数理解析研究所講究録
第1939巻 2015年 247-255

クラウド上のビッグデータによる外的要因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 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，事前準備が不要といった観点から，オープンソースを利用したクラウドサービスが多く提供されるようになっている．しかしながら，ソフトウェアの設計図にあたるソースコードが世界中に公開されているため，クラッキングによる情報事故のように，悪意のあるサイト攻撃や情報流出の標的になり易く，なかなか導入に踏み切れないのが現状である．周知の通り，日本政府により2013年10月に公表された「サイバーセキュリティ国際連携取組方針」の中にも，安全・安心にクラウドサービスを利用できるようにするため，クラウドセキュリティ活動の普及とともに，米国を含めた諸外国との連携を推進することが重要課題の一つとして掲げられている．しかしながら，クラウドサービスが普及しつつある現在においても，クラウド関連の障害や情報事故が後を絶たない．クラウドサービスの特徴により，ひとたび障害が発生すれば世界規模のトラブルに波及するとともに，その影響は瞬時に表面化する．ソフトウェアの設計図にあたるソースコードが世界中に公開されているため，最近のソニーPSNクラッキングによる情報事故のように，悪意のあるサイト攻撃や情報流出の標的になり易いだけでなく，Amazon.comのデータセンター大規模障害のように多くのサービスインフラへの大規模波及障害へつながる可能性があるのが現状である．「現金は自宅に置くより銀行に預ける方が安心」という例にもあるように，クラウド環境の信頼性が確保されれば，その普及は爆発的に増加するものと思われる．近い将来，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セキュリティ・信頼性に関する課題解決が特に重要となる．

こうした背景から，ビッグデータ時代を支える安心・安全なクラウド環境が必要とされている．しかしながら，そのデータ肥大化に伴い，セキュリティおよび信頼性の問題に多くの企業が悩まされている．こうしたオープンソースソフトウェアに依存したクラウドの信頼性を定量的に評価する手法は未だ提案されておらず，職人的・試行錯誤的に行われているのが現状である．特に，ビッグデータを扱うクラウドにおいては，ひとたび障害が発生すると個人情報の漏洩だけではなく多大な財産の損失を招くものが多く，セキュリティ・信頼性評価に関する技術の確立は急務である．その運用段階においてセキュリティおよび信頼性を定量的に評価することが可能となれば，

- 医療・行政・IT産業・教育機関においてタグ付けされたデータの一元管理による利便性向上や，利用者の移動コストおよびエネルギーの削減
- 調査会社IDC Japanの2013年6月6日「国内オープンソースソフトウェアエコシステム市場予測」によれば，国内におけるオープンソースソフトウェアのエコシステムの市場規模が，2017年にはオープンクラウドを含め1兆962億円規模に到達

- 情報一元化に伴うデータ管理の簡略化およびビッグデータの活用による人類の知的生産活動の活性化

など，様々な面での波及効果が期待され，その影響は非常に大きく計り知れない．

最近は，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．しかしながら，ソフトウェアの設計図にあたるソースコードが世界中に公開されているため，最近のソニー PSN クラッキングによる情報事故のように，悪意のあるサイト攻撃や情報流出の標的になり易く，なかなか導入に踏み切れないのが現状である．

OSS に対する現在の研究動向としては，設計工程や開発手法，セキュリティを対象とした文献はいくつか提案されているが [1–5]，動的解析に基づいた OSS に対する有効な信頼性評価に関する研究はほとんど行われていないのが現状である．また，クラウド環境に対する最近の研究動向としては，ハードウェア，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おり，最近ではモバイルクラウドを対象とした研究もいくつか行われている [6–10]．しかしながら，そのほとんどがハードウェアやサービス形態の事例研究，データストレージ技術などの性能評価に関するものであり，OSS を利用したクラウド基盤ソフトウェアに対する動的解析に基づく信頼性評価に関する研究は行われていないのが現状である．

従来から，フォールトデータに基づいてソフトウェアの信頼性を評価するアプローチが数多く提案されてきた [11]．しかしながら，クラウドコンピューティングにおいては，ネットワークに常時接続された状態で運用されるため，常に，ユーザ，開発者，およびクラッカーからの影響を受けやすいという特徴がある．さらに，リソース管理を行うためのプロビジョニングと呼ばれるプロセスにより，クラウド環境の特性が変化することも特徴的な点として挙げられる．したがって，信頼性評価のためのデータとして，フォールトデータを直接的な要因として考慮するだけでなく，ネットワークトラフィックの変化を間接的な要因として考慮することは，クラウドコンピューティング全体の信頼性を評価するためには重要であると考えられる．

