A bi-criteria scheduling heuristic for distributed embedded systems under reliability and real-time constraints

Ismail ASSAYAD, Alain GIRAULT, Hamoudi Kalla

Verimag/ST Micro, Inria Rhône-Alpes
The global methodology picture

real-time objective $R_{t, obj}$
reliability objective $R_{el, obj}$

architecture specification $Arc$
algorithm specification $Alg$
distribution constraints $Dis$
execution / transmission times $Exe$
reliability characteristics $Rel$

**Reliable Bi-criteria Scheduling Algorithm**

reliable distributed static schedule

SynDEx code generator

reliable distributed embedded code
Algorithm model

A data-flow graph $\mathcal{Alg}$, where:

- each vertex is an operation
- each edge is a data-dependency
- a vertex without predecessor is an input operation
- a vertex without successor is an output operation

The graph is executed repeatedly for each input event from the sensors, in order to compute the output events for the actuators.
Architecture model

A graph \( \mathcal{A}_r \) of procs linked by point-to-point communication links

Example:

(a)

(b)

\( o_1 \rightarrow o_2 \rightarrow o_4 \)

\( o_1 \rightarrow o_3 \)

\( p_1 \rightarrow p_2 \)

\( l_{12} \)

\( l_{23} \)

\( l_{24} \)

\( l_{34} \)

\( p_3 \)

\( p_4 \)
### Worst-case execution times

<table>
<thead>
<tr>
<th>processor</th>
<th>operation</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>time</td>
<td>$o_1$</td>
<td>$o_2$</td>
<td>$o_3$</td>
</tr>
<tr>
<td>$p_1$</td>
<td>20</td>
<td>3</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>$p_2$</td>
<td>3</td>
<td>4</td>
<td>$\infty$</td>
<td>4</td>
</tr>
<tr>
<td>$p_3$</td>
<td>5</td>
<td>2</td>
<td>2</td>
<td>$\infty$</td>
</tr>
<tr>
<td>$p_4$</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>data-dependency</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>time</td>
<td>$o_1 \triangleright o_2$</td>
<td>$o_1 \triangleright o_3$</td>
<td>$o_2 \triangleright o_4$</td>
<td>$o_3 \triangleright o_4$</td>
</tr>
<tr>
<td>$l_{12}$</td>
<td>2</td>
<td>5</td>
<td>3</td>
<td>1.5</td>
</tr>
<tr>
<td>$l_{23}$</td>
<td>3</td>
<td>6</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>$l_{24}$</td>
<td>3</td>
<td>6</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>$l_{34}$</td>
<td>2</td>
<td>5</td>
<td>3</td>
<td>1.5</td>
</tr>
</tbody>
</table>
Worst-case execution times

<table>
<thead>
<tr>
<th>processor</th>
<th>operation</th>
<th>data-dependency</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$o_1$</td>
<td>$o_1 \triangleright o_2$</td>
</tr>
<tr>
<td>time</td>
<td>$o_2$</td>
<td>$o_2 \triangleright o_3$</td>
</tr>
<tr>
<td>$p_1$</td>
<td>20</td>
<td>3</td>
</tr>
<tr>
<td>$p_2$</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>$p_3$</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>$p_4$</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

$\infty$ indicates that the operation cannot be executed on the processor
Reliability model

We consider only component failures (procs and links)

We assume that component failures are independant

We assume that all the algorithms are correct

Failures have an exponential distribution [Shatz, Wang, & Goto 92]

<table>
<thead>
<tr>
<th>processor</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_1$</td>
</tr>
<tr>
<td>failure rate $\lambda$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>communication link</th>
</tr>
</thead>
<tbody>
<tr>
<td>$l_{12}$</td>
</tr>
<tr>
<td>failure rate $\lambda$</td>
</tr>
</tbody>
</table>
Reliability Block Diagram (RBD)

To each partial schedule of $Alg$ onto $Arc$, with or without replication, we associate a RBD [Abd-allah 97] [Figiel & Sule 90]
Reliability Block Diagram (RBD)

To each partial schedule of $A_{lg}$ onto $A_{rc}$, with or without replication, we associate a RBD [Abd-allah 97] [Figiel & Sule 90]
To each partial schedule of $\text{Alg}$ onto $\text{Arc}$, with or without replication, we associate a RBD [Abd-allah 97] [Figiel & Sule 90]
Reliability Block Diagram (RBD)

To each partial schedule of $A_{lg}$ onto $A_{rc}$, with or without replication, we associate a RBD [Abd-allah 97] [Figiel & Sule 90]
To each partial schedule of \( \text{Alg} \) onto \( \text{Arc} \), with or without replication, we associate a RBD [Abd-allah 97] [Figiel & Sule 90]
Reliability Block Diagram (RBD)

Then we compute the **minimal cut sets** of the RDB: the minimum combination of failures that might cause the system to fail.

