野村総合研究所

NRI 未来創発
Dream up the future.

コボルコンソーシアムセミナー in XDev 2009

セッション2A-3

COBOLとクラウド

2009年 9月16日

株式会社野村総合研究所
情報技術本部
先端技術開発部
久保　順一

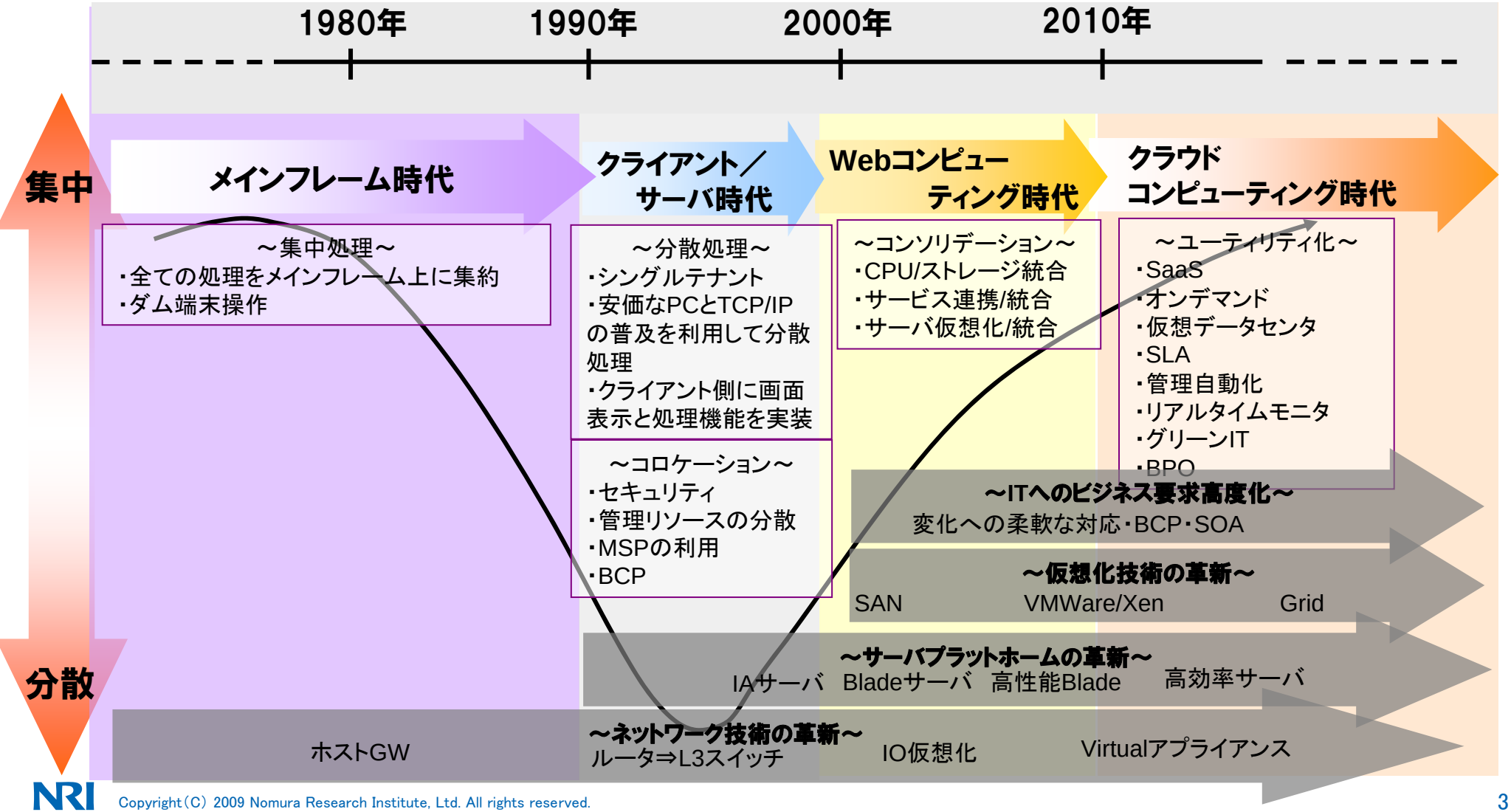
〒100-0005
東京都千代田区丸の内1-6-5 丸の内北口ビル

目次

1.　クラウドコンピューティングとは

1．1　次世代のコンピューティング・パラダイム

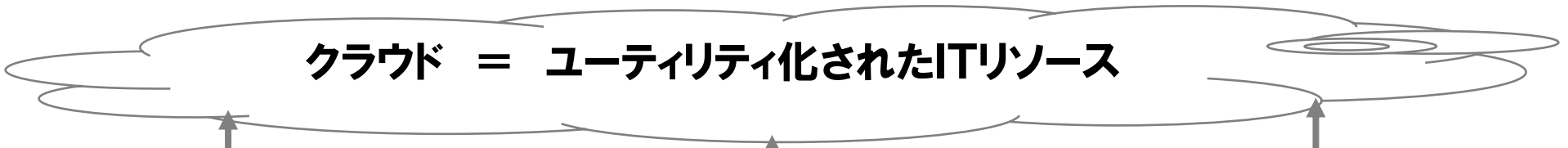
・メインフレーム全盛期の集中処理
・オープンシステムの抬頭によるクライアント・サーバなどの分散処理
・インターネットに代表されるネットワーク中心の、新しい集中処理
・更に、接続先を意識せずにサービスを受ける、クラウド・コンピューティングの処理形態



1．　クラウドコンピューティングとは
1．2　クラウド・コンピューティング（Cloud Computing）の定義

■ クラウド・コンピューティングの定義

「クラウド（日本語で雲の意）」とはインターネットを指す。雲のようにつかみどころのないインターネット上のコンピュータリソースを必要に応じサービスとして利用するという概念



