リクルートにおける
検索エンジンSolrの活用

中野 猛
tf0054@r.recruit.co.jp

システム基盤推進室 / FIT / **株式会社リクルート**

まだ、ここにない、出会い。

アジェンダ

01) リクルートの紹介
02) Solrとは(概要と利用イメージ)
03) 利用サイト紹介(Solr利用サイト)
04) 採用した理由(ドリルダウンでSolr!)
05) 採用までの道のり
06) 利用の詳細(インフラ)
07) 利用の詳細(アプリ/開発容易化の施策)
08) 今後の展望(大規模/v1.3)

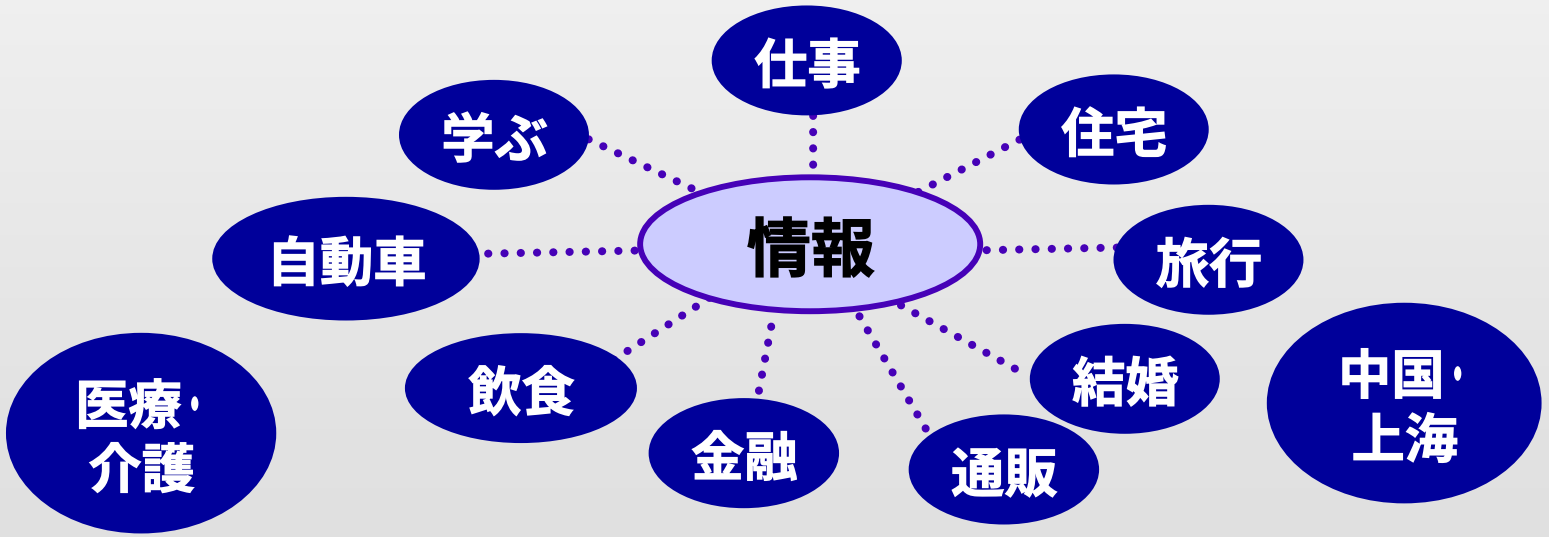
リクルートのご紹介

リクルートの会社概要

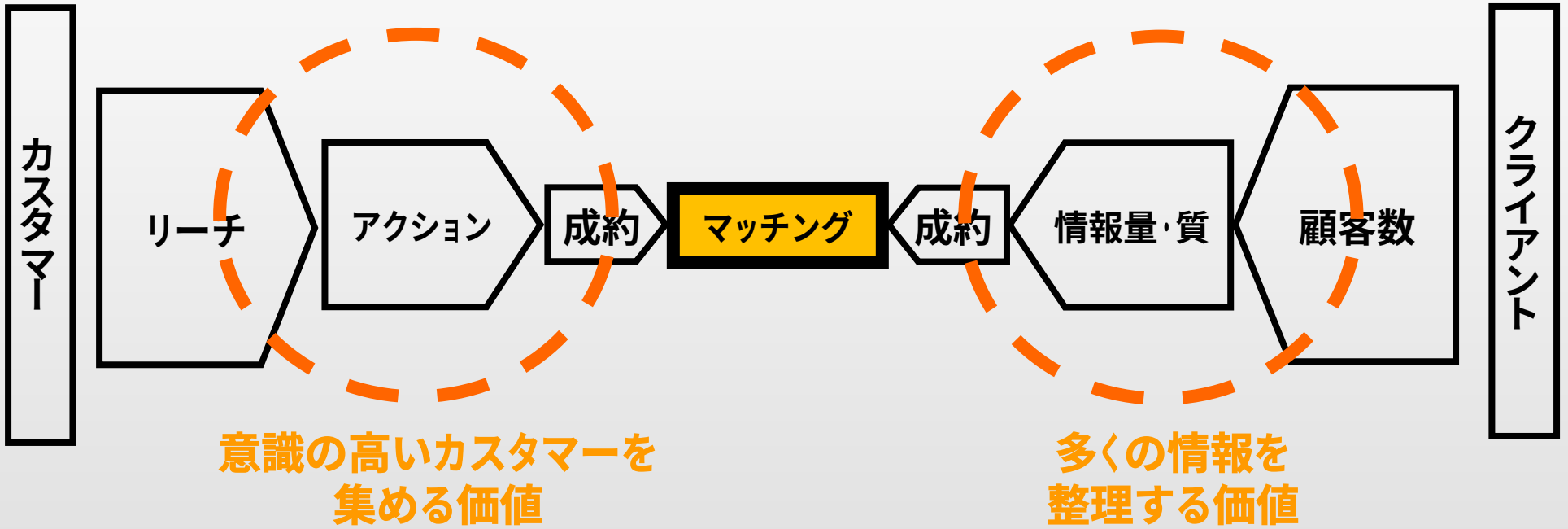
- 創業　昭和35年3月31日
- 資本金　30億264万
- 売上高　4,436億円 (2006年3月期)
- 営業利益　1,297億円 (2006年3月期)
- 従業員数　6,298名
- 平均年令　30.8歳

リクルートの事業概要

- 人材総合サービス(HR)
  - 求人情報サービス、人材斡旋サービス、個人のキャリア形成支援サービス、人材派遣etc.
- 商品とカスタマーとマッチングサービス(販促系)
