リソース消費量に着目したLinux Kernelの分析

雫石 卓耶　松原 克弥

汎用性の高い OSS ライブラリや OS の普及にともなって，それらを組み合わせるシステム開発手法が一般的になっている．これにより，開発効率が大幅に向上するという利点がある一方，必要のない機能もシステムに組み込まれてしまうという課題がある．そのため，汎用的なソフトウェアから，必要な機能だけを効率的に取り出す最適化技術が求められている．

本研究では，組み込み機器での活用も拡がっている Linux を対象として，その最適化を支援する手法を提案する．Linux の Kernel Config 機構を用いることで，システムに必要のない機能を無効にできる．しかし，その設定項目が約 10,000 個もあるため，最適な設定を手動で見つけることは困難である．本提案手法では，Linux の各機能が，システムのメモリ使用量と電力消費量に与える影響を自動的に計測した．これにより，約 10,000 個の設定項目のうち，最適化に有効と思われるもののみに着目できる．

Reusable libraries and rich functional OSes have been greatly improving system development in terms of time-to-market and product quality. On the other hand, such the general-purpose software must have code and data which could be useless for the target system. Therefore, an optimization mechanism to efficiently extract only the necessary and sufficient functions from the whole software code must be important.

This study aims to support to optimize Linux, which is the most used OS in various systems, including embedded devices. Especially, embedded devices may require it because of limited computing resources and costs. The Linux Kernel Config allows you to remove unnecessary functions explicitly in system building. Nevertheless, it must hard to specify an optimal configuration setting manually for each target system because there are about 10,000 selectable items in the Linux kernel config. We propose a method to automatically estimate the influence on the memory usage and power consumption of each kernel config item in the target system. And an experimental result shows possible to automatically specify items that could effectively shrink resource usage for a target system.

1　はじめに

汎用的なソフトウェアから，必要な機能だけを取り出す技術の必要性が高まっている．汎用性の高い OSS ライブラリや OS の普及にともなって，それらを組み合わせるシステム開発手法が一般化している．汎用的なソフトウェアを使用することで，様々な機能の設計や実装を省略できるため，開発効率が大幅に向上するという利点がある．一方，汎用的なソフトウェアは，様々なユースケースに対応できるよう設計されているため，特定のユースケースでは必要のない機能も含まれている．そのため，汎用的なソフトウェアを使用することで，不要な機能もシステムに組み込まれてしまうという課題がある．不要な機能は，システムのリソースを浪費してしまう．したがって，汎用的なソフトウェアから，必要十分な機能を取り出す必要がある．このことは，ソフトウェアの Tinification と呼ばれる．

Tinification が求められているソフトウェアの 1 つに，Linux がある．デバイスドライバやネットワークスタックなどの，豊富なソフトウェア資産を持つ Linux は，組み込み用途の OS としても広く活用されている [5]．一方，システムにとって不要な機能が，メ

A Method of Analysing the Linux Kernel Focusing Resource Consumption

Takuya Shizukuishi, Katsuya Matsubara, 公立はこだて未来大学, Future University Hakodate.

モリ使用量や電力消費量を増大させてしまうという問題がある．一般的に，組み込み機器には，メモリ使用量や電力消費量などの資源に制限がある．そのため，Linux の Tinification が求められる．Linux には，その機能を取捨選択するための，Kernel Config という機構がある．そのため，この設定を調整することで，システムのリソース消費量を削減できる．

しかし，Kernel Config には，手動での調整が難しいという問題がある．Kernel Config を調整することで，不要な機能を無効化できるが，次の 3 つの理由により容易ではない．1 つ目の理由は，各設定項目の必要性を判断することが難しいということである．各設定項目がシステムにとって必要な場合，その設定項目を無効化してしまうと．システムが正常に動作しなくなってしまう．各設定項目の必要性を判断するためには，Linux Kernel やユーザープロセスに関する深い知識が必要である．そのため，各設定項目の必要性は．容易に判断できない．2 つ目の理由は，各設定項目がリソース消費量に与える影響を判断しにくいということである．設定項目の中には，無効にすることで，リソース消費量を増加させてしまうものも存在する場合がある．そのため，設定項目を無効化することで，メモリ使用量や電力消費量が，減少するのか，増加するのか，または変化しないのかを判断する必要がある．これは，設定項目を一見しただけでは判断できない．3 つ目の理由は，Kernel Config の設定項目が，約 10,000 項目もあるということである †1．必要性の判断や，リソース消費量に与える影響の確認を，全ての設定項目に対して，手動で行うことは困難である．

