Asakusa Frameworkのご紹介

NAUTILUS

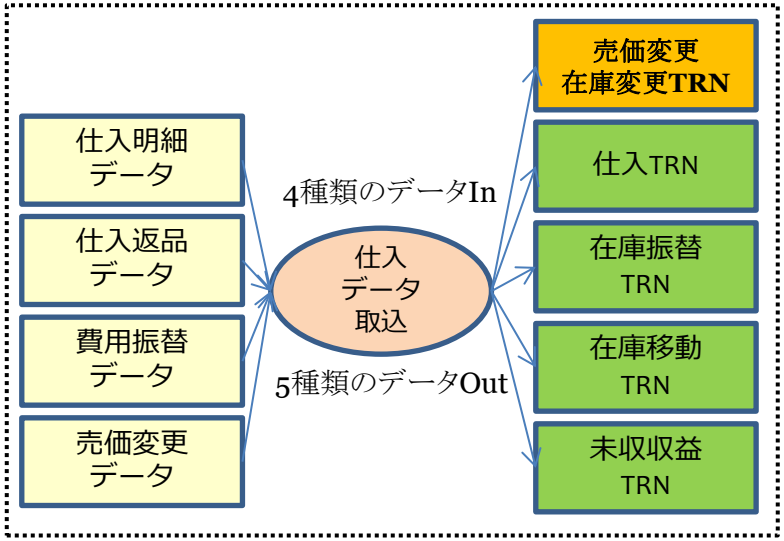
株式会社ノーチラス・テクノロジーズ
http://www.nautilus-technologies.com/
mailto:contact@nautilus-technologies.com
Tel: 03-6712-0636

Hadoop/分散処理のパワーを活かして解決する課題

- **既存バッチは遅く、もう早くするのは難しいと思っていた**
  - **増えるデータとバッチ処理の遅延の課題**
    - バッチ処理の遅延対策として、ハード増強以外に手が無い
    - 多重度の高い処理は、今までのRDBMSで処理が終わるか不安
  - **RDBMS、SQLのチューニングも限界**
    - データベースのチューニングも、アプリケーションの修正も限界
    - いずれ、アプリケーションを書き換えないといけない
- **時間が足りないから『やらなかったバッチ処理』がある**
  - 計算量が膨大になるから、バッチ処理できないと諦めていた
    - これ以上、データが増えると、翌日の業務処理に間に合わない
  - 業務側の時間を調整、運用で逃げて後回しにしていた
- **多額な費用が発生するから『やれなかったバッチ処理』がある**
  - DISKも高価で、容量も多過ぎて、データを溜めてなかっただけ
  - DWH・BIでの分析は、効果が確実に出る自信がないと予算をかけられない

Asakusa Frameworkが、なぜ必要だったのか？

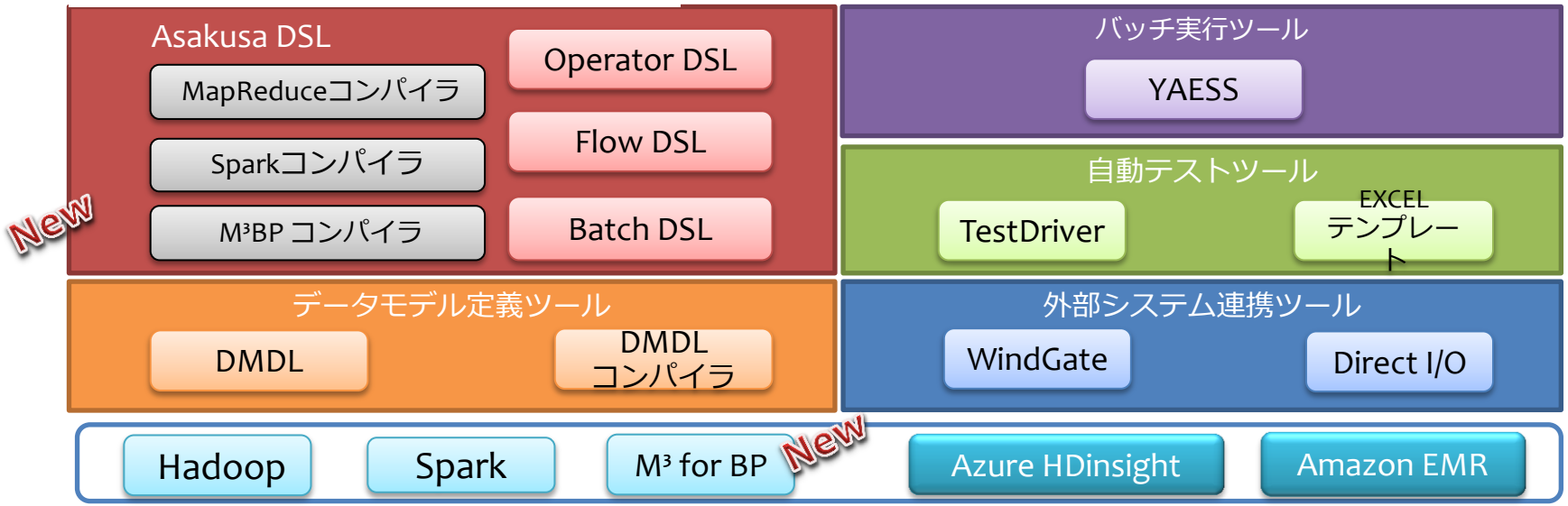
- **Hadoopを基幹バッチ処理に導入する際の問題点**
  - Hadoopには、複雑なバッチ処理の開発手法がなかった
    - 従来のHadoopは、簡単なログ集計用で複雑なバッチ処理の開発は想定外
  - MapReduce特有の設計・実装は非常に難易度が高い
    - MapReduce処理の動きが、そのまま業務処理に合わない
  - テストツールが貧弱
    - 複雑なアプリケーションをテストするツールが提供されてない



Asakusa Frameworkとは

- **分散環境用高速バッチ開発フレームワーク**
  - Asakusa Frameworkは業務システムのバッチ処理に、並列分散処理の能力を活用する開発用フレームワーク
  - Hadoop/Spark/M³ for BP等の分散処理に対応し、分散環境向けアプリケーション開発に必要な開発環境・実行環境・運用環境を用意
- **分散処理のアプリケーション開発を容易にし、更に高速化**
  - 分散処理のプログラムは、Asakusaのコンパイラが自動生成し最適化を実施
  - MapReduce/Spark/M³ for BPのコードを、１つのAsakusaDSLで生成可能

