Fault Tolerant Scheduling of Precedence Task Graphs on Heterogeneous Platforms

Anne Benoit, Mourad Hakem and Yves Robert

LIP Laboratory - ENS Lyon - France

APDCM 2008 - Miami, Florida, USA

April 14, 2008
Motivation

Context
- General context of DAG scheduling (precedence task graphs)
- Goal: minimize the latency (makespan)
- Already a difficult challenge

Failures?
- Software is assumed to be reliable
- Only hardware failures of processors
- Faults are assumed to be fail-silent (fail-stop)

Constraints and objectives
- Precedence constraints between tasks: don’t violate them
- Real time constraint: minimize the latency
- Fault tolerance objective: tolerate at most $\varepsilon$ proc. failures
Motivation

Context
- General context of DAG scheduling (precedence task graphs)
- Goal: minimize the latency (makespan)
- Already a difficult challenge

Failures?
- Software is assumed to be reliable
- Only hardware failures of processors
- Faults are assumed to be fail-silent (fail-stop)

Constraints and objectives
- Precedence constraints between tasks: don’t violate them
- Real time constraint: minimize the latency
- Fault tolerance objective: tolerate at most $\varepsilon$ proc. failures
Motivation

Context
- General context of DAG scheduling (precedence task graphs)
- Goal: minimize the latency (makespan)
- Already a difficult challenge

Failures?
- Software is assumed to be reliable
- Only hardware failures of processors
- Faults are assumed to be fail-silent (fail-stop)

Constraints and objectives
- **Precedence constraints between tasks**: don’t violate them
- **Real time constraint**: minimize the latency
- **Fault tolerance objective**: tolerate at most $\varepsilon$ proc. failures
Problem and solutions

Bi-criteria problem

Find a distributed schedule on heterogeneous platforms which minimizes latency $L$ while tolerating $\varepsilon$ processor failures.

- Primary/Backup (passive replication)
  - all techniques in the literature assume only one proc. failure
  - requires fault detection mechanism

- Active replication
  - tolerates multiple processor failure
  - no fault detection mechanism
  - ... but communication and computation overhead
  - FTBAR algorithm, our approach (off-line scheduling)
Bi-criteria problem

Find a distributed schedule on heterogeneous platforms which minimizes latency $L$ while tolerating $\varepsilon$ processor failures.

- Primary/Backup (passive replication)
  - all techniques in the literature assume only one proc. failure
  - requires fault detection mechanism

- Active replication
  - tolerates multiple processor failure
  - no fault detection mechanism
  - ... but communication and computation overhead
  - FTBAR algorithm, our approach (off-line scheduling)
Problem and solutions

Bi-criteria problem

Find a distributed schedule on heterogeneous platforms which minimizes latency \( L \) while tolerating \( \varepsilon \) processor failures.

- Primary/Backup (passive replication)
  - all techniques in the literature assume only one proc. failure
  - requires fault detection mechanism

- Active replication
  - tolerates multiple processor failure
  - no fault detection mechanism
  - ... but communication and computation overhead
  - FTBAR algorithm, our approach (off-line scheduling)
Example: passive/active replication schemes, $\varepsilon = 1$
Basic definitions and notations

- Parallel application: DAG → $G = (V, E)$
- $\Gamma^-(t), \Gamma^+(t)$: set of predecessors and successors of $t$
- **Free task**: all predecessors are already scheduled

- **Top level** $tl$ of a free task: computed from predecessors top levels (including communication)
- **Bottom level** $bl$ of a task: computed from
  - average computation time of the task
  - average communication cost to successors
  - bottom level of successors

- **Task criticalness**: task $t$ with the highest priority: $tl(t) + bl(t)$
Basic definitions and notations

- Parallel application: \( \text{DAG} \rightarrow G = (V, E) \)
- \( \Gamma^{-}(t), \Gamma^{+}(t) \): set of predecessors and successors of \( t \)
- **Free task**: all predecessors are already scheduled
- **Top level** \( t_l \) of a free task: computed from predecessors top levels (including communication)
- **Bottom level** \( b_l \) of a task: computed from
  - average computation time of the task
  - average communication cost to successors
  - bottom level of successors
- **Task criticalness**: task \( t \) with the highest priority:
  \( t_l(t) + b_l(t) \)
Basic definitions and notations

