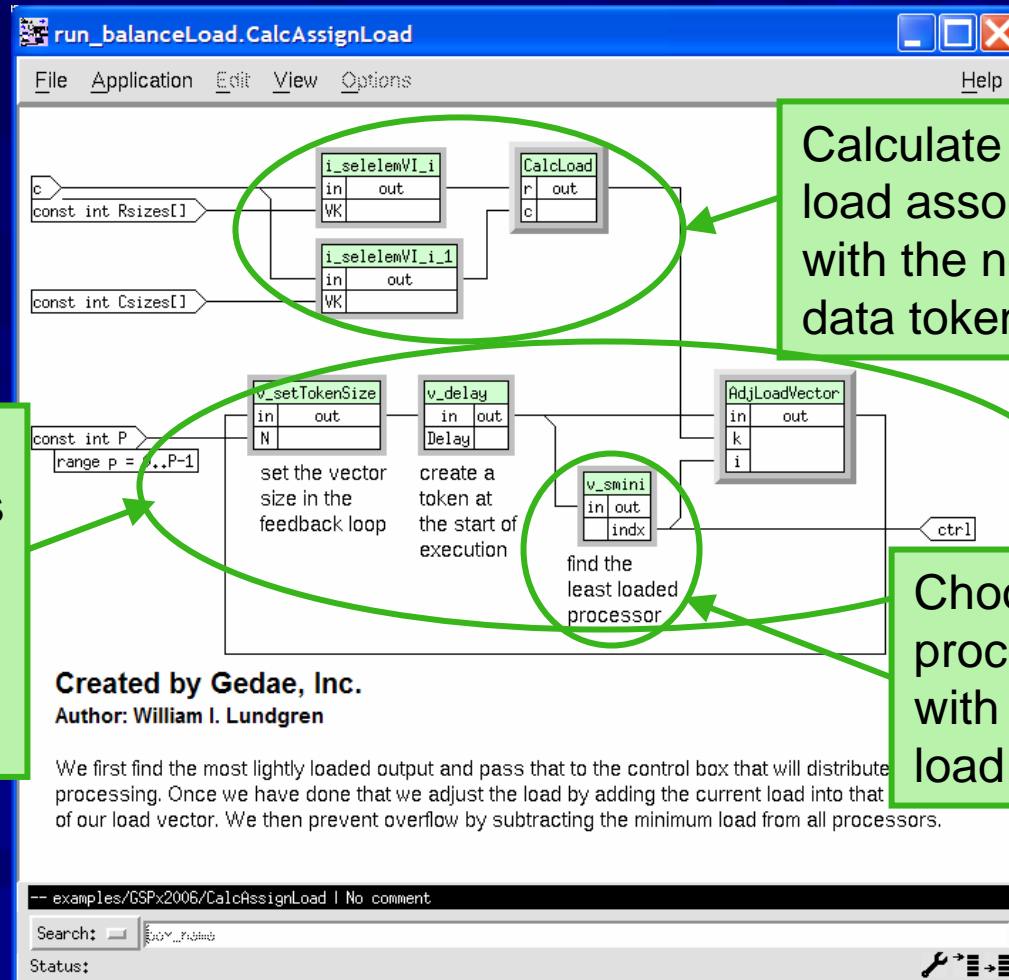


- One of the core requirements of RADAR processing in a dynamic environment is the changing size of the data sets. Combined with the real time system requirements often implies a need for dealing with a dynamically changing processor load.
- This paper explores automatic realtime load balancing using a Gedae graph:
 - Using Gedae's simulation capability, Gedae-SIM, we create a stream of matrices by randomly choosing 1 of 4 sizes.
 - We calculate the expected load on the processor and distribute accordingly.
 - We distribute the processing to 4 (actually a parameterized number) of processor and observe the density of processing.

Top Level Graph Showing Structure of Application



Calculation of Processor Loads

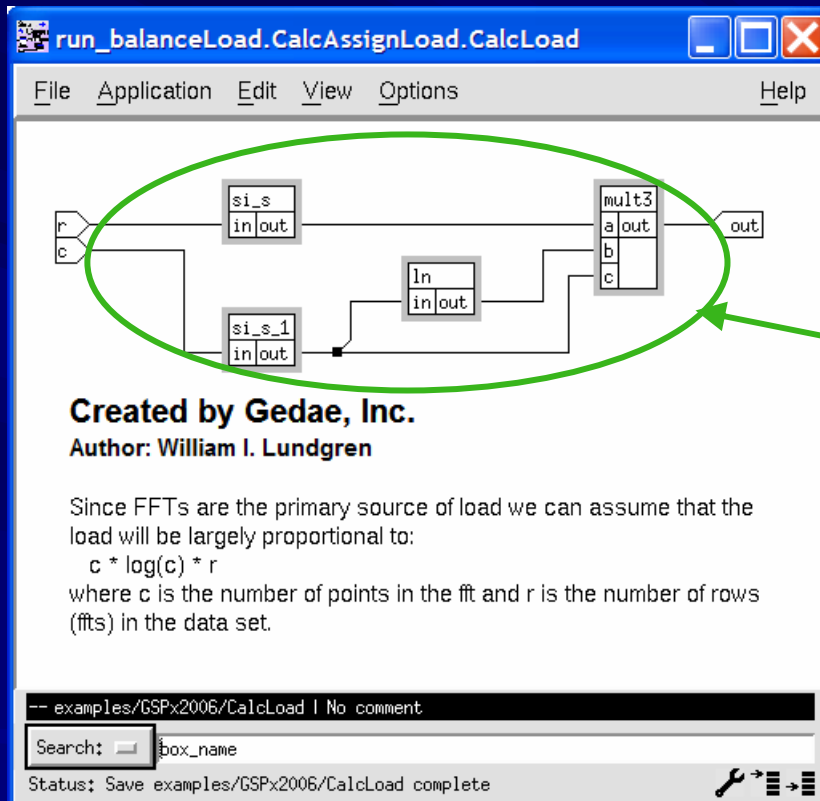


Maintain the processor loads as a vector of data. Add the new load in to the chosen processor.

Calculate the load associated with the new data token

Choose processor with minimum load

Calculating Load



run_balanceLoad.CalcAssignLoad.CalcLoad

File Application Edit View Options Help

Created by Gedae, Inc.
Author: William I. Lundgren

Since FFTs are the primary source of load we can assume that the load will be largely proportional to:
 $c * \log(c) * r$
where c is the number of points in the fft and r is the number of rows (ffts) in the data set.

