NAME

lqns − solve layered queueing network models

SYNOPSIS

lqns [**−MnpvV**] [**−d** *debug*] [**−e adiw**] [**−o** *output*] [**−t** *trace*] [**−z** *special*] [**−P** *pragma*] [filename . . .]

DESCRIPTION

Lqns is used to solve layered queueing network models using analytic techniques. Refer to *"The Stochastic Rendezvous Network Model for Performance of Synchronous Multi-tasking Distributed Software"* for details of the model. Refer to *"SRVN Input File Format"* for a complete description of the input file format for the programs. See below for program restrictions and limits.

Lqns reads its input from *filename*, specified at the command line if present, or from the standard input otherwise. Output for an input file *filename* specified on the command line will be placed in the file *filename***.out**. If several files are named, then each is treated as a separate model and output will be placed in separate output files. If input is from the standard input, output will be directed to the standard output. The file name '−' is used to specify standard input.

The **−o***output* option can be used to direct all output to the file *output* regardless of the source of input. Multiple input files cannot be specified when using this option. Output can be directed to standard output by using **−o**− (i.e., the output file name is '−'.)

OPTIONS

−d Turn on debugging output.

all Activate all debugging options.

activities
Print out activity service times.

calls Print rendezvous rates from clients to servers during each iteration of the solver.

forks Print out information about forks found in model.

interlock
Print out interlocking tables during initialization.

joins Print out information about joins found in model.

layers Print out construction of layers on standard output during initialization.

overtaking
Print out overtaking probability calculations during each iteration of the solver.

variance
Print out the service time variance during iteration of the solver.

See also **-t**.

−e adiw
Floating point error disposition. By default, the floating point errors overflow, divide by zero and invalid operation, will stop the solution of the current model. The default case, **−ed**, uses the imprecise exception model of the floating point hardware to permit the fastest possible execution. The exact exception model, useful when debugging the solver, is used by selecting **−ea** (see **fpsetdefaults(3)** (hp) or **floating_point(3)** (sun)). Selecting **−ei** causes floating point errors to be ignored. **−ew** prints a warning on floating point error (not implemented at present).

−M Do not merge the send-no-reply waiting time with the rendezvous delay in the parseable output. The default in this release is to merge the results because some tools do not support the new fields in the parseable output.

−n Read input, but do not solve. The input is checked for validity. No output is generated.

−o *output*
: Direct analysis results to *output*. A filename of '−' directs output to standard output. If **lqns** is invoked with this option, only one input file can be specified.

−p
: Generate parseable output suitable as input to other programs such as **multisrvn** and **srvndiff**(1). If input is from *filename*, parseable output is directed to *filename***.p**. If standard input is used for input, then the parseable output is sent to the standard output device. (In this case, only parseable output is emitted.) If the **-o** *output* option is used, the parseable output is sent to a file whose name is derived from *output* by the addition of the suffix **.p**. If a suffix already exists on *output* then that suffix is replaced.

−P *pragma*
: Change the default solution strategy. Refer to the PRAGMAS section below for more information.

-r
: Rename all of the tasks, entries, activities and processors to generic names starting with *t1*, *e1*, *a1* and 1. This option is useful for 'sanitizing' the output by changing the names of everthing.

−t *trace*
: Tracing options, used to print out various intermediate results while a model is being solved.

activities
: Print out results of activity aggregation.

convergence
: Print out convergence value after each submodel is solved. This option is useful for tracking the rate of convergence for a model.

delays
: Print waiting time for each rendezvous in the model after it has been computed.

delta-wait
: Print out difference in entry service time after each submodel is solved.

forks
: Print out overlap table for forks prior to submodel solution.

idle-time
: Print out computed idle time after each submodel is solved.

interlock
: Print out interlocking adjustment before each submodel is solved.

joins
: Print out computed join delay and join overlap table prior to submodel solution.

mva
: Print out MVA submodel and its solution.

overtaking
: Print out overtaking calculations.

print
: Print out intermediate solutions at the print interval specified in the model. The print interval field in the input is ignored otherwise.

processor=*name*
: Print out results for processor *name* after each submodel is solved. *Name* can be a regular expression. Not implemented.

task=*name*
: Print out results for task *name* after each submodel is solved. *Name* can be a regular expression. Not implemented.

variance
: Print out calculated variances after each submodel is solved.