- Parallel application: DAG $\rightarrow G = (V, E)$
- $\Gamma^-(t), \Gamma^+(t)$: set of predecessors and successors of $t$
- Free task: all predecessors are already scheduled
- Top level $t_\ell$ of a free task: computed from predecessors top levels (including communication)
- Bottom level $b_\ell$ of a task: computed from
  - average computation time of the task
  - average communication cost to successors
  - bottom level of successors
- Task criticalness: task $t$ with the highest priority: $t_\ell(t) + b_\ell(t)$
Examples of top and bottom levels

Example: Homogeneous platforms

- $t\ell(t_4) = 9$
- $b\ell(t_4) = 10$
- Priority$(t_4) = 19$
A brief description of FTSA algorithm

Principle

- Software solution
- Uses the active software replication scheme to mask failures
- Can tolerate a fixed number $\varepsilon$ of arbitrary processor failures

The algorithm:

- Select a critical free task $t$ (keep ordered list)
- Simulate its mapping on all processors using equation:
  \[
  \forall 1 \leq j \leq m, \quad F(t, P_j) = \varepsilon(t, P_j) + \max_{t^* \in \Gamma^- (t)} \left\{ \max_{k=1}^{\varepsilon+1} \min \left\{ F(t^*_k, P(t^*_k)), W(t^*_k, t) \right\}, r(P_j) \right\}
  \]
- Keep $\varepsilon + 1$ processors allowing minimum finish time of $t$;
- Schedule $t^*_k, 1 \leq k \leq \varepsilon + 1$ on selected $\varepsilon + 1$ distinct proc.
A brief description of FTSA algorithm

Principle

- **Software solution**
- Uses the active software *replication scheme* to *mask failures*
- Can tolerate a fixed number \( \varepsilon \) of arbitrary processor failures

The algorithm:

- Select a critical **free task** \( t \) (keep ordered list)
- Simulate its mapping on all processors using equation:

\[
\forall 1 \leq j \leq m, \quad F(t, P_j) =
\]

\[
\varepsilon(t, P_j) + \max \left( \max_{t^* \in \Gamma(t)} \left\{ \min_{k=1}^{\varepsilon+1} \{ F(t^*_k, P(t^*_k)) + W(t^*_k, t) \} \right\}, r(P_j) \right)
\]

- Keep \( \varepsilon + 1 \) processors allowing *minimum finish time* of \( t \);
- Schedule \( t^k, 1 \leq k \leq \varepsilon + 1 \) on selected \( \varepsilon + 1 \) distinct proc.
A brief description of FTSA algorithm

Principle

- Software solution
- Uses the active software replication scheme to mask failures
- Can tolerate a fixed number $\varepsilon$ of arbitrary processor failures

The algorithm:

- Select a critical free task $t$ (keep ordered list)
- Simulate its mapping on all processors using equation:
  \[
  \forall 1 \leq j \leq m, \quad F(t, P_j) = E(t, P_j) + \max_{t^* \in \Gamma^-(t)} \left\{ \min_{k=1}^{\varepsilon+1} \{ F(t^*_{t_k}, P(t^*_{t_k})) + W(t^*_{t_k}, t) \} \right\}, r(P_j)
  \]
- Keep $\varepsilon + 1$ processors allowing minimum finish time of $t$;
- Schedule $t^k, 1 \leq k \leq \varepsilon + 1$ on selected $\varepsilon + 1$ distinct proc.
A brief description of FTSA algorithm

Principle

- Software solution
- Uses the active software replication scheme to mask failures
- Can tolerate a fixed number \( \varepsilon \) of arbitrary processor failures

The algorithm:

- Select a critical free task \( t \) (keep ordered list)
- Simulate its mapping on all processors using equation:
  \[
  \forall 1 \leq j \leq m, \quad F(t, P_j) =
  \varepsilon(t, P_j) + \max \left\{ \begin{array}{cl}
  \max_{t^* \in \Gamma(t)} & \min_{k=1}^{\varepsilon+1} \{ F(t^*_k, P(t^*_k)) + W(t^*_k, t) \} , \ r(P_j) \end{array} \right\}
  \]
- Keep \( \varepsilon + 1 \) processors allowing minimum finish time of \( t \);
- Schedule \( t^k \), \( 1 \leq k \leq \varepsilon + 1 \) on selected \( \varepsilon + 1 \) distinct proc.
A brief description of FTSA algorithm

**Principle**
- Software solution
- Uses the active software replication scheme to *mask failures*
- Can tolerate a fixed number $\varepsilon$ of arbitrary processor failures

**The algorithm:**
- Select a critical free task $t$ (keep ordered list)
- Simulate its mapping on all processors using equation:
  \[
  \forall 1 \leq j \leq m, \quad F(t, P_j) = E(t, P_j) + \max_{t^* \in \Gamma^- (t)} \left\{ \min_{k=1}^{\varepsilon+1} \{ F(t^*_k, P(t^*_k)) + W(t^*_k, t) \} \right\} + r(P_j)
  \]
- Keep $\varepsilon + 1$ processors allowing *minimum finish time* of $t$;
- Schedule $t^*_k, 1 \leq k \leq \varepsilon + 1$ on selected $\varepsilon + 1$ distinct proc.
A brief description of FTSA algorithm

Principle

- Software solution
- Uses the active software replication scheme to mask failures
- Can tolerate a fixed number $\varepsilon$ of arbitrary processor failures

The algorithm:

- Select a critical free task $t$ (keep ordered list)
- Simulate its mapping on all processors using equation:
  $$\forall 1 \leq j \leq m, \quad F(t, P_j) =$$
  $$E(t, P_j) + \max \left( \max_{t_\ast \in \Gamma^-(t)} \left\{ \min_{k=1}^{\varepsilon+1} \{ F(t_\ast, P(t_\ast^k)) + W(t_\ast^k, t) \} \right\}, r(P_j) \right)$$
- Keep $\varepsilon + 1$ processors allowing minimum finish time of $t$;
- Schedule $t^k, 1 \leq k \leq \varepsilon + 1$ on selected $\varepsilon + 1$ distinct proc.
FTSA Algorithm - *Time and Bounds*

Time complexity of FTSA: $O(em^2 + v \log \omega)$

$e$: nb edges, $m$: nb procs, $v$: nb tasks, $\omega$: graph width

**Lower Bound $M^*$**

$\forall 1 \leq j \leq m, \quad \mathcal{F}(t, P_j)$ computed as in the algorithm

$\rightarrow M^* = \max_t \left\{ \min_{1 \leq k \leq \varepsilon + 1} \{ \mathcal{F}(t^k, P(t^k)) \} \right\}$

first replica to complete
FTSA Algorithm - *Time and Bounds*

Time complexity of FTSA: \( O(em^2 + v \log \omega) \)
\( e \): nb edges, \( m \): nb procs, \( v \): nb tasks, \( \omega \): graph width

**Lower Bound \( M^* \)**

\[ \forall 1 \leq j \leq m, \quad F(t, P_j) \text{ computed as in the algorithm} \]
\[ \rightarrow M^* = \max_t \left\{ \min_{1 \leq k \leq \epsilon + 1} \{ F(t^k, P(t^k)) \} \right\} \quad \text{first replica to complete} \]
FTSA Algorithm - *Time and Bounds*

Time complexity of FTSA: $O(em^2 + v \log \omega)$

$e$: nb edges, $m$: nb procs, $v$: nb tasks, $\omega$: graph width

**Lower Bound $M^*$**

\[ \forall 1 \leq j \leq m, \quad F(t, P_j) \text{ computed as in the algorithm} \]

\[ \rightarrow M^* = \max_t \left\{ \min_{1 \leq k \leq \varepsilon + 1} \{ F(t^k, P(t^k)) \} \right\} \quad \text{first replica to complete} \]