本研究では，各設定項目の必要性とリソース消費量への影響を自動計測することで，Tinification 作業を支援することを目的とする．この計測結果により，約 10,000 個の設定項目の中から，リソース消費量の削減に効果的な項目を優先的に着目して，Tinification 作業を効率的に進められる．この目的を達成するためには，2 つの手法を確立する必要がある．1 つは，設定項目の必要性を評価する手法である．この手法に関しては，著者らが 2018 年に行った研究で提案と評価を行った [3]．もう 1 つは，各設定項目がリソース消費量へ与える影響を評価する手法である．著者らによる以前の研究 [3] では，計測値の分散による影響を排除できていないという課題があった．そのため，計測結果の精度が低く，リソース消費量への影響が明確ではなかった．また，解析可能なリソース消費量が，メモリ使用量だけであるという課題もあった．

本稿では，Kernel Config の設定項目ごとに，メモリ使用量と電力消費量に与える影響を評価する手法を提案する．各設定項目を無効にした Linux Kernel を実際に起動し，ユーザープロセスを動作させることで，リソース消費量への影響を計測する．計測を複数回行い，統計的検定手法を用いることで，リソース消費量への影響が有意なものかを判定する．これにより，リソース消費量の分散による影響を最小限に抑える．また，解析可能なリソース消費量に，電力消費量を追加することで，新たな視点での Tinification を可能にする．

以降，本稿の構成は，次のようになっている．第 2 章では，Linux における Kernel Configuration システムの概念と仕組みについて述べる．第 3 章では，本稿で提案する手法の詳細について述べる．第 4 章では，提案手法の具体的な実装方法について述べる．第 5 章では，組み込み用途の Linux ディストリビューションである Tiny Core Linux の，Raspberry Pi 向けパッケージである piCore を用いて，本手法の有用性を検証するための評価結果を示す．第 6 章では，本研究と関連する研究についてまとめ，本研究との類似点や相違点について述べる．第 7 章では，本稿の結論と今後の課題を述べる．

2 Linux における Kernel Configuration システム

Kernel Config は，Linux Kernel のコンパイル時に，その機能を取捨選択するための機構である．Linux のソースコードには，Kernel Config を設定するための，nconfig などの様々なツールが含まれている．nconfig の実行例を図 1 に示す．図 1 の各行は，設定項目またはメニューを表している．メニューは, 設定項目の集まりである．メニューを展開することで，更

†1 Linux Kernel v4.9.22 における設定項目のうち，設定画面に現れ得る（prompt 属性を持つ）項目数

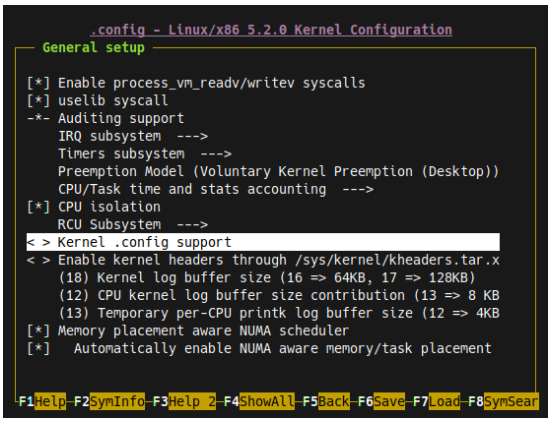

図 1　nconfig の実行例

表 1　Config 項目の型と Linux Kernel v4.9.22 における項目数

型	設定できる値	項目数
bool	y(有効) か n(無効)	5237
tristate	y(有効)，m(カーネルモジュールとして有効)，n(無効) のいずれか	7436
hex	16 進数	39
int	整数	228
string	文字列	42
合計		12982