−v
: Print out statistics about the solution on the standard output device.

−V
: Print out version and copyright information.

−z *special-opts*
This flag is used to select special options. Arguments of the form *nn* are integers while arguments of the form *nn.n* are real numbers.

convergence=*nn.n*
Set the convergence value to *nn.n*.

iteration-limit=*nn*
Set the maximum number of iterations to *nn*. *nn* must be greater than 0. The default value is 50.

generate=*file*
Generate MVA submodel C++ input files, one for each submodel. The filename ...

mol-underrelaxation

overtaking
Print out overtaking probabilities.

print-interval=*nn*
Set the printing interval to *nn*. The **−d** or **−v** options must also be selected to display intermediate results. The default value is 10.

skip=*n* Ignore submodel *n* during solution.

step Stop after each MVA submodel is solved. Any character typed at the terminal except end-of-file will resume the calculation. End-of-file will cancel single-stepping altogether.

underrelaxation=*nn.n*
Set the underrelaxation to *nn.n*. The default value is 0.9.

If any one of *convergence*, *iteration-limit*, or *print-interval* are arguments used, the corresponding value specified in the input file for general information, ‘G’, is ignored.

PRAGMAS

Pragmas are used to alter the behaviour of the solver in a variety of ways. They can be specified in the input file with “#pragma”, on the command line with the **−P** option, or through the environment variable *LQNS_PRAGMAS*. Command line specification of pragmas overrides those defined in the environment variable which in turn override those defined in the input file. The following pragmas are supported. Invalid pragma specification at the command line will stop the solver. Invalid pragmas defined in the environment variable or in the input file are ignored as they might be used by other solvers.

cycles=*{disallow,allow}*

disallow
Disallow cycles in the call graph. Cycles may indicate the presence of dead locks.

allow Allow cycles in the call graph.

The default is *disallow*.

interlocking=*{none,throughput}*

none Do not perform interlock adjustment.

throughput
Perform interlocking by adjusting throughputs.

The default is *throughput*.

layering=*{batched,batched-back,loose,squashed,strict,strict-back}*

batched
Batched layering -- solve layers composed of as many servers as possible from top to bottom.

batched-back
Batched layering with back propagation -- solve layers composed of as many servers as possible from top to bottom, then from bottom to top to improve solution speed.

loose Loose layers -- solve layers composed of only one server. This method of solution is comparable to the technique used by the **srvn** solver. See also *-Pmva*.

squashed
Squashed layers -- All the tasks and processors are placed into one layer. Solution speed may suffer because this method generates the most number of chains in the MVA solution. See also *-Pmva*.

strict Strict layers -- solve layers using the Method of Layers. Layer spanning is performed by allowing clients to appear in more than one layer.

strict-back
Strict layers -- solve layers using the Method of Layers. Software submodels are solved top-down then bottom up to improve solution speed.

The default layering technique is *batched-back*.

multiserver=*{default,conway,reiser,reiser-ps,rolia,rolia-ps,bruell,schmidt}*

default The default multiserver calculation uses the the Conway multiserver for multiservers with less than five servers, and the Rolia multiserver otherwise.

conway
Use the Conway multiserver calculation for all multiservers.

reiser Use the Reiser multiserver calculation for all multiservers.

reiser-ps
Use the Reiser multiserver calculation for all multiservers. For multiservers with multiple entries, scheduling is processor sharing, not FIFO.

rolia Use the Rolia multiserver calculation for all multiservers.

rolia-ps
Use the Rolia multiserver calculation for all multiservers. For multiservers with multiple entries, scheduling is processor sharing, not FIFO.

schmidt
Use the Schmidt multiserver calculation for all multiservers.

bruell Use the Bruell multiserver calculation for all multiservers.

mva=*{exact,schweitzer,linearizer,fast,one-step,one-step-linearizer}*

exact Exact MVA. Not suitable for large systems.

schweitzer
Bard-Schweitzer approximate MVA.

linearizer
Linearizer.

one-step
Perform one step of Bard Schweitzer approximate MVA for each iteration of a submodel. The default is to perform Bard Schweitzer approximate MVA until convergence for each submodel. This option, combined with *-Playering=loose* most closely approximates the solution technique used by the **srvn** solver.

one-step-linearizer
Perform one step of Linearizer approximate MVA for each iteration of a submodel. The default is to perform Linearizer approximate MVA until convergence for each submodel.

