現代的RDB/Tsurugi＋超低遅延AI

NAUTILUS

TSURUGI

株式会社ノーチラス・テクノロジーズ
http://www.nautilus-technologies.com/
Tel: 03-6712-0636 Fax: 03-6712-0664

ノーチラステクノロジーズ：会社概要

- 会社名 ：株式会社ノーチラス・テクノロジーズ
- 住所 ：〒107-0051 東京都港区元赤坂1-5-12 住友不動産元赤坂ビル7F
- 代表 ：代表取締役会長 神林 飛志
  代表取締役社長 目黒 雄一
- 設立 ：2011年10月3日
- 資本金：45百万円
- 関連会社
  - オーシャンブリッジ
- 加盟団体
  - OSSコンソーシアム、日本情報システムユーザ協会
- パートナー
  - Databricks、AWS ISVパートナー
- 事業内容
  - 次世代型RDBMS “Tsurugi” の開発及びデータ活用ソリューションの提供
  - 分散処理技術を中心としたシステムインテグレーション,運用,保守サービスの提供
  - 分散処理基盤用アプリケーションの効率的な開発を可能にする製品「Asakusa Framework」の開発・サポート
  - 消費財流通のEDI 標準である流通 BMS 準拠のEDIソフトウェアパッケージ製品「UJX」の開発・販売・サポート。等

劔 “Tsurugi”
- インメモリ・メニーコアを利用する次世代型のデータベース
- 開発メンバーすべて日本企業及び団体（大学等）で国産データベース
- バッチ処理とオンライン処理を併存させながら一貫性を担保する

Asakusa on M³BP
- 単ノードのメニーコアで並列処理を行うエンジン
- 他の分散処理プラットフォームより高速処理で運用負荷も軽い
- Asakusa Frameworkでアプリケーション開発を実施可能
- 開発・運用などのサービスを提供

Asakusa Framework
- 分散処理のアプリケーション開発と運用をサポートするフレームワーク
- 分散処理エンジンのM³BPとHadoop/Sparkにも一部対応している
- 開発や運用などのサービスを提供

すべてをOSSとして公開
- 多くの方に、まずは使ってみていただくことで効果を試してほしい
- 今後の成長し更新し続けるシステムに対する取り組み方を検討する一助にしてほしい

Tsurugi（劔）とは

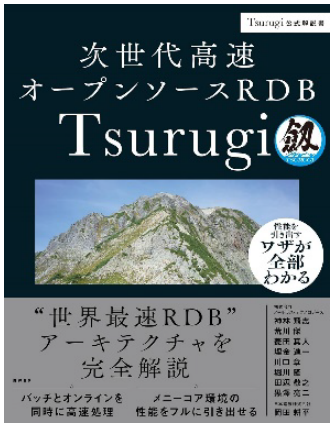
■OSSで、メニーコア・大容量メモリーの現在のハードウェア環境に合致する、モダンアーキテクチャなRDB