Finally we compute the **overall system reliability**: 

\[
\mathcal{R}_{sced}^* \leq \prod_{i=1}^{k} \left( 1 - \prod_{(o,c) \in \mathcal{Mcs}_i} (1 - \mathcal{R}_{sced}(o,c)) \right)
\]
The two criteria for scheduling

- $R_{tobj}$ is the schedule length objective
- $R_{elobj}$ is the reliability objective
The two criteria for scheduling

- $R_{t obj}$ is the schedule length objective
- $R_{el obj}$ is the reliability objective

These criteria are antagonistic, because usually:

- introducing replication improves the reliability
- but also increases the schedule length
The two criteria for scheduling

- $R_{t_{obj}}$ is the schedule length objective
- $R_{el_{obj}}$ is the reliability objective

These criteria are antagonistic, because usually:

- introducing replication improves the reliability
- but also increases the schedule length

Here, we also use replication to increase the locality of computations

- in some situations, it also reduces the schedule length!
Reliable Bi-criteria Scheduling Heuristic

Greedy list scheduling [Yang & Gerasoulis 93]

Maintains two lists of operation of \( Alg \):

- \( O_{sched}^{(n)} \) = list of scheduled operations; initially empty
- \( O_{cand}^{(n)} \) = list of candidate operations; contains initially the operations without predecessors

At the step \( (n) \), we compute the compromise function for each pair \( \langle \text{candidate operation}, \text{processor set} \rangle \), and we choose the best compromise

If the processor set has more than one element, then the chosen operation is replicated
RBSA uses a compromise function based on the reliability loss and the schedule length gain (both are normalised)

\[
\mathcal{L}^{(n)}(o_i, \{p_1, \ldots, p_j\}) = \frac{\text{Rel}^{(n)}_{\text{sched}}(o_i, \{p_1, \ldots, p_j\}) - \text{Rel}^{(n-1)}_{\text{sched}}}{\text{Rel}_{\text{obj}} - \text{Rel}^{(n-1)}_{\text{sched}}}
\]

\[
\mathcal{S}^{(n)}(o_i, \{p_1, \ldots, p_j\}) = \frac{\text{Rt}^{(n)}_{\text{sched}}(o_i, \{p_1, \ldots, p_j\}) - \text{Rt}^{(n-1)}_{\text{sched}}}{\text{Rt}_{\text{obj}} - \text{Rt}^{(n-1)}_{\text{sched}}}
\]
Compromise function

**RBSA** uses a compromise function based on the **reliability loss** and the **schedule length gain** (both are normalised)
Selection of the best candidate operation

To each candidate operation corresponds a point in the $\mathcal{L}$-$\mathcal{G}$ plane.

We project all the points onto a line, and select the point which has the best projection ($\theta$ is a parameter of the heuristics).

$$\mathcal{L} = \tan(\theta) \mathcal{G}$$

- Relative gain $\mathcal{G}^{(n)}(o_1, \mathcal{P}_j)$
- Relative loss $\mathcal{L}^{(n)}(o_1, \mathcal{P}_j)$

- $(o_1, \{p_1\})$
- $(o_1, \{p_2\})$
- $(o_1, \{p_3\})$
- $(o_1, \{p_1, p_3\})$

Diagram showing points in the $\mathcal{L}$-$\mathcal{G}$ plane and the projection line $\mathcal{L} = \tan(\theta) \mathcal{G}$. The points are projected onto a line, and the best projection is selected.
What if RBSA fails?

We modify $\theta$ and iterate

The new $\theta$ depends on which objective was not reached
The final methodology picture

Reliable Bi-criteria Scheduling Algorithm

real-time objective $R_{tobj}$
reliability objective $R_{elobj}$

architecture specification $Arc$
algorithm specification $Alg$
distribution constraints $Dis$
execution / transmission times $Exe$
reliability characteristics $Rel$

change $\theta$
(\(\theta\) is a parameter of the algorithm)
re-execute the algorithm

check objectives
not satisfied
satisfied

reliable distributed static schedule
fails to satisfy objectives

SynDEx code generator

reliable distributed embedded code
The architecture is assumed to be fully connected

\( \text{RBSA} (0^\circ) \) against \( \text{FTBAR}(0) \) [Girault, Kalla, Sighireanu & Sorel 03] and \( \text{HBP}(0) \) [Hashimoto, Tsuchiya & Kikuno 02]
RBSA simulations (II)

RBSA(45°) against FTBAR(1) and HBP(1)

Number of operations

Average Normalized Schedule Length

Replication rate

FTBAR(1) = HBP(1) = 2 copies
RBSA(45°) = 1.22 copies

CCR=0.1 CCR=1 CCR=10

FTBAR(1)
HBP(1)
RBSA(45°)

CCR=0.1 CCR=10 CCR=1

FTBAR(1)
HBP(1)
RBSA(45°)
**RBSA simulations (II)**

**RBSA(45°) against FTBAR(1) and HBP(1)**

![Graph showing the comparison between RBSA(45°), FTBAR(1), and HBP(1) with different CCR values across various number of operations.](image-url)
Analysis

If only $R_{t_{obj}}$ is considered, $RBSA$ is almost as good as FTBAR

If both objectives are considered, $RBSA$ is significantly better, and all the more when $CCR \gg 1$

thanks to our use of replication to improve both the reliability and the locality of computations

More simulations are required to compare $RBSA$ with FTBAR(2), on bigger architectures (6 processors)