MicrosoftのCloud OS Windows Azureについて

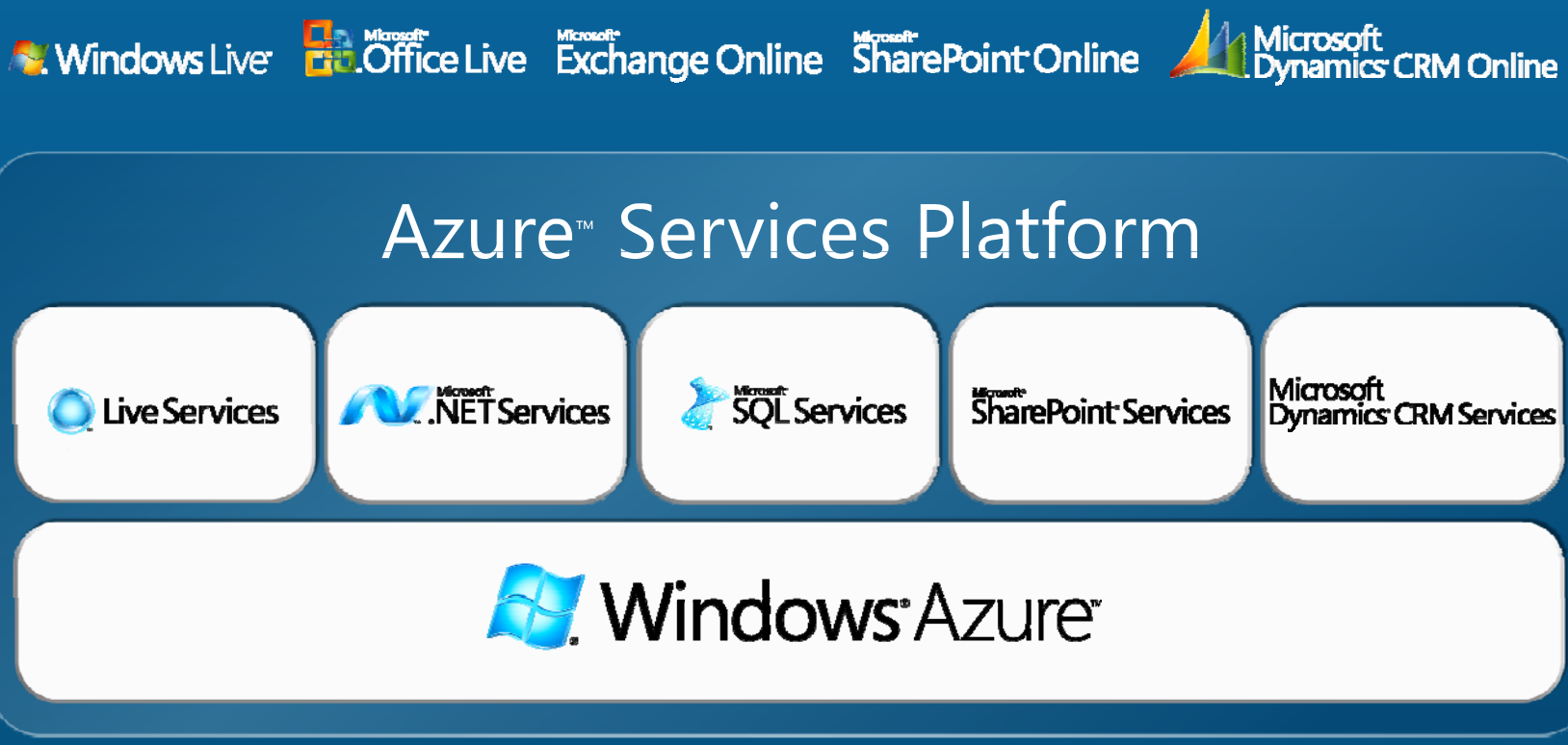
早稲田大学
丸山不二夫

Agenda

- Windows Azure とは何か?
- WebアプリケーションのScale-outと分散メモリー・キャッシュ
- Azure WebアプリケーションのScale-out
- Azure SDS（SQL Data Service）
  - Azure SDSのデータモデルとテーブルの構造
  - Azure SDSのPartition
- Azure のService Model
- Azure Fabric Conrollerと Routing and Data Overlay

マルレク2008

Windows Azure とは何か?


Windows Live
Office Live
Exchange Online
SharePoint Online
Microsoft Dynamics CRM Online
Azure™ Services Platform
Live Services
.NET Services
SQL Services
SharePoint Services
Microsoft Dynamics CRM Services
Windows Azure

Windows Azureとは何か?

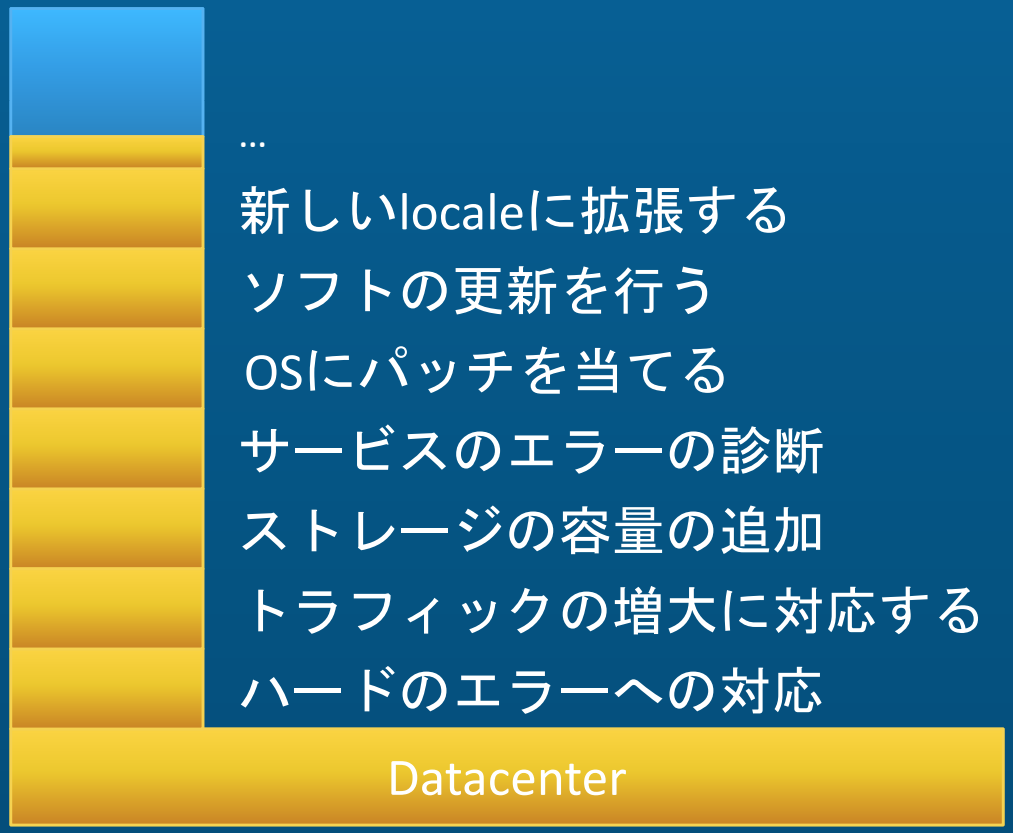
- CloudのOperating System
- Utility computingの為にデザインされた
- 次の4つの基本的な特徴を持つ
  - Service management
  - Compute
  - Storage
  - Developer experience

デスクトップのアプリを次のような手順で開発することを想像してみよう

- ハードを選んで、組み立てる
- デバイスドライバを見つける
- ファイルシステムを書く
- ジョブスケジューラを書く
- アプリのインストーラを書く
- …

とんでもなく手間がかかる！

しかし、これがCloudサービスの開発者が、今日では行っていること！



Business logic

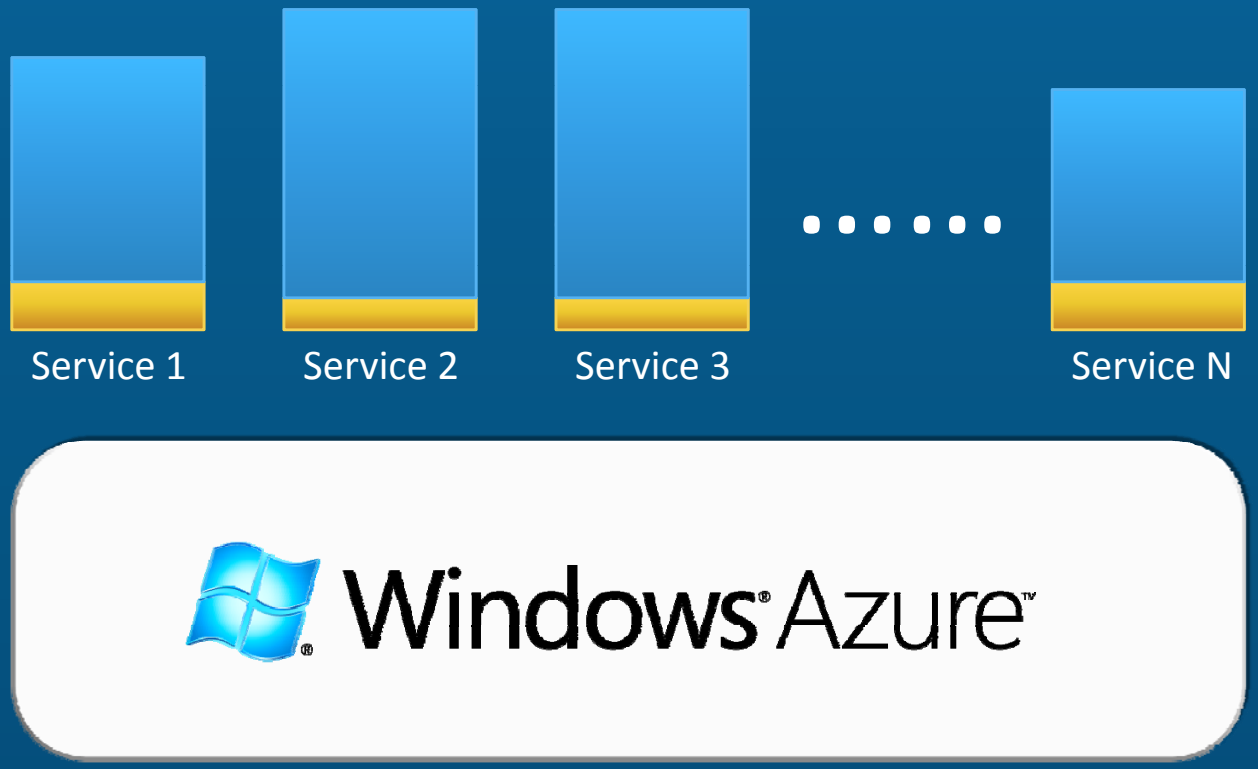
Service “glue” and operations

デスクトップでの解答は？
Operating System:

- ハードからは抽象化されたアプリの実行環境
- アクセスコントロール付きの共有ファイルシステム
- 共有プールからのリソースの割り当て
- 強力な開発環境のサポート
- 他のシステムとのインターオペラビリティ
- ……

Cloudで欠けていたものは?

CloudのためのOperating System



Cloud OSが提供すべきものは何か?

- デスクトップOSが提供するのと同じ機能。ただし結合されたサーバ上で。
  - 抽象的な実行環境
  - 共有ファイルシステム
  - リソースの割り当て
  - 開発環境
- さらに、 Utility computing
  - 24時間/7日 の可用性
  - 従量制課金
  - 簡単で、見通しのよい管理

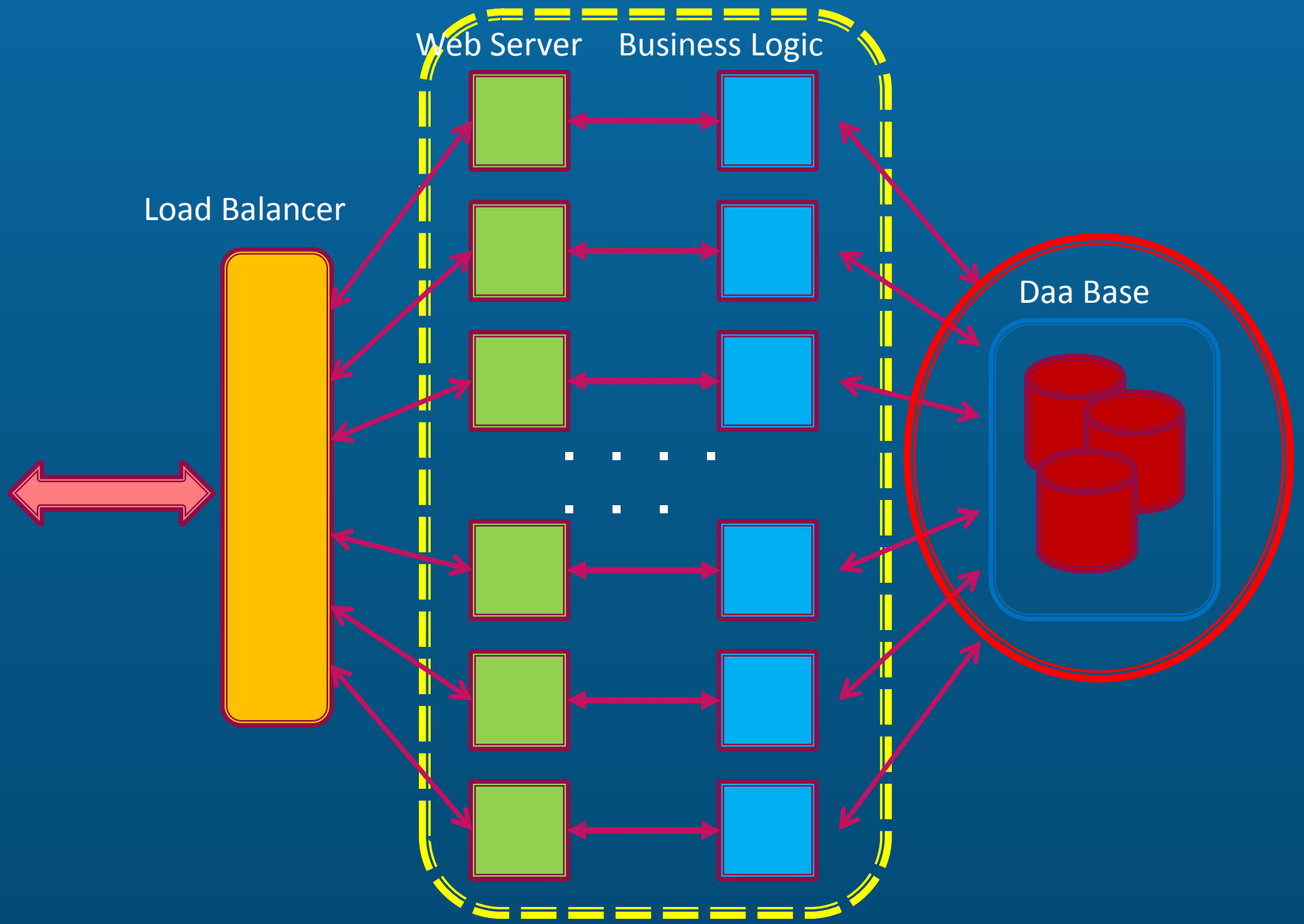
Azure Cloud OS が可能としたもの

- サービス管理の自動化
  - ルールを定義してコードを与える
  - プラットフォームはルールに従って、サービスを配備しモニタし管理する
- 強力なサービスのホスティング環境
  - 全てのハードウェア:
    servers; load balancers; …
  - 仮想化と直接の実行
- スケーラブルで可用性の高いCloudストレージ
  - Blobs, tables, queues, …
- 豊富で、かつ身近な開発環境

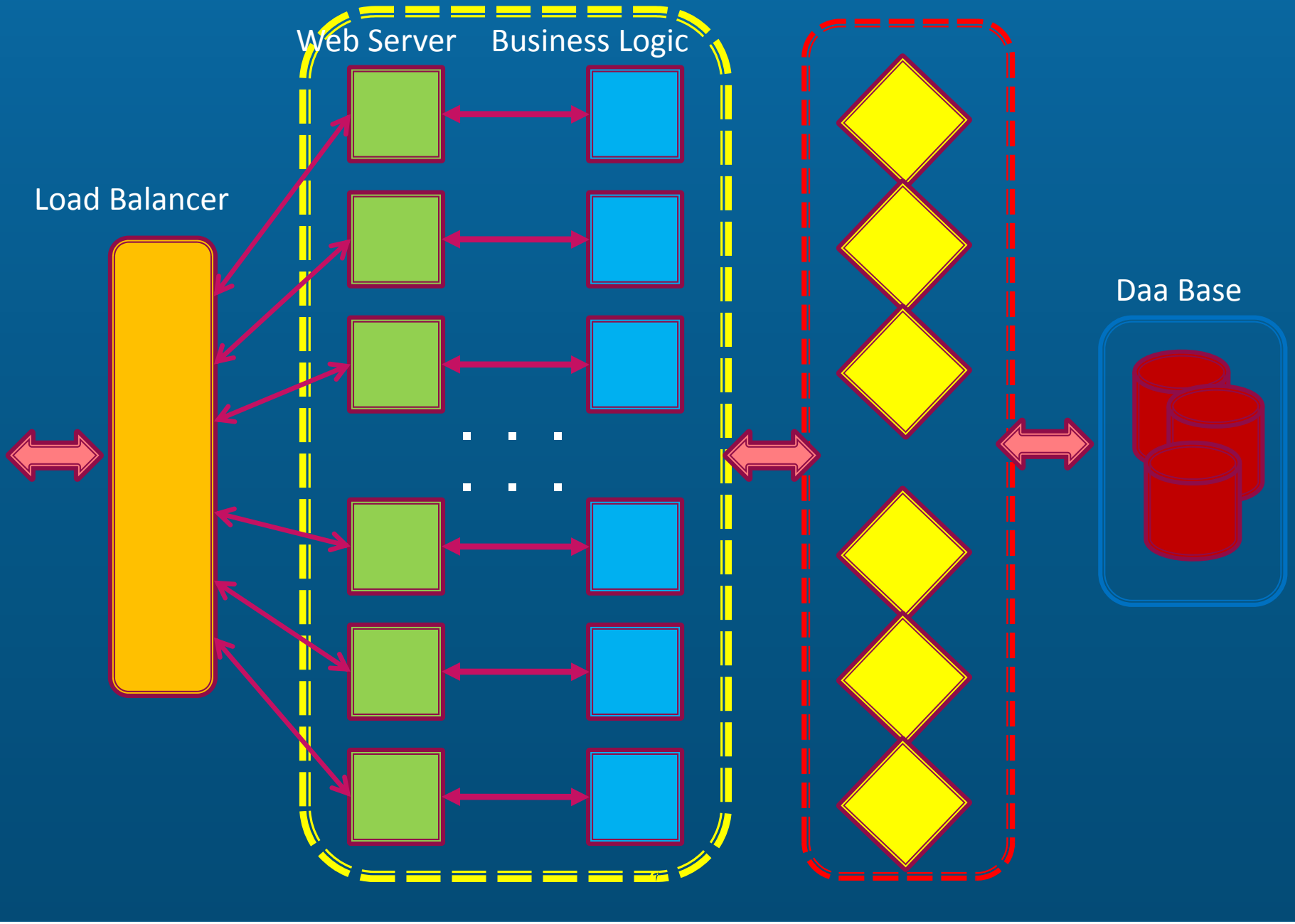
マルレク2008

Webアプリケーションの Scale-outと分散メモリーキャッシュ

Web Appli Multi-tier のScale-outのBottle Neck



Multi-tier のScale-outと分散メモリーCache

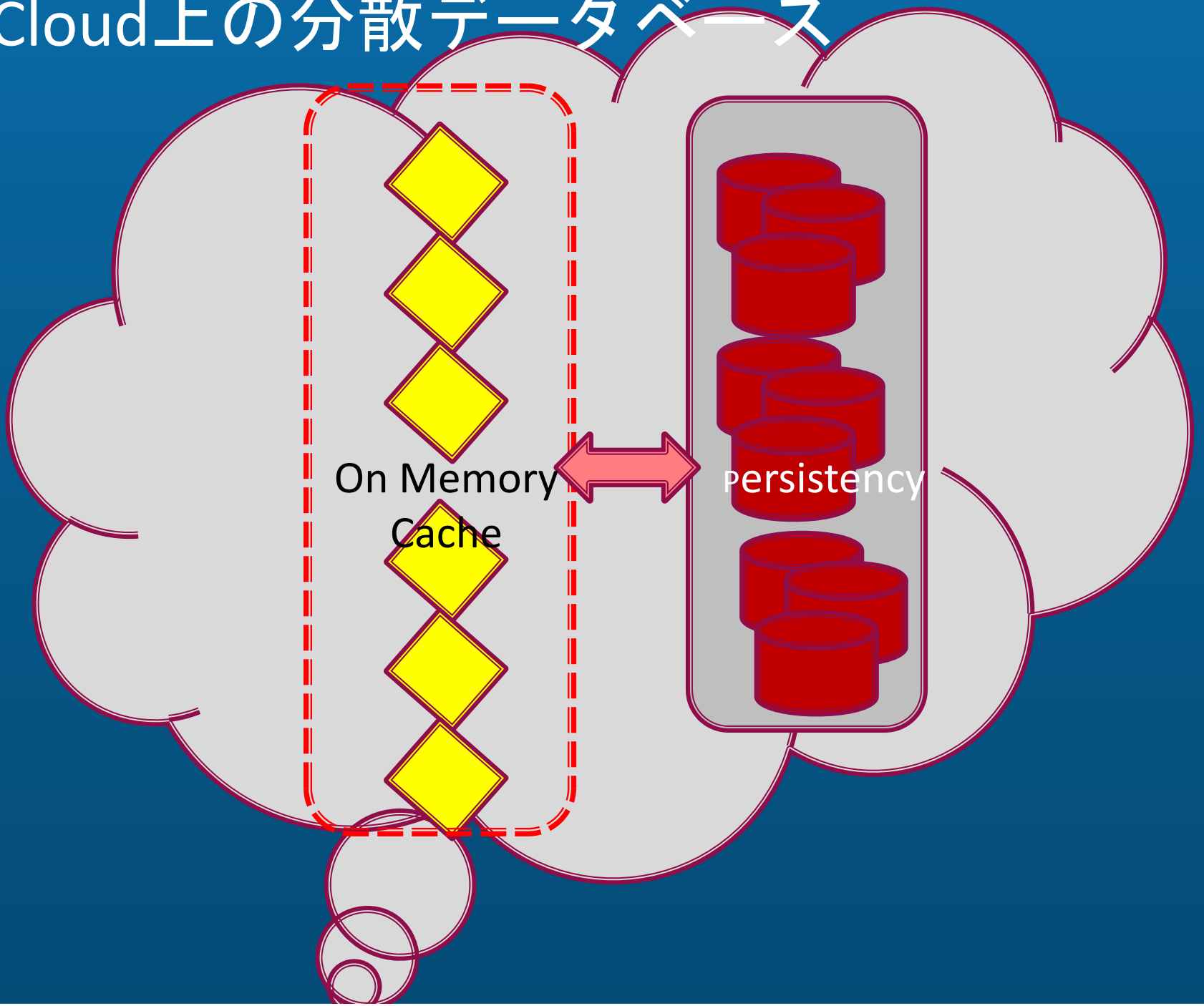


分散メモリーCache / Data Grid

- Oracle Coherence
  http://coherence.oracle.com/display/COH33UG/Coherence+3.3+Home
- IBM Object Grid
  http://www.ibm.com/developerworks/downloads/ws/wsdg/learn.html?S_TACT=105AGY09
- GiGaSpace
  http://www.gigaspaces.com/
- Terracotta
  http://terracottatech.com/