に詳細な設定項目の値を変更できる．

設定項目には，型と値がある．Kernel Config における型と，それに設定可能な値，および，Linux Kernel v5.2.0 における各項目数を表 1 に示す．Kernel Config における設定項目のほとんどが，bool か tristate 型である．例として，図 1 の「uselib syscall」や「Kernel .config support」が挙げられる．「uselib syscall」の行に，「*」記号が表示されている．これは，その設定項目に，「y（有効）」が設定されていることを示している．一方，「Kernel .config support」の行には，「*」記号が表示されていない．これは，その設定項目に「n（無効）」が設定されていることを示している．

設定項目には，一意な名前がある．また，プロンプト，依存関係，逆依存関係を持つ場合がある．設定項目は，Kernel ソースファイルの各所にある，Kconfig という名前のファイルによって定義されている．Kconfig ファイルによる，設定項目の定義例を図 2 に示す．図 2 の「SCSI」は，設定項目の名前である．この名前は，全設定項目の中で一意である．「scsi device support」は，プロンプトである．プロンプトは，nconfig などのツールの各行に表示される名前である．なお，設定項目の中には，プロンプトを持たず，nconfig に表示されないものもある．そのような設定項目は，Kernel Config の状態をわかりやすくするためのダミーとして，Kernel ソースコードの内部で使用される．「depends on BLOCK」は，依存関係を表している．この例では，SCSI が BLOCK という設定項目の機能に依存していることを示している．そのため，BLOCK を有効にしなければ，SCSI を有効にできない．「select SG_POOL」は，逆依存関係を表している．逆依存関係を用いることで，他の設定項目を強制的に有効にできる．この例では，SCSI が有効なときに，SG_POOL が強制的に有効になる．そのため，逆依存関係によって，設定項目が「ロック」される場合がある．例えば，図 1 の「Auditing support」は，他の設定項目の逆依存関係によって，値が「y」にロックされている．

```
config SCSI
  tristate "scsi device support"
  depends on BLOCK
  select SG_POOL
  ...
```

図 2　kconfig ファイルによる設定項目の定義例

3　リソース消費量の自動計測手法

本手法では，各設定項目を無効にした Linux Kernel を実際に起動し，ユーザープロセスを動作させる．これにより，各設定項目における，リソース消費量への影響を計測する．また，計測を複数回行い，統計的検定手法を用いることで，リソース消費量への影響が有意なものかを判定する．

3.1 計測対象とする設定項目

本節では，本手法が対象とする設定項目を定義する．本手法が解析対象とする設定項目は，次のすべての条件を満たすものとする．

1. 設定項目の型が，bool か tristate である
   bool や tristate 以外の型の設定項目は，対応する機能を無効化するための値が自明ではないため，解析の対象から除外する．
2. Tinification を実施する前の設定（以降，デフォルト設定）の値が，y か m である．
   Tinification 実施前から無効となっている設定項目はシステムの動作に必要ではないことが自明であるため，解析の対象から除外する．
3. 設定項目が，prompt 属性を持っている．
   prompt 属性を持たない設定項目は，Kernel 利用者が設定するものではないため，解析の対象から除外する．
4. 設定項目が，ロックされていない．
   逆依存関係によってロックされている設定項目は，値を変更できないため，解析の対象から除外する．

3.2 各設定項目のリソース消費量の計測方法

本手法では，デフォルト設定の Linux Kernel と，各設定項目を無効にした Linux Kernel を起動，比較することで，各設定項目の必要性と，リソース消費量に与える影響を計測する．

はじめに，デフォルト設定の Linux Kernel のリソース消費量を計測する．そのために，デフォルト設定の Linux Kernel を，コンパイルして起動し，ユーザープロセスを実行する．ユーザープロセスの実行中は，システムの CPU とメモリ，ネットワークの状態を記録する．実行が終了した後に，記録された情報から，テストケース実行直後のメモリ使用量と，テストケース実行中の電力消費量を算出する．

次に，各設定項目のリソース消費量を計測する．そのために，計測対象の設定項目それぞれについて，設定項目を無効にした Linux Kernel をコンパイルし，起動する．そして，デフォルト設定の Linux Kernel の場合と同様に，リソース消費量を算出する．このリソース消費量とデフォルト設定のリソース消費量の差分をとることで，各設定項目を無効にした際のリソース消費量の変化を計算し，これを各設定項目のリソース消費量とする．このとき，起動に失敗した Kernel は，その旨をログファイルに記録する．そして，この後の解析対象から除外する．