■NEDOの支援を受けて、各大学研究機関・NEC・ノーチラス各社が協力してR&D/製品化/リリース・有償サポートを行っている

■RDB本体だけはなく、利用手法が理解できるプロトアプリケーションも作成され公開されている

■一貫性を担保したSerializableなRDB（OLTP）

劔 TSURUGI



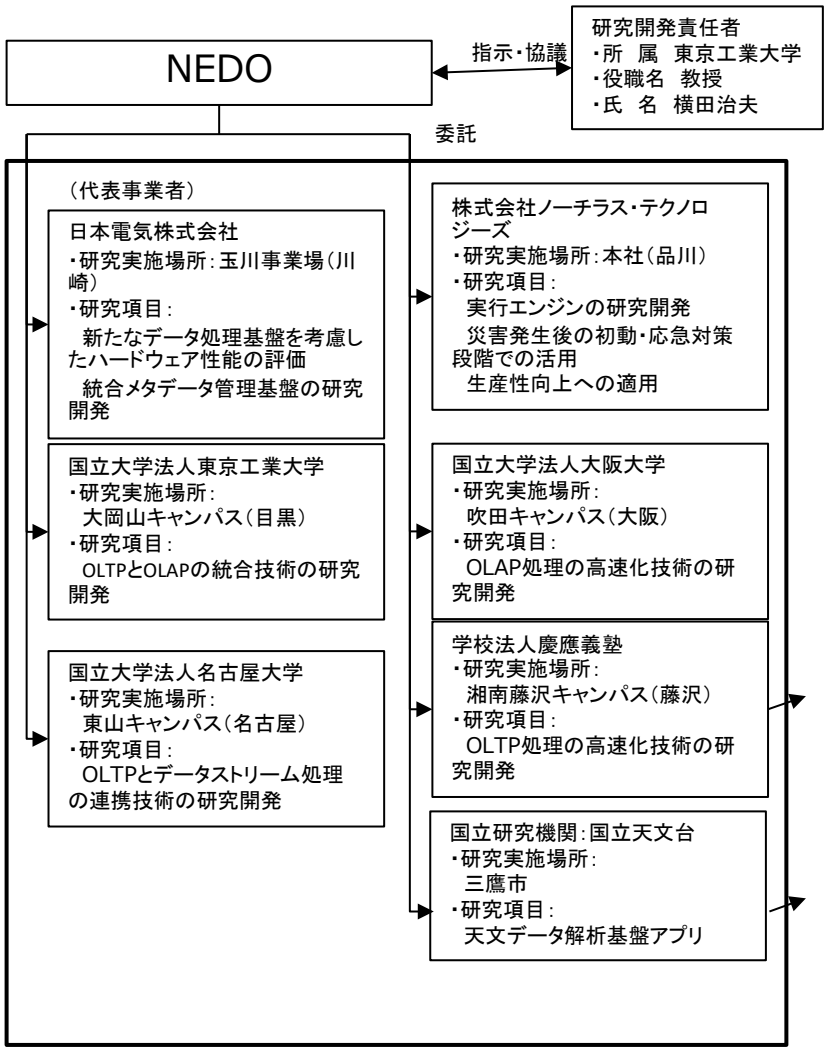
成立までの経緯

- 有志での勉強会
  - Tx関連・論文輪読等々を行うエンジニアのコミュニティ
  - いつまでも出てこないので手弁当でなにか作る？という流れで・・・

- NEDOとの連携
  - 高効率・高速処理を可能とするAIチップ・次世代コンピューティングの技術開発に応札
    - 事業期間：2018年度～2022年度
    - 現在では次のフェーズがスタートしている
  - 参加組と非参加組がそれぞれできる形で協力
    - 非参加でも一部を除きプロジェクト外サポート

- ノーチラスのみならず様々な企業・団体が参加して作られたRDB
  - 単独企業が作ったというミドルウェアではなく、DBに携わる日本人が様々な形で関与してできたRDB

