数理解析研究所講究録
第1990巻 2016年 183-190

モバイルクラウドに対する最適データ分割を考慮したソフトウェアの最適メンテナンス問題

山口大学大学院・理工学研究科 田村 慶信（Yoshinobu Tamura）†
†Graduate School of Science and Engineering, Yamaguchi University
鳥取大学大学院・工学研究科 信川 ゆみ（Yumi Nobukawa）‡
鳥取大学大学院・工学研究科 山田 茂（Shigeru Yamada）‡
‡Graduate School of Engineering, Tottori University

1 はじめに

データの一元管理，低コスト，保守・運用が容易といった観点から，OpenStack や Eucalyptus などのオープンソースソフトウェア（open source software, 以下 OSS と略す）を利用したクラウド環境の構築に注目が集まっている．クラウドコンピューティングにおいてもデータ肥大化に伴う大規模データ，すなわちビッグデータが扱われるようになってきた．例えば，クラウド上でビッグデータを扱う場合においては，データベースとクラウドのソフトウェア間でのデータ連携処理の頻度が多くなる．クラウド環境全体におけるソフトウェア故障について考えた場合，クラウドソフトウェア内で発生するものだけではなく，データベースソフトウェア上で発生するソフトウェア故障を考慮することは非常に重要となる．特に，クライアントとデータノードとの間に位置するネームノードに障害が発生した場合，クラウドソフトウェアにおける障害ではなく，データベースソフトウェアとの通信上の障害となる．

ビッグデータを扱うクラウドコンピューティング環境では，主に，ビッグデータを支える Hadoop や NoSQL に代表されるデータベースソフトウェアと，Eucalyptus や OpenStack に代表されるクラウドソフトウェアにより運用されている．最近では，2014年10月に OpenStack の Juno バージョンがリリースされ，OpenStack のフレームワークの中に Hadoop が組み込まれた．Hadoop は大量データの高速処理に適したデータベースソフトウェアであり，ビッグデータを扱う多くのシステムで利用されている．このように，クラウドコンピューティング環境においても，比較的規模の大きなソフトウェアとの連携処理がネットワーク経由で頻繁に行われつつシステムが運用される事例が多くなってきている．

クラウド環境に対する最近の研究動向としては，モバイルクラウド，サービス形態，性能評価等を対象とした文献はいくつか提案されているが [1,2]，ビッグデータによるデータ肥大化に伴うクラウドを対象とした信頼性評価に関する研究は行われていない．従来から，ソフトウェア製品の開発プロセスにおけるテスト進捗管理や出荷品質の把握のための信頼性評価を行うアプローチとして，ソフトウェア故障の発生現象を不確定事象として捉えて確率・統計論的に取り扱う方法がとられている．その1つが，ソフトウェア信頼性モデル（Software Reliability Model, 以下 SRM と略す）である [3]．これまでに数百におよぶソフトウェア信頼性モデルが提案されてきた [3–5]．しかしながら，既存のソフトウェア信頼性モデルの多くは大規模ソフトウェア間の通信環境に伴うフォールト発生事象が考慮されていない．クラウド上のビッグデータを想定したソフトウェアに対して既存のソフトウェア信頼性モデルを適用することは可能であっても，クラウドとビッグデータの信頼性評価に関する新たな知見を得ることはできない．クラウド上のビッグデータを想定したソフトウェアシステム全体の信頼性評価が可能となり，その安全性が確保されれば，その普及は爆発的に増加するものと思われる．近い将来，モバイルクラウドに代表

されるように，データが手元にある不安の方が大きくなる時代を切り開くためには，ビッグデータを想定したクラウド環境の信頼性に関する課題解決が特に重要となる．

特に，モバイルクラウドの運用環境では，ネットワークに接続した状態で常時運用が行われている．こうしたモバイルクラウドのネットワーク環境から受けるソフトウェア信頼性への影響を考慮することは重要である．最近では，モバイルクラウドの特徴から，ビッグデータを扱うデータベースソフトウェアがクラウドソフトウェア内に統合されつつある．こうした状況から，データベース上においても，クラウドにアクセスし管理を行うためのクラウドベースデータと，デジタル画像や音楽データなどのコンテンツデータが蓄積されるようになっている．これらのコンテンツデータはストレージ容量の肥大化を招くだけでなく，ソフトウェアシステムの複雑化や運用管理の煩雑化を招く原因の一つとなっている．

