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 ε 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 ε 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 ε proc. failures

Problem and solutions

Bi-criteria problem

Find a distributed schedule on heterogeneous platforms which minimizes latency $\mathcal{L}$ while tolerating ε 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 $\mathcal{L}$ while tolerating ε 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)

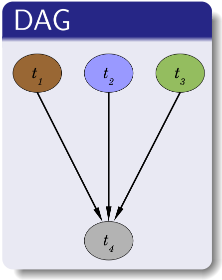
Problem and solutions

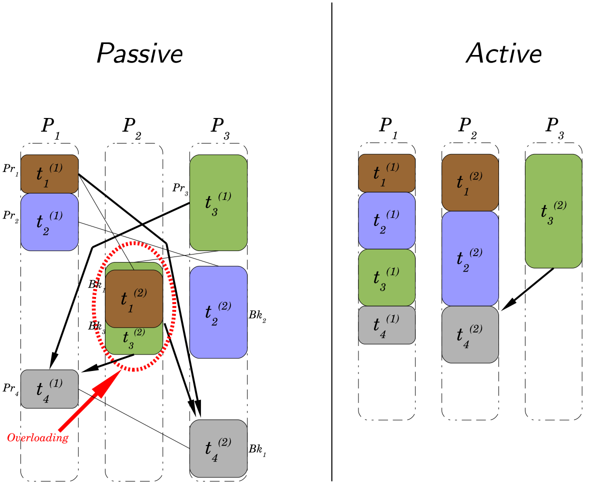
Bi-criteria problem

Find a distributed schedule on heterogeneous platforms which minimizes latency $\mathcal{L}$ while tolerating ε 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 $\rightarrow G = (V, E)$
- $\Gamma^-(t)$, $\Gamma^+(t)$: 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: $t\ell(t) + b\ell(t)$

Basic definitions and notations

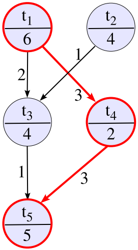
- Parallel application: DAG $\rightarrow$ $G = (V, E)$
- $\Gamma^{-}(t)$, $\Gamma^{+}(t)$: set of predecessors and successors of t
- Free task: all predecessors are already scheduled
- Top level $t\ell$ of a free task: computed from predecessors top levels (including communication)
- Bottom level $b\ell$ 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: $t\ell(t) + b\ell(t)$

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



- $t\ell(t_4) = 9$
- $b\ell(t_4) = 10$
- $\text{Priority}(t_4) = 19$

A brief description of FTSA algorithm

Principle

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

The algorithm:

- Select a critical free task t (keep ordered list)
- Simulate its mapping on all processors using equation:
  $\forall\ 1 \leq j \leq m, \quad \mathcal{F}(t, \mathcal{P}_j) =$
  $$\mathcal{E}(t, \mathcal{P}_j) + \max\left(\max_{t_* \in \Gamma^-(t)} \left\{ \min_{k=1}^{\varepsilon+1} \{ \mathcal{F}(t_*^k, \mathcal{P}(t_*^k)) + W(t_*^k, t) \} \right\}, r(\mathcal{P}_j) \right)$$
- 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 ε of arbitrary processor failures

The algorithm:

- Select a critical free task t (keep ordered list)
- Simulate its mapping on all processors using equation:
  $\forall\ 1 \le j \le m, \quad \mathcal{F}(t, \mathcal{P}_j) =$
  $$\mathcal{E}(t, \mathcal{P}_j) + \max\left(\max_{t_* \in \Gamma^-(t)} \left\{ \min_{k=1}^{\varepsilon+1} \left\{ \mathcal{F}(t_*^k, \mathcal{P}(t_*^k)) + W(t_*^k, t) \right\} \right\}, r(\mathcal{P}_j) \right)$$
- Keep $\varepsilon + 1$ processors allowing *minimum finish time* of t;
- Schedule $t^k, 1 \le k \le \varepsilon + 1$ 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 ε of arbitrary processor failures

The algorithm:

- Select a critical free task t (keep ordered list)
- Simulate its mapping on all processors using equation: $\forall\ 1 \le j \le m, \quad \mathcal{F}(t, \mathcal{P}_j) =$ $$\mathcal{E}(t, \mathcal{P}_j) + \max\left(\max_{t_* \in \Gamma^-(t)} \left\{ \min_{k=1}^{\varepsilon+1} \left\{ \mathcal{F}(t_*^k, \mathcal{P}(t_*^k)) + W(t_*^k, t) \right\} \right\}, r(\mathcal{P}_j) \right)$$
- Keep $\varepsilon + 1$ processors allowing *minimum finish time* of t;
- Schedule $t^k, 1 \le k \le \varepsilon + 1$ 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 ε of arbitrary processor failures

The algorithm:

- Select a critical free task t (keep ordered list)
- Simulate its mapping on all processors using equation:
  $\forall\ 1 \leq j \leq m, \quad \mathcal{F}(t, \mathcal{P}_j) =$
  $$\mathcal{E}(t, \mathcal{P}_j) + \max\left(\max_{t_* \in \Gamma^-(t)} \left\{ \min_{k=1}^{\varepsilon+1} \left\{ \mathcal{F}(t_*^k, \mathcal{P}(t_*^k)) + W(t_*^k, t) \right\} \right\}, r(\mathcal{P}_j) \right)$$
- 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 ε of arbitrary processor failures

The algorithm:

- Select a critical free task t (keep ordered list)
- Simulate its mapping on all processors using equation:
$$\forall\ 1 \leq j \leq m, \quad \mathcal{F}(t,\mathcal{P}_j) = \mathcal{E}(t,\mathcal{P}_j) + \max\left(\max_{t_* \in \Gamma^-(t)} \left\{ \min_{k=1}^{\varepsilon+1} \left\{ \mathcal{F}(t_*^k, \mathcal{P}(t_*^k)) + W(t_*^k, t) \right\} \right\}, r(\mathcal{P}_j) \right)$$
- 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 ε of arbitrary processor failures

The algorithm:

- Select a critical free task t (keep ordered list)
- Simulate its mapping on all processors using equation:
  $\forall\ 1 \leq j \leq m, \quad \mathcal{F}(t, \mathcal{P}_j) =$
  $$\mathcal{E}(t, \mathcal{P}_j) + \max\left(\max_{t_* \in \Gamma^-(t)} \left\{ \min_{k=1}^{\varepsilon+1} \{\mathcal{F}(t_*^k, \mathcal{P}(t_*^k)) + W(t_*^k, t)\} \right\}, r(\mathcal{P}_j) \right)$$
- 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, ω: 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

FTSA Algorithm - *Time and Bounds*

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

FTSA Algorithm - *Time and Bounds*

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

Lower Bound $\mathcal{M}^*$

$\forall\ 1 \leq j \leq m, \quad \mathcal{F}(t, \mathcal{P}_j) =$

$$\mathcal{E}(t, \mathcal{P}_j) + \max\left(\max_{t_* \in \Gamma^-(t)} \left\{ \min_{1 \leq k \leq \varepsilon+1} \{ \mathcal{F}(t_*^k, \mathcal{P}(t_*^k)) + W(t_*^k, t) \} \right\}, r(\mathcal{P}_j) \right)$$

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

FTSA Algorithm - *Time and Bounds*

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

e: nb edges, m: nb procs, v: nb tasks, ω: 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 $\mathcal{M}$

$\forall\ 1 \leq j \leq m, \quad \mathcal{F}(t, \mathcal{P}_j) =$

$$\mathcal{E}(t, \mathcal{P}_j) + \max\left(\max_{t_* \in \Gamma^-(t)} \left\{ \max_{1 \leq k \leq \varepsilon+1} \{\mathcal{F}(t_*^k, \mathcal{P}(t_*^k)) + W(t_*^k, t)\} \right\}, r(\mathcal{P}_j) \right)$$

$\rightarrow \mathcal{M} = \max_t \left\{ \max_{1 \leq k \leq \varepsilon+1} \{\mathcal{F}(t^k, \mathcal{P}(t^k))\} \right\}$ 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 ε failures if and only if $\mathcal{P}(t^k) \neq \mathcal{P}(t^{k'}), 1 \leq k, k' \leq \varepsilon + 1$

Property 2: *Achieved latency*

The latency achieved by FTSA is $\mathcal{L} \leq \mathcal{M}$ despite ε failures

Theorem: *Fault tolerant schedule*

If at most ε 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 ε failures if and only if $\mathcal{P}(t^k) \neq \mathcal{P}(t^{k'}), 1 \leq k, k' \leq \varepsilon + 1$

Property 2: *Achieved latency*

The latency achieved by FTSA is $\mathcal{L} \leq \mathcal{M}$ despite ε failures

Theorem: *Fault tolerant schedule*

If at most ε 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 ε failures if and only if $\mathcal{P}(t^k) \neq \mathcal{P}(t^{k'}), 1 \leq k, k' \leq \varepsilon + 1$

Property 2: *Achieved latency*

The latency achieved by FTSA is $\mathcal{L} \leq \mathcal{M}$ despite ε failures

Theorem: *Fault tolerant schedule*

If at most ε failures occur in the system, then the schedule remains valid

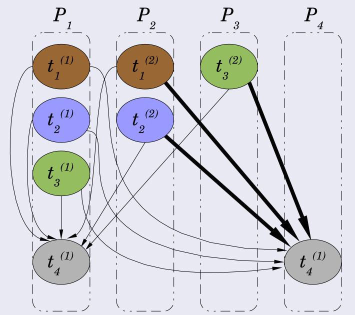
All to all mapping communications

Communication overhead reduction and MC-FTSA algorithm

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 $= \frac{\mathrm{FTSA}^{\ell b} | \mathrm{FTBAR}^{\ell b} | \mathrm{FTSA}^{c} | \mathrm{FTBAR}^{c} - \mathrm{FTSA}^{*}}{\mathrm{FTSA}^{*}}$