Asakusa Frameworkコンポーネント

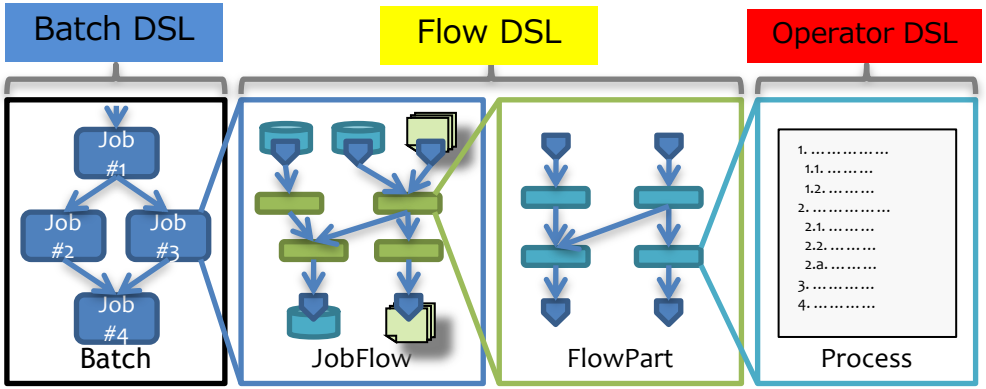


Asakusa Framework のコンポーネント

Asakusa DSL

- **Asakusa DSL**を中心としたフレームワーク
  - 設計は、バッチ処理をデータフロー形式で記述します
  - 入力データ、処理内容、出力データを記述した
    データフロー設計から、そのまま実装が可能
  - ビジネスロジック実装にMapReduceを意識しなくて良い

- 三層DSLによる**シンプルなJavaプログラム**の組合せ



  - 実装フェーズは、演算子 (Operator DSL) から作成し、
    演算子を組み合わせジョブフロー (Flow DSL) を作成
    ジョブフローの実行順序をバッチ処理 (Batch DSL)
    に記述します

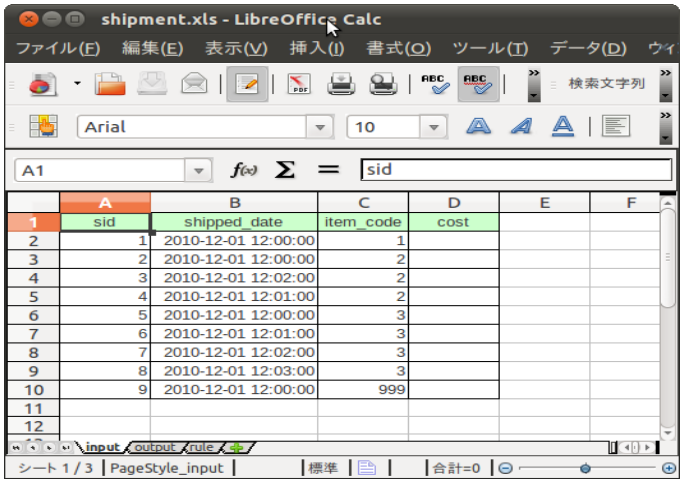
主なOperatorDSL (全26種類)	処理概要
分岐 Branch	レコードを内容に応じた出力に振分
更新 Update	レコードの内容を更新して出力
変換 Convert	別の種類のレコードに変換
マスタ結合 MasterJoin	マスターデータを結合
マスタ分岐 MasterBranch	レコードとマスタデータの内容に　応じて振分
マスタ付き更新 MasterJoinUpdate	マスターデータの内容に応じて振分
単純集計 Summarize	グループ化したコードを集計
グループ結合 CoGroup	複数のレコードをグループ化して任意処理

Asakusa Framework のコンポーネント

Asakusa Frameworkテストツール

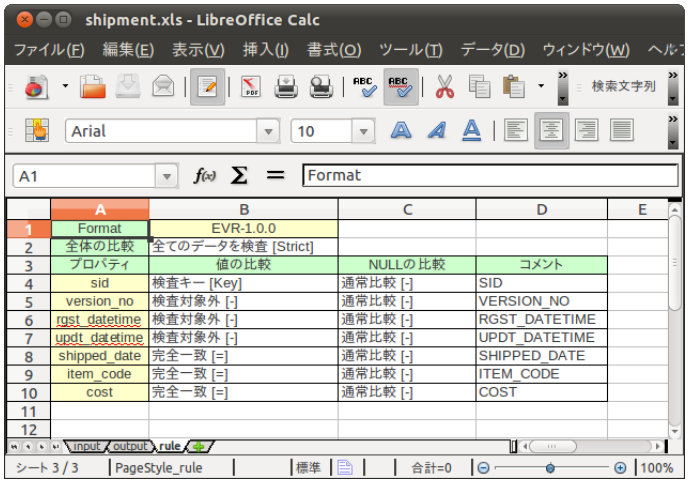
- Asakusa Framework では、Asakusa DSL の単位でのテストを行うための仕組みを提供
  - **演算子のテスト**
    - 演算子に対して通常の Java クラスと同様に、Junitでテスト可能
  - **データフロー、バッチのテスト**
    - DSLコンパイラ、 Hadoop を自動的に起動
    - 一連の処理を自動的に行う『**テストドライバ**』を提供
    - テストデータのひな形を Excel ファイルで定義

入力データ・期待データシート



	A	B	C	D	E	F
1	sid	shipped_date	item_code	cost		
2	1	2010-12-01 12:00:00	1			
3	2	2010-12-01 12:00:00	2			
4	3	2010-12-01 12:02:00	2			
5	4	2010-12-01 12:01:00	2			
6	5	2010-12-01 12:00:00	3			
7	6	2010-12-01 12:01:00	3			
8	7	2010-12-01 12:02:00	3			
9	8	2010-12-01 12:03:00	3			
10	9	2010-12-01 12:00:00	999			
11						
12						

