2015 AsakusaFramework Day

金融機関様向けリスク管理システムへのAsakusaFramework適用事例

２０１５年１１月２７日

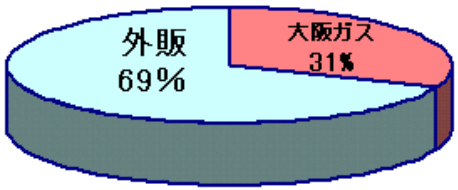
株式会社オージス総研
ソリューション開発本部
エンタープライズソリューション第三部
第二チーム
別所 泰輔

オージス総研

オージス総研のご紹介

株式会社オージス総研

- 代表者：　代表取締役社長　西岡　信也
- 設　立：　1983年6月29日
- 資本金：　4.4億円（大阪ガス株式会社100%出資）
- 事業内容：　システム開発、プラットフォームサービス、コンピュータ機器・ソフトウェアの販売、コンサルティング、研修・トレーニング
- 主な事業所
  - 本　社：　大阪府　大阪市西区千代崎3-南2-37 ICCビル
  - 東京本社：　東京都　港区港南2-15-1 品川インターシティA棟
  - 千里オフィス：　大阪府　豊中市新千里西町1丁目2番1号
  - 名古屋オフィス：　愛知県 名古屋市中区錦1-17-13 名興ビル
- 売上実績：　581億円（連結）　307億円（単体）※2014年度
- 従業員数：　3,126名（連結）　1,279名（単体）※2015年3月31日
- 関連会社：　さくら情報システム（株）、（株）宇部情報システム、（株）システムアンサー、OGIS International,Inc.、上海欧計斯軟件有限公司（中国）
- オージス総研グループ　売上構成比　（連結）



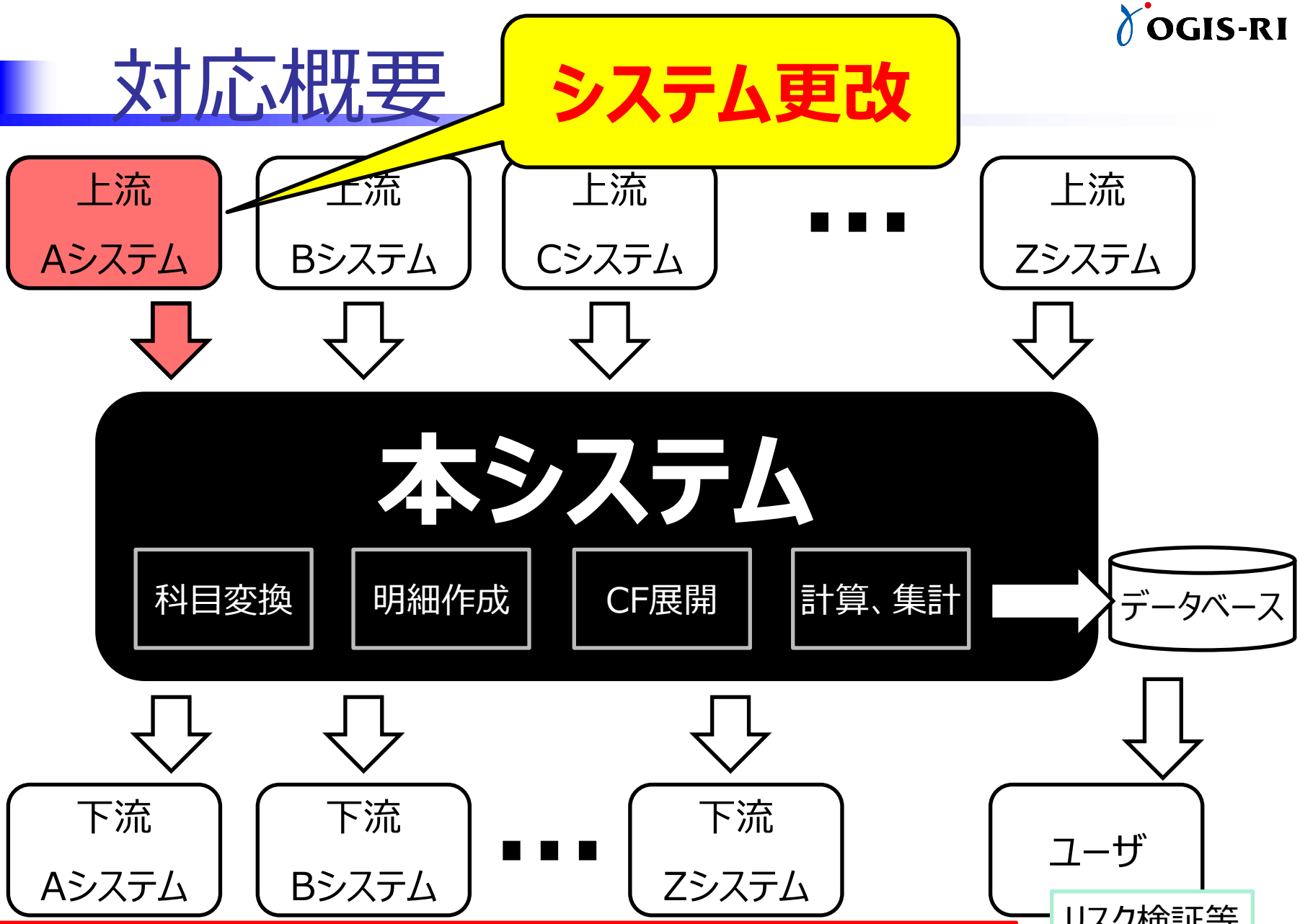
【特徴】
・公共性の高いガス事業のシステム構築
・メーカーに依存しない中立的な立ち位置