コンピューティングリソースのユーティリティ化に必要な三要素

Abstract

■ 利用者にとって、ハードウェア管理が高度に抽象化されている

- 稼働しているサーバ環境はハードウェア環境に対し非依存
- 稼働しているハードウェアは地理的にどこに存在しうる

Elastic

■ 利用者にとって、インフラのキャパシティは上げる方にも下げる方にも伸縮性がある

- キャパシティは劇的かつ即座に増減させることができる（従来のホスティングサービスとの違い）

Variable

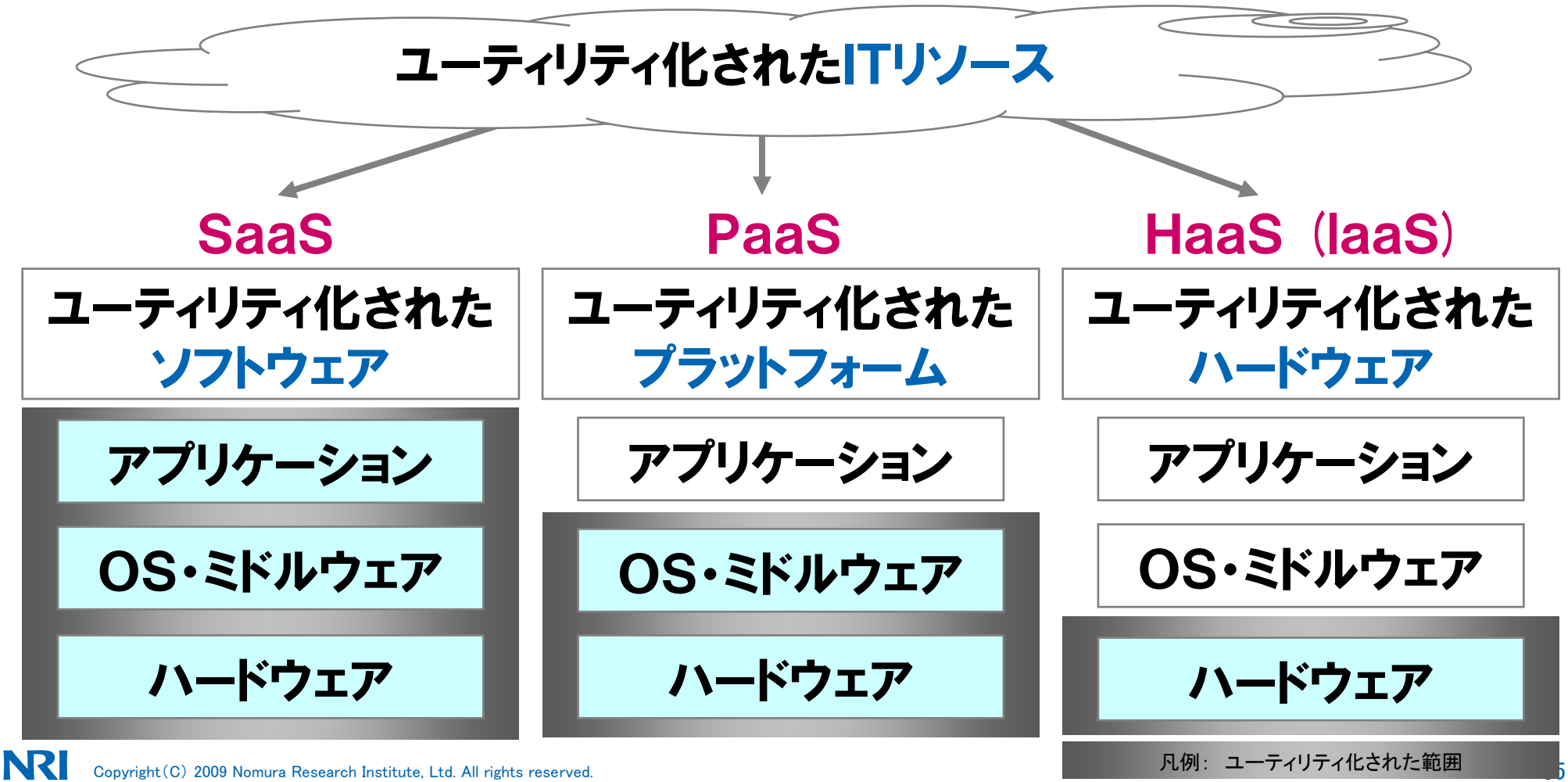
■ 利用者にとって、インフラ費用は変動費化とできる

- CPUやストレージ、ソフトウェアの利用に応じて費用を支払う

1.　クラウド・コンピューティングとは

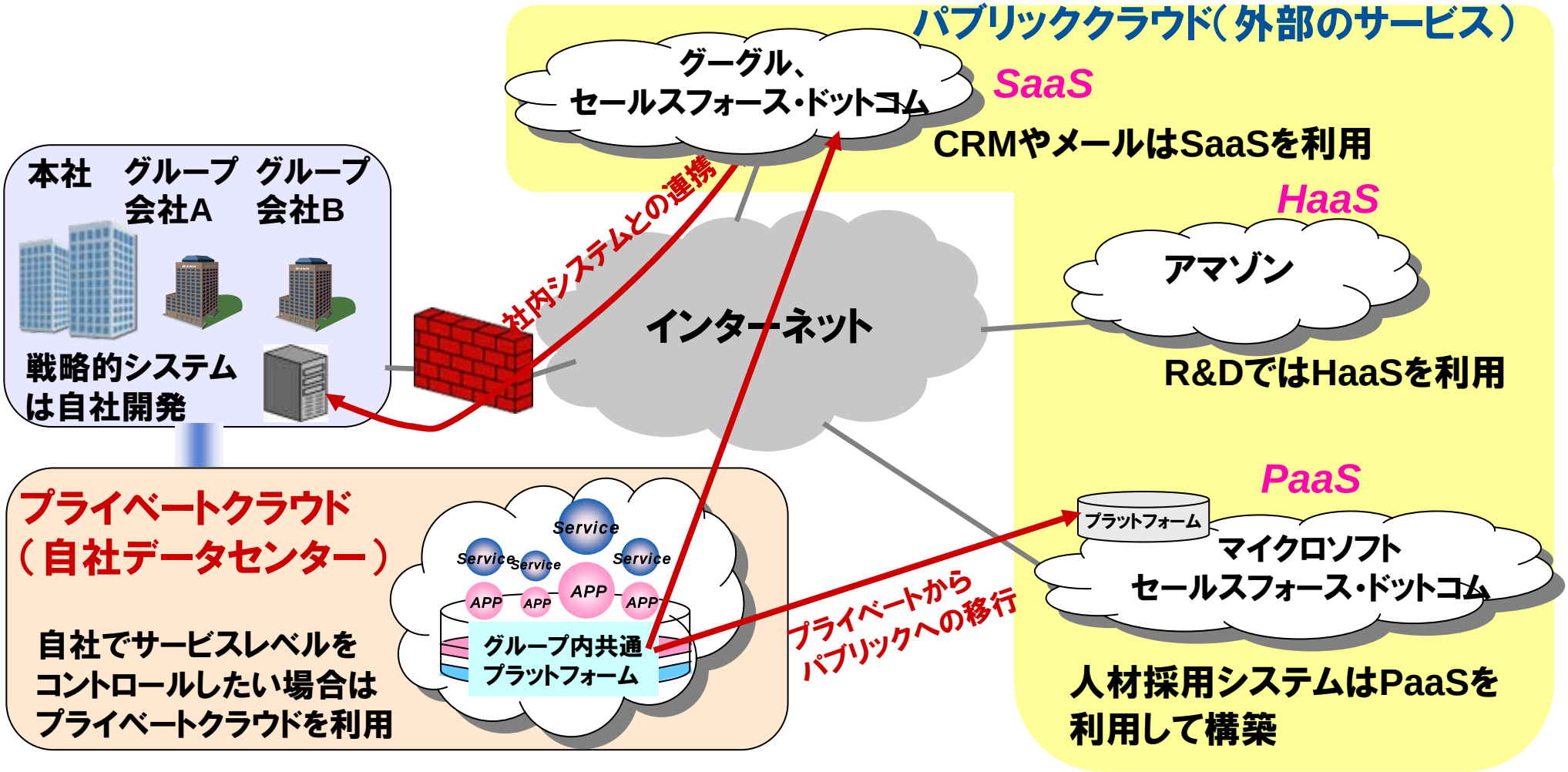
1．3　クラウド・コンピューティングサービスの分類

■ITリソースをユーティリティ化する範囲の違いで、クラウド・コンピューティングを分類できる。 SaaS／PaaS／HaaSの3つが一般的な分類



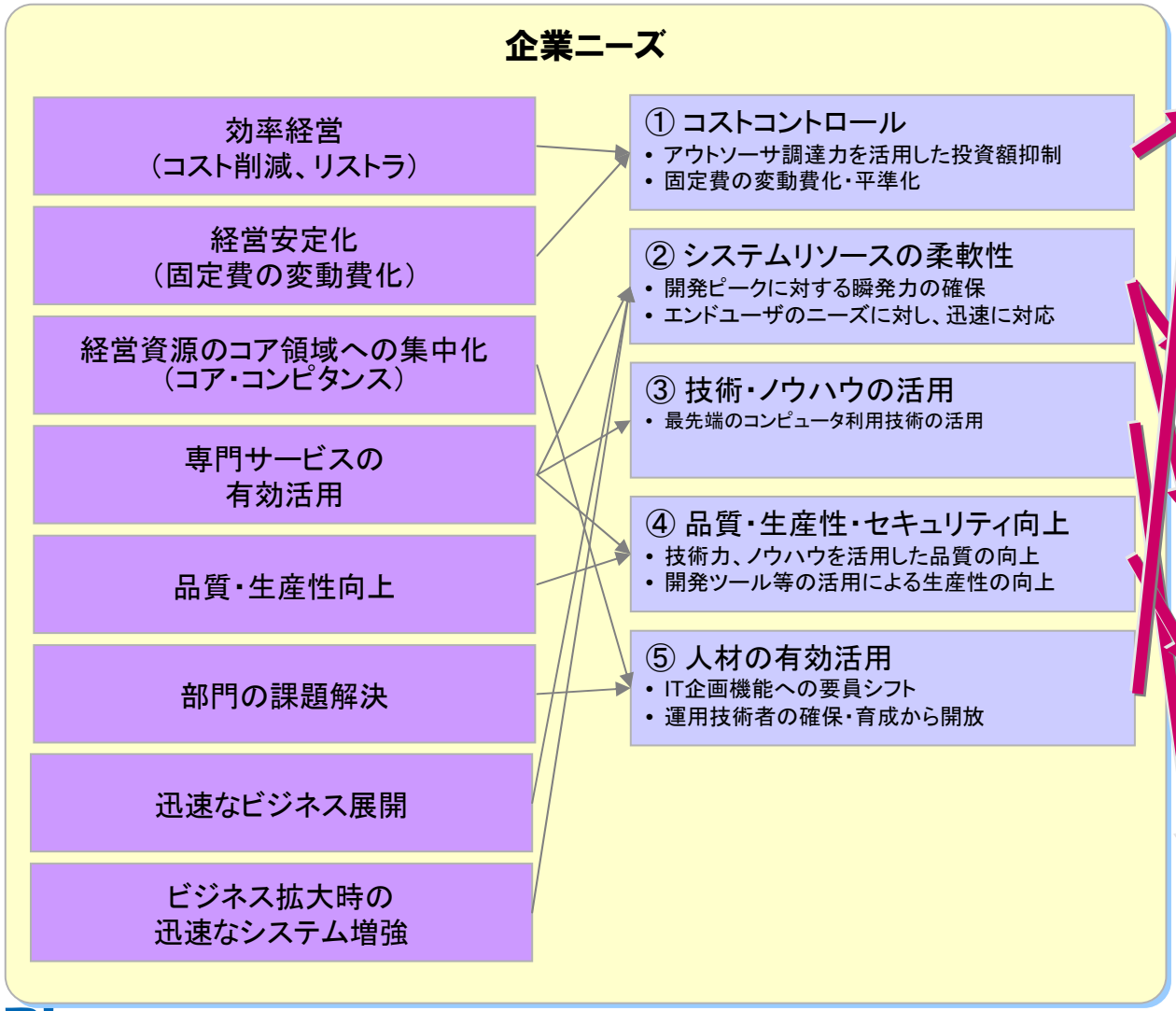
1.　クラウド・コンピューティングとは
1．3　クラウド・コンピューティングサービスの分類

■クラウド時代に企業が求められるのは、
