Forecasting Oracle Performance
- Better than a Crystal Ball

Yuri van Buren
Senior Oracle DBA Specialist
End-2-End Performance Management Engineer
Yuri van Buren

17 Years with Logica – which is now part of CGI

DBA = Database Administrator
- Operational DBA
- DBA Consultant
- Database Specialist
- Project DBA
- APPS (or EBS) DBA
- DBA Architect
- OCP10g &11g; OCE 11g Performance Tuning
- E2E Performance Management Engineer
- Principal Consultant & Practice Lead

Third time presenting at the DOAG
Better than a Crystal Ball

• “Can the system handle a 200% increase in workload?”

• “What will happen to performance if we add 2 more CPUs?”

• “What will happen to performance if we add 30% faster CPUs?”

• “Do 25 more GL users significantly impact the response time?”
Agenda Forecasting Oracle Performance

• Introduction
• Forecasting Models
• Forecasting Method
Introduction

Risk

Service Level Management

Model Types
- Mathematical
- Benchmark
- Simulation

Safety is a myth. Risk is reality. Fear is the mind killer. Seize the day. Take the road less travelled. Dare to be different. Nothing ventured, nothing gained. The adventure is within. Boldness has magic. Take the first step.
Forecasting Models

There are five commonly used forecasting models to forecast Oracle performance:

- Simple Math
- Essential Forecasting Mathematics
- Queuing Theory
- Ratio Modeling
- Linear Regression Modeling
Simple Math

- Can be used for single component forecasts.
- Input can be technical or application metrics
- No queuing effect is involved. Appropriate for memory, basic IO predictions and basic networking.
- The precision is usually low
- Handy for short-duration projects.

Example:
If an Oracle client process consumes 10MB of non-shared resident memory. And plans are there to add another 50 users (resulting in 50 additional clients), then the system will require 500MB of additional memory.
Essential Forecasting Mathematics

Transactions

- Arrival Rate is the amount of work in a certain time interval
- Transaction Processor is a “server” that can process transactions
- Queue – A linked list with a head and a tail – First In First Out queues are used.
- Transaction Flow - When a business transaction is submitted, it flows throughout the computing system consuming CPU, IO, memory, and network resources.

Response time
This is the most important term in forecasting and it is also one of the simplest. Response time is service time (S) plus queue time (W):
\[ R = S + W \]
## Essential Forecasting Mathematics

<table>
<thead>
<tr>
<th>Description</th>
<th>Arrival Rate (Work/Time)</th>
<th>Service Time (Time/Work)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fifty transactions arrive into the system each minute, each consuming 25 ms of CPU time.</td>
<td>50 trx/min</td>
<td>25 ms/trx</td>
</tr>
<tr>
<td>Twenty-five transactions arrive into the system each second. The five 3.4GHz CPUs can process a transaction in 28 ms.</td>
<td>25 trx/s</td>
<td>28 ms/trx</td>
</tr>
<tr>
<td>Five transactions enter the system every 5 ms. It takes a CPU 4 ms to service a transaction.</td>
<td>1 trx/ms = 5/5 trx/ms</td>
<td>4 ms/trx</td>
</tr>
<tr>
<td>Each server can service a transaction in 3 seconds. Every 10 seconds, 350 transactions enter the system.</td>
<td>35 trx/s = 350/10 trx/s</td>
<td>3 s/trx = 3/1 s/trx</td>
</tr>
<tr>
<td>Preliminary testing has shown the application consumes about 20 seconds of server CPU time to complete an order. We process around 500 orders every hour.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[
0.139\text{ orders/s} = \frac{500\text{ orders}}{1\text{ hr}} \times \frac{1\text{ hr}}{60\text{ min}} \times \frac{1\text{ min}}{60\text{ s}} \quad \text{20 s/order}
\]
Essential Forecasting Mathematics

Response Time Curve

![Response Time vs Arrival Rate](image)

- Response Time (ms/trx)
- Arrival Rate (trx/ms)
- Queue Time
- Service Time
- Response Time
### Essential Forecasting Mathematics

