Russian Academy of Sciences
St. Petersburg Institute for Informatics and Automation (SPIIRAS)

A Forth-Simulator of Real-Time Multi-Task Applications

Prof. Sergey Baranov,
Chief Research Associate
SNBaranov@gmail.com
Contents

- Introduction
- Source Data
- Output Data
- Data Structures
- The Simulator
- Four Dining Philosophers
- Conclusions
Real-Time Systems

Hardness
\[ H_i = \frac{T_i}{D_i} \]

Density
\[ Dens = U_{\text{max}} \]

Input messages
\[ T_1, T_2 \]
\[ T_n \]

Output messages
\[ D_1, D_2 \]
\[ D_n \]

Absolute weight \( W_i \) (op)

Relative weight \( C_i = \frac{W_i}{P} \) (seconds)

Utility
\[ u_i = \frac{C_i}{T_i} \]

Application:
- scheduling mode, protocol
- total utility: \( U = \sum_{i=1}^{n} u_i \)

Processor
Performance \( P \) (op/second)

\[ \tau_1, W_1, C_1, u_1 \]
\[ \tau_2, W_2, C_2, u_2 \]
\[ \tau_n, W_n, C_n, u_n \]

\( T_i \) – period (seconds)
\( D_i \) – deadline (seconds)
Utility Depends on CPU Performance

Application

- $T = 15\text{ms}$, $W = 4 \times 10^6 \text{op}$
- $T = 19\text{ms}$, $W = 5 \times 10^6 \text{op}$
- $T = 25\text{ms}$, $W = 7 \times 10^6 \text{op}$

Rate-monotonic scheduling

- $H_i = 1$ : $Dens = 0.81$

Early deadline first scheduling

- $H_i = 1$ : $Dens = 1$

- $P = 1.3 \times 10^6 \text{op/ms}$ $U = 0.62$
- $P = 1.0 \times 10^6 \text{op/ms}$ $U_{\text{max}} = 0.81$
- $P = 1.3 \times 10^6 \text{op/ms}$ $U = 0.62$
- $P = 0.81 \times 10^6 \text{op/ms}$ $U_{\text{max}} = 1.0$

EuroFORTH 2015, October 1-3, 2015, Bath, England
Protocols for Access to Resources

The simplest protocol (no priority inheritance)

- **Condition to entry a critical interval:** the resource is unlocked
- **Action at entering a critical interval:** lock the resource
- **Action at exiting a critical interval:** unlock the resource

Other protocols (basic/transitional priority inheritance)

- **When entering a critical interval:** check additional conditions, perform additional actions
- **When exiting a critical interval:** perform additional actions
Specifying Task Structure

<task name="t_1"
    phase="5" period="15">
    <segment length="1"
        interface="m_1"
        op_type="lock"/>
    <segment length="1"
        interface="m_1"
        op_type="unlock"/>
    <segment length="1"
        op_type="end"/>
</task>
Creating a Resource

: CreateResource ( n--resource-addr) \ Create a new resource with the ID=n
    ResourcePool CELL+ ( n, Pool-addr)
    BEGIN
    DUP @ 0=
    IF \ Add a new resource to the pool
        #Resources 1+!
        #Resources @ Max#Resources > ABORT" ResourcePool overflow!"
    ( n, pool-addr,)
    2DUP ! \ -1 Store the resource number
    CELL+ ( n, new-res-addr)
    DUP 0! \ 0 Resource priority
    DUP CELL+ ( n, new-res-addr, res-status-addr)
    DUP 0! \ 1 Resource status
    CELL+ ( n, new-res-addr, res-queue-addr)
    NULL OVER ! \ 2 Resource queue of jobs waiting for this resource
    CELL+ ( n, new-res-addr, res-#elems-addr)