**Lower Bound $M^*$**

\[ \forall 1 \leq j \leq m, \quad F(t, P_j) = \]

\[ E(t, P_j) + \max_{t_\ast \in \Gamma^- (t)} \left\{ \min_{1 \leq k \leq \varepsilon + 1} \{ F(t_\ast^k, P(t_\ast^k)) + W(t_\ast^k, t) \} \right\}, r(P_j) \]

\[ \rightarrow M^* = \max_t \left\{ \min_{1 \leq k \leq \varepsilon + 1} \{ F(t^k, P(t^k)) \} \right\} \]
FTSA Algorithm - Time and Bounds

Time complexity of FTSA: $O(em^2 + v \log \omega)$

- $e$: nb edges, $m$: nb procs, $v$: nb tasks, $\omega$: graph width

**Lower Bound $\mathcal{M}^*$**

$\forall 1 \leq j \leq m, \quad \mathcal{F}(t, \mathcal{P}_j)$ computed as in the algorithm

$\rightarrow \mathcal{M}^* = \max_t \left\{ \min_{1 \leq k \leq \varepsilon + 1} \{ \mathcal{F}(t^k, \mathcal{P}(t^k)) \} \right\}$

- first replica to complete

**Upper Bound $\mathcal{M}$**

$\forall 1 \leq j \leq m, \quad \mathcal{F}(t, \mathcal{P}_j) = \mathcal{F}(t, \mathcal{P}_j) = \mathcal{E}(t, \mathcal{P}_j) + \max_{t^*_* \in \Gamma^-(t)} \left\{ \max_{1 \leq k \leq \varepsilon + 1} \{ \mathcal{F}(t^k, \mathcal{P}(t^k)) + W(t^k, t) \} \right\}$

$\rightarrow \mathcal{M} = \max_t \left\{ \max_{1 \leq k \leq \varepsilon + 1} \{ \mathcal{F}(t^k, \mathcal{P}(t^k)) \} \right\}$

- longest possible execution time
FTSA Algorithm - Properties

Property 1: **Space exclusion**

For an active replication scheme, a task \( t \in G \) is guaranteed to execute in the presence of \( \varepsilon \) failures if and only if \( P(t^k) \neq P(t^{k'}) \), \( 1 \leq k, k' \leq \varepsilon + 1 \).

Property 2: **Achieved latency**

The latency achieved by FTSA is \( L \leq M \) despite \( \varepsilon \) failures.

Theorem: **Fault tolerant schedule**

If at most \( \varepsilon \) failures occur in the system, then the schedule remains valid.

All to all mapping communications.

Anne.Benoit@ens-lyon.fr APDCM - April 14, 2008
FTSA Algorithm - Properties

Property 1: Space exclusion
For an active replication scheme, a task \( t \in G \) is guaranteed to execute in the presence of \( \varepsilon \) failures if and only if \( P(t^k) \neq P(t^{k'}) \), \( 1 \leq k, k' \leq \varepsilon + 1 \).

Property 2: Achieved latency
The latency achieved by FTSA is \( L \leq M \) despite \( \varepsilon \) failures.

Theorem: Fault tolerant schedule
If at most \( \varepsilon \) failures occur in the system, then the schedule remains valid.

All to all mapping communications.
FTSA Algorithm - *Properties*

**Property 1: Space exclusion**

For an active replication scheme, a task \( t \in G \) is guaranteed to execute in the presence of \( \varepsilon \) failures if and only if \( P(t^k) \neq P(t^{k'}) \), \( 1 \leq k, k' \leq \varepsilon + 1 \).

**Property 2: Achieved latency**

The latency achieved by FTSA is \( L \leq M \) despite \( \varepsilon \) failures.

**Theorem: Fault tolerant schedule**

If at most \( \varepsilon \) failures occur in the system, then the schedule remains valid.