#### Key variables & Formulas

\[
U = \frac{S_t \lambda}{M}
\]

\[
Q = \lambda Q_t
\]

\[
R_{t-cpu} = \frac{S_t}{1 - U^M}
\]

\[
R_{t-io} = \frac{S_t}{1 - U}
\]

\[
R_t = S_t + Q_t
\]

<table>
<thead>
<tr>
<th>Variable</th>
<th>Definition</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\lambda)</td>
<td>The arrival rate is how much work enters the system within a period of time.</td>
<td>10 trx/ms</td>
</tr>
<tr>
<td>(S_t)</td>
<td>Service time is how long it takes a server to service a single transaction and is represented as time per transaction.</td>
<td>2 ms/trx</td>
</tr>
<tr>
<td>(U)</td>
<td>Utilization is the busyness of a server and is represented as a simple percentage.</td>
<td>83%</td>
</tr>
<tr>
<td>(Q)</td>
<td>Queue length is simply the length of the queue. For Oracle systems, we typically do not want transactions waiting in the queue.</td>
<td>0 trx</td>
</tr>
<tr>
<td>(Q_t)</td>
<td>Queue time is how long a transaction waits in the queue before it begins to be served. This is represented as time per transaction.</td>
<td>0.02 ms/trx</td>
</tr>
<tr>
<td>(R_t)</td>
<td>Response time is service time plus queue time and is represented as time per transaction.</td>
<td>2.02 ms/trx</td>
</tr>
<tr>
<td>(M)</td>
<td>Number of servers is how many servers are servicing transactions</td>
<td>24 servers</td>
</tr>
</tbody>
</table>
Essential Forecasting Mathematics Example

• DBA John works on a 12 CPU system. He measures an average CPU utilization of 35% during month-end closing. Oracle runs 6.11 user calls per second during that period. Can the system handle a 200% increase in workload?

\[
U = \frac{S_t \lambda}{M}; S_t = \frac{UM}{\lambda} = \frac{0.35 \times 12}{6.11 \text{uc/s}} = 0.69 \text{s/uc}
\]

\[
R_t = \frac{S_t}{1 - U^M} = \frac{0.69 \text{s/uc}}{1 - 0.35^{12}} = 0.69 \text{s/uc}
\]

\[
R_t = S_t + Q_t
\]

\[
Q_t = R_t - S_t = 0.69 \text{s/uc} - 0.69 \text{s/uc} = 0
\]

\[
Q = \lambda Q_t = 6.11 \text{uc/s} \times 0.00 \text{s/uc} = 0.00
\]

• We get a valuable result when we start to increase the arrival rate while keeping the service time and the number of CPUs constant.
At 110% workload increase queueing starts to occur.

The system cannot handle a 200% workload increase.

More CPU power is needed.
Essential Forecast Mathematics DEMO

Interactive Response Time Learning Graph.xlsx
Queuing Theory

- Little’s Law
- Kendall’s Notation
- Erlang C
Little’s Law - first proved in 1961

\[ N = \lambda T \]

- \( N \) is the average number (serviced and queued) in a stable system.

- \( \lambda \) is the average number of arrivals at some interval (that is, the arrival rate).

- \( T \) is the average time spent in the system (that is, the response time).
Little’s Law Example

A company is running 20 web servers. IT management expects all web servers to respond within 2 seconds (SLM).

The application server administrator has implemented monitoring scripts on the web logs which enables him/her to determine the number of HTTP request within any period of time. The last hour there were 20500 request. The ASA also created a script to gather the number of running HTTP processes. The average in the last hour was 4.

Little’s Law says $N = \lambda T$; We want to know the average response time $T$. 
$T = \frac{N}{\lambda} \Leftrightarrow \frac{4}{20500 \text{req/hour}} \Leftrightarrow \frac{4}{20500 / (60 \times 60 \text{sec})} = 0.70$ seconds/request

Note that this is true, regardless of the arrival and service time distributions.
Kendall’s Notation

A/B/m
With A = Arrival Pattern
With B = Service Time Distribution
With m = Number of servers in the system

A Markovian pattern means each transaction is independent of the other transactions. Both arrival and service times in Oracle database systems have a Markovian pattern. So Oracle queuing systems are denoted as a M/M/m queuing theory model.