本論文では，信頼性の観点からデータベース上における最適なデータ分割状態を考慮するために，クラウドソフトウェアとデータベースソフトウェアからのフォールト発生状況を分析する．また，モバイルクラウド環境全体の信頼性を評価するためにジャンプ拡散過程モデルを適用する．これにより，保存されたデータベース上におけるコンテンツデータとクラウドベースデータとの関係性を信頼性の観点から分析することが可能となる．さらに，データベースソフトウェアとクラウドソフトウェアとの最適なデータ分割を考慮した提案モデルに基づく最適メンテナンス問題について議論するとともに，実際のソフトウェアフォールト発見数データに基づく提案手法に対する数値例を示す．

2　ニューラルネットワークに基づくソフトウェア構成比率の推定

モバイルクラウド環境下におけるソフトウェア構成比率は，クラウドのソフトウェア構成，利用形態，ユーザ数，およびハードウェア構成要素など，様々な要因により影響を受ける．こうした種々の環境要因をパラメータとして考慮し，物理的な観点からモデル化することは可能であっても，それを実際のソフトウェア運用環境に適用することは困難であると思われる．ここでは，モバイルクラウドを構成するクラウドソフトウェアとデータベースソフトウェアにおけるソフトウェア構成比率を信頼性の観点から評価するために，ノンパラメトリックな手法であるニューラルネットワーク [6] に基づく時系列分析手法を利用する．

本論文では，簡単のために 3 層ニューラルネットワークを適用する．このとき，クラウドソフトウェアの一部としてビッグデータを扱うことが可能なデータベースソフトウェアが組み込まれているものと仮定し，クラウドソフトウェアに対するデータベースソフトウェアの累積フォールト発見数の比率を入力データとして適用する．

まず，$w_{ij}^1(i=1,2,\cdots,I;j=1,2,\cdots,J)$ を入力層と中間層の結合係数，また $w_{jk}^2(j=1,2,\cdots,J;k=1,2,\cdots,K)$ は中間層と出力層の結合係数とする．さらに，$x_i(i=1,2,\cdots,I)$ は正規化された入力データを表し，本論文では，時刻 t におけるネットワークトラフィックの変化率 $x_t(t=1,2,\cdots,I)$ とした．ここで，入力層，中間層，出力層におけるユニットの数を，それぞれ I 個，J 個，および K 個とする．また，各層のユニットを示すインデックスを i，j，および k とする．ここで，各層のユニットの出力を h_j, y_k とすると，

$$h_j = f\left(\sum_{i=1}^{I} w_{ij}^1 x_i\right), \tag{1}$$

$$y_k = f\left(\sum_{j=1}^{J} w_{jk}^2 h_j\right), \tag{2}$$

となる．但し，$f(\cdot)$ はシグモイド型関数であり，

$$f(x) = \frac{1}{1+e^{-\theta x}}, \tag{3}$$

として表される．ここで，θ はしきい値と呼ばれる定数である．ネットワークの学習を行うために，誤

差逆伝播法を用いる．ニューラルネットワークの出力層における値を $y_k(k=1,2,\cdots,K)$ とし，教師パターンを $d_k(k=1,2,\cdots,K)$ とすると，式 (2) の y_k は次式で評価される．

$$E=\frac{1}{2}\sum_{k=1}^{K}(y_k-d_k)^2. \tag{4}$$

ここで，教師パターン $d_k(k=1,2,\cdots,K)$ には，実際に観測されたクラウドソフトウェアに対するデータベースソフトウェアの累積フォールト発見数の比率 $d_t(t=2,3,\cdots,K)$ の正規化された値を採用する．すなわち，時刻 t までにおける実際に観測された累積フォールト発見数の比率に基づいて，各時点における累積フォールト発見数の比率の結合状態の特徴をニューラルネットワークの結合係数に蓄積させ，時刻 $t+1$ における累積フォールト発見数の比率の推定・予測が可能なモデルを考える．式 (4) の条件のもとに，結合係数が最急降下法にて決定される．