- 自社開発、クラウドコンピューティングサービスの適切な使い分け
- 大企業であれば、グループ企業が共同で利用可能なプライベートクラウドの構築も有効



1.　クラウド・コンピューティングとは
1．4　企業ニーズとクラウドコンピューティング

■ 企業課題を解消するアプローチとして「クラウド・コンピューティング」が問題解決の１手法として注目されている。



《クラウド・サービス利用の利点》

- ■ **ITコストの変動費化**
  - ✓ 資本・運用コストの削減（スモールスタート、インフラ資産を保持しない）
  - ✓ 必要な時に一定期間利用（自由なオンオフ切替）
  - ✓ インフラ構築SEが不要、運用管理負荷の軽減
- ■ **ITリソースの調達スピード**
  - ✓ インフラ調達・設定の迅速化
  - ✓ ビジネスニーズに応じて迅速な開発、利用が可能
- ■ **ITリソースの柔軟拡張**
  - ✓ オンデマンドでシステムリソースを拡張可能
  - ✓ システム基盤のキャパシティプランニングが不要
- ■ **開発生産性**
  - ✓ 用意された環境、開発フレームワークを利用することで標準化、開発生産性向上が図れる場合がある
  - ✓ インフラエンハンス作業が必要ない
- ■ **最新技術への対応**
  - ✓ 最新のコンピュータ技術、製品を利用可能