  - 学び／住宅／旅行／ブライダル／出産育児／自動車におけるマッチングサービスetc.



リクルートのビジネスモデル



FITとは

- Federation of IT :全社横断の情報システム部門

FIT

- スタッフ — ITマネジメント、支援機能
- センター — 社内基幹システム、PC/NW
- システム基盤推進室 — ウェブ系インフラの設計構築/保守運用
  ミドルウェアの保守運用、OSSの調査検討
  アプリケーション設計/開発、及びその効率化促進
- 事業担当1 ～ 事業担当n — 各事業担当 (オフィスは各事業と同じフロアに分散)
  - ・事業単位のIT戦略、システム化計画の立案
  - ・開発プロジェクト
  - ・システムマネジメント(システム改善＋保守運用)

※ 情報子会社なし
約170名(06年4月現在)

自己紹介

- 中野猛(ナカノタケシ)
  - 2000年入社
- これまでの仕事内容
  - 新ISIZEインフラ設計/構築/保守運用
  - 某新規事業の開発&保守運用
  - R25ウェブサイトの開発＆保守運用
- OSS含む新技術の調査＆社内導入推進
  - PostgreSQL,Solr,ZABBIX,Introscope,etc.
- 次期ウェブサイト設計/開発基盤の検討
  - 画面/アプリ/インフラを通して全体で最適化!