All to all mapping communications
MC-FTSA Algorithm

Idea: Try to decrease communication overhead from $e(\varepsilon + 1)^2$ down to at most $e(\varepsilon + 1)$

- consider mapping returned by FTSA
- enforce internal communication
- greedily select the edges in non decreasing weights order
Experimental results

Aim

- Evaluation of FTSA and MC-FTSA performance
- Comparison with FTBAR heuristic [Girault et al’04]
  (integrated in SynDex: Synchronized Distributed Executive)
- Comparison with fault-free schedule ($\varepsilon = 0$)

Simulation parameters

- 20 processors, 1 – 5 failures
- random graphs, 100 – 150 tasks, granularity [0.2, 2]
  (comp/comm ratio)

Metrics

- Latency bounds, latency with crash
- Overhead = \( \frac{\text{FTSA}^l b | \text{FTBAR}^l b | \text{FTSA}^c | \text{FTBAR}^c - \text{FTSA}^*}{\text{FTSA}^*} \)
Experimental results

Aim

- Evaluation of FTSA and MC-FTSA performance
- Comparison with FTBAR heuristic [Girault et al’04]
  (integrated in SynDex: Synchronized Distributed Executive)
- Comparison with fault-free schedule (\(\varepsilon = 0\))

Simulation parameters

- 20 processors, 1 – 5 failures
- random graphs, 100 – 150 tasks, granularity [0.2, 2] (comp/comm ratio)

Metrics

- Latency bounds, latency with crash
- Overhead = \[\frac{|FTSA^b - FTSA^c - FTSA^*|}{FTSA^*}\]
Experimental results

**Aim**

- Evaluation of FTSA and MC-FTSA performance
- Comparison with FTBAR heuristic [Girault et al’04] (integrated in **SynDex**: Synchronized Distributed Executive)
- Comparison with fault-free schedule ($\varepsilon = 0$)

**Simulation parameters**

- 20 processors, 1 – 5 failures
- random graphs, 100 – 150 tasks, granularity [0.2, 2] (comp/comm ratio)

**Metrics**

- Latency **bounds**, latency with crash
- Overhead = \( \frac{\text{FTSA}^\ell b|\text{FTBAR}^\ell b|\text{FTSA}^c|\text{FTBAR}^c - \text{FTSA}^*}{\text{FTSA}^*} \)
Bounds ($\varepsilon = 1, \varepsilon = 5$)

- FTSA lower bound close to fault-free schedule
- FTSA lower bound better than FTBAR lower bound
- MC-FTSA: upper bound close to lower bound
Latency and overhead with crash ($\varepsilon = 2$)

- **Execution slightly slower when crashes occur**
- **MC-FTSA**: bigger latency (less comm links)
- **MC-FTSA**: still better than FTBAR in some cases
Latency and overhead with crash ($\varepsilon = 5$)

- Similar to case $\varepsilon = 2$
- Many failures: FTBAR better than MC-FTSA with crash
### Running times in seconds

<table>
<thead>
<tr>
<th>Number of tasks</th>
<th>FTSA</th>
<th>MC-FTSA</th>
<th>FTBAR</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>0.01</td>
<td>0.02</td>
<td>0.15</td>
</tr>
<tr>
<td>500</td>
<td>0.08</td>
<td>0.12</td>
<td>4.19</td>
</tr>
<tr>
<td>1000</td>
<td>0.16</td>
<td>0.24</td>
<td>17.10</td>
</tr>
<tr>
<td>2000</td>
<td>0.30</td>
<td>0.50</td>
<td>71.22</td>
</tr>
<tr>
<td>3000</td>
<td>0.46</td>
<td>0.75</td>
<td>167.57</td>
</tr>
<tr>
<td>5000</td>
<td>0.77</td>
<td>1.28</td>
<td>465.75</td>
</tr>
</tbody>
</table>

$|\mathcal{P}| = 50$, $\varepsilon = 5$, *language*: C,  
*machine*: Core 2 Duo (CPU 1.66 GHz)
Conclusion

Efficient Fault Tolerant Scheduling Algorithm FTSA

- Based on active replication scheme
- Aims at minimizing latency while supporting failures
- Low time complexity
- Better than standard FTBAR heuristic
- Different objective functions: fixed latency

Future work

- Maximize system reliability (failure probabilities)
- Multicriteria (reliability, failures and latency) scheduling
- Realistic comm. model (one-port, bounded multi-port)
- Already results, good behavior of MC-FTSA