2001年度日本オペレーションズ・リサーチ学会
春季研究発表会

コスト有効性に基づいた通信ソフトウェアシステムに対する予防保全スケジュールの決定

土肥正 † (01307065), 海生直人 ‡ (01105445), 尾崎俊治 † (01002265)
† 広島大学大学院, ‡ 広島修道大学

1. はじめに

現在，コンピュータネットワークの爆発的な普及により，実時間ネットワークシステムの信頼性を評価するための手法を確立することが急務となっている．特に，ネットワークシステムの障害は，ハードウェア障害よりもむしろソフトウェアによる障害に起因することが多く，ソフトウェアシステムの信頼性を向上させることが最重要課題となっている．ソフトウェアの障害を大別すると，

- ソフトウェアプログラムに含まれる固有フォールトによる障害
- ソフトウェアシステムの経年劣化による障害

に分類される．前者は，ソフトウェアの開発工程中に作り込まれたフォールト（バグ）が運用期間中に表面化することを意味し，後者は，ソフトウェアの運用期間が経過するにつれてソフトウェアシステムの内部構造が変化することにより生じる障害を意味する．このような現象はソフトウェアエージング (software aging) と呼ばれ [1]，オペレーティングシステムやミッドウェア系システムだけでなく，通信ソフトウェア [2-4] や Netscape, xrn といった典型的なアプリケーションソフトウェアの運用時においても頻繁に観測されている．

ソフトウェアエージングによる障害は一過性の障害 (transient failure) であることが多い．すなわち，障害が発生した後，若干異なる内容（データ，環境）でシステムをリトライすることにより，あたかも障害が発生していなかったかのような操作可能状況に復帰する可能性がある．反面，このような一過性の障害は，ソフトウェアシステムのソースコード上で障害の原因を特定することが極めて困難であることから，その対処法について数多くの研究がなされてきた．特に，ソフトウェア若化 (software rejuvenation) と呼ばれる方策は，ソフトウェアエージングによる一過性の障害を予防するための有効な方法として認識されており，ソフトウェアシステムの稼働を一時的に停止し，その内部構造を浄化した後にシステムを再稼働する一連の予防保全手続きを意味する [2-4]．ここで，ソフトウェアシステムの内部構造の浄化とは，ガーベージコレクション (garbage collection) やオペレーティングシステムにおけるカーネルテーブル (operating system kernel tables) の洗浄 (flushing)，データ構造の初期化 (reinitializing) 等を示す．アプリケーションシステムの運用上，極端であるが最も頻繁に行われているソフトウェア若化の一例として，ハードウェアリブート (hardware reboot) が挙げられる．

本稿では，ある通信ソフトウェアシステムを対象に，コスト有効性に基づいた最適なソフトウェア若化 (software rejuvenation) スケジュールを決定するための確率モデルについて考察する．対象となるシステムの確率的挙動をセミマルコフ過程に基づいて記述し，コスト有効性を最大にする最適若化スケジュールを導出する．さらに，システムの故障データが与えられた場合に，最適若化スケジュールをノンパラメトリックに推定するための統計アルゴリズムについても言及する．

2. セミマルコフモデル

Huang *et al.* [2] と同様に，通信アプリケーションソフトウェアの時間的挙動は，正常稼働状態（状態 0），障害が発生可能な状態（状態 1），障害の発生状態（状態 2），ソフトウェア若化状態（状態 3）の 4 状態をもつセミマルコフ過程に従って推移するものと仮定する．いま，システムが正常状態から障害発生可能な状態に推移する時間を非負で連続な確率変数 Z によって表現し，その確率分布関数を $\Pr\{Z \le t\} = F_0(t)$，平均を μ_0 (> 0) とする．システムが障害発生可能な状態に推移すると，ソフトウェア若化を行うか否かの決定をするものとしよう．システムが障害発生可能な状態から具体的に故障に至るまでの時間は，非負で連続な確率変数 X によって記述され，その確率分布関数を $\Pr\{X \le t\} = F_f(t)$，平均を μ_f (> 0) とする．一旦障害が発生すると，事後保全が直ちに開始される．ここで事後保全とは，障害が発生した後に事後的にシステムを若化することを意味し，本稿では修理という言葉によって代用する．修理に要する時間 Y もまた非負の連続形確率変数であり，その確率分布関数を $\Pr\{Y \le t\} = F_a(t)$，平均を μ_a (> 0) とする．修理が完了すると，システムの障害発生率は正常稼働状態における初期状態まで復旧される．

他方，ソフトウェアの予防的な若化は，システムが障害発生可能な状態に推移した後の T 時間経過後になされるものとする．ここで，T は非負で連続な確率変数であり，その確率分布関数を $F_r(t)$，平均を t_0 (> 0) とする．システム障害が発生する前に時刻 T が経過した場合は，直ちに予防的に若化を開始し，そのシステムオーバーヘッド V に対する確率分布関数を $\Pr\{V \le t\} = F_c(t)$，平均を μ_c (> 0) とする．修理の場合と同様に，予防的若化が完了すると，システムの障害発生率は正常稼働状態における初期状態まで復旧される．特に，障害発生可能な状態から予防的に若化を実施するまでの

時間 T が一定であるとすれば，確率分布関数 $F_r(t)$ を次のようなユニット関数

$$F_r(t) = U(t - t_0) = \begin{cases} 1 & \text{if } t \geq t_0 \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