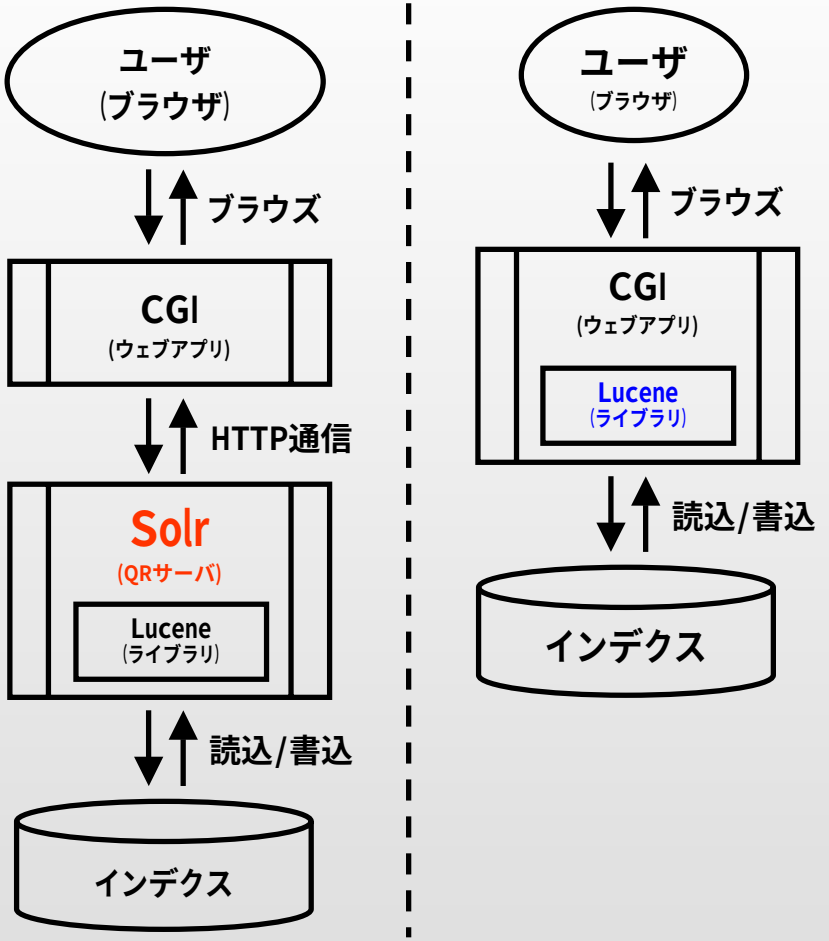
検索エンジンSolr

Solrとは

- OSSの検索エンジンミドルウェア
  - サーバアプリケーション
  - CNET社内製造後オープンソースに
  - apache.orgのTOPドメインプロジェクト
- Solrの良いところ
  - ファセット機能 (ダイナミックドリルダウン!)
  - 結果並び順など挙動の解明可能 (OSS故)
  - 必要なHWが小規模で安価 (検索機能に特化)
- Solrの弱いところ
  - ロボット等でのクロール取込み
  - 検索語の集計や分析

Solrとは

- 形態
  - Tomcatにデプロイ
  - XML@HTTPでI/F
  - コアライブラリはLucene
- サーバアプリ
  - 開発工数の削減
  - 維持運用者の機能分離
  - 障害切り分けに有利



Solrとその他検索エンジン

- オープンソースの検索エンジンは数多存在
  - 独自にI/O機能を有する(DB活用型/サーバ型)
  - ライブラリでCGI等プログラムに組込む(組込み型)
    - DB活用型
      - Tritonn
      - Luida

      など
    - サーバ型
      - Solr
      - JiroSearch

      など
    - 組込み型
      - Hyper Estraier
      - Namazu
      - Lucene

      など
- Solrのその他ポイント
  - DiggやInternetArciveでの利用実績
  - 拡張可能なAPIが多く組込まれた構造
  - ファセット機能を考慮したキャッシュを持つ