Key/Value の形式

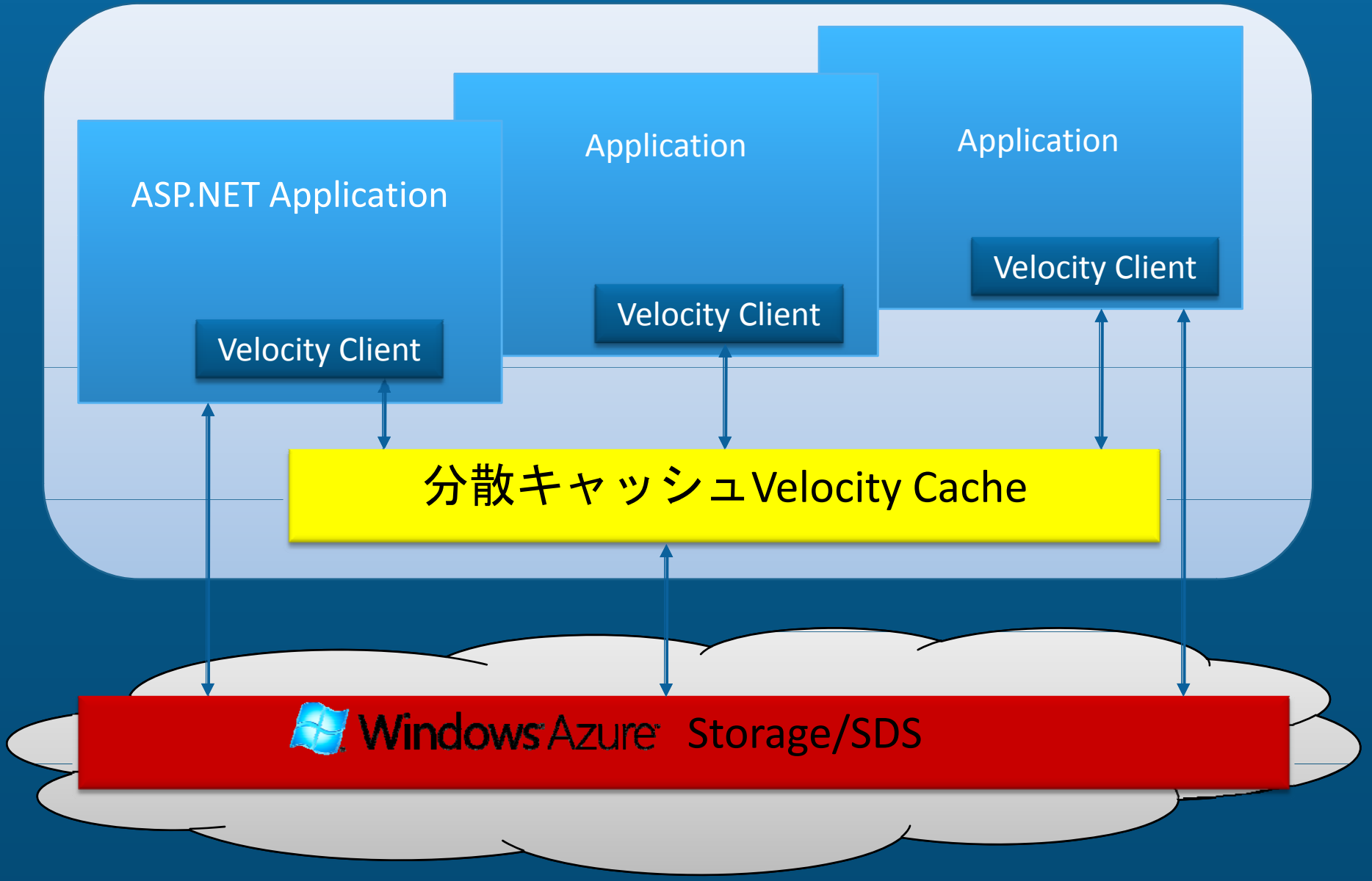
Cloud上の分散データベース



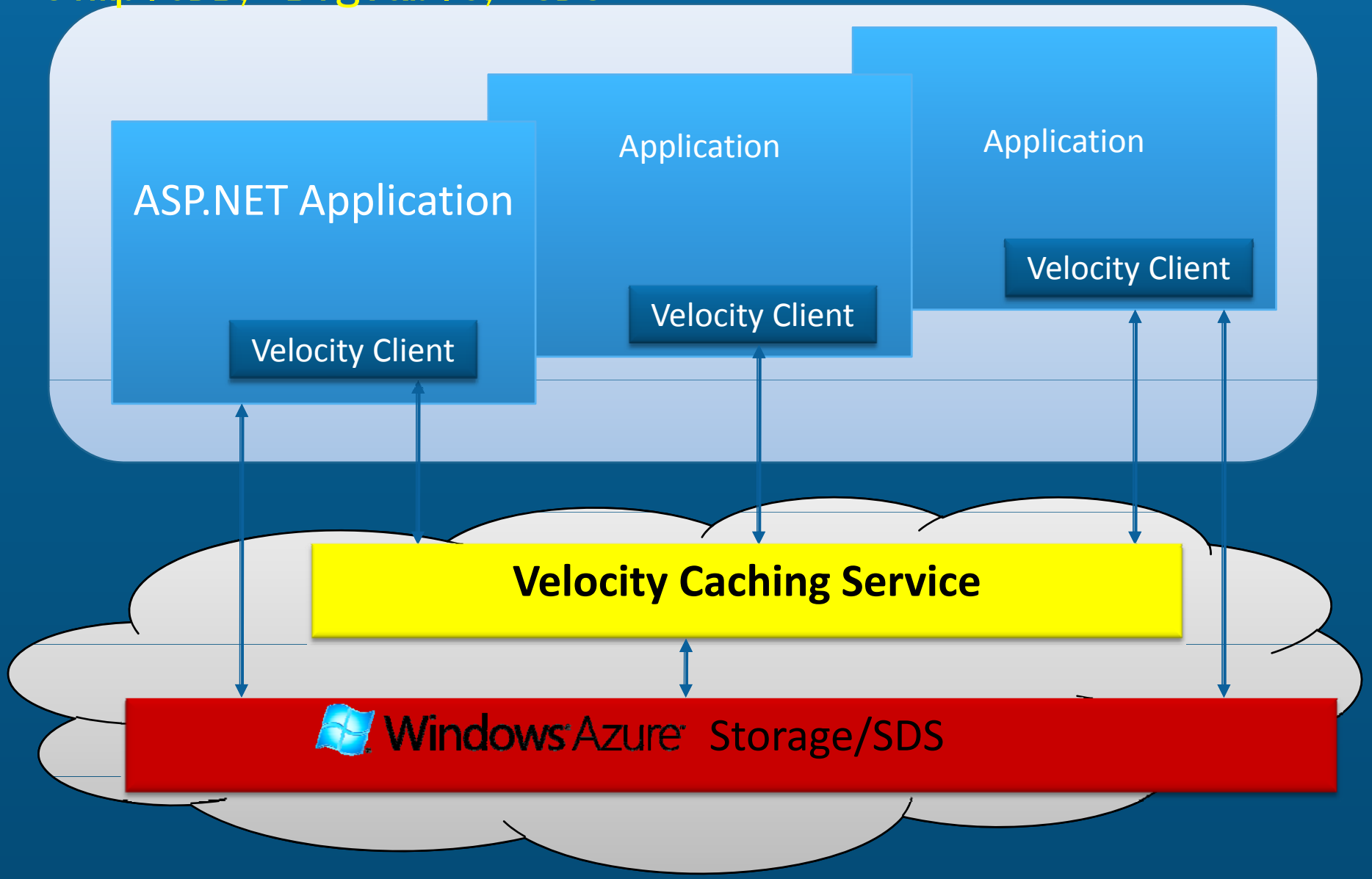
Cloud 上の分散データベース

- Google:　BigTable + Google App Engine
  http://labs.google.com/papers/bigtable-osdi06.pdf
- Microsoft: SSDS(SQL Server Data Service)
  http://www.microsoft.com/sql/dataservices/default.mspx
- Amazon: SimpleDB
  http://www.amazon.com/SimpleDB-AWS-Service-Pricing/b?ie=UTF8&node=342335011
- Amazon: Dynamo
  http://www.allthingsdistributed.com/2007/10/amazons_dynamo.html

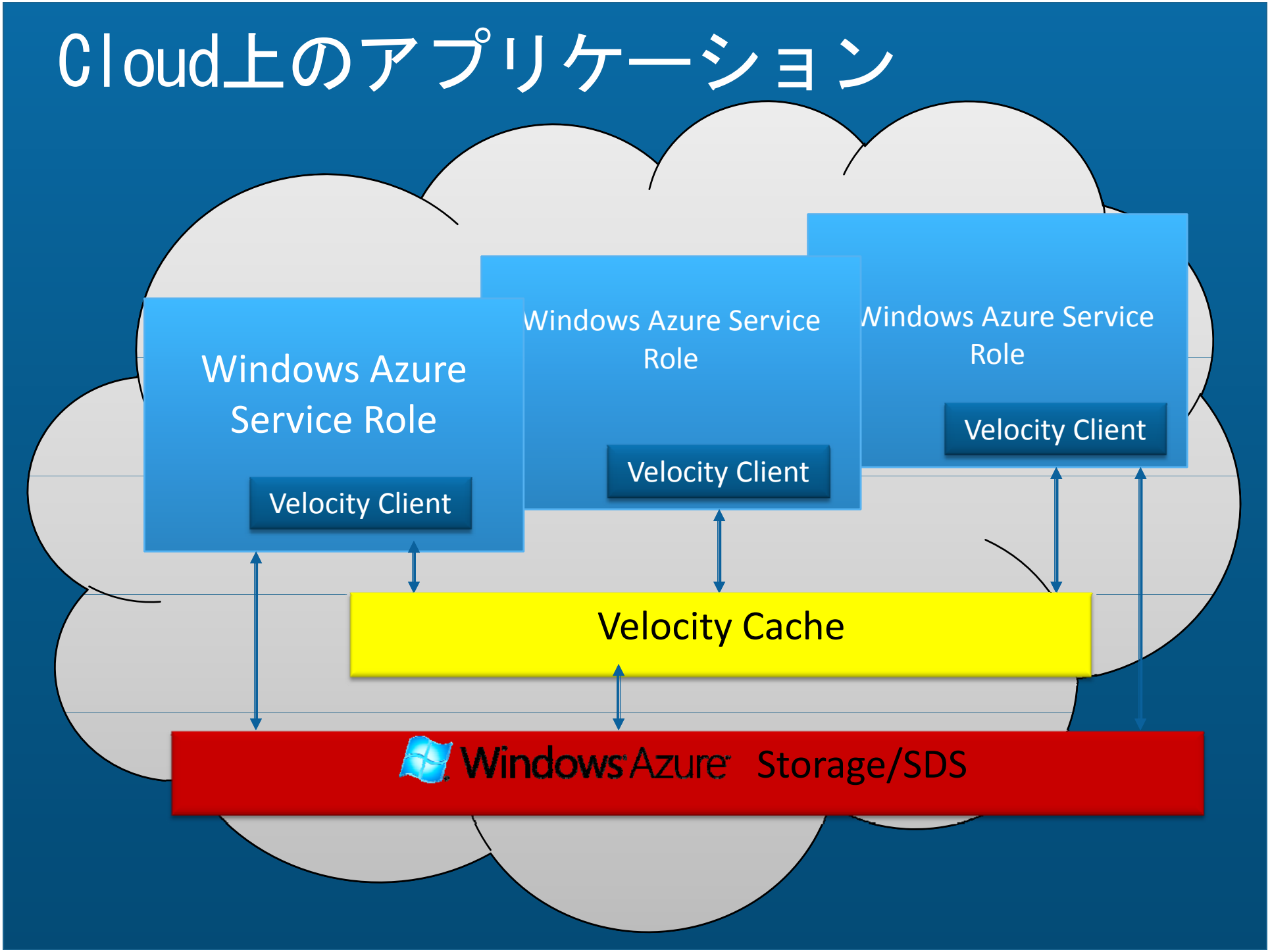
企業内での分散キャッシュの利用
Coherence, Object Grid, Velocityなど