比較条件シート

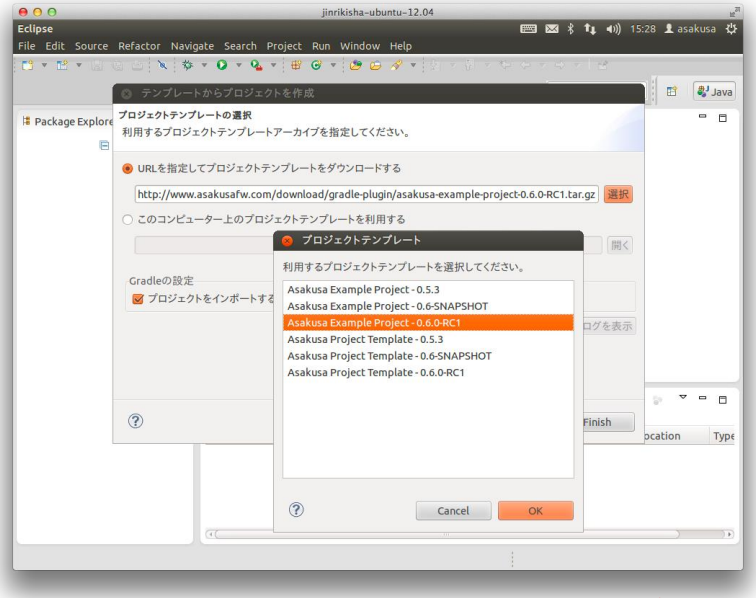


	A	B	C	D	E
1	Format	EVR-1.0.0			
2	全体の比較	全てのデータを検査 [Strict]			
3	プロパティ	値の比較	NULLの比較	コメント	
4	sid	検査キー [Key]	通常比較 [-]	SID	
5	version_no	検査対象外 [-]	通常比較 [-]	VERSION_NO	
6	rgst_datetime	検査対象外 [-]	通常比較 [-]	RGST_DATETIME	
7	updt_datetime	検査対象外 [-]	通常比較 [-]	UPDT_DATETIME	
8	shipped_date	完全一致 [=]	通常比較 [-]	SHIPPED_DATE	
9	item_code	完全一致 [=]	通常比較 [-]	ITEM_CODE	
10	cost	完全一致 [=]	通常比較 [-]	COST	
11					
12					

バッチ処理開発に必要なツールをALL in Oneで提供

- 開発環境は、一括インストールツール**『Jinrikisha』**を用意
  - http://asakusafw.s3.amazonaws.com/documents/jinrikisha/ja/html/index.html

開発・テストに必要なツールをAll in One パッケージに



既存の他システムとの連携

RDBMS連携、HDFS連携モジュール提供
Asteria/DataSpider/Hulft等のEAI/ETL連携

ジョブ管理ツールとの結合

JP1やSystemWalker、千手、A-Auto等のジョブ管理製品との連携

Windows環境で開発・テスト・ビルド

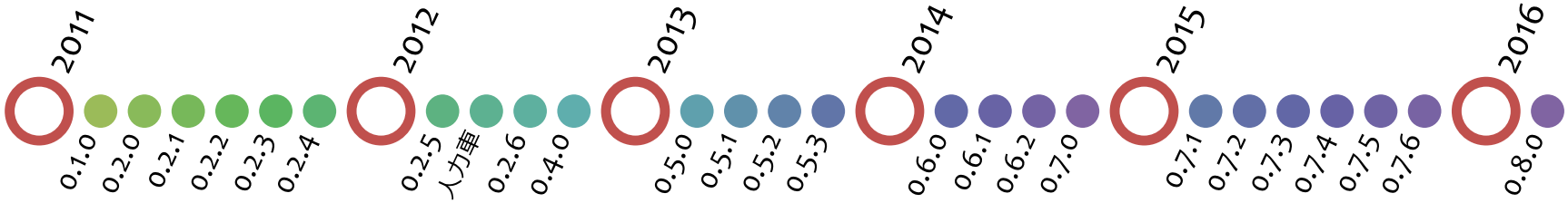
Eclipseプラグイン『Shafu』を用意
バッチテストランナーにてテスト時間を短縮

強力な補完・チェック機能

Asakusa Frameworkは標準的なJavaプログラムであり、IDEの補完・エラーチェック機能を100%活用できる

Asakusa Frameworkのリリースと歩み

- 2011/3: OSSとして、初版 (Version 0.1.0) リリース
  - 2-3ヶ月毎に、安定したリリースを実施
  - バッチ処理、データ活用の外部連携、各種分散処理エンジンに対応



時期	他社製品との連携・サービス追加・協業
2011年11月	日立ソリューションズ様　統合運用監視　「JP1」と連携
2012年１月	EMCジャパン様とEnterprise Hadoop開発運用ソリューション「GreenplumHDEE」連携
2012年4月	BSP様　ジョブ管理ツール　「A-Auto」と連携
2012年7月	アプレッソ様　ETLツール　「DataSpider　Servista」と連携
2012年10月	IIJ様　「GIO　Hadoopソリューション」に採用
2012年11月	インフォテリア様　ETLツール　「Asteria」と連携
2013年4月	AsakusaのPaaSサービス「Node0 DBR」を開始
2014年11月	NTTコムウェア様　「SmartCloud」に採用
2015年7月	**Asakusa on Spark リリース**
2016年4月	**Asakusa on M³BPリリース**

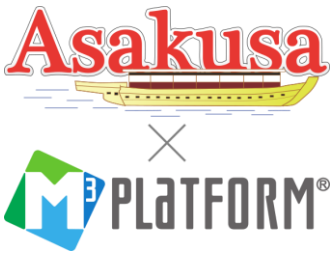
Asakusa on Spark

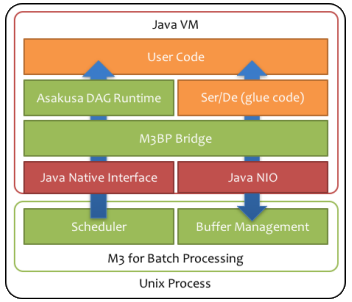
- **Asakusa on Spark**
  - Asakusa DSLをはじめとするAsakusa Framework開発基盤を利用し、**既存のAsakusa DSLを一切改修せず**にSpark処理基盤で利用可能な機能セットを提供
  - Asakusa Framework0.8.0より、Developer previewから正式機能に
- **Sparkの優位性**
  - 業務系バッチにおける複雑な処理、数十GBレンジのデータボリュームにおいて、既存**MapReduceよりも高速処理**が可能
  - まったく同一のAsakusa DSLで、MapReduce、Sparkの両環境で動作可能のため、速度の比較検証も容易に可能で、バッチアプリケーションごとに**適切な分散処理基盤を選択可能**

Asakusa Framework新機能

Asakusa on M³BP

- 分散処理エンジンをフィックスターズ社と共同開発
  - マルチコア環境でDAG形式で並列処理をするインメモリエンジンを提供
    - 小規模データでの複雑な処理を、単一ノード上のマルチコア用に最適化
  - C++でネイティブアプリケーションで動作させて高速化
  - M³ for BPの実行環境には、Hadoop/Sparkは不要
    - クラスタの構築・運用する場合の課題が解消され、高い費用対効果を生む
- 単一ノード・マルチコア環境用のAsakusa on M³BP
  - Asakusa on M³BPにてCPUコアの制御やメモリ管理はC++でM³ for BPで処理し、アプリケーション部分はJVMに振分け高速化
  - Asakusa DSLには一切変更を加えることなくM³ for BPと、Hadoop/Sparkの処理基盤を切り替えて使うことが可能

