監視システムの設定ファイル内でのアラート設定箇所の集約と機能ごとの分割による可読性の向上

平尾 真斗[1]　大野 有樹[2]　串田 高幸[1]

概要：監視ソフトウェアは障害の検知を迅速に行うために用いられるアラートを出す際の監視対象と監視条件を設定ファイル内に記述する．監視ソフトウェアの設定ファイルはサーバの増設やアラートを出す際の監視条件の見直しにより修正される．課題は監視対象と監視条件の記述箇所が増加することによる可読性の低下である．可読性が低下する要因として一行あたりの文字数，行数の増加，機能の混在化をあげる．本稿では記述箇所を監視対象，監視条件，配置方法に分割しアラートの設定箇所を一箇所に集約することで可読性を向上させる設定ファイルの形式を提案する．実験では提案の設定ファイルと Prometheus のアラートマネージャの設定ファイルの形式に 10 個の監視項目を記述し，可読性の指標である一行あたりの文字数と行数を比較した．行数では提案の設定ファイルの形式が Prometheus のアラートマネージャの設定ファイルの形式に比べて 155 行削減し，約 29%削減できた．一行あたりの文字数では 10 個目の監視項目を記述した際に 116 文字削減した．その後，提案の設定ファイルの形式の可読性が向上したかを証明するため，修正実験を行った．修正実験では 2 つの設定ファイルの形式にそれぞれ 5 人ずつ記述し修正時間を比較した．それぞれの設定ファイルで修正は 1 回ずつ，合計 10 回行った．比較の結果，修正時間は平均で約 18%削減し，提案の設定ファイルの形式の可読性が高いことが分かった．

1. はじめに

背景

新型コロナウイルスワクチン予約サイトはユーザがワクチンを予約する際に用いられる．予約サイトがダウンするとユーザは予約処理ができなくなるため監視システムを導入し障害の検知を行う[*1]．

監視システムには監視ソフトウェアが用いられる．監視ソフトウェアは障害発生時にシステム管理者にアラートを通知するために用いられる [1,2]．監視ソフトウェアの一例として Prometheus や Nagios がある [3,4]．また，監視ソフトウェアは設定ファイルに記述された対象，条件を元に監視を行い，アラートを通知する．この設定ファイルはサーバの増強や監視条件の見直しにより修正される [5,6][*2]．修正作業により設定ファイル内の記述箇所は増加する．

監視ソフトウェアの設定ファイルを修正をする際に重要な要素に可読性がある [7]．可読性は一般的に文章の読みやすさや理解のしやすさを示す指標である．一方でシステム管理者にとっては設定ファイルやプログラミングコードの読みやすさや理解のしやすさである [8,9]．可読性はこれらに影響を与える点から重要な要素となる [10]．

課題

課題は設定ファイル内で監視を行うための対象と条件の記述箇所が増加することによる可読性の低下である．例えば監視ソフトウェアである Prometheus のアラートマネージャを用いた際に可読性が低下する例を設定ファイル 1 に示す．設定ファイル 1 は監視を行う対象のメモリ使用率が 60%を超えた際にアラートを出す設定を記述している．1 行目はアラート名を定義している．2 行目と 3 行目は監視を行う対象と条件を設定する箇所である．設定ファイルで可読性を低下させている原因は以下の 3 つである．

(1) **行数の増加**: 監視を行う対象と条件の増加に伴い行数が増える [11]．行数の増加は可読性の低下につながる．

(2) **一行の文字数の増加**: 一行の文字数が長くなり可読性に影響を与える [12]．設定ファイル 1 の行番号 3 番の文字数は 91 文字となっており一行の文字数は 70 文字

[1] 東京工科大学コンピュータサイエンス学部
〒 192–0982 東京都八王子市片倉町 1404-1
[2] 東京工科大学大学院バイオ・情報メディア研究科コンピュータサイエンス専攻
〒 192–0982 東京都八王子市片倉町 1404-1

[*1] URL:https://www.scsk.jp/news/2021/press/product/20210712.html，閲覧日 2023 年 10 月 16 日

[*2] URL:https://itpfdoc.hitachi.co.jp/manuals/3021/30213L0200/IMDS0616.HTM，閲覧日 2023 年 10 月 16 日

設定ファイル 1: アラートマネージャの設定ファイルの例

```
1 - alert : HighCPUUsage
2   expr : |
3     (sum(container_memory_usage_bytes
      {namespace="test"})) / (3 * 1024 *
      1024 * 1024) *  100 > 60
```

を超えると可読性が低下する [13][*3][*4].

(3) **機能の混在化**: 監視を行う対象と条件の記述箇所が一行で記述されていると可読性が低下する [14]．例えば行番号 3 番の箇所は監視を行う対象を示す namespace=test と条件を示す (sum(container_memory_usage_bytes が同じ箇所にある．

これらの 3 点から本稿で定義する可読性の低下は，行数の増加，一行あたりの文字数の増加，機能の混在化とする．また一行あたりの文字数を最大限に削減する方法として一行に記述する箇所を細かく分割することがあげられる．しかしそうすると行数が増える．行数を最大限削減する方法としては一行で全ての記述箇所を記述する方法があげられるがそうすると一行あたりの平均文字数が増加する．可読性の低下は新たに監視を行う対象と条件を修正する際に修正する箇所を探す時間の増加や誤った設定を追加することにつながる [15, 16].

各章の概要

第 2 章では，関連研究について解説する．3 章では, 記述箇所を監視対象，監視条件，配置方法に分割しアラートの設定箇所を一箇所に集約することで可読性を向上させる設定ファイルの形式を提案する．4 章では, 実装した 2 つのソフトウェアと実験を行う方法について解説する．5 章では, 可読性の指標である行数，一行あたりの文字数，修正時間，修正量を比較する．6 章では, 提案の設定ファイルの形式における可読性の影響について議論し改善案を提示する．最後に 7 章で本稿のまとめを行う.