取得許可認定

目次

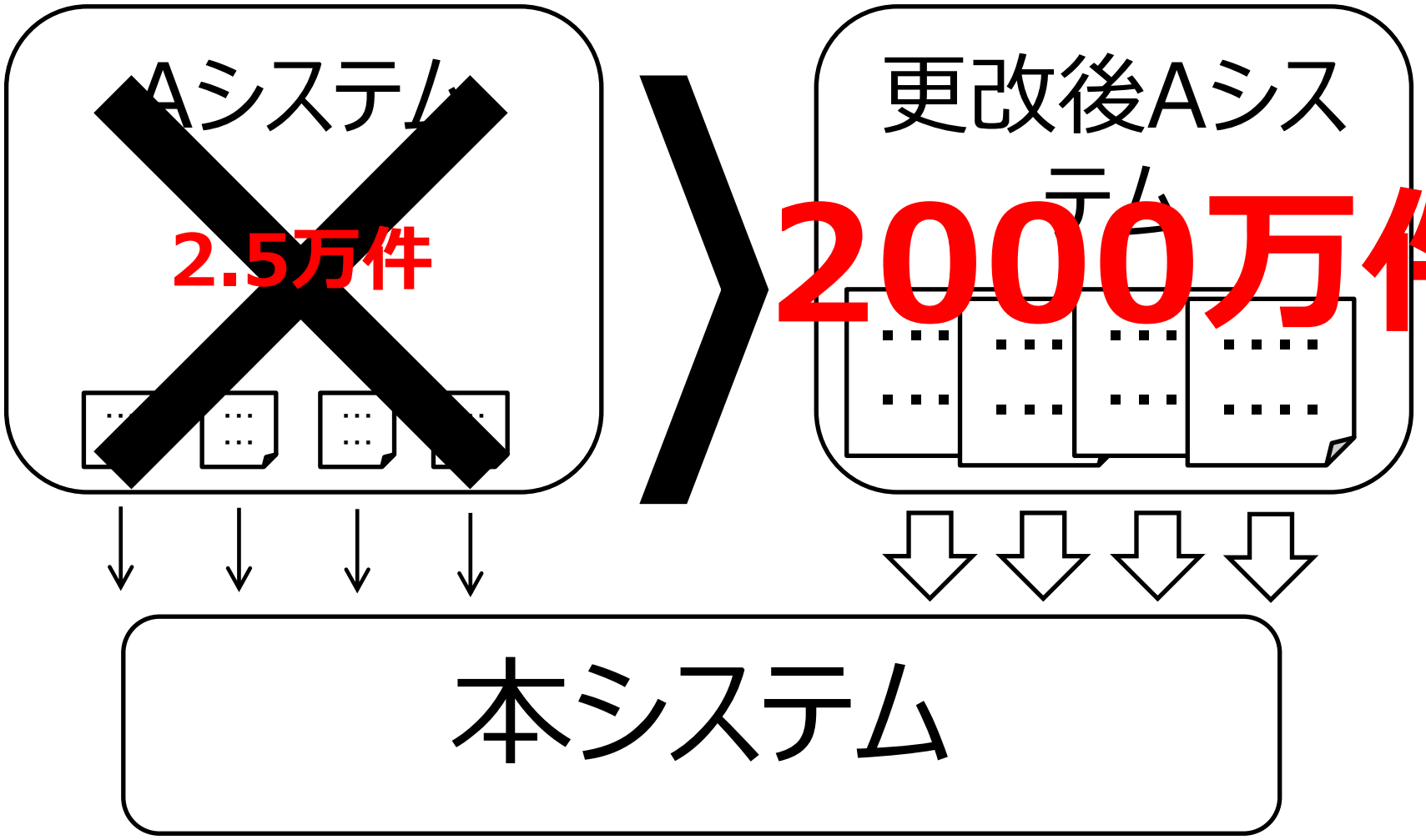
- 対応概要
- 対応方式検討
- 本当にできるの？
- 結果
- 今後の展開

対応概要



上流システム更改に伴うI/Fデータ変更

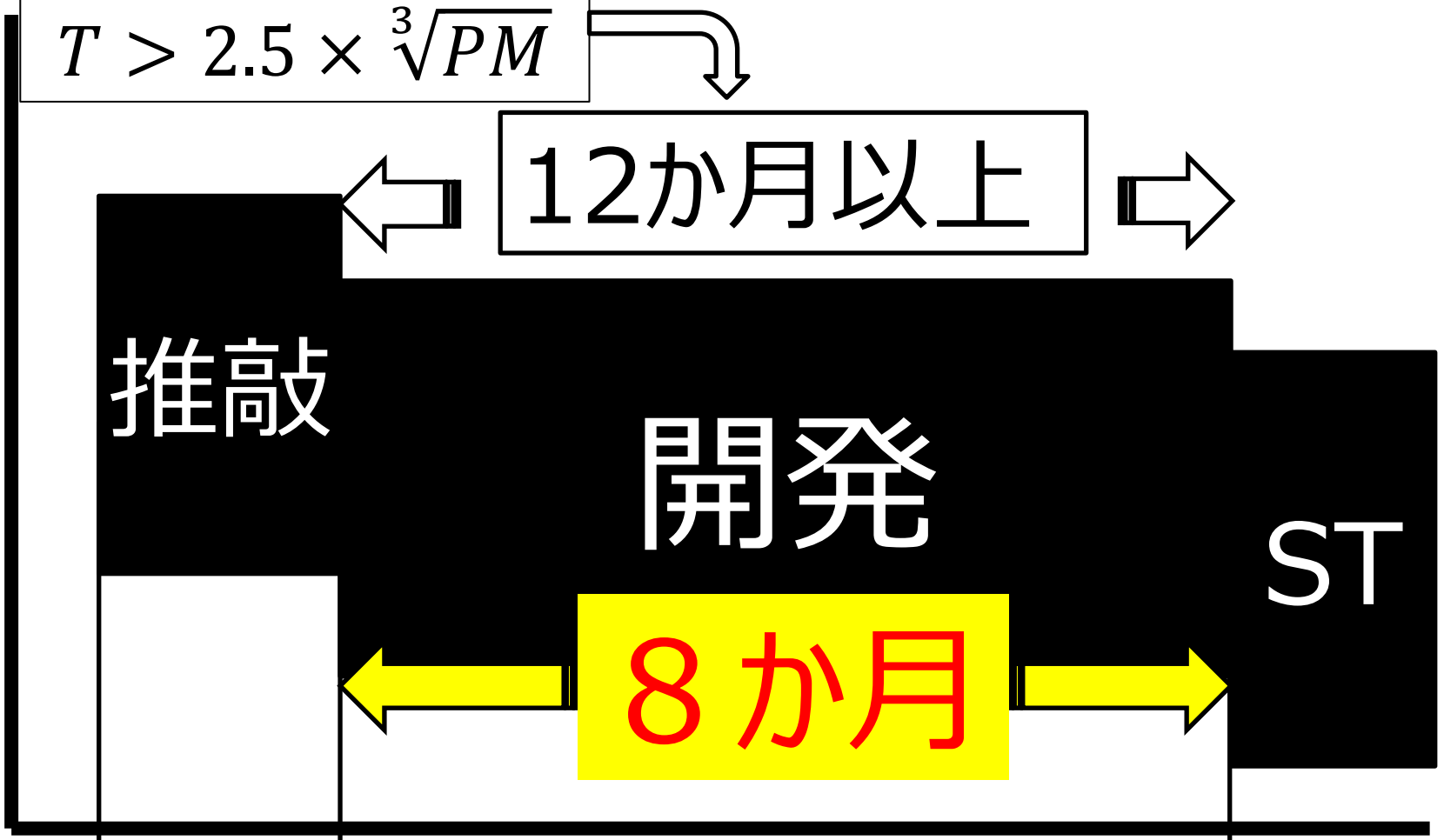
対応概要（要求）



処理対象データ８００倍

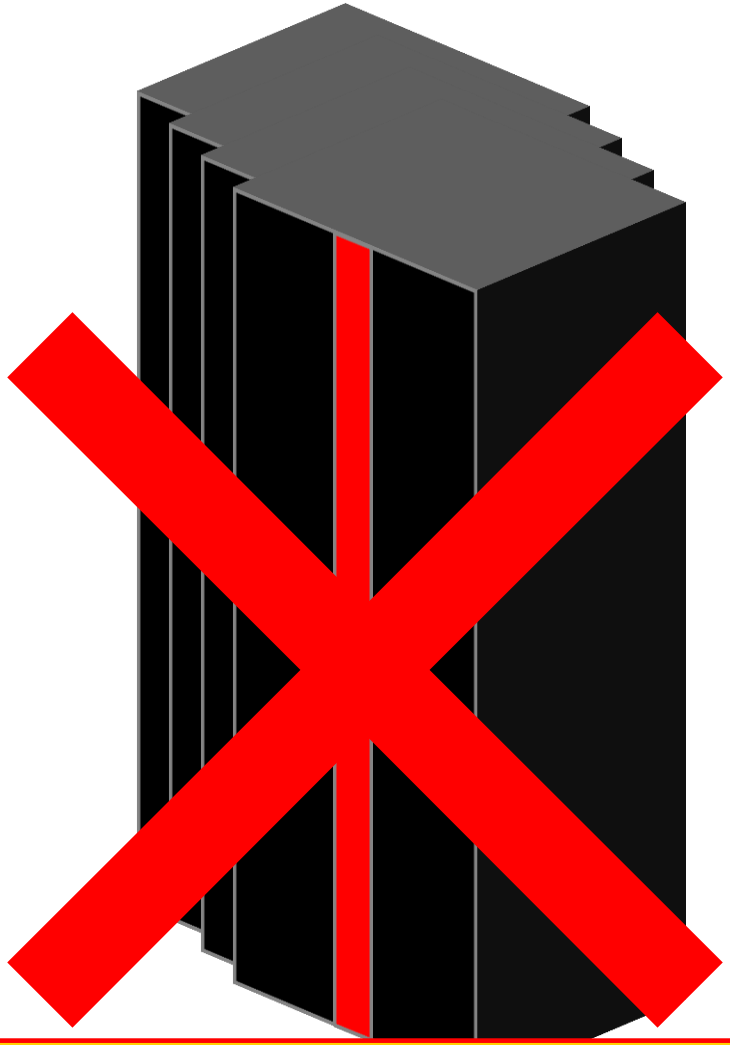
対応概要（要求）

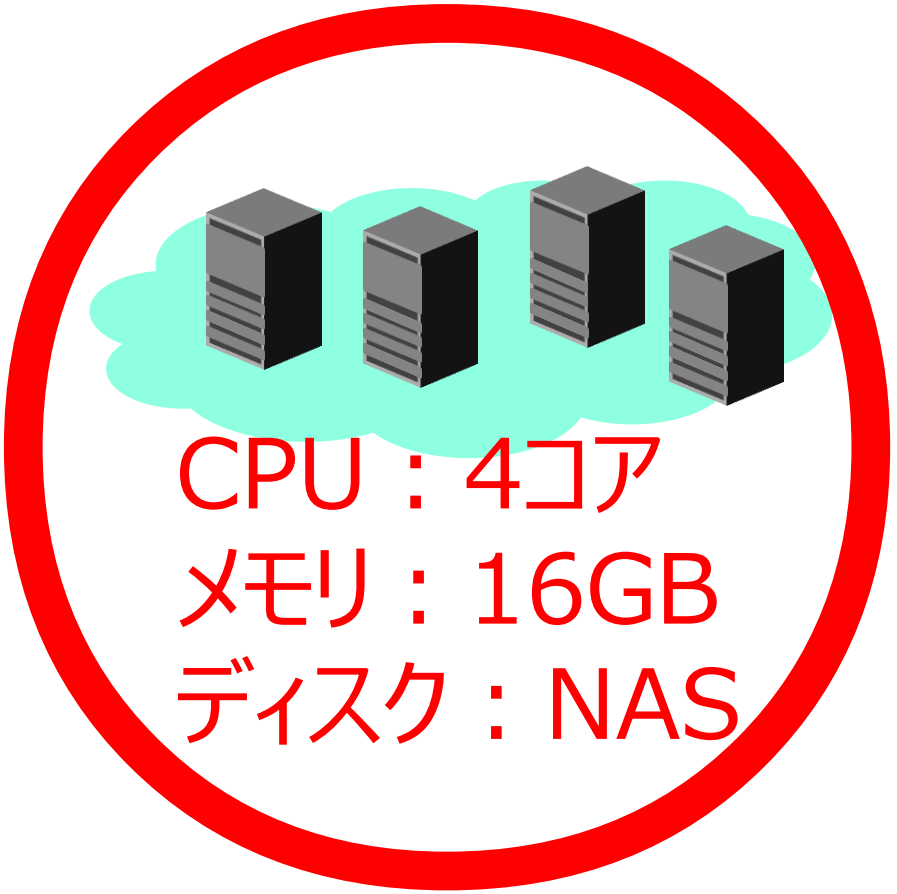
$$T > 2.5 \times \sqrt[3]{PM}$$



短い開発期間

対応概要（要求）



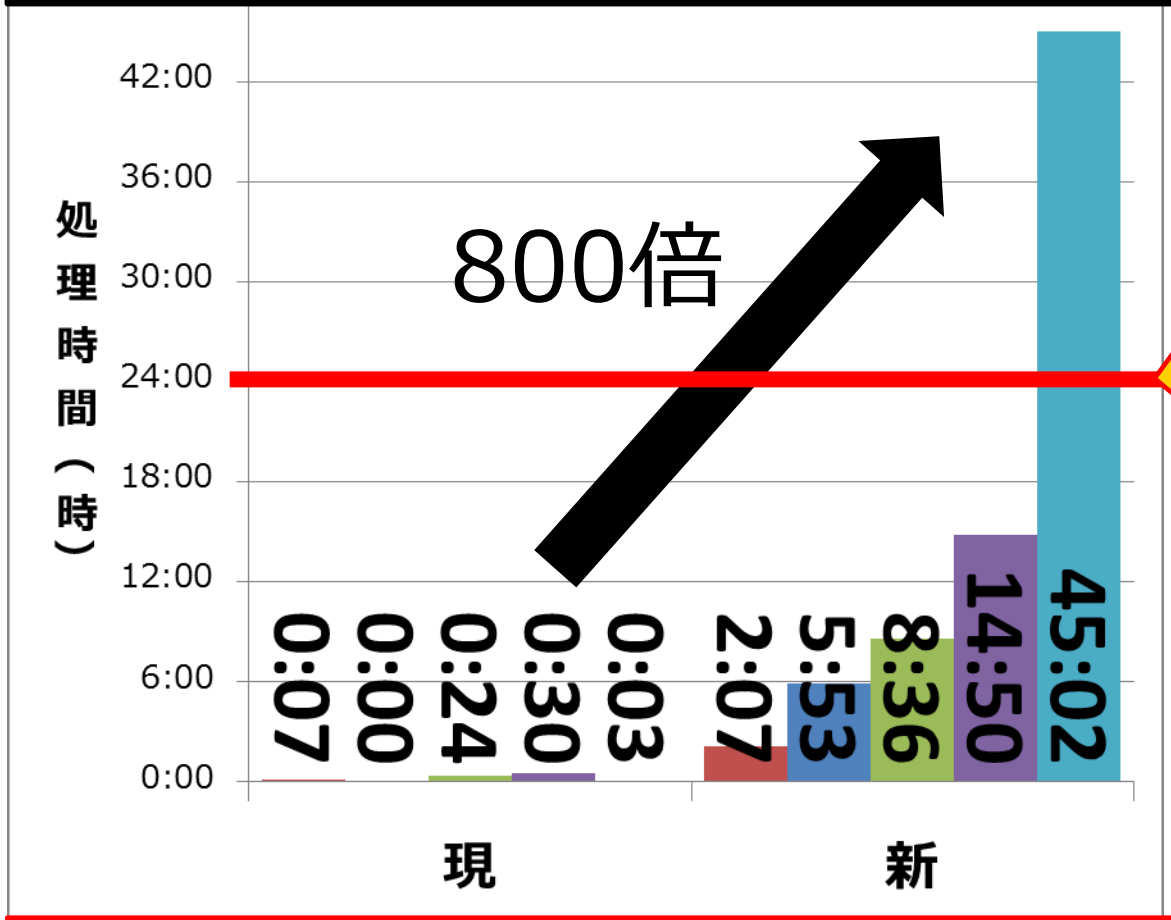


PCサーバかつ仮想サーバ（数台）

対応方式検討