-- examples/GSPx2006/CalcLoad | No comment

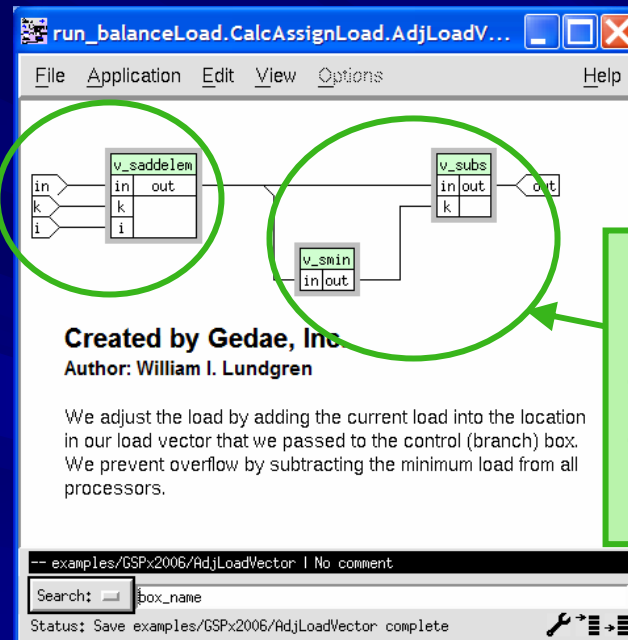
Search: box_name

Status: Save examples/GSPx2006/CalcLoad complete

Calculate the load based on the number of FFTs, r , and then size of the FFT, c . The load is proportional to:
 $load = r * c * \ln(c)$

Add Load into Load Vector and Adjust

Add scalar load to
ith element of the
vector.

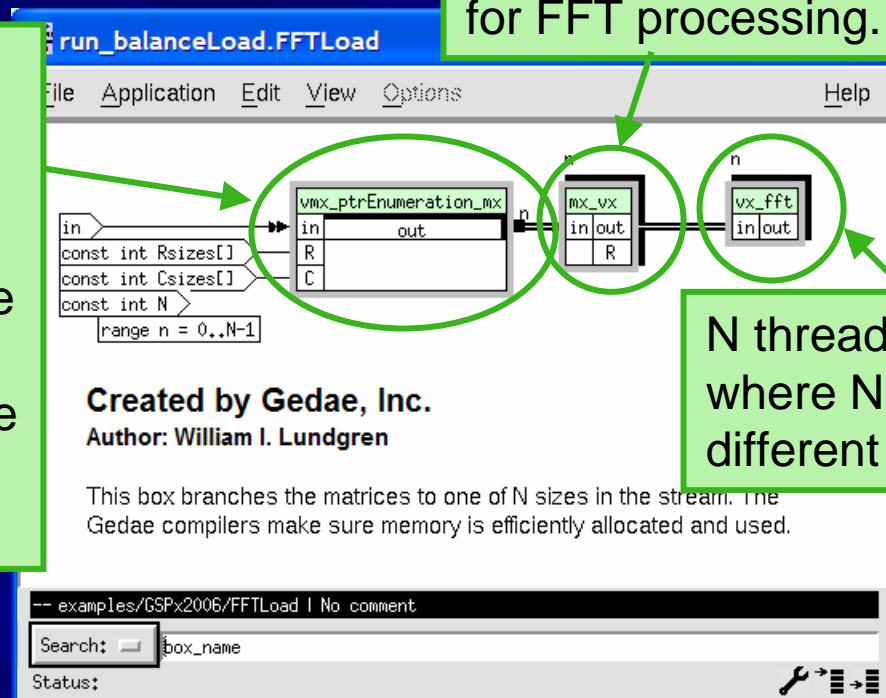


Find the minimum
load and subtract
from all elements of
the vector to prevent
unbounded growth.

Processor Load – FFT on Each Row

Convert the rows of the matrix to vectors for FFT processing.

Pass data to the thread set up with the appropriate data size. The size is passed along the arc during compilation and the data is passed during runtime.



N threads of processing where N is the number of different data sizes.

Created by Gedae, Inc.
Author: William I. Lundgren

This box branches the matrices to one of N sizes in the stream. The Gedae compilers make sure memory is efficiently allocated and used.

-- examples/GSPx2006/FFTLoad | No comment

Search: box_name

Status:

Creating the Application



- The application functionality is complete. While our simple example only shows a few boxes, modern complex systems often require 100s, 1000s or 10,000s of function boxes to complete the application.
- The implementation specification tools are designed to handle applications with that sort of complexity.
- In this example we will specify:
 - How to partition the functionality for mapping to processors
 - Which processor each partition will run on
- We also note that the transfer buffer sizes are left to be the default size, in this case 1024 bytes.

Specifying the Implementation

Group 1 Partiti...

Name	Part	Sub:
MkInput	h *	
CalcAssignLoad	h *	
vmx_mergebranchf_	h *	
[0]FFTLoad	p0="p"+\$1	
[1]FFTLoad	p1="p"+\$1	

Partition the functionality into 3 partitions named h, p0 and p1. Note that equations simplify the implementation specification.