Cloud上のデータサービスの利用
SimpleDB, BigTable, SDS



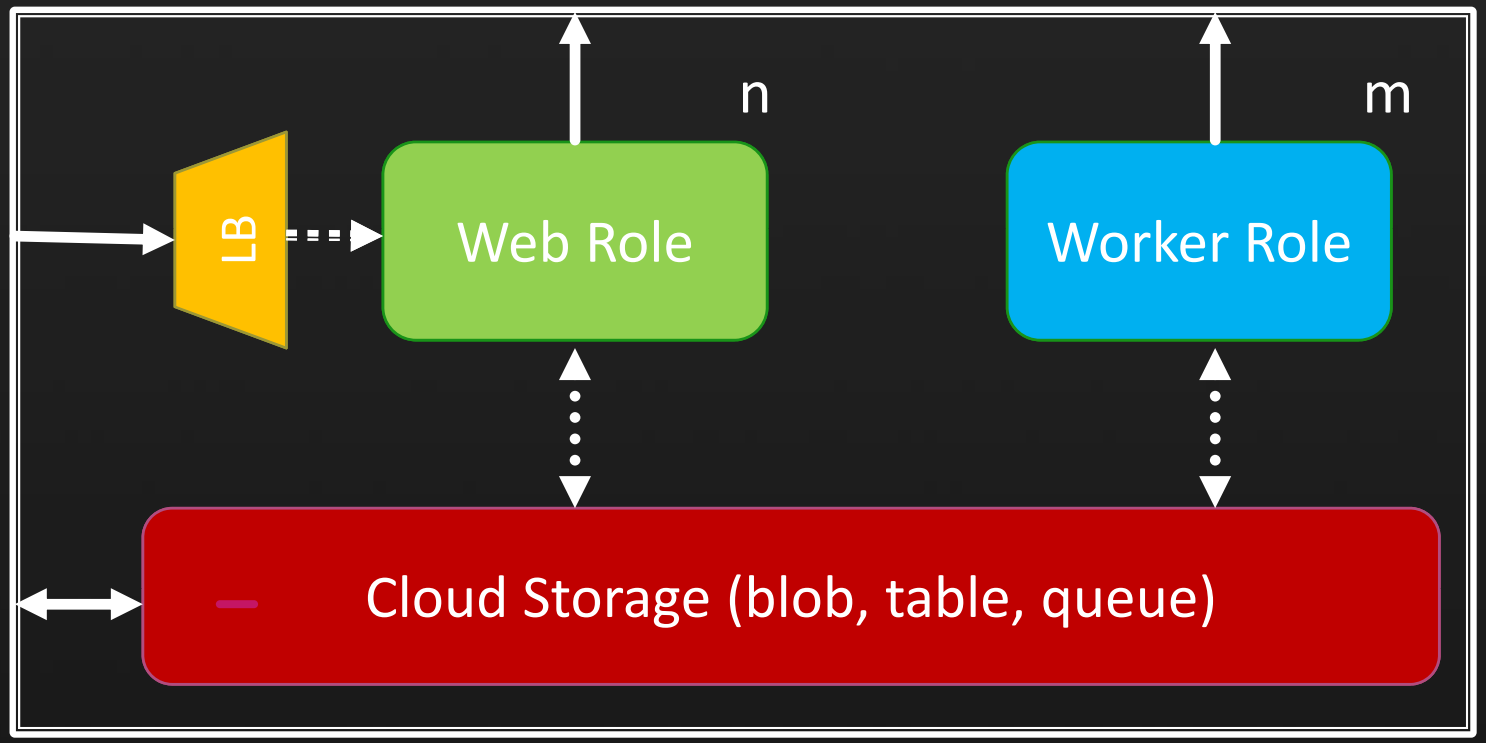
Cloud上のアプリケーション



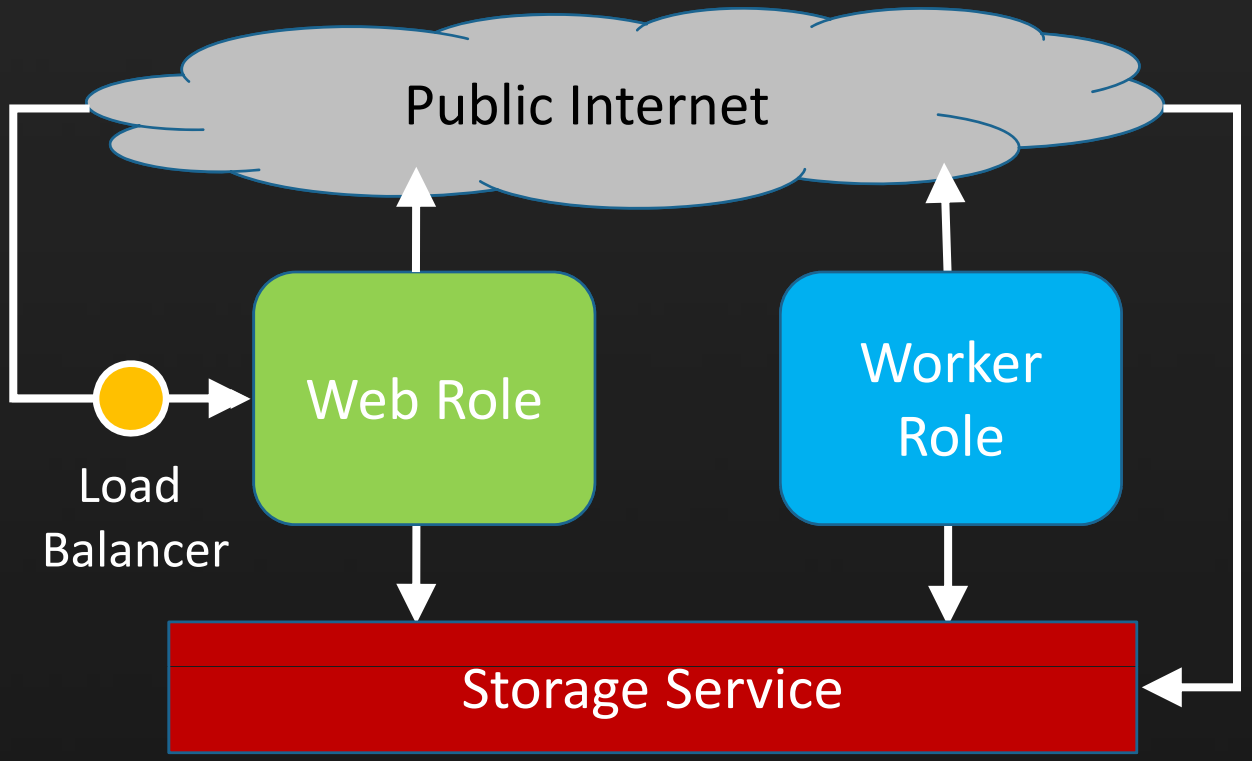
マルレク2008

Azure Webアプリケーションの Scale-out

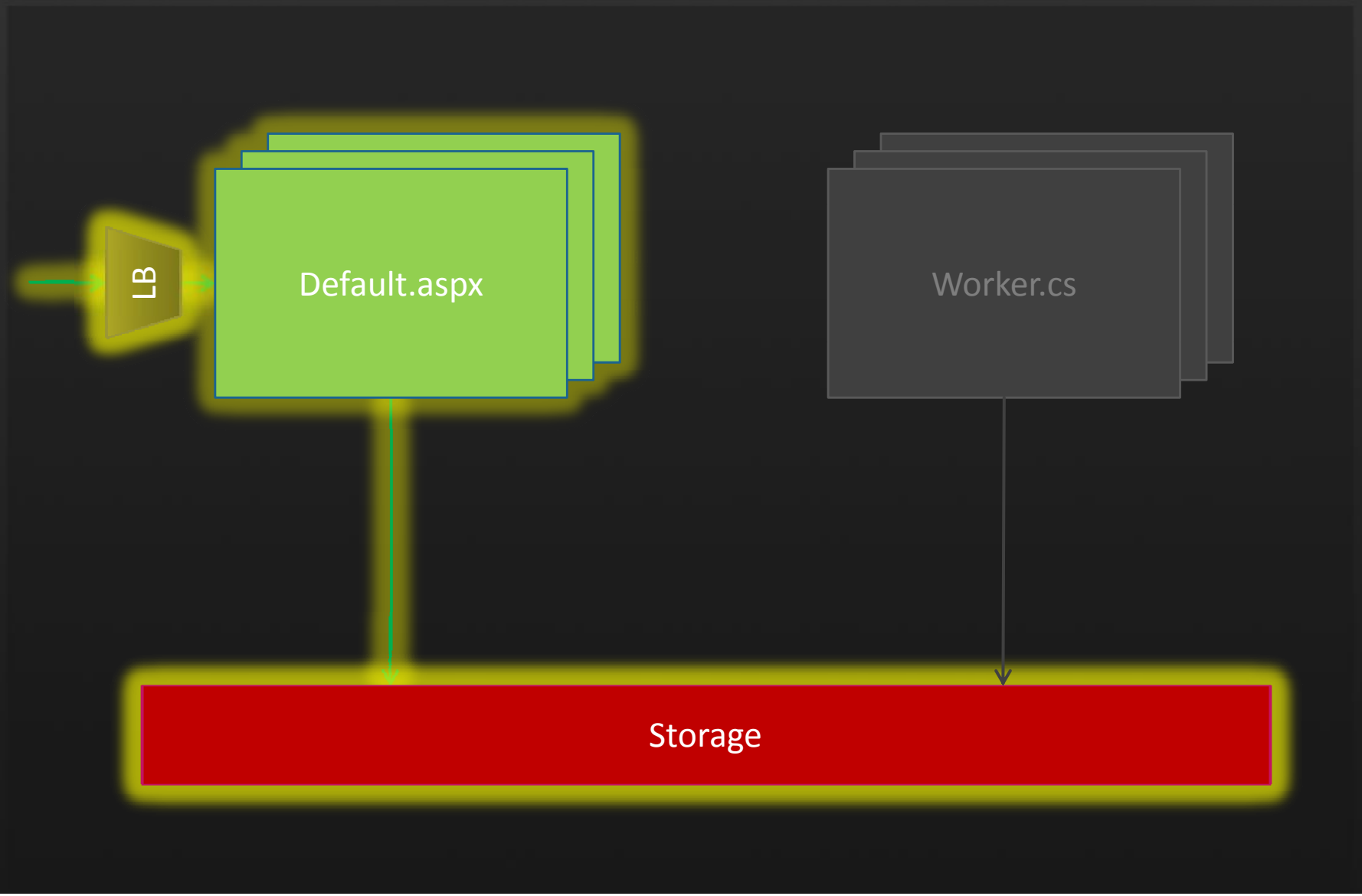
AzureでのWebアプリの構成



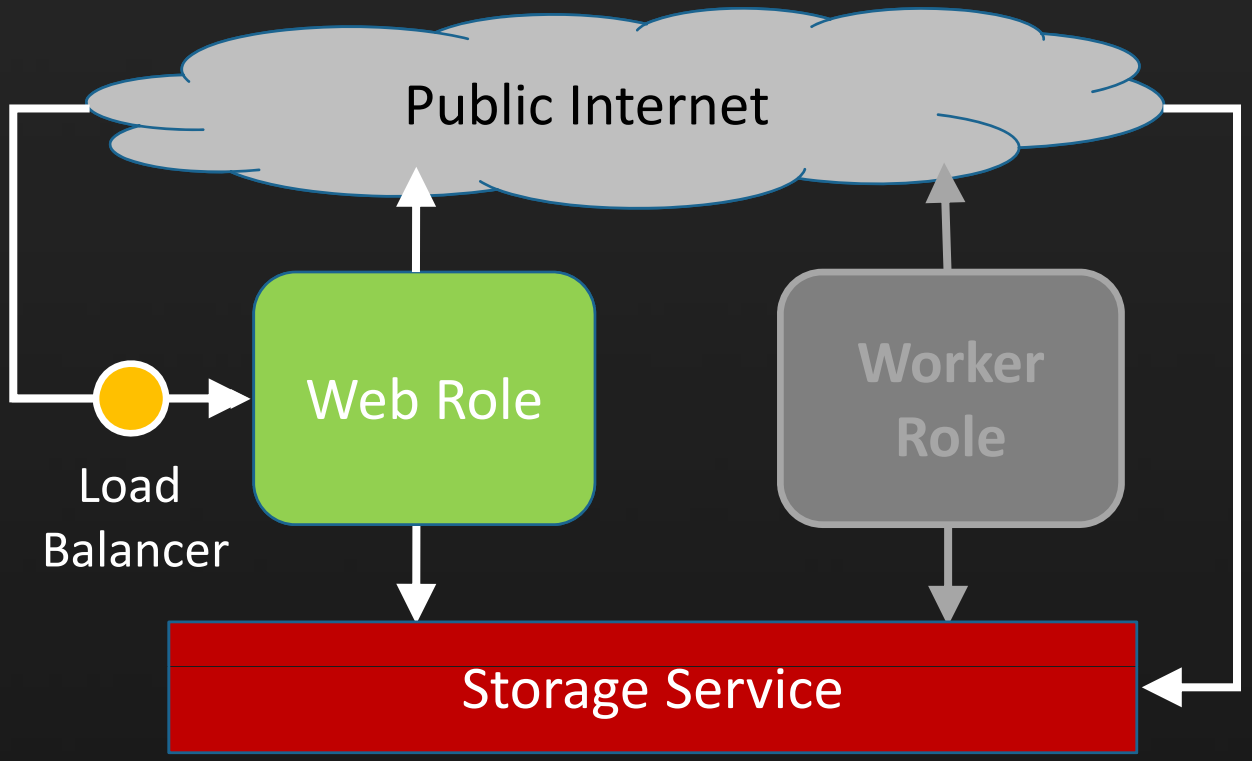
Azure とインターネットの関係



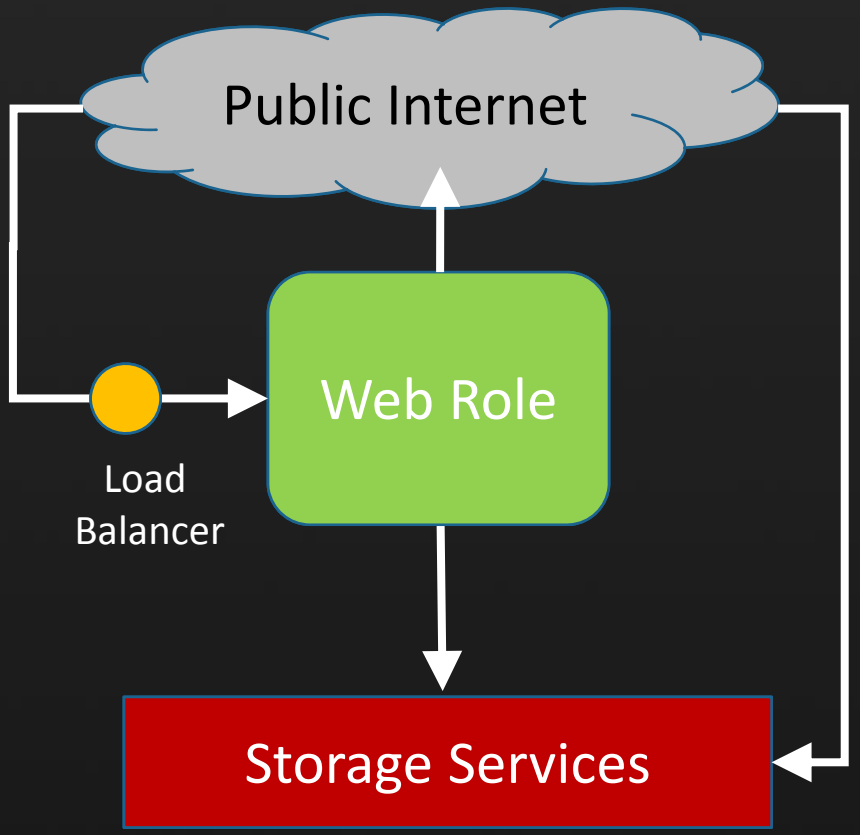
Service Architectures Web role



Web Role　Serving Dynamic Content

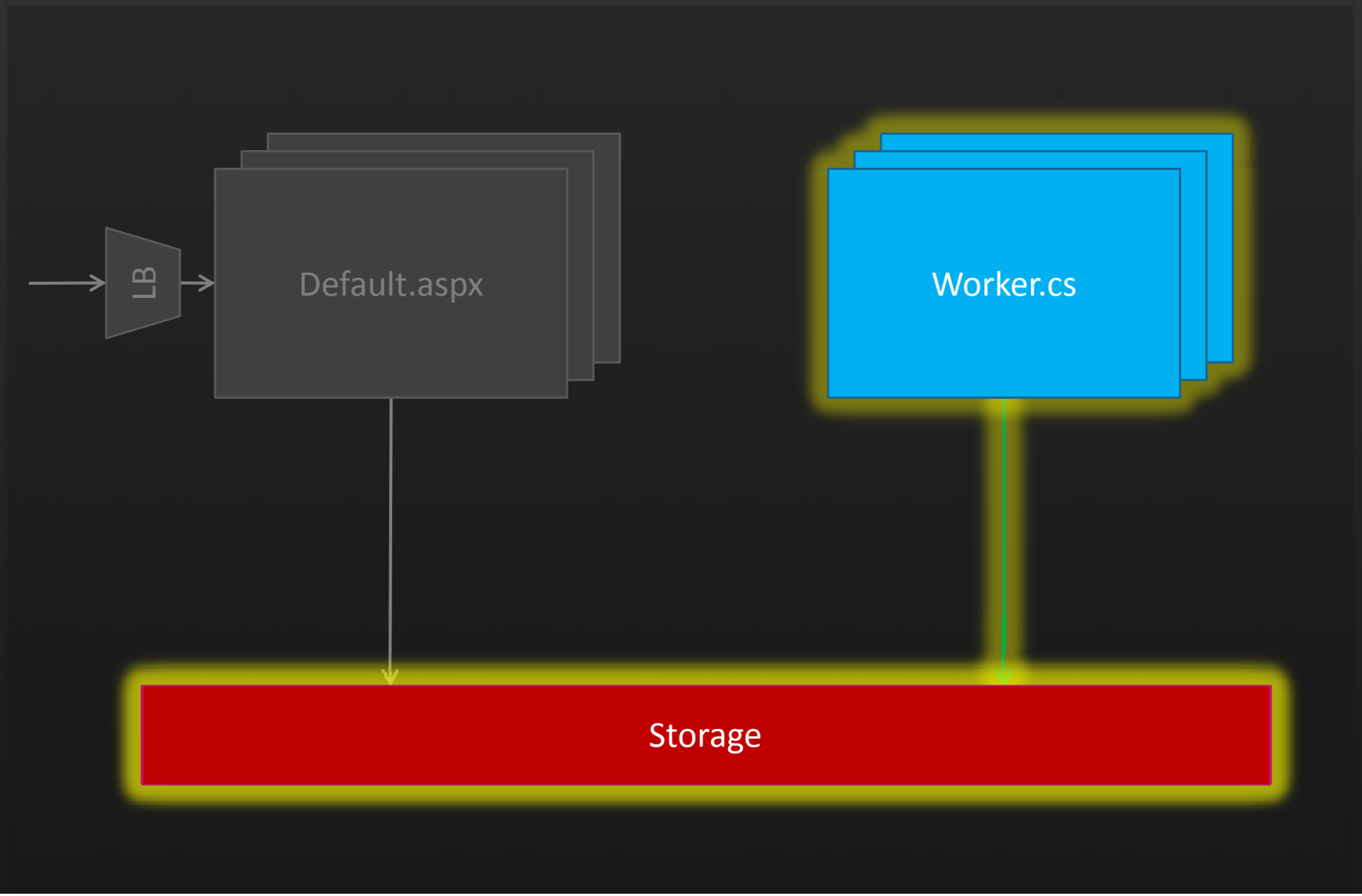


Web Role

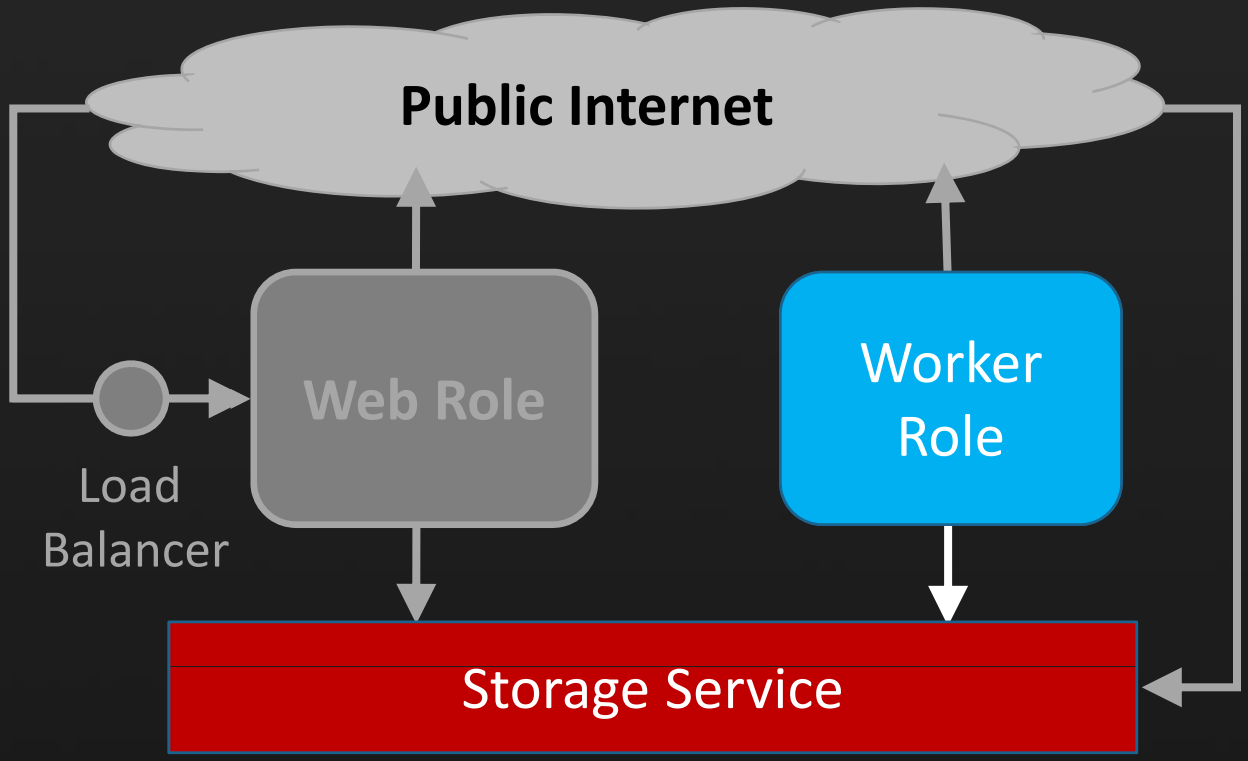


- インターネットからのリクエストを処理するWeb層
- IIS7 がホストする web core
  - ASP.NETをホスト
  - XML ベースの IIS7の設定
  - Integrated managed pipeline
  - SSLのサポート
  - 管理されたコードに対するWindows Azure code access security ポリシー (CAS)

Service Architectures Worker role

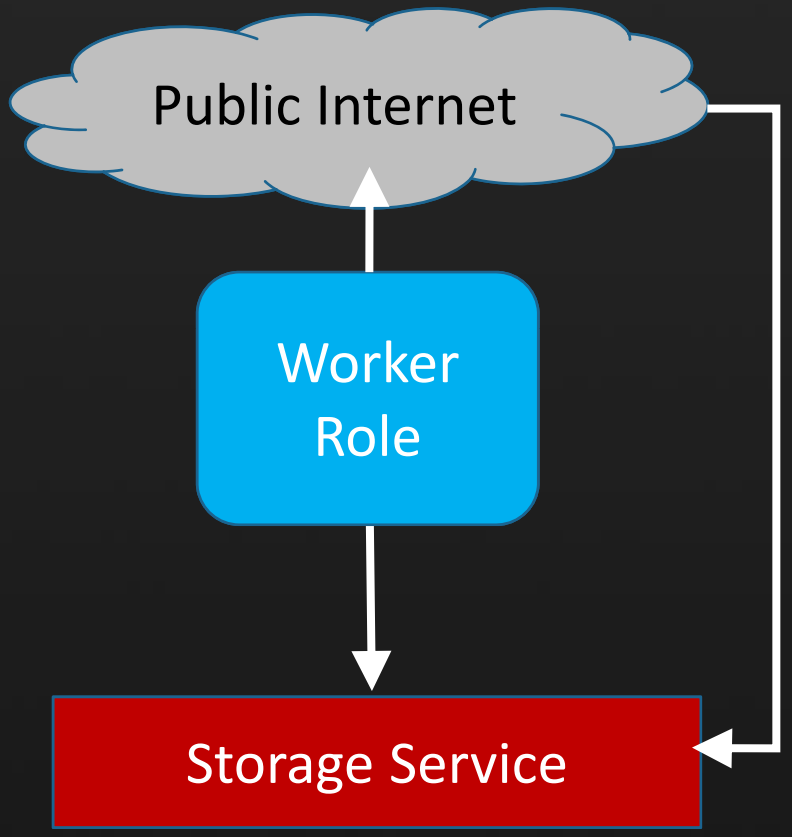


Worker Role Background Tasks

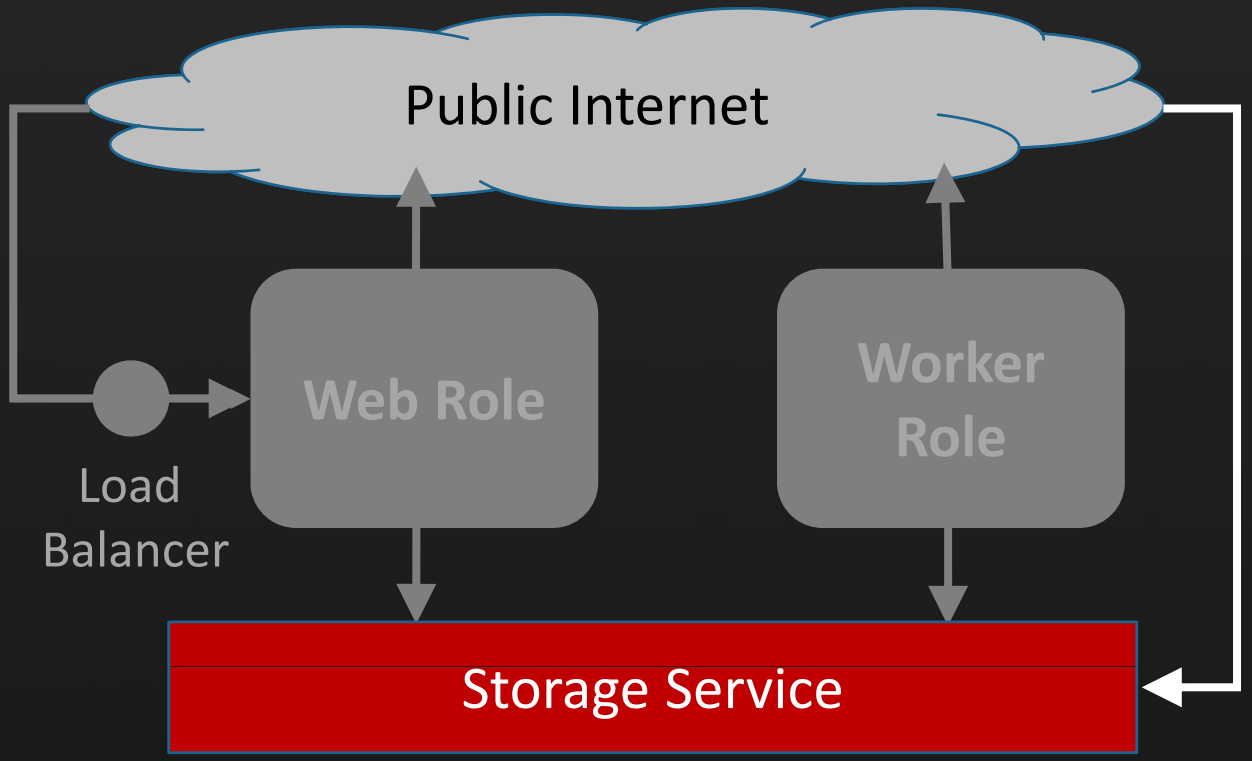


Worker Role

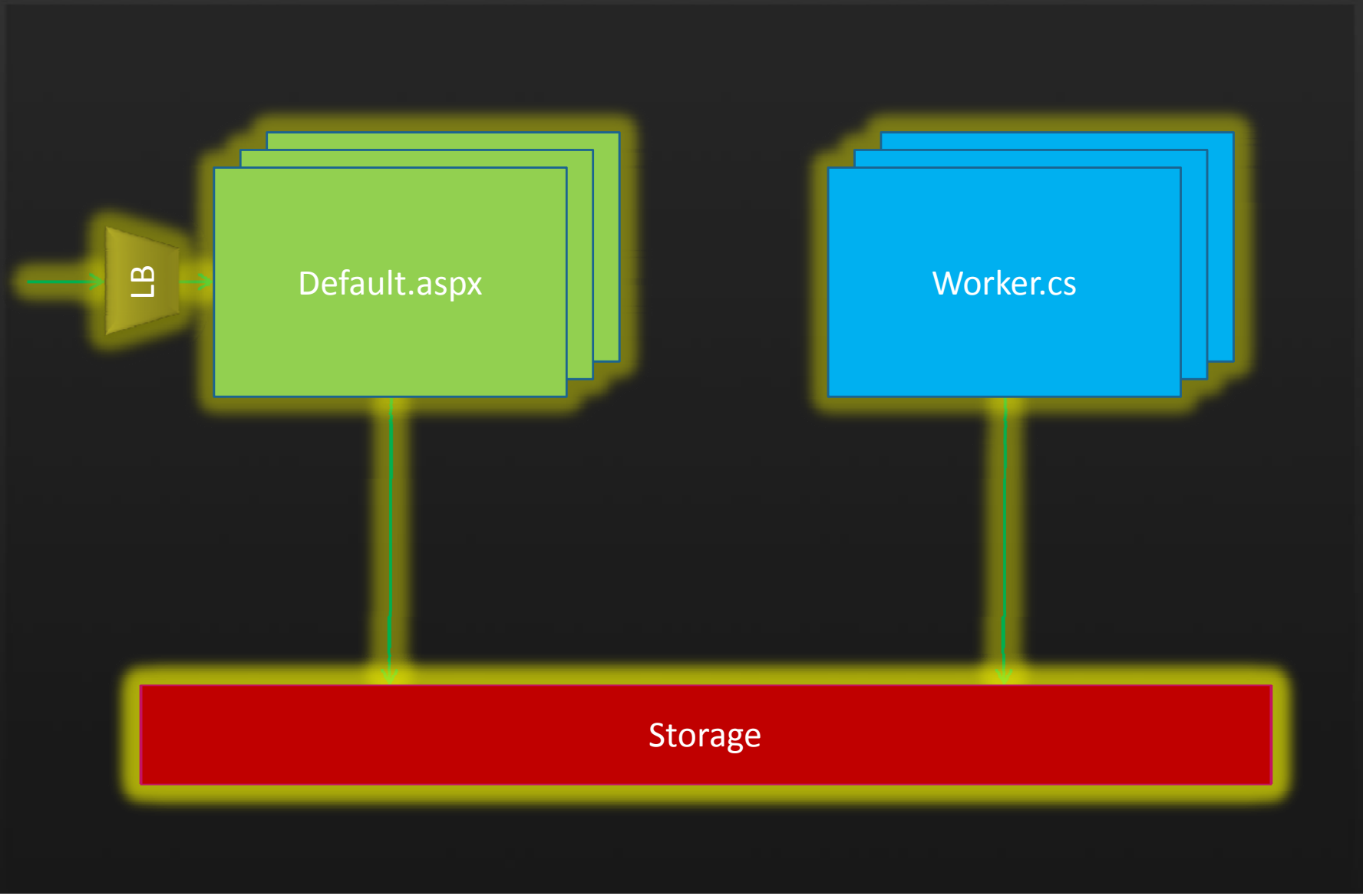
- Worker Roleには、内向きのネットワーク・コネクションはない
- ストレージのキューからリクエストを読むことはできる。
  - Windows Azure specific CAS policy for managed code



