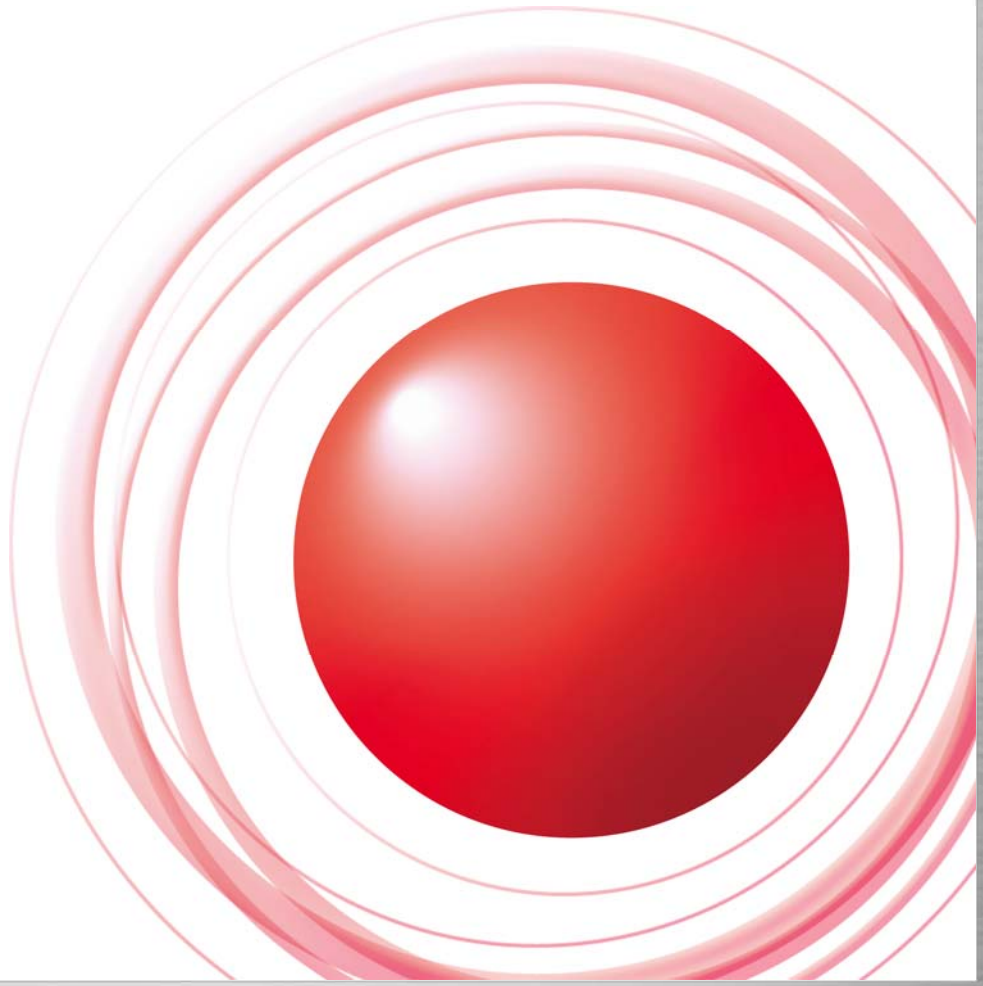
相関関係を用いた新監視システム

IIJ Internet Initiative Japan



2010/11/19

アプリケーションサービス部
岩永　義弘

Ongoing Innovation

背景

- 近年の傾向
  - システムの大規模化による管理ホスト数の増大
  - 仮想化による複雑化
- その影響
  - 障害発生件数が増加
  - 障害の原因調査にかかる時間が増加
  - 監視システムのターゲット追加作業が増加

