SQL Anywhere
自己管理データベースシステム

ゼロアドミニストレーション

GLENN PAULLEY
DIRECTOR, ENGINEERING
HTTP://IABLOG.SYBASE.COM/PAULLEY
PAULLEY@SYBASE.COM

このプレゼンテーションについて

このセッションでは以下について説明します。

- 自動管理データベースの必要性
- SQL Anywhere の設計思想
- SQL Anywhereの関連機能の紹介
  - –Self management : 自己管理
  - –Self tuning : 自己チューニング
  - –Self healing : 自己回復

弊社製品デモの御紹介

ビデオ:

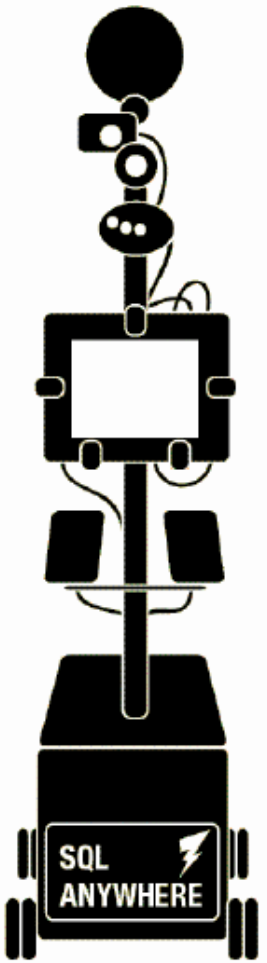
IVANANYWHERE ドキュメンタリー

YOUTUBEのIVANANYWHEREチャンネルで視聴できます。
http://www.youtube.com/user/IvanAnywhere

SYBASE iAnywhere

IVANANYWHERE & SQL ANYWHERE

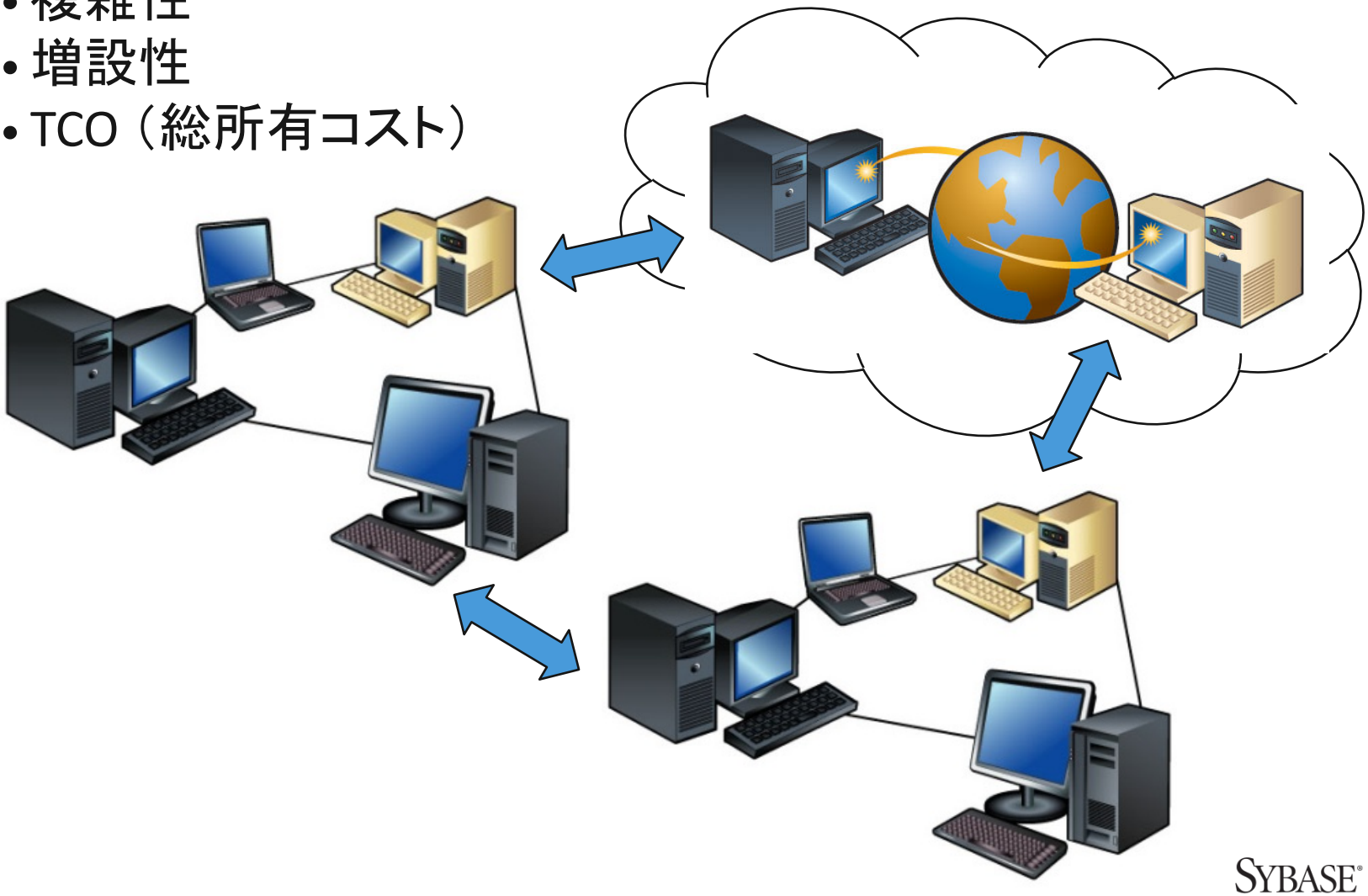
- 御紹介したビデオで使用されている「IvanAnywhere遠隔会議ロボット」は管理不要なモバイルアプライアンスの一例
- IvanAnywhereに搭載されたSQL Anywhereデータベースサーバがロボットが収集した各種センサーなどから得られるデータをリアルタイムで保管
- 収集されたデータは定期的にデスクトップサーバに同期し、分析を実行
- IvanAnywhereロボットの（cpu,disk,memory等の）ワークロードは搭載されているタブレットPCで実行されるソフトウェアにより変化
- IvanAnywhereに搭載されたSQL Anywhereデータベースサーバは IvanAnywhere全体のワークロードの変化に応じて変化し、他のアプリケーションと協調してリソース（disk,memory）を共有する必要がある