本論文では，クラウド上のビッグデータがソフトウェア信頼性に対して与える間接的な影響を考慮した 3 次元 Wiener 過程に基づく確率微分方程式モデルについて議論する．また，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．さらに，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に対する数値例を示す．

2 3種類のノイズをもつ確率微分方程式モデル

まず，時刻 $t = 0$ で OSS の運用が開始され，任意の時刻 t までの総検出フォールト数 $\{M(t), t \geq 0\}$ 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．

$$\frac{dM(t)}{dt} = b(t)\{R(t) - M(t)\}. \tag{1}$$

ここで，$b(t)(> 0)$ は時刻 t におけるフォールト発見率を，$R(t)$ は要求仕様の変化を考慮した場合における時刻 t での OSS 内に潜在する総フォールト数を示す．

$$R(t) = \alpha e^{-\beta t}. \tag{2}$$

ここで，α は OSS に潜在するフォールト数を，β は要求仕様の変更率を表す．本論文では，OSS の要求仕様は運用時刻 t に伴い指数関数的に増加または減少するものと仮定する [12,13]．また，クラウド OSS の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 (2) は，

$$\frac{dM(t)}{dt} = \{b(t) + \sigma\gamma(t)\}\{R(t) - M(t)\}, \tag{3}$$

となる [14,15]．ここで，σ (>0) は定数パラメータ，$\gamma(t)$ は解過程の Markov 性を保証するために標準化された Gauss 型白色雑音を表す．さらに，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やサービスアプリケーション数の増減，プロビジョニングプロセスによるクラウド環境の特性変化などにより不規則に変動するものと仮定し，以下の 3 種類の Wiener 過程を導入する．式 (3) を，3 次元 Wiener 過程を考慮した以下の Itô 型の確率微分方程式に拡張して考える [16,17]．

$$dM_1(t) = \left\{b_1(t) - \frac{1}{2}\sigma_1^2\right\}\{R_1(t) - M_1(t)\}dt + \sigma_1\{R_1(t) - M_1(t)\}d\omega_1(t), \tag{4}$$

$$dM_2(t) = \left\{b_2(t) - \frac{1}{2}\sigma_2^2\right\}\{R_2(t) - M_2(t)\}dt + \sigma_2\{R_2(t) - M_2(t)\}d\omega_2(t), \tag{5}$$

$$dM_3(t) = \left\{b_3(t) - \frac{1}{2}\sigma_3^2\right\}\{R_3(t) - M_3(t)\}dt + \sigma_3\{R_3(t) - M_3(t)\}d\omega_3(t). \tag{6}$$

ここで，$\omega_i(t)$ は i 番目の Wiener 過程であり，形式的には白色雑音の時間積分 $\int_0^t \gamma_i(s)ds$ で定義されるものである．各白色雑音の独立性を考慮することにより，以下のような確率微分方程式を得ることができる．

$$\begin{aligned} dM(t) &= \left\{b(t) - \frac{1}{2}(\sigma_1^2 + \sigma_2^2 + \sigma_3^2)\right\}\{R(t) - M(t)\}dt \\ &+ \sigma_1\{R(t) - M(t)\}d\omega_1(t) + \sigma_2\{R(t) - M(t)\}d\omega_2(t) + \sigma_3\{R(t) - M(t)\}d\omega_3(t). \end{aligned} \tag{7}$$

本論文では，3 次元 Wiener 過程 $[\omega_1(t), \omega_2(t), \omega_3(t)]$ を以下のように定義する [17]．

$$\widetilde{\omega}(t) = \left(\sigma_1^2 + \sigma_2^2 + \sigma_3^2\right)^{-\frac{1}{2}}\{\sigma_1\omega_1(t) + \sigma_2\omega_2(t) + \sigma_3\omega_3(t)\}. \tag{8}$$

式 (8) で定義された，$\widetilde{\omega}(t)$ の性質は，次の通りである．

$$\Pr[\widetilde{\omega}(0) = 0] = 1, \tag{9}$$

$$\mathrm{E}[\widetilde{\omega}(t)] = 0, \tag{10}$$

$$\mathrm{E}[\widetilde{\omega}(t)\widetilde{\omega}(t')] = \mathrm{Min}[t, t']. \tag{11}$$

