Asakusaソースコードリーディング #5 - YAESS & WindGate

2011/11/30

Suguru ARAKAWA

対象ソースコード

- https://github.com/asakusafw/asakusafw/tree/SCR-05
  - git://github.com/asakusafw/asakusafw.git
  - Tag: SCR-05
  - 2011/11/30時点の開発ブランチ最新

お題

- リリース紹介
  - リリース済み: 0.2.2, 0.2.3
  - 現在: 0.2.4
- コンポーネント紹介
  - YAESS
    - ポータブルなジョブ実行ツール
  - WindGate
    - ポータブルなデータ転送ツール

リリース紹介

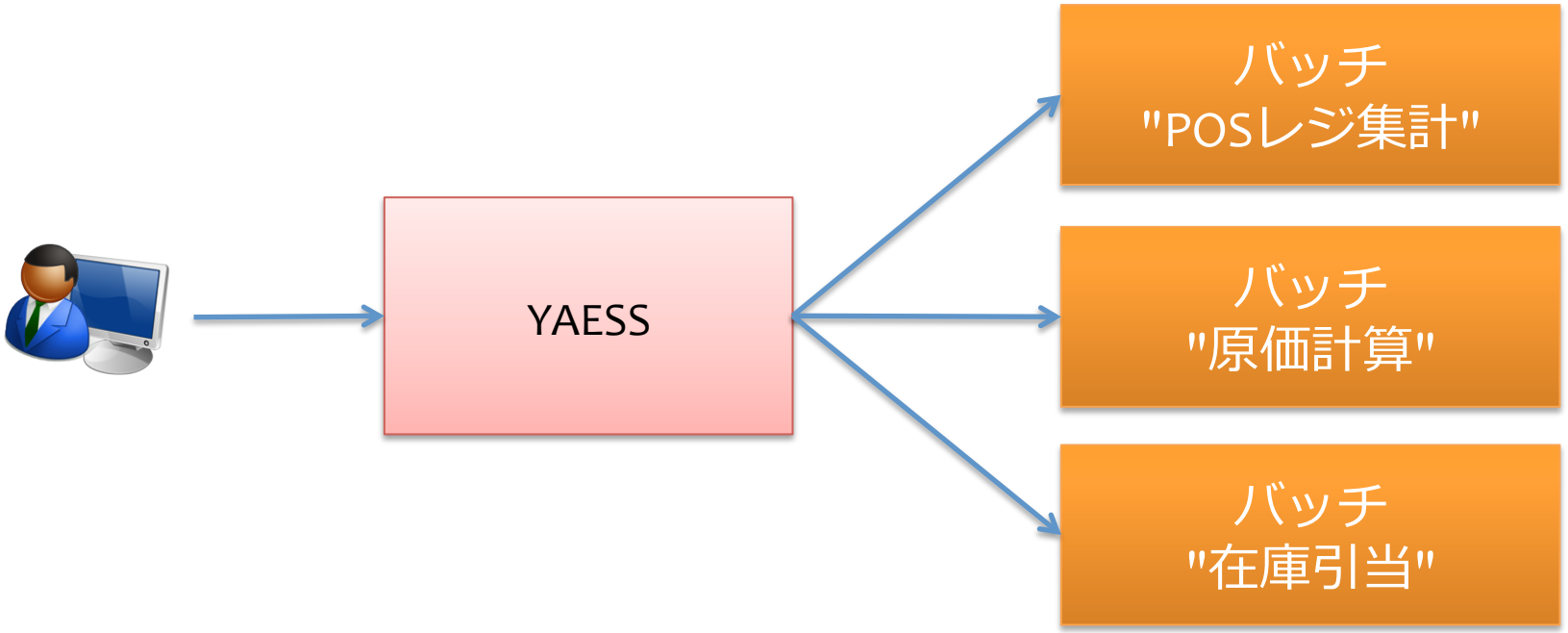
- 0.2.2 – 2011/09/29
  - WindGate
- 0.2.3 – 2011/11/16
  - YAESS
  - Cache for ThunderGate
- 0.2.4 – working
  - WindGate CSV
  - Documentations
  - ?

ポータブルなジョブ実行ツール

YAESS

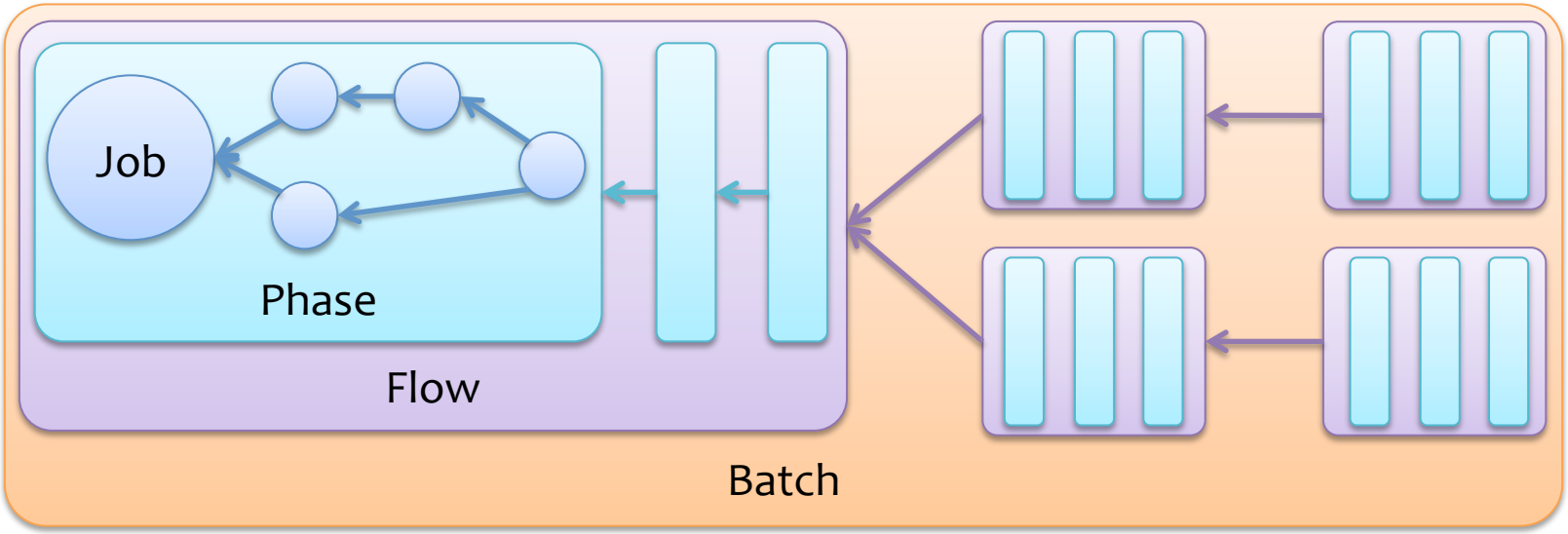
YAESS

- Asakusaのバッチを実行するコマンドライン ツール
  - バッチの内部構造を気にせず統一的に起動
  - 環境構成の差異をYAESSの設定で吸収



バッチの構造

- フロー
  - トランザクション処理単位
- フェーズ
  - インポート、エクスポートなどの処理単位
- ジョブ
  - 個々のHadoopジョブなど

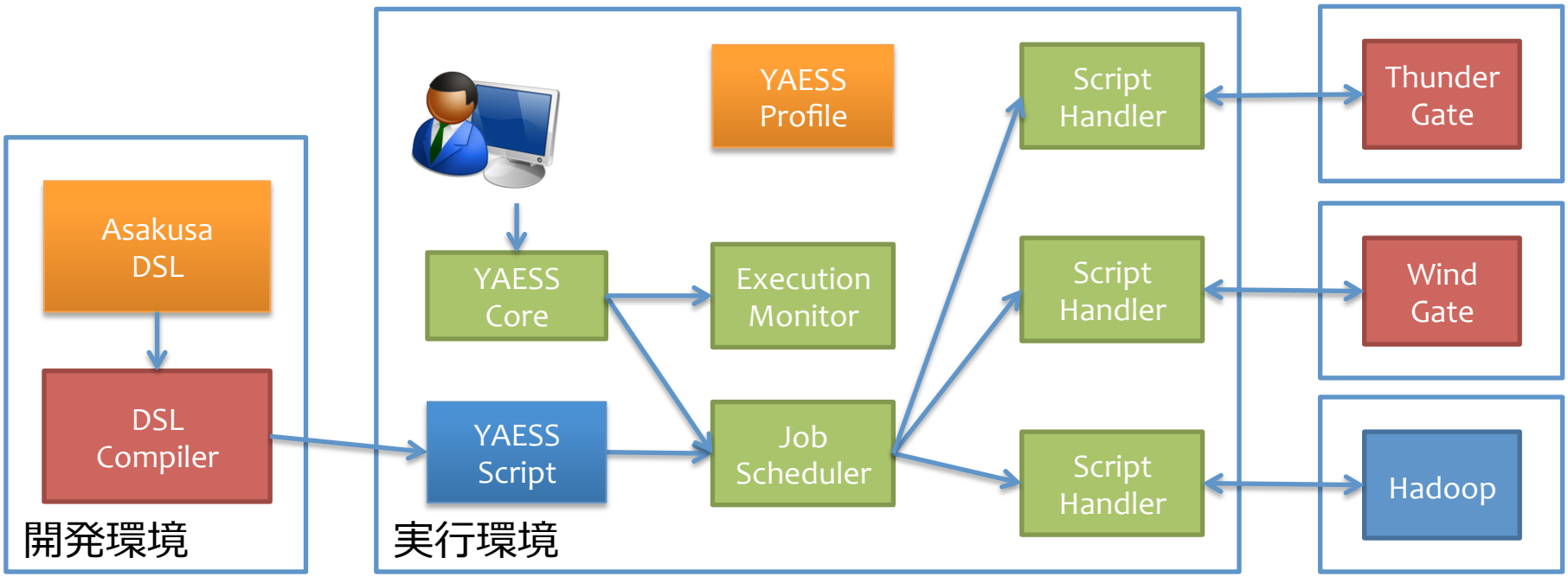


YAESSのコマンド群

- yaess-batch
  - バッチ全体を起動
- yaess-phase
  - 個々のフェーズを起動
- yaess-explain
  - バッチの構造をダンプ