最後に，統計的検定手法を用いて，リソース消費量の削減に，統計的に有意な設定項目を判別する．上記の計測結果には分散がある可能性がある．そのため，計測を何度か行い，平均をとることで，結果の精度を向上させる．また，デフォルト設定と各設定項目を無効にした場合で，有意差の検定を行う．これにより，各設定項目を無効にしたことによるリソース消費量の変化が，統計的な有意差であるかを判定する．

3.2.1 テストケース

各 Kernel 上でユーザープロセスを動作させる際に，テストケースを実行することで，本番環境に近い状態で計測を行う．本手法におけるテストケースとは，システムが実際に取りうる動作のことである．例えば，組み込みアプリケーションのテストコードなどが挙げられる．テストケースがとる動作のパターンは，できる限り多く，網羅的であるほうが望ましい．テストケースの動作パターンが少ない場合，計測結果が汎用的なものとなってしまう．多くの動作パターンを持つテストケースを動作させることで，より多くの Kernel の機能を使用して，計測作業を行える．そのため，よりシステムに特化した計測結果が得られる．

3.2.2 依存関係を考慮した設定項目の無効化

設定項目を無効化する際に，その他の設定項目を無効化しなければならない場合がある．本手法では，Kernel の設定項目を無効化することで，リソース消費量の影響を計測する．ここで，ある設定項目を無効化する際，その設定項目に依存している有効な設定項目を無効にする必要がある．依存関係の例を図 3 に示す．図 3 の例 1 では，項目 A と B の間に依存関係がある．ここで，項目 B の機能は，項目 A の機能を用いて動作している．項目 A を無効化する場合，項目 B は動作できなくなるため，項目 B も無効化しなければならない．また，項目 B を無効化する際も同様の処理を行わなければならない．一方，例 2 では，

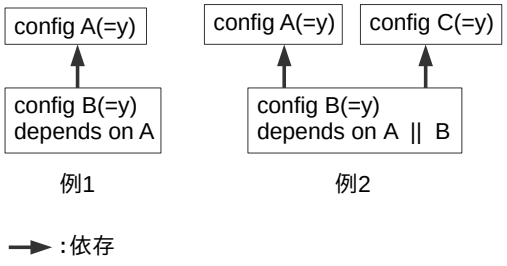

図 3　設定項目の依存関係の例

```
MemTotal:          249188 kB
MemFree:           153084 kB
MemAvailable:      167976 kB
...
```

図 4　meminfo ファイルの出力例

項目 B の機能は，項目 A か C が有効であれば動作する．ここで項目 A を無効化する場合は，項目 B を無効化する必要がない．なぜなら，項目 A を無効化しても，項目 C が有効なため，項目 B の機能は問題なく動作するからである．

4　実装

4.1　メモリ使用量と電力消費量の計測方法

メモリ使用量の計測には，procfs の meminfo ファイルを用いた．meminfo ファイルの出力例を図 4 に示す．MemTotal は，システム全体のメモリ量である．MemAvailable は，利用可能なメモリ量（空きメモリ量）である．メモリ使用量は，$MemTotal - MemAvailable$ として算出した値を計測した．

電力消費量の計測には，Fabian Kaup ら（2014）が提案した電力モデル [1] を用いた．Kaup らは，procfs から得られる情報だけをもとに，機器の電力消費量を算出する，電力モデルを提案した．この電力モデルを用いることで，Raspberry Pi の電力消費量を，9.3 %の誤差で算出できている．Kaup らの手法を用いることで，電気的な計測が不要になるという利点が得られる．

4.2　統計的検定手法

リソース消費量への影響が有意な設定項目を判別するために，対応のある 2 群の t 検定による両側検定を行った．検定は，設定項目ごとの，メモリ使用量と電力消費量ごとに行った．例えば，設定項目 A を無効化することによる，メモリ使用量への影響を検定する場合は，次のような手順を行う．はじめに，デフォルト設定の Kernel と，検定対象の設定項目を無効にした Kernel それぞれのメモリ使用量を計測し，2 つの値を対標本とする．そして，計測の精度を高めるため，同様の計測を何度か行う．次に，帰無仮説を「2 つの Kernel から計測されたメモリ使用量には，差がない」と設定する．最後に，検定手法にしたがって t 値を求め，それが棄却域に入るかを確かめる．帰無仮説が棄却された場合は，「2 つの Kernel から計測されたメモリ使用量には，有意な差がある」と考えられる．この検定作業を，各設定項目の各リソース消費量に対して行うことで，それぞれが有意な差なのかを判断できる．