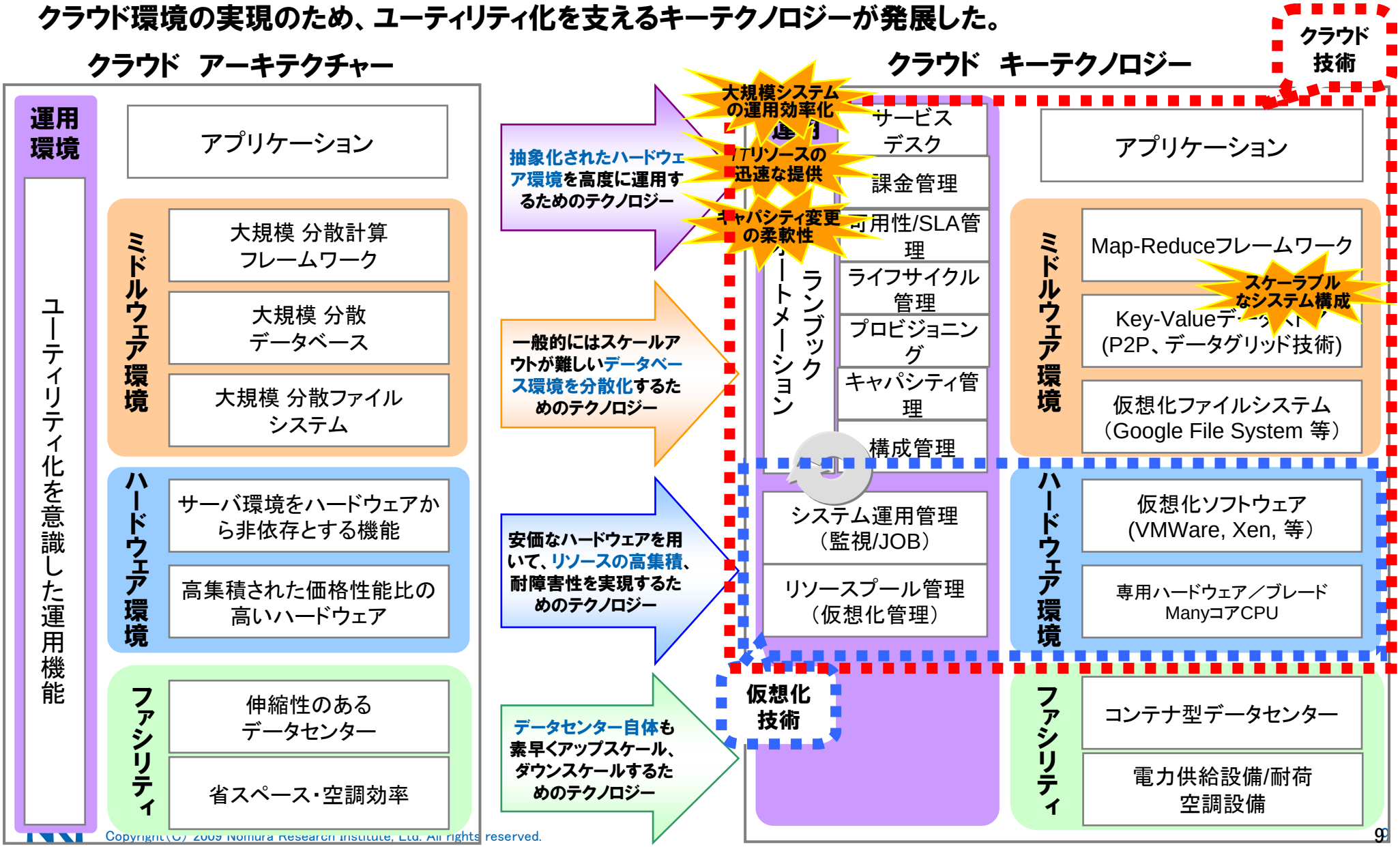
２　クラウド・コンピューティングのアーキテクチャー

- ２．１　クラウド・コンピューティングのキーテクノロジー
- ２．２　クラウドコンピューティングの構成パターン
- ２．３　クラウド・コンピューティングの一般的なアーキテクチャー
- ２．４　アプリケーションのスケールアウトの課題はDB
- ２．５　スケーラブルなアーキテクチャでの並列分散処理