3 ジャンプ拡散過程モデル

まず，時刻 $t=0$ で OSS の運用が開始され，任意の時刻 t における検出フォールト数 $\{N(t),t\geq 0\}$ は以下の常微分方程式によって記述されるものと仮定する．

$$\frac{dN(t)}{dt}=b(t)\{a-N(t)\}. \tag{5}$$

ここで，$b(t)(>0)$ は時刻 t におけるフォールト発見率を，a はソフトウェア内に潜在する総フォールト数を示す．また，モバイルクラウド環境の変化に伴う運用形態の特徴を考慮するために，フォールト発見率 $b(t)$ に不規則性を導入すると，式 (5) は，

$$\frac{dN(t)}{dt}=\{b(t)+\sigma\gamma(t)\}\{a-N(t)\}, \tag{6}$$

となる．ここで，$\sigma(>0)$ は定数パラメータ，$\gamma(t)$ は解過程の Markov 性を保証するための標準化された Gauss 型白色雑音を表す．さらに，モバイルクラウドの運用段階におけるフォールト発見事象が，ログインするユーザ数，サービスアプリケーション数の変化，さらにはプロビジョニングプロセスなどにより不規則に変動するものと仮定し，ジャンプ項を導入する [7]．式 (6) を，以下の Itô 型の確率微分方程式 [8] に拡張して考える．

$$dN(t)=\left\{b(t)-\frac{1}{2}\sigma^2\right\}\{a-N(t)\}dt+\sigma\{a-N(t)\}dW(t)+d\left(\sum_{i=1}^{M_t(\lambda)}(V_i-1)\right). \tag{7}$$

ここで，$M_t(\lambda)$ は，$W(t)$ とは独立な強度パラメータ λ をもつポアソン過程であり，時刻 t までにジャンプが発生した回数を表す．λ はジャンプ事象が生じる確率的な頻度であり，V_i は i 回目のジャンプ幅を表す独立な確率変数を意味する．特に，ジャンプ拡散過程のタイプとしては，Merton モデル [7] などにおいて，以下に示す対数正規分布が利用されている．

$$f(x)=\frac{1}{\sqrt{2\pi}\tau x}\exp\left[-\frac{(\log x-\mu)^2}{2\tau^2}\right]. \tag{8}$$

式 (7) の確率微分方程式を Itô の公式を用いて変換すると，

$$N(t)=a\left[1-\exp\left\{-\int_0^t b(t)ds-\sigma W(t)+\sum_{i=1}^{M_t(\lambda)}\log V_i\right\}\right], \tag{9}$$

を得る [9,10]．ここで，$W(t)$ は Wiener 過程であり，形式的には白色雑音の時間積分 $\int_0^t\gamma(s)ds$ で定義されるものである．本論文では，フォールト発見率 $b(t)$ は，次式を満たすものとする．

$$b(t)\simeq\frac{b^2t}{1+bt}. \tag{10}$$

ここで，bはフォールト1個当りのフォールト発見率を表す．このとき，時刻tでの残存フォールト数は，

$$R(t) = a(1+bt)\exp\left\{-bt - \sigma W(t) + \sum_{i=1}^{M_t(\lambda)} \log V_i\right\}, \tag{11}$$

により与えられる．

4 パラメータ推定

4.1 最尤推定法

提案モデルの推移確率分布に含まれているパラメータa，b，およびσは一般には既知ではないので，実測データなどの利用可能なデータを使って値を推定しなければならない．本論文では，未知パラメータを推定する方法として最尤法 (method of maximum-likelihood) を用いる．