本実装では，検定を複数回行うため，検定の多重性問題に留意する必要がある．検定の多重性とは，「複数の検定の，どれかが有意であれば成り立つ，全体の仮説」を設定した場合に発生する問題である．このような状況では，「全体の仮説が α エラーとなる確率」が，「それぞれの検定が α エラーとなる確率」よりも大きくなってしまうという問題がある．しかし，本研究では，複数の検定に関する仮説を設定していない．各設定項目の有意性を，それぞれ独立した検定によって評価しているだけである．したがって，本研究における，検定の多重性の問題は無いと判断した．

5　評価

仮のシステムで本手法を実行し，設定項目がリソース消費量に与える影響を，適切に計測できるかを評価する．今回想定したシステムは，テストケースとして，lmbench によるベンチマークを実行するものである．lmbench は，汎用的な UNIX システムの機能を網羅的に実行し，その性能を評価するためのソフトウェアである．そのため，特定の組み込み機器に特化したものではない．しかし本評価では，あえて汎用的

表 2　想定したシステムの構成

ハードウェア構成	Raspberry Pi
Linux	
ディストリビューション	piCore v9.0.3
Linux Kernel	v4.9.22
テストケース	lmbench（OS サブセット）

表 3　評価に用いた QEMU の設定

アーキテクチャ	arm
マシン	versatilepb
CPU	arm1176
メモリサイズ	256MB
その他のオプション	nographic, no-reboot

なベンチマークソフトウェアをテストケースとして用いた．これにより，本手法がどのような Linux の機能に有効なのかを網羅的に評価する．

5.1　評価条件

今回想定したシステムの構成を表 2 に示す．システムは，Raspberry Pi 3 B に piCore v9.0.3 を導入し，テストケースとして lmbench を実行するものとした．なお，lmbench の実行するベンチマークを，OS サブセットに限定した．これにより，OS に関係の無いベンチマークを実行しないことで，解析時間を短縮した．

計測時間を短縮するため，並列実行された仮想環境上で計測を行った．仮想化ハイパーバイザとして，QEMU v2.11.1 を用いた．評価に用いた QEMU の設定を表 3 に示す．QEMU マシンとして versatilepb を用いているのは，QEMU の Raspberry Pi 3 サポートが完全でないため，Raspberry Pi と最も構成の近いもので代用したからである．また，piCore を versatilepb で動作させるために，Kernel Config を手動で調整した．

以上の条件でシステムを起動し，各設定項目のリソース消費量を計測した．計測作業を行ったマシンの

表 4　計測作業を行ったマシンの環境

OS	Ubuntu 18.04.2 LTS
CPU	Intel(R) Xeon(R) CPU E5-2697 v3 @ 2.60GHz 48 コア vCPU
メモリ	110GB vRAM

環境を表 4 に示す．消費電力を算出する際の電力モデルには，Fabian Kaup ら（2014）の Raspberry Pi のモデル [1] を用いた．この電力モデルが想定している構成と，今回用いた QEMU の構成は異なるため，モデルの互換性に留意する必要がある．電力消費量の計測に用いるのは，CPU 使用率とネットワーク通信量の値である．今回のテストケースではネットワーク通信を行っていないため，CPU の挙動にのみ留意すれば良い．今回 QEMU で用いた CPU アーキテクチャは，Raspberry Pi のものと同じため，モデルには互換性があると判断した．各設定項目の計測作業は，10 回行い，平均を計算した．その後，統計的検定手法によって，各設定項目のリソース消費量の変化を検定した．検定の有意水準は $a = 0.05$ とした．t 分布の自由度は 9 である．そのため，$t > 2.2622$ のとき，リソース消費量の変化が有意だと判断した．

