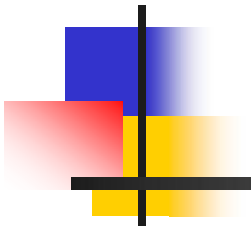
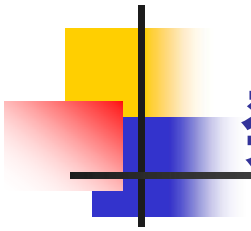


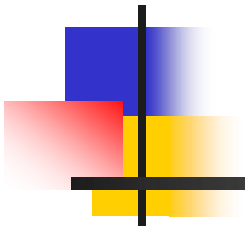
高信頼性HAクラスタに向けた障害通知フレームワークの実装

NTTコムウェア株式会社
オープンソースソフトウェア推進部
大塚 憲司
E-Mail: kenji.otsuka@nttcom.co.jp



発表内容

- 背景
- 現状のHAクラスタソフト
- 設計の観点
- 障害通知フレームワーク
- ハートビートプラグイン
- 検証
- まとめと課題

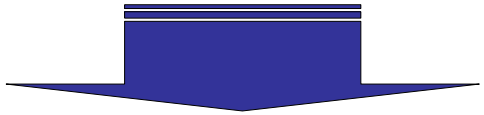


背景

背景

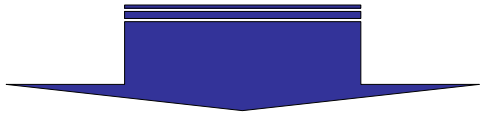
Linuxの適用範囲

Webサーバ等から大規模システムへのLinux適用



大規模システムに求められるもの

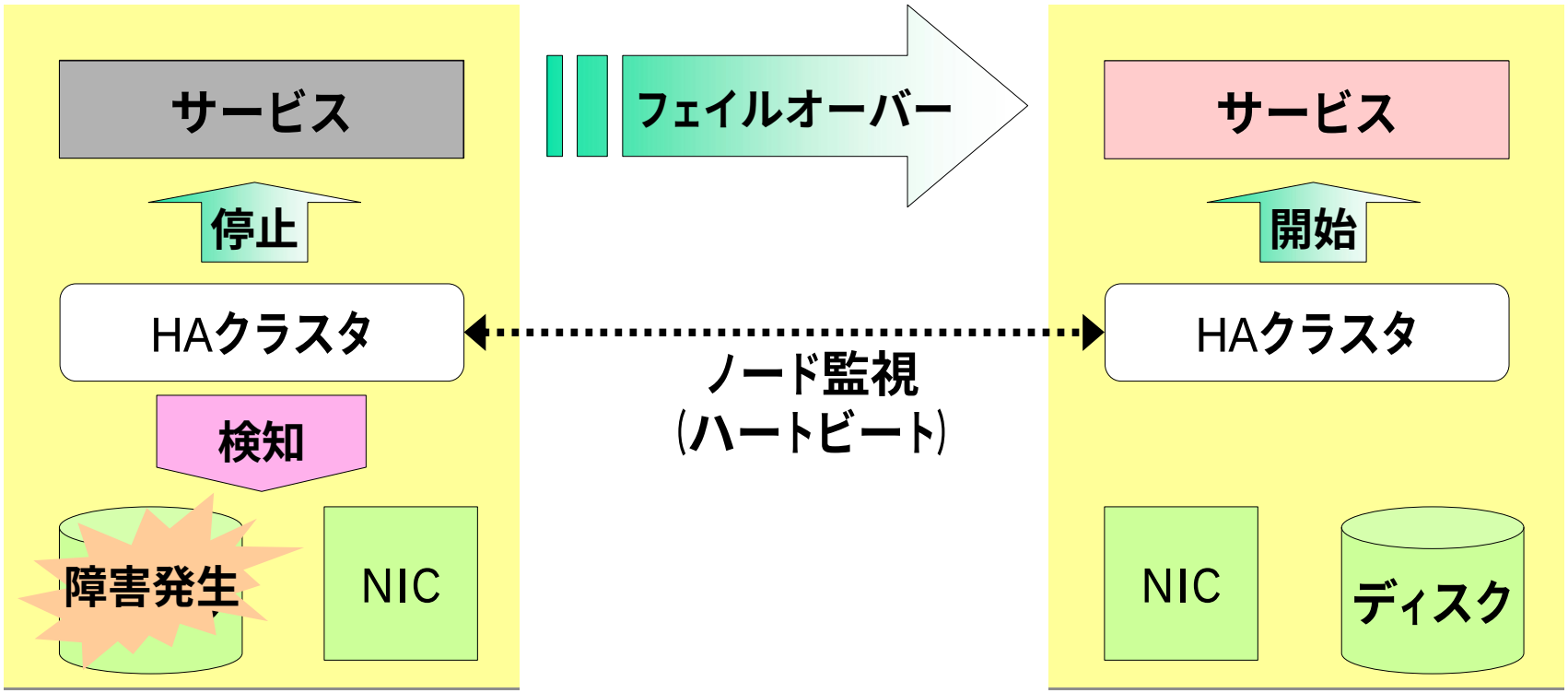
高いRASIS (Reliability Availability Serviceability Integrity Security)



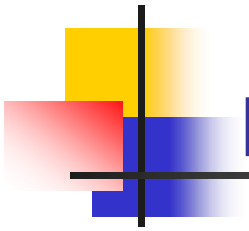
Availability (可用性) を視てみると

HA型クラスタソフトを用いたシステム構成がある

HA型クラスタソフトとは



- ハードやソフトの障害の検出
- サービスの切り替え



HA型クラスタソフトの問題点

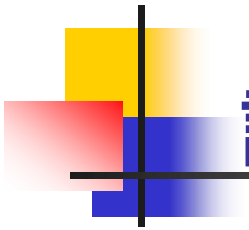
- メモリ高負荷状態におけるノード障害の誤検知
  - カーネルメモリ高負荷状態においてハートビートが遅延し障害と誤検出する
- ディスク障害においてデータ不整合が発生する
  - 障害検知から、その対処を行う際に他の処理が入り込める隙間があるため、データの不整合が発生する危険性がる

原因と解決策

- メモリ高負荷状態におけるハートビートの遅延
  - システムコール内で行われるメモリ確保に時間がかかるため
- ディスク障害において障害検知からその対処まで時間がかかる
  - ドライバのような低いレイヤで発生した障害をユーザ空間プロセスで対処しているため

ユーザ空間プロセスによる制御には限界

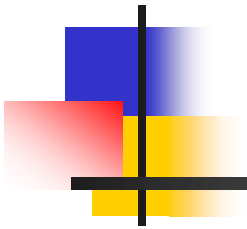
カーネル内に障害検知機能を組み込む



設計の観点

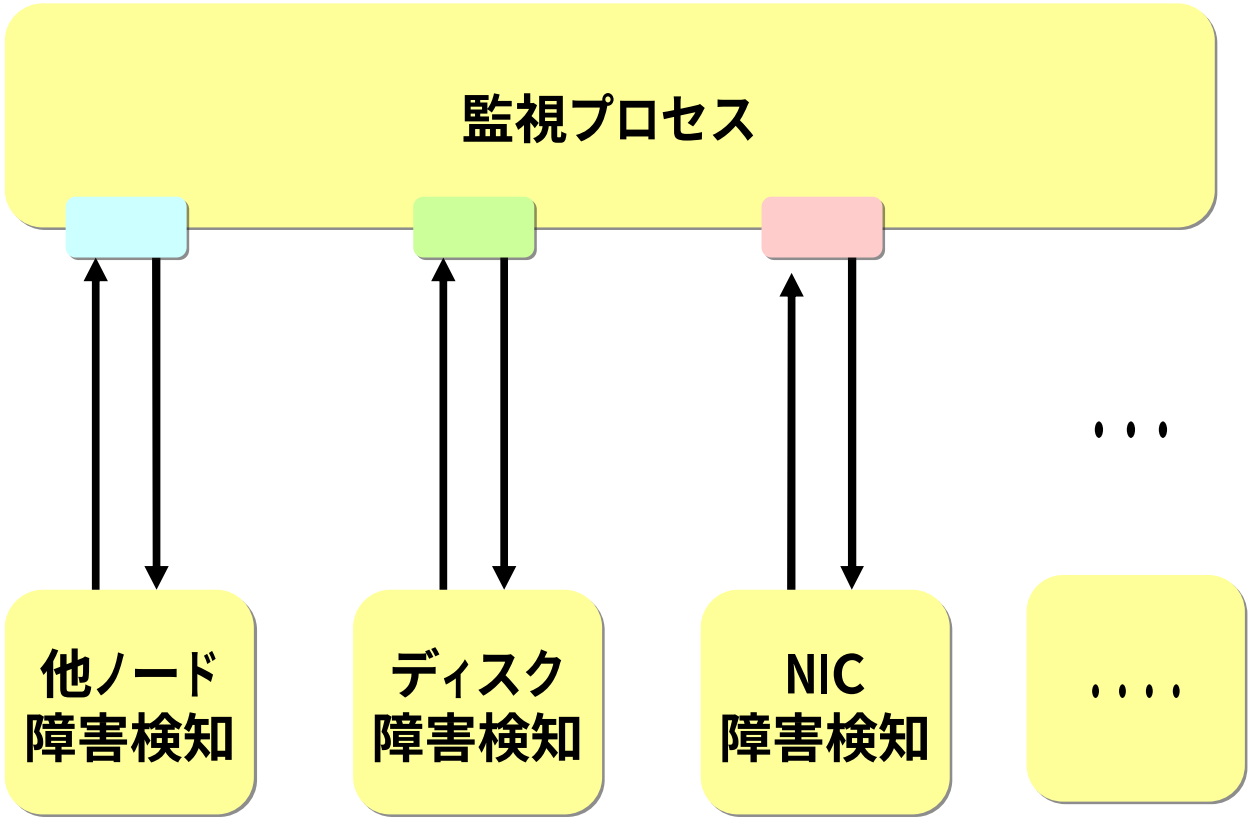
- 独立性
  - HA型クラスタソフト等の監視プロセスと障害検知機能が互いに依存しない
- 利便性
  - 様々な障害検知機能を動的に変更したい
- メンテナンス性
  - カーネルのバージョンアップに極力影響を受けない