式 (7) の確率微分方程式を初期条件 $M(0) = 0$ の下で Itô の公式を用いて変換すると，

$$M(t) = R(t)\left[1 - \exp\left\{-\int_0^t b(s)ds - \sigma_1\omega_1(t) - \sigma_2\omega_2(t) - \sigma_3\omega_3(t)\right\}\right]. \tag{12}$$

となる．

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．

$$\begin{aligned} \int_0^t b(s)ds &\doteq \frac{\frac{dI(t)}{dt}}{a - I(t)} \\ &= \frac{1+c}{1 + c\cdot\exp(-bt)}. \end{aligned} \tag{13}$$

ここで，$I(t)$ は非同次ポアソン過程に基づく習熟 S 字形ソフトウェア信頼度成長モデルの平均値関数を表す [11]．また，b はフォールト 1 個当りのフォールト発見率を，c は $\frac{(1-l)}{l}$ として定義され，l はネットワークトラフィックの変化率の平均値を表すものと仮定する．したがって，任意の時刻 t における発見フォールト数は，

$$M(t) = R(t)\left[1 - \frac{1+c}{1 + c\cdot\exp(-bt)}\cdot\exp\left\{-bt - \sigma_1\omega_1(t) - \sigma_2\omega_2(t) - \sigma_3\omega_3(t)\right\}\right], \tag{14}$$

となる．ここで，クラウドコンピューティングの運用環境を想定し，σ_1 は b に依存し，σ_2 は c に依存するものと仮定する．さらに，σ_3 はビッグデータのデータ更新率に依存するものと仮定する．また，σ_2 はネットワークトラフィックの変動量を表す標準偏差により近似的に表現できるものと仮定する．

3 信頼性評価尺度

3.1 発見フォールト数の期待値と分散

任意の時刻 t における発見フォールト数の期待値 $\mathrm{E}[M(t)]$ および分散 $\mathrm{Var}[M(t)]$ は，ソフトウェア信頼性を評価する上で重要な尺度となる．これらは，Wiener 過程 $\widetilde{\omega}(t)$ の密度関数が，

$$f(\widetilde{\omega}(t)) = \frac{1}{\sqrt{2\pi t}} \exp\left\{-\frac{\widetilde{\omega}(t)^2}{2t}\right\}, \tag{15}$$

であることから，

$$\mathrm{E}[M(t)] = R(t)\left[1 - \frac{1+c}{1+c\cdot\exp(-bt)}\exp\left\{-bt + \frac{\sigma_1^2}{2}t + \frac{\sigma_2^2}{2}t + \frac{\sigma_3^2}{2}t\right\}\right], \tag{16}$$

となる．

同様にして，$M(t)$ の分散も次の式により求めることができる．

$$\mathrm{Var}[M(t)] \quad = \quad \mathrm{E}[\{M(t) - \mathrm{E}[M(t)]\}^2]. \tag{17}$$

3.2 平均ソフトウェア故障発生時間間隔

平均ソフトウェア故障発生時間間隔（mean time between software failures: MTBF）は，ソフトウェア故障の発生頻度を表すのに有益な尺度である．また，MTBF が大きな値を取ることは，それだけフォールトが発見し難くなり，ソフトウェア信頼性が向上したと判断できることになる．運用時刻 t における瞬間 MTBF （instantaneous MTBF: $\mathrm{MTBF_I}$）および累積 MTBF（cumulative MTBF: $\mathrm{MTBF_C}$）は，以下のように導出できる（文献 [16] 参照）．

任意の時刻 t における瞬間的なフォールト発見間隔の平均を意味する瞬間 MTBF は，

$$MTBF_I(t) = \mathrm{E}\left[\frac{1}{dM(t)/dt}\right], \tag{18}$$

により表すことができる．ここでは，計算の簡略化のために，

$$MTBF_I(t) = \frac{1}{\mathrm{E}[dM(t)/dt]}, \tag{19}$$

で近似的に計算する．

さらに，リリース時点から考えたときの発見フォールト 1 個当りに要する発見時間の平均を意味する累積 MTBF は，

$$MTBF_C(t) = \mathrm{E}\left[\frac{t}{M(t)}\right], \tag{20}$$

により表すことができる．ここでも，計算の簡略化のために，

$$MTBF_C(t) = \frac{t}{\mathrm{E}[M(t)]}, \tag{21}$$

で近似的に計算する．よって，累積 MTBF は，

$$MTBF_C(t) = \frac{t}{R(t)\left[1 - \frac{1+c}{1+c\cdot\exp(-bt)}\cdot\exp\left\{-bt + \frac{\sigma_1^2}{2}t + \frac{\sigma_2^2}{2}t + \frac{\sigma_3^2}{2}t\right\}\right]}, \tag{22}$$

