数理解析研究所講究録 1457 巻 2005 年 208-215

分散システムにおける最適レジュビネーション方策

岡村寛之，岩本一樹，土肥正
Hiroyuki Okamura, Kazuki Iwamoto and Tadashi Dohi
Department of Information Engineering, Graduate School of Engineering,
Hiroshima University, Japan

1 はじめに

高度情報化社会である現代において，ソフトウェアシステムは今や必要不可欠なものとなってきている．航空管制システムや軍事システム，各種ライフラインシステム等，現代の重要なシステムのコンピュータシステム無くしての運用は不可能に近い．しかし，ソフトウェアシステムは長時間連続で稼動し続けることにより，ソフトウェアエージング (software aging) [1] と呼ばれる経年劣化現象が観測されることが知られている．ソフトウェアエージングの原因としてはメモリリーク，データのフラグメンテーション等が挙げられ，これらによってシステムはパフォーマンスの低下，障害発生確率の上昇といった影響を受ける．ソフトウェアエージングは様々なソフトウェア上で観測されており，一般的なサーバマシン用の OS である UNIX 上のソフトウェアにおいても，長時間の稼動に伴うメモリ使用量の増大が報告されている [2]．このようにエージングを起こしたソフトウェアシステムは，システムの再起動やデータのデフラグメンテーションを行うことにより正常な状態に回復させることができる．これはソフトウェアレジュビネーション (software rejuvenation) [3,4] と呼ばれ，現在盛んに研究が行われるようになってきている．

本稿では，分散処理を行うシステムに対するレジュビネーション方策に関する考察を行う．分散処理とは，複数のコンピュータやプロセッサを利用して，分散して計算処理を行なうことである．例えば，1 台のコンピュータに多数のプロセッサを搭載する場合や，ネットワークを通じて複数のコンピュータが処理を行う場合などがこれにあたる．分散処理は遺伝子解析や気象予測などの大規模な計算でしばしば用いられる方法である．特に本稿では同期あるいは非同期的にメッセージ交換を行いながら複数のプロセスが協調して作業を行う環境を考え，作業の完了時間を最小にするような最適なレジュビネーション方策について考察する．

2 分散処理

分散処理とは，システムを多重化することで処理速度や信頼性を高める技術である．分散処理の具体的な形は様々であるが，ここでは抽象的に処理を行う単位を「プロセス」として扱う．通常，分散処理を行う際には各プロセス間の状態を伝達する必要がある．このやりとりする情報のことを「メッセージ」と呼ぶ．分散あるいは並列処理においてはこのメッセージ伝達に付随する特徴的な問題が存在する．分散処理においては各プロセスが他のプロセスに関する情報を部分的にしか取得できない．つまり，あるプロセスが障害状態であるかどうかを検出するためメッセージの交換を行う必要があるが，比較的ゆるい条件の下では他のプロセスに障害が発生していることを検出できない．これは分散ネットワーク上での合意問題として古くから議論されており，通信路あるいは障害プロセス数に関して仮定を設けることが必要となる．

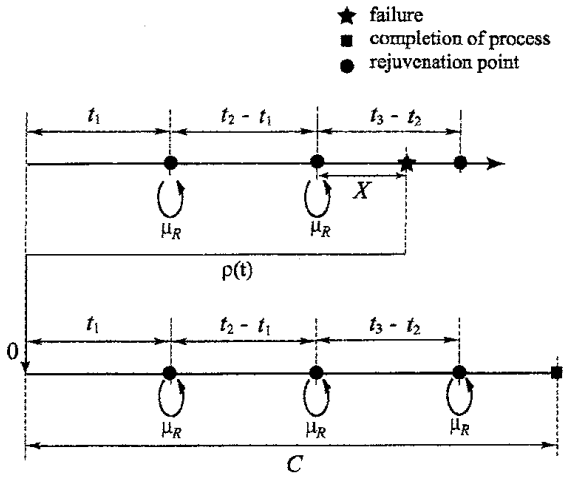

Figure 1: 同期環境におけるレジュビネーションのふるまい.