**障害検知機能をまとめて管理する
フレームワークを導入する**



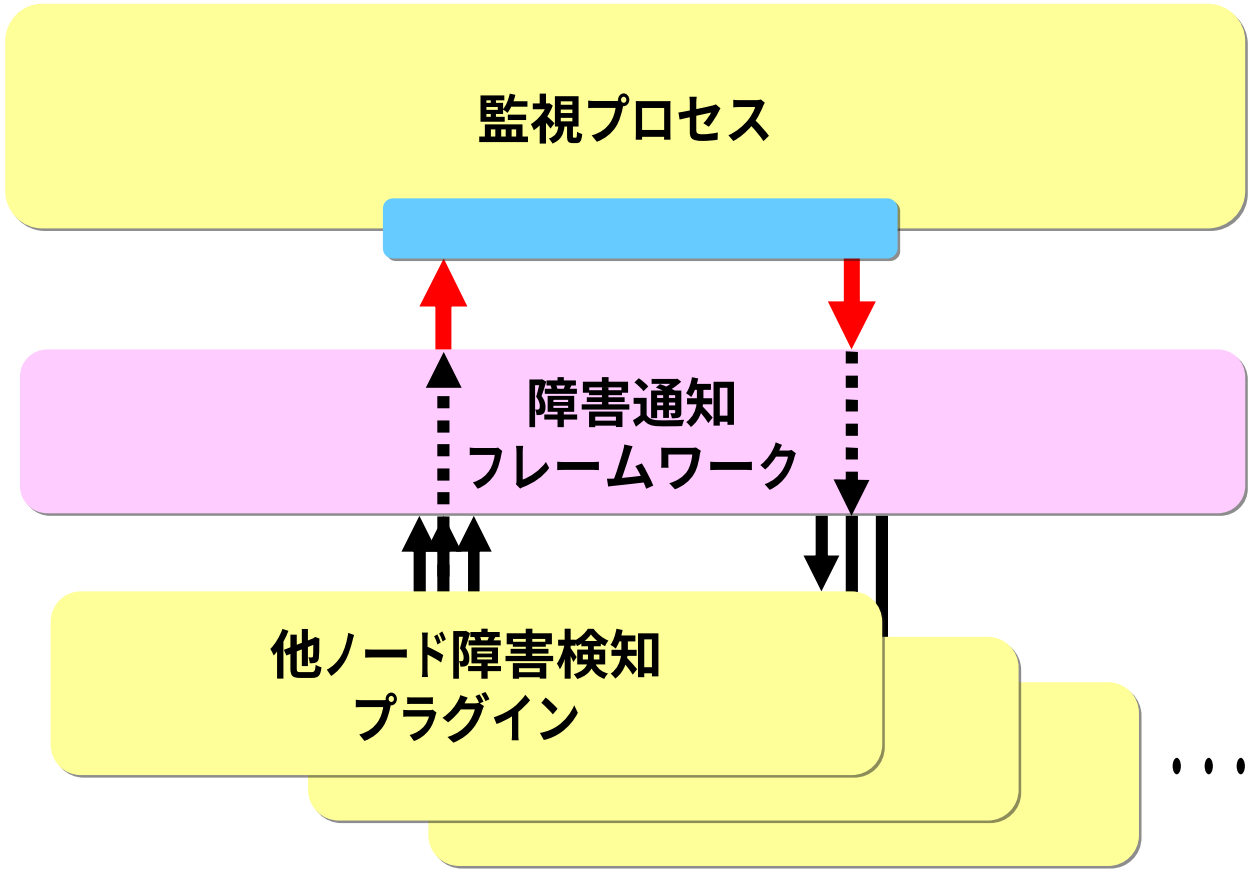
障害通知フレームワーク

障害通知フレームワークを入れないと



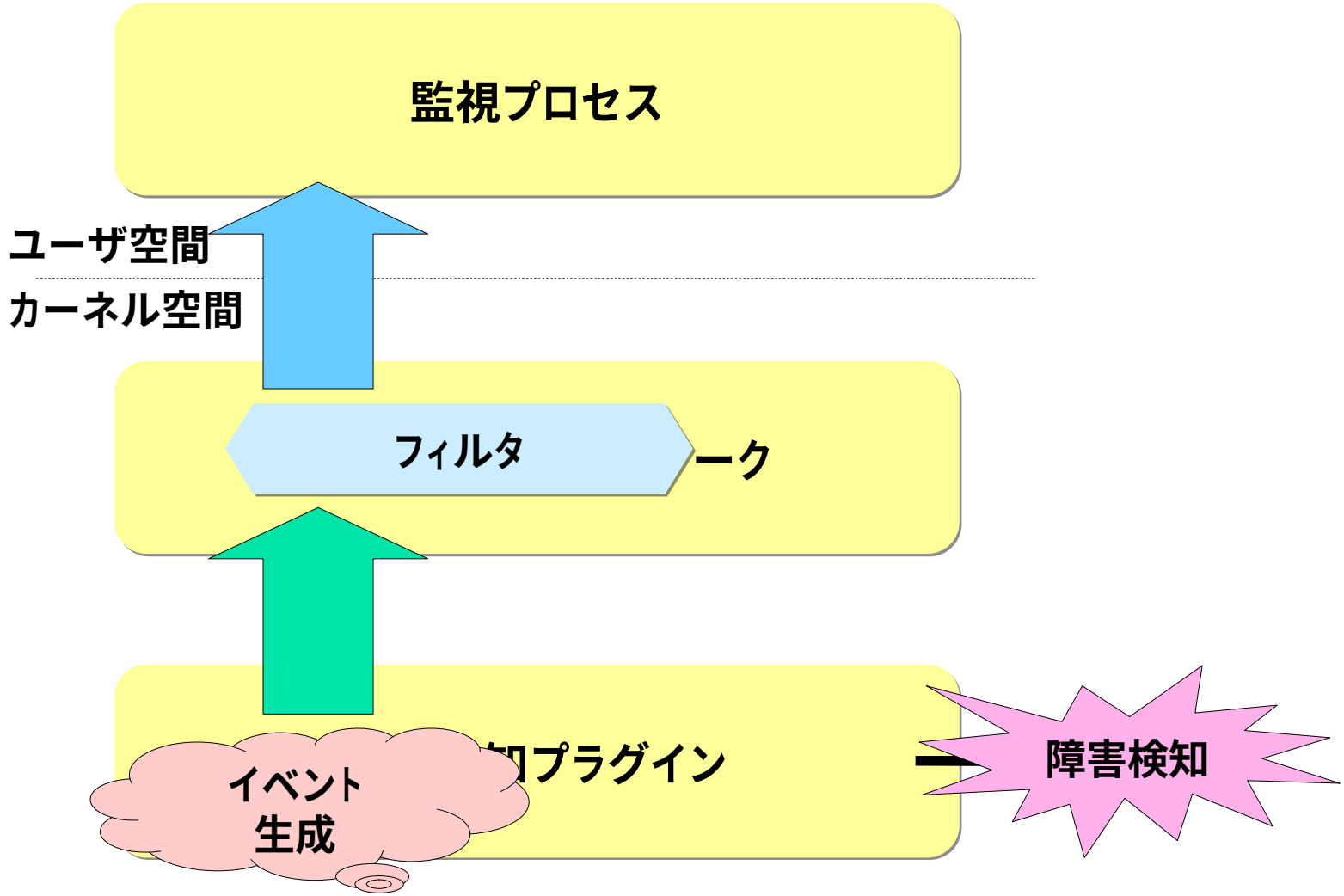
- 監視プロセスは障害ごとに対応する必要

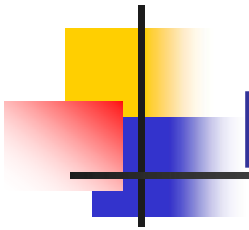
障害通知フレームワークの導入



- 監視プロセスはフレームワークのみに対応
- 各障害検知機能の差をフレームワークで吸収

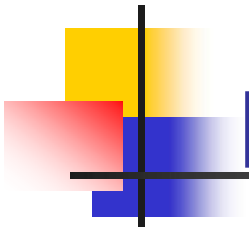
障害通知の基本的な流れ





障害通知フレームワークの特徴

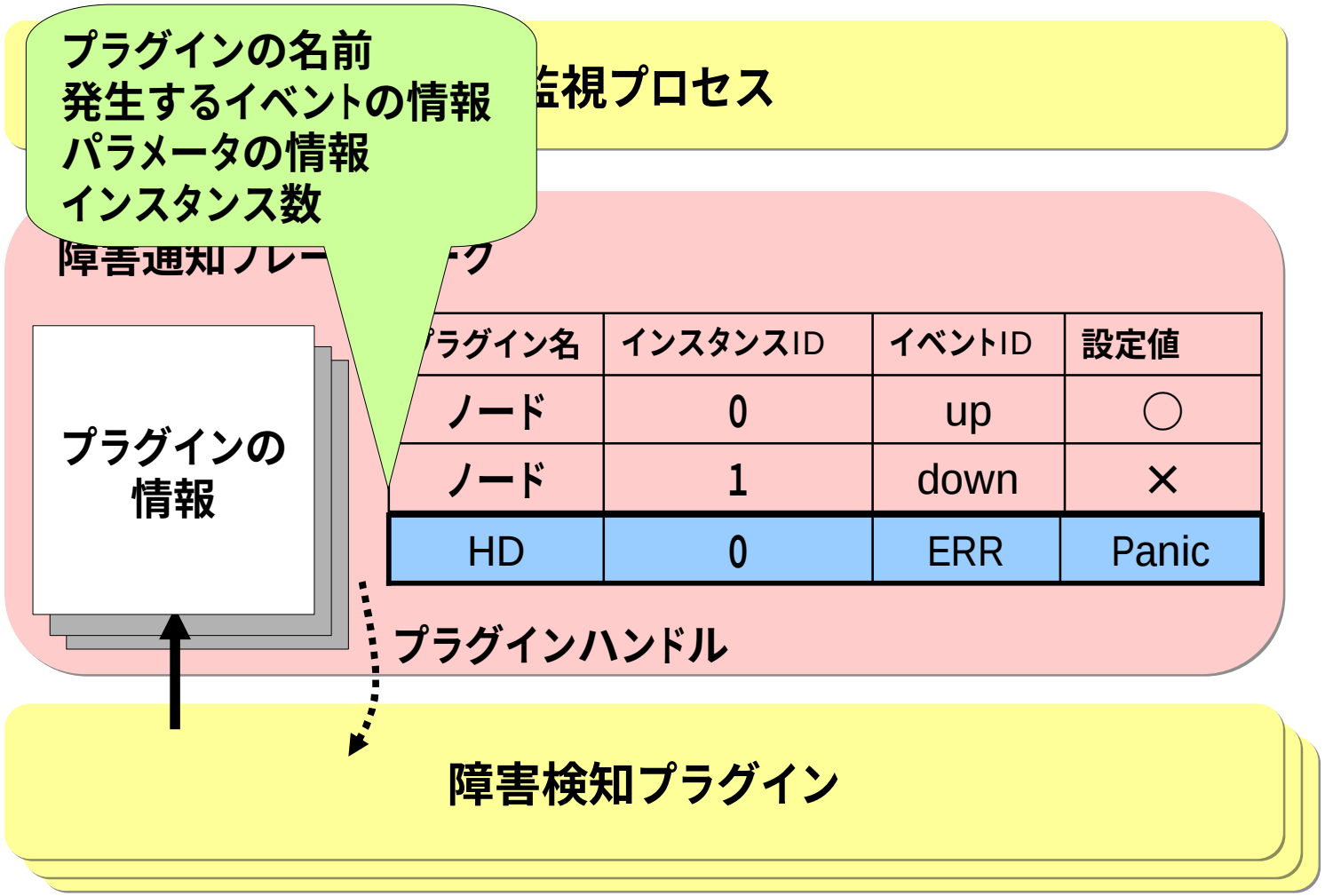
- 障害検知機能の管理
  - 障害検知機能はプラグインとして管理
  - 動的な追加･削除が可能