For example, 1xM/M/4 is the notation for an Oracle database system with four CPUs, with one queue associated with four servers. In contrast, a 4xM/M/1 system would represent an IO subsystem containing four devices, each of the four queues is associated with a single server (that is a device).
Kendall Example

Modeling an airline check-in area with first class & economy class check-in desks

A $1x/M/M/2$ and a $1x/M/M/4$ queuing system
CPU & IO Subsystem Modeling

CPU and IO subsystems have fundamentally different queuing structures and therefore are modelled differently.
M/M/m model for 1 to 64 CPU
Compare a 1xM/M/4 model with a 4x/M/M/1 model. Which model will provide more throughput?

**Queuing Model Data Entry**

<table>
<thead>
<tr>
<th>Description</th>
<th>Case 1 Values</th>
<th>Case 2 Values</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Queues in system</td>
<td>1</td>
<td>4</td>
<td>#</td>
</tr>
<tr>
<td>Servers per queue</td>
<td>4</td>
<td>1</td>
<td>#</td>
</tr>
<tr>
<td>Service time</td>
<td>3,0000</td>
<td>3,0000</td>
<td>min/person</td>
</tr>
<tr>
<td>System Arrival rate</td>
<td>1,0000</td>
<td>1,0000</td>
<td>person/min</td>
</tr>
<tr>
<td>Response time tolerance</td>
<td>0,00</td>
<td></td>
<td>min</td>
</tr>
<tr>
<td>Server Arrival rate</td>
<td>1,00</td>
<td>0,25</td>
<td>person/min</td>
</tr>
<tr>
<td>Traffic intensity</td>
<td>3,00</td>
<td>0,75</td>
<td>#</td>
</tr>
<tr>
<td>Utilization</td>
<td>75,0%</td>
<td>75,0%</td>
<td>#</td>
</tr>
<tr>
<td>Queue time</td>
<td>1,5283</td>
<td>9,0000</td>
<td>min</td>
</tr>
<tr>
<td>Queue length</td>
<td>1,5283</td>
<td>2,2500</td>
<td>#</td>
</tr>
<tr>
<td>Response time expected</td>
<td>4,5283</td>
<td>12,0000</td>
<td>min</td>
</tr>
</tbody>
</table>
We can also solve the question: “What will happen to performance if we add 2 more CPUs?”

<table>
<thead>
<tr>
<th>Queuing Model Data Entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
</tr>
<tr>
<td>Queues in system</td>
</tr>
<tr>
<td>Servers per queue</td>
</tr>
<tr>
<td>Service time</td>
</tr>
<tr>
<td>System Arrival rate</td>
</tr>
<tr>
<td>Response time tolerance</td>
</tr>
<tr>
<td>Server Arrival rate</td>
</tr>
<tr>
<td>Traffic intensity</td>
</tr>
<tr>
<td>Utilization</td>
</tr>
<tr>
<td>Queue time</td>
</tr>
<tr>
<td>Queue length</td>
</tr>
<tr>
<td>Response time expected</td>
</tr>
</tbody>
</table>
We can also solve the question:
“What will happen to performance if we add 30% faster CPUs?”