- OSSで展開
  - 誰でも自由に利用することができる
  - 展開用のアプリケーションも開発

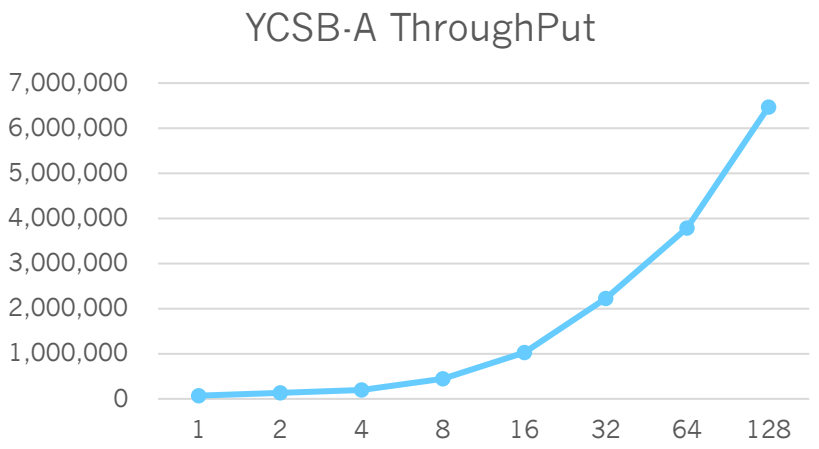


基本パフォーマンス（v1.5）

・YCSB-A/C

YCSB-A			YCSB-C	
	latency(ns)	ThroughPut(TPS)	latency(ns)	ThroughPut(TPS)
1	13,990	71,480	12,661	78,980
2	7,589	131,763	12,771	78,300
4	5,177	193,144	4,133	241,975
8	2,258	442,805	1,745	572,977
16	971	1,029,927	790	1,266,392
32	450	2,223,026	418	2,394,028
64	264	3,782,115	239	4,180,759
128	155	6,470,201	139	7,175,573

基礎性能：YCSB-A（w/r 50/50）
120コアレベルで
スループットが、秒あたり640万tx
遅延は155ナノ秒



CB-YCSB-Threads on:spr112 scenario:all threads:[1,2,4,8,16,32,64,128] duration:30 epoch_time:3000 build_pwal:OFF 2024/09/27 #522

強一貫性(serializable)・インメモリー（永続化済み）

1.基本アーキテクチャ

1.基本アーキテクチャが次世代型
メニーコア・インメモリー処理

- 強一貫性（のみ）を提供
  - serializableを常に提供
    - ユーザサイドで一貫性担保の仕組みを準備する必要がない
    - 他のRDBと異なりserializableで高いパフォーマンスを維持

- 処理データをすべてメモリー上にもつ
  - データはすべてメモリー上にもって、処理する
  - 永続化はすべてSSD・HDDに行っている
    - コミットはあくまで永続化した後になる

- ロックフリー
  - 一貫性を担保しているにも関わらず、read/writeをロックしていないため、高いパフォーマンスを提供している

- アーキテクチャが分散システム志向
  - コアスケールさせるために、実行計画から分散処理している

ロック・フリー

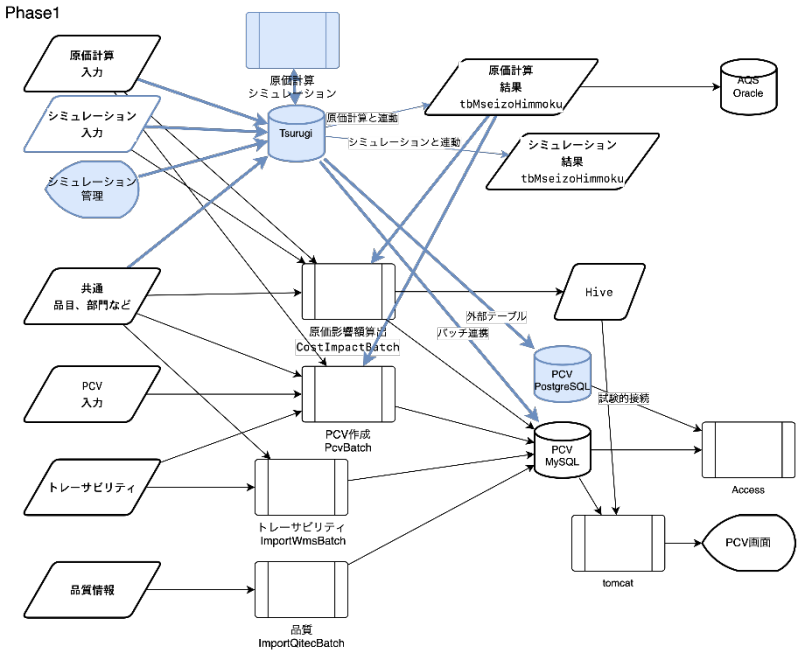
- リード・ライトともにロックを取らない
  - ロックは取らずに強一貫性(serializable)を確保
  - 物理順序と論理順序を別々に管理・計算する
    - 不整合が発生する場合はabort/retry（例：read modify writeの競合）
  - write側はreadを気にせずに一方的にデータを書き込める
  - read側はwriteがどう書き込もうが、一貫性を持ったまま読み込める

- ロックフリーのメリット
  - 速度が極めて速い～低遅延処理に秀でる
  - 分散処理に強い
    - ロックをとらないため分散処理で必須の同期処理が極めて有利になる
  - 低遅延処理をメニーコアで分散処理するため、スループットも劇的に上がる