2. 関連研究

PromQL は，Prometheus の独自のクエリ言語であり、監視を行う対象と条件を記述する際に用いられる [17]．監視を行う条件組み合わせることができる．一方で監視を行う条件を組み合わせると一行あたりの文字数が多くなり可読性の低下につながる点が改善点となる．

設定ファイルのフォーマットである XM-ADL を提案した研究がある [18]．設定項目を囲んで表現するタグ付けを用いている．タグ付けの手法は設定項目を囲んで表現するため一行あたりの文字数が多くなるデメリットがある．

設定ファイルのフォーマットの一つに JSON がある．[19]．このフォーマットでは設定項目を Key と Value のペアで記述し {} で囲む．対応する Key と Value のペアが増えるとそれに応じて {} の量が増えてしまい結果的に可読性の低下につながる．

設定ファイルのフォーマットとして用いられる YAML は辞書型の値を Key と Value のペアで表現しインデントごとに設定項目を分ける [20]．YAML の形式では設定項目が増えると対応するインデントが深くなり可読性の低下につながる．

Nagios は OSS の監視ソフトウェアであり Config 形式の設定ファイルを用いて監視を行う条件を記述する [4][*5]．監視を行う条件と対象を一行の同じ箇所に記述するため文字数が増え，可読性の低下につながる．

3. 提案方式

本稿では監視システムでアラートの送信先と障害対応時のドキュメント URL の記述箇所を一箇所に集約し，監視対象部，監視条件部，配置方法部の記述箇所を分割することで可読性を向上させるの設定ファイルの形式を提案する．その際に設定ファイルを読み取り，監視を行う監視ソフトウェアは独自で作成した．提案の設定ファイルの形式は，この監視ソフトウェアのみで使用可能な独自形式で作成した．また，前提として監視方法はモジュール化されているものとする．監視対象部は監視モジュールに引数を記述する箇所である．監視条件部は監視モジュール内に異常条件を記述する箇所である．配置方法部は監視対象部，監視条件部で設定した監視モジュールをどのように配置するかを記述する箇所である．監視対象部，監視条件部，配置方法部に分割することを機能ごとの分割とする．アラートの送信先と障害対応時のドキュメント URL の記述箇所は，監視を行う対象と条件ごとに記述すると行数が増える．そのため記述箇所を集約し行数を削減する.

機能ごとの分割

機能ごとに記述箇所を分割し一行あたりの文字数を削減する提案の設定ファイルの形式を図 1 に示す．提案の設定ファイルの形式は機能ごとに分かれておりそれぞれ監視対

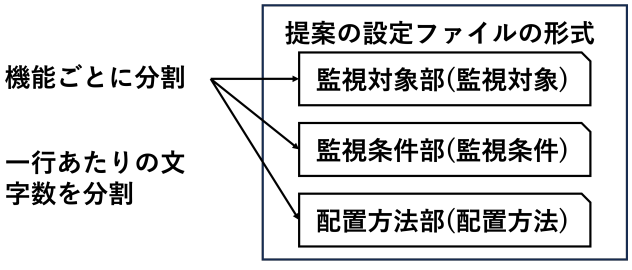

図 1: 記述箇所を分割する提案の設定ファイルの形式

[*3] URL:http://webtypography.net/2.1.2
[*4] URL:https://baymard.com/blog/line-length-readability
[*5] URL:https://monami.sourceforge.net/tutorial/ar01s07.html

象部，監視条件部，配置方法部である．この機能を一行で記述する場合一行あたりの文字数が増加する．そのため提案の設定ファイルの形式ではそれらの機能を一行で記述するのではなく分割して記述することで一行あたりの文字数を削減する．

アラート設定箇所の集約

アラートの送信先と障害対応時のドキュメント URL の記述箇所は監視を行う対象と条件を設定するたびに記述すると行数が増える．そのため提案の設定ファイルの形式では alert モジュールを用意しアラートの送信先と障害対応時のドキュメント URL を一箇所に集約する．一箇所に集約することで行数を削減する．

構文ルール

提案の設定ファイルの構文には予約語がある．予約語の例として設定ファイル 2 の中で記述されている module, module-rule，condition，loop がある．本提案の課題として一行あたりの文字数をあげた．予約語を一文字で表現することや略語を使うことで一行あたりの文字数を最大限削減することができる．しかし一文字で表現することや略語は可読性を下げる原因となる [21][*6]．そのため一文字で表現することや略語は使わず記述する．

設定ファイルの構文と構文に入れるコマンドを A から A-3，記述例を B に示す．

A: 設定ファイルの形式の構文

提案の設定ファイル形式の構文を設定ファイル 2 に示す．1 行目の module は監視対象部を示す記述箇所である．2 行目は監視モジュール名を記述する箇所である．例えば Web サーバに対して HTTP リクエストを送る外形監視では Web という監視モジュールを用いる．同じモジュールを複数回使用する場合は監視モジュール名＋数字のように記述する．3 行目の監視モジュール名の引数は監視モジュールに対応する引数を選択する．例えば Web モジュールでは引数値として監視対象の URL を記述する．4 行目の alert はアラートを送るためのモジュールである．5 行目はアラートを送る先の URL を記述する箇所である．6 行目はアラートのメッセージに送る障害対応時のドキュメントが記載されている URL を記述する箇所である．4 行目から 6 行目でアラートの設定をまとめることで行数を削減する．

8 行目は監視条件部を示す記述箇所である．9 行目は 2 行目で記述した監視モジュール名を記述する．10 行目は監視の取得回数と取得間隔を記述する箇所である．loop の 中に取得回数と取得間隔を記述する．11 行目は監視モジュールが異常を示す条件を記述する箇所である．監視モジュール名は 2 行目で記述した監視モジュールと同じである．監視内容は監視モジュールに対応する監視内容を記述する．例えば Web モジュールを使用する際に HTTP リクエストのステータスコードが見たい場合は.status を使用する．またこの異常条件を記述するために不等号，数値か論理値を使用する．13 行目の condition は配置方法部を示す記述箇所である．条件式と監視モジュール名，不等号，論理値で監視モジュールをどのように配置するかを記述する．15 行目は 4 行目から 6 行目で記述した alert モジュールに対してアラートメッセージを記述する箇所である．

