SB Intuitions

独自日本語Webコーパス構築のための巡回クロール基盤構築

SB Intuitions株式会社

石原慧人

背景: LLM向けの事前学習Webコーパス

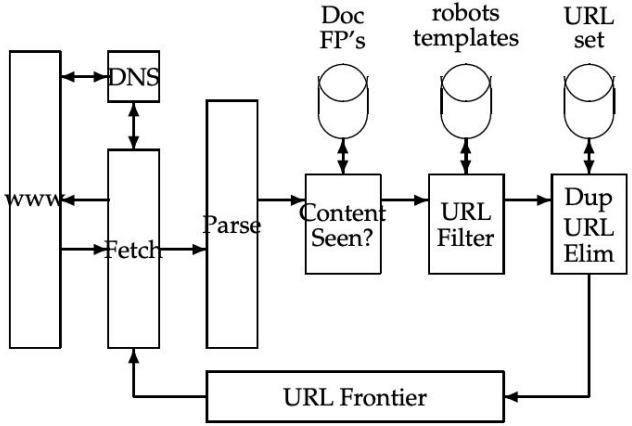
SB Intuitions

- LLM 開発における日本語コーパス
  - LLMの性能は学習データの質と量に大きく依存
  - 日本語ＬＬＭの事前学習時では Web テキストデータをベースとして日本語用の前処理を加えて作成した日本語コーパスが用いられることが多い
- CommonCrawlについて
  - 既存研究では最大級のオープンクロールデータとして知られる**CommonCrawl**がベースとして用いられていることが多い
  - ただしCommonCrawlの日本語比率は5%程度と限られている
  - 1ドメインに対して取得するURL数に制限があり、特に有用なサイトが掘りきれていない可能性が高い

→ **独自にクロールすることで更なる有用な日本語データの獲得が期待できる**

背景: クローラー概念図

- **クローラー**とは、インターネット上のWebページを自動で次々と巡回し、情報を収集するプログラムのことをいいます
- 下の図のような**ページの取得・解析**と**新規URLのキュー追加**をループさせる構造が一般的です

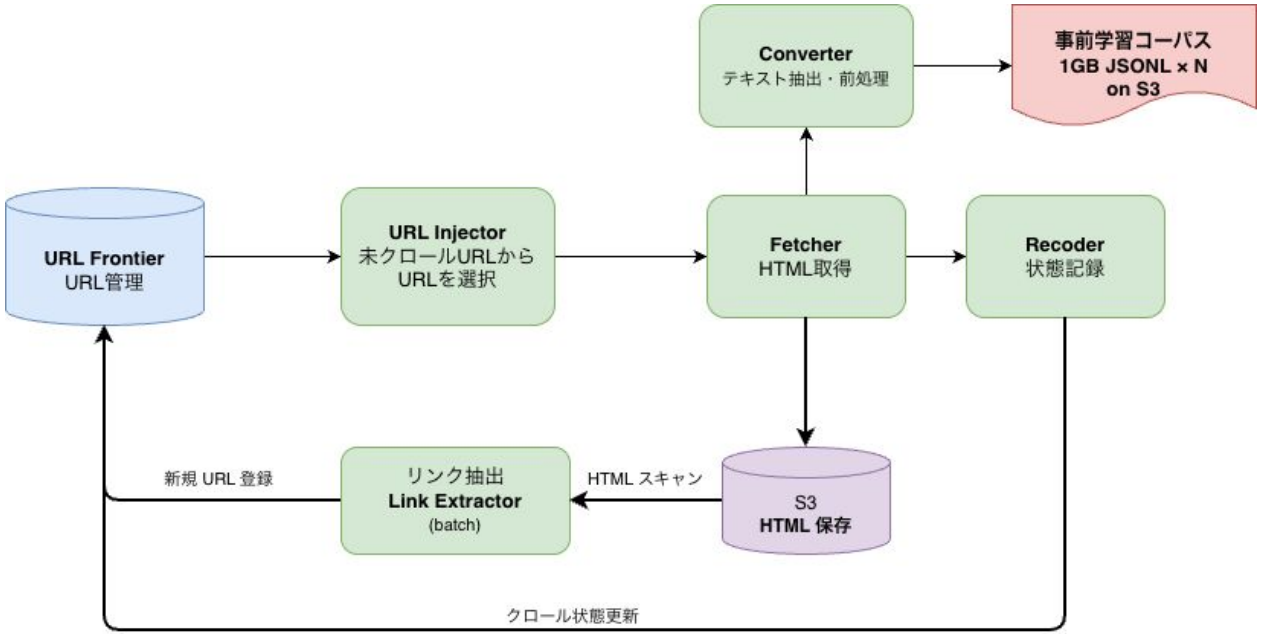


出典：Manning et al., Introduction to Information Retrieval (2008), Figure 20.2 (https://nlp.stanford.edu/IR-book/pdf/20crawl.pdf)