- イベントのフィルタリング
  - 監視プロセスへ通知するイベントのフィルタリング
  - イベントの発生を契機とするアクション
- インタフェースの統一
  - 監視プロセス側のインタフェースの統一
  - 障害検知機能側のインタフェースの統一



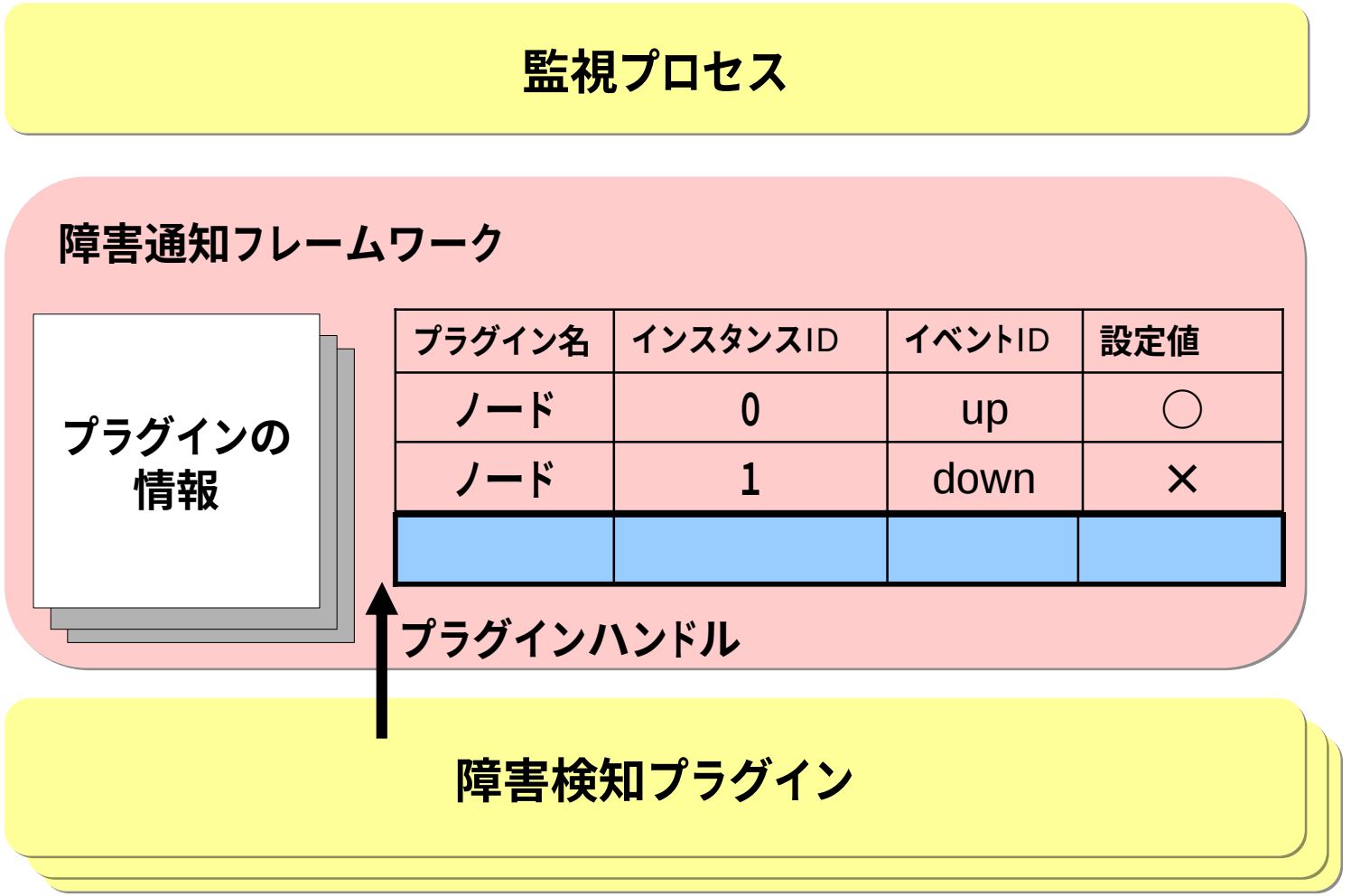
障害検知機能の管理

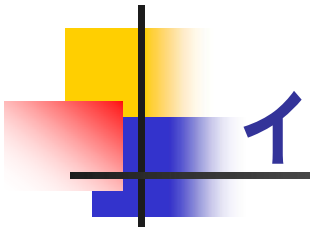
- 障害検知機能毎に情報を管理
  - プラグインの名前
  - 発生するイベントの情報
  - 必要なパラメータの情報
  - インスタンス数

プラグイン登録


プラグインの名前
発生するイベントの情報
パラメータの情報
インスタンス数
監視プロセス
障害通知フレー ク
プラグインの情報
プラグイン名
インスタンスID
イベントID
設定値
ノード
0
up
○
ノード
1
down
×
HD
0
ERR
Panic
プラグインハンドル
障害検知プラグイン