本番適用中

既存システムリプレース＋新規

- 原価計算
  - 原価計算、シミュレーションはTsurugi DBを利用し指定した基準単価により入出力データを切り替える。
  - 入力データは現行連携済みデータからTsurugiにインポートする。
  - 計算結果データはTsurugiDBに生成され、現行に合わせた形でエクスポートする。
- シミュレーション管理
  - シミュレーションの元となる単価は現行の方法にて登録する。
  - シミュレーション実行は画面にて必要な情報を指定して実行を行う。
  - 実行状況は画面にて確認可能とする。
  - シミュレーション結果を保管可能。
  - シミュレーション結果の取得は現行と類似した方式とする。
- データ連携
  - 外部からの連携データは現行の連携後データを利用する。
  - 原価計算結果を利用する現行処理に対しては同等のデータを連携する。
  - 稼働後も現行の原価計算処理への切り戻しを可能とする。



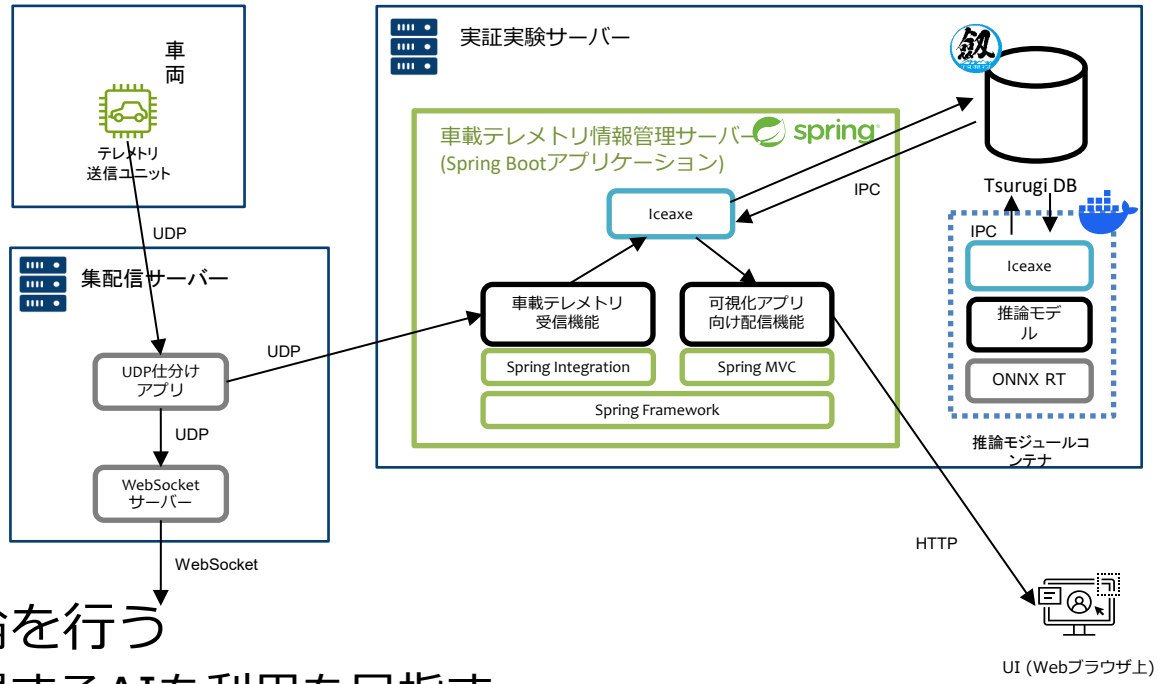
AIの実行基盤として

2.低遅延処理の基盤として

低遅延処理でのTsurugiの優位性

- ロックフリーかつ低遅延
  - 一貫性を担保しつつ、read/writeロックを取らない
  - 処理データを書きつつ、同時に読み込み・推論を実行できる
- フォーミュラーカーレースのリアルタイム解析を分散コンピューティング・AIで行う（2024/11：鈴鹿決勝で運用）