クローラーシステム全体像

SB Intuitions

- 複数のSlurmジョブを連携させたシステムとして実装
- ジョブ間のデータ連携は**Valkey Streams**で非同期接続



システム要件

要件	詳細
規模	デイリー1千万URL目標
品質	テキスト抽出・言語判定
倫理性	robots.txt遵守、ドメインレートリミット、学習に利用できないドメイン除去
耐障害性	多少のロストは取り直せば良いので許容
環境	HPCクラスタ(Slurm)+AWS

- インフラ
  - 社内環境の都合上HPCクラスタが主な実行環境となる
  - 必要に応じてAWSも使用

SB Intuitions

常駐サービスを前提としない世界

観点	クラウド（K8s / ECS）	HPC（Slurm）
プロセスモデル	常駐コンテナ（自動再起動）	バッチジョブ（実行時間制限あり）
ヘルスチェック	Liveness / Readiness Probe	なし（自前で実装が必要）
スケーリング	HPA / Auto Scaling Group	Array Job のタスク数を手動指定
障害復旧	Pod 再起動・Deployment ロールバック	ジョブ再投入を自前で管理
停止制御	SIGTERM + preStop Hook	scancel → 再投入抑制が必要
コンテナ	Docker / containerd（デーモン常駐）	Enroot / Pyxis（rootless、デーモン不要）
オーケストレーション	宣言的（YAML マニフェスト）	命令的（sbatch スクリプト）
ログ	CloudWatch / Fluentd 等	ファイルベース、自前ローテーション

Slurmは本来計算ジョブを投入して結果を待つための仕組みであるため
常駐型の分散サービスを動かすには追加の工夫が必要

Slurm上での工夫

SB Intuitions

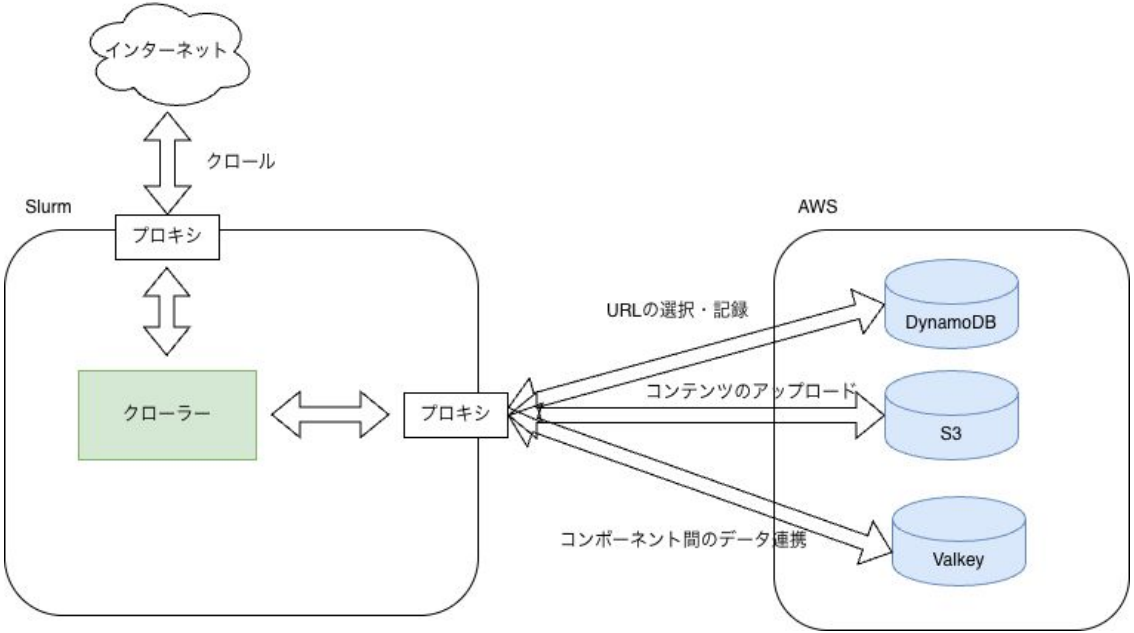
k8sが担う機能をslurm上で代替

- **Watchdogスクリプト（≒ Liveness Probe）**
  - 60秒ごとにsqueueコマンドでジョブ生存確認
  - 欠落ワーカーだけsbatchで回復
- **定期実行（≒ CronJob）**
  - sbatch --begin="now+${INTERVAL}"
- **統合管理スクリプト（≒ kubectl）**
  - **起動/停止:** crawler_ctl.sh start/stop
  - **ステータス確認:** crawler_ctl.sh status
- **Array Job（≒ ReplicaSet）**
  - sbatch --array=0-14 fetcher.slurm