5.2　評価結果

本手法による，リソース消費量の計測結果の概要を表 5 に示す．各行は，各分類に該当する設定項目の数を示す．字下げが行われている分類名は，その 1 つ前の分類に属すことを示す．メモリ使用量，電力消費量ともに，それらを統計的に有意に増加・減少させる設定項目があるという結果となった．また，解析には 33 時間を要した．

メモリ使用量への影響が有意であると判定された設定項目の上位 20 件を表 6 に示す．1 行目はデフォルト設定の Kernel による測定結果である．2 行目以降は，メモリ使用量への影響が有意な設定項目である．2 列目の内容のうち，左に書かれている値は，デフォルト設定の値との差を表している．つまり，各設定項目を無効にすることで，メモリ使用量がどのように変化したかを表している．例えば，TMPFS の行は，

表 5　リソース消費量の計測結果の概要

分類	設定項目数
解析対象の設定項目	1137
├起動に失敗した	35
└起動に成功した	1102
メモリ使用量の計測	1102
├減少した	360
│ └有意差あり ($p < 0.05$)	116
├増加した	734
│ └有意差あり ($p < 0.05$)	19
└変化なし	8
電力消費量の計測	1102
├減少した	742
│ └有意差あり ($p < 0.05$)	18
└増加した	360
└有意差あり ($p < 0.05$)	3

その設定項目を無効化することで，メモリ使用量がデフォルトのものから 47760.80KB 減少したことを示している．2 列目の内容のうち，括弧の中に書かれているものは，各設定項目とデフォルト設定の差を検定した結果の t 値である．また，電力消費量への影響が有意であると判断された設定項目を表 7 に示す．表の形式の説明は，表 6 と同様である．

5.3　考察

表 5 において，メモリ使用量が増加した設定項目は，減少したものよりも多かった．設定項目を無効にすることで，メモリ使用量が増加するのは，一見，予想と反する結果であるようにも思える．しかし，メモリ使用量が有意に減少した項目と，有意に増加した項目を比較すると，有意に減少した項目のほうが多いとわかる．そのため，前述した結果は，メモリ使用量への影響が有意ではない項目によるものであるとわかる．計測値の誤差により，偶然このような結果が得られたと考えることができる．

表 6 において，システムで必要とされていない Kernel の機能が多く現れているとわかる．例えば，CFG80211 や WLAN などの無線 LAN 関係の項目，SND や SOUND などのオーディオ関係の項目，BTRFS_FS などの使用していないファイルシステムの項目などがある．これらの機能は，本評価のシステムでは使用していない．そのため，本手法によって，システムに必要のない設定項目を判定できたと考える．

また，表 6 において，無効にすることでメモリ使用量が約 47000KB も削減される設定項目があることがわかる．TMPFS や SHMEM を無効にすることで，デフォルト設定の半分以上のメモリ使用量を削減可能であるという結果が出ている．メモリ使用量が大幅に減少した原因を調査するため，デフォルト

表 6　メモリ使用量への影響が有意であると判定された設定項目

（t 値の絶対値で降順に整列した場合の上位 20 件）

設定項目名	設定項目の無効化によるメモリ使用量の変化 (KB) と t 値
デフォルト設定	81971.20
TMPFS	-47760.80(-1358.987)
SHMEM	-47744.40(-1052.689)
SQUASHFS	-20688.40(-740.983)
MISC_FILESYSTEMS	-21559.60(-732.651)
PROC_SYSCTL	-20836.00(-699.420)
SQUASHFS_ZLIB	-20378.00(-602.324)
BLK_DEV_LOOP	-21622.80(-484.750)
FUTEX	-20817.20(-483.668)
BLK_DEV	-23236.80(-391.937)
CFG80211	-3908.80(-99.245)
WLAN	-2543.60(-98.686)
FTRACE	-4548.00(-98.149)
NETDEVICES	-4179.60(-88.384)
NETFILTER	-2819.20(-76.589)
MODVERSIONS	-2138.80(-57.093)
MAC80211	-2193.60(-43.816)
SOUND	-1319.60(-38.918)
SND	-1345.20(-36.194)
BTRFS_FS	-1430.80(-35.643)
MD	-1191.20(-34.501)

