認証基盤の冗長化　技術編

学認CAMP 2013 @千葉大学,
2013-09-11,　国立情報学研究所　西村　健

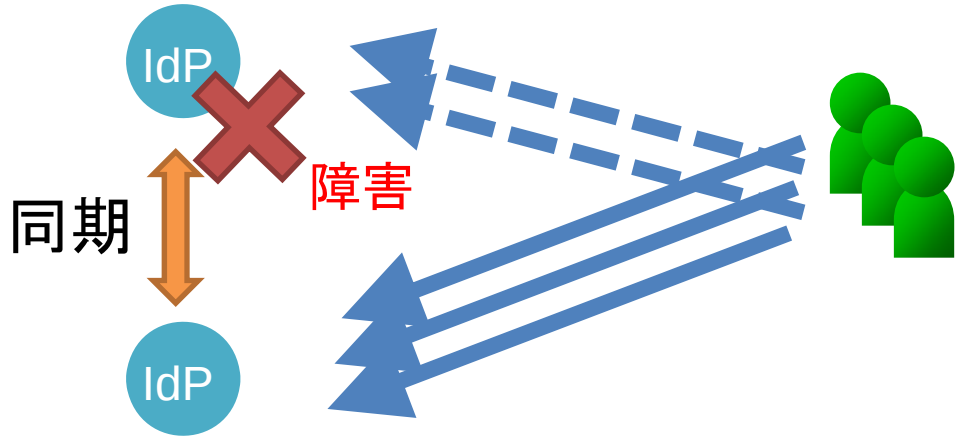
認証基盤冗長化のために

冗長化（クラスタリング）:

- システムを複数ノードで起動しておく

↓

- 1つのノードが故障等で停止してもサービスを続けられる
  →問題なく続けるためにはデータの同期が必要



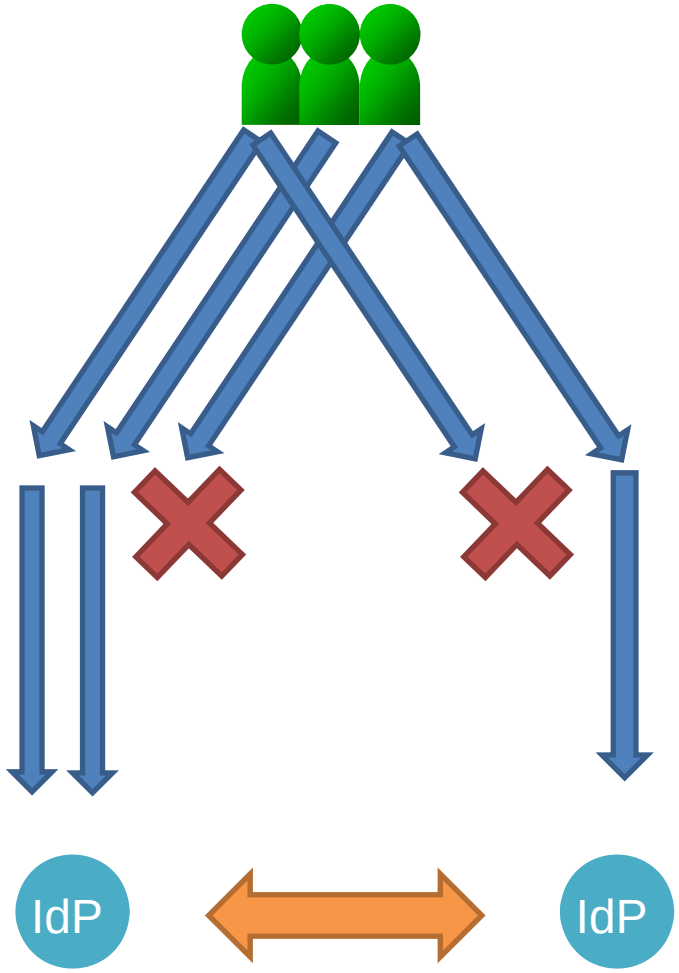
※特殊なケースを諦める（再度認証してもらう）のも重要

認証基盤の分解

- (A) Shibboleth IdP
  - 以下を含めて
    - StoredIDのためのDB
    - uApprove.jpのためのDB
- (B) LDAP or DBなど
  - いわゆるバックエンド