SYBASE iAnywhere

業務アプリケーションに自己管理はなぜ必要か？

- 複雑性
- 増設性
- TCO（総所有コスト）



SYBASE iAnywhere

SQL Anywhereの設計目標

- 自己管理処理:
- クロスプラットフォームのサポート:
  - 32- and 64-bit Windows
    - Windows 7, Vista, XP, Server, 2008, 2003, 2000
  - Windows CE/Pocket PC
  - Linux 32- and 64-bit
  - HP-UX, AIX, Solaris (SPARC and Intel), Mac OS/X
  - コンパイラー: Microsoft Visual Studio, GCC, SunPro, XLC (AIX), aCC (HP-UX)
- 相互運用性:
- 箱から出した状態で発揮できるパフォーマンス:
- 自動管理データベースへの総合的なアプローチ:

SQL Anywhereの物理的なデータベース設計

SQL Anywhereの物理的なデータベース設計

- SQL Anywhereデータベースは最大15のデータベース領域で構成
  - データベース領域は普通のOSファイル
  - 内2つはテンポラリーファイルとトランザクションログの為に予約
  - 残りはユーザーデータに使用可能
  - UNIXのrawデバイスは非サポート

- データベースはOSのファイルコピーだけでコピー
  - イメージコピーユーティリティーは不要
  - 違うマシンやモバイルデバイスへのデータベースの配備が容易
  - ファイルは、全対応プラットフォームで相互にコピーするだけで使用可能。変換作業も不要
    - DB内の文字コードを変えたい時等の為に変換ツールも用意しています。