運用段階における観測データは，一般に$(t_j, n_j)(j = 1, 2, \cdots, K)$という形で与えられているものとする．ここで$n_j$は，運用時刻$t_j$までに発見された総フォールト数である．確率過程$N(t)$の$K$次の同時確率分布を

$$P(t_1, n_1; t_2, n_2; \cdots; t_K, n_K) = \Pr[N(t_1) \leq n_1, N(t_2) \leq n_2, \cdots, N(t_K) \leq n_K | N(0) = 0], \tag{12}$$

とし，その同時確率密度を

$$p(t_1, n_1; t_2, n_2; \cdots; t_K, n_K) = \frac{\partial^K P(t_1, n_1; t_2, n_2; \cdots; t_K, n_K)}{\partial n_1 \partial n_2 \cdots \partial n_K}, \tag{13}$$

とする．

$N(t)$は連続値を取るので，データ(t_j, n_j)に対し，尤度関数を

$$l = p(t_1, n_1; t_2, n_2; \cdots; t_K, n_K), \tag{14}$$

と表す．さらに，対数尤度関数をLとすると，

$$L = \log l, \tag{15}$$

となり，提案モデルでは，未知パラメータa，b，およびσを同時尤度方程式，

$$\frac{\partial L}{\partial a} = \frac{\partial L}{\partial b} = \frac{\partial L}{\partial \sigma} = 0, \tag{16}$$

の解として得ることができる．

4.2 ジャンプ項に含まれるパラメータ推定

一般的に，確率微分方程式モデルのジャンプ拡散項に含まれるパラメータを推定することは難しいことが知られている．また，ジャンプ拡散項に含まれるパラメータ推定法については，既にいくつか提案されているが，決定的な推定法が存在する訳ではない．本論文では，ジャンプ拡散項に含まれる未知パラメータを推定するために，遺伝的アルゴリズム（Genetic Algorithm，以下GAと略す）[11]により探査対象関数の最小値を到達可能な範囲で捜査する手法を適用する．GAは，生物の遺伝と進化のメカニズムを工学的にモデル化して，さまざまな問題解法やシステムの学習などに応用しようとするものである [11]．コンピュータ上に仮想生命を生成し，その環境に対する適応度を最適化問題の目的関数に一致させ，進化の過程をシミュレーションすることで，最適化問題を解くことが可能となる．

まず，ジャンプ拡散項には，λ, μ, およびτのパラメータが含まれるものと仮定する．ここで，μおよびτは，i番目のジャンプ幅V_iに含まれる発生頻度に関するパラメータである．本論文では，μおよびτは，ニューラルネットワークにより推定された，クラウドソフトウェアに対するデータベースソフトウェアの累積フォールト発見数の比率を適用し，$M_t(\lambda)$に含まれる強度パラメータλを以下のGAにより推定する．

Step. 1 初期個体（染色体）をランダムに生成し，初期個体集合を生成する．その後，評価値の計算を行う．評価値は，初期個体集合を 2 進数にビット変換することにより表される．

Step. 2 任意の 2 つの個体（親）をランダムに選び，選択された個体間の染色体の組み換えにより新しい個体を生成するための交叉を行う．

Step. 3 各個体の評価値から適応度を計算する．適応度は評価値と，バグトラッキングシステム上の採取されたデータの評価関数から算出される．評価関数は以下の式で表される．

$$\min_{\lambda} F_i(\lambda),$$
$$F_i = \sum_{i=0}^{K} \{N(i) - n_i\}^2. \tag{17}$$