冗長化技術の分解

- ①振り分け
- ②固定
  - ユーザ毎に特定の
    ノードに固定するとか
- ③同期

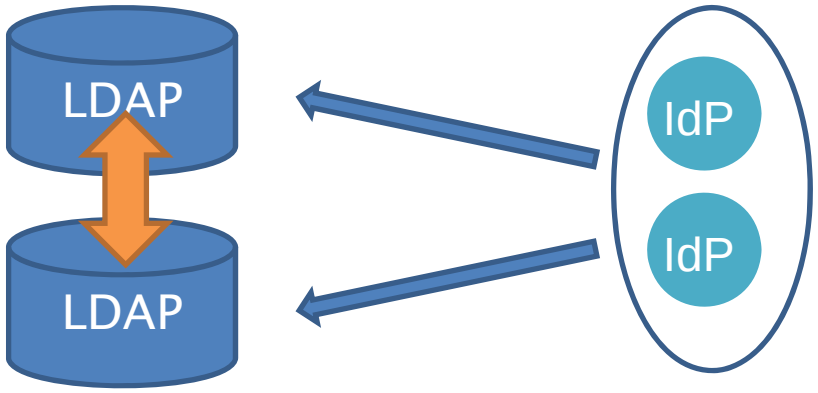


検討マトリクス…

	(A)IdP	(B)LDAP
①振り分け	？	？
②固定	？	？
③同期	？	？

「(B)LDAP」の冗長化技術

- 既存技術でカバーされると思うのでばっさり省略
  （実際に運用している方から情報をいただければありがたいです）
  - 「(A)IdP」内のDBの冗長化についても既存技術で
- このあと紹介する各種技術とは独立



③同期　①振り分け　（②固定 は不要）

検討マトリクス…

	(A)IdP	(B)LDAP
①振り分け	?	OK
②固定	?	OK
③同期	?	OK

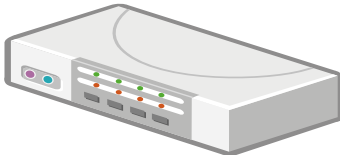
「(A)IdP」の「①振り分け」「②固定」の解決

「①振り分け」について：

- ネットワークプロトコルレベルで解決
  - VRRP
  - Anycast
- ロードバランサ

「②固定」について：

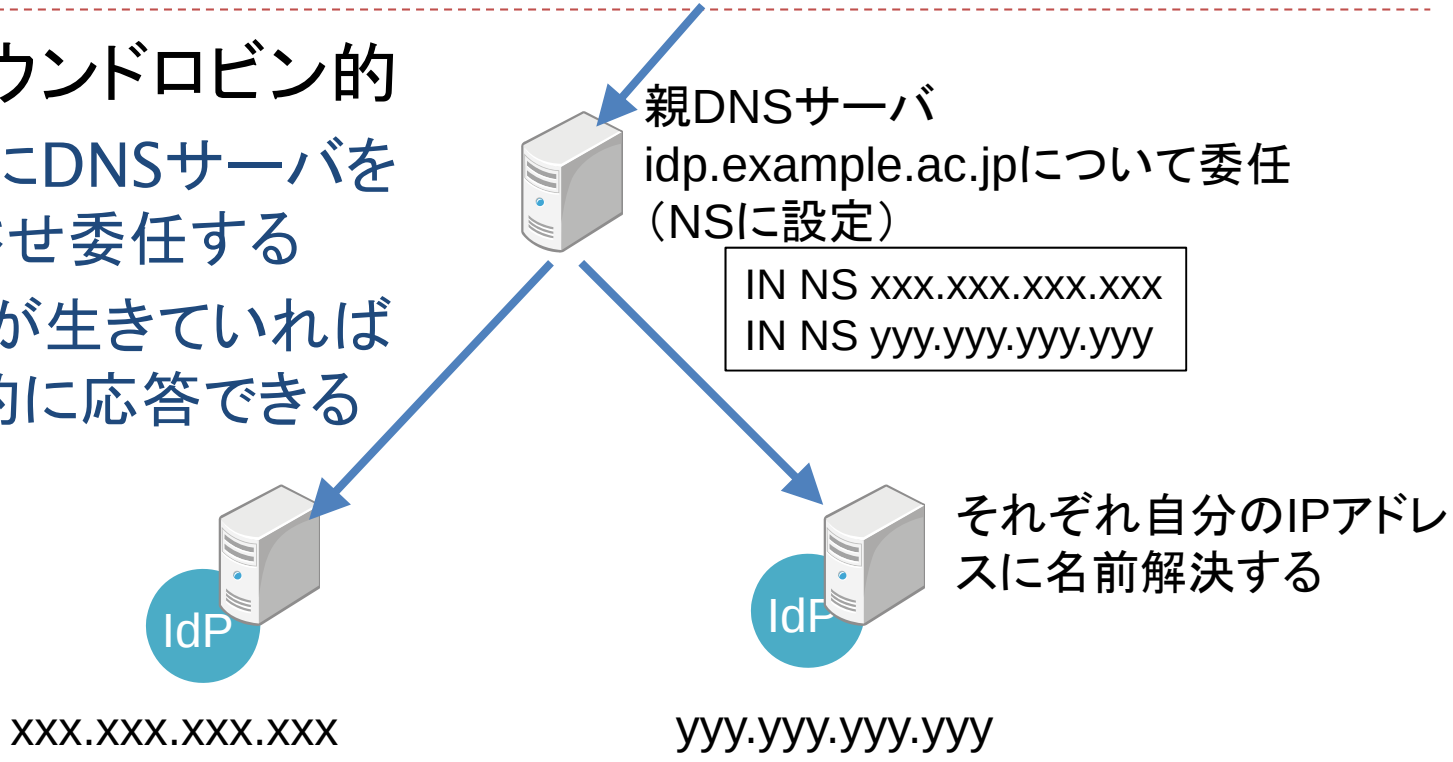
- Active-Standby構成
- ロードバランサを入れる
  - sticky機能でcookie情報等を元に特定のノードへの紐付けが可能
  - そもそもShibbolethは
    ハードウェアロードバランサ＋Terracotta推奨（情報が古いかも）