となる．

4 パラメータ推定

提案モデルの推移確率分布に含まれているパラメータ α, β, b, および σ_1 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．なお，σ_2 および σ_3 は，事前のビッグデータ解析結果に基づいて得られる既知パラメータと仮定する．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 $M(t)$ の K 次の同時確率分布を

$$P(t_1, n_1; t_2, n_2; \cdots; t_K, n_K) = \Pr[M(t_1) \leq n_1, M(t_2) \leq n_2, \cdots, M(t_K) \leq n_K | M(0) = 0], \tag{23}$$

とし，その同時確率密度を

$$p(t_1, n_1; t_2, n_2; \cdots; t_K, n_K) = \frac{\partial^K P(t_1, n_1; t_2, n_2; \cdots; t_K, n_K)}{\partial_{n_1} \partial_{n_2} \cdots \partial_{n_K}}, \tag{24}$$

とする．

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

$$\lambda = p(t_1, n_1; t_2, n_2; \cdots; t_K, n_K), \tag{25}$$

と表す．さらに，対数尤度関数を Λ とすると，

$$\Lambda = \log \lambda, \tag{26}$$

となり，提案モデルでは，未知パラメータ α, β, b, および σ_1 を同時尤度方程式，

$$\frac{\partial \Lambda}{\partial \alpha} = \frac{\partial \Lambda}{\partial \beta} = \frac{\partial \Lambda}{\partial b} = \frac{\partial \Lambda}{\partial \sigma_1} = 0, \tag{27}$$

の解として得ることができる．

5 最適メンテナンス問題

本論文では，従来の最適リリース問題 [18,19] を応用した最適メンテナンス問題について議論する．特に，ビッグデータを想定したクラウドコンピューティングに対する 3 次元確率微分方程式モデルに基づく総ソフトウェアコストを定式化し，ネットワークトラフィックの変化率およびデータ更新率を考慮した最適メンテナンス時刻の推定法を提案する．運用段階における総コストを定式化するために，以下のパラメータを定義する．

c_1: 単位時間当りの運用コスト $(c_1 > 0)$,

c_2: フォールト 1 個当りの修正コスト $(c_2 > 0)$,

c_3: 運用段階におけるフォールト 1 個当りの保守コスト $(c_3 > c_2)$.

ここで，c_2 はバグトラッキングシステム上に登録されたフォールトを対象とし，c_3 は実際のクラウドの運用環境に起因するフォールトを対象とする．このとき，以下のようなソフトウェア開発コストが得られる．

$$C_1(t) = c_1 t + c_2 M(t). \tag{28}$$

ここで，c_1, c_2, および c_3 は，システムの開発・保守に関わる SE の人数および人件費等から算出される．このとき，保守コストは以下のように定式化できる．但し，$R(t) > M(t)$ と仮定する．

$$C_2(t) = c_3\{R(t) - M(t)\}. \tag{29}$$

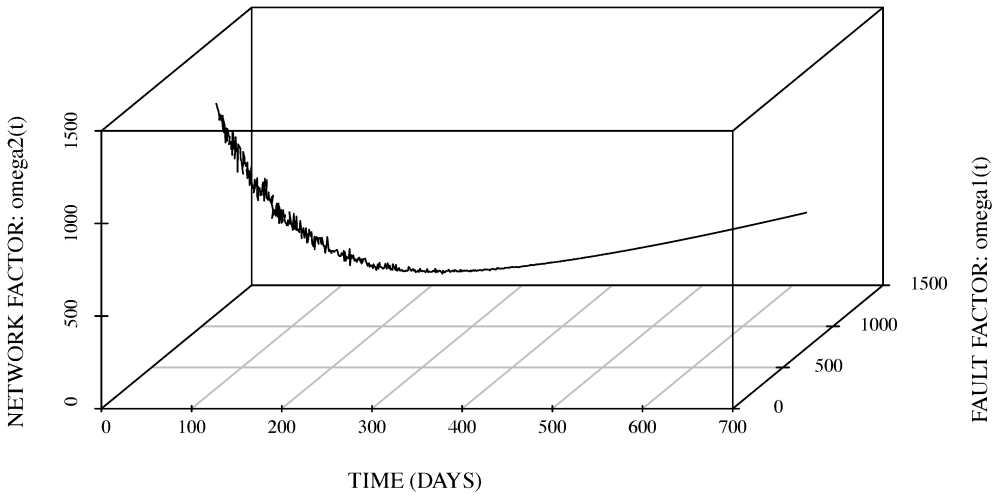

図 1： 推定された総ソフトウェアコストのサンプルパス（Fault × Network）.