### Queuing Model Data Entry

<table>
<thead>
<tr>
<th>Description</th>
<th>Case 1 Values</th>
<th>Case 2 Values</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Queues in system</td>
<td>1</td>
<td>1</td>
<td>#</td>
</tr>
<tr>
<td>Servers per queue</td>
<td>4</td>
<td>4</td>
<td>#</td>
</tr>
<tr>
<td>Service time</td>
<td>3,0000</td>
<td>2,1000</td>
<td>ms/trx</td>
</tr>
<tr>
<td>System Arrival rate</td>
<td>1,0000</td>
<td>1,0000</td>
<td>trx/ms</td>
</tr>
<tr>
<td>Response time tolerance</td>
<td>0,00</td>
<td></td>
<td>ms</td>
</tr>
<tr>
<td>Server Arrival rate</td>
<td>1,00</td>
<td>1,00</td>
<td>trx/ms</td>
</tr>
<tr>
<td>Traffic intensity</td>
<td>3,00</td>
<td>2,10</td>
<td>#</td>
</tr>
<tr>
<td>Utilization</td>
<td>75,0%</td>
<td>52,5%</td>
<td>#</td>
</tr>
<tr>
<td>Queue time</td>
<td>1,5283</td>
<td>0,2204</td>
<td>ms</td>
</tr>
<tr>
<td>Queue length</td>
<td>1,5283</td>
<td>0,2204</td>
<td>#</td>
</tr>
<tr>
<td>Response time expected</td>
<td>4,5283</td>
<td>2,3204</td>
<td>ms</td>
</tr>
</tbody>
</table>
Improving Essential Forecasting Mathematics with Erlang C

Erlang C Formula

Just specify
- number of servers per queue
- service time
- system arrival rate

Use an ErlangC.pl perl script

The single new queue time formula can be applied to both CPU and IO subsystems.
Erlang C Example

In a 24 CPU system a transaction arrives once every 0.20 seconds. The transaction is processed in 4 seconds. What are the CPU Utilization, response time, queue time and queue length?

```
[oracle@orc6 ~]$ perl erlangc.pl
Forecasting Oracle Performance by C. Shallahamer

More Precise Queueing Mathematics with ErlangC

System arrival rate = 5
Queues in system = 1
Queue arrival rate = 5
Servers per queue = 24
Queue traffic = 20
Erlang C = 0.298072299793829
Utilization = 0.8333333333333333
Service time = 4
Queue time = 0.298072299793829
Response time = 4.29807229979383
Queue length = 1.49036149896914
```
Ratio Modeling

\[ P = \frac{C_1}{R_1} + \ldots + \frac{C_n}{R_n} \]

\[ P = UM \]

P is the number of fully utilized CPU’s.
C is the number of workload category occurrences. (E.g. OLTP and Batch users).
R is the ratio linking a single workload category occurrence to a CPU.
U is the CPU Utilization.
M is the number of CPUs (servers).
Ratio Modeling Example

On a E-Business Suite environment the APPS-DBA knows that the OLTP-to-CPU ratio is 125 and the batch-to-CPU ratio is 2. The system has 18 CPU’s and has a utilization of around 44% while serving 12 batchjobs and 250 OLTP users.

Do 25 more GL users significantly impact the response time?

\[
P = \frac{C_{oltp}}{R_{oltp}} + \frac{C_{batch}}{R_{batch}} = \left(\frac{250+25}{125}\right) + \frac{12}{2} = \frac{275}{125} + 6 = 2.2 + 6 = 8.2
\]

\[
P = UM; \quad U = \frac{P}{M} = \frac{8.2}{18} = .46 = 46\%
\]
Linear Regression Modeling

Typically used to determine how much of some business activity can occur before a system “runs out of gas”.

E.g. Determine the maximum of orders the system can process within an hour.

We want to investigate if there is a strong relationship between CPU Utilization and the number of orders per hour.

Avoid non-linear areas.
CPU systems above 75% Utilization
IO systems above 65% Utilization
Determining a Linear Relationship

1) View the raw data
2) View the data graph
3) View the residual data
4) View the residual data graph
5) View the regression formula
6) View the correlation strength
7) If everything is OK, then forecast.