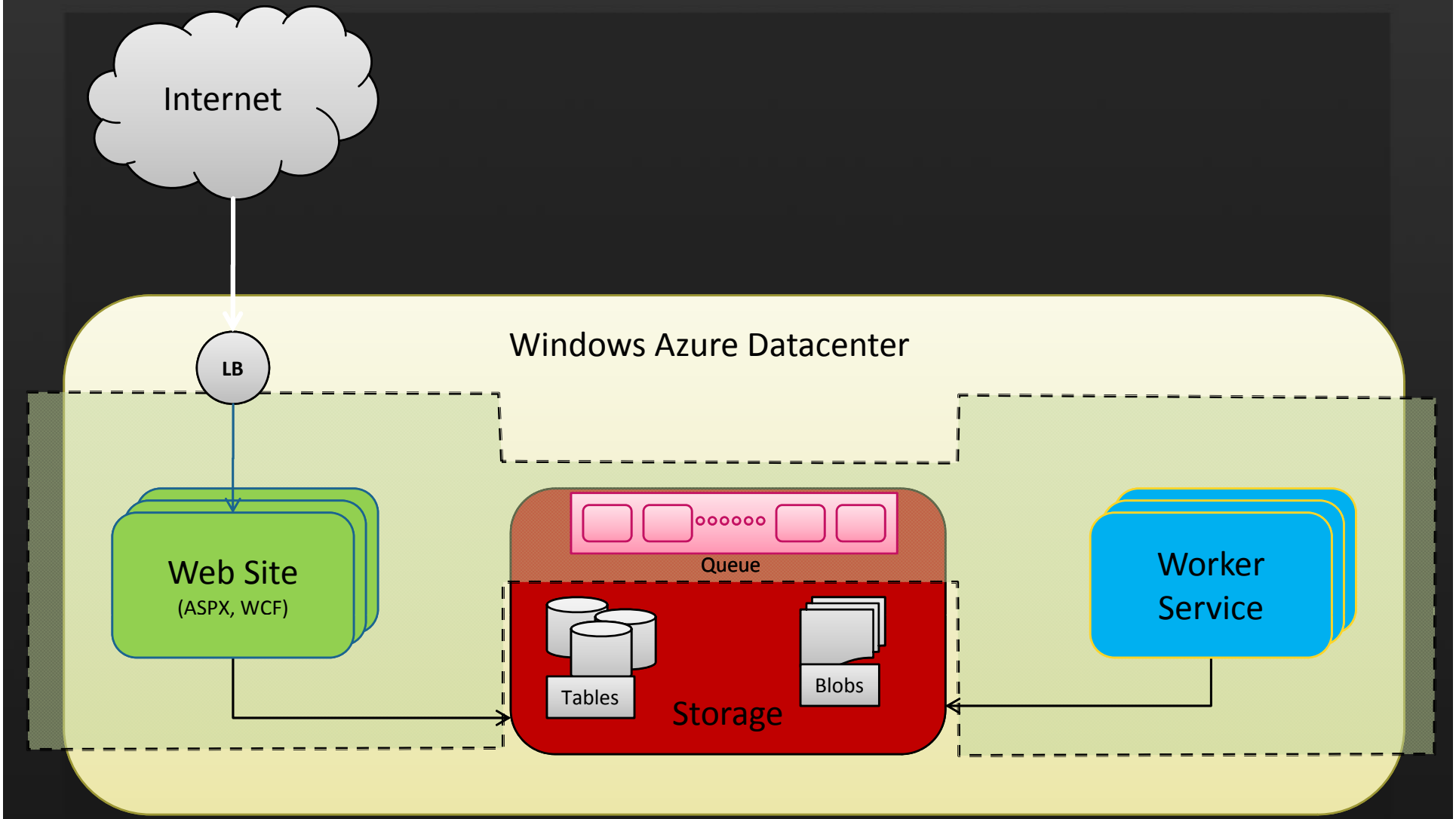
Uploading Static Content



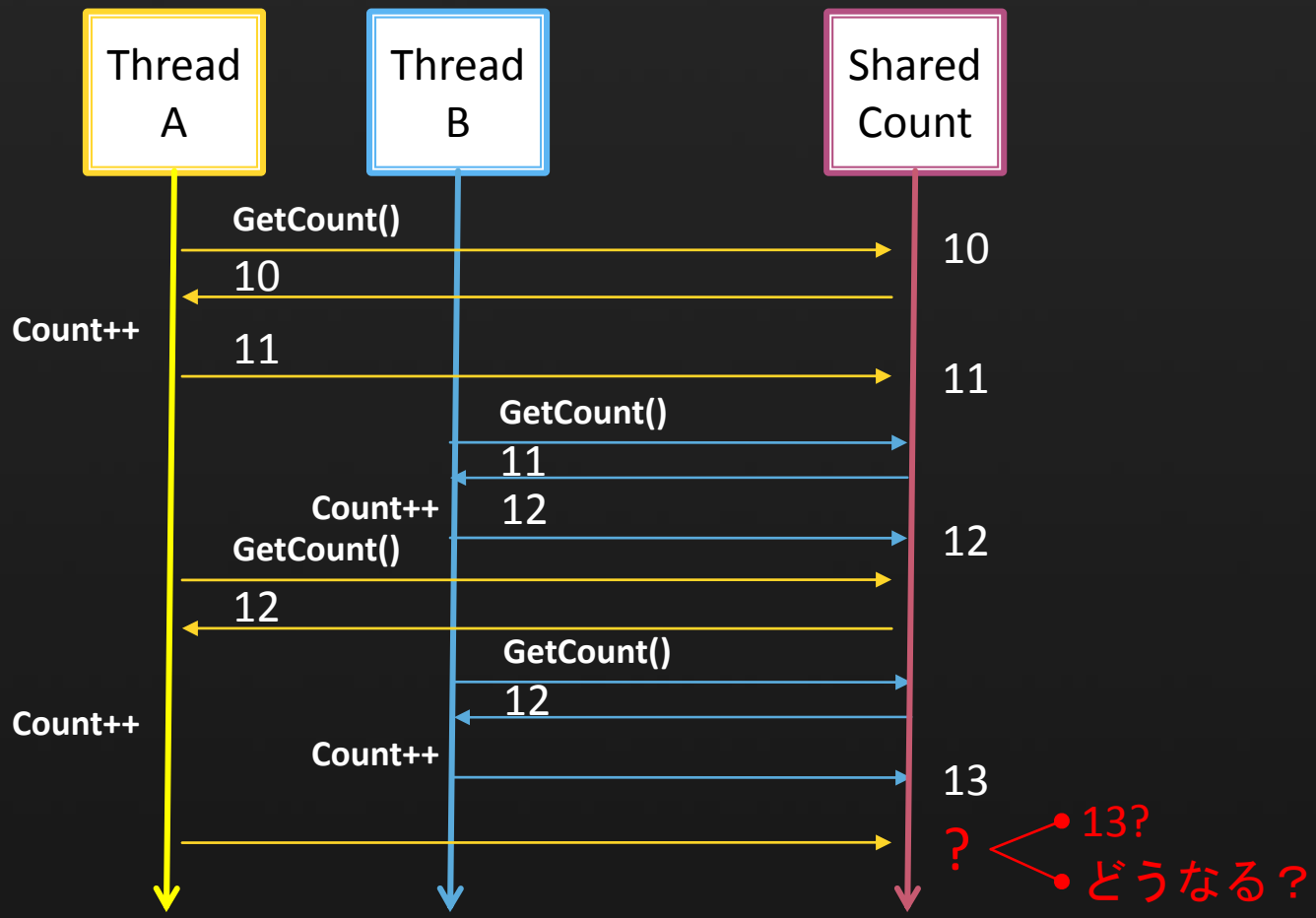
Service Architectures
Web and Worker roles



Service Architectures
Web Site + Worker

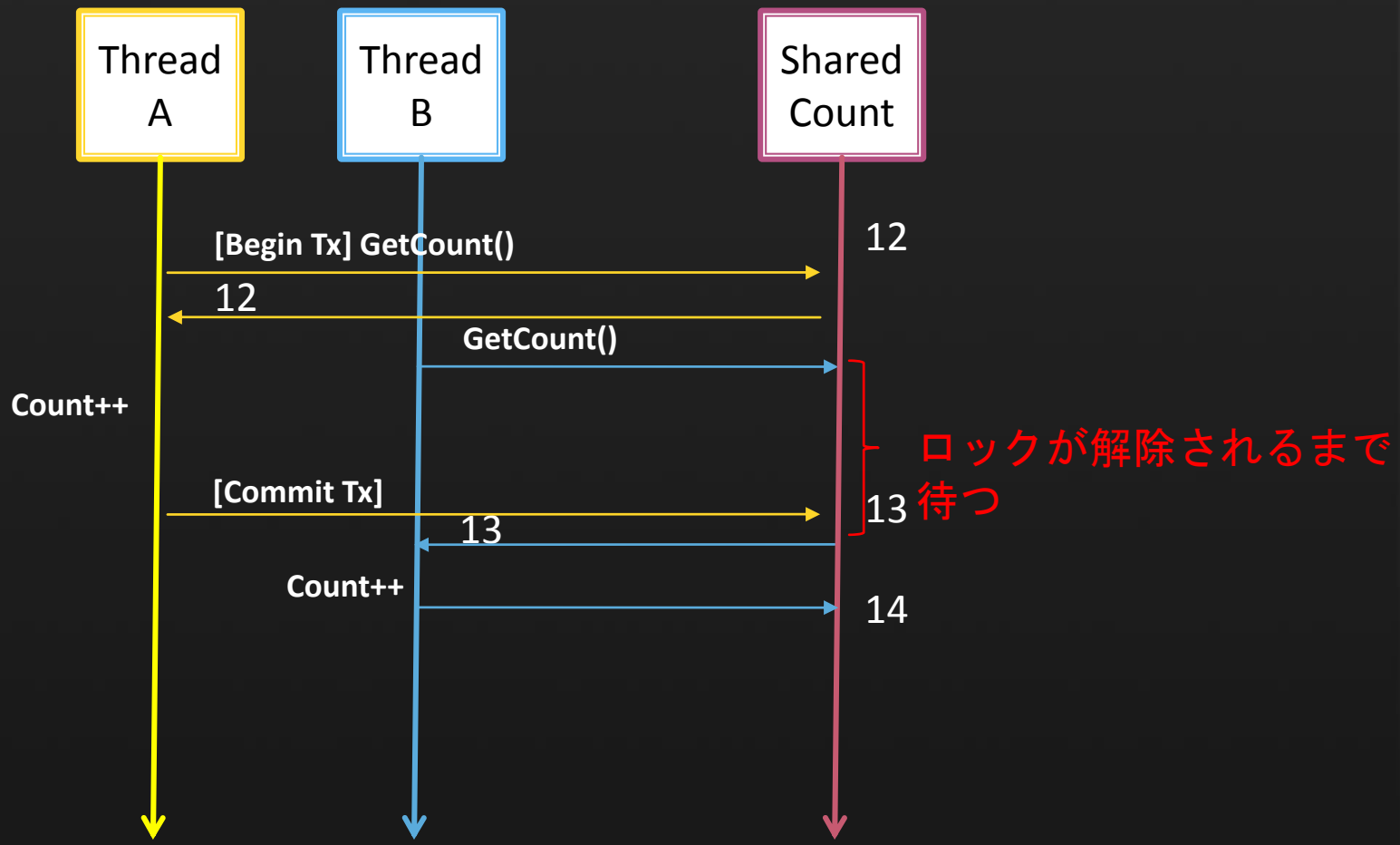


並行処理で共有領域にアクセスする際問題が起きる例



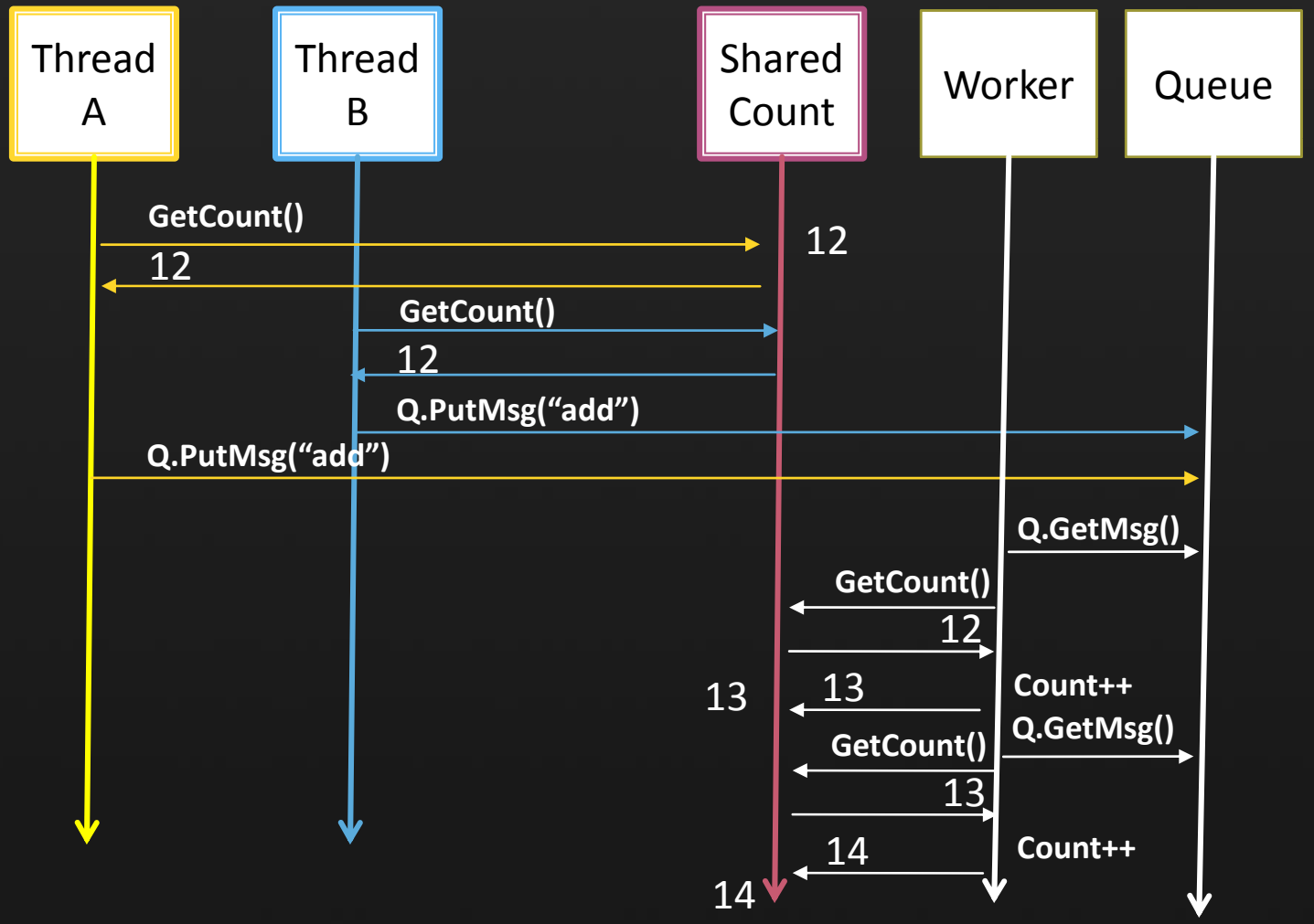
並行処理時の競合を防ぐには？

- 選択１：ロック



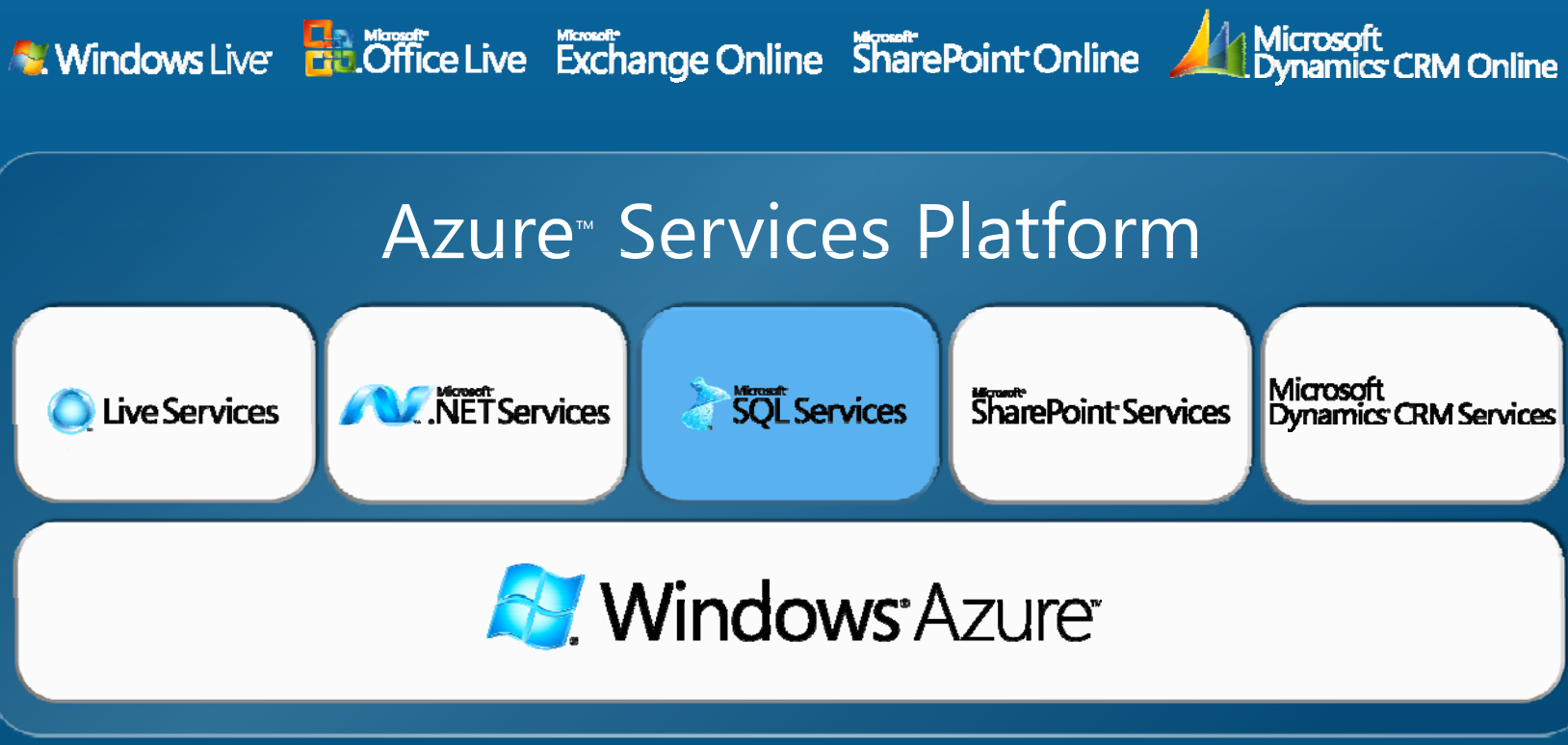
並行処理時の競合を防ぐには？

- 選択２：キューイング



マルレク2008

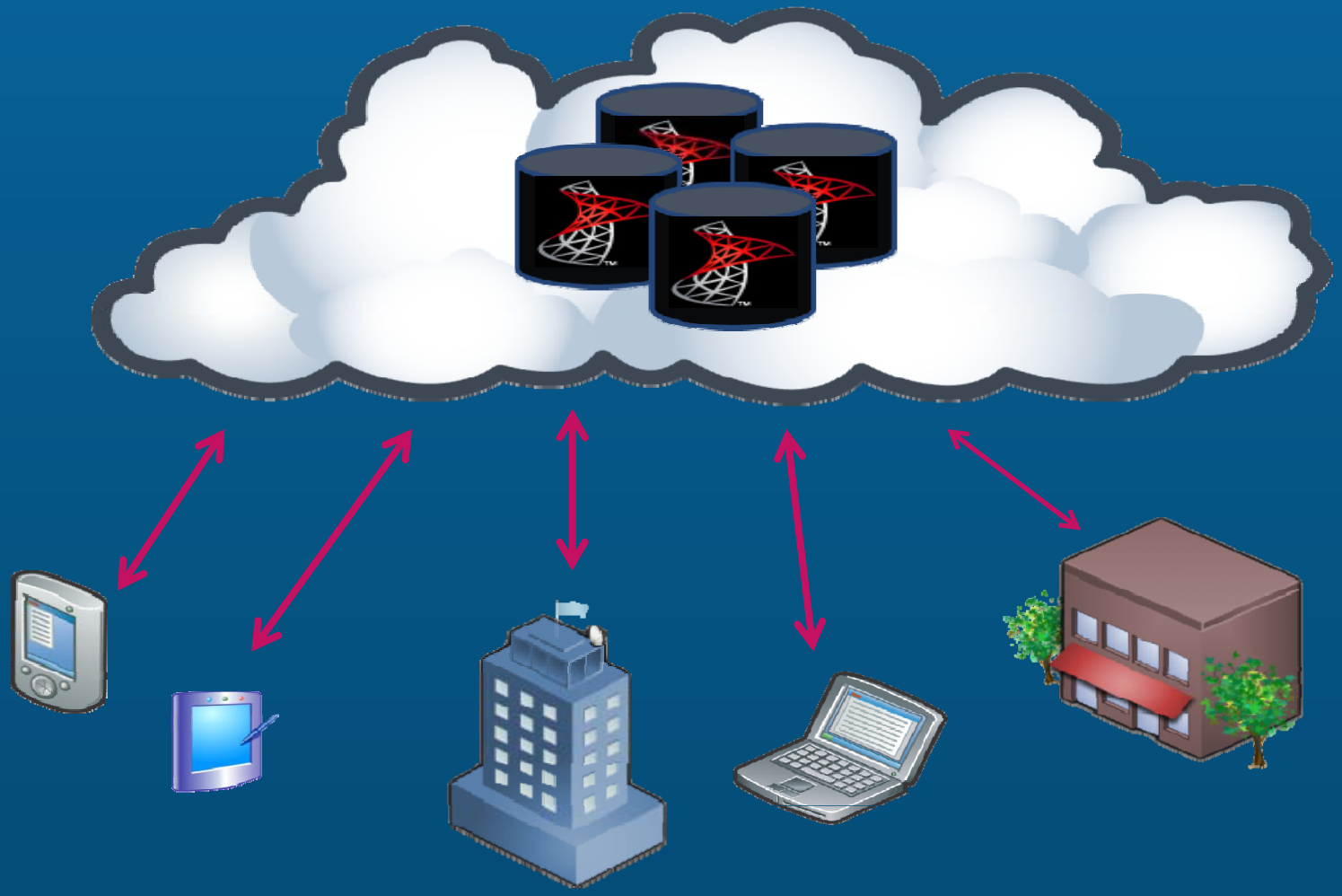
Windows Azure SDS
SQL Data Services


Windows Live
Office Live
Exchange Online
SharePoint Online
Microsoft Dynamics CRM Online
Azure™ Services Platform
Live Services
.NET Services
SQL Services
SharePoint Services
Microsoft Dynamics CRM Services
Windows Azure