現行システムの拡張を検討する

対応方式検討

課題①：処理データ８００倍

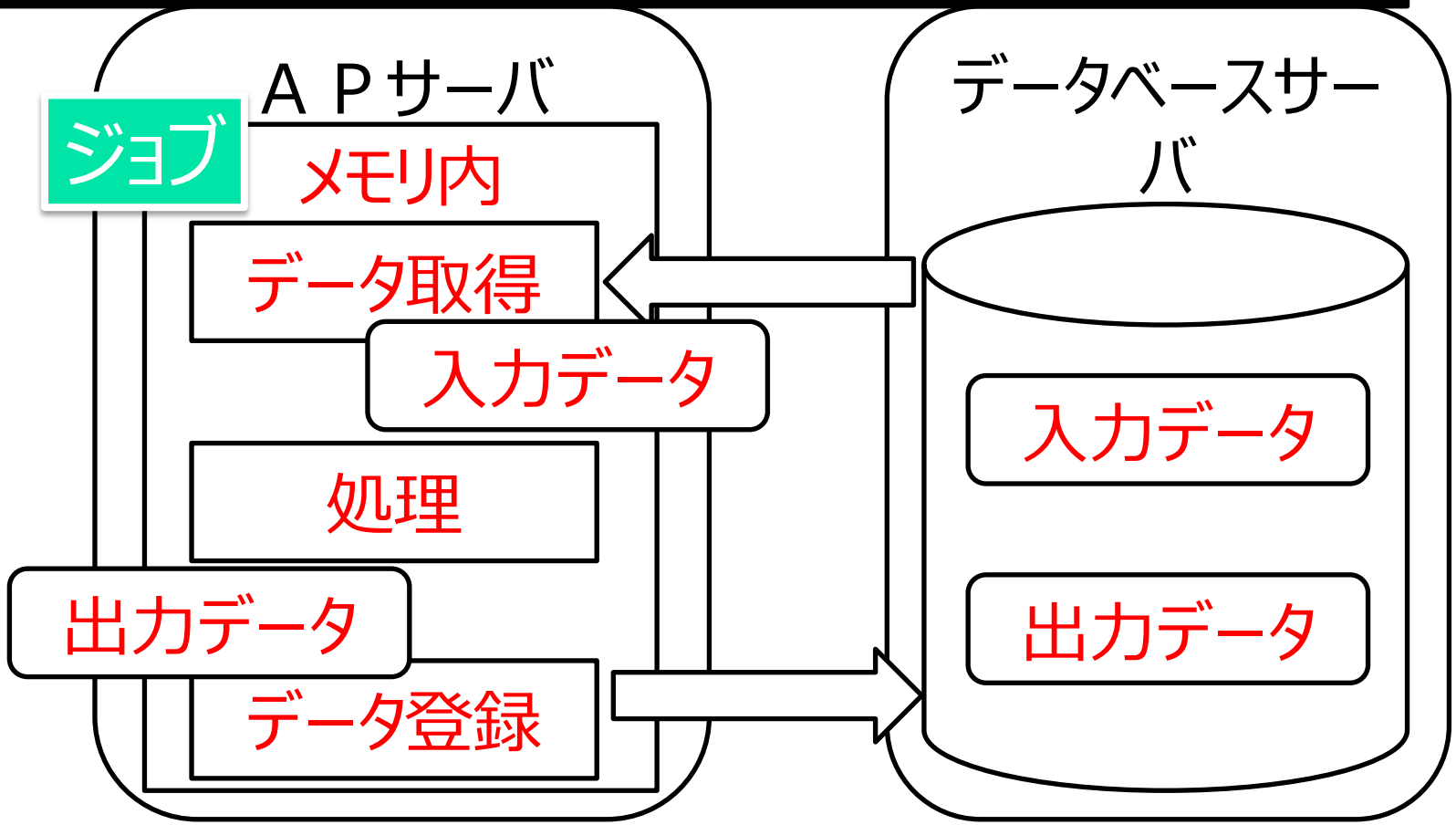


１日たっても終わらない

性能要求（6時間）を満たせない

対応方式検討

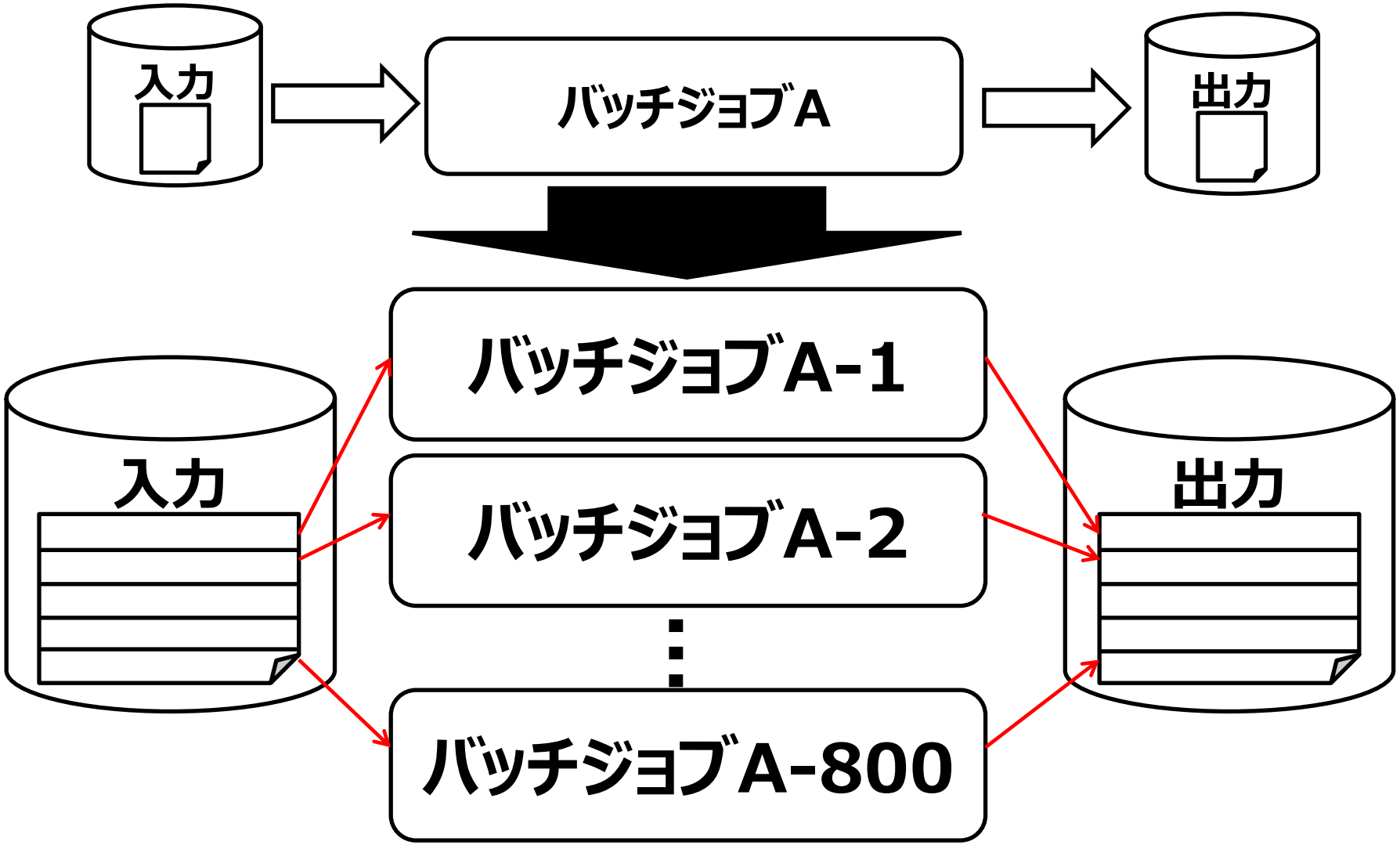
課題①：処理データ８００倍



物理メモリを超過すると処理不能

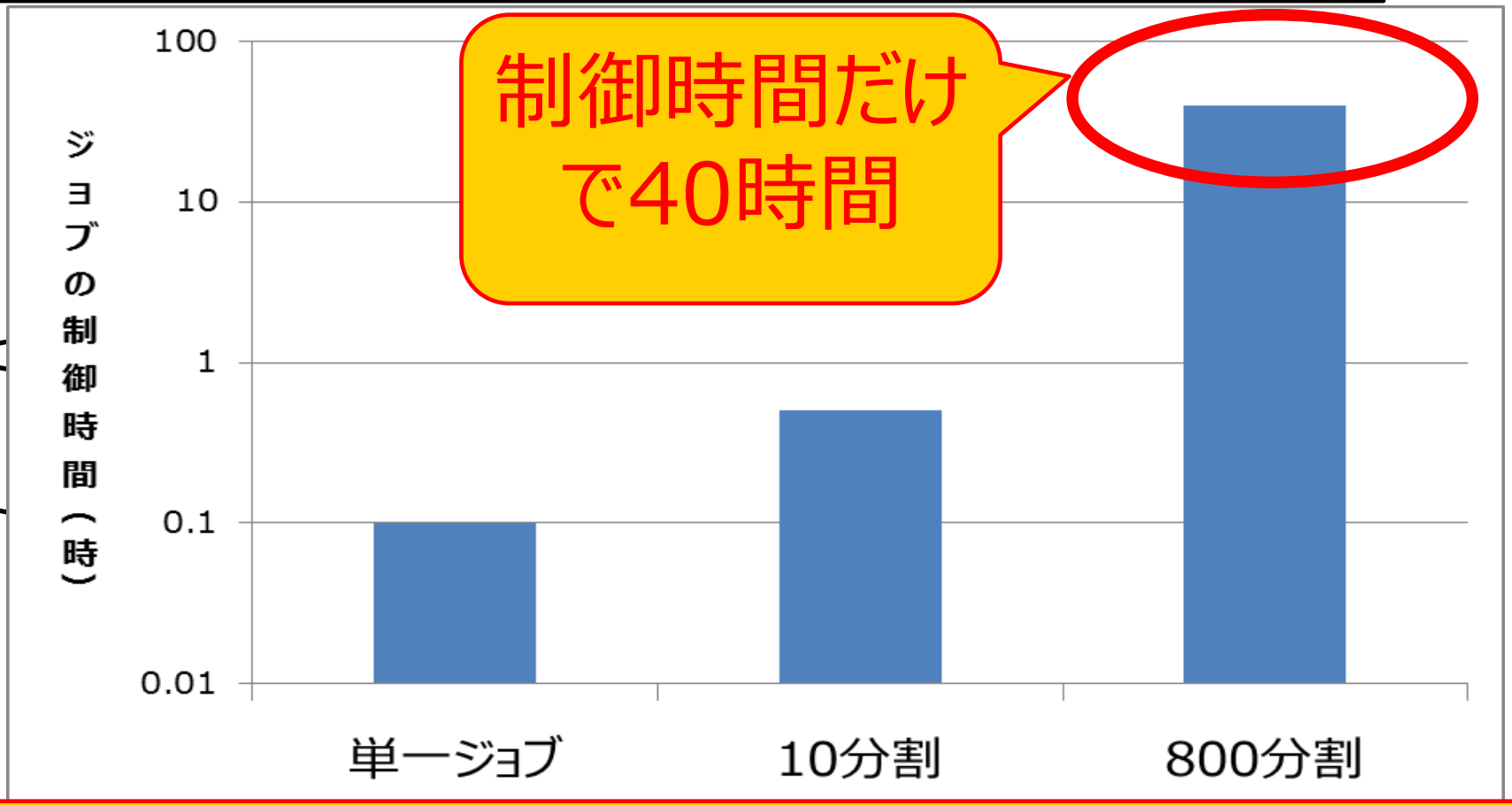
対応方式検討

ジョブ分割＋並列実行で対応する？



対応方式検討

課題①：処理データ８００倍



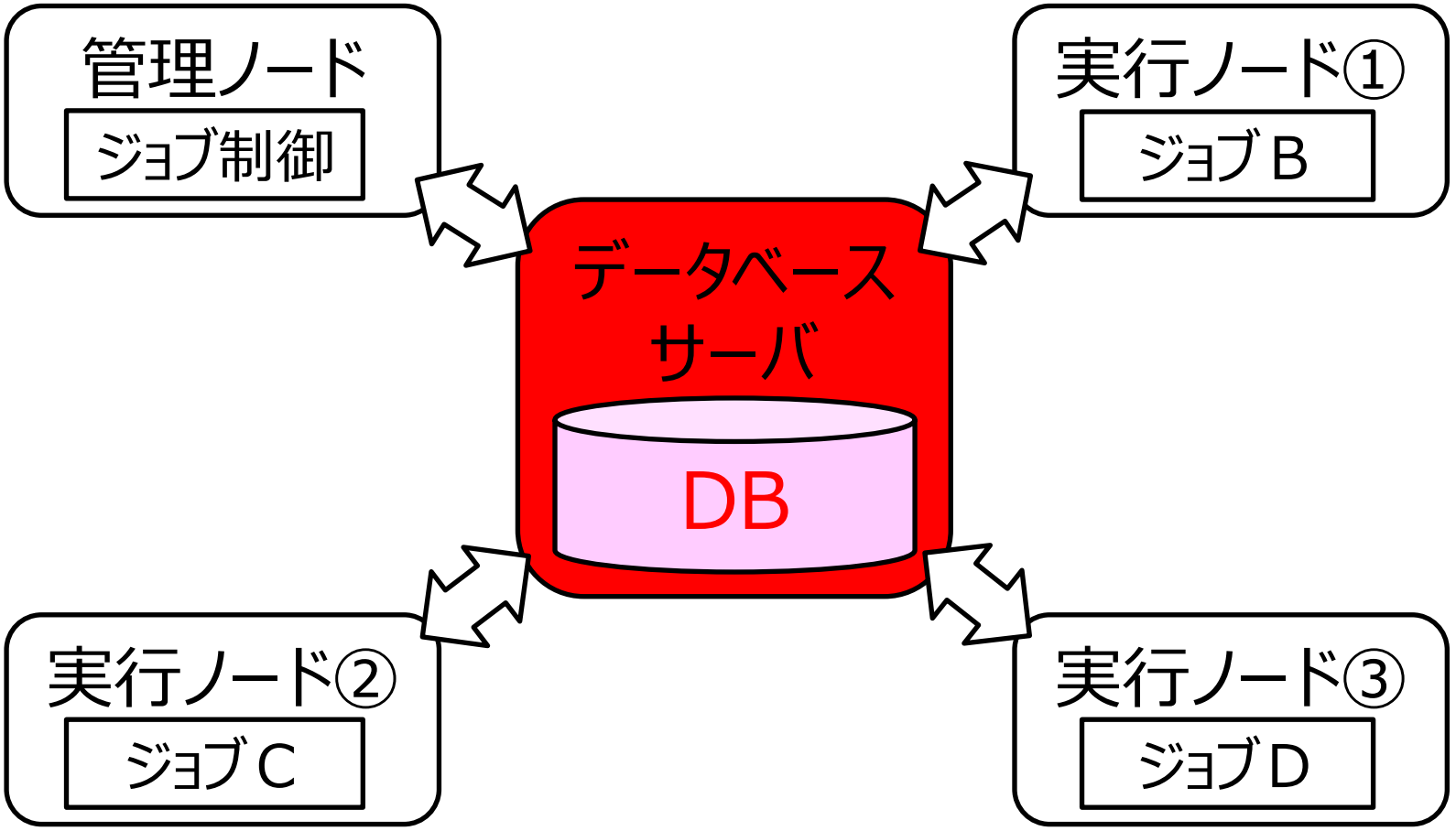
ジョブの増加に耐えられない

対応方式検討

現行システムの拡張では対応が困難
他にも課題が存在する、、、

対応方式検討