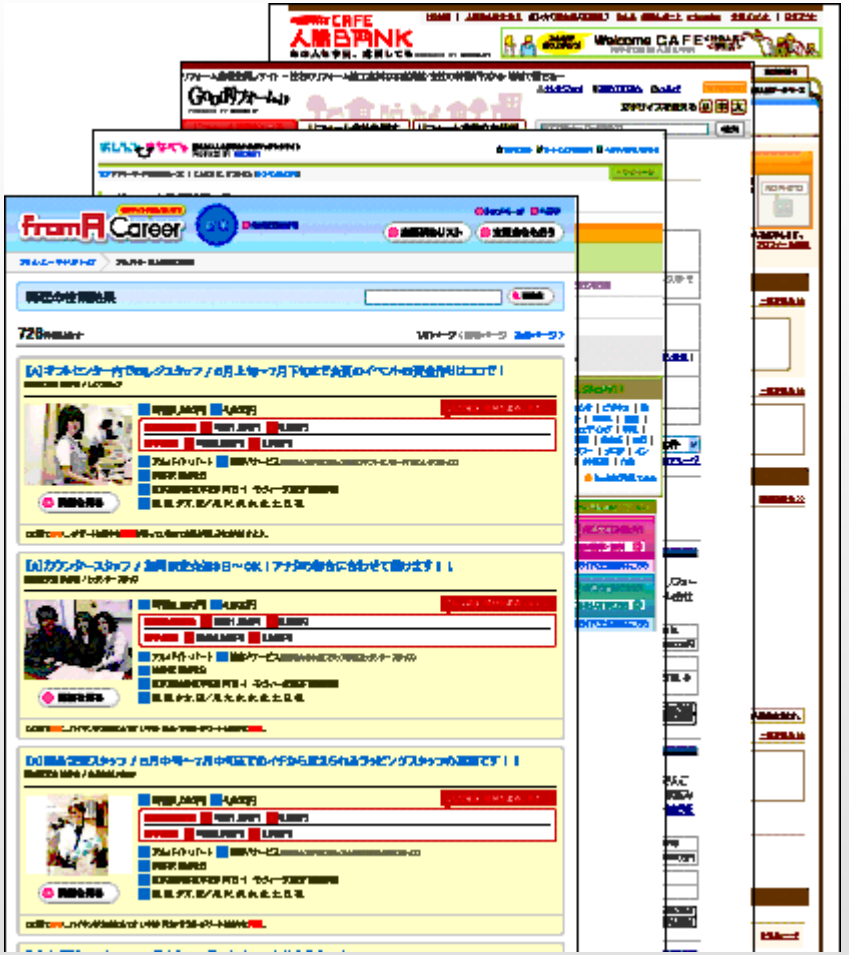
利用サイトのご紹介

Solrを活用したサイト例

- 人脈バンク
- グッドリフォーム
- おしえるまなべる
- FromAキャリア

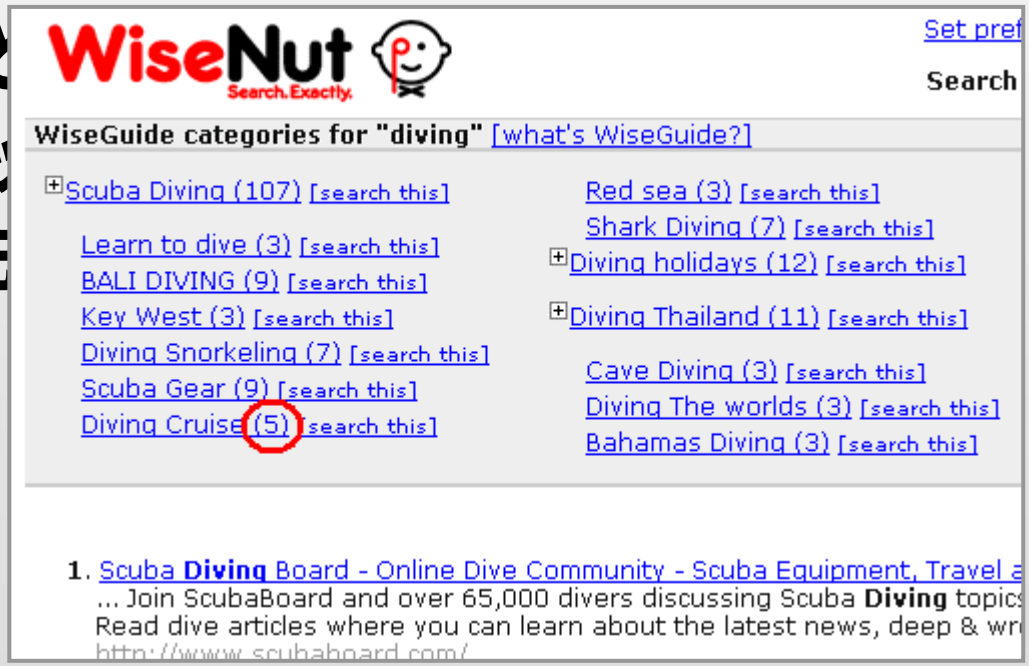
＋

- 2サイト開発中…



Solrを選んだ理由

- 弊社で必要とされているのは
  - 日本語キーワードでの検索機能よりも
    主に欲しい機能はファセット
  - 商用ミドルの載せ換えも考えると
    サーバ型が有利
- ファセットと
  - ダイナミッ
  - 情報を先



Solrを使う前は…

- ファセットの種類
  - 特定カラムをユニークに集計
  - 特定カラムを自由に集計("1000円台の商品"など))
- これをDBでやると大変…
  - 前者はgroup-by
  - 件数は毎回条件付でcount()する