3 分散処理におけるレジュビネーション

3.1 同期環境における最適レジュビネーションポイント配置

複数プロセスによる協調作業をする環境における最適なレジュビネーション方策について議論する.

いま, 複数のプロセスが互いにメッセージ交換を行いながら, 同期的に単一の作業を行う状況を考える. すなわち, あるプロセスは他のプロセスにメッセージを送信した場合, メッセージ送信したプロセスから応答があるまで待つ. 各プロセスにおける処理は正しいものと仮定する. 障害は通信バッファが不足することによる通信デッドロックによって発生する. つまり, 送信メッセージに対する応答が一定時間帰って来ないことによって検出することが可能である. この種の障害は一過性であり, 作業全体の再実行によって回復する場合が多い. また, 予防策として定期的に通信バッファのメンテナンス (レジュビネーション) を行うことにより障害を避けることができる. ここでは, 作業終了までにどのようなタイミングでレジュビネーションを行うかという問題を取り扱う.

作業は時刻 $t = 0$ から開始され, 全体の作業が終了する時間を C とする. 一般性を失うことなく, 作業完了時間 C は確率分布関数 $\Pr\{C \leq x\} = G(x)$ をもつ確率変数とする. 障害の検出は送信メッセージに対する応答がない場合に行われ, 障害が発生した場合は, ロールバック処理を行い作業の再実行を行う. また, ロールバック処理はそれまで行った処理時間 t の関数であり $\rho(t)$ とする. 障害発生時間 X は確率分布 $\Pr\{X \leq x\} = F(x)$ に従う確率変数とする.

レジュビネーションは特定のスケジュール $\pi = \{t_1, t_2, \ldots\}$ に従って行われ, レジュビネーションを行う時刻をレジュビネーションポイントと呼ぶ. 単一のレジュビネーションは通信バッファのメンテナンスをするものとし, メンテナンスに要する時間 (レジュビネーションオーバーヘッド) は μ_R (> 0) である. また, レジュビネーションを行うことによって, 障害時間分布の年齢が初期化される. モデルの概念図を図 1 に示す.

このような環境において, 作業完了時間の期待値を最小にする最適なレジュビネーションポイント配置を考える. 特にここでは, 以下の方針を与える.

周期的レジュビネーション: レジュビネーション時刻が等間隔. すなわち, ある時間間隔 τ が与えられる

と，レジュビネーションポイントは $\pi=\{\tau,2\tau,3\tau,\ldots\}$ と配置される．

非周期的レジュビネーション： レジュビネーションポイントが有限の数 $N\ (\geq 1)$ まで任意の時刻に配置可能．つまり $\pi=\{t_1,t_2,\ldots,t_N\}$ となる．特に，非周期的レジュビネーションでは，$T\ (>0)$ を処理限界時間として，T 時刻以内に作業が完了しない場合も通常の障害と同様に作業の再実行が行われるものとする．

(i) 周期的レジュビネーション

作業開始から完了のまでの期待時間を $CT(\tau)$ とおくと，障害発生により再実行となるので

$$\begin{aligned} CT(\tau) &= \int_0^\tau\int_0^s\{u+\rho(u)+CT(\tau)\}dF(u)dG(s) \\ &\quad+\sum_{k=1}^{\infty}\int_{k\tau}^{(k+1)\tau}\left(\sum_{i=0}^{k-1}\overline{F}(\tau)^i\int_0^\tau\{i\mu_R+i\tau+u+\rho(i\tau+u)+CT(\tau)\}dF(u)\right. \\ &\quad\left.+\overline{F}(\tau)^k\int_0^{s-k\tau}\{i\mu_R+i\tau+u+\rho(i\tau+u)+CT(\tau)\}dF(u)\right)dG(s). \end{aligned} \tag{1}$$