SQL Services
SQL データプラットフォームを Cloudに拡大したもの

- Azure のデータサービス層
- データプラットフォームをCloudに拡大したもの
- 豊富なデータプラットフォームサービス

SDS (SQL Data Services)
Cloudの中のデータベース



SDS(SQL Data Services)
Cloudの中のデータベース

- インターネットベースの データベース・サービス
  - リレーショナルなクエリー処理
  - トランザクションの一貫性と並列処理をサポート
- サービスのオペレーションは、HTTPベースのWebサービス
  - 標準的なREST,SOAP に準拠
- 柔軟なデータ・モデル

SDS (SQL Data Services)
Cloudの中のデータベース

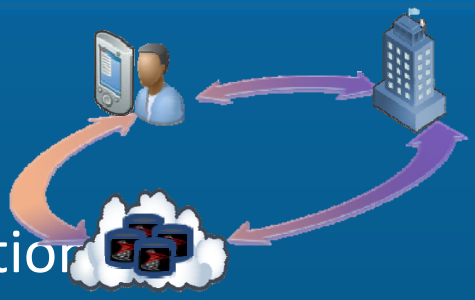
ビジネスレベルのサービス品質

- 高可用性と全面的な冗長性
- すぐにビジネスに利用できるレベルのSLA
- すぐれた操作性
- 柔軟な認証と権限付与
- 利用に応じた課金

SDS(SQL Data Services)
Cloudの中のデータベース

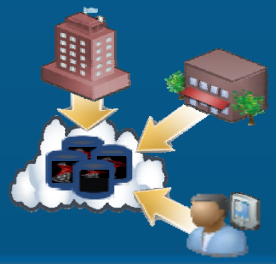
Cloudへの拡張が可能

- 既存のソリューションとの統合
- CloudにスケールしたData Platform solution



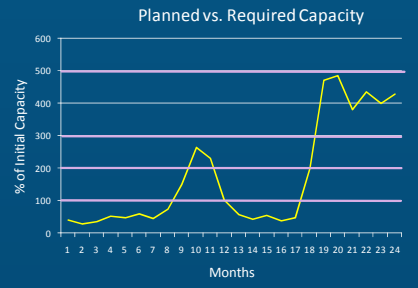
データ・ハブの創出

- セキュアなアクセスと構造化データの出し入れ
- デバイス、場所、顧客を越えた運用



利用、規模拡大に低い障壁

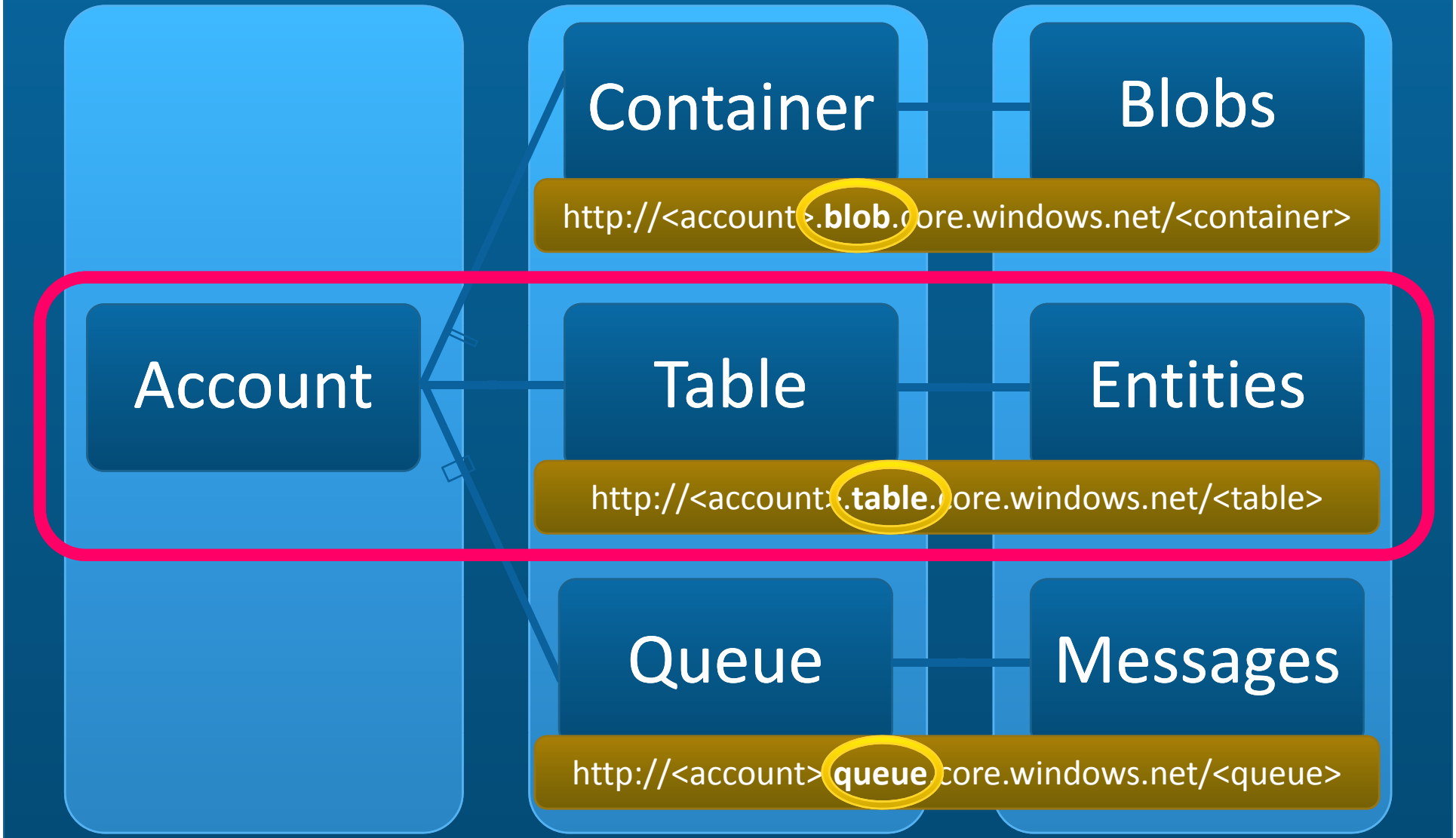
- 規模拡大に従って支払う
- ピーク時の要求に見合う施設を用意する必要がない



マルレク2008

Windows Azure SDSの
データモデルとテーブルの構造

Windows Azure のData Storage (Blob, Table, Queue)



Windows Azure のテーブル

- 構造化されたストレージを提供する
  - 非常に大規模にスケーラブルな テーブル
    - 数億行の エンティティ、テラバイト級のデータ
    - トラフィックが増大するにつれ、自動的に数千台のサーバにスケールする
  - 高可用性
    - 常にデータにアクセスできる
  - 耐久性
    - データは、最低でも3回コピーされる
- 親しみやすく簡単なプログラムインターフェース
  - ADO.NET Data Services – .NET 3.5 SP1
    - .NET クラスと LINQ
    - REST – どんなプラットフォーム、言語にも対応

テーブル・データ・モデル

- テーブル
  - ストレージのAccountがあれば、複数のテーブルを生成できる
  - テーブル名は、Accountで範囲づけられる
- データはテーブルに蓄えられる
  - テーブルはエンティティ（行）の集合
  - エンティティはプロパティ（カラム）の集合
- エンティティ
  - 二つの “key” プロパティが一緒になって、テーブル内のエンティティを一意に定める
    - PartitionKey – スケーラビリティを可能にする
    - RowKey – パーティション内のエンティティを一意に定める

エンティティとプロパティ

- それぞれのEntityは、255個までのプロパティを持つ
- 全てのエンティティにとって必須のプロパティ
  - Partition Key
  - Row Key
- 全てのエンティティは、システムが管理するversionをプロパティとして持つ
- その他のプロパティについては、決まったスキーマはない
  - それぞれのプロパティは、<名前, 型を持つ値>のペアとして蓄えられる
    - テーブルのスキーマ情報は、存在しない
  - 同じテーブル内の二つのエンティティは、違ったプロパティを持つことができる

サポートされているプロパティの型

- Partition Key と Row Key
  - String (64KBまで)
- プロパティの型
  - String (64KBまで)
  - Binary (64KBまで)
  - Bool
  - DateTime
  - GUID
  - Int
  - Int64
  - Double

SDSのエンティティの特徴

エンティティのプロパティの型は、エンティティごとに違っていても構わない

Property		Type	Value
Metadata	ID	EntityId	VWGOLF-01
	Kind	EntityKind	
FlexProps	Description	String	Reliable, one owner, …
	Price	Numeric	12000.00
	ListingDate		01-01-2008
	LocationZip	String	98052

Property		Type	Value
Metadata	ID	EntityId	MINICOOPER-264
	Kind	EntityKind	
FlexProps	Description	String	Reliable, one owner, …
	Price	Numeric	12000.00
	ListingDate		1st January, 2008
	LocationZip	String	98052

異なった Kinds

異なった Instance Types

プロパティの追加

Relational DBのテーブルとは異なった仕方でのData Modeling

Relational Tables

CId	Conf Title
1	PDC
2	Tech Ready

TId	CId	Track Title
1	1	Cloud Compute
1	2	SQL Server 2008

SId	CId	TId	Session Subject
1	1	1	Live Meeting
1	2	1	SQL Server FILESTREAM

Windows Azure Table

Partitioning Key　Row Key

Conf Id	Track Id	Session Id	Conf Title	Track Subject	Session Subject
1	Null	Null	PDC	Null	Null
1	1	Null	Null	Cloud Compute	Null
1	1	1	Null	Null	Live Meeting
2	Null	Null	Tech Ready	Null	Null
2	1	Null	Null	SQL Server 2008	Null
2	2	1	Null	Null	SQL Server FILESTREAM

Rowgroup

Rowgroup

Primary Key

マルレク2008

Windows Azure SDSの Partition

Partition Key とPartitions

- 全てのテーブルは、Partition Keyを持つ
  - テーブルの第一カラム
  - テーブルをPartitionに分割するのに利用される
- テーブルPartition
  - あるテーブル内で、同じPartitionKeyの値を持つ全てのエンティティ
- Partition Keyは、プログラムから操作可能
  - アプリケーションが、Partitionの粒度をコントロールして、スケーラビリティを持つことを可能とする

Partition の例

Partition Key Document Name	Row Key Version	Property 3 Modification Time	.....	Property N Description	
Examples Doc	V1.0	8/2/2007	.....	Committed version	Partition 1
Examples Doc	V2.0.1	9/28/2007		Alice's working version	
FAQ Doc	V1.0	5/2/2007		Committed version	
FAQ Doc	V1.0.1	7/6/2007		Alice's working version	Partition 2
FAQ Doc	V1.0.2	8/1/2007		Sally's working version	

- Partition : あるテーブル内で、同じ PartitionKeyの値を持つ全てのエンティティ
- アプリケーションは、Partitionの粒度をコントロールできる

Partitionの目的

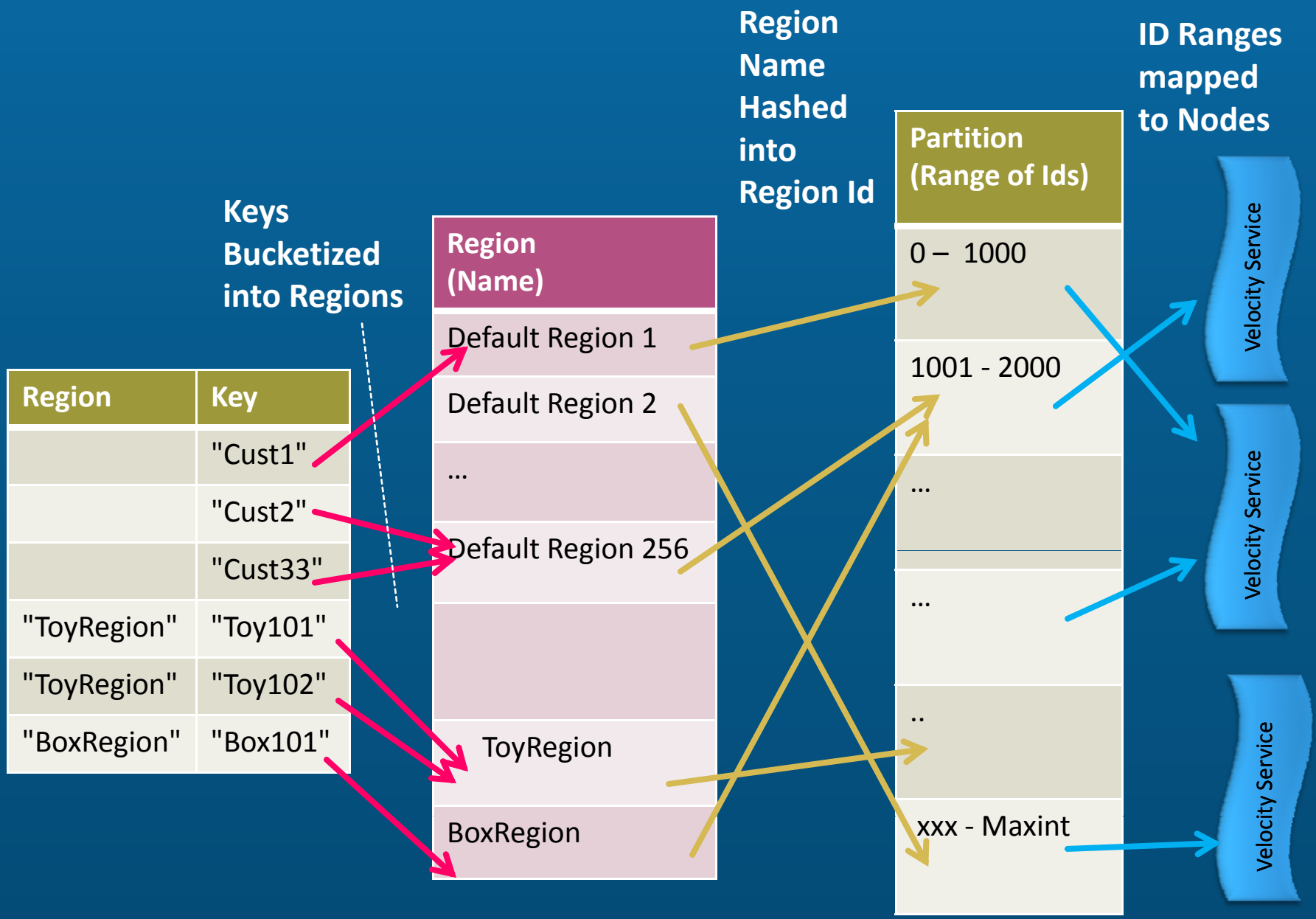
- パフォーマンスとエンティティの局所性
  - 同じPartition内のエンティティは、一緒に格納される
    - 効率的な検索とキャッシュの局所性
- テーブルのスケーラビリティ
  - Partitionの利用パターンをモニタし、
  - 自動的にPartitionのロードバランスを行う
    - それぞれのPartitionは、異なるストレージノードに格納されることができる
    - テーブルのトラフィックの必要に見合うようにスケールできる

パフォーマンスとスケーラビリティ

Partition Key Document Name	Row Key Version	Property 3 Modification Time	.....	Property N Description	
Examples Doc	V1.0	8/2/2007	.....	Committed version	Partition 1
Examples Doc	V2.0.1	9/28/2007		Alice's working version	
FAQ Doc	V1.0	5/2/2007		Committed version	Partition 2
FAQ Doc	V1.0.1	7/6/2007		Alice's working version	
FAQ Doc	V1.0.2	8/1/2007		Sally's working version	

- FAQ DOCの全てのVersionを取得するのは効率的に出来る。なぜなら、一つのPartitionにアクセスすればよいから
- アクセスをスケールアウトするために、二つのPartitionを、異なるサーバに置くことができる

Key Mapping


Keys Bucketized into Regions
Region Name Hashed into Region Id
ID Ranges mapped to Nodes
Region
Key
"Cust1"
"Cust2"
"Cust33"
"ToyRegion"
"Toy101"
"ToyRegion"
"Toy102"
"BoxRegion"
"Box101"
Region (Name)
Default Region 1
Default Region 2
...
Default Region 256
ToyRegion
BoxRegion
Partition (Range of Ids)
0 – 1000
1001 - 2000
...
...
..
xxx - Maxint
Velocity Service
Velocity Service
Velocity Service

Partition Keyをどう選ぶか

- 検索でよくつかわれるものをPartition Keyとして使う
  - Partition Key が Queryの一部であれば
    - 一つのPartition内のエンティティを取得するのに、高速にアクセスできる
  - もし、Partition KeyがQueryで指定されないと
    - その時には、すべてのPartitionがスキャンされなければいけない

PartionKeyの有無と検索の効率

Partition Key Document Name	Row Key Version	Property 3 Modification Time	.....	Property N Description	
Examples Doc	V1.0	8/2/2007	.....	Committed version	Partition 1
Examples Doc	V2.0.1	9/28/2007		Alice's working version	
FAQ Doc	V1.0	5/2/2007		Committed version	Partition 2
FAQ Doc	V1.0.1	7/6/2007		Alice's working version	
FAQ Doc	V1.0.2	8/1/2007		Sally's working version	

- (PartitionKey ==“FAQ Doc”)である全てのエンティティの取得は高速である。一つのPartitionにアクセスすればいいから
- (ModifiedTime < 6/01/2007)なる全てのDocを取得するのは、高価な処理である。全てのPartitionを横断する必要あり

Partition Keyをどう選ぶか

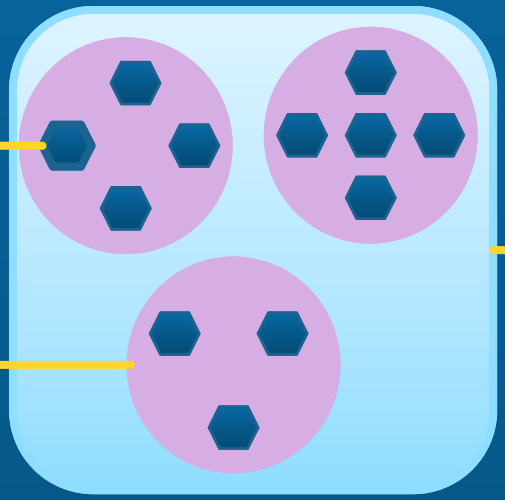
- 検索でよくつかわれるものをPartition Keyとして使う
  - Partition Key が Queryの一部であれば
    - 一つのPartition内のエンティティを取得するのに、高速にアクセスできる
  - もし、Partition KeyがQueryで指定されないと
    - その時には、すべてのPartitionがスキャンされなければいけない
- 負荷をPartition間で分散する
  - Partition Key は、テーブルPartitionにアクセスするものをコントロールすることを可能にする
  - 沢山Partitionがあればある程、負荷のバランスを自動的にとるのが容易になる
    - エンティティの局所性とはトレードオフになる

エンティティのPrimary Key

Partition Key Document Name	Row Key Version	Property 3 Modification Time	.....	Property N Description	
Examples Doc	V1.0	8/2/2007	.....	Committed version	Partition 1
Examples Doc	V2.0.1	9/28/2007		Alice's working version	
FAQ Doc	V1.0	5/2/2007		Committed version	
FAQ Doc	V1.0.1	7/6/2007		Alice's working version	Partition 2
FAQ Doc	V1.0.2	8/1/2007		Sally's working version	

- エンティティのPrimary Keyは、次の二つの組合せ
  - PartitionKey
    - エンティティが属するテーブルのPartitionの一意なID
  - RowKey
    - Partition内のエンティティの一意なID

データの分割にかかわる三つのUnit



Storage Unit

- CRUD 操作をサポート
- 例） DB row, SDS entity

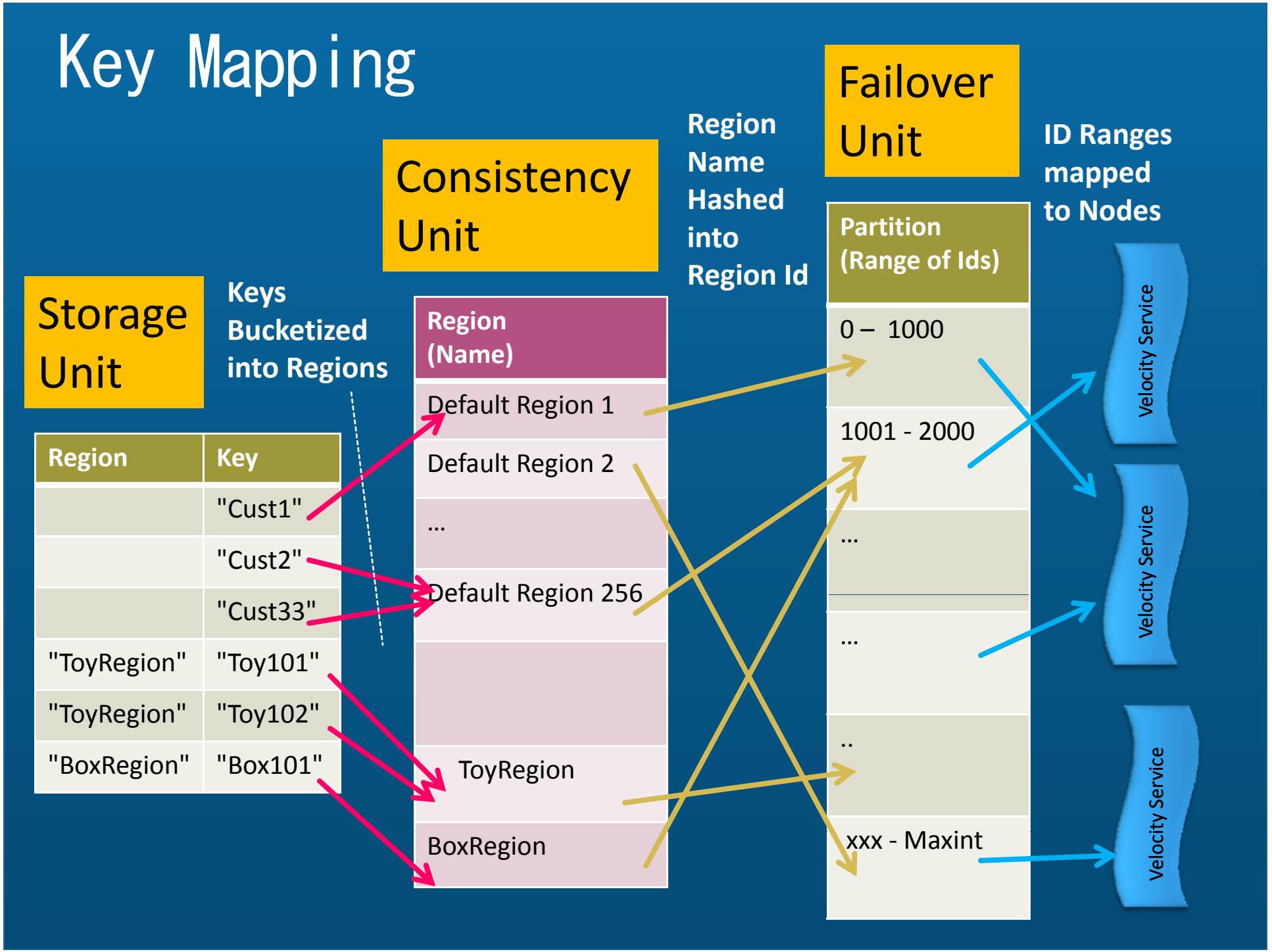
Consistency Unit

- Storage unitsの集合
- アプリで指定可能
- Hashされたり、Partition Keyで、範囲分割される
- 例） Row group, Database, SDS Container