- 将来、障害対応に要するコストは大きくなる

新監視システムが目指すもの

障害件数の減少

- 障害の兆候を事前に検知できれば・・・
  - 事前対策を施す → 障害の発生を予防

原因調査の時間短縮

- 障害発生時にシステムが原因を提示してくれたら・・・
  - 調査の手間が省ける

運用負荷の低減

- 設定作業が不要になれば・・・
  - ホスト追加の度に発生する閾値の設定が不要
  - 運用中の閾値変更も不要

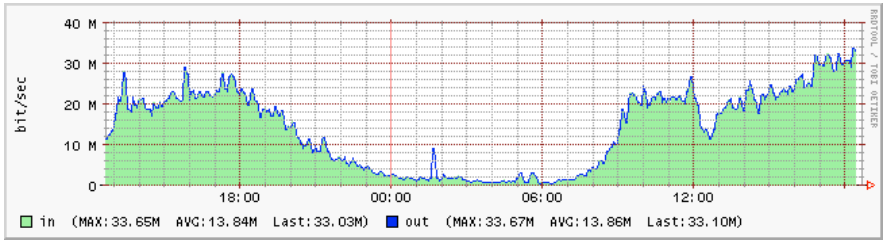
着眼点

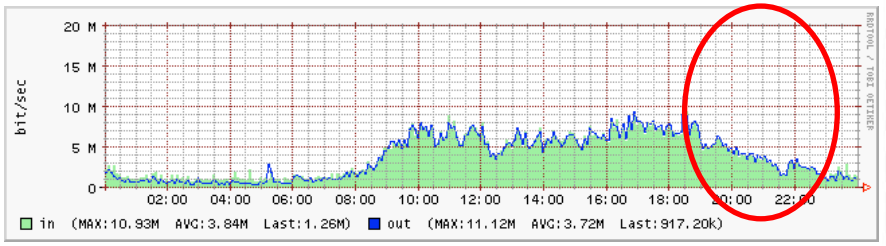
- 連動して機能するリソース同士には相関がある
  - 例えば
    - 「前段ホストのトラフィック」と「後段ホストのCPU使用率」
    - 並列に負荷分散された機器同士のリソース