したがって，総期待ソフトウェアコストは，式 (28) および式 (29) より，

$$C(t) = C_1(t) + C_2(t), \tag{30}$$

により与えられる．この式 (30) を最小にする時刻 t^* が，最適メンテナンス時刻となる．

6 数値例

実際のクラウド OSS のオープンソースプロジェクトである OpenStack [20] 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．

（Fault × Network）の場合における推定された総ソフトウェアコストを図 1 に示す．また，(Fault × Data）の場合における推定された総ソフトウェアコストを図 2 に示す．さらに，(Network × Data）の場合における推定された総ソフトウェアコストを図 3 に示す．図 1～ 図 3 から，フォールト発見率に影響するノイズは小さいことが分かる．特に，図 3 から，ネットワーク要因およびデータ要因によるノイズの振る舞いが大きいことが確認できる．また，最適メンテナンス時刻は，322.55 日となり，そのときの総ソフトウェアコストは 513.43 であることが確認できる．

以上の結果から，クラウドコンピューティングを支えるクラウドソフトウェアの信頼性は，運用段階において安定していることが分かる．ソフトウェア開発管理者は，クラウドソフトウェアの運用段階において，ネットワークトラフィックの変化率やデータ更新率に関する情報に基づいて，クラウド環境の監視を重点的に行う必要があることが分かる．

7 おわりに

本論文では，クラウド上のビッグデータがソフトウェア信頼性に対して与える間接的な影響を考慮した 3 次元 Wiener 過程に基づく確率微分方程式モデルを提案した．また，提案モデルに基づいて総ソフトウェアコストを定式化し，ネットワークトラフィックの変化率およびデータ更新率を考慮した最適メンテ

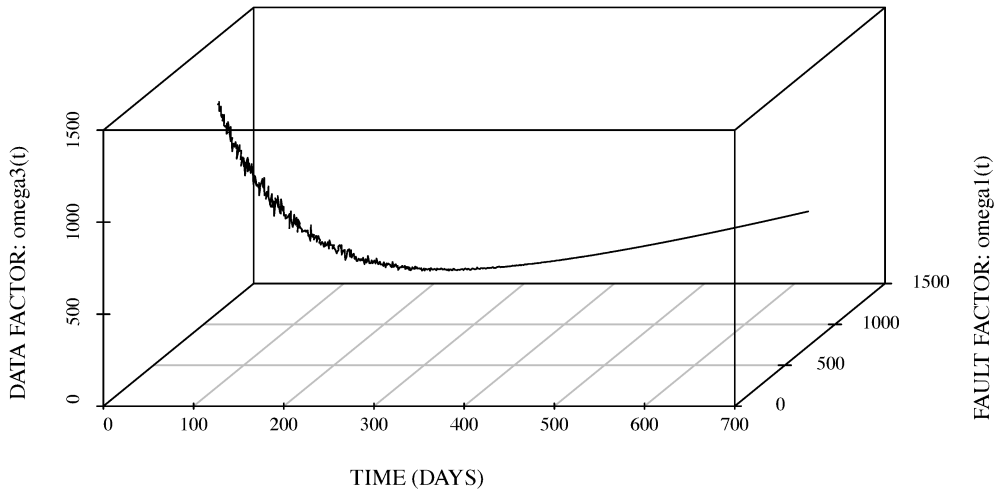

図 2： 推定された総ソフトウェアコストのサンプルパス（Fault × Data）.

ナンス時刻の推定法を提案した．さらに，実際の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することにより数値例を示した．

ビッグデータを扱うクラウドにおいては，ひとたび障害が発生すると個人情報の漏洩だけではなく多大な財産の損失を招くものが多く，セキュリティ・信頼性に対する対策が必要とされている．本論文で提案された手法は，こうしたビッグデータがクラウドコンピューティングへ及ぼす間接的な影響を考慮した信頼性評価法として利用できるものと考える．

謝辞

本研究の一部は，JSPS 科研費基盤研究（C）（課題番号 24500066 および 25350445）の援助を受けたことを付記する．

参考文献

[1] A. MacCormack, J. Rusnak, and C.Y. Baldwin, “Exploring the structure of complex software designs: an empirical study of open source and proprietary code,” *Informs Journal of Management Science*, vol. 52, no. 7, pp. 1015–1030, 2006.