従って，

$$\begin{aligned} CT(\tau) &= \left\{\int_0^\tau\int_0^s\{u+\rho(u)\}dF(u)dG(s)\right. \\ &\quad+\sum_{k=1}^{\infty}\int_{k\tau}^{(k+1)\tau}\left(\sum_{i=0}^{k-1}\overline{F}(\tau)^i\int_0^\tau\{i\mu_R+i\tau+u+\rho(i\tau+u)\}dF(u)\right. \\ &\quad\left.\left.+\overline{F}(\tau)^k\int_0^{s-k\tau}\{i\mu_R+i\tau+u+\rho(i\tau+u)\}dF(u)\right)dG(s)\right\} \\ &\quad/\left\{1-\sum_{k=0}^{\infty}\overline{F}(\tau)^k\Big(G((k+1)\tau)-G(k\tau)\Big)\right. \\ &\quad\left.+\sum_{k=0}^{\infty}\overline{F}(\tau)^k\int_{k\tau}^{(k+1)\tau}F(s-k\tau)dG(s)\right\}. \end{aligned} \tag{2}$$

ここで，$\overline{F}(\cdot)=1-F(\cdot)$ および $\overline{G}(\cdot)=1-G(\cdot)$ である．問題は $CT(\tau)$ を最小にするレジュビネーション周期 τ^* を見つける問題となる．一般的にはニュートン法などの非線形最適化手法を用いることにより τ^* を算出することができる．

(ii) 非周期的レジュビネーション

いま，$i-1$ 番目のレジュビネーション完了時に次のレジュビネーションポイントの配置を t_i とし，以降最適な配置に従った場合において，作業が完了するまでの期待時間を $CT_i(t_i)$ とする．最適な配置を $\pi^*=\{t_1^*,\ldots,t_N^*\}$ とすると，$t_{i-1}^*\leq t_i\leq t_{i+1}^*$ であることに注意する．最小期待作業時間に関して

$$v_i^*=\min_{t_{i-1}^*\leq t_i\leq t_{i+1}^*}CT_i(t_i),\quad i=1,2,\ldots,N \tag{3}$$

と定義すると，最適性の原理から以下の最適性方程式が成立する．

$$v_i^*=\min_{t_{i-1}^*\leq t_i\leq t_{i+1}^*}CT_i(t_i), \tag{4}$$

$$
\begin{aligned}
CT_i(t_i) &= \int_0^{t_i-t_{i-1}} \int_0^s \{u+\rho(t_{i-1}+u)+v_1^*\} dF(u) dG(s|t_{i-1}) \\
&+ \int_0^{t_i-t_{i-1}} \int_s^{\infty} s dF(u) dG(s|t_{i-1}) \\
&+ \int_{t_i-t_{i-1}}^{\infty} \int_0^{t_i-t_{i-1}} \{u+\rho(t_{i-1}+u)+v_1^*\} dF(u) dG(s|t_{i-1}) \\
&+ \int_{t_i-t_{i-1}}^{\infty} \int_{t_i-t_{i-1}}^{\infty} \{t_i-t_{i-1}+\mu_R+v_{i+1}^*\} dF(u) dG(s|t_{i-1}) \qquad (5) \\
&\quad i=1,\ldots,N.
\end{aligned}
$$

ここで，$t_0=0$, $t_{N+1}=T$ である．また，$CT_i(t_i)$ はより簡潔に

$$
\begin{aligned}
CT_i(t_i) &= \int_0^{t_i-t_{i-1}} \overline{F}(s)\overline{G}(s|t_{i-1})\{1+\rho(t_{i-1}+s)\lambda(s)+v_1^*\lambda(s)\} ds \\
&+(\mu_R+v_i^*)\overline{F}(t_i-t_{i-1})\overline{G}(t_i-t_{i-1}|t_{i-1}) \qquad (6) \\
&\quad i=1,\ldots,N.
\end{aligned}
$$

と書くことができる．このとき，$G(\cdot|\cdot)$ と $\lambda(\cdot)$ はそれぞれ作業時間分布に対する条件付き確率と障害に対する故障率であり，

$$
G(s|t_{i-1}) = 1-\overline{G}(t_{i-1}+s)/\overline{G}(t_{i-1}), \qquad (7)
$$

$$
\lambda(s) = \frac{dF(s)/ds}{\overline{F}(s)} \qquad (8)
$$

と定義される．

式 (4)–(6) で表現される最適性方程式を解くことで期待完了時間を最小にする最適なレジュビネーションポイント配置を導出することができる．いま，式 (5) で表される期待処理時間を t_{i-1}, t_i, v_i^* の関数として $CT_i(t_{i-1},t_i,v_i^*)$ と表現する．さらに期待処理時間から構成される関数

$$
WT_i(t|t_{i-1},t_{i+1},v_{i+2}) = CT_i(t_{i-1},t,CT_{i+1}(t,t_{i+1},v_{i+2})) \qquad (9)
$$