プラグイン解除

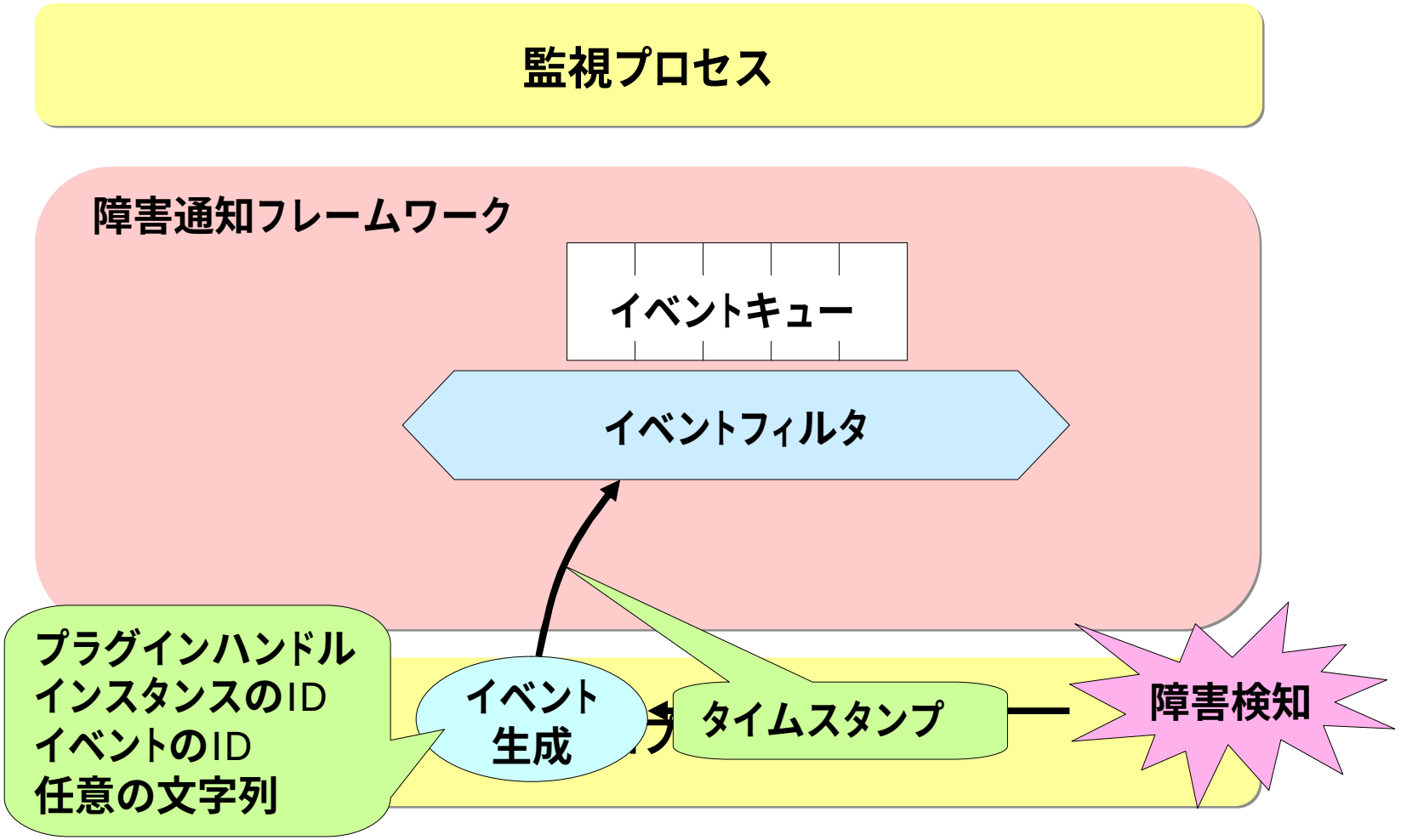

監視プロセス
障害通知フレームワーク
プラグインの情報
プラグイン名
インスタンスID
イベントID
設定値
ノード
0
up
○
ノード
1
down
×
プラグインハンドル
障害検知プラグイン



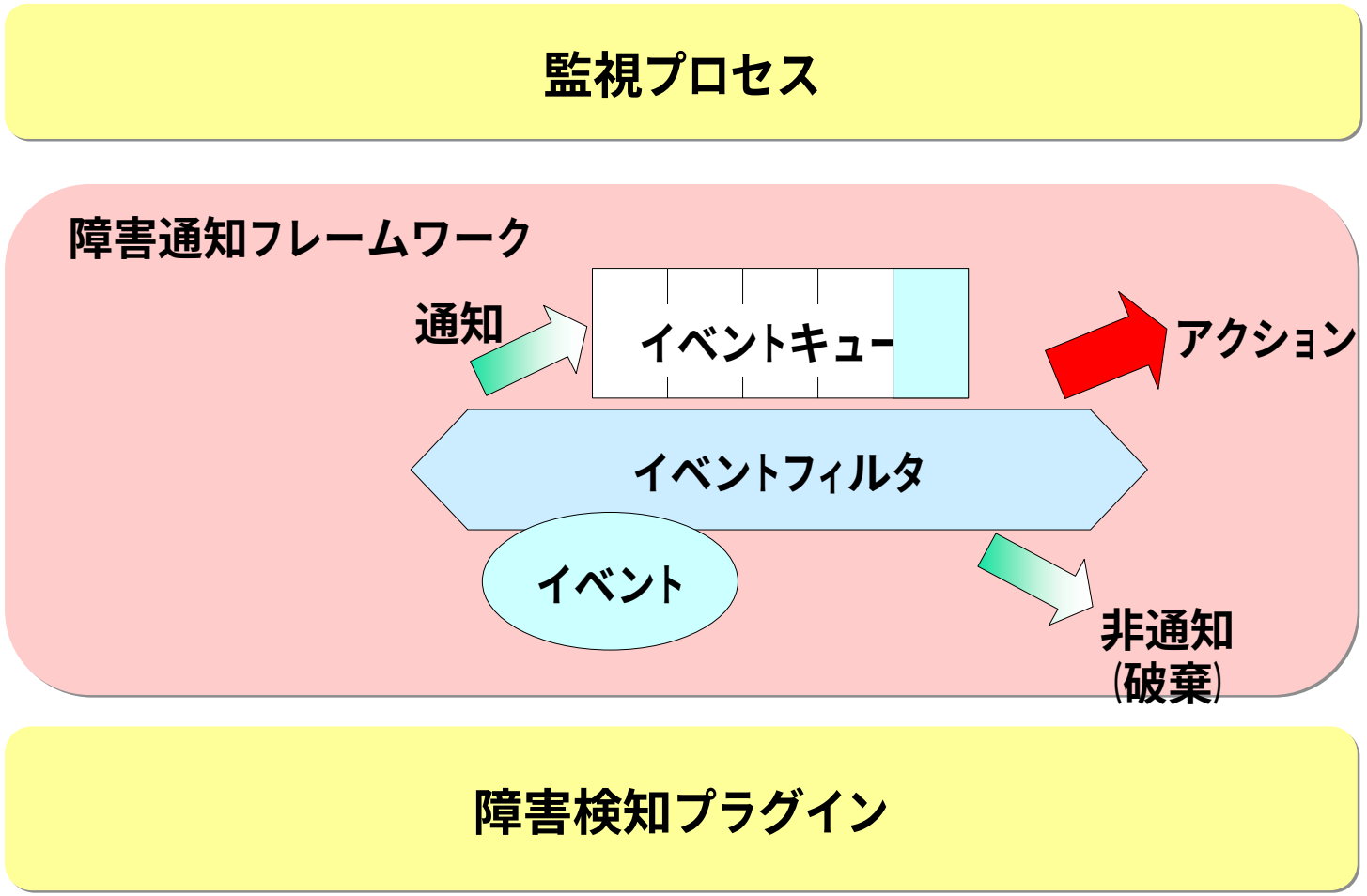
イベントのフィルタリング

- イベントのフィルタリング
  - 監視プロセスが必要とするイベントのみを通知
- イベントの発生を契機とするアクション
  - できるだけ早い段階で障害に対する対処を行う

