Efficient Feasibility Analysis of DAG Scheduling with Timing Constraints in the Presence of Faults

Author:
Xiaotong Cui, Jun Zhang, Kaijie Wu, Edwin Sha.
College of Computer Science, Chongqing Univ., Chongqing, China.
Outline

• Introduction
  – Real-time systems and applications
  – DAG scheduling
  – Fault-tolerance

• Problem Definition

• Our Technique

• Experiments

• Conclusion
Real-Time systems and Applications

• Real-time systems and applications become more common in our lives.

• The total correctness of an operation in real-time systems depends upon:
  – its logical correctness
  – the time it used
DAG Scheduling on Multi-Cores

- We usually use weighted directed acyclic graph (WDAG) to model a set of tasks.

![Diagram of a directed acyclic graph (WDAG) with tasks T1 to T6 and data dependencies.]

Communication delay on
The same processor: 0
Different processors: 1

Fig. A scheduled task set
Guaranteeing logical correctness

• However, the execution of a task may not be successful sometimes.
• In real-time systems, finishing execution before deadline is important though, logical correctness is fundamental.
• So it is necessary to provide fault-tolerance in real-time schedules.
Fault-tolerant Mechanism

- Fault-tolerance is built into the schedule using two types of backups:
  - Active backup
  - Passive backup
For the sake of simplicity and energy conservation, in our problem, we use passive backup to build fault-tolerance following two rules.

– Backup is executed immediately after the error task;
– All tasks dependent on it are delayed.

It should be noted that the re-executed task may incur error again.
Problem Definition

• In multi-core systems, when given
  – A scheduled task set with $N$ related tasks, say $T=\{t_1, t_2, \ldots, t_n\}$;
  – the maximum number of faults that could occur during the execution frame in the system, say $X$.

• Then, what’s the worst-case finish time ($WCFT$) of this scheduled task set?
• After finding $WCFT$, then the feasibility can be found, too.
Problem complexity

• Exactly, if a task set consists of \( N \) tasks and is subject to a maximum number of \( X \) faults, there would be

\[
\binom{N + X - 1}{X}
\]

distinct cases of fault occurrences.

• With a given fault occurrence, to compute the WCFT of the scheduled task set, the time complexity will be \( O(N^2) \), which is the longest path of the scheduled task set.

• So, the total time complexity would be

\[
O \left( \frac{(N + X - 1)!}{X! (N - 1)!} \frac{N^2}{X!} \right)
\]

if all cases are computed which is very exhaustive.
Properties of a task set

- For a task set modeled by WDAG, we add two dummy vertices.
  - $T_{sr}$: has an edge for each vertex that has no incoming edge in original DAG,
  - $T_{sk}$: has an edge for each vertex that has no outgoing edge in original DAG.
  - One task $T$’s critical paths: the longest paths from $T_{sr}$ to $T$

- Let $T_c$ be the current task under investigation,
  - $PS_{T_c}$ be the Parents Set of $T_c$,
  - $AS_{T_c}$ be the Ancestors Set of $T_c$.
  - Obviously, $PS_{T_c} \subseteq AS_{T_c}$,
Some Critical Proofs

• Lemma 1: \( T \) is not in any critical paths of \( T_c \), if critical paths of \( T_c \) don’t change when \( T \) incurs \( x \) faults, reducing \( x \) will not affect the finish time of \( T_c \).
Some Critical Proofs

• Lemma 2: if more than one task in AST\(_c\) incur faults, a worse or status quo finish time of T\(_c\) can be always be found by letting one task \(T \in \text{AST}c\) incur all the X faults.

Theorem: there exists at least one critical task for any task T\(_c\) such that if this task incurs all the expected X faults, task T\(_c\) experiences its worst-case finish time.
Preprocessing

- When given a scheduled task set and its original data dependency modeled by DAG, there are two kinds of dependency.
  - data dependency
  - schedule dependency
Our Technique

- Base on our theorem, we can solve the problem recursively.

\[
\begin{align*}
WCFT(T_1) &= C_{T_1} + X \cdot C_{T_1} \quad CT(T_1) = T_1 \\
WCFT(T_2) &= C_{T_2} + X \cdot C_{T_2} \quad CT(T_2) = T_2 \\
WCFT(T_3) &= \text{Max} \\
BCFT(T_3) + X \cdot C_{T_3}
\end{align*}
\]
Recursively solve the problem

Ancestors set of task $T_{sk}$

$$\begin{align*}
WCFT(T_{sk}) &= \max \left\{ WCFT(T_5) + W(T_5, T) + C_{Tsk}, BCFT(T_{sk}) + X \times C_{Tsk} \right\} \\
&= WCFT(T_5) + W(T_5, T) + C_{Tsk}
\end{align*}$$

Else

$$CT(T_{sk}) = CT(T_6)$$

task $T_{sk}$

Solve the problem recursively
Experimental Setup

• A common practice that bet WCFT using the task with longest execution time could under-estimate the finish time of the task set.

• We compare the WCFT obtained by our algorithm with the finish time when all faults occur on the task with the longest execution time.

• Six benchmarks from DSPstone are used.
Conclusion

• Given a task set with N tasks and X being the maximum number of faults could occur, we conclude that there exists at least one critical task for each task.

• A task undergoes it worst-case finish time when one of its critical tasks incurs all X faults.

• We propose a recursive algorithm which can identify the critical task and the worst-case finish time of a scheduled task set.

• For a task set with N tasks and The algorithm takes only $O(N^2)$. 
Thank You!