を定義する．このとき，最適性方程式を解くためのアルゴリズムを得る．

Step 1: $k:=0$ とする．

Step 2: 初期レジュビネーションポイント配置 $\pi^{(k)}:=(t_1^{(k)},\ldots,t_N^{(k)})$ と期待完了時間 $w_i^{(k)}$, $i=1,\ldots,N$ を与える．

Step 3: 各 $i=1,\ldots,N$ に対して，$t_i^{(k)}$ と $w_i^{(k)}$ を用いて $WT_i(t|t_{i-1}^{(k)},t_{i+1}^{(k)},w_i^{(k)})$ を最小にする t とその時の最小値を導出し，それぞれ $t_i^{(k+1)}$ と $w_i^{(k+1)}$ とする．つまり，

$$
w_i^{(k+1)} := \min_{t_{i-1}^{(k)} \le t \le t_{i+1}^{(k)}} WT_i(t|t_{i-1}^{(k)},t_{i+1}^{(k)},w_i^{(k)}), \qquad (10)
$$

$$
t_i^{(k+1)} := \operatorname*{argmin}_{t_{i-1}^{(k)} \le t \le t_{i+1}^{(k)}} WT_i(t|t_{i-1}^{(k)},t_{i+1}^{(k)},w_i^{(k)}). \qquad (11)
$$

Step 4: 許容誤差 ϵ に対して，すべの $i=1,\ldots,N$ において $|t_i^{(k+1)}-t_i^{(k)}|<\epsilon$ ならば終了．そうでなければ，$k:=k+1$ として Step 3 へ．

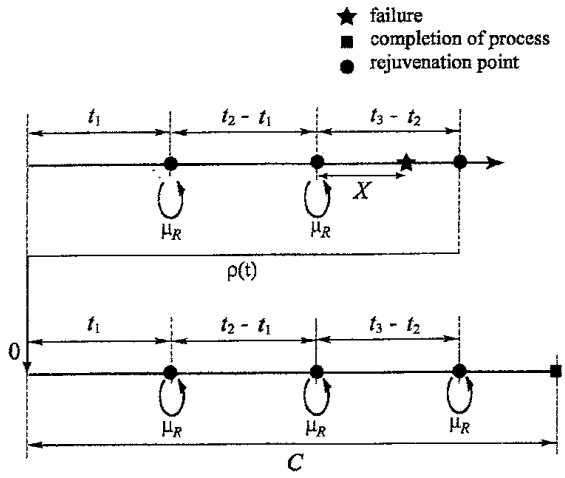

Figure 2: 非同期環境におけるレジュビネーションのふるまい.

3.2 非同期環境における最適レジュビネーションポイント配置

非同期的に複数のプロセスが協調作業を行う環境におけるレジュビネーションについて議論する.

同期的な作業と異なり非同期的な作業環境では送信メッセージに対する応答を必要としないため, タイムアウトによる障害検出を行うことができない. つまり, あるプロセスに障害が発生していても他のプロセスがそれを知ることができないため, 全プロセスが合意のものに再実行を行うためには障害検出作業を行う必要がある. すなわち, 前出した同期プロセスにおけるモデルとは異なり, レジュビネーションポイントにおける作業は, (i) 障害を検出する, (ii) メンテナンスを行うという 2 段階の作業となる.

同期的な作業と同様に, 作業は時刻 $t = 0$ から開始され, 全体の作業が終了する時間を確率分布関数 $G(t)$ を持つ確率変数 C とする. レジュビネーションポイントは特定のスケジュール $\pi = \{t_1, t_2, \ldots\}$ に従って配置される. 単一のレジュビネーションは障害の検出と通信バッファ等のメンテナンスをするものとし, レジュビネーションオーバーヘッドは μ_R (> 0) である. また, レジュビネーションを行うことによって, 障害時間分布の年齢が初期化されるものと仮定する.