外部環境との接続

SB Intuitions

- **Slurm (HPC) + AWSのハイブリッド構成**



インフラ選定

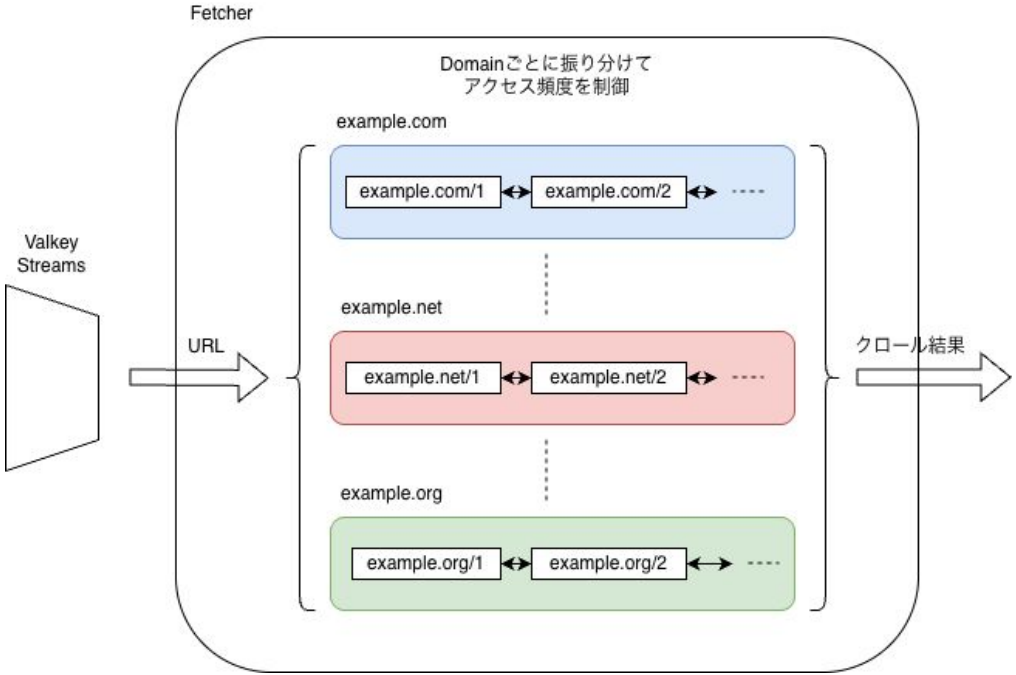
SB Intuitions

- **URLFrontier: DynamoDB**
  - 保有するURLとクロール済みかのフラグなどを管理するデータベース
  - 当初はKVSで良いと考えていたが、クロール対象URLを抽出するクエリの負荷が支配的
  - GSIで暫定解決したがDBの移行も検討中
- **メッセージキュー: Valkey Streams**
  - 複数のコンポーネントを適切に協働させるために各コンポーネント間にメッセージキューを挟んでいる
  - AWSにはSQSなど専用のメッセージキューサービスも存在するが、キャッシュの保存などの需要もあったのでRedis互換のDBであるValkeyを採用

コンポーネント詳細: Fetcher

SB Intuitions

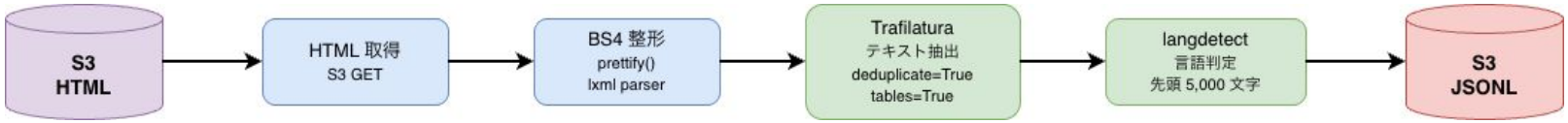
- 同一のDomainに対して過度なアクセスが行われないようFetcher内で適切にアクセス頻度をコントロールする



コンポーネント詳細: Converterによるテキスト抽出

SB Intuitions

- **HTML→事前学習用テキストの変換**



- **フィルタリング条件**

条件	内容
HTTPステータス	200のみ
コンテンツ種別	htmlのみ
テキスト長	２０文字未満はスキップ

コンポーネント詳細: Converterの出力形式

SB Intuitions

- JSONLスキーマ

```
{
  "text": "抽出されたページ本文テキスト...",
  "url": "https://example.com/article/123",
  "download_date": "2026-02-15T10:30:00",
  "language": "ja",
  "domain": "example.com"
}
```