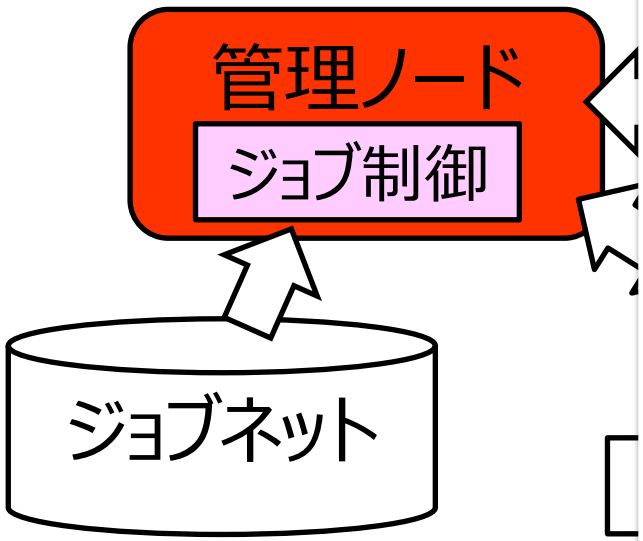
課題②：I/Oボトルネック



データベースがボトルネック

対応方式検討

課題③：ジョブ制御機構が止まるとすべて止まる



・ジョブ実行、終了管理
・ジョブネット管理
・実行ノード割り振り
・同一ジョブ複数基準日実行管理
・・・などなど

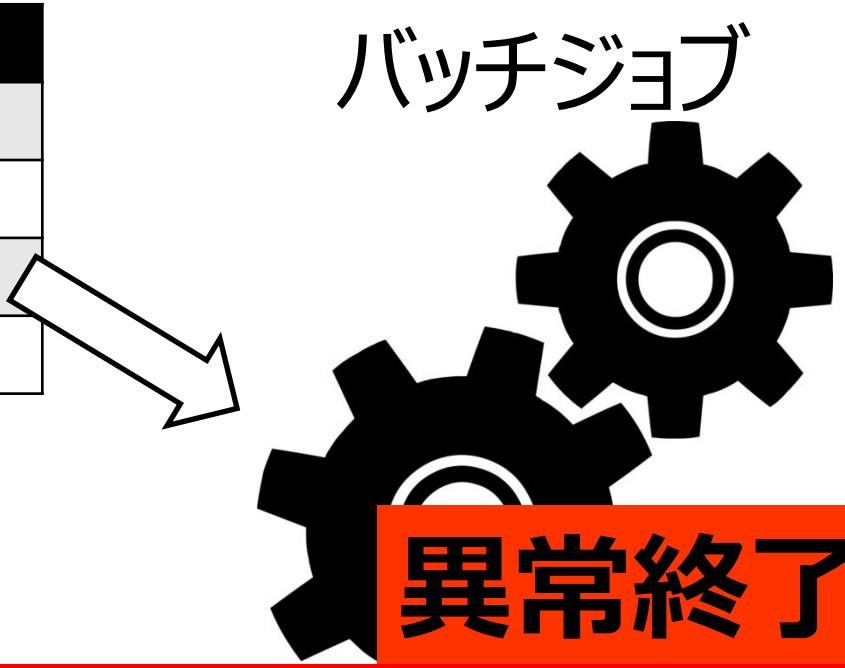
非常に複雑な制御を
ジョブ管理ソフトウェアでは実現できないため当システム独自で構築

ジョブ制御機構がSPOF

対応方式検討

課題④：不正データで処理停止

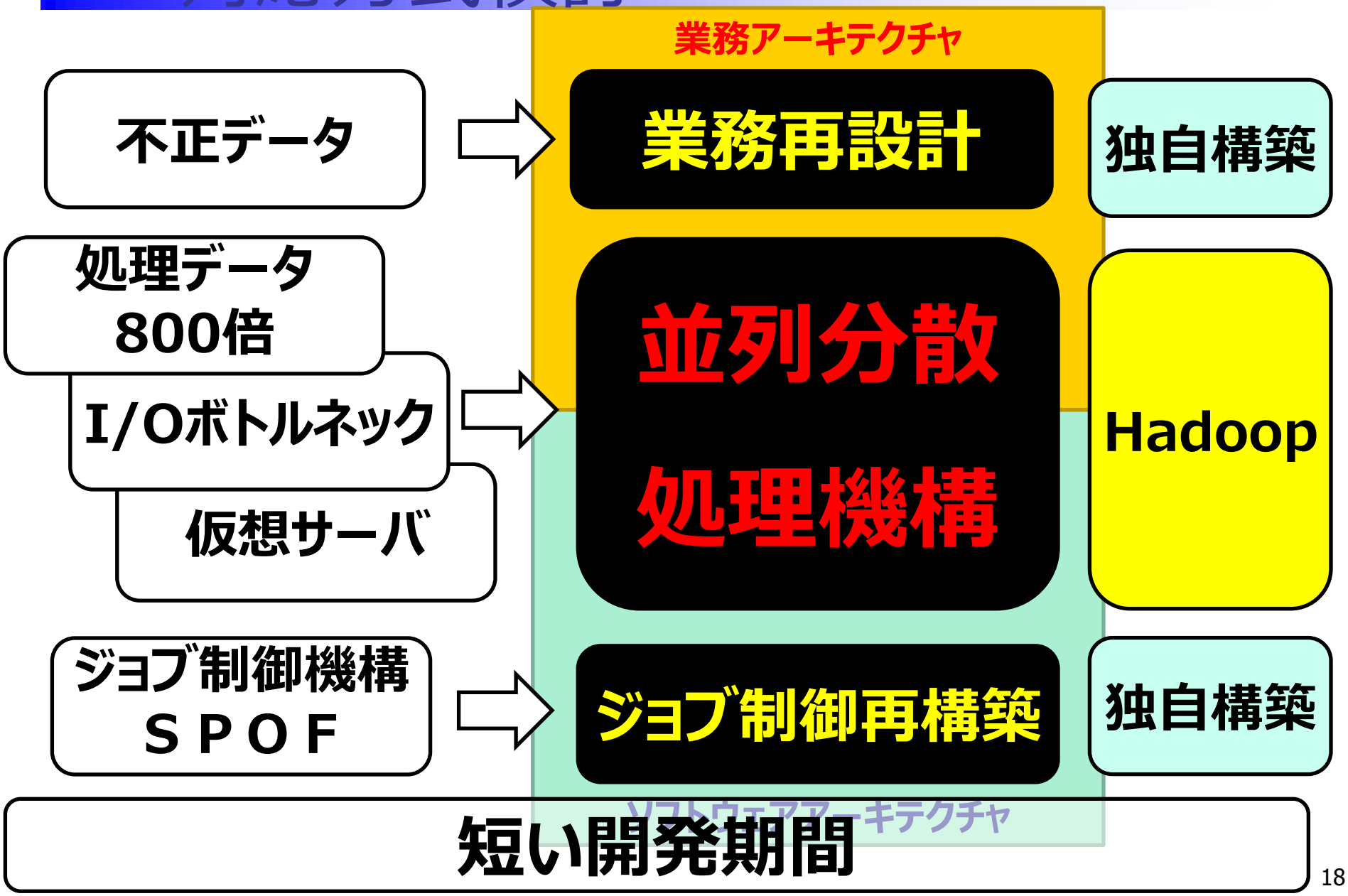
No	データ
001	Normal Record
002	Normal Record
9999	**Bad Record**
004	Normal Record