設定ファイル 2: 提案の設定ファイルの形式の構文

```
1  module: #監視対象部
2    監視モジュール名:
3      監視モジュールの引数: 引数値
4    alert:
5      url: アラートの送り先のURL
6      runbookURL: 障害対応時のドキュメント
7
8  module-rule: #監視条件部
9    監視モジュール名:
10     loop(取得回数, 取得間隔)
11       監視モジュール名.監視内容 不等号
         数値OR論理値:
12
13 condition: #配置方法部
14   条件式 監視モジュール名 不等号 論理値:
15     alert("アラートメッセージ")
```

[*6] URL:https://www.nottingham.ac.uk/brand/styleguide/?id=42a72988-5cdc-4795-931c-cfd6e2c0d2d5, 閲覧日 2023:10/30

A-1: 監視モジュール

監視モジュールと引数値の一覧を表 1 に示す．alive は監視を行う対象に ICMP の応答確認を行う [22]．引数値として IP アドレスを使用する．web は監視を行う対象が Web サーバの時に使用する．監視対象に対して HTTP リクエストを送り応答確認を行う [23]．dns は監視対象が DNS サーバの時に使用する．監視対象である DNS サーバがドメインを IP アドレスに変換できているかを監視する．引数値としてドメインを指定する．pod-cpu，pod-memory，pod-status，oom-pod はそれぞれ監視対象の CPU，Memory，

表 1: 監視モジュール一覧

監視モジュール	監視機能	引数値
alive	ICMP の監視	IP アドレス
web	HTTP の監視	URL
dns	DNS の監視	ドメイン
pod-cpu	CPU の監視 (Pod)	namespace
pod-memory	Memory の監視 (Pod)	namespace
pod-status	ステータス値の監視 (Pod)	namespace
storage	ストレージの監視	IP アドレス
network	network の監視	IP アドレス
oom-pod	oom の監視 (Pod)	namespace

表 2: 監視内容一覧

監視内容	使用可能な監視モジュール
.percent	pod-cpu，pod-memory，storage，network
.status	alive,web,dns,oom-pod
.time	全監視モジュール
.usage	pod-cpu，pod-memory，storage，network
.readusage	storage
.writeusage	storage
.connection	network

表 3: 監視モジュールを組み合わせるコマンド

コマンド	コマンドの意味
if	条件分岐を示すコマンド (配置方法部で使用)
elseif	条件分岐を示すコマンド (配置方法部で使用)
else	条件分岐を示すコマンド (配置方法部で使用)
and	組み合わせコマンド
or	組み合わせコマンド

ステータス，oom の監視を行う．これらの監視モジュールは Kubernetes を用いた Pod の監視に使用する．storage は監視を行う対象のストレージの監視を行う．引数値として IP アドレスを使用する．network は監視対象の network の監視を行う．引数値として IP アドレスを指定する．

A-2: 監視内容

監視条件部内に記述する監視内容を表 2 に示す．監視内容は監視モジュール名. 監視内容で記述する．

.percent は pod-cpu，pod-memory，storage，network の監視モジュールで使用可能であり使用率を返す．.status は alive，web，dns，oom-pod の監視モジュールで使用可能であり，ステータスを返す.time は全監視モジュールで使用可能であり監視モジュールが監視を行った際の応答時間を返す．.usage は pod-cpu，pod-memory，storage，network の監視モジュールで使用可能であり使用量を返す．.readusage は storage の監視モジュールで使用可能であり書き込み量を返す．.connection は network の監視モジュールで使用可能でありコネクション数を返す．

A-3: 監視モジュールを組み合わせるコマンド

監視モジュールを組み合わせるコマンドを表 3 に示す．

if，elseif，else は条件分岐を示すコマンドであり配置方法部で監視モジュールを使用する際に記述する．and と or は監視モジュールを組み合わせるコマンドである．

B: 記述例

記述例を設定ファイル 3 に示す．設定内容として監視対象である Web サーバに対して HTTP リクエストを送りステータスコードが 500 番以上の場合，異常とする設定をする．1 行目から 6 行目は監視対象部を記述する箇所である．1 行目には監視対象部を示す module を記述する．2 行目には Web サーバに対して HTTP リクエストを送る監視モジュールである web を記述する．3 行目は監視対象である Web サーバの url を記述する．4 行目は alert を出すモジュールを示す．アラートを送る対象と障害対応時のドキュメント URL をそれぞれ 5 行目と 6 行目のように記述する．

設定ファイル 3: 提案の設定ファイルの形式の記述例

```
1  module: #監視対象部
2    web:
3      url: http://example.com
4    alert:
5      url: http://slack.com
6      runbookURL: http://runbook.com
7
8  module-rule: #監視条件部
9    web:
10     loop(3, 3)
11       web.status > 500:
12
13 condition: #配置方法部
14   if web == True:
15     alert("500 Error")
```

8 行目から 11 行目は監視条件部を示す記述箇所である．8 行目には監視条件部を示す module-rule を記述する．10 行目には loop を記述し () 内に監視の取得回数と取得間隔を記述する．11 行目には監視モジュールである web が異常とする条件を記述する．13 行目から 15 行目は配置方法部を示す記述箇所である．14 行目に監視モジュールの配置ルールを記述し，15 行目にアラートのメッセージを記述する．

