異常プロセスの検知とコンテナ隔離による セルフヒーリングシステムの設計と実装

長谷川 弘享† 菅谷 みどり†

障害発生時において,コストや二次障害を抑えるために自律的に障害処理,回復を行うセルフヒーリング技術が提案されている.セルフヒーリングシステムでは検知と回復を連携して自動で行うことも期待されるが,異常が確認されたプロセスの自律的な隔離とその効果は十分検証されていない.本研究では,こうしたシステムを設計,構築し,その効果を検証する事を目的とする.提案として,暴走している恐れのあるプロセスなどを検知し,シグナルによりコンテナに隔離する.そして,コンテナに制限を加えることでシステムに対して無害な状態にし,それ以外のプロセスには影響を与えないものとした.論文では,こうした動作を自律的に行うセルフヒーリングシステムの設計および実装をする.

DETECTION OF AN UNUSUAL PROCESS, THE DESIGN OF THE SELF-HEALING SYSTEM BY CONTAINER ISOLATION, AND MOUNTING

Hirotaka Hasegawa Midori Sugaya

In obstacle developmental time, in order to suppress cost and a secondary obstacle, the Self-Healing technology of performing obstacle processing and recovery autonomously is proposed. Although it is expected in a Self-Healing system that detection and recovery will cooperate and it will perform automatically, the effect by the autonomous isolation and it of a process by which abnormalities were checked is not verified enough. It is the purpose to design and build such a system in this research, and to verify an effect. By a proposal, an unusual process etc. shall be detected, and it shall isolate to a container by the function called a signal, shall change into a state harmless to a system by adding restriction to a container, and shall not affect the other process. In a paper, the design and mounting of a Self-Healing system which perform such operation autonomously are carried out.

1. はじめに

近年,システム障害が深刻な社会問題となっている[1].障害発生時において,コストや二次障害を抑えるために自律的に障害処理,回復を行うセルフヒーリング技術が提案されている[2].だが、自律的な障害処理,回復処理とその効果は十分検証されていないという問題がある.

本研究では,セルフヒーリングシステムを設計,構築し,その効果を検証する事を目的とする.障害発生時にその原因を検知するための処理,障害から回復する処理によって,システムに対し無害な状態にする.こうした動作を自律的に行うセルフヒーリングシステムを設計,実装し,その評価を行う.

本節の構成は以下の通りである.2 節でセルフヒーリングシステムの課題,3 節で本研究での提案手法,4 節で今後の課題を示す.

†芝浦工業大学 情報工学科

2. セルフヒーリングシステムの課題

セルフヒーリングシステムは生物学的な傷を癒すという概念に由来しており，障害から回復し規範のパフォーマンスレベルを取り戻すという意味で自分自身を"治癒すること"を試みるシステムである．これは IBM の提唱するオートノミックコンピューティング[4]に含まれる 1 つの要素である.

セルフヒーリングシステムの実現に当たって，障害の検知と障害からの回復機能が必要となる．現在利用できる検知システムは数多く提案されているが，回復及びこれらが連携したシステムは十分に提案されていない．既存の商用検知システムとして Zabbix 社が提供する Zabbix[5]，またオープンソース製品でも，Nagios [6], Hinemos [7]等が存在する．しかし， これらは監視し通知することが目的で,主にモニタリングおよび異常検知のフィルタリング機能を提供する．しかし，その後，異常を自動的に排除するなどの対応は行わ

ない．そのため，状況が悪化して結局ハードウエアリセットが必要な状態になってしまうなど，緊急対策が十分に行われない問題がある．本来であれば，一時的な負荷を何らかの形で回避して隔離するなどの対処をしていれば防げたような障害を放置することで，より大きな二次災害などにつながる問題がある．

3. 提案

3.1. 目的と提案

課題で示した問題に対応するため，我々は一時的な負荷などの場合に，緊急的に自律的にシステムの安定化のための対処を行うことで，より深刻な障害を防ぐことを目的とした．実現のために，こうした状態を検知するための検知システム，また，検知した状態に合わせて，負荷の隔離を行うための自律的なセルフヒーリングシステムを提案するものとした．

3.2. システム構成

全体のシステム構成を図 1 に示す．

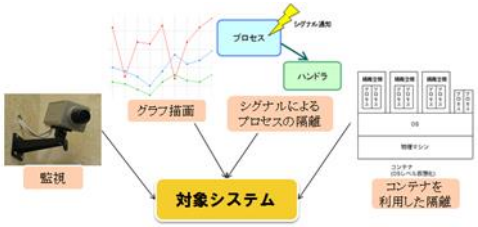

図 1:システム構成

監査対象システムに対して監視を行い,単位時間ごとの CPU 使用率を数値,グラフとして出力.異常プロセスを特定した場合はシグナルを発行して,異常プロセス側のシグナルハンドラによりコンテナへの移動を促す.

3.3. モニタリングシステムの設計と実装

モニタリングシステムでは,端末への CPU 使用率の数値の出力と,グラフ描画ソフト gnuplot によるグラフの出力を行う.また,異常プロセスを検知した際には端末へ警告を出力し,ユーザーへ報告する.

異常なプロセスを検知する方法として,正常な状態での CPU 使用率を記録し,それを基に異常と判別するための基準値を設ける.基準値の設定として単位時間あたりの CPU 使用率を通常平均,加重平均でそれぞれ保存する.通常平均(Ave),加重平均(Wave)とした場合,[Ave]∨[Wave]<[異常プロセスの CPU 使用率]となった際,異常プロセスとして検知する.また,基準値を設ける際の時間周期はユーザーが決定できるものとする.

3.4. プロセス隔離を行うシステムの設計と実装

本システムでは,異常が検知された場合,自動的に異常プロセスを隔離する必要がある.隔離のためには,検知プログラムが検出した異常プロセスに対して通知(シグナル)を発行し,そのシグナルを受け取った異常プロセス側のシグナルハンドラにより,隔離を実現する必要がある.コンテナには,あらかじめ CPU 利用率 40%の制限を設けておく.これらにより,他の実行中のプロセスへの弊害を避けることが見込める.

3.5. 検証,評価

作成したシステムにて,モニタリングの周期や,使用する基準値を変化させ,異常プロセスの検知精度の比較,またコンテナに隔離した際の効果を確認する.

図 2:システム実行時の様子

4. 今後の課題

モニタリングシステムの実装までは終えたが,隔離を行うためのシステムが未完成のため早急に仕上げる.異常判別する際のアルゴリズムがどれほどの精度かの評価を行う.

参考文献

[1] 安本哲之助: 情報システム監査の重要性と課題 2009-10 pp.2-12.
[2] Debanjan Ghosh, Raj Sharmanb,H. Raghav Rao,Shambhu Upadhyaya: Self-healing systems-survey and synthesis 2006-8 pp.1-4.
[3] 寺島 広大: Nagios,Hobbit,ZABBIX,Hinemos を徹底比較 2007-6.
[4] IBM オートノミックコンピューティング http://www-06.ibm.com/software/jp/tivoli/autonomic/
[5] Zabbix 公式サイト http://www.zabbix.com/jp/
[6] Nagios 公式サイト http://www.nagios.org/
[7] Hinemos 公式サイト http://www.hinemos.info/