Failover Unit (a.k.a. Partition)

- 管理の単位
- Consistency unitsのグループ
- システムによって決定される
- Consistency Unitの境界でマージされたり分割される

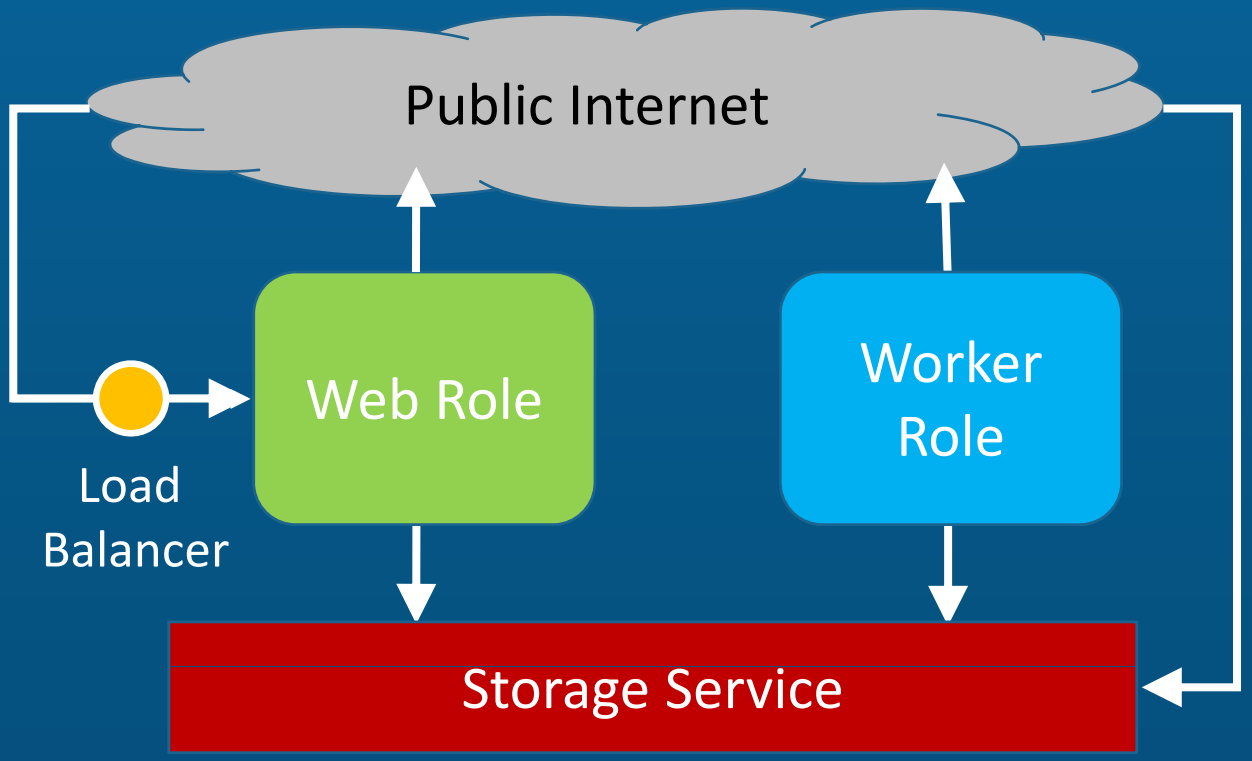
Key Mapping


Storage Unit
Keys Bucketized into Regions
Consistency Unit
Region Name Hashed into Region Id
Failover Unit
ID Ranges mapped to Nodes
Region
Key
"Cust1"
"Cust2"
"Cust33"
"ToyRegion"
"Toy101"
"ToyRegion"
"Toy102"
"BoxRegion"
"Box101"
Region (Name)
Default Region 1
Default Region 2
...
Default Region 256
ToyRegion
BoxRegion
Partition (Range of Ids)
0 – 1000
1001 - 2000
...
...
..
xxx - Maxint
Velocity Service
Velocity Service
Velocity Service

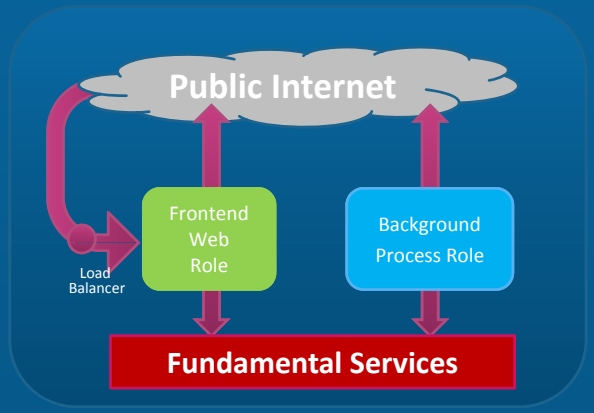
マルレク2008

Windows Azure Service Model

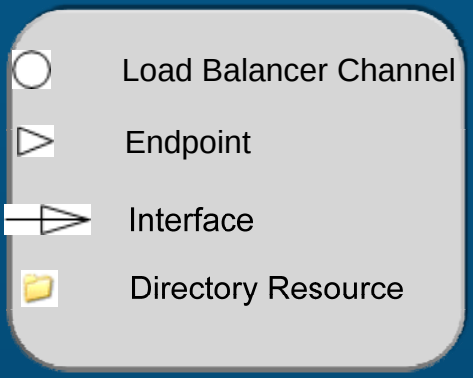
AzureのWebアプリの標準的な構成

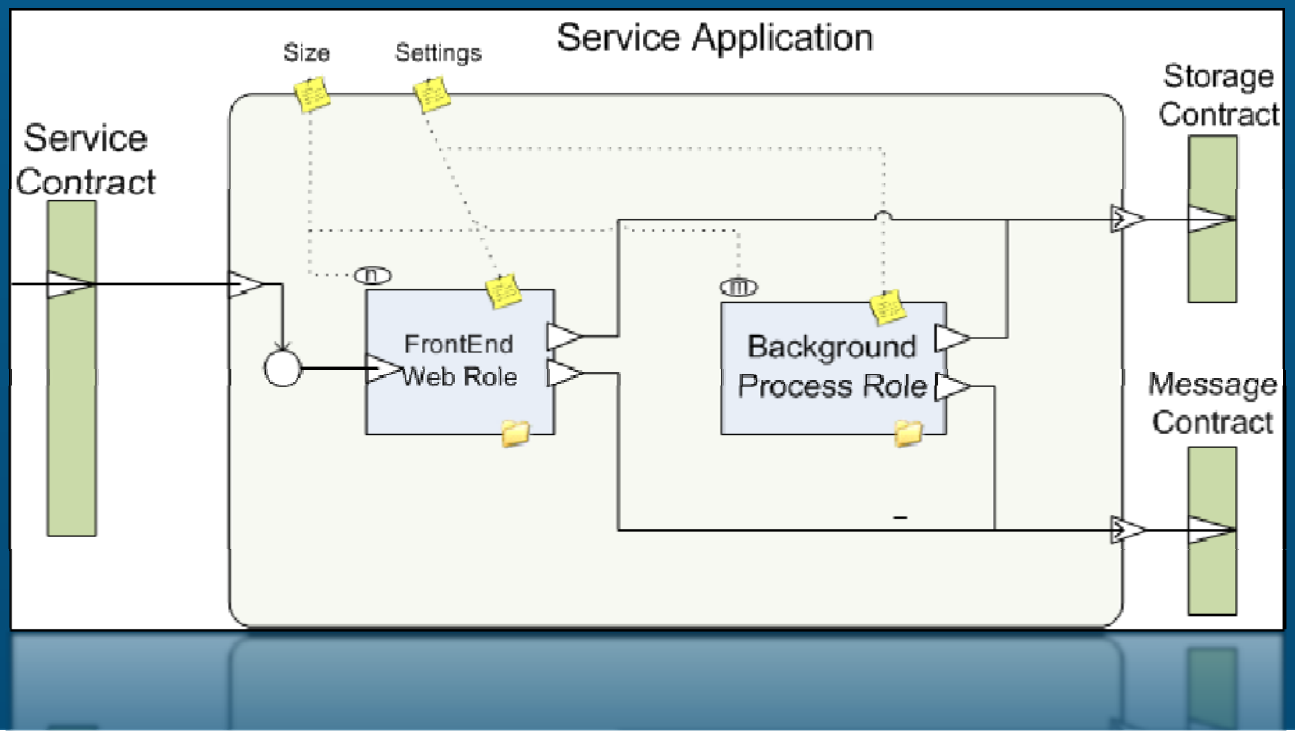


Service Model
単純なサービスの構造をモデリングする



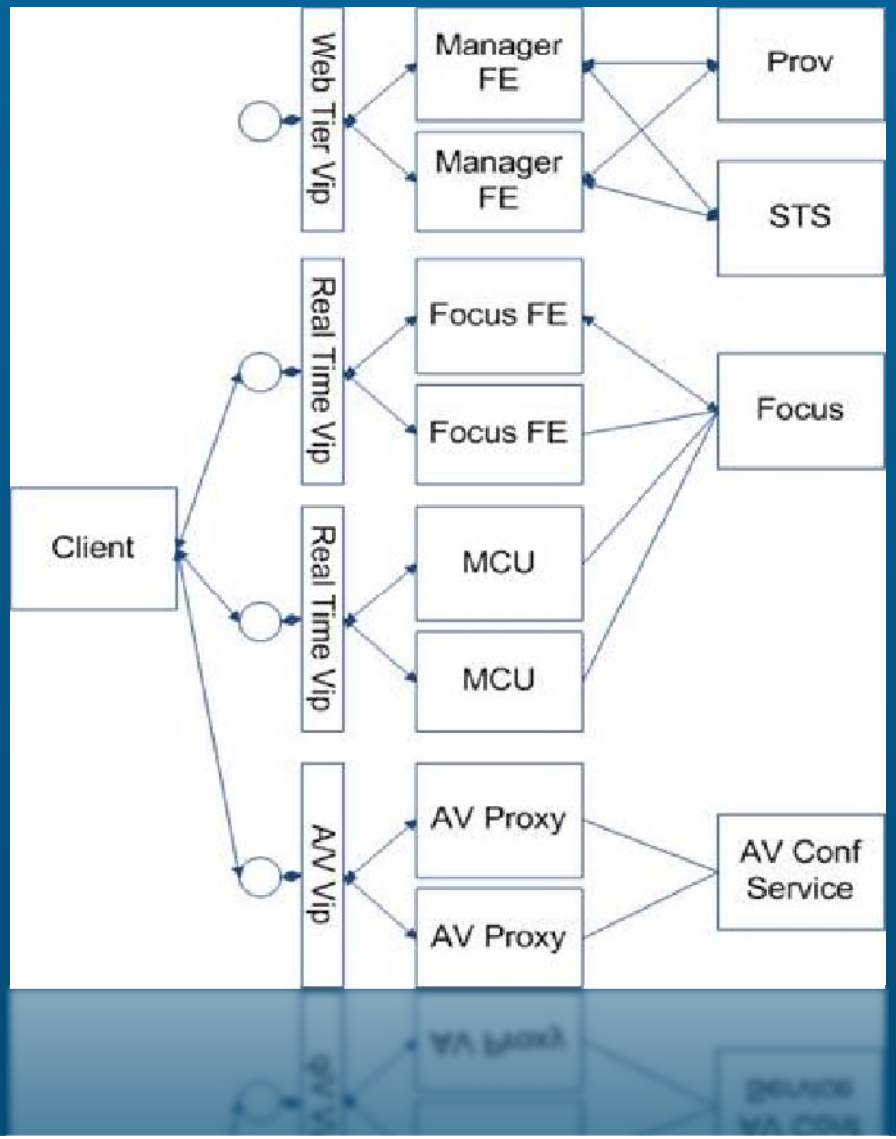
テンプレートは自動的に
サービスモデルにマップされる



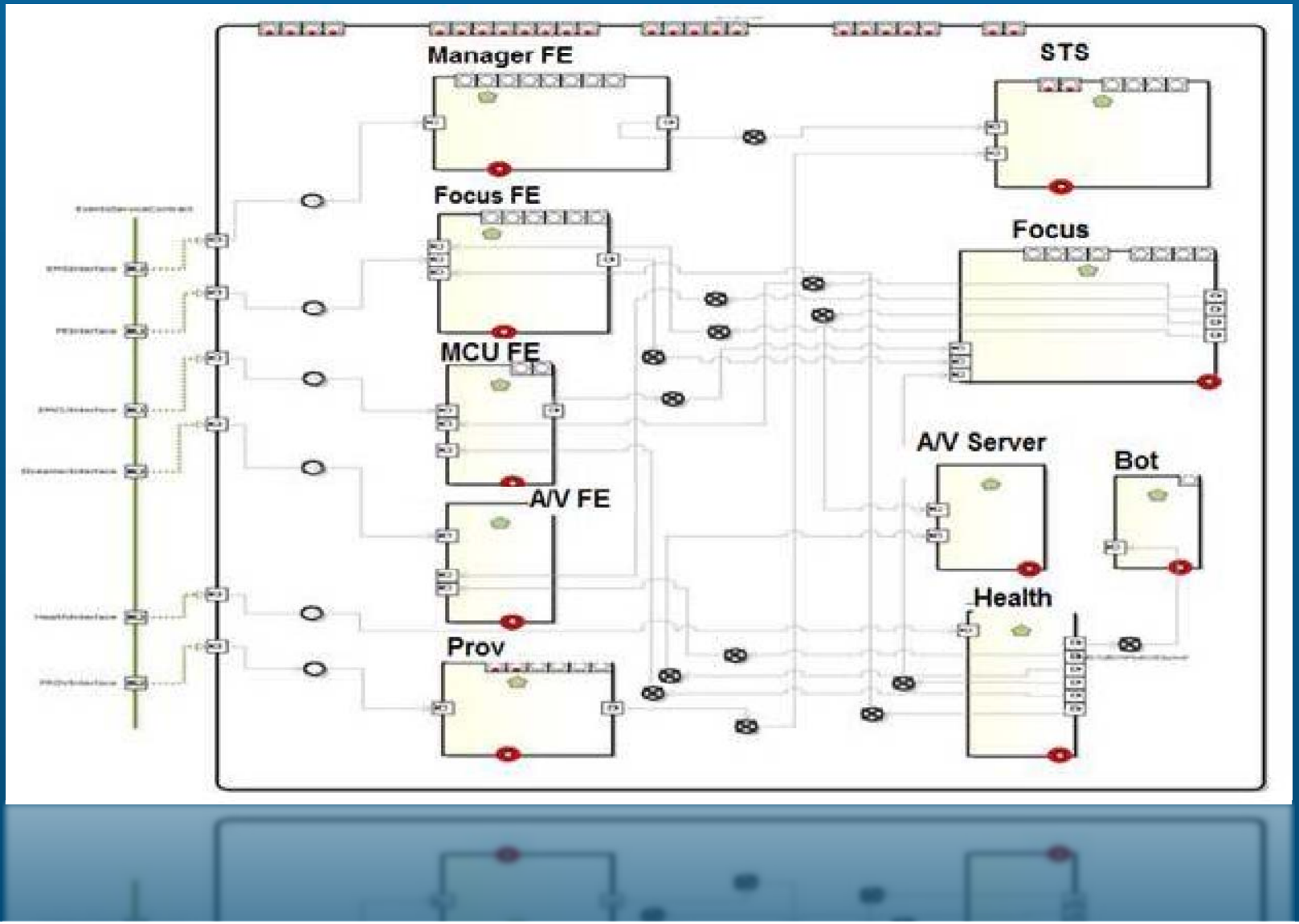


Service Model
複雑なサービスの構造をモデリングする

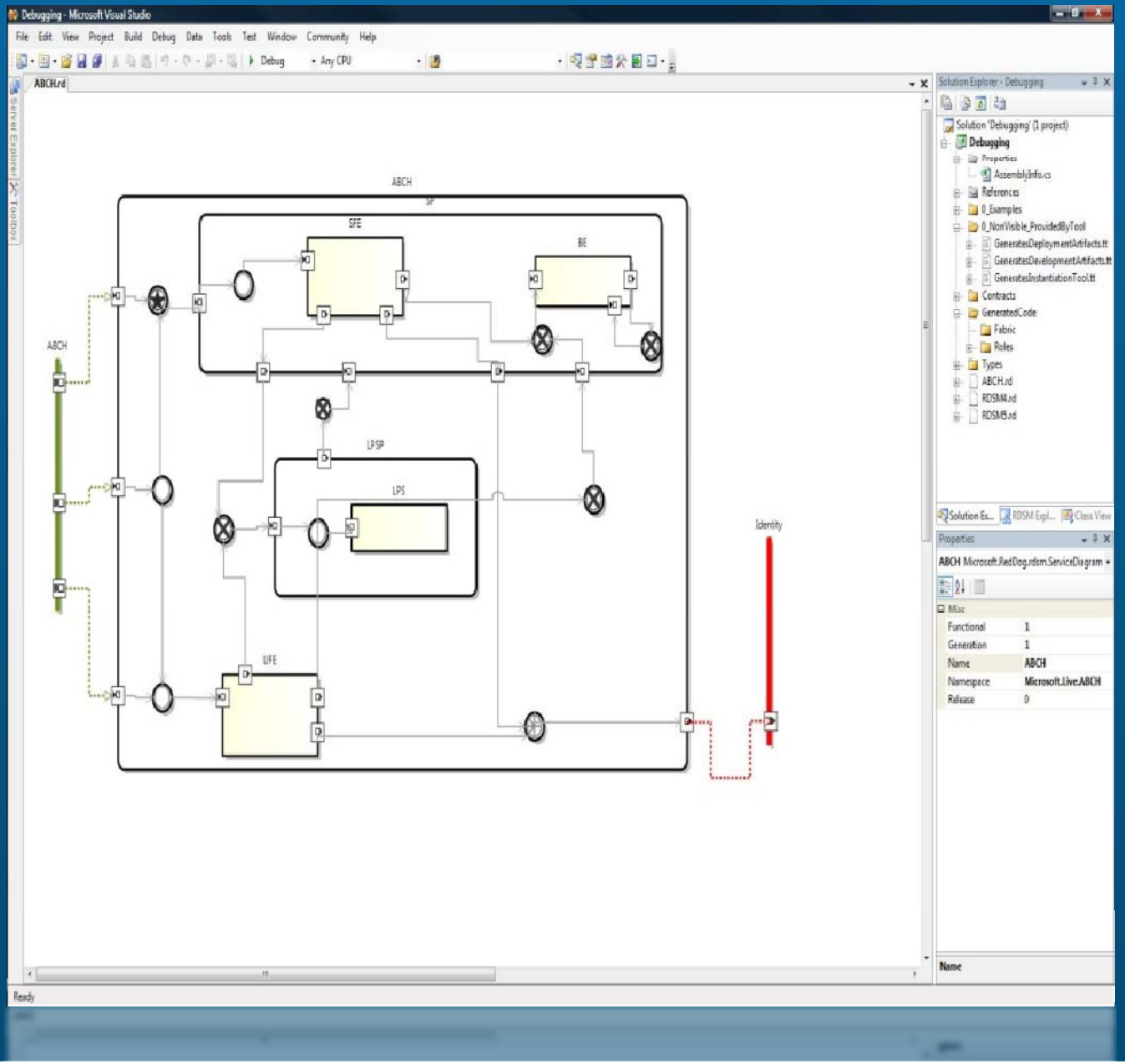
- 高精細ビデオ会議システムの例
  - 高いパフォーマンスと信頼性が要求される
  - 自動的にスケール調整が行われる必要がある