ユースケース・シナリオ

本ユースケースシナリオでは，新型コロナウイルスワクチン予約サイトの監視を行う際に，設定ファイルの修正を行う場面を想定する．ユースケースシナリオを図 2 に示す．

システム管理者はワクチン予約サイトの運用と管理を行っている．設定ファイルには監視を行う対象と条件を記

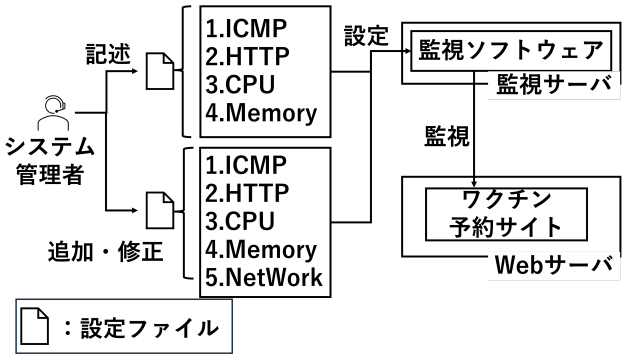

図 2: ユースケースシナリオ

述する．監視サーバ内の監視ソフトウェアは設定ファイルの内容を読み取りワクチン予約サイトを監視する．設定ファイルの監視対象と監視条件はサーバの増設やアラートの条件の見直しにより修正される[*7][*8]. 例えば宮城県大崎市の新型コロナウイルスワクチン予約サイトではアクセス集中が起きたためサーバを増強した[*9]．その際に提案の設定ファイルの形式では監視を行う対象や条件の修正を行う際に可読性が低下しない設定ファイルの形式になるため，修正箇所を探す時間の短縮や誤った設定項目の追加の防止ができる．

4. 実装

提案の設定ファイルの読み取りと監視をするソフトウェアは実験を簡易的に行うために独自で作成した．実装したソフトウェアは 2 つある．実装をしたソフトウェアの概要を図 3 に示す．提案の設定ファイルの内容の読み取りと解析そして監視エージェントの監視情報を読み取り異常判定を行う実行ソフトウェアと実行ソフトウェアに監視情報を送信する監視エージェントである．監視エージェントは用途ごとに分かれている．監視情報は CPU，メモリ，ストレージ，ネットワークの使用量，外形監視，死活監視の状態がある．それぞれ Python3.10.6 で実装した．

実行ソフトウェアは提案の設定ファイルの内容を読み取り単語ごとに分割，辞書型への変換，監視モジュールの選択と異常の判定を行う．単語ごとの分割では提案の設定ファイルの記述箇所を単語ごとに分割する．辞書型への変換では分割された単語を監視対象部，監視条件部，配置方法部ごとに辞書型に変換する．監視モジュールの選択では，提案の設定ファイル内に記述されている監視モジュールを選択する．監視モジュールは異常判定をする．異常判定は監視エージェントから取得した監視情報の値と提案の設定ファイルに記述された異常条件を用いて判定を行う．

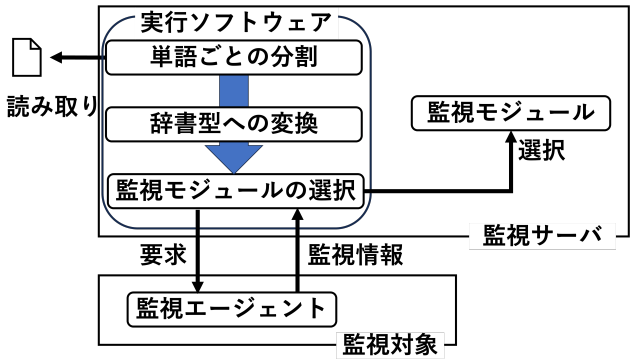

図 3: 実装のソフトウェア

監視エージェントは実行ソフトウェアの要求に対して監視対象の情報を送る．監視対象内に監視エージェントは複数配置する．

5. 評価実験

実験では提案の設定ファイルと Prometeus のアラートマネージャの設定ファイルの形式に監視を行う監視項目を記述する．記述した設定ファイルはそれぞれのソフトウェアに適応させる．提案の設定ファイルは実装した独自の監視ソフトウェアに適応させる．対して既存形式である Prometheus のアラートマネージャの設定ファイルは Prometheus に適応させる．監視対象は新型コロナウイルスワクチン予約サイトを想定する．その際に可読性の指標となる行数，一行あたりの文字数を比較する．その後，新型コロナウイルスワクチン予約サイトで監視対象を追加し監視ソフトウェアの設定ファイルに修正を行うシナリオを用意する．このシナリオに沿って設定ファイルの修正を行う．修正は提案の設定ファイルの形式を記述する 5 人と Prometeus のアラートマネージャの設定ファイルの形式を記述する 5 人が記述を行う．修正はそれぞれの設定ファイルで一回ずつ，合計で 10 回行う．記述後，修正にかかった時間と追加した文字数を比較する．修正にかかる時間は可読性の低下とともに遅くなるため可読性の指標として加えた [15,16]．記述を行う監視項目を図 4 に示す．監視項目は監視を行う項目である[*10]．(1)，(7)，(10) の条件はノードごとの監視項目である．(1) はノードに対する ICMP の監視項目である．(2) は WordPress に対して HTTP の監視を行う監視項目である．(3) は NFS サーバに対しての NFS クライアントの接続数の監視を行う監視項目である．(4) は NFS サーバに対しての TCP,UDP のパケット使用量の監視を行う監視項目である．(5) は WordPress と MySQL の OMM キラーの監視項目である．(6) は WordPress と

	監視項目
(1)	ノードに対するICMPの監視
(2)	WordPressに対するHTTP通信の監視
(3)	NFSサーバに対してのパケット量の監視
(4)	NFSサーバに対してのNFSクライアント接続数の監視
(5)	WordPressとMySQLのOOMの監視
(6)	WordPressとMySQLのCPU使用率の監視
(7)	ノードのCPU使用率
(8)	WordPressとMySQLのメモリ使用率の監視
(9)	NFSサーバへの組合せの監視(disk 使用率,read,write)
(10)	ノードのメモリ使用率

図 4: 監視項目

[*7] URL:https://www.fujitsu.com/jp/products/computing/servers/primequest/solution/costdown/，閲覧日 2023 年 11 月 1 日

[*8] URL:https://itpfdoc.hitachi.co.jp/manuals/3021/30213L0200/IMDS0616.HTM，閲覧日 2023 年 11 月 1 日

[*9] URL:https://www.asahi.com/articles/ASP5B7394P5BUNHB009.html，閲覧日 2023 年 11 月 15 日

[*10] URL:https://www.rworks.jp/monitoring/monitoring-column/monitoring-design/25591/, 閲覧日 2023:11/11