リソース負荷が高い
もしくは集計テーブルを自前でメンテ

- Solrではインデクス時にある程度**事前計算**
  - 単語一覧　＞インデクスに一覧表を持つ
  - 件数カウント＞インデクスにまま入っている

利用開始までの道のり

これまでの流れ

- ステージ
  - 2006年度4Q　調査/検証を開始
  - 2007年度1Q　準備を開始
  - 2007年度2Q　案件へ適用
  - ∫
  - 2008年度1Q　大規模対応を検討

これまでの流れ

- ステージ
  - 2006年度4Q　調査/検証を開始
  - 2007年度1Q　準備を開始
  - 2007年度2Q　案件へ適用
  - ～
  - 2008年度1Q　大規模対応を検討

これまでの流れ

- ステージ
  - 2006年度4Q　調査/検証を開始
    - 複数の商用検索ミドルウェアが稼動
      - ノウハウや人的リソースが分散
      - 包括契約ではなくサポートコストがうなぎ上り
    - 社内各所でLuceneを調査(書籍が出た為)
      - ノウハウや人的リソースを集約
      - 社外リソースの開拓と一本化
  - 2007年度1Q　準備を開始
  - 2007年度2Q　案件へ適用
  - ∫
  - 2008年度1Q　大規模対応を検討

調査/検証

- 日本語での利用
  - 変更無くとも動きはする
    - Solrチームの牧野勝氏の記事参照(Software Design 2007/12)
  - 文章の単語分割が必要
    - 形態素(MeCabのJNIバインディングを利用)

我輩は猫である	=	我輩	は	猫	で	ある

    - N-gram(Solrの標準ライブラリで対応可)

我輩は猫である	=	我輩	輩は	は猫	猫で	であ	ある

  - どちらが良いも言えない
    - 前者は精度は高いがモレも在る(⇔後者はその逆)

調査/検証

- 調査
  - 構造解析
    - SolrのクラスやLuceneインデクスの構造
  - 負荷試験
    - 800万件程度までテストし商用エンジンと比較
- その他
  - 運用監視
    - 監視すべき項目を検討(キャッシュヒット率他)
      - 管理画面をパースして監視アプリへ流すことに
    - 運用が必要な項目を想定
      - 単語追加や順位等の問い合わせ(後者は想定ほど無し)