2.　クラウド・コンピューティングのアーキテクチャー

2．1　クラウド・コンピューティングのキーテクノロジー

クラウド環境の実現のため、ユーティリティ化を支えるキーテクノロジーが発展した。

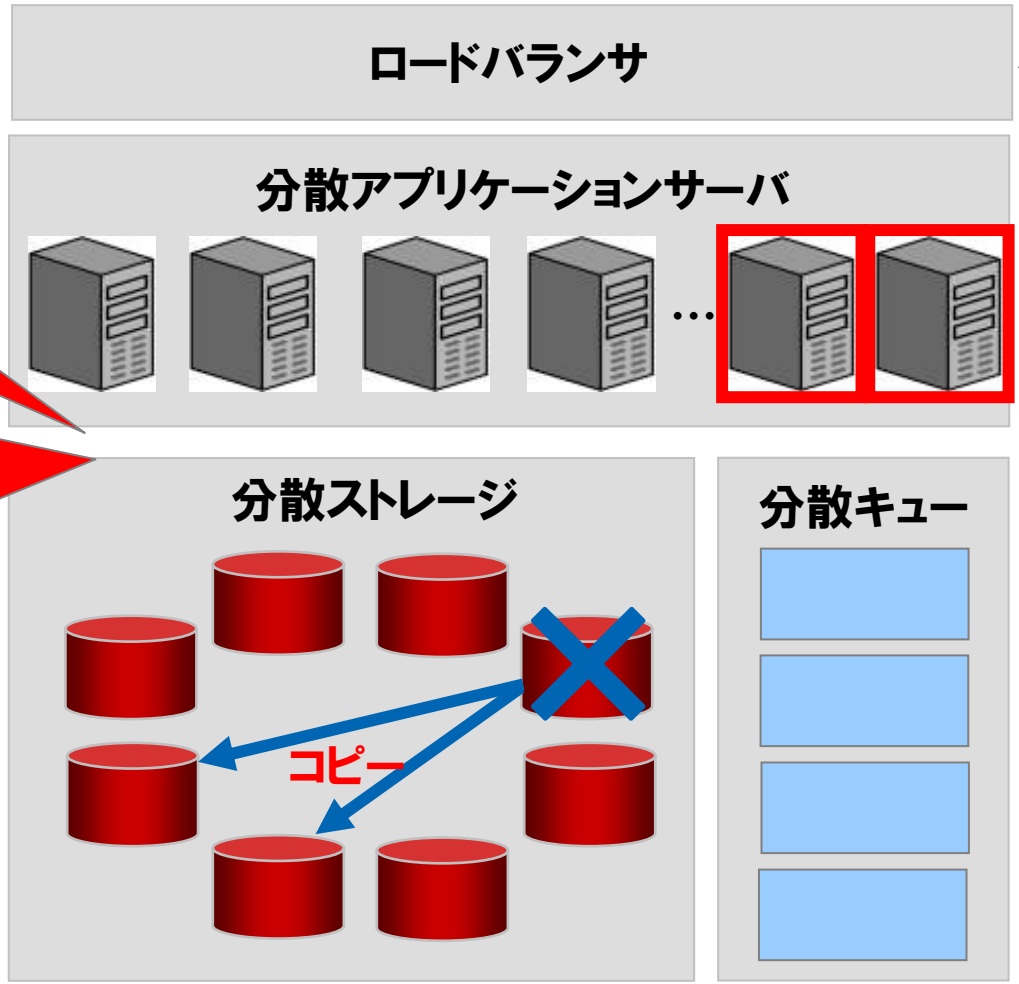


2.　クラウド・コンピューティングのアーキテクチャー
2．2　クラウドコンピューティングの構成パターン

	クラウドパターン			パブリッククラウド		
プラットフォーム	既存互換型	既存互換 運用高度化型	スケーラブル型	AmazonEC2	Google AppEngine	Windows Azure
特性	仮想化	仮想化 オートノミック	仮想化 オートノミック スケーラブルDB	仮想化 オートノミック スケーラブルDB	仮想化 オートノミック スケーラブルDB	仮想化 オートノミック スケーラブルDB
形態	IaaS/PaaS/SaaS			IaaS	PaaS	PaaS
言語	IaaS：自由 PaaS：指定	IaaS：自由 PaaS：指定	IaaS：自由 PaaS：指定	自由	Python Java	.NET Frameworkサポート言語 (C#,VB,COBOL)
データ操作 DBMS	SQL RDBMS	SQL RDBMS	GetSet KVS	SQL/GetSet RDBMS KVS（SinpleDB）	GQL KVS（BigTable）	SQL/GetSet RDBMS KVS（AzureStorage）
仮想化 マルチテナント	ハイパーバイザ	ハイパーバイザ	ハイパーバイザ/ ミドルレイヤ	ハイパーバイザ (Xen)	ミドルレイヤー (WebAP)	ハイパーバイザ (Hyper-V)
ファシリティ	従来型	従来型	従来型 コンテナ型	コンテナ型	コンテナ型	従来型
仮想化運用	監視/JOB運用/ /ﾘｿｰｽﾌﾟｰﾙ管理	監視/JOB運用/ /ﾘｿｰｽﾌﾟｰﾙ管理	監視/JOB運用/ /ﾘｿｰｽﾌﾟｰﾙ管理	監視/ﾘｿｰｽﾌﾟｰﾙ 管理	監視/ﾘｿｰｽﾌﾟｰﾙ 管理	監視/ﾘｿｰｽﾌﾟｰﾙ管理
クラウド運用 （オートノミック）	なし/ﾌﾟﾛﾋﾞｼﾞｮﾆﾝｸﾞﾂｰﾙ	動的ﾌﾟﾛﾋﾞｼﾞｮﾆﾝｸﾞ/ｵｰﾄｽｹｰﾘﾝｸﾞ	動的ﾌﾟﾛﾋﾞｼﾞｮﾆﾝｸﾞ/ｵｰﾄｽｹｰﾘﾝｸﾞ	ﾌﾟﾛﾋﾞｼﾞｮﾆﾝｸﾞ/ｵｰﾄｽｹｰﾘﾝｸﾞ	動的ﾌﾟﾛﾋﾞｼﾞｮﾆﾝｸﾞ/ｵｰﾄｽｹｰﾘﾝｸﾞ	動的ﾌﾟﾛﾋﾞｼﾞｮﾆﾝｸﾞ/ｵｰﾄｽｹｰﾘﾝｸﾞ

2．3　クラウド・コンピューティングの一般的なアーキテクチャー

- 価格対性能比の高い、安価なサーバを用いたスケールアウト構成が基本
- 個々のサーバではなく、システム全体として、高パフォーマンス、高可用性を実現



スケールアウトが容易なアーキテクチャ

価格対性能比の高い安価なサーバ数万～数百万台で構成

基本的なサーバの考え方

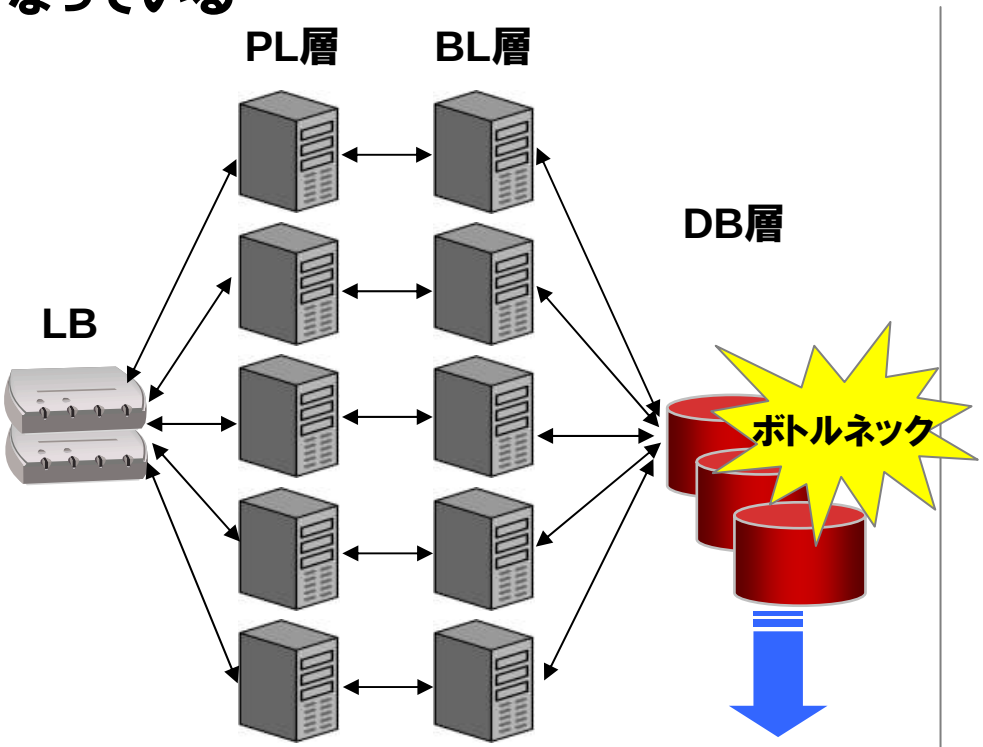
個々のハードウェアは、高性能・高信頼性、高可用性・冗長構成高価格

個々のハードウェアでの障害発生は当然

ハードウェア単体ではなく、システム全体として高性能・高可用性を担保

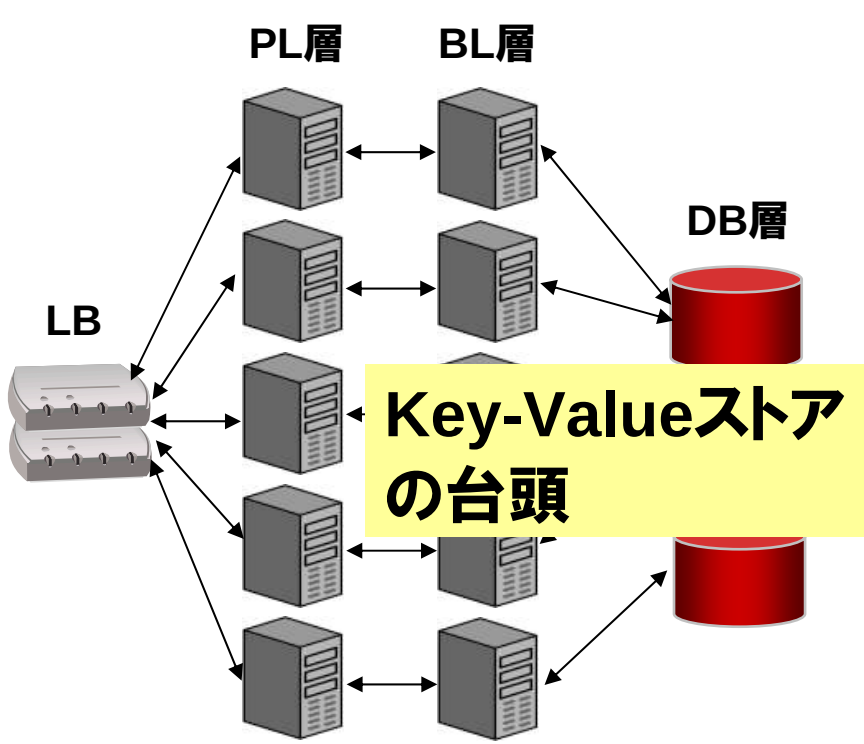
2．4　アプリケーションのスケールアウトの課題はDB　「ACID」から「BASE」