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 $= \frac{\text{FTSA}^{\ell b}|\text{FTBAR}^{\ell b}|\text{FTSA}^{c}|\text{FTBAR}^{c} - \text{FTSA}^{*}}{\text{FTSA}^{*}}$

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

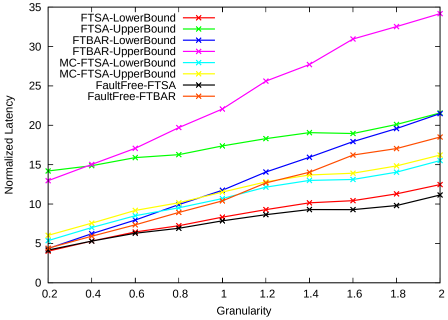
- 20 processors, $1 - 5$ failures
- random graphs, $100 - 150$ tasks, granularity $[0.2, 2]$ (comp/comm ratio)

Metrics

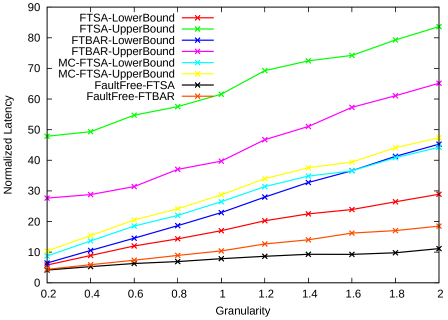
- Latency bounds, latency with crash
- Overhead $= \frac{\mathrm{FTSA}^{\ell b}|\mathrm{FTBAR}^{\ell b}|\mathrm{FTSA}^{c}|\mathrm{FTBAR}^{c} - \mathrm{FTSA}^{*}}{\mathrm{FTSA}^{*}}$

Bounds ($\varepsilon = 1, \varepsilon = 5$)

$\varepsilon = 1$



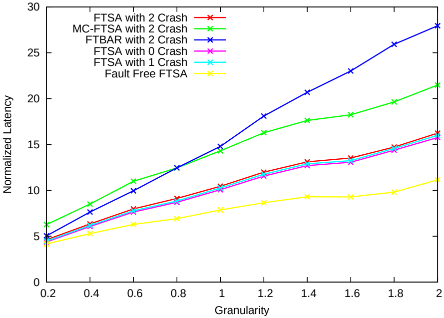
$\varepsilon = 5$



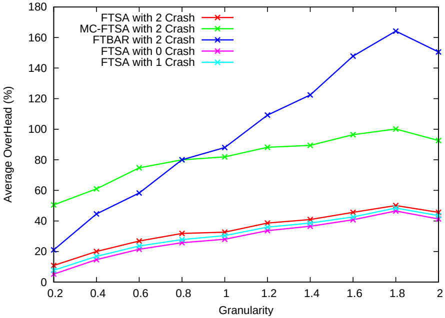
- FTSA lower bound close to fault-free schedule
- FTSA lower bound better than FTBAR lower bound
- MC-FTSA: upper bound close to lower bound

Latency and overhead with crash ($\varepsilon = 2$)

Latency with crash ($\varepsilon = 2$)



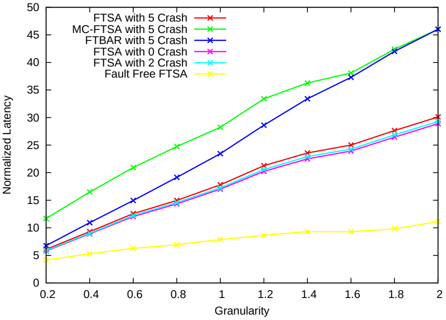
Overhead with crash ($\varepsilon = 2$)



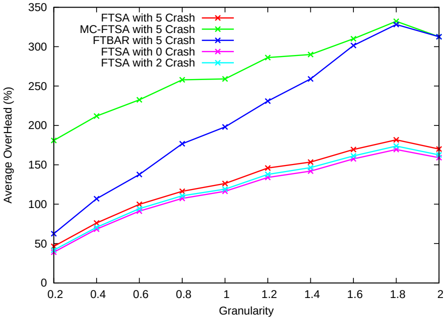
- Execution slightly slower when crashes occur
- MC-FTSA: bigger latency (less comm links)
- MC-FTSA: still better than FTBAR in some cases

Latency and overhead with crash ($\varepsilon = 5$)

Latency with crash ($\varepsilon = 5$)



Overhead with crash ($\varepsilon = 5$)



- Similar to case $\varepsilon = 2$
- Many failures: FTBAR better than MC-FTSA with crash

Running times in seconds

Number of tasks	FTSA	MC-FTSA	FTBAR
100	0.01	0.02	0.15
500	0.08	0.12	4.19
1000	0.16	0.24	17.10
2000	0.30	0.50	71.22
3000	0.46	0.75	167.57
5000	0.77	1.28	465.75

$|\mathcal{P}| = 50$, $\varepsilon = 5$, *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*