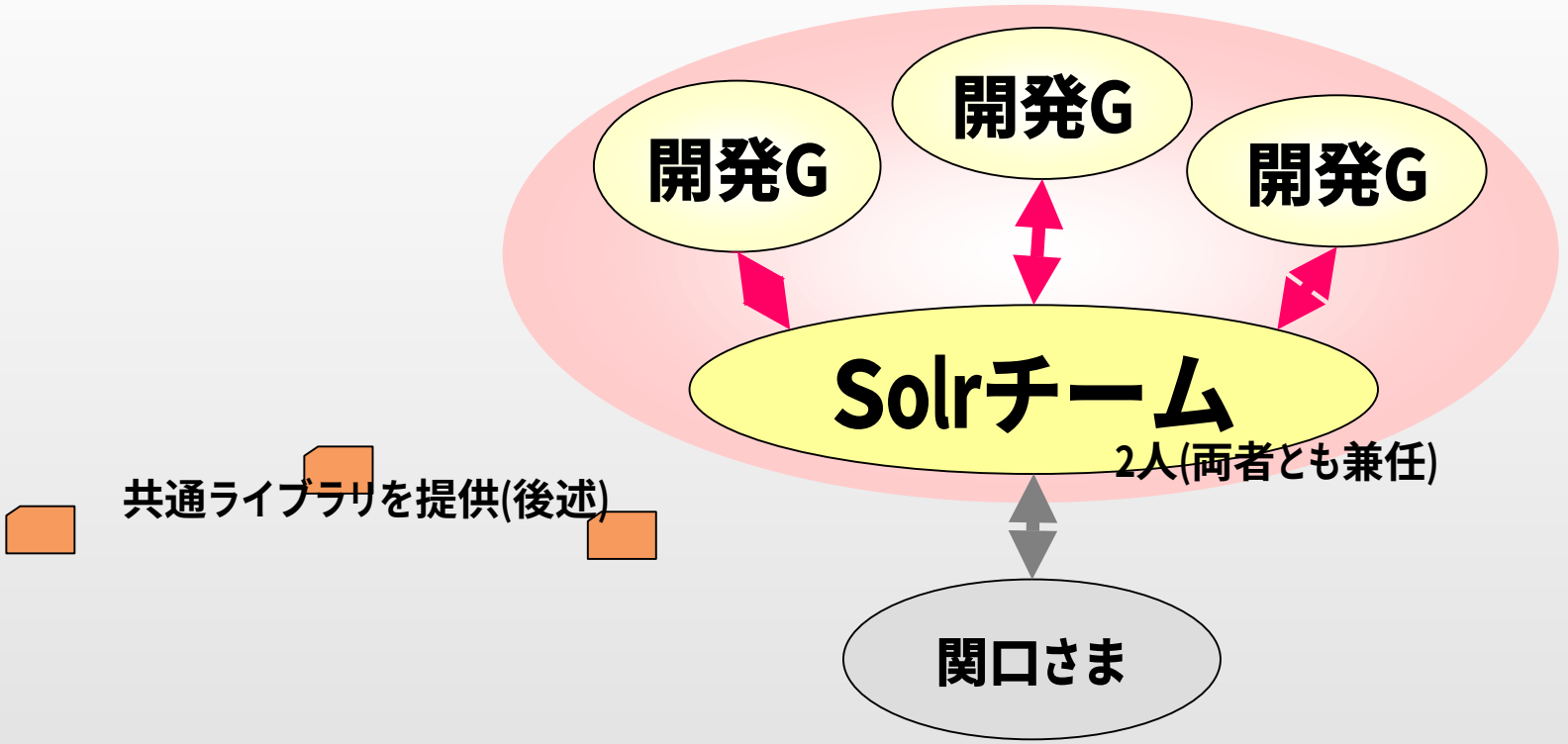
利用の促進

- 商品掲載サイト
  - じゃらんに代表されるサイト
  - 要求SLは極めて高い
- 集客に注力するサイト
  - 広告掲載などに代表されるサイト
  - 要求SLの許容幅が比較的大きい(インプレッション保障故)
- 後者から始めて徐々に展開
  - 結果的にはさっそく商品掲載サイトへ
  - (Solrに起因した)サイト停止は無し
  - (今期)更なる大規模サイトへ展開予定

利用の詳細

Solrの利用推進体制(社内)

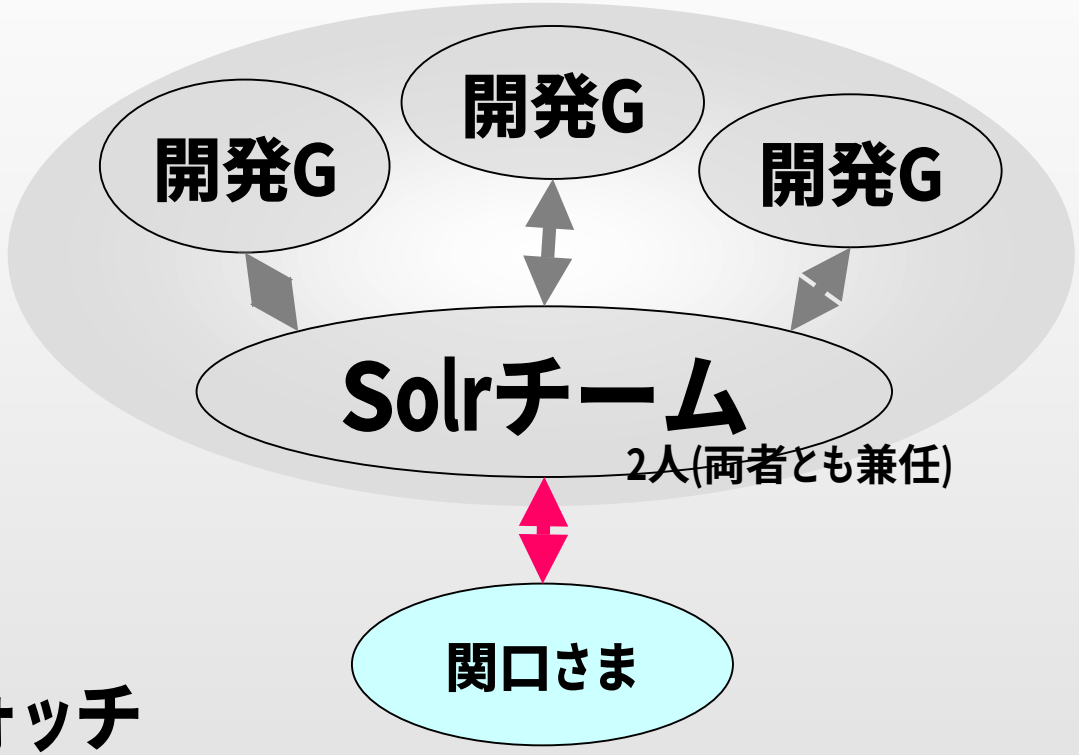
- OSSに強いメンバを調査時からアサイン



- この体制のまま開発サポートチームへシフト
- これにより開発利用がスムーズに立ち上げ可能に

Solrの利用推進体制(社外)