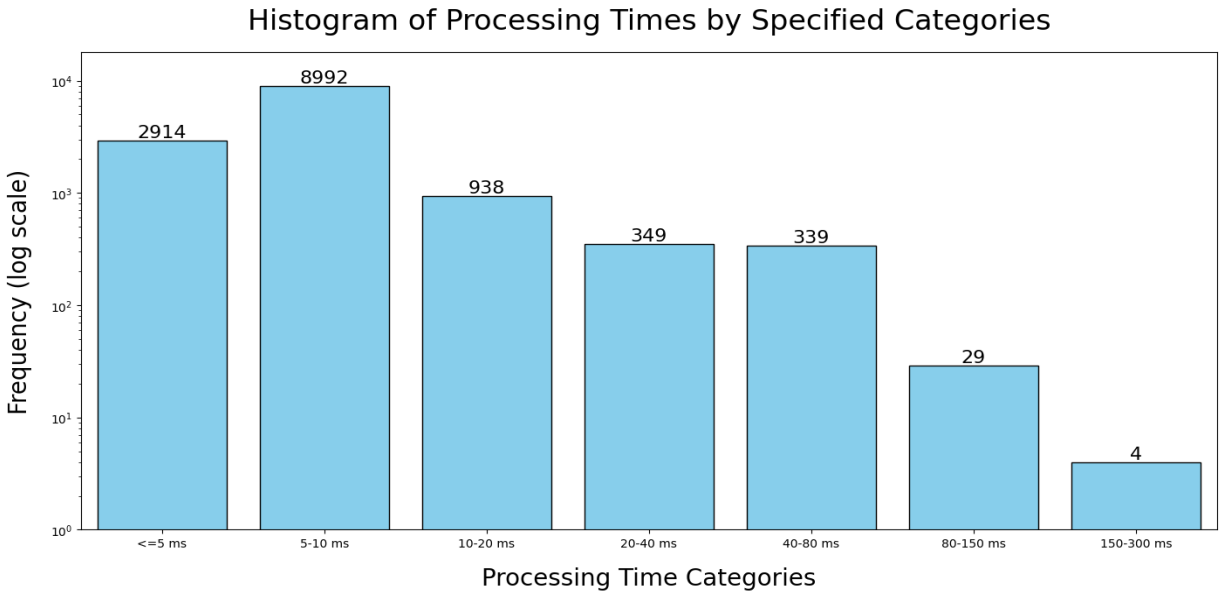
- テレメトリクスのリアルタイム取得
  - 6msでの低遅延書き込み



- 低遅延環境でのAI推論を行う
  - 人間の反射神経を凌駕するAIを利用を目指す

処理性能-テレメトリデータの登録時間：6ms

テレメトリ登録サーバ
UDPパケットのテレメトリデータを受信、Tsurugiに登録
パケットサイズは768Byte
毎秒1,050パケットを受信処理し、9テーブルに登録(Insert)
1パケット1TXで処理