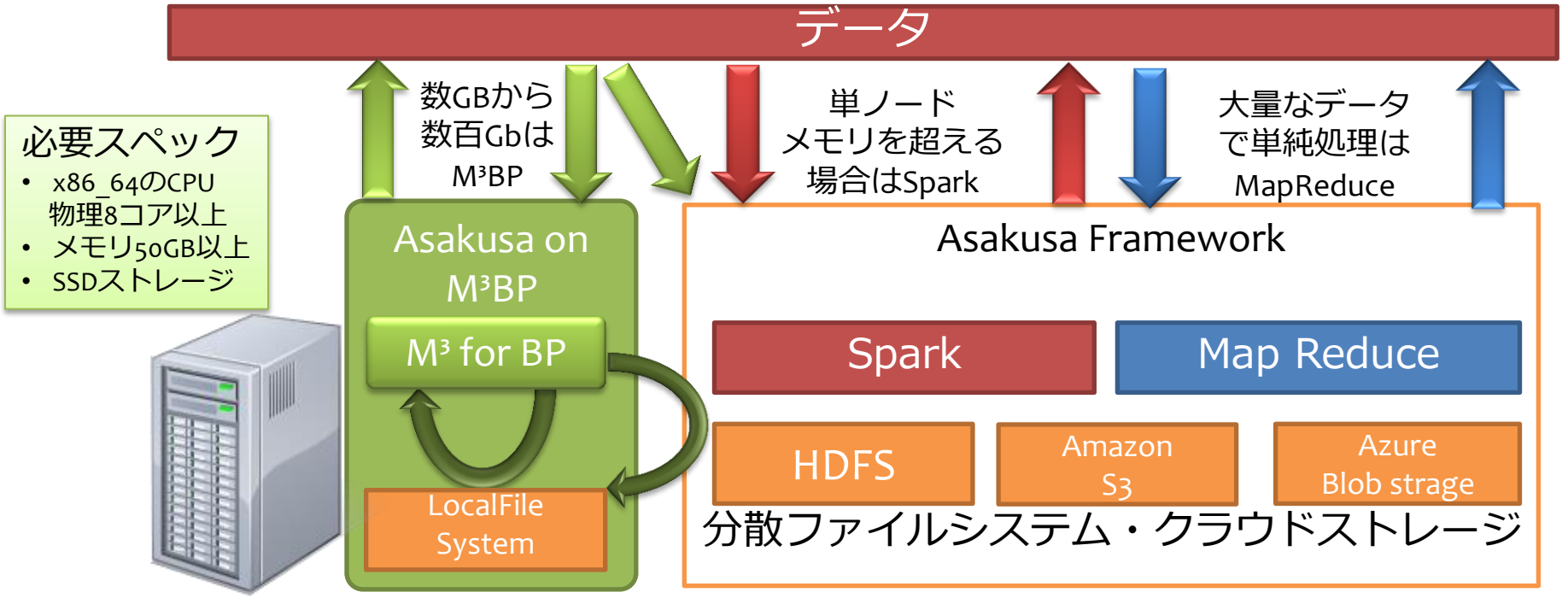




単一ノード・マルチコア・大量メモリにて、小規模バッチ処理を
Sparkよりも高速処理するM³ for Batch ProcessingとAsakusa on M³BP