MySQL の CPU 使用率の監視を行う監視項目である．(7) はノードの CPU 使用率を監視する監視項目である．(8) は WordPress と MySQL のメモリ使用率の監視を行う監視項目である．(9) は NFS サーバへの組合せの監視項目である．ディスクの使用率，読み込み量，書き込み量の監視を行う監視項目である．(10) はノードのメモリ使用率を監視する監視項目である．監視項目は Prometheus のアラートマネージャの設定ファイル内で監視条件を記述する箇所の文字数が多い順に並べる．

修正シナリオでは新型コロナウイルスワクチン予約サイトでサーバの増設に伴い監視対象を設定ファイル内に追加する．監視対象は Kubernetes クラスタのワーカを 5 台追加し監視項目 (1)，(7)，(10) をそれぞれ追加する．ワクチン予約サイトはユースケースで示した宮城県大崎市の事例を想定した．

実験環境

実験で使用する環境を図 5 に示す．仮想マシンには Ubuntu22.04 をインストールした．仮想マシンの台数は 15 台である．仮想マシンは同じ性能の (vCPU: 3[core]，RAM:3[GB]，SSD:25[GB]) で作成した．また仮想マシン上に K3s による Kubernetes クラスタを用意しマスター 1 台，ワーカ 14 台の構成とした．また修正シナリオで監視対象を追加する際の仮想マシンを新たに 5 台用意する．設定ファイルは提案の設定ファイルと Prometheus のアラートマネージャの設定ファイルの形式である．実行ソフトウェアは監視エージェントに対して監視情報の要求を行い異常判定をする．監視情報は CPU，メモリ，ストレージ，ネットワークの使用量，外形監視，死活監視の状態がある．監視エージェントは実行ソフトウェアに対して監視情報の提供をする．Prometheus と Exporter は監視ソフトウェアであり監視をする対象の WordPress と MySQL，NFS サーバを監視する．新型コロナウイルスワクチン予約サイトを WordPress，MySQL，NFS サーバで実装した．

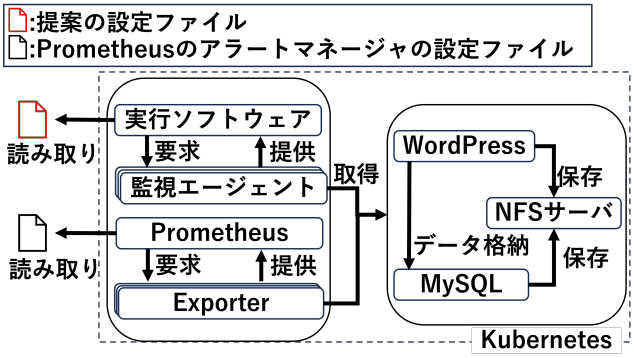

図 5: 実験環境

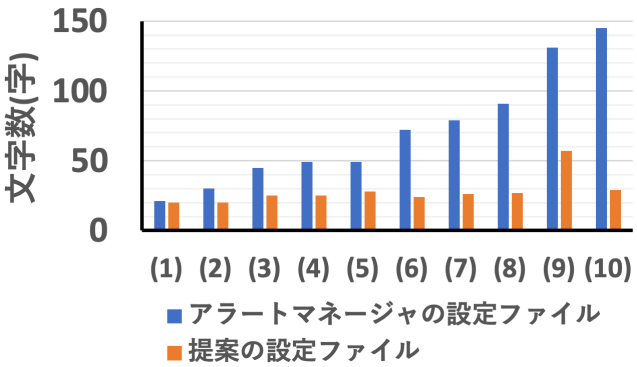

図 6: 各設定ファイルの一行あたりの文字数の変化

実験結果と分析

最初に監視項目を記述した 2 つの設定ファイルの形式を可読性の指標となる一行あたりの文字数，行数，一行あたりの平均文字数を比較する．記述した設定ファイルは提案の設定ファイルと Prometheus のアラートマネージャの形式をする．一行あたりの文字数では監視項目の記述箇所を比較する．提案の設定ファイルは監視項目の記述箇所が 3 つに分割されている．そのため 3 つに分けられた記述箇所の中で最も文字数の多い箇所を選択する．Promtheus のアラートマネージャの設定ファイルでは監視項目の記述箇所を選択する．行数の比較では 2 つの設定ファイルの形式に監視項目 (1) から (10) まで記述した際の行数の変化を比較する．

最初に一行あたりの文字数を比較する．ここでは提案の設定ファイルの形式と Prometheus のアラートマネージャの設定ファイルの形式で監視項目に沿って記述をした．提案の設定ファイルは監視項目の記述箇所が 3 つに分割されているため 3 つに分けられた記述箇所の中で最も文字数の多い箇所を選択する．Prometheus のアラートマネージャの設定ファイルの形式では監視項目を記述する箇所を選択する．図 6 に一行あたりの文字数の変化を示す．

監視項目 (1) では提案の設定ファイルの形式が 20 文字，Prometheus のアラートマネージャの設定ファイルの形式が 21 文字となった．監視項目 (2) では提案の設定ファイルの形式が 20 文字，Prometheus のアラートマネージャの設定ファイルの形式が 30 文字となった．監視項目 (3) では提案の設定ファイルの形式が 25 文字，Prometheus のアラートマネージャの設定ファイルの形式が 45 文字となった．監視項目 (4) では提案の設定ファイルの形式が 25 文字，Prometheus のアラートマネージャの設定ファイルの形式が 49 文字となった．監視項目 (5) では提案の設定ファイルの形式が 28 文字，Prometheus のアラートマネージャの設定ファイルの形式が 49 文字となった．監視項目 (6) では提案の設定ファイルの形式が 24 文字，Prometheus のアラートマネージャの設定ファイルの形式が 72 文字となった．監視項目 (7) では提案の設定ファイルの形式が 26 文字，