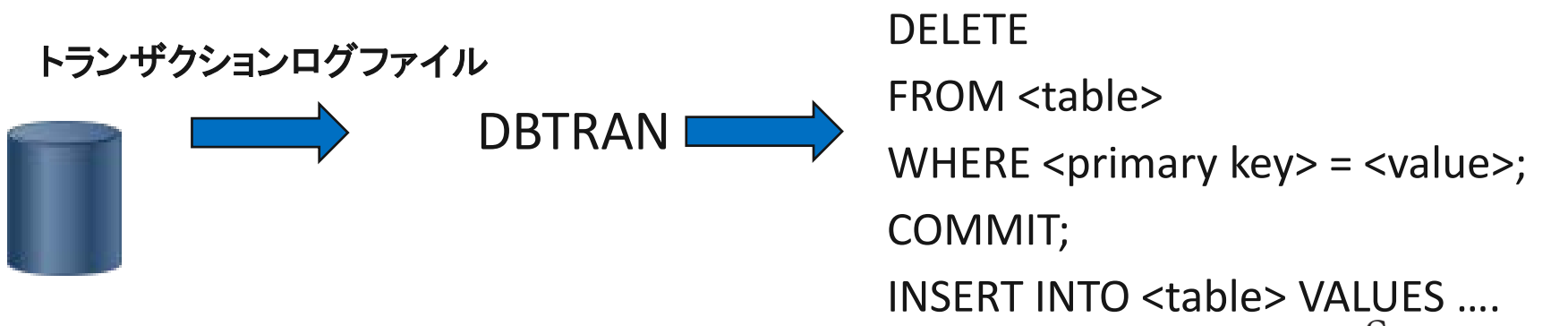
物理的なデータベース設計

- データベースファイルサイズはデータ量の増大に応じて自動的に拡大
  - ディスクが一杯になった場合はSQL Anywhereデータベースサーバが（予め設定しておいた）コールバック関数を呼び出すことが可能
  - SQL処理で一時ファイルが必要な時に枯渇しないようにテンポラリファイルガバナー機構が使用可能

- プライマリー、外部キーのインデックスは自動的に作成
  - サーバーが自動的に同義、あるいは重複するインデックスを検知
    - データベースに定義された複数個の同義・重複インデックスがディスク上の同じインデックスデータを共有することができます。

SQL Anywhereのトランザクションログ

- SQL Anywhereは行レベルの論理的なトランザクションログ機構やトランザクションリカバリー機構を使用（物理コピーやページベースではない）
- トランザクションログからINSERT, UPDATE, DELETE, COMMIT, ROLLBACK 等ログに書かれた際に実行されたSQL文に変換が可能
  - DBTRANユーティリティーで変換可能
  - 単純なバックアップのリストアからでは復帰出来ない障害からの復旧に役に立ちます。
- 弊社が提供する「Mobile Link」や「SQL Remote」等のデータベース双方向データ同期ソリューションの根幹としても使用



バッファープール自己管理

サーバーキャッシュを動的に拡大・縮小可能

動的バッファープールサイジング

- SQL Anywhereサーバーはバッファープールを必要に応じて自動的に拡大・縮小可能
  - これは以下の時に発生：
    - データベースサーバのワークロードに応じて
    - データベース上で同時に実行されるSQLに応じて
    - 同じコンピューター上の他のアプリケーションの物理メモリーの要求に応じて
      - 組み込みアプリケエーションに必須
- 全サポートプラットフォームでデフォルトで使用可能
  - 管理者はバッファープールの初期サイズと上限・下限の設定が可能