- Solrの構造解析をメインに調査



- コミュニティのウォッチ
  - 新規公開パッチの機能紹介(1個/週程度)
- 不明挙動のQA/調査/パッチ作成
  - 実績では月1件あるかないか程度

アプリの開発サポート(スキーマ)

- 1インスタンスは1スキーマで固定
  - DBで考えると1つのTBL構造に固定される
  - Solrには設定XMLファイル内に記載して渡す

#	カラム名	タイプ	単語分解	データ格納
0	ID	Int	-	×
1	NAME	String	MeCab	○
2	PRICE	SortableInt	-	×
3	CATEGORY1	String	2-gram	×
4	CATEGORY2	String	-	×
5	…	…	…	…

  - 元データが複数ある場合は汎用的に作る必要有
  - 各カラムによって単語分解方法を変更可能
    - 商品名など検索モレが許されないところはN-gram等

アプリの開発サポート(ワークシート)

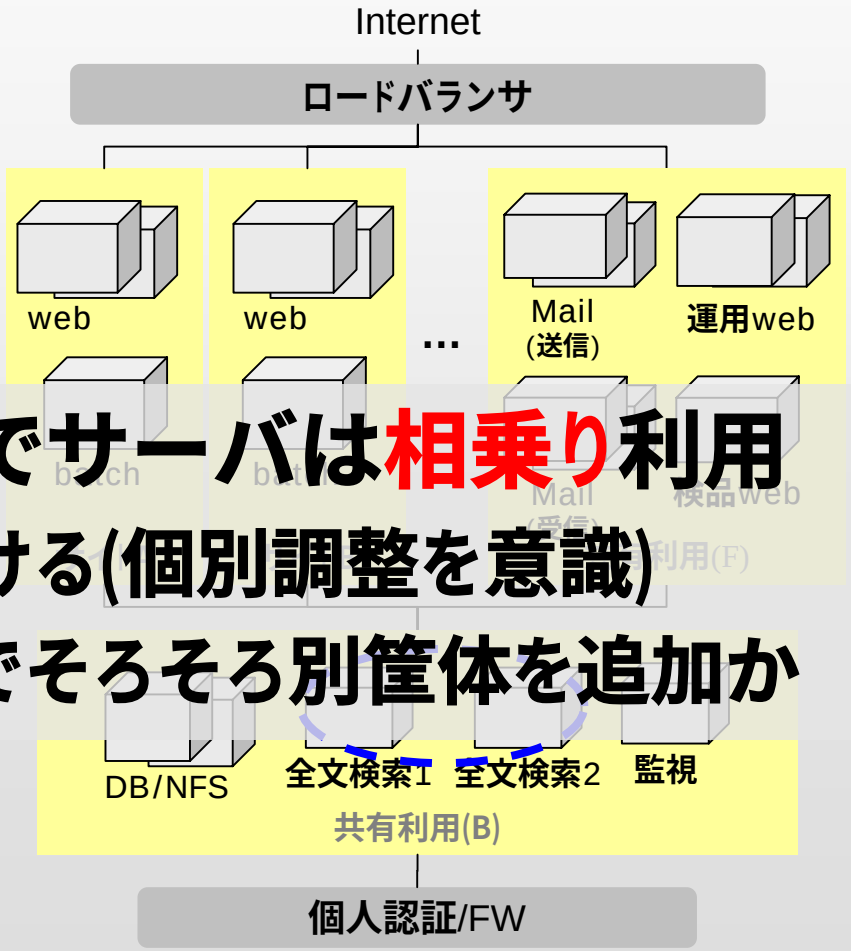
- スキーマ設計にはワークシートを準備
  - 開発者のスキルはまちまち
    - スキーマはサイト開発者が作成
    - Solrチームがレビュー(ワークシートで会話)
  - エクセルにから直接設定(XML)を生成
    - 設定ファイル作成の手間を軽減