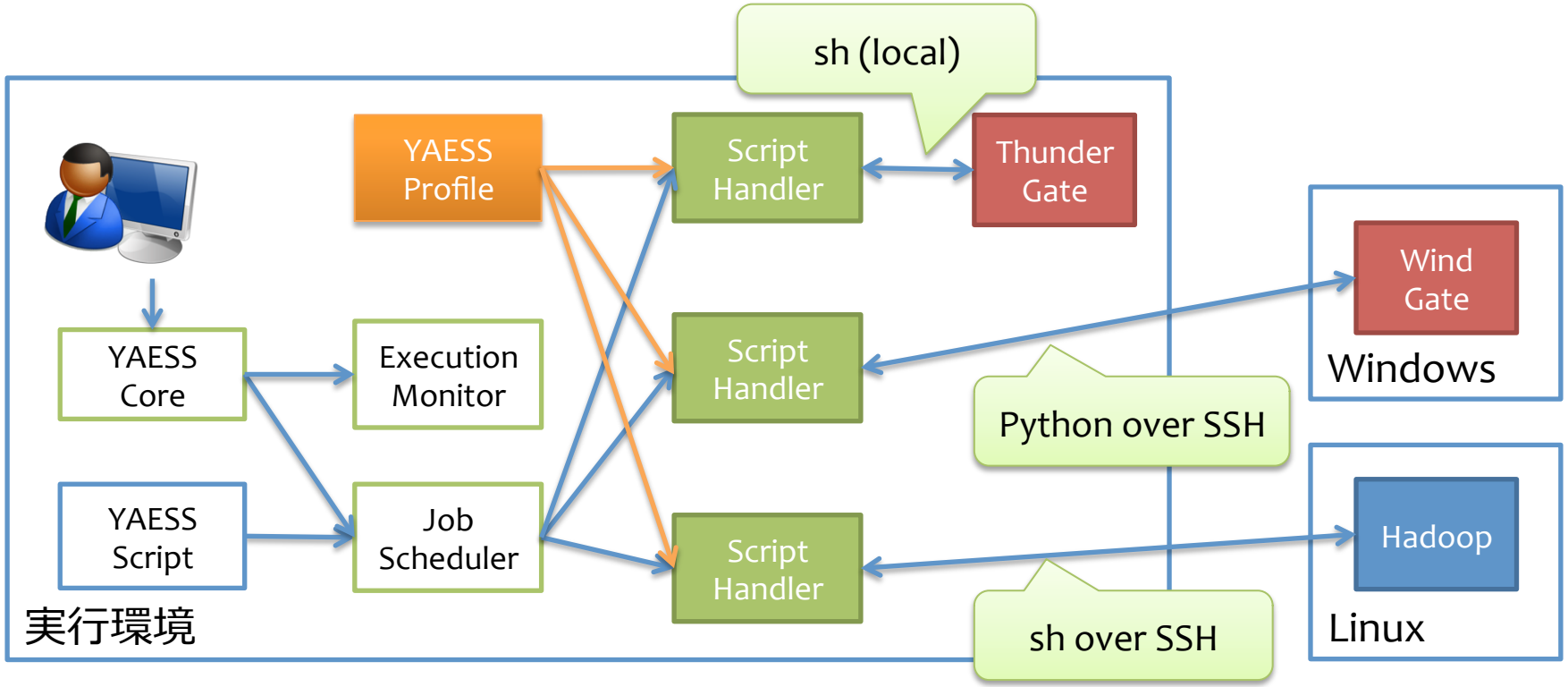
YAESSのアーキテクチャ

- プラガブルな構成
  - Script Handler, Job Scheduler, Execution Monitorなど
  - それぞれのコンポーネントはプロファイルで設定