SQL Anywhereのバッファープール管理

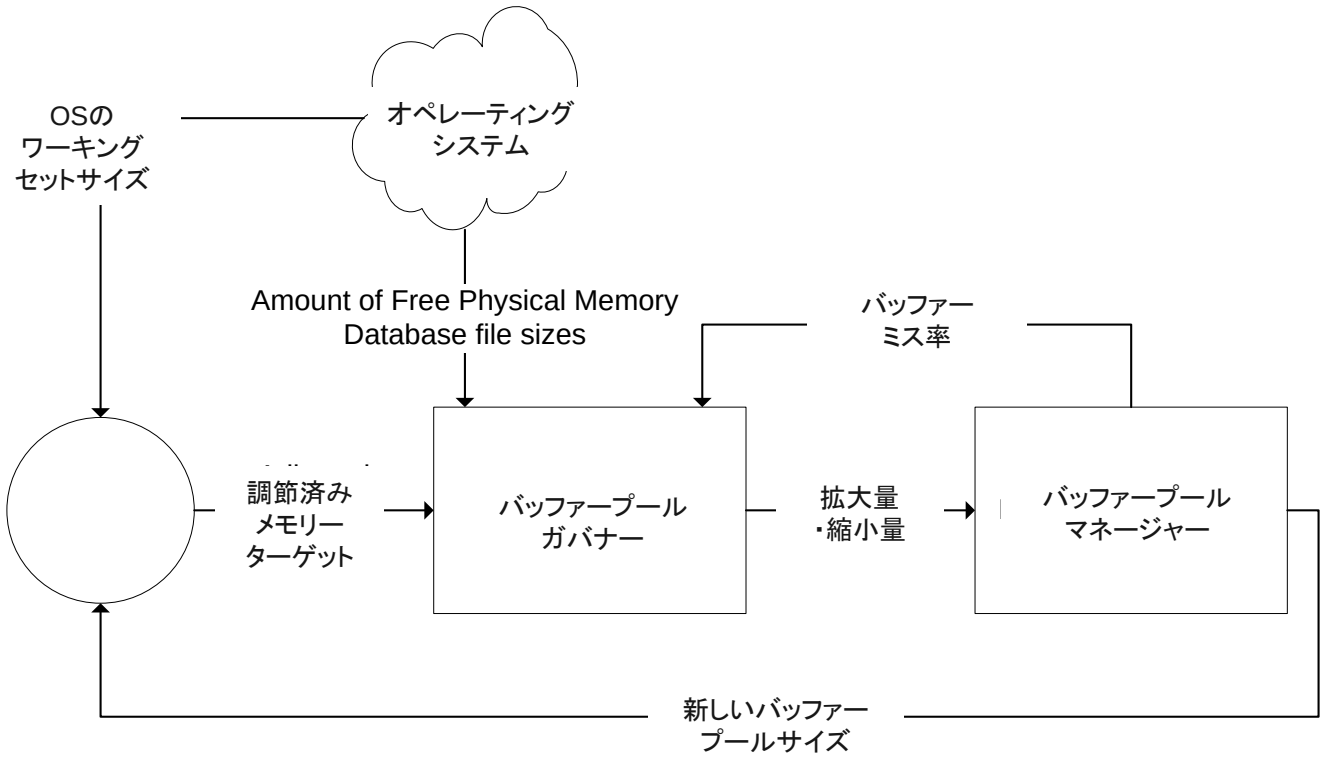
- バッファープールは一つ（用途別に分かれていない）
  - バッファーの各用途別に事前割り当てもされていない
- バッファープールは以下に使用:
  - テーブルとインデックスデータページ
  - チェックポイントログページ
  - ビットマップページ
  - ヒープページ（クエリ実行プラン、最適化グラフ、接続構造、ストアドプロシージャー、トリガーのデータ構造）
  - フリー（使用可能）ページ
- バッファーページのサイズは共通

SQL Anywhere バッファープール管理

- 単一のバッファープールは自己管理には必須
- サーバースレッド/ファイバで必要な全メモリーはバッファープールから取得
  - これによりSQL Anywhereサーバー全体のメモリー容量の管理を実現
- バッファープール内のページ変更スキーマは、バッファーページに発生する様々な変更要求をそれに応じてデマンドベースで管理可能でなければならない
  - マシンのワークロードに応じて処理時間が変化