[2] G. Kuk, “Strategic interaction and knowledge sharing in the KDE developer mailing list,” *Informs Journal of Management Science*, vol. 52, no. 7, pp. 1031–1042, 2006.

[3] X. Li, Y.F. Li, M. Xie, and S.H. Ng, “Reliability analysis and optimal version-updating for open source software,” *Journal of Information and Software Technology*, vol. 53, issue 9, pp. 929-936, 2011.

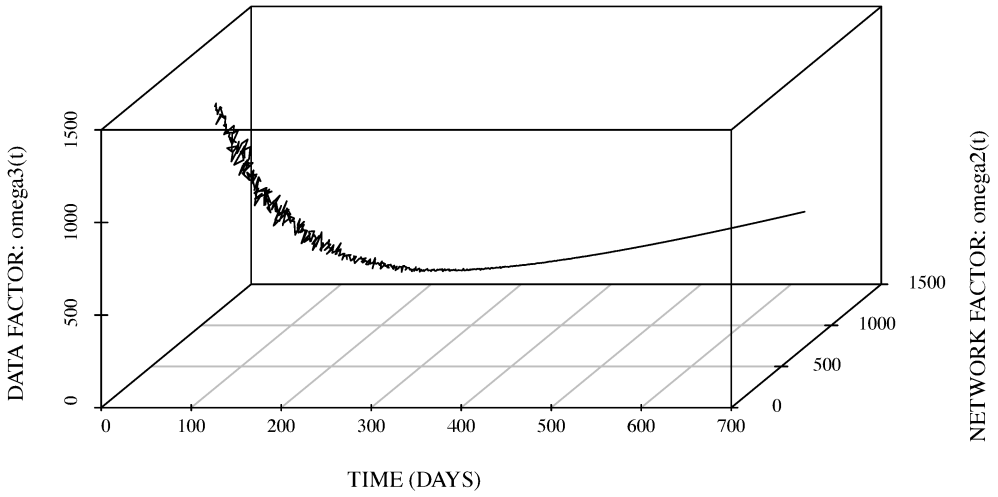

図 3： 推定された総ソフトウェアコストのサンプルパス（Network × Data）.

[4] N. Ullah, M. Morisio, and A. Vetro, "A comparative analysis of software reliability growth models using defects data of closed and open source software," *Proceedings of the 35th IEEE Software Engineering Workshop*, Greece, 2012, pp. 187-192.

[5] D. Cotroneo, M. Grottke, R. Natella, R. Pietrantuono, and K.S. Trivedi, "Fault triggers in open-source software: an experience report," *Proceedings of the 24th IEEE International Symposium on Software Reliability Engineering*, Pasadena, CA, 2013, pp. 178-187.

[6] B. Yang, F. Tan, and Y.S. Dai, "Performance evaluation of cloud service considering fault recovery," *Journal of Supercomputing*, Springer, published online: 23 Feb., 2011.

[7] A. Iosup, S. Ostermann, M.N. Yigitbasi, R. Prodan, T. Fahringer, and D.H.J. Epema, "Performance analysis of cloud computing services for many-tasks scientific computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 6, pp. 931–945, 2011.

[8] J. Park, H.C. Yu, and E.Y. Lee, "Resource allocation techniques based on availability and movement reliability for mobile cloud computing," *in Distributed Computing and Internet Technology*, Lecture Notes in Computer Science, Springer–Verlag, Berlin, vol. 7154, pp. 263-264, 2012.

[9] 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*, Cagliari, Italy, 2013, pp. 655-659.

[10] 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*, Valencia, Spain, 2013, pp. 115-121.

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

[12] T. Fujiwara and S. Yamada, "A testing-domain dependent software reliability growth model for practical application, " *Proceedings of the Second World Congress for Software Quality*, pp. 821–826, 2000.

[13] S. Yamada and T. Fujiwara, "Testing-domain dependent software reliability growth models and their comparisons of goodness-of-fit," *International Journal of Reliability, Quality and Safety Engineering*, vol. 8, no. 3, pp. 205–218, 2001.

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

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

[16] 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.

[17] T. Mikosch, *Elementary Stochastic Calculus, with Finance in View*, Advanced Series on Statistical Science and Applied Probability: vol. 6, World Scientific, Singapore, 1998.

[18] 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.

[19] 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.

[20] The OpenStack project, OpenStack. [Online]. Available: http://www.openstack.org/