平均処理時間: 8.36 ms
95%タイルの値: 20.96 ms
95%タイル以下の平均処理時間: 6.44 ms

- 1TXの処理時間をサンプリングして得られたデータのヒストグラム
- Serializableで永続化

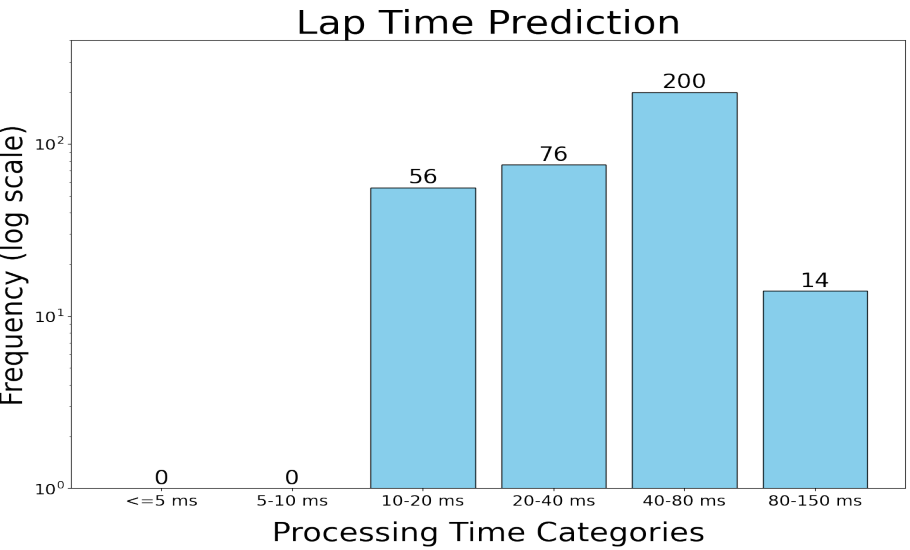
処理性能-AI推論処理：7~50m秒

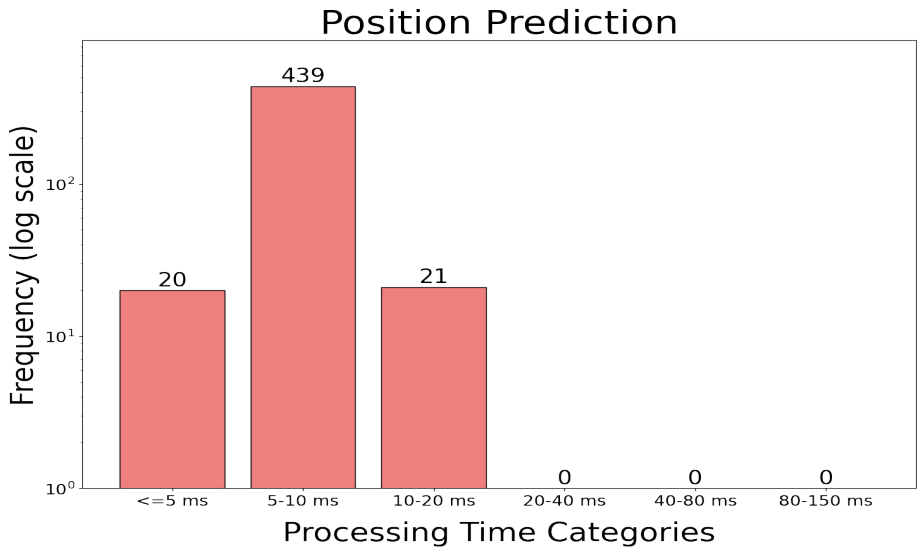
推論エンジン
　　ラップタイム予測と順位予測
　　21スレッド(21は車の台数)が、連続してTsurugiをポーリングし、ラップタイムを推論する
　　1回のポーリングで、11テーブルをIndex Joinで結合して、最大25レコードを取得(Select)する
　　推論結果を2テーブルに書き込む(Insert)

平均処理時間: 52.38 ms
95%タイルの値: 78.61 ms
95%タイル以下の平均処理時間: 49.92 ms

Histograms of Processing Times

平均処理時間: 7.19 ms
95%タイルの値: 9.83 ms
95%タイル以下の平均処理時間: 6.97 ms





- 以下の2つの推論処理を実行
  - ラップタイム予測(Lap Time Prediction)
  - 順位予測(Position Prediction)
- 上のグラフは、各スレッドごとに1,000回推論を実行した時の処理時間から1回の推論処理の平均処理時間を算出しヒストグラム化したもの
- Serializableで永続化

低遅延環境としてのエッジ・高機能プラットフォーム

- エッジAIの3っつの課題

1.エッジの高付加価値化と低コスト化の両立が困難

- AI処理/最適化の高度化が必要
  - AI処理の導入により、より高度な推論・最適化が付加価値として必要
  - 外乱要因への対応/リアルタイムでの連携・最適化
  - より高機能の端末（エッジ）側の処理では追加チップ等を端末側に搭載する必要も出てきている
- エッジ側の高機能化は、エッジ自体の高コスト化を招くことになってしまう