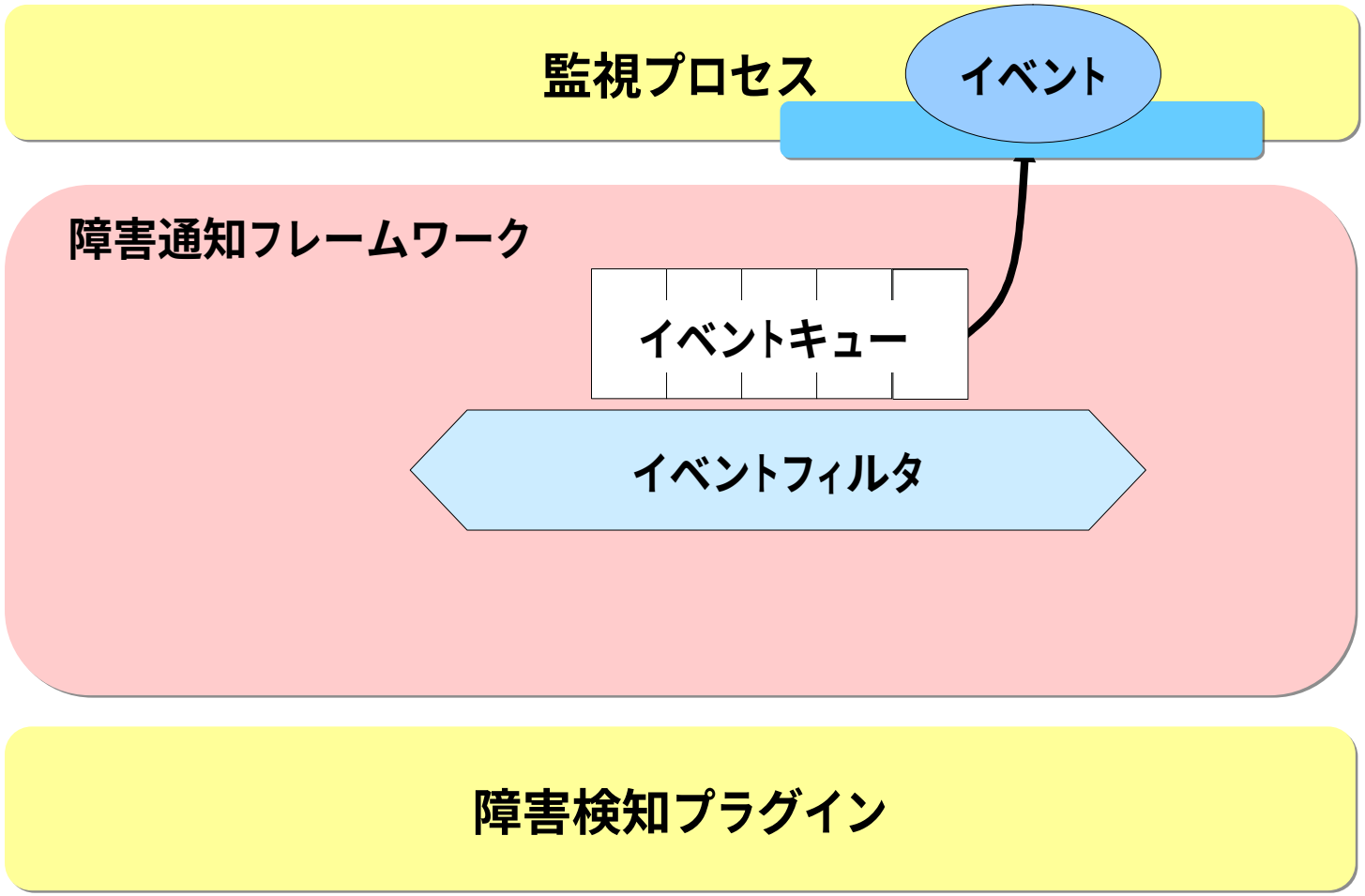
イベント受信

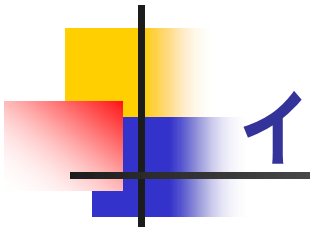

監視プロセス
障害通知フレームワーク
イベントキュー
イベントフィルタ
プラグインハンドル
インスタンスのID
イベントのID
任意の文字列
イベント
生成
タイムスタンプ
障害検知

イベントフィルタ


監視プロセス
障害通知フレームワーク
通知
イベントキュー
アクション
イベントフィルタ
イベント
非通知
(破棄)
障害検知プラグイン

イベント通知

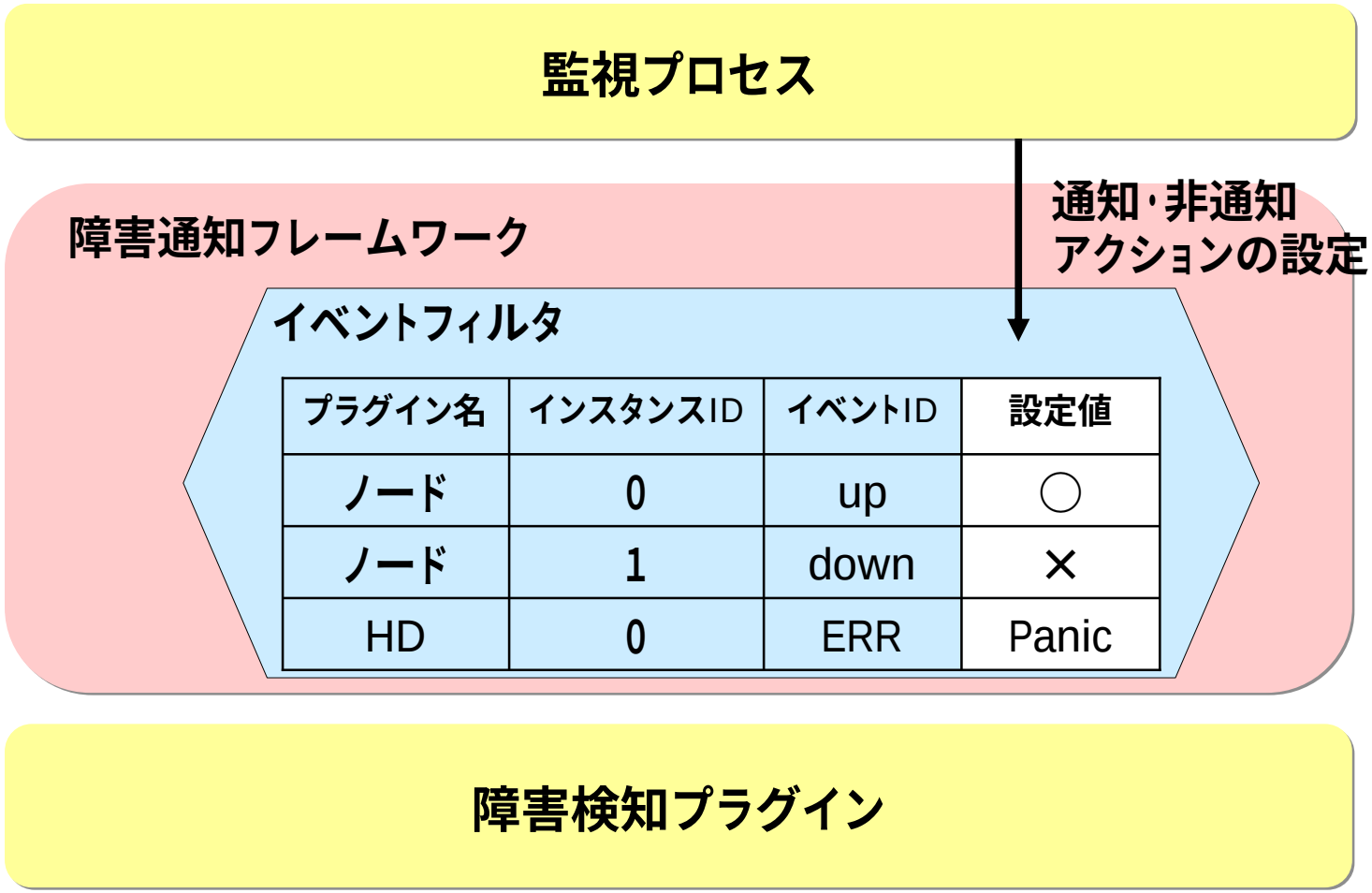




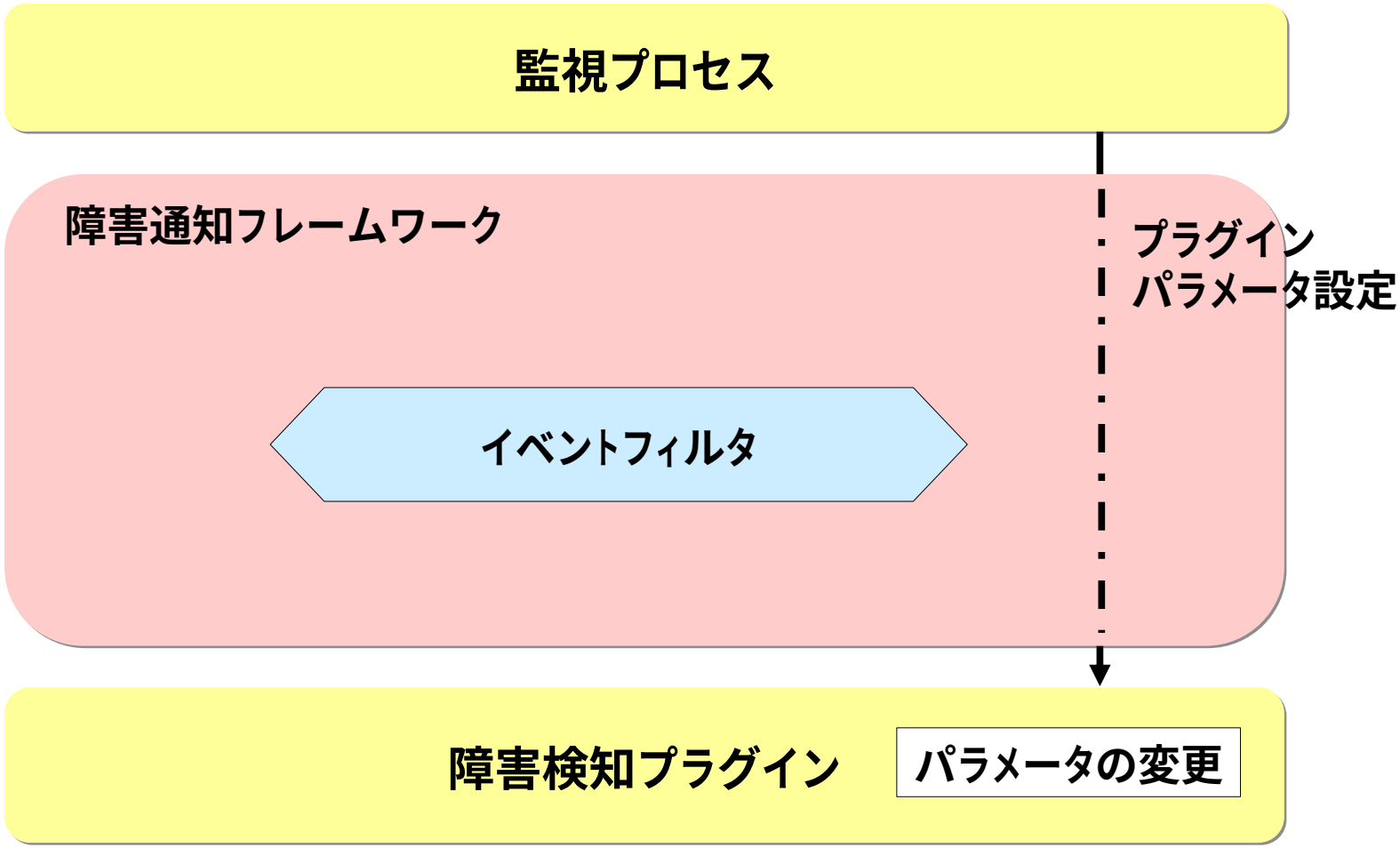
インタフェースの統一

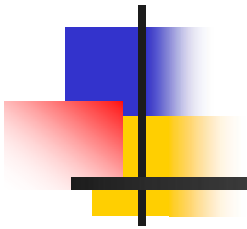
- イベントフィルタの設定
  - 通知／非通知（破棄）／アクションを指定
- プラグインパラメータの設定
  - 障害検知フレームワーク経由での設定
  - 監視プロセスからプラグインの隠蔽