同期環境と同様に, 障害発生時間 X は確率分布 $F(x)$ に従う確率変数とする. ただし, 障害はレジュビネーションポイントのみで検出され, 障害が検出された場合は, ロールバック処理を行い作業の再実行を行う. ロールバック処理はそれまで行った処理時間 t の関数であり $\rho(t)$ とする. モデルの概念図を以下に示す.

このような環境において, 作業完了時間の期待値を最小にする最適なレジュビネーションポイント配置を考える. 特にここでは, 周期的レジュビネーションと非周期的レジュビネーションを考える. また, 前出した記号を用いることに注意する.

(i) 周期的レジュビネーション

作業開始から完了のまでの期待時間を $CT(\tau)$ とおくと,

$$
\begin{aligned}
CT(\tau) &= \int_0^{\tau}\int_0^{s}\{\tau + \rho(\tau) + CT(\tau)\}dF(u)dG(s) \\
&\quad + \sum_{k=1}^{\infty}\int_{k\tau}^{(k+1)\tau}\left(\sum_{i=0}^{k-1}\overline{F}(\tau)^i\int_0^{\tau}\{i\mu_R + (i+1)\tau + \rho((i+1)\tau) + CT(\tau)\}dF(u)\right.
\end{aligned}
$$

$$+\overline{F}(\tau)^k\int_0^{s-k\tau}\{k\mu_R+s+\rho(s)+CT(\tau)\}dF(u)\Bigg)dG(s). \tag{12}$$

従って，

$$\begin{aligned}CT(\tau) &= \Bigg\{\int_0^{\tau}\int_0^{s}\{\tau+\rho(\tau)\}dF(u)dG(s)\\ &\quad+\sum_{k=1}^{\infty}\int_{k\tau}^{(k+1)\tau}\Bigg(\sum_{i=0}^{k-1}\overline{F}(\tau)^i\int_0^{\tau}\{i\mu_R+(i+1)\tau+\rho((i+1)\tau)\}dF(u)\\ &\quad+\overline{F}(\tau)^k\int_0^{s-k\tau}\{k\mu_R+s+\rho(s)\}dF(u)\Bigg)dG(s)\Bigg\}\\ &\quad/\Bigg\{1-\sum_{k=0}^{\infty}\overline{F}(\tau)^k\Big(G((k+1)\tau)-G(k\tau)\Big)\\ &\quad+\sum_{k=0}^{\infty}\overline{F}(\tau)^k\int_{k\tau}^{(k+1)\tau}F(s-k\tau)dG(s)\Bigg\}.\end{aligned} \tag{13}$$

問題は $CT(\tau)$ を最小にするレジュビネーション周期 τ^* を見つける問題となる．

(ii) 非周期的レジュビネーション

いま，$i-1$ 番目のレジュビネーション完了時に次のレジュビネーションポイント t_i を配置し，以降最適な配置に従った場合における作業が完了するまでの期待時間を $CT_i(t_i)$ とおく．このとき最適な方策は $\pi^*=\{t_1^*,\ldots,t_N^*\}$ であるため，$t_{i-1}^*\le t_i\le t_{i+1}^*$ であることに注意する．最小期待作業時間に関して

$$v_i^*=\min_{t_{i-1}^*\le t_i\le t_{i+1}^*}CT_i(t_i),\quad i=1,2,\ldots,N \tag{14}$$

と定義すると，最適性の原理から以下の最適性方程式が成立する．

$$v_i^* = \min_{t_{i-1}^*\le t_i\le t_{i+1}^*}CT_i(t_i), \tag{15}$$

$$\begin{aligned}CT_i(t_i) &= \int_0^{t_i-t_{i-1}}\int_0^{s}\{s+\rho(t_{i-1}+s)+v_1^*\}dF(u)dG(s|t_{i-1})\\ &\quad+\int_0^{t_i-t_{i-1}}\int_s^{\infty}sdF(u)dG(s|t_{i-1})\\ &\quad+\int_{t_i-t_{i-1}}^{\infty}\int_0^{t_i-t_{i-1}}\{t_i-t_{i-1}+\rho(t_i)+v_1^*\}dF(u)dG(s|t_{i-1})\\ &\quad+\int_{t_i-t_{i-1}}^{\infty}\int_{t_i-t_{i-1}}^{\infty}\{t_i-t_{i-1}+\mu_R+v_{i+1}^*\}dF(u)dG(s|t_{i-1})\\ &\quad i=1,\ldots,N.\end{aligned} \tag{16}$$