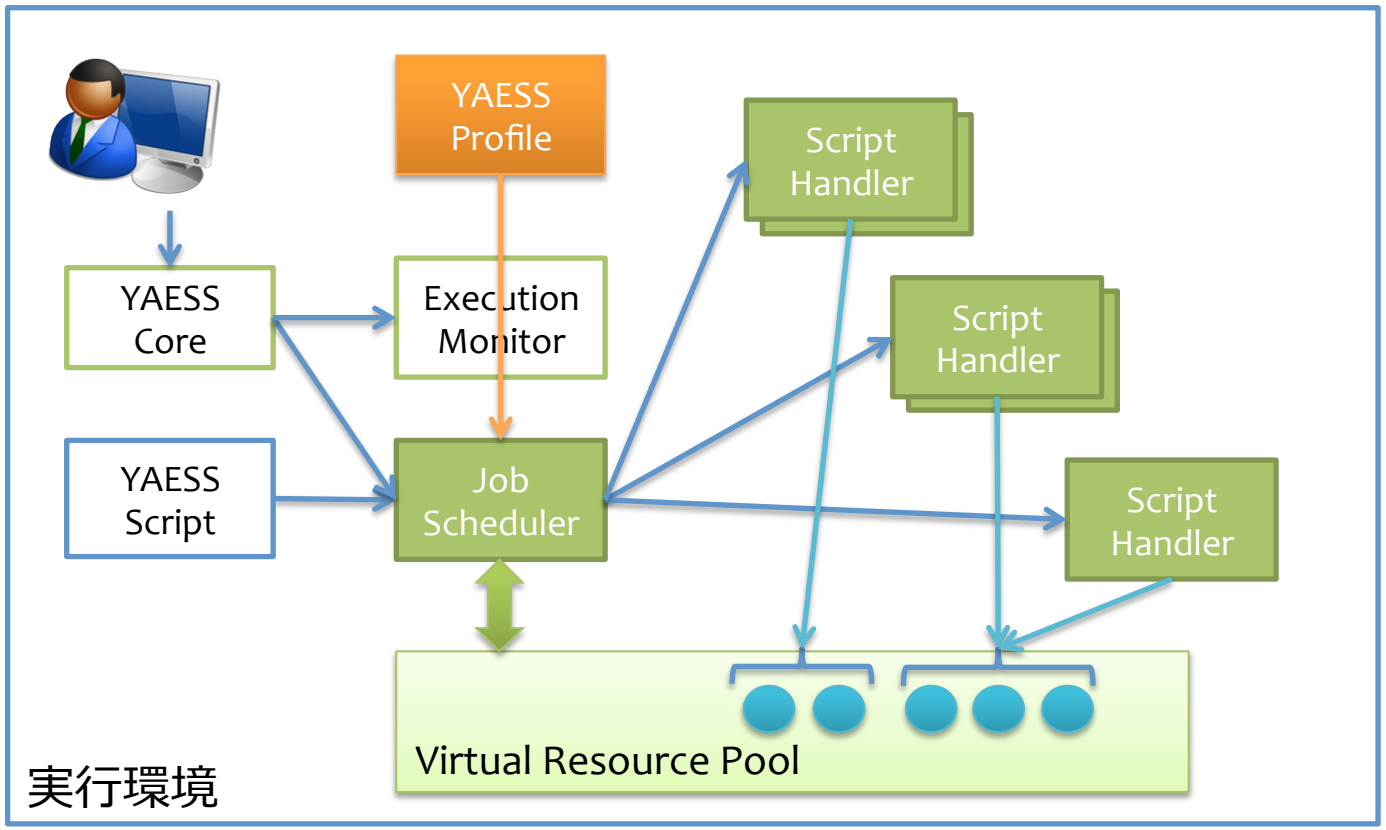
Script Handler

- 実際にコマンドを発行する箇所をフック
  - DSLでの設定に応じて適切なハンドラに振り分け
  - SSHを経由したり、Pythonなどを経由したり



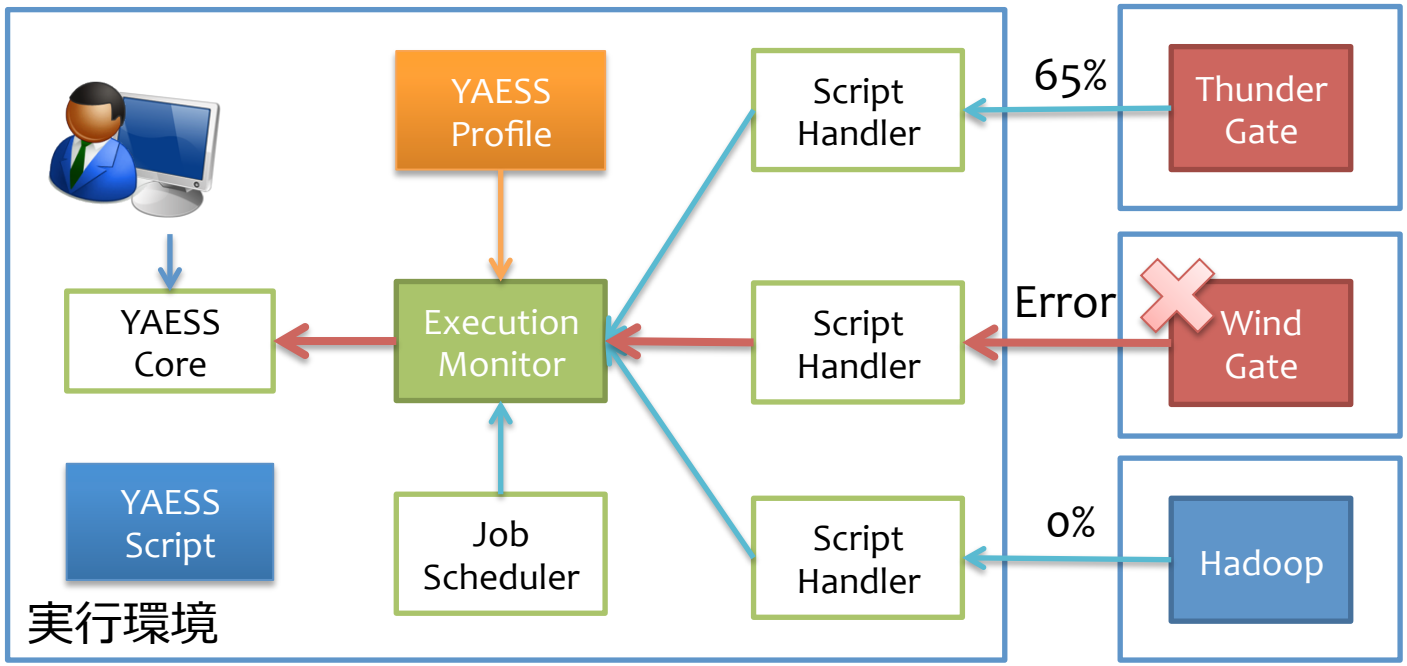
Job Scheduler

- リソースの利用状況に応じて多重度を調整する
  - 標準では大したことをやっていない
  - プラグインなので差し替え可能



Execution Monitor

- 進捗状況を受け取るフック
  - 進捗表示やキャンセル処理などのため
  - プラグインなので差し替え可能



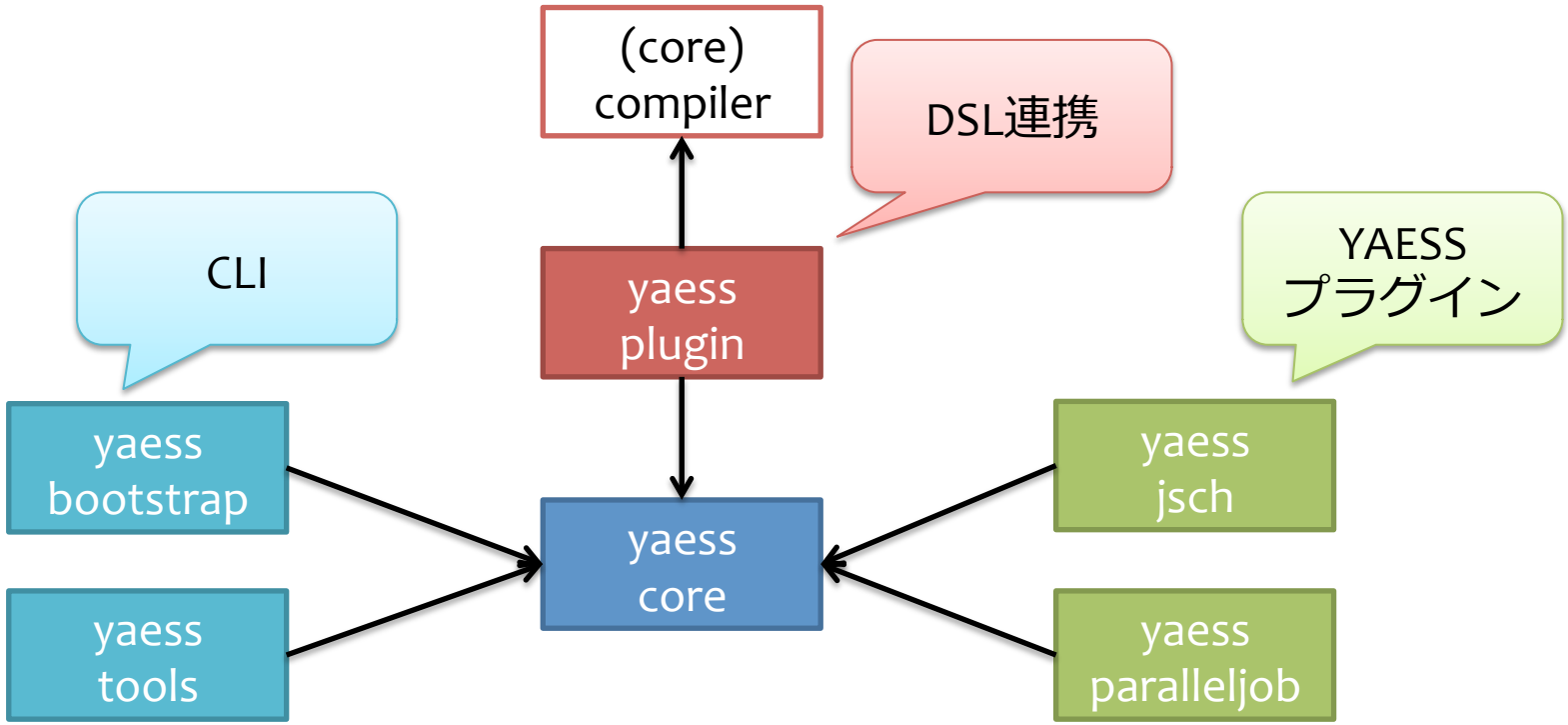
YAESSの立ち位置

- ポータブルなバッチ実行「インターフェース」
  - Asakusaの実行計画詳細を隠蔽
    - 動的実行計画に切り替えても同じ使い方
  - バッチ実行の環境構成を隠蔽
    - Hadoopのクラスタを切り替えても同じ使い方
  - 個々のジョブの起動方法を隠蔽
    - Hadoopを使わなくなっても同じ使い方
- 外部ジョブ運用ツールとの連携も想定
  - yaess-explain + yaess-phase
  - インターフェースとしての利点を得られる

YAESSのモジュール構成

- asakusa-yaess-core
  - コアライブラリ群
- asakusa-yaess-bootstrap
  - コマンドラインインターフェース
- asakusa-yaess-tools
  - 周辺ツール集
- asakusa-yaess-plugin
  - コンパイラプラグイン
- asakusa-yaess-jsch
  - SSHを利用したスクリプトハンドラの実装 (プラグイン)
- asakusa-yaess-paralleljob
  - ジョブの並列実行用スケジューラ (プラグイン)

YAESSのモジュール構成 (俯瞰)

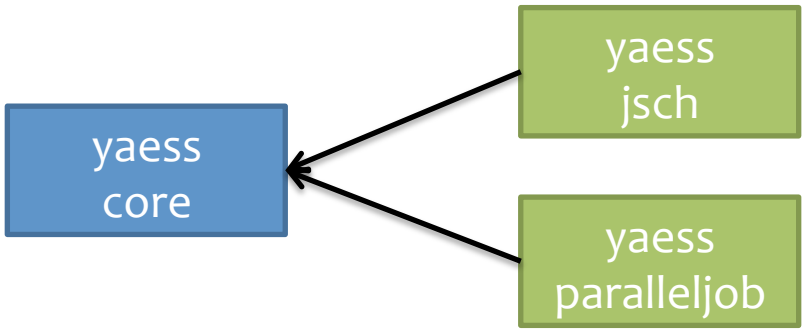


YAESSのプラグイン機構

- Service Provider Interface (SPI) を提供
  - 必要なインターフェースを実装すればプラグインに
- SPIの一覧 (in "asakusa-yaess-core")
  - com.asakusafw.yaess.core.ExecutionScriptHandler
    - ジョブ起動リクエストを処理するハンドラ
  - com.asakusafw.yaess.core.JobScheduler
    - フェーズ内のジョブ実行をスケジュールする機構
  - com.asakusafw.yaess.core.ExecutionMonitorProvider
    - フェーズの実行状況を通知するモニタ機構
  - com.asakusafw.yaess.core.ExecutionLockProvider
    - 多重起動を抑制するロック機構

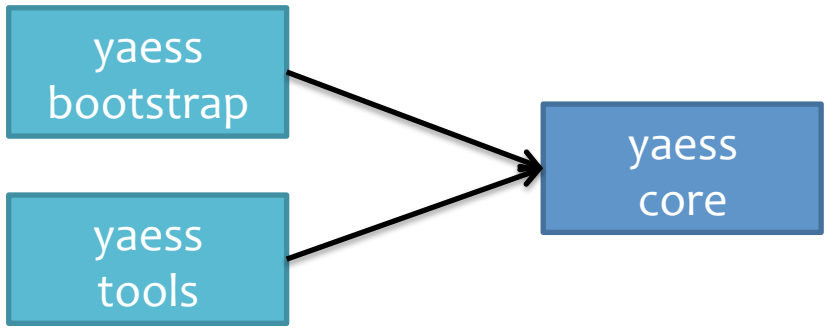
プラグインの例 – ExecutionScriptHandler

- ...yaess.basic.BasicHadoopScriptHandler
  - OSのコマンド実行でHadoopジョブを起動する
  - ほとんどのコードは ProcessHadoopScriptHandler 内
- ...yaess.jsch.SshHadoopScriptHandler
  - SSH経由でHadoopジョブを実行する
  - ほとんどのコードは ProcessHadoopScriptHandler 内



YAESSのモジュール構成 – CLI

- com.asakusafw.yaess.bootstrap.Yaess
  - in "asakusa-yaess-bootstrap"
  - L119: Yaess.parseConfiguration()
    - コマンドライン引数の解析
  - L141: ExecutionTask.load()
    - 構成のロード
  - L152: ExecutionTask.executeBatch()
    - バッチの実行