....
Finding the Scaling Factor

ScaleInf 0!
Sf @ DUP * Utility /Round ScaleSup !
BEGIN
  ScaleSup @ ScaleInf @ - 1 >
  WHILE
  ScaleInf @ ScaleSup @ + 2/ ScaleMed !
  #Violations 0!
  Simul
\ checking that all deadlines were met
ScaleMed @ #Violations @ 0=
  IF
    ScaleInf
  ELSE
    ScaleSup
  THEN
!   REPEAT

Catching a Lion in a Desert
Working through Ordered Lists

: List ( list-element-size, max-list-length -- ) \ Define a list
  CREATE
    NULL , \ The "Next" field
    0 , \ The current number of elements in the list
    , \ The maximal number of list elements
    DOES> ( -- list addr) ;

20 List TaskList \ Ordered by their static priorities
120 List EventList \ Ordered by their time to occur
10000 List JobList \ Ordered by their dynamic (current) priorities

: >List ( new-elem-addr, list-addr -- ) \ Place a new element into the ordered list

: List@ ( list-addr-- elem-addr) \ Get the first (heading) element of the list

: List> ( list-addr-- elem-addr) \ Delete the first element from the list

: List>> ( ordering-value, list-addr--) \ Find and delete a list element
The Simulator

Configure and initialize

While-condition OK?

Yes

Advance time
Process events
Process jobs

No

Advance time
Process events
Process jobs

EventList

Time-sake events

\[ time = t_1 \]
\[ t_1 < t_2 < t_3 < \ldots \]

Time-sake events

Time-sake events

JobList

Job1 \rightarrow Job2 \rightarrow Job3 \rightarrow \ldots

While-condition: (Time < TimeLimit)
\[ \land \#Jobs < JobLimit \] \land
\[ \#Violations < ViolationsLimit \]

Events:
- Activate a task (create a new job)
- Lock a resource
- Unlock a resource
- Terminate a job

Job: Consume processor time by task segment and add a new event to the EventList

Prio(Job_1) \geq Prio(Job_2) \geq Prio(Job_3) \geq \ldots
Logs of Two Simulation Sessions

| TimeLimit=25 JobLimit=0 ViolationLimit=1 SchedulingMode=RM InheritanceMode=NI Configuration file name: c:\MPE\App_4t2r.txt |
| TimeLimit=25 JobLimit=0 ViolationLimit=1 SchedulingMode=RM InheritanceMode=BI Configuration file name: c:\MPE\App_4t2r.txt |

Time=0 Proc=0 for 0 A 4.1
Time=2 Proc=4.1 for 2 L 4.1 of 2
Time=3 Proc=4.1 for 1 A 3.2
Time=4 Proc=3.2 for 1 L 3.2 of 1
Time=5 Proc=3.2 for 1 A 1.3 A 2.4
Time=6 Proc=1.3 for 1 W 1.3 of 1
Time=15 Proc=2.4 for 9 E 2.4
Time=16 Proc=3.2 for 1 W 3.2 of 2
Time=19 Proc=4.1 for 3 U 4.1 of 2 L 3.2 of 2
Time=20 Proc=3.2 for 1 U 3.2 of 2
Time=21 Proc=3.2 for 1 U 3.2 of 1 L 1.3 of 1
Time=22 Proc=1.3 for 1 U 1.3 of 1
Time=23 Proc=1.3 for 1 E 1.3
Time=24 Proc=3.2 for 1 E 3.2
Time=25 Proc=4.1 for 1 E 4.1
Time=25 Hardness=1,0000 1/Hardness=1,0000 Density=0,6056 ScalingFactor=1,0000

ERROR: Deadline violation in Task 1  ok

Time=7 Proc=3.2 for 1 W 3.2 of 2
Time=10 Proc=4.1 for 3 U 4.1 of 2 L 3.2 of 2
Time=11 Proc=3.2 for 1 U 3.2 of 2
Time=12 Proc=3.2 for 1 U 3.2 of 1 L 1.3 of 1
Time=13 Proc=1.3 for 1 U 1.3 of 1
Time=14 Proc=1.3 for 1 E 1.3
Time=23 Proc=2.4 for 9 E 2.4
Time=24 Proc=3.2 for 1 E 3.2
Time=25 Proc=4.1 for 1 E 4.1
Time=25 Hardness=1,0000 1/Hardness=1,0000 Density=0,6056 ScalingFactor=1,0000 ok