また，$CT_i(t_i)$ はより簡潔に

$$\begin{aligned}CT_i(t_i) &= \int_0^{t_i-t_{i-1}}\overline{F}(s)\overline{G}(s|t_{i-1})\{1+\rho(t_{i-1}+s)\lambda(s)+v_1^*\lambda(s)\}ds\\ &\quad+\int_0^{t_i-t_{i-1}}F(s)\overline{G}(s|t_{i-1})\rho'(t_{i-1}+s)ds\\ &\quad+(\mu_R+v_i^*)\overline{F}(t_i-t_{i-1})\overline{G}(t_i-t_{i-1}|t_{i-1})\\ &\quad i=1,\ldots,N.\end{aligned} \tag{17}$$

と書くことができる．ここで $\rho'(s) = d\rho(s)/ds$ である．式 (15)–(17) で表現される最適性方程式を先に提案したアルゴリズムで解くことによって，期待完了時間を最小にする最適なレジュビネーションポイント配置を導出することができる．

4 数値例

ここでは，同期環境における非周期的レジュビネーションの例を示す．特に障害時間分布と処理時間分布に関する違いが最適レジュビネーションポイント配置に与える影響について検証する．Case 1 から Case 3 では障害時間分布と処理時間分布ともに以下に示すワイブル分布を仮定する．

$$F(t) = 1 - \exp\{-\eta_f t^{m_f}\}, \tag{18}$$

$$G(t) = 1 - \exp\{-\eta_g t^{m_g}\}. \tag{19}$$

ここで，m_f と m_g は形状パラメータと呼ばれ，分布の障害に関する特性を決定するパラメータである．また，η_f と η_g は尺度パラメータと呼ばれ，平均を決定するパラメータである．一般に，ワイブル分布の形状パラメータと平均値が与えられると，尺度パラメータは一意に決定できるため，以下では形状パラメータと平均に関する仮定を与える．

Case 1: $m_f = 2$, $\mathrm{E}[X] = 15$, $m_g = 2$, $\mathrm{E}[C] = 10$.

Case 2: $m_f = 2$, $\mathrm{E}[X] = 15$, $m_g = 5$, $\mathrm{E}[C] = 10$.

Case 3: $m_f = 2$, $\mathrm{E}[X] = 5$, $m_g = 2$, $\mathrm{E}[C] = 10$.

Case 1 を標準的な設定として，Case 2 は処理時間に関するばらつきが少ない状況を想定している．また，Case 3 は Case 1 と比較して，より障害が発生しやすい状況を想定している．これらに加えて分布による違いを検証するために処理時間分布が対数正規分布に従う場合も考慮する．すなわち

Case 4: $m_f = 2$, $\mathrm{E}[X] = 15$, $\sigma = 1.5$, $\mathrm{E}[C] = 10$. ただし，処理時間分布は以下の密度関数を持つ対数正規分布で与えられる．

$$g(t) = \frac{1}{\sqrt{2\pi}\sigma t} \exp\left\{-\frac{(\log(t) - \mu)^2}{2\sigma^2}\right\} \tag{20}$$

また，その他のパラメータは $\mu_R = 0$, $\rho(t) = 0$ とする．つまり，レジュビネーションおよびロールバックのオーバーヘッドはないものとし，すべての場合においてレジュビネーションポイント数が $N = 1, 2, 5, 20$ としたときの最適な配置を算出した．図 3 は最適なレジュビネーションポイント配置，障害密度 $f(t)$, 処理密度 $g(t)$ を示している．図中の点は上から $N = 1, 2, 5, 20$ におけるレジュビネーションポイントを示している．また表 1 は各環境における最小期待処理時間を表している．これらの結果から最適レジュビネーションポイント配置は障害密度よりも処理密度に強く依存していることがわかる．また，Case 3 のように比較的障害が発生しやすい状況においても適度なレジュビネーションポイントを設けることにより，期待処理時間が障害発生がない場合の期待値 $\mathrm{E}[C] = 10$ に近い値をとることができる．