ExecutionTask

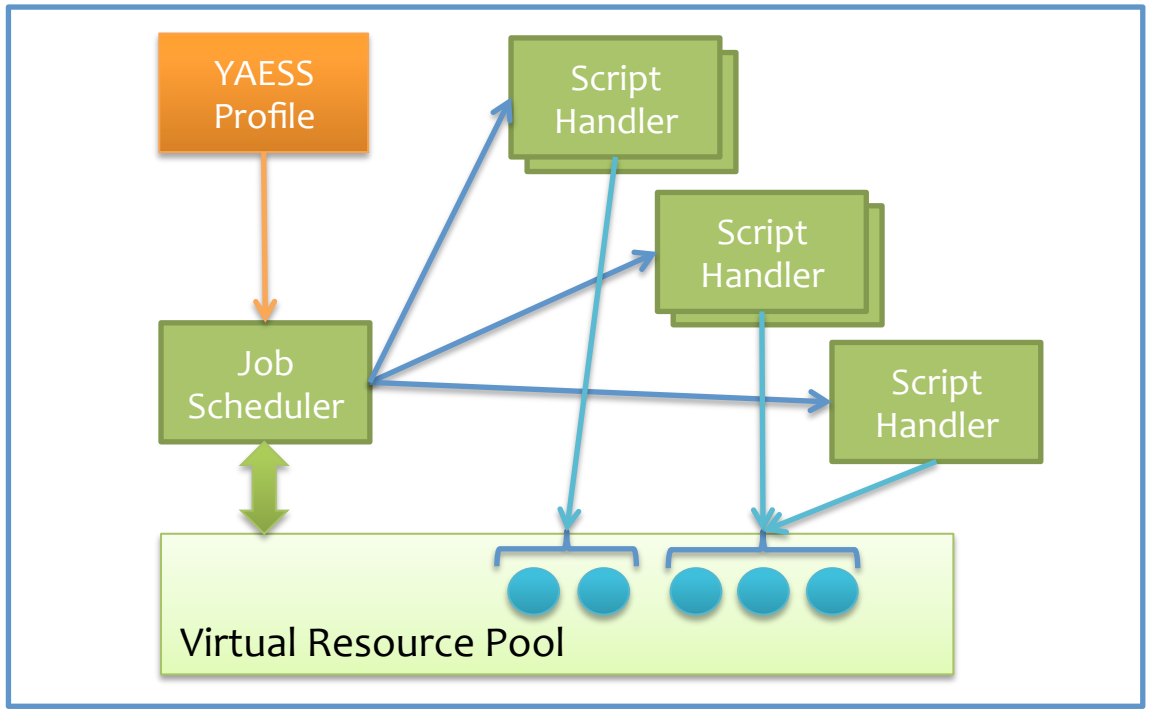
- com.asakusafw.yaess.core.task.ExecutionTask
  - in "asakusa-yaess-core"
  - YAESS全体の実行シーケンス
  - 最小単位は L259: "executePhase()"
- 主な流れ
  - 多重起動を抑制するロックを取得
  - ジョブネット構造を取得
  - エラーハンドラを選択
  - 通知用のモニタを作成
  - スケジューラにジョブネットを投入
  - モニタを完了
  - ロックを開放

AbstractJobScheduler

- com.asakusafw.yaess.basic.AbstractJobScheduler
  - スケジューラの骨格実装
  - L129: "Engine.run()" が本体
- 主な流れ
  - L162: "Engine.submitAllWaiting()"
    - WAITINGプール内の発行可能なジョブをすべて発行
      - 実際の発行方法はサブクラスで決める (プラグイン)
      - ジョブはEXECUTINGプールに
      - 完了したらDONEキューに行く
  - L179: "Engine.waitForDone()"
    - DONEキューから一つ以上取り出す
  - L130: WAITプールが空になるまで繰り返す
  - L149: EXECUTINGプールが空になるまで待つ

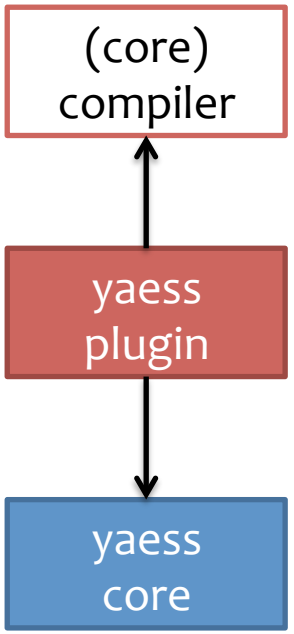
ParallelJobExecutor

- ...yaess.paralleljob.ParallelJobExecutor
  - in "asakusa-yaess-paralleljob"
  - ジョブを実際に発行する部分の実装 (L170: "submit()")
  - ここからExecutionScriptHandlerをさらに呼び出す



YAESSのモジュール構成 – DSL連携

- ...compiler.yaess.YaessWorkflowProcessor
  - in "asakusa-yaess-plugin"
  - DSLからYAESSのワークフロー情報を生成
- コンパイル時に自動的に呼ばれる
  - DSLコンパイラ本体のプラグイン機構を利用
  - 本体には手を加えていない



YAESSまとめ

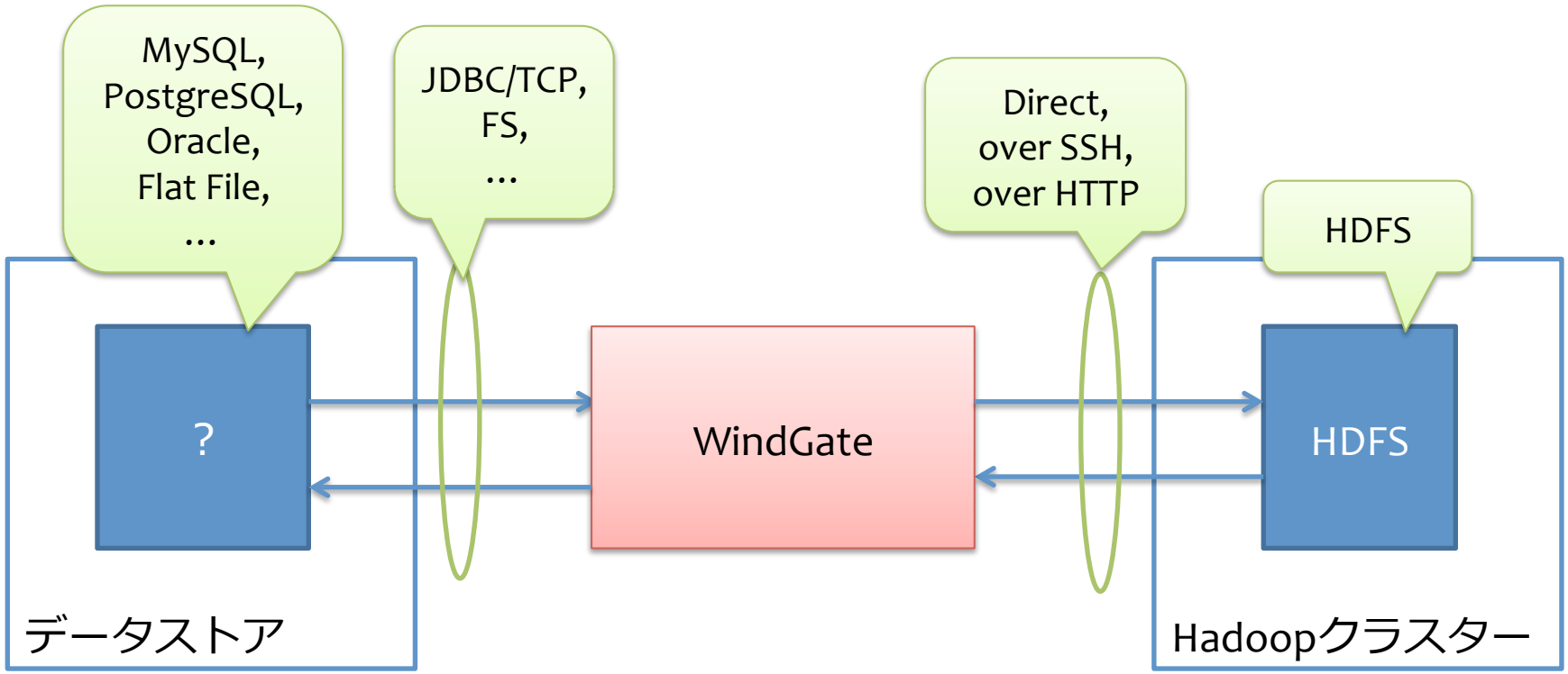
- ポータブルなバッチ実行ツール
  - 様々なものをプラグインとして外部化
    - ジョブのスケジューリングをする部分
    - ジョブを実際に実行する部分
    - など
  - MinGWとの連携もできたらしい
    - 起動時に "bash.exe" を経由してシェルスクリプトを叩いてた
- DSL連携も実施
  - DSLで記述したバッチからYAESSのワークフロー情報を自動的に出力
  - コンパイラプラグインを利用

ポータブルなデータ転送ツール

WindGate

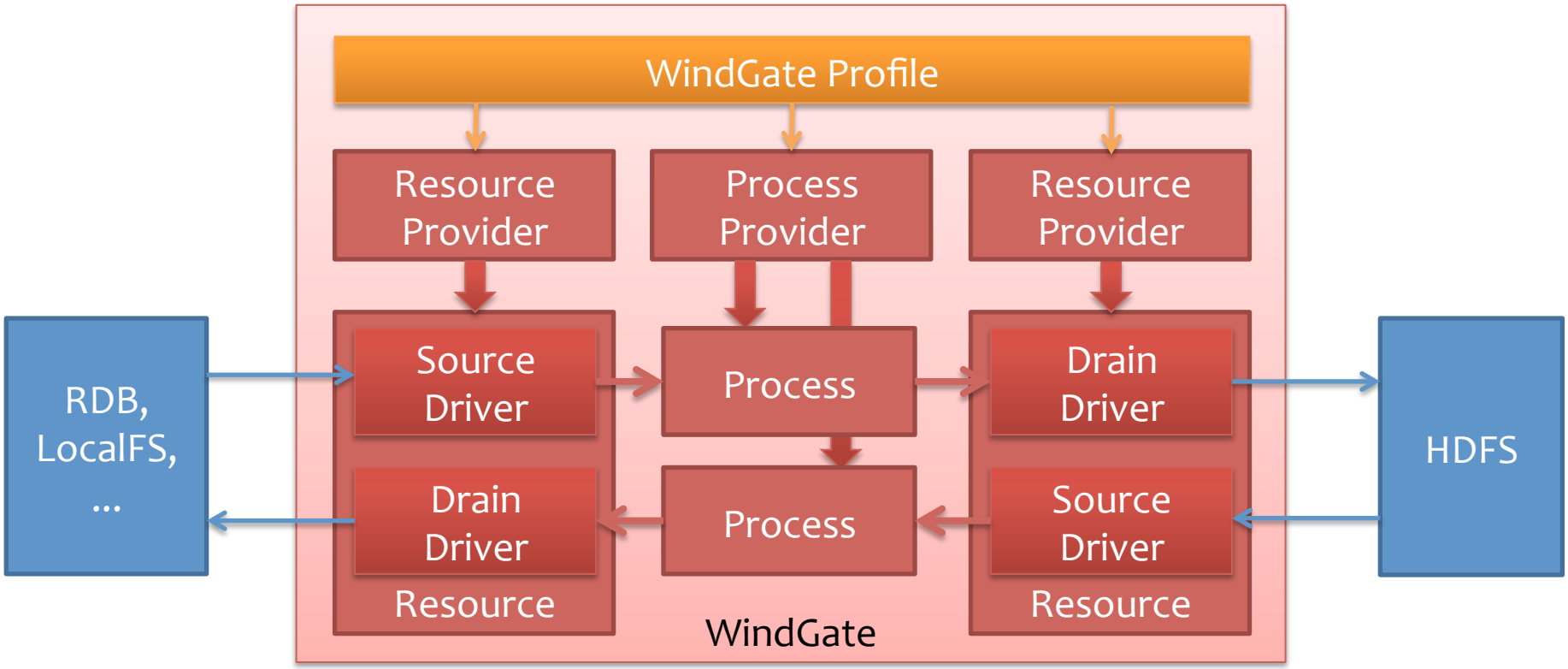
WindGate

- ポータブルなデータ転送ツール
  - JDBC経由の転送(0.2.2)やファイルの転送(0.2.4?)