<table>
<thead>
<tr>
<th>Sample</th>
<th>CPU Util %</th>
<th>Order/Hr</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>15</td>
<td>71.00</td>
</tr>
<tr>
<td>2</td>
<td>27</td>
<td>159.00</td>
</tr>
<tr>
<td>3</td>
<td>28</td>
<td>167.00</td>
</tr>
<tr>
<td>4</td>
<td>-3</td>
<td>157.00</td>
</tr>
<tr>
<td>5</td>
<td>31</td>
<td>188.00</td>
</tr>
<tr>
<td>6</td>
<td>30</td>
<td>192.00</td>
</tr>
<tr>
<td>7</td>
<td>32</td>
<td>198.00</td>
</tr>
<tr>
<td>8</td>
<td>40</td>
<td>266.00</td>
</tr>
<tr>
<td>9</td>
<td>44</td>
<td>309.00</td>
</tr>
<tr>
<td>10</td>
<td>43</td>
<td>316.00</td>
</tr>
<tr>
<td>11</td>
<td>49</td>
<td>377.00</td>
</tr>
<tr>
<td>12</td>
<td>110</td>
<td>210.00</td>
</tr>
<tr>
<td>13</td>
<td>53</td>
<td>386.00</td>
</tr>
<tr>
<td>14</td>
<td>21</td>
<td>74.00</td>
</tr>
<tr>
<td>15</td>
<td>13</td>
<td>20.00</td>
</tr>
<tr>
<td>16</td>
<td>64</td>
<td>2431.00</td>
</tr>
<tr>
<td>17</td>
<td>16</td>
<td>19.00</td>
</tr>
</tbody>
</table>
## Determining a Linear Relationship

<table>
<thead>
<tr>
<th>Sample</th>
<th>CPU Util %</th>
<th>Orders/Hr</th>
<th>Predicted CPU Util %</th>
<th>Residual</th>
<th>Residual Squared</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>15</td>
<td>71.00</td>
<td>17.9</td>
<td>-3.2</td>
<td>10.3</td>
</tr>
<tr>
<td>2</td>
<td>27</td>
<td>159.00</td>
<td>27.2</td>
<td>-0.5</td>
<td>0.3</td>
</tr>
<tr>
<td>3</td>
<td>28</td>
<td>167.00</td>
<td>28.1</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>4</td>
<td>31</td>
<td>188.00</td>
<td>30.3</td>
<td>0.2</td>
<td>0.0</td>
</tr>
<tr>
<td>5</td>
<td>30</td>
<td>192.00</td>
<td>30.7</td>
<td>-0.2</td>
<td>0.1</td>
</tr>
<tr>
<td>6</td>
<td>32</td>
<td>198.00</td>
<td>31.4</td>
<td>0.2</td>
<td>0.0</td>
</tr>
<tr>
<td>7</td>
<td>40</td>
<td>266.00</td>
<td>38.6</td>
<td>1.2</td>
<td>1.4</td>
</tr>
<tr>
<td>8</td>
<td>44</td>
<td>309.00</td>
<td>43.1</td>
<td>1.0</td>
<td>1.1</td>
</tr>
<tr>
<td>9</td>
<td>43</td>
<td>316.00</td>
<td>43.9</td>
<td>-1.1</td>
<td>1.3</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>48</td>
<td>52</td>
<td>347.00</td>
<td>47.1</td>
<td>4.5</td>
<td>20.3</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Sum</td>
<td></td>
<td></td>
<td></td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>Avg.</td>
<td>27.6</td>
<td>162.6</td>
<td>27.6</td>
<td>0.0</td>
<td>2.4</td>
</tr>
</tbody>
</table>
Determining a Linear Relationship

\[ y = m_1 x_1 + b \]

<table>
<thead>
<tr>
<th>Correlation Coefficient ((r))</th>
<th>Practical Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0 to 0.2</td>
<td>Very weak, negligible</td>
</tr>
<tr>
<td>0.2 to 0.4</td>
<td>Weak, low</td>
</tr>
<tr>
<td>0.4 to 0.7</td>
<td>Moderate</td>
</tr>
<tr>
<td>0.7 to 0.9</td>
<td>Strong, high, marked</td>
</tr>
<tr>
<td>0.9 to 1.0</td>
<td>Very strong, very high</td>
</tr>
</tbody>
</table>

Most regression systems determine the best linear line by iteratively changing the line’s equation until the sum of all the residuals equals zero.

Only forecast with strong or very strong correlation coefficients.
What’s Next

Know you know something about mathematical models you still:
• Need a method
• Need to collect data
• Need to know about risk mitigation strategies
Forecasting Method

Step 1: Determine the study question
Step 2: Gather & Characterize workload data
Step 3: Select a forecast model
Step 4: Forecast & Validate
Step 5: Risk Mitigation or “What to tell management”.
Step 1: Determine the study question

Keep it simple

Get a (crystal) clear study question.