ただし，（ハードウェアのものは特に）高い！

「①振り分け」についてDNSを冗長化に用いる簡易な方法

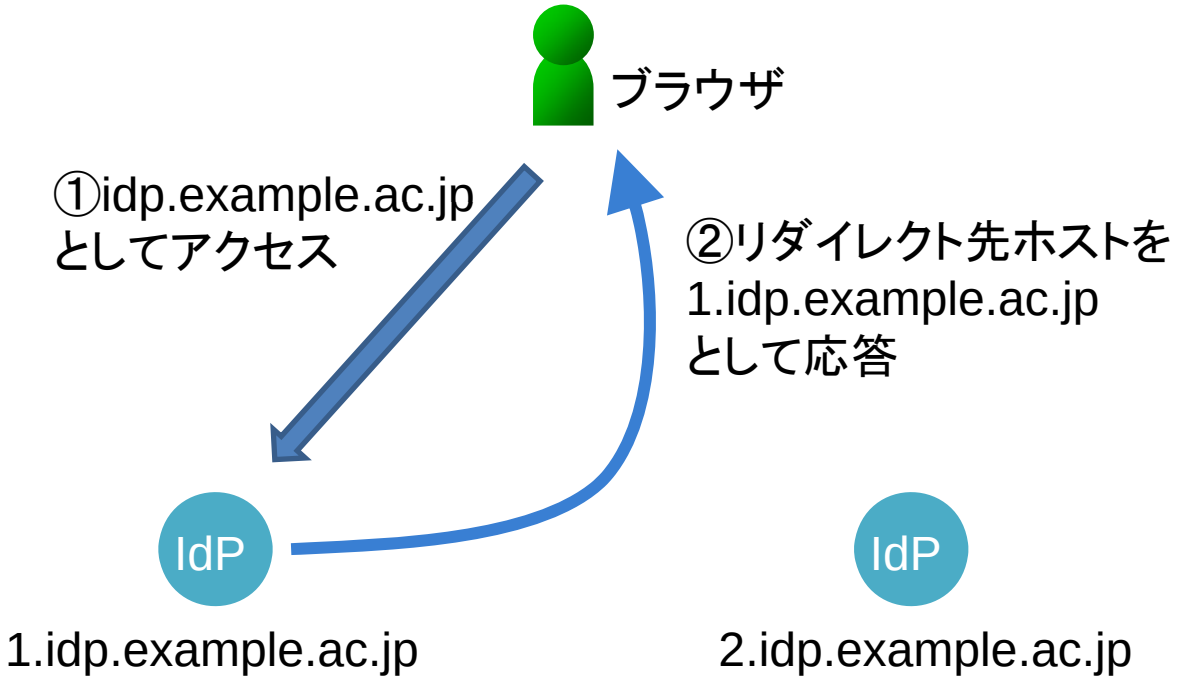
- DNSラウンドロビン的
  - ノードにDNSサーバを同居させ委任する
  - ノードが生きていればDNS的に応答できる