2.メンテナンス・コストの増大

- エッジのメンテナンスは、トラブル時には最終的には人手が必要になる
- このとき、現在の人員の不足・結果としてのノウハウの欠如、また人件費の高騰に直面している
- 可能な限り、最低限のコストでメンテナンスできることが必要

3.クラウド接続の課題

- 上位系（例：)の仕組みをクラウドで置くことの、デメリット・課題が顕在化しつつある。
  - コスト問題：特にクラウドインフラが為替の影響もあり、想定以上に値上がりにしている。また、AIを利用する場合のトークン課金は、本格的に利用する場合は高額になってしまう。
  - 遅延問題：クラウドのN/W接続のコストが上がることに加えて、そもそも遅延が高く、リアルタイム性に欠けることになる。またN/W障害によるクラウドとの接続・極端な高遅延はシステム障害につながってしまう。
  - セキュリティ
    - インターネット依存によるクラッキング
    - ランサムウェア・外部攻撃
    - 機密性情報の漏えい

解決案としてOTとITの間のTsurugi

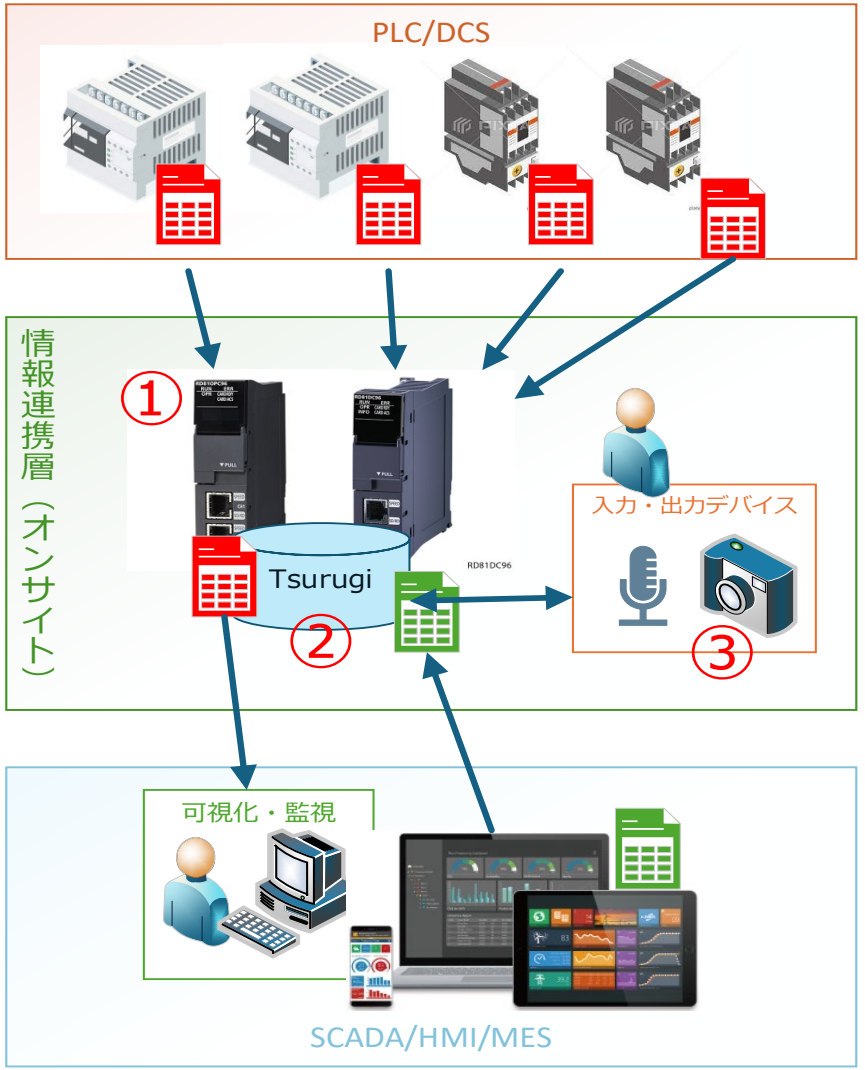
サイト内部に閉じた、OTのオンサイト・サーバアーキテクチャを強化する

- 下位層（OT処理）と上位層（分析・可視化）とつなぐ、**情報連携ユニットをTsurugiで強化**、課題解決につなげる

①プロトコル・ユニットのTsurugi連携
・DBへのインプロセス化
・または外部接続

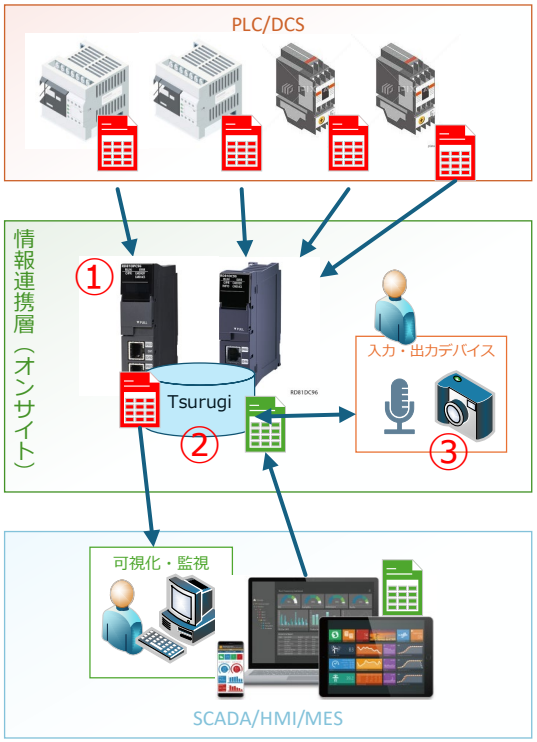
②Tsurugiで上り下りの超低遅延処理・連携指示を実行させる。超低遅延での最適化ロジック（AI）の実行を行い、最適化の精度・スピードをあげる

③後述するAI（MCP）を実装し、メンテ・問い合わせ処理や高度な処理を自然言語・AIで行うことにより、ユーザビリティを大幅に向上させる。