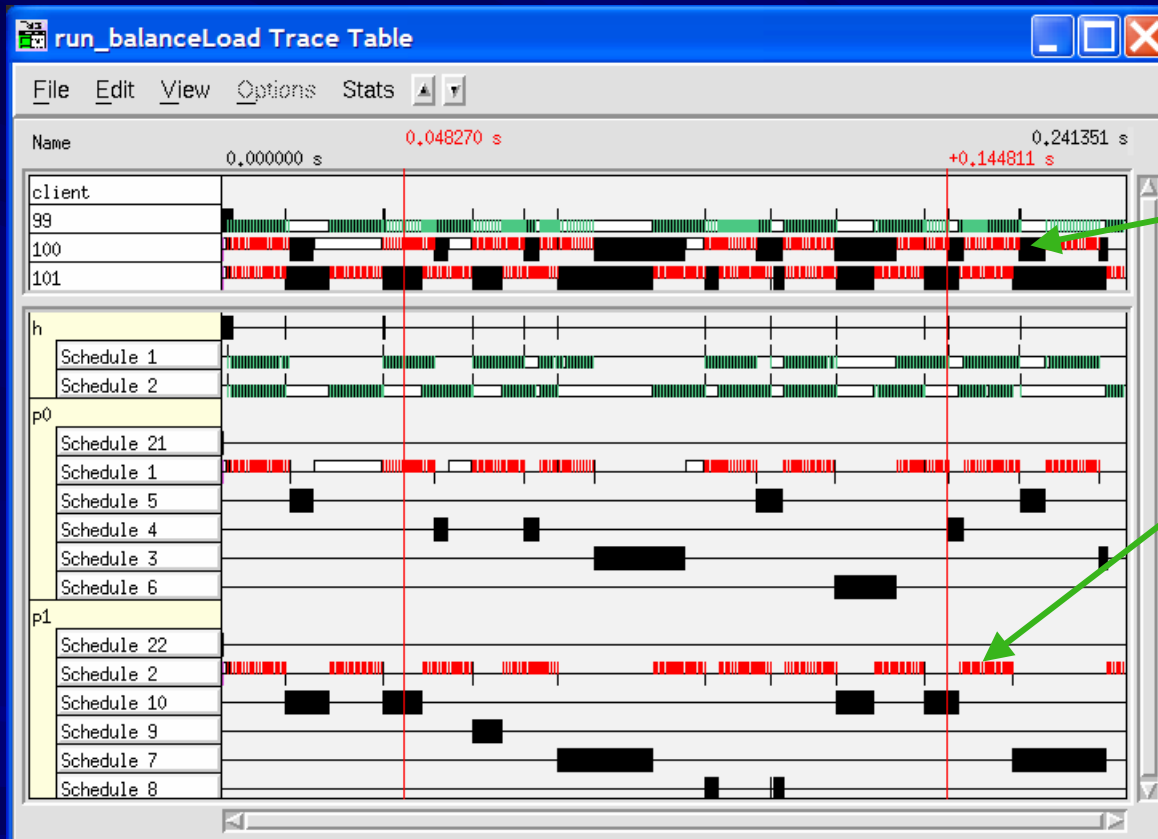
Partition Table

Name	CP	ProcNum	System Name	Trace Size	Trace MemType	Params
h	99	*	gsim_host	1000000 *	default	
p0	100	100+\$1	gsim_host	1000000 *	default	
p1	101	101+\$1	gsim_host	1000000 *	default	

Name	Id	Source	Dest	Xfer Type	NBsize	Send
[0]FFTLoad	vmx_ptrEnumeration_mx<in	host	100	host>gsim_host	0	
	c	host	100	host>gsim_host	0	
	r	host	100	host>gsim_host	0	
[1]FFTLoad	vmx_ptrEnumeration_mx<in	host	101	host>gsim_host	0	
	c	host	101	host>gsim_host	0	
	r	host	101	host>gsim_host	0	

Map the partitions to processors 99, 100 and 101 as shown. Note that once again we used an equation to simplify the mapping.

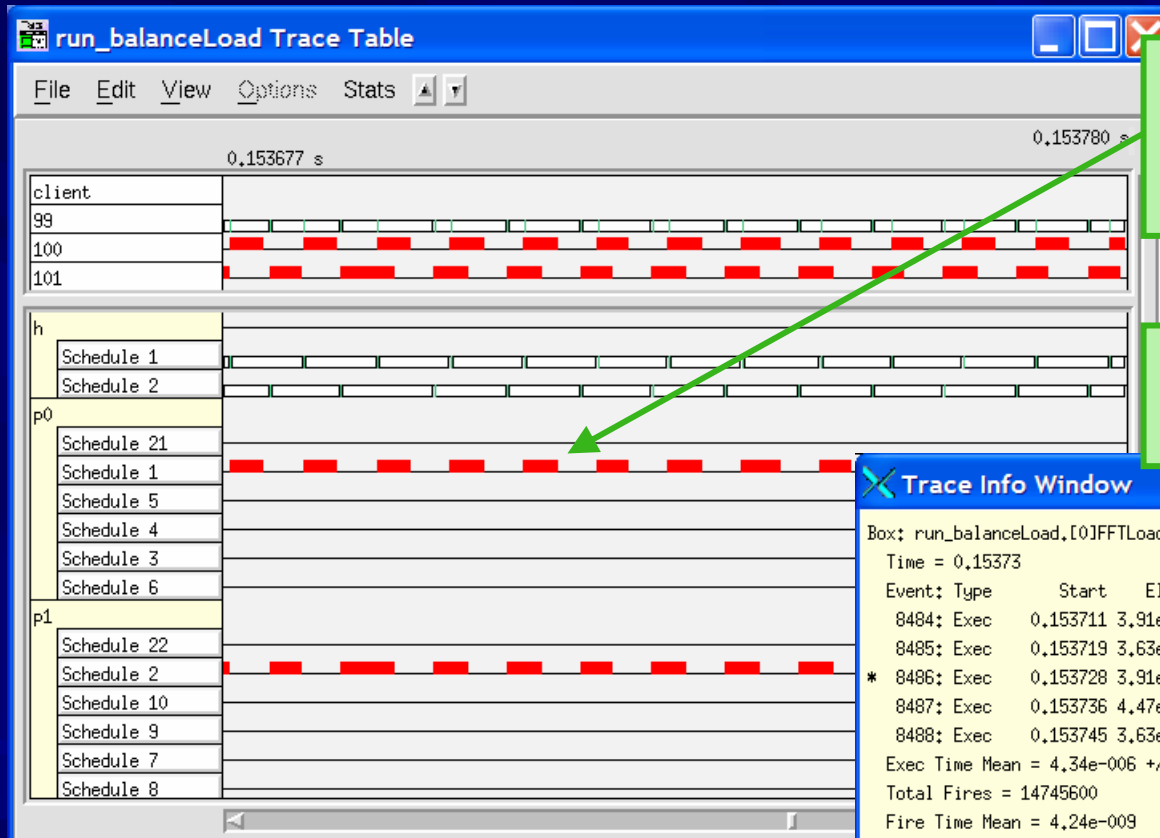
Trace Table With Default Transfer Methods