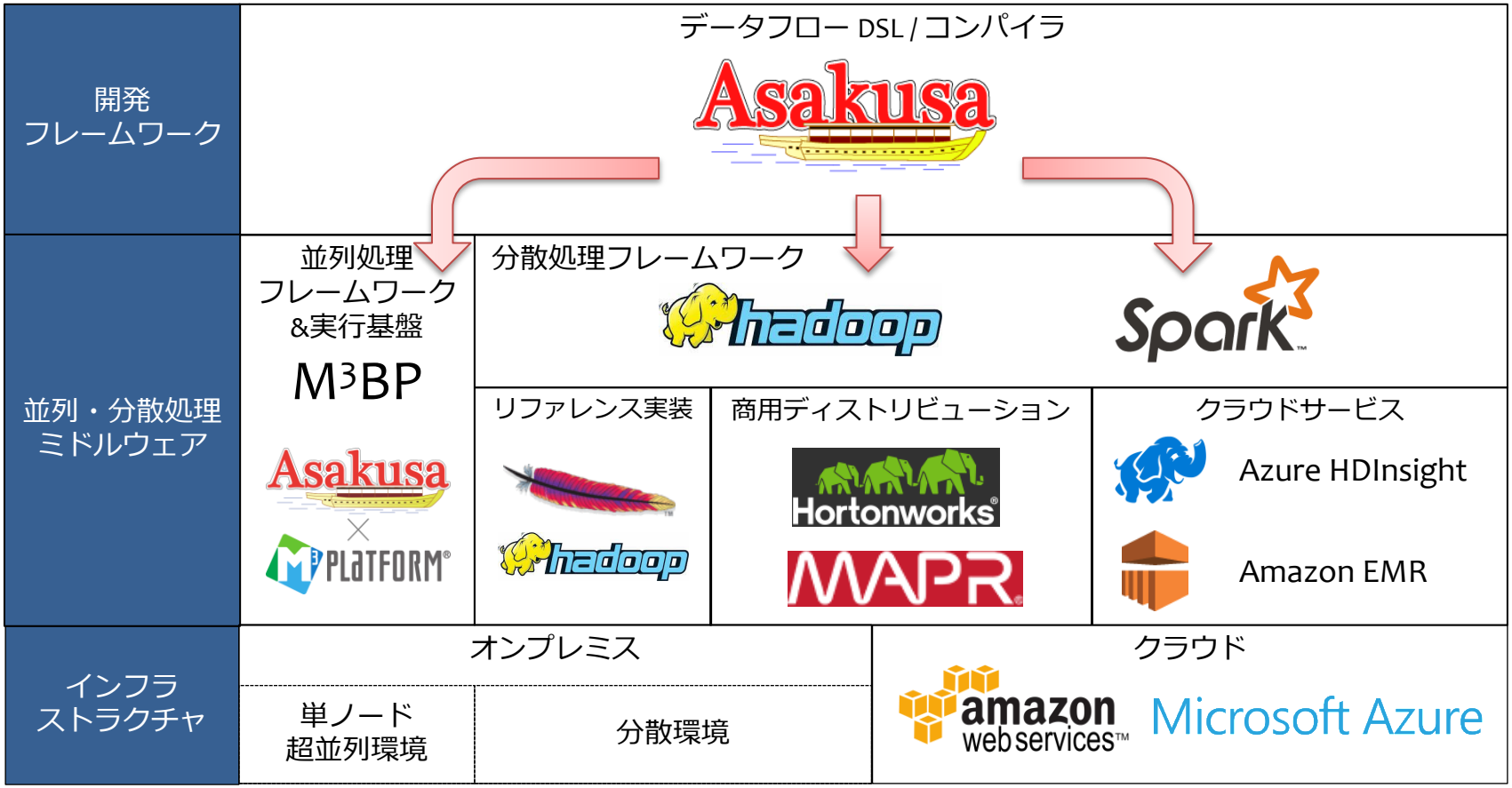
Asakusa on M³BPの動作環境

- ローカルと分散の両方にデータアクセス可能
  - 単ノードで、Linuxのローカルファイルシステムやマウント可能なストレージにアクセス
  - データだけ分散ファイルシステム上に配置し、M³ for BP単ノードで実行をすることも可能
    - クラウドも利用可能だが、性能面で物理コンピュータを推奨
    - Hadoop/Spark上で、Asakusa on M³BPを動作させるのは非推奨(リソースの干渉)
- 数GBから～数百GBのデータについてM³ for BPを利用、データ量が増えたらアプリケーションに負荷をかけずに、Hadoop/Sparkにエンジンを切り替え複数ノードにスケールアップ可能



Asakusa Frameworkのポータビリティ

様々な分散・並列処理基盤をAsakusa Frameworkにより選択可能に



Asakusa Framework導入イメージ

Asakusa Frameworkの想定ケース

Asakusa Frameworkの想定活用ケース

NAUTILUS

Asakusa Frameworkの想定される活用領域

例えば、月次でしか実行出来なかったシミュレーション・確定処理・クレンジング・引当・受発注・在庫管理・予測といった処理も、日次あるいは即時で実行可能
今まで、時間的あるいはデータ量的に、実現が難しかった集計・分析が、Hadoopの分散処理により、高速処理して、経営判断に必要な情報がより得やすくなります

基幹バッチ処理型

既存バッチ処理リプレース、ホストリプレース

- DISK　I/Oボトルネックの解消
- データベースのチューニングも限界
- ハード増強は、費用が高く頭打ち
- ホストからのバッチ移行時の代替え

データウェアハウス型

既存DWHのオフロード、またはDWHの新規構築

- オンライン処理時間の延長要請
- DWHに投入するデータ量の増加
- DWHの前処理バッチでの高速化
- 高価なDWHやBIツールを導入したくない
- 高価な、ストレージ製品等を導入したくない

ビッグデータ型

大量データ、計算量が多い新規バッチ処理

- 毎日増え続けるログデータの蓄積
- 逐次更新される会計データの集計
- データベースでは処理が遅い複雑なデータの処理
- 各種センサーログ、利用記録等

シミュレーション型

業務データの粒度を詳細化してシミュレーション

- 詳細な業務データの蓄積
- 様々な分析軸の立案
- 様々なパターンの検証
- 組み合わせ数が爆発する計算処理

AsakusaFramework導入事例

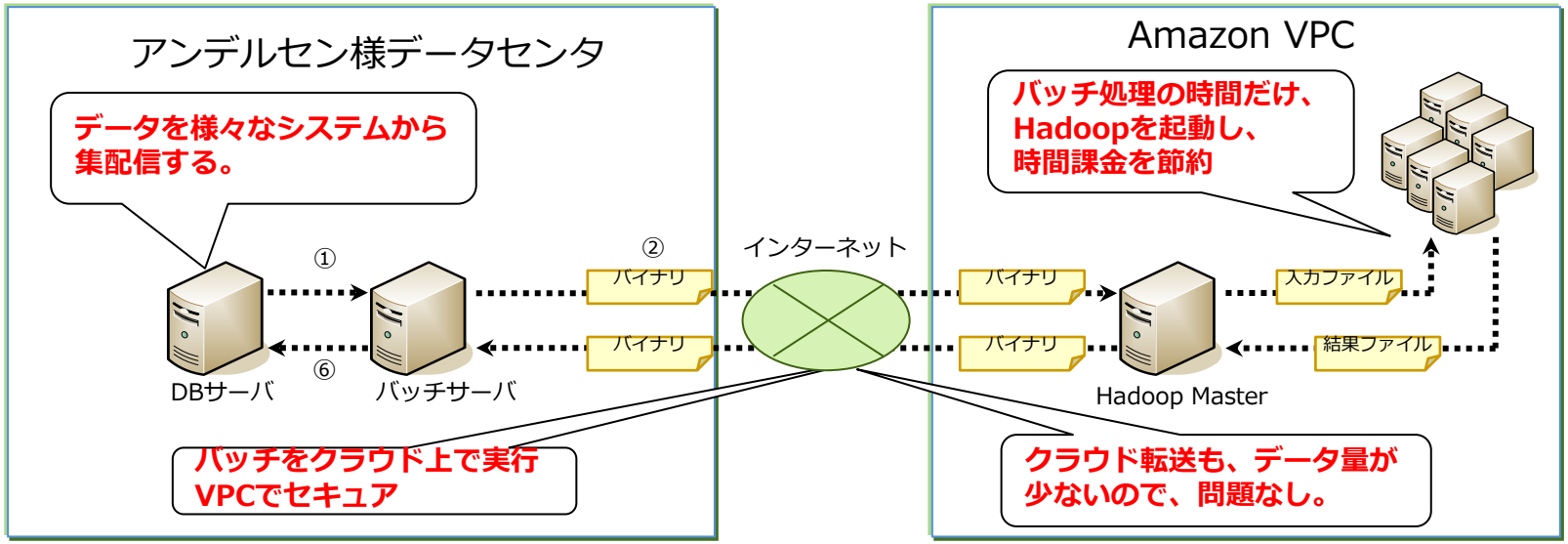
- 金融業、社会インフラ、製造業、流通業など様々な業界にご採用いただいております