Have key players fully embrace the study question

Specify how the study question will be answered

Example:
If business revenue doubles, can the current systems meet service-level requirements?
Step 2: Gather & Characterize workload data

Gather just what is needed. The data collection should be repeatable. Gather OS data with tools like sar, vmstat, iostat. Gather Oracle data from v$sysstat.

Example script for Linux: gather_driver.sh

wl_app.dat

1282517401, user calls, 1755
1282517401, session logical reads, 414642
1282517401, physical reads, 48322
1282517401, db block changes, 605
1282517401, execute count, 31919

wl_cpu.dat

1282517401, 10.69, 0.27, 89.04

wl_io.dat

1282517401, cciss/c0d0, 0.67, 0.12, 0.05
1282517401, dm-0, 0.54, 0.04, 0.06
1282517401, dm-1, 0.00, 0.00, 0.00
Step 2: Gather & Characterize workload data

You can classify Oracle workload models into three groups:

• The simple workload model is application centric (think number of concurrent OLTP users and concurrent batch processes) and used for quick, low precision forecasts.

• The single-category workload model is used when a general workload centric metric statistic (think v$sysstat) is used to represent the entire workload.

• The multiple-category workload model provides increased precision capabilities and usability flexibility, but gathering the workload can be intrusive.

⇒ Each workload model is valuable and effective when used appropriately.
Step 3: Select a forecast model

Remember the 5 models?

- Simple Math
- Essential Forecasting Mathematics
- Queuing Theory
- Ratio Modeling
- Regression Analysis
Step 4: Forecast & Validate

When validating a model you can use a simple five-step process:

1. Look for numerical error
2. Analyze statistical error
3. Perform a histogram analysis
4. Perform a residual analysis
5. Make a go / no-go decision

<table>
<thead>
<tr>
<th>Sample</th>
<th>Actual CPU Util%</th>
<th>Predicted CPU Util%</th>
<th>Predicted Error</th>
<th>Abs (Predicted Error)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>47.9</td>
<td>35.6</td>
<td>-12.3</td>
<td>12.3</td>
</tr>
<tr>
<td>2</td>
<td>47.9</td>
<td>36.1</td>
<td>-11.8</td>
<td>11.8</td>
</tr>
<tr>
<td>3</td>
<td>43.2</td>
<td>39.4</td>
<td>-3.8</td>
<td>3.8</td>
</tr>
<tr>
<td>4</td>
<td>47.4</td>
<td>35.5</td>
<td>-11.9</td>
<td>11.9</td>
</tr>
<tr>
<td>5</td>
<td>47.4</td>
<td>36.0</td>
<td>-11.4</td>
<td>11.4</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>76</td>
<td>39.0</td>
<td>31.6</td>
<td>-7.4</td>
<td>7.4</td>
</tr>
<tr>
<td>77</td>
<td>45.1</td>
<td>31.4</td>
<td>-13.7</td>
<td>13.7</td>
</tr>
</tbody>
</table>

Average: 0.9
Standard deviation: 9.0
Skew: 0.21
Data points: 77

Histogram of Forecast Error

Forecast Error Residual Analysis
Step 5 Risk Mitigation or “What to tell management”

⇒ 3 Strategies

A) CPU Workload on a 4 CPU server

B) Add Capacity: Buy 2 extra CPUs

Tune - Reduce load by 40%

C) Balance Workload
Acknowledgements & Reference

Further reading → Craig Shallahamer. His website is www.orapub.com

Visit http://www.apress.com
Questions?
Thank you

Yuri van Buren

Contact information:

Name: Yuri van Buren
Company: Logica Nederland B.V.
Address: Prof. W.H. Keesomlaan 14
1183DJ Amstelveen
The Netherlands
Phone: +31 (0)88 5640000
Email: yuri.van.buren@logica.com
yuri.van.buren cgi.com
Internet: www.logica.com
www.cgi.com