Load balancer is distributing work, but utilization is low

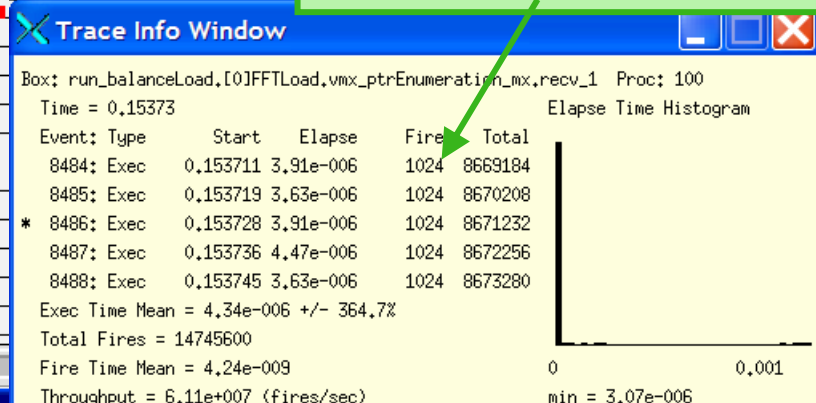
Too much time spent in communication

Blocking Transfers at a Low Granularity



Send boxes must execute many times to transfer data

Granularity is only 1024 tokens



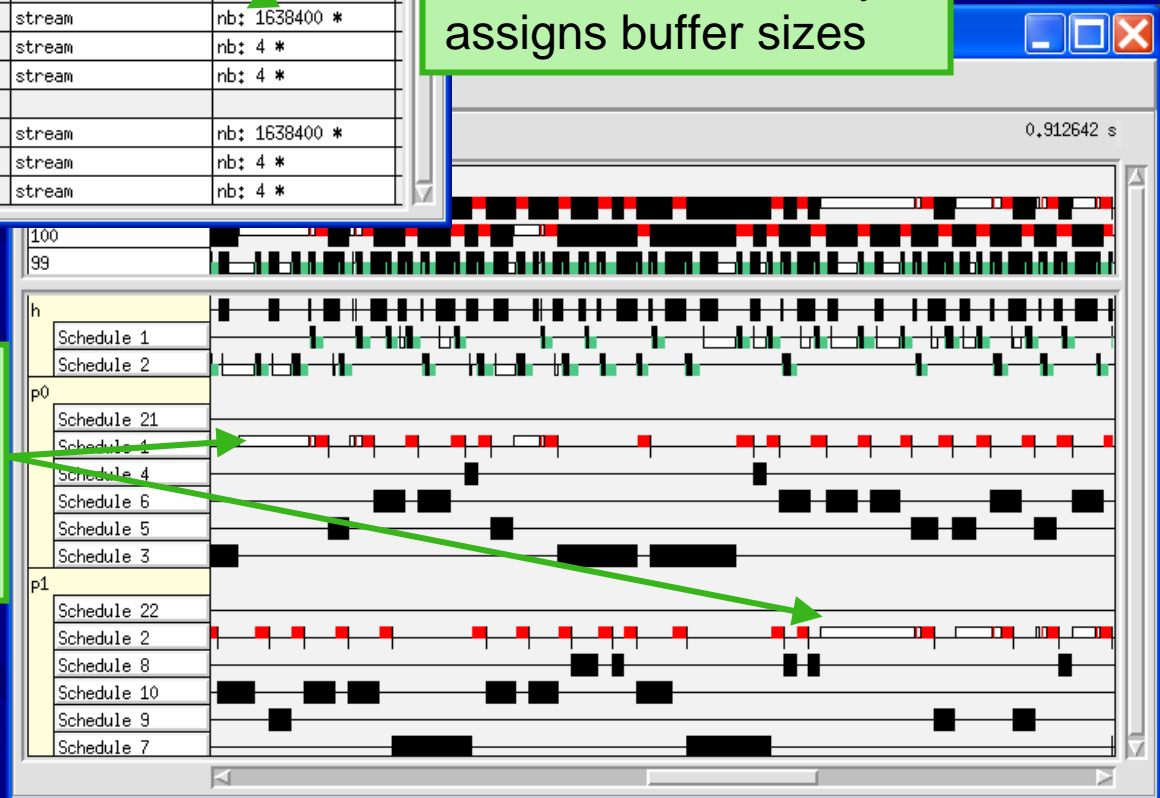
Nonblocking Communication Between Processors

Group 1 Transfer Table

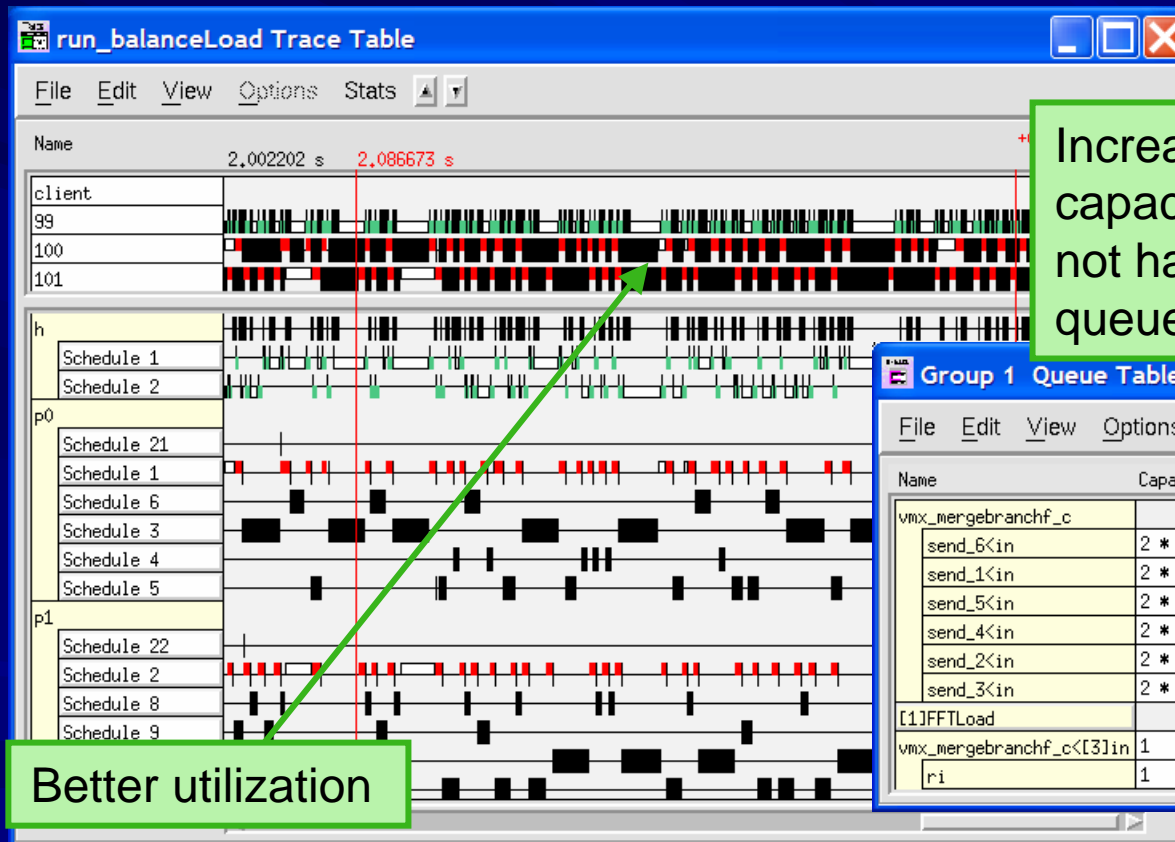
Name	Id	Source	Dest	Xfer Type	NBsize
[0]FFTLoad					
vmx_ptrEnumeration_mx<in	1	99	100	stream	nb: 1638400 *
c	5	99	100	stream	nb: 4 *
r	6	99	100	stream	nb: 4 *
[1]FFTLoad					
vmx_ptrEnumeration_mx<in	2	99	101	stream	nb: 1638400 *
c	3	99	101	stream	nb: 4 *
r	4	99	101	stream	nb: 4 *