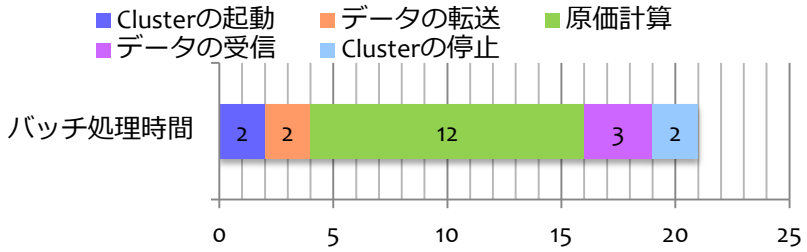
にしてつストア 本部基幹・会計システム	CTC Challenging Tomorrow's Changes クラウド高速データ処理基盤	通信キャリア パケット料金計算システム
ANDERSEN 原価計算・生産管理システム	大手金融機関 与信管理シミュレーション	設備サービス 基幹　年次バッチ処理
九州電力 設備巡視計画書作成システム	さくらインターネット データセンター原価計算処理	ネット通販 原価計算システム

アンデルセン様　原価見える化事例

原材料の製品原価計算バッチが遅い

- 原材料からの製品原価計算で４時間かかっていた
  - 110万の原材料、3000品目の原価をツリー構造の積上げ計算を実施
  - BOMの展開・原価の４時間バッチで、週に２回実行するのが限界





4時間のバッチ処理が
20分で終了

アンデルセン様　原価見える化事例

緊急の調達先の変更への対応と原価シミュレーション

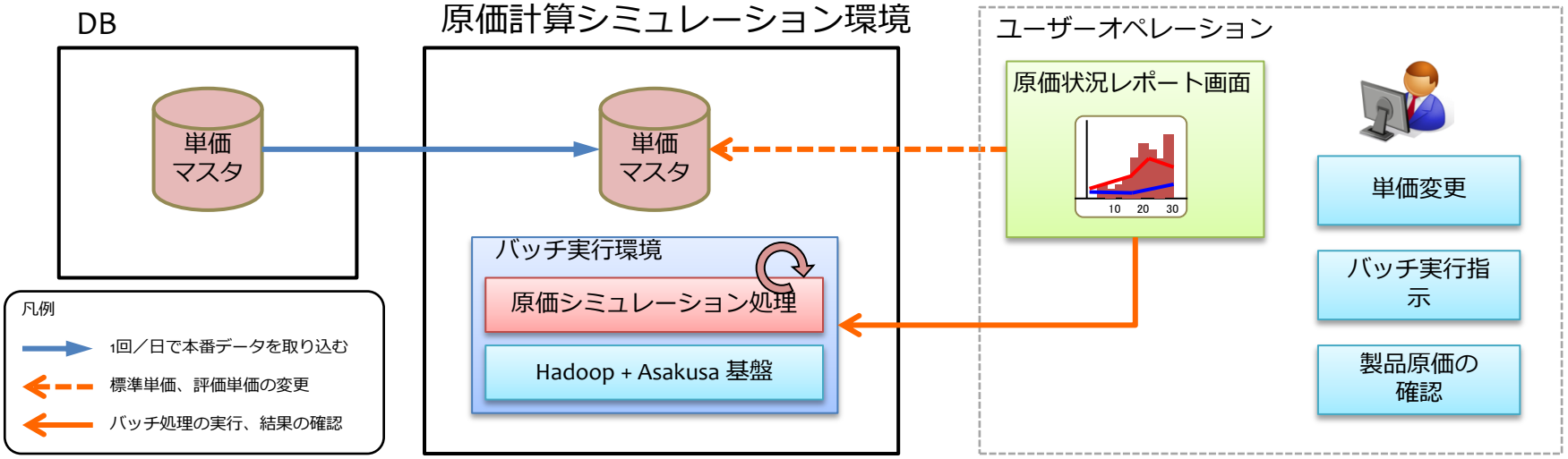
原価計算の課題

- 3・11が原因で、仕入先の変更と原材料の高騰
- 原価の変動に伴う影響をシミュレーションしたいが、バッチが４時間かかっていた。
- H/W増強しても現状のSQLではそれほど高速化しない

原価シミュレーション

- Hadoop・Asakusa Framework でバッチ処理を高速化
- 経営企画部門が、変更したい原価を設定して、影響額を即時に把握することが可能になった。
- CSVで全変更価格を取り込んでのバッチ処理も、画面での原価入力も可能な環境を作成

このデータを更に活用したい



アンデルセン様 原価見える化事例

課題と効果

工場での原価把握に対する課題

- ✓ 棚卸を実施しないと材料の使用量や原価率が分からない
- ✓ 月次でしか原価率の把握ができていない
- ✓ 材料の使用実績はわかるが製品毎の標準的な使用量が把握できない
- ✓ システムが複数ありデータが分断されている
- ✓ 実績の登録漏れや構成表(BOM)が不整備

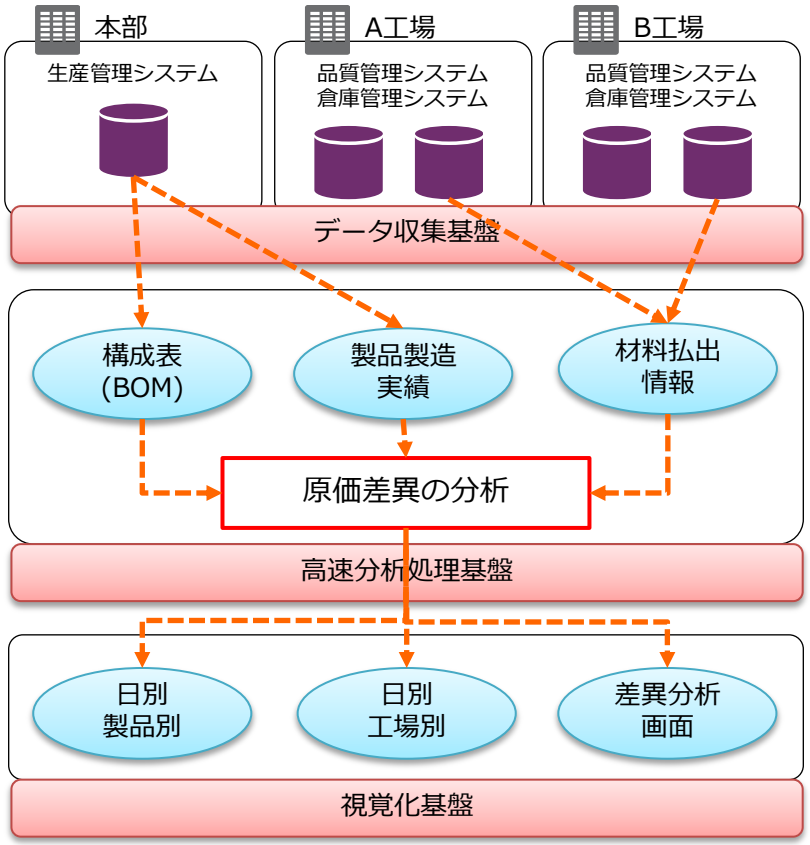
ロスや棚卸差異の理由を忘れて原因分析ができない
どのライン、製品にロスが多いか判断できない
結果として打ち手が取れない