に置き換えればよい．

Dohi *et al.* [3, 4] の結果を直接用いることによって，上述のモデルにおけるコスト有効性は

$$E(t_0) = \frac{\mu_0 + \int_0^{t_0} \overline{F}_f(t)dt}{c_s \mu_a F_f(t_0) + c_p} \tag{2}$$

のように求めることが出来る．ここで，障害発生時の単位時間当たりの修理費用を c_s (> 0)，予防的に若化を行う場合の単位時間当たりの費用を c_p (> 0) とする．

3. ノンパラメトリック推定アルゴリズム

ここでは，実際に障害発生時間データが与えられているという状況において，コスト有効性を最大にする最適ソフトウェア若化スケジュールを（理論分布を仮定することなく）ノンパラメトリックに推定するためのアルゴリズムについて考察する．いま，障害発生時間分布 $F_f(t)$ に対する標準総試験時間変換を次のように定義する．

$$\phi(p) = (1/\mu_f) \int_0^{F_f^{-1}(p)} \overline{F}_f(t)dt \tag{3}$$

ここで，$F_f(t)$ は絶対連続かつ非減少関数であるので，その逆関数

$$F_f^{-1}(p) = \inf\{t_0; F_f(t_0) \geq p\}, \quad 0 \leq p \leq 1 \tag{4}$$

が必ず存在する．よく知られた結果として，確率分布関数 $F_f(t)$ が IFR (DFR) であるための必要かつ十分条件は，関数 $\phi(p)$ が $p \in [0, 1]$ に関して凹 (凸) 関数であることである．

定理 1: コスト有効性 $E(t_0)$ を最大にする最適ソフトウェア若化スケジュールを求める問題は，以下の問題の解 p^* $(0 \leq p^* \leq 1)$ を求めることと等価である．

$$\max_{0 \leq p \leq 1} \frac{\phi(p) + \alpha}{p + \beta} \tag{5}$$

ここで，

$$\alpha = \mu_0/\mu_f \tag{6}$$

$$\beta = c_p \mu_c (c_s \mu_a - c_p \mu_c) \tag{7}$$

次に，障害発生時間分布 $F_f(t)$ は未知であるが，対応する障害発生時間データの順序統計量 $x_0 = 0, x_1, x_2, \cdots, x_n$ が観測されており，これらは打ち切りのない完全データであると仮定する．障害発生時間分布 $F_f(t)$ の推定量を x_j $(j = 0, 1, 2, \cdots, n)$ に基づいた経験分布関数

$$F_n(x) = \begin{cases} j/n & \text{for} \quad x_j \leq x < x_{j+1} \\ 1 & \text{for} \quad x_n \leq x \end{cases} \tag{8}$$

によって定義する．さらに，経験分布関数に基づいた標準総試験時間変換の推定量として，次にような標準総試験時間統計量を定義する．

$$\phi_{nj} = \psi_j / \psi_n \tag{9}$$

ここで，関数

$$\psi_j = \sum_{k=1}^{j} (n - k + 1)(x_k - x_{k-1}) \qquad j = 1, 2, \cdots, n; \ \psi_0 = 0 \tag{10}$$

は総試験時間統計量と呼ばれる．最終的に，点列 $(j/n, \phi_{nj})$ $(j = 0, 1, 2, \cdots, n)$ を 2 次元平面上にプロットし，それらを線分で繋ぐことにより，標準総試験時間プロット (scaled total time on test plot) を得る．推定量 $(j/n, \phi_{nj})$ $(j = 0, 1, 2, \cdots, n)$ は $(p, \phi(p))$ $(p \in [0, 1])$ のノンパラメトリック推定量であるので，定理 1 の結果を直接適用することにより，最適ソフトウェア若化スケジュールに関する以下の定理を得る．

定理 2: 障害発生時間に対する n (> 0) 個の完全データの順序統計量 $x_1 \leq x_2 \leq \cdots \leq x_n$ が観測されているものとする．

(i) コスト有効性を最大にする最適ソフトウェア若化スケジュールのノンパラメトリック推定量は，次の式を満たす x_{j^*} によって与えられる．

$$j^* = \left\{ j \mid \max_{0 \leq j \leq n} \frac{\phi_{nj} + \alpha}{j/n + \beta} \right\} \tag{11}$$

ここで，式 (6) の μ_f はサンプル平均 $\sum_{k=1}^{n} x_k/n$ によって置き換えられる．

(ii) (i) で与えられた最適ソフトウェア若化スケジュールのノンパラメトリック推定量は強一致推定量である．すなわち，x_{j^*} は $n \to \infty$ のとき（未知であるが）真の最適解 $t_0{}^*$ に確率 1 で漸近収束する．

参考文献

[1] D. L. Parnas, "Software aging", *Proc. 16th Int'l Conf. on Software Engineering*, pp. 279–287, ACM Press (1994).

[2] Y. Huang, C. Kintala, N. Kolettis and N. D. Funton, "Software rejuvenation: analysis, module and applications", *Proc. 25th IEEE Int'l Symp. on Fault Tolerant Computing*, pp. 381–390, IEEE Computer Society Press (1995).

[3] T. Dohi, K. Goševa-Popstojanova and K. S. Trivedi, "Analysis of software cost models with rejuvenation," *Proc. 5th IEEE Int'l Symp. on High Assurance Systems Engineering*, pp. 25–34, IEEE Computer Society Press (2000).

[4] T. Dohi, K. Goševa-Popstojanova and K. S. Trivedi, "Statistical non-parametric algorithms to estimate the optimal software rejuvenation schedule", to appear in *Proc. 2000 Pacific Rim Int'l Symp. on Dependable Computing*, IEEE Computer Society Press (2000).