B-011

リソース制御を用いたメモリ枯渇時での公平な継続動作方式の提案
A proposal of fair continuous execution method using resource control in out-of-memory conditions

小野 優也† 茂田井 寛隆† 水口 武尚†
Yuya Ono Hirotaka Motai Takehisa Mizuguchi

1. はじめに

本報告では，Linux のリソース制御機構である cgroups を応用して組込みシステムの可用性を向上させる機構を提案する．

メモリ資源が少ない組込みシステムでは，あるプロセスの一時的なメモリ大量使用によるメモリ枯渇が発生した場合，当該プロセスや他プロセスの動作に影響する問題がある．従来は，メモリ枯渇を回避するために，事前にプロセス毎のメモリ領域を確保していた．しかし，プロセスの機能やメモリ使用特性が不明な場合に上記方式は困難である．そこで本報告では，メモリ枯渇発生時にプロセス毎にリソース制限を設定することで，プロセスを公平に継続動作できる機構を提案する．

2. 背景

2.1 課題

一般的に，組込みシステムはメモリ資源が少ない．近年，ネットワークを介した保守サービスやデータ分析が必要となるなど，組込みシステムにウェブサーバ機能や機器データ収集機能等を搭載するケースが増えてきた．これらサービスでは，運用後の機器構成変更や，サービス更新などの要因で，想定以上のリソース負荷が発生する可能性がある．例えば，ウェブサーバは，コネクション毎にプロセスを生成し，サービスを提供する．そのため，あるコネクションのプロセスが，メモリを一時的に大量使用するとメモリ枯渇が発生し，他コネクションのプロセスが停止してしまう問題がある．

従来，動作プロセスを事前に把握し，プロセス毎に必要なメモリ領域を確保し，メモリ枯渇を回避していた．しかし，プロセスが動的に生成される場合，プロセスのメモリ使用状態やシステム情報を用いて動的に制御する必要があり，全てのプロセスを適切に制御するには，考慮すべき条件数が膨大になるため処理が複雑化する．さらに，運用者側で自由にアプリケーションを変更できる場合，プロセスの事前調査は困難である．

2.2 リソース制御(cgroups)

Linux では，cgroups(control groups)[1]と呼ばれるリソース制御機構が導入されている．cgroups はタスクグループ(プロセス群)に対して CPU 使用率やメモリ使用量などのリソースの上限値を設定できる．タスクグループは階層的に構造化され，グループ毎にリソース使用量を制限できる(図 1).

メモリ使用量が制限値に達した時の動作は，OOM-Killer(Out of Memory-Killer)の有効/無効によってプロセスの挙動が異なる．OOM-Killer は，システムのメモリ不足時に特定プロセスを強制的に終了させる機構である．OOM-Killer が有効の場合は，制限値に達したグループに属するいずれかのプロセスが kill される．そのため，メモリを大量使用するプロセス以外のプロセスが kill される恐れがある．無効の場合は，制限値に達したグループに属する全てのプロセスが停止するため，メモリ枯渇以降サービスが停止する．したがって，cgroups のリソース制御のみではサービスを継続的に動作させることができない．

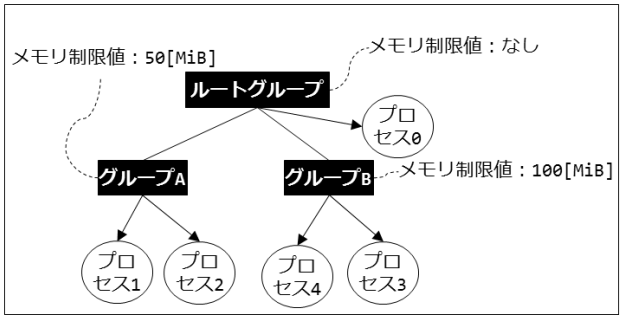

図 1 リソース制御構造

3. 提案手法

本報告では，cgroups を応用し，メモリを一時的に大量使用するプロセスが存在しても，サービスを継続動作させる手法を提案する．

提案手法は，次の四つの処理から成る．

1. リソース制御グループの作成とリソース初期制限
2. メモリ枯渇状態の検出
3. プロセス毎のリソース制御分割と制限値緩和
4. メモリ枯渇からの復帰処理

以降では，これらの処理について説明する．

3.1 リソース制御グループの作成とリソース初期制限

提案手法を実行する監視制御プロセスは，メモリ枯渇時でも動作できるようにリソース制限対象外で動作する．リソース状態の監視対象プロセス 1～nは，監視対象グループに属す．はじめに，監視対象グループに制限値$limit$を設定する．制限値$limit$は，メモリ枯渇時に動作を継続するためのマージンx%を確保しておく．例えば，図 2 のように，メモリ総量$total$の相対値$(1-x/100)$としておく．

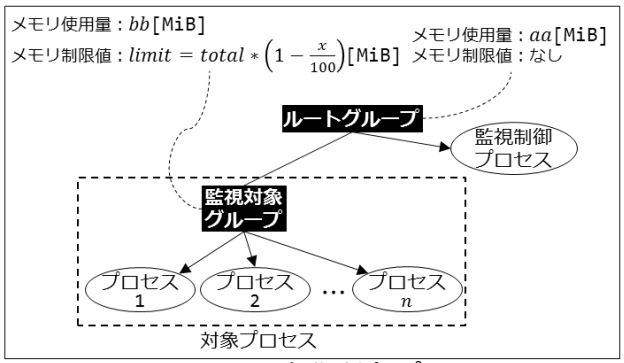

図 2 リソース制御対象プロセス

† 三菱電機株式会社， Mitsubishi Electric Corporation

3.2 メモリ枯渇状態を検出

プロセスのメモリの総使用量が制限値$limit$に達したら，全てのプロセスが停止する．この時点で，いずれかのプロセスがメモリを一時的に大量使用したことによるメモリ枯渇が発生しており，正常なプロセスを含む全てのプロセスが停止していることが分かる．監視制御プロセスは，監視対象プロセスのプロセス状態が停止になっているかを定期的に監視し，メモリ枯渇を検出する．メモリ枯渇を検出したら，次節で述べるプロセス毎のリソース制御を実施する．