EuroFORTH 2015, October 1-3, 2015, Bath, England
Simulation of 4 Tasks with 2 Resources

a) NI – deadline violation in $\tau_1$

- $\tau_i$ is owns the processor
- $\tau_i$ is waiting for the processor

b) BI – no violations

- $\tau_i$ is waiting for access to $g_j$
- $\tau_i$ owns the resource $g_j$
RM vs. EDF in the Same Application

<table>
<thead>
<tr>
<th>1/H</th>
<th>RM</th>
<th>EDF</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>0.0336</td>
<td>0.0336</td>
</tr>
<tr>
<td>0.05</td>
<td>0.1009</td>
<td>0.1009</td>
</tr>
<tr>
<td>0.10</td>
<td>0.1682</td>
<td>0.1682</td>
</tr>
<tr>
<td>0.15</td>
<td>0.3028</td>
<td>0.3028</td>
</tr>
<tr>
<td>0.20</td>
<td>0.3028</td>
<td>0.3028</td>
</tr>
<tr>
<td>0.25</td>
<td>0.3028</td>
<td>0.3028</td>
</tr>
<tr>
<td>0.30</td>
<td>0.3028</td>
<td>0.3028</td>
</tr>
<tr>
<td>0.35</td>
<td>0.3028</td>
<td>0.3784</td>
</tr>
<tr>
<td>0.40</td>
<td>0.3784</td>
<td>0.4541</td>
</tr>
<tr>
<td>0.45</td>
<td>0.4541</td>
<td>0.5046</td>
</tr>
<tr>
<td>0.50</td>
<td>0.5046</td>
<td>0.5719</td>
</tr>
<tr>
<td>0.55</td>
<td>0.6392</td>
<td>0.7065</td>
</tr>
<tr>
<td>0.60</td>
<td>0.7065</td>
<td>0.7568</td>
</tr>
<tr>
<td>0.65</td>
<td>0.7568</td>
<td>0.8326</td>
</tr>
<tr>
<td>0.70</td>
<td>0.8326</td>
<td>0.9083</td>
</tr>
<tr>
<td>0.75</td>
<td>0.9083</td>
<td>0.9083</td>
</tr>
</tbody>
</table>
Four (Five) Dining Philosophers

...Five philosophers, numbered from 0 through 4 are living in a house where the table laid for them, each philosopher having his own place at the table:

![Diagram of five philosophers with forks]

Their only problem - besides those of philosophy - is that the dish served is a very difficult kind of spaghetti, that has to be eaten with two forks. There are two forks next to each plate, so that presents no difficulty: as a consequence, however, no two neighbours may be eating simultaneously. ...

Configuration Data for 4 Philosophers

<task name="t_1" phase="10" period="1000">
  <segment length=2 interface="r_1" op_type="lock"/>
  <segment length=4 interface="r_2" op_type="lock"/>
  <segment length=20 interface="r_1" op_type="unlock"/>
  <segment length=68 interface="r_2" op_type="unlock"/>
  <segment length=2 op_type="end"/>
</task>

<task name="t_2" phase="7" period="1000">
  <segment length=2 interface="r_2" op_type="lock"/>
  <segment length=4 interface="r_3" op_type="lock"/>
  <segment length=20 interface="r_2" op_type="unlock"/>
  <segment length=73 interface="r_3" op_type="unlock"/>
  <segment length=2 op_type="end"/>
</task>

<task name="t_3" phase="4" period="1000">
  <segment length=2 interface="r_3" op_type="lock"/>
  <segment length=4 interface="r_4" op_type="lock"/>
  <segment length=20 interface="r_3" op_type="unlock"/>
  <segment length=79 interface="r_4" op_type="unlock"/>
  <segment length=2 op_type="end"/>
</task>