ダイナミックバッファープールサイジング

- 基本の考え方: バッファーサイズをOSのサーバープロセスのワーキングセットサイズに合わせる
  - フィードバックコントロールループ



SYBASE iAnywhere

サーバーのマルチプログラミングレベルの自己チューニング

SQL Anywhere 12での新機能

SYBASE iAnywhere

タスクスケジューリングモデル

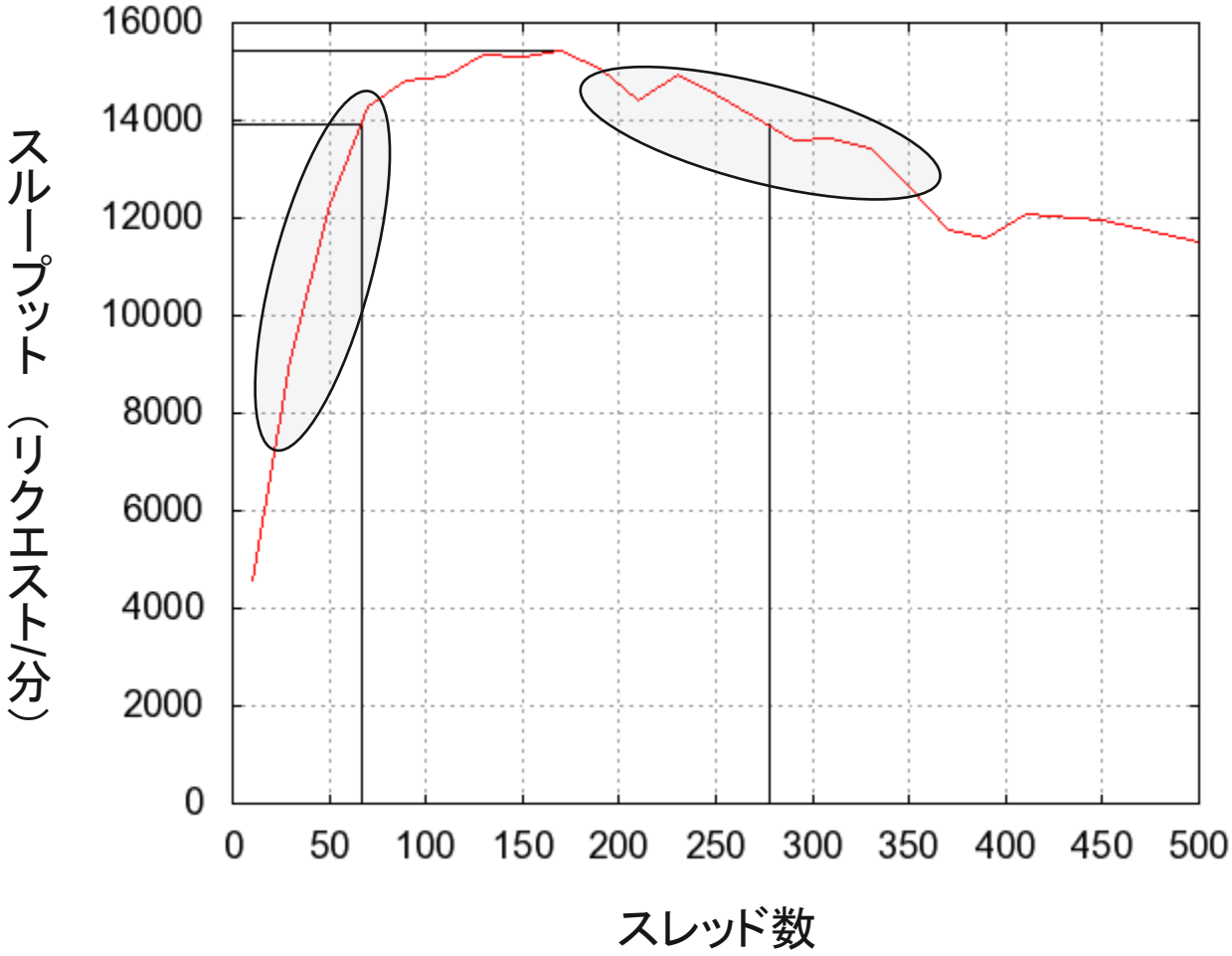
- データベースサーバーには主に2つのアーキテクチャがある:
  - Worker-per-connection:
    - ユーザー接続毎に要求を処理する“Worker”が割り当てられる
  - Worker-per-request:
    - ユーザーからの処理要求に応じて“Worker Pool”から“Worker”が割り当てられる。Workerはその処理が完了したらWorker Poolへ返される
    - 同一接続中に複数回処理を要求しても、同じ“Worker”がそのユーザーからの処理を実行する保証は無い。毎回違う“Worker”が割り当てられる

タスクスケジューリングモデル

- SQL Anywhereではworker-per-requestモデルを採用
  - 少量のリソースで動作させる上で効率的
  - マルチスレッドモデルと親和性が高い
    - ファイバをサポートしているプラットフォームではファイバを使用(Windows, Linux)
- マルチプログラミングレベル：ワーカープールのサイズ
  - ワーカープールのサイズが同時に実行できるSQLリクエストの最大数を決定
- ワーカープールサイズが使用者数、同時実行数等の「ニーズ」に合っている必要がある
  - クラウドでは、仮想化がマルチプログラミングレベルの手動設定を非常に難しくしている。

MPL（multi programing level)をワークロードにマッチングさせる

- SQL Anywhereを使用したTPC-Cに似たワークロード
- 32bit インテルデュアルコアプロセッサー、1GBバッファープール



SYBASE iAnywhere

マルチプログラミングレベルのトレードオフ

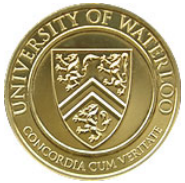
- ワークロードの大小によるトレードオフ:
  - ワーカープールが大きい場合:
    - サーバーの並列処理数が上がる
    - サーバーリソース(memory, I/O)の競合が増える
    - サーバーのバッファープールサイズが増える
    - スラッシング発生のリスクが増加する
  - ワーカープールが小さい場合:
    - ハードウェアのリソース使用率が低くなる
    - 並列処理数が制限される
    - 数によっては即処理できずに一旦「キュー」に入って待たされる可能性は増える