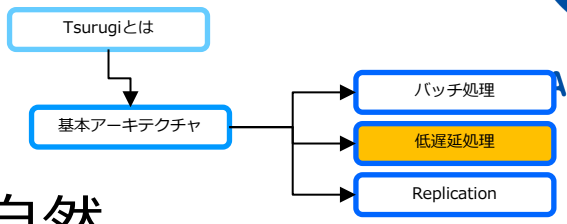
超低遅延処理と連携指示①/②

- Tsurugiでは超低遅延での書き込みが可能で、一貫性を担保しつつ、永続化し、ロックフリーにより、同時に読みだすことが可能である。
  - このハイパフォーマンスを利用して、エッジ系（例：PLC/DCS）コントローラからのログや状態情報を書き込み保存すると同時にロジックを実行し、必要な判定・実行命令の発行を情報連携ユニットで行い、PLCに実行させることができる
  - 従来であれば、エッジ側に制御ロジックを実装しておく必要があったが、サーバ側での実装が可能になり、応用性・汎用性・メンテナンス性が著しく向上する



- 超低遅延での最適化ロジック（AI）実装
  - 上記のTsurugiの高機能をさらに高度に利用して、超低遅延AIの実行を行う
  - **エッジ間を統合し、より高度なAI処理を実行することができる**
  - 単体のエッジAIではできなかった情報連携を実行することができる
  - 例：自動ドアと空調のリアルタイムなAI連動
    - 自動ドアと空調はそれぞれ独自のAI処理
    - 連動させることでトータルの効率性が見込まれる

自然言語処理の導入③



- MCPを実装し、メンテ・問い合わせ処理を自然言語で実行する

  - ローカルLLMをTsurugi側で実装し、**情報連携ユニットに保存されている、ログ情報や設定情報等を日本語（自然言語）で問い合わせる**、または更新することが可能となる。
    - 参照・処理のために画面を呼び出す必要がなく、音声入力等により、データを参照・更新することが可能となり、システム全体のユーザビリティが向上する。

  - メンテナンスを自然言語で実行して、可能であればLLMとの連携により、スムーズにトラブル回収を実施する
    - **より高度なメンテナンスをLLMで実行する**