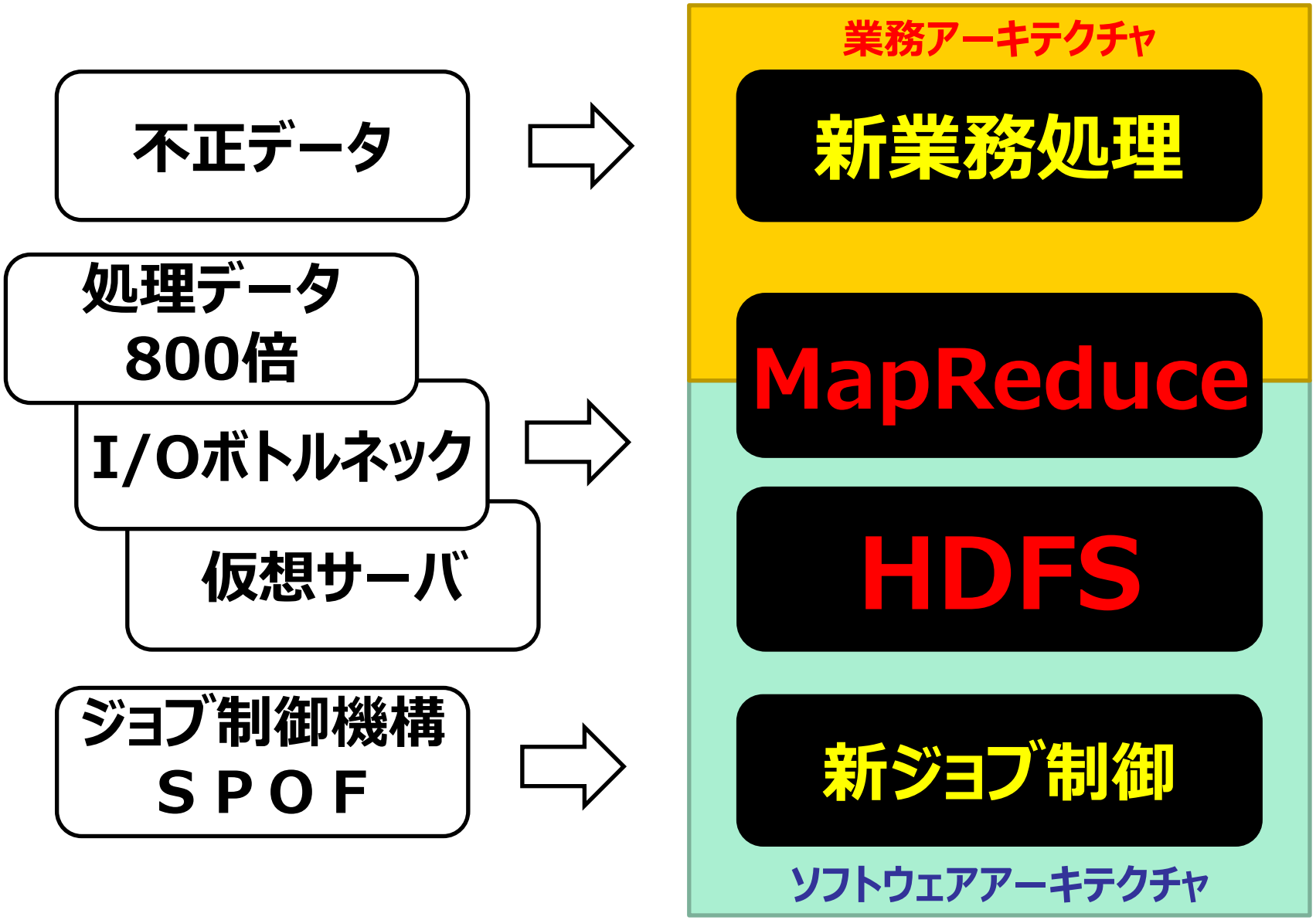
時間のかかる処理で
都度リランは困難

現行システム拡張では難しい、、、
さぁ、どうする？

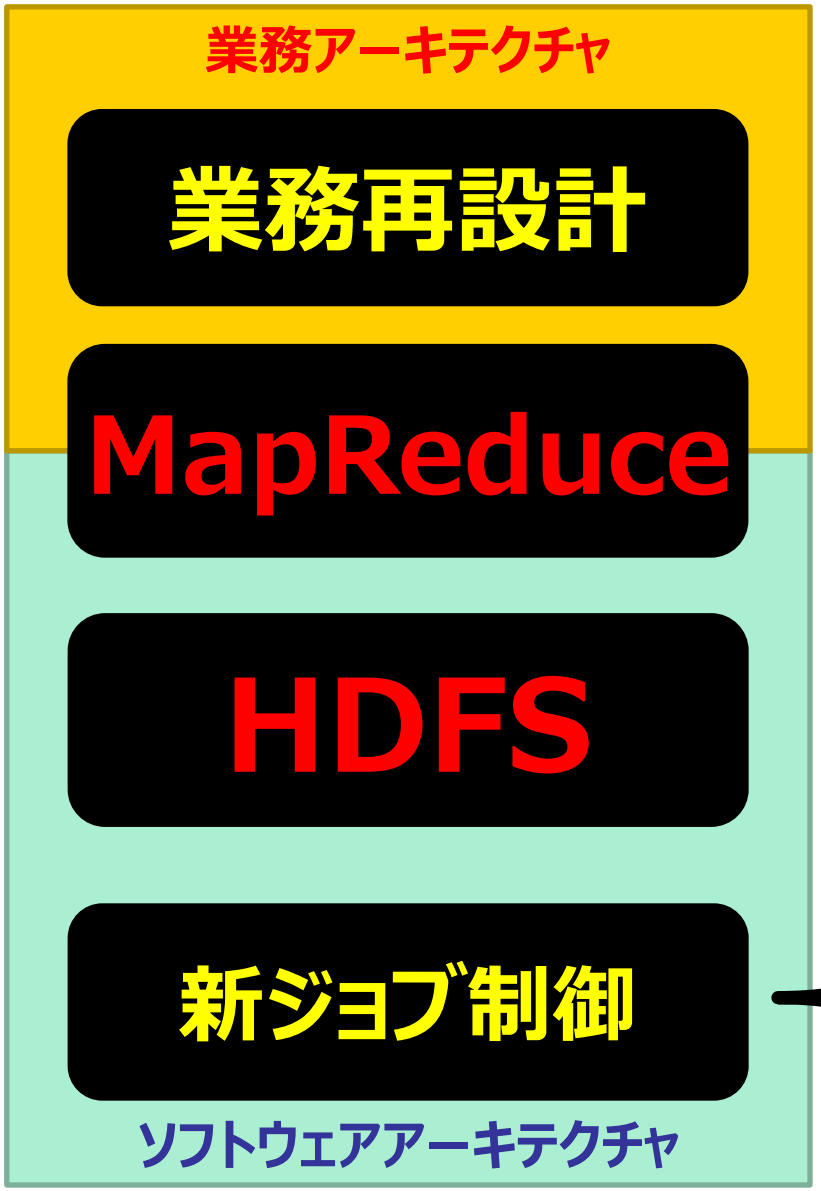
対応方式検討


業務アーキテクチャ
不正データ
業務再設計
独自構築
処理データ
800倍
I/Oボトルネック
仮想サーバ
並列分散
処理機構
Hadoop
ジョブ制御機構
ＳＰＯＦ
ジョブ制御再構築
独自構築
ソフトウェアアーキテクチャ
短い開発期間

対応方式検討


不正データ
処理データ
800倍
I/Oボトルネック
仮想サーバ
ジョブ制御機構
ＳＰＯＦ
業務アーキテクチャ
新業務処理
MapReduce
HDFS
新ジョブ制御
ソフトウェアアーキテクチャ

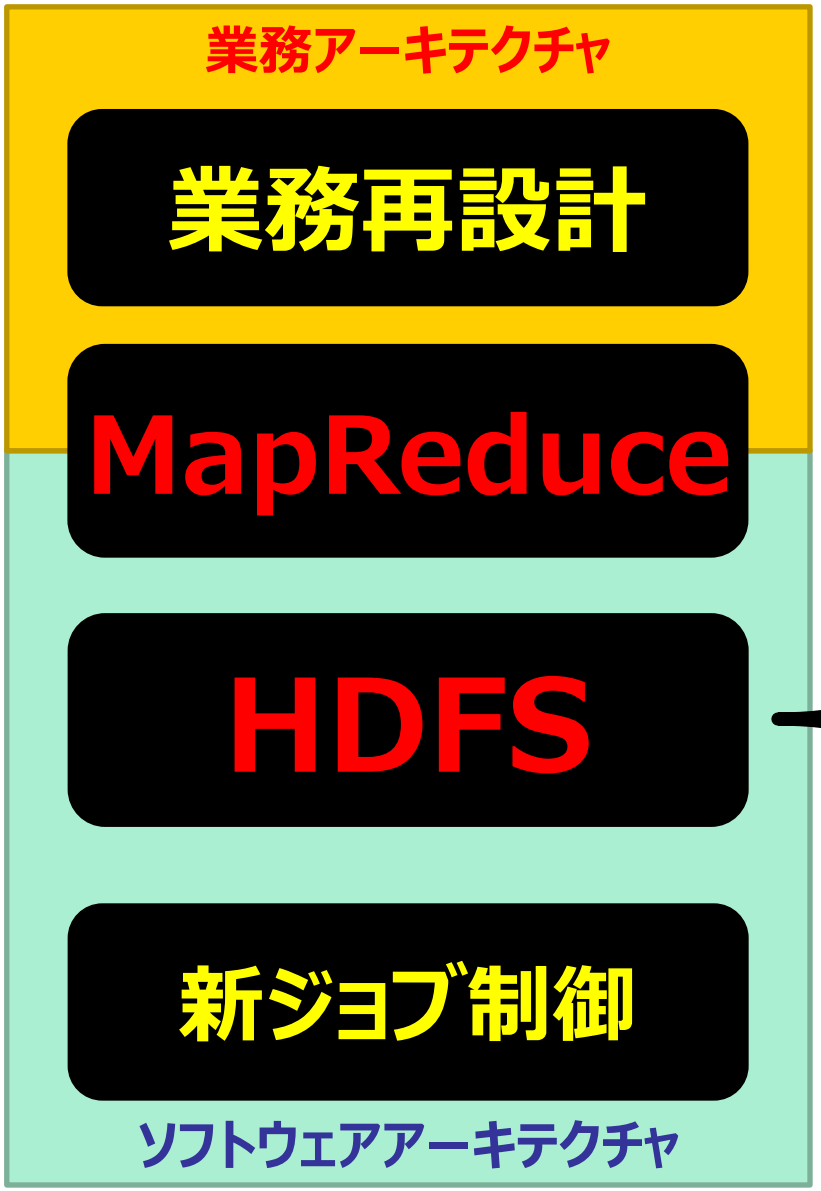
対応方式検討



【開発範囲の削減】
JP1/AJS各種機能の活用
- ジョブネット定義
- 実行ノードラウンドロビン
- ファイル監視ジョブ
- ・・・

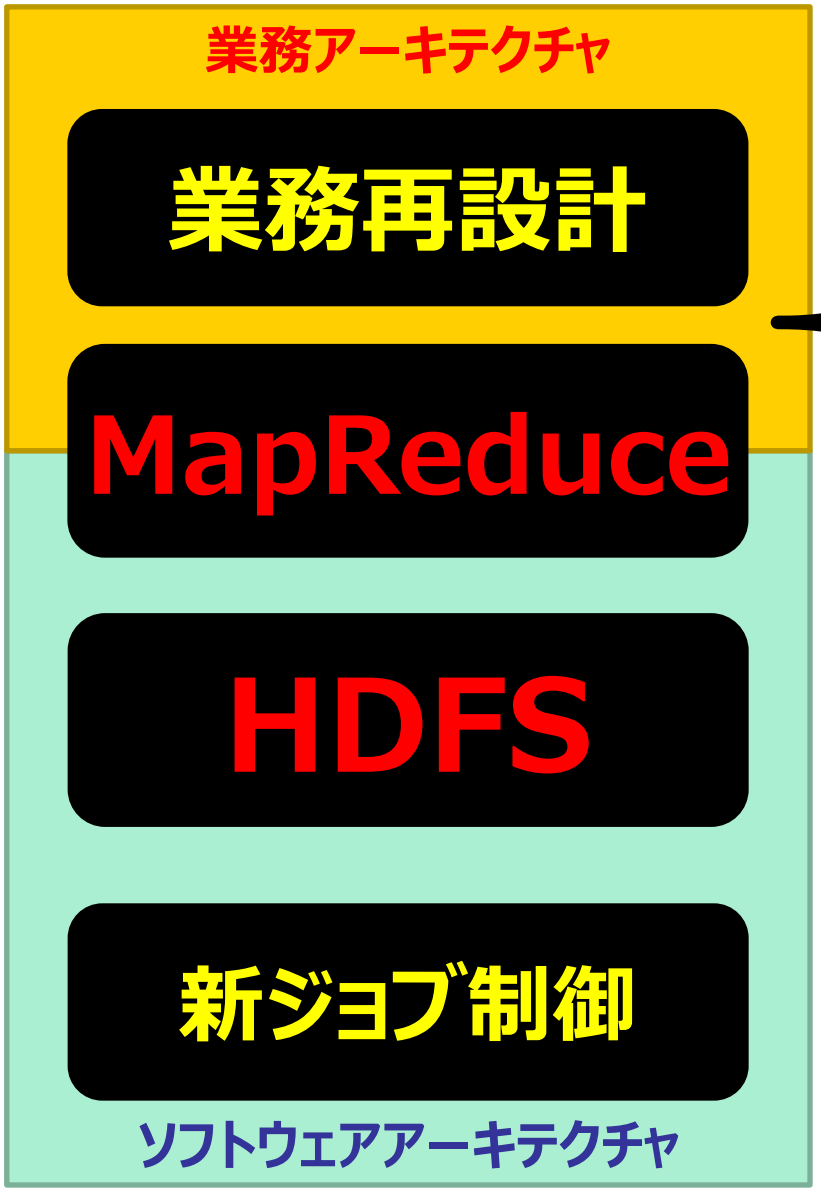
【開発生産性の向上】
SpringBootの利用
SpringData-jpaの利用
- ツール活用
  STS、VirtualBox、h2等