■PL層とBL層のスケールアウトは可能でも、DB層へのアクセスが全体のボトルネックとなっている



Atomicity　（原子性）
Consistency（一貫性）
Isolation　（独立性）
Durability　（永続性）

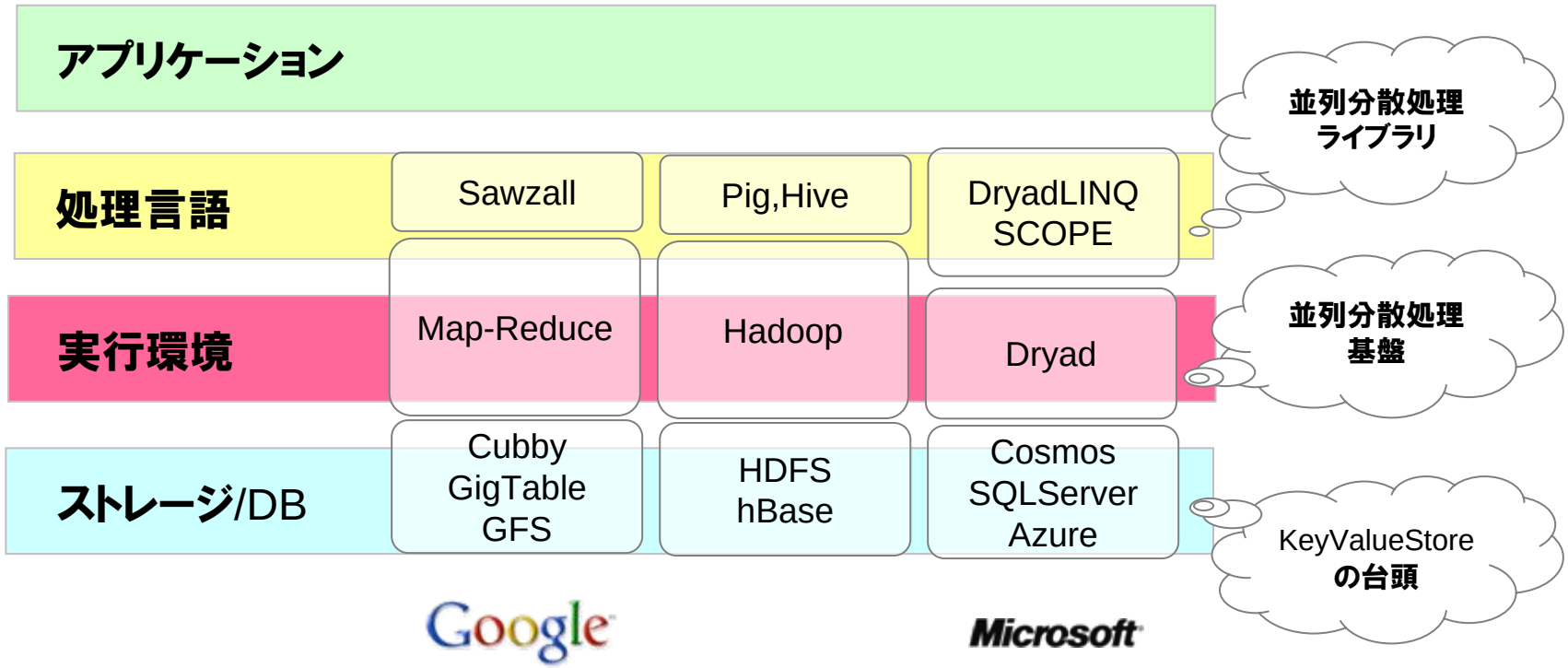
スケーラビリティを追求するとRDBが備えるACID特性の維持が困難



Basically **A**vailable　（可用性優先）
Soft-state　（柔軟な状態）
Eventual consistency（結果一貫性）
いつかは全ての複製の同期が取れて、システム全体が一貫した状態になること