相関がある状態

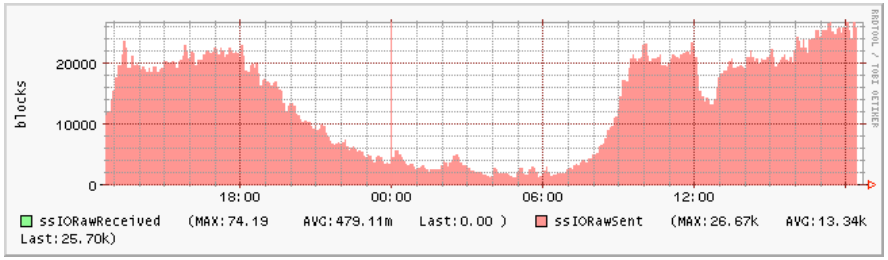
相関が崩れた状態

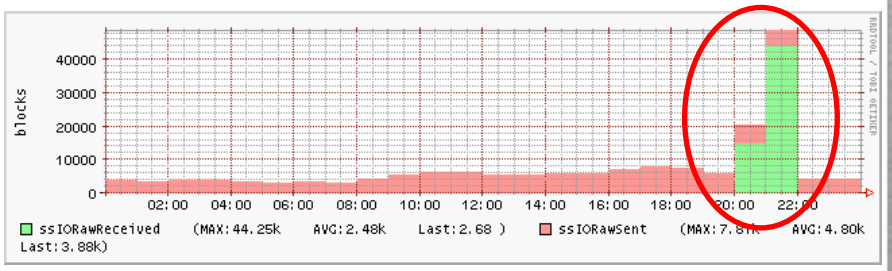
前段ホストのトラフィック





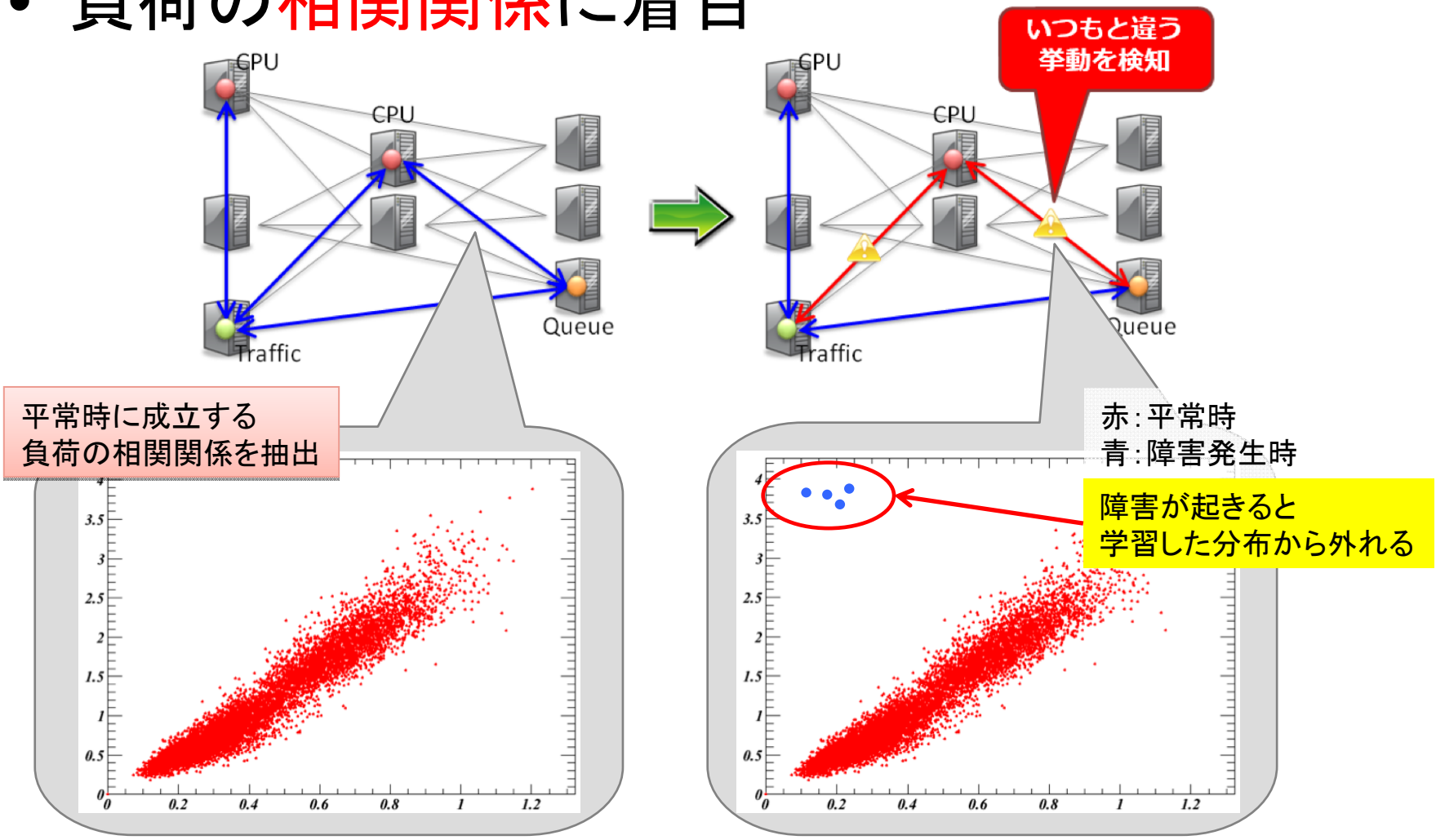
後段ホストのCPU使用率





検知の仕組み

- 負荷の相関関係に着目

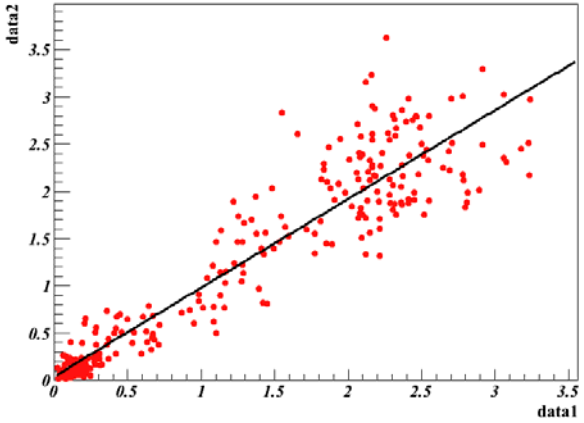


負荷の散布図　⇒ 線形関係になっている

問題が発生したホスト名、リソースを知ることができる

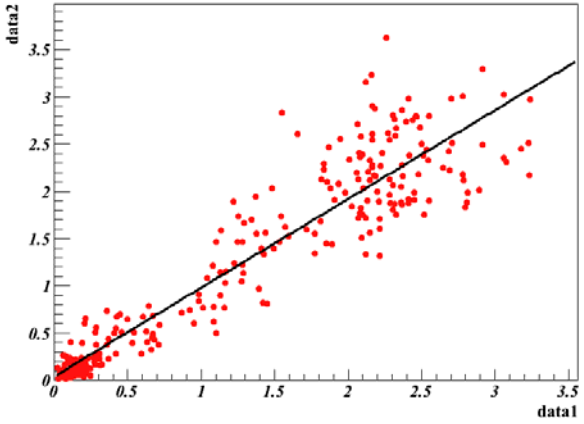
モデル化と異常の判定

- 学習対象
  - 全てのリソースの組合わせで相関関係を抽出
  - 相関のあるものだけを選び出し、モデル化
- 検知の方法