対応方式検討



【運用設計の削減】
利用用途をバッチ処理の一次領域。
バックアップ等の運用、ユーザからの参照機能の構築、運用を考慮しない。

対応方式検討



さぁ
どうする？

対応方式検討


業務バッチ処理標準定義
データ入力
データクレンジング
データ標準化
業務ロジック
データ出力
不正データ処理
MapReduce
業務ロジックをどうやって表現すればよいか？
標準化に時間がかかりそう。。。

対応方式検討

- 大量データを処理するため、Hadoopを前提とする。
- 短い開発期間において、効率よく業務ロジックを構築する。

- 上記を実現するために、フレークワークを探しました。

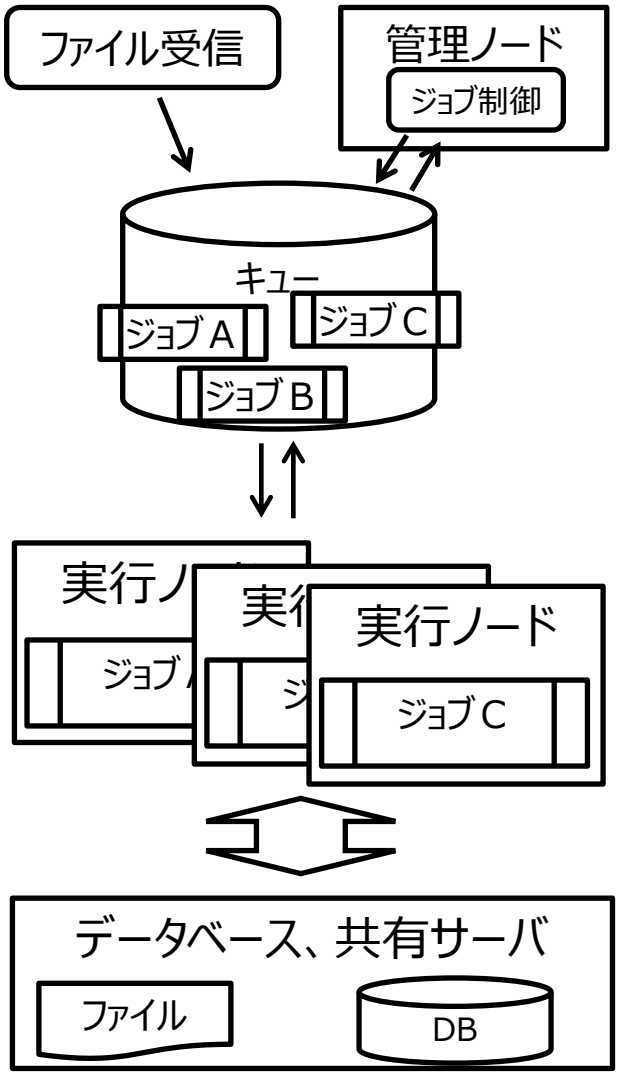
対応方式検討

複雑な処理もHadoopを意識せずに開発可能です。

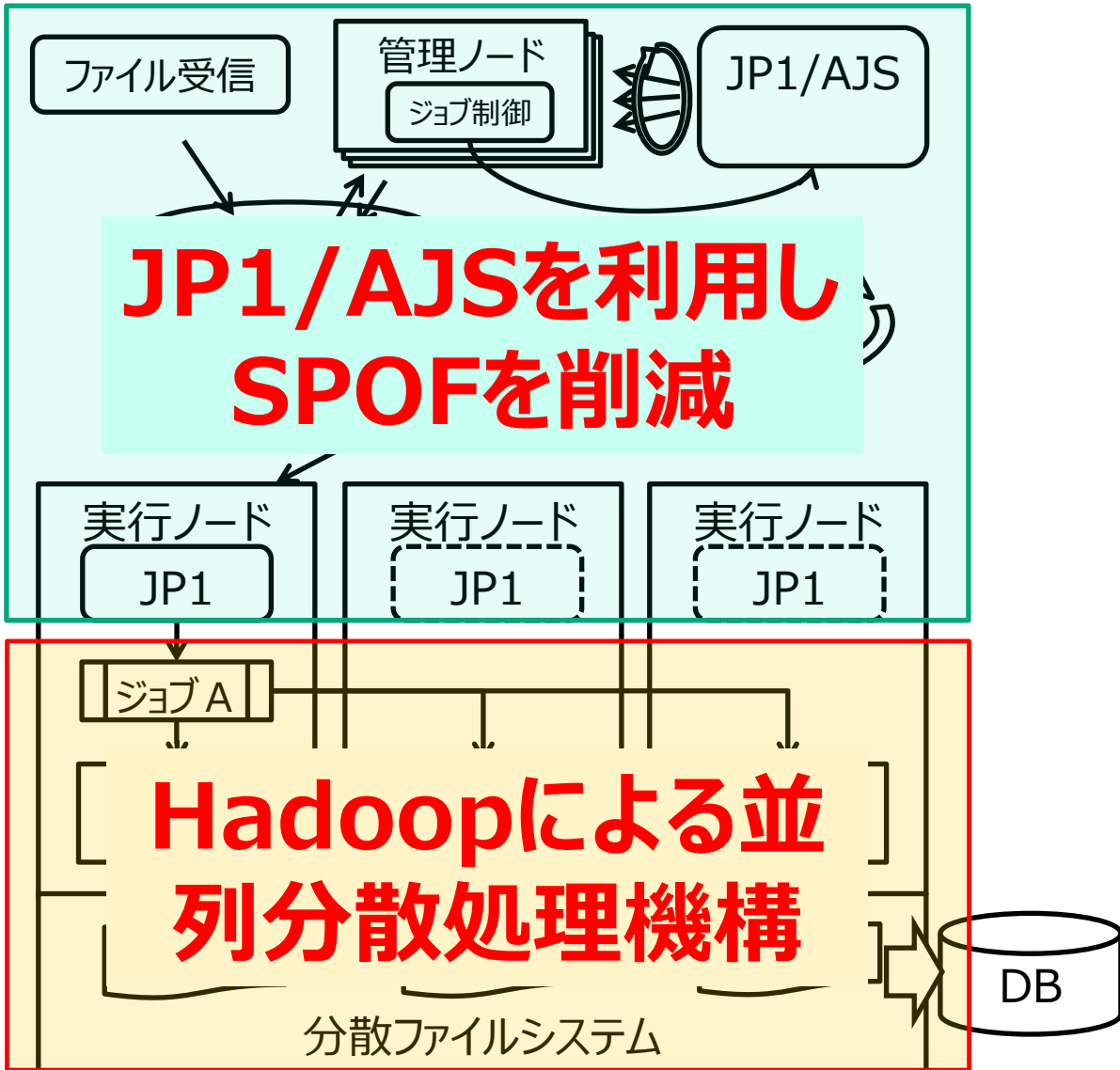
信じます！

対応方式検討

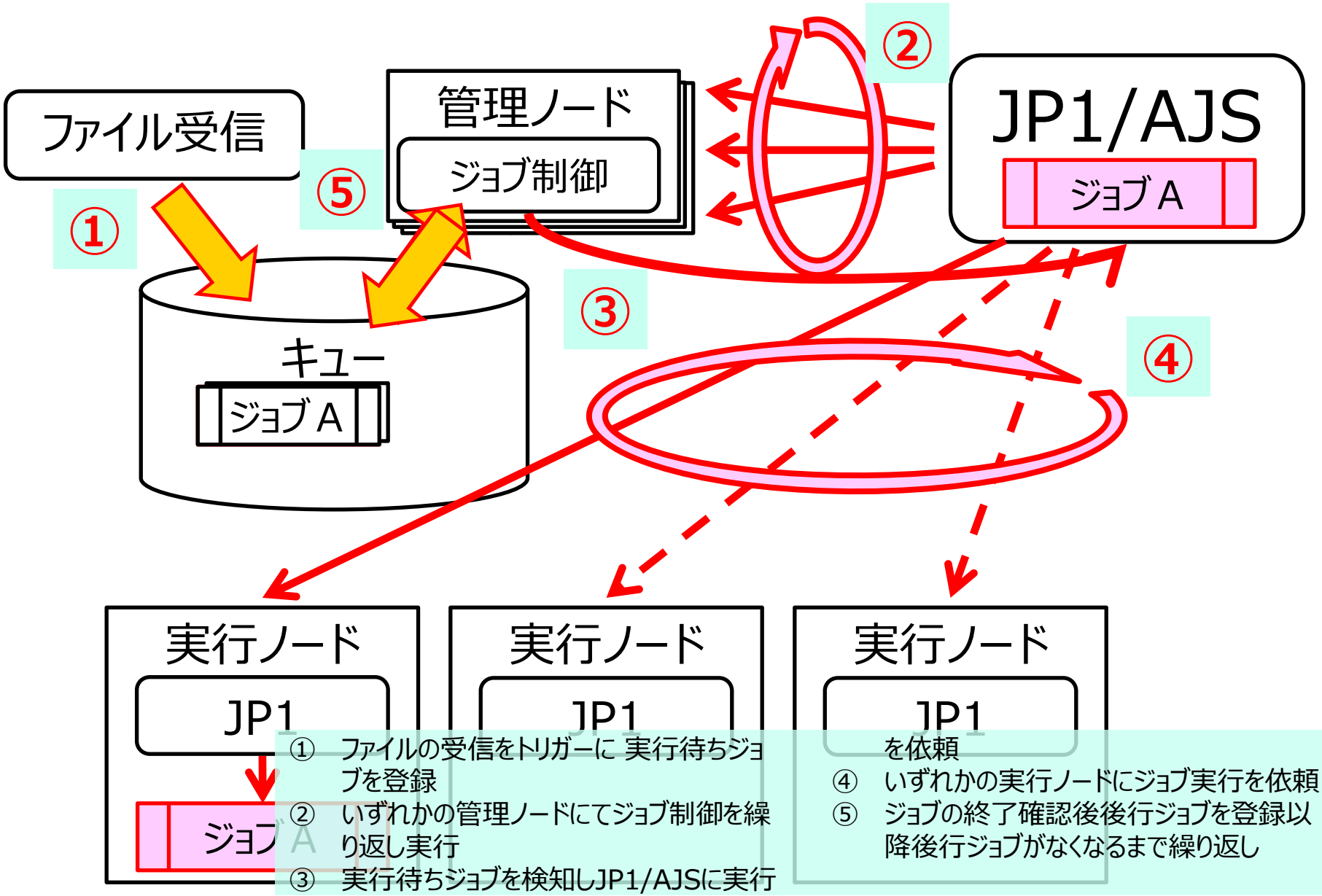
- 現行システム



- 新システム



ご参考）新ジョブ制御機構



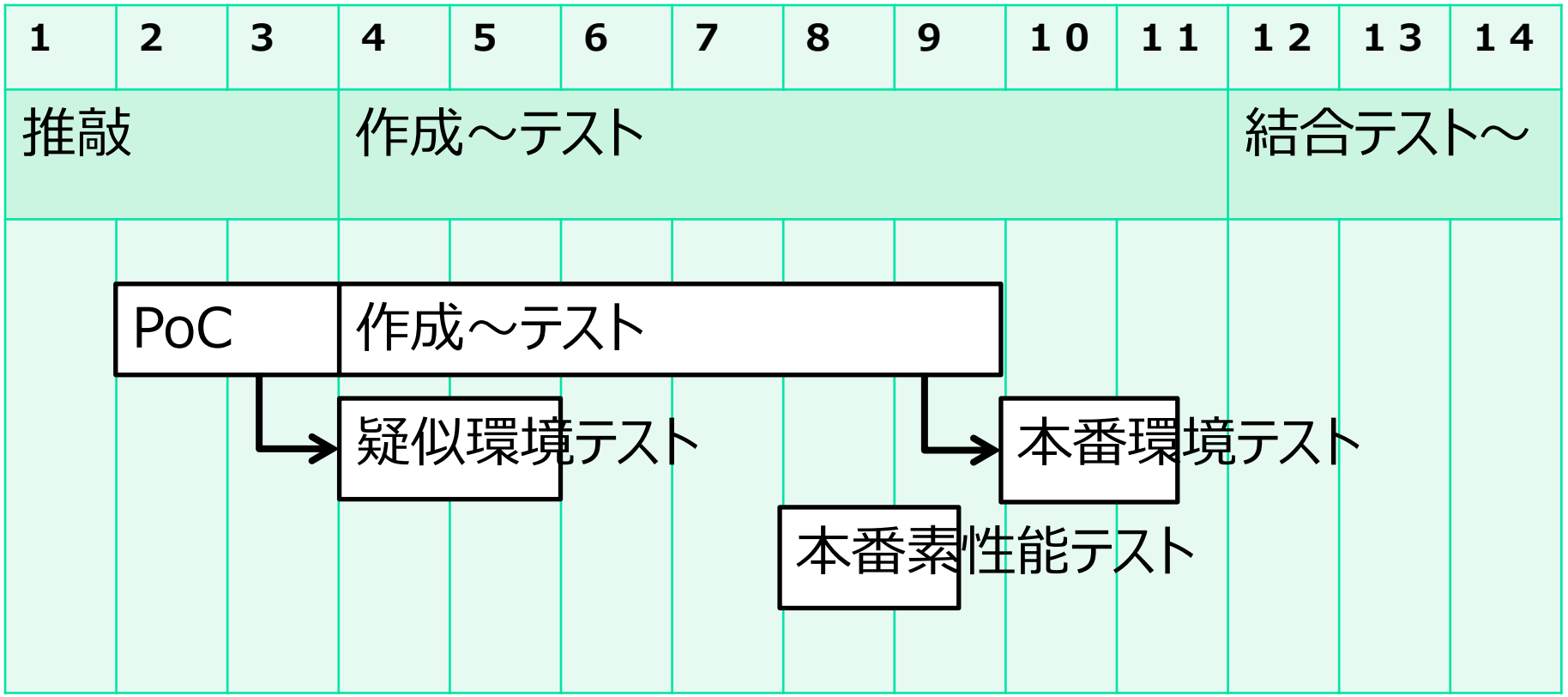
① ファイルの受信をトリガーに 実行待ちジョブを登録
② いずれかの管理ノードにてジョブ制御を繰り返し実行
③ 実行待ちジョブを検知しJP1/AJSに実行を依頼
④ いずれかの実行ノードにジョブ実行を依頼
⑤ ジョブの終了確認後後行ジョブを登録以降後行ジョブがなくなるまで繰り返し

本当にできるの？

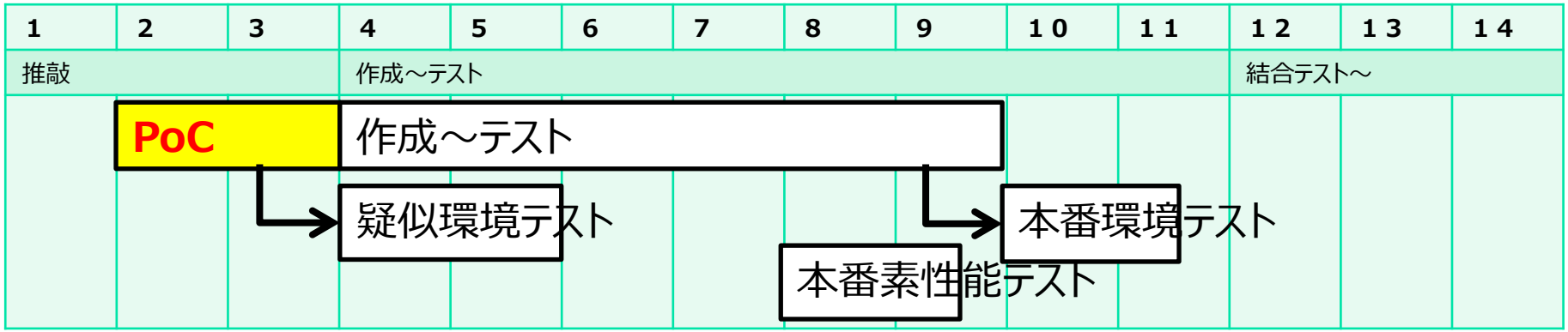
本当にできるの？

- アーキテクチャとして採用できることはわかったが、、

⇒正直やってみなきゃわからん！



本当にできるの？



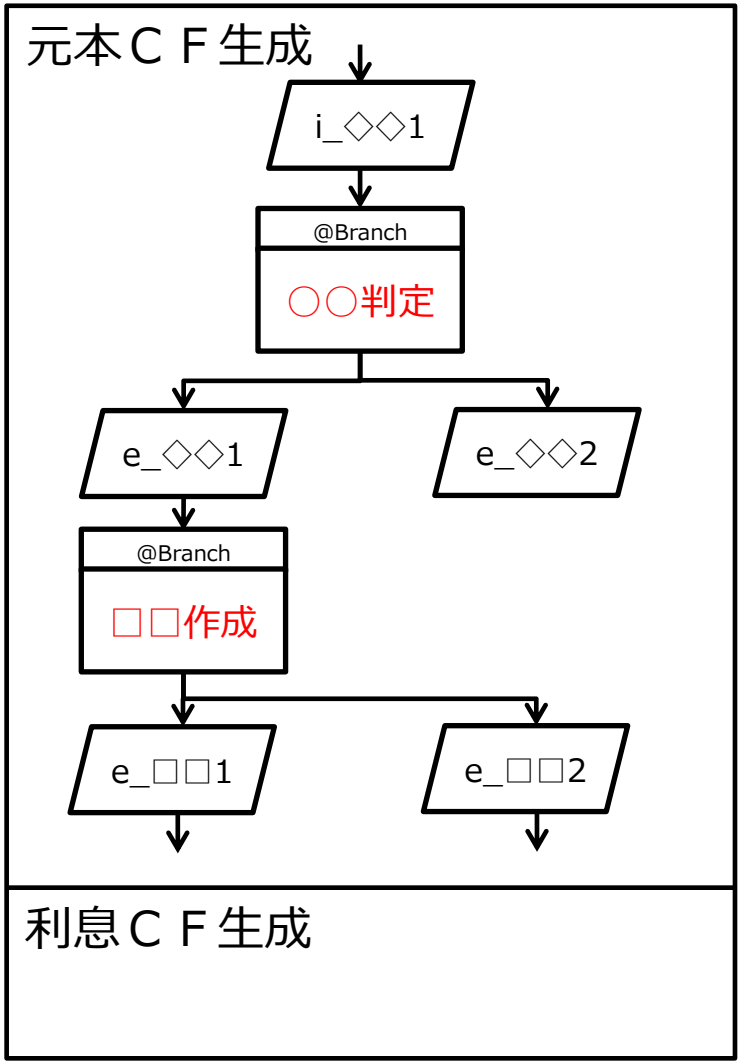
- 機能性
  - 入出力にCSV、Oracleを利用できる
  - 1バッチの入力データ1000万件（1.7GB）を処理できる事
  - 1日の入力データ総量（3GB弱）を処理でき
  - データ項目の移送が実現できる事
  - 複数種類のデータの突合せを実現できる事