ここで，$N(i)$ は，運用時刻 i におけるジャンプ拡散モデルにより推定された総発見フォールト数であり，n_i は，実際の累積発見フォールト数を表す．

Step. 4 世代が一定数に達するまで，Step2 および Step3 を繰り返す．

上述したステップにより，ジャンプ拡散項に含まれるパラメータ λ を推定することができる．

5 最適メンテナンス問題

既存のコスト評価基準に基づくソフトウェアの最適リリース問題 [12,13] を応用し，運用段階における最適メンテナンス問題について議論する．まず，以下のようなコストパラメータを定義する．

c_1: 運用段階におけるフォールト 1 個当りの修正コスト，

c_2: 運用段階における単位時間当りの保守コスト，

c_3: メンテナンス後のフォールト 1 個当りの保守コスト．

このとき，クラウドサービス開始後における運用コストは，以下のように定式化できる．

$$C_1(t) = c_1 N(t) + c_2 t. \tag{18}$$

また，メンテナンス後におけるクラウドソフトウェアの保守コストは次式で与えられる．

$$C_2(t) = c_3 R(t). \tag{19}$$

上記から，クラウドサービスに必要な総期待ソフトウェアコストは，以下のように定式化できる．

$$C(t) = C_1(t) + C_2(t). \tag{20}$$

式 (20) を最小にする時刻 t^* が，クラウドソフトウェアの最適メンテナンス時刻となる．

6 数値例

オープンソースソフトウェアとして開発および公開されている，データベースソフトウェアである Hadoop [14] およびクラウドソフトウェアである OpenStack [15] におけるバグトラッキングシステム上に登録されたフォールトデータを適用した数値例を示す．

まず，累積発見フォールト発見数に基づく **2.** で議論したソフトウェア構成比率の推定結果を図 1 に示す．図 1 から，信頼性の観点から評価した場合において，クラウドソフトウェアに対するデータベースソフトウェアの構成比率は時間の経過とともに一定の比率に収束する様子が確認できる．この結果は，デ

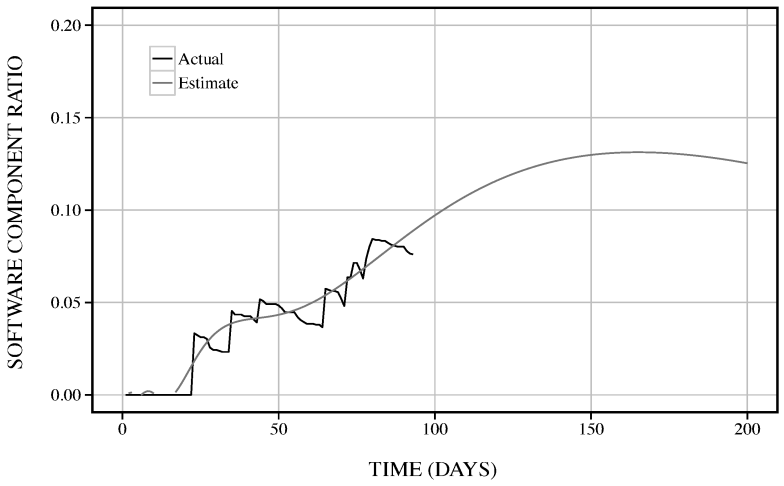

図 1： 累積フォールト発見数に基づくソフトウェア構成比率の推定結果.

ジタル画像や音楽データなどのコンテンツデータを扱うデータベースソフトウェアに対する管理工数の配分の見直しといったようなソフトウェア運用管理上における評価指標として利用できるものと考える.

さらに，図 1 に示すようなモバイルクラウドを構成するデータベースソフトウェアの影響度合いを考慮することにより，保存されたデータ管理上におけるコンテンツデータとクラウドベースデータとの関係性を信頼性の観点から分析することが可能となる．推定された残存フォールト数 $R(t)$ のサンプルパスを図 2 に示す．この結果から，Wiener 過程に基づく雑音が時間の経過とともに小さくなる様子が確認できる．一方，ジャンプ拡散過程に基づく雑音は，フォールト報告の約 100 日前後にピークを迎え，その後は時間の経過とともにジャンプの幅と大きさが小さくなっていく様子が確認できる.