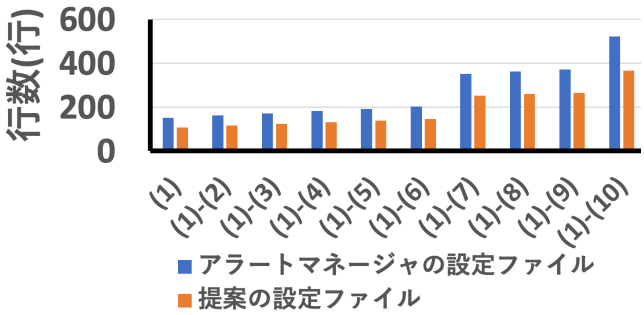

図 7: 各設定ファイルの行数の変化

Prometheus のアラートマネージャの設定ファイルの形式が 79 文字となった．監視項目 (8) では提案の設定ファイルの形式が 27 文字，Prometheus のアラートマネージャの設定ファイルの形式が 91 文字となった．監視項目 (9) では提案の設定ファイルの形式が 57 文字，Prometheus のアラートマネージャの設定ファイルの形式が 131 文字となった．監視項目 (10) では提案の設定ファイルの形式が 29 文字，Prometheus のアラートマネージャの設定ファイルの形式が 145 文字となった．提案の設定ファイルは Prometheus と比較して一行あたりの文字数を平均で 61%削減した．

次に行数の評価である．図 7 に行数の変化を示す．行数では監視項目を (1) から (10) を 1 つずつ追加していった際に行数がどのように変化するかを比較する．監視項目の (1) を追加した場合，提案の設定ファイルの形式が 107 行でアラートマネージャの設定ファイルの形式が 152 行となった．監視項目の (2) を追加した場合，提案の設定ファイルの形式が 117 行でアラートマネージャの設定ファイルの形式が 162 行となった．監視項目の (3) を追加した場合，提案の設定ファイルの形式が 124 行でアラートマネージャの設定ファイルの形式が 172 行となった．監視項目の (4) を追加した場合，提案の設定ファイルの形式が 131 行でアラートマネージャの設定ファイルの形式が 182 行となった．監視項目の (5) を追加した場合，提案の設定ファイルの形式が 138 行でアラートマネージャの設定ファイルの形式が 192 行となった．監視項目の (6) を追加した場合，提案の設定ファイルの形式が 145 行でアラートマネージャの設定ファイルの形式が 202 行となった．監視の (7) を追加した場合，提案の設定ファイルの形式が 252 行でアラートマネージャの設定ファイルの形式が 352 行となった．監視項目の (8) を追加した場合，提案の設定ファイルの形式が 259 行でアラートマネージャの設定ファイルの形式が 362 行となった．監視項目の (9) を追加した場合，提案の設定ファイルの形式が 266 行でアラートマネージャの設定ファイルの形式が 372 行となった．監視項目の (10) を追加した場合，提案の設定ファイルの形式が 367 行でアラートマネージャの設定ファイルの形式が 522 行となった．全ての監視項目を記述した際に最大で 29%行数を削減した．一行あたりの文字数と行数の指標で可読性を高めることができた．

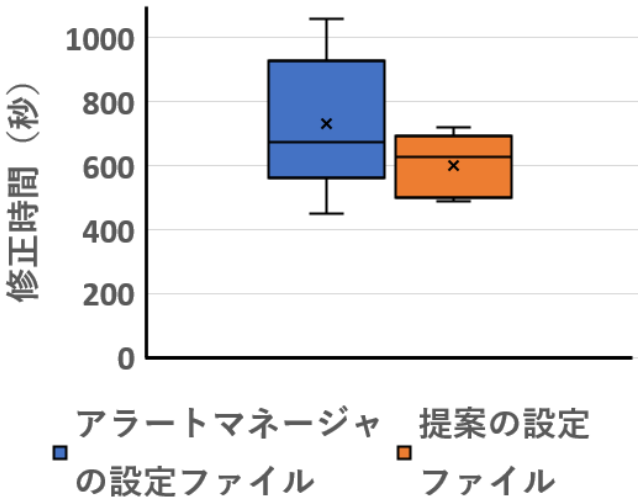

図 8: 修正量の秒数

修正時

設定ファイルの修正時の比較では監視対象である Kubernetes クラスタのワーカを 5 台追加し監視項目 (1)，(7)，(10) をそれぞれ追加する．修正は提案の設定ファイルと Prometheus のアラートマネージャの設定ファイルの形式に行う．修正を行う記述者は提案の設定ファイルと Prometheus のアラートマネージャの設定ファイルの形式でそれぞれ 5 人ずつとした．その際に修正にかかった秒数と修正にかかった追加した文字数を比較する．

修正の際にかかった秒数を図 8 に示す．修正を行なった際の修正時間の中央値は提案の設定ファイルの形式が 626 秒で Prometheus のアラートマネージャの設定ファイルの形式が 673 秒となった．平均値は提案の設定ファイルの形式が 601 秒で Prometheus のアラートマネージャの設定ファイルの形式が 731 秒となった．平均値と中央値の結果から提案の設定ファイルの形式が可読性が高く記述を行う際に修正時間が短くなったと言える．最大値は提案の設定ファイルの形式が 721 秒で Prometheus のアラートマネージャの設定ファイルの形式が 1,060 秒となった．最小値は提案の設定ファイルの形式が 488 秒で Prometheus のアラートマネージャの設定ファイルの形式が 450 秒となった．最大値から最小値の値を引くと提案の設定ファイルの形式が 610 秒で Prometheus のアラートマネージャの設定ファイルの形式が 233 秒となった．この結果から提案の設定ファイルの形式の方が可読性が低下する影響を受けにくいと言える．

次に追加した文字数を評価する．追加した文字数の評価では新たに Kubernetes クラスタ内にワーカ 5 台を追加し追加した際の文字数を比較する．図 9 に追加した文字数を示す．修正する際の監視条件は監視項目 (1) のノードに対する ICMP の監視を一台ずつ追加する．ワーカを 1 台追加した際の文字数は提案の設定ファイルの形式が 99 文字

でアラートマネージャの設定ファイルの形式が 125 文字である．ワーカを 2 台追加した際の文字数は提案の設定ファイルの形式が 198 文字でアラートマネージャの設定ファイルの形式が 250 文字である．ワーカを 3 台追加した際の文字数は提案の設定ファイルの形式が 297 文字でアラートマネージャの設定ファイルの形式が 375 文字である．ワーカを 4 台追加した際の文字数は提案の設定ファイルの形式が 396 文字でアラートマネージャの設定ファイルの形式が 500 文字である．ワーカを 5 台追加した際の文字数は提案の設定ファイルの形式が 495 文字でアラートマネージャの設定ファイルの形式が 625 文字である．文字数では最大で 125 文字削減できた．