2．5　スケーラブルなアーキテクチャでの並列分散処理

- これまでのパラダイムでは実現不可能であった超大規模データボリュームの処理実現
- 並列分散処理を支えるミドルウェア群の顔ぶれ



３　ITモダナイゼーション

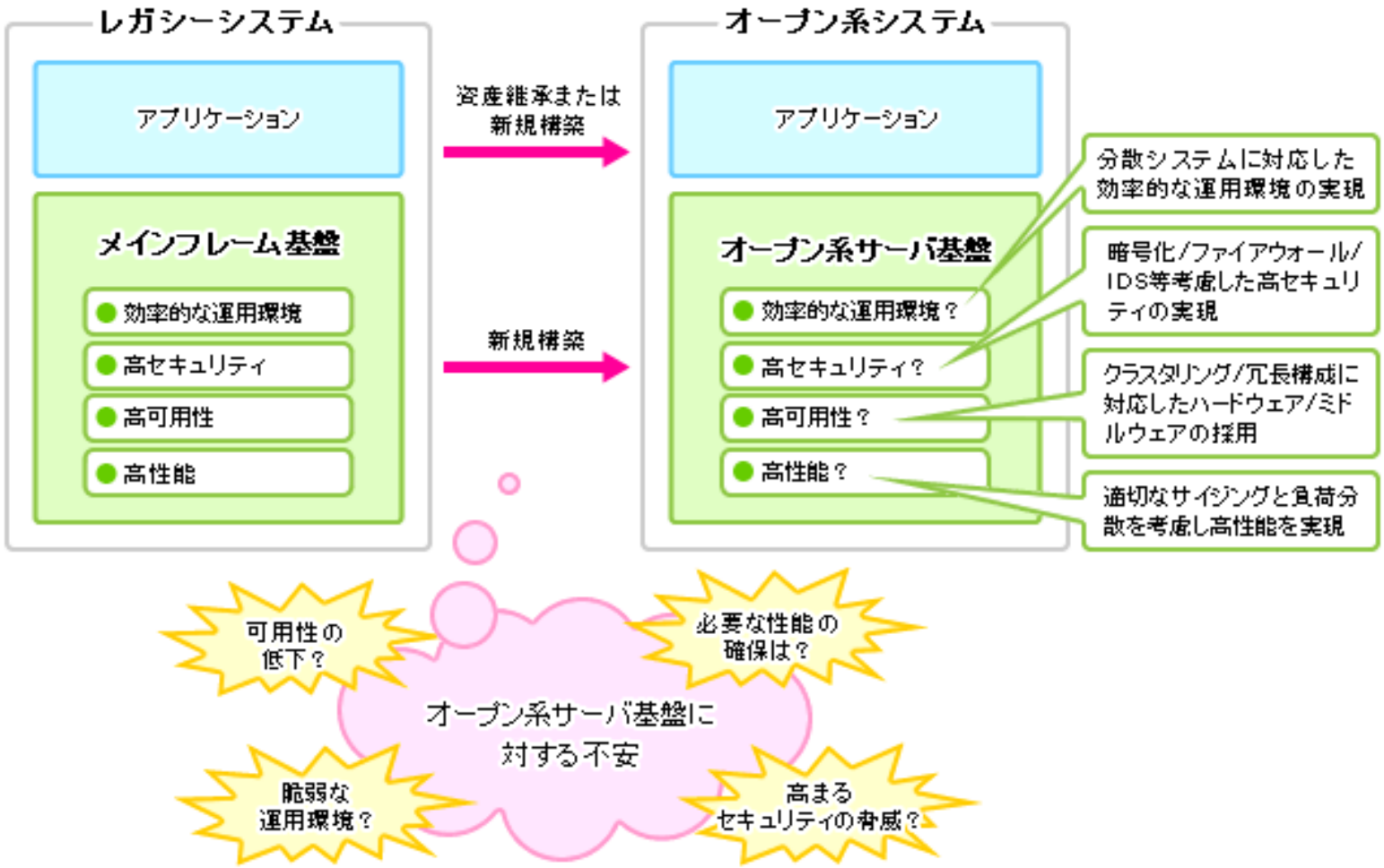
３．１　レガシーマイグレーションの難しさ

３．２　クラウド時代のITモダナイゼーション

3.　ITモダナイゼーション
3．1　レガシーマイグレーションの難しさ

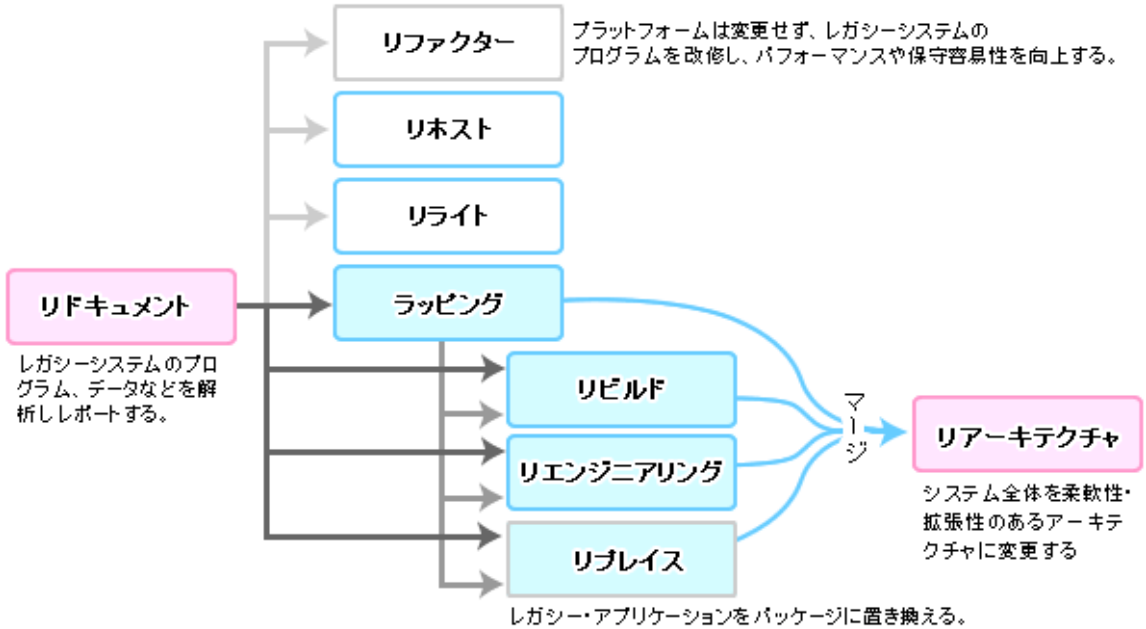
■レガシーマイグレーションによる基盤運用コスト削減の実現は容易ではなかった

■マイグレーションにかかるコストや難易度も高く、メリットに対するリスクが大きい



3.　ITモダナイゼーション
3．2　クラウド時代のITモダナイゼーション

- モダナイゼーション（Modernization）とは「近代化」を意味する言葉。個々のシステムを新しいプラットフォームに移行することがゴールではなく、システム全体を柔軟性・拡張性のあるアーキテクチャに近代化させることをゴールとしている。
- 寿命の長い基幹系システムにおいて「柔軟性」「拡張性」を得るためには、ラッピング・リビルド・リエンジニアリング・リプレイスを活用し、システム全体のリアーキテクチャが必要
- またCOBOLで記述された品質の良いビジネスロジックはSOA的アプローチによる再利用も有効である
- アプリケーション特性に合わせて各手法を検討することになるが、基盤プラットホームとして、クラウドという選択肢も「リアーキテクチャ」の視野に入れる時代になりつつある