イベントフィルタの設定



プラグインパラメータの設定

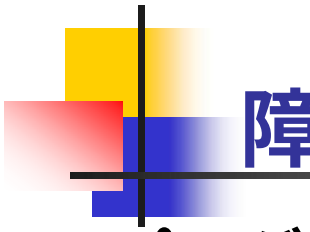




障害通知フレームワークの実装

実装状況

- 障害検知プラグインとのインタフェース
  - カーネル内の関数として実装
- 監視プロセスとのインタフェース
  - procファイルシステムを利用
    - イベントフィルタの設定
    - プラグインパラメータの設定
    - 監視プロセスへのイベント通知
- 実装レベル
  - カーネルモジュールとして実装
  - プロトタイプレベル

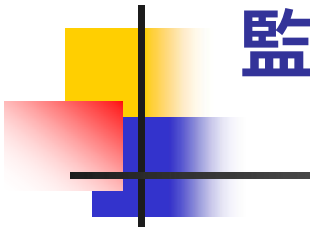


障害検知プラグインとのインタフェース

- プラグインの登録
  - void *fn_plugin_register(struct fn_plugin*, int)
  - 引数としてプラグインの情報を持つfn_plugin構造体とインスタンス数を渡す
  - 返り値としてプラグインのハンドルが返る
- プラグインの解除
  - void fn_plugin_unregister(void *)
  - 引数としてプラグインのハンドルを渡す
- プラグインからのイベントの通知
  - void fn_event_notify(void, int, unsigned long, char *)
  - 引数としてプラグインのハンドルやインスタンスID、イベントID、付属する情報を渡す

監視プロセスとのインタフェース
イベントフィルタの設定

- /proc/fn/configに設定値をwriteする
  - seteventコマンド
    - フィルタの設定を行う
    - setevent (プラグイン名) (インスタンスID) (イベント名) (通知・非通知・アクション)
  - cleareventコマンド
    - フィルタの設定を初期化する



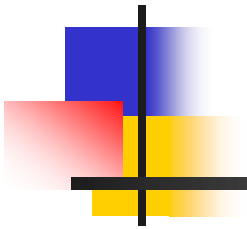
監視プロセスとのインタフェース
プラグインパラメータの設定

- /proc/fn/<plugin-name>/<parameter-name>**に設定値を**write**する**
  - **各プラグイン共通のパラメータ**

- /proc/fn/<plugin-name>/<instance-number>/<parameter-name>**に設定値を**write**する**
  - **各プラグインのインスタンス個別のパラメータ**

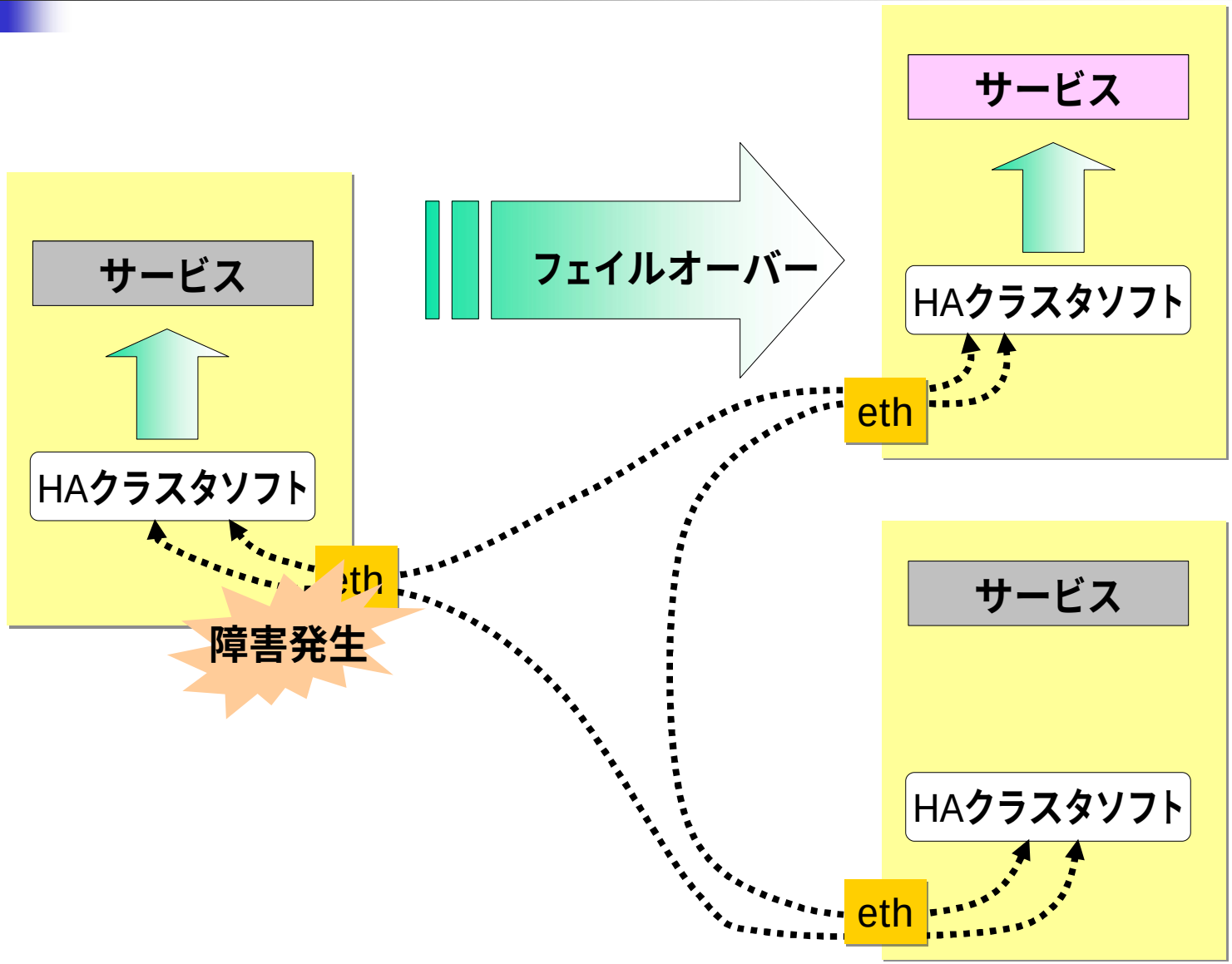
監視プロセスとのインタフェース
イベント通知

- /proc/fn/eventをreadする
  - (発生時刻) (プラグイン名) (インスタンス番号) (イベント名) (イベントに応じたデータ)
  - 複数のイベントがキューイングされている場合、1イベント毎のreadが必要