Set transfer to nonblocking, and Gedae automatically assigns buffer sizes

Better utilization, but still room for improvement as sends are blocking

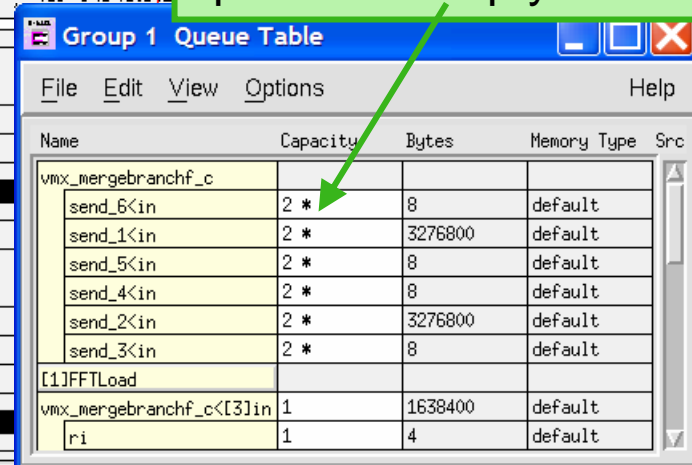


Increase Queue Capacities to Achieve Better Utilization



Increase queue capacity so sends do not have to wait for queues to empty

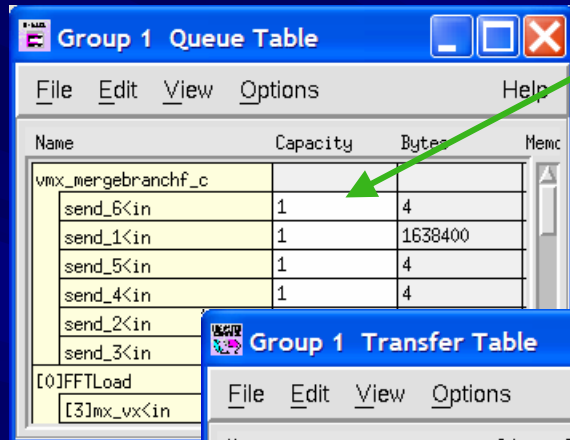
Better utilization



The figure shows a 'Group 1 Queue Table' window. It displays a table of queue configurations. The table has columns for Name, Capacity, Bytes, Memory Type, and Src. The table lists several queues, including vmx_mergebranchf_c, send_6<in, send_1<in, send_5<in, send_4<in, send_2<in, send_3<in, [1]FFTLoad, vmx_mergebranchf_c<[3]in, and ri.

Name	Capacity	Bytes	Memory Type	Src
vmx_mergebranchf_c				
send_6<in	2 *	8	default	
send_1<in	2 *	3276800	default	
send_5<in	2 *	8	default	
send_4<in	2 *	8	default	
send_2<in	2 *	3276800	default	
send_3<in	2 *	8	default	
[1]FFTLoad				
vmx_mergebranchf_c<[3]in	1	1638400	default	
ri	1	4	default	

Alternatively, Increase Nonblocking Buffer Size

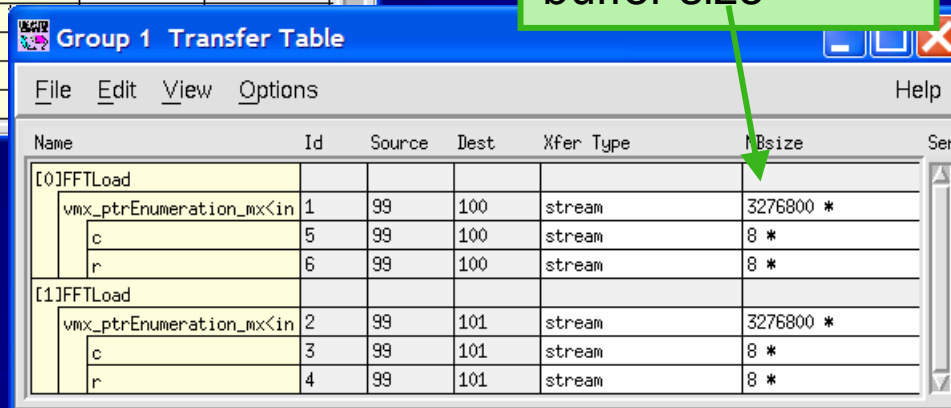


Group 1 Queue Table

Name	Capacity	Bytes	Memc
vmx_mergebranchf_c			
send_6<in	1	4	
send_1<in	1	1638400	
send_5<in	1	4	
send_4<in	1	4	
send_2<in			
send_3<in			
[0]FFTLload			
[3]mx_vx<in			