6. 議論

可読性を下げる原因の一つに空白行があげられる [24]．本提案の設定ファイルの形式では監視対象，監視条件，配置方法の間の空白行を自由に定めることができる．しかし空白行の改行の仕方は人によって異なる．プログラミング言語である Python では 2 行の空白を開けることが推奨されている[*11]．そのため監視対象，監視条件，配置方法の間の空白行の行数を 2 行で固定化することで可読性の低下を抑えることができる．

修正時の評価の際に提案の設定ファイルを記述する記述者と Prometheus のアラートマネージャの設定ファイルを記述する記述者を用意した．人によって理解力や記述力は異なるため記述時間に差が出る．そのため同じ人が 2 つの設定ファイルの形式に記述することが望ましい．しかし 2 つの設定ファイルを続けて記述する際に記述する間隔が短いと一回目に設定ファイル内に記述した内容を覚えてしまうため 2 回目以降に記述する際に 1 回目に記述した場合よりも早くなる．そのため同じ人で 1 回目に記述した後に 1 ヶ月後に 2 回目の記述をすることで理解した内容をなくした上で公平な記述ができる．人が覚えたことは 1 ヶ月経つと忘れるため 1 ヶ月とした[*12]．

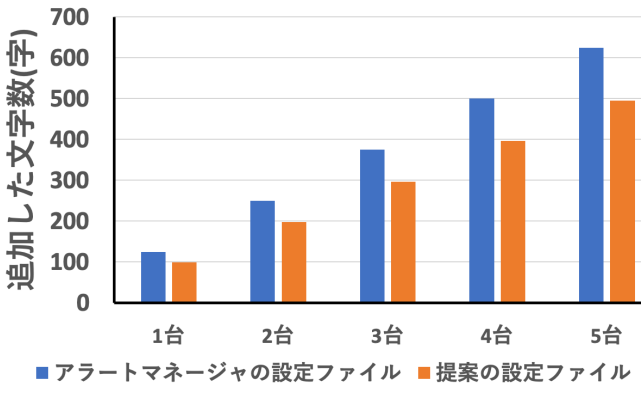

図 9: 追加した文字数

可読性を下げる要素としてインデントの深さがある．インデントは本提案の設定ファイルの形式で監視条件を組み合わせた際に深くなる．インデントの深さは可読性の低下に加え複雑度の増加につながる [25, 26]．プログラミング言語の複雑度を計算する式として循環的複雑度があり数式を用いることで提案の設定ファイルの形式で条件を記述した際の複雑度の数値がわかる．

一行あたりの文字数は増加すると可読性の低下につながる．その解決策として予約語やコマンドを一文字や略語で記述することで改善できる．しかし一文字で表現することや略語は可読性を下げる原因となる [21][*13]．本提案の設定ファイルの形式では予約語のみに構文ルールを適応したが，監視モジュールにも適応することで文字数が短くなった上で可読性が高くなったと言える．

修正時の実験で監視対象として追加した Kubernetes クラスタのワーカは VM で作成している．企業で使われているシステムは VM だけで数百台から数千台をこえる．そのため実験の規模を合わせるには監視対象として VM を数百台から数千台用意する必要がある．しかしその分の VM を用意するのは物理機器の配置や電力設備の配置の面から手間である．そのため設定ファイルで数百台から数千台の監視をしている想定をし記述実験を行うことで手間を省くことができる．

7. おわりに

課題は監視対象や監視条件の記述箇所が増えると可読性が低下することである．可読性が低下する要因として一行あたりの文字数，行数の増加，機能の混在化をあげた．本稿では監視システムでアラートの送信先と障害対応時の URL の記述箇所を一箇所に集約し，監視対象部，監視条件部，配置方法部の記述箇所を分割することで可読性を向上させる設定ファイルの形式を提案した．その際に設定ファイルを読み取り監視を行う監視ソフトウェアは独自で作成した．実験評価では 10 個の監視項目を記述した提案の設定ファイルと Prometheus のアラートマネージャの形式を可読性の指標となる一行あたりの文字数，行数，修正時間，追加した文字数を比較した．監視項目を記述した際の行数は提案の設定ファイルの形式が Prometheus のアラートマネージャの設定ファイルの形式に比べて 155 行削減でき，約 29%削減した．一行あたりの文字数では 10 個目の監視項目を記述した際に 116 文字削減した．修正作業として監視対象と監視条件を追加した．修正作業では修正にかかった時間と追加した際の文字数を比較した．比較の修正時間では平均で 18%削減でき，提案の設定ファイルの可読性が

[*11] URL:https://pep8-ja.readthedocs.io/ja/latest/，閲覧日 2023 年 11 月 22 日

[*12] URL:https://ferret-plus.com/4951, 閲覧日 2023:11/21

[*13] URL:https://www.nottingham.ac.uk/brand/styleguide/?id=42a72988-5cdc-4795-931c-cfd6e2c0d2d5, 閲覧日 2023:10/30

高いことがわかった．追加した文字数では最大で 125 文字削減できた．

参考文献

[1] Shan, Y. G., Chao, L., Guangjian, G. and Gao, F.: Research on Monitoring of Information Equipment Based on Zabbix for Power Supply Company, *2021 3rd International Conference on Applied Machine Learning (ICAML)*, pp. 487–491 (online), DOI: 10.1109/ICAML54311.2021.00108 (2021).

[2] bin Mohd Shuhaimi, M. A. A., binti Zainal Abidin, Z., binti Roslan, I. and binti Anawar, S.: The new services in Nagios: Network bandwidth utility, email notification and sms alert in improving the network performance, *2011 7th International Conference on Information Assurance and Security (IAS)*, pp. 86–91 (online), DOI: 10.1109/ISIAS.2011.6122800 (2011).