  - １データレコードより複数レコードを生成でき
  - 誤差なく数値計算できる事
  - 仮想環境にて稼働可能であること
- 信頼性
  - ノード障害時に処理を継続できる事
- 効率性
  - 稼働時間内（20時間）に処理を完了できる事
- 保守性
  - ログ出力可能であること
  - スケールアウト可能なアーキテクチャであること
  - 5年後データ量（3%/年）に耐えられること

ＯＫ

- **Asakusa/Hadoopが使えることを確認**
  - **業務ロジックを表現できる**
  - **仮想OS上でも稼働する**
  - **想定されるデータ量も処理可能**
- **業務APの実装に時間がかかっている**
  - **設計書をプログラム仕様書レベルまで詳細化する**
  - **Asakusa教育プログラムを作る**
- **データベースへのデータ格納が遅い**
  - **データベースアクセス部分はAsakusaの外に出す**

ご参考）設計書フォーマット

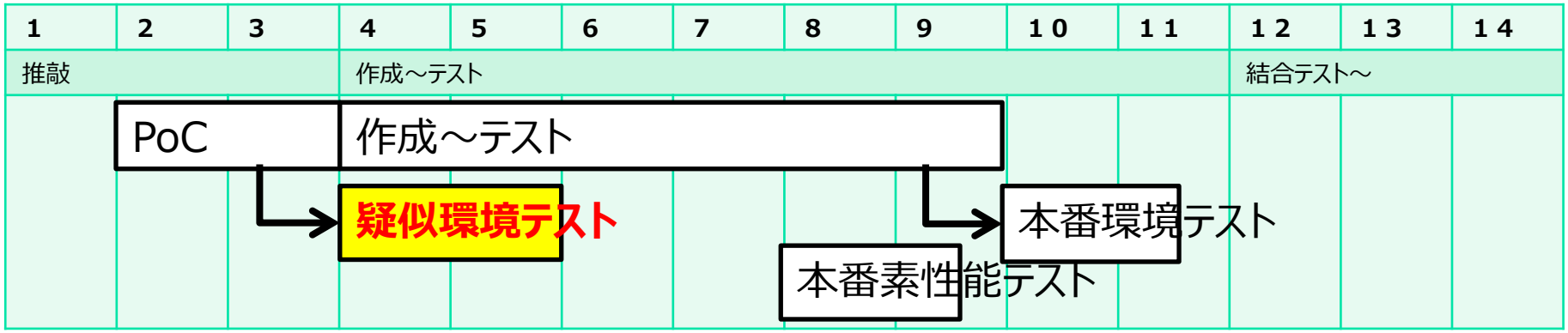
JobFlow設計書



演算子詳細記述

業務機能定義（FolowPart）	最小機能定義（演算子）	入出力定義（データモデル）
元本ＣＦ生成 ・・・・・・ ・・・・・・・・ ・・・・・・・ ・・・・ ・・・・・・・	○○判定(@Branch) ○○Branch ・・・・・・ ・・・・・・・・	入力◇◇モデル： i_◇◇1 出力□□モデル： e_◇◇1 e_◇◇2
	□□作成(@Extract) □□Extract ・・・・・・ ・・・・・・・・	入力□□モデル： i_□□1 出力□□モデル： e_□□1 e_□□2
利息ＣＦ生成 ・・・・・・ ・・・・・・・・ ・・・・・・・ ・・・・ ・・・・・・・	××判定(@Branch) ・・・・・・ ・・・・・・・・	入力◇◇モデル： i_◇◇3 出力□□モデル： e_◇◇3
	××作成(@Extract) ・・・・・・ ・・・・・・・・	入力◇◇モデル： i_◇◇4 出力□□モデル： e_◇◇4
	××集計(@Fold) ・・・・・・ ・・・・・・・・	入力◇◇モデル： i_◇◇5 出力□□モデル： e_◇◇5