3.3 プロセス毎のリソース制御分割と制限値緩和

3.2 節にて，メモリ枯渇を検出したら，停止した各プロセスそれぞれのメモリ使用量($b1$~bn)と同一の制限値($l1$~ln)を設定したタスクグループを作成する(グループ 1～n)．そして，作成したタスクグループにプロセスを移行(図 3)し，個別にメモリ制限する．これにより，各プロセスは他プロセスと独立したリソース制限となる．

その後，それぞれのグループの制限値を緩和し，メモリ余剰を増やすことで，停止プロセスの動作を再開させる．3.2 節での例を用いて，図 3 のグループnの制限値を緩和する場合，制限値$ln = ln/(1 - x/100)$とする．

制限値を緩和する順序によって，プロセスが正常動作できない可能性があるため，プロセス間で，依存関係がある場合，依存元のプロセスから優先して緩和する．例えば，プロセスの起動時刻の昇順で緩和する．

3.4 メモリ枯渇からの復帰処理

この処理では，メモリの一時的な大量使用が終了し，正常状態に復帰できるまで各プロセスのメモリ使用量を監視する．全プロセスのメモリ使用量の合計が，正常時の制限値$limit$以下であれば，全プロセスを監視対象グループに戻すことで，正常状態に復帰させる．一方で，緩和後に再びプロセスが停止していたら，そのプロセスが一時的にメモリを大量使用していると推定されるため，当該プロセスを再起動させる．

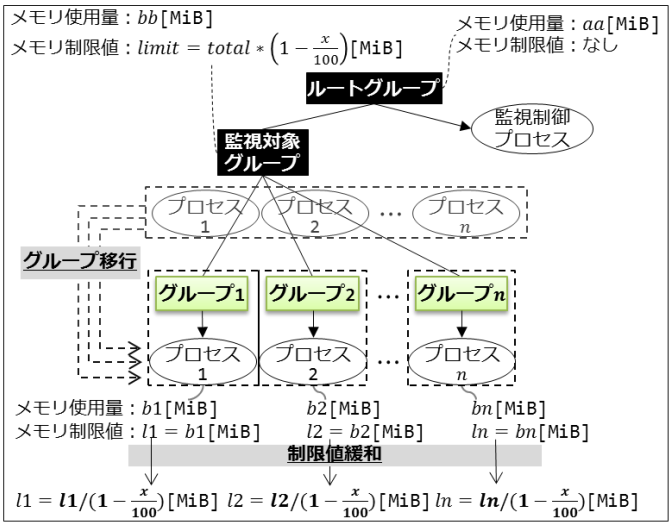

図 3 リソース制御分割と制限値緩和

4. 提案手法の効果

本章では，提案手法の効果を机上評価した．表 1 にシステムのメモリ設定と想定するプロセスの挙動を示す．正常プロセスはメモリを一時的に大量使用しない安定したプロセスであり，異常プロセスは一時的にメモリを大量使用するプロセスである．

表 1 想定システム構成とプロセス挙動

項目		値
メモリ総量		500[MiB]
マージン x		10[%]
正常プロセス		
	数	5
	メモリ使用特性	固定的に 50[MiB]確保
異常プロセス		
	数	1
	メモリ使用特性	固定的に 50[MiB]確保し，ある時点 $t[s]$ ～ $t+90[s]$ の期間に +2[MiB/s]のメモリを確保

上記の設定で，提案手法を適用すると，監視対象グループの制限値$limit$は 450[MiB]である．サービスを開始すると，$t+75[s]$で異常プロセスは 198[MiB]確保済みであり，200[MiB]目を要求する．この時点で，総メモリ使用量が 450[MiB](=$limit$)となり，全てのプロセスが停止する．監視制御プロセスはそれを検知し，プロセス毎のリソース制限へ移行する．その後，制限値を緩和し，正常プロセスはそれぞれ 50[MiB]から 55.6 [MiB]，異常プロセスは 198[MiB]から 220[MiB]に緩和される．正常プロセスは，この時点で動作を再開し，異常プロセスは$t+75[s]$で要求した 2[MiB]のメモリを確保し，さらに，メモリ確保を継続する．異常プロセスは，$t+85[s]$で 220[MiB]のメモリ確保を要求し，再度停止するため，監視制御プロセスは，異常プロセスを再起動させる．これにより，異常プロセスのメモリが解放され，全プロセスのメモリ使用量が$limit$以下になる．

したがって，上記設定において，異常プロセスによりメモリ枯渇が発生しプロセスが停止しても，正常プロセスの動作を即座に再開させ，異常プロセスを再起動させるため，公平な動作継続を実現できる．

5. おわりに

本報告では，組込みシステムにおいて，あるプロセスのメモリの一時的な大量使用によりメモリ枯渇が発生しても，正常なプロセスを公平に継続動作させる機構を提案した．

上記機構により，事前にプロセスのメモリ使用特性を把握できない場合においても，正常なサービスの停止を回避できるため，可用性向上が期待できる．今後は，提案方式を試作し，有効性を確認する．

参考文献

[1] Cgroups, linux documentation, 2018, https://www.kernel.org/doc/Documentation/cgroup-v1/cgroups.txt.