マルチプログラミングレベルのトレードオフ

- ワーカープールのサイズはどのように決定すべきか？
  - 考慮すべき項目:
    - メモリー、I/O、CPUリソースの利用と競合
    - 並行処理の際のロック待ちのインパクト
    - 求めるパフォーマンスと実際のパフォーマンス

SQL Anywhereのソリューション

- ワーカープールサイズの動的な自己調整機能
- （例）サーバーのマルチプログラミングレベル（MPL）
  - 処理に必要なスレッド数は必要に応じて増減
- これは要求の完了数と要求の新規受付数を監視することで行なっている。
- システムの負荷により自己調整するアルゴリズムは2つの異なった技術から構成
  - 特許申請中
  - カナダ・ウォータールー大学の支援を受けて開発

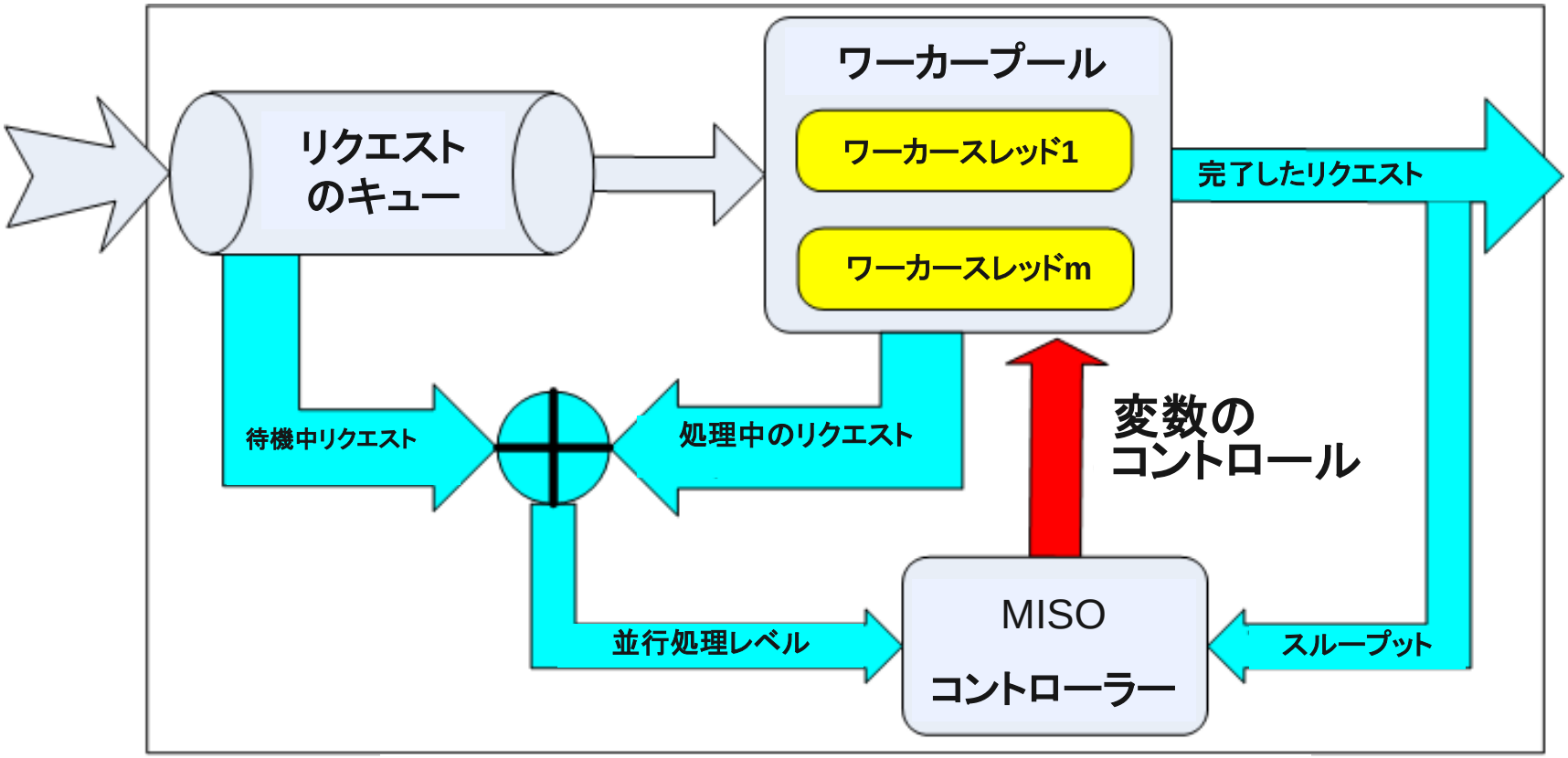


SYBASE iAnywhere

SQL Anywhereのソリューション

- 動的マルチプログラミングレベルの長所:
  - データベース管理者が考慮しなければならないパラメーターを1つ削減
  - データベースサーバの再スタートの必要なくスループットを改善
  - データベースサーバの負荷やシステムリソース状況の変化に自動的に適応
    - クラウド環境では非常に重要
- 注意:
  - 一瞬増大してその後沈静化するような「爆発的な」処理増大に対しての動的MPLの効果は限定的

SQL Anywhere スケジューラー



データベースサーバー

（worker-per-request)

SYBASE iAnywhere

SQL Anywhereのタスクスケジューリングアーキテクチャー

- サーバーは動的内部並列処理をサポート
  - 並列処理の程度は使用可能なリソースによって変動
  - 一つの処理要求から分解された各（並列）処理プランは異なるワーカーを使用
- ディスクにスワップされるのは待機中の処理要求
  - 仮想メモリーを使用し、できるだけ物理メモリに確保された容量を一定に維持しようとします。
  - サーバーの負荷増大した場合、ユーザー接続のヒープをディスクにスワップ
    - 1989年のSQL Anywhereの最初のバージョンで実装
  - 要求ベースでヒープのロードを許可するためヒープ内のポインターは適宜書き換えが発生

適応型クエリ処理

SQL Anywhere クエリオプティマイザー

- SQL AnywhereではSQLは実行する際にオプティマイザにより解析・最適化
- サーバーの状態（リソース状態やディスクの速度等）を考慮
- 複雑なクエリでも最適化処理は高速
  - left-deep treeを構成する結合が主の場合はコストベースのジョインアルゴリズムを使用