  - 学習したモデルから大きく外れていたら異常と判定
  - 通知しないケース
    - 一瞬だけ相関が崩れた場合（スパイク）
    - 相関の崩れた箇所が少ない場合
- 学習期間
  - ２週間
  - 平日と休日を分ける必要は無い



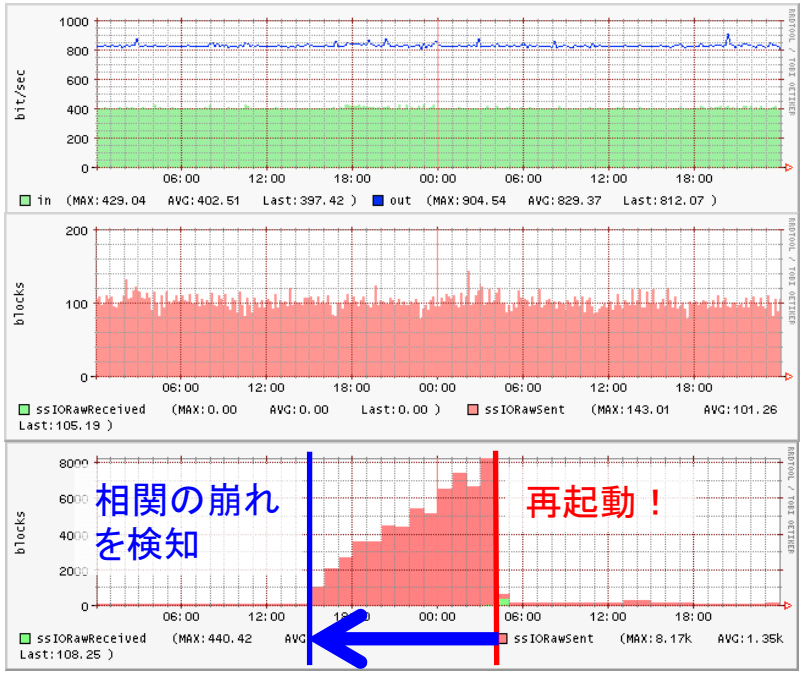
モデル化と異常の判定

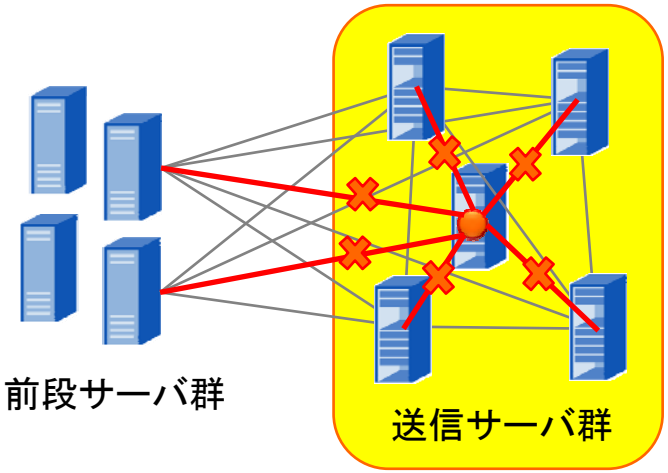
- 学習対象
  - 全てのリソースの組合わせで相関関係を抽出
  - 相関のあるものだけを選び出し、モデル化
- 検知の方法
  - 学習したモデルから大きく外れていたら異常と判定
  - 通知しないケース
    - 一瞬だけ相関が崩れた場合（スパイク）
    - 相関の崩れた箇所が少ない場合
- 学習期間
  - ２週間
  - 平日と休日を分ける必要は無い



障害検出事例(1)