<task name="t_4" phase="1" period="1000">
  <segment length=2 interface="r_4" op_type="lock"/>
  <segment length=4 interface="r_1" op_type="lock"/>
  <segment length=20 interface="r_4" op_type="unlock"/>
  <segment length=85 interface="r_1" op_type="unlock"/>
  <segment length=2 op_type="end"/>
</task>
## Log for the 4 Philosophers Puzzle

<table>
<thead>
<tr>
<th>System Log</th>
<th>Interpretation/Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>TimeLimit=1000000 JobLimit=0</td>
<td>Rate Monotonic with Priority Inheritance</td>
</tr>
<tr>
<td>ViolationLimit=0 SchedulingMode=RM InheritanceMode=PI</td>
<td></td>
</tr>
<tr>
<td>Time=1 Proc=0 for 1 A 4.1</td>
<td>Task 4 (job 4.1) is activated at time=1</td>
</tr>
<tr>
<td>Time=3 Proc=4.1 for 2 L 4.1 of 4</td>
<td>Task 4 (job 4.1) locks resource 4 at time=3</td>
</tr>
<tr>
<td>Time=4 Proc=4.1 for 1 A 3.2</td>
<td>Task 3 (job 3.2) is activated at time=4</td>
</tr>
<tr>
<td>Time=6 Proc=3.2 for 2 L 3.2 of 3</td>
<td>Task 3 (job 3.2) locks resource 3 at time=6</td>
</tr>
<tr>
<td>Time=7 Proc=3.2 for 1 A 2.3</td>
<td>Task 2 (job 2.3) is activated at time=7</td>
</tr>
<tr>
<td>Time=9 Proc=2.3 for 2 L 2.3 of 2</td>
<td>Task 2 (job 2.3) locks resource 2 at time=9</td>
</tr>
<tr>
<td>Time=10 Proc=2.3 for 1 A 1.4</td>
<td>Task 1 (job 1.4) is activated at time=10</td>
</tr>
<tr>
<td>Time=12 Proc=1.4 for 2 L 1.4 of 1</td>
<td>Task 1 (job 1.4) locks resource 1 at time=12</td>
</tr>
<tr>
<td>Time=16 Proc=1.4 for 4 W 1.4 of 2</td>
<td>Task 1 (job 1.4) waits for resource 2 at time=16</td>
</tr>
<tr>
<td>Time=19 Proc=2.3 for 3 W 2.3 of 3</td>
<td>Task 2 (job 2.3) waits for resource 3 at time=19</td>
</tr>
<tr>
<td>Time=22 Proc=3.2 for 3 W 3.2 of 4</td>
<td>Task 3 (job 3.2) waits for resource 4 at time=22</td>
</tr>
<tr>
<td>Time=25 Proc=4.1 for 3</td>
<td>Clinch detected for task 4 (job 4.1) when it tried</td>
</tr>
<tr>
<td><strong>Mutual clinch for job 4.1 on resource 1</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>to lock resource 1 at time=25</td>
</tr>
</tbody>
</table>

| Resource_1 Prio=0 Status=Job 1.4 JobsWaiting=NULL | |
| Resource_2 Prio=0 Status=Job 2.3 JobsWaiting=Job 1.4 | |
| Resource_3 Prio=0 Status=Job 3.2 JobsWaiting=Job 2.3 | |
| Resource_4 Prio=0 Status=Job 4.1 JobsWaiting=Job 3.2 | |
Conclusions

- VFX Forth for Windows, version 4.70
- Only 985 lines of code in Forth
- Only fixed-point arithmetic used
- Special memory allocation to avoid overflow

Future plans:
- Improve user’s interface
- More access protocols
- More scheduling modes
- Multi-core processors simulation
Acknowledgements

SPIIRAS and Prof. Victor Nikiforov for the work opportunity and fruitful cooperation
http://www.spiiras.nw.ru

ITMO University and Prof. Vladimir Parfenov for the Grant 074-U01 from the Government of the Russian Federation http://www.ifmo.ru


Thank you for your attention!