  - 複雑で入れ子となるLEFT, RIGHT, FULL OUTER JOIN　のような際はBushy tree を使用
  - 最適化プロセスはヒューリスティックとコストベースの複雑な書換の両方を含む
  - 数量的な限界（結合数等）は ない
- 長所：プランはサーバー環境、バッファープールコンテンツ/サイズ、データの変化に応じたものになります。パッケージ・バインドファイルのようなものを管理する必要がない

クエリオプティマイザーを使用しないパターン

- アクセスプランの迅速な生成のため、単一テーブルの単純なクエリは、クエリオプティマイザーで最適化「しない」
  - 例:
    - SELECT * FROM <table> WHERE <primary key column> = <value>
- ストアドプロシージャー/トリガー/イベントのクエリのアクセスプランは、将来の実行に備えてキャッシュし、再利用可
  - プランが分散(再実行の際にキャッシュしたものと違う事が多い場合)している場合はプランは「トレーニング期間」に入る
  - (変数値がない場合でも)プランの分散がない場合には、プランはキャッシュして、再利用
  - プランが“準”最適にならないようにクエリを対数的なスケールで定期的に再最適化

適応型クエリ処理

- SQL実行途中の中間結果サイズがあまりにオプティマイザが想定したものと異なる際は、オプティマイザーが（その時点から）実行可能な代替アクセスプランを作成することがある。
  - サーバーがランタイム時に自動的に代替プランにスイッチさせる
- メモリー集中処理、例えばハッシュジョインなど:
  - クエリ実行中の使用可能メモリーの変化に対して自動的に結合アルゴリズム等が変化することがある。
  - 例えばバッファープールの使用率が高い場合は代わりにメモリーの使用量が低い結合方式（例えばhash-join から nested-loop joinなど）に変化することがある
- メモリーが集中する処理、例えば、大規模なソートなどで、メモリーの割り当てをコントロールするためにサーバーにはメモリーガバナーが含まれている
  - 増大・縮小に合わせてメモリーガバナーはバッファープールサイズに適応する必要がある

適応型クエリ処理

- 幾つかの処理、例えばデータベースのバックアップなどではストレージに使用されているI/Oデバイスの性能を調査するためサンプリングプロセスが含まれる。
  - 一番の目的は使用可能なディスクヘッドの数を決定すること
  - プロセスはスループットを最大にするために「正しい」CPU数を利用可能
  - 最適化アルゴリズムは、システムの他の部分で必要なCPUに敏感。必要に応じてCPUの使用を自動的にスケールダウン