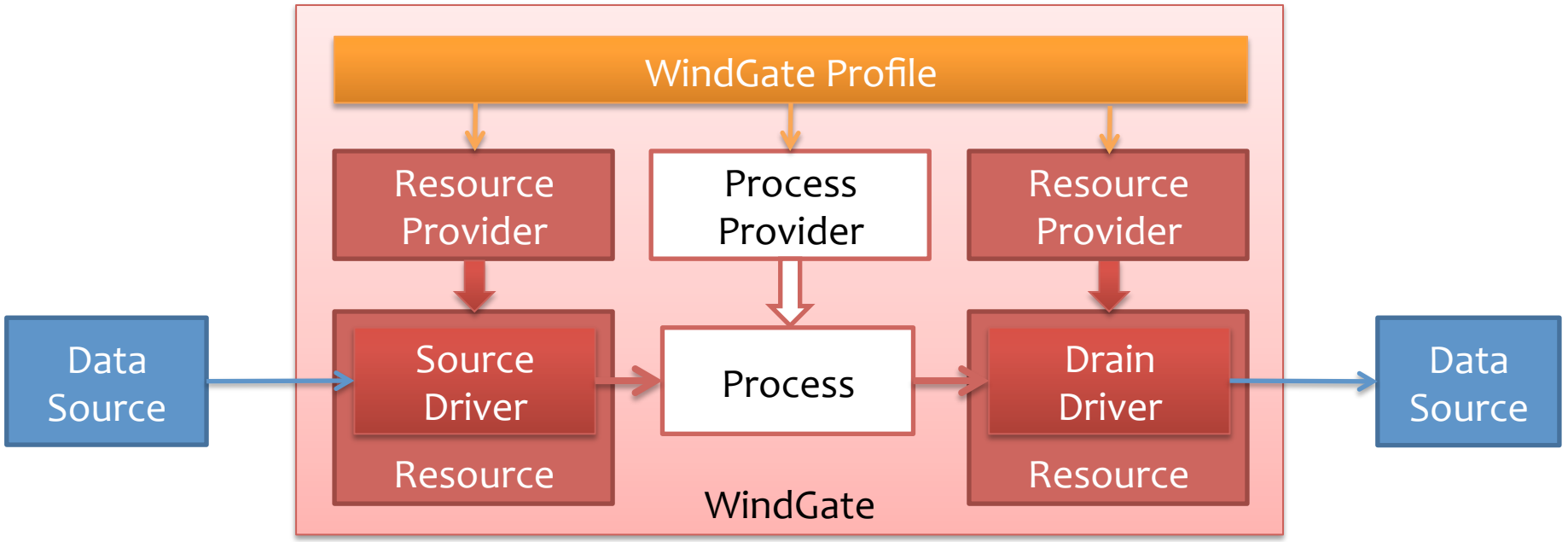
WindGateのアーキテクチャ

- データソースを「リソース」としてプラグイン化
  - 対象データソースをプラグインで自由に選択



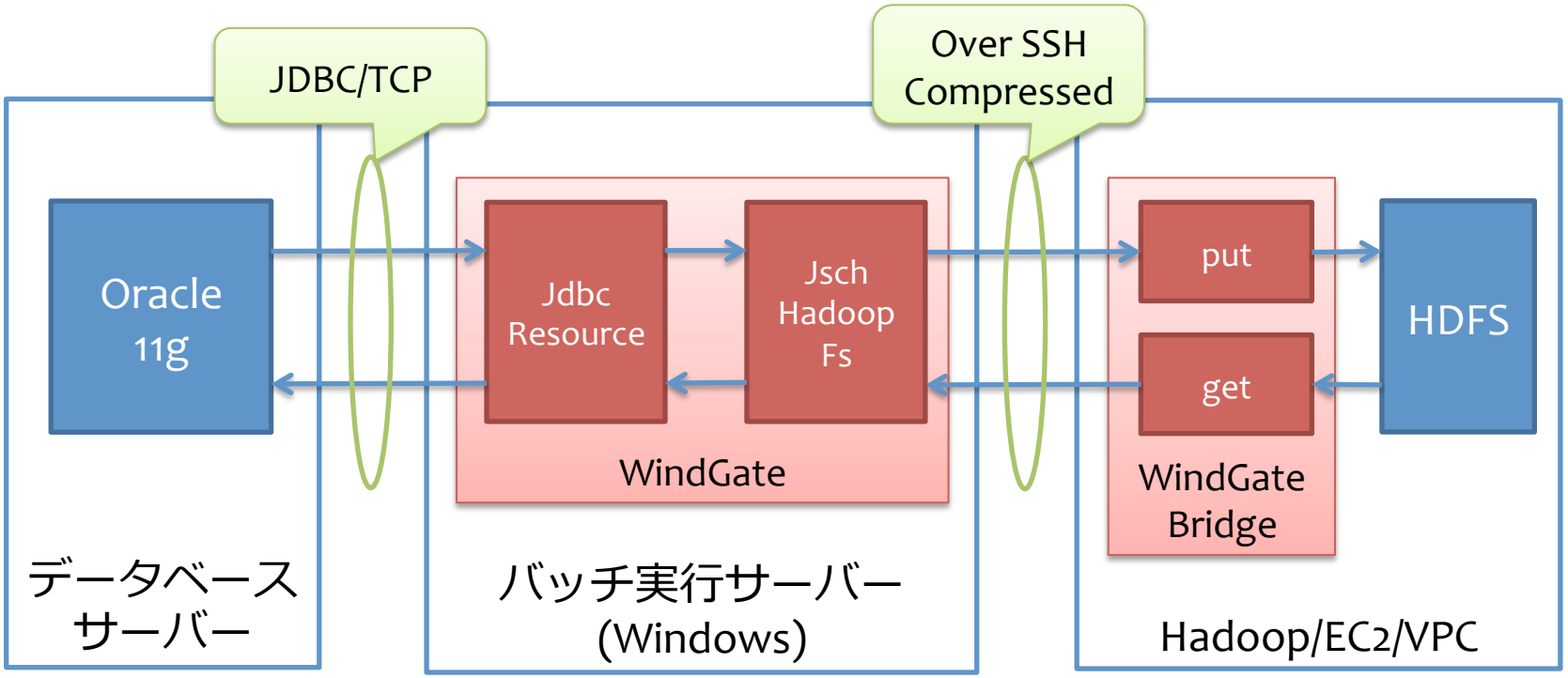
WindGateの「リソース」

- リソースのデータ形式や転送手段を吸収
  - 内部ではデータモデルオブジェクトでやり取り
  - プラガブルな作りで自由に差し替えられる
    - 相手がHadoopである必要すらない