アプリの開発サポート(ライブラリ)

- 共通ライブラリを準備(Java)
  - 文書の投入や検索が可能
    - HTTPで通信していることを意識させない
    - 2つのSolrを使い分け可能(冗長化)
  - コミュニティ標準(?)のSolrJをカスタマイズ
    - エラー処理などを社内ライブラリ向けに
  - 結果をオブジェクトにマッピング
    - 開発者に優しい

インフラは相乗り設備

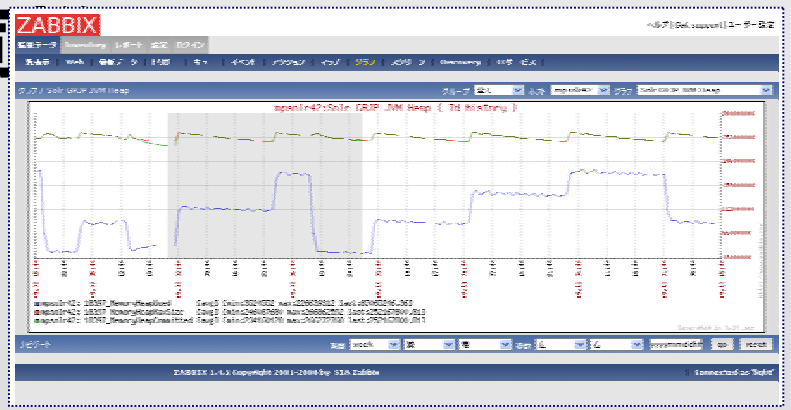
- Solrには個別にサーバ筐体を割当て
  - 「2-Xeon+4G+RAID1」程度



- 複数のサイトでサーバは相乗り利用
  - Tomcatで分ける(個別調整を意識)
  - 現在4サイトでそろそろ別筐体を追加か

インフラでの監視と運用

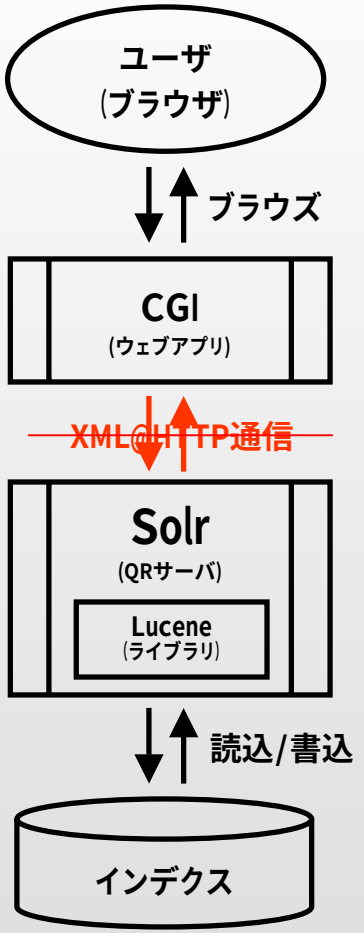
- 監視
  - Solrは管理画面をパースして流し込み(前述)
    - 最近JMXで取れるよう機能追加がなされた
  - 監視/モニタツールZABBIXにデータを集約
    - オープンソースのツール
    - DBデータ監視等と同じ環境で統合モニタリング
- 運用
  - 通常のJVM/Tomcatアプリと同
    - 初期導入や簡単な設定調整
    - セキュリティパッチの適用



今後の展開

リクルート内部から

- 利用の拡大
  - 高アクセス/大規模サイト
    - Solrの機能的キャパより人材
  - 既存サイトへの検索アドオン
    - 自サイトのクロール要件もあり
    - アプリ構造がシンプルに
- 新版ライブラリのリリース
  - 問い合わせ数が増え高負荷に
    - XMLパースは大変＞バイナリへ変更
      - プラグイン的に結果返却機能を差換え可
    - ハイライトは大変＞検討中(n-gram?)
      - MeCabでJNIも調整次第か



Solrとして

- 新版(v1.3)がもうすぐ
  - 分散ノードでの串刺し検索
  - 文書数では現行(v1.2)でも1億強＝1インスタンス
- DBと連携
  - ThinkITに関口さんの記事(在庫が無い>順位を下げる等)
  - DB連携型の開発容易性を取り込む
- 評判検索
  - 企画ありきで世に問う
  - 技術ドリブンでのトライ