図 1 および図 2 から，クラウドベースデータとコンテンツデータとの最適データ配分状態を信頼性の観点から考慮した場合，約 200 日目において，最適な状態となることが確認できる．このとこから，ソフトウェア管理者は，運用を開始して 200 日目までは，モバイルクラウド環境全体について監視する必要があることが分かる.

式 (20) における推定された総ソフトウェアコストのサンプルパスを図 3 に示す．図 3 から，最適メンテナンス時刻は 153.3 日となり，そのときの総ソフトウェアコストは 1049.2 であることが確認できる．また，図 3 から，フォールト報告終了時点以降から 200 日目までは雑音が大きく，その後は時間の経過とともに徐々に小さくなる様子が確認できる．このことから，従来の期待値に基づく最適メンテナンス時刻 [16] よりも約 50 日経過後にメンテナンスを行うことが望ましいことが分かる.

7 おわりに

本論文では，モバイルクラウドの実利用環境を想定し，信頼性の観点からデータベース上における最適なデータ分割を判断するために，クラウドソフトウェアとデータベースソフトウェアからのフォールト発生状況に基づき，ソフトウェア構成比率をニューラルネットワーク手法により分析した．さらに，モバイルクラウド環境全体の信頼性を評価するためにジャンプ拡散過程モデルを適用した．また，クラウドコンピューティングの最適メンテナンス時刻を推定するために，従来のコスト評価基準に基づくソフトウェアの最適リリース問題を応用することにより，データベースソフトウェアとクラウドソフトウェアとの通信環境を考慮した提案モデルに基づく最適メンテナンス問題について議論した.

実際のクラウド OSS のソフトウェアフォールト発見数データを適用し，ジャンプ拡散過程モデルに対

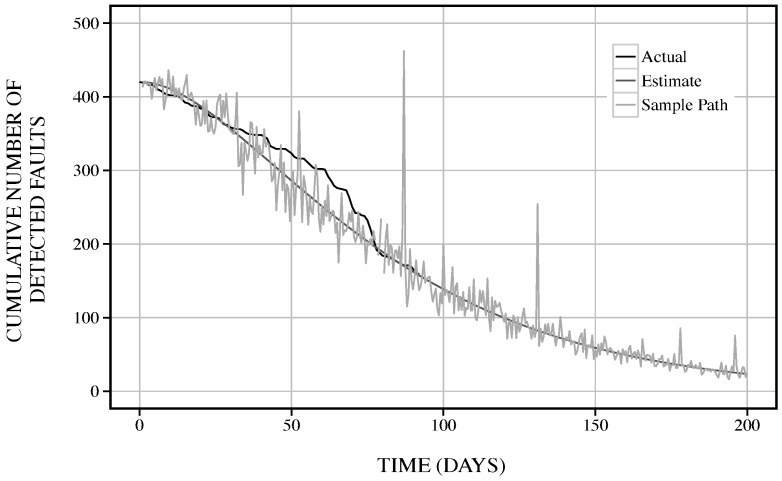

図 2： 推定された残存フォールト発見数.

する数値例を示すことにより，保存されたデータベース上におけるコンテンツデータとクラウドベースデータとの関係性を信頼性の観点から分析することが可能となった．特に，デジタル画像や音楽データなどのコンテンツデータを扱うデータベースソフトウェアに対する管理工数の配分の見直し時期を推定するようなソフトウェア運用管理上における評価指標を示した．また，モバイルクラウド環境全体に対する信頼性評価尺度として，ジャンプ拡散過程モデルに基づく信頼性評価結果を示すとともに，最適メンテナンス時刻の推定例を示した．これにより，クラウドベースデータとコンテンツデータとの最適データ配分状態を信頼性の観点から考慮した総合的な信頼性評価が可能となるものと考える．

謝辞

本研究の一部は，公益財団法人 電気通信普及財団調査研究助成および JSPS 科研費基盤研究（C）（課題番号 15K00102 および 25350445）の援助を受けたことを付記する．

参考文献

[1] H. Suo, Z. Liu, J. Wan, and K. Zhou, "Security and privacy in mobile cloud computing," *Proceedings of the 9th International Wireless Communications and Mobile Computing Conference*, pp. 655–659, 2013.