WindGateの構成例

- 実際に利用している構成



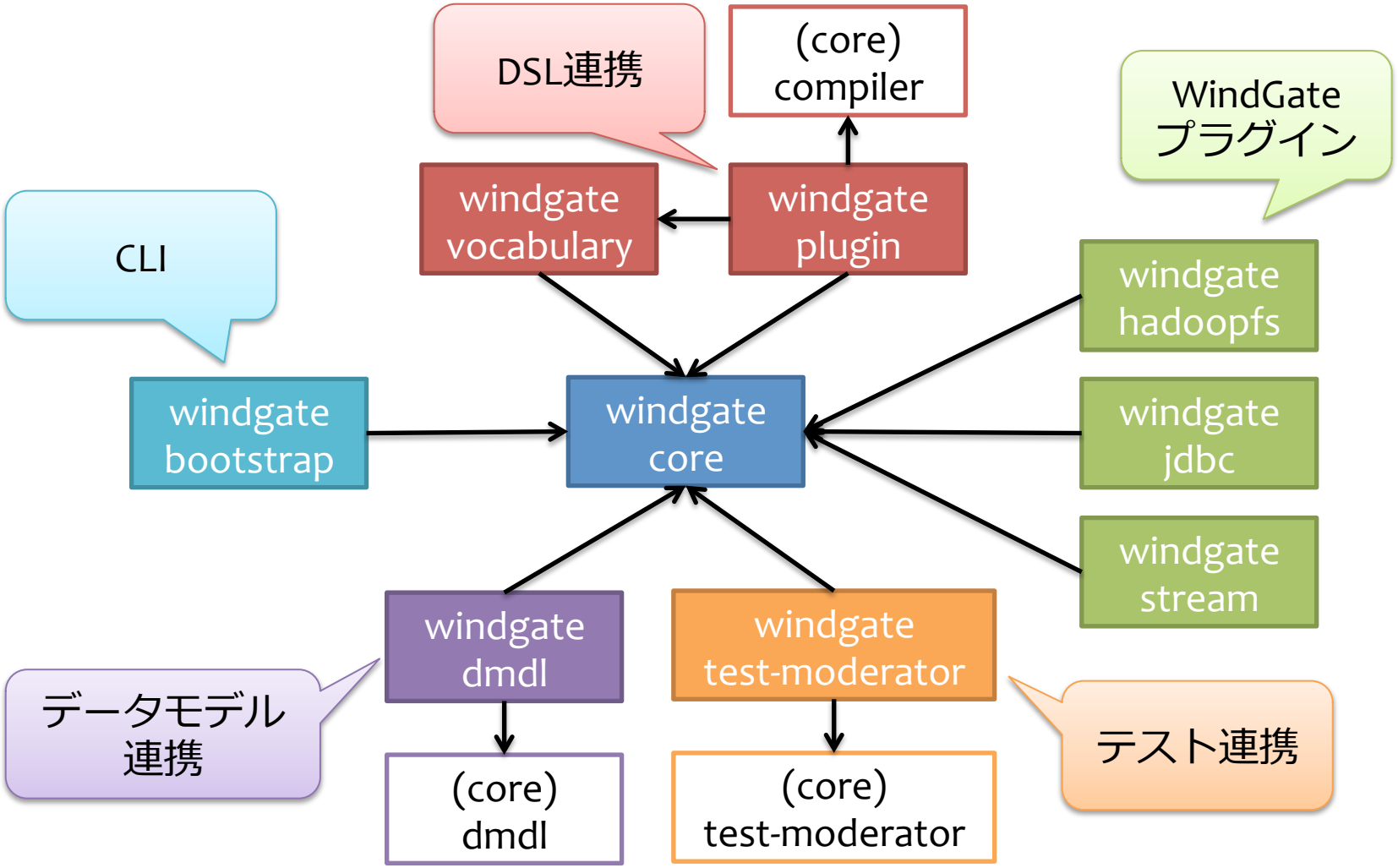
WindGateの立ち位置

- 「プッシュ型」のデータ転送ツール
  - 外部からHDFSへデータを投入する
  - 外部からHDFSのデータを取得する
  - パフォーマンス優位性は低い
    - 外部とHadoop間のネットワーク通信が発生する
    - データ転送に関してHadoopクラスタの性能を生かせない
- 「システム」として考えたときに難しい箇所
  - データの投入をどう考えるか
  - データの取得をどう考えるか
  - データの保全をどう考えるか

WindGateのモジュール構成

- asakusa-windgate-core
  - コアライブラリ群
- asakusa-windgate-bootstrap
  - コマンドラインインターフェース
- asakusa-windgate-plugin
  - DSL連携
- asakusa-windgate-test-moderator
  - テスト連携
- asauksa-windgate-dmdl
  - データモデル連携
- asakusa-windgate-(hadoopfs|jdbc|stream)
  - 各種リソースのプラグイン

WindGateのモジュール構成 (俯瞰)



おわび

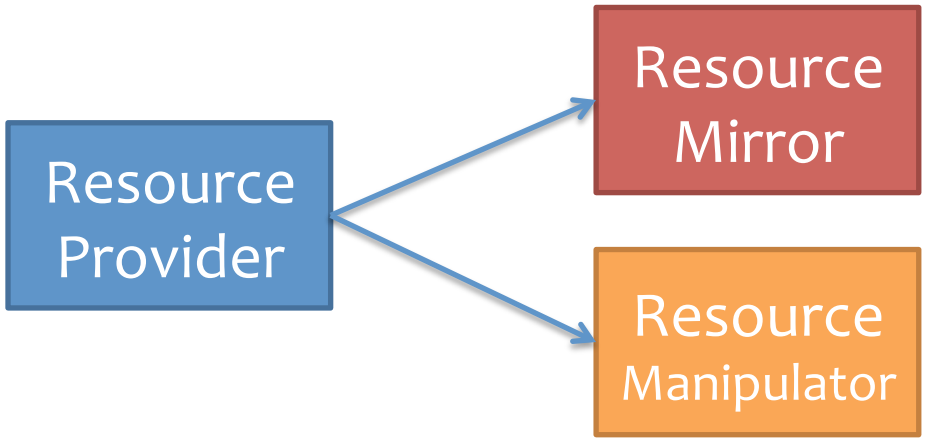
- あんまりソースコード読みません

WindGateのプラグイン機構

- Service Provider Interface (SPI) を提供
  - 必要なインターフェースを実装すればプラグインに
- SPIの一覧 (in "asakusa-windgate-core")
  - ...windgate.core.resource.ResourceProvider
    - データソースを抽象化するリソースを提供する機構
  - ...windgate.core.process.ProcessProvider
    - リソース間のデータ転送を行うプロセスを提供する機構
  - ...windgate.core.session.SessionProvider
    - 一連の処理をまとめるセッションの管理機構

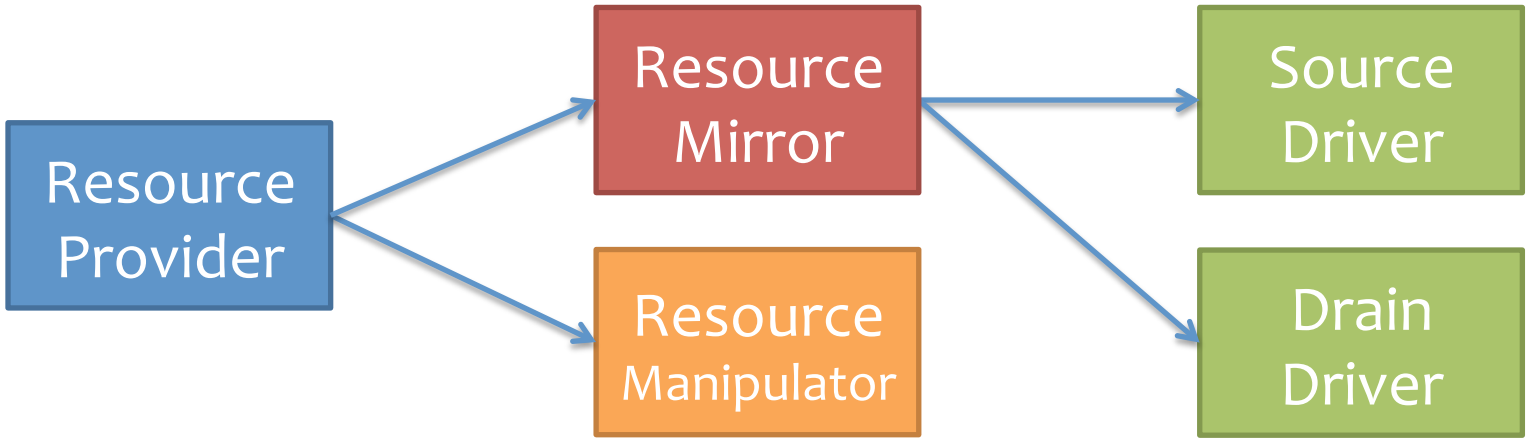
ResourceProvider

- ...windgate.resource.ResourceProvider
  - in "asakusa-windgate-core"
  - L39: create() : ResourceMirror
    - データ転送用のリソースを作成
  - L67: createManipulator() : ResourceManipulator
    - テスト用のリソースを作成



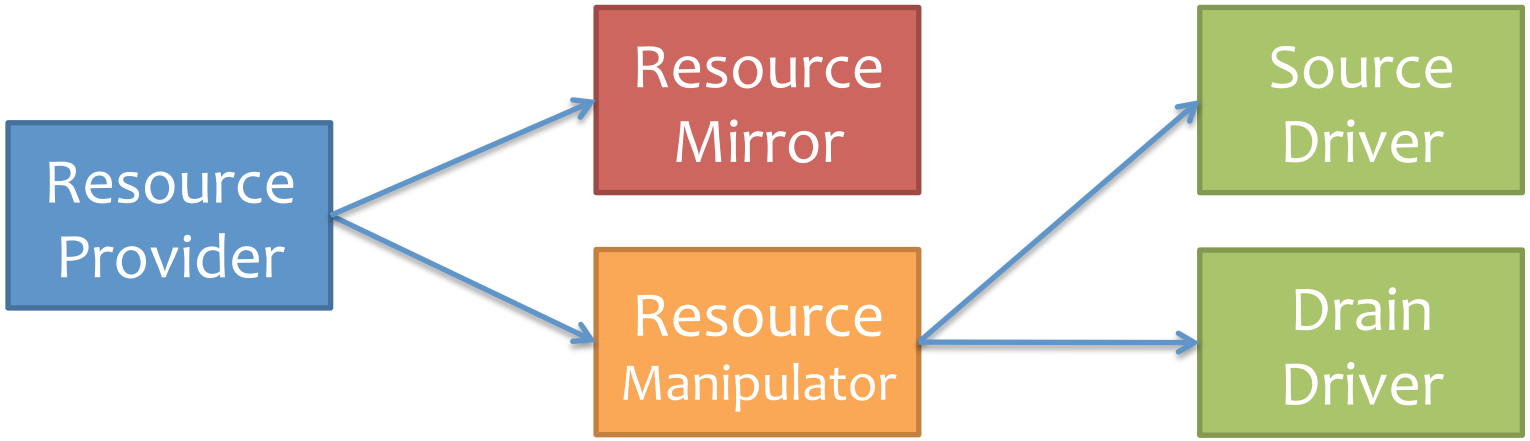
ResourceMirror – リソース

- ...windgate.resource.ResourceMirror
  - in "asakusa-windgate-core"
  - L87: createSource() : SourceDriver<T>
    - データ入力用のオブジェクトを生成
  - L98: createDrain(): DrainDriver<T>
    - データ出力用のオブジェクトを生成



ResourceManipulator – テスト用リソース

- ...windgate.resource.ResourceManipulator
  - in "asakusa-windgate-core"
  - L68: createDrainForSource(): DrainDriver<T>
    - 入力データを作成する
  - L78: createSourceForDrain() : SourceDriver<T>
    - 結果を取得する