課題に対するAsakusaのアプローチ

- ✓ 生産管理や品質管理システムからの実績収集
- ✓ 構成表(BOM)を展開し、製品単位での材料使用量の把握
- ✓ 上記データを元にした日次での原価集計処理
- ✓ 材料の使用実績と論理使用料の差異計算処理
- ✓ 工場、ライン、製品までのドリルダウンによる見える化

原価差異を日々正確に可視化し、データ不備、登録業務の是正、製品単位でのロス把握など運用業務の課題を可視化

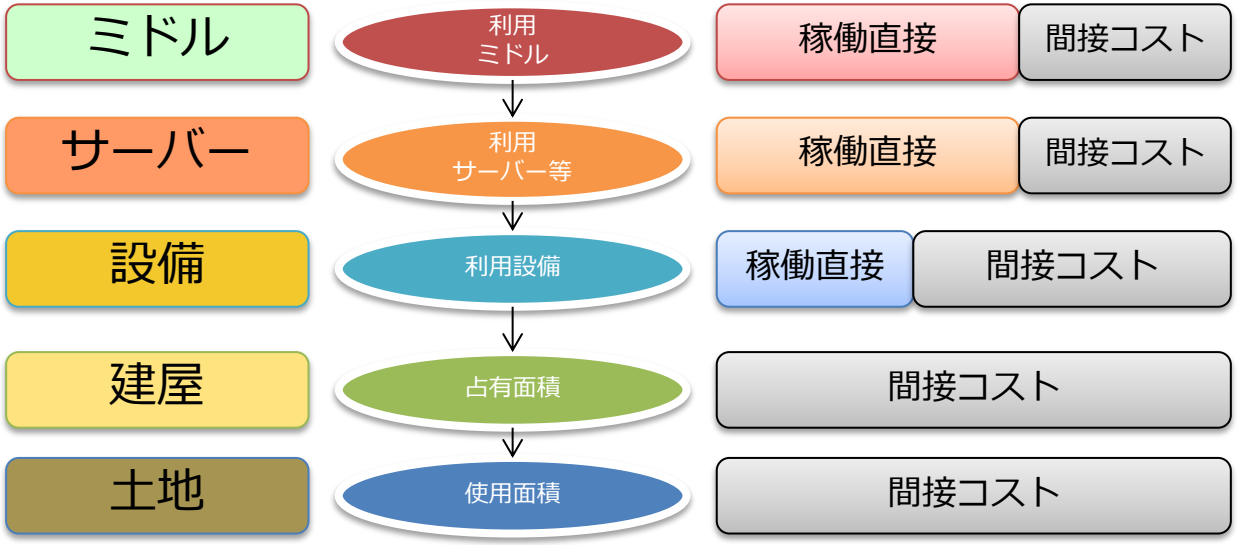
施策の検討、業務改善の実施といったPDCAサイクルを実現



さくらインターネット様

原価分析ソリューション事例

- **データセンターの原価処理の課題と対策**
  - 土地、建屋、設備、サーバー、ソフトに電力料金を積み上げて原価を計算してきた
  - サービス改善、新サービスの追加するために、ユーザ単位の原価把握が必要
    - どのユーザが、どの程度利用をしているのか？が不明で、追加投資ができるかどうか不明
- **データセンターの管理会計に挑戦**
  - 電力量、トラフィック量などの100数十万のログデータをユーザ単位で集計する
    - 各構成コストの原価ドライバーの因果関係のモデル化する
    - ハードウェア/ソフトウェア/人件費/修理費の会計データを積み上げ、ユーザ毎に利用量を振り分ける
    - 各ファクターの限界コスト（マージナルコスト）を算出
      - 例）サーバーを一単位増やした場合の追加的に発生するコスト