[2] A. Khalifa and M. Eltoweissy, "Collaborative autonomic resource management system for mobile cloud computing," *Proceedings of the Fourth International Conference on Cloud Computing, GRIDs, and Virtualization*, pp. 115–121, 2013.

[3] S. Yamada, *Software Reliability Modeling: Fundamentals and Applications*, Springer–Verlag, Tokyo/Heidelberg, 2013.

[4] M.R. Lyu, ed., *Handbook of Software Reliability Engineering*, IEEE Computer Society Press, Los Alamitos, CA, 1996.

[5] P.K. Kapur, H. Pham, A. Gupta, and P.C. Jha, *Software Reliability Assessment with OR Applications*, Springer–Verlag, London, 2011.

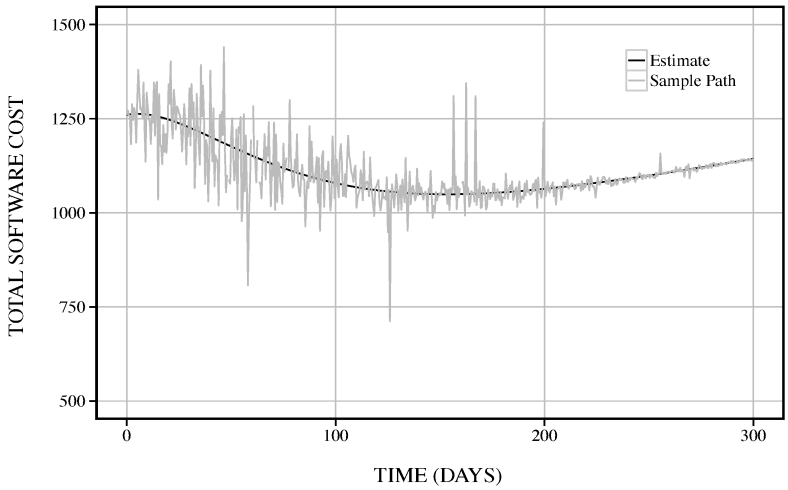

図 3： 推定された総ソフトウェアコスト.

[6] E. D. Karnin, "A simple procedure for pruning back-propagation trained neural networks," *IEEE Transactions on Neural Networks*, vol. 1, June 1990, pp. 239-242.

[7] R.C. Merton, "Option pricing when underlying stock returns are discontinous," *Journal of Financial Economics*, vol. 3, pp. 125–144, 1976.

[8] L. Arnold, *Stochastic Differential Equations-Theory and Applications*, John Wiley & Sons, New York, 1974.

[9] E. Wong, *Stochastic Processes in Information and Systems*, McGraw-Hill, New York, 1971.

[10] S. Yamada, M. Kimura, H. Tanaka, and S. Osaki, "Software reliability measurement and assessment with stochastic differential equations," *IEICE Transactions on Fundamentals*, vol. E77–A, no. 1, pp. 109–116, 1994.

[11] J.H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, 1975.

[12] S. Yamada and S. Osaki, "Cost-reliability optimal software release policies for software systems," *IEEE Transactions on Reliability*, vol. R–34, no. 5, pp. 422–424, 1985.

[13] S. Yamada and S. Osaki, "Optimal software release policies with simultaneous cost and reliability requirements," *European Journal of Operational Research*, vol. 31, no. 1, pp. 46–51, 1987.

[14] The Apache Software Foundation, Apache Hadoop, http://hadoop.apache.org/

[15] The OpenStack project, OpenStack, http://www.openstack.org/

[16] Y. Tamura and S. Yamada, "Optimization analysis based on stochastic differential equation model for cloud computing," *International Journal of Reliability, Quality and Safety Engineering*, vol. 21, no. 4, pp. 1450020-1–1450020-13, 2014.