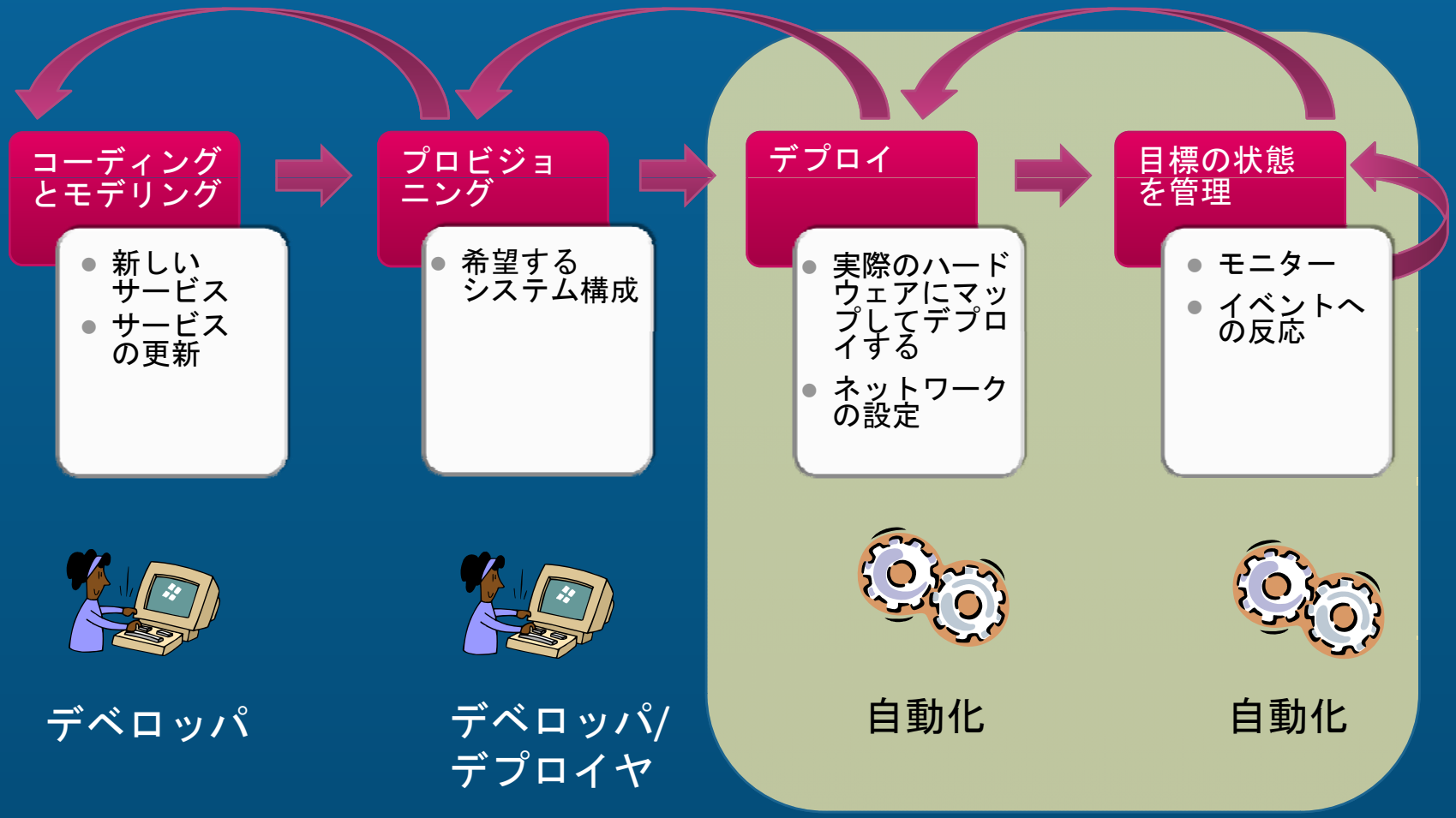
Service Model
ビデオ会議サービスの例

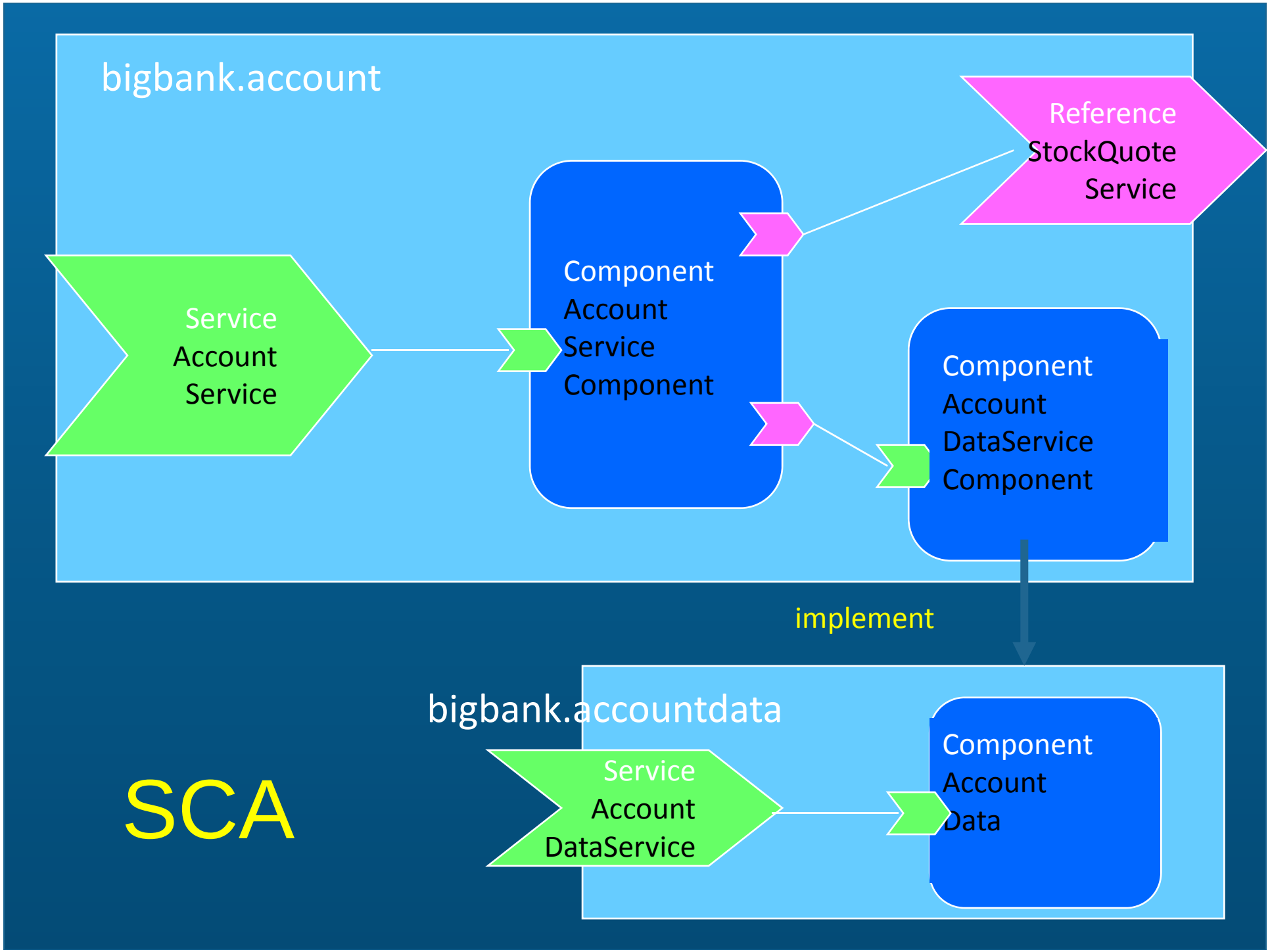


Visual StudioでのService Modelの編集

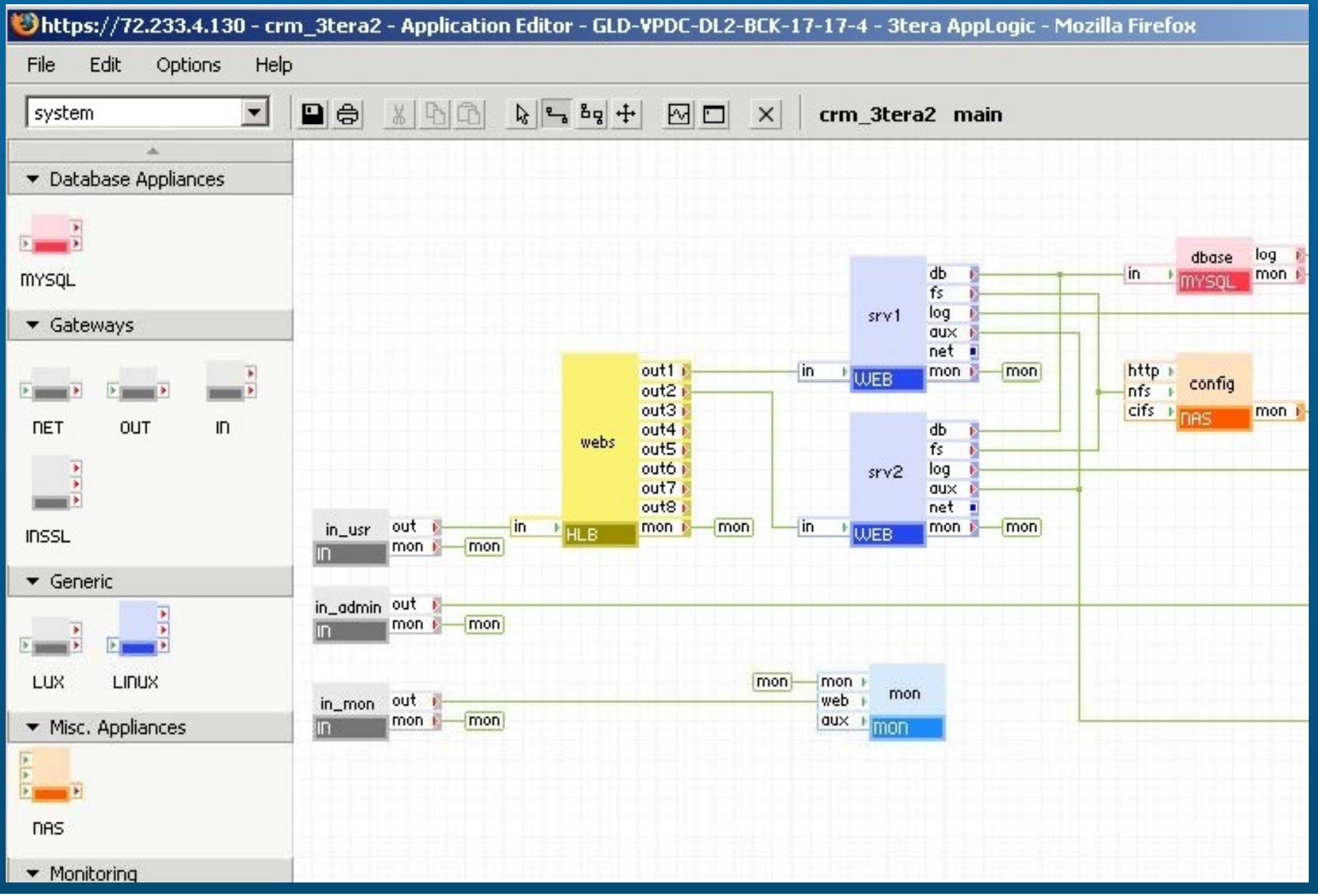


Windows Azure Service Lifecycle
可能な限りライフサイクルを自動化する




bigbank.account
Reference
StockQuote
Service
Service
Account
Service
Component
Component
Account
Service
Component
Component
Account
DataService
Component
implement
bigbank.accountdata
Service
Account
DataService
Component
Account
Data
SCA

AppLogic - Grid Operating System for Web Applications



Service Model が自動化を可能にする

- サービスを分散したエンティティからなるものとして記述する
  - サービス・デベロッパが書き
  - サービス・デプロイヤがコンフィグする
- サービスの論理的な記述を行う
  - 同じモデルが、テスト段階でも実用段階でも利用される
  - デプロイ時に、実際のハードウェアにマップされる
- 強力な、コンポジションを記述する宣言型の言語
  - 単純なものから非常に複雑なサービスまで記述できる

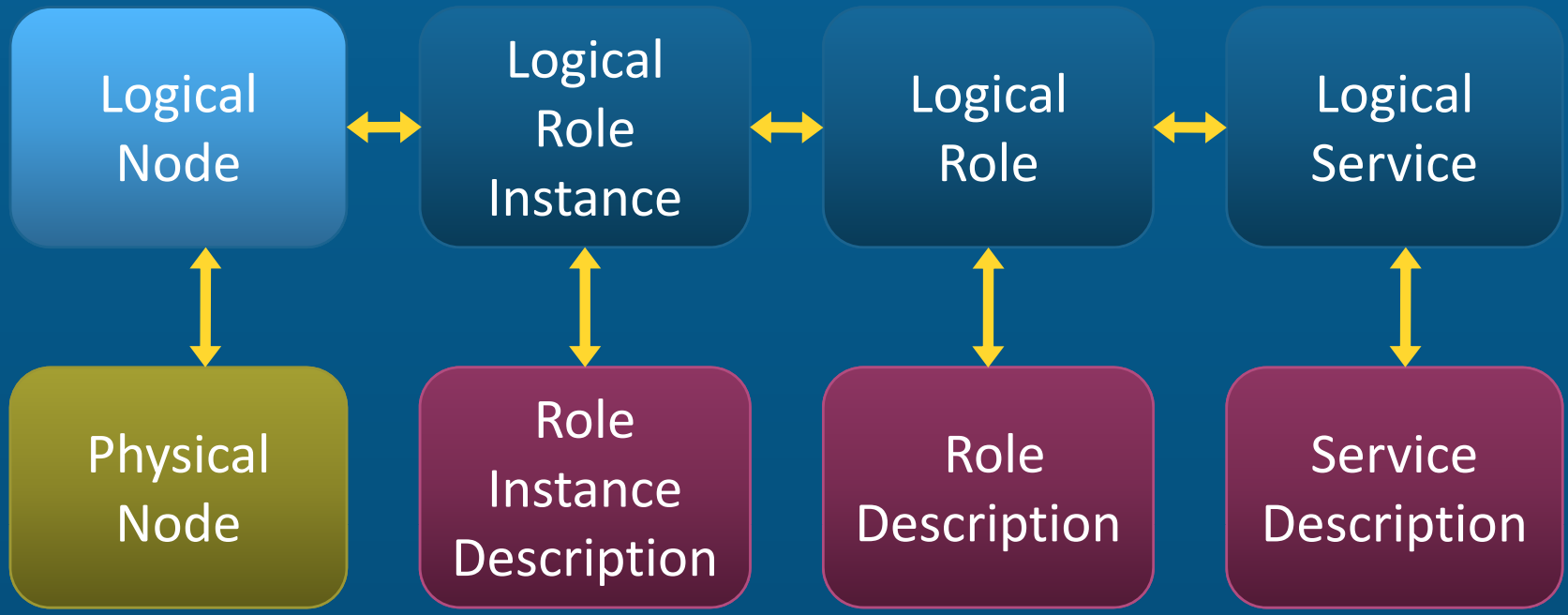
Service Modelの要素達

- Service
  - Role, GroupとChannelの集合
- Role
  - プログラム、実行されるエンティティ
- Group
  - 他のGroup,Role,とChannelの集合
- Endpoint
  - Roleによってエキスポートされる通信用の端点
- Channel
  - 論理的なロードバランサとスイッチ
- Interface
  - Serviceで用いられる
- Configuration settings
  - デベロッパの設定
  - システムの設定

Windows Azure Serviceのライフサイクル

- リソースの割り当て
  - ノードは、Service Modelで記述された制約に基づいて選ばれる
    - Fault domains, update domains, resource utilization, hosting environment,など
  - VIPs/LBは、モデルのそれぞれの外部インターフェースの記述ごとに確保される
- プロビジョニング
  - 割り当てられたハードウェアには、新しいゴールの状態が与えられる
  - FC は、ハードウェアをゴール状態に持っていく
- 更新
  - FC は、実行中のサービスを更新できる
- サービスを良好な状態に保つ
  - ソフトウェア・エラーはハンドルされなければならない
  - ハードウェアのエラーは、いつか起きるもの
  - ロギングのインフラ

Key FC Data Structures

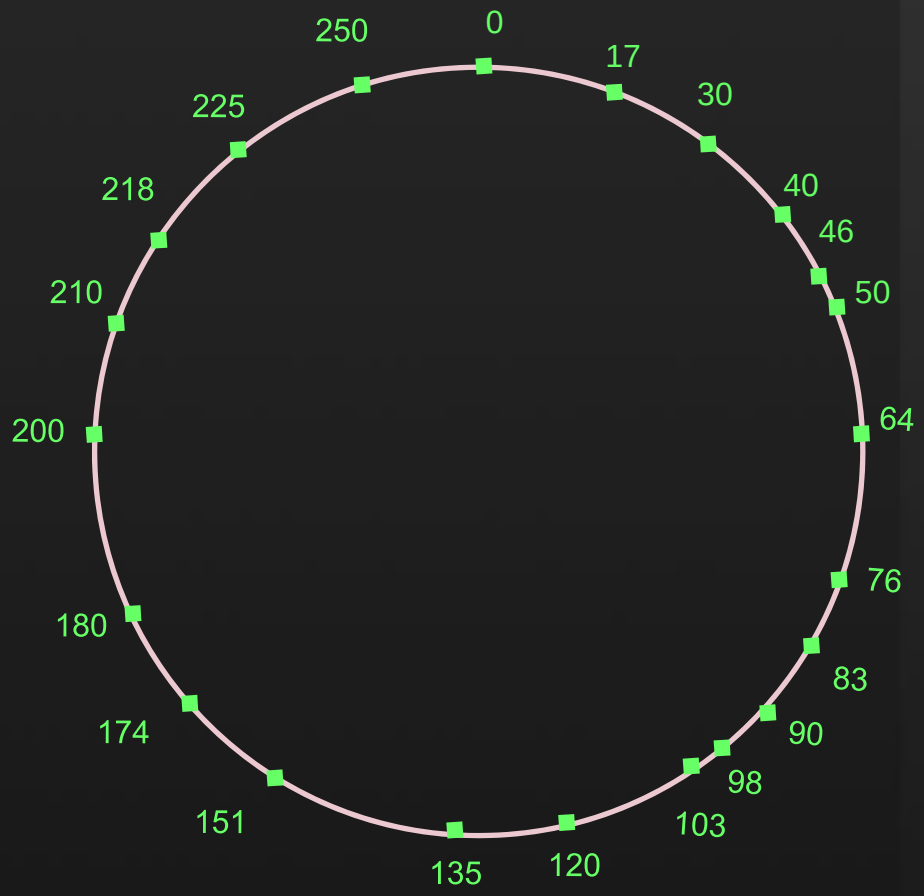


マルレク2008

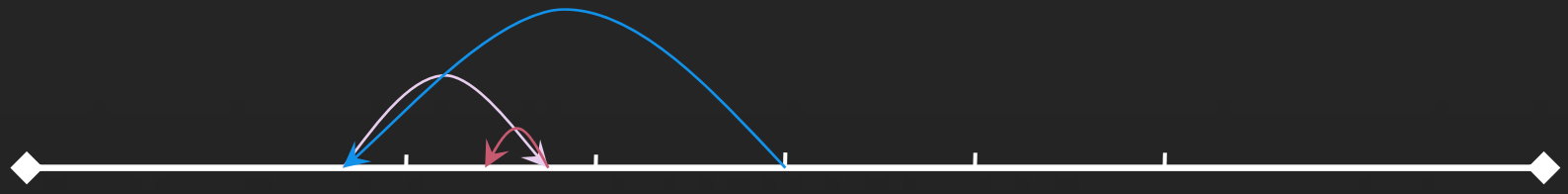
Fabric Conroller Routing and Data Overlay

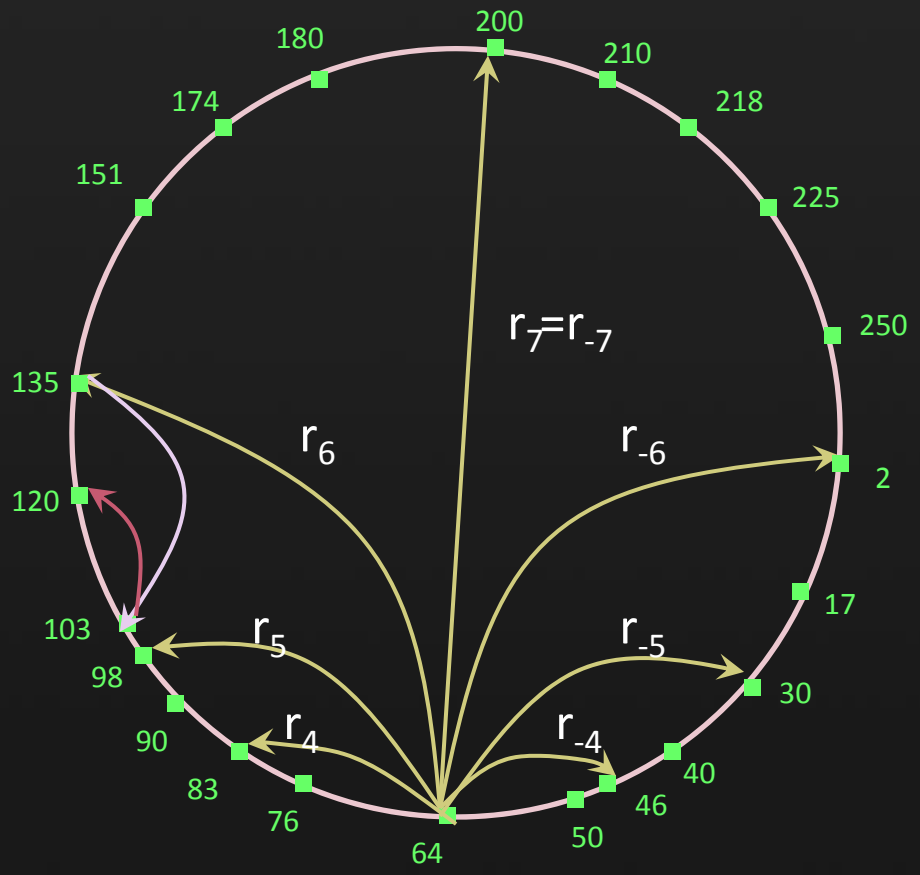
リング構造

- 全てのノードはユニークなIDを与えられる。（典型的には、128-bit か160-bitの数値である)
- アクティブなノードは、順序を持つ双方向リンクで、しっかり結び付けられ、自分たちで、その形を維持する。
- 最高位のIDと最低位のIDを持つノードは、相互にリンクする
- こうして、アクティブなノードたちはリングの形をとるようになる。
- リングは、最初の一つのノードからブートストラップされる。



ルーティング・プロトコル
分散バイナリーハッシュ



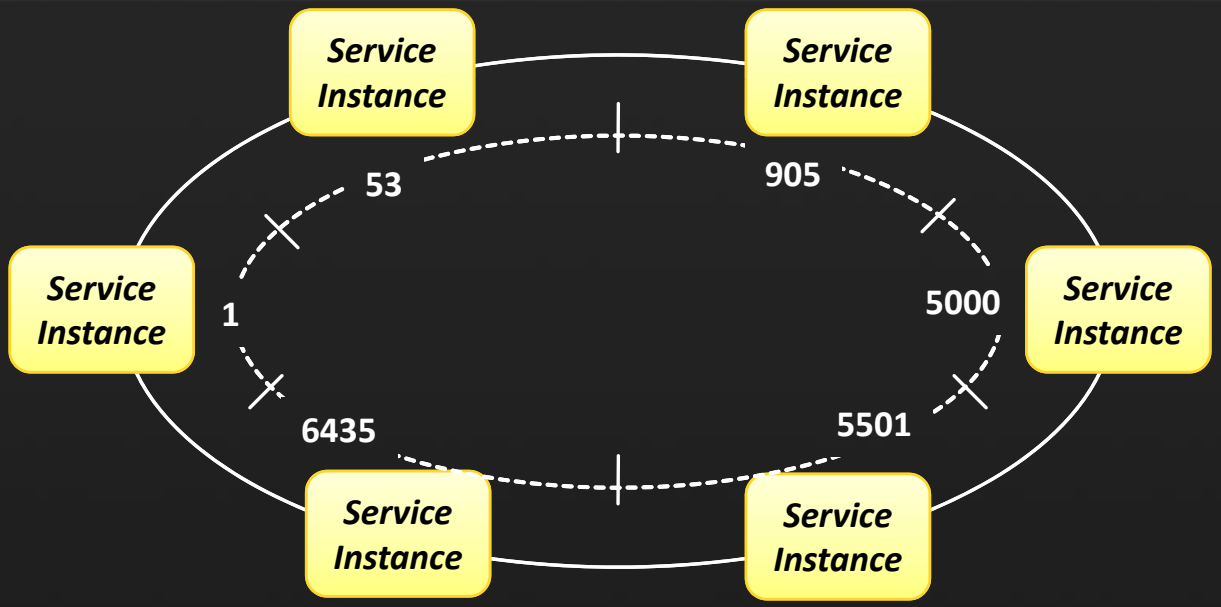


ノード64のルーティングテーブル:
Successor = 76
Predecessor = 50
Neighborhood = (83, 76, 50, 46)
Routing nodes = (200, 2, 30, 46, 50,
64, 64, 64, 64, 64,
83, 98, 135, 200)