本当にできるの？

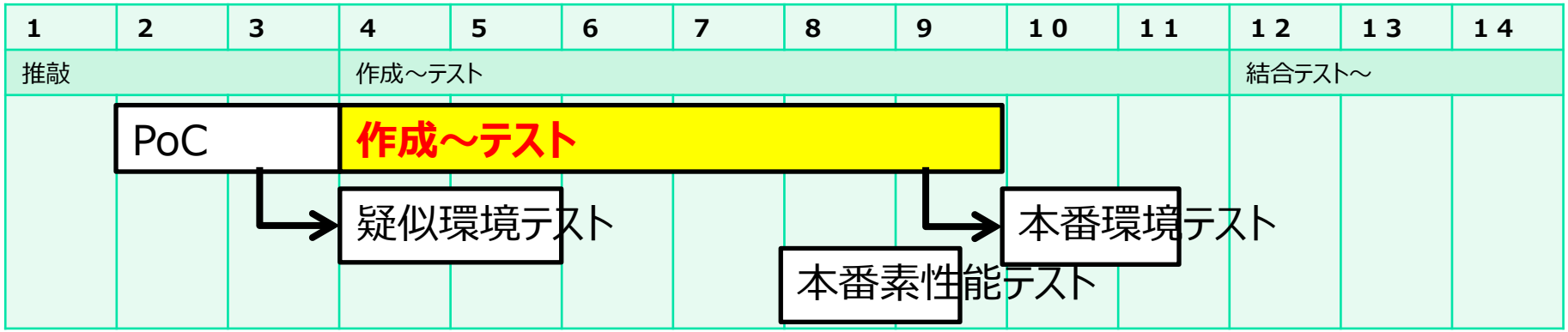


- **業務処理は期待どおりの性能**
  - **入力：54GB**
    （1000万件×30回同時実行）

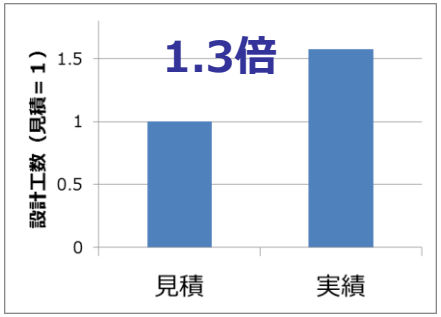
⇒3.5時間

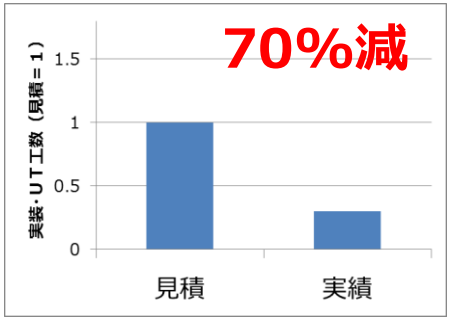
結果

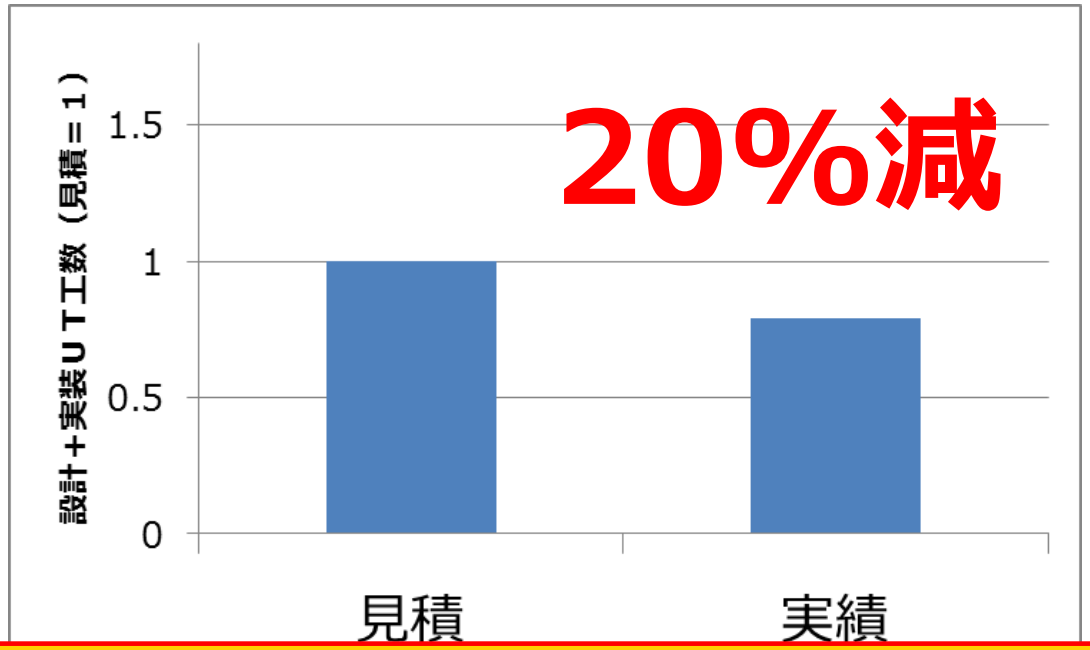
結果



- 生産性







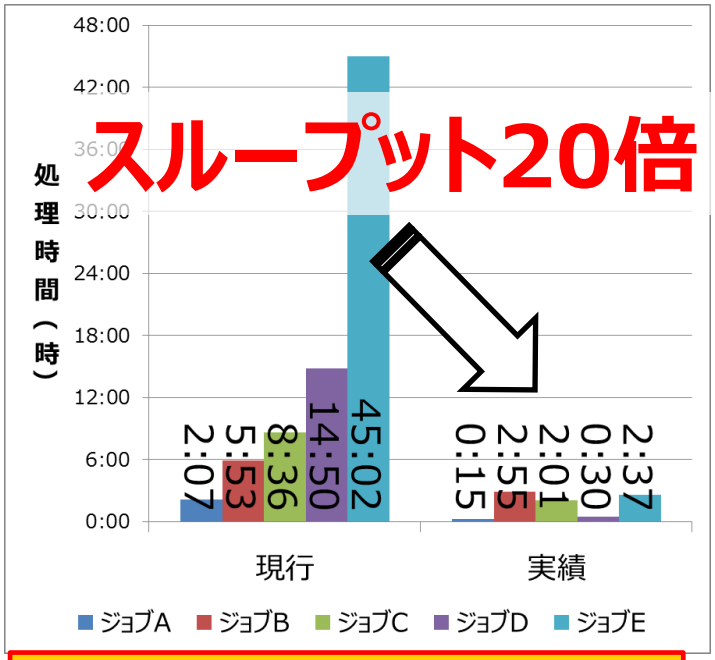
Asakusa自身の生産性＋設計詳細化により
設計～実装・ＵＴトータルで生産性向上

結果

1	2	3	4	5	6	7	8	9	10	11	12	13	14
推敲			作成～テスト								結合テスト～		

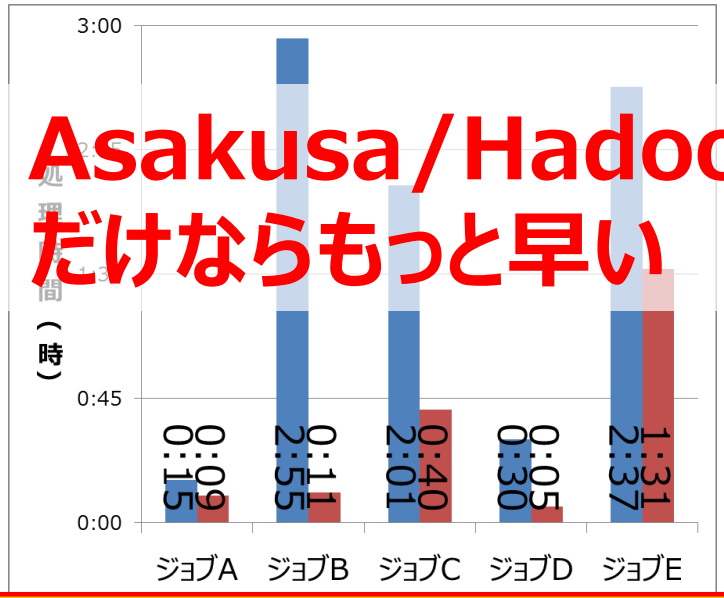
PoC
作成～テスト
疑似環境テスト
本番環境テスト
本番素性能テスト

- 性能



スループット20倍

処理性能は目標達成

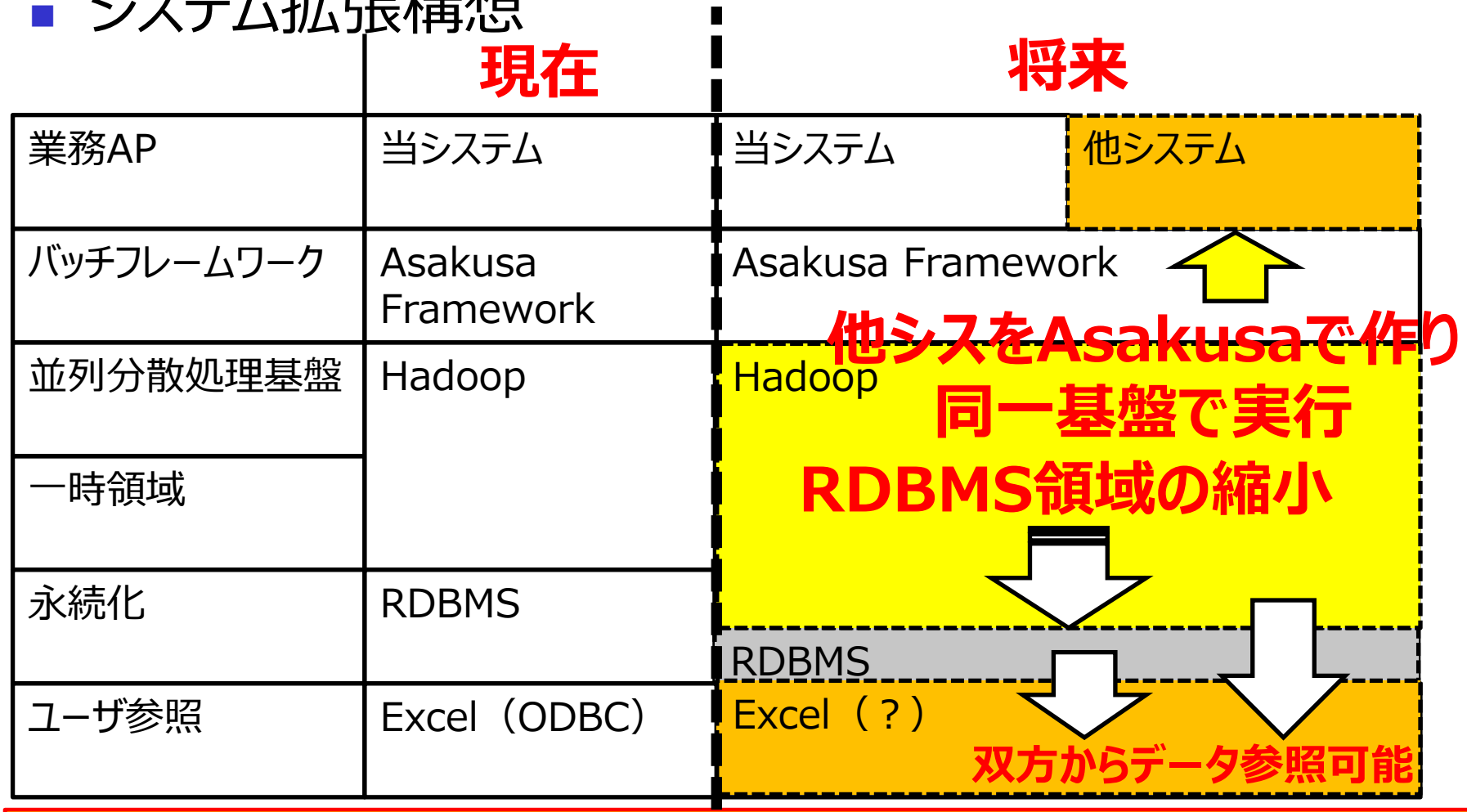


Asakusa/Hadoop
だけならもっと早い

Asakusa/Hadoop部分だけに着目すると
目標値をはるかに上回る

今後

- システム拡張構想



処理実行基盤や運用の統合

業務システムはAsakusaで作り処理実行基盤に依頼する

ご清聴ありがとうございました。