The default MVA solver is *linearizer*.

overtaking=*{markov,simple}*

markov
Markov phase 2 calculation.

simple Simpler, but faster approximation.

The default overtaking calculation is *markov*.

processor=*{default, fcfs, hol, ppr, ps, ps-hol, ps-ppr}*
Force the scheduling type of all uni-processors to the type specfied.

default The scheduling type is determined by the input file.

fcfs All uni-processors are scheduled first-come, first-served. **hol** All uni-processors are scheduled using head-of-line priority. **ppr** All uni-processors are scheduled using priority, pre-emptive resume. **ps** All uni-processors are scheduled using processor sharing. **ps-hol** Use high variability servers for processor where warranted. **ps-ppr** Use high variability servers for processor where warranted.

stop-on-message-loss=*{false,true}*
Determine the action if messages are lost at servers with open arrivals or send-no-reply interactions.

true Stop if messages are lost.

false Ignore lost messages. Waiting times are reported as infinite.

The default is **true**.

reschedule=*{slice,call}*
Reschedule the CPU after each slice (default) or call (not implemented).

threads=*{none,default,hyper,mak}*

none Do not perform overlap calculation for forks.

default Use three-point approximation for join delays.

hyper Inflate overlap probabilities based on arrival instant estimates.

mak Use Mak-Lundstrom approximations for join delays.

variance=*{mol,mol-entry,none,srvn}*

mol Use mol variance calculation.

mol-entry
Use mol variance calculation. Do not blend variances for final waiting time calculation.

none Disable variance adjustment. All stations in the MVA submodels are either delay or first-come-first-served.

srvn Use srvn variance calculation. This pragma must be used if deterministic phases are used in the model.

The default variance calculation is *mol*.

MODEL LIMITS

The following table lists the acceptable parameter types for **lqns**, **petrisrvn**(1), and **parasrvn**(1). An error will be reported if an unsupported parameter is supplied except when the value supplied is the same as the default.

Parameter	lqns	parasrvn	petrisrvn
Phases	3	3	3
Scheduling	F	FH	FHPR
Open arrivals	yes	yes	no
Phase type	SD	SD	S
Coefficient of variation	yes	yes	no
Think Time	yes	yes	yes
Interprocessor-delay	no	yes	yes
Asynchronous connections	yes	yes	no
Forwarding	yes	yes	yes
Multi-servers	yes	yes	yes
Infinite-servers	yes	yes	no
Max Entries	1000	--	30
Max Tasks	1000	1000	15
Max Processors	1000	1000	15
Max Entries per Task	--	--	--
Scheduling:	F: FIFO, H: Head-of-Line, P: Priority, R: Random		
Phase type:	S: Stochastic, D: Deterministic		

DIAGNOSTICS

Most diagnostic messages result from errors in the input file. If the solver reports errors, then no solution will be generated for the model being solved. Models which generate warnings may not be correct. However, the solver will generate output.

BUGS

XML input and output is not implemented.

SEE ALSO

C. M. Woodside et. al., ‘‘The Stochastic Rendezvous Network Model for Performance of Synchronous Multi-tasking Distributed Software’’, *IEEE Trans. Comp.*, Vol. 44, No. 8, Aug 1995, pp. 20-34.

J. A. Rolia and K. A. Sevcik, ‘‘The Method of Layers’’, *IEEE Trans. SE*, Vol. 21, No. 8, Aug. 1995, pp 689-700.

“LQNS User Guide” (not available).

“SRVN Input File Format” by Dorina Petriu et al.

parasrvn(1), petrisrvn(1), srvn(1), srvn2eepic(1), srvntda(1), srvndiff(1), egrep(1), floating_point(3)