Source/DrainDriver

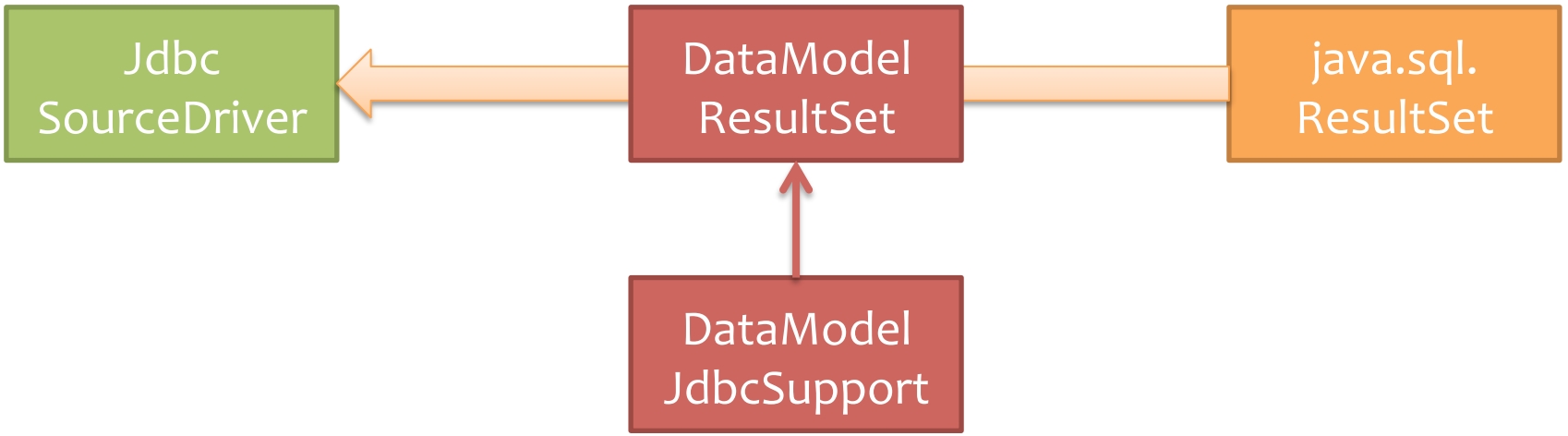
- ...windgate.resource.SourceDriver<T>
  - in "asakusa-windgate-core"
  - L56: get() : T
  - リソースからデータを読み込む
- ...windgate.resource.DrainDriver<T>
  - in "asakusa-windgate-core"
  - L46: put(T)
  - リソースにデータを書き出す

JdbcSourceDriver

- ...windgate.jdbc.JdbcSourceDriver
  - in "asakusa-windgate-jdbc"
  - JDBCを利用したSourceDriver
    - ...JdbcResourceProviderから生成される
- L110: "support = ..." in prepare()
  - supportはDataModelResultSet型
  - ResultSetの内容をオブジェクトにマッピングする
- L168: "support.next()"
  - ResultSetから一行読んでオブジェクトに詰めている

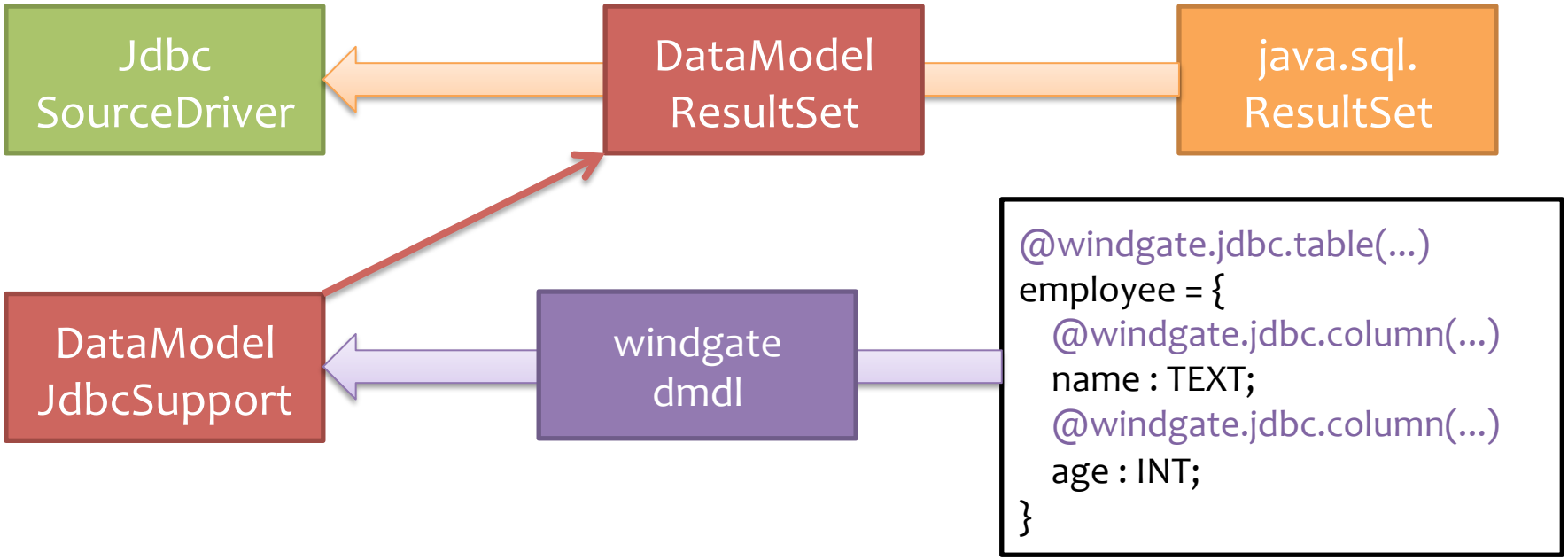
DataModelJdbcSupport

- ...windgate.vocabulary.DataModelJdbcSupport
  - DataModelResultSet
    - ResultSetの内容をオブジェクトにマッピング
  - DataModelPreparedStatement
    - オブジェクトの内容をPreparedStatementのパラメータに指定



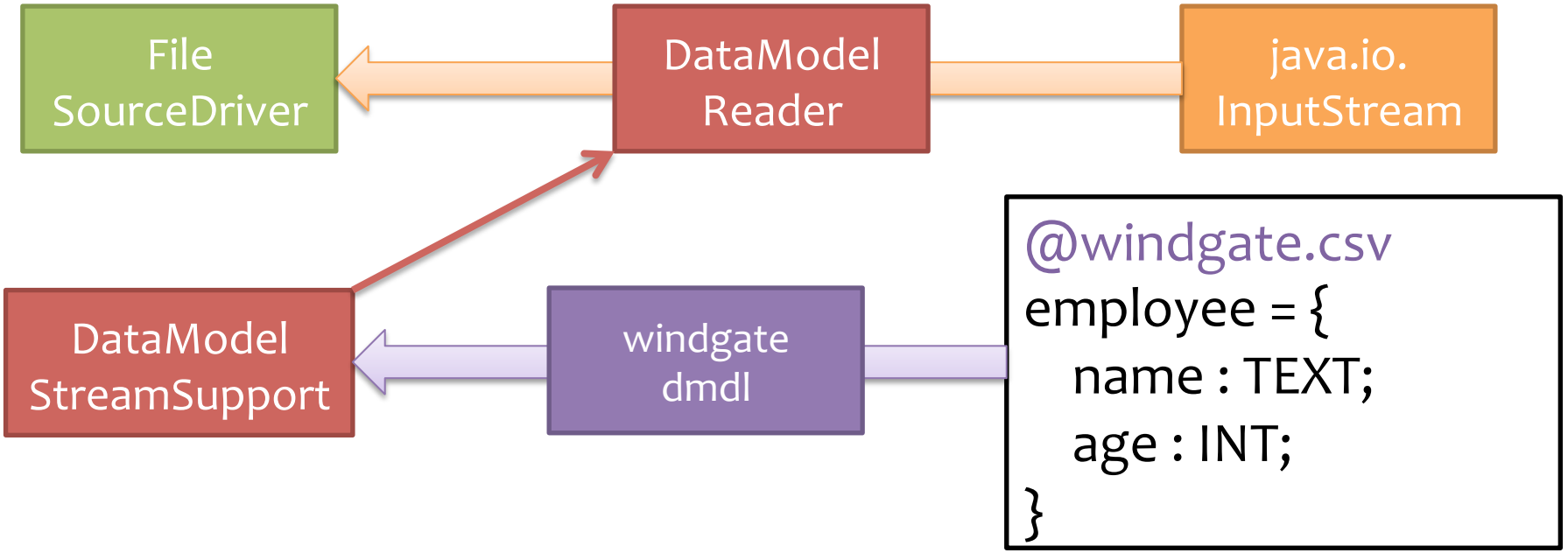
DataModelJdbcSupportの生成

- データモデルの記述から自動生成
  - ...windgate.jdbc.driver.JdbcSupportEmitter
    - in "asakusa-windgate-dmdl"
    - DMDLのコンパイラプラグインを利用



CSVファイルでの構成

- CSVファイルの利用もJDBCと似たような構成
  - ただしInputStream/OutputStreamを対象に
  - DataModelStreamSupportが仲介