- 詳細はmeatwiki.nii.ac.jp GakuNinShare の DNSの委任(delegation)を用いたDSの広域分散 にて

「②固定」についてノード別ホスト名を用いた簡易な方法

- 同一ブラウザが一度アクセスしたノードにアクセスするように



- 詳細はmeatwiki.nii.ac.jp GakuNinShare の IdPクラスタ向けセッション固定方式 – Sticky Login にて

検討マトリクス…

	(A)IdP	(B)LDAP
①振り分け	OK	OK
②固定	OK	OK
③同期	？	OK

「(A)IdP」の「③同期」の問題 –
IdPで同期すべき情報には2種類ある

- ユーザが認証済みという情報
  - ログイン後
  - 長期間存在する（～8時間）

- ログイン処理中のみ存在する情報
  - ログイン画面を出して，
ID/パスワードを送信するまでの間
  - 両者が別のノードだったらどうなる？
  - 短期間

IdPの3つの冗長化方式

- 昔からの定番
  - Terracotta方式
- 最近登場
  - Stateless Clustering方式
  - repcached(memcached)方式
  - （DB方式）…現在調査中

Terracotta

Tomcatの汎用的な冗長化ソフトウェア

特長:

- 完全な冗長化
  - 全てのデータを(JavaのObject単位で)レプリケートする
  - 「②固定」を考えなくても良い

デメリット:

- 複雑
- 相性問題(バージョン間の差異が問題に)
- Java SE 7は非サポート?

Stateless Clustering

osuidpextプラグイン使用

- レプリケートすべきデータをcookieやID(transientID)に埋め込む！
- 軽い！
- ノード間の通信がゼロ
  - 同一ネットワークセグメントである必要がない

- 不完全な冗長化
  - 短期間情報について「②固定」を考慮する必要
- IdP機能制約あり　ー　学認が規定する範囲では問題なし
  - Single Log-Out(SLO)できないとか
- IDが想定外に長くなることによりSP側で誤動作の可能性
  - 例: Web of Knowledge
    - SAML1でもフロントチャネルを使うことで回避(includeAttributeStatement="true")

repcached(memcached)

Memcached StorageServiceプラグイン使用

- レプリケートすべきデータをmemcachedに置く
  - memcachedをDBとして用いる
  - repcachedはmemcachedをレプリケート対応したもの
- ある程度軽い
- cookie/IDなど、外部に対するデータ形式は変更しない

デメリット:

- 不完全な冗長化
  - 短期間情報について「②固定」を考慮する必要
- ノード間通信が少し

ちょっとだけ性能比較

	MeanTestTime(ms)	TPS（トランザクション/s）
Terracotta	1235.99	15.22
Stateless	398.61	47.20
repcached	494.44	38.89