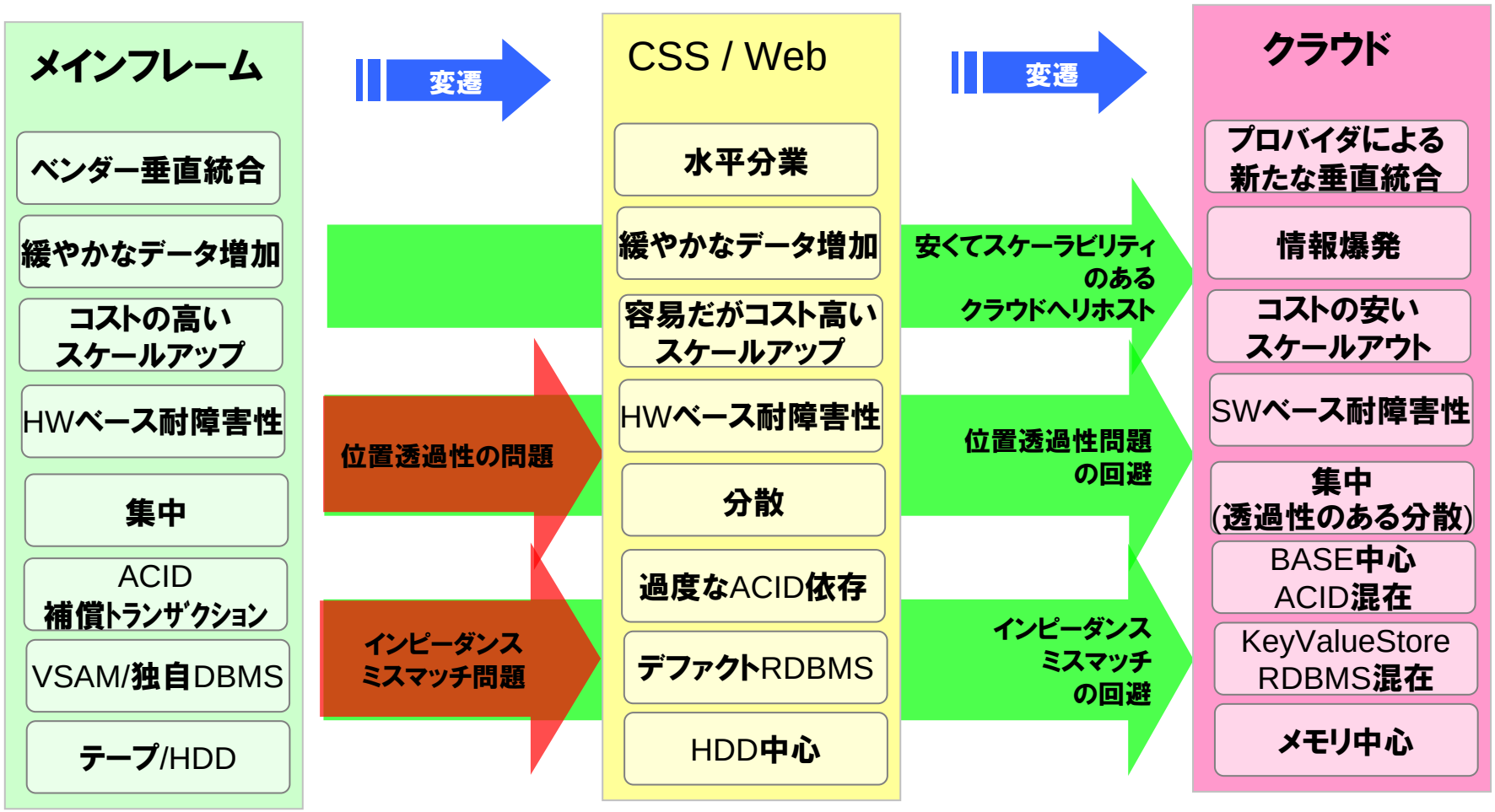
４　COBOL開発とクラウド

- ４．１　従来型コンピューティングとクラウドコンピューティングの特徴
- ４．２　レガシーシステムからクラウドへ
- ４．３　ISAM/VSAMベースとKeyValueStore
- ４．４　クラウドプラットフォーム＆COBOLベンダーの方向性と対応状況

4.　COBOL開発とクラウド
4．1　従来型コンピューティングとクラウドコンピューティングの特徴

■各世代のコンピューティングパラダイムの視点の変遷・特徴の整理（基盤・アプリ）

■レガシーシステムからのマイグレーションパスの課題点や期待していること



4.　COBOL開発とクラウド
4．2　レガシーシステムからクラウドへ

ISAM/VSAMベースのCOBOLアプリと新DBMSとしてのKeyValueStore

【課題意識】

- これまでのレガシーマイグレーションにおいてはRDBMS等へのリライト等を行い、ダウンサイジング等のシステム再構築を経ていた。特にISAM/VSAMベースのアプリケーションはRDBMSのインピーダンスミスマッチな部分の問題が大きかった。
- クライアントサーバ・Webコンピューティング時代のアプリケーションロジックは、BL層とDB層（SQL）に記述可能であり、業務システムが扱うデータ量の増大と伴い、RDBMSへの過度な依存が性能面・基盤コスト（巨大で高価なSMPマシン）・アプリ保守性のバランスをとることが非常に難しくなってしまった。
- 業務システムにおいては、ファイルベースのバッチ処理が合理的な処理パターンとして現在も多く存在している。特にオンライン化が必要とされていないバッチ中心のホスト上でのCOBOLアプリも膨大に存在している。

- クラウド時代になり、安価で均質なマシンをソフトウェアの技術により「可用性」と「スケーラビリティ」を得られるようになりホストのDBMS基盤からクラウド分散DBMSへの基盤リアーキテクトの可能性
- COBOL時代のアプリケーション処理を新DBMSとしてのKeyValueStoreとして、リアーキテクトできる可能性があると考える。
- ホスト上のCOBOLバッチがクラウドの技術的特徴であるMapReduce等の分散並列処理により、非常にコストパフォーマンスの良いシステムアーキテクチャの実現が可能になる可能性がある。

4.　COBOL開発とクラウド
4．3　ISAM/VSAMベースとKeyValueStore

金融システムの多くは口座をキーとした処理であるので、口座単位DB・口座単位並列処理とすることで、スケールとパフォーマンスを実現できる、との仮説を置いた。

従来型〔RDBMSの時代〕

ー業務別にテーブルを持ち、店群分散でスケール対応
ー“店群”が実装され、“店群変更”によるシステム変更が広範囲におよぶ

従来型（≒業務別テーブル）

顧客証券預り_tbl
- 明細レコード　A口座、X銘柄、・・・
- 明細レコード　B口座、Y銘柄、・・・
- ・・・

拡張限界が低く、苦労も多い。

処理方式変革

口座単位

●口座単位にデータセットを持つ
●データと同様に処理系も口座単位に分散させ、口座単位のスケールアウトを実現
●ビジネス拡大と同じ軸でシステムを拡張していく

口座主体型（≒key-valueストア）

A口座管理_tbl
- 証券預り明細レコード　X銘柄、・・・
- 受発注明細レコード　M銘柄、・・・
- 金銭残高レコード　・・・
- ・・・

DBと処理系を口座単位で拡張していく。

実はVSAM時代はこういうデータアクセスをしていた。

◆口座単位DBの概念は（クラウド技術の）key-valueストアに近似
◆クラウド＝スケーラブルDBと大規模並列処理に適する
⇒クラウド技術動向を注視し、適用可能性を探っていく。

4. COBOL開発とクラウド

4.4 クラウドプラットフォーム＆COBOLベンダーの方向性と対応状況

COBOLベンダー / クラウドベンダー	概況	マイクロフォーカス	日立	富士通	備考
マイクロソフト「Azure」	COBOLベンダー各社ともにAzure未対応。 マイクロフォーカスが対応を発表。日立、富士通は検討中。 マネージドコードとしてのCOBOLはAzure前提稼動条件を満たせば稼動可能だが、アンマネージドコードとしてのCOBOL対応は各社検討の段階。	Azure未対応。 2008/10 PDCにて、英マイクロフォーカス社との提携を発表。 NET3.5/VS2008への対応済み	Azure未対応。 対応検討中。 Azure前提対応されていない。	Azure未対応。 対応検討中。 Azur前提対応はされている状況。 また.NET4.0/VS2010への対応も検討中	Azure前提対応 .NET3.5/VS2008 将来的にはNET4.0/VS2010に。
Amazon「AWS」	IaaSであるため、まだ試用フェーズであるが、COBOLベンダーとの協力は推進している	AWS上でβ提供。 MicrofocusServer for SOA/EEの試用が開始されている(非常に戦略的なミドルである)	対応検討中。	対応検討中。	
日立「Harmonious Cloud」		ー	日立PaaSの一部としてサービス提供を可能とするミドルについては、開発環境、実行環境などの想定されるケースを見極めながら検討中	ー	
富士通「Trusted-Service Platform」		ー	ー	富士通ミドルの「Interstage」上でのサポートを表明。	

５　まとめ

5.　まとめ

■クラウド時代になり、安価で「可用性」と「スケーラビリティ」を得られるようになってきた

■COBOL時代のアプリケーション処理を新DBMSとしてのKeyValueStoreとして、リアーキテクトできる可能性がある

特に、ISAM/VSAMベースのCOBOLソースはクラウドのKeyValueStoreにマッチしやすく、スケーラビリティを持ったアプリケーションに生まれ変わるという点では、クラウド移行を新たなITモダナイゼーションの１アプローチと考えると面白いのではないか