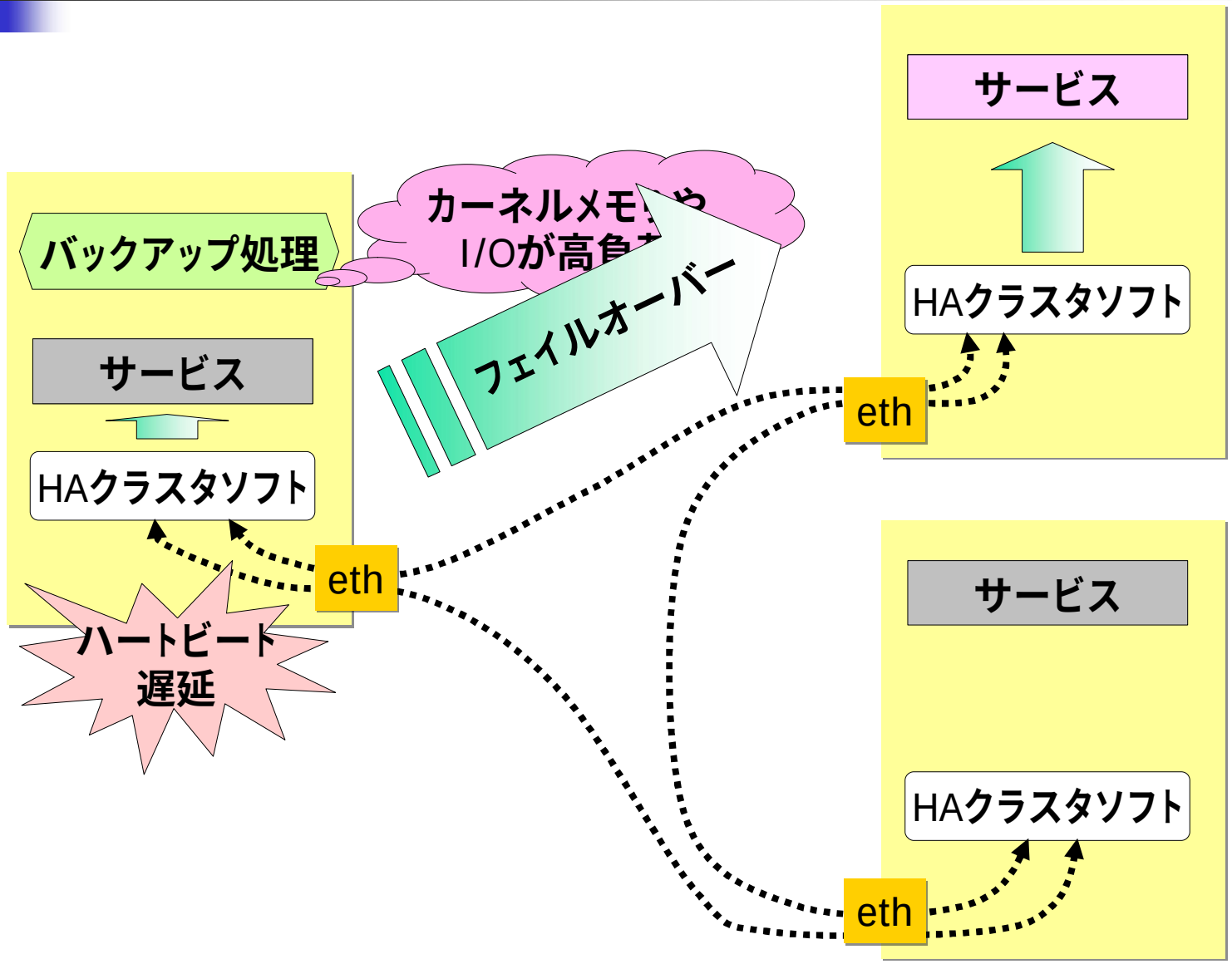


ハートビートプラグインの実装

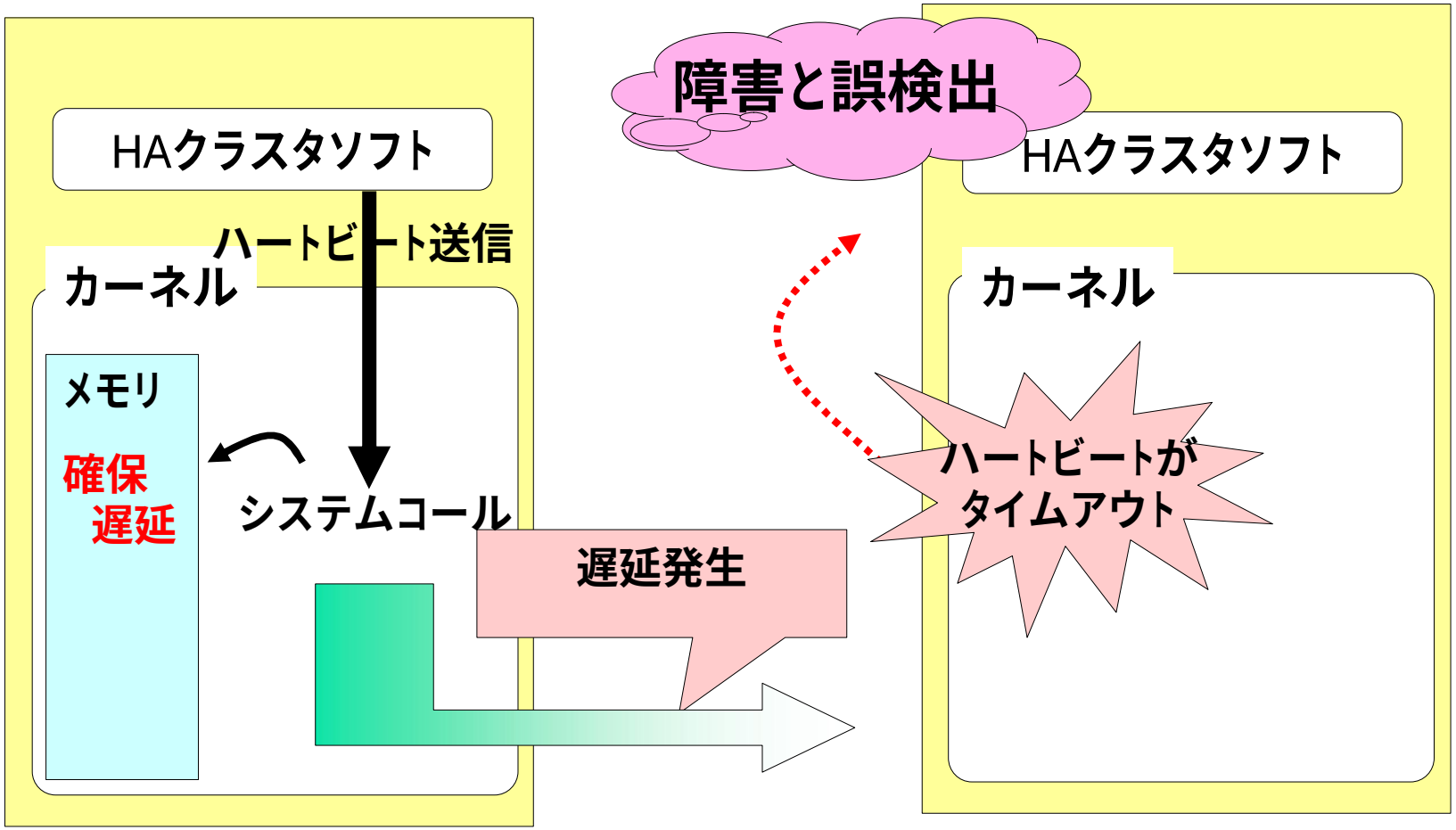
ノード監視

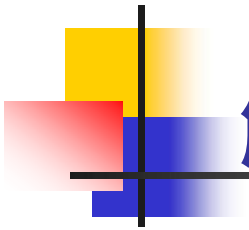

サービス
フェイルオーバー
サービス
HAクラスタソフト
HAクラスタソフト
eth
eth
障害発生
サービス
HAクラスタソフト
eth

ノード監視の問題点


サービス
バックアップ処理
カーネルメモリや
I/Oが高負荷
フェイルオーバー
サービス
HAクラスタソフト
HAクラスタソフト
eth
eth
ハートビート
遅延
サービス
HAクラスタソフト
eth

ノード監視の問題点


障害と誤検出
HAクラスタソフト
ハートビート送信
カーネル
メモリ
確保
遅延
システムコール
遅延発生
HAクラスタソフト
カーネル
ハートビートが
タイムアウト



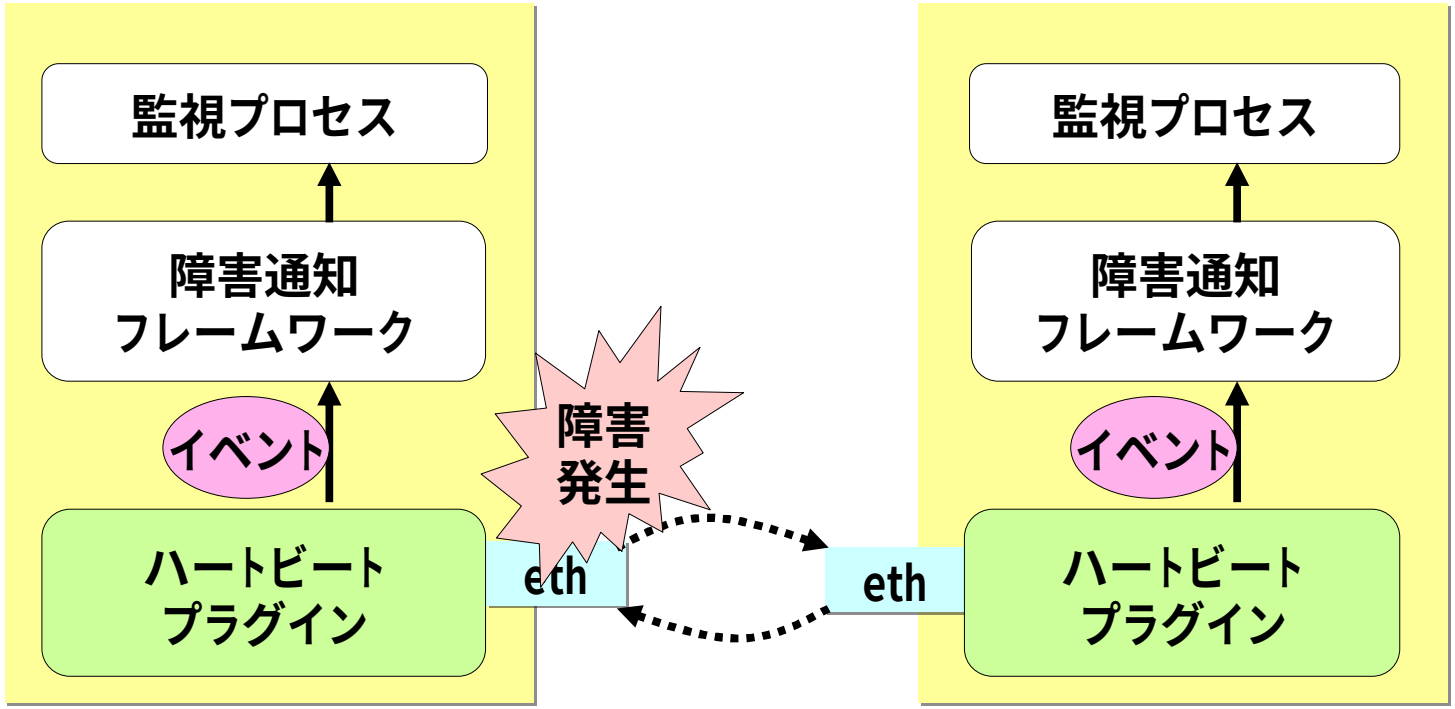
解決策

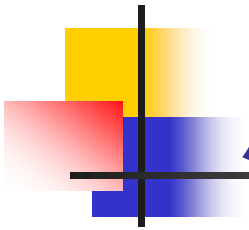
- 一般的なプロトコルスタックを使う以上は、メモリ高負荷には遅延の発生を避けられない。