5 今後の課題

本稿では分散処理におけるレジュビネーション方策に関する考察をおこなった．特に同期あるいは非同期なメッセージ交換が行われる分散環境において，処理時間を最小にする最適なレジュビネーションポイント

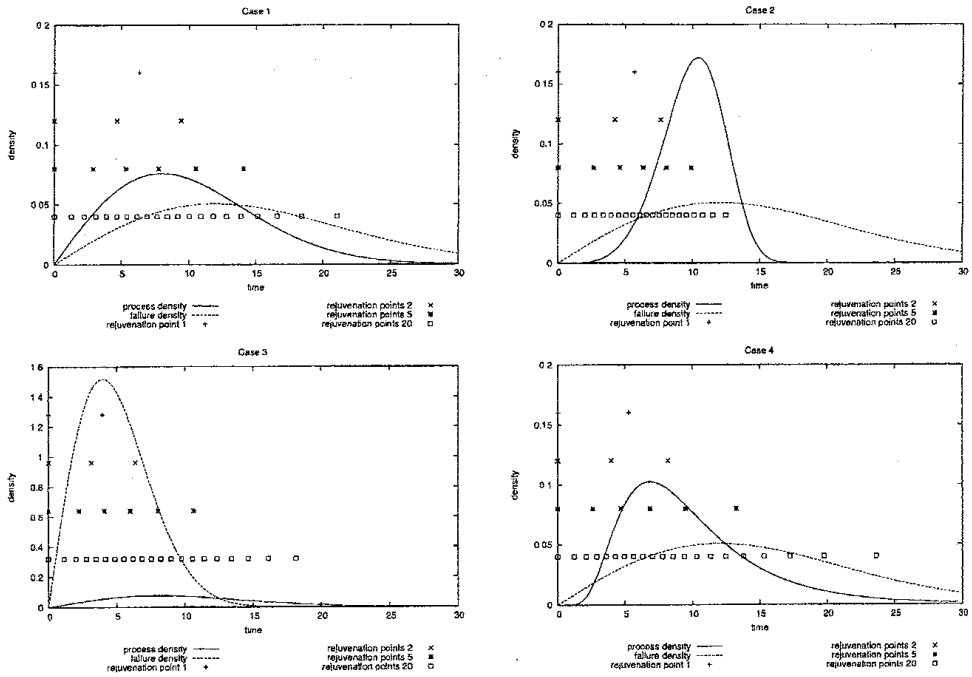

Figure 3: 最適レジュビネーションポイント配置.

Table 1: 最小期待処理時間.

	$N=1$	$N=2$	$N=5$	$N=20$
Case 1	11.12	10.78	10.42	10.13
Case 2	11.14	10.73	10.36	10.10
Case 3	17.64	15.65	13.25	11.08
Case 4	11.03	10.73	10.41	10.13

配置を導出するための手続きを示した．今後は，ベイズ推定によるパラメータ推定を導入することにより，動的にレジュビネーションポイントを配置するオンラインアルゴリズムについて考察する．また，実際の分散処理環境をシミュレートし，提案したアルゴリムが効果的に機能することを検証する．さらに，チェックポイントを同時に考慮した自律的なアルゴリズムについても検討する予定である．

References

[1] D. L. Pasnas, "Software aging," *Proc. 16th Int'l Conf. on Software Eng.*, 279–287 (1994).

[2] V.Castelli and others, "Proactive management of software aging," *IBM J. Res. & Dev.*, 45(2), 311–332 (2001).

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

[4] S. Garg, Y. Huang, C. Kintala and K. S. Trivedi, "Minimizing Completion Time of a Program by Checkpointing and Rejuvenation," *ACM SIGMETRICS 1996*, (1996).