If Capacity is left at 1

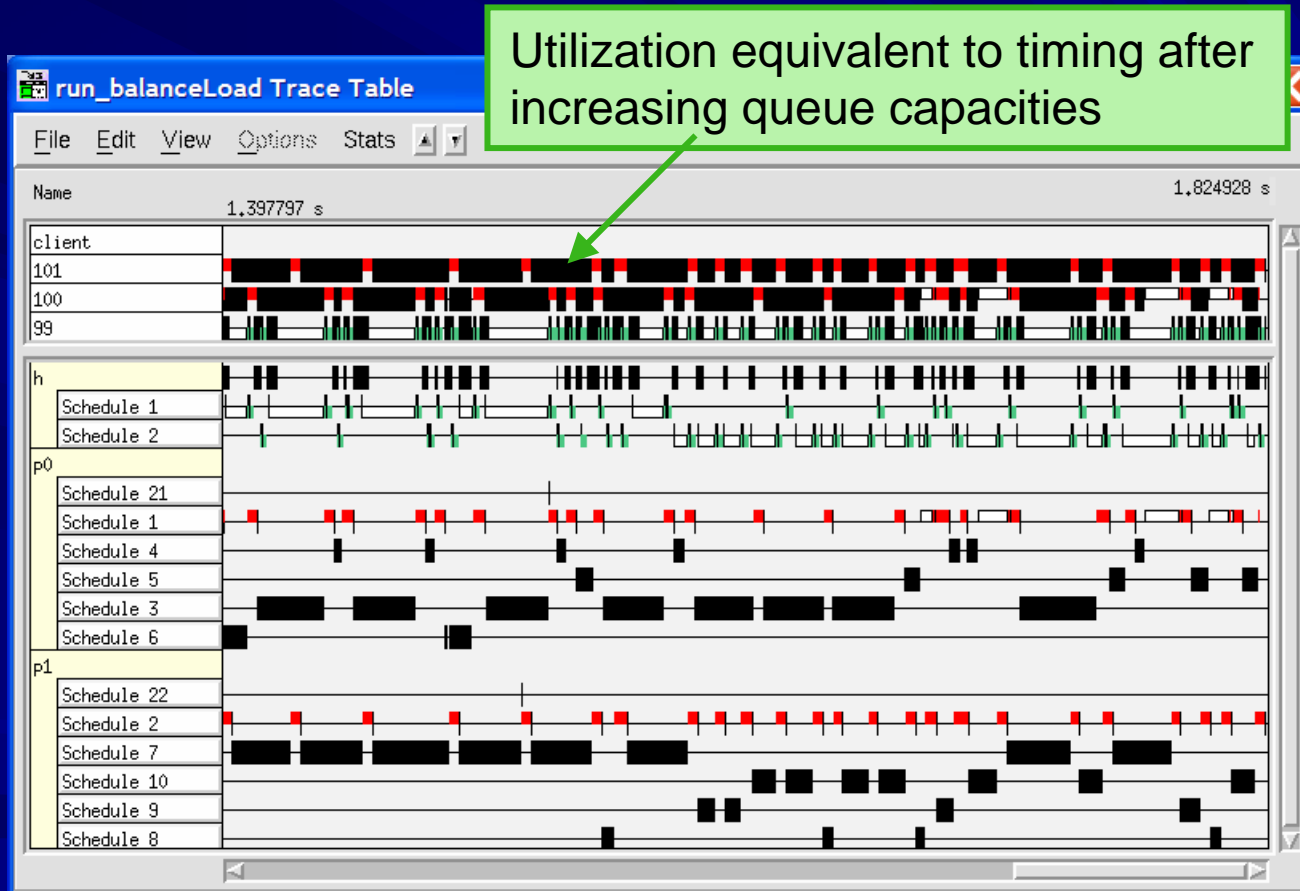
Instead double nonblocking buffer size



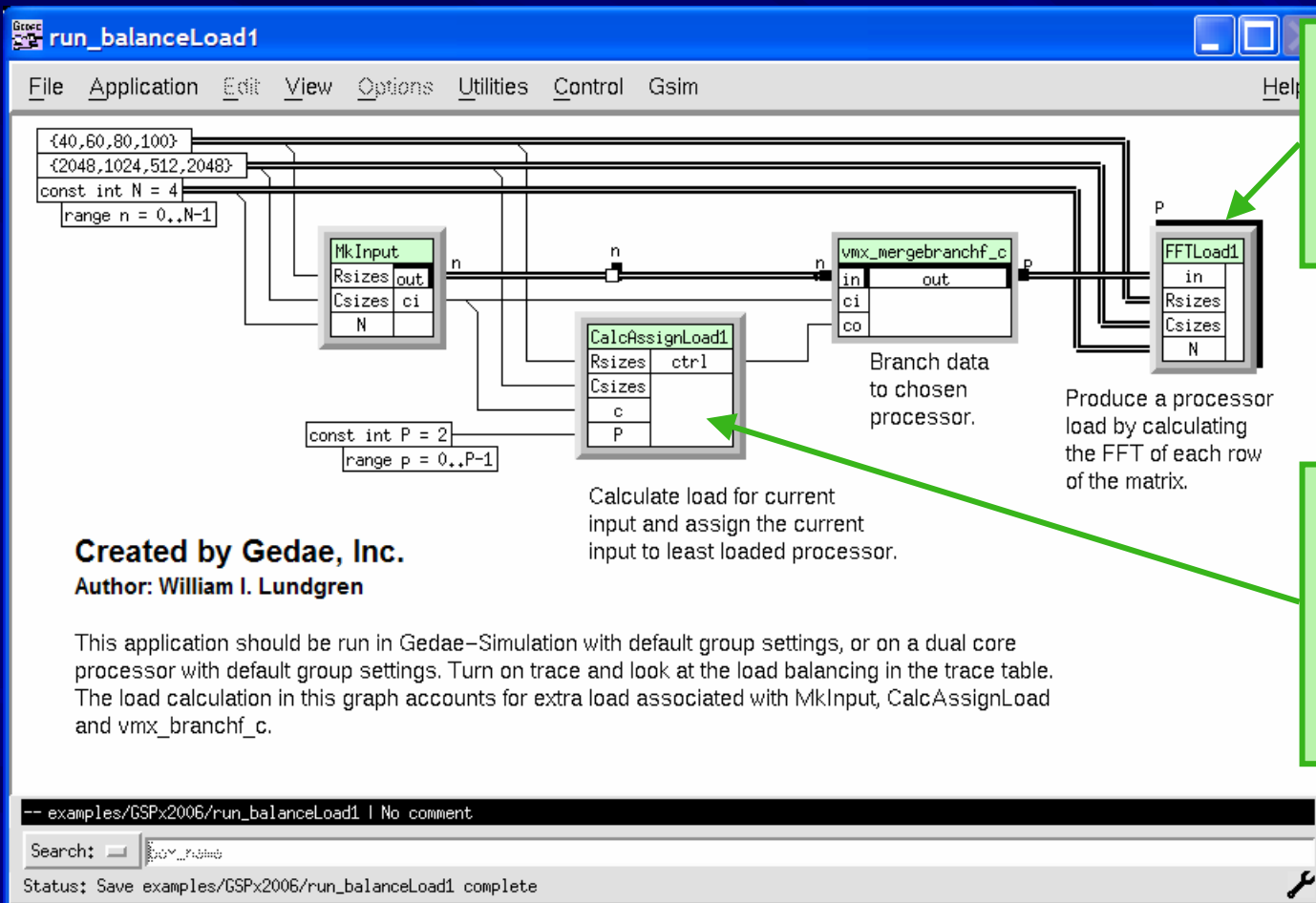
Group 1 Transfer Table

Name	Id	Source	Dest	Xfer Type	Bsize	Ser
[0]FFTLload						
vmx_ptrEnumeration_mx<in	1	99	100	stream	3276800 *	
c	5	99	100	stream	8 *	
r	6	99	100	stream	8 *	
[1]FFTLload						
vmx_ptrEnumeration_mx<in	2	99	101	stream	3276800 *	
c	3	99	101	stream	8 *	
r	4	99	101	stream	8 *	