- 高負荷時にも安定して動作するカーネル内のハートビートを実装する

ハートビートプラグイン

- カーネル内ハートビート機能
  - このフレームワークのプラグインとして実装

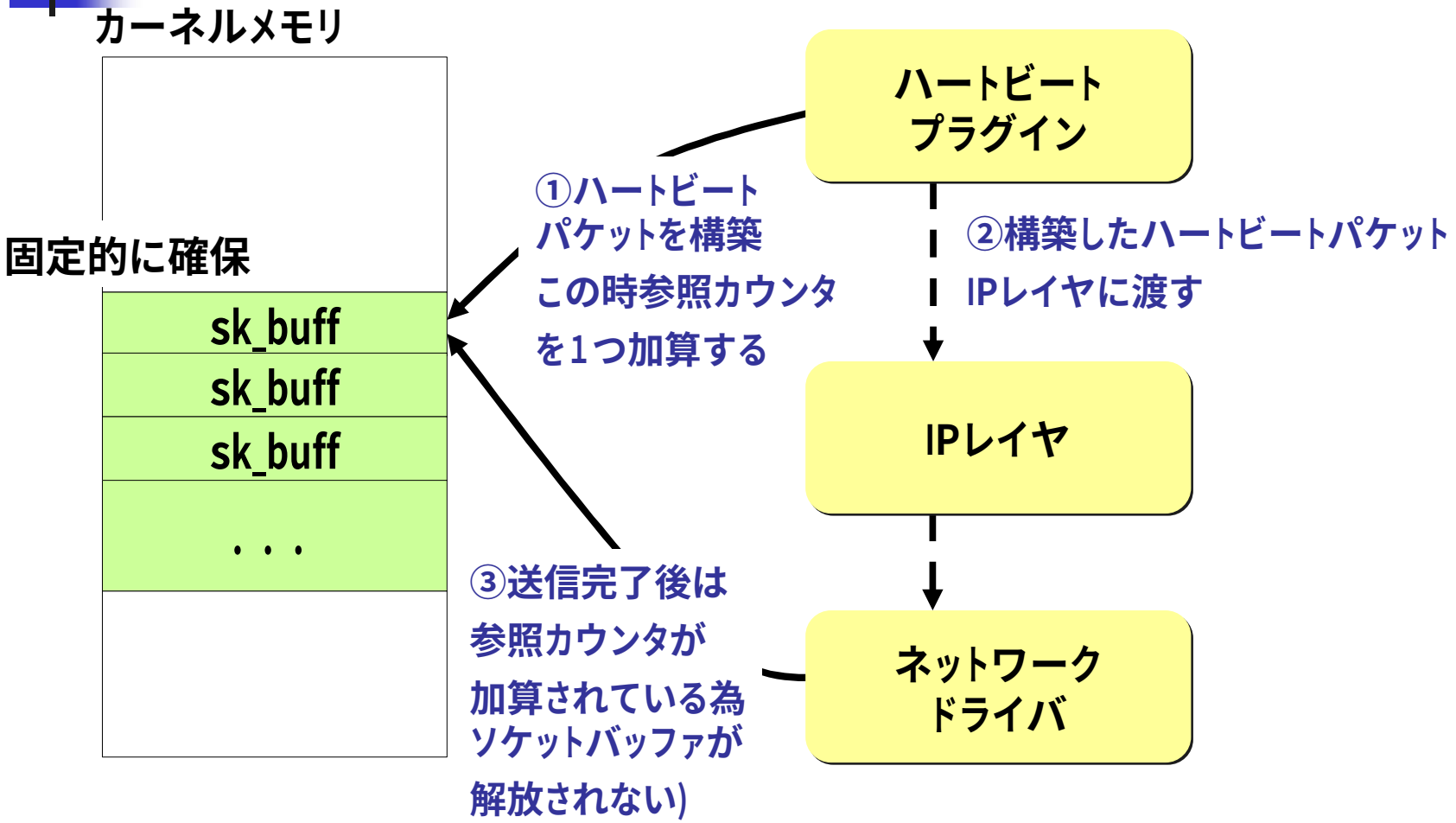




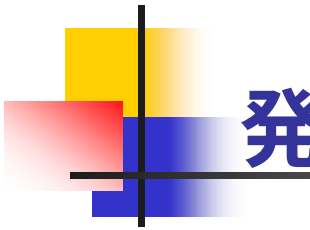
パケット送信処理遅延の回避

- ハートビートパケットの送信が遅延する
  - メモリ高負荷時にメモリの確保が遅延するため
- 解決策として
  - ハートビート専用のプロトコルスタックを実装する
  - 用途毎にメモリ割当て閾値を設ける
  - メモリを極力固定的に確保しパケットを送信する
  - メモリ確保処理にプライオリティを設ける

パケット送信処理遅延の回避

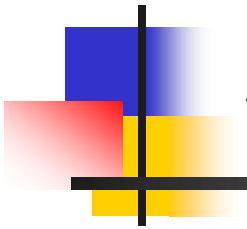


ハートビート用ソケットバッファの領域を固定的に確保

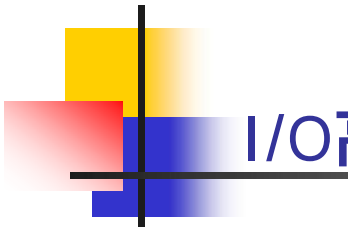


発生するイベント

- 監視対象ノードのダウン･アップ
  - あるノードからのハートビートを受信しているか
  - nodedown、nodeup
- ネットワークのダウン･アップ
  - あるNICがハートビートを受信しているか
  - linkdown、linkup
- 送受信異常
  - ソケットバッファが利用可能とならなかった場合など
  - sendfail

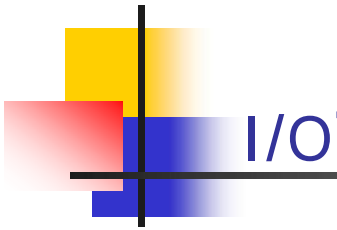


検証



I/O高負荷時におけるハートビート機能の検証 (環境)

- **ハードウェア**
  - CPU：Xeon 3.40GHz × 2
  - メモリ：2GBytes
  - ディスク：SCSI 42.8GBytes
    RAID-1
  - NIC：Gigabit Ethernet × 2
- **ソフトウェア**
  - OS：Debian Sarge (kernel 2.6.10)



I/O高負荷時におけるハートビート機能の検証（条件）

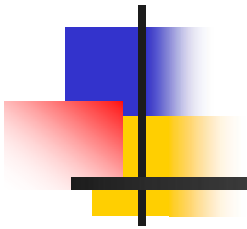
- ハートビートの設定
  - 送信間隔：750ms
  - ノードダウンと判定するまでの時間：4500ms

- I/O高負荷状態の発生方法
  - バックグラウンドでsyncを1秒間隔で連続的に実行
  - ddコマンドを同時に35プロセス起動
    - 1つのddコマンドは512MBytesのファイルを作成
  - 全てのddコマンドの完了を待ち合わせた後、ddを繰り返す

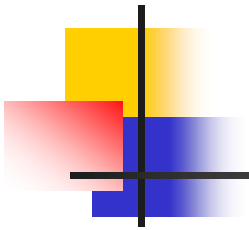
heartbeatとの比較結果

- I/O高負荷状態を2時間継続
- heartbeatでの検証
  - ハートビートの遅延警告 (Warning) が発生
- 障害検知フレームワークとハートビートプラグインでの検証
  - 750ms間隔でのパケット送信を確認

	障害検知フレームワーク ハートビートプラグイン	heatbeat
ハートビート遅延	0回	2～3回

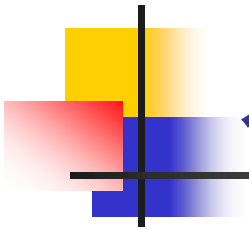


まとめと課題



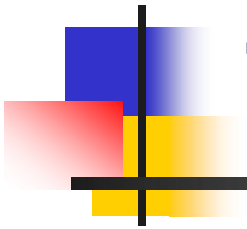
まとめ

- 障害通知フレームワーク
  - 監視プロセスへの障害情報の通知の形式を統一
  - 障害検知プラグインの動的追加／削除が可能
- ハートビートプラグイン
  - 指定通りの間隔でパケットの送受信
  - 固定的にソケットバッファを確保することにより送信処理の遅延を回避



今後の課題

- フレームワークやプラグインの機能強化
  - 障害通知フレームワーク
    - アクションの追加
    - シグナルによるイベントの通知
  - ハートビートプラグイン
    - リンクダウン検出機能の強化
- 新しいプラグインの実装
  - I/Oエラー検出プラグイン
  - メモリ負荷状態監視プラグイン



ご静聴いただきまして誠にありがとうございました

大塚憲司' 佐々木博正' 黒澤崇宏''

ソースコードや関連ドキュメントを以下のアドレスで公開予定
http://sourceforge.net/projects/linux-fn/

所属:'=NTTコムウェア株式会社 ''=VA Linux Systems Japan株式会社