表 7　電力消費量への影響が有意であると判定された設定項目

（t 値の絶対値の降順）

設定項目名	設定項目の無効化による電力消費量の変化（KB）と t 値
デフォルト設定	669.09
VT	+164.36(7.676)
NLS_CODEPAGE_1250	+19.22(5.709)
SCHED_AUTOGROUP	-39.19(-4.924)
FTRACE	-39.25(-4.779)
BRIDGE_EBT_AMONG	-28.87(-3.061)
SND_SOC_WM8731	-22.67(-3.005)
IP_VS_LC	-18.24(-2.962)
FB_TFT_UPD161704	-22.16(-2.754)
MEMCG	-28.34(-2.753)
LEDS_GPIO	-41.27(-2.533)
KEYBOARD_GPIO	-49.44(-2.532)
IP_PNP	-12.30(-2.528)
DYNAMIC_FTRACE	+25.39(2.501)
CIFS_SMB2	-20.14(-2.387)
CAN_BCM	-30.80(-2.356)
INET6_XFRM_MODE_TRANSPORT	-34.83(-2.338)
NLS_CODEPAGE_857	-43.33(-2.306)
NETFILTER_XT_TARGET_NOTRACK	-19.11(-2.304)
W1_SLAVE_DS2431	-16.07(-2.303)
BATTERY_DS2760	-19.75(-2.293)
TASK_IO_ACCOUNTING	-21.25(-2.263)

設定の Kernel と，TMPFS を無効化した Kernel における「/proc/meminfo」ファイルを比較した．その結果を表 8 に示す．表 8 を見ると，Shmem の容量が大きく減少し，MemAvailable が増加している．ここで，デフォルト設定の Kernel における Shmem の容量の正体は，initramfs であると考えられる．なぜなら，Shmem の実体は tmpfs であり，initramfs は，通常 tmpfs を用いてマウントされるためである．initramfs の容量を計測するため，デフォルト設定の

表 8　TMPFS を無効化した際の「/proc/meminfo」ファイルの変化

meminfo の項目	TMPFS 無効化前（KB）	TMPFS 無効化後（KB）	無効化前後の差
MemAvailable	167884	215608	+47724
Shmem	44912	0	-44912

```
$ df
Filesystem  ... Used  ... Mounted on
rootfs      ... 43.9M ... /
```

図 5　デフォルト設定の Kernel 上で df コマンドを実行した結果

Kernel 上で df を実行した結果を図 5 に示す．rootfs の使用容量が，initramfs の容量である．その容量は，減少した Shmem の容量とおおよそ一致する．したがって，TMPFS を無効化することで，initramfs の容量が MemAvailable に含まれてしまうことが，メモリ使用量が大幅に減少した要因である．initramfs の容量が MemAvailable に含まれてしまう原因については，さらなる調査が必要である．また，これらの設定項目が無効化された場合でも，initramfs 自体はメモリ上に存在しているため，この計測結果は誤りである．そのため，メモリ使用量の計測方法を見直す必要がある．

さらに，表 6 において，SQUASHFS などの設定項目を無効化することで，メモリ使用量が 20000KB も減少することがわかる．この原因として，piCore がパッケージ管理に squashfs を用いていることが考えられる．本評価で用いたシステムには，lmbench のコンパイル等に用いた piCore のパッケージが含まれる．これらは，システムの起動時に，squashfs を用いてメモリ上にマウントされることで，利用可能になる．しかし，SQUASHFS が無効となることで，マウントに失敗するため，メモリ使用量が減少したと思われる．これらのパッケージは，テストケースの実行時には用いないため，システムの動作にとって不要である．そのため，SQUASHFS などの設定項目は，

無効にするべき項目である．

表 7 を見ると，VT という設定項目を無効にすることで，電力消費量が有意に増加していることがわかる．この設定項目は，疑似端末機能（tty など）を有効にする設定項目である．この設定項目を無効にした状態の，システムの挙動を確認したところ，getty がログインに何度も失敗していることがわかった．getty は，疑似端末を制御するためのプログラムである．デフォルトの piCore では，起動時に getty が tty1 で自動ログインするように設定されている．しかし，VT を無効にすることで，tty そのものの機能が無効になるため，getty がログインに失敗する．getty はログインに失敗すると，再度ログインを試みる挙動をするため，無限にログインの失敗が起こり，CPU 資源などを消費する．その結果，システムの電力消費量が増加したと考える．