Increasing Nonblocking Buffer Size Provides the Same Utilization Gain



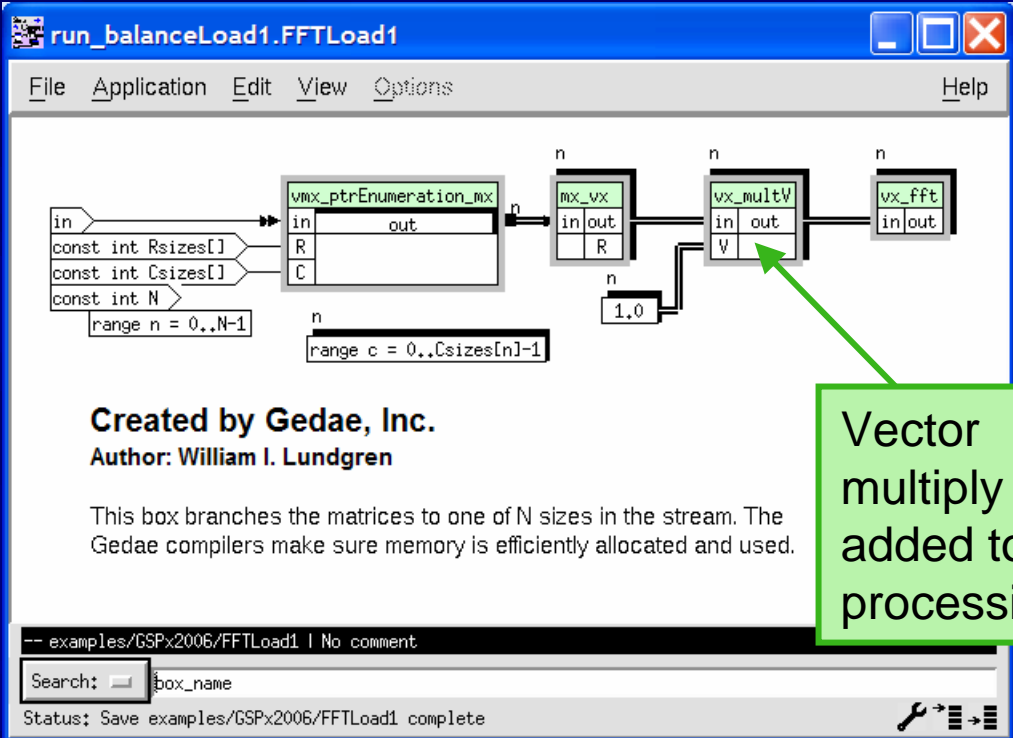
Top Level Graph for 2nd Load Balancing Example



Incorporate a vector multiply to the FFTLoad subgraph

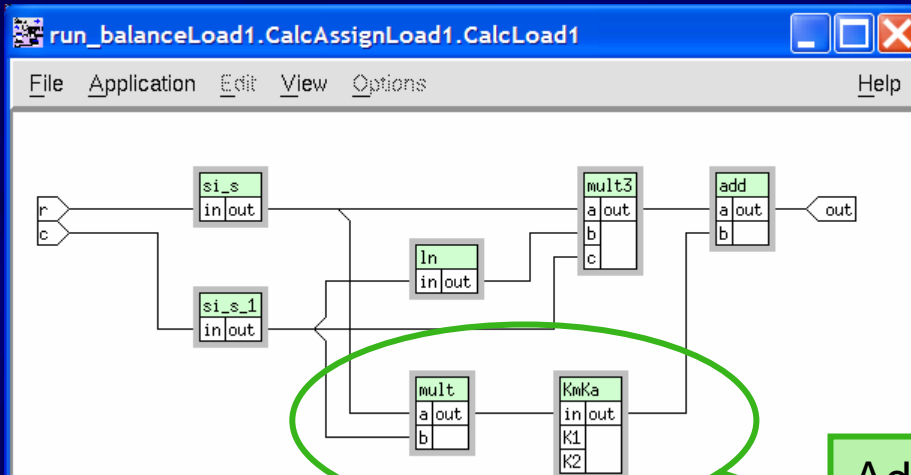
Adjust CalcAssignLoad to take account for the vector multiply

Processor Load – FFT and Vector Multiply on Each Row



Vector multiply added to processing

Calculating Load



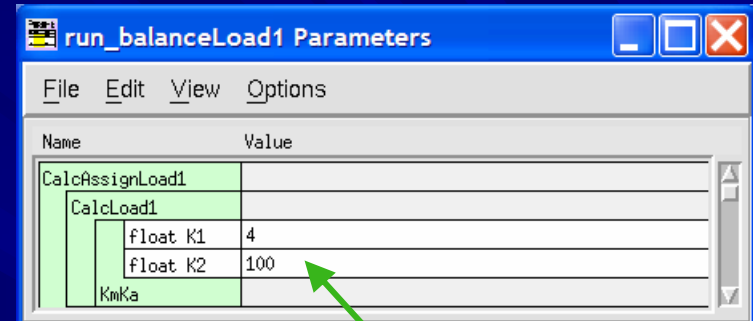
Created by Gedae, Inc.
Author: William I. Lundgren

The FFTs are an important source of load of the load on all processors we can assume that that part of the load will be largely proportional to:
 $c * \log(c) * r$
where c is the number of points in the fft and r is the number of rows (ffts) in the data set.
The remaining load will be associated with the vector multiply and is proportional to the number of elements in the matrix. We will calculate this load add it to load from the FFTs.

```
-- examples/GSPx2006/CalcLoad1 | No comment
```

Search:

Status: Save examples/GSPx2006/CalcLoad1 complete



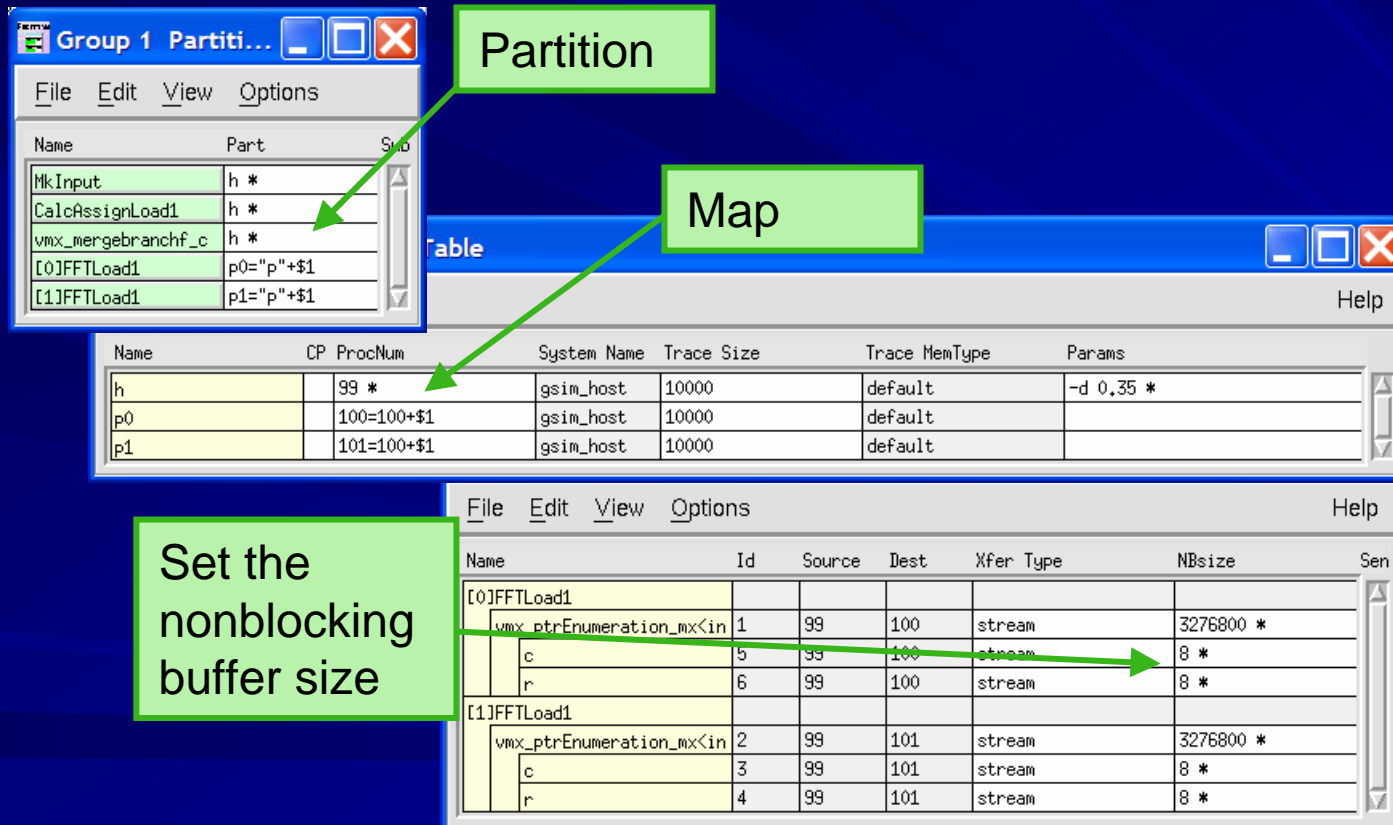
Name	Value
CalcAssignLoad1	
CalcLoad1	
float K1	4
float K2	100
KnKa	

Add approximation of time for vector multiply to heuristic:
 $load = r * c * \ln(c) + K1 * r * c + K2$

Use Parameter Table to set and experiment with scale factors for the heuristic

Specifying the Implementation

- Apply same implementation as previous example



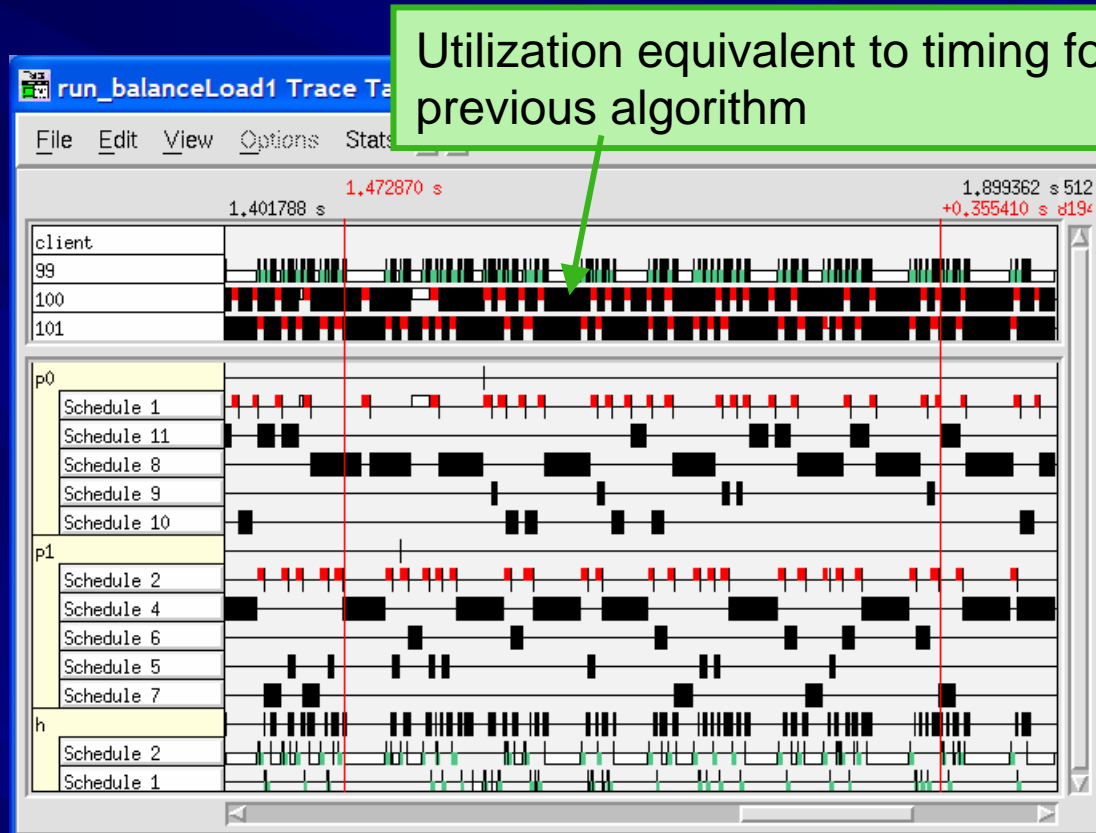
The screenshot shows the Geda interface with three windows and annotations:

- Partition Table:** A table with columns Name, Part, and Sub. It lists components like MkInput, CalcAssignLoad1, vmx_mergebranchf_c, [0]FFTLoad1, and [1]FFTLoad1.
- Map Table:** A table with columns Name, CP, ProcNum, System Name, Trace Size, Trace MemType, and Params. It lists processes h, p0, and p1.
- Transfer Table:** A table with columns Name, Id, Source, Dest, Xfer Type, NBsize, and Sen. It shows data flow between [0]FFTLoad1 and [1]FFTLoad1.

Annotations include:

- A green box labeled "Partition" with an arrow pointing to the Partition table.
- A green box labeled "Map" with an arrow pointing to the Map table.
- A green box labeled "Set the nonblocking buffer size" with an arrow pointing to the NBsize column in the Transfer table.

Same Strategy Achieves Good Utilization for New Algorithm



Conclusions



- Gedae provides a graphical method for describing load balancing of parallel processing
- The Gedae Trace Table is invaluable for diagnosing problems with the parallel implementation
- Gedae allows for optimization of load balancing through the setting of transfer methods and changes to the queue capacity and nonblocking buffer sizes
- Applications can be quickly changed, quickly regenerated, and quickly relaunched