- ホストが再起動を起こしたケース
  - 該当ホストのCPU使用率に相関の崩れが集中

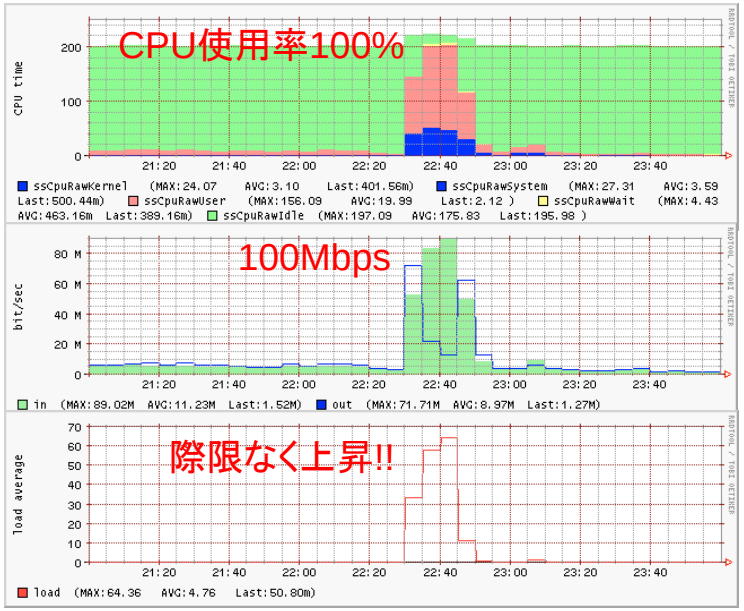


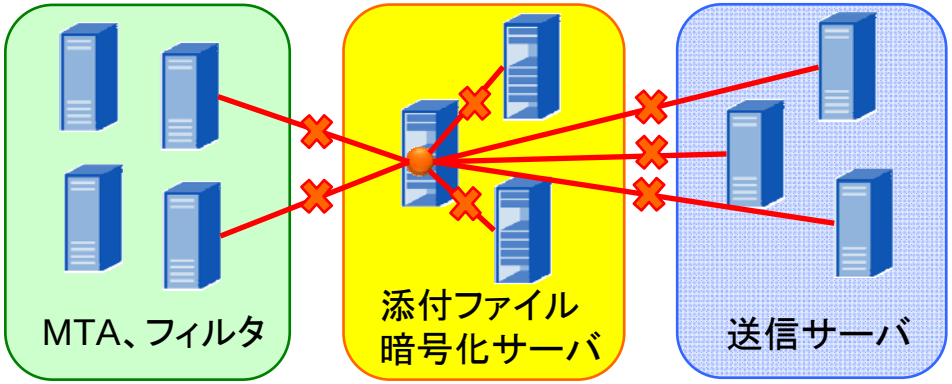


  - そのホストのCPU負荷だけが徐々に増加し、相関が崩れた
  - 兆候を事前に検出
    - 予防できるようになる

障害検出事例(2)

- メールサービスの配送遅延障害
  - 暗号化サーバのロードアベレージに相関の異常





  - 大サイズファイルが添付されたメールを大量に受信
    - 全体の負荷が上昇し、フル稼働状態
    - 一方、暗号化処理が追いつかずロードアベレージだけが上がり続け相関が崩れた
  - サービスレベルの低下を検出
    - 影響を最小限にできていたはず

うまくいかないケース

- 相関が大きく崩れていても障害になっていない
  - 従来の閾値監視では、サービススペックに関わるデータを直接監視
  - 必ずしも「相関の崩れた＝障害発生」となるわけではない
  - 現在対策を検討中

今後の取り組み

- 障害原因の提示
  - 現段階では相関が崩れた場所を知らせるのみ
  - 障害原因を推測するシステムに
    - 「相関の崩れ方」と「障害原因」をセットで学習
    - パターンマッチによって障害原因を提示
    - 誤検知抑制にも効果が期待できる

まとめ

- 負荷の相関関係を用いる
  - 個々のリソースを監視するのではない
  - データの組み合わせから不変関係をモデル化
- 障害の検知に成功
  - 過負荷による再起動の兆候
  - 局所的な負荷増加によるメール配送遅延
- 課題
  - 誤判定の抑制
- 今後の取り組み
  - 障害原因の学習と提示