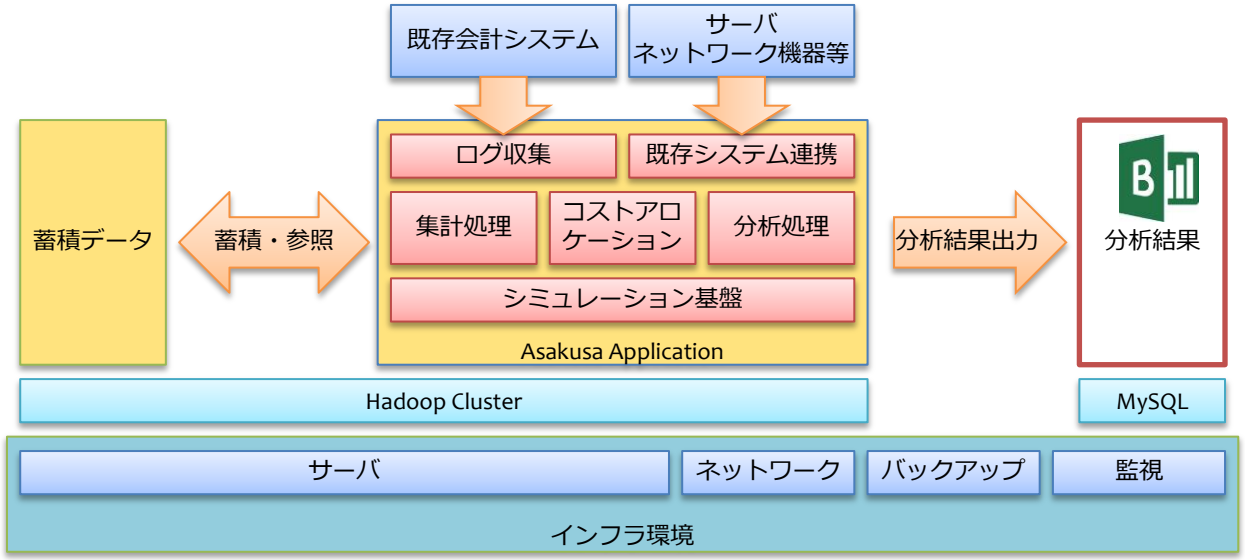


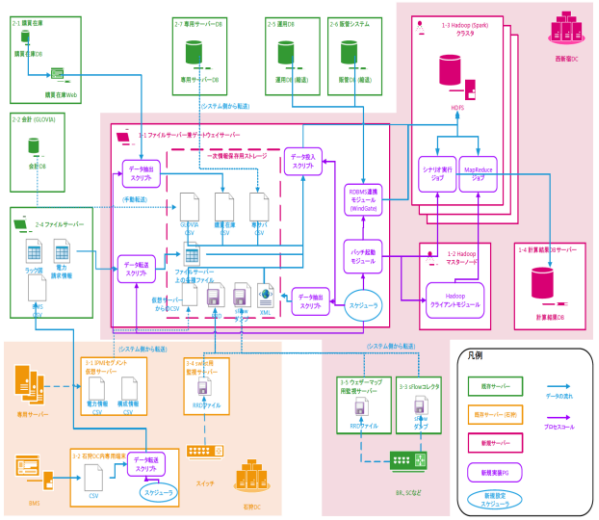
発生コストを
稼働ベースの
因果関係で
モデル化する

さくらインターネット様

原価分析ソリューション事例

- **バッチシステム概要**
  - 電力量やトラフィック量等の膨大なログデータと、ユーザ毎の会計データを紐つけ原価を算出するシステムを構築し、最初に石狩データセンターで提供されるVPSと専用サーバの原価計算を実施
  - RDBMSで想定20時間かかるバッチをMapReduceで１時間まで短縮。更に、Sparkを使って約10分まで短縮
- **原価分析ソリューションの成果**
  - データセンター毎の原価が把握でき、データセンター別、サービス別の利益率にも差があり、対外接続先が多くあるデータセンターほど原価が多く発生していることが明確になった





お問い合わせ

Asakusa Framework コミュニティと提供サービス

コミュニティによる勉強会や、OSS団体活動も活発化

- ユーザ主導にて勉強会も開催されています。

- オープンソースビジネスを推進する団体であるOSSコンソーシアムにて2014年7月よりAsakusa Framework部会の設立
  - Asakusaパートナーと参加企業メンバー
    - 株式会社日立ソリューションズ
    - 新日鉄住金ソリューションズ株式会社
    - 東芝ソリューション株式会社
    - NTTコムウェア株式会社
    - アクセンチュア株式会社
    - 伊藤忠テクノソリューションズ株式会社
    - みずほ情報総研株式会社

Asakusa Frameworkトレーニング

- Asakusaを触って見たいという方向けの入門編トレーニングコース
  「Asakusaによる分散バッチアプリケーション開発入門」を
  日本サード・パーティ株式会社様との協業にて開催しています。
  - 二日間のコース　座学とハンズオン
    - **１日目　Asakusa Frameworkの講義実習**
      - Asakusa Framework 概要
      - Asakusa Framework 開発環境の準備
      - Asakusa Data Model
      - Asakusa DSL
      - Asakusa Framework アプリケーションテスト
    - **２日目　Asakusa Framework ハンズオン実習**
      - ハンズオン形式で実際にAsakusa Frameworkの導入からバッチのサンプルプログラムを作成し、テストを行うまでを実習します
        - ハンズオンに必要なPC、受講に必要なテキスト等は、準備いたします
  - 毎月一回程度の開催
  - 受講費用
    - 198,000円／１名

Nautilus-technologies提供サービス

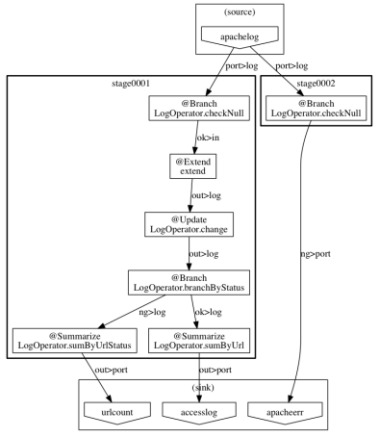
- 弊社が提供する主要なサービス

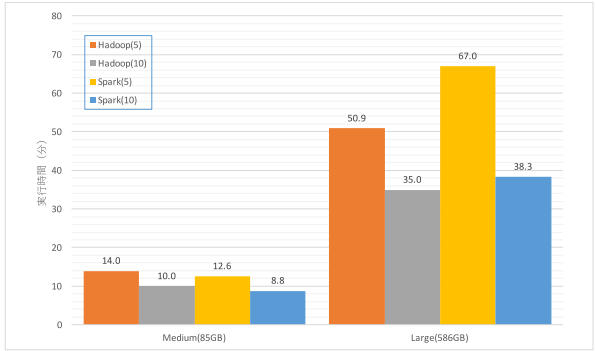
サービス	内容
Asakusa Framework サブスクリプションサポート	Asakusa Frameworkを、有償サブスクリプションとして提供し問合せ対応やV-up版の提供をします
Asakusa Framework プロフェッショナルサービス	• Asakusa Frameworkでの導入を前提に、要件定義やAsakusa DSLでの設計の支援を行います • Asakusa Frameworkでのアプリケーション開発をお客様で実施される場合に、問合せ対応、レビュー等の技術支援を行います
Asakusa Framework PoC プルーフオブコンセプト	Asakusa Framework/Hadoopでシステム導入の前に、実現性の可否を確かめるための検証を行います
Asakusa Framework アプリケーション開発	Askausa Frameworkを使用したアプリケーション開発をします
Hadoop販売/保守	Hadoopのサブスクリプション(MapR,HDP)の販売/保守の提供を行います
トレーニング/PaaS	• Asakusa Frameworkトレーニング(入門編) • クラウドプラットフォームサービス(Node0 DBR)

Asakusa Frameworkの導入・展開支援

- 現行バッチの分析、Asakusaによる実装
  - Asakusaの適用範囲、実装イメージを掴むサンプル提供も可
  - 設計例や考慮事項などを提示します（性能検証：オプション）

現在	Asakusa	
データ展開処理	@Update もしくは不要	Hadoop/Sparkは転送時にはデータを圧縮するため、自前の圧縮処理は不要となるケースも。独自形式の展開は必要。
保険料集計処理	@Summarize @Fold	単純なGroup Byであればsummarize。より複雑な処理が必要な場合はFoldを使う
当月処理対象引き当て処理	@MasterJoinUpdate @CoGroup	引き当てにはJoin系演算子を使う。複雑ならCoGroup





- Asakusaカスタム開発のご相談など
  - 特殊入出力形式への対応
  - 個社機能追加要望

Asakusa Framework お問い合わせ

- **Asakusa Framework**
  - Apache2.0ですので、どうぞご自由にお使いください
    - http://www.asakusafw.com/
      - 各種技術資料も豊富な、専用サイトを用意
      - 無料問合せ可能なメーリングリストを用意
    - https://github.com/asakusafw
      - gitHub上に、公開しています