適応型内部並列処理

- SQL Anywhereのアプローチは、アクセスプランを並列処理すると「メリットがある場合にそれを実行する」こと。メリットがないと判断すれば並列処理はしない。
  - 並列処理の程度は、最適化中のクエリのコスト計算によって決定
- 並列ワークはワーカープールからそれぞれ独立して割り当て
  - プランは、並列処理の程度によって広範囲に自己チューニング
  - 使用可能なワーカー数が少ない場合は、並列処理によるワーカーの枯渇を防ぐ設計

自己ヒーリング統計

SQL ANYWHERE 12 新機能

自律型統計管理

- 1992年に実装されたSQL Anywhereの機能
  - 初期の実装では、カラムの密度と頻出値統計を管理するためにhash-baseの構造を使用
- 今日
  - 自己管理のカラムヒストグラム
    - ベーステーブルとテンポラリーテーブルの両方
    - 自動的に統計を更新
  - 最適化処理中の中間結果分析のためのジョインヒストグラム
  - サーバーはリアルタイムで 永続的インデックス統計を維持
  - さらに、最適化中にサーバーでインデックスサンプリングを実行可

カラムヒストグラム

- 自己チューニングの実装
  - スタンダード[範囲]バケットや頻出値統計の両方を実装
  - 述部の評価とINSERT, UPDATE, MERGE、DELETE 文の結果とともにリアルタイムにアップデート
  - デフォルトでは、統計は毎データ操作SQL リクエストを実行する間に計算
- LIKE述部の最適化に使用するストリングスの統計のキャプチャにはNovelの技術を使用
- ヒストグラムはLOAD TABLEまたはCREATE INDEX文で自動的に計算
  - 必要に応じて明示的に作成/ストップ
  - ヒストグラムはデフォルトでアンロード/リロードにわたって保持

自己ヒーリング統計へのモチベーション

- 統計の自己チューニングはトランザクション的ではない
  - オーバーヘッドを低く保つ
  - 統計はROLLBACK文やコンカレントな行修正の際にはエラーを招く可能性がある
- 設計による、自己チューニングアルゴリズム、近似を使用
  - より効果的、しかし少々不正確な方法を単一行のUPDATE, INSERT DELETE 文に使用
  - 統計の生成にはデータを一度スキャンすることが必要
  - 正確さとオーバーヘッドのトレードオフ;完璧は難しい
    - 厳しいデータの偏りでは評価エラーが発生する可能性あり
- 自己チューニングは、ビジーなサーバーでは「キープ」できない可能性がある
  - 自己調整した統計の質は独断で下げてしまう。
  - システムは自身を監視して修復する必要がある
    - クラウドの配備には必要不可欠

SQL Anywhere のソリューション

- 統計的ガバナー: SQL Anywhereの新機能
  - バックグラウンドサーバー処理の内部システム
  - 使用につれての統計の「質」を自己監視
  - 「おそまつな」統計を自己ヒーリング
  - 「悪い」統計はとり除く
- 方法
  - クエリ処理中の評価エラーを自動的にカテゴリ分けし記録
  - 様々な技術を使用して自動的に統計を収集
    - 他のSQL クエリのピギーバッキング、インデックス分析、階層化サンプリング
  - 統計的保全性を決定
  - オプティマイザで統計的使用を監視
    - 使用されていないままの場合はヒストグラムをメモリーからアンロード
  - 低オーバーヘッド

まとめ

まとめ

- SQL Anywhereでは、それぞれの機能が自己管理できるように設計
- 自己管理機能は総合的に考慮しないといけない
  - just-in-timeの最適化と適応型クエリ処理のどちらもサポートしなければ動的バッファープールサイジング機能をサポートする意味ない
- 実装には多くの困難・課題
  - 製品テストが困難
  - 自己管理の要件と全体のサーバーのパフォーマンスとのバランスをとる必要がある
  - オーバーヘッドを低くするのは難しい
  - 制御不能な動作の回避の必要性

Thank You!

- QUESTIONS OR COMMENTS?

- PAULLEY@SYBASE.COM
- HTTP://SQLANYWHERE-FORUM.SYBASE.COM
- HTTP://IABLOG.SYBASE.COM/PAULLEY

SYBASE® iAnywhere®