[3] Chen, L., Xian, M. and Liu, J.: Monitoring System of OpenStack Cloud Platform Based on Prometheus, *2020 International Conference on Computer Vision, Image and Deep Learning (CVIDL)*, pp. 206–209 (online), DOI: 10.1109/CVIDL51233.2020.0-100 (2020).

[4] Liu, J., Qu, C. and Zhou, T.: A Novel Cloud Computing Platform Monitoring System based on Nagios, *2023 3rd International Conference on Smart Data Intelligence (ICSMDI)*, pp. 169–172 (2023).

[5] Kim, S.-T., Park, H.-J. and Kim, Y.-C.: The load monitoring of Web server using mobile agent, *2001 International Conferences on Info-Tech and Info-Net. Proceedings (Cat. No.01EX479)*, Vol. 5, pp. 89–94 vol.5 (2001).

[6] Ahlgren, V., Andersson, S., Brandt, J., Cardo, N., Chunduri, S., Enos, J., Fields, P., Gentile, A., Gerber, R., Gienger, M., Greenseid, J., Greiner, A., Hadri, B., He, Y., Hoppe, D., Kaila, U., Kelly, K., Klein, M., Kristiansen, A., Leak, S., Mason, M., Pedretti, K., Piccinali, J.-G., Repik, J., Rogers, J., Salminen, S., Showerman, M., Whitney, C. and Williams, J.: Large-Scale System Monitoring Experiences and Recommendations, *2018 IEEE International Conference on Cluster Computing (CLUSTER)*, pp. 532–542 (2018).

[7] Karmakar, S. and Zhu, Y.: Visualizing multiple text readability indexes, *2010 International Conference on Education and Management Technology*, pp. 133–137 (online), DOI: 10.1109/ICEMT.2010.5657684 (2010).

[8] Johnson, J., Lubo, S., Yedla, N., Aponte, J. and Sharif, B.: An Empirical Study Assessing Source Code Readability in Comprehension, *2019 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pp. 513–523 (2019).

[9] Peterson, C. S., Park, K.-i., Baysinger, I. and Sharif, B.: An Eye Tracking Perspective on How Developers Rate Source Code Readability Rules, *2021 36th IEEE/ACM International Conference on Automated Software Engineering Workshops (ASEW)*, pp. 138–139 (2021).

[10] Piantadosi, V.: On the Evolution of Code Readability, *2022 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pp. 597–601 (online), DOI: 10.1109/ICSME55016.2022.00082 (2022).

[11] Buse, R. P. L. and Weimer, W.: A metric for software readability, *International Symposium on Software Testing and Analysis* (2008).

[12] Buse, R. P. and Weimer, W. R.: Learning a Metric for Code Readability, *IEEE Transactions on Software Engineering*, Vol. 36, No. 4, pp. 546–558 (online), DOI: 10.1109/TSE.2009.70 (2010).

[13] Nanavati, A. and Bias, R.: Optimal Line Length in Reading–A Literature Review, *Visible Language* (2005).

[14] Sedano, T.: Code Readability Testing, an Empirical Study (2016).

[15] Akour, M. and Falah, B.: Application domain and programming language readability yardsticks, *2016 7th International Conference on Computer Science and Information Technology (CSIT)*, pp. 1–6 (online), DOI: 10.1109/CSIT.2016.7549476 (2016).

[16] Wiese, E. S., Rafferty, A. N. and Fox, A.: Linking Code Readability, Structure, and Comprehension Among Novices: It's Complicated, *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*, pp. 84–94 (online), DOI: 10.1109/ICSE-SEET.2019.00017 (2019).

[17] Curtis-Black, A., Willig, A. and Galster, M.: Scout: A Framework for Querying Networks, *2019 15th International Conference on Network and Service Management (CNSM)*, pp. 1–7 (online), DOI: 10.23919/CNSM46954.2019.9012704 (2019).

[18] X'i li Kue, L. L. and Zhou, X. X.: XM-ADL, AN EXTENSIBLE MARKUP ARCHITECTURE DESCRIPTION LANGUAGE.

[19] Pezoa, F., Reutter, J. L., Suarez, F., Ugarte, M. and Vrgoč, D.: Foundations of JSON schema, *Proceedings of the 25th international conference on World Wide Web*, pp. 263–273 (2016).

[20] Eriksson, M. and Hallberg, V.: Comparison between JSON and YAML for data serialization, *The School of Computer Science and Engineering Royal Institute of Technology*, pp. 1–25 (2011).

[21] Mon, C. T., Hlaing, S. S., Tin, M. M., Khin, M. M., Lwin, T. M. M. and Myo, K. M.: Code Readability Metric for PHP, *2019 IEEE 8th Global Conference on Consumer Electronics (GCCE)*, pp. 929–930 (online), DOI: 10.1109/GCCE46687.2019.9015229 (2019).

[22] Sinha, A., Sejwal, L., Kumar, N. and Yadav, A.: Implementation of ICMP based Network Management System for heterogeneous networks, *2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom)*, pp. 382–387 (2015).

[23] López, C., Morato, D., Magaña, E. and Izal, M.: Validation of HTTP Response Time From Network Traffic as an Alternative to Web Browser Instrumentation, *IEEE Transactions on Network and Service Management*, Vol. 19, No. 2, pp. 976–990 (online), DOI: 10.1109/TNSM.2021.3121468 (2022).

[24] Scalabrino, S., Linares-Vásquez, M., Poshyvanyk, D. and Oliveto, R.: Improving code readability models with textual features, *2016 IEEE 24th International Conference on Program Comprehension (ICPC)*, pp. 1–10 (2016).

[25] Ebert, C., Cain, J., Antoniol, G., Counsell, S. and Laplante, P.: Cyclomatic Complexity, *IEEE Software*, Vol. 33, No. 6, pp. 27–29 (online), DOI: 10.1109/MS.2016.147 (2016).

[26] Gill, G. K. and Kemerer, C. F.: Cyclomatic complexity density and software maintenance productivity, *IEEE transactions on software engineering*, Vol. 17, No. 12, pp. 1284–1288 (1991).