- 1GBチャンキング
  - ドキュメントは1GB区切りでS3に書き出す
  - 後続の学習パイプラインで行単位のストリーミング処理が可能

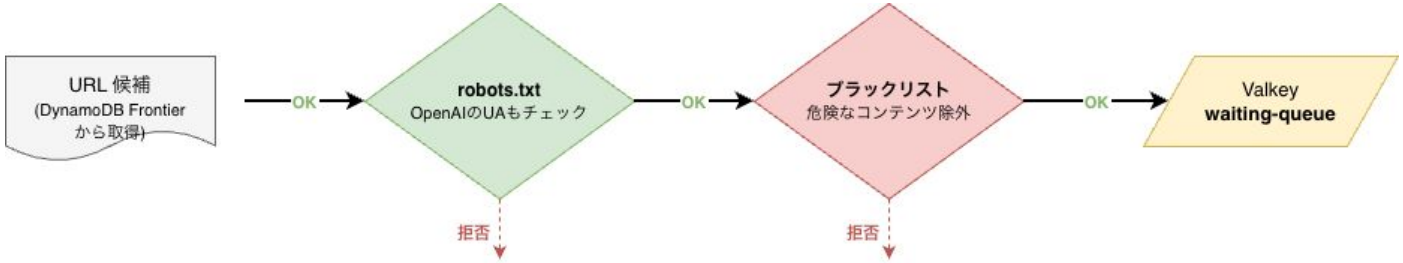
コンポーネント詳細: LinkExtractor

SB Intuitions

- クロール対象URLの自律的な拡大
  - クロール結果のHTMLからURLを抽出することで新たなURLを発見する
  - 定期バッチで実行しクロール日を元に前回からの新たなHTMLをS3から取得して処理を実行

コンポーネント詳細: Injector

- DynamoDBからURLを取得し、フィルタリングしてキューに投入



施策	内容
robots.txtの遵守	自社のUAだけでなく、OpenAIのクローラーのUAもチェック
ブラックリスト	学習利用が問題になりそうなドメインをあらかじめ除去

シャーディング戦略

SB Intuitions

- これまでの構成はそれぞれのWorkerが単体であれば機能するが、Slurmの性能を活かすために並列化しようとするとDomainのレートリミットなどが制御できなくなり破綻する
  → シャーディングにより破綻なく独立したスケーリングを実現する

1. Write Shard (DynamoDB 書き込み分散)
   a. PK = "{reverse_host}#{MD5(host) % 10}"
2. GSI Shard (DynamoDB 読み取り分散)
   a. 100 シャードを各ワーカーが分担してクエリ
3. Queue Shard (Valkey キュー分散)
   a. 同一 FQDN を必ず同じValkey キューに流れるように → レートリミット実現

はまったポイント

SB Intuitions

- **Shard計算ロジックの不整合**
  - InjectorとRecoderでWrite Shardの計算ロジックがずれていた
  - Shard使ってフラグ更新クエリも行うので別のシャードとして更新してしまいInjectorが同じURLを再取得し続ける
- **解決方法**
  - 全データのシャードを再計算
  - テーブルを再構築
  - シャードロジックを修正
- **教訓**
  - エラーにならず処理が進むため発見が遅れた
  - テーブル全て作り直しとなったのでコストもそれなりに発生

パフォーマンス測定

SB Intuitions

- スループット

ステージ	スループット	備考
Fetcher(HTML取得)	69 件/秒（約25万件/時）	S3アップロード含む
Recoder(状態記録)	103〜129 件/秒（約37〜46万件/時）	
Injector(URL注入)	約188万件/回	バッチ実行、所要約 30分

- 分析
  - デイリーの理論値は約600万件程度になるため当初目標だったデイリー1000万件にはまだ届かない
  - DBとの通信を仲介するプロキシの負荷がかなり高いことを確認しており、ここがボトルネックになりつつある

今後の展望

SB Intuitions

- **コーパス品質向上**
  - CommonCrawlのURLを元にした未取得のコンテンツの効率的な取得
  - 取得したページのカテゴリなどの**メタ情報**の拡充
  - 同一のページでも取得日付が古いものの**再取得**
- **スケーラビリティ**
  - まだ当初目標としていたクロール速度(デイリー1000万ページ)に到達できていないので速度の向上にも取り組む
  - 採用するインフラプラットフォームの見直し

まとめ

SB Intuitions

- **内製クローラーにより独自のWebコーパスを構築**
  対象ページを自分たちで制御可能なので、より多くの有用なテキストを収集できることが期待できる
- **SlurmとAWSを組み合わせて安定したクロール基盤を構築**
  Valkey Streams + DynamoDB + 3 層シャーディングで各ボトルネックを個別に解消
- **クローリングを行う際に求められるポライトネスに配慮した収集**
  robots.txt 遵守・ドメインレートリミット・多層フィルタリングを設計に組み込む

直感を、知性へ


SB Intuitions