ここまでのまとめ

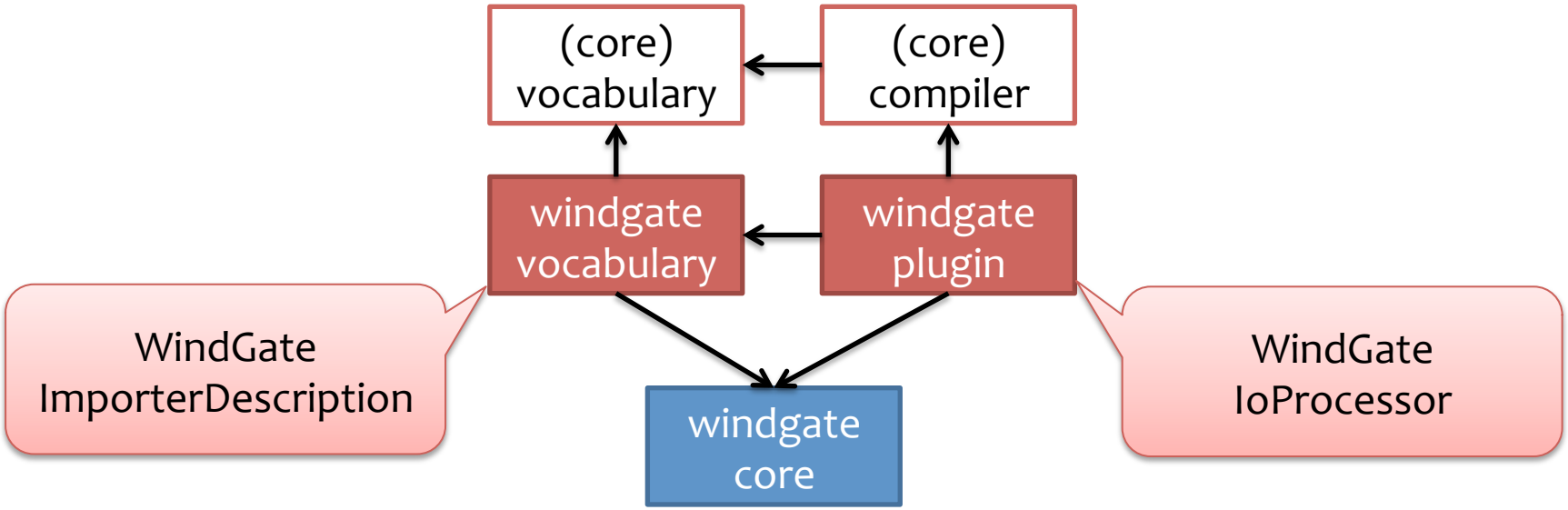
- リソースがプラガブル
  - ソースとドレインでデータの読み書き
  - データモデル記述のコンパイラと組み合わせて動く
- WindGate本体部分の仕組みはこれくらいに
  - ...windgate.bootstrap.WindGate がプログラムエントリ
  - 時間の関係で省略

JdbcImporterDescription

- ...vocabulary.JdbcImporterDescription
  - in "asakusa-windgate-vocabulary"
  - データベースからインポートするためのDSL語彙
  - L45: getJdbcSupport() : DataModelJdbcSupport
    - 先ほどのResultSetを変換するやつ
  - L51: getTableName() : String
    - インポート対象のテーブル名
- WindGateImporterDescriptionを継承している
  - L69: getDriverScript() : DriverScript
    - Asakusa DSLの語彙からWindGateの処理内容を生成
      - getJdbcSupport(), getTableName(), ... の内容を元に構築
    - ここに「SourceDriver」がどう動くかが記載されている

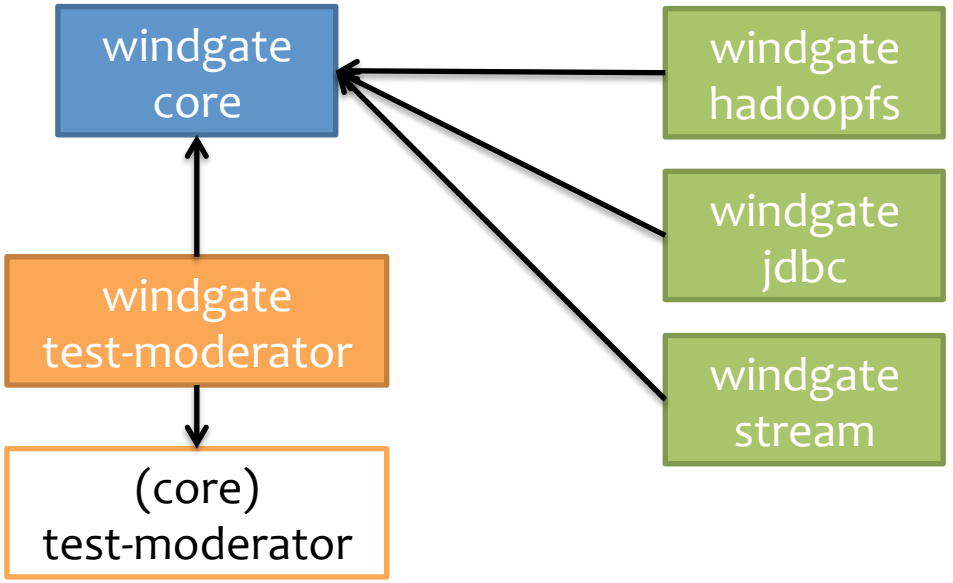
WindGateIoProcessor

- ...compiler.windgate.WindGateIoProcessor
  - in "asakusa-windgate-plugin"
  - 先ほどの語彙を解析するDSLコンパイラプラグイン
    - DriverScriptをかき集めてコンパイル結果に追加



WindGateImporterPreparator

- ...testdriver.windgat.WindGateImporterPreparator
  - in "windgate-test-moderator"
  - 先ほどの語彙をテスト時に処理するプラグイン
    - ResourceManipulatorを利用してテストデータの準備



WindGateまとめ

- ポータブルなデータ転送ツール
  - 様々なものをプラグインとして外部化
    - データソースと通信する部分
    - リソース間でデータを転送する部分
    - など
- DSLから透過的に利用可能
  - コンパイラがWindGate用のコードを自動生成
  - DSL上でのテストに対応

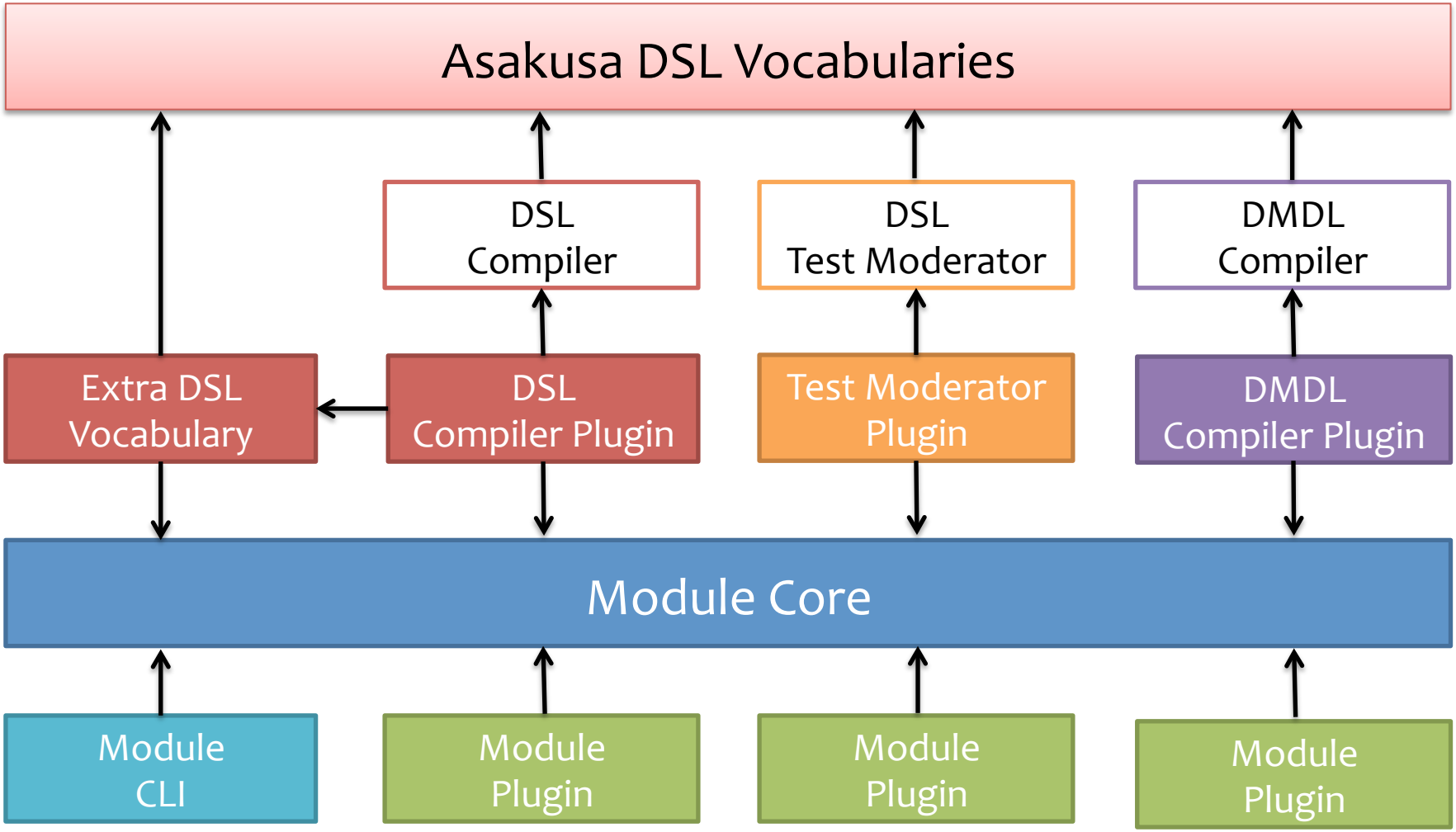
YAESSとWindGateの総括

全体のまとめ

全体のまとめ

- 「ポータブル」なツールを整備
  - YAESS: ポータブルなジョブ実行ツール
  - WindGate: ポータブルなデータ転送ツール
  - 「向こう側」がどうなっているのかを気にしない
- 外部から拡張可能
  - 個別にツールを作るよりは楽
  - 「向こう側」の変化に追従できうる
- 基本的にはAsakusa DSLと関係なく動く
  - どちらも依存関係の薄いツール
  - 本体のプラグイン機構でDSLとつなげている

メタフレームワークとしてのAsakusa



今後の方針

- 「プル型」のデータ転送ライブラリ
  - WindGateと逆
- DSLの整備
  - かゆいところに手が届くように
- コンパイラ周りの整備
  - もう少し強めの最適化を入れる
  - 8ヶ月くらい本体のコンパイラから離れていた