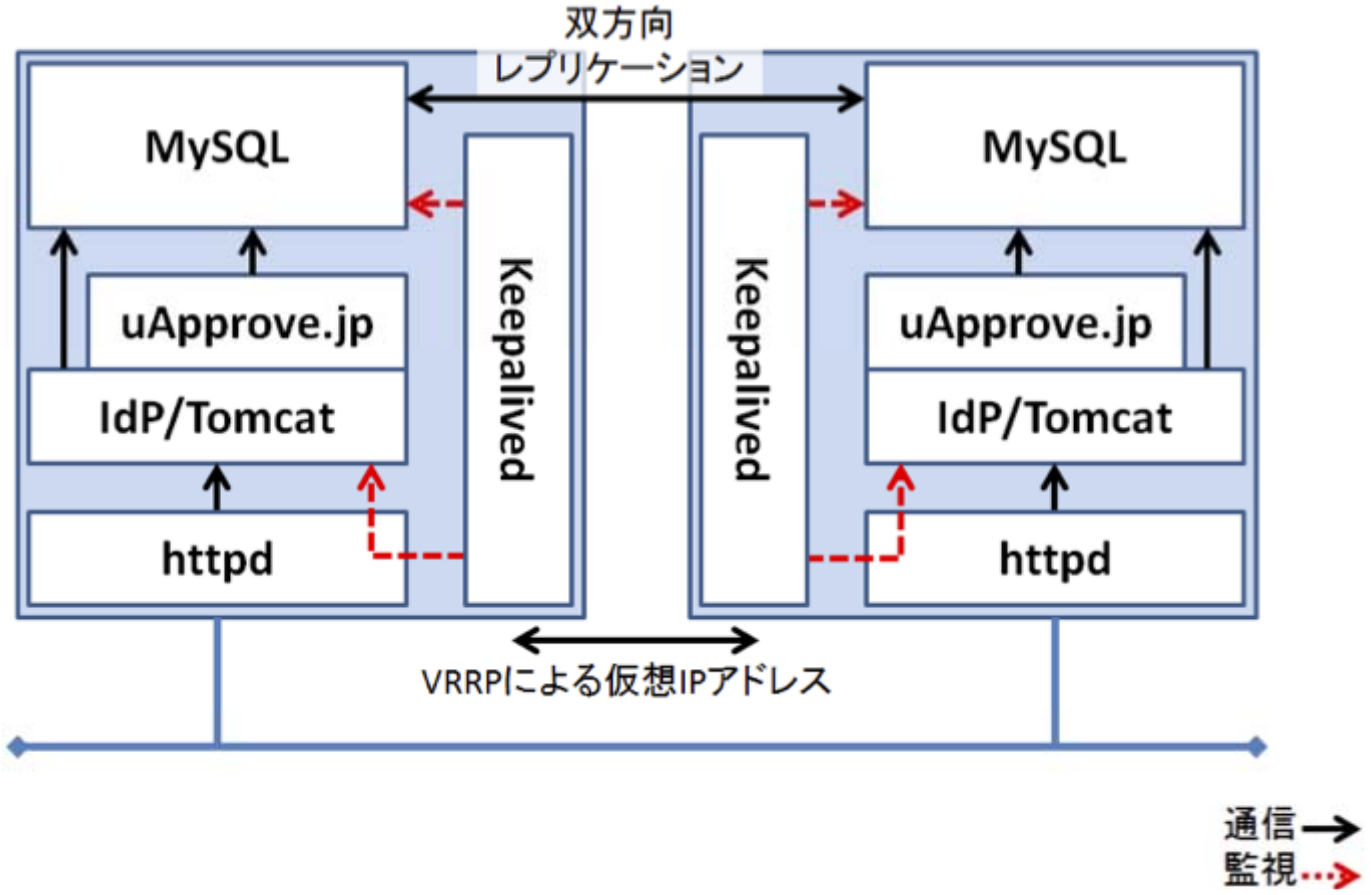
- あまり最適化してませんが…
- statelessはCryptoTransientID無効での計測
- 詳細は後述のウェブページ参照

NII IdPでの適用例

- 2012年から冗長構成で運用中
- 採用した技術
  - VRRP + Keepalived
  - Stateless Clustering
  - StoredIDやuApprove.jpはDBとしてMySQLを使用
  - MySQLマルチマスター構成
  - 上記を2ノードで運用
  - LDAPは別途Active Directoryのクラスタ構成

NII IdPの構成図

※冗長構成のActive Directoryは除きます



まとめ

- 今選択するならstatelessかrepcachedか？
  - 最も単純　ー　Stateless Clustering
    - 機能的に制限あり
  - 中間的　ー　repcached
  - 最も複雑　ー　Terracotta
    - 将来的なサポートが不安

※バックエンドの冗長化も忘れずに

IdP冗長化のための資料

- Shibboleth-IdP冗長化環境構築手順書(Stateless Clustering編)
- Shibboleth-IdP冗長化環境構築手順書(memcached編)
  - repcachedは最後にちょこっと
- Shibboleth-IdP冗長化環境構築手順書(Terracotta編)

金沢大学提供

  - 学認Webサイト ＞ 技術ガイド ＞ IdPカスタマイズ　の「ロードバランサー配下のシボレスIdP環境設定に関する検証実験」の項も参照のこと.

- Shibboleth-IdP冗長化パフォーマンス比較試験報告書
- IdPクラスタ向けセッション固定方式 - Sticky Login
- DNSの委任(delegation)を用いたDSの広域分散

いずれもmeatwiki.nii.ac.jpの

GakuNinShare ＞ 技術情報

以下にて公開しています.