なお，本手法の実行には，約 33 時間を要した．この時間が長いかどうかは，Tinification 実行者の置かれる環境に依存するため，一概には判断できない．しかし，本手法は全て自動化されているため，Tinification 実行者への負担は少ないと考える．

6 関連研究

池田（2006）は，Kernel の機能と，Kernel イメージサイズやメモリ使用量の関係を調査した [4]．この研究では，Kernel の各機能のみを有効にした Kernel をコンパイルし，起動して，Kernel イメージサイズやメモリ使用量を計測した．Kernel の各機能ごとのメモリ使用量に着目している点で，本研究との関連が深い．しかし，ユーザープロセスの動作を考慮しないという点が，本研究と異なる．本研究では，ユーザープロセスの挙動も考慮した，Linux Kernel Tinification を目指す．

Anil Kurmus ら（2013）は，アプリケーションを動作させた Linux Kernel を動的解析することで，必要十分な機能を特定し，Linux Kernel Tinification を行う手法を確立した [2]．Kernel 内関数の呼び出しに着目することで，Linux の各機能が使用されているかどうかを判断した．Linux Kernel の不要な機能を特定しているという点が，本研究と類似している．しかし，研究の目的が本研究とは異なる．Kurmus らの研究の目的は，Linux Kernel の攻撃面を排除するために，Linux Kernel Tinificaion を行った．そのため，メモリ使用量や電力消費量といったリソース消費量に関しては考慮していない．本研究では，リソース消費量の削減を目的として，Linux の各機能がリソース消費量に与える影響を調査した．

7 まとめ

本研究では，Kernel Config の設定項目がリソース消費量に与える影響の評価手法を提案した．これにより，多くの設定項目の中から，リソース消費量の削減に効果のあるもののみに着目して，Tinification 作業を行うことができるようになる．本手法では，各設定項目を無効にした Linux Kernel をコンパイルし，起動することで，各設定項目のリソース消費量を計測した．また，統計的検定手法を用いることで，リソース消費量が有意に変化する設定項目を判別した．さらに，組み込み用途の Linux ディストリビューションである piCore 上で lmbench を動作させることで，本手法の有用性を評価した．その結果，リソース消費量を有意に低下させると思われる，いくつかの設定項目を判別できた．

今後，2 つの課題を解決する必要がある．1 つは，テストケースの検討である．本研究の評価で用いたテストケースには，ネットワーク通信やセンシングなどの処理がない．そのため，一般的な IoT 機器の行う動作と相違がある．また，lmbench による負荷は，IoT 機器を通常使用している場合のものとは異なる．より実際のユースケースに近いテストケースを用いることで，本手法の有用性を更に評価する必要がある．もう 1 つの課題は，Tinification の評価項目の追加である．本研究では，メモリ使用量と電力消費量に着目して Linux Kernel の Tinification を行った．しかし，組み込みシステムに求められる特性は，それらだけではない．起動時間の短さや，レイテンシの少なさなどにも着目した，より発展した手法を確立したい．

謝辞 本稿の統計的検定手法に関するアドバイスをしていただいた，公立はこだて未来大学の新美礼彦氏に感謝する．

参 考 文 献

[1] Kaup, F., Gottschling, P., and Hausheer, D.: PowerPi: Measuring and modeling the power consumption of the Raspberry Pi, *39th Annual IEEE Conference on Local Computer Networks*, IEEE, 2014, pp. 236–243.

[2] Kurmus, A., Tartler, R., Dorneanu, D., Heinloth, B., Rothberg, V., Ruprecht, A., Schröder-Preikschat, W., Lohmann, D., and Kapitza, R.: Attack Surface Metrics and Automated Compile-Time OS Kernel Tailoring., *NDSS*, 2013.

[3] 雫石卓耶, 松原克弥, ほか: カーネル利用状況とメモリ使用量に着目した Linux Kernel Config の解析手法, コンピュータシステム・シンポジウム論文集, Vol. 2018(2018), pp. 81–89.

[4] 池田宗広: Linux カーネルサイズ・使用メモリ検証／改善ＰＪ, 2006.

[5] 中島達夫ほか: 組込みシステム用標準 OS としての Linux, 情報処理, Vol. 43, No. 2(2002), pp. 148–153.