$r_{\pm n} = a \pm 2^n$

ルーティングは、分散ハッシュテーブル（DHT）を作る基礎となる。

ルーティング・オバーレ Partitioning

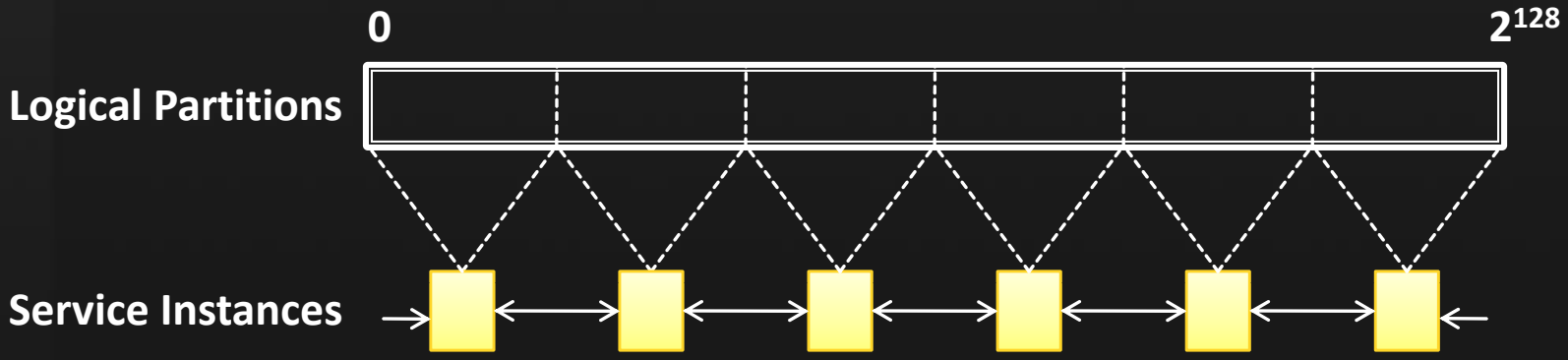


構造化された
P2Pネットワーク

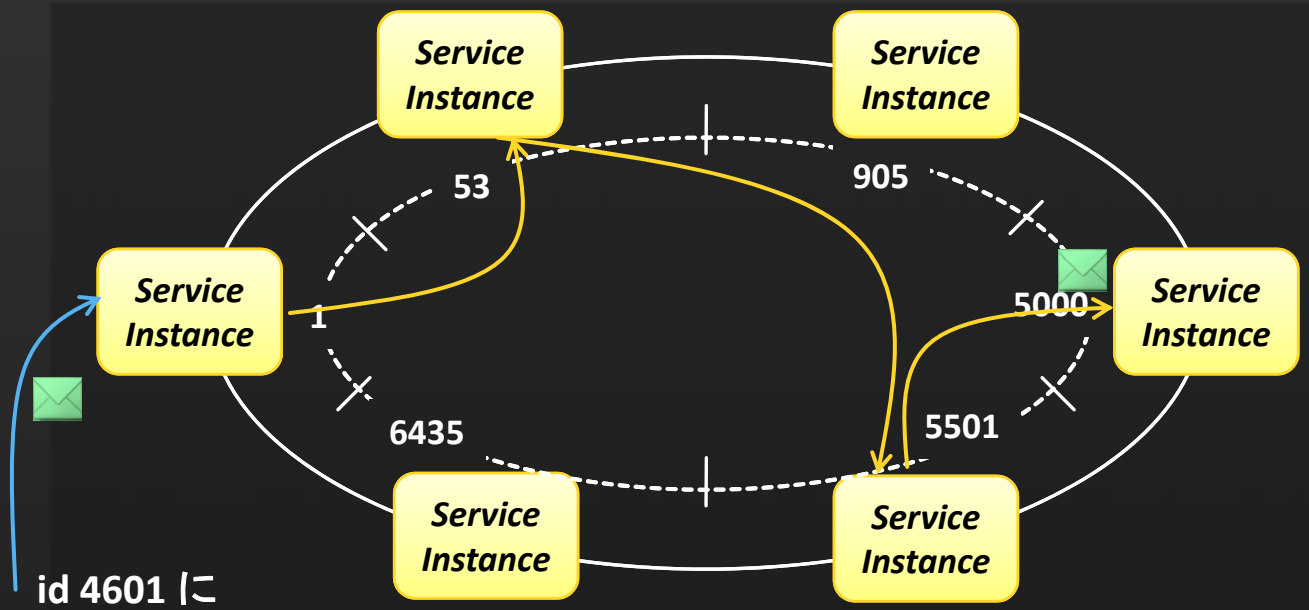
リング・トポロジー

動的で特定の中心を
持たないメンバー
シップ構造

連続した仮想アドレ
ス空間



ルーティング・オバーレ – Messaging



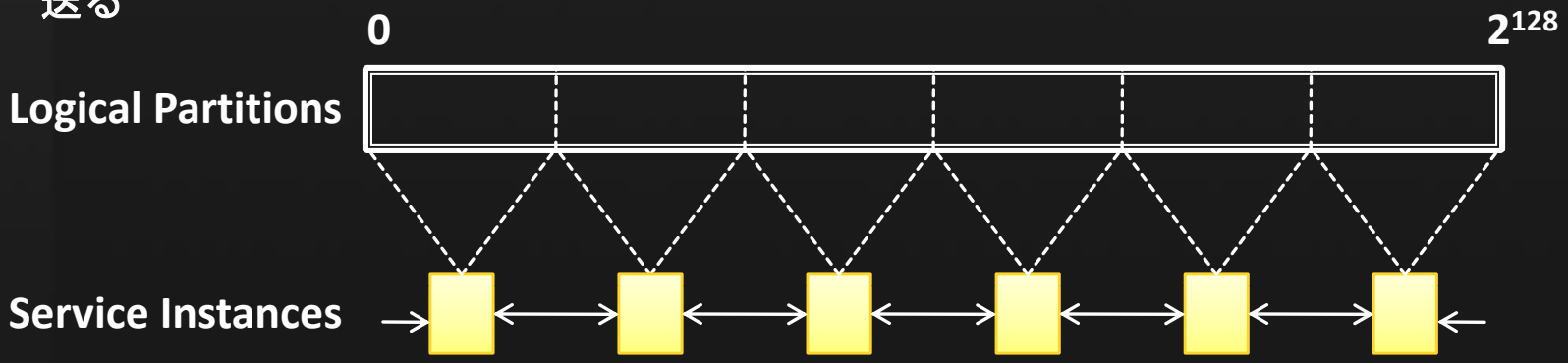
id 4601 にメッセージを送る

WCF トランスポートチャンネルを使う

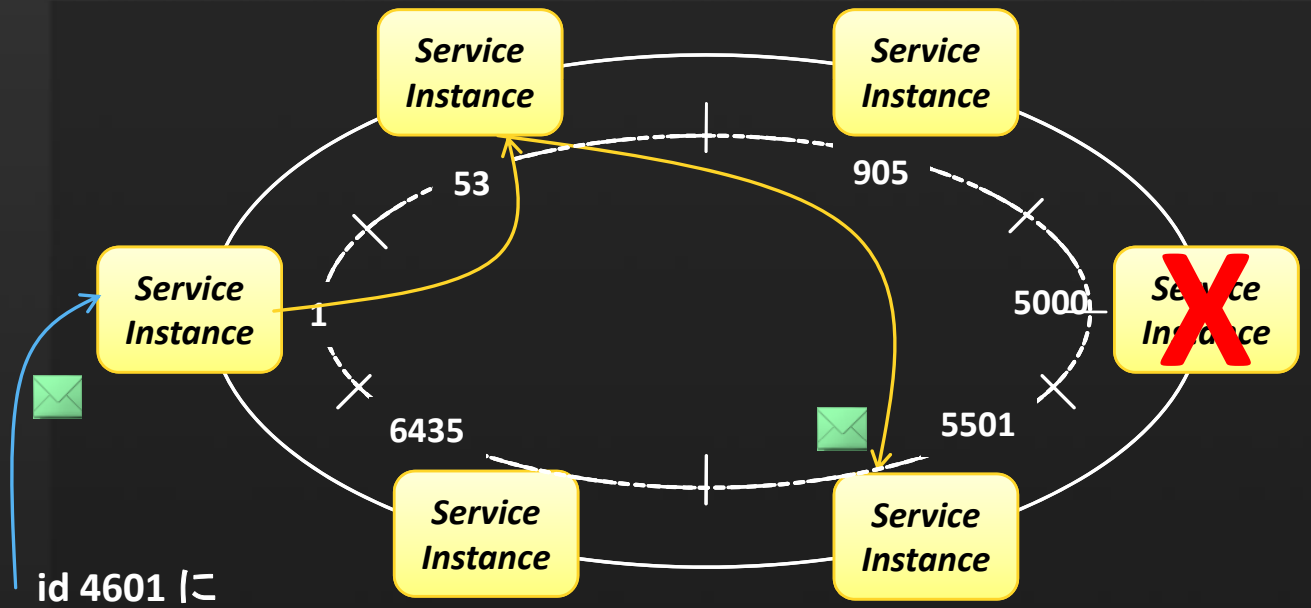
WCF TCP チャンネル

アドレスは、partition IDである

ルーティングのホップ数は、 O(Log(n))



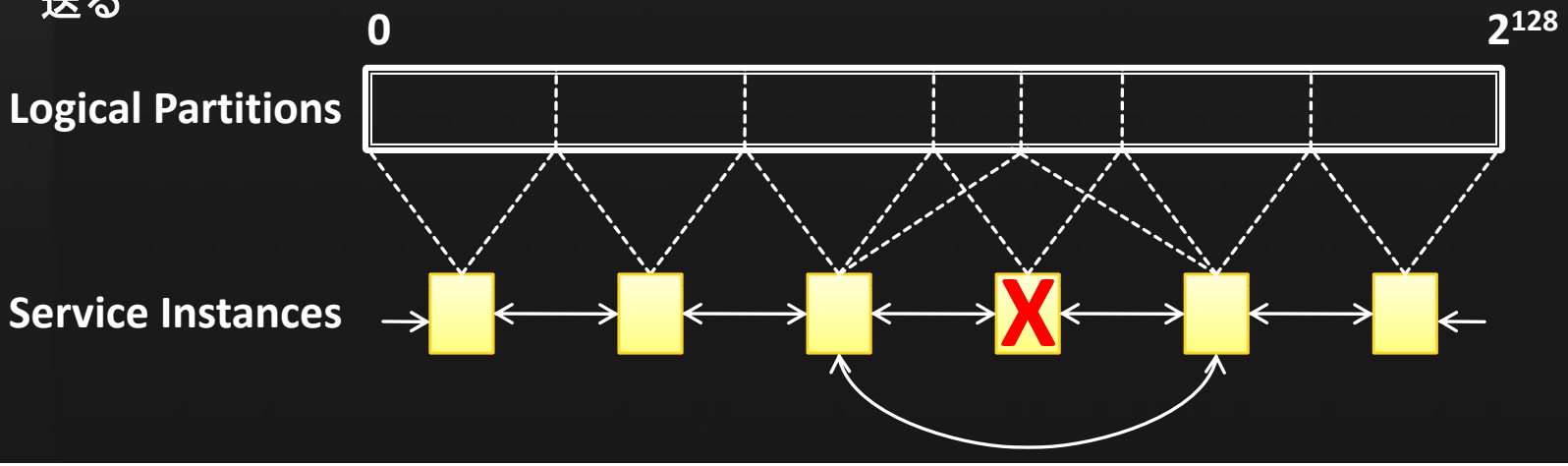
ルーティング・オバーレ – Availability



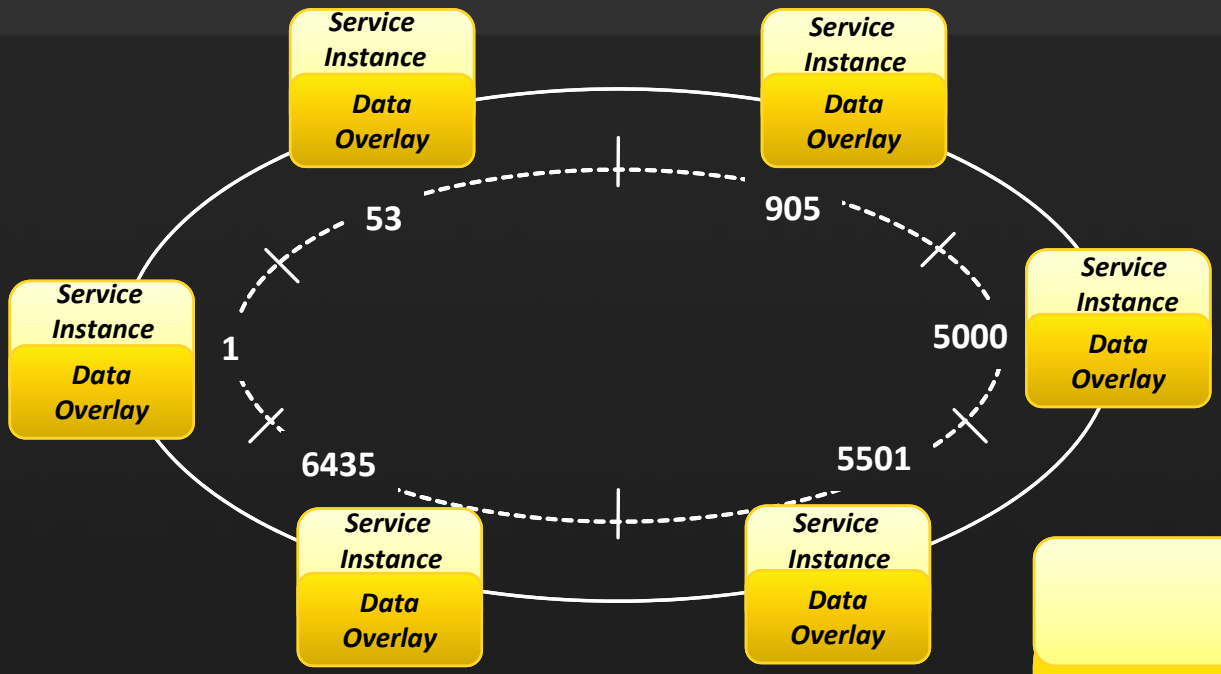
自己モニターリングと自己修復機能

柔軟なホスト環境

Partitionsは、常にアドレス可能である



データ・オバーレ



データ・オバーレーは、ルーティング・オバーレーの上に作られる

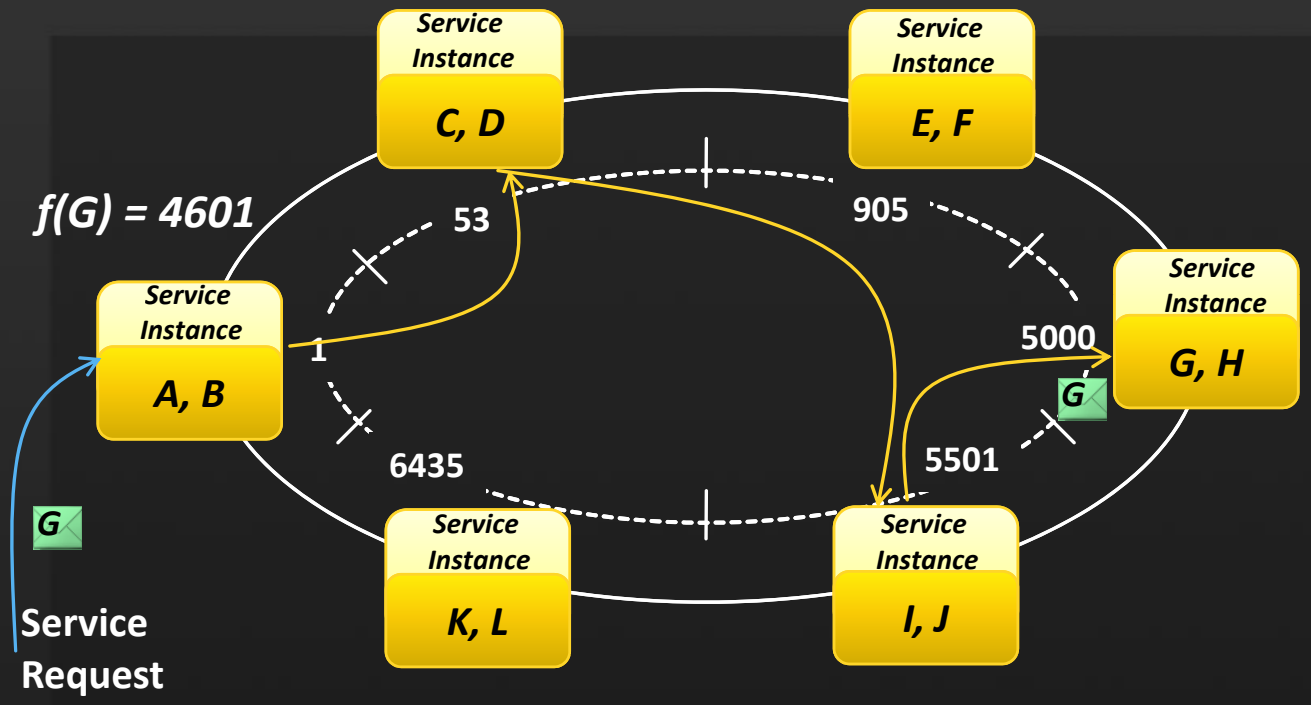
エンティティは、Partitionに格納される

Node 5000

Data Overlay

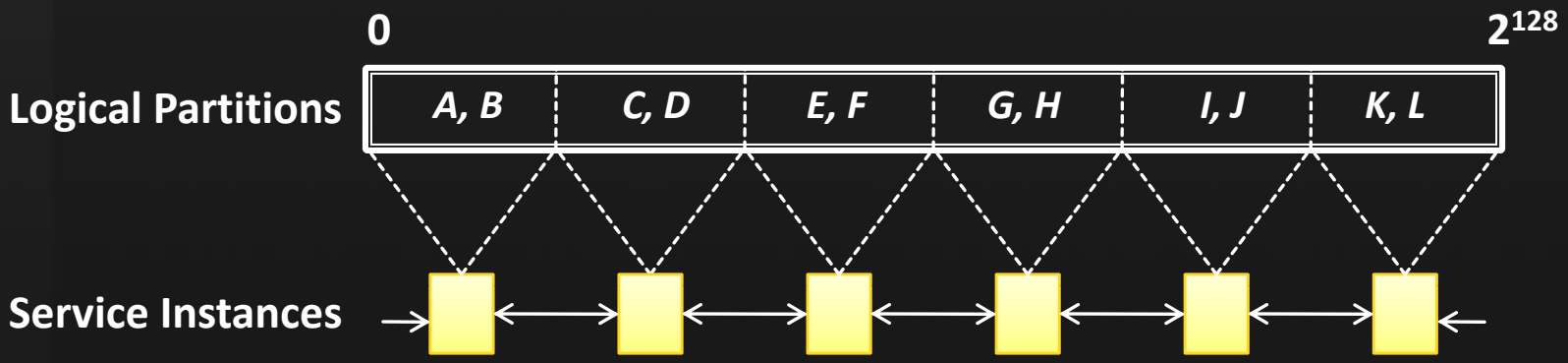
Partition Id	Name	Contents
4601	WF #12	<Message Q for #12>
2953	WF #67	<Message Q for #67>
…		…

データ・オバーレ – Partitioning

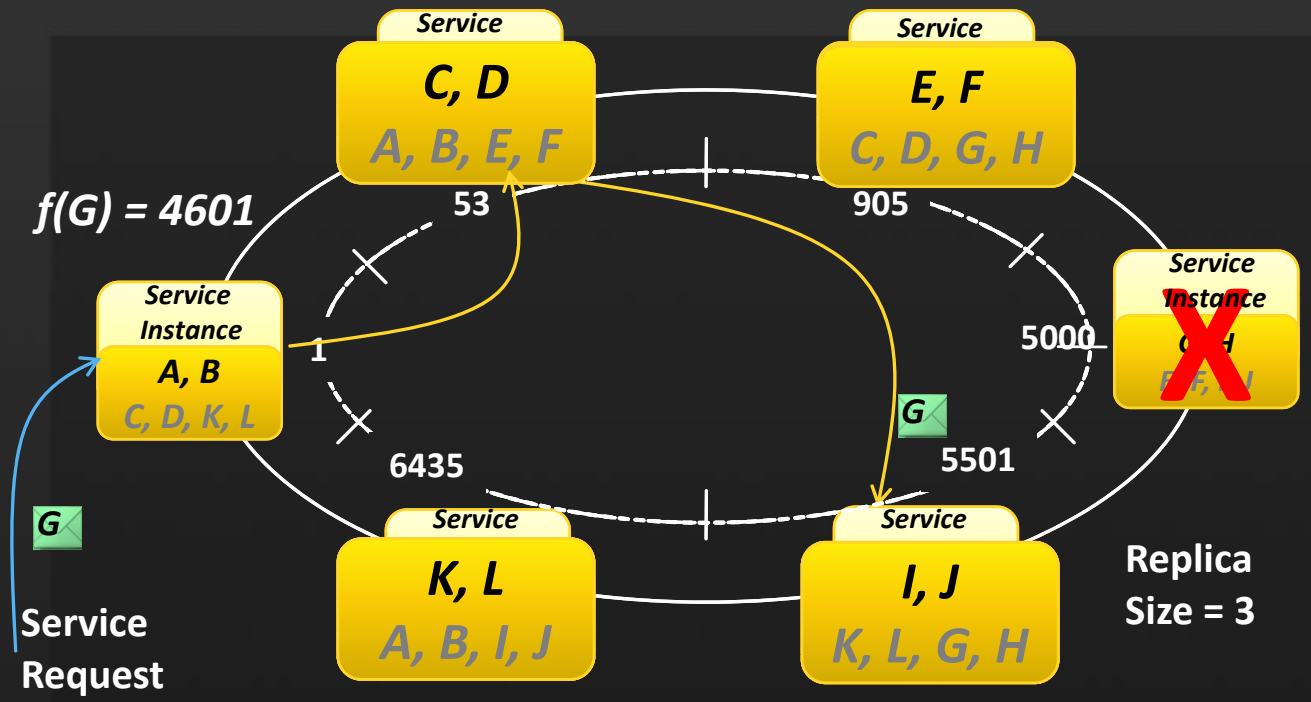


サービスは、エンティティからPartitionへのマッピングを持っている

Data Overlay の操作は、ノード上でローカルに行われる



データ・オバーレ – Availability



可用性を高めるためにノードに複数のレプリカを置く

プライマリが壊れたら、自動的にセカンダリへのフェールオーバが行われる

可用性と無矛盾性は、システムデザイン時の指針になっている